A Fast Method for Adapting Lookup Tables Applied to Changes in Lighting Colour

Trent Houliston(✉), Mitchell Metcalfe, and Stephan K. Chalup

School of Electrical Engineering and Computer Science,
The University of Newcastle, Callaghan, NSW 2308, Australia
{trent.houliston,mitchell.metcalfe,stephan.chalup}@newcastle.edu.au

Abstract. This paper proposes a simple and fast method for adapting colour lookup tables to lighting changes in real-time. The method adjusts the classified colour space regions keeping both their surface area and volume constant. Two variations of the method were compared and tested in a RoboCup soccer setting. Detection success rate was measured as a function of the speed and magnitude of hue change to the lighting environment. Compared to a static lookup table, these experimental results show improved robustness against lighting changes for detection of coloured objects.

Keywords: Color lookup table · Illumination invariance · Color space · Computer vision · Robotics

1 Introduction

Mobile robots competing in the RoboCup humanoid and standard platform leagues use vision as their primary sensor. Colour perception is useful as the typical RoboCup soccer field environment has several specified colours. These included an orange ball and blue and yellow coloured goal posts. Consequently, most teams at RoboCup investigate methods to implement stable and efficient colour vision, including colour segmentation in the presence of illumination changes on their robots [4,6–8]. This is in contrast to general mobile robots that avoid colour perception issues by using laser range finders and sonar not unlike those used for the DARPA challenges. Sridharan and Stone [14] have provided a review of some of the most significant work on illumination invariance on mobile robots up until 2008. The challenge for robots that rely on video sensors is that they have to be able to deal with shadows, reflections, natural light changes and other artefacts caused by robot motion in RoboCup and real-world domains. For example when a robot approaches a ball it often ends up in a shadow cast by another robot. Another challenge is that many methods require significant human supervision [11] or substantial computational resources and time [2,12,13]. Only recently more appropriate solutions accessible to small robots such as the DARwIn-OP or Nao have been developed [9].

L. Almeida et al. (Eds.): RoboCup 2015, LNAI 9513, pp. 190–201, 2015.
DOI: 10.1007/978-3-319-29339-4_16

This paper proposes a fast hands-on implementation of a basic procedure to adapt lookup tables to varying lighting conditions. This implementation improves an initial lookup table and ensures its robustness to illumination changes in a RoboCup environment with a limited number of uniformly coloured objects.

2 Adaptive Lookup Tables

A Lookup Table (LUT) for colour classification is a map from the colour space of an input image to a set of colour classes. The LUT can be represented as a cube of discrete voxels, indexed by the three components of the pixel colours in the input image. For example, y, u, and v, for the YUV colour space.

Robots at RoboCup often have difficulty classifying colour when the field lighting is not uniforn or changes during a game. This paper outlines a method of dynamically adapting a LUT to the current lighting conditions in order to resolve this problem. This is achieved through continually updating a LUT using feedback from object detectors, like a ball detector. Each object detector recognises a single type of object that is coloured with a single class of colour, like an orange ball, yellow or white goalposts or white field lines.

An initial seed LUT is required to begin the feedback process. This LUT must be able to classify objects sufficiently for detection under the startup illumination environment. The quality of the initial LUT will be improved automatically by the proposed algorithm, due to the feedback process, allowing an initial LUT of lower quality to be used. This initial seed LUT can be provided by hand classification, or more expensive algorithms [1,5] that can detect and classify the salient regions of the colour space.

Upon each object detection, the pixel colours within the image region of the detected object are passed to the algorithm. For each of these pixels, the corresponding voxel in the current LUT is found. If any of the voxel's neighbours are of the same colour class as the detected object and the voxel is unclassified, the voxel is assigned the object's colour class, updating the LUT. Two voxels are considered to be neighbours if they share a face.

Once the LUT has been updated for each pixel in the detected object's image region, the *volume*, and *surface area* of the colour class are compared to preset maximum values. The volume of a colour class is defined as the number of voxels belonging to the class. The surface area of a colour class is defined as the number of *removable* voxels belonging to the class (see Sect. 2.1). If the volume or surface area of the class exceed their preset values, a *shedding* process is performed that removes a layer of voxels from the exterior of the colour class in the LUT. This works to remove old voxels from the LUT that no longer accurately represent their colour class due to illumination changes.

The shedding step of the algorithm can be performed through two different methods: layer based shedding and voting based shedding. These shedding methods are compared in Sect. 2.1. The process described earlier is the basic update process that is used with the layer based shedding method. Pseudocode for this update process is presented in Algorithm 1.

A different update process is required to support the voting shedding method. In the voting update process, an integral vote count is maintained for individual voxel in each colour class. Each voxel belonging to a colour class in the initial LUT has its vote count set to a preset maximum value. When an object of a given colour class is detected, the vote count of each voxel in that class is first decremented by a preset value. Then, as in the layer update method, colours from the detected object's image region are used to classify voxels in the LUT, with their votes also updated. When a new voxel is added to the colour class, its vote is set to zero. When a voxel of the current colour class is encountered, its vote count is incremented by a preset value (as presented in the pseudocode in Algorithm 3). Voxel vote counts are capped at a preset maximum value. If the volume or surface area of the class exceed their preset values, a shedding method is run that considers vote counts during shedding. Pseudocode for the voting based update process is described in Algorithm 1.

Algorithm 1. UPDATELUTLAYER(S, c)

```
Input: The set $\mathcal{S} \in \mathbb{R}^3$ of pixel colours within the image region of a detected
       object
Input: $c$ the colour class of the detected object
1 foreach $x \in \mathcal{S}$ do
2     $v \leftarrow$ GETLUTVOXEL($x$)
3     if CLASS($v$) $= \varnothing \wedge c \in \{$CLASS($w$) : $w \in$ NEIGHBOURS($v$)$\}$ then
4         CLASS($v$) $\leftarrow c$
5     end
6 end
7 if
  VOLUME($c$) $>$ MAXVOLUME($c$) or SURFACEAREA($c$) $>$ MAXSURFACEAREA($c$)
  then
8     PERFORMSHEDDING($c$)
9 end
```

In addition to these procedures and functions in Table 1, there are configuration parameters that influence the actions of the algorithm at runtime. Appropriate values for MAXSURFACEAREA(c), MAXVOLUME(c), MAXVOTES(c), VOTEGROWTHRATE(c), VOTESHRINKRATE(c), must be chosen by experimentation before the algorithm is run.

2.1 Shedding

Shedding is the process of reducing the volume of a colour class by removing its outermost voxels. To describe the shedding process, three different classifications of LUT voxel are defined:

Internal. A voxel that has six neighbours, all of the same class as in Fig. 1a.

Table 1. Procedures and functions that appear in the algorithms

$\textsc{Class}(v)$	Returns the current colour class assigned to the voxel v.
$\textsc{DecrementVotes}(c)$	Decrements the vote count of all voxels with the colour class c by $\textsc{VoteShrinkRate}(c)$.
$\textsc{GetLUTVoxel}(x)$	Returns the voxel in the LUT indexed by the pixel x.
$\textsc{Neighbours}(v)$	Returns the six neighbours of the given voxel, adjacent to each of its six cube faces.
$\textsc{SurfaceArea}(c)$	Returns the surface area of colour class c.
$\textsc{Volume}(c)$	Returns the number of voxels with the given colour class c in the LUT.
$\textsc{RemoveableSurface}(c)$	The voxels of a colour class that are removable (as defined by the method in Sect. 2.1)
$\textsc{Vote}(v)$	Returns the vote count of the given voxel.

Algorithm 2. $\textsc{UpdateLUTVoting}(S, c)$

```
Input: The set $\mathcal{S} \in \mathbb{R}^3$ of pixel colours within the image region of a detected
       object
Input: $c$ the colour class of the detected object
Input: The Lookup Table $\mathcal{L}$
1  $\textsc{DecrementVotes}(c)$
2  foreach $x \in S$ do
3      $v \leftarrow \textsc{GetLUTVoxel}(x)$
4      if $\textsc{Class}(v) = c$ then
5          $\textsc{IncrementVote}(c, v)$
6      end
7      else if $\textsc{Class}(v) = \varnothing \wedge \textsc{Class}(c) \in \{\textsc{Class}(w) : w \in \textsc{Neighbours}(v)\}$ then
8          $\textsc{Class}(v) \leftarrow c$
9          $\textsc{Vote}(v) \leftarrow 0$
10     end
11 end
12 if
   $\textsc{Volume}(c) > \textsc{MaxVolume}(c)$ or $\textsc{SurfaceArea}(c) > \textsc{MaxSurfaceArea}(c)$
   then
13     $\textsc{PerformShedding}(c)$
14 end
```

Algorithm 3. $\textsc{IncrementVote}(c, v)$

```
Input: The colour class $c$
Input: The voxel $v$
1  $n \leftarrow \textsc{Vote}(v) + \textsc{VoteGrowthRate}(c)$
2  $\textsc{Vote}(v) \leftarrow \min(n, \textsc{MaxVotes}(c))$
```

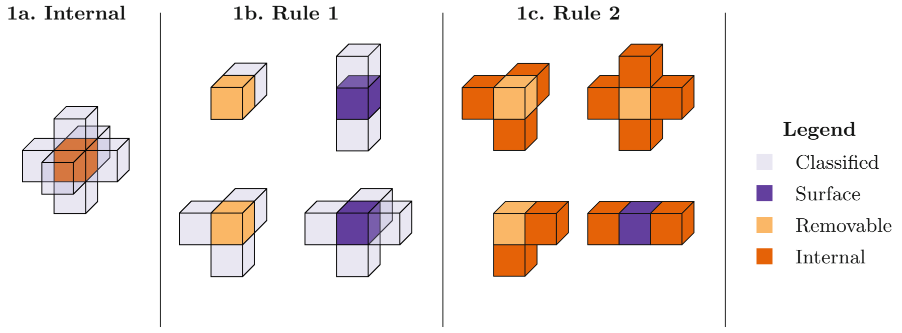

Fig. 1. Examples of the different classifications of voxels due to the rules outlined in Sect. 2.1

Surface. Any voxel that is not internal.

Removable. A surface voxel that satisfies at least one of the following conditions:

(a) For every neighbouring voxel of the same colour class, the opposite neighbouring voxel must be vacant (example Fig. 1b).

(b) Has at least two non-opposite internal voxels as neighbours. At least two neighbouring voxels must be internal voxels and those internal voxels must share a neighbour that is not the central voxel (example Fig. 1c).

These conditions are heuristics designed to preserve the topology of the colour class volume by avoiding the creation of new holes or splitting connected components of the colour class in the LUT. However, there are two notable cases where the topology could change. Firstly, when a thin section of volume develops a 'staircase' pattern. These staircases can be removed, resulting in two separate volumes. Secondly, when the volume extends into a torus shape, creating a hole in the resulting volume. The algorithm makes no attempt to address these problems, as avoiding them significantly increases the number of voxels that must be checked. Also, if these cases do arise, the system is able to recover as one half of the shape can disappear or the two halves will rejoin.

The two shedding algorithms that have been developed are:

Layer Shedding. Removes the class labels from all removable voxels of the given colour class in the LUT.

Voting Shedding. Zeroes all votes for and removes class labels of all voxels of the given colour class that have a vote count below the mean vote count among the surface voxels of that class.

Pseudocode for the layer shedding and voting shedding algorithms is included as Algorithms 4 and 5.

Algorithm 4. PerformLayerShedding(c)

```
   Input: The colour class $c$ to be shed
1  $S_r \leftarrow$ RemoveableSurface($c$)
2  foreach $v \in S_r$ do
3      Class($v$) $\leftarrow \varnothing$
4  end
```

Algorithm 5. PerformSheddingVoting(c)

```
   Input: The colour class $c$ to be shed
1  $S_r \leftarrow$ RemoveableSurface($c$)
2  $m \leftarrow$ Average($\{$Vote($w$) : $w \in S_r\}$)
3  foreach $v \in S_r$ do
4      if Vote($v$) $< m$ then
5          Class($v$) $\leftarrow \varnothing$
6          Vote($v$) $\leftarrow 0$
7      end
8  end
```

2.2 Implementation

During the implementation of the algorithm a number of optimisations were made. These optimisations were not included in the description of the algorithm as they obfuscate its intention.

The surface voxels for each colour class are stored in a hashset. The set is updated any time that a voxel is inserted or removed as this could change a voxel's classification to or from removable.

The voting algorithm stores a value to represent zero for each colour class. This value is incremented rather than decrementing the vote counts of each voxel when DecrementVotes is called.

The worst case complexity of the algorithm is $\mathcal{O}(nr)$ where n is the number of pixels in the detected object and r is the number of removable voxels for the colour class. Shedding should only happen with regularity when the volume is moving. When the volume is stationary and shedding does not occur, the complexity of this algorithm is $\mathcal{O}(n)$.

In addition to these optimisations, the pixels sent to this algorithm from the object detectors can be subsampled in order to reduce the number of updates to the LUT at the cost of reduced adaptability.

3 Experiments

An experiment was conducted to evaluate the ability of the developed methods to adapt to changes in illumination colour. The robot was placed on a 2013 style kid size humanoid league soccer field [10] with the goals and ball entirely within its camera frame. A ceiling mounted digital projector was pointed at the goals and made to project continuously changing coloured light onto the robot's field of view. This was the only source of light in the room. The experimental setup is illustrated in Fig. 2.

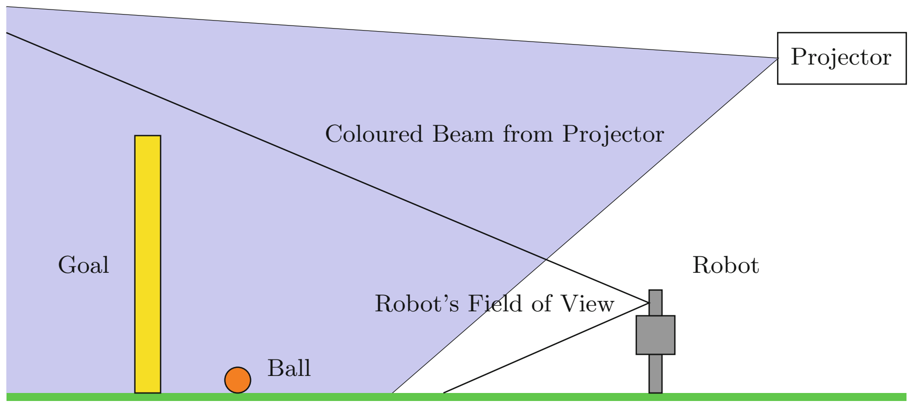

Fig. 2. The experimental setup. The robot views the ball and goals, which are illuminated by a ceiling mounted projector.

3.1 Colour Sequences

Twenty-one separate video sequences were captured for the experiment. Each video sequence was two minutes long with varying illumination colours $\mathcal{C}$ produced by the projector over time. Each of the videos set a different value of a parameter α that controlled the amount of colour variation produced by the projector. This is referred to as 'saturation' in the results. However, it is not equivalent to the saturation parameter of the HSV colour space. Each video was resampled to generate twelve videos of different durations ranging from 5–60 s long at 5 s intervals. This simulates the slower and faster colour changes that might occur during normal operation of the robot. As a result, a total of 252 videos were tested. The formula for the colour displayed by the projector at time t in a trial of duration d is given in Eq. (1), where the function HSV creates an HSV colour and the function RGB converts an HSV colour to its RGB representation.

$$\mathcal{C} = \frac{\alpha}{100}\,\mathrm{RGB}(\mathrm{HSV}(\frac{360t}{d}, 1, 1)) + (1 - \frac{\alpha}{100})\,\mathrm{RGB}(\mathrm{HSV}(0, 0, 1)) \qquad (1)$$

Each video was recorded at a resolution of 320×240, and a frame rate of 30 Hz. Figure 3 presents a sequence of selected frames from one of the videos.

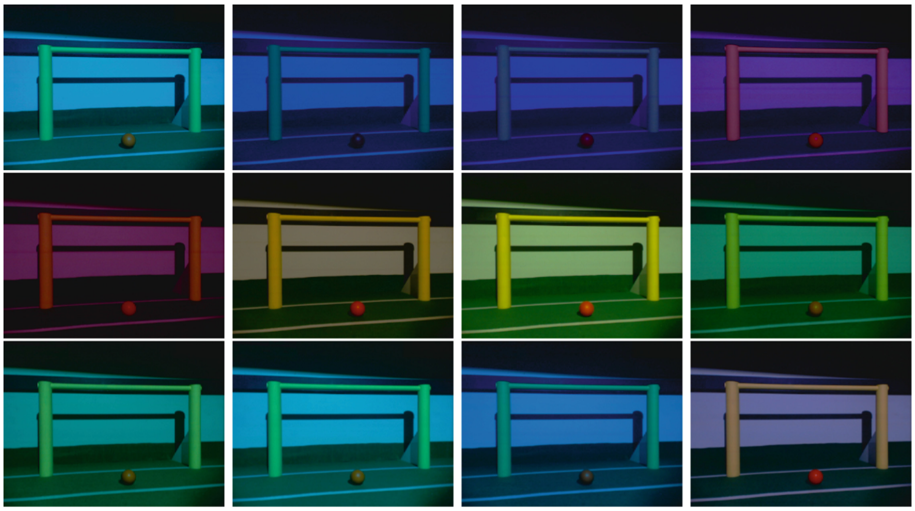

Fig. 3. Selected frames from one of the colour sequences from the experiment. The ball and goals are both visible and are lit only by coloured light from the projector. The colour throughout the sequence follows a full hue cycle while maintaining constant saturation parameter α of 60 % (Color figure online).

3.2 Trials

Three methods were run on each of the 252 videos: a regular static LUT, an adaptive LUT using the layer shedding scheme and an adaptive LUT using the voting shedding scheme. The colour space used for the LUT was YUV, with a resolution of 6 bits for y, 7 bits for u and 7 bits for v.

Two object detectors were used in the experiments: a ball detector that detects an orange ball and a goal post detector that detects yellow goal posts. Both of these detectors were based on the RANSAC algorithm [3]. The output of the detectors were used to track the orange and yellow colour classes.

During each trial, the positions of the detected objects in each frame were compared to the known positions of the objects in the frame. Detections that were within a threshold of the true object positions were recorded as successful.

Detection of the ball was considered successful if the x and y coordinate of the ball's centre and the ball's radius were each detected with an error of less than 2 pixels from their true values. The true radius of the ball was 9.3 pixels. A goalpost detection was considered successful if each of its corner points differed from its corresponding true corner point by less than 10 pixels in both dimensions. The left and right goalposts were 130 pixels and 110 pixels tall respectively. Both goalposts were approximately 20 pixels wide.

A detection rate for each sequence was calculated by dividing the total number of successful detections in a trial by the number of frames in the video and by the number of objects in each frame (a constant three for the goals and ball).

Each trial was run in realtime on a late 2013 Macbook Pro 15" 2.6 GHz Intel Core i7.

4 Results

Figure 4 summarises the performance of the static LUT, the adaptive LUT with layer based shedding and the adaptive LUT with voting based shedding during the experiment described in Sect. 3. Each figure presents a contour plot where the x and y axes are the saturation (defined in Sect. 3.1) and trial duration and the z axis is the detection rate as a percentage of possible detections of the trial.

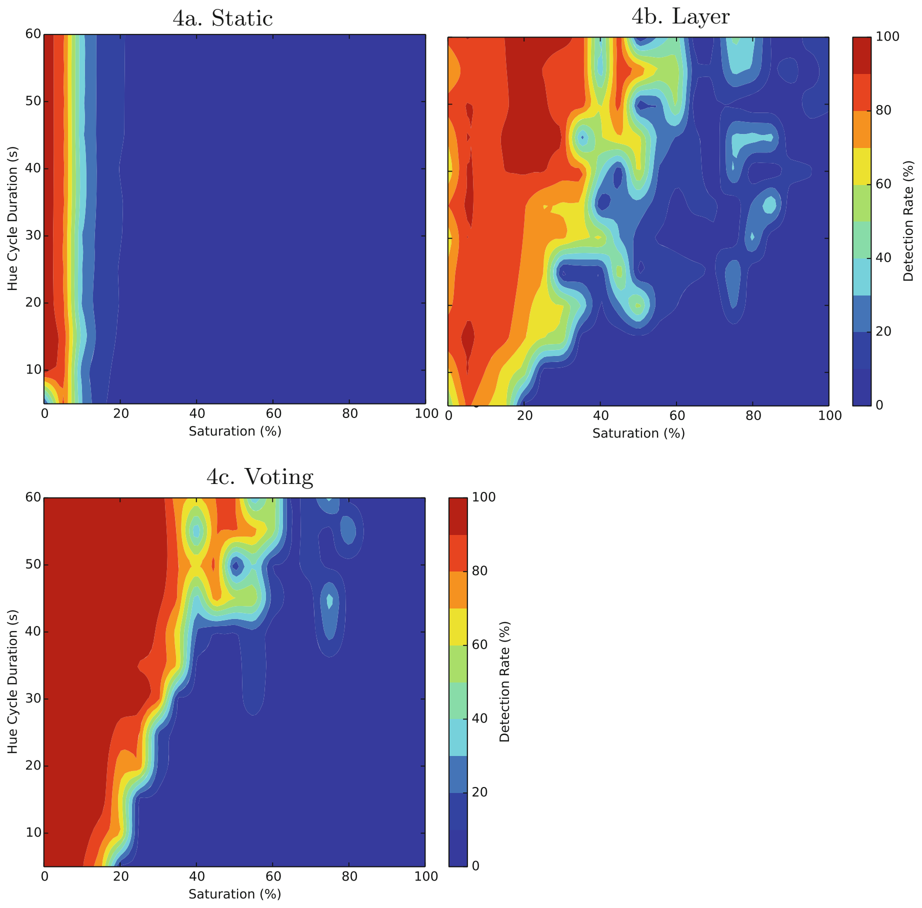

Fig. 4. Contour plot describing the performance of the three different LUT methods. The dark red area on the left is where the methods performed best (Color figure online).

As Fig. 4a shows, the static LUT achieves a detection rate above 90 % for trials with very low colour variation (i.e. saturation below 5 %) that falls to 0 %

before saturation reaches 20 %. The performance of the detector is independent of the duration of the trial.

The results for the adaptive LUT using the layer shedding method are presented in Fig. 4b. A detection rate of over 80 % is achieved for trials with 5 % saturation and duration ranging from 10 s to 50 s and trials with saturation between 10 % and 30 %, and durations above 40 s. Outside of these regions, the detection rate falls to zero in an irregular manner and approaches zero for most trials with a saturation value above 50 %. It falls to zero faster for lower durations.

The adaptive LUT using the voting shedding scheme produced the results in Fig. 4c. It achieved a detection rate above 90 % for all trials with a saturation value lower than 10 %, for all trials with a saturation parameter below 30 % and a duration longer than 30 s. The detection rate falls to 0 % for the shortest trials as the saturation parameter approaches 20 %, and falls to 0 % for most other trials as it increases beyond 40 %.

5 Discussion

The experiment showed that a static LUT performed very poorly when the illumination colour varied. Reasonable detection performance was only achieved for a very narrow range of hue values. This is in contrast with the two adaptive LUTs that performed well across a much wider range of illumination parameters. As a result, the voting shedding scheme displayed a clear advantage over the simpler, layer shedding scheme. It achieved high detection rates over a larger range of parameters, while exhibiting more predictable performance across the range of parameters tested.

In addition to the formal experiment, the performance of the algorithm was tested on a DARwIn-OP in a simulated RoboCup game environment. In this test, the robot's camera was running at 640×480 pixels at 30 Hz. The system was tested detecting goals and balls while the robot was walking. Other robots were also on the field occasionally occluding the ball and goals. During these tests, the detection and computational performance of the system was monitored with and without the adaptive LUT running.

It was discovered that the algorithm could cause the LUT to diverge from useful classification. This would occur whenever the volume in the LUT included black. Black acts as a singularity in the colour space as all hues tend to black with reduced brightness. Given enough examples of heavy shading or occlusion from dark objects, the LUT would include black ($y \rightarrow 0$) and the classification of false positives would increase. This exasperated the problem. Including minimum brightness constraints for pixels used by the algorithm resolved this issue as the colour classes could no longer reach this singularity.

To measure the computational performance of the system, the frame rate and CPU usage were analysed during testing both with and without the adaptive LUT. The frame rate of the system in both cases was 22 Hz. This is due to an IO bottleneck, rather then a processing bottleneck, which resulted in the total CPU usage being below 100 %. The additional computational overhead for the adaptive LUT was measured to be 10 percentile points.

6 Conclusion

This paper has presented a new technique for adapting lookup tables in realtime to achieve robust colour classification under varying illumination. The approach used information from object detections to add missing colour class information to the lookup table. It proposed the use of a shedding procedure to remove old colour class information that no longer accurately represented its colour class from the LUT. Together, these update and shedding procedures are intended to allow the LUT to adapt to changes in illumination. Two different shedding procedures were designed - a layer based method and a voting based method. They were compared based on their ability to adapt to illumination changes in a controlled experiment. The results show that the voting based shedding algorithm is both more accurate and consistent than the layer based method and that both adaptive LUT methods significantly outperformed a simple static LUT.

In contrast to most of the already existing studies that focus on illumination intensity changes, this method evaluated the hue changes of lighting.

Potential directions for future work could include testing the new method on a wider variety of real world scenarios and applying the algorithm to other problems requiring adaptive classification in a continuous space, in particular the classification of texture.

Acknowledgments. The authors would like to acknowledge Brendan Annable for early discussions and implementation ideas as well as Ellie-Mae Simpson, Jake Fountain and Amy Kendrick for their assistance in proofreading this document and their helpful suggestions.

References

1. Budden, D., Mendes, A.: Unsupervised recognition of salient colour for real-time image processing. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 373–384. Springer, Heidelberg (2014)
2. Serhan Daniş, F., Meriçli, T., Levent Akın, H.: Using saliency-based visual attention methods for achieving illumination invariance in robot soccer. In: Chen, X., Stone, P., Sucar, L.E., van der Zant, T. (eds.) RoboCup 2012. LNCS (LNAI), vol. 7500, pp. 273–285. Springer, Heidelberg (2013)
3. Flannery, M., Fenn, S., Budden, D.: RANSAC: identification of higher-order geometric features and applications in humanoid robot soccer. In: Proceedings of the 2014 Australasian Conference on Robotics and Automation (ACRA 2014). ARAA (on-line) (2014)
4. Heinemann, P., Sehnke, F., Streichert, F., Zell, A.: An automatic approach to online color training in RoboCup environments. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4880–4885 (2006)
5. Henderson, N., King, R., Chalup, S.K.: An automated colour calibration system using multivariate gaussian mixtures to segment hsi colour space. In: Kim, J., Mahony, R. (eds.) Proceedings of the 2008 Australasian Conference on Robotics and Automation (ACRA 2008), vol. 6. ARAA (on-line) (2008)

6. Iocchi, L.: Robust color segmentation through adaptive color distribution transformation. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 287–295. Springer, Heidelberg (2007)
7. Luan, X., Qi, W., Song, D., Chen, M., Zhu, T., Wang, L.: Illumination invariant color model for object recognition in robot soccer. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010, Part II. LNCS, vol. 6146, pp. 680–687. Springer, Heidelberg (2010)
8. Mayer, G., Utz, H., Kraetzschmar, G.K.: Playing robot soccer under natural light: a case study. In: Polani, D., Browning, B., Bonarini, A., Yoshida, K. (eds.) RoboCup 2003. LNCS (LNAI), vol. 3020, pp. 238–249. Springer, Heidelberg (2004)
9. Neves, A.J.R., Trifan, A., Cunha, B.: Self-calibration of colormetric parameters in vision systems for autonomous soccer robots. In: Behnke, S., Veloso, M., Visser, A., Xiong, R. (eds.) RoboCup 2013. LNCS (LNAI), vol. 8371, pp. 183–194. Springer, Heidelberg (2014)
10. RoboCup Technical Committee: RoboCup soccer humanoid league rules and setup for the 2013 competition in Eindhoven (2013). http://www.robocuphumanoid.org/wp-content/uploads/HumanoidLeagueRules2013-05-28.pdf
11. Schulz, D., Fox, D.: Bayesian color estimation for adaptive vision-based robot localization. In: Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004), vol. 2, pp. 1884–1889 (2004)
12. Sridharan, M., Stone, P.: Autonomous color learning on a mobile robot. In: Proceedings of the National Conference on Artificial Intelligence, vol. 20, pp. 1318–1323. AAAI Press/MIT Press, Menlo Park, Cambridge, London (1999, 2005)
13. Sridharan, M., Stone, P.: Structure-based color learning on a mobile robot under changing illumination. Auton. Robots **23**(3), 161–182 (2007)
14. Sridharan, M., Stone, P.: Color learning and illumination invariance on mobile robots: a survey. Robot. Auton. Syst. **57**(6), 629–644 (2009)