Unsupervised Recognition of Salient Colour for Real-Time Image Processing

David Budden 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,alexandre.mendes}@newcastle.edu.au

Abstract. Humans have the subconscious ability to create simple abstractions from observations of our physical environment. The ability to consider the colour of an object in terms of "red" or "blue", rather than spatial distributions of reflected light wavelengths, is vital in processing and communicating information about important features within our local environment. The real-time identification of such features in image processing necessitates the software implementation of such a process; segmenting an image into regions of salient colour, and in doing so reducing the information stored and processed from 3-dimensional pixel values to a simple colour class label. This paper details a method by which colour segmentation may be performed offline and stored in a static data structure, allowing for constant time dimensionality reduction in an arbitrary environment of coloured features. The machine learning framework requires no human supervision, and its performance is evaluated in terms of feature classification performance within a RoboCup robot soccer environment. The developed system is demonstrated to yield an 8% improvement over slower traditional methods of manual colour mapping.

Keywords: Computer vision, colour vision, robotics, RoboCup

1 Introduction

Szeliski describes image segmentation simply as the task of finding groups of pixels that "go together" [10]. This is an abstract notion that corresponds with an inherently subconscious human process: the ability to look at an image and identify salient features, such as a person, landmark or household item. Hundreds of algorithms exist for image segmentation, with most relying on the assignment of a *feature vector* (containing spatial and/or colour information) to each pixel. For real-time applications, the dimensionality of this feature vector is commonly reduced to contain only colour information, with each pixel assigned a colour *class label* corresponding with higher level notions of colour (such as "red" or "yellow"). This reduction has two primary advantages: the reduction of computational complexity, by processing over a lower dimensional search space; and the removal of spatial class dependencies, allowing for the pixel-label mapping

to be calculated offline and stored in a static data structure for constant time access.

This work focuses on this reduced task of *colour segmentation*, within an environment where salient features exhibit some significant degree of colour-coding. RoboCup robot soccer [7] is chosen as such as scenario for experimentation, where field lines, goals and the ball are each assigned unique colours, and where maintaining real-time processing performance is critical for robot responsiveness. A system for generating and storing a pixel-label mapping without human supervision is developed for the Robotis DARwIn-OP humanoid robot, with the mapping quality evaluated in terms of feature classification performance. Finally, the developed method is quantitatively compared to mappings generated manually by a human expert, both in terms of the aforementioned performance and required time for mapping generation.

2 Colour Vision Methodology

2.1 Colour Spaces

In image processing and computer vision, the colour of a pixel is most commonly represented by a triplet of values, with each value quantifying the contribution of some channel (representative of some chromatic or brightness property) to that colour. The concrete definition of each channel varies between *colour spaces*, with the most well-known example being the *RGB* space (700.00 nm red, 546.1 nm green and 435.8 nm blue components [10]). Due to the intuitive physical definition of three chromatic channels, most other colour spaces (including *HSV*, YC_bC_r and *CIE L*a*b*) are defined in terms of its transformation from the *RGB* space.

Past research has demonstrated the YC_bC_r colour space as optimal (among the examples listed above) for unsupervised colour segmentation, in terms of both *internal* and *external* validation techniques [2]. Similarly to *HSV*, YC_bC_r separates chrominance information into two channels C_b (blue chroma) and C_r (red chroma), and intensity into a third channel Y (luma) [4, 10]. The YC_bC_r colour space can be obtained applying the following linear transformation to the RGB space

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix},$$

where R', G' and B' are 8-bit gamma-compressed colour components [10]. Although much of the component correlation found in *RGB* is removed, it still exists in part due to the linear nature of the transformation [4].

As YC_bC_r is both the native colour space for many cameras (including the Logitech C905 fitted to the utilised DARwIn-OP robot platform) and experimentally verified as optimal for colour segmentation tasks [2], all pixel values throughout this work are processed and represented in YC_bC_r format.

2.2 Colour Look-Up Tables

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 $c_j \in C$ [3]. 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

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

Concretely, in a colour space C, each pixel in an image is represented by a triplet with each value representing the contribution of each component to the overall colour of that pixel. Projecting the pixel values into the colour space constructed by the orthogonal component axes results in a projected colour space distribution of the original image. Where computational resources are limited, the colour segmentation process is performed offline, 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; as such, the focus of this work reduces to the task of generating effective LUTs without human supervision.

3 Mean Shift and Mode Finding Algorithms

As described in Sec. 1, image segmentation algorithms typically require the assignment of a feature vector to each pixel in an image. Where this feature vector contains only colour (i.e. not spatial) information, the segmentation task is reduced to that of *colour segmentation*. Although many algorithms exist for the task of colour segmentation (and image segmentation is general), this work considers only *mean shift* and *mode finding* techniques. Such techniques involve the assumption that each pixel's feature vector is sampled from some unknown underlying probability distribution, and attempts to locate *clusters* (modes) within this distribution [10].

Specifically, three algorithms are considered: k-means clustering [6], which parameterises the underlying distribution as a superposition of hyperspherical distributions; *expectation maximisation* [1], which generalises k-means by assuming a mixture of Gaussians; and *mean shift* [5], which generates a non-parametric model of the entire distribution by convolving all feature vectors by some *kernel function*.

3.1 *k*-Means Clustering

Given a set of m data points (corresponding with feature vectors in the scenario of colour segmentation) $P = \{x^{(1)}, \ldots, x^{(N)}\}$ ($x^{(i)} \in \mathbb{R}^m$), 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 [6, 10]. This is accomplished by 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:

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

3.2 Expectation Maximisation

As opposed to k-means, which associated each data point to a respective label via the *nearest neighbours* method (i.e. minimum Euclidean distance), *expectation maximisation* (EM) utilises a *Mahalanobis distance* [8, 10]

$$d(x^{(i)}, \mu_k; \boldsymbol{\Sigma}_k) = (x^{(i)} - \mu_k)^T \boldsymbol{\Sigma}_k^{-1} (x^{(i)} - \mu_k),$$

where $x^{(i)}$ and μ_k are data points and cluster centroids respectively ($x^{(i)}, \mu_k \in \mathbb{R}^n$), and $\boldsymbol{\Sigma}_k$ are their corresponding covariance estimates. As opposed to the hard assignment result of k-means, EM then allows for each data point to be softly assigned to several clusters. This is performed by iteratively re-estimating the parameters for a mixture of Gaussians density function [1, 10]

$$p(x|\pi_k, \mu_k, \boldsymbol{\Sigma}_k) = \sum_k \pi_k \, \mathcal{N}(x|\mu_k, \boldsymbol{\Sigma}_k),$$

where π_k is the mixing coefficient, μ_k and $\boldsymbol{\Sigma}_k$ are the Gaussian mean and covariance respectively, and $\mathcal{N}(x|\mu_k, \boldsymbol{\Sigma}_k)$ is the multivariate Gaussian distribution [1, 10]

$$\mathcal{N}(x|\mu_k, \boldsymbol{\Sigma}_k) = \frac{1}{|\boldsymbol{\Sigma}_k|} e^{-d(x, \mu_k; \boldsymbol{\Sigma}_k)}.$$

Concretely, this is accomplished by repeating the following two-step algorithm until convergence:

Step 1: The *expectation* step estimates the likelihood of a data point $x^{(i)}$ having been generated from the kth Gaussian cluster

$$z_{ik} = \frac{1}{Z_i} \pi_k \, \mathcal{N}(x|\mu_k, \boldsymbol{\Sigma}_k), \quad \sum_k z_{ik} = 1.$$

Step 2: The *maximisation* step updates the parameter values

$$\begin{aligned}\mu_k &= \frac{1}{N_k}\sum_i z_{ik}x^{(i)},\\ \boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_i z_{ik}(x^{(i)}-\mu_k)(x^{(i)}-\mu_k)^T\\ \pi_k &= \frac{N_k}{N},\end{aligned}$$

where N_k is an approximation of the number of data points belonging to each cluster

$$N_k = \sum_i z_{ik}.$$

It has been demonstrated that by setting the EM covariance matrices to equal $\epsilon\mathbf{I}$ (where ϵ is a shared covariance value and $\mathbf{I}$ is the identity matrix) and considering the limit $\epsilon \to 0$, π_k becomes equivalent to the fraction of data points assigned to cluster k, and therefore the algorithm is reduced exactly to k-means [1]. Where hard cluster assignment is required in the generalised case (including the traditional colour segmentation scenario), each data point $x^{(i)}$ may be mapped simply to the cluster k that maximises z_{ik}.

3.3 Mean Shift

As opposed to k-means and EM, which rely on explicitly *parametric* mixture models of the underlying distribution, the mean shift algorithm estimates the density function $f(x)$ by convolving the sparse set of data points (corresponding with pixel feature vectors) by some kernel function $k(r)$ [5, 10]

$$\begin{aligned}f(x) &= \sum_i K(x - x^{(i)})\\ &= \sum_i k\left(\frac{\|x - x^{(i)}\|^2}{h^2}\right),\end{aligned}$$

where h is the kernel width and $x^{(i)}$ are the data points ($x^{(i)} \in \mathbb{R}^n$). Unfortunately for a very large number of data points (potentially millions of pixels in the case of an entire image), calculating the density function over the entire search space can become too computationally expensive. Instead, a method known as *multiple restart gradient descent* [10] can be utilised, which iteratively estimates the gradient vector at some point y_t and takes an "uphill" step in that direction. The gradient of the probability density function with *derivative kernel* $G(x)$ is

estimated as

$$\begin{aligned}\nabla f(x) &= \sum_i (x^{(i)} - x) G(x - x^{(i)}) \\ &= \sum_i (x^{(i)} - x) g\left(\frac{\|x - x^{(i)}\|^2}{h^2}\right) \\ &= \left(\sum_i G(x - x^{(i)})\right) m(x),\end{aligned}$$

where $g(r) = -k'(r)$ is the first derivative of $k(r)$, and m(x) (the *mean shift*) is calculated as

$$m(x) = \frac{\sum_i x^{(i)} G(x - x^{(i)})}{\sum_i G(x - x^{(i)})} - x.$$

Finally, the procedure updates the point y_t (at iteration t) by its locally weighted mean ($y_{t+1} = y_t + m(y_t)$) [10]. Although a non-parametric method of clustering by definition, the estimated probability density function $f(x)$ is implicitly parameterised by the selection of kernel function $k(r)$. Two such kernels were considered for colour segmentation: the *Gaussian kernel function*

$$k_N(r) = e^{-\frac{1}{2}r},$$

and *flat kernel function*

$$k_F(r) = \begin{cases} 1 & \text{if } |r| \leq 1 \\ 0 & \text{otherwise} \end{cases}.$$

The latter was chosen due to the simplified implementation and improved computational efficiency. As long as the kernel $k(r)$ is chosen to be monotonically decreasing, the mean shift algorithm is proven to converge [5].

4 System Implementation

4.1 Image Stream Generation

A MATLAB utility was developed to allow for the creation of DARwIn-OP compatible *image streams* from a set of selected images. Specifically, this work utilises a library of 100 images captured by the DARwIn-OP for LUT generation, with each salient RoboCup feature (goals, lines, ball, etc.) equally represented. This library is partitioned into equal sized training and evaluation sets, with each partition maintaining this equal representation [1].

Although this library of images was captured manually, a known initial placement of the DARwIn-OP allows for simple development of a behaviour to automatically capture equivalent representative images.

[1] Available at: `http://davidbudden.com/experimental_data/testImages0912.zip`

4.2 Look-Up Table Generation

Given an input training library of 50 images, the process of LUT generation involves the following four steps:

1. **Selection of training images:** It was hypothesised that the classification performance of an autonomously generated LUT would be some function of the number of training images considered during its generation. Too few images would provide insufficient information to allow robust classification, whereas too many would result in the over-classification of colours, introducing substantial background noise. As such, for each proposed method of LUT generation, LUTs were generated for different sized subsets of the training set of 50 images. Concretely, for each method, the number of training images ranged from 5 to 50, increasing in multiples of 5, with each image selected at random from the initial training set while ensuring the preservation of feature distribution.
2. **Cluster generation:** Each of the three clustering algorithms described in Sec. 3 (k-means, expectation maximisation and mean shift) were applied separately to each selected training image within the 3-dimensional YC_bC_r colour space. The pixel-cluster mapping for each pixel is stored.
3. **Cluster merging:** Although cluster validation assigns a label to each pixel within the original image, there are two fundamental problems. Firstly, the number of clusters is in no way representative of the number of salient colours actually present within the image, but rather determined by some algorithm parameter. Secondly, there is no direct relation between cluster labels (an arbitrary enumeration) and actual high-level colours (red, blue, etc.) These issues are solved by *cluster merging*, which requires a human user to define a number of optimal YC_bC_r (or RGB equivalent) triplets for potential colours of interest [2]. Each pixel is assigned its final colour class label by determining which optimal triplet is the nearest neighbour of that pixel's assigned cluster centroid.
4. **LUT generation:** After each clustering is performed, the mapping between every YC_bC_r triplet and respective colour class label is stored. However, there are often clashes between mappings across a set of images; for example, a triplet may map to the yellow class label in one image, but the orange class label in another. To deal with this, a voting system is implemented, where the class label most frequently associated with a given triplet defines the value eventually stored in the LUT. The LUT is simply a data file storing a flattened $256 \times 256 \times 256$ matrix of colour class labels (0 for unclassified), indexed by YC_bC_r value and generated in a raw format (no headers or formatting bytes) understood by the remainder of the DARwIn-OP vision system.

[2] In the RoboCup environment, all salient features are encoded with colours corresponding with the corners of the RGB cube.

4.3 Look-Up Table Generalisation

Previous research has indicated that the quality of a manually generated LUT may be improved via the process of *generalisation*; filling gaps and removing outliers by *support vector machine* (SVM) post-processing. For each colour class label, a single one-class SVM was created. The purpose of each SVM is to determine the likelihood (between -1 and 1) of each of the 256^3 data points (i.e. YC_bC_r pixel coordinates, also scaled between -1 and 1 for each axis) belonging to that respective class. Concretely, a MATLAB implementation of the LIBSVM one-class SVM implementation was used [11].

Despite the fact that previous research has indicated the implementation of such a generalisation process improves LUT performance [9], a course-grain grid search of SVM parameters yielded consistent reduction in performance. This discrepancy may be influenced by a number of factors: the improved quality of input LUTs, the smoother nature of input class regions (i.e. resulting from clustering rather than some manual process) and the selection of a different colour space (YC_bC_r rather than HSV).

4.4 Performance Evaluation

As several thousand LUTs were generated for this work, it is infeasible to qualitatively assess the performance of each by "how good" the corresponding classified images appear (the standard approach the evaluating a manually generated LUT within the NUview utility). Furthermore, although previous research has demonstrated a correlation between *internal* and *external cluster validation techniques* in the context of colour classification [2], a more explicit method of evaluating LUT performance is necessary.

As described in Sec. 4.1, a library of 50 images are used for the purpose of LUT performance evaluation. Concretely, a feature vector of binary features is manually generated for each image, which each bit corresponding with the presence of some salient RoboCup feature (goals, lines, ball, etc.) within that frame. By generating a similar feature vector by processing each LUT-classified image with the DARwIn-OP robot vision system, classification performance can be calculated directly. Furthermore, by maintaining the same vision system version and same set of 50 images between experiments, the colour segmentation represented by the LUT becomes the only experimental variable. In this case, it is claimed that the following metrics are representative of the performance of a LUT:

- **Sensitivity:** Also known as recall. Measures the probability of classifying some feature, given that the feature is present. Concretely,

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}.$$

- **Specificity:** Also known as precision. Measures the probability of not classifying some feature, given that the feature is not present. Concretely,

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}.$$

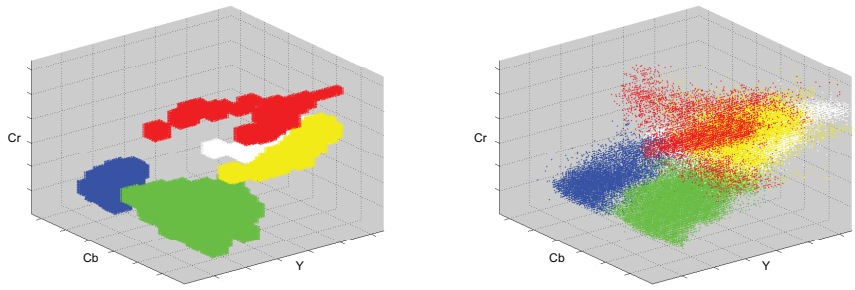

Fig. 1. Two examples of colour look-up tables generated for the training set of 50 images: a manually tuned LUT (left), created within the NUview utility using a point-and-click approach over the period of 1 hour; and an autonomously generated LUT (right), utilising k-means clustering over a subset of 30 training images and generated in less than 10 seconds.

5 Results

As described in Section 4, look-up tables (LUTs) were generated for differing sized subsets of the initial training set of 50 images, based on the presumption that an optimal quantity of training data should be evident. As such, for each clustering technique, 10 LUTs were generated to assess a range of training set sizes from 5 to 50 images. Furthermore, to ensure the statistical significance of the experimental results, each of these 10 LUTs were generated 120 times, with the presented results representing the mean performance. This step is necessary due to two stochastic elements of the generation process: the random selection of image subsets from the original training set; and the random initial placement of cluster centroids, resulting in final clusters of varying density and separation.

Fig. 2 presents the mean sensitivity and specificity values for LUTs produced using the k-means (blue), expectation maximisation (red) and mean shift (magenta) algorithms. These plots represent data collected from 3600 LUTs, each tested over 50 images, resulting in the binary classification of in excess of 1.2 million features.

Some interesting observations follow from these results. Firstly, the specificity curve for all methods follow the expected trend, decreasing as the number of training images increases as the result of over-classification of colour. Secondly, the sensitivity curve for mean shift and expectation maximisation confirmed the initial hypothesis: poor classification for too few images due to a lack of training data. Similarly, convergence to relatively poor classification was evident for too many images, due to the excess classification of background noise preventing effective feature extraction. For some intermediate values (10 for mean shift, and 20 for EM), optimum sensitivity is achieved.

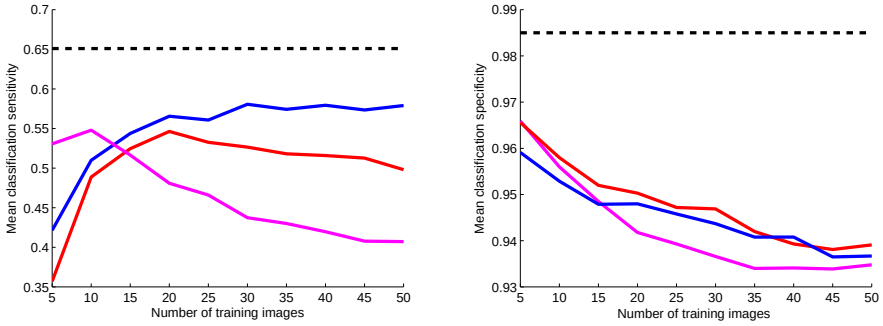

Fig. 2. Comparison of colour look-up table feature classification sensitivity (left) and specificity (right) as a function of the number of training images, for k-means (blue), expectation maximisation (red) and mean shift, utilising a flat kernel (magenta). The performance of a manually tuned LUT is indicated by the dashed lines. It is observed that k-means clustering yields the best classification sensitivity for most training set sizes, with little difference evident between the specificities of each method.

Fig. 2 demonstrates that k-means does not experience the same overshoot in sensitivity as EM and mean shift, but rather asymptotically approaches its optimum sensitivity value. Furthermore, k-means overall outperforms both EM and mean shift algorithms. As k-means is a specific case of EM that assumes underlying hyperspherical distributions of data, it can be inferred that human-perceived salient colours may form approximately spherical distributions within the YC_bC_r colour space, therefore allowing more optimal partitions to be formed within the algorithm's internal iteration limit. This observation is consistent with earlier research suggesting that clusters of maximum density and separation occur within the YC_bC_r space, and correspond with qualitatively accurate segmentation of an image into regions of salient colour [2].

5.1 Optimal Method Selection and Evaluation

From the previous experimentation, it has been demonstrated that the best performing colour look-up tables are produced by k-means clustering. Given this information, the remaining task is to determine the optimal set of internal clustering parameters. Specifically, the MATLAB implementation of k-means clustering is parameterised by two values of interest: k, the initial number of parameters (later reduced to the number of colour class labels by the LUT generation implementation); and the *iteration count*. For an iteration count greater than 1 (the default value used previously in experimentation), each clustering is repeated an equal number of times, with only the clusters yielding the best spatial properties (lowest average distance from centroids to corresponding points) being retained. Although it is intuitive that clusters of improved spatial prop-

erties may yield LUTs which exhibit improved classification performance, this correlation has not been experimentally verified.

The experimental range for k was set to be from 5 to 14 (with the previous default as 10), and that for the iteration count set from 1 to 10. For each of the 100 resulting pairs of parameter values, 120 LUTs were generated, resulting final autonomous creation of 15600 LUTs, and respective evaluation of in excess of 5 million binary features. The average sensitivity and specificity for each parameter pair are presented in Fig. 3.

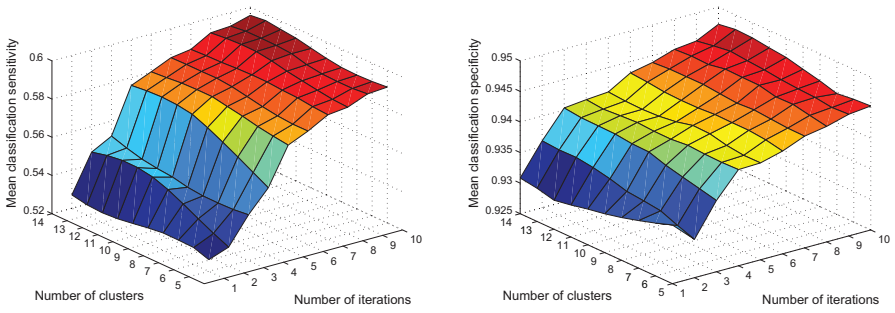

Fig. 3. Mean classification performance for colour look-up tables generated by k-means clustering, parameterised by both the number of clusters and clustering iteration count, presented in terms of sensitivity (left) and specificity (right). See Sec. 4.4 for definitions.

Fig. 3 demonstrates that increasing both the number of clusters and number of iterations results in an overall diminishing performance increase, in terms of both sensitivity and specificity. The optimal pair of parameters was experimentally determined to be: number of clusters $k = 11$, and number of iterations $= 10$. The mean performance of the 120 LUTs generated for this parameter pair is presented in the following table, along with the performances of the manually generated benchmark LUT, and the overall best performing LUT from the set of 120.

	Benchmark	**Mean**	**Best**	**Performance**
Sensitivity	0.6596	0.5964	0.7163	1.08
Specificity	0.9768	0.9465	0.9747	1.00

It can be seen that, although the mean performance of the 120 generated LUTs is overall slightly worse than the manually generated benchmark, the best performing LUTs yield an 8% increase in classification sensitivity, for zero tradeoff in specificity.

6 Conclusion

It has been demonstrated that a system can be developed to solve a seemingly subjective problem (the mapping of millions of pixel values to a small set of colour class labels) without human supervision, despite the inherent similarity to subconscious human abstraction. Moreover, it has been demonstrated that, in addition to solving the problem in orders of magnitude less time than is required by a human expert (in the order of seconds, rather than hours), the autonomous colour look-up table (LUT) generation system is able to produce LUTs that yield up to an 8% improvement in classication performance over those manually tuned within the NUview utility. Although demonstrated solely within a RoboCup robot soccer environment, this development has strong potential in any image processing or machine vision system requiring real-time recognition of coloured features.

References

1. Bishop, C.: Pattern recognition and machine learning, vol. 4. Springer (2006)
2. Budden, D., Fenn, S., Mendes, A., Chalup, S.: Evaluation of colour models for computer vision using cluster validation techniques. In: RoboCup 2012: Robot Soccer World Cup XVI. Springer (In Press)
3. Budden, D., Fenn, S., Walker, J., Mendes, A.: A novel approach to ball detection for humanoid robot soccer. In: Advances in Artificial Intelligence (LNAI 7691). Springer (2012)
4. Cheng, H., Jiang, X., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern recognition 34(12), 2259–2281 (2001)
5. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24(5), 603–619 (2002)
6. 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)
7. Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E.: Robocup: The robot world cup initiative. In: Proceedings of the first international conference on Autonomous agents. pp. 340–347. ACM (1997)
8. Mahalanobis, P.C.: On the generalized distance in statistics. In: Proceedings of the National Institute of Sciences of India. vol. 2, pp. 49–55. New Delhi (1936)
9. Quinlan, M.J., Chalup, S.K., Middleton, R.H.: Application of svms for colour classification and collision detection with aibo robots. Advances in Neural Information Processing Systems 16 (2003)
10. Szeliski, R.: Computer vision: algorithms and applications. Springer (2010)
11. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann (2005)