ACCURATE EDGE DETECTION AND GAUGING WITH CCD-CAMERAS.

by Arets J.M.A.
Acknowledgement.

I’m sincerely grateful to Prof. Ir. F.J. Kylstra and Ir. A.C.M. Gieles for the opportunity they gave me to carry out my final practical training period within the Philips Centre For manufacturing Technology in Eindhoven. Especially I would like to thank Ing. A.J.M. van Lier and Ir. R.G. van Vliet, whose advises and guidance were very valuable for bringing the project to a successful end. Finally I want to thank all members of the group Industrial Measurement and Inspection for their help and for making my stay in the group an enjoyable one.
## CONTENTS

1. Introduction 1

2. The digital imaging processing system 2
   2.1. Introduction 2
   2.2. The components 2
       2.2.1. The CCD-matrix camera 2
       2.2.2. The frame grabber 5

3. Image acquisition 6
   3.1. Image forming 6
   3.2. Contrast 8
   3.3. Errors in imaging 10
       3.3.1. Spherical aberration 10
       3.3.2. Coma 10
       3.3.3. Astigmatism and field curvature 11
       3.3.4. Distortion 12
       3.3.5. Chromatic aberration 12

4. Edge detection 13
   4.1. Introduction 13
   4.2. Edge profile 13
   4.3. Thresholding 15
   4.4. Gradient 15
   4.5. Extending the gradient conditions 17

5. Rulers 18
   5.1. Introduction 18
   5.2. The ruler algorithm 18
   5.3. Realization of rulers for omnidirectional scanning 20
   5.4. Results 23

6. Edgels 25
   6.1. Introduction 25
   6.2. The edgel algorithm 25
   6.3. Investigation on the threshold 26

7. 2-Dimensional rulers 28

8. Application 1: Bar code technology 30
   8.1 Introduction 30
   8.2. Bar code symbology 30

9. Reading bar codes 32
   9.1. Optics and illumination 32
   9.2. Detection and decoding 32
   9.3. Extending the edge conditions 34
   9.4. Results 37

10. Application 2: The shaverhead 40
11. Optics and illumination ........................................ 41
  11.1. Image acquisition ........................................ 41
  11.2. Adjusting the light source .............................. 43

12. Centre, radius and image resolution .......................... 44

13. Gap width gauging ............................................ 47
  13.1. Method of measurement .................................. 47
  13.2. Results for rulers and edgels ........................ 50
  13.3. Defining the scan position ............................. 55

14. Total inspection ............................................... 57

15. Correcting distortion ........................................ 58
  15.1. Introduction ............................................. 58
  15.2. Measuring the distortion .............................. 58
  15.3. Correction ................................................ 62
  15.4. Results .................................................... 64

16. Conclusions and recommendations .............................. 67

Literature ..................................................................... 69

Appendix 1 .................................................................. 70
Appendix 2 .................................................................. 79
Appendix 3 .................................................................. 81
Appendix 4 .................................................................. 82
Appendix 5 .................................................................. 89
Appendix 6 .................................................................. 102
Appendix 7 .................................................................. 103
1. INTRODUCTION.

Machine vision is concerned with the extraction of information from a digital image. Finding edges and obtaining geometric information of objects are fundamental stages of image analysis. Studying and developing accurate optical edge detection and measurements techniques for CCD-cameras, regardless of the image distortion that can appear in the imaging system, is the research objective of a final project of a college career. The investigation is divided into three tasks:

- accurate edge detection which is implicated in a barcode decoding algorithm.
- accurate gauging which is implicated in a feasibility study for optical inspection of the gaps of shaverheads.
- correcting digital image distortion to improve the performance of the techniques mentioned above.

Chapter 2 and 3 describe the digital image processing system that has been used and the limitations that has to be regarded for optical inspection.

To detect edges, tools have to be developed to registrate their presence and define their exact locations with sub-pixel accuracy, which is discussed in chapter 4, 5, 6 and 7.

After explaining barcode technology in chapter 8, chapter 9 describes the image acquisition and the improvement of the edge detection methods for barcodes to extract as much as possible information from an image of a barcode symbol.

Chapter 10 deals with the shaverheads geometry and chapter 11, 12, 13 and 14 discuss the methods and algorithms used to inspect the gaps of the shaverhead. To detect the best inspection method, the subpixel methods of chapter 4, 5, 6 and 7 are compared.

In chapter 15 a method to measure and correct image distortion as a function of the location in the image is described which is applied on the optical system used for gap width inspection.

The report ends with some conclusions and recommendations for further investigation in chapter 16.
2. THE DIGITAL IMAGE PROCESSING SYSTEM.

2.1. Introduction.

The fundamental components of the image processing system which has been used are shown in figure 2.1. The image originates from a Philips CCD -matrix camera. The analog signal produced by this camera is digitized and every sample or pixel is converted to a digital value which is stored in a frame memory. The stored image can be displayed on an external monitor after transforming the pixel values back into an analog signal. The digitizing, storing and displaying is performed by a frame grabber (PCVISIONplus), which, placed into a personal computer, allows complex digital image processing functions to be performed.

Fig. 2.1. The image processing system.

2.2. The components.

2.2.1. The CCD-matrix camera.

Solid state arrays have become the most important type of image sensors in industrial instrumentation. The advent of silicon integrated circuit technology enabled the manufacture of a large number of photosensitive elements on one silicon chip. CCD (charge coupled device) - operation is based on the principle of optical generation, storing and transport of electrical charges within silicon [1]. Incident photons can knock electrons out of the atoms in the crystal structure. These electrons and the positive 'holes' they leave behind become temporary mobile within
the silicon and travel a certain distance before being destroyed by recombination. The mobile charges are collected by an electric field, which can be established in two ways. Either by a diode junction formed between two different types of silicon, namely n-type and p-type, or by placing an electrode on top of the silicon surface with a certain voltage. This electrode is separated from the silicon by a layer of insulating silicon dioxide. A CCD-sensor consists of a large number of such sensitive elements, ranged in a one or two dimensional array. In figure 2.2 such an element is formed by an area of p-type silicon, bounded by a p+ -channelstop, with a positive voltage on the electrode.

Fig. 2.2. Charge storing in a MOS-capacitor.

Because of the positive potential on the electrode the majority carriers (in this case the positive holes) will travel downwards in the silicon while the minority carriers will be pulled up to the surface. In this situation the yield of recombination is very small. Optically generated charges also split up and practically do not recombine. The amount of charges, generated in this way is proportional to the intensity of light falling on that particular part of the chip.

The readout of CCD’s takes place by transporting the generated charges quickly through the silicon chip by means of CCD - shiftregisters. Therefor the electrodes on the surface are divided into three groups P1, P2 and P3, see figure 2.3. By applying an out of phase clock signal to these three sets, the electric field below the surface can be arranged as potential wells which remain separated and move through the surface carrying the charge packets.
At $t=t_1$ only $P_1$ is positive and optically generated electrons are collected below $P_1$. At $t_2$ both $P_1$ and $P_2$ are positive so the charges are spread below $P_1$ and $P_2$. At $t_3$ only $P_2$ is positive and all the electrons have been transported to the area below $P_2$. In this way the charge packets can be read out. The camera used is of the frame transfer type in which the sensor is divided into an imaging and a storage area, see figure 2.4. During exposure to light, charge generation takes place in the imaging area, while steady states are established on the electrodes. At the end of the exposure time the charge packets are clocked at a high rate into the storage area, which is shielded from light. During the next exposure time the charges are read out. The advantages of the frame transfer CCD-camera are the high pixel resolution, the relative low noise level and the low price.
2.2.2. The frame grabber.

The PCVISIONplus frame grabber is the heart of the digital image processing system. It digitizes the incoming video signal to an eight bit value at a rate of 30 frames per second. The video signal contains the analog video information as well as the timing information for digitizing. The resulting values are stored in a $512 \times 512 \times 8$ bit frame memory. Every pixel in the frame memory, which can be addressed by its video coordinates, represents one out of 256 possible intensity levels. Those values are converted back into an analog signal by the display logic, so the image can be viewed on a video monitor. Three output channels are provided for color display and every channel contains eight programmable LUT's for transformation of the 256 intensity levels. The frame grabber is directly compatible with the bus structure of an IBM (compatible) personal computer. It is supplied with a software package ITEXPCplus that includes several basic digital image processing functions. These functions are used to create more specific algorithms, which can be developed with the help of a C-interpreter namely Cterp. Since the pixels are directly addressable by their coordinates in the frame memory, complex image processing operations can be performed at pixel level.
3. IMAGE ACQUISITION.

3.1. Image forming.

To realize the limitations of optical inspection with CCD-cameras, the way in which an image is formed has to be considered. When an object is imaged under incoherent illumination by an optical system as in figure 3.1, the intensity signal is represented as

\[
o(\alpha, \beta) = \text{intensity signal in object plane} \\
i(x, y) = \text{intensity signal in image plane}
\]

in which \((\alpha, \beta)\) and \((x, y)\) are the coordinates in object cq. image plane [2, 3].

![Image forming diagram](image)

Fig 3.1. Image forming.

In the ideal case the signal in an image point is unambiguously determined by the corresponding object point. Because of diffraction by the lens system, the signal of an arbitrary object point is influencing the intensity spread of a local area around the corresponding image point. This is expressed in the so called spreadfunction,

\[
s(x-\alpha, y-\beta),
\]

which governs the intensity spread \(i(x, y)\) as a function of \(o(\alpha, \beta)\)

\[
i(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} s(x-\alpha, y-\beta) o(\alpha, \beta) \, d\alpha \, d\beta \quad (3-1)
\]
Every 2-dimensional intensity distribution can be represented as a combination of sinusoidal intensity profiles with different frequencies, like grids as shown in figure 3.2.

This results in

\[ o(\alpha,\beta) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} O(W) \cdot e^{2\pi i (r \cdot W)} dW_{x} dW_{y} \]  \hspace{1cm} (3-2)

\[ i(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(W) \cdot e^{2\pi i (r \cdot W)} dW_{x} dW_{y} \]  \hspace{1cm} (3-3)

and

\[ s(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} S(W) \cdot e^{2\pi i (r \cdot W)} dW_{x} dW_{y} \]  \hspace{1cm} (3-4)

in which \( W \) is a vector with spectral components \( W_{x} \) \((W_x)\) in \( \alpha \) - (x) direction and \( W_{y} \) \((W_y)\) in \( \beta \) - (y) direction. \( I(W) \) \( O(W) \) is the amplitude of the component with spatial frequency \( W \) in image \( O(W) \) object space and \( r \) is the vector \( (\alpha,\beta) \) \((x,y)\). Combining (3-1), (3-2), (3-3) and (3-4) provides the following expression for \( S(W) \), which is normalized for \( W_{x}=W_{y}=0 \):

\[ S(w) = \frac{I(w)}{O(w)} = |S(w)| e^{i\theta(w)} \]
This expression is called the optical transfer function (OTF). If $|O(W)|$ is the amplitude with which the component with frequency $W$ takes part in the construction of the object, $|S(W)|$, the modulation transfer function (MTF) is the factor with which the contrast of that grid is reduced in the image.

3.2. Contrast.

As shown, optical imaging can be regarded as signal transfer in which the optical system functions as a frequency filter. An ideal filter does not influence the incoming signal so $S(W)=1$ or $i(x,y)=o(x,y)$ for the whole frequency range. However, because of diffraction by the lens the MTF will fall off to zero if $W$ increases. This can be explained as follows. Two lightspots in object or image space can be distinguished only if the distance between them is larger than the so called Airy disc radius which is

$$r_1 = \frac{0.61 \times \mu}{\sin \alpha_1} \quad \text{for points in the object space}$$

and

$$r_2 = \frac{0.61 \times \mu}{\sin \alpha_2} \quad \text{for points in the image space.}$$

in which $\mu$ is the wavelength of the used light (see figure 3.3).

![Fig 3.3. Diffraction.](image)
If the object distance \( R >> d \) (=diameter of the lens) then \( \sin \alpha = \frac{d}{2f} \) (\( f \) = focal distance) and the minimal distance for two points in the image space is

\[
\frac{1.22 \mu f}{d}.
\]

Starting from an object with an intensity distribution as in figure 3.2, the corresponding image consists of an also sinusoidal intensity distribution with the same spatial frequency. The original contrast \( \tau \) in the object

\[
\tau = \frac{\text{maximum level} - \text{minimum level}}{\text{maximum level} + \text{minimum level}}
\]

is reduced by \(|S(W)|\) and there is also a phase shift \( \theta(W) \) which gives rise to a relative shift of the image. However because of the digitizing process the MTF falls off faster to zero than expected. The digitizing process integrates the incoming light rays of the image points on a pixel area and samples the result. The smallest distinguishable unity in the image can now be seen as a pixel. The profile of the MTF for digitized images compared to the MTF for non-digitized images is shown in figure 3.4. If \( W \) exceeds \( W_n \), the contrast in the image is less than the noise level so no contrast is visible for spatial frequencies above \( W_n \). \( W_n \) is the maximal detectable spatial frequency.

![MTF for discrete imaging](image.png)

**Fig 3.4. MTF for discrete imaging.**
3.3. Errors in imaging.

Optical systems show imperfections in imaging, which also influence the MTF. These errors result in blurred, shifted and distorted images. Some of the most important errors will be discussed [4].

3.3.1. Spherical aberration.

Light rays passing off-centre through the lens will be refracted more strongly than rays passing centrally (see figure 3.5). This causes blurred images. Parallel incoming light rays come to focus at different distances from the lens, which can be solved by giving the lens an aspherical surface.

![Fig. 3.5. Spherical aberration.](image)

3.3.2. Coma.

Coma appears when incident light rays form a relative large angle with the main optical axis. Again there is a difference of refraction, resulting in a blurred image. The exterior rays are refracted more than the central rays, see figure 3.6. The corresponding image points for the lower rays are shifted.

![Fig. 3.6. Coma.](image)
3.3.3. Astigmatism and field curvature.

For object points that do not coincide with the main optical axis, light rays are differently focussed (see figure 3.7). This phenomenon is called astigmatism. Rays in the vertical plane \( t \) are focussed in \( B_t \), rays in the horizontal plane \( s \) are focussed in \( B_s \). \( V \) is projected as a horizontal \( \text{cq. vertical line in } B_t \) \( \text{cq. } B_s \). Between those two planes \( V \) is projected as an ellips, except in \( B_c \), where \( V \) is projected as a circle.

Fig. 3.7. Astigmatism.

If \( V \) is on the optical axis it’s projected as a point in \( B_c \). The distance between \( B_t \) and \( B_s \) will increase with increasing distance between \( V \) and the optical axis, which leads to field curvature (see figure 3.8).

Fig. 3.8. Field curvature.
3.3.4. Distortion.

Distortion is the result of a varying magnification for different object points. If the magnification increases in the direction away from the optical axis the result is pincushion distortion, if the magnification decreases the result is barrel distortion, see figure 3.9. This is a serious problem for measurement applications.

![Pincushion and Barrel Distortion](image)

**Fig 3.9.** Distortion.

3.3.5. Chromatic aberration.

If white light is used for imaging, two kinds of aberration can occur, namely longitudinal and transverse chromatic aberration. Longitudinal chromatic aberration provides a blurred image because light of different colours comes to focus at different distances from the lens. Transverse chromatic aberration provides a shifted image because there's a chromatic difference of magnification. Light of different colours passing obliquely through a lens strikes the image plane at different points.

All the imperfections mentioned above cause a deviation from the MTF in figure 3.4., which reduces the maximal spatial frequency that can be seen in the image. For detecting the presence and the pixel location of edges in an image, those errors are only critical for small objects, when the contrast level is below the noise level in the image. Then no distinction can be made between noise or edges. If sizes have to be obtained from an image the accuracy of measurement falls off with increasing imperfections of the optical system.
4. EDGE DETECTION.

4.1. Introduction.

Detecting edges in a digital image is the registration of rapid local light-to-dark or dark-to-light transitions. The existence of edges can be detected with several techniques, depending on the progression of the grey values of the pixels across an edge. The conditions for edge registration depend on:

- the rapidity of change of the intensity level
- the magnitude of change of the intensity level
- the distance between two succeeding edges (e.g. the width of an object)
- the noise level in the image.

The performance of edge detectors is very different. Some methods may find all the edges but also respond to noise, others may be noise-insensitive but miss some crucial edges. This chapter deals with the problem of developing conditions to detect the presence of all edges in an image, regardless of the factors mentioned above. This means that sharp as well as blurred, bright as well as dark and small as well as wide objects should be registered, independent on the noise level.

4.2. Edge profile.

The edge profile in an image depends on the way of illumination. The intensity level of an edge under incoherent illumination is shown in figure 4.1.

![Fig. 4.1. Edge profile under incoherent illumination.](image-url)
The ideal edge profile is transformed by optical lenses. The edge becomes further blurred if the lens is defocussed but its true position, defined as the 50% intensity point, is independent on the degree of blur. Under coherent illumination the intensity profile across an edge is very different from figure 4.1. The true position is now at the 25% intensity point. Although the edge is much sharper, it is not perfectly smooth since the profile is more susceptible to distortion when spurious reflections occur, see figure 4.2.

![Fig. 4.2. Edge profile under coherent illumination.](image)

However, the edge profile in the eight bit video signal after sampling differs strongly from the analog signals, see figure 4.3. Besides, electronic noise and the fixed pixel sizes can cause errors in the edge position.

![Fig. 4.3. Edge profile in the sampled video signal.](image)
4.3. Thresholding.

This edge detection method compares all grey values in an image with a threshold level. Edges are detected where a pixel value \(f\) is under the threshold level and the value of a neighbour pixel exceeds that level (see figure 4.4).

![Thresholding diagram](image)

Fig. 4.4. Thresholding.

This procedure demands a minimum contrast which means that objects have to be large enough so that the pixel values across an edge can pass the threshold. Under proper illumination and good focussing the minimum distance between two succeeding edges has to be three pixels. When the distance becomes smaller or the image quality gets worse, this method is not sufficient anymore. The magnitude of the intensity change or the contrast is too small to cross the threshold level. The grey values of the background pixels are dominating the pixels which belong to the relative small object area.

4.4. Gradient.

Gradient operators measure the absolute magnitude of the difference between two neighbouring pixels and compares it with a certain limit value. An edge is registrated whenever this gradient exceeds the limit, which is yet very critical. Two problems arise if the video image contains small as well as large objects. First, two or more transitions across an edge might fulfil the condition if the limit is relative small compared with the magnitude of the intensity change over the whole edge. This occurs at the edges of large objects where the grey value \(f\) increases or decreases monotonously over several pixels. As shown in figure 4.5 four registrations will take place instead of two. The
transitions \( |f_1-f_0| \) and \( |f_2-f_1| \) on the same flank for example both exceed the limit value.

Fig. 4.5. Edge detection with a too small limit.

The second problem arises for small objects when the limit value is relatively high. No detections will take place because of the low contrast in the image (see figure 4.6). The edges at 1, 2, 3 and 4 do not fulfil the condition. So the performance of this gradient method depends on the variety of widths of the objects in the image and on the choice of the limit value.

Fig 4.6. Edge detection with a too high limit.
4.5. Extending the gradient conditions.

Several methods use the sum of two or more differences as basis of edge detection. However the results of those methods, like the Roberts, the Prewitt or the Sobel operators, are only satisfying for sharp images with a relatively large contrast. The problems mentioned above can be prevented by measuring not only one intensity change but also taking two more into account. The edge detection will become less dependent on the choice of the limit value.

By defining the largest gradient at a rising or a descending flank of the intensity level, even for relative small limit values only one registration across an edge takes place. The first problem of detecting too many edges for large objects is solved in this way and by setting the limit at a low level, the second can be solved to. The three conditions which have to be fulfilled for detecting an edge at \( r_1 \) (see fig 4.7) are:

\[
\begin{align*}
|d_1| &> \text{limit} \\
|d_1| &> |d_0| \\
|d_1| &> |d_2|
\end{align*}
\]

in which \( d_0 = f_1 - f_0 \)
\( d_1 = f_2 - f_1 \)
\( d_2 = f_3 - f_2 \)

Now the presence of all objects that are larger than two pixels can be detected. The constraint is that the limit must have a value which is higher than the noise level in the image, to prevent noise of being detected as an edge.
5. **RULERS.**

5.1. **Introduction.**

In grey-valued images a transition can be followed from the background video level to the object video level. With the edge conditions from chapter 4, the two pixels with the largest intensity change across an edge are detected. To estimate the true location of the edge a tool has to be developed to determine this position with an accuracy better than one pixel, so called subpixel resolution. A ruler is an image processing operator which searches for edges along a directed line in an image and calculates their positions [5].

5.2. **The ruler algorithm.**

When the largest gradient along an edge has been detected, the true edge position can be estimated in several ways:

- compute the average position of the maximum and the minimum of the whole edge profile as the edge position.
- compute the central point of the largest intensity change as the edge position
- fit a n-th degree polynomial through the pixel values around the largest intensity change and compute the point of the maximum of the derivative as the edge position.

Since the largest gradient is known, the first method is redundant. The local rapidity of the intensity change settles the accuracy that can be achieved. With the second method the highest accuracy that can be achieved is 0.5 pixel. The third method offers theoretically the best potentialities to detect edges with a high subpixel accuracy. The edge profile can be modelled by a third degree polynomial. The true edge position is assumed to be the point where the second derivative is equal to zero. Starting from the general equation of a third degree polynomial

\[ f[r] = a*r^3 + b*r^2 + c*r + d \]

the coefficients of this equation can be computed, based on the pixel values. (see figure 5.1).
The largest intensity change takes place between the points on the scanline at \( r_1 \) and \( r_2 \). Replacing \( r \) by \( r' = r - r_1 \) the following equations have to be solved:

\[
\begin{align*}
    f_0 &= a(-1)^3 + b(-1)^2 + c(-1) + d \\
    f_1 &= a(0)^3 + b(0)^2 + c(0) + d \\
    f_2 &= a(1)^3 + b(1)^2 + c(1) + d \\
    f_3 &= a(2)^3 + b(2)^2 + c(1) + d
\end{align*}
\]

Computing the second derivative to be zero results in

\[
    f'' = 6a r' + 2b = 0
\]

Solving (5.1) gives the following formulae for the coefficients \( a \) to \( d \):

\[
\begin{align*}
    a &= \frac{(-f_0 + 3f_1 - 3f_2 + f_3)}{6} \\
    b &= \frac{(f_0 + f_2 - 2f_1)}{2} \\
    c &= \frac{(-2f_0 - 3f_1 + 6f_2 - f_3)}{6} \\
    d &= f_1
\end{align*}
\]
The true edge position $r$ is equal to $r' + r_1$

$$r' = \frac{(f_0 + f_2 - 2*f_1)}{(f_0 - 3*f_1 + 3*f_2 - f_3)}$$
$$r = r_1 + r'$$

5.3. Realization of rulers for omnidirectional scanning.

By scanning for edges along a line in an arbitrary direction, those edges which are oriented perpendicular to the scan direction will be highlighted most. In this case the magnitude of the intensity level change is largest. However, locating the edge by fitting a polynomial and calculating the second derivative to be zero is affected by the noise level of the image. To reduce this influence the grey values on the scanline are computed by averaging the weighted pixel values of those pixels which positions are perpendicular to the scan direction. Former investigation [6] showed that averaging the value of three whole pixels is the best way of computing those grey values. Whenever the scanline is horizontal, vertical or diagonal it's simple to determine those values (see figure 5.2). In order to compute the grey values on the scanlines for other directions the following algorithm has been developed.

![Diagram](image)

horizontal scanning vertical scanning diagonal scanning

Fig. 5.2. Computing grey values in horizontal, vertical and diagonal scan directions.

In case the scan direction differs from horizontal, vertical or diagonal, the points on the mathematical line will almost never coincide with the pixel coordinates, so problems arise when computing the grey values. Yet every
scanline is completely determined by its starting \((x_1, y_1)\) and ending point \((x_2, y_2)\). The grey values can be computed with the slope of the line

\[
\frac{x_2-x_1}{y_2-y_1} = \Delta x / \Delta y.
\]

By sorting the starting and ending points, it's taken care that \(\Delta y \geq 0\) for all lines. Now the set of scanlines can be divided into four main groups (see figure 5.3):

1. \(0 \leq \Delta x / \Delta y \leq 1\)
2. \(1 < \Delta x / \Delta y\)
3. \(-1 \leq \Delta x / \Delta y \leq 0\)
4. \(-1 > \Delta x / \Delta y\)

with \(-511 \leq \Delta x \leq 511\) and \(0 \leq \Delta y \leq 511\), because of the image format being 512 * 512 pixels.

![Figure 5.3. Division of the set of scanlines.](image)

First the scanline is approximated by those pixels whose distances to the mathematical line are as small as possible. Those pixels will be called the 'centre pixels' (see figure 5.4) and are computed as follows [7]. Starting at \((x_1, y_1)\), one of the coordinates is changed by \pm 1. If \(\text{abs}(\Delta x) \geq \Delta y\) the x-coordinate is changed, else the y-coordinate. So the change takes place in the direction of the axis of greatest movement, the main direction. Then the other coordinate is changed or not, depending on an error
term maintained by the algorithm. This error term $e$ (see figure 5.4) records the distance to the mathematical line, measured perpendicular to the axis of greatest movement. The whole procedure is iterated until the ending point is reached.

![Figure 5.4. The centre pixels.](image)

For every $x$- respectively $y$-coordinate exactly one $y$- resp. $x$- coordinate is computed. E.g. if $\Delta y \geq \text{abs} (\Delta x)$ the axis of greatest movement is the positive $y$-axis. The centrepixels are computed by the following equations. Starting at $(x_1, y_1)$ the $y$-coordinate is incremented by one and then the equations

$$\frac{x - x_1 + 0.5}{y - y_1} \leq \frac{\Delta x}{\Delta y} \quad (\text{if } \Delta x \geq 0)$$

or

$$\frac{x - x_1 - 0.5}{y - y_1} \geq \frac{\Delta x}{\Delta y} \quad (\text{if } \Delta x \leq 0)$$

are investigated and if the comparison gives 'true' $x$ is increased or decreased by one, depending on the sign of $\Delta x$. After computing the centre pixels a grey value on the scanline is determined by the pixel value of the centre pixel and the weighted values of four neighbourpixels. The choice of those neighbourpixels depends on the scan-direction $\Delta x/\Delta y$ and the coefficient with which the values are multiplied depends on the absolute values of $\Delta x/\Delta y$ (see figure 5.5a to d). The sum of the weighting coefficients must still be three. Now the grey values approximate the average of the pixel values perpendicular to the scanline. The corresponding flowdiagram and software are in appendix 1.
Fig 5.5.a \( 0 \leq \Delta x/\Delta y \leq 1 \)  
Fig 5.5.b \( 1 \leq \Delta x/\Delta y \)  
Fig 5.5.c \(-1 \leq \Delta x/\Delta y \leq 0 \)  
Fig 5.5.d \(-1 \geq \Delta x/\Delta y \)  

Fig 5.5. Computing the grey values on the scanline with \( s = \text{abs} (\Delta x/\Delta y) \).

5.4. Results.

The accuracy of locating the edge position by polynomial-fitting as a function of the slope of the scanline is tested. Only the scanlines with a slope \( 0 \leq \Delta x/\Delta y \leq 1 \) are taken in account, because all the four groups can be transformed to this set by exchanging the x- and y-coordinates or by multiplying the x-coordinates with -1. The accuracy is measured by shifting a testobject with a constant small step in the scan direction and computing the edge coordinate in the main direction (appendix 2).
test object is oriented perpendicular to the scan direction and the test object is over three pixels large to provide a large contrast in the image. The process is done within a small part of the image area to reduce the influence of the system imperfections. The largest errors appear when the slope of the scanline is not equal to 0 or 1 (or -1 or +1 for the other three sets), which are the horizontal, vertical and diagonal lines. However when the error, made in the main direction is equal to ε then the total distance between computed edge position and the true position is equal to

\[ \sqrt{e^2 + (s\epsilon)^2} \]

So the edge locating error that is made for an arbitrary scan direction has to be adjusted by taking this added fault into account as well, as shown in figure 5.6.

Fig. 5.6. Edge locating error vs. slope s of the scanline.

The increase of inaccuracy with a rising slope of the scanline is the result of the way in which the pixel values are determined. This can be done exactly for horizontal, vertical and diagonal but not for other directions. For other directions the centrepixels mostly do not satisfy the mathematical line equation and the more the slope differs from 0 or 1, the more the mean distance between the centrepixels and the mathematical line increases and the more the computed grey value on the scanline differs from the true value in that particular point of the scanline.
6. EDGELS.

6.1. Introduction.

Edges can be defined by their location, amplitude and orientation. To extract all this quantitative information from a digital image, a tool has to be developed which divides a pixel that is located on an edge into an object part and a background part. In this way the array of pixel values can be transformed into geometric information.

6.2. The edgel algorithm.

The edgel algorithm is based on three standard image processing operators, namely Sobel filters in x- and y-direction and a lowpass filter [8]. With those three filters a plane through the neighbourhood of each pixel can be defined. The equation of a plane in the three dimensional image space is

\[ I = a*x + b*y + c \]

where \( I \) is the pixel luminance and \( x, y \) are the pixel coordinates. The coefficients \( a \) and \( b \) are computed by the Sobel operations mentioned above and represent the slope of the plane in x- and y-direction. Coefficient \( c \) is the result of the lowpass filter and represents the 'offset' of the intensity plane. The following 3*3 arrays of weights are used to calculate the 3 coefficients (see figure 6.1).

\[
\begin{align*}
S_x &= \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \\
S_y &= \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \\
S_i &= \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}
\end{align*}
\]

Fig 6.1. Convolution masks.

Those three kernels provide after normalization the following best fit plane through the area of nine pixels.

\[ 2*S_x*x + 2*S_y*y + S_i = 16*I \]
By crossing the best fit plane with a threshold plane a so-called threshold line is computed with the following equation:

$$S_x \cdot x + S_y \cdot y = 16 \cdot I_t - S_i$$

where $I_t$ is the threshold value. If this line can be projected in $I$-direction on the pixel under investigation, an edgel is computed (see figure 6.2), which is defined as the smallest part of an edge within a pixel.

![Fig. 6.2. Edgels.](image)

Based on the edge conditions of chapter 4 and the line computing of chapter 5, the edgel calculation is done on the two pixels with the largest intensity change along an edge. The most critical part of the edgel algorithm is to determine the threshold. Especially when the slope of the best fit plane is small, a slight change of the threshold value gives a considerable shift of the edgel position.

6.3. Investigation on the threshold.

The threshold level must be an approximation of the 50% level of the whole transition in incoherent illumination (or 25% for coherent illumination). At first the threshold
is computed as the mean pixel value within the 3*3 array. However, this way of computing the threshold is not satisfying for the whole range of orientations of objects. The speed of the transition within the 'threshold mask' fixes the threshold level. A slow transition provides a lower value than a rapid one. To prevent the threshold calculation of being dependent on this speed, it's computed as the mean of the corner pixels of the 3*3 mask (see figure 6.3). These corner pixels are a better estimation of the extreme values of a transition.

Fig. 6.3. Thresholding with the corner pixels of a 3*3 kernel.

However this approximation is only satisfying if the mask covers the whole transition and therefore its maximum and minimum pixel values. This means that the accuracy decreases with an increasing angle between the object and one of the axes of the pixel coordinate system, or with an increasing degree of defocussing or blur. In this case the 3*3 mask does not cover the whole transition of the edge in both directions anymore, but only a part of it. The variation in the computed threshold level is higher the more the angle between the test object and the horizontal axis differs from 0. Since the true edge position is assumed to be at the 50% or 25% intensity level, the dimensions of the 'threshold mask' has to be enlarged to cover the extreme pixel values of the whole transition. Another improvement is to compute the mean of the minimum and maximum pixel values within the mask instead of the mean value of all the pixels or of the corner pixels of the whole mask. This provides a better approximation of the 50% level. When coherent illumination is used it's easy to adapt this value to the 25% level. The only requirement for this method to be accurate is that the maximum and minimum intensity levels across an edge are both included in the threshold mask at the same time, so again enlarging of the mask is needed. The optimal mask dimensions depend on the quality of the image. For every application these dimensions have to be defined.
7. 2-DIMENSIONAL RULERS.

With the subpixel methods like the edgel and the ruler technique, a third way of computing edge points is developed. It's a combination of one-dimensional polynomial and two dimensional plane fitting. Whenever a scan takes place along a line, two centre pixels with the largest intensity change are detected. Whenever these points are not located in the same column or row, a two dimensional polynomial through a local area of 4 by 4 pixels around the centre pixels can be fitted, as shown in figure 7.1.

![Fig. 7.1. Two dimensional, third degree polynomial fitting.](image)

This fitting can be done with the least squared error technique. The common equation for a polynomial in the three dimensional image space is

\[ I = a*x^3 + b*y^3 + c*x^2*y + d*y^2*x + e*x^2 + f*y^2 + g*xy + h*x + i*y + j \]  

(7-1)

in which \( I \) = the intensity and \( x, y \) are the pixel coordinates. With the equation of the scanline being

\[ y = s*x + t, \]  

(7-2)

which is a plane in the three dimensional image space, a one dimensional polynomial is extracted out of (7-1), see
2d-ruler in figure 7.1, which can be regarded as the crossline between the two dimensional polynomial and the plane $y=sx+t$. Computing the second derivative of this polynomial to be zero provides the $x$-coordinate of the edge point on the scanline being

$$x = \frac{6bs^2t + 2ct + 4dts + 2fs^2 + 2e + 2gs}{6a + 6bs^3 + 6cs + 6ds^2}$$

With (7-2) the $y$-coordinate can be computed. If the centre pixels are in the same column or row the best fit polynomial is computed for an area of 3*4 or 4*3 pixels. The advantage of this technique with respect to the one-dimensional ruler is the fact that the grey values on the scanline are not computed as an average of grey values perpendicular to the scan direction so no information in this perpendicular direction is neglected. However, since computing a best fit two dimensional polynomial through 12 or 16 pixels demands quite a long time (300 - 400 ms) compared with the edgel or ruler technique (10 - 40 ms), this method is not suitable for inspection tasks. Time is very often a critical factor in these applications which decreases the practical value of this two dimensional rulers. Based on this only, the edgel and the one dimensional rulers are taken for further investigation.
8. APPLICATION 1 : BAR CODE TECHNOLOGY.

8.1. Introduction.
Bar code technology provides an accurate and inexpensive method of datamanagement for computerized information systems, which is often used for identification purposes. The traditional way of reading bar codes is by scanning with a laserbeam. The differences in reflectivity between bright and dark elements of the symbol are used as a basis to provide a logic signal that can be processed and decoded by a computer system. The disadvantage of scanning with a laser-beam is that there is a scan along only one line, so local defects can have a considerable influence and may lead to a 'mis-read'. Moreover a laser scanner is quite expensive. The advantages of using CCD-cameras with respect to laser scanners are:

- a CCD-camera is less expensive
- the scan area is equal to the whole field of view
- local defects are less critical.

8.2. Bar code symbology.
Every bar code symbol consists of a regular pattern of parallel wide and narrow bars and spaces, printed on a contrasting medium [8,9,10]. The term symbology refers to the structured characteristics of the symbols. The information of a symbol is determined by the used symbology, the relative widths and the sequence of the elements. Some of the most frequently used symbologies are shown in figure 8.1.

Fig 8.1. Frequently used symbologies.
Every bar code symbol consists, regardless of the used symbology, of the following zones (see figure 8.2):

- **Start margin**: zone without elements, usually white, to indicate that a symbol follows.
- **Start character**: a special bar/space pattern, which precedes the first character, used to identify the beginning of a symbol.
- **Data characters**: a sequence of bar/space patterns which represent the true information (data) of the symbol.
- **Stop character**: a special bar/space pattern to signal the end of the symbol.
- **Stop margin**: a zone without elements to distinguish between more symbols.

Fig. 8.2. Common bar code symbol structure.

The ratio between narrow and wide elements varies between 2 and 3. The number of wide and narrow bars and spaces per character is fixed and determined by the used symbology. Most symbologies also contain a checksum character, created by an arithmetic operation performed on the data characters in the message. When decoding the symbol, this operation is done also and the result is compared with the checksum character in the symbol.

The common structure of a bar code symbol is implemented in different ways in the existing symbologies, which can be classified by the available set of characters, the information density and the decoding strategy. After determining the relative width and the sequence of the elements, the symbol can be decoded if the used symbology is known.
9. READING BAR CODES.

9.1. Optics and illumination.

The investigation of accurate optical edge detection is implicated in the study of decoding bar codes with CCD-cameras, because decoding includes registering the presence and locating the edges of all elements of a barcode symbol. First problem that rises in accurate detecting and gauging is obtaining a good image. Because of the great difference in reflectivity between the bars and the spaces, the symbol has to be viewed without glinting. This is achieved by the system as shown in figure 9.1. Because of the symbols not being transparent, they have to be illuminated in reflection. To prevent light of being reflected directly into the camera a ring is used so that the incoming light rays make an angle of 45 degrees with the object surface. The front face and the lens of the camera are placed in the dark so no reflections of the camera will be visible.

![Diagram of Illuminating the bar code symbol.](image)

Fig 9.1. Illuminating the bar code symbol.

9.2. Detection and decoding

Reading bar codes with CCD-cameras can be divided into two tasks. First the presence, the sequence and the relative widths of all elements of the symbol have to be detected and secondly the information determined by these widths has to be decoded. This asks for knowledge of the used symbology and the global orientation of the symbol within the field of view. The algorithm for decoding had already been
developed and was available for all kinds of symbologies. This paragraph deals with the problem of performing the first task, starting with detecting the presence of all the elements. A problem that rises is the decreasing contrast in the image if the widths of the elements become smaller as seen in chapter 3 and 4. However, the more information a symbol contains, the higher the number of elements. This results in smaller elements with less contrast in the image. The smallest possible detectable width sets the maximal information density of a symbol, for a fixed field of view.

After having detected the presence of an element, its relative width has to be measured. This can introduce 2 kinds of errors for the decoding algorithm.

non-read: A wide cq. small element is detected as a small cq. wide element. The number of small and wide elements is not according to the used symbology anymore and the symbol can not be decoded. This can be caused by a local print error (see figure 9.2).

![Non-read example image](image)

Fig. 9.2. Non-read because of a local print error.

substitution: A small element is detected to be wide and a wide element is detected to be small. This causes a mis-read, the wrong character is decoded which might have considerable consequences.

Both errors can be the result of the inaccuracy in printing the symbol on the medium, of the ratio between wide and small elements not being constant and of the edge locating method not being accurate for small objects with little contrast. Summarizing, the main problems in measuring the relative widths of the elements are detecting the presence of all elements, independent on the noise level or the small contrast, and making a distinguished between small and wide elements.
9.3. Extending the edge conditions.

Assuming that the bar code symbol can be positioned in a way so that the scan direction is known, the presence of all elements wider than two pixels can be registered with the conditions developed in chapter 4. For smaller objects the intensity profile across an edge is shown in figure 9.3. In this case there are two edges within less than two pixels but the condition $\text{abs}(d_1) > \text{abs}(d_2)$ is not fulfilled.

![Diagram of edge detection for objects smaller than 2 pixels.](image)

Fig. 9.3. Edge detection for objects smaller than 2 pixels.

This means that the sign of the intensity change between two pixels has to be included in the edge conditions to separate dark- to light and light- to dark transitions. The resulting conditions for this situation are:

- For light- to dark transitions: 
  - $d_1 < \text{limit}$
  - $d_1 \leq d_0$
  - $d_1 < d_2$

- And for dark- to light: 
  - $d_1 > \text{limit}$
  - $d_1 \geq d_0$
  - $d_1 > d_2$

The limit value has to be higher than the noise level in the image. By placing a uniform white object under the camera and measuring the spread in the pixel values, this noise level is determined. The limit value for the edge conditions is set to twice the noise level, which improves the performance of the edge detection. The presence of all objects which width is over 1.4 pixels is detected and noise cannot be detected as an edge. If the distance becomes smaller the detection depends on the way in which the object is divided over the pixels, see figure 9.4. Here
an object of ± 1.2 pixels wide being positioned at the middle of a pixel or between two pixels means a great difference in detectability.

Fig. 9.4 Detectable and non-detectable small objects.

Now the exact location of the edge has to be computed. This is done with the ruler and with the edgel technique. By using the ruler technique the edge location can be computed whenever the distance between two succeeding edges is over 1.4 pixels. There's always a polynomial that, after detecting the edge presence, can be fitted through the pixel values and provides an edge location. For the edgel technique the elements can be too small to compute a best fit plane so that the threshold line can be projected on the centre pixel. This is shown for a one dimensional example in figure 9.5. The element is about 1.5 pixel wide but the Sobel kernel falls over two transitions instead of one. In this case the Sobel operation can even give a negative slope at a rising flank.

Fig 9.5. Sobel operation for objects smaller than 2 pixels.
This can be prevented by using the following 1x2 and 2x1 kernels to provide the slopes in x- and y-direction, although the slope of the best-fit plane is governed by the local speed of the intensity change now.

\[
S_y = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad S_x = \begin{bmatrix} -1 \\ 1 \end{bmatrix}
\]

The optimal threshold mask has to be defined by investigating the spread of the transitions. To cover the total transition at the edges of elements, wider than 2 pixels, a 6x6 mask is sufficient. Such a mask covers the extreme values. When the elements become smaller, the 6x6 mask covers not only the extreme values of one transition but also of a neighbour edge. In figure 9.6 a one-dimensional example is given to illustrate this phenomenon. A small bar and space are preceded by a wide space. The threshold value is not set by the edge under investigation. In this case the edges of small elements can not be computed because there's no crossline between the best fit plane and the threshold plane that can be projected on the pixel under investigation, pixel nr 3. This means that the mask has to be shrinked to a 3x3 mask. With this threshold mask and the 2 kernels mentioned above, all the elements can be detected and gauged with the edgel method. However using these kernels deteriorate the performance for the wide elements of the symbol, which require a larger mask.

![Threshold area diagram](image)

**Fig. 9.6. Detecting small elements with the edgel method.**
9.4. Results.

As an example a symbol is inspected with the ruler and the adjusted edgel technique. The ratio between the wide and small elements, which are about 1.4 pixel, is 2.5. The intensity profile along a scanline for this bar code symbol is plotted in figure 9.7.

![Intensity profile of the bar code elements.](image)

The relative widths of the elements can be defined by the distance between two points at two succeeding edges. The discrimination between bars and spaces is made, based on the sequence of two transitions. Dark-to-light transitions followed by a light-to-dark transition provides a space, the reverse succession provides a bar. The results are in table 9.1.

<table>
<thead>
<tr>
<th>Table 9.1 Inspection results for one bar code symbol (widths in pixels).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>mean width of small elements</td>
</tr>
<tr>
<td>mean width of wide elements</td>
</tr>
<tr>
<td>maximal measured width (wide element)</td>
</tr>
<tr>
<td>minimal measured width (wide element)</td>
</tr>
<tr>
<td>maximal measured width (small element)</td>
</tr>
<tr>
<td>minimal measured width (small element)</td>
</tr>
</tbody>
</table>
It is obvious that the adjusted edgel technique is less accurate for wide elements. There's less separation between small and wide. The best way to read barcodes is using the ruler technique, since this technique is less influenced by the high spatial frequencies. Moreover, the accuracy for such frequencies is very low since the spread in the measured widths is over 40% for both methods. However, using the relative widths in the decoding algorithm that corresponds with the used symbology results in a proper decoding of the bar code symbol. All symbols with elements which are wider than 1.4 pixel can be decoded. If the elements are smaller, the decoding result depends on the division of the symbol over the pixels.

Further, the influence of the image quality is investigated. By defocussing the lens, a blurred image is formed. For the wide elements, the situation can occur in which a transition is spread over several pixels. Then a kink in the signal can cause a double detection on one flank when using the former conditions. To prevent this, the value of a fifth pixel has to be taken into account to detect faint as well as steep edges just as one edge, see figure 9.8.

![Figure 9.8](image)

Fig. 9.8. Edge detection based on 4 grey values.

Depending on $d_2$ the value of the difference between $f_4$ and $f_3$ is taken into account or not. For a rising flank like in figure 9.6, it's assumed that there are two edges within 2 pixels if $d_2 < \text{limit}^*-1$, in which case $f_4$ is not important. If $d_2 > \text{limit}^*-1$ and $d_2 < d_1$ of course, then $d_3$ has to be smaller than $d_1$ also, to register an edge at $d_1$. This results in the following conditions for dark to light transitions.
\[ d_1 > \text{limit} \]
\[ d_1 > d_0 \]
\[ d_1 > d_2 \]
\[ \text{if}\ (d_2 > \text{limit} \times -1) \text{ then } d_1 > d_3 \]

and for light to dark

\[ d_1 < \text{limit} \times -1 \]
\[ d_1 < d_0 \]
\[ d_1 < d_2 \]
\[ \text{if}\ (d_2 < \text{limit}) \text{ then } d_1 < d_3 \]

Now all elements which are wider than 1.4 pixel can be detected and measured by the ruler technique even if there is a certain degree of blur in the image. This is an improvement with a factor 2.2 compared with the ordinary thresholding technique, which means that the information contained by the symbol can be 2.2 times as much as it was when the thresholding technique was used for decoding.
10. APPLICATION 2: THE SHAVERHEAD.

Every shaver, produced by Philips in Drachten consists among other things of three shaverheads in which a wreath of fifteen knifes is rotating. In figure 10.1, a cross section view of a shaverhead is shown. The shaverhead is made of stainless steel. The chamber of the head is flatted at the inside and polished at the top to obtain its desired thickness. Then the 90 gaps between the lamellae are sparked away. While shaving, the hairs of the beard fall between the lamellae and are cut by the rotating knifes. The shaving result strongly depends on the width of the gaps between the lamellae, which is one of the inspection objects for the head. This inspection is done at random now. The investigation of accurate optical measurement methods is implicated in the study of the feasibility of automated optical inspection of shaverhead gaps, because measuring the width of the gaps means measuring objects which can be oriented in every possible direction. The inspection demands are:

- $260 \leq \text{width} \leq 300$ microns
- inspecting at three different places within every gap
- measurements integrated over 50 microns
- considering the inspection time.

Fig. 10.1. Cross section view of a shaverhead.
11. OPTICS AND ILLUMINATION.

11.1. Image acquisition.

The first problem that rises for accurate optical inspection of the shaverhead is to obtain a good image. The main aspects in image acquisition are the illumination and the optics, which are both determined by the geometrical shape of the object and the inspection demands. It's important that all the contours that have to be inspected become clearly visible. For the shaverhead the edges of the gaps must provide a sharp transition in the image and no shadow or reflection in the camera may occur at the positions of the gaps. By illuminating the head in reflection, because of the polished top, too much light will be reflected directly into the camera and the gaps can not be detected. By illuminating the head in transmission, reflection and shadow is prevented and an image of the silhouette is provided with a large contrast. However, the magnification is highly dependent on the position of the object with respect to the optical axis and the distance between the lens and the object. This is expressed in the lens formulae \( \frac{1}{u} + \frac{1}{v} = \frac{1}{f} \) and \( m = \frac{v}{u} \) (\( f \) = focal length, \( u \) = object to lens distance, \( v \) = image to lens distance, \( m \) = magnification). It can introduce some considerable errors, when the lamellae of the head are not at the same height. Moreover, the image will become blurred when the object distance changes slightly. That's why a so called telecentric imaging method is more suitable for shaverhead inspection. The principle of telecentric imaging is shown in figure 11.1.

![Fig. 11.1. Telecentric imaging.](image-url)
The essential feature of telecentric optics is the 'stop' placed around the back focal point of the imaging lens. Any light rays passing through the back focal point must be parallel to the optical axis in the object space. The focal stop therefore constrains the system to accept only collimated light passing the object. Light beams with another direction will be held by the stop and will not contribute to the image acquisition. Lens 1 reduces the incoming light to light rays parallel to the optical axis, lens two is the imaging lens. If the object moves along the optical axis the image will be less sharp but the magnification remains the same.

To utilize the whole image area (512 * 512 pixels) the magnification can be established by taking a third lens into the system of which the focal point must coincide with the back focal point of lens 2 (see figure 11.2).

Fig. 11.2. Telecentric imaging for shaverheadinspection.

Now the system is telecentric in both object and image space. The total system magnification is \( f_2 : f_3 \). Because of the Philips CCD matrix camera having a sensor with dimensions 6.0 * 4.5 mm which contains 604 * 588 pixels, the diameter of the shaverhead being 22.1 mm and the field of view containing 512 * 512 pixels (frame memory size of the framegrabber) the maximum magnification is

\[
(4.5 \times 512 : 588) : 22.1 = 0.18
\]
The lenses which are chosen have focal distances $f_1=25$ mm and $f_2=160$ mm which provides a magnification of 0.16. This magnification gives rise to a slight tolerance in placing the shaverhead in the field of view.

11.2. Adjusting the light source.

There are two requirements for the source of light:

- the dimensions must be small to obtain as parallel as possible light beams behind the collimator lens.
- the light intensity must be low to prevent the gaps of becoming saturated, in which case the measured widths would be too large.

The first condition can be fulfilled just by placing a stop after the source of light in the back focal point of the collimator lens. A LED is chosen as source because of its already small dimensions, so little compensation by the stop is necessary. Since the intensity has to be very low, the influence of light of the environment of the imaging system is very large. Reflections because of this 'false' light dominate the image. By implementing a filter in the system which reduces the incoming light intensity by 100 and by increasing the intensity of the LED by 100, the quality of the image is settled by light from the LED. Another improvement is screening the system from this light by placing it in the dark. However because of the small light intensity at the camera surface the signal to noise ratio in the image is very bad. This ratio can be improved by controlling the camera gain manually. The optimal gain can be extracted by raising the light intensity of the LED to the maximum first, determined by the maximum diode current. Then one gap is measured as a function of the camera gain (see appendix 3). The intensity is established in the part of the function in which the measured width is nearly constant.
12. Centre, radius and image resolution.

To measure the gapwidths, the location of the shaving head in the field of view, the centre point and the diameter, has to be defined. After computing these two quantities the positions of the gaps where the measurements have to take place, are known. The image, as obtained by the telecentric system is shown in figure 12.1. By defining some edges at the inside circle of the head, the centre point and the radius can be computed. Since only the contours of the gaps are visible, these circle points have to coincide with the contour of a gap.

Fig. 12.1. Image of the shaverhead with the telecentric system.

Assuming that the shavinghead can be precisely positioned so that it completely fits in the field of view, the circle points are computed as follows. In four different directions a scan for light- to dark transitions in the video image is performed along five parallel scanlines. The starting coordinates for every group of five scanlines lie at a line perpendicular to the scandirection. Per group the transition, which is furthest removed from the starting coordinates of its scanline is assumed to be at the inside circle. The reason to scan along five lines to define just one circle point is shown in figure 12.2. Whenever a scanline exactly coincides with a lamella, no transition will be detected (line a). If the line crosses a gap, there's a possibility that the transition, although located at the contour of the gap, does not form a part of the inside circle (line b). By scanning along five lines,
there's at least one transition which coordinates are considered to be located on the inside circle, namely the transition which is furthest removed from the starting coordinates of its scanline (line c).

Fig. 12.2 Edges at the inside circle of the shaverhead.

In this way four points of the inside circle are provided. However because of the ratio of the pixel x- and y- dimensions being not equal to 1, the circle is projected into the pixel space as an ellips. The ratio of the pixel x- and y dimensions is 2/3. By multiplying all y- coordinates by 1.5 first, the ellips is transformed back into a circle. To define the middle point of the circle, a line equation of the lines through every pair of the transformed circle points is computed. The centre point is equal to the crosspoint of the perpendicular middle lines. This is shown for three circle points in figure 12.3.

Fig. 12.3. Middle point location.
L1 and L2 are the lines through a pair of circle points (A,B) and (B,C). The perpendicular middle lines ML1 and ML2 cross in the centre M of the circle. With 4 circle points there are 12 possible middle point calculations. The true centre is estimated by the average result. The average distance between the transformed circle points and the middle point of the shaver head is equal to the radius of the inside circle. To express the sizes in the transformed image afterwards in microns the radius is used to compute the image resolution since it's one of the best specified sizes of the head (see figure 10.1). To transform the gap width from pixels into microns the transform factor is

15.43 mm / radius in pixels

The total flow diagram and corresponding software for defining the middle point, radius, and image resolution is shown in appendix 4.
13. GAP WIDTH GAUGING.

13.1 Method of measurement.

Distances and sizes in an image can be measured by defining the coordinates of edges. Starting from a rectangular object like a gap, one scanline is obviously sufficient to determine the width, if the angle between scanline and gap is equal to 90 degrees. This scan provides a point on both sides of the object, $X_1$ and $X_2$ (see figure 13.1). The gap width is equal to the distance between those points,

$$
\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}
$$

Fig. 13.1. Gauging with one scanline.

The precision of this gauging method depends on the accuracy with which the object rotation and $(x_1, y_1), (x_2, y_2)$ at the edges can be computed. The expected maximum error is twice the error made in edge locating. By approximating the shaverhead by a 90-angle of scanlines (because there are 90 gaps), the widths of the gaps can be measured. However the angle between the scanline and the gap must be equal to 90 degrees. If this angle $\alpha$ differs, an extra error is introduced. As a result the ratio between the measured width $D'$ and the true value $D$ is a function of $\alpha$, namely $D'/D = 1 / \cos (90-\alpha)$.

By introducing a second scanline this error can be reduced. At every edge of the gap two points are detected, $X_{11}, X_{12}$ and $X_{21}, X_{22}$ (see figure 13.2). First the equation of the line through $X_{11}$ and $X_{21}$ is computed, line 1. The width of the gap is equal to the perpendicular distance between $X_{12}$
or X22 and this line.

Fig. 13.2. Gauging with two scanlines.

The largest error in gauging with this method occurs if the slope of the computed line 1 differs most from the real edge, see figure 13.3, in which case $\beta$ is maximal.

Fig. 13.3. Maximal error made by measuring with two scanlines.
The ratio between the measured width \( D' \) and the true value \( D \) is

\[
\frac{D'}{D} = \frac{(D \pm 2d) \cos \beta}{D}
\]

In which \( \beta \) = maximal angle between line 1 and the edge.
\( d \) = maximal error in edge locating, measured in perpendicular direction to the edge.

The error made because of the angle \( \beta \) between line 1 and the corresponding edge, or the factor \( \cos \beta \), can be eliminated by using a third scanline for gauging, parallel to and located between the other two, see figure 13.4.

13.4. Gauging with three scanlines

First the line equation of the edge is computed by fitting a line through the exterior points \( X_{11} \) and \( X_{21} \), line 1. Then the perpendicular distance between \( X_{32} \) and line 1 is measured as the width of the gap. The maximal error that can be made is twice the maximal error made in edge locating, independent on the angle between line 1 and its corresponding edge or of the angle between the scanlines and the object.
13.2. Results for rulers and edgels.

The accuracy of this gauging method is examined for the ruler technique by measuring the width of several test objects which are over three pixels wide. The maximum error is measured for the scanlines with a slope $s$ which satisfies $0 \leq s = dx/dy \leq 1$. The object is rotated so that the angle between the scanline and the object varies from 90 to 45 degrees. The maximum errors that occur are shown in figure 13.5.

![Gauging accuracy vs. angle between scanline and object](image)

Fig. 13.5. Gauging accuracy with rulers versus object rotation for several scanline slopes $s$.

If the angle between object and scanline is equal to 90 degrees, the maximum error is twice the maximum error made in edge locating, as expected, but when the slope $s$ of the scanlines rises there is a decrease in the gauging accuracy
which is more for larger slopes. If the angle between scanline and object differs from 90 degrees, the error \( e \) (see figure 13.6) in computing the coordinates of the edge point on the scanline increases. However this is partly compensated since the distance between the computed edge point and the real edge is equal to \( d \), which does not increases as quickly as \( e \).

![Fig. 13.6. Error in scanning not perpendicular.](image)

There are two explanations for this phenomenon. First the centre pixels of the scanline can be in or outside the object although the corresponding mathematical position on the scanline is at the other side. This error disturbs the computed grey values and increases with a rising slope \( s \) and with a rising angle between scanline and object. Secondly the intensity profile of an edge is spread over more pixels if the object is less parallel to one of the axes of the pixel coordinate system.

Furthermore it's notable that the rulers with \( s=0 \) (e.g. the horizontal or vertical ruler) are most accurate, independent on the rotation of the object. The maximum error that can be made is 0.27 pixel for the most unfavourable case in which the angle between these rulers and the object is 45 degrees. Obviously the influence of the factors mentioned above is less for those two rulers and the compensation is maximal. Summarizing, when inspecting the shaverhead with the ruler method, the highest accuracy that can be achieved is 0.27 pixel if only horizontal and vertical rulers are used.
The accuracy of measuring with the edgel method is tested as a function of the orientation of the same test object used for investigation of the ruler technique, by inspecting the middle of the edgels. This is considered to be the most stable and accurate estimation of the true position of an edge point. The conditions of chapter 4 are used to register the presence of an edge. After detecting the two pixels with the largest intensity change, pixel 1 and pixel 2 in Figure 13.7, the edgel technique is executed on those pixels. Whenever the middle point of the edgel is in the corners of pixel 1, it is discarded and pixel 2 is examined. In every row or col across an edge the longest edgel is found in this way.

Fig. 13.7. Neglecting the smallest edgels.

To detect the best dimensions for the threshold mask, the accuracy has been tested with several kernels. The scan direction is not of any importance for the edgel technique, since the edgel computation is based on a local environment of the pixel under investigation, so the only parameters in the test are the object rotation with respect to one of the axes and the dimensions of the threshold mask. The results are shown in Figure 13.8. By enlarging the dimensions of the mask, the kernel covers the maximum and minimum pixel value of a transition more and more, even if the object is rotated 45 degrees. The operational area increases as the dimensions of the mask are incremented. Using a 6*6 mask covers the whole transition for all rotations. Now the error made in gauging the gap width is independent with respect to the rotation and is equal to 0.18 pixel.
Fig. 13.8. Gauging error versus object rotation, for several threshold masks.

For reasons of symmetry the threshold mask is enlarged to a 7*7 mask. The dimensions of the Sobel and lowpass filters are also odd so they can be placed in the centre of the threshold mask, see figure 13.9.

Fig. 13.9. Mask configuration.
Now the accuracy is independent on the object rotation or the scan direction and is equal to 0.18 pixel. This can be explained, since the edgel method takes transitions in horizontal and vertical direction into account at the same time while the ruler technique only considers the magnitude and not the orientation of the transitions perpendicular to the scan direction.

To compare the two subpixel resolution techniques totally, the influence of the maximum contrast in the image is investigated. The widths of the testobjects are measured as a function of the maximum contrast in the image. The results are shown in figure 13.10. The contrast is represented by the total magnitude of the intensity change at the edges. The measured width varies within a range of 1% if the maximum contrast varies by a factor three. For both methods the influence of the maximum contrast can be neglected within this range of light intensity, because the computed image resolution is obviously also dependent on the maximal contrast.

![Graph showing measured width vs. maximal contrast](image)

**Fig. 13.10. Influence of the maximal contrast.**

Finally the same tests are done for both methods on the so called SBIP (Single Board Image Processor) to investigate the influence of the clock of the framegrabber, since the SBIP's clock is much more stable than the clock of the PCVISIONplus framegrabber. There were no differences in the results so the influence of the instability of the framegrabbers clock can be discarded. The best way to inspect the shaver head is by use of the edgel technique.
13.3. Defining the scan position.

The location where the gap under investigation has to be measured can be extracted from the information about the middle point \((X_m, Y_m)\) and the radius \((R)\). This will be explained for gaps within the range of 0 to +45 degrees with the vertical \(=x\)-axis. The scan area for gap 1 is determined by, see figure 13.11

\[
X_m + R < x < X_m + R + R'
\]

![Diagram showing scanlines for gap width inspection](image)

Fig. 13.11 Defining the scanlines for gap width inspection.

The starting \(y\)-coordinate of the first 3 scanlines = \(Y_m\), the starting \(x\)-coordinates lie equidistant in the range, mentioned above and their scan direction is horizontal to the right. On every line the first black to white and the next white to black transitions are computed to detect the first gap. If the starting coordinates of the scanlines lie within a gap, the first edge will be neglected. Then with the gauging method the gap width can be expressed in
pixels. Since the angle between two gaps is 4 degrees the starting x-coordinates of the i-th set of 3 scanlines are within the area

\[ X_{m} + R \times (\sin(90 - (i-1) \times 4)) < x < X_{m} + (R + R') \times (\sin(90 - (i-1) \times 4)) \]

The starting y-coordinates are equal to the y-coordinates of the edge points found on the corresponding scanlines for gap i-1. The flowdiagram for measuring the gap widths this way and the associated software are in appendix 5.
14. Total inspection.

First the middle point and the radius to define the transform factor of the image are computed with the edgel technique. The repeatability in determining the radius in pixels is 0.18 pixel so the transform factor or the image resolution is computed with an accuracy of 0.4%. This is almost equal to the tolerance of the diameter of the shaverhead. The repeatability of the gap width measurements is 0.18 pixel, so the maximal absolute error in the measured width of a gap

\[ B' = (B \pm 0.18) \times (\text{res} \pm 0.004\times\text{res}) \]

is equal to

\[ (0.18+0.000072)\times\text{res} + B\times0.004\times\text{res} = 0.18\times\text{res} \quad (14-1) \]

in which \( B' \) = measured width in microns
\( B \) = real width in pixels.
\( \text{res} \) = real image resolution in microns/pixel.

In the case of the shaverhead this maximal error is about 8 microns. The total inspection time of the shaverhead is 4.26 seconds when inspected with the hardware system of chapter 1. This time is divided over the several inspection tasks as shown in tabel 14.1.


<table>
<thead>
<tr>
<th>Inspection task</th>
<th>time</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gauging time per gap</td>
<td>43 ms</td>
<td>3870 ms</td>
</tr>
<tr>
<td>Middle point definition</td>
<td>340 ms</td>
<td>340 ms</td>
</tr>
<tr>
<td>Computing the radius and resolution</td>
<td>20 ms</td>
<td>20 ms</td>
</tr>
<tr>
<td><strong>Total inspection time</strong></td>
<td>4260 ms</td>
<td></td>
</tr>
</tbody>
</table>

The accuracy of the inspection can be increased by zooming in the object which reduces the factor \( \text{res} \) in formula 13-1. In this case the shaverhead has to be turned around in the field of view of the camera or several cameras have to be used which all inspect a part of the shaverhead. The total inspection time can be reduced 5-10 times by implementing the software on a digital image processor (like the SBIP) in assembler language.
15. CORRECTING DISTORTION.

15.1. Introduction.

Digital images, produced by area-array cameras and frame-grabbers are subject to several kinds of spatial distortion as seen in chapter 3. Unless corrected, distortion can severely limit the accuracy of dimensional measurements, like the gap width inspection, made by an imaging system. This chapter discusses a practical way to measure and correct distortion in digital imaging systems.

15.2. Measuring the distortion.

In order to correct the present image distortion, its magnitude and direction is measured as a function of the pixel position. Therefore, an image is acquired of a calibration target that contains a pattern of so called control points, i.e. points with known locations in a world coordinate system [11]. A draught-board structure is chosen as calibration target and the contact points between the fields of the board are serving as control points, of which the coordinates with respect to the origine 0 (X₀, Y₀) in the world coordinate system are equal to (see figure 15.1):

\[ P_{ij} = (X_{ij}, Y_{ij}) \]

Fig. 15.1. Calibration target, with control points in a world coordinate system, for image correction.
The coordinates of \( P \) satisfy the following equations:

\[
X_{ij} = X_0 + j \cdot \Delta X \\
Y_{ij} = Y_0 + i \cdot \Delta Y
\]  

(15-1)

The positions of the control points are defined in the pixel coordinate system by computing all the crosspoints of the boundary lines of the fields and are represented by:

\[ P_{ij} = (x_{ij}, y_{ij}) \]

These are the pixel coordinates based on imaging without image correction. The distortion is equal to the difference between these uncorrected coordinates and the expected coordinates for perfect imaging, represented by:

\[ \Pi_{ij} = (\alpha_{ij}, \beta_{ij}) \]

which satisfy:

\[
\alpha_{ij} = \alpha_0 + j \cdot \Delta \alpha \\
\beta_{ij} = \beta_0 + i \cdot \Delta \beta
\]  

(15-2)

This distortion can be divided in an error in \( x\)-direction \( f_x \) and an error in \( y\)-direction \( f_y \), with

\[
f_x = x - \alpha, \\
f_y = y - \beta
\]  

(15-3)

In order to compute the coordinates \( (\alpha, \beta) \) which belong to ideal imaging, the following method is applied. The image point \( o_0 \), \( (\alpha_0, \beta_0) \) of the origine 0 is considered as reference point for the ideal world- to pixel coordinate transformation, denoted as \( G \). With respect to \( o_0 \),

\[
\alpha_0 = G(X_0) \\
\beta_0 = G(Y_0)
\]  

(15-4)

the coordinates of the other pixels have to be corrected. Then \( \Delta \alpha \) and \( \Delta \beta \) are defined as the mean distance in \( \alpha\)-cq. \( \beta\)-direction between two succeeding control points.
\[
\Delta \alpha = \frac{\sum_{i=1}^{M-1} \sum_{j=1}^{N} (x_{i,j+1} - x_{ij})}{M \times N} \\
\Delta \beta = \frac{\sum_{i=1}^{M-1} \sum_{j=1}^{N} (y_{i+1,j} - y_{ij})}{M \times N}
\]

in which \(M\) is the number of control points in vertical direction and \(N\) is equal to the number of control points in horizontal direction. Now the error made at position \(\Pi\) in the framegrabber coordinate system is equal to

\[G(P) - p = \Pi - p,\]

in which \(G\) can be defined as

\[
G(X,Y) = \begin{bmatrix}
\Delta \alpha/\Delta X & 0 \\
0 & \Delta \beta/\Delta Y
\end{bmatrix} \cdot \begin{bmatrix} X - X_0 \\ Y - Y_0 \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \end{bmatrix}.
\]

The accuracy of measuring this image distortion depends on the following factors:

- The number of control points that is included in the measurements. This factor influences especially \(\Delta \alpha\) and \(\Delta \beta\).

- The accuracy in determining the position of the control points in the framegrabber coordinate system. Whenever the measured distortion is less than the error that can be made in locating the control points, there's no unambiguous distinction between errors caused by the imaging system and errors caused by the measurement method.

- The accuracy in making the calibration target. Here also the measured distortion has to be more than the maximal tolerance in placing the fields on the draught-board.

- The accuracy in placing the calibration target so that the rows and columns of the target run parallel with the axes of the framegrabber coordinate system, see figure.
15.2. When the axes of both the coordinate systems do not run parallel an error $e_1$ in $y$- and $e_2$ in $x$-direction between the expected and computed edge location is measured which are not caused by the imaging system and satisfy:

$$\frac{e_2}{e_1} = \tan (90 - \tau)$$

in which $\tau$ is the angle between the corresponding axes of the world and framegrabber coordinate system.

Fig. 15.2. Error caused by placing the calibration target not properly.

This error can be eliminated by not starting from control points but from control lines, which will be explained for the measurements of distortion in horizontal direction. By scanning for edges in horizontal direction at equidistant heights in the image, points on the scanlines will be computed with the same $x$-coordinates $X_s$, regardless of the orientation of the calibration target (see figure 15.3). The line $y=Y_1$ is assumed to be the reference line. The measured distance $\Delta y$ between two succeeding points on one scanline is a weight for the local resolution in horizontal direction of that particular part of the image. After computing the mean distance $\Delta \beta$, the error in $y$-direction per field $i$ on the line $x=X_s$ is equal to $e_i = \Delta y_i - \Delta \beta$. The error $f_y$ with respect to the control line
$y = y_i$ progresses cumulatively as

$$f(x_s, y_{ij}) = \sum_{k=i}^{k=i} \varepsilon_i = \sum_{k=1}^{k=1} (\Delta y_k - \Delta \beta),$$  \hspace{1cm} (15-7)

---

**Fig. 15.3.** Measuring the distortion in y-direction with horizontal scanlines.

In this way, the error in horizontal direction for every control point can be measured. For the distortion in vertical direction, the same procedure is applied with vertical scanlines.

### 15.3 Correction

In order to correct the measured distortion, the coordinates of the computed control points have to be corrected by transforming their coordinates $(x_{ij}, y_{ij})$ into

$$\alpha_{ij} = x_{ij} - f_x$$

and

$$\beta_{ij} = y_{ij} - f_y.$$  \hspace{1cm} (15-8)
This can be done by the following procedure [12]. Starting from a triangle formed by 3 control points (see figure 15.4), a correction matrix

\[
D = \begin{bmatrix}
D_{11} & D_{12} \\
D_{21} & D_{22}
\end{bmatrix}
\]

and a correction vector

\[
e = \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}
\]

are computed.

![Correction per triangle diagram](image)

**Fig. 15.4. Correction per triangle.**

In this way a set of matrices and vectors is computed to apply a correction \( C \) so that for all triangle the following equation is satisfied:

\[
C(p_{i,j}) = \Pi_{i,j} \\
C(p_{i+1,j}) = \Pi_{i+1,j} \\
C(p_{i+1,j+1}) = \Pi_{i+1,j+1}
\]

This correction is exact for all control points of the calibration target. A point \( p \) in the framegrabber coordinate system is corrected by first determining the triangle of which \( p \) is part of and then correcting the coordinates with the particular matrix and vector. However this method
demands a large amount of memory to store all the correction data. Furthermore the correction can not be computed quickly. This is problematic since time is often a critical factor for inspection applications. That's why the least squared error method is used to compute one continuously derivable correction function $f$ for the whole field of view. Although this function does not exactly satisfy (15-8), it's the best approximation for $f_x$ cq. $f_y$ in all control points.

15.4. Results.

With a distorted draught-board structure the influence of the choice of the reference line and the type of the function $f$ is investigated for the correction of errors in horizontal direction. The spread of the distances between two succeeding control lines before being corrected is shown in figure 15.5.

![Graph](image-url)

**Fig. 15.5. Distance distribution without correction.**

Four corrections are implemented with use of a second and a third degree correction polynomial and with respect of the two most obvious reference lines namely $Y_1=0$ en $Y_1=255$. These methods provide a distance spread as shown in
appendix 6. Based on these results the method with a third degree polynomial with respect to $Y_1=0$ is chosen. Correcting in this way provides the strongest decrease of the spread in measured distances. Correcting with respect to $Y_1=255$ inclines to the maximum distance of the uncorrected image. Since $f_x$ or $f_y$ progress cumulatively, they can not be properly estimated by a 2-degree polynomial. To reach a possible improvement of the inspection results for the gap width measurements of the shaverhead, the correction method is implied for the telecentric imaging system of figure 10.2. As expected the spread in horizontal as well as in vertical direction is very low since the focal distances of the optical lenses used are relatively large.

Although the spread is little, there's a slight decrease in the measured distances when the control lines are further removed from the reference line. This is not an error of the lenses or the camera since rotating these does not have any influence on this distortion. The phenomenon is caused by the fact that the source of light is not proper positioned in the focal point of the collimator lens but somewhere else in the focal plane. By shifting the diafragma in front of the source the perfect situation is better approximated and the spread is so little, that the appearing distortions are less than the gauging errors as shown in figure 15.6 for the correction of horizontal distortion.

![Graph](image_url)

Fig. 15.6. Spread of the distances in horizontal direction without correction for the telecentric system.
Performing a correction could possibly introduce larger errors than those errors which appear in the image without correction. This means that no correction has to be implied for inspecting the shaverhead with this telecentric system. For the distortion in y-direction the spread is also so little that no correction is needed to improve the inspection results. The corresponding flowdiagram and software for measuring and correcting appearing image distortion are provided in appendix 7.
15. CONCLUSIONS AND RECOMMENDATIONS.

Subpixel interpolation techniques are very promising to achieve high resolution measurements with CCD-cameras for many applications.

The presence of objects in an image which are over 1.4 pixels wide, can be registered with edge conditions based on the progression of the pixel values along an edge. The two pixels with the largest intensity change between them are detected. Noise cannot be detected as an edge.

With rulers, it is possible to compute edge locations with an accuracy better than one pixel. This computation is done by fitting a third degree polynomial through four succeeding grey values on a scan line and computing the second derivative to be zero. The one-dimensional ruler can be applied in an arbitrary direction and the grey values on the scan line are determined by the scan direction and the absolute value of the slope of the scan line. The maximal error that can be made in edge locating for objects which provide a maximal contrast in the image, i.e. objects which are over three pixels wide, is 0.16 pixel, when scanning takes place in perpendicular direction to the edge. When only horizontal and vertical scans are applied the accuracy increases to 0.14 pixel.

By fitting a two dimensional plane through a local area of 3 by 3 pixels around one of the two detected pixels along an edge and projecting the crossline of this plane with a threshold plane on that particular pixel, a part of the edge within that pixel, a so-called edgel, is computed. The threshold value is equal to the mean of maximal and minimal pixel value along the edge. The maximal error that can be made now in edge locating is independent on the scan direction and is 0.09 pixel.

The two subpixel methods can be combined by fitting a two dimensional third degree polynomial through a local area around the two detected pixels with the least squared error method and compute the second derivative in a one dimensional part of this polynomial to be zero. However, the practical value of this method compared with the other two is almost nil, since the execution time is 10 to 40 times as much, so this method is neglected.

For objects which do not provide the maximal contrast in the image, like small barcode elements the accuracy of edge locating decreases and with the former edgel technique small elements cannot even be located, which leads to an adjustment of the edgel technique. But still the decrease of the accuracy is much more for the edgel technique. For elements smaller than 3 pixels, as in the case of barcode
decoding, the best method to detect and measure all the elements is the ruler technique.

Geometric measurements like measuring the width of the gaps of a shaverhead can be done with CCD-cameras, by using a telecentric system in which the optical magnification is independent on the object location.

When measuring objects, which are wider than three pixels with the ruler technique, the best way is to use only the vertical and horizontal rulers which provide the most accurate inspection results. The maximal gauging error in this case is 0.27 pixel, which is twice the error in edge locating. When using the edgel technique the maximal gauging error is 0.18 pixel independent on the object rotation.

These errors increase with a decreasing image quality. To correct the errors made by the imaging system, a calibration can be performed, which measures and corrects the image distortion by transforming the distorted pixel coordinates with a correction polynomial into proper coordinates. For every optical inspection task illumination and optics are critical but high resolution measurements are possible.

To improve inspection with the edgel technique with respect to inspection time, the computation of edgels should be implemented in hardware, which allows real time edge detection.

The two dimensional ruler could be implemented by performing LU decomposition instead of the least squared error method to compute the best fit two dimensional third order polynomial.
Literature.


7 Newmann, W.M., Sproull, R.F., Principles of computer graphics, 1979, p.17-29.

8 Hewlett Packard, Elements of a barcode system, application note 1013, 1983.


Appendix 1. Flowdiagram and corresponding software for omni directional scanning with the ruler technique.

- Input starting and ending coordinates of the scanline.
- Sort the coordinates so that $\Delta y \geq 0$.
- Select one of the four scan areas (1 to 4).
- Perform the grey value computation for the particular scanline.
- If edge presence detected, perform polynomial fitting.
- Compute the second derivative to be zero.
- Output edge $x$- and $y$-coordinates.
#include <math.h>
#include <stdio.h>
#include <conio.h>
#include <itexpfg.h>
float swn[512];
float edgex[512], edgey[512];
int tellerv, tellerh;

scanline (x1,y1,x2,y2,limiet)  
/* (x1,y1) and (x2,y2) are the starting and ending coordinates of 
the scanline. This procedure sorts these two points so that 
y2-y1=dy>0. Then it looks for the scan area based on the slope 
dx/dy of the line. */
int x1,y1,x2,y2,limiet;
{
    int h;
    float s,dx,dy;
    if ((y2-y1) < 0) {
        h=x1; x1=x2; x2=h;
        y1=y2; y2=h;
    }
    dx=x2-x1; dy=y2-y1;
    if (dx==0) scanhor (x1,y1,x2,y2,limiet);
    else if (dy==0) scanvert (x1,y1,x2,y2,limiet);
    else if (0<dx/dy && dx/dy<=1) scan1 (x1,y1,x2,y2,limiet);
    else if (-1<=dx/dy && dx/dy<0) scan2 (x1,y1,x2,y2,limiet);
    else if (dx/dy>1) scan3 (x1,y1,x2,y2,limiet);
    else if (dx/dy<-1) scan4 (x1,y1,x2,y2,limiet);
}

scan1 (x1,y1,x2,y2,limiet)  
/* scan1 performs a scan in the area 0<dx/dy<1, locates the 
present edges with subpixel accuracy, and places the edge 
coordinates in edgex[], and edgey[]. (x1,y1) and (x2,y2) are the 
starting cq. ending coordinates of the scanline and limiet is the 
value that must be exceeded by the largest gradient at an edge 
profile. */
int x1,y1,x2,y2,limiet;
{
    int k,z,d0,d1,d2,d3,col,row,i;
    float s,a,b,h,x,y,dx,dy;
    limiet=limiet*3; 
    col=y1;row=x1; dx=x2-x1; dy=y2-y1; i=0; k=0; s=dx/dy;
    a=(float) (y2-y1) /(x2-x1);
    b=(float) (y2+y1-(a*(x2+xl)/2; 
    while (col<y2) {
        sum[i]=s*(brpixel(col+1,row-1)+brpixel(col-1,row+1));
        sum[i]=sum[i]+brpixel(col,row);
        sum[i]=sum[i]+(1-s)*(brpixel(col,row-1)+brpixel(col,row+1));
        i++; col++;
        if (((row-xl+0.5)/(col-y1)) <=dx/dy) row++;
    }
    d0=sum[1]-sum[0]; d1=sum[2]-sum[1]; d2=sum[3]-sum[2];
for (z=4; z<i; z++)
{
    d3=sum[z]-sum[z-1];
    if (d1>limiet)
    {
        if (d1>d0 && d1>=d2)
        {
            if (d2<limiet*1 || (d2>limiet*1 && d1>d3))
            {
                h=(sum[z-4]-sum[z-3]*2+sum[z-2])*64;
                h=h/(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
                y=(z-3+y1)*64; x=(y-b*64)/a;
                edqey[k]=y+h; edqex[k]=x+s*h;
                k++;
            }
        }
    }
    else if (d1<-1*limiet)
    {
        if (d1<d0 && d1<=d2)
        {
            if (sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]!=0)
            {
                h=(sum[z-4]-sum[z-3]*2+sum[z-2])*64;
                h=h/(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
                y=(z-3+y1)*64; x=(y-b*64)/a;
                edqey[k]=y+h; edqex[k]=x+s*h;
                k++;
            }
        }
    }
    if (k==2) break;
    d0=d1; d1=d2; d2=d3;
}
edqey[k]=-1; edqex[k]=-1;

scan2(x1,y1,x2,y2,limiet)
/* scan2 performs a scan in the area -1≤dx/dy≤0, locates the present edges with subpixel accuracy, and places the edge coordinates in edqex[], and edqey[]. (x1,y1) and (x2,y2) are the starting cq. ending coordinates of the scanline and limiet is the value that must be exceeded by the largest gradient at an edge profile. */

int x1,y1,x2,y2,limiet;
{
    int k,z,d0,d1,d2,d3,col,row,i;
    float s,a,b,h,x,y,dx,dy,p,q;
    limiet=limiet*3; col=y1; row=x1; dx=x2-x1; dy=y2-y1; i=0;
    k=0; s=-dx/dy;
a = (float) (y2-y1)/(x2-x1);
b = (float) ((y2+y1-(a*(x2+x1)))/2;
    
    while (col < y2)
    {
        sum[i] = s*(brpixel(col-1,row-1)+brpixel(col+1,row+1));
        sum[i] = sum[i] + brpixel(col,row);
        sum[i] = sum[i] + (1-s)*(brpixel(col,row-1)+brpixel(col,row+1));
        i++; col++;
        if (((row-x1-0.5)/(col-y1)) >= dx/dy) row--;
    }

for (z = 4; z < i; z++)
{
    d3 = sum[z] - sum[z-1];
    if (d1 > limiet)
    {
        if (d1 <= limiet)
        {
            if (d2 < limiet && d1 >= d2)
            {
               h = (sum[z-4] - sum[z-3]*2 + sum[z-2])**64;
               h = h / (sum[z-4] - sum[z-3]*3 + sum[z-2]*3 - sum[z-1]);
               y = (z-3 + y1)**64 / a;
               edgex[k] = y+h; edgey[k] = x-s*h;
               k++;
            }
        }
    }
}
if (k == 2) break;
d0 = d1; d1 = d2; d2 = d3;
}
edgex[k] = -1; edgey[k] = -1;
}{
scan3(x1, y1, x2, y2, limiet)
/* scan3 performs a scan in the area dx/dy > 1, locates the present
edges with subpixel accuracy, and places the edge coordinates in edgex[], and edgey[]. \((x1,y1)\) and \((x2,y2)\) are the starting and ending coordinates of the scanline and limiet is the value that must be exceeded by the largest gradient at an edge profile. */

```c
int x1,y1,x2,y2,limiet;
{
    int k,z,d0,d1,d2,d3,col,row,i;
    float s,a,b,h,x,y,dx,dy;
    limiet=limiet*3;iol=ylirow= x1 idx=x2-xl;dy=y2-yl;i=O;
    k=O;s=dy/dx;
    a=(float) (y2-y1) / (x2-x1);
    b=(float) (y2+y1- (a* (x2+x1) »/2;
    while (row < x2)
        |
    sum[i]=s*(brpixel(col+1,row-1)+brpixel(col-1,row+1));
    sum[i]=sum[i]+brpixel(col,row);
    sum[i]=sum[i]+(l-s) * (brpixel(col-1,row)+brpixel (col+1,row) );
      i++;row++;
    if ((col-yl+0.5)/(row-x1» <=dy/dx C01++i
    }
    d0=sum[1]-sum[0];d1=sum[2]-sum[1];d2=sum[3]-sum[2];
    for (z=4;z<i;i++)
    
    d3=sum[z]-sum[z-1];
    if (d1>limiet )
    {
        if (d1>d0 && d1>=d2)
        {
            if (d2<limiet*1 || (d2<limiet*1 & d1>d3))
            {
                if (sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]!=0)
                {
                    h=(sum[z-4]-sum[z-3]*2+sum[z-2]) *64;
                    h=h/(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
                    x=(z-3+x1) *64;y=a*x+b*64;
                    edgex[k]=x+h;edgey[k]=y+h*s;
                    k++;
                }
            }
        }
        else if (d1<limiet & m==1)
        {
            if (d1<d0 & d1<=d2)
            {
                if (d2<limiet || (d2<limiet & d1<d3))
                {
                    if (sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]!=0)
                    {
                        h=(sum[z-4]-sum[z-3]*2+sum[z-2]) *64;
                        h=h/(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
                        x=(z-3+x1) *64;y=a*x+b*64;
                        edgex[k]=x+h;edgey[k]=y+h*s;
                    }
                }
            }
        }
    }
```
m=0; k++;  
if(k==2) break;  
d0=d1; d1=d2; d2=d3;  
edgex[k]=-1; edgey[k]=-1;  
}
}
}
}
if (k==2) break;
d0=d1; d1=d2; d2=d3;
edgex[k]=-1; edgey[k]=-1;
}

scan4(x1,y1,x2,y2,limiet)
/* scan4 performs a scan in the area \frac{dx}{dy} \leq -1, locates the present edges with subpixel accuracy, and places the edge coordinates in edgex[], and edgey[]. (x1,y1) and (x2,y2) are the starting cq. ending coordinates of the scanline and limiet is the value that must be exceeded by the largest gradient at an edge profile. */

int x1,y1,x2,y2,limiet;
{
  int m,k,z,d0,d1,d2,d3,col,row,i;
  float s,a,b,h,x,y,dx,dy;
  limiet=limiet*3; m=0; col=y1; row=x1; dx=x2-x1; dy=y2-y1; i=0;
  k=0; s=-dy/dx;
  a= (float) (y2-y1) / (x2-x1);
  b= (float) (y2+y1- (a* (x2+x1)))/2;
  while (col < y2) {
    sum[i]=s*(brpixel(col-1,row-1)+brpixel(col+1,row+1));
    sum[i]=sum[i]+brpixel(col,row);
    sum[i]=sum[i]+(1-s) * (brpixel(col-1,row)+brpixel(col+1,row));
    i++; row--;
    if (((col-y1-0.5)/(row-x1)) >=dx/dy) col++;
  }
  d0=sum[1]-sum[0]; d1=sum[2]-sum[1]; d2=sum[3]-sum[2];
  for (z=4; z<i; z++) {
    d3=sum[z]-sum[z-1];
    if (d1>limiet && m==0) {
      if (d1>d0 && d1>d2) {
        if (d2<limiet-1 || (d2>limiet-1 && d1<d3)) {
          h=(sum[z-4]-sum[z-3]*2+sum[z-2]) *64;
          h=h*(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
          x=(x1-z+3)*64; y=a*x+b*64;
          edgex[k]=x-h; edgey[k]=y+h*s;
          m=1; k++;
        }
      }
    }
  }
  return 0;
}
else if (dl<-1*limiet && m==1)
{
  if (dl<d0 && dl<=d2)
  {
    if (d2>limiet || (d2<limiet && dl<d3))
    {
      if (sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1] != 0)
      {
        h=(sum[z-4]-sum[z-3]*2+sum[z-2])*64;
        h=h/(sum[z-4]-sum[z-3]*3+sum[z-2]*3-sum[z-1]);
        x=(x1-z+3)*64; y=a*x+b*64;
        edgex[k]=x-h; edgey[k]=y+h*s;
        m=0; k++;
      }
    }
  }
  if(k==2) break;
  d0=d1; d1=d2; d2=d3;
}
edgex[k]=-1; edgey[k]=-1;

scanvert (x1,y1,x2,y2,limiet)
/* scanvert performs a scan in vertical direction, locates the
present edges with subpixel accuracy, and places the edge
coordinates in edgex[], and edgey[]. (x1,y1) and (x2,y2) are the
starting cq. ending coordinates of the scanline and limiet is the
value that must be exceeded by the largest gradient at an edge
profile. */

int x1,x2,y1,y2,limiet;
{
  int m,b,d0,d1,d2,d3,k,row,col;
  float f0,f1,f2,f3,f4,h;
  limiet=limiet*3; m=0; k=0; f0=f1=f2=f3=f4=0; col=y1;
  if (x1>x2) {b=x1; x1=x2; x2=b;}
  row=x1;
  f0=brpixel(col+1,row)+brpixel(col,row)+brpixel(col-1,row);
  f1=brpixel(col+1,row+1)+brpixel(col,row+1)+brpixel(col-1,row+1);
  f2=brpixel(col+1,row+2)+brpixel(col,row+2)+brpixel(col-1,row+2);
  f3=brpixel(col+1,row+3)+brpixel(col,row+3)+brpixel(col-1,row+3);
  d0=f1-f0; d1=f2-f1; d2=d3-f2;
  row++; row++; row++;
  while (row<x2)
  {
    row++; 
    f4=brpixel(col+1,row)+brpixel(col,row)+brpixel(col-1,row);
    d3=f4-f3;
    if ((f3-f2*3+f1*3-f0) != 0)
    {
      if ( (d1>=limiet && d1>d0 && d1>=d2 && m==0 )
{ if (d2<=limiet*-1 || (d2>=limiet*1 & & d1>=d3))
    edgex[k]=(row-3)*64-((f2+f0-fl*2)*64/(f3-f2*3+f1*3-f0));
    edgey[k]=yl*64;
    m=1; k++;
}

if ( (d1<=(limiet*1) & & d1<d0 & & d1<=d2 )
    if (d2>=limiet || (d2<=limiet & & d1<=d3))
        edgex[k]=(row-3)*64-((f2+f0-fl*2)*64)/(f3-f2*3+f1*3-f0);
        edgey[k]=yl*64;
        k++;}

if (k==2) break;
    f0=f1; f1=f2; f2=f3; f3=f4;
    d0=d1; d1=d2; d2=d3; d3=0;
}
edgex[k]=-1; edgey[k]=-1;
}

scanhor (xl,yl,x2,y2,limiet,width)
/* scanhor performs a scan in horizontal direction, locates the
   present edges with subpixel accuracy, and places the edge
   coordinates in edgex[], and edgey[]. (xl,yl) and (x2,y2) are the
   starting cq. ending coordinates of the scanline and limiet is the
   value that must be exceeded by the largest gradient at an edge
   profile. */

int xl,yl,x2,y2,limiet,width;
{
    int i,d0,d1,d2,d3,k,col,row;
    long int f0,f1,f2,f3,f4,h;
    limiet=limiet*width;xl=x1-width/2;k=0;f0=f1=f2=f3=f4=0;row=x1;
    if (yl>y2) (h=yl;yl=y2;y2=h;)
    col=yl;
    for (i=0;i<width;i++)
        {
            f0+=brpixel(col,xl+i);
            f1+=brpixel(col+1,xl+i);
            f2+=brpixel(col+2,xl+i);
            f3+=brpixel(col+3,xl+i);
        }
    d0=f1-f0; d1=f2-f1; d2=f3-f2;
    col++; col++; col++; while (col<y2)
    {
        col++;
        for (i=0;i<width;i++)
            {
                f4+=brpixel(col,xl+i);
\begin{verbatim}
} 
d3=f4-f3;
if (((f3-f2*3+f1*3-f0)!)=0)
{
    if ( (d1>=limiet && d1>d0 && d1>=d2 )
    { if (d2<=(limiet*-1) || (d2>=(limiet*-1)& d1>d3))

    { edgey[k]=(col-3)*64-(f2+f0-f1*2)*64/(f3-f2*3+f1*3-f0);
    edgex[k]=(x1+width/2)*64;
    k++;
    }
    }
else if ( d1<=(limiet*-1) && d1<d0 && d1<=d2 )
{ if ( (d2>=limiet || (d2<=limiet && d1>d3))
    { edgey[k]=(col-3)*64-(f2+f0-f1*2)*64/(f3-f2*3+f1*3-f0);
    edgex[k]=(x1+width/2)*64;
    k++;
    }
    }
}
}
f0=f1;f1=f2;f2=f3;f3=f4;f4=0;
d0=d1;di=d2;d2=d3;d3=0;

edgey[k]=-1;edgey[k]=-1;
}
\end{verbatim}
Appendix 2. Error in edge locating for all directions with the ruler technique.

1. Maximal error in edge locating for S4PwOv/s 0.00

2. Maximal error in edge locating for S4PwOv/s 0.01

3. Maximal error in edge locating for S4PwOv/s 0.05

4. Maximal error in edge locating for S4PwOv/s 0.10

5. Maximal error in edge locating for S4PwOv/s 0.15

6. Maximal error in edge locating for S4PwOv/s 0.20
Maximal error in edge locating in main direction versus slope of the scanline.

dat. 30-01-89

Maximal error in pixels

Slope $S = \frac{Dy}{Dx}$ of the scanline
Appendix 3: Adjustment of the light source of the telecentric system.

Measured width versus camera gain
Camera gain manually controlled.

dat. 17-03-89

Measured width in pixels

Established point

Relative step in camera gain
Appendix 4. Flowdiagram and corresponding software for defining the middle point, the radius and the image resolution for the shaverhead.

- Input 4 scandirections and starting lines.
- Define the light- to dark transition at the inside circle for every group.
- Transform the coordinates to a real circle equation.
- Locate the middle point of the circle.
- Calculate the average distance between the circle points and the centre.
- Transform the results back into pixel coordinates.
- Output centre location, x- and y-radius.
- Stop.
#include <math.h>
#include <itexpfg.h>
#include <stdtyp.h>
#include <stdio.h>
#include <conio.h>

long int xxc, yyc, rxx, ryy;
int xc, yc, rx, ry;
long int edge[512];
float rc[3], bc[3];
int res;

long int scanhormin (row, fromcol, tocol, limit, width)
{
    int -, dml, d0, dl, i, m, n, fml, f0, f1, f2;
    long h;
    short col, aintrow;
    aintrow=row-width/2; limit=limit*width;
    col=fromcol; c=0; fml=f0=f1=f2=0;
    for (i=0; i<width; i++)
    {
        fml+=brpixel (col, aintrow+i); f0+=brpixel (col+1, aintrow+i);
        f1+=brpixel (col+2, aintrow+i);
    }
    dml=f0-fml; d0=f1-f0;
    col+=2;
    while (col<tocol)
    {
        col++;
        for (i=0; i<width; i++)
        {
            f2+=brpixel (col, aintrow+i);
        }
        dl=f2-f1;
        if ((dl+dml-2*d0)! =0)
        {
            if (d0<=(limit*-1) && d0<dml && d0<=d1)
            {
                h=col-2;
                h=(h*64)-((d0-dml)*64)/(d1+dml-2*d0);
                edge[c]=h; c++;
            }
            f1=f2; f2=0; dml=d0; d0=d1; d1=0;
        }
        edge[c]=-1;
        if (c!=0) return (edge[c-1]);
        else return(0);
    }
}
long int scanvertmin (col,fromrow,torow,limit,width)

/* This function performs a scan in vertical direction from (fromrow,col) to (torow,col) and saves the coordinates of the last light- to dark edge found on the scanline. */
short col,fromrow,torow,limit,width;
int dml,d0,d1,i,m,n,c,fml,f0,f1,f2;
long h;
short aintcol;
aintcol=col-width/2;limit=limit*width;
c=0;fml=f0=f1=f2=0;
for (i=0;i<width;i++)
{
    fml+=brpixel(aintcol+i,fromrow);
    f0+=brpixel(aintcol+i,fromrow+1);
    f1+=brpixel(aintcol+i,fromrow+2);
}
dml=f0-fml;d0=f1-f0;
fromrow+=2;
while (fromrow<torow)
{
    fromrow++;
    for (i=0;i<width;i++)
    {
        f2+=brpixel(aintcol+i,fromrow);
    }
    d1=f2-f1;
    if ((d1+dml-2*d0)!=0)
    {
        if (d0<=(limit*-1) & & d0<dml & & d0<=d1)
        {
            h=fromrow-2;
            h=(h*64)-(f1+fml-f0*2)*64)/(f2-f1*3+f0*3-fml);
            edge[c]=h;c++;
        }
    }
    f1=f2;f2=0;dml=d0;d0=d1;d1=0;
}
edge[c]=-1;
if (c!=0) return(edge[c-1]);
else return(0);
}

long int scanhorplus (row,fromcol,tocol,limit,width)

/* This function performs a scan in horizontal direction from (row,fromcol) to (row,tocol) and saves the coordinates of the first dark- to light edge found on the scanline. */
short row,fromcol,tocol,limit,width;
int c,dml,d0,d1,i,m,n,c,fml,f0,f1,f2;
long h;
short col,aintrow;
aintrow=row-width/2;limit=limit*width;
col=fromcol;c=0;fml=f0=f1=f2=0;
for (i=0; i<width; i++)
{
    fml+=brpixel (col, aintrow+i); f0+=brpixel (col+1, aintrow+i);
    f1+=brpixel (col+2, aintrow+i);
}
dml=f0-fml; d0=f1-f0;
col++;
col++;
while (col<tocol)
{
    col++;
    for (i=0; i<width; i++)
    {
        f2+=brpixel (col, aintrow+i);
    }
    dl=f2-f1;
    if ((d1+dml-2*d0)! =0)
    {
        if ( d0>limit && d0>dml && d0>=dl)
        {
            h=col-2;
            h=(h*64)-((d0-dml)*64)/(d1+dml-2*d0);
            edge[c]=h; c++;
            break;
        }
    }
    f1=f2; f2=0; dml=d0; d0=d1; d1=0;
}
edge[c]=1;
if (c !=0) return (edge[0]);
else return (32000);
}

long int scanvertplus (col, fromrow, torow, limit, width)

/* This function performs a scan in vertical direction
   from (fromrow, col) to (torow, col) and saves the
   coordinates of the first dark- to light edge found on
   the scanline. */

short col, fromrow, torow, limit, width;
{
    int dml, d0, dl, i, m, n, c, fml, f0, f1, f2;
    long h;
    short aintcol;
    aintcol=col-width/2; limit=limit*width;
    c=0; fml=f0=f1=f2=0;
    for (i=0; i<width; i++)
    {
        fml+=brpixel (aintcol+i, fromrow);
        f0+=brpixel (aintcol+i, fromrow+1);
        f1+=brpixel (aintcol+i, fromrow+2);
    }
    dml=f0-fml; d0=f1-f0;
    fromrow+=2;
    while (fromrow<torow)
    {
Appendix 4.

```c
fromrow++;  
for (i=0;i<width;i++)  
{  
  f2+=brpixel(aintcol+i,fromrow);  
}  
d1=f2-f1;  
if (d1+dml-2*d0)! =0)  
{  
  if (d0>=limit && d0>dml && d0>=d1)  
  {  
    h=fromrow-2;  
    h=(h*64)-((f1+fml-f0*2)*64)/(f2-f1*3+f0*3-fml);  
    edge[c]=h;c++;  
    break;  
  }  
}  
fl=f2;f2=0;dml=d0;d0=d1;d1=0;  
}  
edge[c]=1;  
if (c!=0) return(edge[0]);  
else return (32000);  
}

void centr (xl,yl,x2,y2,x3,y3)  
/* The function fits a line 1 through (xl,yl) and (x2,y2)  
and a line m through (x2,y2) and (x3,y3). Then a line l'  
through the middle of 1, rectangular to 1, and a line m'  
through the middle of m, rectangular to m, are calculated.  
The crosspoint of l' and m' is determined. */  

long int xl,yl,x2,y2,x3,y3;  
{  
  if ((y2-yl)!=0 && (y3-y2)!=0)  
  {  
    rc[0]=(float)-(x2-xl)/(y2-yl);  
    bc[0]=(float) (yl+y2)/2-rc[0]*(x2+xl)/2;  
    if (y3-y2!=0)  
    rc[1]=(float)-(x3-x2)/(y3-y2);  
    bc[1]=(float) (y3+y2)/2-rc[1]*(x3+x2)/2;  
  }  
  xxc=(long int) (bc[1]-bc[0])/(rc[0]-rc[1]);  
  yyc=(long int) (rc[0]*xxc+bc[0]);  
}

long int max (a,b,c,d,e)  
/* The function returns the maximum value of a,b,c,d,e. */  
long int a,b,c,d,e;  
{  
  long int maxi;  
  if (a>b) maxi=a; else maxi=b;  
  if (c>maxi) maxi=c;if (d>maxi) maxi=d;  
  if (e>maxi) maxi=e;return(maxi);  
}  
```
long int min (a,b,c,d,e)

/* The function returns the minimum value of a,b,c,d,e. */
long int a,b,c,d,e;
{long int mini;
if (a<b) mini=a;else mini=b;
if (c<mini) mini=c;if (d<mini) mini=d;
if (e<mini) mini=e;return(mini);
}

long straal(x1,y1,x2,y2,x3,y3,x4,y4,a,b)

/* This function calculates the radius of a circle through
(x1,y1)...(x4,y4) with centrepoint (a,b)*/
long int x1,y1,x2,y2,x3,y3,x4,y4;
long a,b;
{
float d1,d2,d3,d4;
long d;

 d1=sqrt((float) (x1-a) * (x1-a) + (y1-b) * (y1-b));
d2=sqrt((float) (x2-a) * (x2-a) + (y2-b) * (y2-b));
d3=sqrt((float) (x3-a) * (x3-a) + (y3-b) * (y3-b));
d4=sqrt((float) (x4-a) * (x4-a) + (y4-b) * (y4-b));
d= (long) (d1+d2+d3+d4+2)/4;
return (d);
}

void middeldia()

/* The function determines the middlepoint and the radius
and the image resolution of the shaverhead*/
{int point[3];
 long int a,b,c,d,e,x1,x2,x3,x4,y1,y2,y3,y4;
 long f,k,l;
a=scanhormin(250,75,200,50,3);b=scanhormin(252,75,200,50,3);
c=scanhormin(254,75,200,50,3);d=scanhormin(256,75,200,50,3);
e=scanhormin(258,75,200,50,3);
y1=max(a,b,c,d,e);
if (y1==a) x1=250;if (y1==b) x1=252;if (y1==c) x1=254;
if (y1==d) x1=256;if (y1==e) x1=258;
x1=x1*64;
a=scanvertmin(250,50,200,50,3);b=scanvertmin(252,50,200,50,3);
c=scanvertmin(254,50,200,50,3);d=scanvertmin(256,50,200,50,3);
e=scanvertmin(258,50,200,50,3);
x2=max(a,b,c,d,e);
if (x2==a) y2=250;if (x2==b) y2=252;if (x2==c) y2=254;
if (x2==d) y2=256;if (x2==e) y2=258;
y2=y2*64;
a=scanvertplus(250,300,500,50,3);
b=scanvertplus(252,300,500,50,3);
c=scanvertplus(254,300,500,50,3);
d=scanvertplus(256,300,500,50,3);
e=scanvertplus(258,300,500,50,3);
x3=min(a,b,c,d,e);
if (x3==a) y3=250;if (x3==b) y3=252;if (x3==c) y3=254;
if (x3==d) y3=256;if (x3==e) y3=258;
y3=y3*64;
a=scanhorplus(250,300,500,50,3);b=scanhorplus(252,300,500,50,3);
c=scanhorplus(254,300,500,50,3);d=scanhorplus(256,300,500,50,3);
e=scanhorplus(258,300,500,50,3);
y4=min(a,b,c,d,e);
if (y4==a) x4=250;if (y4==b) x4=252;if (y4==c) x4=254;
if (y4==d) x4=256;if (y4==e) x4=258;
x4=x4*64;
x1=(x1*2)/3;x2=(x2*2)/3;x3=(x3*2)/3;x4=(x4*2)/3;
centr(x2,y2,x1,y1,x4,y4);
k=xxc;l=yyc;
centr(x3,y3,x1,y1,x4,y4);
k=k+xxc;l=l+yyc;
xxc=((k+1)*3+128)/(64*4);yyc=(l+1+64)/128;
f=straal(x1,y1,x2,y2,x3,y3,x4,y4,k/2,1/2);
rxx=(f*3)/128.0;ryy=(f/64.0);
xc=(int)xxc;yc=(int)yyc;
res=15.43*64*1000/(2*f+32);
Appendix 5. Flowdiagram and corresponding software for the optical inspection of the width of the 90 gaps of a shaverhead.

1. Input x,y-coordinates of the middle point and the radius of the shaverhead.

2. Compute the starting coordinates and scan direction of a set of three scan lines for every gap.

3. Perform per gap a scan along three parallel lines for dark to light and succeeding light to dark transitions.

4. Fit a line 1 through 2 exterior points on the dark to light edge and compute the distance between 1 and the middle light to dark transition in pixels.

5. Transform the computed distance into microns

Stop.
#include <math.h>
#include <itexpfg.h>
#include <stdtype.h>
#include <stdio.h>
#include <conio.h>
long int xxc, yyc, rxx, ryy;
int xc, yc, rx, ry;
int edgex[512], edgy[512];
int pntx[3][512];
int pnty[3][512];
tellerv;
float rc[3], bc[3];
float xcoor, ycoor;
int res;

inspect1(xc, rx, yc, ry)
/* Measuring the width of the gaps in the first quadrant of the
shavercap in which 0 <= the angle between the y-axis and the
gap <= 45 degrees. */
int xc, rx, yc, ry;
{
    int c, z, a, b, i;
    double alfa;
    alfa = 0.0698131; c = 1; pnty[1][1] = 64*(yc-10);
    for (z = 0; z < 17; z++)
    {
        rx = sin((double)1.5707963 - alfa)*rxx;
        rx = (int)rx;
        a = edgexh(xc-rx-30+2*(z/5), (int)pnty[1][2*c-1]/64, xc-rx-30+2*(z/5), (int)pnty[1][2*c-1]/64+20, 60-2*z);
        for (i = 0; i <= 2*a-1; i++)
        {
            pntx[0][i] = edgex[i]; pnty[0][i] = edgy[i];
        }
        b = edgexh(xc-rx-20+z/5, (int)edgy[0]/64-5-2*(z/5), xc-rx-20+z/5, (int)edgy[2*a-1]/64+5, 60-2*z);
        for (i = 0; i <= 2*b-1; i++)
        {
            pntx[1][i] = edgex[i]; pnty[1][i] = edgy[i];
        }
        c = edgexh(xc-rx-10, (int)edgy[0]/64-5-2*(z/4), xc-rx-10, (int)edgy[2*b-1]/64+5, 60-2*z);
        for (i = 0; i <= 2*c-1; i++)
        {
            pntx[2][i] = edgex[i]; pnty[2][i] = edgy[i];
        }
        totaal(pntx[0][0], pnty[0][0], pntx[2][0], pnty[2][0], pntx[0][1], pnty[0][1]);
        totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[2][1], pnty[2][1]);
        totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[1][1], pnty[1][1]);
    }
}
inspect2(xc, rx, yc, ry)
/* Measuring the width of the gaps in the second quadrant of the
shavercap in which 0 <= the angle between the y-axis and
the gap <=45 degrees.*/
int xc, rx, yc, ry;
{
int c, z, a, b, i;
double alfa;
alfa = 0.0698131; c = 1; pnty[1][0] = 64*yc;
for (z = 0; z < 13; z++)
{
  rx = sin((double)1.5707963 - z * alfa) * rxx;
  rx = (int)rx;
a = edgelh(xc - rx - 30 + (z / 3), (int)pnty[1][0] / 64 - 20, xc - rx - 30 + (z / 3),
  (int)pnty[1][0] / 64, 60 - 2*z);
  for (i = 0; i <= 2*a - 1; i++) {
    pntx[0][i] = edgex[i]; pnty[0][i] = edgy[i];
  }
  b = edgelh(xc - rx - 20, (int)edgey[0] / 64 - 2, xc - rx - 20, (int)edgey[2*a - 1] -
  (int)edgey[2*a - 1] / 64 + 7 + 2*(z / 3), 60 - 2 * z);
  for (i = 0; i <= 2*b - 1; i++) {
    pntx[1][i] = edgex[i]; pnty[1][i] = edgy[i];
  }
  c = edgelh(xc - rx - 10 - (z / 3), (int)edgey[0] / 64 - 2, xc - rx - 10 - (z / 3), (int)
  edgey[2*b - 1] / 64 + 7 + 2*(z / 3), 60 - 2 * z);
  for (i = 0; i <= 2*c - 1; i++) {
    pntx[2][i] = edgex[i]; pnty[2][i] = edgy[i];
  }
  totaal(pntx[0][0], pnty[0][0], pnty[2][0], pnty[2][0], pntx[0][1], pnty[1][1]);
  totaal(pntx[2][0], pnty[2][0], pnty[0][0], pnty[0][0], pntx[2][1], pnty[2][1]);
  totaal(pntx[2][0], pnty[2][0], pnty[0][0], pnty[0][0], pntx[1][1], pnty[1][1]);
}
}

inspect3(xc, rx, yc, ry)
/* Measuring the width of the gaps in the fourth quadrant of the
shavercap in which 0 <= the angle between the y-axis and the
gap <=45 degrees.*/
int xc, rx, yc, ry;
{
int c, z, a, b, i;
double alfa;
alfa = 0.0698131; c = 1; pnty[1][1] = 64*(yc - 10);
for (z = 0; z < 11; z++)
{
  rx = sin((double)1.5707963 - z * alfa) * rxx;
  rx = (int)rx;
a = edgelh(xc + rx + 30 - (z / 3), (int)pnty[1][1] / 64, xc + rx + 30 - (z / 3),
  (int)pnty[1][1] / 64 + 20, 50 - 2 * z);
  for (i = 0; i <= 2*a - 1; i++) {
  }
pntx[0][i]=edgex[i]; pnty[0][i]=edgey[i];
} b=edgelh (xc+rx+20, (int)edgey[0]/64-7-2*(z/5), xc+rx+20, (int)edgey[2*a-1]/64+5, 50-2*z);
for (i=0; i<=2*b-1; i++) {
    pntx[1][i]=edgex[i]; pnty[1][i]=edgey[i];
}
c=edgelh (xc+rx+10+(z/3), (int)edgey[0]/64-7-2*(z/4), xc+rx+10+(z/-3), (int)edgey[2*b-1]/64+5, 50-2*z);
for (i=0; i<=2*c-1; i++) {
    pntx[2][i]=edgex[i]; pnty[2][i]=edgey[i];
}

totaal (pntx[0][0], pnty[0][0], pntx[2][0], pnty[2][0], pntx[0][1], pnty[0][1]);
totaal (pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[1][1], pnty[1][1]);
}

inspect4(xc, rx, yc, ry)
/* Measuring the width of the gaps in the third quadrant of the
shavercap in which 0 <= the angle between the y-axis and the
gap <= 45 degrees.*/
int xc, rx, yc, ry;
{
    int c, z, a, b, i;
    double alfa;
    alfa=0.0698131; c=1; pnty[1][0]=64*yc;
    for (z=1; z<17; z++)
        { 
            rx=sin((double)1.5707963-z*alfa)*rxx;
            rx=(int)rx;
            a=edgelh (xc+rx+30-2*(z/5), (int)pnty[1][0]/64-15, xc+rx+30-2*(z/5) -
                       (int)pnty[1][0]/64+5, 60-2*z);
            for (i=0; i<=2*a-1; i++) {
                pntx[0][i]=edgex[i]; pnty[0][i]=edgey[i];
            }
        b=edgelh (xc+rx+20- (z/5), (int)edgey[0]/64-2, xc+rx+20- (z/5), (int) -
           edgey[2*a-1]/64+7+2*(z/3), 60-2*z);
            for (i=0; i<=2*b-1; i++) {
                pntx[1][i]=edgex[i]; pnty[1][i]=edgey[i];
            }
        c=edgelh (xc+rx+10, (int)edgey[0]/64-2, xc+rx+10, (int)edgey[2*b-1] -
                   /64+7+2*(z/3), 60-2*z);
            for (i=0; i<=2*c-1; i++) {
                pntx[2][i]=edgex[i]; pnty[2][i]=edgey[i];
            }
        totaal (pntx[0][0], pnty[0][0], pntx[2][0], pnty[2][0], pntx[0][1], pnty[0][1]);
        totaal (pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[2][1], pnty[2][1]);
}
inspektor(xc,rx,yc,ry)
/* Measuring the width of the gaps in the fourth quadrant of the shaver cap in which 0 <= the angle between the x-axis and the gap <=45 degrees.*/
int xc,rx,yc,ry;
{
    int c,z,a,b,i;
    double alfa;
    alfa=0.0698131;c=1;pntx[1][1]=64*(xc-15);
    for (z=1;z<13;z++)
    {
        ry=sin((double)1.5707963-z*alfa)*ryy;
        ry=(int)ry;
        a=edgelv((int)pntx[1][1]/64,yc+ry+30,(int)pntx[1][1]/64+30,60-2*z);
        for (i=0;i<=2*a-1;i++) {
            pntx[0][i]=edgex[i];pnty[0][i]=edgey[i];
        }
        b=edgelv((int)edgex[0]/64-7-2*(z/3),yc+ry+22+(z/5),(int)edgex[2-a-1]/64+8,yc+ry+22+(z/5),60-2*z);
        for (i=0;i<=2*b-1;i++) {
            pntx[1][i]=edgex[i];pnty[1][i]=edgey[i];
        }
        c=edgelv((int)edgex[0]/64-7-2*(z/3),yc+ry+14+2*(z/5),(int)edgex[-2*b-1]/64+8+(z/5),yc+ry+14+2*(z/5),60-2*z);
        for (i=0;i<=2*c-1;i++) {
            pntx[2][i]=edgex[i];pnty[2][i]=edgey[i];
        }
        totaal(pntx[0][0],pnty[0][0],pntx[2][0],pnty[2][0],pntx[0][1],pnty[0][1]);
        totaal(pntx[2][0],pnty[2][0],pntx[0][0],pnty[0][0],pntx[2][1],pnty[2][1]);
        totaal(pntx[2][0],pnty[2][0],pntx[0][0],pnty[0][0],pntx[1][1],pnty[1][1]);
    }
}

inspektor(xc,rx,yc,ry)
/* Measuring the width of the gaps in the third quadrant of the shaver cap in which 0 <= the angle between the x-axis and the gap <=45 degrees.*/
int xc,rx,yc,ry;
{
    int c,z,a,b,i;
    double alfa;
    alfa=0.0698131;c=1;pntx[1][1]=64*xc;
for (z=1;z<7;z++)
{
    ry=sin((double)1.5707963-z*alfa)*ryy;
    ry=(int)ry;
    a=edgelv((int)pntx[1][1]/64,yc-ry-30,(int)pntx[1][1]/64+30,yc-ry-
            30,50-2*z);
    for (i=0;i<=2*a-1;i++)
    {
        pntx[0][i]=edgex[i];pnty[0][i]=edgey[i];
    }
    b=edgelv((int)edgex[0]/64-10-3*(z/3),yc-ry-22-(z/5),(int)edgex[2-
                  a-1]/64+3,yc-ry-22-(z/5),50-2*z);
    for (i=0;i<=2*b-1;i++)
    {
        pntx[1][i]=edgex[i];pnty[1][i]=edgey[i];
    }
    c=edgelv((int)edgex[0]/64-10-3*(z/3),yc-ry-14-2*(z/5),(int)edgex-
              [2*b-1]/64+3,yc-ry-14-2*(z/5),50-2*z);
    for (i=0;i<=2*c-1;i++)
    {
        pntx[2][i]=edgex[i];pnty[2][i]=edgey[i];
    }
    totaal(pntx[0][0],pnty[0][0],pntx[2][0],pnty[2][0],pntx[0][1],pn-
    ty[0][1]);
    totaal(pntx[2][0],pnty[2][0],pntx[0][0],pnty[0][0],pntx[2][1],pn-
    ty[2][1]);
    totaal(pntx[2][0],pnty[2][0],pntx[0][0],pnty[0][0],pntx[1][1],pn-
    ty[1][1]);
}

inspect7(xc,rx,yc,ry)
/* Measuring the width of the gaps in the second quadrant of the
shavercap in which 0 <= the angle between the x-axis and the
gap <=45 degrees.*/
int xc,rx,yc,ry;
{
    int c,z,a,b,i;
    double alfa;
    alfa=0.0698131;c=1;pntx[1][0]=64*(xc+15);
    for (z=0;z<10;z++)
    {
        ry=sin((double)1.5707963-z*alfa)*ryy;
        ry=(int)ry;
        a=edgelv((int)pntx[1][0]/64-25,yc-ry-30+3*(z/5),(int)pntx[1][0]/-
                64+5,yc-ry-30+3*(z/5),50-2*z);
        for (i=0;i<=2*a-1;i++)
        {
            pntx[0][i]=edgex[i];pnty[0][i]=edgey[i];
        }
        b=edgelv((int)edgex[0]/64-5,yc-ry-22+2*(z/5),(int)edgex[2*a-1]/6-
                  4+3+3*(z/3),yc-ry-22+2*(z/5),50-2*z);
        for (i=0;i<=2*b-1;i++)
        {
            pntx[1][i]=edgex[i];pnty[1][i]=edgey[i];
        }
        c=edgelv((int)edgex[0]/64-5,yc-ry-14+(z/5),(int)edgex[2*b-1]/64+
\[3+3(z/3), yc-ry-14+(z/5), 50-2*z;\]

for \(i=0; i<2*c-1; i++\) {
    pntx[2][i]=edgex[i]; pnty[2][i]=edgey[i];
}

totaal(pntx[0][0], pnty[0][0], pntx[2][0], pnty[2][0], pntx[0][1], pnty[0][1]);
totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[2][1], pnty[2][1]);
totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[1][1], pnty[1][1]);
}

inspect8(xc, rx, yc, ry)
/* Measuring the width of the gaps in the first quadrant of the
    shearcap in which 0 <= the angle between the x-axis and the
gap <=45 degrees. */

int xc, rx, yc, ry;
{
    int c, z, a, b, i;
    double alfa;
    alfa=0.0698131; c=1; pntx[1][0]=64*(xc);
    for (z=1; z<=c; z++)
    { ry=sin((double)1.5707963-z*alfa)*ryy;
        ry=(int)ry;
        a=edgelv((int)pntx[1][0]/64-25, yc+ry+30-3*(z/5), (int)pntx[1][0]/
            64+5, yc+ry+30-3*(z/5), 40-2*z);
        for (i=0; i<=2*a-1; i++)
            pntx[0][i]=edgex[i]; pnty[0][i]=edgey[i];
    }
    b=edgelv((int)edgex[0]/64-5, yc+ry+22-2*(z/5), (int)edgex[2*a-1]/6-
            4+6+3*(z/3), yc+ry+22-2*(z/5), 40-2*z);
    for (i=0; i<=2*b-1; i++)
        pntx[1][i]=edgex[i]; pnty[1][i]=edgey[i];
    c=edgelv((int)edgex[0]/64-3, yc+ry+14-(z/5), (int)edgex[2*b-1]/64+
            8+3*(z/3), yc+ry+14-(z/5), 40-2*z);
    for (i=0; i<=2*c-1; i++)
        pntx[2][i]=edgex[i]; pnty[2][i]=edgey[i];
}

totaal(pntx[0][0], pnty[0][0], pntx[2][0], pnty[2][0], pntx[0][1], pnty[0][1]);
totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[2][1], pnty[2][1]);
totaal(pntx[2][0], pnty[2][0], pntx[0][0], pnty[0][0], pntx[1][1], pnty[1][1]);
}
totaal(xxl,yy1,xx2,yy2,xx3,yy3)
/* The function totaal fits a line through the points
(\(xx1,yy1\)) and (\(xx2,yy2\)) and calculates the distance
between this line and a point (\(xx3,yy3\)). This distance
is expressed in pixels \(\times\) resolution.*/
int xx1,yy1,xx2,yy2,xx3,yy3;
{
float a0,dist,b0,a1,b1,x1,y1,x2,y2,x3,y3,xc,yc;
x1=(float)xx1*0.67; x2=(float)xx2*0.67; x3=(float)xx3*0.67;
y1=(float)yy1; y2=(float)yy2; y3=(float)yy3;
if ((x1-x2)==0) dist=(float) (x3-x2)/64.0;
else if ((y1-y2)==0)
dist=(float) (y3-y2)/64.0;
else
{
a0=(float) (y1-y2)/(x1-x2); b0=(float) (y1+y2)/2-a0*(x1+x2)/2;
a1=(float)-1.0/a0; b1=(float)y3-a1*x3;
xc=(b1-b0)/(a0-a1); yc=a0*xc+b0;
dist=(float)sqrt((x3-xc)*(x3-xc)+(y3-yc)*(y3-yc))/64.0;
}
}

float holdh(row,fromcol,tocol)
/* holdh computes the threshold value for a scan in horizontal
 direction of a 5*6 pixel area */
int row,fromcol,tocol;
{
int i,j,ma,mi,max[3],min[3],a;
float thresh;
for (j=row-2;j<=row+2;j++) {
for (i=fromcol;i<=tocol;i++) {
    a=brpixel(i,j);
    if (a>max[2]) { max[0]=max[1];max[1]=max[2];max[2]=a; }
    else if (a>max[1]) { max[0]=max[1];max[1]=a; }
    else if (a>max[0]) max[0]=a;
    if (a<min[2]) { min[0]=min[1];min[1]=min[2];min[2]=a; }
    else if (a<min[1]) { min[0]=min[1];min[1]=a; }
    else if (a<min[0]) min[0]=a;
}
    thresh=(max[2]+min[2]);thresh=(max[1]+min[1]);thresh=thresh/2.0;
return ((float)thresh);
}

float holdv(col,fromrow,torow)
/* holdv computes the threshold value for a scan in vertical
direction of a 6*5 pixel area*/
int col, fromrow, torow;
{
    int i, j, ma, mi, max[3], min[3], a;
    float thresh;
    for (j=col-2; j<=col+2; j++) {
        for (i=fromrow; i<=torow; i++) {
            a=brpixel(j, i);
            if (a>=max[2]) { max[0]=max[1]; max[1]=max[2]; max[2]=a; }
            else if (a>=max[1]) { max[0]=max[1]; max[1]=a; }
            else if (a>=max[0]) max[0]=a;
            if (a<=min[2]) { min[0]=min[1]; min[1]=min[2]; min[2]=a; }
            else if (a<=min[1]) { min[0]=min[1]; min[1]=a; }
            else if (a<=min[0]) min[0]=a;
        }
    }
    thresh=(max[2]+min[2]); thresh=(max[1]+min[1]); thresh=thresh/2.0;
    return ((float)thresh);
}

int edgelh(fromrow, fromcol, torow, tocol, limit)
/*edgelh performs a scan in horizontal direction and computes
edge points with the edgel technique */
int fromrow, fromcol, torow, tocol, limit;
{
    int row, m, k, t, a, b, c, col, x1, d0, d1, d2, f0, f1, f2, f3, i, j, e;
    float f, h;
    tellerv=0; limit=limit*3; row=fromrow; k=0; m=0; x1=row-1;
    col=fromcol; f0=f1=f2=f3=0;
    for (i=0; i<3; i++) {
        f0+=brpixel(col, x1+i);
        f1+=brpixel(col+1, x1+i);
        f2+=brpixel(col+2, x1+i);
    }
    d0=f1-f0; d1=f2-f1;
    for (col=fromcol+3; col<tocol+col+1) {
        for (i=0; i<3; i++) {
            f3+=brpixel(col, x1+i);
        }
        d2=f3-f2;
        if (d1>limit & & d1>d0 & & d1>d2 & & m==0)
            {
                e=col-2;
                h=(float)holdh(row, e-2, e+3);
                tellerv=0;
                for (i=e-1; i<=e+1; i++) {
int edgelv(fromrow,fromcol,torow,tocol,limit)
/* edgelv performs a scan in vertical direction and computes edge points with the edgel technique */
int fromrow,fromcol,torow,tocol,limit;
{
    int row,m,k,t,a,b,c,col,xl,d0,d1,d2,f0,f1,f2,f3,i,j,e;
    float f,h;
    tellerv=0;limit=limit*3;col=fromcol;k=0;m=0;xl=col-1;
    row=fromrow;f0=f1=f2=f3=0;
    for (i=0;i<3;i++)
    {
        f0+=brpixel(xl+i,row);
        f1+=brpixel(xl+i,row+1);
        f2+=brpixel(xl+i,row+2);
    }
    d0=f1-f0;d1=f2-f1;
    for (row=fromrow+3;row<torow;row++)
    {
        for (i=0;i<3;i++)
        {
            f3+=brpixel(xl+i,row);
        }
        d2=f3-f2;
        if (d1>limit && d1>=d0 && d1>d2 && m==0)
Appendix 5.

```c

e=row-2;
h=(float)holdv(col,e-2,e+3);
t=tellerv=0;
for (i=e-1;i<=e+1;i++) {
    j=col;
    if (t==tellerv){
        a=sx(j,i); b=sy(j,i); c=sz(j,i);
        line(a,b,c,h,i,j);
    }
}
if (tellerv==1) { edgex[k]=xcoor*64;edgey[k]=ycoor*64;k++;m=1;tellerv=0;
}
else if ( d1<limit*-1 && d1<=d0 && d1<d2 && m==1 ) {
    e=row-2;
h=(float)holdv(col,e-2,e+3);
t=tellerv=0;
for (i=e-1;i<=e+1;i++) {
    j=col;
    if (t==tellerv){
        a=sx(j,i); b=sy(j,i); c=sz(j,i);
        line(a,b,c,h,i,j);
    }
}
if (tellerv==1) { edgex[k]=xcoor*64;edgey[k]=ycoor*64;k++;m=0;tellerv=0;
return(1);break;}
}
d0=d1;d1=d2;d2=0;f0=f1;f1=f2;f2=f3;f3=0;
}

line (sx,sy,sz,thr,row,col)
/* line computes the coordinates of the middle point of the
   largest edgel along a scan column or a scan row at an edge. */
int sx,sy,sz, row, col;
float thr;
{
    float x, y;
    float a, b, c;
    a=0;
xcoor=0;ycoor=0;
    if (sy!=0) { x=-0.5;
y=(((16*thr-sz)/2) -sx*x)/sy;
    if (y>=-0.5 && y<0.5) { xcoor+=row+x; ycoor+=col+y; a++; }
x=0.5;
y=(((16*thr-sz)/2) -sx*x)/sy;
    if (y>=-0.5 && y<0.5) { xcoor+=row+x; ycoor+=col+y; a++; }
}
if (sx!=0){
```

---

The code snippet provided calculates the coordinates of the middle point of the largest edgel along a scan column or a scan row at an edge. The function `line` computes these coordinates, and the main loop iterates over possible edge locations within a specified limit. The variables used include `row`, `col`, `sx`, `sy`, `sz`, `thr`, `xcoor`, and `ycoor`, among others, to store and calculate positions and thresholds.
\[y = -0.5;\]
\[x = \left(\frac{16 \times \text{thr} - \text{sz}}{2}\right) - \frac{\text{sy} \times y}{\text{sx}};\]
\[
\text{if } (x >= -0.5 \&\& x <= 0.5) \{ \text{ycoor} = \text{col} + y; \text{xcoor} = \text{row} + x; \\
\quad \text{a}++; \}
\]
\[y = 0.5;\]
\[x = \left(\frac{16 \times \text{thr} - \text{sz}}{2}\right) - \frac{\text{sy} \times y}{\text{sx}};\]
\[
\text{if } (x >= -0.5 \&\& x <= 0.5) \{ \text{ycoor} = \text{col} + y; \text{xcoor} = \text{row} + x; \\
\quad \text{a}++; \}
\]
\[
\text{if } (\text{a} == 2) \{ \\
\quad \text{b} = (\text{xcoor} - \text{a} \times \text{row})/\text{a}; \text{c} = (\text{ycoor} - \text{a} \times \text{col})/\text{a}; \\
\quad \text{if } ((-0.25 < \text{b} \&\& \text{b} <= 0.25) \quad || \quad (-0.25 < \text{c} \&\& \text{c} <= 0.25)) \{ \\
\quad \quad \text{xcoor} = \text{xcoor}/\text{a}; \text{ycoor} = \text{ycoor}/\text{a}; \text{tellerv}++; \text{a} = 0; \\
\quad \}
\}
\]

\textbf{int sy(col,row)}

/* sy computes the y-coefficient of the best fit plane*/
\textbf{int} col,row;
\{
\indent \textbf{int} sy,x,y;
\indent x=row;y=col;sy=0;
\indent sy+=\text{brpixel}(y+1,x-1);sy+=2*\text{brpixel}(y+1,x);sy+=\text{brpixel}(y+1,x+1);
\indent sy-=\text{brpixel}(y-1,x-1);sy-=2*\text{brpixel}(y-1,x);sy-=\text{brpixel}(y-1,x+1);
\indent \text{return } (\text{sy});
\}

\textbf{int sx(col,row)}

/* sx computes the x-coefficient of the best fit plane */
\textbf{int} col,row;
\{
\indent \textbf{int} sx,x,y;
\indent x=row;y=col;sx=0;
\indent sx+=\text{brpixel}(y+1,x-1);sx+=2*\text{brpixel}(y,x-1);sx+=\text{brpixel}(y+1,x-1);
\indent sx-=\text{brpixel}(y-1,x+1);sx-=2*\text{brpixel}(y,x+1);sx-=\text{brpixel}(y-1,x+1);
\indent \text{return } (\text{sx});
\}

\textbf{int sz(col,row)}

/* sz computes the z-coefficient of the best fit plane */
\textbf{int} col,row;
\{
\indent \textbf{int} sz,x,y;
\indent x=row;y=col;sz=0;
\indent sz+=\text{brpixel}(y+1,x-1);sz+=\text{brpixel}(y-1,x+1);sz+=\text{brpixel}(y+1,x-1);
\indent sz+=\text{brpixel}(y-1,x-1);sz+=2*\text{brpixel}(y,x);sz+=2*\text{brpixel}(y-1,x-1);
\indent sz+=2*\text{brpixel}(y,x+1);sz+=2*\text{brpixel}(y+1,x);sz+=4*\text{brpixel}(y,x);
\indent \text{return } (\text{sz});
\}
main()
{
    int k, a, b, c, i, j, z;
    double alfa;
    alfa=0.0698131;
    ITEX_initialize();
    getimage();
    middeldia();
    xc=(int)x=x; yc=(int)y=y;
    rx=(int)x+x+20; ry=(int)y+y+10;
    inspect1(xc,rx,yc,ry); inspect2(xc,rx,yc,ry);
    inspect3(xc,rx,yc,ry); inspect4(xc,rx,yc,ry);
    inspect5(xc,rx,yc,ry); inspect6(xc,rx,yc,ry);
    inspect7(xc,rx,yc,ry); inspect8(xc,rx,yc,ry);
}
Appendix 6. Distribution of the distances between the control lines after correction with four different correction methods.

Calibration with respect to $Y=255$

Calibration with respect to $Y=0$

Calibration with respect to $Y=255$

Calibration with respect to $Y=0$
Appendix 7: Flowdiagram and corresponding software for the correction of image distortion in horizontal (vertical) direction.

1. Compute the coordinates of edges of the draught board with parallel equidistant horizontal (vertical) scan lines.

2. Define the mean distance between two succeeding edges on one scanline = $\Delta x$.

3. Compute per edge point the error in horizontal (vertical) direction = $f_x$ ($f_y$).

4. Fit a correction polynomial through the error data with the least squared error method.

Stop.

#include <stdio.h>
#include <conio.h>
#include <itexpfg.h>
define mmax 600
#define nmax 10
float edge[512];
float A[500][10];
float B[500];
float F[500];
float E[500][2];
float H[10];
float V[10];

init ()
/* init initializes the coefficients which have to be used for
   computing the correction polynomial */
{
    int i, j;
    for (i=0; i<10; i++)
    {
        for (j=0; j<500; j++) { A[j][i]=0; }
    }
    for (j=0; j<500; j++) B[j]=F[j]=0;
    for (i=0; i<10; i++) { H[i]=V[i]=0; }
    for (i=0; i<2; i++)
    {
        for (j=0; j<500; j++) { E[j][i]=0; }
    }
}

void polyfit(a, b, y, m, n)
/* polyfit solves m equations with n unknowns A * y = b; destroys
   A, b */
float a[mmax][nmax], b[], y[];
short m, n;
{ short q, z, i, j, k;
  double h, dk;
  float r[nmax][nmax];
  float eps;
  for (k = 0; k < n; k++) /* step k*/
  {
    dk = 0.0;
    for (i = 0; i < m; i++) dk += a[i][k] * a[i][k];
    for (j = k+1; j < n; j++)
    {
      h = 0.0;
      for (i = 0; i < m; i++) h += a[i][k] * a[i][j];
      h /= dk; r[k][j] = h;
      for (i = 0; i < m; i++) a[i][j] -= h*a[i][k];
    }
    h = 0.0;
    for (i = 0; i < m; i++) h += a[i][k] * b[i];
    h /= dk; y[k] = h;
    for (i = 0; i < m; i++) b[i] -= h * a[i][k];
  }
  for ( i = n-2; i >= 0; i--)
  { for (k=i+1; k < n; k++) y[i] -= r[i][k] * y[k];
  }
for (i=0;i<n;i++) {
    printf("%f\t",y[i]);
}

eps = 0.0;
for (i = 0; i < m; i++) eps += b[i] * b[i];
printf("eps = %f\n",eps);

ijkhor()
/* ijkhor measures the distortion in horizontal direction by
scanning for edges at the calibration target with parallel
equidistant horizontal scan lines and fits a third degree
polynomial through the correction data */
{
    int limit,b,totaal,x,x11,x12,x1, dx,c,1,k,a,z;
    float totaal,f, fact, xx1, xx2, e, corr;
    float postmax, postmin, premax, premin;
    postmax=preamx=0;
    postmin=preamin=512;
    movecursori();
    printf("x-coordinaten e\ seab horizontale scanlijn = ?\n")
    scanf("%d",&x1);
    printf("begincoordinaten eerste horizontale scanlijn = ?\n")
    scanf("%d",&x11);
    printf("eindcoordinaten eerste horizontale scanlijn = ?\n")
    scanf("%d",&x12);
    printf("stap in x-richting = ?\n")
    scanf("%d",&dx);
    printf("threshold =?\n")
    scanf("%d",&limit);
    totaal=0;
    for (x=x1;x<=450;x+=dx)
    {
        b=4;
        for (l=0;l<b;l++)
        getimage();
        c=1;
        a=edgelh(x,x11,x,x12,limit);
        k=totaal;
        for(c=2;c<a;c+=2)
        {
            e=edgelh(c)/64.0;
            A[k][0]=1; A[k][1]=+(float)x; A[k][2]=+e; A[k][3]=+(float)x *(float)x;
            A[k][4]=+e*e; A[k][5]=+((float)x*(float)x*(float)x); A[k][6]=+e*e*e;
            A[k][7]=+((float)x*(float)x*e); A[k][8]=+e*(float)x; A[k][9]=+e*(float)x;
            B[k]+=((edge[c]-edge[c-2])/64.0;
            F[k]=B[k];
            k++;
        }
    }
    for (z=totaal; z<k; z++)
\[
A[z]_0 = A[z][0]/b; \ A[z]_1 = A[z][1]/b; \ A[z]_2 = A[z][2]/b;
A[z]_3 = A[z][3]/b; \ A[z]_4 = A[z][4]/b; \ A[z]_5 = A[z][5]/b;
A[z]_6 = A[z][6]/b; \ A[z]_7 = A[z][7]/b; \ A[z]_8 = A[z][8]/b;
A[z]_9 = A[z][9]/b; \ B[z] = B[z]/b;
\]
\[
F[z] = F[z]/b;
\]
if (B[z] > premax) premax = B[z];
if (B[z] < premin && B[z] > 0) premin = B[z];
totaal = k;
movecursori();
hlclear(0, x, 512, 255);

\[
\begin{align*}
&\text{fact} = 0; \\
&\text{for} (z = 0; z < \text{totaal}; z++) \text{ fact} += B[z]; \\
&\text{fact} = \text{fact}/\text{totaal}; \\
&\text{printf}("\text{gemiddelde afstand = } %6.3f \text{n", fact); \\
&\text{corr} = 0; \\
&\text{totaalf} = 0; \\
&\text{for} (z = 0; z < \text{totaal}; z++) \\
&\quad \text{if} (z != 0) f = A[z-1][2]; \\
&\quad \text{else} f = A[z][2]; \\
&\quad \text{if} (A[z][2] > f) \\
&\quad\quad \text{corr} += B[z] - \text{fact}; \\
&\quad\quad \text{totaalf} += B[z] - \text{fact}; \\
&\quad \text{else} \\
&\quad\quad \text{corr} = (A[z][2] - x11)/B[z] * (B[z] - \text{fact}); \\
&\quad\quad \text{totaalf} += (A[z][2] - x11)/B[z] * (B[z] - \text{fact}); \\
&\quad B[z] = \text{corr}; \\
&\quad E[z][0] = A[z][1]; \\
&\quad E[z][1] = A[z][2]; \\
&\text{totaalf} = \text{totaalf}/\text{totaal}; \\
&\text{totaal} = z; \\
&\text{for} (z = 0; z < \text{totaal}; z++) \\
&\quad B[z] = B[z] - \text{totaalf}; \\
&\text{printf}("\ %d %6.3f %6.3f %6.3f\t", z, A[z][1], A[z][2], B[z]); \\
&\text{polyfit ( A,B,H,z,10); \\
&\text{aclear(0, 0, 512, 512, 0); \\
&\text{for} (z = 0; z < \text{totaal}; ++z) \\
&\quad xx1 = E[z][0]; xx2 = E[z][1]; \\
&\quad\quad H[5] * xx1 * xx1 * xx1); \\
&\quad\quad H[9] * xx1 * xx2; \\
&\text{printf("%f %f %f %f\n", xx1, xx2, F[z], xx2 - corr-e); \\
&\text{if} (xx2 - corr-e > postmax) postmax = xx2 - corr-e; \\
&\text{if} (xx2 - corr-e < postmin && xx2 - corr-e > 0) postmin = xx2 - corr-e; \\
&\quad a = (\text{int})(corr); b = (\text{int})xx1; c = (\text{int})xx2; \\
&\text{if} (a >= 0) \\
&\quad \text{hlclear(c-a,b-1,a,255); hlclear(c-a,b,a,255);}
\end{align*}
\]
}  
}  
else if (a<0)  
}  
else  
}  
   
e=xx2-corr;  
}  
printf ( "maximum voor ijking = %f",premax);  
printf( "maximum na ijking = %f
",postmax);  
printf( "minimum voor ijking = %f",premin);  
printf( "minimum na ijking = %f
",postmin);  
}  

Appendix 7.  

/* ijkvert measures the distortion in vertical direction by   
scanning for edges at the calibration target with parallel equidistant vertical scan lines and fits a third degree polynomial through the correction data */  
{
  int limit,b,totaal,y,y11,y12,y1,dy,c,l,k,a,z;  
  float totaal,f,fact,xx1,xx2,e,corr;  
  float postmax,postmin,premax,premin;  
  movcur();  
  printf("y-coordinaat eerste vertikale scanlijn = ?\n");  
  scanf("%d",&y1);  
  printf("begincoordinaat eerste vertikale scanlijn = ?\n");  
  scanf("%d",&y11);  
  printf("eindcoordinaat eerste vertikale scanlijn = ?\n");  
  scanf("%d",&y12);  
  printf("stap in y-richting = ?\n");  
  scanf("%d",&dy);  
  printf("threshold =? \n");  
  scanf("%d",&limit);  
  totaal=0;  
  for (y=y1;y<=450;y+=dy)  
  {  
      b=4;  
      for (l=0;l<b;l++) {  
          getimage();  
          c=l;  
          a=edgelv(y11,y,y12,y,limit);  
          k=totaal;  
          for(c=2;c<a;c+=2) {  
              e=edge[c]/64.0;  
              A[k][0]+=1;A[k][2]+=(float)y;A[k][1]+=e;  
              A[k][4]+=(float)y*(float)y;A[k][3]+=e*e;  
              A[k][6]+=(float)y*(float)y*(float)y;A[k][5]+=e*e*e;  
              A[k][8]+=(float)y*(float)y*e;A[k][7]+=e*e*(float)y;  
              A[k][9]+=e*(float)y;  
              B[k]+=(edge[c]-edge[c-2])/64.0;  
              F[k]=B[k];  
}  
printf("threshold =? \n");  
scanf("%d",&limit);  
}  
printf("maximum voor ijking = %f",premax);  
printf( "maximum na ijking = %f\n",postmax);  
printf( "minimum voor ijking = %f",premin);  
printf( "minimum na ijking = %f\n",postmin);  
}
k++; 
}

for (z=totaal;z<k;z++)
{
   A[z][9]=A[z][9]/b; B[z]=B[z]/b;
   F[z]=F[z]/b;
   if (B[z]>premax) premax=B[z];
   if (B[z]<premin && B[z]>0) premin=B[z];
   totaal=k;
   movecursori();
   vlclear(y,0,512,255);
}

fact=0;
for (z=0;z<totaal;z++) fact+=B[z];
fact=fact/totaal;
printf("gemiddelde afstand = %6.3f\n",fact);
corr=0;
totaalf=0;
for (z=0;z<totaal;z++)
{
   if (z!=0) f=A[z-1][1];
   else f=A[z][1];
   if (A[z][1]>f) {
      corr+=B[z]-fact;
      totaal+=B[z]-fact;
   }
   else {
      corr=(A[z][1]-y11)/B[z]*(B[z]-fact);
      totaal+==(A[z][1]-y11)/B[z]*(B[z]-fact);
   }
   B[z]=corr;
   E[z][0]=A[z][1];
   E[z][1]=A[z][2];
}

totaalf=totaalf/totaal;
totaal=z;
for (z=0;z<totaal;z++)
{
   B[z]=B[z]-totaalf;
   printf("%d %6.3f %6.3f %6.3f\t",z,A[z][1],A[z][2],B[z]);
}

d=0;
polyfit ( A,B,V,z,10);
aclear(0,0,512,512,0);
for (z=0;z<totaal;z++)
{
   xx1=E[z][0];xx2=E[z][1];
   corr = (V[1]*xx1+V[2]*xx2+V[3]*xx1*xx1+V[4]*xx2*xx2+ 
          V[5]*xx1*xx1*xx1);
   corr+=V[6]*xx2*xx2*xx2+V[7]*xx1*xx1*xx2+V[8]*xx2*xx2*xx1+ 
          V[9]*xx1*xx2;
   printf("%f %f %f %f\n",xx1,xx2,F[z],xx1-corr-e);
   if (xx1-corr-e>postmax) postmax=xx1-corr-e;
if (xx1-corr-e<postmin && xx1-corr-e > 0) postmin=xx1-corr-e;
    a=(int)( corr);
    b=(int)xx1;
    c=(int)xx2;
    if (a>=0) {
        vlclear(b-1,c-a,a,255); vlclear(b,c-a,a,255);
    }
    else if (a<0) {
        a=a*-1;
        vlclear(b-1,c,a,255); vlclear(b,c,a,255);
    }
    e=xx1-corr;
}

printf( "maximum voor ijkking = %4.2f\t",premax);
printf( "maximum na ijkking = %4.2f\n",postmax);
printf( "minimum voor ijkking = %4.2f\t",premin);
printf( "minimum na ijkking = %4.2f\n",postmin);
}

main()
{
    ITEX_initialize();
    grab(0);
    movecursori();
    getimage();
    init();
    ijkhor();
    init();
    getimage();
    ijkvert();
}