Continuous Copy-Paste for One-stage Multi-object Tracking and Segmentation

Zhenbo Xu[1,2,3,4,5*], Ajin Meng[1*], Zhenbo Shi[1], Wei Yang[1†], Zhi Chen[1], Liusheng Huang[1]
[1]University of Science and Technology of China
[2]Hangzhou Innovation Institute, Beihang University, Hangzhou, China
[3]Beihang Univ, Beijing Key Lab Digital Media, Sch Comp Sci & Engn, Beijing, China
[4]Beihang Univ, State Key Lab Virtual Real Technol & Syst, Beijing, China
[5]ShiFang Technology Inc., Hangzhou, China
[†]Corresponding Author. E-mail: qubit@ustc.edu.cn

Abstract

Current one-step multi-object tracking and segmentation (MOTS) methods lag behind recent two-step methods. By separating the instance segmentation stage from the tracking stage, two-step methods can exploit non-video datasets as extra data for training instance segmentation. Moreover, instances belonging to different IDs on different frames, rather than limited numbers of instances in raw consecutive frames, can be gathered to allow more effective hard example mining in the training of trackers. In this paper, we bridge this gap by presenting a novel data augmentation strategy named continuous copy-paste (CCP). Our intuition behind CCP is to fully exploit the pixel-wise annotations provided by MOTS to actively increase the number of instances as well as unique instance IDs in training. Without any modifications to frameworks, current MOTS methods achieve significant performance gains when trained with CCP. Based on CCP, we propose the first effective one-stage online MOTS method named CCPNet, which generates instance masks as well as the tracking results in one shot. Our CCPNet surpasses all state-of-the-art methods by large margins (3.8% higher sMOTSA and 4.1% higher MOTSA for pedestrians on the KITTI MOTS Validation) and ranks 1st on the KITTI MOTS leaderboard. Evaluations across three datasets also demonstrate the effectiveness of both CCP and CCPNet. Our codes are publicly available at: https://github.com/detectRecog/CCP.

1. Introduction

Multi-object tracking (MOT) [30] is of fundamental importance in the field of autonomous driving and video surveillance. Recently, multi-object tracking and segmentation (MOTS) [23] is introduced as a popular extension of bounding box (bbox) based MOT. MOTS provides per-pixel segmentation masks that locate objects more accurately than relative coarse bboxes. As instance masks precisely delineate the visible object boundaries in crowded scenes, MOTS largely eliminates the ambiguities caused by severely overlapped bboxes in both detection and tracking.

Current one-step MOTS approaches struggle to adapt an additional re-ID branch to existing instance segmentation methods to obtain both instance masks and their re-ID features in a single forward pass. Voigtlaender *et al*. [23] build TRCNN upon Mask-RCNN and adopt a fully connected layer to predict association vectors for object proposals. Also built on Mask-RCNN, MOTSNet [16] proposes a novel mask-pooling layer to focus on the foreground segment area rather than bboxes in the association vector extraction. Recently, two-step methods PointTrack [26] and PointTrack++ [27] separate the tracking phase from the instance segmentation phase and convert the compact image representation to un-ordered 2D point cloud representation for learning discriminative instance embeddings. This separation brings two advantages: (i) Extra frame-level instance segmentation datasets can also be used for training [26]; (ii) The training data for tracking is not restricted to consecutive frames, thus allowing more unique instance IDs in training mini-batches [27]. When tracking on the instance segmentation results produced by TRCNN, PointTrack reduces ID switches (IDS) by 55% [26]. To date, the main successes have been primarily in multi-step approaches [12, 26, 27]. The significant gap between one-step methods and multi-step methods suggests that combing tracking with instance segmentation is a non-trivial problem.

In this paper, we posit that two major causes for the performance gap between one-step methods and multi-step methods [26, 27] are: (i) Challenging training data for instance segmentation are limited; (ii) High-quality training samples for tracking are limited. As pixel-level annotations required by MOTS are known to be expensive to obtain,

[*] The first two authors contribute equally to this work.

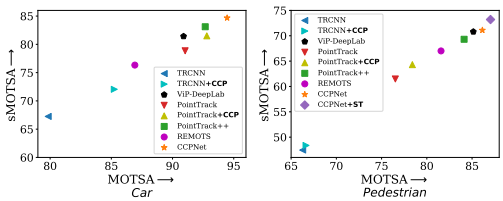

Figure 1. Comparison between our CCPNet and state-of-the-art methods on **KITTI MOTS leaderboard** for cars (Left) and pedestrians (Right). ST denotes self-training (see subsection 4.3).

current MOTS datasets [23, 26] usually have limited frames and instances, especially for non-rigid objects like pedestrians. Crowded scenes where detection failures often occur are even rarer (66% of pedestrians in KITTI MOTS are not adjacent to anyone). Moreover, for KITTI MOTS pedestrians, the probability of two adjacent frames containing valid triplets for training trackers in the training set is only 26.8%. Nevertheless, to make the instance segmentation network and the tracker more robust, more instances with different instance IDs that allow for mining harder samples are desired in training. Therefore, current one-step methods [23, 9, 16] put great efforts to increase the number of instances in training by incorporating more frames (e.g. 8 frames [23]) into a mini-batch and reducing the input image size to save GPU memory accordingly. However, we argue that simply stacking more frames does not solve the problem of lacking high-quality training data for tracking. Assuming a training mini-batch for KITTI MOTS pedestrians contains n adjacent frames, when $n = 2$, the average number of instance IDs is 1.56 and the average number of instances is 3.1. When $n = 8$, though the average number of instances increases to 12.5, the average number of instance IDs (1.70) has not changed dramatically. The scarcity of instance IDs makes it difficult to mine hard triplets, resulting in limited tracking performance.

To bridge the gap between one-step methods and multi-step methods, in this paper, we propose a novel video copy-paste data augmentation strategy named Continuous Copy-Paste (CCP). The intuition behind CCP is to fully exploit the pixel-wise annotations provided by MOTS to actively increase the number of instances and instance IDs in training. To construct a training mini-batch consisting of n frames, CCP first obtains n consecutive frames as templates from real videos or videos forged from a single image. Then, we retrieve several instance blocks from the database built during initialization. Each instance block has n crops belonging to the same inst ID. These crops are extracted from n frames that are close in time but not necessarily adjacent to each other. To mimic the newly emerging instances and the leaving ones, two of n instance blocks will be shifted to the left and right boundary respectively. Other instance blocks will be pasted on their original positions. It is worth noting that we preserve the relative positions of n crops in each instance block for all operations on instance blocks. Lastly, we paste these instance blocks on prepared n templates in descending order of the number of foreground pixels. CCP differs from Copy-Paste proposed in recent works [27, 6] in two aspects. First, we regard the instance block as the basic pasting unit and maintain the relative offset of crops within the instance block. CCP not only increases the instance density but also concentrates on creating high-quality triplets for tracking. Second, we do not model surrounding visual context or randomly choose positions for pasting. Except for instance blocks shifted to image boundaries, instances in instance blocks that are cropped from different images stay in their original positions. As shown in Fig. 1, the effectiveness of CCP is examined by combining it with TRCNN [23] and PointTrack [26].

Based on CCP, we further propose the first effective one-stage method, named CCPNet, for online MOTS. CCPNet follows an encoder-decoder structure and predicts pixel-wise classification confidence, 2D offset pointing to instance centers, clustering parameters, foreground reconstruction, and pixel embeddings. In the post-processing process, pixels are grouped into instances according to their distance from 2D centers. For each grouped instance, we apply max-pooling on embeddings of all foreground pixels to obtain the instance embedding. After that, instances are associated according to the distance between their embeddings and the mask IOU between their masks. Thanks to CCP, though CCPNet operates in an online manner, it beats state-of-the-art methods including both 3D tracking methods and offline tracking methods on multiple datasets. Furthermore, we present a self-training method for CCPNet that brings additional gains (see Fig. 1).

Our main contributions are summarized as follows:

- We propose a novel data augmentation strategy named CCP for training MOTS approaches. CCP brings significant performance gains for current MOTS methods without modifying their frameworks.
- The first effective one-stage MOTS approach termed CCPNet is presented, which performs instance segmentation and tracking in one shot.
- Evaluations across three datasets show that CCPNet outperforms all existing MOTS methods by large margins. Moreover, CCPNet ranks 1st on the KITTI MOTS leaderboard.

2. Related Work

MOTS Recent methods for MOTS can be grouped into two types: one-step methods [23, 16, 19, 20] and multi-

Please check the leaderboard at http://www.cvlibs.net/datasets/kitti/old_eval_mots.php

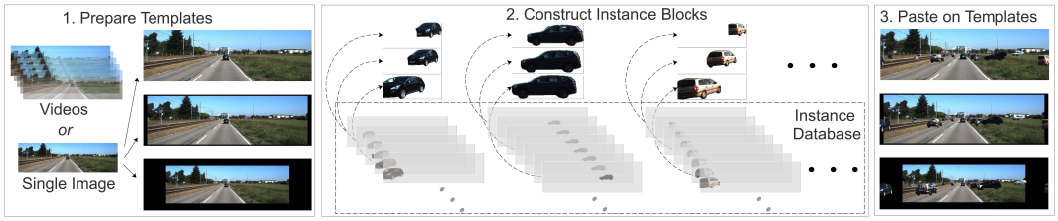

Figure 2. Illustrations of CCP when each training mini-batch contains three frames.

step methods [26, 27, 12, 22]. Pioneering one-step works like TRCNN [23] and MOTSNet [16] modify Mask-RCNN by employing an additional re-ID branch to predict association vectors for object proposals. Besides, different from Mask-RCNN based approaches, the first one-stage online method STE [9] learns instance segmentation with monocular depth estimation and introduces 3D convolutions to learn a spatial-temporal pixel-wise embedding. Pixels whose embeddings are close are grouped into instances and the mean embedding of all belonging foreground pixels is regarded as the embedding of the current instance. More recently, similar to STE, the offline method ViP-DeepLab [19] extends Panoptic-DeepLab [2] by adding a depth prediction head to perform monocular depth estimation and a next-frame instance branch for object association between frames. Unlike one-step methods, multi-step methods break MOTS into multiple stages. PointTrack [26] and PointTrack++ [27] train two separate networks for the tracking phase and the instance segmentation phase respectively. 3D tracking method MOTSFusion [12] builds up short tracklets using 2D optical flow, and then fuses these short tracklets into dynamic 3D object reconstructions. Current multi-step methods achieve more competitive results than one-step methods. Our proposed CCP bridges this gap by actively increasing the number of instances and instance IDs.

Copy-Paste General purposed data augmentation methods like random resizing have been widely used for bbox based vision tasks [7]. Recently, Copy-Paste has been found to be effective for both instance segmentation methods [4] and detection methods [3, 10]. However, copy-paste in the video processing domain has rarely been studied.

3. Continuous Copy-Paste

As shown in Fig. 2, CCP contains three major stages: (i) Prepare templates; (ii) Construct instance blocks from the instance database; (iii) Paste on templates. Our CCP data augmentation strategy can be integrated into the training pipeline of any existing MOTS frameworks because it operates on the training data that are completely decoupled from the training framework. For the sake of brevity, we take cars as an example for illustration in Fig. 2. More details are introduced as below.

0) Initialization. Before following three major stages to obtain a training mini-batch, we need to construct an instance database $\{E_i|i = 1, 2, ...\}$, which is plotted at the bottom of Fig. 2. Each entry $\{E_k = \{S_j|j = 1, 2, ..., L_k\}\}$ in the instance database corresponds to a unique instance ID U_k in the training set. The segmentation item, or say instance, S_j, contains the value of foreground pixels as well as their positions on the 2D image plane at a specific timestamp. The length of the entry L_k equals the length of the track of U_k. It is important to save the segmentation data of instances according to the instance ID for MOTS data augmentation strategies, because we can conveniently enlarge the number of instance IDs in limited training templates by selecting instances belonging to different instance IDs.

1) Prepare Templates. In the first step of preparing a training mini-batch, n frames $\{F_m|m = 1, 2, ..., n\}$ are selected from training videos or are generated based on a single frame. When selected from training videos, we first randomly select $2n + 1$ consecutive frames. After that, n frames are randomly chosen in selected frames. To exploit non-video instance segmentation datasets, CCP also supports generating consecutive frames from a single image. When generated from a single image I, a re-scale ratio seed r_0 and a re-scale step r_δ are randomly generated. Then, we compute the re-scale ratio for n frames as $\{R_m = r_0 + (m - 1) * r_\delta|m = 1, 2, ..., n\}$. Then, n images are generated by re-scaling I. Lastly, we adjust the resolution of n images to the same input resolution by center padding or center cropping. As shown in Fig. 2, By continuously re-scaling individual images, we can generate visually reasonable images F_m.

2) Construct Instance Blocks. We construct t instance blocks $\{(B_i)|i = 1, 2, ..., t\}$, where t is a hyper-parameter that represents how many additional instance IDs will be added in a training mini-batch. An instance block is defined as n crops that are close in time but not necessarily adjacent to each other. As shown in Fig. 2, we randomly select t entries $\{E_i|i = 1, 2, ..., t\}$ from the instance database. Then, for each entry E_k, similar to the frame selection from videos, we randomly select $2n + 1$ consecutive segmentation items. After that, n segmentation items are randomly chosen to construct the instance block $\{B_{E_k} = \{b^1, b^2, ..., b^n\}\}$. We adopt the instance block as the basic unit for copy-paste instead of crops because the

	Cars		Pedestrians	
	wo CCP	w CCP	wo CCP	w CCP
AN of instances	10.79	37.79	4.63	29.35
AN of instances IDs	3.77	12.99	1.58	10.37

Table 1. Training with CCP vs. training without CCP, where $n = 3$ and $t = 15$. AN denotes the average number.

instance block preserves the changes in the position of the same instance ID in a continuous period of time, which is essential for training trackers. For each instance block in Fig. 2, we only plot the minimum enclosing rectangle area of all crops for the sake of clarity.

3) Paste on Templates. Before pasting instance blocks on n prepared templates, we apply three transformations to each instance block: (i) Random replay; (ii) Boundary shift; (iii) Random bottom shearing. Firstly, we randomly replay the instance block by changing $\{b^1, b^2, ..., b^n\}$ to $\{b^n, b^{n-1}, ..., b^1\}$. Secondly, to mimic the newly emerging instances and the leaving instances, we randomly choose two instance blocks and move them to the left and the right boundary, respectively. As shown in Fig. 2, among the three plotted instance blocks, the left instance block is moved to the left boundary and the right instance block is moved to the right boundary. Thirdly, for non-rigid objects like pedestrians, the bottom of instances are often occluded by barriers or other objects. Therefore, we also apply random bottom shearing for non-rigid objects with a small probability. After these transformations, we paste processed t instance blocks on $\{F_m | m = 1, 2, ..., n\}$ in descending order of instance sizes that are measured by the number of pasted pixels.

As shown in Fig. 2, our CCP strategy can actively increase the number of instance IDs and generate visually reasonable images. After CCP, the number of instance IDs increases from 1 to 16 and the number of instances increases from 1 to 48. Table 3 lists the differences on KITTI MOTS between training with CCP and training without CCP.

4. Method

In this section, we present CCPNet, which consists of a shared encoder and four different decoders: (i) Clustering decoder; (ii) Classification decoder; (iii) Reconstruction decoder; (iv) Embedding decoder. The first three decoders have the same network structure and accept the multi-scale features output by the encoder as the input. As shown in Fig. 3, for the embedding decoder, to make it aware of pixel-wise position information, we add an additional position embedding layer similar to [31] to the input. CCPNet extends SpatialEmbedding [15] to jointly perform instance segmentation and online association by adding the embedding decoder to learn pixel-wise embeddings and the reconstruction decoder to reconstruct the input image on the foreground area. At the testing phase, CCPNet predicts instance segmentation results and instance embeddings in one shot. When compared with PointTrack [26] on the same segmentation results, CCPNet achieves better tracking performance (see Table 2 and Table 4). At the training phase, CCPNet accepts multiple frames as input and exploits instances on all input frames for metric learning. For simplicity, we only plot two frames and only consider a single class 'pedestrians'. Instances are tracked online by jointly considering similarities between instance embeddings and the IOU of instance masks in two adjacent frames.

4.1. CCPNet

As shown in Fig. 3, for an input image I^T at timestamp T, five different maps are predicted by four decoders. Here we first introduce the four decoders and then formulate the loss function of CCPNet.

Clustering decoder. Following SpatialEmbedding [15], the clustering decoder predicts a 2-dim offset map (T^u, T^v) as well as a 2-dim sigma map (σ^u, σ^v). For each pixel p_i whose coordinates are (x_i, y_i), $(T^u_{p_i}, T^v_{p_i})$ denotes the 2D offset from (x_i, y_i) to its corresponding instance center on the image plane. $(\sigma^u_{p_i}, \sigma^v_{p_i})$ represents the learned pixel-wise clustering bandwidths. The clustering decoder learns class-agnostic instance clustering parameters that are essential for grouping foreground pixels into instances.

Classification decoder. Assuming there are c classes, the classification decoder predicts $(c+1)$-dim classification map. For simplicity, we only plot two frames and only consider a single class (e.g. pedestrians) in Fig. 3. The classification decoder outputs instance-agnostic semantic segmentation maps that represent foreground pixels to be clustered for each class.

Embedding decoder. The embedding map M contains 32-dim pixel-wise embeddings, which are exploited to construct instance embeddings with respect to all foreground pixels of different instances. Different from STE [9], the learning target of CCPNet is based on triplets of instances rather than instance pixels. We think that it is difficult to force embeddings of pixels belonging to the same instance to be the same, especially for pixels that lie on the boundary of two adjacent instances in crowded scenes. On the contrary, learning on triplets of instances not only eases the difficulty of learning, but also encourages the network to focus on differentiating instances rather than pixels.

Reconstruction decoder. PointTrack [26] proposes that deep 2D/3D convolutional layers inevitably mix up features between adjacent instances in the convolution process and thus making instance embeddings learned for association less discriminative. Therefore, in PointTrack [26], raw image pixels rather than deep convolutional features are used as input, and multiple MLPs rather than convolution layers are used to extract features. Different from PointTrack, we think that the main weakness of current trackers results

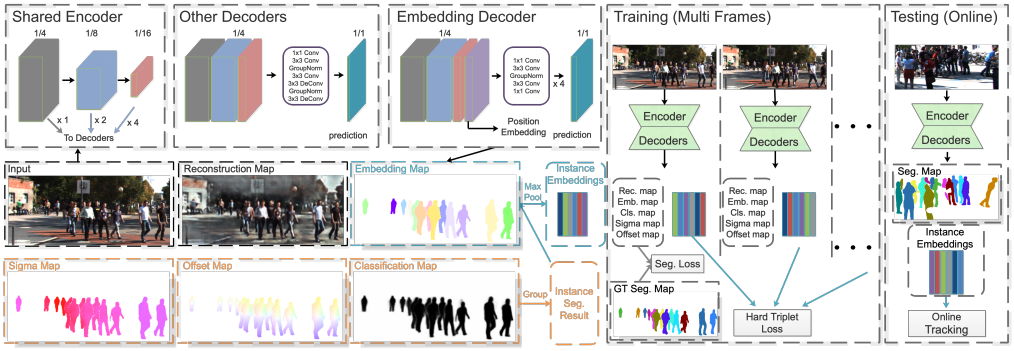

Figure 3. CCPNet consists of a shared encoder and multiple decoders designed for different prediction targets.

from the training in-efficiency rather than 2D/3D convolutions. However, the success of PointTrack reveals that low-level features like color and texture serve as strong clues for discriminating instances. Therefore, to preserve more low-level features in the encoder, we propose to learn foreground pixels reconstruction together with other targets.

Now we would like to describe the loss functions of CCPNet, and before that, we introduce some notations first. Suppose the resolution of I^T is (H, W) and there are K_T instances $\{S_k|k = 1, ..., K_T\}$ on I^T. For each pixel p_i, we denote its coordinate in the image plane as $(x_i, y_i)|x_i \in [0, W-1], y_i \in [0, H-1]$. The predicted center $\mathbb{C}$ of each pixel p_i is formulated as:

$$\mathbb{C}_{p_i} = (\mathbb{C}^x_{p_i}, \mathbb{C}^y_{p_i}) = (T^u_{p_i} + x_i, T^v_{p_i} + y_i) \tag{1}$$

where (T^u, T^v) are predicted by the clustering decoder.

Though each pixel points to its predicted center, it is difficult to force all pixels to point to their centers accurately. To relax the loss for pixels far away from the instance center, we adopt learnable clustering bandwidths σ^u and σ^v [15]. For each instance S_k, the clustering bandwidths are computed as follows:

$$\sigma^u_k = \frac{1}{|S_k|}\sum_{p_i \in S_k} \sigma^u_{p_i}, \sigma^v_k = \frac{1}{|S_k|}\sum_{p_i \in S_k} \sigma^v_{p_i} \tag{2}$$

where $|S_k|$ denotes the number of foreground pixels.

Based on the learned clustering bandwidths, the distance between pixels and the instance center (C^x_k, C^y_k) of S_k is computed by a Gaussian function ϕ:

$$\phi_{p_i} = \exp(-\frac{(\mathbb{C}^x_{p_i} - C^x_k)^2}{2(\sigma^u_{p_i})^2} - \frac{(\mathbb{C}^y_{p_i} - C^y_k)^2}{2(\sigma^v_{p_i})^2}) \tag{3}$$

Besides, to obtain the instance embedding for instance, say S_k, we gather all embeddings of all foreground pixels p of S_k. The shape of the resulting tensor is $|S_k| \times 32$. Then, as shown in Fig. 3, we apply max pooling to the resulting tensor along the first dimension to obtain the final 32-dim instance embedding A_k.

Classification loss L_{cls}. We adopt the Focal loss [11] with Online Hard Example Mining [21] to train the classification decoder. Only 50% of pixels with higher losses are considered in L_{cls}.

$$L_{cls} = \frac{1}{|I^T|}\sum_{p_i \in I^T} \max_{50\%} FL(p_i) \tag{4}$$

where $|I^T|$ is the number of pixels not belonging to 'Dont-Care' [23].

Clustering loss L_{clu}. The clustering loss consists of the sigma loss L_{sigma} and the instance loss. L_{sigma} is exploited to force all foreground pixels belonging to the same instance to have the same sigma value.

$$L_{sigma} = \frac{1}{|S_k|}\sum_{p_i \in S_k} \|\sigma^u_{p_i} - \sigma^u_k\|^2 + \|\sigma^v_{p_i} - \sigma^v_k\|^2 \tag{5}$$

For instance S_k, the instance loss is formulated as the Lovasz-hinge loss [29] between ϕ_k and the ground truth binary mask GT_k. ϕ_k is the segmentation confidence map computed according to Eq. (3). Both the sigma loss and the instance loss are averaged by all instances. Thus, the overall clustering loss is formulated as:

$$L_{clu} = \frac{1}{K_T}\sum_{1 \le k \le K_T} (\beta * L_{sigma} + lovasz(\phi_k, GT_k)) \tag{6}$$

where β is set to 10 by default.

Reconstruction loss L_{rec}. L_{rec} is the averaged L2 loss between I^T and the reconstructed $\hat{I}^T$ for all foreground pixels.

$$L_{rec} = \frac{1}{|p|}\sum_{p \in \cup S_k (1 \le k \le K_T)} (I^T_p - \hat{I}^T_p)^2 \tag{7}$$

where $|p|$ denotes the number of all foreground instance pixels.

Embedding loss L_{emb}. Assuming there are two frames I^T with K_T instances and I^{T-1} with K_{T-1} instances in the training mini-batch, we first gather instance embeddings A for all $K_T + K_{T-1}$ instances. Then, triplets are constructed according to the IDs of these instances. After that, we exploit the batch hard triplet loss [8] for training.

Loss function. The total loss function for CCPNet is formulated as:

$$L = \alpha L_{cls} + L_{clu} + \gamma L_{rec} + L_{emb} \tag{8}$$

where α is set to 1.0 by default and γ is set to 0.1 by default.

4.2. Post-process

In the inference stage, instances are clustered class by class. For each class, the post-processing works as follows. As shown in the three orange bboxes in Fig. 3, we select all foreground pixels by applying the confidence threshold t^{cls} to the classification map. For all foreground pixels, we group instances in two steps in a recursive manner. Firstly, we select the pixel p with the highest classification confidence and obtain its predicted instance center $\mathbb{C}_p$. Secondly, the distances between the predicted centers of all foreground pixels and $\mathbb{C}_p$ are computed according to Eq. (3). A pixel whose distance is below the distance threshold t^{dist} is considered to belong to the instance to be grouped. After these pixels are grouped to a new instance, pixels belonging to this instance are considered as background. All instances are grouped recursively by applying these two steps. After each instance mask is obtained, we aggregate the embeddings of all foreground pixels on the embedding map M. Inspired by PointNet [17], the max-pooling operation is used to obtain the final fixed-length instance embedding. Online instance association is performed by jointly considering the Euclidean distance between cross-frame instances and mask IOU between instances in adjacent frames. Given instance segment $S_i^{t_0}$ at time t_0 and instance segment $S_j^{t_1}$ at time t_1 as well as their instance embeddings A_i and A_j, respectively, the similarity SI is computed as follows:

$$SI = -D(A_i, A_j) + IOU(S_i^{t_0}, S_j^{t_1}) * (\|t_1 - t_0\| == 1) \tag{9}$$

where D represents the Euclidean distance and mask IOU is considered only for instances belonging to adjacent frames.

Two instances are associated if and only if SI is higher than a threshold t^{sim}. We set a default alive threshold t^{a_0} to all tracks. If a track does not update in t^{a_0} frames, it will be terminated. Moreover, for instances that are close to the boundary, we assign a much smaller alive threshold t^{a_1}. It is worth noting that our post-processing does not involve sophisticated tracking strategies (e.g. Kalman filter (KF) [24]), because it is out of the scope of this paper. If considered, we believe CCPNet can achieve higher tracking performances.

	Cars		Pedestrians	
	sMOTSA	MOTSA	sMOTSA	MOTSA
STE [9]	46.1	61.3	-	-
TRCNN [23]	76.2	87.8	46.8	65.1
MOTSNet [16]	78.1	87.2	54.6	69.3
MOTSFusion [12]	85.5	94.6	58.9	71.9
PointTrack [26]	85.5	94.9	62.4	77.3
PointTrack++ [27]	86.81	95.95	65.51	81.54
CCPNet	**87.36**	**96.23**	**69.35**	**85.69**
CCPNet (wo Rec.)	86.85	95.74	68.35	84.55
CCPNet+PointTrack	87.07	96.00	69.29	84.97

Table 2. Results on KITTI MOTS Validation. Rec. denotes the reconstruction decoder.

4.3. Self-training

We further present a self-training strategy that can be combined with CCP to allow CCPNet to learn on unlabelled data. After each training epoch, we exploit CCPNet to detect and track instances in raw data. The pseudo labels provided by CCPNet are considered as the ground-truth MOTS annotations. It is worth noting that, to alleviate the classification ambiguities in instance boundaries. We set pixels whose classification confidence is between $t^{cls}/2$ and t^{cls} to 'DontCare' [23], which means these pixels are not considered in the classification loss. In each training epoch, we exploit the pseudo-labeled raw data as templates with slightly more instance blocks pasted. The results on KITTI MOTS testset (see Table 3) validates that self-training helps CCPNet achieve higher performances.

5. Experiments

In this section, we first present our results on KITTI MOTS [23], APOLLO MOTS [23], as well as MOTS20 [23]. As our CCP can be integrated into the training pipeline of many MOTS frameworks, we also combine it with TRCNN [23] and PointTrack [27] respectively to examine its effectiveness. Then, we show the ablation study on CCP.

Metric. Following previous works [9, 23, 16, 12, 26, 27, 28, 19], we concentrate on sMOTSA and MOTSA. We do not focus on ID switches (IDS) because it varies with the instance segmentation results. For example, in some cases, more false negatives lead to fewer IDS. We compare IDS only under the same segmentation results. Besides, the main metric for the KITTI MOTS is updated from sMOTSA to HOTA [13] at the end of February 2021. HOTA is a higher-order metric for MOT and focuses more on the tracking performance. As previous methods do not provide the results on the new HOTA metric, we provide the main results with HOTA on the KITTI MOTS leaderboard (see Table 3) rather than KITTI MOTS Validation.

We use their open-sourced code with minor modifications to the data loader to make them work with CCP.

HOTA is updated at the submission of this paper. The original Evaluation server can be found at http://www.cvlibs.net/datasets/kitti/old_eval_mots.php

	Cars			Pedestrians		
	sMOTSA	MOTSA	HOTA	sMOTSA	MOTSA	HOTA
TRCNN [23]	67.00	79.60	56.63	47.30	66.10	41.93
TRCNN**+CCP**	72.05	85.24	57.89	48.37	66.65	43.92
MOTSNet [16]	71.00	81.70	-	48.70	62.00	-
MOTSFusion [12]	75.00	84.10	73.63	58.70	72.90	54.04
PointTrack [26]	78.50	90.90	61.95	61.50	76.50	54.44
PointTrack**+CCP**	81.52	92.81	67.94	64.34	78.41	58.44
PointTrack++ [27]	82.80	92.60	67.28	68.10	83.60	56.67
REMOTS [28]	75.92	86.74	71.61	65.97	81.33	58.81
ViP-DeepLab [19]	81.03	90.74	**76.38**	68.76	84.52	**64.31**
CCPNet	**84.47**	**94.40**	73.61	70.16	85.85	60.50
CCPNet+Self-Train	**84.47**	94.36	75.12	**70.55**	**86.36**	62.22

Table 3. Results on KITTI MOTS Leaderboard.

Experimental Setup. For KITTI MOTS, following PointTrack [26], we pre-train CCPNet on the combination of the KINS dataset [18] and KITTI MOTS. Images from the KINS dataset are exploited as an additional source of templates. The pre-training of CCPNet takes 30 epochs at a learning rate of $5 \cdot 10^{-4}$. For APOLLO MOTS as well as MOTS20, we train CCPNet from scratch using CCP without additional training data. The training of CCPNet takes 15 epochs at a learning rate of $2 \cdot 10^{-4}$. It is worth noting that, similar to recent Copy-Paste [6], training with CCP is much faster than training without CCP due to much higher instance densities.

For CCP, we adopt $n = 3$ frames by default. For video frames, each frame will be regarded as an isolated image at a probability of 0.2 and is used to generate n frames by re-scaling this frame. We add at most 15 instance blocks (comparisons are available in Table 6) for KITTI MOTS and APOLLO MOTS, respectively. As the input image of MOTS20 is four times bigger than KITTI MOTS, we add at most 25 instance blocks for MOTS20. The number of instance blocks added to each frame is randomly chosen between 0 and the maximum number. Moreover, when we select segmentation items for each instance block, n_b is set to 7 by default. The probability of random replay is 0.5 and the probability of random bottom shearing is 0.2. It is worth noting that, based on the observation of the characteristics of KITTI MOTS and MOTS20, we apply the random bottom shearing and the boundary shift only to relatively bigger instances.

For CCPNet, the input resolution of KITTI MOTS and APOLLO MOTS is 1248×384. As to MOTS20, the input resolution is 1088×1280. Due to the much larger input resolution, we train CCPNet at half-precision. t^{cls} is set to 0.4 and t^{dist} is set to 0.41 for both pedestrians and cars. t^{sim} is 7.5 by default. Besides, t^{a_0} and t^{a_1} are set to 8 and 3 respectively. Moreover, we exploit the raw KITTI data [5] for self-training. On KITTI MOTS, CCPNet processes images at a speed of 7 FPS when test on a single 2080Ti card.

5.1. Main Results

We compare recent works on MOTS: STE [9], TRCNN [23], MOTSNet [16], MOTSFusion [12], PointTrack [26], PointTrack++ [27], REMOTS [28], and ViP-DeepLab [19].

Results on KITTI MOTS Validation. The results on KITTI MOTS Validation are summarized in Table 2. For cars, CCPNet only achieves a small improvement of 0.5% on sMOTSA. However, for non-rigid instances like pedestrians, CCPNet surpasses PointTrack++ by nearly 4% on sMOTSA. The large improvements on pedestrians not only demonstrate the effectiveness of CCP, but also show that there is still much room for improvement in the MOTS performance for pedestrians. Moreover, without the reconstruction decoder, MOTSA that is closely related to the tracking performance decreases by 1% for pedestrians. Note that adding the reconstruction decoder does not impact the speed of CCPNet, as it is abandoned in the inference stage. Qualitative results are shown in Fig. 4.

Results on KITTI MOTS Leaderboard. We present the main results in Table 3. When trained with the self-training strategy, CCPNet achieves the highest sMOTSA score on the leaderboard. The demo videos are available in the supplementary material. Moreover, it is worth noting that combing CCP with current MOTS methods brings significant performance improvements without any modifications to their frameworks. It brings TRCNN a gain of 5.0 % on sMOTSA for cars and PointTrack a gain of over 3% on sMOTSA for both cars and pedestrians. For the very recently updated metric HOTA, we are slightly behind the offline tracking method ViP-Deeplab. We believe that a more sophisticated tracking strategy can bridge this gap. However, it is out of the scope of this paper.

Results on APOLLO MOTS validation. We follow PointTrack [26] to validate the effectiveness of CCPNet on APOLLO MOTS. CCPNet surpasses PointTrack by 2.5% on both sMOTSA and MOTSA. Besides, when examined on the same segmentation results produced by CCPNet, CCP-

Please check the leaderboard at http://www.cvlibs.net/datasets/kitti/old_eval_mots.php

Figure 4. **Quantitative results of CCPNet.** Instances of the same track id are plotted in the same color.

	Seg.	sMOTSA	MOTSA
DeepSort [25]	TRCNN	45.71	57.06
TRCNN [23]	TRCNN	49.84	61.19
DeepSort	PointTrack	64.69	73.97
PointTrack [26]	PointTrack	70.76	80.05
PointTrack	CCPNet	72.89	82.22
CCPNet	CCPNet	**73.20**	**82.53**

Table 4. Results on APOLLO MOTS validation.

		sMOTSA	MOTSA
Private Detection	TRCNN [23]	40.6	55.2
	UBVision	52.8	67.4
	SORTS [1]	55.0	68.3
	CCPNet	**59.3**	**75.5**
Public Detection	SORTS_ReID	55.8	69.1
	REMOTS [28]	70.4	84.4

Table 5. Results on MOTS20 Leaderboard.

	Cars		Pedestrians	
	sMOTSA	MOTSA	sMOTSA	MOTSA
n=2 (MA=15)	87.33	96.18	**69.35**	**85.69**
n=3 (MA=15)	**87.36**	**96.23**	68.76	85.45
wo BS	87.23	96.09	68.65	84.94
MA=5	86.77	95.42	67.98	84.55
MA=10	87.26	96.09	68.39	84.79
MA=25	86.72	95.59	67.49	83.63

Table 6. Ablation study on CCP. BS denotes boundary shift.

Net obtains 0.3% higher MOTSA and the IDS decreases by 15%. When adopting PointTrack [26] to track on the segmentation results of CCPNet (see the last row of Table 2 and the second last row of Table 4), CCPNet shows higher tracking performances.

Results on MOTS20 Leaderboard. MOTS20, which is built on MOT16 [14], is a very challenging MOTS dataset with many crowded scenarios. With strong precomputed public detection results generated by models that are pre-trained by large datasets, methods in the 5th BMTT MOTChallenge Workshop like REMOTS [28] achieve very high performances. By comparison, our CCPNet is trained from scratch on MOTS20 with around 2000 images as the training set. For a fair comparison, we compare CCPNet with methods using private detections. As shown in Table 5, our CCPNet achieves state-of-the-art performances on the MOTS20 leaderboard with much higher sMOTSA and MOTSA.

5.2. Ablation Study

Impact of input frames of CCP. We only test $n = 2$ and $n = 3$ due to the GPU memory limitation. As shown in the first two rows of Table. 6, training with $n = 3$ shows better MOTS performances for cars. However, the gap between $n = 2$ and $n = 3$ is small, which shows that the number of frames has little effect on the effectiveness of CCP.

Impact of border shift of CCP. To validate the effectiveness of border shift, we fix the parameters of the encoder and the decoders except for the embedding decoder to make sure that the segmentation results will not change. Then, we train CCPNet under the condition of border shift as well as removing border shift. As shown in the third row of Table. 6, though the MOTSA only decreases negligibly, the IDS decreases from 24 to 17 for cars and decreases from 23 to 18 for pedestrians.

Impact of the maximum number of instance blocks of CCP. We examine the impact of the maximum number of instance blocks on KITTI MOTS. As shown in the last three rows in Table 6, the performance gain is notable when the maximum number increase from 5 to 10. When the maximum number increases to 25, the performance starts to decrease. Though adding more instances brings more crowded scenes, it makes the domain deviation so significant that harms the performance on the validation set.

6. Conclusion

In this paper, we proposed an effective data augmentation method named CCP to help MOTS methods achieve higher performances without any modifications to their frameworks. CCP can exploit non-video frames into MOTS training and continuously copy-paste instances to continuous frames to increase both the number of instances and the number of unique instance IDs. When trained with CCP, current methods achieve great performance gains. Moreover, we put forward the first effective one-stage MOTS method named CCPNet that accomplishes instance segmentation and instance association in a single forward pass. Evaluations across three datasets demonstrate that our CCPNet achieves state-of-the-art results, outperforming previous methods by large margins.

Acknowledgment

This work was supported by the Anhui Initiative in Quantum Information Technologies (No. AHY150300).

References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468. IEEE, 2016. 8

[2] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12475–12485, 2020. 3

[3] Debidatta Dwibedi, Ishan Misra, and Martial Hebert. Cut, paste and learn: Surprisingly easy synthesis for instance detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1301–1310, 2017. 3

[4] Hao-Shu Fang, Jianhua Sun, Runzhong Wang, Minghao Gou, Yong-Lu Li, and Cewu Lu. Instaboost: Boosting instance segmentation via probability map guided copy-pasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 682–691, 2019. 3

[5] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 7

[6] Golnaz Ghiasi, Yin Cui, Aravind Srinivas, Rui Qian, Tsung-Yi Lin, Ekin D. Cubuk, Quoc V. Le, and Barret Zoph. Simple copy-paste is a strong data augmentation method for instance segmentation. *arXiv preprint arXiv:2012.07177*, 2020. 2, 7

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 3

[8] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 6

[9] Anthony Hu, Alex Kendall, and Roberto Cipolla. Learning a spatio-temporal embedding for video instance segmentation. *arXiv preprint arXiv:1912.08969*, 2019. 2, 3, 4, 6, 7

[10] Mate Kisantal, Zbigniew Wojna, Jakub Murawski, Jacek Naruniec, and Kyunghyun Cho. Augmentation for small object detection. In *9th International Conference on Advances in Computing and Information Technology (ACITY 2019)*, pages 119–133. Aircc Publishing Corporation, Dec. 2019. 3

[11] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 5

[12] J. Luiten, T. Fischer, and B. Leibe. Track to reconstruct and reconstruct to track. *IEEE Robotics and Automation Letters*, 5(2):1803–1810, 2020. 1, 3, 6, 7

[13] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixe, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578, Feb 2021. 6

[14] 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. 8

[15] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8837–8845, 2019. 4, 5

[16] Lorenzo Porzi, Markus Hofinger, Idoia Ruiz, Joan Serrat, Samuel Rota Bulò, and Peter Kontschieder. Learning multi-object tracking and segmentation from automatic annotations. *arXiv preprint arXiv:1912.02096*, 2019. 1, 2, 3, 6, 7

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 6

[18] Lu Qi, Li Jiang, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Amodal instance segmentation with kins dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2019. 7

[19] Siyuan Qiao, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. ViP-Deeplab: Learning visual perception with depth-aware video panoptic segmentation. *arXiv preprint arXiv:2012.05258*, 2020. 2, 3, 6, 7

[20] Sarthak Sharma, Junaid Ahmed Ansari, J Krishna Murthy, and K Madhava Krishna. Beyond pixels: Leveraging geometry and shape cues for online multi-object tracking. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3508–3515. IEEE, 2018. 2

[21] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training Region-Based Object Detectors with Online Hard Example Mining. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 761–769, Las Vegas, NV, USA, June 2016. IEEE. 5

[22] Young-min Song and Moongu Jeon. Online multi-object tracking and segmentation with gmphd filter and simple affinity fusion. *arXiv preprint arXiv:2009.00100*, 2020. 3

[23] Paul Voigtlaender, Michael Krause, Aljosa Osep, Jonathon Luiten, Berin Balachandar Gnana Sekar, Andreas Geiger, and Bastian Leibe. Mots: Multi-object tracking and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7942–7951, 2019. 1, 2, 3, 5, 6, 7, 8

[24] Zhongdao Wang, Liang Zheng, Yixuan Liu, and Shengjin Wang. Towards real-time multi-object tracking. *arXiv preprint arXiv:1909.12605*, 2(3):4, 2019. 6

[25] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017. 8

[26] Zhenbo Xu, Wei Zhang, Xiao Tan, Wei Yang, Huan Huang, Shilei Wen, Errui Ding, and Liusheng Huang. Segment as points for efficient online multi-object tracking and segmentation. In *European Conference on Computer Vision*, pages 264–281. Springer, 2020. 1, 2, 3, 4, 6, 7, 8

[27] Zhenbo Xu, Wei Zhang, Xiao Tan, Wei Yang, Xiangbo Su, Yuchen Yuan, Hongwu Zhang, Shilei Wen, Errui Ding, and Liusheng Huang. Pointtrack++ for effective online multi-object tracking and segmentation. *arXiv preprint arXiv:2007.01549*, 2020. 1, 2, 3, 6, 7

[28] Fan Yang, Xin Chang, Chenyu Dang, Ziqiang Zheng, Sakriani Sakti, Satoshi Nakamura, and Yang Wu. Remots: Refining multi-object tracking and segmentation. *arXiv preprint arXiv:2007.03200*, 2020. 6, 7, 8

[29] Jiaqian Yu and Matthew Blaschko. Learning submodular losses with the lovász hinge. In *International Conference on Machine Learning*, pages 1623–1631, 2015. 5

[30] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *arXiv preprint arXiv:2004.01888*, 2020. 1

[31] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020. 4