Semantička segmentacija strukturiranog okruženja mobilnog robota

Yassin, Baraa

Undergraduate thesis / Završni rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Electrical Engineering and Computing / Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:168:292856

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-23**

Repository / Repozitorij:

FER Repository - University of Zagreb Faculty of Electrical Engineering and Computing repozitory

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 1543

SEMANTIC SEGMENTATION OF A STRUCTURED ENVIRONMENT FOR MOBILE ROBOTS

Baraa Yassin

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

BACHELOR THESIS No. 1543

SEMANTIC SEGMENTATION OF A STRUCTURED ENVIRONMENT FOR MOBILE ROBOTS

Baraa Yassin

Zagreb, June 2024

UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

Zagreb, 04 March 2024

BACHELOR THESIS ASSIGNMENT No. 1543

Student:	**Baraa Yassin (0036542255)**
Study:	Electrical Engineering and Information Technology and Computing
Module:	Computing
Mentor:	assoc. prof. Matko Orsag

Title: **Semantic segmentation of a structured environment for mobile robots**

Description:

As part of the task, it is necessary to research and review the existing research area of object detection and segmentation in the environment. It is required to select a subset of the most common features in the structured environment of a mobile robot, for which a balanced dataset based on synthetic and automatically labeled data will be built. It is necessary to train a combination of neural networks that will be able to robustly segment features from the environment for the purpose of locating them in space. The algorithm needs to be tested in a simulation environment.

Submission date: 14 June 2024

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

ZAVRŠNI ZADATAK br. 1543

Pristupnik:	**Baraa Yassin (0036542255)**
Studij:	Elektrotehnika i informacijska tehnologija i Računarstvo
Modul:	Računarstvo
Mentor:	izv. prof. dr. sc. Matko Orsag

Zadatak: **Semantička segmentacija strukturiranog okruženja mobilnog robota**

Opis zadatka:

U sklopu zadatka potrebno je istražiti i pregledati postojeće istraživačko područje detekcije i segmentacije predmeta u okolini. Potrebno je odabrati podskup najčešćih značajki u strukturiranoj okolini mobilnog robota, za koji će se izgraditi balansirani skup podataka temeljen na sintetskim i automatski označenim podacima. Potrebno je istrenirati kombinaciju neuronskih mreža koji će moći robusno segmentirati značajke iz okoline u svrhu lociranja istih u prostoru. Algoritam je potrebno ispitati u simulacijskom okruženju.

Rok za predaju rada: 14. lipnja 2024.

Filip Zorić, univ. mag. ing. el. techn. inf.

Contents

1 Introduction . . . 3

1.1 Background and Motivation . . . 3

1.2 Problem Statement . . . 4

1.3 Objectives . . . 4

2 Theoretical Background . . . 6

2.1 Relevant Concepts from Machine Learning . . . 6

2.2 What is Semantic Segmentation and Computer Vision . . . 7

2.3 Components Needed for Segmentation . . . 9

3 Methodology . . . 14

3.1 Overview of Available Datasets . . . 14

3.2 Description of Neural Network Architectures . . . 16

3.3 Training Process . . . 16

3.4 Evaluation Metrics for Semantic Segmentation . . . 18

4 Tools and Technologies . . . 20

4.1 Introduction to Python and its Libraries for Machine Learning . . . 20

4.2 Linux, Docker, Hugging Face . . . 21

4.3 Overview of ROS 2 (Robot Operating System) and its Components . . . 22

4.4 Justification for the Chosen Tools and Technologies . . . 22

5 Experimental Results . . . 23

5.1 Presentation of Segmentation Results . . . 23

5.2 Analysis of Performance Metrics . . . 26

5.3 Challenges Encountered during Experiments . . . 26

6 Conclusion **29**
6.1 Summary of Achievements 29
6.2 Contributions to the Field 29
6.3 Limitations and Future Work 30

7 References **32**

Abstract **37**

Sažetak **38**

8 Appendix **39**

1 Introduction

1.1 Background and Motivation

Semantic segmentation, a key part of computer vision, involves determining a class for every pixel in an image or video. It is also one of the fundamental computer vision tasks in automated driving, medical imaging, industrial inspection, and other critical applications. In automated driving, semantic segmentation is crucial for enabling vehicles to map out every part of their environment, allowing them to understand their environment and dynamically adjust speed for safe and efficient navigation. For example, accurately distinguishing between roads, sidewalks, pedestrians, and vehicles is essential for the vehicle to make real-time decisions that ensure safety at every moment.

Figure 1.1: Example of road image being segmented into classes, where every color is a different class

While the uses of semantic segmentation are easy to understand, the task of semantic segmentation is a challenging problem in the world of artificial intelligence. The complexity arises from the need to accurately and efficiently process large-sized images, often in real-time. Semantic segmentation models must be robust to variations in lighting, occlusions, and object appearances, and must generalize well across different environments. Furthermore, the models need to be computationally efficient to meet the real-

time processing requirements of applications such as autonomous driving and robotic navigation.

1.2 Problem Statement

Semantic segmentation that works in real-time has been one of the more challenging aspects of computer vision because it is crucial for it to be both fast and accurate. Real-time performance is essential in applications like autonomous driving, where decisions must be made in milliseconds to ensure safety. This requires models that are built for efficiency, accuracy and speed. This dynamic is often difficult to achieve due to accurate models often being slow and fast models are often not accurate enough. Semantic segmentation approaches of today often face significant challenges in quality and efficiency, particularly when applied to synthetic and automatically labeled datasets. These challenges hinder the robots ability to reliably locate and interact with objects in their environment.

1.3 Objectives

The primary objectives of this thesis are:

1. Extensive research in the field of detection and segmentation
   - Review of the current state-of-the-art techniques in semantic segmentation and object detection
   - Identify the strengths and weaknesses of existing methods
   - Understand the methods and algorithms used in the segmentation process

2. Developing an algorithm for examining the segmentation model
   - Develop a dockerfile for an isolated environment
   - Develop a ROS 2 node that output a mask for the segmented object
   - Expand the ROS 2 node to work on a video stream

3. Comparison of multiple semantic segmentation models based on robustness, prediction time and accuracy

- Test multiple models on using the ROS 2 node and visualize through RVIZ
- Calculate the plotting time and prediction time
- Examine the data

2 Theoretical Background

2.1 Relevant Concepts from Machine Learning

Machine learning introduces several concepts that are crucial to semantic segmentation: Dataset, Model, Optimization process. Understanding semantic segmentation requires an understanding of these elements.

A dataset is a collection of data points used to train, validate and test machine learning models. The data points are split into these three categories to prevent overfitting, which is when the machine learning model gives accurate predictions for training data but not for new data. The percentage split between categories is defined by the creator of the model but the usual split is: 40 percent for training, 30 percent to validate, 30 percent for testing. In the context of semantic segmentation the dataset usually consist of images and their corresponding labels, where each pixel is annotated with a specific class.

Training split of dataset contains images which are used to train the model. It contains a large number of labeled examples that the model uses to learn patterns and features in the data. Validation Dataset split of dataset is used to tune the hyperparameters of the model and to monitor its performance during training. Hyperparameters are variables, for example the depth of a decision tree, that are used to get the best possible version of a specific model. By tuning these parameters we can prevent overfittting and underfitting. Test Dataset split of dataset is used to evaluate the final performance of the model. It provides an unbiased estimate of how well the model will perform on new, unseen data.

A model in machine learning is a mathematical representation that maps input data to output predictions. In semantic segmentation, models are typically deep neural networks designed to assign a class label to each pixel in an image. Models are trained using

datasets. The data is passed through the model to generate predictions using non-linear functions.

The optimization process in machine learning involves adjusting the models parameters to minimize the loss function, thereby improving the models predictions. The loss function is a measure of how well the model performs on a task, such as minimizing the error on a set of training data. There are a lot of optimization algorithms used in machine learning one of the most notable is Gradient Descent. Gradient Descent is a first-order iterative optimization algorithm broadly used in machine learning and optimization problems. Its primary purpose is to minimize a differentiable cost or loss function by adjusting the parameters of a model [6].

2.2 What is Semantic Segmentation and Computer Vision

Computer vision is a subfield of Artificial intelligence that uses machine learning and neural networks to process images and to extract meaningful information and determine what they contain. The field has risen in popularity due to recent advancements, which in turn fuel further progress. Computer vision operates on the principle of learning followed by evaluation. For example, for an algorithm to recognize the difference between a sidewalk and a street, it needs a vast collection of labeled data. The algorithm repeatedly analyzes this data to discern distinctions and recognize patterns, which constitutes the foundation of computer vision. Two essential technologies are used to achieve this: deep learning, a type of machine learning, and convolutional neural networks (CNNs) [10], which will be explained in the next sub-chapter 2.3. Real-world applications demonstrate how important computer vision is to endeavors in entertainment, transportation, healthcare, and everyday life. A few examples of already established computer vision tasks are image classification, object detection, object tracking, content-based image retrieval, and more. Image Classification is the task of classifying an entire image into its predicted class. This involves training models to recognize and categorize different elements within an image accurately. Object Detection involves identifying and classifying individual objects within an image or video. This process segments the visual input into distinct objects that can be independently classified. Object Tracking follows the object

once it is detected and tracks its location based on its movements between frames in a video stream.

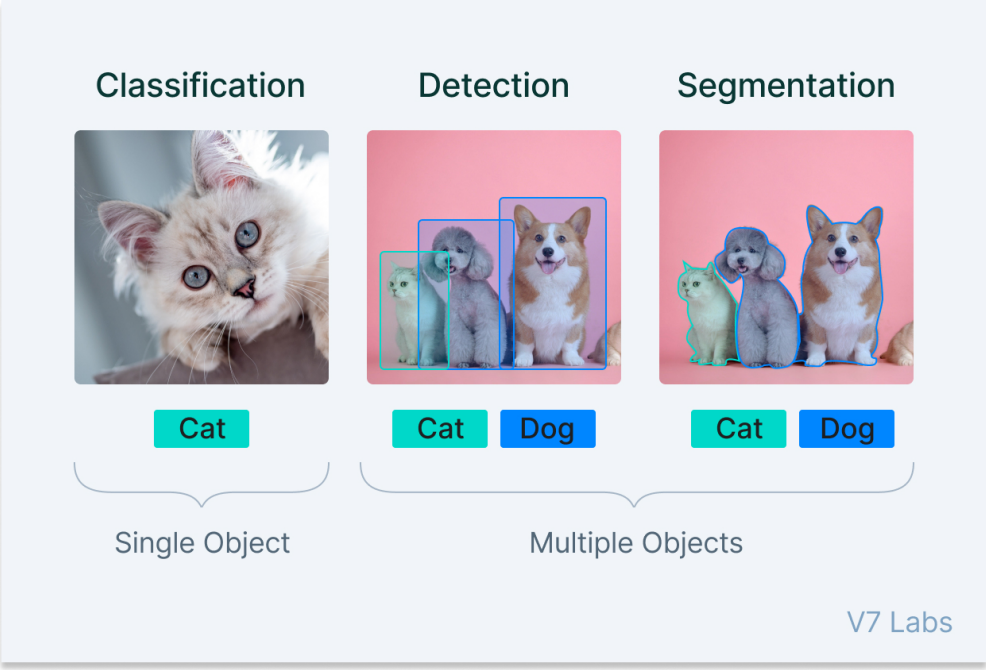

Figure 2.1: The difference between image classification, object detection and instance segmentation

Combining these methods and fields, we get semantic segmentation, which refers to the computer vision task where we classify each pixel on the image into certain category. It is used to recognize a collection of pixels that form distinct categories or classes using classification. For example, an autonomous vehicle needs to identify vehicles, pedestrians, traffic signs, pavement, and other road features.

Semantic segmentation models often use convolutional neural networks (CNNs) due to their ability to effectively capture spatial hierarchies in images. Architectures like Fully Convolutional Networks (FCNs) [7], U-Net [10], and SegNet [1] have been developed specifically for semantic segmentation tasks.

These models are trained on large annotated datasets and learn to identify and segment various classes within images. In the context of autonomous cars and mobile robots, semantic segmentation is indispensable. The vehicle receives data as a stream of images, requiring the model to perform inference in real-time without prior knowledge of the route. The model must be highly sensitive in detecting obstacles. For example, in instances where road segmentation is not perfectly clear due to other objects, cars,

weather etc. the classification of obstacles must stay consistent and ensure safety.

Humans are naturally adept at segmenting images, even without knowing what the objects are. This ability highlights the importance of semantic image segmentation for autonomous navigation. While other detection models can classify and locate obstacles, they are limited to recognizing only the obstacles they have been trained on. For instance, an obstacle detector designed to identify pedestrians in autonomous cars will only alert the vehicle in the presence of pedestrians, as it has been trained specifically for that category. However, autonomous driving scenarios present an unpredictable variety of obstacles that can take any shape or form, making it impractical to create a dataset encompassing all possible obstacles. This is where semantic image segmentation becomes crucial. An ideal semantic image segmentation model can define the boundaries of any object, even those it has not previously encountered. This ability ensures that autonomous vehicles can navigate safely and efficiently in diverse and unpredictable environments.

That is why an effective semantic segmentation model for autonomous navigation must meet several requirements:

- **Real-time Inference:** The model should process images quickly enough to make real-time decisions.
- **High sensitivity and accuracy:** It must accurately detect and classify obstacles, even in challenging conditions.
- **Generalization:** The model should generalize well to new, unseen obstacles, maintaining high performance across various scenarios.
- **Continuous Learning:** The ability to learn from new data and improve over time is crucial for adapting to ever-changing environments.

2.3 Components Needed for Segmentation

Semantic segmentation relies on several core components in machine learning and computer vision (like image classification, object detection etc.), including machine learning algorithms, neural networks, and specifically convolutional neural networks (CNNs).

These components work together to enable machines to understand and interpret visual data at a pixel level and classify them based on the output class of the neural network of the model.

Machine learning is a subset of artificial intelligence that focuses on observing a mass collection of data and drawing patterns in them so it can determine these patterns in unseen data. Machine learning uses complex algorithms and statistics to enable computers to improve their performance on a specific task through experience. In the context of semantic segmentation, machine learning techniques are used to train models on large datasets of labeled images. According to Arthur Samuel Machine learning is defined as the field of study that gives computers the ability to learn without being explicitly programmed. Arthur Samuel was famous for his checkers playing program. Machine learning (ML) is used to teach machines how to handle the data more efficiently. Sometimes after viewing the data, we cannot interpret the extract information from the data. In that case, we apply machine learning. With the abundance of datasets available, the demand for machine learning is in rise. Many industries apply machine learning to extract relevant data. The purpose of machine learning is to learn from the data [8].

Neural network is a type of model in artificial intelligence that teaches computers to process and compute data inspired by the human brain. The neural network consists of interconnected layers that are ordered in a linear fashion of course to mimic the human brain. We differentiate between a couple of types of layers: Input layer which receives the initial data, hidden layers in which the data is processed to find patterns and output layer which gives us the final result of the neural network. Each layer has nodes called (artificial) neurons for computing a certain kind of pattern or reference. It creates an adaptive system that computers use to learn from their mistakes and improve with time. Artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.

Neural networks are a set of dependent non-linear functions. Each individual function consists of a neuron (or a perception). These (artificial) neurons receive input from the previous layer (or raw data if the neuron is in the input layer) and extract features and patterns using the non-linear function and output the result to the next layer.

The perceptron function is defined as:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where:

- f is the activation function.
- w_i represents the weight associated with the i-th input.
- x_i represents the i-th input.
- b is the bias term.
- n is the number of inputs.

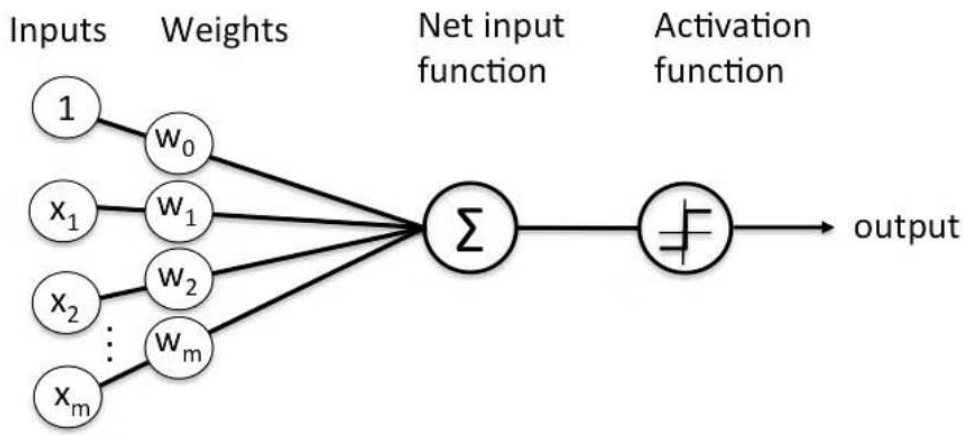

Figure 2.2: Perceptron architecture example

Fully Connected Layers is a structure of neural networks where every neuron in the preceding layer is connected to every neuron in the layer after it. This structure is so each neuron can apply a linear transformation to the input vector through a weights matrix. As a result every input of the input vector influences every output of the output vector. This is used to combine features extracted from previous layers and make final predictions. In the context of semantic segmentation fully connected layers are used for making accurate pixel-level predictions.

A convolutional neural network (CNN) assists a machine learning or deep learning model in "seeing" by analyzing images at the pixel level, assigning tags or labels to these

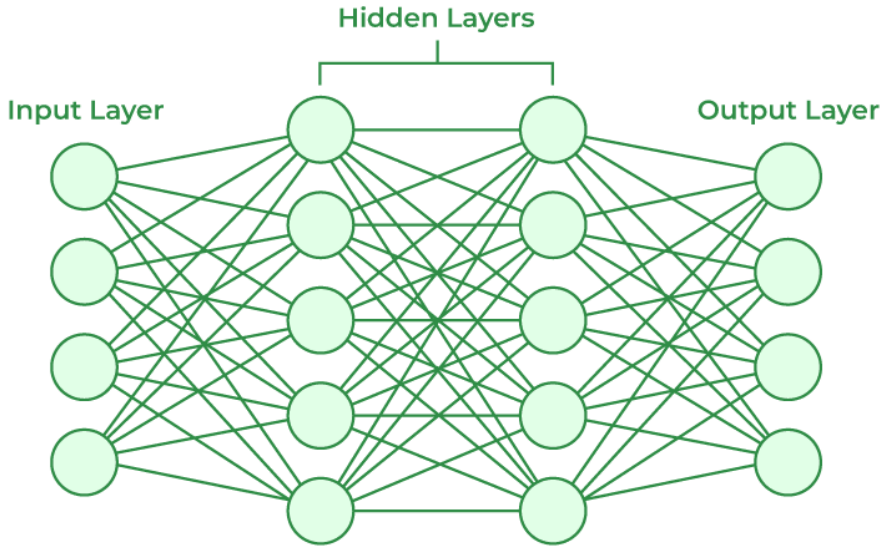

Figure 2.3: Example of a simple neural network architecture

pixels. It employs these labels to perform convolutions (a mathematical operation that combines two functions to produce a third function) and makes predictions about the content of the image. The neural network continuously runs these convolutions and refines its predictions through multiple iterations, gradually improving its accuracy. Eventually, the model begins to recognize and interpret images in a manner similar to human vision.

The convolution of two functions f and g is defined as:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)\,d\tau$$

where: 2.4

- τ is a dummy variable of integration, representing a shift parameter.
- t is the point at which the convolution is evaluated.

Attention Mechanism is a technique in machine learning and artificial intelligence to improve the performance of models by focusing on relevant information. They are designed to mimic the human visual attention process, where focus is selectively directed to important parts of an image. This is done by assigning weights to different neurons/features with the most important neurons/features receiving the highest weights.[12]

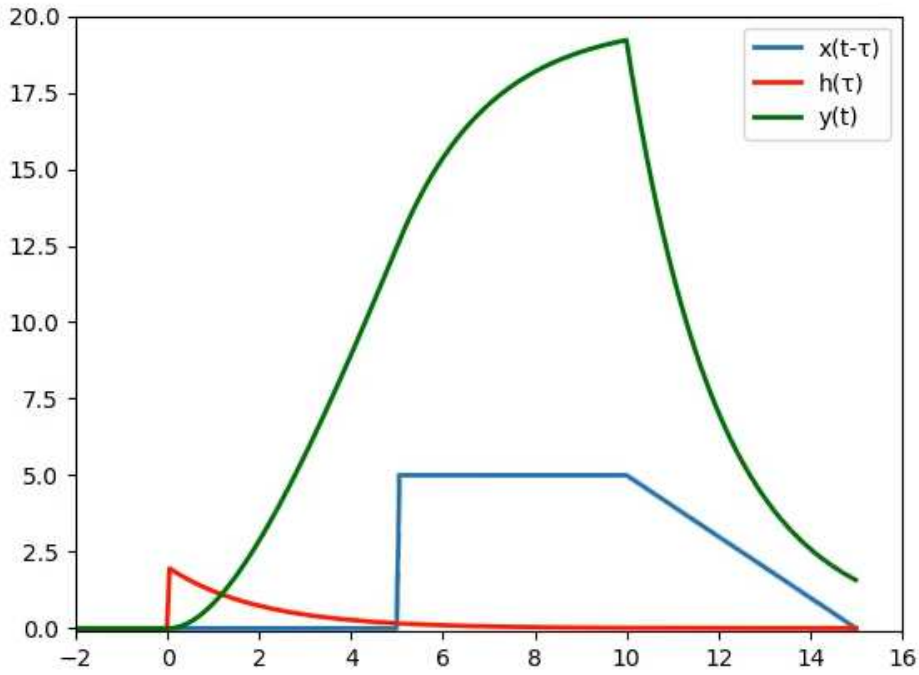

Figure 2.4: Visualization of a convolution function

The plot shows three curves:

- **Blue curve ($x(t-\tau)$)**: This represents the function $x(t-\tau)$ over the time interval shown.
- **Red curve ($h(\tau)$)**: This represents the function $h(\tau)$ over the time interval shown.
- **Green curve ($y(t)$)**: This represents the resulting function $y(t)$, which is the convolution of $x(t-\tau)$ and $h(\tau)$.

3 Methodology

3.1 Overview of Available Datasets

In the field of semantic segmentation, a variety of datasets are available that cater to different types of environments and applications. Some of the most widely used datasets include:

- **Common Objects in COntext-Coco Dataset** is a large scale object detection and image segmentation dataset. It contains 91 classes with more than 200,000 labeled images with segmentation masks.

- **PASCAL Visual Object Classes (PASCAL VOC)** The PASCAL VOC dataset is one of the earliest and most influential datasets for object detection and segmentation. It includes images annotated with bounding boxes, object class labels, and segmentation masks. The dataset covers 20 object classes.

- **The Cityscapes Dataset** The Cityscapes dataset is specifically designed for urban scene understanding, making it particularly useful for autonomous driving applications. The dataset includes 30 classes related to road objects, such as cars, pedestrians, traffic signs

- **ADE20k** dataset is a comprehensive dataset for scene parsing, with more than 20,000 images and annotations covering over 150 object categories making it one of the most detailed and diverse datasets available for semantic segmentation. The images in ADE20K span a wide range of environments and scenarios, including indoor scenes (such as bedrooms, kitchens, and offices) and outdoor scenes (such as streets, parks, and beaches). This diversity ensures that models trained on ADE20K can generalize well across different types of scenes and objects, which is crucial for

real-world applications.

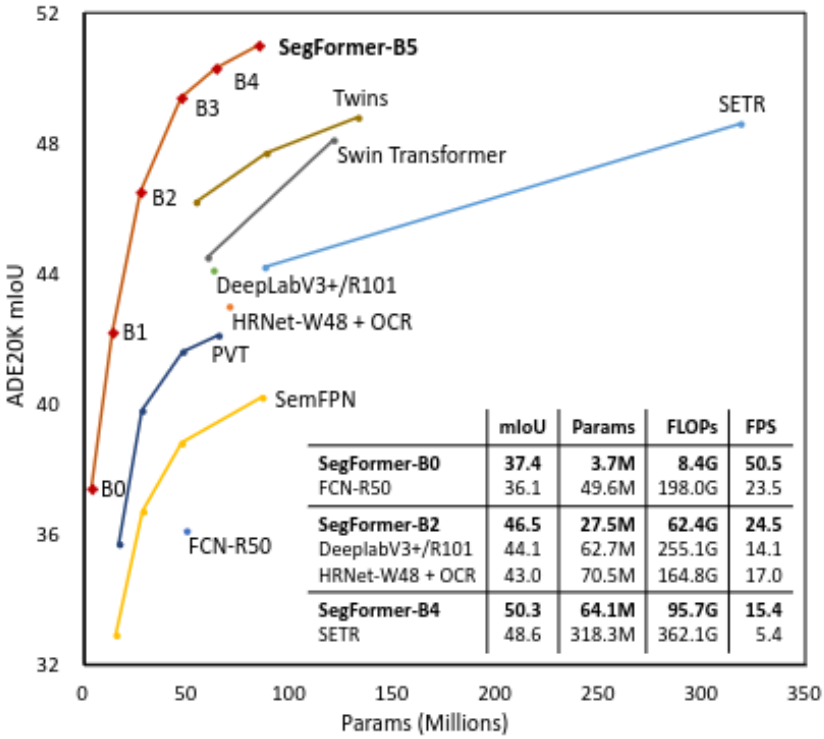

Figure 3.1: Performance vs. model efficiency on ADE20K. [13]

Most of the experiments for this work were done on three specific models from NVIDIA's SegFormer[13] series, which have been fine-tuned on the ADE20K dataset. The ADE20K dataset's extensive variety of scene categories and objects provides a solid foundation for training robust semantic segmentation models. The used models are:

Table 3.1: Overview of NVIDIA's SegFormer Models

Model	**Parameters**	**Description**
nvidia/segformer-b0	3.8 Million	The smallest and most lightweight in the SegFormer series. Balances efficiency and performance, ideal for resource-limited applications.
nvidia/segformer-b1	13.7 Million	Offers increased capacity and improved performance over the b0 model. Suitable for more complex segmentation tasks, still considered small.
nvidia/segformer-b3	47.2 Million	A more advanced model with higher accuracy and performance, suitable for applications demanding high precision, with increased computational requirements.

3.2 Description of Neural Network Architectures

NVIDIA's SegFormer models use a hierarchical Transformer architecture known as the Mix Vision Transformer (MiT) [2], which is based on the Vision Transformer (ViT) [5] architecture, as its encoder and a lightweight decoder for segmentation. The main difference between hierarchical transformers and regular transformers is in their computing time. The attention mechanism used in regular transformers takes $O(n^2)$ time complexity to run, where n is the sequence length. This complexity makes regular transformers inefficient for processing long sequences of data. Hierarchical transformers resolved this issue by allowing the model to operate on different levels of the input. For instance, in natural language processing, hierarchical transformers can process words, sentences, paragraphs, and so on. Unlike traditional Vision Transformers that use patches of size 16×16 pixels, the Mix Vision Transformer (MiT) used in SegFormer employs smaller patches, which favors the prediction task. By using smaller patches, the model can capture finer details in the image, which is crucial for semantic segmentation where precise boundaries between different segments are needed. The hierarchical Transformer encoder in MiT operates on these smaller patches to obtain multi-level features at scales of 1/4, 1/8, 1/16, 1/32 of the original image resolution. This hierarchical nature of MiT allows it to capture features at multiple scales, enhancing its ability to understand both local and global contexts within an image. For example, capturing the local context and the global context simultaneously.

3.3 Training Process

The training process includes preparing the dataset, configuring the neural network, and optimizing the model parameters. Preparing the dataset includes labeling each image with the desired classes so that the model can differentiate between classes. To enhance the model and also to prevent it from underfitting, augmentations are applied to the images in the dataset. Augmentations include techniques such as random cropping, flipping, rotation, and color jittering. These techniques are applied to increase dataset diversity and improve model generalization. It is important to note that the dataset is going to be split into three categories: Training, validating and testing. This means that for training the neural network model only the training section of the dataset is used.

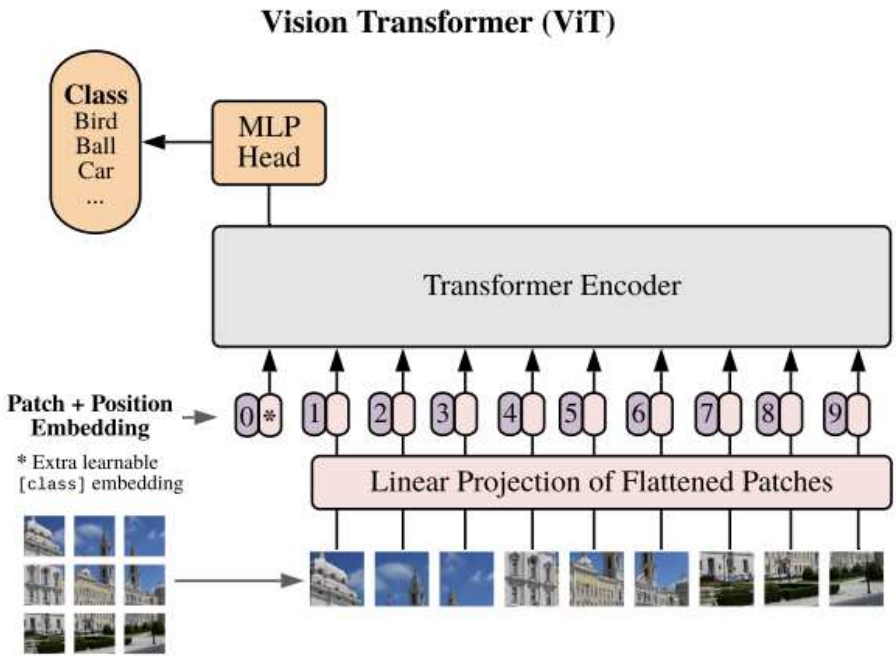

Figure 3.2: Vision transformer architecture [4]

Along side dataset preparation the training configuration also must be set. This includes configuring the GPUs to handle the computational load of training the model because of its capabilities of parallel processing. GPUs can greatly accelerate deep learning model training, as they are specialized for performing the tensor operations at the heart of neural networks. Preforming these tensor operation is somewhat difficult from the software perspective so frameworks like PyTorch and TensorFlow are used because of the number of methods and compatibility with machine learning operations. To achieve the best performance, two primary training strategies are compared: training from scratch and transfer learning. Training from scratch involves initializing the model weights randomly and allowing the model to learn from the dataset from the beginning. This approach can be highly effective but requires large amounts of data and significant computational resources. The model gradually improves as it is exposed to more data, learning the features and patterns from the ground up. Transfer learning uses pretrained weights from large datasets and fine-tunes them on the target dataset. This strategy capitalizes on the prior knowledge embedded in the pre-trained model, allowing it to converge faster and achieve better performance, particularly on smaller datasets.

3.4 Evaluation Metrics for Semantic Segmentation

Evaluation metrics are used to determine how well the model functions. The goal of evaluation metrics is to examine the accuracy of the models prediction by comparing the predicted and annotated segmentation. The most common metrics are: Precision and Recall (Sensitivity), Accuracy/Rand index, Dice coefficient and Jaccard index (IoU). All of these metrics are based on the results of the confusion matrix, which is a two-by-two matrix that holds values of true-positive/TP (the model gave the correct result based on the annotated segmentation segmenting the object), true-negative/TN (the models output is correct by not segmenting background pixels),false-positive/FP (the model did not respond correctly, not segmenting pixels that it should have) and false-negative/FN (the model classified background as part of the segmentation which is also the wrong output).

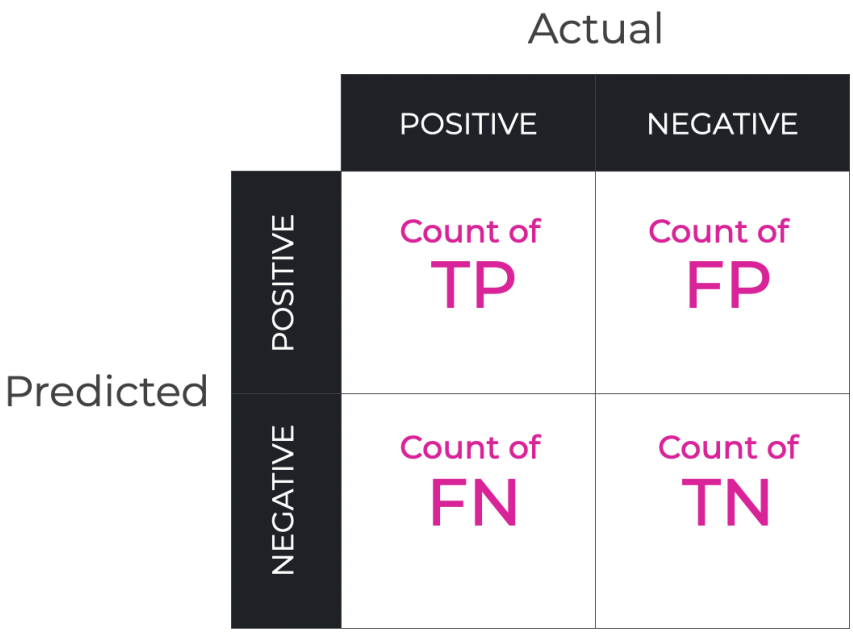

Figure 3.3: Visualisation example of a confusion matrix

Precision value is calculated as true-positive pixels pixels divided by all positive pixels. Used, for example, in spam detection, a high precision means that most of the emails classified as spam are actually spam, reducing the risk of marking important emails as spam.

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

Recall value is the true-positive results divided by the number of all samples that should have been identified as positive (true-positive and false-negative). For example, in medical diagnostics, a high recall ensures that most of the actual positive cases are correctly identified.

$$Recall = \frac{TP}{TP + FP} \tag{3.2}$$

Accuracy score, also known as Rand index is the number of correct predictions, consisting of correct positive and negative predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.3}$$

Dice coefficient is a statistical measure used to gauge the similarity between two sets. It is defined as 2*intersection divided by the total number of pixel in both images. It is commonly used in image segmentation tasks to measure the overlap between the predicted segmentation and the ground truth.

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{3.4}$$

Jaccard index (IoU/Intersection-over-Union) measures the similarity between two finite sample sets by comparing their intersection and union. It is the area of the intersection over union of the predicted segmentation and the ground truth.

$$IoU = \frac{TP}{TP + FP + FN} \tag{3.5}$$

4 Tools and Technologies

4.1 Introduction to Python and its Libraries for Machine Learning

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically typed and garbage-collected. With the risen popularity of artificial intelligence and neural networks, not programming with Python has become a disadvantage for everyone trying to develop new and state-of-the-art models, programs, and algorithms for artificial intelligence, computer vision, and semantic segmentation. This is due to the vast number of support, libraries, and frameworks available for such tasks.

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs. Based on the Torch library it is used for applications such as computer vision and natural language processing. PyTorch has become one of the most popular frameworks for research and production.

Tensor is a mathematical representation of a scalar. For example a vector is a one-dimensional tensor, a matrix is a two-dimensional tensor. Handling multiple tensors or one large-scale tensor can be difficult to comprehend, so there is an API that has a convenient library for developing with tensors called TensorFlow. Tensorflow provides tools and libraries that allow the user to create machine learning models that can run in any environment.

PIL (Python Image LIbrary) adds image processing capabilities to the Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is

designed for fast access to data stored in a few basic pixel formats. It provides a solid foundation for a general image processing tool.(dadati source)

OpenCV (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms. It is designed to provide a common infrastructure for computer vision applications

4.2 Linux, Docker, Hugging Face

Linux is family of open-sourced UNIX operating systems for computers, servers, mainframes, mobile devices and embedded devices. Linux flexible light-weight compared to other operating systems and supports almost every device.

Docker is an open-source software that enables the user to run and test software and application in an isolated environment called Docker container containing all dependencies, libraries and frameworks of choice. Docker ensures consistency across different computing environments, making it easier to manage and deploy applications. Dockerfile is a way to identify what dependencies and libraries the user wants to isolate and package into a Docker Image. By using a Dockerfile, developers can create a reproducible environment for their applications. A Docker image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. Docker images are built from Dockerfiles and are used to create Docker containers. A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another. Container images become containers at runtime and in the case of Docker containers – images become containers when they run on Docker Engine [3].

HuggingFace is an open-source provider of natural language processing (NLP) models. It has a comprehensive library of tools and pre-trained model for various tasks including but not limited to semantic segmentation. The flexibility and ease of integration offered by HuggingFace make it a valuable tool for researchers and developers working on semantic segmentation and other machine learning projects.

4.3 Overview of ROS 2 (Robot Operating System) and its Components

The Robot Operating System (ROS) is an open source set of software libraries and tools for building robot applications. From drivers and state-of-the-art algorithms to powerful developer tools [9]. ROS was created in 2007 and since a lot of changes and advances were made in the field of robotics since then in 2015 ROS 2 was created which is an upgraded version of its predecessor. ROS nodes are the cornerstone of the ROS system. They are the executable file within a ROS package. ROS nodes use a ROS client library to communicate with other nodes. Nodes can publish or subscribe to a Topic. Nodes can also provide or use a Service. ROS Topic is a structure of a data stream that can be broadcasted via ROS Publisher or received via ROS Subscriber. ROS Publisher is node type that sends a structured message to a ROS Topic. Other topics can listen to this message if they are subscribed t said message using a ROS Subscriber. Same as ROS Publisher, ROS Subscriber is also a node type but what is specific about it is that it is used to listen to published messages. The data from the subscribed messages can be processed in a function specific to the subscriber node called a callback function. Launch file is a way of structuring and deploying ROS nodes in a way where the user can define the order of execution and change the parameters of topics.

4.4 Justification for the Chosen Tools and Technologies

Python has the broadest and most extensive library collection and frameworks for artificial intelligence, image processing and semantic segmentation. Libraries such as TensorFlow, PyTorch and OpenCV offer flexible and accurate method for developing and deploying models. The Linux operating system is the only type of system that supports ROS and ROS 2 distributions. It also offers a stable and flexible work environment as well as file management. Docker was used to create a stable and consistent development environment that eliminates issues related to dependencies and configurations that can arise when running software on different hardware or operating systems and it ensures that the program will behave the same on every machine. Huggingface has an extensive library of semantic segmentation models. Huggingface allows rapid evaluation and comparison of different models.

5 Experimental Results

5.1 Presentation of Segmentation Results

The first objective after developing a dockerfile and creating the ROS 2 node that can "coat" the original image with the segmentation mask was to test it on a still image to verify that the ROS 2 node was operational and that it correctly segments the input image.

Figure 5.1: The original input for the segmentation experiment

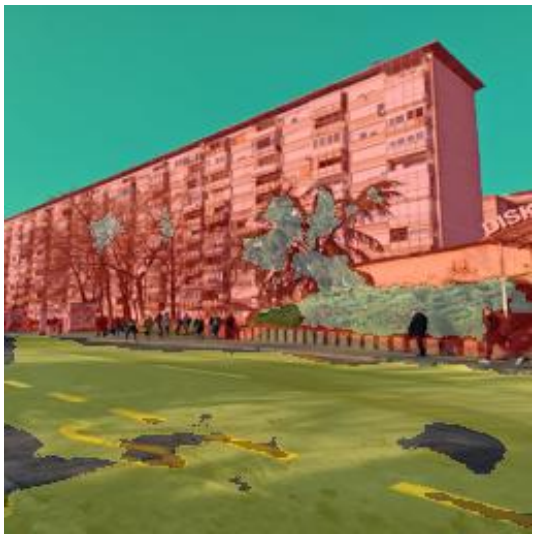

Figure 5.2: First successful experimental result of semantic segmentation on still images

The first impression of the segmented output image might be that the output image is a different dimension than its original counterpart, this is due to fixed dimension variables that were placed into the code, meaning that every image inputted into the code will have the same dimensions in output. The dimensions were hard-coded to ensure consistency in the output, which is crucial for downstream processing steps and eval-

uation. By standardizing the dimensions, the model can perform more efficiently and uniformly across different images.

After successfully creating the ROS 2 node for segmenting still images, the objective was upgraded to segmenting frames of a video stream. This required significant modifications in the code. The new task also implied that a source was needed to publish a raw video stream as a ROS 2 topic. For this purpose, the usb-cam repository from GitHub was used as a reference [11]. Testing during the day-time and night-time were conducted after completing the task.

Figure 5.3: The original input for the day-time video stream segmentation experiment

Figure 5.4: The result of the day-time video stream segmentation experiment

The daytime testing proceeded as expected, with the segmentation process successfully identifying and classifying the various elements within each frame of the video stream. The light conditions during the day provided good lighting, allowing the model to accurately segment the objects and features. The performance was consistent, with clear and precise segmentation results, demonstrating the effectiveness of the model under well-lit conditions.

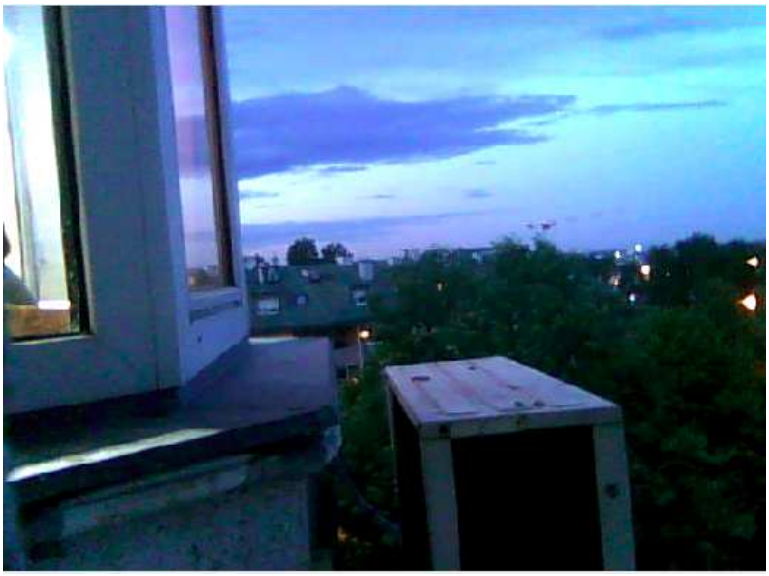

Figure 5.5: The original input for the night-time video stream segmentation experiment

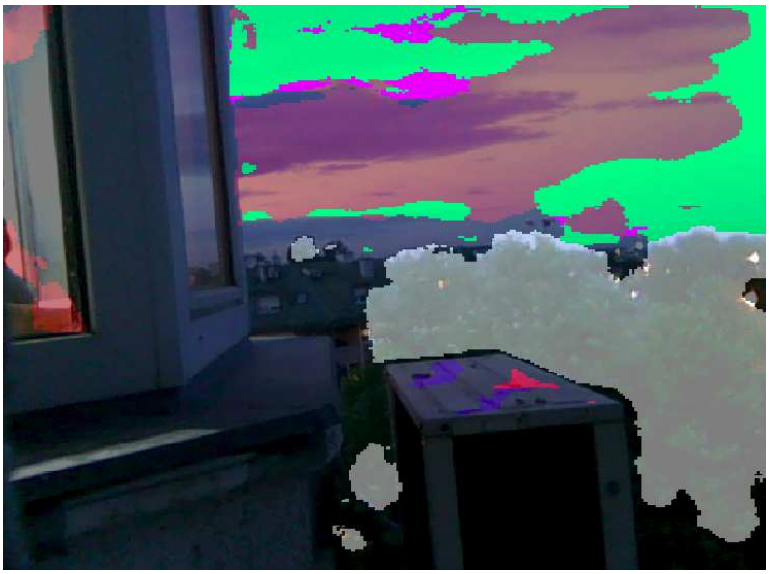

Figure 5.6: The result of the night-time video stream segmentation experiment

Testing at night presented several challenges. Unlike the daytime testing, the night-time video stream showed some patches that were not segmented correctly or were seg-

mented incorrectly. This issue was primarily due to the lack of sufficient light sources, which significantly affected the models ability to accurately identify and classify objects. It became evident that the model struggled with low-light conditions, leading to a decrease in segmentation accuracy.

5.2 Analysis of Performance Metrics

Among the three models that were more extensively used in the analysis the next metrics were observed:

Table 5.1: Performanse NVIDIA SegFormer Modela

Model	Plotting Time (s)	Prediction Time (s)
nvidia/segformer-b0-finetuned-ade-512-512	0.018 - 0.052	2.3 - 2.8
nvidia/segformer-b1-finetuned-ade-512-512	0.026 - 0.067	5.4 - 5.55
nvidia/segformer-b3-finetuned-ade-512-512	0.027 - 0.064	12.4 - 14.3

Overall, the segformer-b3 model performs the best across all the common evaluation metrics, despite having a longer prediction time compared to the segformer-b0 and segformer-b1 models. The higher computational cost is justified by its superior segmentation accuracy and reliability.

5.3 Challenges Encountered during Experiments

During testing, it was observed that the NVIDIA GPUs were not performing to their maximum capacity on the machine used. The drivers were utilizing only 14 percent of their capacity. This underutilization significantly impacted the performance of the models. Due to the computational demands of the models, the prediction time was noticeably higher because of the lack of computational power. This extended prediction time created a lag in the output, where the segmentation mask was not aligned with the current frame of the input. This issue is illustrated in 5.8, where the mask does not cover the entire sky. This misalignment is not due to a segmentation fault but rather because the

frame was updated before the prediction was completed. The more complex models required additional processing time, which exacerbated this lag. It is important to note that it is crucial to validate that the lag is smaller than the system dynamic to ensure safety. The difference between such a system and the system presented and used for experimentation is hardware capabilities.

Figure 5.7: The original input of one of the experiments where a lag between original input and segmentation was documented

Figure 5.8: The result of the experiment where the lag can be seen between the sky and the building

The increased prediction time caused by the insufficient GPU utilization led to several challenges:

- **Lag in Output:** The most significant issue was the lag between the input frames and the output segmentation masks. This lag means that the segmentation mask was often applied to a frame that had already changed, resulting in inaccurate visual representations.

- **Frame Misalignment:** As shown in 5.8, the segmentation masks did not correspond to the correct frames, causing parts of the scene, such as the sky, to be inaccurately segmented.

- **Reduced Performance of Larger Models:** Larger models, such as the nvidia/segformer-b3-finetuned-ade-512-512, required significantly more time for prediction, further compounding the issue. This extended processing time made it impractical for real-time applications, especially in scenarios requiring quick and responsive segmentation.

6 Conclusion

6.1 Summary of Achievements

This work successfully developed and implemented a semantic segmentation system using NVIDIA's Segformer models. The system was encapsulated within a Docker environment and integrated into a ROS 2 framework, enabling segmentation of both still images and video streams. The initial goal of segmenting still images was achieved, demonstrating the functionality and accuracy of the segmentation model. Following this, the system was extended to handle video streams, utilizing the usb-cam repository to publish raw video streams as ROS 2 topics. Comprehensive testing was conducted under different lighting conditions, with the system performing well during daytime tests.

6.2 Contributions to the Field

This project contributes to the field of computer vision and semantic segmentation in several ways:

- **Implementation of Huggingface libraries in a Docker Environment**
- **Implementation of Segformer Models:** By utilizing NVIDIA's Segformer models, this work demonstrates the effectiveness of hierarchical transformers for semantic segmentation tasks.
- **ROS 2 Integration:** The integration of the segmentation models within a ROS 2 framework provides a scalable approach, making it easier to incorporate into various robotic and autonomous systems.
- **Real-time Application:** Although challenges were encountered, the extension to

real-time video stream segmentation showcases the potential for real-world applications, such as autonomous driving and surveillance.

6.3 Limitations and Future Work

Several limitations were identified during the project, which also present opportunities for future work:

- **Computational Power:** The testing machine's GPUs were not utilized to their full capacity, with only 14 percent of their potential being used. This under-utilization led to higher prediction times and lag in real-time segmentation.
- **Lighting Conditions:** The system struggled with segmentation accuracy during nighttime testing due to insufficient lighting. This highlights the need for models trained specifically for low-light conditions.
- **Model Optimization:** The high computational demands of the Segformer models resulted in extended prediction times, especially for larger models like segformer-b3. Optimizing these models will be important to enhance performance.

Future work should focus on:

- **Enhancing Computational Power:** Utilizing more powerful GPUs and ensuring full GPU utilization through optimized drivers and software configurations.
- **Attaching the Model to a Vehicle:** Implementing the segmentation system on a vehicle for autonomous navigation and testing in dynamic environments.
- **Augmented Reality Integration:** Using augmented reality goggles to overlay segmentation results in real-time, providing users with enhanced situational awareness.
- **Model Training for Low-light Conditions:** Training models specifically on datasets that include low-light and nighttime conditions to improve segmentation accuracy under these challenging scenarios.
- **Batch Processing and Pipeline Optimization:** Implementing batch processing

and optimizing the data pipeline to reduce frame lag and enhance real-time performance.

By addressing these limitations and pursuing these future directions, it is possible to improve the accuracy and efficiency while removing bottlenecks of semantic segmentation systems, making them more suitable for a wide range of real-time applications.

7 References

- **MathWorks**, *Semantic Segmentation in Image Processing*,
  https://www.mathworks.com/solutions/image-video-processing/semantic-segmentation.html

- **IBM**, *Computer Vision*,
  https://www.ibm.com/topics/computer-vision

- **Sanchez Escalonilla Plaza, Alberto**, *Semantic Segmentation for Robotic Bin Picking*,
  Master's thesis, University of Twente, 2019,
  https://essay.utwente.nl/79787/1/SanchezEscalonillaPlaza_MA_EWI.pdf

- **Open Source Robotics Foundation**, *Robot Operating System (ROS) Documentation*,
  https://docs.ros.org/en/humble/index.html#

- **TechTarget**, *Linux Operating System (OS)*,
  https://www.techtarget.com/searchdatacenter/definition/Linux-operating-system

- **Docker**, *What is a Container?*,
  https://www.docker.com/resources/what-container/

- **Amazon Web Services**, *Processing Jobs Using Hugging Face*,
  https://docs.aws.amazon.com/sagemaker/latest/dg/processing-job-frameworks-hugging-face.html

- **PyTorch**, *PyTorch Documentation*,

https://pytorch.org/docs/2.3/

- **TensorFlow**, *TensorFlow Documentation*,
  https://www.tensorflow.org/

- **Pillow**, *Pillow Documentation*,
  https://pillow.readthedocs.io/en/stable/

- **OpenCV**, *OpenCV Documentation*,
  https://docs.opencv.org/4.x/

- **Mahesh, Batta**, *Machine Learning Algorithms - A Review*,
  ResearchGate, 2020,
  https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096

- **Amazon Web Services**, *What is a Neural Network?*,
  https://aws.amazon.com/what-is/neural-network/

- **BuiltIn**, *Fully Connected Layer*,
  https://builtin.com/machine-learning/fully-connected-layer

- **H2O.ai**, *Attention Mechanism*,
  https://h2o.ai/wiki/attention-mechanism/

- **Koushik, Kushal**, *Optimization Algorithms in Machine Learning: A Comprehensive Guide*,
  Medium, 2019,
  https://medium.com/@koushikkushal95/optimization-algorithms-in-machine-learning-a-comprehensive-guide-to-understand-the-concept-and-3db1df7a2f59

- **Huynh, Nghia**, *Understanding Evaluation Metrics in Segmentation*,
  Kaggle, 2020,
  https://www.kaggle.com/code/nghihuynh/understanding-evaluation-metrics-in-segmentation

- **Vaswani, Ashish et al.**, *Attention is All You Need*,
  Google Research, 2017,
  https://research.google/pubs/attention-is-all-you-need/

- **Neptune.ai**, *Image Segmentation: Introduction to Image Segmentation*,
  https://neptune.ai/blog/image-segmentation

- **Towards Data Science**, *Hierarchical Transformers*,
  https://towardsdatascience.com/hierarchical-transformers-54f6d59fa8fc

- **RF5**, *Convolution Relation*,
  https://rf5.github.io/2018/11/25/convolution-relation.html

Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[2] Qiang Chen et al. *MixFormer: Mixing Features across Windows and Dimensions*. 2022. arXiv: 2204.02557 [cs.CV].

[3] Docker, Inc. *Docker Documentation*. https://docs.docker.com.

[4] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [id='cs.CV' full$_name =' ComputerVisionandPatternRecognition' is_active = True alt_name = None in_archive =' cs' is_general = False description =' Coversimageprocessing, computervision, patternrecogni$

[5] Kai Han et al. “A Survey on Vision Transformer”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (Jan. 2023), pp. 87–110. ISSN: 1939-3539. DOI: 10.1109/tpami.2022.3152247. URL: http://dx.doi.org/10.1109/TPAMI.2022.3152247.

[6] Koushik Kushal. *Optimization Algorithms in Machine Learning: A Comprehensive Guide to Understand the Concept and Implementation*. 2021. URL: https://medium.com/@koushikkushal95/optimization-algorithms-in-machine-learning-a-comprehensive-guide-to-understand-the-concept-and-3db1df7a2f59.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[8] Batta Mahesh. “Machine Learning Algorithms - A Review”. In: *International Journal of Science and Research (IJSR)* 9.1 (2020). Licensed Under Creative Commons

Attribution CC BY, pp. 6–10. ISSN: 2319-7064. URL: https://www.ijsr.net/archive/v9i1/SR20201111316.pdf.

[9] Open Robotics. *ROS 2 Documentation*. https://docs.ros.org/en/humble/index.html.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[11] Ken Tossell. *usb_cam*. https://github.com/ros-drivers/usb_cam. 2018.

[12] Ashish Vaswani et al. "Attention is All You Need". In: 2017. URL: https://arxiv.org/pdf/1706.03762.pdf.

[13] Enze Xie et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*. 2021. arXiv: 2105.15203 [cs.CV].

Abstract

Semantic segmentation of a structured environment for mobile robots

Baraa Yassin

This project integrated a semantic segmentation system using NVIDIA's Segformer models, integrated within a ROS 2 framework and encapsulated in a Docker environment. The system successfully segmented still images and video streams, demonstrating robustness during daytime conditions but facing challenges under low-light scenarios and due to insufficient GPU utilization. Key contributions include the implementation of Segformer models and ROS 2 integration, advancing the potential for real-time applications like autonomous driving. Future work will focus on optimizing computational power, improving low-light performance, and enhancing real-time capabilities through model optimization and augmented reality integration.

Keywords: Semantic segmentation; Computer vision; ROS 2; Docker; Neural network; Artificial intelligence; Mobile robots; Machine learning; Dataset; Python; Huggingface;

Sažetak

Semantička segmentacija strukturiranih okruženja mobilnog robota

Baraa Yassin

Ovaj projekt implementirao je sustav semantičke segmentacije koristeći NVIDIA-ine Segformer modele, integrirane unutar okvira ROS 2 i enkapsulirane u Docker okruženju. Sustav je uspješno segmentirao fotografije i video tok podataka, pokazujući robusnost tijekom dnevnih uvjeta, ali suočavajući se s izazovima u uvjetima slabog osvjetljenja i zbog nedovoljne upotrebe GPU-a. Ključni doprinosi uključuju implementaciju modela Segformer i ROS 2 integraciju, unaprjeđujući potencijal za aplikacije u stvarnom vremenu poput autonomne vožnje. Budući rad će se usredotočiti na optimizaciju računalne snage, poboljšanje performansi pri slabom osvjetljenju i poboljšanje mogućnosti u stvarnom vremenu kroz optimizaciju modela i integraciju proširene stvarnosti.

Ključne riječi: Semantička segmentacija; Računalni vid; ROS 2; Docker; Neuronska mreža; Umjetna inteligencija; Mobilni roboti; Strojno učenje; Skup podataka; Python; Huggingface

8 Appendix

The Code `https://github.com/YassinBaraa/Zavrsni-rad`