

Digital Image Processing In Robot Vision

Project by Naomi Henderson

Supervised by Professor Rick Middleton

*A thesis submitted in partial fulfilment of the requirements for the degree of Bachelor of
Engineering in Computer Engineering at The University of Newcastle, Australia.*

Abstract

This report documents the modifications made to the frame processing module of the NUbots vision system for improvement in robot soccer competition and particularly in the variable lighting challenge. The frame processing module involves the classification of images with respect to a look up table (LUT), a new system of secondary colour classification for blue robots that uses edge detection, blob formation and blob filtering.

Soft colour classification was used as the basis of improvement in frame processing. This involved several specific modifications to the previous system. Additional *soft colours* were integrated into colour classification to assist the differentiation between field objects of similar shades. A secondary classification method that uses edge detection was implemented to account for the unique edge-colour characteristics of blue uniformed robots. This method allows the classification of blue robots; this has not been achieved effectively using standard classification methods. Modifications to the blob formation method were required to account for the soft colour classification system. The modified blobs were then filtered to extract useful data, which is interpreted by object recognition.

The LUT system used in the classification of images was modified to increase accuracy and robustness of object data. The modification involved the implementation of a 7*7*7 bit LUT system. Another improvement of the system involved the modification and implementation of a Support Vector Machine (SVM) to partially automate and hence optimise LUT generation.

The modifications to vision have improved accuracy in object data and robustness in colour classification. Additionally, these modifications are effective under wider range of lighting conditions than previously.

List of Contributions

- Implement a Soft Colour Classification System into the NUBot Vision System, which involves implementing additional, conditional colours
- Implement a 7*7*7bit Look up Table (LUT) System into the Vision System
- Implement a secondary colour classification system that uses edge detection to classify blue robots
- LUT files for use in colour classification for test situations and competition.
- Modify blob formation to account for soft colours and 7*7*7bit LUT
- Modify a Support Vector Machine (SVM) application and implement into the NUBot LUT system
- Develop a solution for the Variable Lighting Challenge of RoboCup 2005

Naomi Henderson

*Professor Rick Middleton
Supervisor*

Acknowledgements

I would like to thank Rick for all his help and for giving me the wonderful opportunity to work on this project. Also, Thanks to the members of the NUbots team; Michael, Stephen, Steven, Kenny, Phavanna, Tim and Robin who have made my project so enjoyable, thank you all for your help, support and friendship. Finally a special thank you to my family, who have been so supportive and so patient with me this year.

Contents

1	Background	1
1.1	Introduction	1
1.2	Sony AIBO ERS-7	1
1.3	RoboCup.....	2
1.3.1	Aims	2
1.3.2	Four Legged League.....	2
1.3.3	Variable Lighting Challenge.....	3
1.4	The NUBots (Newcastle University Robots)	3
1.4.1	NUbot Software Process.....	3
1.4.2	Vision Processing System.....	4
2	Input Image	5
2.1	Discussion	5
2.2	Image Quality.....	5
2.2.1	Motion	5
2.2.2	Camera settings	5
2.3	Field Lighting and Effect on Images	9
2.3.1	Original Field Lighting	9
2.3.2	Modified Field Lighting	10
3	Process Frame	11
3.1	Process Frame Software Overview.....	11
3.2	Colour Classification.....	11
3.2.1	Under-classification “Misses”	13
3.2.2	Over-classification “False-Positives”	13
3.2.3	Variable Lighting	14
3.2.4	Soft Colour Classification.....	14
3.2.5	Classified Colours	14
3.3	Secondary Classification using Edge Detection	17
3.3.1	Edge Detection	17
3.3.2	Colour Classification using Edge Detection	18
3.3.3	Robot Blue Classification Algorithm	19
3.4	Blob Formation	21
3.4.1	Required Information from Images.....	21
3.4.2	Blob Formation	22
3.4.3	Blob Formation Logic	22
3.4.4	Blob Formation Rewrite	23
3.5	Soft Colour Filtering.....	26
3.5.1	Soft Colour Principal.....	26
3.5.2	Soft Colour and Blobs Formed	26
3.5.3	Object Characteristics.....	27

3.5.4	Soft Colour Blob Filtering and Blob Size Modification Software...	37
4	<i>Look up Table (LUT) Generation</i>	41
4.1	Manual Generation	41
4.2	Discussion	41
4.3	EOTN	41
4.3.1	Input Image	42
4.3.2	Classification Tools.....	42
4.3.3	Classified Image.....	43
4.3.4	Blobs Formed.....	43
4.3.5	Object Recognition.....	43
4.4	7 Bit LUT	44
4.5	LUT Automation	44
4.5.1	Background.....	44
4.5.2	SVM Modification and Application.....	45
5	<i>Variable Lighting Challenge</i>	48
5.1	Soft Colour Classification under Varying Lighting Conditions	48
5.1.1	Ball	48
5.1.2	Red Robots and Ball.....	49
5.1.3	Yellow Goal	49
5.2	Variable Shutter Speed	50
5.2.1	Image Testing.....	51
5.3	Variable Lighting Challenge Behaviour Routine	52
	<i>Conclusion</i>	53
	<i>References</i>	54

1 Background

1.1 Introduction

This report documents the modifications made to digital image processing of a AIBO ERS-7 vision system for use in robotic soccer. This section offers background information on the hardware used, robot soccer and the robot software system.

1.2 Sony AIBO ERS-7

The project involves interpretation of data from, and programming of Sony AIBO ERS-7 Hardware. The robots have a 576 MHz 64bit RISC CPU and are programmed using a Sony Memory Stick with code written in C++ and Python. The primary role of this project is the interpretation of images from the 350K-pixel image sensor; also the project involves data interpretation and calibration of the distance sensors.

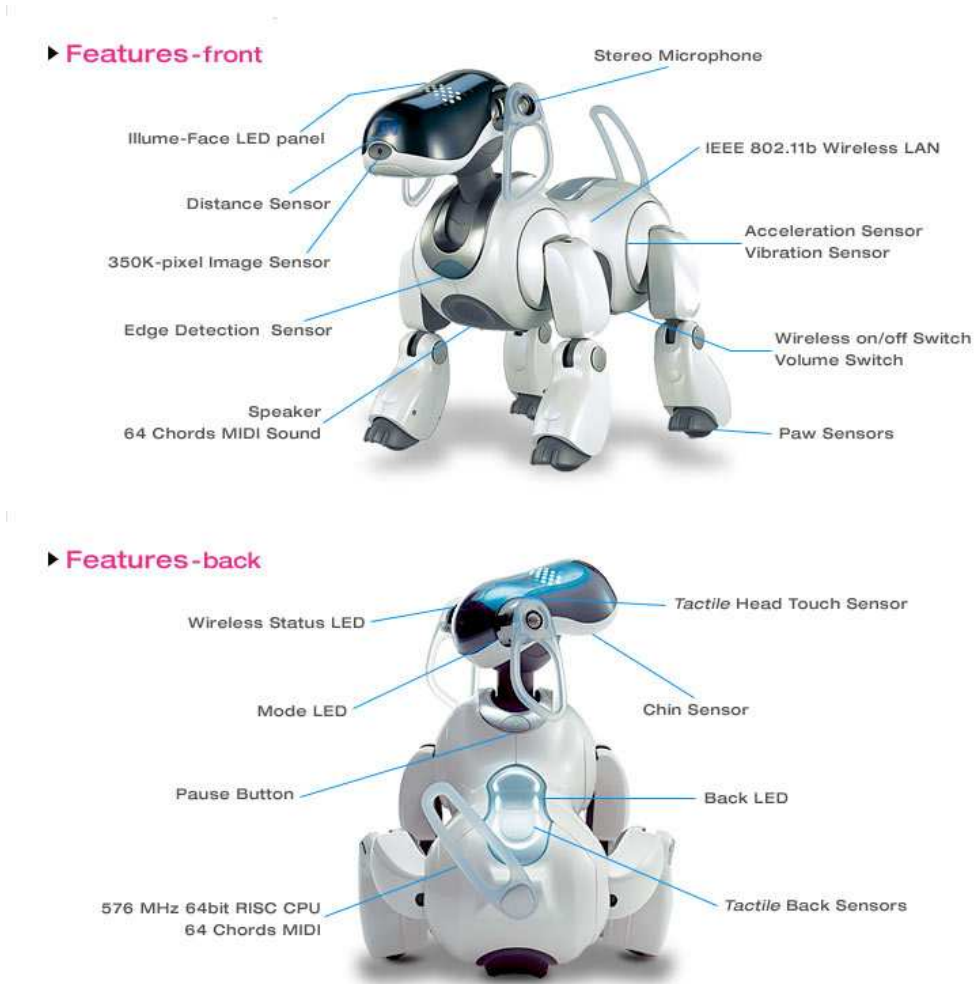


Figure 1-1 Features of AIBO ERS-7

Additional information is available from the Sony AIBO web site [1]

1.3 RoboCup

1.3.1 Aims

RoboCup is an initiative designed to promote research internationally into robotics and artificial intelligence. It uses the game of soccer as the base to apply the complex skills and intelligence that humans take for granted. Skills which include vision processing, behaviour, artificial intelligence, locomotion and localisation are applied to game situations in simulation, small size, medium size, humanoid and 4-legged robots. The ultimate aim of RoboCup is that:

“By 2050, develop a team of fully autonomous humanoid robots that can win against the human world champion team in soccer.” [2]

This year RoboCup is held in Osaka, Japan

1.3.2 Four Legged League

The RoboCup Four Legged League uses Sony AIBO robots as competitors in a fully autonomous game situation. The twenty minute competition involves a game of robot soccer where a ‘red team’ competes against a ‘blue team’ to try and score as many goals as possible in the opponent defender goal. There are rules as to offside position of robots, the out of bounds position of the ball and holding time of the ball. The field set up is colour coded (see Figure 1-2) and must be correctly interpreted in vision for success in the competition.

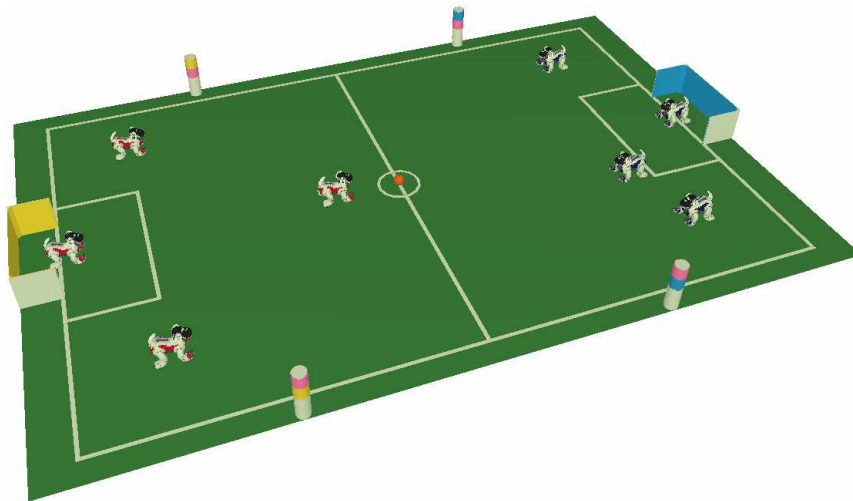


Figure 1-2 Field arrangement as stated in the RoboCup2005 Competition Rules [3]

1.3.3 Variable Lighting Challenge

There are a series of challenges in addition to the soccer games in RoboCup competition. The challenges are an initiative of RoboCup to increase research into general areas, by having an open challenge and specialised areas of localisation and vision under variable lighting conditions.

Currently most vision systems are venerable to a change in lighting conditions. The Variable Lighting Challenge is intended to increase robustness in vision by functioning under conditions when illumination is non-uniform. A blue robot must score as many yellow goals as possible in three minutes whilst avoiding two stationary red robots. This is to be done under random lighting conditions that include constant lighting, uneven lighting, slow lighting changes and rapid changes. [4]

It is an aim of the project to increase performance in this year's variable lighting challenge.

1.4 The NUbots (Newcastle University Robots)

The NUbots are The University of Newcastle's robotics team that have annually competed in RoboCup since 2002. Their record of performance is high as they have run 3rd in the world each year of competition. For additional information including previous team reports, refer to The University of Newcastle's Robotics Laboratory website [5]

1.4.1 NUbots Software Process

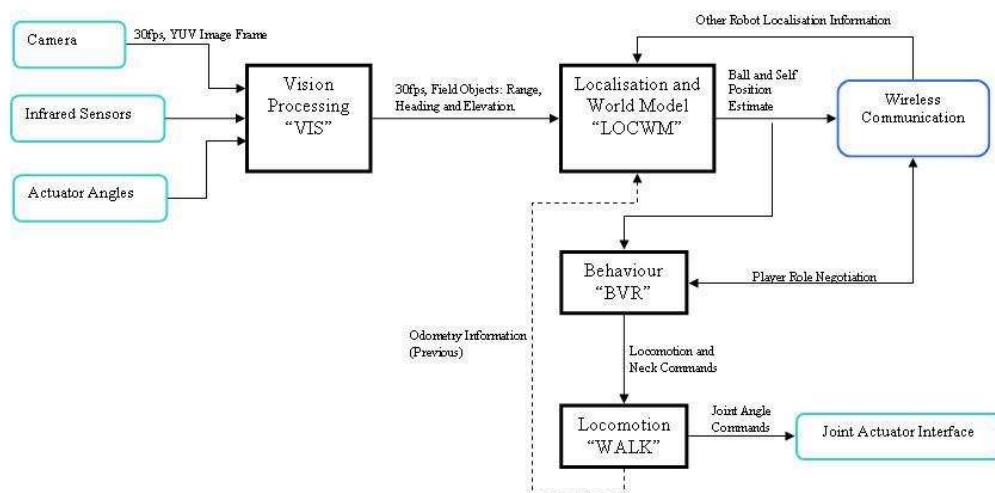


Figure 1-3 2005 NUbots Software Architecture, modified from 2004 NUbots Team Report [6]

Figure 1-3 shows the NUBot software architecture of 2005, which inputs images at 30 frames per second along with the sensor values of Figure 1-2 into the vision system. Images and sensor values are processed to interpret objects that are then sent to Localisation and World Model (LOCWM). LOCWM uses object information along with other robots localisation information (via wireless communication) to determine the robots own position and ball location on the field. Depending on localisation data of all robots, a behaviour decision is made that requires commands to be sent to locomotion and hence joint angle commands sent to joint actuators.

The project focuses on the vision system, a detailed breakdown of the vision system architecture follows.

1.4.2 Vision Processing System

A 208*160 pixel YUV bitmap image is input to the vision processing system thirty times per second. The images are interpreted for object information in the Process Frame module and data is then sent to Object Recognition, which determines what data is reliable object information. The project involves the modification of the Process Frame module to ensure that reliable object information is sent to Object Recognition.

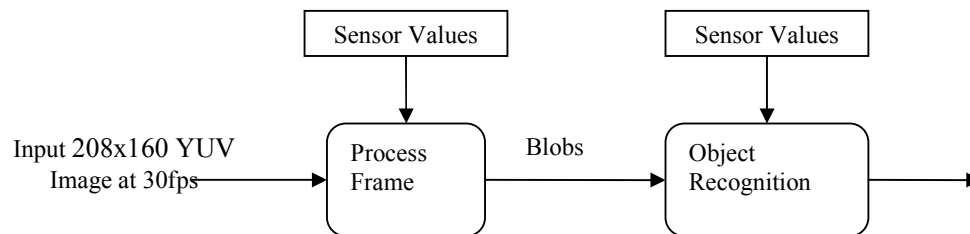


Figure 1-4 Vision Processing Software Architecture.

2 Input Image

2.1 Discussion

The AIBO's camera is located on the robots 'nose' as shown in Figure 2-1. It inputs the YUV bitmap image for classifying and interpretation. The quality of the image is dependent on the robots motion, camera settings and lighting conditions surrounding the image.

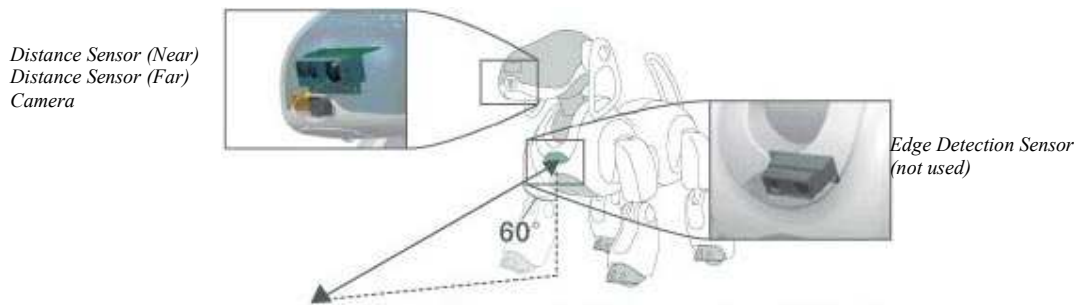


Figure 2-1 Position of the camera that inputs images from the AIBO ERS-7 [2].

2.2 Image Quality

2.2.1 Motion

The image tends to blur during fast head motion. Using a slow shutter speed also increases the occurrence of blurring. Figure 2-2 shows images of blurring.

2.2.2 Camera settings

The image input to the system is a 208*160 YUV bitmap image with three additional channels for edge detection. The camera settings influence the quality of the image. Modifiable settings are:

Shutter speed: This is set to fast. This minimises blurring but minimises brightness of image; Medium to slow may be implemented in variable lighting for increased brightness however slower head pan is required. The effect on images is seen in Figure 2-3.

Camera Mode: Fluro, outdoor, indoor. Fluro is appropriate as the other settings have a blue impact on the images as seen in Figure 2-4.

Gain: The White Balance can be set to high, medium or low. The gain is set to high as it has the best effect on images as seen in Figure 2-5.

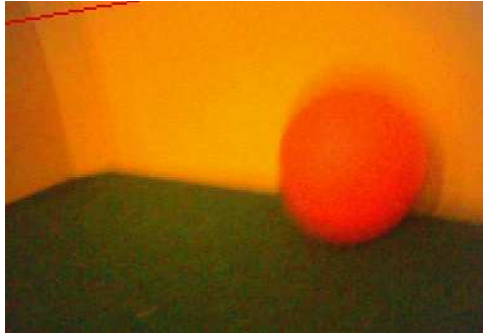


Figure 2-2 Effect of blurring



Figure 2-3 Shutter speed: fast, medium and slow



Figure 2-4 Camera modes: outdoor has a definite blue hue; indoor has a subtle blue tinge.

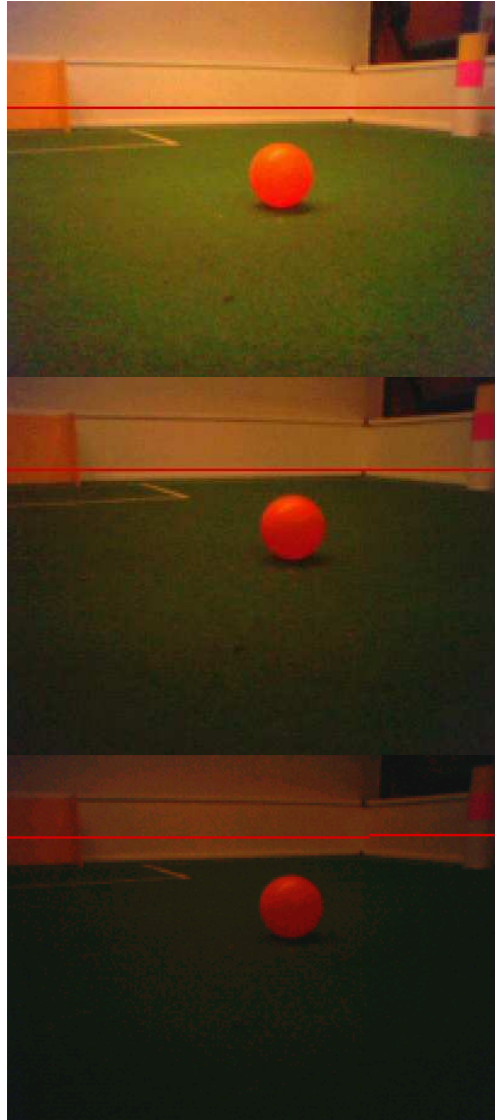


Figure 2-5 Gain settings: high, medium and low

Ring

There is distortion in the brightness of the image proportional to the distance from the centre. This creates a circular pattern of darkness around the edge of the image as shown in Figure 2-6

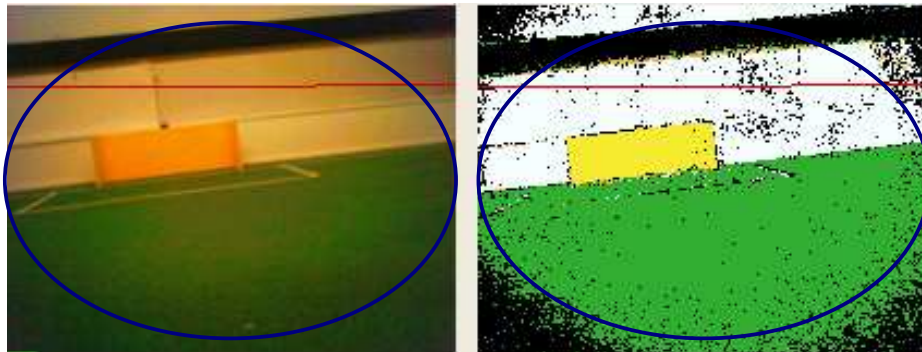


Figure 2-6 Distortion of image in a ring pattern.

2.3 Field Lighting and Effect on Images

RoboCup rules state that the field is lit to approx 1000 LUX. Lighting dramatically effects image quality and field lighting was modified to create a more even lighting spread.

2.3.1 Original Field Lighting

The original lighting of the field was lit with six 300W lights with the LUX values across the field plotted in Figure 2-7.

It can be seen from the graph that the lighting levels were quite dark, particularly in the shadows of the goals, with some areas below the recommended minimum camera operating brightness level of 650 LUX. The effect of the lower lighting can be seen in images from the robot shown in Figure 2-8.

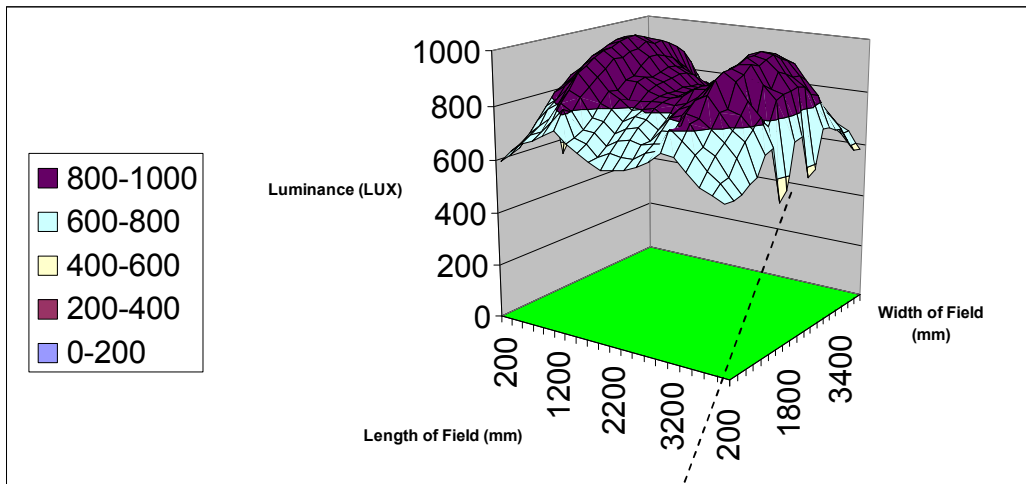


Figure 2-7 Graph of original field lighting



Figure 2-8 Effect of low lighting on images, large area of shadow cast on the yellow goal.

2.3.2 Modified Field Lighting

The field lighting was modified to use ten 300W lights positioned in a way as to evenly light the field and ensure that critical objects (goals and beacons) were effectively lit to approx 1000 LUX. The effect of this modification can be seen in the improvement of images in Figure 2-9.



Figure 2-9 Previous dark image of goal vs. image of yellow goal under improved lighting conditions.

3 Process Frame

3.1 Process Frame Software Overview

The project involved the modification of the Process Frame component of the Vision Processing System that can be broken down into the following modules:

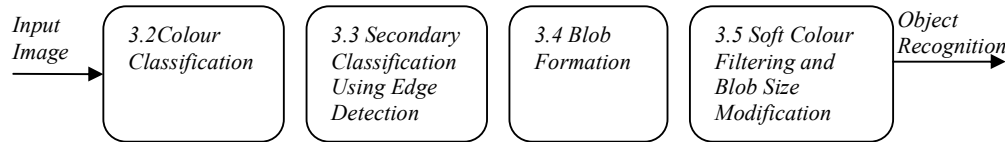


Figure 3-1 Process Frame software flow chart of the Vision System

The Colour Classification and Blob Formation processes were originally used to interpret images and send information to Object Recognition; however they have been modified in this project. Secondary Classification Using Edge Detection and Soft Colour Filtering and Blob Size Modification are entirely new vision concepts that had to be designed, tested and integrated into the Vision System. Each module is explained in detail in this Section.

3.2 Colour Classification

Colour classification is used to interpret each pixel of the 208x160 YUV bitmap image in the image stream. Each pixel of the frame has six 8bit components; three YUV components and three edge detection components. In the colour classification process, only the three YUV components are used. Originally each YUV component was made to be 6 bits by cropping the last 2 bits but was modified to be 7 bits by only cropping 1bit. The three cropped YUV components are used to reference a 3-Dimensional Look Up Table (LUT). The LUT stores an 8bit colour classified value for every possible YUV value.

The LUT and classification software was modified incorporate a 7*7*7 bit LUT system. This was to increase accuracy in colour classification. This is a design decision made due to the increase in colour options with the introduction of soft colour classification (3.2.4). The generation and implementation of the 7*7*7 bit LUT is detailed in 4.4. The process of classifying an input image is detailed below.

$$\text{Input Image} = \begin{bmatrix} x_0 & x_1 & \dots & x_j \end{bmatrix} * \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_k \end{bmatrix} = \begin{bmatrix} x_0 y_0 & x_1 y_0 & \dots & x_j y_0 \\ x_0 y_1 & x_1 y_1 & \dots & x_j y_1 \\ \vdots & \vdots & & \vdots \\ x_0 y_k & x_1 y_k & \dots & x_j y_k \end{bmatrix}$$

$$\text{Classified Image} = \begin{bmatrix} c_{x_0 y_0} & c_{x_1 y_0} & \dots & c_{x_j y_0} \\ c_{x_0 y_1} & c_{x_1 y_1} & \dots & c_{x_j y_1} \\ \vdots & \vdots & & \vdots \\ c_{x_0 y_k} & c_{x_1 y_k} & \dots & c_{x_j y_k} \end{bmatrix}$$

Where X = image width in pixels = 208

Y = image height in pixels = 160

$j = X - 1$

$k = Y - 1$

Each $c_{x_m y_n}$ value is an 8 bit number

Each $x_m y_n$ value corresponds to an 8*8*8 bit YUV value defined by:

$$x_m y_n = YUV_{8bit} = \begin{bmatrix} Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 & Y_0 \\ U_7 & U_6 & U_5 & U_4 & U_3 & U_2 & U_1 & U_0 \\ V_7 & V_6 & V_5 & V_4 & V_3 & V_2 & V_1 & V_0 \end{bmatrix} *$$

Each 8 bit number is then cropped to 7 bits.

$$x_m y_n = YUV_{7bit} = \begin{bmatrix} Y_7 & Y_6 & Y_5 & Y_4 & Y_3 & Y_2 & Y_1 \\ U_7 & U_6 & U_5 & U_4 & U_3 & U_2 & U_1 \\ V_7 & V_6 & V_5 & V_4 & V_3 & V_2 & V_1 \end{bmatrix} *$$

Each 7*7*7 bit value can then be mapped to the 7*7*7 bit LUT

$$LUT = \begin{bmatrix} L_{Y_0} & L_{Y_1} & L_{Y_2} & L_{Y_3} & L_{Y_4} & L_{Y_5} & L_{Y_6} \\ L_{U_0} & L_{U_1} & L_{U_2} & L_{U_3} & L_{U_4} & L_{U_5} & L_{U_6} \\ L_{V_0} & L_{V_1} & L_{V_2} & L_{V_3} & L_{V_4} & L_{V_5} & L_{V_6} \end{bmatrix} *$$

$$= \begin{bmatrix} L_{Y_0 U_0 V_0} & L_{Y_0 U_0 V_1} & \dots & L_{Y_0 U_0 V_6} & L_{Y_0 U_1 V_0} & L_{Y_0 U_1 V_1} & \dots & L_{Y_0 U_6 V_6} & L_{Y_1 U_0 V_0} & L_{Y_1 U_0 V_1} & \dots & L_{Y_6 U_6 V_6} \end{bmatrix}$$

Each LUT value $L_{Y_\alpha U_\beta V_\gamma} = c$, where c is an 8 bit colour classification code. Therefore each classified

image array value for the set of integer values $m = [0, X)$ and $n = [0, Y)$ is

$$c_{x_m y_n} = L_{x_m y_n \rightarrow Y_\alpha U_\beta V_\gamma}$$

3.2.1 Under-classification “Misses”

Under-classification or a *miss* occurs when not enough pixel values have been classified to the appropriate colour. When this happens blobs of inaccurate size are formed or completely missed due to holes, this leads to complications in distance calculations and hence localisation. See

Figure 3-2

3.2.2 Over-classification “False-Positives”

Over-classification or a false positive occurs when too many pixel values are classified for an object. Even if the shade belongs to an object when classifying it, if it is common with other objects and interferes with their blob formation then the image is over-classified, see Figure 3-3. Over-classification is dangerous as the probability of an error in object recognition increases with the number of blobs that are sent for processing increases.

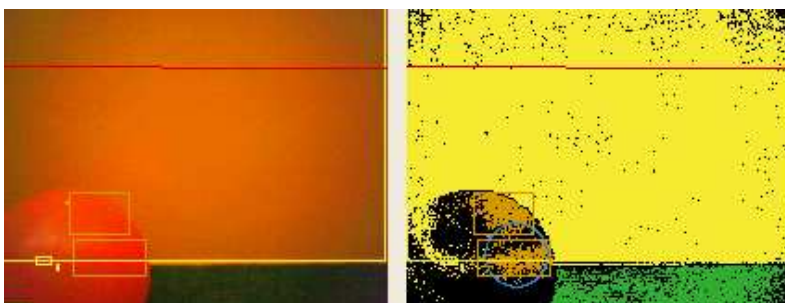


Figure 3-2 An example of under-classification. The size of the ball has not been detected properly (missed) due to not enough classified values.

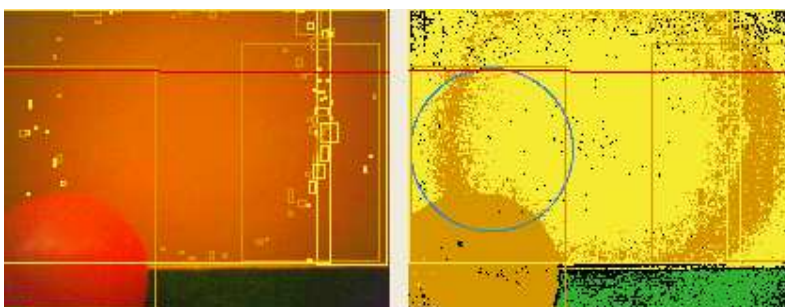


Figure 3-3 An example of over-classification. The entire ball values have been classified however shared shades with the yellow goal have also been classified. The size of the ball has not been detected properly (creating a false positive).

3.2.3 Variable Lighting

Variable lighting cannot be accounted for in basic classification as a critical shade classified at multiple brightness leads to over-classification. Classification in one shade maps to an error in classification at a different lighting level/ brightness. Hence, a change in lighting conditions requires a new LUT to account for the new shades of critical colours.

3.2.4 Soft Colour Classification

Soft colour classification is the classification of not only object-critical colours but additional ‘in between’ colours these colours are not definite like basic colour classification, rather they are conditional relative to basic colours and considered after blob formation. Soft colour classification acts as a buffer between basic colours and allows for 3-Dimensional objects shading properties to be accounted for under varying lighting conditions and at different distances away from the camera. This is implemented by classifying the shades of basic colours that overlap as conditional ‘soft’ colours.

Soft colour classification is an entirely new colour classification system in the NUBot vision system that was implemented to solve critical errors in vision; these issues are discussed in detail in 3.2.4. Soft colours are **conditional** that is they are conditioned by initial sanity checks in soft colour filtering process but must not be confused with **object recognition sanity checking**. By thoroughly studying the colour shading characteristics of objects a set of conditions was derived for each object/ soft colour relation, this is detailed in 3.5.3.

3.2.5 Classified Colours

The basic colours of objects and their colour names are shown below..

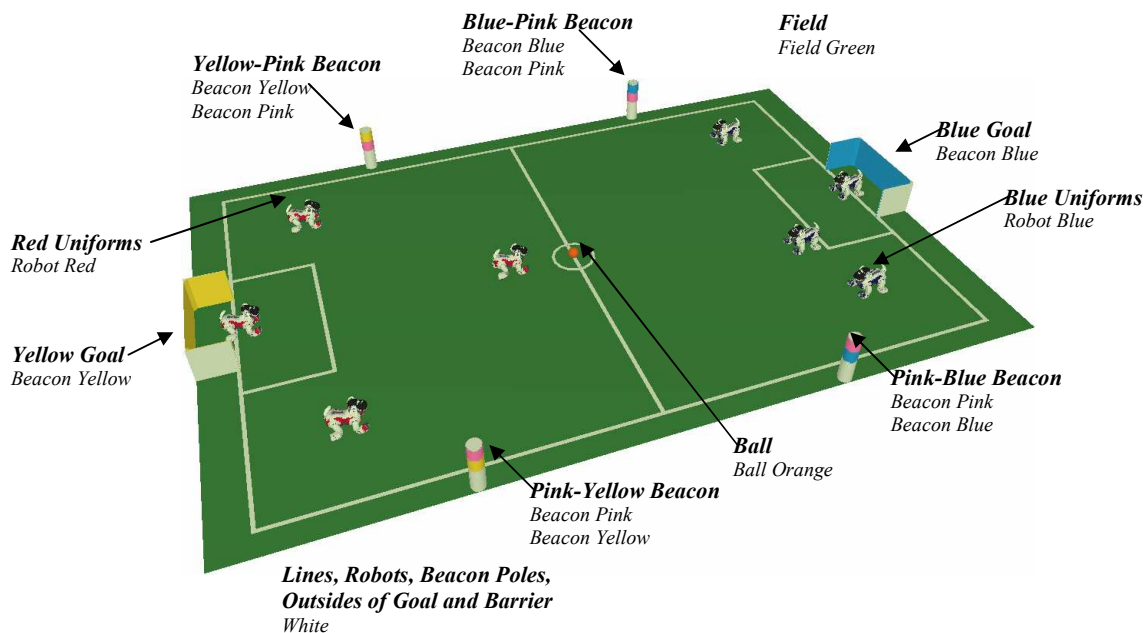


Figure 3-4 The basic colours that correspond to field objects

3.2.5.1 Basic Colours

Pixels are classified according to which object they correspond to. The basic colours are as follows:

White is the colour of field lines, beacon poles, robot bodies and barrier walls.

Field Green is the colour of the field. In circumstances where dark shades of green have been classified as field green the colour is spread throughout dark objects particularly the image background.

Ball Orange-The Ball has a distinct orange colour whilst in good lighting; however when light is reflected or a shadow cast it shares shades of colour with yellow and red respectively. It is these shared shades that are the most commonly spread throughout the image as noise.

Beacon Yellow is the colour of a goal and the yellow half beacons.

Beacon Blue is the colour of a goal and the blue half beacons

Beacon Pink is the second colour on all beacons the height indicates the left or right hand side of the field.

Robot Red is the colour of the red teams' uniform. Under certain lighting red can mistakenly be classified as orange with catastrophic effects in game.

Robot Blue is the colour of the blue teams' uniform. Classification is very difficult as the values of navy blue are shared with black and dark shades of blue and green. For this reason robot blue is not classified in the initial LUT, rather, classification relies on unique combinational logic of soft colour classification and edge detection in a secondary classification process.

3.2.5.2 Soft Colours

Soft colours were introduced to solve critical errors that affected the vision system such as:

Ball: the ball could not be seen at close distances to the camera with out over classifying orange. Also, the ball could not be accurately detected in close yellow goal images, particularly in corners.

Red Robots: in images, many shades of the red uniform are shared with that of shadow shades of the orange ball, to correctly classify the ball in multiple lighting situations (critical in a game situation) using only *ball orange* results in orange classification in red uniforms. This could lead to a detriment of the team's performance in a game if a red uniformed robot was chased instead of the ball.

Blue Robots: the YUV values for blue uniforms are nearly impossible to classify solely as the shades are shared with dark shades of the blue goal, robot bodies and field green including shadows on the field and dark corners of images.

Goals and Beacons: the entire height and width of the goals and beacons cannot always be fully classified in images using only basic colours with out a lot of excess noise (over-classification) in the classified image.

The soft colours added act as a buffer between colours and may belong to more than one object and are conditional. Additional colours were derived by studying the problem situations described. They are as follows:

Yellow Orange is the overlap of shades between yellow and orange. It is the colour of a shadow cast on the goal (eg goal posts) as well as the light patch on the ball.

Red Orange is the overlap between orange shades of red and darker shades of orange. By classifying dark shades of orange it results in a spread of red orange in dark images, dark objects and throughout the background of an image. Consequently, red orange blobs formed in images are conditional.

Pink Orange is classified as the shades of beacon pink that overlap with orange or red orange and occurs in images of beacons and in shades of the ball.

Shadow Blue is the colour that corresponds to a shadow cast on beacon blue.

The value of c, the colour classified code value is defined to be a number that corresponds to a colour. The values are:

- 0 Unused
- 1 Undefined
- 2 White
- 3 Field Green
- 4 Shadow Green
- 5 Shadow Blue
- 6 Red Orange
- 7 Yellow Orange
- 8 Pink Orange
- 9 Ball Orange
- 10 Robot Red
- 11 Robot Blue
- 12 Beacon Pink
- 13 Beacon Yellow
- 14 Beacon Blue
- 15 Shadow Object

3.3 Secondary Classification using Edge Detection

3.3.1 Edge Detection

In a YUV image the change in brightness between pixels can be detected. This occurs in an image when there is an edge of two contrasting colours. For each pixel of the image there is a corresponding 'edge classified' 8bit value. A value greater than 10 was chosen to be the acceptable range of a definite edge. The size of the value determines the more definite the contrast. Edge Detection has the potential to assist decision making in Colour Classification and Blob Formation.

In the edge-classified image, an edge (values of acceptable range) may be more than one pixel in width or height. When comparing images vertically to that of an edge vertically, it is useful to know if you are within the edge or not, for this reason positive and negative edge was defined.

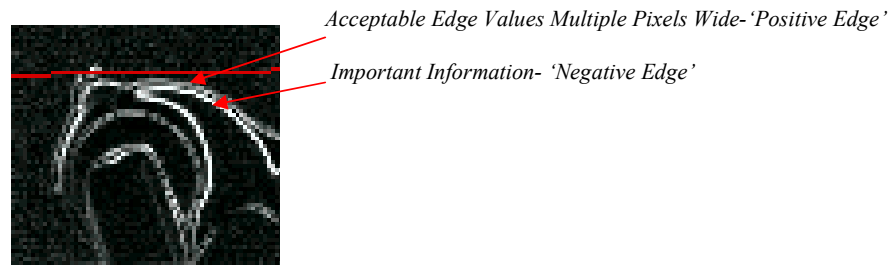


Figure 3-5 Edge-classified image. 'Positive edge' indicates that current pixel is within the acceptable edge value. 'Negative edge' is useful for logic scanning down an image.

3.3.1.1 Positive Edge

Positive-Edge is defined to be the occurrence of an edge whilst scanning vertically down an image. That is, positive edge is set to true when there is a transition from no edge detected to an edge detected and status maintained whilst within acceptable edge values.

3.3.1.2 Negative Edge

Negative-Edge is defined to be the transition from edge detected to no edge found whilst scanning vertically down an image. When negative edge is set to true, positive edge is set to false and vice-versa.

3.3.2 Colour Classification using Edge Detection

Edge Detection is particularly useful for detecting robot blue due to the object/ edge characteristics. The Sony ERS-7 AIBO robots are white, in RoboCup competition the *blue team* wears a navy blue uniform, the contrast of brightness between the two YUV colours is distinct and a definite edge results.

The blue uniforms are initially classified to be *shadow* colour. This is due to the fact that the blue uniform has YUV values close to that of the corners of image, dark corners of the blue goal, object shadows on the field and the dark shades of every robot. However, all of these objects do not have the same edge-colour characteristics. After initial Colour Classification is performed, Secondary Classification is carried out to classify robot blue.

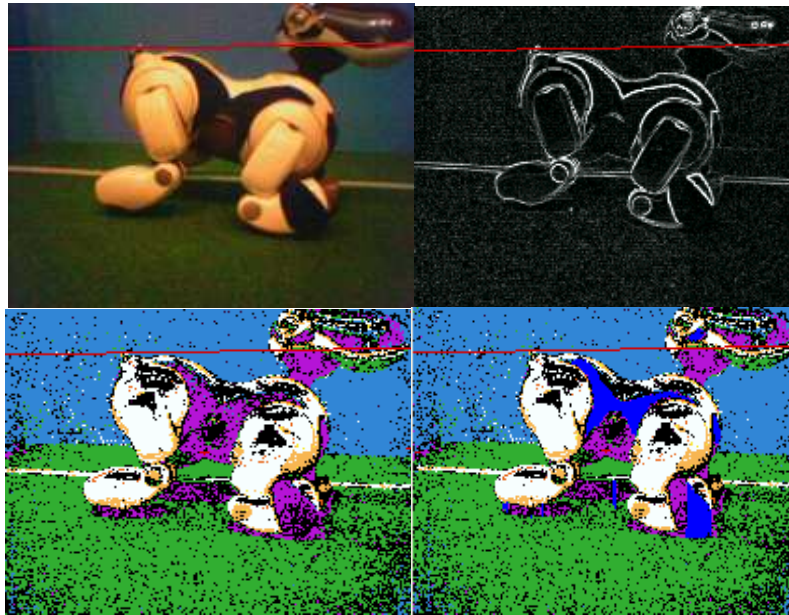


Figure 3-6 YUV image, edge-classified image, colour classified image, secondary classified image of blue robots.

3.3.3 Robot Blue Classification Algorithm

The general edge-colour characteristics were derived from studying images similar to that of Figure 3-6 YUV image, edge-classified image, colour classified image, secondary classified image of blue robots.. There is always a definite edge between the white robots and the blue uniforms hence this is the correlation that is to be searched for.

For the width of the image, the image is scanned downwards. If *white* is found a negative edge is searched for, if a negative edge is found the process then checks what colours follow white. If a run of *shadow* colour is found then the pixels are updated to be *robot blue* until a positive edge is found. However, if a run of another colour occurs then it will break the check. If a run of another colour is found after white is found it will not change pixels to robot blue, if the run is found after a run of shadow colour then it will break the pixel modification. Figure 3-7 explains the Robot Blue Classification logic. Note: an edge between blue and green is not recognised as there is a slow transition of similar shades similar shades.

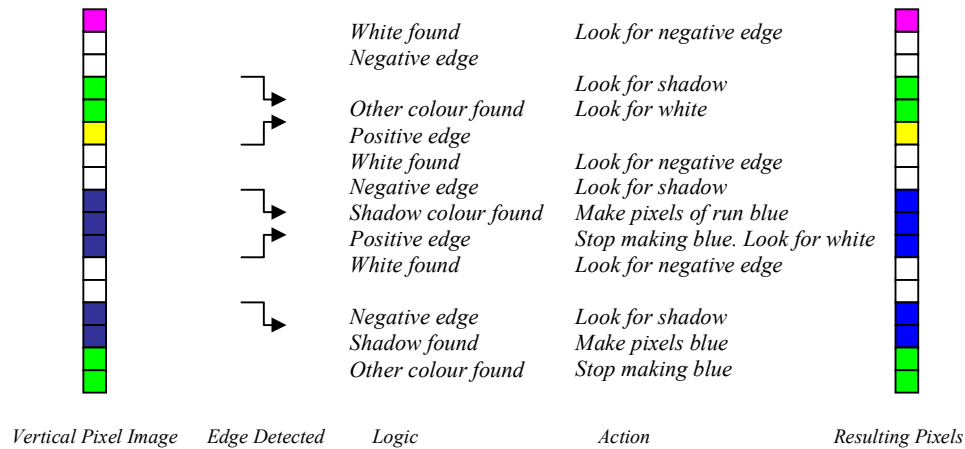


Figure 3-7 Vertical Pixel Image, Edge Detected Colour Runs and Logic Result

Software Logic

- 1 For width, scan column
- 2 For each pixel of height check for
 - 2.1 Edge?
 - 2.1.1 Yes
 - 2.1.1.1 If no edge found previously and edge found set positive edge to true and negative edge to false. Set Make Blue to false
 - 2.1.2 No
 - 2.1.2.1 If positive edge set previously and no edge found on current
 - 2.1.3 White?
 - 2.1.3.1 Yes
 - 2.1.3.1.1 Then look for edge- set check edge
 - 2.1.4 Check edge and negative edge
 - 2.1.4.1 Check Colour
 - 2.1.5 Colour == Shadow
 - 2.1.5.1 Set make blue =true
 - 2.1.6 Colour == OTHER
 - 2.1.6.1 Increment odd colour count
 - 2.1.6.2 If odd colour count ==max run length permitted
 - 2.1.6.2.1 Make blue =false
 - 2.1.7 If Make Blue = true
 - 2.1.8 Make pixels robot blue

3.4 Blob Formation

3.4.1 Required Information from Images

Grouping of classified colours is used for the transfer of information from an image to object recognition. The process of creating these groups is called 'blob formation'. When forming blobs the following information is required.

<i>colour</i>	<i>The colour of the associated pixels, used to determine which object the blob corresponds to</i>
<i>minx, maxx,</i>	<i>The minimum and maximum x coordinates of the associated pixels</i>
<i>miny, maxy,</i>	<i>The minimum and maximum y coordinates of the associated pixels</i>
<i>width</i>	<i>The width of the blob calculated as $(maxx - minx + 1)$</i>
<i>height</i>	<i>The height of the blob calculated as $(maxy - miny + 1)$</i>
<i>area</i>	<i>The area of entire blob calculated as $width * height$</i>
<i>correctPixels</i>	<i>The true area, the number of associated colour pixels in the blob area.</i>
<i>ignoreBlob-</i>	<i>A boolean flag used to indicate if the blob is to be used or ignored due to failed conditions</i>
<i>minyx, maxyx,</i>	<i>The minimum and maximum x values along the minimum and maximum y coordinates respectively</i>
<i>minxy, maxxy,</i>	<i>The minimum and maximum y values along the minimum and maximum x coordinates respectively</i>
<i>blobNumber</i>	<i>A count of how many blobs have been formed</i>

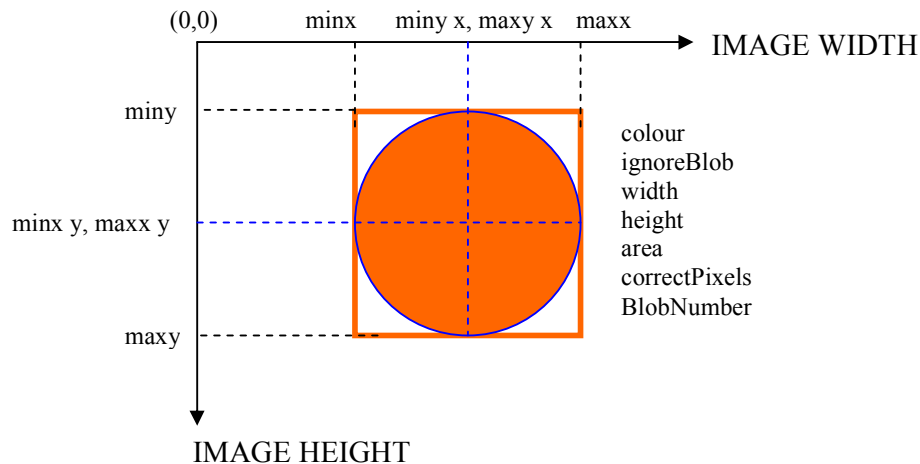


Figure 3-8 Diagram of Blob parameters

3.4.2 Blob Formation

Blobs are formed using the classified image values greater than shadow blue (refer to ordering of colours 3.2.5.2). Blobs are formed based on classified colour. To save cpu processing the colours: undefined, white and field green are not included in blob formation. This is due to the fact that size and shape information is not required for objects of these colours. It is also due to the nature of classification of these colours. They are all abundantly spread throughout images and scattered; forming blobs on images like those in fig x.x would result in a lot of unnecessary processing. The basic colours that are used in blob formation are termed **object** blobs and the blobs formed on soft colours are termed **soft** blobs.

3.4.3 Blob Formation Logic

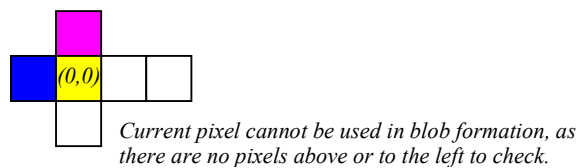
For the width and height of the image each pixel is made the '*current pixel*' and compared to that of the pixel prior horizontally and vertically in the methods '*check horizontal pixel*' and '*check vertical pixel*' respectively to see if it is of the same colour. The pixels can be *associated* if the classified colour values are the same and they are then allocated the same blob number and the blob object parameters are expanded. This logic is carried out for the entire area of the image. As blobs are formed they are pointed to by the array **blobs[]** with the size equal to the amount of blobs formed.

3.4.3.1 Horizontal and Vertical Check Method

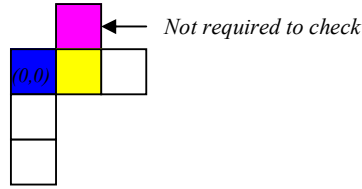


The method *check horizontal pixel*, checks the pixel previous to *current pixel* for blob association. The vertical pixel is then checked in the *check vertical pixel* method. In the event that blob association is found, the blob parameters are updated and the pixel is associated with the blob. This is done for the *current pixel* for the width and height of the image.

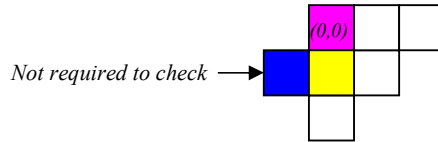
Design considerations of this method of blob formation are:



Due to this logic the fist pixel at (0,0) cannot be classified.



There is no need to check the vertical pixel for the first row

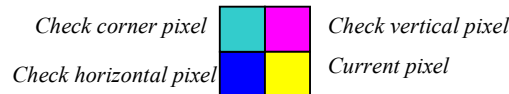


There is no need to check the left horizontal pixel for the first column.

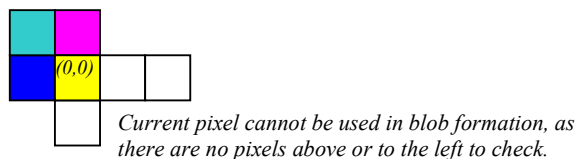
3.4.4 Blob Formation Rewrite

Problems occur in blob formation in the event of pixelisation. A large amount of blobs can form within the true blob when there is scattering of classified pixels. The original form blobs structure relied on an iterative searching process during merging of blobs; this meant that processing of blobs was proportional to the amount of blobs formed rather than the number of pixels in the image. Two blob formation techniques were designed to assist blob formation. ‘*Corner check*’ with methods: *check horizontal pixel*, *check vertical pixel*, and *check corner pixel* or ‘*second pixel check*’ with methods: *check second horizontal pixel*, if blob association not found then *check horizontal pixel*, *check second vertical pixel*, if blob association not found then *check vertical pixel*. The *corner check* blob formation method was implemented and the *second pixel check* to still be integrated. Blob formation was also restructured into methods for comprehensibility.

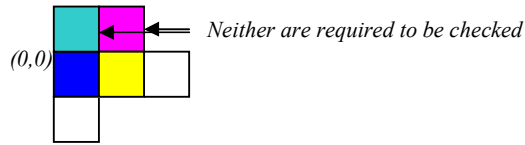
3.4.4.1 Blob Formation using Additional Corner Check:



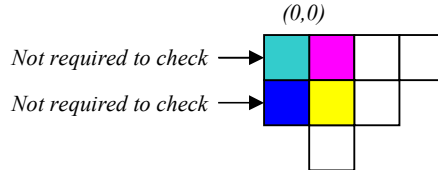
This method checks the corner pixel in addition to the horizontal and vertical pixel for blob association. This allows for “checker board” arranged pixels to be grouped into a single blob. It has similar design considerations to that of the horizontal and vertical blob formation method.



Due to this logic the first pixel at (0,0) cannot be classified.

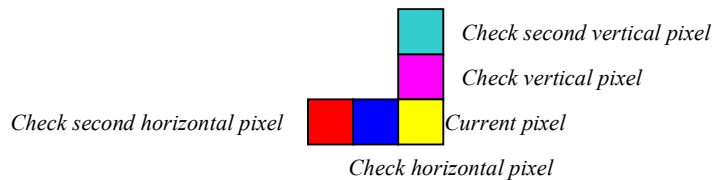


There is no need to check the vertical pixel or the corner pixel for the first row.

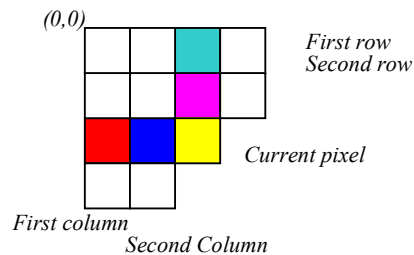


There is no need to check the left horizontal pixel or the corner pixel for the first column.

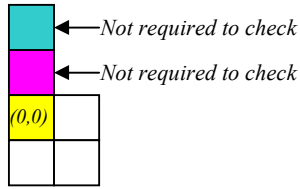
3.4.4.2 Blob Formation using Additional Second Pixel Check:



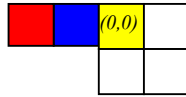
Second pixel check essentially acts as a 'skip two' pixel check with an additional safety net of checking one pixel as well if no match is initially found. This reduces the processing of large classified areas and also allows for pixelisation considerations. This method of blob formation is ideal for large areas of the same classified colour and close images, however distant objects in images i.e. small areas of classified colour are more likely to have an error in blob formation as a two pixel error could join the actual object pixels to a noisy blob.



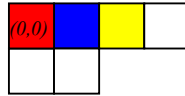
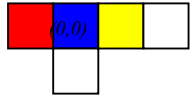
The first two rows and columns must be treated specifically in the design of second pixel blob formation.



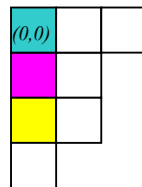
For the first row there is no need to check second vertical pixel or to check vertical pixel.



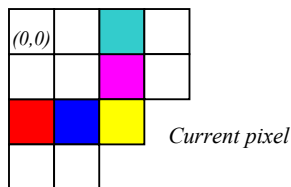
For the first column there is no need to check second horizontal pixel or to check horizontal pixel.



The first and second row can be checked one back from the second column and two back from the third column onwards.



The first and second column can be checked one above from the second row on and two above from the third row on.



For the third row and column onwards rules can be generalised to check second horizontal pixel for blob association, if not found then check horizontal pixel, check second vertical pixel, if blob association not found then check vertical pixel.

3.5 Soft Colour Filtering

3.5.1 Soft Colour Principal

Soft colour classification is a modification of basic colour classification as it allows for additional image information to be used. Previously there were a lot of issues involving under-classification and over-classification (3.2). For example shades of the orange ball could not be classified as they were shared with shades of the red team uniforms. This would result in either not enough size information being sent to object recognition for the ball or orange 'seen' in red uniforms. The implementation of soft colour classification allows for this and other classification problems, as it classifies shared shades as *soft colour*. Blob Formation creates blobs of classified colours- object blobs and soft blobs, which are then filtered and modified in the Soft Colour Filtering Process. Soft colour blobs are **conditioned** against corresponding object colour blobs and if conditions passed then either directly passed through to object recognition, or blob size modified and then passed on, or associated via object array and passed through to object recognition.

An object blob is linked to any associated soft colour with use of **object arrays**. A 2-dimensional array is created for the objects: ball, yellow goal and beacon, and blue goal and beacon. The structure of the array is *objectname[object colour][soft colour]* of size up to 100*100 i.e. the number of object blobs in an image with a maximum of 100, and the number of associated soft colour blobs up to 100.

3.5.2 Soft Colour and Blobs Formed

The *soft blobs* are conditioned against the *object blobs* using coordinates of the standard blob format is used in creating compare test conditions as shown Figure 3-9.

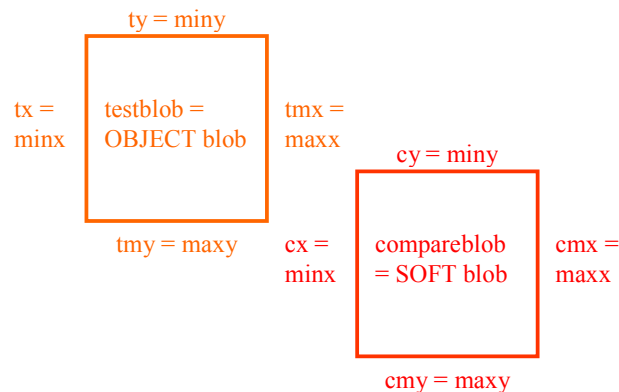


Figure 3-9 Parameters of blob used in soft colour filtering. *testblob* is the object-colour blob and *compareblob* is the conditional soft-colour blob.

3.5.3 Object Characteristics

Each object has particular blob formations depending on lighting, shadowing, angle and distance. These characteristics were thoroughly studied and tested using collections of image streams to create conditions for the soft colour filter software.

3.5.3.1 Ball

The ball is classified as shown in Figure 3-10. Red orange is classified to be that of the shadow, orange is classified to be the main area of the ball, yellow orange is that of the light patch. Examples of blob formation using this classification system are shown in Figure 3-11, Figure 3-12, Figure 3-13, Figure 3-14. Through study of blob formation conditions for each soft colour were determined.

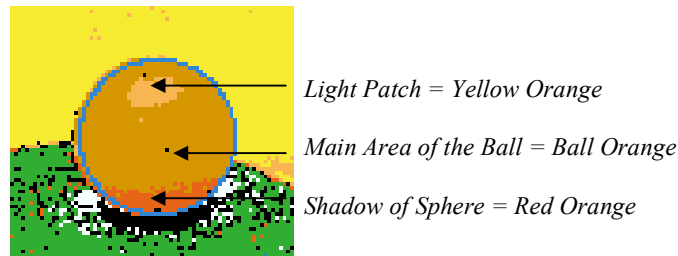


Figure 3-10 Ball classification principal.

3.5.3.1.1 Red Orange Conditions

A red-orange blob must overlap an orange blob to be apart of a ball. The red-orange blob may be wider than the orange blob, narrower, offset or enclosing it as shown in Figure 3-15. However the as red orange classifies the shadow of the ball a red orange blob cannot be an upper overlap i.e. the maximum-y of the red-orange blob cannot be above the maximum-y value of the orange blob, as shown in Figure 3-16.

If there is no orange in the image yet a large red orange blob exists, it is likely to be the ball at a close proximity with shadow cast by the robot. Large red orange blobs are passed in the initial size checks and do no need to be conditioned against orange.

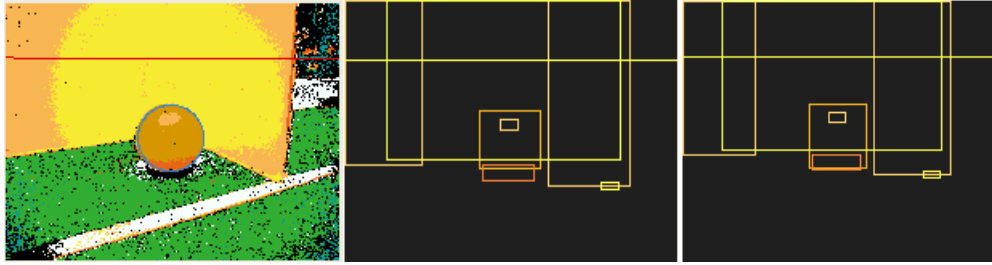


Figure 3-11 Ideal lighting of ball, its blob formation and its filtered size modified blob formation.

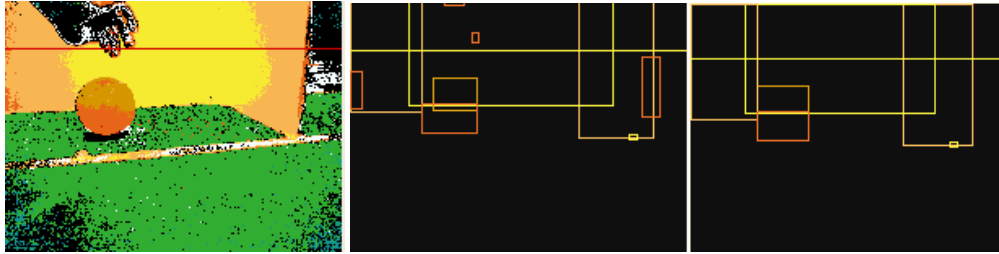


Figure 3-12 Shadowing of ball, unnecessary red orange blob formation, filtered size modified blob formation

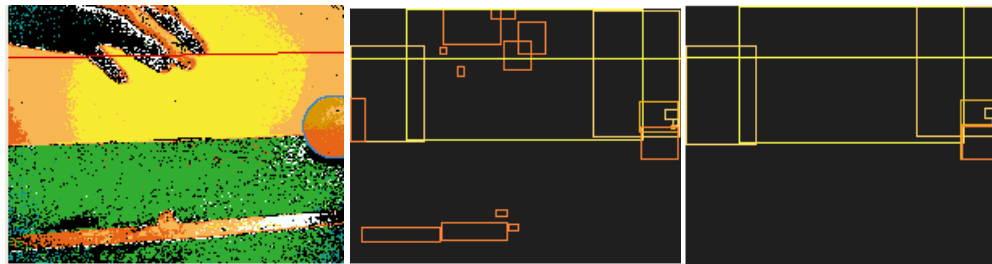


Figure 3-13 Corner images have increased noise, shadow creates unwanted red orange blobs, these are filtered out and the size of the orange blob is modified to the maximum dimensions of the overlapping red orange blob.

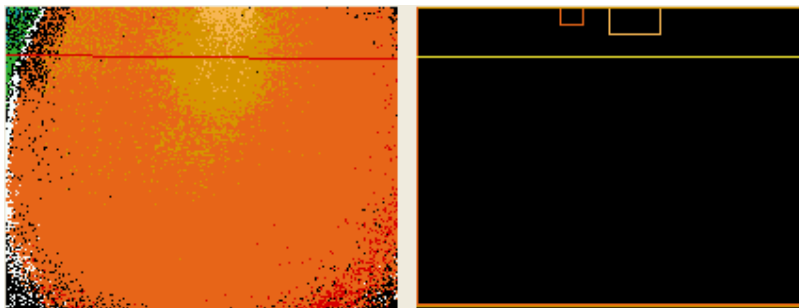


Figure 3-14 Close image of ball, the orange blob parameters are within the red-orange blob, the orange blob parameters modified to the most minimum and maximum of the two, in this case the size of the red-orange blob.

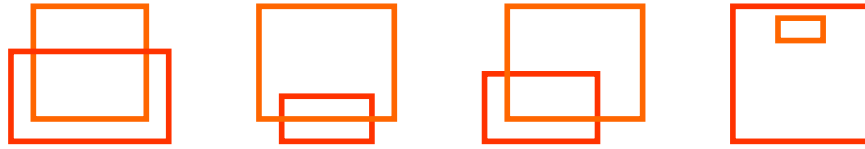


Figure 3-15 Acceptable red-orange / orange blob formations

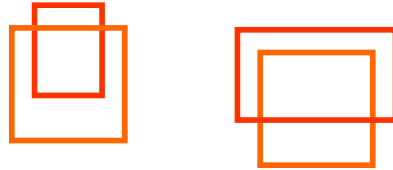
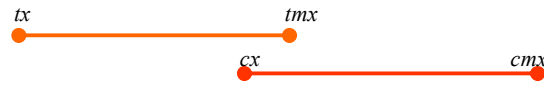


Figure 3-16 Unnecessary blob formation

Horizontally

The compare blob can be wider or narrower than the test blob but must overlap.

Overlap



$$\boxed{((tx \leq cx) \bullet (cx \leq tmx)) + ((tx \leq cmx) \bullet (cmx \leq tmx))}$$

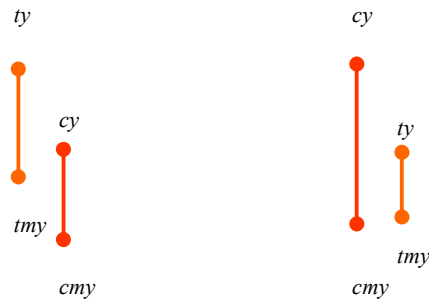
OR

$$\boxed{((cx \leq tx) \bullet (tx \leq cmx)) + ((cx \leq tmx) \bullet (tmx \leq cmx))}$$

Vertically

The compare blob can be below or enclose test blob but must overlap.

Lower Overlap or Fully Enclose



$$\boxed{ty \leq cy \leq tmy} \text{ OR } \boxed{(cy \leq ty \leq cmy) \bullet (cy \leq tmy \leq cmy)}$$

3.5.3.1.2 Blob Modification

Each red-orange blob that is passed (has acceptable overlap) is then compared to the associated orange blob. Comparison is of minimum and maximum values and the orange blob modified to fully enclose the red-orange blob with results shown in Figure 3-11, Figure 3-12, Figure 3-13, Figure 3-14.

3.5.3.1.3 Yellow Orange Light Patch Conditions

The light patch must be within an orange blob or overlap the top of an orange blob, but must not be wider as shown in Figure 3-17.



Figure 3-17 Light patch blob formation

Horizontally



$$(tx \leq cx) \bullet (cmx \leq tmx)$$

Vertically



$$ty \leq cmy \leq tmy$$

3.5.3.2 Yellow Goal and Yellow Beacon Conditions

The goal and beacons are classified to be *beacon yellow* for the main area under good lighting conditions. The darker shades of beacon yellow are classified *yellow orange* and tend to be the corners of the goal posts and edges of close images as shown in Figure 3-18.

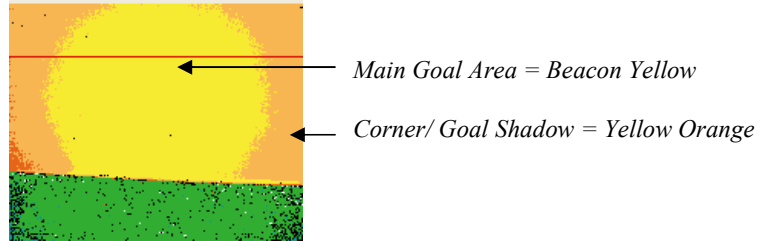
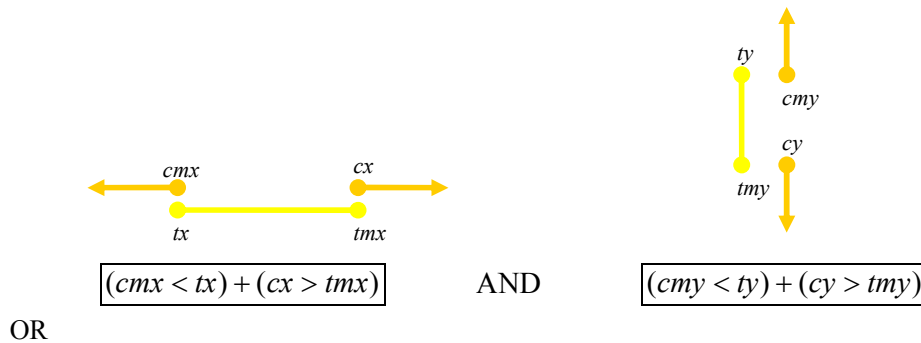


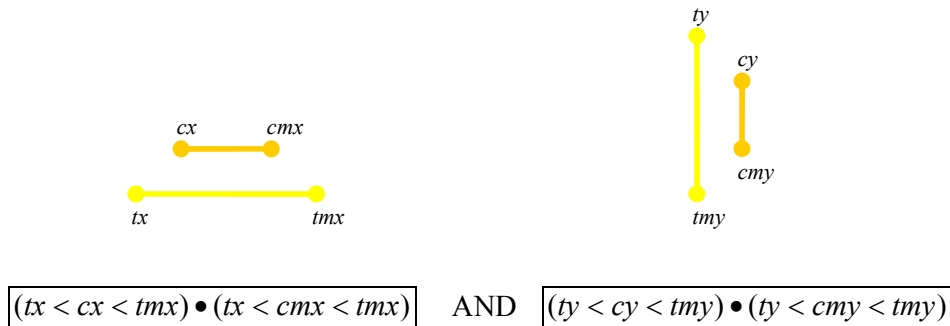
Figure 3-18 Yellow Goal Classification Principal.

By studying blob formation of goals it was noted that any overlap patterns between the object blob and soft blob were acceptable. Some of these are shown in Figure 3-22. For goals and beacons the conditions are: if the compare blob is not inside or outside the test blob then it must be overlapping.

NOT (Horizontal AND Vertical)



NOT (Horizontal AND Vertical)



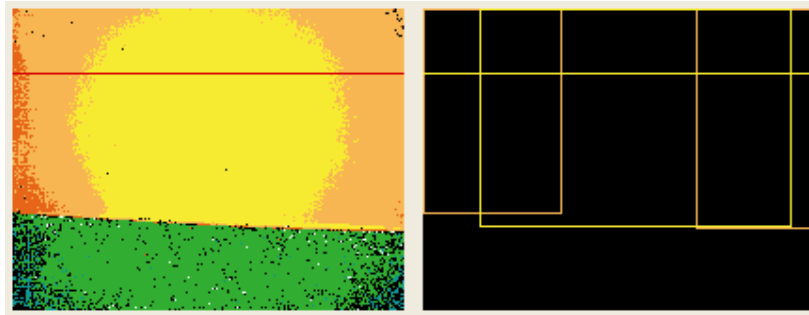


Figure 3-19 Classified image of goal and blob formation. Information of goal width is maintained without over-classification.

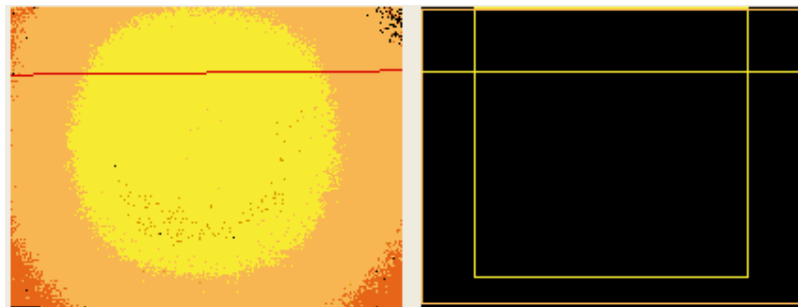


Figure 3-20 Image of close goal and blob formation.

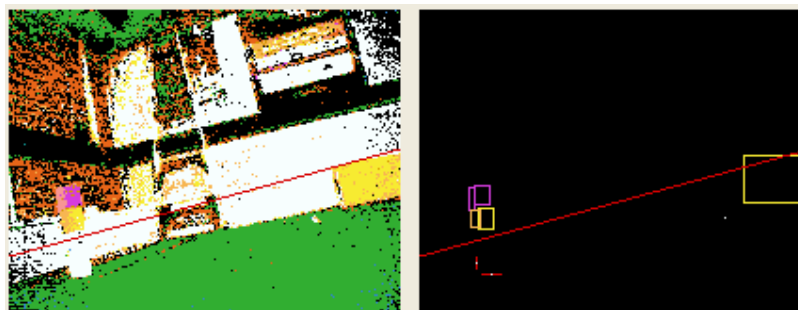


Figure 3-21 Image of beacon and soft colour blob formation



Figure 3-22 Acceptable blob formations for goals and beacons

3.5.3.3 Blue Goal and Blue Beacon Conditions

The blue goal and beacons are classified to be *beacon blue* and darker shades (typically the edges of the object and corners of the goal) as shown in Figure 3-23.

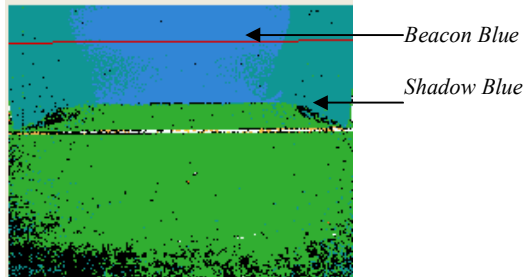


Figure 3-23 Classification Principal of Blue Goal.

For acceptable blob formations of blue goals and beacons see Figure 3-22. Additional filtering is required to remove inner blue blobs characteristic to blue shadow classification as seen in Figure 3-24.

Large shadow blue blobs located at the sides of an image have the possibility of being a goal corner and are passed in the initial size checks therefore do not need to be conditioned against an object blob. Figure 3-25

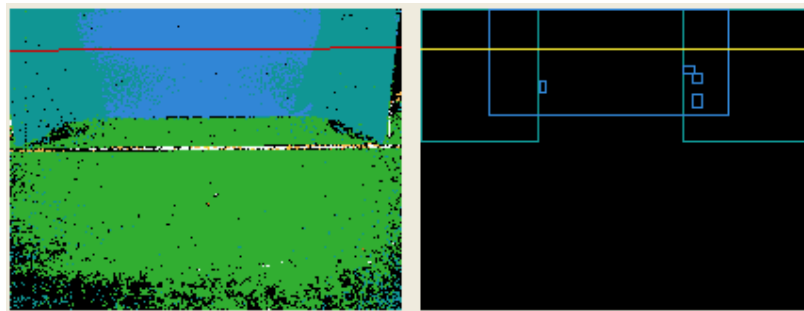


Figure 3-24 Blue blobs formed inside blue goal blob due to dispersion of shadow blue classification

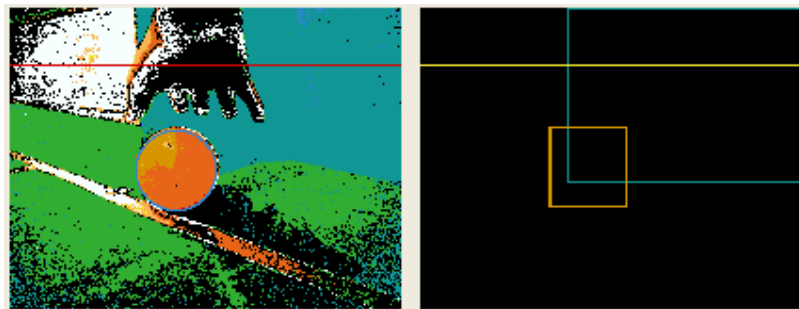


Figure 3-25 Corners of blue goal are very dark and classified as shadow blue. Occasionally there is no blue seen in the image, for this reason shadow blue blobs of considerable size located along the edge of an image are passed.

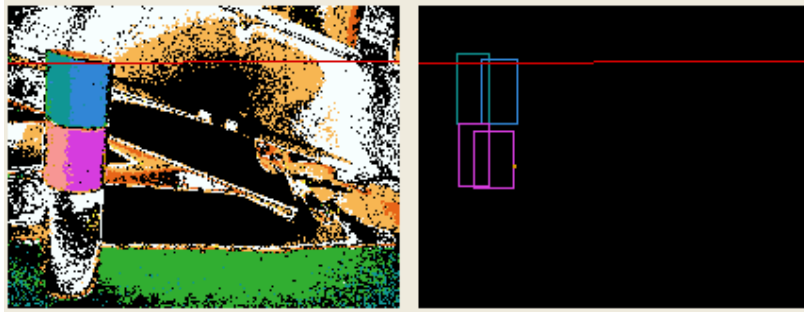


Figure 3-26 Blob formation of 3-Dimensional beacon characteristics

3.5.3.4 Pink Beacon Conditions

The blob formation of pink-orange and beacon-pink can be seen in Figure 3-26 and the logic for the filter seen in Figure 3-22.

Beacon pink is also overlap checked against yellow orange for the case that the lighting of a beacon is insufficient to make see it as yellow. The yellow orange blobs passed are used in a secondary beacon check after yellow.

3.5.3.5 Red Robot Conditions

Robot red is classified to be that of the red uniform and any overlap between red uniform shades and ball shadow shades are classified to be *red orange*.

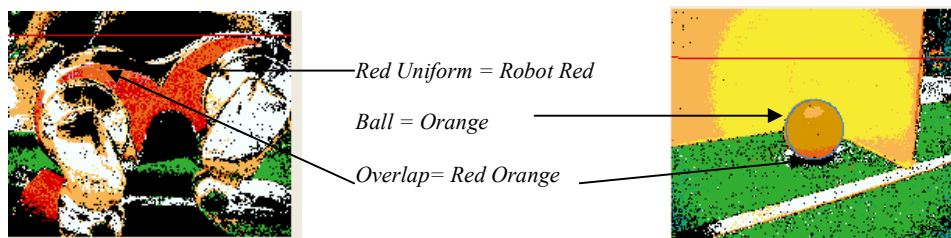
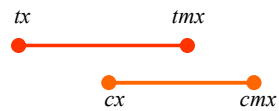


Figure 3-27 Red / Red orange Classification principal of red robot and shades shared with ball.

For red orange to be associated with robot red it must overlap. Any overlap of robot red and red orange results in the expansion of maximum and minimum parameters for the robot red blob. After conditioning of blobs and expansion, and also, due to the nature of scattered blob formations (similar to that of figure x.x) overlap occurs between two expanded red blobs. Any secondary overlap is also acceptable between robot red and robot red and expansion occurs.

Horizontal

Overlap of any kind



$$(tx \leq cx \leq tmx) + (tx \leq cmx \leq tmx)$$

Vertical

Overlap of any kind



$$(ty \leq cy \leq tmy) + (ty \leq cmy \leq tmy)$$

Maximum and minimum also found, red parameters expanded.

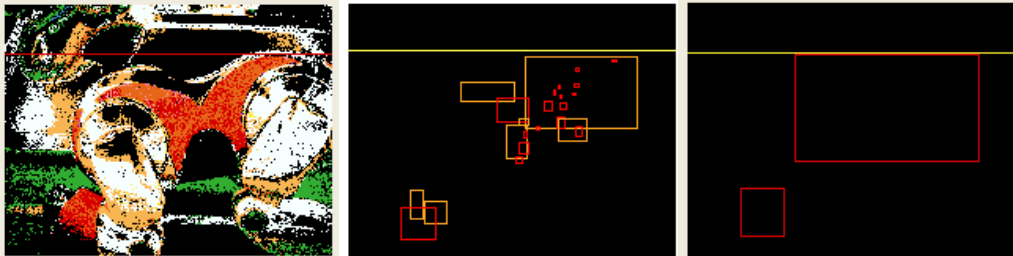


Figure 3-28 Red and red-orange are dispersed creating multiple blobs. Red orange overlapping red are ignored but red parameters expanded to account for actual size. After all red/ red-orange blobs are sorted the search is done again to account for previous

3.5.3.6 Blue Robot Conditions

One method of blue robot blob formation was to classify the entire spectrum of dark shades to be blue and then pass through initially based on size and later eliminated based on sanity checks.

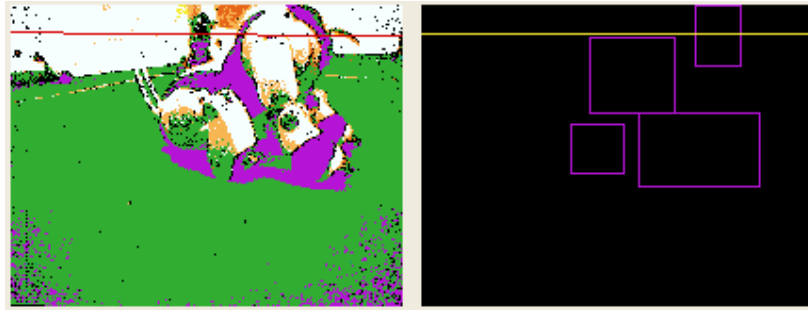


Figure 3-29 Blue Robot uniform shares shades of dark green and black objects, size is the main differentiator between robots and random objects.

However, with the introduction of secondary colour classification using edge detection for *robot blue* this method is not required. The *robot blue* blobs formed essentially are already conditioned and only need to be sanity checked in object recognition for cases where a non-blue robot pixel arrangement passed the secondary classification conditions.

3.5.4 Soft Colour Blob Filtering and Blob Size Modification Software

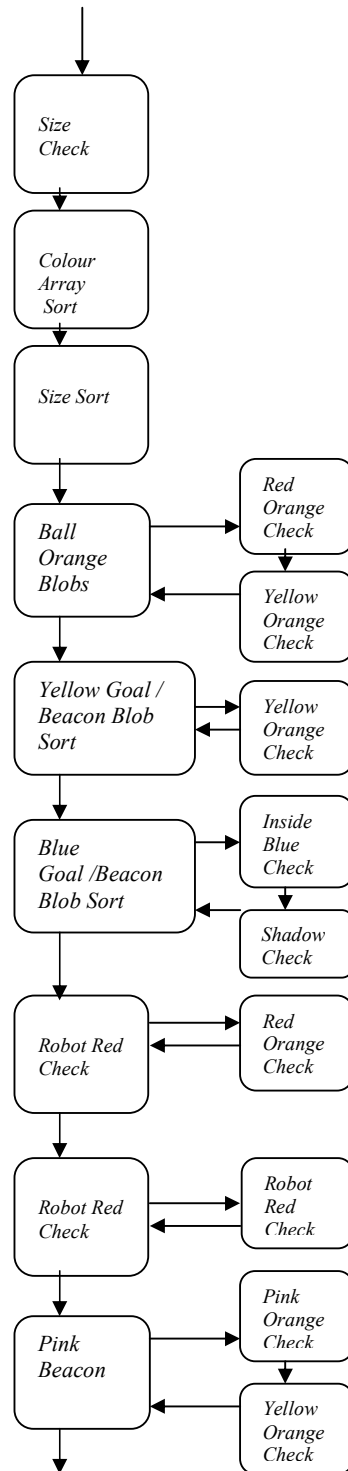


Figure2.18 Soft Colour Blob Filtering and Blob Size Modification Software Flow Chart

3.5.4.1 Definition of Terms

Passed, when *ignore blob* is set to false.

Ignored, when *ignore blob* is set to true.

Sanity checks, the process of sorting blobs for objects in object recognition.

Conditioned, the soft colour term for sanity checking.

When blobs are formed they are arranged in an the array *blobs[]* with the size that of the number of blobs formed. **Conditioning uses the blob property “ignore blob”**(see section x.x) **to pass or eliminate blobs**. By default *ignoreBlob is set to false*; i.e. all blobs are initially passed so that later they can be sorted.

Each blob of *blobs[]* is initially conditioned and then sorted into a 2-Dimensional colour-blob array.

3.5.4.2 Initial Ignore Conditions

3.5.4.2.1 Initial conditioning: size

Initial size conditions are dependent on the colour of the blob. Blobs of *object* classified colour are passed if width or height is greater than or equal to three pixels, for *soft* colours it is five. All soft colour blobs of acceptable size are initially set to ignore but passed into array. If shadow blue or red orange blobs are larger than forty pixels in height or width then they are passed, as they may be objects close to the robot.

3.5.4.2.2 2-Dimensional Colour Array Sort

As the array *blobs[]* is initially conditioned the blob pointer array *colourblobstest[colour][colour index]* is built. The array allows for blobs to be organised by colour and index of the colour blob, for use in filtering.

3.5.4.3 Size Sort

The 2-dimensional colour array is rearranged so that when indexed, the largest blob to the smallest is pointed to for each colour.

3.5.4.4 Ball Orange Blobs

Each orange blob is copied to a test blob with identical maximum and minimum values and then compared to each corresponding soft colour blob. A pointer and orange index is kept for all orange blobs, any passed soft blobs are then linked by the use of a 2-Dimensional ball-object array.

3.5.4.4.1 Red Orange Check

Each red orange blobs' parameters are compared to that of the orange blobs' and conditioned according to logic of 3.5.3.1.1. If passed the red orange blob is not ignored and is indexed and pointed to by the ball object array. The parameters of the orange blob are then modified to account for the soft colour blob overlap.

3.5.4.4.2 Yellow Orange Check

The parameters of each yellow orange blob are compared to that of the orange test blob according to the logic of 3.5.3.1.3. If passed the blob is not ignored and is assigned to an object array.

The modified test blob parameters are then copied to the original blob.

3.5.4.5 Yellow Goal / Beacon Blob Sort

Each yellow blob maximum and minimum value is compared to that of yellow orange. A goal/beacon array is indexed and blob pointed to.

3.5.4.5.1 Yellow Orange Check

Each yellow orange blob parameters are compared to each yellow blob. If overlap occurs (according to 3.5.3.2 filter conditions) then the yellow orange blob parameter ignore blob is set to false.

3.5.4.6 Blue Goal / Blue Beacon Blob Sort

Each blue blob maximum and minimum parameters are copied to a test blob and then compared to check for inner blue blobs and overlapping shadows. Ignore blob of blue blobs are set to true and then conditioned.

3.5.4.6.1 Inside Blue Check

Each blue blob is compared to every blue blob (including itself) by creating another test blob "compare blob" and comparing the two. If passed then compare-blob is not inside any others and ignore blob is set to false.

3.5.4.6.2 Shadow Blue Check

Each blue blob parameters are compared to each shadow blue parameters. If overlap does not occur according to 3.5.3.3, then ignore blob is set to true (as it is set to false by default). Indexed and pointed to by blue goal object array.

3.5.4.7 Red Robot Blob Sort

Each red blob is copied to a test blob.

3.5.4.7.1 Red Orange Check

Each red orange blob is checked to see if it overlaps with red. The red-orange ignore blob status remains at true but the area of red is modified.

3.5.4.7.2 Secondary Red Check

The new modified red blob array is then checked again for overlap against red orange to account for two modified blobs overlapping.

3.5.4.8 Pink Beacon Blob Modification

Each beacon pink blob maximum and minimum parameters are copied to a test blob and compared with each pink orange blob.

3.5.4.8.1 Pink Orange Check

Pink orange blobs are compared to pink blobs 3.5.3.4. If passed then the most minimum and maximum values of the two blobs are found.

3.5.4.8.2 Yellow Orange Check

Yellow orange blobs are compared to pink blobs 3.5.3.4. If conditions are passed yellow orange is *passed* and used in sanity checking of beacons.

3.5.4.9 Secondary Colour Array Sort

After filtering the blob pointer array *blobs[]* points to the modified blobs. *blobs[]* is indexed and a secondary 2-dimensional colour array *colourblobs [colour][colour index]* is created that only points to *passed* blobs. The array points to blobs arranged by colour, indexed from largest to smallest, for use in Object Recognition.

4 Look up Table (LUT) Generation

4.1 Manual Generation

Manual LUT generation is the method where the user must selectively classify each shade. This may lead to holes in the 3-Dimensional LUT.

4.2 Discussion

Classification of images is extremely important to vision, it is a very time consuming operation as attention to detail is required over thousands of test images. The NUbots debugging application Eye Of The Nubot (EOTN) is used for the building of LUTs and testing of the vision system. EOTN was modified to include additional soft colours and to incorporate a 7*7*7bit LUT system.

4.3 EOTN

EOTN has the graphical user interface as shown in Figure 4-1

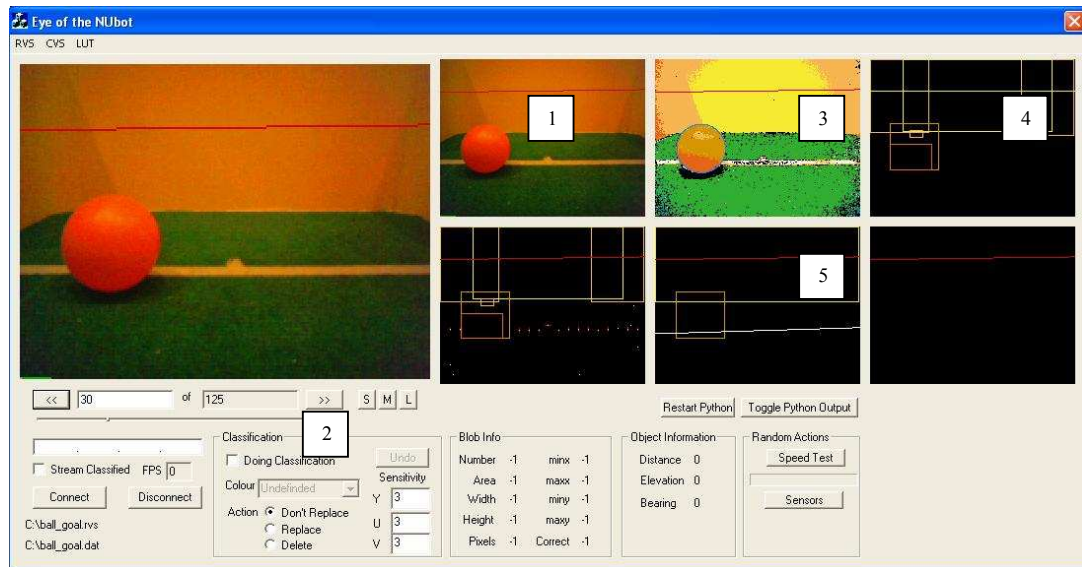


Figure 4-1 EOTN user interface; 1 Bitmap Image, 2 Classification Tools, 3 Classified Image, 4 Blobs Formed, 5 Object Recognition.

4.3.1 Input Image

A raw vision stream (rvs) file is either streamed into EOTN by connecting to a robot via wireless communication or opened from a previously streamed file. An rvs file is a series of YUV bitmap images input from the camera of the robot and is used in classifying images.

The quality of the image is dependent on the robots motion, camera settings and lighting conditions surrounding the image as discussed in 2.2. For the image to be displayed in EOTN each YUV pixel value is converted to RGB.

Y- Luminescence, U- Chrominance a, V- Chrominance b
R- Red Component, G-Green Component, B- Blue Component

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Figure 4-2 Conversion of RGB ↔ YUV was found at Wikipedia, the free online encyclopaedia [7].

4.3.2 Classification Tools

The classification tools of EOTN are used when building a LUT. An image pixel selected for classification has a range of possible operations. All values of a LUT are initially set to *undefined*.

4.3.2.1 Y, U and V Range

The 7bit cropped YUV value of the selected pixel is the base LUT index to set, the radius of Y, U and V values to classify also, is adjustable. The implementation of 7*7*7bit LUT increased the number of indexes of the LUT therefore the range had less impact in filling holes in the LUT. For this reason accuracy is increased in classification however, more images are required when classifying and finer attention to detail.

4.3.2.2 Colour

The possible colours to classify the range of YUV values are listed in sec x.x and are chosen according to the corresponding object in the image. EOTN was modified to account for additional colours.

4.3.2.3 Action

Don't replace, colour classifies the pixels within the selected range that are undefined.

Replace, classifies all pixels within the selected range to be the selected colour.

Delete, resets all pixel values within the selected range to undefined.

Undo, undoes the last action.

4.3.3 Classified Image

The classified image is discussed in 3.2. Vision uses the YUV value of each pixel of the input image to reference a LUT to create the corresponding classified image. In EOTN the classified image is either created by referencing an rvs file to a selected LUT file, or opened directly from a classified vision stream (cvs) file. A cvs file is the classified image that the robot actually “sees”. It is streamed into EOTN directly from the classified image created in the program on the robots memory stick using a stored LUT. A cvs file is streamed into EOTN at a faster rate than an rvs image as there is less information required to be sent (classification codes instead of YUV values) therefore more images per second can be viewed. The classified image of an rvs file is able to be used in LUT generation, however the classified image of a cvs file can only be used for debugging purposes as the LUT used is on the memory stick and not necessarily accessible by EOTN.

4.3.4 Blobs Formed

After the classified image is calculated blobs are formed (3.4), filtered and size modified (3.5), the blobs that have been *passed* are displayed. These are the blobs that are sent to object recognition. By viewing the blobs passed it allows for finer detail in classifying, debugging of soft colour filter logic, and debugging of object recognition

4.3.5 Object Recognition

The blobs detected as objects are displayed in this window and are determined by the Object Recognition module of the Vision System.

4.4 7 Bit LUT

The reduction of YUV component cropping from 2bits to 1bit in the implementation of a 7*7*7bit LUT system increased the number of possible classification values from 262144 to 2097152. With the increased index range, accuracy in separation between colour classifications increased, however pixelisation is more likely to occur.

The implementation of the 7*7*7bit LUT system involved the modification of EOTN software so that it crops the input YUV components by one bit for use in creating and referencing LUTs. Also the application had to be modified to open, modify and save LUT files of size 7*7*7bits.

4.5 LUT Automation

For maximum robustness in the vision system a partially automated colour classification system was implemented. This involved the modification of a Support Vector Machine approach by M. Quinlan [8] that expands, crops and fills each colour space in the 3-Dimensional LUT.

4.5.1 Background

Support Vector Machines (SVMs) [9][10] were first developed to perform two-class classification. The solution involves the construction of a hyperplane that separates two input classes. In its simplest form, known as a maximum margin classifier, an SVM only works on linearly separable data (Figure 4-3).

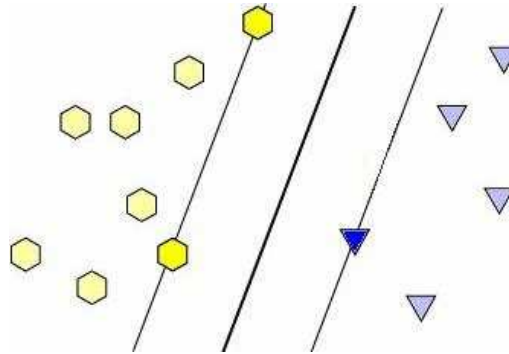


Figure 4-3 Example of a maximal margin classifier: The bold line indicates the separating hyperplane while the “darker” data points are support vectors.

By solving a constrained optimisation problem we can obtain a decision function for the hyperplane -

$$f(x) = \text{sgn} \left(\sum_{i=0}^l y_i \alpha_i \cdot (x \cdot x_i) + b \right)$$

Where $\{x_i, y_i\}$ is the input data, $x \in \mathbf{R}^n$, $y \in \{-1, 1\}$ and α_i is a Lagrange multiplier.

To build a non-linear SVM, the input variables are mapped in to a higher dimensional dot product feature space. This mapping to a higher dimensional feature space is hidden to both the input and output layers, thus allowing the construction of a hyperplane in this feature space with very little to no impact on runtime performance. The mapping is achieved by replacing the dot product ($x \cdot x_i$) with a Mercer kernel of the form -

$$k(x_i \cdot x_j) = \Phi(x_i) \cdot \Phi(x_j)$$

This results in a more general decision function -

$$f(x) = \text{sgn} \left(\sum_{i=0}^l y_i \alpha_i \cdot k(x_i \cdot x_j) + b \right)$$

In this implementation a Radial Basis Function (RBF) will be used so the substituted kernel becomes -

$$k(x_i \cdot x_j) = e^{-\gamma \|x_i - x_j\|^2}$$

The method proposed by Quinlan [8] uses a variation on SVMs called one-class support vector machines. One-class SVM was first introduced by Schölkopf [11], it differs from standard SVMs in the fact that it only requires one class of training data (i.e. We only need to provide positive training data and not a negative training data). The hyperplane constructed separates the input data from unclassified space. This enables SVMs to be used as a form of novelty detection, thus allowing the user to classify a point as either belonging to the known data set or not belonging to the set.

4.5.2 SVM Modification and Application

Because the support vector machine uses single-classification the order of colour operations develops a hierarchy of importance. Any operation that overlaps another colour space overwrites it. Therefore the order of operations is important, as is the action to perform in the event of overlap. For each colour the SVM checks the entire colour space to reduce processing it was limited to search between the minimum and maximum x, y, and z values.

4.5.2.1 Shorter Range

The minimum and maximum values of x, y and z are initially found in the LUT and then operations performed within that range, rather than over the entire 7*7*7bit range to reduce processing time.

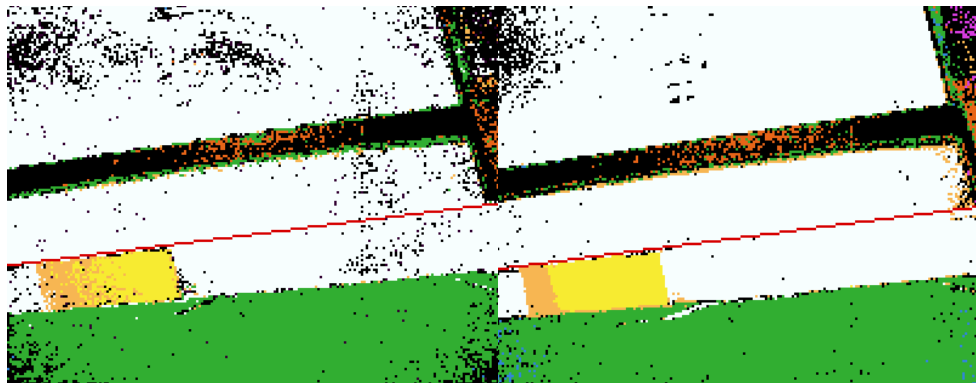
4.5.2.2 Sequence

It was found that by classifying non-object colours, basic colours and then soft colours was the most reliable structure for overlap issues.

4.5.2.3 Overlap Rules

In the case where two basic colours overlap the corresponding soft colour is assigned. This resulted in an over-classification of soft colour and is still in the process of testing and debugging.

4.5.2.4 Application to LUT generation



*Figure 4-4 Yellow is classified after yellow orange therefore yellow is the dominating colour.
Separation between colours is definite.*

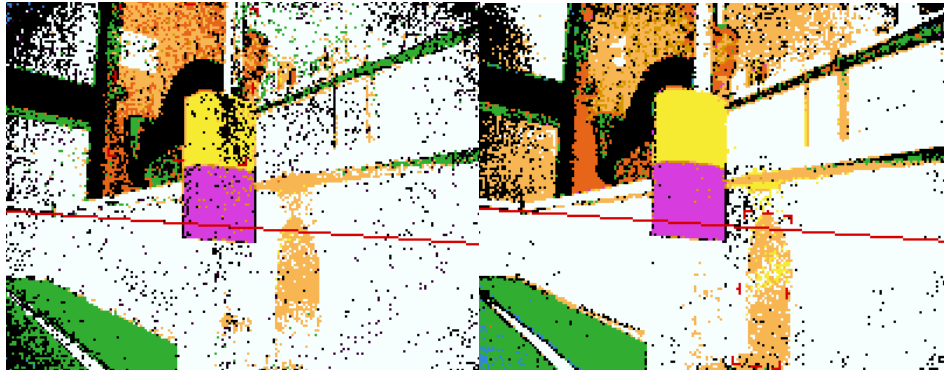
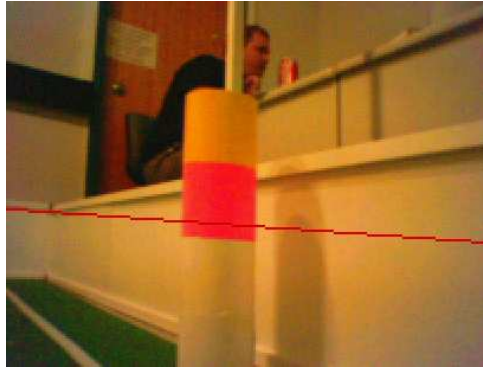


Figure 4-5 Holes in LUT filled.

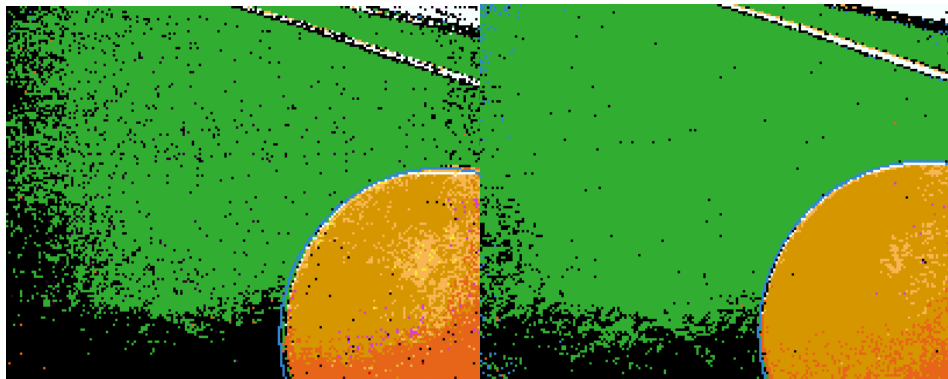


Figure 4-6 Cropping of colours and basic colour overlap.

5 Variable Lighting Challenge

The Variable Lighting Challenge is designed to increase efficiency in vision when illumination is non-uniform. A blue robot must score as many yellow goals as possible in three minutes whilst avoiding two stationary red robots. This is to be done under random lighting conditions that include constant lighting, uneven lighting, slow lighting changes, rapid changes with periods of no lighting.

The important objects that must be seen and distinguishable in all lighting conditions are the ball, red robots and yellow goal.

5.1 Soft Colour Classification under Varying Lighting Conditions

Soft colour classification allows for variation in lighting conditions. The ratio of basic colour blob size to soft colour blob size is not restricted therefore a wider range of lighting conditions can be accounted for.

5.1.1 Ball

The variation of lighting varies the ratio of orange to red orange in classified image of the ball; the lighter the image the greater the orange, the darker the image the greater the red orange. Whilst the classified image includes orange the blob size is modified to include red orange. If no orange is visible then the ball is searched for in red orange blobs.

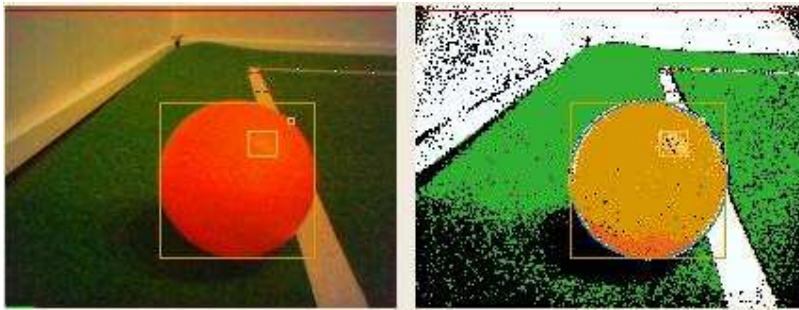


Figure 5-1 Normal lighting conditions orange: red orange is approx 8:2

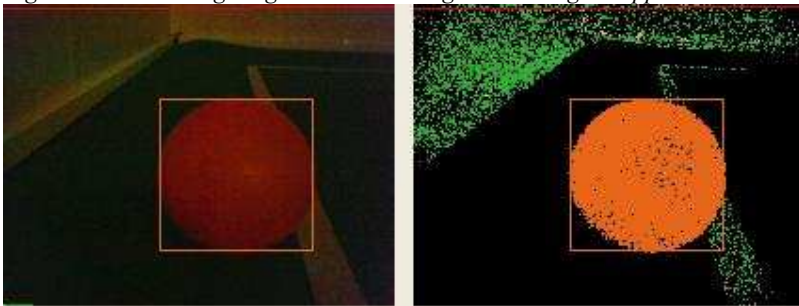


Figure 5-2 Dark lighting conditions orange: red orange is 0:1

5.1.2 Red Robots and Ball

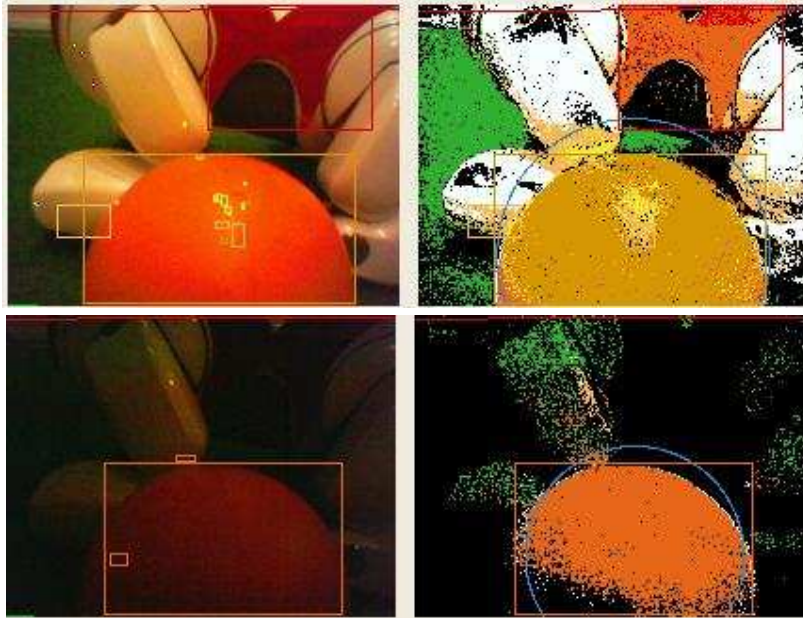


Figure 5-3 YUV Images and soft colour classified image of red robot and ball under normal and dark lighting conditions.

5.1.3 Yellow Goal

Limitations of soft colour buffering of variable lighting are seen in Figure 5-4, in the occasion where a yellow goal is cannot be seen in dark lighting or over-classification of red orange occurs in the yellow goal.

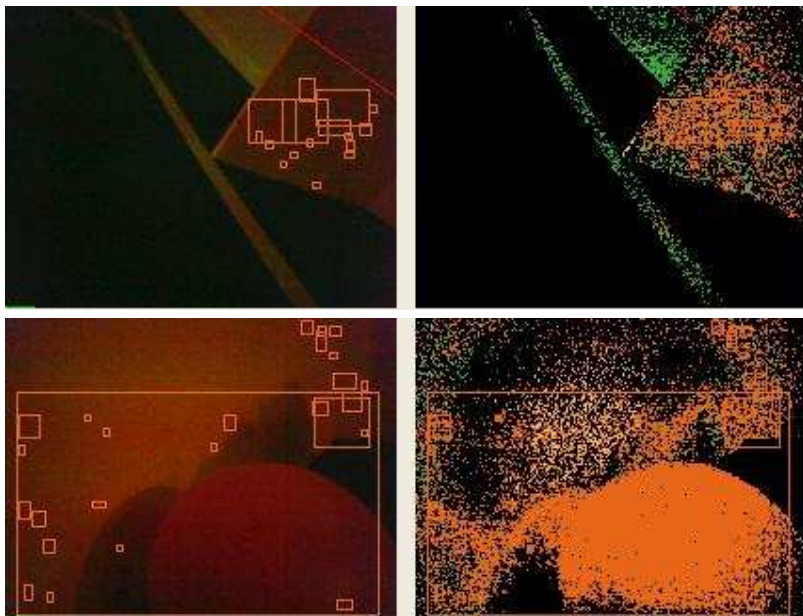


Figure 5-4 Instances of yellow goal images and ball in yellow goal images under dark lighting conditions are problems when relying on soft colour for the variable lighting challenge.

5.2 Variable Shutter Speed

There is a limit as to the reliability of soft colour blobs buffering the reduction of lighting levels. For this reason variable shutter speed was implemented for lighting below x LUX. The adjustment of shutter speed is dependent on the YUV input image.

$$Input\ Image = \begin{bmatrix} x_0y_0 & x_1y_0 & \dots & x_jy_0 \\ x_0y_1 & x_1y_1 & \dots & x_jy_1 \\ \vdots & \vdots & & \vdots \\ x_0y_k & x_1y_k & \dots & x_jy_k \end{bmatrix}$$

Where $X = \text{image width in pixels} = 208$

$Y = \text{image height in pixels} = 160$

$j = X - 1$

$k = Y - 1$

Each x_my_n value corresponds to an 8*8*8 bit YUV value defined by:

$$x_my_n = Y_\alpha U_\beta V_\gamma$$

To find Y_{av} Each 8 bit Y value of x_my_n for $m=0 \rightarrow X$ and $n=0 \rightarrow Y$ is averaged over the image area $X*Y$.

The average Y component value taken over the area of the image indicates the lighting level of the field. By setting the shutter speed in accordance with Y_{av} , dynamic lighting conditions can be accounted for. Resulting images are shown in figure x.x

5.2.1 Image Testing



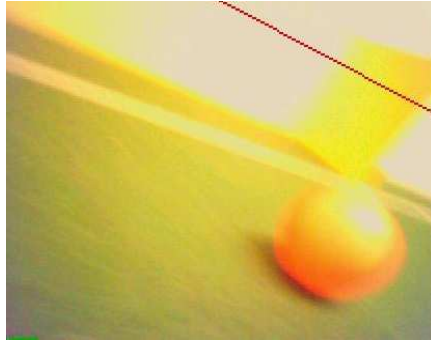
Shutter speed set to fast, lighting decreases $Y_{av}=22$ therefore change shutter speed.



Result of medium shutter speed



Result of slow shutter speed



Shutter speed set to slow, an increase in lighting therefore increase shutter speed.



Result of normal lighting with shutter speed set to fast.

5.3 Variable Lighting Challenge Behaviour Routine

The robot must score as many yellow goals as possible within a three-minute period of variable lighting conditions whilst avoiding two stationary red robots. After each goal the ball is positioned in the centre of the field.

With consistent reliable vision of the ball, yellow goal and red robots under a range of lighting levels, a behaviour routine is then required to ensure success in the challenge.

A design consideration with variable shutter speed is the effect of blurring in images. With the reduction of shutter speed, motion speed must also be reduced. Soft colours are sent to object recognition as secondary dark lighting object checks.

Behaviour routine involves; search for ball, chase, dodge red, aim for yellow goal.

Conclusion

The modifications made to the frame processing module improved the reliability of blob information for use in object recognition.

The soft colour classification system resulted in improved discrimination between colour classified objects by buffering basic colours. The effect of this improved discrimination on the vision system is the reduction of the probability of error due to shared colour shades.

Colour classification in combination with edge detection has solved the problem of classifying blue robots. This has not previously been effectively achieved.

Blob formation modification has resulted in additional soft blobs formed, increased in efficiency due to restructure and pixelisation addressed with additional blob formation checks.

The blobs sent to object recognition have improved reliability in values. Issues of over-classification and under-classification have been addressed in the soft colour classification process. Also, due to soft colour filtering and blob size modification image information is retained.

The LUT system has also assisted in the discrimination between colours. The increase in the number of possible classification values from the implementation of a $7*7*7$ bit LUT has resulted in increased accuracy separating colours when classifying. Also, the use of an SVM in generating LUT s resulted in well defined colour regions that reduce overlap between colours.

The Variable Lighting Challenge strategy has integrated the frame processing modifications and proved effective in improving vision under changing lighting conditions.

The improvements made to the frame processing system as discussed have resulted in increased reliability of blob information and hence improved object recognition. Vision is the first process in the robot system and hence improvement in vision results impacts all other modules. For this reason reliable vision is valuable to robot system in terms of RoboCup competition and additionally in the Variable Lighting Challenge.

References

- [1] Sony Aibo website, <http://www.aibo.com>
- [2] Official RoboCup website, <http://www.robocup.org>
- [3] RoboCup 2005 Legged League Competition Rules.
- [4] Technical Challenges for the RoboCup 2005 Legged League Competition
- [5] M. Quinlan, C. Murch, T. Moore, R. Middleton, L. A. Yung Li, R. King, and S. Chalup “The 2004 NUbots Team Report”, 2004, University of Newcastle. Available from <http://www.robots.newcastle.edu.au/publications.html>
- [6] Newcastle Robotics Laboratory www.robots.newcastle.edu.au
- [7] Wikipedia, the free online dictionary, www.wikipedia.org
- [8] Michael J. Quinlan, Stephan K. Chalup, and Richard H. Middleton. *Application of SVMs for Colour Classification and Collision Detection with AIBO Robots*. Advances in Neural Information Processing Systems (NIPS'2003), 2004.
- [9] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.
- [10] Bernhard Scholköpfung and Alexander J. Smola. *Learning with Kernels, Support Vector Machines, Regularization, Optimisation and Beyond*.
- [11] Bernhard Schölkopf , J.C. Platt, J. Shawe-Taylor , A. J. Smola, and R.C. Williamson Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443-1471, 2001.