Extracting straight lines and elliptic arcs from edge images

Broertjes, R.A.J.

Award date:
1994

Eindhoven University of Technology
Department of Electrical Engineering
Measurement and Control Section

Extracting straight lines and elliptic arcs from edge images

R.A.J. Broertjes

M.Sc. Thesis,
carried out from March 1993 to November 1993.

Commissioned by prof.dr.ir. P.P.J. v.d. Bosch,
under supervision of ir. M. Hanajík and ir. R.G. v. Vliet.

Abstract

Broertjes, R.A.J.; **Extracting straight lines and elliptic arcs from edge images**
M.Sc. Thesis, Measurement and Control Section ER, Department of Electrical Engineering, Eindhoven University of Technology, The Netherlands, November 1993.

Esprit project 6042 deals with automatically recognizing workpieces and data acquisition for robot programs to automatically weld these workpieces. A Vision survey system (VSS) is being developed for this purpose. This Thesis discusses the low level vision part of the VSS, and especially the extraction of straight lines and elliptic arcs from edge images.

The edge images of the original gray level images are obtained by using an accurate, one dimensional edge detector. For both vertical and horizontal direction an edge image is obtained in which the position of the edgepoints is with subpixel accuracy.

To reduce the amount of data, some features like straight lines and elliptic arcs are extracted from these edge images. Two line extraction algorithms are described which extract sets of straight line segments from the edge images. These algorithms are based on an edge following method. Both algorithms, from which one will be used in the VSS, produce line images of a good quality, with the line segments accuarately placed.

A principle that can be used for extracting straight lines *and* elliptic arcs from the edge images is also briefly discussed.

Samenvatting

Broertjes, R.A.J.; **Extracting straight lines and elliptic arcs from edge images**
Afstudeerverslag, vakgroep Meten en Regelen (ER), Faculteit Electrotechniek, Technische Universiteit Eindhoven, november 1993.

Esprit project 6042 heeft betrekking op het automatisch herkennen van werkstukcatogerieën en het verkrijgen van geometrische data om deze werkstukken automatisch te lassen. Een 'Vision survey system' (VSS) wordt hiervoor ontwikkeld. In dit verslag wordt op het 'low level vision'-deel van het VSS ingegaan, en met name op de extractie van lijnsegmenten en ellipsvormige segmenten uit edgepixelbeelden.

De edgepixelbeelden van de grijsbeelden worden verkregen met een nauwkeurige, één-dimensionale edge-detector. Voor zowel de verticale als de horizontale richting wordt een edgepixelbeeld verkregen, waarvan de positie van de edgepixels in 'subpixel accuracy' is.

Om de hoeveelheid data te verkleinen, worden features uit deze edgepixelbeelden geëxtraheerd, zoals lijnsegmenten en ellipsvormige segmenten. Twee lijnenextractie-algorithmen worden besproken, die een verzameling lijnsegmenten uit de edgepixelbeelden extraheren. Het principe waarop deze algorithmen zijn gebaseerd is een 'edge following'-methode. Deze algorithmen, waarvan er één in het VSS wordt gebruikt, leveren lijnenplaatjes van goede kwaliteit, waarin de lijnsegmenten nauwkeurig zijn geplaatst. Tenslotte wordt er ingegaan op een principe dat voor de extractie van lijnsegmenten *en* ellipsvormige segmenten gebruikt kan worden.

Contents

Glossary

Image processor: Hardware that can be used for applications in the image processing area. The image processor will mainly be used for acquiring images and storing them in the image processor's memory.

Resolution cell: The smallest most elementary areal constituent having an associated image intensity in a digital image.

Pixel: A pair, whose first member is a resolution cell and whose second member is the image intensity value (pixel value).

Gray level image: (Or: Gray image). Image in which each pixel has an integer value. Gray level images typically have pixel values in the range from 0 to 255.

Binary image: Image, in which each pixel has two possible values.

Edge: A sudden change of the intensity in a 2D gray level image.

Edgepoint: (Or: Edgepixel). A pixel in an edge image that is labelled as "edge".

Edge image: Image in which each pixel is labelled as "edge" or "non-edge". In addition to this basic labelling, pixels in an edge image may carry additional information like gradient information or more accurate edge position.

Edge chain: A set of connected edgepoints.

Line: In this report a 'line' means a straight line in 2D space, that can be represented by the equation $ax+by+c=0$.

Line segment: A part of a line, from which the endpoints are determined by two different points on the line considered.

Line image: A set of line segments.

1 Introduction.

Damaged ships are usually repaired by replacing the damaged parts with new constructed ones. Such workpieces are unique and were so far assembled and welded entirely manually. It is desirable to replace the manual labour by automatic robotic welding. The Measurement and Control Section of the Department of Electrical Engineering of the Eindhoven University of Technology is involved in a project that tries to achieve this. This project is Esprit project 6042: "Intelligent robotic welding system for unique fabrications", Hephaestos II [4]. This project is sponsored by the European Community. It is a project, executed by several institutions and industries from different European countries. The primary objective of this three year project is to advance the technology of robotic welding in the field of thick steel fabrication by the construction of an intelligent production robotic welding system which will be installed in the Piraeus ship repair yard of I & T Kalogeridis, the industrial user in the project.

The novel approaches in the project include the development of classification schemes for the shapes, sizes and configurations of the workpieces and the classification of weld seam data. To enable the production robotic welding system to recognize workpiece categories and to collect data for robot programs to automatically weld components, 3D computer vision can be used. The first step in 3D computer vision in general, consists of the acquisition of visual information from a scene, while the rest of the processing depends on tasks, objects, and/or a priori information.

The vision survey system (VSS) that is being designed for the project consists globally of a low level vision part and a high level vision part:

Low level part:

- **Acquires images** of the workpiece, using one or more cameras. These images are digitized in a 512 X 512 pixel matrix.
- **Detects edgepoints** in the gray level images obtained.
- **Extracts straight lines and curved segments** from the edge images obtained.

High level part:

- **Structural matching**. This concerns the recognition and reconstruction of the 3D workpiece, using the obtained line image(s). This part of the VSS is described in [10].
- **Geometrical matching**. This concerns the generation of 3D geometrical data that can be used for the welding process. This part of the VSS is described in [19].

The work described in this report mainly deals with the low level part and is the continuation of the work done by P.Buts [2]. He designed an edge detector, that could detect edgepoints along with properties like gradient direction and gradient magnitude of these points. In this report, the extraction of line segments and curved segments is described. Before this some general aspects of the low level vision, like image acquisition and edge detection, are discussed.

2 The low level part of the vision survey system.

The first part of the vision survey system (VSS) concerns low level vision. The low level vision operation sequence consists of successively:

- image-acquisition.
- edge detection.
- line extraction.

To obtain a 3D description of a workpiece, one or more camera images of the workpiece, viewed from different positions are needed [10]. The low level vision operations have to be done on these images separately. The output of the low level part of the VSS consists of one or more line images. Later in the high level part of the VSS, 'structural matching' and 'geometrical matching' are also done separately on the obtained line images, before the results are combined.

The acquisition of the images will be discussed in Sect.2.1. Edge detection will be done separately in horizontal and in vertical direction by using a one dimensional filter operator. This will be briefly discussed in Sect.2.2. After this, the obtained edge images (two edge images for every gray image) will be processed separately by a line extraction algorithm. Processing the edge image in horizontal direction will produce the vertical orientated line segments and processing the edge image in vertical direction will produce the horizontal orientated line segments. This will be discussed in Chapter 3. Finally, the horizontal orientated line segments and the vertical orientated line segments will be merged into one line image. In Fig.1, the diagram of the low level vision operations to be performed in case of N camera images is illustrated.

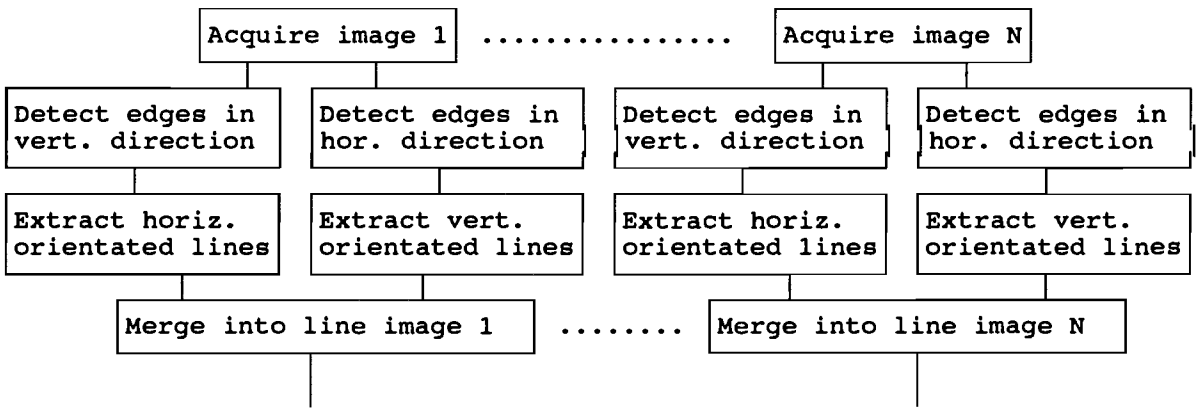


Fig.1. Diagram of the low level vision operations.

From now on, a position (x,y) in an image is defined as illustrated in Fig.2. In this report

N equals 512.

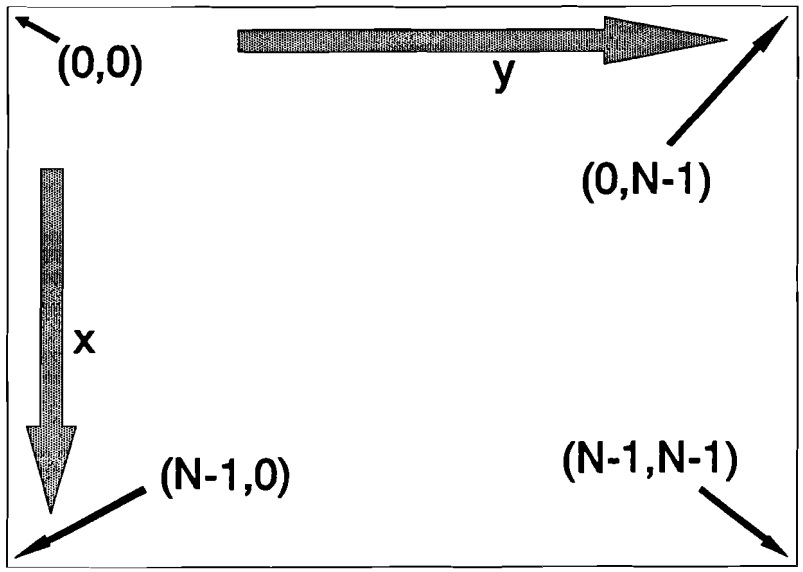


Fig.2. Definition of (x,y)

2.1 Image acquisition.

The first step of the VSS procedure is image acquisition. The cameras which will be used for acquiring images are CCD video cameras. An image of a scene, which is a two-dimensional continuous functions in reality, is spatially sampled and discretized in amplitude by the camera, and stored in an image processor. The CCD cameras used in the project produce images of 512x512 pixels, with 256 gray levels.

An aspect that is often neglected in machine vision applications is lighting [14]. Often complicated and expensive hardware and software are used to solve a vision problem where some attention to the "front end" would have solved the problem. In fact, the performance of an industrial vision system may depend more on well-designed lighting than on sophisticated image analysis.

The purposes of the lighting in an industrial machine vision application in general are:

- To produce a scene that is well matched to the camera and vision system. The brightest areas of the scene should cause a sensor illuminance that is just below the camera's saturation level and the video signal from the darkest significant areas of the scene should lie just above the vision system's noise level. When acquiring an image, a histogram of the gray image is displayed to detect and avoid camera saturation.

- The lighting design should insure that extraneous lights (e.g. daylight) around the machine are not going to interfere with the intentionally installed light sources. In this project, it may be assumed that no daylight will disturb the system's operation.

- The edges that describe the geometrical structure of the object should be visible.

The last mentioned requirement has appeared to be the biggest lighting-problem.

A change in intensity in the 2D gray image originates from [13]:

1. a (sudden) change in surface orientation in 3D.
2. a (sudden) change in surface reflectance (colour,material).
3. a (sudden) change in surface illumination (shadow).
4. a (sudden) change in depth (contours, occluding boundaries).

It is desired that only discontinuities caused by 1 and 4 are visible in the acquired images,

because only they are related to the geometrical structure of the object.

A theoretically perfect way to avoid shadows and reflections is to have illumination on the object coming uniformly from all directions [14]. To achieve this, photographers commonly use an umbrella diffuse reflector. The goal to achieve is light which seems to come from a large surrounding space. The best that can be done in case of large objects is to use as many light sources as possible. The light from these sources should pass through or reflect from diffusers of some type. The idea is to have a very well lit object without having the light going directly from the lamp to the object.

But even if such a perfect lightsource would be available, then it is still possible that certain edges are not visible. This is the case when the light intensity between two different orientated surfaces or between a surface that occludes an other surface is the same.

This problem can be solved as follows. Consider a workpiece that has to be analyzed and which is viewed by a camera from a certain position. Now consider two or more different lighting conditions for this workpiece. For each of these lighting conditions, an image can be acquired. When the desired edges (edges caused by 1 and 4 above) are visible in at least one of these images, then all the edges that describe the 3D workpiece from the current point of view are known. What has to be done now is to detect edges and extract line segments from all the images separately. Finally the line images could be merged into one line image. This final line image will contain more information about the geometrical structure of the workpiece, but will also contain more unwanted edges, originating from shadows and other unwanted changes in intensity in the gray image. In Sect.3.6. some experiments will be discussed to illustrate this.

The light sources which will be used in the experiments in Sect.3.6, are a combination of 'normal room lighting' with fluorescent tubes and an additional light source closer to the object.

2.2 Edge detection.

The second step of the VSS procedure is edge detection. Edges are changes in the intensity of the image, caused by one of the four reasons mentioned in Sect.2.1. The edge detector which is used in the project was proposed by D.Lee, implemented by P.Buts [2] and further modified to achieve detection of edgepoints with subpixel accuracy. From now on this edge detector will be called 'Lee's edge detector'. This edge detector is actually a discrete FIR linear filter, followed by a local maxima and minima detector. The principle of this edge detector is based on the assumption, that edges to be detected in the image are ideal step edges, corrupted by additive white gaussian noise. The response of the filter should be high for a step edge and low for noise. The edge detector is designed in such a way, that this performance criterium is optimal.

Edge detection with Lee's edge detector is actually one-dimensional, as the performance criteria are specified and the filter is designed for a one-dimensional signal (in case of two dimensional edge detection, filters are specified in 'kernels'). Therefore at least two separate edge detectors are necessary, one for edges in vertical direction (filtering and local extremes detection along the image rows), and one for edges in horizontal direction (filtering and local extremes detection along the image columns).

The details concerning the design of Lee's edge detector are stated in [2].

The results of Lee's edge detector are two edge images which contain the edgepoints in vertical and in horizontal direction. The pixel values of the edge images contain both information about the gradient direction and about the gradient magnitude of the considered edgepoint. The gradient direction of an edgepoint has two possible values, because the edge detector is one dimensional. Suppose (x,y) specifies a position in the edge image, with $0 \leq x < 512$ and $0 \leq y < 512$. Suppose p(x,y) is the pixel value of (x,y) which is stored in the image processor. $p(x,y) < 128$ corresponds with a transition from light to dark (seen from left to right or from top to bottom) and $p(x,y) > 128$ corresponds with a transition from dark to light (seen from left to right or from top to bottom). If no edgepoint is present on position (x,y) then $p(x,y) = 128$. The gradient magnitude m(x,y) of an edgepoint is defined as

$$m(x,y) = |128 - p(x,y)| \qquad (1)$$

m(x,y) should in the ideal case be proportional to the edge step size in the gray level image. The position of edgepoints which have been found on a certain row or column could be obtained with subpixel accuracy. The edge detector has been modified so that information about the accurate positions of the edgepoints could be computed from the pixel values. The output format is as follows:

When p(x,y)=128, then there is no edgepoint on position (x,y), just like above. When p(x,y) does not equal 128, then there is an edgepoint present. The pixel value now contains more precise information about the position of this edgepoint instead of information about the gradient of the edgepoint (see Table 1).

Table 1. Output format in case of subpixel accuracy.

p(x,y)	Meaning
0..3	Edgepoint
4..127	Not used
128	No edgepoint
129..251	Not used
252..255	Edgepoint

If an edgepoint is present on (x,y), then p(x,y) is as follows: ($p(x,y) \geq 252$) or ($p(x,y) \leq 3$). Now define p'(x,y) by

$$p'(x,y) = \begin{cases} p(x,y) - 256, & p(x,y) \geq 128 \\ p(x,y), & otherwise \end{cases} \quad (2)$$

Now when an edgepoint is present on position (x,y) then ($-3 \leq p'(x,y) \leq 4$). The more accurate position of edgepoint (x,y) will be called (x_{acc},y_{acc}) and can be computed by (3):

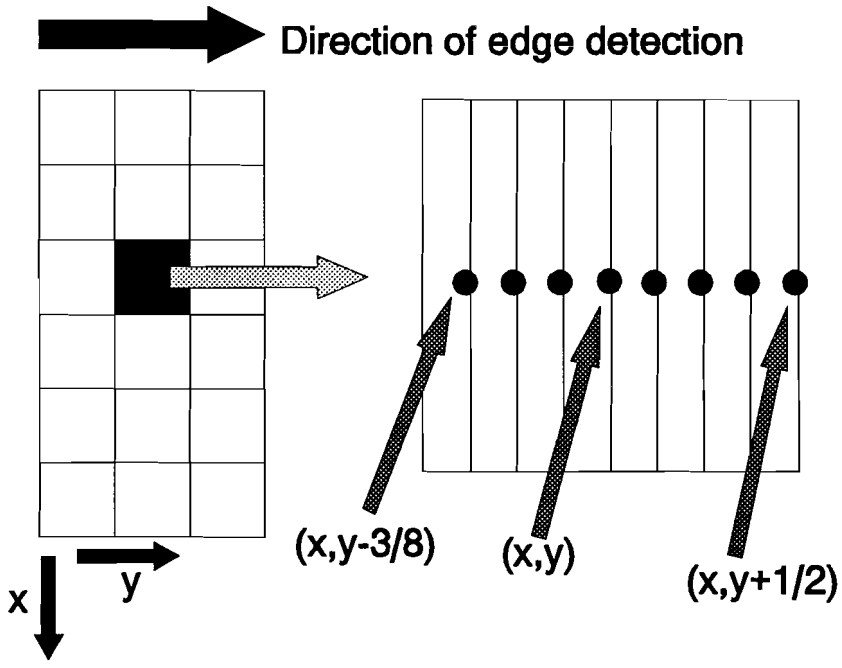


Fig.3. Subpixel accuracy.

$$(x_{acc}, y_{acc}) = \begin{cases} (x, y+p'(x,y)/8) & \text{if row-processing} \\ (x+p'(x,y)/8, y) & \text{if column-processing} \end{cases} \quad (3)$$

where row-processing means that edge detection is taking place in horizontal direction and column-processing means that edge detection is taking place in vertical direction. So now the pixel values are used to give additional spatial information instead of information about the gradient. Fig.3 illustrates a situation where an edgepoint is detected in horizontal direction (*row-processing*). The subpixel accurate position of this edgepoint, (x_{acc},y_{acc}), can be computed by using (3). The 8 possible subpixel accurate positions of (x_{acc},y_{acc}) are also illustrated in this figure.

The information about the gradient of the edgepoints gets lost when this output format is chosen. However, the algorithm has been modified such that it is possible to store the edge images in both output formats mentioned (see Appendix 1).

2.3 Line extraction.

The third step in the VSS procedure is the extraction of meaningful features from edge images, like line segments or curved segments. Successful extraction of such features always simplifies high level processing because the amount of data which has to be processed is much smaller. In Chapter 3, the extraction of line segments from edge images will be discussed. Two line extraction algorithms will be discussed in this Chapter. In Chapter 4, the extraction of line segments *and* elliptic arcs will be briefly discussed.

3 Extracting line segments.

The extraction of line images from perspective views of three-dimensional objects is done in a variety of applications, such as stereo ranging, dynamic scene analysis and robotic vision. Reliable extraction of line images always simplifies high level processing because the amount of data which has to be processed is much smaller. In this report only processing on a low level will be considered. No a-priori knowledge about the objects will be used during the line extraction process.

An 'ideal' line image is a line image in which only line segments originating from (sudden) changes in surface orientation in 3D and (sudden) changes in depth (occluding boundaries) are present. These are the line segments which describe the geometrical structure of the workpiece. Such an 'ideal' line image of a workpiece is shown in Fig.4. The quality of a line image relates to:

a) The similarity between the obtained line image and the 'ideal' line image. This is important for reliable object recognition on a higher level. A method to evaluate this, is to process line images with the structural matching algorithm and to evaluate the results of this high level process [10].

b) The accuracy of positions of line segments. Line segments should be placed as accurate as possible, to obtain precise geometrical data on a higher level.

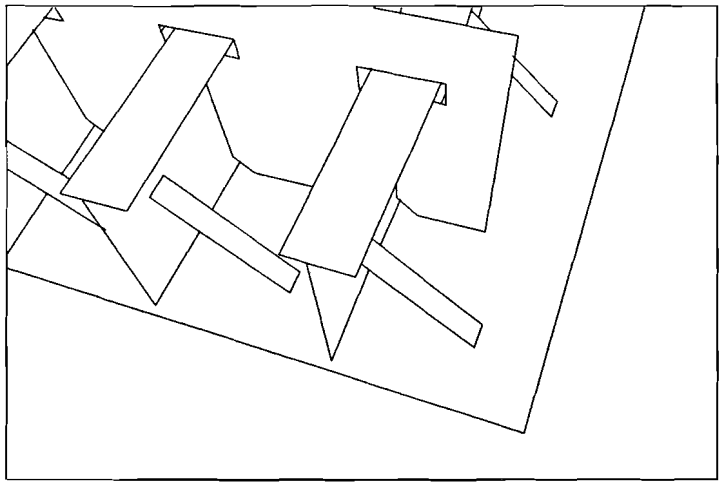


Fig.4. Ideal line image of a part of a 3D workpiece.

The input of a line extraction algorithm is generally a set of edgepoints in an (NxN) image plane (an edge image). These edgepoints could be given with subpixel accuracy (see Sect.2.2). Subpixel accuracy means that the coordinates of edgepoints are available

more accurately than the size of the resolution cells allows, by using the pixel value as additional spatial information. The actual problem of line extraction is to find sets of edgepoints. Through each set of edgepoints, a line segment can be estimated.

In the 1960's, most vision projects extracted line images from gray images in ways similar to Roberts' method [15]: detect edgepoints, link them into sequences, and fit line-equations to the sequences. Later, more sophisticated techniques of determining line images where developed. Some basic principles are [18]:

a) **Using Hough-transformation** [3], to obtain clusters of points in some parameter-space. A difficulty with this method is the fact that endpoints of line segments are not easy to identify. In [12], first the set of straight lines is obtained from these clusters in parameter-space. Then the line-intersections are heuristically searched and finally the line-linking structure is made. A disadvantage of this method is that it is slow.

b) **Edge following methods** first try to link edgepoints into edge chains. Finally these edge chains will be traced to obtain line segments or curved segments [7,16,17]. The edge chains obtained could be segmented by first locating cornerpoints. Basic concepts to obtain a segmentation are:

- merging
- splitting [3,8]
- tolerance-band solutions [20].

In the merging-technique, developed by Shirai [17,18], the curvature of the edge chains is used to estimate cornerpoints. In a splitting scheme, an edge chain is segmented into two sections. Then each section is examined to determine if it satisfies a line fitting criterion. This process goes on recursively, until all subsections are examined. Tolerance-band solutions involve the approximation of lines and curves by linear line segments that are constrained to pass within specified distances of the points.

c) Methods **using line support regions** [1]: Line support regions are regions of adjacent pixels with similar gradient-orientations. In these methods, the first step is to compute intensity gradients for each pixel from the gray image. This method proved to be not good enough for our purposes [2,6].

d) **Local edge linking.** As already stated, Roberts [15] used such an edge linking method. Zucker, Hummel, and Rosenfeld [21] have described a probabilistic

relaxation method, where each point in the image is assigned a probability of being on an edge or not.

The algorithms which are described in Chapter 3 are based on an edge following method. In Sect.3.1. the implemented line extraction algorithm is described. In Sect.3.2. it will be described how this algorithm could be adapted, so that it makes optimal use of the results of Lee's edge detector. The implementation in C will be described in Sect.3.3. Finally, in Sect.3.4, the results of the algorithms are evaluated.

3.1 A general line extraction algorithm.

The obtained line image(s) will be used on a higher level for the following:

- Reconstruction of the 3D-workpiece. For this task it is important that the line image(s) describe(s) the geometrical structure of the concerning workpiece as good as possible. However, it is not necessary that the line segments in the line image form a connected structure like in Fig.4 [10].

- Generation of geometrical data which will be used for the welding process. For this task it is important that the line segments are placed as accurate as possible.

Using Lee's edge detector [2], two edge images are obtained from the gray level image: The edge image in horizontal direction and the edge image in vertical direction. One of the features of Lee's edge detector is that for every edgepoint, the gradient direction and the gradient magnitude of this edgepoint is available (see Sect.2.2). Edgepoints could also be detected with subpixel accuracy. In the general line extraction algorithm that is discussed in this Section, all this extra information is not used. So the input of this algorithm are two (for both the vertical and the horizontal direction) binary images of the size NxN. The edge images used are binary, carrying information whether there is an edgepoint on a certain position or not.

If a one-dimensional edge-detection algorithm is used (like Lee's edge detector, see Sect.2.2), the edges in horizontal and the edges in vertical direction are available separately in two different edge images.

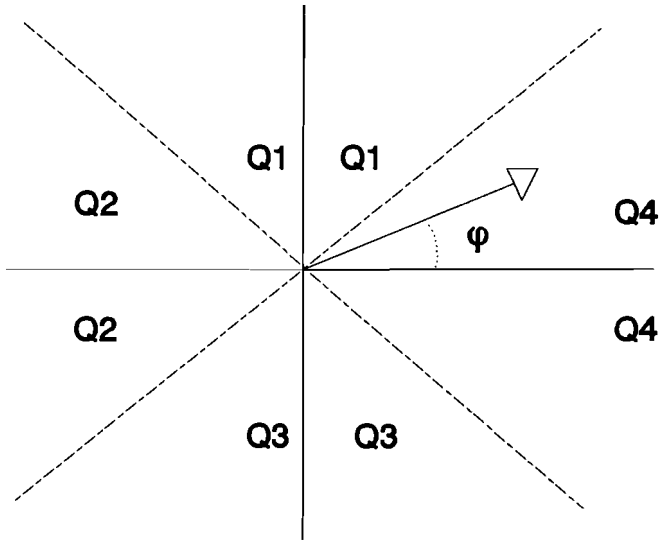


Fig.5. Definition of φ and the 4 quadrants.

There are two ways of extracting a line image from these two images:

- **The two edge images could be processed separately by a line extraction algorithm, before merging the results into one line image.** The edge images in horizontal and in vertical direction, obtained by Lee's edge detector [2] (or by any other one dimensional edge detection algorithm), will now be processed separately by a line extraction algorithm. The horizontally orientated line segments (direction in Q2 or in Q4, see Fig.5) will be obtained by processing the edge image in vertical direction and the vertically orientated line segments (direction in Q1 or in Q3, see Fig.5) will be obtained by processing the edge image in horizontal direction. The two sets of line segments, obtained by the line extraction algorithm will finally be merged into one line image. An advantage of this method is that line segments are probably more accurately placed. One can for example assume that edgepoints, originating from an edge with direction in Q1 or Q3, are more accurately detected by edge detection in horizontal direction. So in this case, a line segment estimated through edgepoints detected in horizontal direction, is probably more accurately placed than a line segment estimated through edgepoints detected in vertical direction.

- **The two edge images could be combined first, before processing the result by a line extraction algorithm.** The edge images could be combined by simply using the OR-operation: if there is an edgepoint in at least one of the edge images, then an edgepoint is defined on that position in the combined edge image:

$$e(x,y) = e_v(x,y) \vee e_h(x,y) \qquad (4)$$

where e_v and e_h are the edge images in respectively vertical and horizontal direction, where e is the resulting edge image and where (x,y) specifies the position in the edge images. If necessary, a threshold could be used to decide if a pixel is considered as an edgepoint or not. The advantage of first combining the edge images is the reduced execution time of the line extraction process: only one instead of two edge images have to be processed by the line extraction algorithm.

Of course, information about the gradient of the edgepoints disappears by using the OR-operation. By processing the results of Lee's edge detector this way, a disadvantage of this method appeared: edgepoints which correspond to diagonal edges for example, (i.e. edges with a vertical and a horizontal gradient component) were not placed exactly on the same place in the two edge images. One of the features of Lee's edge detector is that edgepoints in an edge image that correspond to an edge form edge chains which are one pixel thick (see Sect.2.2). This feature disappears, after combining the two edge images by using (4) (see Fig.6).

The method which has been chosen is separate processing of the edge images. For the line extraction algorithm which will be described in Sect.3.2, this is essential. For the algorithm which will be described in this Section, it is not an essential condition.

The line extraction algorithm which will be described in this chapter is applicable on edge images, obtained by Lee's edge-detector or by any other edge detector. From now on, this algorithm will be called LINEX1. If the edge images in both vertical and horizontal direction are available, then a choice can be made between first combining the edge

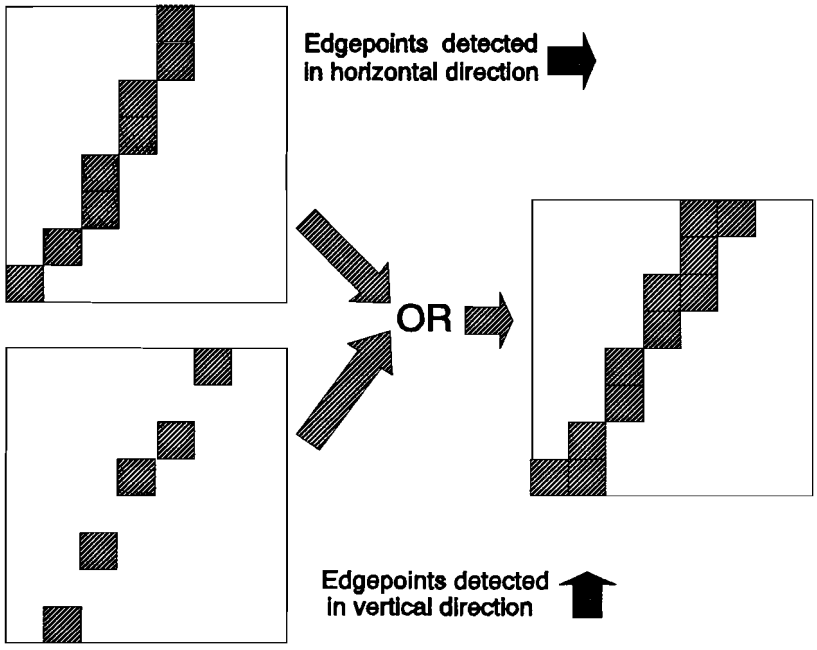


Fig.6. Result of OR-operation.

images before further processing or not. As already stated, the edge images obtained by Lee's edge detector can better be processed separately.

The principle of LINEX1 which has been developed and implemented is based on an edge following method. Normally, in case of such methods, edgepoints will be first linked into edge chains, before these edge chains are segmented into sets of edgepoints that correspond with straight line segments. Finally straight line segments are estimated through each set of edgepoints of a segmented edge chain [7,16,17].

Edges are followed by predicting the next edgepoint by using the set of edgepoints that has already been found. Each time an edgepoint has been added to the set of edgepoints, a straight line estimation through this set of edgepoints is computed. A least squares error

(LSE) criterium of the distance between the edgepoints and the estimated line is used. The principle used in LINEX1 is to use the last line estimation when the last edgepoint that belongs to a line segment has been found (Fig.7). Now both segmentation and estimating lines through the edgepoints of the segments (Fig.7, right) can be skipped. So the line estimation will be used for both predicting edgepoints and obtaining the final line estimation. LINEX1 actually searches sets of edgepoints which seem to be approximately on a straight line.

A 'processed edgepoint' is defined as an edgepoint that already belongs to a line segment, or an isolated edgepoint that already has been tried as a starting point but does not belong

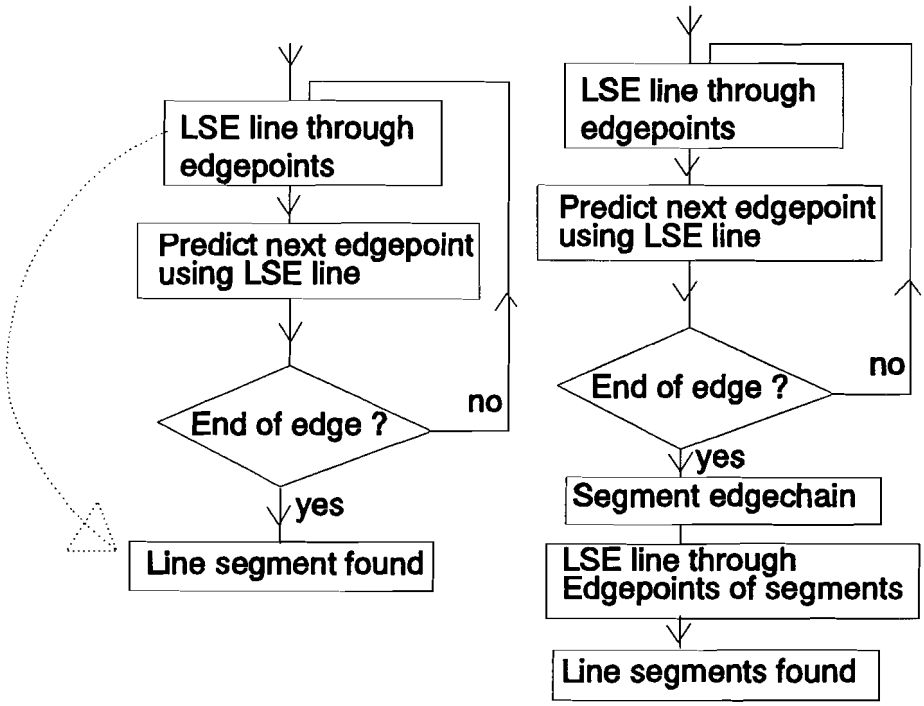


Fig.7. LINEX1 (left) and common edge following method (right)

to a line segment. A binary image associated to the edge image (one bit per pixel) is used to register which edgepoints are already processed during the line extraction process. This binary image, which will be called 'the bitmap', will be used to find starting points of edges. When the search of the next unprocessed edgepoint will take place lexicographically (row after row), then the edgepoint which has been found probably corresponds to one end of an edge. After a starting point has been found this way, this point will be the first point of a set of edgepoints. Then the vicinity of this starting point will be viewed to estimate the initial direction of the edge following process. The search for starting points and the estimation process for this initial direction will be described in detail in Sect.3.1.2.

This estimated direction will be used to predict the second edgepoint. If there exists indeed an edgepoint on the predicted position or in the vicinity of this point, this edgepoint will be added to the set of edgepoints. The direction of the line, estimated through the current set of edgepoints will be used to predict the next edgepoint. The place from which the next edgepoint is predicted will be the last edgepoint found, perpendicularly projected onto the estimated line (Fig.8). Projection onto the estimated line is done, to obtain a set of edgepoints which are approximately on a straight line. The LSE line estimation will be described in Sect.3.1.3. The prediction of edgepoints will be described in Sect.3.1.4. If there is neither a point on the predicted position, nor in it's vicinity, then:

- the last edgepoint of a line segment has been found, or ...
- there are some edgepoints missing.

To deal with the second possibility, the next step is to search for an edgepoint, at some distance. If an edgepoint is found, then it is assumed that some edgepoints were missing, and the edge following continues. This will be described in Sect.3.1.5. If no edgepoint is found, then the last edgepoint of the corresponding edge has been detected: a line segment has been found. If the direction of the line segment is in the correct quadrant (like described on p.19) and the length of the line segment is above some threshold, then the line segment will be stored and the bitmap will be updated. Now the next starting point will be searched for.

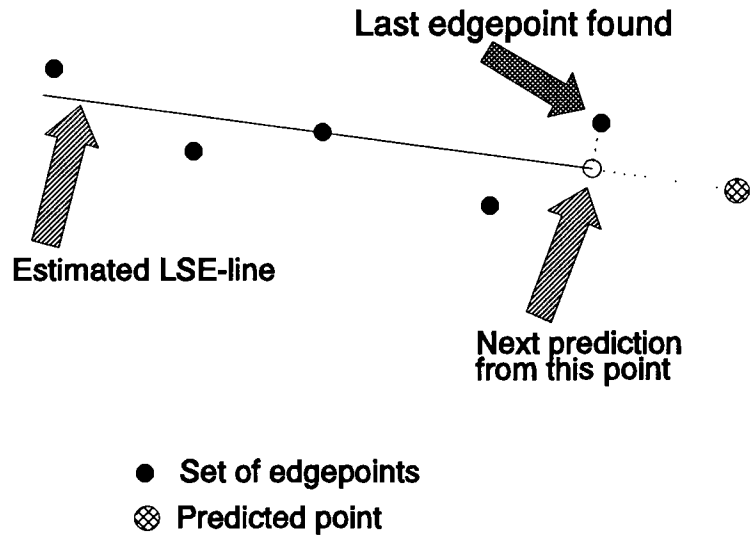


Fig.8. Edgepoint prediction.

3.1.1 Searching starting points.

To initialize the edge following process, a starting point, that means an edgepoint that (hopefully) corresponds to the end of an edge, has to be found. Also an initial search-direction should be estimated.

A binary image of the edge image (one bit per pixel) can be used to register which edge-points are already processed and which are not. As already stated, a 'processed edgepoint' is an edgepoint that already belongs to a line segment, or an isolated edgepoint that already has been tried as a starting point but does not belong to a line segment. This binary image will be used to find starting points of edges.When the search for a starting point takes place lexicographically, then the first edgepoint to be found probably corresponds to the end of an edge. Searching lexicographically means that the image is scanned row after row, from left to right, until an unprocessed edgepoint has been found.

It is possible that a starting point that has been found this way corresponds to some arbitrary point on an edge, as illustrated in Fig.9. Now it is impossible to follow this edge at once and therefore to find the line segment at once; at least two line segments will be extracted out of this edge. Later in this Section it will be described how this problem can be solved.

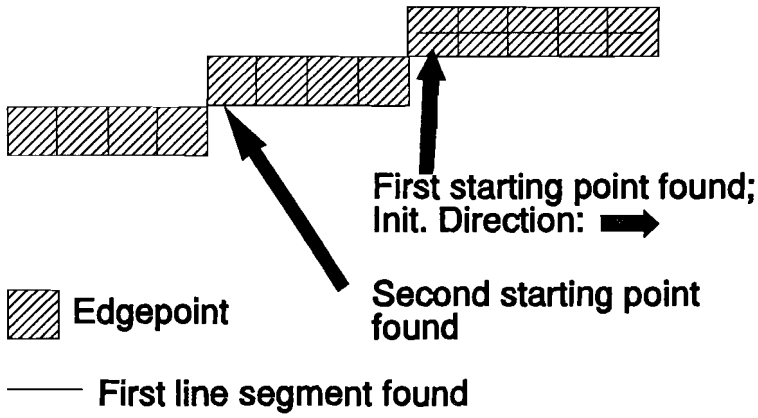


Fig.9.Problem that could occur when edgepoints form an almost horizontal line.

If all edgepoints have been processed, then the line extraction is finished. Fig.10 illustrates the edge following process using a triangle as an example structure.

When a starting point has been found, this point will be the first point of a set of edgepoints. Now the environment of this starting point will be viewed to estimate the initial search-direction of the edge following process. The 5x5 environment of the starting point will be considered in the estimation.

For 8 different directions, a template is defined (see Fig.11). Only unprocessed edgepoints are considered in the estimation. Let (x,y) be a position in the edge image considered. Now U(x,y) is defined as follows: $U(x,y)=1$, if there is an edgepoint at position (x,y), which has not yet been processed, and otherwise $U(x,y)=0$. Suppose (i,j) determines the position in a template, with $-2 \leq i \leq 2$ and $-2 \leq j \leq 2$. Now define $TM(\varphi,i,j)$ as the value in the template determined by φ ($\varphi=0,45,90,$ .... or 315; see Fig.11), where(i,j) determines the place in this template. The initial search-direction (x_s,y_s) of the edge following process can now be estimated by using (5) and (6).

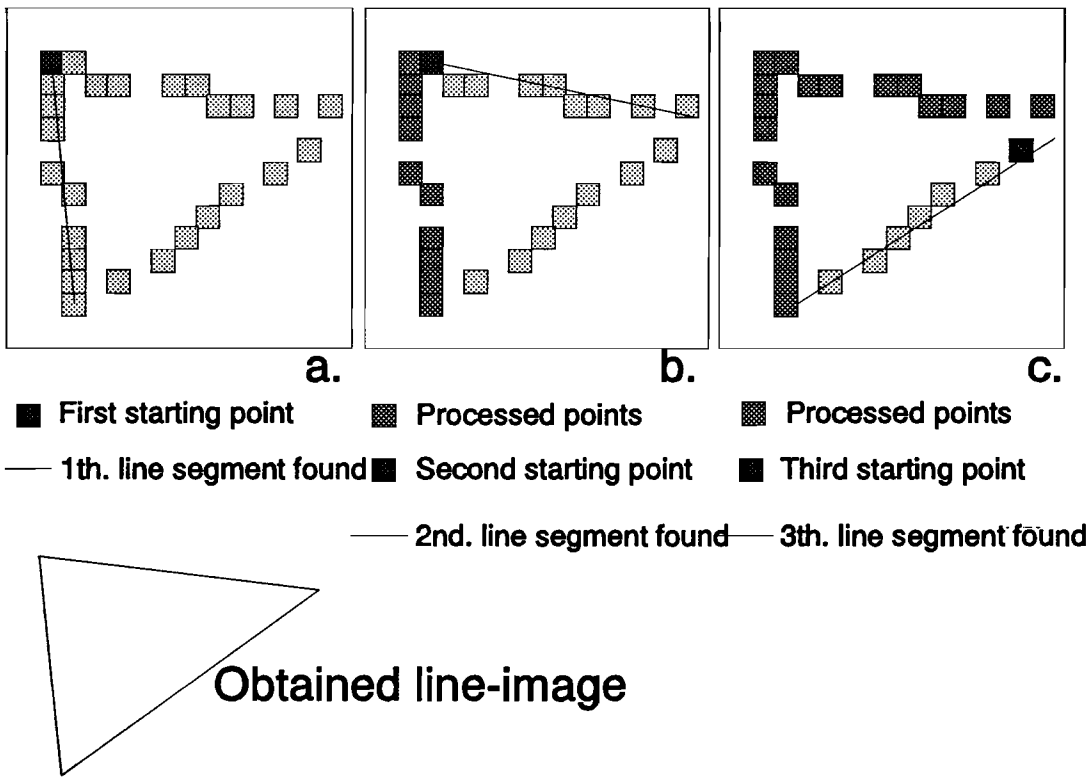


Fig.10. Search for starting points: Processed and unprocessed points.

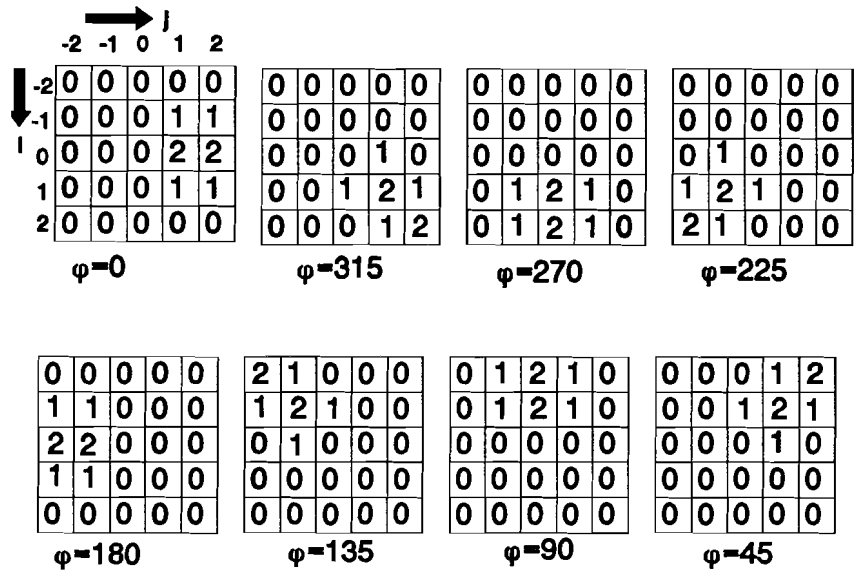


Fig.11. 5X5 Templates, used for estimation of the initial direction.

$$S(\varphi)=\sum_{i=-2}^{2}\sum_{j=-2}^{2}(U(x_s+i,y_s+j).TM(\varphi,i,j)),\ \varphi=0,45....,315 \qquad (5)$$

$$Direction(x_s,y_s)=\varphi:\ MAX(S(\varphi)),\ \varphi=0,45.....,315 \qquad (6)$$

If $S(\varphi_1)=S(\varphi_2)=MAX(S(\varphi))$ then $MAX(S(\varphi))$ can be either φ_1 or φ_2. In Fig.10, it is illustrated why processed points should not be taken into account in the direction estimation. Consider Fig.10b. If the edgepoints of the first line segment that was found would be considered, the direction estimation of the second starting point could be wrong; for example $\varphi=225$ (left-under). When these points would be ignored, then the estimation would clearly have the right value $\varphi=0$, because there are only unprocessed edgepoints on the right side of the starting point.

Now reconsider the earlier mentioned problem which was illustrated in Fig.9. It is clear that the problem only occurs in case of an almost horizontal edge. The direction-estimation of a starting point that corresponds with some arbitrary point on such edge will be 0 or π. The problem can now be solved as follows. When the direction-estimation of the starting point equal 0 or π, then let the edge following process take place with both initial direction 0 *and* initial direction π, from the starting point considered. From both these opposite search-directions, a set of edgepoints (a line segment) will be obtained. Both sets of edgepoints will finally be joined forming one set of edgepoints, before a line estimation through this new set of edgepoints is carried out.

3.1.2 LSE line estimation.

Considering a set of points, a Least squares error (LSE) line estimation can be made for these points. Consider a set of points $(x_1,y_2)....(x_n,y_n)$, and denote the *i*th point (x_i,y_i) as the vector $\mathbf{v}_i$. The estimated line will be used for predicting the next edgepoint and will be stored when the end of the edge has been detected. The problem is to find the best parameters (a,b,c) for the line equation ax+by+c=0. We will define a line to be the best fit to the set of points if it minimizes the sum of squares of the perpendicular distances from the points to the line. Stated otherwise: parameters a,b and c should be found such that

$$\sum_{i=1}^{n} (ax_i+by_i+c)^2 \qquad (7)$$

is minimized.

This line is sometimes called the best eigenvector fit. Now it will be described how the parameters of this line can be computed [3]. Define the following:

$$S_x=\sum_{i=1}^{n} x_i, \quad S_y=\sum_{i=1}^{n} y_i, \quad S_{xx}=\sum_{i=1}^{n} x_i^2, \quad S_{yy}=\sum_{i=1}^{n} y_i^2, \quad S_{xy}=\sum_{i=1}^{n} x_i.y_i \qquad (8)$$

Further define the symmetric matrix

$$S=\sum_{i=1}^{n} v_i v_i^t \qquad (9)$$

as the scatter matrix of the n given points.

The first step [3], is to standardize the points by subtracting the mean of the set from each point. The standardized scatter matrix S' of the set of standardized points is now according (10):

$$S' = \begin{bmatrix} S_{xx}-S_x^2/n & S_{xy}-S_xS_y/n \\ S_{xy}-S_xS_y/n & S_{yy}-S_y^2/n \end{bmatrix} \qquad (10)$$

The second step is to find the principal eigenvector [3] of the standardized scatter matrix S'. The best line estimation is the unique line through the mean of the set of points and parallel to this eigenvector [3]. The eigenvalues of the standardized scatter matrix S' are as follows:

$$\lambda_1=(S'_{11}+S'_{22}-\sqrt{Det})/2, \quad \lambda_2=(S'_{11}+S'_{22}+\sqrt{Det})/2 \tag{11}$$

where

$$Det=(S'_{11}+S'_{22})^2-4(S'_{11}S'_{22}-S'_{12}S'_{21}) \tag{12}$$

The principal eigenvector is now as follows:

$$\begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} (\lambda_1-S'_{11})/\sqrt{S'_{12}S'_{21}+(S'_{11}-\lambda_1)^2} \\ S'_{12}/\sqrt{S'_{12}S'_{21}+(S'_{11}-\lambda_1)^2} \end{bmatrix} \tag{13}$$

Because the best-fitting line is parallel to this eigenvector and because a and b determine the direction of the line, a and b can now be determined:

$$a=\begin{cases} 0 & (S_{xyc}=0)\wedge(S_{xxc}\geq S_{yyc}) \\ 1 & (S_{xyc}=0)\wedge(S_{xxc}<S_{yyc}) \\ e_2 & otherwise \end{cases} \tag{14}$$

$$b=\begin{cases} 0 & (S_{xyc}=0)\wedge(S_{xxc}<S_{yyc}) \\ 1 & (S_{xyc}=0)\wedge(S_{xxc}\geq S_{yyc}) \\ e_1 & otherwise \end{cases} \tag{15}$$

If a=0, the line is horizontal; if b=0, the line is vertical. Now a and b are estimated, c can be estimated by using the line equation ax+by+c=0 and by filling in for x and y the mean values of the x- and the y-coordinates of the set of points (S_x/n and S_y/n respectively):

$$c=-(aS_x+bS_y)/n \tag{16}$$

After a LSE line estimation has been computed, more accurate coordinates of the begin- and endpoint are computed. This is done by projecting these points perpendicularly on the estimated line (Fig.12).

Consider Fig.13. Suppose the point (x',y') is perpendicularly projected onto the line described by ax+by+c=0 and the result of this projection is the point (x'',y'').
Line 1 and line 2 in Fig.13 are perpendicular to each other. Because of this, and by using the coordinates of (x',y'), the parameters of line 2 are computed by

$$a'=-b, \qquad b'=a, \qquad c'=bx'-ay' \tag{17}$$

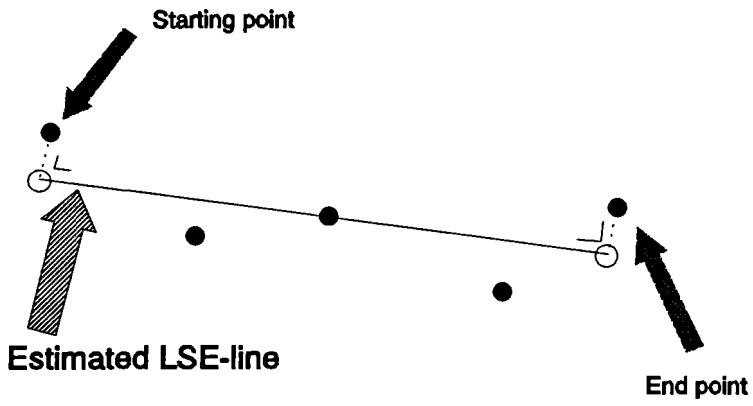


Fig.12. Perpendicular projection of start- and endpoint.

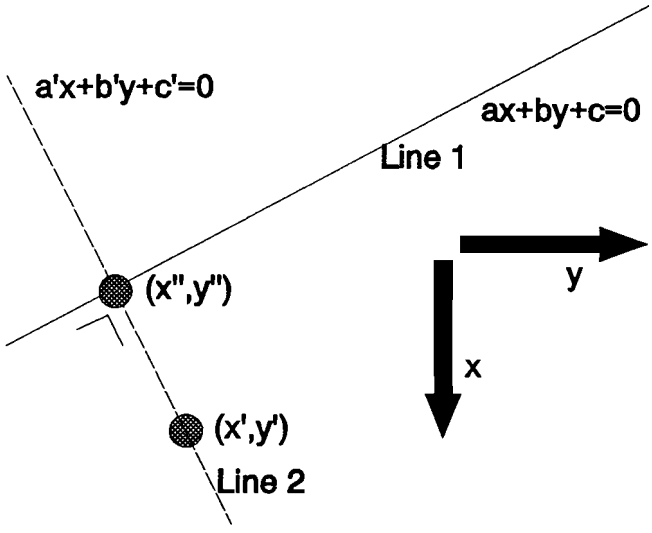


Fig.13. Perpendicular projection on a line.

The coordinates of (x'',y'') are computed, using both line equations ax+by+c=0 and a'x+b'x+c'=0. y'' can be obtained by computing the crosspoint of these two lines:

$$y'' = \frac{-c-ax''}{b} = \frac{-c'-a'x''}{b'} \qquad (18)$$

From (18) we obtain:

$$\frac{a}{b}x'' + \frac{c}{b} = \frac{a'}{b'}x'' + \frac{c'}{b'} \quad \Leftrightarrow \quad x'' = \frac{c'b - cb'}{ab' - a'b} \tag{19}$$

By using (17) and the assumption that $a^2+b^2=1$ and $a'^2+b'^2=1$, we obtain

$$x'' = bc' - b'c \tag{20}$$

Now that x'' is known, y'' can be computed by using (18). When b=0, the part with b'(=a) in the denominator (the right part of (18)) can be used to compute y''. Otherwise the other part can be used.

3.1.3 Edgepoint prediction.

Edges are followed by predicting the next edgepoint, by making use of the set of edgepoints that has already been found. After an edgepoint has been added to the set of edgepoints, a new set of line parameters will be computed as described in the previous Section. When the line parameters a,b and c are available, the direction φ of the line can be computed by

$$\varphi = \begin{cases} 0 \vee \pi & (b=0) \\ \frac{1}{2}\pi - \arctan\left(\frac{a}{b}\right) & (b \neq 0) \end{cases} \tag{21}$$

where φ is defined as in Fig.5, p.18. The decision between $\varphi=0$ and $\varphi=\pi$ when b=0 can be made by considering the last two points of the set of edgepoints: If the y-coordinate of the last point found is higher than the y-coordinate of the one but last point found, than $\varphi=0$; else $\varphi=\pi$. The stepsize, i.e. the distance between the last point found and the predicted point could be parameterized and will be 1 by default. A higher stepsize will probably speed up the algorithm, but also deteriorate the quality of the line images (see Sect.3.5.1). Coordinates of the predicted point are computed by

$$x_{pred} = x_{end} + (stepsize.\sin(\varphi)), \quad y_{pred} = y_{end} + (stepsize.\cos(\varphi)) \tag{22}$$

where (x_{end}, y_{end}) is the position of the perpendicular projection of the last edgepoint found on the estimated line and (x_{pred}, y_{pred}) is the position of the predicted point. The obtained coordinates should be rounded to the nearest pixel. If there is no edgepoint on the predicted position, the environment of the predicted edgepoint will be examined for an edgepoint. If an edgepoint is found this way, this point will be called a region point. For this, 4-connectivity, 8-connectivity or 24-connectivity will be used (Fig.14). However, not all points in the connectivity-environment will be considered. The position of the region-

point (x_{reg}, y_{reg}) should namely satisfy (23) and (24):

$$\begin{cases} x_{reg} \geq x_{pred} & (x_{pred} > x_{end}) \\ x_{reg} \leq x_{pred} & (x_{pred} \leq x_{end}) \end{cases} \quad (23)$$

$$\begin{cases} y_{reg} \geq y_{pred} & (y_{pred} > y_{end}) \\ y_{reg} \leq y_{pred} & (y_{pred} \leq y_{end}) \end{cases} \quad (24)$$

Some examples of search-environments, obtained this way are shown in Fig.14.

In case of a search in a 'high'-connectivity environment, first the 'lower'-connectivity environments will be examined. For example in case of 24-connectivity, first the 4-connectivity environment, then the 8-connectivity environment and finally the 24-connectivity will be examined. The first edgepoint found this way is now probably closest to the predicted position.

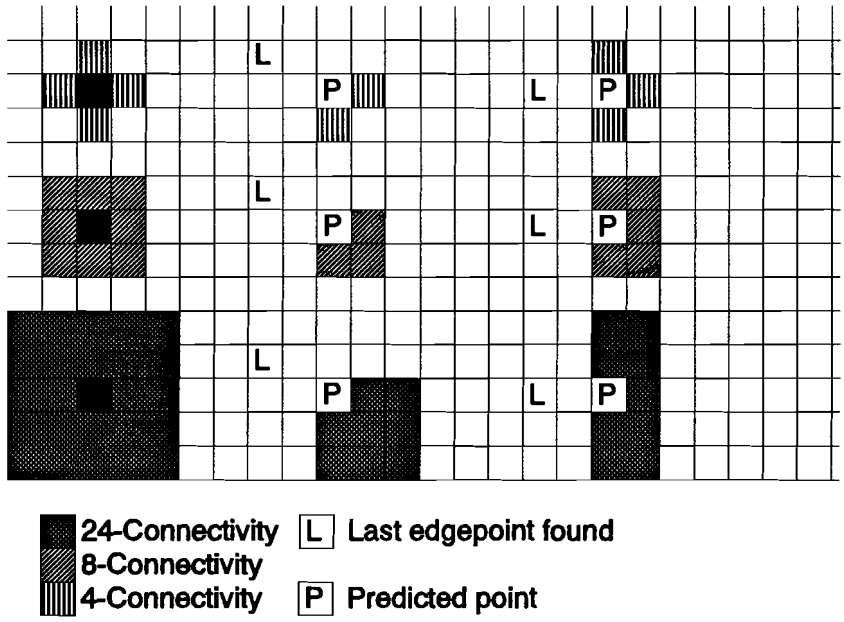


Fig.14. Definition of connectivity and search-environment.

3.1.4 Filling edge interruptions.

In low level vision it often occurs that some edgepoints of the corresponding edge are missing, mostly due to bad illumination of the object (see Sect.2.1). One of the features of LINEX1, is that it is able to fill in edge interruptions.

If there's no point on the predicted position, neither in it's environment, then:

- the last edgepoint of a line segment has been found, or..
- there are some edgepoints missing.

To deal with the second possibility, the next step is to predict an edgepoint at some distance. If there is indeed an edgepoint at some distance, then it is assumed that some edgepoints of the line segment were missing, and the edge following process continues. To detect edgepoints at some distance, the edgepoint from which the next prediction takes place is the same as before (the perpendicular projection of the last edgepoint found, onto the estimated line). Now edgepoints will be predicted in the same direction, on distances which are multiples of the current step-size: respectively 2.step-size, 3.step-size, ... d.step-size, where d will be parameterized and called the 'disturbance-threshold'. Each time when there is no edgepoint on the predicted position, neither in it's environment, this position will be added to a buffer. So this buffer will hold the positions where an edgepoint was expected, but not found. Such points we call 'simulated edgepoints' [17]. If at least two successive edgepoints are found, within the number of iterations determined by the disturbance-threshold, then all points in the buffer are considered as edgepoints and they are added to the set of edgepoints. Otherwise the last edgepoint found will be considered as the end of the corresponding edge and the positions in the buffer are not added to the set of edgepoints. In Fig.15, a situation with 3 missing edgepoints is used to illustrate this. When the disturbance-threshold is at least 3 and when at least two successive edgepoints are found after the interruption, then the interruption will be filled with 3 simulated edgepoints (see Fig.15a). So only one line segment will be obtained in this situation. In Fig.15b the interruption will not be filled because only one edgepoint is found after the interruption.

It should be noted here, that filling edge interruptions this way, is actually not a low level operation. The a-priori knowledge, that there are probably some edgepoints missing, has been used here.

3.1.5 Using the bitmap to avoid double line segments.

When the step-size (see Sect. 3.1.3) is above 1 or when the edgepoints form chains which are more than just one pixel thick, the following problem occurs. Fig.16 illustrates a situation where the edgepoints form an edge chain which is more than one pixel thick and where the stepsize equals two. When the line segment showed in this figure has been found, only the filled points (these are the points which are found during the edge following process and which are used for the line estimation) will be considered as processed points. This means that it is possible that more line segments are estimated

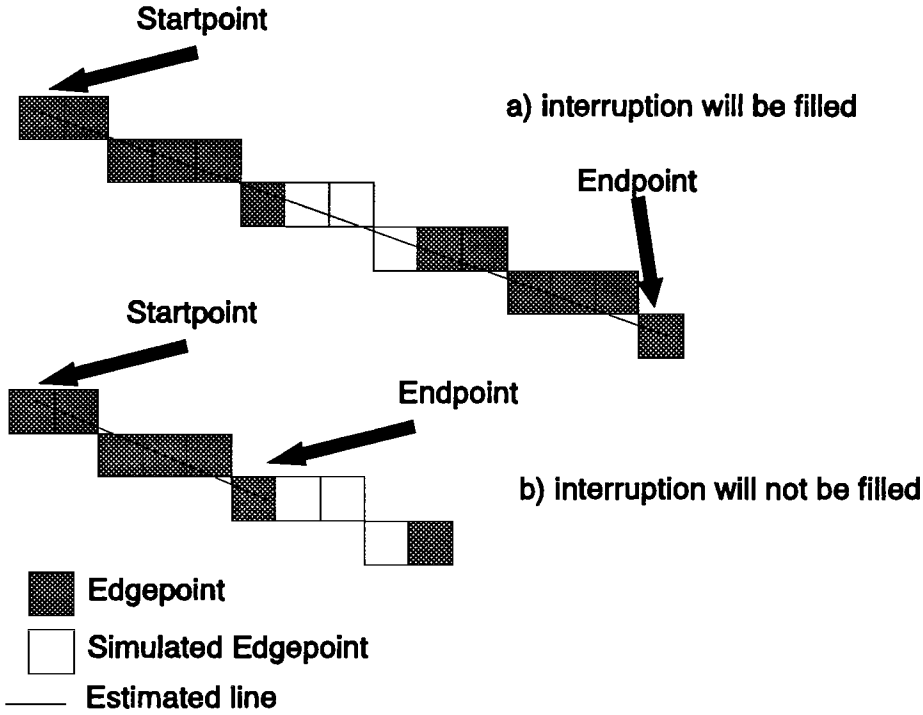


Fig.15. Filling edge interruptions; simulated edgepoints.

through the other edgepoints, the unprocessed points. To avoid double line segments, the bitmap (see p.21) can be used. This is a binary image. When the points around the edgepoints are defined as processed once the line segment has been found, these points will not be tried as starting points and may not be used by other line segments anymore. For each edgepoint that has been found, a rectangular environment is defined. When edgepoints are present in this rectangle, these edgepoints will be defined as processed. The bitmap can be used to indicate this. Fig.16 illustrates a situation with step-size>1. In this figure, 3x3 environments around the edgepoints are used. It is clear that only points p and q will just fall outside these environments of the edgepoints when the rectangle has this size. The size of the rectangle and wether it will be used at all, will be parameterized (see Sect.3.4).

A disadvantage of this method to avoid double line segments is that some edgepoints which actually belong to other line segments could also be defined as processed, like point r in Fig.16. When edgepoints form edge chains which are one pixel thick like the results of Lee's edge detector, and when the stepsize is 1, then no double line segments will occur, so the above described solution does not have to be used (which is often the case in practice).

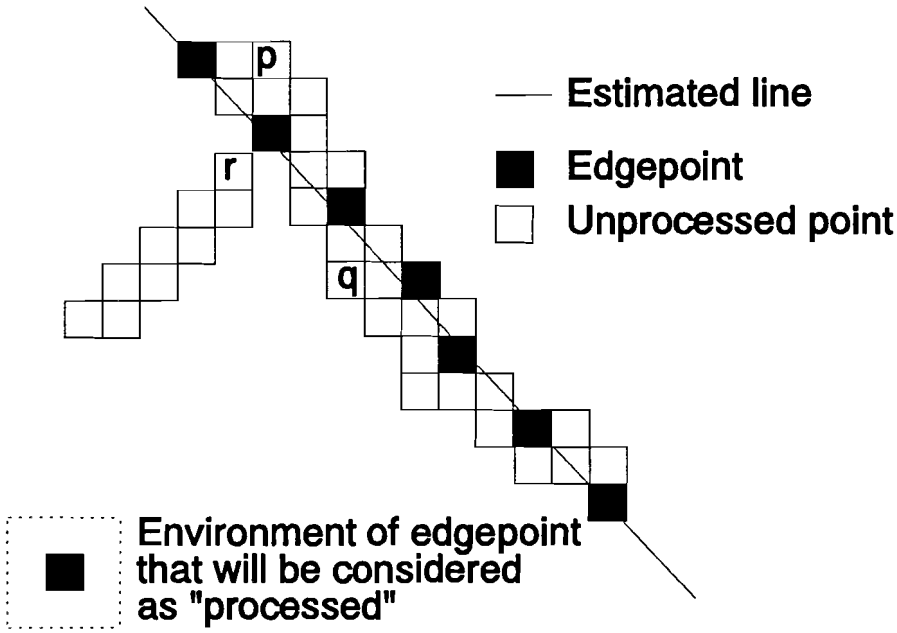


Fig.16. Avoiding double line segments.

3.2 Making optimal use of Lee's edge detector.

The algorithm LINEX1 can be used for results of any edge detector. The algorithm can be modified in such way, that it makes optimal use of Lee's edge detector (see Sect.2.2). The modified algorithm will be called LINEX2 from now on. The two features of Lee's edge detector, which will be used are:

- Edgepoints obtained by Lee's edge detector form edge chains which are one pixel thick.
- Results can be represented with subpixel accuracy.

The following parts of the algorithm LINEX1 have to be changed to make optimal use of both these features:

- Edgepoint prediction.
- The search for edgepoints in the environment of the predicted point.
- Lines will be estimated through edgepoints, given in subpixel accuracy.

The most important modifications of the algorithm are related to the edgepoint prediction process and the search for edgepoints and will be discussed in Sect.3.2.1. The rest of the modifications are discussed in Sect.3.2.2.

3.2.1 Edgepoint prediction and edgepoint searching.

The most important modifications which have to be made on the algorithm LINEX1 are related to edgepoint prediction and the search for edgepoints. The principle which will be used is that successive rows or successive columns will be searched for edgepoints during the edge following process. This will be called 'row-search' and 'column-search' respectively. The positions of the edgepoints on these rows or columns are in subpixel accuracy. The point from which the next edgepoint will be predicted is, (just like in Sect. 3.1.4), the last edgepoint found, perpendicularly projected on the LSE estimated line. Now the coordinates of the predicted point can be computed as follows. Suppose (x_{end},y_{end}) is the position of the last edgepoint found, (x_{pred},y_{pred}) is the position of the predicted point and φ represents the direction of the estimated line through the current set of edgepoints. When the horizontal edge image is being processed, then:

$$x_{pred}=\begin{cases} x_{end}+1 & (\varphi>\pi) \\ x_{end}-1 & (\varphi<\pi) \end{cases}, \qquad y_{pred}=y_{end}+\cos(\varphi) \tag{25}$$

and when the vertical edge image is being processed, then:

$$x_{pred}=x_{end}-\sin(\varphi) \quad , \quad y_{pred}=\begin{cases} y_{end}+1 & (-\pi/2<\varphi<\pi/2) \\ y_{end}-1 & (\pi/2<\varphi<3\pi/2) \end{cases} \qquad (26)$$

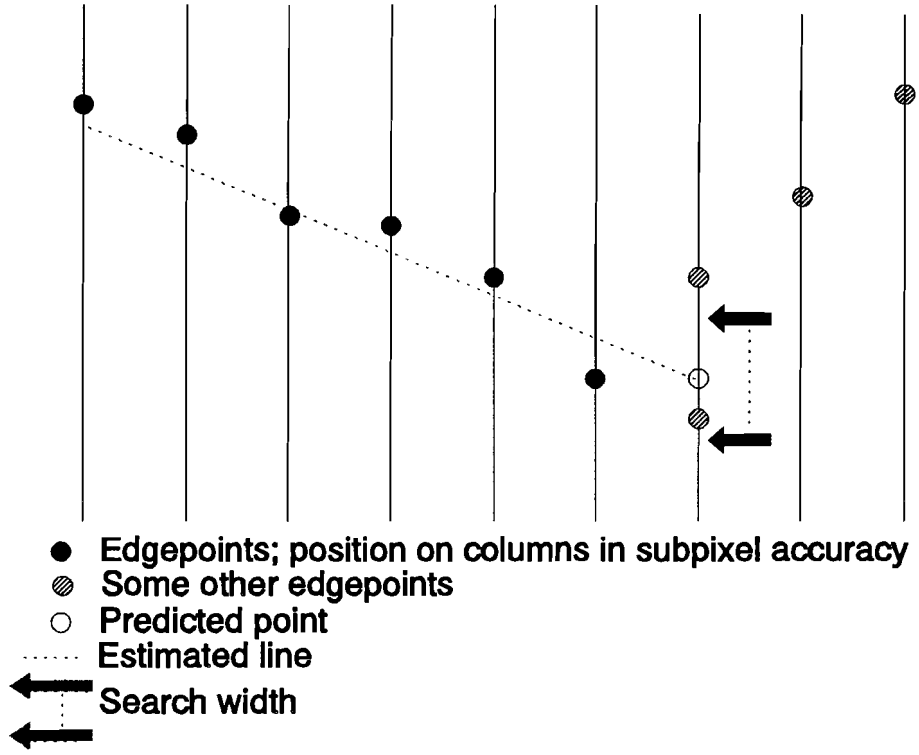


Fig.17. Edgepoint prediction and search for edgepoints on current column.

This is illustrated Fig.17. Here the edge image in vertical direction is being processed, so the successive columns will be examined for edgepoints (column-search). The search-width determines the maximum distance from the predicted point. In Fig.17, the edgepoint that lies underneath the predicted point is just within the search-width, so this point will be the next edgepoint to be added to the set of edgepoints. The search-width can be parameterized and replaces the connectivity in Sect. 3.1.4. If there are more edgepoints present within the search-width, then the edgepoint closest to the predicted point will be chosen to be the next edgepoint.

A problem could occur when the direction of the estimated line (through the edgepoints of the corresponding edge) is almost vertical ($(\varphi \approx \pi/2)$ or $(\varphi \approx 3\pi/2)$) in case of column-search or when the direction of a line is almost horizontal ($(\varphi \approx 0)$ or $(\varphi \approx \pi)$) in case of row-search. The distance between the last edgepoint found and the predicted point could become very large which decreases the reliability of the algorithm. This problem can be solved as follows. The edge following process will be stopped when the direction φ of a

line, estimated through a set of edgepoints, fulfils:

$$\varphi: \begin{cases} 150^\circ<\varphi<210^\circ \vee -30^\circ<\varphi<30^\circ & \textit{if row-search} \\ 60^\circ<\varphi<120^\circ \vee 240^\circ<\varphi<300^\circ & \textit{if column-search} \end{cases} \quad (27)$$

When the edge following process stops because of this, then the line segment concerned will probably be found by processing the edge image in the other direction.

3.2.2 Other modifications.

The search for starting points and the estimation of the initial search-direction is the same as described in Sect.3.1.1. The subpixel accuracy which is available will not be used for this.

The LSE line estimation and the perpendicular projection of points on lines is also essentially the same as described in Sect.3.1.2. However, because of the fact that the coordinates of the edgepoints are in subpixel accuracy, the line segments obtained by the LSE line estimation, will be more accurately placed. This is one of the advantages of LINEX2.

The filling of edge interruptions is the same as described in Sect.3.1.4. The buffer of simulated edgepoints now contains points given in subpixel accuracy.

The operation described in Sect.3.1.5 to avoid double line segments will be the same, but will not be used in practice. This is because the results of Lee's edge detector are one pixel thick and because successive row/columns are examined. When a line segment has been found, then no edgepoints that belong to the considered edge, will be left unprocessed.

3.3 Improving line images using a postprocessing algorithm.

The line images obtained by the algorithms LINEX1 and LINEX2, described in the previous chapters, could be improved by another algorithm. E.Kaptein developed and implemented an algorithm that can be used for this postprocessing. The input of this algorithm, called "CONNECT", is a set of line segments. The algorithm reduces the number of line segments significantly, so the result is easier processed on a higher level. The post-processing performed by this algorithm will now be described in detail.

For each line segment, a rectangular line-environment is defined, as illustrated in Fig.18. The size of these rectangles is defined by the parameters WD and LD ("width-distance" and "length-distance"). Line segments will be merged if and only if:

- A line segment passes the line-environment of an other line segment, and
- The difference between the directions of two such line segments is lower than a certain value. This value will be called MAX_ANGLE.

It is possible that more than 2 two line segments will be merged into one line segment.

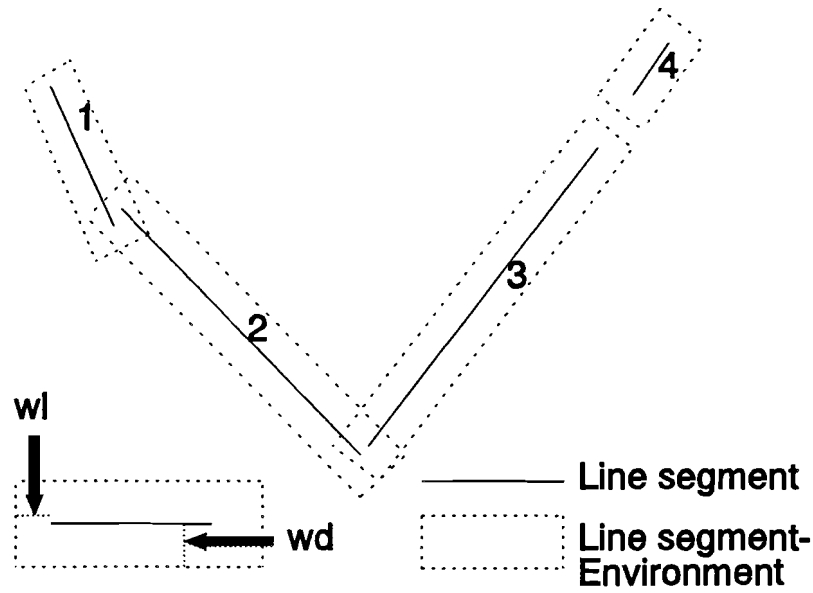


Fig.18. Merging line segments using CONNECT.

In Fig.18, line segment 2 enters the line-environment of line segment 1. If the angle between these line segments is below MAX_ANGLE then these two line segments will be merged. Line segment 2 and line segment 3 both enter the line-environments of each other, but they will probably not be merged, because the angle between these line

segments is probably much higher than MAX_ANGLE. The angle between line segment 3 and line segment 4 is probably below MAX_ANGLE, but they will not be merged because the line-environments do not overlap.

Merging line segments into one line segment is done as follows. Suppose we have a set of N line segments which should be merged into one line segment. Only the begin- and endpoints of the set of line segments are considered in the estimation and the length of the line segments is L(i) ($0<i\leq N$). For L(i) the length of the line segment is chosen. A straight line, described as $ax+by+c=0$, is estimated just as described in Sect.3.1.2. The set of points, through which the new line should be estimated, is as follows. The begin- and endpoint of each line segment i is added L(i) times to the set of points. Now begin/endpoints of long line segments are considered more important then those of short line segments. a,b and c can now be computed using the formulas (8) to (16). Finally the begin- and endpoint of the estimated line segment should be estimated. This can be done by projecting the begin- and endpoints of all line segments perpendicularly on the estimated line (see Sect. 3.1.2). The two points farthest from each other on the estimated line will be chosen as the begin- and endpoint of the line segment. This is illustrated in Fig.19, where two line segments will be combined into one line segment. In Sect.3.6 the effect of CONNECT will be illustrated.

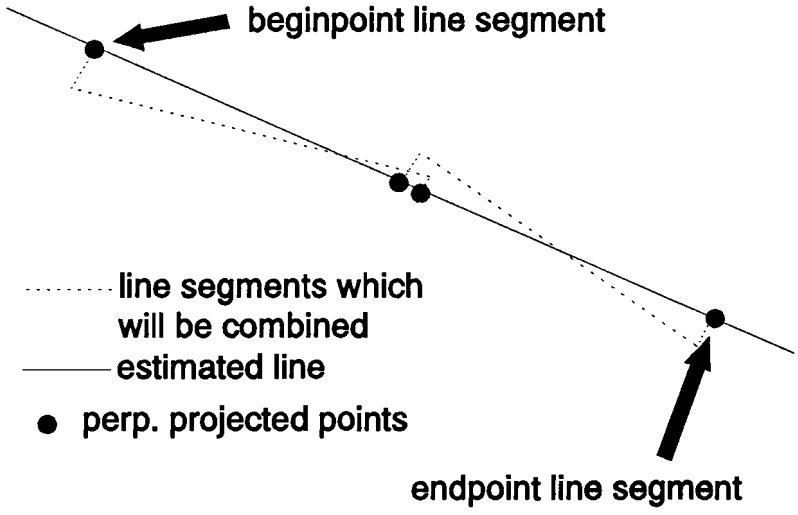


Fig.19. Determination of begin- and endpoint of line segment.

It is especially useful to use CONNECT, after combining two or more line images as described in Sect. 2.1. This is the case because the result will contain many double line segments (line segments that represent the same edge). Double line segments will commonly be combined into one line segment.

3.4 Implementation.

LINEX1 (the general line extraction algorithm (Sect.3.1)) and LINEX2 (the algorithm that makes optimal use of the results of Lee's edge detector (Sect.3.2)), have been implemented in Microsoft C on a 486 PC. The flowchart of LINEX1 (Fig.20) illustrates the structure of the algorithm. When variables are closed within brackets then these are values given to or given back from the corresponding function. For example the statement in the flowchart 'Restore Image(gradient)' corresponds with a call to the function that restores the edge image determined by the variable 'gradient' (gradient ϵ {vertical,horizontal}). Only variables which are needed to understand the flowchart are displayed. The "Incr."-operator means increment by 1 in this flowchart.

The scanpoint is defined as the point from which the lexicographical search for the next unprocessed point (=starting point) will take place (see Sect.3.1.1). When (x_s,y_s) is the current scanpoint, then at least the following points are already processed:

$$(x,y): \quad (x < x_s) \vee ((x=x_s) \wedge (y < y_s)) \qquad (28)$$

The pointbuffer is a buffer that will contain the simulated edgepoints (see Sect.3.1.4) and the last edgepoint that has been found. So normally when no disturbances are present, this buffer will just contain one point: the edgepoint that has been found. It can be seen that a line segment has been found when the predicted point is out of range or when 'disturbance-count' exceeds 'disturbance-threshold'. After checking if the length of the line segment is not smaller then the minimal length, the line segment will be stored and the next starting point will be searched. The variables Sumx, Sumy.... Sumyy contain the values defined in (8) (see Sect.3.1.2).

In LINEX1, the following values are parameterized:

- **Step-Size**. This parameter is used for the prediction of edgepoints (see Sect.3.1.3).
- **Disturbance-Threshold**. This parameter, which can also be found in the flowchart, represents the viewing distance expressed in steps with the current 'step-size' (see Sect.3.1.4).
- **Connectivity**. Determines the size of the region around the predicted point, which will be examined for edgepoints (see Sect.3.1.3).
- **Min-Length**. This parameter, which can also be seen in the flowchart, determines the minimal length of the line segment which should be stored.
- **Rectangle-Size**. This parameter determines the size of the rectangle around a processed edgepoint, as described in Sect.3.1.5. Edgepoints within this rectangle will be defined as processed.

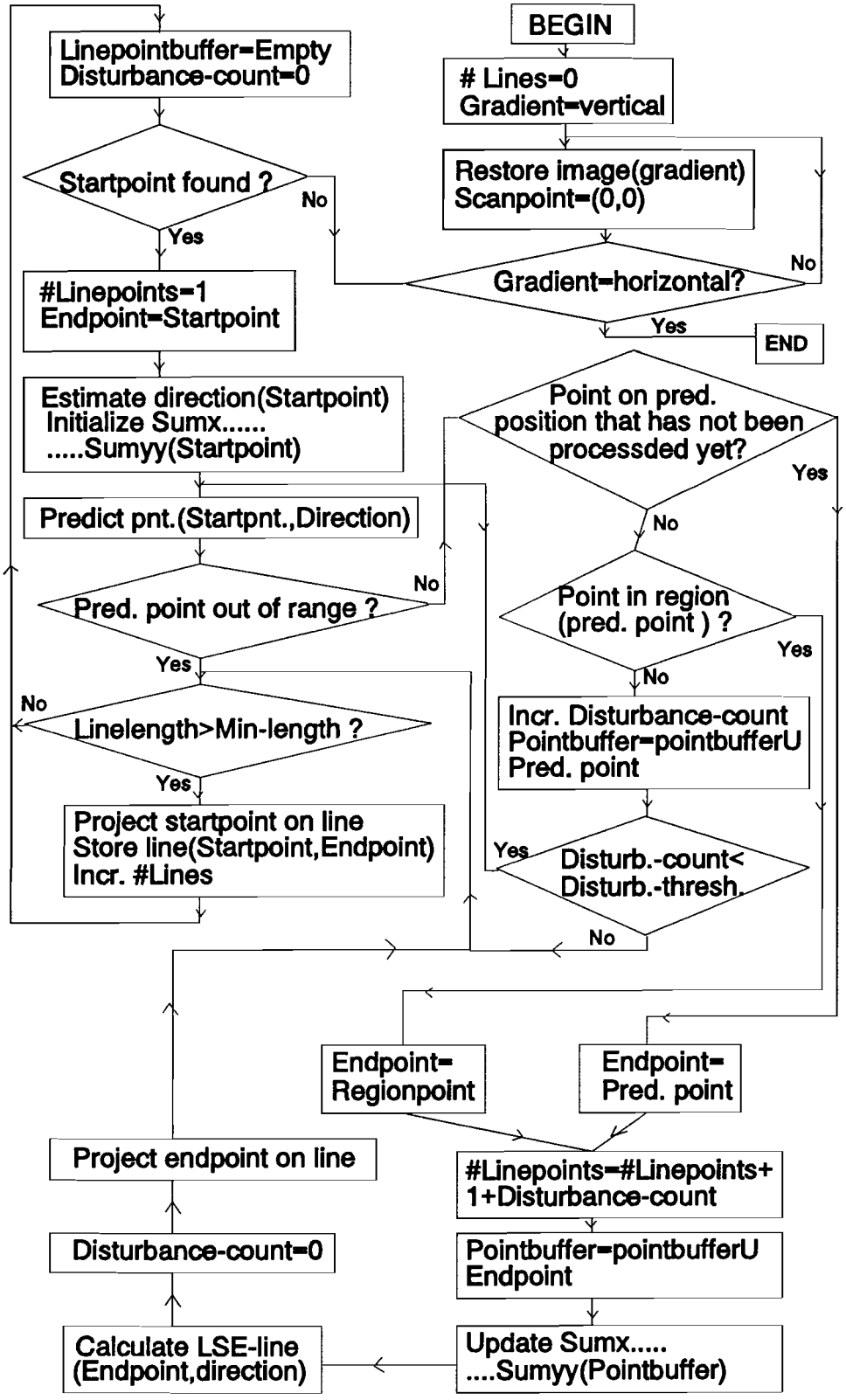


Fig.20. Flowchart line extraction algorithm LINEX1.

In LINEX2 (see Sect.3.2), the parameter "Step-Size" will not be used and "connectivity" will be replaced by:

- **Search-Width**. This parameter determines the maximum distance from the predicted point on the current row/column (see Sect.3.2.1).

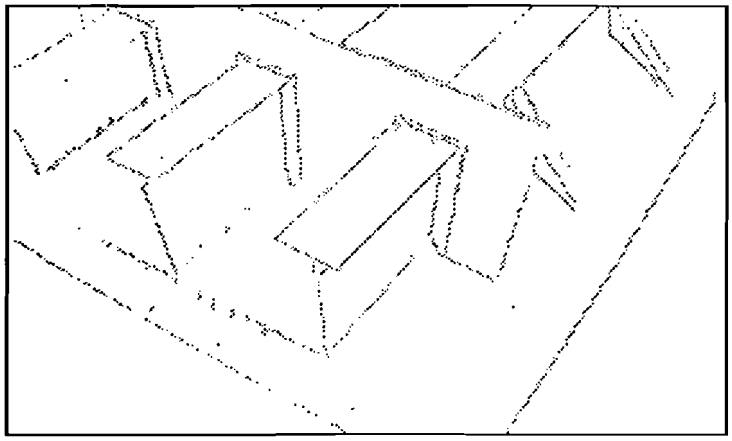
The biggest data structure which will be used is the edge image. Because of the fact that the edge images in horizontal direction and in vertical direction will be processed separately, these images will be restored successively in memory.

In Microsoft C it is just possible to store one 512x512 image (one byte per pixel) in the memory of the PC. Another way is to restore the edge images in a video memory of an image processor. We used the Philips Single Board Image Processor (SBIP2) for this. Pixels can be written/read out of such a video memory.

The execution time proved to be much smaller when the edge images would be stored in the PC memory (see Sect.3.5).

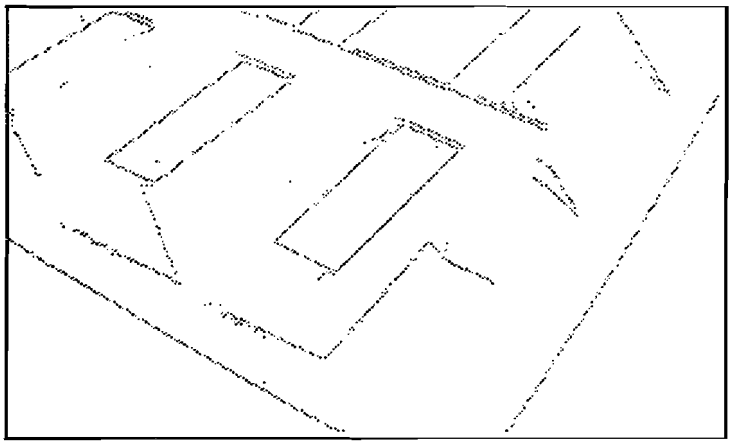
3.5 Results.

In this Section some experiments and results of the developed line extraction algorithms will be discussed. Both LINEX1 and LINEX2 extract line segments from both edge images (edgepoints detected in vertical resp. horizontal direction) separately, before merging the two sets of line segments into the final line image. This will now be illustrated. Two edge images, one with the edgepoints detected in horizontal direction and one with the edgepoints detected in vertical direction, are showed in Fig.21.



a.

Edgepoints detected in horizontal direction



b.

Edgepoints detected in vertical direction

Fig.21. Edge images.

The line segments extracted from these edge images are shown in Fig.22. It is clear that the directions of the line segments in both images satisfy the constraints stated on p.19. Merging these sets of line segments into one line image results in the final line image of Fig.23.

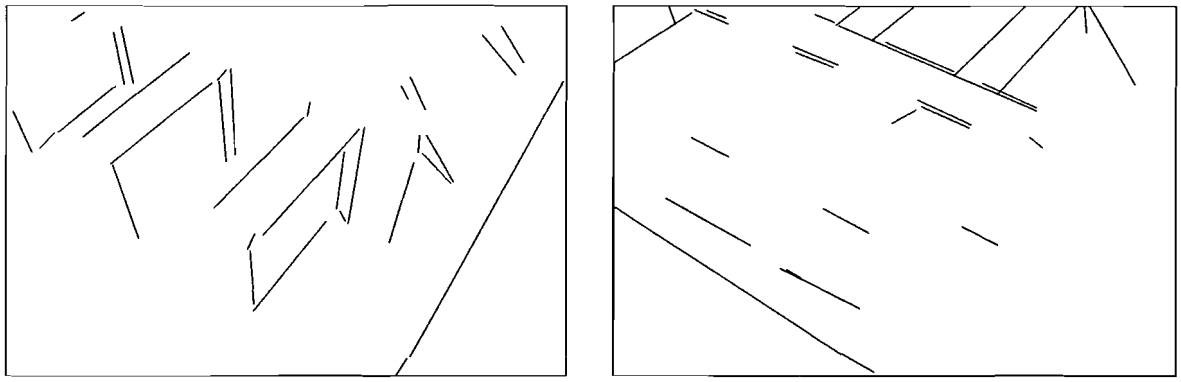


Fig.22. Horizontal orientated lines (left) and vertical orientated lines (right)

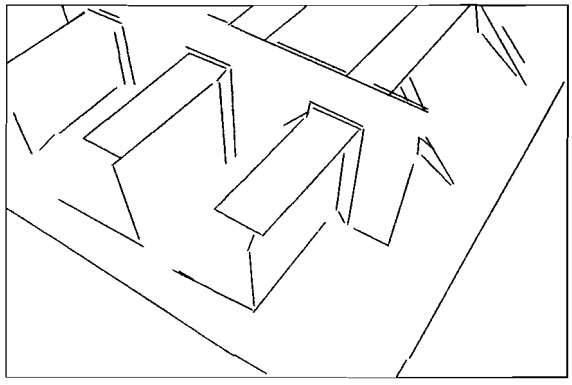


Fig.23. Line image after merging the line images of Fig.22.

The edge images of a typical scene will be used to test the two algorithms LINEX1 and LINEX2, described in Sect.3.1 and Sect.3.2 respectively. This edge image contains the edges of a workpiece with a plate, a web , three T-stiffeners and a strut (Fig.24). First the influence of the parameters which control the algorithms will be discussed. The best parameter settings for the algorithms will be used in Sect.3.5.2, where the performance of the two algorithms will be compared.

3.5.1 Changing parameters.

The parameters which control the algorithms are (see Sect 3.4):

- Connectivity (conn) in LINEX1 or Search-width (sw) in LINEX2
- Min-Length.
- Rectangle-Size (rs).
- Step-Size (ss).
- Disturbance-Threshold (dt).

The abbreviations of these parameters will each time be given in the caption of the line images which will be displayed in this Section.

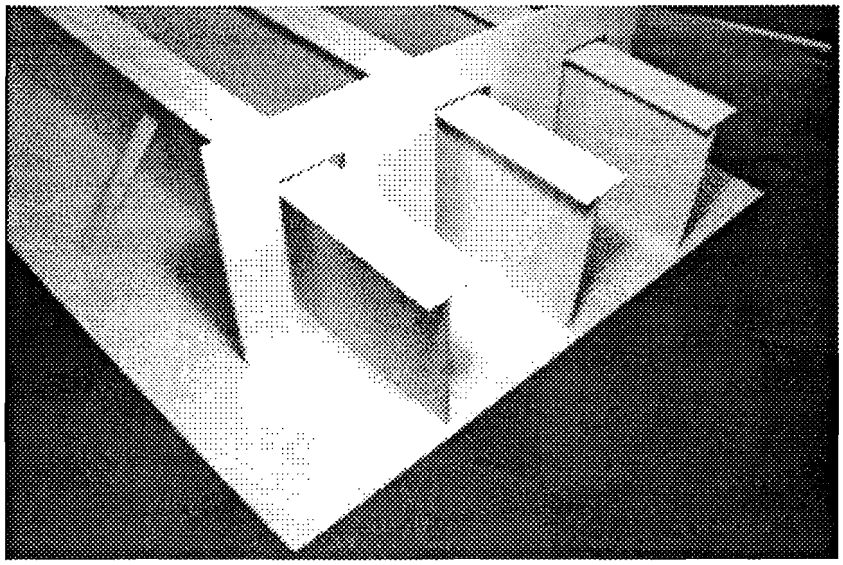


Fig.24. Gray level image that will be used for experiments.

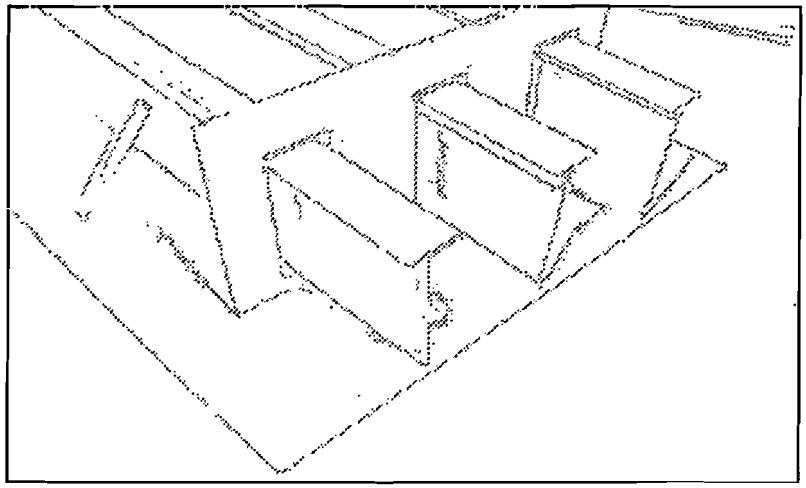
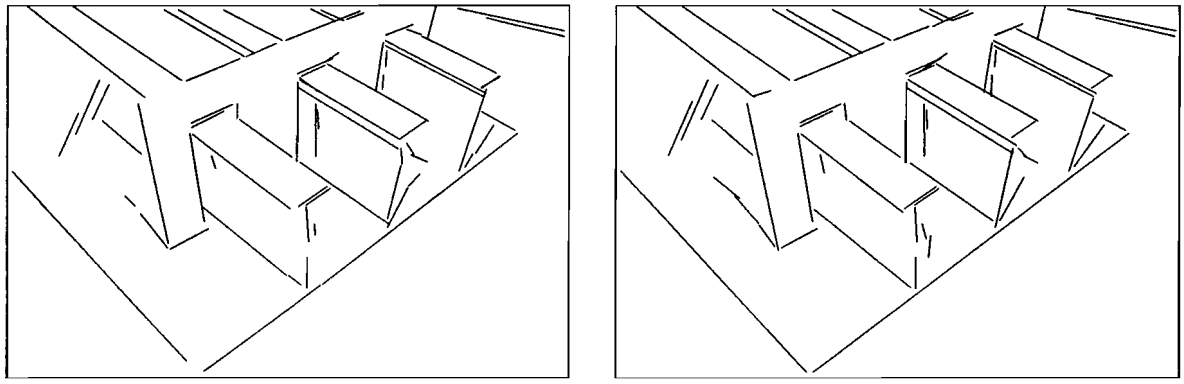


Fig.25. Edge image of the scene of Fig.24, that will be used for experiments.

Now, first the results of LINEX1 will be discussed. It appeared that the connectivity which is used to search the environment of the predicted edgepoints does not influence the results very much. Fig.26 illustrates two line images. The only difference in the parameter sets, used in the line images of Fig.26a and Fig.26b is the connectivity, which equals 8 and 24 respectively. In case of small connectivity, the edge following process of a certain edge stops earlier. So edges will sometimes be segmented in more line segments. This can for example be seen on the front side of the vertical plate of the middle T-stiffener and on the shadow of the right T-stiffener.

In Fig.28 it is illustrated that the disturbance threshold could be increased to deal with missing edgepoints. When comparing this line image with the line image of Fig.26a, one sees that the top side of the web in Fig.28 consists of longer line segments, because some interruptions have been filled.



a. ss=1, dt=6, conn=8, rs=1 b. ss=1, dt=6, conn=24, rs=1

Fig.26. Line images with: a. Connectivity= 8; b. Connectivity= 24.

ss=3
dt=3
conn=8
rs=2

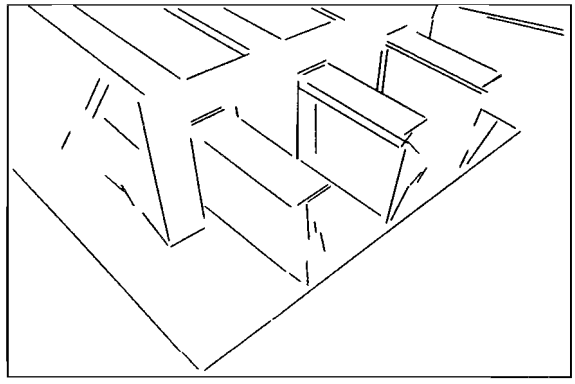


Fig.27. Line image with Stepsize=3.

ss=1
dt=12
conn=8
rs=1

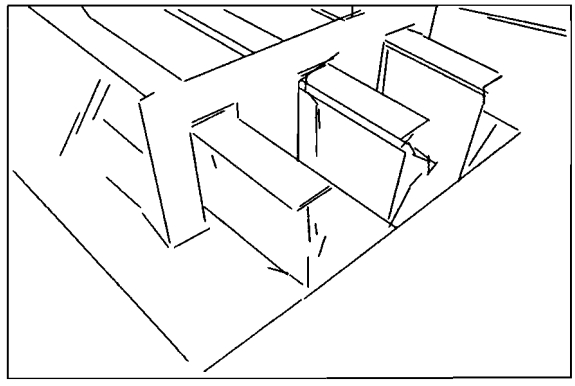
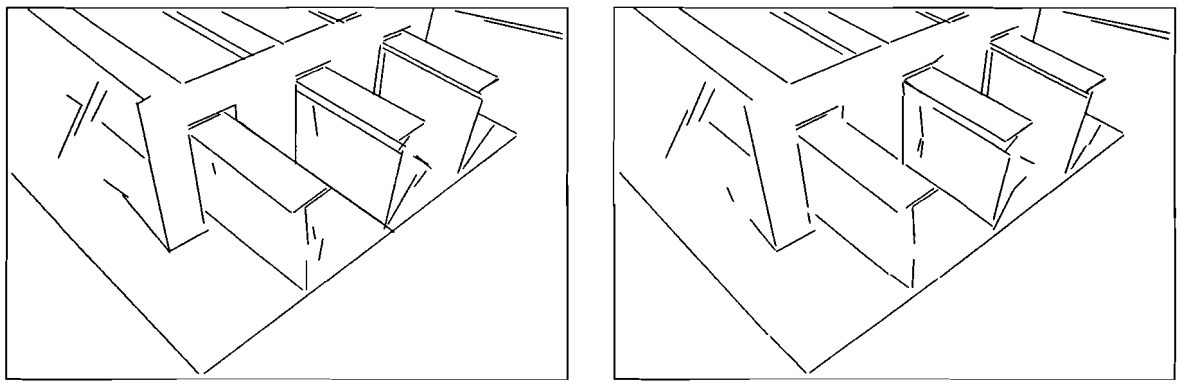


Fig.28. Line image with disturbance-threshold 12.

As expected, using stepsize 1 gives better performance than using a bigger stepsize. This is illustrated in Fig.27 where stepsize 3 is used. Comparing this image with the line image of Fig.26a illustrates that using stepsize 1 results in line images with a little better quality (Considering the right T-stiffener for example). Resuming the experimental results so far, an optimal parameter set (regarding the quality of the line images) for LINEX1 is as follows: Step-Size=1, Disturbance-Threshold=6, Connectivity=8, Rectangle-Size=1.

Now the results of LINEX2 will be discussed. The parameter Connectivity is replaced by the parameter Search-width. In the line images of Fig.29 it is illustrated that increasing the Search-Width corresponds with increasing the connectivity.



a. ss=1, dt=6, sw=2, rs=1 b. ss=1, dt=6, sw=1, rs=1

Fig.29. Line images with: a. Search-Width= 1; b. Search-Width= 2.

Changing the other parameters has the same effect as changing them in LINEX1. An optimal parameter set for LINEX2 is as follows: Step_size=1, Disturbance-Threshold=6, Search-Width=1, Rectangle-Size=1.

3.5.2 The two algorithms compared.

Now the performance of LINEX1 and LINEX2 are compared. When comparing the line images made with LINEX1 (Fig.26) with those made with LINEX2 (Fig.29), one can conclude that there is hardly any difference in the quality of the line images. However, the line segments in the line images made with LINEX2 are probably more accurately placed, because the edgepoints through which the line segments are estimated are in subpixel accuracy (see Sect.3.2).

If the edge images were stored in the video memory of the SBIP2, the algorithms need large execution times. This is because reading separate pixels from the video memory happens very slowly. The execution time proved to be much smaller when the edge

images would be stored in the PC memory instead of video memory. Then the execution time of both LINEX1 and LINEX2 is about 20 seconds for typical line images, like the ones displayed in the previous section.

3.5.3 Postprocessing with CONNECT.

To illustrate the effect of the algorithm CONNECT, described in Sect.3.4, the line image of Fig.29a will be processed with this algorithm. The parameters used in CONNECT are: MAX_ANGLE= 10, LD=20 and WD=3. The resulting line image is shown in Fig.30. The number of line segments has decreased from 74 to 56. CONNECT is a very fast algorithm. On a 486 PC, the execution time is in general less than one second.

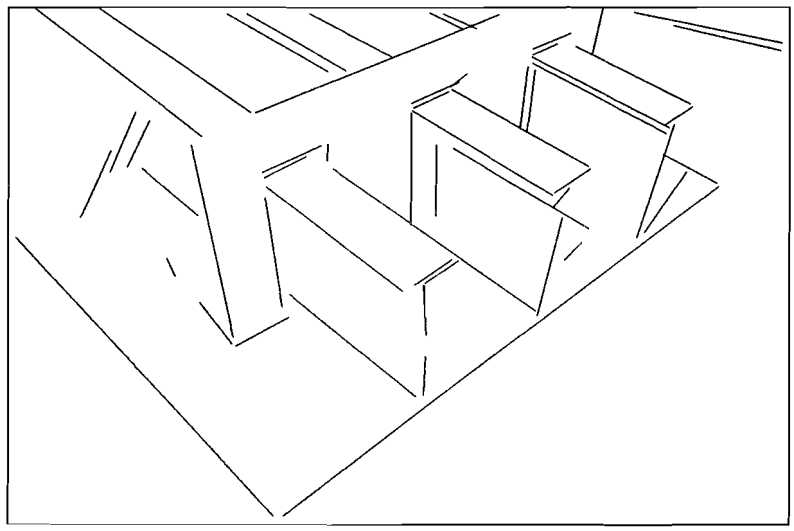


Fig.30. The line image of Fig.29a, after postprocessing with CONNECT.

3.5.4 Combining images with different lighting conditions.

In Sect.2.1, it was mentioned that it is possible to combine scenes of a workpiece with different lighting conditions. When the desired edges are visible in at least one of these images, then all the edges that describe the 3D workpiece from the current point of view are present. Line images from both of these scenes could be combined into one line image. This final line image will contain more information about the geometrical structure of the workpiece, but will also contain more unwanted edges like shadows.

An example of this will now be discussed. A workpiece containing a plate, 3 T-stiffeners, 2 webs and a strut is viewed by a camera from a certain position. Two images are acquired with different lighting conditions (Fig.31). Then from both of these images edgepoints were detected and line segments were extracted. The resulting line images are shown in Fig.32.

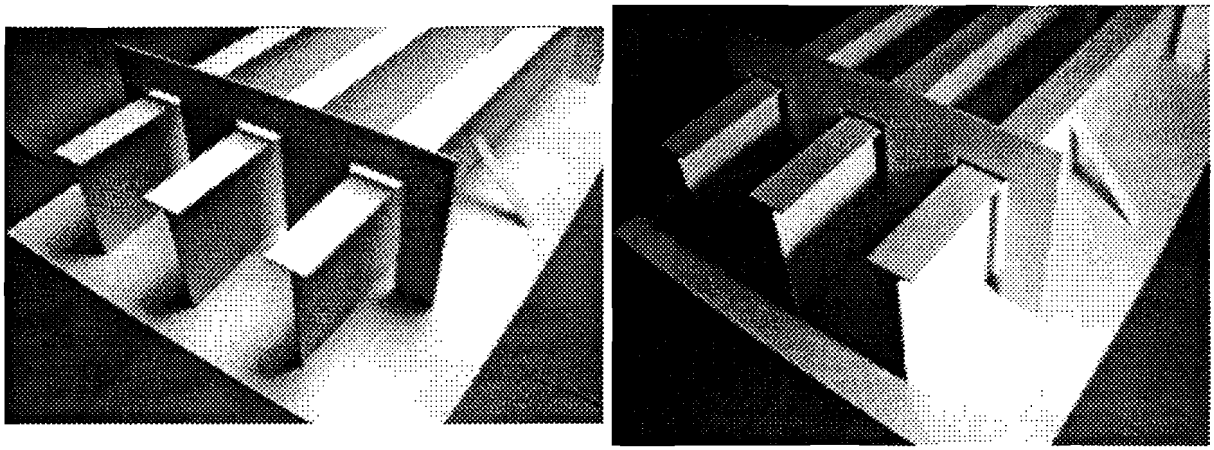


Fig.31. Gray level images of a workpiece with different lighting conditions.

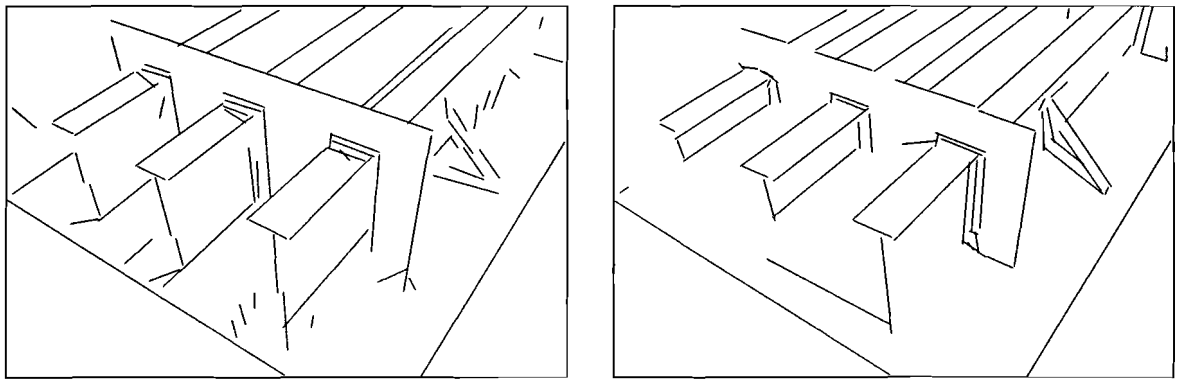


Fig.32. Line images from the same scene, with different lighting conditions.

When these line images are merged and this result is post-processed with CONNECT, the line image of Fig.33 appears.

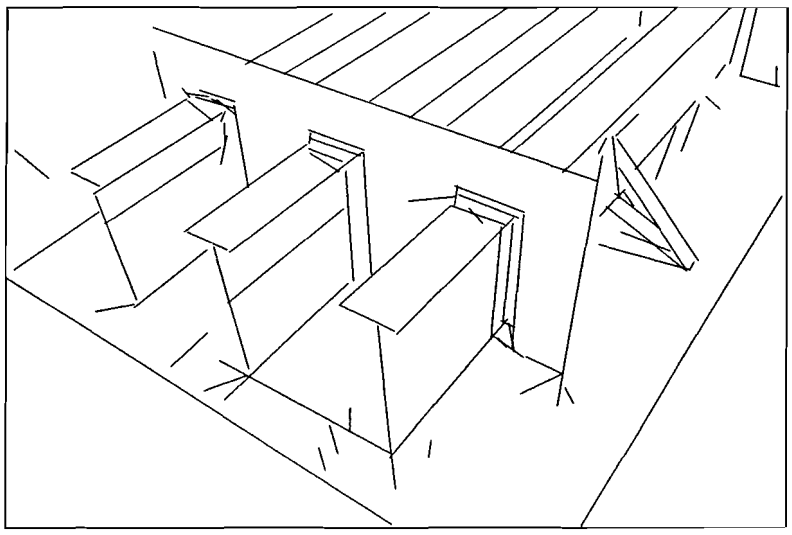


Fig.33. Resulting line image.

Important line segments that are missing in Fig.32 (left) are:

- The boundary between the plate and the front web.
- The boundary between the plate and the right T-stiffener.

These line segments are present in Fig.32 (right). The resulting line image (Fig.33) contains all these line segments.

3.5.5 Accuracy of the low level vision.

The accuracy of the low level vision system (edge detection and line extraction with LINEX2) can be measured by defining a synthetic gray image. The gray image which has been made for this purpose is illustrated in Fig.34. The edges of this image are ideal step edges [2] and can be represented by 4 line segments.

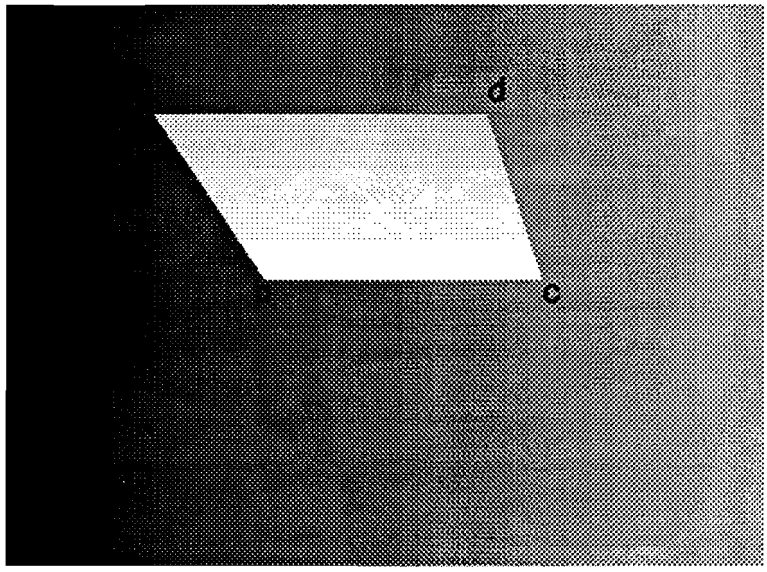


Fig.34. Synthetic image used for accuracy-measurements.

The subpixel accurate coordinates of the corners of this polygon are as follows:

Point a: (99.5 , 99.5)
Point b: (250.5 , 174.5)
Point c: (250.5 , 362.5)
Point d: (99.5 , 325.5)

The line image extracted from this image is illustrated in Fig.35. The accuracy of the 4 extracted line segments can be evaluated by projecting the endpoints of these extracted line segments perpendicularly on the ideal lines, which are determined by the subpixel

coordinates of the corners of the polygon.

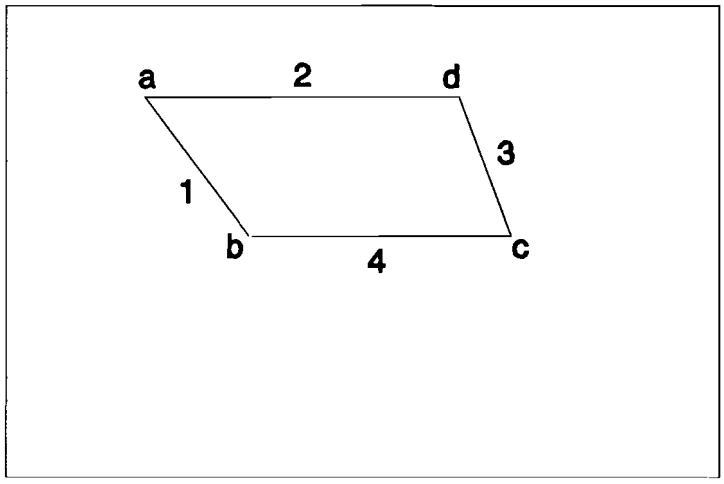


Fig.35. Line image of the gray image of Fig.34.

In Table 2, the subpixel accurate coordinates of the extracted line segments are compared with the ideal coordinates. The first column consists of endpoints of line segments, determined by the line segment number and the corner of the polygon (according to Fig.35). The second and the third column consist of the ideal position of the endpoint (the position of the corresponding corner of the polygon) and the position of the endpoint of the extracted line segment respectively.

The distance between the endpoints of the line segments and their perpendicular projections on the ideal lines (abbreviated as PPL) are put in the right column. These values can be computed by the method described in Sect.3.1.2.

Table 2. Accuracy measurements.

(Line, Point)	Ideal position	endpoint line segment	PPL
(1,b)	(250.5 , 174.5)	(250.00 , 174.63)	0.34
(1,a)	(99.5 , 99.5)	(100.15 , 99.70)	0.11
(2,a)	(99.5 , 99.5)	(99.95 , 100.00)	0.45
(2,d)	(99.5 , 325.5)	(100.03 , 325.00)	0.53
(3,d)	(99.5 , 325.5)	(100.12 , 325.53)	0.12
(3,c)	(250.5 , 362.5)	(250.00 , 362.99)	0.60
(4,c)	(250.5 , 362.5)	(251.02 , 362.00)	0.52
(4,b)	(250.5 , 174.5)	(250.99 , 177.00)	0.49

4 Extraction of elliptic arcs.

In Chapter 3 the extraction of straight line segments was discussed. Using this software it is possible to deal with 2D images of workpieces which were build up of parts like plates, stiffeners, webs and struts, because all edges in such a workpiece are projected into an image as straight line segments. However, later in the project the workpieces could also contain curved edges. For example workpieces from the STRINGER workpiece category [5]. Fig.36 shows a workpiece from this category.

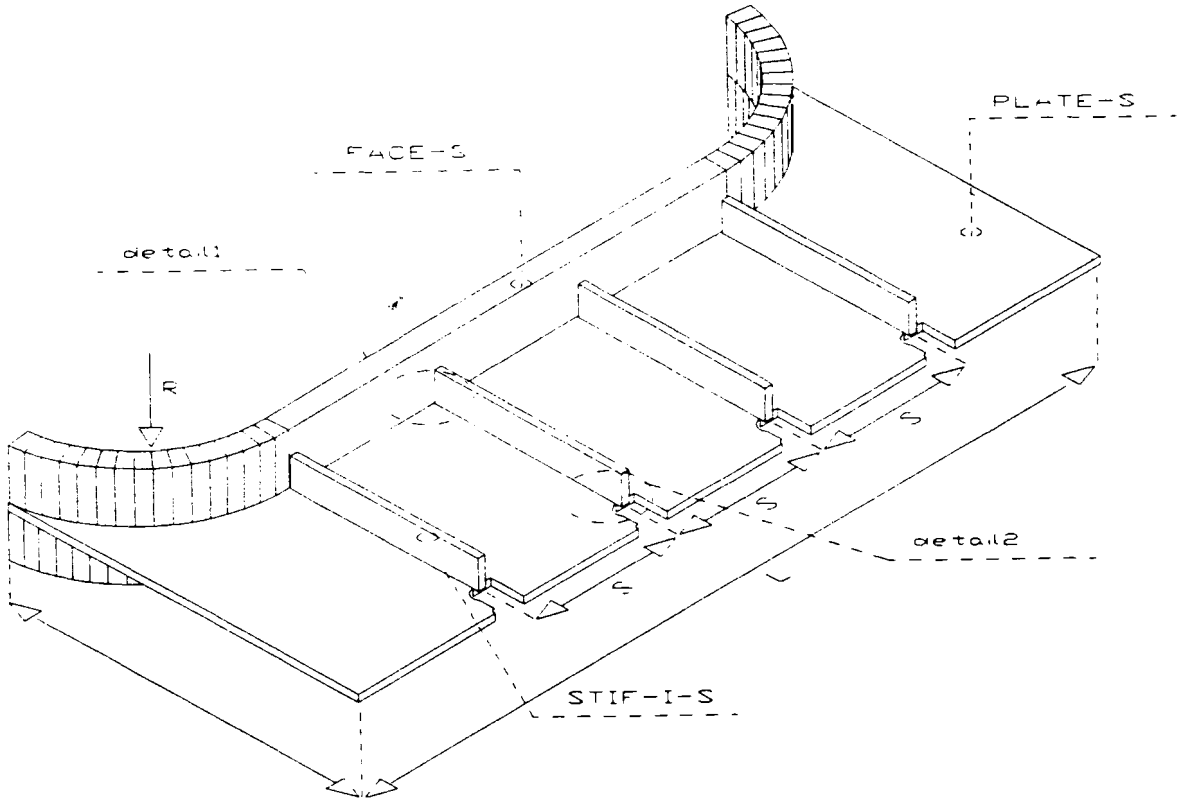


Fig.36. Workpiece from the STRINGER category.

Now consider a part of such a workpiece that could be represented as an elliptic arc in 3D space. Such part of an ellipse would still be a part of an ellipse, or it would become a straight line or a circle, when viewed by a camera (projected on a 2D plane). The algorithms described in the previous Chapter should be modified or a new algorithm should be developed to deal with this: the algorithm should extract both straight line segments and elliptic arcs.

To allow extraction of curved segments, it is useful to obtain first edge chains from the edge image. An edge chain is a set of connected edgepoints. In Sect.4.1 it will be described how the algorithms described in the previous Chapter could be modified to obtain edge chains. These edge chains could be segmented into straight lines and elliptic arcs or other curved segments; this will be described in Sect.4.2.

4.1 Obtaining edge chains.

The principle of the algorithms described in Chapter 3 is based on an edge following method. Normally, in case of such methods, edgepoints will first be linked into edge chains, before line segments and curved segments are estimated through the edgepoints of these edge chains [7,16,17]. The algorithms described do not carry out all these steps (see Fig.7, p.21). However, when a straight line segment has been found, a set of edgepoints is available through which this line segment was estimated. These sets of edgepoints consist of edgepoints which are not necessarily connected. If it would be desired, some points could be added to these sets of edgepoints to obtain connected edge chains. When the algorithms described in the previous Chapter would be modified to obtain such edge chains, then only edge chains which correspond with straight lines are obtained. To extract curved segments, the algorithm should be further modified to obtain not only straight edge chains, but also curved edge chains.

Next, we will discuss the modifications which could be made to the algorithms as described in the previous chapter:

a) To allow the extraction of curved edge chains, both edge images should be used simultaneously because the edgepoints which form a curved edge chain are in general not available all in one edge image.

b) In Sect.3.1.3 it was described that edgepoints were predicted from the last point found, perpendicularly projected on the estimated line. The modification is to predict the next edgepoint simply from the last edgepoint found.

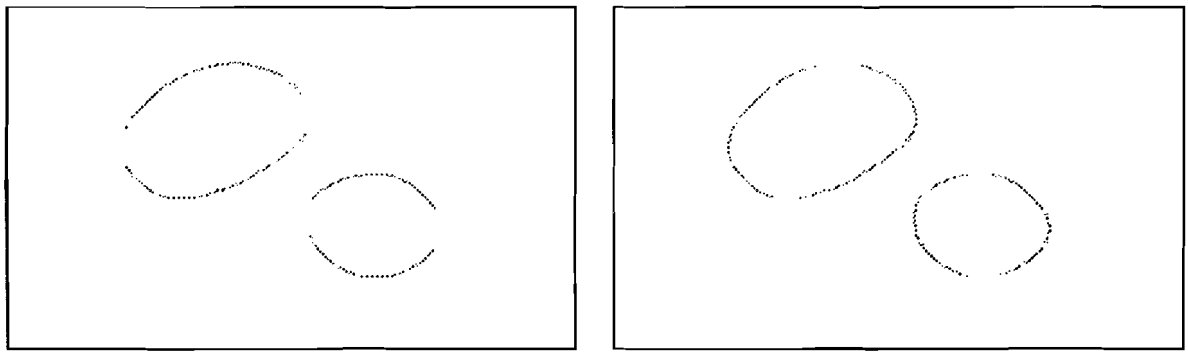
c) The constraints defined in (23) and (24) (p.30) could be neglected or adapted.

d) The search-direction for the next edgepoint should depend on the position of the last 2 (or eventually more) edgepoints which have been found.

Modifications *b)*, *c)* and *d)* are necessary to allow also the extraction of curved edge chains instead of straight edge chains only. These modifications are easy to implement.

Modification *a)* is the biggest modification. A solution is to combine both edge images first with the OR-operation (see Fig.6, p.20). Then all edgepoints are available in one edge image, but edge chains in this resulting edge image are in general not just one pixel thick, like described on p.19. To make optimal use of the subpixel accurate positions of the edgepoints, it is better to use both edge images separately instead of first combining them (like the algorithms described in Chapter 3).

In case of straight edge chains, the edgepoint forming such an edge chain are available in one of the edge images. The direction of the estimated line determines which of the edge images. A curved edge chain however generally consists of a set of edgepoints not available in just one edge image, because the edge detector used is one dimensional. Fig.37 illustrates two edge images of a scene with 2 white elliptic figures with a black background. For both curved figures one edge chain should be obtained. These edge chains represent the boundaries of the figures. It is clear that the edgepoints of which these edge chains should consist are not available in just one edge image. So while following such edge, the edge images should be successively used. A rule should be defined that determines at which moment in the edge following process the algorithm should switch over to the other edge image.



Edgepoints detected in vertical direction *Edgepoints detected in horizontal direction*

Fig.37. Edge images of 2 curved figures.

A practical problem is that it is not possible to store 2 images of 512^2 bytes in the computer's memory at the same time within the MS DOS operating system. It is possible to store both (or one of them) in the video memory of the image processor, but the execution time will then become much larger, as stated in Sect.3.4.

4.2 Approximation by straight lines and elliptic arcs.

To approximate the obtained edge chains by straight lines and arcs, Turner [20] proposed the ψ-s curve for finding straight lines and arcs. For every edgepoint in the set of edgepoints of which an edge chain consists, ψ is defined as the direction of a tangent to the edge chain. Since the edge chain is not always continuous, the tangent is approximated by the direction of the line between the edgepoint and the one that is several points behind (or ahead of) it. ψ is plotted against s, the arc length of the part of the edge chain traversed from the starting point. Horizontal lines in the ψ-s curve correspond to straight lines on the edge chain considered. Other straight lines correspond to arcs, since ψ is changing at a constant rate. Thus the problem of segmentation of an edge chain into straight lines and arcs is reduced to that of segmentation of the ψ-s curve into straight lines. Turner used a splitting method to obtain this segmentation.

As already stated, projections of a workpiece on a 2D image plane, will contain parts of ellipses instead of circular arcs. An elliptic arc yields a curve in the ψ-s curve. A better method is necessary for locating straight lines and simple smooth curves to be approximated by elliptic arcs. Shirai[17,18] proposed curvatures for finding such segments. The curvature at a point is approximated by the 'k-curvature', the angle made between the lines from the point to that k points ahead and to that k points behind (Fig.38).

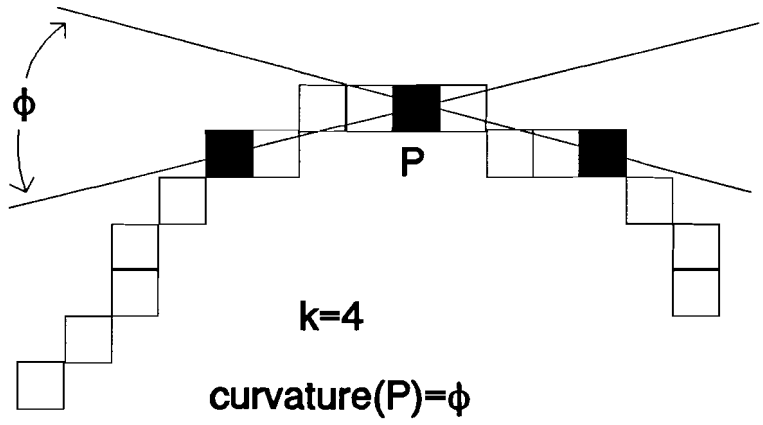


Fig.38. k-curvature.

The curvature ϕ is also plotted against s. An example of the ψ-s curve is shown in Fig.39. Horizontal straight lines in the ϕ-s curve correspond to arcs (as a special case, $\phi(s)=0$ corresponds to straight lines). Peaks in the ϕ-s curve correspond to points where

the tangent is discontinuous. Based on these facts, segmentation of an edge chain proceeds as follows.

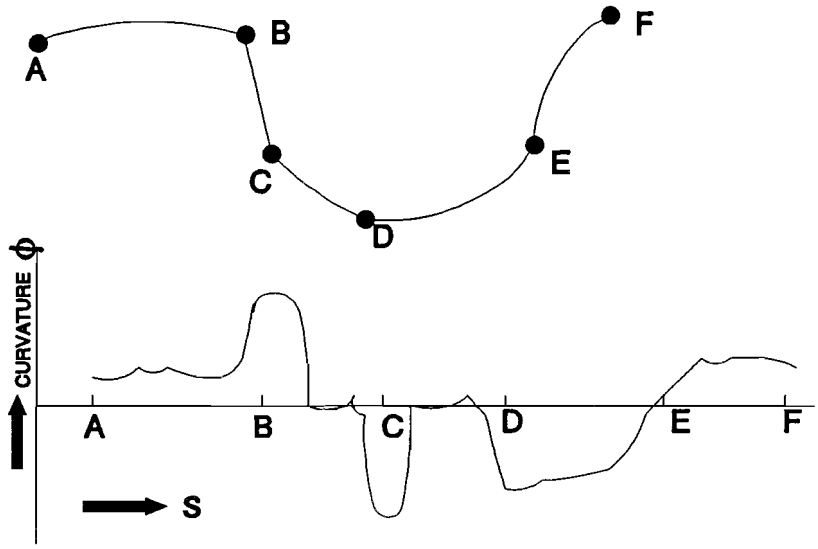


Fig.39. ϕ-s curve.

a) Find obvious candidates for knots and segments. For this, the ϕ-s curve can be used (the knots are the endpoints of the segments).

b) Classify temporary segments as straight lines, curves or unknown. Try to decompose long curves into straight lines and curves.

c) Merge adjacent segments if possible.

d) Determine positions of knots precisely.

For a more detailed description, see [18].

When a segmentation is obtained this way, the next step is to fit a straight line or a curve equation to each set of edgepoints: the problem is to determine the optimal equation for each set of edgepoints. The optimal equation depends on the evaluation function that represents the goodness of the fit. In [18], three typical evaluation functions are introduced and methods of fitting them are described:

- Using errors along a single axis.
- Using errors of line equations with two variables.
- Using distance from each point to fitted line.

In [16], a method is described that first segments the edge chains into straight line

segments. Ellipses are then fitted to the line data. This is much faster than curve fitting directly to the edgepoints since the lines provide a great reduction in data. Also, a method for fitting ellipses to arbitrary edge chains has been developed in [16], based on an iterative Kalman filter. This is guaranteed to produce an elliptical fit even though the best conic fit may be a hyperbola or parabola.

In [11] a new transform for curve detection, called the Curve Fitting Hough Transform (CFHT), is proposed. In the conventional Hough Transform (HT) and its variants, both storage and computation grows exponentially with the number of parameters. The CFHT is advantageous over the conventional HT and its variants in many ways.

5 Conclusions and recommendations.

5.1 Conclusions.

The low level vision part of the VSS, which consists of edge detection and line extraction, has been fully implemented and tested. Only workpieces without curved edges, i.e. workpieces from which the geometrical structure can be described with straight lines, can be successfully processed with this software.

The edge detector which is used in the VSS is an edge detector developed by D.Lee. This edge detector is an accurate one dimensional edge detector. It is applied in two directions and results in two edge images: edgepoints detected in vertical direction and edgepoints detected in horizontal direction. Features of Lee's edge detector are:

a) Edgepoints obtained by this edge detector form edge chains which are one pixel thick.
b) The position of the edgepoints is given in subpixel accuracy.
c) Some extra properties of edgepoints could be obtained with this edge detector, like edge direction and edge intensity.

To get a significant data reduction, features like straight line segments are extracted from the edge images. Two line extraction algorithms have been developed: LINEX1 and LINEX2. The principle used in these algorithms is an edge following method. LINEX2 makes use of the features **a)** and **b)** of Lee's edge detector. Both algorithms have been successfully tested. Both algorithms are fast and produce line images of good quality. Especially the line segments obtained with LINEX2 (using the subpixel accurate edge positions) are located precisely. LINEX2 will be used in the VSS. Line images can be postprocessed with the algorithm CONNECT. This algorithm reduces the number of line segments significantly, so it is easier to process the result on a higher level.

It has been described how images of a workpiece taken under different lighting conditions can be combined, to obtain more information about the 3D workpiece. Combining of images can be provided by merging the corresponding line images.

Later in the project, workpieces could also contain curved edges. The geometrical structure of such workpieces can not be described by only straight line segments. To obtain straight lines *and* elliptic arcs, the following steps will be done:

- obtain a set of edge chains out of the edge images.
- approximate these edge chains by straight line segments and elliptic arcs.

The user manual of the software is included in Appendix 1. In Appendix 2, the extraction of properties of line segments is described.

5.2 Recommendations.

Line images of good quality can be obtained by using the following operations:

1 - Edge detection with Lee's edge detector.
2 - Extraction of straight line segments with LINEX2.

To obtain also curved segments instead of straight line segments only (from the edge images obtained with Lee's edge detector), there are mainly two alternatives:

1) Using the curvature of edge chains to obtain a segmentation of these chains. Each segment is then approximated by either a line segment or an elliptic segment. The method described in [17] and [18] (see Chapter 4).

2) Using the method described in [16]. In this method, ellipses are fitted to line data. This is faster than curve fitting directly to edge data since the lines provide a great reduction in data. The straight line segments obtained with LINEX2 can possibly be used.

A problem in the low level vision appeared to be the bad visibility of some edges, due to bad illumination. This problem can be solved by the method described in Sect.2.1. and Sect.3.5.4. Considering this method, the relation between illumination and the visibility of edges should be further investigated.

6 References.

[1] Burns, J.B., A.R. Hanson and E.M. Riseman
Extracting straight lines. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.8, pp. 425-455 (1986)

[2] Buts, P.J.H.
Edge detection from noisy images for 3D scene reconstruction. M.Sc. Thesis, Eindhoven University of Technology, Measurement and Control Section (1993)

[3] Duda, R.O. and P.E. Hart
Pattern Classification and Scene Analysis. New York: John Wiley (1973)

[4] Esprit Project 6042: Hephaestos II. Technical Annex (May 1992)

[5] Esprit project 6042: Hephaestos II. Technical Appendix WP3 (March 1993)

[6] Farro, M.J.
Straight line detection in low contrast images: Theory and application. M.Sc. Thesis, Eindhoven University of Technology, Measurement and Control Section (1992)

[7] Giraudon, G.
An efficient edge following algorithm. Proc. 5th. Scandinavian Conference on Image Analysis, vol.2, pp. 547-554, Stockholm, Sweden (1987)

[8] Han, M.H., D. Jang and J. Foster
Identification of cornerpoints of two-dimensional images using a line search method. Pattern Recognition, vol.22, no.1, pp. 13-20 (1989)

[9] Haralick, Robert M. and Linda G. Shapiro
Glossary of Computer Vision Terms. Pattern Recognition, vol.24, no.2, pp. 69-93 (1990)

[10] Kaptein, E.M.J.
Knowledge-based Scene Analysis from Line Images. M.Sc. Thesis, Eindhoven University of Technology, Measurement and Control Section (1993)

[11] Liang, Ping
A New Transform for Curve Detection. Proc. third International Conference on Computer Vision, pp.748-751 (1990)

[12] Lie, W. and Y. Chen
Robust line-drawing extraction for polyhedra using weighted polarized hough transform. Pattern Recognition, vol.23, no.3/4, pp. 261-274 (1990)

[13] Marr, D.C. and E. Hildreth
Theory of Edge Detection. Proc. Roy. Soc. London, vol. B 207, pp. 187-217 (1980)

[14] Proceedings of SPIE- The international Society for Optical Engineering: Optics, Illumination and Image Sensing for Machine Vision, vol.728, pp. 35-129 (1987)

[15] Roberts, L.G.
Machine perception of three-dimensional solids, Optical and Electro-optical Information Processing, pp. 159-197. Tippet, J.T. et al. (eds.). MIT Press, Cambridge, MA (1965)

[16] Rosin, Paul L. and Geoff A.W. West
Segmenting Curves into Elliptic Arcs and Straight Lines. Proc. third International Conference on Computer Vision, pp.75-82 (1990)

[17] Shirai, Y.
Edge finding, Segmentation of edge and recognition of complex objects. Proc. 4th. International Joint Conference on Artificial Intelligence, vol.2, pp. 674-681, Tbilisi, Georgia, USSR (1975)

[18] Shirai, Y.
Three-dimensional computer vision. Berlin, Heidelberg, New York: Springer-Verlag (1987)

[19] Tjoa, E.K.L.
Geometrical Matching. M.Sc. Thesis, Eindhoven University of Technology, Measurement and Control Section (1993)

[20] Turner, K.J.
Computer Perception of Curved Objects Using a Television Camera. Ph.D. Thesis, School of Artificial Intelligence, Edinburgh University (1974)

[21] Williams, C.M.
Bounded straight-line approximation of digitized planar curves and lines. Computer Graphics and Image Processing, vol.16, pp. 370-381 (1981)

[22] Zucker, S.W., A. Rosenfeld and R.A. Hummel
An Application of Relaxation Labeling to Line and Curve Enhancement. IEEE Trans. on Computers, vol.C-26, p.394 (1977)

Appendix 1 User manual low level vision software

The low level vision part of the VSS is called "**LLVSS**". The software is written in C, and the source code consists of 4 modules:

- **VSS_LINE.C**: Contains functions concerning the line extraction part of the software. The line extraction algorithm used is LINEX2 (see Sect.3.2).
- **VSS_EDGE.C**: Contains functions concerning the edge detection part of the software. The edge detector used is Lee's edge detector (see Sect.2.2).
- **CONNECT.C**: Contains function concerning the postprocessing algorithm CONNECT (see Sect.3.4).
- **LLVSS.C**: Contains other functions and the **main** programm.

Input files needed are:

- **ACQ.SET**: Contains the parameter setting concerning the acquisition of images.
- **LEE.SET**: Contains the parameter setting concerning edge detection.
- **LINEX2.SET**: Contains the parameter setting concerning line extraction.
- **MASKS.DAT**: Contains data which LINEX2 needs (The templates of Fig.11, p.?).
- **FILENAME.TXT**: Contains the name that the output files will have.
- ***.CDF**: Camera description file; only needed in case of lens distortion correction.

Output files created are:

- ***.GRE**: Gray level image.
- ***.HPO**: Edge image in horizontal direction, with output format=position.
- ***.VPO**: Edge image in vertical direction, with output format=position.
- ***.HGR**: Edge image in horizontal direction, with output format=gradient.
- ***.HPO**: Edge image in vertical direction, with output format=gradient.
- ***.LDR**: Set of straight line segments.
- ***.LDP**: Set of straight line segments, including line properties.

Once this program is executed, a **menu** with the following options appears on the screen:

1 Acquire image
2 Acquire image; change parameters
3 Acquire image; view results
4 Edge detection
5 Edge detection; change parameters
6 Edge detection; view results
7 Line extraction
8 Line extraction; change parameters
9 Line extraction; view results

The actual low level vision operations are done by selecting options **1**, **4** and **7**. The results of these operations can be viewed by selecting options **3**, **6** and **9**. The parameters of the low level vision operations can be edited by choosing options **2**, **5** or **8**. The

following parameters can be edited with the following possible/typical values:

2 Acquire image:

- Lens distortion correction: **Yes** or **No**.

5 Edge detection:

- Filter size: **3..7**.
- Threshold: **4..40**.
- Output format: **Gradient and position** or **Position only**.

8 Line extraction:

- Disturbance threshold: **1..8**.
- Search-width: **1..3**.
- Rectangle size: **0..2**.
- Minimal length: **3..30**.
- Line properties: **Yes** or **No**.
- Connect: **Yes** or **No**.
- Length distance: **4..20**.
- Width distance: **2..12**.
- Max angle: **4..25**.

Appendix 2 Using properties of edges.

A feature of Lee's edge detector is that the algorithm can extract some properties of edges like gradient direction and gradient magnitude (see Sect.2.2). The gradient direction of an edgepoint has two possible values, because the edge detector is one dimensional. To use these properties during the line extraction process, Lee's edge detector should be executed such that it stores the edge images in both output formats (see Appendix 1). The line extraction algorithm LINEX2 can make use of these properties.

As already stated in Sect.2.2, the output format of Lee's edge detector that contains information about the gradient, is as follows. Where no edgepoint is present, the value of the pixel is 128. A pixel value lower than 128 corresponds with a transition from light to dark (seen from left to right or from top to bottom) and a pixel value higher than 128 corresponds with a transition from dark to light (seen from left to right or from top to bottom). Suppose p(x,y) is the pixel value of pixel (x,y). The gradient magnitude m(x,y) of edgepoint (x,y) is defined as

$$m(x,y) = |128 - p(x,y)| \qquad (29)$$

Suppose a line has been estimated through a set of n edge points (x_i,y_i), $1 \leq i \leq n$. The following line properties can now be obtained:

a) **The mean value of m(x,y)** of the edgepoints through which the line has been estimated:

$$\overline{m} = \frac{1}{n}\sum_{i=1}^{n} m(x_i,y_i) \qquad (30)$$

b) **The average gradient direction of a line.** There are two possibilities for this direction:

$$grad.dir. = \begin{cases} 0, & \overline{p} > 128 \\ 1, & \overline{p} \leq 128 \end{cases} \qquad (31)$$

where $\overline{p}$ is defined as:

$$\overline{p} = \frac{1}{n}\sum_{i=1}^{n} p(x_i,y_i) \qquad (32)$$

and where grad.dir=0 corresponds with a transition from light to dark (seen from left to right or from top to bottom) and where grad.dir=1 corresponds with a transition from dark to light (seen from left to right or from top to bottom). Edgepoints through which a line is estimated, can have different gradient directions. In that case propertie *b)* is not usefull.

c) **The variance of p(x,y)** of the edgepoints through which the line has been estimated. The variance of p(x,y), *var p*, is defined as

$$var\ p = \frac{1}{n}\sum_{i=1}^{n} (\overline{p}-p_i)^2 \qquad (33)$$

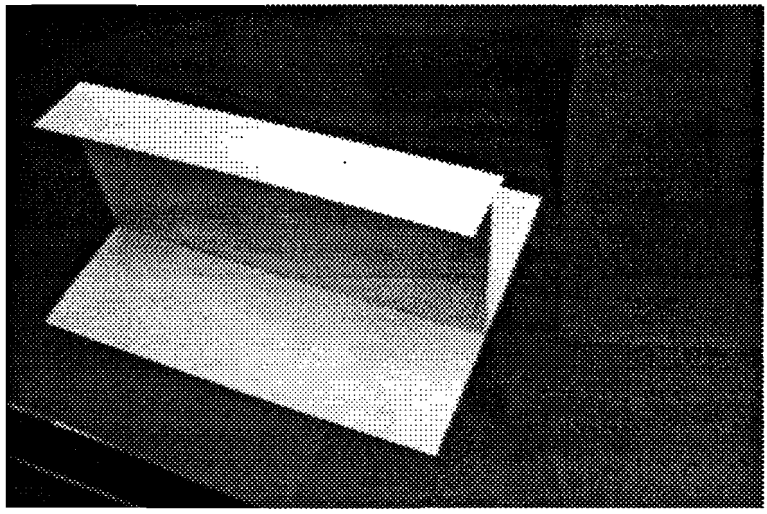


Fig.40. Gray level image used for experiment.

When edge detection and line extraction with properties are done on the gray level image of Fig.40, the results are as follows. Fig.41 illustrates for every line segment the value $\overline{m}$ according to (30). Darker lines correspond with a higher value of $\overline{m}$. Fig.41 illustrates for every line segment the value *grad.dir* according to (31). Black lines correspond with *grad.dir*=1; white lines correspond with *grad.dir*=0.

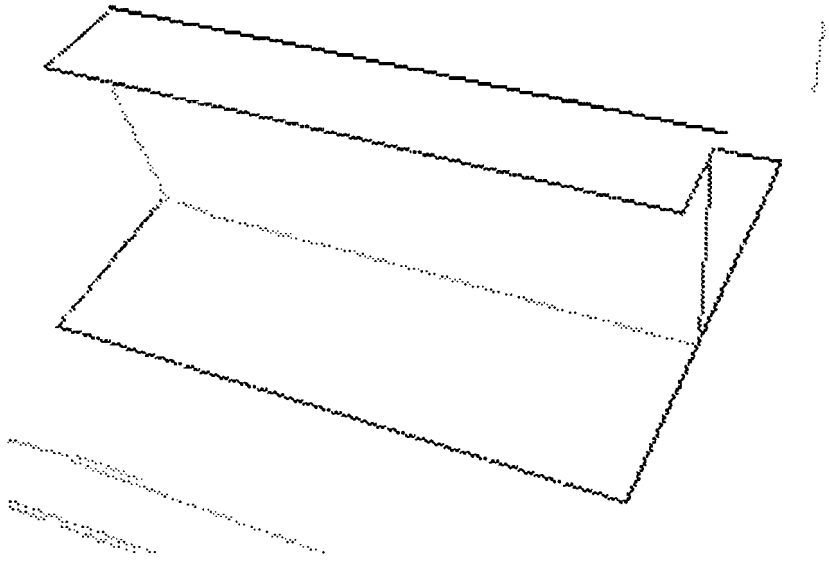


Fig.41. Line image with information about the gradient magnitude of the edges.

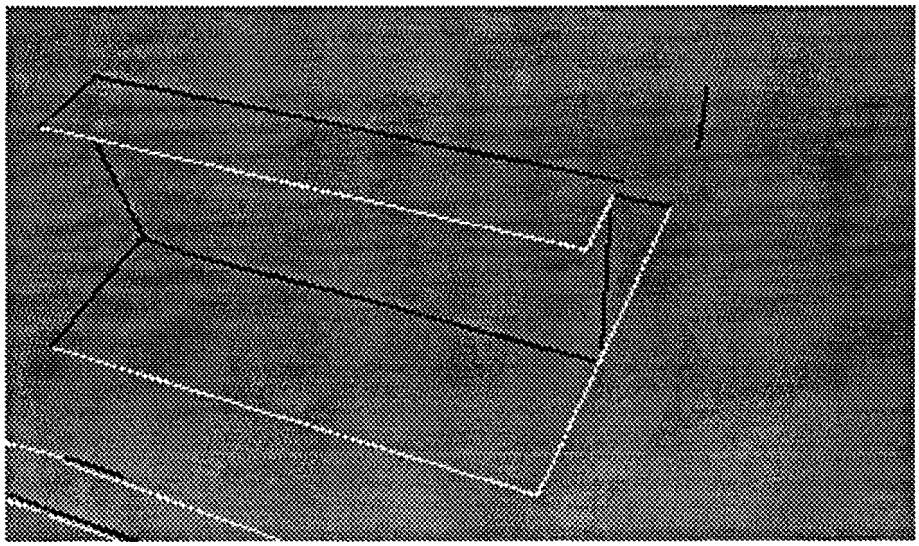


Fig.42. Line image with information about the gradient direction of the edges.

Appendix 3 Output formats.

The files in which the line segments are stored consist of respectively:

- A list of points, from which respectively the x-coordinate and the y-coordinate is given (in subpixel accuracy).
- A list of line segments; two point-numbers from the list of points (which are the endpoints of the line segment) are given for every line segment.

An example of such file in **LDR**-format is given below. This file contains 5 line segments.

LDR-format:

1 353.741241 5.406591
2 367.000000 13.844110
3 394.000000 232.840820
4 369.405212 192.150635
5 366.198059 194.280350
6 196.000000 314.531067
7 265.895569 204.852142
8 177.000000 267.643860
9 392.500000 235.054306
10 276.000000 324.867371
Faces:
1 2.
3 4.
5 6.
7 8.
9 10.

When line segments are extracted **with** line properties (see Appendix 1), then also a file in **LDP**-format will be created. This file is the same as the file in **LDR**-format, except that for every line segment, the value $\overline{p}$ according to (32) and a value proportional to *var* p according to (33), are respectively given (see Appendix 2).

The same file in **LDP**-format is as follows:

LDP-format:

1 353.741241 5.406591
2 367.000000 13.844110
3 394.000000 232.840820
4 369.405212 192.150635
5 366.198059 194.280350
6 196.000000 314.531067
7 265.895569 204.852142
8 177.000000 267.643860
9 392.500000 235.054306
10 276.000000 324.867371
Faces:
1 2 159 0
3 4 81 1
5 6 134 184
7 8 69 0
9 10 151 154