LUCAS OLIVEIRA MAGGI

MaggicVision: A Fast and Lighting-invariant Computer Vision System for IEEE Very Small Size Soccer

Federal University of Pernambuco
graduacao@cin.ufpe.br
www.cin.ufpe.br/~graduacao

Recife
2021

LUCAS OLIVEIRA MAGGI

MaggicVision: A Fast and Lighting-invariant Computer Vision System for IEEE Very Small Size Soccer

A B.Sc. Monography presented to the Center of Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

Concentration Area: *Computer Vision*
Advisor: *Edna Natividade da Silva Barros*
Co-Advisor: *Veronica Teichrieb*

Recife
2021

I dedicate this dissertation to my mother Fátima, my father Natálio, to all my family, who stood-by me my entire student life, giving me financial and material support when needed. To Marcela Rodrigues who stood-by me for more than half of my undergraduate studies and came like a beacon of light in my life. To Lucas Figueiredo who firstly found and invited me to the Voxar Labs, giving me the big opportunity inside the computer science research world. To João Marcelo who has been like a "big brother" in the good sense, and challenged me every time to go beyond what I was already doing. To Maria de Lourdes and Veronica Teichrieb for being together solving for me the intricate ways of doing research in our country. To the RobôCIn team and Edna Barros for giving me the freedom to research and discover new possibilities in computer vision challenges. To Hansenclever Bassani for his time in discussions on how many problems would appear in the path of this segmentation method. To Tsang Ing Ren for being available to discuss critical points on the project. To Silvio Melo, who presented me the advantages of utilizing vectors and vectorial operations. And to Carlos Alexandre Barros de Mello who reviewed this document to the minimum details.

ACKNOWLEDGEMENTS

The completition of this work could not have been possible without the RobôCIn and Voxar Labs research groups, which have been since the early years of my undergraduate the dream teams to work with, and without the financial support not only for me but for the whole team, this work would not have been this important.

I would like to thank the Fundação de Apoio ao Desenvolvimento da UFPE, which have financially supported me during my undergraduate.

Last but not least, I would like to thank my parents for giving me all the needed support, and Marcela; without you, this would have been near impossible.

Then God said, "Let there be light", and there was light.

–The Bible, Genesis 1:3

*"Attempting to computationally solve a problem often involves three separate choices. The first is an **understanding** of the problem; the second is a choice of **mathematical tools**; the third is a choice of **computational method**."*

–Computer Graphics: Principles and Practice, John F. Hughes, Andries Van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, Kurt Akeley.

ABSTRACT

In robotic competitions, as well as in industry, it is necessary an extra care to the performance on the software and hardware used, since these devices need to match more intense operation requirements and real-time response. In the computer cision scope there are many ways to execute an image segmentation, being one of them by color. In the context of the soccer robot competition IEEE Very Small Size Soccer (VSSS), we have that computer vision is one of the proposed challenges in this category. Each team uses its own vision system, therefore they can present very distinct times to configure and calibrate their respecive systems, usually in the range of 5-20 minutes. For this monography, we will present a simple calibration, that turns the whole calibration process faster and easier. Evenmore, it is robust to lighthing variation and has high-performance. The requirement of these competitions are many times defined by the proposed challenges, such as the robot control, that in this category is done based on the robot's location information extracted from the camera's image that is above the field. Moreover, the lesser is the time difference between frames, the faster will be the reaction of the system's control. In turn, high-performance becomes essential to this vision system. For detection of the robots and ball, a color segmentation process is used to classify each pixel of the image. The colored regions are then matched with each of the color patterns of the robots. To achieve this, we use a novel RGB (Weighted) Normalization that considers the original chromaticity and enforces that characteristic, a Grayscale Filter to remove grayish colors, and segmentation of the filtered colors in the Hue dimension of HSV color space. We demonstrate that it reduces the complexity of color segmentation's calibration executed by a human; simultaneously, it brings a more robust segmentation due to light-invariant properties in HSV color space, and possibility to choose between three color normalization methods (or none). This work validades the color segmentation proposal with synthetic and real images dataset. The synthetic images were generated by Blender ray traced techniques. Showing the value of multidiciplinarity in projects that tackles technical challenges.

Keywords: color segmentation, color space, color normalization, robotic soccer, computer vision

RESUMO

Em competições de robótica, assim como na indústria também, é necessário um cuidado extra na performance do *software* e *hardware* utilizados, porque esses dispositivos precisam alcançar requisitos de operação intensa e de resposta em tempo real. No escopo de Visão Computacional existem várias maneiras de executar uma segmentação de imagem, sendo uma delas por cor. No contexto da competição de futebol de robôs da *IEEE Very Small Size Soccer (VSSS)*, nós temos que a visão computacional é um dos desafios propostos nessa categoria. Cada time usa seu próprio sistema de visão, portanto eles podem apresentar tempos bem distintos para configurar e calibrar os respectivos sistemas, usualmente em torno de 5-20 minutos. Para essa monografia, nós vamos apresentar uma calibração mais simples, que torna todo o processo de calibração mais rápido e fácil, além da sua robustez à variação da iluminação e alta-performance. Os requisitos dessas competições são muitas vezes definidas pelos desafios propostos, como o controle do robô, que nessa categoria é feita baseada na informação do local do robô extraído da imagem da câmera que fica acima do campo. Além do mais, quanto menor for o tempo entre quadros, mais rápida será a reação do sistema de controle do robô. Sendo assim, alta-performance se torna essencial para esse sistema de visão. Para detecção dos robôs e bola, o processo de segmentação por cor é utilizado para classificar cada *pixel* da imagem. As regiões coloridas são então combinadas com cada um dos padrões dos robôs. Para se chegar nsso nós utilizamos uma nova técnica de normalização em *RGB* (Ponderada / *Weighted*) que considera a cromaticidade original e reforça essa característica, um filtro de tons de cinza para remover tons acinzentados, e uma segmentação das cores filtradas na dimensão da matiz do espaço de cor *HSV*. Nós demonstramos que isso reduz a complexidade da calibração da segmentação de cor executada por um humano; simultaneamente, traz uma maior robustez devido às propriedades de invariância à luz no espaço de cor *HSV*, e possibilidades de escolher entre três métodos de normalização de cor (ou nenhuma). Este trablaho também trás o *dataset* sintético junto com um *dataset* de imagens reais para validação da segmentação por cor, no qual nós usamos o Blender para gerar as imagens sintetizadas via técnicas de *Ray tracing*. Mostrando o valor de projetos multidisciplinares que atacam desafios técnicos.

Palavras-chave: segmentação por cor, espaço de cor, normalização de cor, futebol de robôs, visão computacional

LIST OF FIGURES

LIST OF TABLES

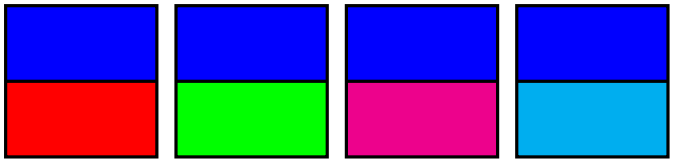
CONTENTS

1

INTRODUCTION

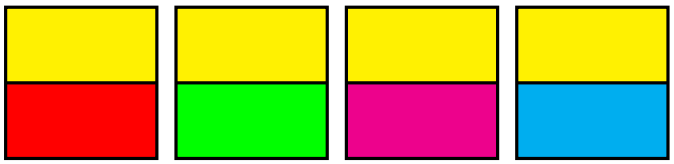
1.1 MOTIVATION

Robot soccer competitions such as Very Small Size Soccer (VSSS) and Small Size League (SSL) are heavily dependent on computer vision systems since it is the official method to acquire robot positions and orientations, defined by the rules of the competition. The robots in these categories must have color patches called *tags* on their topside to identify its team and player id (Figure 1 and Figure 2). Besides, the camera is centered above the field in an effort to avoid further perspective and more substantial lens distortions. Each team has a preparation time between matches to set up the camera, computer, robots, and software properly, meaning that they are expected to work on vision calibration, robot communication tests, and prepare to play the match. The challenge we aim to solve is related to the vision system. The vision system is responsible for detecting and tracking the robots and the ball on the field. The detection process is based on two steps: correctly classifying the pixels into one label (e.g., red, green, blue, yellow, and so on); finding colored regions that are close to each other, assuming that there will be no repetition of the orange region (the ball), neither more than three regions for each team color tag: yellow or blue. Filtering the regions by size is one way of solving possible miscount of respective regions. However, the pixel classification is usually done by defining an acceptable range of colors that belongs to a label (tag). As the competition space does not guarantee perfect light conditions, it is not trivial to define a general configuration and classify each RGB pixel into the corresponding tag. These pixels might be incorrectly set to a different color, meaning the configuration is either not well-tuned, or the light condition has changed, and sometimes we can not be perceived at the naked eye. The parameter tuning process is executed many times in the competition; due to field sharing, and the teams must change their location to a different field for a new match, which changes the environment light conditions. However, this problem brings a huge opportunity to improve the method of segmenting the color space and simplifying the parameters fine-tuning process. The seven colors we chose to segment can be seen in Figure 3 with their description in RGB.



(a) Red player (b) Green player (c) Pink player (d) Cyan player

Figure 1: Blue team color pattern.



(a) Red player (b) Green player (c) Pink player (d) Cyan player

Figure 2: Yellow team color pattern.

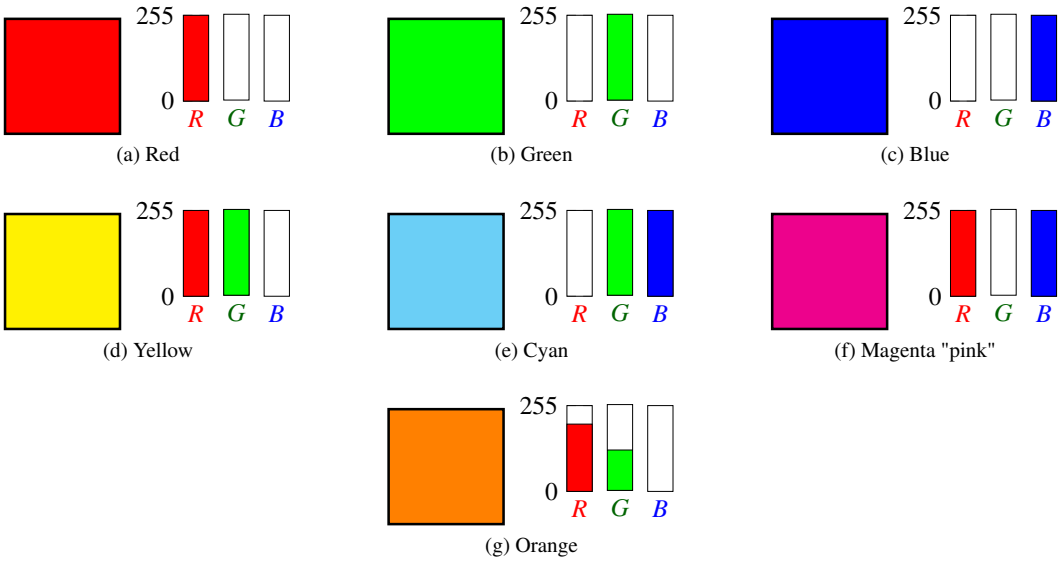


Figure 3: Color description in RGB.

1.2 HYPOTHESIS

In the context of VSSS competition, we have that the lighting setup should be able to avoid external lighting influence. However, this is impractical in most parts of the competition. Therefore, the main hypothesis is that the captured color should be sufficient for its own classification, and the major difficulty in classifying a color happens when the captured color is very different from the perceived color by a human, that takes into account the context.

1.3 OBJECTIVES

This project aims a validation dataset for a proposed method for image segmentation in the context of VSSS competition.

- Create a dataset with real images (from VSSS competition) and synthetic (by 3D modeling and rendering software) as segmentation ground-truth for the chosen lighting cases: Ideal, Normal, Challenging.
- Propose an image segmentation technique by color that presents high-performance and lighting-invariant robustness.
- Validate the proposed method by comparing to other variants.

1.4 LIGHTING CONDITION

Since the competition's physical space usually does not guarantee a perfect lighting condition (controlled lighting to an established intensity and color temperature), the calibration must be done almost every time there is a field change or time in the day. Figure 4 illustrates the position of synthetic light sources (marked as "Light source"), and natural external light source (marked as "External light source"), as well where the camera is usually positioned at a support structure and corresponding VSSS field on the floor. This light condition differs drastically from the training and test setup: a closed space (Figure 5a) with less intense light, different color temperature, and light position. Top light generates strongly illuminated areas while resulting in a higher contrast with darker areas. This effect can be seen in Figure 5a on the monitor visualization of the camera, and similar phenomena happen in some VSSS fields during the match. Figure 5a illustrates the yellow light source reflecting on the field on the top-left corner of the image on the screen and the light-blue light source reflecting below. There is a non-explicit challenge in this environment, the penumbra generated by the spectators close to the field or cloud passing by changing the external light influence in color temperature (usually making it bluer).

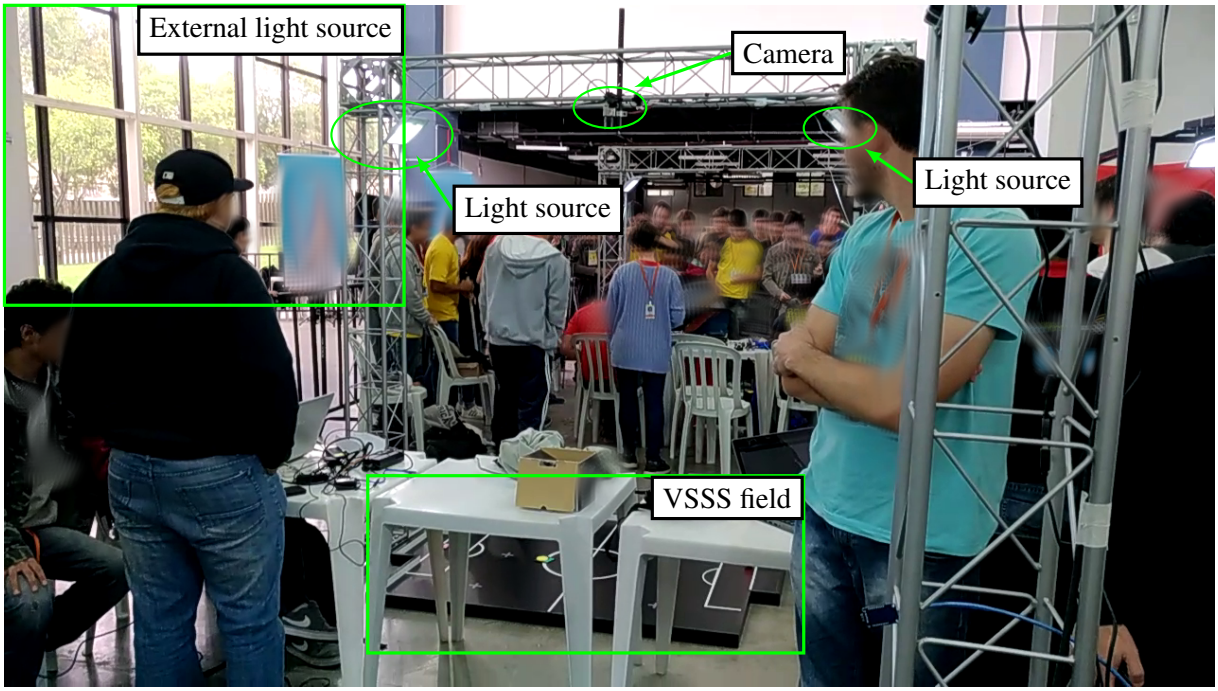
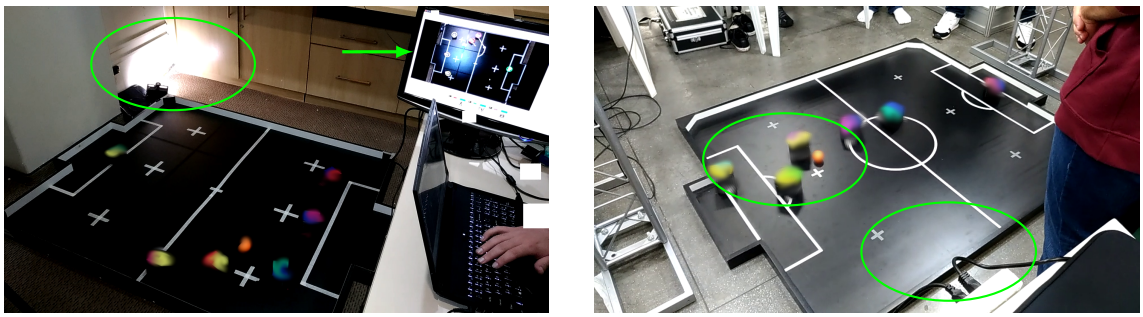


Figure 4: Competition space description.



(a) Closed space light condition.

(b) VSSS Field light condition.

Figure 5: Different light conditions of closed space (training), and VSSS field (match).

2

BASIC CONCEPTS

2.1 LIGHT

In Physics, the nature of the light is modeled by its wave-particle duality. Walker *et al.* (2013) However, we are more interested in the influence of light sources in perception, mainly in a computer vision system. For this, we propose a method to achieve better results in color classification despite some lighting interferences. It is important to remark that there is a common misunderstanding between inherent object color and the visible light color from the same object, even if they are similar colors for the observer's perception - this means that they have similar frequencies or spectrum. There is a difference between light and pigment, which the first is commonly used as both concepts, and the pigment is only used in painting or ink context. The difference can be clarified with two more concepts: Additive and Subtractive color combination.

2.1.1 Additive and Subtractive colors

Additive color is usually defined as the light colors, or emitting colors, as you (the reader) are possibly reading from a computer or cellphone screen right now. Furthermore, it is called additive because the colors are composed by the summation of different frequencies. Since the light presents the superposition property, the light spectrum from different light sources get combined in a wider spectrum. This can be seen in Figure 6. It is important to remember that, like a wave, the light may also have destructive interference. However, these effects are mostly unobservable in our scenarios. In the Subtractive colors, we have the pigment, ink color, or inherent material color. This feature means that the material only reflects some of the incident light, resulting in the observable color. Therefore, the more distinct absorbers (materials with different colors) we have, the more light the mixed material would absorb. For this, we have the Cyan, Magenta, and Yellow colors. The intersection of two primary subtractive colors reestablishes the three light primary colors. With the black in the intersection of all of the pigment primary colors. This can be seen in Figure 6.

Nevertheless, the "visible light" is what humans use to define the "visible light spectrum". Machines can capture inside and beyond the visible light spectrum. Machines can capture

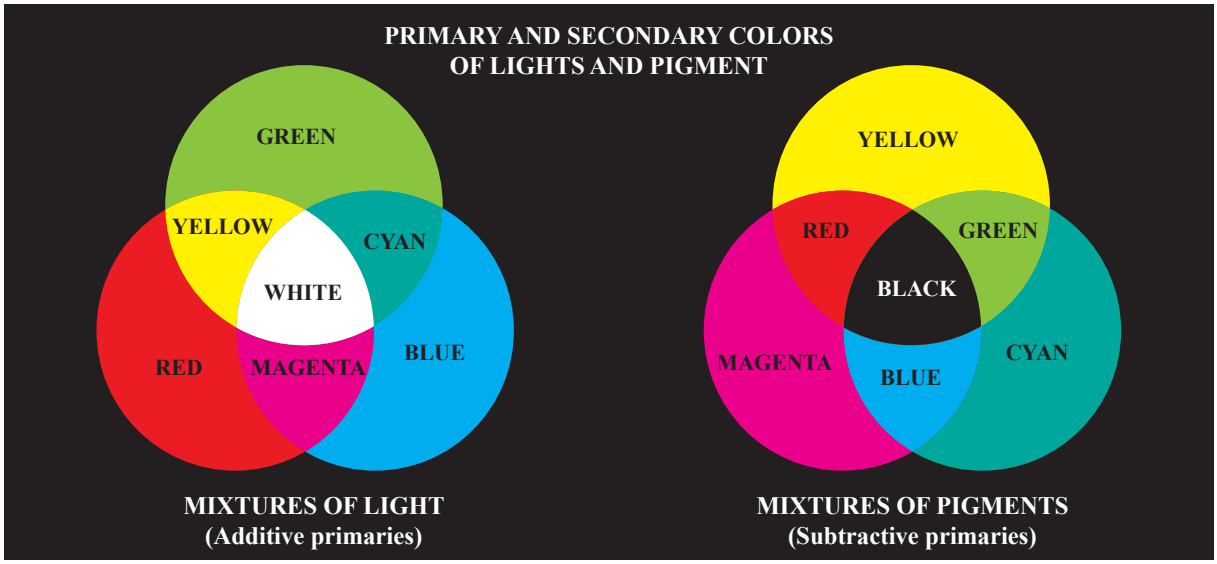
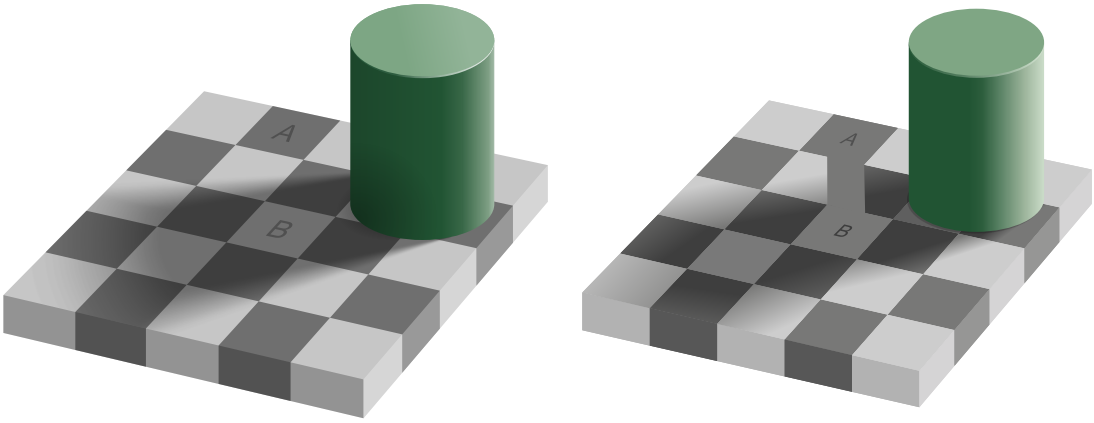


Figure 6: Additive (left) color primary Red-Green-Blue, and the intersections between two of the primary colors resulting the secondary colors. Moreover, white is the combination of all the visible light components. Subtractive (right) color primary Cyan-Magenta-Yellow, and in its intersections between two of the secondary colors resulting the primary colors. Furthermore, black is the combination of all the visible light components. Also called pigment combination. (Adapted from Gonzalez & Woods (2006))

invisible light with appropriate sensors and illumination, this includes infrared and ultraviolet light frequencies.

2.2 HUMAN VISION

The human vision is has a powerful recognition system, we can distinguish faces even if obstructed or within a low resolution or blurred image Sinha *et al.* (2006). But our vision has some difficulties when trying to define colors. One difficulty that is commonly addressed is the apparent color, which depends not only on the light source but also on the object's material. The apparent color also depends on the context and neighbor object's colors, which our brains use to determine where the light source is or if something is shadowed. For example, the classic checker optical color illusion (Figure 7), in which is presented the Figure 7a in order to ask if there is a difference between A and B checker cell colors. However, we can show that they are indeed with the same color. This concept is important to this work because the human perception may genrate contraditory color classifications and mistakenly classify a color based on its surroundings, while computers use captured color for each pixel of an image (without taking into account the context).



(a) Checker shadow color illusion.

(b) Connecting checker cells with a rectangle of the same color.

Figure 7: 7a by Pbrks (2018). Original by Adelson (1995). 7b by Pingstone (2015).

2.3 COMPUTER VISION

The Computer Vision area has grown in the last decade with the advent of cheaper and more powerful sensors. While this growth in hardware availability, the main goal of the area remained stable primarily: "*extract useful information from images*" Prince (2012). What makes the problem of color segmentation a challenging task is when there is a context to take into account. Besides, the segmentation problem faced in this project assumes that we should assume that the color captured by a camera may differ from the human perception. In a way, the computer would classify cells A and B with the same color in the Checker Shadow example (Figure 7).

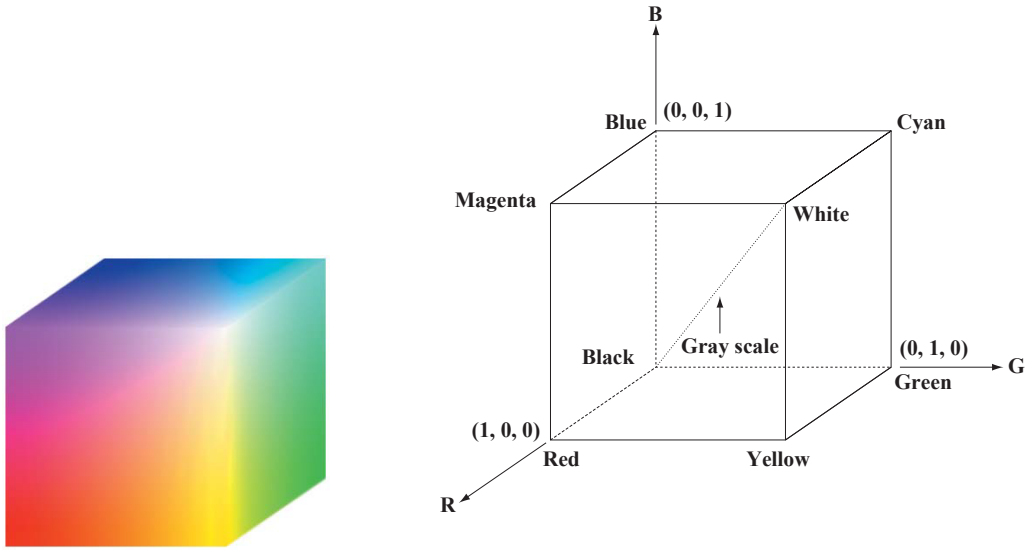
2.3.1 Color Space

To represent a color, one must decide which color space to use. Each color space has its own advantages and disadvantages beyond the numerical precision that can affect multiple conversions.

2.3.1.1 RGB

The RGB space is usually defined as a cube-shaped color space. Every channel (Red, Green, Blue) have equal dimensions; for some applications, they can vary between $[0..1]$ in some representations and between $[0..255]$ in others. The lower the values on all channels, the darker the color is. All the colors are a combination of the primary colors, that are also the three channels in this color system. As seen in Subsection 2.1.1, the white color is the combination of all the maximum values for each channel. This can be seen on the white corner (using the $[0..1]$ range) of the RGB color space representation in Figure 8a, and the corner with $(1, 1, 1)$ coordinates representing white in Figure 8b, and the origin representing the black. Other colors like yellow, magenta and cyan can also be defined in the complementary corners, in which

they are represented by the combination of two primary colors in its maximum values, while the complementary color is at its minimum value. For example, the magenta is represented as $(1,0,1)$ in Figure 8b, which means the red and blue channels in its maximum vaues, and green (the complementary color) is at its minimum value. For the primary colors we only need to maximize the value of the respective channel. For example, for the green color we can use $(0,1,0)$.



(a) Color visualization in RGB.

(b) Color definition in RGB.

Figure 8: RGB color space representation with color localizations. (Adapted from Gonzalez & Woods (2006))

An interesting fact is that the RGB channels have equal values when a color is a tone of gray, and when they have similar values (usually less than 10% of difference between channels). This will be useful in our segmentation approach in Chapter 4 to filter grayish colors and stay with *chromatic colors*. This description can be visualized in Figure 8b.

2.3.1.2 YUV

The YUV space is usually defined as a cube-shaped color space, similarly to the RGB color space. Every channel (Y - gamma, U, V) has equal dimensions; for some applications, they can vary between $[0..1]$ to $[0..255]$, in which the lower the Y, the darker the color is. Moreover, the U and V channels provide the chromaticity combination. Then, a color can be described with this definition. Similarly to the RGB color space, the white color is represented by the maximum value for the Y channel, and the chromaticity provided by U and V can be discarded when Y is at the maximum value. The YUV color space is represented by a slice in UV plane Figure 9.

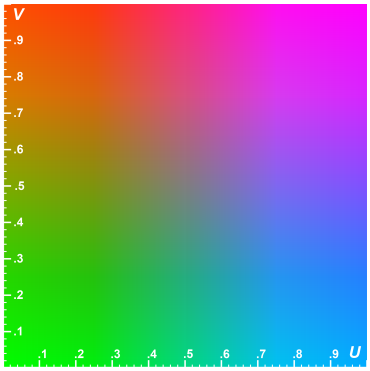


Figure 9: YUV color space slice, showing the U and V coordinates. By Mrsi (2013)

2.3.1.3 HSV

The HSV color space is usually defined in a cylindrical shape, where the hue is an angular position in the chromatic disc, the saturation is the radial distance from its center, and the value is the height in the cylinder. These caracteristics brings interesting properties, such as: the primary colors (red, green, blue) are equally distanced by 120°, the secondary colors (yellow, magenta, cyan) are also equally distanced by 120° in the chromatic disc.

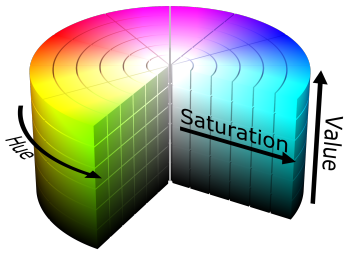


Figure 10: HSV color space represented in a cylinder. (Adapted from Datumizer (2015))

2.3.2 Color Segmentation

As an essential process for many computer vision systems, color segmentation is one of many image segmentation methods. This includes techniques that come from statistics Gonzalez & Woods (2006) to advanced machine learning algorithms, like Convolutional Neural Networks (CNN) Upcroft *et al.* (2014). These techniques aim to estimate an optimal set of configurations to subdivide a color space. According to a given set of classes, these configurations (weights and bias in Neural Network) are then used to classify each pixel in an image. In many scenarios, the color segmentation is enough to directly accomplish the detection task Li & Plataniotis (2018), although the segmentation method we try to enhance depends on correctly classifying more than two classes. The problem grows fast in complexity when we add the influence of external light sources (the Sun) in the context of slightly controlled light conditions. This fact means that even with the addition of strong artificial light sources, the external light sources may affect the

captured light color by the sensor. This case also includes shadows caused by persons that are nearby the field and clouds passing over. Our objective is to propose a fast and lighting-invariant color segmentation method that solves this problem efficiently.

2.4 COMPUTER GRAPHICS

Computer Graphics is the study area of how graphics are generated by computers. These generated graphics (or images) can be photo-realistic or non-photorealistic (some spreadsheet, plotted data, symbol). In the photo-realistic area, we have the Ray Tracing technique Glassner (1991). Since we want to simulate lighting variations, we use this technique to achieve visually similar results to what we would get from the real world. It would take more space and more concepts to retrieve the non-photorealistic techniques that are physically based sometimes, but we would not use them because the results achieved with Ray Tracing were more accurate for our application.

2.4.1 Ray Tracing

Glassner (1991) defines Ray tracing as this: "Ray tracing is a technique for *image synthesis*: creating a 2-D picture of a 3-D world.". This technique can generate realistic lighting interactions with objects, such as reflection, refraction, and scattering. The principle of this technique is from a simple but powerful concept: rays that originate from the camera and extend themselves to the scene objects, and changing directions or branching to more directions. To visualize this, one must start with the modified pinhole camera model (Figure 11). This model states that the camera position defines an origin for rays that intersects all the image plane, defining the viewing frustum region, a section of a rectangular pyramid, in which its top coincides with the camera position.

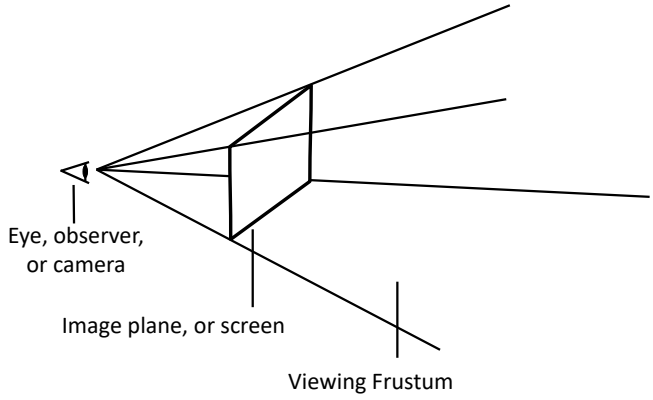


Figure 11: Modified Pinhole Camera Model. (Adapted from Glassner (1991))

Once the camera model is set and the scene is well defined, the Ray Path Tracing process can be executed. This algorithm, besides its conceptual simplicity, is very compute-intensive.

For every pixel of the image plane, the algorithm should correctly approximate the result by having more samples from the many rays it sends. It is important to remember that materials have different properties, such as the refraction index, which brings internal reflections and ray direction deviations. Moreover, for each interaction, the algorithm should branch its calculation for every casted ray after reflections, refractions, and other interaction effects, such as shadowing. Many of these effects can be seen in the algorithm's representation in Figure 12, where S_x are rays that collides, within a tolerance distance, with a light source directly; R_x are reflected rays; T_x are transmitted rays, which could be refracted or just passed through the object; and a unnamed ray at the bottom of the image representing a ray that do not reach any light source. Also, the light sources are named as L_x and E is naming the eye (camera) object pointed to the image plane (with a rectangular hole in it). All the rays are processed until there is a collision between a light source or no more collisions with scene objects (shadowing).

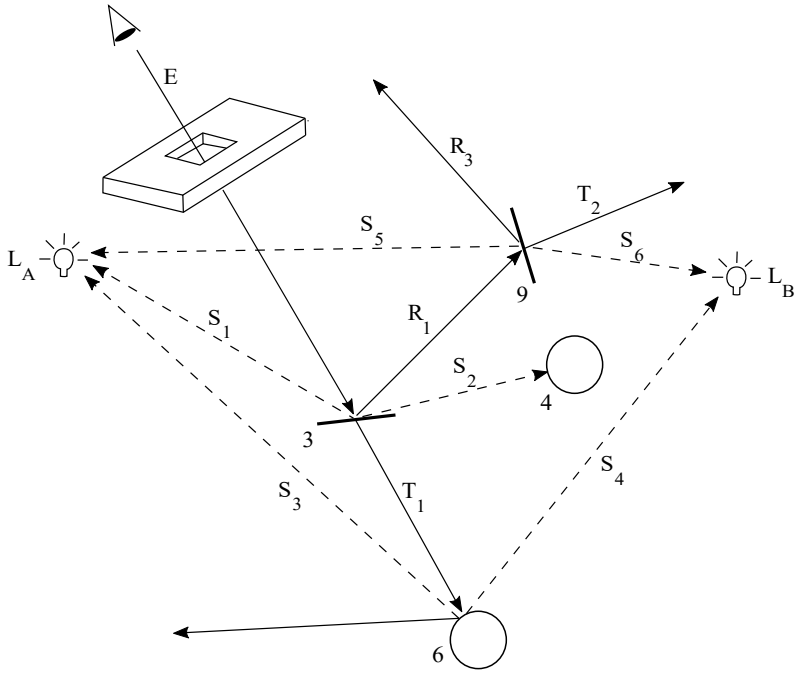


Figure 12: Ray Path Tracing interaction cases examples. (Adapted from Glassner (1991))

3

RELATED WORKS

There are many ways to segment an image Sharma *et al.* (2012), some of them are region based, graph based, clustering, and many more that can be seen on the Lalitha *et al.* (2013) survey. Some of these techniques are also being used in the robotics field Corke (2011), since they are able to solve some indoor problems, or controlled scenarios. However, there are advanced computer vision algorithms that are capable of virtually reconstructing the enviroment in real-time Corke (2011).

3.1 RGB CHROMATIC NORMALIZATION (CNT)

Works on color segmentation can be followed with color normalization to extract the chromaticity information, and clusterize similiar color values. The RGB Chromaticity Normalization was used by Finlayson *et al.* (1998) to normalize colors from two images with different light conditions. This Chromaticity Normalization method is often called as *"RGB Normalization"* or *"Chromatic Normalization"*. We have found the Comprehensive Image Normalization Finlayson *et al.* (1998) as a robust color normalization method, which removes lighting geometry and illumination color. They propose an iterative method that converges the colors of two images to a normalized version of them, thus turning similar colors in both images to the same normalized color. The illumination color problem is defined as changes in color chromaticity due to a non-white light source, affecting the comparison of colors using its chromaticity directly. Moreover, this RGB normalization was used to detect objects even with different light conditions Finlayson & Tian (1999) and the results advanced the studies in perceptual computer vision. In RGB Chromatic Normalization high values get lower, and low values get higher (see Figure 13, the vector field shows the chromatic normalization behavior in the RGB color space). This method brings a strong displacement and a simple definition of the intended behavior. And it is calculated using the Equation 3.1. Note that Figure 13 shows the three dimensional RGB space

and the vectors points to the resulting position of its own origin scaled up to be easily seen.

$$CNT(\vec{v}) = \begin{bmatrix} \frac{R}{(R+G+B)}, \\ \frac{G}{(R+G+B)}, \\ \frac{B}{(R+G+B)} \end{bmatrix} \tag{3.1}$$

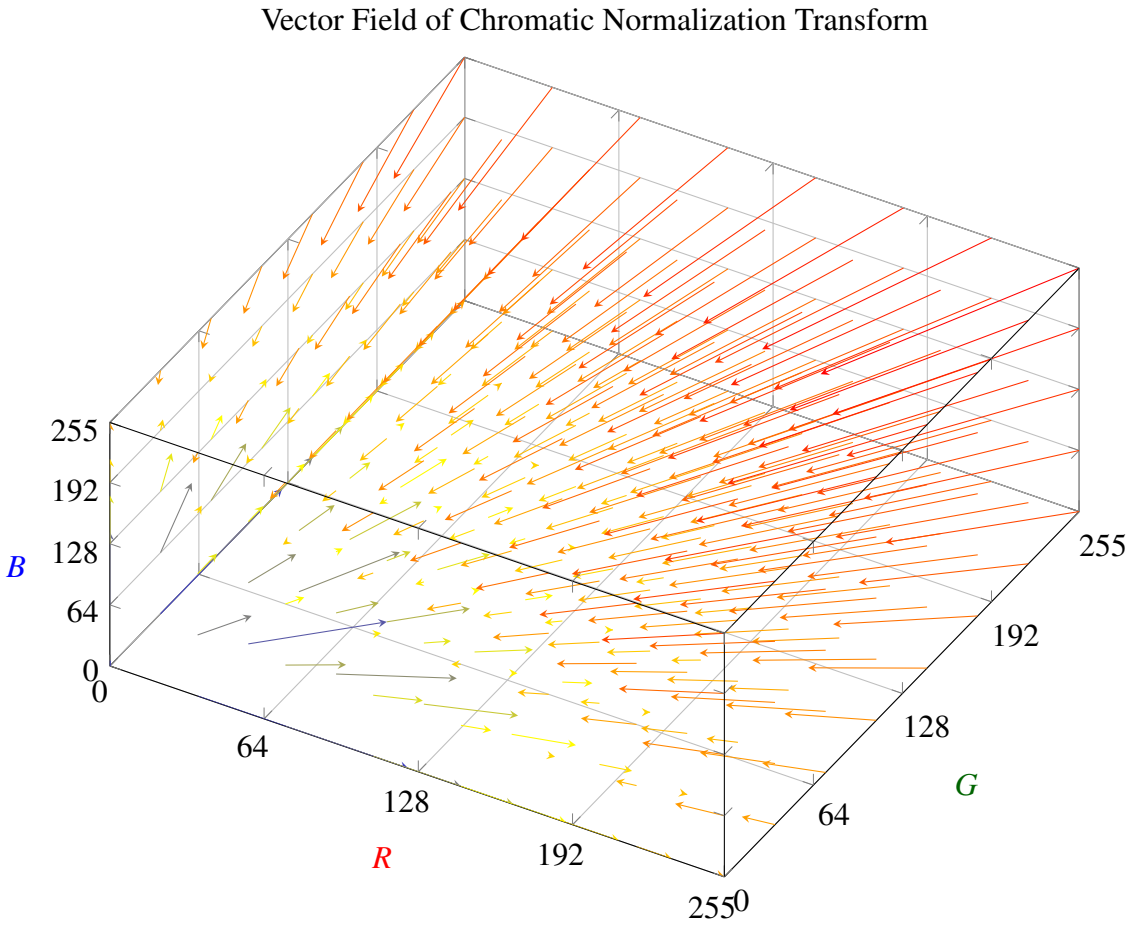


Figure 13: Vector field illustrating the RGB Chromatic Normalization transform.

There are also works on skin color segmentation; some use RGB Normalization Chude-Olisahl *et al.* (2013) to be invariant to light conditions. Moreover, other research works usually have a more controlled light condition Li & Plataniotis (2018) and similarly propose a novel method for describing textures based on color.

It is important to remark that there are many other cases where the computer must be able to segment the image with a lighting invariant classifier. In Upcroft *et al.* (2014) they presented a technique that utilizes neural networks to solve the semantic segmentation. The semantic segmentation is a bit far from what it is proposed in this project, but shows how important is the studies in image segmentation until now. This also includes the biomedical area Li & Plataniotis (2018), where these classics algorithms of segmenting the color space, or creating

descriptors and then use this to segment the images. There are recent works that utilizes neural network to solve soft color segmentation Akimoto *et al.* (2020). The soft color segmentation often involves iterative techniques that can be very slow for some real-world applications. In the robotic competition context we have Saraydaryan *et al.* (2019) that works with k-means clustering algorithm on the HSV color space for color detection, and was aimed to further works with deep learning color naming based on a weakly supervised approach Yu *et al.* (2018). Still in robotic competition context, we have found a work on soccer ball detection using DNN that Szemenyei & Estivill-Castro (2019) also states that it is not yet feaseable for real-time embedded systems, and the lack of datasets for the robotic's vision specific tasks. Moreover, the problem we tackle is a hard color segmentation, where each pixel will have only one label, and we assume that the color information is enough for its labeling.

3.2 PREVIOUS APPROACH

In the early versions of our segmentation software de Sales Júnior *et al.* (2017), we have utilized a Manual YUV Segmentation (Figure 14) as the segmentation method. The main advantage is that it is simple to implement since it verifies if a pixel is inside a 3D Bounding Box of the YUV color space. This advantage, however, brings several limitations:

- some colors could not be correctly segmented due to non-rectangular regions in YUV color space.
- external light changes implied in fine-tuning or resetting the configurations.
- to improve this method means a more complex method, with more parameters.
- it is necessary to reset or reconfigure the camera parameters (brightness, hue, exposure time, and possibly more).

The main disadvantages were not restricted to the robustness, but also to:

- the high amount of time needed to configure and reconfigure the segmentation;
- the high level of expertise needed to master the segmentation process (camera parameters + YUV segmentation parameters, and how they relate to each other);

To completely configure the segmentation with the 3D bounding boxes in YUV, we needed to define 6 parameters for each color to be segmented: Minimum and Maximum values for each limit of the 3D bounding box. Therefore, 8 colors to be segmented means that we have up to 48 parameters to tune. The manual YUV pipeline (Figure 14) is simpler to implement; however, it brings complexity to the parameter tuning process and hard limitations to light changes.

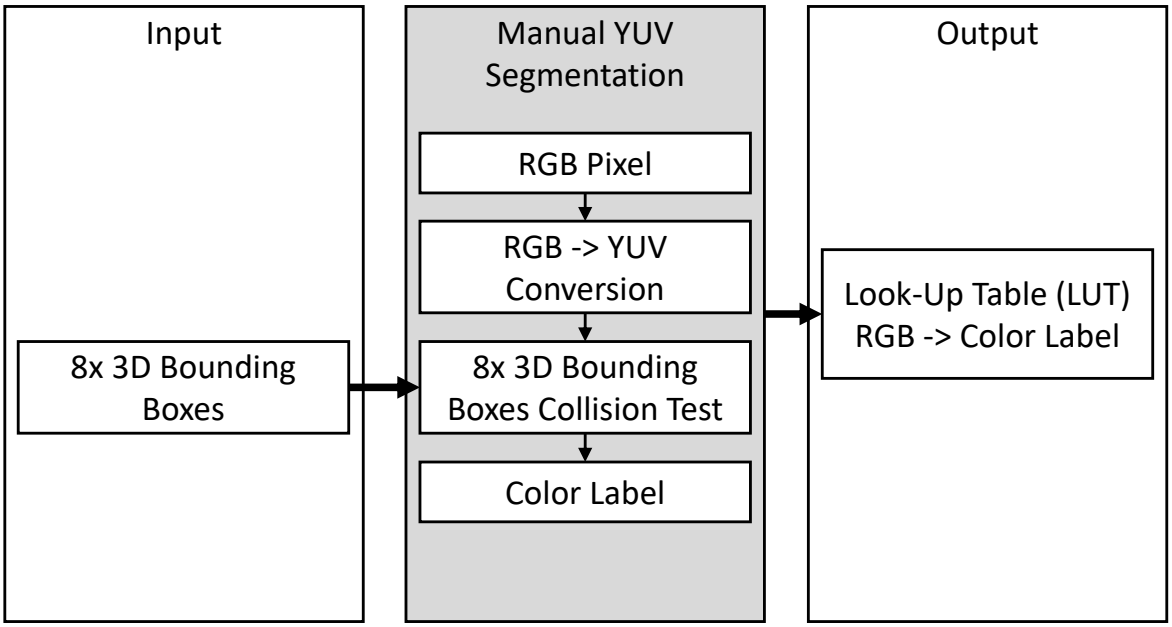


Figure 14: Manual YUV Segmentation pipeline. Every possible combination for red, green, and blue in a pixel is mapped into one color label.

4

PROPOSED APPROACH: MAGGICVISION + MAGGICSEGMENTATION

In this Monography, it is proposed a computer vision system called MaggicVision, that gives a graphic interface to debug and calibrate the segmentation provided by the proposed method called MaggicSegmentation. The LUT generation is usually $10-40ms$ on CPU, and this variation is due to the machine configuration. The combination of both MaggicVision and MaggicSegmentation provides a color segmentation method and a graphic interface that:

- reduce the complexity of the color segmentation process executed by a human.
- reduce the required time to configure the computer vision system.
- has no computational impact on the available vision pipeline when active, since we generate the LUT before the beginning of the match.
- is robust to light changes in intensity and some level of color distortion.

We have been using a Vision Module Pipeline (Figure 15) in which the process is already optimized with the usage of a Look-Up Table (LUT). This LUT accelerates the process of classifying a pixel by mapping a color directly into a label. Then every possible color is mapped to a label. The previous approach was the Manual Segmentation (YUV), in which the user had to deal with the configuration of one bounding box for each color in the YUV color space (Figure 14).

We needed a method that changed the color's mapping function into a label that should be fast to configure, lighting-invariant with some robustness to color deviation. Such segmentation function is presented in Figure 16, which has a more complex flow, with more steps, but this brings simplicity in the input. Therefore, the segmentation's hard work is passed to the computer, giving the user a more intuitive description of the segmentation.

To achieve that, we tried to better understand how we proceeded before. We assumed that we often classify colors due to their tendency to one reference color, or another, based on its similarity when we are not certain of its classification. And this idea is used on a novel **RGB (Weighted) Normalization** technique that considers the original chromaticity and enforces that

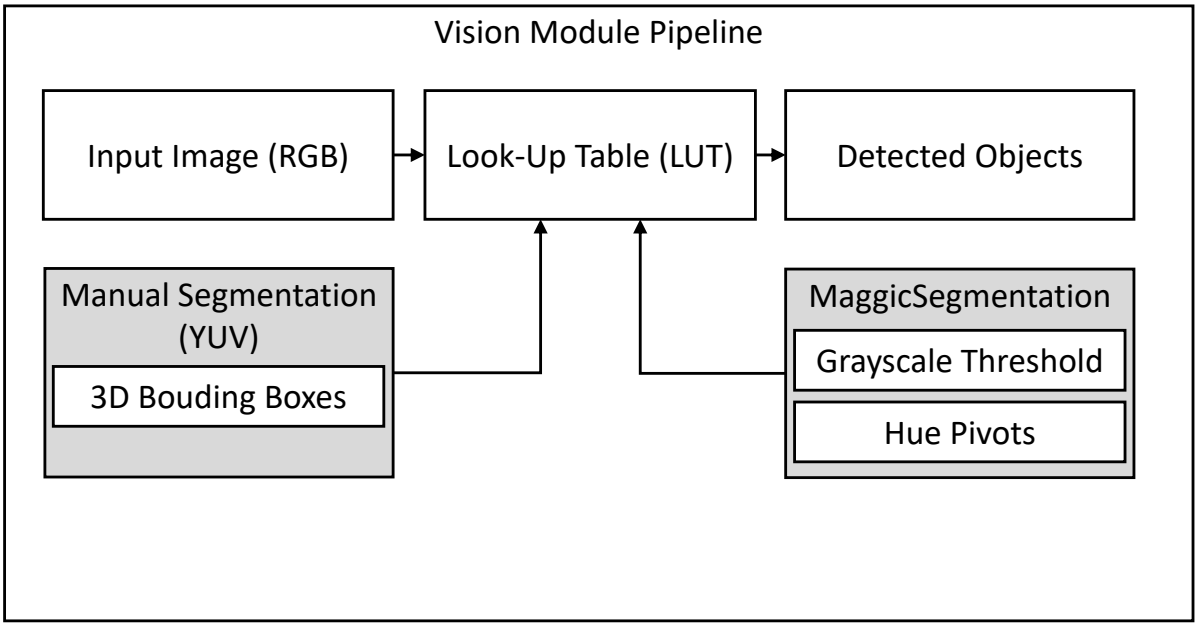


Figure 15: Vision Module pipeline. Every possible combination for red, green, and blue in a pixel is mapped into one color label.

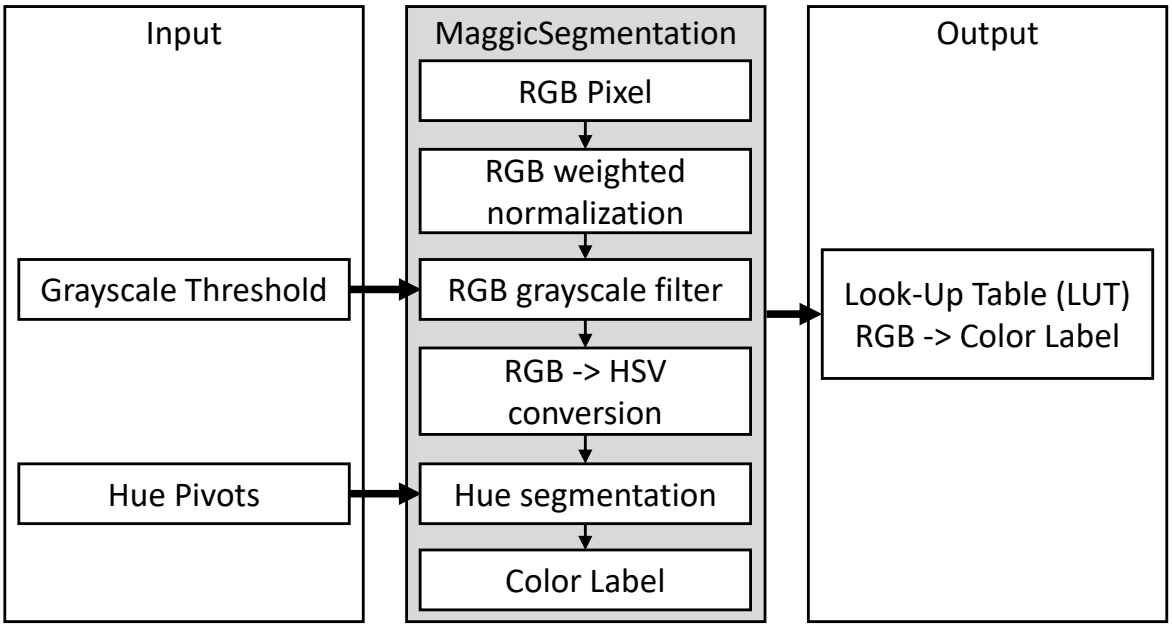


Figure 16: MaggicSegmentation pipeline. Every possible combination for red, green, and blue in a pixel is mapped into one color label.

characteristic, a **RGB Grayscale Filter** to remove grayish colors, and finally **segmenting the filtered colors in the Hue dimension** of HSV color space. This method is implemented with one threshold value for the Grayscale Filter and one Hue Pivot value for each color we want to segment, resulting in the worst case, the need to execute a fine-tuning of up to 8 parameters (7 color values + 1 grayscale threshold). In the average case, we only would need to adjust

one parameter, the grayscale threshold value, since the Hue Pivots (Figure 17) are segmenting non gray color. These non gray colors are defined after eliminating possible colors with low chromatic values due to camera sensor noise and light conditions variation. For the RGB to HSV color space conversion we have utilized the OpenCV's Bradski (2000) *cvtColor*, using the *COLOR_BGR2HSV_FULL*. Note that the BGR order is an implementation detail of utilizing OpenCV's Bradski (2000) data structures.

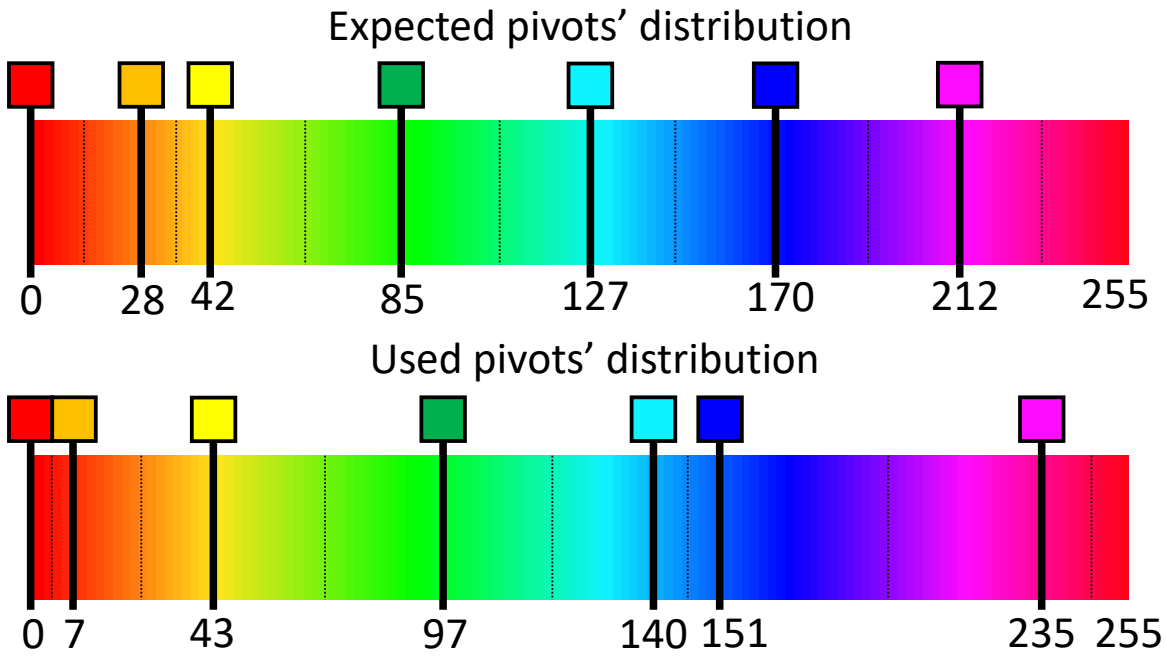


Figure 17: Illustration of Hue Pivots segmenting the Hue channel range.

4.1 RGB NORMALIZATION METHODS

We have utilized a novel kind of color normalization that ensures that reddish colors stay reddish, greenish color stays greenish, and so on, while darker colors are mostly preserved in the process. These darker colors must remain near the same amount of lightness due to numeric precision loss in further color segmentation steps. Furthermore, we turn lighter colors into a darker version of this color, similar to what we expected from an RGB normalization. However, the direction of the transform is changed, taking into account the influence of each channel composing the color: reddish colors must stay reddish, greenish-blue colors must stay greenish-blue. This property gives an essential step to clustering colors with similar component values. After this transformation, similar colors should map to the same or very near positions in the RGB color space as dissimilar colors diverge from each other in the RGB color space. Next, we describe each normalization, and we compare the vector field of each normalization transform to the proposed *Weighted Normalization Transform*. We use a vectorial definition for the triplet of RGB color in Equation 4.1.

$$\vec{v} = [R, G, B] \tag{4.1}$$

4.1.1 RGB Vector Normalization (VNT)

The RGB Vector Normalization is based on algebraic vector normalization. This normalization turns high values to lower values and low values to higher values. Moreover, the transform's strength is inversely proportional to a squared root of the sum of its squared components; the influence of normalization has a lower rate of increase from the intermediate light colors, all pointing to similar colors on the mid-way. The vector magnitude is defined in Equation 4.2, and the RGB color normalization follow the vectorial normalization (Equation 4.3). The complete RGB Vector Normalization is defined in Equation 4.4, where we can directly use values using [0..1] values in each channel. However, if the used range in channels is [0..255] there is the need to recover the scale after normalization (Equation 4.5).

$$|\vec{v}| = \sqrt{R^2 + G^2 + B^2} \tag{4.2}$$

$$\hat{v} = \frac{\vec{v}}{|\vec{v}|} \tag{4.3}$$

$$\hat{v} = \left[\frac{R}{|\vec{v}|} \quad \frac{G}{|\vec{v}|} \quad \frac{B}{|\vec{v}|} \right] \tag{4.4}$$

$$VNT(\vec{v}) = 255 \cdot \hat{v} \tag{4.5}$$

To visualize how the vector normalization transform behaves we utilize a 3D vector field

illustration (Figure 18). In this visualization is possible to verify that the values are normalized to higher values (compared to the RGB Chromatic Normalization) over a quarter of a circle from the origin. This normalization has the behavior of maintaining gray tones in the main diagonal, which is a desirable behavior, however, the colors that are not gray but are close should be put further from the diagonal to restore the chromaticity values. The chromaticity recovery is a desirable feature for the normalization process due to intense light influence, or low exposure we have to deal in the real scenario of the competition.

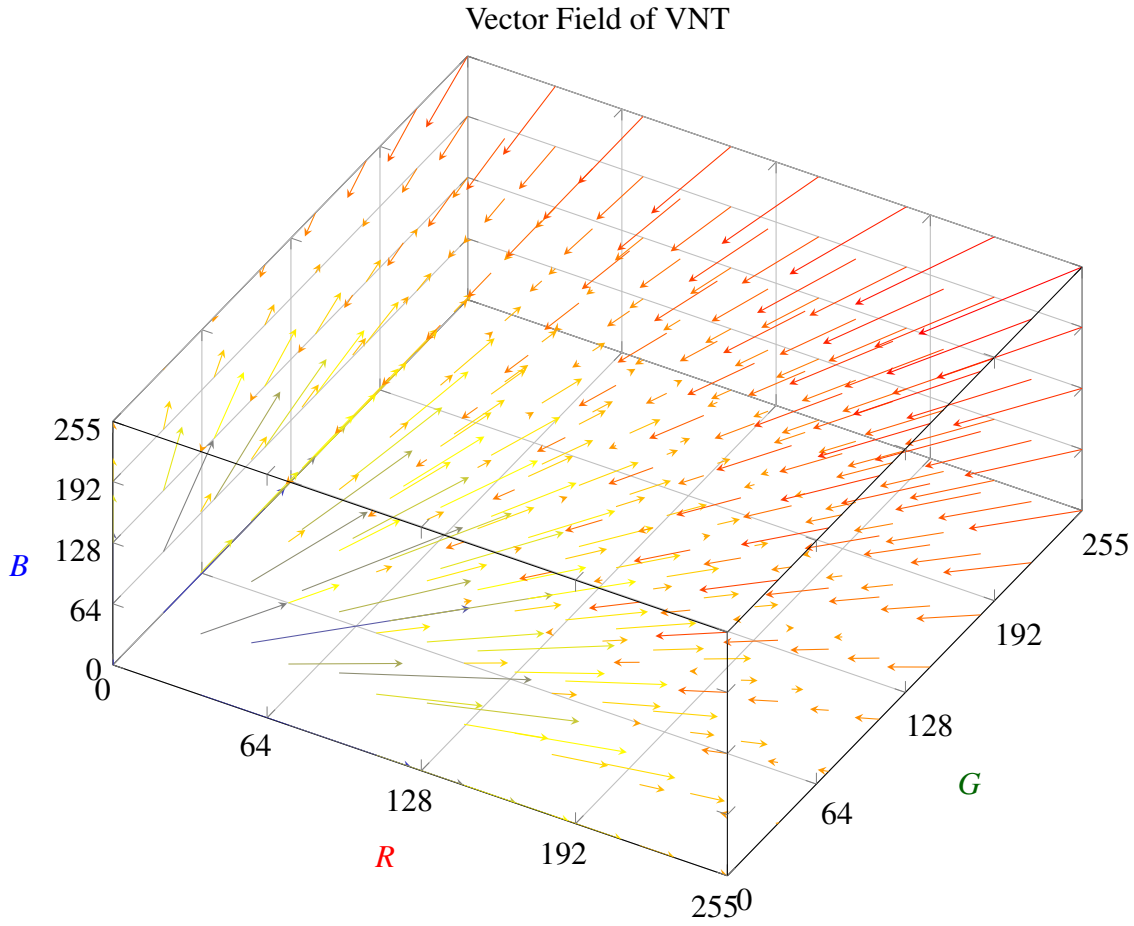


Figure 18: Vector field illustrating the RGB Vector Normalization transform.

4.1.2 RGB Weighted Normalization (WNT)

In the proposed approach, the RGB Weighted Normalization Equation 4.6, high values tend to get lower values, and low values tend to stay more or less where they are, except for the direction. All colors with higher red values tend to be more reddish, and similarly, this also happens to greener and bluer colors. For example, a color described by $(100, 99, 99)$ would become $(58, 56, 56)$, which is more red than before. This can be measured by how big the biggest component is compared to the smallest component. For example, the $(100, 99, 99)$ color have a 1% of difference between the bigger value and the smallest value, and when we apply the WNT

we achieve 3% of difference between the bigger value and the smallest value (the $(58,56,56)$ resulting color). When compared with VNT, the VNT for the same sample achieve 1.4% of difference, and the CNT achieve 1.01%, being the worst result between the three normalization methods. Thus, we try to restore some chromaticity even from gray-like colors as we presented as an example. This follows the hypotesis that we try to decide if a color tends to another color by comparing it to other well defined colors. In our approach we assume that we would compare for red, green, and blue, since these colors already have captured their intensities in each channel. This method generates a transformation, and its behavior can be seen in a vector field (Figure 19) that we can relate to what we expected for an chromaticity recovery. The chromaticity recovery for our scenario means that if we have a higher value in the red channel than the other two channels, we should push the color to the direction of the red vector accordingly to its relative magnitude. This also means that a yellow color $(200,200,30)$ would be equally pushed to both red and green axis, resulting in $(140,140,3)$ for WNT (the bigger value is 46 times bigger than the smallest value), $(179,179,26)$ for VNT (6.88 times), $(118,118,17)$ for CNT (6.66 times). Therefore, we can see that the chromaticity is in this case potentialized with the Weighted Normalization, and this is true for colors that are not gray.

$$WNT(\vec{v}) = \left[\frac{R^2}{|\vec{v}|} \quad \frac{G^2}{|\vec{v}|} \quad \frac{B^2}{|\vec{v}|} \right] \tag{4.6}$$

To visualize how the weighted normalization transform behaves we utilize a 3D vector field illustration (Figure 19). In this visualization is possible to verify that the values are normalized to lower values (compared to the RGB Chromatic Normalization and RGB Vector Normalization). This normalization has the behavior of maintaining gray tones in the main diagonal, which is a desirable behavior. And the main expected behavior of enhancing the chromaticity can be seen as the vector field shows the tendency to push colors that are close to the three main axis (R,G,B) to the corresponding weighted axis. This means that if a color has a combination of one or more axis with higher values than the complementary component(s), this normalization will push the color closer to its more representative components proportionally.

To compare the three normalization methods, we created the Figure 20. In this figure we show equally spaced slices from the Chromatic Normalization Transform (CNT) resulting vector field in the first column, the Vector Normalization Transform (VNT) resulting vector filed in the second column, and the proposed Weighted Normalization Transform (WNT) resulting vector field in the third column. We can see the desired behavior of maintaining the gray colors in the main diagonal on all of the three normalization methods (Figure 20). However, they differ not only on scaling the original colors, but their change of vector directions. For example, in the first column (CNT), we can see that the vectors are scaled mostly in the same directions from the origin to the color position, often achieve a closer position to in the middle (first column, bottom slice), or in the origin corner (first column, top slice). Moreover, in the second column (VNT), we can see that the vectors with small components are scaled up to higher component values,

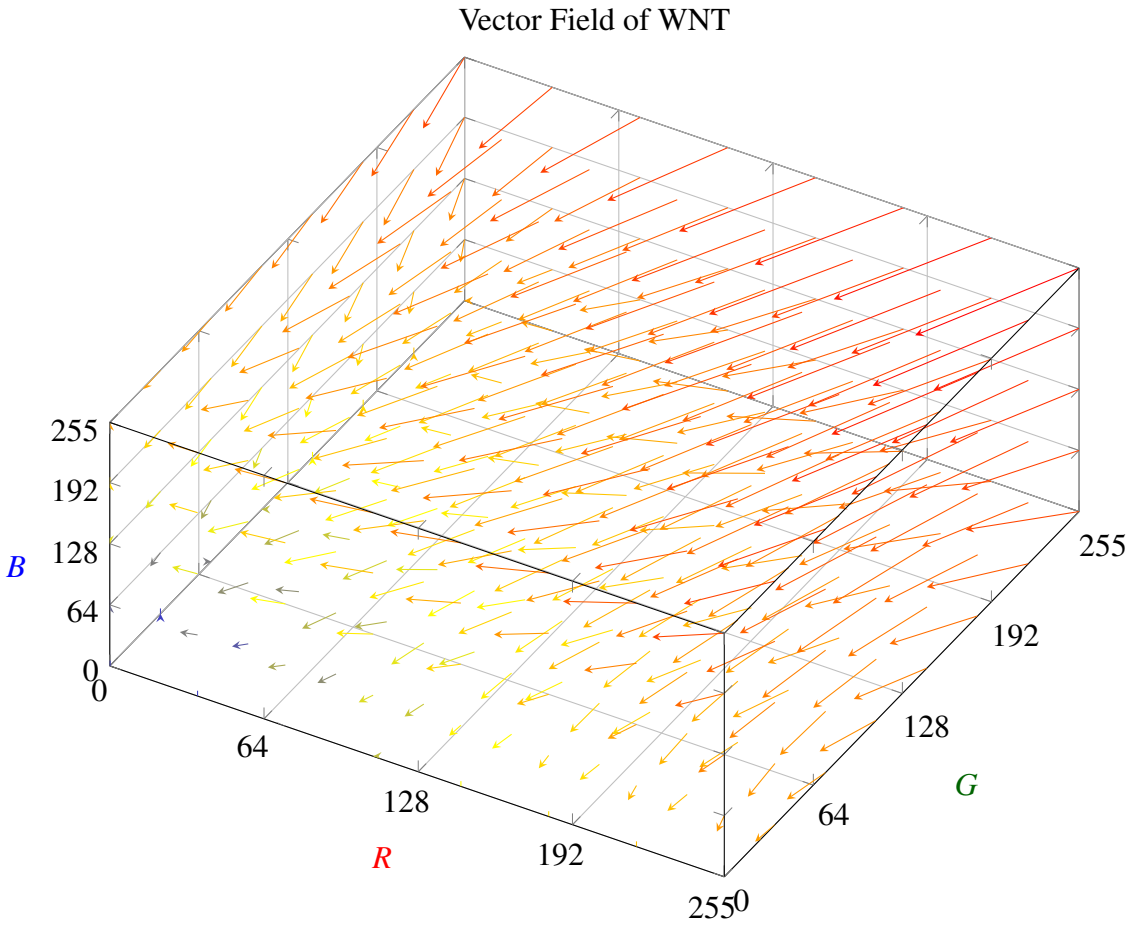


Figure 19: Vector field illustrating the RGB Weighted Normalization transform.

and this time the bottom slice shows us a quarter circle region of convergency from the origin for small and high component values. Also, a stronger influence from red and green colors when the blue value has zero value (second column, bottom slice), and a moderate influence from red and green when the blue value has the maximum value (second column, top slice). Still in Figure 20, in the third column (WNT), we can see that the vectors with small components are torn appart from the main diagonal (in the case of not being a gray color), and they are pushed to a direction proportional to the strongest's components. Note that if a color has a high green value, but lacks on red and blue, it will be pushed for the green axis as shown by the vector field near the G axis (third column, bottom slice). Also, if a color has high green and blue components, we can see that it will be pushed to the green axis and to the origin (third column, top slice).

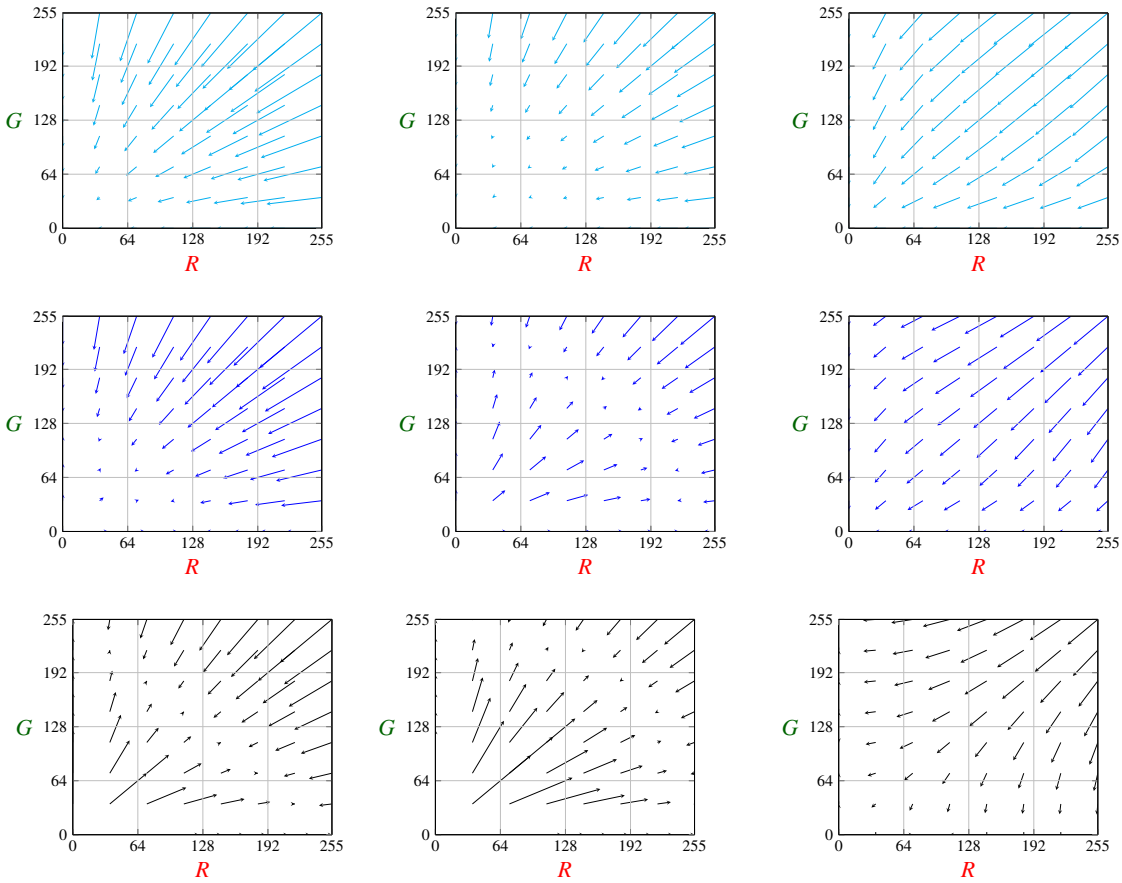


Figure 20: $R \times G$ slices for $B = \{0, 128, 255\}$ (from bottom to top) in black, dark blue and light blue respectively. First column: Chromatic Normalization Transform. Second column: Vector Normalization Transform. Third Column: Weighted Normalization Transform.

4.2 RGB GRAYSCALE FILTER

We have utilized an RGB Grayscale Filter that enabled us to filter colors in the range of a cylinder from black to white (the main diagonal of the RGB color space, usually represented as a cube). This color space has an important contribution to the removal of grayscale colors. Since all gray colors have the same values in all channels (Red, Green, Blue), they share the property of being at the RGB color space's main diagonal. This concept is shown in Figure 21. It is essential to notice that we only apply the RGB Grayscale Filter *after* the RGB Weighted Normalization. In this way, we consider the distortion that the transform executes on the RGB color space and utilize its advantages.

Additionally, we are interested in the removal of colors with low chromaticity (grayish colors), and for this, we use an approximation of the distance function. Due to optimization, we chose the *min* and *max* functions. We get the minimum value of the three channels and then utilize this value as the projected gray value in the diagonal. Whereas all gray colors have $r = g = b$ we can check the distance of a color (r_1, g_1, b_1) with a boundary check from the selected gray value. For example, if the color is $(54, 54, 8)$, we have a dark yellow color, and we

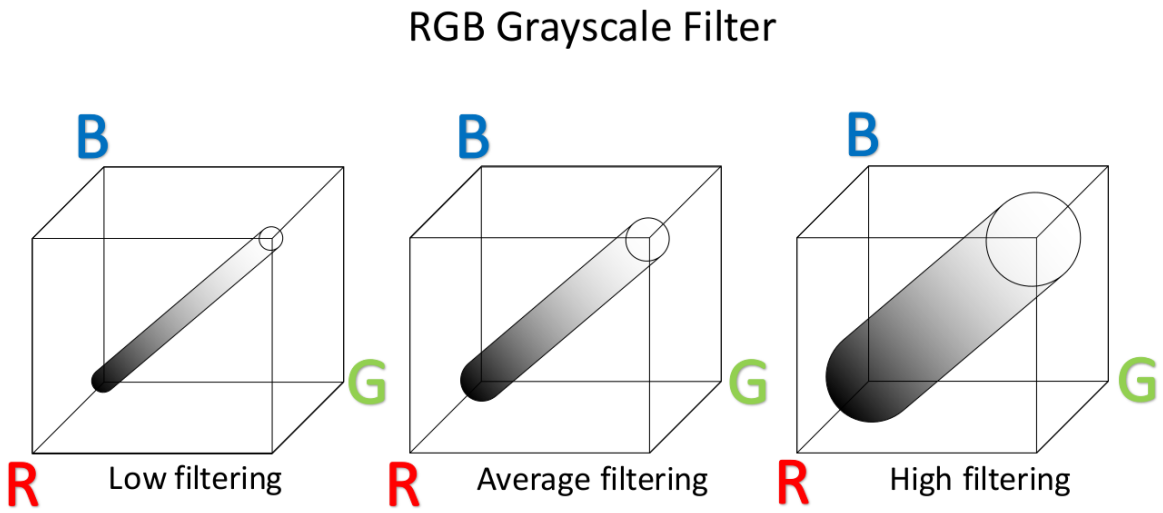
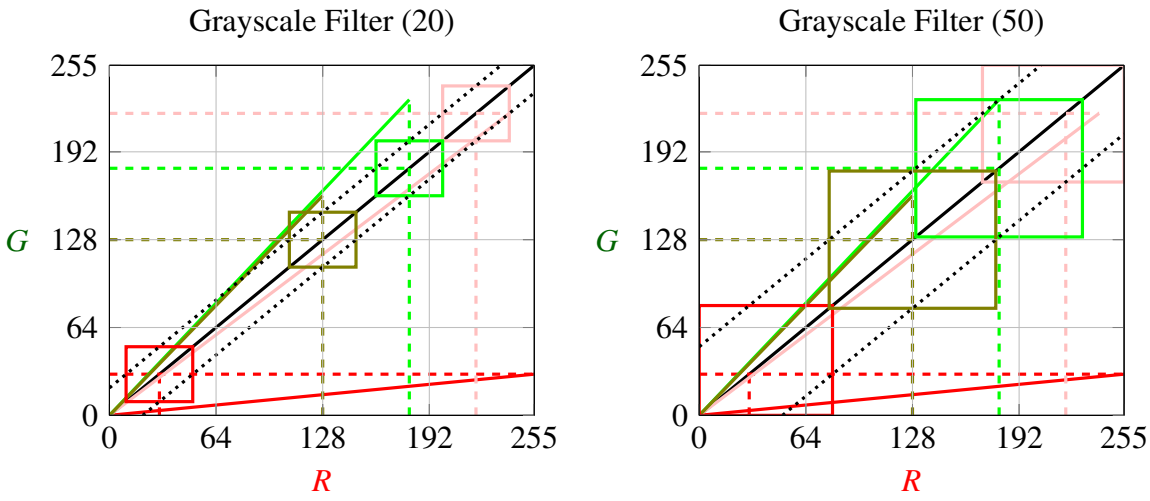


Figure 21: The RGB Grayscale Cylinder concept in RGB color space.

could verify its distance to aproximated closest grayscale $(8,8,8)$. The process is demonstrated in Figure 22 for two threshold values 20 and 50. Suppose we have a color (r,g,b), and it is a gray tone, then the channels should have equal values. And since all the channels are equal, the minimum value of the three channels will be equal to each channel value; this means that the color is in the main diagonal of the RGB color space. The approximation is done by checking if the color is inside a three dimensional box with $(2t+1)x(2t+1)x(2t+1)$ dimensions, centered on the aproximated gray color on the main diagonal, where t is the *grayscale threshold*. We chose this approximation because checking if the (r,g,b) color is inside a box is faster than checking if it is inside the main diagonal cylinder. We present a 3D visualization of the filter (solid line of the gray diagonal, and two dotted lines representing a range example), with some colors (red, green, rose, low saturated green) represented in solid lines, and its components with a dotted lines in Figure 23.



(a) Threshold = 20.

(b) Threshold = 50.

Figure 22: RGB Grayscale Filter Tactics.

4.3 HUE SEGMENTATION

After the RGB Weighted Normalization, Grayscale Filter, the HSV color space was chosen to segment the desired colors. We extend the concept of pivots in way to generate the labeling regions, and propose this method as a better alternative to the user defined ranges for each labeling regions. As seen in Figure 24, we have one Hue Pivot for each color we want to segment. We expected that the expected pivots' distribution was a direct way of segmenting colors by its hue value, since the more close the hue value, the more similar are the colors. However, the expected pivots' distribution did not result in an robust segmentation, this was due to gradient borders that were not defined accordingly.

In Figure 24, each pivot position represent a target color, and the dotted lines define borders between different labeling regions. To segment the hue component we use a classification by distance to the closest hue pivot, assuming that the hue value will be classified with the label of the closest pivot. It is important to remember that the Hue component is circular in the HSV color space, and it is an *angular* component. Then, we have to deal with the regions being able to be circularly defined by connecting the minimum and maximum values as a necklace enclosing would have.

Additionally, we use the *used pivots' distribution* of Hue Pivots (can also be seen in Figure 24); the used pivots' distribution of the Hue Pivots have the most robust configuration we achieved after many trials in laboratory and in competition. The Hue Pivots where thought to define regions for different labels based on proximity to the pivot hue. However, the important aspect of this segmentation method is that we should care more about the border between labeling regions. In fact, in the used pivots' distribution, the dotted lines representing the limits of the labeling regions are mostly in the frontier of a region of ambiguity, while the in the expected pivot's distribution does not have this characteristic. The used pivots' distribution values can be

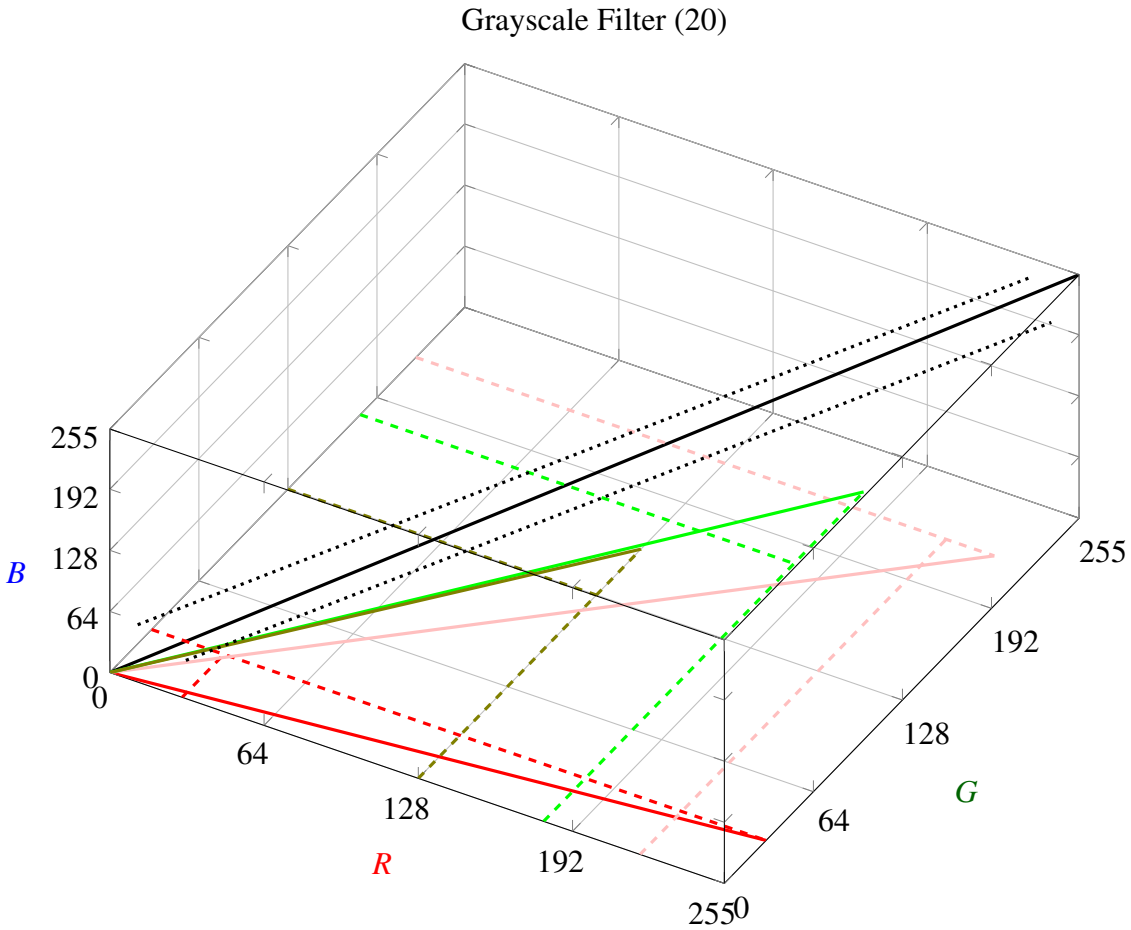


Figure 23: RGB Grayscale Filter 3D space.

seen in Table 1.

Channel	**Red**	**Orange**	**Yellow**	**Green**	**Cyan**	**Blue**	**Magenta**
Hue	0	7	43	97	140	151	235

Table 1: The used pivot distribution values.

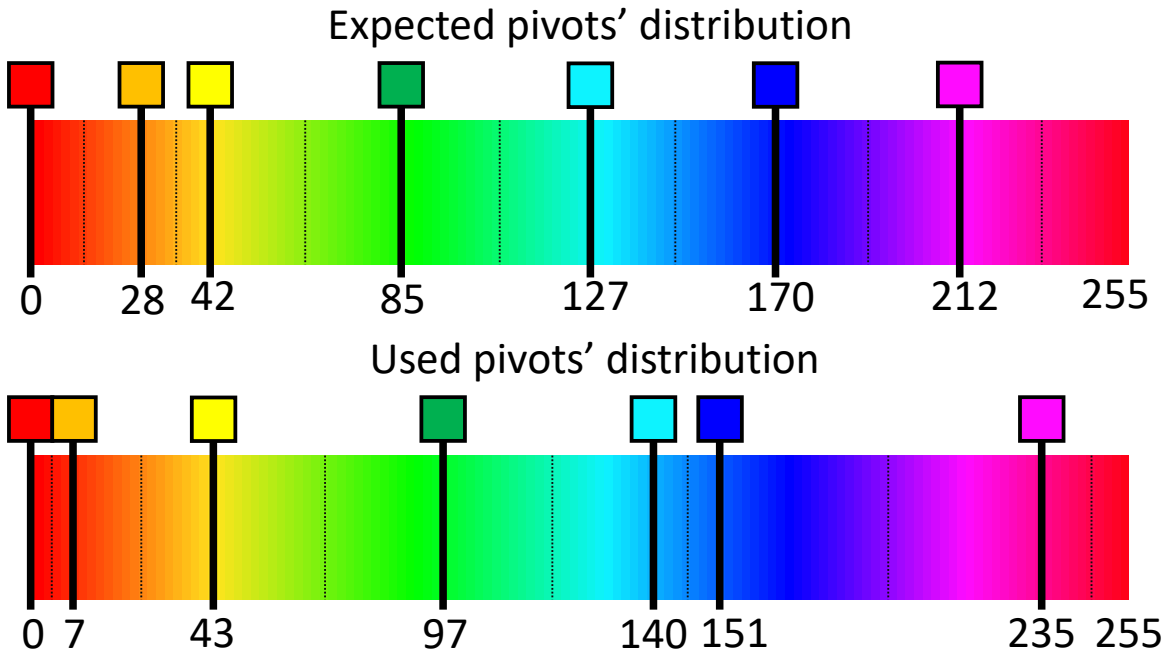
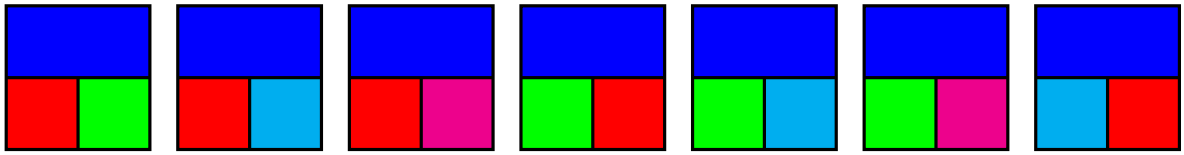


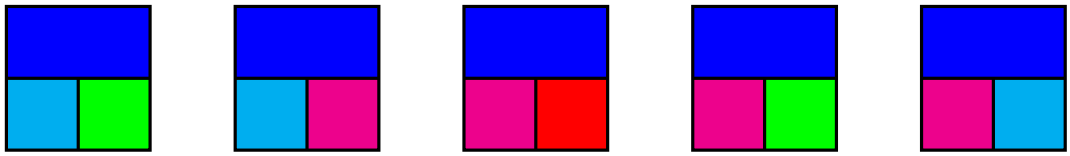
Figure 24: Illustration of Hue Pivots segmenting the Hue channel range.

4.4 EXTENDED TAG PATTERN

It is possible to propose an extended tag pattern for the competition (Figure 25 for the Blue Team, and Figure 26 for the Yellow Team), allowing to have more than 11 players (more than the real soccer game with humans). Each team could utilize the seven color approach for segmentation, and with a proper blob-clustering algorithm, it would be able to have more players in-game and take full advantage of the robustness of those seven color segmentation within the MaggicSegmentation approach. We have done some experiments for the VSSS' experimental division: 5*v*5. Some teams try to utilize other colors, which makes segmentation harder on the usual segmentation approaches (3D ranging in different color spaces) or different visual patterns to address the segmentation problem. Figure 27 presents some of our tests on the detection of the robots for the VSSS' 5*v*5 division, with a challenge of detecting the main pattern (with two tags) and the extended pattern (with three tags). The results can be seen in Chapter 6.

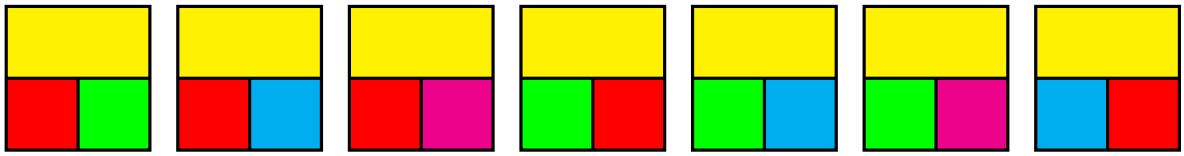


(a) Red-Green (b) Red-Cyan (c) Red-Pink (d) Green-Red (e) Green-Cyan (f) Green-Pink (g) Cyan-Red

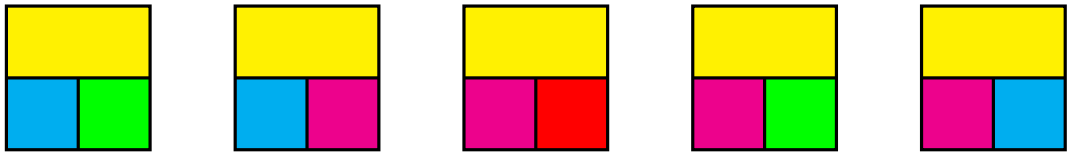


(h) Cyan-Green (i) Cyan-Pink (j) Pink-Red (k) Pink-Green (l) Pink-Cyan

Figure 25: blue team color extended pattern.



(a) Red-Green (b) Red-Cyan (c) Red-Pink (d) Green-Red (e) Green-Cyan (f) Green-Pink (g) Cyan-Red



(h) Cyan-Green (i) Cyan-Pink (j) Pink-Red (k) Pink-Green (l) Pink-Cyan

Figure 26: Yellow team color extended pattern.

Figure 27: 5*v*5 test with both two-tag and three-tag patterns.

5

DATASET GENERATION FOR VALIDATION

This chapter discusses the proposed dataset generation to quantify how well the segmentation would compare to other variants. The dataset has been generated from real-world and synthetic images. The real-world images were extracted from captures that happened in the competition or laboratory tests before the pandemic. The synthetic images were generated within the Blender Community (2018) software, using advanced shading pipelines and the Ray-Tracing Glassner (1991) technique to achieve realistic lighting rendering results. With the A.I. accelerated denoiser available in OptiX library Parker *et al.* (2010).

5.1 REAL IMAGE DATASET

The real-world image dataset was built from competition video captures and manually segmented using software to map the colors. We assumed some arbitrary color values to segment the image fast, making it easier to verify if it is segmented correctly. One of the reasons we chose to create a real-world image dataset only from previously recorded videos is the pandemic limitation policies to the laboratories in the university. However, we have found some difficulties in trying to create the ideal conditions in real-life experiments. These difficulties are not exclusively due to pandemic policies, but the acquirement of resources and the challenge of building a particular closed space for controlled light conditions experiments. Then, we only classify the Real Image Dataset into two categories: Normal and Challenging. There are some examples below of this dataset:

5.2 SYNTHETIC IMAGE DATASET

One of the reasons why we chose to create a synthetic image dataset for this is that we could not create an Ideal real-life dataset. We envision that we can classify the lighting conditions in three main categories: Ideal, Normal, and Challenging.

- **Ideal lighting condition**: Can be defined as a scenario with uniform lighting, with white light sources, completely avoiding a color bias.

- **Normal lighting condition**: Can be defined as a scenario with mostly uniform lighting (even if biased), with blueish light sources and Sun-like light sources (yellowish, reddish), having a small color bias in certain regions and bigger in other regions.
- **Challenging lighting condition**: This can be defined as a scenario with non-uniform lighting, with colorized light sources and Sun-like light sources (yellowish, reddish), having a heterogeneous color bias throughout the field area. Alternatively, even not well-mapped colors in our solution, such as brown and purple, due to its lighting ambiguity with orange and pink.

To generate this synthetic image dataset we chose Blender. The Blender software Community (2018) is a powerful tool for modeling, animating, and rendering 3D content. Also, it is available for virtually any operating system with diverse hardware capabilities. One of the software-hardware capabilities that we are interested in is the GPU accelerated Ray Tracing with Cycles Renderer Valenza (2015), which is available on OpenCL Stone *et al.* (2010), and OptiX Parker *et al.* (2010) implementations.

It has been used reference images for the ball, robot, and fieldmaterials to understand better how the light interacts and how they are textured. Some of these reference images can be seen in Figure 28.

5.2.1 Construction of Scenes

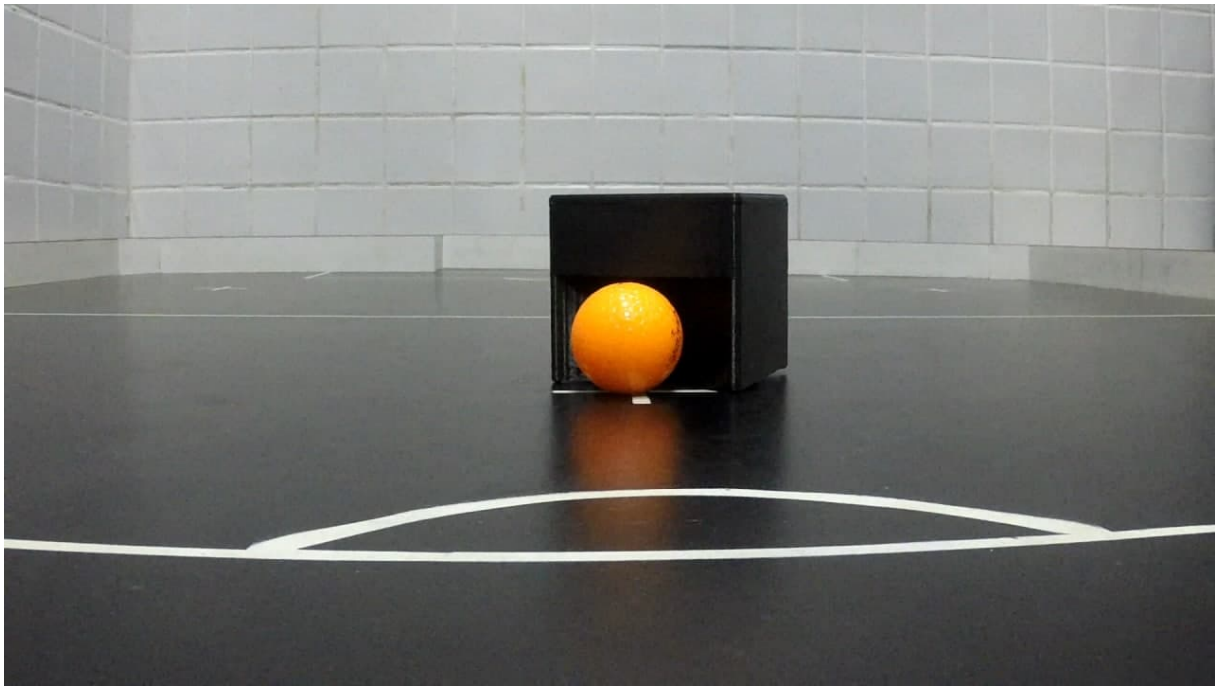
The scene is built so that if one targets an indoor scenario and wants to test if the sunlight would affect the resulting image, it would be necessary just to activate the Sun inside the Collections and Objects outliner (Figure 29). And the scene composition can be seen in Figure 30.

5.2.1.1 Tag Model

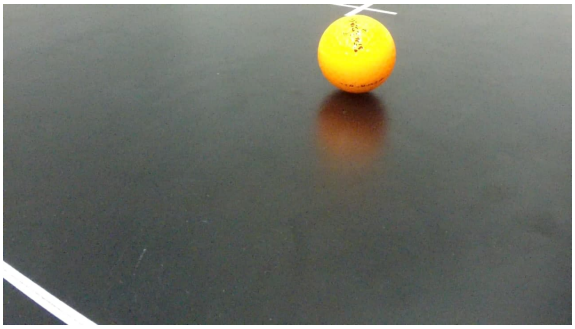
The tag was modeled by a limited plane primitive. Moreover, we have utilized a Glossy BSDF (bidirectional scattering distribution function) Hughes *et al.* (2014) material shading as the initial module for diffuse properties. Further details of the material in Subsection 5.2.2.5 and following subsection.

5.2.1.2 Robot Model

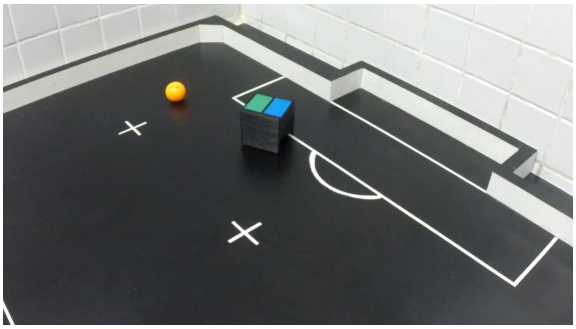
A cube with the folowing dimensions modeled the robot: 7.5 *cm x* 7.5 *cm x* 7.5 *cm*. These dimensions are defined in the competition rules Pinto (2020). We attached the tags over the cube evenly distributed and without intersections over the robot's cube by parenting the cube object over the tags.



(a) Ball and Robot from a close-up position



(b) Ball from a close-up position



(c) Ball and Robot with tags from a upper-view

Figure 28: Reference images to generate synthetic models and materials.

5.2.2 Construction of Materials

The materials were carefully created based on some reference images from captures. Moreover, some properties such as texture were imitated by a noise generator shading component on the shading tool inside Blender. Most surface materials were created based on the Glossy BSDF and Principled BSDF material shader. Some of them were modified to deal with different textures in certain regions or specific texturing, such as the Field Material in Subsection 5.2.2.2. The Glossy BSDF can describe a range of surface roughness, which gives us a sense of metallic or plastic look caused by the level of reflection sampling in each ray collision. To exemplify this we use Figure 32.

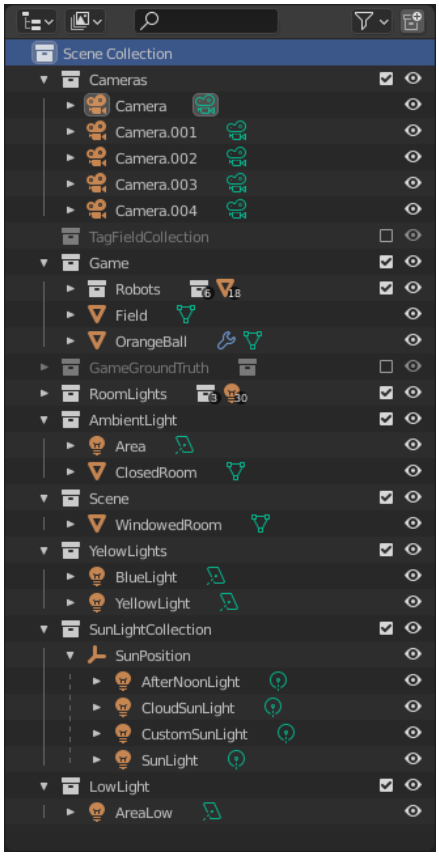


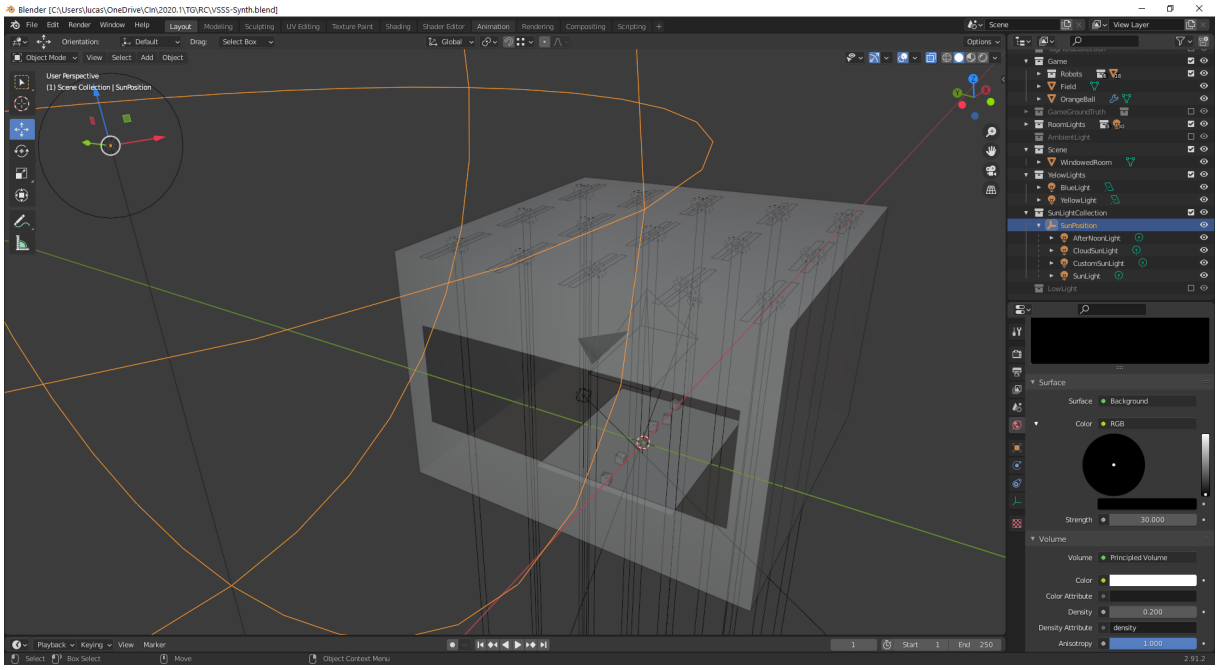
Figure 29: Scene Outliner with Game Collection containing the robots and field, and Lighting configurations.

5.2.2.1 Ball Material

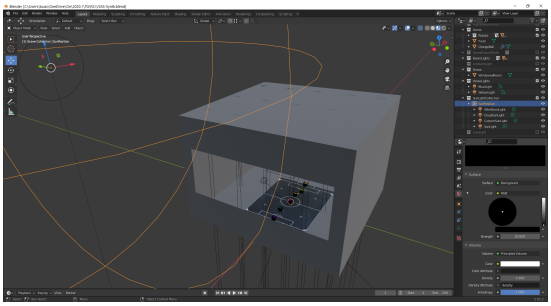
It was chosen the Principled BSDF material as the initial shading module and added a noise texture generator module to the base color property. As a sample of tag materials shader pipelines, the BallOrange shader in Figure 33 represents the ball material shader. The color-specific values are available for each color on Table 2.

Property Name	**Property Value**
Distribution	GGX (Default)
Subsurface Method	Christensen-Burley (Default)
Subsurface	0.170
Metallic	0.0
Specular	0.5
Specular Tint	0.0
Roughness	- -

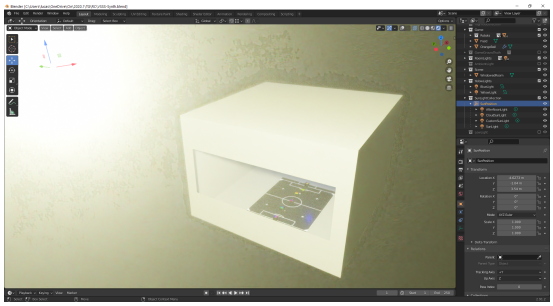
Table 2: Ball Material Properties Values



(a) Scene with active gizmos of all available objects



(b) Textured physically based visualization



(c) Ray traced visualization of the scene with the sun light active

Figure 30: Three ways of visualizing the scene's elements and previewing render result.

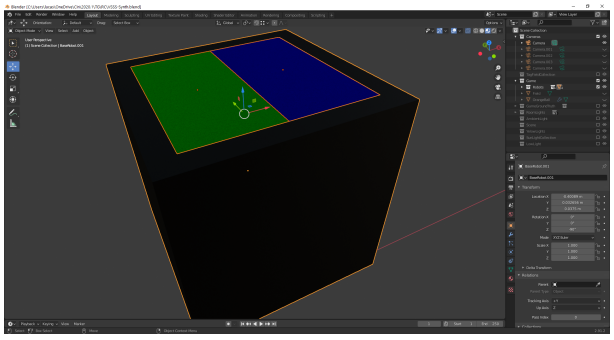
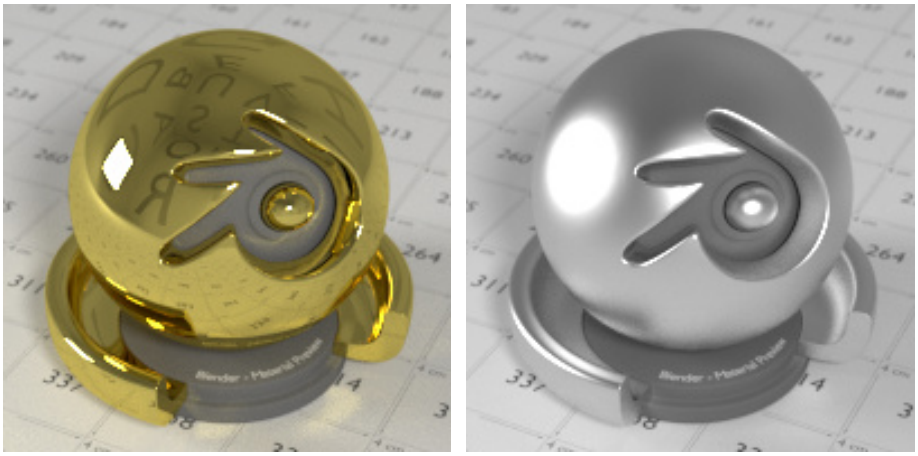


Figure 31: Simple VSSS robot model in Blender.

5.2.2.2 *Field Material*

It was chosen the Principled BSDF material as the initial shading module and added a noise texture generator module to the base color property, connected with the field texture (Figure 34) as a mask for the black noised regions of the field as seen in Figure 35. Since we will



(a) Polished-like surface example (b) Rough surface example

Figure 32: Glossy Material. (By Team (2021))

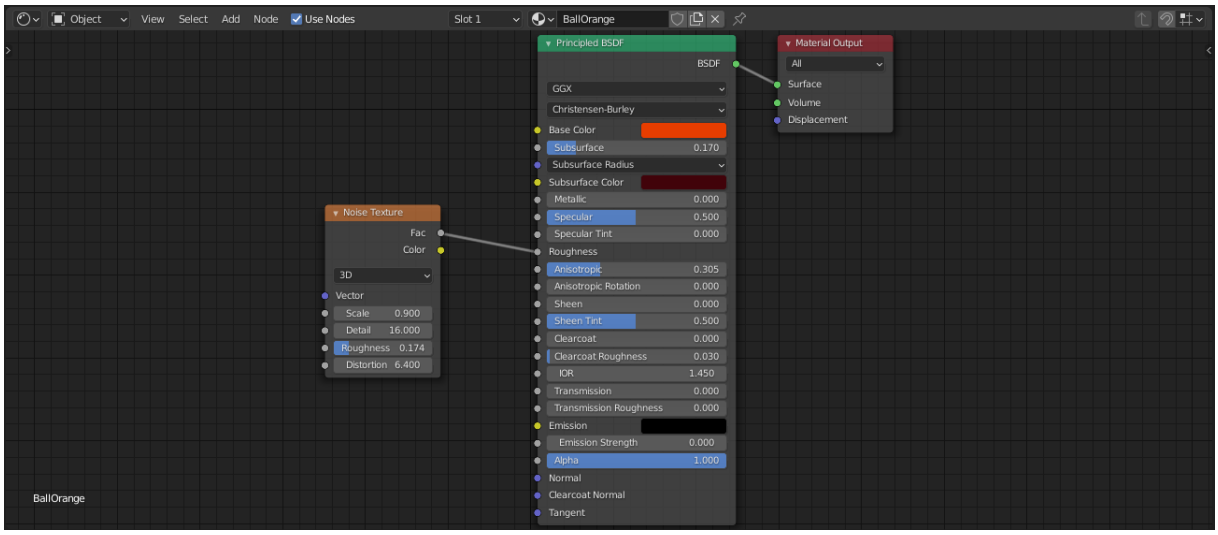


Figure 33: BallOrange shader in Shader editor.

not simulate robot movement and collision with the field walls, we can simplify our model to a plane with the proper dimensions of $1.7m \; x \; 1.3m$.

5.2.2.3 *Room Material*

It was chosen the Glossy BSDF material as the initial shading module for the room material (see Table 3 for details). The room we want should imitate a white painted room (closer to the real-world and usual scenario), and the white color would not interfere with the dominant color.

5.2.2.4 *Robot Material*

The Glossy BSDF material was chosen as the initial shading module and added a noise texture generator module to the normal and clearcoat properties. The robot material shader is

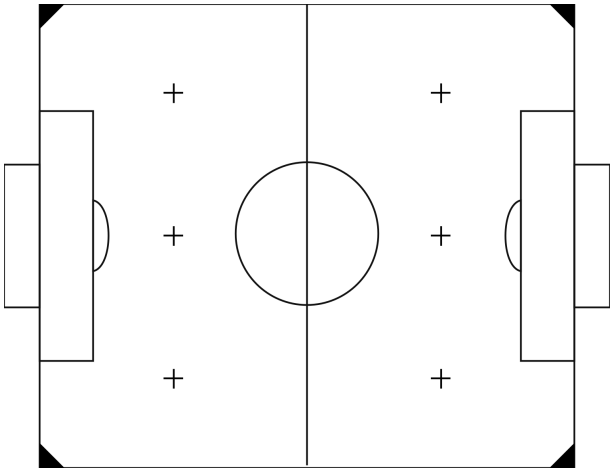


Figure 34: Field texture. (Source: Asset from our VSS-Vision software)

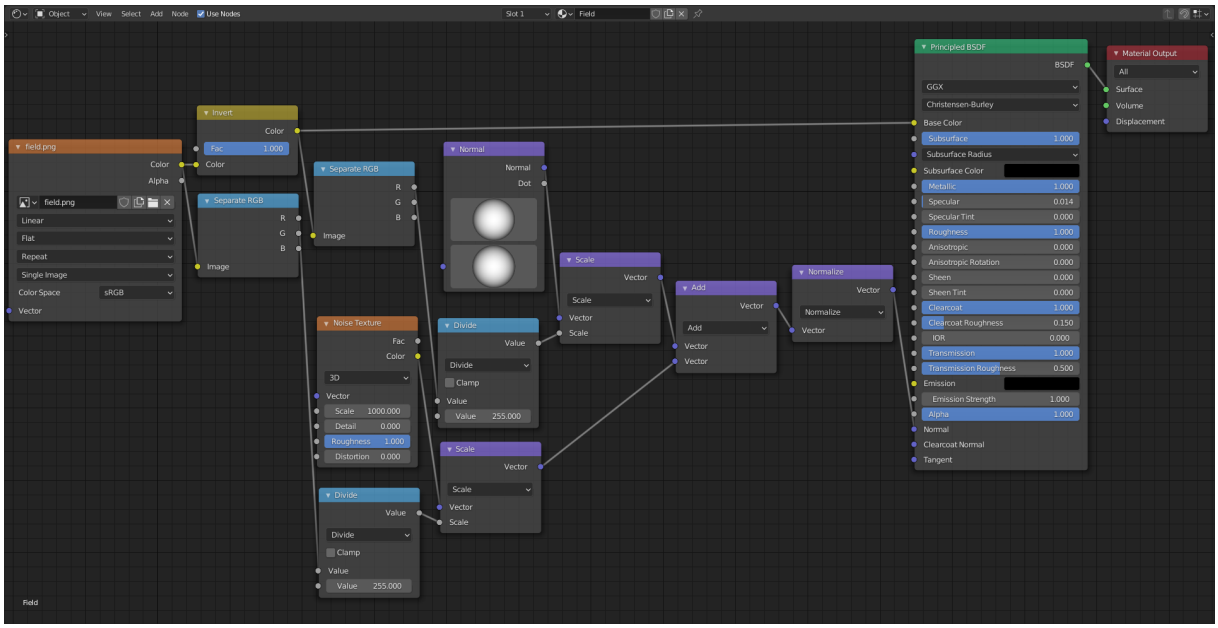


Figure 35: Field Shader pipeline.

shown in Figure 36.

5.2.2.5 *Tag Materials*

The Glossy BSDF material was chosen as the initial shading module and added a noise texture generator module to the base color property. As a sample of tag materials shader pipelines, the RedTag shader (Figure 37) represents all the tag materials shaders. The color-specific values are available for each color on Table 4. The simplicity of the Glossy BSDF inputs gives us easy control over an object's color and roughness.

Property Name	Property Value
Distribution	GGX (Default)
Color.Red	0.170
Color.Green	0.0
Color.Blue	0.5
Roughness	1.0
Normal	Default

Table 3: Room Material Properties Values

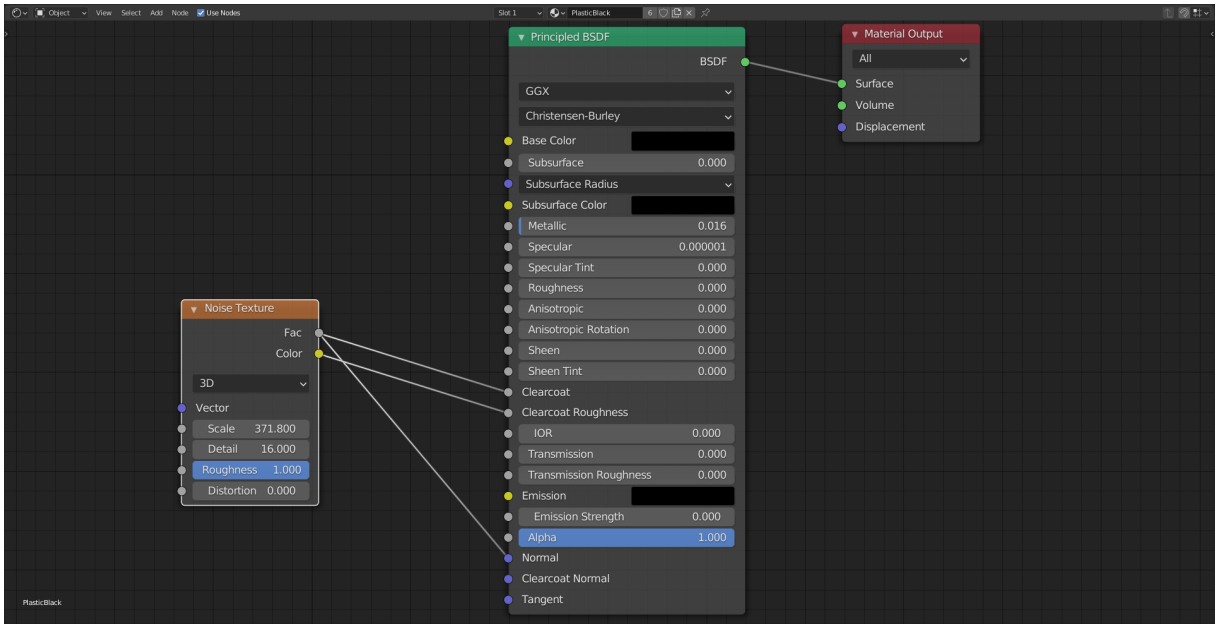


Figure 36: Robot material shader pipeline.

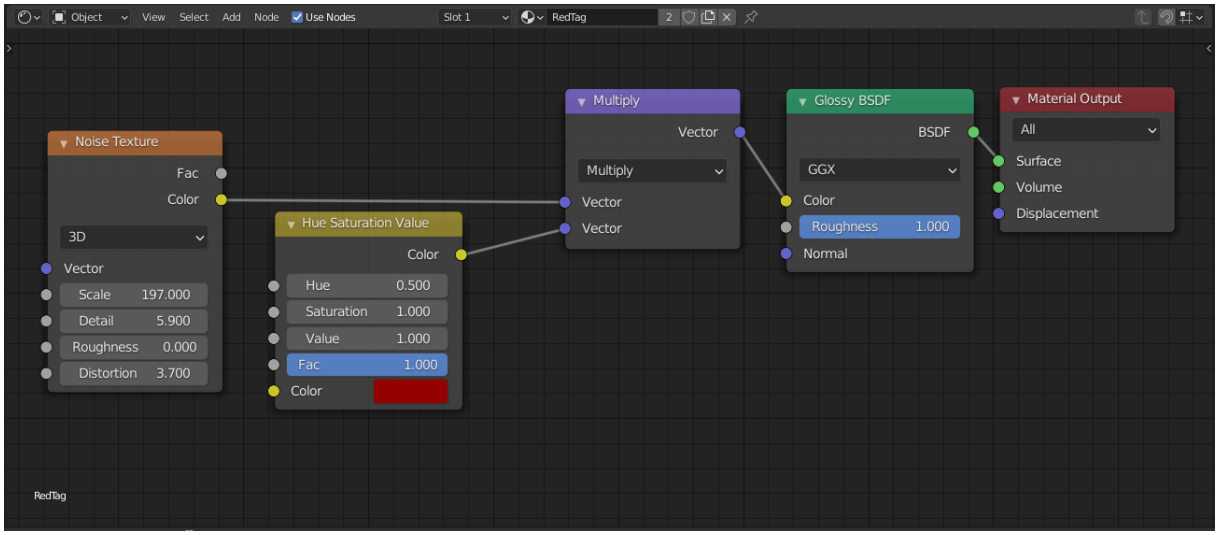


Figure 37: RedTag shader in Shader editor.

Property Name	Red	Green	Blue	Yellow	Magenta	Orange	Cyan
Distribution	GGX (Default)	=	=	=	=	=	=
Color.Red	0.3	0.5	0.0	1.0	1.0	1.0	0.0
Color.Green	0.0	1.0	0.0	1.0	0.0	0.5	1.0
Color.Blue	0.0	1.0	0.3	0.0	1.0	0.0	1.0
Roughness	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Normal	Default	=	=	=	=	=	=

Table 4: Tag materials properties values for each tag color

5.2.3 Virtual Camera Modeling

In the real-world scenario, we would need to execute the lens correction to remove the effects of lens distortion from the image. However, we do not need to implement such a thing to get realistic input to our scenario. Our problem is on the pixel classification, and this does not matter its position on the image. Thus, we do not need to solve or replicate artifacts from lenses, such as chromatic aberration, lens-flare, or diffraction spikes. We assume that we do not need to apply any distortion correction on the input image generated by our virtual camera, and we do not generate this effect on the synthetic dataset. Also, there is no need for homography calculation for perspective correction since it would only alter the morphology and pixel distribution and interpolate neighboring pixels. Moreover, the top field camera was configured with the following values (Table 5):

Parameter Name	Parameter Value
Field of View	75.5º
Sensor Size	36 mm
Depth of Field (Focus Object)	Field
Depth of Field (Aperture)	2.8
Depth of Field (Blades)	0
Depth of Field (Rotation)	0º
Depth of Field (Ratio)	1.0

Table 5: Virtual Camera Parameters

5.2.4 Renderer Configuration

The selected renderer we use in Blender Community (2018) is the Cycles Renderer Valenza (2015), together with the OptiX Parker *et al.* (2010) implementation of it's Ray Tracing Glassner (1991) engine. The OptiX Parker *et al.* (2010) brings an inherent hardware acceleration when available. Thus, we utilize an NVidia GeForce RTX 2070 with Max-Q Design GPU within a gamer notebook. This GPU has 8GB of RAM and dedicated modules for Ray Tracing Glassner (1991) operations, which accelerates the rendering process. We have configured the Blender's Cycles Renderer with OptiX Parker *et al.* (2010).

5.2.5 Render Results

In this section we show some validation images for our dataset generation. Since we want to simulate the influence of the light over the materials, we should expect that if there is a white material (such as the field drawings), it should reflect any of the available light source that interact with it.

5.2.5.1 Previewing Lighting from different points of view

Validating the how the materials responds to lighting changes as important as the segmentation we apply on it. An example of this is that if we have a blue light, the field drawings should be blue; if we have a yellow light, the field drawings should be yellow. However, when we combine the blue and yellow light sources, we have all three tricolor components (blue from the blue light source; red and green from the yellow light source). This effect can be observed on the right column in Figure 38. On the left column of Figure 38 we have the multi-color sun-like light sources.

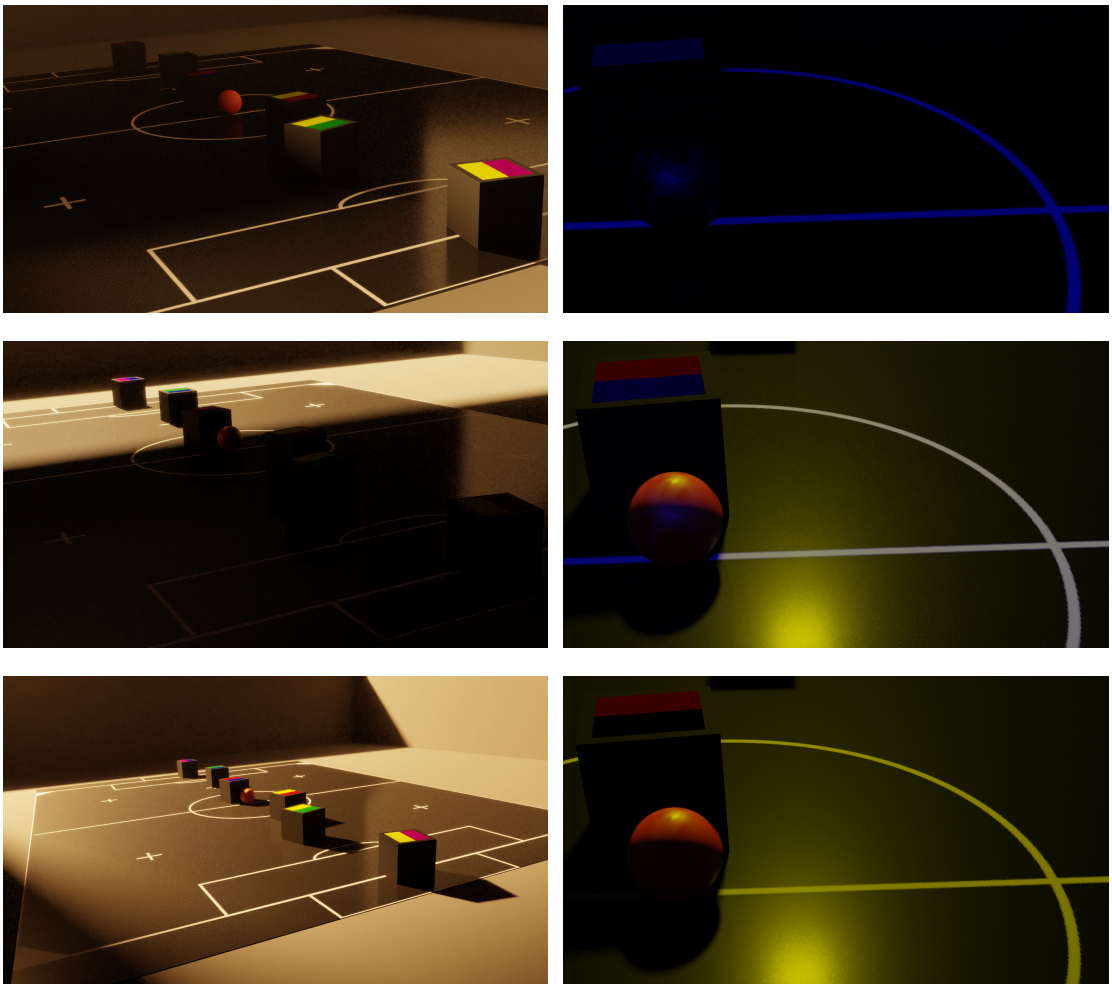


Figure 38: Different points of view rendering for material validation.

5.2.5.2 *Previewing Lighting from the Camera on Top of the field*

Another important validation is to test with a controlled light scenario, where we utilized the red, green, blue primary colors, and the yellow, magenta and cyan secondary colors to test how the tags and materials would respond to these conditions. We expected that the blue tag would only respond to blue, cyan, and magenta colors, since cyan and magenta both have a blue component. This is also true to green tag, that would only respond to green, yellow, and cyan colors, for a analogous reason from the blue tag example. Of course, the other tags would present their own colors to the same color light sources. Besides, when there is a tag that have no color component in common with the light source, it will be darker. Since we want to replicate a realistic-looking and realistic light interaction, we chose to set a little light dispersion in the 3D world, to simulate diffuse light comming from many directions, including from particles in air. This process causes aparent *black* tags to have some background noise in its texture. All these effects can be verified to happen in the modeled scene in Figure 39. And for a white light rendered reference image we have Figure 40.

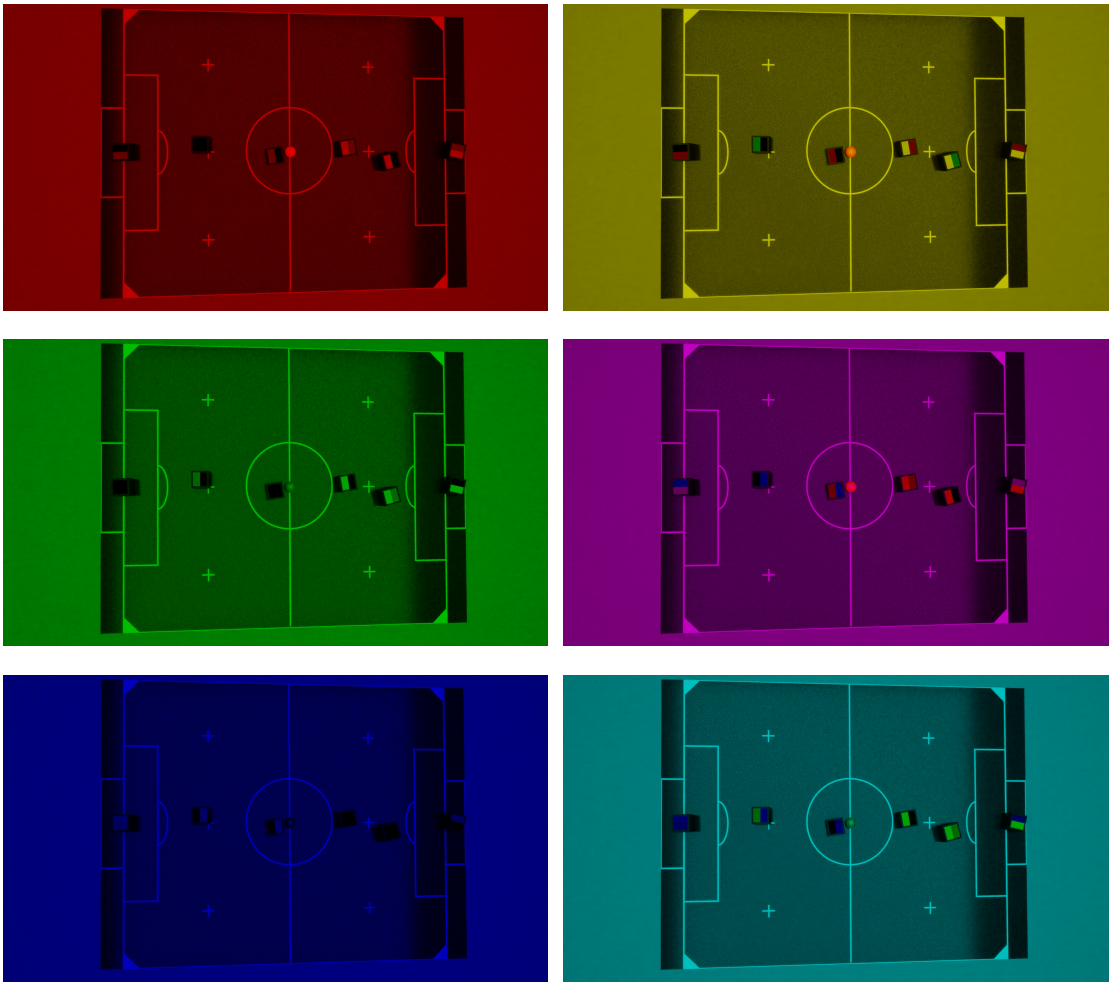


Figure 39: Different color light sources for material validation.

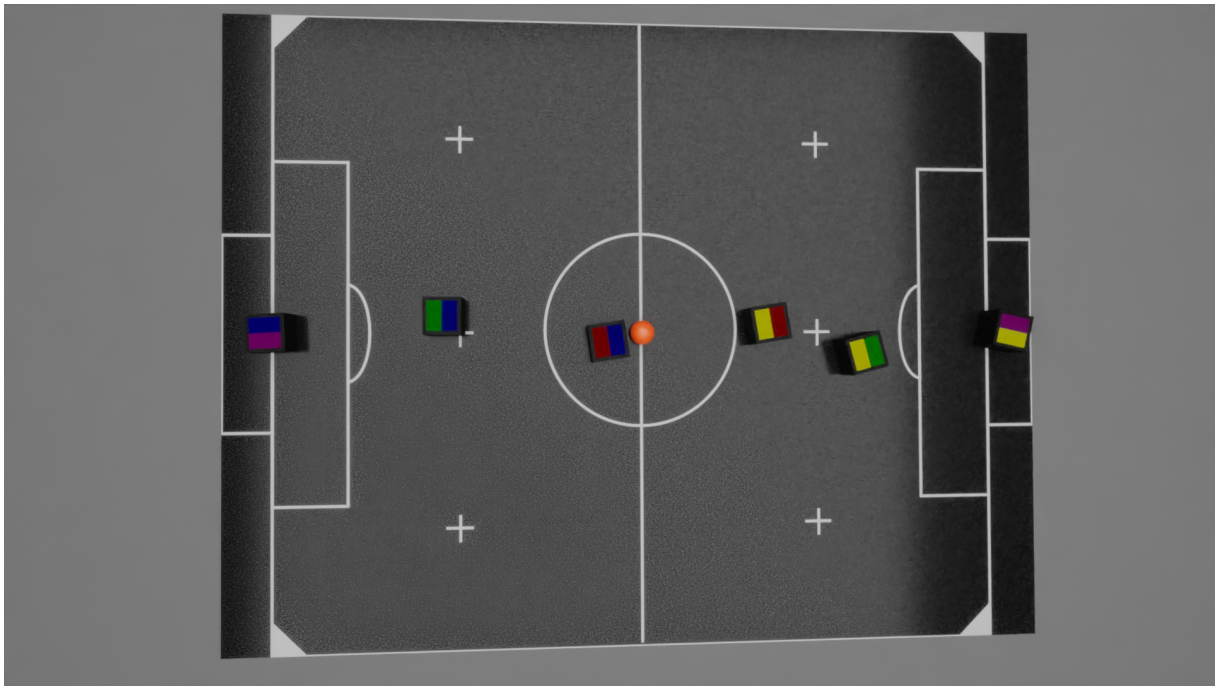


Figure 40: Rendered reference image with white light source.

6

RESULTS

We can demonstrate the combination of an RGB Normalization, followed by the Grayscale Filter, and finally, the HSV segmentation using the Hue Pivots on the Hue component. The detailed graphic interface (Figure 41) is intuitive since the team has been greatly accepted as easy to use and little to understand the concept of Hue Pivots. The detailed graphic interface is composed of:

- one big rectangle representing the colors from red to red in the hue component from left to right
- the value component from the top to the middle
- the saturation component from the bottom to the middle
- all the complementary values are set to the maximum
- this means that the upper half has the saturation component maximized, and the lower half has the value component maximized
- seven-colored squares to define which color the Hue Pivot must define
- and seven vertical lines to represent each one a Hue Pivot in its respective position

Each vertical line inside the big colored rectangle means a Hue Pivot, associated with a color tag. This information can be easily accessed with a left-click on the line that turns thicker and yellow when the user is closer. After the click is done, the corresponding box with the color tag associated with this Hue Pivot is surrounded by a white rectangle. The user can easily change the pivot's hue by dragging the line to the desired position (note that the user does not need to know the actual values) and can verify the segmentation's step in the other tab.

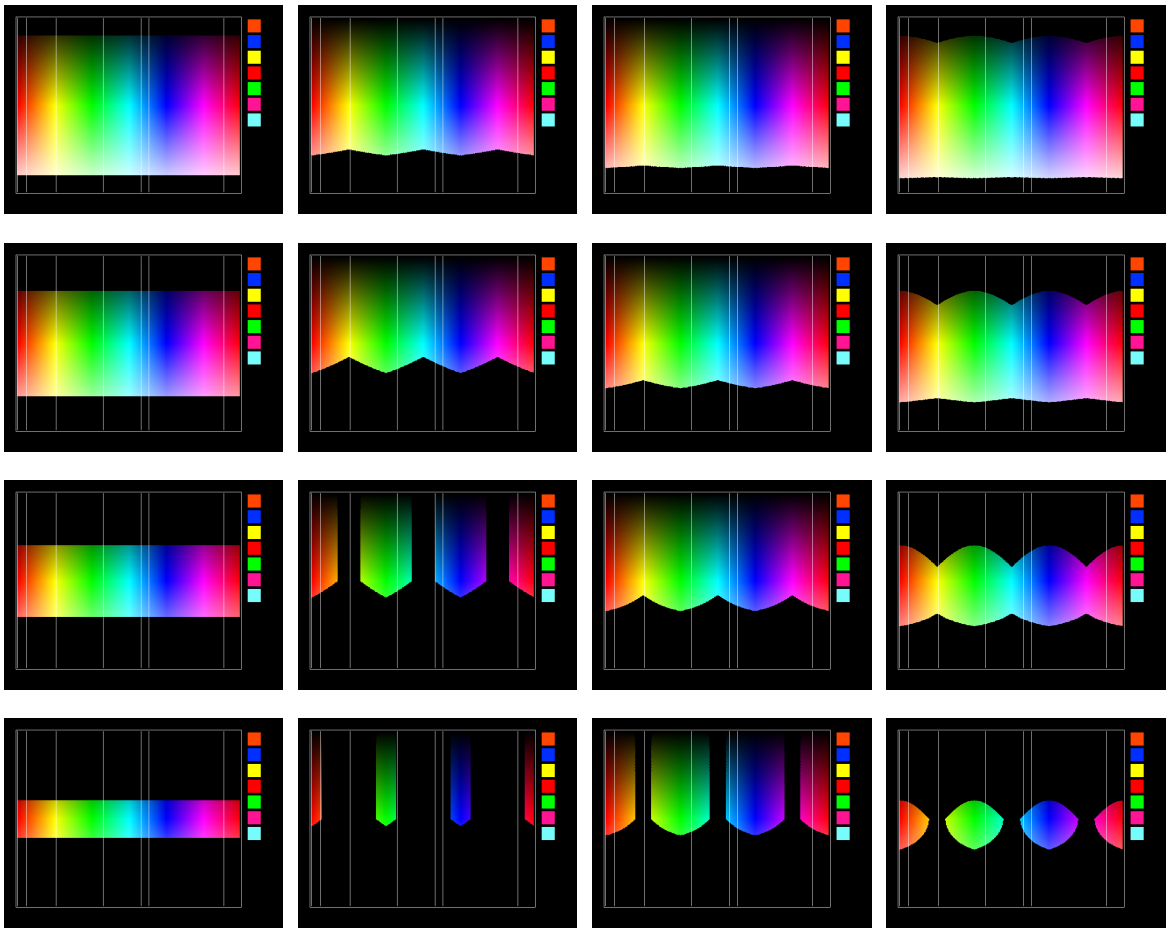
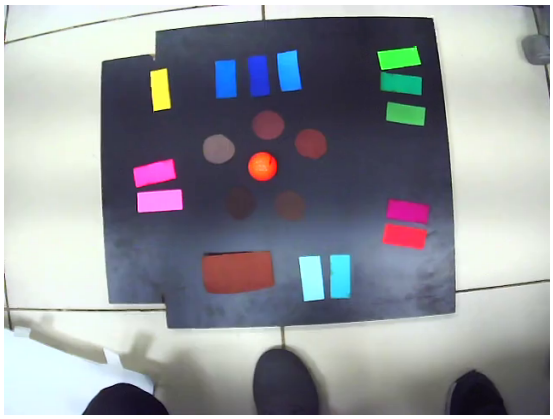


Figure 41: Detailed Graphic Interface comparison with different Normalization combined with Grayscale filter and a HSV planar visualization. First column: without normalization. Second column: Chromatic Normalization. Third column: Vector Normalization. Fourth column: Weighted Normalization. From top to bottom: $threshold = \{50, 100, 150, 200\}$.

The resulting segmentation is shown in Figure 42, in which we have the original image on the top and the segmentation result of each variation of normalization, including the first column without the normalization process. All of the results use the same pipeline with the Grayscale Filter and HSV segmentation (Hue Pivots).



(a) Original image

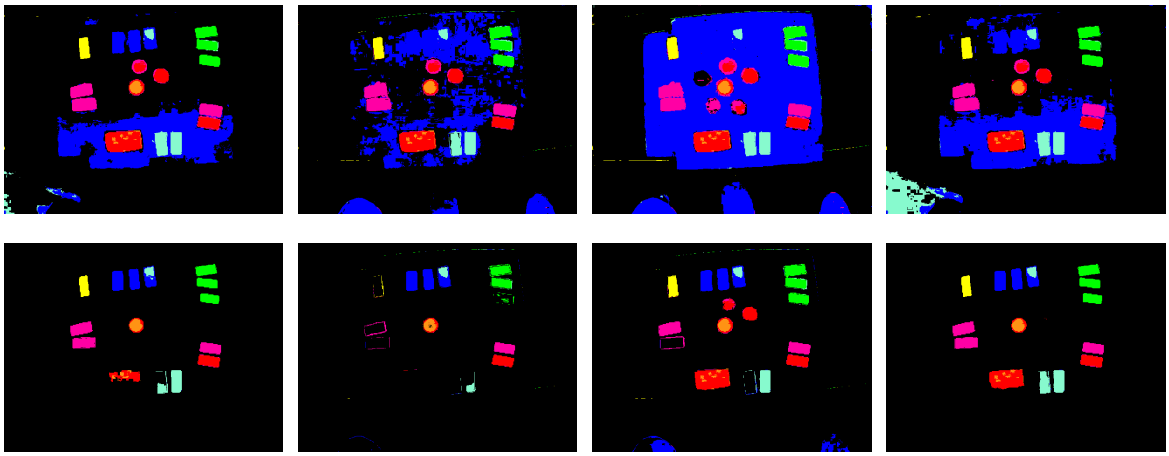


Figure 42: Original image and segmentation results in each column: without a normalization, CNT, VNT, WNT. From top to bottom the Grayscale Filter threshold value of each line $\{28, 86\}$

It is essential to note that we selected a black field part on the floor in this scenario and used all the tags with different colors. Moreover, the objective was to test the method's robustness and a filter to remove the ambiguity of brown color with orange and sometimes with red. Note that the CNT removed the brown tags, but also the tags that we want to keep. While the VNT preserved the most tags, the brown tags were also preserved. The difference is relevant when we compare the WNT to the approach *without normalization*. It shows that the WNT was able to recover the missing light-blue tag, and, at the same time, it filters mostly the brown tags that were marked as partially orange. (and depending on the illumination, this would happen with all the other methods).

We have also tested in real scenarios (Figure 43). We can verify that the Grayscale Filter has a significant impact on filtering the field. Moreover, since our camera is usually set to high-framerate (60fps), there is less light coming into the sensor, and the WNT behaves as

expected.

(a) Original image

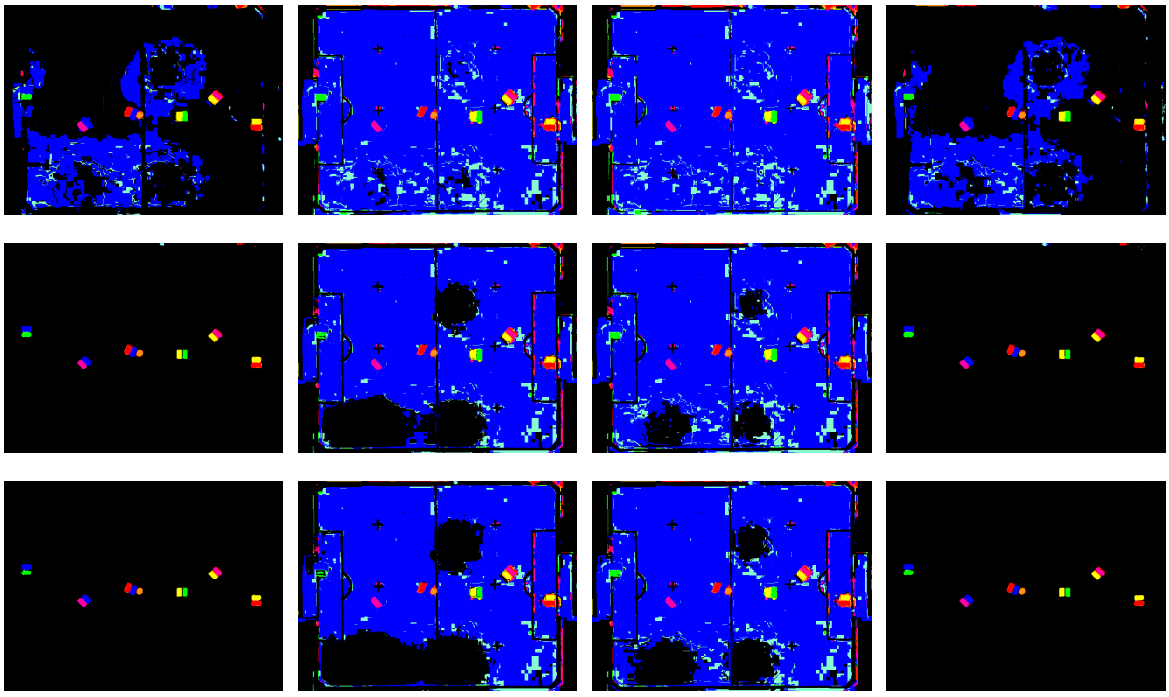
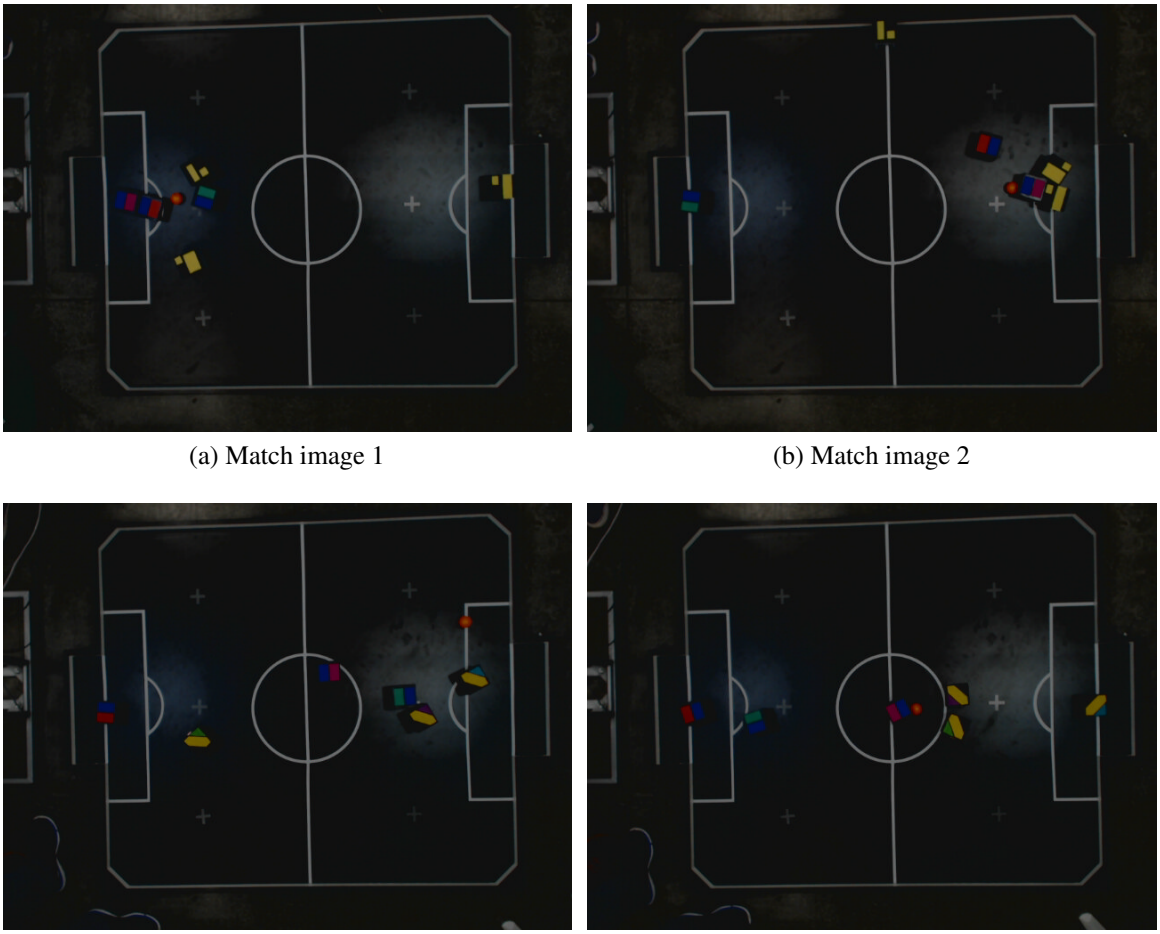


Figure 43: Competition video results in each column: without a normalization, CNT, VNT, WNT. From top to bottom the threshold value of each line $\{10, 28, 37\}$.

6.1 REAL DATASET RESULTS

In this scenario, we used the real-world images captured in laboratory tests or competitions. Figure 44 presents a normal condition of competition lighting after the camera setup, and some parameters configuration, such as exposure to lower the light intensity due to many lights contributions.



(a) Match image 1

(b) Match image 2

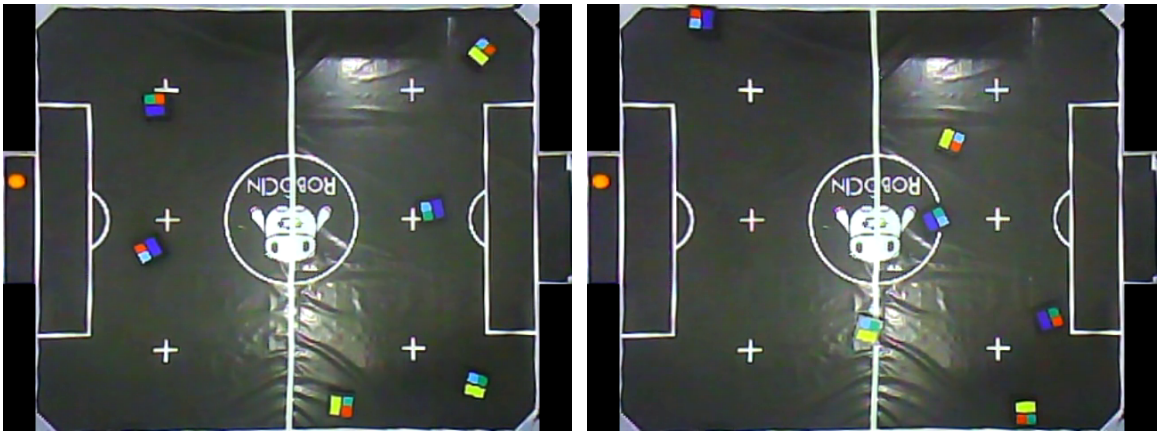
(c) Match image 3

(d) Match image 4

Figure 44: Normal samples on 3*v*3 competition match.

Figure 45 presents a challenging real scenario, where we have a window very close to the field and the influence of the sunlight in the middle of the field. There is also our flexible field in this scene, which reflects even more light than the wooden field. However, we can see that we have

Figure 46 and Figure 47 shows the resulting matches between the ground-truth segmentation and the MaggicSegmentation's segmentation for each RGB Grayscale Filter's threshold value. We can see that the CNT has the smallest range in which the most colors have the most matches. And the best normalization for this scenario is the WNT, where its values are mostly higher than the No Normalization. At the same time, both have a wider range of usable threshold values without compromising the segmentation. This means that the CNT and VNT were not a



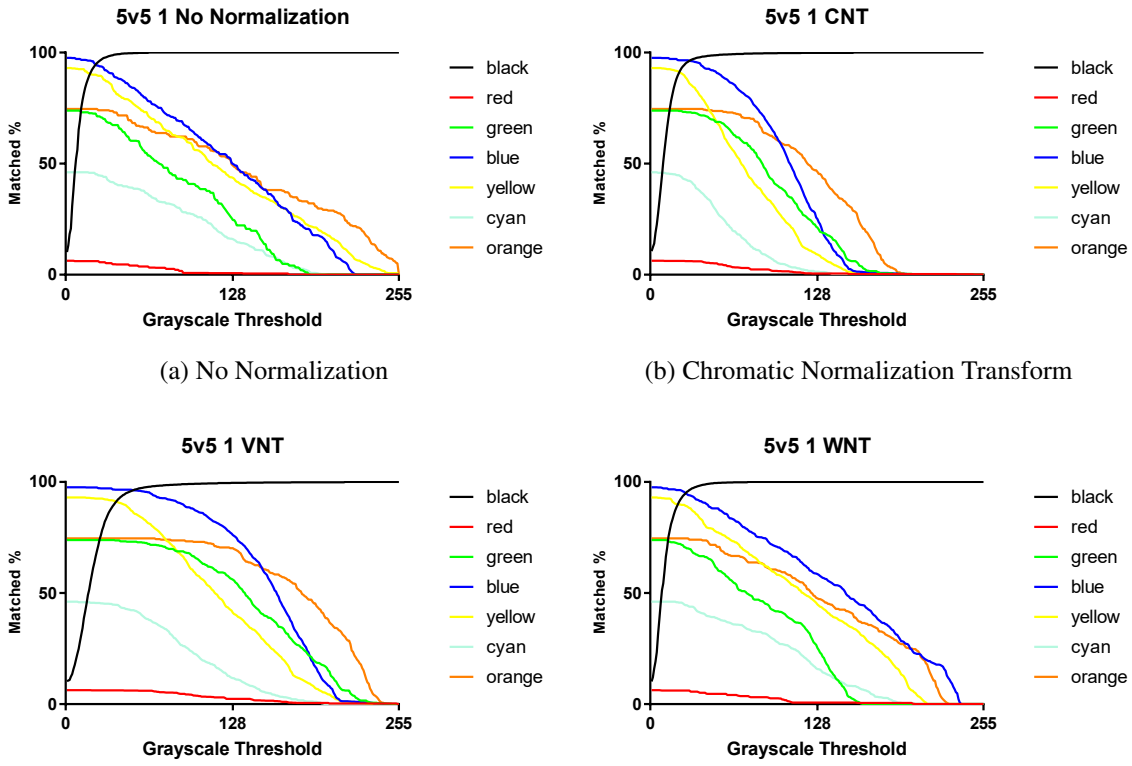
(a) 5*v*5 tests (Challenging)

(b) 5*v*5 test (More Challenging): robots in the middle-field

Figure 45: Challenging samples on 5*v*5 tests.

good option in this scenario, and the WNT and No Normalization would be preferable. However, we shall take into account that the No Normalization have the first color to fall near zero percent of match with the orange color, this means that we were losing the ball (orange) out of sight by the segmentation. Then, the WNT preserved a farther fall for the orange color, meaning we have a wider range for light variability that we should expect to capture the ball and robots' tags positions.

Another important feature shown in Figure 46, is that even if the VNT shows a longer range for orange, we also consider the black color matches. The black color is meaningful, we label everything that should not be an identifiable color (the color we use to identify the robots or the ball) with black. That means we will get less mistakenly labeled pixels with some identifiable color.



(a) No Normalization

(b) Chromatic Normalization Transform

(c) Vector Normalization Transform

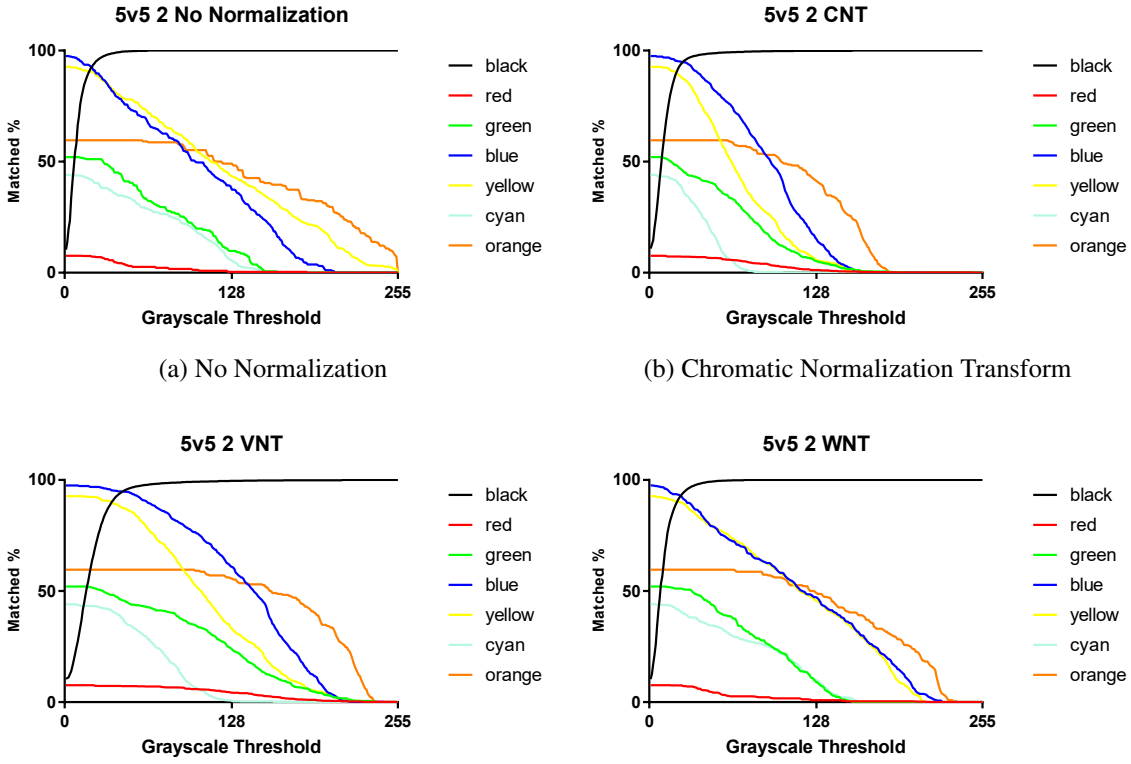
(d) Weighted Normalization Transform

Figure 46: RGB Normalization methods' comparison on 5*v*5 (1).

6.2 SYNTHETIC DATASET SEGMENTATION RESULTS

This section shows the segmentation results using the MaggicSegmentation (RGB WNT + RGB Grayscale Filter + Hue Pivots) on synthetic images we generated in Blender from Chapter 5. Figure 48 shows challenging lighting conditions simulating a sun-like light source and one almost uniform low-light scenario (top-right image in Figure 48).

A clear example of the challenge involved in this scenario can be seen in Figure 49. We were able to classify the target pixels (the tag and ball) correctly, but only when we use low values (1 in this case, see Figure 49a) for the RGB Grayscale Filter's threshold. And when we use a value slightly higher (4 in this case, see Figure 49b) we lose many pixels from tags and from the ball. Note that with the threshold set to 4 we already lost all the magenta tag from the yellow team on the image's right. As we progress to 10 (Figure 49c) in the RGB Grayscale Filter's threshold the blue tags are almost all gone, and some of the red and green tag from the blue team are very "damaged". And for the threshold $= 25$ (Figure 49d) we have lost all players on the field and only a piece of the ball is remanescent. It is important to note that this scenario is a challenging scenario. We do not propose to solve it in this work. Mainly because we do not use contextual information to identify the big yellow region at the top-left of the image, that has a higher luminance than the rest of the image. A more robust technique could overcome this limitation and take into account in the segmentation process.



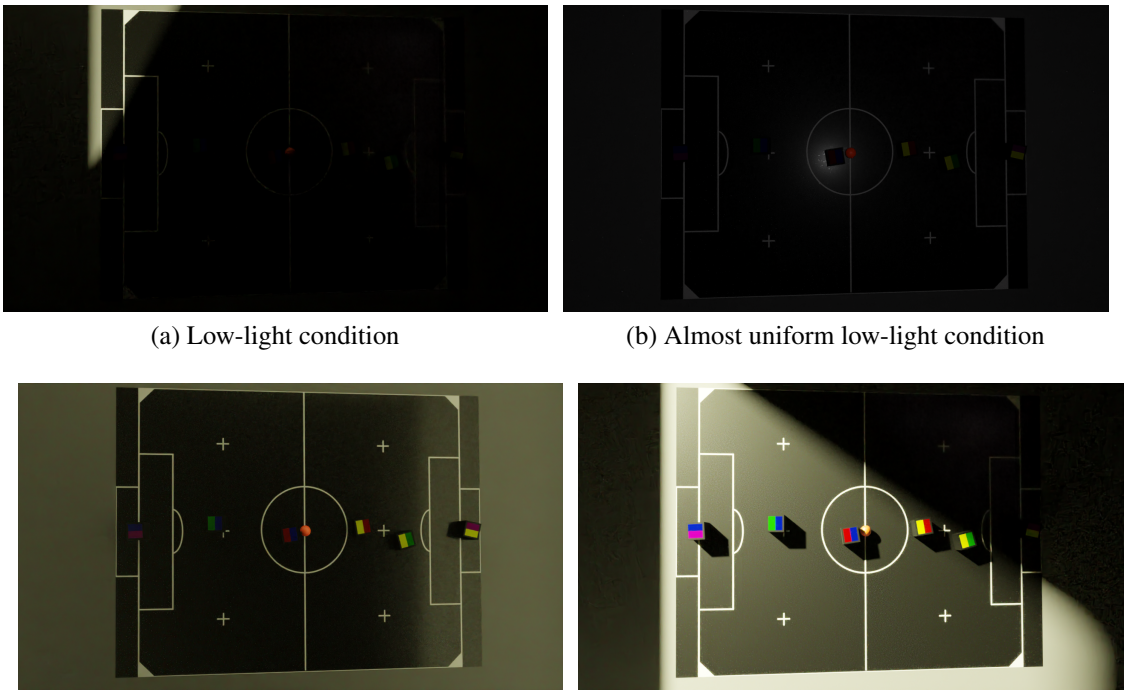
(a) No Normalization

(b) Chromatic Normalization Transform

(c) Vector Normalization Transform

(d) Weighted Normalization Transform

Figure 47: RGB Normalization methods' comparison on 5*v*5 (2).



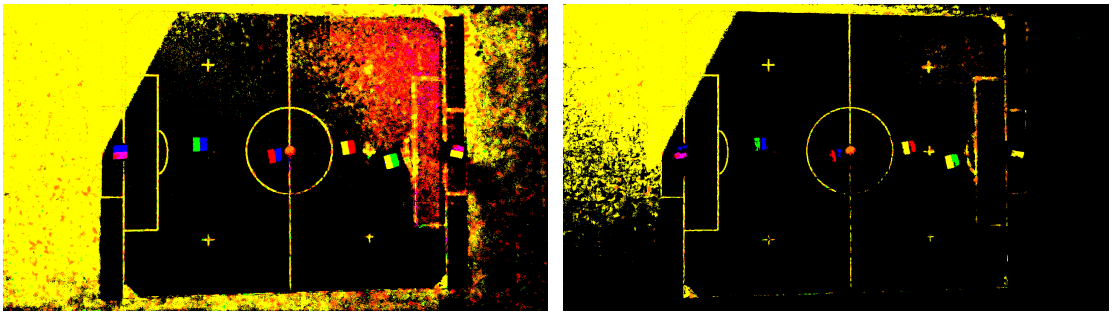
(a) Low-light condition

(b) Almost uniform low-light condition

(c) Penumbra illumination

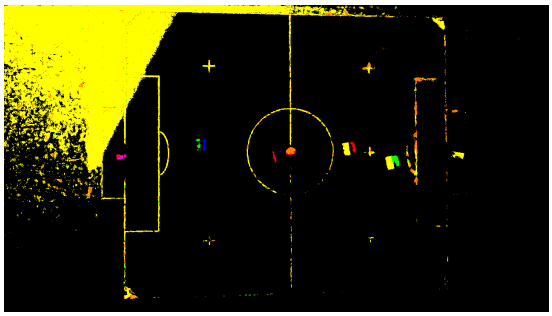
(d) High contrast illumination slicing the field

Figure 48: Different lighting conditions that are challenging for low lighting or high contrast.

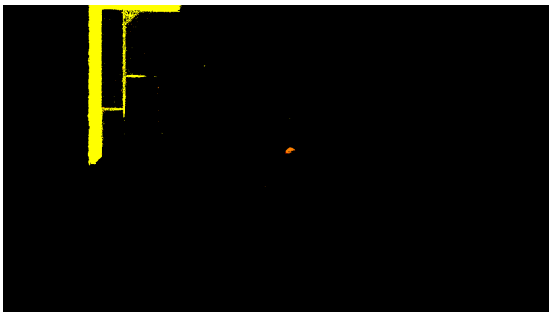


(a) threshold = 1

(b) threshold = 4



(c) threshold = 10



(d) threshold = 25

Figure 49: Resulting segmentation using the MaggicSegmentation on the challenging scenario of Low-light condition.

Figure 51 shows the resulting matches between the ground-truth and the segmentation for each RGB Grayscale Filter's threshold value. In this scenario we used the Figure 50 "Game" rendered as a ideal image to compare all the four variations of normalization (including No Normalization). We can see that the WNT and No Normalization have smaller ranges in which the most colors have the most matches. With a very strong fall in matches near the 100's, for the yellow and blue respectively. And the best normalization is the VNT, where its values are mostly higher than the CNT. At the same time, both have a wider range of usable threshold values without compromising the segmentation. However, these results conflicts with the real world dataset results. Probably, the results shown different behaviors due to some missing or mistaken material configuration during the modeling and rendering process of the synthetic dataset. One possible way of verifying this would test with a wider variety of color tag materials.

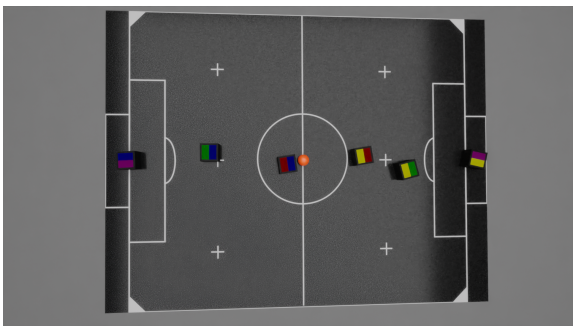
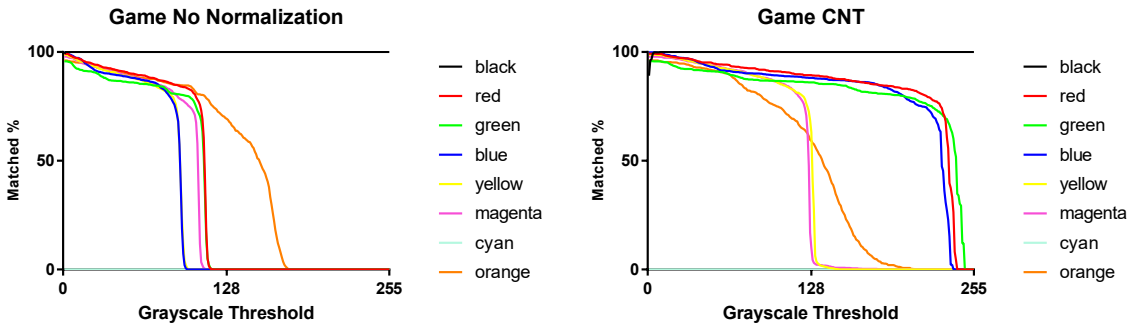
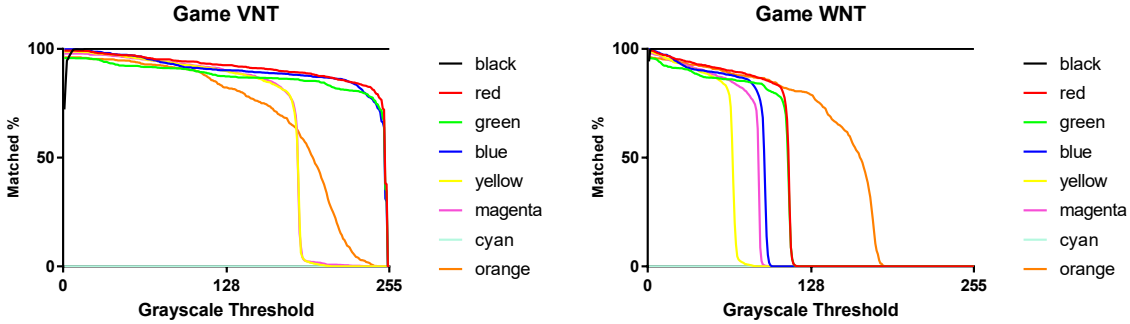


Figure 50: Rendered "Game" image with white light source and mostly uniform.



(a) No Normalization

(b) Chromatic Normalization Transform



(c) Vector Normalization Transform

(d) Weighted Normalization Transform

Figure 51: RGB Normalization methods' comparison on the ideal synthetic scenario.

7

CONCLUSIONS

As a challenging task, the color segmentation in IEEE Very Small Size Soccer is essential for a winner team. Each team has its own vision system, and takes time to configure it. The vision system is the only responsible for identifying and tracking the robots and the ball on the field. Each team has its own pattern with at least one color (blue or yellow), and a minimum coverage of the robot area. And it is essential to track your team's robots, the ball, and your adversary robots too, in order to take a good decision. Then, the vision system should be robust to the lighting conditions during the competition, and also the tag material colors, that are from different manufacturers.

We tackle this challenge by assuming each pixel has enough information for its classification. This hypothesis has some limitations, such as contextual information. Contextual information could help us recover the right color label. However, we want the system to be fully context-free and very fast to diminish the delay time between the image capture and the control being executed in the robot. This time requirement is important to maintain a good robot control and a fast response during the soccer match.

In order to do an holistic evaluation of the proposed method we used vector fields (to verify the normalization behavior), generated a dataset with real images, and synthetic images that could tackle grading challenging scenarios, and the RGB Grayscale Filter's threshold value influence in the correct classification of the pixels.

In our proposed method, we were able to achieve better results in real scenarios, by utilizing the RGB Weighted Normalization. Our RGB Weighted Normalization Transform effectively diminish ambiguity of between grayish colors and colors with some level of chromaticity (reddish, greenish, blueish colors). We achieved this by increasing the distance between the color components in RGB, pending for a choice based on the subtle difference in color composition. Even if it does not give the same result for yellow, magenta, cyan like colors.

Moreover, we have demonstrated that the RGB Grayscale Filter also brings a reduction in false-positive classification, since it allows the user to remove colors that have low chromatic contrast. We demonstrate that it reduces the complexity of color segmentation's calibration executed by a human; simultaneously, it brings a more robust segmentation due to light invariant properties in HSV color space. It takes the competition to another level since the team does not

need to focus on reconfiguring the vision module once it is calibrated.

The segmentation results in the normalization comparisons were consistent with the VSSS match reality. The MaggicSegmentation method is used now for more than three years by our team in VSSS competitions, and the WNT has been the default since then. Furthermore, the MaggicVision makes the segmentation process so simple that our team could focus on strategy, electronics, mechanics, and prototyping during competition and on daily research. The faster the configuration, the faster we were up to testing new strategies, or robot modifications.

One contribution of this project, beyond the MaggicVision + MaggicSegmentation is the real and synthetic dataset, which tries to validate the segmentation in different levels of difficulties. Since we are in control of the lighting conditions in the synthetic generation, a future work would consider other aspects to the dataset generation, such as: image noise, compression artifacts, YUV input losses in cromaticity, and more. Also, the dataset should grow in size and possibly in two versions. Each version would assume different philosophies: first, which color should the pixel be labeled to, and second, what is this color's label. This means, the first assumes that it is already known what color label is expected for the color, since it would have knowledge of the context. The second means that it does not matter the pixel surroundings, it should be only labeled by its own components.

REFERENCES

Adelson, E. H. (1995). Checkershadow illusion. http://persci.mit.edu/gallery/checkershadow.

Akimoto, N., Zhu, H., Jin, Y., & Aoki, Y. (2020). Fast soft color segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8277–8286.

Bradski, G. (2000). The OpenCV Library. *Dr. Dobb's Journal of Software Tools*.

Chude-Olisahl, C. C., SUlong, G., Chude-Okonkwo, U. A. K., & z. M. Hashim, S. (2013). Illumination normalization for edge-based face recognition using the fusion of rgb normalization and gamma correction. *IEEE International Conference on Signal and Image Processing Applications*.

Community, B. O. (2018). *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam.

Corke, P. (2011). *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg.

Datumizer (2015). Hsv color solid cylinder. https://commons.wikimedia.org/wiki/File:HSV_color_solid_cylinder.png.

de Sales Júnior, A. L. D., de Abreu, C. C., Pena, C. H. C., de Oliveira, C. S., de França Tenório, D. R., da Silva Júnior, E. B. L., Bandeira, G. M., Medeiros, H. R., da Silva, J. G. M., Santos, L. H. C., Maggi, L. O., *et al.* (2017). Robocin - equipe de robótica do centro de informática. http://sistemaolimpo.org/midias/uploads/4d7bb6bd60dedbf70d6f4cd1ebeb9ce3.pdf.

Finlayson, G. D., Schiele, B., & Crowley, J. L. (1998). Comprehensive colour image normalization. *Lecture Notes in Computer Science*, 475—-490.

Finlayson, G. D. & Tian, G. Y. (1999). Color normalization for color object recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 1271—-1285.

Glassner, A. S. (1991). *An Introduction to Ray Tracing*. Academic Press, USA.

Gonzalez, R. C. & Woods, R. E. (2006). *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA.

Hughes, J. F., Dam, A. V., McGuire, M., Sklar, D. F., Foley, J., Feiner, S., & Akeley, K. (2014). *Computer Graphics - Principles and Practice*. Addison-Wesley Professional, 3 edition.

Lalitha, M., Kiruthiga, M., & Loganathan, C. (2013). A survey on image segmentation through clustering algorithm. *International Journal of Science and Research*, 2(2):348–358.

Li, X. & Plataniotis, K. N. (2018). Novel chromaticity similarity based color texture descriptor for digital pathology image analysis. *PloS one*, 13(11):e0206996.

Mrsi, I. (2013). Yuv uv plane. https://en.wikipedia.org/wiki/File:YUV_UV_plane.svg.

Parker, S. G., Bigler, J., Dietrich, A., Friedrich, H., Hoberock, J., Luebke, D., McAllister, D., McGuire, M., Morley, K., Robison, A., & Stich, M. (2010). Optix: A general purpose ray tracing engine. *ACM Trans. Graph.*, 29(4).

Pbrks (2018). Checker shadow illusion. `https://en.wikipedia.org/wiki/File:Checker_shadow_illusion.svg`.

Pingstone, A. (2015). Grey square optical illusion proof2. `https://commons.wikimedia.org/wiki/File:Grey_square_optical_illusion_proof2.svg`.

Pinto, A. H. M. (2020). *IEEE Very Small Size Soccer (VSSS) Rules*.

Prince, S. J. D. (2012). *Computer Vision: Models, Learning, and Inference*.

Saraydaryan, J., Leber, R., & Jumel, F. (2019). People management framework using a 2d camera for human-robot social interactions. In Chalup, S., Niemueller, T., Suthakorn, J., & Williams, M.-A., editors, *RoboCup 2019: Robot World Cup XXIII*, 268–280.

Sharma, N., Mishra, M., & Shrivastava, M. (2012). Colour image segmentation techniques and issues: an approach. *International Journal of Scientific & Technology Research*, 1(4):9–12.

Sinha, P., Balas, B., Ostrovsky, Y., & Russell, R. (2006). Face recognition by humans: Nineteen results all computer vision researchers should know about. *Proceedings of the IEEE*, 94(11):1948–1962.

Stone, J. E., Gohara, D., & Shi, G. (2010). Opencl: A parallel programming standard for heterogeneous computing systems. *Computing in Science Engineering*, 12(3):66–73.

Szemenyei, M. & Estivill-Castro, V. (2019). Robo: Robust, fully neural object detection for robot soccer. In Chalup, S., Niemueller, T., Suthakorn, J., & Williams, M.-A., editors, *RoboCup 2019: Robot World Cup XXIII*, 309–322.

Team, B. D. (2021). Glossy bsdf. `https://docs.blender.org/manual/en/latest/render/shader_nodes/shader/glossy.html`.

Upcroft, B., McManus, C., Churchill, W., Maddern, W., & Newman, P. (2014). Lighting invariant urban street classification. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 1712–1718.

Valenza, E. (2015). *Blender Cycles: Materials and Textures Cookbook, Third Edition*. Packt Publishing, 3rd edition.

Walker, J., Halliday, D., Resnick, R., & Merrill, F. E. J. J. (2013). *Fundamentals of Physics - Extended (10th Edition)*. John Wiley & Sons, Inc, USA.

Yu, L., Cheng, Y., & van de Weijer, J. (2018). Weakly supervised domain-specific color naming based on attention. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 3019–3024.