Tartu University

Faculty of Science and Technology

Institute of Technology

Erik Kõiv

Leveraging neural models for data processing and analysis automation

Master's thesis (30 EAP)
Robotics and Computer Engineering

Supervisor:

Joosep Kivastik

Tartu 2023

Abstract/Resümee

Leveraging neural models for data processing and analysis automation

Unmanned Ground Vehicles (UGVs) are a staple in some industries and are entering the market in others. Development of these UGVs and their automation is resource intensive and time-consuming work. Specifically the job of processing and analysing data collected by the various sensors and cameras has so far been done by human workers. In recent years however, it has become possible to propose the automation of these tasks. This thesis describes the development of a pipeline application aimed at reducing the workload of the workers doing these jobs by leveraging neural models such as CLIPSeg, capable of zero-shot text-prompt image segmentation, to extract data from video frames based on specified classes of interest. A proof of concept demo was developed and presented to potential users, leading to the extraction of requirements for a minimum viable product (MVP). The MVP requirements included avoiding image resizing distortion, a command-line interface, and additional post-inference data analysis. The CLIPSeg model was evaluated alongside CLIPSurgery, another zero-shot image segmentation model, using a testing dataset. CLIPSeg demonstrated higher viability for the selected classes and was further evaluated using an 80% model score and 0.05% image area threshold to eliminate false positive results with great success. The final MVP application fulfilled all presented requirements and proved the viability of the CLIPSeg model for the use-case.

CERCS: T111 Imaging, image processing; T125 Automation, robotics, control engineering; T330 Military science and technology; P170 Computer science, numerical analysis, systems, control; P176 Artificial intelligence

Keywords: machine learning, machine vision, neural model, automated guided vehicle, unmanned ground vehicle, image segmentation

Närvimudelite kasutamine andmetöötluse ja -analüüsi automatiseerimisel

Mehitamata maismaasõidukid (UGVd) on mõnedes tööstusharudes laialt levinud ning teistes kanda kinnitamas. UGVde arendamine ja automatiseerimine on ressurssiderohke ja ajakulukas töö. Eriti on seda UGV sensorite ja kaamerate salvestatud andmete töötlemine ja analüüs. Hiljuti on aga tekkinud võimalus seda tööd automatiseerida. Selles magistritöös kirjeldatakse rakenduse arendamist, mille eesmärk on vähendada nende töötajate koormust kasutades närvimudeleid nagu CLIPseg, mis on võimeline tekstisisendi põhise tundmatute klasside segmentatsiooni abil videokaadritest andmeid eraldama. Töö käigus loodi rakenduse kontseptversioon, mida esitleti

võimalikele kasutajatele, et kirjeldada minimaalse elujõulise toote (MVP) nõuded ning need rakenduses täita. Need nõuded hõlmasid pildimoonutuste vältimist, käsurea liidest ja täiendavat andmete analüüsi. CLIPSeg mudelile lisaks uuriti ka teise, samuti tekstisisendi põhise segmentatsiooni võimelise mudeli CLIPSurgery võimekust kasutades testandmestikku. CLIPSegi tulemused valitud klassidega olid tugevamad, ning seda hinnati uuesti kasutades 80% tulemus- ja 0.05% pildi pindala lävendit, vähendamaks valepositiivseid tulemusi. Lõplik rakendus täitis esitatud nõuded ning tõestas CLIPSeg mudeli sobivust antud kasutusjuhuks.

CERCS: T111 Pilditehnika; T125 Automatiseerimine, robootika, juhtimistehnika; T330 Sõjandus ja militaartehnoloogia; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria); P176 Tehisintellekt

Märksõnad: masinõpe, masinnägemine, närvimudel, automatiseeritud juhitav sõiduk, mehitamata maismaasõiduk, pildisegmentatsioon

Contents

Abstract/Resümee **2**

List of Figures **6**

List of Tables **7**

Listings **8**

List of Symbols and Abbreviations **9**

1 Introduction **10**
1.1 Problem Statement 10
1.2 Objectives 11

2 Theoretical Background **12**
2.1 Hardware 12
2.2 Video Capture 12
2.3 Neural Networks 12
2.4 Image Segmentation 13
2.5 Model Evaluation 13

3 Code Development **15**
3.1 Tools Used In Code Development 15
3.2 Requirements For Proof-Of-Concept Demo 16
3.3 Development Of Proof-Of-Concept Demo 16
3.4 Requirements For MVP 20
3.5 Development Of MVP 20
3.6 Model Selection 23
3.7 Evaluation Code Development 25

4 Results **26**
4.1 Evaluation Of Selected Image Segmentation Models 26
4.1.1 Evaluation Method And Parameters 26
4.1.2 Evaluation Results 28
4.1.3 Discussion 32
4.2 Output And Capabilities Of MVP Pipeline Application 32

5 Conclusions and Future Work **34**

Acknowledgements **35**

Bibliography **36**

Appendices **39**
Appendix 1. Pos./Neg. Evaluation Results and Metrics 39
Appendix 2. CLIPSeg TP/FP/TN/FN Evaluation Results and Metrics 41

Licence **43**

List of Figures

3.1 Proof-of-concept GUI 17
3.2 Initial demo of CLIPSeg with possible classes in a possible UGV environment . 18
3.3 Kangas DataGrid with CLIPSeg output 19
3.4 Example of stitched grayscale segmentation mask 21
3.5 DataGrid with additional metadata and added thresholded boolean masks . . . 22
3.6 Visual evaluation outputs of CLIPSeg (top) and CLIP Surgery (bottom) 25

4.1 Example of indistinction between short, tall, and general grass, and brush . . . 27
4.2 Results of CLIPSeg on a possible UGV environment 28
4.3 CLIPSeg Model Evaluation; Positive/Negative 29
4.4 CLIP Surgery + SAM Model Evaluation; Positive/Negative 29
4.5 Evaluation metrics; not accounting for activation threshold; Online Images . . . 30
4.6 Evaluation metrics; accounting for 80/0.05 activation threshold; Online Images 30
4.7 Evaluation metrics; not accounting for activation threshold; Video 31
4.8 Evaluation metrics; accounting for 80/0.05 activation threshold; Video 31
4.9 Final GUI configuration 33

List of Tables

CLIPSeg Positive/Negative Evaluation Results and Metrics 39
CLIPSurgery Positive/Negative Evaluation Results and Metrics 40
TP/FP/TN/FN Evaluation Results and Metrics; No 80/0.05 Activation; Online Images 41
TP/FP/TN/FN Evaluation Results and Metrics; 80/0.05 Activation; Online Images . . 41
TP/FP/TN/FN Evaluation Results and Metrics; No 80/0.05 Activation; Video 42
TP/FP/TN/FN Evaluation Results and Metrics; 80/0.05 Activation; Video 42

Listings

3.1 Example of a possible configuration file with two prompts 22

List of Acronyms and Abbreviations

AP - Average Precision

API - Application Programming Interface

CLI - Command Line Interface

GUI - Graphical User Interface

MVP - Minimum Viable Product

POC - Proof of Concept

ROS - Robot Operating System

UGV - Unmanned Ground Vehicle

YAML - YAML Ain't Markup Language

1 Introduction

Unmanned Ground Vehicles (UGVs) have become or are becoming an extremely popular choice in various industries, such as manufacturing [1], warehousing [2], medicine [3], and the military [4]. These industries develop and use UGVs to make labour intensive tasks less so or to reduce or eliminate human labour requirements as a whole. These tasks include, but are not limited to, transporting medical samples and guiding patients [3], transporting items in warehouses [2], and assisting human operatives in performing surveillance and logistical missions [4].

Developing these UGVs and implementing and improving the various levels of vehicle autonomy [5] is time-consuming and resource-intensive work. Specifically processing and analysing the data collected by the various sensors and cameras present on the UGVs is currently usually accomplished by large numbers of human workers or volunteers. While various tools [6] have been created to assist human workers in the minutiae of the above-mentioned tasks, only recently has it become possible to propose automation of these processing and analysis tasks in a way as to make the part of human workers to act as quality assurance or provide slight improvements in accuracy or cohesiveness of the final results.

1.1 Problem Statement

As mentioned above, the processing and analysis of the recorded data is time-consuming and resource-intensive work. One minute of recordings from the various sensors from a Milrem UGV provides approximately one gigabyte of raw data to be processed, most of which is image data in the form of video. So far, this has been a job for human workers in a specific department. Some public tools [7] have been created to assist in the minutiae of the general work, but the general process has not been automated; workers still need to manually analyse the video for metadata, trace segmentation masks, and compile the datasets to be made available to the various development teams.

In the past couple years researchers have explored the possibilities of training neural networks to automatically perform image (video frame) segmentation based on text previously unknown to the trained network, known as zero- or one-shot text-prompt segmentation [8], or other input such as point coordinates [9] to specify regions of interest. Applying these novel techniques and technologies would make it possible to greatly reduce the workload of workers doing data processing and analysis currently, and allow reduction and reassignment of resources locked in this area, which would speed up development in other areas, and allow opening avenues to new ones. These are the reasons why Tartu Observatory is working with Milrem AS to develop a pipeline application that would allow automatic preliminary data processing and analysis of data generated by Milrem AS UGVs.

1.2 Objectives

The thesis has two main goals: developing a code pipeline to input data from Milrem UGVs and output data structures containing image segmentation masks and specified related image metadata, and evaluate cutting-edge approaches to accomplishing generation of image segmentation masks in the frame of Milrem AS requirements. These two goals in turn divide into the following:

- getting requirements for a proof-of-concept demo from Milrem AS
- creating the proof-of-concept demo based on the requirements above
- gathering feedback from potential users and/or integrators of the demo
- developing an MVP based on the requirements condensed from the feedback
- finding applicable cutting-edge implementations of image segmentation
- evaluating and testing above implementations and technologies based on the requirements

2 Theoretical Background

This chapter introduces the reader with to terminology and the general technologies used in the thesis. This will cover video capture, neural networks, image segmentation, and model evaluation, along with limited related hardware terminology.

2.1 Hardware

Device in the context of neural model inference refers to the device which the processing is done on.

2.2 Video Capture

Encoding is the compression of a video file based on an encoding algorithm or **codec**. [10]

Decoding is the decompression of a video file based on an encoding alogrithm or **codec**. [10]

H264 is a lossy video compresison algorithm or **codec**. [10]

Keyframes are frames where the codec algorithm has detected enough change in the scene to warrant storing the entire frame as opposed to just the changes between two frames. [10]

2.3 Neural Networks

Image segmentation divides an image into segments where each pixel or set of pixels in the image is mapped to a label. [11]

Model is a neural network that has been trained on a dataset, altering the internal parameters of the network to allow for the recognition patterns, ascribed to specific labels depending on the input. [12]

Model checkpoint is a partial dump of the model parameters at a certain point in time, containing in addition to the model's internal learned parameters also the training or learning parameters such as learning rate [13]. A model checkpoint is preferred when continued training of the same model is expected at some point in the future.

Model weights are a smaller subset, with reduced size, of the model parameters than the model checkpoint [13], from which it is generally not possible to continue training as effectively. Model weights are preferred when the model is to be used for inference only, or training on a different dataset is expected.

Transformer is a special machine learning model that can track context in sequential data such as videos or text, by using internal attention functions. Attention functions use **encoders** to convert text or video frames to vectors, assigning each word or image feature a relevancy value, or weight, which a **decoder** can use with its own corresponding internal values to, for example, translate a sentence or summarize a video. [14]

2.4 Image Segmentation

Semantic segmentation allows for segmentation of an image into detected classes, but does not distinguish between instances of the same class. [15]

Instance segmentation allows for segmentation of an image into detected classes, while also distinguishing between instances of the same class. [15]

Segmentation mask is an image where each pixel or set of pixels is mapped to a class either as a confidence level or boolean value. [15]

Text prompt is an input to a neural network in the form of text.

Zero-shot segmentation concerns segmentation of classes that were not present in the training data, based on estimating using features present in the training data. [16]

One-/multi-shot segmentation is segmentation of classes that were not present during training of the model, but for which one or more examples, from which the model incurs minor changes, have been provided during inference. [17]

2.5 Model Evaluation

True positive, or TP, means the model correctly detected the class. [18]

False positive, or FP, means the model incorrectly detected the class. [18]

True negative, or TN, means the model correctly did not detect the class. [18]

False negative, or FN, means the model incorrectly did not detect the class. [18]

Precision [18] is calculated as,

$$Precision = \frac{TP}{TP + FP} \tag{2.1}$$

Recall [18] is calculated as,

$$Recall = \frac{TP}{TP + FN} \tag{2.2}$$

F1 score accounts for both false positives and false negatives [18], and is calculated as,

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{2.3}$$

3 Code Development

The code development chapter aims to describe the process of step-by-step creation of the pipeline application and the reasoning behind the choices made during the process. This means describing the process of gathering feedback from potential users and/or integrators of the demo, and extracting the requirements for the MVP. The chapter will also describe applicable implementations of text-prompt image segmentation, and the process of creating an evaluation suite for said implementations and technologies in the Milrem AS frame of requirements.

3.1 Tools Used In Code Development

The entire project was developed on **Ubuntu 20.04 LTS** using **Python 3.8.10**. These were chosen as the 20.04 release of Ubuntu was approximately in the middle of its lifetime [20], thus increasing probability of general support for any current, but also previous generation libraries and programs that could be used during development. In addition, the version of ROS used on the UGVs is not supported on Ubuntu 22.04. Python is also one of the most popular programming languages for data analysis, neural networks, and quick prototyping.

The main libraries used in the project were as follows:

PyAV - a media manipulation library for Python, used to ease conversions to Numpy and Pillow formats. [21]

rosbag - a ROS built-in Python library to read ROS bag files. [22]

Kangas - an Estonian made multimedia data structure library for Python. [23]

Numpy - an expansive numerical computation library for Python. [24]

Pandas - a Python data analysis library. [25]

Pillow - a Python image manipulation library. [26]

PyQt5 - a library to build GUIs in Python. [27]

PyTorch - a commonly used Python machine learning library. [28]

Jupyter Notebook - a Python environment for quick prototyping and data analysis. [29]

Some of these libraries have unmentioned dependencies, which, along with the specific versions for each library, are included in the **requirements.txt** file in the project repository. Python built-in libraries are not included in the requirements file or generally described in the scope of this work.

3.2 Requirements For Proof-Of-Concept Demo

The requirements from the proof-of-concept demo are very concise and straightforward, as they stem from the main problem of the thesis: to provide a pipeline application to data processing workers that would allow them to automatically generate highlights, related metadata, and rudimentary segmentation masks for image data recorded by Milrem UGVs.

The use of language to define requirements is specific, and some words that will be used here and in 3.4 are defined for clarity:

- ”SHALL” and ”SHALL NOT” indicate a mandatory requirement.
- ”SHOULD” and ”SHOULD NOT” indicate a recommendation.
- ”WILL” indicates a statement of fact or intention. [19]

The requirements are as follows:

- The application shall take as input a ROS bag file which contains sensor data from a single UGV mission.
- These bag files shall be selectable by the user.
- The user shall be able to select the video channels in the bagfile to be processed.
- The user shall be able to provide a list of classes, considered a prompt, to be used in the segmentation process.
- The selected video channel will be processed.
- The resulting segmentation masks and timestamps shall be saved to a data structure.

3.3 Development Of Proof-Of-Concept Demo

A basic GUI was created using PyQt5 to allow the user to select the bag file and video channel to be processed, and to provide the prompt for the segmentation process. It was decided to make the application with a GUI to make it more user-friendly, as the target users are not necessarily familiar with command-line interfaces. The resulting GUI can be seen in Figure 3.1.

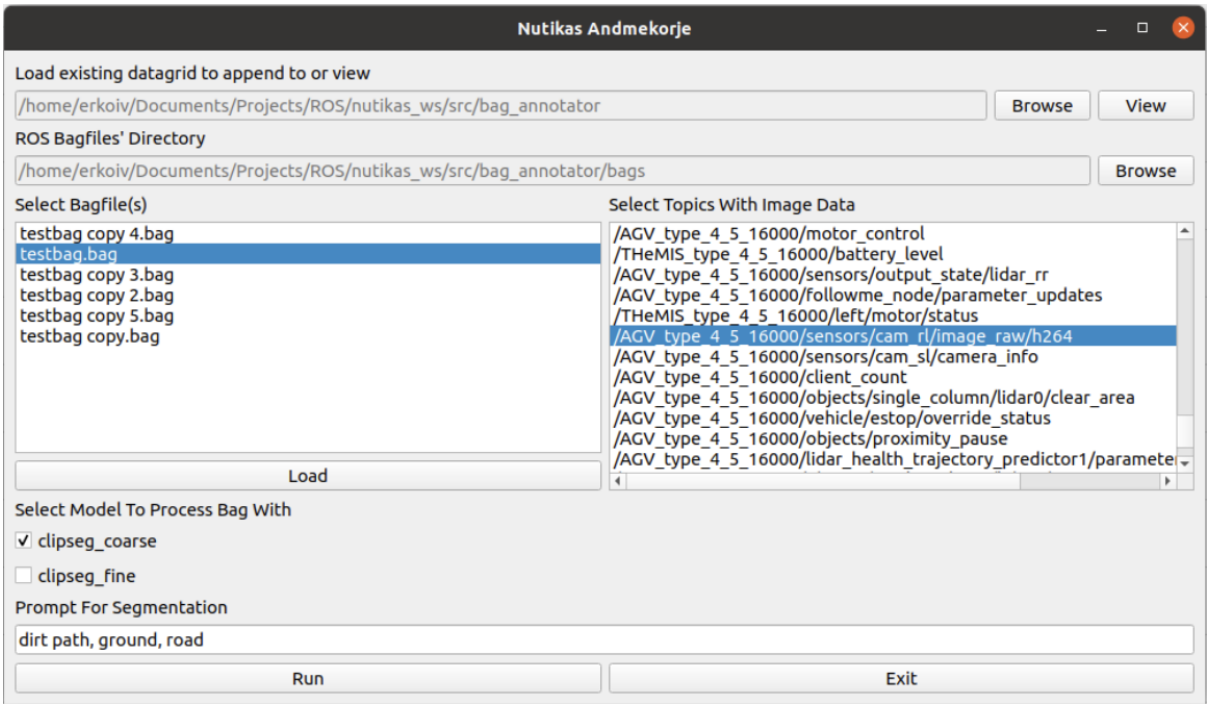

Figure 3.1: Proof-of-concept GUI

To process the image data from the user-selected video channel, it was necessary to extract the video frames from the channel. Due to the types of cameras used in the Milrem UGVs, the video frames are encoded using H264 encoding. Because the nature of how ROS handles data, as individual ROS specific messages for each collection of datapoints, each frame is already its own separate package, ready to be read and decoded. This was done using PyAV.

Each frame can then be passed into an image segmentation model that should generate a segmentation mask. For the proof-of-concept application, the model chosen was the CLIPSeg model [32], more closely described in 3.6. Specifically CLIPSeg was chosen for its zero-shot segmentation functionality, and the fact that it is built on OpenAI's [30] CLIP model [31] as a backbone, which is a novel, yet well-established model, with OpenAI's reputation helping garner interest from neural network researchers. A preliminary test was done using the CLIPSeg included demo code, to verify that the model was truly able to zero-shot segment unknown classes, and was implementable into custom code.

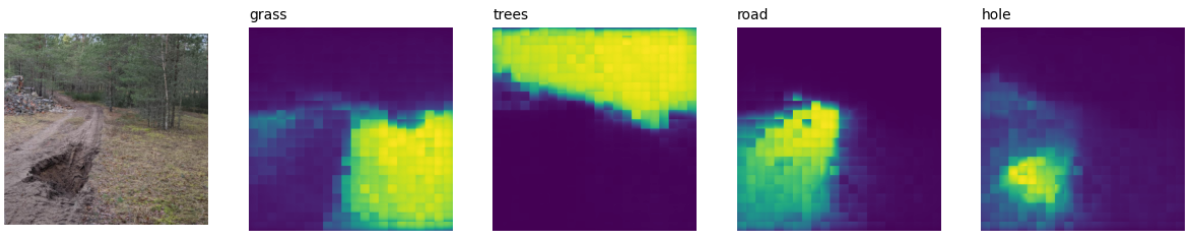

Figure 3.2: Initial demo of CLIPSeg with possible classes in a possible UGV environment

During testing of the concept code, it was found that CLIPSeg was able to process only square images, and the output segmentation masks were also square. The UGV camera output however, was 1080p, thus not rectangular. This was noted as a future improvement, but for the proof-of-concept demo, the images were resized to be square. This was done using Pillow.

It was also noted that there were many extracted frames that were virtually identical to each other in content due to 60 FPS recording, but a relatively slowly moving UGV. There was no periodic way to guess into the future which frames would be useful, but luckily, the automatically pre-applied H264 codec, used to reduce storage space of the video channels, also encoded keyframes, which allowed for a way to filter out the duplicate frames. This was done by checking the frame's keyframe flag, and only processing the frames that matched, because the flag is only enabled on a frame where the encoding algorithm detects sufficient scene change. This was accomplished using PyAV.

The output greyscale masks of CLIPSeg were saved into a Kangas DataGrid (Figure 3.3), which is a further development on the Pandas DataFrame, specifically designed to also handle multimedia content, such as images. It is also convertable to a DataFrame, which was something Milrem AS was ready to accept as a final output.

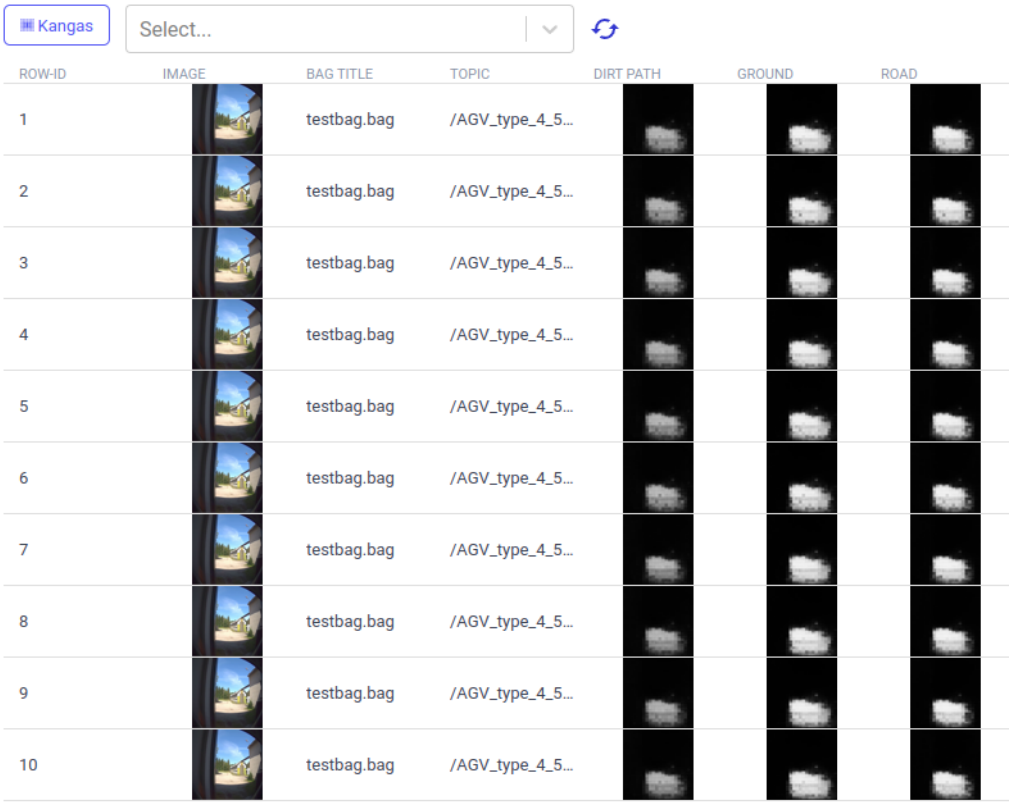

Figure 3.3: Kangas DataGrid with CLIPSeg output

This was considered a successful proof-of-concept demo, as it fulfilled all the requirements set out for it. The next step was to gather feedback from potential users and/or integrators of the demo, and to extract the requirements for the minimum viable product (MVP).

3.4 Requirements For MVP

To get the requirements for the MVP, firstly a demonstrative video was created and distributed alongside the demo code to potential users. Along with it was a request to review the user experience and provide feedback on possible additional features that would be useful to them.

Based on the distributed demo, a meeting was set up with possible users and integrators to discuss it and the future MVP requirements more closely. From this meeting the final requirements for the MVP were distilled:

- Both the ROS message types *sensor_msgs/Image* and *sensor_msgs/CompressedImage* shall be supported.
- There shall be a way to filter ROS topics, as there are many of them due to the multitude of sensors on the UGV.
- Images shall keep their aspect ratio to avoid distortion.
- Additional metadata shall be added to the output:
  - Max score achieved in the segmentation mask.
  - Possibility to threshold the mask based on model score to create boolean masks.
  - Timestamps of the frames.
  - Coverage percentage of the mask in the frame.
- A CLI shall be added.
- Parameters for the CLI shall be configurable via a configuration file.

3.5 Development Of MVP

To develop the MVP, firstly, support for both *sensor_msgs/Image* and *sensor_msgs/CompressedImage* was already there by happenstance, because the ROS built-in API for bag files that was used to interact with the ROS bag is ROS message type agnostic, and the H264 compression is a given for the video channels, stemming from the UGV architecture.

Secondly, a keyword inclusion filter was added via string comparison to filter out any topics that might not be relevant to the user.

Thirdly, there was no way to force CLIPSeg to accept a non-square input image, so a workaround was implemented. A dynamic method was added to the code to pre-process the image before passing it to CLIPSeg. More specifically, the top of each image from the bag file was cropped to make the aspect ratio 2:1, which allowed for the image to be split to two squares, that would then be processed by CLIPSeg. It was decided to crop the top pixels of the image as they would theoretically contain the least useful information to the UGVs' continued operation. The resulting masks were then stitched back together to form a single mask without any distortion. The caveats to this process are discussed in 4.1.3.

It should also be noted that the resulting code is image resolution agnostic in most cases of common display resolutions [33], to accommodate for possible differences in camera aspect ratios.

The setback with this approach was that instead of a single processing operation, two are used instead, doubling the processing time. This was considered acceptable as eliminating image distortion was decided to be more important. The added positive side-effect was also that the processed image was effectively twice the resolution it would be otherwise, allowing for more detail in the result.

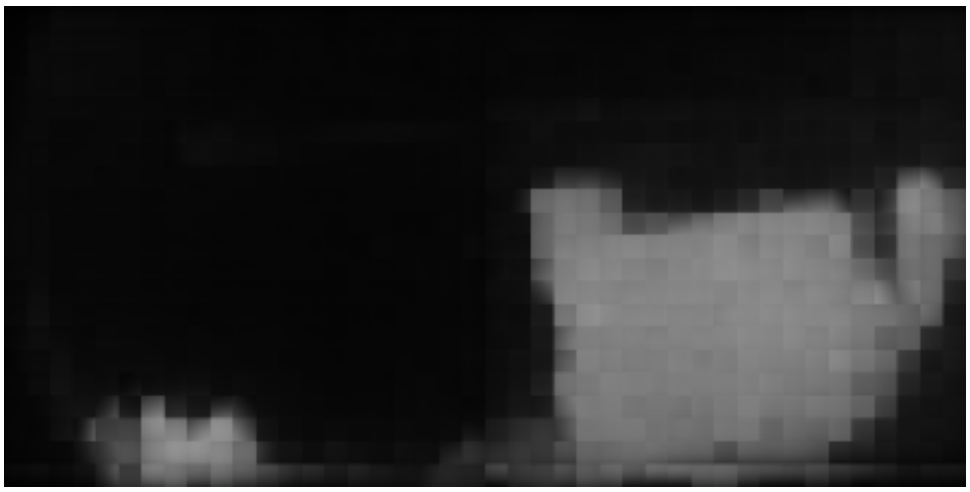

Figure 3.4: Example of stitched grayscale segmentation mask

The next step was to add the additional metadata. The max score was the maximum value in the mask, and the coverage percentage was calculated by dividing the number of pixels with a value above the threshold by the total number of pixels in the mask. The timestamps were already available from the ROS bag file, so they were added to the metadata with minor internal processing. The thresholding was a matter of comparing the mask values to the threshold value, and setting the value to 1 if it was above the threshold, and 0 if it was below. This was done using the Pytorch function *torch.where* [34].

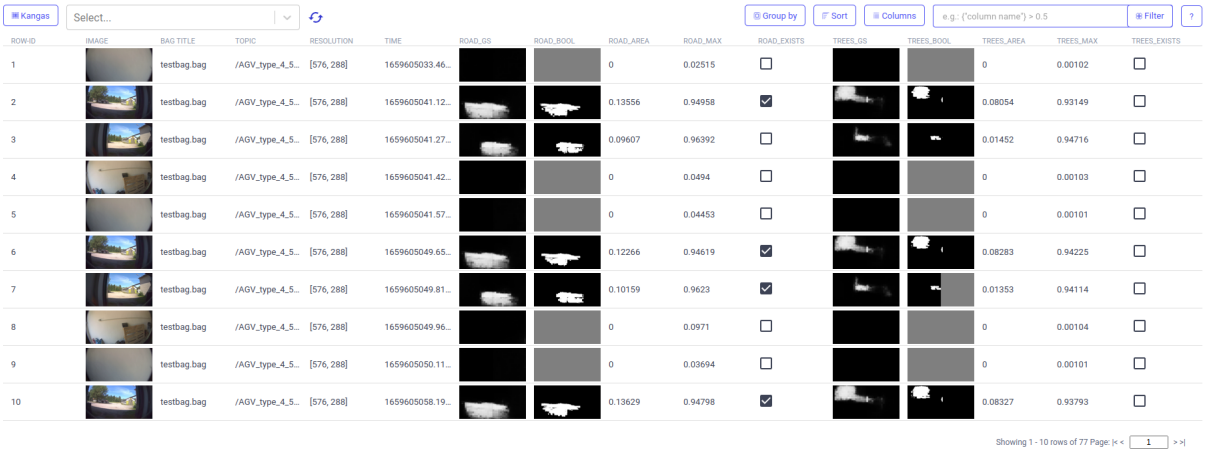

Figure 3.5: DataGrid with additional metadata and added thresholded boolean masks

The final step was to add the CLI. This was done using the Python built-in library *argparse*. The CLI was made to accept the same parameters as the GUI, and the parameters were made configurable via a config file in the YAML format [35]. The YAML format was chosen for ease of parsing the file into Python using the built-in library *yaml*. The final configuration file is shown in 3.1.

```
inputf: ~/Documents/Projects/Observatory/bag_annotator/bags
   # Requires ~/folders format
outputf: ~/Documents/Projects/Observatory/bag_annotator/
   results # Requires ~/folders format
bagfilter: testbag 3 # inclusion keywords
topicfilter: cam_rr/image # h264 # inclusion keywords
model: coarse # fine
prompts:
road:
   - 0.8 # min score of detection area to be counted as
      prompt
   - 0.05 # min % of image to be counted as "existing"
sky:
   - 0.8
   - 0.05
probs: false
```

```
masks:
save: true
type: both # gs (greyscale), bool (black-white) or both
resize: 0.3
```

Listing 3.1: Example of a possible configuration file with two prompts

In addition to the explicit requirements of the MVP, the performance of the application was greatly improved in this phase, by reducing the amount of DataFrame copying operations, parallelizing the ROS bag file reading, and only decoding the H264 frame as it gets passed to the segmentation model for processing. The MVP final full processing sequence speed was approximately 15-20 minutes per gigabyte of bag file size, result over 10 runs of 6 gigabyte files, depending on amount of keyframes.

With the classes/tags specified in 4.1.1 after removing the ones that CLIPSeg was not able to detect, the average processing time per single image for CLIPSeg was approximately 10 seconds on an Nvidia A3000 GPU set as PyTorch execution device, calculated during runtime, with added overhead for the in-processing of the data from the bag file(s).

3.6 Model Selection

Although CLIPSeg was the initial subject of development and testing for the POC and later more sophisticated evaluation, during the entire development process other candidates of similar functionality were sought and considered. These, and the reasoning behind there selection or rejection, mainly due to lack of code documentation, are described next.

Image Segmentation Using Text and Image Prompts - CLIPSeg [32]

The authors of the CLIPSeg model aimed to create a system that can generate image segmentations based on arbitrary text prompt input at inference time. They built on a CLIP model backbone, extending the functionality with a transformer-based decoder. The model is described as having both zero- and multi-shot capabilities, and working code along with sufficient documentation was made available, making it a prime candidate for the POC and later evaluation.

However, during development of the POC and later the MVP, it was discovered that documentation was provided only for the zero-shot capabilities, and both the training and multi-shot code was left for interested readers to reverse-engineer in order to implement. Despite this, due to easily implemented zero-shot capabilities, and an 82.3% AP score achieved on the Pascal-5i dataset, the model was still selected for the MVP and later evaluation.

Zero-Shot Semantic Segmentation - ZS3Net [8]

The ZS3Net model was developed with the aim of providing a zero-shot semantic segmentation model that can be used to segment images based on unseen classes. The model is based on a

ResNet-101 backbone and has a self-training step at inference time, which was of great interest as a half-way between zero- and multi-shot, further supported by promising demonstrative images. Code with well documented training and evaluation process was also made available, once more showing promise of possibility of deeper development and implementation. However, due to the relatively large age of the research, complications with dependent libraries and lacking documentation arose when implementing an initial workable demo of the model, disqualifying it from further consideration due to rapid development needs.

Language-Driven Semantic Segmentation - LSEG [38]

The LDSS model takes a very similar approach to CLIPSeg, instead using either ViT [36] or ResNet [37] backbones. However, possibly due to the difference in backbone and encoder-decoder structure, showed worse performance on the same Pascal-5i dataset, and lacking a workable demo, was not considered for further development.

Modulated Detection for End-to-End Multi-Modal Understanding - MDETR [39]

The MDETR model sparked interest due to being partially backed by Facebook AI Research group, but was discovered to have a complex multi-step training process for which it lacked consolidated documentation and reasonably implementable inference code. In addition, the two following models appeared to be, at the very least, it's spritual successors, and thus were considered instead.

Segment Anything Model - SAM [9]

Taking once more a very similar approach to CLIPSeg, the SA Model was developed to use image and prompt enoders along with a mask decoder to provide zero-shot segmentation capability. Specifically interesting were the reported 1.1 billion mask training dataset, and the promise of instance segmentation for future development of the pipeline application considered in the work at hand. However, testing the demo code provided revealed that the specific implementation was largely aimed at aiding manual segmentation, specifically using input points from user clicks to generate the masks. As the more hands-off approach of text-prompting for directing of the segmentation capabilities was desired, but not sufficiently described by the code documentation, despite being shown in the paper itself, and the model being made public at a late point in the MVP development process, it was not considered for further development. These shortcomings were however improved upon by the final considered model.

CLIP Surgery for better Explainability with Enhancement in Open-Vocabulary Tasks [40]

CLIP Surgery aims to dissect the CLIP model, and improve some parts, such as noisy detections, while leveraging the underlying capabilities to generate rudimentary segmentation masks, named heatmaps by the authors. These heatmaps are then used, to generate both positive and negative points, described in the SAM section, and fed into SAM to generate the masks based on those points, making use of the very large training dataset SAM was trained on. This combines the text-prompting capabilities of CLIPSeg with the mask outputs of SAM, courtesy of using both positive and negative segmentation inputs. The code provided by the authors of CLIP Surgery also deeply and clearly demonstrates its capabilities, opening up the possibility of complex inspired implementation. For these reasons, this model was given a preliminary evaluation, such as CLIPSeg, but due to its recency, was not included in the secondary, more in-depth evaluation.

3.7 Evaluation Code Development

The evaluation code was developed in parallel to the evaluation process in 4.1. This was done in the Jupyter Notebook environment, as it allows for easy code execution and visualization of results. The evaluation code was developed in a modular fashion, with each model having its own evaluation implementation based on the respective demo code provided by the model developers. This amounted to similar, but not identical outputs from the evaluated models as seen in Figure 3.6. All the evaluated models are listed in 3.6.

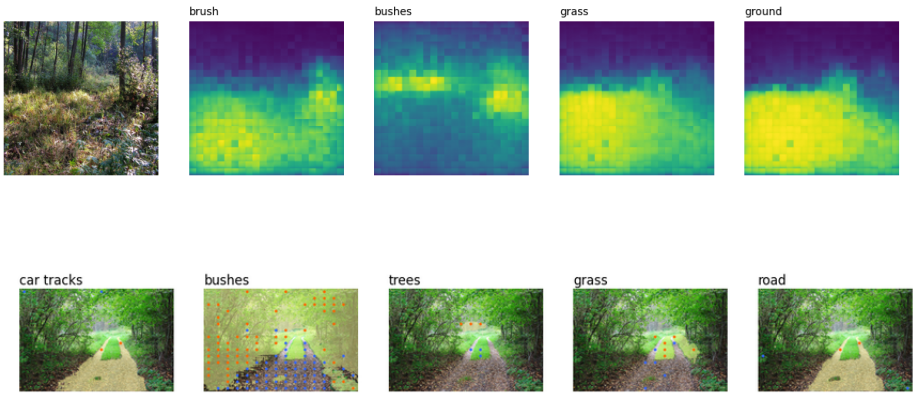

Figure 3.6: Visual evaluation outputs of CLIPSeg (top) and CLIP Surgery (bottom)

The homogenized output is then passed into a singular evaluation Notebook for visual categorisation using basic image displaying and keypress recording for recording evaluations.

At first the evaluation code was written to categorise output images into Positive and Negative occurrences only, but this was found to be too coarse, as it did not allow for the calculation of the F1 score to account for both precision and recall values. Despite this, the Positive and Negative evaluation method was used to crudely evaluate preliminary results. However, for the final evaluation of CLIPSeg, the output images were categorized into True Positive, True Negative, False Positive, and False Negative to allow for the calculation of recall, precision, and F1 score.

4 Results

This chapter aims to present and describe the results of the evaluation of the selected image segmentation models, along with demonstrating the output of MVP of the created pipeline application for processing ROS bag files.

4.1 Evaluation Of Selected Image Segmentation Models

Evaluation was conducted on only two models, although several others were considered and are listed as such in 3.6, of which the two selected were the most novel, and it was possible to run the respective demos without encountering extensive bugs and library errors, meaning it was also possible to re-implement the demos for the purpose of evaluation.

4.1.1 Evaluation Method And Parameters

The methods of evaluation were already briefly touched upon in 3.7, but to reiterate, each occurrence of a detected class was categorised first into Positive and Negative occurrences for a crude preliminary evaluation, and later into TP, TN, FP, or FN, from which the precision, recall, and F1-score were calculated.

The classes, or tags, used during evaluation, were provided by Milrem AS during the process of developing the MVP. These tags were as follows:

- tree stump
- rocks
- fallen tree
- bushes
- tall/high grass
- short/low grass
- obstacles
- rubble
- trash/garbage
- holes/pits
- water
- ditches/trenches
- piles of dirt/mud
- mounds of dirt
- crop fields
- roads
- gravel road/path
- forest trail/path
- car tracks

During testing of the MVP, it was discovered that the CLIPSeg model, inherited from the proof-of-concept, was not able to detect the tags of ”tall/high/short/low grass”, ”obstacles”, ”trash/garbage”, ”holes/pits”, ”water”, ”ditches/trenches”, ”piles/mounds of dirt/mud” and ”crop fields” very well. Some tags were eliminated based on visual testing before clear empirical analysis, however, the low performance of others can be seen in Figure 4.3. Based on those negative results, more were eliminated from further testing to reduce the amount of tags and thus greatly reduce the amount of time iterating evaluation would take. Some negative results in the form of segmentation masks are visible in Figure 4.1.

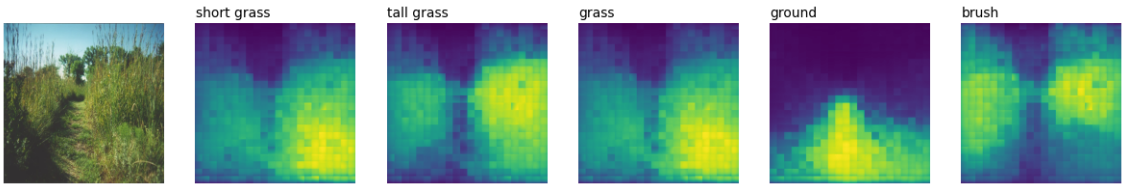

Figure 4.1: Example of indistinction between short, tall, and general grass, and brush

The remaining tags/classes were as follows with a few additions:

- brush
- bushes
- car tracks
- fallen tree
- forest path
- grass
- gravel path
- ground
- road
- rocks
- rubble
- tree stump
- trees
- walking path

In addition to simply visually evaluating the unprocessed result masks, an 80/0.05 activation threshold was also applied to augment the results. The 80/0.05 activation method consists of counting any pixels with over 80% confidence, and if this number amounts to 0.05% or more of all pixels on the image, then the class/tag is considered to be present in the image. This method was discovered to greatly reduce the amount of false positives that accounted for most of the results that were considered negative, since many false positives were weak detections, meaning they had low confidence.

All of the evaluated models were tested on one set of images taken from Google image search to provide clear representations of the classes/tags in question unachievable by the varied and unspecific existing set of ROS bag file data available from the Milrem database. In addition, processing the ROS bag files added significant overhead to the iterative development process. Search queries for these images were simply each of the classes.

To provide a more real-world, yet still targeted, approach to the evaluation process, CLIPSeg was also tested on a video of compiled clips, recorded specifically to contain examples of a varied set of the classes. This recording process was conducted on public grounds in the Tartu city and Tõravere grounds. The video still conforms with the expected 1080p resolution and the H264 encoding, essentially mimicing the UGV image output.

4.1.2 Evaluation Results

Proof Of Concept Evaluation Of CLIPSeg

In the proof of concept phase CLIPSeg was the only model explored and showed promising results on visual examination of some possible class labels in a possible UGV environment. The results of this can be seen in Figure 3.2 and another is provided below.

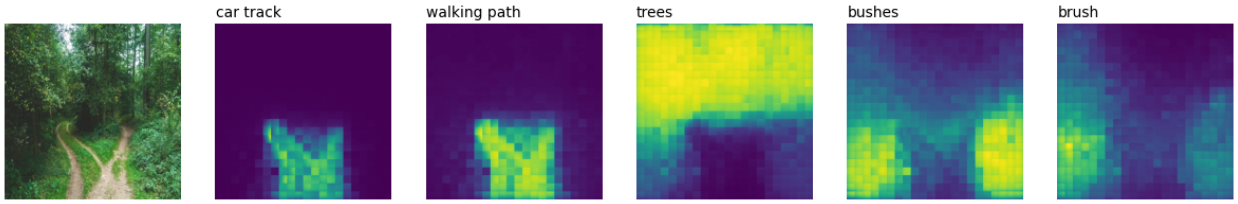

Figure 4.2: Results of CLIPSeg on a possible UGV environment

Evaluation Of CLIPSeg and CLIP Surgery On Ideal Class Examples

Extending the evaluation of CLIPSeg and adding CLIP Surgery, both models were evaluated on a 68 image set of ideal class examples of online images, as described in 4.1.1, considering only Positive and Negative evaluations, and applying no further processing of the results. The results of this can be seen in Figure 4.3 and Figure 4.4.

The most positive results are achieved in classes such as trees, bushes, and road/path related classes, where it can be seen that CLIPSeg trumps CLIP Surgery.

As these results were achieved during the development of the MVP, some of the initial classes/tags hypothetically provided by Milrem AS are also evaluated, and their low performance is visible, specifically in tags which were evaluated in large numbers, yet still showed low performance, such as any ditch/trench related tags and underbrush as an alternative to bushes. Leaning on these results, the list of tags was reduced to more viable options only.

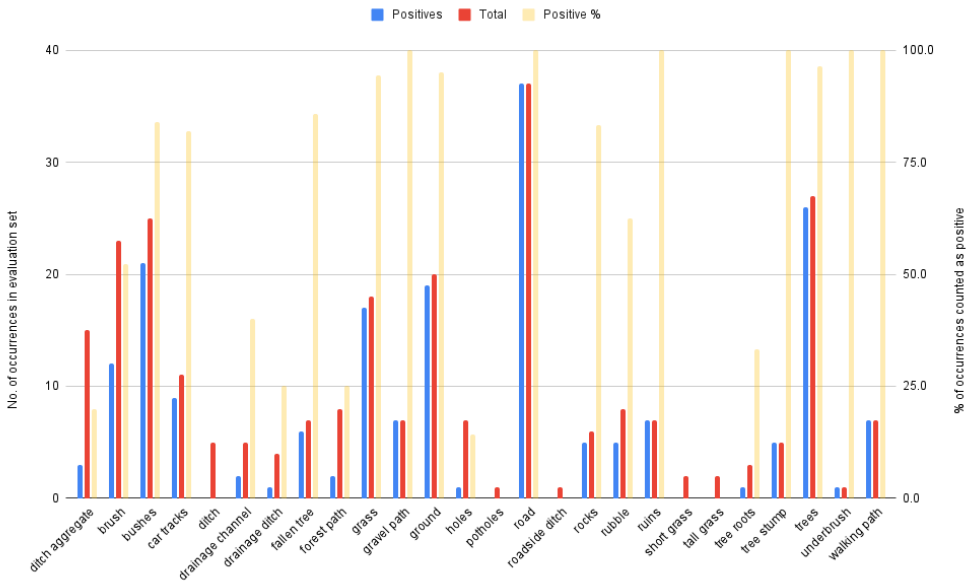

Figure 4.3: CLIPSeg Model Evaluation; Positive/Negative

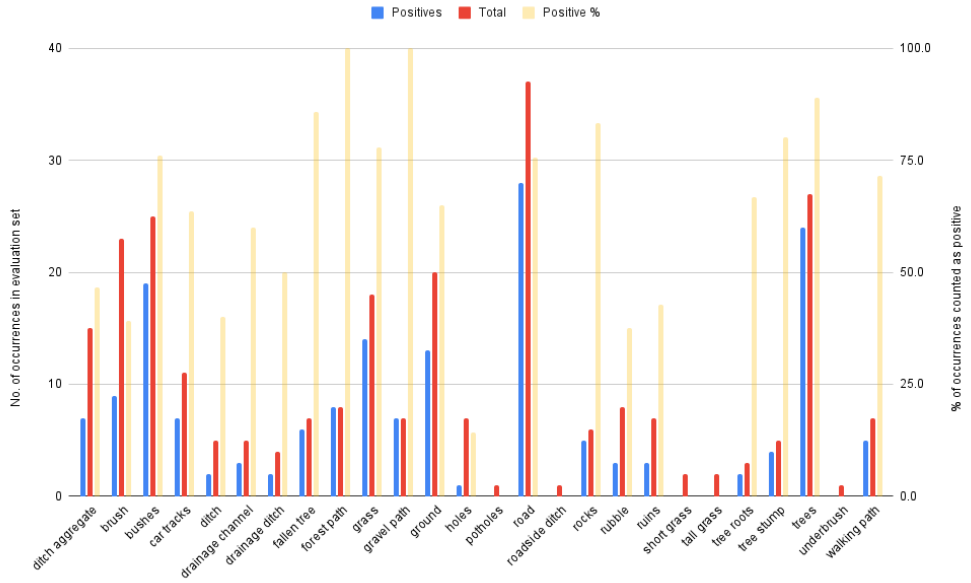

Figure 4.4: CLIP Surgery + SAM Model Evaluation; Positive/Negative

Secondary Evaluation Of CLIPSeg

CLIPSeg was then evaluated again on all 68 images, but this time all classes were sought from all images, rather than what was known and expected to be in them, increasing the number of occurrences for some common classes such as trees, bushes, and roads greatly. Also, the 80/0.05 activation threshold principle described in 4.1.1 was applied. The results of this can be seen in Figure 4.5 and Figure 4.6. Viewing the graphs, the contrast between results thresholded and those not is clear. The false positives have been reduced overwhelmingly, from 342 without thresholding, down to 6 afterwards.

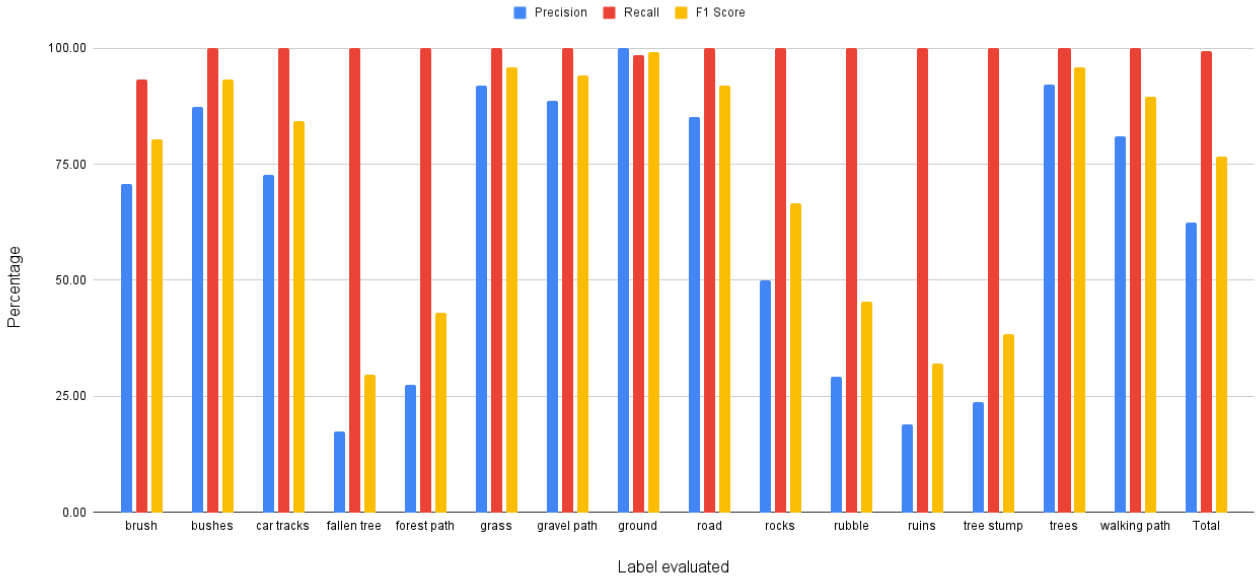

Figure 4.5: Evaluation metrics; not accounting for activation threshold; Online Images

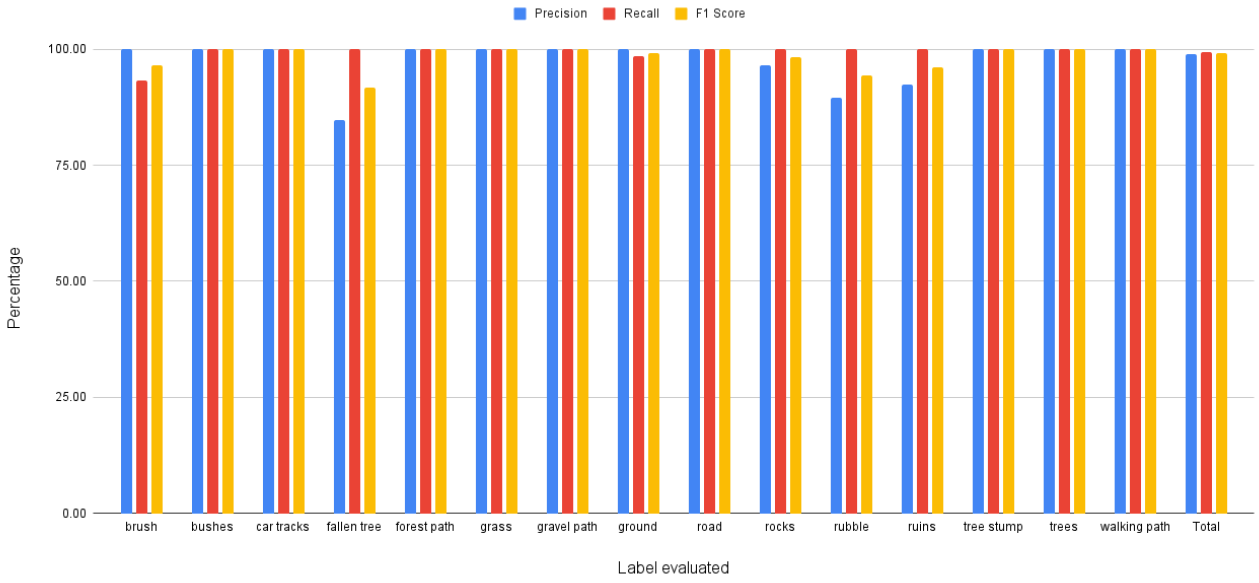

Figure 4.6: Evaluation metrics; accounting for 80/0.05 activation threshold; Online Images

Lastly, CLIPSeg was evaluated on a video of compiled clips from a more real-world setting, and results are provided both with and without applying the 80/0.05 activation threshold principle described in 4.1.1. The results of this can be seen in Figure 4.7 and Figure 4.8. Same changes are visible here as in the evaluation on the set of ideal images - false positives all but disappear, from 375 before, to 3 afterwards. The fallen tree class/tag was not found on the video, and thus the metrics cannot be calculated in this case, however raw data shows all False Positives were converted to True Negatives by the thresholding.

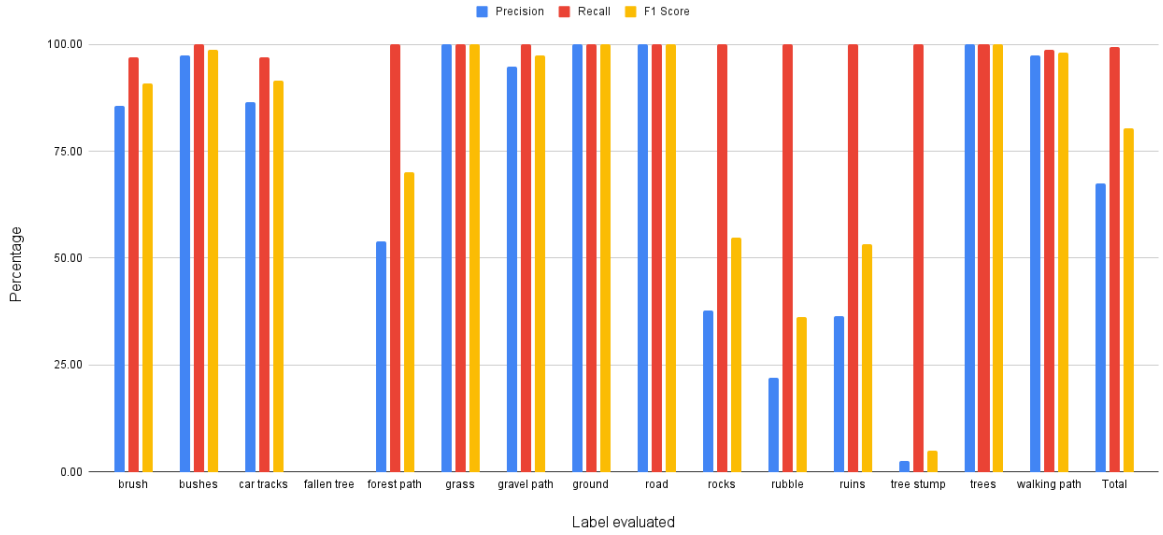

Figure 4.7: Evaluation metrics; not accounting for activation threshold; Video

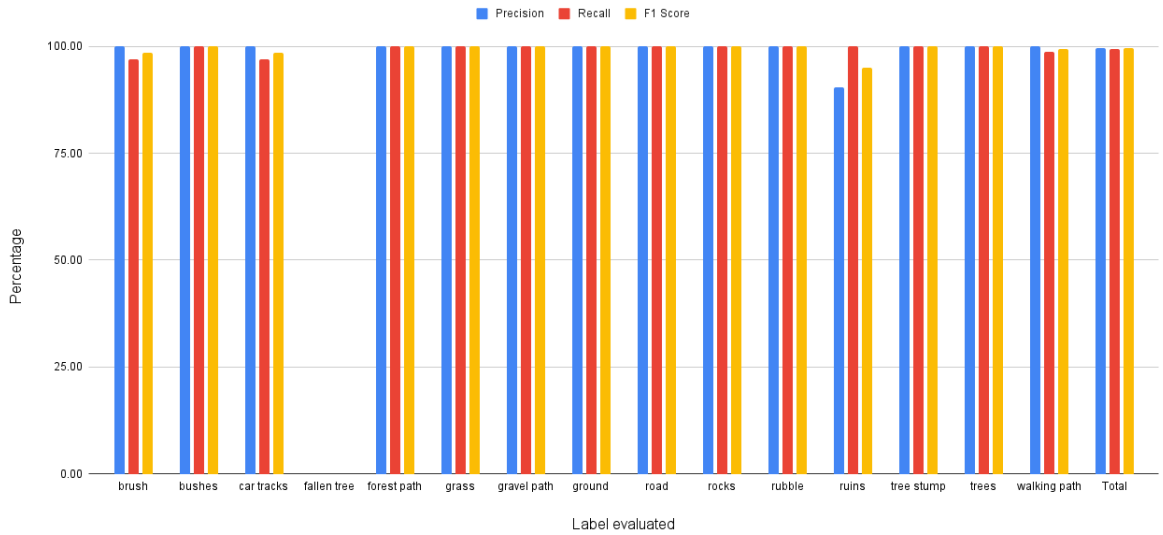

Figure 4.8: Evaluation metrics; accounting for 80/0.05 activation threshold; Video

4.1.3 Discussion

It must be noted, that the evaluations were not done on a hard ground truth set by creating a dataset of annotated images and comparing the results against it, but visually evaluating results for each class, taking as a basis whether the class detection activation was distinguishable to the human eye. It is believed that this is a sufficient method of evaluation to demonstrate the capabilities of the two explored models, and further explore the performance of CLIPSeg on the specific set of classes/tags provided by Milrem AS, to use as a basis for further development.

Furthermore, considering the outputs of the application itself, as seen in Figure 3.5, the output images that have been squared and then stitched together often have artefacting in the center due to CLIPSeg detections being generally weaker at the edges of the images. This is suspected to stem from the fact that most training photos are normal, generic photos where the subject, then to be segmented, is centered on the photo for natural composition, leading to the model being inconfident more-so around the edges of the image. However, it should not impact the overall goal of the application, since the detections are generally only weaker, rather than completely absent.

4.2 Output And Capabilities Of MVP Pipeline Application

The capabilties of the final MVP pipeline application are as follows:

- A GUI, seen in 4.9, available to:
  - Select one or multiple bagfiles to be scanned for image channels automatically
  - Select between lower and higher resolution CLIPSeg options, lowering and increasing time cost and workload respectively
  - Define any list of prompts that should be looked for in the selected image streams
  - Configure settings for processing the output data, such as detection area required to count a class as 'detected', and confidence threshold for generating a boolean mask
  - Configure options for generating masks, such as saving the probability matrix used for generation, and which type of masks to save
- A CLI configured by a YAML file as seen in 3.1 capable of the same options for more process automation availability
- Generation of greyscale masks
- Generation of boolean masks based on set confidence threshold
- Generation of basic metadata such as timestamps, maximum confidence, and area of mask above set confidence
- Setting of a class detection flag based on area of mask above set confidence
- Live and post-run viewing of the results, using the Kangas DataFrame integration, as seen in Figure 3.5
- Sorting, grouping, and filtering of the results, using the Kangas DataFrame integration

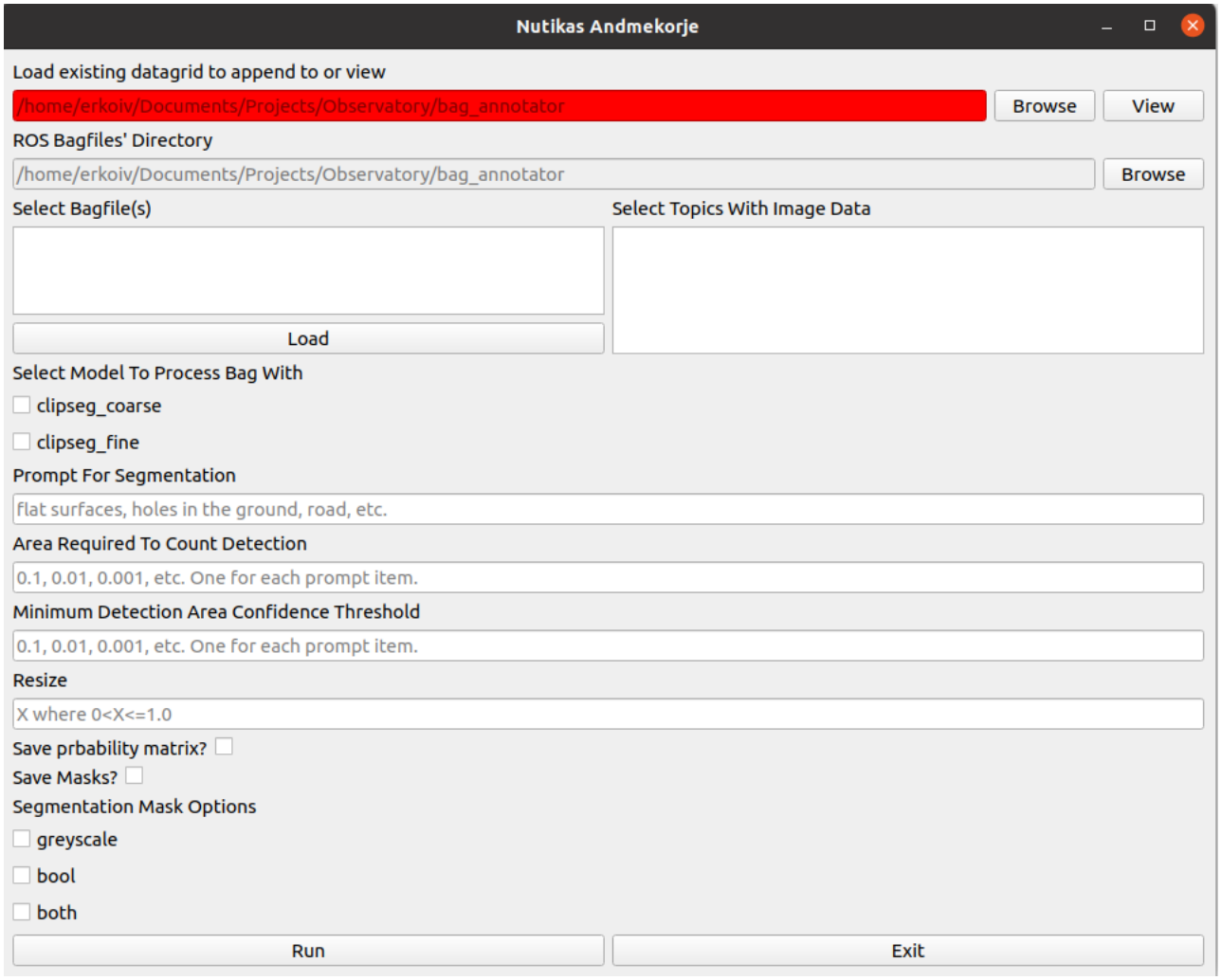

Figure 4.9: Final GUI configuration

5 Conclusions and Future Work

In this thesis a pipeline application was developed to reduce the workload of processing video recorded by UGVs by leveraging neural models for image data processing and analysis. For this, proof of concept expectations were received from Milrem AS, from which a working demo was developed. This demo was realized as a GUI application capable of taking in ROS bag files, and processing the video data within. The processing was accomplished by using CLIPSeg, a neural model capable of zero-shot text-prompt image segmentation, to extract data from the video frames based on classes of interest provided by Milrem AS. After presenting the POC demo to possible users and integrators, and gathering a meeting with them, requirements for a minimum viable product were extracted. These requirements included avoiding image resizing based distortion, adding a command-line interface to the application and providing additional post-inference data analysis, such as confidence-thresholded boolean masks, and detected class coverage data. During the development of the MVP, research on multiple other neural models with similar functionality to CLIPSeg was explored, and selections based on performance and implementability were made to evaluate one other model - CLIPSurgery. The evaluation process saw the gathering of a testing dataset composed of 68 online images clearly representing one or more of the provided classes, and counting the positive and negative segmentation results of both models. This showed the positive viability of some of the classes, and negative results from others, based on which some low-performing classes were discarded. Positive results were seen to slightly favor the CLIPSeg model. The evaluation process was then refined to count TP, FP, TN, and FN results, and CLIPSeg was once more evaluated on the testing dataset, this time looking for all classes from all images. Results further showed viability in the selected classes. In addition, a further processing step was added - an 80/0.05 activation thresold, seeing the elimination of all FP results where the amount of pixels above 80% confidence did not cross 0.05% of the total pixels in the image. This was found to nearly eliminate the false positive results. As a final evaluation a test video was compiled from clips taken in the Tartu and Tõravere area, to provide a more real-setting. These results confirmed the results from the evaluation on the testing dataset. The final MVP application fulfilled all presented requirements, and the evaluation process showed the viability of the CLIPSeg model.

The project will continue until August 2023 at the least. Further development will see fine-tuning of the confidence-area activation threshold for each class individually, the gathering of more, and varied, real-world data, and the creation of a hard ground-truth testing dataset for further evaluations. In addition, the custom implementation of CLIPSurgery will be more deeply explored as an alternative to CLIPSeg. The final goal is to integrate the application into the Milrem AS UGV data gathering pipeline.

Acknowledgements

I would like to thank my supervisor Joosep Kivastik for keeping me on track with the things I didn't want to do, and Mihkel Pajusalu who connected me with the work you read in this thesis. Furthermore, I am grateful to my extroverted friends from Bachelor's studies Abdullah Siddiqui and Adam Larger, who set me on the path to be where I am now.

This work was conducted in the scope of the project "Rakendusuuring metsauuendussüsteemi Robotic Forester juhtimisautonoomia suurendamiseks (NutikasMetsarobot)", which was supported by the European Union European Regional Fund.

Signature / Allkiri:

Erik Kõiv

Bibliography

[1] International Society of Automation - Automated guided vehicles improve production
https://www.isa.org/intech-home/2018/july-august/features/automated-guided-vehicles-improve-production 19.05.2023

[2] Rocla AGV Solutions - Warehouse Logistics AGVs
https://rocla-agv.com/agv-solution/automated-guided-vehicles/warehouse-logistics-vehicles/ 19.05.2023

[3] Robert Valner, Houman Mansavi, Igor Rybalskii, Rauno Põlluäär, Erik Kõiv, Alvo Aabloo, Karl Kruusamäe, Arun Kumar Singh, "Scalable and heterogenous mobile robot fleet-based task automation in crowded hospital environments—a field test", *Frontiers in Robotics and AI*, **Volume 9**, 2022, 10.3389/frobt.2022.922835
https://www.frontiersin.org/articles/10.3389/frobt.2022.922835/full 19.05.2023

[4] Milrem Robotics - THe THeMIS UGV
https://milremrobotics.com/defence/ 19.05.2023

[5] United States National Highway Traffic Safety Administration
https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety 19.05.2023

[6] Kollmorgen - How Does Kollmorgen Utilize Data Analytics for AGVs?
https://www.kollmorgen.com/en-us/blogs/how-does-kollmorgen-utilize-data-analytics-agvs 19.05.2023

[7] Roboflow
https://roboflow.com/ 19.05.2023

[8] Maxime Bucher, Tuan-Hung Vu, Matthieu Cord, Patrick Pérez, "Zero-Shot Semantic Segmentation", *Computer Vision and Pattern Recognition*, arXiv:1703.06870v3, 2018, No DOI
https://arxiv.org/abs/1906.00817 19.05.2023

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, Ross Girshick, "Segment Anything", *Computer Vision and Pattern Recognition*, arXiv:2304.02643, 2023, 8-9, No DOI
https://arxiv.org/abs/2304.02643 19.05.2023

[10] IBM - What is Video Encoding? Codecs and Compression Techniques
https://blog.video.ibm.com/streaming-video-tips/what-is-video-encoding-codecs-compression-techniques/ 19.05.2023

[11] TensorFlow - Image Segmentation
https://www.tensorflow.org/tutorials/images/segmentation 19.05.2023

[12] Microsoft - What is a machine learning model?
https://learn.microsoft.com/en-us/windows/ai/windows-ml/what-is-a-machine-learning-model 19.05.2023

[13] Keras - ModelCheckpoint
`https://keras.io/api/callbacks/model_checkpoint` 19.05.2023

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need", *Computation and Language*, arXiv:1706.03762, 2-4, 2017, No DOI
https://arxiv.org/abs/1706.03762 19.05.2023

[15] Roboflow - Semantic Segmentation vs. Instance Segmentation: Explained
`https://blog.roboflow.com/difference-semantic-segmentation-instance-s`
19.05.2023

[16] Roboflow - What is Zero Shot Learning in Computer Vision?
https://blog.roboflow.com/zero-shot-learning-computer-vision/

[17] Archit Parnami, Minwoo Lee, "Learning from Few Examples: A Summary of Approaches to Few-Shot Learning", *Machine Learning*, arXiv:2203.04291, 2022, No DOI
https://arxiv.org/abs/2203.04291

[18] Google Cloud - Evaluate the performance of processors
https://cloud.google.com/document-ai/docs/workbench/evaluate 19.05.2023

[19] ECSS system - Glossary of terms (ECSS-S-ST-00-01C), Tech. rep., European Cooperation for Space Standardization (Oct. 2012).

[20] Canonical - Ubuntu release cycle
https://ubuntu.com/about/release-cycle 19.05.2023

[21] PyAV - PyAV Documentation
https://pyav.org/docs/develop/index.html 19.05.2023

[22] ROS - rosbag
http://wiki.ros.org/rosbag 19.05.2023

[23] Kangas - README
https://github.com/comet-ml/kangas/blob/main/README.md 19.05.2023

[24] NumPy - NumPy Documentation
https://numpy.org/doc/stable/ 19.05.2023

[25] pandas - pandas documentation
https://pandas.pydata.org/docs/ 19.05.2023

[26] Pillow - Pillow (PIL Fork) Documentation
https://pillow.readthedocs.io/en/stable/index.html 19.05.2023

[27] PyPi - PyQt5
https://pypi.org/project/PyQt5/ 19.05.2023

[28] PyTorch - PyTorch Documentation
https://pytorch.org/docs/stable/index.html 19.05.2023

[29] Jupyter - Jupyter Project Documentation
https://docs.jupyter.org/en/latest/ 19.05.2023

[30] OpenAI - About
https://openai.com/about 19.05.2023

[31] OpenAI - CLIP: Connecting Text and Images
https://openai.com/research/clip 19.05.2023

[32] Timo Lüddecke, Alexander S. Ecker, "Image Segmentation Using Text and Image Prompts", *Computer Vision and Pattern Recognition*, arXiv:2112.10003, 2021, No DOI
https://arxiv.org/abs/2112.10003 19.05.2023

[33] HP - What are typical monitor sizes?
https://www.hp.com/us-en/shop/tech-takes/what-are-typical-monitor-sizes 19.05.2023

[34] PyTorch - torch.where
https://pytorch.org/docs/stable/generated/torch.where.html 19.05.2023

[35] YAML - YAML Ain't Markup Language
https://yaml.org/ 19.05.2023

[36] Hugging Face - ViT: Vision Transformer
`https://huggingface.co/docs/transformers/model_doc/vit`
19.05.2023

[37] Hugging Face - ResNet
`https://huggingface.co/docs/transformers/model_doc/resnet`
19.05.2023

[38] Boyi Li, Kilian Q. Weinberger, Serge Belongie, Vladlen Koltun, René Ranftl, "Language-Driven Semantic Segmentation", *Computer Vision and Pattern Recognition*, arXiv:2201.03546, 2022, No DOI
https://arxiv.org/abs/2201.03546 19.05.2023

[39] Aishwarya Kamath, Mannat Singh, Yann LeCun, Gabriel Synnaeve, Ishan Misra, Nicolas Carion, "MDETR: Modulated Detection for End-to-End Multi-Modal Understanding", *Computer Vision and Pattern Recognition*, arXiv:2104.12763, 2021, No DOI
https://arxiv.org/abs/2104.12763 19.05.2023

[40] Yi Li, Hualiang Wang, Yiqun Duan, Xiaomeng Li, "CLIP Surgery for Better Explainability with Enhancement in Open-Vocabulary Tasks", *Computer Vision and Pattern Recognition*, arXiv:2304.05653, 2023, No DOI
https://arxiv.org/abs/2304.05653 19.05.2023

Appendices

Appendix 1. Pos./Neg. Evaluation Results and Metrics

CLIPSeg Positive/Negative Evaluation Results and Metrics

Unique labels	Positives	Total	Positive %
ditch aggregate	3	15	20.0
brush	12	23	52.2
bushes	21	25	84.0
car tracks	9	11	81.8
ditch	0	5	0.0
drainage channel	2	5	40.0
drainage ditch	1	4	25.0
fallen tree	6	7	85.7
forest path	2	8	25.0
grass	17	18	94.4
gravel path	7	7	100.0
ground	19	20	95.0
holes	1	7	14.3
potholes	0	1	0.0
road	37	37	100.0
roadside ditch	0	1	0.0
rocks	5	6	83.3
rubble	5	8	62.5
ruins	7	7	100.0
short grass	0	2	0.0
tall grass	0	2	0.0
tree roots	1	3	33.3
tree stump	5	5	100.0
trees	26	27	96.3
underbrush	1	1	100.0
walking path	7	7	100.0

CLIPSurgery Positive/Negative Evaluation Results and Metrics

Unique labels	**Positives**	**Total**	**Positive %**
ditch aggregate	7	15	46.7
brush	9	23	39.1
bushes	19	25	76.0
car tracks	7	11	63.6
ditch	2	5	40.0
drainage channel	3	5	60.0
drainage ditch	2	4	50.0
fallen tree	6	7	85.7
forest path	8	8	100.0
grass	14	18	77.8
gravel path	7	7	100.0
ground	13	20	65.0
holes	1	7	14.3
potholes	0	1	0.0
road	28	37	75.7
roadside ditch	0	1	0.0
rocks	5	6	83.3
rubble	3	8	37.5
ruins	3	7	42.9
short grass	0	2	0.0
tall grass	0	2	0.0
tree roots	2	3	66.7
tree stump	4	5	80.0
trees	24	27	88.9
underbrush	0	1	0.0
walking path	5	7	71.4

Appendix 2. CLIPSeg TP/FP/TN/FN Evaluation Results and Metrics

TP/FP/TN/FN Evaluation Results and Metrics; No 80/0.05 Activation; Online Images

Label	False Negative	False Positive	True Negative	True Positive	Precision	Recall	F1 Score
brush	3	17	2	41	70.69	93.18	80.39
bushes	0	8	0	55	87.30	100.00	93.22
car tracks	0	15	8	40	72.73	100.00	84.21
fallen tree	0	52	0	11	17.46	100.00	29.73
forest path	0	45	1	17	27.42	100.00	43.04
grass	0	5	1	57	91.94	100.00	95.80
gravel path	0	7	1	55	88.71	100.00	94.02
ground	1	0	0	62	100.00	98.41	99.20
road	0	9	2	52	85.25	100.00	92.04
rocks	0	28	7	28	50.00	100.00	66.67
rubble	0	41	5	17	29.31	100.00	45.33
ruins	0	51	0	12	19.05	100.00	32.00
tree stump	0	48	0	15	23.81	100.00	38.46
trees	0	5	0	58	92.06	100.00	95.87
walking path	0	11	5	47	81.03	100.00	89.52
Total	4	342	32	567	62.38	99.30	76.62

TP/FP/TN/FN Evaluation Results and Metrics; 80/0.05 Activation; Online Images

Label	False Negative	False Positive	True Negative	True Positive	Precision	Recall	F1 Score
brush	3	0	19	41	100.00	93.18	96.47
bushes	0	0	8	55	100.00	100.00	100.00
car tracks	0	0	23	40	100.00	100.00	100.00
fallen tree	0	2	50	11	84.62	100.00	91.67
forest path	0	0	46	17	100.00	100.00	100.00
grass	0	0	6	57	100.00	100.00	100.00
gravel path	0	0	8	55	100.00	100.00	100.00
ground	1	0	0	62	100.00	98.41	99.20
road	0	0	11	52	100.00	100.00	100.00
rocks	0	1	34	28	96.55	100.00	98.25
rubble	0	2	44	17	89.47	100.00	94.44
ruins	0	1	50	12	92.31	100.00	96.00
tree stump	0	0	48	15	100.00	100.00	100.00
trees	0	0	5	58	100.00	100.00	100.00
walking path	0	0	16	47	100.00	100.00	100.00
Total	4	6	368	567	98.95	99.30	99.13

TP/FP/TN/FN Evaluation Results and Metrics; No 80/0.05 Activation; Video

Label	False Negative	False Positive	True Negative	True Positive	Precision	Recall	F1 Score
brush	2	11	0	65	85.53	97.01	90.91
bushes	0	2	0	76	97.44	100.00	98.70
car tracks	2	10	2	64	86.49	96.97	91.43
fallen tree	0	78	0	0	0.00	0.00	0.00
forest path	0	36	0	42	53.85	100.00	70.00
grass	0	0	0	78	100.00	100.00	100.00
gravel path	0	4	0	74	94.87	100.00	97.37
ground	0	0	0	78	100.00	100.00	100.00
road	0	0	3	75	100.00	100.00	100.00
rocks	0	48	1	29	37.66	100.00	54.72
rubble	0	60	1	17	22.08	100.00	36.17
ruins	0	49	1	28	36.36	100.00	53.33
tree stump	0	75	1	2	2.60	100.00	5.06
trees	0	0	0	78	100.00	100.00	100.00
walking path	1	2	1	74	97.37	98.67	98.01
Total	5	375	10	780	67.53	99.36	80.41

TP/FP/TN/FN Evaluation Results and Metrics; 80/0.05 Activation; Video

Label	False Negative	False Positive	True Negative	True Positive	Precision	Recall	F1 Score
brush	2	0	11	65	100.00	97.01	98.48
bushes	0	0	2	76	100.00	100.00	100.00
car tracks	2	0	12	64	100.00	96.97	98.46
fallen tree	0	0	78	0	0.00	0.00	0.00
forest path	0	0	36	42	100.00	100.00	100.00
grass	0	0	0	78	100.00	100.00	100.00
gravel path	0	0	4	74	100.00	100.00	100.00
ground	0	0	0	78	100.00	100.00	100.00
road	0	0	3	75	100.00	100.00	100.00
rocks	0	0	49	29	100.00	100.00	100.00
rubble	0	0	61	17	100.00	100.00	100.00
ruins	0	3	47	28	90.32	100.00	94.92
tree stump	0	0	76	2	100.00	100.00	100.00
trees	0	0	0	78	100.00	100.00	100.00
walking path	1	0	3	74	100.00	98.67	99.33
Total	5	3	382	780	99.62	99.36	99.49

Non-exclusive licence to reproduce the thesis and make the thesis public

I, Erik Kõiv,

1. grant the University of Tartu a free permit (non-exclusive licence) to:

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Leveraging neural models for data processing and analysis automation

2. grant the University of Tartu the permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from 25/05/2028 until the expiry of the term of copyright,

3. am aware that the author retains the rights specified in points 1 and 2.

4. confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Erik Kõiv
20/05/2023