Mask Transfiner for High-Quality Instance Segmentation (Supplemental material)

Lei Ke[1,2] Martin Danelljan[1] Xia Li[1] Yu-Wing Tai[3] Chi-Keung Tang[2] Fisher Yu[1]
[1]ETH Zürich [2]HKUST [3]Kuaishou Technology

We first provide more implementation and training/inference details of Mask Transfiner on three instance segmentation benchmarks (Section 1). Then we conduct more experimental analysis and discussion of comparison between Mask Transfiner and other methods (Section 2). We further present more qualitative results comparisons on COCO [12], BDD100K [15] and Cityscapes [5] datasets in various scenes (Section 3). Finally, we visualize quadtree attention weights, detected incoherent regions and segmentation results with various quadtree depths (Section 4), including failure cases analysis.

1. More Implementation Details

Implementation and Training/Inference Details We implement Mask Transfiner based on Detectron2 [14], where SGD is used with 0.9 momentum and 1K constant warm-up iterations. The weight decay is set to 0.0001. On the two-stage and query-based frameworks, we employ Mask Transfiner using Faster R-CNN [13] and DETR [2] detectors respectively while leaving the RoI pyramid construction and refinement transformer unchanged.

To make the detection on incoherent regions more robust, we adopt jittering operations along the boundaries of the ground truth incoherent regions because in our case, the recall rate of the detection to cover all the incoherent regions play a more critical role in influencing final performance. We use 0.5 as the threshold for the binary incoherence classifier. For the experiment of Table 2 in the paper, the boundary regions are pixels within two-pixel Euclidean distance to the detected object mask contours on all three levels of the object feature pyramid, where the object boundary detector [10] is used. The coarse mask head is composed of a FCN network with four 3×3 Convs attached on the ROI feature of size 28×28.

During training, we randomly permute the order of the incoherent points for each object and select 300 of them (100 per quadtree level), so as to maintain the same sequence length for each object for batch efficiency. We adopt the horizontal flipping and scale data augmentation during training following [11].

During inference, no test-time augmentation is used. We employ a hierarchical propagation scheme based on the quadtree structure from coarse to finer scales (detailed in Section 3.1 of the paper) and the refined incoherent nodes predictions. In Figure 1, we further illustrate the mask propagation process with a simplified 3-level quadtree. The incoherent nodes number N with their refined predictions value V_n are formatted in $N : V_n$, where $\{1 : \text{v1}, 2 : \text{v2}, 5 : \text{v5}, 7 : \text{v7}, 8 : \text{v8}, 10 : \text{v10}, 13 : \text{v13}\}$ are incoherent nodes numbers and prediction values pairs in our given example. We break down the mask correction and propagation into 3 steps corresponding to 3 levels of the quadtree with visualizations. Comparing to only correcting the labels of finest leaf nodes on the quadtree, it enlarges the refinement areas with negligible cost by propagating refinement labeled to leaf nodes $\{3, 4, 6, 9, 11, 12\}$. We validate the effect of quadtree mask propagation in Table 8 of the paper.

COCO: We set 16 images per mini-batch. Following [11], our training schedule is 60k / 20k / 10k with updating learning rates 0.02 / 0.002 / 0.0002 respectively. For ablation study, our method is trained on four GPUs using ResNet-50, where we use SGD for optimization and set initial learning rate to 0.01 with total batch size 8. We train Mask Transfiner for 12 epochs (taking about 8 hours with NVIDIA RTX 2080 Ti), and decrease the learning rate by 0.1 after 8 and 11 epochs.

Cityscapes: We adopt 8 images per mini-batch and the training schedule is 18k / 6k updates at learning rates of 0.01 / 0.001 respectively. During training, the images are resized randomly to a shorter edge from [800, 1024] pixels with a step of 32 pixels. The inference images are resized to a shorter edge size of 1024 pixels. For Cityscapes evaluation, we train the models on the fine annotations of the train set with 64 epochs following [4, 11].

BDD100K: We use 16 images per mini-batch and and the training schedule is 22k / 4k/ 4k updates at learning rates of 0.02 / 0.002/ 0.0002 respectively. During training, the images are resized randomly to a shorter edge from [600, 720] pixels with a step of 24 pixels. During inference, the images are resized to a shorter edge size of 720 pixels. Note all compared methods are trained with the same schedules and image size settings.

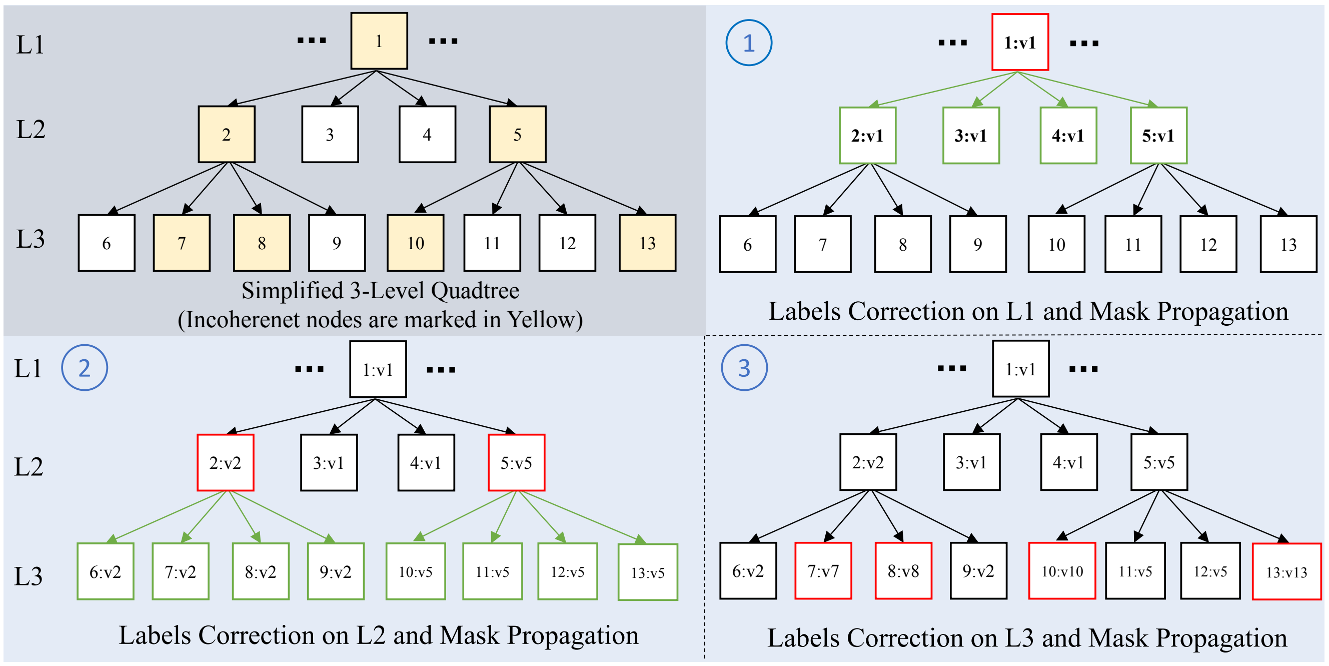
L1
L2
L3
1
2
3
4
5
6
7
8
9
10
11
12
13
Simplified 3-Level Quadtree
(Incoherenet nodes are marked in Yellow)
1
1:v1
2:v1
3:v1
4:v1
5:v1
Labels Correction on L1 and Mask Propagation
2
2:v2
5:v5
6:v2
7:v2
8:v2
9:v2
10:v5
11:v5
12:v5
13:v5
Labels Correction on L2 and Mask Propagation
3
7:v7
8:v8
10:v10
13:v13
Labels Correction on L3 and Mask Propagation

Figure 1. The simplified illustration of mask propagation on a 3-level quadtree during inference. Given the detected incoherent nodes and their refined predictions, in Step 1, Mask Transfiner corrects the nodes labels belonging to L1 level of the quadtree, and then propagates these corrected labels to their corresponding four quadrants in L2 level. In Step 2, the process of labels correction is efficiently conducted on the incoherent nodes in L2 and further propagating to L3. This process is recursive until reaching the finest quadtree level.

2. More Experimental Analysis

Accuracy Comparison In Table 9 of the main text, we compare the accuracy of Mask Transfiner with previous methods and find that Mask Transfiner achieves consistently large improvements on different backbones and object detectors. We further observe that the usage of DCN [16] with Mask Transfiner can bring a surge in performance. We compare Transfiner with Mask Scoring R-CNN [9] trained with DCN under the same setting and training schedules. Using ResNet-101 and Faster R-CNN [13] detector, the mask AP of Mask Transfiner on COCO *test-dev* is 42.2, while Mask Scoring R-CNN is 39.6 in Table 9 of the paper. For more comprehensive comparisons on two-stage instance segmentation methods, in Table 1, we also train Mask R-CNN [7], PointRend [11], BCNet [10], Cascade Mask R-CNN [1] and HTC [3] with the multi-scale 3× training schedule with DCN, and submit their predictions to the evaluation server for obtaining their accuracies on the test-dev split. The performance advantages of Mask Transfiner are consistently significant, improving the baseline Mask R-CNN† for 2.8 mask AP and outperforming PointRend by 0.9 AP.

Table 1. Performance comparison between two-stage instance segmentation methods on COCO *test-dev* set using R101-FPN. The dagger † denotes training with DCN [16] and 3× training schedule in our implementation. HTC and Cascade Mask R-CNN use 3-stage cascade refinement with multiple object detectors and mask heads. The standard transformer with output size 112×112 runs out of memory in our experiments.

Method	Output Size	AP	AP_S	AP_M	AP_L	FPS
Mask R-CNN† [7] (Baseline) [7]	28×28	39.4	18.6	42.8	54.5	**9.6**
Mask Scoring R-CNN† [9]	28×28	39.6	18.9	42.7	55.1	9.2
BCNet† [10]	28×28	41.2	23.6	43.9	52.8	8.9
PointRend† [11]	224×224	41.3	20.6	44.0	55.3	7.2
Cascade Mask R-CNN† [1]	28×28	41.5	22.1	42.6	54.2	4.8
HTC† [3]	28×28	41.7	23.3	44.2	53.8	2.1
Standard Transformer†	56×56	41.3	23.4	43.5	53.2	1.4
Mask Transfiner† (Ours: Quadtree Transformer)	112×112	**42.2**	**24.1**	**44.8**	**55.4**	6.1

Inference Speed We adopt frames per second (FPS) to evaluate the inference speed of the models. In Table 1, we benchmark all the compared two-stage methods using a Titan RTX GPU. The reported FPS is the average obtained in five runs, where each run measures the FPS of a model through 200 iterations. Compared to the Cascade Mask R-CNN and HTC with three-stage cascade refinement and multiple object detectors/mask heads (output size 28×28), our Transfiner using 3-level

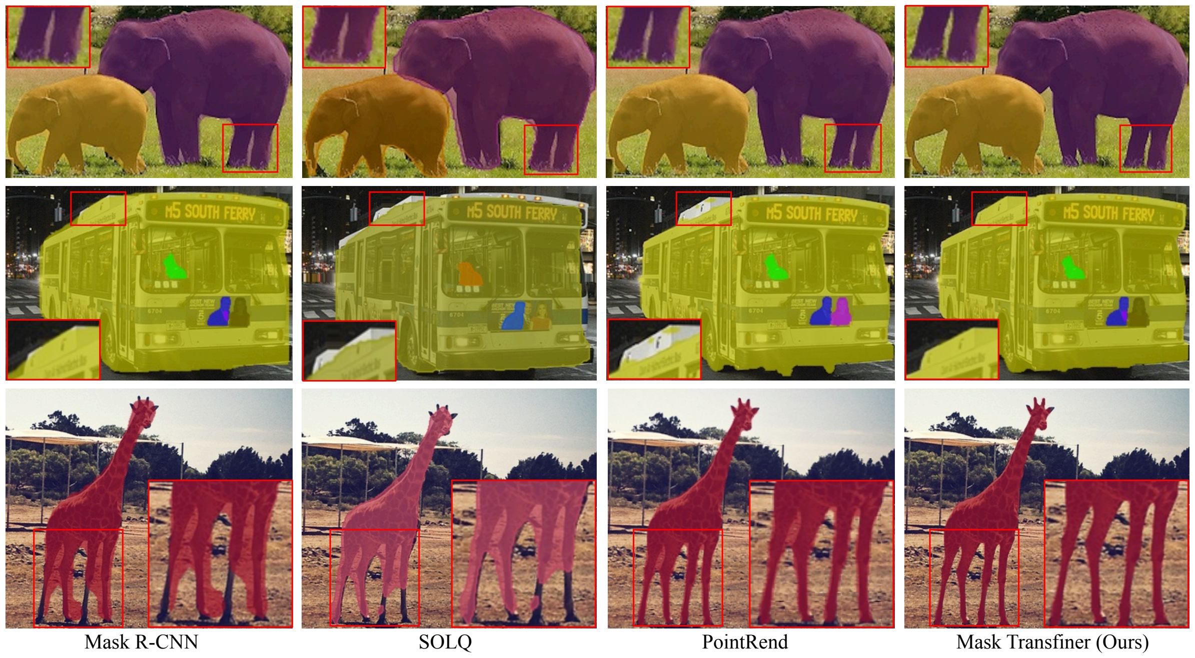

Figure 2. Instance Segmentation on COCO [12] validation set by a) Mask R-CNN [7], b) SOLQ [6], c) PointRend [11], d) Mask Transfiner (Ours) using R50-FPN as backbone, where Mask Transfiner produces significantly more detailed results at high-frequency image regions by replacing Mask R-CNN's default mask head. Zoom in for better view.

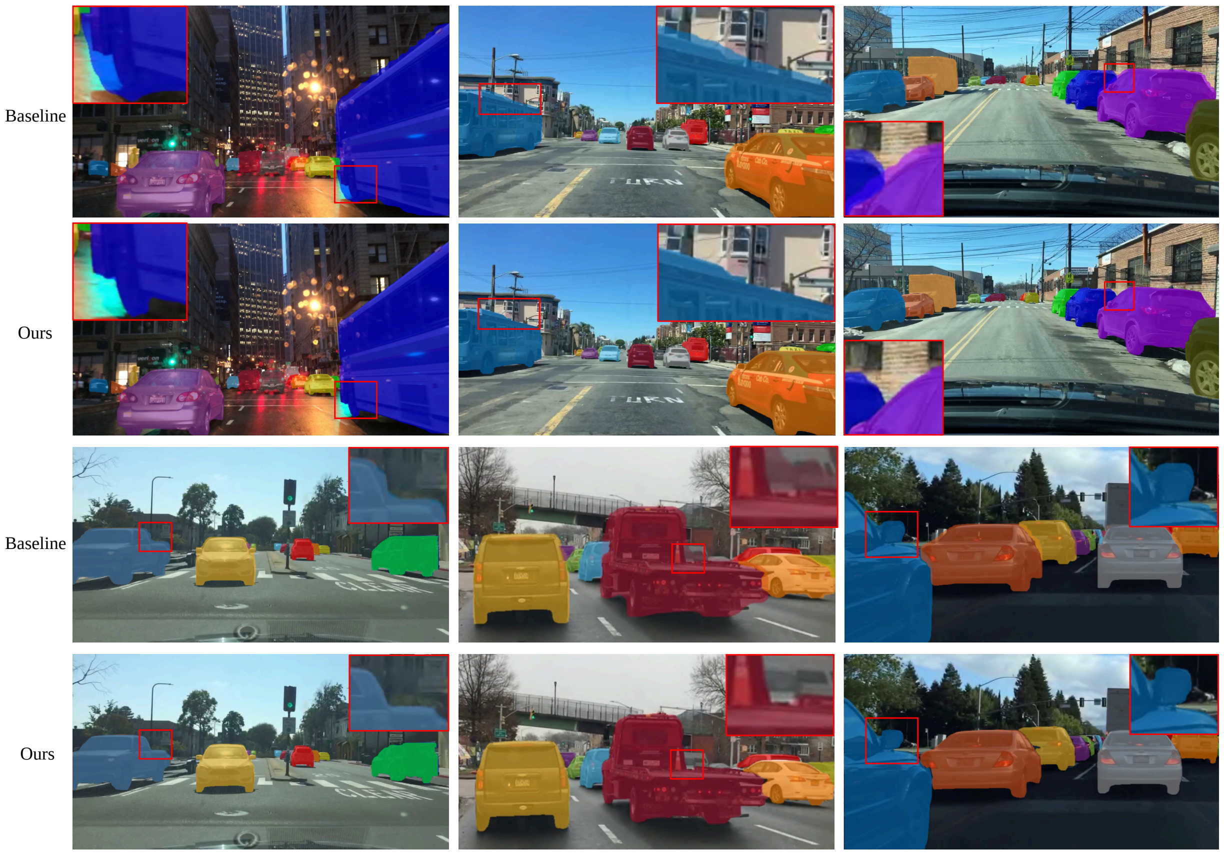

Figure 3. Qualitative comparisons with baseline method Mask R-CNN [7] and our Mask Transfiner on BDD100K [15] *val* set. Mask Transfiner produces more correct and natural segmentation results by revealing details for high-frequency regions.

quadtree is much faster and more accurate with higher-resolution predictions (112×112). Comparing to the baseline Mask R-CNN, although there is a drop on inference speed for about 35% due to multi-head attention modeling between hierarchical incoherent regions, the significant performance boost of 2.8 mask AP and 4 times larger output height/width are good compensation trade-offs. Note that standard transformer (3 layers and 4 attention heads in each layer) operating on uniform grids with output size 56×56 only runs at 1.4 FPS, which is much slower than our method.

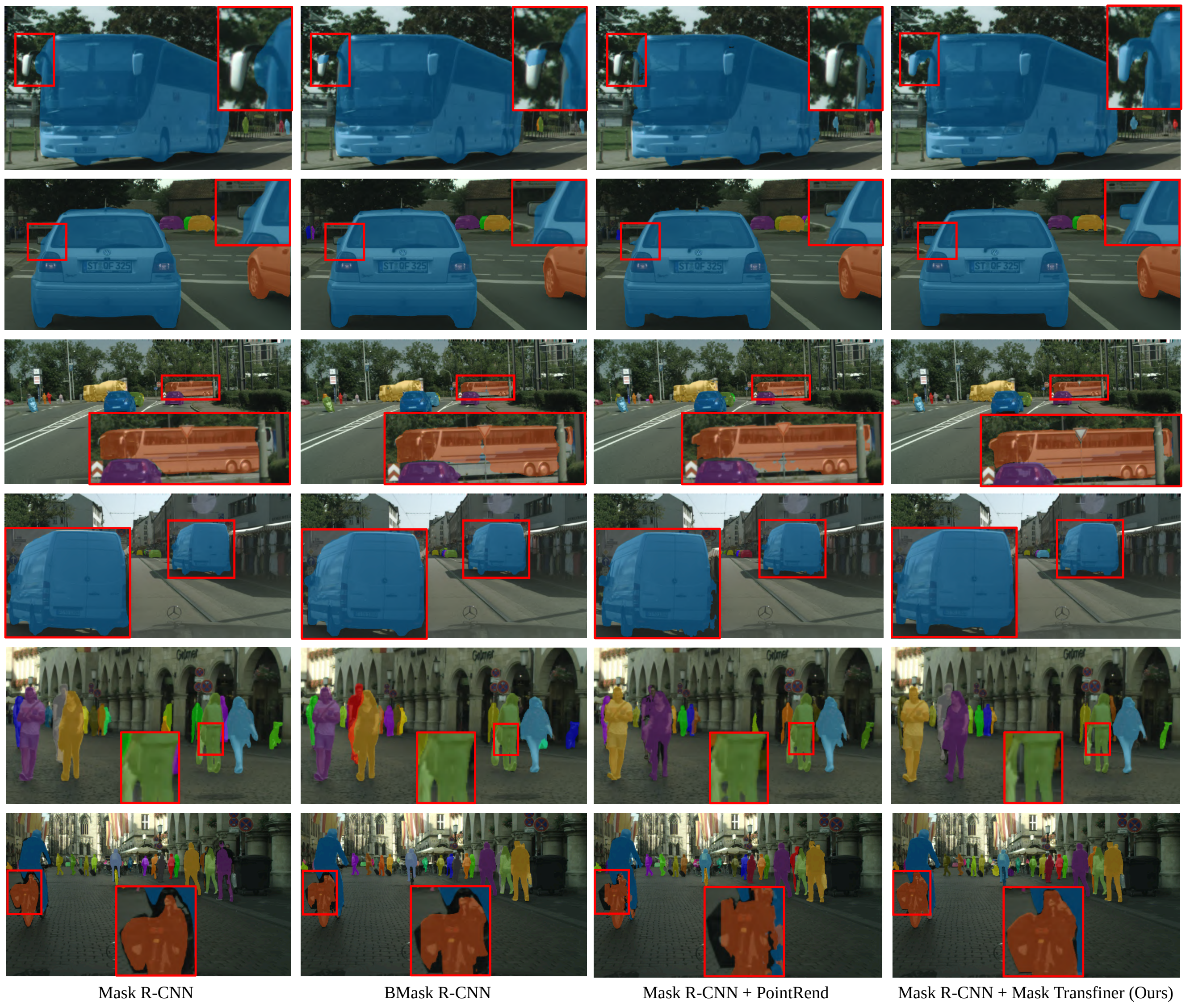

Figure 4. Qualitative comparisons with instance segmentation methods Mask R-CNN [7], BMask R-CNN [4], PointRend [11] and our Mask Transfiner on Cityscapes [5] *val* set. Mask Transfiner produces more precise and natural segmentation results, where even the small triangle-shaped traffic sign occluding the bus (3rd row) and the gap between the hand and leg (5th row) could be correctly separated. Zoom in for better view.

3. More Qualitative Comparisons

We provide more qualitative results comparisons on three evaluation benchmarks COCO (Figure 2), B100K (Figure 3) and Cityscapes (Figure 4), where our Mask Transfiner consistently produces masks with substantially higher precision and quality than previous methods [4, 6, 7, 11]. Take the third case in Figure 2 as an example, SOLQ and the baseline Mask R-CNN only provides very coarse mask predictions in the high-frequency regions, such as the giraffe's head and feet regions, due to the low-resolution output size 28×28. Although PointRend uses large output size 224×224, it still fails to delineate the thin gap between the left legs of giraffe. These segmentation errors on ambiguous regions reveals the limitation of segmenting each pixel separately only by a share MLP [11] without global reasoning and pixel-wise relations modeling.

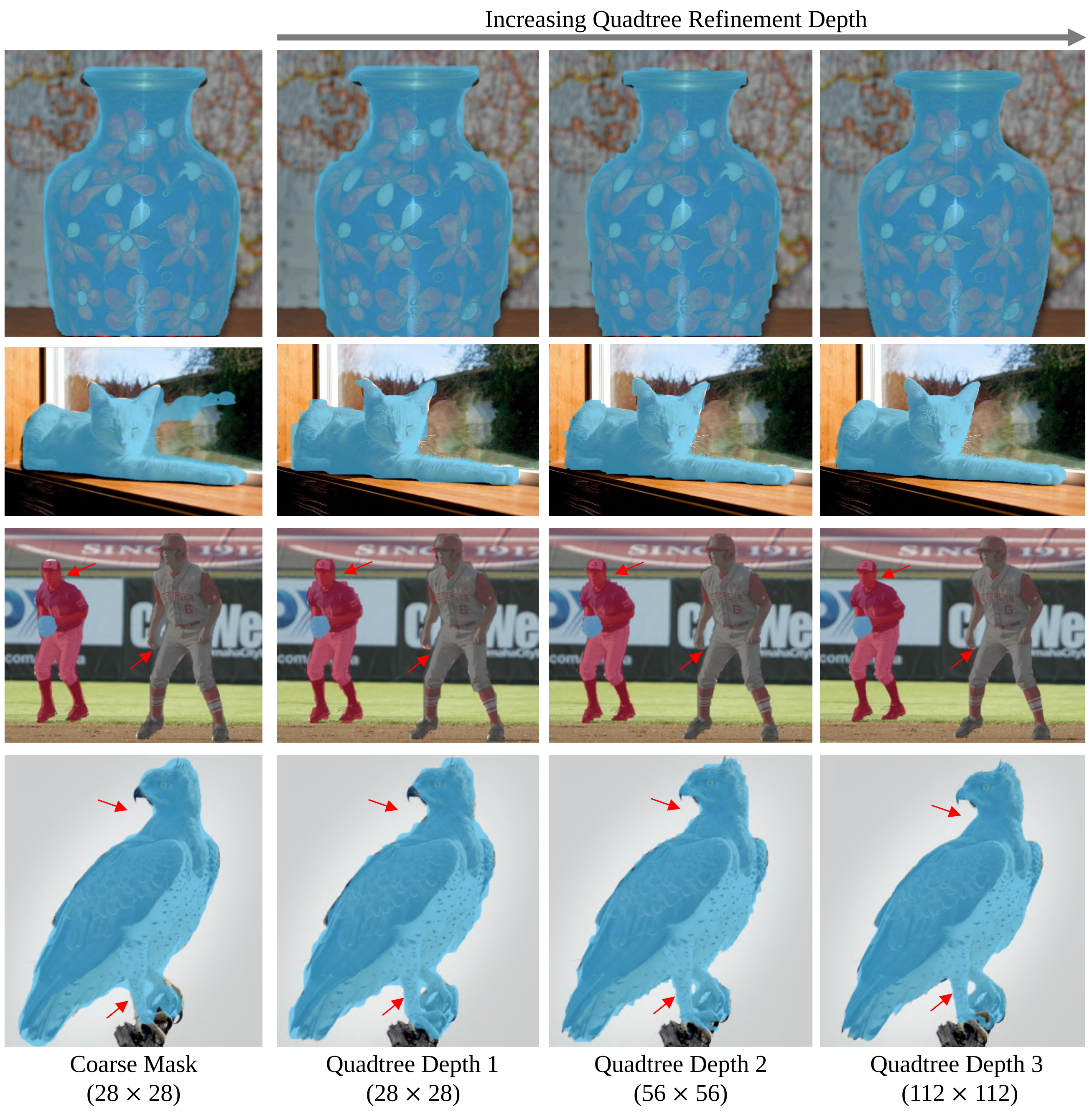

Figure 5. Qualitative results comparison between the coarse mask predictions by our baseline [8] and the refinement results of Mask Transfiner on COCO with various depths of the quadtree built on detected incoherent regions.

4. Visual Analysis

Visualization Multi-level Refinement In Figure 5, we analyze how the mask predictions evolve with increasing quadtree depths. The predicted masks become substantially finer in detail around object boundaries, which reveals that the quadtree nodes with more levels at larger output sizes for an object preserves more low-level details for fine-grained segmentation.

Failure Cases We also analyze the failure cases and find one typical failure mode shown in the last row of Figure 5, where a small portion of the bird's paw is wrongly predicted as background wood due to their highly similar appearance and texture.

Visualization on Quadtree Attention and Incoherent Regions In Figure 6 and Figure 7, we further provide more quadtree attention visualization examples and their detected incoherent regions on RoI pyramid, where the outline of objects can be observed and the sparsity of quadtree attention is clearly shown. The quadtree nodes with higher appearance or positional similarity has larger attention weights attending between them.

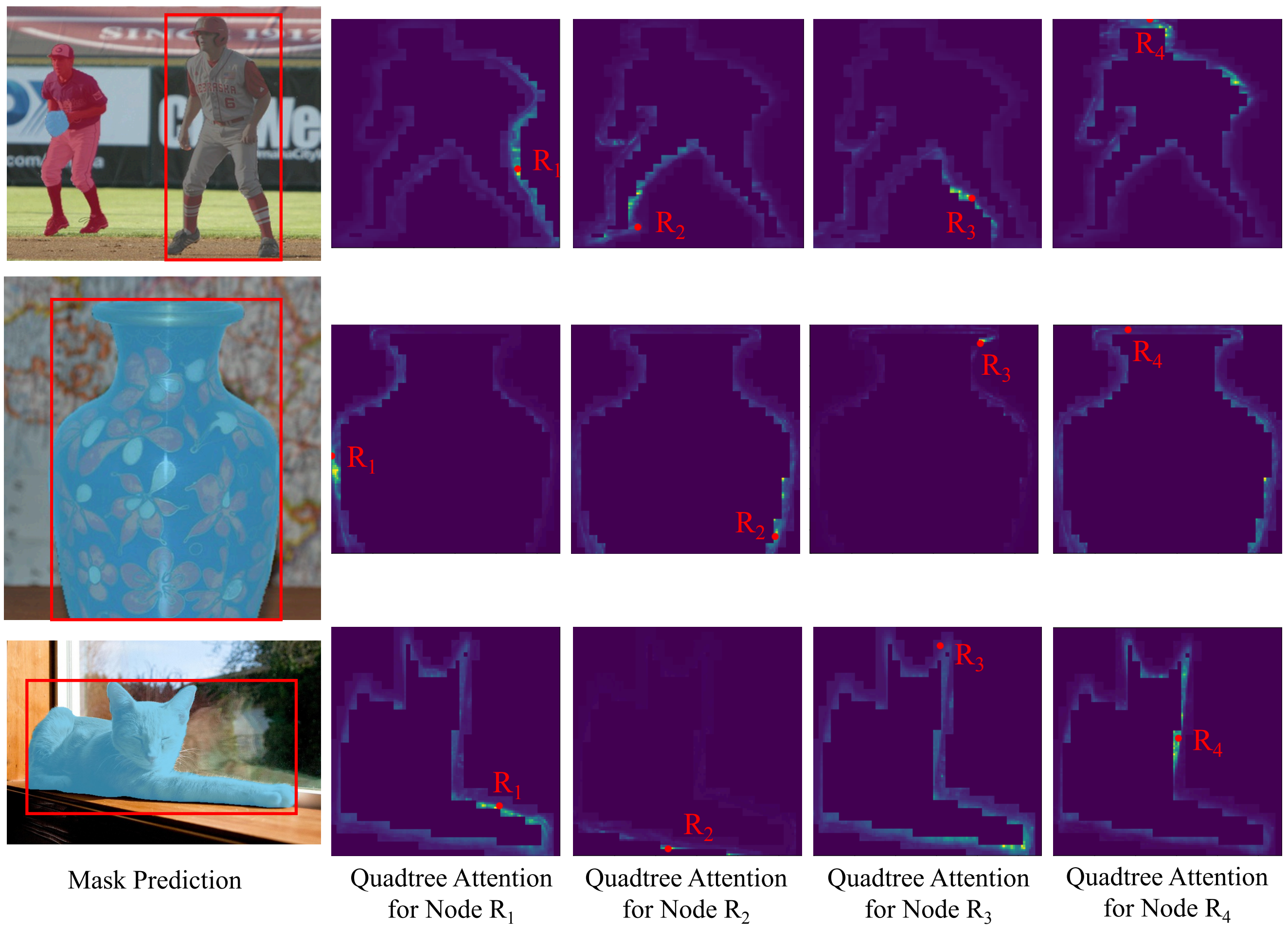

Figure 6. Visualization on the quadtree attention weights distribution in the sparse incoherent regions for four sampled red nodes.

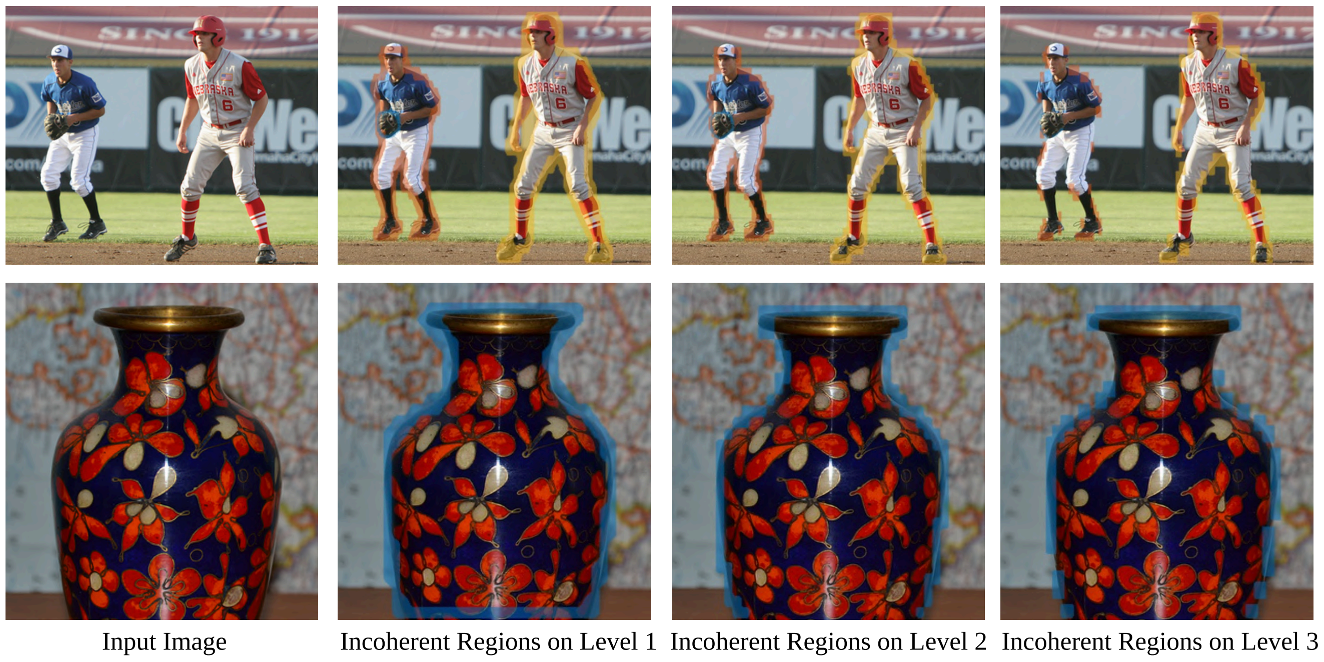
Input Image
Incoherent Regions on Level 1
Incoherent Regions on Level 2
Incoherent Regions on Level 3

Figure 7. Visualization of detected incoherent regions on different levels of the constructed quadtree based on the RoI pyramid, where the incoherent nodes regions in deeper quadtree levels with larger resolution size are distributed more sparsely.

References

[1] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. 2019. 2

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1

[3] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 2

[4] Tianheng Cheng, Xinggang Wang, Lichao Huang, and Wenyu Liu. Boundary-preserving mask r-cnn. In *ECCV*, 2020. 1, 4

[5] 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 *CVPR*, 2016. 1, 4

[6] Bin Dong, Fangao Zeng, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Solq: Segmenting objects by learning queries. In *NeurIPS*, 2021. 3, 4

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2, 3, 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[9] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019. 2

[10] Lei Ke, Yu-Wing Tai, and Chi-Keung Tang. Deep occlusion-aware instance segmentation with overlapping bilayers. In *CVPR*, 2021. 1, 2

[11] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 1, 2, 3, 4

[12] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1, 3

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2

[14] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 1

[15] 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 *CVPR*, 2020. 1, 3

[16] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019. 2