TECHNICAL UNIVERSITY OF CRETE

Efficient Color Recognition Under Varying Illumination Conditions For Robotic Soccer

Andreas C. Panakos

Department of Electronic and Computer Engineering

Thesis committee:

Michail G. Lagoudakis, Supervisor
Michalis Zervakis
Nikos Vlassis

October 2009

"The more you know, the more you realize you know nothing."

Socrates

Abstract

Visual perception is a central problem in robotics because of the richness of information in color images, but also because of the difficulty in coping with image variations due to illumination conditions. In the RoboCup competition (Robot Soccer World Cup), where robots act autonomously using mainly visual cues, the objects of interest are characterized by unique colors and their recognition relies on the correct identification of image areas corresponding to the same color. This thesis addresses the problem of color recognition under varying illumination, namely the classification of any image pixel to the correct color class of the corresponding object, even when illumination conditions vary. The proposed approach is based on labeling by hand a representative set of images from the robot camera and training a classifier which generalizes over the entire color space. The illumination problem is addressed either by including special illuminant features (average color values from a large region or from the entire image) in the classifier, or by transforming the input to an appropriate reference illumination level through histogram specification before classification. These procedures have been integrated into the Kouretes Color Classifier (KC^2) graphical tool, which provides intuitive means for labeling images (regions, clusters), selecting features (neighborhood, illuminant), training classifiers (Decision Trees, Support Vector Machines, Neural Networks), and generating code for efficient execution on the robot. The KC^2 tool minimizes the user time required and delivers excellent color classifiers using only a few images. The proposed method has been tested successfully on the Sony Aibo and the Aldebaran Nao robots and can be used in a wide range of applications beyond RoboCup.

Acknowledgements

Although this chapter is among the last to write, is usually the first to read in a thesis. First of all, i would like to thank my advisor , Michail G. Lagoudakis, expressing appreciation for his support and encouragement throughout this thesis. He is not a usual professor for me (as he wants me dead), caring deeply about his students being an example of teacher for the academic community. I also thank the members of my graduate committee for their guidance and suggestions. I am also highly thankful to the people that have helped me throughout this study and especially the members of Kouretes Robocup team (cbm,rambo,vagvaz,psilos) without whose knowledge and assistance have helped me to finish my thesis. Undoubtedly, I am especially indebted to all my friends that over these years have supported me in my ″living nightmares‟. Of course, i would like to thank my family members, for supporting and encouraging me to finish my thesis.

P.S: Special thanks to the employees of military offices for their patience and accommodation.

CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Cognitive robotics is a growing research field, studying the robot reaction with its environment. The ability of a robot to understand its environment from visual inputs is central problem since autonomous robotics is an ideal platform for the application of computer vision techniques.

1.1 General Description

Machine vision is usually used as a primary sensory input because in cases where the camera is generally stationary, in a known lighting condition and with no temporal performance restrictions. RoboCup (Robot Soccer World Cup) is an international robotics competition where each robot player acts autonomously, relying on determine visual field features. Thus, in RoboCup conditions a challenging task is image segmentation, that is, the association of image regions with scene objects. Especially, in domain of Standard Platform League (SPL), where teams use identical Aldebaran Nao humanoid robot (NAO), colors comprise important information for analysis and are specialy important at the robot soccer domain environment. This thesis describes a Color Recognition System, that comprises the first stage of a pipeline architecture used in environment comprehension. Color recognition method that has as primary goal to recognise colors of an image on the basis of similarity of known color characteristics using pattern recognition techniques under varying illumination conditions. In addition, it is illustrated the Kouretes Color Classifier (KCC), a freely-available interactive software tool used in creation of a classification function that determines image membership. KCC offers easy to use training data selection methods, while a wide variety of options enabling robust and efficient color classification under a wide range of environmental conditions.

Moreover, we introduce two alternative method that adapt Color Recognition to varying illumination conditions. Image analysis is a intensive especially under unknown conditions especially under unknown lighting conditions where the appearance of color can dramatically affect method efficieancy. The illumination problem is addressed either by including special illuminant features (average color values from a large region or from the entire image) in the classifier, or by transforming the input to an appropriate reference illumination level through histogram specification before classification.

1.2 Contribution

The contribution of this thesis is to advance the state of the art in vision systems for mobile, autonomous robotics. The proposed method has a number of desirable properties that make it very suitable for use in real, low cost, high performance systems, addressing a number of major problems in mobile robot vision systems:

The first problem that we address is task of adaptation to dynamic conditions that remains a challenging problem for mobile autonomous robotics. Our robotic vision systemis built on the basis of color segmentation which works well in known lighting for which the robot has been calibrated and can overcome many of the problems associated with vision in dynamic and variable lighting environments. We show experimentally that our technique allows for robust object recognition even when lighting, temperature and intensity vary dramatically, as well as in the presence of dynamic shadows in natural lighting conditions.

The second major problem area is the development of Kouretes Color Classifier (KC^2) graphical tool, which provides intuitive means for labeling images (regions, clusters), selecting features (neighborhood, illuminant), training classifiers (Decision Trees, Support Vector Machines, Neural Networks), and generating code for efficient execution on the robot. The KC^2 tool minimizes the user time required and delivers excellent color classifiers using only a few images. The proposed method has been tested successfully on the Sony Aibo and the Aldebaran Nao robots and can be used in a wide range of applications beyond RoboCup.

1.3 Thesis outline

Chapter 2 provides the necessary information that reader needs to understand the broader context of the ideas this thesis introduce. At first, a brief introduction is given to the RoboCup competition and especially to Standar Platform League, field specifications and players who participate(Nao by Aldebaran Robotics). This chapter also contains an image processing overview which presents the principle of color theory along with a brief description of two image enhancement techniques. Finally, the basic concept of pattern recognition is given, followed by an introduction on three classification algorithms: Decision Trees, Neural Networks and Support Vector Machines.

Chapter 3 states the incentive to this work explaining thhe need for Color Recognition is explained. A summary of related work follows, that in many cases helped inproblemcomprehension along with evolution and validation of our approach. Finally the strengths and weaknesses of existing approaches are discussed trying to investigate the optimal solution.

Chapter 4 introduces the Kouretes Color Recognition Method,an approach for real time color recognition used in RoboCup competition. Implementation considerations and efficiency analysis are discussed. Finally technicalinformation of Kouretes Color Classifier (KC^2), a semi-automated tool which implements Kouretes color recognition method, is taken into account.

Chapter 5 provides valueable information about Kouretes Color Classifier KC^2 along with a brief description of Kouretes Color Recognition Method implementation.

Chapter 6 presents the evaluation of the proposed method. Afterwards, a discussion is done about its applicability and efficiency compared to other color recognition methods.

In chapter 7 an outline about the strengths and weaknesses of the approach is done, giving suggestions for future improvements and finally concluding this thesis with a brief summary.

CHAPTER 2

Background

2.1 RoboCup Competition

This section provides, information about RoboCup competition [1] and its origins. Especially, we will focus in RoboCupSoccer domain and, in particular in Standar Platform League whose field, rules and players (Aldebaran Robotics Nao) will be described. To begin with, RoboCup (Robot Soccer World Cup) is an international robotics competition, founded in 1993 having as major purpose to encourage research in Artificial Intelligence and Machine Learning through a friendly competition. The ultimate vision of RoboCup Federation is the conduct of a soccer game between a team of fully autonomous humanoid robots and the human world soccer champions by the year of 2050. Thus, every year the best robotic teams from all over the world, take part in RoboCup trying to overcome problems of robotics such as perception, cognition, action, coordination under real-time constraints. Afterwards, the proposed solutions are evaluated using sets of benchmarks with intention to boost Robotics. Beyond soccer, there are also competitions in developing service and assertive used in domestic tasks (RoboCup@Home), search and rescue missions (RoboRescue) and simplified soccer leagues for students up through age 19 (RoboCup Junior).

FIGURE 2.1: Standard Platform League at RoboCup 2008 in Suzhou, China.

2.1.1 Standard Platform League

Standard Platform League (SPL) [2] is one of the most popular leagues of the RoboCup competition (Figure 2.1), every game takes place between two teams of two to four humanoid Aldebaran Nao robot (Figure 2.2) players each, whereas before 2008 four-legged Sony AIBO used instead. Each game consists of two 10-minute halves and a 10-minute half-time break between them. Because of the abundance of rules, the need for a referee who will enforce the rules during the game is vital. Pushing an opponent, grabbing the ball, reflects disregard to the rules, that comes with a penalty of 30 seconds' exclusion from the game. Moreover, a penalty will be enforced if a player enters his own goal area as a defender. In addition, there are two assistant referees who place the robots manually to standard positions. However, robotsoperatefully autonomously, since during the game the only interaction between robots and humans is that of Game Controller that uses standard wireless connection to set the action state of robots (start, pause, goals, penalty, end of game). Last but not least, is the characteristic of the Standard Platform League is that no hardware changes are allowed; all teams use identical robots, diverging only in software development. Thus, the aforesaid principles come with by a unique combination of features that characterise the league: autonomous player operation, vision-based perception, legged locomotion and action. Given that the underlying robotic hardware is common for all competing teams, research efforts have focused on developing more efficient algorithms and techniques for visual perception, active localisation, omnidirectional motion, skill learning, and coordination strategies.

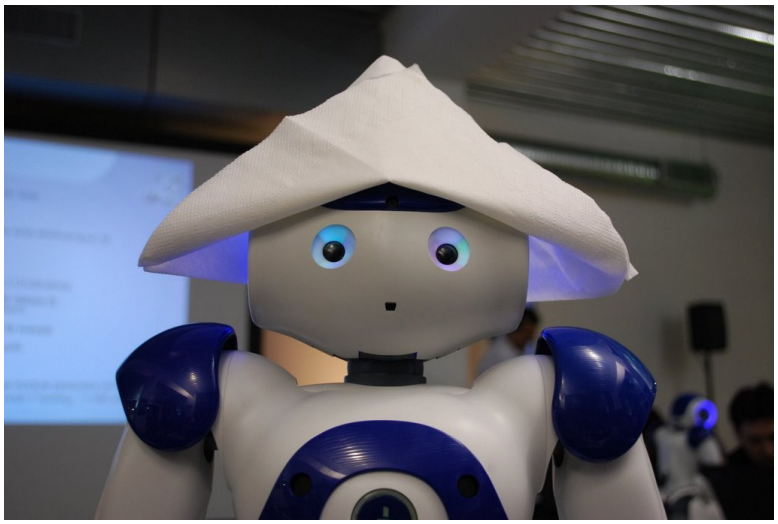
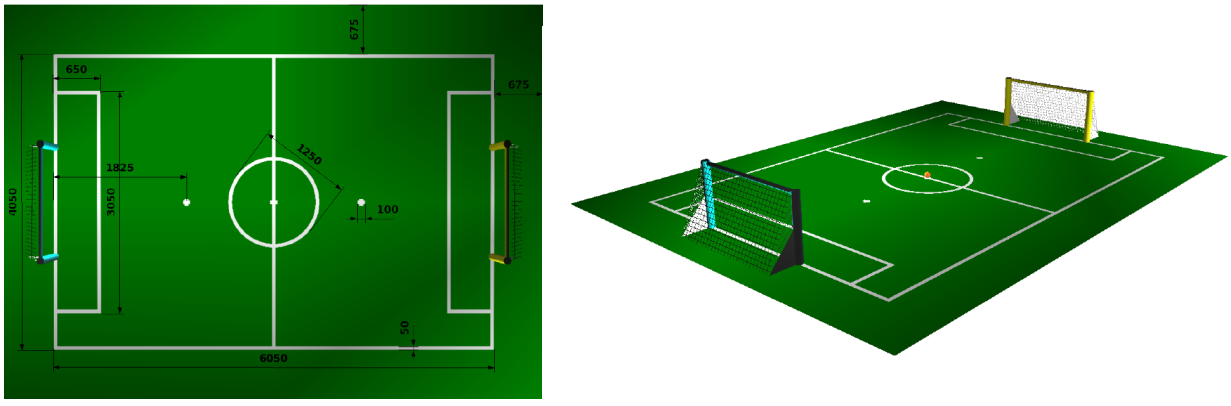


FIGURE 2.2: Aldebaran Nao Robot



(A) Scale diagram of entire field(dimensions in mm)

(B) Field colors and layout

FIGURE 2.3: Field Specifications

Field Dimensions

Games of SPL take place on a green carpeted field that is 6.05m long by 4.05m wide marked with thick white lines. All lines of field are 0.05m in width and the diameter of centre circle has an outside diameter of 1.25 m. There are also two colored goals (skyblue and yellow) that function as landmarks for localisation. Dimensions, colors and the layout of entire soccer field are shown in Figure 2.3.

Field Lighting Conditions

This is a milestone year for field lighting conditions since only natural lighting conditions of the actual site ceiling lights are allowed.

Field Colors

Each point in an certain color space corresponds to a certain color. A color class is a set of all color values that can be observed in pixels corresponding to an object having a known color. In other words, each color class is a subset of the color space, enclosing all variations of a certain color of interest in the real world. For RoboCup SPL games, the basic colors of interest are listed below (in decreasing importance) along with some comments on the difficulties associated with each one.

1. **Orange**: This is the color of the ball. The ball has a distinct orange color whilst in good lighting; however, when light is reflected by the ball or a shadow is cast over the ball, the perceived pixels shares shades of color with the yellow and the red colors, respectively.

2. **Yellow** : This is the color of the goal defended by the red team. Under certain circumstances, the orange may be misclassified as yellow.

3. **SkyBlue**: This is the color of the goal defended by the blue team. Under intense lighting, the dark blue color may be misclassified as the goal blue.

4. **White**: This is the color of field lines, robot bodies, and barrier walls.

5. **Green**: This is the color of the field. Dark shades of green occasionally may be perceived similarly to the dark image background outside the field.

6. **Red** : This is the color of the red teams uniform. Under certain lighting, red can mistakenly be classified as orange with catastrophic eects in the game.

7. **Blue**: This is the color of the blue teams uniform. Classification is very difficult as the values of blue are shared with the black and the dark shades of blue and green.

8. **NoColor** : This is an artificial color class for characterising any color not belonging to any of the above classes.

2.1.2 Aldebaran Robotics Nao-Players

Aldebaran Robotics, a French company based in Paris, has developed a fully programmable, autonomous humanoid robot, 58 cm tall and 4.3 Kg heavy (Figure 2.4). The name of this humanoid robot is *Nao*, which respectively stands amongst its robotic brethren. Nao has not been released commercially yet, but will be offered to public in 2010. However, Nao Academics Edition is already available for research purpose, promoting Nao as an educational robotic platform and entertainment robot affordable to most budgets. The initial limited edition of the robot selected to be the official robot platform of the Standard Platform League in 2007.

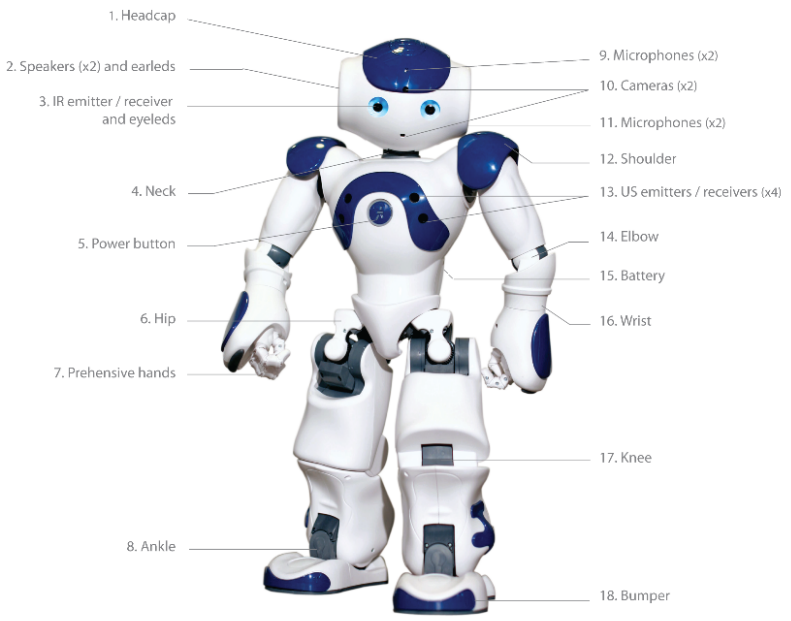


FIGURE 2.4: Nao parts

Nao specifications

Nao specifications has been developed using latest technologies providing a solid an efficient robotic system. The Nao robot consists of a full computer on board with an x86 AMD Geode processor at 500 MHz, a 256 MB SDRAM, and 1 GB ash memory running an Embedded Linux distribution. A 6-cell Lithium-Ion battery give to Nao a operational time of about 45 minutes, while communication is done via an IEE 802.11g wireless or a wired ethernet link. Four Force Sensitive Resistors (FSRs) sit under the foot plate on each of the Naos feet reading relative to how much force is squeezing and measuring the force distribution over the foot surface. while a 2-axis gyroscope/gyrometer, and a 3-axis accelerometer are mounted in the lower torso of the Nao. The gyroscope measures rotational velocity around the X and Y axis whereas , accelerometer measures linear acceleration along all three axes. This huge variety of sensors and actuators combined with embedded software modules allowing Nao to feature abilities such as obstacle detection, sound localisation, text to speech, visual pattern detection an finally communication through the many LEDs. This powerful multimedia system consists of a pair of microphones that allows for stereo audio perception, two ultrasound sensors on the chest allow Nao to sense obstacles in front of it.

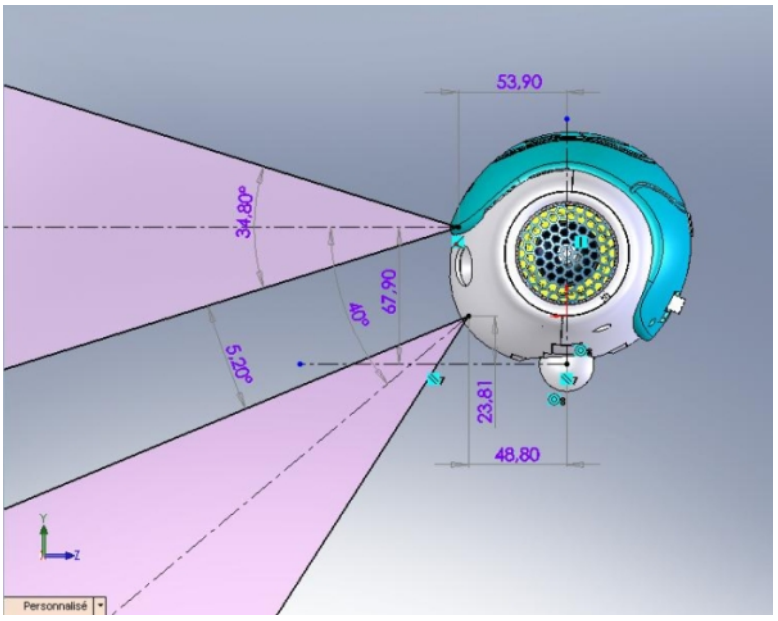


FIGURE 2.5: Side view of cameras offsets and positions

The Nao has 21 magnetically encoded motors. Twenty joints are individually powered, and two mechanically-linked diagonal hip joints are powered by a single motor. This joint configuration forms five kinematic chains. Finally, two 30fps, 640×480 color cameras are mounted on the head in vertical alignment, but only one is active at each time and the view can be switched from one to the other almost instantaneously. The two cameras provide non-overlapping views of the lower and distant frontal areas (Figure 2.5).

Nao programming environment

The Nao programming environment is based on the proprietary Naoqi framework, an open framework which serves as a middleware between the robot and high-level languages. NaoQi runs as a start-up process on a minimal Linux distribution, offering a distributed programming and debugging environment and an abstraction for event-based parallel and sequential execution. Its architecture is based on modules and brokers which can be executed on board the robot or remotely allowing your modules to register and integrate with NaoQis broker system over TCP/IP. A simple, higher-level, user-friendly programming environment is also provided by the recently released proprietary Choregraphe software. Finally, there are realistic computer models of the Nao robot available for both the Webots robot simulator and the Microsoft Robotics Developer Studio.

Camera Settings

The subject of this thesis is related to images. Thus, an essential prerequisite, is the cameras of Nao that are mounted on its head and are used as the primary mean of interaction between robot and its environment. Thus, Nao comes equipped with a low-voltage CMOS image sensor, which provides the full functionality of the VGA camera. It provides full-frame, sub-sampled, or windowed 8-bit images in a wide range of formats. The camera is capable of operating at up to 30 frames per second (fps) in VGA with complete user control over image quality, formatting, and output data transfer. The user is able to configure the gain, exposure, gamma, white balance, color saturation,hue control, and many more settings.

White balance: This setting is a color-correction system to accommodate for varying lighting conditions. The idea is that the white balance must be configured properly in order to set the white point, so that the other colors are mapped properly.

Gain: This setting controls the camera gain. From a qualitative point of view, higher gain makes the image look brighter.

Exposure: This setting denotes the amount of time the shutter allows light to enter through the camera lens. The lower settings are better when we want to decimate the number of the blurred images. For a qualitative view of point, lower exposure makes the image darker.

2.1.3 Kouretes

The team Kouretes founded in February 2006 by Prof. Michail G. Lagoudakis and in January 2007 Prof. Nikos Vlassis joined the team. Team activities in early stage where restricted to the four-legged league and later were extended to the simulation league. The team is currently active in the Standard Platform League in both simulated and real robot domains. The first exposure of team to RoboCup was at the RoboCup 2006 event in Bremen, Germany, where it participated in the Technical Challenges of the Four-Legged league. In the following year, team participated in the Four-Legged league of the RoboCup German Open 2007 competition in Hanover, Germany and were ranked in the 7th/8th place among ten teams from all over the world, featuring the team's first win and first goals. In Spring 2007, the team began working with the newly-released Microsoft Robotics Studio (MSRS), developing its software from scratch. The team's participation in the MSRS Simulation Challenge at RoboCup 2007 in Atlanta led to the placement of the team at the 2nd place worldwide bringing the first trophy home. Kouretes was the only European participating team among the nine teams from all over the world.

FIGURE 2.6: Kouretes Team at RoboCup 2008 in Suzhou, China.

In the most recent RoboCup 2008 competition in Suzhou, China, team participated in all divisions of the Standard Platform league (AIBO robots, Nao robots, Nao Webots simulation, Nao MSRS simulation). The team's efforts were rewarded in the best possible way: 3rd place in Nao league, 1st place in the MSRS simulation, and among the top 8 teams in the Webots simulation.

2.2 Understanding Images

This section provides information about different aspects of digital images. It is explained a way that a vision system works, what a digital image is and how digital images are represented. Furthermore, two digital image enhancement techniques are introduced, used to improve the interpretability or perception of information in images for human viewers, or to provide 'better' input for other automated image processing techniques.

2.2.1 What is a digital image?

Digital images are created by digital cameras that work in the same way as human eye. Like the light sensors in the eye create chemicals that tell the brain how much light is seen, electronic sensors detect photons of light and create a voltage that represents the intensity of light hitting the sensor. An image has a definite height and width consisting a grid of pixels.

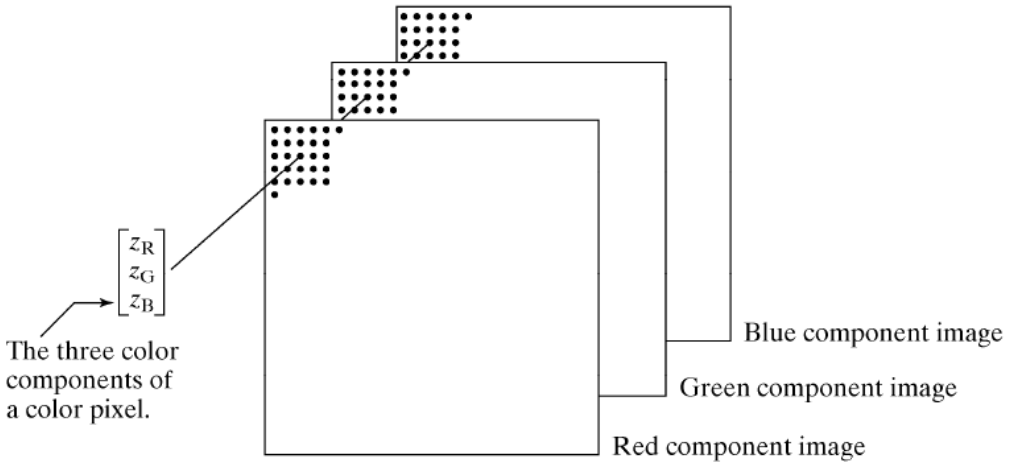


FIGURE 2.7: Structure of a pixel in a digital image.

Each pixel is described by three values that specify the intensities of three different color channels (Figure 2.7). The combination of these intensities have as result the color of each pixel. This color is represented by a 32-bit integer, hence intensities of each channel can vary between 0 and 255. In addition, we need to make a brief introduction to the *Color Theory* which pertains what humans perceive as color.

2.2.2 What color is?

Generally, color is the brains' reaction to a specific visual stimulus. Color derives from the spectrum of light (distribution of light energy versus wavelength) interacting in the eye with the spectral sensitivities of the light receptors. Typically, only features of the composition of light that are detectable by humans (wavelength spectrum from 380 nm to 740 nm, roughly) are included. The reason for this redundancy is that the eyes retina samples color using only three broad bands, roughly corresponding to red, green and blue light. The signals from these color sensitive cells are combined in the brain to create a multidimensional space of color components. These three primary colors are sometimes called tristimulus values. The response curve, as a function of wavelength for each type of cone, is obtained by a linear combination of the three primary colors (R, G, B) with the particular weights cR(l), cG(l) , cB(l) and is illustrated in Figure 2.8. Because the curves overlap, some tristimulus values do not occur for any incoming light combination. The set of all possible tristimulus values determines the human color space. It has been estimated that humans can distinguish roughly 10 million different colors.

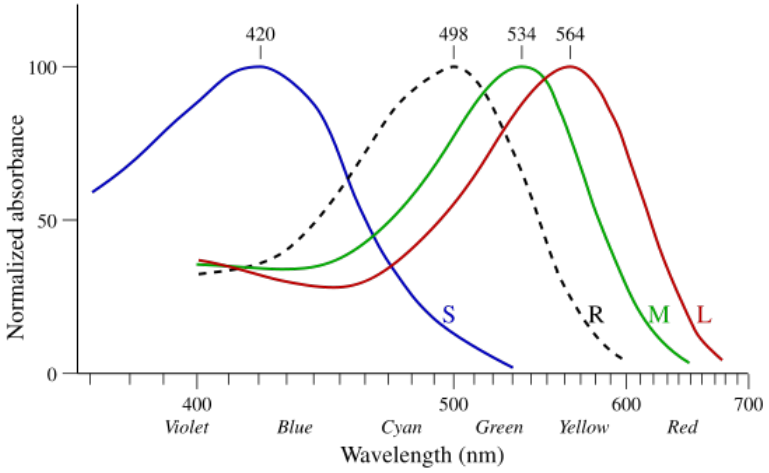


FIGURE 2.8: Normalised absorption spectra of human cone (S,M,L) and rod (R) cells.

2.2.3 Color Spaces

The native format of the two intergated cameras that Nao provides, is the YUV422. In that color space the Y-channel represents the brightness of the pixel, the U-channel the blueness, and the V-channel the redness. To understand what information is stored in an image, it is useful to know the different types of information that can be extracted from an image and how to display them. Moreover, we explore different ways of color representation that are called color spaces. In particular, using a color space we can specify, create and visualise color. Furthermore, as humans can define a color by its attributes of brightness, hue and colourfulness, computers can describe a color using the amounts of red, green and blue. Because different color spaces are better for different applications, we need to outline the strengths and weaknesses of the most prominent color spaces. For instance, limiting factors that dictate the size and type of color space are the color space linearity, the color space intuition and finally the device dependability [3].

RGB color space

RGB (Figure 2.9a) encoding is based on tri-chromatic theory and used for displaying color on the television or computer screens. RGB consists of three components: red, green, and blue. The RGB color space can be naturally represented as a cube. It is useful for television screens because red, green, and blue pixels can easily be combined to create any color, however the RGB color space is not very intuitive and not an ideal format for manual color classification.

YUV color space

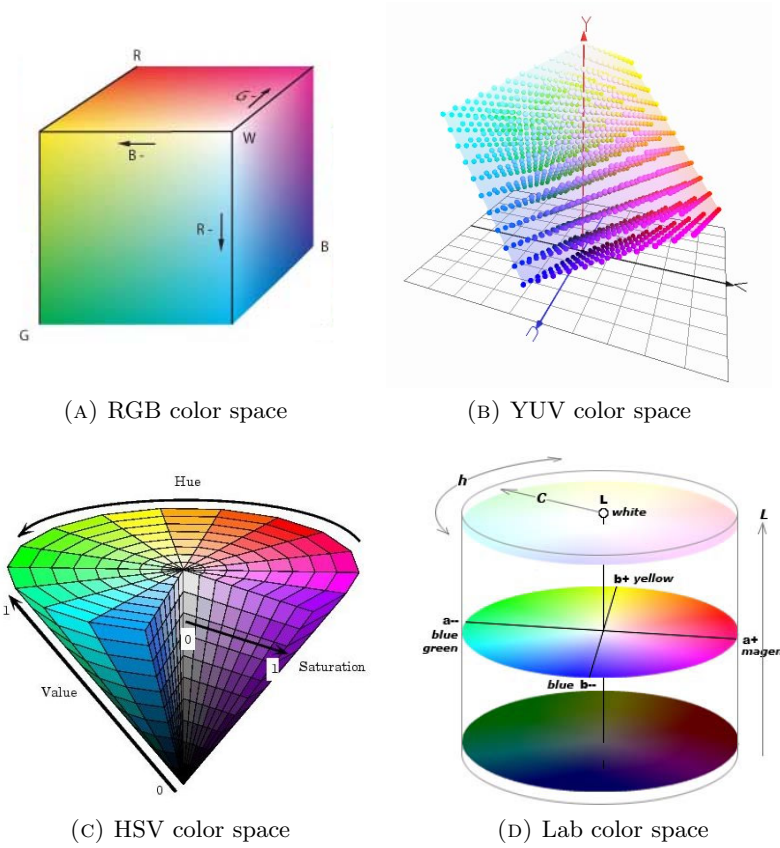
Similarly, *YUV* (Figure 2.9b) is the encoding used when television broadcasts are transmitted. The Y component is intensity, a weighted average of the R, G,and B values. Displayed alone the Y component creates a grayscale image, used in black-and-white television sets. The U and V components are chrominance values, which can be used with the Y value to obtain the R, G, and B values.

HSV color space

In contrast, the *HSV* model (Figure 2.9c) is commonly used in computer graphics applications. In various application contexts, a user must choose a color to be applied to a particular graphical element. When used in this way, the HSV color wheel is often used. In it, the hue is represented by a circular region; a separate triangular region may be used to represent saturation and value. Typically, the vertical axis of the triangle indicates saturation, while the horizontal axis corresponds to value. In this way, a color can be chosen by first picking the hue from the circular region, then selecting the desired saturation and value from the triangular region.

LAB color space

Last but not least, the *CIELAB* (Figure 2.9d) Color Model (L*a*b*). The Lab color model was developed to improve color representation and is indisputably the most complete color model used conventionally to describe all the colors visible to the human eye. It is a three-dimensional color space in which color differences perceived to be equally large. Each color can be precisely designated using its specific a and b values and its brightness - L. The three parameters in the model represent the luminance of the color L its position between red and green - a and its position between yellow and blue b, scaled to a reference white point. The advantage of this color space, is its device-independence and its resultant objectivity.



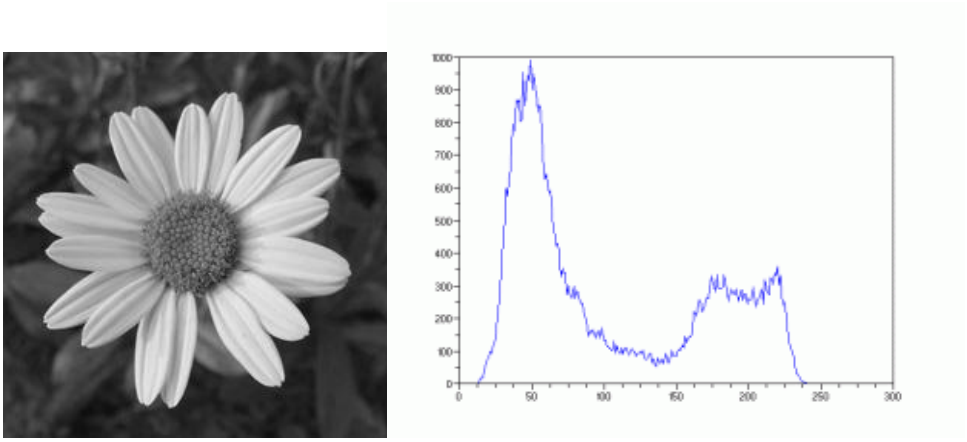
(A) RGB color space (B) YUV color space

(C) HSV color space (D) Lab color space

FIGURE 2.9: Used color spaces

2.2.4 Image Enhancement

Image enhancement has as principle objective the process of an image in such way, so that the result image is more suitable for a specific application. The word *specific* is important, because it establishes that the techniques are problem orientated; hence, methods and objectives vary with the application. The the objective of image enhancement is to improve the perceptual aspects such as, image quality, intelligibility or visual appearance. On the other hand, in applications such as object detection by a machine, image enhancement should result in an efficient machine performance. Generally, criteria for enhancement are often subjective or too complex since objective of image enhancement is application-depended. Thus, image enhancement algorithms tend to be simple, qualitative and often non-generalisable. Image enhancement approaches fall into two broad categories - spatial domain methods and frequency domain methods. While frequency domain methods are based on modifying the Fourier transform of an image, spatial domain methods, are based on pixels manipulation of an image. *Histogram Equalisation* and *Histogram Specification* are two image enhancement methods which are based on spatial domain and will be briefly introduced.



(A) Grayscale image (B) Histogram of image

FIGURE 2.10: Grayscale image and its histogram.

Image Histogram

The histogram of an image (Figure 2.10) can be obtained very simply and with minimum computation cost, offering useful information. In addition, various mappings can be applied to the histogram levels in order to emphasise certain image features in a relatively simple way.The image histogram is basically a graph that shows what proportion of pixels of an image that have a particular value (typically grey scale values between 0 and 255). We need to denote that the histogram of an image is unique, but the reverse of this statement is not true. Each image can only have one unique histogram, but several different images could be re-created from the one histogram, since the histogram gives pixel grey scale values but no indication about their location on the image. Lastly, histograms can only show one channel and can be be used to distinguish large objects with significant peaks in histogram. Geneally, histogram of a digital image with grey levels in the range [0, L-1] is a discrete function $p(r_k) = n_k/n$, where r_k is the k^{th} grey level, n_k is the number of pixels in the image with that grey level, n is the total number of pixels in the image and k = 0, 1, 2, . . . , L - 1. Therefore, $p(r_k)$ gives the probability of occurrence of grey level r_k.

Histogram Equalisation

This method provides a sophisticated method for modifying the dynamic range and contrast of an image. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. Histogram equalisation employs a monotonic, non-linear mapping which re-assigns the values of pixels in the input image such that the output image contains a uniform distribution of pixel values. Thus, if the histogram of equalisation function is known, then the original histogram can be recovered. Although this calculation is not computationally intensive, this method is indiscriminate and may increase the contrast of background noise, while decreasing the usable signal.

In Figure 2.11 we notice that for a given mapping function $y = f(x)$, the following holds:

$$p(y)dy = p(x)dx,$$

where *p(x)* is the propability density function of the given image and *p(y)* is uniform distribution. Thus, to commit equalisation need to assume that $(0 \leq x \leq 1, 0 \leq y \leq 1)$. Then we have:

$$dy = p(x)dx \Longleftrightarrow$$
$$y = \int_0^x p(u)du = F(x) - F(0) = F(x),$$

where $F(x)$ is the cumulative propability distribution of the input image, which monotonically increases.

Histogram Specification

As previously referred, image enhancement techniques are problem orientated. As a result, histogram equalisation method is quite useful, but suitable for only for some applications. This happens because this method is only capable of generating an approximation to a uniform histogram. Sometimes the ability to specify particular histogram shapes capable of highlighting certain grey level ranges in an image, is desirable. This is the process of histogram specification and the technique is developed in stages as follows:

• Find the histogram of input image $p_x(i)$ and find the histogram equalisation mapping:

$$F_x(l) = \sum_{l=0}^{l} p_x(i)$$

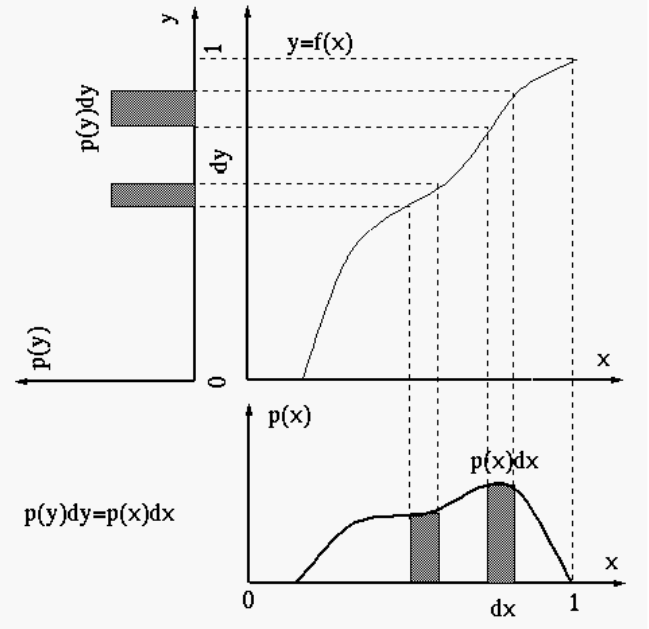


FIGURE 2.11: Histogram Equalisation

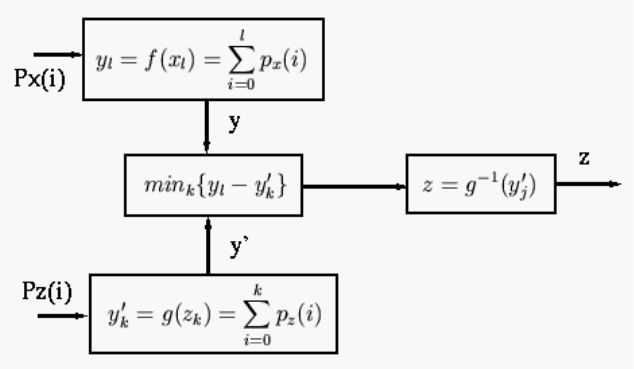


FIGURE 2.12: Histogram Specification minimisation

• Specify the reference histogram $p_z(i)$ and find the histogram equalisation mapping:

$$F_z(j) = \sum_{l=0}^{j} p_z(i)$$

• For each level l find a level j, so that $F_x(l)$ best matches $F_z(j)$:

$|F_x(l) - F_z(j)| = argmin(|F_x(l) - F_z(j))$ (Figure 2.12)

2.3 Machine Learning

The important question that is raised in the eld of Machine Learning is how to make machines able to learn. Thus, Machine Learning field tries to answer this, borrowing infrastructure from statistics, computer science, engineering, cognitive science, optimisation theory and many other disciplines of science and mathematics. Machine learning is trying to use example data or past experience to solve a problem, mimicking intelligent abilities of humans. Many successful applications of machine learning exist already, including systems that recognise faces or spoken speech, optimise robot. In addition, machine learning can be categorised in supervised, unsupervised and reinforcement learning. Focusing in the first two types, supervised learning is the type of learning that takes place when the training instances are labelled with the correct result, which gives feedback about how learning is progressing. This involves the presence of a supervisor who can tell the agent whether or not it was correct. In contrast, unsupervised learning is an easy way to assign values to actions being harder since there are no pre-determined categorisations.

2.3.1 Supervised Learning

Supervised learning is a fairly common technique that is used for learning a function from training data. This process is called classication, also known as pattern recognition. The training data consist of pairs of input objects, and desired outputs. The output of the function can be a continuous value, or can predict a class label of the input object. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples. Supervised learning generates a global model that maps input objects to desired outputs. Supervised learning is the most common technique for training neural networks and decision trees. Both of these techniques are highly dependent on the information given by the pre-determined classifications. In the case of neural networks, the classification is used to determine the error of the network and then adjust the network to minimise it, and in decision trees, the classifications are used to determine what attributes provide the most information that can be used to solve the classification puzzle. We'll look at both of these in more detail. In the classification problem, the goal of the learning algorithm is to minimise the error with respect to the given inputs. These inputs, often called the "training set", are the examples from which the agent tries to learn. But learning the training set well usually create over-fitting on the data and essentially memorising the training set rather than learning a more general classification technique. Thus, it is a challenge to find algorithms that are both powerful enough to learn complex functions and robust enough to produce generalisable results.

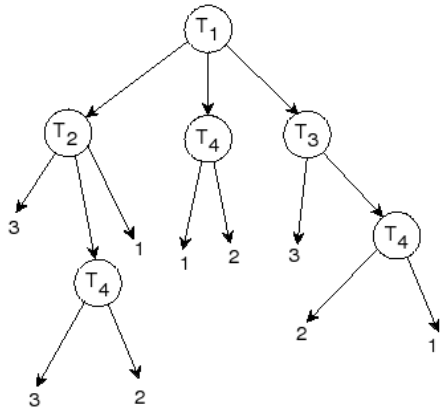


FIGURE 2.13: Decision Trees example.

Decision Trees

A decision tree is a predictive mode that trying using observations of an item to conclude about its target value. If target value is discrete, decision tree is called classification tree whereas, for continuous target value decision tree is called regression tree. In these tree structures, leaves represent classifications and branches represent conjunctions of features that lead to those classifications. The machine learning technique for inducing a decision tree from data is called decision tree learning, or decision trees. In general, a decision tree is a tree whose internal nodes are tests and whose leaf nodes are categories. We show an example in Figure 2.13 of a decision tree that assigns a class number to an input pattern by crossing the pattern down through the tests in the tree.

C4.5, developed by Ross Quinlan [4] is a prominent decision tree implementation that builds decision trees from a set of training data using the concept of information entropy. The training data consist of a set $S = \{s1, s2, ..., sn\}$ of n labelled samples. The labels are indicated by a vector $C = \{c1, c2, ..., cn\}$, whereby ci represents the class sample si belongs to. Each sample $si = \{x1, x2, ..., xk\}$ is described by a k-dimensional vector, where the xj s represent the attributes or features of the sample si . C4.5 uses the fact that each attribute of the data can be used to make a decision that splits the data into smaller subsets. C4.5 examines the normalised information gain, that is, the difference in entropy that results from choosing an attribute for splitting the data. The attribute with the highest normalised information gain is the one used to make the decision at that point. The algorithm then continues recursively on the smaller subsets. This algorithm has a few base cases, the most common base case is when all the samples in your list belong to the same class. Once this happens, you simply create a leaf node for your decision tree telling you to choose that class.

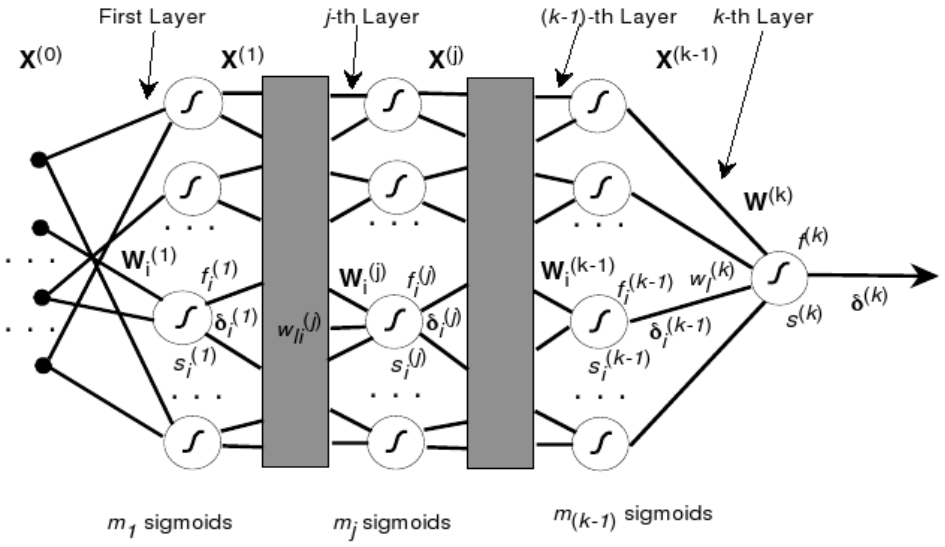


FIGURE 2.14: Neural Networks example.

It might also happen that none of the features give you any information gain, in this case C4.5 creates a decision node higher up the tree using the expected value of the class. It also might happen that youve never seen any instances of a class; again, C4.5 creates a decision node higher up the tree using the expected value.

Neural Networks

Neural networks have taken its name because the non-linear elements have as their inputs a weighted sum of the outputs of other elements—much like networks of biological neurons. Neural Networks offer two important properties in pattern recognition tasks, namely, high degree of parallelism, which allows very fast computational times and makes them suitable for realtime applications, and good robustness to disturbances, which provides reliable estimates.Neural Networks are perhaps one of the most commonly used approaches to classication, inspired by the connectivity of neurons in animate nervous systems, approximating any function mapping. A simple scheme for a neural network is shown in Figure 2.14. Each circle denotes a computational element referred to as a neuron, which computes a weighted sum of its inputs. If certain classes of hidden and output layer is a non-linear functions, the function computed can approximate any function provided.

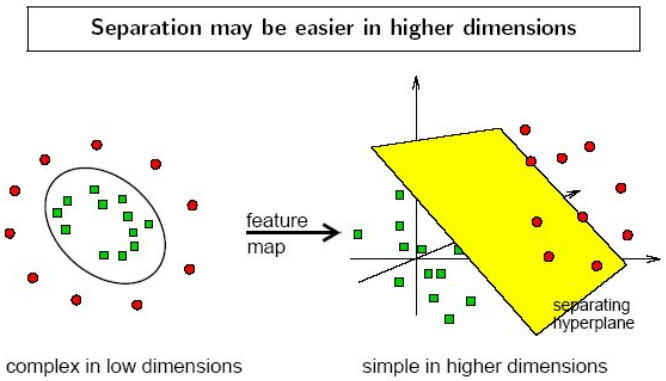


FIGURE 2.15: Support Vector Machines example.

Support Vector Machines

Support Vector Machines work by mapping the training data into a feature space by the aid of a so-called kernel function and then separating the data using a large margin hyperplane. Most commonly used kernel functions are RBF kernels and polynomial kernels. The SVM nds a large margin separation between the training examples and previously unseen examples. Hence, the large margin then ensures that these examples are correctly classied. .SVM is a useful technique for data classication. Even though people consider that it is an easy to use method. However, users who are not familiar with SVM often get unsatisfactory results at rst. A classication task usually involves with training and testing data which consist of some data instances. Each instance in the training set contains one target value and several attributes. The goal of SVM is to produce a model which predicts target value of data instances in the testing set which are given only the attributes. In Figure 2.15 we can see how SVM mapping the training data into a feature space by the aid of a kernel function, separating the data using a large margin hyperplane.

2.3.2 Unsupervised Learning

Unsupervised learning is a class of problems in which one seeks to determine how the data are organised. It is distinguished from supervised learning and reinforcement learning in that the learner is given only unlabelled examples. One form of unsupervised learning is clustering. Unsupervised learning is much harder since computer has to learn how to do something that without having any information on how to do it. In this type of learning, the goal is simply to find similarities in the training data. The assumption is often that the clusters discovered will match reasonably well with an intuitive classification. Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data.

Clustering

Clustering is a non-supervised classification method for objects which have to be classified into classes or partitions without any a priori knowledge. Clustering can be precisely stated as, once given a certain number of patterns, determining the set of regions such that every pattern belongs to one of these regions and never to two adjacent regions at the same time. One among the commonest and simplest unsupervised learning algorithms that solve the well known clustering is the *k-means clustering*. K-means tries to partition n observations into the k clusters in which each observation belongs to the cluster with the nearest mean. It is similar to the expectation-maximisation algorithm for mixtures of Gaussians in that they both attempt to find the centres of natural clusters in the data.

Given a set of observations $(x1, x2, ..., xn)$, where each observation is a d-dimensional real vector, then k-means clustering aims to partition the n observations into k sets $(k < n) S = S1, S2, , Sk$ so as to minimise the within-cluster sum of squares:

$$argmin \sum_{i=1}^{k} \sum_{x_J \in S_i} ||x_j - \mu_i||^2$$

where μ_i is the mean of S_i.

The algorithm is composed of the following steps:

• Define k points into the space initialising group centroids.

• Assign each object to the k-th group that has the closest centroid.

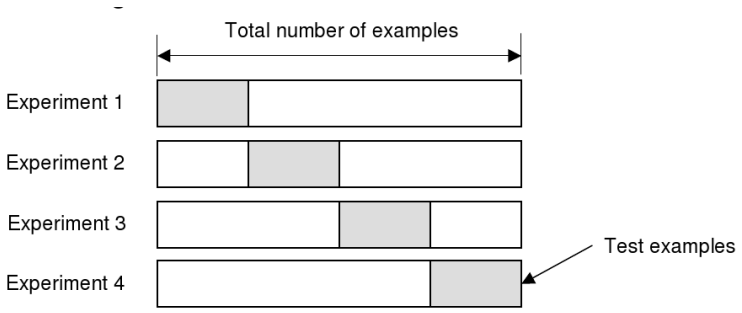


FIGURE 2.16: K-Fold Cross Validation

• When all objects have been assigned, recalculate the positions of the K centroids.

• Repeat until the centroids no longer move.

2.3.3 Validation

Validation techniques are mainly used to solve two fundamental problems in pattern recognition: model selection and performance estimation. Obviously, if we had an unlimited number of examples, these questions would have an answer. However, in real applications we only have a finite set of examples. One of the cross validation techniques that usually comes to solve this problem is the k-fold method. In K-fold cross-validation, the original sample is partitioned into K subsamples, from which a single subsample is retained as the validation data for testing the model, and the remaining K 1 subsamples are used as training data. The cross-validation process is then repeated K times, with each of the K subsamples used exactly once as validation data (Figure 2.16). The K results from the folds can be then averaged to produce a single estimation. Thus, the estimation can be computed as :

$$E = \frac{1}{k} \sum_{i=1}^{k} E_i.$$

The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. The advantage is that you can independently choose how large each test set is and how many trials you average over and that all observations are used for both training and validation process.

CHAPTER 3

Problem Statement

3.1 The Need For Color Recognition in RoboCup

RoboCup Standard Platform League (see 2.1.1) is a color-coded environment, where colors are used to define principal objects (see 2.1.1). Thus, it is necessary to state the objective and the essential specifications that will help in problem comprehension. To begin with, the primary objective of this thesis copes with the creation of a robust and efficient *color recognition* method that will be applied within the RoboCup Standard Platform League. Color recognition, which the last years has met great success and significant development, is typically the first step of the vision system of robots playing RoboCup soccer, offering essential information used for object recognition.

Generally, object recognition in machine vision is the task of finding a given object in an image or video sequence. Thus, in vision system of robots playing RoboCup soccer, object recognition is a fundamental task. Furthermore, in such robotic vision systems there are certainly many methodologies to solve this problem, depending on application specifications. However, in problem domains such as Standard Platform League the sequence of techniques that are used for object recognition are applied in a pipeline architecture. The first stage of this architecture involves color recognition, a fundamental part of object recognition, where each pixel in the image is labelled as

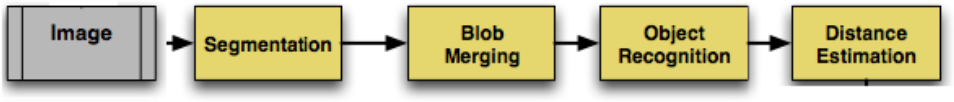
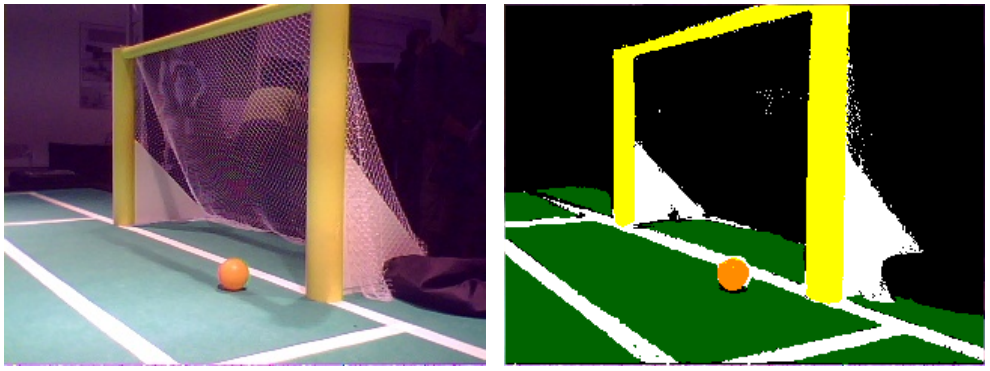


FIGURE 3.1: Vision pipeline



(A) Original image (B) Classified image

FIGURE 3.2: Resulted image after color recognition.

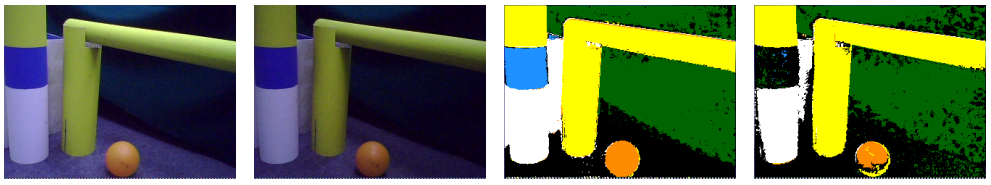
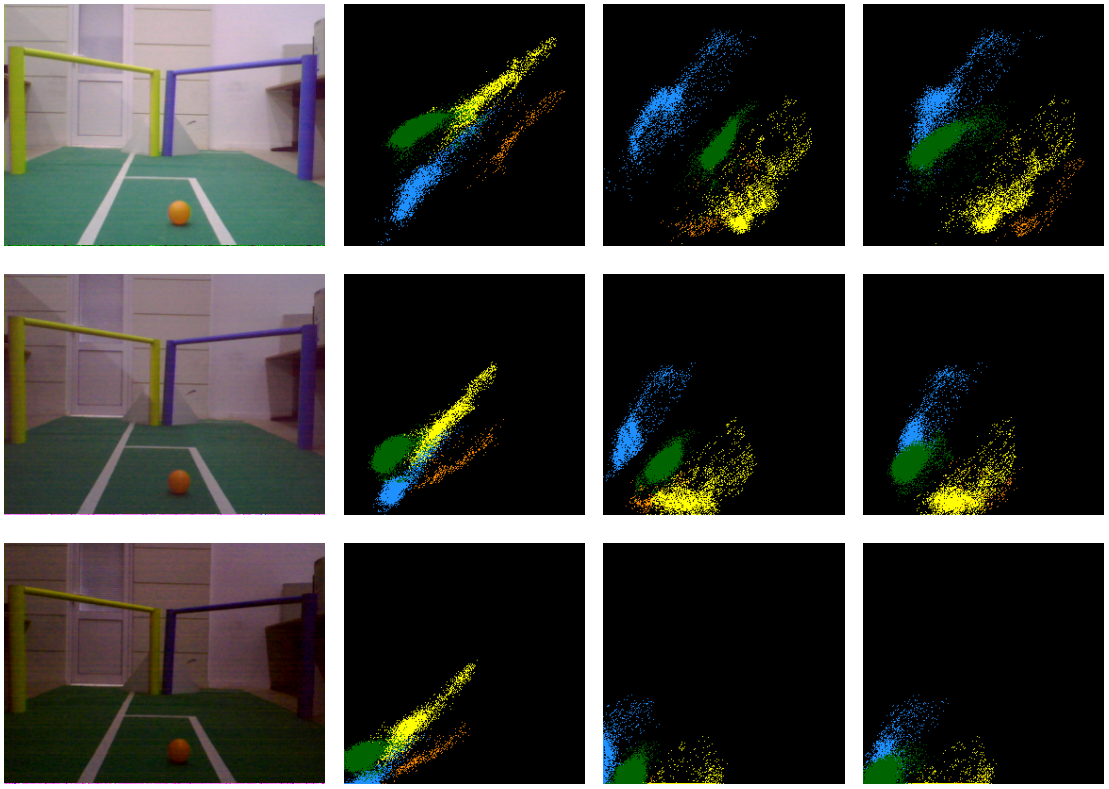


FIGURE 3.3: Color recognition under two levels of illumination.

one of a set of color classes. The resulted image passed in second stage, where blob formation is committed, trying to perform ball and goal detection , forming, sorting and combining the given blobs. This pipelined architecture can be seen in Figure 3.1.

Color recognition is a color-based segmentation approach that has as principle objective to recognise colors of an image on the basis of similarity of known color characteristics. Generally, color-based segmentation techniques range from color classification, to clustering using region based methods, to object separation using edge-based methods. The last two techniques are color segmentation methods that use color to separate objects without a priori knowledge about specific surfaces, whereas the first one attempts to recognise colors of known color characteristics. Since, color classification is the first stage in the software system, any error in classification has a ow on effect to the entire system. Thus, it has to be robust, giving the right results under nominal conditions and under slightly different lighting conditions. In Figure 3.2, the left image obtained by Nao's camera whereas in the right image, we can see the resulted image after the method application.



(I) Original Image (J) rg-plane (K) rb-plane (L) gb-plane

FIGURE 3.4: 2D representation in RGB color space under three illumination levels.

Varying illumination conditions can directly affect color recognition methods raising one of the biggest challenges for vision systems in mobile, autonomous robotics that is adaptivity to changing visual conditions. In addition, the majority of robotic platforms use color segmentation as the basis for vision, despite this technique's inefficiency in cases where lighting environment in which calibration was occurred, slightly changed. In particular, the nature of a mobile platform with cameras on board, such as Nao, suggests that lighting conditions will change. Indeed, many experiments in the RoboCup competition have shown that even the smallest lighting changes can significantly affect the performance of color segmentation methods. Thus, vision systems need to be calibrated many times over the day, even on the same field, because of the variance of light in the room. Since no additional information is provided about illumination conditions, most of the attempts to resolve the problem involve geometric information related to the objects of interest. For instance, ball recognition can be achieved using information related to its round shape. Nevertheless, such computationally demanding techniques are not suitable for robotic soccer, since it is a real time, non-deterministic task, on a platform with standard specifications. To show the problem of varying illumination levels, we have labelled the same image pixels under three illumination levels. In Figure 3.4 we can see that locus of selected pixel is shrinked increasing the classes overlaps.

3.2 Related Work

A number of new methods [5–12] have been developed to find ways to extend color segmentation, to real time, non-deterministic tasks such as robotic soccer. Although these methods present great success and significant development, each suffers from one or more disadvantages. Generally these methods can be classified into the following major categories:

- Threshold-based methods
- Edge-based methods
- Region-based methods
- Hybrid methods

3.2.1 Threshold-based methods

Many teams in SPL domain use this kind of methods with slight modifications to increase its performance. Threshold techniques are quite simple having low computational requirements. Nevertheless, threshold techniques need a careful tuning of thresholds being inefficient under varying illumination conditions. Generally, threshold techniques rely on the assumption that pixels that belong to the same class or object, have color values that lie within a certain range. Thus, this assumption can be used in color images using separate threshold for tristimulus components of the image and then combining them. Furthermore, in threshold techniques are often used the HSL and the HSV color models, due to its correspondence with the way that people understand color. Under varying conditions, the inefficiency of these techniques can be overcomed using classification methos such as multivariate decision trees and neural networks.

3.2.2 Edge-based methods

Edge-based methods are closely connected with edge detection techniques. Since, region boundaries and edges are closely related, pixel values change rapidly at the edge between two regions. This kind of methods are very computationally expensive because the major edge detection techniques such as Sobel, Canny and Susan not only use first-order or second-order derivative, but are also used in conjunction with some edge linking procedures in order to obtain the closed region boundaries.

3.2.3 Region-based methods

Compared with the edge-based methods, region-based methods rely on the assumption that adjacent pixels that belong to the same class or object have similar color values. This similarity criterion is called homogeneity criterion, and is usually based on a threshold value. In seeded region growing methods there is a set of seeds as input along with the image which can be selected manually or the automatically. The seeds mark each of the objects to be segmented. The regions are iteratively grown by comparing all unallocated neighbouring pixels to the regions. The difference between a pixel's intensity value and the region's mean, , is used as a measure of similarity. The pixel with the smallest difference measured this way is allocated to the respective region. This process continues until all pixels are allocated to a region. In addition, there are also unseeded region growing methods which do not require explicit seeds.

3.2.4 Hybrid methods

There have been some hybrid methods developed that combine two or more techniques, aimed at achieving a particular objective.

3.2.5 Existing Approaches

MiPal Team

Lovell, Nathan Hains and Estivill-Castro, Vladimir [5] propose a feasible solution applying in not too widely variable illumination conditions, such as RoboCup Conditions. This technique is robust to changes in color intensity and temperature, identifying objects on real time complex and dynamic lighting environments. This technique consists of two major stages creating a pipeline image processing architecture. The first stage includes a sparse classifier which identifies those pixels values that remain on the same color class across a wide variety of illumination conditions labelling the rest of color values as unknown. In the second stage an optimised edge-detection algorithm is implemented, which finds the border of objects without performing complete edge detection.

Sparse classification along with robust classification comprise the two types of calibration in classification. Robust classification rely on that if a pixel with values (x, y, z) is recognised as orange in any of the images in the training set, then the classifier should label this pixel as class orange for any future images. This may not be possible due to that even the same (x, y, z) tuple may be assigned to two different classes in the training set even within the context of the same image. Furthermore, humans define color taking into account the surrounding context of the image.

Thus, an ideal classifier would classify a pixel as class orange only when the predominant color of the surrounding is orange. However, color classification algorithm does not yet know the surrounding of the pixels, since color recognition is usually the first step in object recognition. The solution for this, is *sparse classification* which recognises those values (x, y, z) that remain on the same class across a wide variety of illumination conditions, ignoring pixels that could fall into more than one class depending on surrounding context.

Optimised edge detection relies on the assumption that if we can identify some points inside interesting objects using sparse classification, it is then possible to use edge detection to locate the edges of that object and use them for feature extraction. Thus, assuming that we have identified a pixel that is contained within an object, we use this point to the extract the boundary of this object using optimised implementations of Sobel and Canny edge detectors. However, it is not always necessary to find the complete edge of each object, since we are only required to know three points on the edge to find the correct parametrisation of a circle .

SPQRL Team

Luca Iocchi [6], team leader of the SPQRLegged RoboCup team of the University of Rome, ″La Sapienza‟, in Italy, proposes an adaptive color recognition technique robust to noise and light changes. This method produces an adaptive transformation of the color distribution of an image, having as objective the modification the color distribution in such a way that the subsequent use of prior manually fixed thresholds can be effective for color segmentation. Especially,for a given color space C the adaptive transformation can be represented as a function $\tau : C \rightarrow C$, calling τ-distribution the new color distribution obtained by applying the function τ to the color distribution of an image, and τ-transformation the operation of computing the τ-distribution of an image. The estimation of function τ is achieved by computing periodically the τ-distribution. The resulted distribution has a noticeable distance between the peaks increasing the method efficiency without requiring fine tuning for the thresholds, making also the method not sensible to small variations. In addition, the computation of function τ is not obligatory for every image frame , because variable light conditions do not affect this transformation. Thus, once τ is computed, it can be used to correct the image distribution producing τ-distribution in which fixed thresholds will be applied for color segmentation. Several strategies can be used to determine when it is convenient to recompute such as a degree related to the light conditions stability. Furthermore, color distribution of an image is computed by analysing only a subset of the points that sited around and below the horizon line in the image. Finally, the estimated function computes a static look up table (or color table) that is used for color classification with a single memory access for a grid of image.

TeamChaos Team

TeamChaos RoboCup team [7] constitutes a multiple collaboration which mainly includes Spanish Universities. TeamChaos approach is a hybrid method that relies on color segmentation that integrates the thresholding and the seeded region growing(SRG) methods. Since seeded region growing methods need a set of seeds as input, threshold technique can be used to generate it for each color of interest, and then use SRG to grow color regions from these seeds. This integration of the two methods improves robustness to changing lighting conditions due to conservative thresholding, and robustness to blurred edges due to conservative homogeneity criterion in SRG. The seed rigion growing algorithm consists of two major stages. In the first stage a set of colors to be recognised is fixed and threshold technique is used to produce a labelled image with pixels that belong only to one of these colors. Afterwards, each labelled pixel is used as an initial seed for a region of that color. Hence, a blob starts growing on each one of these seeds, including adjacent pixels. Each of these blobs stops growing due to homogeneity criterion that embodies the notion of a significant discontinuity: The choice of the homogeneity criterion is critical in SRG techniques because if it is too strict, the blob growing can stop prematurely because of local variations whereas if it is too liberal, the blob can ood to an adjacent region though a weak edge.

Numanoids Team

The Numanoids RoboCup team [8] is the result of the coordination between University of Newcastle, Australia and National University of Ireland, Maynooth(NUIM). Their color recognition method is based to pixel classification using additional ´soft´color classes that reduce the risk of false positive classification. *Soft color classification* is a modification of sparse classification; hence, Soft color is used to classify pixels that is less or more saturated due to lighting conditions in a ´soft´color class. As a result, the decision about what object the soft color belongs to, is delayed until the entire image is processed, reducing the false positives and allowing true color to be classified as the shades of color that are highly saturated. Thus, object decisions are based on the presence of true color and then soft color can be used for additional size information.

Austin Villa Team

Prof. Mohan Sridharan at Texas Tech University and Prof. Peter Stone [9],who are the team leaders of TT-UT Austin Villa Robcup team of the University of Texas at Austin, present an approach that works in real-time to achieve color constancy on mobile robots in the RoboCup domain. Their technique is applied in three discrete illumination conditions (bright, intermediate,and dark) resulting an efficient algorithm.

This method relies on the usage of color space distributions that are used as an illumination metric along with easy to train color cubes. Furthermore, assuming that images from the same lighting conditions would have measurably similar distributions of pixels in color space, a color distribution per image was created to the normalised RGB space using the (r, g, b) pixel values that by definition are:

$$r = \frac{R+1}{R+G+B+3},\ g = \frac{G+1}{R+G+B+3},\ b = \frac{B+1}{R+G+B+3} \text{ and r+g+b=1}$$

Thus, creating a color cube and color distributions for each of three illumination condition, it was used KL-divergence, a popular measure for comparing distributions, to enable the robot to recognise and adapt its current color cube to the appropriate one. As new images are processed, one is periodically tested for membership in one of the three illumination class, using a fixed number of training samples from each class for comparison. The distribution obtained from a test image is compared with the training samples using the KL-divergence measure, to determine the training sample that is most similar to it. The test image is then assigned the same illumination class label as this training sample. If sufficient number of consecutive images are classified as being from an illumination condition, the robot transitions to another color cube (representing the new illumination condition) and uses that for subsequent operations.

German Team

The German Team [10] is a collaboration of the Humboldt University of Berlin, the University of Bremen and the Technical University of Darmstadt. Their vision system uses three layers of color precision where each of them has its own method to assign colors to color classes. Green of the carpet is used as reference color, producing a subcube of upper and lower bounds in the three dimensions. In addition, other colors are defined in relation to this cube. For example, yellow is defined as having a U and V values lower than the upper bound of the reference cube. However, this method is not possible to separate colors that have similar color values such as yellow and orange, grouping them into one color class. Despite this disadvantage, this method is applicable due to the simplicity of auto-adaptation of the reference color and the segmentation of the color cube.

Especially, the first layer is the layer with the lowest precision representing classes green1, white1, and black1 using a cuboid representation. Moreover, for a set of images that have been taken under similar illumination conditions, green1 and white1 calibration is achieved using the average intensity for each channel taking into consideration the minimal and maximal value of each image.

The second layer comprises the next layer of color precision distinguishing between more colors. Similar to the first layer, green2 is used as a a reference color defining the yellow2 , skyblue2 , and orange2. Nevertheless, colors such as yellow2 and orange2 can be overlapping in color space, which means that some colors can be misclassified. The problem of misclassification is solved by the third layer, which uses a color look-up table to represent white3, green3, yellow3, skyblue3, and orange3 . This look-up table is constructed by identified segments that belong to a certain object for sure based on the layer 1 and layer 2 color classes and on spatial constraints for the position of the segments in the image. Under varying illumination conditions the positions of all colors move along the three axes and the intensities are amplified or weakened. However, relative position of the colors to each other remains as is. Thus, the vital task of updating the reference cube to the new lighting conditions can be achieved by extracting from each image a set of green pixels and recalculating the reference cube. Especially, green pixel extraction is done watching changes in the YUV channels on a scan line. For example, a drop in the luminosity channel Y occurs when the color changes from the white of the border to the green of the field.

Canuck Team

Team Canuck is the RoboCup team hailing from the University of Alberta located in Edmonton, Alberta, Canada, composed mostly of Computing Science and Electrical Engineering graduate students [11]. They present a method that uses the Gaussian mixture model (GMM) of two components in the YUV space to model the distribution of a color class, according to the dichromatic color reectance mode. In addition, the GMM is adapted to temporal lighting variation with an exponentially decaying function. The initialisation of GMM is achieved by using the standard Expectation-Maximisation(EM) algorithm applied to classified color pixels, while the color model is iteratively updated over time. This procedure consists of two major calibration stages: the offline stage,in which the initialisation of Gaussian mixture model takes place, and the on-line stage,in which the learned color model is updated over time.

The offline color calibration step, is a supervision task which produces the Gaussian mixture model that is used to initialise the color recognition system. The calibration step starts when user selects the pixels of interest on an image using an interactive tool. Afterwards, the selected pixels initialise a color tracker which follows the movement of objects of interest using the selected pixels as seeds in a region growing process to include neighboring pixels of the same color. The selected pixels along with their locations are used to update the position of the tracking window in the consequent image frame. As a result, the robot can move around over the field, meeting a large amount of spatial variation of illumination, collecting sample pixels that are used to derive the GMM of two components using a standard EM algorithm.

From the other side, the on line color calibration starts when a game starts. While pre-calibrated color models are used, an adaptation procedure is executed to update the color models in real-time due to dynamic lighting. Specifically, color classification is performed on an image using the current look-up tables (LUT). Meanwhile, region growing is applied to include the neighboring pixels within a threshold color distance reducing false negatives due to spatial or temporal variation of illumination.Consequently the measured data can update more accurately the model. The evaluation of the GMM is done once every 20 frames since it is unnecessary to re-evaluate the GMM for each frame unless the illumination condition goes through a rapid change. However, the benefits of re-evaluating the GMM after a time interval is the efficiency and the provision of a more complete set of sample pixels that capture the spatial variation of illumination.

UChile1 Team

The U-Chile-1 RoboCup team is an effort of the Department of Electrical Engineering of the University of Chile composed by a group of undergraduate and graduate students of electrical engineering and computer science [12]. They propose a method in which the color space mapping idea is used to propose the remapping applied to color classes instead of pixels. Their method starts defining a class-relative in which classes are represented in terms of their spatial relation to a base color class. Afterward, the class-relative color spaces are used as starting points to remap the remaining classes. In addition, the remapped classes are used in object detection generating a complete color look-up table from scratch, and adapting quickly to severe lighting condition changes making a smooth transition from the remapped estimations of the color classes to trained estimations of the classes. Finally this method does need to solve the natural ambiguity in color classes intersections due to concept of soft-colors.

3.2.6 Observations and Conclusions

So far, we have described eight different methods that try to solve the problem of color recognition including the aspect of varying illumination conditions. Nevertheless, we need to outline the strengths and weaknesses of these methods in order to receive an evaluation and efficiency degree on each of these.

• **MiPal Team**

MiPal Team has created an adaptive method that operates under wide variety of illumination conditions. First stage of this technique includes a sparse classifier, which identifies those pixels values that remain on the same color class across this variety of illumination conditions, labelling the rest of color values as unknown.

In the second stage, an optimised edge-detection algorithm commits object recognition, performing partial edge detection. Nevertheless, this object recognition does not follow a modular architecture since is based on the optimised edge-detection algorithm that is specialised on object specifications. Thus, even small changes in objects may cause an inefficiency of object recognition architecture.

• **SPQRL Team**

SPQRL Team has created an adaptive method of color recognition not sensible to small lightning variations. This method tries to produce an adaptive transformation of the color distribution of an image, modifying the color distribution in such a way that the manually fixed thresholds can be effective for color segmentation. In addition, method does not require fine tuning for the thresholds of the resulted distribution since there is noticeable distance between the peaks on the resulted distribution. However, this method takes as input the image histogram of each frame making a computationally demanding technique.

• **TeamChaos Team**

TeamChaos Team has created an hybrid method that integrates the thresholding and the seeded region growing methods. This seeded region growing method first acquires the prerequisite input seeds, generated from threshold technique and then uses SRG to grow color regions from these seeds. Furthermore, conservative thresholding improves robustness to changing lighting conditions whereas conservative homogeneity criterion improves robustness to blurred edges. Even though, a conservative homogeneity criterion can stop the blob growing because of local variations whereas a liberal homogeneity criterion can the blob can make he blob growing algorithm to ood to an adjacent region though a weak edge.

• **Numanoids Team**

Color recognition method of Numanoids Team is based to pixel classification using additional ´soft´color classes reducing the risk of false positive classification. Nevertheless, any additional information about method adaptivity on illumination conditions is not provided.

• **Austin Villa Team**

Austin Villa Team method relies on classification techniques. In addition, this method can operates under varying illumination condition since they have developed a method that is based on the usage of color space distributions that are used as an illumination metric along with easy to train color cubes.

Thus, assuming that images from the same lighting conditions would have measurably similar distributions of pixels in a color space, a color distribution per image was created to determine and adapt to three discrete illumination conditions. In addition, this illumination metric helped them to indicate what of the training color cubes would be used to perform color recognition. This method is robust and efficient under a wide range of illumination condition increasing the human effort for a color cube creation per illumination.

• **German Team**

German Team has created a complex vision system that use three layers of color precision each of these has its own method to assign colors to color classes. Green of the carpet is used as reference color, whereas other colors bounds are estimated in relation to the computed bounds of green. However, colors that have similar color values such as yellow and orange is is difficult to be separated. Despite this, method is simple and adaptive under different illumination conditions in which positions of all colors are amplified or weakened. However, relative positions of colors remain, enabling the all color bounds update through reference cube bounds recalculation.

• **Canuck Team**

Canuck Team presents an adaptive method that uses the Gaussian mixture model of two components in the YUV space to model the distribution of a color class, according to the dichromatic color reectance mode. The GMM is adapted to temporal lighting variation with an exponentially decaying function. The initialisation of GMM is ac hived by using the standard Expectation-Maximisation(EM) algorithm applied to classified color pixels, while the color model is iteratively updated over time. Similarly to SPQRL Team this method is computationally demanding; hence inappropriate for a real-time task.

• **UChile1 Team**

UChile1 Team propose an adaptive complex method that uses color space mapping idea that is applied to color classes instead of pixels. Their method defines a class-relative representation in terms of their spatial relation to a base color class. Class-relative color spaces are used as starting points to remap the remaining classes, generating a complete color look-up table from scratch, and adapting severe lighting condition. In addition, this method use concept of soft-colors to solve classes overlaps.

CHAPTER 4

The Proposed Approach

This chapter, introduces the Kouretes Color Recognition Method, an approach for real time color recognition. Implementation considerations and efficiency analysis , along with two alternative considerations of this method, are discussed indicating both the similarities and the essential differences among them.

4.1 Method Description

Kouretes Color Recognition Method passes through four stages in order to achieve its principle objective;color recognition of an image. These stages are the following; Images Capturing-Selection , Training Data Selection, Training Process, Segmentation Process (Figure 4.1).

4.1.1 Images Capturing-Selection

The first major stage of our method is "Images Capturing-Selection". As it can been easily seen in Figure 4.1, this stage consists of two parts: image capturing in blue blocks and image selection in pink blocks. Each of them is substantial part of our method, since the selected images of first part will be supplied in the next stage of "Training Data Selection". Thus, not only the quality but also the quantity of selected images are vital in the creation of robust and efficient classification function.

Image capturing includes the process of taking pictures from the two integrated cameras on the robot's head. Although, it is a quite simple process, the classes of interest

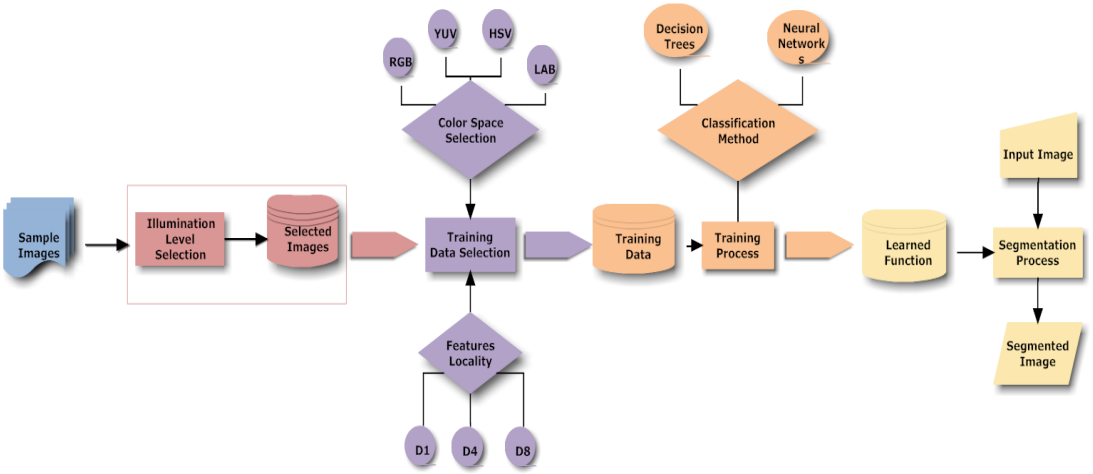


FIGURE 4.1: Kouretes Color Recognition Method

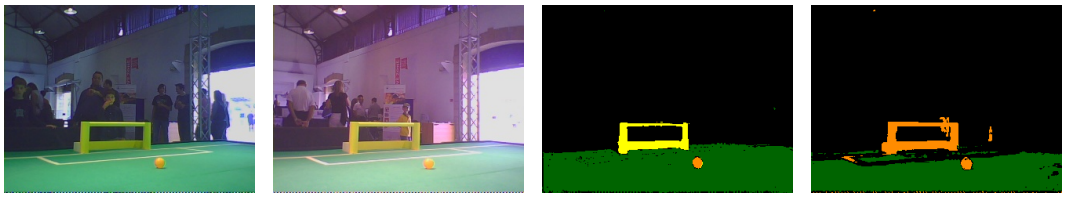


FIGURE 4.2: Original Images and Segmented Images taken under the same illumination condition using auto white balance configuration.

must be included in prospective images along with an abundance of lightning conditions, since the lightning of the field is non-uniformly diffused. Furthermore, the user needs to increase his effort in configuring camera settings since camera gain, exposure, and white balance affect image quality. Thus, configuration accomplishment of these camera's parameters must improve robustness of our classification method, decimating the rate of the misclassified data.In our preliminary work, we used the auto configuration settings, since the auto-adjustment in gain, white balance, and exposure helped in improving the image quality, while eliminating the problem of the non-uniform illumination. However, we noticed that the auto white balance changes the locus of our color classes in the 3-dimensional color space causing problems to our classification method. Thus, we currently manually configure the white balance, the gain, and the exposure for best results. In Figure 4.2 we see the same picture taken under the same illumination conditions using the native automatic configuration method. The first picture was taken when the automatic method has been finished whereas the second one was taken during the process of configuration. As a result, our classification algorithm stands inefficient in this situation.

In image selection, the user must select the captured image that will include the set of colors classes to be recognised. For RoboCup SPL games these classes are orange (ball), skyblue (goal posts), yellow (goal posts), white (lines), green (field), red (team color), and blue (team color). While our color segmentation method currently accommodates all eight colors above, we occasionally restrict focus only on the first three colors (orange, yellow, skyblue), which correspond to the key objects in the field; all other colors are grouped under the *NoColor class*.

4.1.2 Training Data Selection

Although "Images Capturing-Selection"can be completed using an pre-existing software, the accomplishment of next method stages necessitate a semi-automated calibration system. Particularly, after selected imaged loading, a process of image labelling will be followed. Image labelling includes training data selection using efficient selection techniques that enables user to collect an abundance of pixel samples in a efficient way minimising the human effort. Finally, the process of selected training data extraction need to be configured optimising global method efficiency. Although, image pixels can be labelled into one of eight classes, focus occasionally is restricted only on the first three color classes (orange, yellow, skyblue), requiring human effort to label additional training data. In Figure 4.1 training data selection along with method configuration options can be seen in purple stages.

4.1.3 Training Process

A color class is a set of all colors that can be observed in pixels corresponding to an object having a known color, enclosing all variations of a certain color of interest in the real world. Thus, the"Training Stage"of our approach uses classification methods such as decision trees and neural networks that rely on the fact that if a pixel with values (x, y, z) is classified in a specific class in any of the images in the training set, then the classifier should label this pixel in the same class for any future images. Generally, classification technique evolves from a given set of sample pixels with known color class to a learning function that assigns a color class to pixel values. Thus, assuming that the tristimulus space XYZ is used for color representation, then a color classifier can be represented as a function

$$color\_class : X \times Y \times Z \rightarrow colors$$

that given a triplet *(x, y, z)* produces a color class. In addition, this function may be easily represented as a single characteristic function for each member of colors.In Figure 4.1 training process can be seen in orange stages.

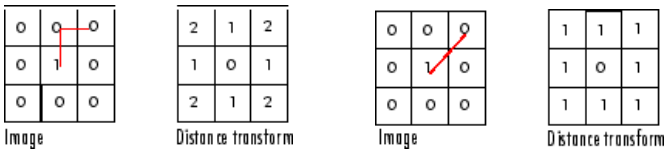


FIGURE 4.3: N5 and N9 schemes respectively.

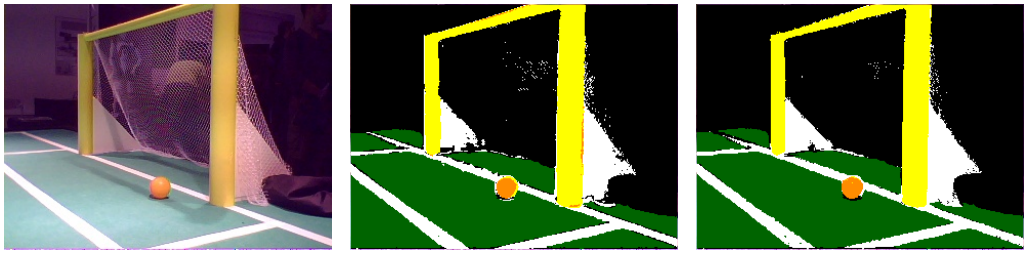


FIGURE 4.4: Original image and color segmented images (N1 and N5 schemes).

4.1.4 Segmentation Process

Last but not least, segmentation process comprises the accomplishment of principle objective:color recognition of an image. Hence, color recognition of an image requires the feed of classification function with pixel values, acquiring and visualising the classification results.

4.2 Method Configuration

Our classification method has been implemented in such a way that the color space, the number of input attributes of the training data, the classifier model, and the training algorithm can be easily modified. With this parametric design, we are able to test various configurations and choose the most appropriate.

4.2.1 Neighbourhood

To begin with, the simplest choice of the input attributes that training process will use, is the values (x, y, z) of the current pixel (N1 scheme). However, the non-uniform diffused lightning on the field along with the complex environment background (crowd) can create difficult situations that can be overcome, exploiting color locality by using as input attributes not only the values of the current pixel, but also the values of pixels in its immediate "neighbourhood". Thus, these can be the 4 orthonormal (N5 scheme) or the 8 adjacent pixels (N9 scheme).

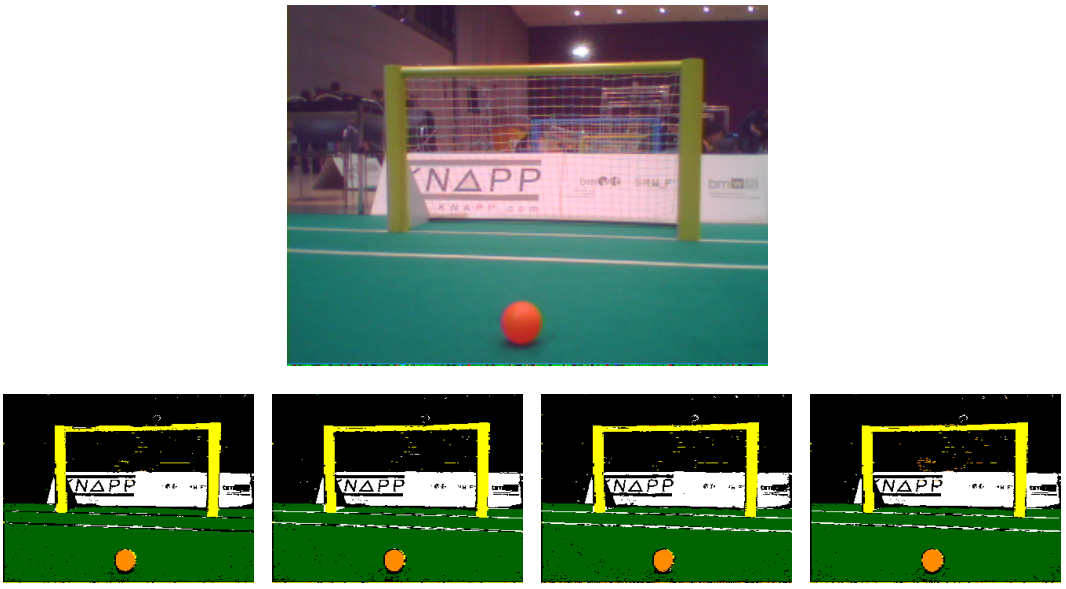


FIGURE 4.5: Original and segmented images (using RGB, YUV, HSV, LAB color spaces.)

Nevertheless, through extensive experimentation (see 6) we found that the neighbourhood of the 4 orthonormal neighboring pixels works best for our purposes (see Figure 4.4). In particular, if we label a pixel with values (x, y, z) in a class using the N5 scheme, then the extracted training data will contain 15 attributes (current pixel values + 4 orthonormal pixels values). Similarly, in N9 scheme the extracted training data will contain 27 attributes (current pixel values + 8 adjacent pixels values). In Figure 4.3a we can see the specification of N5 scheme and in Figure 4.3b we can see the specification of N9 scheme.

4.2.2 ColorSpace

There is a variety of color spaces that are usually used in image processing, including RGB, YUV, HSV and LAB. However, the choice of a color space for classification depends on several factors including which is the native format of camera hardware. Thus, the default format that the two intergated cameras of Nao provide is the YUV422. In that format the U-channel represents the blueness, the V-channel represents the redness whereas, the Y-channel represents the brightness of the pixel, having the advantage that chrominance is coded in two dimensions while intensity is coded in the third. Thus, YUV and HSV color spaces have the advantage in threshold techniques due to the face of variations in the brightness of illumination. In addition, our method uses classification techniques enabling the usage of all these color spaces. However, using the native format provided by hardware, we can earn the performance penalty of a software color space transformation. In Figure 4.5 we can see the results of color recognition applied in RGB, YUV, HSV, LAB color space using N5 scheme.

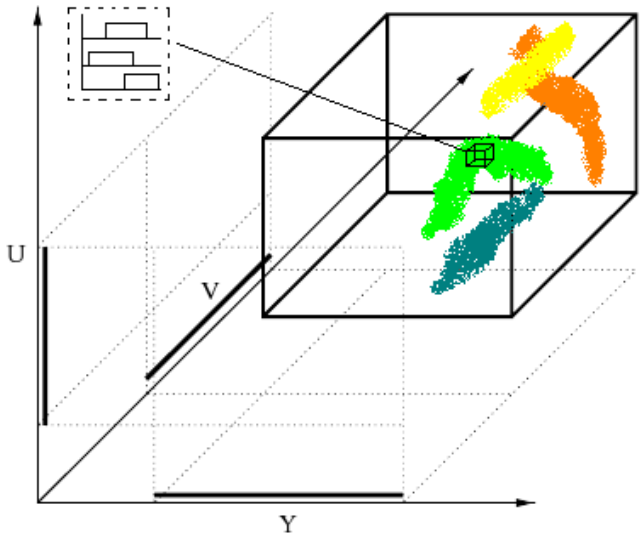


FIGURE 4.6: Decision trees rules formation

4.2.3 Classification Method

For color classification we use modern classification technology. We have mainly tested decision trees (DT) neural networks (NN) and support vector machines (SVM), but decision trees were found to be faster and more robust for the successful classification of colors. For training a decision tree we use the C4.5 algorithm, often referred to as statistical classifier, developed by Ross Quinlan. C4.5 builds decision trees from a set of training data using the concept of information entropy. Once training is finished, a decision tree whose leaf nodes determine the appropriate class is created. Especially, C4.5 produces a set of rules with this formation:

Rule #:
{
    Attribute $\{<, >, \leq, \geq\}$ # {0 - 255}
    .
    .
    .
        class: # {1,2...8}
}

Moreover, the set of produced rules can be seen as a set of thresholds that is bigger and more complex than the threshold techniques one 3.2.1. In Figure 4.6, we can see that C4.5 tries to solve the problem of classification producing a complex set of rules in YUV color space.

4.2.4 Exported Rules Formation

Since the extact formation of produced rules is known, we can use it by applying to them a modification, creating a compatible version that can be easily intergrated in Kouretes Vision System (KVS). Two different approaches can be followed giving as two different ways of integration of exported rules in KVS. Approaches implementation considerations and efficiency, are introduced.

• Simple Rules Formation

The first approach includes the creation of a number of conditional statements equal to the classification method's exported rules. Each one of these conditional statements is produced analysing each rule of the decision tree. The conditional statements' formation is:

if (Attribute $\{<, >, \leq, \geq\}$ {0 ... 255} && . . .)
return: {1,2 ... 8}

This transformation of native rules can be easily achieved using by flex lexer analyser due to the known formation of exported decision trees rules. Nevertheless, in complex environments a large set of rules can be created, requiring many conditional branches to determine membership in one color class for each pixel. This can be especially inefficient on pipelined processors with speculative instruction execution. In addition, each time that a new set of classification rules is produced, the KVS needs to recompiled.

• Complex Rules Formation

To overcome these problems we create a more complex version of rules that uses a boolean valued decomposition of the multidimensional threshold. To be more specific, we create one decomposed representation per class, each of which consists of a set of array elements equal to axes size in the space . Moreover, these arrays elements take a value in these positions where the threshold rules classify a specified class (see Figure 4.7). Thus, each pixel's class membership can be computed by applying the bitwise AND on each of the elements of each array:

class_membership = Component[x] AND Component[y] AND Component[z]

The result indicates whether the pixel belongs to the class or not. This approach can be implemented as a few array lookups per pixel that will be extensively introduced later.Thus, the classification operation is much faster than the naive approach because the bitwise AND cost less than an integer comparison on modern processors.

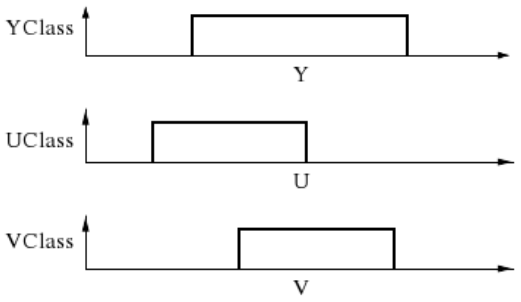


FIGURE 4.7: Decomposed representation of thresholding rules

This method requires a to split its componet of a tristimulus color space in 256 levels. However,in order to illustrate the approach, we split each dimension of the XYZ color space into 10 levels. Thus, a supposing class A can be represented by assigning the following values to the elements of each array:

$$\text{XComponent}[\,] = \{0,0,0,1,1,1,1,1,1,1\}$$
$$\text{YComponent}[\,] = \{0,1,0,1,1,0,0,1,1,1\}$$
$$\text{ZComponent}[\,] = \{1,0,0,0,1,0,0,0,1,1\}$$

Afterwards, we consider a pixel with values (4, 5, 9). If we want to check if this pixel is a member of class A we need to evaluate the expression Component[4] AND Component[5] AND Component[9], which in this case is 1 in binary representation. In addition, we can determine the participation of a pixel in multiple color classes simultaneously. This can be achieved combining the different arrays of all classes creating a region of three arrays. Each of these arrays elements will include the array element of all classes arrays. However, we decided to use this alternative representation in a large number of threshold rules representation simultaneously instand of multiple color classes determination. Thus, if we have one more rule that can be represented as follows:

$$\text{XComponent}[\,] = \{0,1,1,0,0,0,1,1,1,1\}$$
$$\text{YComponent}[\,] = \{1,1,1,0,1,1,1,0,0,0\}$$
$$\text{ZComponent}[\,] = \{1,0,0,1,0,1,0,0,1,1\}$$

the representation will become the following:

$$\text{XComponent}[\,] = \{00,01,10,01,01,11,11,11,11,11\}$$
$$\text{YComponent}[\,] = \{01,11,01,10,11,01,01,10,10,10\}$$
$$\text{ZComponent}[\,] = \{11,00,00,01,10,01,00,00,11,11\}$$

Finally, we achieve to evaluate pixels participation in multiple rules at once, applying the AND operation to each array element. As a result, evaluating the expression Component[4] AND Component[5] AND Component[9] we take 10 in binary representation. This means that there is at least one rule that classify this pixel in class A. Additionally, due to the small size of the color class representation, the algorithm can take advantage of memory caching effects. Nevertheless, we need to create a region of arrays per class since, we can not check the multiple pixels participation simultaneously. However, the upper bound of AND operation is limited to a class number that in our case is eight. The algorithm that is used to classify a pixel on class is described as follows:

Algorithm 1 Decision List Classification

```
INPUT: XComponent[8][256], YComponent[8][256], ZComponent[8][256]. The pixel
values to classify are (x,y,z).
OUTPUT: The color class of (x,y,z)
y=0
class
while y=0 do
  class++
  y = XComponent[class][x] & YComponent[class][y] & ZComponent[class][z]
end while
```

4.2.5 Classification Mode

The nature of our problem suggests that even the smallest lighting changes can significantly affect the performance of color segmentation method. Although, this problem can be overcome in many cases, by exploiting color locality, there are many cases such as less saturated objects or overlaps with another color classes that create noisy shades on objects, affecting valuable information such as object size. To overcome this problem we use binary classification that in contrast with multiclass classification, it is the task of classifying the members of a given set of objects into one of two groups. Thus, we create a binary classifier for each color class checking if a pixel belongs to a color class or not. Hence, if a pixel belongs in more than one classes, it can be ignored until the entire image is processed. Thus, The risk of false positives is reduced by allowing true color to be classed as the ignored pixel will be used for additional size information. Generally, binary classification was applied to separate the red uniform and orange ball, the orange ball and yellow goal, the blue goal from the blue uniform.

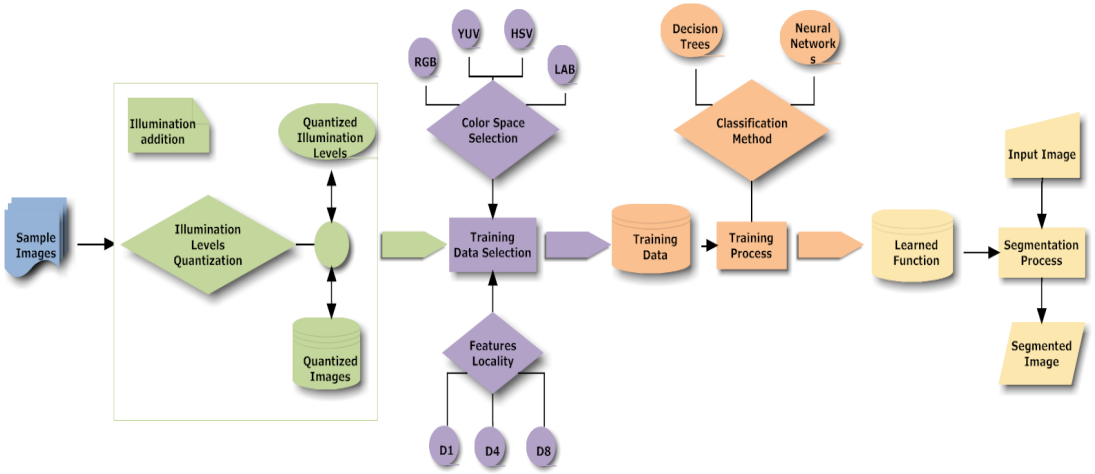


FIGURE 4.8: Illuminant Features

FIGURE 4.9: Original (two illuminations) and segmented images (N5 in two illuminations and N5+3 schemes).

4.3 Coping With Varying Illumination Conditions

We have already mentioned that our method uses pixel values to learn a classification function that will determine the membership of a pixel. However, under varying illumination conditions pixel values can change rapidly directly affecting method's efficiency. Thus, vision systems need to be calibrated many times over the day even on the same field, since the RoboCup 2009 competitions will be held by using only the ceiling lights of the venue. Most of vision algorithm involve solutions that take into consideration the geometric information related to the objects of interest. Nevertheless, since robotic soccer is a real time task, these computationally demanding techniques are not suitable. For this reason, we have created two alternative modifications of our vision system that can overcome these problems.

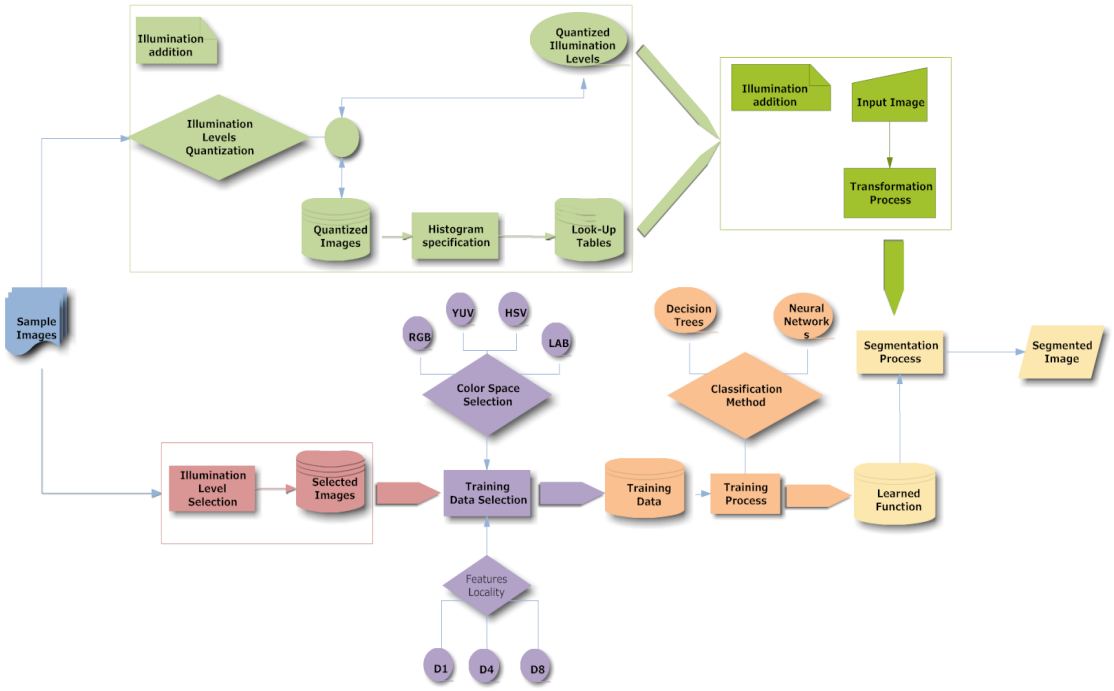


FIGURE 4.10: Histogram Specification Modification

4.3.1 Illuminant Features

This alternative method overcomes the problem of varying illumination, by providing three additional inputs to classifier: the average values of the three color dimensions over the image. These features provide information about the global illumination level. This alternative method has slight differences from our main method, that are mainly located in image capturing-selection and in training process. To begin with, in image capturing-selection, the user needs to determine the illumination levels that vision system will operate. Afterwards, images for each of these stages have to be captured, in order to create training data that contain information of all the illumination levels. In training process stage, there is an option about illuminant features usage that enables to the method to use the additional inputs as classification features. Thus, the learned classification function will be able to understand each color class under the specific illumination level using these additional inputs. Figure 4.9 demonstrates the benefits of the N5+3 scheme, trained with data from both the bright and the dark images over the N5 scheme on the dark raw image, whereas Figure 4.8 illustrates the alternative color recognition procedure.

4.3.2 Histogram Specification

This color recognition alternative method is related to histogram specification technique, that gives us the ability to specify the transformation function that is needed to specify particular histogram shapes. In addition, we have studied the possibility of applying complex transformations over a raw camera image with the goal to create a modified image with desirable characteristics. Thus, capturing images in multiple illumination levels between the lowest and highest illumination over a field, it is possible to create a discrete set of illumination levels over this area. Afterwards, selecting one image taken under one of these illumination levels as reference, histogram specification method can create transformations are needed to be applied to images of the other illumination levels to transit to the reference one. In addition, we can learn classification functions over the reference illumination that will be used to classify the transformed images. Thus, histogram specification technique can be integrated into our basic method in stages of "Images Capturing-Selection"and in "Segmentation Process".

Images Capturing-Selection

Similarly to our standard method, the first stage of this alternative method consists of two parts: image capturing and image selection that can be seen in Figure 4.10 as the blue and pink stages. Firstly, the user needs to determine the different illumination levels in which this method will operate. In particular, we need to capture the same image under selected illumination levels. Afterwards, images that include classes of interest under these illumination levels are captured. In addition, "Image Selection"is a task in which user needs to select which of captured images will be used in training process.

Transformation Functions Creation

This is the most important stage of this alternative method, since it is responsible for the creation of a look-up table which includes the transformations for all the discrete illumination levels . At first, we need to calculate the histogram of each component of the selected image that lies in reference illumination. Similarly, the component histograms of the selected image, captured in each of the discrete illumination levels, must be calculated. The next step of the procedure is to apply the histogram specification method to each of the calculated component histograms. This will result a set of transformations, that applied in the image taken under the different illumination levels resulting the image of reference illumination.

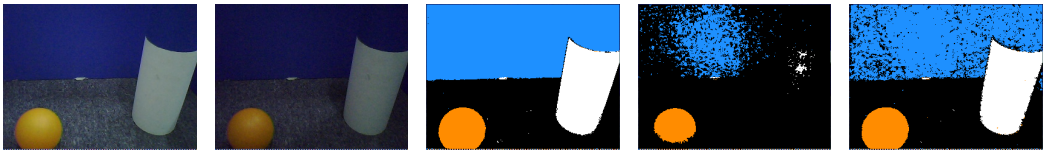


FIGURE 4.11: Original (two illuminations) and segmented images (N5 in two illuminations and HS Modified).

The resulted look-up table makes this method appropriate for a real time color recognition process. Nevertheless, we need to define a metric of illumination that will be used as an indicator to look-up table transformations. Similarly to the previous method we use the average values of the three color dimensions over the image creating a classifier that will determine the illumination level providing these three inputs. Thus, the learned classification function using the average values of the three color dimensions over the image, will determine what is the most similar of these discrete illumination levels, creating an index to transformation table. The procedure of transformation tables creation is described as follows:

Algorithm 2 Transformation Table Creation

Require: ReferenceHistogram[3][256], IlluminationLevel1Histogram[3][256] ... IlluminationLevelNHistogram[3][256]
Ensure: TransformationTable[n][3][256]

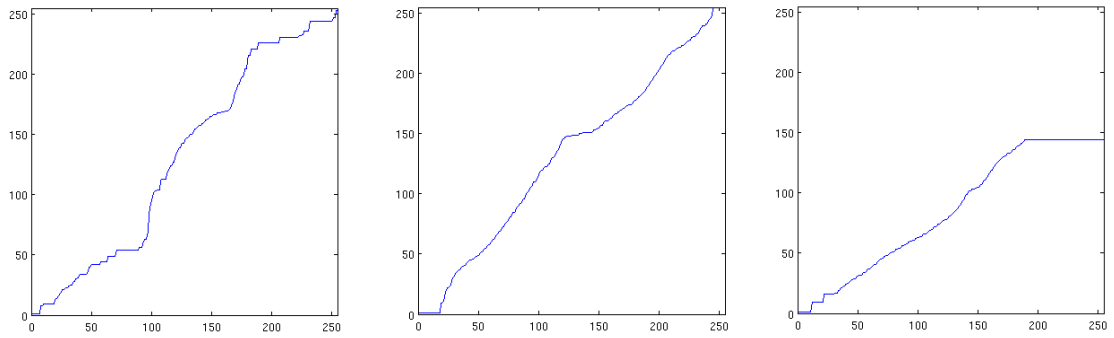
for j = 1...N **do**
  for i = 1...3 **do**

$p_x = ReferenceHistogram[i][256]$
$p_z = IlluminationLeveljHistogram[i][256]$

$$F_x(l) = \sum_{l=0}^{l} p_x$$
$$F_z(j) = \sum_{l=0}^{j} p_z$$
$$|F_x(l) - F_z(j)| = argmin(|F_x(l) - F_z(j))$$

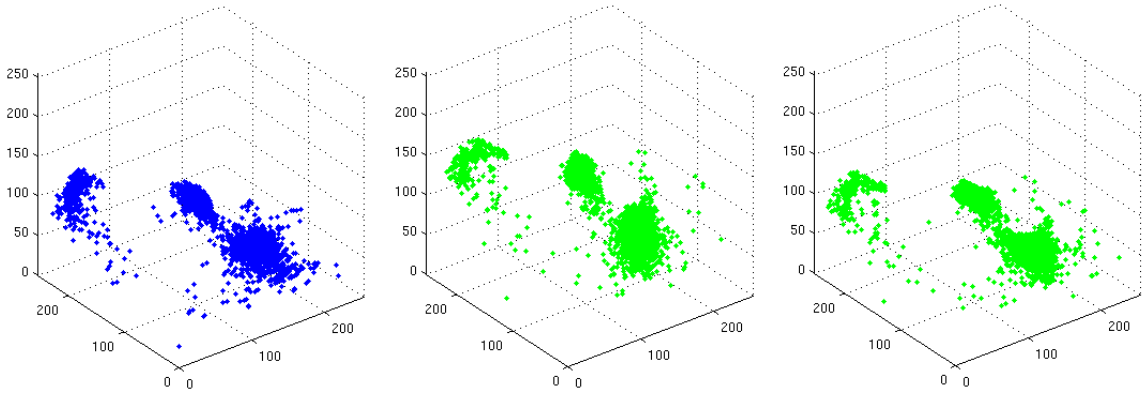
$$TransformationTable[j][i][256] = F_z(j)^{-1}$$

  end for
end for



(A) Hue (B) Saturation (C) Intensity

FIGURE 4.12: Calculated Transformations



(A) level1 (B) level2 (C) modified level2

FIGURE 4.13: Image locus representation of original image

Segmentation Process

Segmentation process is the stage in which learned classification function is applied over an image. Nevertheless, we need to apply the appropriate transformation due to illumination metric, before the membership evaluation of this image. In Figure 4.12 we can see the transformations in HSV color space that have been created by applying the method described above in images of Figure 4.11. Moreover, in Figure 4.11 we can also see the classification results before and after the method's application. In Figure 4.13 we have the locus of images in HSV seen in Figure 4.11 in two illumination levels. The level is selected as reference illumination whereas the second is modified applying the method. After transformations application on the image of second level we acquire a locus that is similar to the first one.

CHAPTER 5

Implementation Overview

5.1 Color Recognition Method Overview

Our Classification method can be dived into two major stage. The first stage consists of an offline procedure that produces an classifying function that will be responsible for image pixes categorisation. The second stage comprise the online procedure of color recognition. In Figure 5.1 we can see an overview of color recorg intion method along with the stage of two procedures. The offline procedure consists of four major stages:Images Capturing-Selection, Training Data Selection, Training Process, Segmentation whereas the online procedure consists of Image Restoration and Color Recognition.

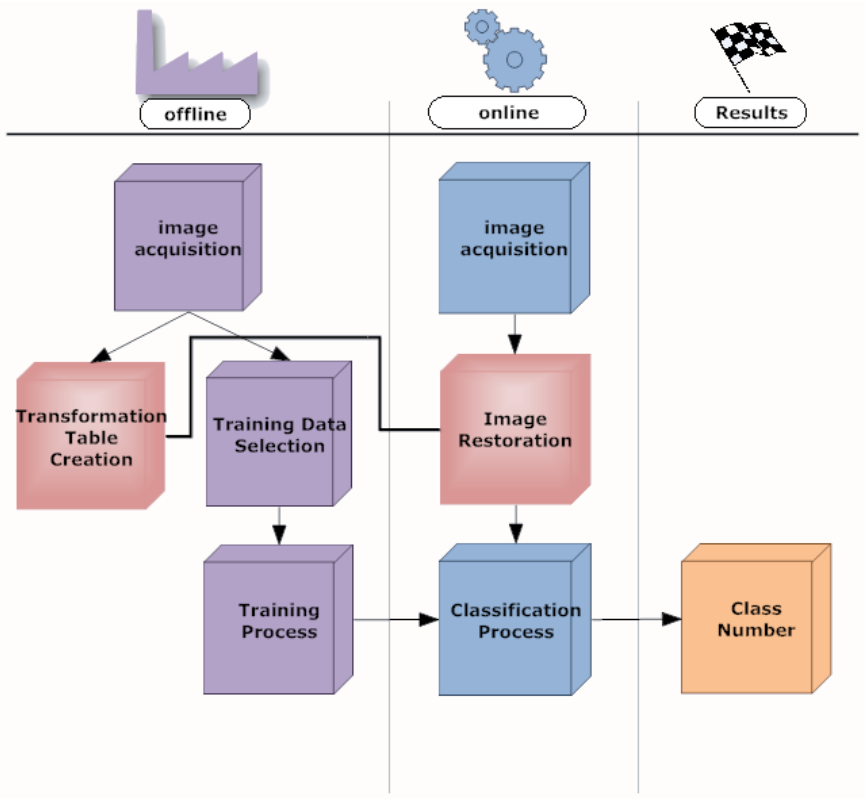


FIGURE 5.1: Color Recognition Method Overview.

5.2 The Kouretes Color Classifier

In this chapter we introduce our software package system Kouretes Color Classifier (KC^2) which is designed and implemented in Matlab to be used as a toolbox for the solution of recognition. Major objectives is the creation of a Graphical User Interface, that should be fast, light, simple, and intuitive being at the same time easily maintained, by others.The correct color segmentation of an image frame is a fairly complex process. Not only this is a fairly subjective procedure, but it also requires a new classifier for any location and lighting scheme that take place. Due to the different angles, an object can be viewed from and due to different lighting conditions even for the same object, a large number of images must be taken and labeled, which often requires up to one hour of manual work. The primary goal of this team effort was to make the procedure more precise and efficient, minimising the human effort and involvement. The classifier needs to be trained using manually labeled data, which are taken from the camera images. After training, it is fairly easy to classify each pixel of the image using the learned classifier. For training purposes, we need to take samples for each color class from several camera images and manually label them. This is a feature of the Nao Color Classifier tool, shown in Figure 5.2. Large areas of the raw image taken by the Nao can be selected using a graphical interface and all pixels within each area are associated with the desired color class label. The YUV locus (projected in each of the three dimensions) is displayed

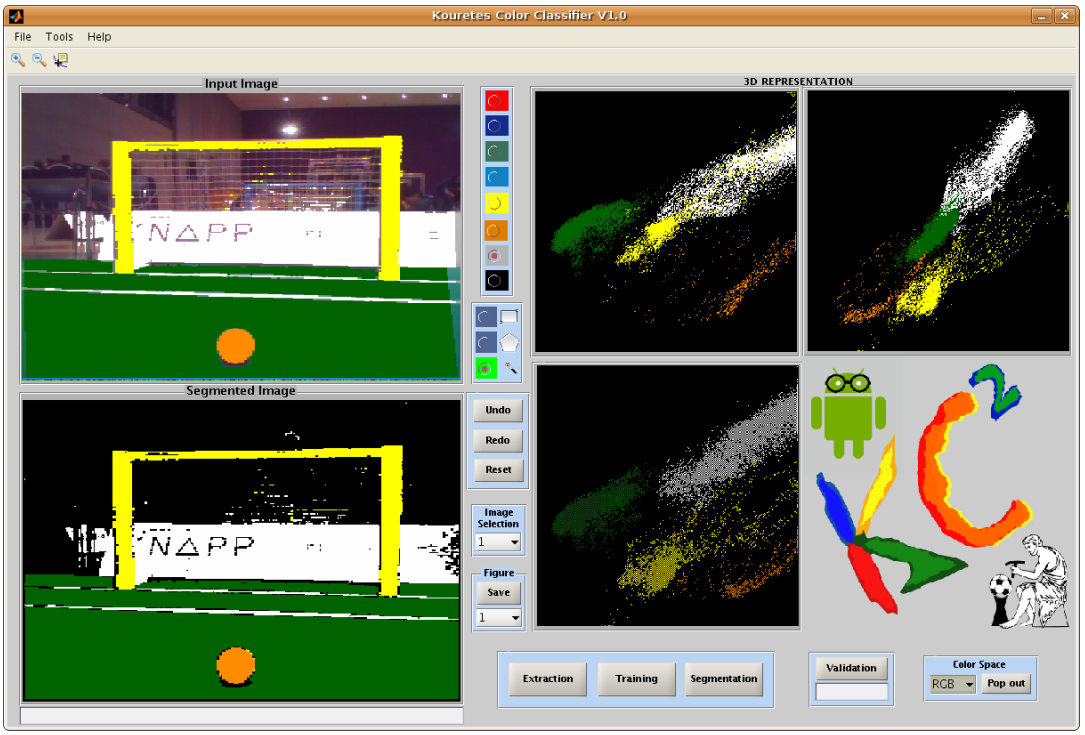


FIGURE 5.2: Kouretes Color Classifier (KC^2)

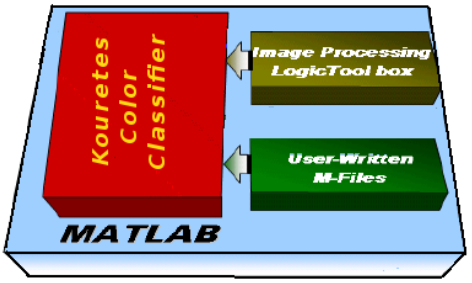


FIGURE 5.3: Inter-relations between KC^2 and MATLAB

during the process. Once a sufficient number of data has been collected from several images, we can train the classifier to learn a good set of parameters. Kouretes Color Classifier was implemented in Matlab using Image processing toolbox (see Figure 5.3) and the implemented in such way that will be easily maintained in future.

5.2.1 Software Architecture

Although KC^2 is implemented in Matlab environment we have the ability to create a stand-alone C or C++ graphics application, use the MATLAB Compiler (mcc). The MATLAB Compiler translates the specified M-file into a C or C++ source code module generating additional C or C++ source code modules, called wrapper files, required by stand-alone applications. Finally, invokes your C or C++ compiler and linker to create

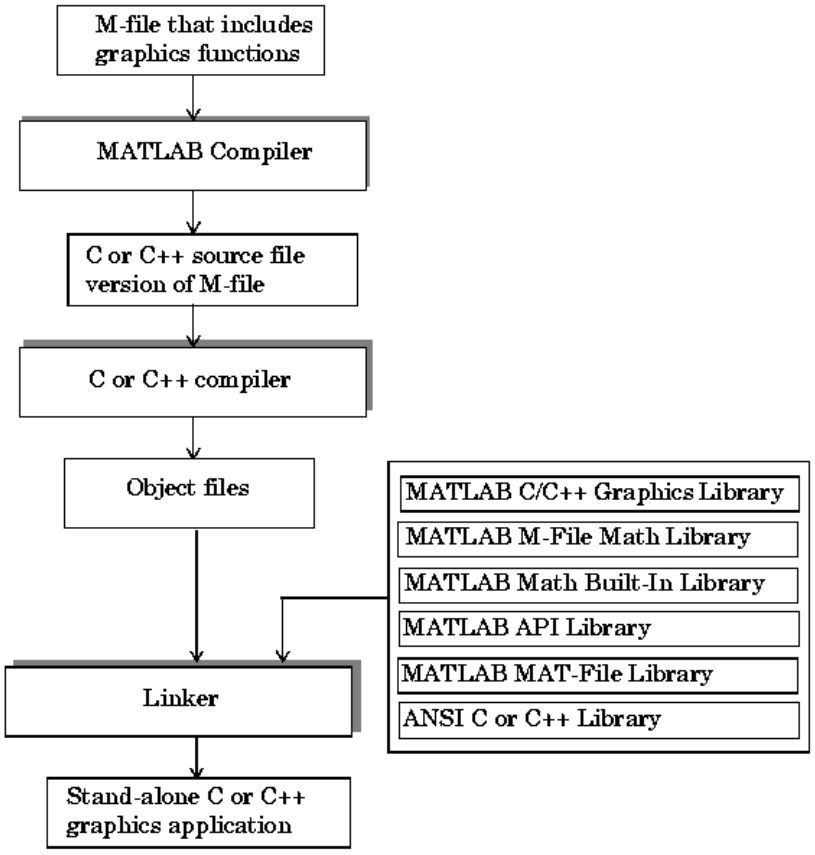


FIGURE 5.4: Creating of a Stand-Alone C or C++ Graphics Applications.

the stand-alone application executable Figure 5.4 shows how the Compiler works in conjunction with other tools and libraries to create a stand-alone graphics application.

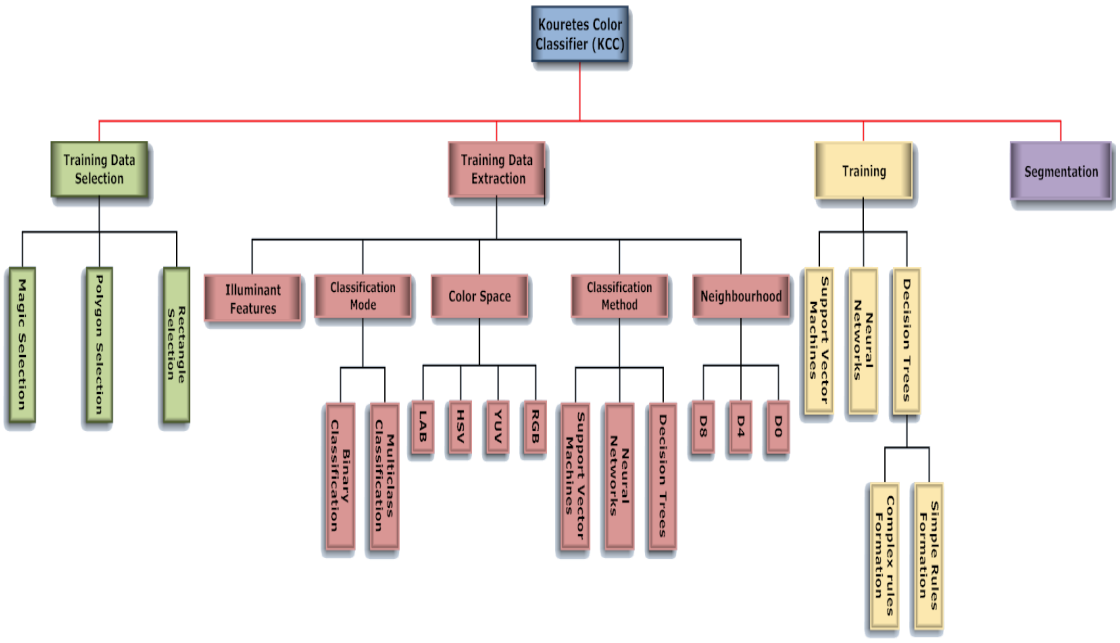


FIGURE 5.5: KC^2 functionality.

5.2.2 The Graphical User Interface

The KME graphical user interface (GUI) provides the means for the creation, capture, management, storage, reproduction, and export of any sequence of robot poses. The entire GUI consists of three components as shown in Figure 5.3. The CCT graphic user interface embraces a large number of topics, including creation, management and storage of classifiers. The entire interface consists of three major components. The first component is responsible for the selection of the pixels that consist the vital data of the essential training process. The second component is responsible for the training process and last, but not least, is the component that is responsible for the classifier validation.

5.2.3 Process Description

The classifier needs to be trained using manually labeled data taken from the camera images. After training, it is fairly easy to classify each pixel of the image using the learned classifier. For training purposes, we need to take samples for each color class from several camera images and manually label them. Large areas of the raw image taken by the Nao can be selected using a graphical interface and all pixels within each area are associated with the desired color class label. Once a sufficient number of data has been collected from several images, we can train the classifier to learn a good set of parameters.

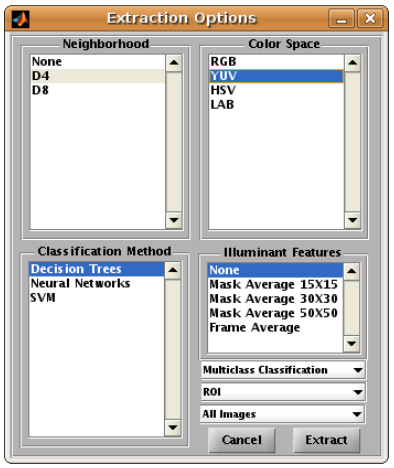


FIGURE 5.6: $K_C{}^2$ extraction configurations.

5.2.4 Data Selection Process

Recognition is a major goal in machine vision. In this field, we can define an object with textural property. Textural property is related to this fact because, the image of real objects often does not exhibit regions of uniform intensities and textural image, defined as a function of the spatial variation in pixel intensities. In addition, in real time object recognition a method has been established that consists of two levels. The first level includes color segmentation and the second level commits the object recognition using the resulted color blobs of the first level. Thus, the simplest way to classify an image is by using the color attributes in a classification algorithm. Take into consideration, that some rules about feature selection are desirable in the training process. Using a smaller feature set may improve classification accuracy by eliminating noise inducing features. Small feature sets should be more generalisable to unseen data. If training data is in short supply, the use of a small number of features may reduce the risk of over-fitting the parameters of a classifier to the training data. The use of a small feature set raises the credibility of the estimated performance of the classifier. Our data selection is done independently of the learning algorithm. In addition, in training process most of classification algorithms follow a wrapper approach and some others the filter approach. The filter approach is computationally more efficient but its major drawback is that an optimal selection of features may not be independent of the inductive and representational biases of the learning algorithm that is used to build the classifier. On the other hand, the wrapper approach involves the computational overhead of evaluating a candidate feature subset by executing a selected learning algorithm on the database using each feature subset under consideration. Data Selection Process provides the selection methods that minimise human effort. The first method involves rectangle selection, the second involves polygon selection and th last involves clustering algorithms (see Figire 5.7).

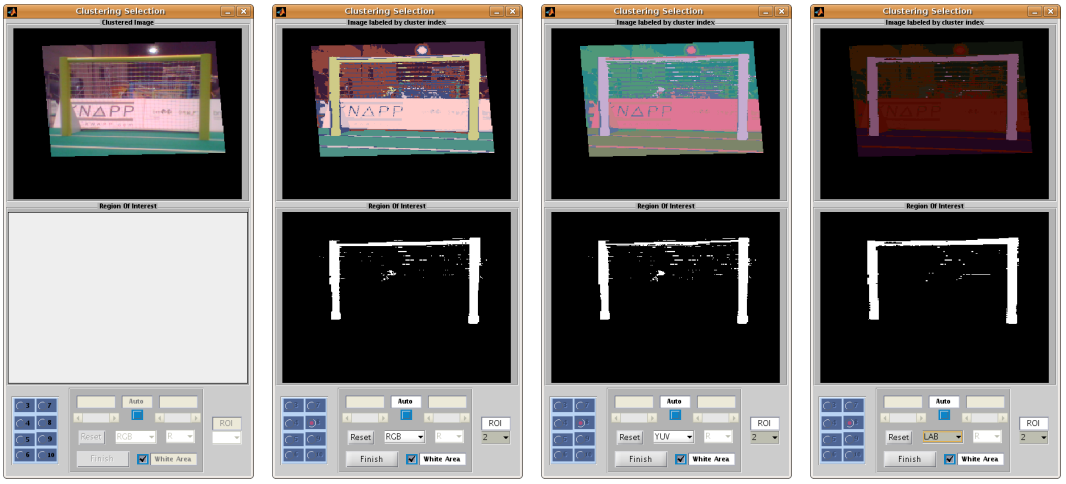


FIGURE 5.7: Magic Selection

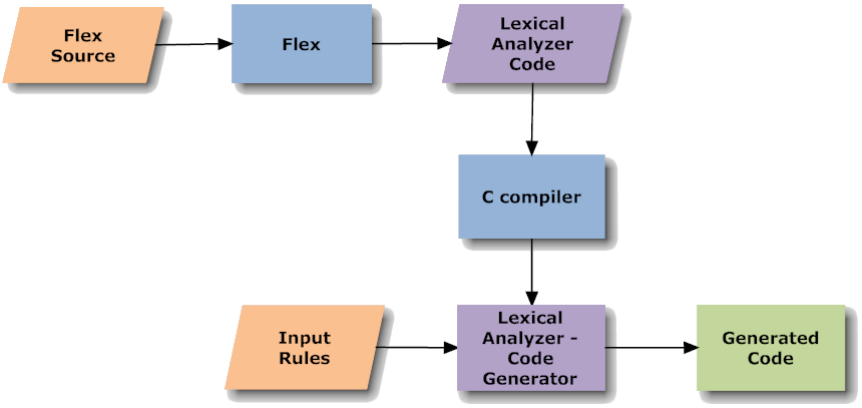


FIGURE 5.8: Lexer Analyser.

5.2.5 Extraction and Training Processes

Once a sufficient number of data has been collected from several images, we need to extract the data into files. Afterwards we can train the classifier to learn a good set of parameters. In addition, before data extracting user must select among configurations that referred in number of classification features, in data nature and in classification method. Extraction involves classification method configuration options (Figure 5.6) that increase classification efficiency (see chapter 4). The training process involves the modification stage that decision trees need for code generation that will be integrated on robot. In Figure 5.8 the procedure that must be follow to create the complex rules format.

Features Locality

Under difficult lighting situations, we can exploit color locality by using as input attributes not only the values of the current pixel, but also the values of pixels in its immediate neighbourhood. These can be the 4 orthonormal or the 8 orthonormal neighboring pixels. Through extensive experimentation we found that the neighbourhood of the 4 diagonal neighboring pixels works best for our purposes. Figure 8 demonstrates our color segmentation method. A decision-tree classifier has been trained under the N1 scheme using over 50,000 training examples by labelling various regions in images taken by the robot camera. This moderate number of training data works well in avoiding under- and over-fitting. Similarly, another decision-tree classifier has been training under the N5 scheme using over 100,000 training examples; the increase in the number of training data is necessary given the increase in the number of input attributes. The segmented image for each of these two classifiers is shown in Figure 8. In either case, the colors of interest have correctly been identified, however the N5 scheme seems to yield somewhat better results than the N1 scheme.

Classification Methods

Statistical classification is a procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items and based on a training set of previously labeled items.Formally, the problem can be stated as follows: given training data

$\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$ produce a classifier

h: $\mathcal{X} \rightarrow \mathcal{Y}$

which maps an object $\mathbf{x} \in \mathcal{X}$ to its classification label y $\in \mathcal{Y}$.In our tool we use the below statistical classification algorithms :

⋆ Decision Tree

⋆ Neural Network

⋆ Support Vector Machines

Classification Mode

Under difficult illumination conditions we can create some more color classes in order to achieve a better estimation of pixel classes elimination the classification error. Thus, it is vital to create a method of soft color classes estimation. The term soft color class is referred in a class that includes a pixel that belongs to the boundary of two or more color classes. Thus, the classification of these pixels is very difficult and the classification error too high. Hence, binary classification can improve the classification efficiency, creating n binary classifiers for the n color classes instead of creating one multiclass classifier for these n classes. This is an easy way to detect the pixels that belong in these soft color classes simply by searching if a pixel is classified in more than one binary classifiers. As a consequence user can classify these pixels in the second level of Vision using temporal information.

5.2.6 Validation Process

The final step is to validate the resulting classifier estimating the efficiency. Thus, we can use two ways of validation. The first way is the K-fold cross validation and the second is visualising the classification results.

5.2.6.1 K-Fold Cross Validation

Randomly break the dataset into k partitions. For each of k partitions, use k-1 folds for training and the remaining one for testing. The advantage of K-Fold Cross validation is that all the examples in the dataset are eventually used for both training and testing. Finally, the true error is estimated as the average error rate.

5.2.6.2 Visualised Validation

After training, it is fairly easy to classify each pixel of the image using the learned classifier. Moreover, after each training the resulted classifier is saved in a proper form and can be used independently to classify the pixels of an image offering a visual validation of method efficiency.

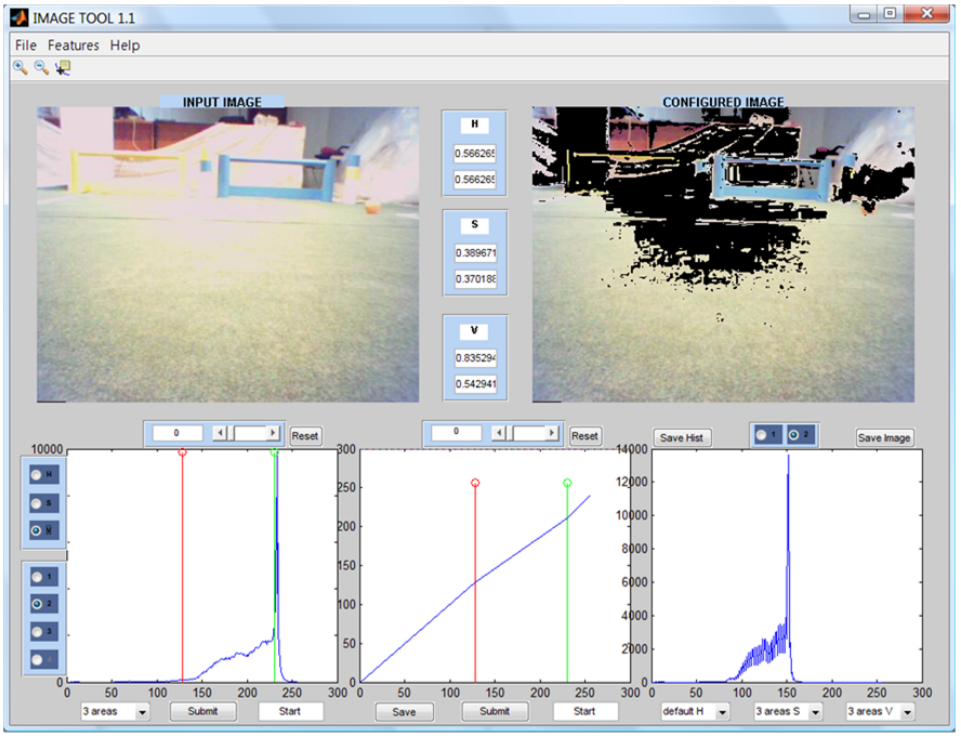


FIGURE 5.9: The graphical tool for studying transformations using piecewise linear lters.

5.2.7 Additional Tools

Past experience has shown that a failure in the color segmentation procedure can be catastrophic for the robot team. Therefore, we have studied the possibility of applying complex transformations on the raw camera image with the goal of achieving best color separation which in turn will allow for robust color segmentation. To this end, we have developed a couple of tools for studying images in RGB, YUV, and HSI formats. In particular, the rst tool analyses the image and displays histograms for each color component (R/G/B, Y/U/V, or H/S/I). Furthermore, it allows the direct additive or multiplicative modication of all R/G/B, Y/U/V, or H/S/I values and the results are displayed on screen in real time. The other tool enables the study of the effect of various transformations on the image when applying piece-wise linear lters to each of the three color dimensions (Figure 9). The range of each dimension can be split into any number of intervals and an arbitrary linear transformation can be applied to each one of them. The resulting non-linear transformation of the image is displayed in real time on screen. The nal version of these tools will be incorporated into the Nao Color Classier tool to allow the user to use the transformed images as input to the color segmentation process and observe the combined result. In Figure 5.9 we can see the graphical tool for studying transformations using piecewise linear lters whereas in Figure 2.12 we can see the graphical tool for studying image histograms.l

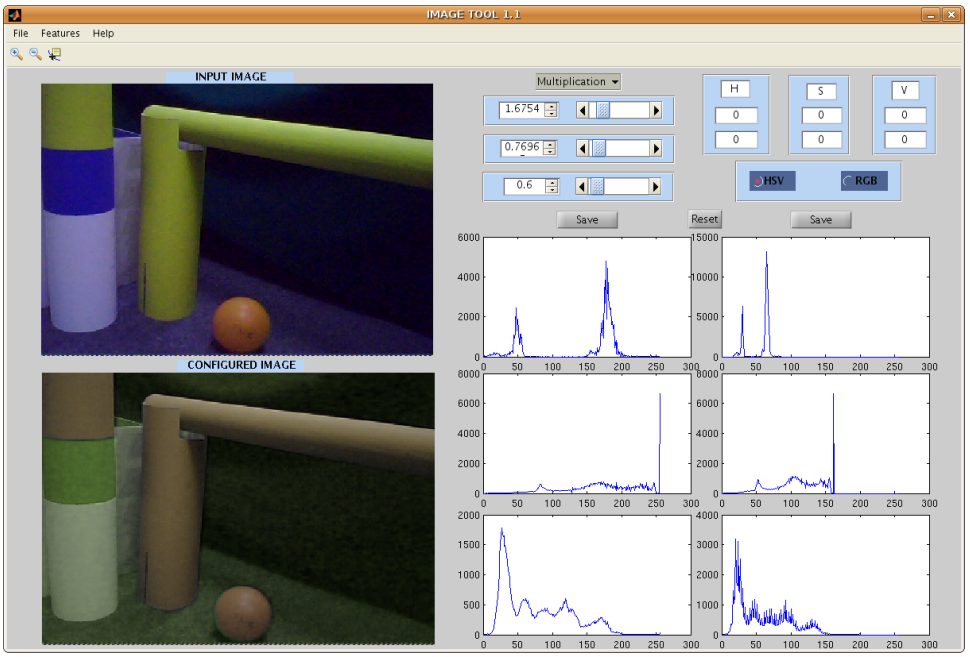


FIGURE 5.10: The graphical tool for studying image histograms.

5.3 System Integration on Robot

In Figure 5.11 we can see the class that used in Color Recognition Module along with their major methods. The class KCam is responsible from frame capturing. This class provide the next class of Segmentation with the frame pixel needed for membership checking. This class consists of four function in simple rules format and five in complex.

getColor: This function is teh top level part of color recognition. Takes as input pixels values along with the adjacent ones returning the segmented image.

classifier: This function that commits color recognition. Takes as input pixels values along with the adjacent ones returning the pixel's class num.

findindex: This function takes as input a pointer to the current camera frame and pixe's position returninf pixel values.

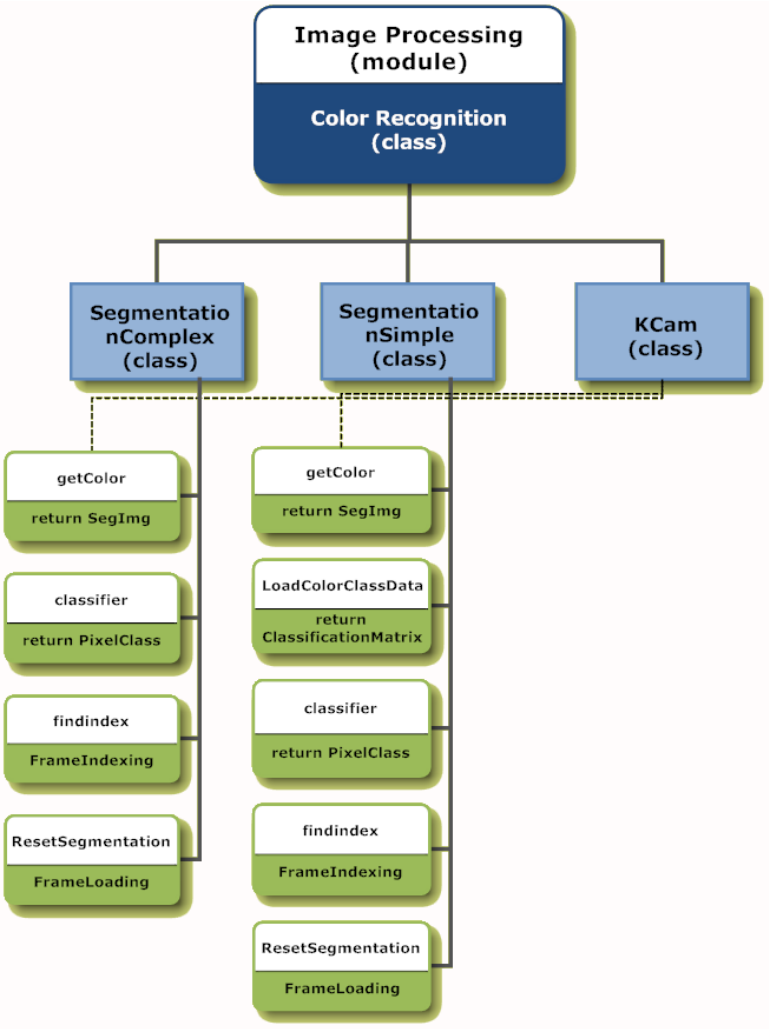


FIGURE 5.11: Color Classification Classes Block Diagram.

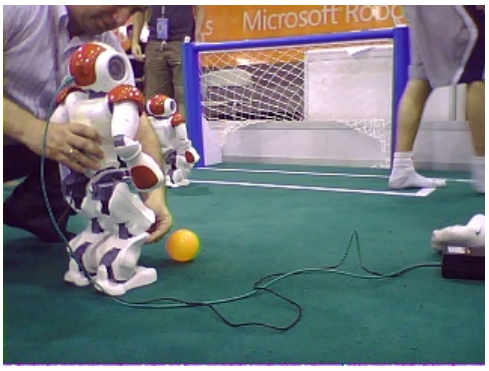
CHAPTER 6

Empirical Evaluation

RoboCups nature is so different than anything else scientists may ever attend. RoboCup is a very competitive environment, where most researchers benchmark their work in real-world problems. Consequently, all our proposed solutions were to be tested and evaluated during this competition. Thelastchapterofthepresentthesisdealswiththeevaluationoftheproposedapproach.

6.1 ColorSpace

We need to check the classification options efficient trying to take into consideration the alternative configuration options. Figure 6.1 demonstrates the classification results of four differect classification function that have been raised for the same training data labelling using the same classification method but different color space. In this way we tried to investigate if the choice any color space affect the classification results. Thus, in Figure 6.1 we can see that classification procedure is independent from color space.



(A) original image

(B) RGB

(C) YUV

(D) HSV

(E) LAB

FIGURE 6.1: Color Space Comparison.

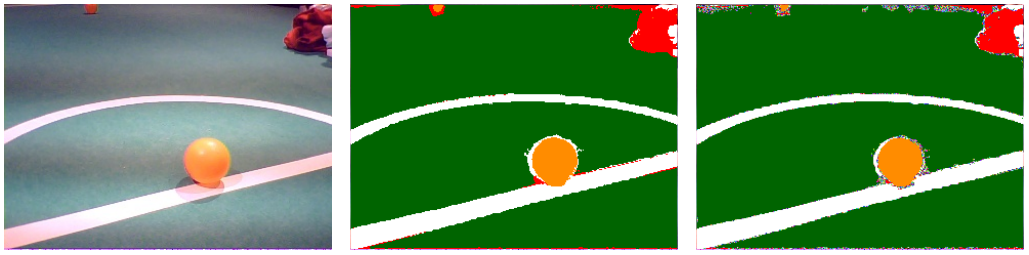


FIGURE 6.2: Binary Classification Method (Original Image, Multiclass Classification, Binary classification).

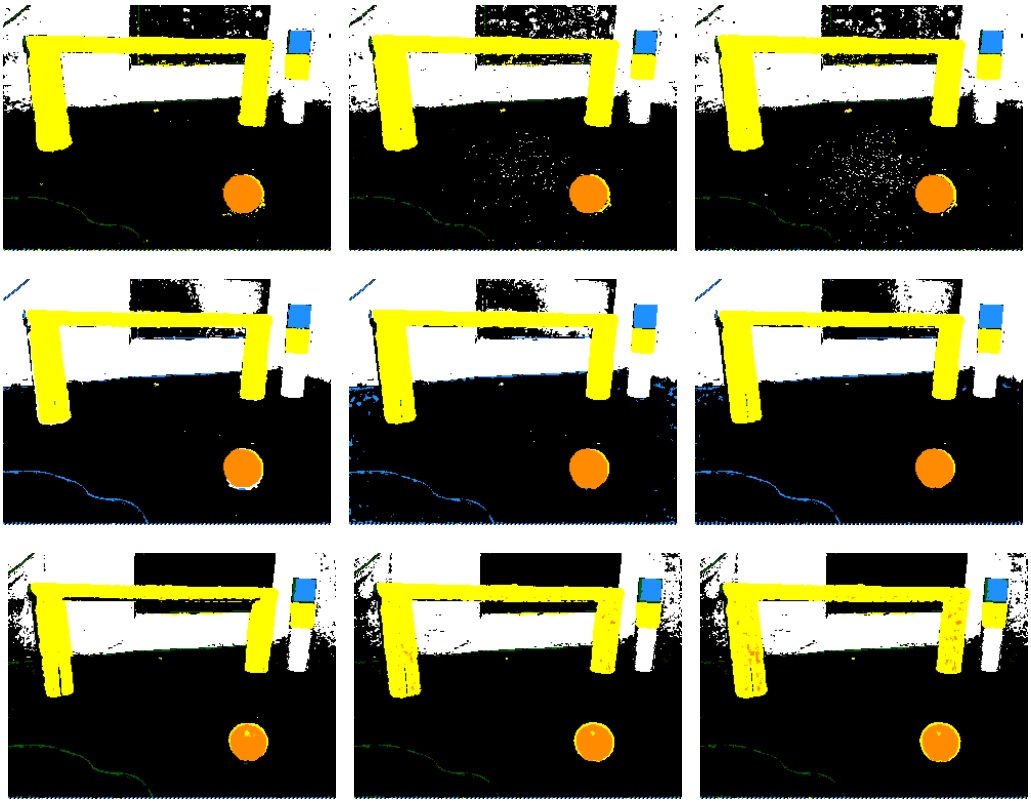
6.2 ClassificationMode

In Figure 6.2 we can see how binary classification can detect the noisy shades (figured in random color) on objects reducing the false positives.

6.3 Classification Method

Training procedure was committed under the all possible scheme using the same labelling various regions in images taken by the robot camera using all the possible classification methods. In this way we try to investigate algorithms efficiency and robustness. As we can see in figure 6.3 we can see that all classification methods works efficiently. Nevertheless, in second case (Figure 6.4) SVM and NN seems to have some problems due to these classification parameters tuning.

(A) original image

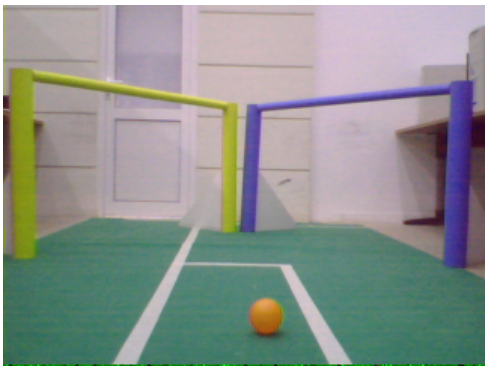


(H) N1 scheme (I) N5 scheme (J) N9 scheme

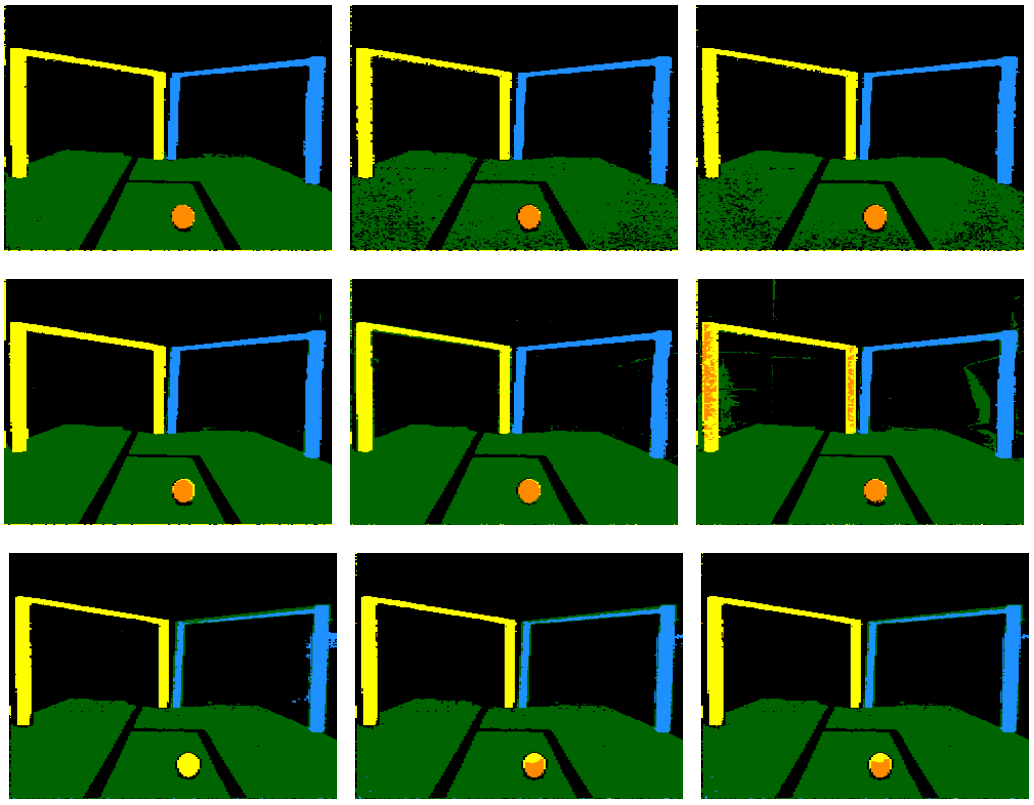
FIGURE 6.3: DT,NN, SVM classification methos comparison (First Case).

6.4 Neighbourhood

Training procedure was committed under the all possible scheme using the same labelling various regions in images taken by the robot camera. This moderate number of training data works well in avoiding under- and over-fitting. The segmented image for each of these four classifiers is shown in Figure 6.5 In either case, the colors of interest have correctly been identified. In first image the Neighbourhood extensions is not necessary whereas in Figure 6.6 the N5 scheme seems to yield somewhat better results than the N1 scheme and the N9 scheme seems to best one..Thus, we see in Figure 6.10 that this



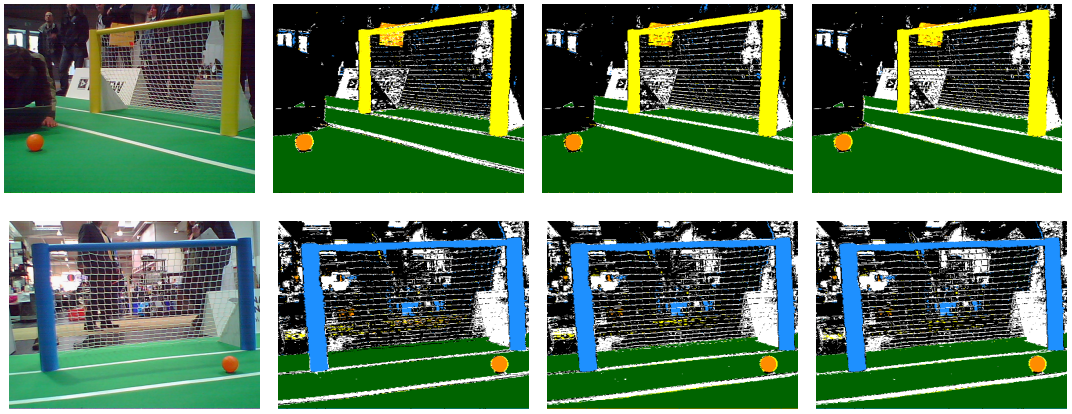
(A) original image



(H) N1 scheme (I) N5 scheme (J) N9 scheme

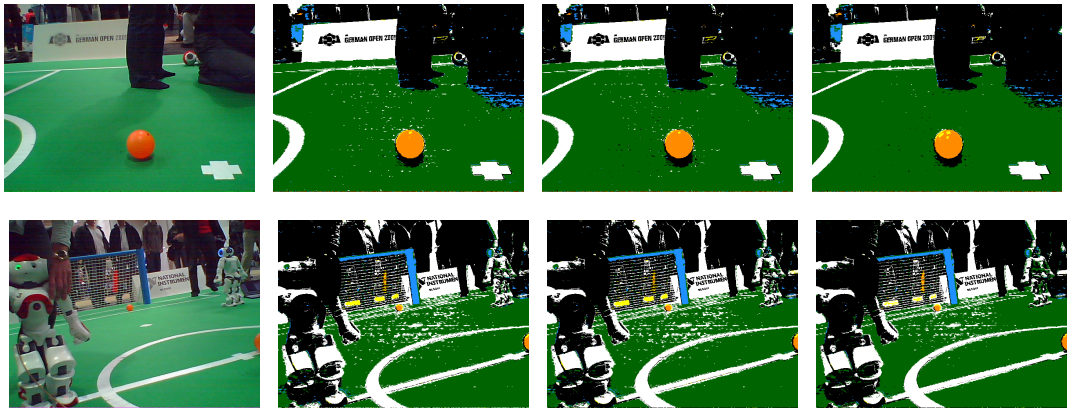
FIGURE 6.4: DT,NN, SVM classification methos comparison (Second Case).

level where classification error is minimum. This happens, because this reference level than the minimum distance from both levels.



(E) Original Images (F) N1 scheme (G) N5 scheme (H) N9 scheme

FIGURE 6.5: DT classification in which neighbourhood extension was not needed.



(E) Original Images (F) N1 scheme (G) N5 scheme (H) N9 scheme

FIGURE 6.6: SVM classification in which neighbourhood extension was needed.

6.5 Varying Illumination

6.5.1 Illuminant Features

6.5.2 Reference level selection.

In this experiment we are trying to investigate illumination effects on color recognition method. Thus, we used the same labelling for all the three illumination levels. The resulted data provided in DT an SVM in all possible neighbourhood schemes.

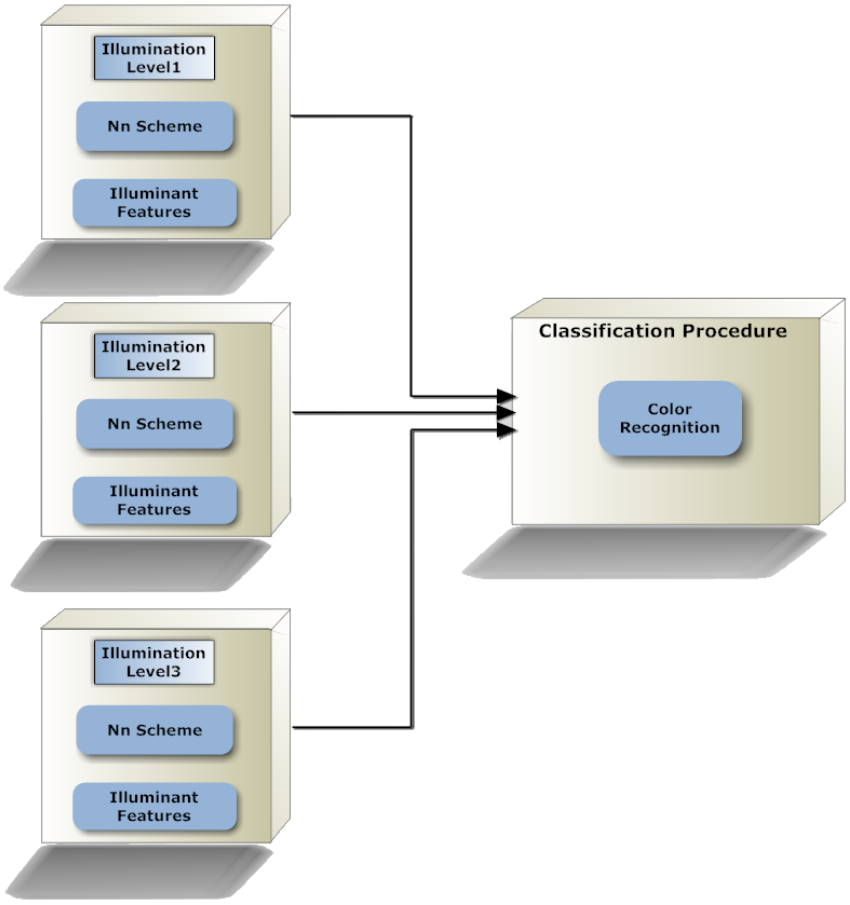
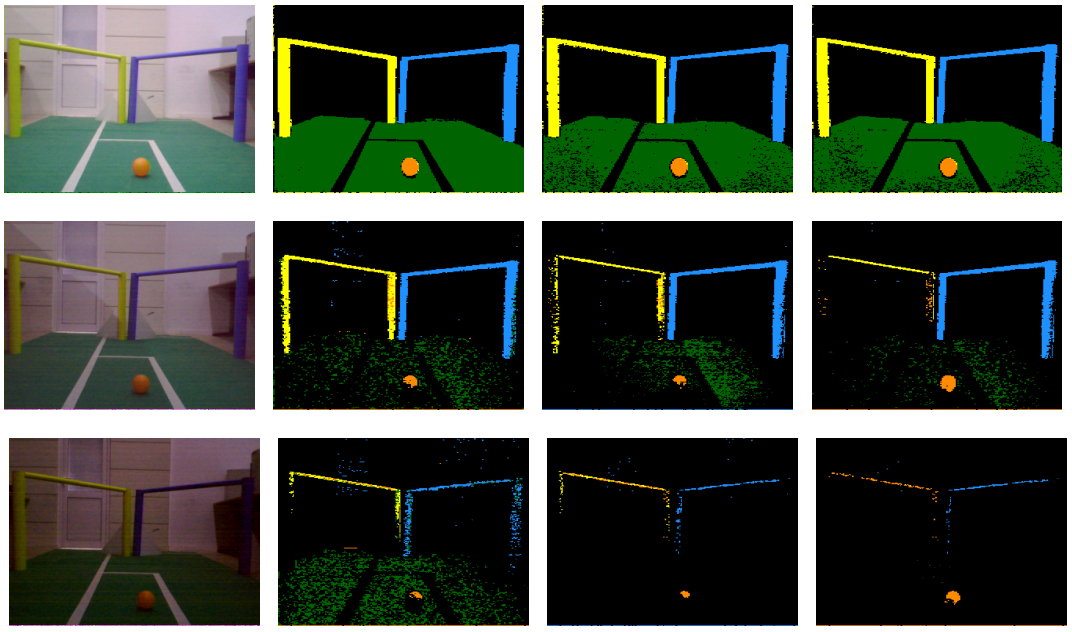


FIGURE 6.7: ILL exp 1.

In addition, we provide classification methods with data taken from oneillumination level each time trying to investigate ho these results affected due to illumination reference. In figure 6.8 we can see that larger illuminant variation increase classification error. In addition, the neighbourhood extension also increase classification error due to that in this cases the change ratio per training sample. In addition, we can observe that SVM classification is not applicable due to training parameters tuning. This procedure is described in Figure 6.7.

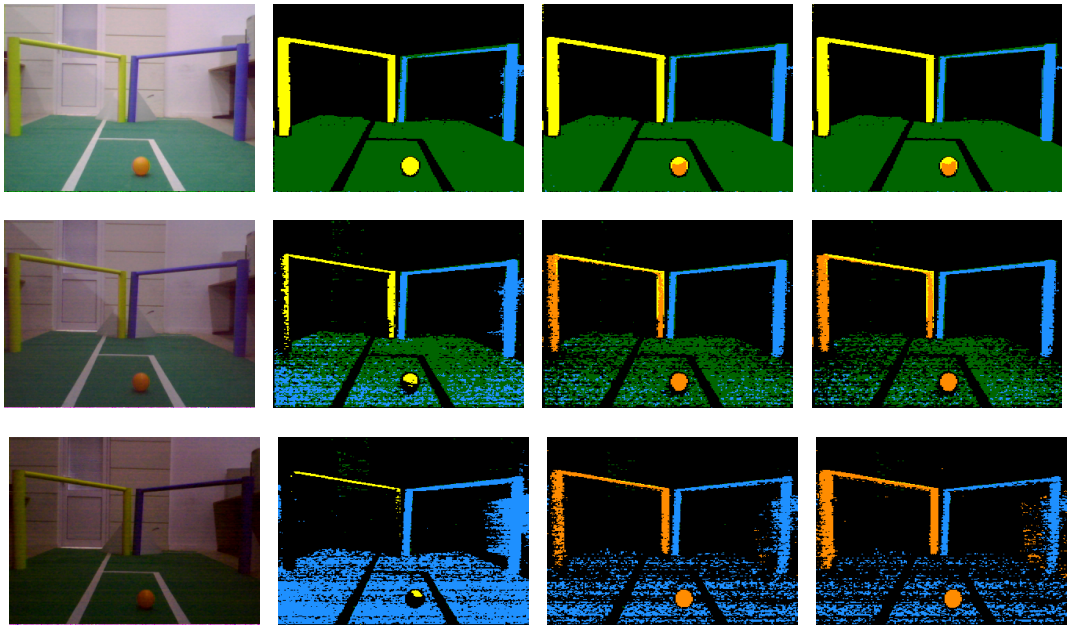
6.5.3 Illuminant Features Efficiency

In this experiment we are trying to find out the illuminant features efficiency. From Figures 6.12, 6.13, 6.14, 6.15 we can see that under varying illumination condition illuminant features incease the method efficiency (see Figures 6.13, 6.14) unless the image environment is not complex (see Figure 6.12). in addition SVM seems to work efficiently with the addition of illuminant features.



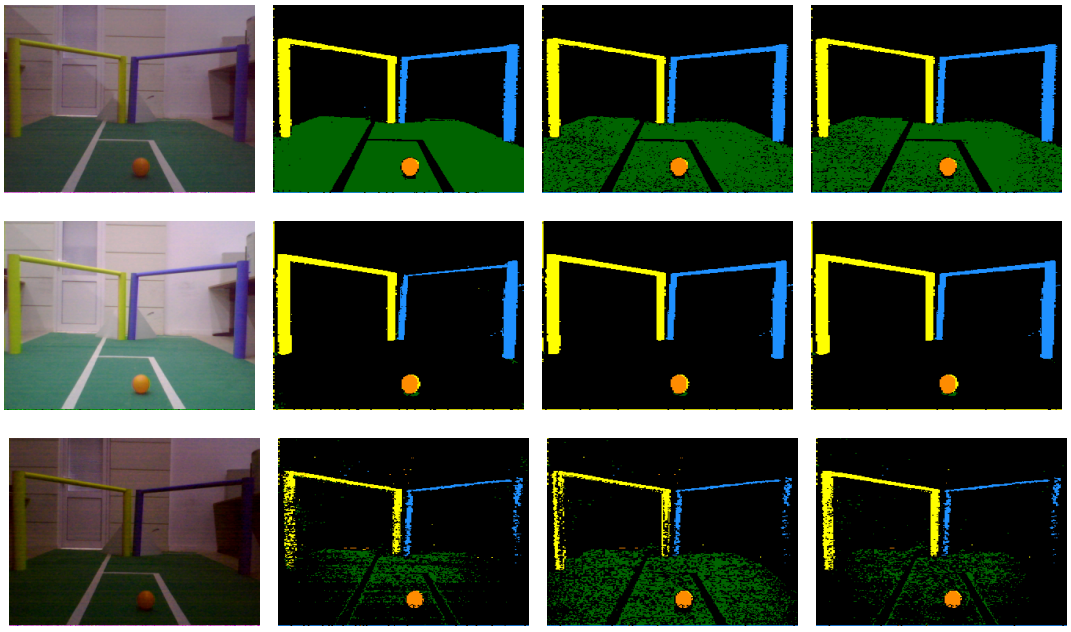
(I) Original Images (J) N1 scheme (K) N5 scheme (L) N9 scheme

FIGURE 6.8: Illumination effects using DT classification on reference level1 due to neighbourhood scheme.



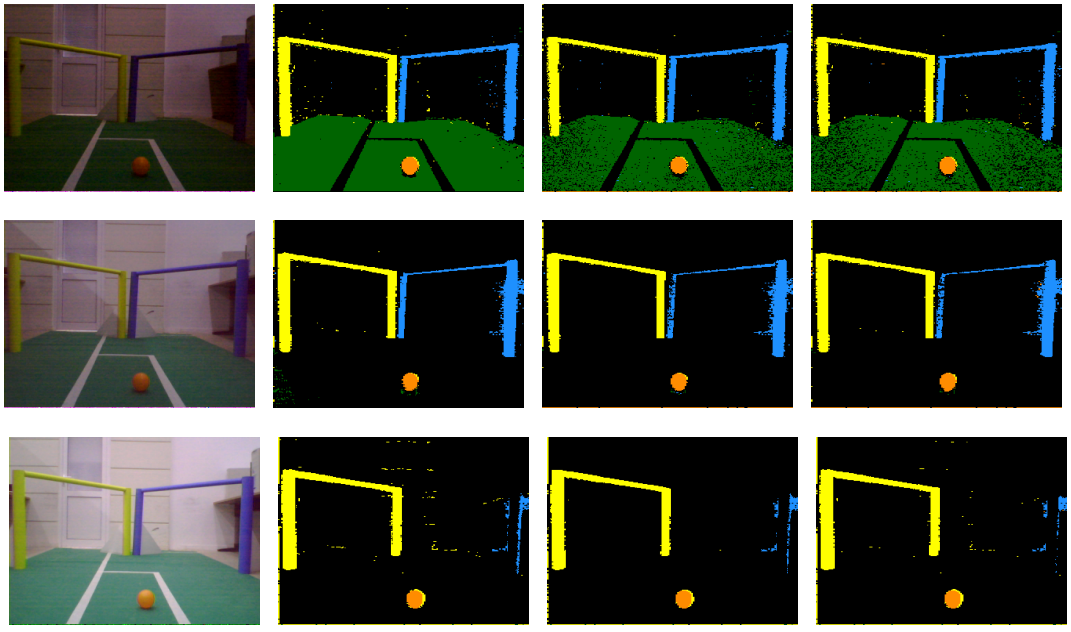
(I) Original Images (J) N1 scheme (K) N5 scheme (L) N9 scheme

FIGURE 6.9: Illumination effects using SVM classification on reference level1 due to neighbourhood scheme.



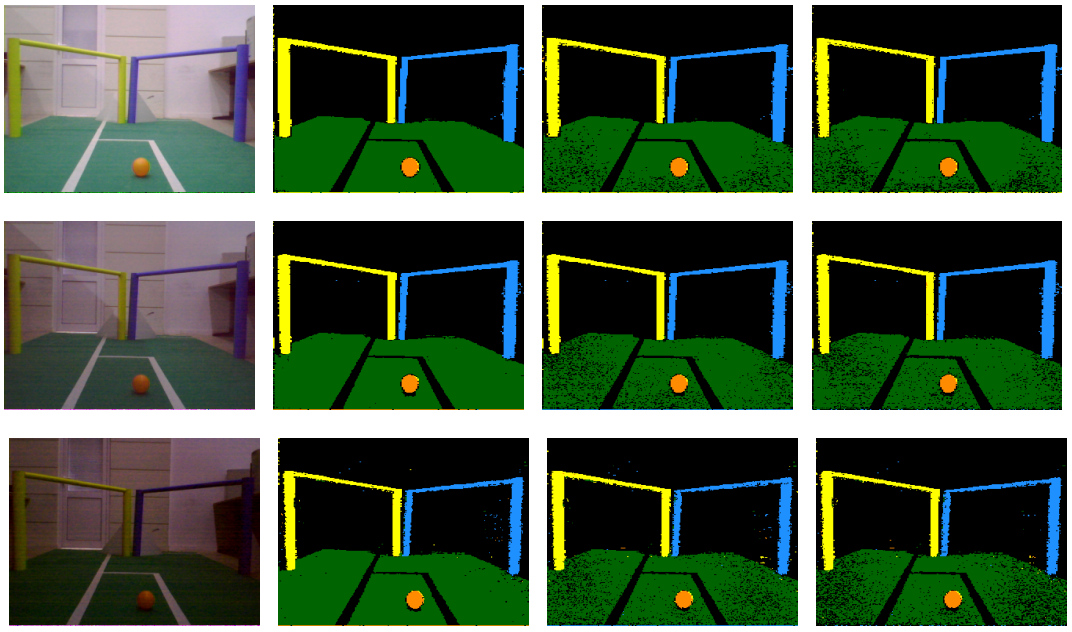
(I) Original Images (J) N1 scheme (K) N5 scheme (L) N9 scheme

FIGURE 6.10: Illumination effects using DT classification on reference level2 due to Neighbourhood scheme.



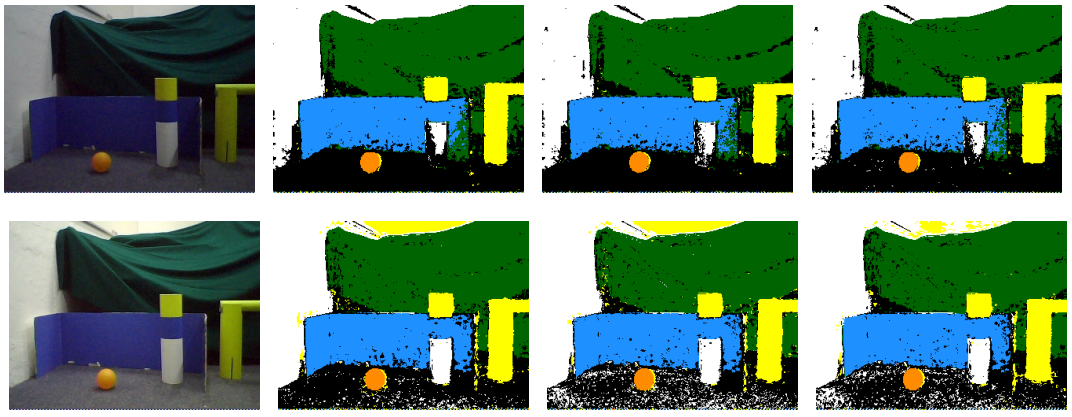
(I) Original Images (J) N1 scheme (K) N5 scheme (L) N9 scheme

FIGURE 6.11: Illumination effects using DT classification on reference level3 due to Neighbourhood scheme.



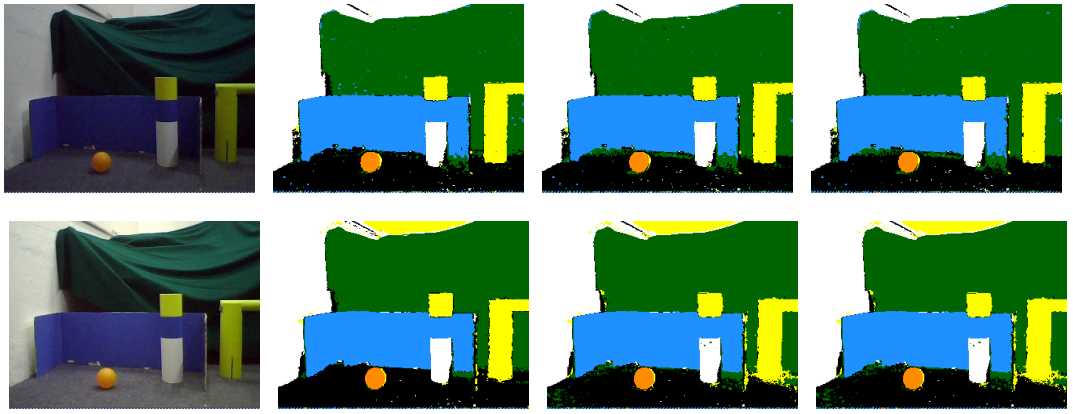
(I) Original Images (J) N1 scheme (K) N5 scheme (L) N9 scheme

FIGURE 6.12: Illumination effects using DT classification on all tree levels.



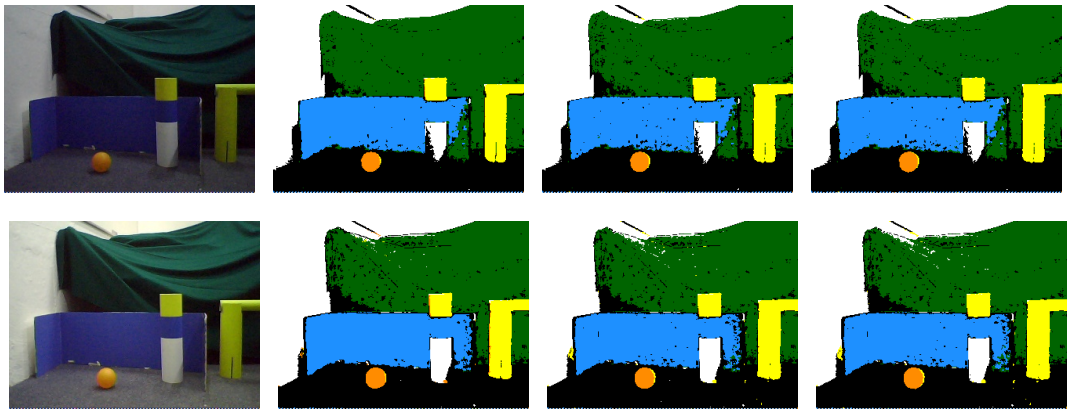
(E) Original Images (F) N1 scheme (G) N5 scheme (H) N9 scheme

FIGURE 6.13: Illumination effects using DT classification on all tree levels.



(E) Original Images (F) N1 scheme (G) N5 scheme (H) N9 scheme

FIGURE 6.14: Illumination effects using DT classification on all tree levels with illuminant features.



(E) Original Images (F) N1 scheme (G) N5 scheme (H) N9 scheme

FIGURE 6.15: Illumination effects using SVM classification on all tree levels with illuminant features.

6.5.4 HistogramSpecification

6.5.5 Multiple Image Histograms

In this experiment we are trying to test the transformation created by Histogram specification method providing it multiple image histograms from 2 levels. In figure 6.17 we can see the efficiency of method and in Figure 6.18 we can see the resulted transformations. This merhod described in Figure 6.16.

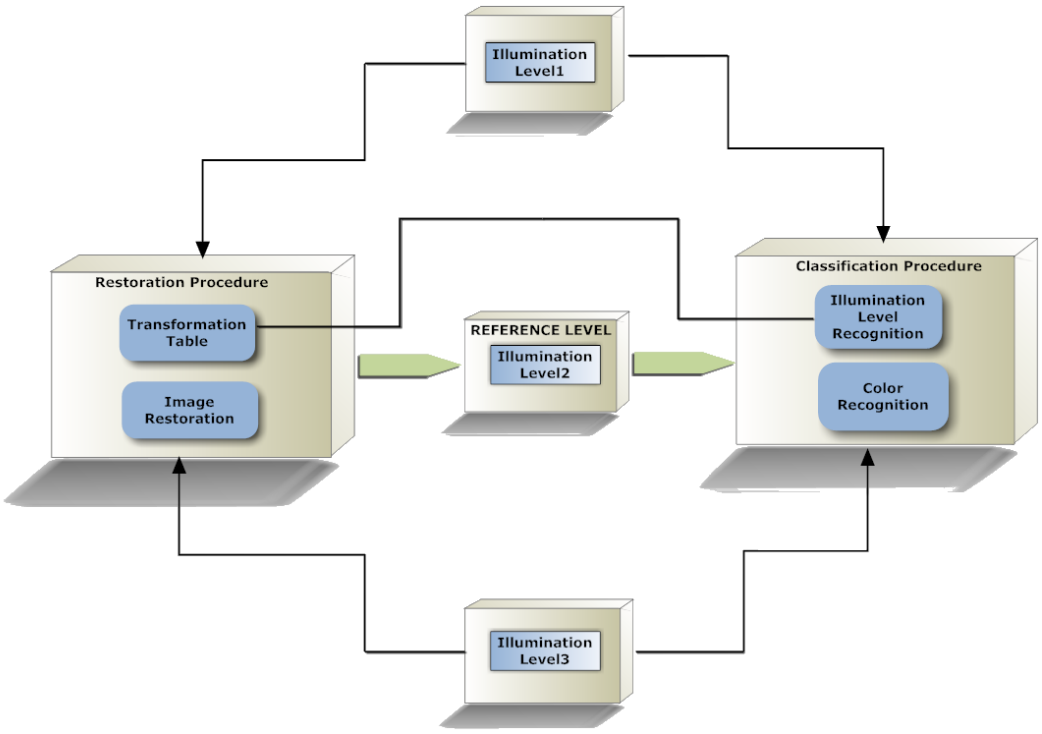


FIGURE 6.16: HS exp 1.

6.5.6 Color Space Selection

In this experiment we are trying to identify if the any color space affect the results of Histogram Specification Method applycation. Thus, we see that Histogram Specification method is color space depentable. (See Figures) 6.19, 6.20, 6.21, 6.22, 6.23, 6.24.

6.5.7 Rules Efficincy Created Using Histogram Specification Method

In figure 6.25 is described the procedure in which we capture image taken under illumination conditions close to these that histogram specification used to create transformation tables. In Figure 6.26 we can see the testing images along with the resulted images for HS method. In this figure we can see that for figure that illumination was to close to reference illumination transformation was no necessery. Thus, applying illumination level recognition we are sure about the right transformation application.

6.5.8 Rules Efficincy Created Using Illuminant Features

In figure 6.25 is described the procedure in which we capture image taken under illumination conditions close to these that trainig data selection was done. In Figures 6.28, 6.29, 6.30 we can see the testing images using all calssification methods.

~~asdf~~

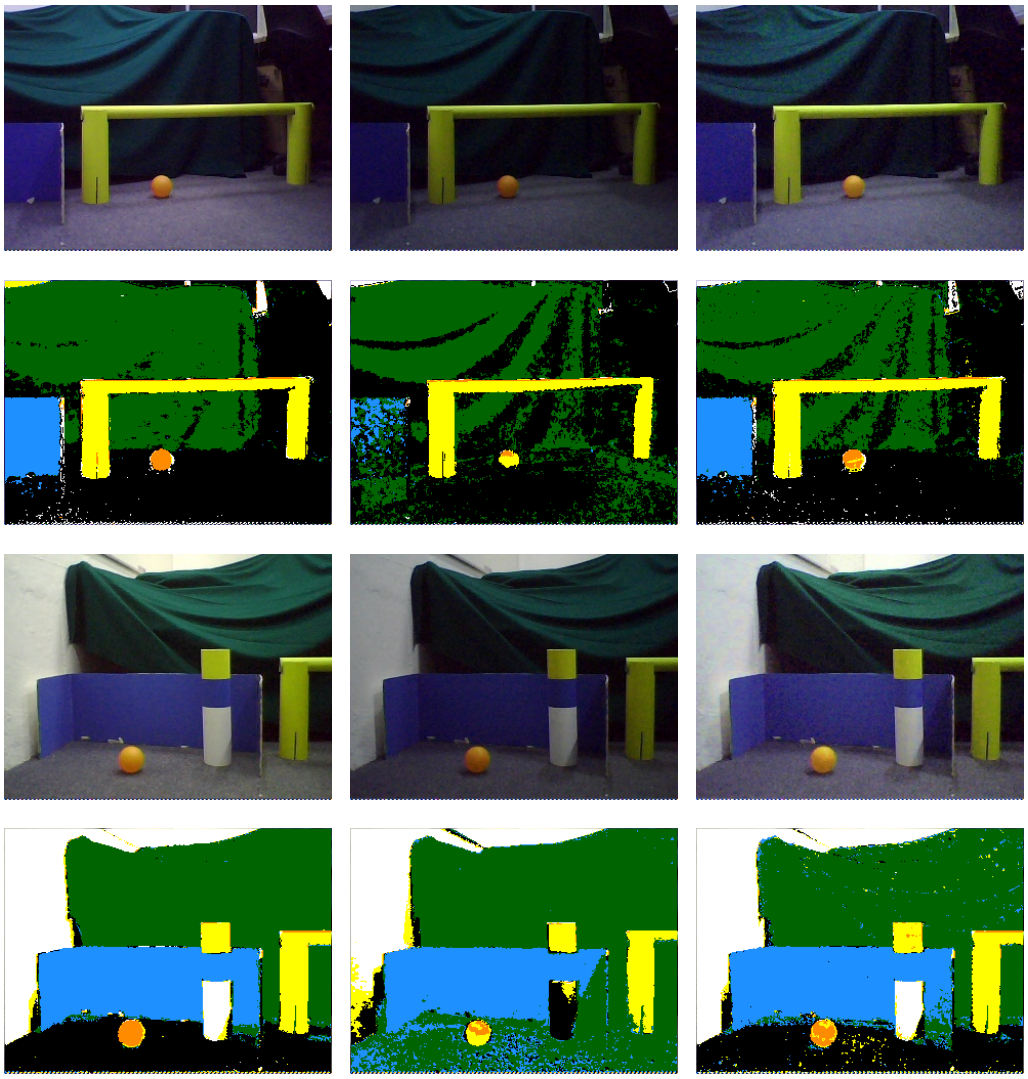


FIGURE 6.17: Histogram Specification in two illumination levels(all images).

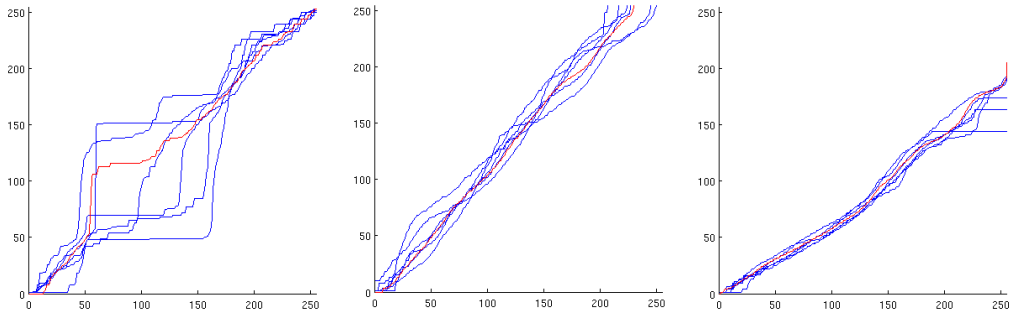


FIGURE 6.18: Histogram Specification transformations in two illumination levels using data from seven images).

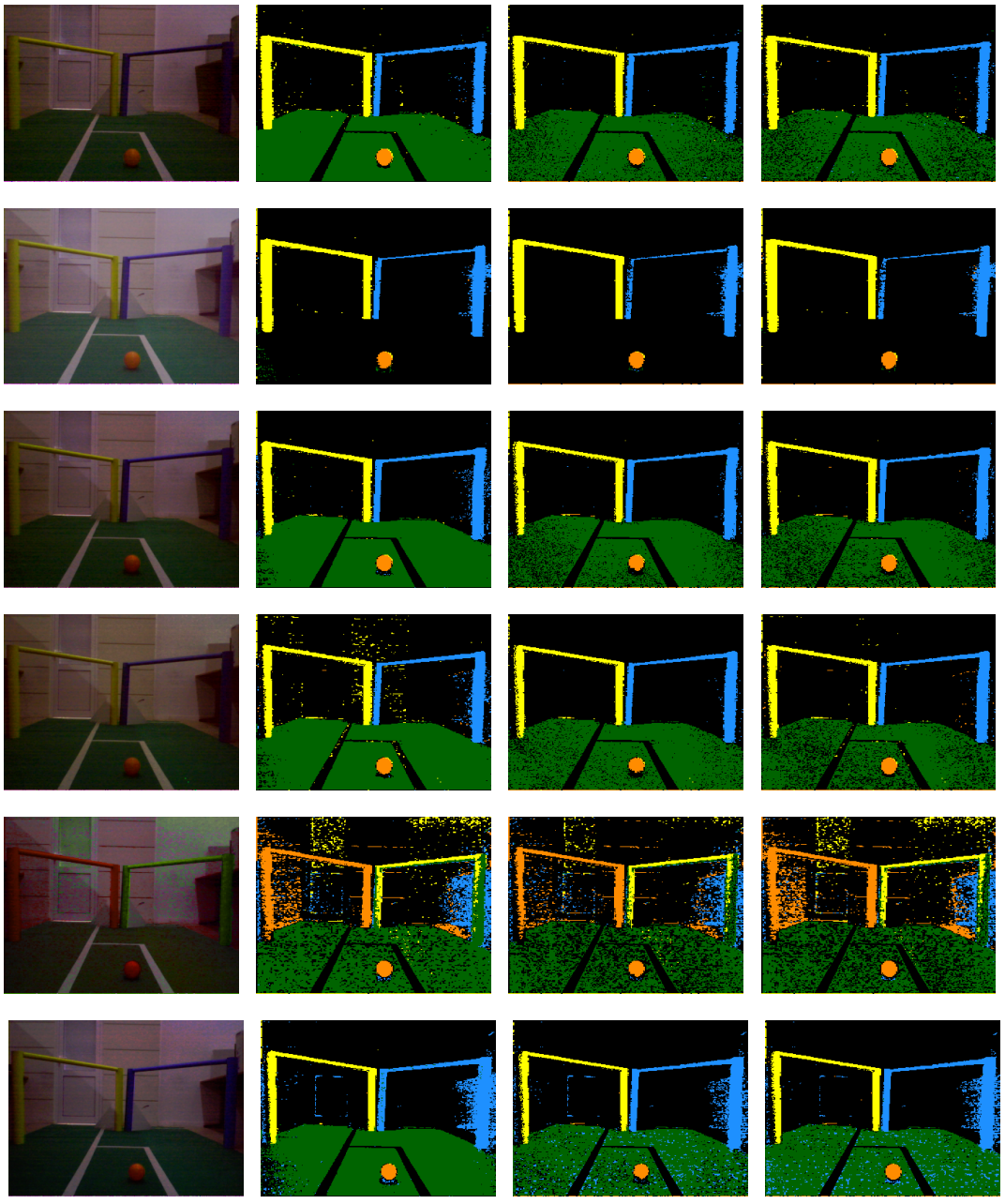


FIGURE 6.19: Histogram Specification in three illumination levels ,reference level1 - level2 (original,init,RGB,YUV,HSV,LAB).

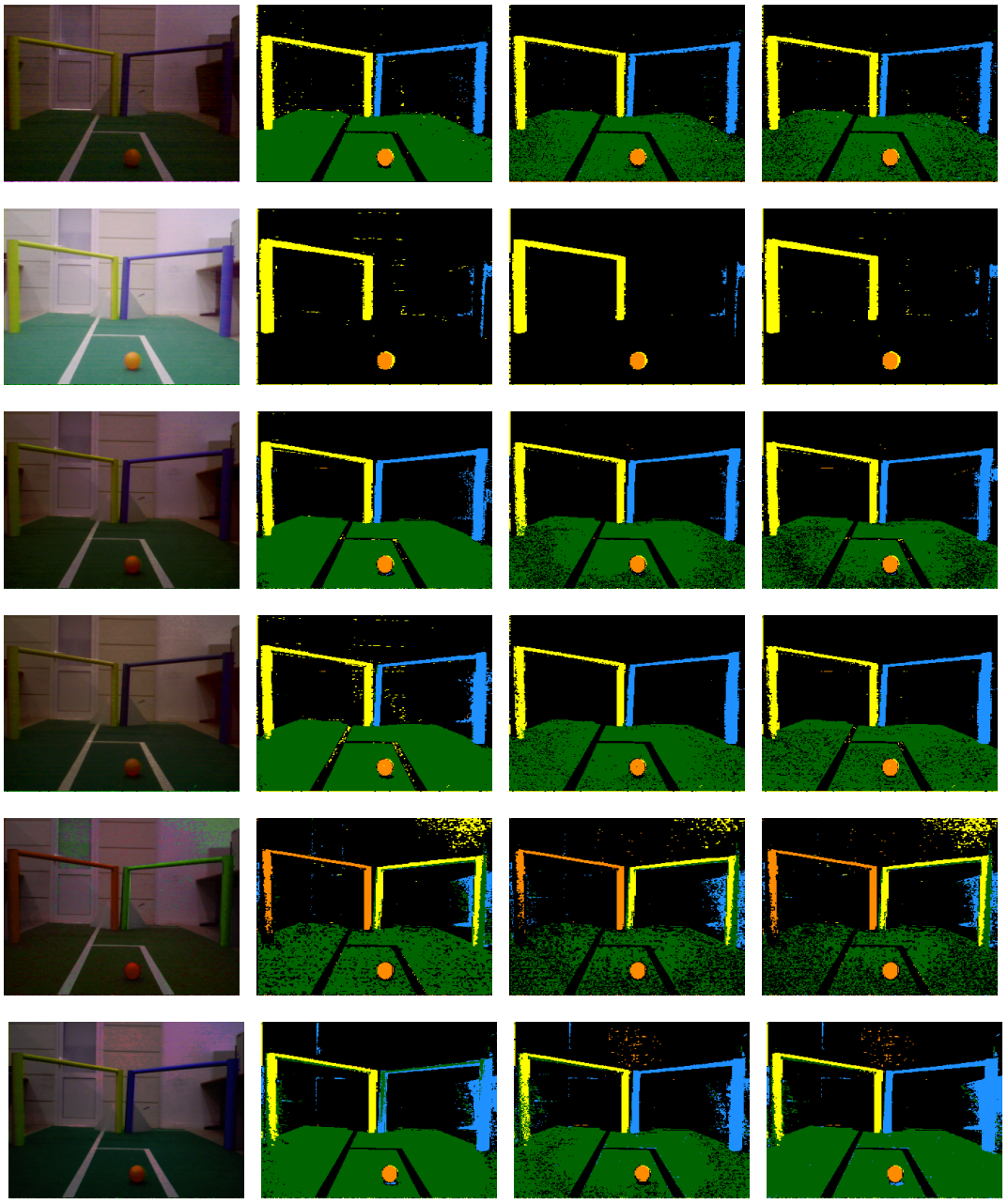


FIGURE 6.20: Histogram Specification in three illumination levels ,reference level1 - level3 (original,init,RGB,YUV,HSV,LAB).

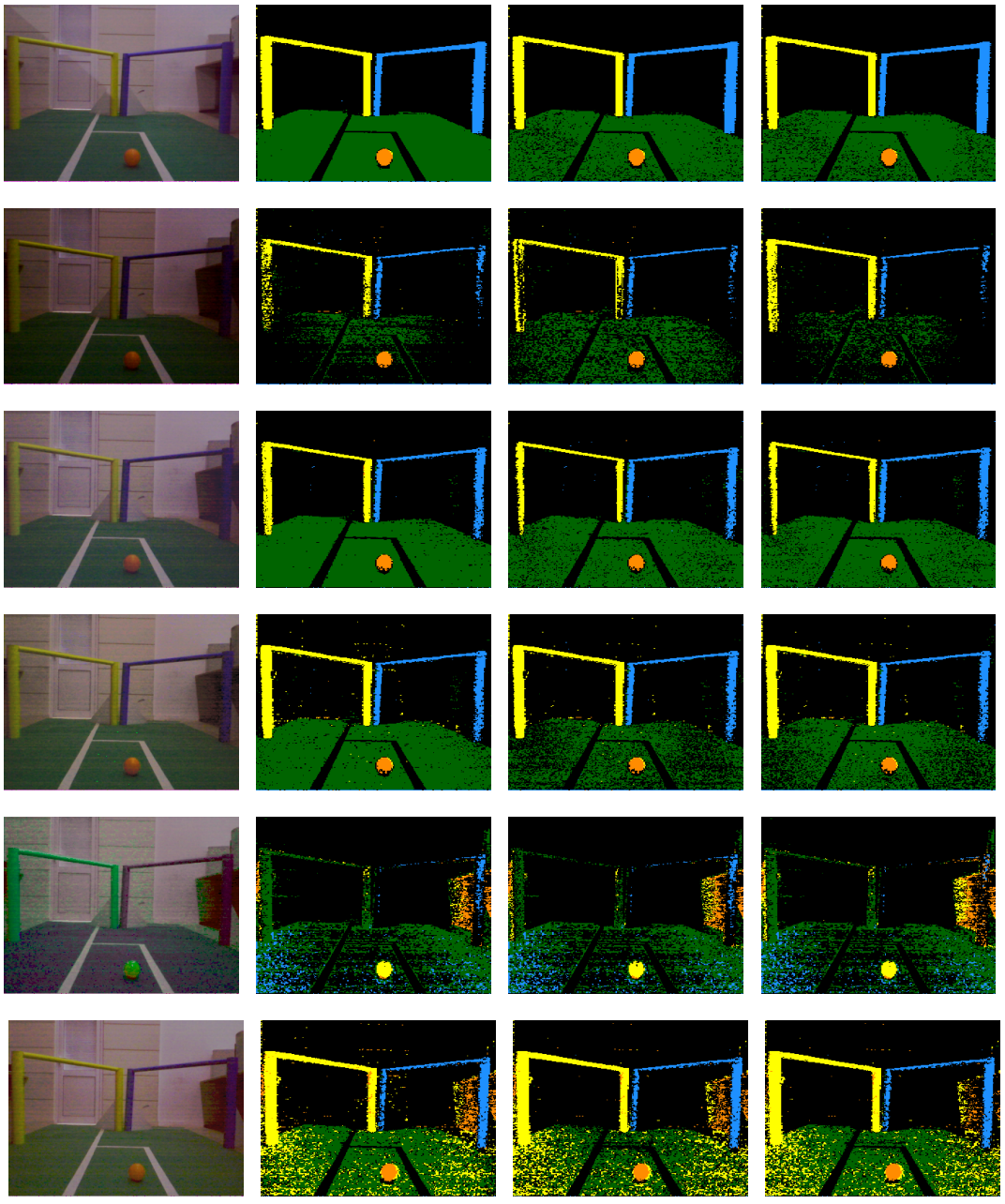


FIGURE 6.21: Histogram Specification in three illumination levels ,reference level2 - level1 (original,init,RGB,YUV,HSV,LAB).

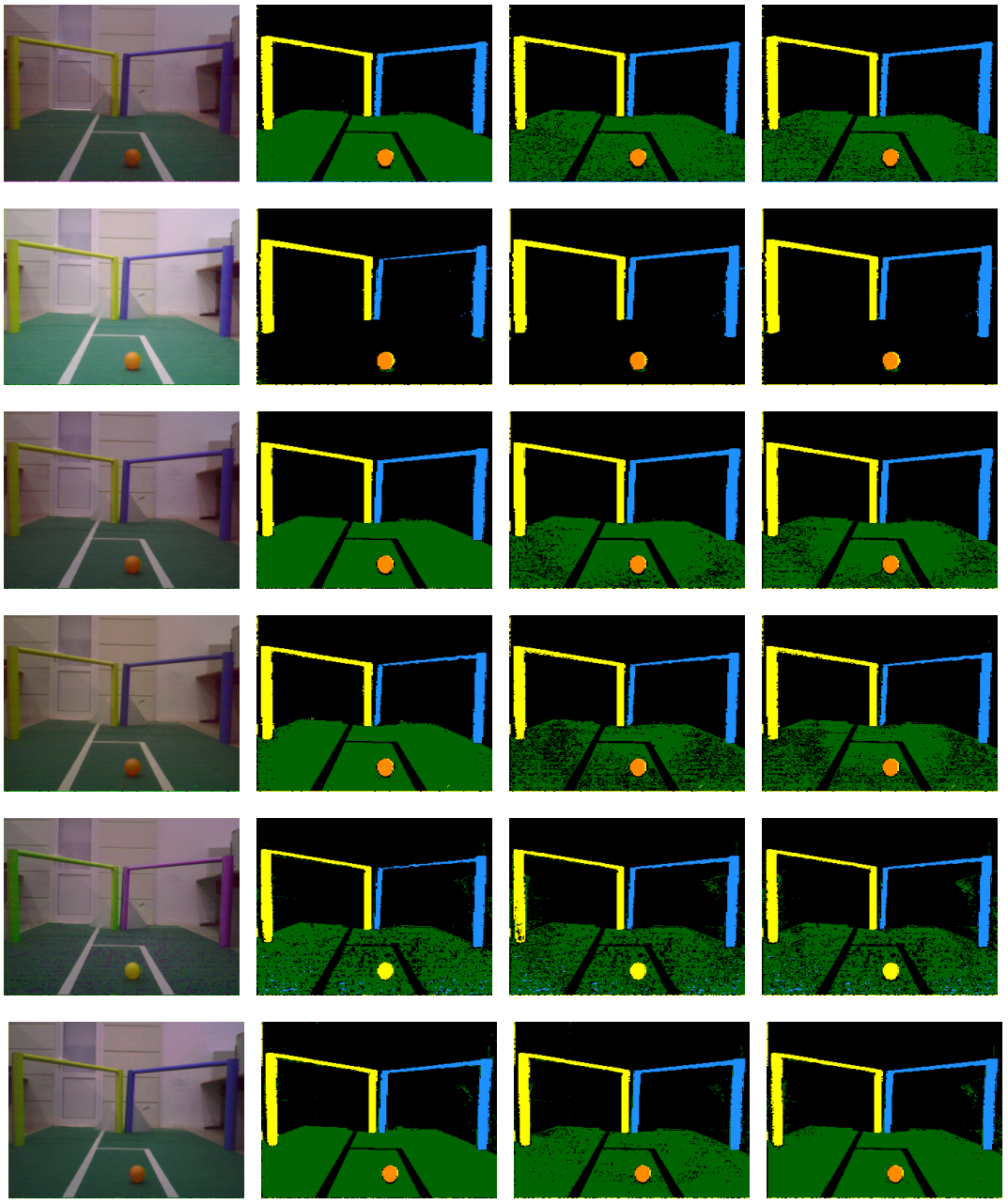


FIGURE 6.22: Histogram Specification in three illumination levels ,reference level2 - level3 (original,init,RGB,YUV,HSV,LAB).

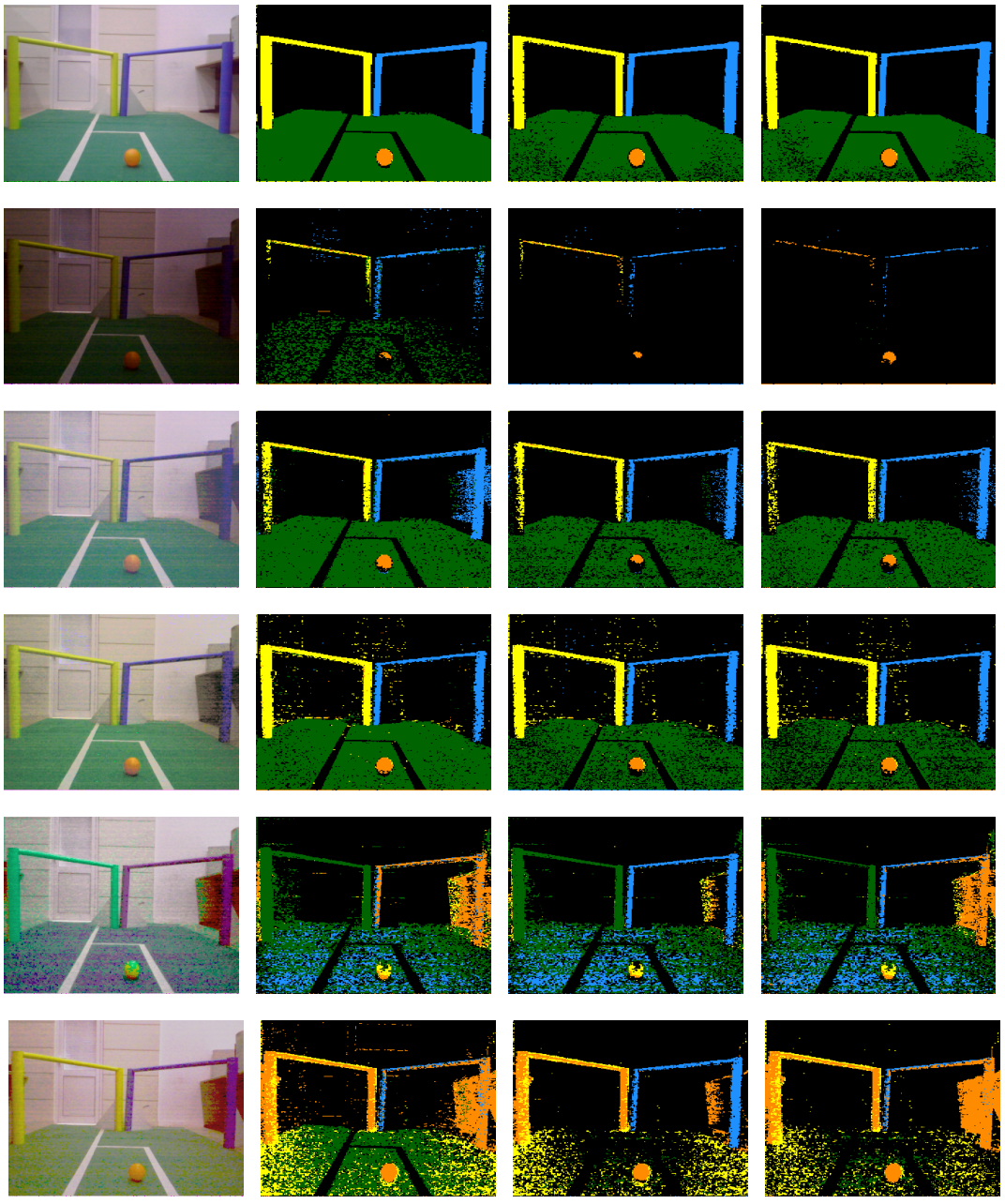


FIGURE 6.23: Histogram Specification in three illumination levels ,reference level3 - level1 (original,init,RGB,YUV,HSV,LAB).

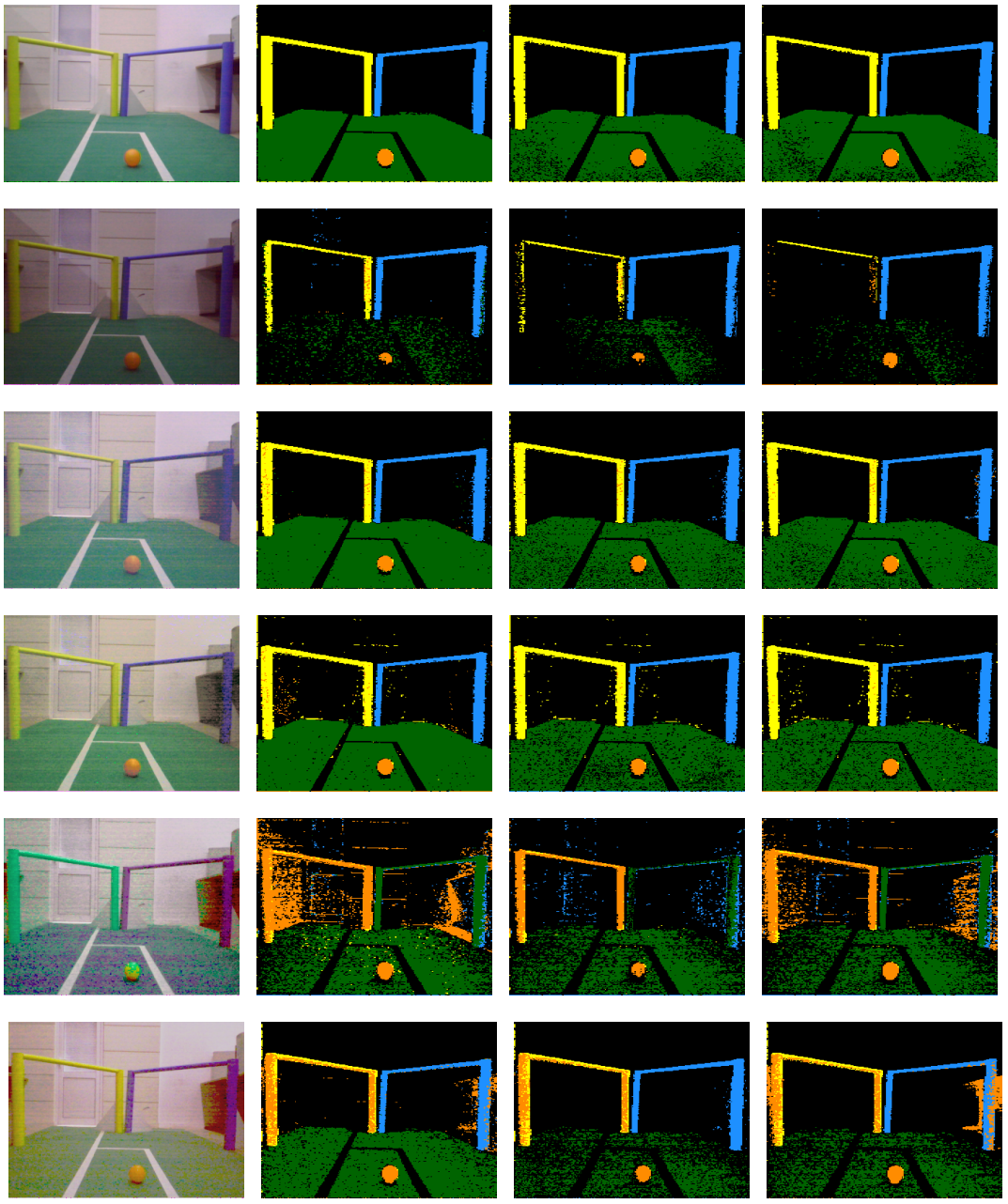


FIGURE 6.24: Histogram Specification in three illumination levels ,reference level3 - level2 (original,init,RGB,YUV,HSV,LAB).

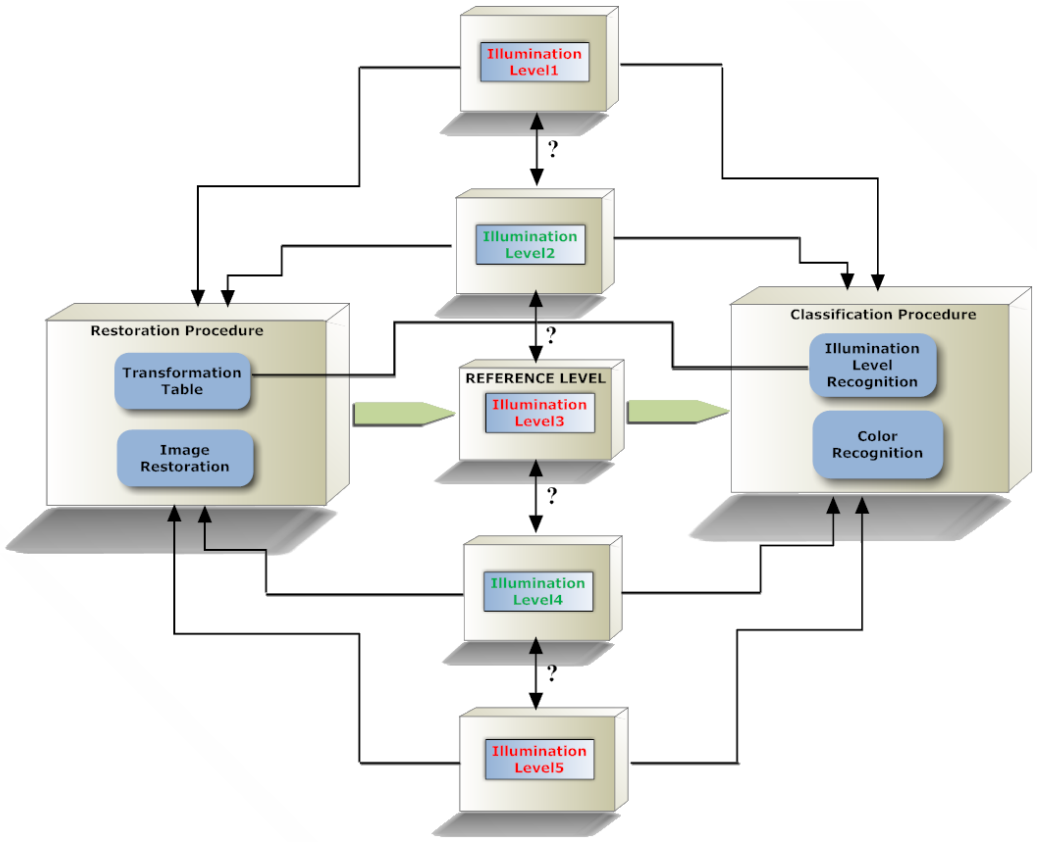


FIGURE 6.25: HS exp 2.

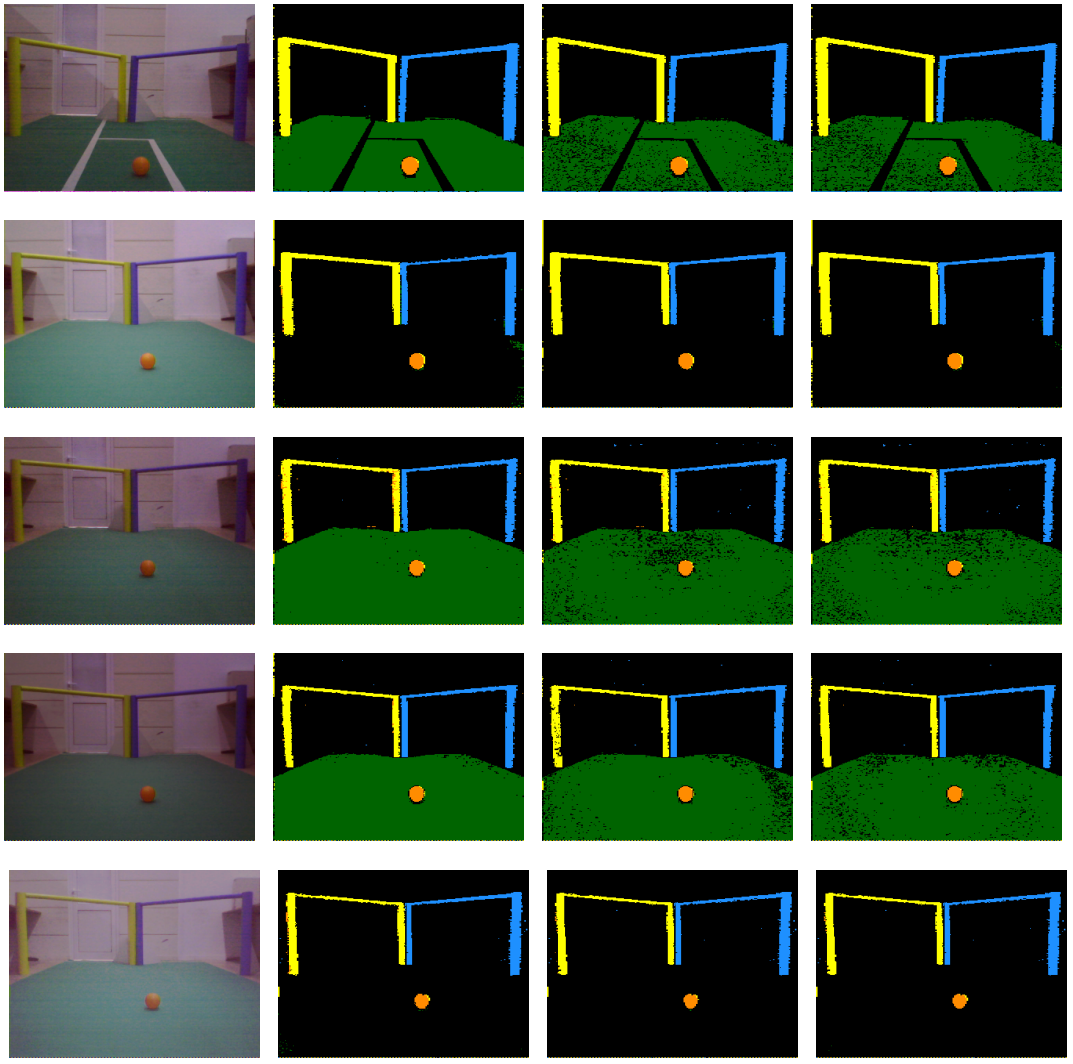


FIGURE 6.26: Histogram Specification in 5 illumination levels ,reference level3.

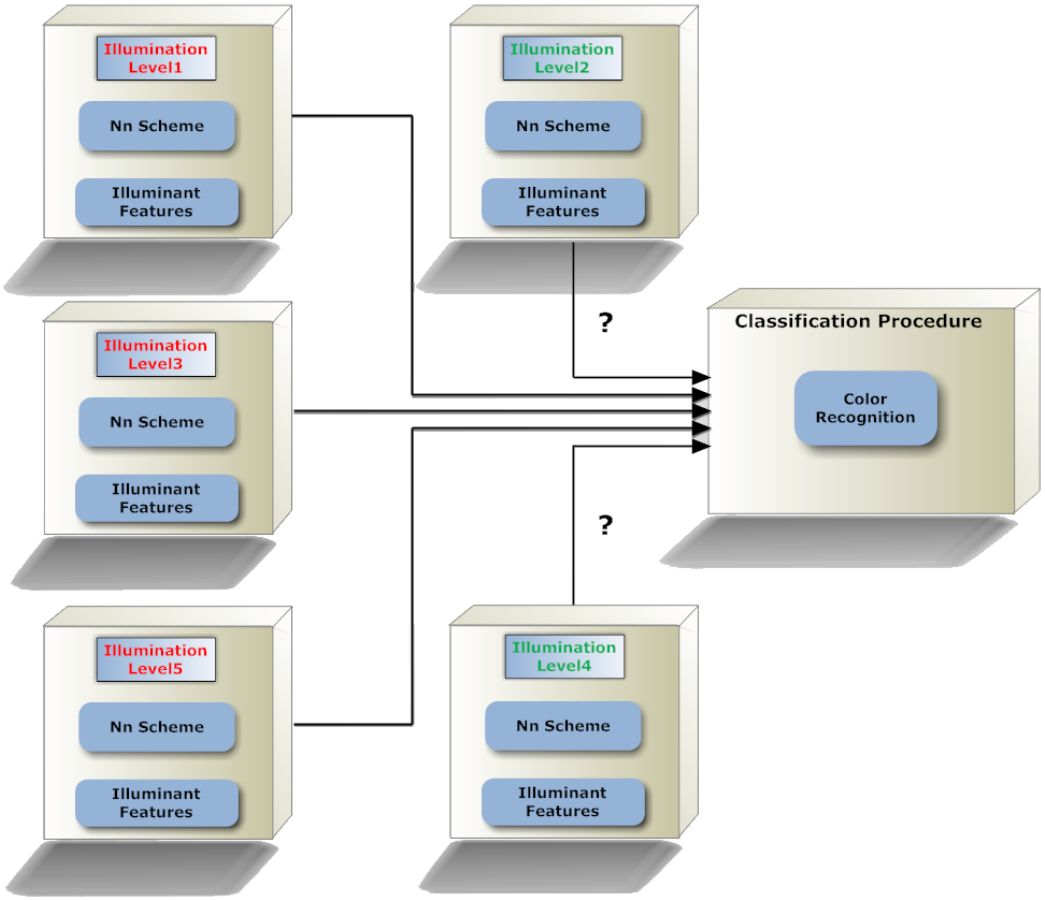


FIGURE 6.27: ILL exp 2.

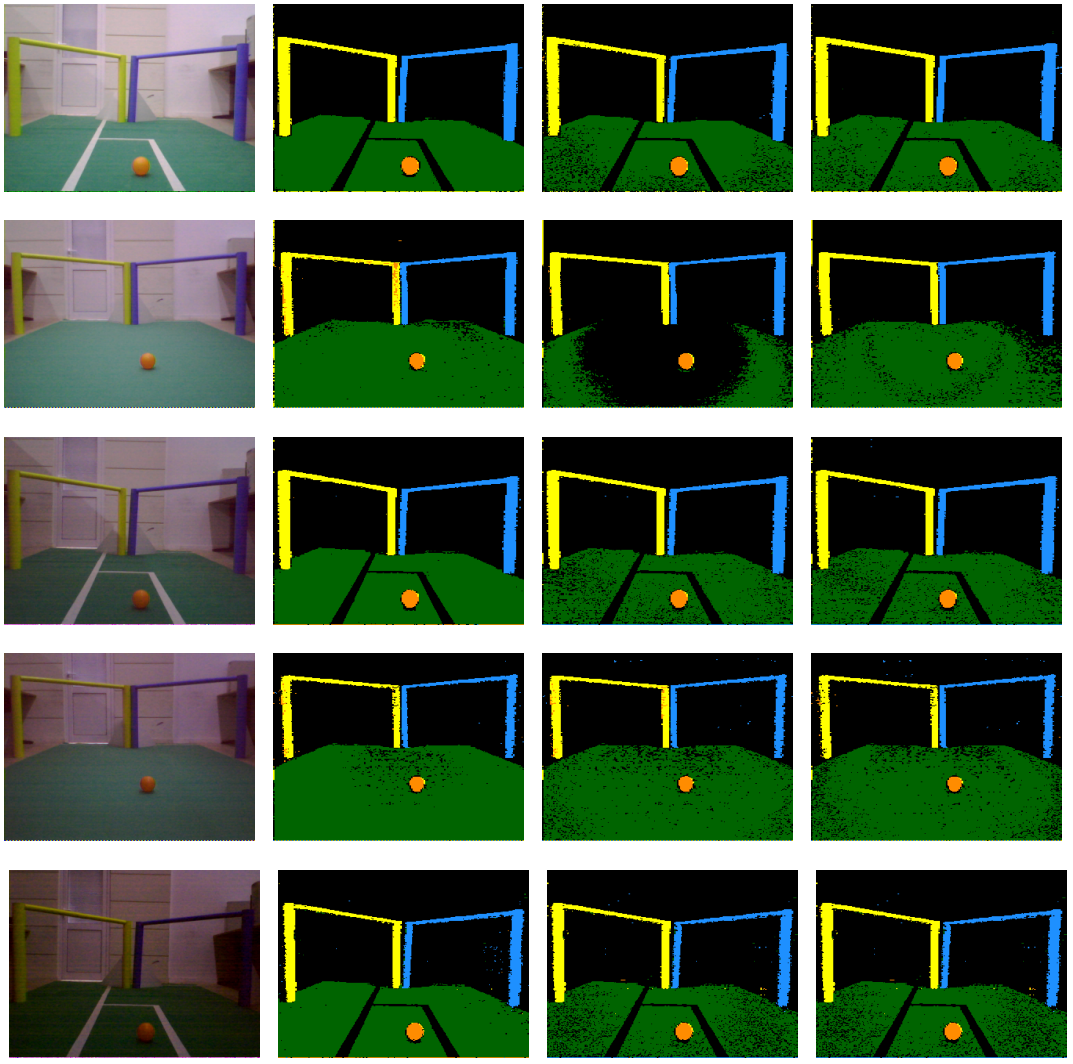


FIGURE 6.28: DT classification in 5 illumination levels ,illuminant

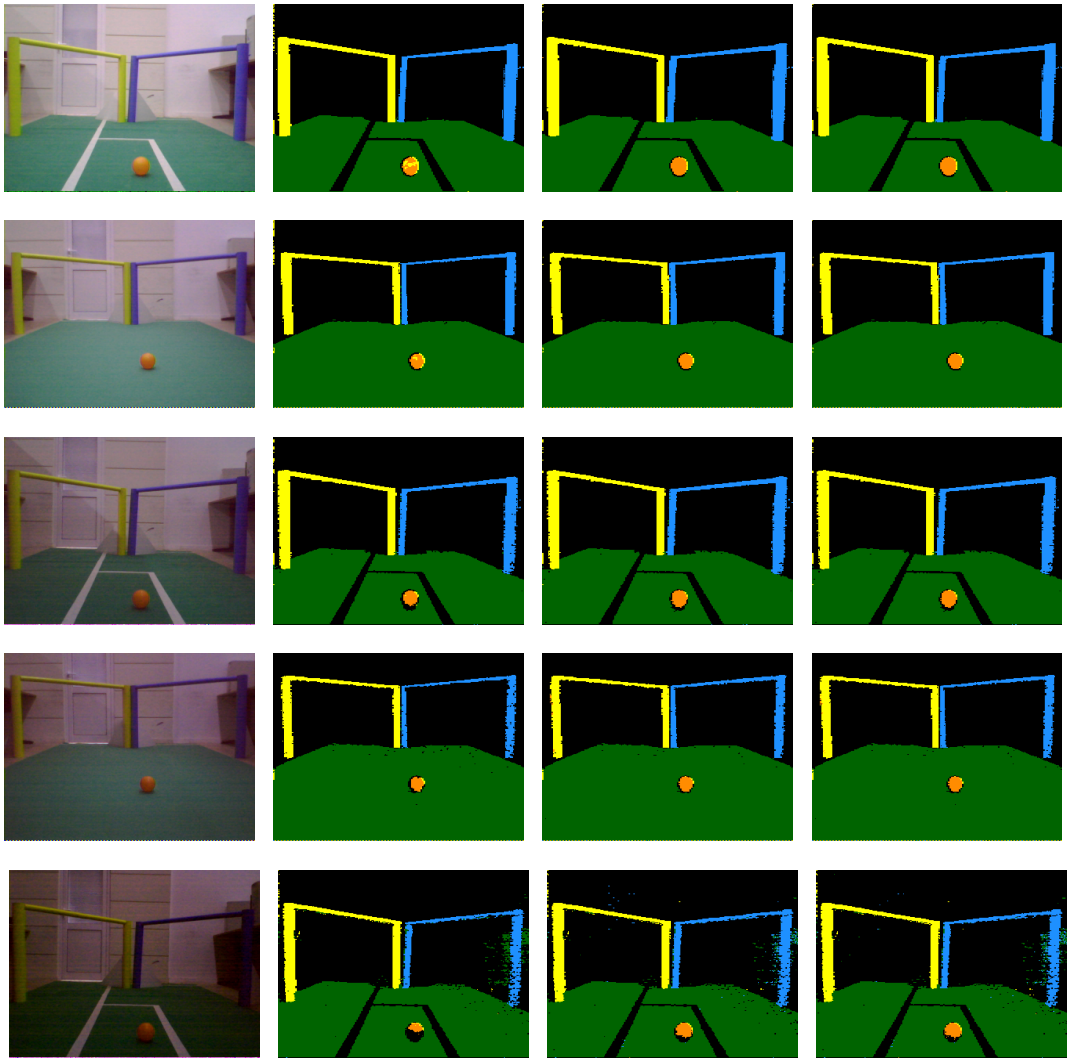


FIGURE 6.29: SVM classification in 5 illumination levels ,illuminant

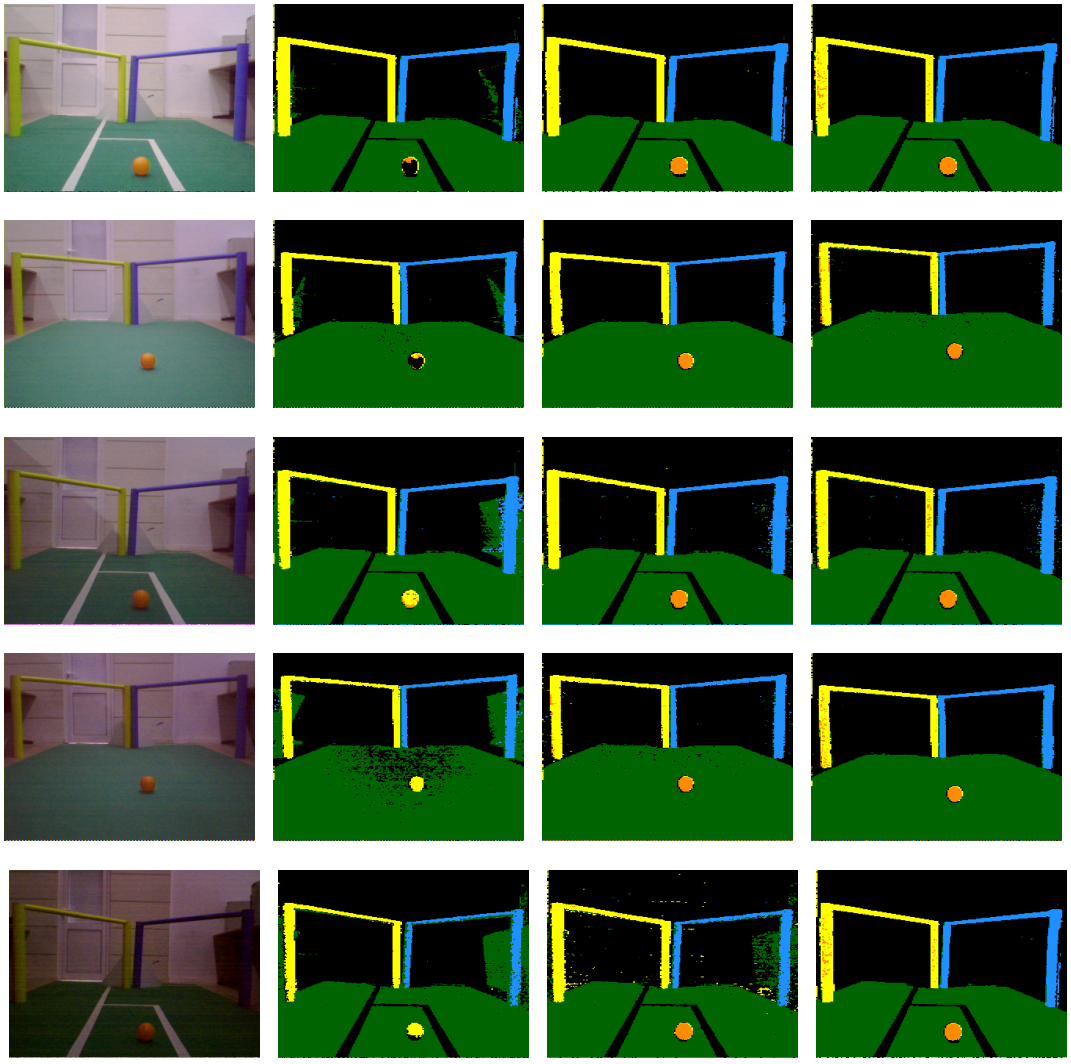


FIGURE 6.30: NN classification in 5 illumination levels ,illuminant

CHAPTER 7

Discussion and Conclusion

We have observed that there are several critical areas when considering machine vision as the primary sensory input for mobile autonomous robotics. In this thesis we have presented significant contributions to each of these areas in order to develop a robotic vision system that is fast, robust, tolerant to changing conditions and easily adaptable. There are many areas that still need to be addressed and many techniques that we have not examined. Nevertheless, we built build a vision system that can be used not only in RoboCup but also can be suitable for implementation on robots with low computational resources and effective in presence of noise and illumination changes. Experimental Results This section shows the most important experiments that were performed to evaluate the reliability of the vision system. It was tested, if the vision system is able to recognize all objects, how the vision system reacts to changing lighting conditions, and how time consuming the calculations of the vision system are.

7.1 Strengths and weaknesses

The method presented in this thesis, has a number of advantages that makes suitable for mobile robotic vision system that works under time constraints. One of the greatest advantages of the proposed scheme, is its simplicity. It is very easy to implement, both conceptually and in terms of the amount of code required. Requires little, or no tuning having very low computational requirements. However, our algorithm is not a perfect solution to the problem. While our algorithm is successful over a very wide range of illumination conditions, if the lighting conditions change significantly then our algorithm will still fail.

7.2 Future Work

Current effort focuses on creating an automatic method for configuring camera settings. This new method will replace the manual configuration by setting the gain, the exposure, and the white balance using as metric the separability of the color classes. As a result the robustness of our classification method will be increased and the rate of the misclassified data will be decimated.

7.3 Lessons Learned

Color recognition is considered a very important task. Since the robot depends on knowing its the position based on object recognition and localization, color recognition automatically becomes a necessary condition. Regarding what has been acquired through this thesis, thebenefits can be counted in knowledge and experience. Firstofall, knowledge has been acquired in the fields of robotics, artificial intelligence, imageprocessing(colorsegmentation and classification, several color spaces encodings). Last but not least, since the vision module is the only one needed and used by the robot in order to be able to play soccer within the field, discussion and cooperation with the people in the team, who had undertaken modules highly related to vision, wereessential. Thus, theexperience of working within a team through this thesis has been invaluable in developing collaboration and communicationskills.

BIBLIOGRAPHY

[1] Y. Kuniyoshi I. Noda E. Osawa H. Kitano, M. Asada and H. Mat-subara. A challenge problem for ai. *AI Magazine*, 18(1):73–85, 1997.

[2] SPL Technical Committee. robocup spl (nao) rule book,. 2009. URL http://www.tzi.de/spl/bin/view/Website/WebHome.

[3] Adrian Ford and Alan Roberts. Colour space conversions. August 1998.

[4] R.J Quinlan. C4.5: Programs for machine learning. Morgan Kaufmann (1993).

[5] Lovell Nathan Hains and Vladimir Estivill-Castro. Color classification and object recognition for robot soccer under variable illumination. *Vienna, Austria: I-Tech Education and Publishing*, pages 71–94, 2007. URL http://s.i-techonline.com/Book/Robotic-Soccer/ISBN978-3-902613-21-9.html.

[6] Luca Iocchi. Robust color segmentation through adaptive color distribution transformation. *RoboCup 2006: Robot Soccer World Cup X*, 4434/2007:287–295, September 04 2007.

[7] Humberto Martnez Barbera Juan Jose Alcaraz Jimenez Carlos A. Zegarra Ortiz David Herrero Perez. Teamchaos robocup, team description paper. 2008.

[8] The numanoids team report. 1 November 2008. URL http://www.robots.newcastle.edu.au.

[9] Mohan Sridharan and Peter Stone. Towards illumination invariance in the legged league. *Springer Verlag*, pages 196–208, Berlin, Germany, 2005. URL http://www.cs.utexas.edu/~pstone.

[10] Matthias Jongel Jan Hoffmann Martin Lotzsch. A real-time auto-adjusting vision system for robotic soccer. *Robot world cup soccer and rescue competitions and conferences*, 3020:214–225, 2004.

[11] Xiaohu Lu and Hong Zhang. Color classication using adaptive dichromatic model. *Robotics and Automation. Proceedings 2006 IEEE International Conference*, pages 3411 – 3416, May 2006.

[12] Pablo Guerrero Javier Ruiz del Solar Josu Fredes Rodrigo Palma-Amestoy. Automatic on-line color calibration using class- relative color spaces. *Springer, RoboCup 2007: Robot Soccer World Cup XI*, 5001/2008:246–253, July 18 .