Project in Computer Science and Engineering,

First Cycle 15 credits

Optimizing Object Detection in Autonomous Vehicles Using Grayscale Computer Vision Models

DIDRIK LIU & ALEXANDRE MOCH

Optimizing Object Detection in Autonomous Vehicles Using Grayscale Computer Vision Models

ALEXANDRE MOCH & DIDRIK LIU

Degree Project in Computer Science and Engineering, First Cycle 15 credits
Date: May 21, 2024
Supervisor: Erik Fransén
Examiner: Pawel Herman
Swedish title: Optimering av objektdetektering i autonoma fordon med hjälp av gråskaliga datorsynmodeller
School of Electrical Engineering and Computer Science

Abstract

This study explores the implementation of grayscale computer vision models in autonomous vehicles. It investigates whether machine learning models trained on grayscale images can maintain classification accuracy while potentially increasing frame rates when compared to using RGB-images. The study also analyzes the trade-offs between accuracy and frame rate, possible thanks to the advantages of monochrome cameras that offer higher frame rates and better low-light performance than color cameras. In particular, the study tests this usage in the Formula Student competitions. The research uses the YOLOv8 architecture due to its effectiveness and common usage in Formula Student, comparing its performance on grayscale images with color images.

Several grayscaling methods were compared to determine their impact on the object detection accuracy of the YOLOv8 model. The results show that certain grayscale methods can achieve comparable accuracy to color-based models, suggesting a potential for enhancing the speed of autonomous vehicle decision-making processes with minimal loss in reliability.

The findings can be used for the broader field of autonomous vehicles, particularly in optimizing vision systems for speed and accuracy in diverse driving conditions. The study's approach and results contribute to the ongoing development of more efficient and adaptable driverless technologies.

Sammanfattning

Denna studie utforskar användningen av gråskaliga datorseendemodeller i autonoma fordon. Den undersöker om maskininlärningsmodeller som tränats på gråskalade bilder kan bibehålla noggrannheten i klassificering samtidigt som de potentiellt ökar bildhastigheten jämfört med användning av RGB-bilder. Studien analyserar också avvägningarna mellan noggrannhet och bildhastighet, möjlig tack vare fördelarna med monokroma kameror som erbjuder högre bildhastigheter än färgkameror och bättre prestanda vid svagt ljus än färgkameror. Specifikt testas denna användning i Formula Student-tävlingar. Forskningen använder YOLOv8-arkitekturen på grund av dennes effektivitet och vanliga användning i Formula Student, och jämför dess prestanda på gråskalebilder med färgbilder.

Flera metoder för att skapa gråskalebilder jämfördes för att bestämma deras inverkan på nogrannheten av detektionen av objekt hos YOLOv8-modellen. Resultaten visar att vissa gråskalemetoder kan uppnå jämförbar noggrannhet med färgbaserade modeller, vilket tyder på en potential för att förbättra hastigheten i autonoma fordons beslutsfattande processer med minimal förlust av tillförlitlighet.

Resultaten kan användas för fältet av autonoma fordon, särskilt för att optimera synsystem för hastighet och noggrannhet under varierande körförhållanden. Studiens tillvägagångssätt och resultat bidrar till den pågående utvecklingen av mer effektiva och anpassningsbara förarlösa teknologier.

Contents

1 Introduction

The birth of autonomous vehicles marks a transformative era in the automotive industry, promising efficiency and safety in transport with minimal effort from the passenger. Most systems today implement a combination of LiDAR technology and computer vision to determine object type, distance and other relevant parameters for autonomous driving. LiDAR techonology is a "remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to the Earth" [1]. The integration of autonomous vehicle technology not only revolutionizes personal transport but also paves the way for innovations in competitive engineering environments, such as Formula Student (FS) competitions. A pivotal challenge within autonomous vehicle development, particularly highlighted in the high-stakes environment of FS competitions, is the accurate and rapid detection of objects. This capability is crucial for navigational decision-making, where speed and precision can determine the outcome of a race. The ability to rapidly and accurately detect cones can be translated into a wider range of real world contexts, such as in near-accident or low image-fidelity scenarios.

1.1 Purpose

This research is primarily motivated by the broader ambition to expedite decision-making processes in commercial autonomous vehicles, particularly in scenarios demanding rapid response times.

Additionally, a higher frame rate may contribute to a better perception of a traffic situation through a combination of different angles solving overlapping object issues. It also gives the model the ability to keep up with the video stream and avoids lag. [2]

In the context of Formula Student competitions, the enhancement of cone detection models holds the promise of significantly improving performance by enabling faster navigational decisions under the high-pressure conditions of a race. This research look to validate the concept that grayscale-based object detection can accelerate decision-making in autonomous vehicles with minimal trade off in reliability, offering insights that can be pivotal for both the advancement of Formula Student Driverless technologies and the broader application in commercial autonomous vehicle operations.

1.2 Problem Statement

- To what extent can object detection machine learning models trained on grayscaled imaging maintain an accuracy level with a higher frame rate that can compete with current Formula Student Driverless models trained on color images?

1.3 Scope

In the pursuit of refining object detection within Driverless systems for Formula Student (FS) competitions, this study adopts YOLOv8 (You Only Look Once version 8) due to its recognized accuracy and established utility in FS car systems. The core of the analysis explores the trade-off between accuracy and image properties and machine learning model properties from grayscaled images. Several key parameters are delineated for comparison: the impact of varying levels of grayscale on accuracy, the relationship between image resolution and frame rate on model performance, and the effects of hyperparameter optimization through grid search techniques. This potential rise in efficiency was noticed while studying the effects of monochrome cameras within the Formula Student competitions. The use of monochrome cameras should allow for a rise in performance in some key aspects of the Driverless architecture.

Building upon these comparative metrics, the study aims to devise bespoke indexes. These indexes will serve to evaluate the efficiency of YOLOv8 in processing grayscale images against its performance with the standard color-based approach, and to formulate a comprehensive measure that encapsulates the balance between processing speed and detection accuracy. This approach seeks to identify an optimal configuration that maximizes both speed and accuracy, pinpointing the most effective model configuration for application in the dynamic and demanding context of Formula Student racing. Through this methodology, the research contributes a tailored solution that not only advances the capabilities of autonomous vehicles in FS competitions but also informs broader applications in autonomous vehicle technology.

2 Background

Full self-driving (FSD) technology is a rapidly evolving field, aiming to revolutionize transportation by enabling vehicles to operate without human intervention. This technology integrates various sensors and algorithms to perceive and navigate the environment autonomously, ensuring safety and efficiency. For most developers, both LiDAR and cameras are crucial components, each offering unique benefits for accurate object detection and decision-making.

In the realm of competitive engineering, Formula Student (FS) is on the cutting edge of development of self-driving technology. The competition challenges universities worldwide to design, build, and race electric autonomous vehicles. Within this competitive landscape, the Driverless team at the Royal Institute of Technology (KTHFS) is tasked with the autonomous conversion of these vehicles utilizing many different technologies in both hardware and software. Historically, the team has used Lidar technology exclusively for object detection tasks. However, recent initiatives have aimed at integrating camera systems to enhance this capability. Object detection within this context primarily focuses on the detection of cones, utilizing their color and positioning to inform the vehicle's navigational decisions. This integration seeks to advance the autonomous vehicle's environmental interaction and decision-making processes.

2.1 Related work

The related work section has been split up into five different parts. Firstly, there are earlier works regarding the quality of different grayscaling methods. Secondly, work has previously been done within object detection with grayscaled images. These have mostly been for very precise uses. The third will mention previous work on the Formula Student Driverless System from a few different universities. This will serve as an overview of the architecture of the whole system where this work could be implemented. For the final two parts, the YOLOv8 architecture, its performance and its metrics will be mentioned and explained.

2.1.1 Grayscaling methods

To begin with, some work has been done comparing different grayscaling methods and whether or not the method that is used will change the results in image recognition [3]. In this paper, they compare 13 different grayscaling methods to demonstrate that all grayscaling methods do not work equally well. Some of these methods rely on gamma-corrected channels that use this formula: $\Gamma(t) = t' = t^{\frac{1}{2.2}}$[3]. The Intensity' method consists of Γ(Intensity) with the Intensity being is $\frac{1}{3}(R + G + B)$. The Gleam method on the other hand is defined as $\frac{1}{3}(\Gamma(R) + \Gamma(G) + \Gamma(B)) = \frac{1}{3}(R' + G' + B')$. Following the experiments, a mean rank is produced through a comparison of results based

on different databases. This mean rank indicates that the Gleam and Intensity' methods achieve the best results overall by quite a significant margin compared to the other methods [3].

2.1.2 Grayscaled Object Detection

Some research has also been done in regards to the implementation of grayscaled images for object detection. In the paper, the goal is to make an object detection model more reliable by combining the results of the object detection model on colored images with the ones from grayscaled images. The aim is for these model to compensate for the discrepancies that the other images may lead to [4]. This research was made with the intention of applying this to a robotics competition. The results are therefore for the accuracy of the object detection model when recognising robots. The authors do however mention that: "the approach can be easily be extended to detect a variety of different objects" [4].

2.1.3 Formula Student Driverless System

Finally, another aspect that is necessary to look at in regards to related work, is the overall architecture of the Formula Student Driverless System.

There have been multiple papers regarding this and it is a domain prone to changes on a yearly basis.

The AMZ Racing team presents their whole driverless architecture in a paper. This includes "environment perception, state estimation, SLAM, planning, and vehicle dynamics control" [5]. Through this, it is very clear that the design of a driverless system is extensive and consists of much more than the perception of the cones.

Another important work regarding the Formula Student Driverless teams is the creation of the Formula Student Objects in Context (FSOCO) dataset [6]. This dataset contains a collection of images with cones submitted by 18 Formula Student teams. It contains "11,572 annotated images with bounding boxes (on average 19.09 boxes per image) 1,517 annotated images with segmentation masks (on average 15.81 masks per image)" [6].

2.1.4 YOLOv8 Architecture

The YOLOv8 model is a real time object detection and segmentation model made by Ultralytics. The YOLO models are known for their speed and for their accuracy.

Previous object detection models were built on the sliding window approaches. This method was both expensive and slow. The difference with the YOLO models is that they treat object detection like a single regression problem. The 8th version of their model (YOLOv8) has been built on the successful parts

of their previous models (YOLOv5, YOLOv6, YOLOv7) and has showed a lot of promise and yielded very good results in both speed and accuracy. [7]

It consists of 5 different models that can be used for different use cases. Each model is of different size. A smaller size in model leads to a higher speed and smaller storage usage but leads to a reduction in accuracy.

Model	**Size (pixels)**	**mAP-val 50-95**	**Speed CPU ONNX (ms)**	**Speed A100 TensorRT (ms)**	**params (M)**	**FLOPs (B)**
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Table 1: Performance metrics of YOLOv8 models [8]

As shown in Table 1, the YOLOv8n model seems built with the aim to achieve fast detection with a fairly good accuracy while the other, larger models seem to favour accuracy at the expense of speed. The amount of epochs used in previous iterations of the KTH Formula Student Driverless System have varied a lot. The range of the amount of epochs is typically between 50-300 epochs. This range has seemed to yield the best accuracy without leading to overfitting.

The model itself is a Convolutional Neural Network that consists of three main parts: the Backbone, the Neck and the Head. The Backbone is in charge of extracting the main features inside an image. It finds simple patterns within the image. The Neck retrieves different feature maps from different stages within the Backbone. It then combines them through fusion (for detecting different object sizes), contextual information (to give a broader context to the image) and Reduction of "spatial resolution and dimensionality of resources" which increases speed but can make the model less accurate. The Head is the final part of the network. It outputs bounding boxes as well as a confidence for that box. It also assign the bounding boxes to a class with the hopes that it will classify it correctly. [9]

2.1.5 YOLOv8 Metrics

Mean Average Precision (mAP): "mAP extends the concept of AP by calculating the average AP values across multiple object classes. This is useful in

multi-class object detection scenarios to provide a comprehensive evaluation of the model's performance." [10]

Precision and Recall: "Precision quantifies the proportion of true positives among all positive predictions, assessing the model's capability to avoid false positives. On the other hand, Recall calculates the proportion of true positives among all actual positives, measuring the model's ability to detect all instances of a class." [10]

F1 Score: "The F1 Score is the harmonic mean of precision and recall, providing a balanced assessment of a model's performance while considering both false positives and false negatives." [10]

More specifically, the metrics that will be looked at are mAP50, mAP50-95, F1 Score, Precision and Recall.

The mAP50 indicates the model's accuracy through it's mean average precision when predicting certain classes "calculated at an intersection over union (IoU) threshold of 0.50" [10]

The mAP50-95 is similar to the mAP50 but will look at the accuracy of different predictions for thresholds ranging from 50 to 95.

2.2 Monochrome Cameras and the difference with color cameras

In the KTH Formula Student Driverless team, a color camera is used to detect the cones. The camera has a frame rate of 60 fps and resolution of 2448 x 2048.

Color cameras, like the model being used, work in the following way:

Light is captured through sensors that are made up of pixels. On a color camera, a Color Filter Array is put on top of sensors to detect the desired wavelength [11].

A Color Filter Array is "an arrangement of color filters on top of a camera sensor array that allows light only of a particular wavelength to fall on a single pixel" [11].

The most popular filter is the Bayer filter. It is half green, a quarter blue and a quarter red [12], as shown in Figure 1.

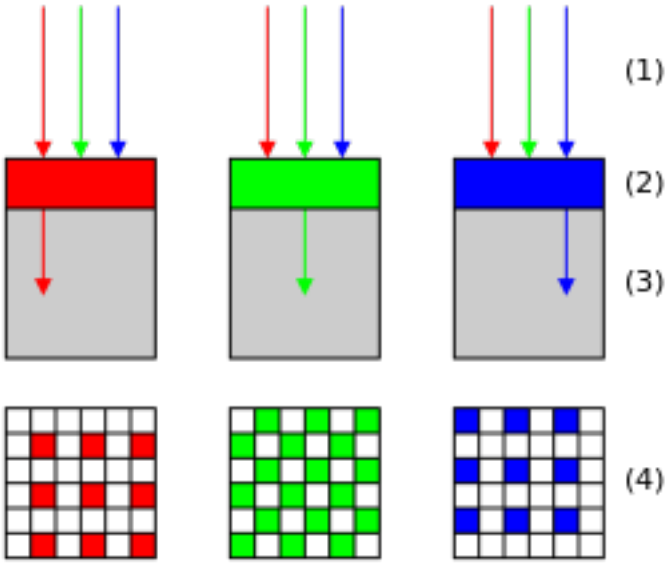

Figure 1: Bayer Filter by Cburnett [13]

Monochrome cameras, on the other hand, work a bit differently. They capture all the light on each pixel, regardless of the color. Give that it takes in red, green and blue channels as one it therefore takes 3 times as much light. [14]

Some of the advantages of using monochrome cameras are:

"1. Monochrome cameras perform better in low lighting conditions
2. Monochrome sensors have intrinsically higher frame rates
3. Monochrome algorithms are well-tuned" [14]

Point 1 and 2 are two extremely useful aspects for Formula Student applications given the speed of the vehicle and varying visibility.

3 Methods

3.1 Dataset

The dataset chosen for this work is the FSOCO dataset. It offers data regarding the cones in various forms. The images are annotated in 5 classes: Orange Cone, Yellow Cone, Blue Cone, Large Orange Cone and Other Cone. The dataset is extensive and offers pictures from cones in different settings. There are pictures in different weather conditions, indoors, outdoors as well as different luminosity levels. The aim is that by using a dataset with such varying training data, the model will avoid overfitting to a certain condition and be more adapted to varying conditions.

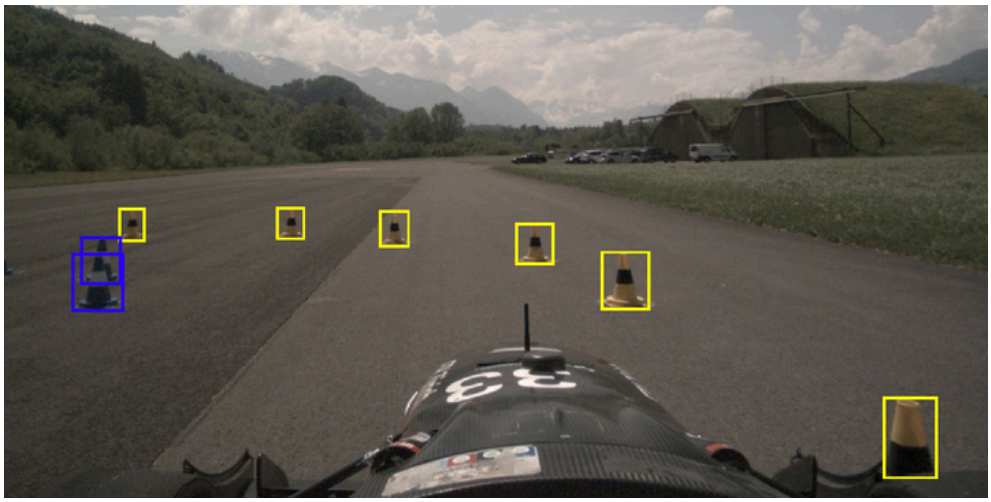

Figure 2: Example of image annotations

Class	Example
Blue	
Yellow	
Small orange	
Large orange	
Other cone (not rules compliant)	

Figure 3: All classes of cones

3.2 Computing Platform – Google Cloud Platform

Google Cloud Platform is chosen as the main laboratory platform for this study, specifically a VertexAI VM instance. The main reason is for the performance boost given by Nvidia L4 graphics cards available in the VM, as well as its compatibility with PyTorch that gives Ultralytics the ability to train the model on GPU, instead of solely on the CPU. The specific hardware used to conduct this study is 1x Nvidia L4 graphics card and 16 cores virtual CPU.

Additionally, Google Cloud is a partner of the KTH Formula Student team which enabled us to have access to GPUs with much higher computational power than any machines that we had access to elsewhere. Thanks to the free use of these services, a large amount of different models could be compared in order to try and achieve the highest precision.

3.3 Preprocessing

3.3.1 Grayscaling

Given that there are different grayscaling methods that yield different results, the different values of Red, Green and Blue channels need to be analysed in order to determine which method yields the best accuracy.

Different methods are tested:

The first method to be tested will be the Lambda function. In this case, the Lambda function serves as a reducer of the color depth in the image. The factor used in this case is 8. This means that the the colors will be split into 32 buckets.

```
def lambda_gray(img):
    img.convert('L')
    img = img.point(lambda x: int(x / 8) * 8)
    return img
```

Secondly, the Intensity and Gleam method will be compared and used. In terms of pixel values, Intensity and Gleam produce very different results. Since is a concave function, Jensen's inequality [7] implies that Gleam will never produce a representation with values greater than gamma corrected Intensity, and it follows the inequality: $\text{Intensity}(t) < \text{Gleam}(t) < \Gamma(\text{Intensity}(t))$ where $\Gamma(\alpha) = \alpha' = \alpha^{\frac{1}{2.2}}$[3] and $\text{Intensity} = \frac{1}{3}(R + G + B)$.

The Intensity method is implemented below.

```
def intensity_grayscale(img):
    img = img.convert('RGB')
    img_norm = np.asarray(img) / 255.0
    mean = np.mean(img, axis = 2)
    res = Image.fromarray((mean * 255).astype('uint8'))
    return res
```

The Gleam method for grayscaling will also be tested. It is defined as $\frac{1}{3}(\Gamma(R)+\Gamma(G)+\Gamma(B)) = \frac{1}{3}(R'+G'+B')$, where $\Gamma(t)$ is the same function as defined above.

```
def gleam(img):
    img = img.convert('RGB')
    img_norm = np.asarray(img) / 255.0
    img_gamma = np.power(img_norm, gamma)
    mean = np.mean(img_gamma, axis=2)
    gleam_gray = Image.fromarray((mean * 255).astype('uint8))
    return gleam_gray
```

Another method that will be tested is the Desaturation method. It is defined as Gray $= \frac{\max(R,G,B)+\min(R,G,B)}{2}$. This method averages out the R, G and B values of every pixel.

```
def desat(img):
    img = img.convert('RGB')
    img_norm = np.asarray(img) / 255.0
    values = (np.max(img_norm)+np.min(img_norm))/2
    img_desat = Image.fromarray((values*255).astype('uint8))
    return img_desat
```

Last but not least, an incremental method will be used to determine the optimal grayscale transformation for object detection in autonomous vehicles using the YOLOv8 model. An exhaustive search will be conducted by adjusting the weights assigned to the red, green, and blue channels, ultimately identifying the combination that maximized detection accuracy. This process involved systematically testing different channel weight combinations, initially using a broad range and then narrowing down to finer increments where significant improvements were observed.

To refine the process, an exhaustive search was conducted by adjusting the weights assigned to the red, green, and blue channels, ultimately identifying the combination that maximized detection accuracy. This process involved systematically testing different channel weight combinations, initially using a broad range and then narrowing down to finer increments where significant improvements were observed. The best-performing grayscale weights (R: 0.4, G: 0.6, B: 0.0) were selected based on their superior mAP scores.

These grayscaled images were then used to train the YOLOv8n model, which was assessed on various metrics such as mAP50, mAP50-95, Precision, Recall, and F1 Score to ensure the model's effectiveness in detecting cones in a Formula Student competition environment.

The actual code implementations are optimized for cloud computing and multi-core executions for faster results in grayscaling, however the logic stays the same for all grayscaling methods.

3.3.2 Annotations

While the FSOCO dataset is very extensive, the annotations with it are not compatible with the Ultralytics YOLOv8 annotations. The reason behind this being the formatting difference: YOLOv8 expects a txt file while FSOCO contains JSON files.

Before being able to train the model, a reformatting of all the annotations is necessary.

This will be done by transforming the JSON annotations into the format needed for YOLOv8 which is:

[class] [x-coordinate] [y-coordinate] [width] [height]

3.4 Model

The model that will be used for this project is the YOLOv8n model.

YOLOv8n was chosen given that this work does not need a more complex model. In a competition like Formula Student, speed is by far the most important factor. A more complex version of YOLOv8 would lead to a slower model which would defeat the purpose of making the current model faster with grayscaled images.

Given previous work it is also clear to see that the amount of epochs for each training should be chosen wisely. It seems that given previous work the range of epochs the model should be trained on is between 50 to 300 epochs. Given limitations in computational power and time the training will range from 20 to 100 epochs and will be adapted depending on how fast the model converges.

Different parameters will be tested on the model to see which yield the best results.

Another aspect that will be evaluated is the relevance of the pretrained model 'yolov8n.pt'. A model will be trained in the same exact conditions without pretraining and another model will be tested with the pretrained model to see whether pretraining has an effect between the "COCO" dataset and the FSOCO dataset. For that reason the model will also be deterministic to fit with the criteria of being in the same conditions as the KTH FS Driverless model.

The COCO dataset is defined as: "a large-scale object detection, segmentation, and captioning dataset."[15] It contains a lot of different objects within it to enable it to be a suitable dataset for training many models on detection of different objects.

3.5 Analysis and Metrics

As shown earlier in the previous work section, YOLOv8 is a determinstic model. This means that both the current model at KTH Formula Student Driverless and the model generated with the grayscaled images will not vary for different trainings as long as the settings and the training and validation data remain the same.

The best way to compare the two models will therefore be through other metrics than Analysis of Variance, t-tests and Kolmogorov Smirnov tests. This because these tests rely on variance and these determinstic models do not yield any variance.

The metrics that will be used to determine the efficiency of the models are the mAP50, mAP50-95, Confidence, Recall, F1 Score and Precision.

4 Results

4.1 Diverse grayscaling methods

Method	Description	Compatibility[1]	mAP50[2]
Lambda	reduce the color depth by rounding to nearest multiple	Post Processing	0.383
Intensity'	$\Gamma\left(\frac{1}{3}(R+G+B)\right)$	Camera config	0.390
Gleam	$\frac{1}{3}(\Gamma(R)+\Gamma(G)+\Gamma(B))$	Post Processing	0.418
Desaturation	$\frac{\max(R,G,B)+\min(R,G,B)}{2}$	Camera Config	0.265
Averaging	$\frac{R+B+G}{3}$	Camera Config	0.397

Table 2: All grayscaling methods tested

4.2 Parameters

Following the results in Section 4.1, a choice was made to look at a version of averaging with different weights for each of the Red, Green and Blue channels. This was to see whether a change of weights could achieve an mAP50 value better than the one the Gleam and Averaging methods generated.

The aim was to find values that separated the blue and yellow cones as much as possible.

Firstly, every combination of the three values were computed (at a resolution of 0.1).

[1]Camera config: available in specific cameras or settings
Post processing: available through a script after image is taken

[2]Result after 5 epochs and using a small sample dataset

R	G	B	mAP50
0.3	0.7	0.0	0.445
0.3	0.6	0.1	0.445
0.3	0.5	0.2	0.445
0.4	0.6	0.0	0.453
0.4	0.5	0.1	0.453
0.4	0.4	0.2	0.453
0.5	0.5	0.0	0.431
0.5	0.4	0.1	0.431
0.5	0.3	0.2	0.431
0.6	0.4	0.0	0.475
0.6	0.3	0.1	0.475
0.6	0.2	0.2	0.475

Table 3: All RGB color combinations and their mAP 50 values

However, after noticing that different values of Green and Blue had no effect on the accuracy of the model, a choice was made to mainly look at different weights for the Red values but at a higher resolution.

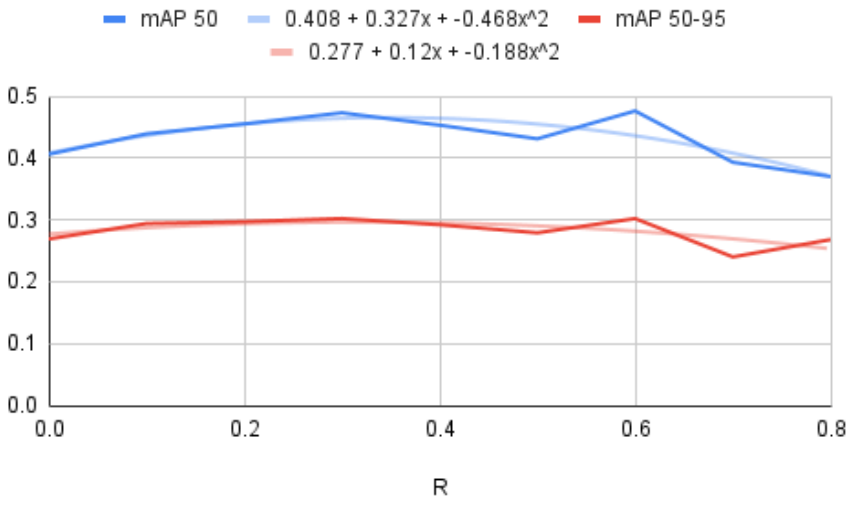

Figure 4: mAP50 based on level of Red grayscaling (Range: 0 to 0.8, Resolution: 0.1)

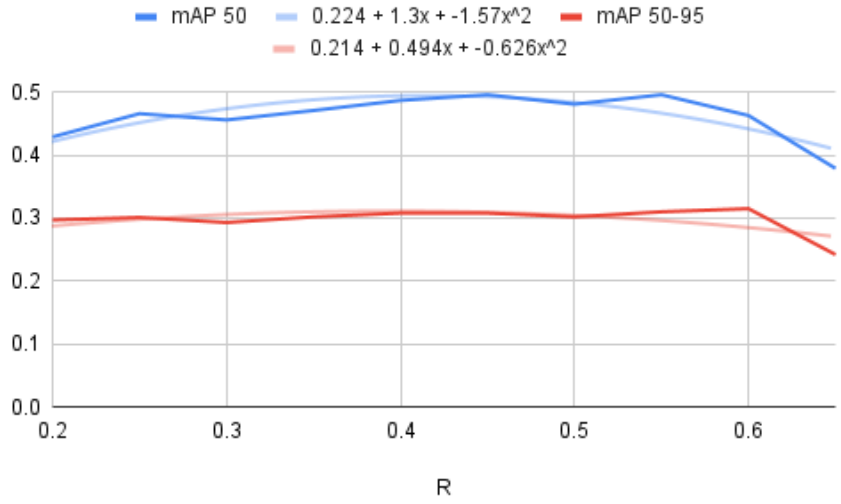

Figure 5: mAP50 based on level of Red grayscaling (Range: 0.2 to 0.7, Resolution: 0.05)

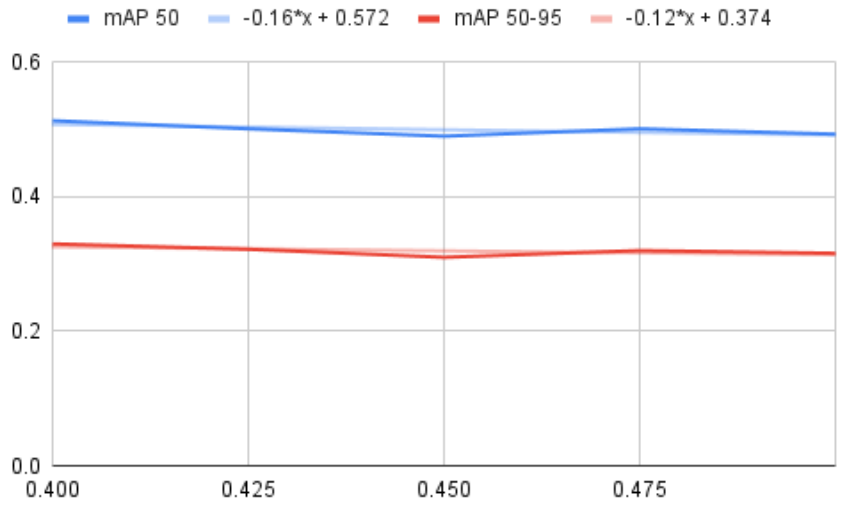

Figure 6: mAP50 based on level of Red grayscaling (Range: 0.4 to 0.5, Resolution: 0.025)

4.3 Grayscaled images

The final weights for the grayscaling were chosen as R: 0.4, G: 0.6 and B: 0 following the results from 4.2.

The grayscaling algorithm takes an image from the dataset as an input.

Figure 7: Example image

And converts it into a gray scaled image with the weights listed above in order to try to separate the cones from eachother as much as possible.

Figure 8: Example grayscaled image

4.4 Final model with optimal grayscaling weights

After training YOLOv8n on the images that follow the grayscale ratio of:

R = 0.6; G = 0.4; B = 0.0

After training the YOLOv8n model with these settings: Epochs=40, batch=16, image_size=1024, Pretrained=True, Dropout=0.2, Confidence=0.2

Whether the model is pretrained or not, the end result is exactly the same. Therefore the results below are exactly the same whether or not the model is pretrained on the "COCO" dataset.

The model's metrics are shown below. The mAP50 is approximately at 0.75 and the mAP50-95 is around 0.55.

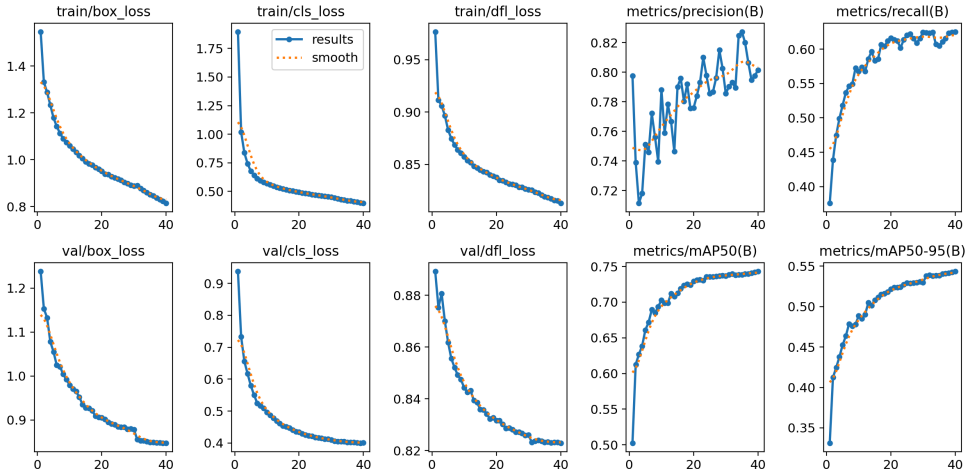

Figure 9: Metrics from the final model

Explanation of the metrics:

Train/Validation Box Loss (Top Left and Bottom Left Graphs) show the loss during training and validation. Loss functions measure the difference between the predicted outputs by the model and the actual labels, with lower values indicating better performance. The 'train/box_loss' and 'val/box_loss' show a decreasing trend, which suggests that the model is learning effectively and generalizing well to unseen data. Train/Validation Loss (Second from Left, Top and Bottom) shows similar statistics to box loss, but it could include other components of the loss function used in training the model, like classification loss or other regularization losses.

Precision (metrics/precision@0.5) indicates the ratio of correctly predicted positive observations to the total predicted positives. High precision relates to the low false positive rate.

Recall (metrics/recall@0.5) indicates the ratio of correctly predicted positive observations to all observations in actual class. High recall relates to the low false negative rate.

mAP@0.5 measures the model's average precision at an Intersection over Union (IoU) threshold of 0.5. This metric is particularly useful for assessing how well the model detects objects with at least 50% overlap with the ground truths.

mAP@0.5:0.95 is a more stringent and comprehensive metric, it calculates the average precision at multiple IoU thresholds from 0.5 to 0.95 (in 0.05 increments), offering a detailed view of the model's performance across different levels of detection difficulty.

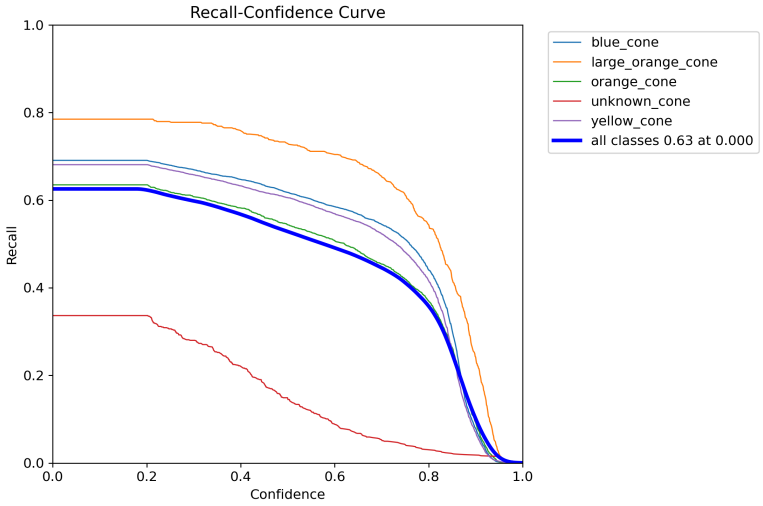

Figure 10: Recall-Confidence curve for the final model

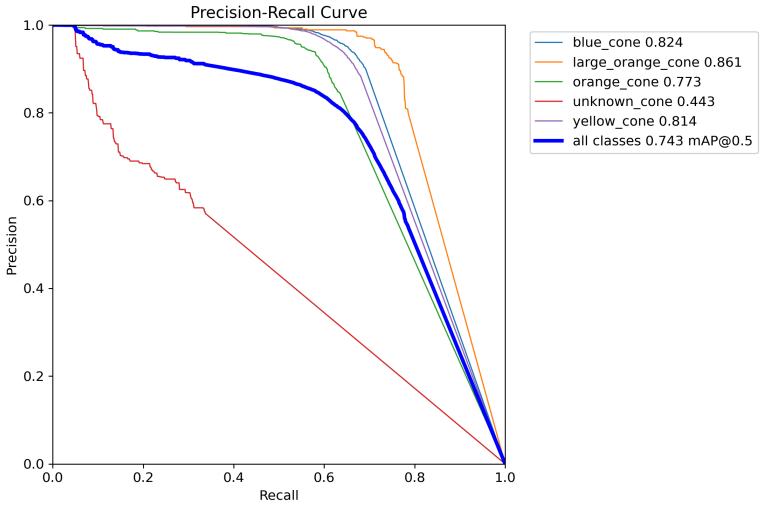

Figure 11: Precision-Recall curve for the final model

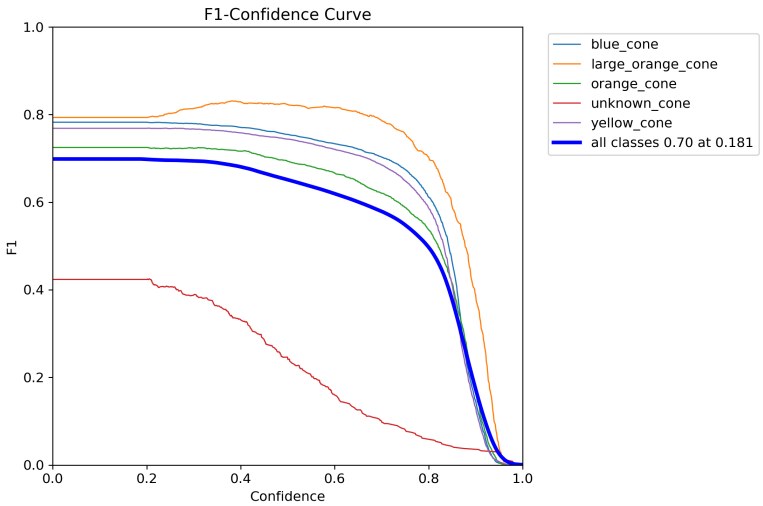

Figure 12: F1-Confidence curve for the final model

Finally we also have the Normalized Confusion Matrix that shows the accuracy for all of the classes as well as how often it is confused with a background.

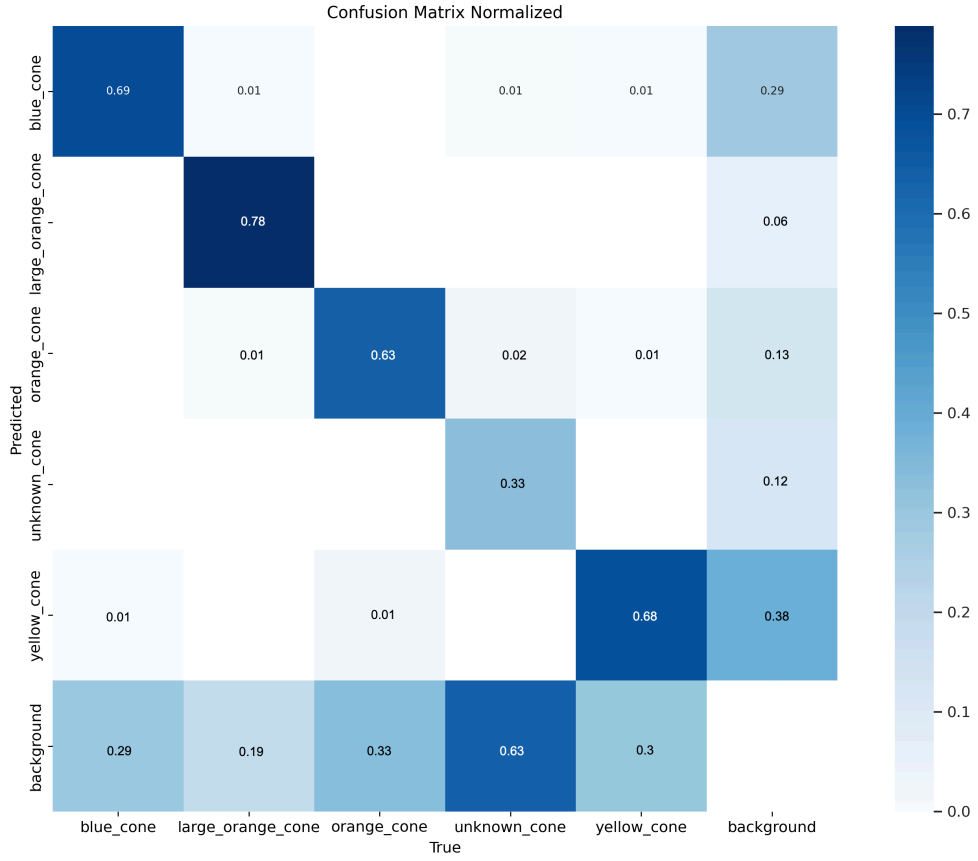

Figure 13: Normalized Confusion Matrix for the model

5 Discussion

5.1 Result Analysis

The final model achieved an mAP50 of 0.75 and an mAP50-95 of 0.55 after 40 epochs. We can also see that unknown cones is the worst performing class, which is to be expected. As the confusion matrix shows that most unknown cones are recognized as background, this is acceptable for our usage since the unknown class are not prevelent in the training nor test data. The large orange cone has by far the highest average precision, the conclusion for this is that the large orange cone visually deviates the most from other cones, especially when the color is removed. Since the large orange cone has two instead of one line on them, as picture shows below.

Figure 14: Example of large orange cone

5.1.1 Comparison with the KTHFS model

The current model used by KTH Formula Student Driverless has the following configuration:

model=yolov8n.pt epochs=300 image_size=1024 dropout=0.2

And achieves a mAP50 value of approximately 0.86 after 300 epochs even if the model already converges towards that value after 100 epochs of training.

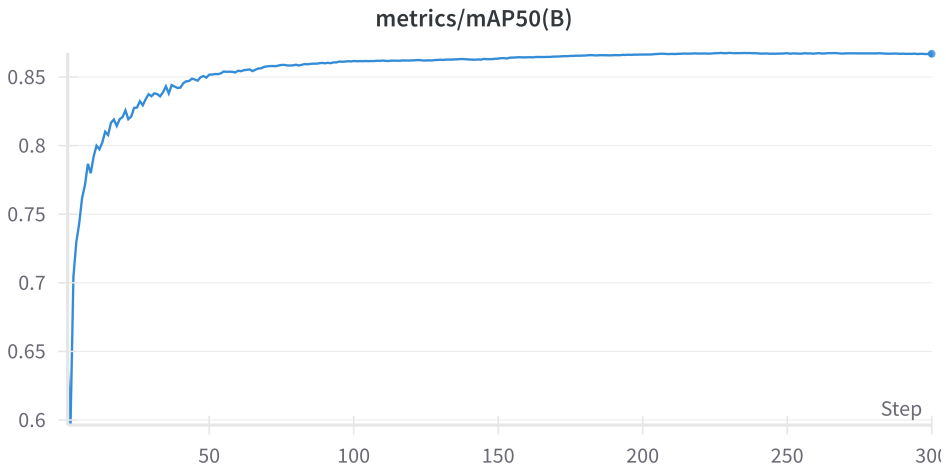

Figure 15: mAP50 for the KTHFS DV Model

The mAP50-95 value after those 300 epochs is approximately 0.61. Like the mAP50 value, this value also seems to converge towards 0.61 after 100 epochs.

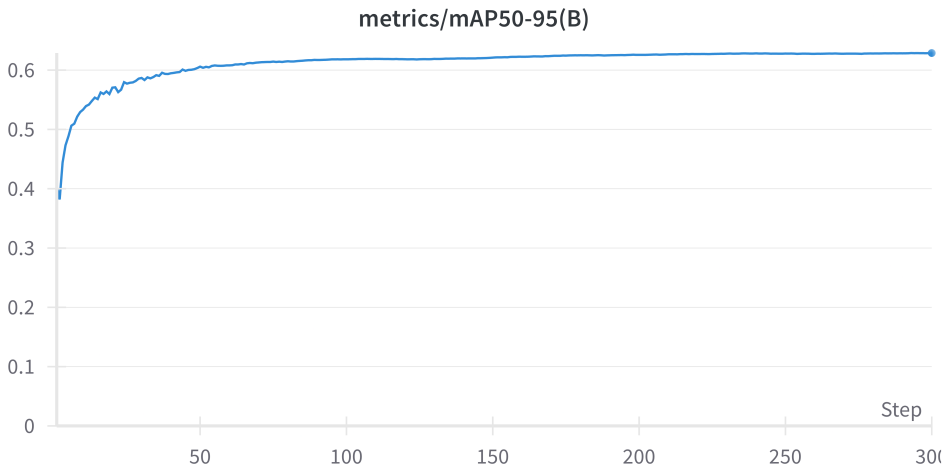

Figure 16: mAP50-95 for the KTHFS DV Model

As observed in the KTHFS DV model mAP graphs, both the maP50 and mAP50-95 started to converge between 40-50 epochs. This is similiar to our observations and a qualified comparison can be made between the models.

5.1.2 Performance evaluation

One of the main objectives for this study is to gain performance for the driverless algorithm on the KTH Formula Student race car. As mentioned above, the race car uses a combination of LiDAR and computer vision to achieve maximum efficiency on track. The current colored camera used for cone detection can achieves resolution of 2448 x 2048 and frame rate of 60fps. With a

monochrome camera around the same budget, the performance could be significantly upgraded [11]. Let's assume the frame rate doubles to 120fps. This makes a significant difference in high speed scenarios. The formula student cars top speed is 120km/h, and averages around 80km/h on the high-speed sections on track. A formula can be used to calculate the distance covered by the race car per frame(D). Let the velocity be V and fps be F, then $D = \frac{V}{F}$. 80km/h is 22.2 meters per second, this means with a 60fps camera, D is 37 centimeters per frame. This number reduces by half with a 120fps camera to 18.5 centimeter per frame. This means in a potential decision making scenario, the decision could be made 18.5 centimeters in prior when using a monochrome camera. This is a significant distance in racing terms since the race car is agile and the track narrow and twisty which favors fast decision-making.

Higher frame rate will as well lead to clearer images for the model to evaluate. This is more significant for cones further away since details of the cone are more difficult to capture at a distance. The result can be further analyzed by creating a new dataset with a high frame rate camera, as discussed in section below.

Another aspect to consider is the probability a cone is detected over time increases as the frame rate increase, see graph below. We can as well make an reasonable deduction that the lighting conditions as well as angle could change between frames leading to detection of previously ignored cones. However, because of the complex nature of the model, it's impossible to produce an accurate value of the model's confidence with twice the available frames. Nonetheless the performance should improve and converge towards the original model.

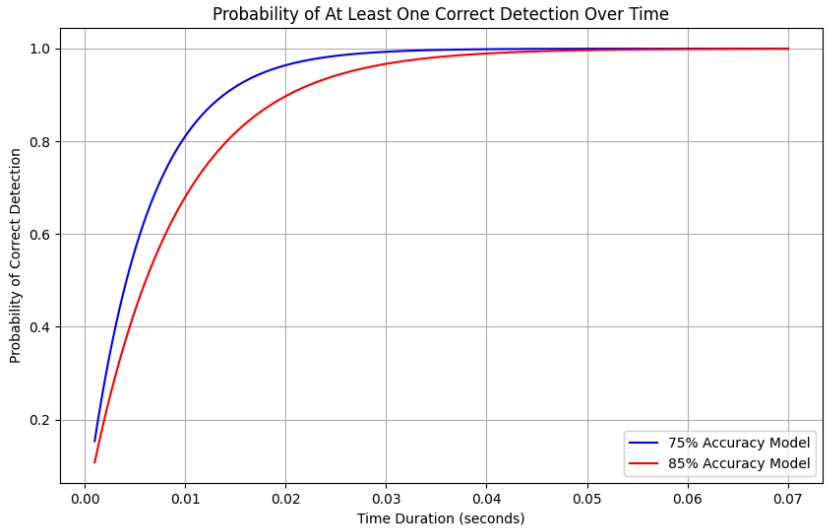

Figure 17: Probability of one detection if frame rate is twice as high for the 75% model than the 85% model

5.2 Alternative Solutions and Improvements

During the pilot phase of this study, a new pre-trained model was published by Ultralytics, YOLOv9. YOLOv9 introduces Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Network (GELAN), which are designed to minimize information loss and optimize computational paths, respectively. These features help in maintaining crucial data through deeper network layers and improve parameter utilization, leading to more accurate and efficient object detection.

Another solution fit to our purpose is to build our own deep convolutional neutral network with customized layers. This is viable in the future since a tailored architecture allows for optimization focused on the specific characteristics of grayscaled data. A custom model can be designed to be leaner and more computationally efficient, as it doesn't need to process the additional color channel data, thus potentially running faster and using less memory. Such a model can also learn more relevant features and patterns specific to grayscale data, leading to better performance in specific applications like low-light imaging or motion detection. [16]

At the pilot phase of this study, an alternate approach was considered about collecting image data. Rule compliant cones were available at request from KTH Formula Student and there was a possibility of collecting our own photos. Another possible approach was to create tracks and cones in game engine software such as Unreal Engine 5. Both abovementioned approaches was viable since the gathered data would be more fit to a specific monochrome

camera, thus making the model more use case specific. However, this study aims to compare against and optimize the KTH Formula Student model which is trained on a specific dataset, our approach allows us to make more accurate comparisons between the two model's performances.

5.3 FSD Implications

FSD, or full self driving, is one of the most relevant subjects in modern vehicle development. In most hardware implementations of autonomous vehicles, both LiDar and cameras are used to optimize accuracy and decision making proficiency. Using grayscale cameras in the development of FSD vehicles offers several advanages and considerations, particularly in processing efficiency, computational requirements and night vision optimization. Our study further shows that optimizing the object detection pipeline by using grayscaled images only lead to minor compromizes in accuracy with gains in other relevant areas. One of the main implementations is that monochrome cameras often perform better in low light conditions compared to color cameras. They can capture more detail in dimly lit environments, which is advantageous for nighttime driving or conditions with poor visibility and allow the vehicle to gather more real time information with lower latency. [16]

6 Conclusion

In conclusion, this study demonstrates the viability of using grayscale computer vision models in autonomous vehicles, specifically within the context of Formula Student competitions. By comparing various grayscaling methods and evaluating their impact on the YOLOv8 object detection model, the research reveals that certain grayscale approaches can achieve classification accuracies comparable to those of color-based models. This finding suggests that implementing grayscale imaging can enhance the frame rate and decision-making speed of autonomous systems without significantly compromising accuracy.

The advantages of grayscale cameras, such as higher frame rates and better performance in low-light conditions, make them particularly suitable for the dynamic and high-speed environments of Formula Student races. The study's results indicate that a strategic use of grayscale methods can optimize the performance of autonomous vehicle vision systems, leading to faster and more reliable object detection.

These insights can contribute to the broader field of autonomous driving, offering a potential method for developing more efficient and adaptable driverless technologies. The research underscores the importance of continued exploration and optimization of vision systems to enhance the safety of autonomous vehicles across various driving conditions.

Overall, this work provides a solid foundation for future advancements in the integration of grayscale vision models in autonomous vehicle technology, guiding the way for innovations that could extend beyond competitive racing to commercial applications in everyday driving scenarios.

Bibliography

[1] NOAA, "What is Lidar?." [Online]. Available: https://oceanservice.noaa.gov/facts/lidar.html#:~:text=Lidar%2C%20which%20stands%20for%20Light,variable%20distances)%20to%20the%20Earth.

[2] N. Malviya, "Evaluation of Object Detection Models — Flops, FPS, Latency, Params, Size, Memory, Storage, mAP, AP." [Online]. Available: https://medium.com/@nikitamalviya/evaluation-of-object-detection-models-flops-fps-latency-params-size-memory-storage-map-8dc9c7763cfe

[3] G. W. Kanan Christopher AND Cottrell, "Color-to-Grayscale: Does the Method Matter in Image Recognition?," *PLOS ONE*, vol. 7, no. 1, pp. 1–7, 2012, doi: 10.1371/journal.pone.0029740.

[4] J. Fasola and M. Veloso, "Real-time object detection using segmented and grayscale images," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 4088–4093. doi: 10.1109/ROBOT.2006.1642330.

[5] J. Kabzan *et al.*, "AMZ Driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020, doi: https://doi.org/10.1002/rob.21977.

[6] N. Vödisch, D. Dodel, and M. Schötz, "FSOCO: The Formula Student Objects in Context Dataset," *SAE International Journal of Connected and Automated Vehicles*, vol. 5, no. 12–5–1–3, 2022.

[7] J. Torres, "YOLOv8 Architecture: A Deep Dive into its Architecture." [Online]. Available: https://yolov8.org/yolov8-architecture/

[8] Ultralytics, "Ultralytics github." [Online]. Available: https://github.com/ultralytics/ultralytics

[9] J. Pedro, "Detailed Explanation of YOLOv8 Architecture — Part 1." [Online]. Available: https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e

[10] a.-v. glenn-jocher (4) RizwanMunawar (1), "Performance Metrics Deep Dive." 2023.

[11] TechNexion, "A short guide to why monochrome cameras have the edge over color cameras." [Online]. Available: https://www.technexion.com/resources/monochrome-camera-vs-color-camera-all-you-need-to-know/

[12] P. Wilson, "Chapter 7 - High Speed Video Application." [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080971292000076

[13] Wikipedia, “Bayer filter – Wikipedia, The Free Encyclopedia.” [Online]. Available: https://en.wikipedia.org/wiki/Bayer_filter

[14] P. Kumar, “A short guide to why monochrome cameras have the edge over color cameras.” [Online]. Available: https://www.e-consystems.com/blog/camera/technology/a-short-guide-to-why-monochrome-cameras-have-the-edge-over-color-cameras

[15] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context.” 2015.

[16] “Night Sight - Competing technologies for the vision systems in autonomous vehicles.” [Online]. Available: https://possibility.teledyneimaging.com/night-sight-competing-technologies-for-the-vision-systems-in-autonomous-vehicles/

TRITA – XXX-XXX 20XX:XX
Stockholm, Sverige 2024

www.kth.se