Beat the MTurkers: Automatic Image Labeling from Weak 3D Supervision

Liang-Chieh Chen[1] Sanja Fidler[2,3] Alan L. Yuille[1] Raquel Urtasun[2,3]
[1]UCLA [2]University of Toronto [3]TTI Chicago
{lcchen@cs, yuille@stat}.ucla.edu {urtasun, fidler}@cs.toronto.edu

Abstract

Labeling large-scale datasets with very accurate object segmentations is an elaborate task that requires a high degree of quality control and a budget of tens or hundreds of thousands of dollars. Thus, developing solutions that can automatically perform the labeling given only weak supervision is key to reduce this cost. In this paper, we show how to exploit 3D information to automatically generate very accurate object segmentations given annotated 3D bounding boxes. We formulate the problem as the one of inference in a binary Markov random field which exploits appearance models, stereo and/or noisy point clouds, a repository of 3D CAD models as well as topological constraints. We demonstrate the effectiveness of our approach in the context of autonomous driving, and show that we can segment cars with the accuracy of 86% *intersection-over-union, performing as well as highly recommended MTurkers!*

1. Introduction

Over the past few years, we have witnessed immense progress in solving visual recognition tasks. This is due to the availability of new sources of labeled data as well as the development of richer, holistic representations that combine multiple tasks [9, 15, 37]. Most recent datasets provide annotations for multiple recognition tasks. For example, PASCAL VOC [6] provides classification, detection and semantic segmentation labels for a handful of objects. ImageNet [4] provides large-scale image classification and object localization annotations. Acquiring ground-truth data for each of these recognition tasks is very expensive even when employing crowd-sourcing systems such as MTurk. Thus, recently a number of approaches have tackled semantic segmentation in scenarios where weak annotations such as image tags, 2D bounding boxes, or strokes, are available.

In this paper, our aim is to leverage the already labeled tasks in order to automate labeling of the more time consuming ones. In particular, we show how to exploit annotated 3D bounding boxes (available *e.g.*, in KITTI [11]) to perform accurate object segmentation *without* any user in the loop. This is in contrast to interactive segmentation techniques, where the user can iteratively correct mistakes by giving additional strokes/seeds. Towards this goal, we develop an approach that exploits 3D information in the form of stereo, noisy point clouds, 3D CAD models, as well as appearance models and topological constraints (Fig. 1).

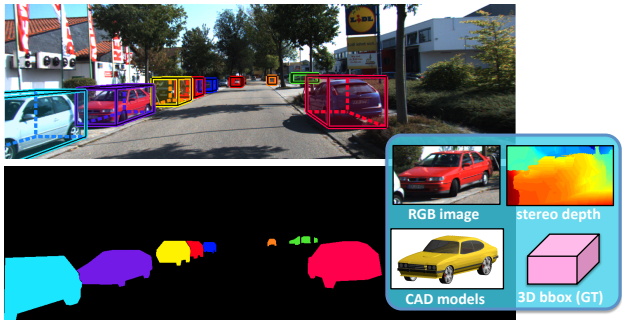


Figure 1: Our goal is to **automatically** generate segmentation ground-truth (bottom) using weak labels (top image).

We demonstrate the effectiveness of our approach in the context of the challenging KITTI autonomous driving dataset [11], which has been annotated with 3D bounding boxes, but not segmentation. This dataset is particularly rich as it contains image pairs collected with a stereo rig as well as point clouds captured with a LIDAR. It is also very challenging, as both the objects present in the scene as well as the ego-car (where the sensors are mounted) are moving. This poses many difficulties, some of which are illustrated in Fig. 2. First, the ground-truth 3D bounding boxes can be fairly noisy, as annotation in the presence of occlusion is very difficult. Non-reflective materials such as car windows and certain types of paint make the LIDAR point clouds very sparse. The point clouds also contain outliers, particularly near occlusion boundaries due to errors in registration. Furthermore, the objects in the scene move (fast) and the LIDAR does not perform instantaneous capture. The 3D information is also particularly ambiguous for the far-away objects as the density of the point clouds decreases with the distance to the sensor. Similarly, errors in the stereo estimation process also increase dramatically with the distance to the camera. Occlusion, low-resolution and saturated areas are other sources of ambiguities.

Our approach successfully deals with these difficulties and is able to perform as well as human annotators (i.e., MTurk) while being fully automatic. We conducted our experiments on a subset of KITTI [11], which contains 950 cars with different scales, lighting/shading conditions, and occlusion levels. We reach a remarkable accuracy of 86% IOU in this challenging setting, outperforming GrabCut [24] by 23%. Importantly, we require as little as 10 to 20 labeled objects to train the system. Furthermore, our approach can also be used to de-noise MTurk annotations, improving by additional 3%. Our code and annotations are available at: https://bitbucket.org/liang_chieh_chen/segkitti.

2. Related Work

Interactive segmentation: In [1], scribbles were used as seeds to model the appearance of foreground and background, and segmentation was performed via graph-cuts by combining appearance cues with a smoothness term [2]. GrabCut [24] utilizes annotations in the form of 2D bounding boxes, and computes the foreground/background models using EM. [12] extended this idea to 3D by employing point clouds. 3D information as well as video has been used to ease the labeling task. Optical flow and structure from motion constraints are used in [32] to propagate image labels in RGB-D videos. Our work bears similarity with [18, 35] as we also exploit existing annotations for segmentation. However, while they utilize 2D boxes provided in ImageNet or tags in SIFT flow, we leverage 3D boxes and define a rich set of 2D and 3D potentials.

Point clouds: Point clouds have been exploited for semantic segmentation [3]. [33] enforced multiview consistency by feature tracking. In [34], neighboring points in a point cloud were grouped to form planar patches to which labels (walls, floors, ceilings, clutter) were assigned. [17] over-segmented the point clouds and modeled object co-occurrences and geometric arrangement. In [38], the physical stability of objects is employed when segmenting point clouds. [15, 23, 38] proposed to explore the compatibility between multiple segmentation hypotheses and 3D maps in order to perform 3D object detection. In contrast, we seek to exploit labeling compatibility between 2D image pixels and 3D point clouds. Besides, we focus on object segmentation, rather than object localization.

Shape priors and CAD models: Many approaches have incorporated shape priors into MRFs. [14] combined MRFs and deformable models, [10] utilize a template with a level-set formulation, and [37] incorporate shape priors computed from deformable part-based models in holistic scene understanding. [31] use DijkstraGC to impose object connectivity priors into segmentation. Objcut [19] employed learned object poses, while PoseCut [16] exploits a human pose-specific shape prior. [20] encodes tightness of the segmentation to the bounding boxes provided by users. The star shape prior of [30] enforces connectivity along rays launched from a user-specified center. This was improved in [13] by using multiple stars and Geodesic paths. The prevalent use of CAD models has been to augment the training data with synthetically generated views or to learn a geometric model from them [21, 22, 39], for the task of object detection. In [25], whole synthetic scenes are matched to real ones in order to transfer segmentation labels. Similarly, we fit CAD models to our examples and use the mask of the best fit as a potential.

3. Segmentation from Weakly Labeled Data

In this paper, we are interested in performing automatic segmentation given annotated 3D bounding boxes, a collection of CAD models, LIDAR point clouds and/or stereo image pairs. We frame the problem as the one of figure/ground segmentation in a Markov Random Field (MRF), which exploits appearance, smoothness, shape priors from CAD models and 3D information. To exploit 3D, we use two alternative depth sources: LIDAR, which provides us with sparse point clouds, and depth from stereo. We first describe how we compute depth in both settings. We then detail the energy of our MRF and discuss learning and inference.

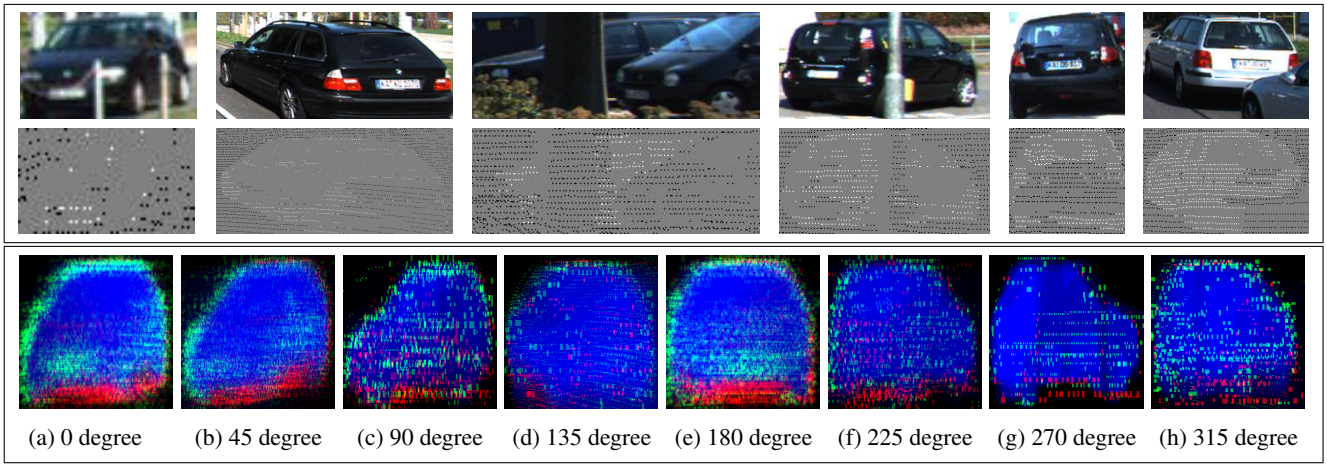
3.1. Obtaining Dense Depth Maps

We employ PCBP [36] to compute **stereo depth maps**, as it is the current state-of-the-art in KITTI. PCBP is a slanted plane MRF model with explicit reasoning about the type of boundary (coplanar, hinge, occlusion) between neighboring superpixels. This allows us to encode physical validity at junctions, resulting in better plane estimates.

As shown in Fig. 2, the **LIDAR point clouds** can be very sparse, even for nearby objects. To overcome this issue, we reconstruct a dense depth map from the sparse set of points. Let d_i be a continuous random variable encoding the depth of the i-th pixel, and let $\hat{d}_i$, with $i \in \mathcal{V}_D$, be the observed depth at pixel i, with $\mathcal{V}_D$ the sparse subset. We formulate the dense reconstruction as the one of inference in a continuous MRF. Specifically, the MRF encourages the reconstructed depth to be close to the observed depth measurements as well as the depth of neighboring pixels to be smooth. We define the energy as follows:

$$E(\mathbf{d}|I, \hat{\mathbf{d}}) = \sum_{i \in \mathcal{V}_D} \|d_i - \hat{d}_i\|_p^p + \sum_{(i,j) \in \mathcal{E}} w_{ij} \|d_i - d_j\|_p^p$$

with $\mathbf{d} = (d_1, \cdots, d_N)$ the set of variables encoding depth for all pixels, $\hat{\mathbf{d}}$ the set of observed depth values, I the color image and $\mathcal{E}$ the set of 4-connected neighbors. We use $w_{ij} = \lambda \cdot \exp(-\frac{\|I_i - I_j\|_2^2}{\beta})$, with λ the relative weight between the first and the second term, and β a scalar. Note that this energy generalizes [5] to p-norms: When $p = 2$ we obtain a Gaussian MRF, and when $p = 1$ a Laplacian MRF. Exact inference can be performed in both cases [26].



(a) 0 degree (b) 45 degree (c) 90 degree (d) 135 degree (e) 180 degree (f) 225 degree (g) 270 degree (h) 315 degree

Figure 2: **Challenges of LIDAR: (Top box)** Points falling inside the ground-truth 3D boxes are white, and black outside. Gray pixels indicate there is no LIDAR observation. Points are sparse for small cars as well as large ones due to non-reflective materials (see first two examples). The projected points do not align well with image boundaries (see trunk and pole in the third and fourth examples). The GT 3D boxes are noisy, clipping car points or including background ones, *e.g.*, floor (see last two images). **(Bottom box)** Point clouds are averaged over the dataset for 8 different viewpoints. Red: car points that fall outside the foreground GT mask (false positives). Green: background points that fall within the GT mask (false negatives).

3.2. Semantic Segmentation using 3D Data

Our setting is the following: we assume we are given an example annotated with a 3D bounding box. Our goal is to produce a figure-ground segmentation of the depicted object. Note that the image within the projected box can in fact contain two objects due to occlusion. Our evaluation measures instance-level overlap, and thus our goal is to only label the correct object corresponding to the given 3D box.

Let $y_i \in \{0, 1\}$ be a random variable encoding whether the i-th pixel is background or foreground, respectively. We now describe the potentials in our segmentation MRF.

Appearance Model: Following [24], we utilize two Gaussian Mixture Models (GMMs) to model the appearance of foreground/background. However, instead of using a human in the loop to define the seeds, we employ the available 3D information. We define the foreground seeds $\mathcal{F}$ to be the set of pixels inside the 2D bounding box which, when projected to 3D (using depth from either LIDAR or stereo), lie inside the 3D box (white pixels in Fig. 2). To compute the background seeds $\mathcal{B}$, we take the remaining pixels in the 2D box (black pixels in Fig. 2) as well as a 2-pixel band around the box. To make this process more robust, we compute a 2D convex hull around $\mathcal{F}$. We then remove all background seeds that fall inside this hull and have larger depth values than the mean depth value of the foreground seeds. These outliers are usually caused by the laser rays passing through the car windows. This simple procedure led to a slight improvement in performance. Then

$$\phi_i^{app}(y_i) = \begin{cases} -\log Pr(I_i|\theta_F^I) & \text{if } y_i = 1 \\ -\log Pr(I_i|\theta_B^I) & \text{if } y_i = 0 \end{cases}$$

where I_i is the color of pixel i, and θ_F^I and θ_B^I are the color appearance models for the foreground and background.

Smoothness: We employed an Ising prior

$$\phi_{ij}^{ising}(y_i, y_j) = \mathbb{1}_{(y_i \neq y_j)}$$

as well as a contrast sensitive Potts model

$$\phi_{ij}^{cs}(y_i, y_j) = \exp\{-\frac{\|I_i - I_j\|_2^2}{\beta}\} \cdot \mathbb{1}_{(y_i \neq y_j)}$$

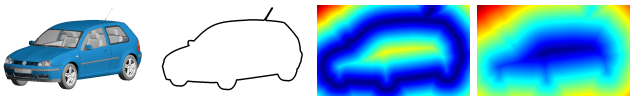
where $\beta = \mathbb{E}(2 \cdot \|I_i - I_j\|_2^2)$ and $\mathbb{E}(\cdot)$ denotes the expectation over an image sample [1].

Topological Information: We modify the generic shape prior of [13, 30], which enforces the connectivity of segmentation pixels along the rays originating from a point (*i.e.*, the star), to be asymmetric as follows

$$\phi_{ij}^{topo}(y_i, y_j) = \mathbb{1}(y_i = 1, y_j = 0)$$

for pixels on the ray going from j to i. Unlike [13, 30], we automatically obtain the stars via K-means on the set of foreground seeds $\mathcal{F}$. Furthermore, we learn the importance of this potential instead of assigning it an infinite weight.

Leveraging CAD models: We use a collection of 180 car CAD models collected in [8]. The idea is to find, for each 3D KITTI object, a CAD model that best fits this example and use the resulting CAD's shape as a prior for segmentation. Towards this goal, we first solve for the best 3D transformation that aligns each CAD model with each input 3D bounding box in terms of vertex error in 3D. This can be done in closed form via Procrustes analysis. The next step is to select the model which best represents each example. We chose the model that minimizes an error function consisting of 3 terms: 3D and 2D vertex error and real-world dimensions error. To compute the 2D error we project the



(a) CAD shape (b) contour (c) dist. trans. (d) signed d.t.

Figure 3: (a) CAD model projected to image plane, (b) contour, (c,d) distance and signed distance transform.

CAD's 3D box to the image, fit a 2D box around it and compute the L2 distance between its vertices and the vertices of the input 2D box. To compute the last error term, we use the real-world dimensions of each CAD model. This captures, for example, that a VW Golf is smaller than a Toyota Tundra. This cannot be obtained from the CAD models, as they have arbitrary scale. Instead, we obtain this information from the car manufacturers and other online resources, as each CAD model's meta-data contains information about the car brand and model [8]. Real-world dimension error is then the sum of L1-distances between the real-world dims (length, height, width) of a CAD and the dims of the input 3D box. We weigh each error term differently and learn the weights by cross-validation on the training set. The best CAD model is then projected to the image plane and its contour is extracted. We perform the signed distance transform [7] on the contour (the sign is negative within the car contour), and normalize it to be between -1 and 1 for every example in order to not bias large cars. Thus

$$\phi_i^{CAD}(y_i) = sdt(i|CAD) \cdot \mathbb{1}(y_i = 1)$$

where $sdt(i|CAD)$ is the value of pixel i after signed distance transform. This is illustrated in Fig.!3.

Depth seeds: We penalize labelings that misclassify foreground and background seed points as follows

$$\phi_i^{depth,f}(y_i) = \mathbb{1}(y_i = 0, i \in \mathcal{F})$$
$$\phi_i^{depth,b}(y_i) = \mathbb{1}(y_i = 1, i \in \mathcal{B})$$

with $\mathcal{F}, \mathcal{B}$ the set of foreground and background seeds.

Dense depth: We learn two GMMs to represent the histogram of depth for the background and foreground. Thus

$$\phi_i^{depth}(y_i) = \begin{cases} -\log Pr(d_i|\theta_F^S) & \text{if } y_i = 1 \\ -\log Pr(d_i|\theta_B^S) & \text{if } y_i = 0 \end{cases}$$

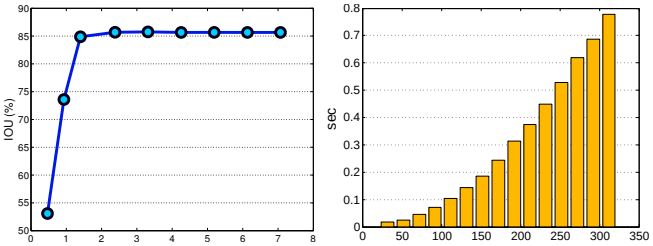
with d_i is the dense reconstructed depth (either from stereo or the MRF defined in section 3.1), and θ_F^S and θ_B^S are the depth appearance models for foreground and background.

3.3. Learning and Inference

We use a log-linear model

$$E(\boldsymbol{y}|I, \mathbf{d}) = \boldsymbol{w}^T \cdot \Psi(\boldsymbol{y}|I, \mathbf{d})$$

where $\Psi(\boldsymbol{y}|I, \mathbf{d})$ is the concatenation of all potentials summed across cliques (as we share the weights between them). This results in $\boldsymbol{w} \in \mathbb{R}^8$, with one weight for appearance, two for smoothness, two for discrete depth, one for continuous depth, one for the topology and one for CAD.



(Training time in minutes) (Avg. inference time)

Figure 5: **Time: (Left)** IOU as a function of training time, **(Right)** average inference time as a function of scale.

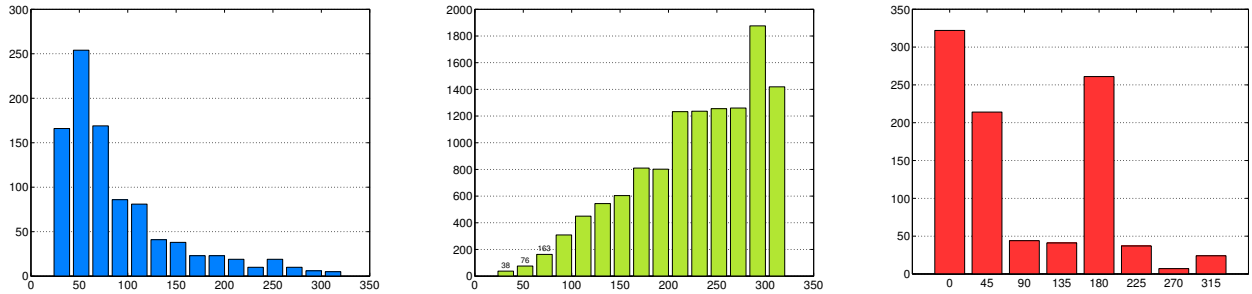
Inference in our model can be done exactly via graph-cuts, as we have a binary MRF with sub-modular potentials. We use the graph-cut implementation of [2]. We employ structural SVMs [28, 29] to learn the parameters of the model, and use the parallel cutting plane algorithm of [27]. As loss we use Hamming distance, which counts the percentage of pixels that are wrongly labeled. This loss decomposes into unary potentials, and thus the loss-augmented inference can be solved exactly via graph-cuts.

4. Experimental Evaluation

We selected a random subset of images from the KITTI `raw data` [11], having a total of 950 cars. We asked 9 in-house annotators to provide very high-quality segmentations. It took on average 60 seconds to label each car, and 16h to label the full dataset, where each image is labeled by a single annotator. As shown in Fig. 4 (left), our dataset contains cars at very different resolutions. The projection of the 3D bounding boxes into the image ranges from 20×24 to 279×372 pixels. In order to maintain the same distribution of scales, we build a 30 bin histogram of object sizes and select randomly within each bin $40\%, 10\%$, and 50% of cars as training, validation, and test set, resulting in 382 examples for training, 100 for validation, and 468 for testing. Following PASCAL VOC we utilize intersection-over-union (IOU) as our evaluation metric.

The 3D LIDAR point clouds are very sparse. As shown in Fig. 4 (center), small cars contain only 38.2 points on average, which is 3.7% of the 2D bounding box. This makes segmenting small cars very difficult. As expected, the number of points increases with scale. However, for the largest scale (the last bin), we find that there is one car with very few points due to its metallic paint. This biases the statistics as there are not that many examples of this size.

KITTI was collected with sensors (*i.e.*, cameras and LIDAR) mounted on top of a driving car. As a consequence, most of the cars in the images are oriented with 0, 45 and 180 degrees (driving towards or away from the ego-car). Fig. 4 (right) shows the distribution of car orientations (at



(BBox size) (LIDAR point clouds) (Orientation histogram)

Figure 4: **Dataset statistics:** **(Left)** Histogram of car scales in terms of bounding box size, defined as the square root of the bounding box area. **(Center)** Average number of LIDAR points for different scales. **(Right)** Histogram of car orientations.

0° the car is facing south, and west at 90°). To further analyze the point clouds, we overlay their projection with our ground-truth segmentations and average them for 8 different azimuth angles, after resizing to a common scale to compensate for the size difference among examples. As shown in the bottom row of Fig. 2, most of the mistakes (green and red) are along the car boundaries due to registration errors and the fact that objects and the ego-car are moving. Besides, the noise introduced when labeling 3D bounding boxes causes mistakes on the car and near the ground plane.

To benchmark and compare crowd-sourcing annotations, we collected segmentations via MTurk in three batches. For the first two batches our quality requirements were lower, requiring MTurkers to have at least 75% approval rate. We paid 1 cent per car. For the third batch we required 95% approval rate and 500 approved tasks, and paid them 5 cents per car. We asked the MTurkers to draw the outer boundary of the car within the marked box as accurately as possible. For images with low contrast we asked them to make an educated guess. If a car was occluded (by *e.g.*, a tree), they were asked to draw a boundary *also* around each occluding region. If the car was occluded by another car, they were asked to mark the other car as an occluder. We showed three examples of good annotations, a fully visible car, a low contrast example, and a partially occluded car.

Comparison to state-of-the-art: Table 1 compares our approach to a set of GrabCuts based baselines, which, like our approach, do not use a user in the loop. The first baseline (A) projects the 3D bounding box onto the image, and uses the pixels inside as the foreground seeds and a band outside as background seeds. The second baseline (B) utilizes only pixels inside 1/4 of the box around the center as seeds. The last baseline (C), which is an instance of our model, employs the projected LIDAR points as seeds for foreground or background, depending on whether they are inside or outside the 3D bounding box. Our approach outperforms the baselines by more than 20% in terms of IOU when employing LIDAR and/or stereo. Moreover, utilizing the point clouds as seeds outperforms the other heuristics.

Size matters: As shown in Fig. 6 (left) segmenting big cars is easier than smaller ones.

	Baselines			Our Model		
Method	[24] A	[24] B	[24] C	Stereo	LIDAR	Hybrid
IOU (%)	52.9	59.1	62.1	**84.4**	**85.6**	**85.9**

Table 1: **Comparison to GrabCuts:** Baseline A uses all pixels inside the 2D bounding box as foreground seeds, Baseline B uses pixels inside 1/4 of the 2D box as foreground seeds and baseline C uses projected LIDAR points as seeds. By hybrid we mean an approach that uses dense depth from stereo and depth seeds from LIDAR.

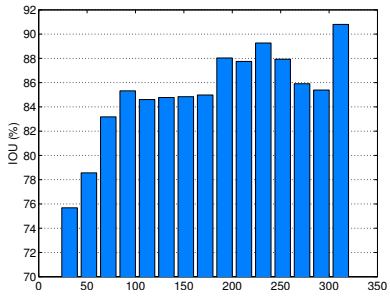
Importance of the features: As shown in Table 2, each potential increases performance. The CAD model is the potential that contributes the most.

Stereo vs LIDAR: As shown in Tables 1 and 2, we can reach almost the same performance with stereo or LIDAR. This is great news for autonomous driving as cameras are much cheaper sensors. The highest accuracy can be obtained when we use a hybrid approach, which takes advantage of both stereo and LIDAR. Fig. 6 (center) shows that LIDAR is particularly beneficial wrt. stereo for small cars.

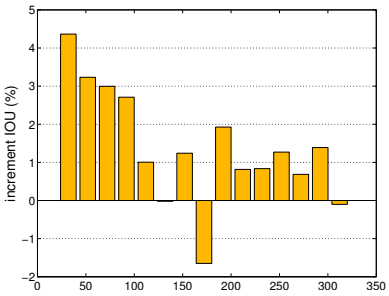
Time: Fig. 5 shows the training and average inference time as a function of size. Learning converges within 2 minutes. The parallel cutting plane algorithm of [27] makes the training time 3.3 times faster than the original algorithm (when using four threads). The average inference time is proportional to the car scale. Inference for the full test set takes 44 seconds using a single core.

Automatic vs MTurk: As shown in Table 3, our approach performs as well as MTurker, while being fully automatic. We can also **de-noise** MTurk annotations by using their segmentations as an additional potential in the model. The potential that we use is similar to the one for the CAD models, where instead of the CAD mask we use the MTurk segmentation. This pushes the performance even higher. Such a setting can for example be used to reduce the number of annotators per image. "Oracle" here means a scenario in which we can choose the best annotation within the three batches. Note that, even in this setting, we can de-noise the results using our model.

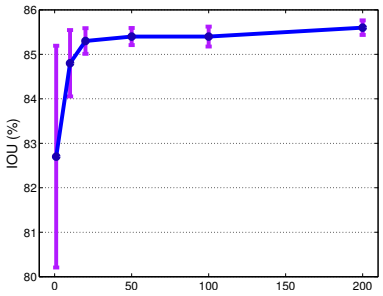
Training Set Size: We next investigate performance of our full model as a function of the number of training im-



(LIDAR)



(Improvement LIDAR wrt. stereo)



(Training set size)

Figure 6: **(Left)** IOU as a function of bounding box size when using LIDAR. **(Center)** IOU difference when employing our full model with LIDAR vs Stereo as a function of size. **(Right)** IOU as a function of number of training examples. With only 10-20 train-val images, our model can reach performance similar to MTurk. Results are averaged across 5 partitions.

	MTurk accuracy	Ours with MTurk pot.
Batch 1	85.3	**87.1**
Batch 2	85.9	**88.3**
Batch 3	86.7	**87.6**
Batch 1,2,3	–	**88.9**
Oracle	90.2	**90.6**

Table 3: **Denoising MTurk:** At 85.9% our model's accuracy is comparable to MTurkers'. Our model is also able to de-noise MTurk annotations, further boosting performance.

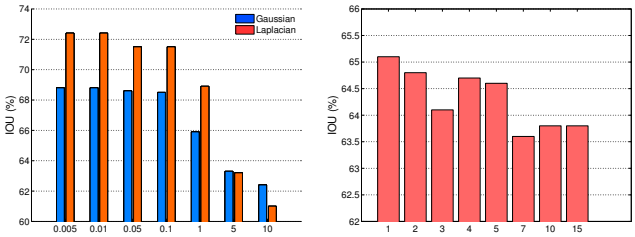
ages. As shown in Fig. 6 (right), results similar to MTurk can be obtained when training with as little as 10-20 images.

Gaussian vs Laplacian MRF: Fig. 7 (left) shows performance of the basic model (appearance + smoothness) when enriching it with continuous depth as a function of the smoothness term in the dense reconstruction. Note that $p = 1$ (i.e., Laplacian MRF) is more robust to outliers, and a wide range of weights result in good performance.

Number of Stars: As shown in Fig. 7 (right) using a single star yields the best performance for a model containing appearance, smoothness and topology. This is expected as cars are a single connected component, with the exception of occlusion (*e.g.*, car behind a pole).

Average shape: An alternative shape prior to CAD models is to cluster the data by orientation, and average the training segmentations for each bin (see Fig. 2). Adding this to the base model with appearance and smoothness results in 67.6% IOU, while using it in the full model results in 80.3%, *i.e.*, 5% worse than when using CAD.

Depth as a segmentation algorithm: We also look into whether the dense reconstructions alone can be used to perform segmentation. Towards this goal, we classify a point as car if it is inside the 3D bounding box, and background otherwise. Laplacian MRF performs best, with 76.2% IOU. The performance of the Gaussian MRF is 4% lower since it is less robust. The performance when using stereo is 69.8%, which is 15% worse than our full model. This shows that depth alone is not sufficient and that we greatly benefit from using multiple diverse cues.



(Smoothness weight λ) (Number of stars)

Figure 7: **Importance of parameters: (Left)** smoothness weight for the continuous MRF, and **(Right)** the number of stars for the topology potential. Result are on validation set.

Qualitative results: Fig. 8 depicts segmentation results when using our full model with stereo and LIDAR (4th and 6th columns respectively). Our model is robust to low-resolution imagery, saturation, noise, sparse point clouds, depth estimation errors and can successfully segment out occluders. The last three rows show failure modes. Our approach has difficulty with very small cars that are heavily occluded, and point clouds with a large number of outliers.

5. Conclusions

We have presented an approach that can generate segmentations of the same quality as human labelers (i.e., MTurkers) using only weak supervision in the form of 3D bounding boxes. Towards this goal, we have exploited appearance models, stereo and/or noisy point clouds, a repository of 3D CAD models as well as topological constraints. In the future we plan to exploit video information in order to automatically label moving objects over time.

Acknowledgements The first author was partly supported by ARO 62250-CS and ONR N000014-10-1-0933. The authors thank Kaustav Kundu, Wenjie Luo, Chen Kong, Zhou Ren, Jia Xu, Meng Ye, and Edgar Simo-Serra for helping us with annotation.

References

[1] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, 2001. 2, 3

Model	Appearance	Smooth	Depth seeds	Dense depth	Topo	CAD	Stereo IOU (%)	LIDAR IOU (%)
Basic (no ising)	✓	✓					60.1	61.1
Basic	✓	✓					61.6	62.1
Topology	✓	✓			✓		64.4	64.2
Dense depth	✓	✓		✓			68.4	71.3
Discrete depth	✓	✓	✓				–	74.2
All depth	✓	✓	✓	✓			–	76.2
CAD	✓	✓				✓	82.9	83.5
Full Model w/o CAD	✓	✓	✓	✓	✓		70.2	77.6
Full Model w/o Dense depth	✓	✓	✓		✓	✓	83.4	84.1
Full Model w/o Depth Seeds	✓	✓		✓	✓	✓	–	85.4
Full Model w/o Topology	✓	✓	✓	✓		✓	84.2	85.5
Full Model	✓	✓	✓	✓	✓	✓	**84.4**	**85.6**

Table 2: **Importance of Features:** Every feature helps, and the CAD model potential helps the most. Notably, performance with stereo is similar to LIDAR. Note that when using stereo, we do not use Depth seeds.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004. 2, 4

[3] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008. 2

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 1

[5] J. Diebel and S. Thrun. An application of markov random fields to range sensing. In *NIPS*, 2006. 2

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

[7] P. F. Felzenszwalb and D. P. Huttenlocher. Distance transforms of sampled functions. *Theory of Computing*, 2012. 4

[8] S. Fidler, S. Dickinson, and R. Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NIPS*, December 2012. 3, 4

[9] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013. 1

[10] D. Freedman and T. Zhang. Interactive graph cut based segmentation with shape priors. In *CVPR*, 2005. 2

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 2, 4

[12] A. Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *ICCV Workshops*, 2009. 2

[13] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010. 2, 3

[14] R. Huang, V. Pavlovic, and D. N. Metaxas. A graphical model framework for coupling mrfs and deformable models. In *CVPR*, 2004. 2

[15] B. Kim, S. Xu, and S. Savarese. Accurate localization of 3D objects from RGB-D data using segmentation hypotheses. In *CVPR*, 2013. 1, 2

[16] P. Kohli, J. Rihan, M. Bray, and P. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *IJCV*, 79(3):285–298, 2008. 2

[17] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011. 2

[18] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in imagenet. In *ECCV*, 2012. 2

[19] M. Kumar, P. Torr, and A. Zisserman. Objcut: Efficient segmentation using top-down and bottom-up cues. *PAMI*, 32(3):530–545, 2010. 2

[20] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009. 2

[21] J. Liebelt and C. Schmid. Multi-view object class detection with a 3d geometric model. In *CVPR*, 2010. 2

[22] J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *ICCV*, 2013. 2

[23] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *ICCV*, 2013. 2

[24] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 2, 3, 5

[25] S. Satkin, J. Lin, and M. Hebert. Data-driven scene understanding from 3D models. In *BMVC*, 2012. 2

[26] A. Saxena, S. Chung, and A. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 2007. 2

[27] A. Schwing, S. Fidler, M. Pollefeys, and R. Urtasun. Box in the box: Joint 3d layout and object reasoning from single images. In *ICCV*, 2013. 4, 5

[28] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2004. 4

[29] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, September 2005. 4

[30] O. Veksler. Star shape prior for graph-cut image segmentation. In *ECCV*, 2008. 2, 3

[31] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008. 2

[32] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013. 2

[33] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*, 2009. 2

[34] X. Xiong and D. Huber. Using context to create semantic 3d models of indoor environments. In *BMVC*, 2010. 2

[35] J. Xu, A. Schwing, and R. Urtasun. Tell me what you see and i will show you where it is. In *CVPR*, 2014. 2

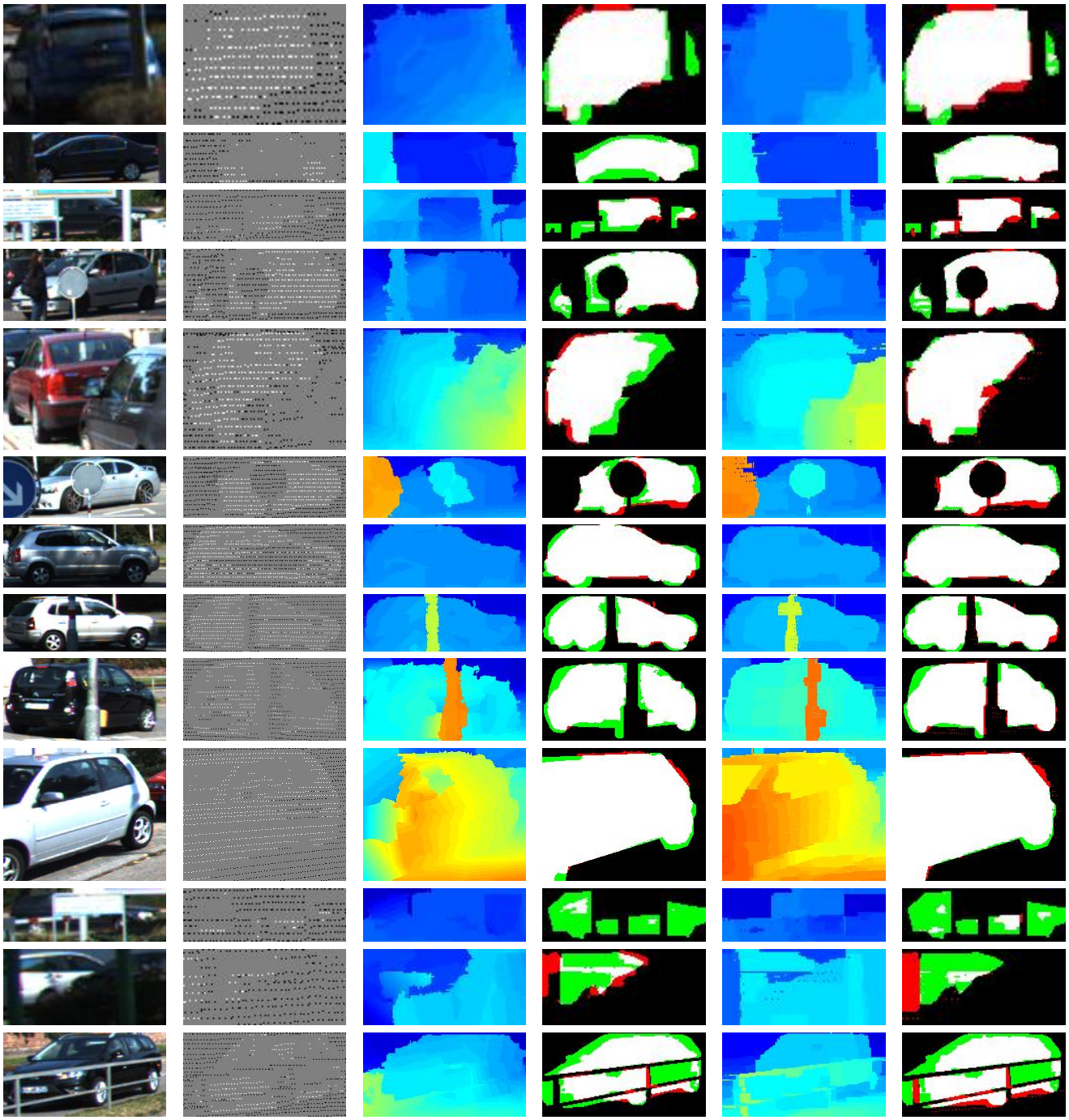


Figure 8: **Segmentation results:** Each row shows the image, LIDAR points (White: car, Black: bckgr.), stereo depth, results of our full model with stereo (White: True Positive, Black: True Negative, Red: False Positive, Green: False Negative), depth images reconstructed by Laplacian MRF, and results of our full model with LIDAR. Last three rows show our failure modes.

[36] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, 2013. 2

[37] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 1, 2

[38] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S.-C. Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *CVPR*, 2013. 2

[39] Z. Zia, M. Stark, B. Schiele, and K. Schindler. Detailed 3d representations for object recognition and modeling. *PAMI*, 2013. 2