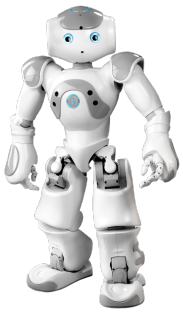
Reliable Object Recognition for the RoboCup Domain



Emmanouel Orfanoudakis

Intelligent Systems Laboratory

Department of Electronic and Computer Engineering

Technical University of Crete

Committee:

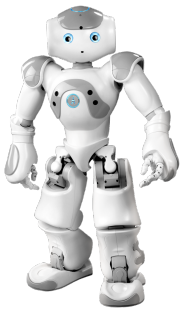
Assistant Professor Michail G. Lagoudakis (supervisor)

Professor Michael Zervakis

Professor Minos Garofalakis

Chania, December 2011

Αξιόπιστη Αναγνώριση Αντικειμένων στο Περιβάλλον του RoboCup



Εμμανουήλ Ορφανουδάκης

Εργαστήριο Προγραμματισμού &
Τεχνολογίας Ευφυών Υπολογιστικών Συστημάτων

Τμήμα Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών

Πολυτεχνείο Κρήτης

Επιτροπή:
Επίκουρος Καθηγητής Μιχαήλ. Γ. Λαγουδάκης (επιβλέπων)
Καθηγητής Μιχάλης Ζερβάκης
Καθηγητής Μίνως Γαροφαλάκης

Χανία, Δεκέμβριος 2011

Abstract

RoboCup is an international robotic soccer competition aiming at advancing the state-of-the-art in autonomous robotics and artificial intelligence. Real-time visual object recognition is a key problem in robotics and becomes particularly challenging in RoboCup due to the volatility of the environment, the scarcity of resources, and the needs for reliability and efficiency. This thesis describes KVision, a light-weight image processing method for visual object recognition on articulated mobile robots with a head-mounted camera, such as humanoid robots, focusing on reliability and efficiency. The proposed vision pipeline uses real-time sensory information from the joints along with the robot's kinematic chain to determine the exact camera position in the 3-dimensional space and subsequently the view horizon and the sampling grid, so that scanning is approximately uniformly projected over the ground (field), not over the image matrix. Special attention is paid on the precise synchronization of images with robot joint values using time-stamped data messages. The next phase employs a noise-tolerant, auto-calibrated color recognition scheme on the pixels of the sampling grid to identify regions of interest. In the last phase, detailed analysis of the identified regions of interests seeks potential matches for the corresponding target object. These matches are evaluated and filtered by several heuristics, so that the best match (if any) in terms of color, shape, and size for a target object is finally extracted. The proposed method has been implemented on the Aldebaran Nao humanoid robots for visual recognition of the ball, the goal posts, and obstacles in the Standard Platform League of the RoboCup competition. The proposed machine vision system has been deployed by the Technical University of Crete RoboCup team "Kouretes" in various RoboCup competitions with remarkable results in terms of reliability and efficiency.

Περίληψη

Το RoboCup είναι ένα διεθνές πρωτάθλημα ρομποτικού ποδοσφαίρου που στοχεύει στην προώθηση της τεχνολογίας στην αυτόνομη ρομποτική και την τεχνητή νοημοσύνη. Η οπτική αναγνώριση αντικειμένων σε πραγματικό χρόνο είναι ένα βασικό πρόβλημα στη ρομποτική και γίνεται ιδιαίτερα δύσκολο στα πλαίσια του Ροβοῦπ, λόγω της αστάθειας του περιβάλλοντος, της περιορισμένης διαθεσιμότητας πόρων και της ανάγκης για αξιοπιστία και αποτελεσματικότητα. Αυτή η διπλωματική εργασία περιγράφει το KVision , μια μέθοδο επεξεργασίας εικόνας για την οπτική αναγνώριση αντικειμένων από αρθρωτά κινητά ρομπότ με κινούμενη κάμερα, όπως ανθρωποειδή ρομπότ, δίνοντας έμφαση σε θέματα αξιοπιστίας και αποδοτικότητας. Η προτεινόμενη αλληλουχία επεξεργασίας χρησιμοποιεί πληροφορίες πραγματικού χρόνου από τους αισθητήρες των αρθρώσεων μαζί με την κινηματική αλυσίδα του ρομπότ για να καθοριστεί η ακριβής θέση της κάμερας στο 3-διάστατο χώρο και στη συνέχεια ο ορίζοντας θέασης και το πλέγμα δειγματοληψίας, ώστε η σάρωση να προβάλλεται προσεγγιστικά ομοιόμορφα πάνω στο έδαφος (γήπεδο) και όχι στο πλαίσιο της εικόνας. Ιδιαίτερη έμφαση δίνεται στον ακριβή συγχρονισμό των εικόνων με τις τιμές των αρθρώσεων του ρομπότ κάνοντας χρήση χρονισμένων μηνυμάτων δεδομένων. Η επόμενη φάση εφαρμόζει μια μέθοδο αναγνώρισης χρώματος, η οποία είναι ανεκτική στο θόρυβο και αυτόματα προσαρμόσιμη, στα εικονοστοιχεία του πλέγματος δειγματοληψίας για τον εντοπισμό περιοχών ενδιαφέροντος. Στην τελευταία φάση, μια διαδικασία λεπτομερούς ανάλυσης των προσδιορισμένων περιοχών ενδιαφέροντος επιδιώκει να εντοπίσει υποψήφιες ταυτοποιήσεις για το αντίστοιχο αντικείμενο-στόχο. Αυτές οι ταυτοποιήσεις αξιολογούνται και φιλτράρονται μέσω ευρετικών κανόνων, ώστε να εξαχθεί η καλύτερη ταυτοποίηση (αν υπάρχει) ως προς το χρώμα, το σχήμα και το μέγεθος για κάποιο αντικείμενο-στόχο. Στη συνέχεια το αντίστοιχο αντικείμενο καταχωρείται ως αντιληπτό, μαζί με μια εκτίμηση της τρέχουσας απόστασης και κατεύθυνσης όπου βρίσκεται, καθώς και τις κατάλληλες γωνίες του κεφαλιού για την εστίαση και παρακολούθησή του. Η προτεινόμενη μέθοδος έχει υλοποιηθεί στο ανθρωποειδές ρομπότ Aldebaran Nao για την οπτική αναγνώριση της μπάλας, των δοκαριών των τερμάτων, και εμποδίων στο Πρωτάθλημα Προκαθορισμένης Πλατφόρμας (Standard Platform League) του διαγωνισμού RoboCup . Το προτεινόμενο σύστημα μηχανικής όρασης έχει αξιοποιηθεί από την ομάδα RoboCup του Πολυτεχνείου Κρήτης 'Κουρήτες' σε διάφορους διαγωνισμούς RoboCup , με αξιοσημείωτα αποτελέσματα όσον αφορά την αξιοπιστία και την αποδοτικότητα.

Acknowledgements

Despite being the first lines one reads from this text, these are the words that will remain in my heart as memories from my years at the Technical University of Crete as an undergraduate student.

I owe gratitude to many people for their contribution to my finally getting a degree! First the whole RoboCup team, as Kouretes may be just a research project in paper, but it is also a place where a student becomes a close friend, and a friend becomes a smile. A few "sets of reading glasses" stand out. My attending professor, who shared his inspiration for this team with all of us. Next, two dear friends, currently far away from me, who will not get to see me smile with the "precious piece of paper" in my hand: Lefteris and Alex, your guidance was bitter, but vital.

My parents, whose strength and patience I have exhausted. I hope you read this with a smile. The person dearest to my heart, Effie: you have been given me the motivation I needed for the last push. I promise to make up for the lost time.

The rest of you, you won't see your name here, but you know you have a place in my heart.

So long.

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

RoboCup [6] is an annual international robotic competition that was established in 1997. The competition focuses on robotic soccer with the ultimate goal of creating a soccer team of autonomous robots that, by the year 2050, can defeat the winner team of the most recent World Cup. This goal is a motivation to advance autonomous robotics and artificial intelligence. The competition consists of various leagues, each one with different challenges and restrictions.

One of these leagues is the Standard Platform League (SPL). This league selects a standard commercial robot, which is to be programmed by all participating teams. The robot that is currently used in the SPL is the Aldebaran Nao [7]. Our work in this thesis concentrates on this league and on this robot, but it does not have any strong dependencies from them. A generic approach to the problem at hand is taken, so that the developed system can be ported to either other RoboCup leagues, or to similar autonomous robotics problems. The goal of our work is to describe, analyze, and develop an *object recognition* scheme, which can overcome the specific difficulties presented in the RoboCup soccer fields and provide solutions suited to the limited on-board capabilities of the Nao and other similar robots. The proposed system utilizes visual information provided by the video camera the robot is equipped with. Other internal sensors are integrated into the solution, with the intent to combine all available information into a unified approach that dominates over other simplistic approaches to the problem.

Real-time object recognition using video information is an open problem in robotics, and a particularly challenging one in RoboCup due to the volatility and the complexity of the environment. These characteristics, combined with the scarcity of resources on a fully autonomous mobile platform, raise the need for a solution that is reliable and

efficient.

This thesis describes KVision, a light-weight image processing method for visual object recognition on articulated mobile robots with a camera, such as humanoid robots, focusing on fast operation and robustness to the volatile and noisy environment of the SPL soccer field. Our vision pipeline uses real-time sensory information from the robot joints, along with the robot's kinematic chain to determine the precise position of the camera in the 3-dimensional space, and makes best use of this information along with the camera image to identify objects in the environment.

1.1 Motivation and Contribution

Previous approaches to the problem typically study various aspects of it, as independent simpler sub-problems:

- The control and operation of the visual sensor (camera).
- The processing of kinematics information from other sensors.
- The processing of the image to detect objects and extract measurements from them possibly utilizing kinematics information.

We aim at improving this approach by integrating all these aspects of the problem to improve robustness and efficiency. The management of the visual sensor is fused with the image processing to gain feedback from the environment and maximize the quality of the video stream. Image processing is tightly coupled with information from other sensors (joint encoders, gyroscopes). Object detection not only validates the results using this information, but also unifies visual data with kinematics to deliver a robust detection model that relies on physical a-priori information about the target objects.

Special attention is paid to the precise synchronization of images with robot joint values using time-stamped data. Other approaches typically deal with this issue using approximations.

A color segmentation technique, that uses feedback from the camera for additional robustness, is utilized to extract regions of interest on the image. An efficient sampling scheme that relies on the detection of the ground plane is used to achieve this, scanning efficiently without creating blind spots in the environment. Detailed analysis of the identified regions of interests seeks potential matches for the corresponding target object. These matches are evaluated and filtered by several heuristic rules, so that the best match (if any) in terms of color, shape, and size for a target object is

finally extracted. Then, the corresponding object is returned as perceived, along with an estimate of its current distance and bearing as well as appropriate head joint angles for fixating and tracking it.

The proposed scheme provides robustness, efficiency, and reliability over traditional methods. This vision system has been deployed by the Technical University of Crete RoboCup team "Kouretes" in various RoboCup competitions with remarkable results.

1.2 Outline

Chapter 2: Background This chapter defines and analyzes fundamental concepts behind the proposed system. Important details are also defined and discussed there and these concepts are needed at various points along. However, the experienced reader might find it useful to only skim though the chapter.

Chapter 3: Problem Statement The outline of the required functionality is described here, along with brief review of previous related work.

Chapter 4: Our Approach The design of the system is defined, and discussed in this chapter. Aspects of the problem at hand are dealt with individually to produce a complete solution with the desirable features and improvements over past work.

Chapter 5: Implementation Details of the implementation of the system are provided, focusing on those that are deemed to be critical for the achievement of an optimized result. Further analysis of various components is also included..

Chapter 6: Results A quantification and evaluation of the resulting scheme is included here. Experimental evidence of the effectiveness of the system is also presented.

Chapter 7: Conclusion and Future Work Open issues are presented, with a brief outline of possible solutions. Various future enhancements to the present scheme are proposed. The outcome of this work is also presented.

Chapter 2

Background

2.1 Robocup Environment

In its short history, the RoboCup competition [6] has grown to a well-established annual event bringing together the best robotics researchers from all over the world. The uniqueness of RoboCup stems from the real-world challenge it poses, whereby the core problems of artificial intelligence and robotics (perception, cognition, action, coordination) must be addressed simultaneously under real-time constraints. The proposed solutions are tested on a common benchmark environment through soccer games in various leagues. A key aspect of most RoboCup leagues is the multi-agent environment; each team features 11 (simulation), 5 (small-size), 6 (mid-size), 3 (humanoid and standard platform league) robots. These robots cannot simply act as individuals; they must focus on teamwork in order to cope effectively with an unknown opponent team and such teamwork requires coordination.

RoboCup is a highly active research domain, that focuses on developing autonomous robotic platforms capable of playing soccer. Using a popular sport as a basis, the aim of the competition is to use the environment of a soccer field to promote research and education in the field of *Artificial Intelligence*. This environment is a challenging autonomous agents for multiple reasons:

Multi-Agent System: All player in the field are autonomous agents split in two opposing teams. Each team's players must cooperate to compete against the other team's players, aiming to score more goals than the opponent. The agents must coordinate their movement, objectives and combine their knowledge about the current state of the game to achieve an good result.

Partially-Observable Environment: Each agent has limited knowledge about the

field and the environment in general, and all the decisions they make have an inherent probability for mistakes.

Real-time stochastic environment: The physical properties of the agent's motion control and sensory input have an inherent stochasticity. Actions and observations of the agents are imprecise, and have an inherent possibility of failure. Also the environment itself is unpredictable, due to the random nature of disturbances on and off the field.

Resource Scarce Problem: Each agent has limited resources to exploit, and must process sensory data and respond accordingly.

Kouretes RoboCup Team Team Kouretes [8] is the TUC RoboCup team and was founded in February 2006. Team activities at this early stage were restricted to the four-legged league, but the team later extended to the simulation league. The team is currently active in the Standard Platform League. The team's members are mainly students who assist the team while finishing their degree, by writing their theses as an extension to the Kouretes Team code-base.

Standard Platform League The Standard Platform League (SPL) is among the most popular leagues, featuring four humanoid Aldebaran Nao robot players in each team. This league uses a preselected commercial robot. Our work concentrates its efforts to this league, but it does not have any dependencies to it. A generalized approach is followed to the problem at hand, and the developed system can be ported to other leagues, or to similar problems. The robot that is currently used in the league is the Aldebaran Nao.

2.2 Nao and NaoQi Framework

The Nao is currently the chosen hardware platform of the SPL League. To utilize the robot effectively, one must understand its capabilities and limitations.

Hardware The Nao is a 58cm, 4.3Kg humanoid robot developed by Aldebaran Robotics in Paris, France. It is equipped with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM, 2 GB flash memory, and wifi and ethernet networking.

The provided software lies on top of a embedded linux distribution: *OpenNao*, that is based on the popular *OpenEmbedded* linux system.

The available sensors are:

Figure 2.1: The RoboCup Standard Platform League.

- Two color cameras.
- Two microphones.
- Two ultrasound distance sensors.
- An inertial unit (2 gyroscopes and 3 accelerometers).
- An array of force sensitive resistors on each foot, arranged to detect *center of pressure* on the foot.
- A total of 21 joints (4 in each arm, 5 in each leg, 2 in the head, and 1 in the pelvis), with 12-bit precision encoders attached to them. Each joint is powered by a position actuation motor, with an adjustable power output.

The majority of the sensors, with the notable exception of the cameras and the ultrasound distance sensors. Are governed by a *10 ms* cycle, the software can send commands and receives data 100 times a second, on a fairly strict *real-time* execution model.

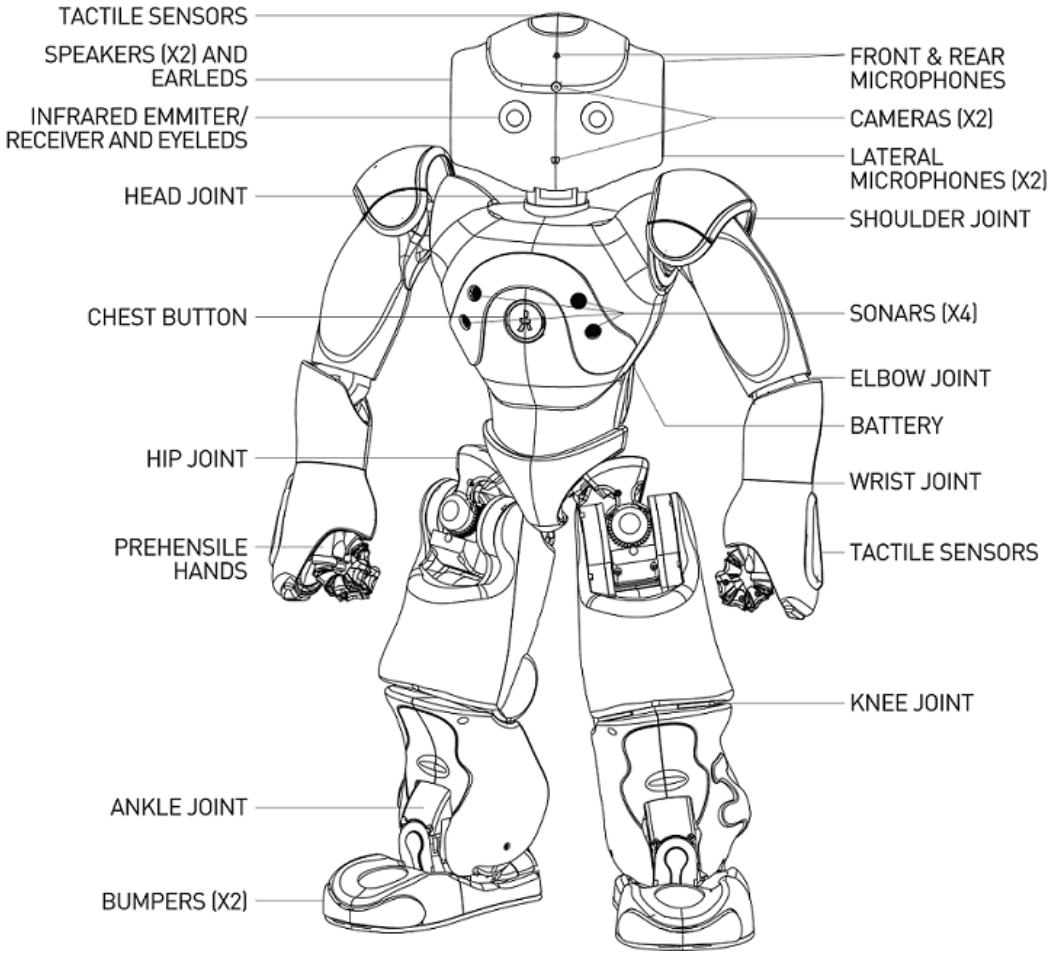


Figure 2.2: The Nao.

Nao Camera The Nao is equipped with a pair of OmniVision OV7670 CMOS cameras [9]. These provide fairly standard VGA images at a rate of 30 frames per second (fps). They are interconnected on an *i2c* bus, and they are accessible *Video for linux 2(v4l2)* arhcitecture. Only one camera is accessible at a given time, with the switch between them possible at any time. Available hardware settings for the camera are:

- Exposure time
- Exposure gain
- R-Channel gain
- G-Channel gain
- Brightness

- Contrast
- Hue

Currently the native output of the sensor is limited to YCbCr422 colorspace, but internally the sensor is of the typical RGB type.

Inertial Sensors The Nao is also equipped with an *inertial measurement unit*. The sensors that are integrated on the unit are:

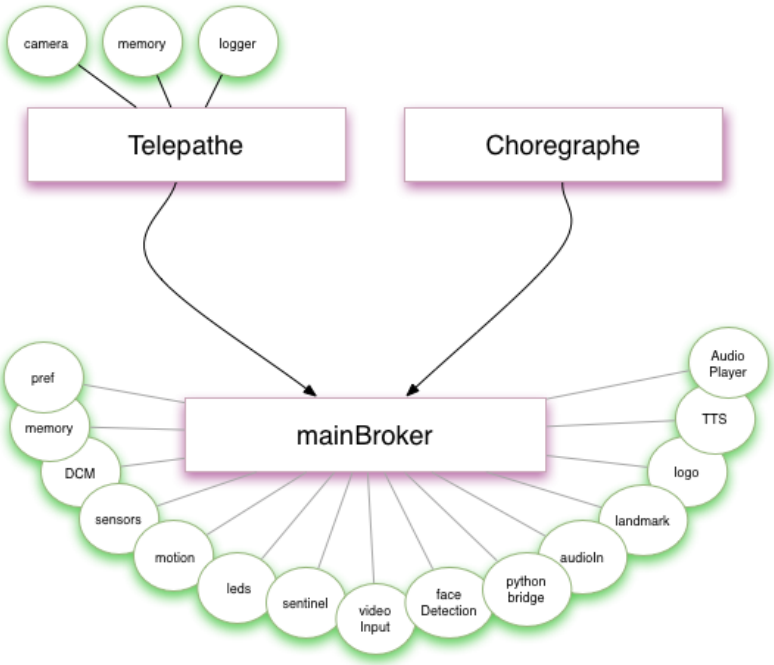
- An *IDG-300* microelectromechanical dual gyroscope [10] that provides angular velocity detection.
- An *LIS302DL* microelectromechanical accelerometer [11] that provides linear acceleration detection.

NaoQi The manufacturing company provides a basic *API framework* to complement the hardware. The framework, named *NaoQi*, relies on independent logical units called *modules* interconnected by a system that provides asynchronous connectivity. Each module is an independent entity, that can be simplified as a collection of code and data. Each module exposes part of its API, as a set of *asynchronously invoked functions-procedures*. Each module, becomes available to the others, through a set of *routing agents*, called *brokers*, where each broker routes each *remote procedure call* to the corresponding module that provides this *functionality*. Execution of each *function-procedure* is materialized on the *host machine* of the module itself. The module utilizes *CPU time* of the host machine by implicitly acquiring ownership of a previously idle *thread* residing in a *thread pool*.

This system provides a logical abstract layer from the underlying hardware, since modules can be *instantiated* and *invoked* on a set of interconnected machines. However, procedure calls routed through *NaoQi* have a noticeable overhead that cannot to be disregarded. Interconnection using *remote calls* should be as minimal and as efficient as possible.

2.3 Kouretes Platform: Monas

Monas [12] is the software architecture in which the code of the team resides. It is designed to be a flexible and configurable multi-threaded programming platform that can be used for multiple purposes and is currently being used and maintained for the



"Naoqi Architecture"

needs of the *Kouretes Team*. It allows concurrent execution of different basic logically-independent units that conform to specific requirements. These units, called *activities*, produce and use information in discrete entities called *messages*.

Messages are read and written on a structure designed for this purpose, following the *Blackboard architecture* [13]. The blackboard API handles messages from arbitrary producers and allows each activity to read a specific message using a complex retrieval scheme.

Each activity must implement and provide a method call that is invoked periodically. During the execution of this specific method, symbolically named `Execute`, each activity is allowed to consume CPU time and exchange messages with the rest of the activities, by feeding to or reading from the accompanying blackboard structure.

The developer is free to organize these activities in coherent groups. Each group forms an *execution thread* that is automatically scheduled for execution in a configurable interval. During each cycle:

- The blackboard structure reads any new information produced by other threads from a specified *synchronization buffer*.

- Each of the activities in the is executed once in a prespecified order.
- The blackboard structure transmits any new messages to other groups, via a specified synchronized buffer.

Each message is routed through an asynchronous architecture, that handles the correct delivery of messages to and from various threads.

2.4 The Eye: Modeling a Familiar Sensor

Computer vision is a diverse and relatively new field of study. In a nutshell, it is concerned with the extraction of information from images. Therefore, it is closely related to the physics of digital image acquisition and digital image representation and is mainly concerned with the signal processing, image processing, and image analysis to guide the extraction of useful information from digital images.

Figure 2.3: "Machines that see"

2.4.1 Biology of Vision

The eye The sense of light has been one of the marvels of evolution. From a simple photosensitive bacteria cell to the more complex eye as we know it, light has become a tool for the survival of all organisms that have the ability to sense it. Guided by evolution, vision in living organisms has resulted in many diverse, and sometimes strange eyes. However, from the compound eyes of the ordinary house fly [Figure 2.4], to the acute vision of predatory birds, such as the eagle, all higher organisms that have organs capable of spatially perceiving light share some basic fundamental elements [14]:

diaphragm: a means of selectively collecting light into the eye with a specific directional sensitivity (i.e. light coming from a particular direction).

lens: a fixed or adjustable assembly that focuses light from the diaphragm onto the photosensitive tissue using the principle of refraction.

photo-receptors: a collection of biological tissue that is capable of detecting light and transforming it into electrical signals sent to the brain through a neural pathway.

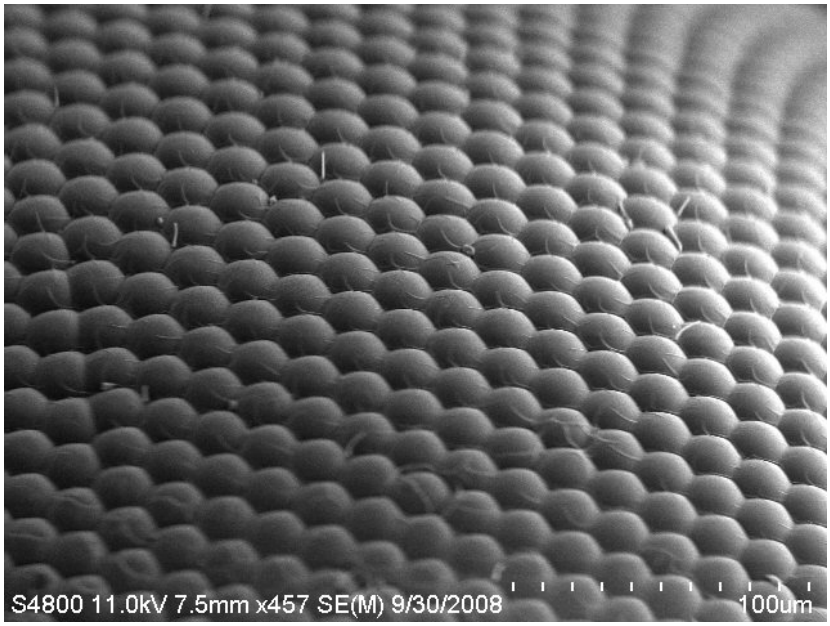


Figure 2.4: House fly eye surface viewed through a Scanning Electron Microscope [1].

Human vision, appearing familiar and simple to an ordinary human, consists of elements that map comfortably to the above list. The human pupil is an excellent

light diaphragm, that allows the brain to control the amount of light entering the eye by expanding and shrinking according to the environment. The lens, located behind the pupil, provide for consistent viewing acuity of objects, regardless of their distance from the viewer. The retina with its photosensitive cells provides for the perception of the light that passes into the eye, and is effectively the mechanism for forming the perception of intensity, color, and acuity of the world around us into an image.

The result is a highly sophisticated organ, which combined with an extraordinary complex processing machine, the brain, produces a wealth of information for the world around us. From the simple detection of light to the perception of shapes and movement, either under direct sunlight or in the darkness of night, the brain cannot be matched by any machine the human has been able to conceive and manufacture.

Binocular vision Binocular vision is a delicate procedure that takes place "behind the scenes" in all two-eyed organisms, and provides for what appears to be an intrinsic ability for us, namely the perception of depth, using the combined processing of the images from both eyes by the human brain. This processing, uses binocular disparity, a phenomenon related to the subtle differences in the images captured by the eyes due to their different positions in relation to the observed scene. By recognizing an object observed by both eyes, the brain also estimates the position from that object in space [14].

Visual Acuity The human retina is a very complex sensor, allowing the detection of color using three basic photo-receptor cell types (cone cells), each one being sensitive to a different (but not mutually exclusive) range of wavelengths, with varying levels of sensitivity [14]. It also provides a color-less imaging pathway using rod cells, which function better in low-light conditions and provide a color oblivious "night vision" image. Therefore, all colors are reduced to three sensory quantities, called the *tristimulus values* [14]. This is not the only configuration in the animal kingdom: most birds' eyes contain a fourth colored photo-receptor cell type, sensitive to the near-ultraviolet spectrum, allowing them to see the world in a way unfamiliar to us. Many nocturnal creatures have extremely better performing vision in low light conditions.

An interesting detail about our vision, is that although our brain perceives the entire world as a continuously coherent image with the same acuity and color vividness, our retinas are only able to provide this fine image only in the very center of the visual field, i.e. when we look directly at an object. The central area of the retina, named fovea [14], contains only cones in maximal density and is highly effective in regular/

intense lighting. The fovea corresponds to only a small amount of our field of view, with the vast majority of the images we see at a given moment being provided by the peripheral retina, which is a coarsely populated area. This is the reason humans unconsciously look directly at the object they intent to observe, so as to provide images with maximal clarity for the brain to interpret. It is merely the brain, filling in the gaps of the environment from previous observations, the real reason we have such detailed views of the surroundings in our perception.

2.4.2 Camera Models

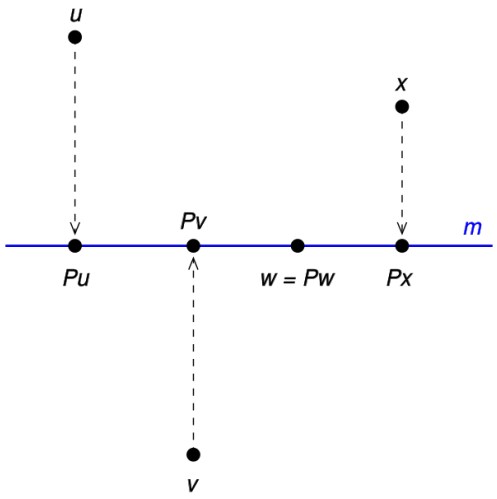
From the early days of photography, capturing devices produce images that are inherently familiar to the viewers. From film photography to digital imaging, precise images are considered those that depict objects as a human would observe them with a naked eye: color, geometric shapes and sizes, preservation of the common properties of the images we see with our eyes and interpret with our brain. This affinity to the biology of vision is what enables us to see familiar shapes and objects in photographs. Regular photography relies on the reproduction of the mechanism we use to see: the eye. By modeling our eyes using a coherent and "meaningful" way, we can produce images that we comprehend as "correct".

Planar Graphical Projection We, humans, have devised multiple methods to have our *Euclidean 3-Dimensional* world captured on a *plane* (an image), each one with different principles and purposes. The mathematical procedure for all projection models share a common purpose: to represent an *N-Dimensional space* ("world") onto an *M-Dimensional* one. The simple example that follows, shows how a set of points is orthogonally projected onto a line.

Notice how the word *orthogonal* is used to define the *planar projection* itself: one could define multiple different types of projections depending on the purpose of the projection itself.

Depending on its properties, or its *invariants*, we categorize projections [16] as follows:

- *Parallel Projections*, where all points of an object's surface are projected onto the viewing plane via a set of *parallel lines (or rays)*. These types of projections have an inherent feature useful to engineering and architectural drawings: all lines that are parallel in the initial space (i.e. 3-D space) remain parallel in the projection space as well. Most common parallel projections are *orthographic projections*, where the projection rays are perpendicular to the projection plane. All

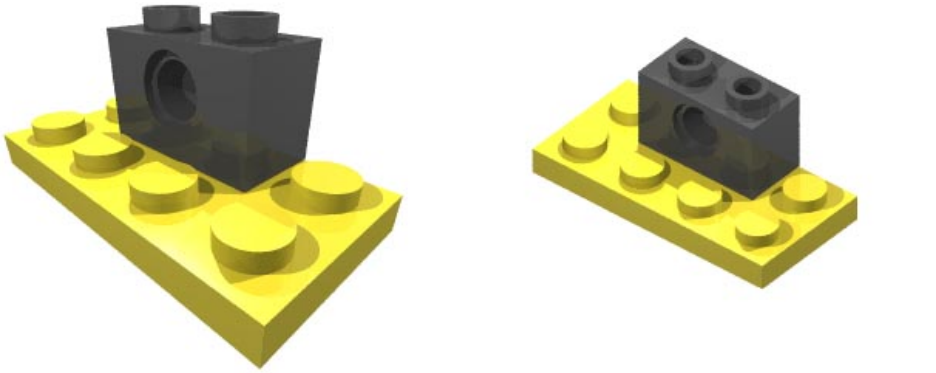


The orthogonal projection of points onto the line m [15].

architectural pictorials, *axonometric projections* (isometic, dimetric, trimetric) are orthographic projections, and therefore are parallel projections. The inherent difference between parallel projections and human perception provides the ability to create what are commonly referred to as "impossible images", that is deceptive images, which the human tries to interpret as typical projections when in fact they cannot be interpreted as such [Figure 2.6].

- *Perspective Projections*, where all points of an object's surface are projected onto the viewing surface though a defined point, called the *focal point* - all projection rays intersect at this point. This projection is the algebraic modeling of the human eye and therefore all common imaging methods are imperfect *perspective projectors* onto a plane where the image capturing occurs, either a camera film or a digital sensor. In the extreme, placing the focal point at an infinite distance from the projection plane makes this perspective projection a parallel projection, as the lines passing from the focal point must be parallel to each other to strike the plane which lies at an infinite distance. Perspective Projection does not preserve parallel lines, but it does preserve linearity. All collinear points in the initial space (i.e. 3-D space) appear collinear in the projection space as well.

All projections have a well defined algebraic modeling, but perspective projections are not *linear transformations* and they require further modeling. Examples of both categories of projections are shown in Figures 2.5(a) and 2.5(b). Figure 2.7 compares various projections of the same object.



(a) Perspective Projection **(b)** Parallel (Orthographic) Projection

Figure 2.5: Different projections for a set of lego pieces r [2].

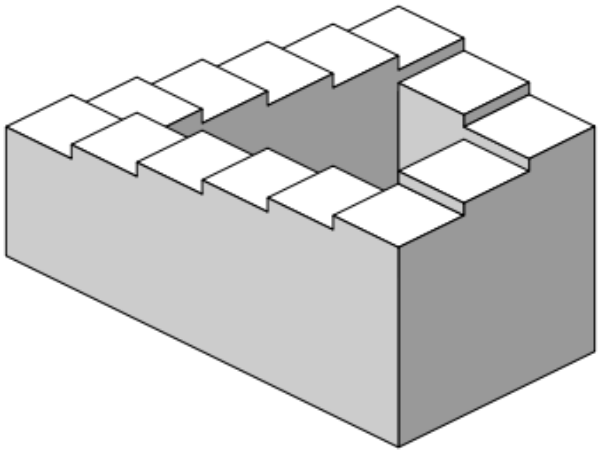


Figure 2.6: The impossible staircase, created by Lionel Penrose and his son Roger.

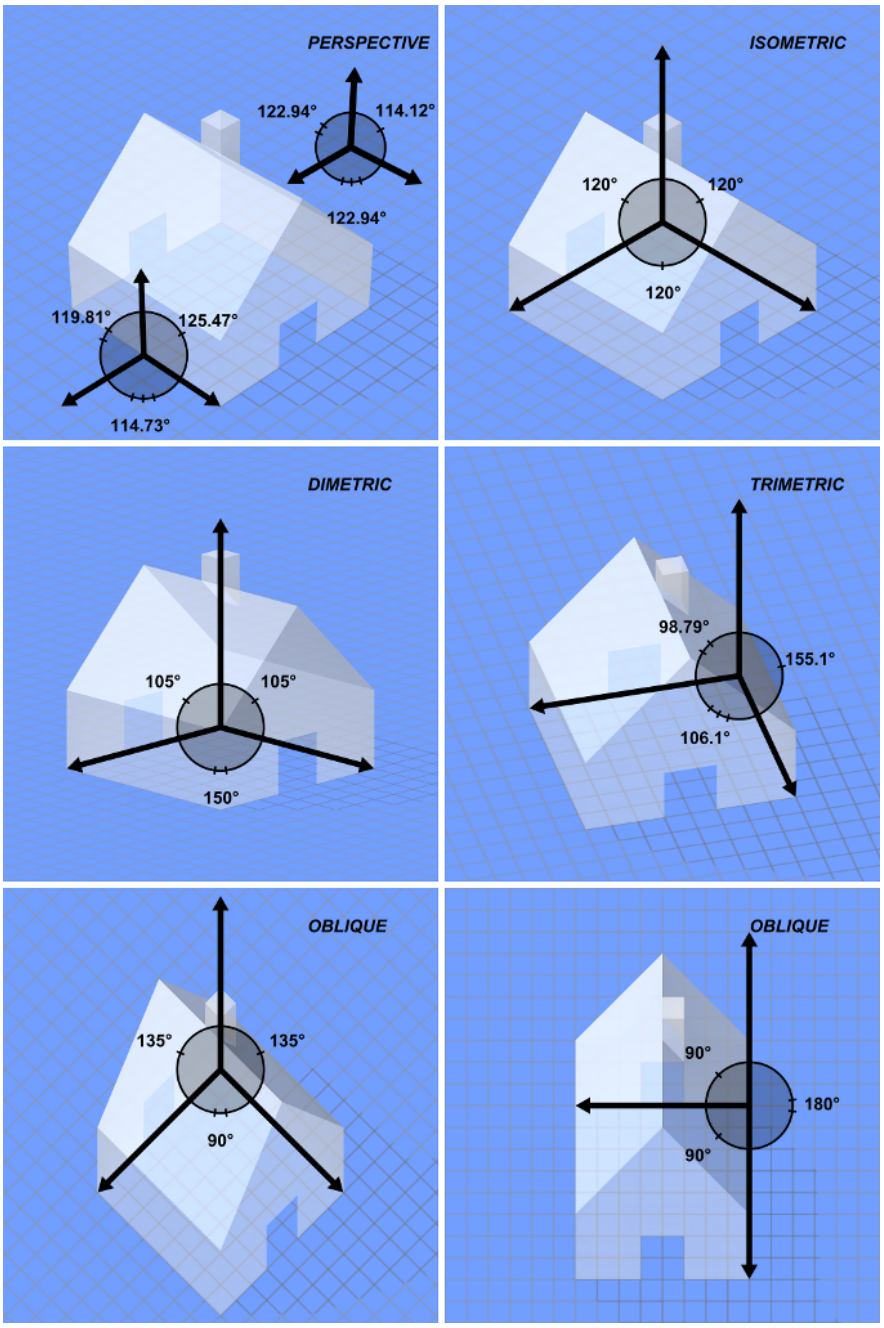


Figure 2.7: Comparison of different projection methods [3].

The pinhole Camera Model Based on the principles of the human eye the most fundamental imaging model is the pinhole camera model. The model is formulated as a perspective projection of all points of the surfaces of the captured objects onto the capturing medium. The capturing plane is positioned somewhere in front of the focal point, which constitutes the *center of projection*. The perpendicular distance of this point from the projection plane, is known as the *focal length* $\boldsymbol{f}$ and is the basic parameter that defines the properties of the camera. The line that passes through the center of projection and is perpendicular to the projection plane is the *projection axis*. This is a very good algebraic model of our eyes for various reasons:

- All the light (photons) which strikes our *retina* originates from any viewed object of the world and passes through our *iris* to do so. Our iris is actually the *center of projection* of our eye.
- Light travels on a straight line, forming *projection lines* though the iris, and onto the retina.
- The optical system comprised by the retina and the lens of the eye, is a *planar projector*.

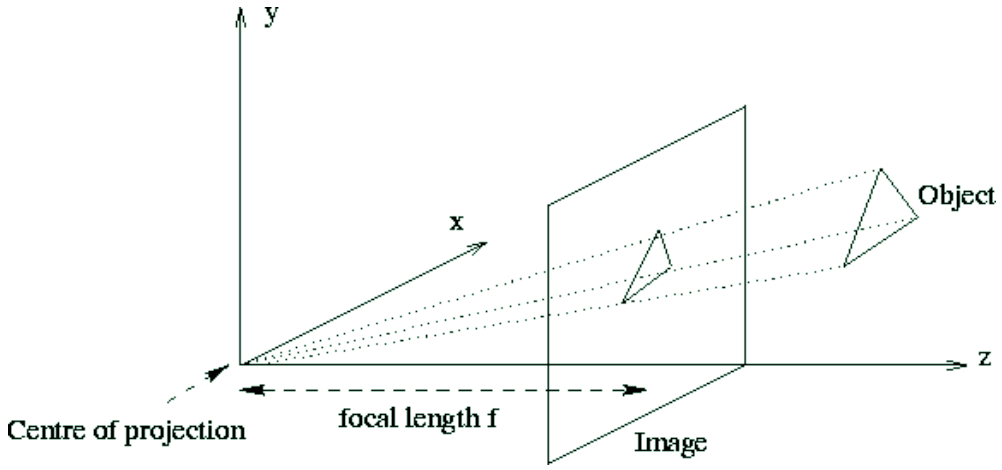


Figure 2.8: Basic Pinhole camera Model

Therefore, the images we see with the naked eye, are in fact imperfect *perspective projections* of the world. All home video and still-cameras are systems designed to mimic the eye. They are engineered as close approximations of the *pinhole model*.

Our eyes have a limited angular range: they can only capture photon rays that deviate from the projection axis only up to a certain angle. This angle is the *Field of*

View (FOV) and for both eyes combined is almost 180°. However, inherent distortions are apparent when we try to capture similar images on paper. Perspective projection does not preserve parallel lines in general and, because of the projection itself, any object placed at [or near] the edge of the *Field of View* appears "distorted", or at least our intuition guides us to this conclusion. Figure 2.9 shows a rendered perspective

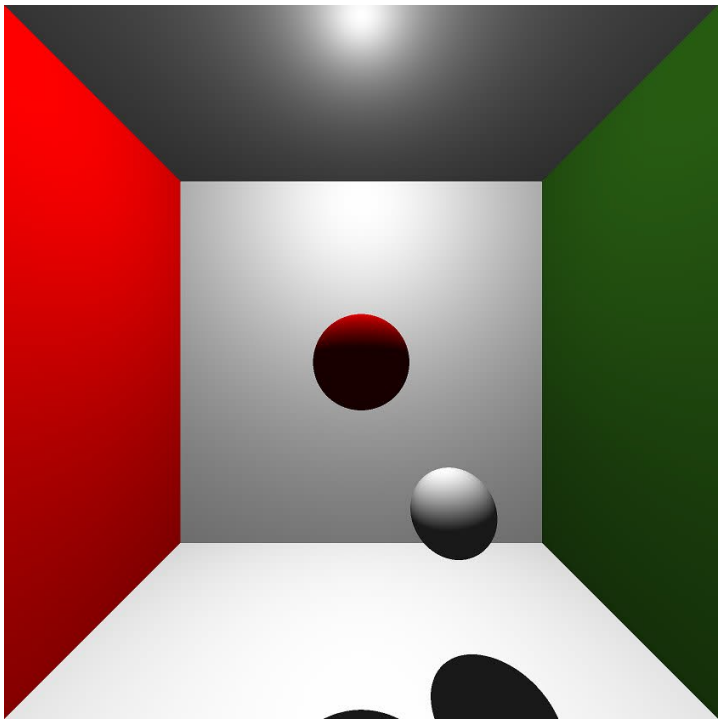


Figure 2.9: Rendered Perspective Projection with a large field of view.

projection of a scenery with spheres. Notice how the sphere that lies at the edge of the scene appears as ellipse and not as a circle, as our intuition dictates. In contrast, the sphere at the center of the scene is a perfect circle. This is a perfect perspective projection, not a distorted one: although the resulting image appears distorted to the eye, it is actually an effect of the projection itself.

Given the fact that every type of projection has an inherent loss of information, there is a growing trend to use a specific type of cameras for the purpose of extracting metric measurements from scenery. Omni-directional cameras provide a full 360 ° view of the surroundings [Figure 2.10(b)]. Although the image captured is "bizarre", it provides a superior view of the environment.

(a) Catadioptric Image

(b) Extracted Panorama Image

Figure 2.10: Omni-directional camera imaging [4].

2.5 Digital Image Representation

2.5.1 Image Sensors

Regardless of the geometric quality of an optical system, image sensors are predominantly planar devices. These can be considered as the *projection plane* of the optical system. Digital image sensors convert light (photons) that strikes the sensor to an electrical charge, and quantify it. Two aspects of the sensor must be examined to provide meaningful physical information from the data:

- The model that describes the conversion from photon flux (radiation power) to digital data is important to provide true lightness (brightness) information for

the captured scene.

- The spatial aspect of the quantification process is the correspondence of the quantified data to the sensor area (and therefore the projection plane of the system). This is an analytical study of the physical arrangement of the sensor itself.

Generic Model The two predominant imaging technologies, CMOS and CCD sensors, both can be simplified by an abstract model that provides a systemic way to extract physical information from the sensor itself. The sensor area is covered by a large number of *photosensitive units* that have the ability to capture photons (using the *photoelectric effect*).

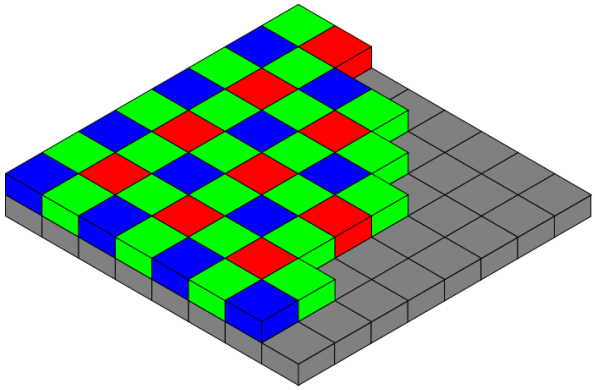
These units act as light integrators (over time), by collecting photons that strike their surface, providing an charge voltage, that depends on the time duration that the sensor is exposed to light and the intensity of the light itself. This is one important factor of the sensor typically referred to as **exposure time**. This time period, should optimally be as little as possible, to avoid the phenomenon of *motion blur*. As individual objects move around during the period which the sensor captures light, their *projection* on the captured image becomes distorted. Avoiding motion blur is not always possible: the motion of the environment must be minimal during the exposure time, which in turn must be large enough for the given light intensity of the environment to allow for proper integration on the sensor and produce values at its operational range .

Therefore, another important parameter arises: the *sensor sensitivity*, an intrinsic factor of the device, which expresses the required amount of light that must hit a photosensitive unit, for it to reach a certain amount of recorded output value.

All such units are typically arranged in a rectangular grid. To provide a simple image with no color information (i.e. gray image) the sensor is exposed to a desired scene, and the recorded data is stored as a 2-Dimensional binary array in memory. Each value represents an quantized intensity value, corresponding to the amount of light that was captured by a particular photosensitive unit. Each unit corresponds to a **pixel** (picture element).

The digitization process of each of the sensor's units is called the *readout process*. Each unit's accumulated charge (voltage) is amplified using an amplifier with a high and predefined *gain value*. Combined with a known *sensitivity* value, it is possible to reconstruct the actual power (*radiant flux*) of the depicted scene, for each pixel individually. This voltage is then *quantized* using an *analog-to-digital converter* (*ADC*). The *ADC* is typically one with a *logarithmic* response. This is closer to the sensitivity response of the human eye, and provides the ability to depict more diverse values in a

digital value of given bit-length since the quantizer has a larger *Dynamic Range* (the ratio between the largest and smallest possible values of a given pixel).



A typical digital sensor surface: a Bayes pattern. [17]

For colored images, it is necessary to provide more information (values) for each pixel to accommodate for the representation of color. Mimicking the human eye, where color is detected by having three different color-sensitive cells on the retina, color is typically represented by *tristimulus values*, or more generally triplets of values. These can be interpreted as a specific 'color' or 'hue' or 'nuance', based on the mathematical modeling of the values (**color model**) and their physical interpretation (**color space**) [18].

Therefore, each pixel of a color image requires information from more than one photosensitive unit. Typically three different types of adjacent photosensitive units with different characteristics are used. Each type is designed to have a different sensitivity response curve over the range of wavelengths that cover the *visible spectrum* of light. Adjacent units are combined to a single color pixel.

One important aspect of the correctness of color representation is the environmental lighting. The human eye has evolved to adopt to an extreme variety of environments, and is immune to different lighting conditions. Image sensors, however have an inherent inability to account for differences in chromaticity, between various lighting sources. This chromaticity correcting is called **white balancing**, and is achieved by correctly scaling the relative sensitivity (or the output image) of the three color channels. Intuitively one can consider a case where a "reddish" light produces "reddish" images, something that can be accounted for by reducing the sensitivity of the red channel of the image. Practically this is a more complex procedure which must be performed in conjunction with a correct exposure time.

2.5.2 Color Spaces and Color Models

Given an arbitrary imaging sensor, (raw) data captured by the device cannot be interpreted without any knowledge of the performance of the photosensitive units. The

output recorded by each of them, has to be interpreted by the *Spectral Sensitivity* of the unit. For example, humans ordinarily have a average standard sensitivity curve for each of the color-sensitive cells of the retina, as depicted in Figure 2.11.

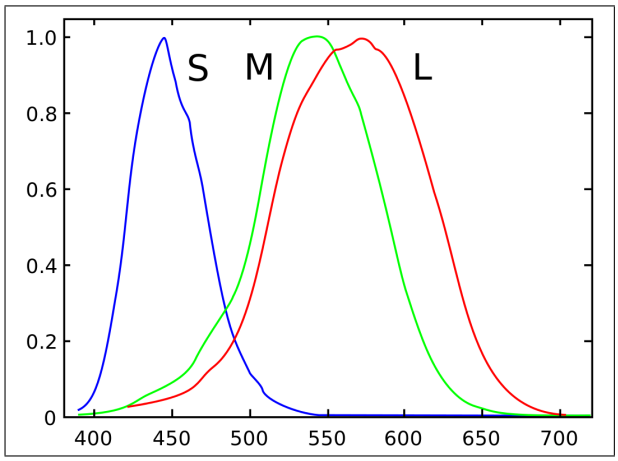


Figure 2.11: Spectral sensitivities (normalized responsivity spectra) of human cone cells, S, M, and L types. [5]

Even intuitively one can see that the human retina cannot detect an infinite number of *perceived colors*. In fact each object's apparent color is function of its intrinsic properties (*reflectance*) and the properties of the lighting source (its *spectral power distribution*).There are unlimited possibilities for the 'color' of an observed object. However, the *spectral power distributions* of all possible objects one can observe are reduced to the *tristimulus values*. This has a strange consequence, different spectral power distributions might correspond to the same stimuli to the retina. This phenomenon, is called **metamerism**. For example, a bright red ball under a bright blue light appears almost black, but so does a truly black one.

This has a profound effect on the *colorimetric* properties of an arbitrary color sensor. Ideally a 'perfect' sensor would provide similar information as the human eye: it should provide three photometric units with similar responses as the human eye. Such a sensor would approximate as much as possible the way an average human sees.

Unfortunately, there is no such thing as an ideal sensor. Each type of device has a different response to light, and therefore it produces different images when compared to another: their *color space* is different. One can map the spectral sensitivity of the sensor and therefore compare it to the ability of the human eye.

Typical tri-stimulus models use 1 byte per channel, yielding 3-bytes per pixel. High performance scientific sensors might use 10 or 12 bits per channel, providing much more colorfulness resolution.

RGB Color Model Based on the human vision, sensors produce RGB images, that reproduce color as a linear combination of three primary quantities, each sensitive approximately to the red, green, and blue ranges of wavelengths. Its an *additive color model* since black corresponds to (0,0,0) and white to (1,1,1) assuming normalized values between zero and one. This basic principle follows intuitively the human eye. If one defines explicitly the *spectral sensitivity* of the primary values, that particular RGB *color model* becomes a *color space*: for example, a popular RGB model, sRGB, is also a color space.

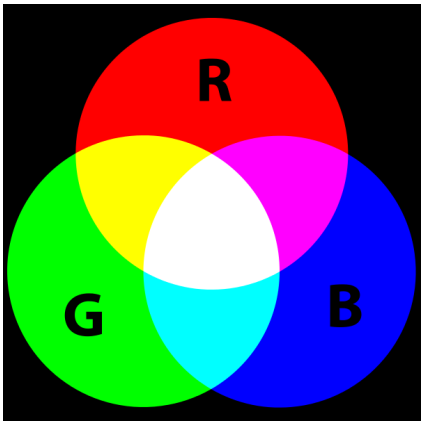


Figure 2.12: RGB additive model

RGB Derived Spaces For numerous reasons, the RGB model, is not an intuitively arranged model. Given an arbitrary RGB value (triplet) one cannot extract photometric qualities for that value. Features such 'luminance', 'saturation','intensity','hue','chromaticity' are often useful in some application and transformations to such quantities are modeled usually upon the rgb model. Such color models based on an rgb model, are equivalently *color spaces* when the initial RGB model is also a *color space* itself.

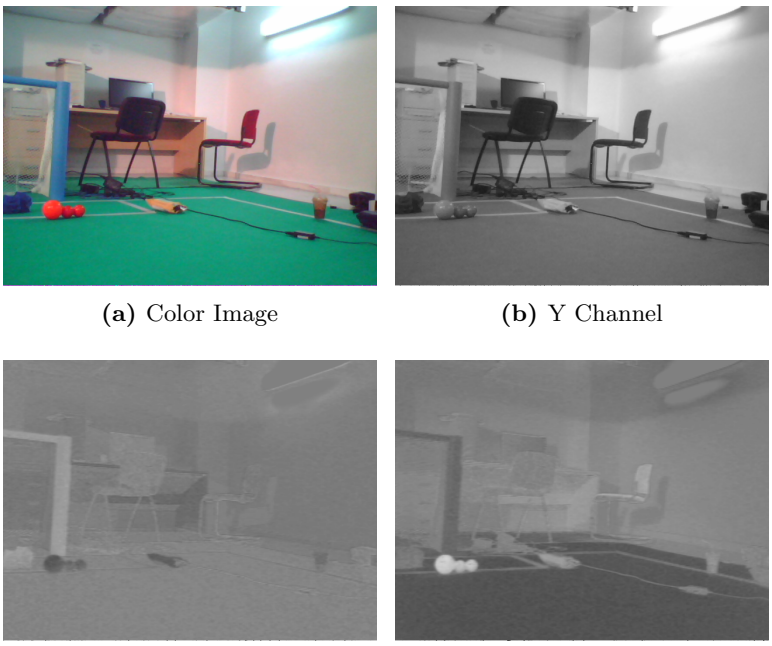
YCbCr model The Y-Cb-Cr model is a digital encoding, equivalent of the YUV model used in analog video. It is standardized as a *linear transformation* over RGB. Assuming 8-bit representation per value (24-bit color depth) in both RGB and YCbCr, the *JPEG standard* defines the conversions [19] as:

$$\begin{aligned} Y' &= &+ (0.299 \cdot R'_D) &+ (0.587 \cdot G'_D) + (0.114 \cdot B'_D) \\ C_B &= 128 &- (0.168736 \cdot R'_D) &- (0.331264 \cdot G'_D) + (0.5 \cdot B'_D) \\ C_R &= 128 &+ (0.5 \cdot R'_D) &- (0.418688 \cdot G'_D) - (0.081312 \cdot B'_D) \end{aligned}$$

The inverse transformation is similarly defined. Unfortunately there are a few other standardized variations that are very close to the aforementioned one, using slightly different constants. So unless one is provided with further information, it is very difficult to interpret a YCbCr image with optimal color accuracy.

YCbCr is roughly equivalent to RGB (in terms of representable colors), but offers readily a *relative luminance* value for each pixel:the 'Y value' is named '*luma*' and

represents *apparent brightness* as a value between black and white, in a fashion that is intuitive for a human: regardless of the color ('hue') the '*luma*' value can be used to estimate how bright a pixel is . One can simply create a grayscale version of an original image by using the '*luma*' value as a grayscale value.



(a) Color Image **(b)** Y Channel

(c) Cb Channel **(d)** Cr Channel

Figure 2.13: Image captured by the *Nao* robot, and its YCbCr decomposition

The remaining two values (Cb,Cr) are *chroma values*, and represent purely color information. They correspond roughly to Blue Chromaticity and Red-Chromaticity, hence the terms Cb,Cr respectively. YCbCr is usually used in video applications because it allows for a simple (lossy) compression method that is simple, and can be streamlined into hardware to reduce the required bandwidth:

Chroma Sub-sampling Based on the properties of human vision, one can reduce the chromatic information of an image with a lesser effect on its apparent quality. This is the result of the human eye being much more sensitive to changes in luminance differences than in chromatic differences. Therefore, in color models such as YCbCr where chromatic information is separated from relative luminosity ('luma') it is possible to directly reducing the spatial resolution of the Cb and Cr channels with great performance. Instead of providing 3 values for each pixel, chroma sub-sampled images

share the Cb and Cr values between adjacent pixels, with each pixel 'owning' a distinct Y value. Typically this is performed by dividing the spatial resolution of Cb,Cr by a factor two in the horizontal and/or vertical dimension.

A full resolution image in Y Cb Cr model is stored as consecutive triplets :

$$\begin{bmatrix} Y_{1,1} & Cb_{1,1} & Cr_{1,1} & \cdots & Y_{1,n} & Cb_{1,n} & Cr_{1,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ Y_{m,1} & Cb_{m,1} & Cr_{m,1} & \cdots & Y_{m,n} & Cb_{m,n} & Cr_{m,n} \end{bmatrix}$$

This results in $n \cdot m \cdot 3$ values for a nxm image, or equivalently 3 values per pixel.

One can simply reduce the in the horizontal dimension for Cb and Cr values, resulting in an average of 2 values per pixel. This is reduces the required amount of space (and bandwidth in the case of video) by one third. Each consecutive pair of pixels, can be reduced from : $[Y_1\ Cb_1\ Cr_1\ Y_2\ Cb_2\ Cr_2]$

2.6 Algebraic Representation of Euclidean Spaces

The tool we need to describe the euclidean rigid body world around us, is linear algebra. The *3-Dimensional euclidean (affine) space* and the *2-Dimensional Cartesian plane* that describe the (idealized) *rigid body* world can be sufficiently described using *Affine Transformation Matrices* and *Homogeneous Coordinates* [16].

2.6.1 Homogeneous Coordinates

The *3-Dimensional rigid body world* can be described using basic $\mathbb{R}^3$ *column vectors*, to represent all vertices (points) . To sufficiently accommodate perspective projections and *translations*, we extend all *3-D coordinates* with another *real* value, the *homogeneous component*. Every *3-D column vector* maps a *point-vertex* :

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^{\mathrm{T}} \qquad where\ \mathbf{x} \in \mathbb{R}^3$$

The notion of *3-D vertices* is extended to $\mathbb{R}^4$ using *homogeneous coordinates* with the last element being a *weighting value*. Let that value be w:

$$\mathbf{x}_h = \begin{bmatrix} wx_1 & wx_2 & wx_3 & w \end{bmatrix}^{\mathrm{T}}$$

is an infinite set of vectors that is for all our purposes **equivalent** to the 3-D column vector $\mathbf{x}$:

$$\mathbf{x} \sim \frac{\mathbf{x}_h}{w} = \begin{bmatrix} x_1 & x_2 & x_3 & 1 \end{bmatrix}^{\mathrm{T}} ,\ \forall\ w \neq 0\ ,\ w \in \mathbb{R}$$

For all such $\mathbf{x}_h$ iff $w = 1$ we can ignore the homogeneous term and obtain the initial vector $\mathbf{x}$. For $w = 0$ the transformation of $\mathbf{x}_h$ back to $\mathbf{x}$ is impossible. In all such cases $\mathbf{x}_h$ represents a point at **infinity** .

Similarly, the notion of *2-D coordinates*, for representing vertices-points on the *2-D Cartesian plane* can be extended with a homogeneous term, to accommodate transformations.

This notation enables the use of *4 x 4 transformation matrices* on these *4-D column vectors* to accommodate a variety of useful rigid body transformations for the 3-D euclidean space, and the use of *3 x 3 transformation matrices* on *3-D column vectors* to accommodate useful transformations for the *2-D cartesian plane*

2.6.2 Affine Transformations

Every useful *linear transformation* for the purposes of our application can be described by an *affine transformation matrix*. Such a matrix $\mathbf{M}$ used for the transformation for *n-Dimensional homogeneous points* has the following augmented form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \ , \mathbf{M} \in \mathbb{R}^{(n+1,n+1)} \tag{2.1}$$

where:

- $\mathbf{A}$ is an $\mathbb{R}^{(n,n)}$ matrix.
- $\mathbf{b}$ is an $\mathbb{R}^{(1,n)}$ column vector that corresponds to the *translational* part of the transformation.
- the last row is an augmentation of the form: $\begin{bmatrix} \overbrace{0 \ \dots \ 0}^{n} & 1 \end{bmatrix}$ which leaves the homogeneous part of the right operand unaltered.

The transformation $\mathbf{M}$ is *invertible* **if and only if** matrix $\mathbf{A}$ is *invertible*, and the inverted transformation becomes:

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix}$$

The transformation can be applied on a given column vector (expressing a point in homogeneous coordinates) $\mathbf{u_h}$ to produce a resulting vector $\mathbf{v_h}$:

$$\mathbf{v_h} = \mathbf{M} \cdot \mathbf{u_h}$$

This notation of *pre-multiplication* where a transformation matrix is placed on the *left* of the "target" column vector is preserved throughout this text. Multiple transformations can be chained together to produce a single matrix of the same augmented form:

$$\mathbf{M}_{1-n} = \overbrace{\mathbf{M}_n \cdots \mathbf{M}_2\mathbf{M}_1}^{n}$$

The resulting matrix is equivalent to applying all $\mathbf{M}_*$ transformations in a **right-to-left** order :

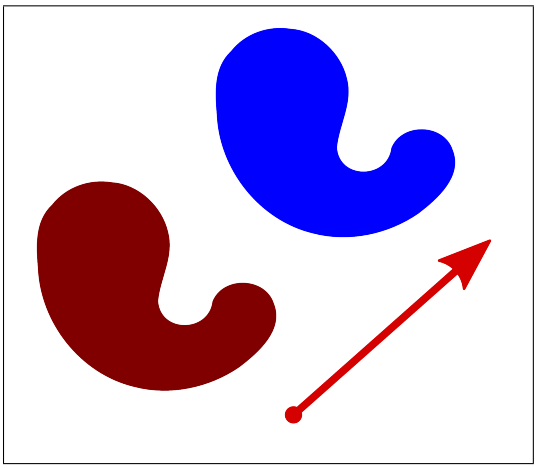
$$\mathbf{v_h} = \mathbf{M}_{1-n} \cdot \mathbf{u_h} \Leftrightarrow \mathbf{v_h} = \overbrace{\mathbf{M}_n \cdots \mathbf{M}_2\mathbf{M}_1}^{n} \cdot \mathbf{u_h}$$

For all transformations analyzed below, it is implied that the *coordinate system* is a Cartesian euclidean system with standard orientation (*right-handed*-it obeys the *right-hand rule*). Also, to avoid ambiguities, all transformations are **"Alibi Transformations"**. They represent transformations over a *fixed coordinate system* in which *vectors are transformed*. In contrast, the duality of an **"alias transformation"** where the vectors are represented in a new, transformed *coordinate system* is obtained by *inverting* the aforementioned transformation [20].

Translation A set of *N-D points* expressed in *homogeneous coordinates* can be translated in a fashion that preserves parallel lines, euclidean distances, and orientations. It is a means to describe **rigid linear motions**.

Every point $\mathbf{p}$ expressed in homogeneous coordinates as $\mathbf{p}_h$ can be *translated* using an *affine transformation matrix* $\mathbf{T}$. Each *translation matrix* that describes a translation, moves a point $\mathbf{p}$ by adding a (constant) vector $\mathbf{v}$ when applied to it:

$$\mathbf{p_h}' = \mathbf{T}(\mathbf{v}) \cdot \mathbf{p_h} \Leftrightarrow \mathbf{p}' = \mathbf{p} + \mathbf{v}$$



A translation moves every point by a constant displacement. [21]

One can form such a *translation matrix* as:

$$\mathbf{T}(\mathbf{v}) = \begin{bmatrix} \mathbf{I}^n & \mathbf{v} \\ \mathbf{0} & 1 \end{bmatrix}$$

with $\mathbf{I}^n$ being the n x n *identity matrix*. The transformation $\mathbf{T}(\mathbf{v})$, when applied to a column vector expressing homogeneous coordinates as $\mathbf{p_h} = \begin{bmatrix} wp_1 & , & \cdots & , & wp_n & , & w \end{bmatrix}^{\mathrm{T}}$, produces the point $\mathbf{p_h}'$:

$$\mathbf{p_h}' = \begin{bmatrix} wp_1 + wv_1 & , & \cdots & , & wp_n + wv_n & , & w \end{bmatrix}^{\mathrm{T}}$$

Since vector addition is *commutative*, multiplication of translation matrices is also *commutative*. The product of multiple *translation* matrices can be given by adding the *translation vectors* of each of the operands:

$$\mathbf{T}(\mathbf{v}) \cdot \mathbf{T}(\mathbf{u}) = \mathbf{T}(\mathbf{u}) \cdot \mathbf{T}(\mathbf{v}) = \mathbf{T}(\mathbf{v} + \mathbf{u}) = \begin{bmatrix} \mathbf{I}^n & \mathbf{v} + \mathbf{u} \\ \mathbf{0} & 1 \end{bmatrix}$$

The *inverse* of a translation matrix is produced by reversing the direction of the vector:

$$\mathbf{T}^{-1}(\mathbf{v}) = \mathbf{T}(-\mathbf{v})$$

For our needs we adopt a specific notation for Translation transformations along the coordinate axis x,y and z in the context of a *3-Dimensional Cartesian space*:

$$\begin{aligned} \mathbf{T}_x(a) = \mathbf{T}(\mathbf{v}) \quad , \quad \mathbf{v} &= \begin{bmatrix} a & 0 & 0 \end{bmatrix}^{\mathrm{T}} \\ \mathbf{T}_y(b) = \mathbf{T}(\mathbf{u}) \quad , \quad \mathbf{u} &= \begin{bmatrix} 0 & b & 0 \end{bmatrix}^{\mathrm{T}} \\ \mathbf{T}_z(c) = \mathbf{T}(\mathbf{w}) \quad , \quad \mathbf{w} &= \begin{bmatrix} 0 & 0 & c \end{bmatrix}^{\mathrm{T}} \end{aligned}$$

Rotation An arbitrary *rotation* of an *euclidean space* can be described by a *square transformation matrix*. All such matrices in the $\mathbb{R}^n$ domain can be composed as a product of n matrices that each describes a rotation around each of the *N-Axes* of the *Cartesian coordinate system*. Another representation for any rotation is the *Axis-Angle* representation: every rotation can be analyzed to an arbitrary *rotation axis*, and an *angle*, about that axis (*Euler's rotation theorem*). All rotations preserve parallel lines,vector angles, and euclidean distances. Rotation matrices are *orthogonal matrices* with *determinant* 1. Rotation matrices are *special orthogonal matrices*. If matrix $\mathbf{R}$ is a rotation matrix:

$$\mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1} \ , \ det(\mathbf{R}) = 1$$

For our convenience, we assume all *rotation matrices* are defines as *affine transformations*, implemented to be applied to *homogeneous coordinates*. The basic *2-Dimensional*

rotation matrix around the *axis origin* is :

$$\mathbf{R_2}(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ is the rotation parameters, in degrees . For the *3-Dimensional space* the analytical representations for rotations around the axis of the system are:

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Every matrix $\mathbf{R}$ that represents a rotation, always leaves the *zero vector* unaltered, and the vector always is its *null-space*.

$$\mathbf{R} \cdot \mathbf{x} = \mathbf{0} \Leftrightarrow \mathbf{x} = \mathbf{0}$$

The product of two rotations is always a rotation, since the result is a *special orthogonal matrix*. If $\mathbf{R}_p = \mathbf{R}_1 \cdot \mathbf{R}_2$:

$$\begin{aligned} \mathbf{R}_p^{\mathrm{T}} \cdot \mathbf{R}_p &= (\mathbf{R}_1 \cdot \mathbf{R}_2)^{\mathrm{T}} \cdot (\mathbf{R}_1 \cdot \mathbf{R}_2) \\ &= \mathbf{R}_2^{\mathrm{T}} \mathbf{R}_1^{\mathrm{T}} \mathbf{R}_1 \mathbf{R}_2 = \\ &= \mathbf{R}_2^{\mathrm{T}} \mathbf{I} \mathbf{R}_2 = \mathbf{I} \end{aligned}$$

$$det(\mathbf{R}_p) = det(\mathbf{R}_1 \cdot \mathbf{R}_2) = det\mathbf{R}_1 \cdot det\mathbf{R}_2 = +1$$

Transformation Decomposition Every transformation that is composed from an arbitrary set of rotations and translations, can be decomposed as a single rotation followed by a translation: $\mathbf{T} \cdot \mathbf{R}$. This can be accomplished iteratively by transforming

all reverse combinations (rotation after translation) into correct ones and combining groups of translations or groups of rotations into a single one.

Given a translation and a rotation matrix ,$\mathbf{T}$ and $\mathbf{R}$ respectively each is by definition a matrix of the form:

$$\mathbf{R} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix}$$

The product of these yields:

$$\mathbf{R} \cdot \mathbf{T} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{I} & \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{A} \cdot \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix}$$

This product can always be decomposed as a translation-after-rotation pair. Given:

$$\mathbf{T}_{AB} = \begin{bmatrix} \mathbf{I} & \mathbf{A} \cdot \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix}$$

It can be shown that:

$$\mathbf{T}_{AB} \cdot \mathbf{R} = \begin{bmatrix} \mathbf{I} & \mathbf{A} \cdot \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{A} \cdot \mathbf{B} \\ \mathbf{0} & 1 \end{bmatrix} = \mathbf{R} \cdot \mathbf{T}$$

2.6.3 Robot Kinematics

Forward Kinematics Robots typically have more than one manipulators with which interact with the environment.Each manipulator can be viewed as an independent rotary or prismatic joint with exactly one controllable parameter (Degree of Freedom-DOF). Not all possible combinations of joint positions are valid. Some constitute a collision of part of a robot with another, or with some item of the environment. If a combination of positions for all joints is valid, it is part of the robot's *joint space*. Each valid combination is uniquely identified by a point in this space. However the *joint space* of a robot has little or no information about the geometrical shape state of the robot: it provides no information about the *pose-state* of the robot in the *3-Dimensional* world. To obtain such information, robot (forward) *kinematics* are used.

For each joint independently, using its geometrical properties (sizes, lengths,angles) it is possible to construct a *transformation matrix* that describes points in one end of the joint in a coordinate system that is fixed to the other end, as a function of the joint's state. Such a matrix, can describe the location (and orientation) of items that are fixed onto the other end of joint. For ordinary rotary joints, as they are most

commonly found in modern robotic these *kinematic matrices* are constructed with the use of the **Denavit-Hartenberg parameters** convention [22].

The Denavit-Hartenberg convention describes the necessary 4*x*4 *affine transformation matrix* for each joint using only four parameters. These parameters are unique for a given joint, with typically one parameter being depended on the physical state (position) of the joint. The formal naming convention for D-H parameters are :d,θ,α and a. However, to avoid confusion between α and a, we adopt a different symbol to replace a, the symbol r. So the parameters become:

$$d,\theta,r \text{ and } a$$

Given an analytical modeling of a robotic joint, two reference *coordinate systems* are established at each end of the joint (fixated at each end), and these parameters are determined. Each reference coordinate system is called a *reference frame*. A transformation matrix between the *reference frames* is computed as a function of the physical state of the joint. For a rotary joint, the free variable is typically θ and the transformation is composed [22] as:

$$\mathbf{DH}(\theta) = \mathbf{T}_z(d) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{T}_x(r) \cdot \mathbf{R}_x(\alpha)$$

The composition of the four matrices has an analytical form:

$$\mathbf{DH}(\theta) = \left[\begin{array}{cccc} \cos\theta_n & -\sin\theta_n\cos\alpha_n & \sin\theta_n\sin\alpha_n & r_n\cos\theta_n \\ \sin\theta_n & \cos\theta_n\cos\alpha_n & -\cos\theta_n\sin\alpha_n & r_n\sin\theta_n \\ 0 & \sin\alpha_n & \cos\alpha_n & d_n \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

When joints are arranged sequentially, one attached at the end of the previous one, they form *kinematic chains*. The kinematic transformation of the end of the chain to the base *reference frame*, fixated at the start of the chain, can be obtained by arranging the individual *DH* matrices in a chain. For a chain of n joints, we can establish $n+1$ reference frames, and chain the corresponding transformations:

$$\mathbf{P}_0^n = \mathbf{DH}_0^1 \cdots \mathbf{DH}_{n-1}^n$$

Where the P_i^j matrix generally represents the transformation of points in the *jth reference frame* to the *ith reference frame* - a coordinate frame transformation.

Generally, the *forward kinematics* problem solution represents the unique and unambiguous transformation from the *Joint space* to the *Cartesian space*. The *inverse kinematics* problem constitutes the transformation from the *Cartesian space* to the

Joint space. This problem has no generic properties, it can have an analytical solution, or it can be approximated (typically with a *Jacobian approximation* method). This problem is application specific and there is no generic approach to suit all needs. A widely accepted solution has been provided by the *B-Human* team [23].

Chapter 3

Problem Statement

The first step to achieve an intelligent agent on the RoboCup field, is visual awareness of the surrounding space. The environment where the agent is operating is a highly volatile and challenging one. Other agents are constantly moving, and also the soccer ball is inherently hard to manipulate. Agents are required to operated with full autonomy, without any aid from humans. A fully autonomous robot must be be able to intelligently perceive its position and coordinate with teammates to challenge the opponent, and score.

3.1 The Robots

The Nao robots are equipped with fairly restricted electronics. The main processor is of the embedded family, and computations must be fairly limited. The main sensor that we utilize, the video cameras, are located on the Nao's forehead and chin. These are fairly average VGA cameras, providing images at approximately 30 times a second. Processing the images is a computationally intense process, that can potentially result in shortage of CPU power.

The vision system, must be able to process the video input with sufficient speed, to allow the agent to act and react promptly. Of course, it is not the only processing that takes place in real time, limiting even more the available processing time. To quantify this, given an operating rate of 30 fps the available processing time is **33.33 ms**. Taking into account that the system must leave ample headroom for other systems, we further limit this time to **20 ms**. This is an extraordinarily little time for video processing.

Also, to aid the vision system, the platform must complement the vision stream system with a robust and efficient joint sensor system. This system is needed to over-

come an important limitation of the Nao: images that captured from the camera, are delivered to the user with a fairly large amount of delay. Measurements indicate that this delay is around **140 ms**.

As with every vision based system, the following aspects of the camera sensor need further understanding (see also 2.5.1):

Motion blur: Every captured image frame of the camera sensor has the possibility of *blurred* areas on it, due to the motion of the robot, or its environment. This distorts the boundaries of objects, which appear stretched and distorted.

Exposure time and gain: To maximize the quality of the captured images, the control *exposure time* and *gain* settings of the sensor is critical to the system.

Color correctness: Due to the nature of the objects that must be recognized correctly, correct color perception is critical. *White Balance* settings must be adjusted accordingly to achieve this.

The goal of this system is to provide numerical information for the location of any object visible on the image. This requires substantial computational effort, and a great challenge is to minimize the amount of resources consumed during the processing of visual data. The available information is restricted to sensory input from the robot itself, but coordination is allowed, so robots can exchange data and synchronize their strategies using *wireless networking*.

3.2 Rules

The Nao robots, as one would expect, are split in two teams, red and blue . Each team is formed four robots (2011 rules). The robots of each team are marked using a cloth band around the waist, colored light red and navy blue respectively.

Year by year, the rules that the robots must obey become more challenging. They are expected to:

- Restrict their location on the field. Robots are punished for moving outside the field area.

- The robots must intelligently maneuver around each other, and avoid collisions. Robots are punished for running onto some other robot, either teammate or opponent.

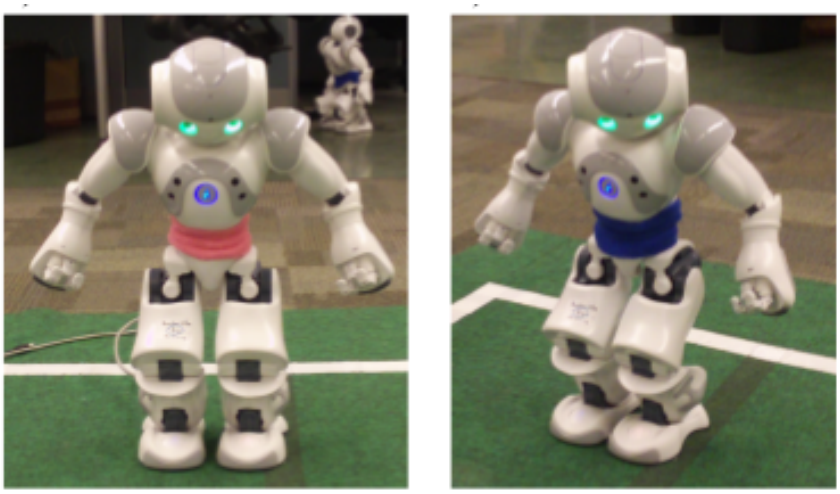


Figure 3.1: Nao robots: two agents from opponent teams.

- The result of the match is the result of the goals scored. False actions that result in a own goal are not treated specially, the robots must avoid such actions autonomously.

- Ball possession is restricted. Robots must avoid holding the ball for large amounts of time. Doing so results in penalties.

Robot that have violated some rule, are removed from the field for some time (30 seconds for SPL 2011 [24]), and repeated violations result in the permanent removal of the player.

3.3 The Field

The area where the Nao robots are called to operate is the SPL soccer field. It has a total area of 7.4 m by width 5.4 m (SPL2010 and SPL2011) [24]. The dimensions of the soccer field are shown in Figure 3.2. The field is build from a thin green carpet, to resemble a real soccer field.

Lines have a predefined pattern and a width of 50 mm. These are constructed typically from ordinary white adhesive tape.

Motion control on the field is a challenging problem, due to the difference in friction between the lines and the carpet, but also due to the fact that the soft carpet is not an ideal hard-immovable surface.

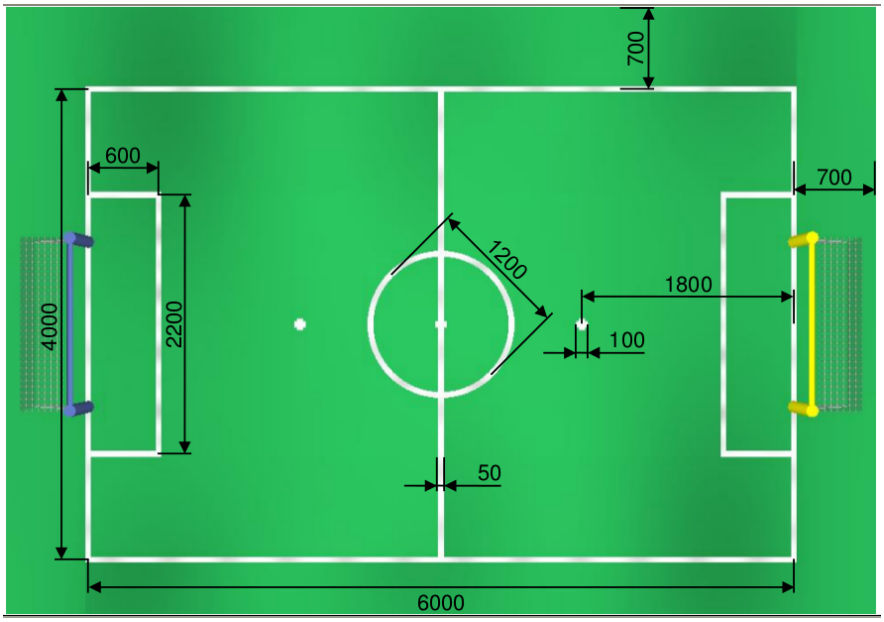


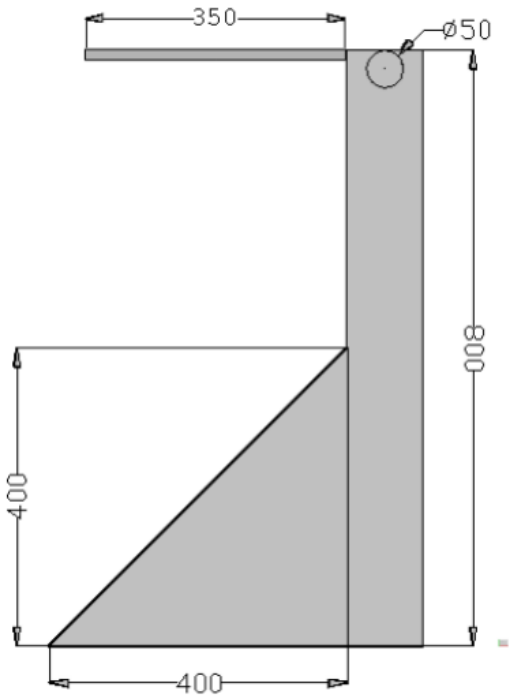
Figure 3.2: Soccer field: the SPL 2009-2010 league.

3.4 Target Objects

An autonomous robot on the field must be able to correctly quantify its position and the position of the ball. For the self-localization process the most prominent *landmarks* that can provide information are the goalposts. The ball is also a vital object for correct play. All these objects have predefined physical characteristics, one of them being the color. This is a key element for all SPL vision systems.

Goalposts Goalposts are fixed to a predefined position and have predetermined color and size. The red team occupies the yellow goalpost, and the blue team occupies the sky-blue goalpost. They are 800 mm in height, with a width of 1400 mm.

Ball The official SPL ball is a Mylec orange street hockey ball. They are 65 mm in diameter and weigh 55 grams. Their surface is smooth and glossy. Detecting a ball placed near a corner from the opposite corner of the field (up to 7m away), and correctly estimating its distance is a challenging task.



(a) Side view

(b) Top view

Figure 3.3: Goalposts: the primary landmark of the SPL soccer field.

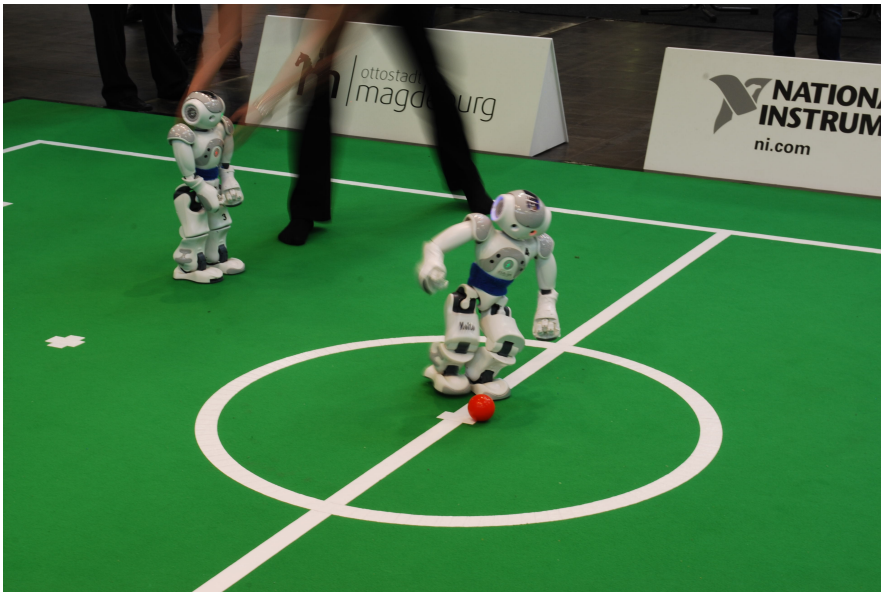


Figure 3.4: Kouretes kicking the ball at the **GermanOpen 2011** event.

Figure 3.5: Real SPL game at the **RoboCup 2011** event.

3.5 Design Goals

The system that was integrated into the Monas platform for the Kouretes team, was intended to provide **distance and bearing** measurements of object in the robot's vicinity, in real time. The following concepts are key elements of the system that was developed:

Robustness The system leans towards eliminating false-positives. Past experience has shown that disastrous results can occur if the vision system incorrectly identifies objects in erroneous locations. Due to the large amount of video data and their prompt availability, the impact of falsely rejecting a possible object is far less damaging than falsely accepting a possible match.

Further along this concept, the system must be able to cope multiple environments. The main concern for this, is the effective perception of colors in a wide variety of lighting conditions without the need of adjustments.

Speed Given the above, the vision system must be able to operate near the operating cycle of the input data: 30 times a second. This allows the robustness guideline to take effect, and provide both accurate and prompt results.

Extensibility The rules of the SPL field change over year, and sometimes drastically. It is imperative that the vision system be flexible, configurable and agile. Most importantly, other visual queues that are currently ignored, such as lines or other robots, must be easily added to the system.

Self-Validation During the execution, the system must be able to estimate its performance and feed this information to the system. Although the use of the information is outside the scope of the system, it is a good aspect for the future members of the team.

3.6 Related Work

To further understand the level of the competition, the work of other teams with notable performance in the games must be briefly explained. Specifically we shall outline the individual solutions of the **B-Human** [23], **Nao Devils** [25], and **rUNSWift** [26] teams. Their work was the mostly influential to our system.

3.6.1 B-Human Perception System

B-Human use a scan line-based perception system. The YUV422 images provided by the Nao camera have a resolution of 640 × 480 pixels. They are interpreted as YUV444 images with a resolution of 320 × 240 pixels by ignoring every second row and columon of the input image. The images are scanned on vertical scan lines. Thereby, the actual amount of scanned pixels is much smaller than the image size. The perception system is comprised of various modules provide representations for the different features:

- Ball detection.
- Goal perception.
- Line, line intersections, and center-circle detection.

All information provided by the perception modules is relative to the robot's position.

Coordinate Systems Various coordinate systems are used for the detection process:

- The global coordinate system is described by its origin lying at the center of the field, the x-axis pointing toward the opponent goal, the y-axis pointing to the left, and the z-axis pointing upward.
- In the egocentric system of coordinates the axes are defined as followed: the x-axis points forward, the y-axis points to the left, and the z-axis points upward.

Camera Transformation A representation containing the transformation matrix of the camera b(B-Human uses only use the lower camera) is computed based on the orientation and position of a specific point within the robot's torso relative to the ground. In addition to this some robot-specific parameters are integrated that are necessary because the camera cannot be mounted perfectly plain and the torso is not always perfectly vertical. A small variation in the camera's orientation can lead to significant errors when projecting farther objects onto the field.

For the purpose of calibrating the mentioned robot-specific parameters there is a debug drawing that projects the field lines into the camera image.

Based on the camera transformation matrix, another coordinate system is provided which applies to the camera image. This representation provides a mechanism to compensate for different recording times of images and joint angles as well as for image distortion caused by the rolling shutter. It provides a mechanism to compensate for

different recording times of images and joint angles as well as for image distortion caused by the rolling shutter. It uses interpolates the robot orientation when recording each image row using the rate of change of the egocentric system of coordinates.

If the robot sees parts of its body, it might confuse white areas with field lines. However, by using forward kinematics, the robot can actually know where its body is visible in the camera image using the coordinate systems provided, and projecting the robot contour to the camera image and use it as a boundary.

Image processing The image processing uses a color segmentation scheme to guide a region building and classification system. First the field borders are detected. This is done by running scan lines, starting from the horizon,downwards until a green segment of a minimum length is found. From these points the upper half of the convex hull is used as field border. Since the goals are the only features that are above the horizon and the field border, the goal detection is based on special segments and is not integrated in the region classification.

Scan lines running from the field border to the bottom of the image,are used to create segments of continuous color. These segments are used to build regions of uniform color. The result of this process is line and center-circle estimations from the field. This is a robust process, that uses the projection of the lines on the field to exclude erroneous regions.

Goalpost Detection Since the foot of a post must be below the horizon and the head of a post must be above (or not visible) the post must cross the projection of the horizon to the image. Therefor it is sufficient to scan the projection of the horizon in the image for blue or yellow segments to detect points of interest for further processing. If the horizon is above the upper image border, a line a few pixels below the upper image border will be scanned.

From these points of interest the process tries to find the foot and head of a potential post. This is done by running up and down from a point of interest and continuing the run a few pixels left or right if a border was found.

For each of these possible matches further runs orthogonal to the vector connecting the head and foot are done to examine the width of the potential post. f the difference between this examined width and the expected width for a post at the position of the foot exceeds a certain threshold, the match is discarded. Also, all possible matches that do not have any green pixels below them are discarded.

All remaining items are transformed to field coordinates. For each match the lower

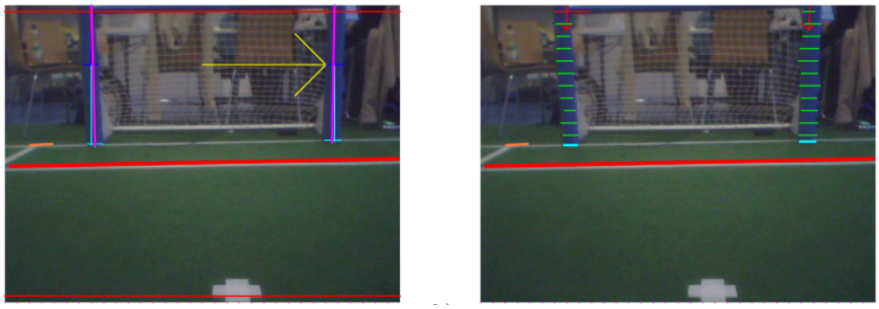


Figure 3.6: B-Human: Goalpost detection.

edge is transformed to field coordinates, assuming that it is lies on the field plane. Similarly the top edge is projected. If the two measurements differ above a certain threshold, the item is rejected.

If the remaining items are more than the maximum of two, or if items of different goalposts (yellow and blue) remain, all matches are rejected. Otherwise, the remaining items make up the detected goals.

Ball detection This process utilizes a list containing the center of mass of each orange region which could be considered as a ball, as it is produced by the region building process. For each item, a circle is extracted, by scanning radially from the center of mass. The edges must by a similar color to the center. If the ball lies outside the field dimensions it is rejected.

3.6.2 Nao Devils' Scanning scheme

Their key idea is to use as much a-priori information as possible to reduce both the scanning effort and the subsequent calculations needed. To process only a small fraction of all pixels the image is scanned along scan lines of varying length depending on the horizon's projection in the image. These scan lines are projections of radial lines originating from the point on the field below the camera center. Keeping the angular difference constant allows the implicit usage of this information during the scanning process, optimizes transformations between image and field coordinate systems and significantly simplifies the inter-scan-line clustering and reasoning about detected objects. In case of ambiguity additional verification scans are carried out. A sparse second set of horizontal scan lines is introduced to detect far goals.

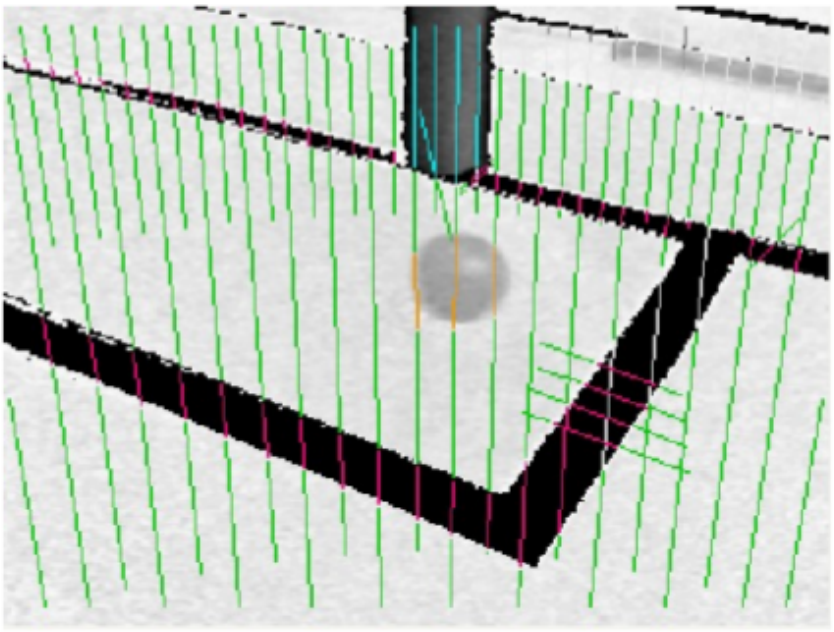


Figure 3.7: Nao Devils: Radial scan lines.

3.6.3 rUNSWift Vision system

The team's approach to object identification relies heavily on both color classification and edge detection. Color classification allows fast identification of approximate areas of objects such as goals and balls, while edge detection allows the positions of these objects to be found very accurately in the image, and provides a substantial amount of robustness to uneven and changing lighting conditions.

In order to make the object identification run as efficiently as possible, the processing starts on a frame by sub-sampling it to produce a 160 by 120 pixel color classified image. From this **saliency scan**, probable locations of the various objects we need to identify are determined. The full resolution image can then be used to accurately determine the location of each of these objects in the frame. During the creation of this subsampled image, a histogram is created in both axes that aids the detection of goalsposts.

A set of coordinates system is established, allowing to detect the horizon on the image, and use the projection of various body parts (mainly the torso) to exclude regions of the image from the processing. These coordinates systems use forward kinematics of the robot.

However, the Nao robot has physical inaccuracies in its build so correct kinematic transforms to determine the distance and heading to objects perceived in ac camera

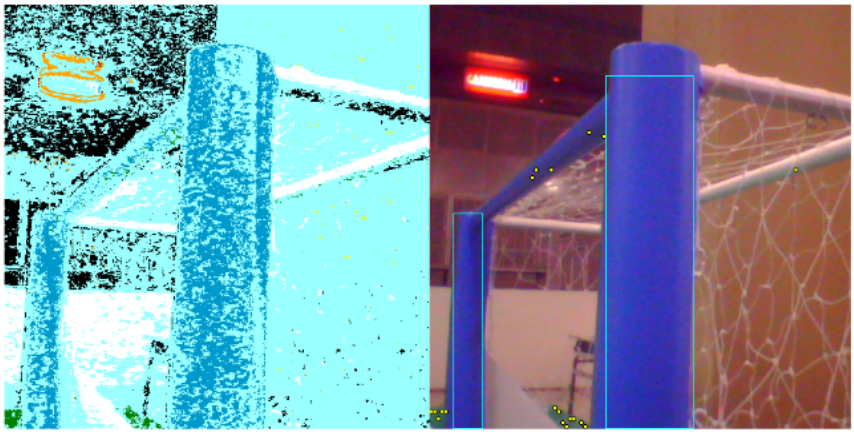


Figure 3.8: rUNSWift: Edge based goalpost detection.

frame produce a systematic error. This is thought to be primarily caused through small offsets in the angle of the camera mount. To account for this, an offline calibration tool is used

Goalpost Detection Starting from the histogram data of the saliency scan, the process scans around the expected locations of the goal posts. Region edges are detected using an edge detection process, that operates on the input image. This process avoids the color segmentation process all together, A vital advantage over other techniques.

Ball Detection The ball detection uses the regions identified in the region builder as being possible balls to locate the mostly likely position and size of the ball in the image, and then uses edge detection to identify the outline of the ball as accurately as possible. Firstly, the regions that have been classified as possible balls are scanned and the one which some constrains and contains the largest number of pixels is chosen to be the region that the ball detection will examine.

Similarly to the goalpost process, the edges of the correctly colored region are detected using a gradient detector. The edges are used to fit a ball from three points, in iterative method that selects such points from all the possible edges.

Chapter 4

Our Approach

The Kouretes team SPL code base, needed a basic data flow system to complement the framework of the team[12]. The system at hand, KVision , is based on the *message passing* protocol of the framework, to utilize sensor information from the robot hardware, and process the images provided by the CMOS camera sensor in sync. The goal is to extract and provide *visual object detection* coupled with *distance and bearing* measurements. These measurements are then processed with the purpose of *self-localization* and object localization on the soccer field.

The first difficulty arises from the need to establish a stream of information from the hardware sensors. Different sensors may have asynchronous operation to each other, and this must be rectified. Having established a stream of sensor data, these must be coupled with a video stream to extract information.

Subsequently, the acquired video frames are processed to extract regions of interest. The heart of the process relies on an efficient *color segmentation* process that is used to selectively search the image for useful colors. Regions are then tested for conformity to the restrains of some known useful object. Valid objects are stored in a message and forwarded to the rest of the platform.

4.1 Sensor Modeling

The *Nao Robot* provides sensory information every 10 ms. This information is required for the solution of *Forward Kinematics* that are used by the vision system. Sensor related code executes in real-time priority on the robot, providing new values for sensors approximately 100 times per second. Sensor value acquisition is a delicate procedure, because the rate of information is large. The flow of this information (mainly) to the

Vision activity [Section 2.3] must be controlled efficiently. Every run of the code, fetches a series of predefined values using the provided API, and feeds a timestamped message containing this information to the message processing subsystem. It is then utilized by other activities. Available information includes the following values:

- Position (angle) of all rotary joints of the robot.
- A 3-axis linear accelerometer along with a 2-axis gyroscope attached to the robot's torso.

All values are processed and transformed into a a message that is propagated through the *Monas* system. Real-time filtering is applied as needed to various values.

At the same time, the hardware provides a video stream through a CMOS camera. The rate of information is very large, up to 30 captured images of a 640 x 480 resolution every second. The camera provides various hardware settings that must be adjusted correctly, so that the sensor captures correctly exposed images.

4.1.1 Camera Calibration

The quality of images that the camera produces is of substantial importance. Without proper images, the vision system is inhibited from operating. Since an independence from camera settings is desired, an integral component of the system is to correctly set the desired values for camera *exposure time*, *gain*, and *white balance* [2.5.1]. While all these settings can be automatically configured by the camera it is undesirable to do so uncontrollably, because a-prior knowledge of the environment of the soccer field can provide much improved results

Our policy for setting the exposure time and gain is first to allow these values to be defined automatically by the camera, and then to restrict these so that the resulting images does not present any *blurriness* caused by the robot‘s motion. Specifically, the exposure time is limited to approximately **10-15ms**. If the optimal exposure, as selected by the camera, is higher than this, the setting is forced to this amount. This has an important effect on the color segmentation process which must take into account for a - different than optimal - exposure.

Assuming that the optimal exposure has settled to value O, and that the desired exposure time is chosen to be A then the sensor will capture images that have an altered radiant flux: it will be scaled by a factor of A/O compared to ideal. This factor has a non-linear effect on the digital values of the pixels, because the digitization process of the sensor features a *logarithmic response*.

This factor, once computed is used by color segmentation. While over or under-exposed images will have a different appearance (lighter or darker respectively), the segmentation is robust against this effect. Regions that reach the limits of the sensor's *dynamic range* are depicted as black or white and contain corrupted data, but such regions on the soccer field are practically absent and are not a cause for trouble.

After the exposure of the camera has been successfully tuned, the camera is pointed to a gray surface (a part of the robot's white arms) and left there for a few seconds to select a correct *white balance* setting.

This process is elemental to the robustness of the whole system, and it is performed at the start of the soccer game, or at specified moments. Having accomplished the calibration, images of the same objects captured at different environments retain their optical features. Therefore, this procedure greatly enhances the usability of a single color segmentation configuration.

4.1.2 Gyroscope Filtering

Gyroscopic sensors are devices designed to estimate angular velocity. The Nao has a 2-axis device fixed to its torso. The device can (along with the accelerometer device) aid in the estimation of the orientation of the torso.

Gyroscopes are particularly sensitive to temperature, and exhibit a phenomenon called *gyroscope drift*. Ideally, in a stationary robot, sensor output is zero. However, due to the the temperature sensitivity of the gyroscopes, the *zero-rate* output varies with time: the idle output of the sensor is not a constant value, especially during the initial warm-up time of the sensor. Our solution to this problem is to estimate this zero-rate value in each iteration, and account for it. Assuming:

Sensor linearity: The output value of the device is a linear function of its angular velocity, and this function does not depend on the zero-rate output value of the sensor.

Robot mobility: Since the Nao robot has limited mobility, the average output value of the sensor is zero. This is a good estimate for the zero-rate value. Generally speaking, the robot's rotational movement is fairly limited over time.

We can estimate the instant angular velocity along each of the axes of the device as:

$$\tilde{a}(t) = k(t) \cdot (a(t) + w(t)) + z(t)$$

Where:

$\tilde{a}(t)$ is the digitized sensor value at time t.

$a(t)$ is the real angular rate at time t.

$w(t)$ is the sensor noise at time t. It can be modeled as *zero-mean normally distributed*.

$z(t)$ is the zero-rate output of the sensor at time t

The goal of this analysis is to obtain a good estimation of $k(t)$ and $z(t)$.

Zero Rate Estimation On each iteration we can produce an *estimate* of $z(t)$ as:

$$\tilde{z}(t) = \frac{\int_{t-N}^{t} \tilde{a}(t)dt}{N} = \frac{\int_{t-N}^{t} a(t) + w(t) + z(t)dt}{N} \tag{4.1}$$

Given the assumption about the mobility of the robot, one can deduce that the integration of $a(t)$ over a fair amount of time will produce a result that approaches zero. Also, given the statistical properties of sensor noise $w(t)$, the integration of noise over an (infinitely) large amount of time yields zero:

$$\lim_{N\to\infty} \int_{t-N}^{t} a(t) + w(t)dt = \lim_{N\to\infty} \int_{t-N}^{t} a(t)dt + \lim_{N\to\infty} \int_{t-N}^{t} w(t)dt = 0 \tag{4.2}$$

Therefore, equation 4.1 can be simplified as:

$$\tilde{z}(t) = \frac{\int_{t-N}^{t} z(t)dt}{N}$$

Ideally as $N \to 0$, $\tilde{z}(t)$ becomes a better estimate of $z(t)$. However as N becomes smaller, equation 4.2 yields a larger error. In summary, N must be selected as a good compromise:

- It must be large enough so equation 4.2 is a good approximation.
- It must be small enough so $\tilde{z}(t)$ is a good estimate of $z(t)$.

This optimally requires to store all corresponding to the time window $t - N$ to t. An approximate solution is to use an *exponentially weighted moving average* with a suitable weighting factor:

$$\tilde{z}(t) = \gamma \cdot \tilde{a}(t) + (1 - \gamma) \cdot \tilde{z}(t - 1)$$

Due to the transient nature of the angular motion of the robot, an averaging window N in the order of 10-20 seconds is a good value for this application.

Scaling factor Estimation One more important aspect that must be addressed is the scaling factor $k(t)$, which is needed to extract the true angular rate from the digitized value. The scaling factor can be analyzed in two terms.

$$k(t) = p \cdot d(t)$$

First the true angular velocity is converted by the sensor into an analog value, according to sensor specifications. Generally the analog output of the sensor is directly proportional to the measured quantity. Therefore, the sensor introduces a predetermined scaling, p, which can be obtained from the sensor specifications. The analog output of the sensor is consequently digitized, by an *analog-to-digital converter (ADC)*. The converter, produces digital values that are scaled by in factor, intrinsic to device. The digital conversion factor $d(t)$, however has strong correlation to variations in the power supply voltage of the converter, and varies over time. Estimation of $d(t)$ is simplified by the sensor itself. Along with the 2-axis data, a fixed reference value is provided. This value, once digitized, can be used to estimate the state of the ADC converter:

$$d(t) = \frac{r}{a_{ref}(t)}$$

where r is the expected reference value produced by the sensor, and $a_{ref}(t)$ is the digitized value as observed by the system.

Given these, the logical decoding of gyroscope information is feasible in an efficient and effective way.

4.1.3 Accelerometer Filtering

Accelerometers are designed to capture linear acceleration. The Nao has a 3-axis device fixed its torso, alongside the gyroscope device. Accelerometers can be used to estimate linear motion, and aid in the estimation of the orientation of the torso's robot relative to the ground.

Accelerometers have an inherently varying sensitivity, that is the result of the manufacturing process precision. Also, because their principle of operation uses mass inertia, they tend to have a limited *dynamic response*. The devices on the Nao have the following characteristics:

- Output sensitivity is different on each device and must be accounted for.
- Zero-level offset is minimal, we consider the device to produce a near perfect zero output when input is zero.

- Temperature effects on the device are trivial and can be neglected.

To sum up, it is critical to estimate the sensitivity of the sensor on each robot individually, in order to utilize it efficiently. This is accomplished with these assumptions:

Sensor sensitivity: The output sensitivity of the sensor is fairly constant over time. Given the fact that the 3 axes are manufactured on the same integrated circuit, their performance must be similar.

Robot mobility: On average, since the Nao robot has limited mobility, the average acceleration sensed by the device is the force of gravity. Generally speaking, the robot's movement is fairly limited over time.

A good approximation of the sensor output (per axis) is:

$$\tilde{l}(t) = s(t) \cdot (l(t) + w(t))$$

where:

$\tilde{l}(t)$ is the output value at time t.

$s(t)$ is the sensor sensitivity, relatively constant over time.

$l(t)$ is the real linear acceleration at time t.

$w(t)$ is the sensor noise at time t. It can be modeled as independent *zero-mean normally distributed* noise with variance σ^2.

On each iteration we can compute the linear acceleration norm for the 3 axes:

$$n^2(t) = \tilde{l_x}^2(t) + \tilde{l_y}^2(t) + \tilde{l_z}^2(t)$$

The average value of $n^2(t)$ is:

$$\begin{aligned} E[n^2(t)] = E[\tilde{l_x}^2(t) + \tilde{l_y}^2(t) + \tilde{l_z}^2(t)] = \\ E[s^2(t) \cdot (l_x^2(t) + w_x^2(t) + 2l_x(t)w_x(t)) + \\ s^2(t) \cdot (l_y^2(t) + w_y^2(t) + 2l_y(t)w_y(t)) + \\ s^2(t) \cdot (l_z^2(t) + w_z^2(t) + 2l_z(t)w_z(t))] \\ = s^2(t)(E[l_x^2(t) + l_y^2(t) + l_z^2(t)] + 3\sigma^2) \end{aligned}$$

With the given assumptions, the average squared norm of the sensed acceleration is the force of gravity g^2 (approx. $(9.81m/s^2)^2$). A good estimate of noise variance can be

selected according to specifications or it can be derived from measurements. On each iteration we can produce an estimate of the sensitivity $s(t)$ of the sensor by computing an estimation of the average squared norm of the acceleration vector:

$$\tilde{n}(t) = \frac{\int_0^t \tilde{l_x}^2(t) + \tilde{l_y}^2(t) + \tilde{l_z}^2(t)dt}{t}$$

On each iteration the sensitivity of the sensor can be estimated as:

$$\tilde{s}(t) = \frac{\tilde{n}(t)}{g^2 + 3 \cdot \sigma^2}$$

Using this estimation, reliable acceleration values can be generated and utilized.

4.2 Kinematic Chains

Precise information for the position and orientation of the camera with regard to the ground plane is critical to the operation of the vision system. To accomplish this, forward kinematics [2.6.3] are utilized along with a precise estimation of the torso orientation in relation to the ground.

4.2.1 Ground Plane Orientation

While the forward kinematics solution provides the transformation from *joint angles* the Cartesian space, the *reference frames* must be expressed in relation to the ground. For this purpose, a pair of *Kalman Filters* [27] are employed to filter the inertial sensors (gyroscopes and accelerometers).

The goal here is to estimate the *pitch* and *roll* angles (and angular velocities) of the Nao‘s torso relative to the ground using the inertial data. These two variables θ and ϕ respectively, can then be used to construct a virtual reference frame, which follows a specific convention: the frame is arbitrarily placed on the ground plane, with the x and y axes always orientated on the plane according to the robot (i.e. they are ”attached” to the robot), the x-axis pointing forward and y axis pointing to the left of the robot. This leaves the z axis perpendicular to the ground, pointing upwards.

The orientation (but not the translation) of the ground plane can be estimated in relation to the Nao’s torso frame [Fig 4.1]. using the inertial devices. Gyroscopes provide good velocity estimates for the relative angular velocity between the two frames, and the linear accelerometers can be used to provide an angle estimation for pitch and roll.

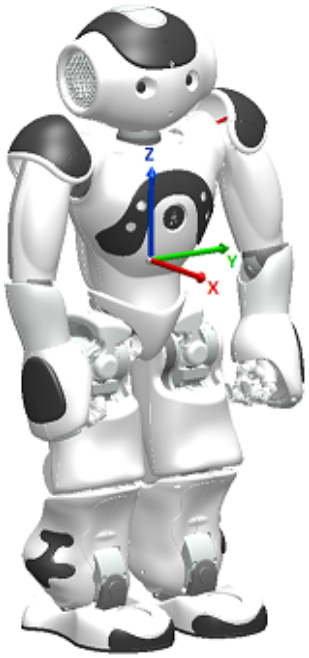


Figure 4.1: Nao's Torso Frame

The 3-axis accelerometer provides a composite vector of the linear acceleration of the torso frame. It is possible to decompose the vector into pitch and roll angles under the following assumptions:

- The ground is perfectly flat. The soccer field is perfectly horizontal and therefore gravity is a force perpendicular to this plane.
- The decomposition is reliable when the robot does not accelerate. Otherwise, additional forces act on the torso, and the accelerometer produces the composition of these forces.

A still robot, in an unknown orientation reads three (filtered) linear acceleration values l_x, l_y, and l_z from the sensors. These values compose into a vector with a norm equal to the force of gravity g :

$$g^2 = l_x^2 + l_y^2 + l_z^2$$

Any deviation from g can be used as a quantification that the robot is accelerating. This is used to produce a deviation amount which is fed into the Kalman filter.

Pitch and roll angles can be produced as:

$$\theta = atan2(l_x, -l_z)$$
$$\phi = atan2(-l_y, -l_z)$$

Intuitively, one can observe that if the torso is perfectly aligned with the ground, the acceleration values will be: $l_x = 0$, $l_y = 0$, and $l_z = -g$. If the robot is in free fall, all values are approximately zero. Given these four quantities, an angle and an angular velocity per axis, we can construct 2 *linear state spaces*:

$$\begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix}, \begin{bmatrix} \phi \\ \dot{\phi} \end{bmatrix}$$

Each variable space is filtered using a *1-Dimensional Kalman Filter*. The filtered data can be used to compose [2.6.2] a rotated torso reference frame as: $\mathbf{T} = \mathbf{R}_y(\theta) \cdot \mathbf{R}_x(\phi)$. This frame has the same orientation (rotation) as the ground plane, but it its displaced (translated) onto the robot's torso. It it trivial to compute a reference frame that lies onto the ground plane by accounting for this displacement along the z-axis.

4.2.2 Forward Kinematics Solution

A vital component of the visual extraction process is the calculation of the forward kinematics solution of the camera sensor, in relation to the ground plane. By utilizing the estimation of the ground plane, the robot is placed onto it by computing the position and orientation of the camera sensor. This relies on joint sensor data from all the rotary joints of the robot.

It is worth mentioning that although this process can provide a means to "place" the robot in the soccer plane, it still depends on the correctness and precision of the forward kinematics solution. One important aspect to this is that joint sensory input contains an systematic error that is caused the sensor itself. Every single joint features a manufacturing error, each one accumulating in the non-linear process that produces Cartesian space data from joint data. Furthermore, due to manufacturing imprecision, the camera sensor itself is less than ideally aligned to the head of the robot.

These errors can be systematically quantified for each individual robot and be accounted for. For the scope of the vision system, a single configurable parameter was introduced to account for systematic discrepancies of kinematics. This is namely the **camera pitch offset** whchi quantifies any systematic errors in the vertical orientation of the camera in relation to the ground plane. This is estimated manually by placing the robot on a predefined position on the field, and adjusting this value so that all measurements of observed landmarks converge into the expected value. This single value can greatly enhance the quality of the results, due to the nature of geometric calculations as explained later.

4.3 Processing Pipeline

The detection of useful objects through vision starts with the acquisition of an image from the camera. First a timestamped image is acquired from the hardware. Then, this timestamp is used to iterate through the joint sensor data stream, and match the message that contains the correct data to utilize projective transformations of the image. This function is provided by the messaging framework itself.

The main part of the Vision system, consists of an efficient scan for *regions of interest* on the image, separated by color, as classified by a color segmentation process. One interesting complication to this, is the fact that the largest amount of processing power is spent on task of reading a pixel value, and converting it into a color, though segmentation. This is a major effect of the limited cache of the small CPU of the robot: each new pixel that is fetched from the main memory is essential a *cache miss*. Therefore, reads to the image are limited, and this is a major design challenge. To overcome this, pixel are selectively sampled from the image, not all pixels are tested.

Initially, to identify possibly useful regions, a grid of pixels is processed. Pixels on the grid that are interesting, are grouped by color, since unique objects that are of interest can each be coupled with its color. Each group can be locally traced for interesting shapes, which in turn can be validated or rejected as useful objects.

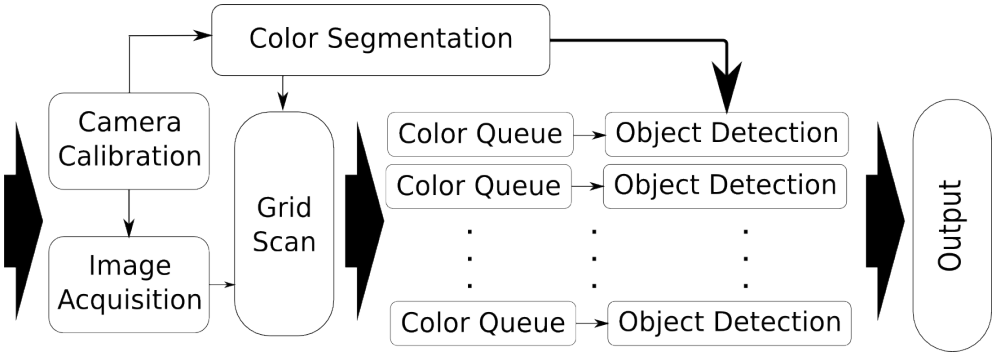


Figure 4.2: Flow of visual processing

4.3.1 Color Segmentation

To categorize image pixels into discrete colors, a color segmentation process is employed. This procedure converts the pixel values from an arbitrary 3-valued colorspace (typically Y-Cb-Cr, the native output format of the camera), into a discrete set of colors. This set is preselected to be the basic colors that can found on the soccer field: green (field),

white (lines-robots), red (ball), yellow (goalpost), and sky-blue(goalpost).

The classification data has been chosen to be stored in look-up table. This table, is *indexed* by the 3-valued pixel and the retrieved value is the resulting color. It is a 3-dimensional array and is essentially a mapping from the original color space to the segmentation output.

This approach decouples the color classification into a separate (possibly offline) process. In depth analysis of such a process, is beyond the scope of the system. Kouretes currently use an offline tool, *Kouretes Color Classifier* or KCC [28]. KCC uses captured images from the robot and, using manual selection of regions, it constructs a compatible table to be used by the Kouretes code-base. This process is improved by utilizing the exposure correction factor as computed by the camera calibration. This factor is computed as follows:

Let o be the output value of a given color channel, for a given object under optimal exposure time t_o settings. Based on the camera specifications , an accurate model for o is:

$$o = k \cdot log_b(g \cdot r \cdot t_o)$$

where :

k**:** is the digitization scaling, intrinsic to the sensor.

b**:** is the digitization base, intrinsic to the sensor.

g**:** is the analog *gain*, intrinsic to the sensor, controlled by the gain setting.

t_o**:** is the *exposure time*, controlled by the exposure setting.

r**:** is radiant flux that reaches the sensor. This is integrated over time t_o and scaled by g in an analog process.

Assuming an adjusted exposure time t_a, the expected output of the sensor is:

$$a = k \cdot log_b(g \cdot r \cdot t_a)$$

The relation between o and a can be established as:

$$\begin{aligned} o = k \cdot log_b(g \cdot r \cdot t_o) &= k \cdot log_b(g \cdot r) + klog_b(t_o) \\ a = k \cdot log_b(g \cdot r \cdot t_a) &= k \cdot log_b(g \cdot r) + klog_b(t_a) \end{aligned}$$

Therefore:

$$o - a = k \cdot log_b(g \cdot r) + klog_b(t_o) - k \cdot log_b(g \cdot r) - klog_b(t_a)$$

$$o = a + klog_b(\frac{t_o}{t_a})$$

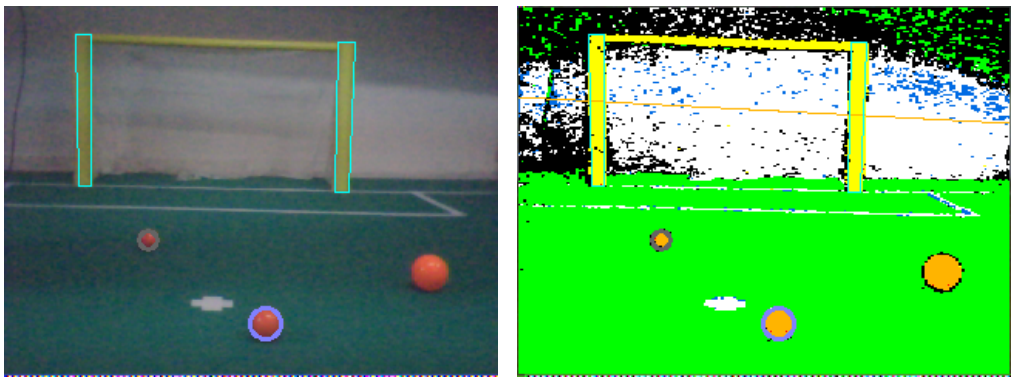
After experimentation, using measurements from images k and b were selected as:

$$k = 64 \quad and \quad b = 2$$

Since the color table requires optimal exposure for correctness, the above formula is used to estimate these from values produced with an adjusted exposure. To obtain the optimal value of a pixel, for all raw channel values of the sensor (in RGB) r_a, g_a, and b_a, we can apply the above formula to obtain t r_o, g_o, and b_o that represent the optimal RGB values. However since the images are internally transformed to YCbCr, we must propagate this transformation to received YCbCr values. It can be shown that this transformation leaves Cb and Cr unaltered, and produces an expected optimal Y value y_o from a measured one y_a as:

$$y_o = y_a + klog_b(\frac{t_o}{t_a})$$

This is intuitively correct, as brightness information in YCbCr is only concentrated in the Y channel, and Cb Cr represent chromaticity which is something unaltered by exposure time. All sample images provided hereafter, are captured using an adjusted exposure time and compensated for this in the segmentation process, unless otherwise noted.



(a) Input Image **(b)** Segmented Image

Figure 4.3: Result of color segmentation, with by markings of recognized objects

4.3.2 Region of Interest

Having established a methodology to classify pixels into colors, it is now possible to process the image for regions of interesting colors. A set of pixel groups is established,

where any unexplored regions are stored. This set is complemented by a set of explored regions, so that processed parts of the image are excluded in case multiple pixels of the same unified region are found during the initial grid search phase. This set contains both regions that have been classified as valid objects, but also rejected ones.

Segmentation Noise Although the segmentation process has been proven fairly robust, there are occurrences of misclassified pixels. This is something that must be taken into consideration, and all the pixel processing methodology has been developed with that in mind. A set of configurable parameters has been introduced, with the purpose to adjust the relaxation of required color segmentation effectiveness.

Generally, color regions are explored radially from a start point, and during the exploration a maximum amount of N consecutive pixels can be misclassified. This overcomes the possibility of incorrectly stopping the search due to imperfections in the image that result in misclassified pixels. This relaxation is also used in the initial grid scan phase, and during all object exploration methodologies.

With this in mind, an heuristic optimization has been added to the relaxation process: instead of linearly processing pixels along a dimension, the image is processed in leaps of N pixels. When a different than expected color is reached, the algorithm backtracks linearly up to N pixels. This effectively reduces the amount of processing by a factor of at most N. Typical speed ups that have been observed are in the region of $N/2$ in real environments, with less than ideal color segmentation performance.

Scanning grid The first exploration of regions consist of selectively traversing the image in a sparse grid. This grid essentially draws possible regions from the image, but server multiple purposes. The grid, consisting of MxN pixels, most typically $32x32$, is constructed as:

- Using the forward kinematics solution, the *camera roll* is used to orientate the rectangular grid onto the image. This means that pixels belonging to the same row, are placed in parallel to the horizon.

- The spacing of the grid in the horizontal direction (with regard to the physical horizon) is uniform.

- The spacing of the grid in vertical direction (with regard to the physical horizon) is a monotonically decreasing one. This means that regions on the lower part of the image are more sporadically tested. Naturally, as we move from the horizon

down, due to perspective, the image captures objects that are closer, and therefore they appear larger. This permits a more relaxed density in the grid.

The selection of M and N is critical for the maximization of the performance of this early search. Too sparse a grid, and this early scan might overlook an important object. Since the smallest useful object on the field is the ball, the grid has been selected so that the ball cannot systematically fall in an blind spot.

The iteration of the grid pixels is done from left to right and from the bottom up, with respect to the ground plane. The expected background color of the scene is green and white, and these colors are ignored. All other colors are of interest and are stored into color queues. Each vertical set of pixels is processed independently from the rest, and further processing of it stops when one of the following conditions is met:

- The line reaches a region that is not part of the field colors: white, green, and red (the ball). The last valid pixel is stored as a boundary for the field limit on that region.
- The line reaches the horizon. This can happen if the field meets a white surface such as a robot. However, no useful objects can exist above the horizon, and any further exploration is meaningless.

The set of points where the exploration stops, forms a field horizon.

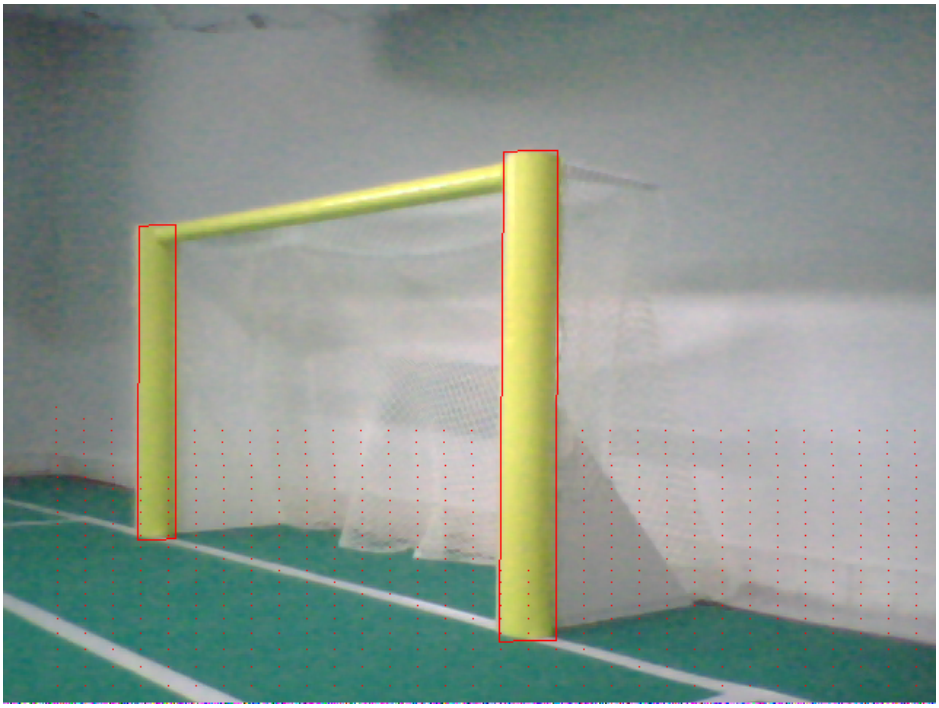
4.4 Object Extraction

Given a set of possibly interestingly colored pixels in the image, the goal now is to process these further, and establish an approximate contour of these areas, with the intent of mapping them into contours of known objects. Each useful object is dealt with differently, owning to different geometric properties of their *perspective projections* onto the image.

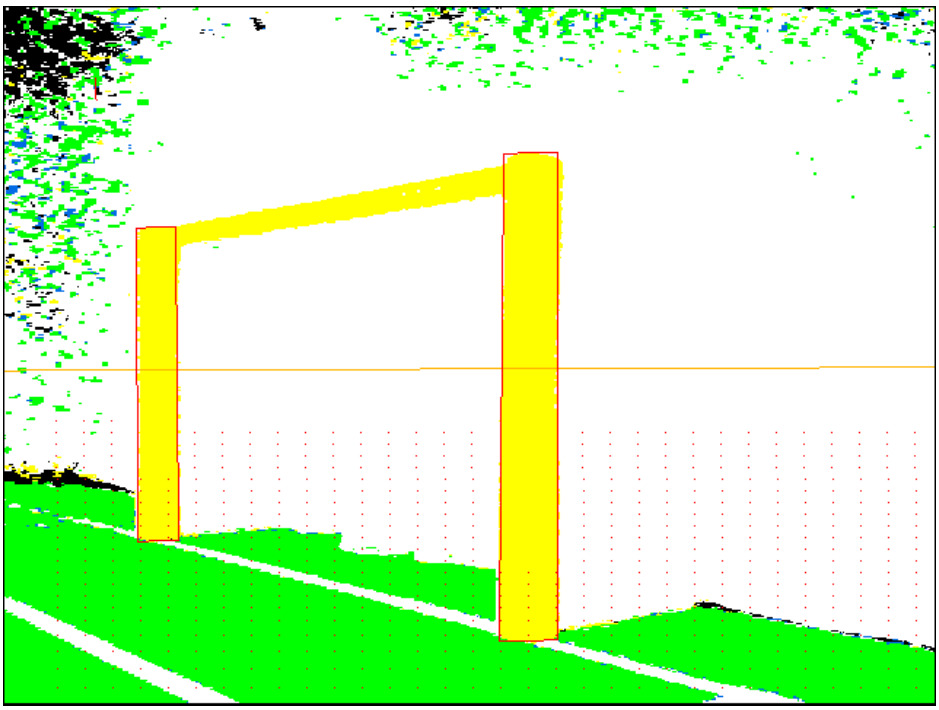
4.4.1 Camera coordinate systems

To correctly extrapolate real data from the image, it is necessary to define the transformation from the data pixels to a logical 2-D coordinate system, that can be extended into the 3D Cartetian space.

Image coordinate system The first coordinate system can be derived from the memory arrangement of the image buffer as it is produced by the camera. This coordinate system is essentially the indexing of the memory array:



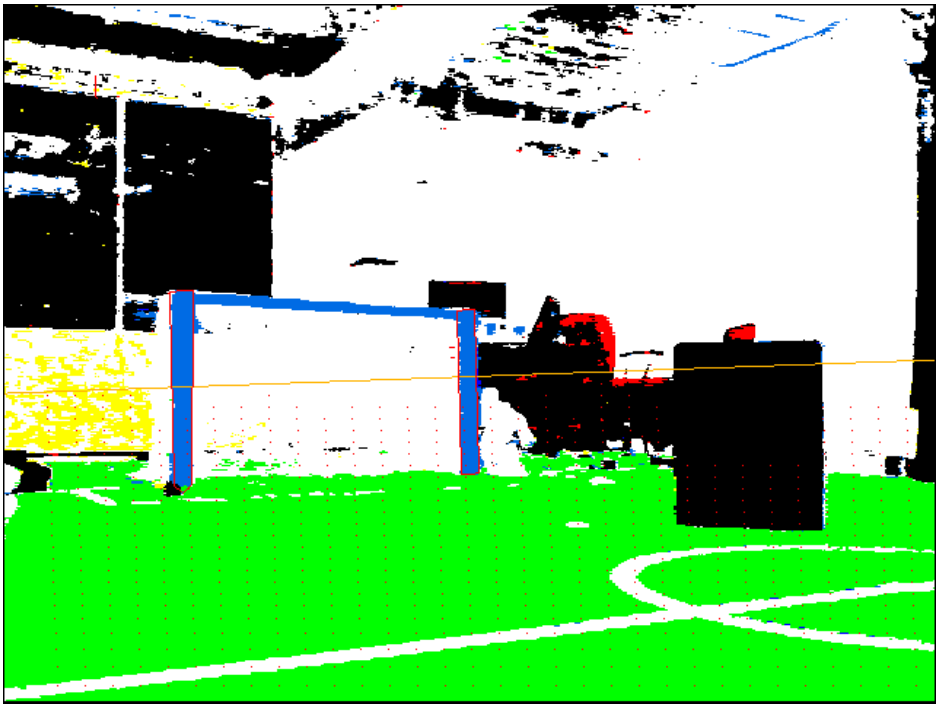
(a)



(b)

Figure 4.4: Grid scan process, with markings of the recognized objects

(a)



(b)

Figure 4.5: Grid scan process, with markings of the recognized objects

- The Origin of the coordinate system is at the top left corner, with the first pixel placed there.
- The axes do not follow the Cartesian convention. Instead the x-axis extends horizontally the right, and the y-axis extends to the bottom of the image.

All pixel processing work is done in this coordinate system.

2-D Camera Coordinate system One important intermediate coordinate system is the 2-D Camera coordinate system. This is defined as a Cartesian system that places the origin at the center of the image. Assuming a resolution of wxh the origin is placed at exactly $(w-1)/2, (h-1)/2$ along the image, half-way across it. The logic that dictates this, is that the center of the image coincides with the intersection of the x-axis of the forward kinematics camera frame, with the image plane. A point in image coordinate system p can be transformed to the 2-D Camera space as :

$$c_i = \begin{bmatrix} p_x - \frac{w-1}{2} \\ -p_y - \frac{h-1}{2} \end{bmatrix}$$

3-D Camera Coordinate System This system is the reference frame for the camera, provided by the forward kinematics. It extends the 2-D camera coordinate system by placing the image plane perpendicular to the x-axis of the 3-D system, at distance d (focal distance in pixels). d is essentially intrinsic to the sensor, and it is computed from the angular *field of view (FOV)* as:

$$d = \frac{\sqrt{w^2 + h^2}}{2 \cdot tan(\frac{FOV}{2})}$$

This essentially fits the plane onto the x-axis, at the distance that is dictated by the sensor specifications.

The 2-D coordinates are transformed to 3-D by substituting the x-axis to the -y-axis and the y-axis to the z-axis. This forms a Cartesian coordinate system that extends "forward" along the new x-axis.

$$v = \begin{bmatrix} d \\ -c_x \\ c_y \end{bmatrix}$$

Pixels expressed in this coordinate system essentially represent a variable length vector that starts at the axes origin and, passes through the pixel coordinates on the image plane.

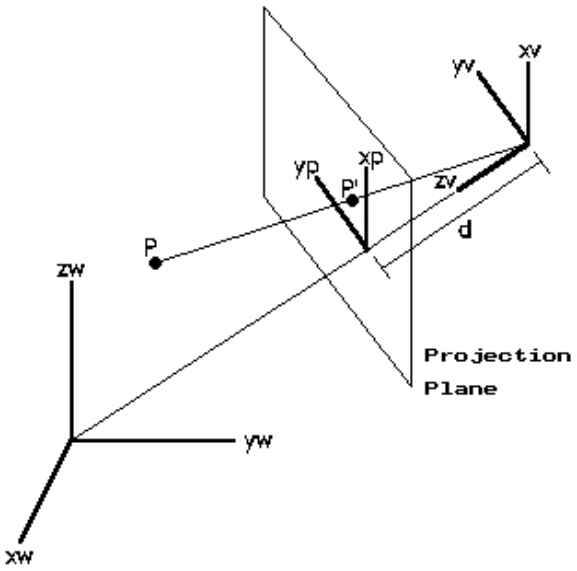


Figure 4.6: 3-D camera coordinates: v is the origin of the camera coordinate system, and the ground plane coordinate system is placed externally at w.

This vector can be scaled as desired, and this provides the opportunity to identify the projection of a point in space onto the image plane. The inverse is also possible, given a set of restrictions, one can estimate where the pixel projects. This is a algebraic representation of the *Collinearity Equation* that describes the perspective projection. Also given 2 pixel vectors, it is possible to compute the angle between them. This is far more precise that a projection operation, since it only involves the sensors, and any external component errors (such as joints), are irrelevant.

4.4.2 Ball Detection

The first object that the system must effectively detect, is the soccer ball. It can be modeled as a perfect sphere of diameter N that lies onto the ground plane. This provides important restrictions for any *red colored* regions, to be classified as balls. For extensibility, the procedure can detect multiple balls, but currently only the nearest one is selected as the playing ball, the rest are just discarded.

Projection Model Generally, the perspective projection of a sphere onto a plane is an ellipse. However, due to the limited *field of view* of the camera, it can be approximated with little loss as a circle. This greatly simplifies the image processing scheme.

To fit a circle from a red pixel region, only 3 points on the circumference are needed:

Given points p_1 , p_2, and p_3 on the image plane, first the slopes of the lines formed by p_1p_2 and p_2p_3 are computed:

$$m_a = \frac{y_2 - y_1}{x_2 x_1}$$
$$m_b = \frac{y_3 - y_2}{x_3 x_2}$$

If $m_a = m_b$ then the 3 points form parallel lines, and no circle exists.

If m_a or m_b is an infinite slope ,this means that 2 points form a vertical line. This can be solved bu rearranging the points, and solving again. The center of the circle then becomes:

$$c_x = \frac{m_a m_b (y_1 - y_3) + m_b(x_1 + x_2) + m_a(x_2 + x_3)}{2(m_b - m_a)}$$
$$c_y = \frac{c_x - \frac{x_1 + x_2}{2}}{m_a} + \frac{y_1 + y_2}{2}$$

This approach can match partially visible balls given enough of the contour of the ball being visible.

Region bounds Every pixel that the grid scan process has identified as the ball color, red, is explored further, to locate the red region boundaries:

1. The region's approximate center on the image is established. This is done by linearly exploring the bounds of the region in the up-down and left-right directions. The center of the region is chosen to be the center of the region as established by the bounds.

2. New bounds are explored radially. A total of 8 points are established along the circumference of the region. And all possible combinations of 3 points are evaluated, and a circle is produced from them. The circle that has the largest radius, is selected. This is done to avoid errors in the process, due to color segmentation classification failures on a part of the ball. Typically this is due to cast shadows from robots or highlights from light sources.

The center of the circle, passes though the center of the ball, and therefore is a distance $N/2$ from the ground.

Physical Modeling Given the fitted circle, a ball that is projected to region must conform to some physical restrictions. Assuming a known ball diameter, the angle between the center of the image and one point in the circumference, $\delta/2$, has a closed form relation to the distance D of the ball from the camera. This is the *angular diameter* formula. Assuming a known diameter d the following is true:

$$\delta = 2 \arcsin\left(\tfrac{1}{2}\, d/D\right) \tag{4.3}$$

Solving for D yields:

$$D = \frac{d}{2sin(\delta/2)} \Leftrightarrow d = 2Dsin(\delta/2) \tag{4.4}$$

Substituting d with N, one can produce a distance (from the camera) for a given circle. This distance can be projected to a ground distance D_r, using the forward kinematics solution. Given the height of the camera from the ground H_r:

$$D_r^2 = D^2 - (H_r - d/2)^2 \tag{4.5}$$

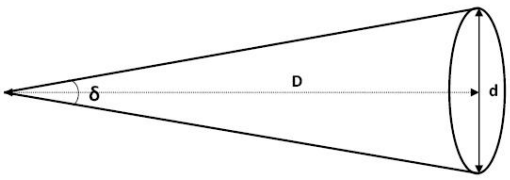


Figure 4.7: Angular Diameter: Known sphere seen from a distance D.

Each measurement is accompanied with a inherent *variance* estimation: Assuming a typical 1-pixel error at each boundary of the circle, multiple measurements are made. The variance of the calculated values is the maximum precision one can aim for.

Another possible distance estimation can be extracted, by projecting the center of the circle at a height of $N/2$ above the ground. Given the vertical angle from the horizon which corresponds to the center θ :

$$tan\theta = \frac{H_r - d/2}{D_r} \tag{4.6}$$

Each measurement is accompanied with a inherent *variance* estimation: Assuming a typical 1-pixel error, each projection is done multiple times, at the $9x9$ region of the starting pixel. The projection yields multiple results, and this variance between measurements is the maximum precision that one can aim for.

Both these methods suffer from poor performance generally, since an assumption is made for the physical size of the ball. For the sake of robustness in the process erroneous red regions, caused by foreign objects on the field, must be discarded. To do this, a combination of the two methods is employed. Lets assume that the ball diameter d is a free parameter of the solution. The combination of all the above yields:

$$\begin{aligned} 4.4 &\Leftrightarrow (\frac{d}{2})^2 = sin^2(\delta/2)D^2 \\ &\overset{4.5}{\Leftrightarrow} (\frac{d}{2})^2 = sin^2(\delta/2)((H_r - d/2)^2 + D_r^2) \\ &\overset{4.6}{\Leftrightarrow} (\frac{d}{2})^2 = sin^2(\delta/2)(tan^2\theta + 1)D_r^2 \\ &\Leftrightarrow D_r = \frac{\frac{d}{2}}{sin(\delta/2)\sqrt{tan^2\theta + 1}} \end{aligned} \tag{4.7}$$

The new free parameter d can be computed from 4.6 as:

$$\begin{aligned} 4.6 &\Leftrightarrow H_r - d/2 = tan\theta D_r \\ &\overset{4.7}{\Leftrightarrow} H_r - d/2 = tan\theta \frac{\frac{d}{2}}{sin(\delta/2)\sqrt{tan^2\theta + 1}} \\ &\overset{4.7}{\Leftrightarrow} H_r = (\frac{tan\theta}{sin(\delta/2)\sqrt{tan^2\theta + 1}} + 1)\frac{d}{2} \\ &\Leftrightarrow \frac{d}{2} = \frac{H_r}{\frac{tan\theta}{sin(\delta/2)\sqrt{tan^2\theta + 1}} + 1} \end{aligned} \tag{4.8}$$

This is an important result: one can estimate how large a ball is, and reject balls that are not the expected size. Noise in the process, makes this value vary. Combining this estimation with 4.7 yields a precise distance measurement.

To estimate the *bearing* of the ball relative to the robot, it is necessary to employ the forward kinematics solution. Bearing can be estimated as the horizontal angle between the forward axis of the robot and the point where the ball is projected to the ground, as projected onto the field.

Validation Having validated the physical properties of the located region, an extra step that improves false-positive detection rate is to visually inspect the entire circle and its surroundings. A square that is 3/2 the size of the circle is centered onto the circle, and pixels are segmented in sparse raster inside the square:

- At least 50% of the pixels inside circle must be of the correct color. If not, the object match is rejected.

- No more 25% of the remaining area can contain pixels of the ball color. If not, the the object match is rejected.

4.4.3 Goalpost Detection

The goalposts are the most useful objects to aid the self-localization process. These can be modeled as two vertical cylinders, fixed on the ground plane, jointed at the top. They have predefined width and height. An effective approximation of the projection of the vertical cylinders is a trapezoid. Each post is a effectively a trapezoid, with the long axis perpendicular to the horizon.

Physical Modeling Every pixel that the grid scan process has identified as the post color, yellow or sky blue, is explored further, to locate the region boundaries:

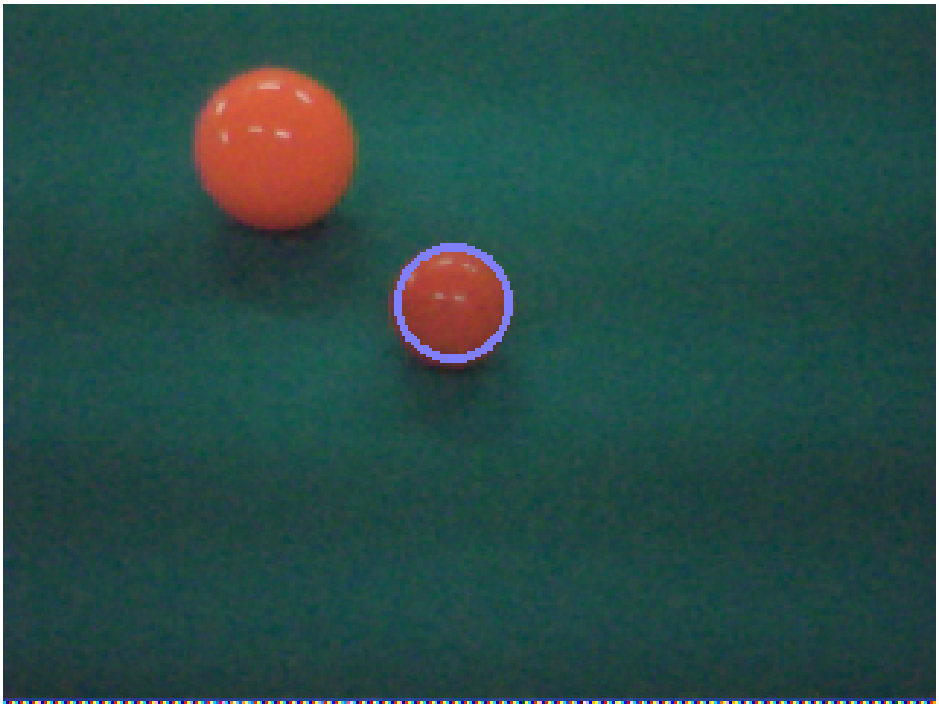
1. First the horizontal dimension of the region is explored: 2 bounds are located, to the left and right (parallel to the horizon) beginning from the input pixel. From these a middle point is extracted.

2. From the middle pixel, two scans are made, on the vertical axis (perpendicular to the horizon) this time, to locate the upper and lower bounds of the trapezoid.

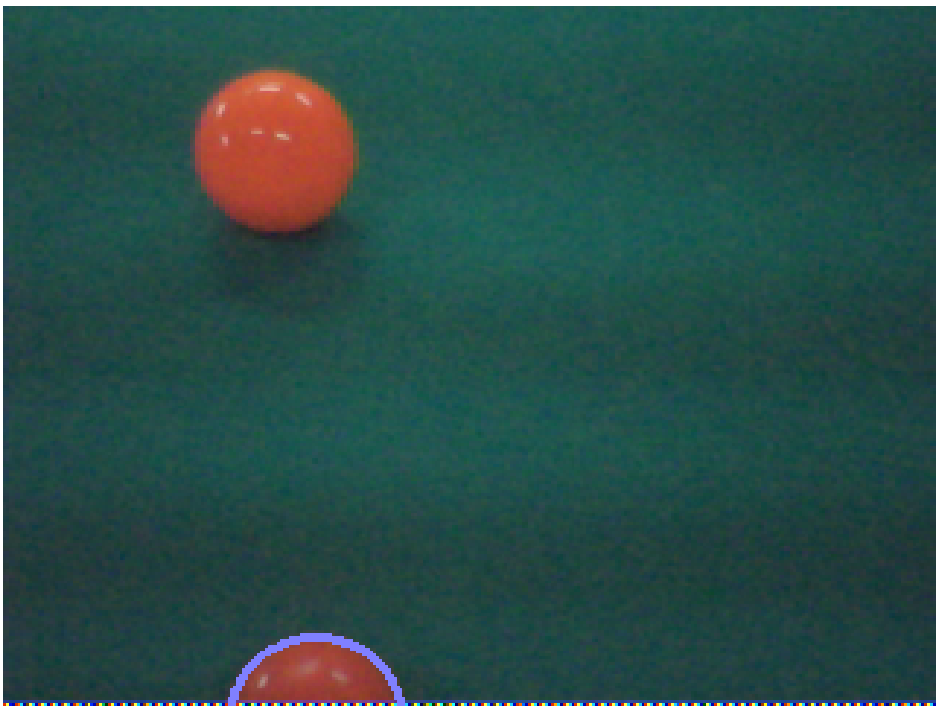
3. Having established a rough contour, width measurements are sampled across the top and the bottom. This ensures that the edges of the trapezoid lay in the corners of the projection.

4. One important detail is that due to the roundness of the projection at the upper and lower edges, especially at close distance observations, width measurements are taken approximately 1/5 of the height of the trapezoid along the edge. This ensures that the rounded regions are avoided, and also, on the top edge, the width measurement does not interfere with the horizontal post.

5. Starting from the top edge of the region, exploration is done in the horizontal axis to detect the horizontal post: If a correctly colored bound is found that lies outside the region, an appropriate marking as left or right goalpost is added to the region. If no such bound is found, this is probably due to a partially visible post, and the region is marked as “lacking top edge”.



(a) Fully visible ball



(b) Partially visible ball

Figure 4.8: Ball detection results, one correct and one rejected ball.

Bottom Validation Given the possibility of a partially visible goalpost, one extra check is performed. For the bottom edge of the trapezoid that has been detected, a region approximately the size of 1 width by 3 widths of the bottom edge is explored. This region, placed directly bellow the trapezoid, must be at least 75% filled with the field color. Should this fail, the trapezoid is marked as "lacking bottom edge".

Projection Information Utilizing the forward kinematics solution, the center pixel of the trapezoid's bottom and top edge, can be projected onto the ground and at goalpost height respectively. For each projection (if the region has not been marked as "lacking top" or "lacking bottom"), the result is a *distance* and *bearing* measurement. Bearing can be extracted by both top and bottom projections (when applicable). Each measurement is accompanied with a inherent *variance* estimation: Assuming a typical 1-pixel error, each projection is done multiple times, at the $9x9$ region of the starting pixel. The projection yields multiple results, and this variance is the maximum precision that one can aim for.

Height Information If the goalpost is fully visible, and both top and bottom edges are detected, a non-projective measurement can be made. The long axis of the vertical posts has a predefined physical size, and it is projected onto the image, having an apparent *angular size* δ. Due to the large height, the visible area corresponds to the tip of cylinder-posts, as it is viewed from the Nao. This means that the visible edge is approximately half a width of a post closer to the robot than the central axis of the post. The process uses as input:

g**:** the real height of the goalpost.

h**:** the height of the camera from the ground.

δ**:** the angular size of the goalpost, in the vertical plane.

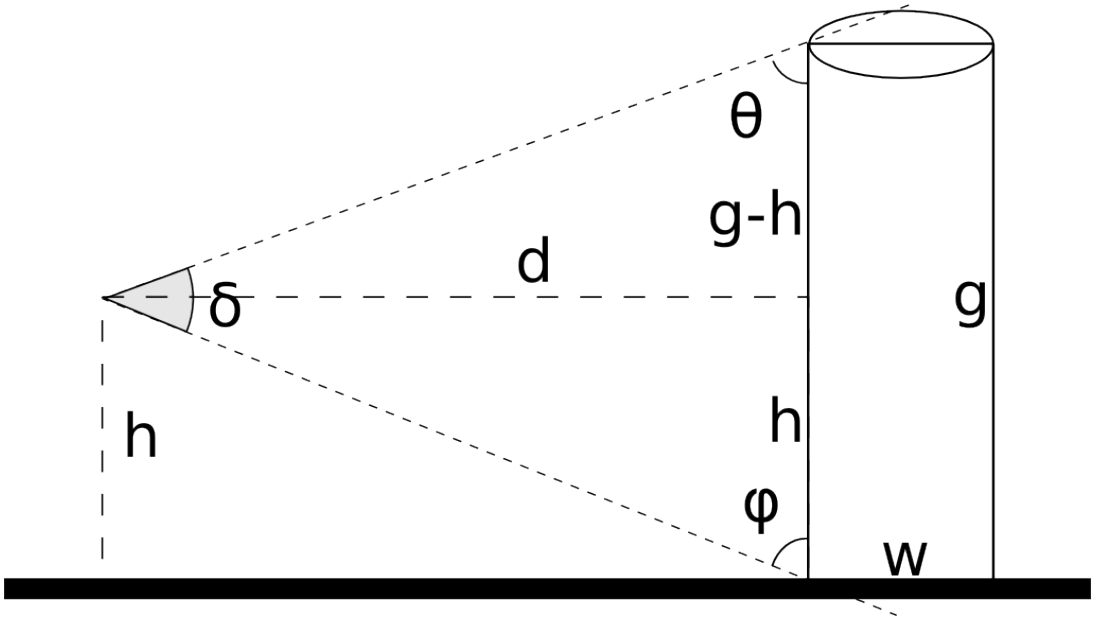


Figure 4.9: Goalpost detection: estimating the distance d from the post.

True ground distance d can be calculated starting from a known trigonometric identity as:

$$\begin{aligned} tan(\theta+\phi) &= \frac{\tan\theta + tan\phi}{1 - tan\theta tan\phi} = \frac{\frac{d}{g-h} + \frac{d}{h}}{1 - \frac{d}{h}\cdot\frac{d}{g-h}} \\ &\Leftrightarrow tan(\theta+\phi) = \frac{dh + d(g-h)}{h(g-h) - d^2} \\ &\overset{tan(\theta+\phi)=t}{\Leftrightarrow} t(h(g-h) - d^2) = dh + d(g-h) \\ &\Leftrightarrow th(g-h) - td^2 = dh + dg - dh \\ &\Leftrightarrow th(g-h) = dg + td^2 \\ &\Leftrightarrow dg + td^2 - th(g-h) = 0 \end{aligned} \tag{4.9}$$

Because angles δ, θ, and ϕ lie in the same triangle we can compute t as:

$$t = tan(\theta+\phi) = tan(\pi - \delta) = tan(-\delta) = -tan(\delta)$$

Equation 4.9 can be solved for d as a second degree polynomial, with only one solution being valid:

$$d = \frac{g + \sqrt{g^2 - 4\cdot tan^2\delta \cdot h \cdot (g-h)}}{tan\delta} \tag{4.10}$$

Validation With the physical properties of the located region established, an extra step that improves false-positive detection rate is to visually inspect the entire trapezoid and its surroundings. An area 3 times the size of the trapezoid is tested, and pixels are segmented in sparse raster inside it:

- At least 50% of the pixels of the trapezoid must be of the correct color. If not, the object match is rejected.
- No more than 25% of the remaining area can contain pixels of the goalpost color. If not, the the object match is rejected.

Measurement composition Since multiple measurements are made utilizing different methods, the combination of these into a single one is performed with an intelligent method. Assuming measurements x_i and variances $\sigma_i$accompanying these, the weighted mean of all measurements is chosen to be:

$$\bar{x} = \frac{\sum_{i=1}^{n}(x_i/{\sigma_i}^2)}{\sum_{i=1}^{n}(1/{\sigma_i}^2)}$$

The variance of the resulting measurements is chosen to be :

$$\sigma_{\bar{x}}^2 = \frac{1}{\sum_{i=1}^{n}(1/{\sigma_i}^2)}$$

This weighting method is *optimal* assuming the samples are normally distributed and independent, with the same mean.

4.5 Integration with the Platform

Since the basic purpose of this system is to feed the self-localization scheme with landmark information, the recognized objects are packed and published onto the *messaging scheme* of the platform. They are utilized for self-localization, but also for ball approaching and kicking.

Despite the fact that much effort has been made to minimize measurement errors, these are inherent in every robotic system, and they must be taken into consideration, so a filtering scheme for all data is needed. Along with the extracted data the image time-stamp is provided, for the real time filtering needs of self-localization and ball localization.

Also, an approximate solution for head motion is provided when a ball is detected. This allows the platform to follow the ball visually, as needed.

Chapter 5

Implementation

5.1 KMat

For the execution of linear algebra operations and calculations of the vision system, and for the needs of the team code in general, a minimalistic *matrix framework* was developed. KMat supports a strict subset of algebraic operations that are refined enough to provide for kinematic calculations. It focuses largely on real matrices, but can be extended to accommodate other features.

The primary design goals of KMat are low memory footprint, and calculation efficiency. While typical linear algebra libraries rely on run-time validation of the compatibility of operands (i.e. dimensional compatibility , type compatibility) and are optimized for larger matrices, KMat follows a different approach. KMat is written in **C++**, and relies on heave use of *template meta-programming* to achieve the desired qualities:

User defined element types: Each matrix is comprised of elements of a desired type. The element type is part of the matrix type. Typically floating point numbers (*float*) are selected as element types, but any type (language or user defined) can be used equally well.

Dimensionality: Matrices have a predefined dimensionality that cannot be altered throughout their usage. Matrix dimensions are part of the matrix type. KMat is already been used with 3 x 3 or 4 x 4 matrices with great results.

Compatibility of operations: Matrix operations are inherently limited to compatible operands since the operator implementation is fully dependent on the operand type. Unnecessary checks are avoided at the expense of fixed matrix properties.

Currently KMat supports two types of matrices:

Dense matrices Dense matrices of arbitrary dimensions are supported by KMat. Template type `GenMatrix<>` encloses all operations that are supported by KMat for such matrices. For computational simplicity operations currently support only some unary operators such as *inversion* and compound assignment operations such as `*=` and `+=`.

These operations do not involve the usage of *temporaries*. In contrast, all typical arithmetic operations typically require a temporary object to store the result. Advanced programming techniques can eradicate this problem, but for the sake of simplicity this was not done.

`GenMatrix<>` objects use a complex *Copy-on-Write (COW)* strategy for efficiency reasons. Matrices are essentially *memory handles* for raw data. This simplifies the process of passing KMat objects as arguments and returning them as results. Memory handles have a fixed (and small) size and can be copied with little overhead, regardless of how large the matrix itself is.

Because *COW* becomes an overhead as the size of the data becomes trivial, the scheme that is implemented has the flexibility to disable the COW strategy for specific types. Currently, COW is disabled on single column matrices (vectors), for sizes up to 3 x 1. For these matrices, template meta-programming allows the compiler to optimize the code by eliminating the COW-specific code completely and improve efficiency.

Affine transformation matrices KMat has been extended to accommodate *Affine Transformation Matrices*. Template type `ATMatrix<>` encapsulates the basic matrices of the block matrix as described in 2.6.2 each as a dense matrix. N x N affine matrices require only N x (N-1) space as one (N-1) x (N-1) and one (N-1) x 1 dense matrix and feature various other optimizations.

5.2 Color Segmentation

The CMOS camera sensor provides a chroma sub-sampled [2.5.2] Y-Cb-Cr image, with a variable resolution. Each pixel can be classified using a color segmentation process.

Bit Masks Colors are internally represented using *bit masks*. Each value represents is a set of classes, as a composition of all possible basic classes. Individual bits store the information for a single class. Currently 8-bit masks are used, providing 8 basic

color classes. In binary form this can be seen as:

$$\begin{aligned} Color1 &= \texttt{00000001} \\ Color2 &= \texttt{00000010} \\ Color3 &= \texttt{00000100} \\ &\vdots \\ Color8 &= \texttt{10000000} \end{aligned}$$

This representation supports soft-classes natively and bit-wise operations is an intuitive way to manipulate colors. Values can be compared for common subsets (logical AND) and combined (logical OR) with ease:

$$\begin{aligned} NoColor &= \texttt{00000000} \\ Color2or5 = Color1|Color2 &= \texttt{00010010} \\ AllColors = Color1|Color2\cdots|Color8 &= \texttt{11111111} \end{aligned}$$

Color table Color classification information can be stored in a 3-Dimensional *lookup table*. This table maps the 256 x 256 x 256 total number of possible pixel values, into a color value.

A common optimization employed by many teams [26][23][25] is to reduce the table size , for efficiency reasons: even if all possible color classes fit in 1-byte values, this makes the color table an impractical 16 MiB structure.

Typically, color classes span across only a small percentage of the whole colorspace, and they are distributed in fairly compact clusters. With this in mind, it is possible to sub-sample the color space classification into a smaller table. Using the bit mask color class modeling one can combine multiple values into a single one using bit wise operations.

By grouping values in each of the three dimensions into groups of 2^N values for some $1 \leq N < 8$, we can decrease the size of each dimension by a factor of 2^N. One can visualize the discrete color space divided into *cuboids*, regularly placed along each dimension. Each cuboid is merged into a single value. This can drastically reduce the required storage size, and also improve the performance of the process by limiting accesses to the main memory.

A practical size of a color table can be as low as $2^8/2^4$ x $2^8/2^4$ x $2^8/2^4$ which produces a table of only 32 KiB in size.

Chapter 6

Results

6.1 Statistics

Execution Speed Various profiling estimations have been extracted during test games. Code execution has been measured and the resulting statistical information is presented below. For each basic operation some execution time analysis was conducted, to provide minimum, maximum, and average execution times.

Each detection method is examined under different conditions, to give an overview of the execution cost at various times. Each entry below is independent from the rest. The composition of the partial operation is a complex process and the result has a different behavior, one cannot for example compute the average time spent on the KVision process by adding the average execution time of sub-components. Time measurement indicates actual cpu-time spend on each component with sub millisecond accuracy.

Table 6.1 shows clearly that the resulting process meets the required time constraints, with an average execution time of **17.04595 msec**. This indicates that the system is fairly robust. However extreme cases where the system executes with substantial delay have occasionally been observed. These cases are to be examined further. However, no systematic delay of the system has ever been observed, under any circumstances.

The results indicate that the **Goalpost detection** scheme is the most time consuming component. Further development of the KVision code, must take into account these findings, to maximize the benefits of any further optimizations.

Execution Rates To complement the execution speed analysis, an analysis of the usage of the sub-components during each cycle of operations was conducted. Table 6.2

Operation	*Minimum time*	*Maximum Time*	*Average Time*
1 Pixel read	0.65036 μsec	1.4564 μsec	1.0420 μsec
1 Pixel Segmentation	0.1504 μsec	0.4569μsec	1.2650 μsec
Color edge detection	7.1903 μsec	1.1903 msec	665.8933μsec
Grid Scan	5.2102 msec	12.2532 msec	10.2354 msec
Ball detection - none visible	81.7425 μsec	734.3784 μsec	214.4636μsec
Ball detection - 1 visible	3.1030 msec	14.3701 msec	6.1095 msec
Goalpost detection - none visible	251.2185 μsec	645.0243 μsec	352.5410μsec
1 Goalpost detection - 1 visible	4.0439 msec	21.0435 msec	17.2435 msec
KVision: no objects	6.24 msec	10.1054 msec	7.9057 msec
KVision: ball + goalpost	11.24 msec	56.103495 msec	26.5645 msec
KVision: total	6.24 msec	56.103495	17.04595 msec

Table 6.1: KVision: execution times

shows the the gathered statistics for components that are used by the detection process. Each component is executed a variable number of times as needed

Operation	*Minimum*	*Maximum*	*Average*
1 Pixel read	1068	7815	3019.033
1 Pixel Segmentation	715	3522	1915.17
Color edge detection-no objects	0	92	22.98
Color edge detection-ball and post	105	253	127.98

Table 6.2: KVision: execution rates per cycle

These indicate that on average, approximately 3000 pixels are only processed from the input image. This corresponds only to **0.976%** of the total input pixels. This is a notable achievement, since pixel processing is a computationally intensive process.

6.2 Ball Detection

An analysis of the *detection rate* of the ball was conducted. The analysis illustrates the expected performance of the ball detection component. *False positive* and *true negative* rates are shown bellow. False positive measurements where taken onto the soccer field. An incorrectly sized ball and various red colored objects were used as a

form of "decoy" for the detector. The robot was left to operate autonomously with no correct ball visible. Each detection was unavoidably a false one. True negative rates were estimated by placing the robot on a stationary position on the field, with various correctly sized balls visible from the camera. This provides an upper limit of the expected performance on the field. The results are shown in Table 6.3

Rate	*Percentage*
False Positives	0.23%
True Negatives	24 %

Table 6.3: KVision: Ball detection error rates.

6.2.1 Measurement Errors

Using a stationary robot placed arbitrarily on the field, some analysis for the accuracy and the precision of the produced *distance* and *bearing* measurements was conducted. The resulting detection of the ball, placed at various locations relative to the robot are presented in Figure 6.1. A systematic error in the bearing detection process was discovered. This error is different on each individual robot and the source of this discrepancy will be analyzed and eliminated. Ball distance estimation is fairly robust and is not a cause of trouble. A minor indication of the imprecision of the projection methodology is apparent: the "combination methodology" dominates the simple projection at all times.

6.3 Goalpost Detection

An analysis of the *detection rate* of the goalposts was conducted. The analysis illustrates the expected performance of the ball detection component. *False positive* and *true negative* rates are shown bellow. False positive measurements where taken onto the soccer field, with the goalposts removed. Various yellow and blue objects where placed randomly around, and the robot was left to navigate autonomously.Each detection was unavoidably a false one. True negative rates were estimated by placing the robot on a stationary position on the field, with the goalposts visible. The results are shown in Table 6.4. These rates show room for improvement, but are satisfactory.

Rate	*Percentage*
False Positives	8.2%
True Negatives	14%

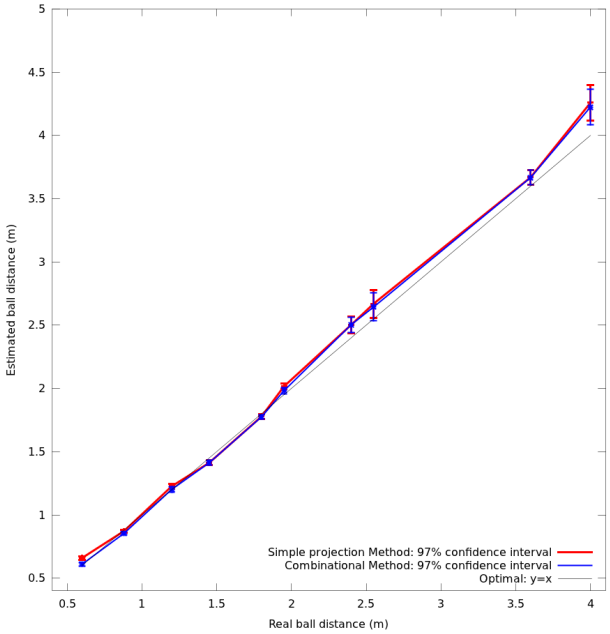
Table 6.4: KVision: Goalpost detection error rates.

6.3.1 Measurement Errors

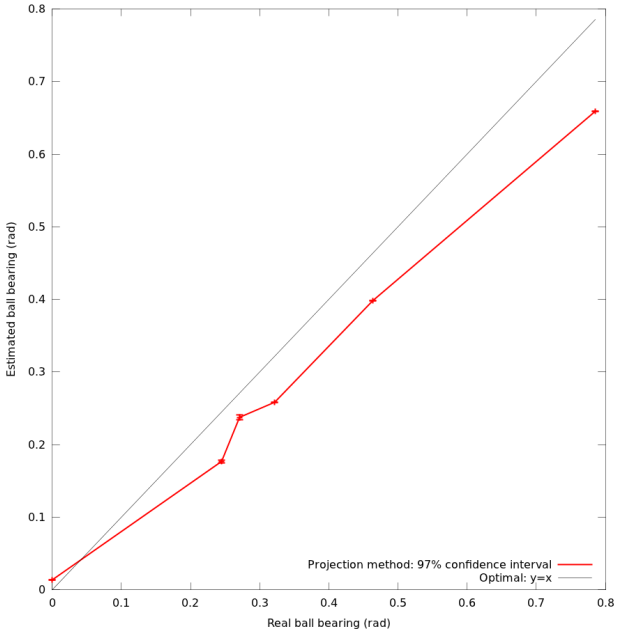
Using a stationary robot placed arbitrarily on the field, some analysis for the accuracy and the precision of the produced *distance* and *bearing* measurements for goalposts was conducted. The resulting detection of a single vertical post, at various locations relative to the robot are presented in Figure 6.2. A systematic error in the bearing detection process was discovered again. This error is different on each individual robot and the source of this discrepancy will be analyzed and eliminated. The imprecision of the simple projection methodology is demonstrated: the composition of various measurements dominates the simple projection at all times.

6.4 Conclusion

Measurements of the system show that the design goals have been met with satisfactorily. The performance of the system good, showing robustness and efficiency. The resulting data is reliable, albeit not with out errors. Various samples of the operation of the system are presented in Figures 6.3, 6.4, 6.5, 6.6, 6.7, and6.8.

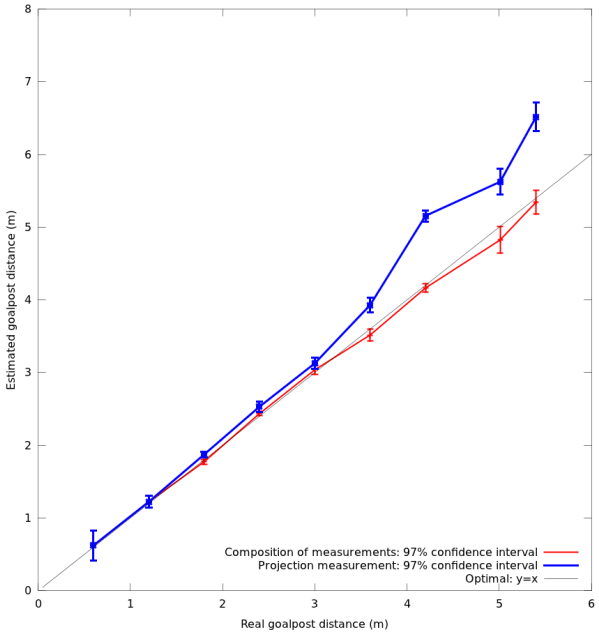


(a) Ball distance estimation

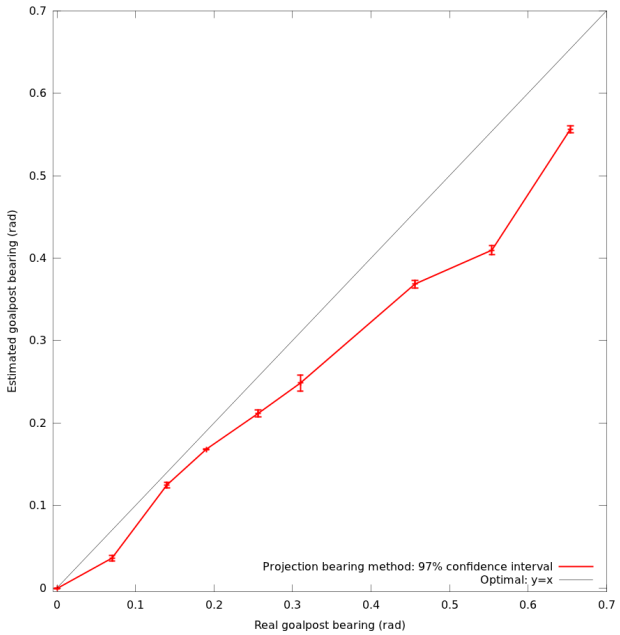


(b) Ball bearing estimation

Figure 6.1: Ball detection: measurement performance.

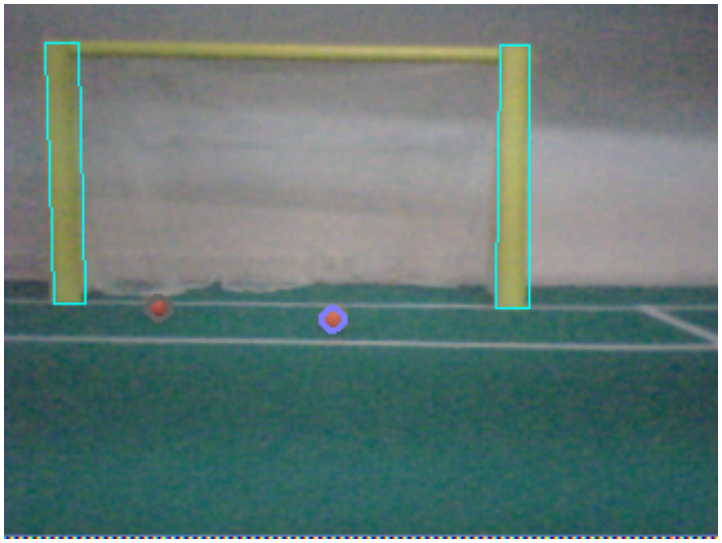


(a) Goalpost distance estimation

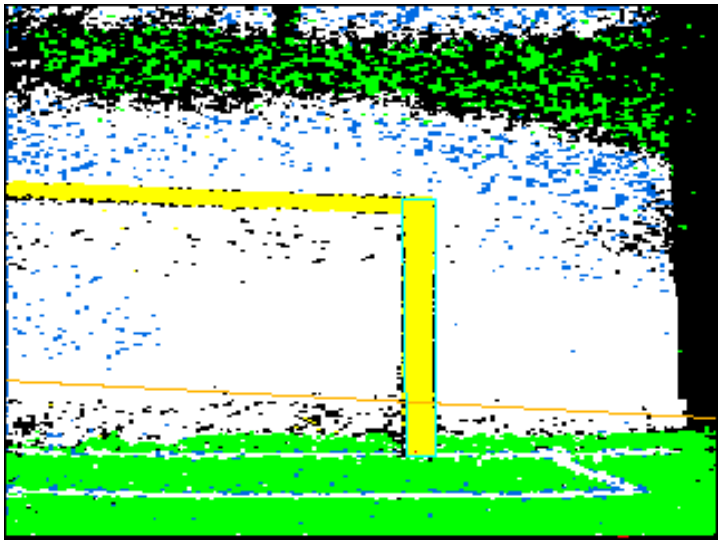


(b) Goalpost bearing estimation

Figure 6.2: Goalpost detection: measurement performance.

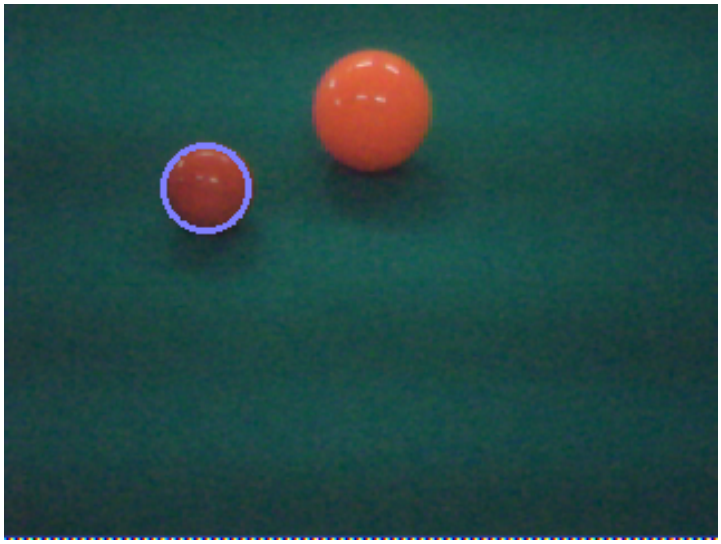


(a) Two correct balls visible and detected, nearest selected.



(b) Segmentation noise in dark environments.

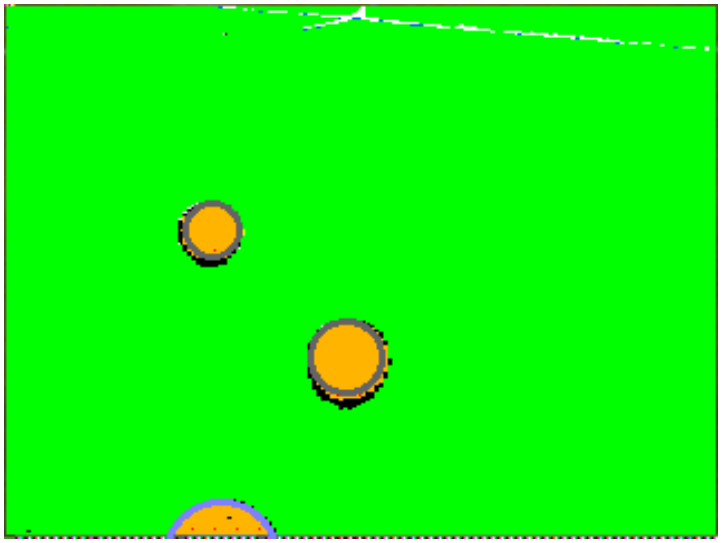
Figure 6.3: KVision: execution samples.



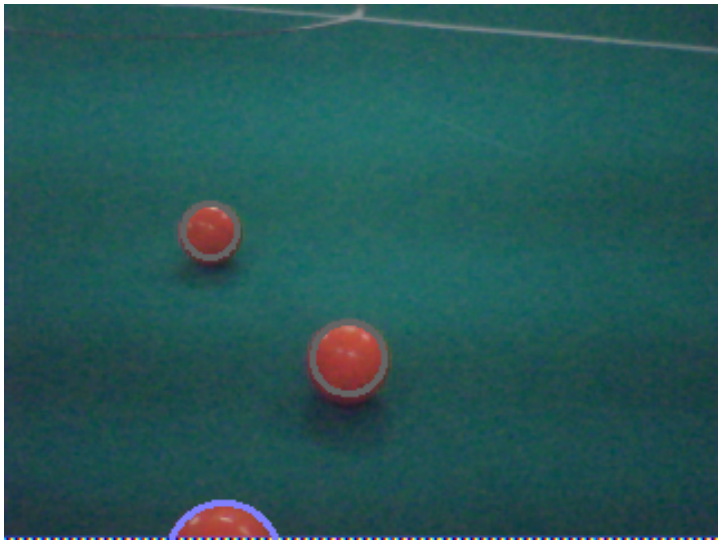
(a) Rejection of random ball.

(b) Three balls seen, two correct, nearest selected. Posts erroneously rejected.

Figure 6.4: KVision: execution samples.

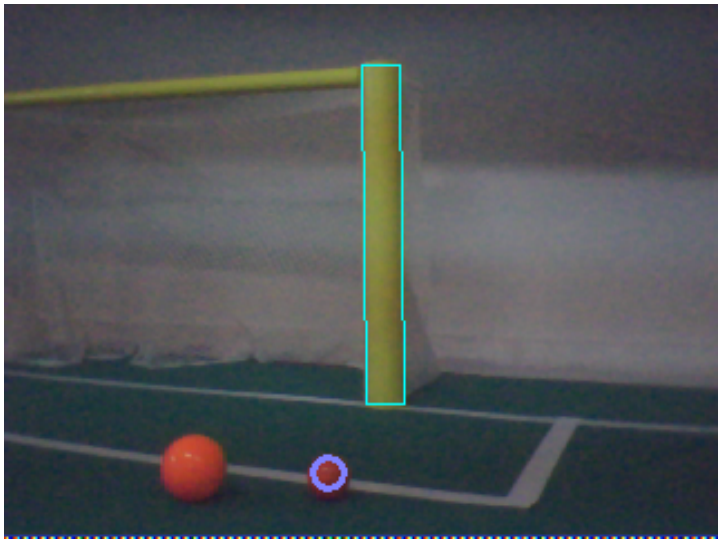


(a) 3 balls visible - segmented image.

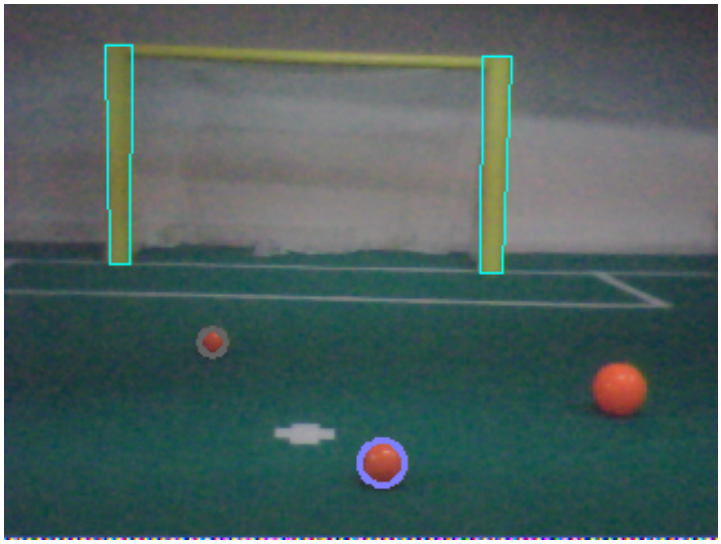


(b) 3 balls visible - input image.

Figure 6.5: KVision: execution samples.

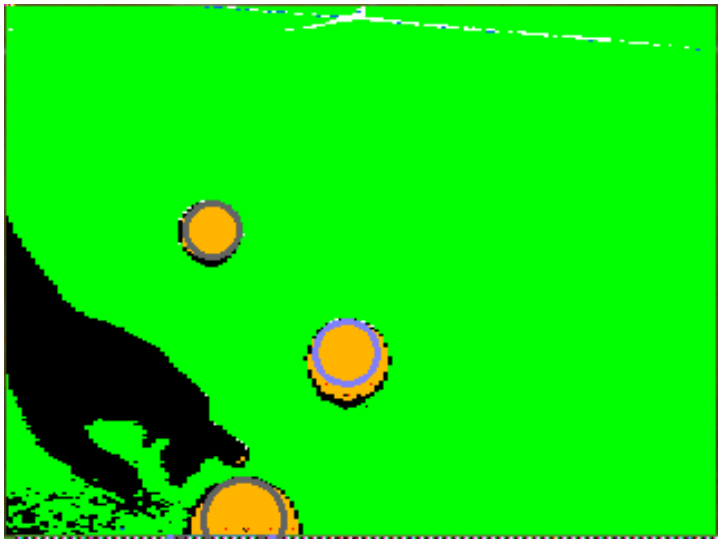


(a) 1 post and 1 correct ball.

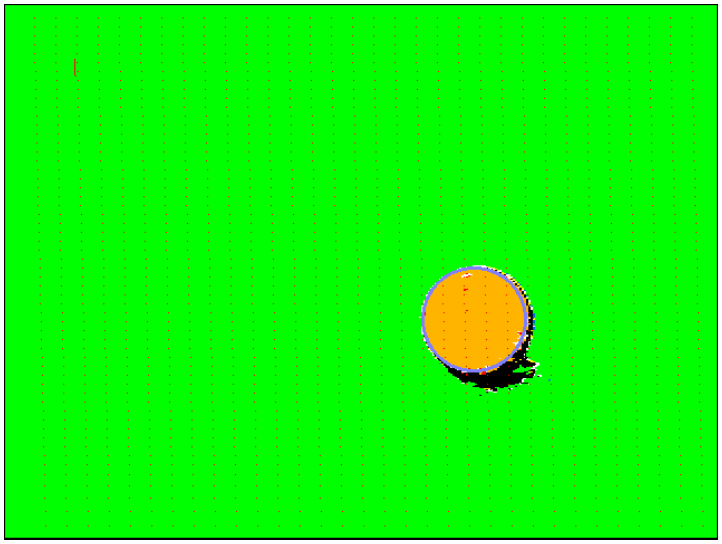


(b) 2 posts and 2 balls visible. The larger ball is rejected.

Figure 6.6: KVision: execution samples.

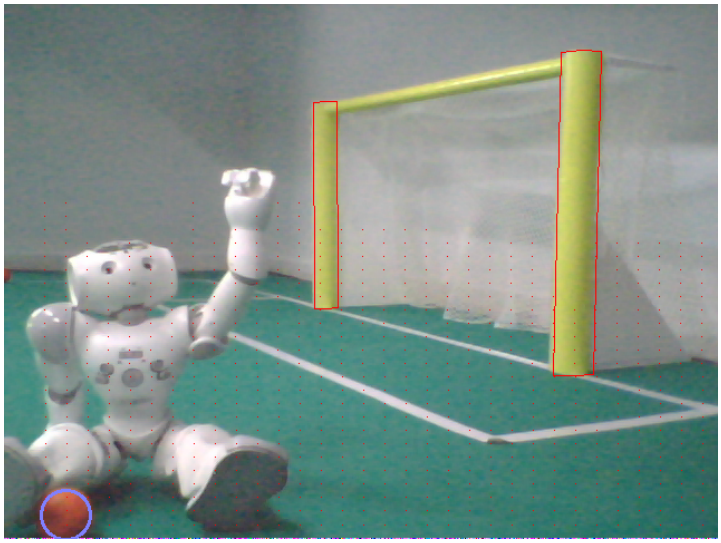


(a) Errors in the size detection of balls.

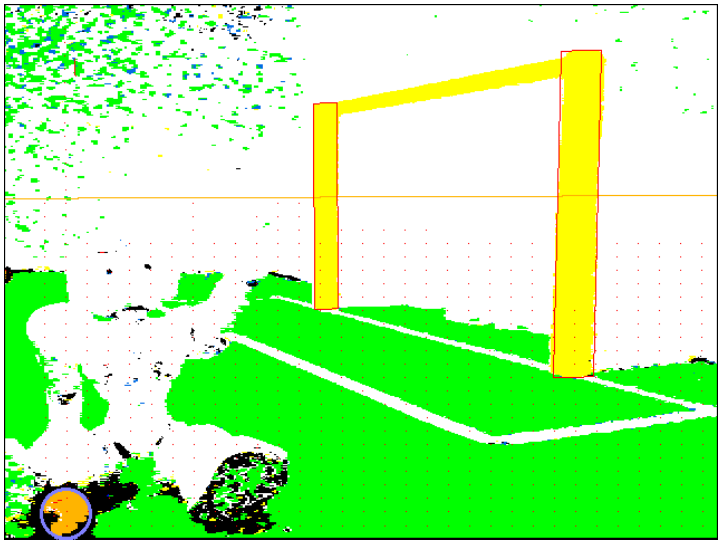


(b) Ball at a short distance.

Figure 6.7: KVision: execution samples.



(a)



(b)

Figure 6.8: KVision: execution samples.

Chapter 7

Conclusion and Future Work

This thesis presented an effective solution to the object recognition problem of the RoboCup competition. Evidence was presented that there is room for improvement over previous related work. The problem of visual object detection remains open for further improvements, and an alternative approach was presented. Results of this approach indicate that further analytical work for the robotic soccer environment could reveal more features that could be. The system has been extensively used by RoboCup team "Kouretes". It is now a vital part of the team's code and has significantly expanded its capabilities. It is efficient, reliable and robust.

7.1 Future Work

The system that has been described and explained, has been used by team Kouretes of the SPL League for the past year with notable success. Its performance has been proved on multiple occasions, albeit some weaknesses have been found.

Camera Calibration The system relies heavily on a calibration operation for the camera, that "steals" precious time during games. This can be addressed by decoupling the dependency on the camera auto-correction features. Analytical modeling of the sensor, combined with feedback from the segmentation process may be the solution to this by providing a method to adjust the camera on the fly. This should provide additional benefits since it could effectively "follow" the changes of the visual environment as the incur.

Color Segmentation Despite its proved effectiveness the color segmentation scheme requires extensive offline work to operate. Following the example of other teams [23],

color segmentation could be optimized by using perceptual information from the camera. Photometric methods that rely on reflection modeling and other such methods have been proven effective in other research fields.

Further Camera Modeling During the development of the system one important intrinsic defect of CMOS camera sensors has shown its face: the *CMOS skew* phenomenon, caused by the rolling shutter system of the camera, is apparent in the images that the system is supplied with. Analytical modeling of this phenomenon has been carried out by researchers [29]. Accounting for this phenomenon should provide significant boost to the performance of the system.

More Objects The goalpost detection scheme is vital for self-localization, but can be complemented by various other methods. Other teams [26][23][25] have implemented a line and circle detection scheme to complement the detection of goalposts. This should provide more data for self-localization.

Recognizing other robots should give further leverage to the coordination possibilities for the robots. Opponents can be avoided, and teammates can be assisted.

References

[1] Wikipedia Article. **Eye**. http://en.wikipedia.org/wiki/Eye. v, 12

[2] Jonathan Knudsen. **Drawing LEGO Models**. http://www.ldraw.org/OLD/reference/tutorials/rendering/101/. v, 16

[3] Wikipedia Article. **Graphical Projection**. http://en.wikipedia.org/wiki/Graphical_projection. v, 17

[4] S.K. Nayar, V.N. Peri, S. Baker. **Catadioptric Cameras for 360 Degree Imaging**. http://www.cs.columbia.edu/CAVE/projects/cat_cam_360/. v, 20

[5] Wikipedia Article. **Spectral Sensitivity**. http://en.wikipedia.org/wiki/Spectral_sensitivity. v, 23

[6] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. **RoboCup: A Challenge Problem for AI**. *AI Magazine*, **18**(1):73–85, 1997. 1, 5

[7] Aldebaran Robotics. http://www.aldebaran-robotics.com. 1

[8] Kouretes RoboCup Team. http://www.kouretes.gr. http://www.kouretes.gr. 6

[9] OmniVision Corp. **OV7670/OV7171 CMOS VGA (640x480) -Advanced Information Preliminary Datasheet v.13**, 2006. 8

[10] InvenSense. **IDG-300 Integrated Dual-Axis Gyro**, 2006. 9

[11] ST Microelectronics. **LIS302DL MEMS 3-axis - ± 2g/± 8g smart digital output "piccolo" accelerometer**, 2008. 9

[12] Alexandros Paraschos. *Monas: A Flexible Software Architechure for Robotic Agents*. Diploma thesis, Technical University of Crete, Greece. Department of Electronic and Computer Engineering, 2010. 9, 47

[13] Daniel D. Corkill. **Blackboard Systems**. *AI Expert, 6(9):40-47*, September 1991. 10

[14] E. Bruce Goldstein. *Sensation and Perceptio*. Wadsworth Pub Co., eighth edition, 2010. 12, 13

[15] Wikipedia Article. **Projection (linear algebra)**. http://en.wikipedia.org/wiki/Projection_(linear_algebra). 15

[16] D. Hearn , M. P. Baker. *Computer Graphics, C Version*. Prentice Hall Press, second edition, 1997. 14, 26

[17] Wikipedia Article. **Bayes Filter**. http://en.wikipedia.org/wiki/Bayer_filter. 22

[18] "Nikolaos H. Papamarkos". *Ψηφιακή Επεξεργασία & Ανάλυση εικόνας - Digital Image Processing & Analysis*. Giourdas B., 2005. 22

[19] C-Cube Microsystems Eric Hamilton. *JPEG File Interchange Format Version 1.02*, 1992. 24

[20] James Foley et Al. *Introduction to Computer Graphics*. Addison-Wesley Professional, second edition, 1994. 28

REFERENCES

[21] Wikipedia Article. **Translation (geometry)**. http://en.wikipedia.org/wiki/Translation_(geometry). 28

[22] John J. Craig. *Introduction to Robotics: Mechanics and Control*. Pearson Hall, third edition, 2004. 32

[23] Thomas Röfer et al. ***B-Human* Team Report and Code Release**, 2010. 33, 41, 75, 89, 90

[24] RoboCup Technical Committee. **Standard Platform League Rule Book**, 2011. 37

[25] Stefan Czarnetzki et al. ***Nao Devils Dortmund* Team Report 2010**, 2010. 41, 75, 90

[26] Claude Sammut et al. ***rUNSWift* Team Report 2010 Robocup Standard Platform League**, 2010. 41, 75, 90

[27] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009. 53

[28] Andreas C. Panakos. *Efficient Color Recognition Under Varying Illumination Conditions For Robotic Soccer*. Diploma thesis, Technical University of Crete, Greece. Department of Electronic and Computer Engineering, 2009. 57

[29] J. Chun H. Jung C. Kyung. **Suppressing rolling-shutter distortion of cmos image sensors by motion vector detection**. 2008. 90