ISSN 1392 – 124X INFORMATION TECHNOLOGY AND CONTROL, 2008, Vol. 37, No. 3

IMAGE ANALYSIS PROBLEMS IN AOI SYSTEMS

Egidijus Paliulis

Department of Information Technologies, Faculty of Technology, Šiauliai University
Vilniaus St. 141, LT – 76353 Šiauliai, Lithuania

Raimondas Zemblys, Gintautas Daunys

Department of Electronics, Faculty of Technology, Šiauliai University
Vilniaus St. 141, LT – 76353 Šiauliai, Lithuania

Abstract. Historically the electronics manufacturing market has relied on a combination of human visual inspection and electrical test methods to ensure product quality. With the advent of the personal computer, the use of "machine vision" in industrial applications gradually became more common. The process where optical sensors (i.e., cameras) are used to make specific pass/fail decisions is usually described as Automated Optical Inspection (AOI).

There are discussed problems of designing AOI system in this paper. The main goal is to select most efficient image analysis algorithm and to study other parameters that have impact on designing a reliable AOI system.

Keywords: Automated optical inspection, AOI, image analysis, image recognition, computer vision.

1. Introduction

Automated Optical Inspection (AOI) systems were introduced to the electronics manufacturing industry during the 1980s, with the hope that they would be a more consistent replacement for human inspectors who had limited overall inspection effectiveness. Early adopters of these systems were usually disappointed with the speed, effectiveness, ease of use and cost of ownership of these systems and their acceptance was very limited. [1]

There were several reasons that AOI systems became more and more popular:

- Predominance of SMT (Surface Mount Technology);
- Increasing Circuit Complexity and Density;
- Improvements in Key Technology Components of AOI systems;
- Integration of key technologies into stable systems.

There are several potential areas where AOI systems can be used on a typical SMT production line:

1. Post-Print-Paste Inspection. Standard camera-based AOI systems can usually inspect for the presence or absence of paste on pads and problems with print offsets.
2. Post-Placement Inspection. In this phase typically defects from the component placement process such as component presence/absence, skew, tomb stoning, polarity and in some cases text marking on the component body are inspected.
3. Post-Reflow Inspection inspects for many of the same component level defects as post-placement inspection as well as visible solder-level defects.

2. Situation overview

Typical AOI system consists of 4 key elements that are lighting source, camera system/optics, image processor (computer) and programming methodology (Figure 1).

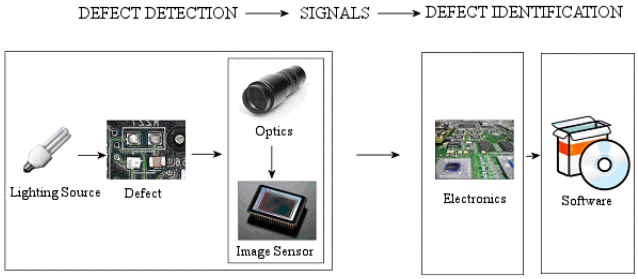

Figure 1. Operations of typical AOI system

AOI system gets an image of object that is illuminated by some light source, transfers image to computer where it is analyzed. The software is the main key for making reliable pass/fail decision. The problem is to select an image analysis and recognition

algorithm that meets requirements of AOI system – processing speed, reliability, cost, etc.

The lighting source can be a monochrome LED or white light or a grouping of these lights in a structure. Structured lighting, sometimes with color, is also used in some systems.

CCD array and line scan cameras predominate in AOI systems. Some systems rely on multiple cameras, or move camera along object's surface to capture separate segments of object (Figure 2).

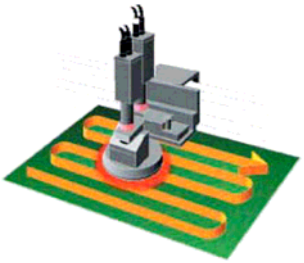

Figure 2. Camera moving along PCB surface

Here are a lot of techniques used in image recognition systems. Some of them are [2]:

- Shape-based Matching;
- 1D Metrology – Measuring;
- 2D Metrology – Subpixel Edge Detection;
- 3D Metrology – Stereo Vision;
- Blob Analysis;
- Morphology;
- 3D Camera Calibration;
- Bar Code & Data Code Reading;
- OCR & OCV (multilayer perceptron neural network).

These techniques may rely on traditional image analysis methods or use more complex ways for making decisions.

3. The aim and tasks of the work

The aim of this work is to analyze traditional image analysis methods and select most efficient for use in AOI systems.

There are some methods used in image processing [3]: edge detection, blob detection, corner detection, ridge detection, feature detection.

We selected to analyze sobel, canny (edge detectors), laplace (blob detector), eigen decomposition, corner detect and the Harris corner detection algorithms.

Also other parameters such as image segmentation size, pass/fail threshold (correlation coefficient), impact of illumination and image acquisition settings will be analyzed.

4. Method

With reference to AOI system there was designed an electronic board testing prototype. It consists of object holder, high resolution CANON photo camera placed in front of it, and a personal computer with image analysis and camera control software. Software remotely triggers camera to take image of object, then the image is transferred to host computer and analyzed (Figure 3).

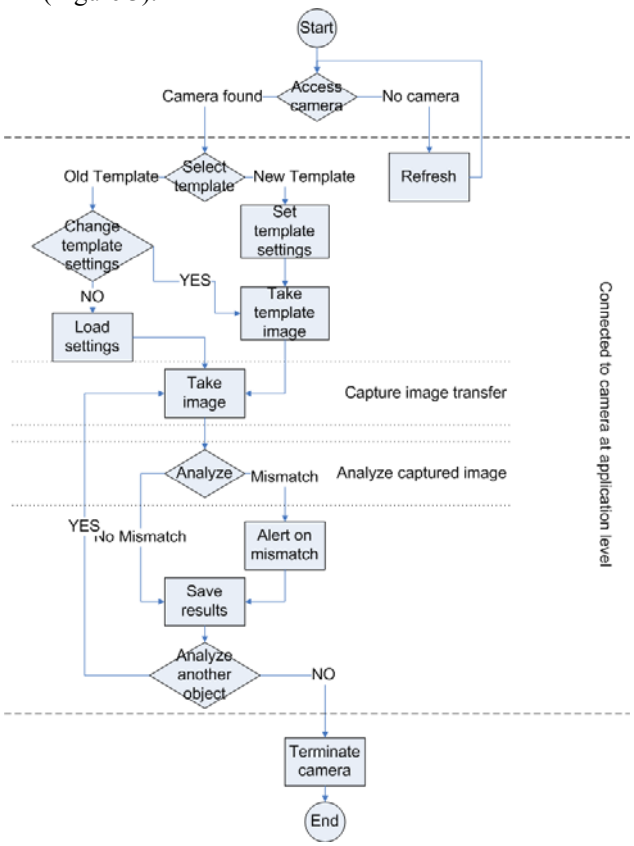

Figure 3. Scheme of the system

There was designed the fallowing algorithm to evaluate assembling quality of printed circuit boards (Figure 4):

1. Get image of "golden" PCB (object without any defects).
2. Divide "golden" image (template) into separate segments.
3. Process template image using selected image processing method.
4. Get image of PCB to test.
5. Divide image of object into separate segments.
6. Process image using same image processing method as template image was processed.
7. Calculate correlation between template segments and corresponding segments of object image.
8. Evaluate assembling quality.

Segments that satisfy selected pass/fail threshold (correlation coefficient) are marked green in output. "Wrong" segments are marked red, so the system operator knows the place of defect if it is present.

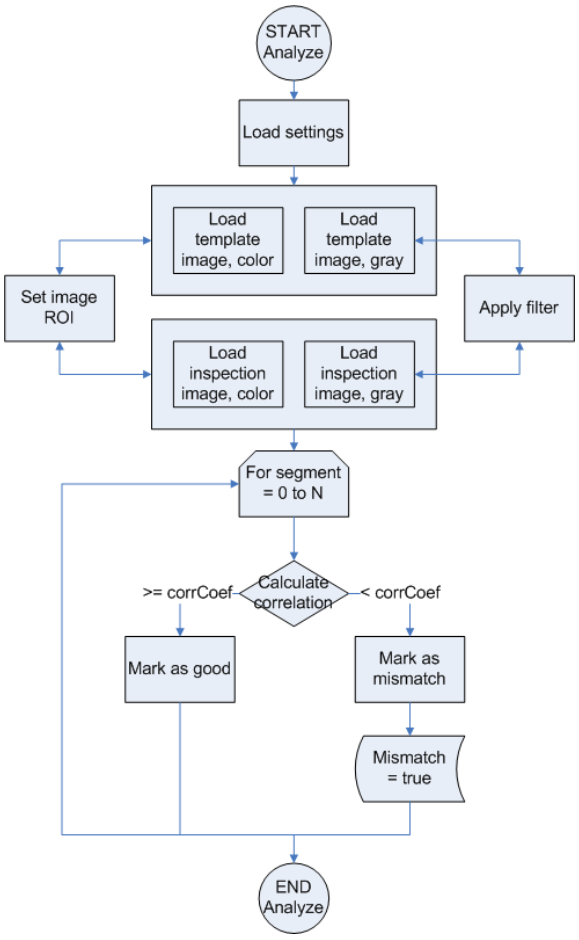

Figure 4. Image analysis scheme

As mentioned above, there were selected 6 image processing algorithms. They were realized using OpenCV [4] library and wxDev-C++ programming environment [5,6].

Sobel operator [7] calculates the gradient of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define **A** as the source image, and $\mathbf{G}_x$ and $\mathbf{G}_y$ are two images which at each point contain the horizontal and vertical derivative approximations of **A** image. The estimation $\mathbf{G}_x$ and $\mathbf{G}_y$ are as follow:

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and}$$

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}' \qquad (1)$$

where "*"denotes the 2-dimensional convolution operation.

At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}} \,. \qquad (2)$$

Canny edge detection operator [8] uses a multi-stage algorithm to detect a wide range of edges in images. It contains a number of adjustable parameters (size of Gaussian filter, thresholds), which can affect the computation time and effectiveness of the algorithm. The stages are:

- *Noise reduction.* In this stage image is convolved with a Gaussian filter to get an image which is not affected by a single noisy pixel to any significant degree.
- *Finding the intensity gradient of the image*. Canny algorithm uses four filters to detect horizontal, vertical and diagonal edges in the blurred image. The edge detection operator (Roberts, Prewitt, Sobel for example) returns a value for the first derivative in the horizontal direction ($\mathbf{G}_y$) and the vertical direction ($\mathbf{G}_x$).
- *Non-maximum suppression*. A search is carried out to determine if the gradient magnitude assumes a local maximum in the gradient direction. This is worked out by passing a 3x3 grid over the intensity map and a set of edge points, in the form of a binary image, is obtained.
- *Tracing edges through the image and hysteresis thresholding.* It requires two thresholds – high and low. Applying a high threshold marks out the edges that can be fairly genuine. Starting from these, using the directional information derived earlier, edges can be traced through the image. While tracing an edge, the lower threshold is applied that allows tracing faint sections of edges as long as a starting point is found.

One of the first and also most common blob detectors is based on the Laplacian of the Gaussian (LoG) [9]. Blob detection refers to visual modules that are aimed at detecting points and/or regions in the image that are either brighter or darker than the surrounding.

The Laplace operator [10] is a second order differential operator in the *n*-dimensional Euclidean space, defined as the divergence of the gradient. Thus if *f* is a twice-differentiable real-valued function, then the Laplacian of *f* is defined by:

$$\Delta f = \nabla^2 f = \nabla \cdot \nabla f \quad . \qquad (3)$$

Equivalently, the Laplacian of *f* is the sum of all the unmixed second partial derivatives in the Cartesian coordinates x_i:

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2} \quad . \qquad (4)$$

In the mathematical discipline of linear algebra, eigen decomposition [11] is the factorization of a matrix into a canonical form, whereby the matrix is

represented in terms of its eigenvalues and eigenvectors. We use calculated eigenvalues [11] to determine if there are differences between images. The calculation is performed using OpenCV function *cvCornerEigenMinVal*.

Corner detection [12] or the more general terminology *interest point detection* is an approach used within computer vision systems to extract certain kinds of features and infer the contents of an image. The most simplified approach used in this work is applying OpenCV function *cvPreCornerDetect* for image [12]. It calculates function:

$$\mathbf{D}_x^2\mathbf{D}_{yy} + \mathbf{D}_y^2\mathbf{D}_{xx} - 2\mathbf{D}_x\mathbf{D}_y\mathbf{D}_{xy}, \quad (5)$$

where $\mathbf{D}_?$ denotes one of the first image derivatives and $\mathbf{D}_{??}$ denotes a second image derivative. The corners can be found as local maximums of the function.

Another approach used is *Harris corner detection algorithm*. Image is given by I; consider taking an image patch over the area (u,v) and shifting it by (x,y). The weighted *sum of square difference* between these two patches, denoted S, is given by:

$$\mathbf{S}(x,y) = \sum_u \sum_v w(u,v)(I(u,v) - I(u-x, v-y))^2 . \quad (6)$$

The Harris matrix **A** is found by approximating **S** with a second order Taylor series expansion.

$$\mathbf{S}(x,y) \approx \mathbf{S}(0,0) + (x,y)\nabla\mathbf{S} + \frac{1}{2}(x,y)\mathbf{A}\begin{pmatrix} x \\ y \end{pmatrix}, \quad (7)$$

where $\nabla\mathbf{S}$ and **A** denote the gradient vector and the Hessian matrix (second derivatives) of **S**.

A corner (or in general an interest point) is characterized by a large variation of **S** in all directions of the vector (x, y). By analyzing the eigenvalues of **A**, this characterization can be expressed in the following way: **A** should have two "large" eigenvalues for an interest point. Based on the magnitudes of the eigenvalues, the following inferences can be made based on this argument:

If $\lambda_1 \approx 0$ and $\lambda_2 \approx 0$, then there are no features of interest at this pixel (x,y).

If $\lambda_1 \approx 0$ and λ_2 is some large positive values, then an edge is found.

If λ_1 and λ_2 are both large, distinct positive values, then a corner is found.

Harris and Stephens note that exact computation of the eigenvalues is computationally expensive, since it requires the computation of a square root, and instead suggest the following function M_c, where κ is a tunable sensitivity parameter:

$$M_c = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 = \det(A) - k \cdot trace^2(A). \quad (8)$$

The value of κ has to be determined empirically, and in the literature values in the range 0.04 - 0.15 have been reported as feasible.

To compare corresponding segments of processed images normalized correlation [11] is used. It is expressed like this:

$$\gamma(u,v) = \frac{\sum_x \sum_y [f(x,y) - \overline{f}_{u,v}][t(x-u, y-v) - \overline{t}]}{\sqrt{\sum_x \sum_y [f(x,y) - \overline{f}_{u,v}]^2 \sum_x \sum_y [t(x-u, y-v) - \overline{t}]^2}} . \quad (9)$$

Here $f(x,y)$ and $t(x,y)$ – distribution of intensity of an image, u, v – offset of pixels, $\overline{f}$, $\overline{t}$ – average of intensity of an image.

The experiments were implemented in this way:

- The object (PCB – printed circuit board) is placed in holders.
- Camera remotely takes 2 images. At first we simulating a reference image of an object and the second – image of an object without defects.
- The object is removed from holders, some parts (resistors, capacitors, etc.) are removed from the board, and the object is placed in holders again.
- Camera remotely takes image of the changed object. It will simulate PCB with defects.
- Software analyses images, using different algorithms and settings (correlation coefficient, segment size, settings of algorithm).

Also different image shooting conditions were tested (illumination, camera settings).

5. Results

Considering that tested image processing algorithms use gray scale images, hypothesis that camera should be set to take gray scale images instead of taking color images, and then converting to gray scale, was tested.

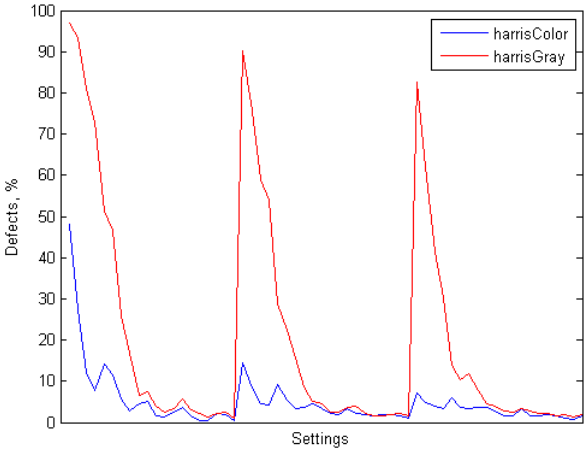

Figure 5. Defect found in gray scale and colour images

In Figure 5, the X axis represents different algorithm settings, while Y axis shows the number of defects found in percents that are calculated this way:

$$rez = \frac{(def - def\_err)}{segx * segy} * 100, \quad (10)$$

where *def* represents the number of segments identified as faulty, *def_error* – the number of segments that represent real defects, *segx* and *segy* – the number of segments in horizontal and vertical directions, respectively. Other settings used in this experiment are as follows:

correlation coefficient – 0.9;
segment size – 50x50 px;
processing algorithm – Harris corner detector.

As seen in Figure 5, we got better results analyzing an image that originally was non-gray scale. This is because selected camera (CANON EOS 400D) still produces image with 3 color channels though it is set to take gray scale image. There is still necessary to convert image to 1 color channel image using software.

As image processing was implemented using openCV library, used algorithms had some adjustable settings. We selected a range of these settings and tested a capability to recognize real defects not making false alarms. To test immunity from noises of algorithms, there was not used additional illumination, and sensitivity of camera was set to ISO800. This produces some noise in images.

Table 1. Test results

#	Defects found	Real defects	Settings	Faulty defects %
1	2	0	1|0|7|0|0|9|0.040|50 x 50|0.90|	0,19
2	1	0	1|0|7|0|0|11|0.080|50 x 50|0.90|	0,09
3	1	0	1|0|7|0|0|15|0.120|50 x 50|0.90|	0,09
4	0	0	1|0|7|0|0|9|0.040|75 x 75|0.90|	0,00
5	0	0	1|0|7|0|0|11|0.080|75 x 75|0.90|	0,00
6	0	0	1|0|7|0|0|15|0.120|75 x 75|0.90|	0,00
7	0	0	1|0|7|0|0|9|0.040|100 x 100|0.90|	0,00
8	0	0	1|0|7|0|0|11|0.080|100 x 100|0.90|	0,00
9	0	0	1|0|7|0|0|15|0.120|100 x 100|0.90|	0,00
10	1	0	1|0|7|0|0|9|0.040|50 x 50|0.80|	0,09
11	1	0	1|0|7|0|0|11|0.080|50 x 50|0.80|	0,09
12	1	0	1|0|7|0|0|15|0.120|50 x 50|0.80|	0,09
13	0	0	1|0|7|0|0|9|0.040|75 x 75|0.80|	0,00
14	0	0	1|0|7|0|0|11|0.080|75 x 75|0.80|	0,00
15	0	0	1|0|7|0|0|15|0.120|75 x 75|0.80|	0,00
16	0	0	1|0|7|0|0|9|0.040|100 x 100|0.80|	0,00
17	0	0	1|0|7|0|0|11|0.080|100 x 100|0.80|	0,00
18	0	0	1|0|7|0|0|15|0.120|100 x 100|0.80|	0,00

In Table 1 we see results when printed circuit board is without real defects. As it is seen, using some filter (Harris corner detector in this case) settings still produce errors.

Table 2 represents results that were obtained analyzing image of an object with real defects. As we see, there are some settings that recognize all real defects without making false alarms.

Table 2. Test results

#	Defects found	Real defects	Settings	Faulty defects, %
1	43	32	1|0|7|0|0|9|0.040|50 x 50|0.90|	1,02
2	37	33	1|0|7|0|0|11|0.080|50 x 50|0.90|	0,37
3	35	33	1|0|7|0|0|15|0.120|50 x 50|0.90|	0,19
4	20	20	1|0|7|0|0|9|0.040|75 x 75|0.90|	0,00
5	19	19	1|0|7|0|0|11|0.080|75 x 75|0.90|	0,00
6	19	19	1|0|7|0|0|15|0.120|75 x 75|0.90|	0,00
7	10	10	1|0|7|0|0|9|0.040|100 x 100|0.90|	0,00
8	10	10	1|0|7|0|0|11|0.080|100 x 100|0.90|	0,00
9	10	10	1|0|7|0|0|15|0.120|100 x 100|0.90|	0,00
10	32	30	1|0|7|0|0|9|0.040|50 x 50|0.80|	0,19
11	32	30	1|0|7|0|0|11|0.080|50 x 50|0.80|	0,19
12	30	30	1|0|7|0|0|15|0.120|50 x 50|0.80|	0,00
13	19	19	1|0|7|0|0|9|0.040|75 x 75|0.80|	0,00
14	19	19	1|0|7|0|0|11|0.080|75 x 75|0.80|	0,00
15	17	17	1|0|7|0|0|15|0.120|75 x 75|0.80|	0,00
16	9	9	1|0|7|0|0|9|0.040|100 x 100|0.80|	0,00
17	9	9	1|0|7|0|0|11|0.080|100 x 100|0.80|	0,00
18	9	9	1|0|7|0|0|15|0.120|100 x 100|0.80|	0,00

Settings of algorithms are represented in this way (note that not all settings are used in particular algorithm):

dx | dy | aperture size | threshold 1| threshold 2 | block size | k | segment size | correlation coefficient. Recommended settings for different algorithms are as follows:

Sobel:

dx = 1, dy = 0, aperture size = 7;

Using these settings false alarm level was 0 – 30 % depending on non-algorithm settings (segment size and correlation coefficient)

Canny:

threshold 1 = 50, threshold 2 = 300, aperture size = 7;

Canny image processing algorithm showed very high level of false alarms. Using recommended settings false alarm level was about 80%.

Laplace:

aperture size = 7;

Laplace operator showed good results when testing image of an object without real defects, but when simulating analysis of an object with defect, false alarm level was 25-30 % using recommended settings. Therefore object can't be placed in exact place when making picture, so even small differences in image where threaded as defects.

Eigenvalues:

aperture size ≥ 5, block size ≥ 9;

Depending on non-algorithm settings this method did not show any or showed very small number (~2%) of false alarms when using these settings.

Corner detector:

aperture size = 7;

Using this setting filter identified 0 – 5% segments as faulty when analyzing object without defects, and 15 - 90% when analyzing object with defects. The amount of false alarms depends on the segment size and correlation coefficient.

Harris corner detector:

aperture size = 7, block size ≥ 9, 0.08 ≥ k ≥ 0.12.

This algorithm identified up to 21% segments as faulty even there were no real defect. Though mentioned parameters, amount of false alarms was reduced to 0.

These settings will be used in further experiments.

There were built additional illumination modules, one using LED illumination, and second – simple bulb (Philips Spotone 100 W E27) illumination. We used these modules selecting optimal segment size for AOI system prototype.

There were used segment sizes from 10 px to 150 px with step 10 px. Correlation coefficient is set to 0.9. Pixels are recalculated to millimeters, because very important to select segment size in real world measurements.

As seen in Figure 6, the optimal segment size is 50x50 px. This corresponds to ~4x4 mm. Using this size of segment, false alarm level was ~1%. Most of the faulty identified segments were at the edges of image. It happens, because at the edges of image, the segment size is smaller than selected (no possibilities to divide image in equal segments when selecting fixed segment size).

Not all real defects were found when segment size was set to 130x130 px (it is represented as negative value of faulty defects). It happens, when a small part is missing in segment, but we still get a good correlation value, and segment isn't identified as faulty.

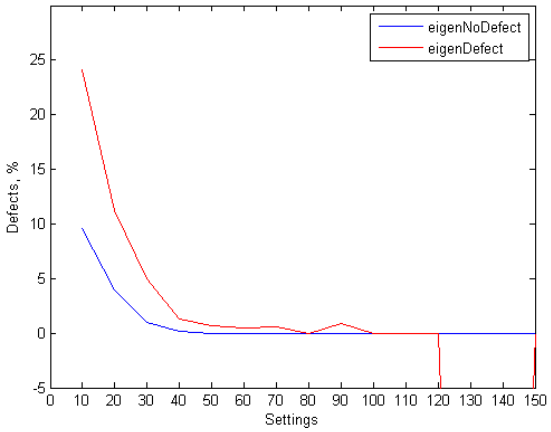

Figure 6. Selection of size of a segment using LED illumination

Label "noDefect" in the graph means that experiment is performed using picture of an object without defects, and on contrary, "*defect" means that picture of an object with real defect is analyzed.

When using a bulb illumination (see Figure 7), optimal segment size is 80x80px (corresponds to ~8x8 mm). Also false alarms were generated at the edges of image.

To get a better image resolution, the object was optically zoomed in. The optimal segment size we got was 50x50 px that corresponded to 2x2mm. (see Figure 8). False alarms (up to 2.5%) also were generated at the edges of image.

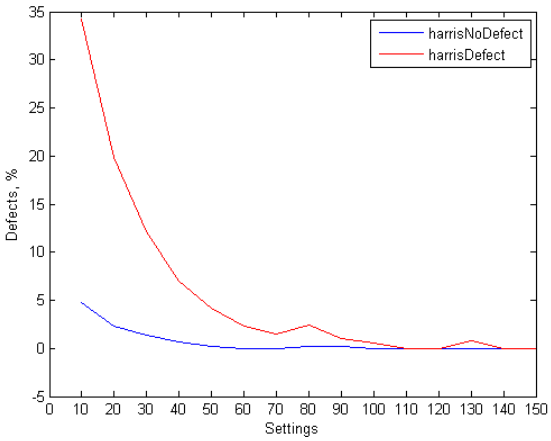

Figure 7. Selection of size of a segment using bulb illumination

There were performed some experiments with different settings of shutter speed and aperture value of camera. However there was no impact on recognition of defects.

6. Conclusions

The picture of object should be taken with all possible information; in this case, the picture should be non-gray scale and converted to gray scale. Special software was development for that purpose.

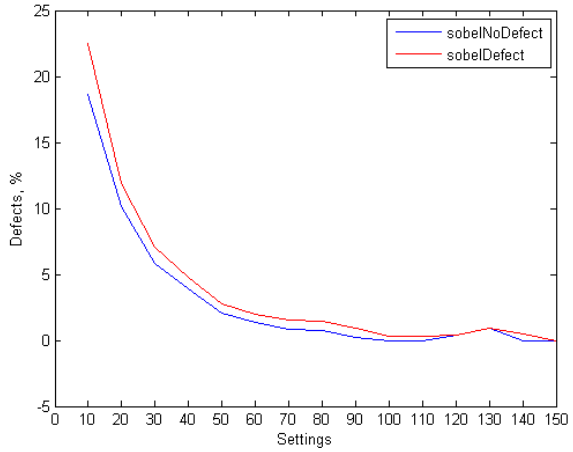

Figure 8. Selection of size of a segment using optical zoom

Most efficient algorithms for recognizing defect on PCB were Sobel, Eigen decomposition and Harris corner detector. As these algorithms have a number of adjustable parameters, it is recommended to use these parameters:

Sobel:
dx = 1, dy = 0, aperture size = 7;
Eigen values:
aperture size ≥ 5, block size ≥ 9;
Harris corner detector:
aperture size = 7, block size ≥ 9, 0.08 ≥ k ≥ 0.12;

When using these parameters, there were no, or small number of false alarms.

If there is no good additional illumination, it isn't recommended to use Sobel operator for processing image, because it is very sensitive to any changes in images.

The size of segment should be selected depending on illumination and size of object. Too small segment size generates a lot of false alarm, and if segment size is big, not all real defects can be detected. The optimal segment size is 4x4mm – 11x11mm. If object is optically zoomed to get better image resolution, the segment size can be smaller.

Segment size should be selected depending on the real size of the object. The object should be divided into equal parts to prevent smaller segments at the edges of object.

Acknowledgment

The research is partially supported by Lithuanian Agency for International Science and Technology Development Programmes "Eureka" project E!3807 – „EKO-FACTORY: Intelligent Factory Production Identification System".

References

[1] Agilent Technologies. Automated Optical Inspection for Electronics Manufacturing. *Article available: http://www.home.agilent.com/agilent/editorial.jspx?cc=EN&lc=por&ckey=205277&nid=-536900137.0.02&id=205277.*

[2] **N. Zuech.** Machine Vision Software. *Posted* 11/03/2005. *Article available: http://www.machinevisiononline.org/public/articles/articlesdetails.cfm?id=2641.*

[3] **Q. Gao, L. Zhang, D. Zhang, J. Yang.** Comments on "On Image Matrix Based Feature Extraction Algorithms". *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 37, *No*.5, 2007, 1373-1374.

[4] OpenCV library. *Started on* 14 Feb 2006. *Resource site: http://opencvlibrary.sourceforge.net/.*

[5] Bloodshed Dev-C++. *Integrated Development Environment. Resource site: http://bloodshed.net/dev/devcpp.html.*

[6] wxDev-C++. *Extension of Dev-C++. Resource site: http://wxdsgn.sourceforge.net/.*

[7] **T.A. Abbasi, M.U. Abbasi.** A Proposed FPGA Based architecture for Sobel Edge Detection Operator. *J. of Active and Passive Electronic Devices, Vol*.2, 2007, 271–277.

[8] **J. Canny.** A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol*.8, *Nov*. 1986, 639-643.

[9] **T. Lindeberg.** Edge detection and ridge detection with automatic scale selection. *International Journal of Computer Vision*, 30, 2, 1998, 117-154.

[10] **Xin Wang.** Laplacian Operator-Based Edge Detectors. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol*.29, *No*.5, 2007, 886-890.

[11] **T. Bose.** Digital Signal and Image Processing. *John Wiley & Sons, Inc*. 2004.

[12] **F. Mohanna, F. Mokhtarian.** Performance evaluation of corner detectors using consistency and accuracy measures. *Computer Vision and Image Understanding*, 102(1), 2006, 81-94.

Received March 2008.