Illumination Independent Object Recognition

Nathan Lovell

School of CIT, Griffith University, Nathan 4111 QLD, Australia

Abstract. Object recognition under uncontrolled illumination conditions remains one of hardest problems in machine vision. Under known lighting parameters, it is a simple task to calculate a transformation that maps sensed values to the expected colors in objects (and minimize the problems of reflections and/or texture). However, RoboCup aims to develop vision systems for natural lighting conditions in which the conditions are not only unknown but also dynamic. This makes fixed color-based image segmentation infeasible. We present a method for color determination under varying illumination conditions that succeeds in tracking the objects of interest in the RoboCup legged league.

1 Introduction

Vision systems that perform colour based object recognition (as in the SONY Aibo Four-Legged RoboCup League) rely heavily on a static lighting environment where objects of interest have known colors (for example, the ball is orange). This simplifies the vision problem significantly and has resulted in standard vision modules consisting of three steps [1]:

Color segmentation: Each pixel in the image is classified as being either one of the important colours or an unknown colour.
Blob formation: Identify groups of same coloured pixels (blobs).
Object recognition: Blobs are analysed to determine objects.

Of course, if pixels are incorrectly classified, then the blobs inaccurately reflect the features of the object resulting in identification error. Even in known and static lighting environments, specular reflection can make (for example) white pixels look pink and orange pixels look yellow. The Aibo camera is very unsophisticated compared to other digital cameras that have very complex mechanisms for dealing with changing lighting conditions including variable shutter speeds and automatic focusing. Work in other leagues shows how these features can be used effectively [2]. The Aibo camera, however, lacks many of these advantages and it provides only the most simple of lighting modes that must be manually set. It does not even provide a consistent colour sampling across single images [3].

As we move toward using natural lighting conditions, with unknown and dynamic lighting temperature and intensity, and including shadows and indirect and reflected light, the problems are compounded significantly. If the lighting conditions are known, then the change in perceived colours for different lighting conditions is well modelled by a computable mathematical transformation [3].

In unknown lighting conditions the problem of correctly classifying a pixel is very difficult on a pixel by pixel basis.

RoboCup has a strong interest in uncontrolled illumination due to its 2050 target. There has been a stream of papers on lighting conditions over the 2003 and 2004 symposiums [4, 5, 6, 7, 8, 9, 10, 11, 12], however, the implementations in the actual robots has been disappointing. Consider as an illustration the technical challenge regarding variable lighting conditions that has been present in the 4-legged league for some years now. All 24 teams attempted a solution to this problem in 2004. However, the result demonstrated not only the difficulty of the problem but also that current research is far from a solution. While most teams did well in the period when the lights were close to normal levels, as soon as the lights were dimmer no team was able to correctly identify both the ball and the goal, let alone score a goal. Research efforts for uncontrolled illumination for RoboCup have also been reflected elsewhere [13, 14, 15, 16, 17, 18]. Still the results have not convincingly produced a working system.

The main problem with previous approaches has been the concern with creating pixel classifiers for color segmentation of high accuracy and high coverage. We maintain that under changing illumination conditions, this is in essence an impossible task and pixel classification must be determined in the context of the image and, at least partly, by what colour we *expect* that pixel to be. The human eye does this unconsciously. For example, tree leaves at sunset still appear green though in a digital picture very few of the pixels that make up the leaves will be in any way green.

We introduce an efficient object recognition system that is robust to changes in colour intensity and temperature because it does not rely solely on colour classification in order to form blobs. We propose a classifier that is very accurate but only on pixels that are at the core of each color class and essentially refuses to make a decision for most other pixels labeling them as unknown. Our approach works while providing a color label for less than 10% of the pixels in each frame. Instead of using the colour of each pixel to form blobs, we first form the blobs and then use the colours of each pixel within it to tell us the colour of the blob. We can now use the context of the entire image rather than a simple pixel-by-pixel analysis in our object recognition process. Our process then is:

Edge detection: Detect efficiently the edges within the image.

Sparse classification: Classify each pixel that is not determined to be an edge according to the sparse classifier.

Object recognition: Detect the blobs within the image and use the classified pixels within them to determine colour.

We now show details that make each step very efficient. Thus, the entire process has insignificant overhead compared to previous object recognition systems.

There have been other attempts at combining edge detection and colour segmentation to improve object recognition [19] however these have been done with the aim of improving vision system performance and accuracy, not with the aim of overcoming problems associated with variable illumination.

Table 1. Comparison of our edge detection algorithm with well-known Sobel edge detection. The average runtime for the algorithm improves more than 75% for a window size of five. The pixels labeled incorrectly are less than 7% overall. The times in the table are measured in AVW2 profile clicks. AVW2 is our vision testbed program [20].

	Min Runtime	Max Runtime	Avg Runtime	False Positives	False Negatives	Total Incorrect
Sobel	75	129	84.2	-	-	-
Our Algorithm	14	21	20.9	4.1%	2.3%	6.4%

2 Our Fast Edge Detection

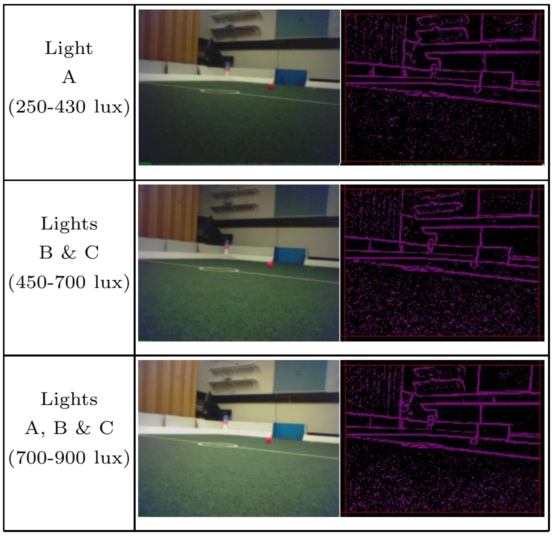
While we are confident that a robust edge detection algorithm would be sufficient for the first step in our process, it must also be very fast. We found that the standard edge detection routines (such as Sobel, Robert's Cross and Canny [21]) are simply too slow to be useful in our real-time processing environment. Therefore, we introduce here a new edge detection routine that executes much faster than traditional edge detection algorithms.

Edge detection routines slide a regular "window" over pixels and determine if the colour between the pixels at the centre of the window varies significantly from the average colour difference over such window. There are variations on this, e.g., Sobel tests hypothesis of edges that could exist within the window, checking each of them.

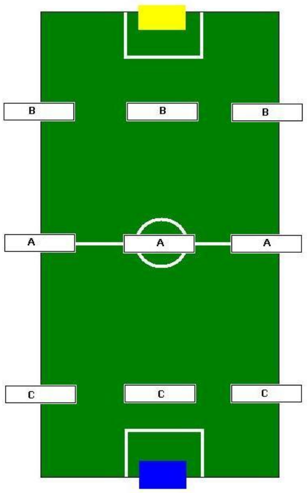
Our edge detection uses a similar idea however we only consider pixels in the row and column of the centre, thus the window we use is not a square but a cross. The difference in pixels at the centre of the window is computed and compared to the average difference between other pixels in both the row and column respectively. If the difference is much higher (either compared to the horizontal difference, or the vertical one), then that this pixel lies on an edge.

Although both our routine and the traditional edge detection algorithms are linear time under Big-O notation[1], there is a big difference in the order of the hidden cost. In traditional edge detection algorithms, each pixel in the window must be compared to each of its neighbors. For a window size w the constant in such algorithms is $2w(w-1) = O(w^2)$. Thus, quadratic on w, and usually very large. For example, a window size of five leads to 40 colour comparisons for each window evaluation and therefore 40 colour comparisons per pixel in the image. Contrast this to our technique where, with size w, the constant is $2w = O(w)$, i.e. linear in w. This means that in the above example, our algorithm will only do 10 colour comparisons per pixel, thus running in a quarter of the time of a traditional edge detection algorithm. There is a very minor trade off in terms of the algorithm quality. Table 1 compares the runtime cost of our algorithm with

[1] Big-O notation expresses the number of operations an algorithm performs, $t(n)$, as a function of the number, n, of input data items. An algorithm is $O(f(n))$ if there is $c > 0$ and $n_1 > 0$ such that $t(n) < cf(n), \forall n > n_1$. Thus, $t(n)$ is $O(ct(n)), \forall c > 0$ and the constant cost, c, is hidden.



(a)



(b)

Fig. 1. (a) Our edge detection routines are very robust to varying illumination conditions. Edge detection remains of high quality. (b) Our experimental setting. Three rows of independently controllable fluorescent lights.

that of a traditional algorithm (Sobel's) a set of 100 images with a (reasonable) window size of five. It also compares the quality of our algorithm against Sobel's by comparing the output of the two algorithms and measuring the percentage of false positives and false negatives.

Despite the use of a smaller window, our edge detection algorithm is very robust (less than 7% error) and remains a solid foundation for the rest of the process as it is particularly resistant to changing lighting conditions. Fig. 1 (a) illustrates the stability. Under the changing illumination conditions of each subsequent image the edge analysis remains remarkably stable. The lighting conditions are described in Fig. 1 (b). Our lab lighting consists of three rows of independently controlled fluorescent lights. The row of lights labeled A runs across the middle of the field. Rows B and C are positioned at the yellow and blue ends of the field respectively. When only row A is on, the field has an inconsistent illumination ranging from 250 to 430 lux. With rows B and C there is a slightly better illumination ranging from 450 to 700 lux. RoboCup conditions are approached only with all three rows of lights on (700 to 900 lux).

Any choice of edge detection algorithm must be implemented carefully to make it feasible to run in real-time on an Aibo. We emphasize an important optimisation that must be performed to make even our improved algorithm suitable. As the window slides over the image, we store and re-use color differences that will occur in comparisons more than once (in fact, proportional to the window size). Consider the pixels at $(10, 10)$ and $(11, 10)$. The difference in colour between these two pixels is calculated for the first time when the window (of size five) is centred on $(5, 10)$ and used in the average to contrast with the difference between pixels at $(5, 10)$ and $(6, 10)$. As the window moves along the tenth row, this difference must be used 10 times. The cost of each comparison is expensive because it involves a three dimensional distance calculation (with a square root). We minimise this cost by keeping a circular buffer of the calculated differences between both the last five pixels and, looking ahead, the next five. Since we iterate across rows first, and then columns, it is also necessary to maintain a circular buffer for each column in the image, in addition to the one for the current row. By using this technique the difference between each two pixels is only calculated once per pixel.

3 Our Sparse Classification

We overcome the problems associated with illumination independence by shifting the focus from colour-based segmentation to edge detection. Colour segmentation in the legged league consists of labeling with one of the environment's colour classes each pixel in the image. We represent this as a colour-segmentation function $c\_class : Y \times U \times V \to C$ that given a triplet (y, u, v) produces the colour class. For the last few years, as many as 10 colours are used for objects of interest in the field (including, the field itself, beacons, goals, opponents, teammates, markings and of course the ball). A representation of the function *c_class* as a complete colour-lookup table that maps every possible YUV triplet to a particular colour class would be extremely large (2^{24} bits). RoboCup researchers have used many techniques to learn the *c_class* function and to represent it. Among these, machine learning and statistical discrimination techniques like linear discriminant, decision trees [22], Artificial Neural Networks [23], Support Vector Machines [24] and instance-based classifiers (k-nearest neighbors and other non-parametric statistics [22]) have been used. However, the state of the art (in terms of accuracy vs speed) for this environment remains a look-up table of characteristic vectors for each of the 3 dimensions [1]. Look-up tables are faster than machine learning artifacts by several orders of magnitude [25].

While all of these methods can learn and represent the 2^{24} bit colour table off-line (colour calibration), they have trouble adapting smoothly and reliably on-line. The problem is that a change in illumination conditions implies a change in the colour table in a non-simple way [24]. Thus, all of these approaches will eventually fail as the colour table increasingly becomes inaccurate in changing illumination conditions.

Whatever the machine learning techniques used and representation of the segmentation function $c\_class : Y \times U \times V \to C$ is, the aim had been to have high accuracy on the entire space of (y, u, v) values and then segment an entire image. Thus, classifiers were built with the requirement to label all pixels which are perceived from a known colour. But, under another set of illumination conditions, the function $c\_class : Y \times U \times V \to C$ is rather different. In fact, one could say the c_class_{IC} depends on the illumination conditions IC. One approach is to detect what IC is frequently during play and switch the colour-segmentation function on board of the robot dynamically [9]. This assumes that one can reliably identify

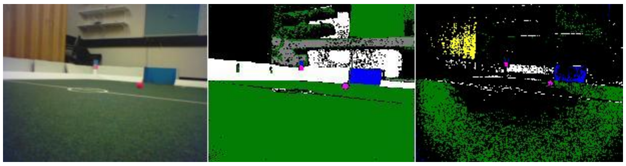


Fig. 2. *Sparse* classification. The image on the left is segmented by a colour classifier under known lighting conditions. The sparse classified image on the right only labels pixels that are in the same color class in a variety of illumination conditions.

IC and have a reasonable rich set of pre-loaded segmentation functions. However, the possible universe of values for IC implies a second stage of classification, namely the reliable identification of the suitable function c_class_{IC}.

In contrast to previous work, we create a classifier that is as *sparse* as possible. That is, we only want to classify a pixel belonging to a colour class if we are sure that it will remain in that class in a wide range of illumination conditions. Our approach is to consider identifying first that region of the YUV space for which given a colour class (say blue) c_class_{IC} remains constant and equal to blue for most values of IC. That is, our experience suggests that there is core region in YUV space where most illumination conditions will regard this as blue. We want our classifier to provide no label for those values where a (y, u, v) triplet fluctuates between green and blue along many illumination conditions.

Fig 2 illustrates this difference. The left segmented image shows the result of a traditional classifier trained with corresponding lighting conditions. The right segmented image results from our idea of a sparse classifier that is trained to classify only those pixels that it can surely labeled correctly across a variety of illumination conditions. We call this "sparse colour segmentation". Standard object recognition systems would find a sparse colour segmentation useless because large blobs of a common colour are non-existent within the image. Our system will not use the colour-labeled pixels for that purpose so, as we will see in the Section 4, a sparse classification is actually preferable.

In supervised learning, a training and test sets are used to produce a classifier. Namely, the function $c\_class : Y \times U \times V \to C$ can be learned from many pairs of the form $[(y, u, v),$ C_CLASS]. Earlier work [25] has shown how to learn a very efficient and succinct representation encodes the classifier as a decision list to take advantage of the parallelism of bytes. This representation decomposes the function *c_class* into simpler functions and observes that the corresponding fast C++ implementation essentially tests conditions as decision lists [26]. The learning algorithm PART [27] as implemented in WEKA [26] has been very successful in learning a representation of *c_class* that is no longer than $256 \times 4 bytes = 1$ KB. The accuracy is above 98% for operating on the same illumination conditions as where the training and test sets were collected.

We describe how to obtain a new classifier that would produce a class label for those regions of the YUV space where c_class_{IC} remains constant across a large number of values for IC. We use a training set T_{IC} to learn a classifier $decision\_list_{IC}$. We repeat this process for 5 different illumination conditions of the same field (additional base classifiers are required if we are moving the field into a different room). We call the resulting classifiers $decision\_list_{IC_i}$, for $i = 1, \ldots, 5$. We then process all source training sets T_{IC_i} used to learn the base-classifiers. For each example $[(y, u, v)$, C_CLASS] we test if all 5 classifiers $decision\_list_{IC_i}$ agree on a classification (and if that classification is C_CLASS). If that is the case, this example is appended into a new global training set T. Otherwise, the example is placed in the training set with the label replaced by the value unknown. Once all training examples for the case classifiers have been processed, the new training set T is used to learn the sparse classifier. All learning stages use the PART algorithm and are represented as decision lists. We should indicate that we were able to obtain a sparse classifiers that required 256×14 bits (that is, 448 bytes) and had a precision of 99%.

4 Our Blob Forming

The final step before object recognition is blob forming. We take advantage of the edges we have found in the first step. The edges of our blobs result from a border-following algorithm that creates a list of pixels that represents the boundary of each region of interest in the image. This avoids the need for complex union-find methods to build blobs from colour labeled pixels. Union-find algorithms are quadratic in complexity on the average diameter of the blob because they must inspect every pixel in the blob. Border tracing is a linear operation on the average diameter of the blob so is clearly a preferable method.

The edges themselves contain insufficient information to locate regions of the image which are interesting so we use the sparse colour information we have obtained by classification. We are fairly sure that anything labeled as, say, blue, is actually blue in any lighting condition. Therefore a blob that contains many more blue pixels than any other colour is likely to be a blue object. We use our colour labeled pixels as a set of sample points and use Algorithm 1 to establish the blob that each sample point lies in. As we build the list of pixels describing the blob in Line 4, it is trivial to compute other properties of the blob such as its bounding box or relation to the horizon of the image. A frequency histogram of colour classes found within the blob is also built by Line 7.

In a naive implementation of this algorithm, the test in Line 3 could be computationally time consuming. Since there will be many blobs, and each blob will contain many pixels, there will be very large list of pixels to search to test if our current pixel is already assigned to a blob. We recommend several optimisations that make this algorithm much more efficient. Instead of searching the existing boundary lists, we label (on the segmented image) each pixel in the boundary of a blob with the blob ID of that pixel. This process is performed in Line 4 of the algorithm as the boundary list is being constructed. Since we have 8-bits per pixel in a segmented image, and only 15 or so colour labels, we are free to use the other 240 values as identifiers for blobs. In this way the test in Line 3 is a single memory access to determine a blob ID.

Algorithm 1. Blob Form

```
Input: A set, $P$, of seed points. An image where pixels on edges have been identified.
Output: A set, $B$, of blobs where each blob is represented by the pixel list that forms
    its boundary.
1: for all Points, $p \in P$ do
2:     Find the pixel, $e$, in the edge above $p$.
3:     if $e$ is not in a blob $b \in B$ then
4:         Trace boundary to find pixel list describing a new blob $b'$.
5:         Insert $b'$ into $B$.
6:     else
7:         Update colour information on $b$.
8:     end if
9: end for
```

5 Our Object Recognition

Fig. 3 illustrates how our object recognition system, based on fast edge detection, is preferable in changing illumination conditions to a traditional object recognition system based on colour segmentation. The images in the left column are the source images in several illumination conditions. Refer again to Fig. 1 for a description of the illumination conditions. We illustrate our illumination independent object recognition by a series of images that represents different stages in our vision processing pipeline. Even though the pixel classification changes in each image (column two), the blob forming stage (column three) remains fairly stable because it is based on our fast edge detection routine. Therefore the objects on the field (a blue/pink beacon, the blue goal and a pink[2] ball) are recognised correctly in each image despite the illumination changes.

The task of object recognition is very similar to traditional systems because the blobs are recognised accurately in a wide variety of lighting conditions. We have several interesting blobs that require further investigation to determine if they are actual objects such as the goals, beacons or the ball. One advantage of our particular system is that blobs inherently are composed of a list of edge pixels. In prior work we have demonstrated a linear time algorithm for straight edge vectorisation [28]. This means that it requires very little processing overhead to vectorise the blob's boundary. This is especially useful when processing the field blobs as it allows us to identify field edges and corners. There are several other situations where having the boundary of the blob already identified is very helpful. For example the centre of the ball is often found by using the mathematical technique of perpendicular bisectors [29]. This relies on knowledge of the boundary of the ball. There are methods now for linear time detection of circles which also require the boundary to be identified [30]. Aside from these, there

[2] The ball is pink in these images rather than the league standard of orange because at the time of writing there was a proposed rule change to use a pink ball. This rule change has since been rejected.

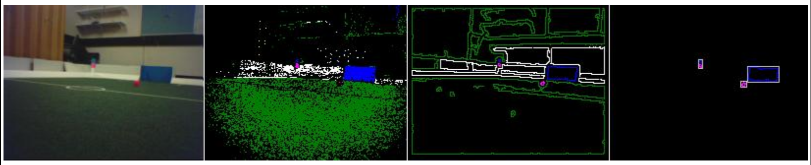
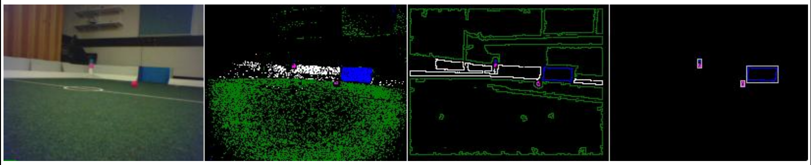
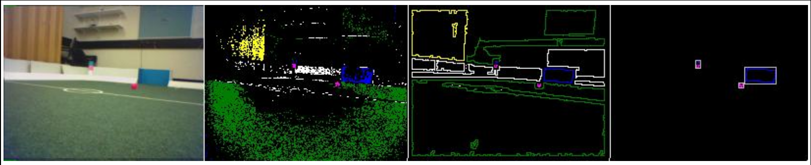
Light A (250-430 lux)	
Lights B & C (450-700 lux)	
Lights A, B & C (700-900 lux)	

Fig. 3. Illustration of our object recognition process under a variety of illumination conditions. Source images are shown before their sparse classification. The result of edge detection can be seen in Fig. 1. The third column shows the result of our blob forming algorithm and the final column illustrates all objects properly identified.

are many techniques in both RoboCup and other literature about mechanisms for analysing blobs to determine the real-world objects properties [31, 32, 3].

We have made available on our website[3] a video that demonstrates our ability to track a standard RoboCup ball in variable lighting conditions. There are four test cases illustrated on the video:

Standard Illumination: Light rows A, B, and C (refer again to Fig. 1 (a)).
Medium Illumination: Light rows A and C.
Low Illumination: Light row B.
Dynamic Shadows in Low Illumination: A solid sheet is passed between the light source and the field in random motion.

The same sparse classifier is used in each of the test cases and all four cases were filmed without rebooting the Aibo. The automatic camera setting on the Aibo (auto white balance) is *off* and the camera is fixed to medium gain, medium shutter speed and fluorescent lighting mode.

This video demonstrates that Aibo tracking the ball in each lighting environment. The lights on the head of the Aibo are indicative of how well it is tracking the ball - bright white lights indicate that it sees the ball clearly and red lights indicate that it is unsure of the ball but it has not lost it completely[4]. No lights

[3] http://www.griffith.edu.au/mipal

[4] This can happen, for example, when the edge detection contains a flaw which allows the border trace algorithm to "escape" the ball at some point around it's border. In this case we end up with a very large, unlikely ball and we wait until the next vision frame to confirm its location.

Table 2. Comparing our old object recognition system to our new one. The old system had no edge detection component and edge detection and classification are merged in the new algorithm. All times are measured in AVW2 profiling clicks described in [20].

	Edge Detection	Classification	Blob Forming	Object Recognition	Total
Standard Algorithm	-	4.2	21.8	12.9	38.9
Our Algorithm	20.9		18.6	12.9	52.4

would indicate a complete loss of the ball although this does not happen in this video. Notice that our object recognition system copes with all of the lighting conditions sufficiently well. Although we see some minor degradation of performance in the final test case, even in this extremely complex lighting environment we illustrate sufficient competence to track a ball. Notice also that our object recognition system does not confuse skin colour with the ball, even in the final test case with low light and dynamic shadows.

Finally, we compare the overall performance of our object recognition algorithm with the performance of our prior one ([25, 31, 29]). Table 2 shows the running time of each stage of the two object recognition algorithms. The tests were done on a set of 100 images. Our new object recognition process is approximately 35% slower than our old one. This is mostly due to the time required for the edge detection step. The algorithm is, however, still well within the performance requirements for running as a real-time vision system on a SONY Aibo.

6 Conclusion

We have been able to overcome the problems associated with illumination independent object recognition by shifting the focus in the process from the step of colour segmentation to that of edge detection. We have developed a very fast edge detection algorithm to make this feasible even on platforms, such as the SONY Aibo, with limited computational capacity. By relying on edges in the real image to form blobs, rather than boundaries in classified images, we have removed the problem of requiring a highly accurate classification. We instead label only those pixels that we are fairly certain will be classified correctly over a range of various lighting conditions.

Traditional colour segmentation algorithms, by nature, do not work in dynamic illumination conditions. They are specifically trained for one particular condition and they become increasingly inaccurate as they are moved further from their initial training condition. Thus the traditional approach of forming blobs based on the results of colour segmentation is severely hampered in any dynamic lighting condition. On the other hand, a suitable classifier can be built for a wide range of lighting conditions provided it is freed from the restriction of being as complete as possible. Thus it is possible to use the colour of pixels, even in highly variable lighting conditions, once the blob is identified in some other way.

We have presented a fast enough edge-detection algorithm to run on the Aibo in RoboCup game conditions. We have used this algorithm as a basis for a new object recognition system that uses colour information but does not rely on training a classifier in a way specific to any single lighting condition. We have been able to produce good results in object recognition over a variety of illumination conditions using this technique.

References

1. Bruce, J., Balch, T., Veloso, M.: Fast and inexpensive color segmentation for interactive robots. In: Int. Conference on Intelligent Robots and Systems, IEEE Computer Society Press (2000)
2. Grillo, E., Matteucci, M., Sorrenti, D.: Getting the most from your color camera in a color-coded world. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
3. Röfer, T., Brunn, R., Dahm, I., Hebbel, M., Hoffmann, J., Jünge, M., Laue, T., M. Lötzsch, W.N., Spranger, M.: German team 2004 - the german national robocup team. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
4. Dahm, I.: Fully autonomous robot color classification. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
5. Cameron, D., Barnes, N.: Knowledge-based autonomous dynamic colour calibration. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
6. Dahm, I., Deutsch, S., Hebbel, M.: Robust color classification for robot soccer. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
7. Jüngel, M., Hoffman, J.: A real-time auto-adjusting vision system for robotic soccer. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
8. Meyer, G., Utz, H., Kraetzchmar, G.: Playing robot soccer under natural light: A case study. In: Proc. of the Int. RoboCup Symposium, Springer-Verlag (2003)
9. Sridharan, M., Stone, P.: Towards illumination invariance in the legged league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
10. Steinbauer, G., Bischof, H.: Illumination insensitive robot self-localization using panoramic eigenspaces. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
11. Gönner, C., Rous, M., Kraiss, K.: Real-time adaptive colour segmentation for the robocup middle size league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
12. Lange, S., Riedmiller, M.: Evolution of computer vision subsystems tasks. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
13. Austin, D., Barnes, N.: Red is the new black - or is it? In: Proc. of the 2003 Australasian Conference on Robotics and Automation. (2003)
14. Finlayson, G., Hordley, S., Hubel, P.: Colour by correlation: A simple, unifying framework for colour constancy. IEEE Transactions on Pattern Analysis and Machine Intelligence **23(11)** (2001)
15. Finlayson, G., Hordley, S.: Improving gamut mapping color constancy. IEEE Tranactions on Image Processing **9(10)** (2000)
16. Finlayson, G.: Color in perspective. IEEE Transactions on Pattern Analysis and Machine Intelligence **18(10)** (1996) 1034–1038
17. Brainhard, D., Freeman, W.: Bayesian colour consistency. Journal of Optical Society of America **14(7)** (1986) 1393–1441

18. Buchsbaum, G.: A spatial processor model for objecct color perception. Journal of Franklin Institute **310** (1980) 1–26
19. Murch, C., Chalup, S.: Combining edge detection and colour segmentation in the four-legged league. In: Proc. of the 2003 Australian Robotics and Automation Association. (2003)
20. Lovell, N.: Real-time embedded vision system development using aibo vision workshop 2. In: Proc. of the Mexican Int. Conference on Computer Science. (2004)
21. Gonzalez, R., Woods, R.: Digital Image Processing. Addison Wesley (1992)
22. Chen, J., Chung, E., Edwards, R., Wong, N., Mak, E., Sheh, R., Kim, M., Tang, A., Sutanto, N., Hengst, B., Sammut, C., Uther, W.: rUNSWift team description. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2003)
23. Kaufmann, U., Mayer, G., Kraetzschmar, G., Palm, G.: Visual robot detection in robocup using neural networks. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
24. Quinlan, M., Chalup, S., Middleton, R.: Techniques for improving vision and locomotion on the sony aibo robot. In: Proc. of the 2003 Australasian Conference on Robotics and Automation. (2003)
25. Estivill-Castro, V., Lovell, N.: Improved object recognition - the robocup 4-legged league. In Liu, J., Cheung, Y., Yin, H., eds.: Intelligent Data Engineering and Automated Learning, Springer (2003) 1123–1130
26. Witten, I., Frank, E.: Data Mining — Practical Machine Learning Tools and Technologies with JAVA implementations. Morgan Kaufmann, California (2000)
27. Frank, E., Witten, I.: Generating accurate rule sets without global optimization. In: Machine Learning: Proc. of the Fifteenth Int. Conference, Morgan Kaufmann (1998)
28. Lovell, N., Fenwick, J.: Linear time construction of vectorial object boundaries. In Hamza, M., ed.: 6th IASTED Int. Conference on Signal and Image Processing, ACTA Press (2004)
29. Estivill-Castro, V., Fenwick, J., Lovell, N., McKenzie, B., Seymon, S.: Mipal team griffith - team description paper. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)
30. Sauer, P.: On the recognition of digital circles in linear time. Computational Geometry: Theory and Applications **2** (1993) 287–302
31. Lovell, N., Estivill-Castro, V.: A descriptive language for flexible and robust object recognition, Springer-Verlag (2004)
32. Seysener, C., Murch, C., Middleton, R.: Extensions to object recognition in the four-legged league. In: Proc. of the Int. Robocup Symposium, Springer-Verlag (2004)