HEIDELBERG UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Evaluating Single Object Tracking for Autonomous Driving Datasets

Project Report

Presented for the module Fortgeschrittenenpraktikum

Presented by

Samuel Kiegeland

1 Abstract

English - In this project, I applied the Single Object Tracking and Segmentation method SiamMask by Wang et al. (2019) to track individual vehicles in the autonomous driving datasets A2D2 (Geyer et al., 2020) and KITTI (Geiger et al., 2012). Given an initial frame of a video sequence, I selected target objects by enclosing them with a bounding box and tracked them until the end of the sequence. The goal was to test, whether the method SiamMask is applicable to the task of tracking individual vehicles, without explicitly fine-tuning it on this task. Furthermore, I analysed, in which situations the tracking fails and how suitable the datasets are for this task.

Deutsch - In diesem Projekt habe ich das Trackingsystem SiamMask von Wang et al. (2019) verwendet, um Fahrzeuge in den Datensätzen A2D2 (Geyer et al., 2020) und KITTI (Geiger et al., 2012) zu tracken. Dabei wird das Zielobjekt im Anfangsbild einer Sequenz mit einem Begrenzungsrahmen markiert und über die gesamte Sequenz hin getrackt. Das Ziel des Projektes war es, zu testen, ob das System SiamMask für Tracken von Fahrzeugen geeignet ist, ohne es explizit für diese Aufgabe zu trainieren. Weiterhin habe ich analysiert, in welchen Situationen das Tracken fehlschlägt und wie geeignet die Datensätze für diese Aufgabe sind.

Contents

List of Figures

List of Tables

2 Introduction

In recent years, there have been many applications of Multi-Object Trackers to autonomous driving, such as Ess et al. (2010) and Zhao et al. (2018). These methods are important, in assisting a vehicle to build an accurate model of its environment. However, Multi-Object Tracking usually requires to first detect an object before tracking it. The detection algorithm itself is depended on the object classes, it has been trained on.
The related task of Single Object Tracking only requires to specify the bounding box which encloses an object in the first frame of a sequence, which makes this approach much more flexible.
In this project, I evaluated Single Object Tracking by applying the method SiamMask by Wang et al. (2019) on two autonomous driving datasets, namely Audi Autonomous Driving Dataset by Geyer et al. (2020) and KITTI (Geiger et al. (2012)).
In order to do this, I relied entirely on the data from the front camera and excluded additional information sources such as LIDAR. The goal was to test, if this method is effective at tracking vehicles without explicitly fine-tuning the model for this type of problem. Moreover, I found weaknesses within the datasets and explored various methods, which could help mitigate these.
In section 3, I give a quick overview of recent progress in Single Object Tracking and the development of Siamese neural networks before I describe the architecture of SiamMask. Section 4 contains a description and comparison of the two datasets, I used in this project. In section 5, I state the evaluation procedure and present the results. At last I describe future additions section 6 and conclude the project in section 7.

3 Method

Single object tracking, describes the problem of tracking an individual object through multiple frames given its initial position in a starting frame which is defined by a bounding box enclosing the object. Over the last years, the most commonly used approaches have developed from Correlation filters (Bolme et al., 2010) to learning a similarity metric to estimate the position of the object in the next frame (Bertinetto et al., 2016), (Held et al., 2016).
Bertinetto et al. (2016) have introduced fully-convolutional Siamese Networks, that learn a similarity function offline. This yields the advantage, that this function can be applied efficiently during testing.
A fully-convolutional neural network (FCN), as developed by Long et al. (2015), is a modification of convolutional neural networks in which every operation (convolution, pooling, activation function) is translation invariant. As a result, the network only depends on relative spatial information, which enables FCNs to deal with images of arbitrary size. This, alongside with performance improvements, is the major advantage of FCNs over non-fully-convolutional neural networks. A FCN can be obtained from a convolutional neural network by converting fully connected layers to convolution layers.
In Bertinetto et al. (2016), a Siamese network (originally developed by Bromley et al. (1993)) learns a similarity estimate f, which can be represented by the following equation:

$$f(x,z) = g(\varphi(x), \varphi(z)) \tag{1}$$

The whole network consists of two identical neural networks which, respectively, take the input images x and z and apply an embedding function ϕ. The resulting representations are then combined using a function g which can be seen as similarity measure. As depicted in Fig. 1, the bounding box of the previous image can be denoted as the template frame while the current image is the detection frame. The FCN then compares the template to different sub-regions of various sizes in the detection frame in order to find the most similar region.

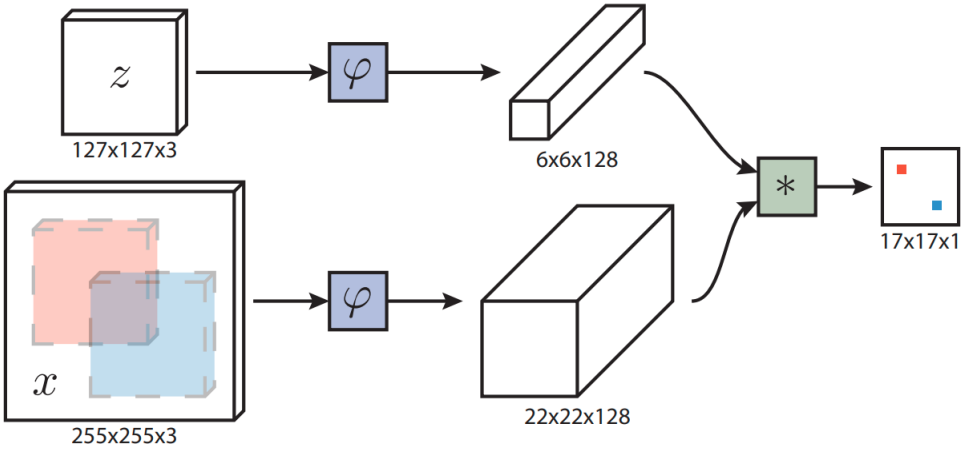

Figure 1: SiamFC, *Source:* Bertinetto et al. (2016)

This system has been further improved by Li et al. (2018). They add a region proposal network (RPN) as proposed by Ren et al. (2015), in order to regress a bounding box to enclose the target object. Their system takes the outputs $\varphi(x)$ and $\varphi(z)$ from the siamese network and passes them to a region proposal sub-network, where each output is split into a classification branch and regression branch. The correlations for each branch are computed by using the convolution $*$:

$$\begin{aligned} A^{cls}_{w\times h\times 2k} &= [\varphi(x)]_{cls} * [\varphi(z)]_{cls} \\ A^{reg}_{w\times h\times 4k} &= [\varphi(x)]_{reg} * [\varphi(z)]_{reg} \end{aligned} \tag{2}$$

The network is then updated by summing the losses for each branch, where L_{reg} is the smooth L_1 loss and L_{cls} is the cross-entropy loss.

The method SiamMask, by Wang et al. (2019), which I used in this project, builds on top these approaches by adding a binary segmentation mask. Given the similarity score $f(x,z)$ from Eqn. 1, they implemented a two layer neural network to learn a segmentation mask, which for each pixel, determines whether the pixel belongs to the target object. In order to do that, they extract binary labels during training for each candidate window and use a binary logistic regression loss L_{mask}, over all windows, to learn the segmentation mask. The total loss is obtained by summing and weighting the three different loss functions:

$$L_{total} = \lambda_1 L_{mask} + \lambda_2 L_{cls} + \lambda_3 L_{reg} \tag{3}$$

where $\lambda_{1,2,3}$ are hyperparameters, with $\lambda_1 = 32$ and $\lambda_2 = \lambda_3 = 1$. In the architecture depicted in Fig. 2, box corresponds to the regressions loss and score to the classification loss.

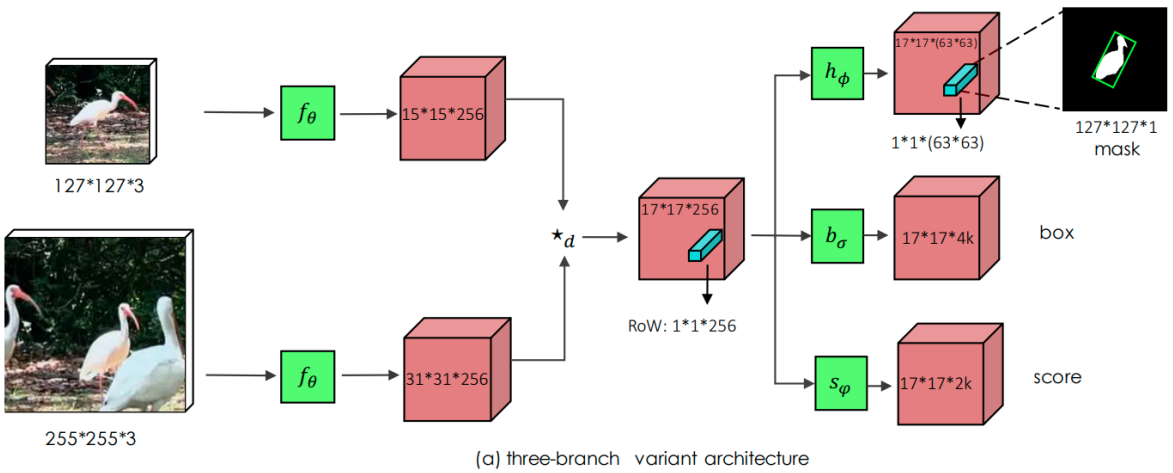

Figure 2: SiamMask, *Source:* Wang et al. (2019)

The SiamMask-model, which I used in this project, has been pretrained on the datasets Microsoft COCO (Lin et al., 2014), ImageNet-VID (Russakovsky et al., 2015) and YouTube-VOS (Xu et al., 2018).

4 Datasets

4.1 A2D2

The first dataset is the Audi Autonomous Driving Dataset (A2D2), which is presented by Geyer et al. (2020). It consists of 41,277 frames labeled with point clouds, semantic segmentation and image segmentation. A subset of 12,497 frames is also annotated with 3D bounding boxes. All of the segmentation images and bounding boxes are created by human annotators. Additionally it contains 392,556 unlabeled sequential frames.
The dataset compromises a variety of scenes like urban, highway, and country sides which is beneficial for a comprehensive analysis. What distinguishes A2D2 from other autonomous driving datasets is the availability of additional sensory information, like GPS, IMU, steering angle, odometry, velocity, etc. and the use of the CC BY-ND 4.0 licence, which makes it accessible for commercial use.

4.1.1 Annotations

The subset labeled with semantic segmentation contains 31,448 images. However, these labels are not very useful for evaluating Single Object Tracking, as there is no way to distinguish between two instances of the same class. Therefore, the labels, which I used for the evaluation, are the instance level segmentation labels. Compared to the semantic segmentation labels, these labels provide a different mask for each instance of a class. There are 26,314 instance segmentation labels, corresponding to the images in the semantic segmentation subset. In order to filter out the images, I preprocessed the camera images from the semantic segmentation dataset to match the instance segmentation labels. Fig. 3 shows an example of a camera image and the corresponding semantic segmentation and instance segmentation. The instance segmentation labels are encoded with an uint16 grey scale, which makes the labels poorly visible. For that reason, I increased the brightness in the example. These labels are annotated for all

Camera image | Instance segmentation | Semantic segmentation

Figure 3: A2D2 camera image and labels

instances belonging to vehicle classes. The different vehicle classes are "Car", "Pedestrian", "Truck", "SmallVehicle", "UtilityVehicle" and "Bicycle".

4.1.2 Missing frames

The subset of 26,314 labeled images, which I use in this project, is created from the 392,556 unlabeled sequential frames, by picking out individual frames and labeling them. As a consequence, there are many instances, where the scene, which is displayed in a frame, differs substantially between two successive frames. This can be seen in Fig. 4. On many occasions, objects, which are present in the first frame, have disappeared in the next frame or moved to a large degree. I expected this to increase the difficulty of tracking individual objects.

Figure 4: Consecutive frames in A2D2

4.2 KITTI

To compare the performance of SiamMask, I selected KITTI by Geiger et al. (2012) as second dataset. In contrast to the A2D2, this dataset is published under Creative Commons Attribution NonCommercial ShareAlike 3.0 license, which prohibits its commercial applications. The data used in this project comes from the Multi-Object and Segmentation (MOTS) benchmark (Voigtlaender et al. (2019)) and is based on the benchmark for Multi-Object Tracking by Milan et al. (2016). In total this dataset consists of 10,870 frames and their corresponding annotations. As with the A2D2, I only use the instance segmentation labels to evaluate the tracker. The labels are designed in a similar fashion, as the instance segmentation labels of the A2D2. However, there only two different classes, namely "Car" and "Pedestrian". In contrast to the A2D2, there are no missing between successive frames as can be seen in Fig. 5. This results in small differences between successive frames compared to the A2D2, which makes this dataset is suitable to compare the effectiveness of the SiamMask tracker.

Figure 5: Consecutive frames in KITTI

5 Experiments

I implemented the evaluation by interfacing with the official repository of SiamMask [1]. The code and configurations, I used to conduct the experiments can be found on Github [2], alongside with the preprocessing of the datasets.

I first divided the evaluation into individual tracking scenarios. A scenario starts by choosing an instance from the instance segmentation labels at a given frame, enclosing it with the corresponding bound box and calculating the mask, which defines the coordinates of the target object. The mask is then passed onto the tracker, which estimates the position of the object in the next frame. From that estimation, the tracker segments the target object from its background and calculates the bounding box which encloses the segmented area. In the following, I denote the bounding box, that the model predicted for a given frame as the current tracking window.

The first problem, I encountered, when evaluating SiamMask on the A2D2 and KITTI dataset, is to detect, when a tracking scenario is over. There numerous cases, where the tracker fails, by selecting the wrong instance of an object for the next frame. On top of that, if the target object disappears from the frame, the tracker automatically switches to another object and resumes to track it. This problem is illustrated in Fig. 6. The tracking window contains the target object in the first to frames, but switches to another object, after the target has disappeared in the third frame. Leaving the model to continue tracking the next object, drastically reduces the score, when automatically evaluating the model by calculating the IoU. On the other hand, selecting the configuration which maximizes the IoU favors tracker, which predict the end of

[1] https://github.com/foolwood/SiamMask
[2] https://github.com/samukie/SingleObjectTracking

Figure 6: Tracker switches to other object

a tracking scenario earlier. For that reason, I split the experiments into two subsections: In the first subsection, I describe experiments, which aimed to find stopping criteria, to detect the end of a tracking scenario. In order to do this, I first calculated the end of a track using the instance segmentation labels and tried to approximate this result without the use of labels, by testing various stopping criteria. In the second subsection, I selected the best configuration and reran the experiments to calculate the IoU of the SiamMask tracker.

5.1 End of track detection

There are two situations in which a tracking scenario is over:

1. The target object disappeared from the frame
2. The tracker failed and the current tracking window does not contain any pixels, which belong to the target object

This simplification does not consider the situation when an object first disappears and later reappears in the scene. However, the goal for these experiments is just to determine, whether it is possible to correctly identify the end of a tracking scenario.
In order to evaluate the complete dataset, it would be necessary iterate over every scene, take every single frame in the that scene as a starting point, identify all occurring objects, and track them until they disappear. However, this comes at large computational costs, which is why I approximate the evaluation by sampling five entry-points at random for each scene. There are 23 scenes for the A2D2 and 21 for the KITTI datasets, which results in 115 and 105 entry points. For each entry-point, I identify all occurring objects using the instance segmentation labels and track all of the objects for at most 50 frames. I repeat each experiment with two additional random seeds and report the mean and standard deviation. Furthermore, I exclude all scenarios, where the target object does not disappear at some point. Including these scenarios would result in a configuration, which is biased towards rarely predicting the end of a tracking scenario.

For this section, I defined that an object is not present in a frame or in a tracking window, if none of its pixels are. Furthermore, I treated this problem as a binary classification task, where the class "positive" means, the tracking scenario is over and "negative" denotes that the scenario continues. I decided for this definition, as the goal is to measure how good the system is at detecting the end of a tracking scenario, i.e., the number of "true positives". The evaluation metrics, I used for this, are Precision and Recall:

$$\text{Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{4}$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{5}$$

and the F-measure, which can be seen as a harmonic mean of precision and recall:

$$\text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{6}$$

As previously stated, a true positive (tp) denotes the case, where the model correctly detects the end of a tracking scenario. A false positive (fp) indicates that the system falsely predicted, that the target object is not present in the frame anymore. A false negative (fn) means, the system missed the end of a tracking scenario. I distinguish between two different evaluation settings:

1. In this setting I used the instances segmentation labels to determine the frame in which tracking window does not include the target objects anymore. This serves as an upper bound to the second evaluation setting.

2. In this setting I tried to detect the end without using labels. In order to do that, I tested a variety of similarity metrics to detect when the target object disappears.

5.1.1 Instance segmentation labels

Using instance segmentation labels, the mean Precision, Recall and F-measure are depicted in Tab. 1. The mean recall for both datasets is at 1. This makes sense, as every time the

Dataset	Precision	Recall	F-measure
A2D2	0.346 $\pm$0.014	1 $\pm$0	0.515 $\pm$0.016
KITTI	0.470 $\pm$0.024	1 $\pm$0	0.639 $\pm$0.022

Table 1: End of track detection with instances segmentation labels

target object really disappears from the whole frame, it also has to disappear from the tracking

window. Therefore, the amount of false negatives has to be 0, as I'm using the instance segmentation labels to check, whether the target object has disappeared.
Consequently, the amount of false positives denotes precisely the amount of times, where the SiamMask Tracker fails by selecting and area, which does not include pixels belonging to the target object even though it is still in the frame.
As can be seen in Tab. 1, the mean Precision and F-measure are significantly higher, when evaluating the SiamMask tracker on the KITTI dataset. A possible explanation for this are the gaps between successive frames. This problem can also be observed, when applying the tracker to the examples in Fig. 4 and Fig. 5 from the previous section. The results in Fig. 7 show that the tracker correctly predicts the position of the vehicle in the second frame, where the vehicle is at a similar relative position. In the third frame however, the tracker fails, as

Figure 7: A2D2 tracking scenario

the target vehicle is in a different region, i.e. the middle of the frame and much further away than before. Contrasting, the tracker reliably tracks the vehicle in Fig. 8, where the difference between two consecutive frames is very small.

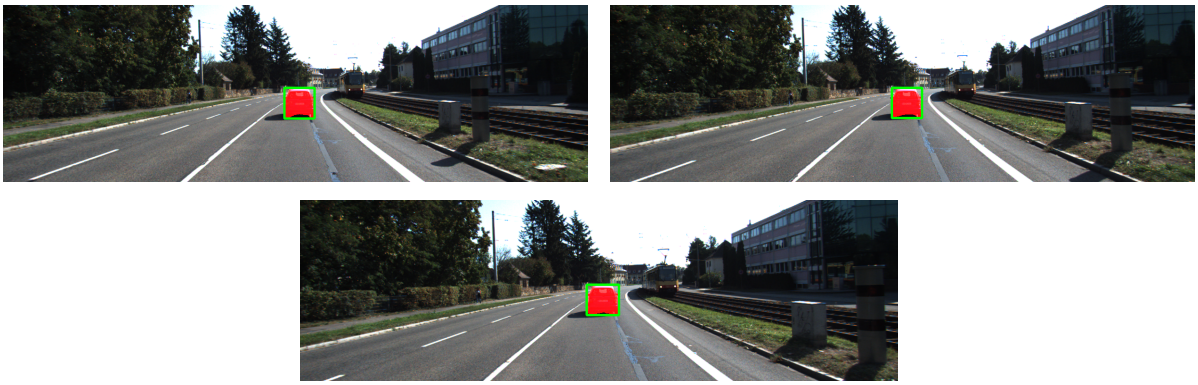

Figure 8: KITTI tracking scenario

The number of frames in a tracking scenario also differs between the datasets. There are on average 6.452 frames for the A2D2 and 9.427 for the KITTI dataset. In the following, I

describe attempts to approximate the scores in Tab. 1 without using the instance segmentation labels.

5.1.2 Similarity measures

The intuition for these experiments is that two tracking windows are more similar, if the target object is present in both and less similar, if it disappeared. I extracted the tracking window of each frame and computed a similarity score for each pair of successive frames. The point, where score falls beneath a certain threshold indicates that the tracking window does not contain the target anymore.

SiamMask confidence score

My first idea was to extract the maximal similarity score from Eqn. 1, which is used to find the most similar tracking window and use it as a confidence measure. A lower score indicates that none of the possible tracking windows are particularly similar to the previous window, which could be a sign, that the target object is not present anymore. I examined the effects of ending a tracking scenario, after this score falls under a certain threshold. Fig. 9 and 10 shows the mean and standard deviation of Precision, Recall and F-measure for various thresholds. To compare the results, I also investigated external measures, which do not have access to the

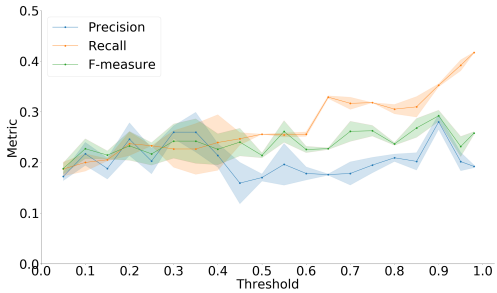

Figure 9: A2D2 confidence score thresholds

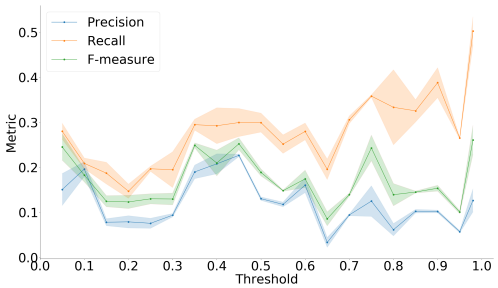

Figure 10: KITTI confidence score thresholds

internal parameters of SiamMask.

Structural similarity index measure (SSIM)

This metric has been introduced by Zhou Wang et al. (2004). While it was originally designed to measure the loss in quality of an image due to compression, it can also applied to similarity classification. I extracted the tracking window of each frame and computed the SSIM of

pairwise to consecutive windows x and y as follows:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{7}$$

with

1. μ_x, μ_y the mean of x, y σ_x^2andσ_y^2 the variance of x, y
2. σ_{xy} the covariance of x and y
3. $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator
4. L the dynamic range of the pixel-values (typically this is $2^{\text{bits per pixel}-1}$)
5. $k_1 = 0.01$ and $k_2 = 0.03$.

Similar to the confidence score in computed Precision, Recall and F-measure for using various thresholds to determine the end of track (see Fig. 11 and 12) However, this measure only captures structural, pixel-wise differences and no semantic information.

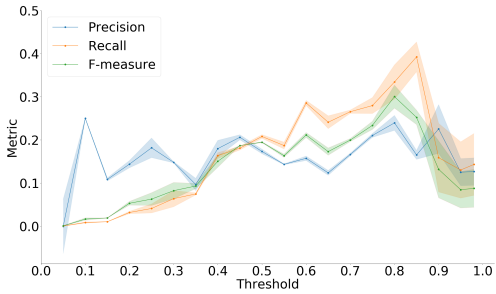

Figure 11: A2D2 SSIM thresholds

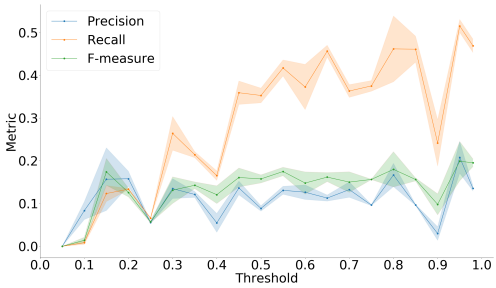

Figure 12: KITTI SSIM thresholds

Autoencoder

To capture semantic information, I included a ResNet50 autoencoder as third similarity measure. ResNet50 is a 50-layer convolutional neural network which has been pretrained on the ImageNet database (Deng et al. (2009)) to classify objects into 1000 categories. I applied the network to encode the extracted tracking windows in order to obtain a vectorized representation of length n. I compared the representations A and B of two tracking windows by computing the cosine similarity:

$$cos(A,B) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i} \times \sqrt{\sum_{i=1}^{n} B_i}} \tag{8}$$

Fig. 13 and 14 show the scores for varying thresholds.

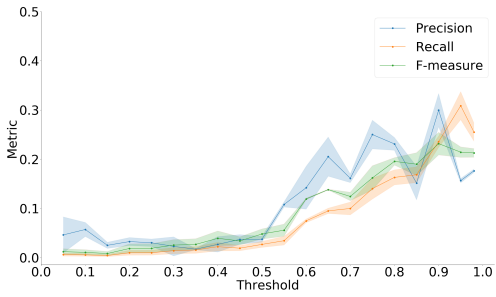

Figure 13: A2D2 Autoencoder thresholds

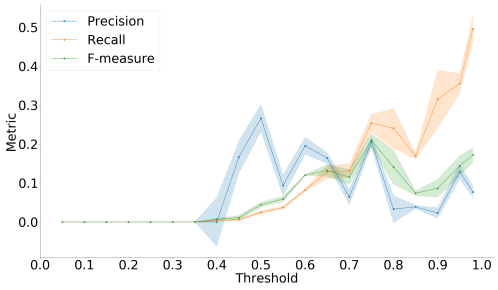

Figure 14: KITTI Autoencoder thresholds

The results for all three methods suggest, that overall, it is possible to find reasonable stopping criteria by trying various similarity metrics. However, the scores vary strongly between different thresholds and random seeds which makes it impossible to conclude that one specific similarity measure generally outperforms the others. The best method for each threshold values for each measure are depicted in Tab. 2 and 3. Based on these results I choose the SSIM for the A2D2 and the confidence score for the KITTI dataset to detect the end of track while calulating the IoU.

Method	Precision	Recall	F-measure	Threshold
Confidence score	0.280 ±0.030	0.352 ±0.006	0.292 ±0.139	0.9
SSIM	0.204 ±0.035	0.334 ±0.079	**0.300 ±0.050**	0.8
Autoencoder	0.300 ±0.067	0.236 ±0.023	0.232 ±0.046	0.9

Table 2: A2D2 best similarity measure

Method	Precision	Recall	F-measure	Threshold
Confidence score	0.227 ±0.006	0.299 ±0.062	**0.252 ±0.026**	0.45
SSIM	0.208 ±0.074	0.515 ±0.030	0.199 ±0.092	0.95
Autoencoder	0.206 ±0.027	0.254 ±0.044	0.21 ±0.034	0.75

Table 3: KITTI best similarity measure

5.2 IoU

As stated in section 3, SiamMask outputs a bounding box and a mask for every step. The predicted mask determines, which pixels belong to the target object. To compute the overlap

between the set of pixels in the predicted mask (M) and the pixels in the instance segmentation label (L), I calculated the Intersection over Union (IoU):

$$\text{IoU(L, M)} = \frac{\text{L} \cap \text{M}}{\text{L} \cup \text{M}} \tag{9}$$

I calculated the IoU for each frame and averaged them over each tracking scenario. The mean IoU for each object for the A2D2 is depicted in Tab. 4. As in the previous sections, I repeated the experiments with three random seeds. I calculated the mean and standard deviation over all tracking scenarios.

An important observation is that the standard deviation is larger than the mean for every class,

Class	#Instances	#Pixels	Gold label mIoU	Similarity mIoU
Car	2165.667	7928.520	0.031 ±0.078	0.031 ±0.077
Pedestrian	286.333	2524.331	0.011 ±0.024	0.011 ±0.023
Truck	388.333	20027.244	0.099 ±0.173	0.098 ±0.169
Small vehicle	7.667	943.500	0.014 ±0.033	0.014 ±0.032
Utility vehicle	30.667	11498.140	0.046 ±0.067	0.043 ±0.061
Bicycle	199.667	2229.037	0.018 ±0.045	0.016 ±0.034
Total	3078.333	7525.129	0.037 ±0.094	0.037 ±0.093

Table 4: A2D2 IoU

which shows that the performance of the tracker varies strongly between different instances and scenarios.

When using the best similarity metric as stopping criterion, the mean IoU is almost the same as using the instance segmentation labels. This shows, that on average, the stopping criterion is effective at ending the tracking scenario before the tracker jumps to another object.

Another observable trend is the tendency of higher IoU scores for larger object classes , i.e., objects with a larger amount of pixels per instances on average, like "Truck" and "Car" compared to smaller classes like "Pedestrian" and "Bicycle".

The mean IoU for each object in the KITTI dataset is given in Tab. 5.

Comparing the classes which are shared between the two datasets, namely "Car" and "Pedestrian", shows that the tracker yields about a threefold improvement on the KITTI dataset, compared to the A2D2. However, there is a difference between the two datasets, which could be responsible for this discrepancy. The the average amount of instances per frame is 6.460 for the KITTI dataset and more than twice for the A2D2 with 14.356. As previously described, SiamMask was not explicitly trained to distinguish between different instances of the same

Class	#Instances	#Pixels	Gold label mIoU	Similarity mIoU
Car	843	9115.629	0.117 ±0.136	0.117 ±0.139
Pedestrian	300.333	2250.230	0.030 ±0.027	0.030 ±0.027
Total	1143.333	5682.930	0.094 ±0.124	0.094±0.126

Table 5: KITTI IoU

vehicle. This leads to a decreasing performance with an increasing number of instances per frame.

To correct for this difference, I repeated the previous experiments but excluded all scenes, where the amount of different instances in the initial frame is larger than 10. The resulting number of instances per frame are 5.323 for the A2D2 and 5.442 for the KITTI dataset. Tab. 6 shows, that the total number of instances drastically decreases for A2D2. For some classes like "Pedestrian", "Small vehicle", "Utility vehicle" and "Bycicle" the number of instances is too small to accurately determine the mean IoU. However, for the classes "Car" and "Truck", the mean IoU strongly increases. On the other side, the number of instances in the KITTI

Class	#Instances	#Pixels	Gold label mIoU	Similarity mIoU
Car	333.667	11495.939	0.087 ±0.147	0.088 ±0.148
Pedestrian	13.000	4974.050	0.016 ±0.026	0.016 ±0.026
Truck	121.667	38390.308	0.200 ±0.235	0.194 ±0.228
Small vehicle	1	263.0	0.121 ±0	0.121 ±0.0
Utility vehicle	6.333	4272.625	0.074 ±0.077	0.074 ±0.077
Bicycle	9.000	8452.368	0.094 ±0.138	0.073 ±0.091
Total	484.666	11308.048	0.114 ±0.178	0.112 ±0.174

Table 6: A2D2 IoU with less instances

Class	#Instances	#Pixels	Gold label mIoU	Similarity mIoU
Car	626.000	9956.807	0.136±0.145	0.136 ±0.148
Pedestrian	191.0	2295.058	0.027 ±0.028	0.027 ±0.028
Total	817.000	5682.930	0.112 ±0.124	0.111±0.138

Table 7: KITTI IoU with less instances

dataset, depicted in Tab. 7 decreases less and the differences in mean IoU are much smaller.

6 Future work

There are numerous extensions to this project. A first approach, which likely improves the performance of the SiamMask tracker on the datasets, is to fine-tune the model on an autonomous driving dataset. The analysis showed, the model performs worse, if the number of instances in a frame increases. I suspected this to be a consequence of the training procedure, which did not require the model to learn to differentiate between vehicle instances. Consequentially, fine-tuning the model on instance segmentation labels should reduce the amount of cases, in which the tracker selects the wrong instance of a vehicle.
Similar to pretraining the whole model, the end-of-track detection could also be improved by pretraining a separate network to distinguish between different vehicle instances. Using this network as an autoencoder should give a superior performance.
A disadvantage of those approaches is the requirement of high-quality labels. A method to improve the end of track detection without labels would be asking human annotators to mark the frame, in which an object disappears. These new labels could then be used to updated the model.
An improvement for the A2D2 would be to preprocess the labeled subset by filling the gaps between two frames with the missing frames from the unlabeled dataset. According to the results of the end-of-track detection, it seems likely that these gaps worsen the performance of the tracker. Filling them with the correct but unlabeled frames could help the tracker to follow the object. On the other side, this approach would increase the difficulty of evaluating the SiamMask tracker, as without labels, it is not possible to determine, at which exact frame an object disappears.

7 Conclusion

In this project I applied Single Object Tracking on the autonomous driving datasets A2D2 and KITTI, using the Object Tracking method SiamMask. During the analysis, I encountered difficulties in detecting, when the tracker fails, or when an object disappears from a video frame. For the A2D2, this problem is amplified by the fact, that there are missing frames in the labeled subset, resulting in large differences between two successive frames. However, the experiments indicated, that it is possible to find suitable stopping criteria, which ameliorate this problem. I applied these criteria to evaluated how accurate the tracker is, by calculating the mean IoU for each vehicle in the dataset. The results showed a large discrepancy between the scores for different classes. Overall, SiamMask appeared to be more effective at tracking larger objects and objects in scenes with less object instances. Initially, measuring the mean IoU on the KITTI dataset showed a threefold increase over the A2D2. However, correcting for the number of object instances, decreased this difference. While results are not competitive, the analysis showed that it is possible overcome problems within the datasets. For further experiments, it would be interesting to see, how the system can be improved through fine-tuning or other modifications.

References

Luca Bertinetto, Jack Valmadre, João F Henriques, Andrea Vedaldi, and Philip H S Torr. Fully-convolutional siamese networks for object tracking. In *ECCV 2016 Workshops*, pages 850–865, 2016.

D. Bolme, J. Beveridge, B. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a"siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7 (4):669–688, 1993.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

Andreas Ess, K. Schindler, B. Leibe, and L. Gool. Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research*, 29:1707 – 1725, 2010.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Jakob Geyer, Yohannes Kassahun, Mentar Mahmudi, Xavier Ricou, Rupesh Durgesh, Andrew S. Chung, Lorenz Hauswald, Viet Hoang Pham, Maximilian Mühlegg, Sebastian Dorn, Tiffany Fernandez, Martin Jänicke, Sudesh Mirashi, Chiragkumar Savani, Martin Sturm, Oleksandr Vorobiov, Martin Oelker, Sebastian Garreis, and Peter Schuberth. A2d2: Audi autonomous driving dataset, 2020.

David Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *ECCV*, 2016.

Bo Li, Junjie Yan, Wei Wu, Zheng Zhu, and Xiaolin Hu. High performance visual tracking with siamese region proposal network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Tsung-Yi Lin, M. Maire, Serge J. Belongie, James Hays, P. Perona, D. Ramanan, Piotr Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler. MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*, March 2016. arXiv: 1603.00831.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, December 2015. ISSN 0920-5691. doi: 10.1007/s11263-015-0816-y. Publisher Copyright: © 2015, Springer Science+Business Media New York. Copyright: Copyright 2015 Elsevier B.V., All rights reserved.

Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Qiang Wang, Li Zhang, Luca Bertinetto, Weiming Hu, and Philip HS Torr. Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019.

Ning Xu, Linjie Yang, Yuchen Fan, Jianchao Yang, Dingcheng Yue, Yuchen Liang, Brian L. Price, Scott Cohen, and Thomas S. Huang. Youtube-vos: Sequence-to-sequence video object segmentation. *CoRR*, abs/1809.00461, 2018.

Dawei Zhao, Hao Fu, Liang Xiao, Tao Wu, and Bin Dai. Multi-object tracking with correlation filter for autonomous vehicle. *Sensors*, 18(7), 2018. ISSN 1424-8220.

Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13 (4):600–612, 2004.