Supplemental Material for ECO-TR: Efficient Correspondences Finding Via Coarse-to-Fine Refinement

Dongli Tan[1,3⋆], Jiang-Jiang Liu[2,3⋆], Xingyu Chen[3], Chao Chen[3], Ruixin Zhang[3], Yunhang Shen[3], Shouhong Ding[3], and Rongrong Ji[1,4] ✉

[1] Media Analytics and Computing Lab, School of Informatics, Xiamen University
[2] TMCC, CS, Nankai University
[3] Youtu Lab, Tencent Technology (Shanghai) Co.,Ltd
[4] Institute of Artificial Intelligence, Xiamen University
{dltan921,j04.liu}@gmail.com, rrji@xmu.edu.cn

In this supplementary material, we give additional details about our experiment settings and results. We first introduce more details during the training and testing stages in Sec. 1 and Sec. 2, respectively. Then, we describe more experimental results for speed comparison in Sec. 3. Finally, we show more qualitative results in Sec. 4.

1 More training details

Architecture. We implemented our model in PyTorch [2]. The local feature CNN uses a modified version of ResNet-50 as a backbone without pretraining. The sizes of the coarse-, middle-, and fine-level feature maps are 1/32, 1/16, and 1/4 of the original image size, respectively. The channel number of each layer's feature is converted to 256 by 1×1 convolutions. For the transformer, we use three layers for both encoder and decoder. Same with COTR, we disallow self-attention among the query points in the decode stage. For coarse-to-fine refinement modules, we set the crop window size $w^M = 17$, $w^F = 13$. For the AQC module, we set $t = 1$, $K_{num} = 128$. The distance threshold Th is set to 0.8 times of the corresponding side of patches during training and 0.6 times during inference.

Training data. Our model is trained on the outdoor dataset MegaDepth. Training pairs are generated in an on-the-fly manner based on the ground truth intrinsics, poses and depth maps. For a fair comparison with COTR, We use the same 115 scenes for training. There are two kinds of pairs, zoomed pairs and non-zoomed pairs. For zoomed pairs, an extra random zoom level between $1\times$ and $3\times$ is selected among ten levels in the log scale and applied to images. We mix these two kinds of pairs up during training.

⋆ Authors contributed equally.

Generating correspondences We take 20 images from the same scene for each image in the training set, which have the largest common areas as candidates. Every image in the training set selects one corresponding image among candidates and forms image pairs for each epoch. Then, we re-project the depth map of one image to the other and filter out matches with a depth difference larger than 0.5. To accelerate the pipeline mentioned above, we re-project sampled 20,000 pixels rather than all pixels in one image. We only select two pairs among all cropped patches that have the most matches in a single iteration for each image.

Training details. Our model is trained using Adam [1] with an initial learning rate of 10^{-4} and a batch size of 256. Uncertainty branches are detached during the first 1500 epochs, while other parameters except uncertainty branches are detached for the last 100 epochs. 32 Tesla A100 GPUs are used during training, and synchronized batch normalization is applied. N = 512 correspondences are sampled for each level's crops.

2 More testing details

Using cycle consistency check strategy For all experiments(except the ablations), cycle consistency check is applied as a complementary filter besides the uncertainty filter. We find cycle consistency check is effective in large geometric transformation(e.g., pairs in HPatches).

Adaptive inference batch size Due to the limitation of GPU memory, we separate cropped patches to mini-batch of 32 during refinement stages.

Resizing image pairs Moreover, we find that simply expanding image pairs gives better results. We resize images by bilinear interpolation. Images are 1.5× enlarged at inference time.

3 Details for speed test in Fig. 1 of the main paper

We run the inference speed test in Fig. 1 of the main paper on a single NVIDIA Tesla V100. Input images are resized to 832×832 before being fed into COTR's and ECO-TR's networks. COTR's framework takes 8,182 MB of GPU memory during inference, while ECO-TR's framework takes 7,752 MB.

Additionally, we compare the inference speed of image pairs with two different resolutions in Table 1. Taking a pair of images with the resolution of $1,248 \times 1,248$ as input, ECO-TR uses up to 13,161 MB of memory during inference. The inference speed of images with $1,248 \times 1,248$ pixels are slightly slower than images with 832×832 pixels, but it is still dozens of times faster than the previous method.

Table 1. Inference speed under different resolutions We test the inference speed of COTR and ECO-TR with input images of different resolutions. A single NVIDIA Tesla V100 is used here.

Time (s) # Queries / Method	100	300	500	1,000	2,000	5,000	10,000
COTR(832×832)	5.33	13.46	22.21	43.56	87.28	217.16	434.81
ECO-TR(832×832)	0.31	0.62	0.93	1.02	1.12	1.62	2.36
COTR($1,248 \times 1,248$)	7.14	17.82	29.49	59.24	117.49	295.80	585.01
ECO-TR($1,248 \times 1,248$)	0.39	0.73	1.01	1.16	1.38	2.26	3.68

4 More qualitative results

More qualitative results of ECO-TR on HPatches, KITTI, and MegaDepth are shown in Fig. 1, Fig. 2 and Fig. 3, respectively. ECO-TR can handle images pairs in different aspect ratios, though images are resized to the same size before being fed into the network.

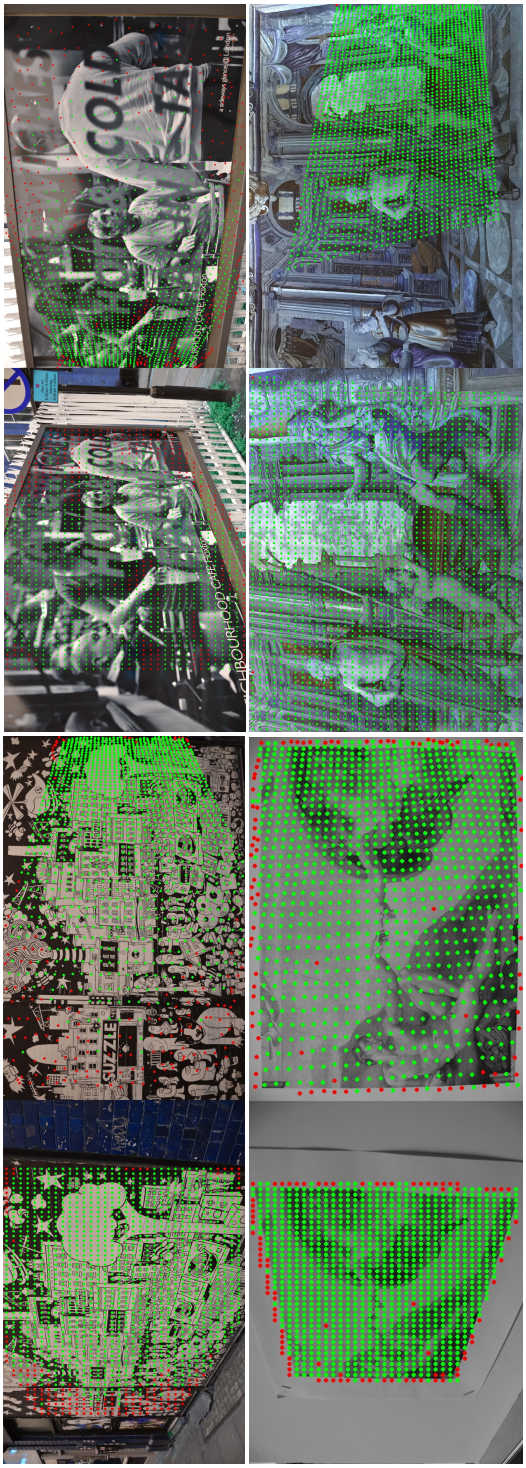

Fig. 1. Qualitative results on HPatch dataset.

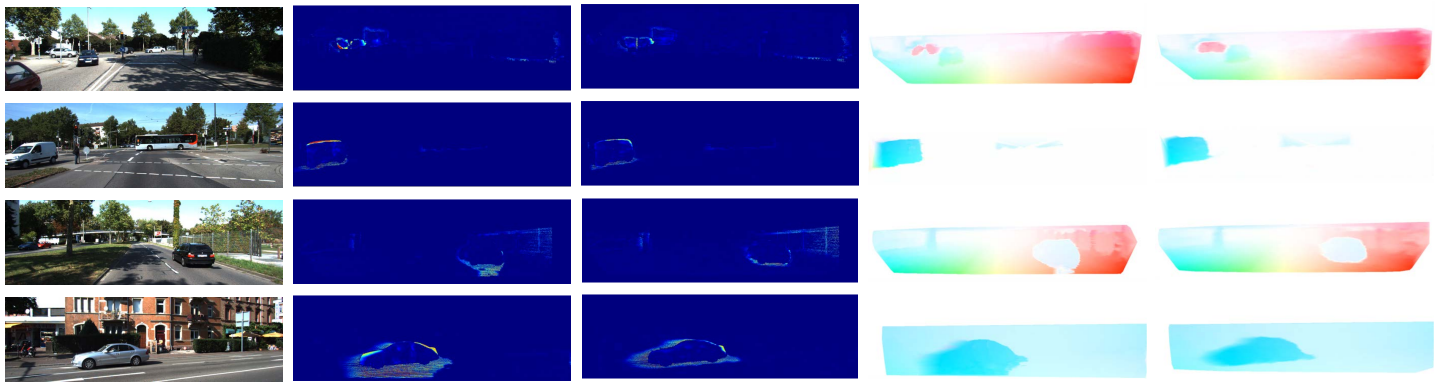

Fig. 2. Qualitative results on KITTI dataset.

Fig. 3. Qualitative results on MegaDepth dataset

References

1. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
2. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)