A Novel Approach to Ball Detection for Humanoid Robot Soccer

David Budden, Shannon Fenn, Josiah Walker and Alexandre Mendes

School of Electrical Engineering and Computer Science
Faculty of Engineering and Built Environment
The University of Newcastle, Callaghan, NSW, 2308, Australia.
{david.budden,shannon.fenn,josiah.walker}@uon.edu.au
alexandre.mendes@newcastle.edu.au

Abstract. The ability to accurately track a ball is a critical issue in humanoid robot soccer, made difficult by processor limitations and resultant inability to process all available data from a high-definition image. This paper proposes a computationally efficient method of determining position and size of balls in a RoboCup environment, and compares the performance to two common methods: one utilising Levenberg-Marquardt least squares circle fitting, and the other utilising a circular Hough transform. The proposed method is able to determine the position of a non-occluded tennis ball with less than 10% error at a distance of 5 meters, and a half-occluded ball with less than 20% error, overall outperforming both compared methods whilst executing 300 times faster than the circular Hough transform method. The proposed method is described fully in the context of a colour based vision system, with an explanation of how it may be implemented independent of system paradigm. An extension to allow tracking of multiple balls utilising unsupervised learning and internal cluster validation is described.

Keywords: Robotics, robotic soccer, computer vision, feature extraction, object recognition, clustering

1 Introduction

The problem of developing a team of humanoid robots capable of defeating the FIFA World Cup champion team, coined "The Millennium Challenge" [9], has been a milestone that has driven research in the fields of artificial intelligence, robotics and computer vision for over a decade. One crucial skill of soccer, the accurate, robust and efficient determination and tracking of ball size and location, has proven to be a challenging subset of this task and the focus of much research [11–13, 16]. With the evolution of robot platforms and subsequent advances in processor performance over the last decade, from the 384 MHz RISC-based processors of the Sony AIBO ERS-210 (2002) to the 1.6 GHz Intel Atom processors of the Robotis DARwIn-OP [7] platform (2012), the temporal and spatial complexity of feature extraction algorithms to solve this task has grown accordingly.

With past research suggesting that colour-based algorithms are suboptimal for object recognition in a RoboCup environment [12, 13], particularly in the presence of varying lighting conditions, a paradigm shift from colour-based to shape-based feature extraction has been evident amongst RoboCup teams [12]. This shift has been amplified as a result of the evolution of RoboCup rules, with "colour-coded" objects, such as landmark beacons and uniquely-coloured goals, being phased out entirely from the Standard Platform League to facilitate convergence with the FIFA rules of human soccer [2, 9].

Despite this trend, it is important to note that the paradigm-shift from colour-based to shape-based feature extraction is not universal, with many RoboCup teams, including the University of Newcastle's *NUbots* and 2012 kid-sized humanoid league champions *Team DARwIn*, depending primarily on colour look-up tables (LUTs) to facilitate the process of feature extraction. With this in mind, this paper presents a method of ball detection which can be implemented independently of the adopted paradigm, requiring only a set of points marking the edges of potential salient features. The algorithm is very efficient, implementing only basic geometric operations, and yet effective at locating the ball from the opposite side of a SPL RoboCup field (up to 5 meters). This method, which was effectively utilised by the NUbots team at RoboCup 2012 (Mexico City), is demonstrated to be robust against both considerable levels of noise and occlusion, and can readily be extended to cater for a non-RoboCup environment with multiple present balls.

The remainder of this paper, firstly, presents a description of how a ball candidate can be determined in the context of a colour-based vision system. Extensions of this method to a shape-based system are described, in addition to detailing a method by which multiple candidates can be generated to facilitate tracking of multiple balls. The refinement of these candidates to calculate exact ball position and size is then described, for both ideal and occluded ball scenarios. Finally, the accuracy and efficiency of the algorithm is compared to previous ball detection approaches, including the previous NUbots system [1], implementing Levenberg-Marquardt least squares circle fitting [10]; and a circular Hough transform based method [16], similar to those implemented by many RoboCup teams [11, 15].

2 Ball Detection in Context

In computer vision, a mapping from an arbitrary 3-component colour space C to a set of colours M assigns a class label $m_i \in M$ to every point $c_j \in C$ [5]. If each channel is represented by an n-bit value and $k = |M|$ represents the number of defined class labels, then

$$C \to M,$$

where

$$C = \{0, 1, \ldots, 2^n - 1\}^3 \quad \text{and} \quad M = \{m_0, m_1, \ldots, m_{k-1}\}.$$

Where computational resources are limited, the colour segmentation process is performed off-line, with the resultant mapping represented in the form of a $2^n \times 2^n \times 2^n$ look-up table (LUT). This LUT can then be used for efficient, real-time colour classification [5], and as such colour-based vision systems utilising LUTs are still commonplace amongst RoboCup teams [1]. Specifically, the NUbots vision system, for which this ball detection method was initially developed, adopts the following methodology:

1. *Generate scan lines:* As current processor limitations do not allow complex operations over every pixel of a 2-megapixel image without significant framerate reduction, only the pixels along a set of vertical and horizontal *scan lines* are considered for candidate determination. This method is preferred over reduced camera resolution, as it provides the same performance increase (i.e. the same number of pixels are considered) whilst still allowing for small, high resolution portions of the image to be processed to resolve finer detail. Scan lines may be either equidistant on the image plane, or spaced in such as way as to be equidistant on the field plane (requires robot kinematics data).
2. *Determine field border:* Determination of the field border, or *green horizon*, allows for specific knowledge of the RoboCup environment to be applied to reduce the required image processing. For example, a ball and field lines should only ever be found beneath the horizon, whereas the majority of the goal post area will be found above. Starting at the top of the image, each pixel along each vertical scan line is inspected until a certain threshold of consecutive green pixels is exceeded, at which stage the top green pixel coordinates are added to a list of points. The green horizon then becomes the upper convex hull of these points, determined by a modified implementation of Andrew's monotone chain algorithm [3].
3. *Generate colour transitions:* Processing of the image to locate potential field object candidates is a *colour transition* level operation. To generate colour transitions, each pixel along each scan line is considered, and wherever the colour class label of a pixel differs from that of the previous adjacent pixel, a transition is generated. The information stored in each transition includes its (x, y) image coordinate, start colour class label and end colour class label.
4. *Determine candidates:* Colour transitions are considered to determine potential field object candidates. This process may or may not consider transitions of opposite direction or orthogonal orientation as equivalent (e.g. ball detection does, but goal detection does not).
5. *Refine candidates:* The area surrounding each candidate is processed at a pixel level to determine the exact location, dimensions and confidence of a particular field object.

Steps 1-3 simply describe one method by which a series of points, representing the positions of edges of various image features, may be generated. Any system capable of returning equivalent information, whether it be primarily colour, intensity gradient or shape-based, may be implemented as a substitute. As such, the ball detection method described may be implemented somewhat independently of the adopted paradigm.

Sect. 3 and 4 describe how steps 4 and 5 of the above list above are realised for the proposed ball detection method.

3 Determining Candidates

As outlined in Sect. 2, given a set of points corresponding approximately with edges of image features (herein assumed to be colour transitions, consistent with the NUbots vision system), the next step of ball detection is to determine the (x, y) image coordinates of potential ball candidates. The remainder of this section, firstly, describes a simple and computationally efficient method by which a single candidate may be determined. Finally, a generalised method is proposed, which utilises unsupervised learning to allow for the determination of any number of ball candidates.

3.1 Single Ball

Given a set of colour transitions, the first step of determining the ball candidate is to ignore all transitions which do not fulfill the following criteria:

- Must have a start or end colour class label consistent with the ball colour (typically orange).
- Must be located beneath the green horizon.

Transitions are considered independent of their direction, i.e. an orange-white transition is equivalent to a white-orange transition. Following this, the candidate position is calculated as the geometric mean of the transition coordinates. Concretely, given a set of transitions $\{t_1 = (x_1, y_1),\ \ldots,\ t_n = (x_n, y_n)\}$,

$$p_{cand} = \left(\left(\prod_{i=1}^{n} x_i \right)^{1/n}, \left(\prod_{i=1}^{n} y_i \right)^{1/n} \right).$$

To prevent arithmetic overflow in a noisy image, in which several hundred transitions may be present, the following observation is utilised:

$$\left(\prod_{i=1}^{n} x_i \right)^{1/n} = \left(\prod_{i=1}^{k} x_i \right)^{1/n} \times \left(\prod_{i=k+1}^{2k} x_i \right)^{1/n} \times \cdots \times \left(\prod_{i=n-k+1}^{n} x_i \right)^{1/n}.$$

Maximum computational efficiency is obtained by determining the largest value of k for which the data type chosen to store the intermediate value is guaranteed not to overflow. Concretely, for an image of width w pixels and a data type of length n bits,

$$k = \lfloor \log_w (2^n) \rfloor.$$

As an example, given a full HD image (1920×1080 resolution) and a C++ `unsigned long long` data type[1] ($n = 64$ bits), k is calculated to be 5.

[1] Increasing k will also reduce the average rounding error for integer data types.

3.2 Multiple Balls

Given a non-RoboCup environment with multiple balls, the process of ball detection can be extended simply by replacing the *determine candidates* module (see Sect. 2) by a generalised module capable of determining multiple candidate points. In an environment where the maximum number of balls is known a priori, this is accomplished via k-means clustering [8, 14]. Concretely, given a set of m data points $P = \{x^{(1)}, \ldots, x^{(N)}\}$ ($x^{(i)} \in \mathbb{R}^m$), k-means clustering attempts to partition P into K sets (known as *clusters*) $\mathbf{S} = \{S_1, \ldots, S_K\}$ such that the following objective function J is minimised.

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2,$$

where c^i is the index of the cluster $(1, \ldots, K)$ to which data point $x^{(i)}$ is currently assigned, μ_k is the *cluster centroid* of S_k ($\mu_k \in \mathbb{R}^n$), and therefore $\mu_{c^{(i)}}$ is the centroid of the cluster to which $x^{(i)}$ has been assigned [8, 14]. This is accomplished via the repeating the following two-step algorithm until convergence.

Step 1: Assignment step:

$$S_i^{(t)} = \{x^{(p)} : \|x^{(p)} - \mu_i^{(t)}\| \leq \|x^{(p)} - \mu_j^{(t)}\| \; \forall \, 1 \leq j \leq k\}.$$

Step 2: Update step:

$$\mu_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x^{(j)} \in S_i^{(t)}} x^{(j)}.$$

The resultant *cluster centroids* form the ball candidates. The benefits of utilising k-means clustering for determining candidates are twofold. Firstly, compared to other common clustering techniques such as mean shift and expectation maximisation, k-means is computationally efficient, with time complexity $\mathcal{O}(Km)$ [4]. In addition, as clustering only takes place over the set of colour transitions and very few iterations are required, this method is able to be executed in real time on the DARwIn-OP platform [7]. Secondly, k-means utilises an implicit representation of the underlying probability distribution as a superposition of spherically symmetric distributions [14], which performs well given a set of colour transitions positioned approximately on the circular border of a ball.

For an image where the number of balls, b, is known a priori, the number of clusters, K, may simply be set to equal the number of balls. In general, for an environment where the number of balls is known but with no guarantee every ball is present in a given image, an *internal cluster validation criteria* (such as the *Dunn's based index* [5, 6]) is applied to the each cluster for $K = \{1, \ldots, b\}$, with the K value yielding the best results indicating the number of balls in the current image.

4 Determining Location and Size

As outlined in Sect. 2, once the ball candidate(s) have been determined, the final step in ball detection is to inspect each candidate point to calculate the exact location, dimensions and confidence of the ball itself. The remainder of this section, firstly, describes a computationally efficient method by which the position and size may be determined, assuming an occlusion-free ball. Next, a generalised version of this method is explained, which extends the functionality to deal with balls suffering from up to 50% occlusion, either from a direction parallel to the x or y-axes (4-*point occlusion detection*) or an arbitrary direction (n-*point occlusion detection*). Finally, systems of verifying the correctness of the ball detection results are described; this is particularly vital in a multiple candidate scenario.

4.1 Occlusion-Free

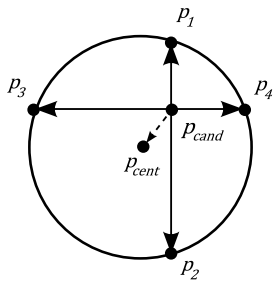


Fig. 1. *Center update* method for determining center of a non-occluded ball.

Given a candidate point p_{cand}, as determined by one of the two methods described in Sect. 3, the process of calculating the exact dimensions of an occlusion-free ball is a straightforward process that operates at a pixel level to maximise the detail which can be resolved. Concretely, given an image where each possible pixel value maps to some colour class label via a look-up table (LUT), each pixel along the x and y-axes (relative to origin p_{cand}) is inspected until a certain threshold, ϑ, of consecutive "non-ball" pixels is exceeded[2], at which time the last-considered pixel belonging to the ball is stored. The four resultant points, $p_1, \ldots, p_4$ (see Fig. 1), therefore correspond ideally with four points on the edge of circle.

As balls in images are rarely perfect circles (due to motion blur and pixel quantization) and LUT mappings are often noisy (due to variations in ambient illumination), the ball location and dimensions are not determined by substituting

[2] In the NUbots vision system, "non-ball" pixels are simply those which do not map to the orange colour class label.

the above points into the equation of a circle. Instead, a single-iteration *center update* method is applied to find the exact center of the ball. For a non-occluded ball, the center update consists of two steps:

1. *Determine center:* Considering the points $p_1 = (x_1, y_1),\ \ldots,\ p_4 = (x_4, y_4)$ (see Fig. 1), the center of the ball, p_{cent}, is calculated as
$$p_{cent} = \left(\frac{x_3 + x_4}{2}, \frac{y_1 + y_2}{2} \right).$$
2. *Determine diameter:* Considering p_{cent} as the origin, each pixel along the x and y-axes is inspected until a certain threshold, ϑ, of consecutive "non-ball" pixels is exceeded, at which time the last-considered pixel belonging to the ball is stored. Considering the resultant points, $p'_1,\ \ldots,\ p'_4$ (labeled in the same orientation as $p_1,\ \ldots,\ p_4$, see Fig. 1), the diameter of the ball, d, is calculated as
$$d = \max\left\{ ||p'_1 - p'_2||, ||p'_3 - p'_4|| \right\}.$$

It can be demonstrated that, given a candidate point p_{cand} within some ideal circle C, this method is guaranteed to yield correct results[3]. An example is illustrated in Fig. 2a.

4.2 4-Point Occlusion Detection

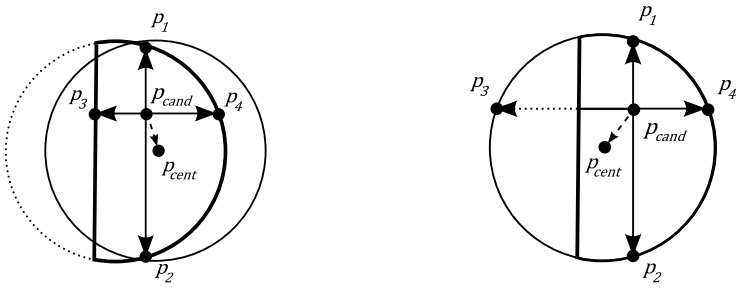


Fig. 2. Error in applying initial *center update* method to occluded ball (a), and the corrected result of applying *4-point occlusion compensation* (b).

Although the threshold parameter ϑ may be tuned to compensate for classification noise, the method presented in Sect. 4.1 is unable to deal with any ball occlusion. This is illustrated in Fig. 2a; any occlusion from any side of the ball

[3] Proof omitted. Available at: http://www.davidbudden.com/research/ball-detection/

will result in a shift of the ball center p_{cent} and reduction of diameter d. Fortunately, the process of *4-point occlusion detection* may be used to compensate for occlusion of the ball from a single direction, as illustrated in Fig. 2b. This method requires the addition of the following three steps to those presented in Sect. 4.1:

1. *Detect occlusion:* Consider a ray, r_1, with initial point p_{cent} and passing through p_1. Starting at p_1, inspect φ pixels in the direction of r_1, where φ is the *occlusion sensitivity* parameter. If none of these are "field"[4] pixels, mark p_1 as occluded. Repeat for $p_2, \ldots, p_4$.
2. *Determine if correction is required:* If none of the points $p_1, \ldots, p_4$ are marked as occluded, no ball occlusion is detected, and the method of updating center and diameter reduces to that of Sect. 4.1. Likewise, if more than one point is marked as occluded, the ball is occluded from multiple directions; this is a scenario unable to be corrected by 4-point occlusion detection, so the method also reduced to that of Sect. 4.1 (with error).
3. *Correct points:* Assume p_3 is marked as occluded. The position of p_3 is updated such that its distance from p_4 becomes exactly $\max\{||p_1 - p_2||, ||p_3 - p_4||\}$, as illustrated in Fig. 4.2b. An equivalent method is applied for all possible occluded points.

Although not guaranteeing correctness, this method presents allows for fast execution, whilst maintaining high accuracy in position determination. To function correctly, 4-point occlusion detection requires the direction of the occlusion source be parallel to either the x or y-axes. This limitation can be reduced by extending the method to *n-point occlusion detection*, where n points about the circle C are utilised, $p_1, \ldots, p_n$. Ignoring pixel quantisation and assuming an ideal ball, n-point occlusion detection yields accurate results for 50% occlusion as $n \to \infty$, independent of the direction and nature of the occlusion. The process of 4-point occlusion detection is illustrated in Fig. 2b.

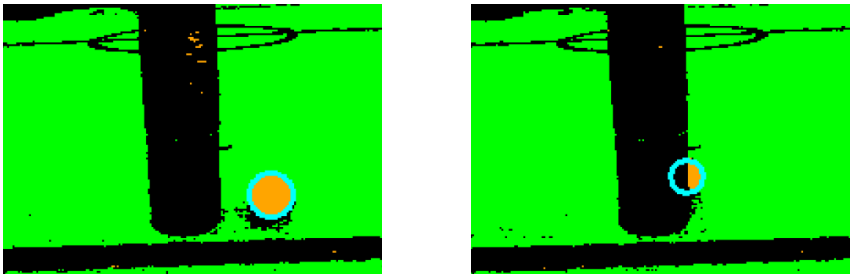
4.3 Ball Verification

Although the proposed method is robust against both noise and ball occlusion, it does not inherently deal well with images where no ball is present; specifically, in an image with no ball, the algorithm will attempt to locate a ball in any appropriate transitions caused by noise or over-classification, such as in a shadowed yellow goalpost. This is corrected by the addition of a *ball verification* stage, where the distance to the potential ball is calculated by two different metrics:

– *Distance by width:* The distance between the robot and the ball is calculated directly from determined ball width:

$$d_w = \sqrt{\left(\frac{r_{cm}}{\tan\left(\gamma_x r_{px}\right)}\right)^2 - h_{cam}^2}\,, \quad \gamma_x = \frac{\theta_x}{w_{img}} \tag{1}$$

[4] In the NUbots vision system, "field" pixels are simply those which map to the green colour class label.



(a) (b)

Fig. 3. Correct ball detection for a LUT-classified RoboCup image, for both non-occluded (a) and 50% occluded (b) ball scenarios (calculated ball represented by cyan circle).

where r_{cm} is the actual radius of the ball (cm), r_{px} is the radius of the ball located in the image (pixels) and h_{cam} is the current height of the camera from the ground (calculated from kinematics data). The *pixel angular width*, γ_x, is a conversion factor that approximates the relationship between the horizontal field of view of the camera, θ_x, and the pixel width of the image, w_{img}. This approximation is most accurate at the center of the frame.

– *Distance to point:* This method utilises a ground-plane projection to calculate the distance to any point on the field, knowing only the (x, y) pixel coordinates, pixel angular width γ_x and height γ_y, and the kinematics data of the robot[5]:

$$d_p = \frac{h_{cam} \tan(\theta_{head} + \beta)}{\cos \alpha},$$

where θ_{head} is the robot head elevation and α and β, the point bearing and elevation, are defined as

$$\alpha = \gamma_x \left(\frac{w_{img}}{2} - x \right), \quad \beta = \gamma_y \left(\frac{h_{img}}{2} - y \right).$$

If the absolute difference between the two calculated distances exceeds a certain error threshold value, the ball is disregarded. This process prevents the detection of false balls, except in the scenario that the offending object is actually ball-sized. Additional checks, such as maximum ball distance and minimum orange pixel density, may also be applied such that false positives can only result from ball-sized, ball-coloured objects positioned on the field of play.

5 Computational Results

The performance of the proposed ball detection method was evaluated by calculating the distance to the ball from its pixel width, as described in Sect. 4.3

[5] γ_y is defined as per γ_x in (1), in terms of θ_y and h_{img}, the image pixel height.

(1). This distance was calculated at 0.5 meter intervals, from a distance of 0.5 to 5.0 meters, for non-occluded, 25% and 50% occluded balls[6]. The accuracy of the algorithm is compared to previous ball detection approaches, including the previous NUbots system [1], implementing Levenberg-Marquardt least squares circle fitting [10]; and a circular Hough transform based method [16] similar to those implemented by many RoboCup teams [11, 15].

Fig. 4 illustrates the experimental results, with the proposed ball detection method indicated by blue, the previous NUbots method by magenta and the Hough transform based method by red. It can be seen from Fig. 4a that, for a non-occluded ball, the proposed method yields the most accurate results for balls greater than 1.5 meters away. For closer balls, the Hough transform method performs best, although the error in both instances is very small. Increasing the level of occlusion from to 25% and 50% increases the distance for which the Hough method is most accurate, to 2.0 m and 3.5 m respectively. Overall, the proposed method produces the most consistently accurate results; the error is consistently less than 10% for non-occluded and 25% occluded balls, and less than 20% for 50% occluded balls. The previous NUbots method performed worst in all cases, and was incapable of detecting balls at a distance greater than 3.5, 3.0 and 2.5 meters for 0%, 25% and 50% occlusion respectively.

The dashed lines in Fig. 4 represent the mean calculated distance error over 100 frames of a stationary robot. To minimise error introduced by the approximated *pixel angular width* (see Sect. 4.3), all images were captured with the ball positioned in the center of the frame. As pixels are an inherently quantised unit, it follows that, for a fixed camera height, certain distances will be less susceptible to error than others. For example, at a distance of 4 meters, the optimal ball width is 4.94 pixels, whereas for 3.5 meters the optimal width is 5.64. As 4.94 is much closer to an integer value than 5.64, it is reasonable that some methods perform better at 4.0 meters than 3.5 (see the blue dashed line in Fig. 4a and c). As this *pixel quantization error* is a function of camera height, which varies as the robot walks, the solid lines (representing the current maximum error) are introduced to Fig. 4 to provide a better indication of the expected error at a given distance.

As robots in humanoid soccer typically suffer from significant processor constraints due to power consumption requirements, it is vital that any implemented algorithm is as computationally efficient as possible. Table 5 contains the execution times for the *refine candidates* module of the proposed ball detection system (see Sect. 2) and equivalent sections of the compared methods, as measured on the DARwIn-OP platform [7]. Results demonstrate that the proposed system executes 1.5 times faster than the previous NUbots system, and over 300 times faster than the Hough transform based system[7].

[6] The official kid size league match ball was utilised for all experiments.

[7] Performance overhead may have been introduced by the OpenCV C++ Hough transform implementation.

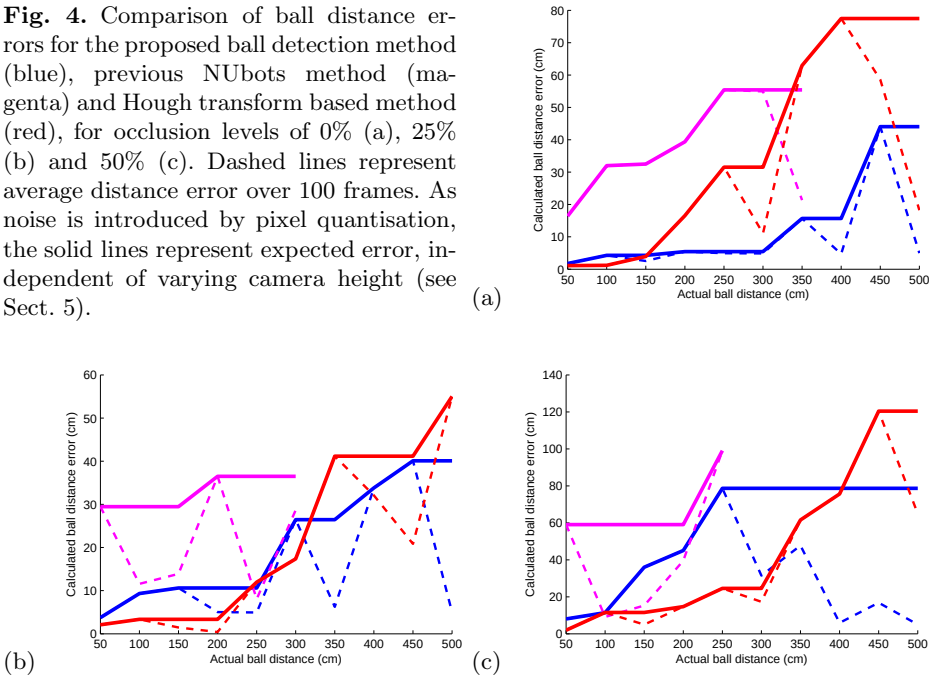


Fig. 4. Comparison of ball distance errors for the proposed ball detection method (blue), previous NUbots method (magenta) and Hough transform based method (red), for occlusion levels of 0% (a), 25% (b) and 50% (c). Dashed lines represent average distance error over 100 frames. As noise is introduced by pixel quantisation, the solid lines represent expected error, independent of varying camera height (see Sect. 5).

Method	μ **(ms)**	σ **(ms)**
Center update w/ 4-point occlusion detection	0.395	0.088
Previous NUbots (least squares circle fitting)	0.615	0.103
Circular Hough transform	128	18.4

Table 1. Execution time mean μ and standard deviation σ for the proposed ball detection method, previous NUbots method and Hough transform based method.

6 Conclusion

The proposed ball detection method overall demonstrated the most accurate results, maintaining an error in calculated ball distance of less than 10% for non-occluded and 25% occluded balls, and less than 20% for 50% occluded balls, over a range of distances from 0.5 to 5.0 meters. The compared Hough transform method performed slightly better for balls closer than a certain distance, which increased from 1.5 to 3.5 meters as occlusion levels were raised. The previous NUbots ball detection method, which implemented Levenberg-Marquardt least squares circle fitting, performed worst for all distances and occlusion levels.

In addition to accurate performance, the proposed method executed over 300 times faster than the Hough transform based method on the DARwIn-OP

platform [7]. Much of this performance gain was leveraged by utilising specific knowledge of the RoboCup environment, such as the ball size and quantity known a priori, in addition to the green field border. Future research will focus on dynamically identifying these salient features in real time, such that similar performance advantages may be leveraged in an arbitrary environment.

References

1. The 2009 NUbots team report (2009), `http://www.robots.newcastle.edu.au/publications/`
2. Robocup standard platform league (nao) rule book (online) (2012), `http://www.tzi.de/spl/pub/Website/Downloads/Rules2012.pdf`
3. Andrew, A.: Another efficient algorithm for convex hulls in two dimensions. Information Processing Letters 9(5), 216–219 (1979)
4. Bishop, C., en ligne), S.S.: Pattern recognition and machine learning, vol. 4. springer New York (2006)
5. Budden, D., Fenn, S., Mendes, A., Chalup, S.: Evaluation of colour models for computer vision using cluster validation techniques. In: (Accepted) RoboCup 2012: Robot Soccer World Cup XVI (LNAI). Springer (2013)
6. Dunn, J.: Well-separated clusters and optimal fuzzy partitions. Journal of cybernetics 4(1), 95–104 (1974)
7. Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.: Development of open humanoid platform DARwIn-OP. In: SICE Annual Conference (SICE), 2011 Proceedings of. pp. 2178–2181. IEEE (2011)
8. Hartigan, J., Wong, M.: Algorithm as 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics) 28(1), 100–108 (1979)
9. Kitano, H., Asada, M.: The robocup humanoid challenge as the millennium challenge for advanced robotics. Advanced Robotics 13(8), 723–736 (1998)
10. Marquardt, D.: An algorithm for least-squares estimation of nonlinear parameters. Journal of the society for Industrial and Applied Mathematics 11(2), 431–441 (1963)
11. Martins, D., Neves, A., Pinho, A.: Real-time generic ball recognition in robocup domain. In: Proc. of the 3rd International Workshop on Intelligent Robotics, IROBOT. pp. 37–48 (2008)
12. Murch, C., Chalup, S.: Combining edge detection and colour segmentation in the four-legged league. In: Proceedings of the Australasian Conference on Robotics and Automation 2004 (2004)
13. Seysener, C., Murch, C., Middleton, R.: Extensions to object recognition in the four-legged league. RoboCup 2004: Robot Soccer World Cup VIII pp. 274–285 (2005)
14. Szeliski, R.: Computer vision: algorithms and applications. Springer-Verlag New York Inc (2010)
15. Velthuis, D., Verschoor, C., Wiggers, A., Cabot, M., Keune, A., Nugteren, S., Egmond, H., Rossum, T., Molen, H., Rozeboom, R., Becht, I., Jonge, M., Prong, R., Kooijman, C., Slaap, R., Visser, A.: Dutch nao team, team description paper for robocup 2012 (2009)
16. Yuen, H., Princen, J., Illingworth, J., Kittler, J.: Comparative study of hough transform methods for circle finding. Image and Vision Computing 8(1), 71–77 (1990)