Supplementary for "STEERER: Resolving Scale Variations via Selective Inheritance Learning"

Tao Han[1], Lei Bai[1]*, Lingbo Liu[2], Wanli Ouyang[1]
[1]Shanghai Artificial Intelligence Laboratory, [2]The Hong Kong Polytechnic University
{hantao10200, baisanshi,liulingbo918}@gmail.com, wanli.ouyang@sydney.edu.au

The supplementary provides additional details on various aspects related to the paper, which are organized as follows:

1) **Extra Experiments Analysis**,
2) **Detailed Network Architectures**,
3) **Localization Algorithm**,
4) **Computational statistics**,
5) **Limitations**,
6) **More Visualization Results**.

1. Extra Experiments Analysis

1.1. Evaluations Metrics

Object counting: In line with previous studies [23, 7, 21], we employ the following metrics to evaluate counting performance: Mean Absolute Error (MAE), root Mean Squared Error (MSE), and mean Normalized Absolute Error (NAE). Given a test set comprising K images, we define C_k and $\hat{C}_k$ as the ground truth count and predicted count, respectively, for the k-th image.

$$MAE = \frac{1}{K}\sum_{k=1}^{K}\left|C_k - \hat{C}_k\right|; RMSE = \sqrt{\frac{1}{K}\sum_{k=1}^{K}\left(C_k - \hat{C}_k\right)^2}; NAE = \frac{1}{K}\sum_{k=1}^{K}\frac{\left|C_k - \hat{C}_k\right|}{C_k}, \tag{1}$$

where some negative samples (i.e., images with no objects) are excluded from the calculation of NAE to avoid zero denominators.

Object Localization. To assess the localization performance, we employ three standard metrics: Precision (Pre.), Recall (Rec.), and F-measure (F1-m). We determine the True Positive (TP), False Positive (FP), and False Negative (FN) values by using the Hungarian algorithm to compare the two point sets from the prediction results ($\mathbf{P}_p$) and the ground truth ($\mathbf{P}_g$). The metrics are then calculated as follows:

$$\begin{aligned} \text{Precision} &= \tfrac{\text{TP}}{\text{TP+FP}} \\ \text{Recall} &= \tfrac{\text{TP}}{\text{TP+FN}} \\ \text{F-measure} &= 2\tfrac{\text{Pre.}\times\text{Rec.}}{\text{Pre.+Rec.}} \end{aligned} \tag{2}$$

where true positives, false positives, and false negatives are determined by matching predicted locations and ground truth locations with a threshold σ. When assessing crowd localization on datasets with box-level annotations, such as NWPU-Crowd [21] and FDST [3], we define the distance threshold σ as $\sqrt{w^2 + h^2}$ for each ground-truth

*Corresponding author

point, where w and h represent the width and height of the box, respectively. For datasets without box-level annotations, we follow the approach of IIM [4] and use proxy box annotations to determine the distance threshold. For URBAN_TREE [18], the distance threshold is set at 6 meters, or 10 pixels (where each pixel represents $0.6m \times 0.6m$).

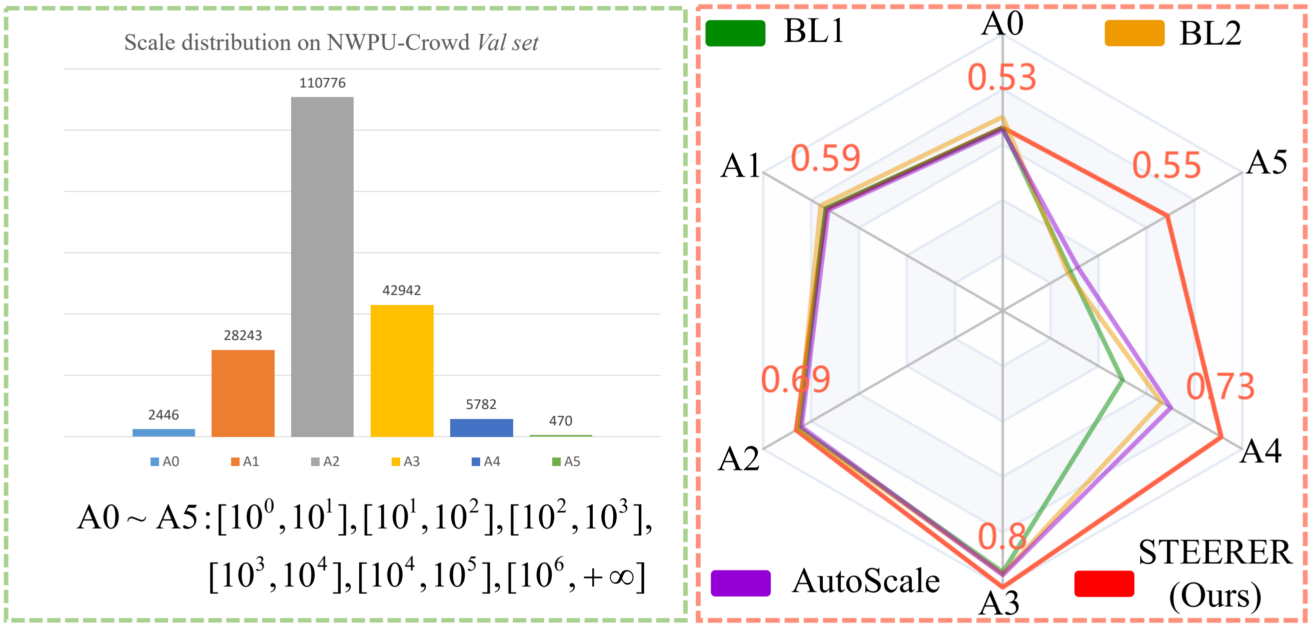

Figure 1. The left plot shows the scale distribution of head box areas in the NWPU-Crowd dataset, where A_i denotes the corresponding range. On the right plot, we compare the scale generalization ability on the NWPU-Crowd *val* set among several methods, including our baseline methods BL1 (concatenation fusion) and BL2 (FPN [9] fusion), as well as the pre-trained model of AutoScale [22]. Our proposed method, STEERER, exhibits substantial improvement in scale generalization for large objects.

1.2. Quantification of Scale Generalization

Most scale-aware object counting methods typically assess scale generalization ability by comparing simple and easily calculated metrics such as MAE and MSE. However, these metrics lack specificity. In this work, we propose a fine-grained analysis of scale generalization by dividing objects in a dataset into six scale-related groups and evaluating instance-level counting accuracy within each group. Our analysis is conducted on the NWPU-Crowd *val set*, which has separated objects into six groups $A0 \sim A5$ based on head box areas (as shown in Fig. 1 left). For instance, to evaluate scale-instance-level density prediction accuracy within a group $A0$, we use the K ground truth box annotations $B_o = \{x_i, y_i, w_i, h_i\}_{i=o}^{K}$ as indices to retrieve the corresponding predicted and ground truth density slices S_0^{pre} and S_0^{gt}, respectively. We then calculate this cluster's scale-instance-level density prediction accuracy using Eq. (3).

$$ACC_{A_i} = 1 - \frac{1}{K} \sum \frac{\min(|S_i^{gt}|_1, |S_i^{pre} - S_i^{gt}|_1)}{|S_i^{gt}|_1}. \tag{3}$$

Tab. 1 tabulates the scale generalization performance on AutoScale [22], two baselines, and our proposed STEERER. Looking at this table together with Fig. 1, we find that the traditional multi-scale feature fusion methods and the learn-to-scale method both suffer from serious performance reduction on large objects. STEERER, however, can obtain higher performance on medium-to-large objects while maintaining similar accuracy on small objects. Especially, STEERER achieves the accuracy of 73.0% and 55.9% on $A4(10,000 \sim 100,000\text{pixels})$ and $A5(100,000+\text{pixels})$ level, respectively, which significantly improve the scale generalization ability compared with the second-best results. Fig. 1 also shows that STEERER makes a balanced generalization ability on all scales.

Table 1. The comparison of scale generalization ability.

Method	ACC_{A0}(%)	ACC_{A1}(%)	ACC_{A2}(%)	ACC_{A3} (%)	ACC_{A4}(%)	ACC_{A5} (%)
BaseLine1-Concat	52.9	59.2	67.9	75.6	40.0	22.6
BaseLine2-FPN [9]	**56.1**	**60.8**	68.2	76.2	53.0	21.8
AutoScale [22]	52.3	58.3	67.2	76.4	56.1	25.1
STEERER (Ours)	53.8	59.2	**69.2**	**80.7**	**73.0**	**55.9**

1.3. Localization Performance with VGG19-backbone

Similar to the counting experiments, we also utilized the VGG19-BN backbone architecture [14] to assess the generalization performance of STEERER in the localization task. In this regard, we present the localization results of STEERER with VGG19 as the backbone. As shown in Tab. 2, our proposed method outperforms several existing methods, including detection-based methods such as TinyFaces [6] and LSC-CNN [1], a density-based method (RAZ_Loc [10]), and a segmentation-based method (IIM [4]). Specifically, on the SHHA dataset, STEERER with VGG19 achieves a 77.0 F-measure, while IIM achieves a 71.3 F-measure, although their backbones, VGG19 and VGG16, have similar volumes.

Table 2. Comparison of the crowd localization performance on other crowd datasets. The best and second-best results are shown in red and blue, respectively.

Method	Backbone	Label	ShanghaiTech Part A			ShanghaiTech Part B			UCF-QNRF			NWPU		
			F1-m	Pre.	Rec.	**F1-m**	Pre.	Rec.	**F1-m**	Pre.	Rec.	**F1-m**	Pre.	Rec.
TinyFaces [6]	ResNet-101	Box	57.3	43.1	**85.5**	71.1	64.7	79.0	49.4	36.3	**77.3**	56.7	52.9	61.1
RAZ_Loc [10]	VGG-16	Point	69.2	61.3	**79.5**	68.0	60.0	78.3	53.3	59.4	48.3	59.8	66.6	54.3
LSC-CNN [1]	VGG-16	Point	68.0	69.6	66.5	71.2	71.7	70.6	58.2	58.6	57.7	-	-	-
IIM [4]	VGG-16	Point	71.3	74.0	68.8	82.1	**92.0**	74.1	68.4	73.1	64.2	73.2	77.9	69.2
IIM [4]	HRNet	Point	73.3	76.3	70.5	83.8	**89.8**	78.6	71.8	73.7	70.1	**76.0**	**82.9**	70.2
STEERER	VGG-19	point	**77.0**	**76.4**	77.8	**85.9**	85.8	**86.0**	**74.1**	**75.4**	**73.0**	73.7	74.6	**72.7**
STEERER	HRNet	point	**79.8**	**80.0**	79.4	**88.0**	88.8	**87.1**	**75.5**	**78.6**	72.7	**77.0**	**81.4**	**73.0**

1.4. Ablate with baselines

The results in Tab. 3 compare the proposed method, STEERER, with two baselines: Baseline_1, which fuses four resolution features using an upsampling+concatenation operation, and Baseline_2, which uses FPN for bottom-to-up fusion. The table shows that STEERER achieves a significant improvement in counting accuracy compared to the two baselines, indicating its effectiveness in fusing multi-scale features. Furthermore, the results demonstrate that STEERER is more efficient for counting than the classical fusion methods in terms of multi-scale feature processing.

1.5. Datasets

Tab. 4 presents a summary of the details of the counting datasets with different classes used in this paper, including six crowd-counting datasets, two plant-counting datasets, and one vehicle dataset.

2. Network Architectures

Tab. 6 elaborates the detailed network architecture of STEERER, including two multi-scale feature representation backbones and counting head.

Multi-scale Feature Representation Backbones. Tab. 5 explains the VGG19-BN [14] and HRNet-W48 [20] configurations in STEERER. In this table, "Conv{k(3,3),c48,s1}-BN-R" represents the convolutional operation with kernel size of 3×3, output channels of 48, and stride size of 1. The "BN" and "R" mean that the Batch Normalization and ReLU layers are added to this convolutional layer. For the VGG-19 backbone, we call it from the model zoo of Pytorch [12], so we use its feature list to describe each resolution. The details of "BOTTLENECK" and "BASIC" are the same as HRNet-W48. Note that we remove the final fuse layer in HRNet-W48.

Feature Selection and Inheritance Adaptor. Tab. 5 also illustrates the specific setting of FSIA. Note that the input channels for FSIA are 96 channels in the lowest resolution and 48 channels in other resolutions. The

Table 3. The counting performance of the proposed STEERER and two baseline methods.

Method	SHHA (MAE/MSE)		NWPU (MAE/MSE)	
Baseline1: HRNet+concat	60.4	98.5	72.1	356.9
Baseline2: HRNet+FPN	60.4	96.3	78.8	365.7
STEERER_HRNet	**54.6**	**86.9**	**63.7**	**309.8**

Table 4. The summary of the main-stream counting datasets.

Class	Datasets	Total Count	Images
Crowd	SHHA [23]	241,677	482
	SHHB [23]	88,488	716
	FDST [3]	394,081	15,000
	UCF-QNRF [7]	1,251,642	1,525
	JHU-CROWD++ [15]	1,515,005	4250
	NWPU-Crowd [21]	2,133,375	5,109
Plant	MTC [11]	100	361
	URBAN_TREE [18]	39,993	573
Vehicle	TRANCOS [5]	46,796	1244

Table 5. The architecture of STEERER, including two backbones and the proposed FSIA.

Module	Res.	VGG19-bn	HRNet-W48					
			Stage1	Stage 2	Fuse	Stage 3	Fuse	Stage 4
Backbone	4×	features[0:26]	BOTTLENECK	BASIC × 1	✔	BASIC × 4	✔	BASIC × 3
	8×	features[26:39]		BASIC × 1	✔	BASIC × 4	✔	BASIC × 3
	16×	features[39:52]				BASIC × 4	✔	BASIC × 3
	32×	copy(features[39:52])						BASIC × 3

Module	UFN	SMG	SFN
FSIA	Conv{k(3,3),c48,s1}-BN-R	Conv{k(3,3),c48,s1}-BN-R	Conv{k(3,3),c48,s1}-BN-R
	Conv{k(3,3),c48,s1}-BN-R	Conv{k(3,3),c48,s1}-BN-R	Conv{k(3,3),c48,s1}-BN-R
		Conv{k(3,3),c2,s1}-BN-R	

scale-customized feature and scale-uncustomized feature output by FSIA are both 48 channels, and the counting head only accepts the 96 channels' input. Therefore, each resolution at the end of the backbone is followed by a channel down-sample layer to adapt the channel number to 48 except for the 96 in the lowest resolution.

Counting Head Localization. Tab. 6 details the configurations of counting heads for regressing the density map. In this table, "Conv{k(3,3),c64,s1}-BN-R" represents the convolutional operation with kernel size of 3×3, output channels of 64, and stride size of 1. The "BN" and "R" mean that the Batch Normalization and ReLU layers are added to this convolutional layer. In counting head, we regulate its input channels to 96 in each resolution, and it contains two upsampling operations so that the final output density map in $4\times$ resolution has the same spatial dimension as the input image.

Table 6. Detailed architecture of counting head.

Counting Head
Upsample(scale_factor=2)
Conv{k(3,3),c64,s1}-BN-R
Conv{k(3,3),c32,s1}-BN-R
Upsample(scale_factor=2)
Conv{k(3,3),c16,s1}-BN-R
Conv{k(3,3),c1,s1}-R

Algorithm 1 The code of simple but effective **Object Localization Algorithm** in a PyTorch-like style.

```
# density_map: single channel density map, size:[b,c,h,w]
# gaussian_maximum: hyper-parameters, the maximum value of the Gaussian kernel used to generate the ground truth density map
# patch_size: hyper-parameters, set a patch size to find the region's maximum value in a given density map
# threshold: hyper-parameters, a threshold to filter the background noise

def object_localization(density_map, gaussian_maximum,patch_size=128,  threshold=0.15):

    # padding density map and make it can be divided evenly by patch_size
    _,_,h,w = density_map.size()

    if h % patch_size != 0:
        pad_dims = (0, 0, 0, patch_size - h % patch_size)
        h = (h // patch_size + 1) * patch_size
        density_map = F.pad(density_map, pad_dims, "constant")

    if w % patch_size != 0:
        pad_dims = (0, patch_size - w % patch_size, 0, 0)
        w = (w // patch_size + 1) * patch_size
        density_map = F.pad(density_map, pad_dims, "constant")

    # Find the region maximum and limit its upper bound and lower bound with the threshold
    # and the gaussian_maximum
    region_maximum = F.max_pool2d(density_map, (patch_size, patch_size), stride=patch_size)
    region_maximum = region_maximum*threshold
    region_maximum[region_maximum<threshold*gaussian_maximum] = threshold*gaussian_maximum
    region_maximum[region_maximum>0.3*gaussian_maximum] = 0.3*gaussian_maximum

    # upsample region_maximum map, make it has the same spatial dimension with density_map
    region_maximum = F.interpolate(region_maximum, scale_factor=patch_size)

    # use max_pooling to find the local_maximum in density map
    local_maximum = F.max_pool2d(density_map, (3, 3), stride=1, padding=1)
    local_maximum = (local_maximum == density_map).float()
    local_maximum = local_maximum * density_map

    # judge the valid position in local_maximum map
    local_maximum[local_maximum < region_maximum] = 0
    local_maximum[local_maximum > 0] = 1

    # find the index of object positions
    points = torch.nonzero(density_map)[:,[0,1,3,2]].float() # b,c,h,w->b,c,w,h

    return points
```

3. Object Localization Algorithm

The object locations in STEERER are located from its predicted density map. The ground-truth density map is generated by convoluting an annotated dot map with a Gaussian kernel. On the contrary, we can extract the head positions by localizing the local maxima. As illustrated in Algorithm 1. Specifically, we use a 3×3 pooling operation to find the local maxima. These maximum values, however, are not always the truth positives because of the background noise and the small pooling size (compared with the Gaussian kernel (15×15)). Therefore, we make a region maximum map to filter some backbone ground noises, which is based on a patch-level maximum and a threshold multiplied by the Gaussian prior knowledge to filter out false positives. Some visualizations results in Fig. 2 demonstrate that our localization algorithm is stable and barely locates multiple positions in an object.

4. Computational statistics

Table 7 shows the computation statistics of two baselines, typical scale-aware counting methods (AutoScale [22], SASNet [17], Chfl [13], MAN [8], DM-Count [19] and HMoDE [2]), and our proposed STEERER, including parameter size, Multiply-Accumulate Operations (MACs). Flops measure the computation complexity of the model, and we test it by inputting the RGB image with a size of 768×1024. All models are tested under the same condition. AutoScale [1], SASNet [2] and HMoDE [3] *et al.* are tested with their released code. In Table 7, STEERER with HRNet-W48 has the lowest MACs even with the second largest parameters. Overall, STEERER is an energy-saving object counting method and has the potential to be a real-time counting and localization system with further optimization.

Table 7. The computational statistics comparison between the AutoScale [22], SASNet [17], HMoDE [2] and the proposed STEERER.

Method	AutoScale	SASNet	HMoDE	Chfl	MAN	DM-Count	STEERER	
							VGG19	HRNet-W48
Params(MB)	**16.7**	38.9	82.6	21.5	40.4	21.5	30.6	64.6
Flops (GMACs)	332.7	697.8	572.4	323.9	310.3	323.8	345.4	**283.4**

5. Limitations

While our work achieves promising object counting and localization performance, it has two limitations:

Slightly complex implementation compared with baselines: The implementation of our proposed counting method, STEERER, is slightly more complex compared to traditional baselines. During the training phase, STEERER outputs four density maps to determine the optimal resolution for a given patch. As a result, the training process of STEERER involves several additional details compared to classical counting methods. This increased complexity may pose a challenge during implementation; however, it should be noted that this limitation only applies to the training stage. The inference stage of STEERER is as simple as traditional object counting methods.

Despite the additional complexity during training, the performance gains achieved by STEERER make it a promising approach for counting objects in complex scenes with varying scales and densities. Furthermore, the proposed method can be optimized through further research and development to improve its overall efficiency and ease of implementation.

Hyper-parameters fine-tuning: The Patch-winner Determination Principle (PWDP) employed in our proposed method requires setting a patch size in advance. We set the patch size to 256 × 256 based on experiments conducted on the QNRF dataset, instead of searching for an optimal value for each dataset. This decision is made due to the limited time and the significant number of experiments required. Specifically, in the case of QNRF, when patch sizes are set to 128×128, 192×192, and 320 × 320, the resulting MAE/MSE values are 77.3/136.8, 77.0/134.7, and 80.5/139.8, respectively, which are not as good as the 256×256 patch size (74.3/128.3). Based on these experiment results, we fix this hyperparameter for all datasets to demonstrate the robustness of our method. However, we acknowledge that there may be better patch size settings for other datasets, as the density and scale ranges vary among different datasets.

In summary, while we have decided to fix the patch size hyperparameter for all datasets, we recognize that this may not be the optimal choice for every scenario. Therefore, we encourage further research to investigate and optimize patch sizes for different datasets and scenarios to achieve the best possible performance.

By using clean energy sources, we can significantly reduce the carbon footprint of the model training process and contribute to a more sustainable future. It is essential to consider the environmental impact of new technologies and take proactive measures to mitigate their negative effects. This can be achieved through the use of energy-efficient hardware, the optimization of algorithms to reduce computational requirements, and the use of clean energy sources. By promoting sustainable practices, we can ensure that the development and implementation of new technologies do not come at the cost of our environment.

6. More visualization results

In Fig. 2, we compare the quality of predicted density maps between some scale-aware counting methods and STEERER. The figure demonstrates that AutoScale [22], SASNet [17], and HMoDE [2] produce relatively coarse

[1] https://github.com/dk-liang/AutoScale
[2] https://github.com/TencentYoutuResearch/CrowdCounting-SASNet
[3] https://github.com/ZPDu/Redesigning-Multi-Scale-Neural-Network-for-Crowd-Counting

maps, despite accurately estimating crowd counts. In contrast, STEERER achieves fine-grained density map prediction across a wide range of scales.

The visual comparison highlights the superior performance of STEERER in generating high-quality density maps with fine-grained details. This is a critical advantage for many real-world applications that require accurate localization and tracking of individuals in crowded scenes. The ability to generate fine-grained density maps can significantly enhance the effectiveness of crowd management, security monitoring, and urban planning. Our results demonstrate that STEERER is a promising approach for addressing the challenges of scale-aware crowd counting, with potential applications in various domains.

In Fig. 3, we present high-resolution visualization samples from the UCF-QNRF dataset. Compared to our highly competitive baseline, as illustrated in the two red boxes, STEERER significantly improves the regression of large objects. This improvement in regression performance can lead to better localization performance for large objects. In Fig. 4, we provide additional samples with diverse scales and densities from the UCF-QNRF dataset. The figure demonstrates that STEERER is capable of generating high-quality density maps across a wide range of scales and densities, surpassing the performance of the competitive baseline method.

Fig. 6 shows the predicted density maps of STEERER on the NWPU validation set. These figures demonstrate that STEERER does well in producing density maps and locating people in various complicated crowded scenes with diverse scales.

Fig. 5 compares the P2PNet [16] and STEERER with the localization performance on SHHA dataset. From these samples, we find that STEERER is better at handling big objects and suppressing the noise of the background.

Fig. 7, Fig. 8, and Fig. 9 display more examples on other object counting and localization. In all these figures, STEERER produces high-quality density map prediction and accurate locations.

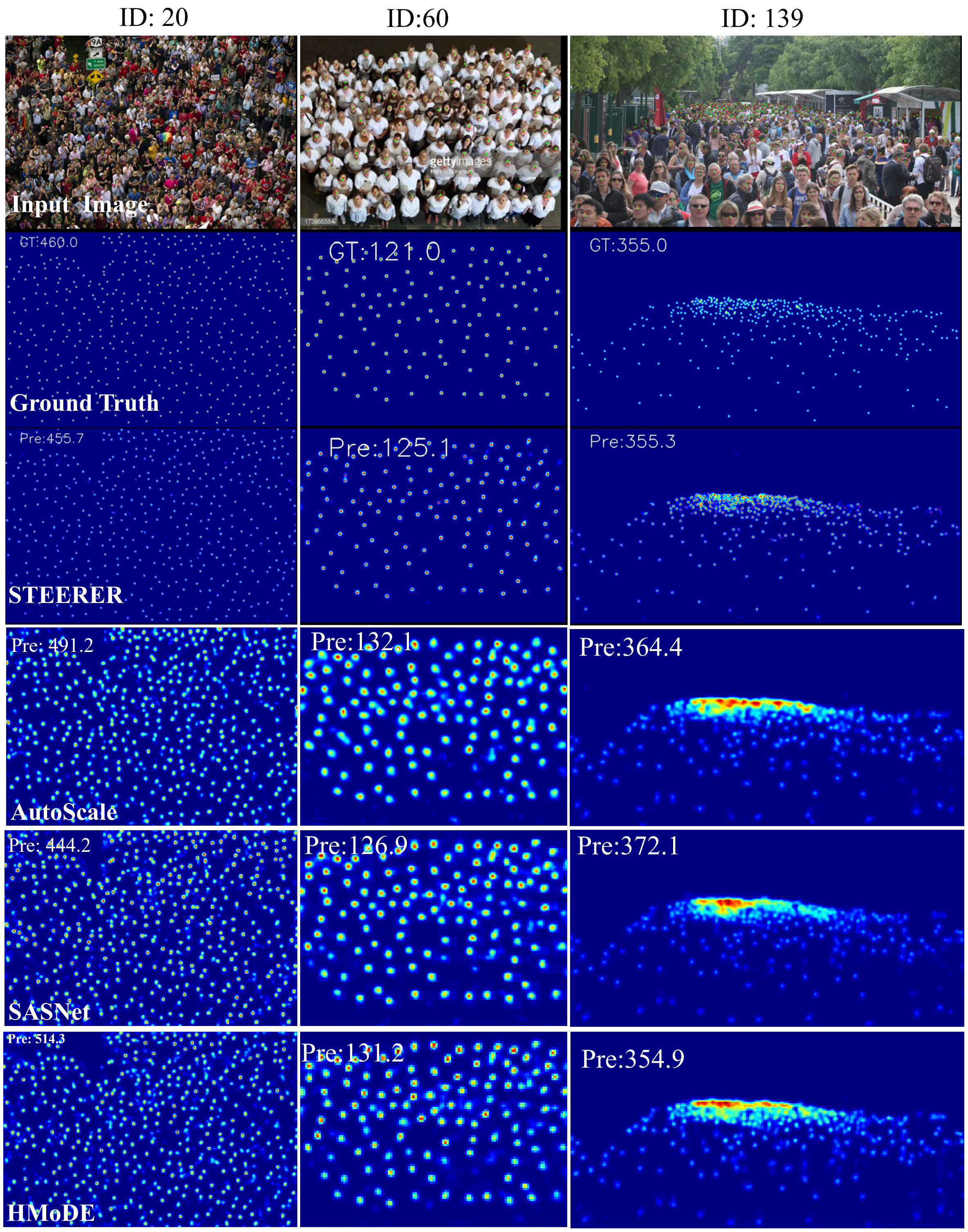

Figure 2. Visualization samples on SHHA dataset. The red and green points in input images denote ground truth locations and the predicted locations with our proposed STEERER, respectively. The results of AutoScale [22], SASNet [17], and HMoDE [2] come from their pretrained models.

References

[1] Deepak Babu Sam, Skand Vishwanath Peri, Mukuntha Narayanan Sundararaman, Amogh Kamath, and Venkatesh Babu Radhakrishnan. Locate, size and count: Accurately resolving people in dense crowds via detection. *IEEE TPAMI*, 2020. 3

[2] Zhipeng Du, Miaojing Shi, Jiankang Deng, and Stefanos Zafeiriou. Redesigning multi-scale neural network for crowd counting. *arXiv preprint arXiv:2208.02894*, 2022. 5, 6, 8

[3] Yanyan Fang, Biyun Zhan, Wandi Cai, Shenghua Gao, and Bo Hu. Locality-constrained spatial transformer network for video crowd counting. In *ICME*, pages 814–819. IEEE, 2019. 1, 4

[4] Junyu Gao, Tao Han, Yuan Yuan, and Qi Wang. Learning independent instance maps for crowd localization. *arXiv preprint arXiv:2012.04164*, 2020. 2, 3

[5] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. Extremely overlapping vehicle counting. In *PRIA*, pages 423–431. Springer, 2015. 4

[6] Peiyun Hu and Deva Ramanan. Finding tiny faces. In *CVPR*, pages 951–959, 2017. 3

[7] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *ECCV*, pages 532–546, 2018. 1, 4

[8] Hui Lin, Zhiheng Ma, Rongrong Ji, Yaowei Wang, and Xiaopeng Hong. Boosting crowd counting via multifaceted attention. In *CVPR*, pages 19628–19637, 2022. 5

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 2, 3

[10] Chenchen Liu, Xinyu Weng, and Yadong Mu. Recurrent attentive zooming for joint crowd counting and precise localization. In *CVPR*, pages 1217–1226, 2019. 3

[11] Hao Lu, Zhiguo Cao, Yang Xiao, Bohan Zhuang, and Chunhua Shen. Tasselnet: counting maize tassels in the wild via local counts regression network. *Plant methods*, 13(1):1–17, 2017. 4

[12] 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. *NeurIPS*, 32:8026–8037, 2019. 3

[13] Weibo Shu, Jia Wan, Kay Chen Tan, Sam Kwong, and Antoni B Chan. Crowd counting in the frequency domain. In *CVPR*, pages 19618–19627, 2022. 5

[14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[15] Vishwanath Sindagi, Rajeev Yasarla, and Vishal MM Patel. Jhu-crowd++: Large-scale crowd counting dataset and a benchmark method. *IEEE TPAMI*, 2020. 4

[16] Qingyu Song, Changan Wang, Zhengkai Jiang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yang Wu. Rethinking counting and localization in crowds: A purely point-based framework. In *ICCV*, pages 3365–3374, 2021. 7, 12, 13

[17] Qingyu Song, Changan Wang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Jian Wu, and Jiayi Ma. To choose or to fuse? scale selection for crowd counting. In *AAAI*, pages 2576–2583, 2021. 5, 6, 8

[18] Jonathan Ventura, Milo Honsberger, Cameron Gonsalves, Julian Rice, Camille Pawlak, Natalie LR Love, Skyler Han, Viet Nguyen, Keilana Sugano, Jacqueline Doremus, et al. Individual tree detection in large-scale urban environments using high-resolution multispectral imagery. *arXiv preprint arXiv:2208.10607*, 2022. 2, 4

[19] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. Distribution matching for crowd counting. *arXiv preprint arXiv:2009.13077*, 2020. 5

[20] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE TPAMI*, 2020. 3

[21] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE TPAMI*, 43(6):2141–2149, 2020. 1, 4

[22] Chenfeng Xu, Kai Qiu, Jianlong Fu, Song Bai, Yongchao Xu, and Xiang Bai. Learn to scale: Generating multipolar normalized density maps for crowd counting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019. 2, 3, 5, 6, 8

[23] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *CVPR*, pages 589–597, 2016. 1, 4

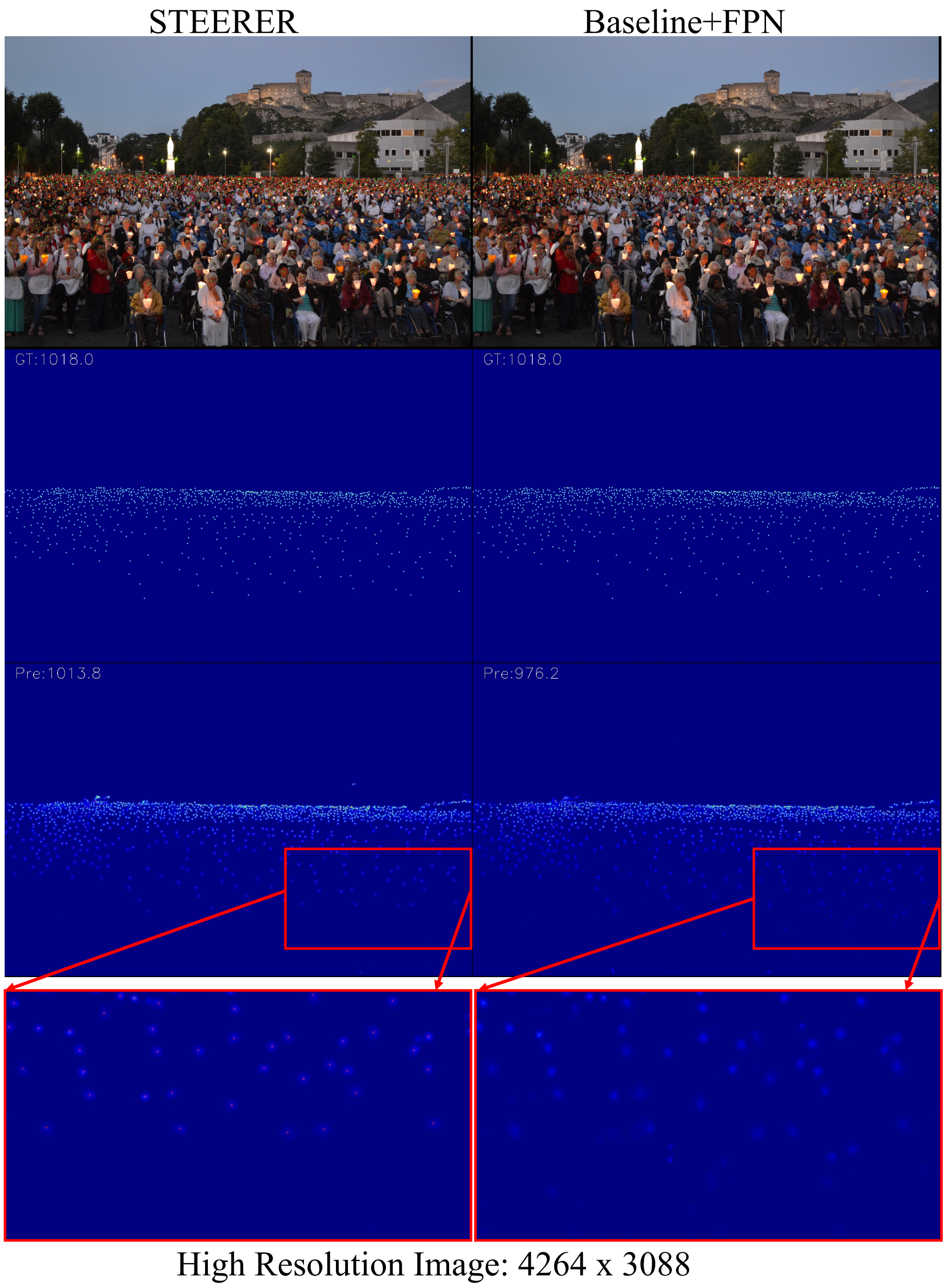

Figure 3. Super high-resolution visualization samples on UCF-QNRF dataset. Compared with our highly competitive baseline, STEERER does a better job of estimating the Gaussian density of large objects. The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

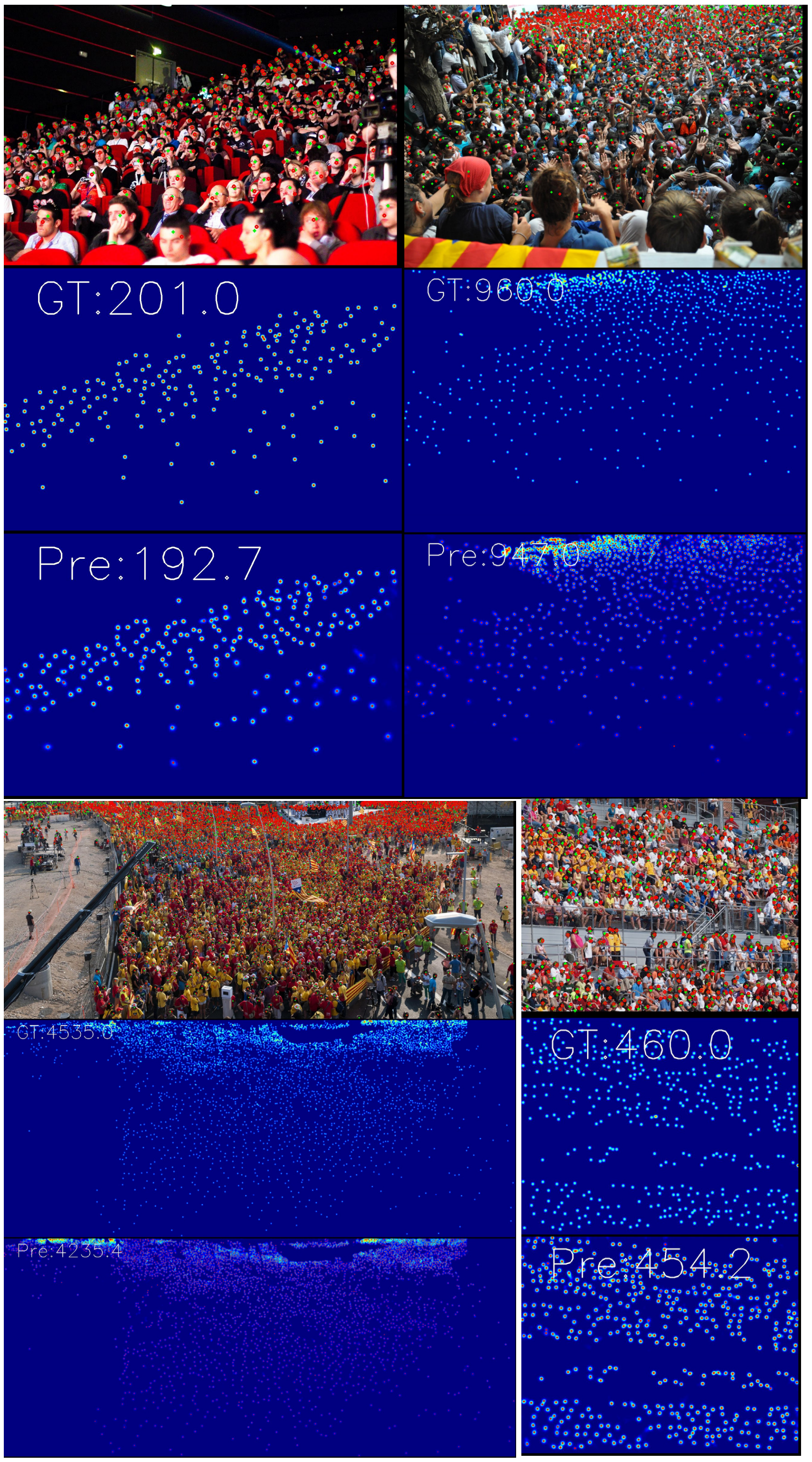

Figure 4. Visualization of diversity scales samples on UCF-QNRF dataset. The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

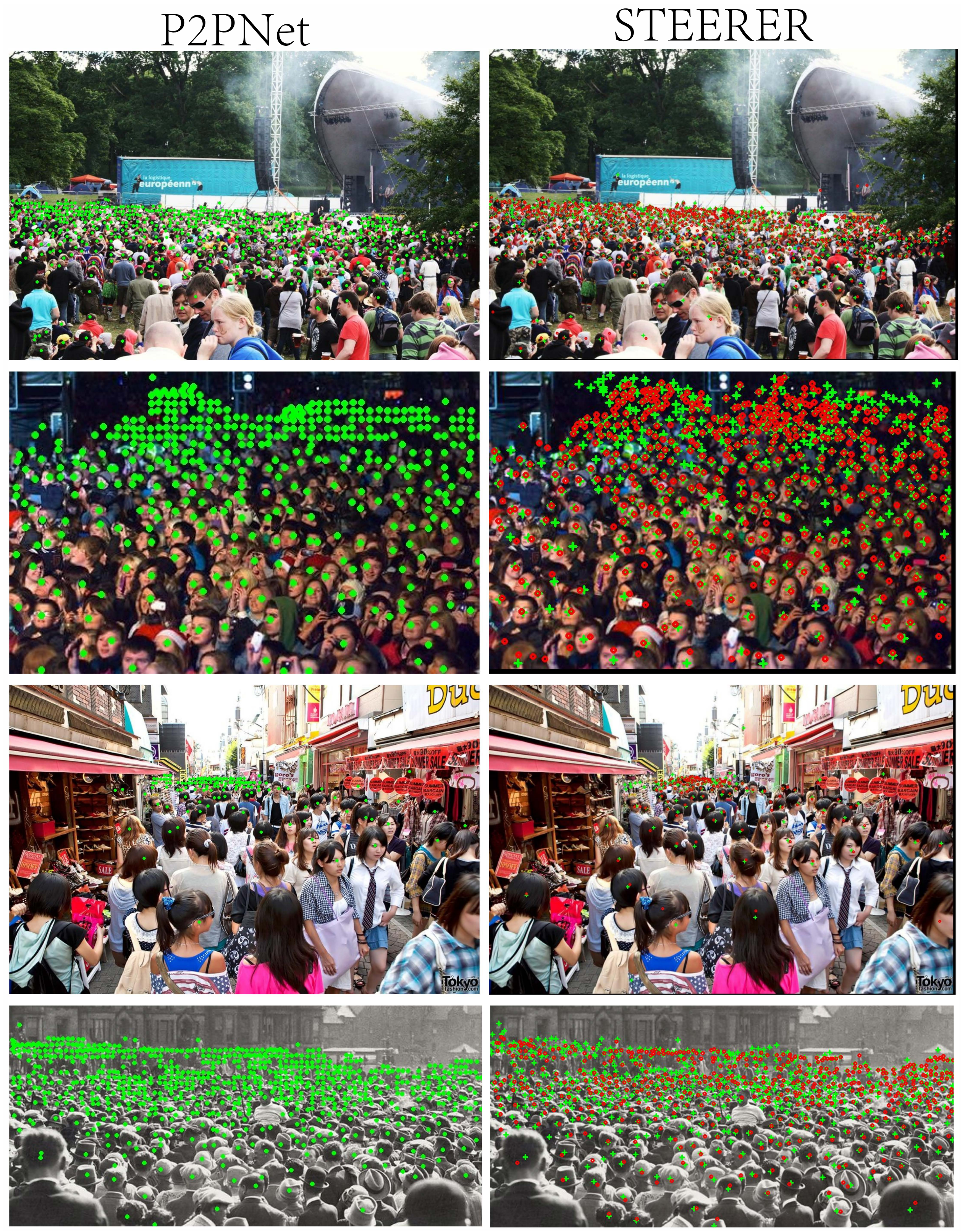

Figure 5. Localization samples comparisoned with P2PNet [16] on SHHA dataset .The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

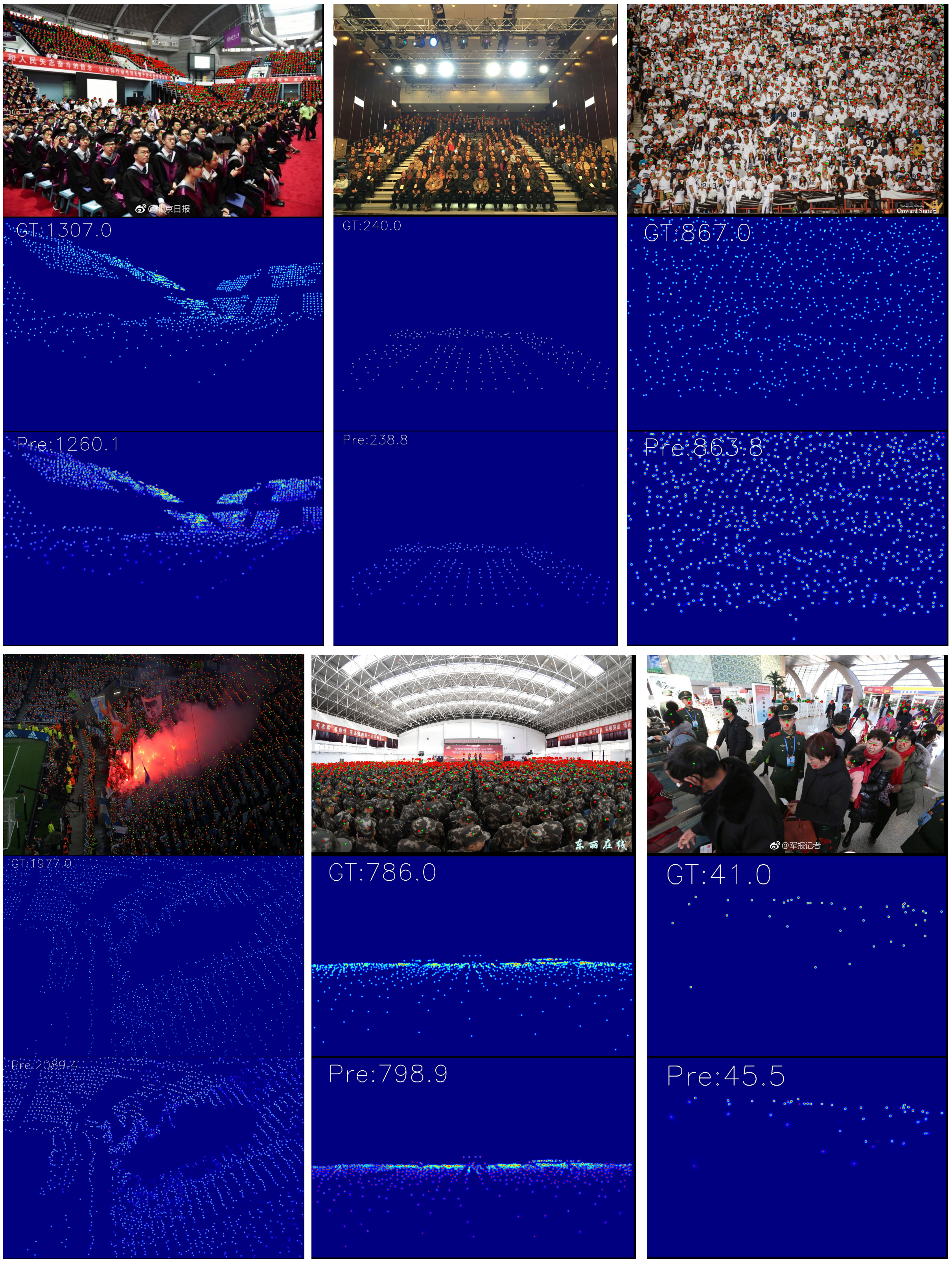

Figure 6. Localization samples comparisoned with P2PNet [16] on SHHA dataset .The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

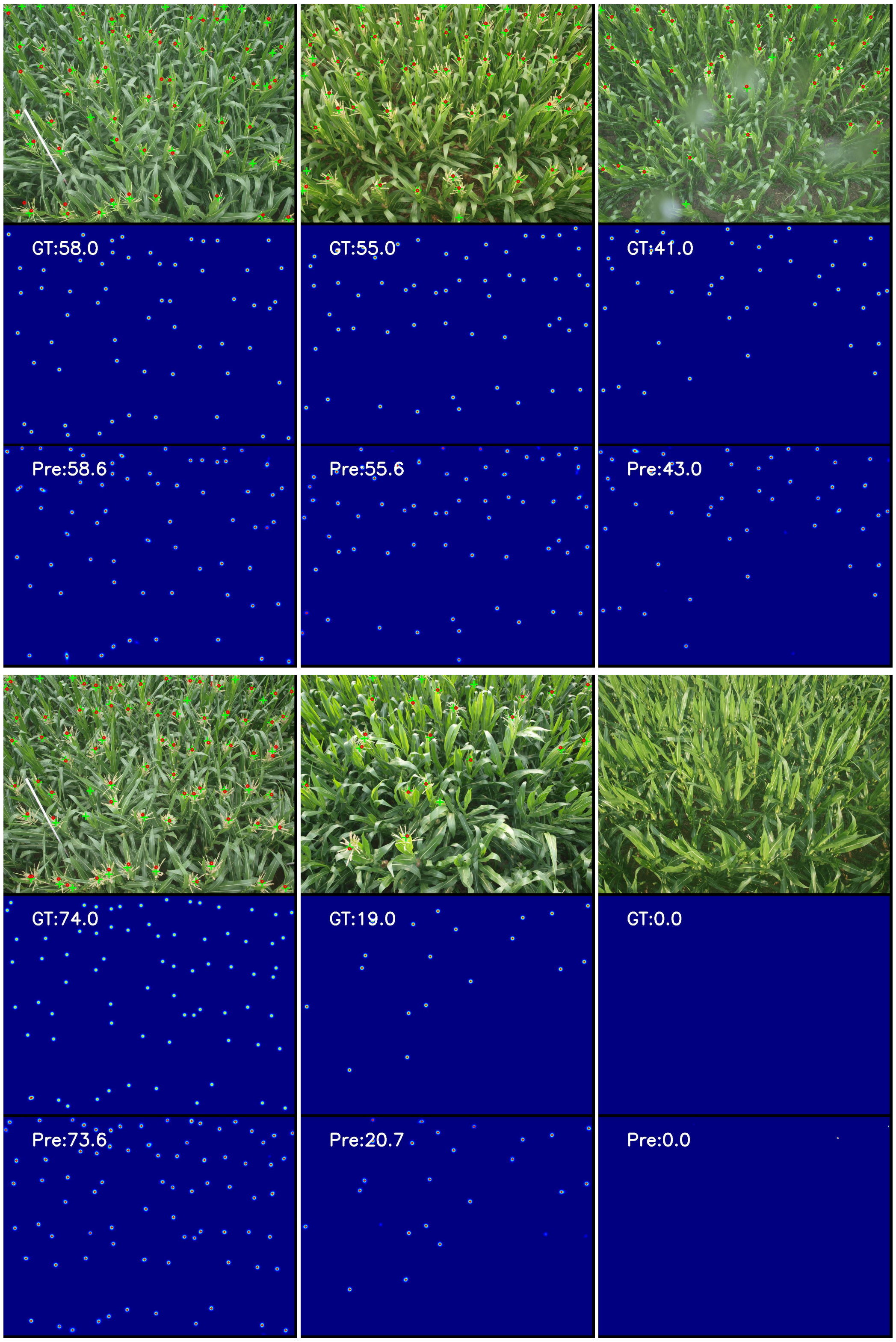

Figure 7. Visualization samples on MTC dataset. The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

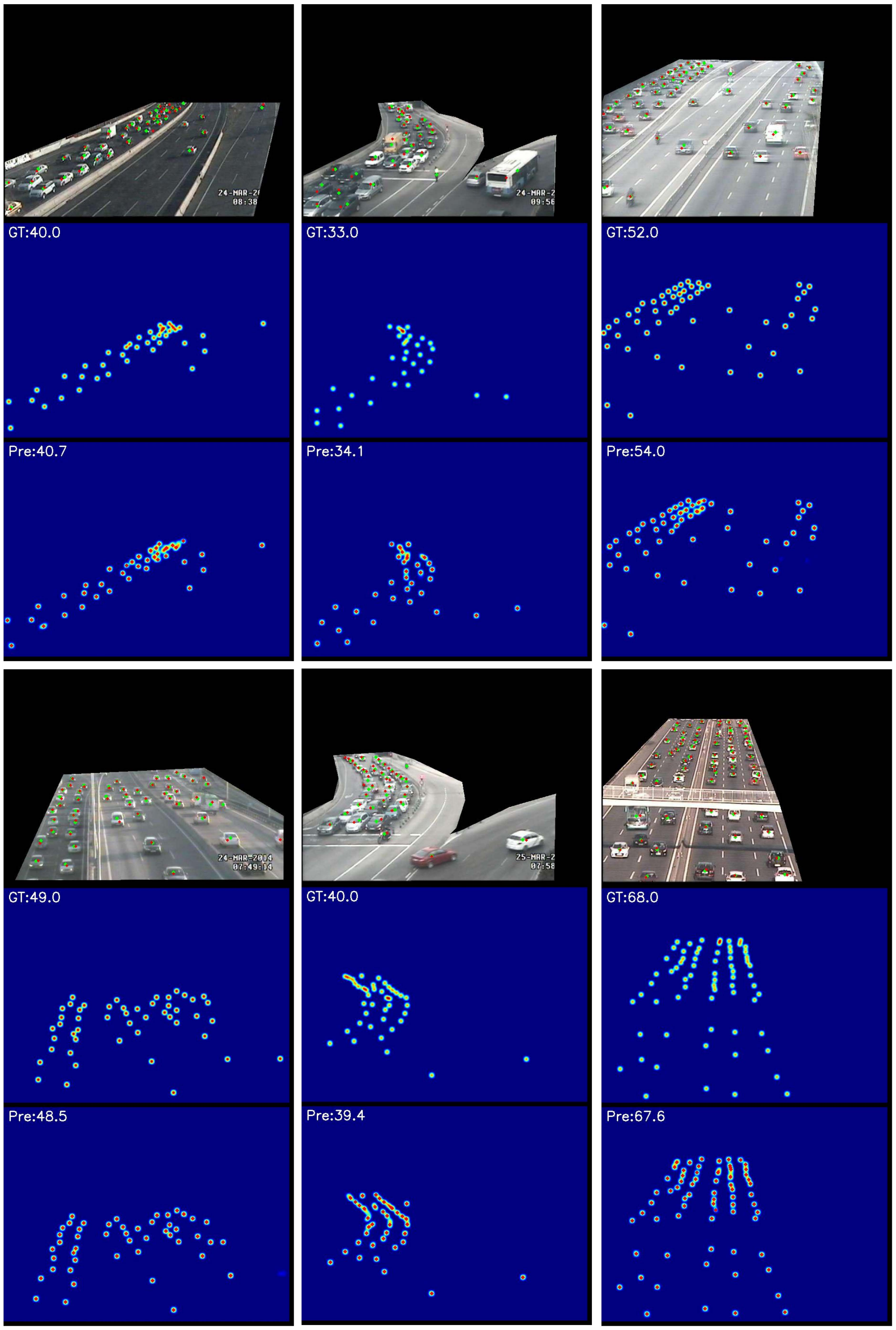

Figure 8. Visualization samples on TRANCOS dataset. The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.

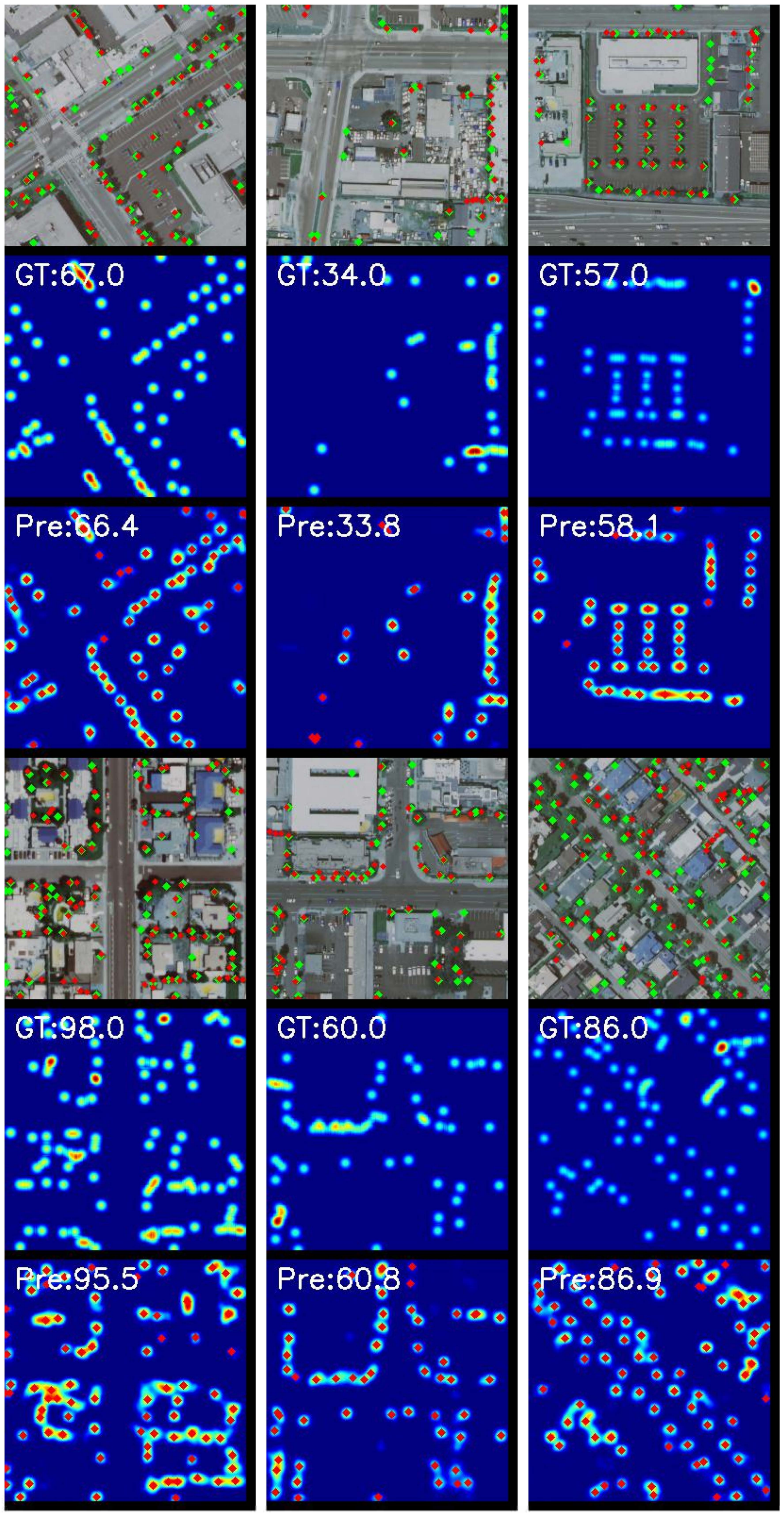

Figure 9. Visualization samples on TREE dataset. The red and green points input images denote ground truth locations and the predicted locations, respectively. Please enlarge it for a better view.