A Dataset of Multi-Illumination Images in the Wild

A. Crowd-sourced data annotation

For each scene, we include dense material labels, segmented by crowd workers. These annotations are inspired by the material annotations collected by Bell *et al.* [1], whose publicly available source code forms the basis for our data annotation system. Departing from Bell *et al.*, we strive to densely label each scene with at least 95% coverage. In comparison, the images published by Bell *et al.* have an average coverage of 20%.

Annotations with above 95% coverage also set our dataset apart from semantic segmentation datasets such as Coco [3] and Pascal [2], which generally exhibit many unlabeled background pixels. In particular, the Coco 2014 training set has an average pixel-level coverage of 29.6%, while Pascal VOC2012 includes annotations for 25.5% of the pixels.

In early crowd sourcing experiments, we found it difficult to achieve high annotation coverage using the polygon-based segmentation user interface from Bell *et al.* [1], which also forms the backbone of the Coco annotations [3]. While segmentation using a single polyline is an effective solution for foreground objects without holes, encouraging a thorough labeling of the background turned out more difficult. We found that background regions must frequently be split into several polygons to avoid accidentally including foreground objects. Further, we observed that labeling both foreground and background would almost double the work required, as each occluding contour must be traced twice.

We overcome these limitations by segmenting objects in order, from front to back. In a typical segmentation session, a worker starts by labeling foreground objects that are not occluded (i.e. objects closest to the camera). Once these occluding objects are labeled, the worker can "extract" them from the image. As background polygons are automatically masked by extracted foreground shapes, the worker can then segment the "second layer" without worrying that their newest polygon might overlap previously segmented areas. Our front-to-back segmentation interface is efficient since occluding contours only have to be traced once, background objects are segmented into a single contiguous polygon, and the front-to-back logic ensures there are neither gaps nor overlap between foreground and background annotations.

We found that crowd workers on Amazon Mechanical Turk were able to reliably segment scenes in front-to-back order after viewing a short tutorial video that introduces our user interface and demonstrates the semantics of front-to-back ordering.

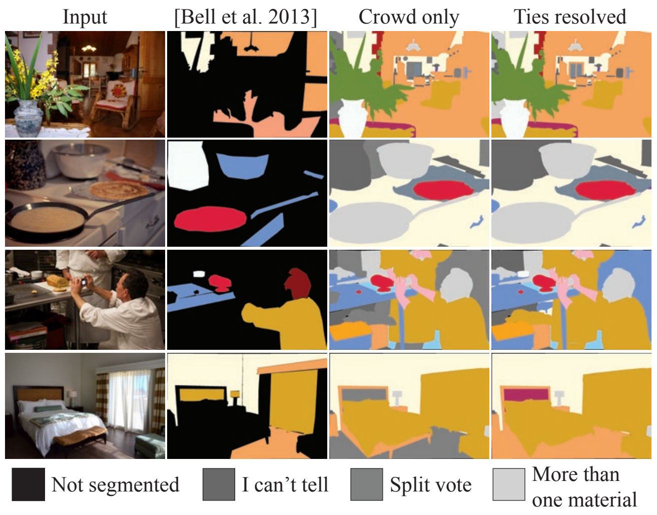

Figure 1. Obtaining full-image annotations using an unmodified version of [1] is challenging (column 2). Our modified version of their UI lets workers segment images front-to-back, which allows background objects to be captured in a single contiguous shape (column 3). In some cased (see wooden floor in bottom row), crowd-sourced votes are split, which we resolve manually for the final result (column 4).

After the segmented shapes are submitted to our server, we add them to a work queue where workers are asked to choose the material for each segment. The choice of material categories and annotation interface follows Bell *et al.* [1]. Each shape is presented to five workers and materials are determined by a simple majority vote. In cases where no material receives a majority, we resolve ties manually. Figure 1 compares the segmentations obtained using our technique with thouse obtained using the original user interface.

B. Network architectures

Let C_k denote a Conv–ReLU layer with k 3×3 kernels. P_k is a $k \times k$ max-pooling operator, and L_k is a 1×1 convolution with k linear outputs (no activation). U_k is a $k \times k$ bilinear upsampling layer.

Illumination prediction Our fully convolutional illumination prediction network takes 256×256 color images as input and produces 16×16 RGB light probe images (i.e. 768 output floats). Its architecture is given by: $C_{32}P_2C_{64}P_2C_{128}P_2C_{256}P_2C_{512}P_4DC_{512}P_4C_{512}L_{768}$.

Relighting and white-balance Both the relighting and white-balance application use a U-net with 7 downsampling stages and take 3-channel images as input. The relighting output is a color image (3 channels), but the white-balance model produces only the chroma components (2 channels), which are then combined with the input luminance. The U-net encoder can be described as:

$(C_{64})^2 P_2 (C_{128})^2 P_2 (C_{256})^2 P_2 (C_{512})^2 P_2 (C_{512})^2 P_2$ $(C_{512})^2 P_2 (C_{512})^2 P_2 (C_{512})^2$.

And the decoder is given by:

$U_2 (C_{64})^2 U_2 (C_{128})^2 U_2 (C_{256})^2 U_2 (C_{512})^2 U_2$ $(C_{512})^2 U_2 (C_{512})^2 U_2 (C_{512})^2 L_{2\text{or}3}$, with additive skip-connections between matching resolutions.

References

[1] S. Bell, P. Upchurch, N. Snavely, and K. Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM SIGGRAPH*, 2013. 1

[2] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010. 1

[3] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 1