– Independent Work Report 2020 –

Single Shot Multiple Object Tracking and Segmentation

Ariel Rakovitsky
Adviser: Professor Jia Deng

Abstract

Multiple object tracking and segmentation (MOTS) is a new problem in the field of computer vision that combines multiple object tracking and pixel level instance segmentation. I propose Single Shot MOTS, a joint detection, embedding, and segmentation system modeled off the work of Wang, et al. My system produces comparable results to the state of the art baseline system: Track R-CNN. I am able to achieve a runtime speed of 3.2 frames per second, which is a slight improvement on the runtime of Track R-CNN. I conclude that a joint detection, embedding, and segmentation system is capable of achieving comparable results to a two-part system such as TrackR-CNN.

1. Introduction

Multiple object tracking (MOT) is an open problem in the field of computer vision concerning the tracking of multiple moving objects in a video. Multiple object tracking and segmentation (MOTS) extends the MOT problem to also include the task of segmentation. A MOTS system must track moving subjects across frames in a video and annotate the video with pixel-level annotations corresponding to subject identities. Effective solutions to the MOTS problem have applications in autonomous vehicles, surveillance, sports analytics, and population studies. Many of these applications, however, rely on realtime or near realtime solutions.

My goal was to design and implement a near realtime system for pedestrian multiple object tracking and segmentation that matches state of the art accuracy. This paper presents Single Shot MOTS, my solution to the MOTS problem. I build on the work of Wang, et al. [15] in designing a joint detection, embedding, and segmentation system. To implement segmentation, I rework the mask prediction head of Mask R-CNN. This differs from the two-step approach of Voigtlaender, et al. in their baseline TrackR-CNN system [14].

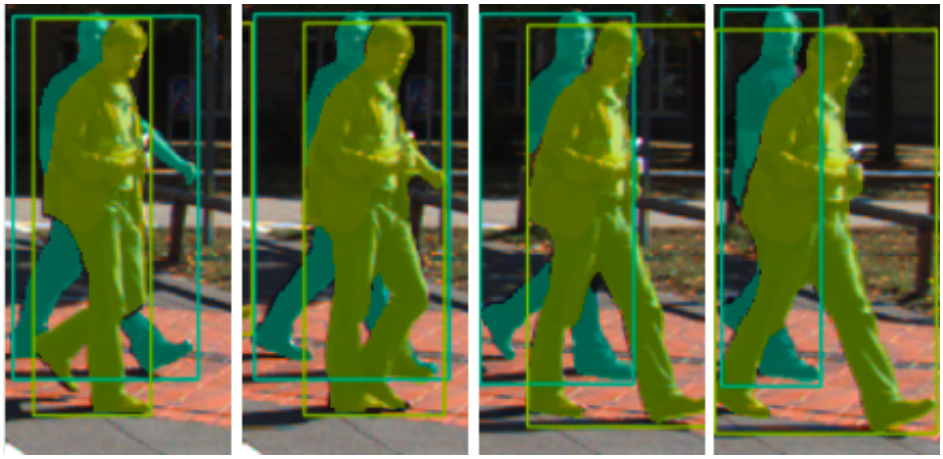

Figure 1: Multiple Object Tracking and Segmentation Example

I was able to achieve comparable results to the baseline TrackR-CNN implementation and even outperform the baseline in critical metrics. Further, my implementation demonstrates considerable speed improvements over the baseline. I was not able, however, to demonstrate near realtime results.

2. Related Work

2.1. Multiple Object Tracking

An effective multiple object tracking and segmentation system must detect objects in a single frame and link detections belonging to the same object across frames. The most common approach to linking objects is to assign each detection an appearance embedding vector. Detections belonging to the same object across frames should have similar embedding vectors. Then, detections can be linked into tracks across frames by applying an optimization algorithm to the embedding vectors, the most common example being the Hungarian algorithm.

Three models exist for finding detections and embeddings [15]. In the "separate detection and embedding model" (SDE), one system comprised of one to many networks is used to find detections across frames and a second network is used to find embeddings for each detection. This approach is also known as tracking-by-detection. In such a model, the novelty lies in how embeddings are generated. The "two part model" is a slightly different approach that still relies on two distinct networks. In the "two part model", one network is used to propose regions of interest (ROIs) [4] in every frame. A separate network is then used to extract embeddings and detections from the ROIs.

The "joint detection and embedding" (JDE) model departs from the previous two in that it only uses one network to produce both embeddings and detections. The advantage of such an approach is a decreased runtime.

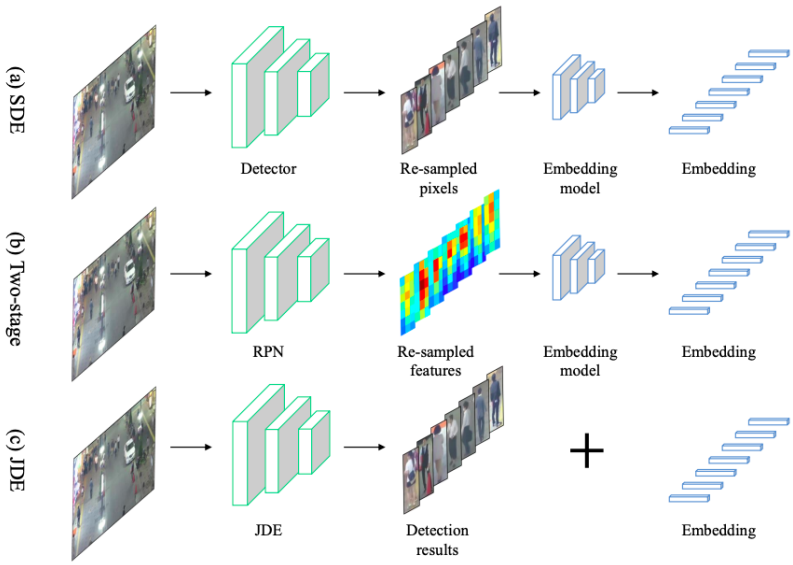

Figure 2: Multiple Object Tracking Approaches

The current state of the art in joint detection and embedding MOT is "Towards Real-Time Multi-Object Tracking" by Wang, et al. Their system uses a Feature Pyramid Network [9] with a DarkNet backbone [11]. The prediction head is made up of a Region Proposal Network [12] used to find bounding boxes. Three individual losses (cross-entropy loss for foreground/background classification, smooth-L1 loss for bounding box regression, and cross-entropy loss for embedding accuracy) are combined in a weighted sum. Weights for this sum are found using the automatic learning scheme of Kendall, et al. [6] The authors demonstrate a system capable of achieving near-best accuracy (64.4% MOTA v.s. 66.1% MOTA on MOT-16 challenge) while running at 18.8 to 24.1 frames per second.

2.2. Multiple Object Tracking and Segmentation

The Multiple Object Tracking and Segmentation problem was established by Voigtlaender, et al. in late 2019 [14]. In their paper, they also lay out a baseline solution to the problem titled

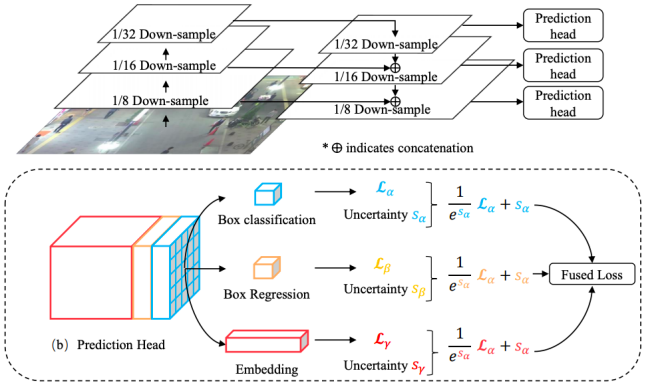

Figure 3: Architecture of System Proposed By Wang, et al.

TrackR-CNN. The solution builds on the earlier Mask R-CNN, which I will discuss first.

Mask R-CNN [5] was introduced by a team of Facebook research in March 2017. It extended the earlier Faster R-CNN, an object detection system [12]. Faster R-CNN used a Region Proposal Network and a convolutional network to produce ROIs in an image. A separate network was used for classification and bounding box regression. Mask R-CNN added to this method a mask classifier, that took as input ROIs and output segmentation masks. The standard approach to single image instance segmentation is currently Mask R-CNN.

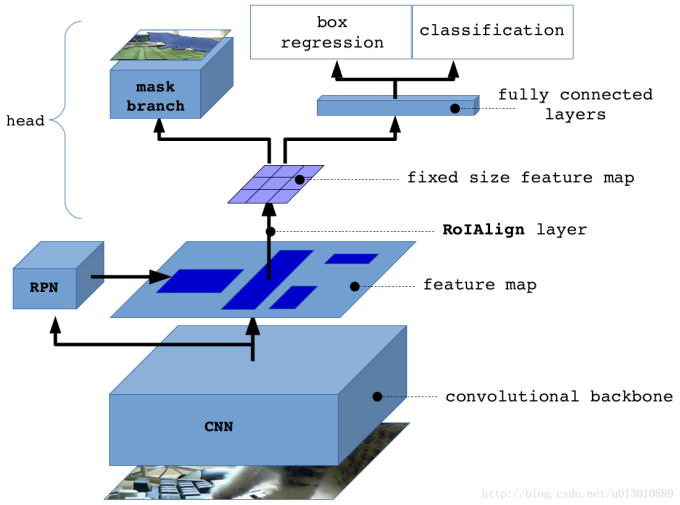

Figure 4: Mask R-CNN Architecture.

TrackR-CNN [14], the proposed baseline solution to the MOTS problem, extends Mask R-CNN to the tracking problem by "3D convolutions to incorporate temporal context and by an association head that produces association vectors for each detection." Because TrackR-CNN builds on Mask

R-CNN, which builds on the two-step design of Faster R-CNN, it is best classified as a two part MOTS system. That is, there is a preliminary network that produces ROIs and a secondary network that outputs embeddings, detections, and segmentations. I hypothesize that such a method is not as fast as a joint detection, embedding, and segmentation MOTS system.

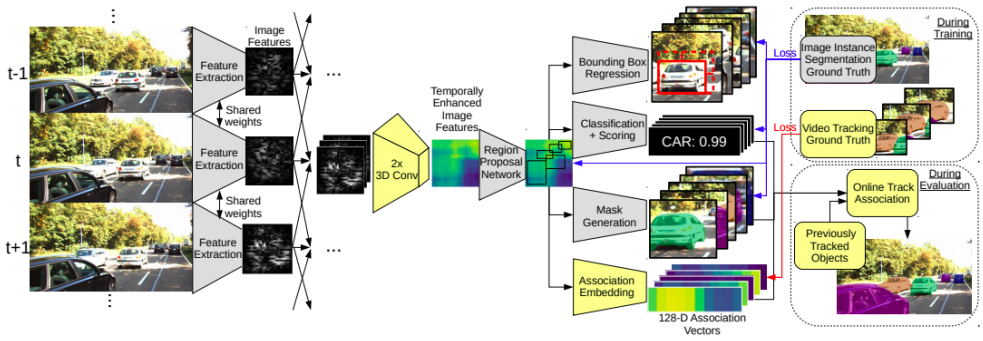

Figure 5: TrackR-CNN Architecture.

3. Approach

I build on the work of Wang, et al. [15] to implement a joint detection, embedding, and segmentation MOTS system. One network, trained with one combined loss, is used to generate detections, embeddings, and segmentations. I hypothesize that such a method is faster than the two-step TrackR-CNN [14] baseline method, but capable of achieving comparable results.

Figure 6: My Approach: Single Shot Multiple Object Tracking and Segmentation.

4. Implementation

I modeled my implementation off of the work of Wang, et al. [15], which is described in detail above.

4.1. CornerNet for Object Detection

The original system of Wang, et al. [15] uses a Region Proposal Network [12] to find suggested bounding box placements, refined using learning based on bounding box regression loss and foreground/background classification loss. During development, I performed an informal ablation study using CornerNet [7] instead of a Region Proposal Network as the network's object detection module. For this ablation study, I did not implement segmentation functionality, opting to test exclusively detection and embedding performance using CornerNet as opposed to a Region Proposal Network (RPN).

CornerNet [7] is a single shot object detector that detects an object as a pair of keypoints. A convolutional network predicts two sets of heatmaps to represent the locations of corners of different object categories, one set for the top-left corners and the other for the bottom-right corners. So that corners can be paired, the network predicts embedding vectors for each corner. These vectors are one dimensional and do not contain any significant information across frames. Rather, they are trained to simply minimize the distance between corners of the same object. "To produce tighter bounding boxes, the network also predicts offsets to slightly adjust the locations of the corners." Given an image, CornerNet produces heatmaps, embeddings, and offsets. A post processing algorithm is used to produce a bounding boxes from this information.

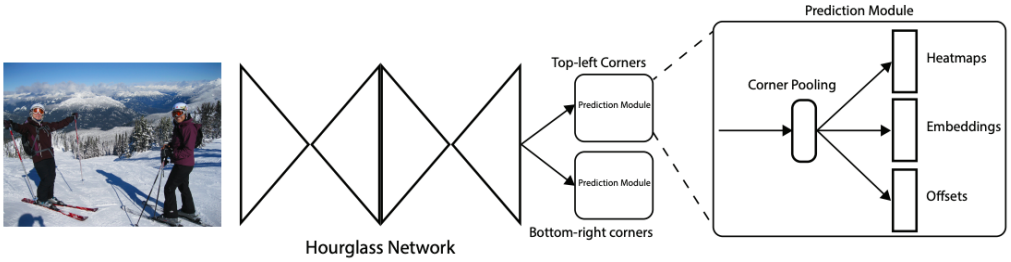

Figure 7: CornerNet Architecture.

CornerNet [7] training loss is composed of four distinct losses: a variant of focal loss to improve corner location prediction, pull and push losses to improve corner pairing, and offset loss to predict corner offsets. In their paper, Law and Deng provide suggested weights to apply to each loss. The RPN model used in the original Wang, et al. paper uses only two losses for bounding box prediction:

cross-entropy loss for foreground/background classification and smooth-L1 loss for bounding box regression. Further, because the Wang, et al. model uses an automatic learning scheme, weights for each loss are not explicitly set. Thus, adapting CornerNet to the multiple object tracking task required some modifications to the Wang, et al. system.

I chose to ignore the suggested weights provided in the CornerNet paper and allow the automated learning scheme to determine their values. I simply replaced the foreground/background and regression losses of the RPN-design with the four losses mentioned in the CornerNet paper. The automated learning scheme was tasked to balance a total of 5 (including the detection embedding loss) losses.

I found that the joint detection and embedding system built with CornerNet performed more poorly at the MOT problem than the base RPN model. I believe that this was caused by CornerNet over-predicting bounding boxes. In turn, the embedding prediction, which relies on accurate foreground objects, was not capable of achieving as accurate embeddings as with the RPN model. There is tremendous room here for future work.

4.2. Segmentation Masks using Artificial ROIs

The key to this MOTS implementation was an effective segmentation mask head. Initially, I used the entirety of Mask R-CNN [5] as a segmentation head. Each detection was passed through the Mask R-CNN architecture and a mask was predicted. The major problem with this approach was its speed. Mask R-CNN is a two-stage system and using it in this manner would have been counter to the goal of building a joint detection, embedding, and segmentation system. Further, Mask R-CNN performs poorly on small, low-resolution images such as a single detection. Thus, even if speed is discounted, such an approach was not effective.

I then experimented with non-deep segmentation approaches including GrabCut [13] and the 2009 system proposed by Lempitsky, et al. "Image Segmentation with a Bounding Box Prior." [8] While these systems performed quickly, they could not match the accuracy of Mask R-CNN in image segmentation. This, however, was not the largest problem. Since MOTS scenes often include

a high amount of occlusion, bounding boxes sometimes overlap and include parts of other subjects. Considering bounding boxes in isolation proved ineffective in such situations as there were areas that belonged to both. Instead, I needed a way to consider the context of the entire scene when segmenting objects.

This ability to consider the entire scene in predicting segmentations is precisely the advantage of Mask R-CNN. To take advantage of this capability, however, I did not need to use the entirety of the Mask R-CNN system, as in my first attempt. Rather, I used my predicted bounding boxes as ROIs and the embedding vectors as ROI feature vectors. The Mask R-CNN prediction head was thus a single element, rather than a two part system.

The Mask R-CNN head architecture, which I used in my implementation, is an extension of two existing Faster R-CNN heads combined with an ROI feature resampling layer known as ROI align. The actual mask prediction head is a convolutional network known as a Feature Pyramidal Network (FPN). I used the same loss formulation (mask loss) as Mask R-CNN. This approach worked extremely well in producing an effective segmentation head to my model.

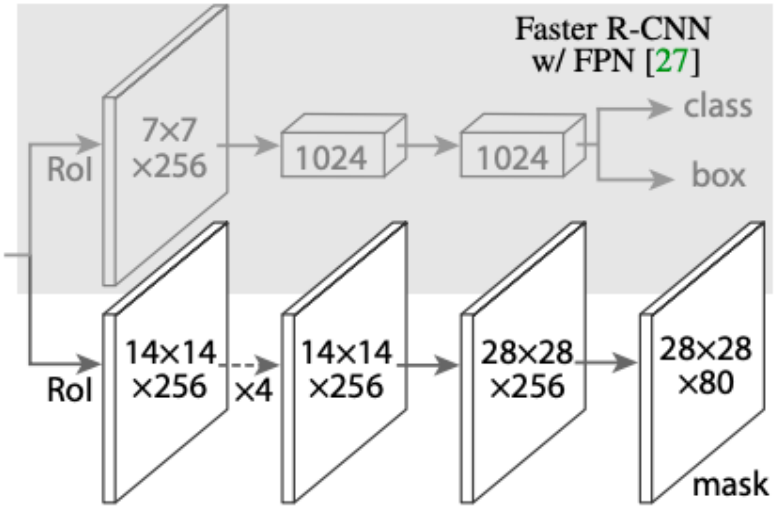

Figure 8: Mask R-CNN Mask Segmentation Head; the Bottom Half is Used as my Segmentation Head.

4.3. Tools

I partly relied on open-source code from existing frameworks in my implementation.

Namely I referenced or utilized parts of:

- The official "Towards Realtime MOT" implementation provided by Wang, et al. (https://github.com/Zhongdao/Towards-Realtime-MOT) [15]
- The official Mask R-CNN implementation (https://github.com/facebookresearch/maskrcnn-benchmark) [5]
- Matterport's Mask R-CNN implementation (https://github.com/matterport/Mask_RCNN) [1]
- The official MOTS evaluation tools (https://github.com/VisualComputingInstitute/mots_tools) [14]
- PyTorch (https://github.com/pytorch/pytorch)
- Open source python packages installed with Anaconda through Anaconda Cloud

5. Data and Evaluation

5.1. Data

Initially, I set out to use the two datasets most frequently used for the MOTS task: the KITTI MOTS dataset [3]a nd the MOTSChallenge dataset [14][10].

KITTI MOTS is an annotated dataset recorded from vehicle mounted cameras. Ultimately, I found that this dataset had too few pedestrians (most frames did not contain any people). For training, this was obviously suboptimal. Testing proved even worse - just one mistake would drastically affect results. Thus, I decided not to use this dataset.

The MOTSChallenge dataset [14] is an adaptation of the earlier MOTChallenge dataset [10]. It features several videos of crowded scenes. Because the test set is only accessible via internal submission for this dataset, I used the training set as a training set and the validation set as a test set.

5.2. MOTSChallenge

Like the previous MOTChallenge, the MOTSChallenge is a benchmark for the task of MOTS. It provides a shared dataset, shared evaluation metrics, and a shared leaderboard. Because the MOTS

Figure 9: Sample Annotated Frame from MOTSChallenge Dataset.

problem is so new, no submissions are yet listed on the MOTSChallenge leaderboard besides the baseline TrackR-CNN implementation. I compete with this baseline implementation as its results as reported through MOTSChallenge.

5.3. MOTS Metrics

In their 2019 paper, Voigtlaender, et al. establish a set of evaluation metrics known as the MOTS metrics [14]. They provide also open source software for evaluating results on the MOTSChallenge dataset using these evaluation metrics. The MOTS metrics are derived from the analogous CLEAR MOT metrics [2]. The evaluation metrics (both CLEAR MOT and MOTS) are as follows:

Measure	Better	Perfect	Description
MOTA	higher	100%	Multiple Object Tracking Accuracy. This measure combines three error sources: false positives, missed targets and identity switches.
MOTSA	higher	100$	Multiple Object Tracking and Segmentation Accuracy. Analogous to MOTA.
MOTP	higher	100%	Multiple Object Tracking Precision. The misalignment between the annotated and the predicted bounding boxes.

MOTSP	higher	100%	Multiple Object Tracking and Segmentation Precision. The misalignment between the annotated and the predicted segmentations.
sMOTSA	higher	100%	Soft MOTS Accuracy. Same as MOTSA but the number of true positives is defined as the sum of IOUs between ground truth and predicted segmentations rather than the number of IOUs over 0.5.
IDF1	higher	100 %	ID F1 Score. The ratio of correctly identified detections over the average number of ground-truth and computed detections.
FAF	lower	0	The average number of false alarms per frame.
MT	higher	100%	Mostly tracked targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at least 80% of their respective life span.
ML	lower	0%	Mostly lost targets. The ratio of ground-truth trajectories that are covered by a track hypothesis for at most 20% of their respective life span.
FP	lower	0	The total number of false positives.
FN	lower	0	The total number of false negatives (missed targets).
ID Sw.	lower	0	The total number of identity switches.

Frag	lower	0	The total number of times a trajectory is fragmented (i.e. interrupted during tracking).
Hz	higher	Inf.	Processing speed (in frames per second excluding the detector) on the benchmark.

6. Results

6.1. Quantitative Results

Quantitative results for my Single Shot MOTS system are as follows:

Sequence	sMOTSA	MOTSA	MOTSP	Recall	Prec	F1	FAR	MT	ML	FP	FN	IDS	Frag
All	40	57.8	73.1	66.1	90.5	76.4	65	21.5	24.1	1863	9109	382	811
1	43	64.3	70.3	71.8	92	80.6	72.9	37.8	16.2	438	1988	86	211
2	26	41.1	72.6	55.1	81.9	65.9	95.6	8.7	28.2	801	2950	116	275
3	40.3	60.1	71.5	69.2	90.8	78.6	63.7	34.6	0	335	1469	103	165
4	48	64	76.7	68.3	95.3	79.5	32.1	27.4	32.3	289	2702	77	160

The following are the results of my method in comparison to TrackR-CNN as reported on the MOTSChallenge leaderboard (bold results indicate better performance):

Method	sMOTSA	MOTSA	MOTSP	Recall	Prec	F1	MT	ML	FP	FN	IDS	Frag
Single Shot MOTS	40	**57.8**	73.1	**66.1**	90.5	**76.4**	21.5	24.1	1863	**9109**	**382**	**811**
TrackR-CNN	**40.6**	55.2	**76.1**	60.8	**94.0**	42.4	**38.7**	**21.6**	**1261**	12641	567	868

Thus, my method achieves comparable results to TrackR-CNN, which is the current state of the art in the MOTS field. My method performs especially well in accuracy categories, having a higher MOTSA score and a lower rate of false negatives. Further, my method minimizes the number of id switches and path fragmentations. Generally, my method performs similarly to TrackR-CNN.

I was able to achieve a runtime of 3.2 frames per second using a Tesla V100 GPU. This is an improvement on TrackR-CNN (which runs at 2.0 frames per second). It does not, however, meet my goal of running in near realtime speed.

6.2. Qualitative Results

Figure 10: Subjects Far From Camera.

The first image sequence demonstrates the accuracy of the system when subjects are in full view and far from the camera. The system produces the tightest masks in this arrangement. The image sequence also demonstrate a fragmented track. The blue subject is not tracked in the third frame, but is in the first, second, and fourth frames. While fragmentations are a common issue, the system is able to preserve identity across fragmentation.

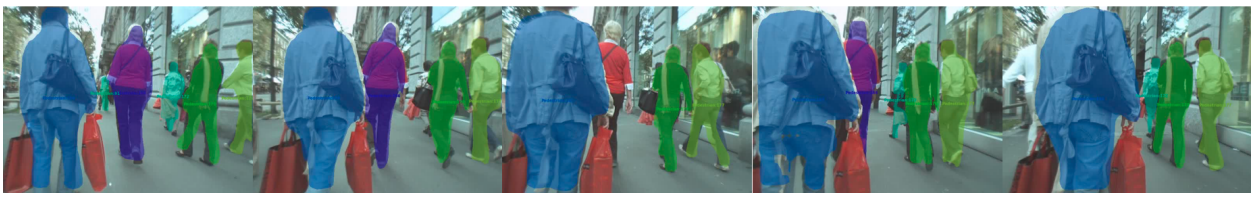

Figure 11: Crowded Scence, Subjects Very Close to Camera.

The second sequence demonstrates a crowded scene where subjects are close to the camera. My system is able to effectively track and segment pedestrians even when the subjects are partly occluded (the blue woman's head and legs are not visible in frames four and five). When a subject

is occluded completely, as is the light blue woman in the far background in frames two and three, my system stops tracking them. When the subject comes back into view, as she does in frame four, my system is able to identify her as a previously seen subject. This series of frames also shows a false negative: the woman in the orange sweater.

Figure 12: Mix of Subjects Relatively Far from the Camera.

This image demonstrates near-perfect functioning of my system in ideal circumstances. The subjects are not close the camera, they are not occluded, and they are in the most common pedestrian pose (i.e. not on a bicycle, in a wheelchair, etc.) A false negative is visible in the last frame as the woman in the white blouse is not tracked. This sequence of frames demonstrates the accuracy of the system of tracking subjects in the far background, such as the two in the store.

Figure 13: Crowded Scence, Subjects Approach Camera.

This sequence of frames is used to further illustrate the effect of distance on segmentation accuracy. As the subjects come closer to the camera, their mask is less accurate than when they are farther from the camera. This is an interesting consequence of a fully convolutional approach to segmentation that does not consider separate body parts, as would, for example, a deformable parts model.

7. Conclusion

I have shown that a joint detection, embedding, and segmentation system is capable of achieving comparable results to a two-part system such as TrackR-CNN. My hypothesis that such a system would lead to near realtime performance, however, was not proven. The MOTS problem is a brand new challenge, and new solutions are being preprinted nearly every week. Future work involves measuring my approach against other proposed methods. As of now, my approach does not incorporate previous segmentation information into present frame segmentations. Because segmentations change little from frame to frame, such functionality is also an important next step.

References

[1] W. Abdulla, "Mask r-cnn for object detection and instance segmentation on keras and tensorflow," https://github.com/matterport/Mask_RCNN, 2017.

[2] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *J. Image Video Process.*, vol. 2008, Jan. 2008. [Online]. Available: https://doi.org/10.1155/2008/246309

[3] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013. [Online]. Available: http://arxiv.org/abs/1311.2524

[5] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, "Mask R-CNN," *CoRR*, vol. abs/1703.06870, 2017. [Online]. Available: http://arxiv.org/abs/1703.06870

[6] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," *CoRR*, vol. abs/1705.07115, 2017. [Online]. Available: http://arxiv.org/abs/1705.07115

[7] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," *CoRR*, vol. abs/1808.01244, 2018. [Online]. Available: http://arxiv.org/abs/1808.01244

[8] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp, "Image segmentation with a bounding box prior," in *2009 IEEE 12th international conference on computer vision*. IEEE, pp. 277–284.

[9] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection," *CoRR*, vol. abs/1612.03144, 2016. [Online]. Available: http://arxiv.org/abs/1612.03144

[10] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, and K. Schindler, "MOT16: A benchmark for multi-object tracking," *arXiv:1603.00831 [cs]*, Mar. 2016, arXiv: 1603.00831. [Online]. Available: http://arxiv.org/abs/1603.00831

[11] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[12] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[13] C. Rother, V. Kolmogorov, and A. Blake, ""grabcut": Interactive foreground extraction using iterated graph cuts," in *ACM SIGGRAPH 2004 Papers*, ser. SIGGRAPH '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 309–314. [Online]. Available: https://doi.org/10.1145/1186562.1015720

[14] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. B. G. Sekar, A. Geiger, and B. Leibe, "MOTS: multi-object tracking and segmentation," *CoRR*, vol. abs/1902.03604, 2019. [Online]. Available: http://arxiv.org/abs/1902.03604

[15] Z. Wang, L. Zheng, Y. Liu, and S. Wang, "Towards real-time multi-object tracking," *arXiv preprint arXiv:1909.12605*, 2019.