Real-time and Online Segmentation Multi-target Tracking with Track Revival Re-identification

Martin Ahrnbom[1][a], Mikael Nilsson[1][b] and Håkan Ardö[2][c]
[1]*Centre for Mathematical Sciences, Lund University, Lund, Sweden*
[2]*Axis Communications AB, Lund, Sweden*

Keywords: Multi-target Tracking, Segmentation Tracking, Instance Segmentation, Real-time, Online Tracking.

Abstract: The first online segmentation multi-target tracking algorithm with reported real-time speeds is presented. Based on the popular and fast bounding box based tracker SORT, our method called SORTS is able to utilize segmentations for tracking while keeping the real-time speeds. To handle occlusions, which neither SORT nor SORTS do, we also present SORTS+RReID, an optional extension which uses ReID vectors to revive lost tracks from SORTS to handle occlusions. Despite only computing ReID vectors for 6.9% of the detections, ID switches are decreased by 45%. We evaluate on the MOTS dataset and run at 54.5 and 36.4 FPS for SORTS and SORT+RReID respectively, while keeping 78-79% of the sMOTSA of the current state of the art, which runs at 0.3 FPS. Furthermore, we include an experiment using a faster instance segmentation method to explore the feasibility of a complete real-time detection and tracking system. Code is available: https://github.com/ahrnbom/sorts.

1 INTRODUCTION

Visual object tracking in videos is a key component in modern Computer Vision research. The use of Convolutional Neural Networks (CNN) for object detection has led to a significant improvement in Tracking-by-Detection approaches. Typically, objects' locations are represented by Axis-Aligned Bounding Boxes (AABBs). For several applications, a more detailed description of the position and pose of objects are needed or useful, which has led to the use of segmentation methods that localize objects on a pixel level. Segmentation has typically been limited to single-image tasks, but recently segmentation tracking, where objects are tracked in videos with pixel-level localization, has begun to receive attention in the Computer Vision community. In particular, the CVPR 2020 MOTS Challenge (Voigtlaender et al., 2019) drew attention to this research field. This paper addresses the problem of segmentation tracking, specifically.

The MOTS dataset (Voigtlaender et al., 2019) is a recent and high quality dataset for comparing segmentation tracking results. For the currently tested methods, the fastest one runs at 10 Frames Per Second (FPS), while the rest are slower than 4 FPS. This makes them poorly suited for real-time applications. We introduce the first, to the best of our knowledge, online segmentation multi-target tracking method with reported real-time speeds. Based on the AABB tracker called Simple, Online and Real-Time Tracker (SORT) (Bewley et al., 2016), we introduce the Simple, Online and Real-Time Tracker with Segmentations (SORTS). We extend SORT by using a

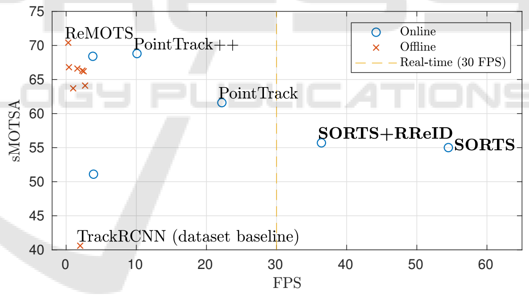

Figure 1: sMOTSA and FPS of all methods tested on the MOTS dataset, including the CVPR 2020 MOTS Challenge. Some highlighted methods are named. All frame rates are reported by the authors. Our methods, in bold, run faster than all other methods, while performing about 78-79% of the state of the art in terms of sMOTSA.

[a] https://orcid.org/0000-0001-9010-7175
[b] https://orcid.org/0000-0003-1712-8345
[c] https://orcid.org/0000-0001-6214-3662

Ahrnbom, M., Nilsson, M. and Ardö, H.
Real-time and Online Segmentation Multi-target Tracking with Track Revival Re-identification.
DOI: 10.5220/0010190907770784
In *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2021) - Volume 5: VISAPP*, pages 777-784
ISBN: 978-989-758-488-6

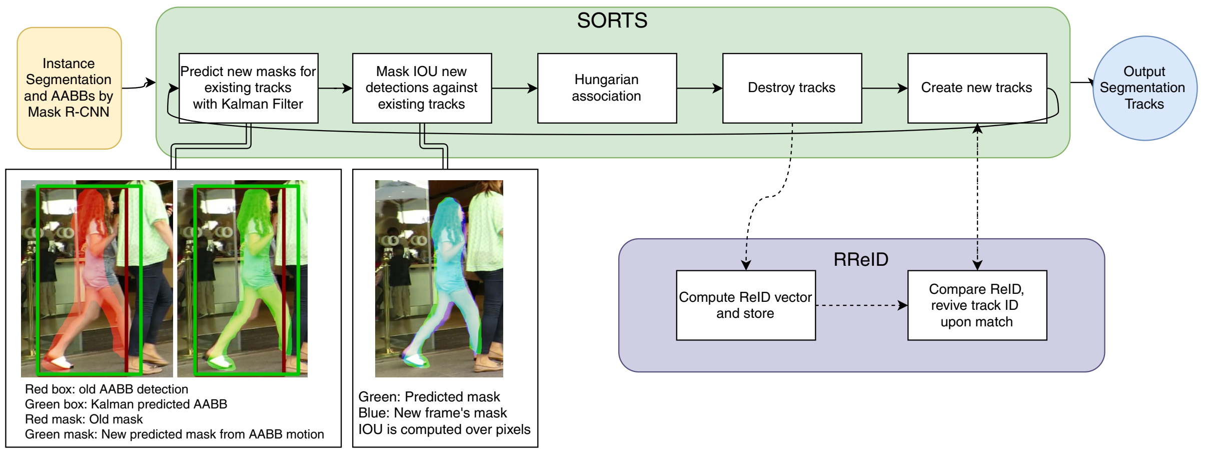
SORTS
Instance Segmentation and AABBs by Mask R-CNN
Predict new masks for existing tracks with Kalman Filter
Mask IOU new detections against existing tracks
Hungarian association
Destroy tracks
Create new tracks
Output Segmentation Tracks
RReID
Compute ReID vector and store
Compare ReID, revive track ID upon match
Red box: old AABB detection
Green box: Kalman predicted AABB
Red mask: Old mask
Green mask: New predicted mask from AABB motion
Green: Predicted mask
Blue: New frame's mask
IOU is computed over pixels

Figure 2: Overview of SORTS and the optional RReID logic (dashed arrows). Existing tracks are updated by a Kalman Filter, and these changes are applied to existing masks. Then, a Mask IOU is used to compare existing tracks with the new detections, and Hungarian association matches them. Tracks that have not been updated for some time are destroyed, and new tracks are started for detections that do have no match. With the optional RReID extension, ReID vectors are computed for destroyed tracks that meet some criteria. When new tracks are created, their ReID vectors are checked against the ReID vectors, and if a match is found, they revive the ID of the previous track. Best viewed in color.

mix of AABB and segmentation logic, to keep the execution speed high while utilizing segmentations for more robust matching of detections.

Because the MOTS dataset does not come with publicly available segmentation detections, we use the popular Detectron2 framework (Wu et al., 2019) and its implementation of Mask R-CNN (He et al., 2017) as a strong baseline for generating the per-frame segmentation detections. SORTS is used for temporally connecting those per-frame segmentations into tracks.

SORT does not handle occlusions, and neither does SORTS, by default. To address this limitation, we also introduce Revival Re-Identification (RReID), an optional extension of our method where ReID vectors, computed by a deep CNN, are used for this purpose. By only computing ReID vectors at carefully chosen times and locations, a high execution speed can still be maintained, although it is significantly slower than the default SORTS method. This is in stark contrast to most segmentation tracking methods that compute ReID vectors at every detection. Alternatively, one could consider SORTS to generate tracklets, that are joined into tracks by the RReID logic.

Our main contributions are:

- We introduce the first online segmentation multi-target tracking algorithm with reported real-time speeds.
- We present an optional extension where ReID vectors are used sparingly to handle occlusions and improve accuracy at the cost of execution speed, while still running above 30 FPS.

2 RELATED WORKS

2.1 Instance Segmentation

One of the most popular instance segmentation methods is Mask R-CNN (He et al., 2017). Based on Faster R-CNN (Ren et al., 2015), the CNN is extended to produce pixel-level segmentation masks for each detected object. While not real-time, Mask R-CNN is widely used for its accurate segmentation masks.

An example of real-time instance segmentation is InstanceMotSeg (Mohamed et al., 2020). They apply segmentation to videos, using optical flow-like motion features, but do not track instances. Their method runs at 39 FPS. Another example is SEG-YOLO (Wang, 2019) which runs at 17-30 FPS, and performs about 5 mAP worse than Mask R-CNN. YOLACT (Bolya et al., 2019) and CenterMask-Lite (Lee and Park, 2020) are other examples of fast instance segmentation networks. Fast methods like these could be used as replacements for Mask R-CNN; an example using CenterMask-Lite is presented in Section 4.7.

2.2 Segmentation Single-target Tracking

These methods only track a single object, and its position is typically given in the first frame. Segmentation single-target tracking has received some attention, for example a Siamese CNN that runs at 55 FPS (Wang et al., 2019), and a method based on an Absorbing Markov Chain model and runs at about 4 FPS (Yeo et al., 2017).

2.3 AABB Multi-target Tracking with Re-identification

AABB-based multi-target tracking is a wide research field and some methods that are related to SORTS and SORTS+RReID are presented here.

SORTs̃ort is a popular and fast AABB-based multi-target tracking method and several approaches are based on it. One example is DeepSORT (Wojke et al., 2017; Wojke and Bewley, 2018) which extends SORT using a ReID network. Unlike SORTS+RReID, they use the ReID network on every detection on every frame, and this contributes to their method being significantly slower, at around 20 FPS. DeepSORT reduced the number of ID switches by 45% compared to SORT, which is the same effect we get when comparing SORTS+RReID and SORTS, indicating that our sparse use of ReID is likely sufficient for this goal, although the numbers were obtained on different datasets.

2.4 Segmentation Multi-target Tracking

A multi-target segmentation tracking method using foreground-background segmentation alongside an AABB-based detector (Milan et al., 2015) frames tracking as a graph cut problem. Their solution is not online, runs at 0.2 FPS and requires static cameras.

A method using a Markov Chain Monte-Carlo approach combined with foreground segmentation (Zhao et al., 2008) performs multi-target segmentation tracking. The method is again limited to scenes with static cameras, and it runs at about 2 FPS.

An example of early works in this field, a segmentation tracking method that runs at 30 FPS, was published in 2010 (Bibby and Reid, 2010). It requires initial AABBs for the targets to be given, so it does not contain logic for creating new tracks as objects appear in the scene. In addition, it can only handle at most 12 tracks at once, so it works more like a single-target tracking algorithm applied on several objects at once. We therefore do not consider this method to be a true multi-target tracking algorithm.

On the CVPR 2020 MOTS Challenge (Voigtlaender et al., 2019), the winner was ReMOTS (Yang et al., 2020), which achieves an excellent sMOTSA score, and runs at 0.3 FPS. This is also the only method with published results on the MOTS dataset, except for the baseline method (TrackRCNN). The closest method to ours in terms of execution speed is PointTrack and PointTrack++ (Xu et al., 2020) that are online and run at 22 and 10 FPS respectively and PointTrack++ won second place at the CVPR 2020 MOTS Challenge. No other competing method was reported to run faster than 4 FPS.

There was another paper at the CVPR 2020 MOTS Challenge that share some ideas with our method (Koeferl et al., 2020), where SORT was used with some tricks to make it work with segmentations. They seem to have run SORT on AABBs while simply outputting the segmentations associated with each AABB. This is in stark contrast to our approach, where the segmentations themselves are used for computing the Intersection Over Union (IOU). They also do not report run time speeds, and their results do not appear on the MOT Challenge website.

In summary, to the best of our knowledge, there has not been any previous online segmentation multi-target tracking methods with reported real-time speeds. Furthermore, in our framework, we introduce a novel revival ReID approach.

3 OUR METHOD

3.1 SORTS

A basic overview of the method is shown in Figure 2. The detector provides both AABBs and segmentation masks for each detection, which are the input to SORTS. The algorithm extends SORT with the following changes:

- When computing IOU, which is done using segmentations to get a pixel-level IOU score. The segmentations are represented as binary maps, NumPy (van der Walt et al., 2011) arrays of `bool`. This allows both intersection and union to be computed as fast binary operations. By cropping the segmentation images based on their AABBs, computations are only performed on the relevant parts of the images. If the AABBs do not overlap, this computation is skipped entirely.
- When the Kalman Filter predicts a new AABB, a predicted segmentation is also created which will then be used when computing IOU with incoming frames. These predicted segmentations are simply the previous segmentation for that track, translated by the same translation as the center points of the AABBs before and after the prediction. No scaling or other modelling is done with the mask.
- When computing IOU, a part at the bottom of each detection is removed. The amount of each detection to be removed is defined by the constant

y_{cutoff}. The reasoning for this is that walking humans typically vary the shape of their legs more quickly as they walk than the rest of their bodies.

- Tracks are not included in the output of SORTS unless it has received detections for a number of frames that correspond to A_{min} seconds, except for the first A_{min} seconds of each sequence. This removes many false and poor tracks typically caused by incorrect detections by Mask R-CNN.
- Detections that are shorter in height than h_{min} are ignored as a preprocessing stage. This avoids unnecessary computations about many tracks that are in the do-not-care regions of the MOTS dataset. The parameter h_{min} is defined as a percentage of the total image height, to scale with different video sizes.

3.2 SORTS+RReID

As an option, we implemented a fast method for using ReID for handling occlusions. The method uses the ReID network defined in (Luo et al., 2019; Luo et al., 2019). In order to keep a high execution speed, ReID vectors are only computed sparingly, on average on only 6.9% of all track appearances.

Whenever a track is considered lost in the method described in Section 3.1, before being discarded, a few checks are performed. If fulfils any of the following conditions, the track is discarded as usual:

- The track is too short with fewer than R_{short} seconds.
- The track's final position was close to the borders of the image within $R_{\text{border}}\%$, except if the track is near *both* the top and bottom of the screen, in which case it is standing close to the camera and should still be included. This prevents RReID from being used on tracks that walk out of the image.
- The height of the track's final AABB is lower than $R_{\text{height}}\%$ of the image height.

Otherwise, a ReID vector is computed for that track. First, a reasonable frame number is chosen, for which the ReID vector will be computed. When SORTS is running in RReID mode, each track stores R_{memory} many previous masks, and if the number of stored masks for the track is at least R_{lookback}, then the R_{lookback} seconds old mask is used. Otherwise, whatever mask is stored in the middle of the track is used instead, to avoid using the first and last frames where it is more likely that the track is partially occluded.

Then, for the chosen frame, the corresponding video frame is loaded into memory, using a cache in case multiple tracks do this for a single frame, and the cache hit-rate is about 40%. The section corresponding to where the track was at that time is extracted from the image, and fed into the ReID network. The ReID network is used without its final pooling layer, providing a spatial map of ReID vectors. The mask that the track had at that time is shrunk by the same factor as the spatial map, using nearest neighbour interpolation. Finally, the average ReID vector corresponding to those pixels that belong to the shrunk mask is computed, and stored for the track. The tracks with ReID vectors computed are stored in a list, where they are kept for R_{storage} seconds.

Then, whenever a track is about to get its ID assigned, after being associated to A_{min} detections, it has a chance to inherit the ID of a previous track, if the ReID vectors match up. Before computing a ReID vector for the track, a few checks are performed, and if any of this conditions hold, the track is simply assigned a new ID and no ReID vector is computed:

- There are no previous tracks with computed ReID vectors.
- The height of the track's latest AABB is lower than $R_{\text{height}}\%$ of the image height.
- The track's latest mask does not have any pixels when shrunk.

Then, the ReID vector of the track is computed as before. The vector is compared to the ReID vectors in the list, picking the one with the highest normalized dot product, assuming it is above R_{thresh}. If such a match can be made, the track inherits the ID of the matched previous track, and the old track is effectively "revived".

4 EXPERIMENTS

4.1 Instance Segmentation

For most of our experiments, we use Detectron2's (Wu et al., 2019) implementation of Mask R-CNN 101 (He et al., 2017). Because the MOTS dataset requires that each pixel belongs to at most one object, a pixel-wise Non-Maximum Suppression (NMS) is implemented as a final stage of the instance segmentation pipeline, similar to (Koeferl et al., 2020). If a single pixel belongs to multiple detections, it gets assigned to the one with the higher confidence score. This operation runs on the GPU at takes about 0.2 ms per image. The total execution time of the instance segmentation step with this model is about 8 FPS, for the batch size one.

The detector was retrained on the MOTS dataset, using pre-trained weights from the MS COCO dataset (Lin et al., 2014) as a starting point. 4-fold cross validation was used for early stopping, using the four sequences as folds. Then, the detector was re-trained for the found mean number of epochs, which was 7500, on the entire training set. All other hyper-parameters were chosen by hand waving, in particular the learning rate (0.00025), batch size (2) and batch size for the ROI heads (256).

4.2 Hyperparameter Optimization

SORTS has several parameters that are non-trivial to tune. From SORT, it has inherited the Kalman Filter (Labbe, 2014) parameters R_2, P_4, P_{scale} and Q_{scale}, as well as a few others:

- A_{max}, the time in seconds without a new detection after which a track is considered finished, although this parameter was measured in frames in SORT
- I_{thresh}, the minimum IOU for a match between new detections and existing tracks

These, and the new y_{cutoff} and A_{min} parameters need to be tuned. A random search was first applied where parameters were picked randomly within hand-picked ranges, and the performance was tested using a score defined by

$$S = \text{sMOTSA} + \frac{\text{FPS}}{500}, \quad (1)$$

where S is the score, sMOTSA is defined by the MOTS dataset, and FPS is the execution speed (in Hz).

The reason for including the FPS in the score is that it was found that certain parameters lead to significantly lower frame rates and those parameters should be avoided if other parameters give similar sMOTSA scores.

The parameters found by random search are then used as a starting point in a Nelder-Mead optimization (Nelder and Mead, 1965; Gao and Han, 2012), optimizing over only the sMOTSA, as smaller changes to the parameters seem to have little impact on the frame rate. To prevent overfitting to the training dataset, only three of the four sequences were used for these optimization strategies, with the fourth (the sequence called '09') being used as a validation set for early stopping.

For SORTS+RReID, the process was similar. The parameters found in Section 3.1 were used as-is. The new parameters R_{thresh}, R_{short}, R_{border}, R_{height}, R_{memory}, R_{lookback} and R_{storage} were optimized, first by random testing over the score S, and then using Nelder-Mead, optimizing over sMOTSA. Again, the '09' sequence was used as a validation set for early stopping.

Table 1: Final parameters for SORTS (left column) and SORTS+RReID (left and right columns).

Parameter	Value	Parameter	Value
R_2	0.171	R_{thresh}	0.897
P_4	3171.191	R_{border}	9.9%
P_{scale}	203.685	R_{memory}	1.739 s
Q_{scale}	0.0277	R_{storage}	1.723 s
A_{min}	0.131 s	R_{short}	0.166 s
A_{max}	0.0487 s	R_{lookback}	0.165 s
y_{cutoff}	35.0%	R_{height}	7.40%
I_{thresh}	0.30		
h_{min}	3.66%		

The optimal parameters found after optimization can be seen in Table 1.

4.3 MOTS Evaluation

The primary experiment of this research was the evaluation of SORTS and SORTS+RReID on the MOTS dataset.

A comparison between SORTS, SORTS+RReID and all other methods with publicly available results on MOTS or the CVPR 2020 MOTS Challenge, in terms of sMOTSA and FPS, is shown in Figure 1. A more detailed table of SORTS and SORTS+RReID for the various metrics is shown in Table 2. Some example output images from the test set can be seen in Figure 5.

4.4 Ablation Study

In order to see the contribution of various aspects of our method, we have evaluated variants of SORTS and SORTS+RReID on the MOTS validation set, sequence '09', because test set evaluations are limited on the MOTS dataset. Parameters are not re-trained in the interest of time. When not training Mask R-CNN, pretrained weights from the MS COCO dataset (Lin et al., 2014) were used instead. The results of the ablation study is presented in Table 3.

4.5 Resolution Dependence

A commonly used approach for creating a trade-off between accuracy and execution speed is to run methods at different image resolutions. To measure this effect on our methods, the three 1080p sequences in the MOTS training set were shrunk to several different commonly used video resolutions. The test was then redone on each resolution, including re-computing the detections. Parameter optimization was skipped

Table 2: Detailed metrics of SORTS and SORTS+RReID on the MOTS test set. The latter is an improvement or the same, in all metrics except the frame rate. Most importantly, the number of ID switches was cut by 45%. Surprisingly, this has very little impact on most metrics even though it can be important for applications.

	sMOTSA	IDF1	MOTSA	MOTSP	MODSA	MT	ML	
SORTS	55.0	57.3	68.3	81.9	70.0	107	52	
SORTS+RReID	55.8	65.8	69.1	81.9	70.0	107	52	
	TP	FP	FN	Recall	Precision	ID Sw.	Frag	FPS
SORTS	23,671	1,076	8,598	73.4	95.7	552	577	54.5
SORTS+RReID	23,671	1,076	8598	73.4	95.7	304	576	36.4

Table 3: Ablation study results on the MOTS validation set (sequence “09”). The two rightmost columns are the final versions of SORTS and SORTS+RReID, respectively. The leftmost column roughly corresponds to standard SORT.

Training Mask R-CNN		✓	✓	✓	✓	✓
Mask IOU			✓	✓	✓	✓
y_{cutoff}				✓	✓	✓
h_{min}					✓	✓
RReID						✓
sMOTSA	63.9	64.5	65.6	65.7	65.7	66.4
FPS	61.4	78.8	48.6	46.5	47.5	40.5

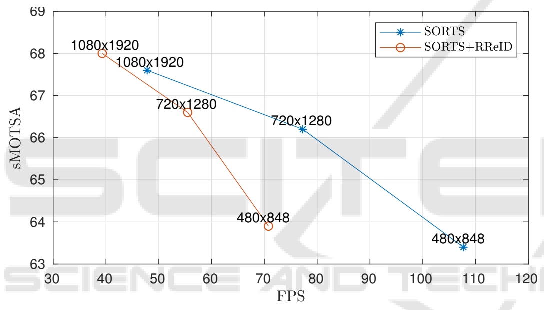

Figure 3: sMOTSA and FPS of SORTS and SORTS+RReID on the high-resolution subset of the training set, downscaled to 720p and 480p. As the resolution decreases, it is possible to get a high execution speed, at the cost of sMOTSA score.

in order to save time. The Mask R-CNN detector ran at 8.2 FPS in 480p, 8.0 FPS in 720p and 7.6 FPS in 1080p. The results can be seen in Figure 3.

4.6 Execution Time Details

All experiments were performed with a Intel Core i7-3930K CPU @ 3.20GHz × 12 CPU, and a Nvidia GeForce GTX 1080 Ti GPU. As is common, not all operations are included in the execution time. We exclude the following, to be consistent with the recommendations of the MOT Challenge (MOT Challenge, 2020):

- The detections
- Reading and writing to files, including images
- Run-length encoding and decoding of segmentation masks to and from text format

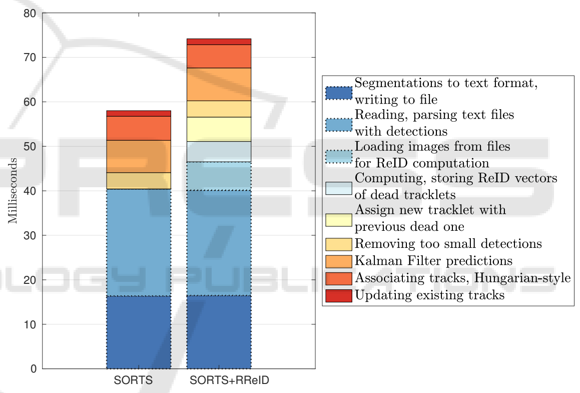

Figure 4: Timing of various parts of SORTS (left) and SORTS+RReID (right) on the test set. The presented times are the average time per frame spent on each task. The “dotted” sections are not included in the calculated FPS, as per the recommendations of the MOT Challenge. Best viewed in color.

We present the execution time of SORTS and SORTS+RReID, divided into different segments for different kinds of computations, in Figure 4.

4.7 Using CenterMask-Lite

For applications where the entire pipeline needs to run in real-time, Mask R-CNN is too slow. For this purpose, we also tested SORTS and SORTS+RReID using CenterMask-Lite (Lee and Park, 2020), a recent and fast instance segmentation method. With our setup, it runs at 17.3 FPS. The difference in execution speed between our setup and the reported 35 FPS could be explained by a combination of factors:

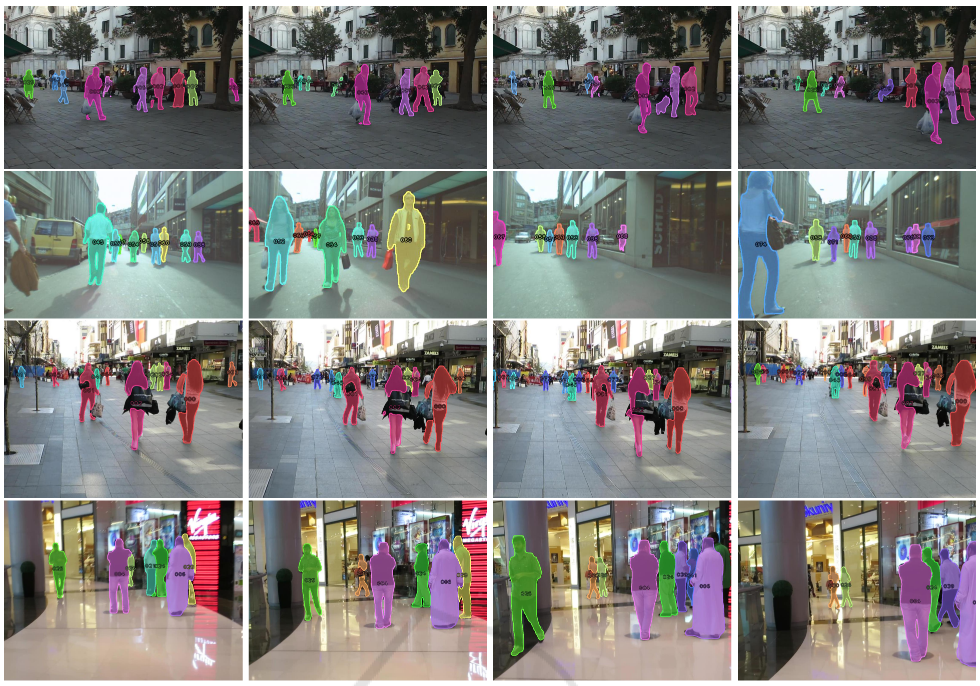

Figure 5: Examples of images from the MOTS test set, with segmentation tracks by SORTS+RReID. Images are taken 30 frames apart. Best viewed in color, preferably on a screen.

them using the more powerful Titan Xp GPU, performing detections in batches and not using pixelwise NMS. Regardless, this is a step towards a true real-time pipeline.

We tested both Mask R-CNN and CenterMask-Lite on our validation set (sequence “09”), after training on the other three sequences from the MOTS training set. We then ran both SORTS and SORTS+RReID on these detections. Going from Mask R-CNN to CenterMask-Lite, the sMOTSA on the validation set dropped from 65.7 to 51.2 for SORTS, and from 66.4 to 52.0 for SORTS+RReID. The execution speed dropped too, to 35 FPS for SORTS and 29 FPS for SORTS+RReID. We believe this is due to CenterMask-Lite creating a larger number of detections, as CenterMask-Lite creates 38% more detections on the MOTS training set than Mask R-CNN, with the same confidence threshold (0.4). While this threshold could be increased, doing so would likely further reduce the tracking accuracy.

5 DISCUSSION

SORTS+RReID greatly improves upon SORTS by reducing the number of ID switches, but this improvement is not really reflected in the sMOTSA score, which puts a large weight on the precise shape of the tracked objects. This should be taken into consideration for real applications, as consistent tracks are sometimes more important than accurate segmentations. Looking at IDF1, which does not depend on the precise shapes, the improvement of SORTS+RReID, compared to SORTS, is more noticeable.

The experiment with CenterMask-Lite shows that real-time instance segmentation is still not quite good enough to compete with Mask R-CNN. But more research will be made, and hopefully a real-time instance segmentation network will soon perform similarly to slower CNNs in terms of accuracy. When that happens, SORTS and SORTS+RReID can be used with it, as a drop-in replacement for Mask R-CNN.

We like to evaluate SORTS and SORTS+RReID on other datasets and in other scenarios, to see how well it works in practice for various applications. We would also like to continue experimenting with faster instance segmentation methods to build a true end-to-end online real-time system. Such a system could have important applications in autonomous driving, robotics and smart city surveillance, for example.

6 CONCLUSION

We present SORTS, and its optional extension SORTS+RReID, the first online multi-target segmentation tracking methods with reported real-time speeds of 54.5 and 36.4 FPS respectively. The sMOTSA scores for these methods were 55.0 and 55.8 on the MOTS test set, which is about 78-79% of the current state of the art which runs at 0.3 FPS. The RReID system is able to cut ID switches by 45% while only computing ReID vectors for about 7% of all track instances, which helps it stay real-time despite the added workload of the ReID network. We have further experimented with and discussed using faster detectors. We hope that SORTS and SORTS+RReID can be a strong baseline for real-time segmentation multi-target tracking in the future.

REFERENCES

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468.

Bibby, C. and Reid, I. (2010). Real-time tracking of multiple occluding objects using level sets. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1307–1314.

Bolya, D., Zhou, C., Xiao, F., and Lee, Y. J. (2019). YOLACT: Real-time instance segmentation. In *ICCV*.

Gao, F. and Han, L. (2012). Implementing the nelder-mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications*, 51:259–277.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Koeferl, F., Link, J., and Eskofier, B. (2020). Application of SORT on Multi-Object Tracking and Segmentation. In *5th BMTT MOTChallenge Workshop: Multi-Object Tracking and Segmentation*.

Labbe, R. (2014). Kalman and bayesian filters in python. https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python.

Lee, Y. and Park, J. (2020). Centermask: Real-time anchor-free instance segmentation.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, Arxiv: 1405.0312.

Luo, H., Gu, Y., Liao, X., Lai, S., and Jiang, W. (2019). Bag of tricks and a strong baseline for deep person re-identification. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Luo, H., Jiang, W., Gu, Y., Liu, F., Liao, X., Lai, S., and Gu, J. (2019). A strong baseline and batch normalization neck for deep person re-identification. *IEEE Transactions on Multimedia*, pages 1–1.

Milan, A., Leal-Taixe, L., Schindler, K., and Reid, I. (2015). Joint tracking and segmentation of multiple targets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Mohamed, E., Ewaisha, M., Siam, M., Rashed, H., Yogamani, S. K., and Sallab, A. E. (2020). Instancemotseg: Real-time instance motion segmentation for autonomous driving. *CoRR*, abs/2008.07008.

Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7:308–313.

MOT Challenge (2020). Mot challenge website. https://motchallenge.net/user_account.php. Accessed 2020-09-02.

Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

van der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The NumPy Array: A structure for efficient numerical computation. *Computing in Science Engineering*, 13(2):22–30.

Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B. (2019). MOTS: Multi-Object Tracking and Segmentation. *arXiv:1902.03604[cs]*. arXiv: 1902.03604.

Wang, Q., Zhang, L., Bertinetto, L., Hu, W., and Torr, P. H. (2019). Fast online object tracking and segmentation: A unifying approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, Z. (2019). *SEG-YOLO: Real-Time Instance Segmentation Using YOLOv3 and Fully Convolutional Network*. PhD thesis.

Wojke, N. and Bewley, A. (2018). Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 3645–3649. IEEE.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. https://github.com/facebookresearch/detectron2.

Xu, Z., Zhang, W., Tan, X., Yang, W., Su, X., Yuan, Y., Zhang, H., Wen, S., Ding, E., and Huang, L. (2020). Pointtrack++ for effective online multi-object tracking and segmentation. In *CVPR Workshops*.

Yang, F., Chang, X., Dang, C., Zheng, Z., Sakti, S., Nakamura, S., and Wu, Y. (2020). ReMOTS: Self-supervised refining multi-object tracking and segmentation.

Yeo, D., Son, J., Han, B., and Hee Han, J. (2017). Superpixel-based tracking-by-segmentation using markov chains. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Zhao, T., Nevatia, R., and Wu, B. (2008). Segmentation and tracking of multiple humans in crowded environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1198–1211.