Technische Universität Dresden

Detection of crack-like indications in digital radiography by global optimisation of a probabilistic estimation function

Dipl.-Ing. Oleksandr Alekseychuk

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs
(Dr.-Ing.)

genehmigte Dissertation

Technische Universität Dresden

Detection of crack-like indications in digital radiography by global optimisation of a probabilistic estimation function

Dipl.-Ing. Oleksandr Alekseychuk

von der Fakultät Elektrotechnik und Informationstechnik
der Technischen Universität Dresden

zur Erlangung des akademischen Grades eines

Doktoringenieurs
(Dr.-Ing.)

genehmigte Dissertation

Vorsitzender:	Prof. Dr.-Ing. habil. Wilfried Sauer
Gutachter:	Prof. Dr.-Ing. habil. Klaus-Jürgen Wolter Prof. Dr. rer. nat. habil. Hans-Jürgen Ullrich ORR Dr. rer. nat. Uwe Zscherpel
Tag der Einreichung:	10. Oktober 2005
Tag der Verteidigung:	10. Mai 2006

Zusammenfassung

In dieser Arbeit wurde ein neuer Algorithmus zur Detektion rissartiger Anzeigen in der digitalen Radiographie entwickelt. Klassische *lokale* Detektionsmethoden versagen wegen des geringen Signal-Rausch-Verhältnisses (von ca. 1) der Rissanzeigen in den Radiographien. Die notwendige Resistenz gegen Rauschen wird durch die Benutzung von *globalen* Merkmalen dieser Anzeigen erzielt. Das ist aber mit einem undurchführbaren Rechenaufwand sowie Problemen bei der formalen Beschreibung der Rissform verbunden. Üblicherweise wird ein übermäßiger Rechenaufwand bei der Lösung vergleichbarer Probleme durch Anwendung von Heuristisken reduziert. Dazu benuzte Heuristiken werden mit der Versuchs-und-Irrtums-Methode ermittelt, sind stark problemangepasst und können die optimale Lösung nicht garantieren. Das Besondere dieser Arbeit ist anderer Lösungsansatz, der jegliche Heuristik bei der Suche nach Rissanzeigen vermeidet. Ein globales wahrscheinlichkeitstheoretisches Merkmal, hier Schätzfunktion genannt, wird konstruiert, dessen Maximum unter allen möglichen Formen, Längen und Positionen der Rissanzeige exakt (d.h. ohne Einsatz jeglicher Heuristik) gefunden werden kann. Diese Schätzfunktion wird als die Summe des *a posteriori Informationsgewinns* bezüglich des Vorhandenseins eines Risses im jeden Punkt entlang der hypothetischen Rissanzeige definiert. Der Informationsgewinn entsteht durch die Überprüfung der Hypothese der Rissanwesenheit anhand der vorhandenen Bildinformation. Eine so definierte Schätzfunktion ist theoretisch gerechtfertigt und besitzt die gewünschten Eigenschaften bei wechselnder Anzeigenintensität. Der Algorithmus wurde in der Programmiersprache C++ implementiert. Seine Detektionseigenschaften wurden sowohl mit simulierten als auch mit realen Bildern untersucht. Der Algorithmus liefert gute Ergenbise (hohe Detektionsrate bei einer vorgegebenen Fehlalarmrate), die jeweils vergleichbar mit den Ergebnissen trainierter menschlicher Auswerter sind.

Abstract

A new algorithm for detection of longitudinal crack-like indications in radiographic images is developed in this work. Conventional *local* detection techniques give unsatisfactory results for this task due to the low signal to noise ratio (SNR $\simeq$ 1) of crack-like indications in radiographic images. The usage of *global* features of crack-like indications provides the necessary noise resistance, but this is connected with prohibitive computational complexities of detection and difficulties in a formal description of the indication shape. Conventionally, the excessive computational complexity of the solution is reduced by usage of heuristics. The heuristics to be used, are selected on a trial and error basis, are problem dependent and do not guarantee the optimal solution. Not following this way is a distinctive feature of the algorithm developed here. Instead, a global characteristic of crack-like indication (the estimation function) is used, whose maximum in the space of all possible positions, lengths and shapes can be found exactly, i.e. without any heuristics. The proposed estimation function is defined as a sum of *a posteriori information gains* about hypothesis of indication presence in each point along the whole hypothetical indication. The gain in the information about hypothesis of indication presence results from the analysis of the underlying image in the local area. Such an estimation function is theoretically justified and exhibits a desirable behaviour on changing signals. The developed algorithm is implemented in the C++ programming language and testet on synthetic as well as on real images. It delivers good results (high correct detection rate by given false alarm rate) which are comparable to the performance of trained human inspectors.

Contents

List of used symbols

A^* – Nilson's generalisation of Dijkstra and Moore ordered graph search algorithm

$a(n_i, n_j)$ – arc which connects nodes n_i and n_j in a graph

$C(a, b)$ – parametric curve of two parameters a and b (Hough Transform)

CD – correct detection rate

$c(n_i, n_j)$ – cost attached to every arc $a(n_i, n_j)$ of a graph

$c(n)$ – cost estimation function, which is an estimate of the cost of the best path through a graph from a start node to an end node constrained to go through the node n

c_i – cost of the graph arc in Martelli's edge detection algorithm

$cur(x, y)$ or $cur(p)$ – curve curvature in the neighbourhood of point $p = (x, y)$

$D(x, y)$ – local area centred around point (x, y)

D_{max} – maximum optical density of radiographic film

dI_1 – *a posteriori* information gain about hypothesis of indication presence, $dI_1 = \log P_{1\ post} - \log P_{1\ prior} = \log \frac{P_{1\ post}}{P_{1\ prior}}$, where $P_{1\ post}$ and $P_{1\ prior}$ are *a posteriori* and *a priori* probabilities of hypothesis of indication presence

ε – mean-square error

E_0 – initial radiation intensity

E_x – intensity of transmitted radiation

$e(n)$ – heuristic estimation of the cost from node n of a graph to a goal node ($c(n) = f(n) + e(n)$ in A^*)

FA – false alarm rate (false detection rate)

$f(n)$ – cost of the lowest cost path from the start node to node n of a graph ($c(n) = f(n) + e(n)$ in A^*)

$f(\cdot)$ – impulse response of filter kernel (Canny filter); template function (profile fit); function under optimisation (dynamic programming)

$f^*(\cdot)$ – path estimation function designed in this work for purpose of detection of elongated crack indications

$f^*_{ch}(\cdot)$ – path estimation function for model of chained indications

FOM – figure of merit

$G(N, A)$ – directed graph having nodes $n_i \in N$ and arcs $a(n_i, n_j) \in A$

$Gr_d(x, y)$ – image gradient in point (x, y) in direction d

$g(x, y)$ – image grey value in point (x, y)

$g_{bkg}(x, y)$ or g_{bkg} – estimated background grey value

$g_{ind}(x,y)$ or g_{ind} – estimated indication *grey value* (not to confuse with indication intensity $z(x,y)$: $z(x,y) = g_{bkg}(x,y) - g_{ind}(x,y)$)

H_1 or $H(h)$ – hypothesis of indication presence (complements $H_0 = H(0)$)

H_0 or $H(0)$ – hypothesis of indication absence (complements $H_1 = H(h)$)

$h(x,y)$ or h – true indication intensity calculated according to the assumed image model in the idealised case of absence of noise or the intensity of the sought indication

K – array of real coefficients (mask, kernel) or binary mask

$L(\cdot)$ – Canny's localisation criterion

L^- – list containing coordinates of the binary mask elements which are zero for the mask centred around the current point, but set to unity for the previous mask position (the list contains relative coordinates of pixels in respect to the current coordinate of the mask centre)

L^+ – list containing coordinates of binary mask elements which are set to unity at the current mask position, but are zero at the previous one (the list contains relative coordinates of pixels in respect to the current coordinate of the mask centre)

L_{sign} – minimal significant length of indication

$L_z(h) = \rho(z|h)$ – *likelihood* of presence of the indication with intensity h given the estimation of indication intensity z, see $\rho(z|h)$

MTF – abbreviation of "Modulation Transfer Function"

μ – radiation linear attenuation coefficient in reciprocal units of thickness

$N(x; M_x, \sigma_x)$ – Gaussian (Normal) probability distribution of argument x with mean M_x and standard deviation σ_x

ν – random component introduced by noise

$O(\cdot)$ – "order of" - an approximate of its argument up to some factor or some finite number

$P(\cdot)$ – probability of event

$P(h)$ – *a priori* probability of occurrence of a crack indication with intensity h (discrete intensities model)

$P_{class}(x,y)$ – probability of membership of image pixel (x,y) to object class *class* (relaxation methods)

P_{trans} – probability of indication prolongation (in chained indications model)

$p = (x,y)$ – coordinate vector of a point (x,y)

ROC – abbreviation of "Receiver Operating Characteristic"

$r(\lambda, \lambda')$ – compatibility coefficient between a pixel with label $\lambda \in \Lambda$ and a pixel with label $\lambda' \in \Lambda$ (relaxation methods)

$\rho(\cdot)$ – density of a probability distribution

$\rho(h)$ – *a priori* probability density of occurrence of a crack indication with intensity h

$\rho(z)$ – *a priori* probability density of estimated indication intensity z

$\rho(h|z)$ – (or $\rho(h|Z = z)$) probability density of presence of the sought discontinuity with intensity h given the estimated indication intensity z

$\rho(z|h)$ – *a priori* probability density of estimated indication intensity z with the condition of a true indication presence with intensity h **or**, if the dependence from h given z is of primary interest, then – the *likelihood* of presence of indication h (now *a posteriori*) and denoted as $L_z(h) = \rho(z|h)$

S – array which stores optimisation information and has the same dimensions as the source image (implementation of the algorithm)

SNR – signal to noise ratio

σ – standard deviation of noise

σ_{bkg} – standard deviation of estimation of background grey value

σ_{ind} – standard deviation of estimation of indication grey value

σ_s – standard deviation of simulated noise

σ_z – standard deviation of estimation of indication intensity

th – grey value threshold (adaptive segmentation methods)

W – width of a spatial operator

w or w_{ind} – width of a crack-like indication

Z – one sample from population of estimated object intensities

$z(x, y)$ or z – estimation of indication intensity (not grey value!) in point (x, y): $z(x, y) = g_{bkg}(x, y) - g_{ind}(x, y)$

Introduction

Cracks are the most dangerous and at the same time the most difficult to detect material defects. Detection of cracks is the primary purpose of in-service inspection of welding seams of austenitic steel pipes in power plants. Radiography is a widely used non-destructive inspection method. During in-service inspection of pipelines big quantities of welding seams have to be examined routinely. As the visual analysis of radiographs is labour intensive and subjective there is a demand for automated crack detection. This work is devoted to the development of an algorithm for automated detection of longitudinal crack indications in radiographic images of welding seams acquired during in-service inspection of pipelines.

Crack indications are the recorded local intensity changes of penetrating radiation which are caused by the presence of cracks in the tested object. Due to the inherent properties of the radiographic method these crack indications are usually of low contrast and sharpness and located on a non-homogeneous and noisy background. A numerical simulation and comparative evaluation of simulated and real radiographs, performed in this work, showed for typical real radiographic images signal to noise ratios in the range from one to two. I.e. signals to be detected have an intensity just slightly higher than the root mean square (RMS) value of the surrounding background noise.

In such conditions conventional *local* detection techniques give unsatisfactory results. Since the crack indications appear as elongated objects which consist of many connected *local crack indications* situated along some curve, the usage of some *global* (integral) feature will allow a more reliable separation of crack-like indications from noise instances.

As the spatial shape, position and length of the crack indication are unknown, a search among all possible shapes, positions and lengths of the indication is necessary. The indications, which fulfil all detection criteria, could be found during this search. However, the two-dimensional nature of the crack indication allows a great number of possible indication shapes and positions and this number grows exponentially with indication length. The complete enumeration of all possible shapes, positions and lengths of the indication is not feasible. Therefore the use of optimised search techniques is obligatory. Thus the task of detection of two-dimensional longitudinal crack indications is converted into the task of searching for an optimal path in a weighted graph. The search can be done using graph theoretic methods or dynamic programming.

In order to accomplish the search, for example by methods of graph theory, it is essential to define an estimation function of each path in a graph of possible indication

positions, shapes and lengths. This path estimation function acts as an optimisation criterion and should reflect the probability, that the path coincides with a crack indication, i.e. should be adequate. In conventional approaches to the solution of this problem complex multi-parameter estimation functions are used to ensure adequacy of the estimation function. However, a complex estimation function causes high computational complexity of the search. Consequently, strong heuristics [41] must be applied in order to reduce the computational complexity [13, 20, 35, 36, 37, 59]. A drawback of heuristic search consists in the impossibility to guarantee robustness and optimality of the solution. Even more: the deviation from the real optimum remains unknown.

A distinctive feature of the algorithm developed in this work is the introduction of a path estimation function which enables the use of *exact* search techniques. The path estimation function is constructed here as a sum of *a posteriori* information gains about the hypothesis of indication presence in all points along the whole hypothetical indication. The maximum of such an estimation function can be found exactly using dynamic programming, because the estimation function is a sum of terms, each of which depends on only a few discrete arguments. The use of dynamic programming in place of the conventionally used graph theoretic methods is allowable because both approaches lead to equivalent solutions [38].

The algorithm developed here uses as local operator a modification of the directional local gradient operator proposed by Rosenfeld [56]. This modified operator assumes a certain image model in the local area. Its response is proportional to the intensity of a crack-like indication. Based on the same image model the likelihood function of presence of a crack-like indication is calculated.

Since the search procedure in the developed algorithm is exact, all heuristic assumptions apply to the path estimation function and the local operator. Whereas conventionally, heuristics concerning the search itself were necessary. Consequently, further improvements or adaptations for other applications can be concentrated on the design of the path estimation function and the local operator, since the optimisation procedure is already exact.

Besides the quality of detection and the speed of execution, the ability of a fully automated operation is another important characteristic of any detection algorithm. Unfortunately, the automation and optimality are controversial properties: in the so-called unsupervised algorithms either all parameters are hard coded for the specific application case or a non-optimal solution is used which gives mediocre results for the general case [19, 34]. To account for this, the following parameters are operator controlled in the developed algorithm (i.e. have to be selected prior to start):

1. the minimum signal to noise ratio of indications of interest (controls algorithm sensitivity);
2. the *a priori* probability of indication presence (influences detection of indications with gaps);
3. the minimal and maximal width of indication (define transversal resolution);
4. the minimal length of indication (defines longitudinal resolution).

By means of these parameters the objects which have to be detected are defined (within the limits of the model used). The choice of these parameters must be done by a human inspector on the basis of his experience and requirements of the particular

detection task.

The developed algorithm was tested on a set of one hundred real radiographs of welding seams for which the true defect locations are known, as well as on a set of synthetic images. The ability of the developed algorithm to perform detection with a quality comparable to human performance is demonstrated. The processing time on up-to-date sequential hardware (at the time of writing: 3 GHz Intel CPU) is still not applicable for interactive processing, but enough for batch execution.

Despite the emphasis on the specific practical application of crack detection, the developed algorithm can be applied for detection of other elongated objects or conventional edge detection in low signal to noise ratio conditions.

This work is structured in the following way. In the first chapter definitions of used terms are given, basics of conventional and digital radiography with application to non-destructive testing are described and the task of this work is formally defined. Subsequently, in the second chapter, an overview is given of conventional approaches to object detection found in the literature. Their limitations and applicability to the task of elongated object detection in low signal to noise ratio conditions is analysed. In the third chapter the developed crack detection algorithm is described in details. Experimental results are presented in the fourth chapter. The results obtained from simulated and real experimental data are compared with humans' performance. Receiver Operating Characteristic (ROC) [42] is used as the performance criterion. The final summary and outlook conclude the description of the new way developed for detection of crack indications on noisy backgrounds and show directions for further improvements.

Chapter 1

Basics of non-destructive radiographic testing, formulation of the task of crack detection

1.1 Basics of industrial radiography

Radiography is a widely used method in non-destructive testing (NDT). The basic principle of radiography is the attenuation of penetrating radiation in materials as a function of radiation energy, material density and material thickness. For a monochromatic source and the absence of scattering in the test object, the dependence of radiation attenuation from material thickness can be mathematically presented by the Lambert-Beer's law:

$$E_x = E_0 \, e^{-\mu x}, \tag{1.1}$$

where E_0 represents initial radiation intensity, E_x – the intensity of transmitted radiation, x – the thickness of absorbing material and μ – the linear attenuation coefficient in reciprocal units of thickness.

Due to the different attenuation coefficients of various materials and dependence of attenuation on the material thickness, discontinuities in the test specimen cause variations in the transmitted radiation intensity. The transmitted radiation beam is then recorded by some means which forms a radiographic image.

Traditional sources of penetrating radiation are X-rays generators (tubes) and radioactive isotopes (industrial sources of γ-rays are $Ir{-}192$, $Co{-}60$ or $Se{-}75$). Each source has its own characteristics and therefore has certain advantages in specific applications. The selection of the particular energy range of radiation, to be used, is one of the major factors responsible for penetrating ability and contrast.

The traditional radiation detector is radiographic film. The principle of image formation is the interaction of the radiation with the silver halide crystals in radio-

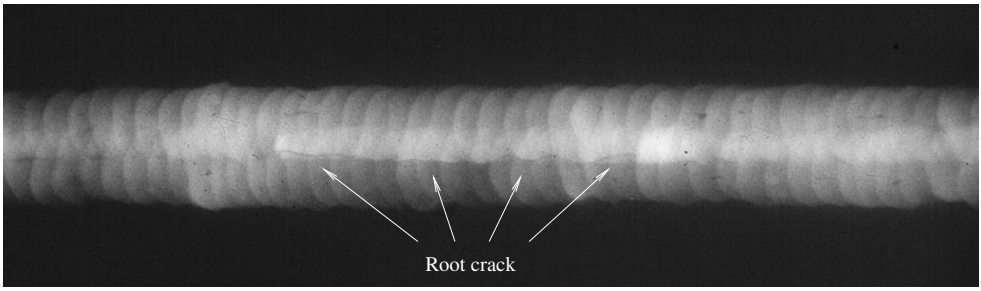


Figure 1.1: A radiograph of a welding seam with longitudinal root crack.

graphic film forming a latent image. This image in not readily visible, but becomes so after processing of the film. The greater the radiation intensity, the greater the number of interactions and the darker the processed film will become. Fluorescent screens are alternatively used to convert radiation into visible light. They can be used as standalone detectors in *radioscopy systems* as well as in combination with radiographic film since the visible light will expose the film.

Electronic imaging methods in traditional radioscopy are represented by systems in which the fluorescent viewing screen is observed with a television system (fluoroscope). These systems can be improved by special X-Ray image intensifiers. In an image intensifier the radiation first hits input screen, which emits electrons. These electrons are accelerated in electrical field and interact with a fluorescent screen at the output window.

A radiation image detector is characterised by several parameters:

- *Sensitivity* is defined by exposure necessary for formation of a standard image.
- *Contrast* is defined as ratio between the change of intensity of recorded signal and the change of intensity of incident radiation ($\frac{(signalchange)}{(radiationchange)}$) and characterises ability of detector to record small changes of incident radiation.
- *Dynamic range* is ratio between highest and smallest undistorted signals.
- *Spatial resolution* of detector is defined by Modulation Transfer Function (*MTF*) or Point Spread Function (*PSF*) and characterises ability of the detector to record small spatial details.
- *Noise level*, *inhomogeneity* in general or *granularity* for radiographic film characterise amount of uncontrolled variation of image intensity.
- Detector size, usage convenience, price and lifetime are another important detector characteristics and there are even more ones.

An example of a radiographic image is given on *Fig. 1.1* showing a radiograph of a welding seam with longitudinal crack. An extensive description of principles of radiography and an overview of techniques and equipment used in traditional radiographic NDT can be found in [5].

Concerning defect detection by means of radiography it is necessary to mention the following characteristic shortcomings of the method:

- Flaw indication on a radiographic image depends strongly on the flaw shape and position in 3D (*Fig. 1.2*). Two similar cracks will be differently indicated on

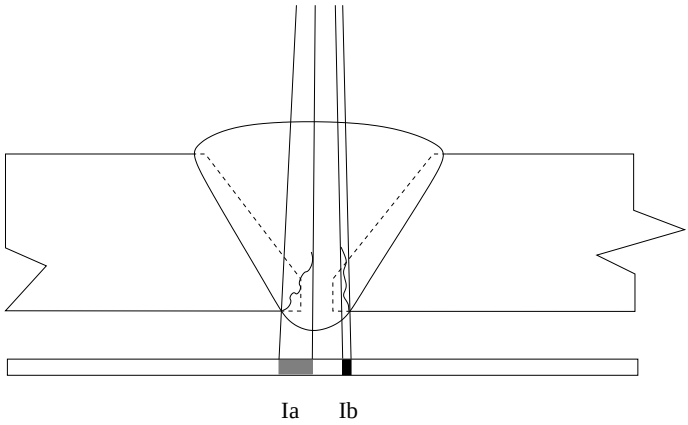


Figure 1.2: Schematic cracks in a welding seam and their indications.

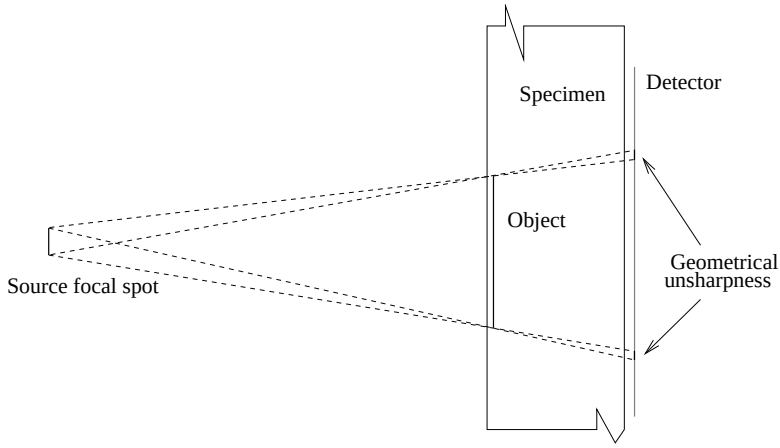


Figure 1.3: Geometrical unsharpness as a result of the central projection and a non-point radiation source.

the image due to the different spatial orientations. Indication Ia (in *Fig. 1.2*) will have lower contrast and worse detectability due to its angular position. The worst case is a flat 3D object located perpendicular to the direction of the radiation beam. It also causes another problem: a crack, deflecting in 3D from its optimal orientation, can partially disappear from the radiograph due to unfavourable positioning that leads to interpretation errors.

- Central projection in combination with the extended size of the focal spot of a radiation source (non-point source) cause inevitable geometrical unsharpness (see *Fig. 1.3*). This unsharpness can be reduced by minimising the size of the focal spot, by positioning of radiation source as far as possible and detector as close as possible to the object. The unsharpness of the resulting image is the convolution of geometrical unsharpness and the inner unsharpness of the detector.

- Inhomogeneous material, quantum noise and detector noise cause stochastic fluctuations of the image intensity which conceal low-intensity indications. Particularly in many practical applications the noise level is comparable or even bigger than the useful signal intensity of flaw indications.

1.2 Digital radiography

Digital radiography is an advanced technique which involves computerised methods of investigation. In digital radiography the image may be directly acquired in digital form or be converted into by means of digitising of an analogue medium. Since the image is available as a computer file it can be archived, copied or transmitted to different places without any loss of image information, digitally processed to enhance required features or to eliminate interfering ones. The list of available processing procedures is large and includes: functional transformations of intensity (brightness-contrast adjustment, histogram transformations), filtering of different kinds (noise reduction, sharpening), background linearisation and elimination, and finally image segmentation, object detection and interpretation.

There are several techniques of digital image acquisition:

- Scanning of the traditional radiographic film is an obvious way to achieve digital images using conventional radiography systems. Because of the high maximum optical density ($D_{max} > 5$) of NDT films in comparison to films used in visible light photography and medical radiography, special scanners are designed for this purpose. At present, this approach is unsurpassed in spatial resolution and signal-to-noise ratio, but requires film processing and is therefore time consuming and labour intensive.

- Phosphor imaging plate technology is a replacement for conventional film which eliminates necessity of dark room processing. They employ a coating of photostimulable storage phosphor on a flexible plate to capture images. When exposed to X-rays, radiation sensitive centres inside the phosphor crystals are excited and electrons are trapped in a semi-stable higher-energy state. A reading device scans the plate by means of a laser beam. The laser energy releases

the trapped electrons, causing visible light to be emitted. This light is registered by a photomultiplier and converted into a digital bit stream which encodes the digital image. After scanning the imaging plate can be erased with surplus light and reused. The applicable dynamic range of imaging plates is even larger than NDT films, but resolution and signal-to-noise ratio are inferior.

- Fluorescent and scintillation screens coupled with photo diode matrices provide means for instant detection (indirect flat panel detectors). Because of optical scattering within the media, some spatial blurring and increased noise can be encountered which degrades image quality as compared to film. However, these systems offer superior performance relative to conventional radioscopy systems (image intensifiers or fluoroscopes), while exhibiting faster read-out times as compared to digitised film and imaging plates.

- Most progressive (at present) are direct registration detectors (direct flat panel detectors). The detector consists of an amorphous selenium (α-Se) or cadmium telluride ($CdTe$) photoconductive layer coating a thin film transistor (TFT) array. X-rays are converted directly into charge carriers. An electrical bias field is applied to separate the charge carriers and to collect them (no photosensitive elements as in the indirect approach). For such systems the resolution is only limited by element size of the TFT matrix (100 μm at time of writing).

A plethora of literature is currently available on digital radiography, eg. [12, 63].

1.3 Task of automated crack detection

In practical application of conventional radiography, as far as in computer aided systems, the final stage of inspection process, i.e. the radiograph analysis, defect detection and evaluation, is performed by a human inspector. As statistic shows, crack indications are considerably difficult to evaluate for human inspectors. It is also a labour intensive process. Unfortunately conventional image processing techniques fail in this case too or their application for the analysis of radiographs is very limited [54, 48, 27]. That is due to the characteristic features of radiographic images (unsharpness, low signal-to-noise ratio, significant inhomogeneity) as well as the absence of formal definition of objects to be detected. As a consequence there is a demand to develop an algorithm giving satisfactory results. Its successful application could enhance overall quality of radiographic inspection and broaden its application range.

The aim of this work is the development of algorithm for automatic detection of crack indications, e.g. caused by IGSCC (intergranular stress corrosion cracking) in welding seams of austenitic steel pipes of nuclear power plants [17]. Only the detection of crack-like indications arose during in-service time is considered here. Volumetric defects from production process are out of scope of this work.

Most obviously crack-like indications on radiographs can be defined as objects characterised by fast local changes of intensity in one direction (local contrast) and longitudinal shape of the discontinuity in the orthogonal direction. Under ideal conditions a crack appears on a film radiograph as thin curved line of higher optical density than the surroundings. But a real crack-like indication can be unsharp, differently curved, loose and gain intensity along its length, overlap with other, non

necessary crack-like, indications. So additionally to the cross-section contrast the shape of the cross-section profile and the 2D shape of the detected (suspected) indication can be analysed. Unfortunately human inspectors are not able to describe their experience on this topic in formal terms. It is because they are trained to recognise defects using representative sets of images showing an existing known flaws (e.g. by reference catalogues of radiographs). Recognition criteria are formed in the mind of the trainee by themself (except very simple starting premises) on the basis of physical knowledge about defect phenomena, common sense and human intuition. Additionally, information about welding technique, material properties, etc., not available from the radiograph but from testing protocols, are included. So the first part of the task of automated defect detection is definition of sufficient criteria allowing successful crack recognition.

It is also evident that putative detection algorithms cannot be restricted to local environments, particularly in case of low signal to noise ratios. A global approach taking into account all indication points is expected to provide a more robust resolution of this problem, because in this case the local abnormalities caused by noise are less decisive for the final result. This leads to another problem which must be solved: the overcoming of excessive computational complexity of global detection algorithms.

The problems described above are typical for crack detection on radiographs, but are not completely unique to this field. They are also present in slightly different form in other applications of digital image processing. So, as a consequence, conventional detection techniques, found in the literature, will be reviewed and analysed in the next chapter.

Chapter 2

Overview of conventional approaches to object detection

The task of object detection can be expressed in terms of image segmentation into regions of two possible categories: object (or set of objects) and background. In the context of this work only longitudinal crack indications are considered as objects of interest.

Since the task of image segmentation is often difficult to solve and some times even to define, numerous different techniques have been developed. The literature is vast. Unfortunately without actually implementing an approach and testing it on a representative image population, it is impossible to determine if it is appropriate for the intended application or if it is not. Although the number of image segmentation techniques available is large, the number of techniques with proved effectiveness is few and some general classes can be recognised.

For the task of crack detection a classification into local and global methods might be useful. Local methods take into account only the close vicinity of a current point in order to make segmentation decision. In the simplest case all image points are tested in sequence. Alternatively the running points might be selected depending on the segmentation result in the previous (already examined) points. The overall segmentation result is then a combination of the outputs of all local results. In a solely local segmentation algorithm the overall result is just a superposition of local results. In contrast to local methods, the global methods use information about the entire image to make a segmentation decision. Methods, which use quite large areas, comparable to the whole image size, can be considered as global too [16].

If objects to be detected have size much bigger than a single pixel in each dimension, another classification is possible: border-based or region-based methods. Border-based methods detect objects by means of detection of their closed boundary (border). Region-based methods do not concentrate on boundary analysis, but on analysis of the whole 2D region supposedly occupied by the object. Because much more pixels are used at a time by region-based methods than by the border-based

ones, the noise resistance of the formers is much better. But due to the fact that crack indications are considered thin in one dimension by definition, the application of region-based methods is of limited use.

Due to the low thickness of crack indication, the task of crack detection has similarities with the task of border detection. In both cases abrupt changes in intensity have to be detected, but in the case of crack detection these changes are two-sided. As a consequence of this similarity a number of border detection techniques can be adopted for the purpose of crack-like indication detection.

In what follows a brief survey of some known techniques of detection of elongated discontinuities and the evaluation of their applicability for the task of detection of crack-like indications is given.

2.1 Local methods of detection of grey value discontinuities

2.1.1 Gradient filters, boxcar, Laplacian of Gaussian and Canny filters

A common method to detect discontinuity is differentiation [47]. Abrupt changes of image intensity are transformed to sharp peaks on differentiated image (also commonly called *gradient image*). Slow changes of image intensity cause low amplitude response. Succeeding threshold operation marks points with derivatives which exceed a given limit (threshold).

A general approach to perform differentiation is to convolve the image with a corresponding mask. For this purpose a differentiation mask is passed through the image and the following sum is computed for each image point:

$$Gr(x,y) = \frac{1}{N} \sum_{i=0}^{X_k-1} \sum_{j=0}^{Y_k-1} K[i,j] g(x+i-X_k/2, y+j-Y_k/2), \tag{2.1}$$

where $g(\cdot,\cdot)$ denotes pixels of the source image, $Gr(\cdot,\cdot)$ pixels of the resulting gradient image, K the convolution mask of dimension $X_k \times Y_k$ and $N = \sum_{i=0}^{X_k-1} \sum_{j=0}^{Y_k-1} K[i,j]$. When the mask is centred around a boundary pixel, the response is computed by using the corresponding partial neighbourhood. Due to the uniform nature of the performed operation the algorithm is also called a *filter* or an *operator*.

The selection of the mask (mask coefficients $K[i,j]$) determines the operation to be performed (see *Tab.2.1*). Usually partial edge gradients are computed in two orthogonal directions (Roberts, Prewitt, Sobel and Frei-Chen operators [49]). Another approach is to compute gradients in a large number of directions by convolving with a set of different masks (Prewitt Compass, Kirch, Robinson 3- and 5-level, Nevatia-Babu gradients [49]). Since the magnitude of the gradient is usually of interest rather than its direction, the partial gradients can be combined in square root form:

$$Gr(x,y) = \sqrt{\sum_{d=1}^{D} Gr_d(x,y)^2} \tag{2.2}$$

Table 2.1: Convolution masks (kernels) of some popular differentiation operators.

Operator	Row gradient	Column gradient
Pixel difference	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Separated pixel diff.	$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$
Roberts	$\begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$
Prewitt	$\frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
Sobel	$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Frei-Chen	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix}$	$\frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$

Nevatia-Babu operator reduced to only four partial gradients (0, 30, 60 and 90 degrees) :

$$\frac{1}{1000}\begin{bmatrix} 100 & 100 & 0 & -100 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 100 & 0 & -100 & -100 \end{bmatrix} \quad \frac{1}{1102}\begin{bmatrix} 100 & -32 & -100 & -100 & -100 \\ 100 & 78 & -92 & -100 & -100 \\ 100 & 100 & 0 & -100 & -100 \\ 100 & 100 & 92 & -78 & -100 \\ 100 & 100 & 100 & 32 & -100 \end{bmatrix}$$

$$\frac{1}{1102}\begin{bmatrix} -100 & -100 & -100 & -100 & -100 \\ 32 & 78 & -100 & -100 & -100 \\ 100 & 92 & 0 & -92 & -100 \\ 100 & 100 & 100 & 78 & -32 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix} \quad \frac{1}{1000}\begin{bmatrix} -100 & -100 & -100 & -100 & -100 \\ -100 & -100 & -100 & -100 & -100 \\ 0 & 0 & 0 & 0 & 0 \\ 100 & 100 & 100 & 100 & 100 \\ 100 & 100 & 100 & 100 & 100 \end{bmatrix}$$

Figure 2.1: Part of typical radiographic image of welding seam. Lead markers (numbers in white squares) show the circumferential weld position in cm. A crack indication is located at position 32-39 under the seam.

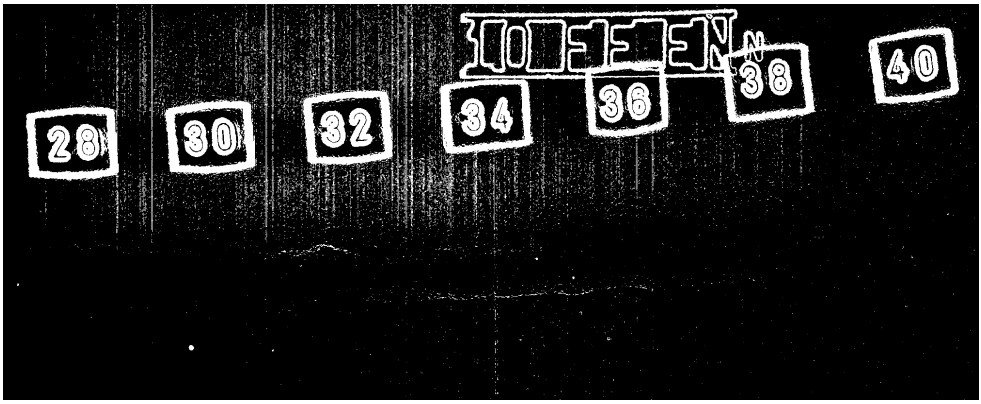


Figure 2.2: Result of application of Sobel operator to the image from *Fig. 2.1* and subsequent thresholding operation with manually selected threshold.

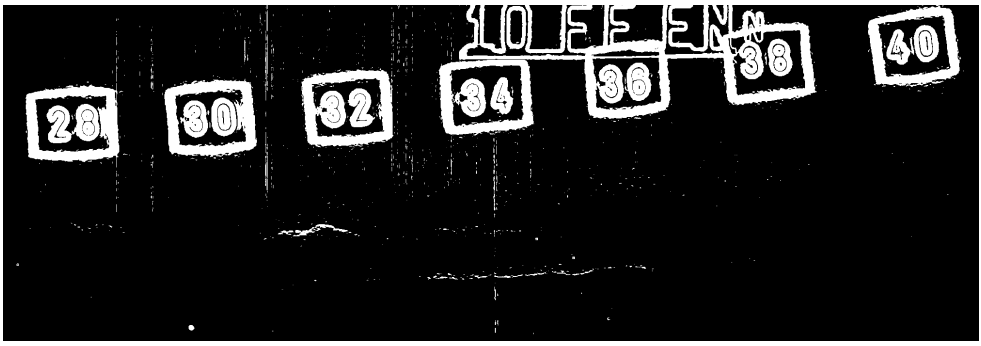


Figure 2.3: Result of application of Nevatia-Babu operator to the image from *Fig. 2.1* and subsequent thresholding operation with manually selected threshold.

or in magnitude form (for the sake of computational simplicity):

$$Gr(x,y) = \sum_{d=1}^{D} |Gr_d(x,y)|, \tag{2.3}$$

where D is number of partial gradients. When a sufficiently large number of partial gradients is computed (> 8), then the maximum of magnitude of the partial gradients is usually taken:

$$Gr(x,y) = \max_{d=1,D}(|Gr_d(x,y)|) \tag{2.4}$$

and the edge direction can be determined by the direction of the largest gradient.

Simple pixel differences such as Roberts, Prewitt, Sobel and Frei-Chen operators are very fast in execution and easy to implement but error-prone in high noise environments (see *Fig. 2.1* and *Fig. 2.2*). This is a result of the small number of image pixels involved in computation (for every mask position). Noise resistance can be achieved by properly extending the size of the area over which the gradients are computed. Operators of this type are called *boxcar operators*. Examples are the operators as proposed by Rosenfeld and Thurson [56] and Nevatia and Babu [49] (see *Tab.2.1* and *Fig. 2.3*).

A truncated pyramid operator was suggested by Abdou [49] which gives a linearly decreasing weighting to pixels distant from the centre of an edge. Evolving this approach Argyle and Macleod [49] have proposed the use of Gaussian-shaped functions for weighting of neighbourhood area as a mean of noise suppression. These operators, unlike the boxcar operator, give decreasing importance to pixels more distant from the centre of the mask.

Extended size gradient operators can be considered to be composite operators in which a smoothing operation is performed on a noisy image followed by a differentiation operation [49]. Well-known examples of compound gradient operators are the derivative of Gaussian (DoG) and Laplacian of Gaussian (LoG), in which Gaussian-shaped smoothing is followed by the differentiation or Laplacian respectively.

All of the differential edge detection operators previously described here have been heuristically derived. Canny [10] has taken an analytical approach to the design of such operators. Canny's development is based on a one-dimensional model of a step edge (arbitrarily shaped) plus additive white noise with standard deviation σ. It is assumed that edge detection is performed by convolving a one-dimensional noisy edge signal $g(x)$ with an anti-symmetric impulse response function $f(x)$, which is of zero amplitude outside the range $[-W, W]$. An edge is marked at the local maximum of the convolved gradient. The impulse response $f(x)$ is derived from maximising of the following three criteria:

1. Good detection. There should be a low probability of failing to mark the real edge point, and a low probability of falsely marking a non-edge point. Since both these probabilities are monotonically decreasing functions of the output signal to noise ratio, this criterion corresponds to maximising of the following functional:

$$SNR(f) = \frac{|\int_{-W}^{+W} g(x)f(x)\,dx|}{\sigma\sqrt{\int_{-W}^{+W} f^2(x)\,dx}}. \tag{2.5}$$

2. Good localisation. Edge points marked by the operator should be as close to the centre of the edge as possible. The localisation criterion is defined by Canny as

$$L(f) = \frac{|\int_{-W}^{+W} g'(x) f'(x)\, dx|}{\sigma \sqrt{\int_{-W}^{+W} f'^2(x)\, dx}}, \tag{2.6}$$

where $g'(x)$ is first derivative of the edge signal and $f'(x)$ is first derivative of the impulse response function of the filter.

3. Single response. There should be only a single response to a true edge. Canny has demonstrated, that if only the two first criteria are used, then the optimal detector for step edges is a truncated step, or the difference of boxes operator. However it has a very high bandwidth and tends to exhibit many maxima in its response to noisy step edges. These extra edges should be considered erroneous according to the first criterion. The average distance between adjacent maxima in the noise response of f, denoted as x_{max}, constraints choice of f according to the single response criterion:

$$x_{max}(f) = kW, \tag{2.7}$$

where k is a given fraction coefficient and W is the operator width.

Canny has combined these three criteria by maximising the product $SNR(f)L(f)$ subject to the constraint x_{max}. The optimisation of the impulse response function $f(x)$ (values of elements in the convolution kernel) is performed with respect to this criterion and the model of the discontinuity to be detected.

Because of the complexity of the formulation, no analytic solution was found, but a variational approach has been developed and optimal operators for some popular edge models were computed. For low values of x_{max} Canny operator resembles a boxcar. In the other limit case (when x_{max} is large) it can be closely approximated by a derivative of Gaussian (DoG) impulse response function.

2.1.2 Profile fit

Ideal discontinuities may be viewed as one-dimensional or two-dimensional discontinuities of the form shown on *Fig. 2.4*. In this case, actual image data can be matched against, or fitted to, the ideal model. If the fit is sufficiently accurate at a given image location, a discontinuity is assumed to exist with the same parameters as the ideal edge model.

In the one-dimensional edge fitting case described by Pratt [49] and illustrated on *Fig. 2.4* the image signal $g(x)$ is fitted to a step function

$$f(x) = \begin{cases} a, & \text{if} \quad x < x_0 \\ a + h, & \text{if} \quad x \geq x_0 \end{cases} \tag{2.8}$$

An edge is assumed to be present if the mean-square error

$$\varepsilon = \int_{x_0 - W}^{x_0 + W} [g(x) - f(x)]^2\, dx \tag{2.9}$$

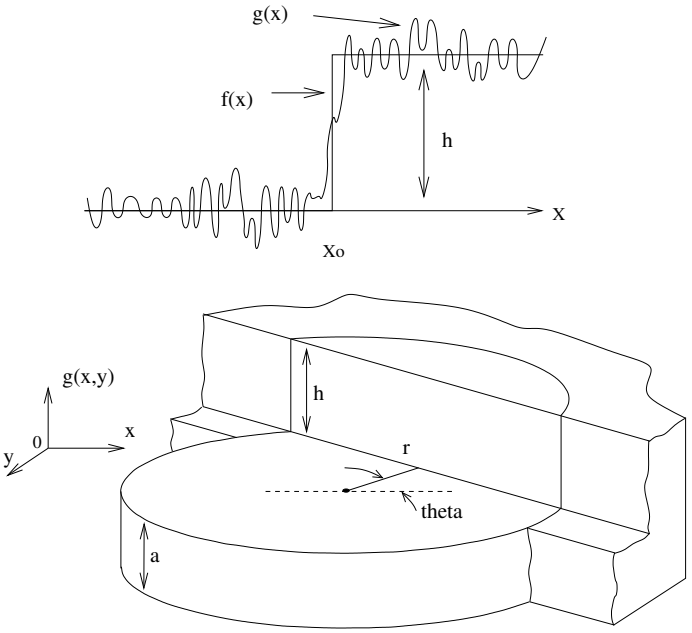


Figure 2.4: One dimensional and two dimensional edge fitting with a function of zero order.

is below some threshold value (assuming the ideal profile is defined within the $[-W, W]$ range). In the two-dimensional formulation the ideal step edge is defined as

$$f(x,y) = \begin{cases} a, & \text{if} \quad (x\cos\theta + y\sin\theta) < r \\ a+h, & \text{if} \quad (x\cos\theta + y\sin\theta) \geq r \end{cases} \tag{2.10}$$

where θ and r specify the polar distance from the centre of a circular test region to the normal point of the edge. The edge fitting error is

$$\varepsilon = \iint_{overcircle} [g(x,y) - f(x,y)]^2 \, dx \, dy. \tag{2.11}$$

The more general approach is fitting to a $n \times n$ neighbourhood a surface of degree $m < n^2$. The best fit is formed by minimising the error between the surface of degree m and the actual image [47].

Hueckel [25] has developed a procedure which is based on the idea to describe the image and the ideal edge by a set of two-dimensional basis functions by a Fourier series in polar coordinates and find the best match based on such a representation.

2.1.3 Analysis of local methods

Profile fitting methods require substantially more computation than derivative-based edge detection methods, but are capable to produce superior results. While computational complexity of the profile fitting methods is still within the acceptable limits, the main problem of their application for the crack detection is the undefined shape

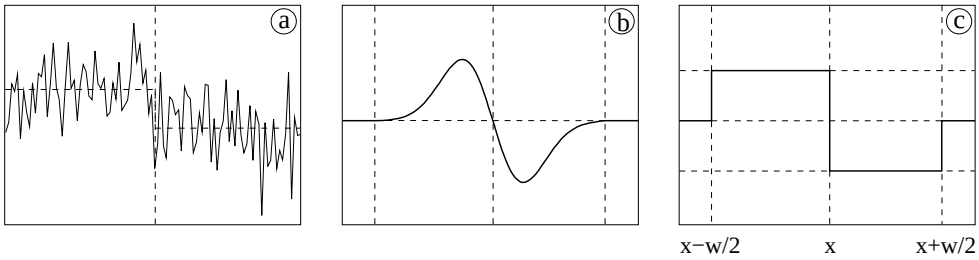


Figure 2.5: a) Noisy edge considered by Canny; b) Canny filter resembles closely first derivative of Gaussian if single response criterion is of primary importance; c) The boxcar is the optimal choice if maximum sensitivity is required.

of the profile to fit the image to - this is due to the nature of the images under consideration: a crack can be arbitrary shaped in 3D and projected to the image plane under arbitrary angles.

The same problem with the varying shape of discontinuity profile decrease efficiency of the Canny filter too. This is due to the impulse response function $f(x)$ of Canny filter must be optimised to the distinct model of discontinuity profile, which is not known.

Additionally, it is important that the good detection criterion and the single response criterion in Canny filter are contradictory. Giving different weights to these criteria results in different optimal convolution kernels and consequently in different properties of the operator [10]. If the single response criterion is of primary importance, then the optimal kernel closely resembles first derivative of a Gaussian as it is shown on *Fig. 2.5*. At the same time this leads to a decreased sensitivity of the operator. If the single response criterion is not used, the operator resembles a boxcar and the sensitivity reaches its maximum.

The use of a boxcar for increased operator sensitivity (instead of a smoothed kernel) causes many maxima in its response to a noisy step edge, since boxcar has a very high bandwidth. This is usually a problem when detection is performed solely on the basis of the output of the local operator. But if the boxcar is followed by a second stage of the detection algorithm which makes decisions on a more global basis, then multiple responses of boxcar could not be a big problem.

That is why a decision to use boxcar as a local operator is made in this work. Besides the increased sensitivity, such simple operator has a big performance benefit – the computationally expensive convolution can be substituted by a relatively fast normal averaging and even faster recursive averaging (see later in section 3.3).

2.2 Segmentation techniques based on global statistics

A number of global segmentation techniques are based on the analysis of the image histogram. They include (but are not limited to): p-tile method [14], Mode method [50], Ostu method [44], histogram concavity analysis method [57], Pun en-

tropic methods [51, 52], Kapur, Sahoo, and Wong entropic method [29], Johansen and Bille entropic method [28], Moment-Preserving method [60], Minimum error method [31]. An overview and comparison of all these methods is given in [58]. In the context of this work all these methods share one common drawback – they depend solely on pixels' grey levels and do not take into account spatial relations between pixels inside or outside a class, so an *a priori* information about spatial features of the object (both local and global) remains unused. This makes all these methods not acceptable for the task of crack detection investigated here.

2.3 Grouping techniques

The initial local segmentation step is conventionally followed by grouping algorithms. The aim of this is to aggregate the local information into more global-like structures and to discard spurious local segmentation results caused by the noise. These methods include relaxation algorithms, adaptive thresholding and contour following, heuristic graph search and dynamic programming goal function optimisation, template matching (curve fitting), Hough- and Generalised Hough Transforms.

2.3.1 Relaxation methods

The idea of relaxation was introduced by Southwell [58] to improve the convergence of recursive solution for system of linear equations. In image segmentation, relaxation is applied as follows. First, a probability of membership of each image pixel to each object class is calculated on the base of pixel grey level. This stage is called *initial classification*. Then, in several iterations, the probability of each pixel is adjusted according to the probabilities of the neighbouring pixels. This provides some noise resistance and the possibility to use image context. Another attractive feature of this approach is the parallel processing technique.

Initial classification

Initial classification can be done with the application of any global segmentation method which makes global segmentation on the basis of the grey-level histogram, but for this specific purpose some special methods were developed. For the case of binary segmentation Rosenfeld and Smith [55] suggested the following method: Let d and l be the globally darkest and lightest grey levels, and $g(x, y)$ be the grey level of a pixel (x, y). Then let

$$P^0_{dark}(x, y) = \frac{l - g(x, y)}{l - d}, \tag{2.12}$$

and

$$P^0_{light}(x, y) = \frac{g(x, y) - d}{l - d}. \tag{2.13}$$

This method fails in cases where the object and background grey levels do not lie on the different halves of the grey level histogram. To avoid this problem, the same

authors propose another initialisation scheme. Let m be the mean grey level. Then, if $g(x,y) > m$, let

$$P^0_{light}(x,y) = \frac{1}{2} + \frac{1}{2}\frac{g(x,y) - m}{l - m}, \tag{2.14}$$

and if $g(x,y) \leq m$, let

$$P^0_{dark}(x,y) = \frac{1}{2} + \frac{1}{2}\frac{m - g(x,y)}{m - d}. \tag{2.15}$$

Fekete, Eklundth and Rosenfeld [18] suggest an approach in which they assume the histogram can be divided into two Gaussian subpopulations so that the grey level distribution can be written as the sum of two Gaussian distributions. The parameters of these Gaussian distributions are determined by a method suggested in [11]. There a faster convergence of the relaxation process was obtained by this method.

Iterative updating of probabilities

As previously mentioned, the updating process consists of adjusting the probabilities of each pixel based on neighbouring probabilities. Let Λ be the set of class labels (the classes of dark and light pixels in case of binary segmentation) and $r(\lambda, \lambda')$ a compatibility coefficient between a pixel with label $\lambda \in \Lambda$ and a pixel with label $\lambda' \in \Lambda$. Then $r(\lambda, \lambda')$ is defined as following:

$$r(\lambda,\lambda') = \begin{cases} -1, & \text{if } \lambda \text{ and } \lambda' \text{ are } \textit{incompatible} \\ 0, & \text{if } \lambda \text{ and } \lambda' \text{ are } \textit{independent} \\ 1, & \text{if } \lambda \text{ and } \lambda' \text{ are } \textit{compatible} \end{cases} \tag{2.16}$$

Zucker, Hummel and Rosenfeld [64] propose the following equation for updating probabilities:

$$\begin{aligned} P^{k+1}_{\lambda}(x,y) &= \frac{P^k_{\lambda}(x,y)[1 + q^k_{\lambda}(x,y)]}{\sum_{\lambda' \in \Lambda} P^k_{\lambda'}(x,y)[1 + q^k_{\lambda'}(x,y)]}\,, \\ q^k_{\lambda}(x,y) &= \frac{1}{8} \sum_{(x',y') \in N(x,y)} \sum_{\lambda' \in \Lambda} r(\lambda,\lambda') P^k_{\lambda'}(x',y')\,, \end{aligned} \tag{2.17}$$

where $N(x,y)$ is the 8-neighbourhood of (x,y) and k is the iteration number.

The attractive feature of the relaxation methods for the crack-like indication detection is the usage of image context, i.e. probabilities of pixels' membership are adjusted depending on the neighbourhood. However, it is difficult to implement complex shape restrictions in the scheme described above even by expanding size of the neighbourhood $N(x,y)$ or changing neighbourhood shape from square to something else. This limits the application of the approach and it is no longer investigated here.

2.3.2 Adaptive segmentation methods

The idea of adaptation is similar to the idea of relaxation in that way that the image context is used, i.e. parameters of the image processing algorithm are adjusted

depending on the image contents. Segmentation with a floating threshold is the most widely used application of this idea. Floating threshold segmentation consists of the adjusting of the threshold value on the base of the estimation of the signal intensity of the already segmented part of the object: $th_i = f(th_{i-1}, z_{i-1})$, where th_i and th_{i-1} are threshold values for steps i and $i-1$, and z_{i-1} is the estimation of the intensity of the object segmented on the previous steps. This idea is used in the numerous curve tracing algorithms. A classic example can be found in [16].

Not only the threshold value can be adjusted. For example, Canny [10] has proposed adaptation of his operator to the image contents in the following way: instead of the global noise estimation, the noise is estimated in some neighbourhood of a moving point. This results in a context dependent sensitivity of the segmentation operator.

Hwang and Haddad [26] proposed to adjust the *operator size* adaptively for each image region. This is to optimally solve the dilemma shown by Canny [10] that the detection process is always a compromise between noise resistance and localisation accuracy: a bigger operator size increase the noise resistance, but reduce the accuracy of localisation at the same time.

Palenichka, Alexeychuk and Zscherpel [2, 45] have used structural adaptation for detection of cracks in radiographic images. A crack indication is considered as a connected set of elementary indication parts – primitives. Detection is performed in several steps. First, primitives are detected which with high probability belong to the object. To exclude false indications on this step, the threshold is set to a relatively high value. In a second step, hypotheses about continuations of the detected crack parts are generated and tested. The threshold value as far as the position and the size of the search area are adjusted according to the available information about the crack part which is already found and according to the constraints of the assumed object model. *Fig. 2.6* illustrates this process. The radius of the search sector is a function of the length of the indication part already found: the longer part of crack is found the bigger area will be searched for a continuation. The opening angle of the search sector depends on the smoothness of the indication part already found: for smooth cracks the continuation is searched in a narrow area, if the crack direction fluctuates significantly – the search area is wider. The threshold is selected by maximising *a posteriori* likelihood of correct detection (see [2] for details). Such an approach works very well at moderate noise levels (signal to noise ratios) and is much faster than global optimisation techniques described in the next sections. Nevertheless, the application of this algorithm is impossible for SNR ¡ 3 – so fully global techniques come into considerations.

2.3.3 Hough Transform

The Hough Transform (HT) [24, 15] is an efficient technique for the detection of simple parametric curves, such as straight lines, circles, arcs, ellipses, etc. The HT considers global relationship between pixels.

Consider the Cartesian coordinate space XY and a parametric curve C of two parameters a and b: $C(a, b, x, y) = 0$, where $x \in X$ and $y \in Y$ are independent variables and a and b are parameters which determine the curve. Image points which belong to the certain curve $C(a', b', x, y) = 0$ with constant but yet unknown

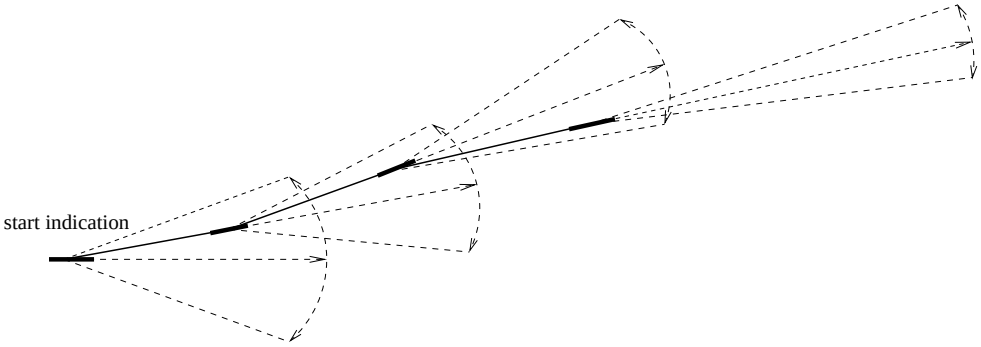


Figure 2.6: A schematic example of structure-adaptive crack tracing. Dashed lines denote borders and centre lines of continuation search areas. Thick solid line segments denote elementary start indication and "best candidate" elementary indications inside this search areas. Thin solid line corresponds to the indication detected in this way.

parameters a' and b' form a set of curve points $\{(x', y')\}$. This curve will correspond to a single point (a', b') in the parameter space AB ($a \in A$ and $b \in B$). To find this point one has to consider a and b as independent variables and plot in the parameter space AB the dependence between a and b for every point $(x_i, y_i) \in \{(x', y')\}$ (one plot per each (x_i, y_i)). All of these plots in AB will intersect exactly in the point (a', b'). To find parameters of the two-parameter curve (as in this example) it is not necessary to make a plot for every point (x_i, y_i) of the curve, but only for two of them, for a three-parameter curve – for three points, and so on.

The procedure described above allows the determination of parameters, given the curve on the image. But in practice it is usually necessary to perform both tasks: to detect the curve and to estimate parameters. For this purpose the parameter space is subdivided into so-called *accumulator cells*: each covers some fixed part of the parameters' range and is associated with mean value of this range. All accumulator cells form an array and initially are set to zero. Then for every image point (x, y) we let the parameter a equal to each of the allowed subdivision values on the A axis and solve for the corresponding b. The resulting b is then rounded to the nearest allowed value on the b axis. If a choice of a_i results in the solution b_i then we increment the corresponding accumulator cell (a_i, b_i) by the value of the current image pixel $g(x, y)$. At the end of this procedure, the searching for the local maxima or minima in the accumulator cells array with following thresholding operation allow detection and estimation of parameters of curves which correspond to the used parametric representation. The accuracy and computational efficiency is determined by the number of subdivisions of the AB space.

To extend the classic HT for curves of arbitrary shape (not only parametric) Ballard [3] has proposed the Generalised Hough Transform (GHT). The GHT uses a non-analytic representation of model shape. The representation corresponds to a table (R-table) which stores edge direction and the position of edge points with respect to an arbitrarily chosen reference point (i.e. vectorial displacement). In the extraction process, an estimate of the *location* of a shape is started by selecting the

entry in the R-table which has the same edge direction as the computed one for the edge point. Then the reference point which corresponds to this shape location is determined and the accumulator cell is incremented by the values of image points defined by the vectorial displacement stored in R-table.

GHT is an efficient method for detecting 2D object shapes if rotation and scale parameters are known *a priori*. However, when the orientation and the scale of an object are not available in advance, GHT *exhaustively* enumerates all possible rotation and scale parameters of the given shape. This requires two additional dimensions in the parameter space.

HT is an original and very interesting idea, but its application to the detection of crack-like object is combined with some problems. The first disadvantage, common to all HT-based approaches, is their large computational requirements depending on the number of parameters in shape representation. Second, in usual implementations of HT/GHT, each dimension of the parameter space is quantised into a finite number of intervals – then the computational complexity and storage requirements of the method depend on the quantisation size of each parameter and increase very fast if the application task requires some more resolution in the parameter estimation. Third, as it was studied by Grimson and Huttenlocher [23], a GHT specific drawback is its relative low noise resistance. But the main disadvantage of HT-like approaches for the task of crack-like objects detection lies in the absence of suitable parametric or tabular representation of crack indications. The parametric representation is principally possible, but the number of parameters vary and is very large – this prohibits the application of HT-methods. In case of a tabular representation for GHT it is necessary to explicitly provide a R-table for all possible indication shapes and this may be impossible too.

2.3.4 Optimal detection of curves using dynamic programming technique

Montanari [38] has proposed a method for optimal detection of curves with no explicit description. He makes use of a so-called *figure of merit* (FOM), which acts as an optimisation criterion and incorporates heuristic contents of the problem to specify the properties which a curve must have and their relative weights. Constraints of various kinds can also be embedded in the FOM.

Because the number of possible curves is usually large and grows exponentially with the length of the curve it is impossible to explicitly build all possible curves and calculate the FOM for them. However, using dynamic programming, the optimum can be found for a large class of FOM's by performing optimisation separately for each variable on which FOM depends and then "restore" the curve which gives the maximal FOM [38]. Dynamic programming is based on the principle of optimality: any optimal subpath of the graph *can* be part of the global optimal path, while any suboptimal path *cannot*. Optimisation using dynamic programming can be shortly described as follows.

The figure of merit in its general form is a function of N arguments: $FOM = f(x_1, x_2, ..., x_N)$, where x_i is discrete and $0 \leq x_i \leq n_i$, $i = 1, ..., N$. The task is to find the maximum value M of the FOM and values of $(x_1, x_2, ..., x_N)$ for which this maximum is achieved. If the FOM is a sum of terms, each of which depends on

only a few variables, then a multistage optimisation procedure can be applied. For instance, if

$$FOM(x_1, x_2, ..., x_N) = f_1(x_1, x_2) + f_2(x_2, x_3) + ... + f_{N-1}(x_{N-1}, x_N), \tag{2.18}$$

then the following recursion formula can be used:

$$\begin{aligned} f_1^*(x_1) &= 0, \\ f_{k+1}^*(x_{k+1}) &= \max_{0 \le x_k \le n_k} (f_k(x_k, x_{k+1}) + f_k^*(x_k)), \\ m_{k+1}(x_{k+1}) &= arg \max_{0 \le x_k \le n_k} (f_k(x_k, x_{k+1}) + f_k^*(x_k)), \end{aligned} \tag{2.19}$$

where $k = 1, ..., (N-1)$ and n_k is the maximum value of k-th argument. The intermediate values $f_{k+1}^*(x_{k+1})$ and $m_{k+1}(x_{k+1})$ must be saved in a table with n_{k+1} entries. Thus for all k, at the end of this phase, N tables have been stored. The formula

$$M = \max_{0 \le x_N \le n_N} f_N^*(x_N) \tag{2.20}$$

is used to find maximum of the whole merit function. The values of $(x_1, x_2, ..., x_N)$ for which the maximum is achieved are determined by scanning the stored tables with the following recursion formula:

$$\begin{aligned} x_N &= m_N = arg \max_{x_N} f_N^*(x_N), \\ x_k &= m_{k+1}(x_{k+1}), \ k = (N-1), ..., 1. \end{aligned} \tag{2.21}$$

This process is called *multistage optimisation*.

In general, the multistage optimisation method can be described as a step-by-step elimination of all the variables. For example, let

$$FOM(x_1, x_2, x_3, x_4, x_5) = f_1(x_1, x_2) + f_2(x_2, x_3) + f_3(x_3, x_4) + f_4(x_1, x_4, x_5). \tag{2.22}$$

Now we eliminate x_2 from the merit function. We get:

$$\begin{aligned} f_5(x_1, x_3) &= \max_{x_2}(f_1(x_1, x_2) + f_2(x_2, x_3)), \\ FOM(x_1, x_3, x_4, x_5) &= f_5(x_1, x_3) + f_3(x_3, x_4) + f_4(x_1, x_4, x_5). \end{aligned} \tag{2.23}$$

After the elimination of the variable x_2, an optimisation problem of the same type is obtained. The remaining variables can be eliminated in the same way.

The computational cost of elimination of a variable x_i depends substantially on the number of variables with which x_i is nonlinearly related. (For the example above there are two variables (f_5 is a function of two arguments).) Therefore, the order in which the variables are eliminated is decisive in determining the amount of computing time and storage required. Thus, a new optimisation problem arises, which is called the *secondary optimisation problem*.

As a practical example of the proposed approach Montanari has suggested a FOM for the optimal detection of low-curvature curves. It is defined as sum of grey levels

along the curve minus the sum of the curvatures at every point. If $p_i = (x_i, y_i)$ are the coordinate vectors of the points of the curve, the figure of merit is:

$$\begin{aligned} FOM(p_i, ..., p_N) &= \sum_{i=1}^{N} g(p_i) - \sum_{i=2}^{N-1} cur(p_i), \\ cur(p_i) &= (d(p_{i+1}, p_i) - d(p_i, p_{i-1})) \, mod \, 8, \end{aligned} \tag{2.24}$$

with the constraints:

$$\begin{aligned} max(|x_{i+1} - x_i|, |y_{i+1} - y_i|) &= 1, \\ cur(p_i) &\leq 1, \end{aligned} \tag{2.25}$$

where $g(p_i)$ is the image grey value in the point p_i, $cur(p_i)$ estimates curvature of the curve in the neighbourhood of the point p_i and $d(p_{i+1}, p_i)$ is the direction from point p_{i+1} to point p_i in terms of 8-membered neighbourhood (an octal number [16]). A detailed analysis of the application of the proposed FOM with the multistage optimisation technique is given in the same work of Montanari [38]. Recent application of the dynamic programming for the task of optimal border/curve detection, comparison to the graph searching technique and many references can be found in [8, 61].

As pointed out by Martelli [35] the optimisation by dynamic programming is "blind", that is, storage and time required do not depend on the curve contrast and the amount of noise added to the picture. A more reasonable procedure would finish faster on quality images and require more time on noisy ones. Although it is true in general, this drawback is not important in the task of crack detection in digital radiographs as the noise level on radiographs is usually very high (SNR $\simeq 1$).

The described approach is usually very effective if starting and ending points of the path are known and a suitable FOM is available. Unfortunately, the solution of these questions is not trivial: no systematic approach is available to build a good FOM for the general case – the solution must be found for each practical application separately and the usage of a naive intuitive FOM usually leads to unfeasible computational complexity of optimisation. Therefore and since:

1. the success of optimisation depends largely on application of a suitable FOM and
2. no application of the dynamic programming for the task of crack detection is found in the literature,

it will be tried in this work to create the required FOM and apply dynamic programming for the solution of the task of detection of crack-like indications in digital radiographs.

2.3.5 Optimal detection of curves using heuristic search of optimal path in a weighted graph

Another approach to curve detection which does not require explicit description of the sought object is based on a representation of the set of all possible curves by a directed graph [35]. The graph representation is very intuitive and offers a convenient way to ensure global optimality of the detected curve.

The structure of the graph is determined by properties, which the indication must have. If a point is supposed to belong to a crack indication, then all possible prolongations of the indication can be represented by a directed graph. Nodes of this graph correspond to image points through which the prolongations go. Arcs connect each node with the preceding and the subsequent nodes. Each path in the graph corresponds to a possible indication in the picture and has an associated *weight* or *cost*. A function which estimates the cost of the path incorporates available criteria of the sought object. A path has low cost if it is likely to be the true indication. If the cost estimation function is known, then the lowest cost path can be found by employing the well-known graph searching techniques [41]. Then the task of optimal curve detection is transformed to the task of finding an optimal path in the graph.

To build such graph practically, a certain spatial object model has to be assumed and at least the following tasks have to be solved:

1. Selection of the graph starting point.
2. Definition of the indication (path) prolongation rule.
3. Definition of the path termination criteria.

If the tasks listed above are accomplished and a characteristic is found which estimate "goodness" of the given path in the graph, then the detection task can be converted to the task of finding the optimal (in the predefined sense) path through the graph.

1. *Where to start the graph:* It is possible to exhaustively explore each and every point of the image by building graphs with roots in these points. Then $X \times Y$ graphs have to be explored (where X and Y are image dimensions). Alternatively, graphs can be started only from points which meets certain requirements. For example, pixels which exhibit high indication intensity z. This reduces computational complexity, but requires definition of such selecting criteria and could compromise the quality of the detection (more on this later in this chapter).

2. *How a given indication can be prolongated:* The connectivity relationships between adjacent pixels in an image array and the chosen 2D model of the indication shape define the graph. Prolongation of the path which correspond to a node is called *opening of the node*. The number of possible prolongations of the path is called *branching coefficient*. The branching coefficient characterises how fast the search tree grows and therefore influences the computational complexity of the search.

3. *When a path has to be terminated:* It is necessary to define the termination of a path. If this is not done, then the search may become in a general case computationally inefficient (as unnecessary nodes will be opened). For example, if a graph of known fixed depth has to be explored, then the path terminates whenever the path length reaches this predefined value. Such length limitation may appear artificially for the application considered here, therefore the approach proposed later makes path termination on a basis of estimation of its goodness.

4. *How to estimate path goodness:* It is a question about synthesis and calculation of the path estimation function. Using the right estimation function is essential,

but often overseen. Beside adequacy to the application problem, it is important to have the estimation function in such a form which allows effective usage of the optimised search techniques. This problem is specially addressed in this work in Chapter 3.

In general a conventional graph searching algorithm can be represented as follows:

1. Let $G(N, A)$ be a directed graph having nodes $n_i \in N$ and arcs $a(n_i, n_j) \in A$. If an arc is directed from a node n_i to a node n_j, then the node n_j is called *successor* of the node n_i, and the node n_i is called *parent* of the node n_j. A cost $c(n_i, n_j)$ is attached to every arc $a(n_i, n_j)$. A sequence of nodes n_1, n_2, ..., n_k with each node n_j is a successor of n_{j-1} is called *path* from node n_1 to node n_k. The value
$$C = \sum_{j=2}^{k} c(n_{j-1}, n_j) \tag{2.26}$$
is its cost. In a general case graph can have several starting nodes and several goal nodes. Goal nodes can be defined explicitly or implicitly, i.e. any node can become goal if some conditions hold. Start nodes must be defined explicitly.

2. Select a start node n_1 and mark it as active. Selection of the start node is made by an external algorithm.

3. Calculate successors of the active node. This process is called *opening* of the node. Pointers are set up from each successor back to its parent node.

4. Check the successor nodes to see if they are goal nodes. If a goal node has been found, the pointers are tracked back to the start node to produce a solution path and algorithm terminates. Otherwise (if a goal node has not been found), the opening process continues.

5. In a general case a parent node has more than one successor. Therefore, in each iteration there are more than one node been yet not opened. Select one of the not-yet-opened nodes and mark it as active. Go to step 3. The node which will be marked as active (and the order in which nodes will be opened) is determined by the used search algorithm and can differ from algorithm to algorithm.

Although all possible prolongations of an indication form a directed graph, *all possible paths in this graph form a tree.* So far the path goodness is a general function of all points through which the path goes, the search must be performed on the tree rather the graph. An example of such a tree is shown on *Fig. 2.7.* Nodes of this tree do not correspond to the image points any longer. Instead, the nodes represent different paths through the tree from the start point to the current one.

The number of possible paths grows exponentially with the length of the paths. A 1 cm long indication corresponds on an image with 50 μm pixel size to a curve of the length of 200 pixels. This means that search tree depth is 200 stages. For the branching coefficient 3 the number of all possible paths is in the order of $3^{200} \simeq 2.7E95$. This is only for the 1 cm long indication. Thus it becomes clear that exhaustive search of crack indication is practically impossible, but employing of heuristic information about the sought object can accelerate the search.

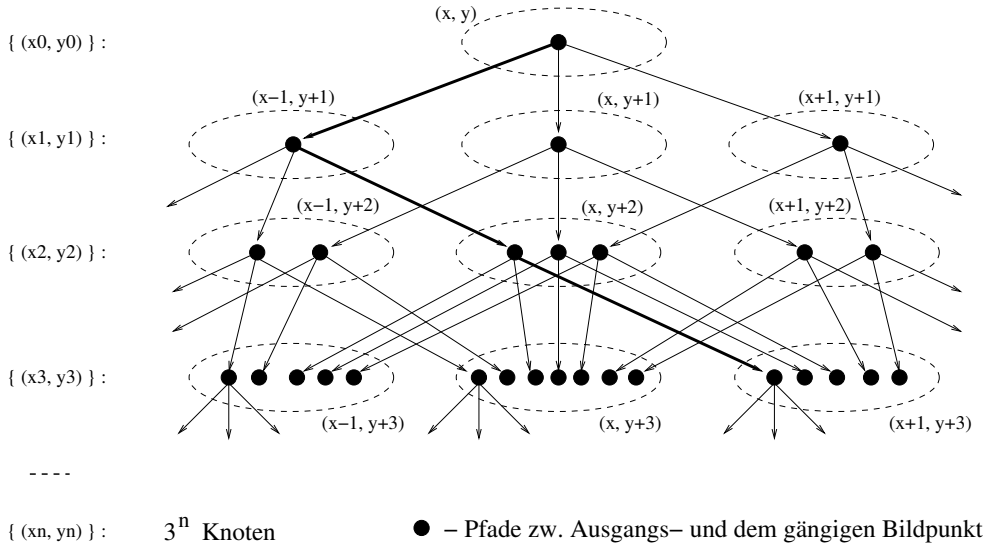


Figure 2.7: Indication prolongations as a search tree. Ovals represent points in the image array. Small black circles denote tree nodes which correspond to the different paths through the image array from the start point to the current one.

Application of A^* algorithm

The problem of finding the optimal path through the graph was widely explored in the game theory [41]. Due to the inapplicability of the exhaustive enumeration approach in this field, several optimised search strategies have been proposed. The optimised search provides a possibility to find an optimal path by building and exploring only a part of the search tree, thus reducing the computational complexity of algorithm. The goal nodes can be defined explicitly (goal image points enumerated) as well as implicitly. For example, if the optimal path with fixed length N has to be found, then goal nodes are all nodes which are N steps apart from the beginning node.

The idea of optimised search (some times also called *"ordered search"*) is as follows: on each step of the algorithm the node with the best value of the estimation function is opened (i.e. a "best first search"). Thus paths laying near the optimal path grow faster than paths laying far from the optimal one. Due to this, only a small fraction of the whole tree is build before the goal is reached. The meaning of the phrase "best value of estimation function" depends on the goal of optimisation: either minimisation or maximisation of the estimation function.

Opening of the node with the most promising perspective on each step requires additional computational expenses on the searching of the node with the best value of estimation function. But usually these expenses are a fair trade off for the benefit of not opening the less perspective nodes. Thus the goal node can be reached faster.

The classic example of the optimised search is finding the minimal cost path through a graph, where cost of the path is the sum of costs of each step [41, 35, 36, 20]. The step costs are implicitly assumed to be positive, thus the sum grows monotonically. As the optimised search is also applicable for more general estimation

functions, it is worth to stress here explicitly: *the ordered search ensures optimality of the solution only if the path goodness estimation is a monotonous function of the path length and changes in the opposite direction than the goal of optimisation.* I.e. if the estimation function monotonically grows, then the minimum can be found using ordered search. And in opposite: if the estimation function decreases, then only the maximum can be found by this method.

The most known and widely used ordered search algorithms are Dijkstra and Moore algorithms generalised by Nilson in its A^* algorithm [41]. The A^* algorithm makes use of *lists* for storage of tree nodes. Each node has a pointer which points to the parent of the node (for start node this pointer is set to null). The node n selected for opening on each step is chosen according to the cost estimation function $c(n)$, which is an estimate of the cost of the best path from the start node s to an end node e constrained to go through n. The A^* algorithm needs a cost estimation function $c(n)$ to be constructed in the following way:

$$c(n) = f(n) + e(n), \tag{2.27}$$

where n is a node of the graph, $c(n)$ estimates cost of a minimal cost path from the starting to a goal node going through the node n, $f(n)$ is the cost of the lowest cost path from the start node to n found so far, $e(n)$ is a heuristic estimation of the cost from n to a goal node. In general, it is not guaranteed, that the A^* algorithm finds the minimum-cost path – its advantage is speed due to the use of heuristics. However, if the search task consists in the minimisation of $c(n)$ and $e(n)$ is a lower bound on the cost of the minimal-cost path from the node n to a goal node, the procedure yields the optimal path to the goal. $e(n)$ can be set to zero if no better estimate is available: $e \equiv 0$. Then the procedure reduces to the uniform-cost algorithm of Dijkstra [41].

The A^* algorithm can be represented as follows:

1. Put the start node s in the list called *Open* and calculate the path estimation function $c(s)$.
2. If *Open* list is empty, then the algorithm ends without finding any solution. Otherwise, continue.
3. Select from *Open* a node n which has the lowest value of $c(n)$ and transfer it to the list called *Closed*. If there are several nodes which has the same c, then n is selected randomly among them, but always preferring goal nodes.
4. If n is a goal node, then the algorithm ends successfully and the optimal path can found by recursively tracing back pointers to the parent nodes. Otherwise, continue.
5. Open node n, by building all possible successors. If there are no successors, then go directly to Step2. Otherwise, for each of the newly created nodes: calculate c, direct pointer to its parent node n and add the newly created node to *Open* list.
6. Go to Step2.

In the A^* algorithm the selection of a node with the best $c(n)$ from *Open* list can be done in two ways. Either the *Open* list is kept unsorted and the search of the best node is performed each time a node has to be opened or *Open* list is maintained

sorted and the best node is simply the first or the last node in the list. To maintain a sorted *Open* list the newly created nodes have to be inserted to the list with respect to the sort criterion. Both approaches have their advantages and disadvantages. For example, inserting a node to a sorted list with length N without violating the sort order requires at maximum $log_2(N)$ comparisons and is less computationally expensive than searching for the best node in an unsorted list which always needs N comparisons. But if the branching coefficient if big, then much more new nodes have to be inserted to the *Open* list than nodes which are taken from it. As far as the optimal path can be found without opening of all nodes from the *Open* list, the maintaining of the sorted list can be more computationally expensive than searching for the best node in the unsorted list each time when the best node has to be selected.

When the optimum path is clearly distinguished from any of the alternatives, A^* is very efficient and can have computational requirements proportional to the number of nodes in the optimal path. However for real images (and especially for the noisy images considered in this work) the number of alternatives which must be examined grows very quickly and the A^* algorithm has to open more nodes than it is acceptable in practice. I.e. the application of A^* algorithm for solution of such large scale problems as crack detection on hight resolution images is unfeasible as seen from the computational standpoint.

Application of heuristic search

Additional heuristics can be applied to reduce the search space. This is possible only at the expense of refusal of the guarantee that the optimal path will be found. The most frequently used heuristics are *staged search*, *limitation of successors* and tweaking of cost estimations $e(n)$ and $f(n)$ [41].

Staged search If a staged search is performed, the tree (so far generated by the search) is truncated according to some rule. Truncation can be performed in regular time intervals, when the available memory is exhausted or when some other conditions have fulfilled. Truncation usually consists in removing of the non-perspective (in the predefined sense) nodes from the *Open* list.

On early stages of this research the A^* algorithm was implemented in combination with the staged search approach. One difficulty was noticed which is not mentioned in the literature. Truncating the *Open* list results only in a reduction of number of nodes *to be opened*, while the *Closed* list continues to grow quite fast. An additional effort is necessary to remove from *Closed* list those nodes which belong to the discarded paths. For this purpose each node has to maintain a list of its successors (children). Then, if some node is removed from *Open* list, it has to be synchronously removed from the list of successors of its parent. The parent node (which is already in *Closed* list) can be removed only if its list of successors is empty. The same procedure must be performed for the parents of all nodes which are about to be removed. Such procedure is necessary to ensure that any of the removed nodes does not lie on the optimal path. It costs the increased average complexity of opening a node and the increased space necessary to store the node. Overall the efficiency of the staged search becomes much lower as it could be expected from the first description of this approach.

Limitation of successors Another widely used heuristic is the limitation of number of successors of each node. In other words, it is the reduction of branching coefficient. Usually the limitation of number of successors to a very low value is required to be effective. One example is the exploration of only two most perspective of many possible prolongations of the path. This works some times for low noise levels and when an absolute optimality of the search result is not required. But in the case explored in this work, i.e. when noise level is high, such approach can obviously lead to the missing of the optimal path and therefore can not be used.

Heuristics on $e(n)$ As mentioned above, to ensure that the optimal solution will be found by the A^* algorithm, $e(n)$ (the estimate of the path from the current node to a goal node) has to be a lower bound on the true best path from the current node to a goal node [41]. To guarantee the optimal solution, $e(n)$ can be set to zero if no better estimation is available. Choosing $e(n)$ as close as possible to the allowed lower bound leads to the search which found yet optimal solution while opening the minimal number of nodes.

The usage of $e(n)$ which is not the lower bound of the estimate of true best path may lead to a suboptimal solution, but often allows to speed up the search significantly. This speed up is a result of the more directed search caused by the more significant contribution of the prognosis function $e(n)$ into overall estimation $c(n)$ and therefore increased importance of the closeness to a goal node in comparison to the optimality of the already found path. The amount of this contribution is also called heuristic power of $e(n)$. The heuristic power of $e(n)$ (and therefore the straightening of the search tree) and can be adjusted simply by multiplying $e(n)$ with a coefficient > 1.

Heuristics on $f(n)$ There are several heuristics which allow to influence the estimation of goodness of the solution path. This is done by respective modifications of the estimation function $f(n)$:

- Adding a positive constant to the cost of each step (adding a new node to the path) tends to smooth and straighten the path (for tasks of finding the minimal cost path). This is because as the cost of each step increases, the length of the path becomes relatively more important in comparison to the local image properties which are estimated by the local operator [20].
- Raising the cost of the node to a power greater than unity introduces an inhibition against going through a point having a low probability of belonging to a crack indication [20, 46].
- Taking the logarithm (and inverting the sign of the result) or raising the cost to a power lower than unity reduces, on opposite, the sensitivity to points with low probability of belonging to a crack indication [46].
- Weighting the different components of the estimation function with different coefficients enhances or suppresses the contribution of the respective path features.

Application of any of these (or several at a time) heuristics may lead to an improvement of separation of the optimal path from the set of other (suboptimal) paths. This straightens the search tree and thus reduces the search complexity. But it may

lead to a missing of the optimal path by following the wrong one – finally these are heuristics only.

History of application of the heuristic search in a weighted graph for the detection of curves To the author's knowledge, Martelli [35] has pioneered the idea to represent the task of optimal curve detection by finding the optimal path in a graph. The A^* algorithm was used.

Several years later Martelli has examined relations between heuristic graph search and dynamic programming approaches [36]. He stated that the multistage optimisation process can be considered, in terms of graph search, as a breadth-first method, i.e. a method, which will open first all nodes on the first level, then on second level, and so on. While using dynamic programming an algorithm has to open all nodes of the search graph. A^* algorithm may open only a small subset of nodes, if heuristic information is available from the application field. This can lead to substantial reduction of computing time. Furthermore, in many applications finding the optimal solution is not a matter of concern, a "good" solution will satisfy. In this case, more powerful heuristics can be used, thus obtaining a more efficient search of the graph. This allows (according to Martelli) to increase the complexity of the $c(n)$ without substantially affecting the complexity of the search process, thus allowing the terms $c(n)$ to take into account more complex properties of edges such as their curvature.

While search algorithms are generally problem independent, good investigated and theoretically well established [41], the construction of a cost estimation function depends on the particular application and is generally very heuristic. For the purpose of edge detection Martelli has proposed [35, 36] to consider boundaries between image pixels as graph's arcs (edge elements) and the arc cost is defined as:

$$c_i = M - D_i + c'_i, \tag{2.28}$$

where D_i is the magnitude of the difference between the grey levels of two adjacent points which build the boundary, M is the maximum possible D_i, c'_i a term which express the edge smoothness. The cost of the full path is defined as sum of costs of all path's elements. The smoothness is taken into account by assigning a cost to the curvature of the contour: 8 consecutive edge elements are considered, the tail of the first element is connected to the head of the fourth element with a straight line and the tail of the fifth element to the head of the eighth element with another straight line. The angle α between this two lines is measured and the curvature is considered to be proportional to this angle

$$c'_i = k\alpha. \tag{2.29}$$

Using the cost estimation proposed by Martelli, nodes on low levels of the search graph will have usually a lower cost than nodes on high levels, because nodes on low levels correspond to shorter edges. If the graph is searched using the A^* algorithm , all nodes whose cost is smaller than the cost of the minimal solution will be opened and these nodes will be mainly on the first levels of the graph. To speed up the search Martelli proposes two heuristic approaches [36] (but they do not longer guarantee the optimal solution):

- if the opened node n_1 of cost c_1 corresponds to an edge of length l_1 and there is a node n_2 not yet opened of cost $c_2 > c_1$ corresponding to an edge of length

$l_2 < l_1$, and if $l_2 \leq (l_1 - t)$, where t is a given threshold, then n_2 is discarded (will be never opened).

- for each node to be opened a mean cost of the optimal path to this node is computed – if the mean cost is higher than a given threshold, the node is not opened.

The second heuristic also acts as a sensitivity control of the the algorithm: if there is no contour, the relative cost of the nodes will be higher than the threshold and eventually the list of the nodes to be opened will be empty - so the algorithm will stop without reaching any goal node.

Fischler, Tenenbaum and Wolf [20] propose further improvements to the scheme described above:

- Different types of operators for the detection of local contour elements can be used simultaneously: operators with good classification ability provide a framework which gets filled in by information supplied by operators which contribute precise local feature characterisation.

- Using *a priori* knowledge: adding a constant bias b to each cost value $c(n_i, n_j)$ in (2.26) tends to smooth and straighten the path. This is because the length of the path gains importance as compared to the value returned by a local operator with increasing bias. Similarly, raising each cost to a power a introduces a very strong inhibition against going through a point which has a low likelihood of being a part of the contour.

Sankar and Sklansky [59] have proposed to use a priori information about the contour to build an approximated *plan of the boundary*. The heuristic search algorithm shall follow the plan when very little edge information is present, but tends to ignore this plan when a great amount of edge information is available. A corresponding heuristic cost estimation function was proposed for detection of lung nodules in medical radiographs.

Mérő [37] has proposed to organise the search in the following way. First, points which are centres of possible path bifurcations are detected by a simple criterion. Second, the algorithm creates a sort of potential field around the bifurcation points. The midpoints of the desired paths are found where two potential values originated from different bifurcations are adjacent and their sum attains a local minimum. The paths can be determined by tracing back the growth of the potential field from the midpoints.

Dobie and Lewis [13] propose a special graph representation for the specific problem of searching minimum cost paths in images. In a general formulation of search for the minimum cost path, the graph's nodes are arranged in ordered or non-ordered lists and links between parent and children nodes are installed explicitly. Updating of cost requires a search of the corresponding node in the lists or each node has to store a list of references to all its parents and children. If each graph node corresponds to a point in the image and the image itself is represented as a grid of points, it is proposed to represent the graph as the grid too. The links between nodes are not stored explicitly but are implicit in the locations of nodes within the grid. A queue of coordinates of nodes to be opened is maintained in parallel. Besides path information each node keeps flag which indicates if the node is currently in the queue.

Summary on heuristic search in a weighted graph Dijkstra's algorithm and ideas described here regarding acceleration of graph search were implemented (with minor variations from original descriptions) by the author on early stages of this research. The gained experience (which is not detailed in this work but can be found in [2, 45]) has led to the *refuse from the heuristic search* (but not from the graph theoretic methods in general). This is due to the following:

1. heuristic search could be unstable, i.e. minor variations in the source data can cause completely different solution found by the algorithm;
2. heuristics are usually very problem orientated and can not be reused for another (even similar) tasks;
3. all of the heuristic methods gain search acceleration at the cost of the cancellation of guarantee of the optimal solution;
4. the deviation of the solution found with use of heuristics from the really optimal solution remains unknown and can not be controlled.

2.3.6 Template matching

More recently, the dynamic programming and heuristic graph search cost minimisation approaches to curve detection have been reworked in the idea of *snakes* [30], which find a local solution to the minimum cost path problem given an initial path by the user. Snake-type approaches appear in the literature under different names which include: deformable contours, active contours, dynamic contours and finally deformable templates [33, 53]. The term "template" will be used in this work. Template matching (rather than object tracing) allows to reduce computational complexity of the detection algorithm (in comparison to dynamic programming and graph theoretic optimisation approaches) and becomes especially effective when the objects to be detected are heavily occluded by noise, but sufficient prior knowledge of the object geometry is available.

In the original version [30], deformable templates operate by minimising an energy function composed of an internal elastic-type term and an external attraction potential. The internal elastic-type term increases with the contour deformations. The external attraction potential links the contour with the image. The aim is to reach a *compromise* between contour smoothness and an adequate fit to the observed data. If the resulting local minimum does not correspond to the south object, additional terms can be added interactively to the cost function to force the template toward the correct position.

From the point of view of Bayesian estimation, deformable templates are interpretable as *maximum a posteriori* estimators [19]. The internal energy and the external potential terms are associated with the *a priori* probability function and the likelihood function, respectively. The Bayesian estimation perspective has the advantage of giving sense to all of the involved entities.

Despite recent improvements and modifications which allow to overcome limitations of the traditional model such as sensitivity to initialisation and inability to parametrise itself during the deformation process (see [33, 53] and references therein), the existence of an explicit object template remains essential. It is different to serial optimisation techniques, such as graph theoretic and dynamic programming approaches. For them the object model can be defined by means of set of rules, i.e.

implicitly. Such an implicit model is much simpler to realise for irregular objects like crack indications. Thus the graph theoretic and the dynamic programming approaches are preferred to the template matching in this work.

2.4 Summary

An overview of the conventional approaches to border and curve detection is given in this section. No algorithm designed for general purpose segmentation and for border detection particularly is found suitable for the detection of crack-like flaws in NDT radiographs. However, the concept of evaluation of local discontinuities and searching for groups of such local indications which can form an object of "macro" size (in limits of the object's model) will be used further in the next chapter.

The grouping stage of such an approach has to have a global character, i.e. the segmentation decision has to be made at the very end of the grouping process. This is due to the low signal to noise ratio of sought objects. On the other side, the algorithms which require explicit object description (such as Hough Transform and template matching) are generally of very limited applicability here, because it is difficult to create such explicit description for crack-like objects. This suggests the usage of dynamic programming optimisation and graph theoretic approaches. Both of them also have some problems, and appropriate solutions can not be found in the literature. The core problems to be solved here are the construction of a suitable cost estimation function (figure of merit) and the reduction of computational complexity of the search.

Since the segmentation decision has to be made at the end of the grouping process, no *segmentation* is required in the local operator domain. The local operator should calculate the likelihood of presence of an elementary crack indication in the local area, instead of performing a segmentation. This prevents information losses on this early stage of detection due to the too early segmentation.

Design of this local operator, the global grouping technique, the cost estimation function and description of whole algorithm of detection of crack-like indications on NDT radiographs will be done in the next chapter.

Chapter 3

Detection of crack-like indications by global optimisation of a probabilistic estimation function

As it is shown in the previous chapter, the detection of local discontinuities followed by a grouping algorithm is a rather common approach to border detection in noisy images. Such a processing schema is generally adopted for this work too. However, the evaluation of local discontinuities and grouping them into macro objects are considered not as two serial steps of detection, rather as two logical layers. The developed local operator does not detect a discontinuity (e.g. generate the two responses "present" or "absent"), but calculates the likelihood of presence of a discontinuity of a given type. Such an operator provides an abstraction of the physical image to the grouping algorithm and prevents information loss, which may occur in the case of local detectors. The development of this local operator, which is suited preferentially for the task of crack-like defect detection, is described in this chapter.

The likelihood of local indications resulting from the local operator serves as an input for the grouping part of the developed algorithm. The idea behind this is conventional: a global (or integral) feature calculated on a set of consecutive points along the full indication length will be less influenced by the noise than a single local estimation. This allows more accurate detection of the indications with low SNR.

Such a global object feature is called here the *estimation function* and is constructed in such a way, that it reflects the probability of presence of a crack indication along the involved pixels. Arguments of the estimation function are coordinates of the involved image points over which it is calculated. These coordinates are unknown and have to be found. Therefore the task of indication detection is reformulated as a task of finding a set of image coordinates, which maximises the estimation function over all possible positions, shapes and lengths of the indication.

In a general case the computational complexity of such search is enormously high for any practical application. In the presented work the problem of excessive compu-

tational complexity is solved by design of the special object estimation function. The proposed estimation function is constructed as the sum of *a posteriori* information gains in each point along the indication. This allows to find the optimum of the estimation function in a sequential way using serial dynamic programming and to avoid the problem of excessive computational complexity.

The crack detection algorithm developed in this work and described in this chapter is therefore characterised by the following key features:

- the elementary discontinuities are evaluated using a local operator which calculates the likelihood of presence of an indication of the given intensity in the given image point instead of a single yes/no-decision;
- a piecewise linear background model and uncorrelated zero-mean Gaussian noise model are assumed;
- crack indications are recognised by only the two most important features: the rapid stepwise change of the intensity in one direction and the elongated shape of discontinuity in the orthogonal direction;
- the Bayesian model [62] is used for calculating the probability of indication presence;
- the original task of detection of crack indications is posed as a task of maximisation of an informative integral feature of the crack indication (the estimation function) over all possible positions, shapes and lengths of the indication;
- as such an integral feature (estimation function) the sum of the *a posteriori* information gains in all points along the indication is introduced;
- serial dynamic programming is used for the maximisation of the designed estimation function; this can be done within a reasonable and *a priori* known computing time and *without* application of any heuristics (i.e. exactly).

In that follows the developed algorithm is described in details.

3.1 Image model in the local area

The definition of an image model is an essential step in the process of developing an optimal local operator. Several aspects are to be considered here: models of the image background intensity and of the indication intensity in the direction orthogonal to the crack direction, the shape of a crack indication in the image plane and a model of the noise added to the image.

3.1.1 Background model

There are not only crack indications depicted on the real radiograph – indications of other defect types and images of other objects such as lead marks, wire type image quality indicators (IQI), etc. may be present too. In this work indications of such objects are not treated separately and are not included in the image model at all.

They will be classified depending on their appearance on the radiograph as crack-like indications (i.e. false detections will occur) or as a background with additional noise. There are two reasons for such a decision: a) the detection/recognition of other defect types (i.e. porosity, slag inclusion, root undercut, etc.) is a subject of its own and is out of scope of this work; b) the objects like lead marks or image quality indicators are placed at different locations than the flaw suspected areas, so with a correct interpretation of results no confusion between flaws and IQIs or lead marks should occur.

The background intensity changes are mainly determined by the geometry of the setup and the test specimen. For the pipe inspection a common setup consists of a radiation source and a detector which are located outside of the test object and on different sides of it. In this setup effective attenuation of radiation rays which pass through the centre of the pipe perpendicular to the pipe wall is the lowest. The path through the material and therefore attenuation is much bigger for the rays which penetrate pipe walls with angles apart from 90 degree. This results in a high dynamic range of the film radiographs: very dark and very bright background areas can be present within the same image. However these intensity changes are usually gradual and rarely abrupt. This allows an approximation of the background intensity in a local restricted area by a linear model.

3.1.2 Model for indication intensity

The intensity plot of a real crack indication perpendicular to crack propagation is shown in *Fig. 3.1.d*. Such plots are commonly called *profiles*. The crack indication and the linear trend of the background intensity are good visible in *Fig. 3.1.d*. The crack indication reveals itself as a trough in the background. This intensity change is of small spatial dimension due to the narrow width of the crack imaged on the radiograph under optimal radiography conditions. An ideal indication is shown in *Fig. 3.1.b*. It corresponds to an ideal regularly shaped 3D crack (*Fig. 3.1.a*) radiographed in the direction parallel to its propagation in the test specimen. Such a direction of projection is considered to be optimal. If the radiograph is done in a different direction which differs from the optimal, the indication will have reduced contrast (depth of the trough) and a broader width as it is shown on *Fig. 3.1.c*. Real cracks are not ideal objects of a regular shape. Radiographic conditions vary. So indications may appear on the radiograph very differently. Image unsharpness, as a result of the central projection (see *Fig. 1.3*), and scattering effects considerably add to the resulting indication. As a consequence it is unrealistic to use the shape of a given profile as an informative feature. Therefore in this work only the estimated indication intensity z is taken into account. The indication intensity is defined as the difference between the indication grey value calculated on the image area of width w (indication width) and the background grey value estimation calculated on the surrounding of the crack indication, as it is shown in *Fig. 3.1.e*.

3.1.3 Indication shape in the image plane

Due to the different shape of cracks in individual specimens and varied projection conditions, it is not possible to assume a deterministic model for the *global* shape of

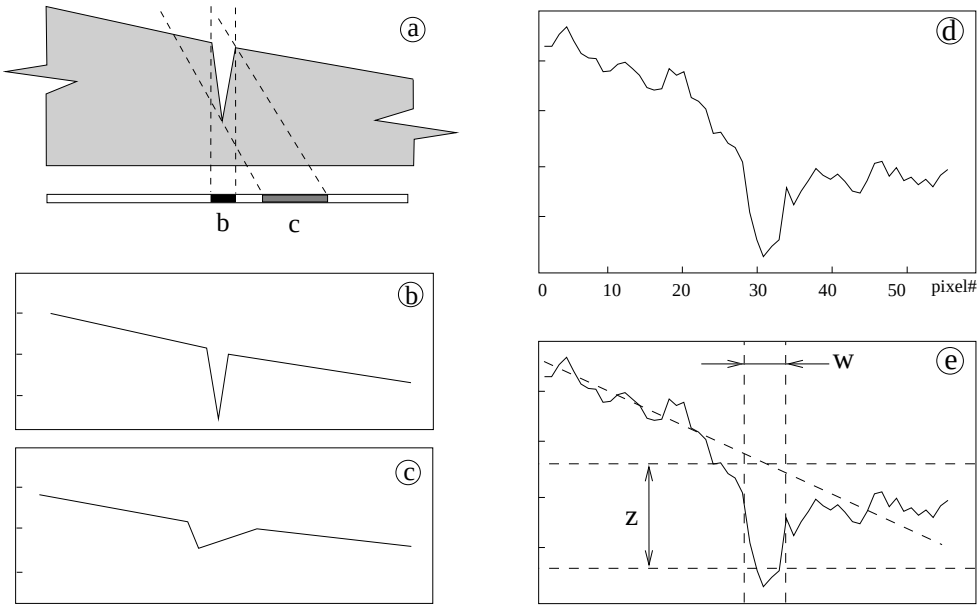


Figure 3.1: a) An ideal regularly shaped crack-like object radiographed in two directions. b-e) Intensity plots (profiles) of crack indications in the direction orthogonal to the crack direction: b) an ideal indication of the ideal crack-like object; c) an ideal indication of the ideal crack-like object under suboptimal radiographic conditions; d) a real crack indication with high SNR; e) estimation of the intensity z of the indication of width w according to the assumed model.

a crack indication in the image plane. It is also not necessary, while only a restricted local area is used for the estimation of elementary indications. Therefore the following assumption is enough for the estimation of elementary indications: in a local area the crack indication can be approximated by a straight line.

3.1.4 Noise model

There are several sources of noise which contribute to the digital radiographic image: test specimen irregularities, radiation quantum noise, film granularity (in case of a film based system) and electronic noise of the detector or noise of a film scanner. These noise sources are inevitably present in any radiography system.

The electronic noise is a result of thermoelectric fluctuations in electronic components of the analog part of the detector. In an appropriately built system, and for the typical exposure times or film densities, the electronic noise can be neglected as compared to the radiation quantum noise.

Quantum noise is a result of the dual nature of radiation. At the energies used for radiography the corpuscular character becomes relevant and the radiation beam shall be considered as a stream of discrete photons. The amount of photons n_i which is emitted by the radiation source or hits a given area of the detector over a given period of time is random and can be described by the Poisson probability distribution [21]:

$$P(n_i) = \frac{\lambda_i^{n_i}}{n_i!} e^{-\lambda_i}, \tag{3.1}$$

where λ_i is the mathematical expectation of n_i. The detector itself does not register all incident photons: each photon can be detected with a certain probability p_d, which describes the detector efficiency, or missed with probability (1-p_d). For the solid state detectors as far as for the film based systems without reciprocity failure, the detections of photons are independent from each other. Therefore the number of detected photons n can be described by the binomial distribution [4]:

$$P(n|n_i) = C_{n_i}^n p_d^n (1 - p_d)^{(n_i - n)}. \tag{3.2}$$

Then:

$$P(n) = \sum_{n_i=0}^{\infty} P(n|n_i) P(n_i) = \frac{(p_d \lambda_i)^n}{n!} e^{-p_d \lambda_i}. \tag{3.3}$$

In other words $P(n)$ is simply Poisson distribution with $\lambda = p_d \lambda_i$. For large λ, the Poisson distribution statistically approaches the normal distribution. These statistical properties of quantum noise are related to the temporal domain. To be useful for the image analysis, an additional assumption about process ergodicity has to be made. This means that statistical properties of noise in the spatial domain are assumed equal to the statistical properties in the temporal domain.

For the film-based systems the film graininess has to be considered. Film graininess is a result of the finite size of the silver halide crystals (grains), which form latent image on the radiographic film. Contribution of the film graininess to the overall system noise becomes significant if the film scanning aperture is of comparable size as of the grains, i.e. by scanning with a very high resolution [4]. Such high resolution scans are not used in industrial radiography because the resolution

is limited in practice by the inherent unsharpness of the radiographic system (see *Fig. 1.3*). Due to this fact the film graininess noise can be ignored.

The inherent unsharpness of the radiographic system (see *Fig. 1.3*) and the scattering of the radiation in the test specimen will cause that each point of the detector will be affected by some part of the specimen volume (not by an abstract line as it is usually shown on illustrations). Although there is no possibility to make any practical assumptions about irregularities in the test specimen or surface imperfections, the influences from all imperfection from a certain volume (due to the considerations above) will be summed on the detector surface.

Scattering effects in the image detector have an additional impact on the resulting average signal intensity and the noise component. The resulting image will be a convolution of the image on the input of the detector and the modulation transfer function (MTF) of the detector, i.e. signal from every element of the detector will be a weighted sum of incident radiation intensities over some neighbourhood. This results in some correlation between neighbour pixels. As this correlation is usually minor, it is not included in the noise model used here.

The considerations given above allow the assumption of Gaussian zero-mean distribution of the noise in the radiographic image.

3.2 Probability of indication presence

Given the image model, it is possible to calculate for each image point (x, y) and given indication width w_{ind} the estimated indication intensity $z(x, y)$. The certain values of $z(x, y)$ in different image points are *instantiation* of a random variable Z. Z is a sample from population of estimated object intensities. Z is a random due to the noise, added to the image:

$$Z = h + \nu, \tag{3.4}$$

where ν is the random noise component (Gaussian zero-mean) and h is the true indication intensity. Under the true indication intensity h the defect intensity is meant which is calculated according to the assumed image model in the idealised case of absence of the noise.

For an ideal system without noise ($\nu = 0$), the estimation of indication intensity $z(x, y)$ is sufficient, since it already equals to $h(x, y)$. But for a system with noise the estimation of $z(x, y)$ alone is not enough to describe a discontinuity since indications of different intensities h may result in the same estimation z due to the noise fluctuations. And vice verse: different h may be estimated as the same z. In such non-ideal system the most complete but yet not excessive information about discontinuity is given by the probability distribution of presence of the sought discontinuity with intensity h given the estimation $z(x, y)$. This probability distribution is characterised by its density function $\rho(h|Z = z(x, y))$. Or shortly: $\rho(h|z)$.

From the Bayesian point of view $\rho(h|z)$ is the *a posteriori* distribution because it has to be calculated for known $z(x, y)$. The composite probability formula can be used for the calculation of $\rho(h|z)$:

$$\rho(z)\rho(h|z) = \rho(h)\rho(z|h) \quad \Rightarrow \quad \rho(h|z) = \frac{\rho(h)\rho(z|h)}{\rho(z)}, \tag{3.5}$$

where: $\rho(z)$ is density of the *a priori* probability distribution that the estimated indication intensity Z takes the value z; $\rho(h)$ is density of the *a priori* probability of occurrence of a crack indication with an intensity h; and $\rho(z|h)$ is density the *a priori* probability that Z takes the value z with the condition of a true indication presence with intensity h.

The term $\rho(z|h)$ of (3.5) is also called *likelihood* of presence of indication h and denoted as $L_z(h) = \rho(z|h)$. The difference between the notations $\rho(z|h)$ and $L_z(h)$ is that the former looks like a function of z for a given h and has to be considered as *a priori* probability. But in reality z can be estimated from the image and the dependence of ρ from h is of primary interest. This leads to necessity of consideration of $\rho(z|h)$ as *a posteriori* probability distribution. Therefore the second notion is more intuitive since $L_z(h)$ is explicitly written as a function of h. The likelihood of presence of the crack indication $L_z(h)$ in a given image point (x, y) can be calculated from the image using the assumed image model without any additional assumptions or simplifications.

For calculation of $\rho(h|z)$ a knowledge of the *a priori* probability distribution $\rho(h)$ is essential. The difficulty consists in the usually unknown $\rho(h)$ in practice and the impossibility of its estimation from the image under consideration. In the case of unknown $\rho(h)$ the making of certain assumptions becomes inevitable. Such assumptions about density of the *a priori* probability distribution of indication intensity $\rho(h)$ is made in this work too (see later in Section 3.5). The assumptions made here are not hard-coded in the algorithm, instead of this the end-user has the possibility to adjust them to fit the particular application best.

3.2.1 Likelihood of indication presence

We denote the image data in the local neighbourhood of the running point (x, y) as $\{g(x, y)\}$. The estimation of indication intensity $z(x, y)$ can be calculated according to our image model as a function of the local image data $\{g(x, y)\}$: $z(x, y) = f(\{g(x, y)\})$. Denoting the estimation of the mean square deviation of the image noise by σ, the likelihood of presence of the crack indication of intensity h can be written a function of z, σ and h:

$$L_z(h) = \rho(z|h) = f(z, \sigma, h) = f(\{g(x, y)\}, \sigma, h). \tag{3.6}$$

Let us consider a profile of an indication of width w_{ind} located on a changing background (*Fig. 3.2*). According to the used image model the background can be represented by a straight line:

$$g_{bkg}(i) = a + k(i - x), \tag{3.7}$$

where g_{bkg} is the estimated intensity of the background, a and k are parameters of the linear model and x is the coordinate of the indication centre (*Fig. 3.2*).

The parameters of the background model can be estimated by minimising the sum of squared differences between the calculated background intensity $g_{bkg}(i)$ and

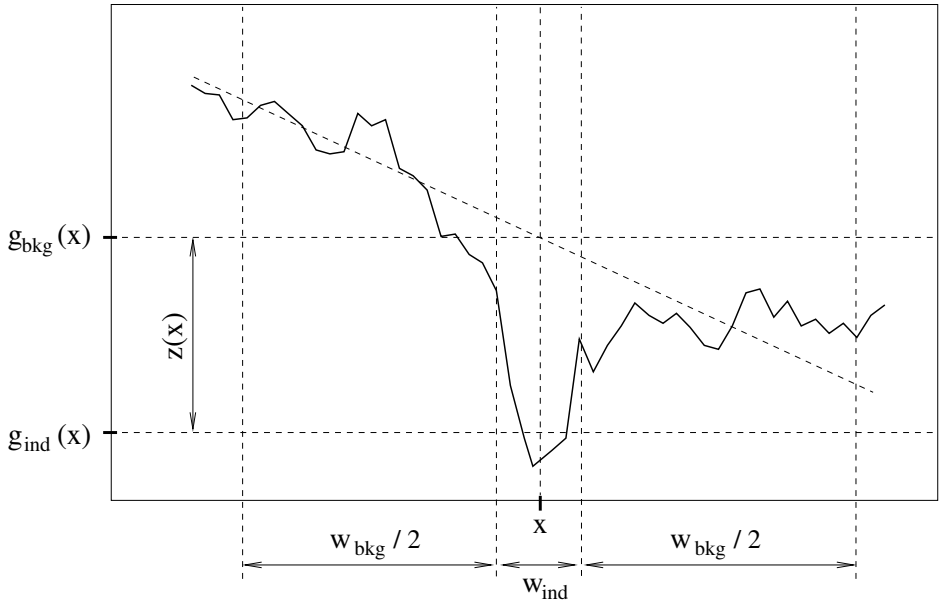


Figure 3.2: Estimation of intensity of a crack indication.

real pixel intensity $g(i)$ (least square method):

$$(a, k) = \arg\min_{(a,k)} \varepsilon, \tag{3.8}$$

$$\varepsilon = \sum_i (g(i) - g_{bkg}(i))^2 = \sum_i (g(i) - a - k(i - x))^2. \tag{3.9}$$

Two areas are used for background estimation, each of width $w_{bkg}/2$, symmetrically located on different sides of the crack indication:

$$i \in [x - \frac{w_{ind}}{2} - \frac{w_{bkg}}{2}, x - \frac{w_{ind}}{2}] \cup [x + \frac{w_{ind}}{2}, x + \frac{w_{ind}}{2} + \frac{w_{bkg}}{2}]. \tag{3.10}$$

The criterion ε reaches its extremum at the point where its partial derivatives simultaneously become zero:

$$\frac{\partial \varepsilon}{\partial a} = 0, \quad \frac{\partial \varepsilon}{\partial k} = 0, \tag{3.11}$$

After solving for a:

$$a = \frac{\sum_i g(i) - k \sum_i (i - x)}{w_{bkg}}. \tag{3.12}$$

The term $\sum_i (i - x)$ identically equals zero because of the symmetry (around x) of the interval on which g_{bkg} is considered. Therefore:

$$a = \frac{1}{w_{bkg}} \sum_i g(i). \tag{3.13}$$

To calculate the intensity of the crack indication it is necessary to know the background intensity in the centre point x only:

$$g_{bkg}(x) = a + k(x - x) = \frac{1}{w_{bkg}} \sum_i g(i). \tag{3.14}$$

As $g_{bkg}(x)$ is defined solely by a, the estimation of the parameter k can be omitted.

Since the background estimation $g_{bkg}(x)$ is a sum of w_{bkg} normally distributed random values, it will be normally distributed too. The mean square deviation of $g_{bkg}(x)$ will be $\sqrt{w_{bkg}}$ times smaller than mean square deviation of noise added to each image pixel:

$$\sigma_{bkg} = \frac{\sigma}{\sqrt{w_{bkg}}}. \tag{3.15}$$

The same considerations are valid for the estimation of the indication grey value and the result is as follows:

$$g_{ind}(x) = \frac{\sum_j g(j)}{w_{ind}}, \quad j \in [x - \frac{w_{ind}}{2}, x + \frac{w_{ind}}{2}], \tag{3.16}$$

$$\sigma_{ind} = \frac{\sigma}{\sqrt{w_{ind}}}. \tag{3.17}$$

The estimation of the indication intensity z is the difference between the background grey value and the indication grey value:

$$\begin{aligned} z(x) = g_{bkg}(x) - g_{ind}(x) &= \frac{\sum_i g(i)}{w_{bkg}} - \frac{\sum_j g(j)}{w_{ind}}, \\ i &\in [x - \frac{w_{ind}}{2} - \frac{w_{bkg}}{2}, x - \frac{w_{ind}}{2}] \cup [x + \frac{w_{ind}}{2}, x + \frac{w_{ind}}{2} + \frac{w_{bkg}}{2}], \\ j &\in [x - \frac{w_{ind}}{2}, x + \frac{w_{ind}}{2}]. \end{aligned} \tag{3.18}$$

And:

$$\sigma_z = \sqrt{\left(\frac{\sigma}{\sqrt{w_{bkg}}}\right)^2 + \left(\frac{\sigma}{\sqrt{w_{ind}}}\right)^2} = \sigma \sqrt{\frac{w_{bkg} + w_{ind}}{w_{bkg} w_{ind}}}. \tag{3.19}$$

Since the presence of a real indication of intensity h in the point x is assumed, the case that $z(x) \neq h$ is only a result of added noise which causes an inaccuracy of estimation. But the mathematical expectation $M(Z)$ equals h because of the zero-mean noise model. That means:

$$Z \sim N(z; h, \sigma_z), \tag{3.20}$$

where $N(z; h, \sigma_z)$ denotes the Gaussian distribution of argument z with mean h and standard deviation σ_z. Therefore the sought $\rho(z|h)$ is just the density of probability that a normally distributed Z possesses value z:

$$L_z(h) = \rho(z|h) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-\frac{(h-z)^2}{2{\sigma_z}^2}}. \tag{3.21}$$

Expression (3.21) shows the necessity and importance of a correct estimation of σ_z and h. For example, two likelihoods $L_z(0) = \rho(z|0)$ (likelihood of "no indication")

and $L_z(h') = \rho(z|h')$ (likelihood of indication with intensity h') have to be computed and compared. Overestimation of σ_z tends to diminish the difference between $L_z(0)$ and $L_z(h')$ and both likelihoods are lowered in comparison to the correct values. This can be explained as follows: if the estimation of noise level is very high, then any closeness of estimation of the crack intensity z to 0 or h' can be considered as random and the respective likelihoods are low. Underestimation of σ_z will, on opposite, amplify the difference between the estimated likelihoods since any closeness to 0 or h' must be considered as appropriate.

The correct estimation of h' is even more important since the sign in comparison $L_z(0) \lessgtr L_z(h')$ is defined by the sign in comparison $|z-0| \lessgtr |z-h'|$. For the same estimation z and the noise level σ_z the $L_z(0)$ or the $L_z(h')$ can become bigger or lower depending of the intensity of the expected indication h'. For example, let us consider $z = 3\sigma_z$ (SNR = 3). For $h = 0$ ("no indication" hypothesis), the probability that z takes a value outside the interval $(\infty, 3\sigma_z)$ can be estimated applying for normally distributed Z the rule of "three sigma":

$$P(z > 3\sigma_z \mid h = 0) \;=\; 1 - P(z \leq 3\sigma_z \mid h = 0) \;\approx\; 0.0014. \tag{3.22}$$

This is a rather low value, so, without taking into account h', the presence of an indication can be assumed. Unfortunately this assumption is wrong if the expected indication h' is taken into account and $h' > 2z$:

$$P(z > 3\sigma_z \mid h = 0) \;>\; P(z < 3\sigma_z \mid h = h'), \tag{3.23}$$

As a consequence, the absence of the indication becomes more likely.

3.3 Implementation of the local operator

3.3.1 Size and shape of the operator

As 3.19 shows, the accuracy of $\rho(z|h)$ estimation depends on the size of the areas used for the estimation of background and indication grey levels. These areas can be defined on practice as conventional: by *masks*. The bigger the masks in use – the less the estimation will be influenced by the noise. At the same time: the bigger the linear mask size – the more probably the real image, covered by the mask, will differ from our background and indication models.

The calculations given above are made on an one-dimensional image profile. One of the ways to enlarge the mask without making it too wide is to extend the mask in the second dimension in the direction orthogonal to profile (see *Fig. 3.3*). Until both masks for background calculation have the same size and remain symmetric in the xy-plane in respect to the running point (x, y), the calculations given above for 1D profiles will be also valid for the 2D case. So the estimation of the background intensity g_{bkg} in the running point (x, y) is the average of pixels covered by the background masks:

$$g_{bkg}(x,y) = \frac{1}{2N_{bkg}} \sum_{i=0}^{X_{bkg}-1} \sum_{j=0}^{Y_{bkg}-1} K_{bkg}[i,j] \times g(x + \frac{X_{bkg}}{2} - i, y + \frac{Y_{bkg}}{2} - j), \tag{3.24}$$

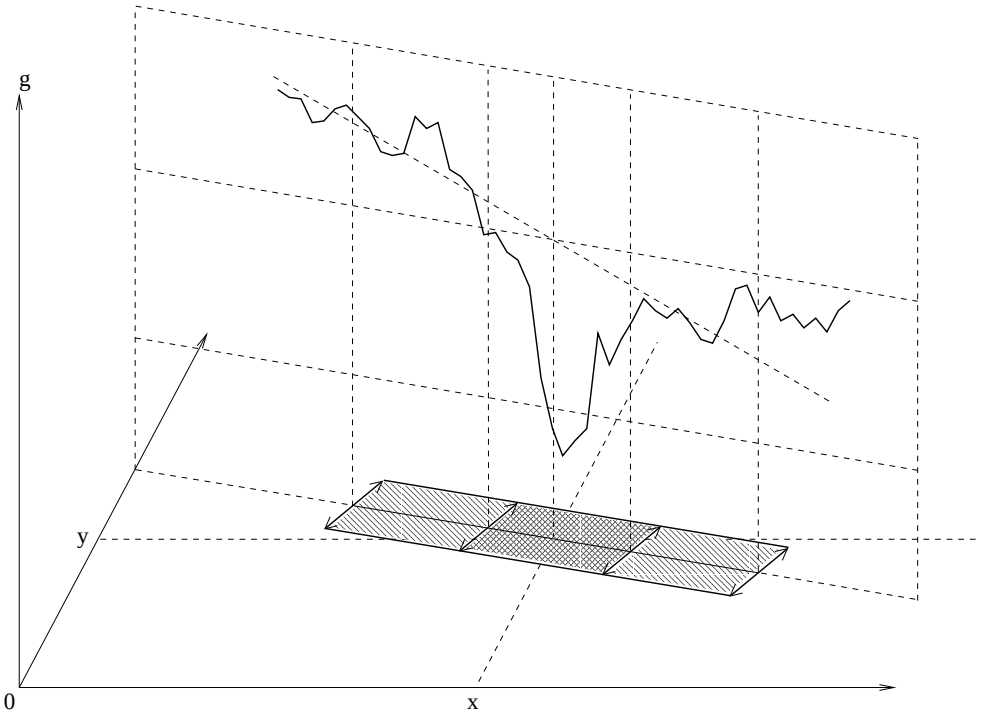


Figure 3.3: Extension of an one-dimensional mask to the second dimension.

where K_{bkg} denotes an rectangular binary mask, wherein non-zero elements define the required area for background estimation, X_{bkg} and Y_{bkg} are the dimensions of the K_{bkg} and N_{bkg} is the number of non-zero elements in K_{bkg}.

Similarly, for the indication grey level:

$$g_{ind}(x,y) = \frac{1}{2N_{ind}} \sum_{i=0}^{X_{ind}-1} \sum_{j=0}^{Y_{ind}-1} K_{ind}[i,j] \times g(x + \frac{X_{ind}}{2} - i, y + \frac{Y_{ind}}{2} - j). \quad (3.25)$$

The 2D operator defined in this way is *directionally* selective, i.e. it responds optimally if indication direction and operator direction coincide. It is a desirable feature. Finding an operator with maximum response over a set of differently oriented operators gives not only information about the indication magnitude but also about the indication direction. Another side of this approach is the necessity of sampling of the 360 degree circle to a set of discrete directions and providing background and indication masks for each of them. An accurate representation requires a big number of discrete directions, a big number of masks and hence high computational complexity.

On *Fig. 3.4.a* examples of masks of size 3 and 4 for the angles from 0° till 90° are given. An additional space is introduced between the indication mask and background masks. This space is addressed to eliminate the influence of the transitions areas between indication and background on the estimation of indication intensity.

An rectangular mask rotated to some degree will not fit perfectly to the discrete image raster (dashed masks on *Fig. 3.4.a*). Therefore the rotated rectangular masks can be approximated by some complex-shaped masks in the best possible way (solid masks in *Fig. 3.4.a*). The necessity of generation, storage and calculation of masks for every direction can be completely eliminated by the following approximation: only one rectangular mask, which has all element set to unity, is used for estimation of indication and background grey levels. The borders of the mask remain parallel to the coordinate axes for all directions and only the position of the mask centre is changed with respect to the running point as a function of the required direction. This is illustrated in *Fig. 3.4.b*. This approach has a significant computational benefit: since for the given indication size w_{ind} all masks have the same size and shape $w_{ind} \times w_{ind}$, it is possible to average an image with this mask only once and then take the required value "on demand" by algebraically calculating the coordinates of centres of background masks for the given position of the running point and direction of the operator.

3.3.2 Recursive calculation of estimation of background and indication grey values

Further acceleration can be achieved by recursive calculation of the moving average [1]. In the conventional approach to the calculation of moving average the sums are calculated independently from each other for each mask position. The recursive approach makes use of the overlapping of masks centred at two neighboured pixels. This allows to use the sum calculated for the previous mask position and "update" it for the current position. A general approach which works for masks of arbitrary shape is proposed here.

On the preprocessing step two lists are build. The first list (let us denote them as L^-) contains coordinates of the binary mask elements which are zero for the mask centred around the current point i, but set to unity for the previous mask position $i-1$. The second list (L^+) contains coordinates of elements which are set to unity at the current mask position, but are zero at the previous one (*Fig. 3.5*). The lists contain relative coordinates of pixels in respect to the current coordinate i of the mask centre. This allows reuse of the lists for arbitrary mask positions.

The recursive calculation of moving average S in the position (x, y) is made by subtraction of grey values of image pixels from L^- and addition of grey values of pixels from L^+ to the moving average calculated in the previous position $(x-1, y)$:

$$S(x,y) = S(x-1,y) - \sum_{(i,j)\in L^-} g(i,j) + \sum_{(i,j)\in L^+} g(i,j). \tag{3.26}$$

For a rectangular binary mask without zero elements with borders parallel to the coordinate axes the L^- is simply the first column of the mask at the $(i-1)$ position and L^+ is the last column of the mask at the i position.

The described approach can be called *one-dimensional recursion*, because overlapping only in one dimension is used. However, it is possible to recursively calculate moving sums during changing of both coordinates.

For this purpose it is necessary to calculate and store partial one dimensional

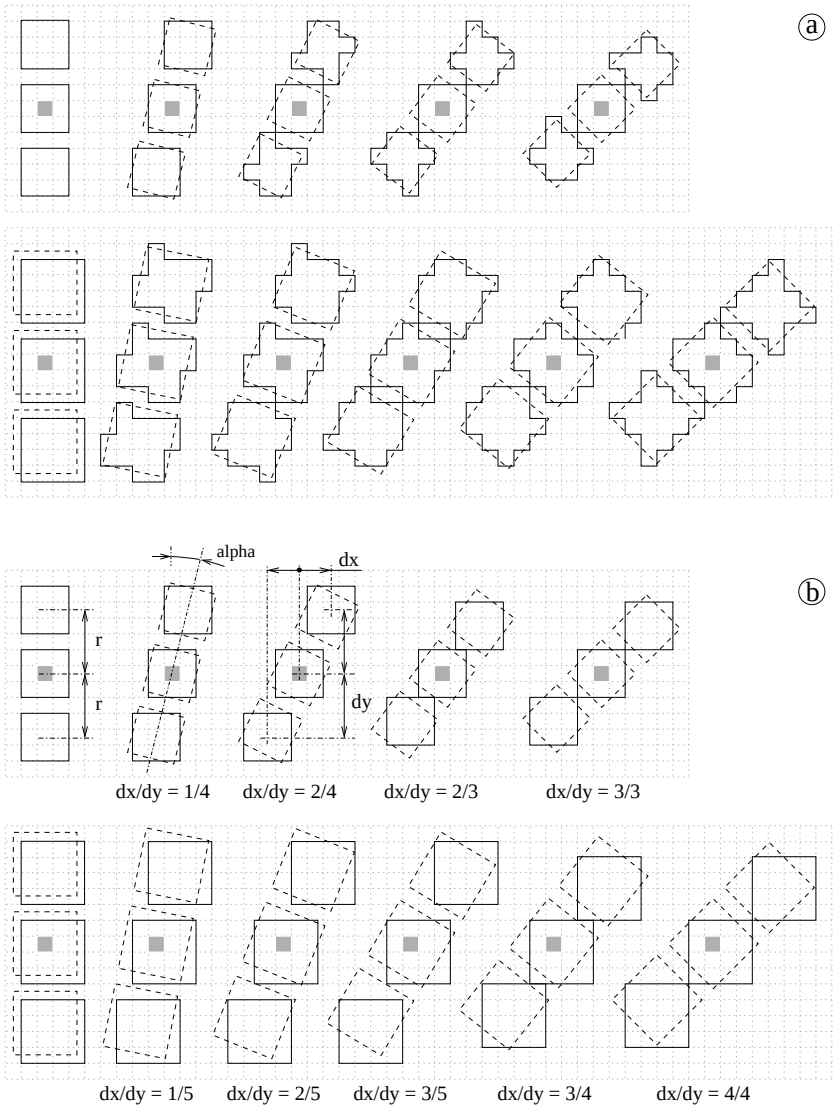


Figure 3.4: Two approaches for calculation of the background and indication grey levels. The dotted mesh denotes image raster, grey squares denote pixels in the centres of the respective masks, dashed squares denote ideal rotated masks. a) Explicit definition of binary masks which approximate the ideal masks to the image raster in the best way (borders are shown by solid lines). b) Averaging by a single rectangular mask and taking the value for a required direction "on demand" by algebraically calculating centre coordinates of the best suited rectangle. This conserves memory and improves calculation speed significantly.

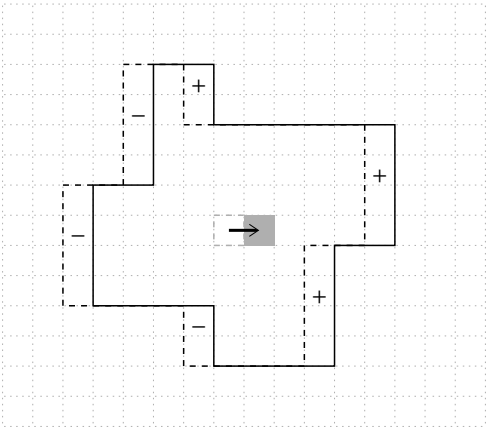


Figure 3.5: Illustration of L^- and L^+ lists idea used for recursive calculation of moving average. Pixels designated by "-" are added to L^- list, pixels designated by "+" - to L^+.

sums:

$$S'(x,y,i) = \sum_{j=0}^{Y_k-1} K[i,j]g(x, y - \frac{Y_k}{2} + j), \\ y = \frac{Y_k}{2}, \;\; x \in [0,X), \;\; i \in [0,X_k), \tag{3.27}$$

where $K[i,j]$ denotes the (i,j) element of the mask, X_k and Y_k are dimensions of the mask and X and Y are image dimension. Then the sum of interest can be computed recursively along the x axis as before (while y is unchanged):

$$S(x,y) = S(x-1,y) - \\ - S'(x - \frac{X_k}{2} - 1, y, 0) + S'(x - \frac{X_k}{2} + X_k - 1, y, X_k - 1), \\ y \in (\frac{Y_k}{2}, Y - Y_k + \frac{Y_k}{2}), \;\; x \in (\frac{X_k}{2}, X - X_k + \frac{X_k}{2}). \tag{3.28}$$

The partial sums $S'(x,y,i)$ are updated recursively too. This has to be done after each increment of the y coordinate (y is incremented first after the sums $S(x,y)$ are calculated for all x coordinates):

$$S'(x,y,i) = S'(x,y-1,i) - \\ - K[i,0]g(y - \frac{Y_k}{2} - 1, x) + K[i,Y_k - 1]g(y - \frac{Y_k}{2} + Y_k - 1, x), \\ y \in (\frac{Y_k}{2}, Y - Y_k + \frac{Y_k}{2}), \;\; x \in [0,X), \;\; i \in [0,X_k). \tag{3.29}$$

The computational complexity of the conventional calculation of moving average for an image of size $X \times Y$ with a mask size $X_k \times Y_k$ is $O(XYX_kY_k)$. The computational complexity of the described algorithm which uses two-dimensional recursion

does not depend on the size of the mask and is only $O(XY)$. The advantage is therefore proportional to the number of kernel elements and can be huge for large kernels.

3.3.3 Workflow for calculation of likelihood of presence of indication in the local area

The range of indication widths, usable for the given application, has to be provided as an input parameter. This range can be limited (by the user) from the upper side by using *a priori* information about the sought indications and from the lower side using knowledge about limits of the acquisition technique in use. Then, the moving average is recursively calculated for each width of the indication mask.

After experimenting with different masks, it have been found that the usage of the same mask for indication and background estimation gives sufficient accuracy. Obviously, this mask is differently centred for indication and background estimation, but has the same shape and size (see *Fig. 3.4.b*). That means that the moving average calculated for estimation of the indication grey level can be used for background grey level estimation too. This speeds up the calculations considerably.

After the moving average is computed, the likelihood of indication presence can be calculated for an arbitrary image point (x, y) using (3.21) and (3.19). The operator needs input values as follows: the indication width w and the direction α to calculate $z(x, y, w, \alpha)$, the estimation of noise intensity σ to calculate $\sigma_z(w)$ and the *a priori* indication intensity h:

$$\begin{aligned} L_h(x, y, w, \alpha) &= \rho(z|h) = \frac{1}{\sqrt{2\pi}\sigma_z(w)} e^{-\frac{(h - z(x,y,w,\alpha))^2}{2(\sigma_z(w))^2}}, \\ z(x, y, w, \alpha) &= g_{bkg}(x, y, w, \alpha) - g_{ind}(x, y, w) = \\ &= \frac{g_m(x - d_x, y + d_y, w) + g_m(x + d_x, y - d_y, w)}{2} - g_m(x, y, w), \\ \sigma_z(w) &= \sqrt{\frac{5}{4}} \frac{\sigma}{w}, \end{aligned} \tag{3.30}$$

where α defines the operator direction as it is shown on *Fig. 3.4.b*, $g_m(x, y, w)$ is the mean grey value calculated with the $w \times w$ mask centred at the point (x, y), $d_x = round(r \sin \alpha)$, $d_y = round(r \cos \alpha)$ and $r = w + 1$.

3.4 Detection of elongated objects as optimisation problem

Only a local estimation of probabilities of indication presence is not enough for reliable crack detection. This is, as it was already mentioned before, due to the low signal to noise ratio of the sought objects. Therefore in the developed crack detection algorithm, described in this work, the crack indications are detected on a global basis. A global indication characteristic called *estimation function* will be designed in this section. The arguments of the estimation function are coordinates of the involved

image points over which it is calculated. Careful design of the estimation function allows to pose the task of detection of crack indications as a task of finding a set of image coordinates which maximises the estimation function.

It is necessary to impose some restrictions on the arguments of the estimation function in order to keep the task of optimisation of the estimation function adequate to the task of crack detection. This is, first of all, the requirement for connectivity of the involved points. In other words, the points have to form a continuous curve. Then, it is possible to define the preferred crack direction (vertical or horizontal cracks), to require certain smoothness of the crack indication or to prohibit sharp turns of the indication shape.

Despite of the restrictions on the indication shape, the maximisation of the estimation function by means of exhaustive enumeration of all possible combinations has a restrictively huge computational complexity. As a consequence, numerous optimised search techniques have been proposed in last decades (see Chapter 2 an references therein). The common feature of the most of those techniques is the employment of heuristics in order to reduce the search complexity. This is the basic difference of the approach developed in this work: the optimum of the estimation function designed here can be found *without* heuristic assumptions during optimisation procedure, i.e exactly. The known techniques such as graph searching and dynamic programming (Chapter 2) can be applied successfully for this optimisation.

3.5 Synthesis of the estimation function

3.5.1 An optimised estimation function from the graph theoretic point of view

As it is shown in Chapter 2, the optimisation of the estimation function in the general form (as an arbitrary function of all path points) is connected in practice with prohibitive computational complexity. This is due to the fact that the original graph of possible indication prolongations must be converted to a tree of unique paths through this graph. To bring the search complexity into the real world limits some strong heuristics are usually applied. The problem of such an approach is, that unpredictable differences can occur between the solution obtained with usage of heuristics and the really optimal solution.

The search complexity can be reduced not only by the usage of heuristics. This can be done if the tree of unique paths is considered as the tree of unique values of the estimation function (finally we want to find the maximum of the estimation function) and some restrictions are posed on the estimation function formula.

The main idea is as follows. If the estimation function f^* can be represented as

$$f^*(p_1,..,p_n) = f(f^*(p_1,..,p_{n-1}), p_{n-m+1}, p_{n-m+2},..,p_n), \tag{3.31}$$

i.e. the value of f^* in each point of the path depends only on the value of f^* in the previous point and some number m of previous points, where $m \ll n$ and n is length of the path, then the size of the search tree can be dramatically reduced. This is due to the following: among several paths which coincide in the last m points (see *Fig. 3.6*) only the one path which has the biggest value of the estimation function have

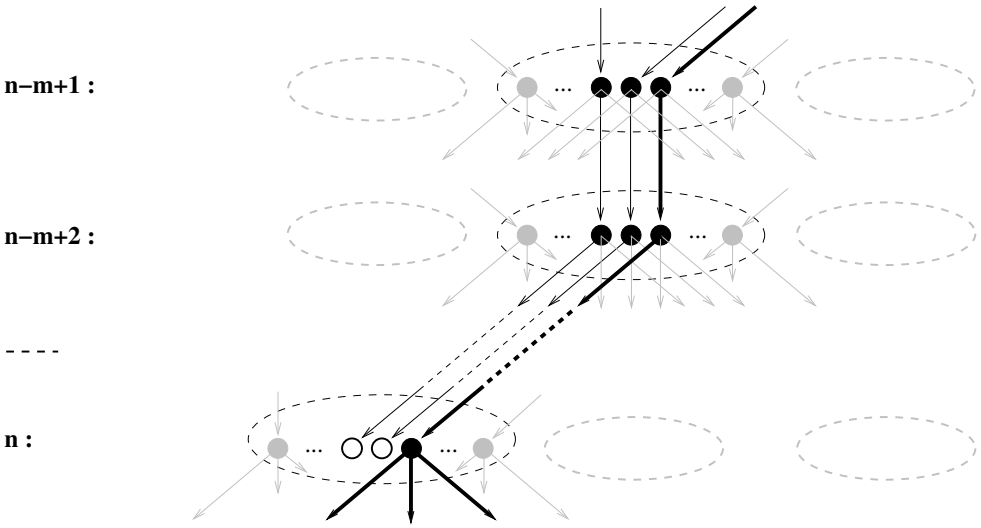


Figure 3.6: If the estimation function f^* depends only on the last m path nodes, then among paths which coincide on the last m steps only the one which has the best f^* has to be continued – the others can be discarded.

to be further explored. All other paths with lower estimations can be discontinued at this point, since they coincide with the best path in the last m points, have lower values of the estimation function and therefore have no chances to outperform the best path. An illustration is given on *Fig. 3.6*.

The smaller the m, the smaller the search tree becomes (which has to be explored). The border case arise if $m \leq 2$ ($m = 1$ or $m = 2$). Then the size of the search tree reduces down to the size of graph of possible paths, which is the absolute minimal tree which is necessary to explore to get a strict solution. The same result can be obtained considering optimisation of the estimation function from the point of view of dynamic programming.

3.5.2 An optimised estimation function from the dynamic programming point of view

Besides the graph searching, the optimum (maximum or minimum) of an estimation function can be found using other optimisation methods. An interesting case arise if the estimation function f^* (which is a function of pixel coordinates through which the path goes) can be represented as a sum of terms each of which depends on only a few variables:

$$f^*((x_0,y_0),(x_1,y_1),...,(x_i,y_i),...,(x_N,y_N)) = f^*(p_0,p_1,...,p_i,...,p_N) = \\ = f_0^*(p_{k_0},...,p_{l_0}) + f_1^*(p_{k_1},...,p_{l_1}) + ...f_j^*(p_{k_j},...,p_{l_j}) + ... + f_M^*(p_{k_M},...,p_{l_M}), \tag{3.32}$$

where p_i denotes coordinate pairs (point) (x_i, y_i) and the indices k_j and l_j define a range of arguments on which each component function f_j^* depends. Since the pixel coordinates x_i, y_i are discrete, the optimum of f^* can be found by a multistage optimisation process called *serial dynamic programming*.

Principles and an example of function optimisation using serial dynamic programming are already reviewed in Chapter 2. Dynamic programming is a general optimisation technique which can be used for different purposes, extensive literature exists on this topic [6, 32, 38]. Here we focus only on one aspect relevant for the creation of an effective detection algorithm: analysis of the computational complexity and the storage requirements of the dynamic programming approach.

In general, a multistage optimisation method can be described as a procedure in which the optimisation is carried out separately for each variable p_i. Of course, this optimisation must be performed for all the values of the independent variables p_j which are non-linearly connected with the variable p_i, i.e. it exists at least one term of the estimation function which depends on both p_i and p_j. For example:

$$f^*(p_0, p_1, p_2, p_3, p_4) = f_0^*(p_0, p_1) + f_1^*(p_0, p_1, p_2) + f_2^*(p_2, p_3) + f_3^*(p_2, p_3, p_4), \quad (3.33)$$

where f_j^* are component functions which cannot be represented as sum of terms. Now we carry on optimisation with respect to say variable p_0. The variables p_1 and p_2 are non-linearly connected with p_0. So the following substitution can be made

$$f_0^{**}(p_0, p_1, p_2) = f_0^*(p_0, p_1) + f_1^*(p_0, p_1, p_2) \quad (3.34)$$

and then f^* can be rewritten as

$$f^*(p_0, p_1, p_2, p_3, p_4) = f_0^{**}(p_0, p_1, p_2) + f_2^*(p_2, p_3) + f_3^*(p_2, p_3, p_4). \quad (3.35)$$

In the last expression the terms f_2^* and f_3^* do not depend on p_0. Thus the optimisation with respect to p_0 can be made independently for f_0^{**}:

$$f_{0\,opt}^{**}(p_1, p_2) = \max_{p_0} f_0^{**}(p_0, p_1, p_2). \quad (3.36)$$

After that f_0^{**} can be substituted by $f_{0\,opt}^{**}$. By this substitution the variable p_0 is "eliminated", so f^* can be rewritten again as a function of only p_1, p_2, p_3 and p_4:

$$f^*(p_1, p_2, p_3, p_4) = f_{0\,opt}^{**}(p_1, p_2) + f_2^*(p_2, p_3) + f_3^*(p_2, p_3, p_4). \quad (3.37)$$

The cost of this operation (in computing time and storage requirements) is dependent on the number of variables K which constitute the domain of function $f_{0\,opt}^{**}$ ($K = 2$ for this example) and number P of possible values each variable can take.

The multistage optimisation can be seen as a step-by-step elimination of all variables. So the computational complexity of this process is therefore $O(NP^K)$, there N is number of arguments in f^* and thus number of necessary eliminations. The obtained estimation of computational complexity helps to analytically define the requirements to the construction of the estimation function in order to be easy to optimise.

3.5.3 On the computational complexity

The above given considerations about the computational complexity of optimisation of the estimation function result in the following statement: if the estimation function can be represented as a sum of simple independent terms, then its maximum can be found exactly (without any heuristics) and with an acceptable computational complexity.

This approach has its drawback: holding the requirement to be representable as a sum of simple terms makes it difficult to incorporate certain features in the estimation function. Nevertheless it was decided to use this approach to the construction of the estimation function and to see what is possible to reach in this way. The motivations are:

- all heuristic assumptions are located in the estimation function – the algorithm which searches for the optimum is free of any heuristics and therefore has predictable behaviour;
- the adequacy of the estimation function to the application problem can be checked and adjusted independently of the search algorithm;
- if some feature cannot be included in the estimation function, then we know, at least, what has been omitted. However, if some features are included, then we know that they have been fully utilised (since the optimisation procedure is exact).

From this point of view the simple average contrast over the indication length can not be used as an estimation function. This is because a representation of the average contrast as a sum of independent terms is not possible (due to the unknown indication length, which is present in each term of the sum).

3.5.4 On shape features

The estimation function which uses global indication context can theoretically utilise more information as only grey values of points along the indication. These are, first of all, several features which characterise the indication shape. For example, the first and the second centred moments of a path curvature: the mean curvature along the full path and the mean square of variations of local curvatures from the mean curvature. A small mean square variation of local curvatures could be a sign of an undercut, because the undercut indication often appears as a straight line. On the opposite, a big variation of local curvatures could be a sign of false detection caused by a random conglomerate of noise fluctuations.

As another shape feature the presence of bifurcation points can be used. A bifurcation point occurs when an indication splits into branches that continue in two different direction. This is characteristic for crack indications. Therefore the presence of bifurcation points is a sign which allows to distinguish between false crack-like objects (i.e. undercuts) and the real crack indications.

Unfortunately humans (inspectors) have difficulties with formal expression of their experience on this topic. This is the same for the curvature analysis as far as for the bifurcation detection.

From the other side it is difficult to embed shape features in the estimation function in such a way that it still can be represented as a sum of simple components. This is due to the global character of shape features. This leads to increased computational complexity of optimisation of the estimation function.

Instead of guessing about the necessity of applying additional shape restrictions and how they have to be formulated, because adding further shape restrictions over and over inevitably results in cumbersome algorithm, an alternative route have been followed here. This work is focused on the research of the detection potential of an algorithm which takes into account only the indication intensity z and requires connectivity of the detected curve. I.e. no shape features are considered.

3.5.5 Parameter estimation vs. hypothesis testing

From the informational standpoint the best to do on solving of the detection task while observing an indication of a certain estimated intensity z is to build a distribution of probabilities of presence of a real indication with intensity h: $\rho(h|z)$ [62]. This does not only apply to local estimations. The probability distribution of presence of an indication h along the full hypothetical path can be calculated too. Based on this distribution the most probable indication intensity can be found:

$$h^* = \arg\max_h \rho(h|z). \tag{3.38}$$

The most probable intensity along the path indication can be used as an estimation function, thus the search procedure which maximise this value will find the path through the image which corresponds to the most intensive indication.

Unfortunately, and this was already addressed in section 3.2 of this chapter, the calculation of $\rho(h|z)$ is not possible without making assumptions about the *a priori* distribution of h: $\rho(h)$. In a general case $\rho(h)$ is a function which has to be defined for all possible h. Therefore the *a posteriori* distribution $\rho(h|z)$ has to be calculated for all h too. This introduces an additional dimension into, already without that, big search space. Therefore a reduction to a simpler model is desirable from the computational standpoint.

Due to this, the following statistical model of the indication intensity is assumed: the indication of the known intensity h can be present with *a priori* probability $P(h)$ or absent with *a priori* probability $(1 - P(h))$. The values $P(h)$ and h must be externally provided to the algorithm.

If such a discrete model of the indication intensity is assumed, the calculation of the most probable intensity h^* is reduced to the calculation of the *a posteriori* probabilities of two possible cases: indication presence $P(h|z)$ and absence $P(0|z)$ with subsequent comparison of them. This can be formulated in terms of hypotheses testing: the hypothesis of indication presence $H_1 = H(h)$ is complemented by the hypothesis of indication absence $H_0 = H(0)$.

3.5.6 The proposed estimation function

Let us denote $P(H(h)) = P(H_1) = P_1$ and $P(H(0)) = P(H_0) = P_0$ - the probabilities of hypotheses of indication presence and absence respectively. The $P(H_1(p_1, p_2, .., p_n)) = P_1(p_1, p_2, .., p_n)$ and $P(H_0(p_1, p_2, .., p_n)) = P_0(p_1, p_2, .., p_n)$ are

the joint *a posteriori* probabilities of indication presence and absence in all points along a hypothetical path respectively.

The *a posteriori* probabilities of both hypotheses for a path of some length n ($n > 1$) can be expressed via local probabilities in each point of the path:

$$P_1(p_1, p_2, .., p_n) = P_1(p_1)P_1(p_2)..P(p_n) = \prod_{i=1}^{n} P_1(p_i), \tag{3.39}$$

$$P_0(p_1, p_2, .., p_n) = 1 - P_1(p_1, p_2, .., p_n), \tag{3.40}$$

where $P_1(p_i)$ is the *a posteriori* probability of indication presence. Expression (3.5) can be used for calculation of $P_1(p_i)$ by substituting of the probability densities of the continuous distribution of by the probabilities of different states of the discrete distribution:

$$\begin{gathered} P_1(p_i) = P(H_1(p_i)) = \frac{P(h)\rho(z(p_i)|h)}{\rho(z(p_i))}, \\ \rho(z(p_i)) = P(h)\rho(z(p_i)|h) + (1 - P(h))\rho(z(p_i)|0). \end{gathered} \tag{3.41}$$

Here, an assumption is made about statistical independence of adjacent pixels and the join probability searched for is just a product of the independent local estimations. A more promising solution would be to assume a dependence of the *a priori* probability of crack presence in each next point of the path from the *a posteriori* probability of crack presence in the previous point of the path (see below).

Expression (3.39) could be used already as the estimation function, but for the sake of easy optimisation the logarithm of $P_1(p_1, p_2, .., p_n)$ is preferred: $f^*(p_1, p_2, .., p_n) = \log P_1(p_1, p_2, .., p_n)$. The logarithm converts the product in the right part of (3.39) to a sum:

$$f^*(p_1, p_2, .., p_n) = \sum_{i=1}^{n} \log P_1(p_i). \tag{3.42}$$

In this sum each term depends only on one variable p_i. So it can be easily optimised.

Nevertheless, the estimation function f^* calculated according (3.42) has one big disadvantage: the joint probability of indication presence $P_1(p_1, p_2, .., p_n)$ monotonically decreases with increased n. The reason is obvious: the probabilities under the product sign are always less than unity. This means that the joint probability of indication presence can be smaller for a path which coincides with an intensive and long indication, than for a path which coincides with a less intensive but short indication. Such behaviour does definitely not meet our requirements. A weighting of f^* by the path length is not possible without substantial increase in the computational complexity of the optimisation. This is similar to the average contrast over the indication length: the representation as a sum of independent terms is not possible because the unknown indication length is present in each term.

An elegant solution can be found if not only the *a posteriori* probability $P_1(p_i)$ is used in each point of the path, but the *a posteriori* probability is compared with the *a priori* probability of indication presence $P_{1\ prior}$. This can be elegantly expressed in terms of entropy and *a posteriori* information gain:

$$dI_1(p_i) = \log P_1(p_i) - \log P_{1\ prior} = \log \frac{P_1(p_i)}{P_{1\ prior}}. \tag{3.43}$$

Thus the cumulative information gain along the path is used in this work as an estimation of path "goodness":

$$f^*(p_1, p_2, .., p_n) = \sum_{i=1}^{n} dI_1(p_i) = \sum_{i=1}^{n} (\log P_1(p_i) - \log P_{1\ prior}). \tag{3.44}$$

The *a posteriori* probability in each point is calculated as before using (3.41) and $P_{1\ prior} = P(h)$. Then (3.43) can be written as follows:

$$dI_1(p_i) = \log P(h|z(p_i)) - \log P(h). \tag{3.45}$$

And (3.44) converts to:

$$\begin{gathered} f^*(p_1, p_2, .., p_n) = \sum_{i=1}^{n} [\log P(h|z(p_i)) - \log P(h)] = \\ \sum_{i=1}^{n} [\log \rho(z(p_i)|h) - \log [(P(h)\rho(z(p_i)|h) + (1 - P(h))\rho(z(p_i)|0))]]. \end{gathered} \tag{3.46}$$

This is the path estimation function which is used in the developed algorithm.

Model for chained indications

The assumption about statistical independence of neighbour pixels is made above. A more promising solution is to assume a dependence of the *a priori* probability of crack presence in each following point of the path from the *a posteriori* probability of crack presence in the previous point of the path. For example:

$$P_{1\ prior}(p_i) = P_{trans} P_1(p_{i-1}), \tag{3.47}$$

where i is step number, $P_{1\ prior}(p_i)$ is the *a priori* probability of indication presence for point p_i, P_{trans} is a probability of indication prolongation, and $P_1(p_{i-1})$ without additional index is the *a posteriori* probability of indication presence in the previous point.

As the *a priori probability* for each point depends (according to (3.47)) on the *a posteriori* probability in the previous point, the joint probability $P_1(p_1, p_2, .., p_n)$ can be calculated only recursively:

$$\begin{aligned} P_{1\ prior}(p_1) &= P_{1\ prior}, \\ P_1^*(p_i) &= P_1(p_i; P_{trans} P_1^*(p_{i-1})), \\ P_1(p_1, p_2, .., p_n) &= \prod_{i=1}^{n} P_1^*(p_i). \end{aligned} \tag{3.48}$$

Then (3.44) converts to:

$$f_{ch}^*(p_1, p_2, .., p_n) = \sum_{i=1}^{n} (\log P_1(p_i; P_{trans} P_1^*(p_{i-1})) - \log P_{1\ prior}). \tag{3.49}$$

Each term of this sum depends now on two variables: p_i and $P_1^*(p_{i-1})$. The strict optimisation of $f_{ch}^*(p_1, p_2, .., p_n)$ by means of dynamic programming is not longer possible, because values of $P_1^*(p_{i-1})$ are not discrete.

An approximated solution could be possible if some heuristics are introduced. For example, quantisation of continuous range of values of $P_1^*(p_{i-1})$ into a fixed number of discrete values. But since one of the ideas of this work is to explore possibilities of object detection without usage of any heuristics, this approach is not of our primary interest.

3.5.7 Analysis of the estimation function

The designed estimation function (3.46) is characterised by the following properties:

- It has a theoretical background and a physical meaning.
- The value of the estimation function can increase or decrease depending on the relation of intensity of the sought object h and the estimated contrast z.
- The amount of increase or decrease of the estimation function does not only depend on the difference between h and z but also on the estimated noise level and the *a priori* assumed probability of presence of the sought object.
- It can be represented as a sum of simple terms, so it can be easy optimised.

For calculation of f^* the values h and $P(h)$ (and P_{trans} if used) must be supplied externally. They act as the algorithm's control parameters. In that follows we analyse their influence on the behaviour of the estimation function.

Object intensity h

The object intensity h and the *a priori* probability of its occurrence define the statistical model of the objects searched for. As it is evident from (3.21) the value of h defines the likelihood of which hypothesis (H_1 or H_0) becomes bigger for the given indication estimation z. If $z > h/2$ then $L_z(h) > L_z(0)$ ($\rho(z|h) > \rho(z|0)$). And vice versa: if $z < h/2$ then $L_z(h) < L_z(0)$. So the expected object intensity h acts as a primary sensitivity control.

A priori probability of hypothesis of presence ($P(h)$)

An analysis of (3.45) can reveal an unexpected dependency, for the first look, between the information gain dI_1 and the *a priori* probability of indication presence. Namely: increasing of $P(h)$ leads to the decreasing of dI_1. This behaviour is a property of Shenon's definition of information and is already discussed in [62]: the big *a priori* knowledge cannot be significantly improved, thus leading to small dI_1, but a small *a priori* knowledge can, thus allowing big dI_1.

Fig. 3.7 demonstrates the dependence of the *a posteriori* information gain $dI_1(p_i)$ versus signal estimation z/σ for hypothesis $h = 3\sigma$ and different *a priori* probabilities $P(h)$. The same estimated signal intensity z influences the information gain dI_1 differently depending on the $P(h)$:

1. Low $P(h)$ ($P(h) = 0.01$): The information gain is nearly proportional to the estimated indication intensity z in the range $0..h$ ($h = 3\sigma$ for the example) and

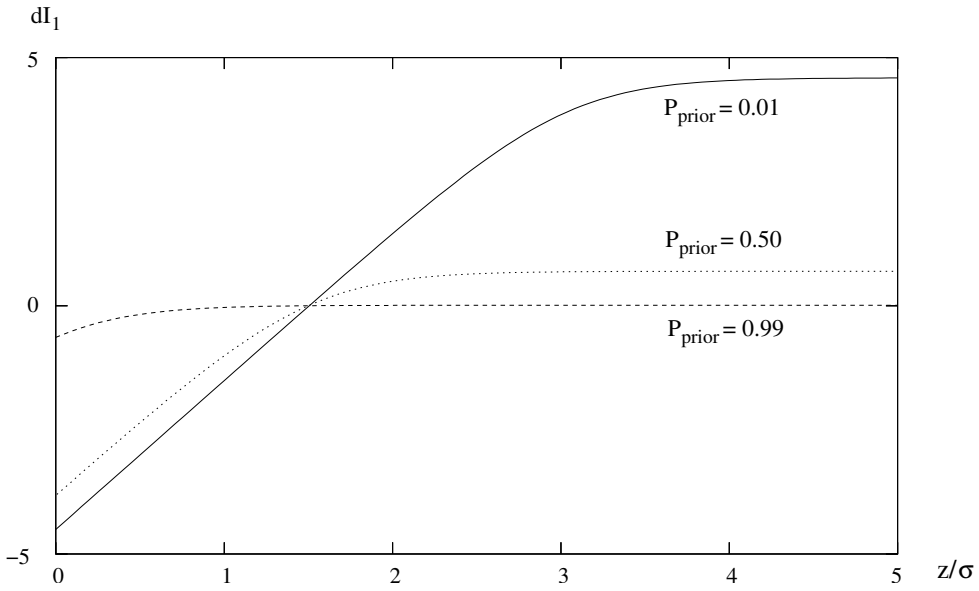


Figure 3.7: *A posteriori* information gain $dI_1(p_i)$ vs. signal estimation z/σ for hypothesis $h = 3\sigma$ and different *a priori* probabilities $P(h)$. dI_1 is negative for $z < h/2$ and positive for $z > h/2$. The information gain dI_1 is nearly proportional to z for small $P(h)$ and substantially nonlinear for bigger $P(h)$.

stabilise swiftly to a constant value for bigger z. This example demonstrates an important fact: an increase in the signal to noise ratio adds a value to the estimation function only until some SNR is reached - a further increase in SNR brings little additional information. It is a desirable and logical behaviour.

2. Middle $P(h)$ ($P(h) = 0.5$): The information gain stabilises even faster for $z > h/2$, but a weak observed signal ($z < h/2$) still adds negative value to the estimation function as it is expected from the common sense. The plot of dI vs. z/σ is sloper, compared to the case of $P(h) = 0.01$, which shows decrease of sensitivity.

3. High $P(h)$ ($P(h) = 0.99$): Information gain is close to zero in range $z \in (\sigma, \inf)$ (slightly higher than zero for $z > h/2$ and slightly lower than zero for $z < h/2$) due to the *a priori* knowledge of the object presence – the *a priori* information is too strong ($P(h) \to 1$) to be influenced by the observed signal. Only a very week z ($\ll \sigma$) is able to change this *a priori* assumption and cause substantially negative dI_1. The sensitivity is the lowest.

This asymmetric behaviour of dI_1 (and the path estimation function f^*) ensures the following properties of the developed algorithm:

1. the *a priori* probability of indication presence $P(h)$ is used to control whether an indication with gaps (areas with low z) will be detected as a single continuous indication or as separate shorter ones;

2. very intensive parts of indication (high z) do not contribute excessively to the path estimation function f^* (while dI_1 stabilises fast after z exceeds $h/2$).

The interesting case arise if $P(h) \to 0$. In this case an usage of probabilities becomes difficult because the *a posteriori* probability approaches zero too:

$$\lim_{P(h)\to 0} P(h|z) = \lim_{P(h)\to 0} \frac{P(h)\rho(z|h)}{P(h)\rho(z|h) + (1-P(h))\rho(z|0)} \to 0. \tag{3.50}$$

Nevertheless the information gain dI_1 remains finite:

$$\begin{aligned}
&\lim_{P(h)\to 0,\ P_i(h)\to 0} dI_1(p_i) = \\
&\lim_{P(h)\to 0,\ P_i(h)\to 0} \left(\log \frac{P_i(h)\rho(z(p_i)|h)}{P_i(h)\rho(z(p_i)|h) + (1-P_i(h))\rho(z(p_i)|0)} - \log P(h) \right) = \\
&\log \frac{\rho(z(p_i)|h)}{\rho(z(p_i)|0)} = ... = \frac{z^2(p_i) - (h - z(p_i))^2}{2\sigma^2} = \frac{h^2}{2\sigma^2} \left(\frac{2z(p_i)}{h} - 1 \right).
\end{aligned} \tag{3.51}$$

So the information gain in this case reduces to the comparison of likelihoods of both hypothesis and then to comparison of intensities $z(p_i)$ and h.

Probability of indication prolongation (P_{trans}) (if the chained indication model is used)

According to (3.47) P_{trans} (probability of indication prolongation) defines the influence of the *a posteriori* probability in some point of the path on the *a priori* probability for adjacent points. This way a large value of P_{trans} results in fast increasing of $dI_1(p_i)$ if an indication is met and slow decreasing of $dI_1(p_i)$ if the indication disappear. So the separated indication segments can be connected into a continuous line and the small gaps between separated indications tend to be ignored. On the opposite, a small value of P_{trans} results in slow increasing of dI_1 if some indication is found and fast decreasing of dI_1 if the indication is lost. This leads to an increased sensitivity to interruptions of the crack indication, so small gaps can cause the detection of two or more separate indications rather than a single continuous one.

A special case is $P_{trans} \leq P(h)$. Then $P_i(h) = P_{trans}P(h|z(p_{i-1})) < P(h)$ (because $P(h|z)$ is always less than unity) - the starting *a priori* probability $P(h)$ becomes bigger than the running *a priori* probability $P_i(h)$ caused by the analysis of neighbour pixels. So the pixels are considered as they are independent.

In case of the chained indication model and $P_{trans} \gg P(h)$ the probability of indication prolongation P_{trans} has more influence on the continuity of the detected curve rather than $P(h)$. The *a priori* probability of indication presence influences in this case more the sensitivity of the algorithm rather than the continuity of the detected curve.

Nevertheless, the considerations about influence of P_{trans} on the behaviour of the path estimation function remain unused in this work since the chained indication model is not used for the developed algorithm.

3.6 Implementation of object detection via optimisation of estimation function

3.6.1 ROI tracing vs. full search

As intended, optimisation of the estimation function described above can be achieved serially. The optimisation starts from some initial hypothesis about the presence of an indication at a given point. Then hypotheses about prolongations of the indication are made and the estimation functions for each of these hypothesis are calculated. There are at least two approaches to the selection of the starting point.

An obvious approach is to start a new search at the point with the biggest probability of object presence – at the so called region of interest (ROI). Normally there is a possibility that more than one indication or some other objects like lead marks or quality indicators may be present in the image. Therefore it is necessary to select not a single point but a set of starting points which have a high probability of indication presence. Then, all possible indications which originate from them (*paths*) have to be investigated. This approach works as long as the objects to be detected have high signal to noise ratios. But if the sought signal is heavily occluded with noise, such distinctly visible points cannot be found or do not correspond to the objects of interest any more. Then this approach fails.

Another possibility (which is used here) is to explore all possible paths. I.e. to assume the beginning of a path in every image point and test prolongations. If it is done in an ordered way, for example from left to right or from top to bottom, then the computational expenses of such approach are not significantly higher than in the case of ROI-tracing. This is due to the fact that several search trees, growing not very far from each other, start to overlap very fast. In case of starting of many trees from neighbour points the trees start to overlap after the first step and the computational expenses are minor.

3.6.2 Basic search algorithm

First, the local operator designed in this chapter is applied to the image. Then, an array S is created which has the same dimensions as the source image and stores optimisation information. Each element of this array is a data structure and corresponds to the image point with the same coordinates. This data structure stores information about the best (so far found) path which reaches the associated point:

- the value of the estimation function f^* for the path;
- the length ln of the path;
- the coordinates of the previous point p_{prev} in the path.

Initially the fields of these structures storing path length are initialised by zero and the estimation function fields – by minus infinity.

The goal of the search is to find the maximum of the estimation function. So the maximum value of the estimation function f^*_{max} reached from the beginning of search and the coordinates of the respective point p_{max} are stored in two additional variables. The f^*_{max} is initially initialised by minus infinity.

If horizontal indications have to be detected the search starts from the right or the left image side (arbitrary which) and paths which grow in the direction of the opposite image side are investigated. If vertical objects are of interest, then search can be done from top to bottom or bottom to top.

For the sake of simplicity of description but without any lose of generality we assume that the horizontal indications are of interest and perform the search in the direction of increasing of the x coordinate. Then for every image column x, starting from the first one, and for every pixel (x, y) in this column, chosen in the arbitrary order:

1. Calculate the *a posteriori* probability of indication presence $P_1(x, y) = P(h|z(x, y))$ using the output of the local operator and (3.5). Test only horizontal orientation of indications.

2. Calculate the respective information gain using (3.45).

3. If the calculated information gain dI_1 exceeds the value stored in the f^*-field of the respective element of S, then update S with the calculated dI_1 and set ln to one. Otherwise, discard new dI_1.

4. Independently of the result of the previous step compare f^*_{max} with the value of f^* for the current point stored in S. If the current f^* exceeds f^*_{max}, then update $f^*_{max} = f^*(x, y)$ and $p_{max} = (x, y)$.

5. Build and test all prolongations. Because the indication path has to be continuous there are only three possible prolongations: to point $(x+1, y)$, $(x+1, y+1)$ and $(x + 1, y - 1)$.

   (a) Calculate the *a posteriori* probability and the information gain $dI(x + 1, y_{prol})$ in the prolongation point $(x+1, y_{prol})$. Use the direction from the current point to the prolongation point as the direction of local indication for the local operator.

   (b) Compare $f^*(x, y) + dI(x + 1, y_{prol})$ (the sum of f^* for the current point and the information gain in the prolongation point) with $f^*(x + 1, y_{prol})$ (the value of f^* for the prolongation point stored in S). If $f^*(x, y) + dI(x + 1, y_{prol}) > f^*(x + 1, y_{prol})$, then update stored $f^*(x + 1, y_{prol})$ to $f^*(x, y)+dI(x+1, y_{prol})$, $ln(x+1, y_{prol})$ to $ln(x, y)+1$ and $p_{prev}(x+1, y_{prol})$ to (x, y).

After all points from the current column are processed, the same algorithm is applied to the next column. This continues until the last column is reached. Then, f^*_{max} is analysed and if it is positive, the best path is tracked back from point p_{max} using p_{prev} stored in each element of S.

The path tracked back from the point p_{max} represents the indication which improve the probability of indication presence from the initial *a priori* level to the *a posteriori* level most significantly (in limits of the assumed crack model and for the used parameters w_{ind}, h and $P(h)$). This is because the estimation function f^* is defined as the sum of information gains at points along the traced path and the maximum of f^* is found by the algorithm described above.

The path tracked back from the point p_{max} is the detection result at first approximation. The basic search algorithm described above is represented in its simplest form to keep workflow clear. Some useful refinements can be made. These are described in that follows.

3.6.3 Minimal significant length

Fluctuations of local image intensity caused by the added noise may randomly generate conglomerates which look similar to the true crack indications. These noise formations can have a rather big intensity and this becomes a problem when detection of objects with low SNRs is of interest. However the noise formations can be distinguished from the true crack-like indications by their short length. This is because the probability of random formation of a crack-looking artifact decreases drastically with increasing length of the artifact.

Therefore the limitation of a minimal significant indication length appears to be useful. So the path with a length less than the minimal significant length should not be considered as a possible result (but continue to grow). According to this, step 4 of the basic optimisation algorithm (as described above) has to be modified as follows: "... compare f^*_{max} with the value of f^* for the current point stored in S. If the current f^* exceeds f^*_{max} *and the length of the current path exceeds the minimal significant length*, then update $f^*_{max} = f^*(x, y)$ and $p_{max} = (x, y)$".

3.6.4 Multiresolution

As it is mentioned already before, the local operator, designed in Chapter 3.3, which uses a mask size $w_{ind} \times w_{ind}$, optimally responds to indications of width w_{ind}. Unfortunately it is not possible to predict the width of the object to be detected. Moreover the width of indication usually changes along the indication length due to variations of crack orientation in a 3D space.

Hence, it is necessary to specify *externally* the range of the useful widths which are of interest and can be imaged by the used acquisition technique. All indication widths which fit in this range from w_{min} to w_{max} have to be tested. As the window size for the estimation of indication grey value defines the spatial resolution of the operator, we call the technique which uses different window sizes – *multiresolution*.

If multiresolution is used, then each element in the indication path is defined not only by pixel coordinates but also by the used window size. So the variable window size adds an additional dimension to the search. Fortunately the range of useful indication widths is not too big, so the optimisation with the increased complexity is still feasible.

The multiresolution-implementation requires the following changes in the basic search algorithm:

- To the array S a new dimension is added so hypothesis can be made about the presence of indications of different width.

- Each element of the array S is extended by a new variable w_{prev} which stores the width of the previous element in the "best yet found" path. This variable

has to be updated accordingly each time when coordinates p_{prev} of previous element are changes.

- The step 1 of the basic search algorithm is modified as follows: "... Test only horizontal orientation of indications *in all resolutions from range* w_{min} *to* w_{max}".

- The step 5 of the basic search algorithm is modified as follows: "Build and test all prolongations. Because the indication path has to be continuous there are three possible prolongations *for each resolution*: to point $(x+1, y)$, $(x+1, y+1)$ and $(x+1, y-1)$. *The prolongations in different resolutions (the current* w*, increased* $w-1$ *and decreased* $w+1$ *ones) have to be tested.* $w-1$ *must not sink below* w_{min} *and* $w+1$ *must not exceed* w_{max}". So the overall number of prolongations now totals to nine at biggest.

3.6.5 Multipass

The result of the basic search algorithm (including the check for the minimal significant length and the multiresolution extension) is a set of connected image points (path), on which the estimation function takes its maximum (for the given image, the chosen search direction, image and path prolongation model, model parameters). The result of the search algorithm depends on the chosen direction of path growing: left-to-right/right-to-left or top-to-bottom/bottom-to-top. So generally a second iteration of the algorithm is necessary to explore paths growing in the opposite direction. For this purpose the array S and variable f^*_{max} are initialised by their start values again and the same search procedure is performed from the opposite image side in the opposite direction. Finally both results (from both opposite directions) are superimposed (added).

Furthermore: normally not only the best path is of interest, but all paths have to be found which improve the *a priori* probability of indication presence. So after the first pass is completed, the path has to be visualised and the points belonging to it are marked as "used". Then the next pass can be executed in which the marked points will be not used neither for beginning nor for the prolongations of the hypothesis. The second pass will end up with a smaller f^*_{max}. The result obtained is superimposed on the result of the previous pass. The iterations have to be continued until f^*_{max} decrease below the predefined limit (in this implementation it is "0"). Then the algorithm terminates and the visualisation of the found paths is returned to user.

3.6.6 Noise level estimation

The noise model is considered already during design of the local operator because an assumption of a certain noise distribution is necessary for a correct estimation of the likelihood of indication presence in local area. Here it is described how the parameter σ of this noise model is estimated.

As long as a small local area of the radiographic image is considered, the assumed linear background model appears to be adequate. The mean square variation of noise fluctuations σ in a local neighbourhood of a point (x, y) can be directly calculated if

the background grey value estimation is available:

$$\sigma(x,y) = \frac{1}{N-1} \sum_{(i,j)\in D(x,y)} (g(i,j) - g_{bkg}(i,j))^2, \tag{3.52}$$

where $D(x,y)$ denotes a local area centred around point (x,y), N is number of pixels in the local area D, $g(i,j)$ is the image grey value and $g_{bkg}(i,j)$ is the background grey value estimation for point (i,j).

Estimations of σ calculated as shown above are stored in an array which has the same dimensions as the source image. Some of these local estimation may be too high if not only the background is present in the local area D. I.e. the presence of a crack-like indications influences the noise estimation substantially. In order to suppress this, the array of estimations of σ can be filtered using median or averaging filter (the averaging is used here) which uses a relatively big square window. Here the smoothing window size is chosen equal to the *minimal significant indication length* (which is an externally supplied parameter). Storing of the noise estimations in the array associated with the source image provides *context sensitive* noise estimations (in opposite to a single σ for the whole image). As far as $\sigma(x,y)$ is known the value of $\sigma_z(x,y)$ which is a function of w_{ind} and w_{bkg} can be calculated using (3.19).

3.6.7 Algorithm control parameters

As it is already mentioned during the description of the algorithm, several control parameters have to be provided externally. They are necessary to specify parameters of the used models and hence the objects which have to be detected. The choice of correct parameters is essential and must be done by the human being (operator) according to requirements of the particular detection task and on the basis of operator's experience. The full list of operator controlled parameters is as follows:

1. *a priori* object SNR h/σ_z in hypothesis of presence controls algorithm sensitivity;
2. *a priori* probability of hypothesis of presence $P(h)$ influences detection of ragged indications (indications with gaps);
3. minimal and maximal width of indication (size in the direction orthogonal to crack orientation) w_{min} and and w_{max} define transversal spatial resolution;
4. minimal length of indication L_{sign} defines longitudinal resolution.

3.7 Estimation of memory requirements and computational complexity

3.7.1 Memory requirements

The array S, which stores optimisation information (information about the best path (found so far) reaching the associated point), has dimensions $(w_{max} - w_{min} + 1) \times X \times Y$, where X and Y are the image dimensions and w_{max} and w_{min} define the range of usable indication widths. Each element of S stores (for images which are smaller than 2^{16} points in each dimension):

- the value of the estimation function f^* as a floating point number,
- the length of path ln as two-byte integer number,
- the coordinates of the previous point of path p_{prev} as a pair of two-byte integers,
- the resolution (indication width) in the previous point of path w_{prev} as a one-byte integer.

As long as a single precision floating point number fits in four bytes, each element of S occupies 11 bytes. Therefore the whole S occupies:

$$\mathrm{Mem}_S = 11\mathrm{bytes} \times (w_{max} - w_{min} + 1) \times X \times Y. \tag{3.53}$$

The local operator, which estimates $L_z(h) = \rho(z|h)$, needs to store the respective grey value estimation (an average calculated on square window) for each image point and indication width. The grey value estimation can be stored as a 16-bit (two-bytes) integer, so:

$$\mathrm{Mem}_L = 2\mathrm{bytes} \times (w_{max} - w_{min} + 1) \times X \times Y. \tag{3.54}$$

It is convenient to pre-calculate the noise estimations and store the result in the same way as the grey value estimations (for each image pixel and indication width). However, noise estimations are stored as floating point values:

$$\mathrm{Mem}_\sigma = 4\mathrm{bytes} \times (w_{max} - w_{min} + 1) \times X \times Y. \tag{3.55}$$

The array which is used for multipass procedure and which contains labelling what a point is already included in a detected indication, needs one byte per source image pixel. Visualisation of the search results is made in the one byte per pixel image. Therefore the approximate memory requirements of the presented implementation of the search algorithm are

$$\begin{aligned}\mathrm{Mem} = 2\mathrm{bytes} \times X \times Y + \mathrm{Mem}_\sigma + \mathrm{Mem}_L + \mathrm{Mem}_S = \\ = (2 + 17 \times (w_{max} - w_{min} + 1)) \times X \times Y\mathrm{bytes}\end{aligned} \tag{3.56}$$

– proportional to the source image size and nearly proportional to the range of indication widths. This results, for example, in about 350 MByte of memory for a 4 MPixel image and a range of indication widths from 2 to 6 and in about 1.7 GByte for a 20 MPixel image and the same widths range. The 20 MPixel image size is, therefore, nearly a limitation for computing architectures which use 32 bits address space (all Intel Pentium CPUs, for example). Bigger images require bigger address space.

3.7.2 Computational complexity

The execution time of the developed algorithm depends largely on the programming language used, compiler and computing hardware. An analytical estimation of computing time is therefore complicated and beyond the scope of this work, but measurements of the execution time of the existing implementation can be easily made. For example, the current implementation, compiled with GCC 3.3, needs on a 3 GHz Intel Pentium-4 CPU about 80 seconds for detection of one the most intensive indication on a 2.5 MPixel image (widths from 2 to 6 pixels). However detection of

all suspicious crack-like indications on the same image can take up to one hour depending on the selected algorithm sensitivity, SNR and number of indications really present in the image. An analysis of the computational complexity of the algorithm can explain these differences of execution time.

The computational complexity can be estimated as follows. Important that optimisation is performed serially from one image side to another. At the moment when a path has to be prolongated from a given point at a given resolution (indication width), all possible paths to this point are already built and the optimal path to this point is found. This means that the prolongation of the optimal path from every image point at every resolution has to be done only once per algorithm execution (pass).

For the assumed prolongation model there are only nine prolongations of the optimal path possible: the three adjacent pixels at three resolutions each. In addition, a hypothesis has to be tested about the beginning of a path from each point. Altogether this gives 10 calculations of estimation function per image pixel and per indication width (resolution). Two logarithms have to be taken for the calculation of the proposed estimation function (see (3.45),(3.5),(3.30)) as well as several floating point multiplications and additions.

An important consequence from these considerations is the independence of the the computational complexity of one pass of the algorithm from the image contents. It depends only (and linearly) on the image size and the range of the indication widths. As the detection of a fixed number of the most intensive crack indications requires exactly the doubled number of algorithm executions, its computational complexity does not depend on the image contents too. Thus the computing time is predictable *a priori*.

However, if *all* crack indications (paths) have to be found for which the estimation function exceeds a certain given threshold, then the computational complexity becomes dependent on the image contents as it is not possible to estimate in advance the number of such indications and therefore the number of necessary algorithm executions.

Chapter 4

Experimental evaluation

4.1 Algorithm implementation

For experimental evaluation of the developed algorithm, it was implemented in standard C++ programming language. The program can be compiled on most UNIX-like systems as well as in the MS-Windows Win32 environment. It is a command line tool and can be executed directly from the command shell or used as computational engine by other applications (for example graphic user interfaces).

The source code is listed in Appendix A. The code is distributed over several files correspondent to the functionality:

- local.h and local.cpp - interface and implementation (respectively) of the estimation of local indication contrast z;
- combine.h and combine.cpp - noise level estimation, calculation of $L_z(h)$ for each image point, global optimisation of the estimation function f using dynamic programming, multipass and multiresolution enhancements, visualisation of result;
- conf.h and conf.cpp - a class which performs parsing of command line and takes care of storage and correctness of the algorithm control parameters;
- cognition.cpp - binds the parts of the code described above into an end-user program. The executable is called cognition.

4.2 ROC as a performance criteria

Each execution of the cognition program results in a image which has the same dimensions as a source image and shows locations of the found crack indications. Influences of the control parameters on the quality of detections can be estimated from this result images visually, i.e subjectively and qualitatively.

More interesting and useful would be to express the detection quality objectively and quantitatively. This can be done by the detection of known crack indications.

Using the truth about the crack positions, the following detection cases can be distinguished: true positive, true negative as well as false positive and false negative detections. The result is conveniently expressed graphically by the "Receiver Operating Characteristic" (ROC). The applicability of ROC to describe the performance of a NDT weld inspection system is discussed in [42].

The ROC represents the dependence of the rate of correct detections vs. the rate of false detections. For the ROC analysis of the detection algorithm the source image has to be partitioned (virtually) into some number of measurement areas for which a decision about the presence of a defect must be made. The correct detection rate is defined as the ratio between the number of areas correctly labelled as defective $N_{detected\ def}$ to the number of truly defective areas $N_{true\ def}$ on the image:

$$CD = \frac{N_{detected\ def}}{N_{true\ def}}. \tag{4.1}$$

The false detection rate (false alarm rate) is defined as the ratio between the number of false detections $N_{false\ detections}$ to the number of truly defect-free areas ($N_{all} - N_{true\ def}$) (where N_{all} is whole number of areas on the image):

$$FA = \frac{N_{false\ detections}}{(N_{all} - N_{true\ def})}. \tag{4.2}$$

The ROC representation allows to describe a system at different sensitivity levels. Two or more different systems or the behaviour of the same system for different values of control parameters can be compared by using ROCs. Principally a ROC is shown on *Fig. 4.1*. The strait line on this plot which runs straight from $(0, 0)$ to $(1, 1)$ describes a system which made decisions just randomly, i.e. without usage of the source data at all. In this case the rate of correct detections will be statistically equal to the rate of false alarms. A detection system which wrongly interpret the source data can have a characteristic which lies beneath the random decisions line. On opposite, a good system which is able to extract useful information from the source data and makes correct decisions, will have a ROC which lies above the random decision line. The better the detection system is, the further away from the random decision line its ROC will be positioned. This automatically means that for the same rate of false alarms the better system has a higher rate of correct detections as compared to the worse system or a smaller rate of false alarms can be reached for the same rate of correct detections (see *Fig. 4.1*). The perfect system (not shown on illustration here) would detect all defects without making any false detections, i.e. would give $CD = 1$ for $FA = 0$.

4.3 Use of synthetic images for algorithm testing

In order to build a ROC of the developed algorithm, the complete information must be available about the positions of true indications in the source image. This can be achieved, for example, by synthetic (simulated) images. The advantage of using simulated images is the ability to control all image properties and to observe how they effect the detection. This allows to study the abilities of the algorithm in a quantitative form. For the generation of such synthetic images a simulation software was developed.

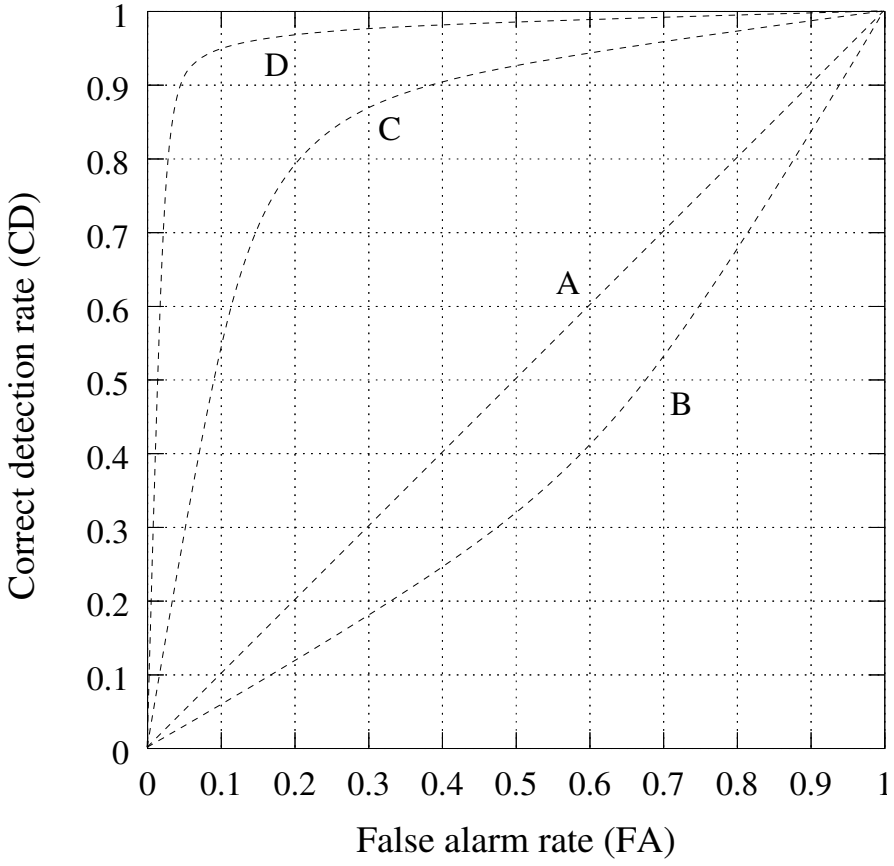


Figure 4.1: Some examples of receiver operating characteristics (ROCs): A - random decision system; B - wrong decision system; C - correctly operating system of moderate efficiency; D - highly efficient system.

4.3.1 Synthetic image generation

Synthetic images of crack indications which visually look very similar to the real radiographs can be simulated, if the influencing properties of the radiographic system are taken into account. These properties are:

- geometrical unsharpness due to the focal size of the X-ray tube;
- presence of noise of high intensity (additive, zero-mean, normally distributed, uncorrelated);
- imperfect MTF of image detector (detector point spread function).

In addition to the properties of the radiographic system, the object grey-level and spatial models have to be included. Taking all these factors into account the generation of synthetic images can be done in the following steps:

1. According to the assumed background model and the the spatial indication model a synthetic indication is simulated as a sharp continuous curve (randomly directed) laying on a linearly changing background (*Fig. 4.2.a*). Sharp turns in direction are not allowed. The difference between indication grey level and background grey level (contrast) as well as the curve width are known, fixed and constant.

2. The image is smoothed in order to simulate the geometrical unsharpness of radiographic system (*Fig. 4.2.b*). To achieve a visually similar effect the size of the smoothing kernel must significantly exceed the indication width. Due to the smoothing the indication contrast on the image will be reduced. This reduced contrast is referred as the true indication intensity h_s.

3. Normally distributed noise (spatially independent, zero mean) is added to the image (*Fig. 4.2.c*). The noise intensity (which is given by the root mean square deviation σ_s) is calculated according to the required signal to noise ratio SNR:

$$\sigma_s = \frac{h_s}{SNR} \tag{4.3}$$

4. Finally the detector unsharpness is simulated. For this purpose the image is convolved with the point spread function of the detector. Resulting image is shown on *Fig. 4.2.d*. The same image with a stretched histogram is presented on *Fig. 4.2.e* for demonstration purpose.

For simulation of realistic images it was essential that the two convolutions (for simulation of the geometric unsharpness and the detector unsharpness) have to be done separately. One before and another after the noise addition. This reflects the physics of image formation correctly.

Using the scheme described above a simulated radiographic image can be created. However, the simulation of the detector unsharpness (step 4) slightly changes the intended signal to noise ratio SNR of the crack indication. This is due to the unequal influence of the smoothing on the indication and the noise intensities. So the signal to noise ratio of the final image will be slightly different from the intended value. The real signal to noise ratio can be precisely measured as follows. Two additional images have to be generated. In generation of the first one the Step 3 is omitted,

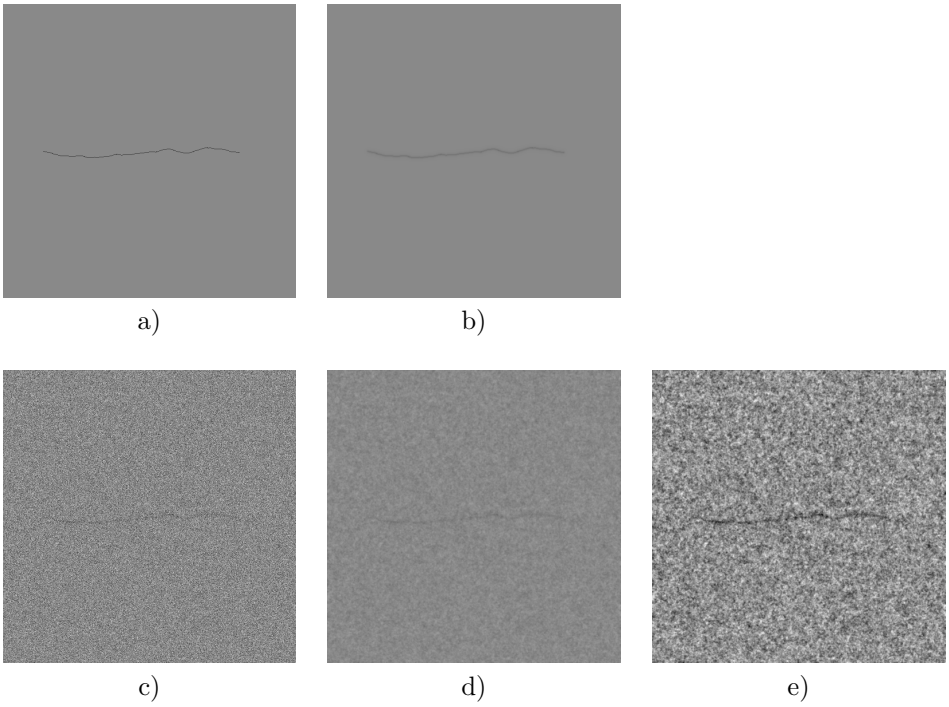


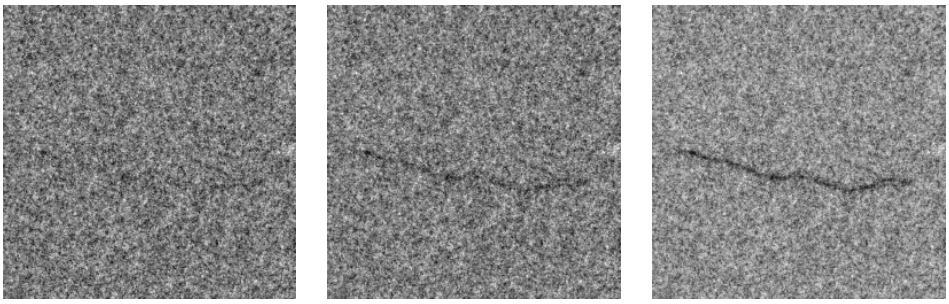
Figure 4.2: Generation of synthetic images with a crack indication. Steps of simulation: a) indication shape only; b) geometrical unsharpness is added; b) noise is added; c) detector unsharpness is added; d) contrast is increased for better representation.

i.e. an image without noise is generated. Then indication intensity can be measured just by examining of the indication profile (no disturbing noise). The second image is generated omitting Step 1, i.e. an image without the indication itself but with the added noise. While the background before addition of noise is known and it is not influenced by the detector unsharpness, the resulting standard deviation (intensity) of the noise can be easily found. In this way the real SNR of the simulated image can be measured.

4.3.2 Analysis of algorithm performance on synthetic images

The algorithm of synthetic image generation, described above, was used to create several sets of synthetic images. Each of these sets contains one hundred images. Images inside one set show different indication shapes, but images with the same number between different sets show the same indication. The same noise pattern was used for images with the same number. The difference between these sets is the different signal to noise ratio. For the performance analysis of the developed crack detection algorithm three sets were generated with SNR = 0.86, 1.7 and 3.4 (measured). On *Fig. 4.3* images with the same number (i.e. same crack indication) but belonging to the different sets (i.e. different noise intensity) are shown.

These sets were processed by the developed detection algorithm with different



a) b) c)

Figure 4.3: The same artificial crack indication which is simulated with different intended signal to noise ratios: a) SNR = 0.86; b) SNR = 1.7; c) SNR = 3.4

combination of control parameters. Each combination of control parameters results in a set of hundred result images which contain results of detection.

As the true positions of the simulated indications are know in every set of source images, the correct detections (CD) and false alarms (FA) rates can be found for every set of result images using (4.1) and (4.2). An additional utility program was written for this purpose. The utility takes two set of images as input (images of true indications and results of detection) and outputs the CD and FA rates for a certain partitioning into measurement areas. These values (CD and FA) define one point on ROC which describes performance of the algorithm for the used combination of control parameters and source signal to noise ratio.

A change of one or more control parameters of the detection algorithm results in an other set of results. Corresponding CD and FA can be calculated and plotted on ROC as a next point. In a such way the influence of each control parameter on the performance of the detection algorithm can be investigated.

Fig. 4.4 shows an example of detection results obtained at different values of expected indication intensity h for one source image with SNR = 1.7 (measured). *Fig. 4.5* shows the ROC of the detection algorithm for the set of one hundred source images with SNR = 1.7 and varying expected indication intensity h. The CD and FA rates were calculated for 3 pixel high and 150 pixel wide measurement areas. Other algorithm control parameters remain constant for this experiment and were: min and max indication width (w_{min} and w_{max}) - 1 and 3 pixels, *a priori* probability of indication present $P(h)$ - 0.1, significant indication length L_{sign} - 128 pixels. It can be seen from this plot that with the expected indication intensity $h = 2.3\sigma$ the algorithm is able to correctly find more than 95% of defect indications while the false alarm rate stays lower than 0.6%.

If the source image set has SNR = 3.4, the detection quality improves even more (*Fig. 4.6*) and $CD > 98\%$ can be obtained with $FA < 0.2\%$. This result can be obtained in range of the expected indication intensities $h = 2.4 \ldots 2.8\sigma$ with $w_{min} = 1$, $w_{max} = 3$, $P(h) = 0.1$, $L_{sign} = 128$.

If, on opposite, the source image set has SNR = 0.86 (*Fig. 4.7*), the algorithm performance characteristics decrease rapidly (see *Fig. 4.8*). As the ROC of the developed algorithm approaches the random decision line it is concluded that the developed al-

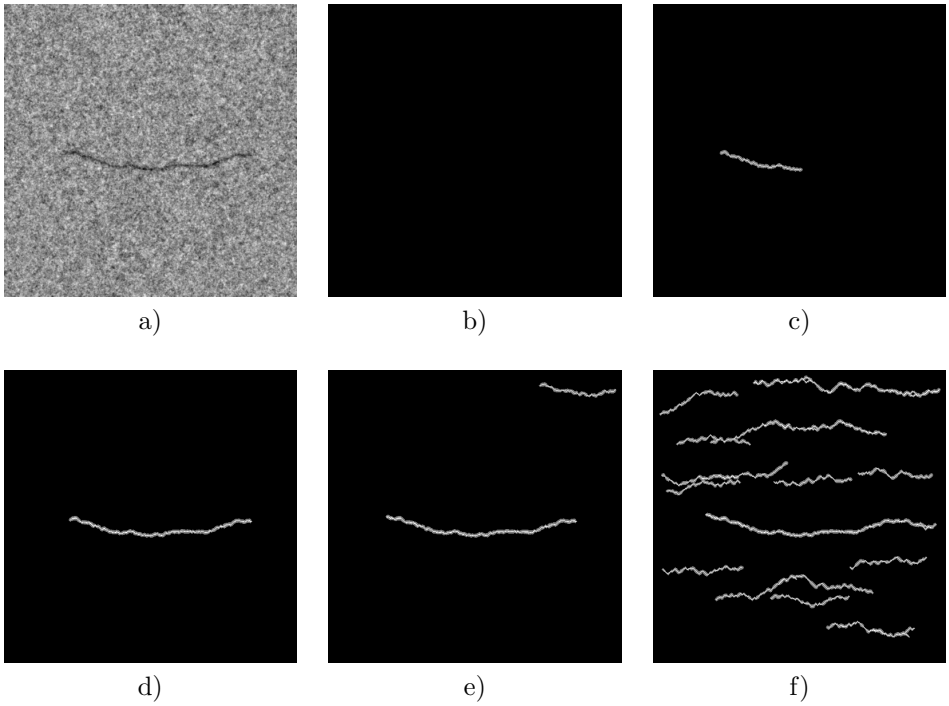


Figure 4.4: a) Source image with SNR = 1.7; b-f) Detection with varying sensitivity h and constant (for this experiment) other control parameters: min and max indication width w_{min} and w_{max} are 1 and 3 pixels, *a priori* probability of indication present $P(h)$ is 0.1, significant indication length L_{sign} is 128 pixels. b) $h = 2.7\sigma$ - too low sensitivity results in no detection - the algorithm fails; c) $h = 2.6\sigma$ - result is better, but CD rate is still insufficient; d) $h = 2.5\sigma...2.3\sigma$ - the range of optimal sensitivities which results in perfect detection with completely no false alarms (for this concrete source image), ROC plot on *Fig. 4.5* shows the optimum average performance for this sensitivities too; e) $h = 2.2\sigma$ - some over-detection can be seen; f) $h = 2.1\sigma$ - too high sensitivity results in wast false alarms - the result is unacceptable.

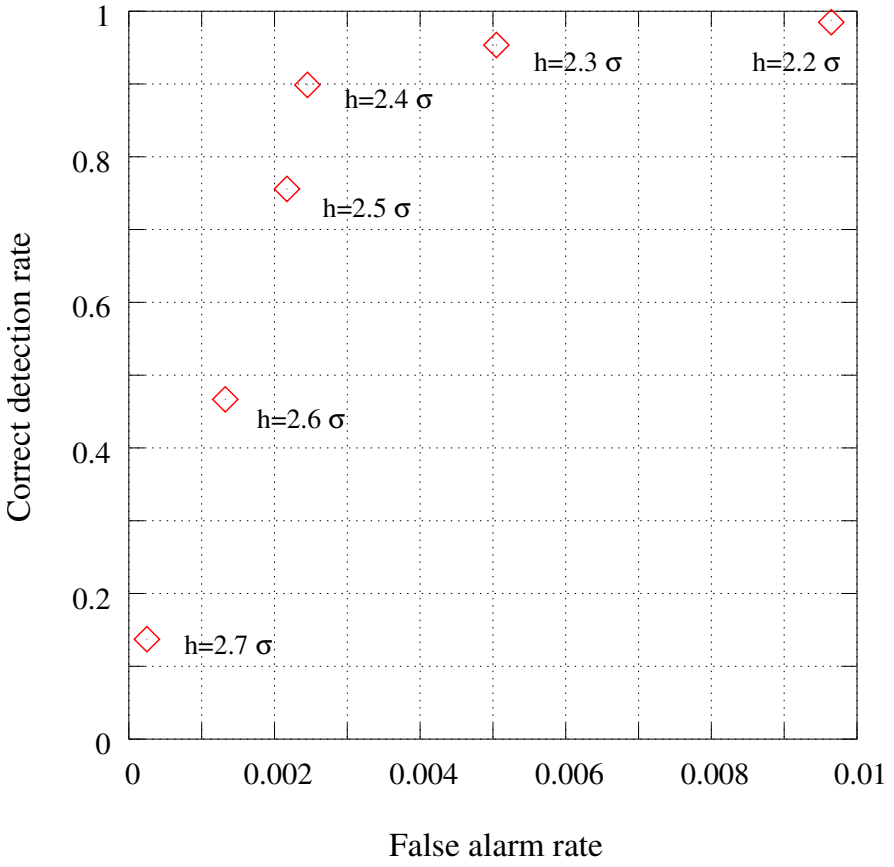


Figure 4.5: ROC of the detection algorithm. FA axis is scaled to 1% false alarms. Each point on the plot describes the average algorithm performance for a certain fixed combination of control parameters and the set of one hundred source images with SNR = 1.7. Different points on the ROC are obtained by changing of the expected indication intensity h (which control algorithm sensitivity) and processing the same dataset. Other control parameters remain unchanged as for *Fig. 4.4*.

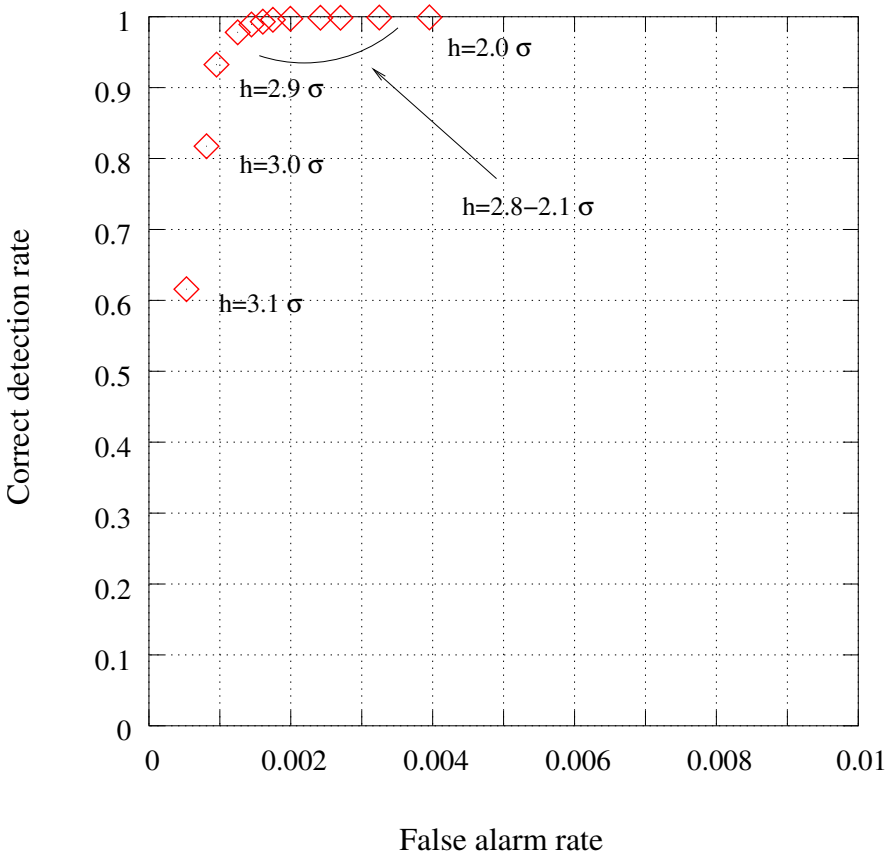


Figure 4.6: ROC of the detection algorithm for the the set of one hundred source images with SNR = 3.4. FA axis is scaled to 1% false alarms. Each point on the plot describes the average algorithm performance for a certain fixed combination of control parameters. Different points on the ROC are obtained by changing of the expected indication intensity h (which control algorithm sensitivity) and processing the same dataset. Other control parameters remain unchanged as for *Fig. 4.4*.

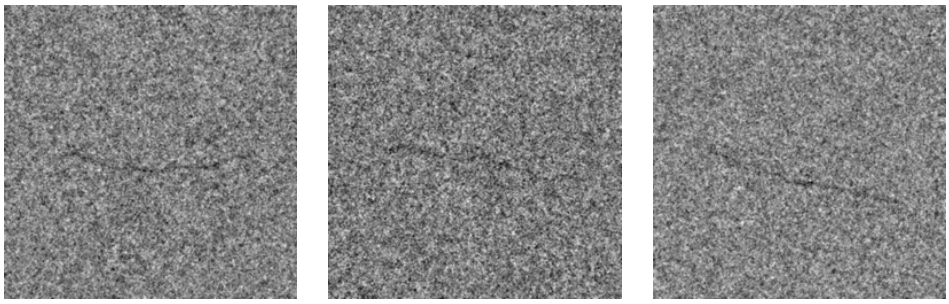


Figure 4.7: Some examples of synthetic images with SNR = 0.86. The indications are hardly visible even for a human due to the highly intensive noise.

gorithm fails for SNRs lower than 1.0. So this is the limit of applicability of the proposed approach. This is in good correspondence with visual evaluation of 2D images. At this noise level human beings have serious difficulties in the correct interpretation of the images too. For trained inspectors the limit of visibility of signals is about SNR = 1.0.

It is possible to see from the figures given above, that the expected indication intensity h influences significantly the CD and FA rates produced by the algorithm. Let us investigate the influence of the other control parameters of the algorithm on its performance. On *Fig. 4.8* and *Fig. 4.9* the series of similarly labelled points correspond to the same combination of all algorithm parameters except the expected indication intensity h, which is varied inside a series. So inside the series of the same labels only h is changed, whereas between different series the other control parameters are changed too.

It is possible to constitute from *Fig. 4.8* and *Fig. 4.9*, that points on the plot which correspond to different combinations of control parameters are grouped around some curve (obvious on *Fig. 4.8* and represented by a dashed line on *Fig. 4.9*). This allows us to expect that nearly all influences on the CD and FA rates from the suboptimal L_{sign} (the minimal significant length) and $P(h)$ (the *a priori* probability of indication presence) can be compensated by the corresponding change of h (the expected intensity of indication) in order to get the required CD or FA figures. I.e.:

- reasonable defaults can be selected for other algorithm control parameters (which ensure satisfactory performance over the selected class of images);

- only the algorithm sensitivity (h) has to be changed in order to get the required FA or CD rates.

This is valid at least for the image in experiment: a solid and relatively long indication. In case of a ragged (dashed) indication or a relatively short indication the role of the minimal significant length L_{sign} and the *a priori* probability of object presence $P(h)$ becomes more important. In that case the usage of default values may become unsatisfactory and adjustments of L_{sign} or $P(h)$ become necessary.

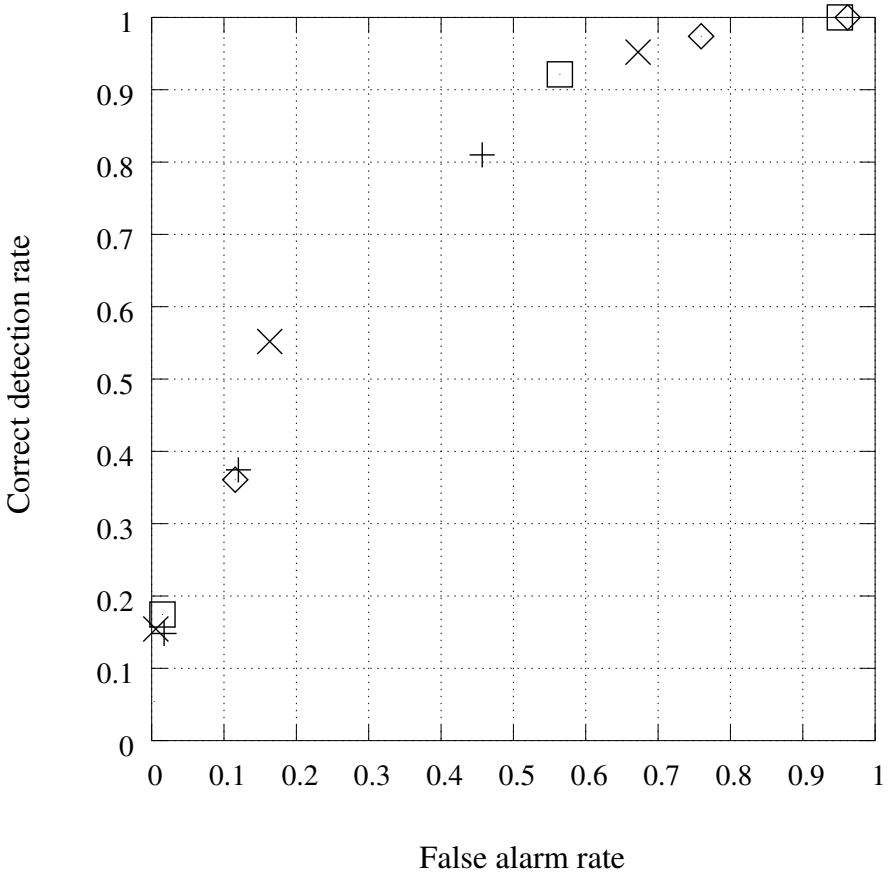


Figure 4.8: ROC of the detection algorithm for the set of one hundred source images with SNR = 0.86. FA axis is scaled to 100% false alarms. Each point on the plot describes the average algorithm performance for a certain fixed combination of control parameters. The points labelled by "$\diamond$" are obtained for $L_{sign} = 128, P(h) = 0.01$ and different values of expected indication intensity $h = 2.1 \ldots 2.3\sigma$. The "+" -points correspond to $L_{sign} = 128, P(h) = 0.50$ and varied $h = 1.8 \ldots 1.9\sigma$, "$\square$" - to $L_{sign} = 256, P(h) = 0.01$ and varied $h = 2.1 \ldots 2.3\sigma$, "$\times$" - to $L_{sign} = 256, P(h) = 0.50$ and varied $h = 1.75 \ldots 1.85\sigma$.

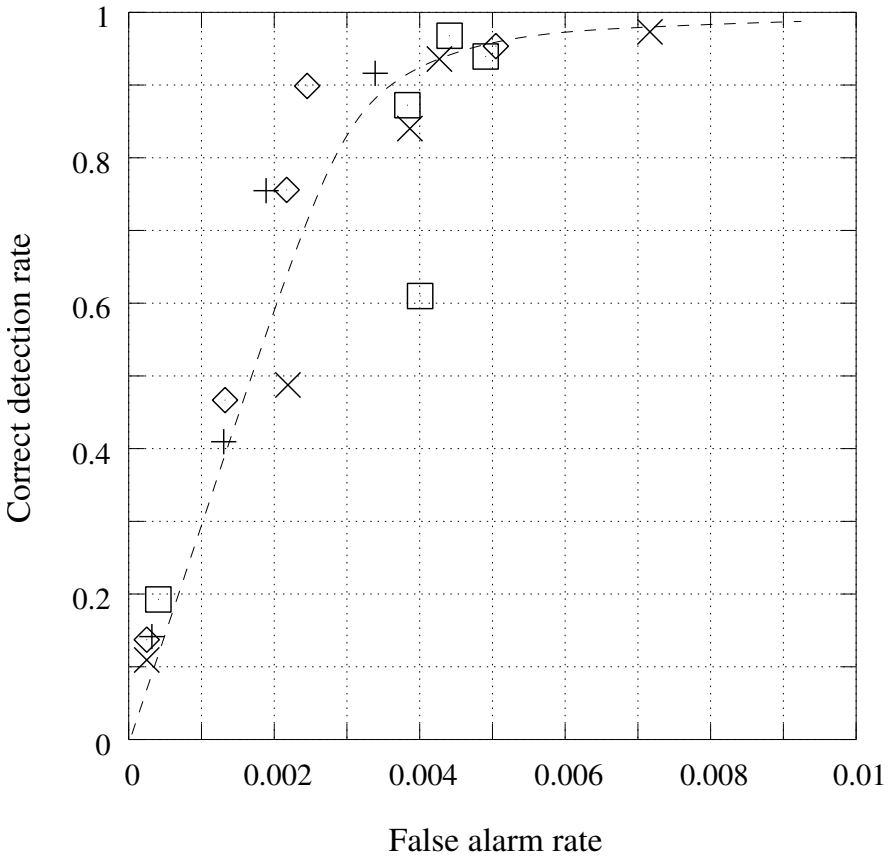


Figure 4.9: ROC of the detection algorithm for the set of one hundred source images with SNR = 1.7. FA axis is scaled to 1% false alarms. Each point on the plot describes the average algorithm performance for a certain fixed combination of control parameters. The points labelled by "$\diamond$" are obtained for $L_{sign} = 128, P(h) = 0.01$ and different values of expected indication intensity $h = 2.2 \ldots 2.7\sigma$. The "+" -points correspond to $L_{sign} = 128, P(h) = 0.50$ and varied $h = 2.0 \ldots 2.3\sigma$, "$\square$" - to $L_{sign} = 256, P(h) = 0.01$ and varied $h = 2.3 \ldots 2.7\sigma$, "$\times$" - to $L_{sign} = 256, P(h) = 0.50$ and varied $h = 1.9 \ldots 2.3\sigma$.

4.3.3 Summary on synthetic images

The use of synthetic images generated according to the assumed image model allows to investigate the detection potential of the proposed algorithm. The figures obtained for images with SNR = 1.7 (SNR = 1.0 intended) show very good algorithm capabilities in detection of indications with such low signal to noise ratios. I.e. the algorithm was able to correctly detects 95% of indications while the probability of false alarms does not exceed 0.5%. It is necessary to stress that these figures are only valid for images which perfectly fit in the assumed image model. This may be not completely true for the case of real radiographs, so in the next section the algorithm performance will be investigated on real images.

4.4 Analysis of real-world radiographs

The experimental set of real images consists of one hundred of digitised real radiographs of austenitic steel pipes from nuclear power plants. The radiographs were taken at standard conditions according to EN1435, testing class B (160 kV X-ray source, C5 system class film according to EN584-1). They were digitised with 70 μm pixel size and 12 bits grey value resolution (with a CCD NDT scanner produced by DBA Systems).

The true data are known for this image set while after the non-destructive testing the welds were explored destructively by means of grinding [39].

An example of a such radiograph and a result of detection are shown on *Fig. 4.10* and *4.11*. The detection algorithm was applied with the following control parameters: $w_{min} = 2$ pixels, $w_{max} = 6$ pixels, $L_{sign} = 128$ pixels, $P(h) = 0.05 \ldots 0.5$ and $h = 2.0 \ldots 3.5\sigma$. All indications are marked for which a positive information gain about hypothesis of presence of an indication ($dI(h)$) was estimated. The sensitivity of the algorithm was adjusted by changing of the expected indication intensity h (*Fig. 4.12*).

Because the positions of existing defects in this set are known the correct detection and false alarm rates can be found. They are presented in a ROC plot on *Fig. 4.13*. The same plot (*Fig. 4.13*) includes the CD and FA rates of several experienced inspectors [7, 39]. They have visually analysed the same set of digitised radiographs using conventional image processing means. Defects were detected with 1 cm resolution in length direction. The whole set of one hundred images was analysed by each inspector. Each point on the ROC plot indicates the CD and FA rates of each inspector averaged over the whole image set. Despite the lower CD and higher FA rates, as compared to synthetic images, the developed algorithm performs comparably to visual (manual) inspection made by human beings. Although some inspectors slightly outperform the developed algorithm, the algorithm is able to deliver more consistent results, which are easily reproducible.

The *Fig. 4.13* shows, additionally to the human results and results obtained with the developed algorithm, the ROC of an other algorithm which was applied to the same dataset [27]. This algorithm uses back-propagation neural network and was trained on the images from the same dataset. The neural network algorithm uses responses of many different local operators as its input. It was designed to discriminate between real crack indications, crack-looking indications which are not cracks

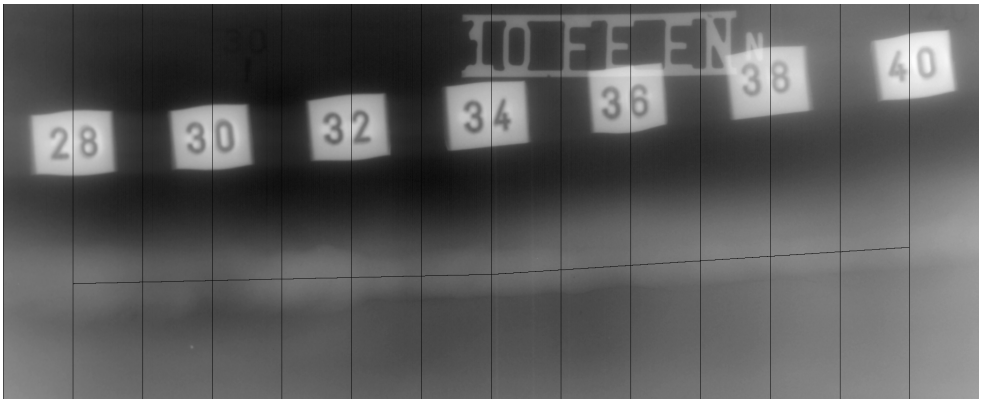


Figure 4.10: Example of a radiograph of a weld. Lead markers (numbers in white squares) are used as references during radiograph interpretation. Grid if thin black lines is presented on this image only for illustration purpose (crack detection algorithm needs as input the clean original image): horizontal line shows centre of the welding seam, vertical lines separate measurement boxes (1cm wide). Decision on success of detection is made by analysing of results for each of the measurement boxes separately.

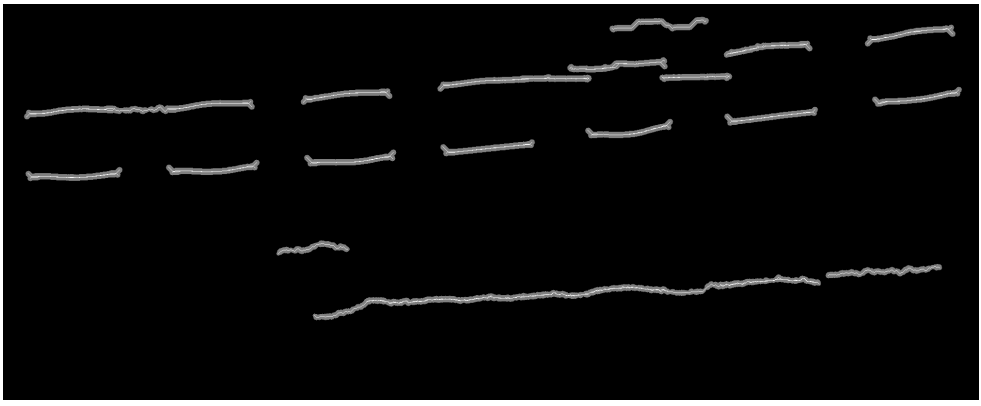


Figure 4.11: Result produced by the developed crack detection algorithm from the radiograph shown on *Fig. 4.10*. True positive detections in boxes 32-39 produced by the algorithm for this image result in 100% correct detection rate. Overdetections were made for boxes 39,40 (below the seam) and 32 (above the seam) which result in 18.75% false alarm rate. The false detections in the upper image part are caused by the lead markers and are not taken into account as they are not inherent to radiographic images and obviously do not belong to the region of interest (the welding seam).

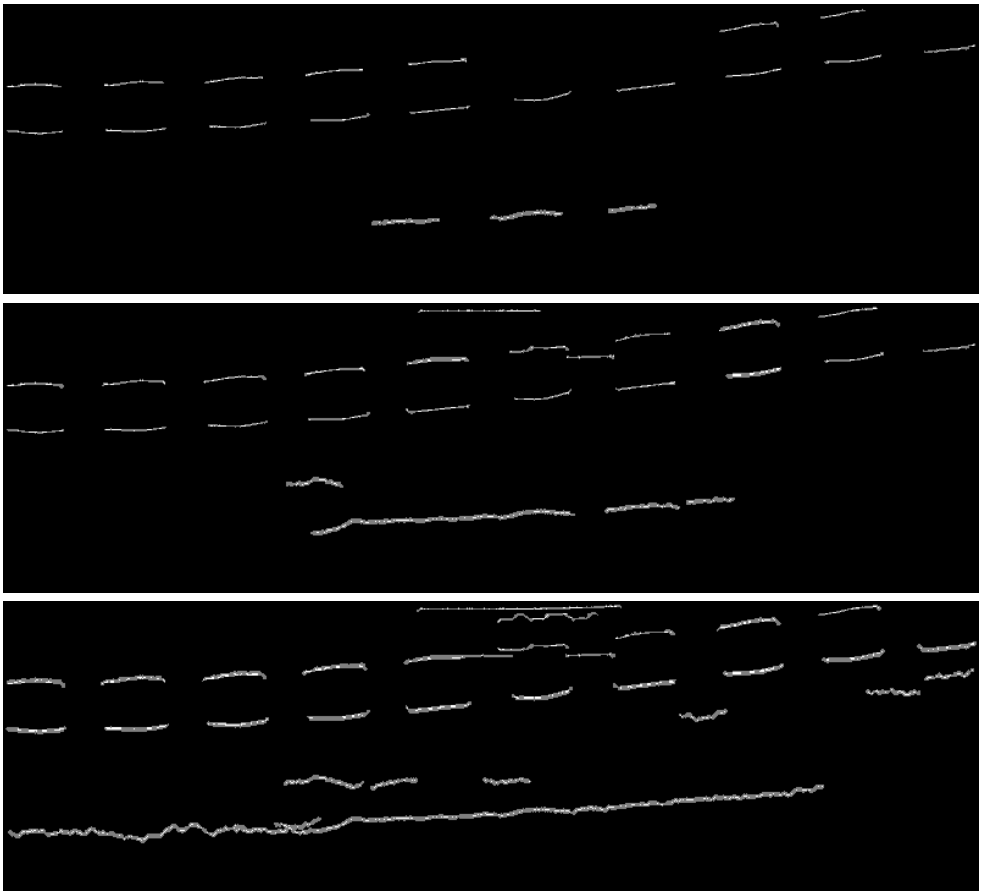


Figure 4.12: Detection of crack indications in real radiographs with different expected indication intensities h. Changing of h influences the algorithm sensitivity and leads therefore to different amounts of found objects. Parameters used: $w_{min} = 2$ pixels, $w_{max} = 6$ pixels, $L_{sign} = 128$ pixels, $P(h) = 0.1$ and $h = \{2.0\sigma, 2.5\sigma, 3.0\sigma\}$

in reality and noise instances. The algorithm works solely in local domain [27].

There is a reason in this trial for the imperfect performance of humans, neural network algorithm [27] as well as the developed algorithm. The radiographs of test specimens have been made only from one direction (only one projection was taken of each part of the weld). Due to that it is very likely that not all crack defects were radiographed from the optimal direction and therefore they are not imaged completely on the radiographs and can not be detected completely. On the opposite, the grinding allows to find all arbitrary located defects inside of each cut, but only at the position of this cut. So the number of correct detections will be referred to the whole number of defect areas even if they are not imaged on the radiographs. As a consequence the CD rates of humans as well as the algorithm can not reach 100%.

From another side, the grinding has been made with 1 cm spacing: it provides no information about defects between the cuts. I.e. if a short crack is located between the cuts, it will be not counted as a true defect and the calculated FA rate will be affected (both for humans and the detection algorithm).

It is also worth to mention again that the developed algorithm detects all elongated crack-like indications. This means that real crack indication and crack-like looking objects, i.e. root undercut from the welding process, which are not crack defects in reality, cannot be distinguished by the algorithm. At the same time, the trained inspectors are able to do so without difficulties. Cracks and undercuts can be easily distinguished in destructive tests too. This may be a reason why the developed algorithm does not perform absolutely better than humans.

4.5 An example of another application

The crack detection algorithm described in this work was designed mainly having weld inspection in mind. Nevertheless the universality of the algorithm for other applications was retained as far as possible and the algorithm has been found useful for the detection of other elongated objects or cracks in other specimens too. Here, the algorithm has been applied to the examination of radioactive waste containers.

The vitrification of high level radioactive waste from nuclear reprocessing facilities is used as the standard method of immobilisation of such waste for a long period of time. The melted glass containing about 15 weight-% of the high level radioactive waste (HLW) is poured into thick walled steel canisters in several charges. As the melt cools down, the inner part solidifies last raising internal tensions that might release forming cracks in the glass body. In the scenarios of long range storage water contact is postulated as a potential risk. Consequently, the surface of contact of the glass block is subject to corrosive attack (lixiviation) and should be avoided or as small as possible. For a determination of the surface area of the vitrified waste the evaluation of the crack length is essential.

Several canisters produced with different cooling procedures were analysed at BAM using the BAM-designed universal tomograph. Details about the experimental setup and other related information can be found in [22] and the references therein. *Fig. 4.14.a* shows an example of reconstructed slice of the test glass canister. The pixel grey values encode the linear attenuation coefficient μ of the material. Multiple crack indications are visible.

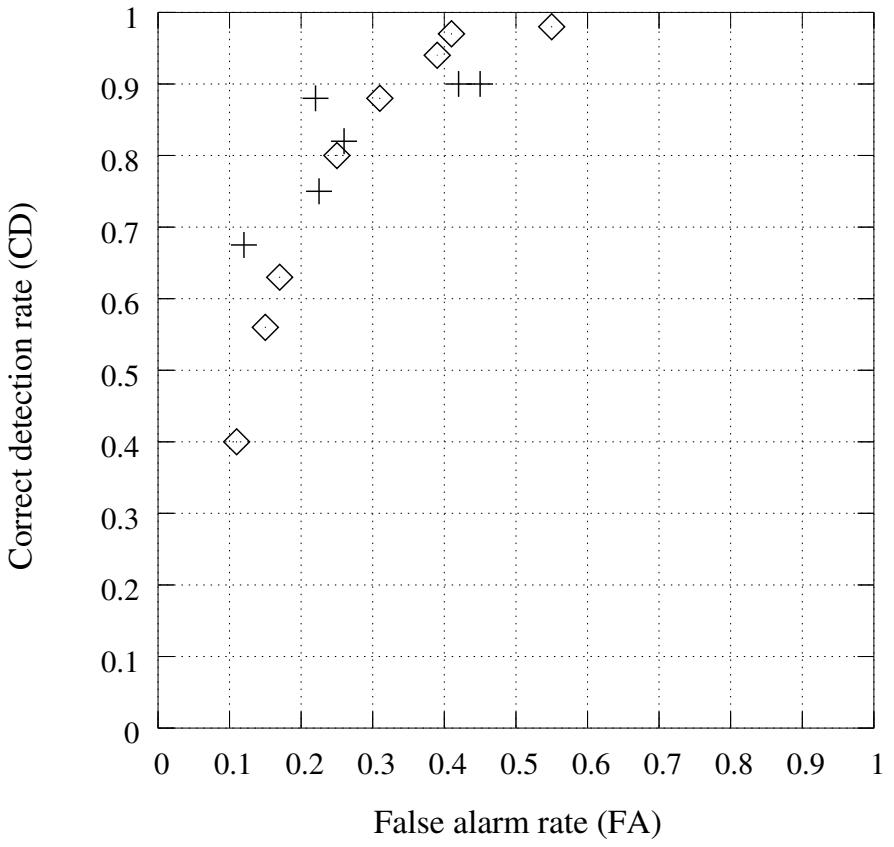
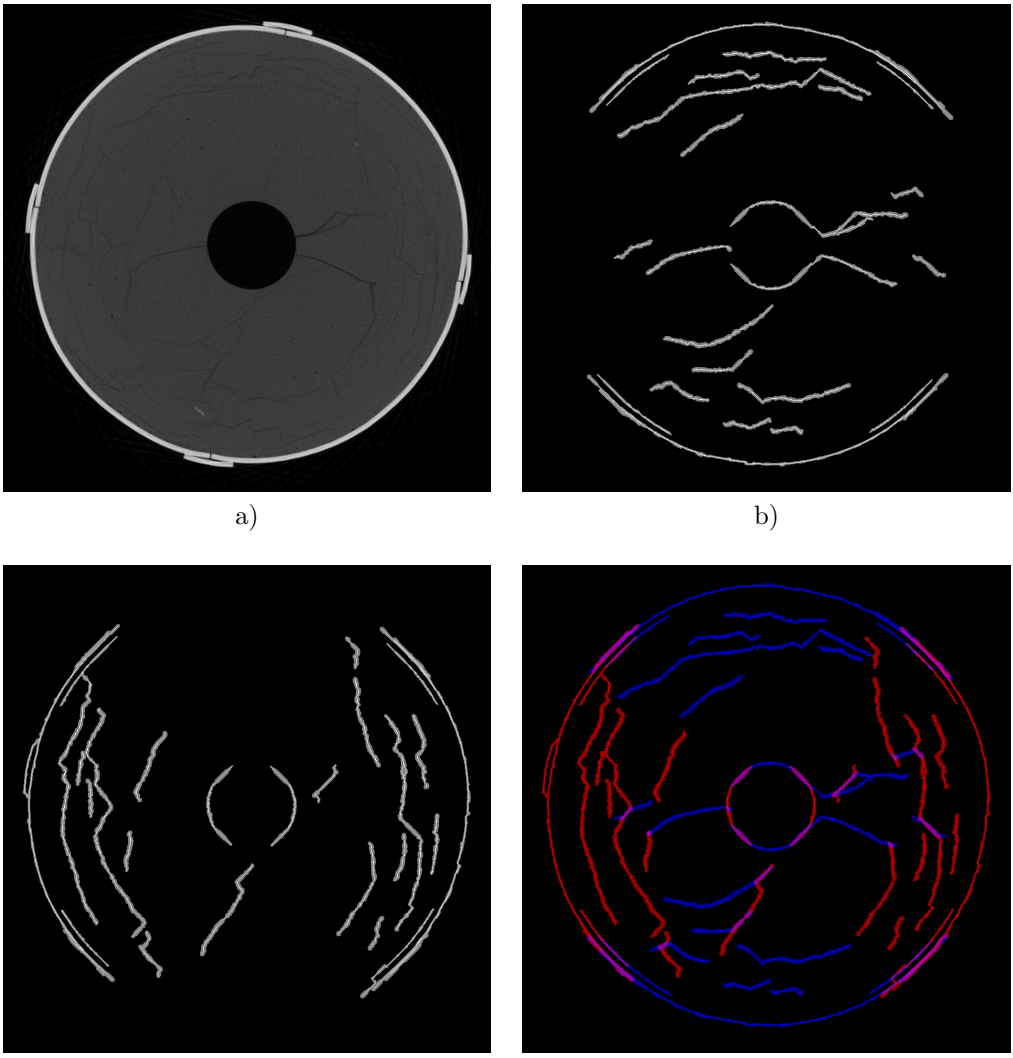


Figure 4.13: Operating characteristic of the current crack detection algorithm in comparison to the performance of several experienced inspectors. The ROC was obtained on the set of one hundred real radiographs for which the true data are known. "+" - human results (different inspectors, one point per inspector, CD and FA averaged over the image set), "□" - neural networks algorithm [27], "◇" - the developed algorithm (each point for different combinations of control parameters). The algorithm control parameters used: $w_{min} = 2$, $w_{max} = 6$, $L_{sign} = 128$, $P(h) = 0.05 - 0.5$ and $h = 2.0 - 3.5$

The developed crack detection algorithm was used to automatically label the crack indications. The present implementation of the algorithm searches only either for horizontal ($\pm 45°$ from horizontal) or vertical ($\pm 45°$ from vertical) directions. These search directions and allowed variations of direction are not inherent property of the proposed approach but are specific for the current implementation (which was made for the detection of longitudinal cracks in welding seams). Nevertheless difficulties with detection of curves, which change their global direction in more than 90 degrees, are inherent for the used search procedure and therefore for the detection algorithm at all. This means that (semi)circular structures which are visible on *Fig. 4.14.a* cannot be detected in a single step.

Therefore the algorithm was applied for horizontal (*Fig. 4.14.b*) and vertical (*Fig. 4.14.c*) directions in series and then these results combined in a single image (*Fig. 4.14.d*). Algorithm control parameters were chosen as follows: several profiles of average crack indication were taken to estimate *minimal* and *maximal* allowed indication width, *a priori* probability is left at default 0.1. Then several tests were made for different significant indication lengths ($L_{sign} = \{64, 128\}$ pixels) and different sensitivities ($h = \{3, 4, 5, 6, 7\}\sigma$) to reveal useful combinations. The combination of both search directions reveal successfully all significant flaws in this tomorgam.



a) b)

c) d)

Figure 4.14: a) An example tomogram of the test glass canister [22]. b)-d) Cracks automatically marked by the developed algorithm: b) horizontal pass, c) vertical pass, d) results b and c merged together. Algorithm control parameters: minimal and maximal indication width (w_{min} and w_{max}) 2 and 5 pixels, *a priori* probability of indication present $P(h)$ is 0.1, significant indication length L_{sign} is 64 pixels and expected SNR is 5.0 ($h = 5.0\sigma$).

Chapter 5

Summary and outlook

The task of automatic detection of crack-like defect indications in digital radiographs of welding seams has been considered in this work. This task is not trivial due to the inherent properties of radiography (Chapter 1): low signal to noise ratio and absence of formal definition of crack shape are the most difficult obstacles. The literature analysis given in Chapter 2 has revealed that certain conventional approaches, i.e. all purely local methods (various gradient thresholders, indication profile analysers, etc.) as well as segmentation techniques based on global pixel statistics, are not suited to the solution of this detection task. On the other hand, the approaches which use structural global features of the objects sought for look promising. This is first of all the integration of an informative feature over the whole indication length, as it allows to eliminate the random component caused by the noise.

Methods which use integration to suppress noise are commonly in use for decades in the field of information transmission and radar [62]. However, their direct application for the detection of crack indications is connected with certain difficulties. In radar or information transmission, the transmitted signal is always one dimensional (evolving over time) and usually of known length. The task consists in the location of the signal in time (one unknown). In the task considered in this work the signal is two dimensional and its spatial shape and length are unknown. Therefore, many unknown variables have to be handled. This introduces high computational complexity of the detection.

In the presented work the problem of excessive computational complexity is solved by design of the object estimation function of a special form. The proposed estimation function is constructed as the sum of *a posteriori* information gains in each point along the indication. This allows to find the optimum of the estimation function in a sequential way using serial dynamic programming and thus avoiding the excessive computational complexity.

The developed algorithm is tested on several sets of synthetic as well as real radiographs. Several ROCs were plotted on the basis of the experimental evaluation and used for the assessment of the algorithm performance. The ROCs show how the percentage of correct detections (CD) and the percentage of false detections (false alarms, FA) made by the algorithm depend on the control parameters and different source images. It is shown, that for synthetic images, which are generated accord-

ing to the assumed image model, the developed algorithm demonstrates very good performance. It detects correctly 95% of indications with an SNR = 1.7, while the probability of false alarms does not exceed 0.5%. Experiments with real radiographs of welding seams show CD and FA rates which are comparable to the results of human inspectors having analysed the same image set (*Fig. 4.13*). This is considered as a good result, although the absolute numbers for real radiographs are naturally not as good as for the synthetic images.

Despite the reported good performance, the following improvements of the developed algorithm can be investigated in the future:

- Approximation of the background by surfaces of higher order may be more adequate for images with a fast changing background (rather than the piecewise linear model presently used).
- The chained spatial statistical indication model (as introduced in Section 3.5.6) looks more appropriate for the description of crack propagation than the model which assumes statistically independent local indications (from the physical point of view). However, if such a model is used, the exact optimisation using dynamic programming becomes problematic.
- Shape features more specific for crack indications can be used. For example, the presence of a bifurcation point (point where the crack splits into two cracks) helps to distinguish the real crack indication from the crack-like ones (e.g. undercut).
- Currently, detection of all suspicious crack-like indications on a image with a size 2000 × 8000 pixels can take (depending of the number of indications and noise level) from 10 minutes up to 2-3 hours on a 3 GHz Intel CPU. A decrease of the execution time to 3-5 minutes is desirable for practical applications. An implementation on parallel hardware (e.g. computing clusters) is the simplest way to satisfy this requirement.

Appendix A

C++ implementation

```
// local.h:

#ifndef _LOCAL_
#define _LOCAL_

#include <math.h>
#include "ip.h"

class Retina
{
  u_short Wmin;
  u_short Wmax;
  ip::RasterImage<ip::gv16bpp> * masks;

public:
  Retina  (const ip::RasterImage<ip::gv16bpp> & source,
           const u_short Wmin, const u_short Wmax);
  ~Retina (void) { delete [] masks; }

  const u_short GetWmin  (void) const { return Wmin; };
  const u_short GetWmax  (void) const { return Wmax; };
  const u_long  GetXSize (void) const { return (masks!=NULL) ? masks->GetXSize() : 0; };
  const u_long  GetYSize (void) const { return (masks!=NULL) ? masks->GetYSize() : 0; };

  // dir = child_x - parent_x -> if dir==0 it means "vertical"
  const float getcontrast_3point_0order (const u_long x, const u_long y,
     const u_short w, const signed short dir) const; // 3-point null-order
  const float getcontrast_3point_1order (const u_long x, const u_long y,
     const u_short w, const signed short dir) const; // 3-point 1st-order
  const float getcontrast_5point_1order (const u_long x, const u_long y,
     const u_short w, const signed short dir) const; // 5-point 1st-order
};

#endif

// end of local.h

// local.cpp:

#include <stdio.h>
#include "ip.h"
```

```
#include "local.h"

Retina::Retina (const ip::RasterImage<ip::gv16bpp> & source,
                const u_short _Wmin, const u_short _Wmax)
{
  try
  {
    if (_Wmin > _Wmax) throw std::invalid_argument
                      ("Wrong cells range is given to Retina's constructor.");
    masks = NULL;
    masks = new ip::RasterImage<ip::gv16bpp> [_Wmax - _Wmin + 1];
    if (masks == NULL) throw iddqd::memory_allocation_fault ();
    Wmin = _Wmin, Wmax = _Wmax;
    for (u_short w = Wmin; w <= Wmax; w++) ip::RunningMean(source, w, w, masks+w-Wmin);
  }
  catch (...)
  {
    delete [] masks;
    masks = NULL, Wmax = Wmin = 0;
    throw;
  }
}

// retina's model is "null-order aprox. by 3 points"
const float Retina::getcontrast_3point_0order
  (const u_long x, const u_long y, const u_short w, const signed short dir) const
{
  ip::RasterImage<ip::gv16bpp>  * w_image = masks + w - Wmin;
  s_int e1 = s_int(w_image->getelement(x+w,y-dir)) - s_int(w_image->getelement(x,y));
  s_int e2 = s_int(w_image->getelement(x-w,y+dir)) - s_int(w_image->getelement(x,y));
  return (e1 < e2) ? e1 : e2;
}

// retina's model is "1st-order aprox. by 3 points"
const float Retina::getcontrast_3point_1order
  (const u_long x, const u_long y, const u_short w, const signed short dir) const
{
  ip::RasterImage<ip::gv16bpp>  * w_image = masks + w - Wmin;
  return (float(w_image->getelement(x+w,y-dir)) + w_image->getelement(x-w,y+dir))/2 -
          w_image->getelement(x,y);
}

// retina's model is "1st-order aprox. by 5 points"
const float Retina::getcontrast_5point_1order
  (const u_long x, const u_long y, const u_short w, const signed short dir) const
{
  ip::RasterImage<ip::gv16bpp>  * w_image = masks + w - Wmin;
  s_int e1 = 2*int(w_image->getelement(x+w,y-dir)) -
             int(w_image->getelement(x+2*w,y-2*dir)) - int(w_image->getelement(x,y));
  s_int e2 = 2*int(w_image->getelement(x-w,y+dir)) -
             int(w_image->getelement(x-2*w,y+2*dir)) - int(w_image->getelement(x,y));
  return (e1 < e2) ? e1 : e2;
}

// end of local.cpp
```

```
// combine.h:

#ifndef _COMBINE_
#define _COMBINE_
```

```
#include "conf.h"
#include "local.h"

void Trace (const Conf * const conf, const Retina * const retina,
            ip::RasterImage<ip::gv8bpp> * const result);

#endif

// end of combine.h
```

```
// combine.cpp:

#include <stdio.h>
#include <string.h>
#include <math.h>
#include "ip.h"
#include "conf.h"
#include "combine.h"

#define GETCONTRAST getcontrast_3point_1order

static const u_short NOT_A_COORD   = 0xFFFF;
static const u_char  NOT_A_WINSIZE = 0xFF;

struct Node
{
  float info;
  float P1;
  u_short ln;
  u_short father_x;
  u_short father_y;
  u_char  father_w;

  Node (void) { Clear(); }

  void Clear (void)
    { info=-FLT_MAX, P1=0, ln=0,
      father_x=father_y=NOT_A_COORD, father_w=NOT_A_WINSIZE; }
  void Set (float _info, float _P1)
    { info=_info, P1=_P1, ln=1, father_x=father_y=NOT_A_COORD, father_w=NOT_A_WINSIZE; }
  void Set (float _info, float _P1, u_long _ln, u_long x, u_long y, u_short w)
    { info=_info, P1=_P1, ln=u_short(_ln),
      father_x=u_short(x), father_y=u_short(y), father_w=u_char(w); }
};

class Grid
{
  u_long  Xsize;
  u_long  Ysize;
  u_short Wmin;
  u_short Wrange;
  iddqd::vector<Node> nodes;

public:
  Grid (u_long Xsize, u_long Ysize, u_short Wmin, u_short Wmax);
  void Clear (void);
  Node * get (u_long x, u_long y, u_short w)
    { return (nodes.GetRaw() + (y*Xsize+x)*Wrange + w - Wmin); };
};
```

```
Grid::Grid (u_long X, u_long Y, u_short Wminimum, u_short Wmaximum)
{
  try
  {
    Xsize = X, Ysize = Y, Wmin = Wminimum, Wrange = Wmaximum - Wminimum + 1;
    nodes.Create(Xsize*Ysize*Wrange);
  }
  catch (...)
  {
    Xsize = Ysize = Wmin = Wrange = 0;
    throw;
  }
}

void Grid::Clear (void)
{
  nodes = Node();
}

static const u_char Nothing = 0;
static const u_char Up2Down = 1;
static const u_char Down2Up = 2; // (Up2Down & Down2Up) must be 0 <- empty intersection

class Tracer
{
  // life-time stable group (all coordinates are REAL!)
  const Conf   * const conf;
  const Retina * const retina;
  ip::RasterImage<ip::gvAnalog> * context_energy_map;
  ip::RasterImage<u_char> status_map;

  // call-to-call variating group
  bool  reverseY;
  Node  best_path_top;
  Grid  grid;          // <- y-coordinate is virtual: if (reverseY==true)
                       // then retina/status_map/context_energy_map.y = Ymax-1-grid.y

  // procs
  float GetP1post (float P1prior, float snr_est)
  {
    register float tmp = snr_est-conf->signal;
    float L1 = P1prior * exp(-0.5*tmp*tmp);
    float L0 = (1-P1prior) * exp(-0.5*snr_est*snr_est);
    return L1/(L1+L0);
  }
  float GetInfoGain (float Pprior, float Ppost)
  {
    return (-log(Pprior) - (-log(Ppost)));
  }

  // search is performed in virtual coordinates!
  void MakeRoot  (const u_long x, const u_long y, const u_short w);
  void MakeChild (const u_long father_x, const u_long father_y, const u_short father_w,
                  const u_long x, const u_long y, const u_short w);
  void OpenNode  (const u_long to_open_x, const u_long to_open_y,
                  const u_short to_open_w);

public:
```

```
  Tracer (const Conf * const _conf, const Retina * const _retina,
          const u_short context_size);
  ~Tracer (void) { delete [] context_energy_map; }
  Node FindNextCurve (const byte search_direction);
  void Visualize     (ip::RasterImage<ip::gv8bpp> * const result);
};

Tracer::Tracer (const Conf * const _conf, const Retina * const _retina,
                const u_short context_size)
  : conf               (_conf),
    retina             (_retina),
    context_energy_map (NULL),
    status_map         (_retina->GetXSize(), _retina->GetYSize(), Nothing),
    grid               (_retina->GetXSize(), _retina->GetYSize(),
                        _retina->GetWmin(), _retina->GetWmax())
{
  context_energy_map =
    new ip::RasterImage<ip::gvAnalog> [retina->GetWmax()-retina->GetWmin()+1];
  for (u_short w = retina->GetWmin(); w <= retina->GetWmax(); w++)
  {
    // make a map of indications distribution
    ip::RasterImage<ip::gvAnalog> indication_map;
    indication_map.Create(retina->GetXSize(), retina->GetYSize());
    for (u_long y = retina->GetWmax(); y < retina->GetYSize()-retina->GetWmax(); y++)
    {
      ip::gvAnalog * pim = indication_map.GetRaw() +
                           y*retina->GetXSize() + retina->GetWmax();
      for (u_long x = retina->GetWmax(); x < retina->GetXSize()-retina->GetWmax(); x++)
      {
        iddqd::averager local_est (0, 0);
for (s_short dir = -w; dir <= w; dir+=w)
  local_est.add (retina->GETCONTRAST(x, y, w, dir));
        *(pim++) = local_est.get();
      }
    }
    // smooth indications map, but preserve unsmoothed version too
    ip::RasterImage<ip::gvAnalog> smoothed_indication_map;
    ip::RunningMean (indication_map, context_size, context_size,
                     &smoothed_indication_map);
    // make a map of deviations distribution
    ip::RasterImage<ip::gvAnalog> * cur_noise_map = context_energy_map +
                                                    w-retina->GetWmin();
    ip::RunningDisp (indication_map, smoothed_indication_map,
                     context_size, context_size, cur_noise_map);
    ip::RunningMean(*cur_noise_map, context_size, context_size, cur_noise_map);
    ip::gvAnalog * pp = cur_noise_map->GetRaw();
    for (u_long l = 0; l < retina->GetXSize()*retina->GetYSize(); l++, pp++)
      *pp = sqrt(*pp);
  }
}

void Tracer::MakeRoot (const u_long x, const u_long y, const u_short w)
{
  u_long real_y = reverseY ? (retina->GetYSize()-1-y) : y;
  float P1post = GetP1post (conf->apriori_prob,
                   retina->GETCONTRAST(x, real_y, w, 0) /
                   (context_energy_map+w-retina->GetWmin())->getelement(x,real_y));
  float new_info = GetInfoGain (conf->apriori_prob, P1post);
  Node * place_to_store = grid.get(x, y, w);
```

```
  if (place_to_store->info < new_info) place_to_store->Set (new_info, P1post);
}

void Tracer::MakeChild (const u_long father_x, const u_long father_y,
                        const u_short father_w, const u_long x,
                        const u_long y, const u_short w)
{
  u_long real_y = reverseY ? (retina->GetYSize()-1-y) : y;
  Node * father = grid.get(father_x, father_y, father_w);
  float P1prior = father->P1 * conf->prolongation_prob;
  if (P1prior < conf->apriori_prob) P1prior = conf->apriori_prob;
  float P1post  = GetP1post (P1prior, retina->GETCONTRAST(x, real_y, w,
    s_short(s_int(reverseY?(father_x-x):(x-father_x))*w)) /
    (context_energy_map+w-retina->GetWmin())->getelement(x,real_y));
  float new_info  = father->info + GetInfoGain (conf->apriori_prob, P1post);
  Node * place_to_store = grid.get(x, y, w);
  if (place_to_store->info < new_info)
    place_to_store->Set (new_info, P1post, father->ln+1, father_x, father_y, father_w);
}

void Tracer::OpenNode (const u_long to_open_x, const u_long to_open_y,
                       const u_short to_open_w)
{
  for (s_char dw = -1; dw <= 1; dw++)
  {
    byte w = to_open_w + dw;
    if (w >= retina->GetWmin() && w <= retina->GetWmax())
    {
      MakeChild (to_open_x, to_open_y, to_open_w, to_open_x-1, to_open_y+1, w);
      MakeChild (to_open_x, to_open_y, to_open_w, to_open_x,   to_open_y+1, w);
      MakeChild (to_open_x, to_open_y, to_open_w, to_open_x+1, to_open_y+1, w);
    }
  }
}

Node Tracer::FindNextCurve (const byte search_direction)
{
  best_path_top.Clear(), grid.Clear();
  reverseY = (search_direction == Down2Up);
  for (u_long y=2*retina->GetWmax(); y<retina->GetYSize()-2*retina->GetWmax(); y++)
    for (u_long x=2*retina->GetWmax(); x<retina->GetXSize()-2*retina->GetWmax(); x++)
      if ((status_map.getelement(x, y) & search_direction) != search_direction)
      {
for (u_short w = retina->GetWmin(); w <= retina->GetWmax(); w++)
{
  MakeRoot (x, y, w); // try to start new path from here
  OpenNode (x, y, w); // open current node
  Node * node_here = grid.get(x, y, w);
  // remember best_path
  if (node_here->ln >= conf->length_min && node_here->info > best_path_top.info)
    best_path_top.Set(node_here->info, node_here->P1, node_here->ln, x, y, w);
}
      }
  return best_path_top;
}

void Tracer::Visualize (ip::RasterImage<ip::gv8bpp> * const result)
{
  if (best_path_top.ln == 0) return;
```

```
  u_char path_mask = reverseY ? Down2Up : Up2Down;

  u_long  x = best_path_top.father_x;
  u_long  y = best_path_top.father_y;
  u_short w = best_path_top.father_w;
  while (y != NOT_A_COORD)
  {
    u_long real_y = reverseY ? (retina->GetYSize()-1-y) : y;
    result->setelement (x, real_y, 0xFF);
    for (s_int dy = -s_int(w); dy <= s_int(w); dy++)
      for (s_int dx = -s_int(w); dx <= s_int(w); dx++)
if (hypot(dx,dy) <= w)
{
  status_map.SetElement
            (x+dx, y+dy, status_map.GetElement(x+dx, y+dy) | path_mask);
  if (result->GetElement(x+dx, real_y+dy) < 0x7F)
            result->SetElement (x+dx, real_y+dy, 0x7F);
}
    Node * this_node = grid.get(x, y, w);
    x = this_node->father_x, y = this_node->father_y, w = this_node->father_w;
  }
}

void Trace (const Conf * const conf, const Retina * const retina,
            ip::RasterImage<ip::gv8bpp> * const result)
{
  result->Create(retina->GetXSize(), retina->GetYSize(), 0);

  Tracer tracer (conf, retina, conf->length_min/2);
  // do Up2Down trace
  Node top_of_path = tracer.FindNextCurve (Up2Down);
  while (top_of_path.info > 0)
  {
    tracer.Visualize (result);
    top_of_path = tracer.FindNextCurve (Up2Down);
  }
  // do Down2Up trace
  top_of_path = tracer.FindNextCurve (Down2Up);
  while (top_of_path.info > 0)
  {
    tracer.Visualize (result);
    top_of_path = tracer.FindNextCurve (Down2Up);
  }
}

// end of combine.cpp
```

```
// conf.h:

#ifndef _CONF_
#define _CONF_

#define SEARCH_VERTICAL   1
#define SEARCH_HORIZONTAL 2

struct Conf
{
  bool verbose;

  char source [128];
```

```
  char result [128];

  signed   int search; // SEARCH_VERTICAL or SEARCH_HORIZONTAL constants
  unsigned int width_min;
  unsigned int width_max;
  unsigned int length_min;
  float        signal;
  float        apriori_prob;
  float        prolongation_prob;

  Conf (int argc, char** argv);
};

#endif

// end of conf.h
```

```
// conf.cpp:

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "iddqd.h"
#include "conf.h"

Conf::Conf (int argc, char** argv)
{
  verbose = 0;

  strcpy (source, "");
  strcpy (result, "");

  search = SEARCH_HORIZONTAL;
  width_min  = 2;
  width_max  = 6;
  length_min = 128;
  apriori_prob      = 0.50;
  prolongation_prob = 0.95;
  signal = 3;

  if (argc < 2)
  {
    iddqd::messager->info((std::string("\nUsage: ") + argv[0] + " <args>\n" +
    "Args:\n" +
    "  source:<filename>        - Set source image (mandatory).                 \n" +
    "  result:<filename>        - Set result image (mandatory).                 \n" +
    "  search-horizontal |                                                      \n" +
    "  search-vertical          - Set approx. direction of objects to search for.\n" +
    "  width-min:<pix>          - Set minimal width of objects to look for. Def.2\n" +
    "  width-max:<pix>          - Set maximal width of objects to look for. Def.6\n" +
    "  length-min:<pix>         - Set minimal length of object to consider.     \n" +
    "                                                          Default is 128\n" +
    "  apriori-prob:<prob>      - Apriori probability. Default is 0.5           \n" +
    "  prolongation-prob:<prob> - Prolongation probability. Default is 0.95     \n" +
    "  signal:<SNR>             - Define hypothesis of presence. Default is 3.0  \n"
    ).c_str());
  }
  for (int i = 1; i < argc; i++)
  {
    if (strcmp(argv[i], "verbose") == 0)
```

```
{
  verbose = true;
  continue;
}
char key [128];
strcpy (key, "source:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  strcpy(source, argv[i]+strlen(key));
  continue;
}
strcpy (key, "result:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  strcpy(result, argv[i]+strlen(key));
  continue;
}
if (strcmp(argv[i], "search-vertical") == 0)
{
  search = SEARCH_VERTICAL;
  continue;
}
if (strcmp(argv[i], "search-horizontal") == 0)
{
  search = SEARCH_HORIZONTAL;
  continue;
}
strcpy (key, "width-min:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  unsigned int ui = 0xFFFF;
  sscanf (argv[i] + strlen(key), "%u", &ui);
  if (ui < 0xFFFF) width_min = ui;
  continue;
}
strcpy (key, "width-max:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  unsigned int ui = 0xFFFF;
  sscanf (argv[i] + strlen(key), "%u", &ui);
  if (ui < 0xFFFF) width_max = ui;
  continue;
}
strcpy (key, "length-min:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  unsigned int ui = 0xFFFF;
  sscanf (argv[i] + strlen(key), "%u", &ui);
  if (ui < 0xFFFF) length_min = ui;
  continue;
}
strcpy (key, "signal:");
if (strncmp (argv[i], key, strlen(key)) == 0)
{
  float f = -1;
  sscanf (argv[i] + strlen(key), "%f", &f);
  if (f > 0) signal = f;
  continue;
}
```

```
    strcpy (key, "apriori-prob:");
    if (strncmp (argv[i], key, strlen(key)) == 0)
    {
      float f = -1;
      sscanf (argv[i] + strlen(key), "%f", &f);
      if (f >= 0 && f <= 1.0) apriori_prob = f;
      continue;
    }
    strcpy (key, "prolongation-prob:");
    if (strncmp (argv[i], key, strlen(key)) == 0)
    {
      float f = -1;
      sscanf (argv[i] + strlen(key), "%f", &f);
      if (f >= 0 && f <= 1.0) prolongation_prob = f;
      continue;
    }
    fprintf (stderr, "Ignoring unknown option \"%s\"\n", argv[i]);
  }
  if (strcmp(source,"")==0)
    throw std::runtime_error ("Mandatory argument 'source:<filename>' is not set!");
  if (strcmp(result,"")==0)
    throw std::runtime_error ("Mandatory argument 'result:<filename>' is not set!");
}

// end of conf.cpp
```

```
// cognition.cpp:

#include <stdio.h>
#include "ip.h"
#include "conf.h"
#include "local.h"
#include "combine.h"

iddqd::ConsoleMessagingInterface console_messager;
iddqd::MessagingInterface * iddqd::messager = &console_messager;

int main (int argc, char * argv[ ])
{
  try
  {
    Conf conf (argc, argv);
    ip::RasterImage<ip::gv16bpp> source;
    source.Load(conf.source);

    if (conf.search == SEARCH_HORIZONTAL) ip::Rotate90CW (source, &source);
    Retina retina (source, conf.width_min, conf.width_max); // do some fast precalc.
    ip::RasterImage<ip::gv8bpp> result;
    Trace (&conf, &retina, &result);  // <-- here control seats whole time
    if (conf.search == SEARCH_HORIZONTAL) ip::Rotate90CCW (result, &result);

    int sz = strlen(conf.result);
    if ((sz > 4) && (strcmp(conf.result+sz-4, ".bmp")==0))
         result.SaveBMP (conf.result);
    else result.SaveTIFF(conf.result);
    return 1;
  }
  catch (std::exception & e)
  {
    if (strlen(e.what()) > 0) iddqd::messager->error(e.what());
```

```
    else iddqd::messager->error(
           "One or more unknown errors were encountered during execution.");
    return 0;
  }
  catch (...)
  {
    iddqd::messager->error(
      "One or more unknown errors were encountered during execution.");
    return 0;
  }
}

// end of cognition.cpp
```

Bibliography

[1] A. Alexeychuk, V. Di Gesu, R. Palenichka, C. Valenti. *A fast recursive algorithm to compute local axial moments*, Proc. of Converging Computing Methodologies in Astronomy Conference, Sonthofen, Germany, 1997, http://astro.u-strasbg.fr/ ccma

[2] A. Alexeychuk, R. Palenichka, U. Zscherpel, *Structure-adaptive algorithm of radiographic image segmentation for crack-like defect detection*, Proc. of CM NDT 98, DGZfP, 1998, 259-266.

[3] D. H. Ballard, *Generalizing the Hough Transform to detect arbitrary shapes*, PR-13, 1981, 111-122.

[4] H. H. Barrett, W. Swindell, *Radiological imaging*, Academic Press, 1981.

[5] G. L. Becker, *Radiographic NDT*, Du Pont NDT Systems, E.I. du Pont de Nemours & Co., Inc., 1989.

[6] R. Bellman, S. Dreyfus, *Applied dynamic programming*, Princeton U.Press, Princeton, N.J., 1962.

[7] G. Brast, H.-J. Maier, P. Knoch, U. Mletzko, *Fortschrittsbericht über einen ZfP-Ringversuch an austenitischen Rohrleitungs-Rundschweißnähten (Progress Report on a NDT Round Robin on Austenitic Circumferential Pipe Welds)*, 23. MPA-Seminar, Stuttgart, Oct. 1997, pp. 41.1-42.16.

[8] M. Brejl, M. Sonka, *Medical image segmentation: Automated design of border detection criteria from examples*, Journal of Electronic Imaging, Vol.8(1), 1999, 54-64.

[9] J. B. Burns, A. R. Hanson, E. M. Riseman, *Extracting straight lines*, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-8, No.4, 1986, 425-455.

[10] J. Canny, *A computational approach to edge detection*, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-8, No.6, 1986, 679-698.

[11] C. K. Chow, T. Kaneko, *Boundary detection of radiographic images by a threshold method*, in Proc. of IFIP Congress 71, 130-134.

[12] AR. Cowen, *Digital X-Ray Imaging*, Meas. Sci. Technol. 2, 1991, 691-707.

[13] M. R. Dobie, P. H. Lewis, *Extracting curvilinear features from remotely sensed images using minimum cost path techniquies*, Proc. of the IEEE International Conference on Image Processing - ICIP'94, 1994, 231-235.

[14] W. Doyle, *Operation useful for similarity-invariant pattern recognition*, J. Assoc. Comput. Mach. 9, 1962, 259-267.

[15] R. O. Duda, P. E. Hart, *Use of Hought Transform to detect lines and curves in pictures*, Communication of the ACN, Vol.15, 1972, 11-15.

[16] R. O. Duda, P. E. Hart, *Pattern Classification and Image Analysis*, Wiley, New York, 1973.

[17] M. Erve, U. Wesseling, R. Kilian, R. Hardt, G. Brümmer, V. Maier, U. Ilg, *Cracking in stabilized austenitic stainless steel piping of german boiling water reactors – Characteristic features and root causes*, 20. MPA-Seminar, 1994, Vol.2, paper 29, pp. 29.1-29.21.

[18] G. Fekete, J. O. Eklundth, A. Rosenfeld, *Relaxation: Evaluation and applications*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3, 1981, 460-469.

[19] M. A. T. Figueiredo, J. M. N. Leitão, *Unsupervised contour estimation*, Proc. of the IEEE Int. Conf. on Image Proc. - ICIP'96, Vol.1, 1996, 821-824.

[20] M. A. Fischler, J. M. Tenenbaum, H. C. Wolf, *Detection of roads and linear structures in low-resolution aerial imagery using a multisource knowledge integration technique*, Computer Graphics and Image Processing 15, 1981, 201-223.

[21] D. L. Fried, *Noise in photoemission current*, Appl. Optic, Vol.4, 1965, 79.

[22] J. Goebbels, B. Illerhaus, P. Reimers, *Nondestructive evaluation of the integrity of full scale HLW ingots*, ASME Radioactive Waste Management and Environmental Remediation, 1995, 249-252.

[23] W. E. L. Grimson, D. P. Huttenlocher, *On the sensitivity of the Hough Transform for object recognition*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-12, 1990, 255-274.

[24] P. V. C. Hough, *Method and means for recognizing complex patterns*, U. S. Patent 3069654, Dec. 18, 1962.

[25] M. Hueckel, *An operator which locates edges in digital pictures*, JACM, Vol.18, No.1, Jan. 1971, 113-125.

[26] H. Hwang, R. Haddad, *Multilevel nonlinear filters for edge detection and noise suppression*, IEEE Trans. Signal Proc., Vol.42, No.2, 1994, 249-258.

[27] C. Jacobsen, *Verbesserung der bildverarbeitungsgestützten Rißdetektion an Schweißnahtradiographien mit neuronalen Netzen*, Ph.D. thesis, Christian-Albrechts-Universität, Kiel, 1999.

[28] G. Johannsen, J. Bille, *A threshold selection using information measures*, In Proceedings 6th Int. Conf. Pattern Recognition, Munich, Germany, 1982, 140-143.

[29] J. N. Kapur, P. K. Sahoo, and A. K. C. Wong, *A new method for gray-level picture thresholding using the entropy of the histogram*, Comput. Vision Graphics Image Process. 29, 1985, 273-285.

[30] M. Kass, A. Witkin, D. Terzopoulos, *Snakes: active contour models*, Int. Journal of Comp. Vision, Sept. 1988, 321-331.

[31] J. Kittler, J. Illingworth, *Minimum error thresholding*, PR-19, 1986, 41-47.

[32] V. A. Kovalewsky, *Sequential optimization in pattern recognition and pattern description*, Proc.IFIP Cong., 1968, 146-151.

[33] F. Leymarie, D. Levine, *Tracking deformable objects in the plane using an active contour model* IEEE Trans. Patt. Anal. Mach. Intelligence (PAMI), Vol.15, No.6, 1993, 617-633.

[34] G. Lim, M. Alder, *A nonparametric approach for detecting lines and curves*, Proc. of the IEEE Int. Conf. on Image Proc. - ICIP'96, Vol.1, 1996, 837-840.

[35] A. Martelli, *Edge detection using heuristic search methods*, Computer Graphics and Image Processing, Vol.1, 1972, 169-182.

[36] A. Martelli, *An application of heuristic search methods to edge and contour detection*, Communications of the ACM, Vol.19, No.2, 1976, 73-83.

[37] L. Mérő, *An optimal line following algorithm*, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-3, No.5, 1981, 593-598.

[38] U. Montanari, *On the optimal detection of curves in noisy pictures*, Communications of ACM, Vol.14, No.5, 1971, 335-345.

[39] *MPA-Bericht zum VGB/MPA-Vorhaben 4.2 Zerstörungsfreie Prüfung von austenitischen und Misch-Schweißverbindungen (Entwicklung von Ringversuchen zum Nachweis der Leistungsfähigkeit von Prüfsystemen) 5.Teilbericht : Ergebnisvergleich - Auswertung der Untersuchungsergebnisse der zerstörenden und der zerstörungsfreien Prüfung (Phase IV.3)*, MPA-Berichts-Nr. VGB/MPA-TB 4.2-5, Stuttgart, 1998.

[40] A. Mukherjee, S. K. Parui, B. B. Chaudhuri, R. Krishnan, K. K. Rao, *Detection of linear features in satellite imagery using robust estimation*, Proc. 12-th Int. Conf. Pattern Recog. - ICPR-A(94), 1994, 514-516.

[41] N. J. Nilson, *Problem-solving methods in artificial intelligence*, McGraw-Hill, New York, 1971.

[42] C. Nockermann, H. Heidt, and N. Thomsen, *Reliability in NDT: ROC study of radiographic weld inspections*, NDT&E International 24, 1991, 235-245.

[43] S. I. Olsen, *Estimation of noise in images: an evaluation*, CVGIP: Graphical Models and Image Processing, Vol.55, No.4, 1993, 319-323.

[44] N. Ostu, *A threshold selection method from gray-level histogram*, IEEE Trans. System Man Cybernet. SMC-8, 1978, 62-66.

[45] R. Palenichka, P. Zinterhof, U. Zscherpel, A. Alexeychuk, *Model-based generation of structural statistical hypotheses for flaw detection in radiographic images*, Proc. of Image and Multidimensional Digital Signal Processing 98, IEEE Signal Processing Society, Alpbach (Austria), 1998, 51-54.

[46] P. L. Palmer, H. S. Dabis, J. Kittler, *A performance measure for boundary detection algorithms*, Proc. of International Conference on Pattern Recognition, 1994, 17-21.

[47] T. Peli, D. Malah, *A study of edge detection algorithms*, Computer Graphics and Image Processing 20, 1982, 1-21.

[48] R. Pohle, *Segmentierung and Klassifikation von Schweißnahtlern in radioskopischen Aufnahmen*, Ph.D. thesis, Fakultät für Maschinenbau der Otto-von-Guericke-Universität Magdeburg, 1995.

[49] W. K. Pratt, *Digital Image Processing*, A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York / Chichester / Brisbane / Toronto / Singapore, 1991.

[50] J. M. S. Prewitt, M. L. Mendelsohn, *The analysis of cell images*, In Ann. New York Acad. Sci. Vol.128, 1966, 1035-1053.

[51] T. Pun, *A new method for gray-level picture thresholding using the entropy of the histogram*, Signal Process. 2, 1980, 223-237.

[52] T. Pun, *Entropic thresholding: A new approach*, Comput. Vision Graphics Image Process. 16, 1981, 210-239.

[53] P. Radeva, J. Serrat, E.Martí, *A snake for model-based segmentation*, In Proc. of the Intern. Conf. on Comp. Vision - ICCV'95, 1995, 816-821.

[54] P. Rose, *Bestimmung charakteristischer Fehlermerkmale zur rechnergestützten Bildauswertung von Schweißnahtradiographien*, Ph.D. thesis, Technische Universität Berlin, 1993.

[55] A. Rosenfeld, R. C.Smith, *Thresholding using relaxation*, IEEE Trans. Pattern Anal. Mach. Intell. PAMI-3, 1981, 598-606.

[56] A. Rosenfeld, M. Thurson, *Edge and curve detection for visual scene analysis*, IEEE Trans. Comput., C-20, 1971, 562-569.

[57] A. Rosenfeld, P. De. La Torre, *Histogram concavity analysis as an aid in threshold selection*, IEEE Trans. System Man Cybernet. SMC-13, 1983, 231-235.

[58] P. K. Sahoo, S. Soltani, A. K. Wong, Y. C. Chen, *A survey of thresholding techniques*, Computer Vision Graphics Image Processing 41, 1988, 233-260.

[59] P. V. Sankar, J. Sklansky, *A gestalt-guided heuristic boundary follower for x-ray images of lung nodules*, IEEE Trans. Pattern Anal. Mach. Intell., PAMI-4, No.3, 1982, 326-331.

[60] W. Tsai, *Moment-preserving thresholding: A new approach*, Comput. Vision Graphics Image Process. 29, 1985, 377-393.

[61] J. K. Udupa, S. Samarasekera, W. A. Barrett, *Boundary detection via dynamic programming*, SPIE Vol.1008 Visualization in Biomedical Computing, 1992, 33-38.

[62] P. M. Woodward, *Probability and information theory, with application to radar*, Pergamon Press Ltd, London, 1953.

[63] MJ. Yaffe, JA. Rowlands, *X-Ray Detectors for Digital Radiology*, Phys. Med. Biol. 42, 1995, 1-39.

[64] S. Zucker, R. Hummel, A. Rosenfeld, *An application of relaxation labelling to line and curve enhancement*, IEEE Trans. Comput. C-26, 1977, 394-403.

Acknowledgment

The work presented here is made during my engagement at the Federal Institute for Material Research and Testing (BAM) in Berlin, Division VIII.3 “Non-destructive Testing and Characterization, Radiology”. I want to thank all co-workers who assist me in performing this work. I particularly thank Dr. Uwe Zscherpel, Prof. Claus-Jürgen Wolter and Prof. Hans-Jürgen Ullrich, as my supervisors, as well as Dr. Gerd-Rüdiger Jaenisch for all useful professional advises, support and general help. Special thanks to Dr. Kurt Osterloh for valuable suggestions about structuring and presentation of material.