Semi-Supervised Learning of Semantic Correspondence with Pseudo-Labels

Jiwon Kim[1*] Kwangrok Ryoo[1*] Junyoung Seo[1*] Gyuseong Lee[1*]
Daehwan Kim[2] Hansang Cho[2] Seungryong Kim[1†]
[1]Korea University, Seoul, Korea [2]Samsung Electro-Mechanics, Suwon, Korea
{naancoco,kwangrok21,se780,jpl358,seungryong_kim}@korea.ac.kr
{daehwan85.kim,hansang.cho}@samsung.com

In this supplementary material, we provide implementation details, extensive analyses, and more experimental results. We first provide the implementation details to reproduce the main experimental results in Sec. 1. To prove our novelty, we provide more analyses on our key components, i.e., confidence constraints on pseudo-labels and data augmentation in Sec 2. In addition, we provide the inference results of randomly generated keypoints to prove that using lots of pseudo-labels is effective for finding detailed correspondences. In Sec 3, we include additional quantitative and qualitative results on the semantic correspondence benchmark dataset (i.e., PF-PASCAL, PF-Willow and SPair-71k).

1. Implementation Details

Hyper-parameters. As mentioned in Sec 4.1 of the main paper, we used almost identical network architecture and hyper-parameters of CATs [1]. We provide additional hyper-parameters we used in Table 1 below. To verify the optimality of the hyper-parameters, we conduct ablation studies on our confidence constraints on pseudo-labels and data augmentation in the main paper. We also conduct additional analyses on the unsupervised loss components to prove generality of our components in the following Sec 2.

hyperparameters	SemiMatch
B	10
τ	0.5
T_u	1.5
$p_{aug\_photo\_source}$	0.2
$p_{aug\_photo\_weak}$	0.2
$p_{aug\_photo\_strong}$	0.2
t_{scale_tps}	0.4
$\gamma_{contrastive}$	0.1

Table 1. **Additional hyperparameter list of SemiMatch**

PyTorch-like Pseudo Code. We provide the PyTorch-like pseudo code for SemiMatch in Algorithm 1.

2. Ablation Study

Loss Configuration. Before we begin the analysis, we first define each of the two key components of the unsupervised loss. First, probability with uncertainty estimation is scaled by T_u, such that $1/\exp(T_u \cdot \sum_j P(i,j) \log P(i,j))$ where $P(i,j)$ is j-th target component of matching probabilities. Based on the importance of adjusting the scale of unsupervised loss in proportional to supervised loss [6], we adjust the scale of unsupervised loss through *dynamic* λ defined as $\lambda =$

*Equal contribution
†Corresponding author

Algorithm 1: SemiMatch Pseudo Code, PyTorch-like

```
for I_S, I_T, GT_keypoints in loader:
    theta = compute_syn_theta(random.rand())
    feat_S = net.feature_extraction(I_S)
    feat_Tw = net.feature_extraction(photometric_weak(I_T))
    feat_Ts = net.feature_extraction(geometric_warp(photometric_strong(I_T, GT_keypoints),theta))

    # dimension of corr.:  (T_feat_H * T_feat_W , S_feat_H * S_feat_W)
    corr_S_Tw, corr_S_Ts = net.correlation(feat_S,feat_Tw), net.correlation(feat_S,feat_Ts)
    corr_Tw_S = net.correlation(feat_Tw, feat_S) # for fb_check

    mask_bbox = get_bbox_mask(feat_Tw.shape,GT_keypoints).int()
    mask_fb = foward_backward_check(corr_S_Tw, corr_Tw_S).int()
    mask_thres = (torch.max(corr_S_Tw, dim=1) > tau).int() * exp(-Tu * uncertainty(corr_S_Tw))
    mask = mask_bbox * mask_fb * mask_thres

    map_S_Tw = soft_argmax(corr_S_Tw)
    masked_map_S_T = mask * map_S_Tw # hard labelling

    pseudo_map = geometric_warp(masked_map_S_Tw, theta)

    flow_pred = convert_flow(map_S_Tw)
    GT_flow = keypoint_to_flow(GT_keypoints)

    sup_loss = EPE(flow_pred,GT_flow)
    unsup_loss = contrastive_loss(pseudo_map.detach(), corr_S_Ts)

    dynamic_lamda = sup_loss.detach() / unsup_loss.detach()
    Loss = sup_loss + dynamic_lamda * unsup_loss

    Loss.backward()
    update(net) # update parameters

def uncertainty(correlation_map) :
    prob = soft_max(correlation_map,temp)
    return (-prob * torch.log(prob)).sum(dim=1)
```

$\mathcal{L}^*_{\mathrm{sup}}/\mathcal{L}^*_{\mathrm{un-sup}}$, where $\mathcal{L}^*$ is the loss value itself and no back propagation happens. We conduct ablation studies on the PCK result, convergence rate, and the requirement of warm-up stage depending on the hyper-parameters of the unsupervised loss.

PCK Results in Details. Table 2 shows the importance of the unsupervised loss configuration through the PCK results compared to the our best experiment setting. In Table 2 (a), we prove that our T_u is appropriate to measure the confidence of the pseudo-labels. In addition, PCK decreases in Table 2 (b) demonstrates that uncertainty-based confidence estimation is important. Furthermore, scaling of unsupervised loss dynamically for each iteration is effective rather than using constant scale factor.

(a) Uncertainty temperature for $\mathrm{M}_{\mathrm{thres}}$

method	PF-PASCAL (PCK@0.05)
(I) Ours (1.5)	**80.1**
(II) Soft (1.3)	79.7
(III) Hard (1.7)	79.3

(b) Components of unsupervised loss

method	PF-PASCAL (PCK@0.05)
(I) Ours	**80.1**
(II) (-) Uncertainty	79.5
(III) (-) Dynamic λ	79.2

Table 2. **Ablation studies of components in unsupervised loss configuration**

Effectiveness of Uncertainty. As shown high PCK compared to (II) in Table. 3, which has the same level of probability thresholding, we prove that uncertainty estimation is effective for stable training without warm-up stage. In addition, through comparison between (I) and (III), it can be seen that it is more effective to use uncertainty than to use high probability thresholding. The effectiveness of uncertainty on confidence estimation is also demonstrated through high PCK in Table 2,

method	PF-PASCAL		
	0.05	0.1	0.15
(I) w/ uncertainty ($\tau = 0.5$)	**78.9**	**93.5**	**96.8**
(II) w/o uncertainty ($\tau = 0.5$)	77.1	93.1	96.3
(III) w/o uncertainty ($\tau = 0.9$)	76.2	92.6	96.4

Table 3. **Ablation studies of uncertainty without warm-up stage**

Data Augmentation. We conduct experiments not only on our novel keypoint-aware cutout (KeyOut), but also on augmentation combinations suitable for dense correspondence that have not been explored in dense correspondence tasks. By organizing new data augmentation combination for dense correspondence, adding Blur and KeyOut to the existing augmentation combination, we can confirm that those data augmentation improves the performance of SemiMatch as shown in Table 4.

Method	Aug	Blur	KeyOut	PCK@0.05
SemiMatch	✓	✓	✓	80.0
	✓	✓	✗	79.4
	✓	✗	✓	79.6

Table 4. **PCK@0.05 results depending on organizing data augmentation combination and using pseudo-labeling.**

Visualization. When learning matching networks with semi-supervised framework, consisting of supervised loss and unsupervised loss, they can find the correspondences on keypoints as well as around their peripheries. To prove this, we generate new matching points at test time and conduct experiments whether SemiMatch can find correspondences for new points compared to baseline and other state-of-the-art algorithms.

(a) (b) (c)

Figure 1. Matching results for random keypoint which is not GT keypoint in the test-set. The left figures are CATs results and the right figures are SemiMatch results.

3. More Experimental Results

Quantitative Results. As shown in Table 1 of the main paper, we achieve a significant performance improvement at PCK@0.05 compared to other previous state-of-the-art methods and our baseline, CATs [1]. It can be attributed to the fact that SemiMatch is much more sensitive to minor difference between keypoint and its periphery, resulting in better PCK results with much stricter matching criteria. To prove this, we show performance comparisons for baseline, CATs [1], and SemiMatch through tables and curve graphs. As shown in the Table. 5, 6, and Table. 7, our approach provides better PCK performance than CATs between α ranges from 0.01 to 0.1 on PF-PASCAL [2], PF-Willow [2] and SPair-71k [4], respectively. In PF-PASCAL, we record the largest difference by 16.16 PCK@0.02. In addition, we also find the significant differences by 4.05 PCK@0.06 in PF-Willow, and 1.42 PCK@0.04 in SPair-71k. As PCK's α increases, PCK difference generally decreases but it is natural phenomenon because the performance is less related to the training status of the network.

α	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
CATs [1]	7.8	28.2	49.9	64.8	75.4	80.9	85.7	88.7	90.8	92.6
SemiMatch	**19.4**	**44.3**	**61.0**	**72.0**	**80.1**	**85.0**	**88.2**	**90.4**	**92.3**	**93.5**

Table 5. **Comparison with CATs [1] in PCK on PF-PASCAL [2]**

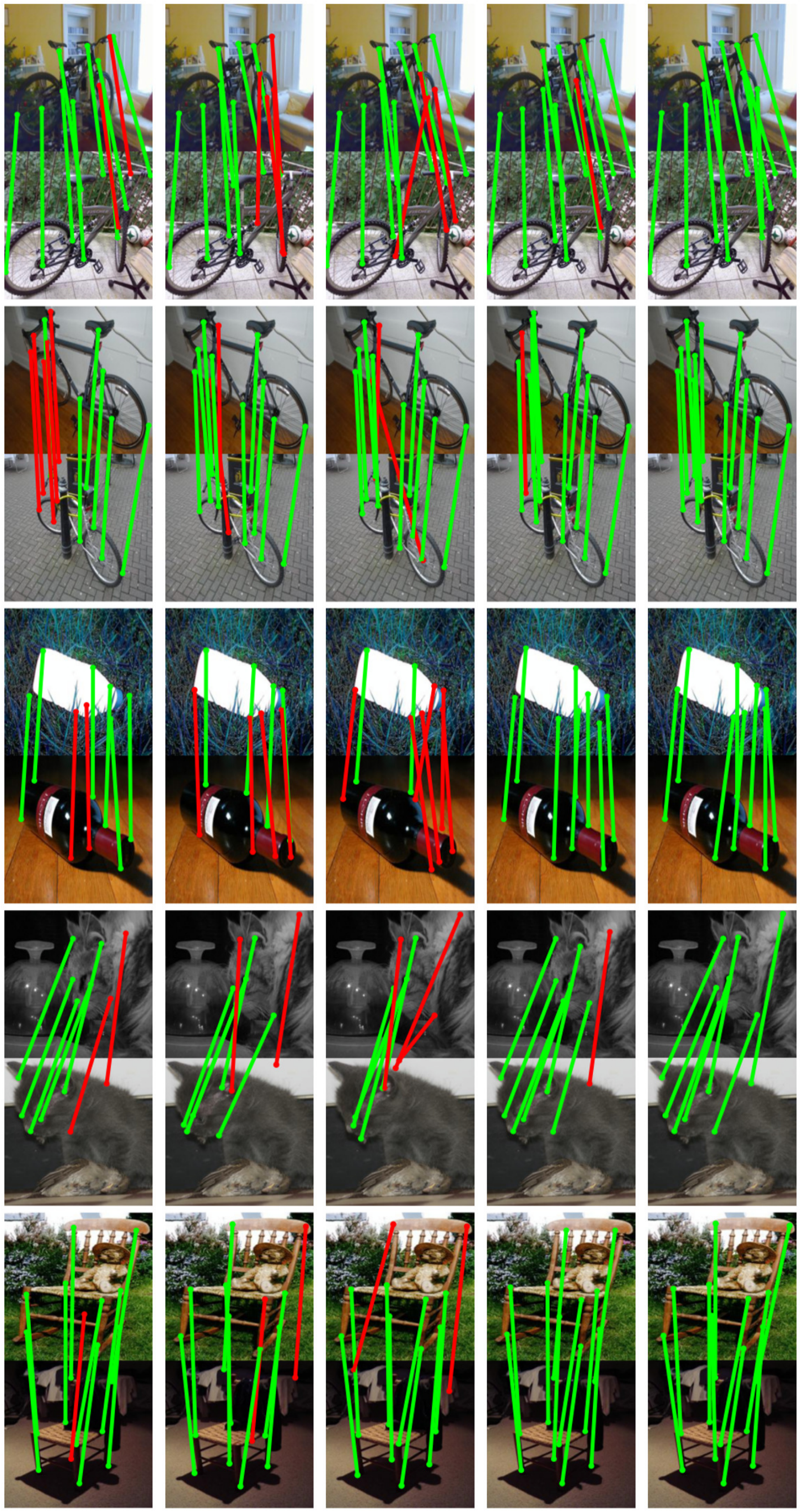
α	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
CATs [1]	**4.3**	14.2	27.3	40.4	50.3	58.5	65.0	70.7	75.5	79.2
SemiMatch	4.0	**15.5**	**30.5**	**43.8**	**54.0**	**62.3**	**69.0**	**74.4**	**78.6**	**82.1**

Table 6. **Comparison with CATs [1] in PCK on PF-Willow [2]**

α	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1
CATs [1]	1.93	7.0	13.8	20.9	27.7	33.6	38.6	43.0	46.8	49.9
SemiMatch	**2.1**	**7.7**	**15.0**	**22.4**	**29.0**	**34.8**	**39.7**	**43.9**	**47.6**	**50.7**

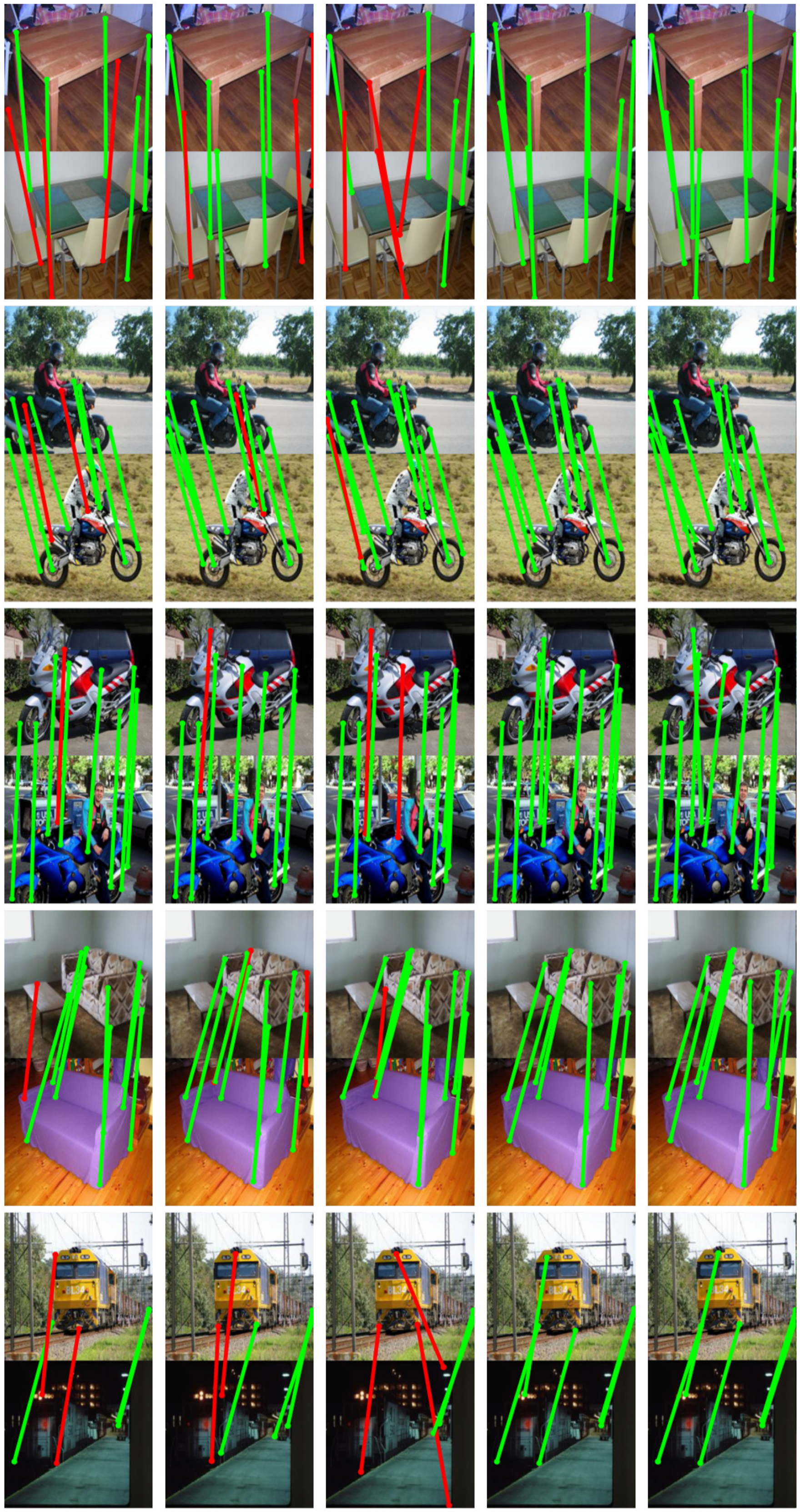
Table 7. **Comparison with CATs [1] in PCK on SPair-71k [4]**

Qualitative Results. We provide the additional qualitative results to complement Fig.1 in the main paper by including more visualization results and comparison on other state-of-the-art algorithms, such as DHPF [5], MMNet [7], CHMNet [3], CATs [1]. We perform the PCK results and visualization mentioned above for PF-PASCAL [2] (Figure. 2, Figure. 3), PF-Willow [2](Figure. 4, Figure. 5), and SPair-71k [4](Figure. 6, Figure. 7) respectively.



(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 2. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **PF-PASCAL** [2] dataset.

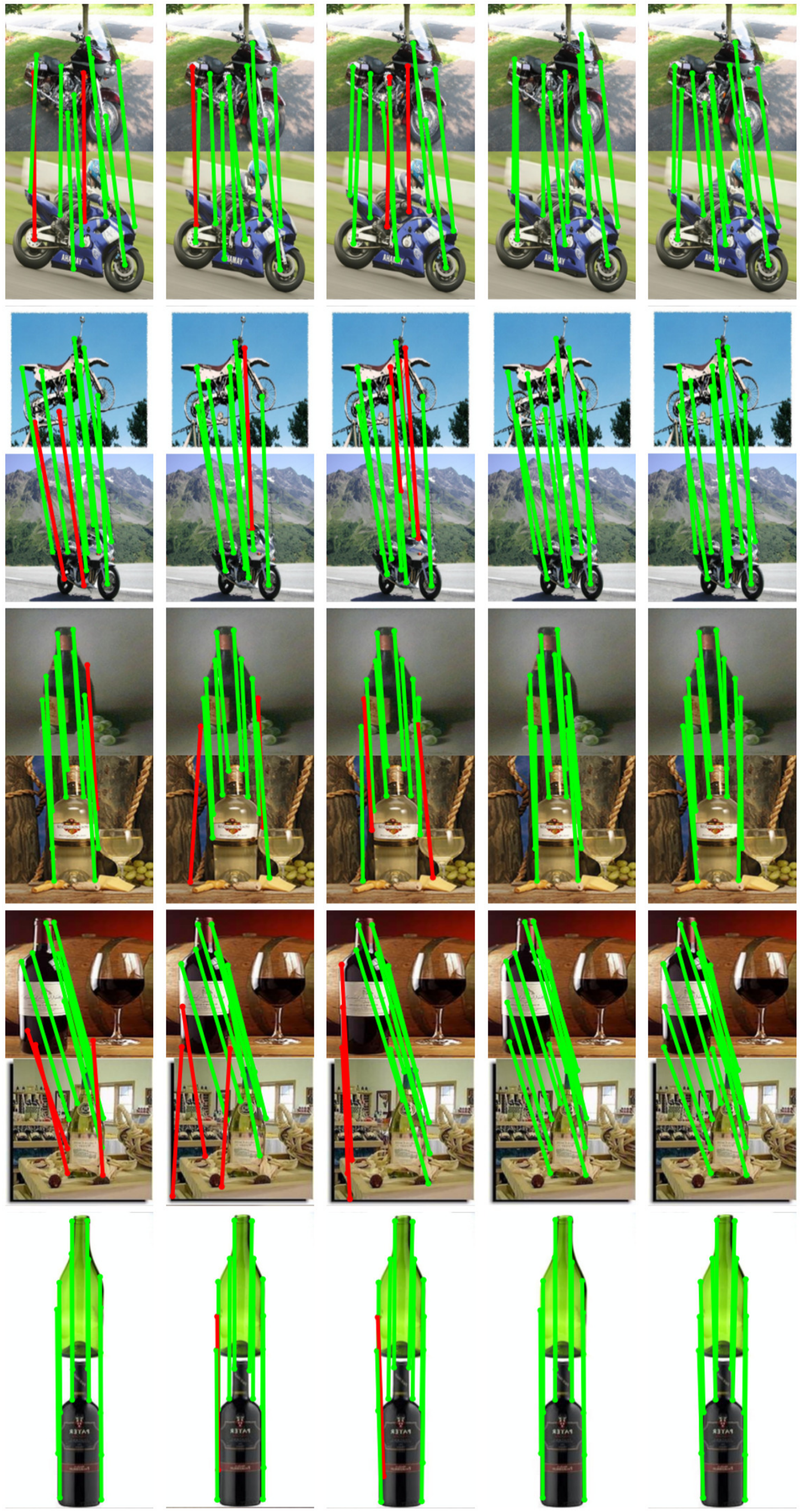


(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 3. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **PF-PASCAL** [2] dataset.

(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 4. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **PF-Willow** [2] dataset.

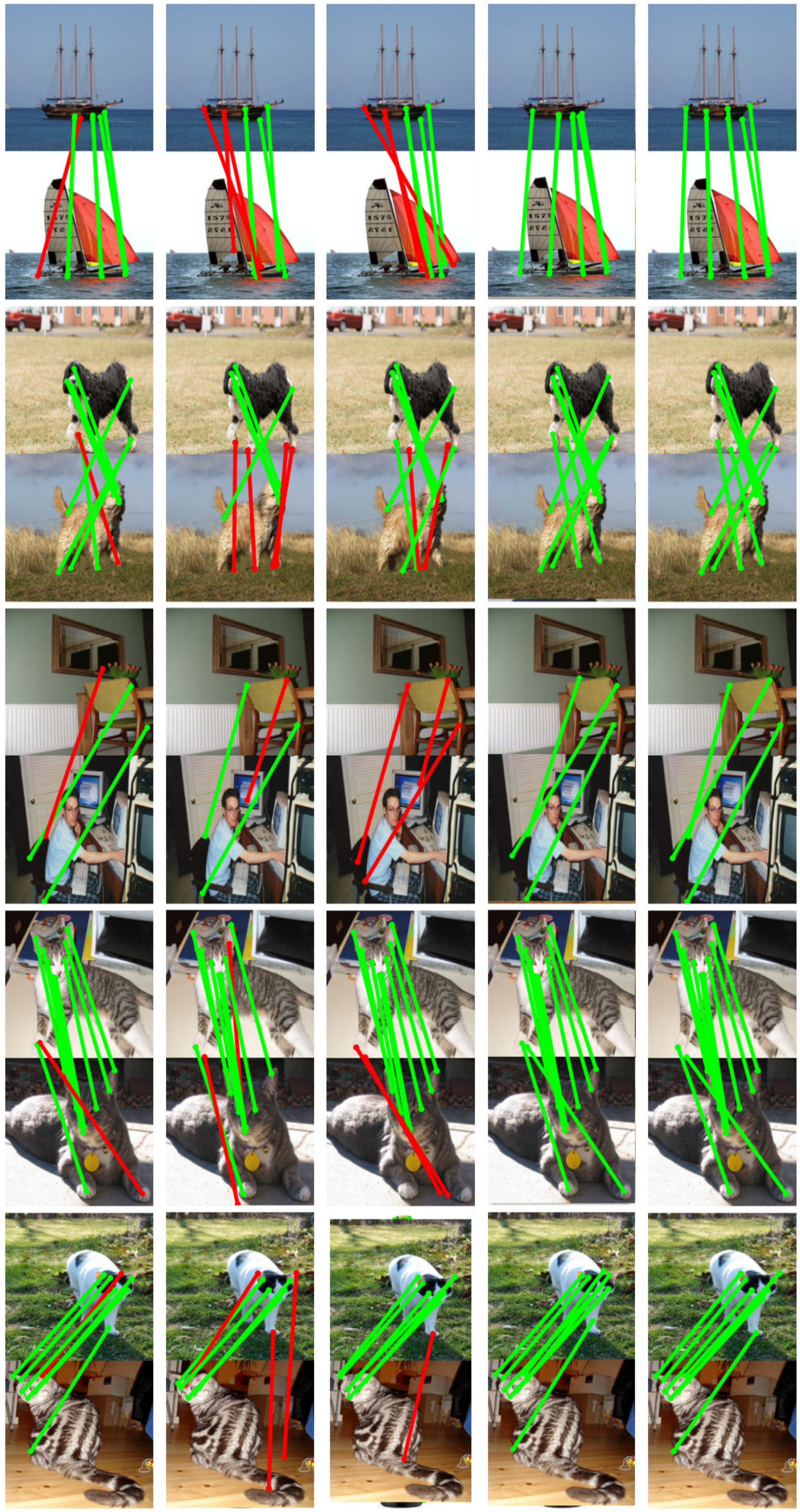


(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 5. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **PF-Willow** [2] dataset.

(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 6. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **SPair-71k** [4] dataset.



(a) CATs [1] (b) DHPF [5] (c) CHMNet [3] (d) SemiMatch (e) Ground Truth

Figure 7. Qualitative examples of multiple networks and our SemiMatch applied to pairs of **SPair-71k** [4] dataset.

References

[1] Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Semantic correspondence with transformers. *arXiv preprint arXiv:2106.02520*, 2021. 1, 3, 4, 5, 6, 7, 8, 9, 10

[2] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow: Semantic correspondences from object proposals. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1711–1725, 2017. 3, 4, 5, 6, 7, 8

[3] Juhong Min and Minsu Cho. Convolutional hough matching networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2940–2950, 2021. 4, 5, 6, 7, 8, 9, 10

[4] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv preprint arXiv:1908.10543*, 2019. 3, 4, 9, 10

[5] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Learning to compose hypercolumns for visual correspondence. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16*, pages 346–363. Springer, 2020. 4, 5, 6, 7, 8, 9, 10

[6] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021. 1

[7] Dongyang Zhao, Ziyang Song, Zhenghao Ji, Gangming Zhao, Weifeng Ge, and Yizhou Yu. Multi-scale matching networks for semantic correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3354–3364, 2021. 4