Robot Identification using Shape Features on an FPGA-Based Smart Camera

Miguel Contreras, Donald Bailey, Gourab Sen Gupta
School of Engineering and Advanced Technology,
Massey University
Palmerston North, New Zealand
M.Contreras@massey.ac.nz, D.G.Bailey@massey.ac.nz, G.SenGupta@massey.ac.nz

ABSTRACT
An FPGA based smart camera captures images of soccer robots, and processes them using stream processing to identify the individual robots. Three standard methods of distinguishing the robots based on shape features are compared, and the advantages and disadvantages of the different methods discussed. Compactness and moment based measures give poor performance because of the sensitivity of the low resolution shapes to relatively small distortions. Normalised area gave the best results in this application.

Categories and Subject Descriptors
I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement – *Size and shape*; I.5.4 [Pattern Recognition]: Applications – *Computer vision.*

General Terms
Algorithms, Measurement, Experimentation.

Keywords
Robot soccer, compactness, moments, area.

1. INTRODUCTION
Robot soccer has often been used as a platform for robotic and robot vision research. Recently, we have been exploring [1, 4] an FPGA-based smart camera for robot identification.

With small robots, a global vision system is usually used to track the robots and the ball. In a global vision system, the camera is mounted over the playing area, and has the complete playing field within the field of view. Software based image processing is usually used to locate the robots and ball, and pass this data on to the strategy processor which determines the desired behaviour, and sends commands to the individual robots.

Moving to an FPGA based smart camera can significantly increase both the spatial and temporal resolution, and reduce the latency [4], all of which lead to improved control of the robots. The smart camera replaces the image processing performed on a conventional computer, and performs all of the processing within the camera itself [3]. Therefore, only the data extracted from the images, rather than the images themselves, need to be sent from the camera, reducing communication overhead. Most smart cameras store the captured image within a frame buffer before beginning processing. Using an FPGA, however, enables the images to be processed directly as they are streamed from the camera, significantly reducing the latency [1]. However, stream processing imposes strict timing constraints (one pixel per clock cycle) and requires all processing to be performed on the pixels in the raster order. These place significant constraints on the algorithms used within the processing.

Individual robots are identified by coloured patches on the top surface of the robots (visible to the global camera, see Figure 1). The orientation of the robot is determined from green and pink triangles positioned on opposite corners of the robots. The two teams are distinguished from the colour of the team patch in the centre of the robot; one team is blue, and the other yellow. The individual team members are distinguished by the shape of the team patch, whether circular, square, or rectangular.

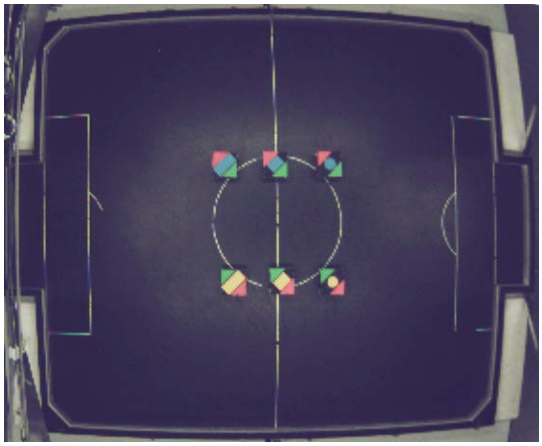

Figure 1. View of robots as seen from the global camera.

This paper explores different approaches to identify the individual robots based on shape features extracted from the team patch, subject to the processing limitations associated with processing streamed data (using only local information). Specifically, three approaches are compared: using compactness or complexity, second moments, and areas.

Section 2 outlines constraints and other complicating factors associated with the images, and describes the pre-processing performed on the images before shape recognition. Section 3 describes the three approaches to identifying the robots, while section 4 discusses the advantages and limitations of each.

2. CONSTRAINTS
When processing the images using an FPGA, stream processing is the most appropriate processing mode [2]. This requires all of the processing to be performed for a pixel as it is streamed from the

IVCNZ '14, November 19–21 2014, Hamilton, New Zealand

ACM 978-1-4503-3184-5/14/11…$15.00
http://dx.doi.org/10.1145/2683405.2683437

camera. Point operations can easily be performed, as can filters with appropriate row buffering [2]. However, region processing and feature extraction are a little more complex. Single pass connected components analysis is able to measure any features which can be accumulated incrementally [8], such as area, moments, perimeter, etc. However more complex region processing, requiring multiple neighbouring pixels, cannot be practically implemented due to the timing and processing constraints. For efficient resource utilisation of the FPGA, limiting calculations to integer or fixed-point arithmetic is a necessity.

The position of the camera over the playing area means that a wide angle lens must be used to fit the complete playing area within the field of view. This results in significant lens distortion. When mounting the camera, it is virtually impossible to have the camera precisely perpendicular to the planar playing surface. Therefore, mild perspective distortion is also inevitable. The effects of these mean that the size of a pixel is not uniform across the image (there is a variation of scale), and there are distortions to the shapes of objects although this is minor when considering shapes locally. Normally such lens distortions would be removed through camera calibration and image rectification. However, this does not fit easily within a stream processing framework, and would significantly impact the latency. Instead, the distorted images are processed, with the camera calibration applied to the extracted features.

More significant is the variation in illumination across the image. This, combined with global colour thresholding for object detection, affects what is detected. The pixels around the edge of each coloured patch, which consist of a mixture of colour and background, are most sensitive to this, resulting in changes to the area detected. When coupled with the relatively small size of each coloured patch, this effect can be significant. Small flaws in the detection of the shape become magnified the further they are from the centre, caused by differences in pixel scale and luminance.

The colour image sensor uses a Bayer pattern to give the coloured pixels. This lowers the effective colour resolution, and the interpolation of the colour channels as part of desmoaicing can introduce artefacts at coloured boundaries.

2.1 Processing

The image processing algorithm [4] starts by capturing raw pixels from the image sensor and converting them to full colour using a bilinear Bayer interpolation filter. With the camera and lens combination used, a pixel skipping mode is required to achieve the desired resolution with the field of view covering the complete playing area. A non-linear colour edge enhancement filter is then used to reduce the blur introduced by the interpolation filter. The colour pixels are then converted to luminance and chrominance components (using a power of 2 conversion matrix [10]) to separate much of the luminance effects of the changing light levels. (Note that with the linear transformation, the chrominance components will still scale with changes in lighting.) The image is then thresholded to detect the colours associated with the patches. The output from this stage is a label image, with each pixel represented by a label corresponding to the detected colour. A connected components analysis algorithm determines groups of adjacent pixels with the same label. A simplified single pass algorithm is able to be used because the coloured patches are known to be convex.

During connected components analysis, the shape features required for shape location and identification are accumulated. These include:

- area;
- perimeter (determined by detecting the region edges);
- sum of x, and sum of y (used for calculating the centre of gravity); and
- sum of x^2 and sum of y^2 (used for calculating second moments).

Each detected region is associated with a robot (or ball), and from the features the robot location, orientation, and identity are determined.

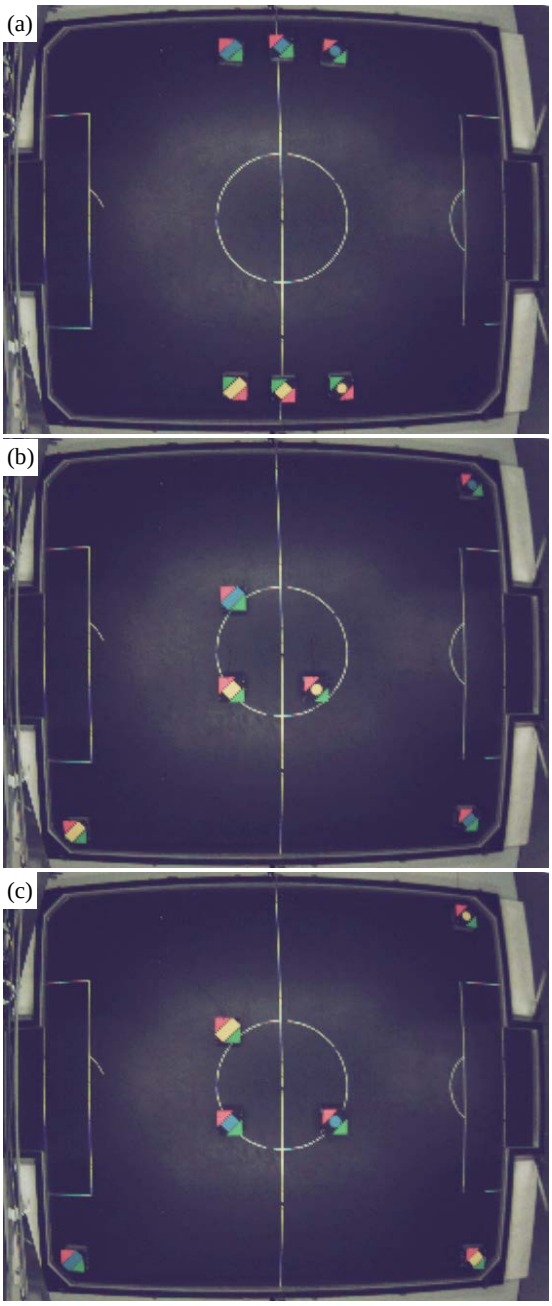

Figure 2. Other test images used.

2.2 Test Images

In addition to the image shown in Figure 1, the images in Figure 2 are also used for testing and analysis of the various methods. Figure 1 shows the robots in the centre, where lighting is the best, and distortion is minimal. This can be considered as the best case conditions. Figure 2(a) shows the robots further from the centre, introducing both equal amounts of distortion errors and variations in lighting to each team. Both the distortion and lighting are worst in the image corners, so are representative of the worst case conditions. Figures 2(b) and 2(c) demonstrate a mixture of both best and worst case conditions, with alternating team positions.

3. METHODS CONSIDERED

The three robots on each team are distinguished by different shaped team patches. The different shapes are:

- a circle
- a square (with aspect ratio of approximately 1:1); and
- a rectangle (with aspect ratio of approximately 2.4:1).

The method used to distinguish these shapes must fit with the stream processing model. This limits the range of features that can be considered, and rules out features that cannot be calculated incrementally (for example those based on the convex hull, minimum area enclosing rectangles, Fourier descriptors, etc.), as these would require multiple passes. The extraction of the region itself would require a two pass component labelling algorithm, requiring external frame buffer memory, and severely impact on the algorithm latency.

Due to the highly dynamic nature of robot soccer, the robot identification method must be robust to every possible situation. It must be rotation invariant, as the robot must be correctly identified in any orientation. It must also be invariant to the distortions present within the image. The primary distortion is lens distortion, which predominantly affects the scale, as a result of the reduction in magnification with radius from the centre of the image. Therefore, without image rectification, scale invariance is also essential.

Three methods were chosen and compared using Matlab, before implementation on the FPGA.

- One of the most commonly used rotation and scale invariant shape descriptors is compactness, or its inverse, complexity [5]. It is dimensionless, making it scale invariant, and has an advantage of being simple to calculate, and its constituent features, area and perimeter, can be calculated incrementally.
- A related shape descriptor is the spread of an object. This is based on a moment invariant using second moments. Moment invariance makes it independent of orientation, and after normalising by the area gives a scale invariant dimensionless descriptor. The moment features are also able to be calculated incrementally.
- The observation is made that the different shaped identification patches have different areas. The patch area on its own cannot be used directly, because of scaling issues resulting from distortion, but the area can be normalised by the area of the orientation patches.

Each of the methods will be described in more detail, and results compared in the following sections.

3.1 Compactness or Complexity

Compactness is a relatively simple and widely used shape descriptor [5, 9]. It uses the perimeter, *P*, and area, *A*, of a binary shape and produces a dimensionless value, irrespective of size or orientation:

$$Compactness = \frac{4\pi A}{P^2} \tag{1}$$

Compactness has a maximum of 1.0 for circles, and decreases as the shape becomes more elongated or has more protrusions (for a given area, the perimeter increases).

From a computation point of view, its inverse, complexity, is more convenient to calculate on an FPGA as a simple fixed point number can be used:

$$Complexity = \frac{P^2}{A} \tag{2}$$

Ideally, the different shapes can be distinguished from their different descriptor values:

$$Complexity_{circle} = \frac{(2\pi r)^2}{\pi r^2} = 4\pi \approx 12.6 \tag{3}$$

$$Complexity_{square} = \frac{(4h)^2}{h^2} = 16 \tag{4}$$

$$Complexity_{rectangle} = \frac{(2w+2h)^2}{wh} \approx 19.3 \tag{5}$$

where *r* is the radius of the circle, and *w* and *h* are the width and height of the rectangle ($w = 2.4h$).

The area can be relatively easily calculated for a binary image by totalling the number of pixels present for each shape. However calculating the perimeter for binary images is more complex [6, 7]. The initial approach was to simply count the number of 8-connected pixels on the edge of the coloured patch (those which have at least one 4-connected neighbouring background pixel). This gave quite a wide spread of descriptor values, particularly for the square and rectangle (see Table 1, first column). The significant overlap in the range of values meant that the recognition results were not robust, with 38% of the robots misclassified. All of the descriptor values were significantly lower than the theoretical values calculated above because the perimeter along the diagonals is significantly under-estimated.

Table 1. Range of complexity descriptors from the test images.

	Simple perimeter	Weighted perimeter
Circle	9.60 – 11.08 $\mu = 10.54$, $\sigma = 0.52$	14.06 – 15.94 $\mu = 14.93$, $\sigma = 0.69$
Square	8.65 – 12.60 $\mu = 10.70$, $\sigma = 1.61$	15.88 – 19.56 $\mu = 17.59$, $\sigma = 1.41$
Rectangle	10.64 – 14.40 $\mu = 12.56$, $\sigma = 1.42$	18.55 – 23.03 $\mu = 20.57$, $\sigma = 1.56$
Misclassification rate	38%	21%

The second approach was to compensate for the under-estimation along the diagonals by giving more weight to diagonal pixels. This was accomplished by giving a weight of 1.5 to edge pixels which had more than one background pixel in its 4-connected neighbourhood, something which can easily be detected locally.

This perimeter measure tended to slightly over-estimate diagonal lengths, giving descriptors that were slightly higher than expected. However, the results are significantly improved (see Table 1, second column), although there was still some overlap between the ranges. Using thresholds of 15.75 between circles and squares, and 19.0 between squares and rectangles gave a 21% misclassification rate – still too high to be practical.

3.2 Moments

The difficulty in accurately measuring the perimeter led to a different approach to estimating the complexity. Moments are another common method of characterising shape [5], with the spread of a shape being characterised by its second moment (the variance about the centre of gravity). Objects which are more spread would have a higher second moment. In two dimensions,

$$\sigma_x^2 = \frac{\sum (x-\overline{x})^2}{A}, \quad \sigma_y^2 = \frac{\sum (y-\overline{y})^2}{A} \tag{6}$$

The sum of horizontal and vertical variances is invariant to rotation, but has units of distance squared. Therefore, to derive a scale invariant spread descriptor, the moment invariant can be normalised by the area:

$$Spread = \frac{\sigma_x^2 + \sigma_y^2}{A} \tag{7}$$

Assuming that the origin is at the centre of gravity then the ideal descriptor values for the different shapes are then

$$Spread_{circle} = \frac{\iint \left(x^2+y^2\right) dxdy}{A^2} = \frac{\frac{2}{4}\pi r^4}{(\pi r^2)^2} \approx 0.159 \tag{8}$$

$$Spread_{square} = \frac{\frac{1}{12}(w^3 h + w h^3)}{(wh)^2} \approx 0.167 \tag{9}$$

$$Spread_{rectangle} = \frac{w^2+h^2}{12wh} \approx 0.235 \tag{10}$$

where w and h are the width and height of the rectangle respectively (and $w = 2.4h$). Central moments cannot be accumulated directly, as this requires knowing the centre of gravity in advance. However

$$\begin{aligned} \sigma_x^2 &= \frac{\sum (x-\overline{x})^2}{A} = \frac{\sum x^2}{A} - \frac{2\overline{x}\sum x}{A} + \overline{x}^2 \\ &= \frac{\sum x^2}{A} - \left(\frac{\sum x}{A}\right)^2 \end{aligned} \tag{11}$$

and similarly for σ_y^2, allowing central moments to be derived from four incremental accumulators:

$$Spread = \frac{\frac{\sum \left(x^2+y^2\right)}{A} - \left(\frac{\sum x}{A}\right)^2 - \left(\frac{\sum y}{A}\right)^2}{A} \tag{12}$$

The range of spread values for the test images is listed in Table 2. The values for the circle and square are slightly higher than expected, and have some overlap. Those for the rectangle are less than expected. Using thresholds of 0.1690 between circles and squares, and 0.1910 between squares and rectangles gave a 13% misclassification rate. While this is a significant improvement over the complexity descriptor, it is still too high to be practical.

Table 2. Range of spread descriptors from the test images.

	Spread
Circle	0.1625 – 0.1681 $\mu = 0.1645$, $\sigma = 0.0019$
Square	0.1634 – 0.1904 $\mu = 0.1743$, $\sigma = 0.0096$
Rectangle	0.1904 – 0.2299 $\mu = 0.2075$, $\sigma = 0.0134$
Misclassification rate	13%

One of the main reasons for the similarity in values between circles and squares is that squares (especially those oriented diagonally) tend to develop more rounded corners as a result of segmentation, making them more circular. It is the corners that increase the spread of squares relative to circles, so losing these makes them more similar. Conversely, circles tend to become more square-like as a result of distortions introduced from the Bayer pattern demosaicing. For larger circles, this effect would be less noticeable, but the smaller size of the circles amplifies this effect. Examples of the ambiguous circles and squares (with overlapping spread descriptors) are shown in Figure 3.

Similarly, distortion has resulted in identical spread values for the square and rectangle shown in Figure 4. The square was in the corner of the playing area, and lens distortion has compressed the radius, giving it an aspect ratio similar to that of the rectangle.

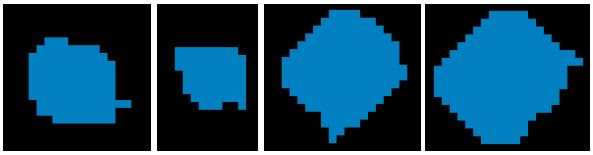

Figure 3. Ambiguous circles and squares. From left to right: blue circles from figures 2(a) and 2(b), blue squares from figures 2(a) and 2(c).

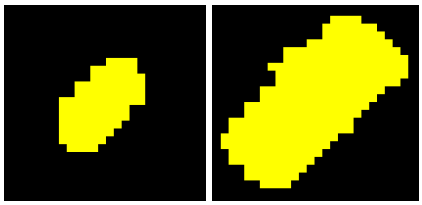

Figure 4. Ambiguous squares and rectangles. Left yellow square and right yellow rectangle both from figure 2(c).

3.3 AREA

Each of the shapes has a different area. However, this on its own is insufficient for distinguishing between the different identities for three reasons. First, the lens distortion has a significant effect on the area of the patch, with the biggest reduction in the corners of the playing area. Second, the illumination is lower around the edges of the playing area, and the dominant effect of this is also a reduction in the area detected. Third, Bayer pattern demosaicing also introduces an uncertainty in the area measurement, increasing the range of values within each of the shape classes. This results in significant overlap between the classes, as is seen in the first column of Table 3.

Table 3. Range of area descriptors from the test images.

	Unnormalised area	Normalised by mean orientation patch area	Normalised by max orientation patch area
Circle	36 – 135 $\mu = 86, \sigma = 34$	0.73 – 1.54 $\mu = 1.06, \sigma = 0.31$	0.65 – 1.25 $\mu = 0.93, \sigma = 0.26$
Square	89 – 204 $\mu = 158, \sigma = 36$	1.32 – 1.77 $\mu = 1.51, \sigma = 0.16$	1.19 – 1.52 $\mu = 1.31, \sigma = 0.14$
Rectangle	179 – 309 $\mu = 260, \sigma = 45$	2.30 – 2.78 $\mu = 2.55, \sigma = 0.18$	2.01 – 2.65 $\mu = 2.28, \sigma = 0.22$
Misclass. rate	13%	4%	13%

However, in the vicinity of the team patch, there are two orientation patches. These will be affected similarly by lighting, and undergo similar distortion. This should enable the area of the orientation patches to be used to normalise the area of the team patch. Normalising with respect to the average of the orientation patches:

$$A_{normalised} = \frac{2A_{team}}{A_{green} + A_{pink}} \tag{13}$$

gives the results listed in the second column of Table 3. There is generally good separation between the classes with a misclassification error rate of 4%.

The segmented image for the one outlier, the blue circle from the image in figure 2(b), is shown in Figure 5. The problem is caused by the green patch only partly being detected (primarily as a result of lower illumination in the corner). The reduced area of the green increases the normalised area, resulting in the misclassification.

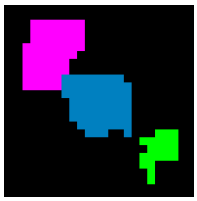

Figure 5. Coloured patches of outlier robot (blue circle team patch from Figure 2(b)).

Rather than using the mean of the orientation patches, selecting the maximum orientation patch solves the problem with this particular image

$$A_{normalised} = \frac{A_{team}}{\max\left(A_{green}, A_{pink}\right)} \tag{14}$$

Unfortunately, this reduces the separation between classes resulting in more overlap, increasing the error rate.

4. COMPARISON AND DISCUSSION

Of the three methods, the normalised area is the easiest to calculate, requiring only a single area accumulator for each detected region. This method also gave the best results of the three methods tested.

The next most complex was the compactness or complexity descriptor. This requires 2 accumulators, one for area and the other for perimeter. The main difficulty is reliably estimating the perimeter, especially for the relatively small region sizes associated with the identity patches. The perimeter is also quite sensitive to segmentation error, and in particular the blocky or zig-zag edges resulting from Bayer pattern interpolation. While a more accurate estimate of the perimeter can be obtained by smoothing the detected curve [6], such smoothing becomes complex to implement within a streaming framework.

The method based on moment invariants overcame the difficulties associated with estimating the perimeter. However it is the most complex method, requiring four accumulators for each detected region in order to be able to calculate the moment invariant. The biggest difficulty with this method was distinguishing between squares and circles because the descriptor values were close, and segmentation errors tended to make them closer. This makes setting of threshold values for the descriptor hard and highly sensitive to changes.

The classification errors highlight the need for careful tuning of the colour thresholds for performing segmentation. As a result of noise and dynamic changes in lighting, occasional errors are inevitable. Since there is only one robot of each identity, such errors can be detected when multiple robots of the same identity are found. In such cases, there are at least two potential methods for resolving such errors:

- By examining the descriptors for the multiple robots classified as the same identity, it may be possible to distinguish between the robots. For example, the smaller descriptor is more likely to belong to the circle than the square.
- The spatio-temporal context can be considered to resolve the errors. Knowing which robot was where in the previous frame will enable the identity of the ambiguous robots to be determined.

5. CONCLUSION AND FUTURE WORK

In theory, and with ideal images, any of the methods discussed would work in determining the identities of the robots. In practise, with limited resolution, a discrete pixel grid, and pixel labelling errors resulting from pre-processing and segmentation mean that misclassifications are inevitable. Particular care is required when making measurements involving perimeter, and colour thresholding.

Having determined by analysing the images in Matlab that the normalised area provides the most reliable identification of the robots, the next step is to implement the measurement within the FPGA.

6. REFERENCES

[1] Bailey, D., Sen Gupta, G. and Contreras, M. 2012. Intelligent camera for object identification and tracking. In *1st International Conference on Robot Intelligence Technology and Applications* (Gwangju, Korea, 16-18 December, 2012). 1003-1013. DOI= 10.1007/978-3-642-37374-9_97.

[2] Bailey, D. G. 2011. *Design for embedded image processing on FPGAs*. John Wiley and Sons (Asia) Pte. Ltd., Singapore. DOI= 10.1002/9780470828519.

[3] Bramberger, M., Doblander, A., Maier, A., Rinner, B. and Schwabach, H. 2006. Distributed embedded smart cameras for surveillance applications. *IEEE Computer* 39, 2, 68-75. DOI= 10.1109/MC.2006.55.

[4] Contreras, M., Bailey, D. G. and Sen Gupta, G. 2013. FPGA implementation of global vision for robot soccer as a smart camera. In *2nd International Conference on Robot Intelligence Technology and Applications* (Denver, Colorado,

USA, 18-20 December, 2013). 657-665. DOI= 10.1007/978-3-319-05582-4_56.

[5] Davies, E. R. 2005. *Machine vision: Theory, algorithms, practicalities*. Morgan Kaufmann, San Francisco, USA.

[6] Ellis, T. J., Proffitt, D., Rosen, D. and Rutkowski, W. 1979. Measurement of the lengths of digitised curved lines. *Computer Graphics and Image Processing* 10, 4, 333-347. DOI= 10.1016/S0146-664X(79)80042-8.

[7] Kulpa, Z. 1977. Area and perimeter measurement of blobs in discrete binary pictures. *Computer Graphics and Image Processing* 6, 5, 434-451. DOI= 10.1016/S0146-664X(77)80021-X.

[8] Ma, N., Bailey, D. and Johnston, C. 2008. Optimised single pass connected components analysis. In *International Conference on Field Programmable Technology* (Taipei, Taiwan, 8-10 December, 2008). 185-192. DOI= 10.1109/FPT.2008.4762382.

[9] Russ, J. C. 2002. *The image processing handbook*. CRC Press, Boca Raton, Florida.

[10] Sen Gupta, G., Bailey, D. and Messom, C. 2004. A new colour-space for efficient and robust segmentation. In *Image and Vision Computing New Zealand (IVCNZ'04)* (Akaroa, NZ, 21-23 November, 2004). 315-320.