

FAST MULTI-RESOLUTION IMAGE PROCESSING FOR PCB MANUFACTURE

S.E. Hodges and R.J. Richards¹

Abstract

This paper addresses the problem of locating visual features on a printed circuit board (PCB) to facilitate automated manufacture and assembly of PCBs. A brief introduction to PCB manufacture is given to provide background information and demonstrate the large variation in types of PCB. An overview of visual servoing explains why the features must be located quickly and robustly. By presenting an algorithm which is able to locate features in the image of a PCB at any given scale quickly and then determine their position accurately, this paper aims to satisfy the criteria outlined.

Following a discussion of the approach taken to detect features in the image, the theory behind a particularly efficient implementation is presented. Based on an approach to range searching in computational geometry, partial summation eliminates much of the processing which would otherwise be involved. It is particularly suited to multi-resolution image processing, and can be used both to find candidate feature locations and to confirm their precise position. Results which demonstrate the power of the technique for processing PCB images are given, and extensions to other applications are considered.

1 Background

1.1 PCB manufacture

Electronic products have relied on *printed circuit boards* (PCBs) for their construction for several decades and there is no reason to believe that this is likely to change in the foreseeable future. In the early 1980s, a radical new approach to electronics assembly – *surface mount technology* (SMT) – was developed. Although a PCB is still at the centre of the process, with SMT much higher component densities are achieved using especially designed components which are bonded to the surface of the PCB. This differs from the traditional, *through hole* (TH) method, where the leads of the component are passed through holes drilled in the board. In both cases, the components are held in place mechanically by soldered joints to metal *pads* on the surface of the PCB. These joints also make the required electrical connections, through a series of interconnecting *tracks*. *Multi-layer* PCBs have several layers of tracks, which are interconnected with conducting *via holes*.

PCBs differ in a number of aspects, depending on the details of the manufacturing process which is to be used. TH boards have round or ring-shaped pads whereas SMT requires rectangular pads which are generally much smaller and closer together. In either case the board may be *plated*, usually with a tin-lead alloy or gold, before use in order to make the pads more receptive to soldering. However, in order to prevent the problems which result when the tracks come into contact with the solder, a *solder resist mask* is often applied to the surface of the board. This is a thin, usually transparent, non-conducting layer which covers the entire board except for the pads. The PCB itself may be made from one of a number of different materials, each with differing characteristics [12], and identification marks are often screen printed onto the board for reference at later stages.

1.2 Visual servoing

The use of visual feedback has been studied in robotic applications for several decades. However, recent advances in VLSI technology have dramatically reduced the cost of the electronic hardware required to

¹S.E. Hodges and R.J. Richards are with the Department of Engineering, University of Cambridge.

successfully implement a visual feedback robot controller. It is perhaps for this reason that there has been a revival in the study and application of *visual servoing* techniques in the last few years [8, 15, 16]. In this method, a camera is used to provide information about the position of the robot's end effector with respect to a visual target, which allows a simple position controller to be used to reduce this error to zero. Implementing such a scheme effectively requires that two problems are overcome. First, the features in the image which are used to generate the feedback information (the *visual cues*) must be recognised reliably and their changing position determined accurately [10]. Secondly, this must be done quickly to produce useful feedback data [3]. These problems can be viewed as conflicting goals; with most algorithms there is a compromise between robustness and speed. Therefore the performance of the visual servoing system is highly dependent on the feature detection algorithm.

1.3 Contribution of this work

This paper outlines a new, fast image processing algorithm for feature detection and localisation which can be applied at any resolution without a penalty in processing speed. Its development was motivated from research into the use of visual servoing for automated PCB manufacturing processes, namely drilling, TH component insertion and surface mount (SM) component placement.

2 Multi-resolution image processing

2.1 Normalising the image

In an industrial environment it is very difficult to achieve uniform lighting conditions. Therefore, it is important that image processing algorithms be robust to variations in lighting, both over time and over the area of the image. One approach to the problem of non-uniform lighting is to determine average grey levels in different parts of the image, fit a mathematical surface to these and then normalise this [6]. This process is fundamental to the successful application of many image processing algorithms. All the images used in this paper have been normalised with respect to lighting.

2.2 Locating features

As outlined in Section 1.1, different features must be detected under varying conditions for PCB manufacture. A feature is defined as a group of connected pixels with similar grey levels. Thresholding the image to group pixels with similar grey levels is a standard technique [1]. The problem with this approach is the noise which will inevitably be present; removing noisy pixels reliably is difficult and computationally demanding [4]. One solution proposed is to consider each pixel in conjunction with its neighbours so that noisy pixels can be detected and rejected. Traditionally a neighbourhood of 3×3 or 5×5 is used; if larger, the processing required (which grows exponentially with the size of the neighbourhood) becomes prohibitive.

If the neighbourhood is extended to an arbitrary shape which exactly covers the feature in question, a potentially very robust detector can be constructed. By summing the grey levels of the pixels in this new neighbourhood, an average grey level for the feature can be calculated. If the neighbourhood is aligned with the feature in question, the average grey level will be exactly that of the feature. This approach is tolerant to noise, which is more likely to average out over the larger neighbourhood. In effect, the neighbourhood becomes a simple template,² which must be passed over the image to search for matches.

For visual servoing, it is crucial that the features on the PCB are detected quickly. For the algorithm outlined above to be useful, the grey levels of the pixels in the neighbourhood defined by the template must be summed quickly. If the template is of an arbitrary shape, this is difficult to do. However, Section 3 presents a very efficient summation technique which can be used if the template is a continuous

²Template matching usually involves correlating a grey level template with the image [6]. Here, a binary template is effectively applied.

rectangular block of pixels. Using this, rectangular features of different dimensions may be detected by applying an appropriate template.

The use of a rectangular template may seem limiting, but pads for SM PCBs are almost exclusively rectangular and oriented horizontally or vertically; this approach is therefore ideally suited to detecting SMT pads. The pads used with TH PCBs are usually round or ring-shaped. However, by relaxing the constraints of the matching, a square template can be successfully used as an approximation. Indeed, most shapes can be simply represented by a coarse template consisting of a small number of rectangular blocks [14] which would allow this technique to be extended to a number of other applications.

2.3 Determining precise positions

Matching features by using coarse templates as outlined above can lead to errors. For this reason, matches are considered as *candidate* feature locations. Occasionally a candidate may not indicate the location of a real feature, or possibly more than one feature will be located at the candidate site. In any case, the candidate does not indicate the precise position of the feature.

A useful approach for overcoming these problems is calculation and analysis of the horizontal and vertical *lateral histograms*³ of the image [4]. These are calculated by summing the grey levels of the pixels in each of the columns and rows of the image respectively (see Figure 1)

$$H(x) = \sum_{y=1}^Y I(x, y) \quad V(y) = \sum_{x=1}^X I(x, y). \quad (1)$$

Lateral histograms can be very useful for highlighting the positions of features in an image [2, 5]. An image usually contains many features, so it is often appropriate to consider parts of the image in isolation. Suitable sub-images are the areas around the candidate feature locations. Simple analysis will confirm or deny the presence of the feature in question, and will also give a more accurate estimate of its location.

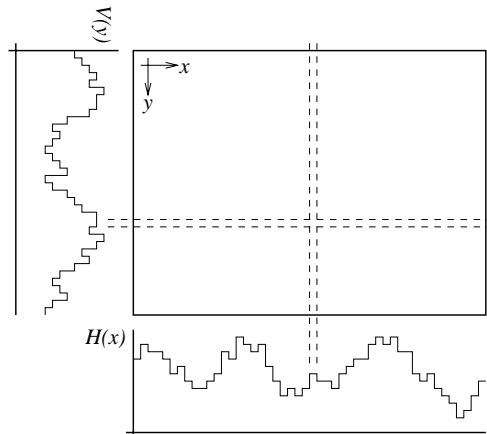


Figure 1: Calculating the lateral histograms

3 Partial summation

The basis of the approach to image analysis proposed in this paper is a technique known as *partial summation*, after Omohundro [11].⁴ Access to the image data is speeded up at the expense of a pre-processing stage, which generates a table of *partial sums*. This table may then be used to calculate the sum of grey levels in any arbitrary rectangular region of the image, as discussed in Section 2.2. The same table can also be used to normalise the image (Section 2.1) and determine precise feature positions (Section 2.3).

Consider a single row of X pixels of image data $I(x)$ where $1 \leq x \leq X$. A second data structure J can be generated, in which each entry is set to the cumulative sum of the pixel values in I

$$J(x) = \begin{cases} 0, & x = 0 \\ \sum_{i=1}^x I(i), & 1 \leq x \leq X. \end{cases} \quad (2)$$

⁴Partial summation is similar to the use of vector dominance to perform range searching – a technique used in computational geometry. A clear introduction to this topic is given by Preparata and Shamos [13].

J contains the partial sums of I , and can be used to calculate the sum of grey levels for any arbitrary run of consecutive pixels in I [7] (from $I(r)$ to $I(s)$ inclusively)

$$\sum_{i=r}^s I(i) = J(s) - J(r - 1). \quad (3)$$

For example, the sum of the grey levels of pixels 3 through 6 inclusive is given by $J(6) - J(2)$. In the example of Figure 2, this gives a value of 11, which can be verified readily from I .

$I =$	2	4	1	3	6	1	1	2	4	3	
	$I(1)$	$I(2)$	$I(3)$	$I(6)$	$I(10)$	
$J =$	0	2	6	7	10	16	17	18	20	24	27
	$J(0)$	$J(1)$	$J(2)$	$J(6)$	$J(10)$	

Figure 2: Example row of image data (I) and corresponding 1D partial sums (J).

This method can be extended to any number of dimensions, but the two dimensional case is of greatest interest due to its application in image processing [9, 11]. In this case, two preprocessing steps are necessary. The first computes partial sums for each row of the image, as outlined above

$$J(x, y) = \begin{cases} 0, & x = 0 \\ \sum_{i=1}^x I(i, y), & x = 1, 2, \dots, X. \end{cases} \quad (4)$$

These are then used in a second, vertical pass which effectively computes partial sums of partial sums

$$K(x, y) = \begin{cases} 0, & x = 0 \text{ or } y = 0 \\ \sum_{j=1}^y J(x, j), & x = 1, 2, \dots, X; y = 1, 2, \dots, Y. \end{cases} \quad (5)$$

Each entry $K(x, y)$ contains the sum of the grey levels of all pixels to the left of, above and including the pixel at position (x, y) in I . It is now possible to sum the grey levels within any rectangular area of the image I enclosed by $I(a, b)$ and $I(c, d)$ (see Figure 3(a)). $K(c, d)$ is the sum of grey levels of all pixels to the left of and above $I(c, d)$. Subtracting $K(a - 1, d)$ from this removes the unwanted pixels to the left of the specified rectangle, and similarly those above it can be subtracted using $K(c, b - 1)$. The pixels which are both to the left of and above $I(a, b)$ have now been discounted twice, so $K(a - 1, b - 1)$ must be added back in (Figure 3(b)). Thus:

$$\sum_{x=a}^c \sum_{y=b}^d I(x, y) = K(c, d) - K(a - 1, d) - K(c, b - 1) + K(a - 1, b - 1). \quad (6)$$

Partial summation therefore enables the sum of grey levels within any arbitrary rectangular region of an image to be calculated from just four values. This dramatically speeds up the implementation of the feature location algorithm of Section 2.2. The same partial sums can also be used to generate lateral histograms, which can be used to normalise the image (Section 2.1) and determine precise feature positions (Section 2.3).

4 Results

Two different PCBs were used for initial testing of the algorithm. Figure 4 shows the original image of an SMT PCB, along with its lateral histograms. Note in particular that the image is brighter towards

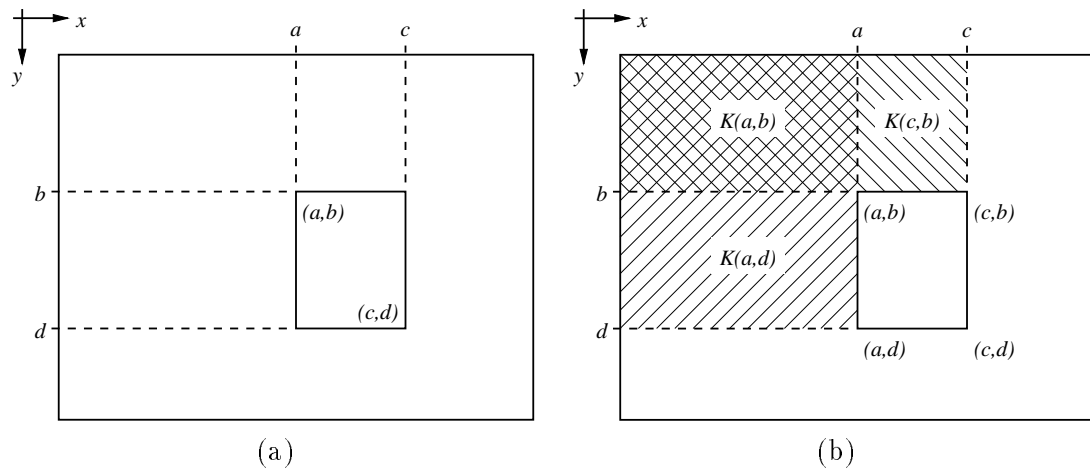


Figure 3: (a) Rectangular region in I defined by two vertices;
 (b) Using K to sum the grey levels in the region.

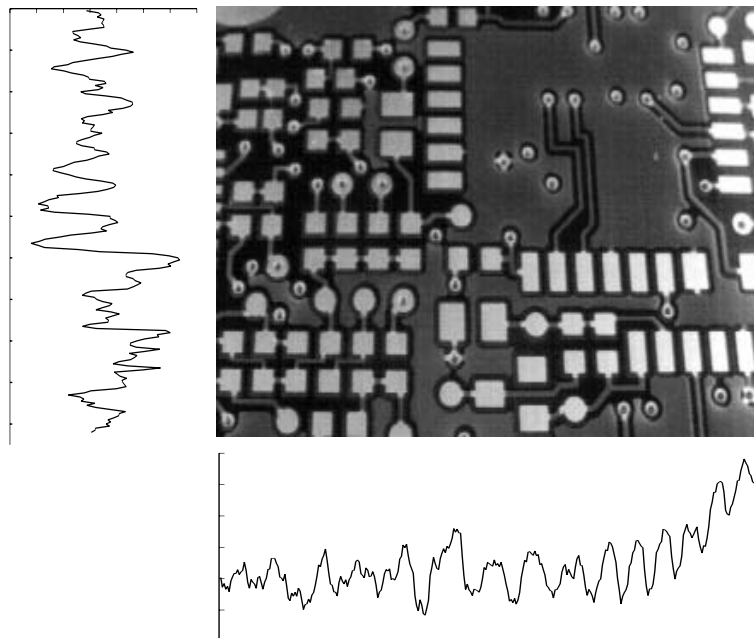
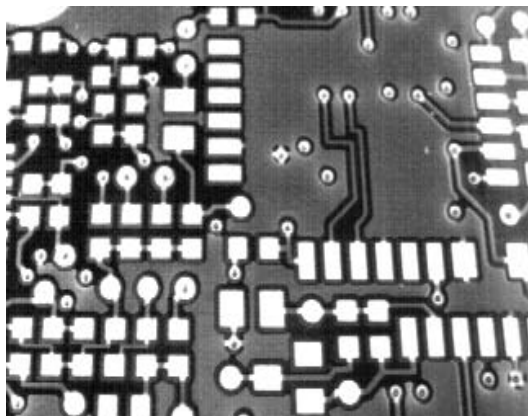
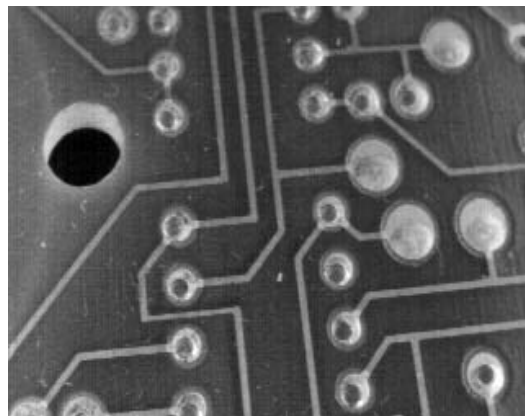


Figure 4: Original image of SMT PCB with lateral histograms.



(a)



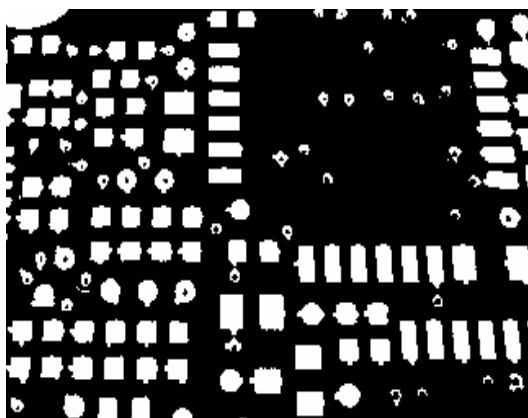
(b)



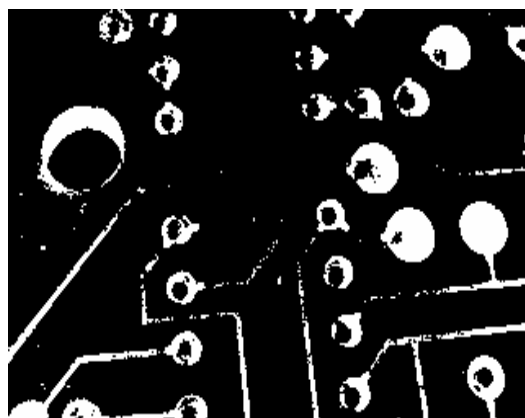
(c)



(d)



(e)



(f)

Figure 5: (a) & (b) Test images; (c) & (d) Results of feature detection; (e) & (f) Results using simple thresholding.

the right, which can be seen in the horizontal histogram. The histograms were used to normalise the image and hence compensate for the non-uniform lighting, see Figure 5(a). Figure 5(b) shows the light-normalised image of a TH PCB.

Preliminary results of applying the feature matching algorithm, where the feature is simply a square region of bright pixels, are given in Figures 5(c) and 5(d). The processed image is smaller than the original because a band of pixels around the edge are lost when the template overhangs the border of the image. In both examples the pads were detected robustly and quickly. Figure 5(c) shows two instances of adjacent pads being joined together and some false candidates corresponding to test points on the PCB, which are very similar to the pads. Figure 5(d) shows just one false candidate (the large ‘horseshoe’ on the left). The algorithm requires just 6 accesses to each pixel in the image (2 to calculate the partial sums and 4 to perform the simple template matching), and therefore executes quickly.

To provide a comparison between the feature matching algorithm and a conventional technique, Figures 5(e) and 5(f) show the normalised images after a thresholding operation. To eliminate unwanted features such as via holes, tracks and noise, a spatial filter must be applied. This will take $O(n^2)$ time; even for a small filter (such as $n = 5$), this will be slower than the new method presented and will still generate false candidates.

As mentioned in Section 2.3, lateral histograms may be used to examine the candidate features in more detail; they can be calculated very quickly from the partial sums already computed. One of the matches from Figure 5(c) (the horizontal rectangular blob towards the top left of the image) is analysed in this way in Figure 6. This shows that the candidate match actually contains two pads; their positions and sizes can be readily determined from the histograms. Similarly, false candidates can be rejected following analysis of their lateral histograms.

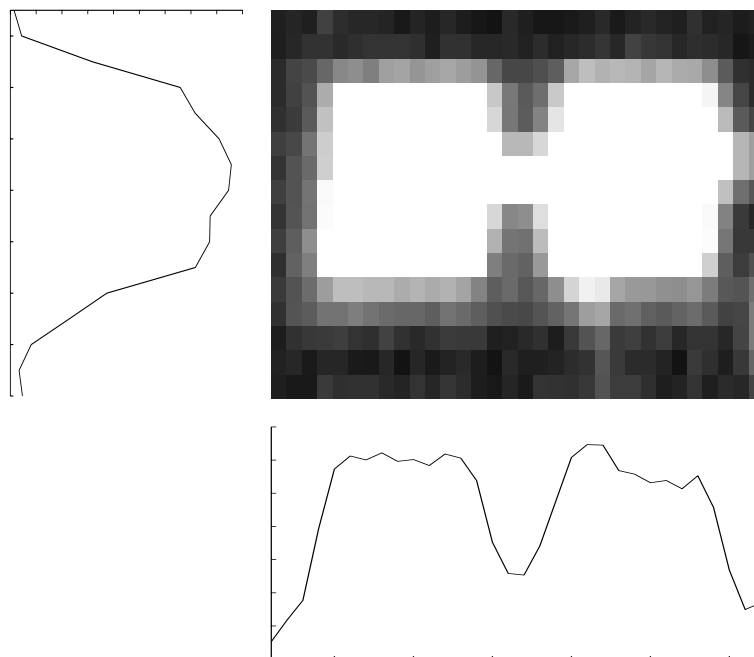


Figure 6: Determining precise details from a candidate area.

5 Conclusions

This paper has presented a new approach to feature detection, based on a form of template matching. By using rectangular templates, it is possible to implement the detector very efficiently with a technique

known as partial summation. Initial experiments have successfully detected pads in images of PCBs; results which demonstrate the effectiveness of the new approach have been presented.

By making the templates more complex, it should be possible to detect a large variety of features robustly and thereby apply this technique in a wide range of applications.

References

- [1] H. Bässerman and Ph.W. Besslich. *Image Processing Ad Oculos*. Springer-Verlag, 1991.
- [2] J. Burke. Assembly with advanced components. *Electronic Production*, 22(7):17–19, September 1993.
- [3] P.I. Corke. Visual control of robot manipulators – a review. In K. Hashimoto, editor, *Visual Servoing: Real-Time Control of Robot Manipulation Based of Visual Sensory Feedback*, Robotics and Automated Systems, pages 1–31. World Scientific, 1993.
- [4] E.R. Davies. *Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, 1990.
- [5] I. Delchamber. Vision in printers. *Electronic Production*, 23(7):17–19, July/August 1994.
- [6] R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison Wesley, second edition, 1992.
- [7] G.D. Hager. Real-time feature tracking and projective invariance as a basis for hand-eye coordination. Technical Report YALEU/DCS/RR-998, Yale University Department of Computer Science, New Haven, CT., December 1993. Also appeared in CVPR 1994.
- [8] K. Hashimoto, editor. *Visual Servoing: Real-Time Control of Robot Manipulation Based on Visual Sensory Feedback*. Robotics and Automated Systems. World Scientific, 1993.
- [9] S.E. Hodges. A new approach to spatial filtering for faster image processing. Technical report, Department of Engineering, University of Cambridge, Trumpington Street, Cambridge, In preparation.
- [10] J. Huang and G.D. Hager. Tracking tools for vision-based navigation. Technical Report YALEU/DCS/RR-1059, Yale University Department of Computer Science, New Haven, CT., January 1995.
- [11] S.M. Omohundro. Efficient algorithms with neural network behavior. Technical Report UIUCDCS-R-87-1331, Department of Computer Science, University of Illinois at Urbana Champaign, Urbana, Illinois, April 1987.
- [12] N. Pearne. Choosing a PCB laminate. *Electronic Production*, 22(3):15–19, April 1993.
- [13] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction*. Springer-Verlag, 1988.
- [14] S. Ranade, A. Rosenfeld, and H. Samet. Shape approximation using quadtrees. *Pattern Recognition*, 15(1):31–40, 1982.
- [15] S.W. Wijesoma, D.F.H. Wolfe, and R.J. Richards. Eye-to-hand coordination for vision-guided robot control applications. *International Journal of Robotics Research*, 12(1):65–78, February 1993.
- [16] B.H. Yoshimi and P.K. Allen. Active, uncalibrated visual servoing. In *Proceedings of the IEEE International Conference on Robotics and Automation*. IEEE, 1994.