Bundle Pooling for Polygonal Architecture Segmentation Problem

Huayi Zeng[1] Kevin Joseph[2] Adam Vest[3] Yasutaka Furukawa[2]
[1]Washington University in St. Louis [2]Simon Fraser University [3]Nike

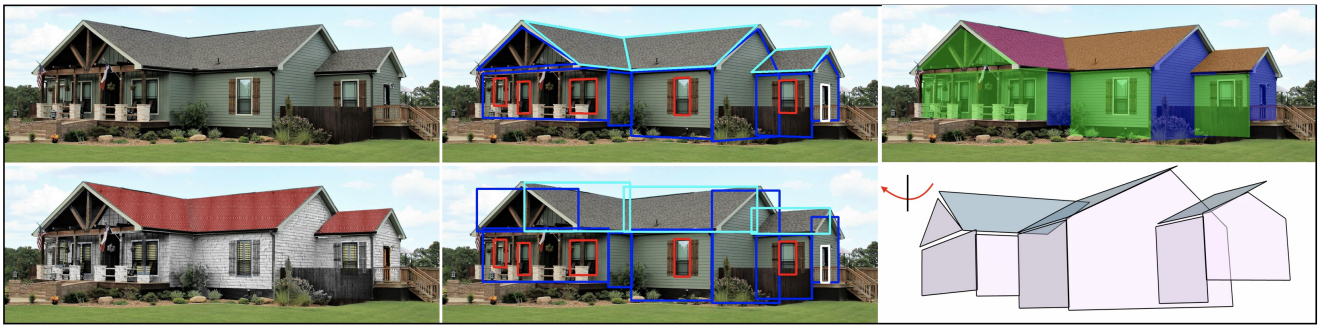

Figure 1. Complex house structures can be parsed as a combination of simple primitives. Our system turns a photograph of a residential house (top left) into an assembly of simple polygonal primitives (top middle), while a standard detection system produces only axis-aligned bounding boxes (bottom middle). The polygonal representation enables applications such as photo-realistic virtual remodeling (bottom left), surface normal estimation (top right) or pop-up 3D billboards (bottom right).

Abstract

This paper introduces a polygonal architecture segmentation problem, proposes bundle-pooling modules for line structure reasoning, and demonstrates a virtual remodeling application that produces production quality results. Given a photograph of a house with a few vanishing point candidates, we decompose the house into a set of architectural components, each of which is represented as a simple geometric primitive. A bundle-pooling module pools convolutional features along a bundle of line segments (e.g., a family of vanishing lines) and fuses the bundle of features to determine polygonal boundaries or assign a corresponding vanishing point. Qualitative and quantitative evaluations demonstrate significant improvements over the existing techniques based on our metric and benchmark dataset. We will share the code for further research.

1. Introduction

Houses are comprised of architectural components such as walls, windows, and roofs. Human vision effortlessly perceives a structure as a combination of simple polygonal shapes such as triangles, quadrilaterals, or trapezoids (See Fig. 1). This representation allows for easy human post-processing as well as virtual remodeling applications such as replacing wall materials or adding a new window.

Image segmentation has seen breakthroughs in recent years thanks to the emergence of deep neural networks (DNNs). A state-of-the-art technique, Mask R-CNN [13], can detect object instances and produce pixel-wise segmentation masks for each instance. However, it fails to delineate the boundaries precisely or represent the shape as a compact polygon, which is critical for high-end graphics applications such as virtual remodeling [1]. The task of extracting compact polygonal segments is a trivial task for human perception but remains difficult for computer vision.

Towards robust polygonal image segmentation, this paper proposes a new polygonal architecture segmentation problem. The input is a photograph of a house and a number of vanishing point candidates. The task is to decompose the house into architectural components (i.e., a wall, a window, a door, a garage, or a roof), each of which is represented as a simple geometric primitive (i.e., a triangle, a quadrilateral, a pentagon, or a trapezoid).

We tackle the problem by proposing a family of novel neural modules that analyze a bundle of lines. Specifically, the paper proposes three *bundle pooling* (BP) modules, each of which (1) considers a bundle of line segments; 2) pools convolutional features along the bundle; and (3) fuses the bundle of pooled features to determine polygonal boundaries or assign a corresponding vanishing point. For instance, given a bounding box containing a window, we consider a bundle of vanishing lines for each vanishing point and infer its polygonal boundaries.

We introduce a new dataset including the ground-truth architectural polygon segmentation, and a metric for evaluations. The qualitative and quantitative evaluations demon-

strate the effectiveness of our approach, significantly outperforming the current state-of-the-art. We also present an automated virtual remodeling application.

In summary, the contributions of the paper are three-fold: (Technical) Bundle pooling modules that effectively analyze line structure; (Benchmark) A new polygonal architecture segmentation problem with ground-truth annotations; and (System) A fully automated virtual remodeling system producing production-quality results.

2. Related Work

Segmentation: Instance segmentation has been an active area of research. Proposal-based techniques such as Mask R-CNN [13] have been a popular approach with great success. Proposal-free approaches via metric learning are also emerging [15]. However, these methods produce a segment as a group of pixels without vector-graphics structure. Our goal is to extract architectural components as compact polygon primitives, following the building perspectives.

Facade Parsing: Facade parsing has a long history in computer vision [25, 24]. A popular approach is to first identify the dominant building facade, rectify the image, then parse components in the rectified domain, following a shape grammar. Nishida et al. turns a single image of a building with multiple facade orientations into a CAD quality 3D model [27]. They also first identify dominant facades and process rectified images one by one. A shape grammar was also proven effective for architectural modeling from sensor images [31]. However, shape grammars need to be simple and are not scalable to more complex domains such as our problem, where non rectangular architectural components are arranged in more cluttered ways.

Architectural modeling: The origin of piece-wise smooth scene modeling dates back to the 90's in the stereo setting [29, 4]. With the emergence of efficient discrete optimization techniques [5], the second wave of piece-wise smooth reconstruction research [10, 11] surged after the millennium. These methods effectively detect components, but in a form of pixels with imprecise boundaries.

Wire-frame parsing seeks to directly model component boundaries as a set of line segments from a single image. The state-of-the-art approach jointly trains a corner detector and a connection classifier end-to-end [33]. 3D wire-frame reconstruction reconstructs wire-frame models with depth information [34]. However, reconstructed wire-frames tend to be fragmented with many missed connections: a poor system as a segmentation method for photography applications.

Instead, there are approaches producing closed polygons from segments. For example, polygons could be directly predicted by recurrent neural network (RNN) [6, 2, 20]. However, they are not designed to exploit structure priors (i.e., vanishing geometry and primitive types), which is the key to our problem.

Room layout estimation has been an active area of research, which produces a complete architectural wire-frame of an indoor scene [9, 14]. More recently, CNNs enable robust solutions [36, 18]. Although these methods exploit highly constrained layout structure, they often assume a rectangular room, which is much simpler than our setting.

Floorplan reconstruction seeks to infer a planar graph of an arbitrary topology with a complete room segmentation [23, 7]. The problem relies heavily on principal directions. Our algorithm also relies on vanishing geometry but handles surfaces not aligned with principal directions, which is almost always the case for roofs.

Vanishing Point Detection: Vanishing points (VPs) are crucial for architectural scene parsing. Line segment extraction and RANSAC is the classical approach for VP detection [3]. Neural network based approaches are proposed more recently, such as CNN with inverse gnomic projection by Kluger et al. [16] or conic convolution by Zhou et al. [32]. Our problem needs to assign a VP to each architectural component. We rely on an existing technique [3] for the detection, while focusing on the VP assignment via a novel bundle-pooling module.

3. Polygonal Architecture Segmentation Problem

We propose an architectural segmentation problem. The section explains the problem, the dataset, and the metrics.

3.1. Problem Definition

Residential buildings consist of architectural components that exhibit strong structural regularities. The input is a photograph of a house and a set of vanishing point candidates. The task is to decompose a house into a set of architectural components such as walls, windows, doors, garages, or roofs, each of which is a simple geometric primitive (i.e., a triangle, a rectangle, a pentagon, or a trapezoid) and adheres to the given perspectives (See left column of Fig. 3). Each polygonal boundary must point to one of the vanishing points, except for (1) the top two triangle edges; (2) the top two pentagon edges; and (3) the left and right trapezoid edges, which are highlighted in red or blue at the left of Fig. 3.

3.2. Annotations

We need 3 annotations for each architectural component in an image: 1) Geometric primitive (either *triangle*, *rectangle*, *pentagon*, or *trapezoid*), 2) Architectural type (either *wall*, *window*, *door*, *garage*, or *roof*), and 3) Corresponding Vanishing Points (VP) from the pre-computed VP set. Bounding boxes are computed from the primitive shapes.

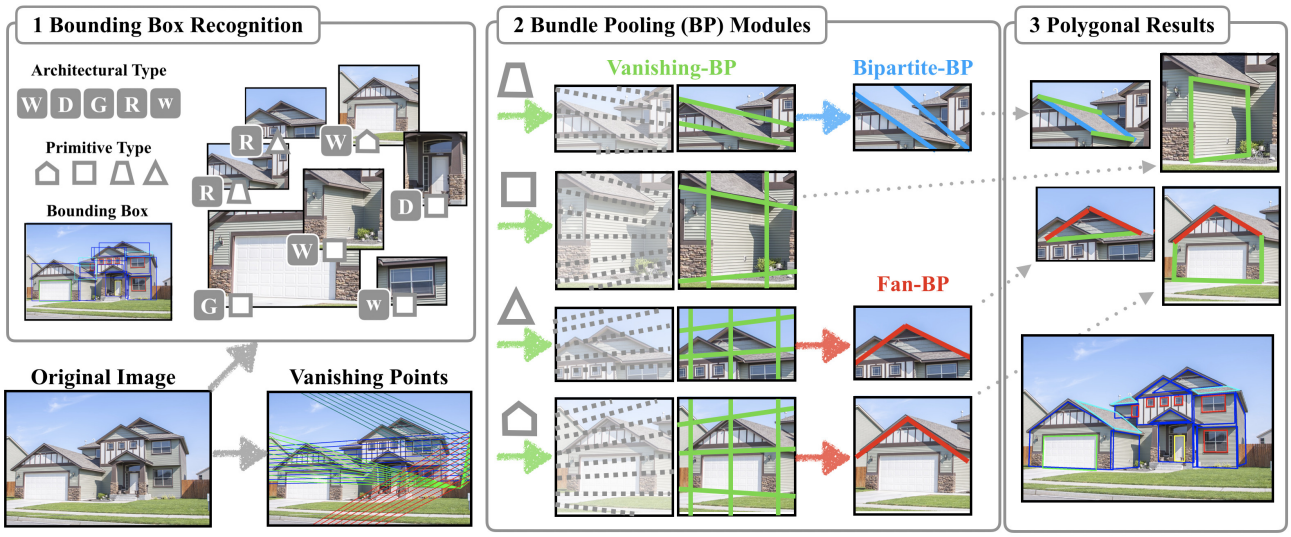

Figure 2. Our system consists of two DNNs. The first DNN is a variant of Faster RCNN, detecting architectural components with a few attributes such as the primitive type. The second DNN uses a combination of 3 bundle pooling modules (shown by Green, Blue and Red arrows resp.) to segment an architectural component as a polygon primitive.

Drawing polygons required educated guesses due to prevalent occlusions (e.g., cars, trees, bushes, or self-occlusions). For simplicity, we assume that all the roof segments are trapezoids and trapezoids appear only for roofs. We excluded exceptions from our dataset, which are less than 5% of the cases and can be safely ignored.

For VP assignment, we first generate four VPs containing the principle directions of the house. While a sophisticated technique exists [32], we take a simple approach: Extracting line segments by LSD [12] and enumerating four VP candidates by RANSAC [3]. Though most houses contain only one or two principle directions, we keep four candidates to make sure that the ground-truth are in the set. We use a heuristic to assign a VP to each component, followed by manual verification.

3.3. Dataset

Our dataset consists of $5,039$ high resolution photographs of residential building exteriors for real commercial applications, offered by a home design company Renoworks [1] [1]. We split the annotated dataset into 4647 images for training, and 392 images for testing. We also collected 60 un-annotated images from completely different sources via Google search.

3.4. Evaluation Metrics

We use mean Average Precision (meanAP) to evaluate the system effectiveness, based on Intersection over Union (IoU) or Object Keypoint Similarity (OKS). IoU is computed between ground-truth shape and predicted shape, while OKS is the distance-based similarity between the corresponding vertices. Both range in $[0, 1]$ to show the match quality from the poorest to perfect. True positive will only be counted when IoU (or OKS) is higher than a threshold with one ground-truth of same architectural type. In particular, we use the implementation provided by MSCOCO [21] to compare both IoU meanAP and OKS meanAP.

4. System Overview

Our system consists of two DNNs (See Fig. 2). The first DNN is a variant of Faster RCNN architecture [28]. Besides detecting bounding boxes of architectural components, this network classifies an architectural type (i.e., *wall*, *window*, *door*, *garage*, or *roof*), and a primitive type (i.e., *triangle*, *quadrilateral*, *pentagon*, or *trapezoid*). The network architecture and training is standard and the details are referred to the supplementary document.

For each detected component, the second DNN uses the combination of three bundle pooling modules to assign the horizontal VP and estimate polygon boundaries (See Fig. 3). First, vanishing-line bundle pooling (V-BP) determines the horizontal VP and estimates the top and bottom polygon boundaries except for pentagons, which require three horizontal boundaries (green in Fig. 3). For quadrilaterals, V-BP with the vertical VP determines the left and right boundaries. For triangles and pentagons, V-BP with the vertical VP determines the vertical boundaries (i.e., left, middle, and right), then fan bundle pooling (F-BP) refines the beveled edges (highlighted in red in Fig. 3). Lastly for trapezoids, bipartite bundle pooling (B-BP) determines the left and right boundaries (blue in Fig. 3).

[1] https://www.renoworks.com/

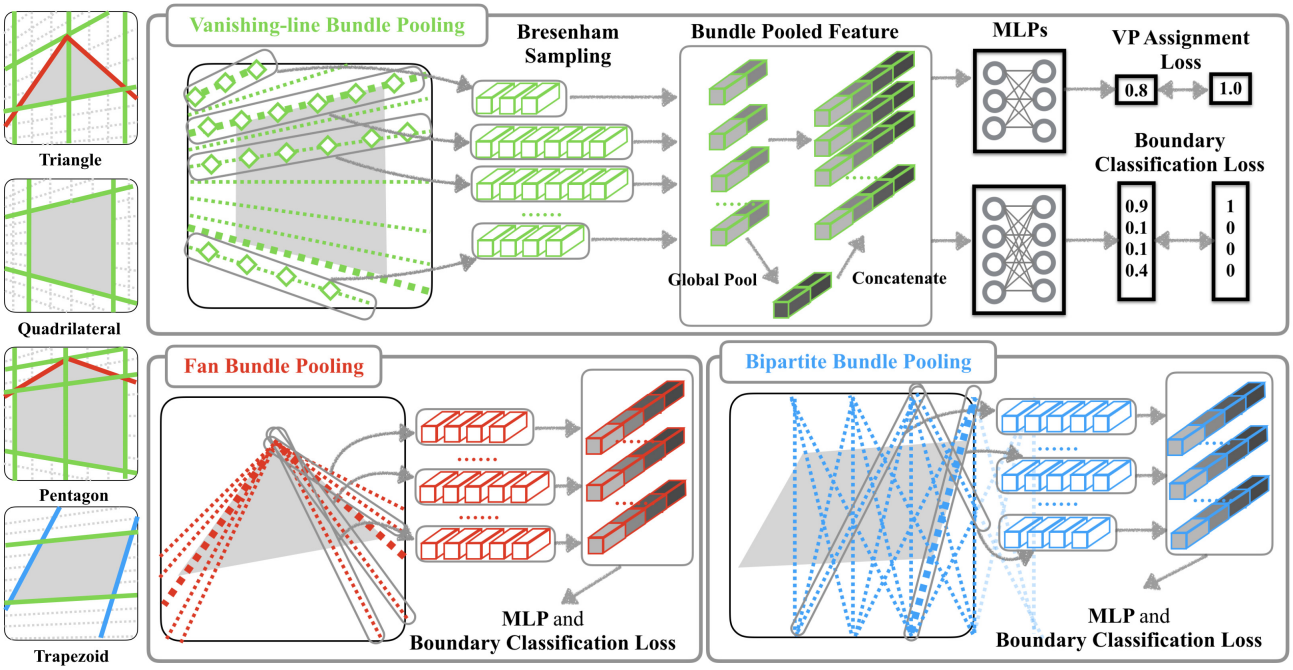

Figure 3. Three bundle pooling modules determine the polygonal segmentation boundaries. Vanishing-line bundle pooling (V-BP), fan bundle pooling (F-BP), and bipartite bundle pooling (B-BP) are highlighted in green, red, and blue, respectively. The line colors at the left of the figure indicates which bundle pooling modules estimate which lines.

5. Bundle Pooling

This section explains the details of the three bundle pooling modules and a post-processing step.

5.1. Vanishing-line Bundle Pooling (V-BP)

V-BP estimates polygonal structures aligned with a given vanishing point (See Fig. 3). V-BP takes a bounding box of an architectural component, its primitive type, and one vanishing point as the input. We first determine the range of vanishing lines that intersect with the bounding box, then evenly split the angular range into $K(= 40)$ lines. For each such vanishing line, we determine pixels along the line inside the bounding box by Bresenham's algorithm. We extract the corresponding feature vector of size $L(= 384)$ from P2 feature map [2] at each pixel from our Faster RCNN variant. Since lines may have different number of pixels, we aggregate vectors by both max and mean operations, and concatenate them into a $2L$-dimensional line-feature vector.

Boundary Classification: Borrowing an idea from non-local networks [30], we aggregate line-features across K vanishing lines into a *global bundle-feature*, again by max and mean poolings. We concatenate the global bundle-feature back to each line-feature, making its dimension $4L$. Lastly, we use a multi-layer perceptron (MLP) for each line-feature and a softmax function across the K line-features to produce the probability distribution of the boundary location. We use a standard cross-entropy loss. In our implementation, V-BP always classifies three boundaries (left to right or top to bottom) and shares weights across different primitive types. Lastly, an order loss is used on the output of multiple MLPs so that the order of the boundaries are consistent (e.g., the top border must not go below the middle or the bottom borders). We formulate it as a simple L_0 norm, computing the number of violating pairwise orders.

Vanishing Point Assignment: Given one of candidate VPs, we use V-BP module again to generate L line-features of length $4K$, which are concatenated and passed through 2 layers of MLP to regress the confidence of the VP. Softmax is used to take confidence values from all the VP candidates with a cross entropy at the end. This allows our system to accept any number of VP candidates. At test time, we apply V-BP to all the VP candidates, and pick the result with the highest confidence.

5.2. Fan Bundle Pooling (F-BP)

While the intersections of vertical and horizontal boundaries could determine the triangle and pentagon shapes (See the left of Fig. 3), their beveled edges are often imprecise due to the occlusions near the "shoulder" vertices such as trees. We fix the top vertex and consider an angular range

[2] P2 are feature map from the finest level defined in Feature Pyramid Network [22], which is a part of our Faster RCNN variant. It encodes sufficient rich semantics. We changed default feature length from 256 to 384.

of ± 20 degrees around the current estimate. We split the angular range into 40 lines evenly to create a bundle. F-BP is only used for the boundary classification, and we add the same feature pooling, MLP, and loss functions as in V-BP.

5.3. Bipartite Bundle Pooling (B-BP)

Trapezoid is the most challenging shape, whose left and right boundaries can have arbitrary directions. Without loss of generality, suppose we seek to infer the right boundary. We take the top-right corner of the bounding box, and evenly sample 11 points at the top border of the bounding box in the range of $1/2W$ (outside) to $-2/3W$ (inside) from the bounding box corner. W is the width of the bounding box. Similarly, we sample 11 points around the bottom right bounding box corner.

We form an 11x11 complete bipartite graph, and use its edges to form a bundle. We use B-BP only for the boundary classifier with the same architecture except for the order loss, which is not used due to the complexity. To improve the localization accuracy, we apply B-BP in two levels hierarchically. More specifically, given the polygon boundary from the first B-BP, we consider a range of $\pm 1/8W$ along the top and the bottom borders of the bounding box away from the current estimate. We evenly sample 13 points, form a bundle of a bipartite graph, and conduct the boundary classification.

5.4. Post-Processing

For large architectural components, line sampling in V-BP is not fine enough to achieve pixel-level accuracy. Fortunately, the remaining displacements are small and we perform a local exhaustive search to refine the boundaries by V-BP. More precisely, we first compute edge-maps by utilizing the VPs as in a prior work [10], then search for the location with the maximum sum of edge-maps in the range of ± 12 pixels around the current estimate (See Fig. 4). [3]

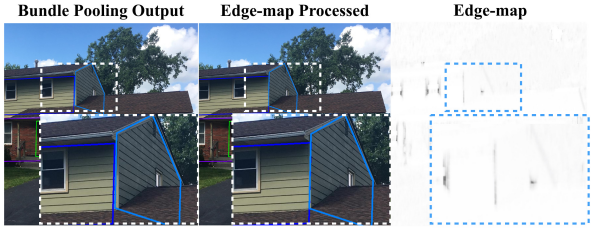

Figure 4. Edge-maps refine the polygon boundaries from V-BP.

6. Experimental results

We have implemented the proposed system in python and PyTorch, and used a NVIDIA 1080 ti 11GB for the experiments. The DNN training takes roughly 48 hours.

[3]To avoid misleading edges, we set edge-maps to be 0 inside the window segments.

6.1. Polygonal segmentation accuracy

Figure 5 shows our polygonal segmentation results with quantitative evaluations in Table 1. The figure and the table also compares against the following three baselines:

Keypoint: Our primitives have 3 to 5 corners. We modify the Keypoint head from Facebook's MaskRCNN code [26] to predict 5 Keypoints for every bounding box. Knowing the primitive type, we pick the appropriate number of estimated Keypoints to form the polygonal shape. For example, we use the first three Keypoints to form a triangle primitive.

Homo + Axis Pooling: Facade parsing literature offers sophisticated axis-aligned recognition techniques. We also compare the recognition quality when shapes are rectified. For each house, we apply homography to generate a rectified image for each dominant facade orientation. [4] We then apply bundle-pooling module along coordinate axes on the rectified images. Polygons are transformed back to the original image by the inverse homography.

LineSweeping: Mask R-CNN produces decent component segmentation. We simply find a polygon whose boundary overlaps with the segmentation boundary the most. More precisely, given a primitive type and a bounding box, we have a heuristic to determine the range of lines for each boundary, then exhaustively search for the optimal line that pass through the most number of segmentation boundaries. Please see the supplementary document for the details.

Table 1 measures the IoU meanAP and OKS meanAP (See Sect.3.4), while Figure 5 provides qualitative evaluations. Only the most difficult architectural types (i.e., walls and roofs) are visualized to avoid clutter. Our bundle pooling consistently outperforms the others except for a few metrics, in which LineSweeping has small performance gains. From qualitative evaluations, we found that LineSweeping is less accurate at the intersections of walls, as the wall boundary is often not distinguishable. Another observation is that the homography rectification is not effective, especially for side walls or small components, where image distortions make it difficult for CNNs to extract reliable image information. As expected, Keypoint performs the worst, which does not utilize the vanishing geometry.

Table 2 verifies that pooling and processing as a bundle improves performance, as opposed to pooling and regressing the confidence of a line independently. More specifically, we change how we pool features in a bundle: Min/Max pooling for a line-feature and a global pooling for a bundle-feature. The table shows that the configuration with all the feature poolings consistently improve the performance. Another observation is that the roof is the

[4]For metric-accurate homography, we need a focal length. With an image with 2 VPs, we assume their orthogonality and calculate focal length. For an image with 1 VP, we assume a default vertical field-of-view of 60 degrees and calculated.

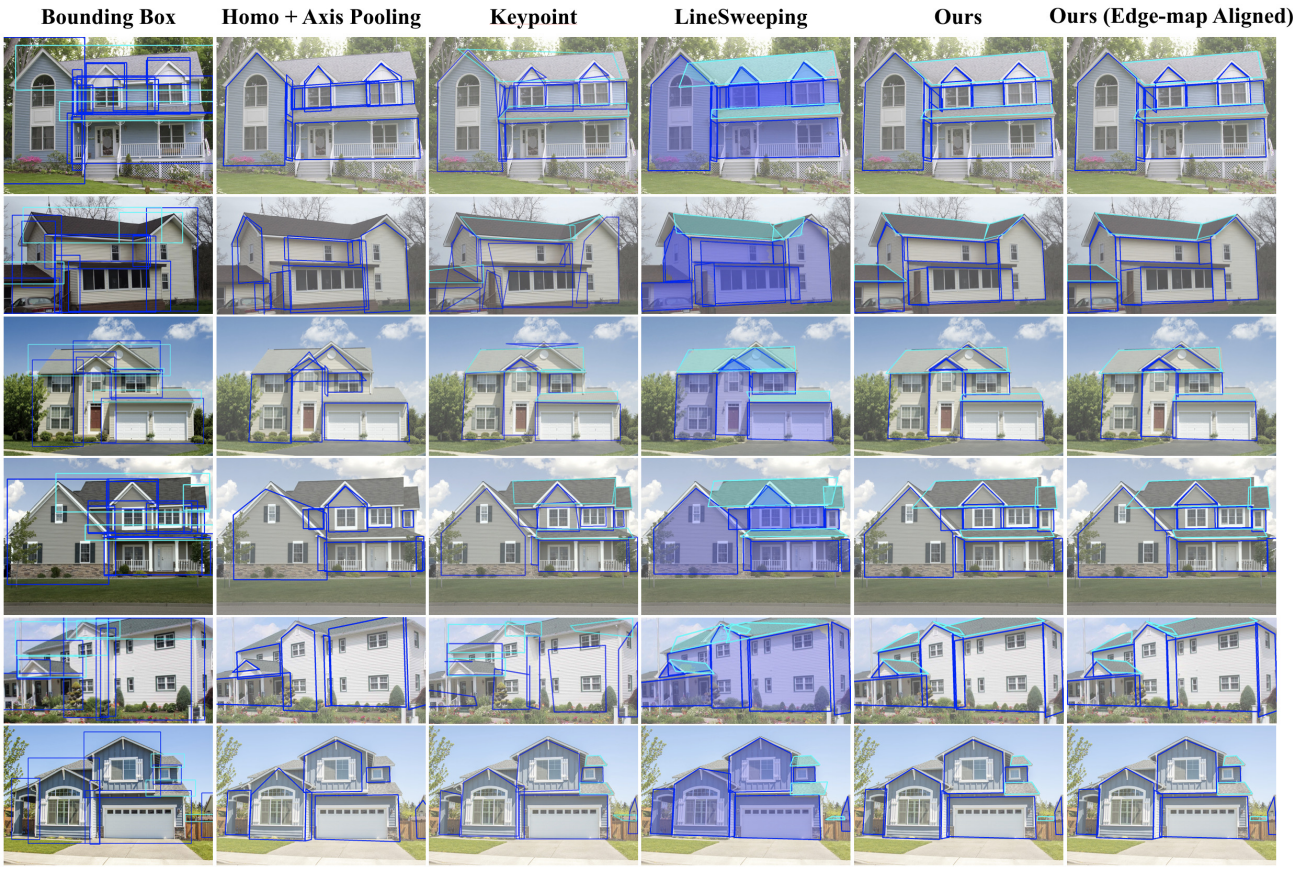

Figure 5. Polygon segmentation results. Only walls and roofs are shown to avoid clutter as they are the most challenging components. Axis Pooling does not work on roofs, because not all the VPs are known. LineSweeping results also visualize Mask R-CNN segments.

Table 1. Polygon segmentation accuracy: (IoU mAP / OKS mAP) and (IoU meanAR / OKS meanAR). Segments whose IoU/OKS above Thred with one ground-truth of same architectural type is positive. Thred=0.5:0.95 is the average mAP for IoU/OKS from 0.5 to 0.95 with a step size of 0.05.

	AP Thred=0.50:0.95	AP Thred=0.50	AP Thred=0.75	AR Thred=0.50:0.95
Homography+AxisPooling	39.1 / 34.0	68.1 / 63.9	42.4 / 32.4	50.7 / 24.7
Keypoint	34.0 / 15.9	63.9 / 35.9	32.4 / 12.8	42.1 / 25.2
Mask+LineSweeping	55.5 / 41.8	80.4 / 63.3	63.2 / 44.6	64.5 / 51.0
Ours	56.4 / 45.1	80.5 / 65.3	63.1 / 46.8	66.1 / 55.4

most difficult architectural type as expected. Their shapes have more variations and we do not have the vanishing point for the the left and the right boundaries. We noticed that door/window/garage IoUs are lower than expected, as they are usually precise in our visualizations. This is partly due to the inconsistencies in the annotations (e.g., to include/exclude frames or to annotate half/full window).

6.2. VP Assignment

Table 3 compares the VP assignment accuracy against a standard RANSAC based method: (1) Use OpenCV LSD software to extract line segments and discard those that lie outside the bounding box; (2) For each VP candidate, count the number of line segments that pass through with an angular tolerance of 5 degrees; and (3) Picking the VP with the highest count. We also vary our pooling operations for an ablation study. The table shows that our approach consistently outperforms RANSAC based approach with a large margin across all the categories (See Fig 6 for examples). Within our approach, max pooling alone is significantly worse than other pooling operations as expected, because we would like to accumulate all the pixel information as in the voting principle.

Table 2. Polygon segmentation accuracy: (Average Keypoint Distance / IoU) with different feature pooling strategies. We either turn on or off Mean/Max pooling in generating a line-feature, and a global pooling in generating a bundle-feature. We show absolute keypoint distance here to avoid effects from manual chosen scale factor in OKS.

Mean	Max	Global	Wall	Door	Window	Garage	Roof	All
x			29.7 / 0.73	10.9 / 0.87	8.4 / 0.76	10.0 / 0.80	67.2 / 0.50	25.2 / 0.73
	x		21.6 / 0.78	9.0 / 0.89	7.3 / 0.78	7.1/ 0.82	57.8 / 0.55	20.1 / 0.76
x	x		18.8 / 0.80	6.3 / 0.91	7.4 / 0.78	7.0 / 0.82	50.3 / 0.61	18.0 / 0.78
x	x	x	15.3 / 0.82	5.1 / 0.92	6.2 / 0.80	6.4 / 0.84	45.9 / 0.66	15.8 / 0.81

Table 3. VP Assignment Accuracy. The left (resp. right) number shows cases where the input is a set of 2 (resp. 4) VP candidates.

Method \ Test Set		One-Facade		Two-Frontal		Two-Facade		All	
		Wall	Roof	Wall	Roof	Wall	Roof	Wall	Roof
RANSAC		94.0 / 87.8	94.0 / 89.8	83.3 / 76.1	73.0 / 73.0	87.4 / 81.9	74.5 / 72.7	89.9 / 83.7	85.8 / 83.3
Ours	Mean	99.7 / 99.7	100.0 / 100.0	91.3 / 91.3	89.2 / 89.2	94.0 / 94.0	87.3 / 87.3	96.3 / 96.3	95.4 / 95.4
	Max	99.1 / 99.1	94.6 / 94.6	71.0 / 71.0	78.4 / 78.4	76.4 / 76.4	67.3 / 67.3	86.9 / 86.9	85.6 / 85.6
	Mean + Max	100.0 / 100.0	100.0 / 100.0	94.9 / 92.8	94.6 / 91.2	93.4 / 92.9	87.3 / 87.3	97.1 / 96.5	96.2 / 95.8

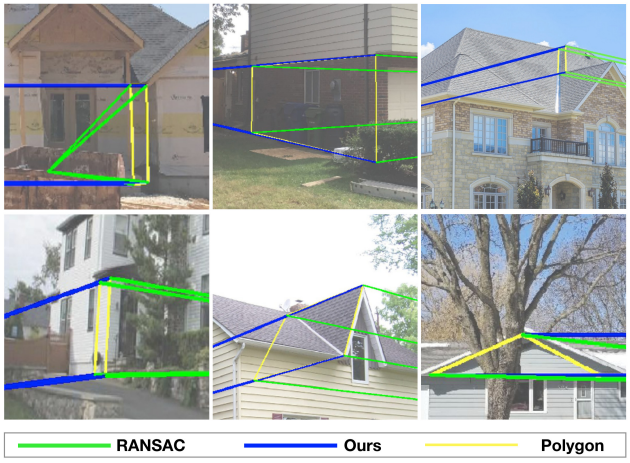

Figure 6. Instances in which our Bundle Pooling modules choose the correct VP from candidates, while RANSAC assignment does not.

6.3. Failure Cases

The proposed system is far from perfect with the following two major failure modes. First, near-frontal shots with two facade orientations exhibit small side walls with severe foreshortening effects. Mask R-CNN misses such components, and our algorithm is not able to recover from missed detections. 15% of the images suffer from this failure mode. Second, roof structures are generally very challenging, because adjacent roof components have similar texture/color, and exhibit unusual shapes without vanishing geometry as a guide.

6.4. Applications

Polygonal segmentation representation allows interesting applications. We demonstrate virtual remodeling and pop-up 3D billboards, which are combinations of standard techniques and refer system details to the supplementary document.

Virtual Remodeling: From an architectural asset database, also offered by a home design company Renoworks, we choose a new asset or material texture for each component. We apply homography to transfer the texture onto the image domain, and use the spectral matting algorithm [19] to composite an image. The tri-map is generated by (1) treating pixels inside the polygon as foreground pixels, and (2) using DeepLab v2 [8] to segment occluders (e.g., cars and greens) as background pixels. After obtaining the matte, we replace the foreground pixels inside the polygon with the new texture.

Figure 7 compares our results with Cycle-GAN [35] by assuming a scenario where we desire to transfer textures between LIGHT color and DARK colors. We manually collect 433 LIGHT and DARK wall images, and train Cycle-GAN, which surprisingly works well and re-colors wall texture properly. However, Cycle-GAN is unaware of building structure or vanishing geometry, introducing various artifacts in close-ups. Furthermore, a user is not able to control what new material or asset to insert, which is a critical disadvantage for a real-estate application, but possible by our system.

Pop-up 3D Billboards: Polygonal representation allows 3D pop-up models (See Fig.8). Kushal et al. [17] proposed a method to estimate surface normals of 2D polygonal shapes. We merge polygon boundaries within 15 pixels and use their method to estimate surface normals, while VPs are used to estimate camera intrinsics. After setting the depth of one vertex to a unit depth, we are able to recover depths of all the vertices and produce a 3D house model.

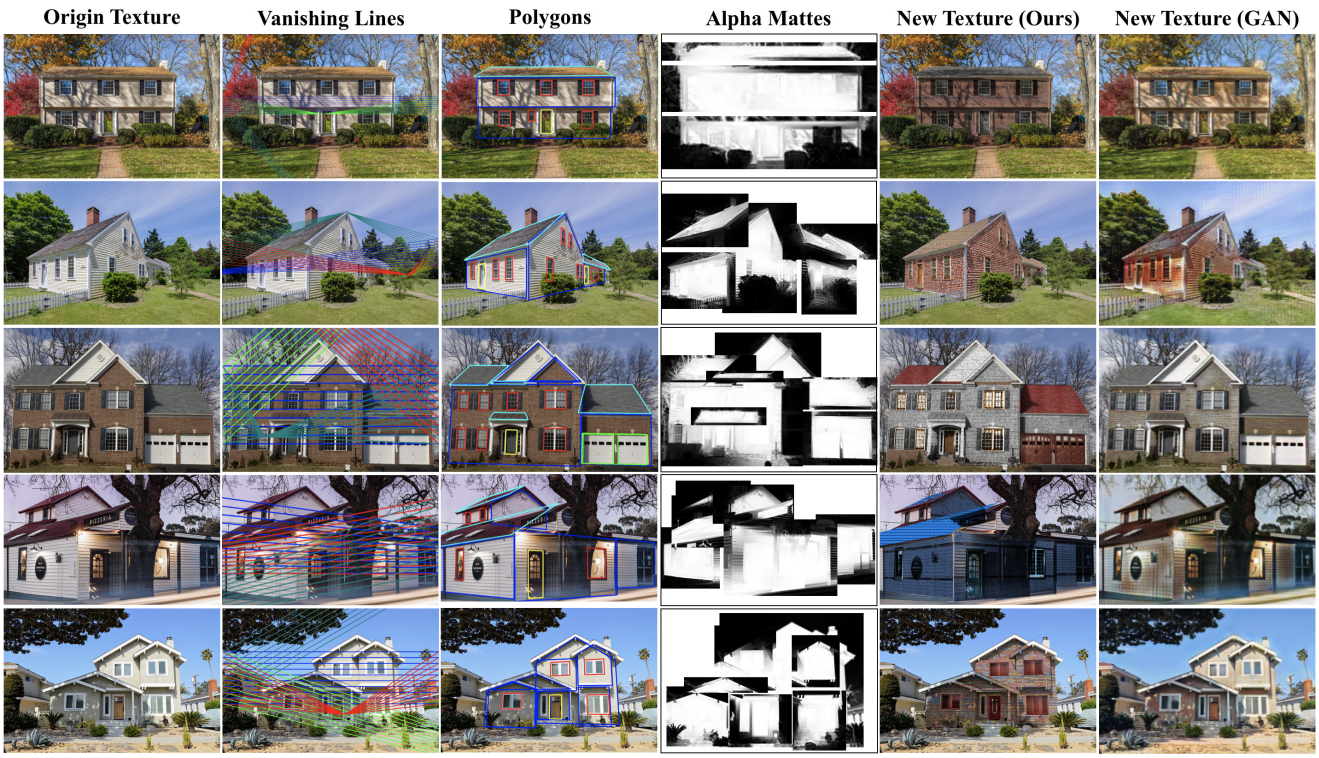

Figure 7. Virtual house remodeling application.

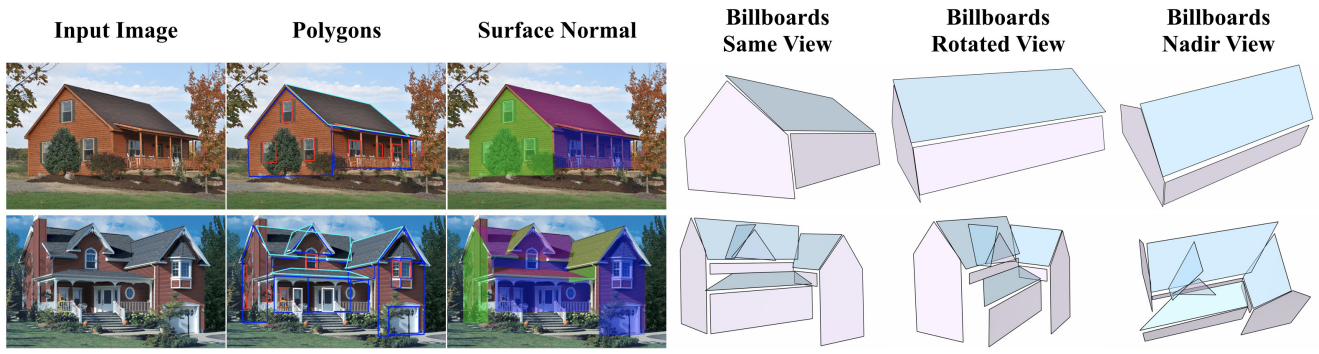

Figure 8. Pop-up 3D billboards application.

7. Conclusion

This paper introduces a polygonal architecture segmentation problem, and proposes a family of bundle-pooling modules which effectively assigns vanishing points and determines polygonal shapes for architectural components. Qualitative and quantitative evaluations demonstrate consistent improvements over the existing techniques based on our metric and benchmark. We also demonstrate virtual remodeling applications producing compelling results at the production quality. One of the limitations of our current approach is that architectural components are segmented independently. Simultaneous segmentation of architectural components with inter-component consistency is one of our future works. We will share our code for further research.

8. Acknowledgement

This research is partially supported by by National Science Foundation under grant IIS 1618685, NSERC Discovery Grants, NSERC Discovery Grants Accelerator Supplements, and DND/NSERC Discovery Grant Supplement. We thank Homayoon Farrahi and Nader Hamekasi at Renoworks [1] for their contributions on data collection.

References

[1] Renoworks. https://www.renoworks.com/. 1, 3, 8

[2] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. pages 859–868, 06 2018. 2

[3] J.-C Bazin and Marc Pollefeys. 3-line ransac for orthogonal vanishing point detection. pages 4282–4287, 10 2012. 2, 3

[4] Stan Birchfield and Carlo Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 1, pages 489–495. IEEE, 1999. 2

[5] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001. 2

[6] L. Castrejón, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4485–4493, 2017. 2

[7] Jiacheng Chen, Chen Liu, Jiaye Wu, and Yasutaka Furukawa. Floor-sp: Inverse cad for floorplans by sequential room-wise shortest path. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2661–2670, 2019. 2

[8] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, April 2018. 7

[9] Alex Flint, Christopher Mei, David Murray, and Ian Reid. A dynamic programming approach to reconstructing building interiors. In *European Conference on Computer Vision*, pages 394–407. Springer, 2010. 2

[10] Yasutaka Furukawa, Brian Curless, S.M. Seitz, and R. Szeliski. Manhattan-world stereo. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 0:1422–1429, 06 2009. 2, 5

[11] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1418–1425. IEEE, 2010. 2

[12] Rafael Gioi, Jeremie Jakubowicz, Jean-Michel Morel, and Gregory Randall. Lsd: A fast line segment detector with a false detection control. *IEEE transactions on pattern analysis and machine intelligence*, 32:722–32, 04 2010. 3

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. 1, 2

[14] Varsha Hedau, Derek Hoiem, and David Forsyth. Recovering the spatial layout of cluttered rooms. In *2009 IEEE 12th international conference on computer vision*, pages 1849–1856. IEEE, 2009. 2

[15] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. Learning to cluster for proposal-free instance segmentation. *CoRR*, abs/1803.06459, 2018. 2

[16] Florian Kluger, Hanno Ackermann, Michael Ying Yang, and Bodo Rosenhahn. Deep learning for vanishing point detection using an inverse gnomonic projection. *CoRR*, abs/1707.02427, 2017. 2

[17] A. Kushal and S. M. Seitz. Single view reconstruction of piecewise swept surfaces. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 239–246, June 2013. 7

[18] Chen-Yu Lee, Vijay Badrinarayanan, Tomasz Malisiewicz, and Andrew Rabinovich. Roomnet: End-to-end room layout estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4865–4874, 2017. 2

[19] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1699–1712, Oct. 2008. 7

[20] Zuoyue Li, Jan D. Wegner, and Aurélien Lucchi. Topological map extraction from overhead images. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1715–1724, 2018. 2

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 3

[22] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 4

[23] Chen Liu, Jiajun Wu, Pushmeet Kohli, and Yasutaka Furukawa. Raster-to-vector: Revisiting floorplan transformation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2214–2222, 2017. 2

[24] Hantang Liu, Jialiang Zhang, Jianke Zhu, and Steven Hoi. Deepfacade: A deep learning approach to facade parsing. pages 2301–2307, 08 2017. 2

[25] Anelo Martinović, Markus Mathias, Julien Weissenberg, and Luc Van Gool. A three-layered approach to facade parsing. In Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, editors, *Computer Vision – ECCV 2012*, pages 416–429, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. 2

[26] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. 5

[27] Gen Nishida, Adrien Bousseau, and Daniel G Aliaga. Procedural modeling of a building from a single image. In *Computer Graphics Forum*, volume 37, pages 415–429. Wiley Online Library, 2018. 2

[28] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(6):1137–1149, June 2017. 3

[29] John YA Wang and Edward H Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994. 2

[30] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018. 4

[31] Huayi Zeng, Jiaye Wu, and Yasutaka Furukawa. Neural procedural reconstruction for residential buildings. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 759–775, Cham, 2018. Springer International Publishing. 2

[32] Yichao Zhou, Haozhi Qi, Jingwei Huang, and Yi Ma. Neurvps: Neural vanishing point scanning via conic convolution. In *NeurIPS*, 2019. 2, 3

[33] Yichao Zhou, Haozhi Qi, and Yi Ma. End-to-end wireframe parsing. In *ICCV 2019*, 2019. 2

[34] Yichao Zhou, Haozhi Qi, Yuexiang Zhai, Qi Sun, Zhili Chen, Li-Yi Wei, and Yi Ma. Learning to reconstruct 3d manhattan wireframes from a single image. In *International Conference on Computer Vision*, 2019. 2

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 7

[36] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2051–2059, 2018. 2