9th INTERNATIONAL SYMPOSIUM ON MEASUREMENT AND QUALITY CONTROL (9th ISMQC)
November 21 – 24, 2007, IIT Madras

AUTOMATED VISION INSPECTION SYSTEM FOR A WATER BOTTLING INDUSTRY

V.G. Sridhar
SG Lecturer , SMBS, VIT University ,vgsridhar@vit.ac.in

Mithun George Jacob
SMBS, VIT University, mithunjacob@gmail.com

Abstract**:** *Delivering quality products has become the top priority in all industries in order to remain competitive in the market. This paper mainly deals with the design and implementation of a high-speed quality inspection system capable of detecting missing bottles caps and askew caps. Image processing technique is used to identify a defective tray in the packaging area and alert the operator*

Key words: *Vision system , quality control inspection, automation.*

1. INTRODUCTION

Vision based solutions offer several advantages over conventional quality inspection systems due to their flexibility convertibility and provides a low cost solution. Several applications of machine vision have been discovered and utilized such as quality inspection, object recognition and object tracking (Groover, 1987; Boucher, 1996; Sreenivasa et al., 1993; Beatty et al., 1993; Zhang et al., 1993). The solution discussed here concerns about the quality inspection system of a Water bottling industry to achieve total quality management (TQM).

Automated visual inspection (AVI) systems can perform inspection by simulating human vision, without the wastage of human intelligence to look at the results. Although no AVI today could ever attain the levels of versatility, flexibility and discrimination of human vision, but the proposed system compensate these demerits with their ability to work continuously, and maintain a constant level of accuracy as well as working faster than any human inspector. Therefore, AVI can be utilized as a powerful tool for monitoring the quality of operations within the industry for zero-defect manufacturing.

2. METHODOLOGY

Fig.1 depicts the step by step procedure and methodology implemented in designing of Automated Visual Inspection System.

2.1 Image Preprocessing

Image preprocessing is vital in the design of inspection system especially an industrial scenario. The disadvantages experienced due to ineffective lighting will be compensated with the preprocessing algorithms ensuring that the pre-processed image can be easily used for segmentation without any loss in data.

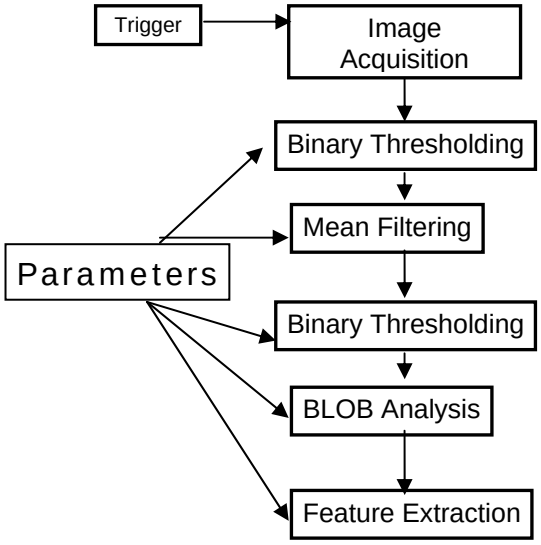


Fig. 1. Methodology

2.2 Intensity Histogram Analysis

Intensity histogram analysis has been repeatedly used to effectively threshold the acquired image. Using this analysis it is possible to identify key values used in thresholding operations.

In an image processing context, the histogram of an image normally refers to a histogram of the pixel intensity values. It is a graph showing the frequency of occurrence of pixels in an image at each different intensity value found in that image. For an 8-bit grayscale image there are 256 different possible intensities and therefore an intensity histogram will show the distribution of pixels amongst those grayscale values. Histograms can also be taken of color images; either individual histograms of red, green and blue channels can be taken (or in the case of an HSV model, the hue, saturation and value channels), or a 3-D histogram can be produced, with the three axes representing the red, blue and green channels, and brightness at each point representing the pixel count. The exact output from the operation depends upon the

implementation; it maybe a GUI interface graph displaying information about each intensity or a snapshot of the file stored as a device-independent bitmap (DIB) or even a comma-separated value (CSV) file containing the data.

The operation is very simple. The image is scanned in a single pass and a running count of the number of pixels found at each intensity value is kept. This is then used to construct a suitable histogram. If the image is suitable for thresholding then the histogram will be *bi-modal i.e.* the pixel intensities will be clustered around two well separated values. To obtain a suitable threshold value, the intensity values between the peaks of the histogram are generally subjected to analysis. If the distribution is not like this then it is unlikely that a good segmentation can be produced by thresholding.

Due to the sharp contrast between the colors of the bottle caps (blue) and the rest of the image (Fig. 2), the acquired images have proved to be highly bi-modal allowing for effective segmentation through thresholding.

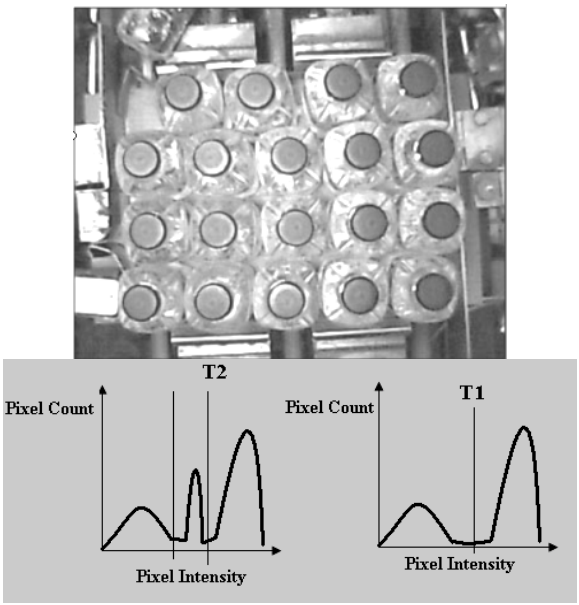


Fig. 2.Acquired Image and Histogram

Histogram analysis software was written using Microsoft Visual C++ 6.0 and was used on the acquired images to ascertain the adequate threshold ranges. Ideal histograms for bi-modal images are displayed above in Fig. 2. The first histogram provides a single threshold and the second histogram shows multiple thresholding.

As observed in Fig. 2. the peaks in the histogram indicate optimum threshold ranges. Histogram analysis of several images were conducted and using this data, the optimum ranges for thresholding was obtained.

2.2.1 Thresholding

Thresholding is achieved by obtaining the optimum threshold ranges from the histogram analysis. The histogram analysis of a sample acquired image has been displayed below in Figure 3

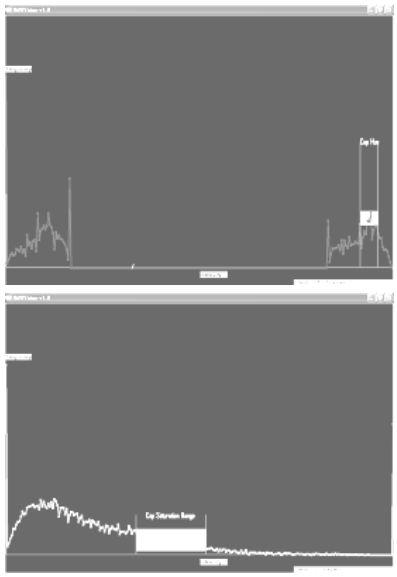


Fig. 3 Hue and Saturation Histogram Analysis

In many vision applications, it is useful to be able to separate out the regions of the image corresponding to objects in which we are interested, from the regions of the image that correspond to background. Thresholding often provides an easy and convenient way to perform this segmentation on the basis of the different intensities or colors in the foreground and background regions of an image.

In addition, thresholding allows us to highlight specific areas in the image based on specific criteria such as pixels whose values lie within a specified range, or band of intensities (or colours).

The input to a thresholding operation is typically a grayscale or color image. In the simplest implementation, the output is a binary image representing the segmentation. Black pixels correspond to background and white pixels correspond to foreground (or vice versa). In this paper, blue pixels denote the foreground before feature extraction and white pixels denote the extracted bottle cap. In simple implementations, the segmentation is determined by a single parameter known as the intensity threshold. In a single pass, each pixel in the image is compared with this threshold. If the pixel's intensity is higher than the threshold, the pixel is set to, say, white, in the output. If it is less than the threshold, it is set to black. In more sophisticated implementations, multiple thresholds can be specified, so that a band of intensity values can be set to white while everything else is set to black.

Due to non-uniform lighting, part of the image is slightly darkened resulting in different threshold values for that particular region. This hurdle has been overcome by dividing the region of interest (ROI) into two sections and applying different threshold values in both regions.

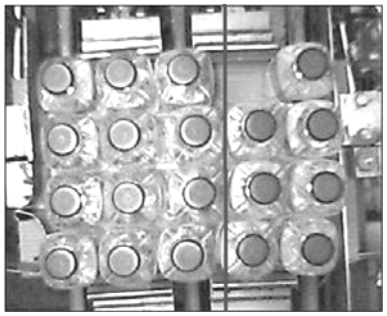


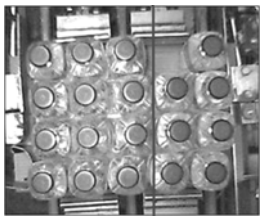
Figure 4. Split Region of Interest (ROI)

The following values have been found to be optimal for thresholding the image.

Production Line	Color Component	Normal	Dark
PL1	Hue	$239 \le H \le 251$	$55 \le H \le 105$
PL1	Saturation	$233 \le S \le 251$	$90 \le S \le 140$
PL2	Hue	$237 \le H \le 245$	$77 \le H \le 90$
PL2	Saturation	$230 \le S \le 245$	$79 \le S \le 90$

Table-1. Threshold Values for Preprocessing

Values from table 1 is used to thresholded, the acquired image to obtain the following image as shown in (Fig.6).



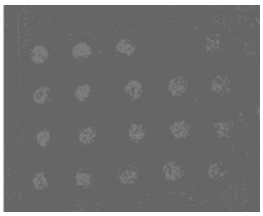


Fig. 5. Acquired Image 6.Pre-processed Image

The amount of noise within the ROI is much less than that outside and hence pre-processing the ROI alone is essential for increasing computational speed as well as reducing computational complexity.

Thresholding was implemented in the acquired image using the following criteria:

$$G_1 = \{T_1 \le f(m,n) : f(m,n) \le T_2\}$$
$$G_2 = \{ T_1 \ge f(m,n) : f(m,n) \ge T_2 \}$$

2.2.2 Convolution

Convolution is a simple mathematical operation which provides a way of `multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality. This can be used in image processing to implement operators whose output pixel values are simple linear combinations of certain input pixel values. In an image processing context, one of the input arrays is normally just a greylevel image.

The second array is usually much smaller, and is also two dimensional (although it may be just a single pixel thick), and is known as the kernel. Figure 7 shows an example image and kernel that we will use to illustrate convolution.

K_{11}	K_{12}
K_{21}	K_{22}

I_{11}	I_{12}	I_{13}	I_{14}	I_{15}
I_{21}	I_{22}	I_{23}	I_{24}	I_{25}
I_{31}	I_{32}	I_{33}	I_{34}	I_{35}
I_{41}	I_{42}	I_{43}	I_{44}	I_{45}
I_{51}	I_{52}	I_{53}	I_{54}	I_{55}

Fig. 7. Convolution Example: Image and Kernel

The convolution is performed by sliding the kernel over the image, generally starting at the top left corner, so as to move the kernel through all the positions where the kernel fits entirely within the boundaries of the image. (Note that implementations differ in what they do at the edges of images as explained below.) Each kernel position corresponds to a single output pixel, the value of which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together.

Mathematically convolution is defined as: $O(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)\ K(k,l)$

Convolution can be utilized to perform a number of image processing tasks.

2.2.3 Spatial Filtering

Spatial filtering involves the use of a certain element known as structuring elements which are used in convolution operations are generally called kernels and have a wide variety of applications.

One of the simplest smoothing filters used is the Mean filter. It is easy to implement and computationally light. The Mean filter convolves the following structuring element across the acquired image. In order to achieve optimal smoothing, this is repeated 3 times.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

The following image displays the ROI after the smoothing operation has been completed. Note the distinct lack of noise as it has faded into the background leaving only the bottle caps as blurred circular shapes.

Fig.8 Mean Filter

2.3 Segmentation

The smoothed image is subjected to binary thresholding in order to extract the individual bottle caps as Binary Large Objects (BLOBs). This final thresholding operation ensures that each bottle cap is present as a whole, cohesive binary object in the processed image with minimal noise. As observed from Fig. 9, the image has been efficiently segmented for feature extraction (BLOB analysis).

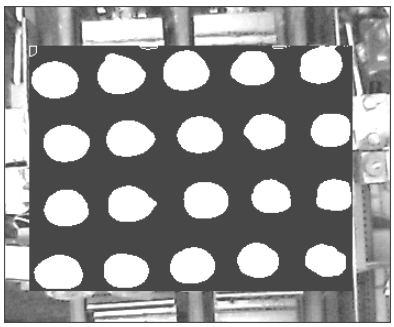


Fig. 9 Segmentation - Preparation for BLOB Analysis

2.4 Feature Detection

In computer vision and image processing the concept of feature detection refers to methods that aim at computing abstractions of image information and making local decisions at every image point whether there is an image feature of a given type at that point or not. The resulting features will be subsets of the image domain, often in the form of isolated points, continuous curves or connected regions.

2.4.1 Edge Detection

Edges are points where there is a boundary (or an edge) between two image regions. In general, an edge can be of almost arbitrary shape, and may include junctions. In practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude. Further, some common algorithms will then chain high gradient points together to form a more complete description of an edge.

In this AVI, edge detection is performed on a binary image using the following algorithm:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} IF\ I(i,j) \neq I(i+1,j)\ THEN\ I(i,j) = w_c$$

$$\sum_{i=1}^{m}\sum_{j=1}^{n} IF\ I(i,j) \neq I(i,j+1)\ THEN\ I(i,j) = w_c$$

Using this algorithm, the edge points have been detected and stored, but to ascertain the edge about individual blobs and define a minimum enclosing rectangle (MER) for each blob, the following approach is utilized.

2.4.2 BLOB Detection

BLOBs provide a complementary description of image structures in terms of regions, as opposed to corners that are more point-like. Nevertheless, BLOB descriptors often contain a preferred point (a local maximum of an operator response or a center of gravity) which means that many BLOB detectors may also be regarded as interest point operators. BLOB detectors can detect areas in an image which are too smooth to be detected by a corner detector.

Once the edges have been identified, the MER for each BLOB must be defined. This will aid in the feature extraction process, allowing us to count the number of individual bottle caps present in the acquired image and differentiating it from noise and other phenomena.

Firstly, an 8-neighbourhood function is defined to identify the next contour point in the neighborhood:

$$N(o_{x,y}) = \begin{cases} o_{x-i,y-j} & if\ o_{x-i,y-j} = w_c \\ 0\ if\ o_{x,y} = o_{a,b} & \\ 0\ if\ D\left(o_{x,y}, N(o_{x,y})\right) > D\min & \end{cases}$$

Where (i, j) are integers satisfying the condition specified above and Oa,b is a random contour point selected initially. The function D calculates the Euclidean distance between two points, and Dmin ensures that the contour point under analysis belongs to the contour of the same BLOB.

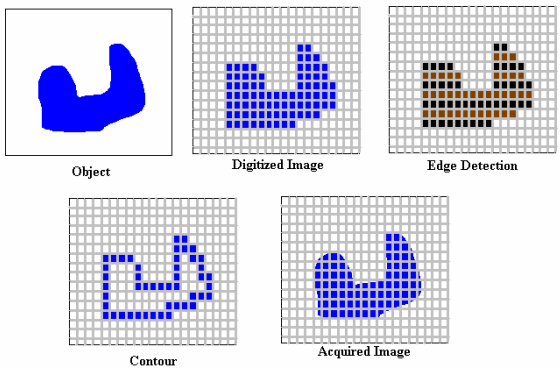


Fig. 10 Contour Extraction

Once the edge points for an individual BLOB has been established, its MER is easily defined. Using the MER, feature extraction becomes extremely simple. Fig. 11 displays the magnified view of two BLOBs (noise and bottle cap).

Fig. 11 Feature Extraction

Since the MERs for both BLOBs are known, the number of foreground pixels (blue or white) is obtained by checking all the pixels in the MER. If the number of foreground pixels, i.e. area of the BLOB crosses a particular threshold, then the BLOB is recognized as a bottle cap (white BLOB). Else, it will be treated as noise (blue BLOB). Using this concise and mathematically precise operational definition of the notion of a BLOB makes this feature detection algorithm efficient and robust for BLOB detection.

Once the BLOBs have been identified as a cap or noise, the total number of caps is counted and if it is under a certain threshold (20 bottles in the cases of PL1 and PL2), an alarm goes off alerting the operator to take corrective action.

3. CONCLUSION:

This paper presents a:

- A robust, high-speed quality inspection system capable of detecting defects in time to implement corrective measures.
- A significant decrease in the number of defective trays reaching the market.

The system has been in operation for over 6 months now and from customer feedback and reports from the Quality Control team, it is apparent that the inspection system is working perfectly as not a single complaint has been registered after the installation of this system. However, within its limitations, this method can easily perform accurately unlike other inspection systems.

4. CORRESPONDING AUTHOURS

Please contact us at the following address at
V.G.SRIDHAR
Selection Grade Lecturer,
School of Mechanical and Building sciences
VIT University, Vellore - 632014 TN. India .
Email: vgsridhar@vit.ac.in / sridhar_vellore@yahoo.com

5. REFERENCES

Shingo, S. (1986). *Zero quality control source inspection and the poke-yoke system*, Productivity Press.

Groover, M. P. (1987). *Automation, production systems and computer-integrated manufacturing*, Prentice-Hall.

Juran, J. M. (1979). *Quality Control Handbook 4th edition*, McGraw-Hill.

Kennedy, C. W., Hoffman, E. G., and Bond, S. D (1987). *Inspection and gauging 6th edition,* Industrial Press

Doif, R. C. (1988). *International Encyclopedia Of Robotics Applications And Automation*, Wiley.

Boucher, T. O. (1996). *Computer Automation In Manufacturing An Introduction*, Chapman & Hall.

Panayiotis Panayiotou and Keith Ridgeway (2000). *Vision and Quality Improvement*, Manufacturing Engineer.

K. K. Sreenivasan, M. Srinath, A. Khotanzad (1993). *Automated Vision System for Inspection of IC Pads and Bonds*, IEEE Transactions On Components, Hybrids, And Manufacturing Technology, Vol. 16, No. 3A. Beatty,

R.G. Gosine, and C.W. de Silva (1993). *Recent Developments In The Application Of Computer Vision For Automated Herring Roe Assessment*, IEEE Pix Rim 93 Pg. 698 – 701.

Francisco J. Barrientos García, Isaac García Incertis, Félix Miguel Trespaderne,Eusebio de la Fuente López, José R. Perán González System (2004). *Production Control and Automatic Packaging of Plastic Air Sleeve Guides*, Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age,1050-4729/04.

A. Khotanzad, H. Bannerjee, M. Srinath (1992). *A Vision System for Inspection of Ball Bonds in Integrated Circuits*, 0-8186-2840-5/92, 1992, IEEE

Lu Xiangju, Fan Guoliang, Wang Yunkuan (2006). *A Robust Barcode Reading Method Based on Image Analysis of a Hierarchical Feature Classification*, Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems October 9-15, 2006, Beijing, China

Omron Electronics LLC (2004). *Inspection Solutions for the Food & Beverage Industry*, Y021-E-02, 10/04/7.5M, Omron Electronics LLC, 2004.

Melles Griot (2003). *Lighting Fundamentals*, Machine Vision Guide, Melles Griot, 2003.

E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit, Jean-Didier Legat (2003). *A Survey On Industrial Vision Systems, Applications and Tools*, Image and Vision Computing 21 2003 171-1