Semantics, Distortion, and Style Matter: Towards Source-free UDA for Panoramic Segmentation
–Supplementary Material–

Xu Zheng[1] Pengyuan Zhou[3] Athanasios V. Vasilakos[4] Lin Wang[1,2*]
[1]AI Thrust, HKUST(GZ) [2]Dept. of CSE, HKUST [3] Aarhus University [4] University of Agder
zhengxu128@gmail.com, pengyuan.zhou@ece.au.dk, th.vasilakos@gmail.com, linwang@ust.hk

Project Page: https://vlislab22.github.io/360SFUDA/

Abstract

Due to the lack of space in the main paper, we provide more details of the proposed method and experimental results in the supplementary material. Specifically, Section 1 includes the Algorithm of the proposed SFUDA framework, providing a step-by-step description of the method. Section 2 presents implementation details, offering insights into the specific techniques used in the implementation. In Section 4, additional quantitative and qualitative experimental results are presented, along with further details of the ablation study. Lastly, Section 5 provides the code implementation of crucial modules.

1. Algorithm

Algorithm 1: Framework of our proposed method.

```
Input: Pre-trained source model: $F_S$;
Unlabeled ERP image $x_T$; FFP images $x^f_T$; TP images $x^t_T$
Output: Target models $F_T$
1  Initialize $F_T$ randomly;
2  for each epoch do
3      for each iteration do
4          Sample a batch of target data $x_T$ from $D_T$ and the corresponding FFP and TP images $x^f_T$ and $x^t_T$;
5          Forward passes of $F_S$ (Eq. 1
6          Project Pseudo labels of the predictions of $F_S$
7          Calculate prototypes $\tau_p, \tau_f$ for adaptation (Eq. 2)
8          Update global panoramic prototype (Eq. 3) and transfer knowledge at prediction level (Eq. 5) and prototype level (Eq. 4)
9          Fine tune $F_S$ (Eq. 6)
10         Align $FandF'$ with BNS from $F_S$ (Eq. 7)
11         Calculate spatial- and channel-wise attention maps (Eq. 8 and Eq. 9)
12         Transfer feature level knowledge (Eq. 10)
13         Update parameters of $F_S$ and $F_T$.
14     end
15 end
16 Final target models $F_T$.
```

2. Implementation Details

Tangent Projection

In the domain of panoramic image processing, the *tangent projection* is a pivotal technique for the transformation of an Equirectangular Projection (ERP) image into a planar two-dimensional tangent image. This method is instrumental in rendering a rectilinear representation of a designated sector of a panoramic vista. The computational expenditure of this projection primarily stems from the requisite transformations and calculations needed to transmute the ERP image into its tangent counterparts. The procedural steps for executing this conversion are methodically outlined as follows:

1. Identification and selection of a pivotal point on the ERP image, corresponding to the central perspective of the targeted panorama segment.
2. Computation of spherical coordinates to ascertain the precise spatial positioning of this point on the sphere.
3. Derivation of the normal vector at the selected spherical point, thereby establishing the orthogonality to the tangent plane, and subsequent selection of two orthogonal vectors that delineate the basis vectors of the tangent plane.
4. Calculation of the spherical coordinates for each pixel within the tangent image to determine their respective

*Corresponding author.

Method	SF	mIoU	Road	S.W.	Build.	Wall	Fence	Pole	Tr.L.	Tr.S.	Veget.	Terr.	Sky	Pers.	Car	Δ
PVT [13] SSL	✗	38.74	55.39	36.87	80.84	19.72	15.18	8.04	5.39	2.17	72.91	32.01	90.81	26.76	57.40	-
PVT [13] MPA	✗	40.90	70.78	42.47	82.13	22.79	10.74	13.54	1.27	0.30	71.15	33.03	89.69	29.07	64.73	-
Trans4PASS [22] SSL	✗	43.17	73.72	43.31	79.88	19.29	16.07	20.02	8.83	1.72	67.84	31.06	86.05	44.77	68.58	-
Trans4PASS [22] MPA	✗	45.29	67.28	43.48	83.18	22.02	21.98	22.72	7.86	1.52	73.12	40.65	91.36	42.69	70.87	-
DATR-M [24] Source	✗	32.37	62.48	17.00	74.55	10.66	7.17	11.48	3.37	0.38	60.21	17.56	81.99	26.38	47.55	-
DATR-T [24] Source	✗	33.82	61.33	16.77	75.44	14.64	7.65	15.11	4.10	1.30	64.06	15.25	82.72	26.13	55.21	-
DATR-S [24] Source	✗	35.15	60.43	13.57	76.69	18.35	5.88	17.33	3.44	2.62	62.68	19.54	83.58	34.30	58.56	-
DATR-M [24] MPA	✗	48.24	77.05	46.43	83.80	25.16	35.25	26.20	19.12	12.54	77.93	23.79	94.23	38.04	67.59	-
DATR-T [24] MPA	✗	52.11	78.40	52.10	85.04	31.52	42.44	30.11	22.78	15.01	77.50	27.96	93.76	47.08	73.80	-
DATR-S [24] MPA	✗	52.76	78.33	52.70	85.15	30.69	42.59	32.19	24.20	17.90	77.72	27.24	93.86	47.98	75.34	-
DATR-M [24] CFA	✗	51.04	77.62	49.03	84.58	28.15	39.70	30.34	23.83	15.95	78.23	24.74	93.73	44.14	74.45	-
DATR-T [24] CFA	✗	53.23	79.09	52.92	85.51	32.02	42.90	31.56	27.17	17.14	77.87	28.71	93.72	48.16	75.26	-
DATR-S [24] CFA	✗	54.05	79.07	52.28	85.98	33.38	45.02	34.47	26.15	18.27	78.21	26.99	94.02	51.21	77.62	-
Source w/ seg-b1	✓	35.81	63.36	24.09	80.13	**15.68**	13.39	16.26	7.42	0.09	62.45	20.20	86.05	23.02	53.37	-
SFDA w/ seg-b1 [7]	✓	38.21	68.78	30.71	80.37	5.26	18.95	20.90	5.25	2.36	70.19	23.30	90.20	22.55	57.90	+2.40
ProDA w/ seg-b1 [23]	✓	37.37	68.93	30.88	80.07	4.17	18.60	19.72	1.77	1.56	70.05	22.73	**90.60**	19.71	57.04	+2.73
GTA w/ seg-b1 [6]	✓	36.00	64.61	20.04	79.04	8.06	15.36	19.86	6.02	2.13	65.77	17.75	84.56	26.71	58.13	+0.19
HCL w/ seg-b1 [4]	✓	38.38	68.82	30.41	80.37	5.88	20.18	20.10	4.23	2.11	70.50	24.74	89.89	22.65	59.04	+2.57
DATC w/ seg-b1 [15]	✓	38.54	69.48	26.96	80.68	11.64	15.24	20.10	**9.33**	0.55	66.11	24.31	85.16	30.90	60.58	+2.73
Simt w/ seg-b1 [2]	✓	37.94	68.47	29.51	79.62	6.78	19.20	19.48	2.31	1.33	68.85	26.55	89.30	22.35	59.49	+2.13
Ours w/ seg-b1	✓	41.78	**70.17**	**33.24**	**81.66**	13.06	23.40	23.37	7.63	3.59	71.04	25.46	89.33	36.60	64.60	+5.97
Ours w/ seg-b2	✓	**42.18**	69.99	32.28	81.34	10.62	**24.35**	**24.29**	9.19	**3.63**	**71.28**	**30.04**	88.75	**37.49**	**65.05**	**+6.37**

Table 1. Experimental results on the S-to-D scenario, the overlapped 13 classes of two datasets are used to test the UDA performance. The **bold** and underline denote the best and the second-best performance in source-free UDA methods, respectively.

loci on the sphere.

5. Projection of each pixel's position vector onto the tangent plane via the dot product with the basis vectors, culminating in the acquisition of the pixel's bidimensional coordinates.
6. Implementation of a color interpolation technique for each pixel in the ERP image to faithfully reproduce the corresponding chromatic elements in the tangent image.

It is imperative to highlight that the computational load induced by the tangent projection is not substantial and falls well within the processing ambit of modern computational apparatus. For a more profound understanding and detailed examination of the tangent projection methodology, the reviewers are encouraged to consult the code implementation provided in the supplementary material.

Datasets. The Cityscapes dataset [1] is a widely recognized real-world benchmark collected to advance autonomous driving research. It encompasses urban street scenes from various cities, annotated with pixel-level precision across 19 semantic categories. The standard partitioning of this dataset designates 2975 images for training purposes and 500 images for validation. In the context of our study, we employ the designated official training set as the source domain to derive the base model for our experiments.

Conversely, the DensePASS dataset [9] provides a panoramic perspective, encompassing 2500 annotated panoramas that encapsulate a wide array of global street scenes. The annotations are meticulously crafted, focusing on categories that are pivotal for navigation. We utilize DensePASS as the target domain for testing our models.

Additionally, the SynPASS dataset [22] contributes a synthetic collection of 9080 panoramic images, annotated across 22 categories. It is structured into official splits for training, validation, and testing, comprising 5700, 1690, and 1690 images, respectively. Our methodology leverages the intersection of 13 categories shared between SynPASS and DensePASS for both training and evaluative purposes.

Overall, our experimental framework is rigorously tested under two distinct paradigms: real-world transfer (Cityscapes-to-DensePASS, C-to-D) and synthetic-to-real domain adaptation (SynPASS-to-DensePASS, S-to-D). The evaluation on the DensePASS validation set is conducted using a single-scale inference at a resolution of 400×2048 pixels. We adopt the mean Intersection-over-Union (mIoU) as the quantitative metric for performance evaluation in both the source and target domains.

Method	SF	mIoU	Road	S.W.	Build.	Wall	Fence	Pole	Light	Sign	Vegt.	Terr.	Sky	Person	Rider	Car	Truck	Bus	Train	Motor	Bike
ERFNet [10]	✗	5.55	63.59	18.22	47.01	9.45	12.79	17.00	8.12	6.41	34.24	10.15	18.43	4.96	2.31	46.03	3.19	0.59	0.00	8.30	5.55
PASS [16]	✗	23.66	67.84	28.75	59.69	19.96	29.41	8.26	4.54	8.07	64.96	13.75	33.50	12.87	3.17	48.26	2.17	0.82	0.29	23.76	19.46
ECANet [17] (O.s.)	✗	43.02	81.60	19.46	81.00	32.02	39.47	25.54	3.85	17.38	79.01	39.75	94.60	46.39	12.98	81.96	49.25	28.29	0.00	55.36	29.47
CLAN [8] (Adv.)	✗	31.46	65.39	21.14	69.10	17.29	25.49	11.17	2.14	7.61	71.03	28.19	55.55	18.86	2.76	71.60	26.42	17.99	59.53	9.44	15.91
CRST [26] (S.t.)	✗	31.67	68.18	15.72	76.78	14.06	26.11	9.90	0.82	2.66	69.36	21.95	80.06	9.71	1.25	65.12	38.76	27.22	48.85	7.10	18.08
P2PDA [19]	✗	41.99	70.21	30.24	78.44	26.72	28.44	14.02	11.67	5.79	68.54	38.20	85.97	28.14	0.00	70.36	60.49	38.90	77.80	39.85	24.02
SIM [14] (S.t.)	✗	44.58	68.16	32.59	80.58	25.68	31.38	23.60	19.39	14.09	72.65	26.41	87.88	41.74	16.09	73.56	47.08	42.81	56.35	47.72	39.30
PCS [18]	✗	53.83	78.10	46.24	86.24	30.33	45.78	34.04	22.74	13.00	79.98	33.07	93.44	47.69	22.53	79.20	61.59	67.09	83.26	58.68	39.80
DAFormer [3]	✗	54.67	73.75	27.34	86.35	35.88	45.56	36.28	25.53	10.65	79.87	41.64	94.74	49.69	25.15	77.70	63.06	65.61	86.68	65.12	48.13
Trans4PASS-T [21]	✗	53.18	78.13	41.19	85.93	29.88	37.02	32.54	21.59	18.94	78.67	45.20	93.88	48.54	16.91	79.58	65.33	55.76	84.63	59.05	37.61
Trans4PASS-S [21]	✗	55.22	78.38	41.58	86.48	31.54	45.54	33.92	22.96	18.27	79.40	41.07	93.82	48.85	23.36	81.02	67.31	69.53	86.13	60.85	39.09
DPPASS-T [25]	✗	55.30	78.74	46.29	87.47	48.62	40.47	35.38	24.97	17.39	79.23	40.85	93.49	52.09	29.40	79.19	58.73	47.24	86.48	66.60	38.11
DPPASS-S [25]	✗	56.28	78.99	48.14	87.63	42.12	44.85	34.95	27.38	19.21	78.55	43.08	92.83	55.99	29.10	80.95	61.42	55.68	79.70	70.42	38.40
DATR-M [24]	✗	52.90	78.71	48.43	86.92	34.92	43.90	33.43	22.39	17.15	78.55	28.38	93.72	52.08	13.24	77.92	56.73	59.53	93.98	51.12	34.06
DATR-T [24]	✗	54.60	79.43	49.70	87.39	37.91	44.85	35.06	25.16	19.33	78.73	25.75	93.60	53.52	20.20	78.07	60.43	55.82	91.11	67.03	34.32
DATR-S [24]	✗	56.81	80.63	51.77	87.80	44.94	43.73	37.23	25.66	21.00	78.61	26.68	93.77	54.62	29.50	80.03	67.35	63.75	87.67	67.57	37.10
USSS [5] (IDD)	✗	26.98	68.85	5.41	67.39	15.10	21.79	13.18	0.12	7.73	70.27	8.84	85.53	22.05	1.71	58.69	16.41	12.01	0.00	23.58	13.90
USSS [5] (Map.)	✗	30.87	71.01	31.85	76.79	12.13	23.61	11.93	3.23	10.15	73.11	31.24	89.59	16.05	3.86	65.27	24.46	18.72	0.00	9.08	14.48
Seamless [11] (Map.)	✗	34.14	59.26	24.48	77.35	12.82	30.91	12.63	15.89	17.73	75.61	33.30	87.30	19.69	4.59	63.94	25.81	57.16	0.00	11.59	19.04
SwiftNet [10] (City.)	✗	25.67	50.73	32.76	70.24	12.63	24.02	18.79	7.18	4.01	64.93	23.70	84.29	14.91	0.97	43.46	8.92	0.04	4.45	12.77	8.77
SwiftNet [20] (M.3)	✗	32.04	68.31	38.59	81.48	15.65	23.91	20.74	5.95	0.00	70.64	25.09	90.93	32.66	0.00	66.91	42.30	5.97	0.07	6.85	12.66
Ours w/ seg-b1	✓	**48.78**	72.90	48.10	82.77	22.19	39.69	26.66	17.90	14.35	74.98	25.95	88.95	45.36	15.83	75.70	49.16	55.68	82.07	54.82	33.76
Ours w/ seg-b2	✓	**50.12**	72.29	43.04	84.48	29.72	37.68	22.83	9.52	14.45	75.26	34.53	91.12	49.92	27.22	76.22	47.81	64.13	79.47	56.83	35.76

Table 2. Per-class results on Cityscapes-to-DensePASS. Comparison with the state-of-the-art panoramic segmentation [16, 17], domain daptation [8, 19, 26], and multi-supervision methods [5, 11, 20]. (Abbreviation: O.s.: Omni-sup; Adv.: Adversarial; S.t.: Self-training; Map.: Mapillary; City.: Cityscapes; M.3.: Merge3.)

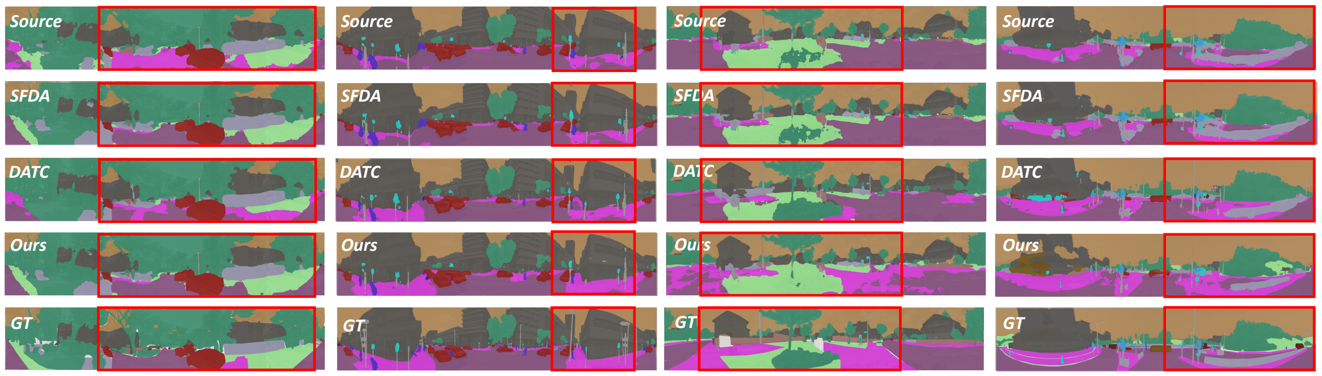

Figure 1. Example visualization results on Cityscapes-to-DensePASS. (a) source, (b) SFDA [7], (c) DATC [15], (d) Ours, (e) Ground Truth.

3. Experiments and Analysis

As the first SFUDA method for panoramic image segmentation, there is no prior method for direct comparison. We thus empirically validate our method by comparing with the existing UDA and panoramic segmentation methods on two widely used benchmarks.

Implementation Our framework is implemented using PyTorch and trained on multiple NVIDIA GPUs. Both the source and target models in our framework are based on the efficient SegFormer architecture. We provide two implementations: **Seg-b1**, which is based on the **SegFormerB1** architecture, and **Seg-b2**, which is based on the **SegFormerB2** architecture. The encoder is initialized with ImageNet-1K pre-trained weights, while the segmentation head (decoder) is randomly initialized. Both models use a batch size of 4 and are trained with the AdamW optimizer for 50,000 iterations. We set the initial learning rate to 0.00006 and employ a polynomial learning rate schedule with a power factor of 1.0. The input size for the ERP path model is 400 pixels in

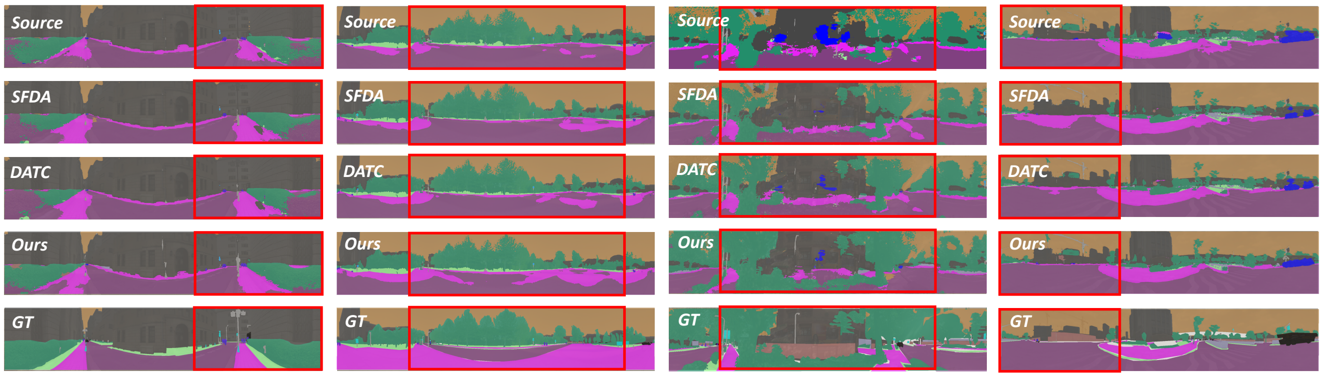

Figure 2. Example visualization results on SynPASS-to-DensePASS. (a) source, (b) SFDA [7], (c) DATC [15], (d) Ours, (e) Ground Truth.

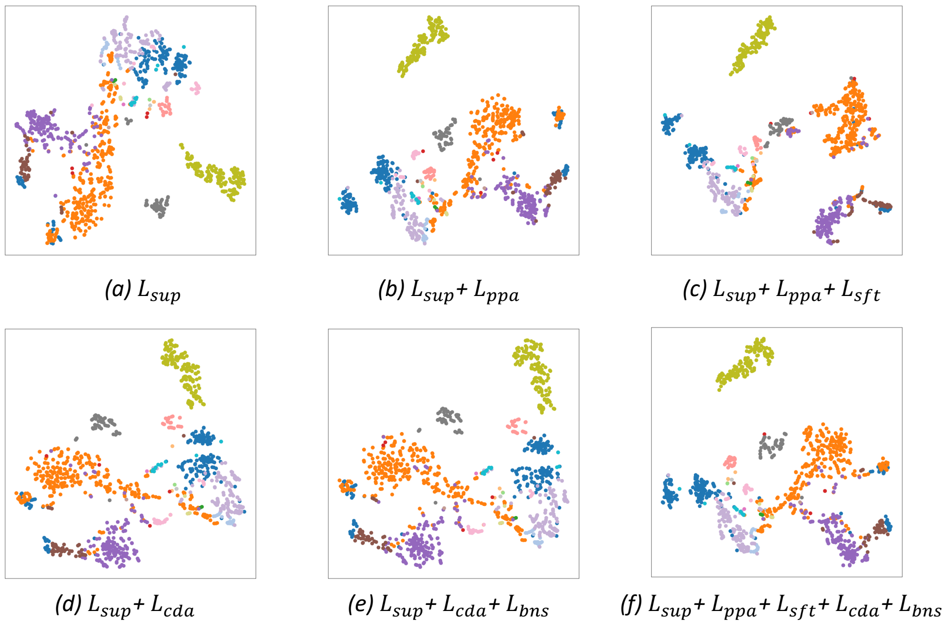

Figure 3. TSNE visualization of different loss function combinations. (a)$\mathcal{L}_{sup}$, (b)$\mathcal{L}_{sup} + \mathcal{L}_{ppa}$, (c) $\mathcal{L}_{sup} + \mathcal{L}_{ppa} + \mathcal{L}_{sft}$, (d)$\mathcal{L}_{sup} + \mathcal{L}_{cda}$, (e)$\mathcal{L}_{sup} + \mathcal{L}_{cda} + \mathcal{L}_{bns}$, (e)$\mathcal{L}_{sup} + \mathcal{L}_{ppa} + \mathcal{L}_{sft} + \mathcal{L}_{cda} + \mathcal{L}_{bns}$.

height and 2048 pixels in width, while for the tangent path model, it is 224 pixels by 224 pixels.

4. Additional Experimental Results

Due to the space limitation, we select the most crucial categories which are related to autonomous driving in Table 1 of the main paper. The detailed per-category results of our methods are listed in Tab. 2.

Fig. 1 and Fig. 2 present additional qualitative results on the DensePASS dataset, demonstrating that our method achieves superior segmentation performance.

As discussed in the main paper, the SAM model performs remarkably well on pinhole images; however, there is still ample room for improvement when it comes to panoramic images with distortion. In Figure 4, we showcase results obtained by applying SAM to panoramic images, which reveal certain shortcomings in areas with structural distortion, particularly around the north and south poles, as well as the equator, where the semantic information becomes complex. Our method holds the potential to adapt SAM to the panoramic image domain, and we leave this as an area for future investigation.

To assess the effectiveness of the proposed modules, we conducted ablation experiments in both real-world and synthetic-to-real scenarios, exploring various combinations of loss functions. The results demonstrate that all of the proposed modules and loss functions positively contribute to

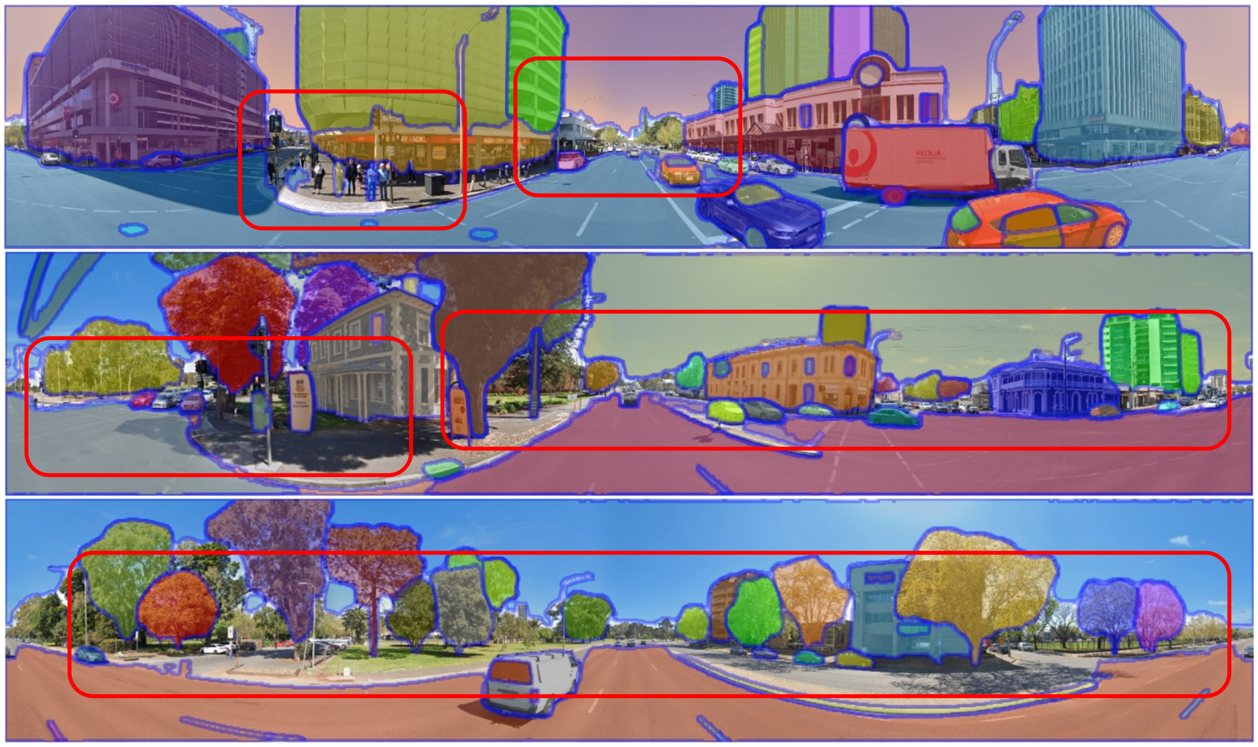
Figure 4. Example visualization results of SAM on panoramic images.

enhancing panoramic semantic segmentation performance. In Fig. 3, we provide TSNE visualizations [12] that clearly illustrate the significant improvements achieved by each of our proposed loss functions in distinguishing pixels from different categories within the high-level feature space.

5. Code Implementation

In this section, we provide the **demo implementation** codes of our proposed prototype extraction operation (PrototypeExtraction.py), the cross dual attention module (CrossDualAttention.py) and the ERP-to-Tangent projection (ERP2Tangent.py) used in this work, enabling replication and further exploration of the proposed SFUDA framework. The complete Version will be publicly available upon acceptance.

```
from math import pi
import math
import numpy as np
import torch
import torch.nn.functional as F
def pair(t):
    return t if isinstance(t, tuple) else (t, t)

def eq2tangent(img, height=224, width=224):
    [erp_h, erp_w, _] = img.shape
    img_new = torch.tensor(img, dtype=torch.
    float32) / 255
    img_new = img_new.permute(2,0,1).unsqueeze(0)
    FOV = [120, 120]
    FOV = [FOV[0]/360.0, FOV[1]/180.0]
    FOV = torch.tensor(FOV, dtype=torch.float32)
    PI = math.pi
    PI_2 = math.pi * 0.5
    PI2 = math.pi * 2
    yy, xx = torch.meshgrid(torch.linspace(0, 1,
    height), torch.linspace(0, 1, width))
    screen_points = torch.stack([xx.flatten(), yy
    .flatten()], -1)
    num_rows = 4
    num_cols = [3, 6, 6, 3]
    phi_centers = [-67.5, -22.5, 22.5, 67.5]
    phi_interval = 180 // num_rows
    all_combos = []
    erp_mask = []
    for i, n_cols in enumerate(num_cols):
        for j in np.arange(n_cols):
            theta_interval = 360 / n_cols
            theta_center = j * theta_interval +
    theta_interval / 2
            center = [theta_center, phi_centers[i
    ]]
            all_combos.append(center)
            up = phi_centers[i] + phi_interval /
    2
            down = phi_centers[i] - phi_interval
    / 2
            left = theta_center - theta_interval
    / 2
            right = theta_center + theta_interval
     / 2
            up = int((up + 90) / 180 * erp_h)
            down = int((down + 90) / 180 * erp_h)
            left = int(left / 360 * erp_w)
            right = int(right / 360 * erp_w)
            mask = np.zeros((erp_h, erp_w), dtype
    =int)
            mask[down:up, left:right] = 1
            erp_mask.append(mask)
    all_combos = np.vstack(all_combos)
    shifts = np.arange(all_combos.shape[0]) *
    width
    shifts = torch.from_numpy(shifts).float()
    erp_mask = np.stack(erp_mask)
    erp_mask = torch.from_numpy(erp_mask).float()
    n_patch = all_combos.shape[0]
    center_point = torch.from_numpy(all_combos).
    float()
    center_point[:, 0] = (center_point[:, 0]) /
    360
    center_point[:, 1] = (center_point[:, 1] +
    90) / 180
    cp = center_point * 2 - 1
    cp[:, 0] = cp[:, 0] * PI
    cp[:, 1] = cp[:, 1] * PI_2
    cp = cp.unsqueeze(1)
    convertedCoord = screen_points * 2 - 1
    convertedCoord[:, 0] = convertedCoord[:, 0] *
     PI
    convertedCoord[:, 1] = convertedCoord[:, 1] *
     PI_2
    convertedCoord = convertedCoord * (torch.ones
    (screen_points.shape, dtype=torch.float32) *
    FOV)
    convertedCoord = convertedCoord.unsqueeze(0).
    repeat(cp.shape[0], 1, 1)
    x = convertedCoord[:, :, 0]
    y = convertedCoord[:, :, 1]

    rou = torch.sqrt(x ** 2 + y ** 2)
    c = torch.atan(rou)
    sin_c = torch.sin(c)
    cos_c = torch.cos(c)
    lat = torch.asin(cos_c * torch.sin(cp[:, :,
    1]) + (y * sin_c * torch.cos(cp[:, :, 1])) /
    rou)
    lon = cp[:, :, 0] + torch.atan2(x * sin_c,
    rou * torch.cos(cp[:, :, 1]) * cos_c - y *
    torch.sin(cp[:, :, 1]) * sin_c)
    lat_new = lat / PI_2
    lon_new = lon / PI
    lon_new[lon_new > 1] -= 2
    lon_new[lon_new<-1] += 2
    lon_new = lon_new.view(1, n_patch, height,
    width).permute(0, 2, 1, 3).contiguous().view(
    height, n_patch*width)
    lat_new = lat_new.view(1, n_patch, height,
    width).permute(0, 2, 1, 3).contiguous().view(
    height, n_patch*width)
    grid = torch.stack([lon_new, lat_new], -1)
    grid = grid.unsqueeze(0).to('cuda')
    persp = F.grid_sample(img_new, grid, mode='
    bilinear', padding_mode='zeros',
    align_corners=True)
    persp_int = persp[0].permute(1, 2, 0)
    persp_int = persp_int * 255
    return persp_int
```

ERP2Tangent.py

```
def PrototypeExtraction(num_classses, feat,
    target):
    size_f = (feat.shape[2], feat.shape[3])
    feat_1 = nn.Upsample(size_f, mode='nearest')(
    target.unsqueeze(1).float()).expand(feat.size
    ())
    center_feat = feat_1.clone()
    for i in range(num_classses):
        mask_feat = (feat_1 == i).float()
        center_feat_T = (1 - mask_feat) *
    center_feat_T + mask_feat * ((mask_feat *
    feat_2).sum(-1).sum(-1) / (mask_feat.sum(-1).
    sum(-1) + 1e-6)).unsqueeze(-1).unsqueeze(-1)
    return center_feat_S
```

PrototypeExtraction.py

```
import torch
import torch.nn as nn
import torch.nn.functional as F

def CrossDualAttention(FFP_feat, ERP_feat):
    kl_loss = nn.KLDivLoss(size_average=None,
    reduce=None, reduction='mean', log_target=
    True)
    b,c,h,w = ERP_feat.size()
    FFP_feat = torch.reshape(FFP_feat, [b,c,h,w])
    FFP_feat = FFP_feat.permute(0,2,3,1)
    ERP_feat = ERP_feat.permute(0,2,3,1)
    FFP_re = torch.reshape(FFP_feat, [b*h*w,c])
    ERP_re = torch.reshape(ERP_feat, [b*h*w,c])
    print(ERP_re.size())
    print(FFP_re.size())
    FFP_tr = FFP_re.transpose(1,0)
    ERP_tr = ERP_re.transpose(1,0)
    print(FFP_tr.size())
    print(ERP_tr.size())

    M_sp_prime = torch.mm(ERP_re, FFP_tr)
    M_sp = torch.mm(FFP_re, ERP_tr)
    M_ch = torch.mm(FFP_tr, ERP_re)
    M_ch_prime = torch.mm(ERP_tr, FFP_re)

    loss_cda = kl_loss(F.softmax(M_sp_prime), F.
    softmax(M_sp)) + kl_loss(F.softmax(M_ch), F.
    softmax(M_ch_prime))
    return loss_cda
```

CrossDualAttention.py

References

[1] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[2] Xiaoqing Guo, Jie Liu, Tongliang Liu, and Yixuan Yuan. Simt: Handling open-set noise for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7032–7041, 2022. 2

[3] Lukas Hoyer, Dengxin Dai, and Luc Van Gool. Daformer: Improving network architectures and training strategies for domain-adaptive semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9924–9935, 2022. 3

[4] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. Model adaptation: Historical contrastive learning for unsupervised domain adaptation without source data. *Advances in Neural Information Processing Systems*, 34:3635–3649, 2021. 2

[5] Tarun Kalluri, Girish Varma, Manmohan Chandraker, and CV Jawahar. Universal semi-supervised semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5259–5270, 2019. 3

[6] Jogendra Nath Kundu, Akshay Kulkarni, Amit Singh, Varun Jampani, and R Venkatesh Babu. Generalize then adapt: Source-free domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7046–7056, 2021. 2

[7] Yuang Liu, Wei Zhang, and Jun Wang. Source-free domain adaptation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1215–1224, 2021. 2, 3, 4

[8] Yawei Luo, Liang Zheng, Tao Guan, Junqing Yu, and Yi Yang. Taking a closer look at domain shift: Category-level adversaries for semantics consistent domain adaptation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2502–2511, 2019. 3

[9] Chaoxiang Ma, Jiaming Zhang, Kailun Yang, Alina Roitberg, and Rainer Stiefelhagen. Densepass: Dense panoramic semantic segmentation via unsupervised domain adaptation with attention-augmented context exchange. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2766–2772. IEEE, 2021. 2

[10] Marin Orsic, Ivan Kreso, Petra Bevandic, and Sinisa Segvic. In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12607–12616, 2019. 3

[11] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Peter Kontschieder. Seamless scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8277–8286, 2019. 3

[12] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 5

[13] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 2

[14] Zhonghao Wang, Mo Yu, Yunchao Wei, Rogerio Feris, Jinjun Xiong, Wen-mei Hwu, Thomas S Huang, and Honghui Shi. Differential treatment for stuff and things: A simple unsupervised domain adaptation method for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12635–12644, 2020. 3

[15] Cheng-Yu Yang, Yuan-Jhe Kuo, and Chiou-Ting Hsu. Source free domain adaptation for semantic segmentation via distribution transfer and adaptive class-balanced self-training. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2022. 2, 3, 4

[16] Kailun Yang, Xinxin Hu, Luis M Bergasa, Eduardo Romera, and Kaiwei Wang. Pass: Panoramic annular semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 21(10):4171–4185, 2019. 3

[17] Kailun Yang, Jiaming Zhang, Simon Reiß, Xinxin Hu, and Rainer Stiefelhagen. Capturing omni-range context for omnidirectional segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1376–1386, 2021. 3

[18] Xiangyu Yue, Zangwei Zheng, Shanghang Zhang, Yang Gao, Trevor Darrell, Kurt Keutzer, and Alberto Sangiovanni Vincentelli. Prototypical cross-domain self-supervised learning for few-shot unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13834–13844, 2021. 3

[19] Jiaming Zhang, Chaoxiang Ma, Kailun Yang, Alina Roitberg, Kunyu Peng, and Rainer Stiefelhagen. Transfer beyond the field of view: Dense panoramic semantic segmentation via unsupervised domain adaptation. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 3

[20] Jiaming Zhang, Kailun Yang, and Rainer Stiefelhagen. Issafe: Improving semantic segmentation in accidents by fusing event-based data. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1132–1139. IEEE, 2021. 3

[21] Jiaming Zhang, Kailun Yang, Chaoxiang Ma, Simon Reiß, Kunyu Peng, and Rainer Stiefelhagen. Bending reality: Distortion-aware transformers for adapting to panoramic semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16917–16927, 2022. 3

[22] Jiaming Zhang, Kailun Yang, Hao Shi, Simon Reiß, Kunyu Peng, Chaoxiang Ma, Haodong Fu, Kaiwei Wang, and Rainer Stiefelhagen. Behind every domain there is a shift: Adapting distortion-aware vision transformers for panoramic semantic segmentation. *arXiv preprint arXiv:2207.11860*, 2022. 2

[23] Pan Zhang, Bo Zhang, Ting Zhang, Dong Chen, Yong Wang, and Fang Wen. Prototypical pseudo label denoising and target

structure learning for domain adaptive semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12414–12424, 2021. 2

[24] Xu Zheng, Tianbo Pan, Yunhao Luo, and Lin Wang. Look at the neighbor: Distortion-aware unsupervised domain adaptation for panoramic semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18687–18698, 2023. 2, 3

[25] Xu Zheng, Jinjing Zhu, Yexin Liu, Zidong Cao, Chong Fu, and Lin Wang. Both style and distortion matter: Dual-path unsupervised domain adaptation for panoramic semantic segmentation. *arXiv preprint arXiv:2303.14360*, 2023. 3

[26] Yang Zou, Zhiding Yu, Xiaofeng Liu, BVK Kumar, and Jinsong Wang. Confidence regularized self-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5982–5991, 2019. 3