Instance-Level Segmentation

Mengye Ren

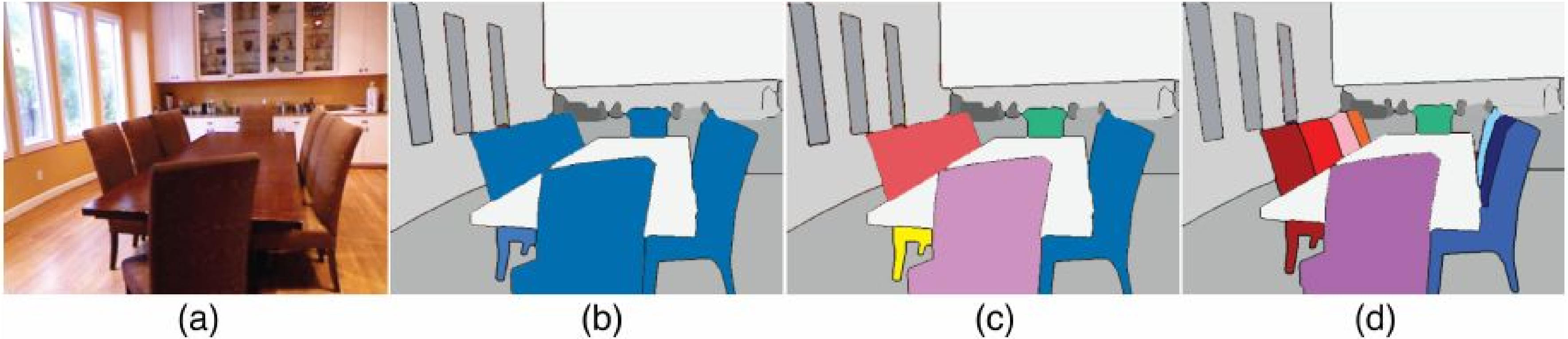
March 15, 2016

Agenda

- Introduction to instance-level segmentation
- N. Silberman, D. Sontag, R. Fergus. Instance Segmentation of Indoor Scenes using a Coverage Loss. ECCV 2014.
- Z. Zhang, S. Fidler, R. Urtasun. Instance-Level Segmentation with Deep Densely Connected MRFs. CVPR 2016.

What is instance-level segmentation

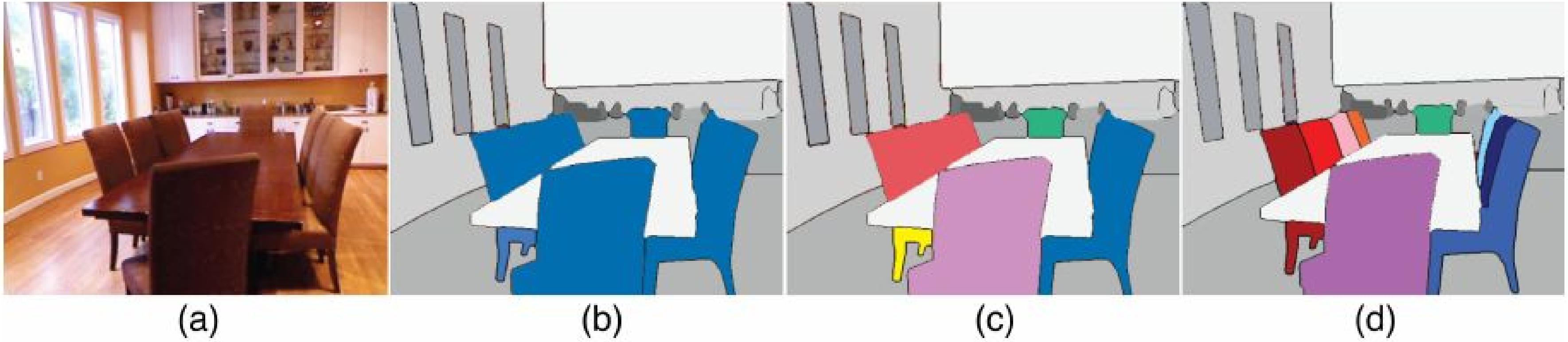
- Assign a label to each pixel of the image.
- Labels are class-aware and instance-aware. E.g. Chair_1, Chair_2, …, Table_1, etc.



(Image from Silberman et al. 2014)

Difference from semantic segmentation

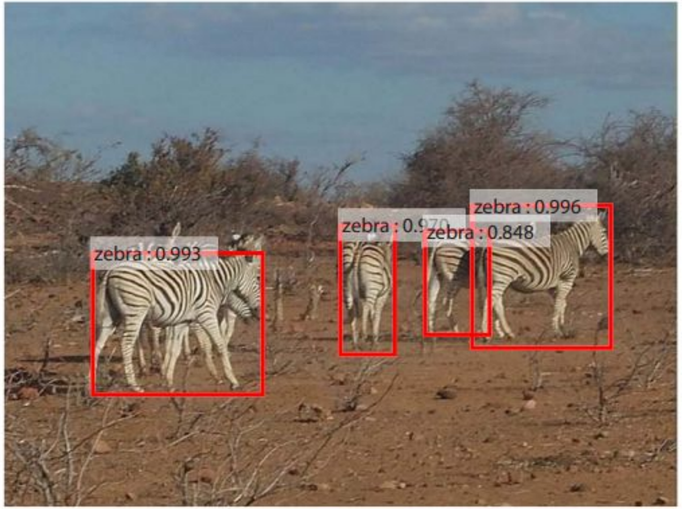
- One level increase in difficulty.
- More understanding on the instance individuals and reasoning about occlusion.
- Essential to tasks such as counting the number of objects.



(Image from Silberman et al. 2014)

Difference from 2D object detection and matting

- A detection box is a very coarse object boundary. NMS will suppress occluded objects or slanted objects.



(Image from Ren et al. 2015)

Instance Segmentation of Indoor Scenes using a Coverage Loss

Instance Segmentation of Indoor Scenes using a Coverage Loss

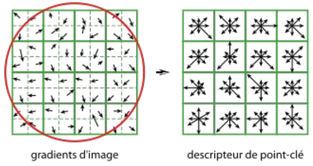
- Paper from Nathan Silberman, David Sontag, Rob Fergus, ECCV 2014.

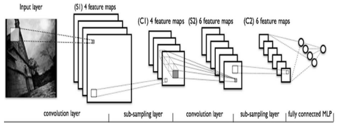
Key contribution:

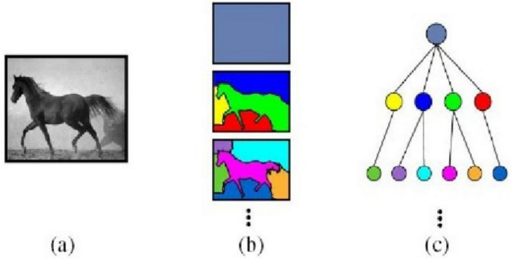
- Segmentation Tree-Cut algorithm
- High order
- A new dataset for indoor scenes: NYU v2 dataset.

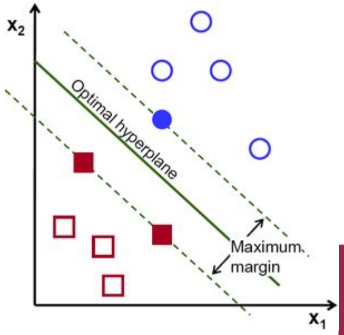
Big Picture of the Pipeline

Feature Extractor (SIFT or CNN) → Boundary Detector (gPb and UCM) → Segmentation Tree → Structured SVM



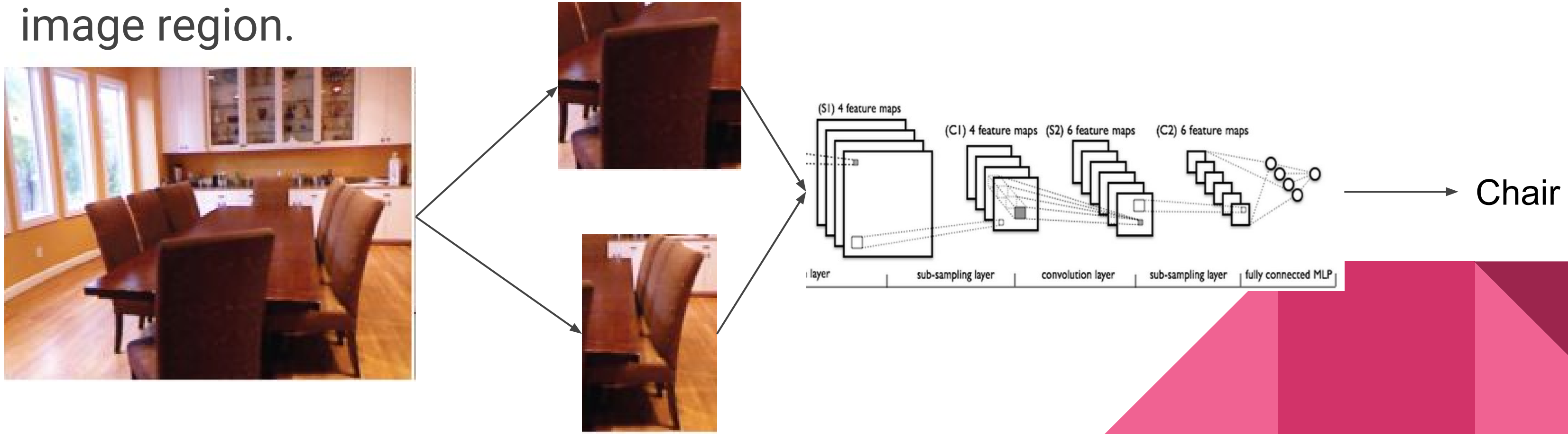






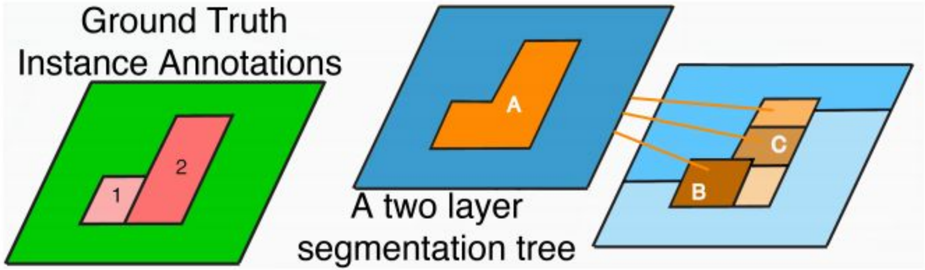
CNN feature extractor

- For each instance in the dataset, compute a tight bounding box plus 10% margin, and feed it into the CNN.
- Train the CNN to predict the semantic labels of each instance.
- During inference, use the fully connected hidden layer as the features of an image region.



Segmentation Tree

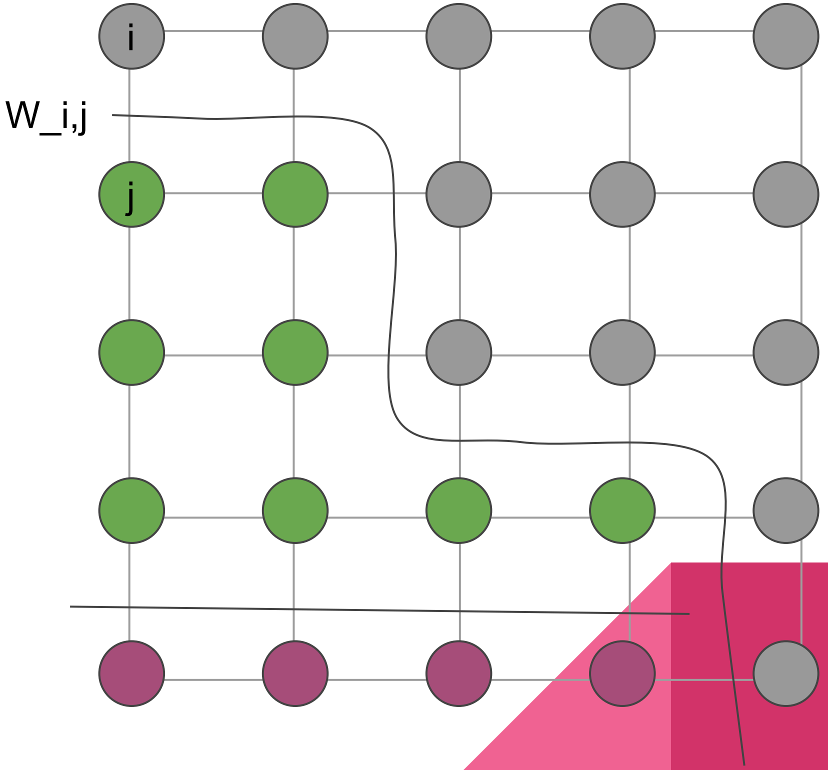
- Motivation: to limit the search space of instance segmentation. Instead of arbitrarily assigning each pixel with a label, it needs to obey the tree structure.
- Completeness: Every pixel I_i is contained in at least one region of S.
- Tree Structure: Each region s_i has at most one parent: $P(si) \in \{\varnothing, s_j\}, j \neq i$
- Strict Nesting: If $P(s_i) = s_j$, then the pixels in si form a strict subset of s_j



(Image from Silberman et al. 2014)

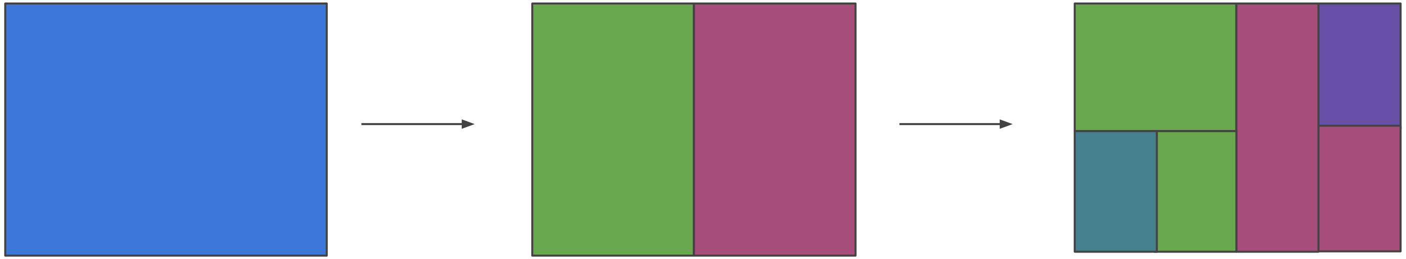
Building Segmentation Tree

- Starts with a 2-D planar graph of H x W.
- Segmentation is equivalent to performing graph cuts.
- Edge weights are computed from boundary probability algorithms (gPb and UCM).
- Edges below thresholds are removed at each iteration.



Building Segmentation Tree

- Then for the next iteration, we can dig into each connected component of the resulting graph and perform finer cuts.
- In the end, we get a coarse-to-fine hierarchy of regions.



Biased Segmentation Tree

- Is tree a good structure in general to solve instance segmentation problems?
- Is it too limiting?
- To investivate this, the authors designed the so-called “**biased segmentation tree**”
- Cut the tree until all groundtruth instance regions can be perfectly segmented by all the regions.
- The performance generated from the **biased segmentation tree** is an **upper bound** of the proposed model.

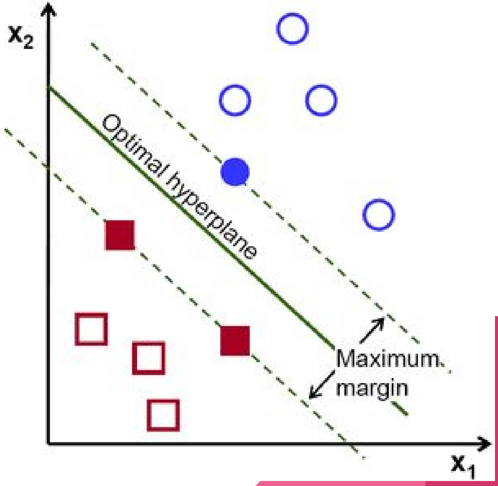
Structured SVM Learning and Inference

Output: y = (A, C) Regions {A: A_i ∈{0, 1}, i = 1...R}, Classes {C: C_i ∈ {1...K}, i=1...R}

$$\mathbf{w_{reg} \cdot \varphi_{reg}(x, y) + \sum_k w_{sem:k} \cdot \varphi_{sem:k}(x, y) + w_{pair} \cdot \varphi_{pair}(x, y) + \varphi_{tree}(y)}$$

Four terms:

- Region (class agnostic)
- Semantic
- Pairwise
- Tree constraint



Structured SVM Learning and Inference

- **Region term:** Sum up feature descriptors for all proposed regions.
- Intuitively, this encodes how good a segmentation is without considering class.

- **Semantic term:** Sum up feature descriptors for all proposed regions that belongs to a certain class.
- This encodes how each region matches with their class label.

Structured SVM Learning and Inference

- **Pair-wise term:** Sum up features that describes neighbouring regions **A_i** and **A_j**.
- This encode how adjacent regions are compatible with each other.

- **Tree constraint term:** Impose very high loss term if the resulting regions do not form tree in the tree proposal.
- For every path from root to leaves, there is only one region gets selected.

Structured SVM Learning and Inference

- Learning: Use structured SVM formulation.
- **argmin ½ $w \cdot w + \lambda \sum \xi_i$ s.t. $w \cdot [\varphi(x_i, y_i) - \varphi(x_i, y)] \geq \Delta(y, y_i) - \xi_i \ \forall i, y$**
- **x_i** and **y_i** are training images and labels
- **$\Delta(y, y_i)$** is the loss function between proposed segmentation and GT.
- **ξ_i** is the slack variable for each training example.
- This is saying, the true label y_i should be the best possible output, and should have a margin of **$\Delta(y, y_i)$** compared to other possible output **y**, up to maybe a slack variable **ξ_i**.

Structured SVM Learning and Inference

- Inference: can be formulated as an integer linear program (ILP).
- **R := number of regions. E := number of edges.**
- **A ∈ [0,1]^{R x 2}, C ∈ [0,1]^{R x K}, P ∈ [0,1]^{E}**
- $\mathbf{a_{i,0}=0}$ indicates a region i is inactive. $\mathbf{a_{i,1}=1}$ indices a region i is active.
- $\mathbf{c_{i,k}=1}$ indicates the semantic class of a region.
- $\mathbf{p_{i,j}=1}$ indicates the neighbouring regions i and j are both active.

Structured SVM Learning and Inference

- Inference:
- $\mathbf{argmax_{a,c,p} \sum_i \theta_r \cdot a_{i,1} + \sum_i \sum_k \theta^s_{ik} \cdot c_{ik} + \sum_i \theta^p_{ij} \cdot p_{ij}}$
- s.t.
- $a_{i,1} + a_{i,0} = 1$ (A region is either active or inactive)
- $\sum_k c_{i,k} = 1$ (A region has one semantic label)
- $\sum_{i \in r} a_{i,1} = 1$ (Tree constraint)
- $p_{i,j} \leq a_{i,1}$ $p_{i,j} \leq a_{j,1}$ $a_{i,1} + a_{j,1} - p_{i,j} \leq 1$ $\forall i,j$ (Pairwise constraint)

Structured SVM Learning and Inference

- Up to now is only on region-semantic level. It cannot merge regions to a instance yet. To do this, they proposed Loss Augmentation for ILP.
- **G:= number of groundtruth instances.**
- **A ∈ [0,1]^{R x 2}, C ∈ [0,1]^{R x K}, P ∈ [0,1]^{E}, O ∈ [0,1]^{G x R}**
- **O is a mapping from active region to groundtruth instance ID.**

More constraints...

- $o_{g,i} \leq a_{i,1}$ $\forall g, i$ (Active regions only)
- $\sum_i o_{g,i} \leq 1$ $\forall g$ (1 region can only map to 1 GT at most)
- $o_{g,i} + a_{j,1} \leq 1$ $\forall g \in G, i,j \in R$ s.t. $IoU(s_g, s_j) > IoU(s_g, s_i)$ (Maximum overlap)

Structured SVM Learning and Inference

- **argmax**$_{a,c,p} \sum_i \theta_r \cdot a_{i,1} + \sum_i \sum_k \theta^s_{ik} \cdot c_{ik} + \sum_i \theta^p_{ij} \cdot p_{ij} - \sum_g \sum_i \theta^o_{gi} \cdot o_{gi}$
- $\theta^o_{gi} = IoU(r^G_g, r^S_s) - IoU(r^G_g, r^S_i)$
- Minimize the difference between the groundtruth instance region and proposed instance region.
- r^S_s is the **surrogate** labelling => maximum overlap possible with the groundtruth instance, given the tree structure.

Structured SVM Learning and Inference

- There is still another problem. How to get the groundtruth that corresponds to the pre-defined segmentation tree regions?
- Solving an ILP problem can give us the surrogate labelling:
- **argmin**$_{a,o}$ $\sum_g \sum_i \theta^o_{gi}\, o_{gi}$
- subj. to.
- $a_{i,0} + a_{i,1} = 1 \;\forall\, i$ (Either active or inactive)
- $\sum_{i \in r} a_{i,1} = 1$ (Tree constraint)
- $o_{g,i} \le a_{i,1} \quad \forall\, g, i$ (Active regions only)
- $\sum_i o_{g,i} \le 1 \quad \forall\, g$ (1 region can only map to 1 GT at most)
- $o_{g,i} + a_{j,1} \le 1 \;\; \forall\, g \in G,\, i,j \in R$ s.t. $IoU(s_g, s_j) > IoU(s_g, s_i)$ (Maximum overlap)

Weighted Coverage Loss

- We haven't introduced the actual form of **Δ(y, y_i)**
- We could use **Hamming Loss** between the class vector **C** and region vector **A** since both are binary vector.

- They proposed **Weighted Coverage Loss**
- For each groundtruth instance, pick the maximum overlap output, and record the IoU between the GT and the best output
- Sum up the IoU, weighted by the area of the groundtruth instance.

Loss Surrogate Labels

- When using surrogate labels, they modified the loss function
- z := surrogate label, y := groundtruth label, y' := model prediction.
- **$\Delta w_2(z, y') = \Delta w_1(y, y') - \Delta w_1(y, z)$**
- **$\Delta w_1(y, z)$** can be pre-computed.
- Compensate for the inaccuracy of surrogate labels.

Experimental results

(Image from Silberman et al. 2014)

Experimental results

- Effect of depth information (upper bound): **70.6** (RGB-D) vs. **50.7** (RGB)
- Effect of CNN features: **62.5** (CNN) vs. **61.8** (SIFT)
- Effect of pairwise terms: **62.5** (with pairwise) vs. **62.4** (without pairwise)
- Effect of biased segmentation tree: **87.4** (biased) vs. **62.5** (standard)
- Effect of weighted coverage loss: **62.5** (Wt coverage) vs. **61.4** (Hamming)

Limitations

- Tree structure assumption. Cannot merge two non-neighbouring regions together (happens in case of occlusion).
- Coverage loss function does not penalize false positives.
- Integer programs may be slow (NP-hard inference).
- Relies on depth information (poor performance without depth).

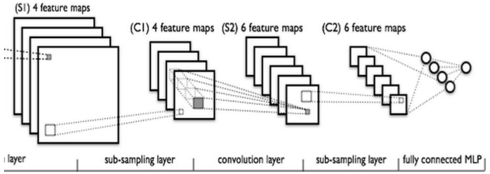
Instance-Level Segmentation with Deep Densely Connected MRFs

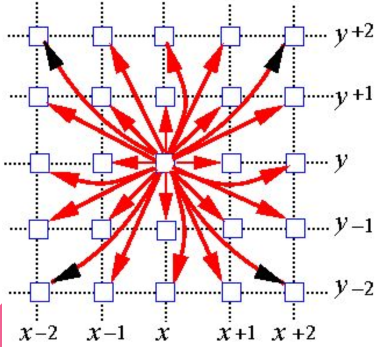
Instance-Level Segmentation with Deep Densely Connected MRFs

- Paper from Ziyu Zhang, Sanja Fidler, and Raquel Urtasun. CVPR 2016 (To appear).
- A new architecture that combines patch-based CNN prediction and global MRF reasoning.

Big Picture

Patches → CNN → MRF



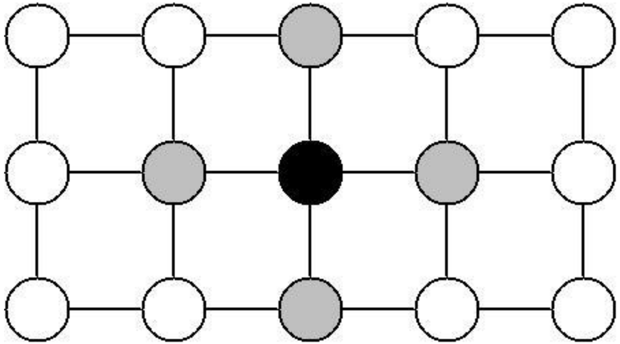


Patch-based CNN

- KITTI dataset, 375 x 1242
- Extract patches of different sizes: 270 x 432, 180 x 288, and 120 x 192
- Run the extracted patches to obtain local instance predictions
- There are less number of instances in the patch, so easier for CNN to assign instance labels.
- The instance ID is not guaranteed to be consistent across different patches.

(Image from Zhang et al. 2015)

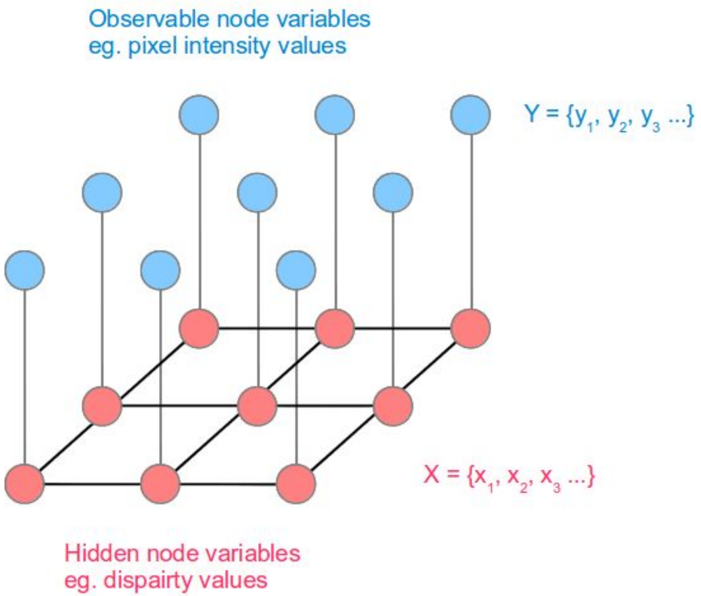
MRF



- Undirected graphical model
- Each vertex represents a random variable
- Edge represents conditional dependence between variables
- $P(x \mid \theta) \propto \exp(-E(x \mid \theta)) = \exp(-\sum_{c} E(x_c \mid \theta))$
- We can factor the graphical model with maximal clique (Hammersley-Clifford Theorem)
- C is the set of all maximal cliques in the graph.

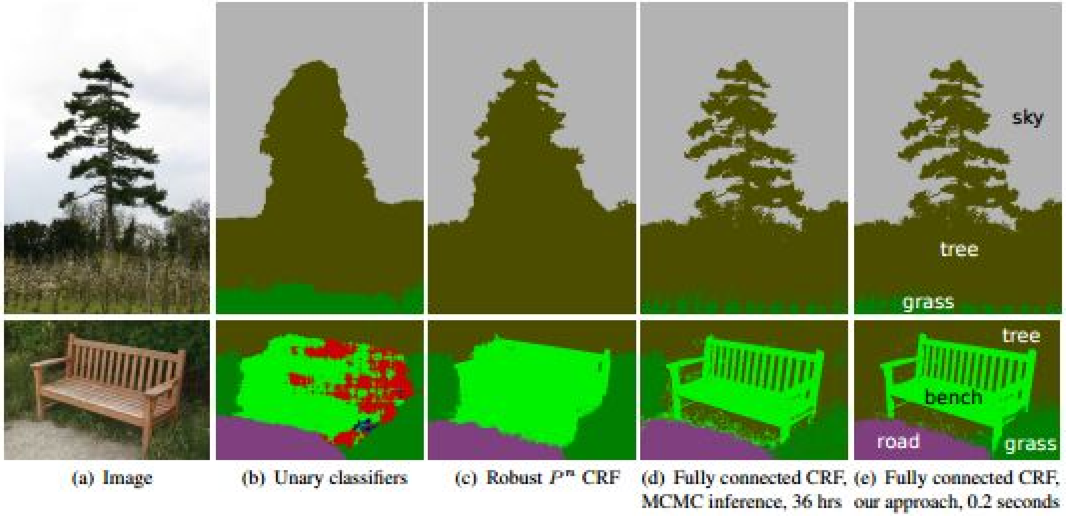
Pairwise MRF

- $P(x \mid \theta) \propto \exp(-E(x \mid \theta))$
- $= \exp\left(- \sum_{\mathbf{c}} E(x_c \mid \theta)\right)$
- $= \exp\left(- \sum_{\mathbf{i}} E(x_i \mid \theta) - \sum_{\mathbf{ij}} E(x_i, x_j \mid \theta)\right)$
- Unary energy: the probability of individual node.
- Pairwise energy: smoothness assumption.



Fully connected MRF

- Pairwise message passing is very myopic.
- Especially very complicated segmentations e.g. chair, tree.
- It would be nice to have each node to be neighbours with all other nodes. => Longer range message passing influence.



(Image from Krahenbuhl & Koltunan 2011)

Fully connected MRF

- Learning and inference could be computationally intractable for fully connected models..
- But this requires that the energy function to be Gaussian.
- But if we define a dot product $|| \cdot ||^2$ for $\phi(x_i)$ (i.e. a kernel),
- And if $E(x) \propto \exp(-|| \phi(x_i) - \phi(x_j) ||^2 / 2\theta^2)$, then we can use Guassian blurring as a mean field approximation to the original graphical model.
- Details can be found in P. Krahenbuhl, V. Koltun. Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials. NIPS 2011.

MRF for instance segmentation

- Here each vertex represents the instance labelling of each pixels.
- In the paper, the authors designed three terms in the energy function.
- **$E(y) = E_{smo}(y) + E_{cnn}(y) + E_{icc}(y)$**
- **$y^* = \text{argmin}_y E(y)$**
- E_{smo}: Smoothness. Close pixels should have similar instance labelling
- E_{cnn}: Local CNN prediction. Local instance boundary should be similar with CNN prediction.
- E_{icc}: Inter-connected component. Same instance should not appear in disconnected component.

MRF for instance segmentation

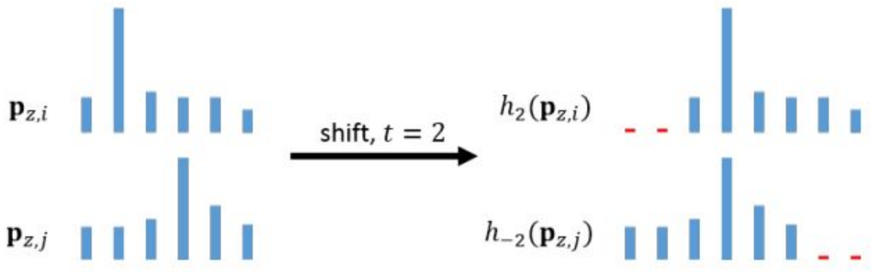
- **E_{smo} Smoothness term**
- 2 Gaussian kernels, output distance and spatial distance
- $k_{smo}(\phi(x_i), \phi(x_j)) = \exp(-\|p_i - p_j\| / 2\theta^2_1 - \|d_i - d_j\| / 2\theta^2_2)$
- p_i: CNN prediction of xi
- d_i: Spatial position of xi
- Penalize pixels with similar positions and CNN predictions to have different labels.
- $E_{smo} = w_{smo}\mu_{smo}(y_i, y_j)\, k_{smo}(\phi(x_i), \phi(x_j))$
- $\mu_{smo}(y_i, y_j) = 1[y_i \neq y_j]$.

MRF for instance segmentation

- **E_{cnn}: Local CNN prediction term.**
- **$E_{cnn}(y) = \sum_z \sum_{i,j,\, i<j} \varphi^z_{cnn}(y_i, y_j)$**
- Sum up all local patch predictions z
- The intuition is that, if the local CNN says that y_i and y_j are from different instances, then their global configurations should respect that.
- Locally fully connected energy function on patch level.
- Encourage asymmetry to kick off the inference, apply penalty when $i < j$ only.
- But this asymmetry does not work as a Gaussian kernel.
- So instead, the authors proposed a series of Gaussian kernels to approximate this potential.

MRF for instance segmentation

- $E_{cnn}(y) = \Sigma_z \Sigma_{i,j,\ i<j} \Sigma_t\ \varphi^t{}_{cnn}(y_i, y_j)$
- $\boldsymbol{\varphi^t{}_{cnn}(y_i, y_j) = w_{cnn}\mu_{cnn}(y_i, y_j)\ k_{cnn}(ht(p_i), h\text{-}t(p_j))}$
- $\mu_{cnn}(y_i, y_j)$ = -1 (i.e. encouraged configuration) if
  - $y_i < y_j$, $t > 0$
  - $y_i > y_j$, $t < 0$
  - $y_i = y_j$, $t = 0$ (No shift, encourage same label)

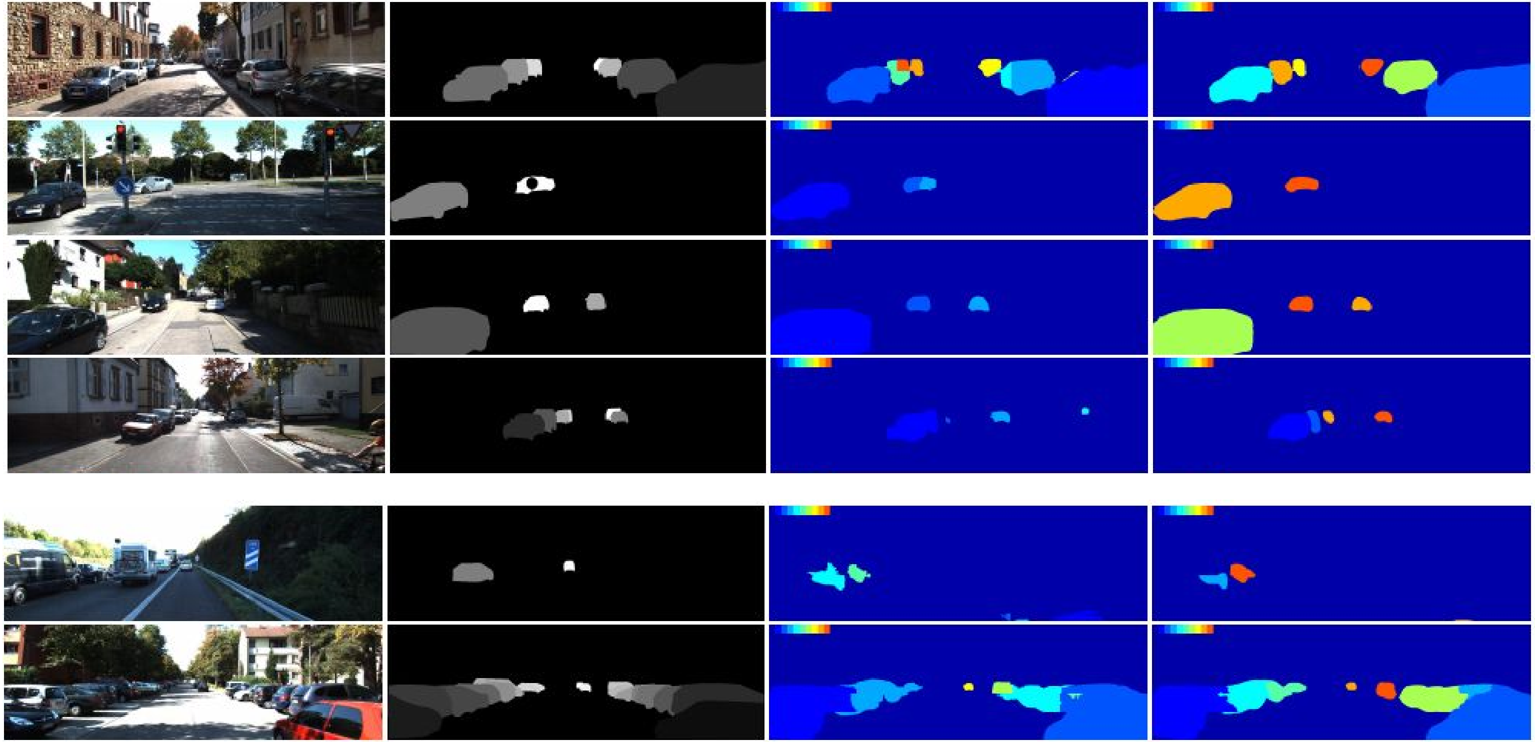


(Image from Zhang et al. 2015)

MRF for instance segmentation

- $\mathbf{E_{icc}(y) = \sum_{m,\, n\; m<n} \sum_{i \in m,\, j \in n} w_{icc} \mu_{icc}(y_i, y_j)}$
- m and n are inter connected components
- $\mu_{icc}(y_i, y_j) = 1$ if $y_i = y_j$
- i.e. discourage same labels across disconnected components.

Experimental results



(Image from Zhang et al. 2015)

Experimental results

	Class Eval	Instance Evaluation								
	IoU	MWCov	MUCov	AvgPr	AvgRe	AvgFP	AvgFN	InsPr	InsRe	InsF1
ConnComp [27]	77.1	66.7	49.1	82.0	60.3	0.465	0.903	49.1	43.0	45.8
Unary [27]	**77.6**	65.0	48.4	81.7	62.1	0.389	0.688	46.6	42.0	44.2
Unary+LongRange [27]	**77.6**	66.1	49.2	**82.6**	62.1	**0.354**	0.688	48.2	43.1	45.5
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**
	With Post-processing									
ConnComp [27]	77.2	66.8	49.2	81.8	60.3	0.465	0.903	49.8	43.0	46.1
Unary [27]	**77.4**	66.7	49.8	81.6	61.2	0.562	0.840	44.1	44.7	44.4
Unary+LongRange [27]	**77.4**	67.0	49.8	82.0	61.3	0.479	0.840	48.9	43.8	46.2
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**

(Image from Zhang et al. 2015)

Limitations

- Works on single object types in the paper.
- Inter-connectedness assumption may fail. In KITTI, there is occlusions such as poles that “cuts” a car into two components.
- Empirically speaking, heavy occlusions and very small cars in distance is not working ideally.

Thanks!