Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Scale Estimation in Visual Object Tracking

Bachelorthesis Addendum
at Research Group Knowledge Technology, WTM
Prof. Dr. Stefan Wermter

Department of Informatics
MIN-Faculty
Universität Hamburg

submitted by
Finn Rietz B.Sc.
Course of study: Informatik
Matrikelnr.: 6799896
on
23.10.2019

Examiners: Dr. Stefan Heinrich

Abstract

This document acts as an addendum to the bachelor thesis “Scale Estimation in Visual Object tracking“ [3]. Here, we address minor flaws, changes and other improvements that were made after the submission of the final thesis. The hyperparameter optimization has been partially redone, as some hyperparameters that were dependent have been optimized independently of one another. Further, slight changes have been made to the Candidates algorithm, which results in slightly better results. Lastly, the configuration of the HIOB tracker has not been ideal, which has also been addressed. The DSST algorithm by Danelljan et al. has not been changed or adjusted [1]. Thus, the experiments that are affected by the abovementioned changes have been repeated and the final results are presented.

Zusammenfassung

Dieses Dokument stellt einen Nachtrag zu der Bachelorarbeit “Scale Estimation in Visual Object tracking“ dar [3]. Hier addressieren wir kleinere Mängel, Änderungen und andere Verbesserungen, welche nach der Einreichung der finalen Arbeit vorgenommen wurden. Die Hyperparameteroptimierung wurde teilweise erneut durchgeführt, da einige abhängigde Hyperparameter unabhängig voneinander optimisiert worden waren. Auerdem wurden kleine Änderungen an dem “Candidates“ Algorithmus vorgenommen, welche zu besseren Ergebnissen führen. Abschlieend, wurde die Konfiguration des HIOB Trackers angepasst, da diese nicht optimal war. Der “DSST“ Algorithmus nach Danelljan et al. wurde weder angepasst noch geändert [1]. Dementsprechend wurden die Experimente, welche von den genannten Änderungen betroffen sind, wiederholt und die Ergebnisse werden präsentiert.

Contents

1 Addressing the original results 1

1.1 Tracker configuration . . . 1

1.2 Non-ideal hyperparameter optimization . . . 1

2 Changes and adjustments 3

2.1 Rating a candidate . . . 3

2.2 Refactoring . . . 4

2.3 Optimal hyperparameter settings . . . 4

2.4 Exhaustive grid search . . . 5

3 New results 9

3.1 Results on the TB100 dataset . . . 9

3.2 Results on NICOVISION dataset . . . 13

3.3 Selected sequence analysis . . . 14

4 Appendix: Original results graphs 21

Bibliography 23

Chapter 1

Addressing the original results

Here, we will quickly address the results regarding the performance of the algorithm in Candidates algorithm, that were presented in the original work and how they were affect by a non-optimal tracker configuration.

1.1 Tracker configuration

In the original work, we were not able to reproduce similarly good results that were previously reported by Heinrich et al. [2]. That is, without using either one of the two implemented scale estimation algorithms, we were not able to achieve comparable results. After the thesis had been submitted, the root of this problem has been identified to be a non-optimal configuration of the baseline HIOB tracker. Concretely, a hyperparameter had not been set, that controlled the resolution of the CNN produced feature map. This caused HIOB to use the default value of 25 times 25 pixels for the feature mask parameter. In the original thesis, in chapter 4, exemplary images of the feature map are provided, where we can inspect the low resolution of the feature map. A second, additional parameter had not been set ideally. This parameter controlls HIOBs internal *sroi size*, which controlls the size of the image that will be fed into the static CNN. Thus, this paramter has been fixed and set to 360 times 360 pixels. The value has been selected since it indicates a reasonable tradeoff between the computational load and the quality of the tracking results [2].

The consequences of this non-ideal hyperparameter setting will be explored in greater detail in section 2.3

1.2 Non-ideal hyperparameter optimization

As mentioned in the previous section, some hyperparameters were not set optimally. These hyperparameters are based on the core HIOB tracker, as implemented by Springstübe [4]. However, a second set of hyperparameters has been introduced by Rietz, which controls the behavior of the two implemented scale estimation algorithms [3]. To determine the optimal setting for these newly introduced hyperparameters, a hyperparameter optimization had been conducted. However, one misstake had been made in the optimization of the Candidates algorithm: The hyperparameters *inner punish threshold* and *outer punish threshold* had been optimized independently of one another. This is critical, since the direct interplay of these two parameters controls how each scaled candidate is rated (see Section 4.2.2 of the original thesis for an in depth explanation how these two parameters interact [3]).

In this addendum, the interplay of these parameters has been considered and a exhaustive grid search has been conducted to determine the optimal setting for the two dependent hyperparameters. See section 2.4 for the results of this optimization.

Chapter 2

Changes and adjustments

In this chapter, we quickly examine the changes that were made to achieve the results presented in chapter 3. Mainly, the way the punishment score for each candidate is calculated has been slightly adjusted. Beyond that, we conducted a grid search regarding two specific hyperparameters, adopted the proper configuration values for the underlying HIOB tracker and did some minor refactoring, which resulted in a noteable speedup of the scale estimation process.

2.1 Rating a candidate

As described in chapter 4 of the original bachelor thesis, each candidate that is generated during the attempt to estimate the scale of the object, is rated based on two scores: the *Outer punish value* (OPV) and the *Inner punish value* (IPV) [3]. Further, the IPV is calculated based on the hyperparameter *inner punish threshold*, which captures the threshold below which a candidate's rating gets punished for containing poor likelihood values. Thus, the IPV part of the overall rating each candidate receives tries to control the theoretical growth of the candidates. If it were not for this IPV score, the best tactic for the algorithm would be to simply increase the size of the object as much as possible, to obtain the highest possible score for the OPV. Simply put, the IPV score punishes a candidate for containing pixel values that have a low likelihood of belonging to the target object, as indicated by the feature map, and serves as a counterpart to the OPV score. Thus, the interplay of the two scores (and thus the interplay of the two hyperparameters) how each scale candidate is rated.

In the original version of the algorithm, the IPV score had been obtained from summing up the values on the feature map that fell below the parameterized threshold. While this is not systematically incorrect, note that those values are *smaller* than the parameterized threshold. Thus, this IPV value was always relatively small compared to it's OPV counterpart. This is because for the OPV, we sum up *good* (thus *high*) values on the feature map and for the IPV, we sum up *bad* (thus *low*) values.

As a consequence, the calculation for IPV has been adjusted, to raise its influ-

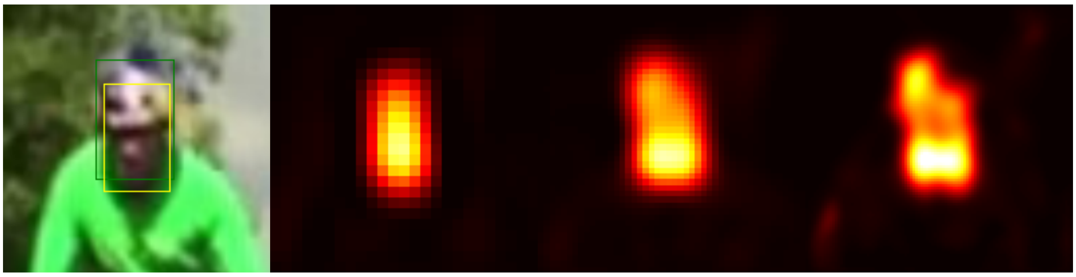

(a) Sroi image (b) 25 × 25px (c) 46 × 46px (d) 100 × 100px

Figure 2.1: An examplary frame, alongside the CNN feature maps at increasing resolutions

ence on the overall rating to be equal to that of the OPV score. Now, instead of directly summing those values on the feature map (which fall below the threshold given by the *inner punish threshold* parameter), we sum up 1 minus that value. Since the feature map contains values between one and zero we can say that we no longer sum the probability of a pixel value belonging to the object, instead, we sum the inverse probability of that pixel belonging to the object (as indicated on the feature map). Thus, the importance of the the IPV score is raised in the updated implementation.

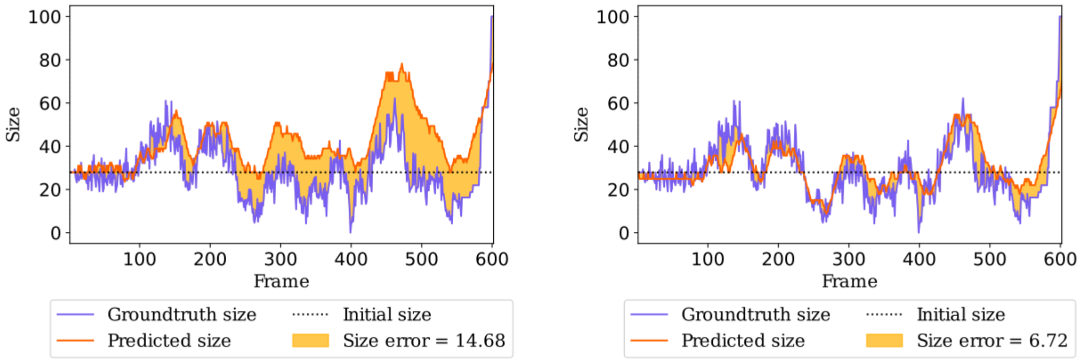
By changing the importance of the IPV score, were able to achieve slightly increased results in the success metric. See Figure 2.3 for a comparsion of the success scores achieved by the two different implementations.

2.2 Refactoring

The adapted version of the HIOB tracker, that had been produces in the scope of the original bachelor thesis and produced the final results that are presented in the thesis, contained some redundant calculations. These resulted from persuing multiple possible solutions to various problems that were encountered during the adaptation of the HIOB tracker. For the results that are presented in the final chapter of this addendum, these redundant calculations have been identified and removed, which caused a

2.3 Optimal hyperparameter settings

As mentioned in section 1.1, some parameters were not set optimally in the configuration file. This has been corrected. To visualize the consequence of the feature mask parameter, consider the images of the feature map in Figure 2.1. Here, the higher resolution of the feature map provides a clearer, more precise representation of the object. Intuitively, we can expect better results using this representation. It should be noted, that a higher resolution of the feature map comes with an increased computational load.



(a) Size graph of Candidates algorithm using 46 × 46px feature mask.

(b) Size graph of Candidates algorithm using 100 × 100px feature mask.

Figure 2.2: Comparasion of the size graphs of the Candidates algorithm (static aspect ratio, max execution strategy) using different feature mask resolutions, to underline the effect of the parameter for scale estimation results.

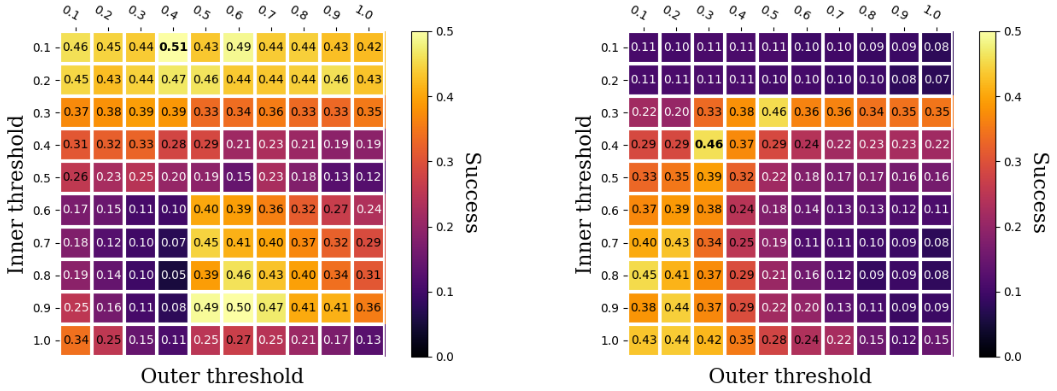
It is apparent, how specifically the Candidates algorithm is affected by the resolution of the feature map since the Candidates algorithm operates *on* the feature map. Specifically, the Candidates algorithm evaluates scaled candidates (here, a candidate simply refers to a bounding box), by mapping the candidate onto the feature map. By having a feature map of higher resolution, this enables the algorithm to make a more precise adjustment to the changing scale of the object. See Figure 2.2 for an example where we can observe better scale estimation results as a result of a higher resolution in the feature map.

The implementation of the Discriminative Scale Space Tracking (DSST) algorithm is only indirectly affected by the change in the hyperparameter setting. The DSST algorithm operates independently of the feature map, specifically, on a Histogram of oriented Gradients (HOG) scale space representation of the target object [1]. Thus we can expect better results from the DSST algorithm, as a consequence of the overall better performance of the HIOB tracker.

2.4 Exhaustive grid search

As explained in section 1.2, the hyperparameter optimization that had been conducted in the original thesis was flawed with respect to the two dependent hyperparameters *inner punish threshold* and *outer punish threshold*. We already mentioned in the original thesis that the separate optimization of the two hyperparameters was not ideal [3]. However, we were unable to react to this realization while staying within the limit of the deadline for the submission of the thesis. Hence, we can only adress the problem in this addendum.

For this addendum, we conducted an exhaustive grid search on the parameter space to dtermine the ideal values for the two hyperparameters. We tested each



Inner threshold \ Outer threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.46	0.45	0.44	**0.51**	0.43	0.49	0.44	0.44	0.43	0.42
0.2	0.45	0.43	0.44	0.47	0.46	0.44	0.44	0.44	0.46	0.43
0.3	0.37	0.38	0.39	0.39	0.33	0.34	0.36	0.33	0.33	0.35
0.4	0.31	0.32	0.33	0.28	0.29	0.21	0.23	0.21	0.19	0.19
0.5	0.26	0.23	0.25	0.20	0.19	0.15	0.23	0.18	0.13	0.12
0.6	0.17	0.15	0.11	0.10	0.40	0.39	0.36	0.32	0.27	0.24
0.7	0.18	0.12	0.10	0.07	0.45	0.41	0.40	0.37	0.32	0.29
0.8	0.19	0.14	0.10	0.05	0.39	0.46	0.43	0.40	0.34	0.31
0.9	0.25	0.16	0.11	0.08	0.49	0.50	0.47	0.41	0.41	0.36
1.0	0.34	0.25	0.15	0.11	0.25	0.27	0.25	0.21	0.17	0.13

(a) New punish score calculation. The highest success score is achieved with the *outer punish threshold* set to 0.4 and the *inner punish threshold* set to 0.1

Inner threshold \ Outer threshold	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0.1	0.11	0.10	0.11	0.11	0.11	0.10	0.10	0.09	0.09	0.08
0.2	0.11	0.11	0.11	0.11	0.10	0.10	0.10	0.10	0.08	0.07
0.3	0.22	0.20	0.33	0.38	0.46	0.36	0.36	0.34	0.35	0.35
0.4	0.29	0.29	**0.46**	0.37	0.29	0.24	0.22	0.23	0.23	0.22
0.5	0.33	0.35	0.39	0.32	0.22	0.18	0.17	0.17	0.16	0.16
0.6	0.37	0.39	0.38	0.24	0.18	0.14	0.13	0.13	0.12	0.11
0.7	0.40	0.43	0.34	0.25	0.19	0.11	0.11	0.10	0.09	0.08
0.8	0.45	0.41	0.37	0.29	0.21	0.16	0.12	0.09	0.09	0.08
0.9	0.38	0.44	0.37	0.29	0.22	0.20	0.13	0.11	0.09	0.09
1.0	0.43	0.44	0.42	0.35	0.28	0.24	0.22	0.15	0.12	0.15

(b) Old punish score calculation. The highest success score is achieved with the *outer punish threshold* set to 0.3 and the *inner punish threshold* set to 0.4

Figure 2.3: Results of the grid search on the parameter space to determine the optimal values for the *inner punish threshold* and *outer punish threshold* parameters

pair of combinations of values for the two hyperparameters. Since the Candidates algorithm itself has been slightly changed in the meantime, the grid search has been executed for both versions of the algorithm. The results from this grid search are visualized in Figure 2.3.

From Figure 2.3 we can observe that the highest success score of the old punishment implementation (which summed the direct probability values that are smaller than the IPV threshold) achieves a maximum success score of 0.46 on training set. For the new implementation of the punishment calculation, the highest achieved success score is 0.51. With the new implementation of the punishment calculation, we can observe interessting results for the IPV parameter space between 0.1 and 0.2. Here, the candidatse algorithm exclusivly achieved success rating greater than 0.4.

For the Candidates algorithm using the old punishment implementation, the results in this area of the parameter space look very different. We can observe that with the old punishment implementation, we achieved alarmingly low success scores, around 0.1. Considering again how this metric is calculated, this score corresponds to an overlap between the predicted bounding box and the ground truth bounding box of roughly 10% during tracking. Those poor success scores emphasize the problem with the old punishment calculation. When the IPV threshold is set to 0.2, we only punish a scaled candidate for containing locations that have a probability of belonging to the object that is smaller than 0.2. For all those pixel locations, the values are than summed up. Comparing this to the OPV threshold, we punish the scaled candidate for not containing pixel locations with a probability greater than, for example, 0.7. Summing up values greater than 0.7 produces a sum that will be greater than summing values smaller than 0.2 (as long as the amount of elements in the sum is roughly equal). This explains why the old punishment implementation achieved such low results in the parameter space between 0.1 and 0.2: The computed IPV value is so small compared to the OPV value, that the best strategy for the algorithm is to make each candidate as large as possible, to minimize the OPV value.

Considering the new implementation of the punishment calculation, this effect has been negated. We now achieve best results when ew set the IPV threshold to a lower value. This means, that the importance of the hyperparameter has been raised and we now only punish a candidate for containing pixel locations with very low (smaller than 0.1) probabilites for belonging to the target object. However, since we now add $1 - \textit{probability of belonging to target}$ for each bad pixel, this produces a large sum, whose impact on the overall rating of each scaled candidate is large enough, so thatcandidates that contain many pixel locations of low probability are punished drastically harder.

Chapter 3

New results

In this chapter, we will present newly obtained results, which result from the most recent version of the HIOB tracker, where the changes from chapter 2 have been applied. Similar to the structure in the original thesis, we will broadly cover the performance of the tracker on the TB100 dataset and the NICOVISON dataset. Further, we will conduct a brief case study on selected sequences from both datasets, which aims at gathering deeper understanding of the capabilites of the improved tracker.

3.1 Results on the TB100 dataset

For now, we will focus on the results the tracker achieved on the TB100 dataset. As for the majority of the original thesis, our main focus lies on the success metric, as this metric measures the overlap between the predicted and ground truth bounding box. We can expect that progress on the scale estimation module of the tracker directly effects the success metric, as better scale estimation correlates with higher overlap between the prediction and the ground truth bounding box.

Before we begin analyzing the results, we should explain how to read the legend: Each legend entry begin with the achieved metric score (either precision or succes) and is followed by the name of the algorithm, that achieved the metric score. This can either be the baseline (No SE), the HIOB tracker with the Candidates algorithm for scale estimation or the HIOB tracker with the DSST algorithm for scale estimation. Next, for both the Candidates or DSST algorithm, either *dyn.* or *stat.* is reported. This refers to the aspect ratio of the generated bounding boxes. In this context, *stat.* means that both the x and the y axis the given bounding box is scale by the same factor, in order to generate scaled candidates for evaluation. Thus, the aspect ratio of the bounding box is given on the initial frame and remains *static* over the entire tracking sequence. The other abbreviation, *dyn.* means that scaled candidates with a *dynamic* aspect ratio are generated for evaluation. These are evaluate in the same way as the other scaled candidates. The motivation behind this was to partly tackle the problem of *in-plane-rotation*, as stated in the original thesis [3].

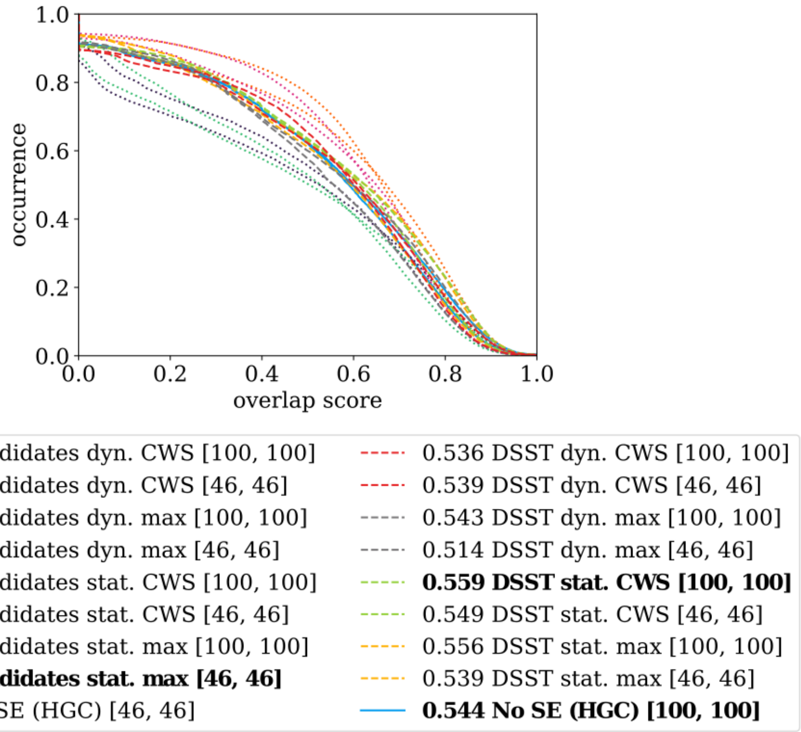


Figure 3.1: Success scores of the of both algorithms and the baseline on thr TB100 dataset. The best version of each algorithm is highlighted in bold.

Following the aspect ration indicator, the labels *max* and *CWS* correspond to the execution strategy of the scale estimtation module. Motivation for this has been drawn from Springstübes original work [4]. Since the scale estimation module can be configured independently of the main tracker, new names have been introduced for the execution strategies. The *max* strategy executes the scale estimation module on every frame, aiming at a continuous and smooth scale prediction, thus *maximizing* the scale calculations. The confidence window strategy (CWS) only executes the scale estimation module on frame that fall in a specified confidence range, where the confidence is a score calculated by HIOB to measure how certain HIOB is regarding the predicted object position. These strategies are explained in greater detail by both Springstübes and Rietz [4, 3]. Finally, the value in the brackets denote the feature map resolution, for which two values have been tested.

Equipped with the above, consider Figure 4.2. The figure shows the the success scores each verion of the algorithm achieved on the TB100 dataset. For each algorithm (Candidates, DSST, Baseline with static size), the version of the algorithm with the highest success score has been highlighted. These results diverge in multiple aspects from the results obtained and reported in the original theses[1]. Firstly, the reported difference between the baseline, the Candidates and the DSST algorithm was marginal [3]. Secondly, on the TB100 dataset, the highest

[1] A slightly improved version of the graphs results reported in the original thesis can be found in the appendix

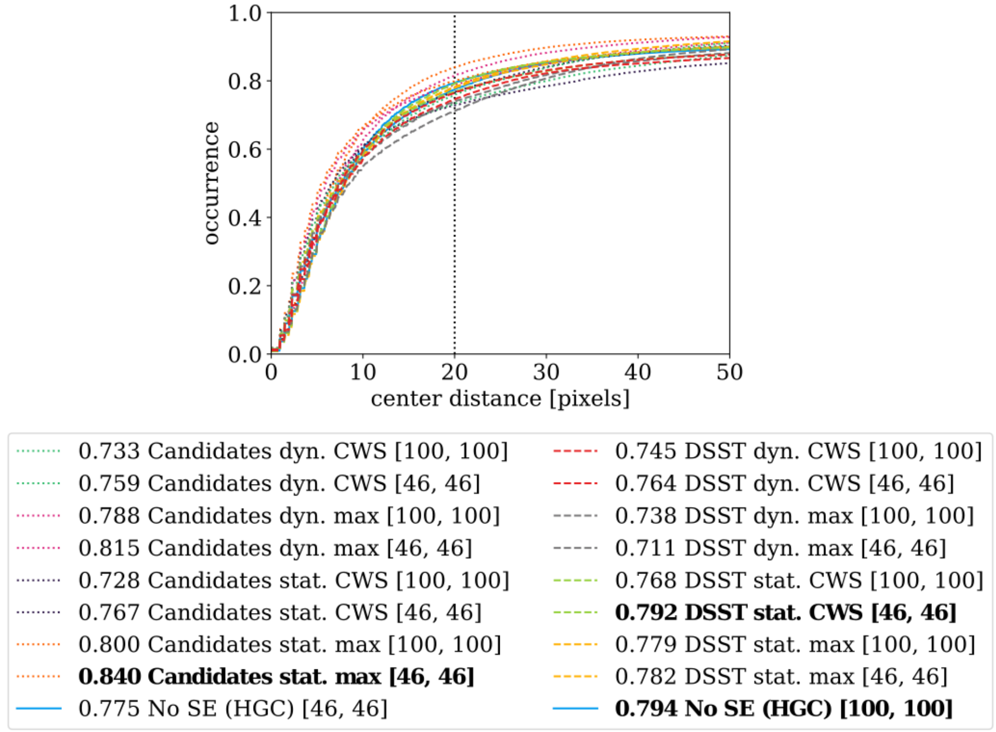


Figure 3.2: Precision scores of the of both algorithms and the baseline on thr TB100 dataset. The best version of each algorithm is highlighted in bold.

success score was obtained by the DSST algorithm using the dynamic aspect ratio implementation and the continuous execution strategy.

In the results presented in Figure 4.2, we can observe that the Candidates algorithm achieves noteably better results that not only the baseline of the HIOB tracker, but also DSST scale estimation algorithm. This can be attributed to the improved punishment calculation of the Candidates algorithm and the higher resolution of the CNN feature map (keep in mind that in the original thesis, the feature map was of resolution 25 × 25px). For the Candiates algorithm, we can further observe that the *max* execution strategy performs drastically better than the *CWS* counterpart. This effect was not identifiable in the results presented in the original thesis. It is likely, that the non-ideal hyperparameter configuration acted as a ceiling effect because the poor tracking accuracy implicitely made the scale estimation problem much harder. Interesstingly, the version of the Candidates algorithm which achieved the highest results, didn't operate on a 100 × 100px feature map. Intuitively, one would expect better results of the algorithm, the higher the quality of the input feature map is. However, this is exactly what is causing the results in Figure 4.2. So far, we didn't consider the precision metric, even though there is a clear correlation between the the precision and the success scores, since the precision metric measures center distance between the predicted and the ground truth bounding box.

With this in mind, consider Figure 4.1. We can observe that the best perform-

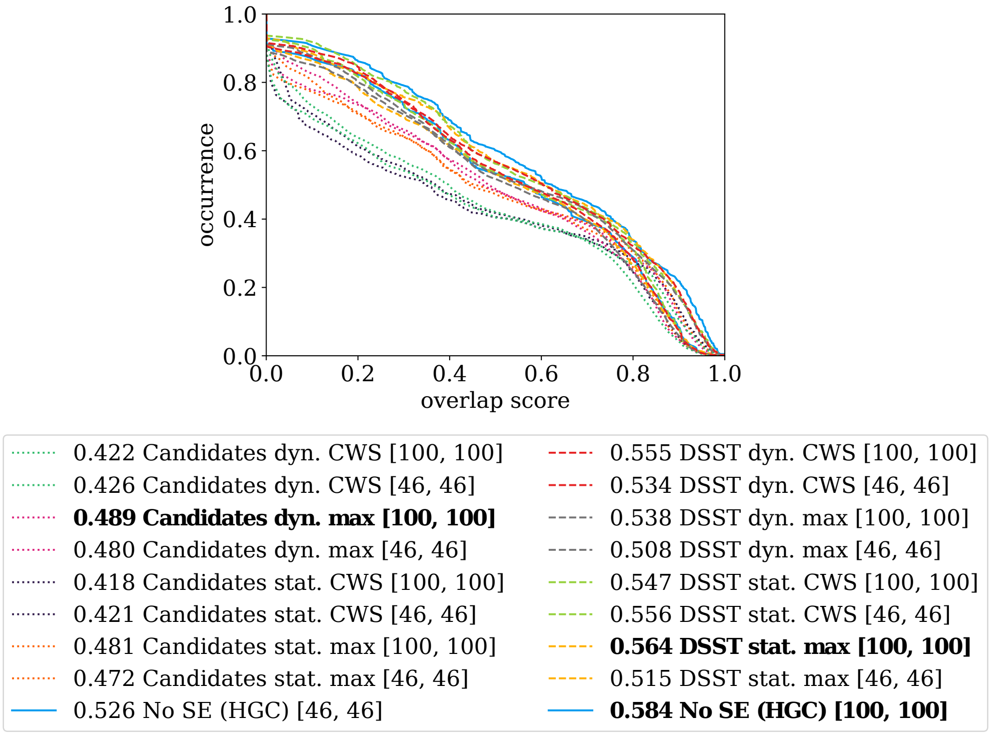


Figure 3.3: Success scores of the of both algorithms and the baseline on thr NICO-VISION dataset. The best version of each algorithm is highlighted in bold.

ing version of the Candidates algorithm with the 46 × 46px feature map has a higher precision rating than its 100 × 100px feature map counterpart. Thus, to a certain extend, this explains why the 46 × 46px version of the Candidates algorithm achieved a higher success score than the 100 × 100px version. The 100 × 100px version had slightly worse tracking input, in terms of precision. Since HIOB generated random positional candidates to find the most likely position for the target object in any frame, tracking results on one sequence cary slightly. Consequently, this also manifests in the success metric and thus affects the scale estimation process. This influence is evidently bidirectional, meaning that dramatic failure of the scale estimation will corrupt hiobs internal object representation, so that HIOB could start considering the background part of the object. However, the takeaway here is that we must consider the precision metric, when we reason about the success metric and the performance of the scale estimation module.

Regading the DSST algorithm, we can observe a much smaller range of success scores. Further, the DSST algorithm still only indicates very slight improvements over the baseline of not running the scale estimation module. We will explore this in greater detail in the sequence analysis. Since the success scores are in such close range, we can hardly make any meaningfull arguments regarding the different versions of the algorithm. Except that there are no significant differences between the versions.

The main take away of these results is the finding, that convolutional features

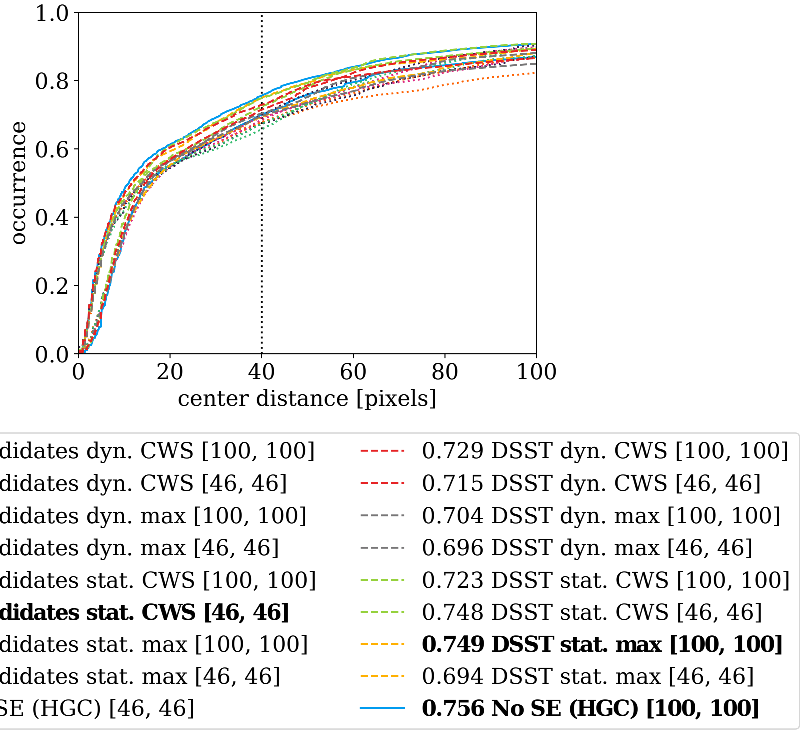
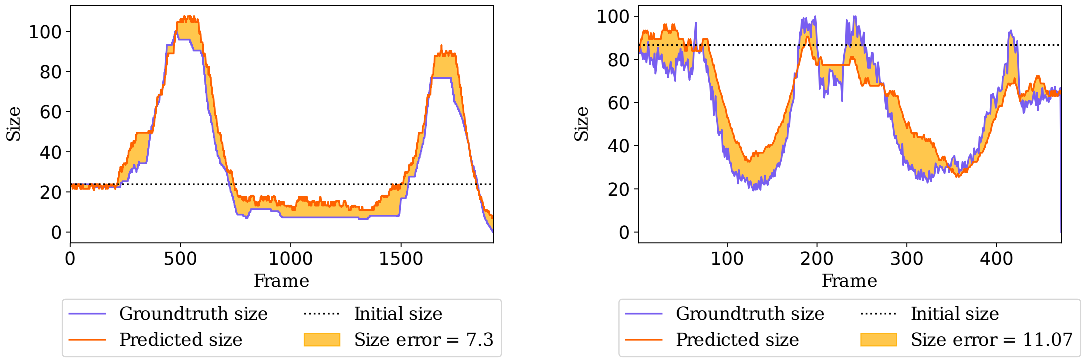


Figure 3.4: Precision scores of the of both algorithms and the baseline on thr NICOVISION dataset. The best version of each algorithm is highlighted in bold.

show strong potential for scale estimation and allow for a drastically better results compared to the baseline.

3.2 Results on NICOVISION dataset

Now, we will expore the results on the NICOVISION dataset. The success plot for the NICOVISION dataset drastically differs from one for the TB100 dataset. Both scale estimation algorithms achieve worse success scores than the baseline, while the DSST algorithm performs noteably better than the Candidates algorithm. We can conclude from this, that the NICOVISION dataset features a set of properties, that not only make scale estimation in general hard but in addition favors the DSST algorithm. The difference between the two datasets has already been noticed by Heinrich et al., who describe strong occlusion by the robot's hands of the object [2]. As already reported in the original thesis, the strong occlusion makes scale esimation on the NICOVISION dataset extremely challenging [3]. In fact, many sequences of the NICOVISION dataset feature NICO grasping an object, followed by rotation or moving the object closer the cameras. This adds additional complexity to the scale estimation problem, which refers to recognizing an object at different scales. In the NICOVISION dataset, we instead have to deal a changing scale under significant transformation of the object (when the object is rotated). Consider **??** for an example of such a sequence.



(a) Size plot of the Candidates algorithm (static aspect ratio, continous execution strategy) on the TB100 sequence *RedTeam*

(b) Size plot of the Candidates algorithm (static aspect ratio, continous execution strategy) on the TB100 sequence *Twinnings*

Figure 3.5: Selected size plots of the Candidates algorithm on the TB100 dataset, indicating strong scale estimation capabilites using using convolutional features.

Considering that the DSST algorithm achieved drastically higher scores than the best performing version of the Candidates algorithm on the NICOVISION dataset allows us to conclude, that the HOG feature representation has some advantages over the convolutional feature for scale estimation on this challenging dataset. For example, the HOG feature representation is a form of edge image (since it consists of image gradients) and as a consequence might be more robust to the changing appearance of the object during manipulation.

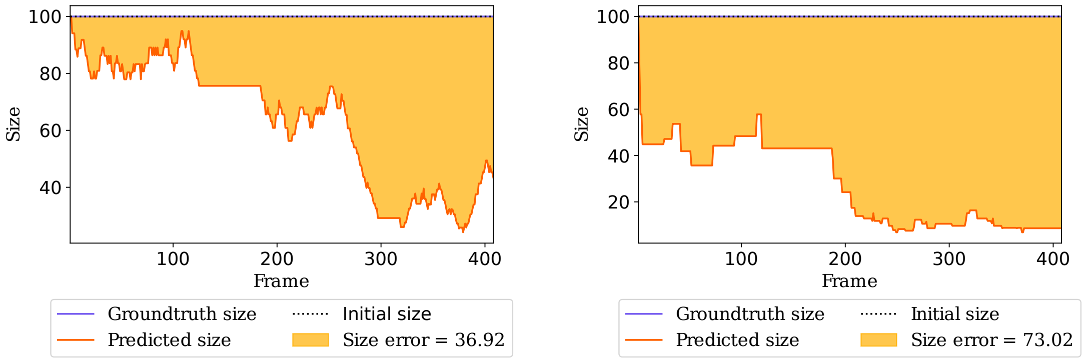
We will explore what causes the different results on the nicovision dataset in depth in subsection 3.3.2.

3.3 Selected sequence analysis

In this section, we explore and explain the results of both algorithms by looking at representative and interesting sequences from both datasets.

3.3.1 TB100 sequences

For the TB100 dataset, we find many sequences which indicate high potential for accurate scale estimation using convolutional features, examplary size plots of such sequences are given in Figure 3.5. However, for the sake of a fair comparision, we want to critically discuss the weaknesses of the scale estimation algorithms, specifically of the Candidates algorithm. Thus, we picked one example, where the baseline implementation of the HIOB tracker achieved good results (0.57 success score), but the Candidates candidates algorithm obtains alarmingly bad results (CWS: 0.21, MAX 0.37 success). Consider the size plots in Figure 3.6. Note how the ground truth size of the object doesn't change throughout the entire tracking



(a) Size plot of the Candidates algorithm (static aspect ratio, continous execution strategy) on the TB100 sequence *Bird1*

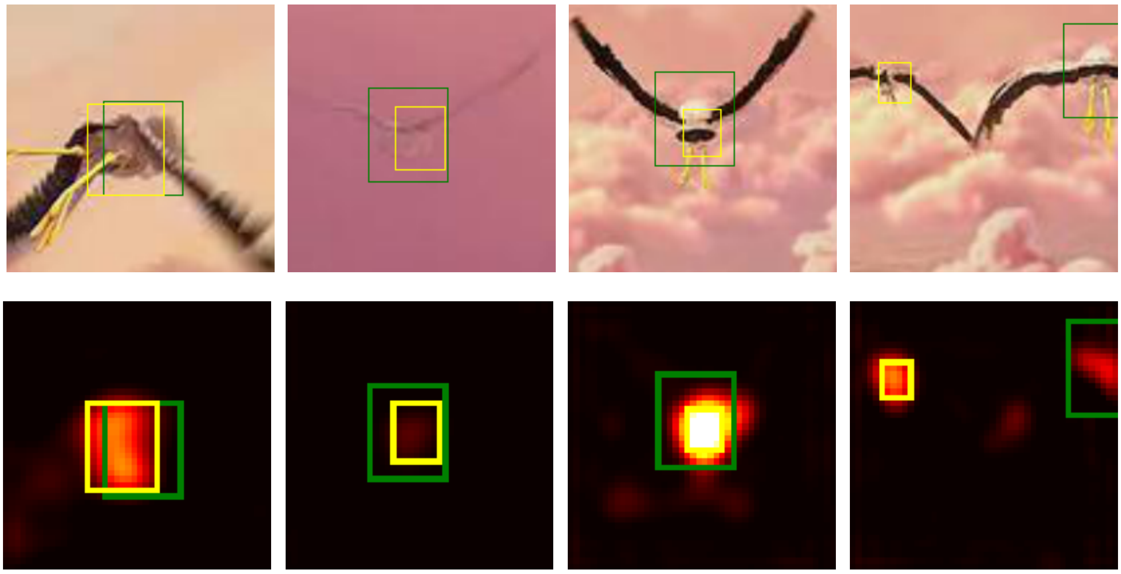
(b) Size plot of the Candidates algorithm (static aspect ratio, CWS execution strategy) on the TB100 sequence *Bird1*

Figure 3.6: Selected size plots of the Candidates algorithm on the TB100 sequence *Bird1*, as an example of poor scale estimation performance using convolutional features.

sequence. While it is unrealistic that an object doesn't change its size at all during navigation in a 3D scene, note how such an annotation essentially eleminates the need to estimathe the scale. This strongly favoes the HIOB baseline, where we never change the size, by always having because we always have a perfect size.

Further, notice how independently of the execution strategy, the scale predicted by the candidates algorithm decreases significantly, even though the ground truth object does not change its size. While the comparasion to the baseline is not fair due to inprecise annotation, we will still explore what caused both versions of the Candidates algorithm to perform this poorly. To understand this behaviour, consider the images in Figure 3.7, which show the convolutional feature map and the SROI image of selected frames for the TB100 sequence *Bird1*.

For both execution strategies, the predicts a strong decrease in the size in the very beginnig of the sequence. For the MAX execution stratevy, this happens gradually, as the scale change is limited to 1% per frame. Thus, the initial decrease in the predicted size isn't as extreme, as this forces the model to update its representation of the blurry body of the bird. For the CWS execution strategy, the scale change is not limited, and the predicted size of the object decrease drastically, within a few frames. Carefull inspection of the ground truth annotation reveals that the bounding box contains a lot of background, which should idially be more precise and contain only the body of the bird. We can attribute the initial drop in size to the annotation of the bounding box, which is too large and secondly, on the blurry *edges* of the bird. This causes the Candidates algorithm to decrease the size of the predicted bounding box to the part of the birds body, where it is can be certain that the pixel actually belong to the bird, which is not given for the blurry edges.



(a) Frame 2 (b) Frame 126 (c) Frame 200 (d) Frame 208

Figure 3.7: The TB100 sequence *Bird1*, on which the Candidates (dynamic bounding box, CWS execution strategy) algorihm performs drastically worse than the baseline.

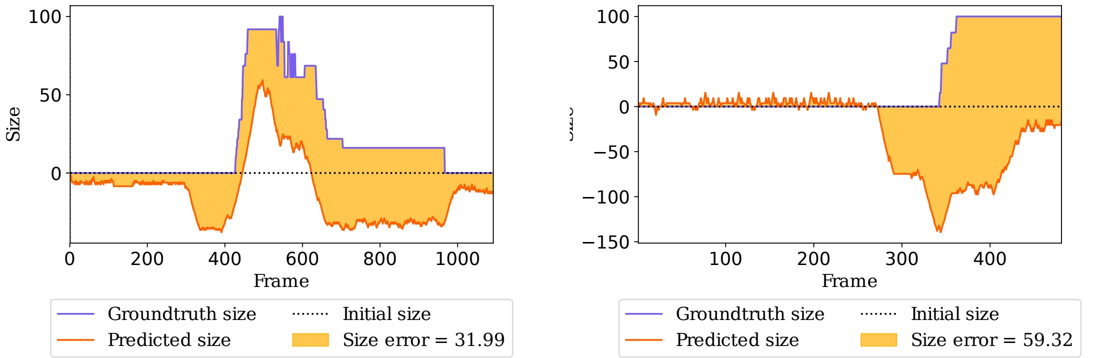
After the initial, strong decrease of the predicted size, the predicted size does not change much, until the bird plunges into a cloud and is fully occluded. Here, the lower limit of the CWS confidence prevents updating the size, since the HIOB's confidence values on the convolutional feature map is too low. This produces the straight line between frame 100 and 200 in the size plots in Figure 3.6.

In the last third of the sequence, a second bird partly occludes the target bird, which causes HIOB to lose the target bird, which was only possible, because hiobs internal representation had already been decrease significantly.

The takeaway from this sequence analysis is that while scale estimation based on convolutional features works well for most cases, in specefic scenarios it can fail and drastically affects the results of the the overall object tracking.

3.3.2 NICOVISION sequences

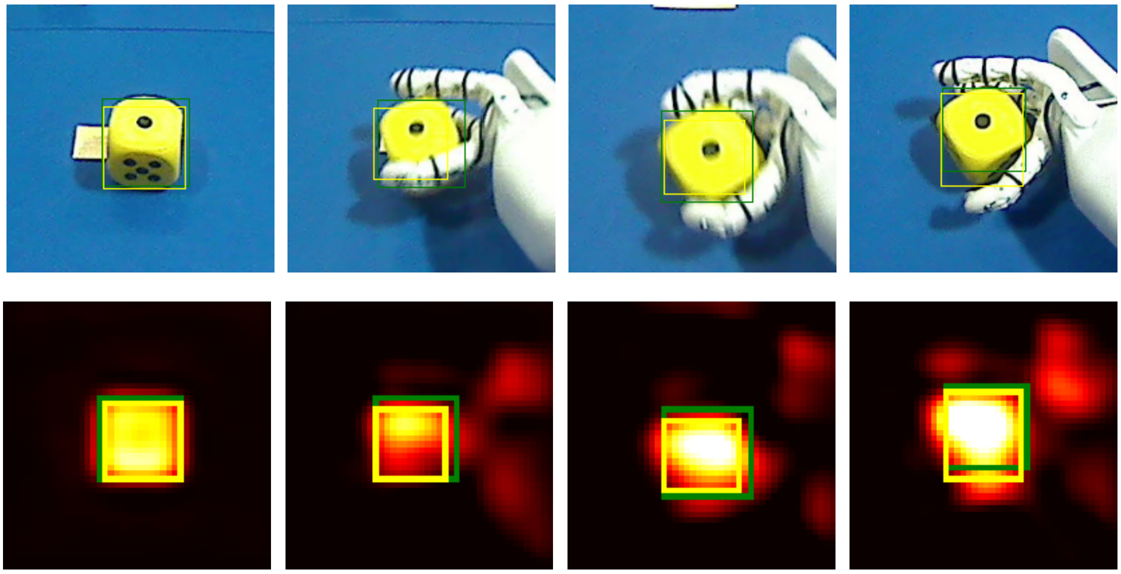
For the nicovisio dataset, the most interesting aspect is why the Candidates algorithm, which achieved strong results on the TB100 dataset, perfroms worse than the baseline of not running a scale estimation algorithm at all. For this, inspect the size graphs in Figure 3.8. For both of these plots, we can observer that the predicted size decreases, shortly before the ground truth size of the object increases. This initial shrinking is caused by the occlusion of the object by the robots hands and makes scale estimation with convolutional features on the NICOVISION dataset particularly challenging. Using the confidence-window strategy (CWS) doesn't help either, in fact, with the confidence window strategy, we get significantly worse results than with the continuous (max) execution strategy. Heinrich et al. report that



(a) Size plot of the Candidates algorithm (dynamic aspect ratio, continous execution strategy) on the NICOVISION sequence *shake-small-yellow-cube-01*

(b) Size plot of the Candidates algorithm (dynamic aspect ratio, continous execution strategy) on the NICOVISION sequence *pull-blue-car-01*

Figure 3.8: Typical size plots for the candidates algorithm using the continuous execution strategy.



(a) Frame 24

(b) Frame 260

(c) Frame 277

(d) Frame 343

Figure 3.9: Example of occlusion and motion blur which produces reducing probability values on the feature map, causing the Candidates algorithm to decrease the predicted sizte.

the confidence windows strategy produced better results (in their work referred to as ”High Gain Combined” (HGC)), specifically when dealing with occlusion [?]. However, for the problem of scale estimation, this does not appear to be the case. At least, the CWS update strategy doesn't solve the problem of occlusion of the object by the the robots hands.

The CWS execution enforces an update of the scale every 20 frames, in the same way as the HGC update strategy in Springstübes work [4]. For the NICOVISION dataset, this in itself could be critical, because the 20'th frame might just be one, where the object is occluded by the robots hands. An additional features distinguishes the CWS strategy from the MAX strategy. For the MAX update strategy, we only allow the object to increase or decrease its size by one percent on each frame, which produces a more stable scale over time. However, for the CWS update strategy, the scale change is not limited, because it is possible that confidence didn't fall within the windows and as such, we didn't update the scale over up to 20 frames. Thus, to allow the algorithm to compensate for the potential strong scale change that could have occured, the scale change under the CWS execution strategy is not limited.

On the NICOVISION dataset, this becomes problematic, because the occlusion of the object often exceeds way over 20 frames, often persisting over the entire interaction of the robot with the object. In such a case, since the object is occluded, there are less values on the feature map that have a high likelihood of belonging to tghe object. As a natural consequence, the scale estimation algorithm will pick up on this, and start to decrease the scale. Since the CWS update strategy is not limited in the scale changes, this decreases the scale much faster, comparred to the continuous upate strategy, where the scale can only change by 1% per frame.

See Figure 3.9 for an example of how the heatmap contains lower values due to occlusion, which deteriorates the scale estimation process.

Understanding how the strong and persisting occlusion of the object manifests on the feature map explains why the Candidates algorithm does not achieve strong results in the NICOVISION dataset. Since the DSST algorithm isn't operating directly on the feature map, it does not face the problem the same problem the Candidates algorithm does. The reason why the Candidates algorithm decreases the size of the object during occlusion is because only what is visible of the object has a high likelihood of belonging to the object. Since the DSST operates on the edge image, it only predicts an increase or decrease in the size, when all edges of the object become bigger or smaller. Thus, occlusion is naturally not as problematic for the algorithm.

However, since the DSST algorithm still doesn't produce stronger results than the baseline, we can conduct that the mixture of occlusion, alongside the rotation and transformation of the object during interaction is too challenging for the edge representation.

Finally, it must be noted that the NICOVISION dataset is annotated in such a way, that the bounding boxes consider the location of the object even under occlusion. From a human perspective, it is clear an intuitive that the object doesn't dissapear when it is occluded. However, for a CNN, which searches for learned patterns in the input image, this human level understanding is missing. Thus, the fact that the Candidates algorithm decreases the size of the object when it is occluded is actually not systematically wrong, it simply acts according to all the information available. To a certain degree, this explains why the scale estimation approach achieves worse results than the baseline under the success metric.

To summarize, the specific set of properties (namely strong occlusion, rotation and transformation of the object) that are present on the NICOVISION dataset are challenging for both algorithms. The DSST algorithm achieves stronger results, because it isn't directly operating on the feature map, which is prone to pick up the effects of occlusion.

Chapter 4

Appendix: Original results graphs

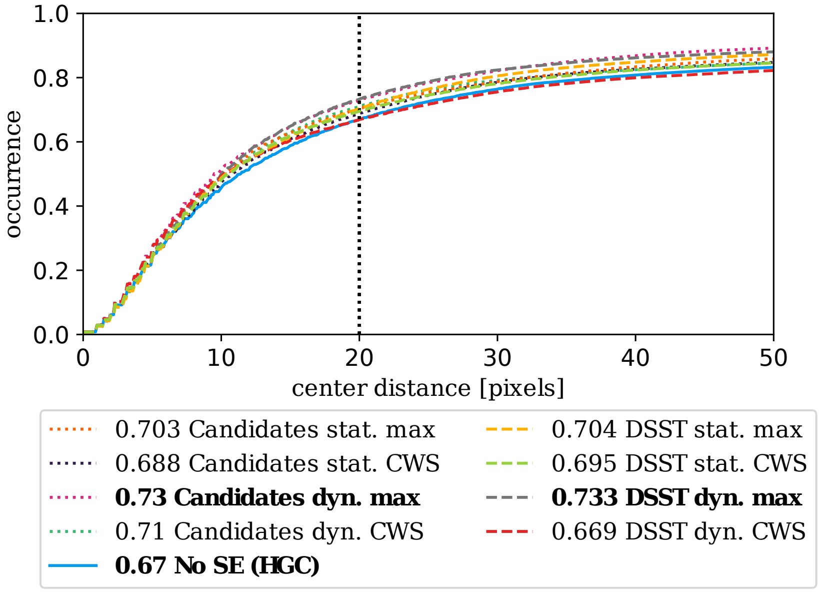


Figure 4.1: Precision scores of the of both algorithms and the baseline on thr TB100 dataset, as reported in the original thesis. The best version of each algorithm is highlighted in bold.

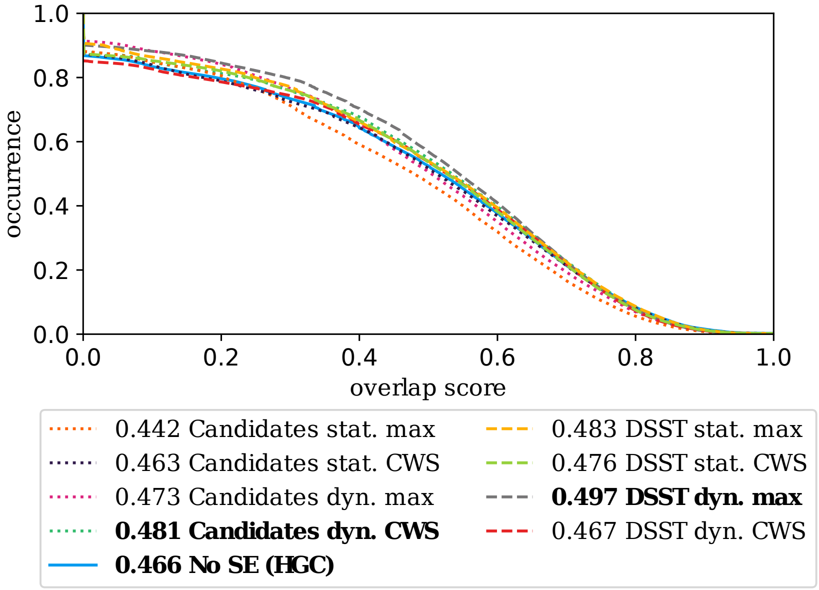


Figure 4.2: Success scores of the of both algorithms and the baseline on thr TB100 dataset, as reported in the original thesis. The best version of each algorithm is highlighted in bold.

Bibliography

[1] Martin Danelljan, Gustav Hger, Fahad Khan, and Michael Felsberg. Discriminative scale space tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8):1561–1575, Aug 2017.

[2] Stefan Heinrich, Peer Springstbe, Tobias Knppler, Matthias Kerzel, and Stefan Wermter. Continuous convolutional object tracking in developmental robot scenarios. *Neurocomputing*, 342:137 – 144, 2019. Advances in artificial neural networks, machine learning and computational intelligence.

[3] Finn Rietz. Scale estimation in visual object tracking. Bachelor's thesis, University of Hamburg, dept. Knowledge Technology, 2019.

[4] Peer Springstübe. Object tracking with convolutional neural networks. Diploma thesis, University of Hamburg, dept. Knowledge Technology, 2017.

Erklärung der Urheberschaft

Hiermit versichere ich an Eides statt, dass ich die vorliegende Bachelorthesis Addendum im Studiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel - insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Ort, Datum Unterschrift

Erklärung zur Veröffentlichung

Ich stimme der Einstellung der Bachelorthesis Addendum in die Bibliothek des Fachbereichs Informatik zu.

Ort, Datum Unterschrift