Multi-Modal Continual Test-Time Adaptation for 3D Semantic Segmentation Supplementary Material

Haozhi Cao[1*] Yuecong Xu[2*] Jianfei Yang[1†] Pengyu Yin[1] Shenghai Yuan[1] Lihua Xie[1]
[1]Nanyang Technological University [2]Institute for Infocomm Research, A*STAR, Singapore
{haozhi002, xuyu0014, yang0478, pengyu001, syuan003}@e.ntu.edu.sg elhxie@ntu.edu.sg

In this appendix, we provide more details about our proposed CoMAC. ***Firstly****, we present more details about our proposed benchmarks, including the generation of synthetic point clouds of Synthia [10] and class mapping for both benchmarks.* ***Secondly****, we introduce the implementation details of the baselines we compared with CoMAC.* ***Thirdly****, the implementation details of CoMAC are included and we provide additional ablation studies to justify our implementation.* ***Fourthly****, we present more visualization comparisons between CoMAC and previous SOTA methods to show that our CoMAC can outperform prior works during the continual adaptation.* ***Last but not least****, we thoroughly compare our CoMAC with previous works to justify the novelty of our work.*

1. Benchmark Details

In Sec. 4.1, we propose two new benchmarks for the exploration of MM-CTTA, including SemanticKITTI-to-Synthia (S-to-S) and SemanticKITTI-to-Waymo (S-to-W). Unlike conventional UDA benchmarks for image semantic segmentation which usually treat simulation datasets (in this case, Synthia [10]) as the source domain, we regard Synthia as the target domain for S-to-S since it contains more various environmental conditions compared to most real-life datasets [1, 3, 12], which enables us to construct a more challenging MM-CTTA benchmark. Although there exist some real-life datasets recorded under adverse conditions [16, 11], they usually contain single-modal information and therefore are not applicable under our settings.

Point cloud generation for Synthia. For S-to-S, we generate synthetic 3D point clouds from depth images as mentioned in Sec. 4.1. More specifically, the 3D point clouds are extracted by mimicking the point clouds from a 64-line LiDAR with a vertical field-of-view (FOV) of 30°. Given the 2D depth images, we first project the 2D depth pixels back to the 3D world based on the official intrinsic matrix. Subsequently, we limit the vertical FOV of pixels by the range $[-15°, 15°]$ and simulate the characteristic of LiDAR channels by vertically selecting 64 lines of pixels that evenly divide the FOV. After filtering out the points located too far ($>$ 100m), the simulated 3D points are then randomly sampled from each line, leading to a total of 20k points for each frame. As shown in Fig. 1, compared to the previous work MMTTA [12] that randomly samples pixels without considering the structural characteristic of points, our simulated point clouds are more similar to those collected by real-world LiDARs.

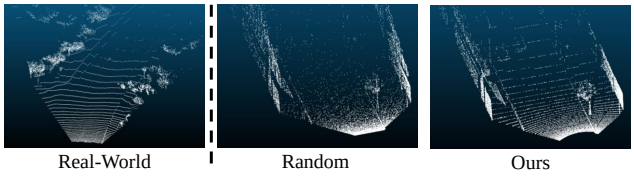

Figure 1: Comparison of the real-world point cloud, randomly sampled depth pixels as point clouds in MMTTA [12] and ours.

Class mapping. To address the label misalignment between the source-domain and target-domain datasets, we utilize a similar label mapping strategy as [5] to form an 11-class mapping for each benchmark. Most of the classes are identical to [5] as shown in Table. 1 and Table. 2. For SemanticKITTI-to-Synthia, we ignore the class "sky" of Synthia since it is neither included in SemanticKITTI nor a common class in 3D segmentation. For SemanticKITTI-to-Waymo, we introduce the new class "trunk" which is not included in [5] considering its important role in other downstream tasks (e.g., localization [4] and map construction [7, 15]). Both mappings form a segmentation task of 11 classes.

2. Details of pre-trained models and baselines

Here we present implementation details of the pre-training stage and baseline methods in our work. For all methods, we conduct a search for learning rates and report the best results.

*Equal contributions.
†Corresponding author.

S-to-S Class	SemanticKITTI classes	Synthia classes
car	car, moving-car, truck, moving-truck	car
bike	bicycle, motorcycle, bicyclist, motorcyclist, moving-bicyclist, moving-motorcyclist	bicycle
person	person, moving-person	pedestrian
road	road, lane-marking, parking	road, lanemarking
sidewalk	sidewalk	sidewalk
building	building	building
nature	vegetation, trunk, terrain	vegetation
poles	pole	pole
fence	fence	fence
traffic-sign	traffic-sign	traffic-sign
other-objects	other-object	traffic-light

Table 1: Class mapping of the benchmark SemanticKITTI-to-Synthia.

S-to-W Class	SemanticKITTI classes	Waymo classes
car	car, moving-car, truck, moving-truck	car, truck, other-vehicle, bus
bike	bicycle, motorcycle, bicyclist, motorcyclist, moving-bicyclist, moving-motorcyclist	bicycle, motorcycle, bicyclist, motorcyclist
person	person, moving-person	pedestrian
road	road, lane-marking, parking	road, lane-marker
sidewalk	sidewalk	sidewalk, curb, walkable
building	building	building
nature	vegetation, terrain	vegetation
poles	pole	pole
trunk	trunk	tree-trunk
traffic-sign	traffic-sign	sign
other-objects	other-object	other-ground, construction-cone, traffic-light

Table 2: Class mapping of the benchmark SemanticKITTI-to-Waymo.

Pre-training on SemanticKITTI. Similar to previous works [5, 12], we follow the official code of [5] to pre-train both 2D and 3D networks on SemanticKITTI [1]. Specifically, the optimizers for both networks are Adam [6] with $\beta = 0.9$ and $\beta = 0.999$, while the base learning rate is set to 0.001 with a batch size of 8 for both networks. The voxel scale and full scale of SCN are set to 20 and 4096, respectively. The total pre-training epoch is set to 100,000, where the learning rate shrinks ten-fold at the epochs 8,000 and 9,000. For 2D augmentations, we utilize a bottom crop of the size $(480, 302)$ followed by a random horizontal flip with a probability of 0.5 as well as color jittering of $(0.4, 0.4, 0.4)$. On the other hand, 3D augmentations during the pre-training stage include a noisy rotation of 0.1, a random flip of the y-axis with a probability of 0.5, a random z-rotation within the range $[0, 360^\circ]$, and a random translation within the receptive field of SCN. The pre-trained models are then utilized as the initial state for all methods.

TTA methods. For TTA methods, we compared our CoMAC with self-training with pseudo-labels (Pslabel), TENT [13], and LAME [2]. Specifically, hard thresholding is introduced in Pslabel to mitigate the side-effect of noisy pseudo-labels, where pseudo-labels with confidence scores less than the median score of their predicted class are filtered out. For TENT [13], the implementation follows the official code. For LAME [2], we implement the variant with the linear affinity matrix instead since either kNN or rbf affinity is too expensive to compute for segmentation.

MM-TTA methods. xMUDA [5] with pseudo-labels (xMUDA-pl) and MMTTA [12] are selected as representatives of MM-TTA methods. Specifically, xMUDA-pl utilizes the same pseudo-label filtering strategy as Pslabel, and we utilize the cross-modal consistency learning of xMUDA in a test-time manner. For MMTTA, we self-implement the proposed method since it does not come with any official code. As we mentioned in Sec. 4.2, we additionally introduce an MM-CTTA variant of MMTTA simply by integrating MMTTA with the model-based stochastical restoration. Specifically, the restoring rate is set to 0.01 for MMTTA-rs for both benchmarks.

CTTA methods. In this work, CoTTA [14] and EATA [8] are the reason works that focus on the CTTA problem. Both CoTTA and EATA follow the official implementations. For CoTTA which is also based on test-time augmentations, we apply the same z-rotation as 3D augmentations for the adaptation of the 3D backbone. For EATA, we adopt the first sequence of each benchmark to train the Fisher regularizer.

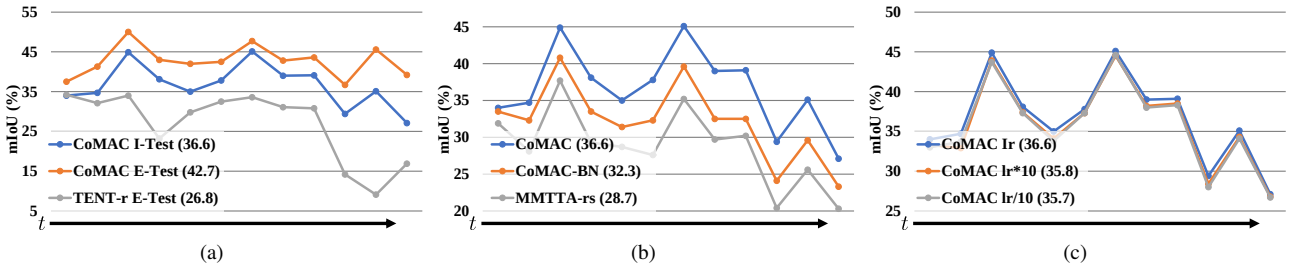

Figure 2: Additional ablation studies to justify the effectiveness of CoMAC. Specifically, Fig. 2a illustrates the performance comparison between testing immediately (I-Test) and testing at the end of each sequence (E-Test), while Fig. 2b compares CoMAC with its variant which only update the parameters of batch normalization layers. Fig. 2c presents the robustness of CoMAC toward different learning rates.

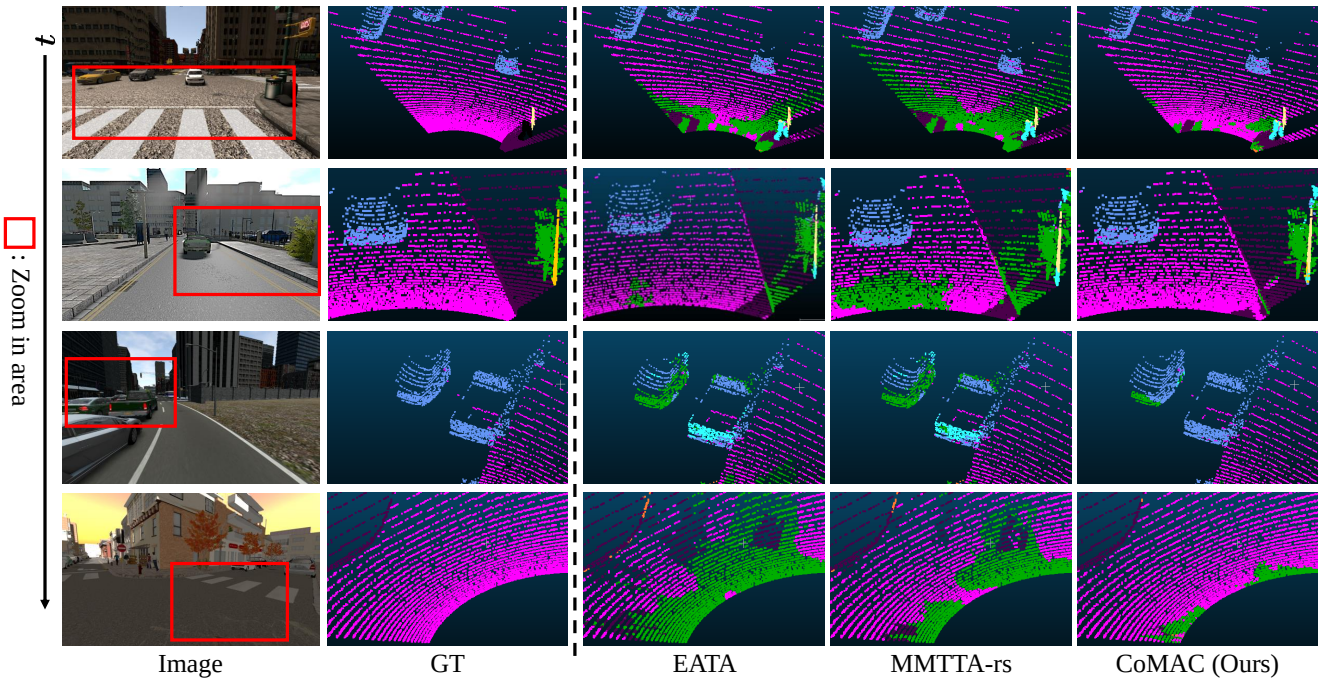

Figure 3: Visualization comparison between EATA [8], MMTTA-rs [12], and CoMAC (ours). Sequences from the top to the bottom are from 02-Summer, 04-Fall, 05-Winter, and 04-Sunset, respectively.

3. Implementation Details of CoMAC

As mentioned in Sec. 3.4, the source class-wise centroids are collected as source knowledge. The source centroids are computed as the average of normalized features of all source training samples based on their labels. During the test-time adaptation, the batch normalization layers utilize the batch statistics of the current iteration for predictions. The optimizers of CoMAC for both networks are identical to the pre-training stage, while the learning rate is set to 1.25E-05. The batch size is set to 1 and the performance is based on the immediate testing results (I-Test) after the encounter of the data, strictly following the one-pass protocol as in previous works [14, 12]. Here CoMAC updates all parameters of both networks. All our experiments are conducted on an NVIDIA RTX 3090. Our code is implemented based on PyTorch [9] of version 1.8.1 with CUDA 11.4.

To justify the effectiveness of CoMAC under different evaluation protocols and settings, we additionally conduct experiments where CoMAC is evaluated at the end of each sequence. Specifically, as shown in Fig. 2a, testing at the end of each sequence (E-Test) results in a noticeable improvement of 6.1% compared to the I-Test version of CoMAC. Here we also present an E-Test version of TENT with auto-reset at the beginning of each sequence (in short, TENT-r E-Test) as a fair comparison, while both I-Test and E-Test versions of CoMAC surpass TENT-r E-Test by more

Method	Publication	Task	Designs
TENT [13]	ICLR-21	Test-Time Adaptation (TTA): inaccessible source data, a **stationary** target domain with the input of **a single modality**, adaptation conducted during the testing process.	(i) TENT proposes to update the parameters of batch normalization layers by minimizing the prediction entropy in the target domain; (ii) TENT highlights **the first TTA protocol** for future exploration.
MMTTA [12]	CVPR-22	Multi-Modal Test-Time Adaptation (MM-TTA): inaccessible source data, a **stationary** target domain with the input of **multiple modalities**, adaptation conducted during the testing process.	(i) MMTTA generates reliable intra-modal pseudo labels by combining predictions from student and teacher models. (ii) The inter-modal pseudo labels are generated as the weighted sum of intra-modal predictions based on **student-teacher consistency**.
CoTTA [14]	CVPR-22	Continual Test-Time Adaptation (CTTA): inaccessible source data, a **continually changing** target domain with the input of **a single modality**, adaptation conducted during the testing process.	(i) CoTTA estimates the reliability of teacher predictions through confidence score, where **hard-selected** unreliable predictions are substituted by augmentation-average for noise suppression. (ii) Parameters of networks are stochastically restored to avoid catastrophic forgetting.
CoMAC (Ours)	-	Multi-Modal Continual Test-Time Adaptation (MM-CTTA): inaccessible source data, a **continually changing** target domain with the input of **multiple modalities**, adaptation conducted during the testing process.	(i) CoMAC utilizes iMPA as **a centroid-based adaptive fusion** between augmentation-average and raw predictions to mitigate the potential inductive bias. (ii) CoMAC **attends to reliable modality** through xMPF for cross-modal noise suppression. (iii) The centroids are actively updated for adaptation while **stochastically revisiting pseudo-source features** as source knowledge to avoid forgetting.

Table 3: Comparison between our CoMAC and previous methods.

than 9.8%, which justifies the effectiveness of CoMAC. Fig. 2b presents the performance of the CoMAC variant which only updates the parameters of batch normalization layers, outperforming MMTTA-rs which also only updates batch normalization layers by about 3.6%. As for the learning rate, Fig. 2c justifies the robustness of CoMAC toward learning rates, falling into a relative margin of 2.4%.

4. Visualization Comparison

To intuitively justify the effectiveness of our CoMAC, we present more visualization results of our CoMAC in comparison with previous SOTA methods, including EATA [8] and MMTTA-rs [12]. Specifically, we present the visualization comparison from different sequences as illustrated in Fig. 3, where our CoMAC generates more accurate predictions compared to both EATA and MMTTA-rs.

5. Comparison with Previous (C)TTA and MM-TTA Methods

In this work, we propose CoMAC to tackle MM-CTTA for 3D semantic segmentation as an extension of CTTA. To highlight our novelty, we compare our proposed CoMAC with previous TTA (TENT [13]), CTTA (CoTTA [14]), and MM-TTA (MMTTA [12]) methods. Specifically, we list out their specific tasks and designs as shown in Table. 3.

References

[1] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9297–9307, 2019. 1, 2

[2] Malik Boudiaf, Romain Mueller, Ismail Ben Ayed, and Luca Bertinetto. Parameter-free online test-time adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8344–8353, 2022. 2

[3] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020. 1

[4] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguere, Jens Behley, and Cyrill Stachniss. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537. IEEE, 2019. 1

[5] Maximilian Jaritz, Tuan-Hung Vu, Raoul de Charette, Emilie Wirbel, and Patrick Perez. xmuda: Cross-modal unsupervised domain adaptation for 3d semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2020. 1, 2

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[7] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2017. 1

[8] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International Conference on Machine Learning*, pages 16888–16905. PMLR, 2022. 2, 3, 4

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 3

[10] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016. 1

[11] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Acdc: The adverse conditions dataset with correspondences for semantic driving scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10765–10775, 2021. 1

[12] Inkyu Shin, Yi-Hsuan Tsai, Bingbing Zhuang, Samuel Schulter, Buyu Liu, Sparsh Garg, In So Kweon, and Kuk-Jin Yoon. Mm-tta: multi-modal test-time adaptation for 3d semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16928–16937, 2022. 1, 2, 3, 4

[13] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021. 2, 4

[14] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022. 2, 3, 4

[15] Zejie Wang, Zhen Zhao, Zhao Jin, Zhengping Che, Jian Tang, Chaomin Shen, and Yaxin Peng. Multi-stage fusion for multi-class 3d lidar detection. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 3113–3121, 2021. 1

[16] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1