THE UNIVERSITY OF NEW SOUTH WALES
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Automated Offline Colour Calibration using a Feature Templating System

Calvin Tam
z3376392
ctam@cse.unsw.edu.au

COMP3902 (Special Project B)

Submitted: June 10, 2013

Supervisor: Dr. Bernhard Hengst
Assessor: A. Prof. Maurice Pagnucco

Acknowledgements

I would like to thank my supervisor, Dr. Bernhard Hengst, for his continued mentoring and support.

Thanks go to this year's rUNSWift team consisting of Alex, Beth, Dan, Jack, Richard and Stephen for putting in a lot of effort into this year's competition. Thank you Jayen and Sean for your enthusiasm and providing helpful advice along the way.

Finally, I would like to thank my dear Vivian for her unwavering support and encouragement.

Abstract

The aim of this project is to address the significant time and effort required to perform manual colour calibration by automating the process altogether. A feature templating system is proposed as a method to achieve automation, with two automatic colour calibration techniques: Scaled Fovea and GrabCut, compared against the existing manual colour calibration process. Using images taken from the competition field, the various techniques were applied to extract the required training data and produce a colour lookup table. The method were then evaluated based on the performance of their colour lookup tables in correctly classifying the pixels of new test images. Experimental results demonstrated that the automated Scaled Fovea approach has the same accuracy as the existing manual approach but with significantly less time and effort, and thus is a viable alternative. This proof-of-concept lays the foundation for integrating automatic colour calibration into the competition preparation routine.

Contents

Chapter 1

Introduction

1.1 RoboCup SPL

The RoboCup Soccer competition consists of five leagues, but for this project the main focus will be the Standard Platform League (SPL). In the SPL, teams use the same hardware, which allows them to focus on creating innovative software solutions. In 2013, the SPL introduced some major changes, notably increasing the field size from 6 x 4m to 9 x 6m and the number of playing robots from 4 to 5. The former poses perhaps the greatest challenge since the robots continue to be of the same specification, while the latter has severely affected the mechanics of team play, both of which means the existing code has to be updated to reflect the change.

1.2 Aldebaran Nao

The Nao is a programmable, 57-cm tall humanoid robot developed by Aldebaran Robotics, France. It is widely used as a research tool by academics from all over the world, mainly because of its large degrees of freedom, wide range of sensors and affordability. The Nao is the standard hardware platform on which the SPL uses for competition. For the 2013 competition, the Nao's used were the Version 4 (V4) H25, which has 25 degrees of freedom.

1.3 rUNSWift

rUNSWift is the team from The University of New South Wales in Sydney, Australia, and has competed in every RoboCup competition since 1999. It mainly consists of undergraduate students who perform research on a plethora of topics such as vision, behaviour, motion and localisation, and have been ranked among the top four within the RoboCup SPL.

1.4 Feature Detection

The main way to detect objects in the robotics fields continues to be vision. However extracting information from 2D computer vision continues to be an inherently difficult problem due to the fact that the data originates from a 3D environment. In the context of the RoboCup competition, all objects are colour-coded (ball, goal, field lines, etc) and thus the dominant feature detection algorithms in the league remain heavily reliant on colour information. The camera images retrieved tend to be negatively affected by factors such as lighting intensity, angle of light projection and colour discrepancies of the objects themselves, which means that with the continued trend of relaxing the requirements of the competition environment setup, the computer vision algorithms employed must be increasingly robust to these fluctuations.

1.5 Colour Calibration

As a result of lighting fluctuations which bring about colour distortion, it is necessary for the robot to perform colour calibration whereby colours in the new domain are remapped to their true colour. This is so that the robots can continue to rely on the camera to perform feature detection. Traditionally this procedure was done manually for each camera on each robot since there tended to be noticeable differences in each camera, but this amounted to repetitive and time-consuming work. Typically (and in the case of the rUNSWift team) this is performed manually offline via a graphical user interface and the output is a colour lookup table that allows for constant time lookup during competition. In recent years, teams have moved towards automated online and offline colour calibration with varying degrees of success.

1.6 Aims

This project aims to present a proof-of-concept which addresses various criteria:

- **Accuracy** - The new methodology should be able to produce a colour lookup table with a similar or better accuracy than those produced by the existing manual process.
- **Speed** - Currently the manual process of performing colour calibration is time-consuming and repetitive, which should instead be automated so that the time spent on colour calibration is reduced from about 1 hour to 5 minutes.
- **Compatibility** - The end result of the new method should produce the same files as the old method, which has the effect of allowing the colour calibration methodology to be swapped easily. This modularity would allow for quick switching between the manual and automatic way without having to modify other components in the rUNSWift vision pipeline.
- **Practicality** - The method used to automate the colour calibration process must be suitable in a time-scarce competition setting. This means that the user should not need any fine-tuning or further adjustments.

Chapter 2

Background

2.1 rUNSWift Vision Overview

The very first step of the rUNSWift vision pipeline involves the extraction of the raw images from the robot camera. In the new V4 Naos cameras, the maximum resolution is 960p but this is rarely used because of the extremely low frame rate, and as such the lower resolution of 640 x 480 pixels is used instead. This is a YUV422 formatted image stream which is then processed through a multitude of steps to produce saliency images that are relied upon by the rest of the rUNSWift system. The saliency images produced are at a reduced size of 160 x 120 pixels (top camera) or 80 x 60 pixels (bottom camera), and consists of:

1. **Grey Scale Saliency** - only luminance information is stored (Y component of YUV) and is used to detect edges for the Edge Saliency image (Figure 2.1b)

2. **Edge Saliency** - provides candidate edges that are used to perform field line detection (Figure 2.1c)

3. **Colour Saliency** - classifies each individual pixel into one of the mapped colours which are used for detecting the ball and goalposts and as a sanity check for field line detection (Figure 2.1d)

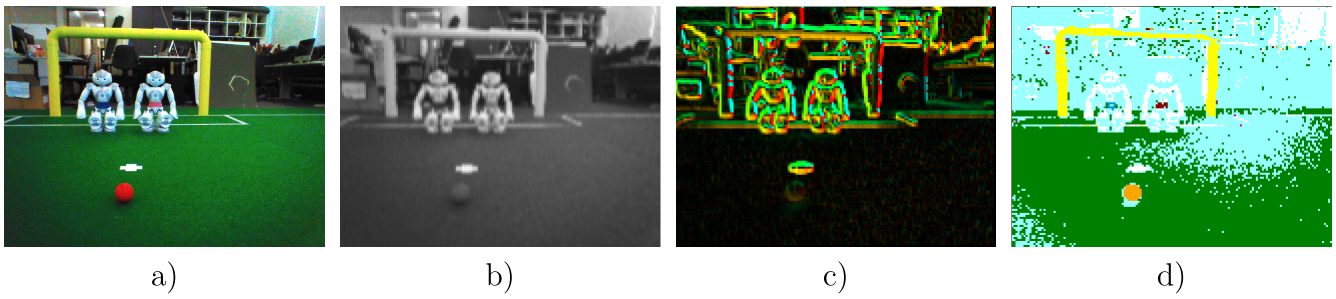


Figure 2.1: from left to right: a) Original RGB image, b) Greyscale Saliency image, c) Edge saliency image, and d) Colour saliency image

2.2 Colour Calibration

This project targets the production of the colour saliency image described in the previous section, whereby the aim is to provide an alternative pathway to colour calibration by replacing the manual colour calibration process with an automated one (Figure 2.2).

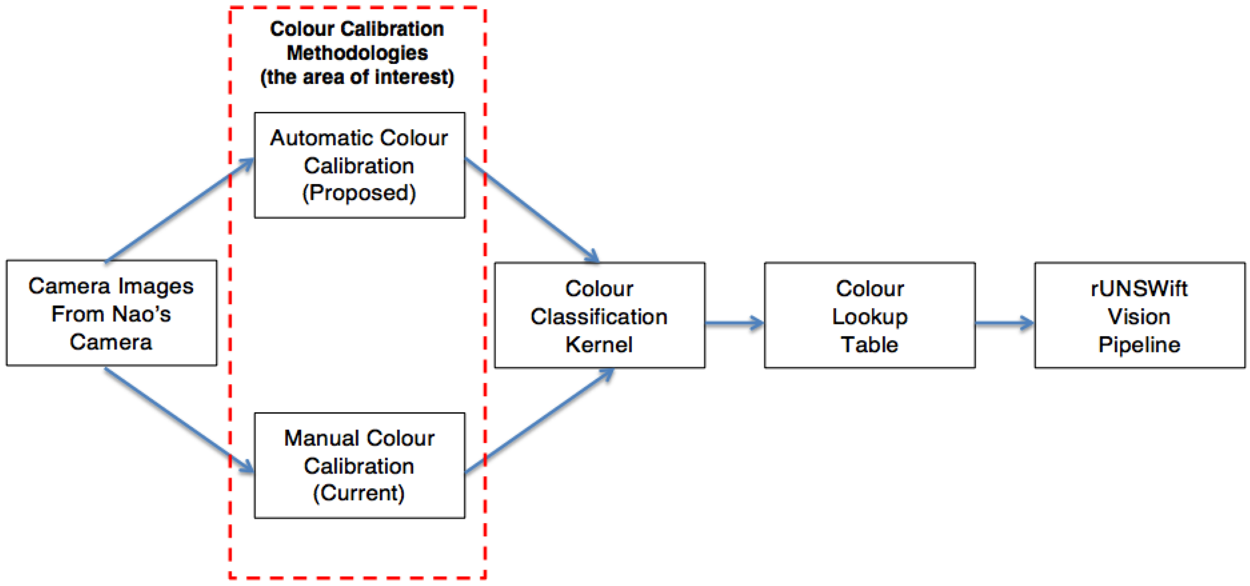

Colour Calibration Methodologies (the area of interest)
Automatic Colour Calibration (Proposed)
Camera Images From Nao's Camera
Manual Colour Calibration (Current)
Colour Classification Kernel
Colour Lookup Table
rUNSWift Vision Pipeline

Figure 2.2: Flowchart showing the part of the rUNSWift vision system that is being targeted for automation

2.3 Related Work

Teams in the SPL have worked towards automatic colour calibration in both online and offline contexts with various degrees of success. There are a variety of strategies employed with a focus on retrieving training data from the image pixels and also statistical methods for colour class separation.

Sridharan and Stone [6] aim to remove the need for manual colour calibration altogether. The authors utilised a colour-coded map of the environment and the known positions of field features in the environment to perform automatic colour calibration in less than five minutes. Though there was no need for labelled training data, the trade-off is that the position of the robot on the map must be determined. The caveat of this approach is that it assumes full availability of the field, which in a competition setting is highly unlikely as many teams share the same field.

Guerrero et al. [7] deviate from most automatic colour calibration methods, combining both offline and online work. The authors used the spatial relationships between colour classes in a given colour space to progressively train the system to differentiate colours. The benefits of an online approach is that it was able to adapt to changing lighting conditions during competition. Though the authors were able to demonstrate the use of such a method online, they did note that there was a significant computational resource usage which rendered it ‘not good enough to play soccer’.

Furtig, Friedrich and Mester [8] performed supervised learning over on a large set of manually labelled images with varying lighting conditions. The authors then relied on the distribution and spatial relationship of colour in terms of their colour class to fit their model of the competition environment. The classifier was ultimately done online, and the end result was a colour lookup table. However this approach relied heavily on a priori knowledge of the distribution of the colours, which may not necessarily map well for ambiguous colours such as the ball’s orange and the robot jersey’s and goalposts’ yellow.

Khandelwal et al. [9] presented a rather interesting approach whereby the hardware parameters of the camera were automatically adjusted to match the color perceptions of a separate high-quality camera, allowing the re-use of the same color table. This approach is particularly interesting because with the adoption with the newer Nao V4s, there is more scope to use both the top and bottom camera simultaneously. However each camera tended to have significant enough discrepancies that made it unreliable to use the same colour classification table. Unfortunately this approach required the use of a ‘known good camera’ to essentially define the ‘true’ colour of the given pixel and then reference that image with the Nao’s, and this must be done with a relatively static background image. This in practice is difficult to achieve on the competition field as many teams share it.

Henderson, King and Chalup [10] converted the YUV colour space to the HSI space and used that instead as additional information to automate the colour separation process. An expectation maximisation algorithm is then performed to estimate the parameters of the multivariate Gaussian mixtures needed to do the colour separation. The trade-off between the traditional manual calibration and this method are the holes present in the colour space. This method tends to ‘over-segment’, creating artificial boundaries that can cause misclassification that would otherwise have been made ‘unclassified’ or at least ‘ambiguous’.

Zrimee and Wyatt [11] used supervised machine learning to train the vision system to recognise the field objects. The authors aim to make it robust against changing lighting conditions. Edge detection using the Sobel filter is used to form regions, in which the average colour in each region is computed, and then fed into the C4.5 machine learning algorithm to progressively improve. However only red, green and blue were determined to be suitable for such learning and as such is not suitable for competition use where there are far more colours to be classified.

There is a common theme of either using a statistical approach with a priori knowledge of what the general structure of the colour classes are, or a machine learning approach whereby the focus is then on how to extract training data in the form of images from the field and progressively improving its colour perception model. There is also a caveat with attempting an online approach as this would constrain the computational power available, but on the other hand attempting an offline approach would mean the robot is unable to adapt to changing environmental conditions. In this project, the machine learning approach is used in an offline context as the aim is to replace the original manual offline colour calibration process in order to save time during the setup process.

2.4 Goal of the project

In this project, human defined bounding box templates are used to guide the user in aligning the camera. This allows for the extraction of the required training data which is then used to produce a colour lookup table. The key to automating the colour calibration process is to develop algorithms that are able to select good training data in the form of colour pixels, these namely being the Scaled Fovea and GrabCut [5] algorithms. The Scaled Fovea algorithm scales down the bounding box by a fixed percentage and uses the pixels in the new bounding box as training data, while the GrabCut algorithm uses the textures and edges of the image to surround the feature of interest and thereby the source of the training data.

The ultimate aim is to automate the existing manual colour calibration procedure, saving preparation time during competition and debugging. The use of automatic colour calibration for rUNSWift would bridge the gap between the other teams in the league, and thus the aim is for this project to address this need.

Chapter 3

Manual Colour Calibration - The Ground Truth

3.1 Motivation

In order to compare the methodologies presented in this project, there needs to be a control from which a performance benchmark can be obtained. In this chapter the construction of the ground truth image is shown.

3.2 The Ground Truth

The point of reference in comparing the manual and automatic methodologies will be a manually classified, hand-tuned calibration which shall be called the ‘ground truth image’. This differs from the existing manual calibration which uses the existing Offnao debugging utility (see Chapter 4), as the process allows noise to be removed with pixel-by-pixel rectifications in an image-editing tool such as GIMP, whereas in Offnao the utility only provides a quick and rough tool for competition use. The process for creating the ground truth image is described in Algorithm 1.

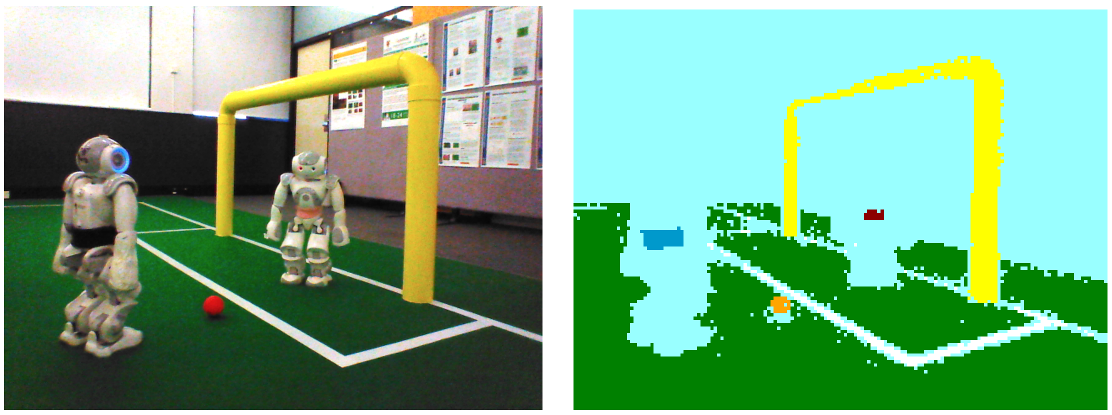
This type of manually labelled data forms the testing suite for comparing the colour lookup tables produced by the various methodologies presented in this project in the form of a pixel-by-pixel error comparison.

3.3 Colours

In order to compare the different methodologies, the classified colours must map to the same colour (Table 3.1). The mapped colour can be any arbitrarily defined colour but for simplicity the saturated version of the colour was chosen.

Feature Type	Mapped Colour
Ball	Orange
Robot blue waistband	Dark cyan
Goalpost	Yellow
Robot red waistband	Dark red
Field green	Green
Field line	White
Other/ Unclassified	Cyan

Table 3.1: Mapping between feature type and colour



a) b)

Figure 3.1: a) Original raw camera image, b) Ground truth image

3.4 Criteria for classifying pixel colour

Each object will have a spectrum of colours within it, which could be a shadow, sheen or even scuff marks. Thus pixels must be selectively classified even within the proximity of the feature as some pixels may not necessarily be representative of the feature itself and adding these candidate points can cause a pollution of the data.

3.5 Conclusion

In this chapter the foundation of comparison in the form of a hand-classified image was demonstrated, which paves the road for showing how well the methodologies explain in the next few chapters perform.

Algorithm 1 Manual Colour Calibration Procedure (The Ground Truth)

```
$truthImage \leftarrow a\ downsampled\ RGB\ image\ of\ the\ original$
for all $pixel \in truthImage$ do
    $pixel \leftarrow (colour \in MappedColours)$        ▷ see Table 3.1 for MappedColours
end for
for all $pixel \in truthImage$ do
    if $pixel \notin MappedColours$ then
        $pixel \leftarrow UNCLASSIFIED$        ▷ as per Table 3.1, UNCLASSIFIED = Cyan
    end if
end for
```

Chapter 4

Manual Colour Calibration - Existing Method

4.1 Motivation

In this chapter, the existing manual and ground truth colour calibration processes demonstrate the need for automation and also how the different methodologies were compared in terms of performance.

4.2 Existing Manual Colour Calibration

rUNSWift has traditionally used manual methods to do colour calibration. Before a new game, a team member would be responsible for moving the robot on the field, placing objects in the robots field of vision so that a representative snapshot of the colours could be recorded in the teams Offnao debugging utility. The video feed would be played back offline and paused at the required images to do the manual colour calibration.

4.3 Procedure

Algorithm 2 presents the current methodology employed by the rUNSWift team for colour calibration. Performing colour calibration for all the robots and for both the top and bottom camera amounts to a large amount of tedious, manual labour, considering that in a typical competition setting there are 5 robots each with 2 cameras, giving a total of 10 cameras to calibrate. This is necessary as the cameras on each of the robots are sufficiently different in manufacturing quality. Currently, the rUNSWift team performs calibration for both cameras on just 1-2 robots and then duplicates the colour lookup table across the fleet. This concession is less than ideal, thus the automatic colour calibration procedure aims to remove the need for such tedious labour.

4.4 Conclusion

In this chapter the existing manual colour calibration method was shown to be a tedious and laborious task, which will be automated using the techniques described in the following chapters.

Algorithm 2 Manual Colour Calibration Procedure (Existing)

```
$manuallyCalibratedImage \leftarrow image\ from\ the\ calibration\ tab\ in\ the\ Offnao\ utility$
$kernel \leftarrow 3D\ colour\ classification\ matrix\ storing\ the\ mapped\ colour\ and\ weight$
$yRadius \leftarrow 10$                          ▷ see Table 8.1 for the predefined Gaussian radii
$uRadius \leftarrow 20$
$vRadius \leftarrow 20$
$weight \leftarrow 1$
for all $robots$ do
    for all $cameras$ do
        for all $colour \in MappedColours$ do         ▷ see Table 3.1 for MappedColours
            while $colour\ is\ not\ fully\ classified$ do
                if $undo$ then
                    $remove\ last\ kernel\ update$
                else
                    $select\ a\ (y, u, v)\ pixel\ in\ manuallyCalibratedImage\ to\ be\ classified\ as\ the\ colour$
                    $update\ kernel\ with\ addGaussian(y, u, v, weight, colour, yRadius, uRadius, vRadius)$
                                                 ▷ see Section 8.4 for the addGaussian procedure
                end if
            end while
        end for
        $colourLookupTable \leftarrow convert\ the\ kernel\ to\ a\ colour\ lookup\ table$
    end for
end for
```

Chapter 5

Templating using Annotated Bounding Boxes

5.1 Motivation

In the manual process, the human user is able to quickly identify the features in the video stream. This chapter lays the foundation for the transition from manual to automatic by developing a system that can quickly approximate the location of the feature, which provides guidance to the Scaled Fovea and GrabCut algorithms.

5.2 Template Definition

The templates contain predefined bounding boxes annotated with data that will be used by the automatic colour calibration procedures (Table 5.1).

Data Field	Comments
colour	An integer representing the different colours in the SPL (e.g. white for the field lines, orange for the ball, etc)
x	An integer representing the x pixel coordinate of the top-left corner of the bounding box
y	An integer representing the y pixel coordinate of the top-left corner of the bounding box
width	An integer representing the pixel distance along the x axis
height	An integer representing the pixel distance along the y axis

Table 5.1: Data fields in the feature templates

OpenCV convention dictates that the top-left corner of the image be the origin $(0, 0)$ and as such the positive x axis runs to the right and the positive y axis runs downwards.
The format of the templates are plain text files with whitespace delimiting the data fields (no leading whitespace is allowed however). A feature is a line of 5 integers, and the rest are ignored as comments. The purpose of the templates is to allow for quick runtime changes during the

debugging phase of creating the initial template itself and also gives the user a plethora of templates to pick from for their custom purposes.

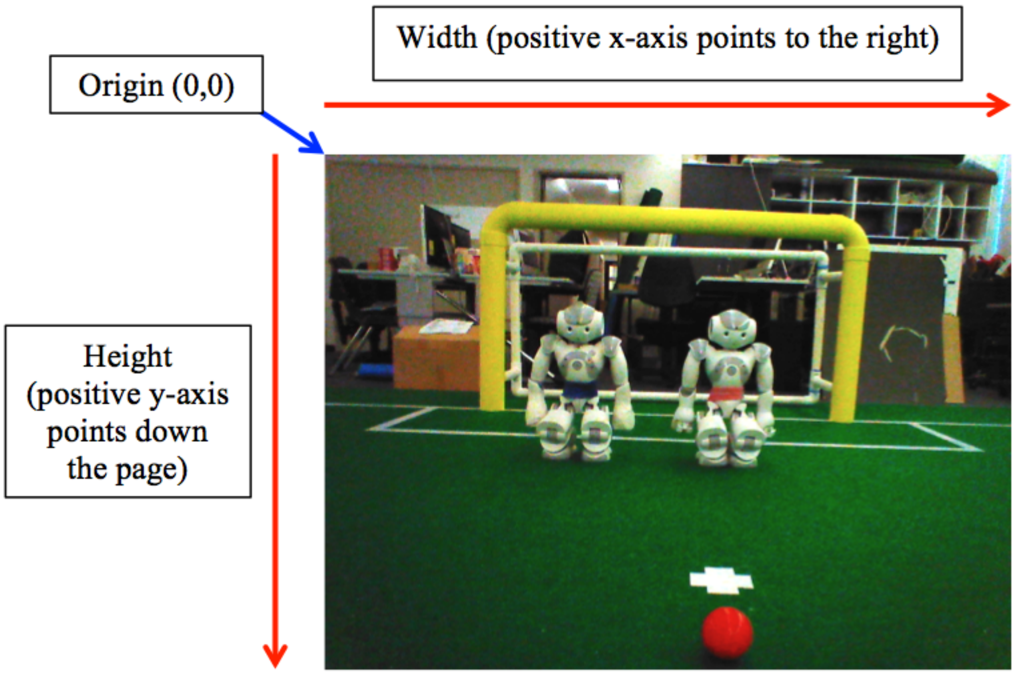
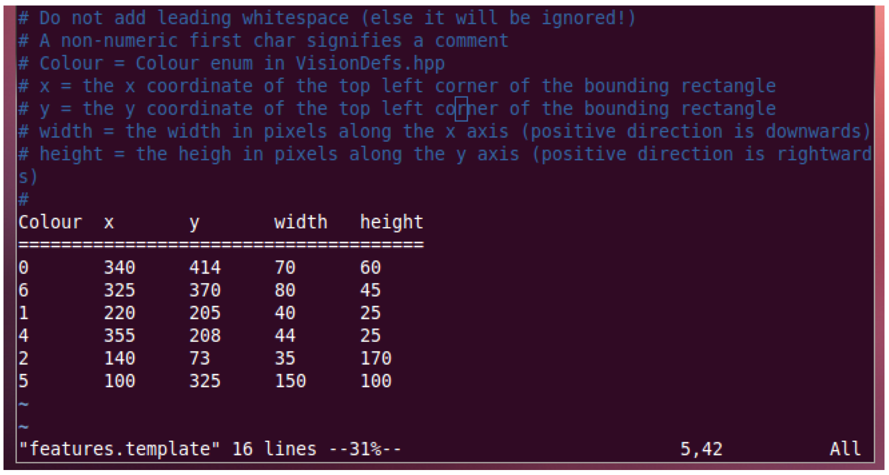

Width (positive x-axis points to the right)
Origin (0,0)
Height (positive y-axis points down the page)

Figure 5.1: Definition of origin and the positive x- and y- axes in OpenCV convention



```
# Do not add leading whitespace (else it will be ignored!)
# A non-numeric first char signifies a comment
# Colour = Colour enum in VisionDefs.hpp
# x = the x coordinate of the top left corner of the bounding rectangle
# y = the y coordinate of the top left corner of the bounding rectangle
# width = the width in pixels along the x axis (positive direction is downwards)
# height = the heigh in pixels along the y axis (positive direction is rightward
s)
#
Colour  x       y       width   height
======================================
0       340     414     70      60
6       325     370     80      45
1       220     205     40      25
4       355     208     44      25
2       140     73      35      170
5       100     325     150     100
~
~
"features.template" 16 lines --31%--                        5,42          All
```

Figure 5.2: Screenshot of a typical features template

5.3 Camera Alignment

With the templates defined, the user then has to align the robot on the field such that the features are within the bounding boxes. Note that the templating system essentially provides a rough estimate of the location of the feature and not a pixel-perfect definition of its location. The main purpose of this is to make it lenient enough for the user to quickly position the camera to the features but bounded enough such that the Scaled Fovea extracts only the core of the feature. It is very important that the Scaled Fovea only select pixels that are certain to be part of the feature, otherwise the fovea used will cause degraded performance for both automatic colour calibration processes.

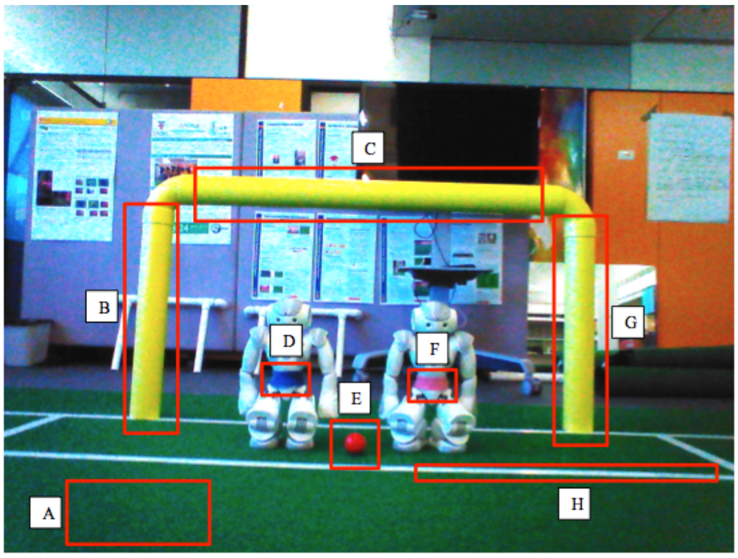


Figure 5.3: Bounding boxes (red) overlaid over raw camera image representing each feature stored in template file: a) Green - field, b) Yellow - goalpost, c) Yellow - goalpost, d) Blue - robot blue waistband, e) Orange - ball, f) Red, robot red waistband, g) Yellow - goalpost, h) White - field line

5.4 Environment Setup

In the colour calibration process, there is a trade-off between time and accuracy of the training data extraction. To save time, the environment can be setup to contain all the features of the field in the one camera image and then allow the automatic colour calibration run once. On the other hand to increase accuracy, the environment can be set up more sparsely by taking multiple runs of the calibration for several point of views of the field. The former is the most ideal way as the core focus of this project is to significantly reduce the time for calibration, which would be fitting for a competition setting.

5.5 Conclusion

In this chapter, the first step of creating templates for the new automatic colour calibration process was examined, which shall provide the context for the pixel extraction algorithms to work under.

Chapter 6

Automatic Colour Calibration - Scaled Fovea

6.1 Motivation

Once a template has been defined, there is then the need to extract training data within the bounding box. In this chapter, the use of a Scaled Fovea provides an algorithm on its own to obtain the training data and also provides a heuristic for the GrabCut algorithm.

6.2 Scaled Fovea

A Scaled Fovea is simply a scaled-down area of the bounding box. For good results and retaining generality, it is best to scale down each dimension by the same amount to retain a similar shape and also centre the fovea to avoid extracting extreme candidate points that have a misleading colour representation. Since the templates essentially defines the approximate location of the feature, a Scaled Fovea would be able to extract most, if not all, of the strong candidate points as training data.

6.3 Calculating the Scaled Fovea

OpenCV has a variety of ways of constructing a rectangular region. For this project, it was chosen to be defined by the (x, y) coordinates of the top-left corner of the region with the width (w) and height (h).

Let $SCALE$ be a proportionality constant in the range 0-1 (in this project $SCALE$ (=0.20) was chosen based on empirical results). Then the Scaled Fovea can be defined by a rectangle with the following (x', y', w', h'):

$$x' = x + \frac{1}{2}w(1 - SCALE)$$

$$y' = y + \frac{1}{2}h(1 - SCALE)$$

$$w' = SCALE \times w$$

$$h' = SCALE \times h$$

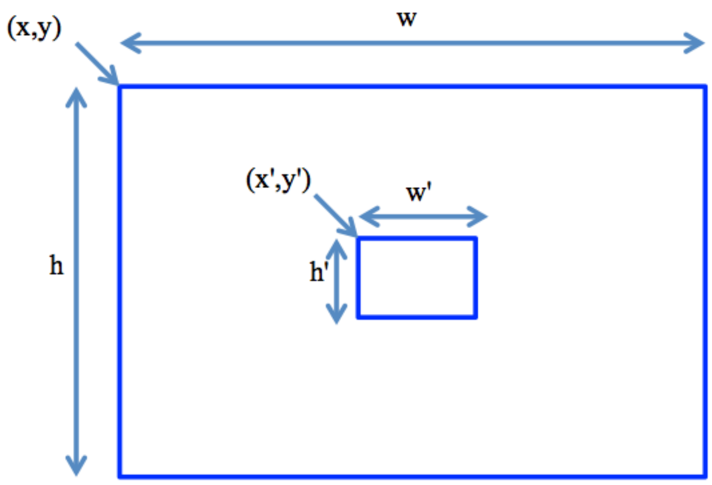


Figure 6.1: Bounding box (outer rectangle) and the Scaled Fovea (inner rectangle).

6.4 Procedure

With the Scaled Fovea calculated, the final process is to select the pixels that will become training data (Figure 6.2). The algorithm then proceeds to pass all pixels within this Scaled Fovea to generate the colour lookup table.

Algorithm 3 presents the proposed new methodology. This assumes the template has already been written beforehand, which only needs to be done once for a given scenario and can be shared between the Scaled Fovea and GrabCut approaches. Writing the template is a simple matter of populating a plain text file with data that describes the feature location and colour. The aim of this new procedure is to delegate the task of selecting the training data to the proposed algorithms, being guided by the use of templates.

6.5 Conclusion

In this chapter, the first automatic colour calibration method was demonstrated.

Algorithm 3 Automatic Colour Calibration Procedure (Scaled Fovea)

```
$autoFoveaImage \leftarrow image\ from\ the\ calibration\ tab\ in\ the\ Offnao\ utility$
$kernel \leftarrow 3D\ colour\ classification\ matrix\ storing\ the\ mapped\ colour\ and\ weight$
$SCALE \leftarrow 0.20$                                          ▷ see Section 6.3
$yRadius \leftarrow 2$                    ▷ see Table 8.1 for the predefined Gaussian radii
$uRadius \leftarrow 4$
$vRadius \leftarrow 4$
$weight \leftarrow 1$
for all $robots$ do
    for all $cameras$ do
        for all $f \in featureTemplate$ do
            $x' \leftarrow f.x + \frac{1}{2}(f.w)(1 - SCALE)$
            $y' \leftarrow f.y + \frac{1}{2}(f.h)(1 - SCALE)$
            $w' \leftarrow SCALE \times f.w$
            $h' \leftarrow SCALE \times f.h$
            $fovea \leftarrow extract\ pixels\ in\ rectangle\ defined\ by\ (x', y', w', h')$
            for all $pixel \in fovea$ do
                $update\ kernel\ with\ addGaussian(y, u, v, weight, colour, yRadius, uRadius, vRadius)$
                                                  ▷ see Section 8.4 for addGaussian
            end for
        end for
        $colourLookupTable \leftarrow convert\ the\ kernel\ to\ a\ colour\ lookup\ table$
    end for
end for
```

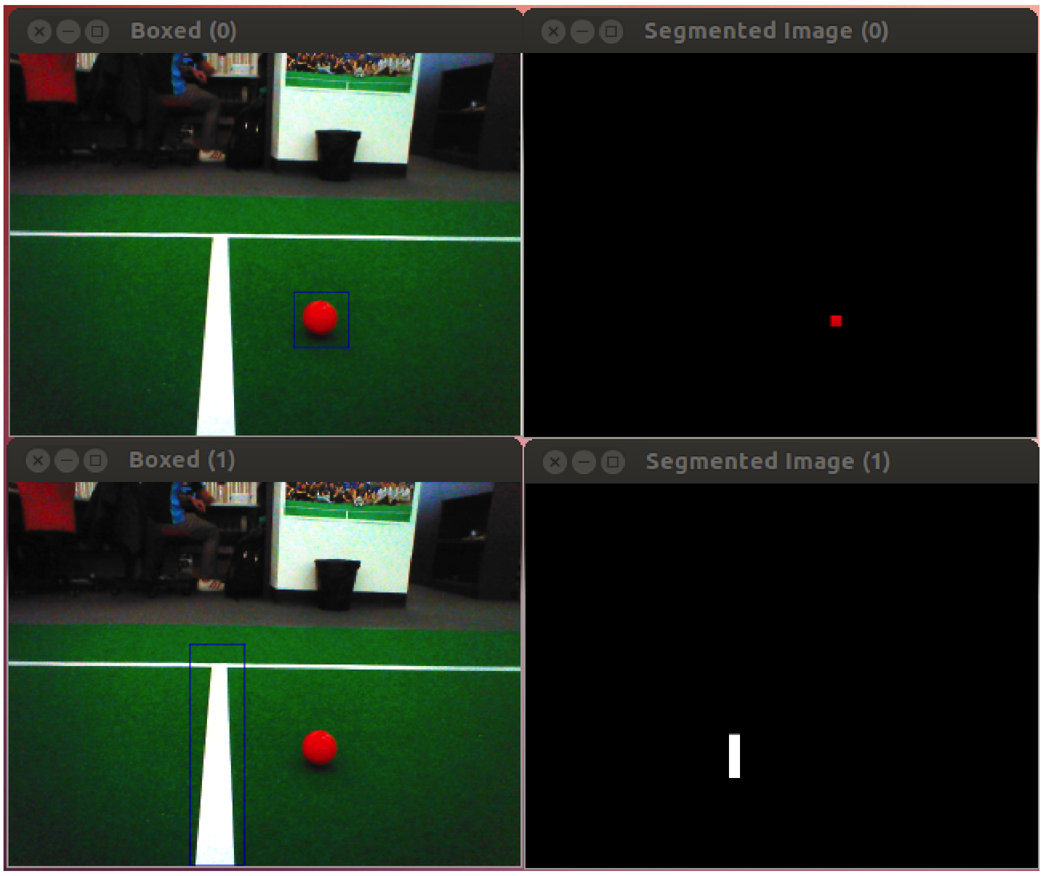


Figure 6.2: (Top left) Bounding box for the ball; (Top right) Extracted foreground pixels of the ball using Scaled Fovea; (Bottom left) Bounding box for the field lines; (Bottom right) Extracted foreground pixels of the field lines using Scaled Fovea

Chapter 7

Automatic Colour Calibration - GrabCut

7.1 Motivation

The bounding boxes in each template define only the rough location of features. In the previous chapter the Scaled Fovea used a down-sized bounding box to select the pixels as training data. In this chapter, the GrabCut algorithm is proposed as a novel approach which is able to extract foreground features by using features from the Scaled Fovea algorithm. The aim of this approach is to compare the performance between using all of the pixels defining the feature and using the centre region of the feature as per the Scaled Fovea approach.

7.2 GrabCut

Rother, Kolmogorov and Blake [5] devised the GrabCut algorithm to efficiently extract foreground images, utilising colour information in the form of texture and edges to iteratively 'surround' the feature of interest. In each iteration, the features boundary is improved by 'energy minimization', whereby the histogram of the foreground and the background depict the 'energy levels'. However this relies on having the foreground being distinct from the background enough such that a fairly distinct border can be formed to separate the two.

The main motive for using GrabCut was the minimal human intervention required to extract the foreground pixels, all that is required is a bounding box with the target object clearly being the main focus of the image. Typically the creation of such bounding boxes is interactive whereby a human user can drag the box over the target region. However since the aim is for an automated calibration process, the bounding boxes can instead be fixated, which only requires the human to position the robots camera to align the features to fit the template. The algorithm extracts the feature in its entirety based on colour. This makes it an immensely powerful tool for pixel extraction. It must be noted however, that this would also mean extracting pixels that may be similar to other features colour due to environmental difference and as such can pollute the data set.

7.3 OpenCV Implementation of GrabCut

OpenCV provides an implementation of the GrabCut algorithm. The function prototype that is of interest is shown below, with the parameters of interest explained in context of this project:

```
void grabCut(InputArray img, InputOutputArray mask, Rect rect,
   InputOutputArray bgdModel, InputOutputArray fgdModel,
   int iterCount, int mode=GC_EVAL)
```

- **img** - Input 8-bit 3-channel image; this is the RGB colour image converted from the raw YUV colour image from the robot's camera
- **mask** - Input/output 8-bit single-channel mask. This marks points on the image where the user thinks to be a certain/probable and foreground/background pixel. After performing GrabCut, it stores the segmentation result which can be used to generate the segmented colour image.
- **rect** - bounding box containing a segmented object. The pixels outside of the region of interest are marked as 'certain background'. The parameter is only used when the mode is `GC_INIT_WITH_RECT` and is provided by the template definition.
- **bgdModel** - Temporary array for internal use, not used for tweaking.
- **fgdModel** - Temporary array for internal use, not used for tweaking.
- **iterCount** - Number of iterations the algorithm should make before returning the result. The GrabCut algorithm is able to refine the segmentation using an iterative process, however in practice this option is only useful for high-resolution images that require fine-tuning and is unnecessary for this particular application.
- **mode** - there are various modes of operation for the GrabCut algorithm that affect the way it initialises the mask

There are two main modes that can be used with the GrabCut algorithm: 1) When the mode is `GC_INIT_WITH_RECT`, the area outside of the rectangle is labeled as background pixels, the area inside as probable foreground pixels; 2) When the mode is `GC_INIT_WITH_MASK`, a single-channel image denotes which pixels are certain/probable and foreground/background, and the area defined by the Scaled Fovea are all marked as certain foreground pixels. Mode 1 requires the least human intervention as it only requires a single bounding box for the feature, however the colour distribution of the foreground object and the background must have a sufficiently different profile for a clean segmentation. On the other hand, Mode 2 requires selection of certain foreground pixels on top of the features bounding box, but this selection can be automated as well by reusing the area defined by the Scaled Fovea. Mode 2 tended to provide the best empirical results and thus was selected as the basis for the GrabCut algorithm.

7.4 Procedure

The initial setup procedure for using GrabCut is the same as the Scaled Fovea approach, however the difference is the algorithms approach to extracting pixels (Algorithm 4). Provided a colour image, the GrabCut implementation can have one of two results:

- A well-segmented image containing the feature, this means GrabCut is returning all pixels marked as 'probably foreground' (Figure 7.1)
- A small window of pixels within the Scaled Fovea, this means GrabCut is returning all pixels marked as 'certain foreground'

The second result is the fallback mechanism that the GrabCut algorithm will revert to, so that in the worse case that GrabCut cannot extract the foreground image properly, it will be the same as the Scaled Fovea approach for that particular feature.

The foreground pixels are then processed as training data for the colour lookup table in the same way described with the Scaled Fovea approach.

7.5 Conclusion

In this chapter we saw the GrabCut algorithm as a way of extracting the features with minimal human intervention and providing the training data necessary for the next step of generating the colour lookup table.

Algorithm 4 Automatic Colour Calibration Procedure (GrabCut)

```
$autoGrabCutImage \leftarrow image\ from\ the\ calibration\ tab\ in\ the\ Offnao\ utility$
$kernel \leftarrow 3D\ colour\ classification\ matrix\ storing\ the\ mapped\ colour\ and\ weight$
$SCALE \leftarrow 0.20$
$yRadius \leftarrow 2$                              ▷ see Table 8.1 for the predefined Gaussian radii
$uRadius \leftarrow 4$
$vRadius \leftarrow 4$
$weight \leftarrow 1$
for all $robots$ do
    for all $cameras$ do
        for all $f \in featureTemplate$ do
            $x' \leftarrow f.x + \frac{1}{2}(f.w)(1 - SCALE)$
            $y' \leftarrow f.y + \frac{1}{2}(f.h)(1 - SCALE)$
            $w' \leftarrow SCALE \times f.w$
            $h' \leftarrow SCALE \times f.h$
            $fovea \leftarrow extract\ pixels\ in\ rectangle\ defined\ by\ (x', y', w', h')$
            for all $pixel \in fovea$ do
                $pixel \leftarrow CERTAIN\_FOREGROUND$
            end for
            $extractedFeatureImage \leftarrow perform\ grabCut\ on\ the\ image\ with\ the\ marked\ fovea$
            if $extractedFeatureImage\ is\ empty$ then
                $candidatePoints \leftarrow fovea$
            else
                $candidatePoints \leftarrow extractedFeatureImage$
            end if
            for all $pixel \in candidatePoints$ do
                $update\ kernel\ with\ addGaussian(y, u, v, weight, colour, yRadius, uRadius, vRadius)$
                                                            ▷ see Section 8.4 for addGaussian
            end for
        end for
        $colourLookupTable \leftarrow convert\ the\ kernel\ to\ a\ colour\ lookup\ table$
    end for
end for
```

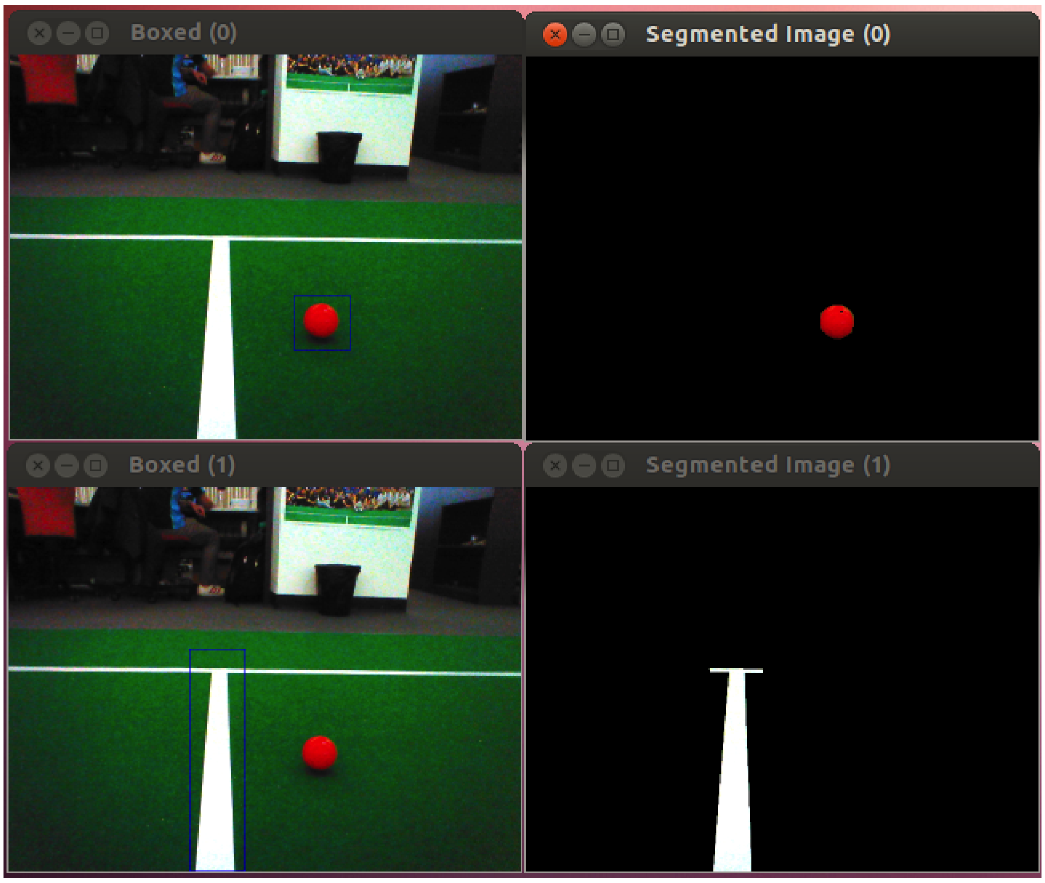


Figure 7.1: (Top left) Bounding box for the ball; (Top right) Extracted foreground pixels of the ball using GrabCut; (Bottom left) Bounding box for the field lines; (Bottom right) Extracted foreground pixels of the field lines using GrabCut

Chapter 8

Generating the Colour Lookup Table

8.1 Motivation

The end product of the colour calibration step is the generation of the colour lookup table. This reference table gives the constant time lookup necessary for the robot to classify the raw pixel colours it perceives and produce a saliency image for the rUNSWift vision pipeline. The aim of the automatic colour calibration is to produce such a lookup table in minimal time.

8.2 Colour Lookup Table

The colour lookup table is essentially a 3D array indexed by the Y, U and V components of a given pixel. This therefore allows for constant time lookup suitable for use during competition for online creation of the colour saliency image.

8.3 Training Data

The training data are the extracted pixels provided as the result of each algorithm. In the manual method, these were determined by the human user, while in the automatic methods these were calculated based on the feature template.

8.4 Weight Distribution using Gaussians

For each pixel in the training data, a 3D Gaussian with axes in the YUV colour space is used to upvote the neighbouring pixels in the colour classification kernel according the the Gaussian distribution. The primary aim is to quickly populate the colour space and this is where the Gaussian distribution fills in most of the voids quite nicely with the appropriate weighting. This greatly reduces the input points needed to build the colour table, at the expense of the ability to fine tune.

To better understand the nature of this Gaussian update, consider the following pseudo-function prototype which forgoes the implementation specificity in the existing rUNSWift codebase:

```
addGaussian(y,u,v,weight,colour,yRadius,uRadius,vRadius)
```

On a high-level view it updates the colour classification kernel with a 3D Gaussian: centred at the (y,u,v) pixel coordinate that has been selected, radii (yRadius,uRadius,vRadius), weight defined arbitrarily as 1 (it is only used on a relative basis) and finally colour as defined in Table 3.1.

Comparing the YUV radii parameters of the Gaussians between the manual and automatic calibration methods, we observe a ratio of 5:1. This value was empirically determined. The reason behind the downscaled radii is that the automatic colour calibration methods have significantly more candidate points than the manual calibration method. The result is that the new method is able to fill the colour space with smaller Gaussians, resulting in a fine-grained colour lookup table.

Axis	Original Radii - Manual Calibration (px)	New Radii - Automatic Calibration (px)
Y	10	2
U	20	4
V	20	4

Table 8.1: Comparison between the Gaussian radii along each axis

Note that the Y radius is less than the U and V radii, which is mainly to reduce the effect that luminance has on the weighting procedure. In terms of visualising this 3D Gaussian, we can imagine it as an ellipsoid.

8.5 Creating the Colour Lookup Table

Once all the pixels have been processed, the point cloud generated is now used to create the colour lookup table. For each pixel in the YUV space, a nearest neighbour algorithm determines what that pixel value should be classified as. This produces an '.nnmc' file that forms the basis for creating the above mentioned colour saliency images.

8.6 Conclusion

In this chapter the final step in the colour calibration process of generating the colour lookup table was outlined.

Chapter 9

Results

9.1 Motivation

In this chapter, the results produced from both manual and automatic colour calibrations are shown and their performance compared and evaluated.

9.2 Methodologies

Using the manually created ground truth images as the point of reference, the following methodologies will be compared:

- Existing Manual Colour Calibration
- Automatic Colour Calibration with the Scaled Fovea algorithm
- Automated Colour Calibration with the GrabCut algorithm

9.3 Measure of Performance

A pixel by pixel comparison was made between the target methodology with the reference ground truth image. If it was misclassified, upvote by one point, otherwise disregard it. The error rate was then calculated to be a percentage, where:

$$Error\ Rate = \frac{Number\ of\ misclassified\ pixels}{Total\ number\ of\ pixels} \times 100\%$$

9.4 Test Suite

As with many supervised machine learning approaches, there is the need for a set of training data and also a separate set of testing data. However the major deviation from traditional machine learning approaches and their benchmarking practices is that the training data was not meant to be a representative set with similar characteristics to the test data. The reason behind this disparity lies in the fact that the produced colour lookup table needs candidate points for all of the colours in the SPL. In the majority of the scenes in competition, it is highly unlikely to see all colours in the same frame, hence the need for an artificial setup (much alike the existing manual colour calibration methodology).

In order to test the performance of the produced colour lookup table, the selection of the test suite was based upon typical images seen by the robot during a competition game. This would involve a variety of scenarios such as goal post with a ball, many robots with lots of field green, etc which would not necessarily include all classified colours. Note that this is different from the artificial setups used in the training process. Having created the ground truth images for all the test data, any new training data would be immediately run against the test suite automatically using a script, providing instant feedback on the colour lookup table's performance.

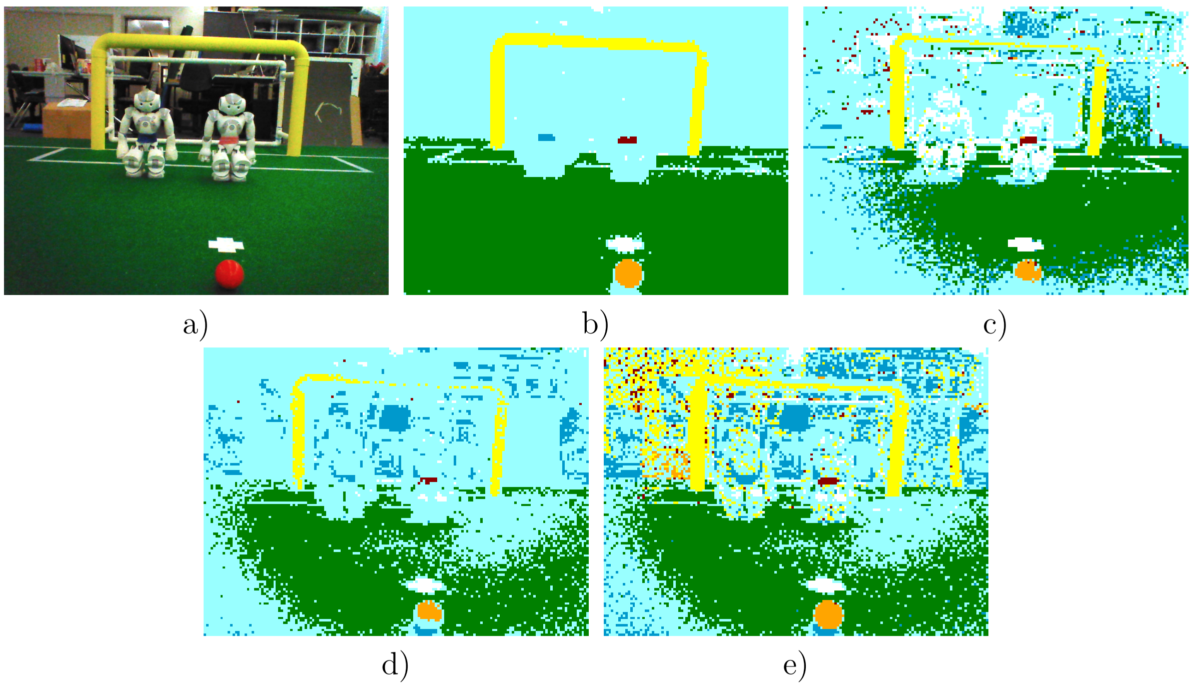


Figure 9.1: Typical set used to evaluate the calibration procedures: a) Original raw camera image, b) Ground truth image, c) Manually calibrated image, d) Automatically calibrated image using Scaled Fovea, and e) Automatically calibrated using Grabcut

9.5 Experimental Results

A total of 6 training images were used, run against 10 testing images (see Appendix 1), resulting in 6 x 10 = 60 colour classification experiments. The pixel-by-pixel comparison was then performed, on which the mean error rate and the standard deviation were then calculated based on these experiments (Table 9.1).

	Manual	Automatic (Scaled Fovea)	Automatic (GrabCut)
Mean Error Rate	34.06%	34.58%	53.75%
Standard Deviation	8.14%	10.36%	16.84%

Table 9.1: Summary table comparing the performance of the different methodologies

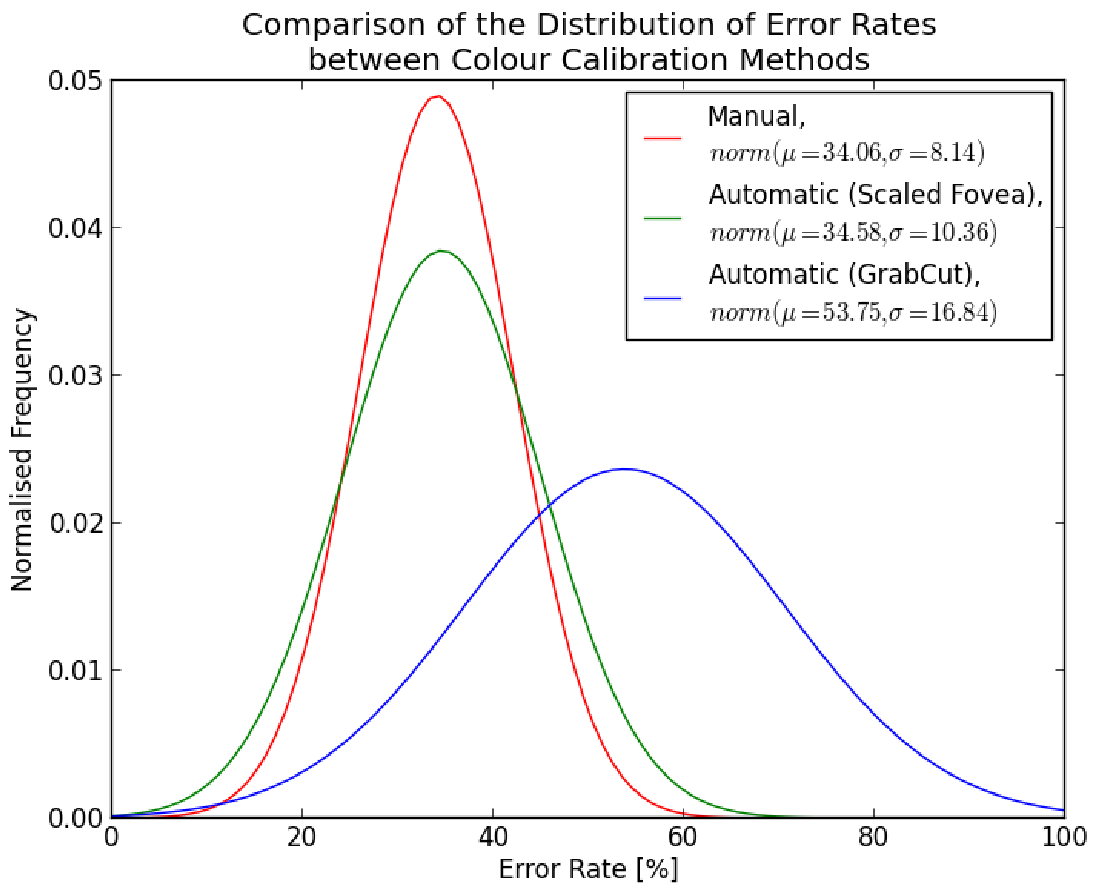


Figure 9.2

9.5.1 Experiment 1

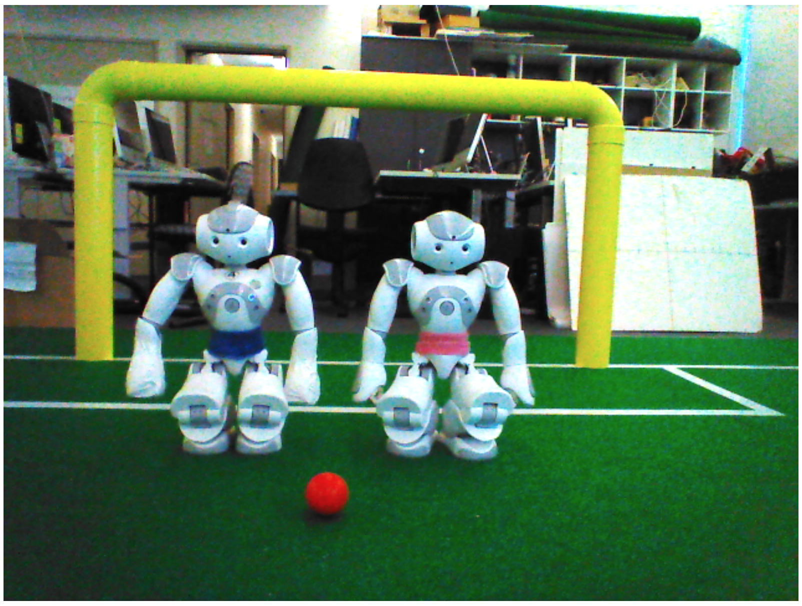


Figure 9.3: Original Raw Image

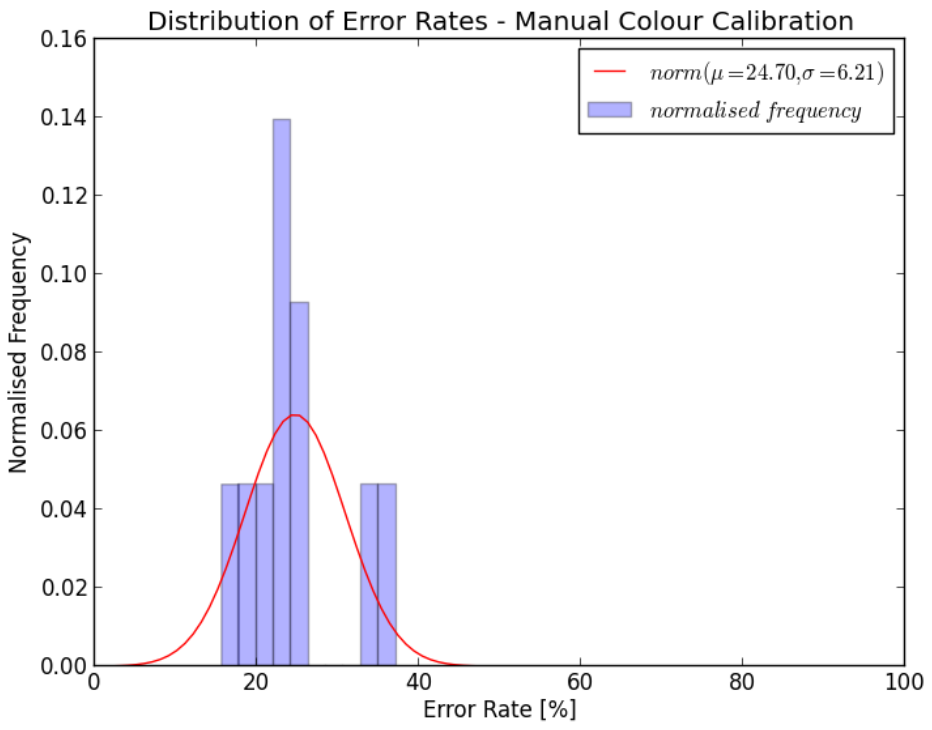


Figure 9.4

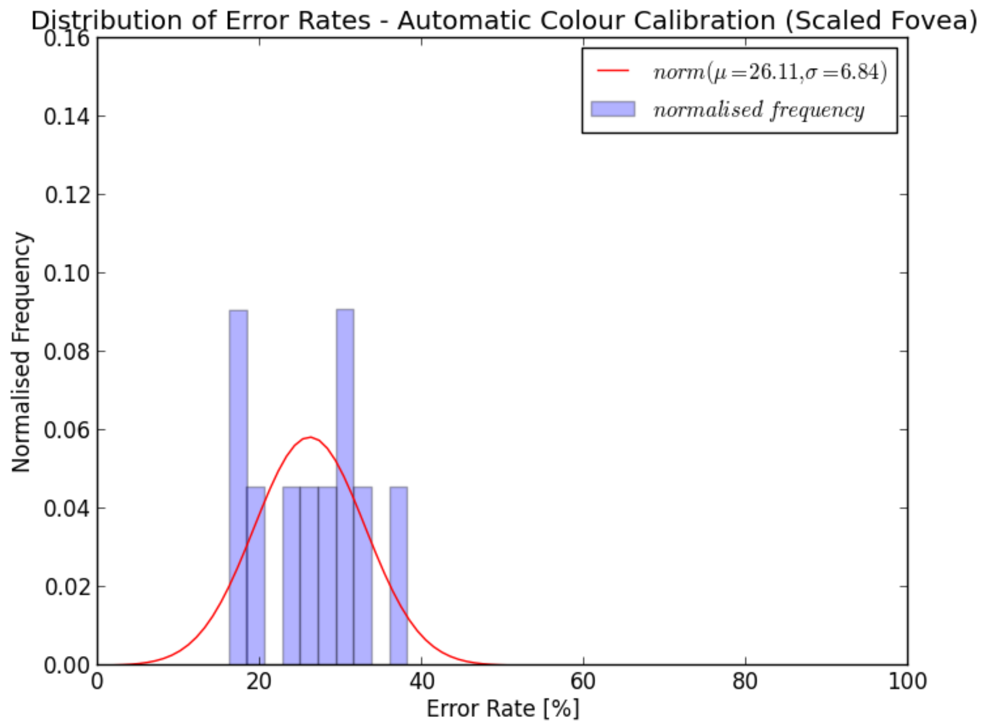


Figure 9.5

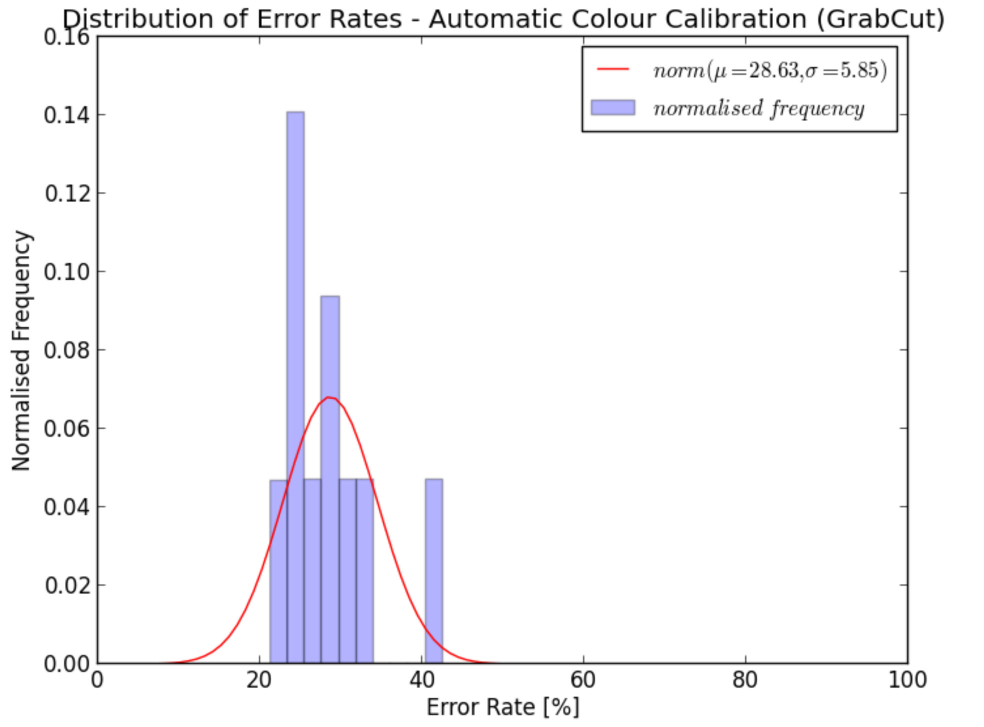


Figure 9.6

9.5.2 Experiment 2

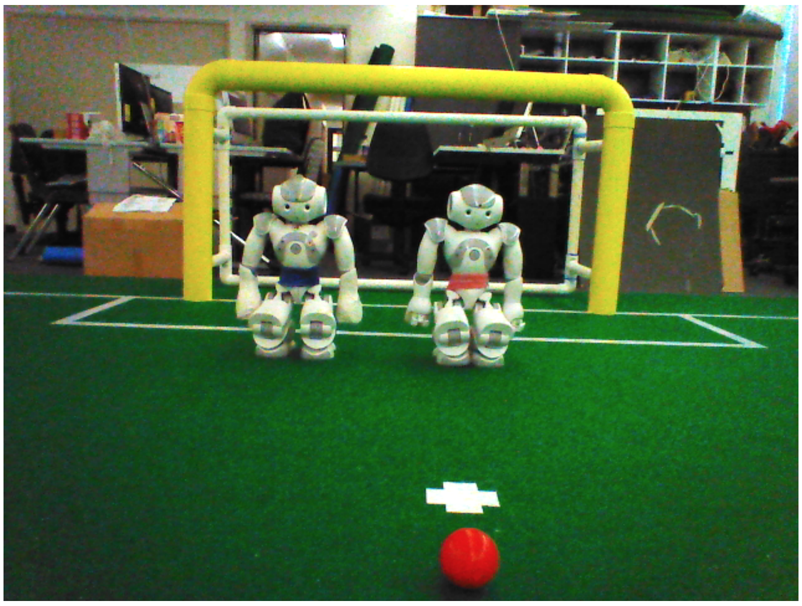


Figure 9.7: Original Raw Image

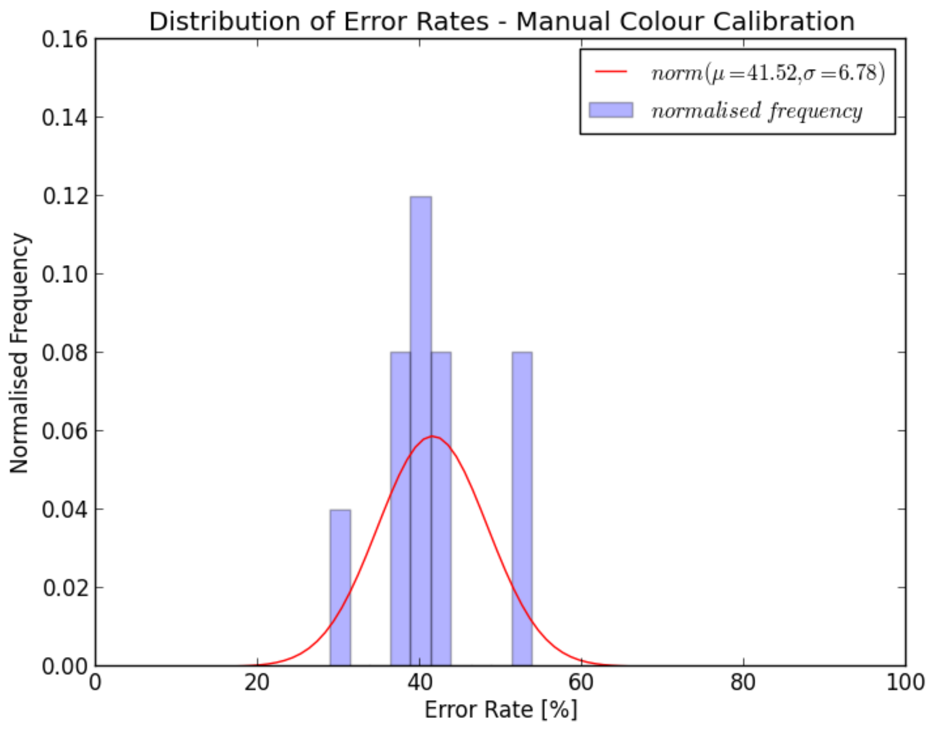


Figure 9.8

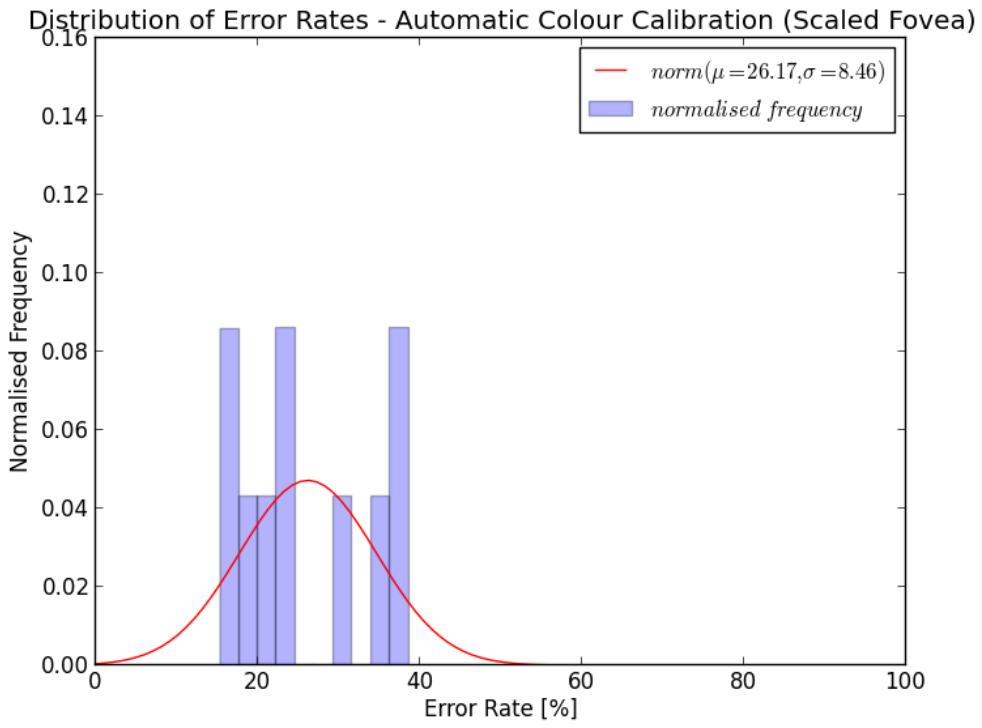


Figure 9.9

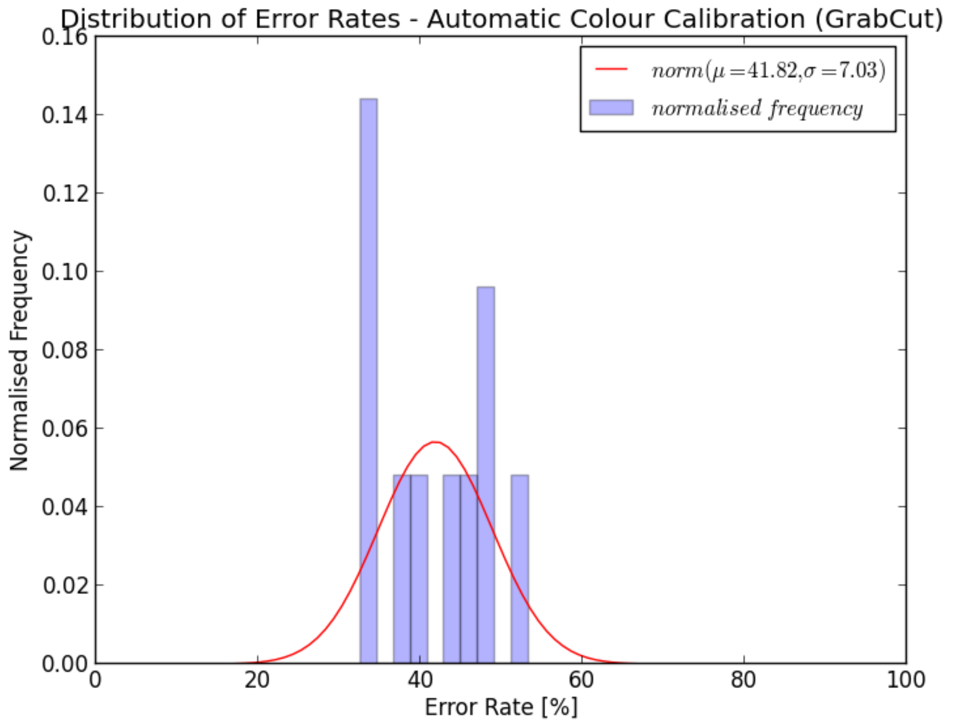


Figure 9.10

9.5.3 Experiment 3

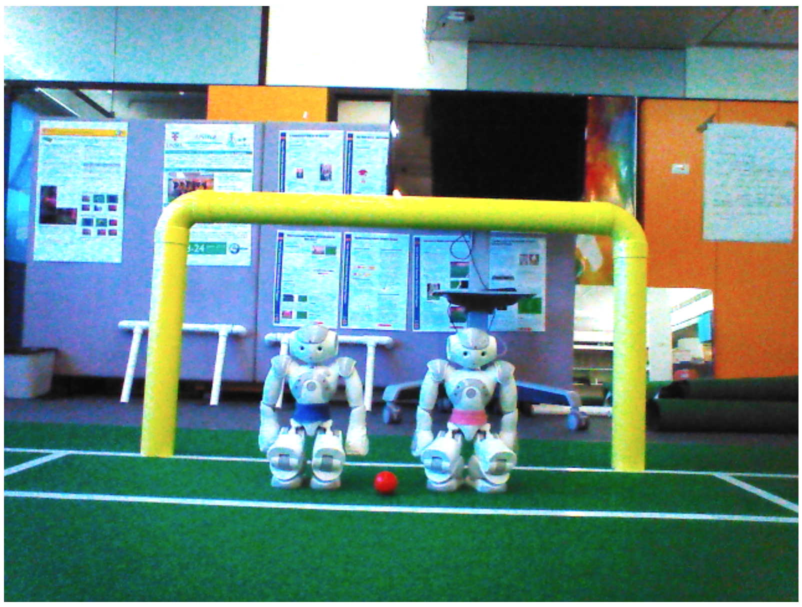


Figure 9.11: Original Raw Image

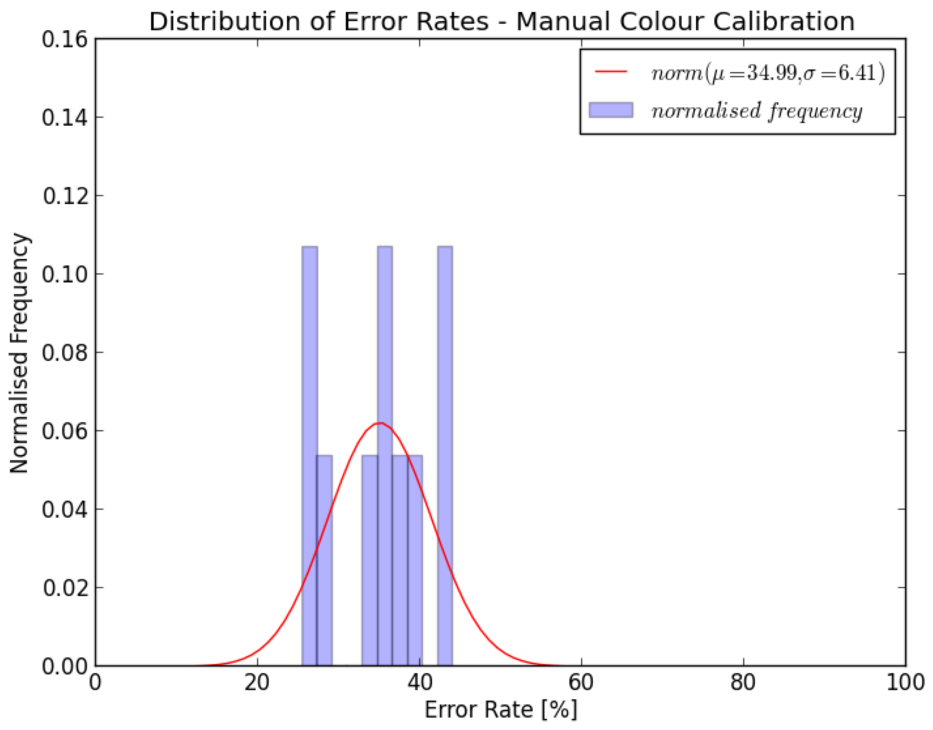


Figure 9.12

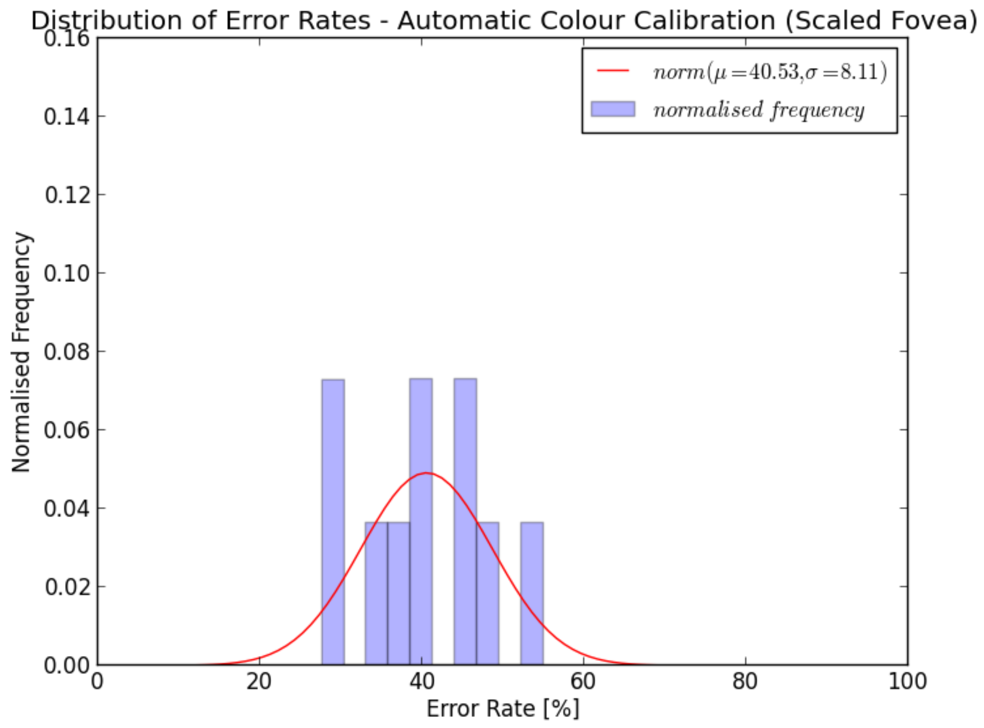


Figure 9.13

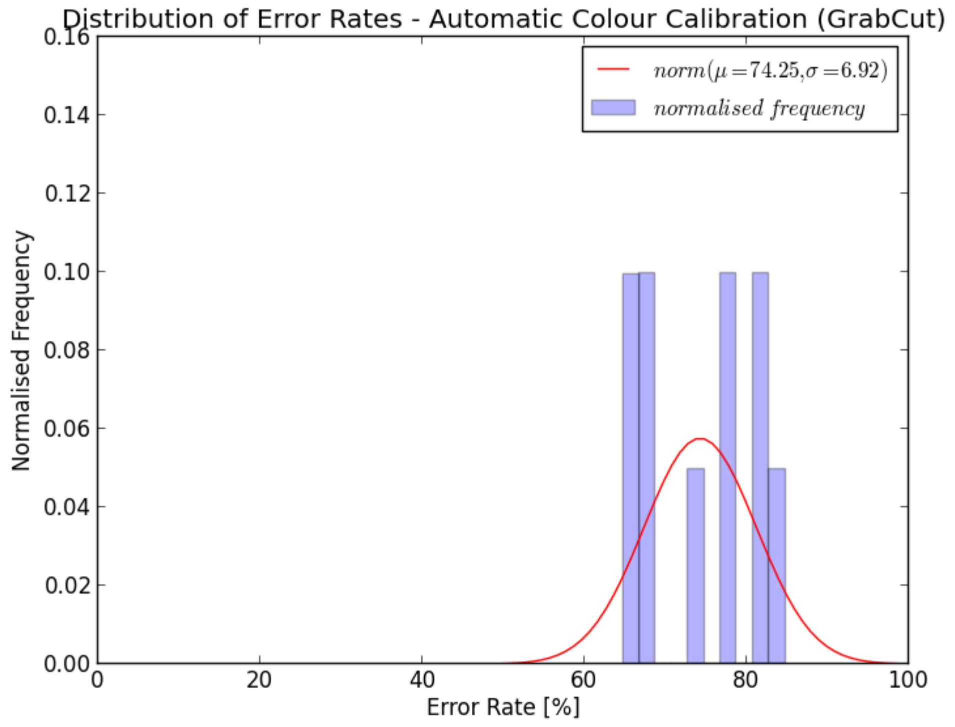


Figure 9.14

9.5.4 Experiment 4

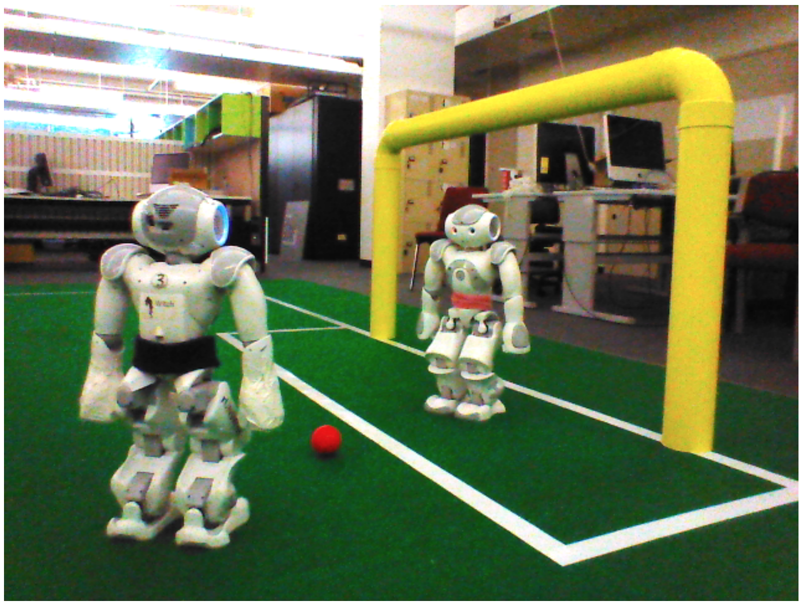


Figure 9.15: Original Raw Image

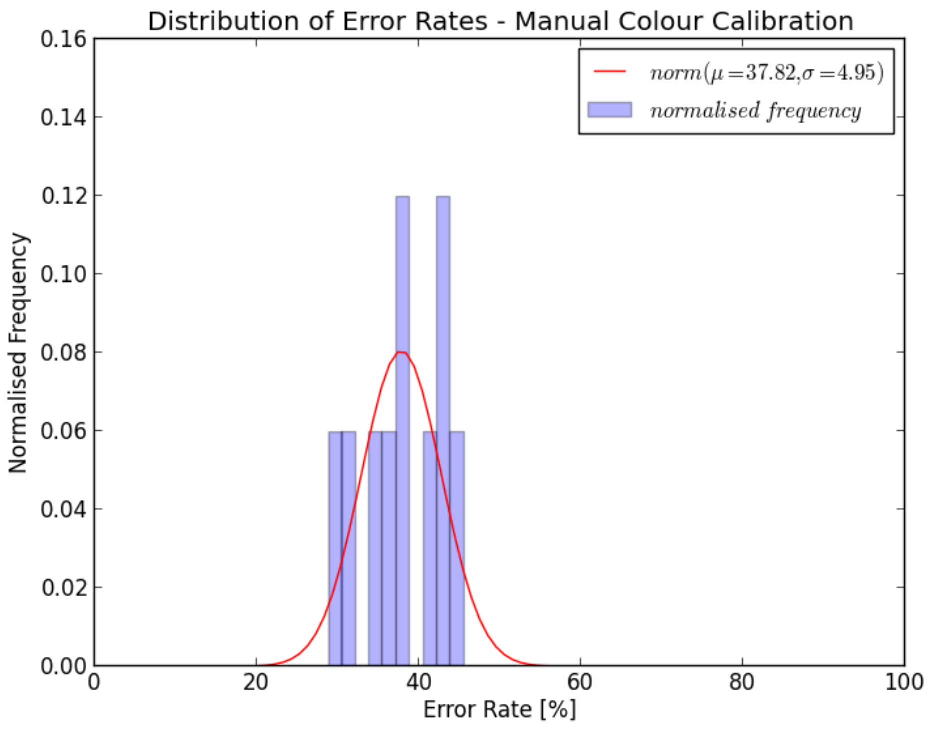


Figure 9.16

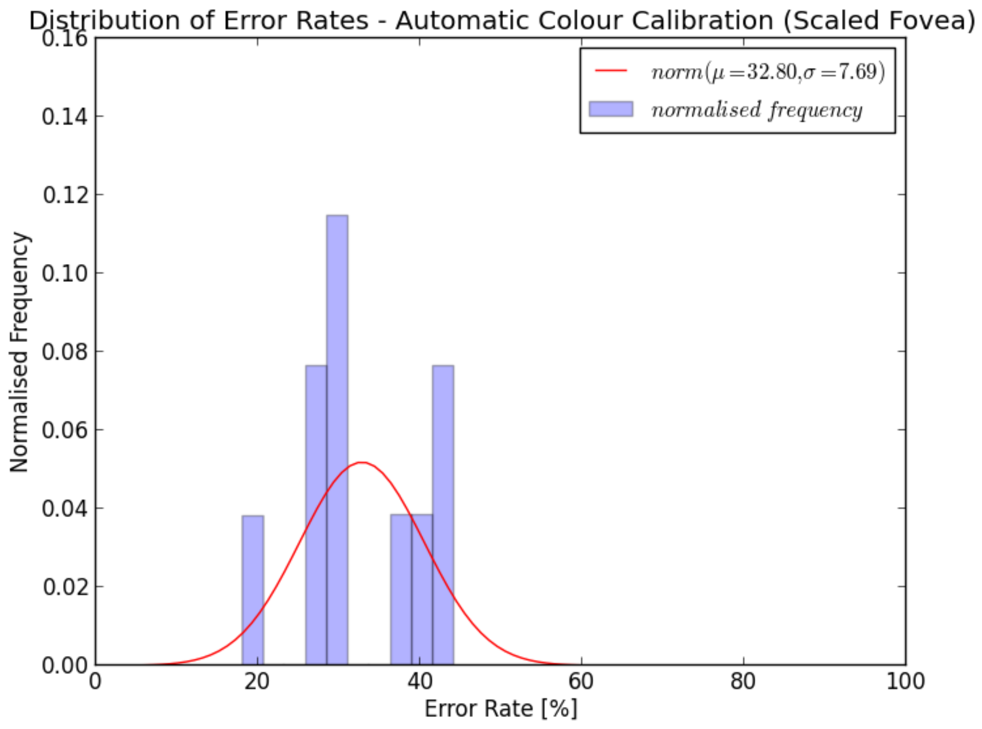


Figure 9.17

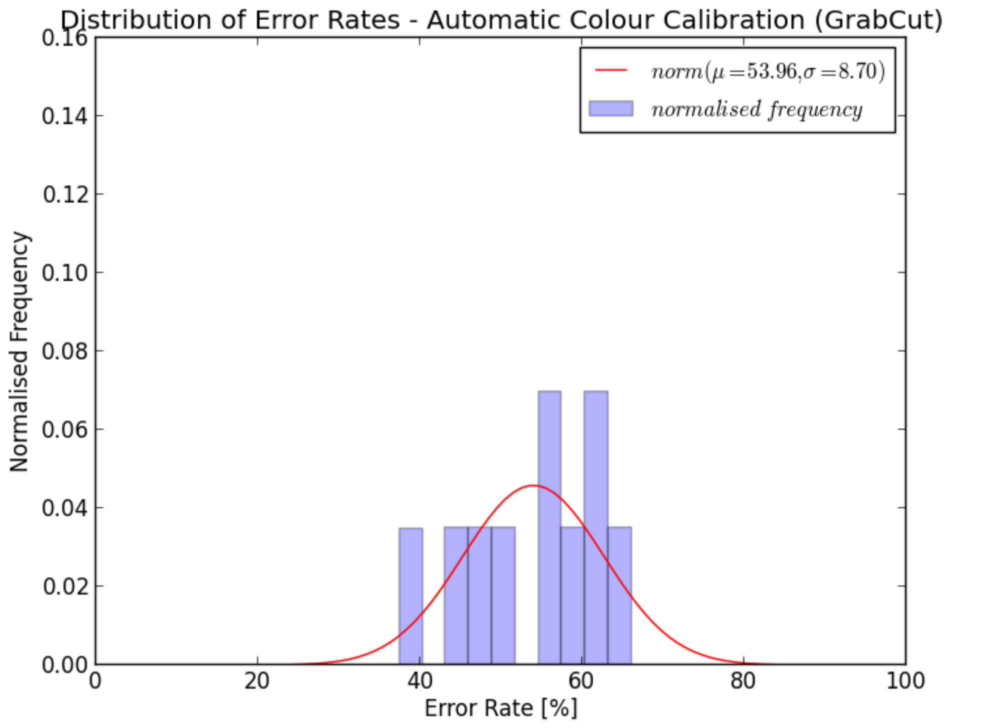


Figure 9.18

9.5.5 Experiment 5

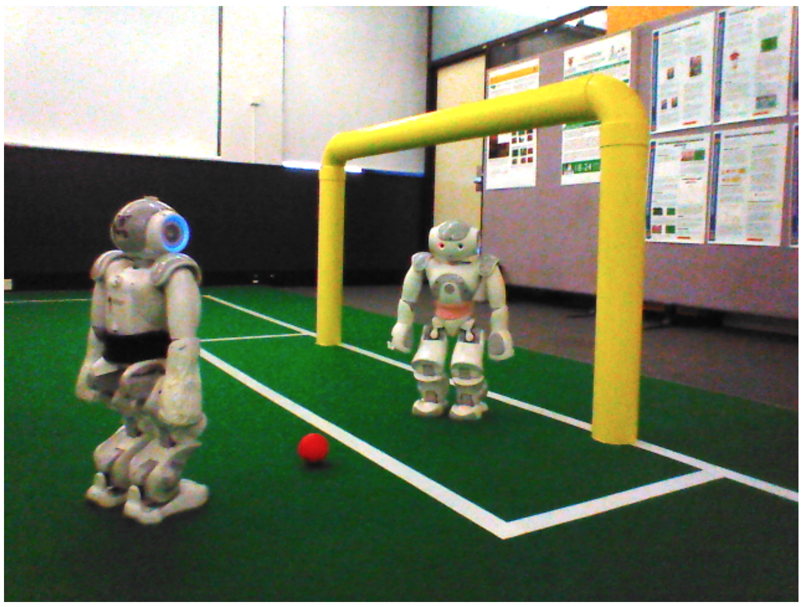


Figure 9.19: Original Raw Image

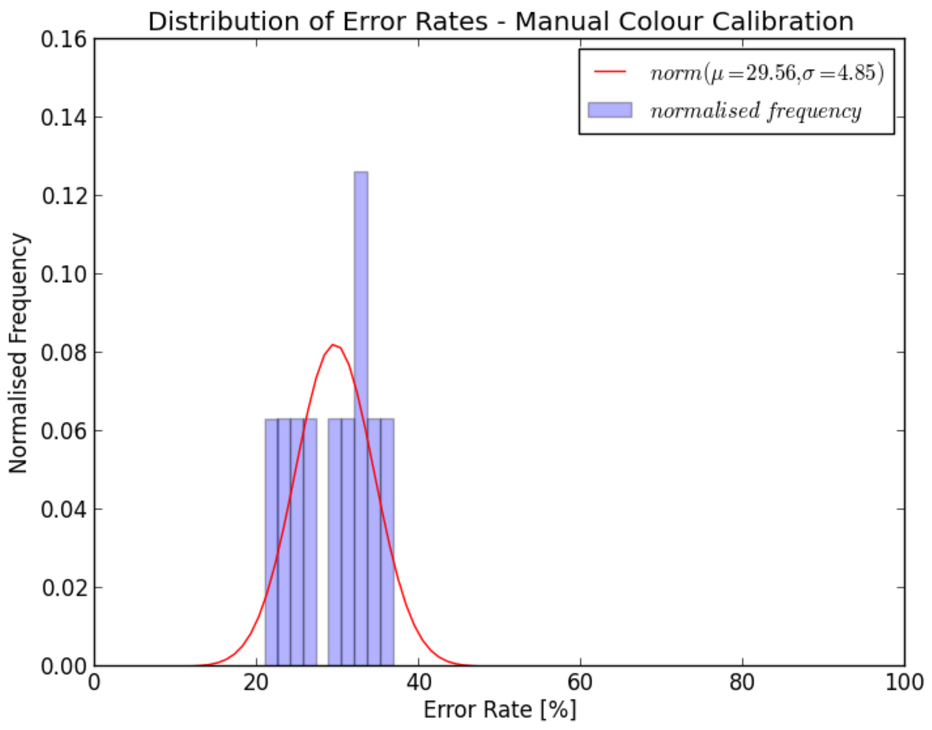


Figure 9.20

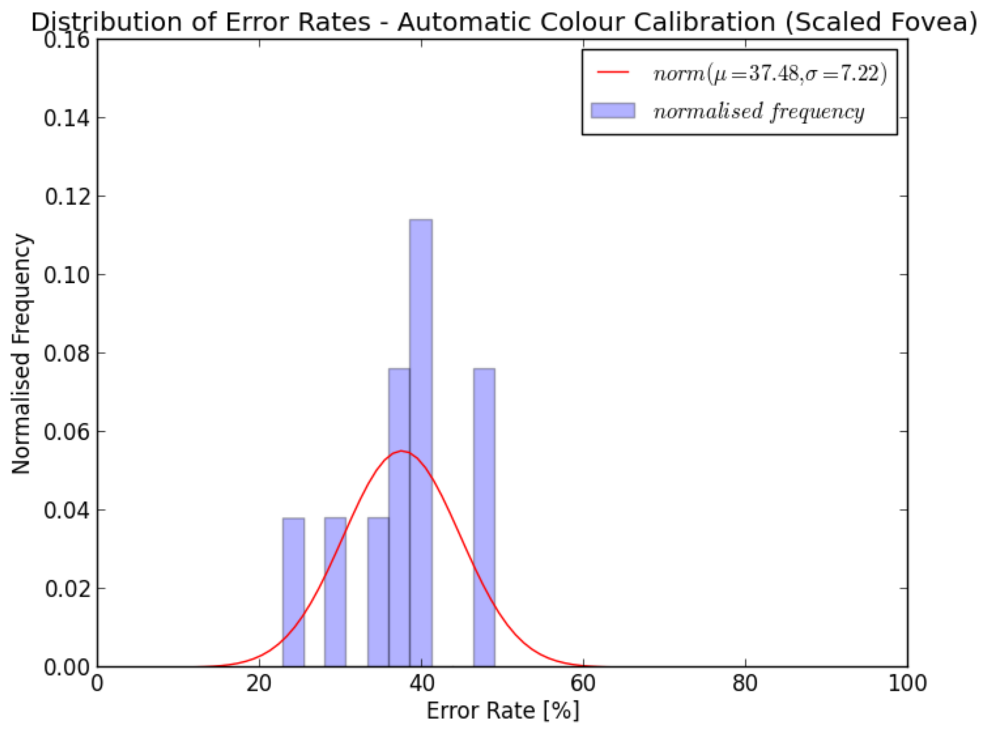


Figure 9.21

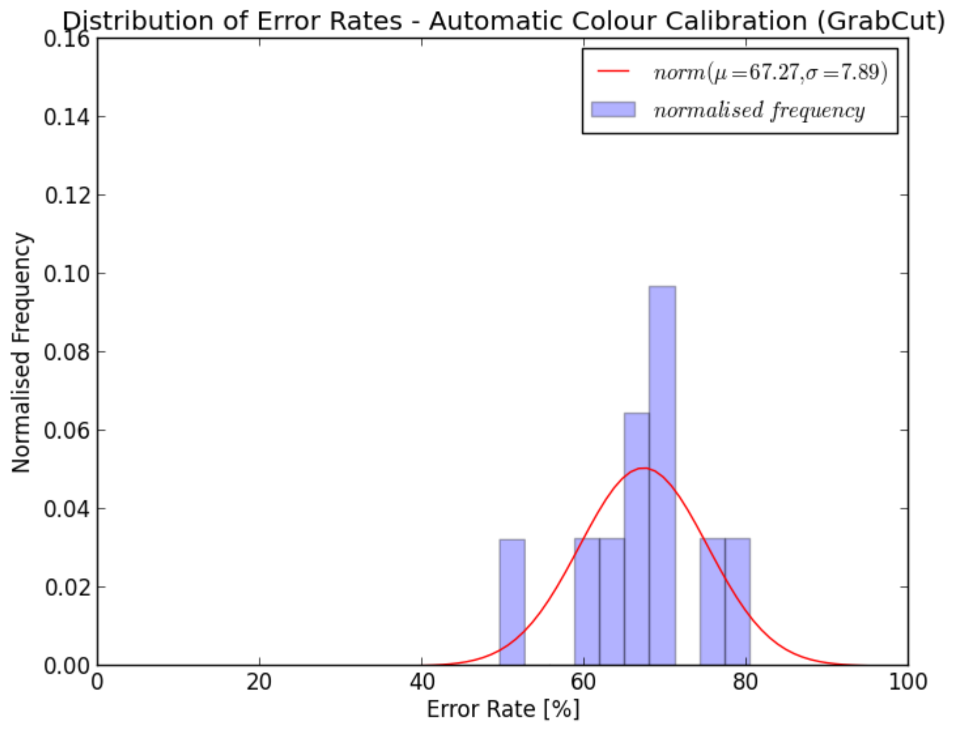


Figure 9.22

9.5.6 Experiment 6

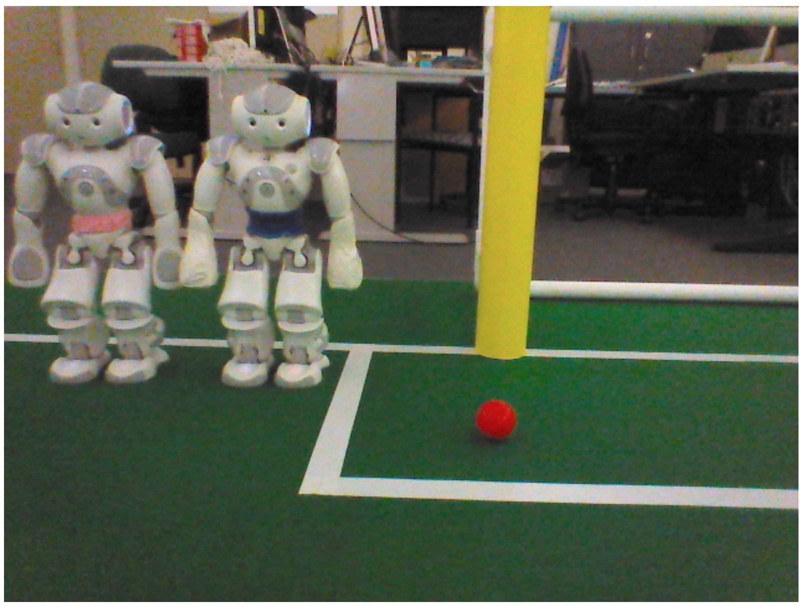


Figure 9.23: Original Raw Image

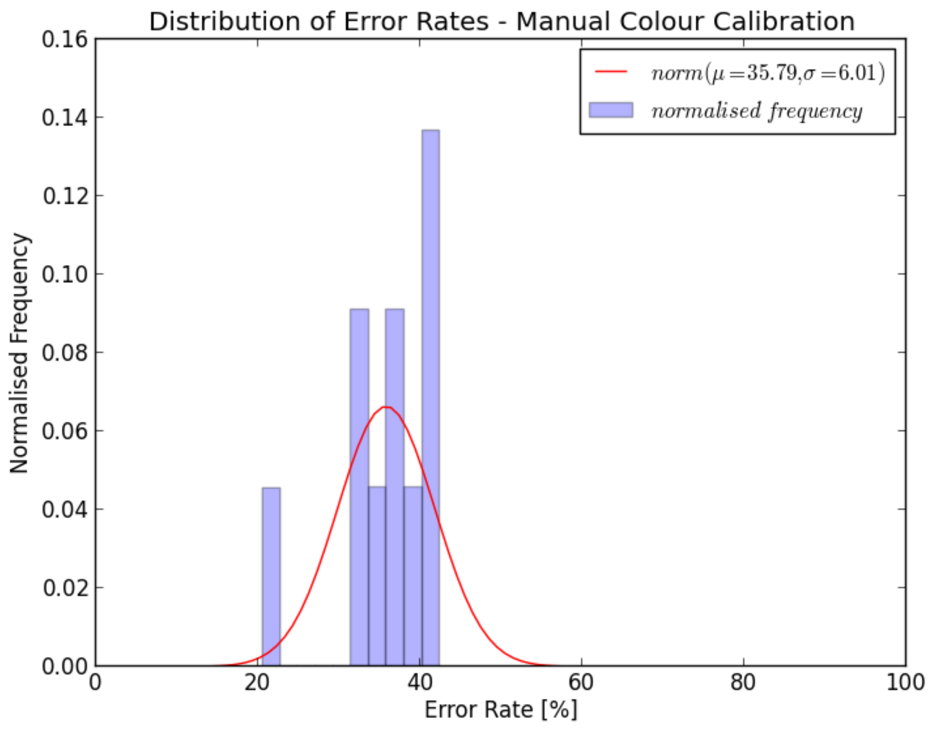


Figure 9.24

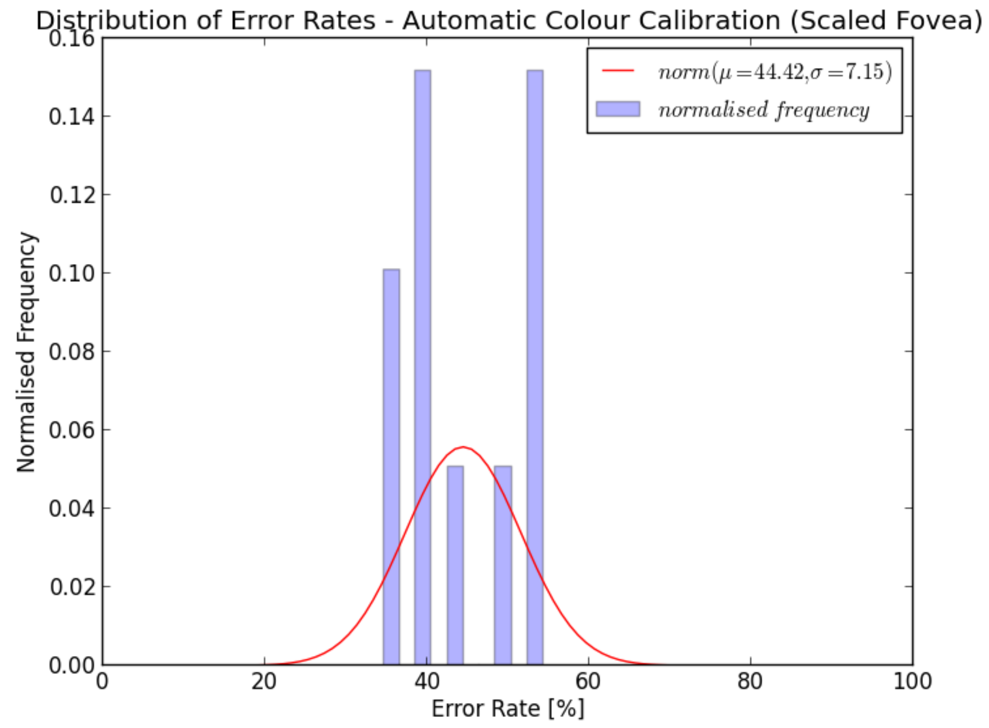


Figure 9.25

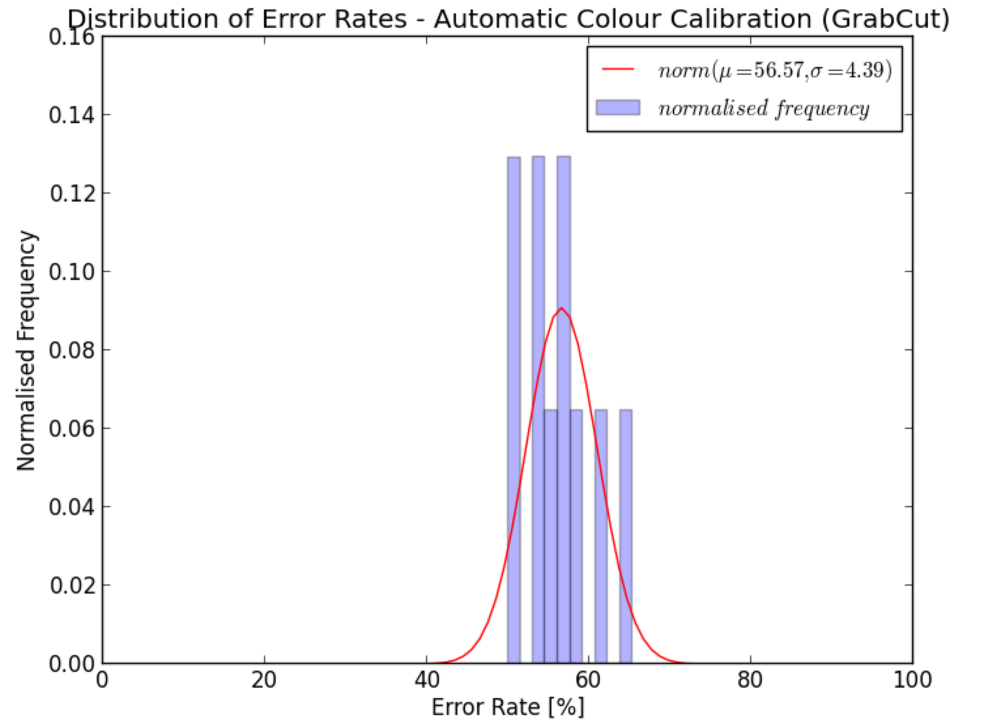


Figure 9.26

9.6 Evaluation

From Table 9.1, it can be observed that the manual and automatic Scaled Fovea methods have statistically similar mean error rates and standard deviations. This suggests that the Scaled Fovea approach is a viable alternative to the manual colour calibration process. On the other hand, the automatic (GrabCut) methodology had significantly higher error rates and larger standard deviations. This makes the GrabCut method a rather unreliable method. A possible explanation of the large spike with the GrabCut method lies in the way the pixels are extracted. Though GrabCut is good at extracting most if not all of the feature pixels, this 'hard segmentation' characteristically leaves a rather jagged surrounding edge. Even with border matting whereby partial transparency is applied to produce a smoother edge, this introduces a lot of other colours labelled with the classified category to the resulting colour lookup table, essentially polluting the data and the likelihood of misclassifying new data.

From the experiments performed:

- Manual performed best 4/6 times
- Automatic Scaled Fovea performed best 2/6 times
- Automatic GrabCut performed the worst in all experiments

In most experiments, the manual and automatic Scaled Fovea produce quite similar results, except in Experiment 2 where the latter performed significantly better due to having more uniform lighting than the rest. The automatic GrabCut algorithm tended to perform significantly worse than the other methods when strong ambient light was present. The data shows that a degradation in performance of the colour calibration occurred with low colour saturation level (Experiment 6) and ambient light (Experiments 3 & 5).

9.7 Discussion

The error rates are largely affected by the colour calibration methods ability to classify the field green. Taking into account that fact that the size of the soccer field has been increased from 6 x 4m to 9 x 6m, the amount of green typically seen in an image will therefore increase substantially. The second largest contributor is the yellow goal post, which can vary largely depending on the robots position on the field. The rest of the colours: orange, red, blue, have a disproportionate representation when determining the error rates and thus the error rates should be used only as an indicator when evaluating the relative performance of the colour calibration techniques.

In this proof-of-concept, the following criteria were addressed successfully:

- **Accuracy** - The automated Scaled Fovea approach was indeed able to colour classify as accurately as the existing manual calibration approach and as such is a viable alternative in future, however the GrabCut approach was deemed an inadequate candidate.
- **Speed** - Both automatic colour calibration methods used very little computational and setup time (using a modern standard laptop/desktop), being able to complete the task on

the order of seconds whereas in comparison the manual colour calibration would require time on the order of minutes.

- **Compatibility** - All calibration methods produced a colour lookup table with the same existing format and therefore the methods can be easily integrated into the rUNSWift vision pipeline.
- **Practicality** - The templating system only requires the user to align the camera with the features and thereafter does not require further intervention. This makes it a rather simple and practical procedure compared to the fine-tuning skills required in manual colour calibration.

Chapter 10

Future Work

10.1 Motivation

This project is a proof-of-concept for the idea of automating the colour calibration process by using templates. However there is still work to be done for this to be a practical utility that can be incorporated into the rUNSWift teams workflow and for use in debugging and competition.

10.2 User-friendly way to create templates

Currently the set of templates used are enough to perform calibration with a given scenario. However in future, if there is a change of the field features, it would be necessary to create new templates. This process can be tedious and unintuitive, and thus there is scope for a GUI utility that would allow the user to describe the bounding boxes with a simple drag-and-drop functionality.

10.3 Template overlays on camera feed

To aid the user to align the robots camera with the template, the template can be overlayed on top of the video stream to provide feedback to perform any adjustments if required.

10.4 Gaussian radii tuning

In the currently rUNSWift codebase, the Gaussian radii are all hard-coded, having been empirically determined to be the most practical given the trade-off between simplicity, colour sensitivity and the amount of manual pixel selection. However this limitation can make it difficult to fine-tune the colour lookup table for different environments. This limitation also applies to the methods presented in this project, since at the present moment they are also empirically determined. It may be a good idea to move the constants to a configuration file and being exposed in the Offnao debugging utility for manual tuning. There may also be further work into optimizing the automatic calibration by running a Hill-Climbing-like algorithm to determine the best parameters for a given environment.

10.5 Skewed Features

In this project the bounding box defined in the template are rectangles. However this may not necessarily be the best shape to describe the approximate location of a feature. For example a ball may perhaps be better defined with a circular bounding ring, or the white field lines may be better defined with a quadrilateral since the cameras perspective will observe the field lines and their intersections at a skewed angle. An investigation into the benefits of doing so while taking into account the additional computational costs and complexity would be an interesting improvement to the current method of describing features with templates.

Chapter 11

Conclusion

The aim of this project was to replace the time and effort of the existing manual colour calibration procedure with an automatic one. A feature templating system was used and two automatic colour calibration techniques assessed: the Scaled Fovea and GrabCut algorithms. The project was successful in demonstrating the Scaled Fovea approach as a viable alternative to the existing manual colour calibration method as it has similar accuracy yet requires significantly less time to perform.

Bibliography

[1] Carl Chatfield. *rUNSWift 2011 Vision System: A Foveated Vision System for Robotic Soccer* (2011). Honours thesis, The University of New South Wales.

[2] Adrian Ratter, Bernhard Hengst, Brad Hall, Brock White, Benjamin Vance, Claude Sammut, David Claridge, Hung Nguyen, Jayen Ashar, Maurice Pagnucco, Stuart Robinson, and Yanjin Zhu. *rUNSWift Team Report 2010 Robocup Standard Platform League* (2010). Only available online: `http://www.cse.unsw.edu.au/~robocup/2010site/reports/report2010.pdf`

[3] Sean Harris. *Efficient Feature Detection Using RANSAC* (2011). Final Year Thesis, The University of New South Wales.

[4] OpenCV Dev Team. *OpenCV 2.4.5.0 documentation: Miscellaneous Image Transformations - grabCut* (2013). Available at `http://docs.opencv.org/modules/imgproc/doc/miscellaneous_transformations.html#grabcut`

[5] Carsten Rother, Vladimir Kolmogorov, Andrew Blake. *'GrabCut' - Interactive Foreground Extraction using Iterated Graph Cuts* (2004). Microsoft Research Cambridge, UK. Available at `http://research.microsoft.com/pubs/67890/siggraph04-grabcut.pdf`

[6] Mohan Sridharan and Peter Stone. *Towards Eliminating Manual Color Calibration at RoboCup* (2004). Available at `http://redwood.cs.ttu.edu/~smohan/Papers/robosym05_autocolorlearn.pdf`

[7] Pablo Guerrero, Javier Ruiz-del-Solar, Josu Fredes, Rodrigo Palma-Amestoy. *Automatic On-Line Color Calibration using Class-Relative Color Spaces* (2008). Available at `http://www.captura.uchile.cl/bitstream/handle/2250/17105/Guerrero_Pablo_Automatic.pdf?sequence=1`

[8] Andreas Frtig, Holger Friedrich, Rudolf Mester. *Robust Pixel Classification for RoboCup* (2010). Available at `http://www.vsi.cs.uni-frankfurt.de/download/Fuertig10ClassificationPreprint.pdf`

[9] Piyush Khandelwal, Matthew Hausknecht, Juhyun Lee, Aibo Tian and Peter Stone. *Vision Calibration and Processing on a Humanoid Soccer Robot* (2010). Available at `http://www.cs.utexas.edu/~pstone/Papers/bib2html-links/HUMANOIDS10-khandelwal.pdf`

[10] Naomi Henderson, Robert King, Stephan K. Chalup. *An Automated Colour Calibration System using Multivariate Gaussian Mixtures to Segment HSI Colour Space* (2008). Available at `http://www.araa.asn.au/acra/acra2008/papers/pap150s1.pdf`

[11] Tatjana Zrimec, Andy Wyatt. *Learning to Recognize Objects - Toward Automatic Calibration of Color Vision for Sony Robots* (2001). Available at `http://www.cse.unsw.edu.au/~icml2002/workshops/MLCV02/MLCV02-TZWayt.pdf`

Appendix 1 - Test Images