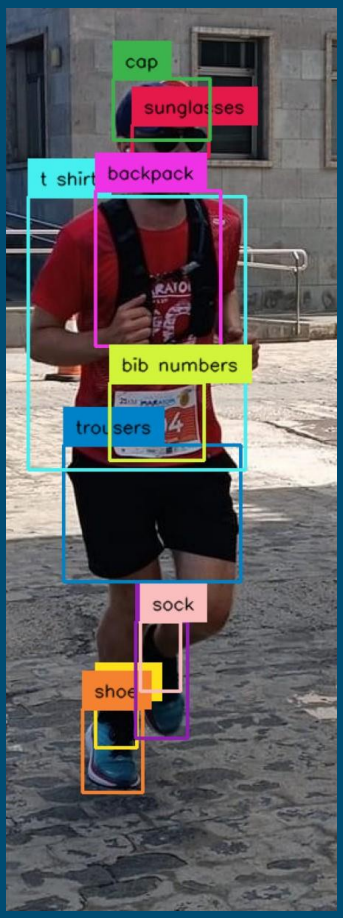
Identifying participants at sporting events using AI and computer vision



4th year Infotronic internship report

Presented by:

Clément GHYS

4A-ILC Schrödinger promotion

Carried out at: Escuela de ingeniería informática de la ULPGC

From 10/04/2023 to 30/06/2023

José Javier Lorenzo Navarro

Charles Meunier

Table of content:

List of figures

Presentation of the host organization

The School of Computer Engineering of the University of Las Palmas de Gran Canaria is dedicated to the training of highly qualified professionals, who are prepared to meet the requirements of the information society. Its main objective is to foster economic development and social well-being by promoting innovation and excellence.

Founded in February 2010, the school is the result of the merger of the University School of Computer Science and the Faculty of Computer Science, following the adaptation of the university to the standards of the European Higher Education Area. The University School of Computer Science was the first university center dedicated to computer science in the Canary Islands, while the Faculty of Computer Science was created in 1986 within the Polytechnic University of Las Palmas.

Originally based at the University College of Las Palmas, the school moved in 1992 to a new building dedicated to computer science and mathematics on the Tafira campus. It offers up-to-date training plans and maintains close links with society and the business world to anticipate changes and requirements in the ever-changing field of information technology.

Within the School of Computer Engineering research and projects are conducted around the fields of artificial intelligence and *computer vision*[1] and it is in this environment that I was integrated.

Thanks

I would like to warmly thank Mr. José Javier LORENZO-NAVARRO and Mr. Modesto CASTRILLÓN-SANTANA for their accompaniment and their precious help throughout the course. Their expert advice, availability and kindness allowed me to deepen my knowledge in computer vision and artificial intelligence. As well as developing essential skills. I would also like to thank Mr. Charles MEUNIER for his advice and help in finding the internship.

[1] Computer vision is a field of artificial intelligence that allows machines to process and analyze visual information using cameras, data, and algorithms. It aims to replicate some of the capabilities of human vision, such as object recognition, distance estimation, and motion detection, but more quickly and accurately.

GENERAL INTRODUCTION

Context

The TransGranCanaria is an internationally renowned trail race that takes place every year on the island of Gran Canaria. The race offers several distances ranging from 17 km to 128 km, it is considered one of the most difficult races in the world due to the variety of its landscapes and elevations of up to 7500 meters. Runners traverse sandy beaches, rugged mountains, pine forests and arid ravines, making it a real physical and mental challenge.

Problematic

The recognition and detection of runners at the various checkpoints is of paramount importance to ensure their safety and monitor their progress throughout the race. Until now, this task has been done manually by a team of volunteers and race staff, which often resulted in delays in tracking runners and a margin of error in identifying participants.

Objective

The main objective of this course is therefore to design, develop a system capable of identifying the runners from the images captured at the different checkpoints.
Indeed, in this context, the use of image analysis and AI can provide an innovative solution by automating the process of recognizing runners.

Methodology

As a first step, it is essential to think about the methodology to be used in order to be able to identify the riders at the different checkpoints.
To recognize runners there are several interesting possibilities.

1. Facial recognition
2. Bib number detection
3. Clothing recognition
4. Recognition of the morphology of the runner

As for the recognition of morphology, there are many parameters that can make the task overly complicated. Indeed, the quality of the images, the angle of the camera as well as the posture and position of the runner can induce major complications to detect his morphology. In addition, many runners have a similar morphology which makes it a very unreliable indicator.
Facial recognition is an interesting avenue but requires a large number of samples and a lot of learning time. It also raises questions about data security and privacy.
The last two options therefore seemed the most promising.

Organization

To realize this ambitious project, we were two to work on this project with parts realized in collaboration and others realized entirely individually.
My colleague Maxime Devoucoux focused on the number detection part of the bib and for my part I mainly worked on the classification of clothes.

Chapter 1: Description of a Runner

Introduction

This chapter aims to present the diverse ways to describe a runner.

Part 1: Bib

Each runner is provided with a bib with a number. In the case of the TransGranCanaria (TGC) the runners have a bib with a 3-digit number.
One of the solutions is therefore to retrieve the number of the runner's bib which therefore makes it possible to identify him formally. This solution is therefore very dependable if it is possible to accurately detect numbers but may be useless if the bib is not visible.
In addition, the bib may not be visible due to the angle or quality of the photo or simply if it is hidden by an external element or by the runner himself.

Part 2: Clothing

It is relevant to study the clothes of runners to predict the identity of a runner, because each runner wears different clothes and of various kinds. In addition, runners wear clothing of an assortment of colors, which is an additional way to identify them. By analyzing the different garments worn by a runner and their colors, it is possible to create a unique identifier for each runner. This approach is interesting because by determining the clothing profile of a runner (types of clothing and colors), it becomes easier to predict his identity.
But this approach presents a major problem. Indeed, during the race, given its long duration it is possible that the weather is not constant and forces the runners to change clothes. For example, by adding a rain jacket or removing sunglasses. The change of clothing during the race can therefore strongly impact the determination of the runner.

Conclusion

In conclusion of this chapter the detection of the runner's bib and clothes are two interesting methods but with defects. Combining them could therefore reduce the inaccuracies and imperfections of each of them.
In *Annex 2* there is a diagram describing the ways to describe a runner.

Chapter 2: Detection

Introduction

This chapter will discuss object detection in an image. Indeed, in order to be able to conduct operations on bibs and clothing it is first necessary to be able to detect the elements of interest in a complete image.

Part 1: Object detectors

Object detectors are especially essential elements in the field of computer vision and artificial intelligence. In addition, they make it possible to locate and identify objects present in each image.
Their use is essential for the automatic recognition of specific entities, such as individuals, vehicles, animals, buildings, and many more, within a particular scene. These detectors are usually based on convolutional neural networks (CNNs) capable of learning to extract distinctive visual features from images that is made possible by *Supervised learning*.

Part 2: Introducing YOLO

On the advice of Mr. Castrillon and after extensive research, special attention was paid to the YOLO (You Only Look Once) object detector. As described in the article [1], Yolo is a state-of-the-art open-source system for real-time object detection, which uses a single convolutional neural network to detect objects in images. To work, the neural network divides the image into regions, predicts *Bounding boxes* and probabilities for each region, and then weights the boxes according to the predicted probabilities. The algorithm learns generalizable representations of objects, which allows low detection error for new inputs, which differ from the training dataset.

Part 3: Learning Model

In order to detect the elements of interest (namely the bib and the clothes) in the raw image it is necessary to use a suitable model, that is to say trained to detect the *Classes* necessary for the project. On the YOLOV7 GitHub there are models pre-trained on the COCO dataset[1]. But as visible on *Annex 3* the categories are quite general and there are no necessary categories (bib, T-shirt, shorts, sunglasses, etc...). Therefore, it is not possible to simply use one of the YOLO models trained on the COCO dataset for the project.

Conclusion

In conclusion, the detection of objects in an image is an essential step to perform operations on bibs and clothing. However, for our project, it is necessary to use a learning model adapted specifically to the classes of objects that interest us. The pre-trained models available on the YOLOV7 GitHub on the COCO dataset do not contain the necessary categories, making them impossible to use directly. The solutions available are therefore to train a customized model or to look for a pre-trained model that can suit the needs of the project.

[1] Coco Dataset (Common Objects in Context Dataset) is a collection of annotated images widely used for computer vision research. The Coco dataset includes more than 200,000 images and covers 80 different object categories.

Chapter 3: Grounding DINO

Introduction

The solution of training a custom model is a solution that could be interesting but very tedious. This solution would first require collecting a sufficiently considerable number of images from each category because the dataset of the images taken from the TGC would not have been sufficient given their too small number.
In addition, it would have been necessary to manually *Labelling* several thousand images which would have taken a considerable time.
That is why looking for a pre-trained model that could suit the needs of the project was a better solution.

Part 1: Grounding Dino Overview

As presented in the article [2] Grounding DINO is an open-set object detector that relies on the DINO detection model which is a *Self supervised training* model, as explained in the article [3]. Grounding DINO represents an improvement of DINO by integrating *Transformer* using the principle of grounding to allow the detection of arbitrary objects from human input such as category names or referential expressions.
This approach effectively combines language and vision, making it possible to detect open objects or concepts using the linguistic information provided by users.
In other words, by using an input prompt that can be a simple sentence or a list of words, Grounding DINO is able to detect, in an image, the elements described by the sentence or present in the list. Details of this feature are described in *Annex 4*.
The paper [2] also presents a comparative table of the performance of Grounding DINO and other similar neural networks. This table also lists the datasets used for model training. It is notable that Grounding DINO has been trained on several datasets, including millions of images and hundreds of classes. These features make it a remarkably interesting model to use for the project.

Part 2: Configuring Dino

The installation of Dino is done by making a git clone of the Repository [4]. As explained in the previous part Dino takes as input a prompt which can be a sentence describing what one wants to look for in the image or a list of things to be detected. For this project we need to detect the different clothes and accessories of the runner, namely: 'cap', 't shirt', 'sunglasses', 'shoe', 'sock', 'backpack', 'walking-sticks', 'bib numbers', 'trousers'.

These strings as well as the path of the runner's image are passed to a model instance configured with the weights of the file [5] previously uploaded to the Grounding Dino GitHub. It is also possible to modify the threshold values (BOX_TRESHOLD and TEXT_TRESHOLD) the precision threshold values below which bounding boxes and labels do not display.
The default values of 0.35 and 0.25 were retained because tests showed that they represented the best compromise. Indeed, by decreasing the thresholds, there was an increased risk of detecting duplicates, while by increasing the thresholds, the number of detected elements decreased significantly.

Part 3: Using Dino

Here is an example of using Grounding Dino on an image of the TGC runner dataset.

Figure 1 - Photo of a runner from the TGC 2020 dataset

Figure 2 - Photo of a runner after detection with Grounding Dino

It can be seen in *Figure 2* that the following elements were detected: 'cap', 't shirt', 'sunglasses', 'shoe', 'sock', 'backpack', 'bib numbers', 'trousers'

It is interesting to note that all the clothes and accessories of the runner and present in the list of elements to look for in an image have been detected. This is not always the case depending on the images, indeed elements such as the quality of the image, its parameters (brightness, exposure, angle, zoom, etc...) as well as the position of the runner can influence the detection.

Conclusion

Using Grounding Dino, a pre-trained model combining language and vision, turned out to be a much better solution. Grounding Dino allows object detection using linguistic information and offers good detection performance. Its use avoids the constraints of training a custom model, thus offering a practical and effective solution thanks to the substantial number of classes it can detect.

However, despite several attempts, it proved impossible to detect the diverse types of clothing. This can be due to two potential reasons: either more precise classes (such as long-sleeved T-shirts) are not part of the classes used for model training, or classes have too much similarity and proximity (such as T-shirts and long-sleeved T-shirts), making them difficult to differentiate using the language-guided query selection method explained in detail in section 3.2 of the article [2].

The size and zoom of images also play a crucial role. Indeed, the larger the image, the longer the detection takes. After several experiments, it turned out that the detection worked much better when cropping the image to keep only the runner.

Chapter 4: Segmentation

Introduction

In order to be able to exploit the different elements detected, it is necessary to divide the areas of interest. The problem could have been solved by extracting the coordinates of the bounding box and cropping the image accordingly. However, this approach would have had major drawbacks. By cropping the image in this way, the result obtained would have included elements of the background or pieces of other detected elements, which is not desirable for performing processing or other detections on the identified objects.
For this we used a segmentation tool released on April 5, 2023: Segment Anything.

Part 1: Introducing Segment Anything

As presented in the article [6], Segment Anything is an innovative project that offers a new task, model, and dataset for image segmentation.
The model used in Segment Anything is called SAM (Segment Anything Model). SAM has three components: an image encoder, a flexible prompt encoder, and a quick mask decoder.
Image encoder uses pre-trained vision transformer (ViT) *MAE* to process high-resolution input. The flexible prompt encoder allows users to enter prompts that guide the model segmentation process. Using SAM in a data collection loop, the Segment Anything team created the largest segmentation dataset to date, comprising over one billion masks across 11 million images.

Part 2: Using SAM

The article also reveals that it is possible to pass a bounding box as a prompt in order to segment the object it contains. This feature will be used by exploiting the bounding boxes of objects detected using Grounding Dino. Here is an example of the use of Grounding Dino on the *Figure 1*.

Figure 3 - Resulting image from Figure 1 after segmentation with SAM

In Figure 4 it is possible to observe that the different elements detected have been segmented and colored. SAM also allows you to retrieve the masks of the different objects detected as shown in Figure 4.

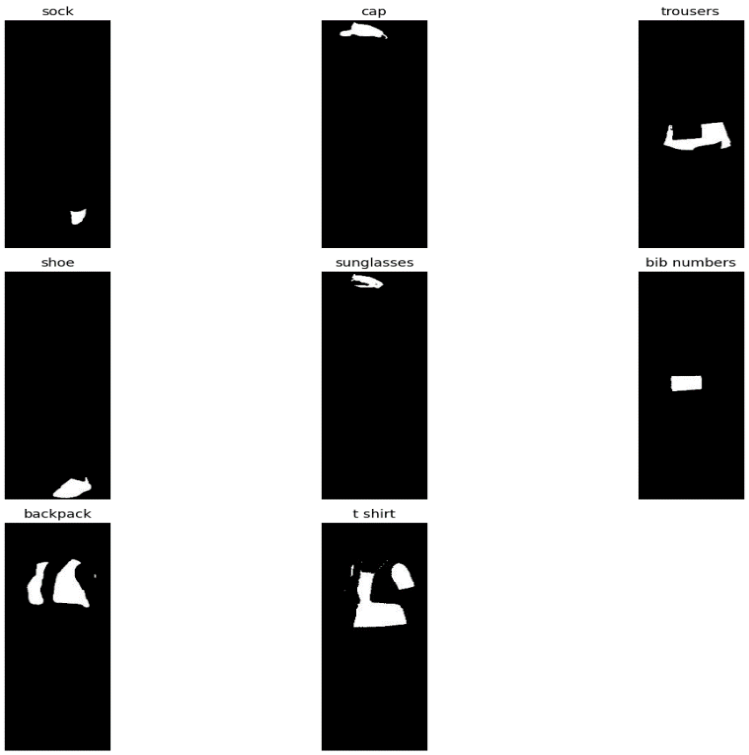


Figure 4 - Masks resulting from segmentation in Figure 1

By adding up the masks that have been previously converted to binary format and the raw image, it is possible to obtain the segmented elements with their original color. The colorized masks resulting from the segmentation in *Figure 1* are available in the *Annex 5*.

Conclusion

In conclusion, image segmentation is essential to effectively exploit the detected elements. The segmentation tool Segment Anything, with its SAM model, allows precise segmentation using bounding boxes as prompts. The masks of segmented objects can be superposed on the raw image, facilitating further processing.

Chapter 5: Primary Color Detection

Introduction

As described in the general introduction, the color of clothing is a relevant criterion to be used to classify runners. After doing a lot of research, it turned out to be impossible to find an AI model that could detect colors. It was therefore necessary to create a program to detect the main color of a garment. This chapter deals with color detection by computer vision and image processing.

Part 1: RGB Detection

From a colorized mask the purpose and extract the main color from it. Thanks to the OpenCV library it is possible to extract the values of the main colors of an image using the K-means algorithm whose operating principle is detailed in the *Annex 6*.Once these values are retrieved, they are stored in an array. We then go through the image by counting the number of occurrences (number of pixels) of each color of the table.
The value 0.0.0 (black in RGB) is removed because it corresponds to the background of the image is therefore overrepresented. From these values a weighted average is made to calculate the RGB value of the average color of the image. This average value is changed to a KNN[1] trained on an RGB value array and the corresponding color. Thanks to this it is possible to detect the main color of a garment. But unfortunately, the results are not at all satisfactory. Indeed, the garment that we see of a certain color is not detected as such because of the exposure of the image as well as the average performed. An example is available in the *Annex 7*. It was necessary to find another approach that would provide more precision to determine the color of a garment.

Part 2: Conversion to HSV

To understand this part, it is necessary to understand the colorimetric models. These are mathematical representations used to describe and specify colors quantitatively. The focus will be mainly on HSV (Hue, Saturation, Lightness) and RGB (Red, Green, Blue) whose 3D representations of their model are available in the *Annex 8*. As detailed in *Figure 10*, the representation of the HSV space is in the form of a cylinder whose Hue value that can be assimilated to pure color is quantified by a value of angle in degree around the cylinder. And the values of *Value* and *Saturation* represent the height and radius of the cylinder, respectively. The principle is to use the K-means algorithm to retrieve the values of the main colors and then convert the RGB values into HSV values. The conversion is carried out thanks to the Equations obtained from the formulas of the article [7] available in the *Annex 9*.

[1] KNN (K-Nearest Neighbors) is a machine learning algorithm used in the field of artificial intelligence. It is used for supervised classification and regression. The main idea of KNN is to predict the class or value of a new instance based on the closest "k" examples in the training dataset. Similarity between instances is usually measured using Euclidean distances or other measures of similarity.

Then, in the same way as with RGB detection, it is necessary to calculate the average value of the main colors retrieved by the K-means algorithm. A first test was conducted by averaging the different components H, S, V between them but given the particular distribution on the model and in particular for the Hue it was not conclusive. Indeed, for example for a red garment the main colors will have Hue around 340-360 ° and around 0-20 ° gold by averaging we do not find at all an average value corresponding to a shade of red. It is therefore necessary to introduce another calculation of average.

Part 3: Cylindrical coordinate mean

The method consists of calculating the average using the polar coordinates on the color wheel (slice of the HSV cylindrical model). To do this, the HSV color must be converted into a tuple of cylindrical coordinates (r,θ,z) where r is the radial distance of the color from the center of the color wheel, θ is the angle of the color with respect to the horizontal axis and z is the height on the vertical axis.
r = S: the distance of the color is equal to the saturation of the color.
θ = H * π / 180: the color angle is equal to the hue multiplied by π / 180 (to convert degrees to radians).
z = V: The distance of the color is equal to the value of the color.
For the calculation of the average of the radii and heights it is enough to calculate an average weighted by the number of occurrences (pi) of each color which gives:

$$r_{avg} = \frac{\sum r_i \times p_i}{\sum pi} \qquad z_{avg} = \frac{\sum z_i \times p_i}{\sum pi}$$

To calculate θ_{mean}, first we must convert θ into Cartesian coordinates and calculate their mean:

$$x_{avg} = \frac{\sum \cos\theta_i \times p_i}{\sum pi} \qquad y_{avg} = \frac{\sum \sin\theta_i \times p_i}{\sum pi}$$

And in a second step we calculate the average angle and convert it into degrees:

$$\theta_{mean} = \tan^{-1}\left(\frac{y_{avg}}{x_{avg}}\right) \times \left(\frac{180}{\pi}\right)$$

Part 4: Determination of color or shade of gray.

*As seen in Figure 11(a) in Annex 8 the central and lower part of the cylinder defines the different shades of gray and the rest of the cylinder the colors with their respective shades. To classify the color of clothes the goal is to ignore the different shades of a color (managed by saturation and value) and to focus only on the "pure" color (hue). In Figure 12 in Annex 10, a cross-sectional representation of the cylindrical model is presented for a hue value (Hue) of 0, which corresponds to the shades of red. From this figure, it is possible to observe two areas bounded by a curve **C** illustrated in Figure 13 in Annex 10 and defined by:*

$$f(x) = \left(\frac{0.1}{x - 0.05}\right) + 0.05$$

A color with saturation values and values above the C curve will be considered color and classified according to hue ignoring saturation values. Conversely, when the saturation and value values located above the curve **C** the color will be classified according to its Value (saturation having few impacts here).

Part 5: Classification

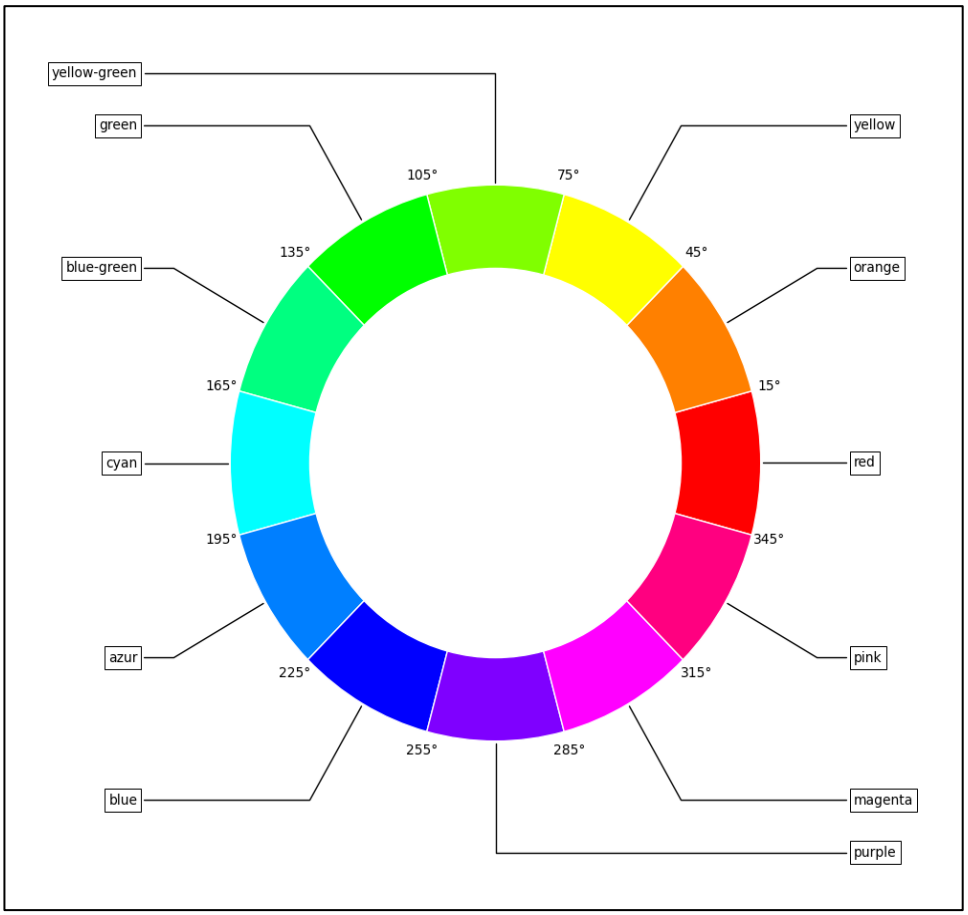


Figure 5 - Hue classification

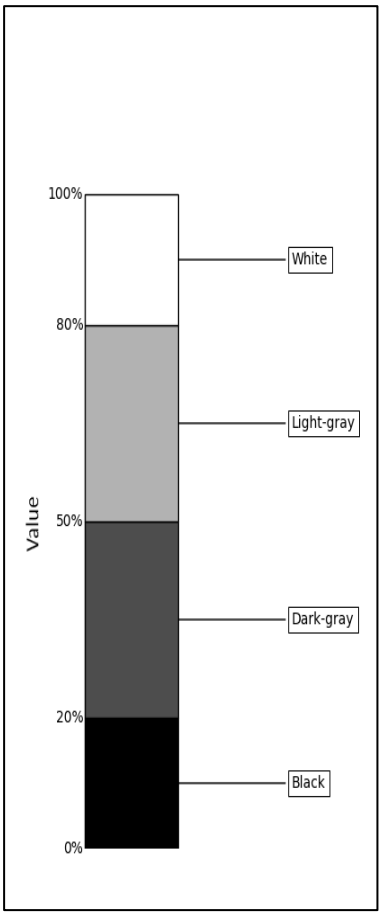


Figure 6 - Gray classification

If we conclude that the garment is colored, we use the information in *Figure 5* to determine the color of the garment among 12 colors represented. On the other hand, if we conclude that the garment is in a shade of gray, we will use the information in *Figure 6*. As far as programming is concerned, the HSV value retrieved is transmitted to a function which determines the position relative to the curve. This result is passed on as a parameter to a function responsible for garment color classification.

Conclusion

Chapter 5 describes the approach to detecting the main color of a garment. The first approach using the RGB model proved unsatisfactory. A second approach based on the HSV model was developed. The K-means algorithm is used to extract the values of the main colors of an image. These values are then converted to HSV values to allow the use of polar coordinates to find the average color. The method consists of calculating the average using the polar coordinates on the color wheel (slice of the HSV cylindrical model). The central and lower part of the cylinder defines the different shades of gray and the rest of the cylinder the colors with their respective shades. To classify the color of clothing, it is necessary to disregard the different shades of a color (managed by saturation and value) and to focus only on the "pure" color (hue). Once the color or shade of grey had been determined, a classification was made by reference figures (*Figure 5* and *Figure 6*).

Overall conclusion:

Conclusion of the project

In conclusion, this project explored various approaches for image element detection, segmentation, and color detection, highlighting the advantages and limitations of each method. The use of Grounding Dino proved practical and effective in detecting objects by exploiting linguistic information, while the Segment Anything tool enabled successful segmentation. The project also explored different color models such as RGB and HSV for color detection. The HSV model proved particularly useful for detecting colors in varying lighting conditions, separating color from brightness.

Overall, this project demonstrated the importance of exploring different methods and models to achieve optimal results in detecting elements in images. The techniques developed can be applied in various fields, such as pattern recognition, computer vision, color detection, object detection and image segmentation. By adding the part to detect bib numbers to the detection of the main colors of the garments the tool gains in reliability (both parts compensating for their respective defects).

Many of the difficulties were encountered at the beginning of the project partly because of the wide scope of the subject and the possibilities of multiple approaches and because of the need to understand and apprehend relatively complex concepts and concepts.

The prospects for improvement are the compilation of the different information of a runner (bib number & color of clothes) and a processing by a neural network to determine from a photo with a certain probability the identity of the runner by comparing the information of the runner with a table grouping the information of all the runners.

Personal conclusion

The subject was really very interesting and touched me directly practicing running myself. I also had the opportunity to do the Half Marathon of the Port of Las Palmas at the beginning of June and test the program on a photo of my race. In addition, this project has provided me with valuable technical knowledge. I was able to deepen my skills in the field of artificial intelligence and computer vision. In addition, I was able to gain hands-on experience with the Python language. From a more general point of view, during this internship I was able to visit the island of Gran Canaria which is full of beautiful landscapes. This internship also allowed me to develop bases in Spanish having done only 6 months of introductory courses at ESIREM before arriving.

To conclude this internship brought me a lot from a technical and cultural point of view.

Back to summary ↑

Bibliography

[1] J. Redmon, S. Dasvvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 08 06 2015. [Online]. Available: https://arxiv.org/abs/1506.02640.

[2] L. Shilong, . Z. Zhaoyang, R. Tianhe, L. Feng, Z. Hao, L. Chunyuan, Y. Jianwei, S. Hang, Z. Jun and Z. Lei, "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection," 09 03 2023. [Online]. Available: https://arxiv.org/abs/2303.05499.

[3] The DINOv2 team, "DINOv2: State-of-the-art computer vision models with self-supervised learning," 17 April 2023. [Online]. Available: https://ai.facebook.com/blog/dino-v2-computer-vision-self-supervised-learning/.

[4] “GroundingDINO.git,” [Online]. Available: https://github.com/IDEA-Research/GroundingDINO.git.

[5] "groundingdino_swint_ogc.pth," [Online]. Available: https://github.com/IDEA-Research/GroundingDINO/releases/download/v0.1.0-alpha/groundingdino_swint_ogc.pth.

[6] A. Kirillov, A. Berg, C. Rolland, E. Mintun, H. Mao, L. Gustafson, N. Ravi, P. Dollard, R. Girshick, S. Whitehead and W.-Y. Lo, "Segment Anything," 05 April 2023. [Online]. Available: https://ai.facebook.com/research/publications/segment-anything/.

[7] A. Hanbury, "Constructing Cylindrical Coordinate," Vienna University of Technology, Vienna, 2007. Available: http://muscle.ercim.eu/images/DocumentPDF/Hanbury_PRL.pdf

ANNEXES

Annex 1 - Glossary

1. Supervised learning

Neural networks learn by processing examples with a known input and result. They establish weighted probabilistic associations between the two, which are stored in their data structure. During training, the network compares its output (usually a prediction) to the expected output and calculates an error. It then adjusts the weights of its associations using a learning rule and the value of the error. These successive adjustments lead the network to produce outputs more and more like the expected outputs. Training ends when certain criteria are met.↑

2. Bounding box

It is a rectangular frame that surrounds an object or region of specific interest in an image or video.
The bounding box is defined by four coordinates: the x and y positions of the upper left corner of the box, as well as its width and height. These coordinates make it possible to precisely delimit the location of the object in the image.↑

3. Classes

In a convolutional neural network (CNN), classes refer to the different categories or labels that the model seeks to predict. When training a CNN for image classification, for example, the classes can correspond to specific objects present in the images.↑

4. Labelling

Data labeling is the process of identifying raw data (images, text files, videos, etc.) and assigning it one or more meaningful and informative labels to provide context, allowing a machine learning model to learn from it.↑

5. Self supervised training

Self-supervised learning is a machine learning method where a model seeks to learn patterns and structures from unlabeled data, without the need for explicit human supervision. Unlike supervised learning, where data is annotated by human-provided labels, self-supervised learning leverages the information present in the data itself to create learning tasks. The model attempts to predict some missing parts of the data or learn how to reconstruct input data from altered versions, which helps it extract useful representations and discover underlying knowledge without the need for external oversight. This approach allows machine learning models to gain knowledge more autonomously and generate rich, informative representations for various tasks.↑

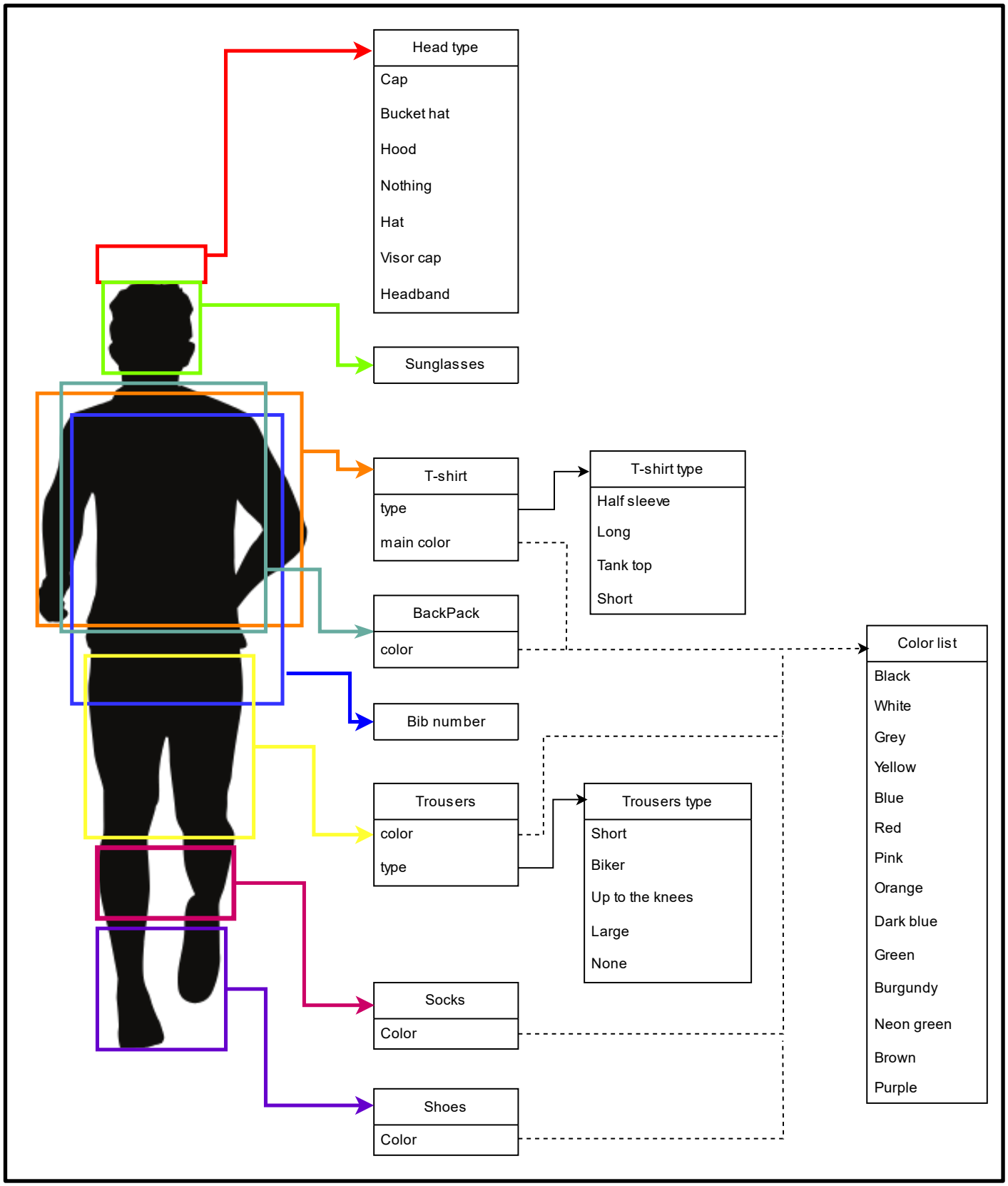
6. Transformer

An AI transform is a type of machine learning algorithm used for natural language processing and other tasks related to sequential information processing. It is based on a neural architecture that allows long-term dependencies to be captured and modeled in input data. Transformers use attention mechanisms to focus on the relevant parts of the input sequence, allowing them to efficiently process sequences of varying length. They are particularly effective for machine translation, text generation, and natural language comprehension tasks.↑

7. MAE

Masked Autoencoder for Distribution Estimation is a self-supervised technique used to pre-train the image encoder. Through the process of masking specific regions of the input image and training the autoencoder to reconstruct it, the model acquires the ability to identify and understand crucial features and patterns within the image.↑

Annex 2 - Diagram describing the different ways to describe a runner.



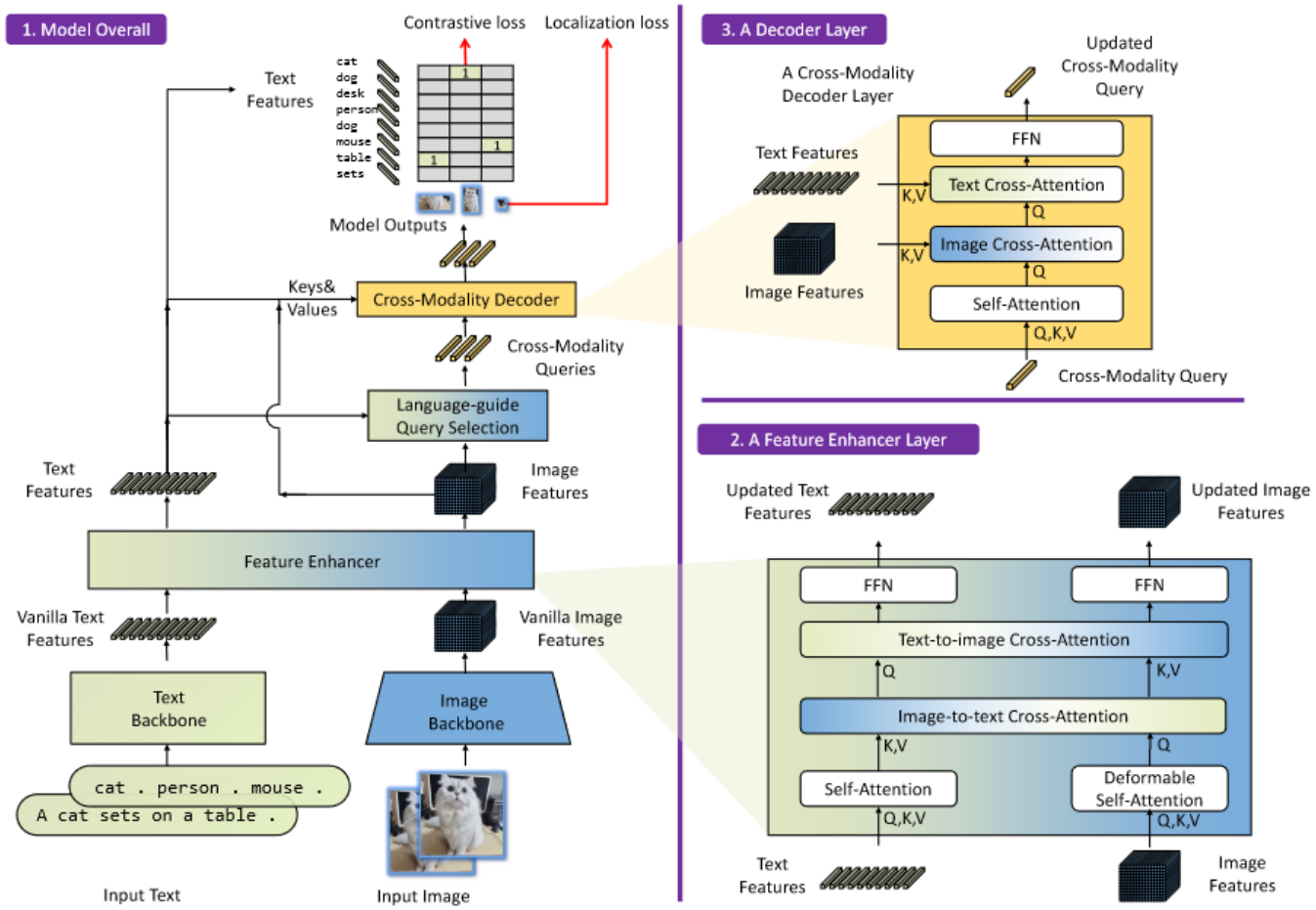
Go back up ↑

Annex 3 - List of COCO Dataset Classes

1: 'person',	17: 'dog',	33: 'sports ball',	49: 'sandwich',	65: 'mouse',
2: 'bicycle',	18: 'horse',	34: 'kite',	50: 'orange',	66: 'remote',
3: 'car',	19: 'sheep',	35: 'baseball bat',	51: 'broccoli',	67: 'keyboard',
4: 'motorcycle',	20: 'cow',	36: 'baseball glove',	52: 'carrot',	68: 'cell phone',
5: 'airplane',	21: 'elephant',	37: 'skateboard',	53: 'hot dog',	69: 'microwave',
6: 'bs',	22: 'bear',	38: 'surfboard',	54: 'pizza',	70: 'oven',
7: 'train',	23: 'zebra',	39: 'tennis racket',	55: 'donut',	71: 'toaster',
8: 'truck',	24: 'giraffe',	40: 'bottle',	56: 'cake',	72: 'sink',
9: 'boat',	25: 'backpack',	41: 'wine glass',	57: 'chair',	73: 'refrigerator',
10: 'traffic light',	26: 'umbrella',	42: 'cp',	58: 'couch',	74: 'book',
11: 'fire hydrant',	27: 'handbag',	43: 'fork',	59: 'potted plant',	75: 'clock',
12: 'stop sign',	28: 'tie',	44: 'knife',	60: 'bed',	76: 'vase',
13: 'parking meter',	29: 'suitcase',	45: 'spoon',	61: 'dining table',	77: 'scissors',
14: 'bench',	30: 'frisbee',	46: 'bowl',	62: 'toilet',	78: 'teddy bear',
15: 'bird',	31: 'skis',	47: 'banana',	63: 'tv',	79: 'hair drier',
16: 'cat',	32: 'snowboard',	48: 'apple',	64: 'laptop',	80: ' toothbrush'

Go back up ↑

Annex 4 - Grounding Dino Operating Diagram



The overall framework, the feature enhancement layer and the decoding layer are show in blocks 1, 2 and 3 respectively, block 2 and block 3, respectively.
Figure retrieved from article [2]

Go back up ↑

Annex 5 - Colorized Figure 4 masks Figure 4

t shirt

sunglasses

trousers

bib numbers

cap

shoe

backpack

sock

Go back up ↑

Annex 6 - Operating principle of the K-means algorithm

K-means is an unsupervised method used to group a dataset based on their similarities. Its operation can be described as follows:

1. Initialization: The algorithm initially randomly selects K points in the data space, called "centroids", which will serve as initial centers for clusters.

2. Assigning points to the nearest cluster: Each data point is then assigned to the cluster represented by the nearest centroid. The distance between a data point and a centroid is usually measured using the Euclidean distance.

3. Centroid readjustment: Once all points have been assigned to clusters, the centroids in each cluster are recalculated by averaging the positions of the points assigned to them. This moves the centroids to the average positions of their respective clusters.

4. Repeating steps 2 and 3: The steps of assigning points to clusters and readjusting centroids are repeated until a convergence condition is reached. This condition can be defined by a fixed number of iterations, when centroids no longer move significantly or when the assignment of points to clusters no longer changes.

5. Result: Once convergence is achieved, the end clusters are formed, and the data points are grouped according to their similarity to the corresponding centroids.

Go back up ↑

Annex 7 - Example of color detection in RGB

Figure 7 - Image of a TGC runner

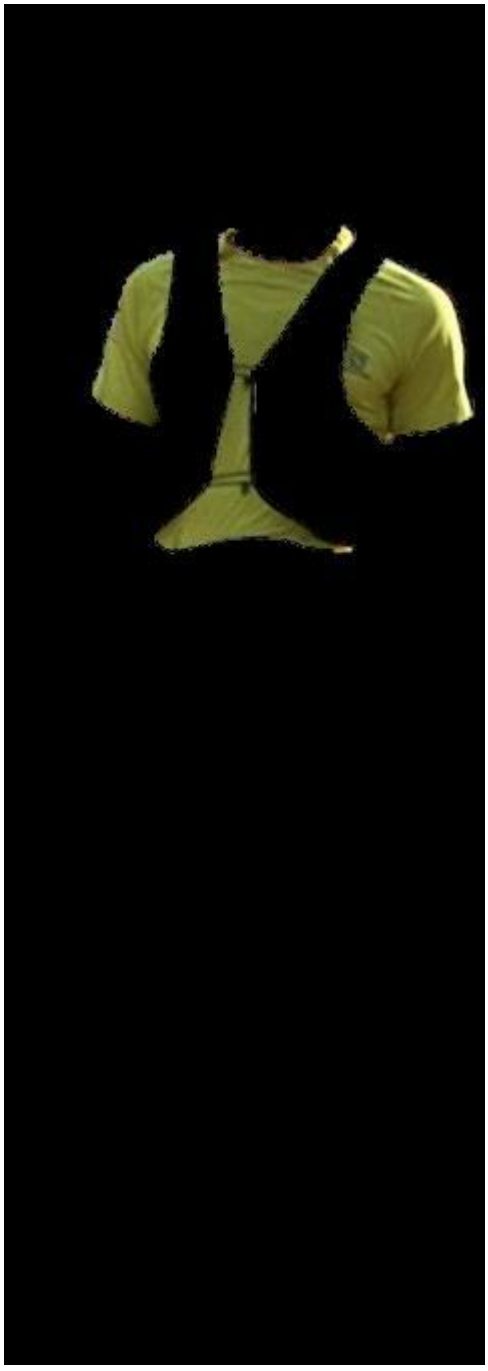


Figure 8 - Segmented & colored t-shirt mask from Figure 7

Using the RGB color detector in Figure 2 the program gives us "Brown" as the detected color.
This is due to the fact of the exposure of the image, in addition the runner has the sun in the back which shades his t-shirt and falsifies the detection.

Go back up ↑

Annex 8 - Representation of color models

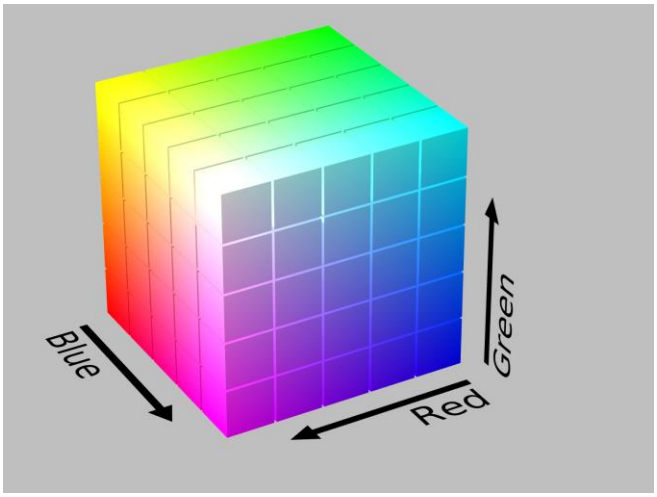


Figure 9 - RGB color model mapped to a cube.

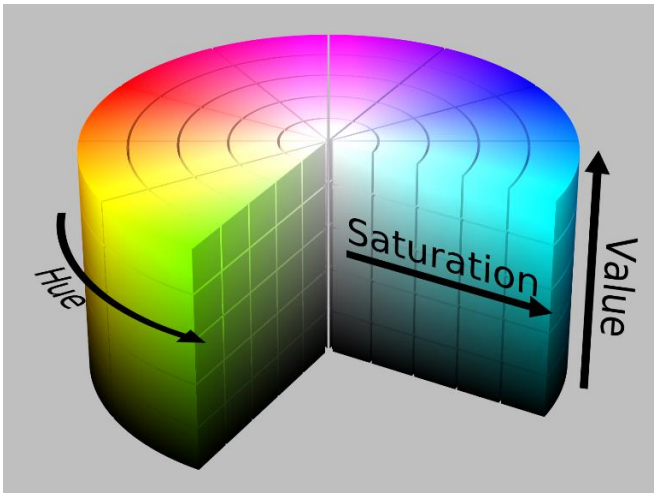


Figure 10 - HSV cylinder

↑

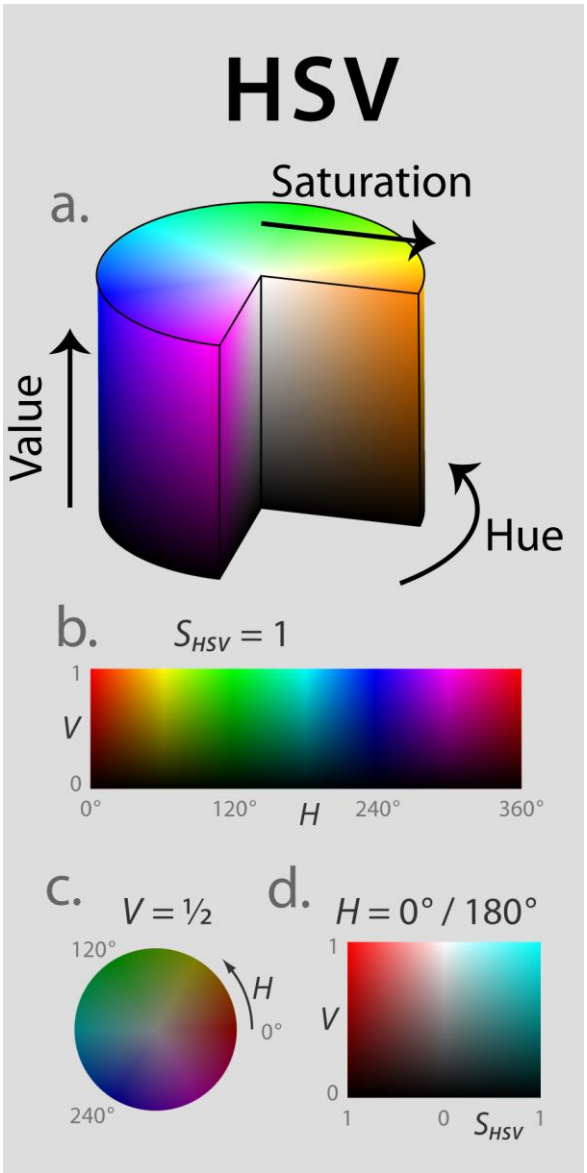


Figure 11 - Detailed HSV cylinder

Cut-away 3D HSV model (a)
Two-dimensional plots showing two of model's three parameters at once, holding the other constant: cylindrical shells of constant saturation (b)
Horizontal cross-section of constant HSV value, in this case the slices halfway down of cylinder (c)
Rectangular vertical cross-sections (d) of constant hue, in this case of hues 0° red and its complement 180° cyan (d) ↑

Figures 9 to 11 retrieved from https://en.wikipedia.org/wiki/HSL_and_HSV

Go back up ↑

Annex 9 - Equations for converting RGB to HSV

1. Normalize RGB values by dividing them by 255 to range from 0 to 1.

$$R' = \frac{R}{255} \quad G' = \frac{G}{255} \quad B' = \frac{B}{255}$$

2. Calculate the maximum and minimum values among R', G', and B'.

$$C_{\max} = \max(R', G', B')$$
$$C_{\min} = \min(R', G', B')$$

3. Calculate the difference between the maximum and minimum values:

$$\Delta = C_{\max} - C_{\min}$$

4. Calculate the hue component H:

$$If(\Delta = 0), then(H = 0)\ (undefinedhue)$$
$$Otherwise, if\ C_{\max} = R', then\ H = 60 \times \left(\frac{G' - B'}{\Delta}\right) mod\ 6$$
$$Otherwise, if\ C_{\max} = G', then\ H = 60 \times \left(\frac{B' - R'}{\Delta} + 2\right)$$
$$Otherwise, if C_{\max} = B', then\ H = 60 \times \left(\frac{R' - G'}{\Delta} + 4\right)$$

5. Calculate the saturation component S:

$$If(C_{\max} = 0), then(S = 0)$$
$$Otherwise, (S = \frac{\Delta}{C_{\max}})$$

6. Calculate the value component V:

$$V = C_{\max}$$

Go back up ↑

Annex 10 - Cross-sections of the HSV cylindrical model

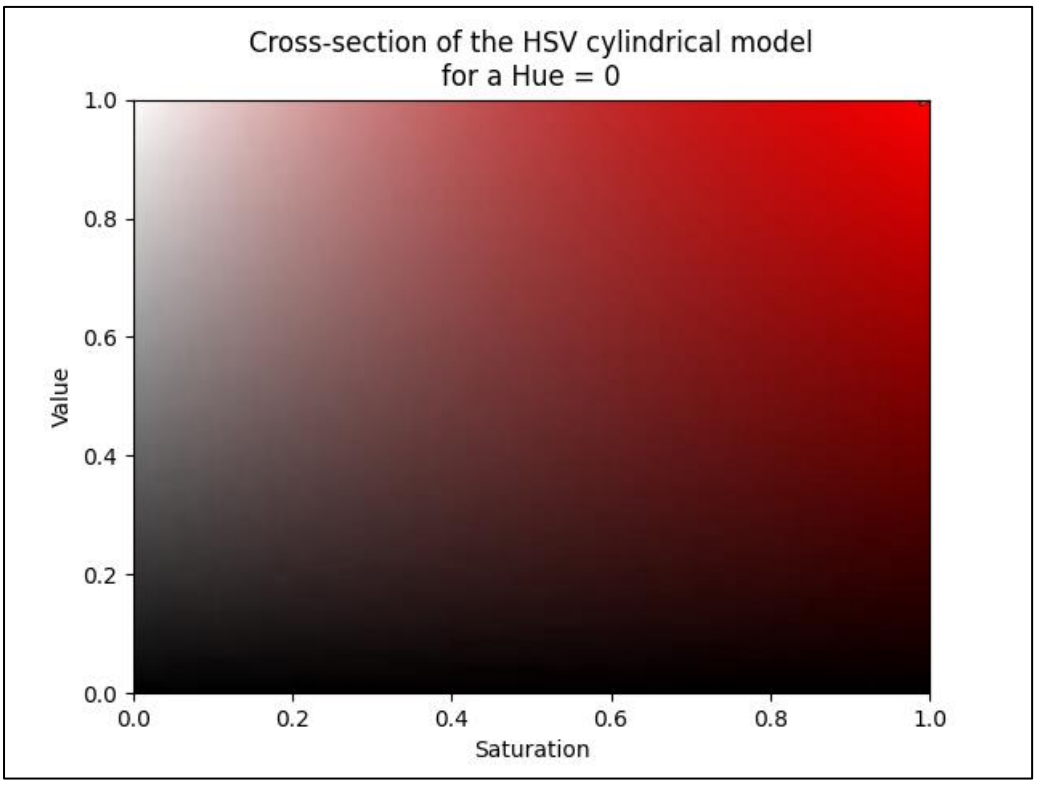


Figure 12 - Cross-section of the HSV cylindrical model

Go back up ↑

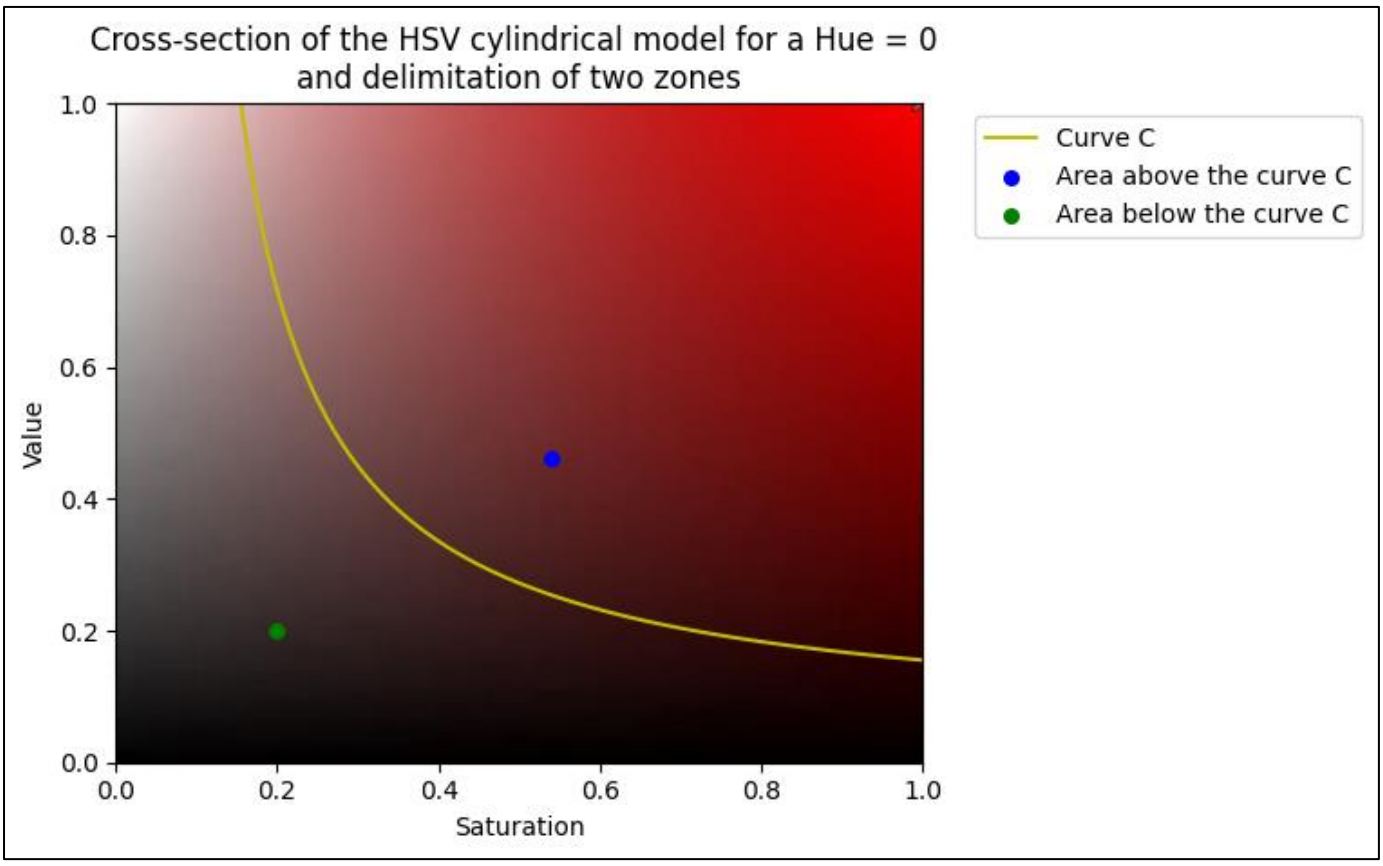


*Figure 13 - Cross-section of the HSV cylindrical model and delimitation of two zones by the curve **C***

Go back up ↑