GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

CIDAS Campus-Institut Data Science

Internship Report: Exploring Methods for Prompting the Segment Anything Model without Manual Interaction

Georg Eckardt
Matriculation number: 23112861
Sixth Semester of Applied Computer Science Bachelor
July 2024
Supervised by Dr. Sharmita Dey
Written at the Research Group of Prof. Dr. Alexander Ecker

Contents

1 Introduction

Segmentation tasks are core to the field of computer vision. They can be used in a wide variety of applications including medical image analysis, autonomous driving and robotics. Image segmentation technology permits to "deconstruct" pictures into individual regions. This is the basis to find the precise outlines of individual objects. One of the most recent models in this field is the Segment Anything Model (SAM), a state-of-the-art promptable segmentation model developed and published by Meta in 2023 [**sam**]. The SAM model provides zero-shot learning transfer to previously unseen image domains through the use of prompts. Due to the cross domain usability, SAM is an attempt to create a foundation model for segmentation tasks. One of its restrictions, is that SAM requires human prompts for domain transfer. The objective of this internship is to identify fully automated prompts for SAM that yield optimal segmentation performance. To assess performance of SAM, in combination with selected prompting methods, different image domains were evaluated. The task of this study was to confirm the effectiveness of refined models against two datasets: First, the Densely-Annotated Video Segmentation (DAVIS) dataset, which consists of short real world video sequences. Second, the MOVI dataset, which displays short videos of computer generated objects. Over the course of the study a wide variety of computer vision approaches were tested and evaluated on the respective dataset.

2 Segment Anything Model

SAM's architecture comprises three different transformers: SAM uses one large image encoder for feature extraction and is pretrained using a Masked Autoencoder. The second transformer is used to encode prompts, which could support a wide variety of prompt types. Current implementations can only be prompted by bounding boxes or a list of positive points (inside the object) and negative points (outside the object). However, point prompts allow to segment only one object, while the use of bounding boxes allows the prediction of multiple objects. Both embeddings (image and prompt) are then passed to a decoder that calculates the output. Advantage of this architecture is that the prompt encoder and the decoder are much smaller than the image encoder. That allows using SAM also in a web application. The image embeddings will be extacted on a server and send to the client. All further computations can be executed in the web-brower of the client. In addition to prompts, SAM can also segment the entire image on its own [**sam˙github**]. SAM can be run in three different model sizes H, L and B, where H is the largest and B is the smallest. SAM was trained in three stages, each with an increasing amount of self supervision. The final dataset, which was published with the model comprises of

11 million images and more than one billion masks and was published along with model and the paper.

3 DAVIS Segmentation

3.1 DAVIS Challenge

The DAVIS Challenge [**davis**] was a video segmentation challenge from 2016 to 2020. Each year with an with an updated dataset. Participating teams were given time to train their models on the dataset. Their solutions were submitted to additional unlabeled data at the end of the challenge. This work used the 2017 challenge dataset. It contains 150 videos, typically 50 to 80 frames long, showing people, animals and other objects such as cars in motion. Sample images are shown in Figure 1. Displayed are the first, middle and last frame of the video, along with the segmentation mask for the last frame. The 2017 dataset, unlike its predecessor, often contains more than one mask, but only segments the primary moving objects. The challenge is designed as a semi-supervised segmentation task, where the model was given the entire video and the first segmentation mask. The task was to then to create all subsequent masks in the following frames of the video. The authors of the challenge proposed two metrics for evaluating the results, a per-mask Intersection over Union (IoU) and a bipartite matching loss of boundary pixels.

Figure 1: Sample Images from the DAVIS dataset

3.2 Task

The DAVIS Challenge poses a different question then examined in this study. All best performing models in the challenge original used mask tracking techniques to track the objects from the previous frame. This method was not in the scope of this internship. Therefore, metrics proposed by the authors of the challenge could not

be utilized for evaluation, because they would not provide misleading results. Using the metrics proposed in the challenge for evaluation, predicting additional objects would cause very low scores (see Figure 3 left column). The research question of the internship was about general segmentation performance of SAM without any prior human prompts. Therefore the model was not shown any masks, but was tasked with segmenting all objects present in the image. Instead of the proposed metrics, two different versions of an IoU were used. These will be called "Mask Only per Object IoU" (MOpOIoU) and "Weighted Mask Only per Object IoU" (WMOpOIoU). MOpOIoU calculates the average of the highest IoU values for each annotated mask. It should therefore remove all object masks that are segmented, but not annotated in the dataset. In general WapoIoU works the same, but weights the masks by their respective sizes. MOpOIoU should be a good indicator of the overall mask quality expected from the combination of YOLO and SAM. WMOpOIoU can be compared to MOpOIoU as a measure of how mask size affects mask quality.

3.3 YOLO

YOLO (You Only Look Once) is a real-time object detection model first published in 2016 [**yolo1**] and since then developed by Ultralytic's. Unlike previous models, YOLO uses only a single convolutional neural network to simultaneously predict class probabilities and bounding box locations. Of all published versions the most common ones are version 5 [**yolo5**] and 8. Version 5 widely used because of its profound documentation. Version 8 is the latest stable version [**yolo8**], released in 2023. Both use the same 80 object classes for recognition, inherited from the COCO (Common Objects in Context) dataset on which all versions are trained on.

3.4 Segmentation Method and Performance

The segmentation method used for the DAVIS dataset is rather simple. The YOLO model predicts bounding boxes for each frame, these are then used to prompt the SAM model. In case of SAM, the largest model (H) proved to be superior to the other two. The performance of the YOLO model got better with model size, but there was no significant difference in performance between versions 5 and 8.

YOLO combined with SAM performance exceeded expectations on for the majority of the videos. The model performed well on both metrics, as shown by the kernel density estimation (KDE) in Figure 2 and the mean and median values in Table 1. The difference in mean and median values between MOpOIoU and WMOpOIoU clearly shows that predicting small objects is more prone to error for larger ones. This is also supported by comparing the WMOpOIoU and MOpOIoU KDE plots

(Figure 2). In the right plot (WMOpOIoU), the density is much lower between 0.1 and 0.6 than in the left plot (MOpOIoU). This indicates better results, as the weight of small masks on the overall result was reduced. Noteworthy about the performance is that the median and mean for WMOpOIoU and MOpOIoU are quite far apart. This discrepancy is likely based on the fact that SAM performs well when prompted for objects, but sometimes does not receive any prompts. This is reinforced by a density bump at 0 in the KDEplots, indicating YOLO did in some cases not predict any bounding boxes. However this problem did only occur for individual frames. One potential solution for this issue might be to lower the confidence level of YOLO to the point where there are bounding box predictions. Another idea could be to apply the bounding boxes of previous frames to the current frame. This is based on the assumption that the location of objects in a video does not significantly change significantly between consecutive frames for almost all videos. However due limited time of this internship these approaches were not examined.

	MOpOIoU	WMOpOIoU
Mean	0.76	0.81
Median	0.86	0.9

Table 1: Mean and Median IoU

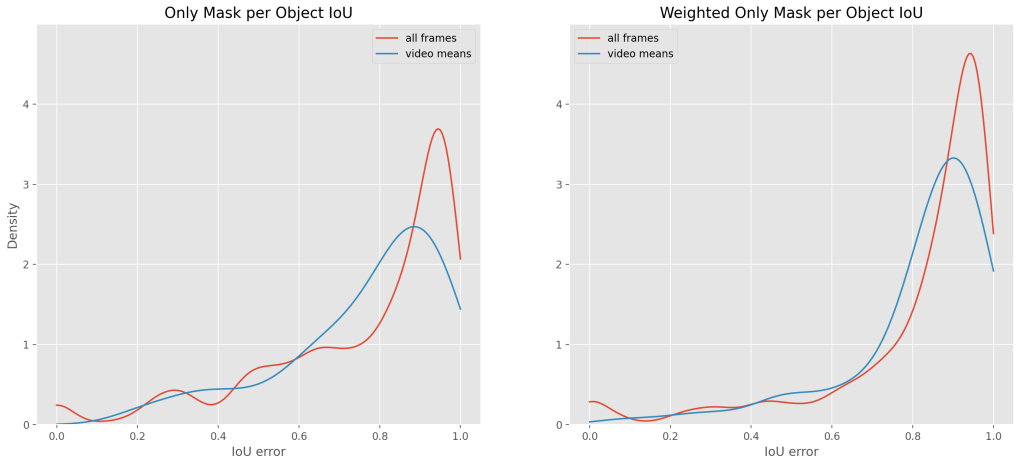

Figure 2: Kernel Density Estimation Plot of IoU at final Evaluation

Visual analysis of the masks confirms the conclusions: if predicted bounding boxes were accurate, the subsequent SAM masks were accurate as well (Figure 3). SAM worked well both for partly occluded objects and overlapping masks. As shown in the middle column of Figure 3 SAM handles both the obfuscating tree and the distinction of the white lamb, in this case, but also in general well. The main limitation of this method is the quality of object detection. As previously discussed using data collected, YOLO sometimes fails to detect smaller objects or fails to recognise objects for which it has not been trained. The observation that YOLO

struggles to detect small objects is also discussed in the original paper [**yolo1**]. This was debated and improved in the version 5 [**yolo5**] and 8 [**yolo8**] papers. However, for larger objects belonging to the COCO classes, YOLO provides reliable bounding boxes. Additionally, it generalizes well to other animal classes not part of COCO it is not trained on.

Figure 3: Annotated Mask, Image with Bounding Boxes, Segmentation Mask (top to bottom)

4 MOVI Segmentation

4.1 MOVI Dataset and the Kubric Dataset Generator

The MOVI dataset is a synthetically created by the Kubric dataset generator [**kubric**] released in 2022. The Kubric dataset generator attempts to address the prevalent problem of data availability in machine learning. The cost of good training data is high, because it often requires human input and in many cases violates the privacy and rights of the owner or creator of the data. To circumvent these difficulties, synthetic (computer generated) data can be used to train models instead of real data. The MOVI dataset is an exemplary dataset generated from Kubric. It is available in five versions, from A (easy) to E (hard). The difficulty ranges from videos with a only one single color background, a stationary camera angle and only 3-10 objects chosen from limited pool of simple symmetric shapes. The most complex setting E consists of a real world background, a randomly moving camera and up to 23 real world objects. Exemplary images from the dataset are displayed in Figure 4. All versions feature 9750 training images and 250 validation images in sizes of either 128x128 or 256x256 pixels. Aside from the raw images and the annotations, further data on depth, optical flow, surface normals and object coordinates of every pixel

are given. When the "MOVI" is referenced in this work it only refers to the version E with 128x128 pixel and only to the raw image data. Moreover, all fitting and testing was done on the much smaller validation set due to the time restraints of the internship and the large run time of SAM.

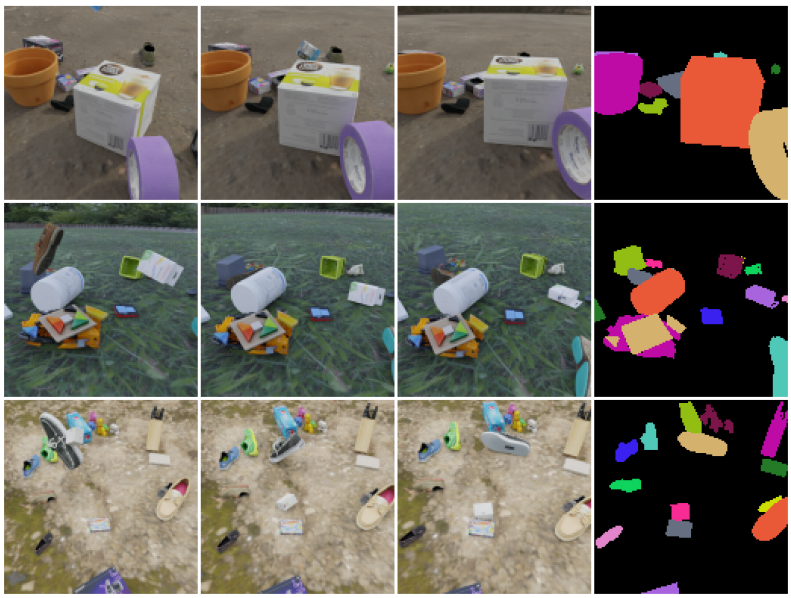

Figure 4: First, Center and Last Image and Annotated Mask for Sample Images

4.2 Review of Standard Methods

4.2.1 YOLO

Since the YOLO model showed strong performance on the DAVIS dataset and it was the first model that was tested on MOVI. However, on MOVI, the performance of YOLO was not sufficient. There were too few bounding box predictions with too low quality. This can be attributed many factors: first and foremost the differences of analysed synthetic data and the real world image domain YOLO was trained on are substantial. On MOVI, YOLO must detect very different image classes. Additionally, the image resolution is considerably than that of the DAVIS or the COCO dataset. From the displayed bounding boxes it in Figure 5 it becomes clear that predictions of YOLO could not be used to prompt SAM delivering acceptable results.

Figure 5: Yolo prediction on MOVI

4.2.2 Otsu's Method (Referenz)

Multi-Otsu-Thresholding is an algorithm that can be applied to gray scale images and aims at finding good thresholds to segment the image with. This is achieved by finding classes with a minimum intra class variance. For different images and number of threshold the results of Otsu's method varied strongly and were unstable. For some thresholds the images were over segmented with many background artifacts, while others were under segmented. As shown in Figure 6 the regions often did not correspond well to the objects.

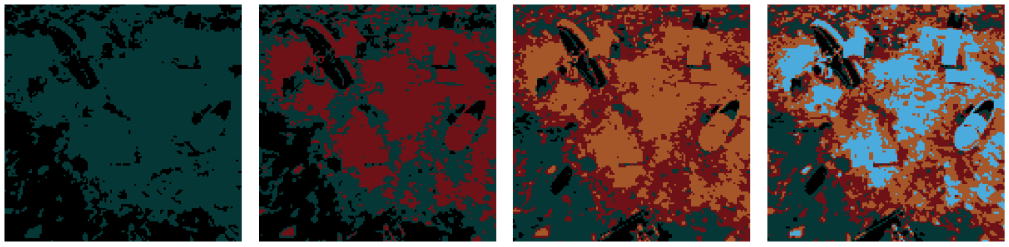

Figure 6: Otsu results with different number of classes

4.2.3 Best x-Sam Masks

For SAM one straightforward approach to generate object masks on the MOVI dataset is first to predict all masks for the entire image and then select the first few masks. This works, because it appears that SAM sorts the masks based on an internal metric, which often aligns closely with the object masks. SAM does output certain metrics, such as "area", "stability score" or "predicted IoU", but the list is not sorted according to these parameters. If the list is sorted by these parameters, the performance drops sharply. This method achieves the best IoU of approximately 0.5 at 31 masks. The quality of the masks is displayed in Figure 7 (same images as in Figure 5). These examples demonstrate that SAM has a solid understanding of the image and is able to produce good results prior without prompting. However, a

weakness of the model can be identified in Figure 7. Often the segmentation masks of SAM, when prompted with points, are quite fine-grained. The box in the left image for example is segmented as six different masks, 4 of whom correspond to color patches printed on the box.

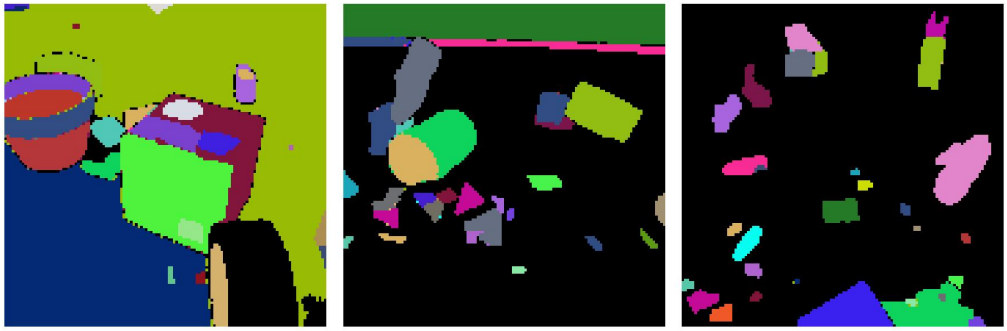

Figure 7: IoU of testset values during parameter optimization

4.2.4 Optical Flow

Optical Flow is a method of tracking the motion of image parts between video frames. There are two main types: Dense and Sparse Optical Flow. In Sparse Optical Flow the flow of given points is tracked. These points are usually corners of objects. In contrast Dense Optical flow targets to track the motion of all pixels between frames. OpenCV implementations were used for both types. Optical Flow was a promising method to evaluate, because there are many moving objects in the target videos. The expectation, was that these should be tractable by Optical Flow. However, both types of Optical Flow exposed certain weaknesses in their application in combination with SAM.

Dense Optical Flow resulted in very blurred images, which made distinction between different objects hard. The problem can be observed in Figure 8. It displays the Dense Flow masks produced by images in the left column of Figure 4. Even with knowledge of image and ground truth, objects are hardly distinctable. This is due to multiple root causes: a low image resolution, a moving camera and objects often with similar trajectories. The combination of these factors made the distinction between objects difficult to the point where even for humans with ground truth knowledge objects were not identifiable.

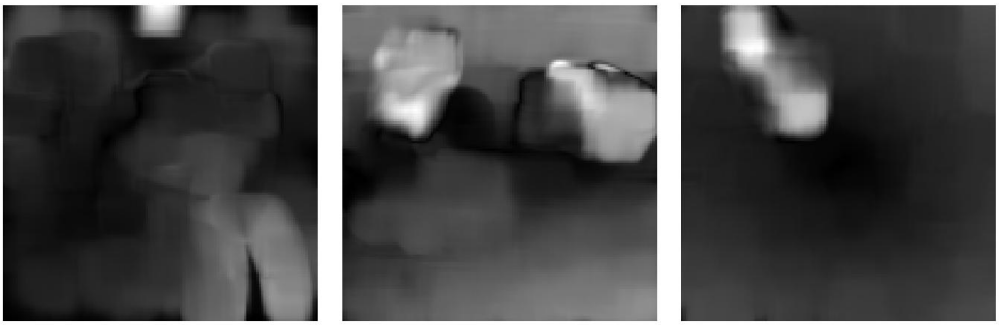

Figure 8: Sample Images of Dense Flow

OpenCV employs the Shi-Tomasi corner detection [**shi˙tomasi**] algorithm to identify points that can then be tracked by Sparse Optical Flow. After optimizing the Shi-Tomasi hyperparamters by hand, points corresponded fairly well to objects. However, these point predictions still exposed weaknesses. Since the Shi-Tomasi algorithm detects and tracks corners, the points often are not located on at the center of an object. As seen in Figure 9 the minority of points is located at the center of objects. Another problem was, that some objects were not tracked. Furthermore, on some backgrounds most tracking points were located in the background (Figure 9 left image).
The tracking of selected points between frames worked well, but it remained unclear how to utilize this effectively. This was due to the described characteristics of the tracked points. Furthermore, since movement of objects is often localized and objects tumble against each other, it was hard to identify which group of points corresponds to which object.

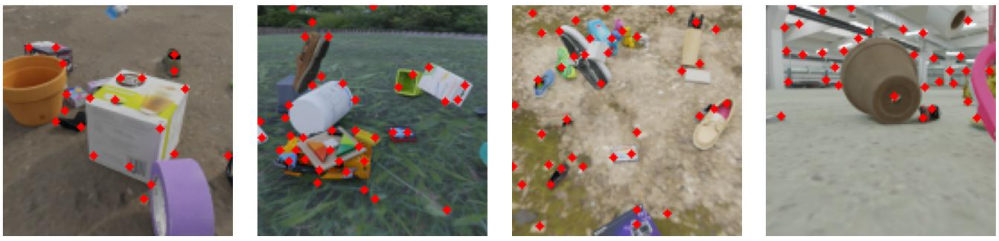

Figure 9: Sample Images of Shi-Tomasi

4.3 Own Approach

Since all of the tested standard methods failed to delivered sufficient results for promoting SAM, a combination of different methods to an ensemble, seemed to be expedient. In the following,it will be described, why and how the methods were employed.

4.3.1 Step 1: Sobel Filters and Connected Components

The method, that delivered best results was Sobel filters. They outlined all objects in the image well. Sobel filters calculate color gradients for every image pixel. The resulting gradient can not be used to prompt SAM (Figure 10 center left column). To translate the identified edges into points, the gradient values where significantly thresholded. The goal was to identify fully separated areas (Figure 10 center right column). In a second step connected components were used to receive individual regions (Figure 10 right column). The resulting regions had a couple of promising properties: Most importantly that objects in comparison to previous discussed

methods were almost always detected. Additionally, there were regions situated in closer proximity to the center of the object. This was identified as a clear advantage compared to the use of Shi-Tomasi. A remaining issue was that there were numerous predictions located on the background, a problem that also existed also in any previous method.

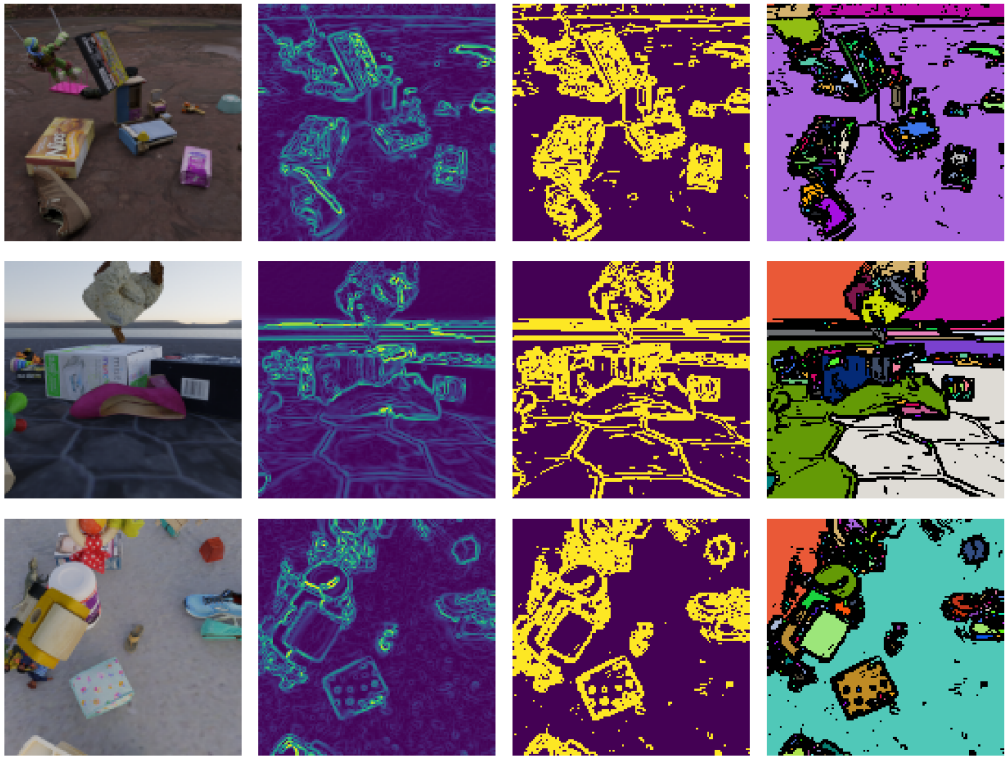

Figure 10: Sobel filter Segmentation approach

4.3.2 Step 2: Region Fusing and SAM Point Predictions

As displayed in Figure 11 (upper row) there are large numbers of markers across the image. Each of these markers represents the center of a region created in the first step. The goal of the second step was to reduce the number of masks. The primary reason this was done, is to decrease the computations required when promping SAM. To reduce the number of regions, regions with a large similarity were fused together. This similarity was calculated based on the distance between the centroids of the regions, the average color and depth. Depth values were created using the Depth-Anything model [**depth˙any**]. The similarity was calculated by modeling an exponential distribution ($\lambda e^{-\lambda x}$) for every parameter (distance, color, depth). The mean of the three function values was then used as a measure of similarity between individual regions. The lambda values for exponential distribution of the three parameters (lambda color, lambda depth, lambda distance) should be optimizable to select a combination of parameters, that accurately fuses masks that have a

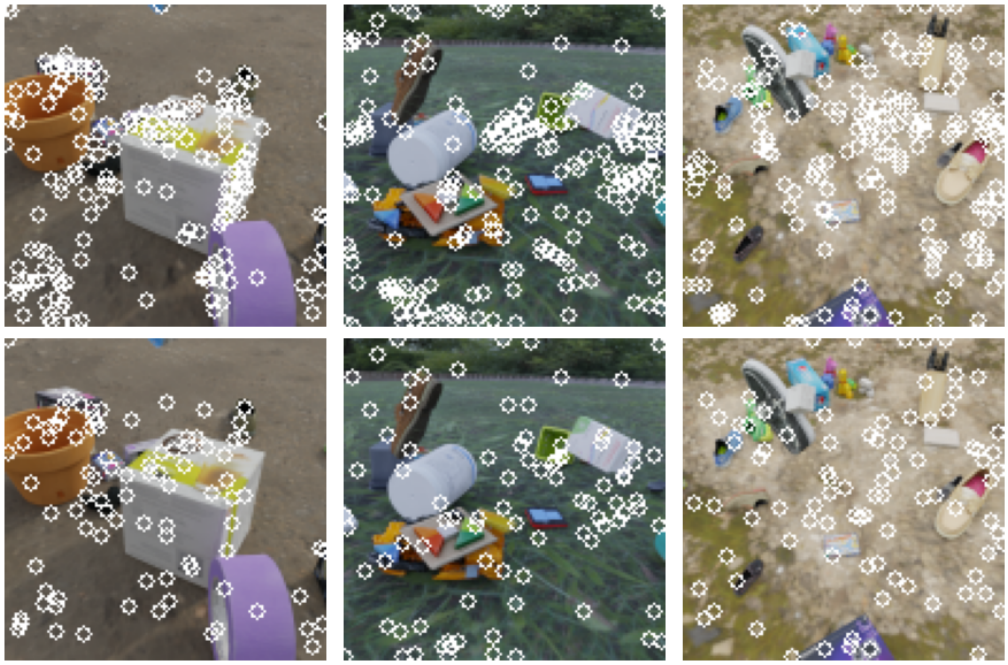

Figure 11: Fusing of points

high likelihood of belonging to the same object. If similarity between values in one parameters is important then, the model can increase the weight of that parameter (by increasing its lambda value). This measure of similarity is then calculated for all region pairs. Then the pair with the highest similarity are fused to one regions. This process is repeated for as long as the number of remaining regions is larger than the "maximun number of masks". The parameter "maximum number of masks" was another parameter that was introduced into the model. It can also be fitted and determines how long masks are merged. It can also remove the entire process of fusing masks, if the merging of masks based on the method described above is not effective.

Once the number of masks was reduced, SAM was prompted with the centroid of every remaining region. Since there was no way to detect which points belong to the same object, SAM had to be prompted with every point individually. As a result of this iterative prompting the resulting masks overlap with each other and there are often multiple predictions for the same object.

4.3.3 Step 3: Create Bounding Boxes and Prediction of Final Masks

SAM can only predict individual objects when prompted with points. In this step the objects masks were transformed into bounding boxes. Based on these bounding boxes the model was prompted again. When transforming masks, boxes are drawn only the around the largest connected component. Otherwise, bounding boxes were

getting too large. Before prompting the model with these boxes, one remaining challenge was that many of the background masks nearly covered the entire image. To remove these masks another parameter controls the maximum mask size. All masks greater that this parameter are discarded. Then the model is prompted with the remaining masks (prompts are displayed in Figure 14, 15, 16) .

4.3.4 Step 4: Background Reduction

The predicted masks still contained background masks. In step 4 the attempt was made to eliminate these. Differentiating object masks from false background predictions was difficult. They were not differentiable further by size or color, because both of these properties did not reliably identify objects. Selecting incorrect masks was done using depth values. It was observable that most masks of actual objects seemed to share a similar depth. Background masks often were comprised of much smaller or much larger depth values. This observation led to the idea of modelling a normal distribution around the mean depth of all predicted masks and then to introduce another parameters ("depth variation") that sorts out masks where the depth difference was too large.

4.3.5 Parameter Fitting

The parameters in this model could not be optimized by using gradient descent like most other models. The only way to identify a nearly optimal combination of parameter was to use a parameter study. In every iteration one of the six parameters (lambda color, lambda depth, lambda distance, max number of masks, bounding box max size and depth variation) was moved closer to its optimal value while all the other parameters remained the same. To determine what the correct change of value was, the model evaluated the same randomly picked subset of the dataset with three different values for the parameter: the parameter was changed positively, negatively and was kept constant. This procedure was repeated until the parameter stopped changing. The parameters were fittet in two consecutive runs with differing parameters displayed in Table 2. One round corresponds to one optimization iteration for one of the six parameters. The "samples per iteration" are the number of images that were being analysed, before changing the models parameters by the "parameter change rate". "Sample increase" and "decrease of change rate" are the parameters by which the "parameter change rate" and "samples per iteration" are changed every six iterations (when all parameters are adjusted once).

	rounds	samples per iteration	sample increase	parameter change rate	decrease of change rate
run 1:	98	5	5	0.1	0.1
run 2:	91	30	5	0.05	0.1

Table 2: Parameters of optimization runs

Parameters of the first optimization round were roughly approximated values, based on some rudimentary test runs. In the second iteration the "samples per iteration" were increased and the parameter change rate" was decreased to achieve more stable results. This allowed to find a a model that produced better results.

4.4 Results

The fitting of the parameters increased IoU performance of the model by 0.08 compared to starting values. It was outperforming the "best x-SAM masks" with fitted parameters by 0.1. Fitting the parameters was difficult, because the variance of the results were very high. Even over many samples results were unstable, making it hard to find a good set of parameters. The parameter updates seemed to contain randomness even when many samples were used. Parameter changes often had a limited or no effect on many images and a huge effect of very few others. Thus, the updates are often based on single or few images even. In particular, the evaluation of the validation set was time consuming, because it encompasses 1250 frames. On the provided hardware, evaluating the entire dataset took around 45 minutes per run (and more than 10 hours in total). Thus, validation values were only calculated every second cycle. In Figure 12 one cycle corresponds to six rounds. The grey line in Figure 12 and 13 shows where the first optimization run stopped and the second started. Some consecutive runs displayed an IoU difference of up to 30%. The high variance is also observable in Figure 13, where even though the graph shows an moving mean the values fluctuate significantly. A better approach would have been to fit the parameters with more images per update and to reduce the variance as much as possible. Unfortunately this was not possible, due to the run time of SAM and the limitations of this internship. Another indication of high variance is that validation values displayed in Figure 12 are consistently above the validation values displayed in Figure 13. A likely explanation for this is that the validation set features more images that are easier to solve and have good IoU scores.
All in all the different the parameters introduced to the model seemed to have a positive effect on the results: The scores on the training and validation set rose fairly consistent. With a better fitting routine and bigger batches the model might even have performed better. This is to be assumed because the model did not seem to show any signs of overfitting. Furthermore, the validation and training values were still slowly rising. Overfitting will likely never be any problem in this model, because

of the small number of heuristics and parameters. This is because the amount of data is large compared to the number of parameters.

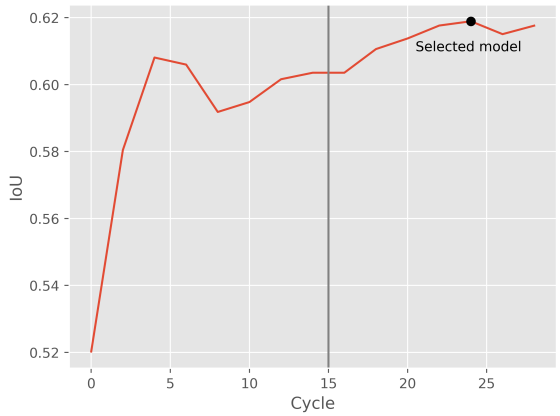

Figure 12: IoU on the validation set

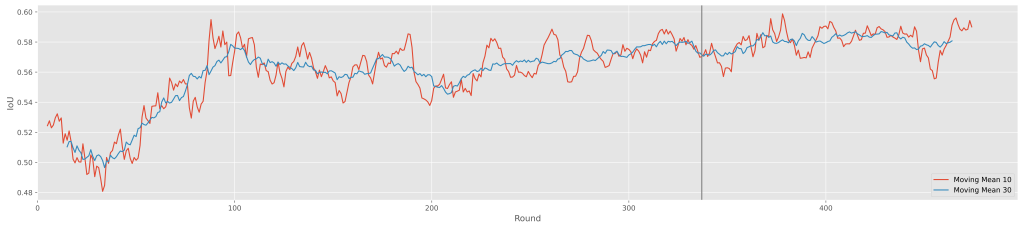

Figure 13: Exponential moving average of IoU during training

The individual results shown in Table 3 reveal which methods worked. The depth threshold did consistently decrease for the first 14 cycles, but then remained constant for the second optimization run. It still removed about 5 masks per image. Even though this parameter is not able to identify all background masks, but it manages to at least alleviate the problem. The maximum number of masks and the bounding box size where the primary reason of the sharp ascent as the start of the parameter optimization. The very high maximum number of masks indicates that the mask fusing process does only work to a certain extent. The initial number of masks is much higher than 90. This indicates that the process seems to be able to achieve some reduction without losing relevant information. However, as shown in Figure 11 there are also many points which could be fused. Two possible reasons were identified: possibly multiple point can be dealt with more effectively or the heuristics implemented did not work well enough. It might even be desirable to have multiple point for the same object, because SAM predictions for some points are incomplete. This would mean that multiple predictions per object could even be advantageous. The three point fusing metricises indicate that color is most important and distance

is more important than depth. Depth might be a redundant criteria to consider when fusing the regions. This could be due to imprecise prediction by the depth model.

	Lambda Color	Lambda Depth	Lambda distance	max number of masks	bounding box max size	depth threshold
Starting Values:	1	1	1	30	5461	0.1
After Run 1:	1.28	0.73	1.11	90	3609	0.02
After Run 2:	1.42	0.52	1	88	3709	0.01

Table 3: Results of optimization runs

In Figure 14 , 15 and 16 the bounding box predictions and the final results are displayed. Figure 15 shows representative results that reflect the majority of masks. The other two figures depict results that are on the positive and negative end of the spectrum. As described earlier, a weakness of SAM tends to output multiple masks for one object. This is prevalent in Figure 14. In Figure 15 is the background reduction on display. In the image on the left it becomes clear that the mask created by the bounding box was not removed. In all the other images this mask was erased, likely because of its depth values being to different from object masks.

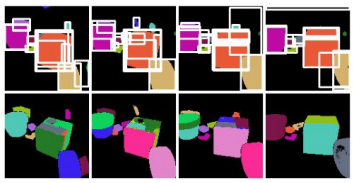

Figure 14

Figure 15

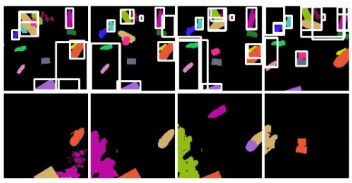

Figure 16

5 Conclusion

The goal of this research was to find automated prompts for the Segment Anything Model that create good segmentation masks. SAM generally displays good performance when prompted by humans. In case of the two examined datasets, SAM manages to transfer segmentation problems into object detection problems. In cases in which good object detection models are available, satisfying performance can be achieved with minimal effort by prompting SAM with the bounding boxes. A good example for this is the performance of the combination of SAM and YOLO on the DAVIS dataset.

However, in cases where no object detection model is available to deliver sufficient object bounding boxes, SAM performance drops significantly. This is the case for MOVI. For MOVI no straight forward way to generate bounding boxes of high quality is present.

As discussed in the results the developed methods to prompt SAM resulted in some promising results, even though some fundamental problems remained unsolved. A

variety of standard methods of computer vision were tested on MOVI. None of those methods delivered satisfying results on their own. An ensemble of methods that tried to minimize weaknesses of individual methods resulted in the best results. The ensemble outperformed the best individual method by 0.1 IoU. One finding is that SAM better for multiple objects with bounding boxes than point predictions. To circumvent this weakness the described method of transferring masks into bounding boxes can be used. Furthermore, the sorting of background masks by their depth values works well. In retrospective there are numerous options to increase performance the model further, for example initializing it with Shi-Tomasi or trying to use additional background reduction heuristics. However, to achieve similar performance on MOVI as on DAVIS, some deep learning object detection model would be required.