Instance-Level Segmentation for Autonomous Driving with Deep Densely Connected MRFs

Ziyu Zhang Sanja Fidler Raquel Urtasun
Department of Computer Science, University of Toronto
{zzhang, fidler, urtasun}@cs.toronto.edu

Abstract

Our aim is to provide a pixel-wise instance-level labeling of a monocular image in the context of autonomous driving. We build on recent work [32] that trained a convolutional neural net to predict instance labeling in local image patches, extracted exhaustively in a stride from an image. A simple Markov random field model using several heuristics was then proposed in [32] to derive a globally consistent instance labeling of the image. In this paper, we formulate the global labeling problem with a novel densely connected Markov random field and show how to encode various intuitive potentials in a way that is amenable to efficient mean field inference [15]. Our potentials encode the compatibility between the global labeling and the patch-level predictions, contrast-sensitive smoothness as well as the fact that separate regions form different instances. Our experiments on the challenging KITTI benchmark [8] demonstrate that our method achieves a significant performance boost over the baseline [32].

1. Introduction

Object detection is one of the fundamental open problems in computer vision. The main objective is to place tight bounding boxes around each object of interest. In the past two years, detection performance has almost doubled thanks to the availability of large datasets as they enable training very deep representations [16, 27]. While detection might have been a good proxy when performance was low, recent work has been trying to go beyond simple boxes by providing a detailed segmentation mask for each object instance [12, 30, 28, 13, 29].

A mask is in many ways richer than a box: it allows an informed reasoning about occlusion and depth layering. For robotics applications where depth is available, it further enables a more precise 3D localization and segmentation [11] which is important for, e.g., obstacle avoidance, route planning and object grasping. An instance mask is also more informative than pixel-wise class labeling as it allows counting, important for applications such as retrieval [21].

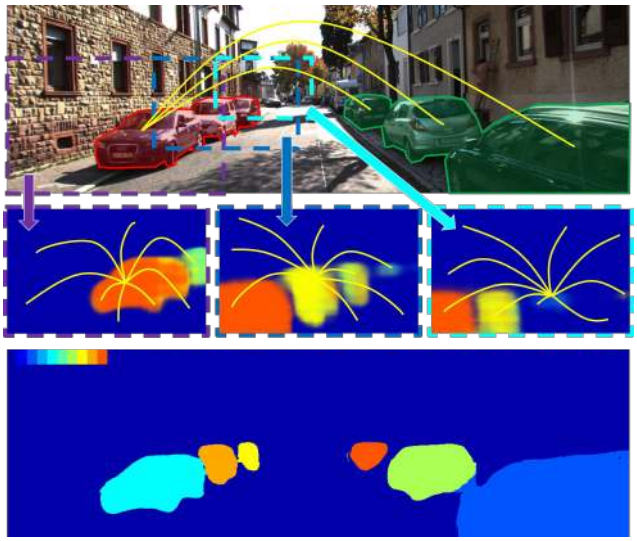


Figure 1: Our approach densely samples patches of different sizes from the image (row 1) and exploits a CNN to provide a soft instance labeling of each patch (row 2). Our MRF connects all pixel pairs inside the patches (yellow curves in row 2), as well as all pixel pairs from far away connected components obtained from patch-level CNN predictions (yellow curves in row 1), to provide a globally consistent instance labeling of the image (row 3).

Instance segmentation has been addressed in a variety of ways. In interactive segmentation, approaches like Grabcut [24, 4] require a user-supplied box, or a scribble on the foreground and background, in order to segment the objects. This is typically done with graph-cuts by combining appearance cues and smoothness. The most common approach to instance segmentation has been to utilize object detections and top-down shape priors to label pixels inside each detected box [17, 9]. Methods that jointly reason about instance labeling and, possibly, depth ordering given object detections and class-level semantic segmentation have shown improved performance [28]. Recently, approaches that train CNNs to predict instance labeling inside densely sampled image patches have shown very promising performance [32]. However, deriving a globally consistent instance labeling of the image from local predictions is a

challenging open problem.

In this paper, our goal is to estimate an accurate pixel-level labeling of object instances from a single monocular image in the context of autonomous driving. We propose a method that combines the soft predictions of a neural net run on many overlapping patches into a consistent global labeling of the entire image. We formulate this problem as a densely connected Markov random field (MRF) with several potentials encoding consistency with local patches, contrast-sensitive smoothness as well as the fact that separate regions form different instances. An overview of our MRF is given in Fig. 1. Our main technical contribution is a formulation that encodes all potentials in a way that is amenable to efficient mean field inference [15], and we go beyond [15] by including Potts potentials as well. Our experimental evaluation shows significant improvements over [32] on the challenging KITTI benchmark [8].

2. Related Work

We focus our review on techniques operating on a single monocular image, and divide them into different types.

Instance-Level Segmentation by Detection. The most common approach to object instance segmentation is to first localize objects with a set of bounding boxes, and then exploit top-down information such as object's shape and appearance in order to accurately segment the object within each box [17]. [18, 31] proposed an MRF model to identify which detections are true positive, and output pixel-wise class labels, thus performing instance segmentation. [22] votes for object centers and uses a MRF to infer the assignment of pixels to object centers. R-CNN [9] was used in [12] to first generate object proposals, and then predictions from two CNNs (box and region-based) were fused to segment the object inside each box. This idea was extended to RGB-D in [11].

In [30], the authors propose a generative model that takes as input candidate detections and jointly assigns a pixel to an object instance as well as a depth layer. Inference is performed with coordinate ascent iterating between updating the layer assignment and the parameters of the appearance models. In contrast, we leverage the efficient inference algorithm for Gaussian MRFs [15] to work directly on the pixel level (rather than layers), thus allowing more freedom in the final label assignment. In [28], semantic segmentation and object detection are run as a first step. The method then solves for instance labeling and depth ordering by minimizing an integer quadratic program. In our approach we use a CNN trained to directly predict instance labeling in a stride of local patches, and then solve for a consistent instance labeling using our densely connected MRF, thus not requiring to explicitly perform object detection.

3D CAD Models. Another line of work matches CAD models to images [2, 20, 10]. An image-aligned CAD model effectively provides an instance labeling of the image. CAD matching is however typically slow, and not very robust, as there is a large difference between the appearance of the synthetic models and objects in real images. Instead of matching CAD models, [14] retrieves object segments from a dataset of labeled objects. Their probabilistic model then aims to find segments that optimally transform into the given image by respecting typical 3D relations. The output is an instance labeling of the image (a "scene collage").

Interactive Segmentation. Instance-level segmentation has also been done without prior knowledge about how the object looks like. In this line of work, the techniques rely on a user-supplied box, or a scribble on the foreground and background, and then derive the pixel-wise labeling of the object instance. For example, GrabCut [24] utilizes annotations in the form of 2D bounding boxes, and computes the foreground/background models using EM. [3] relies on scribbles as seeds to model appearance of foreground/background, and uses graph-cuts by combining appearance cues and a smoothness term [4].

Instances without Object Detection. Recent work has tried to explicitly reason about instance segmentation (no class detectors need to be run in advance). [26] makes an optimal cut in a hierarchical segmentation tree to obtain object instance regions. [19] trained a multi-output CNN that jointly predicts pixel-level class labeling of the image as well as bounding box locations and object instance numbers. Off-the-shelf clustering is used to derive the final object instance labeling of the image. In our work, we exhaustively sample bounding boxes and softly score each pixel belonging to a particular object instance. Our main efforts are then devoted to "clustering" (merging the predictions) which in our work is done via a densely connected MRF. Parallel to our work, [23] proposes a recurrent neural net to label object instances sequential by keeping a memory of which pixels have been labeled so far.

We build on [32] which trains a CNN on local patches to obtain a depth-ordered pixel-wise instance labeling of each patch. [32] then uses a MRF along with a connected component algorithm to merge predictions in the possibly overlapping patches into a global instance labeling of the image. In our paper, we propose a densely connected MRF that exploits fast inference [15], and provides significantly better segmentations due to the dense connectivity in the model.

3. Object Instance Labeling

The goal of this paper is to perform instance-level segmentation given a single monocular image. We follow [32] and learn deep representations to perform this task. Our contribution is then a novel densely connected Markov ran-

dom field that is able to produce a single coherent explanation of the full image and amenable to the efficient mean field inference algorithm [15]. As shown in our experimental evaluation the estimates provided by our approach are significantly better than those of [32] in most metrics.

3.1. Deep Learning for Instance-level Segmentation

We follow [32] to both generate surrogate ground truth and train the CNN. We provide the details for completeness. We generate training examples by extracting overlapping patches at multiple resolutions. Since KITTI does not have instance-level segmentations, we use [5, 7] to obtain the segmentations for our training set. We label the instances according to their depth within the patch. Thus instances farther away from the camera will get higher IDs. Using depth ordering is important to produce a single labeling, breaking the symmetry of permutations of instance-level segmentation. E.g., two instances can be labeled as either (1,2) or (2,1). We then train a CNN to output a pixel-level instance labeling inside each patch. We use the architecture from [25] pre-trained on ImageNet and fine-tune it for instance-level segmentation using our surrogate ground truth. The CNN gives us (probabilistic) pixel-level predictions of instances at the patch level. We propose a model to merge all the local predictions and produce a globally consistent image labeling. This is the contribution of our paper.

3.2. Densely Connected Pixel-wise MRF

Given an image $\mathbf{x}$, we index the image patches with z. Let $\mathcal{P}_z$ be the set of pixels in patch z. Let $\mathbf{p}_{z,i}$ be the output of the softmax for the i-th pixel when we apply the CNN to patch z. Note that the CNN predicts up to 5 instances as well as background. Thus $\mathbf{p}_{z,i}$ is a 6-D vector. The goal is to merge all the patch-level information and come up with a single explanation of the image in terms of all instances. We restrict the maximum number of instances to be 9 per image, which is sufficient for KITTI. Thus our global label space $\mathcal{L}_{\text{g}}$ is $\{0, 1, \cdots, 9\}$ with 0 encoding background. Let $\mathbf{y}$ be the labeling of each pixel in the image with $y_i \in \mathcal{L}_{\text{g}}$. Unlike [32], we are not interested in ordering the instances by depth. Thus any labeling that separates instances is valid.

We propose a novel densely connected pixel-wise MRF to solve for the problem of labeling the full image given the local patch-based predictions. The corresponding Gibbs energy $E(\mathbf{y})$ of our MRF consists of three main terms: a pairwise *smoothness* term, a pairwise *local CNN prediction* term and a pairwise *inter-connected component* term, each encoding different intuitions about the problem:

$$E(\mathbf{y}) = E_{\text{smo}}(\mathbf{y}) + E_{\text{cnn}}(\mathbf{y}) + E_{\text{icc}}(\mathbf{y}). \tag{1}$$

Note that all terms are defined over densely connected pixel pairs. We cannot use the CNN output as a unary potential, as the label space of the local patches and the global image are different, i.e., only 6 labels (including background) possible locally, and instance 2 in a local patch might be totally different from instance 2 in another patch far away.

We now describe each term in more details.

3.2.1 Smoothness: $E_{\text{smo}}(\mathbf{y})$

Following [15], we incorporate a contrast-sensitive smoothness term into our MRF to remove noisy tiny regions. The idea is to describe each pixel with a feature vector, and define a potential that encourages pixels with similar features to be more likely assigned the same label. The typical feature for each pixel has been color and position on image.

In our problem we use the combination of position and the output of the CNN to form our feature space. Our CNN is trained to differentiate between object instances, so the probability vectors that the CNN outputs are a very strong cue of how likely two pixels belong to the same object. Further, we use the position feature so that the smoothness has a lower influence between far apart regions in order not to over-smooth the result. Notice that we do not use color as a feature. This is because different object instances can take similar colors, and color may be somewhat deceiving due to shadows, saturation and specularities.

Formally, let $\mathbf{d}_i$ be the 2-D position vector for pixel i in the image. We define the contrast-sensitive smoothness term as a sum of patch-specific contrast-sensitive smoothness terms, each defined over all pixel pairs in the patch:

$$E_{\text{smo}}(\mathbf{y}) = \sum_{z} \sum_{i,j:i,j\in\mathcal{P}_z, i<j} \varphi_{\text{smo}}^{(z,i,j)}(y_i, y_j), \tag{2}$$

where the potential is defined as

$$\varphi_{\text{smo}}^{(z,i,j)}(y_i, y_j) = w_{\text{smo}} \mu_{\text{smo}}(y_i, y_j) k_{\text{smo}}\left(\mathbf{f}_i^{(z)}, \mathbf{f}_j^{(z)}\right). \tag{3}$$

Here w_{smo} is the weight for the potential (which we learn) controlling the degree of smoothness, and k_{smo} denotes a Gaussian kernel defined as

$$k_{\text{smo}}\left(\mathbf{f}_i^{(z)}, \mathbf{f}_j^{(z)}\right) = \exp\left(-\frac{\|\mathbf{p}_{z,i} - \mathbf{p}_{z,j}\|_2^2}{2\theta_1^2} - \frac{\|\mathbf{d}_i - \mathbf{d}_j\|_2^2}{2\theta_2^2}\right),$$

where $\mathbf{f}_i^{(z)}$ contains both the position $\mathbf{d}_i$ and the output of the CNN $\mathbf{p}_{z,i}$. Note that θ_1 and θ_2 scale the features to reflect our notion of "closeness" in the feature space. Finally, the compatibility function $\mu_{\text{smo}}(y_i, y_j)$ in the potential takes the form of the Potts model:

$$\mu_{\text{smo}}(y_i, y_j) = \begin{cases} 1, & \text{if } y_i \neq y_j \\ 0, & \text{otherwise} \end{cases}.$$

This penalizes two pixels with similar positions and CNN predictions to have different labels.

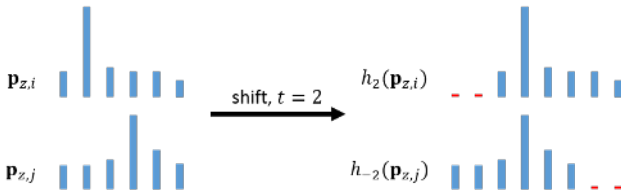


Figure 2: Function $h_2(\cdot)$ **prepends** 2 zeros to vector $\mathbf{p}_{z,i}$ while $h_{-2}(\cdot)$ **appends** 2 zeros to vector $\mathbf{p}_{z,j}$. This is equivalent to shifting $\mathbf{p}_{z,i}$ by 2 units towards right and zero-pad both vectors to make them aligned. The shift matches the modes of $\mathbf{p}_{z,i}$ and $\mathbf{p}_{z,j}$. We input this shifted vector pair into our Gaussian kernel.

3.2.2 Local CNN Prediction: $E_{\text{cnn}}(\mathbf{y})$

Any given patch contains only a subset of the instances present in the full image. When we produce our image-level labeling, we would like to maintain the separation into different instances estimated at the local level (via the CNN), while producing a coherent labeling across all patches. For example, if the CNN predicts that in patch z pixel i belongs to instance 1 while pixel j belongs to instance 2, then any global configuration with $y_i \neq y_j$ should be encouraged. However, it turns out to be important to have some preference over the ordering just to break the symmetry in our model to kick off the inference algorithm. We thus encourage ordering in the global labeling (in this example $y_i < y_j$).

To encode this patch-image compatibility in our energy, we define the local CNN prediction term as a sum of patch-specific compatibility terms, each defined over all pixel pairs in the patch:

$$E_{\text{cnn}}(\mathbf{y}) = \sum_{z} \sum_{i,j:i,j\in\mathcal{P}_z, i<j} \varphi_{\text{cnn}}^{(z,i,j)}(y_i, y_j). \tag{4}$$

The potential $\varphi_{\text{cnn}}^{(z,i,j)}(y_i, y_j)$ should ideally encode the fact that we want global instance labeling to agree with local predictions. That is, if two pixels are likely to be of the same (different) instance at the local level, they should also be the same (different) at the global level. This could be simply encoded with a compatibility potential that y_i and y_j are encouraged to have the same label if the output of the CNN $\mathbf{p}_{z,i}$ and $\mathbf{p}_{z,j}$ are similar, and their relative ordering ($y_i > y_j$ or vice versa) is respected if the CNN predicts them to be of different instances. This naive approach, however, will break the efficiency of inference, as we will no longer be able to use Gaussian filtering. Gaussian filtering is however crucial, since our local CNN prediction term is fully connected at patch level.

Instead, one of the main contributions of our paper is to approximate such compatibility potentials as a series of Gaussian potentials. Each potential is composed of a Gaussian kernel applied to a shifted version of the local softmax probabilities:

$$\varphi_{\text{cnn}}^{(z,i,j)}(y_i, y_j) = \sum_{t=-T}^{T} \varphi_{\text{cnn}}^{(z,t,i,j)}(y_i, y_j), \tag{5}$$

where T is the maximum shift allowed (fixed to be 2 in our experiments).

We define the shifted pairwise potential $\varphi_{\text{cnn}}^{(z,t,i,j)}(y_i, y_j)$ as a product of its weight, a compatibility function and a Gaussian kernel defined over pairs of shifted local CNN predictions:

$$\varphi_{\text{cnn}}^{(z,t,i,j)}(y_i, y_j) = w_{\text{cnn}}^{(s(z))} \mu_{\text{cnn}}^{(t)}(y_i, y_j) k_{\text{cnn}}^{(t)}(h_t(\mathbf{p}_{z,i}), h_{-t}(\mathbf{p}_{z,j})), \tag{6}$$

where $w_{\text{cnn}}^{(s(z))}$ is the weight which depends on the size $s(z)$ of patch z, and $k_{\text{cnn}}^{(t)}$ is a Gaussian kernel characterized by its precision matrix $\Lambda_{\text{cnn}}^{(t)}$.

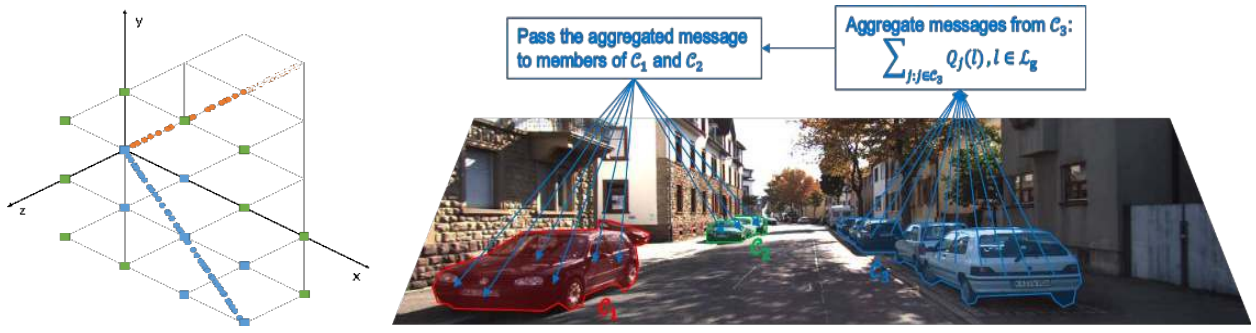
When $t > 0$, a shift towards right is applied to $\mathbf{p}_{z,i}$ to create $h_t(\mathbf{p}_{z,i})$ while a shift towards left is applied to $\mathbf{p}_{z,j}$ to create $h_{-t}(\mathbf{p}_{z,j})$. Note that shifting by t requires **prepending** t zeros, while shifting by $-t$ requires **appending** t zeros. We refer the reader to Fig. 2 for a visualization of this idea. If the modes of $\mathbf{p}_{z,i}$ and $\mathbf{p}_{z,j}$ match for any positive t, it means that the label of pixel i is predicted to be smaller than pixel j in patch z by the CNN. This is the case shown in Fig. 2. Therefore, globally we prefer any configuration with $y_i < y_j$. The reverse is also true that if we achieve a good match with a negative t, then we prefer any configuration with $y_i > y_j$. If the best match is achieved without shift, it means that we prefer $y_i = y_j$. This can be encoded via the following compatibility function:

$$\mu_{\text{cnn}}^{(t)}(y_i, y_j) = \begin{cases} -1, & \text{if } y_i < y_j, t > 0 \\ -1, & \text{if } y_i > y_j, t < 0 \\ -1, & \text{if } y_i = y_j, t = 0 \\ 0, & \text{otherwise} \end{cases}. \tag{7}$$

Note that a negative value in $\mu_{\text{cnn}}^{(t)}(y_i, y_j)$ implies that we encourage the configuration.

3.2.3 Inter-Connected Component: $E_{\text{icc}}(\mathbf{y})$

So far our MRF encourages smoothness as well as that global instance labeling to agree with local predictions. However, nothing prevents instances that are far apart and thus do not appear together in any patch from having the same label. Towards this goal, for each pixel i, we compute the probability that it belongs to foreground, by summing the output of local CNN predictions and re-normalizing. By thresholding this probability, we obtain a binary mask of activation. We index each connected component of the foreground of the binary mask with m, and the pixels it contains with $\mathcal{C}_m$. Each component might contain more than one car.



(a) (b)

Figure 3: **(a)** Suppose that $\{\mathbf{p}_{z,i}\}$ are positive scalars. When $t = 1$, both $h_t(\mathbf{p}_{z,i})$ and $h_{-t}(\mathbf{p}_{z,j})$ are 2-D. The corresponding permutohedral lattice (in gray) lies on a 2-D hyperplane in a 3-D space. $\{h_{-t}(\mathbf{p}_{z,j})\}$ are embedded in the hyperplane (blue dots). The embeddings follow a 1-D subspace due to zero-appending. Value of $\{Q_j(l')\}$ splats onto the vertices of their respective enclosing simplexes (blue squares). After Gaussian blurring, vertices shown as green squares also get non-zero value. An extra step is to embed $\{h_t(\mathbf{p}_{z,i})\}$ (red solid dots and red open dots). The embeddings lie on another 1-D subspace due to zero-prepending. Finally, convolution is evaluated at the new embeddings. Only red solid dots get non-zero values, while red open dots get a zero due to the inactivity of the vertices of their enclosing simplex. **(b)** Three car blobs are shown in the image. During each mean field update, we sum up individual messages from members of $\mathcal{C}_3$ and pass the summation to each member of $\mathcal{C}_1$ and $\mathcal{C}_2$. Similarly for messages from $\mathcal{C}_1$ to $\mathcal{C}_2$ and $\mathcal{C}_3$, and messages from $\mathcal{C}_2$ to $\mathcal{C}_1$ and $\mathcal{C}_3$.

However, it is reasonable to assume that each instance will never appear in two different components. We encode this in the inter-connected component term as a sum of terms defined over component pairs, and each of the terms fully connects cross-component pixel pairs:

$$E_{\text{icc}}(y) = \sum_{m,n:m<n} \sum_{i,j:i\in\mathcal{C}_m, j\in\mathcal{C}_n} w_{\text{icc}}\mu_{\text{icc}}(y_i, y_j), \quad (8)$$

with w_{icc} the weight and $\mu_{\text{icc}}(y_i, y_j)$ a Potts potential

$$\mu_{\text{icc}}(y_i, y_j) = \begin{cases} 1, & \text{if } y_i = y_j \\ 0, & \text{otherwise} \end{cases}. \quad (9)$$

While this potential is not Gaussian, and we have dense connections, in the next section we show that the updates can still be computed in linear time.

3.3. Efficient Inference

Inference in our model consists of estimating the minimum energy configuration

$$\mathbf{y}^* = \underset{\mathbf{y}}{\text{argmin}}\, E_{\text{smo}}(\mathbf{y}) + E_{\text{cnn}}(\mathbf{y}) + E_{\text{icc}}(\mathbf{y}).$$

Unfortunately this is NP-hard. Instead, we perform efficient approximated inference via mean field. Towards this goal, we approximate the Gibbs distribution $P(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})}\exp(-E(\mathbf{y}|\mathbf{x}))$ with a fully decomposable distribution $Q(\mathbf{y}|\mathbf{x}) = \prod_i Q_i(y_i|\mathbf{x})$. Note that we drop the conditioning from now on to simplify notation.

Mean field computes updates by iteratively minimizing the KL-divergence between the approximated distribution $Q(\mathbf{y})$ and the true distribution $P(\mathbf{y})$. We use an iterative algorithm which updates the local distributions $\{Q_i(y_i)\}$ in parallel. In our model, the updates can be derived as

$$\begin{aligned} &\log Q_i(y_i = l) \\ &= -\sum_{z:i\in\mathcal{P}_z} \sum_{l':l'\in\mathcal{L}_g} \sum_{j:j\in\mathcal{P}_z, j\neq i} \varphi^{(z,i,j)}_{\text{smo}}(l,l')Q_j(l') \quad (10) \\ &\quad -\sum_{z:i\in\mathcal{P}_z} \sum_{t=-T}^{T} \sum_{l':l'\in\mathcal{L}_g} \sum_{j:j\in\mathcal{P}_z, j\neq i} \varphi^{(z,t,i,j)}_{\text{cnn}}(l,l')Q_j(l') \quad (11) \\ &\quad - w_{\text{icc}} \sum_{n:n\neq m, i\in\mathcal{C}_m} \sum_{j:j\in\mathcal{C}_n} Q_j(l) - \log(Z_i), \quad (12) \end{aligned}$$

with Z_i the local partition function which is easily computable as it only depends on a single node. We refer the reader to suppl. material for the derivation of these updates. We use the uniform distribution as our initialization. We now describe how to compute the updates efficiently.

Smoothness. Eq. (10) can be computed efficiently using the same high-dimensional Gaussian filtering algorithm of [15]. This results in linear updates in the number of pixels.

Local CNN Prediction. The inner most summation over pixels in Eq. (11) is the fundamental building block for the computation of the entire term. Explicitly it is given as

$$w^{(s(z))}_{\text{cnn}}\mu^{(t)}_{\text{cnn}}(l,l') \sum_{j:j\in\mathcal{P}_z, j\neq i} k^{(t)}_{\text{cnn}}(h_t(\mathbf{p}_{z,i}), h_{-t}(\mathbf{p}_{z,j}))Q_j(l').$$

This can be interpreted as a convolution with a Gaussian kernel $G_{\Lambda^{(t)}_{\text{cnn}}}$ evaluated at $h_t(\mathbf{p}_{z,i})$. Since Gaussian convolution is essentially a low-pass filter, by the sampling theorem we can convolve a downsampled $\{Q_j(l')\}$ with the Gaussian kernel, and upsample the output to compute convolution at $h_t(\mathbf{p}_{z,i})$. Following [15], we use the efficient permutohedral lattice data structure [1] to perform downsampling,

		Cls. Eval	Instance Evaluation								
		IoU	MWCov	MUCov	AvgPr	AvgRe	AvgFP	AvgFN	InsPr	InsRe	InsF1
[32]	ConnComp	77.1	66.7	49.1	82.0	60.3	0.465	0.903	49.1	43.0	45.8
	Unary	77.6	65.0	48.4	81.7	62.1	0.389	0.688	46.6	42.0	44.2
	Unary+LongRange	77.6	66.1	49.2	82.6	62.1	0.354	0.688	48.2	43.1	45.5
	Unary+LR (w/ DeepLab [6])	77.7	68.2	50.2	**85.3**	63.2	**0.285**	**0.562**	39.5	40.1	39.8
Ours	LocCNNPred	77.4	58.3	40.9	80.4	62.6	0.403	0.681	25.3	32.9	28.6
	LocCNNPred+InterConnComp	76.8	65.7	50.3	79.9	**63.4**	0.507	0.618	35.8	46.4	40.4
	Full	77.1	69.3	50.6	80.5	57.7	0.451	1.076	56.3	47.4	51.5
	Full (w/ DeepLab [6])	**78.5**	**73.7**	**54.3**	82.8	61.3	0.458	0.812	**63.3**	**51.6**	**56.8**
	With Post-processing										
[32]	ConnComp	77.2	66.8	49.2	81.8	60.3	0.465	0.903	49.8	43.0	46.1
	Unary	77.4	66.7	49.8	81.6	61.2	0.562	0.840	44.1	44.7	44.4
	Unary+LongRange	77.4	67.0	49.8	82.0	61.3	0.479	0.840	48.9	43.8	46.2
	Unary+LR (w/ DeepLab [6])	77.3	70.9	52.2	**85.7**	61.7	0.597	0.736	40.2	45.9	42.8
Ours	LocCNNPred	76.7	67.5	52.9	82.5	61.3	0.646	0.743	39.4	51.6	44.7
	LocCNNPred+InterConnComp	76.3	68.1	53.9	80.7	**62.2**	0.708	**0.701**	42.1	52.2	46.6
	Full	77.0	69.7	51.8	83.9	57.5	**0.375**	1.139	65.3	50.0	56.6
	Full (w/ DeepLab [6])	**78.5**	**74.1**	**55.2**	84.7	61.3	0.417	0.833	**70.9**	**53.7**	**61.1**

Table 1: Instance-level and Class-level Evaluation on the **Test Set** (144 images). See text for the explanation of the metrics. For 'AvgFP' and 'AvgFN' smaller is better, for the rest higher is better.

convolution and upsampling. In the standard case of Gaussian filtering with permutohedral lattice, we embed a set of features encoding the position of $\{Q_j(l')\}$ in the hyperplane in which the lattice lies. We then downsample by splatting the value of each $Q_j(l')$ onto the vertices of its enclosing simplex with barycentric weights. Then Gaussian blurring is performed over lattice points along each lattice direction. The final result is then computed by gathering values from lattice points for the already embedded features. Due to the fact that we introduced different shifts for the two elements of the kernel $h_t(\mathbf{p}_{z,i})$ and $h_{-t}(\mathbf{p}_{z,j})$, apart from the set of features $\{h_{-t}(\mathbf{p}_{z,j})\}$ we need to embed in the first place, we have to embed an extra set of features $\{h_t(\mathbf{p}_{z,i})\}$ at which we evaluate the convolution, in contrast to the standard case. An example is in Fig. 3a. As in the standard case, we first embed $\{h_{-t}(\mathbf{p}_{z,j})\}$ encoding the position of $\{Q_j(l')\}$ in the hyperplane. Note, however, that the features lie in a subspace of the hyperplane as they are padded with zeros (e.g., a line on a plane in Fig. 3a). We then distribute the value of each $Q_j(l')$ onto the vertices of its enclosing simplex. This is followed by the filtering step over the lattice. As an extra step in contrast to the standard case, we now need to embed the first element of the kernel, i.e., $\{h_t(\mathbf{p}_{z,i})\}$, in the hyperplane, and the embeddings lie in another subspace. Finally we evaluate the convolution at the new embeddings.

Inter-Connected Component. The first term in Eq. (12) is not a Gaussian kernel, however it is densely connected. This means that potentially we have an update quadratic in the number of pixels. We exploit the fact that all members within a connected component have the exact same pairwise interaction with all other pixels not in the component. This implies that all members within a connected component receive the exact same messages passed from other components during each update. Note that it is linear to sum up the individual messages within a connected component and pass this summation $\sum_{j:j\in\mathcal{C}_n} Q_j(l)$ to the members of other components. We visualize the message passing procedure for the inter-connected component potential in Fig. 3b.

4. Experimental Evaluation

We evaluate our approach on the challenging KITTI benchmark [8]. In particular, we use a subset of $3,524$ images from 55 videos and divide the images into training/validation/test sets such that given a video, all its images are exclusively contained in only one of the three sets. Altogether, we use 3260 images for training, 120 for validation, and 144 for testing. 131 images from either our validation or test set have been manually annotated with pixel-wise instance labeling for *Cars* by [5]. We labeled the rest of the 133 images. Thus all validation and test images have ground truth annotations.

Implementation Details. We generate surrogate ground truth for our training images with [5] and train our CNN as in [32]. In another experiment we also use the architecture DeepLab-LargeFOV from [6]. By changing the CNN architecture from the naive adaptation of VGG-16 by [25] to DeepLab-LargeFOV (denoted by 'w/ DeepLab [6]' in our results), we observe substantial performance gain for our approach but a slight drop for the baselines. Thus we report results with both architectures. For our validation/test images (with typical size 375×1242), we extract densely overlapping patches of three sizes: large (270×432), medium (180×288), and small (120×192) in a sliding window fashion. We run the extracted patches through the CNN to

obtain local instance predictions. Following [15], we apply a pixel-wise normalization to Gaussian kernels. We also normalize the aggregated message of each connected component by the number of pixels it contains. Normalization is able to cancel the bias caused by highly variable instance sizes. We tune all weights, hyper-parameters and kernel widths in our MRF on the validation set. We fix the number of iterations of the mean field update to be 50 in all experiments in the paper.

Baselines. We re-train the approach (in three different instantiations) proposed in [32] on our validation set to obtain three strong baselines. Note that [32] and our method use the exact same CNN unaries and patch extraction method, so we evaluate the two different MRFs on equal footing. The first baseline 'ConnComp' is the 'connected components ordering' potential in [32] which applies a connected component algorithm to heuristically merged object instances and orders them according to their positions along the vertical axis. The second baseline 'Unary' additionally adds the 'CNN energy' potential in [32]. The third baseline 'Unary+LongRange' further adds the pairwise 'long-range connections' from [32]. We found that their pairwise 'short-range connections' generally hurts performance, so we do not include it in our baselines.

Evaluation Metrics. Following [32], we use a number of metrics to provide a comprehensive evaluation of our model. We divide the metrics into two categories, namely class-level (i.e., *Car* vs. *non-Car*) and instance-level. For the *class-level evaluation*, we report the standard intersection-over-union score for the foreground 'FIoU'.

For the *instance-level evaluation*, we provide the mean-weighted-coverage score and the mean-unweighted-coverage score introduced in [26], which we denote by 'MWCov' and 'MUCov' respectively. For each ground-truth instance in a given image, we find its maximally overlapping prediction and compute the IoU score between them. The weighted-coverage score for the image is then the average of the IoU scores weighted by the size of the ground-truth instances. Finally, 'MWCov' is obtained by averaging the weighted-coverage score across images. The mean-unweighted-coverage score is computed similarly but treats every ground-truth instance equally. 'MWCov' and 'MUCov' are important, because they directly evaluate how closely our predictions overlap with ground-truth instances, since these two metrics are based on IoU scores. However, they do not penalize false positive instances.

For each predicted instance, we compute the ratio of class-level (*Car* vs. *non-Car*) true positive pixels inside it. The ratio is then averaged across predicted instances and reported as 'AvgPr'. Similarly for each ground-truth instance, we compute the ratio of true positive pixels inside it. The ratio is then averaged across all ground-truth instances and reported as 'AvgRe'.

If a predicted instance does not overlap with any ground-truth instance, we deem it as a false positive instance. We average the number of false positive instances in each image across images, and report it as 'AvgFP'. Similarly, if a ground-truth instance does not overlap with any prediction, we deem it as a false negative instance. We average the number of false negative instances in each image across images, and report it as 'AvgFN'.

Finally, for each ground-truth instance, we find a prediction which overlaps more than 50% with it. We divide the number of such GT-prediction pairs either by the number of predictions to obtain instance-level precision denoted by 'InsPr', or by the number of ground-truth instances to obtain instance-level recall denoted by 'InsRe', and report the corresponding F1 score denoted by 'InsF1'. Intuitively, 'InsPr' and 'InsRe' reflect the model's ability to avoid false positive instances and false negative instances respectively at a 50% threshold. The corresponding 'InsF1' score unifies the previous two metrics. 'InsF1' score (on the validation set) is the metric we use for selecting our model parameters.

Quantitative Results. The evaluation results on our test set are given in Tab. 1. We report results for three instantiations of our model: 'LocCNNPred' uses only the local CNN prediction term; 'LocCNNPred+InterConnComp' adds the inter-connected component term; while 'Full' denotes our full model. Following [32], we additionally apply a few post-processing steps including hole filling, removing tiny isolated regions and splitting the connected components of any prediction into separate instances. This is reported at the bottom of the table.

Notice that without post-processing the model 'LocCNNPred' which has only the local CNN prediction term performs much worse than our baselines, because it allows instances that do not coexist in any patch to have the same labeling. Post-processing removes some of these mistakes and makes the model already outperform the baselines [32] in a number of metrics. With the addition of the inter-connected component term, the model 'LocCNNPred+InterConnComp' encourages far apart instances to take different labels, which improves the performance. Finally, our full model which further adds the smoothness term is able to remove noisy regions scattered around the image, especially around instance boundaries where the CNN predictions are not confident. Our full model boosts instance-level precision by a huge margin compared to 'LocCNNPred+InterConnComp', outperforming the baselines significantly in a number of metrics.

Qualitative Results. We show examples of successes of our model ('Full') without post-processing in Fig. 4. We compare our full model to ground truth and the baseline 'ConnComp' which has the highest 'InsF1' score compared to the others. While the baseline tends to merge neighboring instances into one, our model is more successful in telling

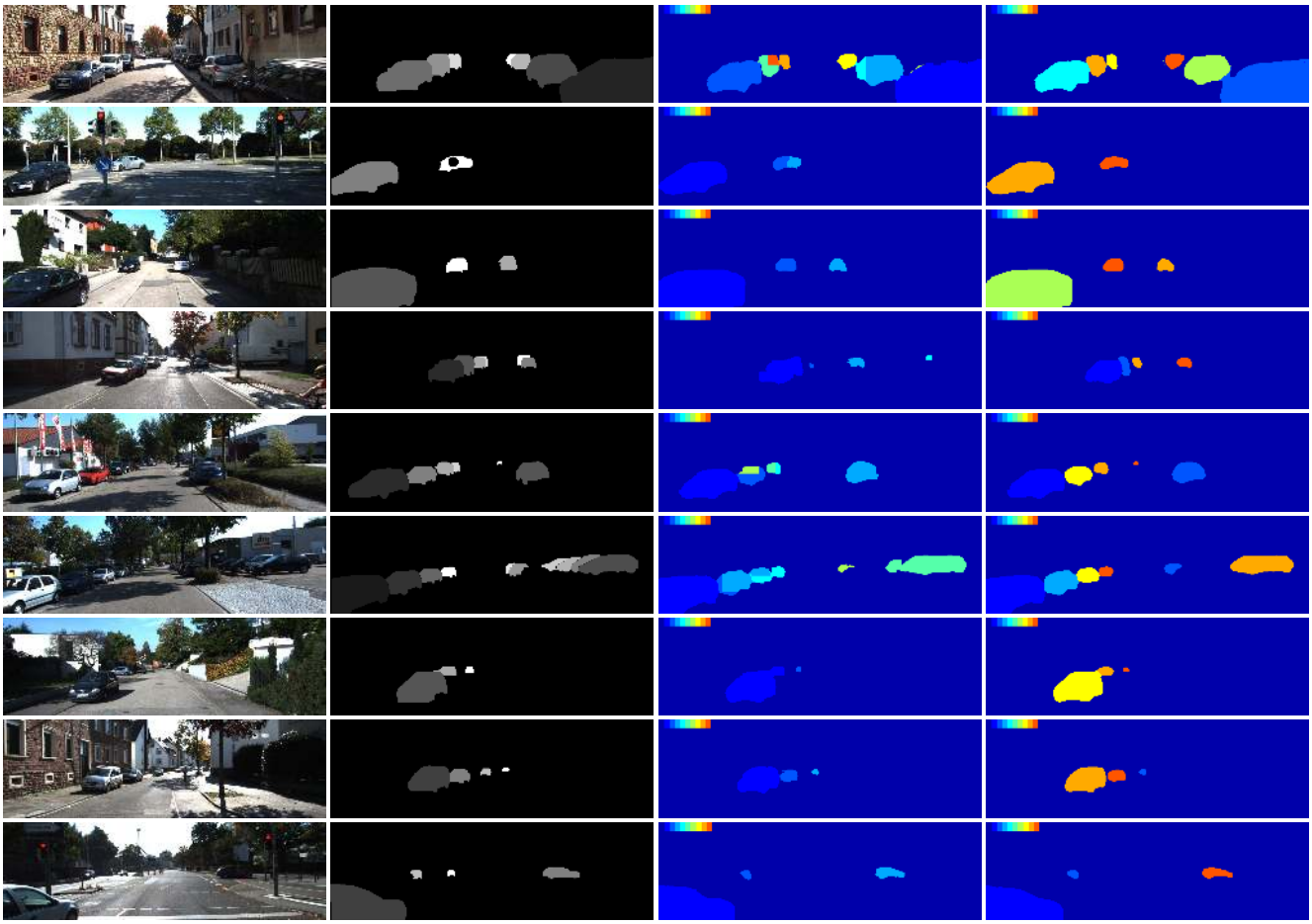


Image Ground Truth [32] Ours

Figure 4: Successes of our model (without post-processing) with comparison to [32].

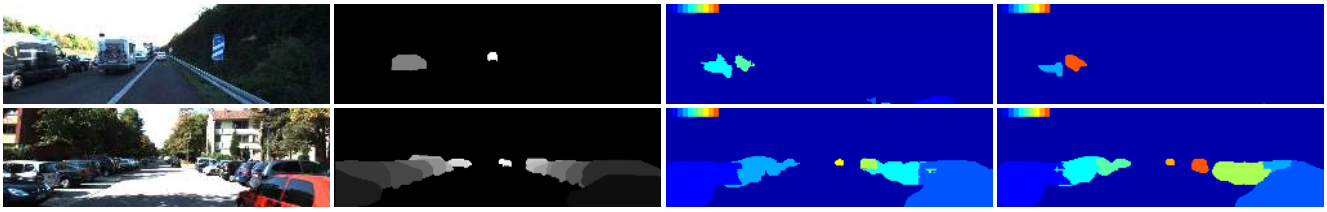


Image Ground Truth [32] Ours

Figure 5: Failures of our model, mostly due to large non-car vehicles, small objects or severe occlusion.

them apart. Patch boundary is clearly noticeable in many of the baseline results, while our model prevents this from happening. This suggests that our model exploits the local CNN predictions in a much better way than the baselines.

A few failure cases of our model are shown in Fig. 5, which are largely due to the CNN output. Vehicles like vans that are similar to cars (but not labeled as cars) tend to confuse the CNN, thus introducing false positives into our results. Heavily occluded cars also pose great challenges.

5. Conclusions

In this paper, we proposed a new approach for instance-level segmentation. Our approach builds upon the recently proposed work [32] which trains a CNN on local patches to obtain soft instance labelings. We propose a densely connected MRF that is amenable to the efficient inference algorithm by [15] to derive a globally consistent instance labeling of the full image. Our MRF exploits local CNN predictions, long-range connections between far apart instances, and contrast-sensitive smoothness. Our experiments show significant improvements over [32].

Acknowledgments: This work was partially supported by ONR-N00014-14-1-0232, Samsung and NSERC.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. *Computer Graphics Forum*, 29(2):753–762, 2010. 5

[2] M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014. 2

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

[4] 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. 1, 2

[5] L.-C. Chen, S. Fidler, A. Yuille, and R. Urtasun. Beat the MTurkers: Automatic Image Labeling from Weak 3D Supervision. In *CVPR*, 2014. 3, 6

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *ICLR*, 2015. 6

[7] S. Fidler, S. Dickinson, and R. Urtasun. 3D Object Detection and Viewpoint Estimation with a Deformable 3D Cuboid Model. In *NIPS*, 2012. 3

[8] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012. 1, 2, 6

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524*, 2013. 1, 2

[10] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015. 2

[11] S. Gupta, R. Girshick, P. Arbelaez, and J. Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014. 1, 2

[12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 1, 2

[13] X. He and S. Gould. An Exemplar-based CRF for Multi-instance Object Segmentation. In *CVPR*, 2014. 1

[14] P. Isola and C. Liu. Scene Collaging: Analysis and Synthesis of Natural Images with Semantic Layers. In *ICCV*, 2013. 2

[15] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, pages 109–117, 2011. 1, 2, 3, 5, 7, 8

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[17] M. P. Kumar, P. H. S. Torr, and A. Zisserman. OBJ CUT. In *CVPR*, 2005. 1, 2

[18] L. Ladický, P. Sturgess, K. Alahari, C. Russell, and P. H. S. Torr. What, Where and How Many? Combining Object Detectors and CRFs. In *ECCV*, 2010. 2

[19] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. In *Arxiv:arXiv:1509.02636*, 2015. 2

[20] J. Lim, H. Pirsiavash, and A. Torralba. Parsing IKEA Objects: Fine Pose Estimation. In *ICCV*, 2013. 2

[21] D. Lin, S. Fidler, C. Kong, and R. Urtasun. Visual Semantic Search: Retrieving Videos via Complex Textual Queries. In *CVPR*, 2014. 1

[22] H. Riemenschneider, S. Sternig, M. Donoser, P. M. Roth, and H. Bischof. Hough Regions for Joining Instance Localization and Segmentation. In *ECCV*, 2012. 2

[23] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. In *arXiv:1511.08250*, 2015. 2

[24] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 1, 2

[25] A. G. Schwing and R. Urtasun. Fully Connected Deep Structured Networks. http://arxiv.org/abs/1503.02351, 2015. 3, 6

[26] N. Silberman, D. Sontag, and R. Fergus. Instance Segmentation of Indoor Scenes using a Coverage Loss. In *ECCV*, 2014. 2, 7

[27] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *arXiv:1409.1556*, 2014. 1

[28] J. Tighe, M. Niethammer, and S. Lazebnik. Scene Parsing with Object Instances and Occlusion Ordering. In *CVPR*, 2014. 1, 2

[29] S. Wang, S. Fidler, and R. Urtasun. Holistic 3D Scene Understanding from a Single Geo-tagged Image. In *CVPR*, 2015. 1

[30] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *PAMI*, 2012. 1, 2

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

[32] Z. Zhang, A. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with cnns. In *ICCV*, 2015. 1, 2, 3, 6, 7, 8