SOME VISUAL INSPECTION PROBLEMS IN THE BELGIAN INDUSTRY

Patrick Wambacq, Piet Dewaele, Rudi Bartels,
Johan Vandeneede, Dirk Van Den Oudenhoven, André Oosterlinck

Katholieke Universiteit Leuven, Div. ESAT/MI2
Kardinaal Mercierlaan 94 B-3030 Heverlee
BELGIUM

ABSTRACT

In this paper, we will review some visual inspection problems that were presented to our laboratory by a consortium of Belgian industrial companies. These problems were selected to serve as a test vehicle for a software package, called LILY (Leuven Image processing LibrarY), that was developed for the consortium by our laboratory. Therefore, this paper consists of several parts : first, an overview of the LILY software package will be given; then, three case studies carried out with the package will be detailed. The case studies are very different in nature, so that a variety of algorithms will be dealt with. The first case study is about defect inspection in unexposed radiographic film. In this case, image data is presented as a continuous stream of lines of pixels. In the study, convolution techniques, curve fitting methods and Fourier analysis were applied. The second case study treats the inspection of textured textiles and is thus essentially a texture inspection problem. Here, we will introduce a novel approach, whereby self-adaptive convolution filters are constructed from test samples. Both the size and the coefficients of the convolution mask are determined from the texture that is to be inspected. A last case study deals with defect detection in solder joints. Here, we will show how a well adapted illumination can simplify the inspection to a great extent.

INTRODUCTION

Belgium is a country with approx. 10,000,000 inhabitants and with an economic situation that is comparable to that of the average state in the US. These thoughts are give here to stress that, when 13 industrial companies, 5 of which are very large, wanted to start and sponsor a research program on visual inspection, there is a definite interest in using visual inspection techniques in the Belgian industry. In 1983, this group of companies decided to fund, together with I.W.O.N.L., which is a governmental institute to encourage scientific research in industry and agriculture, a large research program for six years. The research topics cover optical methods, hardware development, algorithms and software development, and 3D laser techniques. At the end of the project, about 45 man years of research will be invested, distributed over several universities. Our role in the project is to develop algorithms and to implement them in a comprehensive software package that contains both ready-to-use image processing operators and a toolbox with many routines. 18 man years from the mentioned research project were devoted to the development of the LILY package, as well as an unknown amount of effort from contributors from other projects.

Automatic visual inspection is a topic that has been researched already for a long time. Currently, advanced new algorithms are being developed in many laboratories with the hope that the ever increasing demand for more speed and complexity will be fulfilled. However, we must recognise the fact that the practical implementation in a factory environment will always fall behind of what is feasible in laboratories. This is certainly true for the algorithms that will be presented in this paper, but we will give some comments on the feasibility of practical implementations.

In fact, the original goal of the development of the LILY software package that is used in the case studies, was not to provide practical and readily implementable solutions to visual inspection problems, but rather to give the industrial companies who sponsored the research, an insight in concepts of automatic visual inspection and an idea of what might be expected in the future. As a matter of fact, some of the companies that were involved, already expressed their interest to start new projects based on the present one, for the implementation of algorithms at their factories.

THE LILY PACKAGE

A very important issue at the start of a large software project is the choice of the programming language. An image processing software package should consist of independent modules that look like black boxes, that can be easily combined to build more complex modules or programs, and that can be maintained easily. Therefore the programming language must meet some very important requirements:

- the language should guarantee a high degree of modularity;
- a well designed parameter passing mechanism should be available;
- the programming language should be structured, such that it looks like pseudo code. Readability and understandability will benefit from it.

Furthermore the language should be available on VAX and the software should be transportable to other computer systems. The programming language that met these requirements best was standard Pascal. However, by making use of some VAX Pascal features, like environment files, conformant schemes and logical names, we could improve the modularity and structure of the package greatly. Therefore

it was decided to incorporate these features at the cost of a decrease in portability. The language C, although more transportable, was not chosen because C is much more subject to large differences in programming style between program writers and this would decrease uniformity and readability.

In Lily, all routines and modules belong to certain classes according to their characteristic functions. As such, there is a distinction between:

- *slave routines* : they do not interact directly with the user; argument passing is used as an interface to other routines. Mostly they perform well defined tasks which are not complex. Also, they do not validate the input. Nevertheless they are the most important ones, especially because algorithms are embedded in slave routines. Another very important aspect about slave routines is that they should be applicable in many different contexts, which means that they must not rely on machine or device dependent factors.

- *control routines* : they use the slave routines as elementary building blocks in performing a more complex task. They coordinate the dataflow between the slave routines and are responsible for the correctness of the arguments that are passed to a slave routine. They mostly use the terminal to interact with the user and to let him control the program flow. Every time a variable is asked from the terminal or changed for some reason, its validity is checked. These routines don't do the actual number crunching themselves but delegate this task to slave routines.

Apart from this level of decision taking, another distinction is made. Indeed, the routines are subdivided into three categories according to their purpose:

- *general purpose routines* : these don't work on images or vectors as defined in LILY. They constitute an extension to the set of predefined Pascal routines, and perform general tasks like asking an integer constant from a terminal or deleting trailing blanks in a string.

- *image processing routines* : these work on images, vectors and other data types defined in Lily. Image processing algorithms are implemented in the slave routines of this class. In Table 1 an overview is given of algorithms that have been implemented.

- *I/O routines* : they perform an image or vector transfer between the computer's main memory and a peripheral device, like a disk, a terminal or a peripheral (image) computer. They are device independent, in that they perform their task by calling routines from lower level device subpackages.

Above all these classes, another level exists: master routines. These routines may combine all other types of routines arbitrarily. They contain declarations of the data structures that are passed as conformant arrays throughout the other routines. The most important of these data structures are of course the images, which are given actual sizes within master routines. The development of any software environment requires the definition of data structures. In an image processing environment the choice of appropriate data types and structures is of high importance because very large amounts of data are used in most applications. The data representation possibilities also have great influence on the efficiency of the package. Therefore, a lot of attention was paid to the definition of appropriate data types for the LILY package. Besides the standard Pascal data types, we defined the following :

- image data type, containing the pixels in a one-dimensional vector. Pixels may have standard Pascal data types. The type "image" also contains the image dimensions. The reasons for storing an image in a one-dimensional rather than in a two-dimensional array are :
  - compactness;
  - looping through an image requires only one index;
  - mixed-language programming is easier.
- vector data type : may contain histograms, filter shapes, feature lists, etc.
- window and border data types : determine rectangular regions in an image.

Since the package consists of a large amount of routines, a standard for passing data from a program or routine to a LILY-routine has been established. Images and vectors are always passed by a pointer referring to them, which means that array bounds are determined at the time routines are entered. Global variables are not allowed, because of the side-effects they may produce.

Further, a standardisation of the nomenclature is pursued, allowing a user to distinguish easily between routines. To guarantee usefullness of the package, a clear documentation of every module is provided; strict rules must be followed in writing documentation, which has a standard layout so that it is compatible with the standard VMS help facility. This layout is produced automatically by a utility program. Other utility programs exist to facilitate the development of new routines and the maintenance of the package, such as testprogram generators, standards checkers, librarians, etc. A more detailed description of LILY can be found in [4].

DEFECT INSPECTION IN UNEXPOSED RADIOGRAPHIC FILM

To have a good understanding of the nature of the defects, it it best to have a closer look at the production process of radiographic film sheets. In figure 1, this process is depicted. After the substrate is made, it is covered with the X-ray sensitive emulsion as it moves by a casting opening. The moulding operation is the main defect source. After drying, the film is cut in sheets and inspected. The film sheets are scanned with a laser beam in a direction perpendicular to the casting direction. This yields images of 2484x2048x12 bits. The defects can be classified in two groups :

- point defects originating from local irregularities in the substrate or a local thickening or thinning of the emulsion;

Table 1: Overview of most important image processing routines in LILY.

Image enhancement
Contrast enhancement by rescaling, histogram equalisation, unsharp masking; binary and greylevel erosion and dilation; increasing and decreasing resolution and resampling.

Contour coding and manipulation
Contour coding; conversion of one contour description to another; closing of contours; direction of normal to contour; contour enhancement by minimum cost path; drawing of contours, curves, hull curves, ellipses, lines.

Filtering
Filtering in spatial domain (convolutions); autocorrelation and correlation; mean, absolute mean and variance images; non linear filtering; moving average; edge detection; thinning algorithms; one and two dimensional complex Fourier transform; filtering in frequency domain; real power and phase spectrum representations; filters separable in polar coordinates; filters separable in x and y coordinates.

Feature extraction
Histogram calculation; Hough transform; profiles; mean and variance; moments around main axes; all kinds of features in small windows of an image (average difference between pixels, statistical correlation between pixels, standard deviation); all kinds of features of objects based on their contour (area, convex hull, enclosing window, main moments, mean radius, perimeter); texture features based on the cooccurrence matrix, greylevel difference vector and greylevel runlength matrix; variance covariance matrix of an image; correction of moments for camera distortions; determinant and inverse of matrix; curve fitting.

Image segmentation
Automatic threshold computation; image and histogram clipping; slicing; labeling of 8 connected objects; labeling of runlengths; segmentation on the basis of contrast and/or homogeneity measures.

Pyramids
Several methods for pyramid construction; routines for projection of windows from one level to another level in the pyramid; computation of the dimension of a certain level; image features extracted by means of pyramids.

Relaxation
Relaxation labeling with relaxation scheme of Peleg or Rosenfeld.

Image and vector filling
Filling of vector masks for one dimensional filtering; filling of parts of an image or vectors with appropriate constants.

Image and vector copy routines
Data type conversion, concatenation, ...

Window operations

Arithmetical and logical operations

- streak or smear errors coming from the casting process. These errors propagate in the moulding direction and can extend over long distances. Typically, they have a low amplitude.

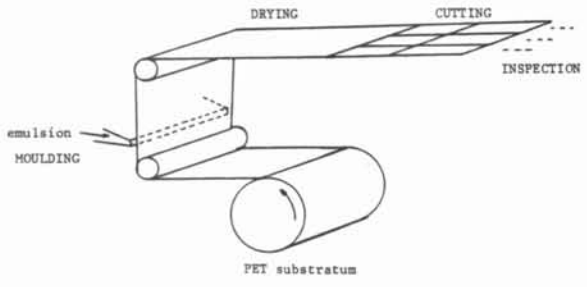

Figure 1: Production process of radiographic film.

Figures 2 and 3 show a 3D plot of the images of two types of defects and figure 4 shows the greyvalues along one scan line through the smear defect. It shows a noisy background variation on which smear errors are superimposed as peaks. Unfortunately, this background variation also exists in the absence of errors. It is also clear from the figure that simple thresholding techniques cannot lead to a good separation of the errors from the background.

Because the films are scanned in a linear fashion, the accent will lie on linear detection methods. Three techniques were adopted : one-dimensional convolution, polynomial fitting and Fourier domain filtering.

One-dimensional convolution

The idea here is to apply a filter that will detect large changes in grey value over a short distance, i.e. some kind of differentiation. After a number of experiments, we found that a combination of a smoothing mask and a gradient mask gave the best results :

$$\frac{1}{9}\,[1\,1\,1\,1\,1\,1\,1\,1\,1] * [-1\,0\,0\,0\,0\,0\,0\,0\,1]$$

The sizes of the masks are adapted to the scale at which the defects present themselves. The result of applying these masks can be seen in figure 5. A suitable threshold now allows a reliable defect detection. In [1] a justification from a theoretical point of view is given for the choice of the

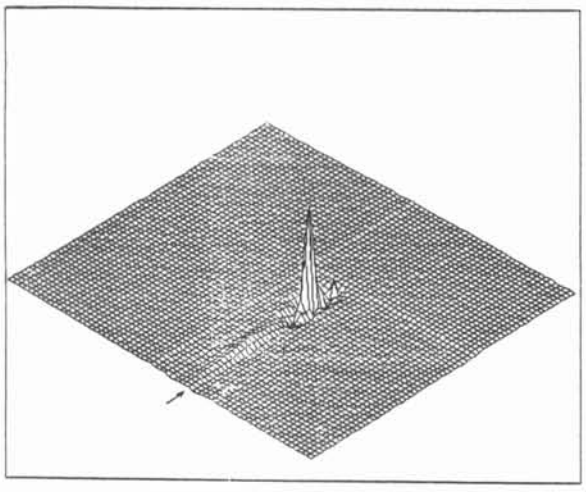

Figure 2: 3D plot of image with point defect.

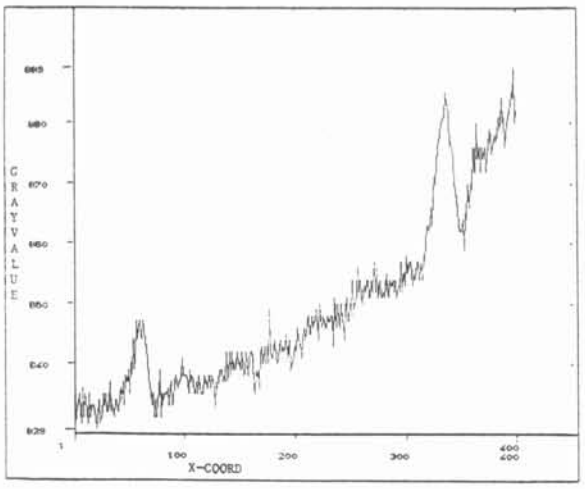

Figure 4: One scan line of image in fig. 3.

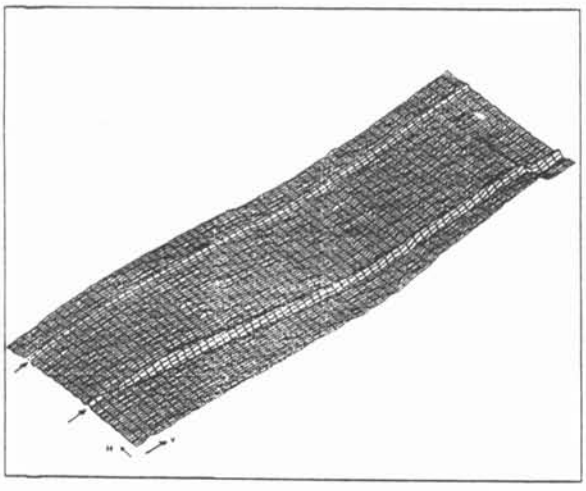

Figure 3: 3D plot of image with smear defects.

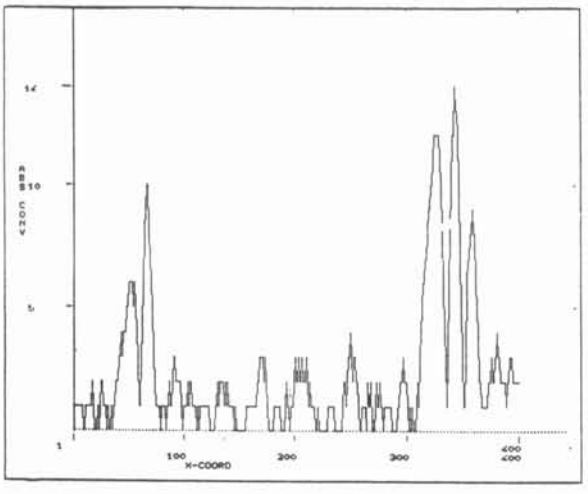

Figure 5: Result of 1D convolution.

masks.

Polynomial fitting

From figure 4 it can be observed that a polynomial fit through the data would give an approximation to the background; the magnitude of the fit error would be largest at the location of the defects. This is based on the assumption that the physical size of the error is small compared to the width of the scan line, and therefore that the error itself does not affect significantly the parameters of the polynomial fit. In our case this poses no problems. Figure 6 shows a scan line and its second order polynomial fit. The required order of the fit can be obtained by computing a series of fits with successively higher order. The standard deviation of the differences between the fit and the actual scan line values rapidly decreases and stabilizes. From that point on it is of no use to try higher orders. Empirically, we found that a second or third order fit gives the best results in all cases. After subtracting the fit from the original, a simple threshold operation is used to locate the errors. Figure 7 shows the result of this operation. One advantage of this method is that here the centre of the defects is found, whereas the previous method indicates the edge transitions.

Fourier domain filtering

The polynomial fit to the scan lines can be used to design a filter that will suppress all background information and amplify the contribution of the defect to the signal. To achieve this, the fit is subtracted from the original signal and the Fourier spectrum of the residue is calculated. The resulting spectrum consists mainly of contributions from the defect, and to a smaller extent from high frequency noise. From this spectrum, a filter can be designed that will extract the desired signal, i.e. the defect, from the original signal. The filter that we use has a very frequency selective bandpass characteristic and a finite impulse response with very small side lobes. The result from applying this filter is

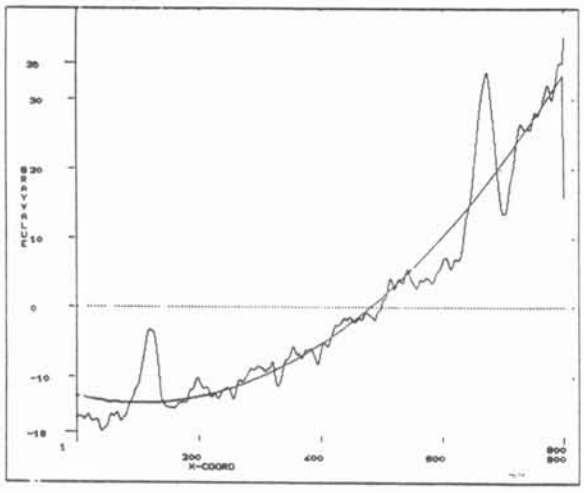

Figure 6: Scan line and fitted polynomial.

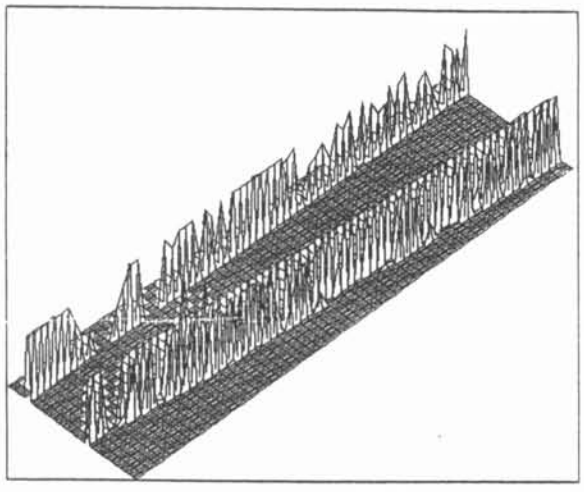

Figure 7: Defect location with polynomial fit method.

shown in figure 8. As can be seen, the edges of the defect can be extracted easily.

A comparison of the three methods reveils that all three of them seem to detect equally well the defects. The convolution method and the Fourier domain filtering basically locate the edges of the defect, whereas the polynomial fit finds the center part of it. All three are line oriented, so that they can be used for continuous production processes, like in this case study. There is however a difference in speed :

- the convolution is very simple to implement in hardware, requires only additions of integer values and can probably work as fast as the image sensor delivers the data;
- the polynomial fit requires the calculation of sums of products and the solution of a set of equations, which will certainly be harder to do in real time, and will also require more hardware;

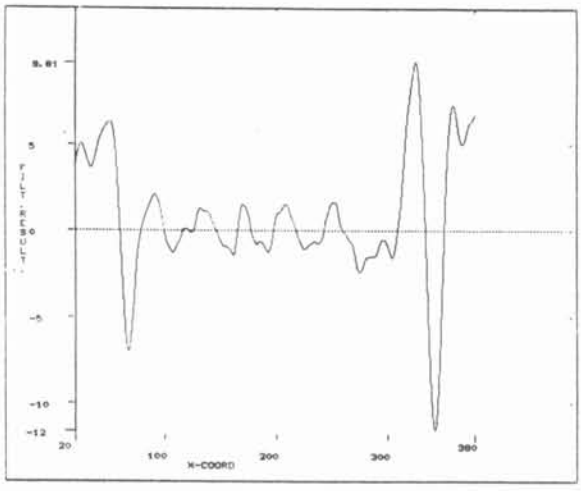

Figure 8: Result of bandpass filter.

- Fourier domain filtering also requires more hardware and will be slower than the first method : FFT's must be calculated, as well as the filtering itself in the frequency domain. The filter that we designed could be applied in the spatial domain, but has a larger impulse response than the simple convolution filter and has also coefficients differing from 2^n. After some refinements however, a suitable impulse response could be derived, but it certainly won't be more efficient than the simple filter.

We can conclude by stating that the first method provides the most efficient solution, and that its speed is high enough to enable on line inspection. The problem of inspection of unexposed radiographic film is treated in much more detail in [1].

INSPECTION OF TEXTURED TEXTILES

For the inspection of textured textiles, a new algorithm was developed. Both the learning phase and the inspection phase run completely unassisted. Starting from a good sample, the required convolution filter set is computed. These filters are then applied to all succeeding samples. A feature extraction is performed on these results, whereafter a Mahalanobis classifier gives the final results. These steps will now be elaborated in some more detail. A complete description of the new algorithm is given in [2], including also a large bibliography.

To make the algorithm less sensitive to varying illumination and acquistion conditions, a preprocessing step is included. The output image is calculated from the input image as follows :

$$z_{k,l} = \frac{x_{k,l} - \bar{x}_{k,l}}{\sigma_{k,l}}$$

where $z_{k,l}$ and $x_{k,l}$ are the output and input images, $\bar{x}_{k,l}$ is a moving average and $\sigma_{k,l}$ is a moving standard deviation.

The next step is to estimate the period of the texture. There are several ways to this; we decided to maximise the correlation that exists between two windows in the image

that are translated. The highest correlation is found when the distance of the translation equals the texture period : $d_a(k', l')$ is then minimal.

$$d_a(k', l') = \frac{1}{N^2} \sum_{k=1}^{N} \sum_{l=1}^{N} | t_{k,l} - t_{k+k', l+l'} |$$

In fact, periodicity has to be determined in every direction where it exists; in our case there are two such directions. This leads to a "periodicity parallelogram".

Then the size of the filter masks is determined from this parallelogram, in fact they are as large as the parallelogram itself. All coefficients are zero except on the begin and end values of the periodicity vector and on the local d_a extremum in between. The filters typically look like this :

X	0	0	X	0	0	X
0	0	0	0	0	0	0
X	0	0	X	0	0	X
0	0	0	0	0	0	0
X	0	0	X	0	0	X

where 0 and X stand for a zero and non-zero coefficient respectively.

After having determined the size of the filter and the position of the non-zero coefficients, the value of these coefficients still remains to be computed. This is done by an eigenfilter technique, developed by Ade [3]. The grey values within the filter mask are put into a vector and the variance-covariance matrix of this vector is computed. The eigenvectors of this matrix yield the filter coefficients.

At this stage, a set of filters is obtained that are specifically adapted to the texture that will be inspected. This process is carried out once for every new texture; all succeeding samples are processed with the same filters.

The absolute values of the images filtered with the sparse convolution masks are then averaged over a window with a size that is a multiple of the periodicity; this gives energy measures that are then fed into a classifier, e.g. a minimum distance Mahalanobis classifier.

Some results of this algorithm are shown in the figures. Figure 9 shows an original image with a line defect. Figure 10 shows the result of the classifier, and figure 11 shows a thresholded result. Another example of point defects is given in figures 12, 13, 14. As can be seen from these figures, all defects are detected very well. Experiments showed that this method outperforms other existing methods, meaning that where other methods fail, this one still gives good results.

When this algorithm has to be implemented in a working system for the factory floor, some refinements will be necessary. The computation of the filter masks doesn't need any changes because it can be done offline. The convolutions and further processing have to be done for every sample and need a closer look. The most time consuming part there is the Mahalanobis classifier, and at the moment we are investigating the use of other, less computation intensive possibilities, e.g. a weighted Euclidian classifier. For reasons of speed, all calculations should be performed on integers, which means that the filter coefficients will have to be quantized. The effect of such a quantization on the accuracy still has to be studied. If however this turns out not to deteriorate the results, a reasonably fast implementation of e.g. 20 Mpixels/second could be achieved.

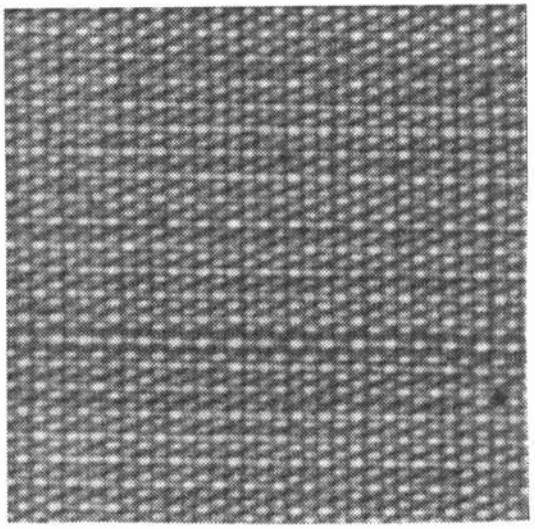

Figure 9: Original image with line defect.

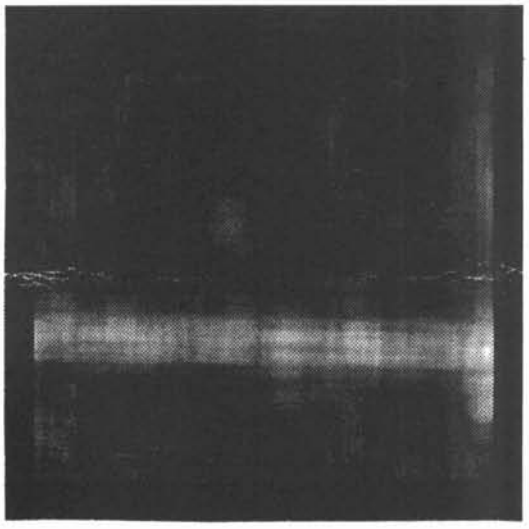

Figure 10: Result of classifier.

DEFECT DETECTION IN SOLDER JOINTS

Solder joints are subject to a large number of possible defects. Some experiments at our lab showed that a good classification of all these possibilities would require a very large effort and would suffer from a bad accuracy. Therefore, we adopted a rather pragmatic approach : a joint is either good or bad; if it is good than all is well; if it is bad than it does not matter too much whether this in fact is a non wetting situation, or whether there is too much solder or anything else : it will be rejected anyway and possibly manually corrected. If our technique is able to distinguish between several classes, then that comes in handy, but it is not essential. Of course this approach is questionable, but at least it works within its limitations.

The key to a good distinction between good and bad solder joints is here the appropriate illumination. Our setup is shown in figure 15. A circular fluorescent tube is used as

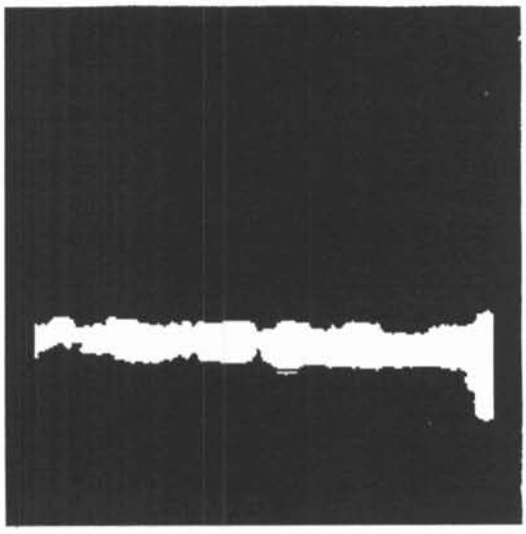

Figure 11: Thresholded version of fig. 10.

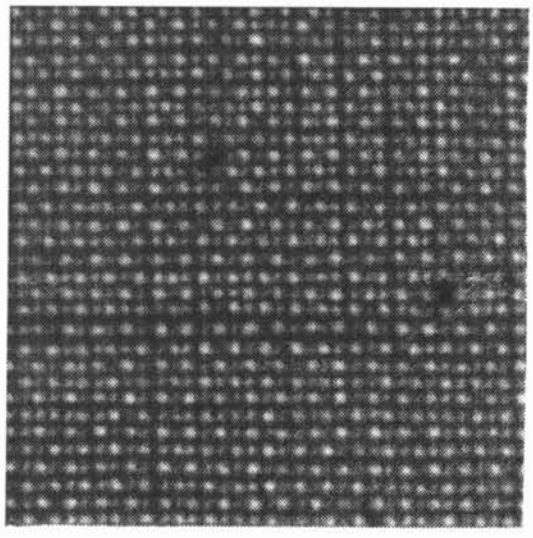

Figure 12: Original image with point defect.

a light source. A diffusor is used to provide a more uniform illumination. A piece of black cloth acts as a shield. A small opening in the cloth makes sure that the incident light rays are more or less perpendicular to the printed circuit board. The amount of shielding determines the angle of a slanted plane that is still observed as illuminated (see figure 16). The solder joints are presented one by one under the camera; their exact position is known from a CAD/CAM database. Good solder joints are observed like in figure 17. Joints without or with too little solder are seen as in figures 18 and 19. Because of the effect explained in figure 16, good joints are observed as being black, whereas bad joints show a white area. The images from the joints are then averaged using two different masks, see figure 20. These average values are used to classify the solder joints:

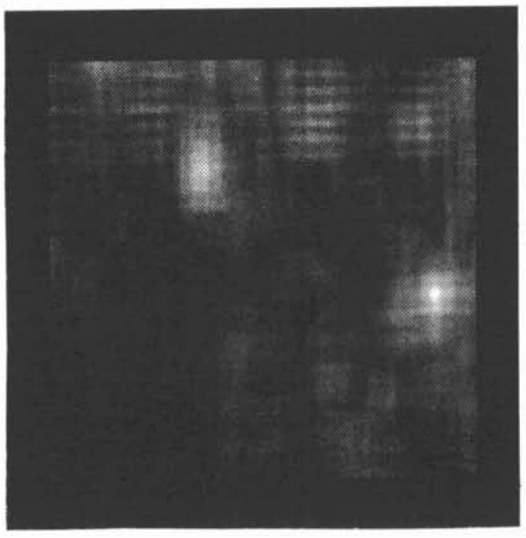

Figure 13: Result of classifier.

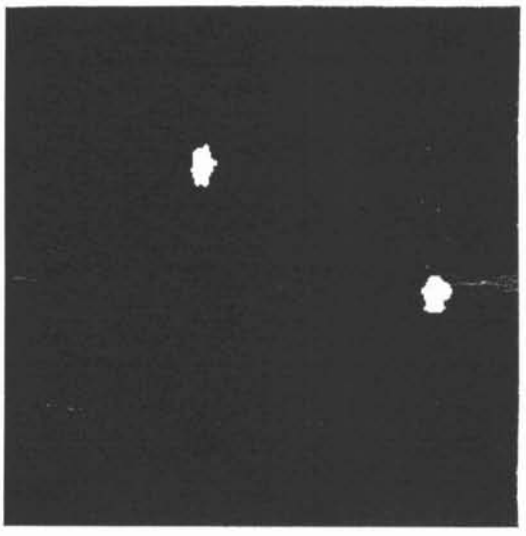

Figure 14: Thresholded version of fig. 13.

Name	average1	average2
Good joints (class 1)	low	low
No solder (class 2)	low	high
Too little solder (class 3)	high	high

The results of the classification of 417 solder joints, taken from a single PCB, are summarized in the following confusion table:

	classified as		
belongs to	class 1	class 2	class 3
class 1	371	0	7
class 2	0	18	0
class 3	7	0	14

The table shows that all joints without or with very little solder (class 2) were classified correctly. About one third of the joints from class 3 were classified as being good, and 7 good joints (class 1) were classified as having too little

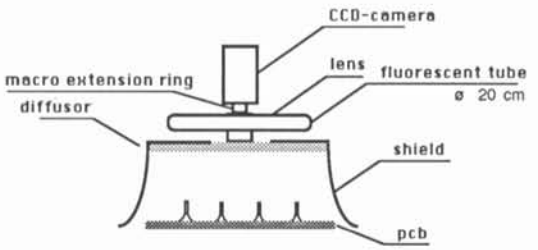

Figure 15: Illumination setup for solder joint inspection.

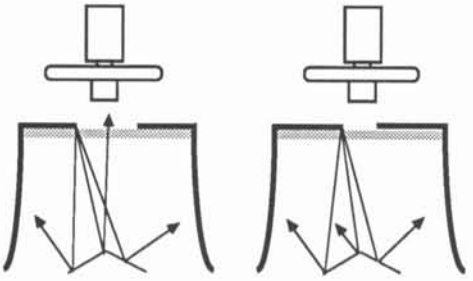

Figure 16: Effect of shielding.

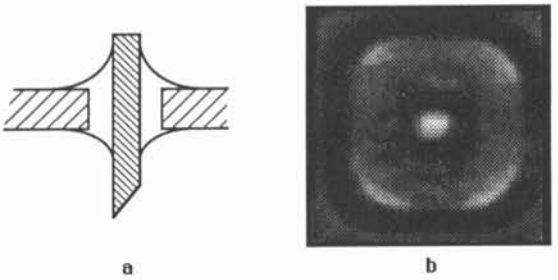

Figure 17: Good solder joint.

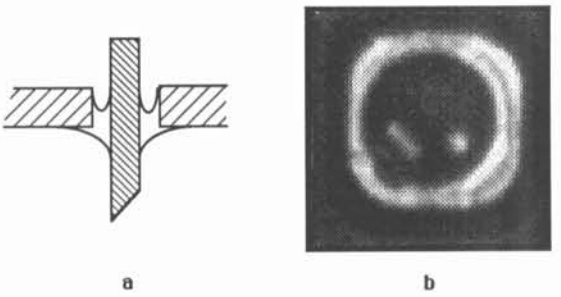

Figure 18: Joint without solder.

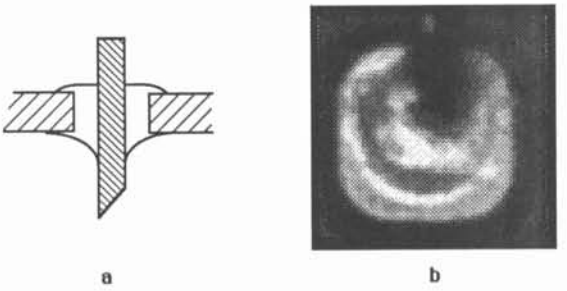

Figure 19: Joint with too little solder.

solder. These results have to be confirmed with larger experiments: only 7 joints from class 2 and 21 from class 3 were present on the PCB so that no good conclusion can be drawn from these figures. It seems however that our approach will need some refinements: the shielding is rather critical and could be improved, and also other masks could be used to provide a better feature extraction. This will be needed anyway if craters, cracks and other defects must be recognised and classified. The implementation in LILY requires about 1 second of computation time per joint, but the speed can be increased by a factor of five if some optimizations are carried out, and by several orders of magnitude if special hardware is built. The current algorithm does not require much hardware: the only things that need to be done are averaging, thresholding the features and make a decision.

CONCLUSION

In this paper we have introduced the LILY software package. It was developed with funding from Belgian industrial companies and from the government. Its main goal was to provide a comprehensive set of programs and routines for all kinds of visual inspection problems. Implementations for practical problems in the factory were not envisaged. However, some of these practical problems were used as a case study to evaluate the quality of the algorithms that were implemented. Three of these case studies were discussed here, showing that a variety of applications can be tackled with the LILY package. Although the results of the case studies prove that the algorithms are adequate, some more work has to be done when on line inspection in the factories is planned.

ACKNOWLEDGEMENTS

The authors wish to thank the I.W.O.N.L. (Institute for the encouragement of scientific research in industry and agriculture) and all cooperating companies for their support.

REFERENCES

[1] P. Dewaele, P. Wambacq, A. Oosterlinck, *An application of the LILY software package to defect inspection in unexposed radiographic film*, submitted to the 6th Meeting of Optical Engineering, dec. 1988, Tel Aviv, Israel.

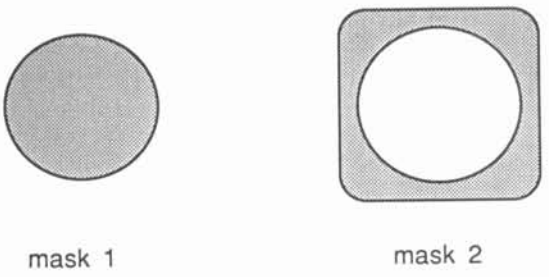

Figure 20: Masks used for averaging.

[2] P. Dewaele, L. Van Gool, P. Wambacq, A. Oosterlinck, *Texture inspection with self-adaptive convolution filters*, accepted for 9th ICPR.

[3] Ade, F., *Characterization of textures by Eigenfilter*, Signal Processing, Vol. 5, pp. 451-457, 1983.

[4] P. Dewaele, D. Van Den Oudenhoven, J. Vandeneede, R. Bartels, P. Wambacq, A. Oosterlinck, *LILY: A Software Package for Image Processing*, Pattern Recognition in Practice III, May 18-20, 1988, Amsterdam.