Analysis: TextonBoost and Semantic Texton Forests

Daniel Munoz

16-721

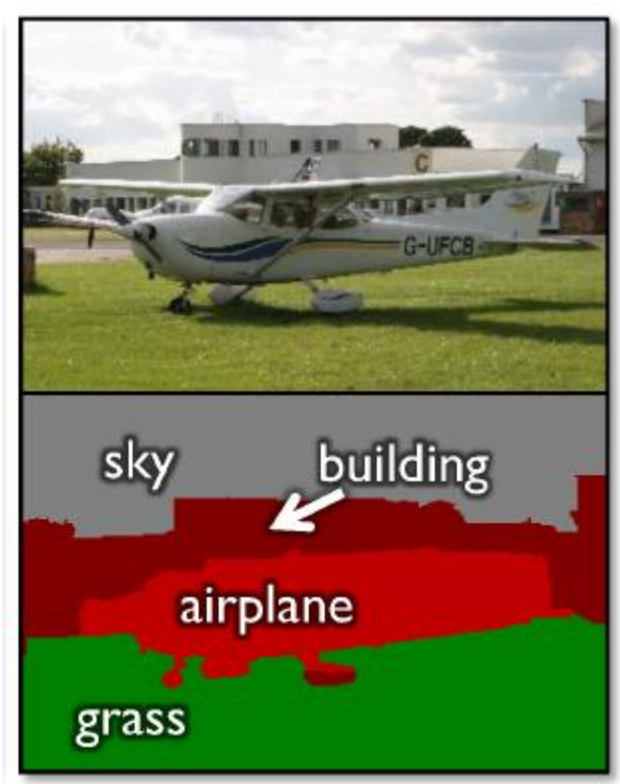
Februrary 9, 2009

Papers

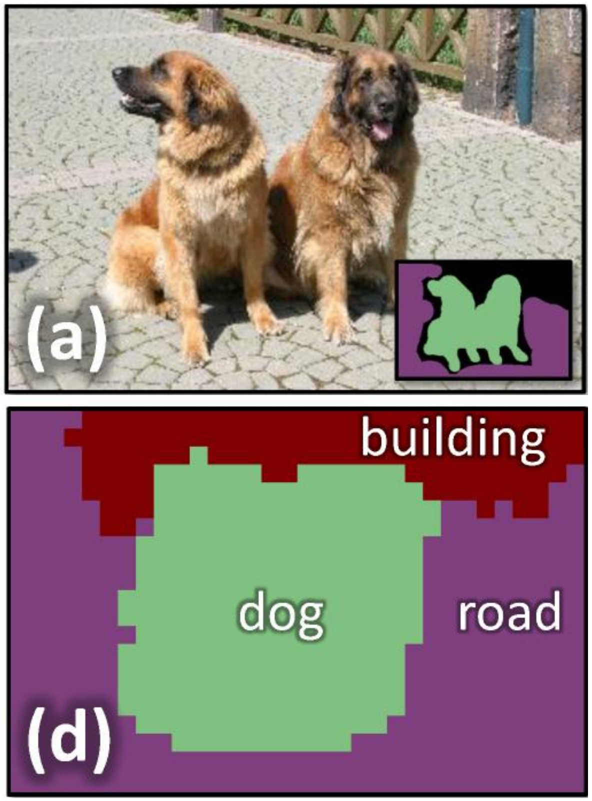
- [shotton-eccv-06] J. Shotton, J. Winn, C. Rother, A. Criminisi, *TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation*, ECCV 2006
- [shotton-cvpr-08] J. Shotton, M. Johnson, R. Cipolla, *Semantic Texton Forests for Image Categorization and Segmentation*, CVPR 2008

Problem

- Ultimate goal for both these papers:



[shotton-eccv-06]



[shotton-cvpr-08]

- Simultaneous segmentation and recognition of objects in images

[shotton-eccv-06]

- [shotton-eccv-06] J. Shotton, J. Winn, C. Rother, A. Criminisi, *TextonBoost: Joint Appearance, Shape and Context Modeling for Multi-Class Object Recognition and Segmentation*, ECCV 2006

Data and Classes

- Goal: assign every pixel to a label

Object classes	Building	Grass	Tree	Cow	Sheep	Sky	Aeroplane	Water	Face	Car
Bike	Flower	Sign	Bird	Book	Chair	Road	Cat	Dog	Body	Boat

  - MSRC-21 database ("void" label ignored for training and testing)

Claimed contributions

- Discriminative model capable of fusing
  - **shape**
  - **appearance**
  - **context**

  information to efficiently recognize and accurately **segment** the object classes present in an image

- New texton-based features which are capable of modeling object shape, appearance and context.

- Efficient training of model on large dataset with many labels
  - Piece-wise CRF training with boosting

Outline

- High-level description of approach:
  - Learn classifier based on relative texture locations for each class
  - Refine classification with Conditional Random Field (CRF)
  - Improve classification with additional pixel information

- Review of CRFs….

Conditional Random Fields

- Main idea:
  - Local classifiers (SVM, LR, etc.) classify each pixel individually

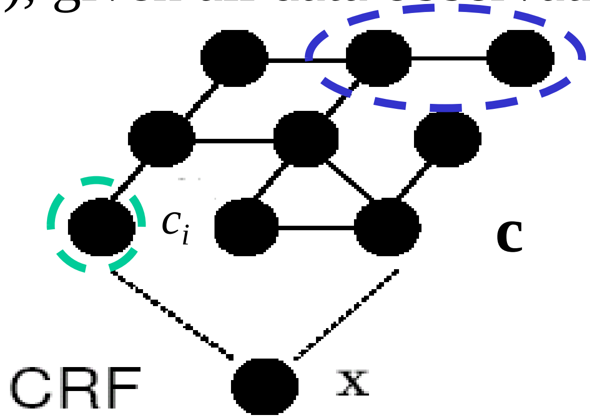
  - Markov Random Field (MRF) framework classifies all pixels jointly
    - ✓ Each pixel is a node in a undirected graph
    - ✓ Interactions/dependencies indicated by linked nodes

- Why?

Images from Szummer DAR'05

Conditional Random Fields

- Discriminative MRF for jointly estimating the label assignments to random variables (**c**), given all data observations (**x**)



- Models the joint distribution

$$P(\mathbf{c}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{i \in V} \Psi_i^{(1)}(c_i, \mathbf{x}; \boldsymbol{\theta}) \prod_{(i,j) \in E} \Psi_{i,j}^{(2)}(c_i, c_j, \mathbf{x}; \boldsymbol{\theta}) \quad (1)$$

  - $\Psi^{(1)}$ models the local score in the label assignment
  - $\Psi^{(2)}$ models the score for the *pairwise* assignment
  - Z costs exponentially to explicitly compute (|L|^|V|)

Inference

- Inference = finding the best joint labeling
  - NP-complete problem in general
- Two options: 1) argmax labeling 2) labeling + confidences
- Argmax labeling with usually Graph-Cut inference
  - Edge potentials need to satisfy submodularity constraints
    - ✓ Pott's model satisfies this (more on this later)
    - ✓ High-order potentials possible
  - Recent research with non-submodular potentials
    - ✓ Quadratic Pseudo-Boolean Optimization (QPBO)

- Labeling + confidences
  - Estimate the marginal probabilities
  - Usually done with Belief Propagation (or one of its variants)
  - Approximate solution if loops present
    - ✓ Computation exponential in size of smallest clique (tree-width)
    - ✓ Hence, most models are *pairwise* (maximal clique size of 2)

Back to TextonBoost…

Learning a local classifier

- The TextonBoost CRF model

$$\log P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i,\mathbf{x};\boldsymbol{\theta}_\psi)}^{\text{shape-texture}} \cdot$$

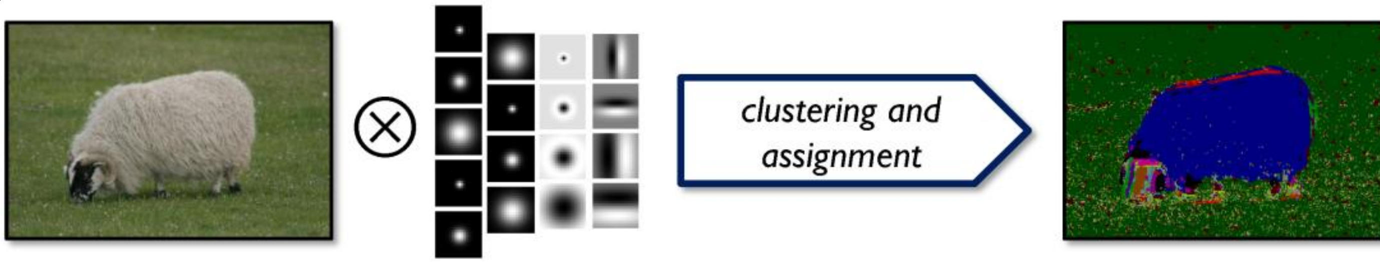
- **Shape-texture Potential**
  - Function based on new features called *shape filters*

- Trained using boosting to produce multi-class logistic classifier
  - See [torralba-pami-07], Yuandong's upcoming analysis (Week 11)

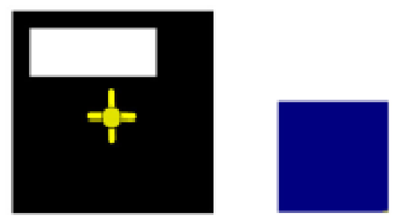
- Most **important** potential in the model

Capturing context

- **Shape-texture Potential** $\psi_i(c_i, \mathbf{x}; \boldsymbol{\theta}_\psi) = \log \widetilde{P}_i(c_i|\mathbf{x})$
  - Main idea: capture the context of relative texton locations for certain classes
- Step 1: Texton Map generation (17 filters, K=400)

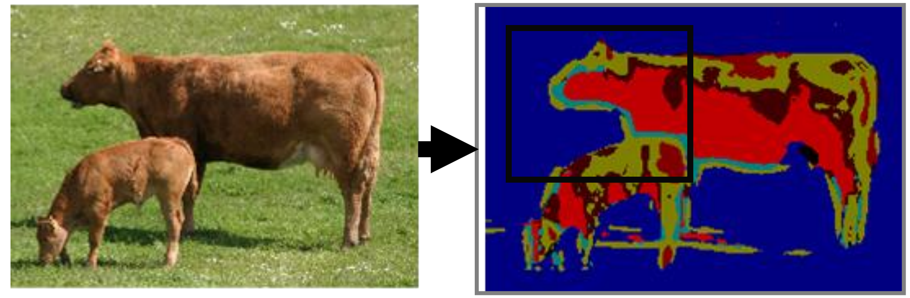


- Step 2: Shape Filter
  - For each texton *t*
    - ✓ Inputs
      - Texton Map
      - (Rectangle mask *r*, texton query *t*)
      - Pixel location *i*
    - ✓ Output
      - Area in rectangle mask that match *t*
  - End result is a texton histogram of area responses
- **How does this capture shape?**

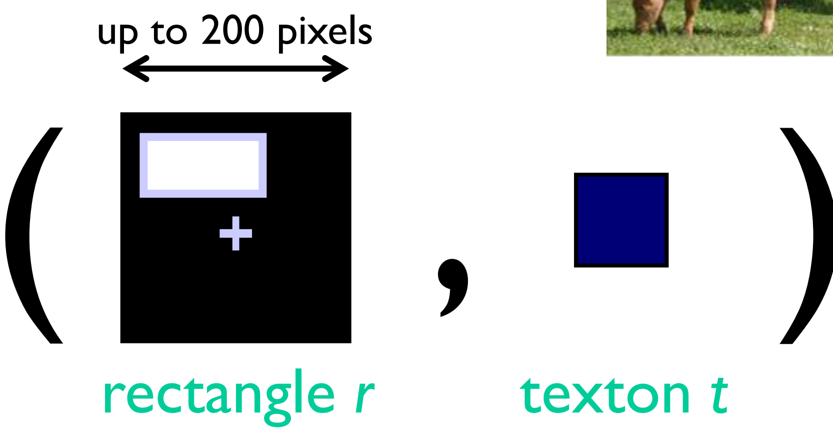


Slides from Shotton's ECCV talk

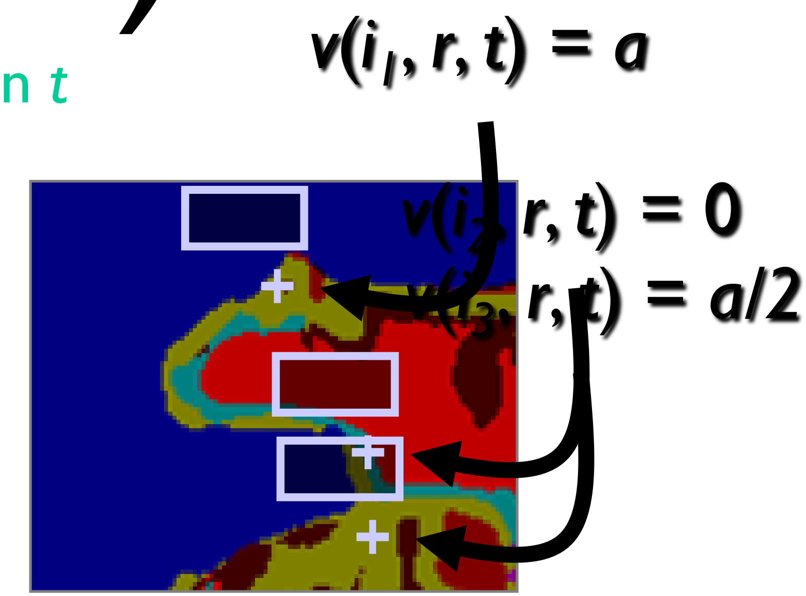
Shape Filters



- Pair:



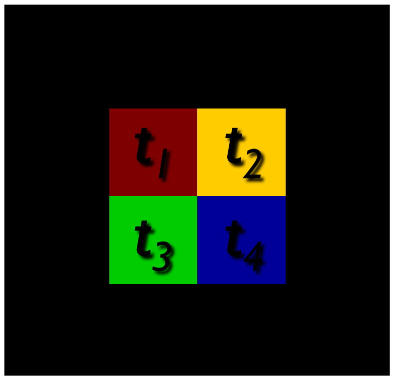
- Feature responses $v(i, r, t)$
- Large bounding boxes enable *long range interactions*
- Integral images



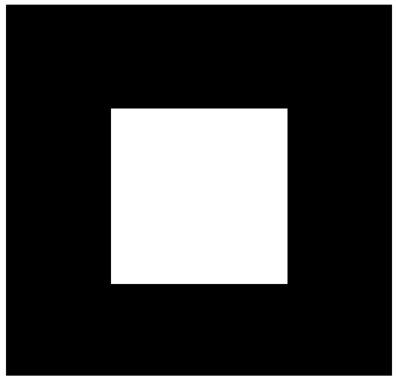
appearance context

Slides from Shotton’s ECCV talk

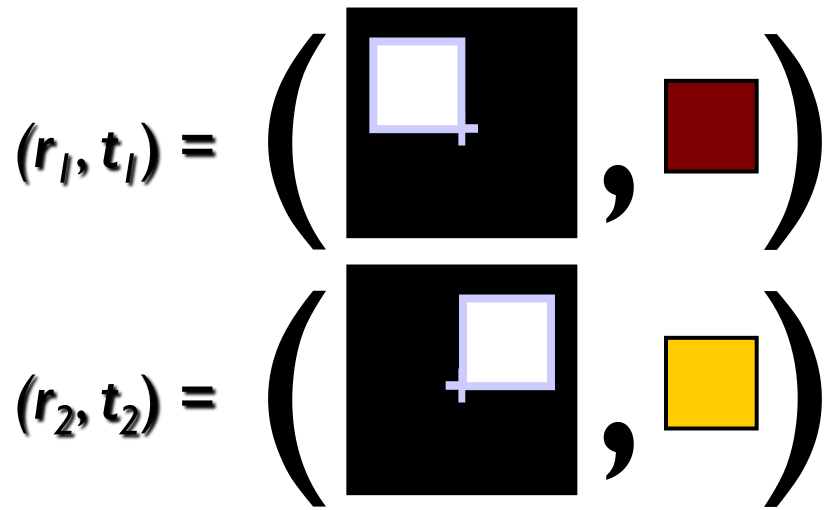
Shape as Texton Layout

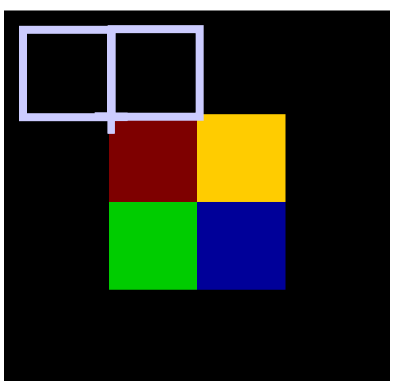


texton map

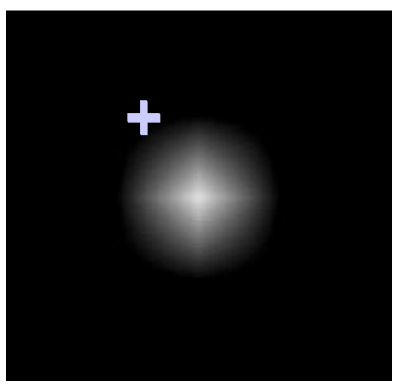


ground truth





texton map

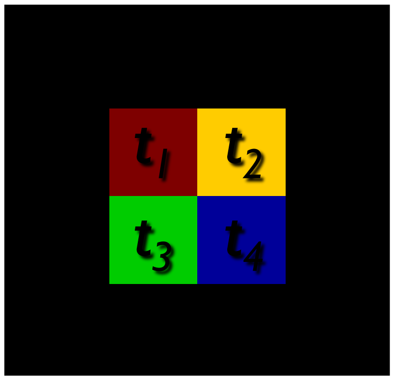


feature response image

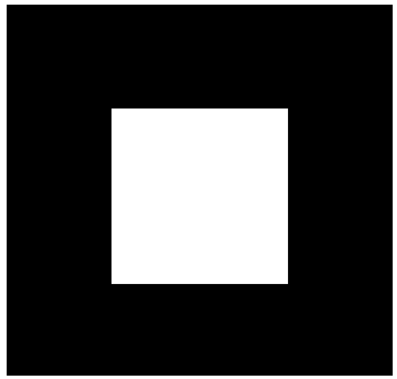
$v(i, r_2, t_2)$

Slides from Shotton's ECCV talk

Shape as Texton Layout



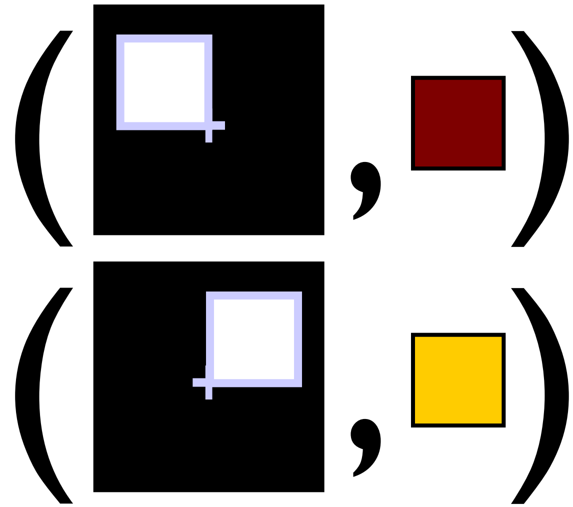
texton map

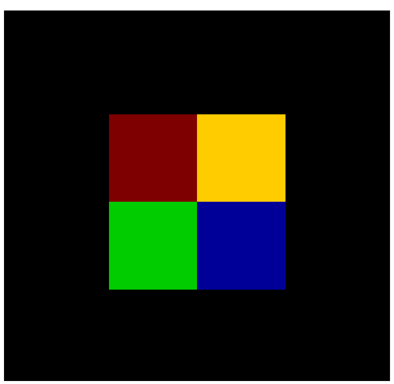


ground truth

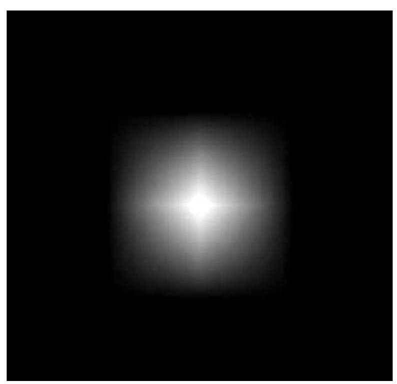
$(r_1, t_1) =$

$(r_2, t_2) =$



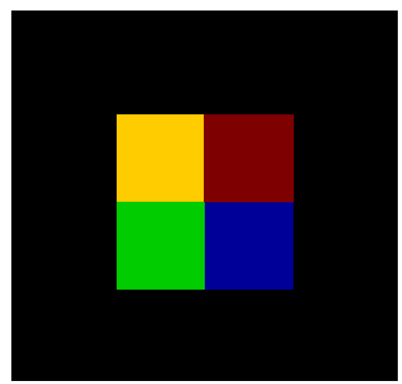


texton map

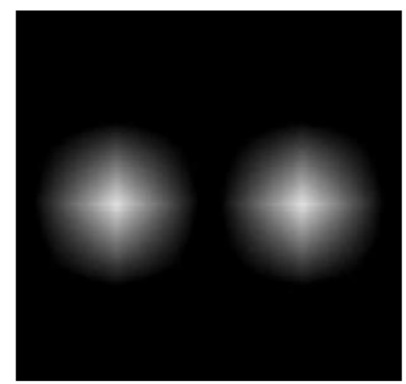


summed response images

$v(i, r_1, t_1) + v(i, r_2, t_2)$



texton map



summed response images

$v(i, r_1, t_1) + v(i, r_2, t_2)$

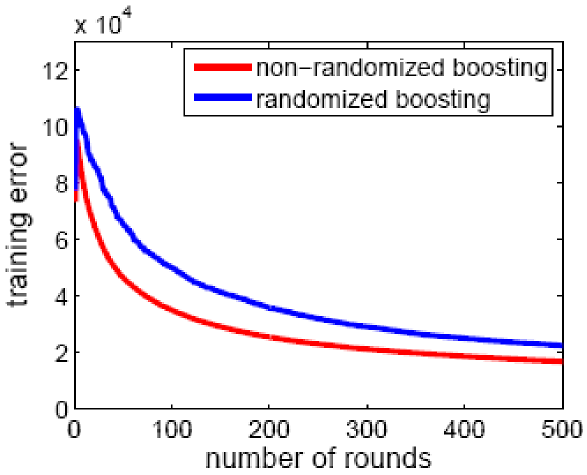
Learning context

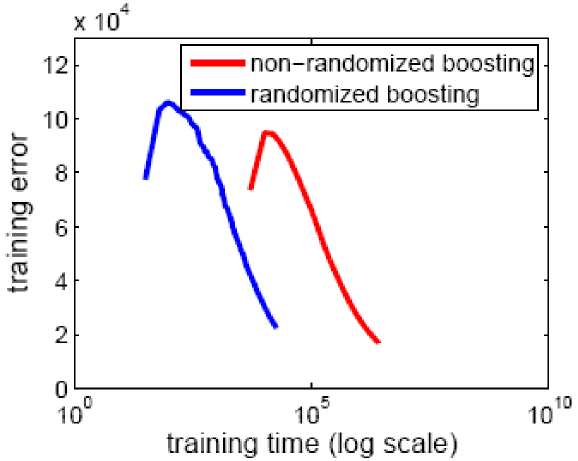
- What do we do with these histograms of shape filters?
  - Boosting over the shape-filter counts of texton *t* in rectangle *r*

$$\widetilde{P}_i(c_i|\mathbf{x}) = \frac{\exp(H(c_i))}{\sum_{c_i'} \exp(H(c_i'))}$$

$$h(c_i) = \begin{cases} a\delta(v(i,r,t) > \theta) + b & \text{if } c_i \in N \\ k_{c_i} & \text{otherwise} \end{cases}$$

- Ideal algorithm:
  - For each pixel in the Texton Map
    - ✓ For each possible rectangle mask orientation
      - – For each texton
        - » Augment shape-filter to training set
- Actual algorithm
  - For each pixel in the **sub-sampled** Texton Map
    - ✓ For **10 random** rectangle masks
      - – For each texton (K=400)
        - » Augment shape-filter to training set with **0.3% probability**
- 42 hours for 5,000 rounds on 276 images





Initial result

- Cumulative Results

shape-texture

Shape-texture potentials only: 69.6%

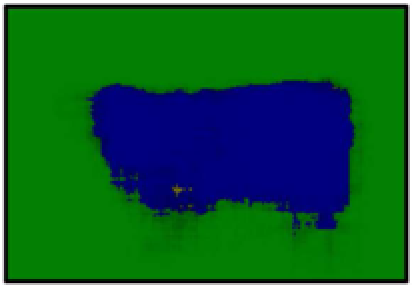
pixel-wise segmentation accuracies

Refining classification

- Let's smooth the borders

$$\log P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i,\mathbf{x};\boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \sum_{(i,j)\in\mathcal{E}} \overbrace{\phi(c_i,c_j,\mathbf{g}_{ij}(\mathbf{x});\boldsymbol{\theta}_\phi)}^{\text{edge}}$$

- **Edge Potential**
  - Use neighborhood to find and enforce boundaries



$$\phi(c_i,c_j,\mathbf{g}_{ij}(\mathbf{x});\boldsymbol{\theta}_\phi) = -\boldsymbol{\theta}_\phi^T \mathbf{g}_{ij}(\mathbf{x})\delta(c_i \neq c_j).$$

- Main idea:

$$\mathbf{g}_{ij} = [\exp(-\beta\|x_i - x_j\|^2), 1]^T$$

  - If class is the **same**, then the pixel **difference** should be **small**
  - If class is **different**, then the pixel **difference** should be **big**

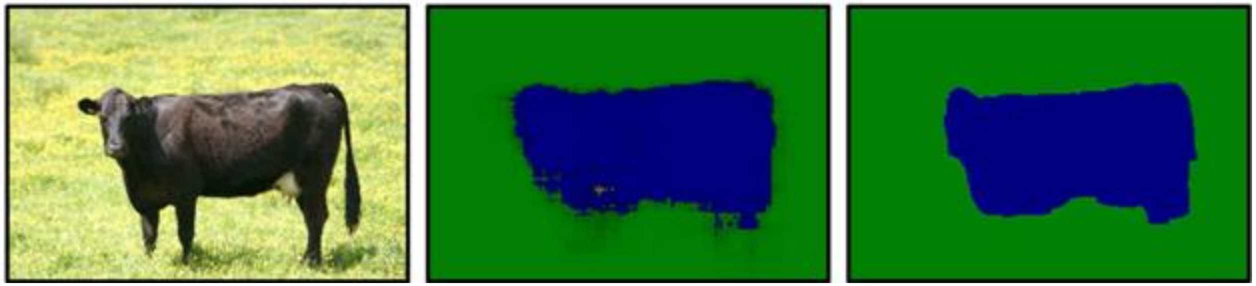
- This is a Pott's model
  - Efficient inference on CRF with graph-cuts

- $\boldsymbol{\theta}_\varphi$ hand tuned with validation data

Slides from Shotton's ECCV talk

Progress

- Cumulative Results



shape-texture

+ edge

Shape-texture potentials only: 69.6%

+ edge potentials: 70.3%

pixel-wise segmentation accuracies

Augmenting the model

- Can we improve?
  - Add pixel color information and a prior on class locations in the image

- Final TextonBoost CRF model

$$\log P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i,\mathbf{x};\boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \overbrace{\pi(c_i,\mathbf{x}_i;\boldsymbol{\theta}_\pi)}^{\text{color}} + \overbrace{\lambda(c_i,i;\boldsymbol{\theta}_\lambda)}^{\text{location}}$$

$$+ \sum_{(i,j)\in\mathcal{E}} \overbrace{\phi(c_i,c_j,\mathbf{g}_{ij}(\mathbf{x});\boldsymbol{\theta}_\phi)}^{\text{edge}} - \log Z(\boldsymbol{\theta},\mathbf{x})$$

A prior on class location

- **Location Potential** $\lambda_i(c_i, i; \boldsymbol{\theta}_\lambda) = \log \boldsymbol{\theta}_\lambda(c_i, \hat{i})$
- Create normalized image coordinates for all images
- Lookup the count of queried class at normalize location in training set

$$\boldsymbol{\theta}_\lambda(c_i, \hat{i}) = \left(\frac{N_{c,\hat{i}} + \alpha_\lambda}{N_{\hat{i}} + \alpha_\lambda} \right)^{w_\lambda}$$

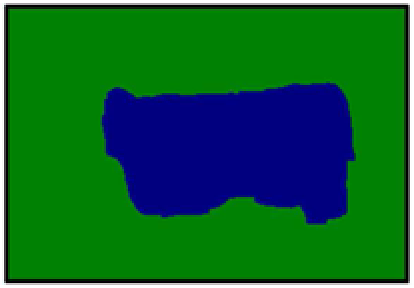
Prevent overfit (tuned)

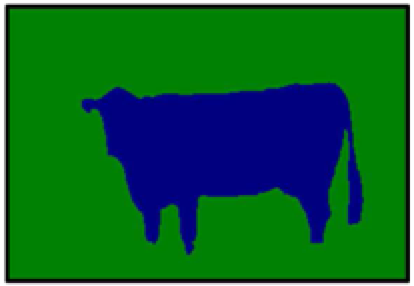
Think Naïve Bayes

- $N_{cow,\star} = 1 \quad N_{\star} = 3$

Modeling color

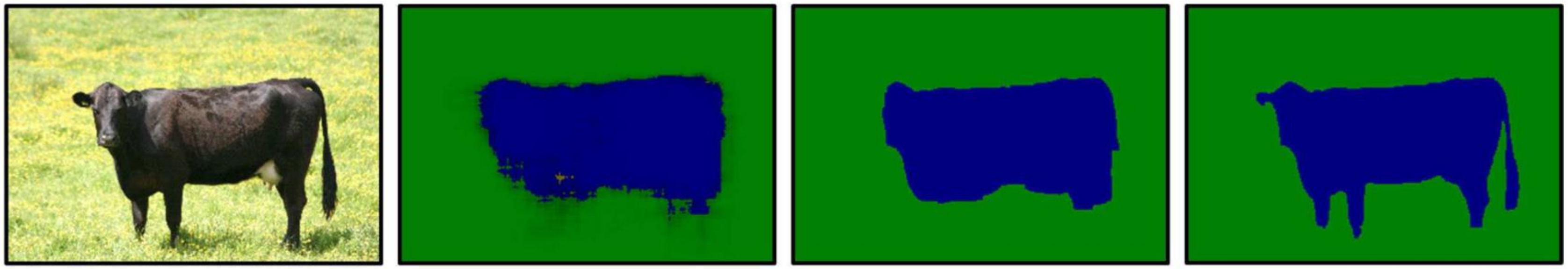
- **Color potential** $\overbrace{\pi(c_i, \mathbf{x}_i; \theta_\pi)}^{\text{color}}$
- **Motivation**: hard to learn model for color across many images due to illumination variances
  - Solution: learn potential independently on each image
- **Main idea**:
  - Use the classification from other potentials as a prior
  - Examine the distribution of color with respect to classes
  - **Keep the classification color-consistent**
    - ✓ Ex: Pixels associated with cows are black → remaining black pixels in the image should be a cow
- (Convoluted) Approach:
  - Gaussian Mixture Model over image CIELab
    - ✓ (Distribution of color)
  - Iteratively weight components using EM-like approach
    - ✓ Inference to get initial image labeling
    - ✓ Weight components so similar color components have same class
    - ✓ Repeat





Putting it together

- Cumulative Results



Shape-texture potentials only:	69.6%	pixel-wise segmentation accuracies
+ edge potentials:	70.3%	
+ colour potentials:	72.0%	
+ location potentials:	72.2%	

Learning reminder

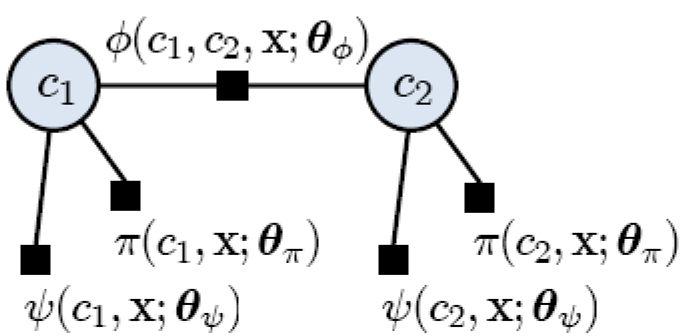
- The TextonBoost CRF model

$$\log P(\mathbf{c}|\mathbf{x},\boldsymbol{\theta}) = \sum_i \overbrace{\psi_i(c_i,\mathbf{x};\boldsymbol{\theta}_\psi)}^{\text{shape-texture}} + \overbrace{\pi(c_i,\mathbf{x}_i;\boldsymbol{\theta}_\pi)}^{\text{color}} + \overbrace{\lambda(c_i,i;\boldsymbol{\theta}_\lambda)}^{\text{location}}$$

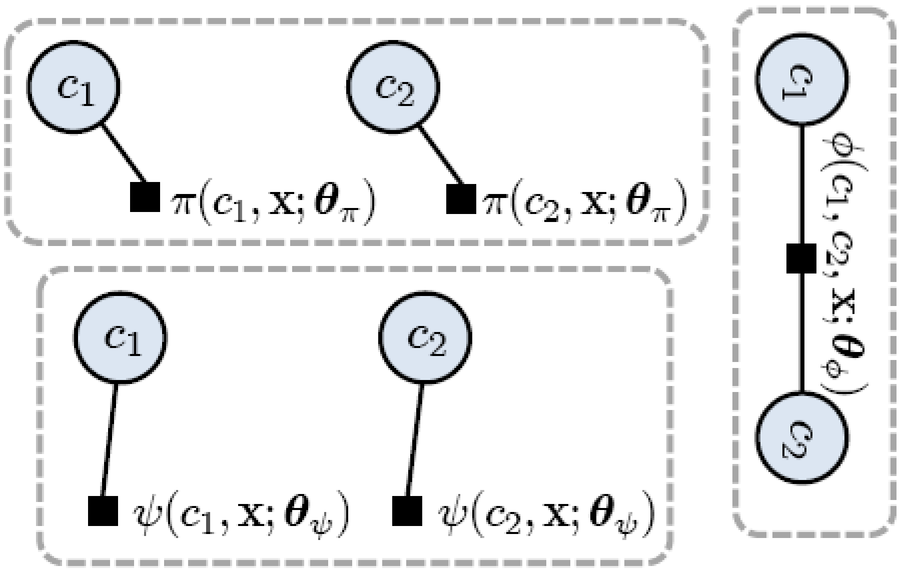
$$+ \sum_{(i,j)\in\mathcal{E}} \overbrace{\phi(c_i,c_j,\mathbf{g}_{ij}(\mathbf{x});\boldsymbol{\theta}_\phi)}^{\text{edge}} - \log Z(\boldsymbol{\theta},\mathbf{x})$$

  - 4-neighborhood graph

- Parameters learned **independently**

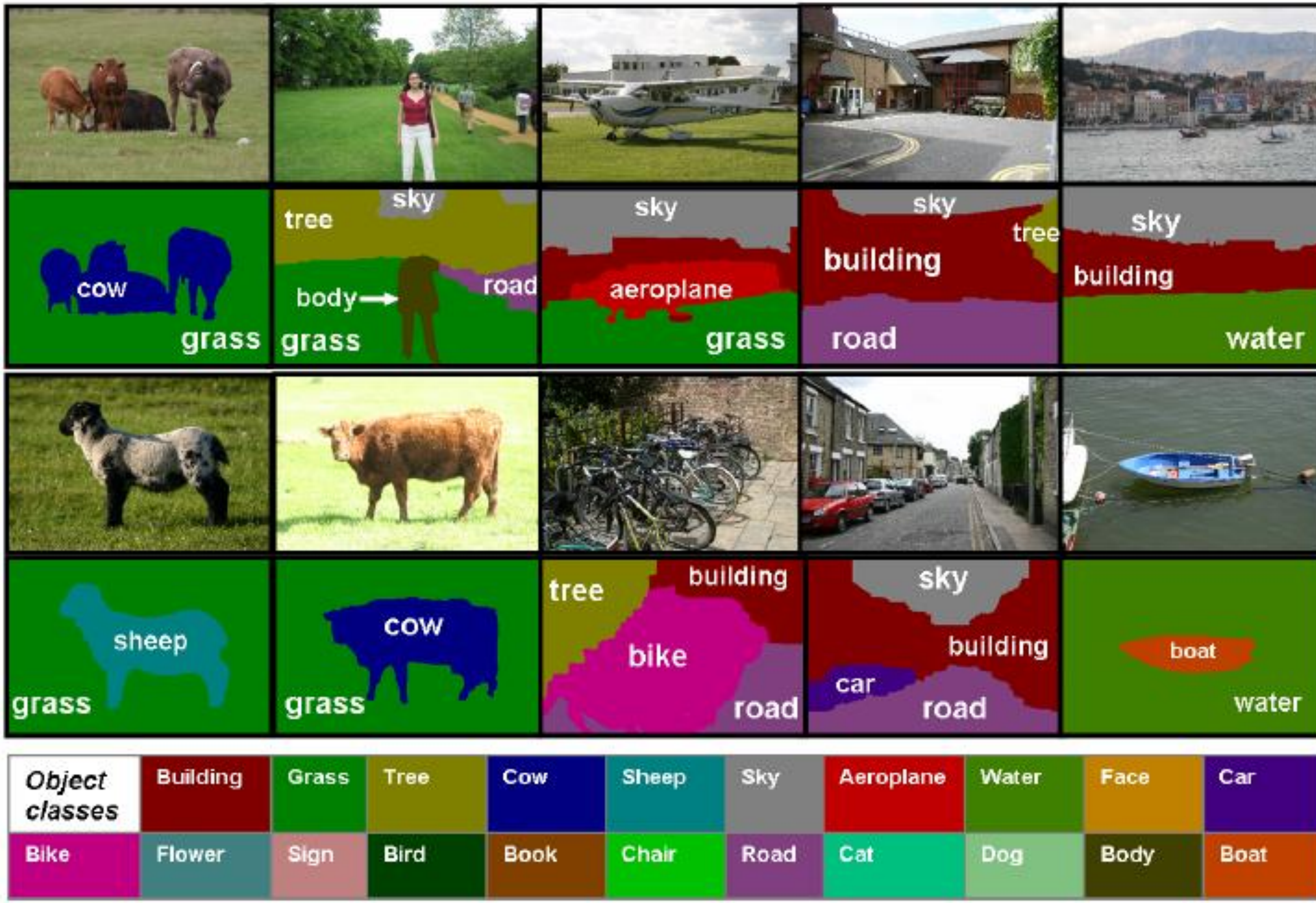


VS



Results

- Successes



Results

- Failures

Results

- Quantitative results on MSRC-21

True class \ Inferred class	building	grass	tree	cow	sheep	sky	aeroplane	water	face	car	bike	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	**61.6**	4.7	9.7	0.3		2.5	0.6	1.3	2.0	2.6	2.1		0.6	0.2	4.8		6.3	0.4		0.5	
grass	0.3	**97.6**	0.5								0.1									1.3	
tree	1.2	4.4	**86.3**	0.5		2.9	1.4	1.9	0.8	0.1							0.1		0.2	0.1	
cow		30.9	0.7	**58.3**				0.9	0.4			0.4			4.2					4.1	
sheep	16.5	25.5	4.8	1.9	**50.4**									0.6			0.2				
sky	3.4	0.2	1.1			**82.6**		7.5									5.2				
aeroplane	21.5	7.2				3.0	**59.6**	8.5													
water	8.7	7.5	1.5	0.2		4.5		**52.9**		0.7	4.9			0.2	4.2		14.1	0.4			
face	4.1		1.1						**73.5**	7.1					8.4			0.4	0.2	5.2	
car	10.1		1.7							**62.5**	3.8		5.9	0.2			15.7				
bike	9.3		1.3							1.0	**74.5**		2.5			3.9	5.9		1.6		
flower		6.6	19.3	3.0								**62.8**			7.3		1.0				
sign	31.5	0.2	11.5	2.1		0.5		6.0		1.5		2.5	**35.1**		3.6	2.7	0.8	0.3		1.8	
bird	16.9	18.4	9.8	6.3	8.9	1.8		9.4						**19.4**			4.6	4.5			
book	2.6		0.6						0.4			2.0			**91.9**					2.4	
chair	20.6	24.8	9.6	18.2		0.2					3.7				1.9	**15.4**	4.5		1.1		
road	5.0	1.1	0.7					3.4	0.3	0.7	0.6		0.1	0.1	1.1		**86.0**			0.7	
cat	5.0		1.1	8.9				0.2		2.0					0.6		28.4	**53.6**	0.2		
dog	29.0	2.2	12.9	7.1				9.7							8.1		11.7		**19.2**		
body	4.6	2.8	2.0	2.1	1.3	0.2			6.0	1.1					9.9		1.7	4.0	2.1	**62.1**	
boat	25.1		11.5			3.8		30.6		2.0	8.6		6.4	5.1			0.3				**6.6**

- Overall pixel-wise accuracy is 72.2%
  - ~15 times better than chance if evenly guessing
  - **What if guessing proportional to the distribution of pixels per class?**
  - **What are the precision rates?**

Comparison with previous work

	Accuracy		Speed (Train/Test)	
	Sowerby	Corel	Sowerby	Corel
This paper – Full CRF model	88.6%	74.6%	5h/10s	12h/30s
This paper – Unary classifier only	85.6%	68.4%		
He et al. – mCRF model [1]	89.5%	80.0%	Gibbs	Gibbs
He et al. – unary classifier only	82.4%	66.9%		

Table 1. Comparison of segmentation/recognition accuracy and efficiency.

Discussion

- What I like about this paper:
  - Classification of many classes
  - Publicly released database
  - Simple approach (minus color potential)

- What I dislike about this paper:
  - Training is ad-hoc
  - Multiple parameters are set by hand
  - Doesn’t improve on referenced work [he-cvpr-04]

Training data split (MSRC-21)

- Distribution of data over training split
  - 7 out of 21 classes > 5% of pixels

building	10.8
grass	19.0
tree	9.1
cow	**3.2**
sheep	**2.2**
sky	9.5
aeroplane	**1.6**
water	8.3
face	**1.8**
car	**3.3**
bicycle	**2.8**
flower	**2.6**
sign	**1.9**
bird	**1.5**
book	5.3
chair	**1.8**
road	9.3
cat	**1.7**
dog	**1.5**
body	**2.3**
boat	**0.7**

Testing data split (MSRC-21)

- Distribution of data over testing split
  - 7 out of 21 classes > 5% of pixels
  - Similar proportions to training split
- Guess random, proportionally → ~9% chance
- TextonBoost is 8 times better than chance

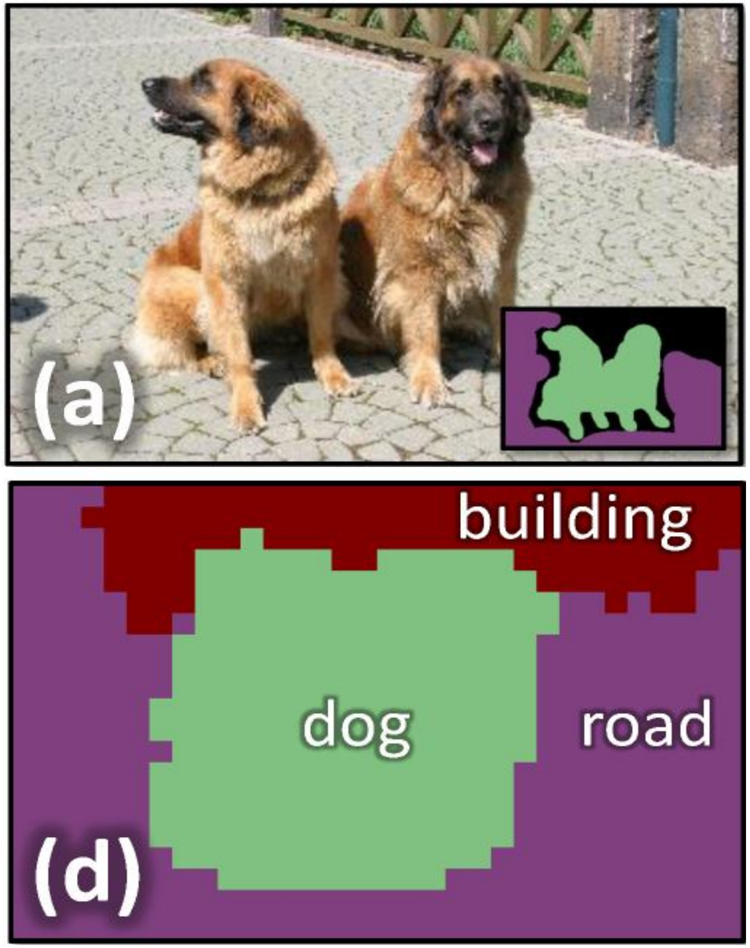
building	10.4
grass	19.8
tree	8.4
cow	**2.9**
sheep	**2.3**
sky	9.8
aeroplane	**1.3**
water	7.8
face	**1.8**
car	**3.4**
bicycle	**2.5**
flower	**3.5**
sign	**3.0**
bird	**1.3**
book	5.3
chair	**2.0**
road	8.1
cat	**1.4**
dog	**2.1**
body	**1.9**
boat	**1.0**

[shotton-cvpr-08]

- [shotton-cvpr-08] J. Shotton, M. Johnson, R. Cipolla, *Semantic Texton Forests for Image Categorization and Segmentation*, CVPR 2008

Overview

- Goal: (same as before)
- Motivation:
  - 1) Visual words approach is slow
    - ✓ Compute feature descriptors
    - ✓ Cluster
    - ✓ Nearest-neighbor assignment
  - 2) CRF is even slower
    - ✓ Inference always a bottle-neck
- Approach: operate on pixel values
  - Simple & efficient
- Result: works well and efficiently



Overview

- Contributions
  - Semantic Texton Forests: local classification with hierarchical information
  - The Bag of Semantic Textons Model
  - Image-level prior to improve semantic segmentation

- Quick decision tree review…

Decision Trees

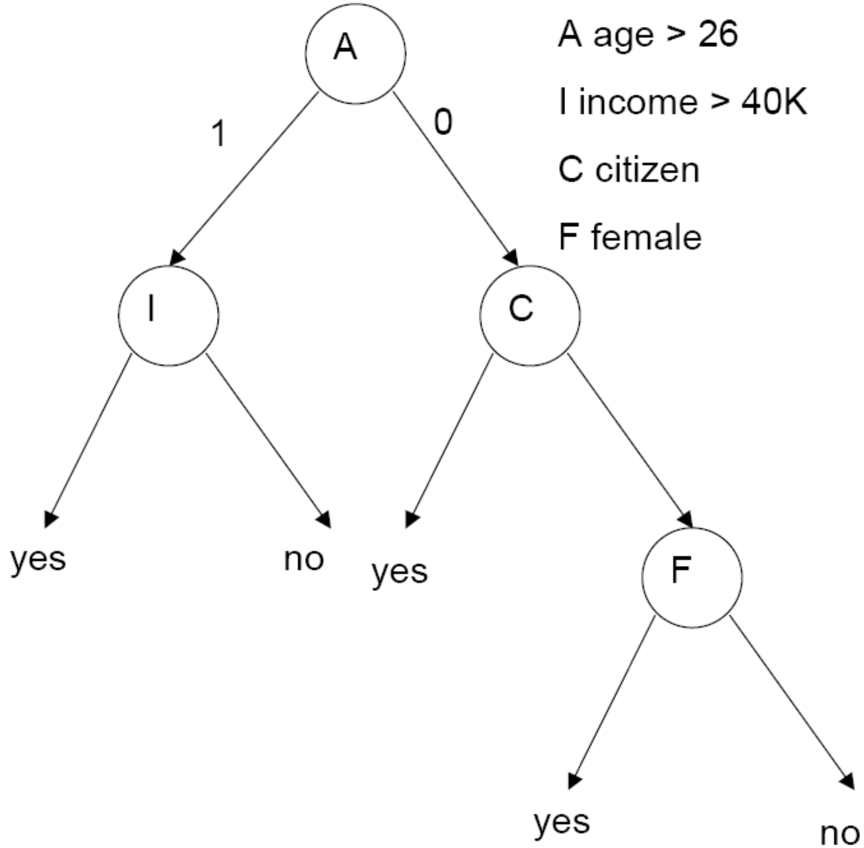
- Who here has a car?

Structure of a decision tree

• Internal nodes correspond to attributes (features)

• Leafs correspond to classification outcome

• edges denote assignment



A age > 26

I income > 40K

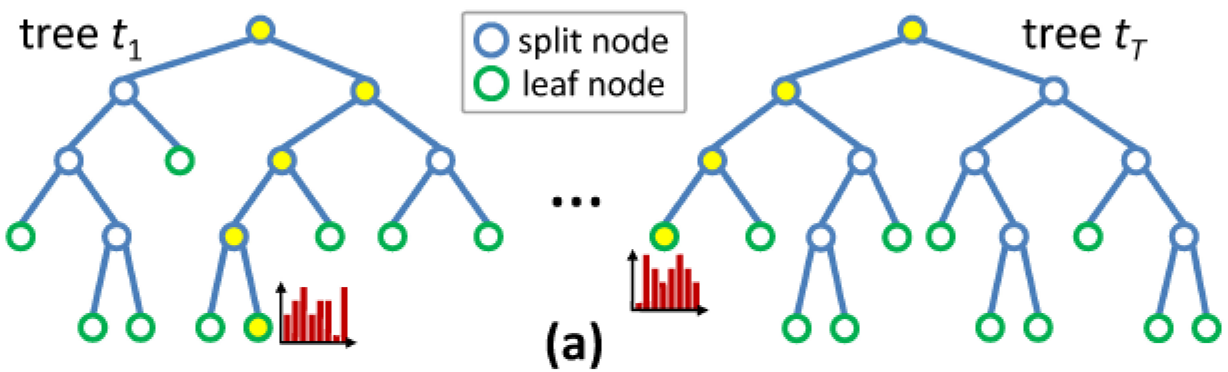
C citizen

F female

- Advantages?
- Drawbacks?

Encoding decisions

- Randomized Decision Forests
  - Input: “features” describing pixel
  - Output: Predicted class distribution
- Approach
  - Each node n in the decision tree contains an empirical class distribution $P(c|n)$
  - **Important**: Learn decision trees such that similar “features” should end up at the **same leaf nodes**

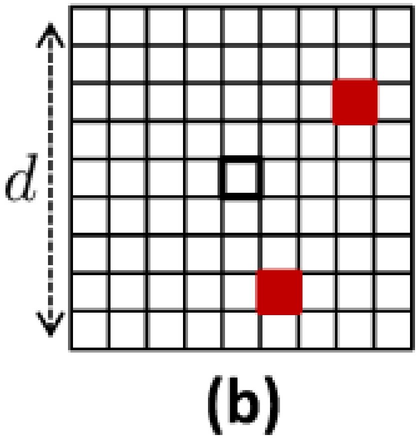


  - The leaves $L = \{l_i\}$ of a tree contain most discriminative information
    - ✓ Classify by averaging

$$P(c|L) = \frac{1}{T}\sum_{t=1}^{T} P(c|l_t)$$

- Another histogram of texton-like per pixel!

Features?



- Think of the **simplest** features you can do.
- Center a *d*-by-*d* patch around a pixel (5x5)
- Possible features:
  - Feature #1: its value in a color channel (CIELab)
  - Feature #2: the sum of two points in the patch
  - Feature #3: the difference of two points in the patch
  - Feature #4: the absolute difference of two points in the patch
- Feature invariance accounted for by rotating, scaling, flipping, affine-ing training data

- Random Decision Tree training:
  - Take random subset of training data
  - Generate random features *f* from above
  - Generate random threshold *t*
  - Split data into left I_l and right I_r subsets according to
  - Repeat for each side
- **Does this actually work?**

This feature maximizes information gain

$$I_l = \{i \in I_n \mid f(\mathbf{v}_i) < t\}$$
$$I_r = I_n \setminus I_l \,.$$

Filters found

- Yes

- Each **patch** represents one leaf node. It is the summation of all the patches from the training data that fell into that leaf.
- Learns colors, orientations, edges, blobs

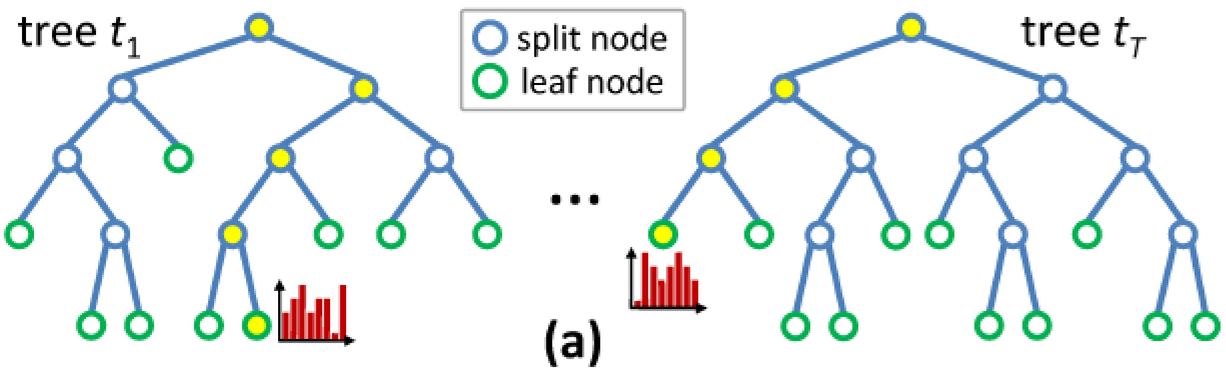
Simple model results

- Semantic Texton Forests are better than chance (~5%)

	Global	Average
supervised	49.7%	34.5%
weakly supervised	14.8%	24.1%

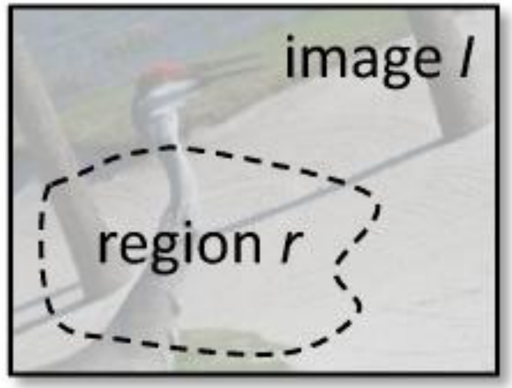
  - MSRC-21 dataset

- Supervised = 1 label per pixel
  - Increase one bin in the histogram at a time
- Weakly-supervised = all labels in image per pixel
  - Increase multiple bins in the histogram at a time

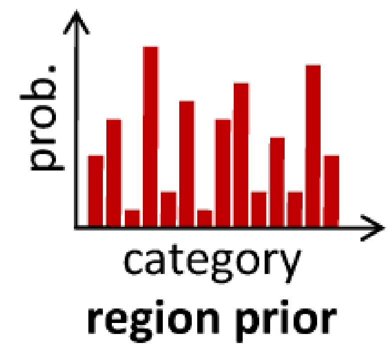


Adding tricks to the model

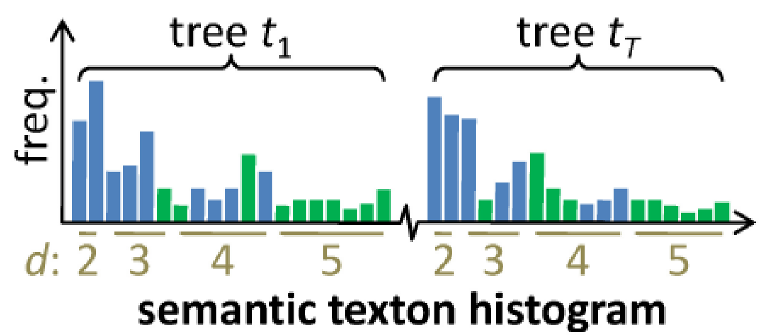
- More extensions with this model: **Bags of Semantic Textons**



- How can we get a prior estimate for what is in region *r*?
- 2 Options:
  - 1) Average leaf histograms in region *r* together $P(c|r)$
    - ✓ Good for segmentation priors



  - 2) Create hierarchy histogram of **node counts** $H_r(n)$ visited in the tree for each classified pixel in region *r*
    - ✓ Want testing and training **decision paths** to match



Histogram-based Classification

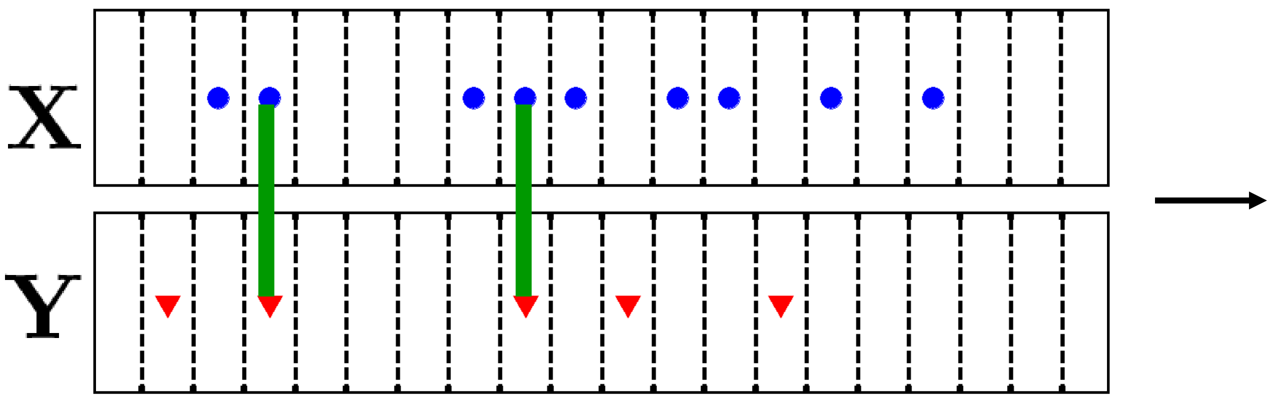
- Main idea:
  - Have 2 vectors as features
    - ✓ (training-tree's histograms, testing-tree's histograms)
  - Want to measure similarity to do classification

- Proposed approach: Kernalized SVM
  - Kernel = Pyramid Match Kernel (PMK)
  - Computes a histogram distance, using hierarchy information
  - Train 1-vs-all classifiers

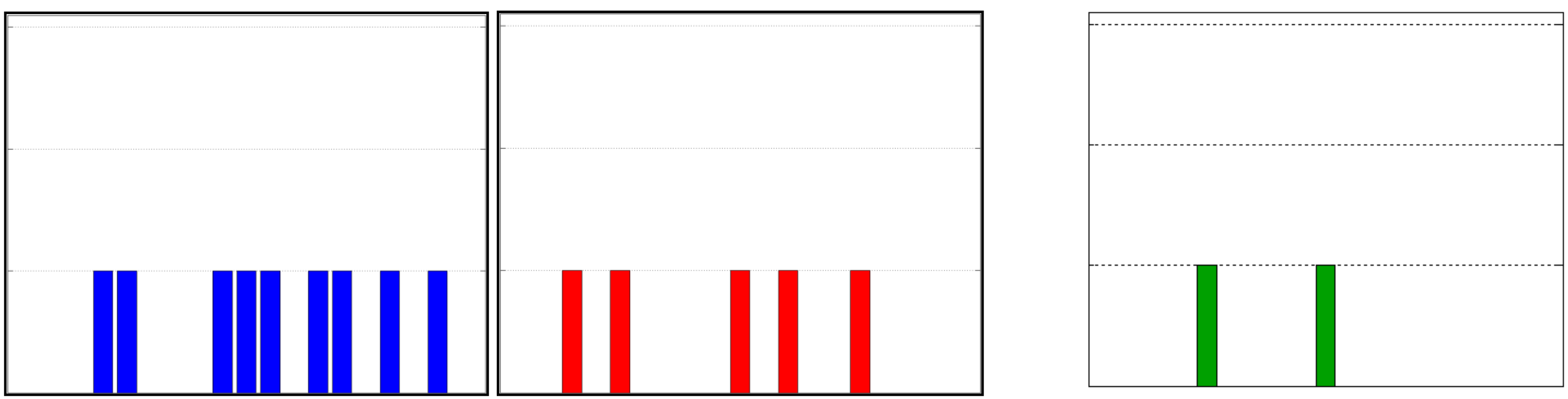
- Review on Pyramid Match Kernel…

Example pyramid match

Level 0



$$N_0 = 2$$
$$w_0 = 1$$

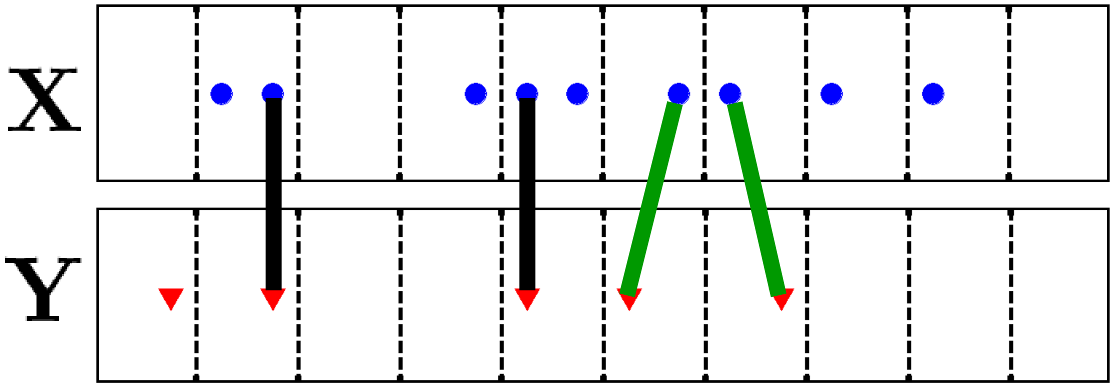


$H_0(\mathbf{X})$ $H_0(\mathbf{Y})$ $\mathcal{I}_0 = 2$

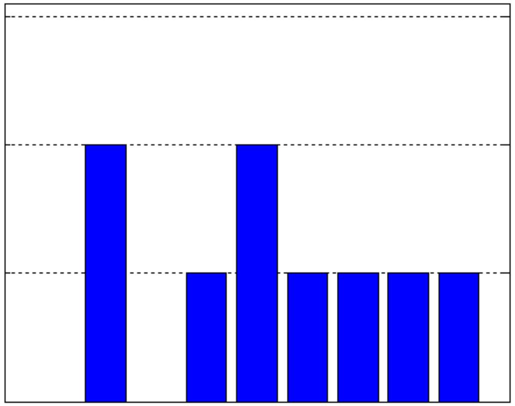
Slides from Grauman's ICCV talk

Example pyramid match

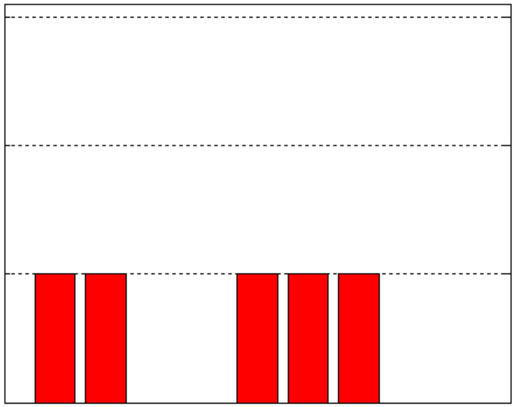
Level 1



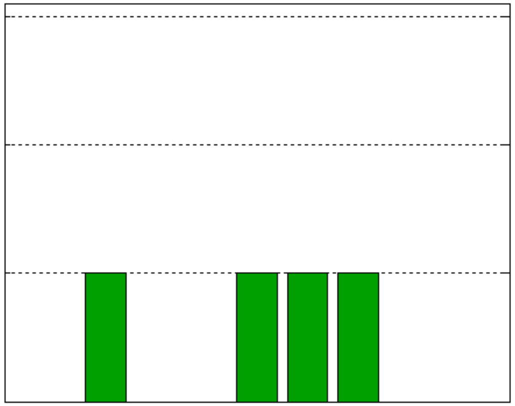
$$N_1 = 4 - 2 = 2$$
$$w_1 = \frac{1}{2}$$



$H_1(\mathbf{X})$



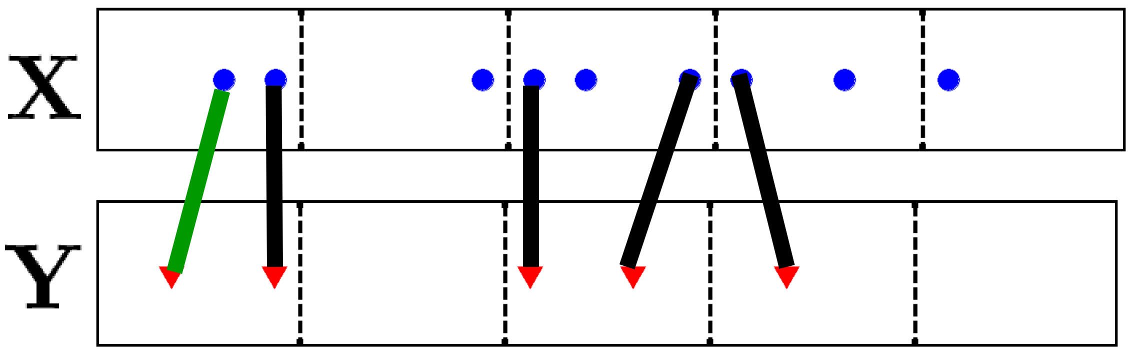
$H_1(\mathbf{Y})$



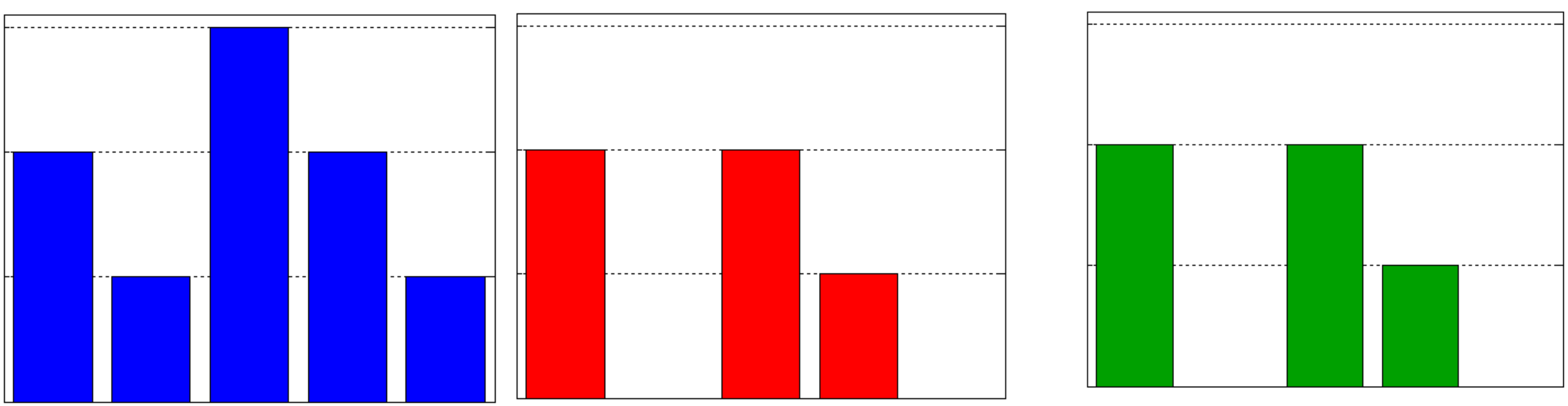
$\mathcal{I}_1 = 4$

Example pyramid match

Level 2



$$\rightarrow \quad N_2 = 5 - 4 = 1, \qquad w_2 = \frac{1}{4}$$



$H_2(\mathbf{X})$ $\quad H_2(\mathbf{Y})$ $\quad \mathcal{I}_2 = 5$

Scene Categorization

- The whole image is one region
  - Using histogram matching approach
  - End result is an **Image-level Prior**

- Comparison with other similarity metric (radial basis function, RBF)
  - Unfair? RBF uses only leaf-level counts, PMK uses entire histogram

- Results

	Global kernel K	Per-category kernel K_c
RBF	49.9	52.5
PMK	76.3	**78.3**

Table 2. **Image categorization results.** (Mean AP).

  - K_c = trick to account for unbalanced classes
  - Note Mean Average Precision reported here, but **not elsewhere**

- Number of trees has diminishing returns

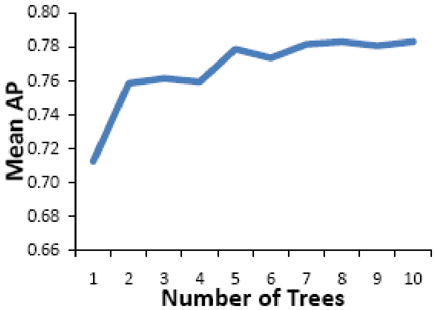
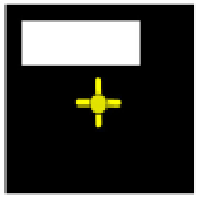


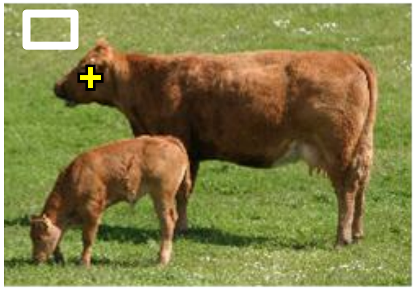
Figure 5. **Categorization accuracy vs number of STF trees.**

Improving Semantic Segmentation

- Use idea of **shape-filters** to improve classification
- Main idea: **After** initial STF classification, learn how a pixel's class interacts with neighboring regions' classes



- Approach: Learn a *second* random decision forest (segmentation forest)
  - Use **different** weak features:
    - ✓ Histogram count at some level $H_{r+i}(?)$
    - ✓ Region prior probability of some class $P(? \mid r+i)$



- Difference with shape filters:
  - Shape-filters learn: cow is adjacent to green-like texture
  - Segmentation forest learn: cow is adjacent to grass

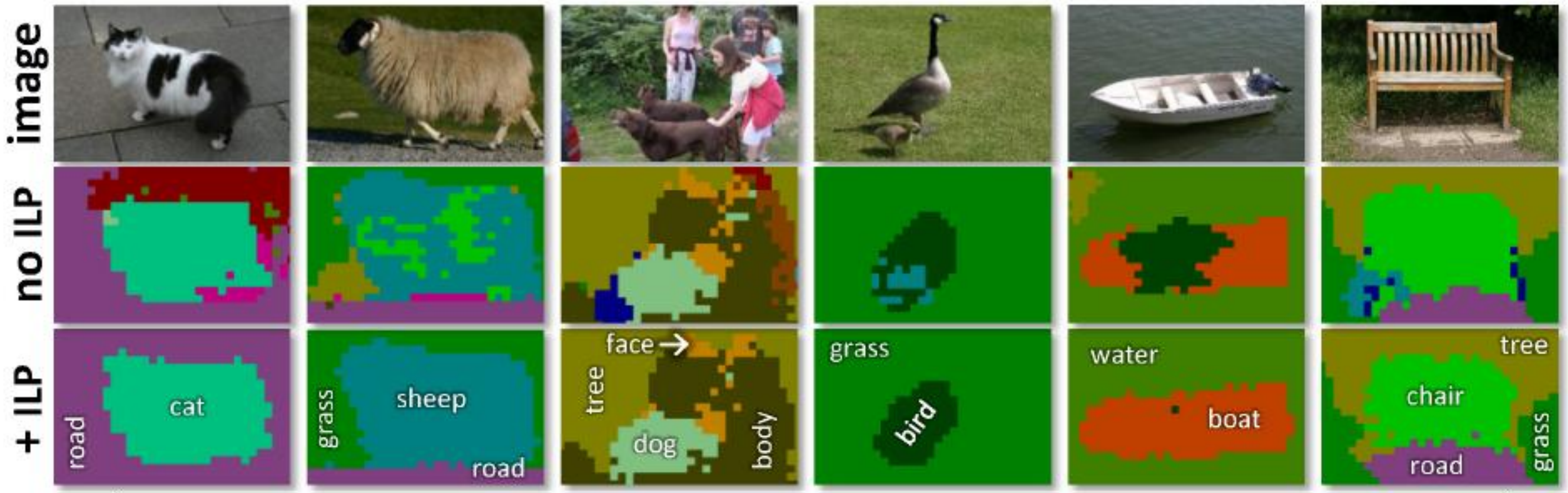
- Trick: multiply with image-level prior for best results
  - Convert SVM decision to probability

Computation time

- **Fast**
  - STF feature extraction = 275 ms
  - Image categorization = 190 ms
  - Segmentation forest = 140 ms
  - Total ~ 605 ms

- TextonBoost = 6000 ms

MSRC-21 Results



	building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat	**Global**	**Average**
[27]	**62**	**98**	**86**	58	50	83	60	53	74	63	**75**	63	35	19	92	15	86	54	19	62	7	71	58
[32]	52	87	68	73	84	**94**	**88**	**73**	70	68	74	**89**	33	19	78	34	**89**	46	**49**	54	**31**	-	64
Ours	41	84	75	89	93	79	86	47	**87**	65	72	61	**36**	26	91	50	70	72	31	61	14	68	63
Ours + ILP	49	88	79	**97**	**97**	78	82	54	**87**	**74**	72	74	**36**	24	**93**	**51**	78	**75**	35	**66**	18	**72**	**67**

VOC 2007 Segmentation

	background	aeroplane	bicycle	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motorbike	person	plant	sheep	sofa	train	tv / monitor	**Average**
Brookes	**78**	6	0	0	0	0	9	5	**10**	1	2	11	0	6	6	29	2	2	0	11	1	9
Ours	33	46	5	14	**11**	14	**34**	8	6	3	10	39	**40**	28	23	32	19	**19**	**8**	24	9	20
Ours + ILP	20	**66**	**6**	**15**	6	**15**	32	**19**	7	**7**	**13**	**44**	31	**44**	**27**	**39**	**35**	12	7	**39**	**23**	**24**
TKK	**23**	19	21	5	16	3	1	**78**	1	3	1	23	**69**	44	42	0	**65**	**30**	**35**	**89**	71	30
Ours + DLP	22	**77**	**45**	**45**	**19**	**14**	**45**	48	**29**	**26**	**20**	**59**	45	**54**	**63**	**37**	40	**42**	10	68	**72**	**42**

Discussion

- What I like about this paper:
  - Simple concept
  - Good result
  - Works fast (testing & training)

- What I dislike about this paper:
  - More difficult to understand
  - Low-resolution classification
    - ✓ Segmentation forest operates at patches
  - Test-time inference is dependent on amount of training
    - ✓ Must iterate through all trees in the forest at test time
  - Many “Implementation Details” scattered through the paper.
    - ✓ What is the trick to get it to work?
  - How dependent is the performance on decision tree parameters?