PointRend: Image Segmentation as Rendering

Alexander Kirillov Yuxin Wu Kaiming He Ross Girshick
Facebook AI Research (FAIR)

Abstract

We present a new method for efficient high-quality image segmentation of objects and scenes. By analogizing classical computer graphics methods for efficient rendering with over- and undersampling challenges faced in pixel labeling tasks, we develop a unique perspective of image segmentation as a rendering problem. From this vantage, we present the PointRend *(Point-based Rendering) neural network module: a module that performs point-based segmentation predictions at adaptively selected locations based on an iterative subdivision algorithm. PointRend can be flexibly applied to both instance and semantic segmentation tasks by building on top of existing state-of-the-art models. While many concrete implementations of the general idea are possible, we show that a simple design already achieves excellent results. Qualitatively, PointRend outputs crisp object boundaries in regions that are over-smoothed by previous methods. Quantitatively, PointRend yields significant gains on COCO and Cityscapes, for both instance and semantic segmentation. PointRend's efficiency enables output resolutions that are otherwise impractical in terms of memory or computation compared to existing approaches. Code has been made available at* https://github.com/facebookresearch/detectron2/tree/master/projects/PointRend.

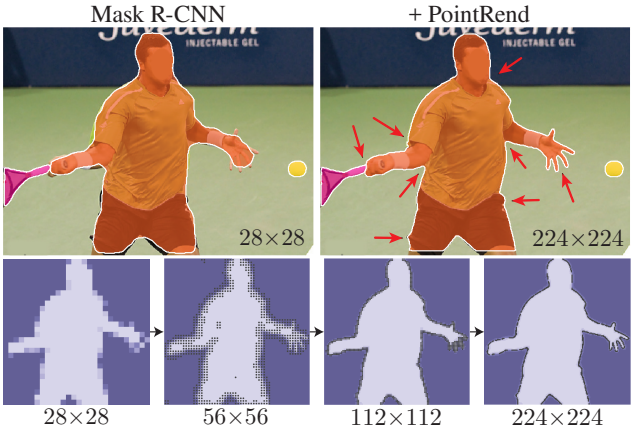

Figure 1: **Instance segmentation with PointRend.** We introduce the PointRend (Point-based Rendering) module that makes predictions at adaptively sampled points on the image using a new point-based feature representation (see Fig. 3). PointRend is general and can be flexibly integrated into existing semantic and instance segmentation systems. When used to replace Mask R-CNN's default mask head [19] (top-left), PointRend yields significantly more detailed results (top-right). (bottom) During inference, PointRend iterative computes its prediction. Each step applies bilinear upsampling in smooth regions and makes higher resolution predictions at a small number of adaptively selected points that are likely to lie on object boundaries (black points). All figures in the paper are best viewed digitally with zoom. Image source: [43].

1. Introduction

Image segmentation tasks involve mapping pixels sampled on a regular grid to a label map, or a set of label maps, on the same grid. For semantic segmentation, the label map indicates the predicted category at each pixel. In the case of instance segmentation, a binary foreground *vs*. background map is predicted for each detected object. The modern tools of choice for these tasks are built on convolutional neural networks (CNNs) [29, 28].

CNNs for image segmentation typically operate on regular grids: the input image is a regular grid of pixels, their hidden representations are feature vectors on a regular grid, and their outputs are label maps on a regular grid. Regular grids are convenient, but not necessarily computationally ideal for image segmentation. The label maps predicted by these networks should be mostly smooth, *i.e*., neighboring pixels often take the same label, because high-frequency regions are restricted to the sparse boundaries between objects. A regular grid will unnecessarily oversample the smooth areas while simultaneously undersampling object boundaries. The result is excess computation in smooth regions and blurry contours (Fig. 1, upper-left). Image segmentation methods often predict labels on a low-resolution regular grid, *e.g*., 1/8-th of the input [37] for semantic segmentation, or 28×28 [19] for instance segmentation, as a compromise between undersampling and oversampling.

Analogous sampling issues have been studied for decades in computer graphics. For example, a *renderer* maps a model (*e.g*., a 3D mesh) to a rasterized image, *i.e*. a

Figure 2: **Example result pairs from Mask R-CNN [19] with its standard mask head (left image) *vs*. with PointRend (right image)**, using ResNet-50 [20] with FPN [30]. Note how PointRend predicts masks with substantially finer detail around object boundaries.

regular grid of pixels. While the output is on a regular grid, computation is not allocated uniformly over the grid. Instead, a common graphics strategy is to compute pixel values at an *irregular* subset of adaptively selected *points* in the image plane. The classical *subdivision* technique of [50], as an example, yields a quadtree-like sampling pattern that efficiently renders an anti-aliased, high-resolution image.

The central idea of this paper is to view image segmentation as a rendering problem and to adapt classical ideas from computer graphics to efficiently "render" high-quality label maps (see Fig. 1, bottom-left). We encapsulate this computational idea in a new neural network module, called **PointRend**, that uses a subdivision strategy to adaptively select a non-uniform set of points at which to compute labels. PointRend can be incorporated into popular meta-architectures for both instance segmentation (*e.g.*, Mask R-CNN [19]) and semantic segmentation (*e.g.*, FCN [37]). Its subdivision strategy efficiently computes high-resolution segmentation maps using an order of magnitude fewer floating-point operations than direct, dense computation.

PointRend is a general module that admits many possible implementations. Viewed abstractly, a PointRend module accepts one or more typical CNN feature maps $f(x_i, y_i)$ that are defined over regular grids, and outputs high-resolution predictions $p(x'_i, y'_i)$ over a finer grid. Instead of making excessive predictions over all points on the output grid, PointRend makes predictions only on carefully selected points. To make these predictions, it extracts a point-wise feature representation for the selected points by interpolating f, and uses a small *point head* subnetwork to predict output labels from the point-wise features. We will present a simple and effective PointRend implementation.

We evaluate PointRend on instance and semantic segmentation tasks using the COCO [31] and Cityscapes [9] benchmarks. Qualitatively, PointRend efficiently computes sharp boundaries between objects, as illustrated in Fig. 2 and Fig. 8. We also observe quantitative improvements even though the standard intersection-over-union based metrics for these tasks (mask AP and mIoU) are biased towards object-interior pixels and are relatively insensitive to boundary improvements. PointRend improves strong Mask R-CNN and DeepLabV3 [5] models by a significant margin.

2. Related Work

Rendering algorithms in computer graphics output a regular grid of pixels. However, they usually compute these pixel values over a non-uniform set of points. Efficient procedures like subdivision [50] and adaptive sampling [40, 44] refine a coarse rasterization in areas where pixel values have larger variance. Ray-tracing renderers often use oversampling [52], a technique that samples some points more densely than the output grid to avoid aliasing effects. Here, we apply classical subdivision to image segmentation.

Non-uniform grid representations. Computation on regular grids is the dominant paradigm for 2D image analysis, but this is not the case for other vision tasks. In 3D shape recognition, large 3D grids are infeasible due to cubic scaling. Most CNN-based approaches do not go beyond coarse 64×64×64 grids [12, 8]. Instead, recent works consider more efficient non-uniform representations such as meshes [49, 14], signed distance functions [39], and octrees [48]. Similar to a signed distance function, PointRend can compute segmentation values at any point.

Recently, Marin *et al*. [38] propose an efficient semantic segmentation network based on non-uniform subsampling of the *input* image prior to processing with a standard semantic segmentation network. PointRend, in contrast, focuses on non-uniform sampling at the *output*. It may be possible to combine the two approaches, though [38] is currently unproven for instance segmentation.

Instance segmentation methods based on the Mask R-CNN meta-architecture [19] occupy top ranks in recent challenges [34, 3]. These region-based architectures typically predict masks on a 28×28 grid irrespective of object size. This is sufficient for small objects, but for large objects it produces undesirable "blobby" output that oversmooths the fine-level details of large objects (see Fig. 1, top-left). Alternative, bottom-up approaches group pixels to form object masks [33, 1, 25]. These methods can produce more detailed output, however, they lag behind region-based approaches on most instance segmentation benchmarks [31, 9, 42]. TensorMask [7], an alternative sliding-window method, uses a sophisticated network design to predict sharp high-resolution masks for large objects, but its accuracy also lags slightly behind. In this paper, we show that a region-based segmentation model equipped with PointRend can produce masks with fine-level details while improving the accuracy of region-based approaches.

Semantic segmentation. Fully convolutional networks (FCNs) [37] are the foundation of modern semantic segmentation approaches. They often predict outputs that have lower resolution than the input grid and use bilinear upsampling to recover the remaining 8-16× resolution. Results may be improved with dilated/atrous convolutions that replace some subsampling layers [4, 5] at the expense of more memory and computation.

Alternative approaches include encoder-decoder achitectures [6, 24, 46, 47] that subsample the grid representation in the encoder and then upsample it in the decoder, using skip connections [46] to recover filtered details. Current approaches combine dilated convolutions with an encoder-decoder structure [6, 32] to produce output on a 4× sparser grid than the input grid before applying bilinear interpolation. In our work, we propose a method that can efficiently predict fine-level details on a grid as dense as the input grid.

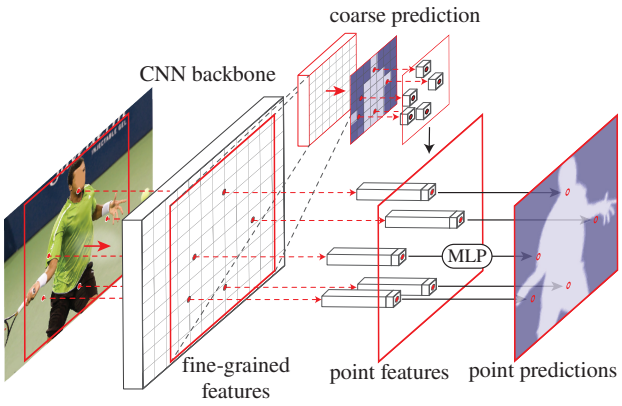

Figure 3: **PointRend applied to instance segmentation**. A standard network for instance segmentation (solid red arrows) takes an input image and yields a coarse (*e.g*. 7×7) mask prediction for each detected object (red box) using a lightweight segmentation head. To refine the coarse mask, PointRend selects a set of points (red dots) and makes prediction for each point independently with a small MLP. The MLP uses interpolated features computed at these points (dashed red arrows) from (1) a fine-grained feature map of the backbone CNN and (2) from the coarse prediction mask. The coarse mask features enable the MLP to make different predictions at a single point that is contained by two or more boxes. The proposed subdivision mask rendering algorithm (see Fig. 4 and §3.1) applies this process iteratively to refine uncertain regions of the predicted mask.

3. Method

We analogize image segmentation (of objects and/or scenes) in computer vision to image rendering in computer graphics. Rendering is about displaying a model (*e.g*., a 3D mesh) as a regular grid of pixels, *i.e*., an image. While the output representation is a regular grid, the underlying physical entity (*e.g*., the 3D model) is continuous and its physical occupancy and other attributes can be queried at any *real-value point* on the image plane using physical and geometric reasoning, such as ray-tracing.

Analogously, in computer vision, we can think of an image segmentation as the occupancy map of an underlying continuous entity, and the segmentation output, which is a regular grid of predicted labels, is "rendered" from it. The entity is encoded in the network's feature maps and can be accessed at any point by interpolation. A parameterized function, that is trained to predict occupancy from these interpolated point-wise feature representations, is the counterpart to physical and geometric reasoning.

Based on this analogy, we propose PointRend (*Point-based Rendering*) as a methodology for image segmentation using point representations. A PointRend module accepts one or more typical CNN feature maps of C channels $f \in \mathbb{R}^{C \times H \times W}$, each defined over a regular grid (that is typically 4× to 16× coarser than the image grid), and

outputs predictions for the K class labels $p \in \mathbb{R}^{K \times H' \times W'}$ over a regular grid of different (and likely higher) resolution. A PointRend module consists of three main components: (i) A *point selection strategy* chooses a small number of real-value points to make predictions on, avoiding excessive computation for all pixels in the high-resolution output grid. (ii) For each selected point, a *point-wise feature representation* is extracted. Features for a real-value point are computed by bilinear interpolation of f, using the point's 4 nearest neighbors that are on the regular grid of f. As a result, it is able to utilize sub-pixel information encoded in the channel dimension of f to predict a segmentation that has higher resolution than f. (iii) A *point head*: a small neural network trained to predict a label from this point-wise feature representation, independently for each point.

The PointRend architecture can be applied to instance segmentation (*e.g.*, on Mask R-CNN [19]) and semantic segmentation (*e.g.*, on FCNs [37]) tasks. For instance segmentation, PointRend is applied to each region. It computes masks in a coarse-to-fine fashion by making predictions over a set of selected points (see Fig. 3). For semantic segmentation, the whole image can be considered as a single region, and thus without loss of generality we will describe PointRend in the context of instance segmentation. We discuss the three main components in more detail next.

3.1. Point Selection for Inference and Training

At the core of our method is the idea of flexibly and adaptively selecting points in the image plane at which to predict segmentation labels. Intuitively, these points should be located more densely near high-frequency areas, such as object boundaries, analogous to the anti-aliasing problem in ray-tracing. We develop this idea for inference and training.

Inference. Our selection strategy for inference is inspired by the classical technique of *adaptive subdivision* [50] in computer graphics. The technique is used to efficiently render high resolutions images (*e.g.*, via ray-tracing) by computing only at locations where there is a high chance that the value is significantly different from its neighbors; for all other locations the values are obtained by interpolating already computed output values (starting from a coarse grid).

For each region, we iteratively "render" the output mask in a coarse-to-fine fashion. The coarsest level prediction is made on the points on a regular grid (*e.g.*, by using a standard coarse segmentation prediction head). In each iteration, PointRend upsamples its previously predicted segmentation using bilinear interpolation and then selects the N most uncertain points (*e.g.*, those with probabilities closest to 0.5 for a binary mask) on this denser grid. PointRend then computes the point-wise feature representation (described shortly in §3.2) for each of these N points and predicts their labels. This process is repeated until the segmentation is upsampled to a desired resolution. One step of this procedure is illustrated on a toy example in Fig. 4.

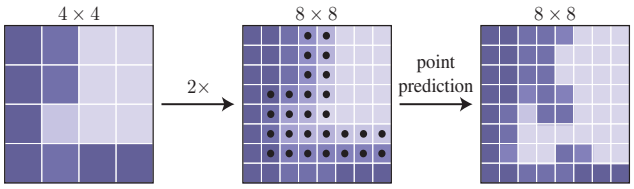

Figure 4: **Example of one adaptive subdivision step**. A prediction on a 4×4 grid is upsampled by 2× using bilinear interpolation. Then, PointRend makes prediction for the N most ambiguous points (black dots) to recover detail on the finer grid. This process is repeated until the desired grid resolution is achieved.

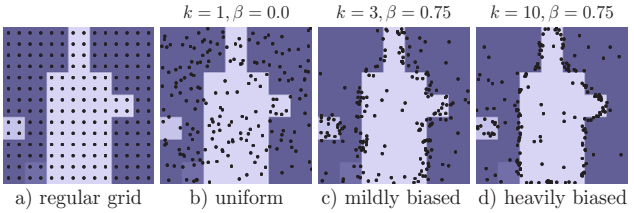

Figure 5: **Point sampling during training.** We show $N{=}14^2$ points sampled using different strategies for the same underlying coarse prediction. To achieve high performance only a small number of points are sampled per region with a mildly biased sampling strategy making the system more efficient during training.

With a desired output resolution of $M \times M$ pixels and a starting resolution of $M_0 \times M_0$, PointRend requires no more than $N \log_2 \frac{M}{M_0}$ point predictions. This is much smaller than $M \times M$, allowing PointRend to make high-resolution predictions much more effectively. For example, if M_0 is 7 and the desired resolutions is $M{=}224$, then 5 subdivision steps are preformed. If we select $N{=}28^2$ points at each step, PointRend makes predictions for only $28^2{\cdot}4.25$ points, which is 15 times smaller than 224^2. Note that fewer than $N \log_2 \frac{M}{M_0}$ points are selected overall because in the first subdivision step only 14^2 points are available.

Training. During training, PointRend also needs to select points at which to construct point-wise features for training the point head. In principle, the point selection strategy can be similar to the subdivision strategy used in inference. However, subdivision introduces sequential steps that are less friendly to training neural networks with backpropagation. Instead, for training we use a non-iterative strategy based on random sampling.

The sampling strategy selects N points on a feature map to train on.[1] It is designed to bias selection towards uncertain regions, while also retaining some degree of uniform coverage, using three principles. (i) *Over generation*: we over-generate candidate points by

[1]The value of N can be different for training and inference selection.

randomly sampling kN points ($k>1$) from a uniform distribution. (ii) *Importance sampling*: we focus on points with uncertain coarse predictions by interpolating the coarse prediction values at all kN points and computing a task-specific uncertainty estimate (defined in §4 and §5). The most uncertain βN points ($\beta \in [0, 1]$) are selected from the kN candidates. (iii) *Coverage*: the remaining $(1-\beta)N$ points are sampled from a uniform distribution. We illustrate this procedure with different settings, and compare it to regular grid selection, in Fig. 5.

At training time, predictions and loss functions are only computed on the N sampled points (in addition to the coarse segmentation), which is simpler and more efficient than backpropagation through subdivision steps. This design is similar to the parallel training of RPN + Fast R-CNN in a Faster R-CNN system [13], whose inference is sequential.

3.2. Point-wise Representation and Point Head

PointRend constructs *point-wise features* at selected points by combining (*e.g.*, concatenating) two feature types, fine-grained and coarse prediction features, described next.

Fine-grained features. To allow PointRend to render fine segmentation details we extract a feature vector at each sampled point from CNN feature maps. Because a point is a real-value 2D coordinate, we perform bilinear interpolation on the feature maps to compute the feature vector, following standard practice [22, 19, 10]. Features can be extracted from a single feature map (*e.g.*, res_2 in a ResNet); they can also be extracted from multiple feature maps (*e.g.*, res_2 to res_5, or their feature pyramid [30] counterparts) and concatenated, following the Hypercolumn method [17].

Coarse prediction features. The fine-grained features enable resolving detail, but are also deficient in two regards. First, they do not contain region-specific information and thus the same point overlapped by two instances' bounding boxes will have the same fine-grained features. Yet, the point can only be in the foreground of one instance. Therefore, for the task of instance segmentation, where different regions may predict different labels for the same point, additional region-specific information is needed.

Second, depending on which feature maps are used for the fine-grained features, the features may contain only relatively low-level information (*e.g.*, we will use res_2 with DeepLabV3). In this case, a feature source with more contextual and semantic information can be helpful. This issue affects both instance and semantic segmentation.

Based on these considerations, the second feature type is a coarse segmentation prediction from the network, *i.e.*, a K-dimensional vector at each point in the region (box) representing a K-class prediction. The coarse resolution, by design, provides more globalized context, while the channels convey the semantic classes. These coarse predictions are similar to the outputs made by the existing architectures, and are supervised during training in the same way as existing models. For instance segmentation, the coarse prediction can be, for example, the output of a lightweight 7×7 resolution mask head in Mask R-CNN. For semantic segmentation, it can be, for example, predictions from a stride 16 feature map.

Point head. Given the point-wise feature representation at each selected point, PointRend makes point-wise segmentation predictions using a simple multi-layer perceptron (MLP). This MLP shares weights across all points (and all regions), analogous to a graph convolution [23] or a PointNet [45]. Since the MLP predicts a segmentation label for each point, it can be trained by standard task-specific segmentation losses (described in §4 and §5).

4. Experiments: Instance Segmentation

Datasets. We use two standard instance segmentation datasets: COCO [31] and Cityscapes [9]. We report the standard mask AP metric [31] using the median of 3 runs for COCO and 5 for Cityscapes (it has higher variance).

COCO has 80 categories with instance-level annotation. We train on `train2017` (~118k images) and report results on `val2017` (5k images). As noted in [16], the COCO ground-truth is often coarse and AP for the dataset may not fully reflect improvements in mask quality. Therefore we supplement COCO results with AP measured using the 80 COCO category subset of LVIS [16], denoted by $\text{AP}^\star$. The LVIS annotations have significantly higher quality. Note that for $\text{AP}^\star$ we use the same models trained on COCO and simply re-evaluate their predictions against the higher-quality LVIS annotations using the LVIS evaluation API.

Cityscapes is an ego-centric street-scene dataset with 8 categories, 2975 train images, and 500 validation images. The images are higher resolution compared to COCO (1024×2048 pixels) and have finer, more pixel-accurate ground-truth instance segmentations.

Architecture. Our experiments use Mask R-CNN with a ResNet-50 [20] + FPN [30] backbone. The default mask head in Mask R-CNN is a region-wise FCN, which we denote by "4× conv".[2] We use this as our baseline for comparison. For PointRend, we make appropriate modifications to this baseline, as described next.

Lightweight, coarse mask prediction head. To compute the coarse prediction, we replace the 4× conv mask head with a lighter weight design that resembles Mask R-CNN's box head and produces a 7×7 mask prediction. Specifically, for each bounding box, we extract a 14×14 feature

[2] Four layers of 3×3 convolutions with 256 output channels are applied to a 14×14 input feature map. Deconvolution with a 2×2 kernel transforms this to 28×28. Finally, a 1×1 convolution predicts mask logits.

map from the P_2 level of the FPN using bilinear interpolation. The features are computed on a regular grid inside the bounding box (this operation can seen as a simple version of RoIAlign). Next, we use a stride-two 2×2 convolution layer with 256 output channels followed by ReLU [41], which reduces the spatial size to 7×7. Finally, similar to Mask R-CNN's box head, an MLP with two 1024-wide hidden layers is applied to yield a 7×7 mask prediction for each of the K classes. ReLU is used on the MLP's hidden layers and the sigmoid activation function is applied to its outputs.

PointRend. At each selected point, a K-dimensional feature vector is extracted from the coarse prediction head's output using bilinear interpolation. PointRend also interpolates a 256-dimensional feature vector from the P_2 level of the FPN. This level has a stride of 4 w.r.t. the input image. These coarse prediction and fine-grained feature vectors are concatenated. We make a K-class prediction at selected points using an MLP with 3 hidden layers with 256 channels. In each layer of the MLP, we supplement the 256 output channels with the K coarse prediction features to make the input vector for the next layer. We use ReLU inside the MLP and apply sigmoid to its output.

Training. We use the standard 1× training schedule and data augmentation from Detectron2 [51] by default (full details are in the appendix). For PointRend, we sample 14^2 points using the biased sampling strategy described in the §3.1 with k=3 and β=0.75. We use the distance between 0.5 and the probability of the ground truth class interpolated from the coarse prediction as the point-wise uncertainty measure. For a predicted box with ground-truth class c, we sum the binary cross-entropy loss for the c-th MLP output over the 14^2 points. The lightweight coarse prediction head uses the average cross-entropy loss for the mask predicted for class c, *i.e.*, the same loss as the baseline 4× conv head. We sum all losses without any re-weighting.

During training, Mask R-CNN applies the box and mask heads in parallel, while during inference they run as a cascade. We found that training as a cascade does not improve the baseline Mask R-CNN, but PointRend can benefit from it by sampling points inside more accurate boxes, slightly improving overall performance (~0.2% AP, absolute).

Inference. For inference on a box with predicted class c, unless otherwise specified, we use the adaptive subdivision technique to refine the coarse 7×7 prediction for class c to the 224×224 in 5 steps. At each step, we select and update (at most) the N=28^2 most uncertain points based on the absolute difference between the predictions and 0.5.

4.1. Main Results

We compare PointRend to the default 4× conv head in Mask R-CNN in Table 1. PointRend outperforms the default head on both datasets. The gap is larger when evaluating the COCO categories using the LVIS annotations (AP*) and for Cityscapes, which we attribute to the superior annotation quality in these datasets. Even with the same output resolution PointRend outperforms the baseline. The difference between 28×28 and 224×224 is relatively small because AP uses intersection-over-union [11] and, therefore, is heavily biased towards object-interior pixels and less sensitive to the boundary quality. Visually, however, the difference in boundary quality is obvious, see Fig. 6.

mask head	output resolution	COCO AP	COCO AP*	Cityscapes AP
4× conv	28×28	35.2	37.6	33.0
PointRend	28×28	36.1 (+0.9)	39.2 (+1.6)	35.5 (+2.5)
PointRend	224×224	**36.3** (+1.1)	**39.7** (+2.1)	**35.8** (+2.8)

Table 1: **PointRend *vs.* the default 4× conv mask head for Mask R-CNN [19].** Mask AP is reported. AP* is COCO mask AP evaluated against the higher-quality LVIS annotations [16] (see text for details). A ResNet-50-FPN backbone is used for both COCO and Cityscapes models. PointRend outperforms the standard 4× conv mask head both quantitively and qualitatively. Higher output resolution leads to more detailed predictions, see Fig. 2 and Fig. 6.

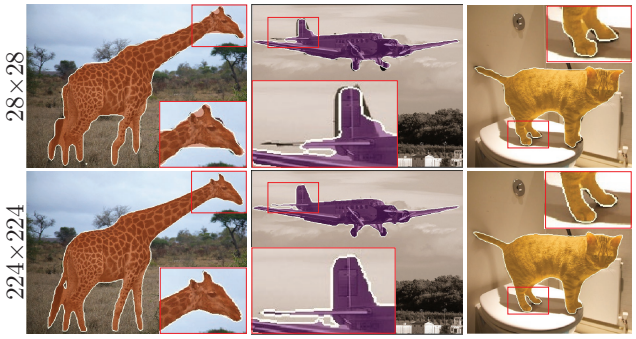

Figure 6: **PointRend inference with different output resolutions**. High resolution masks align better with object boundaries.

mask head	output resolution	FLOPs	# activations
4× conv	28×28	0.5B	0.5M
4× conv	224×224	34B	33M
PointRend	224×224	0.9B	0.7M

Table 2: **FLOPs (multiply-adds) and activation counts for a 224×224 output resolution mask.** PointRend's efficient subdivision makes 224×224 output feasible in contrast to the standard 4× conv mask head modified to use an RoIAlign size of 112×112.

Subdivision inference allows PointRend to yield a high resolution 224×224 prediction using more than 30 times less compute (FLOPs) and memory than the default 4× conv head needs to output the same resolution (based on taking a 112×112 RoIAlign input), see Table 2. PointRend makes high resolution output feasible in the Mask R-CNN framework by ignoring areas of an object where a coarse

output resolution	# points per subdivision step	COCO AP	COCO AP*	Cityscapes AP
28×28	28^2	36.1	39.2	35.4
56×56	28^2	36.2	39.6	35.8
112×112	28^2	36.3	39.7	35.8
224×224	28^2	36.3	39.7	35.8
224×224	14^2	36.1	39.4	35.5
224×224	28^2	36.3	39.7	35.8
224×224	56^2	36.3	39.7	35.8
224×224	112^2	36.3	39.7	35.8

Table 3: **Subdivision inference parameters.** Higher output resolution improves AP. Although improvements saturate quickly (at underlined values) with the number of points sampled at each subdivision step, qualitative results may continue to improve for complex objects. AP* is COCO mask AP evaluated against the higher-quality LVIS annotations [16] (see text for details).

Figure 7: **Anti-aliasing with PointRend.** Precise object delineation requires output mask resolution to match or exceed the resolution of the input image region that the object occupies.

prediction is sufficient (*e.g.*, in the areas far away from object boundaries). In terms of wall-clock runtime, our *unoptimized* implementation outputs 224×224 masks at ~13 fps, which is roughly the same frame-rate as a 4× conv head modified to output 56×56 masks (by doubling the default RoIAlign size), a design that actually has *lower* COCO AP compared to the 28×28 4× conv head (34.5% *vs*. 35.2%).

Table 3 shows PointRend subdivision inference with different output resolutions and number of points selected at each subdivision step. Predicting masks at a higher resolution can improve results. Though AP saturates, visual improvements are still apparent when moving from lower (*e.g.*, 56×56) to higher (*e.g.*, 224×224) resolution outputs, see Fig. 7. AP also saturates with the number of points sampled in each subdivision step because points are selected in the most ambiguous areas first. Additional points may make predictions in the areas where a coarse prediction is already sufficient. For objects with complex boundaries, however, using more points may be beneficial.

selection strategy	COCO AP	COCO AP*	Cityscapes AP
regular grid	35.7	39.1	34.4
uniform (k=1, β=0.0)	35.9	39.0	34.5
mildly biased (k=3, β=0.75)	**36.3**	**39.7**	**35.8**
heavily biased (k=10, β=1.0)	34.4	37.5	34.1

Table 4: **Training-time point selection strategies** with 14^2 points per box. Mildly biasing sampling towards uncertain regions performs the best. Heavily biased sampling performs even worse than uniform or regular grid sampling indicating the importance of coverage. AP* is COCO mask AP evaluated against the higher-quality LVIS annotations [16] (see text for details).

mask head	backbone	COCO AP	COCO AP*
4× conv	R50-FPN	37.2	39.5
PointRend	R50-FPN	**38.2** (+1.0)	**41.5** (+2.0)
4× conv	R101-FPN	38.6	41.4
PointRend	R101-FPN	**39.8** (+1.2)	**43.5** (+2.1)
4× conv	X101-FPN	39.5	42.1
PointRend	X101-FPN	**40.9** (+1.4)	**44.9** (+2.8)

Table 5: **Larger models and a longer 3× schedule [18].** PointRend benefits from more advanced models and the longer training. The gap between PointRend and the default mask head in Mask R-CNN holds. AP* is COCO mask AP evaluated against the higher-quality LVIS annotations [16] (see text for details).

4.2. Ablation Experiments

We conduct a number of ablations to analyze PointRend. In general we note that it is robust to the exact design of the point head MLP. Changes of its depth or width do not show any significant difference in our experiments.

Point selection during training. During training we select 14^2 points per object following the biased sampling strategy (§3.1). Sampling only 14^2 points makes training computationally and memory efficient and we found that using more points does not improve results. Surprisingly, sampling only 49 points per box still maintains AP, though we observe an increased variance in AP.

Table 4 shows PointRend performance with different selection strategies during training. Regular grid selection achieves similar results to uniform sampling. Whereas biasing sampling toward ambiguous areas improves AP. However, a sampling strategy that is biased too heavily towards boundaries of the coarse prediction (k>10 and β close to 1.0) decreases AP. Overall, we find a wide range of parameters $2<k<5$ and $0.75<\beta<1.0$ delivers similar results.

Larger models, longer training. Training ResNet-50 + FPN (denoted R50-FPN) with the 1× schedule under-fits on COCO. In Table 5 we show that the PointRend improvements over the baseline hold with both longer training schedule and larger models (see the appendix for details).

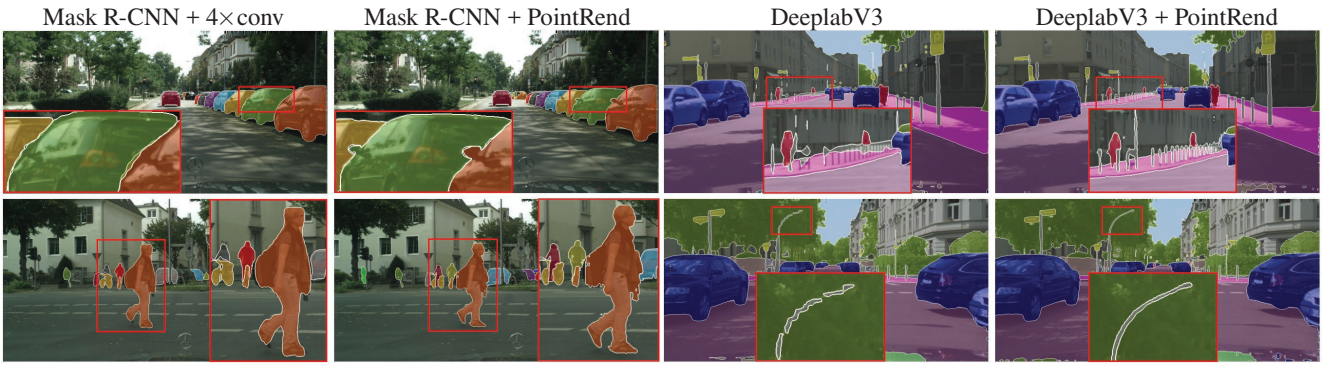

Figure 8: **Cityscapes example results for instance and semantic segmentation.** In instance segmentation larger objects benefit more from PointRend ability to yield high resolution output. Whereas for semantic segmentation PointRend recovers small objects and details.

5. Experiments: Semantic Segmentation

PointRend is not limited to instance segmentation and can be extended to other pixel-level recognition tasks. Here, we demonstrate that PointRend can benefit two semantic segmentation models: DeeplabV3 [5], which uses dilated convolutions to make prediction on a denser grid, and SemanticFPN [24], a simple encoder-decoder architecture.

Dataset. We use the Cityscapes [9] semantic segmentation set with 19 categories, 2975 training images, and 500 validation images. We report the median mIoU of 5 trials.

Implementation details. We reimplemented DeeplabV3 and SemanticFPN following their respective papers. SemanticFPN uses a standard ResNet-101 [20], whereas DeeplabV3 uses the ResNet-103 proposed in [5].[3] We follow the original papers' training schedules and data augmentation (details are in the appendix).

We use the same PointRend architecture as for instance segmentation. Coarse prediction features come from the (already coarse) output of the semantic segmentation model. Fine-grained features are interpolated from res_2 for DeeplabV3 and from P_2 for SemanticFPN. During training we sample as many points as there are on a stride 16 feature map of the input (2304 for deeplabV3 and 2048 for SemanticFPN). We use the same $k{=}3$, $\beta{=}0.75$ point selection strategy. During inference, subdivision uses $N{=}8096$ (*i.e.*, the number of points in the stride 16 map of a 1024×2048 image) until reaching the input image resolution. To measure prediction uncertainty we use the same strategy during training and inference: the difference between the most confident and second most confident class probabilities.

DeeplabV3. In Table 6 we compare DeepLabV3 to DeeplabV3 with PointRend. The output resolution can also be increased by 2× at inference by using dilated convolutions in res_4 stage, as described in [5]. Compared to both, PointRend has higher mIoU. Qualitative improvements are also evident, see Fig. 8. By sampling points adaptively, PointRend reaches 1024×2048 resolution (*i.e.* 2M points) by making predictions for only 32k points, see Fig. 9.

[3] It replaces the ResNet-101 res_1 7×7 convolution with three 3×3 convolutions (hence "ResNet-103").

method	output resolution	mIoU
DeeplabV3-OS-16	64×128	77.2
DeeplabV3-OS-8	128×256	77.8 (+0.6)
DeeplabV3-OS-16 + PointRend	1024×2048	**78.4** (+1.2)

Table 6: **DeeplabV3 with PointRend** for Cityscapes semantic segmentation outperforms baseline DeepLabV3. Dilating the res_4 stage during inference yields a larger, more accurate prediction, but at much higher computational and memory costs; it is still inferior to using PointRend.

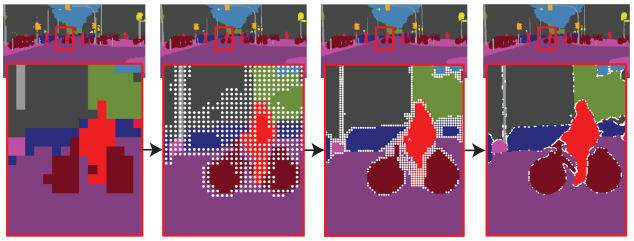

Figure 9: **PointRend inference for semantic segmentation.** PointRend refines prediction scores for areas where a coarser prediction is not sufficient. To visualize the scores at each step we take arg max at given resolution without bilinear interpolation.

method	output resolution	mIoU
SemanticFPN P_2-P_5	256×512	77.7
SemanticFPN P_2-P_5 + PointRend	1024×2048	**78.6** (+0.9)
SemanticFPN P_3-P_5	128×256	77.4
SemanticFPN P_3-P_5 + PointRend	1024×2048	**78.5** (+1.1)

Table 7: **SemanticFPN with PointRend** for Cityscapes semantic segmentation outperform the baseline SemanticFPN.

SemanticFPN. Table 7 shows that SemanticFPN with PointRend improves over both 8× and 4× output stride variants without PointRend.

Appendix A. Instance Segmentation Details

We use SGD with 0.9 momentum; a linear learning rate warmup [15] over 1000 updates starting from a learning rate of 0.001 is applied; weight decay 0.0001 is applied; horizontal flipping and scale train-time data augmentation; the batch normalization (BN) [21] layers from the ImageNet pre-trained models are frozen (*i.e.*, BN is not used); no test-time augmentation is used.

COCO [31]: 16 images per mini-batch; the training schedule is 60k / 20k / 10k updates at learning rates of 0.02 / 0.002 / 0.0002 respectively; training images are resized randomly to a shorter edge from 640 to 800 pixels with a step of 32 pixels and inference images are resized to a shorter edge size of 800 pixels.

Cityscapes [9]: 8 images per mini-batch the training schedule is 18k / 6k updates at learning rates of 0.01 / 0.001 respectively; training images are resized randomly to a shorter edge from 800 to 1024 pixels with a step of 32 pixels and inference images are resized to a shorter edge size of 1024 pixels.

Longer schedule: The 3× schedule for COCO is 210k / 40k / 20k updates at learning rates of 0.02 / 0.002 / 0.0002, respectively; all other details are the same as the setting described above.

Appendix B. Semantic Segmentation Details

DeeplabV3 [5]: We use SGD with 0.9 momentum with 16 images per mini-batch cropped to a fixed 768×768 size; the training schedule is 90k updates with a poly learning rate [36] update strategy, starting from 0.01; a linear learning rate warmup [15] over 1000 updates starting from a learning rate of 0.001 is applied; the learning rate for ASPP and the prediction convolution are multiplied by 10; weight decay of 0.0001 is applied; random horizontal flipping and scaling of 0.5× to 2.0× with a 32 pixel step is used as training data augmentation; BN is applied to 16 images mini-batches; no test-time augmentation is used;

SemanticFPN [24]: We use SGD with 0.9 momentum with 32 images per mini-batch cropped to a fixed 512×1024 size; the training schedule is 40k / 15k / 10k updates at learning rates of 0.01 / 0.001 / 0.0001 respectively; a linear learning rate warmup [15] over 1000 updates starting from a learning rate of 0.001 is applied; weight decay 0.0001 is applied; horizontal flipping, color augmentation [35], and crop bootstrapping [2] are used during training; scale train-time data augmentation resizes an input image from 0.5× to 2.0× with a 32 pixel step; BN layers are frozen (*i.e.*, BN is not used); no test-time augmentation is used.

method	trimap mIoU 8px	trimap mIoU 20px	mIoU
DeeplabV3-OS-16	42.4	57.5	77.2
DeeplabV3-OS-16 + PointRend	**47.3** (+4.9)	**61.2** (+3.7)	**78.4** (+1.2)
SemanticFPN P_2-P_5	47.0	60.6	77.7
SemanticFPN P_2-P_5 + PointRend	**48.6** (+1.6)	**62.1** (+1.5)	**78.6** (+0.9)

Table 8: Cityscapes mIoU within trimaps of different pixel widths. PointRend significantly improves segmentation quality around boundaries as the difference is larger for narrower trimaps.

Appendix C. Semantic Segmentation Boundary Quality

Intersection-over-union (IoU) [11] is heavily biased towards object-interior pixels and less sensitive to the boundary quality. The common approach to evaluate segmentation accuracy around boundaries is to calculate IoU for a "trimap", a narrow band surrounding segment boundaries [27, 38, 4, 26]. In Table 8 we compare mIoU for trimaps of different pixel widths for models with and without PointRend for semantic segmentation on Cityscapes. We confirm that PointRend boosts boundaries quality as the improvement is larger for narrow trimaps.

Appendix D. AP* Computation

The first version (v1) of this paper on arXiv has an error in COCO mask AP evaluated against the LVIS annotations [16] (AP*). The old version used an incorrect list of the categories not present in each evaluation image, which resulted in lower AP* values.

References

[1] Anurag Arnab and Philip HS Torr. Pixelwise instance segmentation with a dynamically instantiated network. In *CVPR*, 2017. 3

[2] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. In-place activated batchnorm for memory-optimized training of DNNs. In *CVPR*, 2018. 9

[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. 3

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *PAMI*, 2018. 3, 9

[5] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 2, 3, 8, 9

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 3

[7] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. TensorMask: A foundation for dense object segmentation. In *ICCV*, 2019. 3

[8] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *ECCV*, 2016. 3

[9] 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. 2, 3, 5, 8, 9

[10] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017. 5

[11] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes challenge: A retrospective. *IJCV*, 2015. 6, 9

[12] Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *ECCV*, 2016. 3

[13] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 5

[14] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019. 3

[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv:1706.02677*, 2017. 9

[16] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *ICCV*, 2019. 5, 6, 7, 9

[17] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 5

[18] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019. 7

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3, 4, 5, 6

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 5, 8

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 9

[22] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 5

[23] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ICLR*, 2017. 5

[24] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019. 3, 8, 9

[25] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. InstanceCut: from edges to instances with multicut. In *CVPR*, 2017. 3

[26] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *IJCV*, 2009. 9

[27] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 9

[28] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[29] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989. 1

[30] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 2, 5

[31] 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. 2, 3, 5, 9

[32] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *CVPR*, 2019. 3

[33] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. SGN: Sequential grouping networks for instance segmentation. In *CVPR*, 2017. 3

[34] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018. 3

[35] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *ECCV*, 2016. 9

[36] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv:1506.04579*, 2015. 9

[37] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1, 2, 3, 4

[38] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *ICCV*, 2019. 3, 9

[39] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 3

[40] Don P Mitchell. Generating antialiased images at low sampling densities. *ACM SIGGRAPH Computer Graphics*, 1987. 2

[41] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010. 6

[42] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *CVPR*, 2017. 3

[43] Paphio. Jo-Wilfried Tsonga [19]. CC BY-NC-SA 2.0. https://www.flickr.com/photos/paphio/2855627782/, 2008. 1

[44] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*, chapter 7. Morgan Kaufmann, 2016. 2

[45] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 5

[46] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 3

[47] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *arXiv:1904.04514*, 2019. 3

[48] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3D outputs. In *ICCV*, 2017. 3

[49] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 3

[50] Turner Whitted. An improved illumination model for shaded display. In *ACM SIGGRAPH Computer Graphics*, 1979. 2, 4

[51] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[52] Kun Zhou, Qiming Hou, Rui Wang, and Baining Guo. Real-time kd-tree construction on graphics hardware. In *ACM Transactions on Graphics (TOG)*, 2008. 2