Multi-Class Segmentation with Relative Location Prior

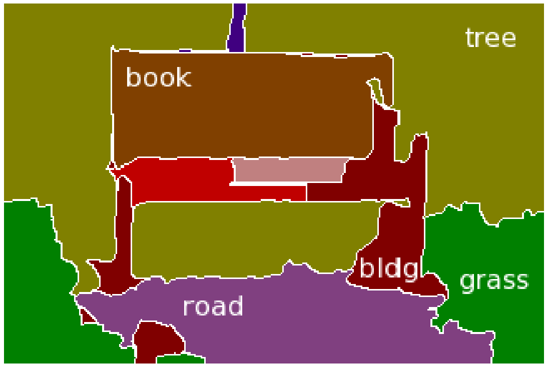
Stephen Gould, Jim Rodgers, David Cohen, Gal Elidan and Daphne Koller
Department of Computer Science, Standford University

Present by: Miss Naddao Thongsak

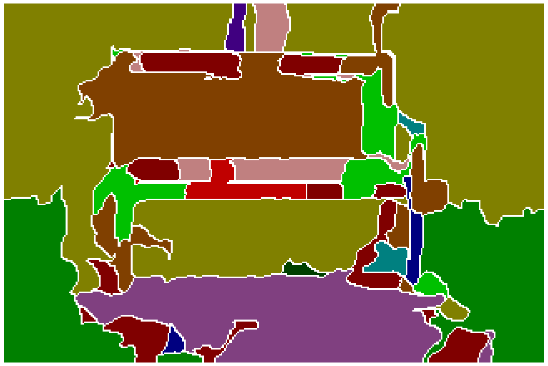
Introduction

(a) Base image

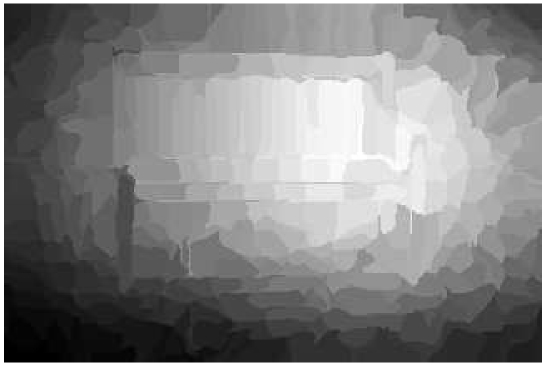
(b) Superpixel over-segmentation



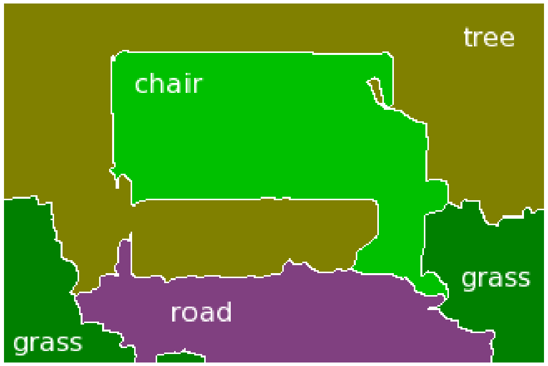
(c) Baseline CRF segmentation



(d) First-stage predictions



(e) Relative location feature (for *chair*)



(f) Second-stage segmentation

Image segmentation

- Over-segmentation algorithm
- Get superpixels region
- $V(I) = \{S_1, S_2, ... S_n\}$ for image I

Appearance Features

- Each superpixel, 83-dimension description incorporating region size, location, color, shape and texture features. Build it with the way of Barnard et al.
- Append the weight average of appearance over the neighbors for each superpixel to description vector

$$\frac{\sum_{S_j \in \mathcal{N}(S_i)} |S_j| \cdot \phi(S_j)}{\sum_{S_j \in \mathcal{N}(S_i)} |S_j|}$$

$$\mathcal{N}(S_i) = \{S_j \mid (S_i, S_j) \in \mathcal{E}(\mathcal{I})\}$$

is the set of super pixels which are neighbors of Si in the images and |Sj| is the number of pixels in superpixel Sj

Appearance features

- Apply AdaBoost that have learned for each class c' to the vector of descriptors and normalize over all classes to get the probability

$$P^{\text{app}}(c_i = c' \mid S_i, \mathcal{I}) = \frac{\exp\{\sigma_{c'}\}}{\sum_c \exp\{\sigma_c\}}$$

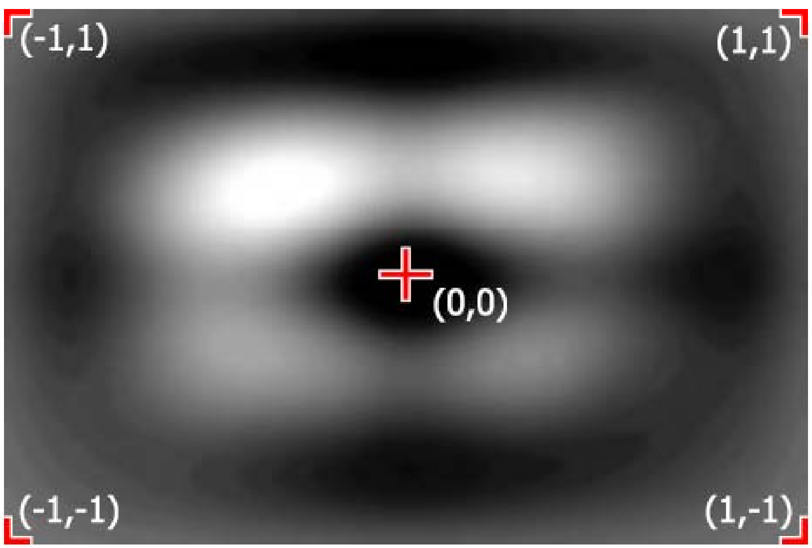
σ_c output of the AdaBoost classifier for class c

Then, define appearance features as:

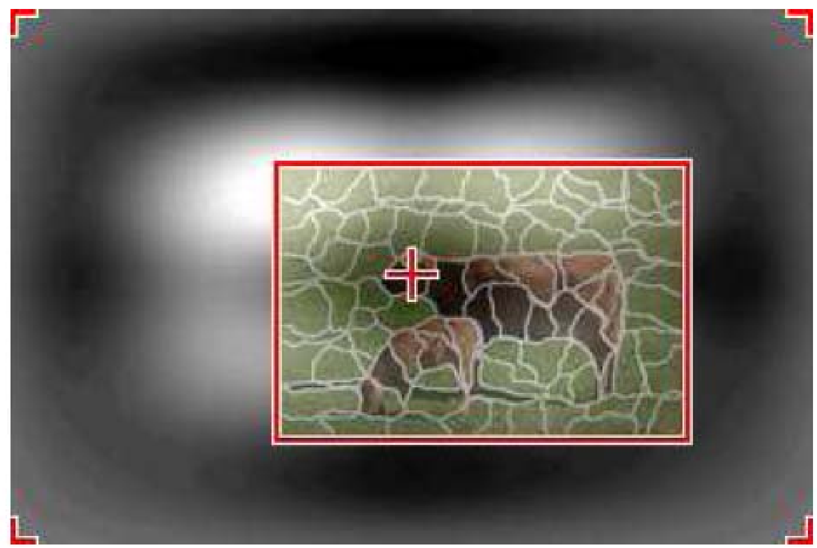
$$f^{\text{app}}(S_i, c_i, \mathcal{I}) = w_{c_i}^{\text{app}} \cdot \log P^{\text{app}}(c_i \mid S_i, \mathcal{I})$$

Relative Location Probability Maps

- Map $M_{c|c'}(u,v)$: encode the probability that pixel p at offset (u,v) from p'(with labeled c') has label c
- $M_{c|c'}(u,v)$ is normalized image coordinates (u,v) E [-1,1]x[-1,1]



(a) Relative location prior (*grass* to *cow*)



(b) Computing relative location offsets

Relative Location Features

1

Input : $\mathcal{I}$ // test image

$\mathcal{H}$ // learned models

$\{\mathcal{M}_{c|c'}\}$ // relative location maps

Output: $\mathcal{L}$ // pixel labels for $\mathcal{I}$

2

// **Initial prediction:** appearance features only

foreach *superpixel* $S_i \in \mathcal{I}$ **do**

  $\hat{c}_i \leftarrow \text{argmax}_c\, P^{\text{app}}(c \mid S_i, \mathcal{I})$

end

3

// Compute relative location votes (Eq. 2,3)

foreach *superpixel* $S_i \in \mathcal{I}$ **do**

  foreach *class* c **do**

$$v_c^{\text{other}}(S_i) = \sum_{j \neq i : \hat{c}_j \neq \hat{c}_i} \alpha_j \cdot \mathcal{M}_{c|\hat{c}_j}(\hat{x}_i - \hat{x}_j, \hat{y}_i - \hat{y}_j)$$

$$v_c^{\text{self}}(S_i) = \sum_{j \neq i : \hat{c}_j = \hat{c}_i} \alpha_j \cdot \mathcal{M}_{c|\hat{c}_j}(\hat{x}_i - \hat{x}_j, \hat{y}_i - \hat{y}_j)$$

  end

end

4

// Compute relative location feature (Eq. 4)

foreach *superpixel* $S_i \in \mathcal{I}$ **do**

  foreach *class* c_i **do**

$$f^{\text{relloc}}(S_i, c_i, \mathcal{I}) = w_{c_i}^{\text{other}} \cdot \log v_{c_i}^{\text{other}}(S_i) + w_{c_i}^{\text{self}} \cdot \log v_{c_i}^{\text{self}}(S_i)$$

  end

end

5

// **Final prediction with relative location:**

// Logistic regression (Eq. 6) or CRF (Eq. 8)

$\hat{\mathbf{c}} \leftarrow \text{argmax}_{\mathbf{c}}\, P(\mathbf{c} \mid \mathcal{I}; \mathbf{w})$

foreach *superpixel* $S_i \in \mathcal{I}$ **do**

  foreach *pixel* $p \in S_i$ **do**

    $\mathcal{L}[p] \leftarrow \hat{c}_i$

  end

end

return $\mathcal{L}$

Complexity

Benefit from using superpixels is that complexity of their methods depends on the inherent complexity of image rather than resolution

Complexity

Sowerby database images

200 superpixels

- computing the relative loaction : 0.2 s
- pre-partition into 200 superpixels: 5.3 s

400 superpixels

- Compute relative location 0.2 s
- Running Inference 31 s

Simple (Logistic Regression) model

- independently to each superpixel, they define feature function into f_{relloc}, f_{app}, and bias term $f^{bias}(c_i)= w^{bias}_{ci}$

$$P(c_i \mid \mathcal{I}; w) \propto \exp\left\{ f^{\mathrm{relloc}}(S_i, c_i, \mathcal{I}) \quad + f^{\mathrm{app}}(S_i, c_i, \mathcal{I}) + f^{\mathrm{bias}}(c_i) \right\}$$

- The weight for logistic model:

$$w = \{w_{c_i}^{\mathrm{other}}, w_{c_i}^{\mathrm{self}}, w_{c_i}^{\mathrm{app}}, w_{c_i}^{\mathrm{bias}} \mid c_i = 1, \ldots, K\}$$

Conditional Random Field

- Pairwise affinity potential
- Such as: given two adjadcent superpixels, a pairwise feature might assign a greater value for labeling of cow and grass than it would for cow and airplane, because cows are often next to grass and rarely next to airplane.
- In addition to , features define for logistic model, we define the pairwise feature between all adjacent pixels

$$f^{\text{pair}}(c_i, c_j, \mathcal{I}) = \frac{w^{\text{pair}}_{c_i, c_j}}{0.5(d_i(\mathcal{I}) + d_j(\mathcal{I}))}$$

Number of superpixels adjacent to Si

Conditional Random Field

Then, the full CRF model is:

$$P(\boldsymbol{c} \mid \mathcal{I}; \boldsymbol{w}) \propto$$
$$\exp\left\{ \sum_{S_i \in \mathcal{V}(\mathcal{I})} \left(f^{\text{relloc}}(S_i, c_i, \mathcal{I}) + f^{\text{app}}(S_i, c_i, \mathcal{I}) + f^{\text{bias}}(c_i) \right) + \sum_{(S_i, S_j) \in \mathcal{E}(\mathcal{I})} f^{\text{pair}}(c_i, c_j, \mathcal{I}) \right\}$$

- The first summation is over individual superpixels
- The second summation is over pairs of adjacent superpixels

Experiment Result

- MSRC Databases [21-class and 9-class objects]
- Corel and Sowerby Databases [7 classes objects]

Experiment Result

- Random separate into balance between testing and training

Training

- train the boosted appearance classifier
- constructing the image dependent relative location prior
- train parameter for logistic regression
- train parameter for CRF models

Testing

- Use the rest to be considered
- Use train parameter to run in testing

Experiment Result

- Each experiment will be tested

1. Baseline logistic regression classifier
2. Baseline CRF
3. Logistic regression model augmented with their image-dependent relative location feature
4. CRF model augmented with their relative location feature.

Note: to ensure robustness, 5 different train/test partitionings for large database and 10 different random prtitionings for small databases and report minimum and maximum and average performance.

Also: contrast to all state-of-the-art methods that normally do evaluated only a single-fold.

MSRC Databases[21-class]

- 21-class: more comprehensive and complex
- Consisting 591 images of building, grass, tree, cow, sheep, sky, airplane, water, face, car, bicycle, flower, sign, bird, book, chair, road, cat, dog, body, boat.
- Ground truth: foreground label often overlapped with background object and have void class that doesn't fall into 21-class. They skip void class in both train and testing

MSRC Databases[21-class]

Algorithm	21-class MSRC Accuracy Min.	Avg.	Max.	Std.	9-class MSRC Accuracy Min.	Avg.	Max.	Std.
Shotton *et al.* [24]		72.2%*		n/a		-		-
Yang *et al.* [31]		75.1%*		n/a		-		-
Schroff *et al.* [22]		-		-		75.2%*		n/a
Baseline Logistic	61.8%	63.6%	65.4%	1.67%	77.5%	78.9%	79.8%	1.05%
Baseline CRF	68.3%	70.1%	72.0%	1.81%	81.2%	83.0%	84.4%	1.28%
Logistic + Rel. Loc.	73.5%	75.7%	77.4%	1.74%	87.6%	88.1%	88.9%	0.67%
CRF + Rel. Loc.	74.0%	**76.5%**	78.1%	1.82%	87.8%	**88.5%**	89.5%	0.82%

* For the other works, results are only reported on a single fold

MSRC Databases[21-class]

	building	grass	tree	cow	sheep	sky	airplane	water	face	car	bicycle	flower	sign	bird	book	chair	road	cat	dog	body	boat
building	**72.3** (72.1)	0.8 (1.2)	3.2 (3.5)	1.1 (1.1)	0.6 (0.3)	2.4 (3.4)	0.5 (1.1)	3.2 (2.5)	1.8 (1.3)	2.6 (2.7)	0.4 (0.3)	- -	1.5 (1.4)	0.4 (0.1)	1.6 (2.4)	0.5 (0.2)	5.4 (5.4)	- (0.2)	0.6 (0.3)	0.5 (0.3)	0.5 (0.1)
grass	0.1 (0.4)	**94.8** (94.3)	2.7 (3.4)	0.8 (0.7)	0.3 (0.3)	- -	0.4 (0.3)	0.1 (0.2)	- -	- -	0.1 -	- -	- -	0.1 -	- -	0.1 -	0.2 (0.2)	- -	[illegible] -	0.2 (0.1)	- -
tree	4.6 (6.7)	5.0 (6.6)	**81.3** (79.1)	0.1 (0.3)	- (0.1)	2.2 (2.3)	0.6 (0.5)	1.9 (1.0)	0.2 (0.1)	0.4 (1.2)	0.7 (0.8)	- (0.1)	0.5 (0.1)	0.6 (0.3)	[illegible] -	1.1 (0.2)	0.1 (0.3)	[illegible] -	0.4 (0.2)	0.2 (0.1)	0.1 (0.1)
cow	0.1 (5.6)	14.9 (14.1)	4.2 (3.3)	**66.3** (58.6)	1.8 (4.1)	0.2 (0.3)	- (0.2)	2.0 (3.3)	0.1 (1.2)	- (0.8)	- (0.1)	0.6 (1.6)	0.1 (0.4)	2.3 (1.0)	- (0.1)	0.5 (0.1)	0.1 (0.3)	1.1 (1.4)	5.2 (2.9)	0.4 (0.7)	- -
sheep	- (7.1)	12.1 (11.6)	0.2 (3.4)	2.6 (3.9)	**71.0** (57.9)	- (0.3)	- (0.1)	0.4 (3.1)	0.1 (0.3)	- (0.1)	- (0.1)	- (0.2)	- -	2.3 (1.2)	- -	0.3 -	8.3 (6.9)	- (0.8)	2.7 (2.8)	- (0.2)	- -
sky	2.2 (2.5)	- -	1.0 (0.6)	0.1 -	- -	**92.6** (91.2)	0.5 (0.4)	3.3 (4.4)	- -	0.1 (0.1)	- -	- -	0.2 (0.2)	- (0.1)	- (0.1)	- -	0.1 (0.4)	- -	- -	- -	- -
airplane	20.2 (30.6)	1.8 (2.7)	1.0 (4.1)	- (0.1)	- -	2.3 (1.7)	**73.6** (53.2)	0.3 (0.4)	- -	0.4 (5.6)	- (0.2)	- -	- (0.7)	- -	- -	- (0.1)	0.3 (0.4)	- -	- -	- -	- (0.3)
water	3.6 (5.5)	4.4 (5.3)	3.0 (4.6)	0.2 (0.2)	0.3 (0.3)	4.4 (4.8)	0.1 (0.1)	**69.6** (65.5)	- -	2.3 (2.0)	1.4 (0.7)	0.1 -	- (0.1)	0.2 (0.3)	- (0.2)	0.3 -	9.2 (9.2)	- (0.1)	0.2 (0.2)	0.3 (0.1)	0.4 (0.9)
face	4.2 (11.9)	0.3 (0.5)	1.5 (2.8)	0.9 (3.5)	- (0.1)	0.1 (0.1)	- -	0.1 (0.1)	**70.2** (66.2)	0.1 (0.5)	- (0.2)	1.1 (0.5)	- (0.1)	0.1 -	7.6 (1.4)	0.3 (0.1)	0.1 (0.3)	2.0 (1.8)	0.3 (1.2)	11.3 (8.7)	- -
car	12.5 (17.3)	- (0.1)	3.7 (4.5)	- -	- -	1.6 (2.4)	- (1.8)	5.9 (9.7)	- (0.2)	**68.9** (53.6)	- (0.8)	1.7 (2.3)	0.8 (0.9)	0.4 (0.1)	- (0.7)	- -	3.4 (3.6)	- (0.2)	- -	- (0.4)	1.1 (1.2)

their method

Baseline CRF

MSRC databases[21-class]

car	12.5	-	3.7	-	-	1.6	-	5.9	-	**68.9**	-	1.7	0.8	0.4	-	-	3.4	-	-	-	1.1
	(17.3)	(0.1)	(4.5)	-	-	(2.4)	(1.8)	(9.7)	(0.2)	(53.6)	(0.8)	(2.3)	(0.9)	(0.1)	(0.7)	-	(3.6)	(0.2)	-	(0.4)	(1.2)
bicycle	16.7	0.2	2.5	-	-	-	-	0.8	-	0.8	**71.7**	0.6	-	-	-	1.2	5.2	-	-	0.3	-
	(26.8)	(0.5)	(8.8)	-	-	(0.1)	(0.2)	(0.8)	(0.2)	(5.0)	(50.3)	(0.3)	(0.2)	-	-	(0.1)	(5.1)	(0.1)	-	(1.4)	(0.1)
flower	0.1	2.9	5.6	3.8	1.2	0.6	-	0.2	3.1	-	1.7	**67.6**	2.8	5.9	0.1	-	-	0.1	1.2	3.1	-
	(1.8)	(5.0)	(6.3)	(7.9)	(1.2)	(1.4)	(0.1)	(1.1)	(3.3)	(0.7)	(2.6)	(54.4)	(2.4)	(1.5)	(3.0)	(0.1)	(0.3)	(0.2)	(0.2)	(6.3)	-
sign	21.0	-	1.3	-	-	1.1	-	0.7	0.2	0.1	-	1.6	**54.8**	0.5	13.2	2.3	1.9	0.9	-	0.5	-
	(26.8)	(0.2)	(3.1)	(0.5)	-	(2.2)	(0.7)	(1.0)	(0.6)	(4.1)	(0.6)	(1.7)	(44.6)	(0.2)	(9.9)	(0.1)	(1.9)	(0.3)	-	(1.2)	(0.4)
bird	5.5	9.3	14.5	3.1	7.2	5.8	0.6	7.8	-	3.5	3.0	-	1.1	**23.0**	-	3.8	8.3	0.1	1.1	0.4	1.8
	(18.0)	(5.5)	(15.8)	(4.8)	(11.3)	(5.0)	(1.3)	(5.7)	(0.6)	(5.9)	(0.2)	-	(1.7)	(11.5)	(0.1)	(1.0)	(5.8)	(0.5)	(2.2)	(2.0)	(0.9)
book	3.1	0.1	0.2	0.1	-	-	-	0.4	0.5	-	-	6.7	0.1	-	**82.5**	0.6	0.2	0.1	-	3.2	2.1
	(9.3)	(1.6)	(1.2)	(2.3)	(0.2)	(0.2)	(0.9)	(0.8)	(0.8)	(2.2)	(0.3)	(4.8)	(1.4)	(0.1)	(67.4)	(0.9)	(1.4)	(0.3)	(0.4)	(2.9)	(0.4)
chair	28.0	6.0	4.8	6.8	-	0.1	0.1	0.3	-	1.0	0.3	2.1	-	1.6	2.0	**39.6**	5.6	0.9	-	0.2	0.6
	(39.2)	(7.5)	(8.5)	(8.1)	(0.1)	(0.1)	(2.9)	(1.1)	(0.6)	(2.1)	(1.4)	(2.0)	(0.4)	(0.4)	(2.6)	(16.5)	(4.2)	(0.4)	(0.2)	(0.9)	(0.7)
road	5.1	0.6	0.3	-	0.3	1.5	0.5	9.1	0.4	2.2	0.6	0.1	-	0.1	-	0.3	**77.0**	0.6	0.5	1.0	-
	(8.2)	(0.9)	(0.4)	(0.1)	(0.1)	(1.7)	(0.2)	(9.9)	(0.3)	(1.7)	(0.3)	-	(0.1)	-	(0.1)	(0.2)	(74.5)	(0.2)	(0.3)	(0.7)	-
cat	2.7	-	2.3	0.8	-	-	-	3.1	0.7	0.4	0.2	9.9	-	2.2	-	-	11.7	**60.4**	5.2	0.3	-
	(12.9)	(0.1)	(3.1)	(8.3)	-	(0.4)	(0.1)	(2.4)	(2.1)	(2.8)	(1.1)	(1.6)	-	(1.7)	(0.6)	-	(9.5)	(43.5)	(7.8)	(1.9)	(0.1)
dog	2.9	2.3	4.8	3.7	1.5	2.9	-	0.2	7.8	0.1	-	-	-	2.6	-	0.4	5.7	11.7	**49.6**	4.0	-
	(9.4)	(2.7)	(2.7)	(7.3)	(4.8)	(5.7)	(0.1)	(2.5)	(8.1)	(0.2)	(0.4)	-	-	(2.5)	-	(0.2)	(7.6)	(7.9)	(34.8)	(2.9)	-
body	5.1	3.3	3.3	7.6	0.2	0.1	-	2.1	8.1	0.5	0.9	8.6	0.6	0.4	1.8	3.4	2.8	0.1	1.6	**49.5**	0.2
	(9.7)	(3.1)	(1.8)	(9.6)	(1.3)	(0.4)	(0.1)	(2.3)	(6.7)	(4.5)	(1.8)	(3.6)	(1.2)	(0.4)	(4.5)	(0.4)	(3.5)	(0.1)	(0.9)	(43.9)	(0.3)
boat	22.4	0.2	0.7	-	-	1.2	-	26.0	-	30.1	1.0	-	0.7	2.0	-	-	1.5	-	-	0.2	**14.0**
	(33.0)	(1.0)	(3.8)	(0.1)	-	(1.0)	(1.2)	(8.8)	(0.1)	(32.8)	(0.9)	-	(0.5)	(1.4)	(0.3)	(0.2)	(1.0)	(0.1)	-	(1.5)	(12.0)

MSRC Databases[9-class]

- 9-class
- 120 images trained, 120 images tested

MSRC Databases[21-class]

Algorithm	21-class MSRC Accuracy Min.	Avg.	Max.	Std.	9-class MSRC Accuracy Min.	Avg.	Max.	Std.
Shotton *et al.* [24]		72.2%*		n/a		-		-
Yang *et al.* [31]		75.1%*		n/a		-		-
Schroff *et al.* [22]		-		-		75.2%*		n/a
Baseline Logistic	61.8%	63.6%	65.4%	1.67%	77.5%	78.9%	79.8%	1.05%
Baseline CRF	68.3%	70.1%	72.0%	1.81%	81.2%	83.0%	84.4%	1.28%
Logistic + Rel. Loc.	73.5%	75.7%	77.4%	1.74%	87.6%	88.1%	88.9%	0.67%
CRF + Rel. Loc.	74.0%	**76.5%**	78.1%	1.82%	87.8%	**88.5%**	89.5%	0.82%

* For the other works, results are only reported on a single fold

Corel and Sowerly Databases

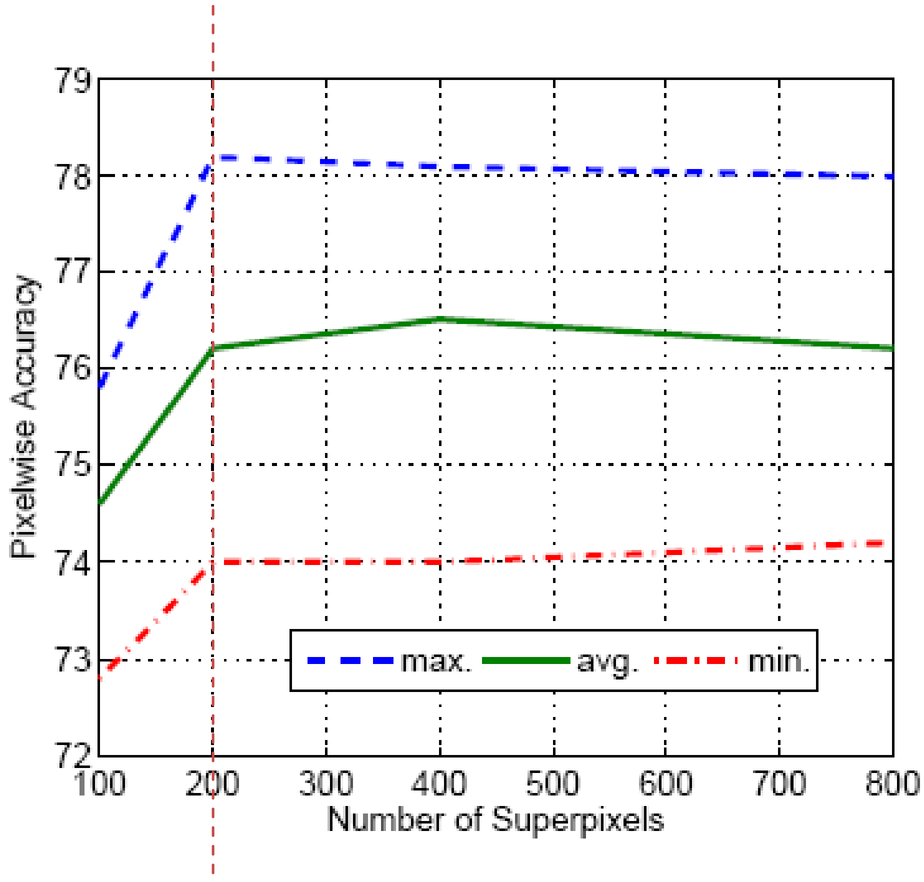
- 7-class
- Corel contain 100 images
- Sowerly contain 104 images
- Train both by 60 images, the rest are used for tested.
- 10 different random for train/test partitioning and report minimum, maximum and average for the test set.

Corel and Sowerly Databases

Algorithm	7-class Corel Accuracy Min.	Avg.	Max.	Std.	7-class Sowerby Accuracy Min.	Avg.	Max.	Std.
He *et al.* [9]		**80.0%***		n/a		**89.5%***		n/a
Kumar *et al.* [12]		-		-		89.3%*		n/a
Shotton *et al.* [24]		74.6%*		n/a		88.6%*		n/a
Yang *et al.* [31]		-		-		88.9%*		n/a
Baseline Logistic	68.2%	72.7%	76.8%	2.68%	84.7%	86.4%	88.0%	0.92%
Baseline CRF	69.6%	74.9%	78.5%	2.80%	84.9%	87.2%	88.6%	1.01%
Logistic + Rel. Loc.	70.7%	76.4%	81.6%	3.07%	84.9%	87.2%	88.5%	0.98%
CRF + Rel. Loc.	71.1%	77.3%	82.5%	3.15%	85.2%	87.5%	88.8%	0.98%

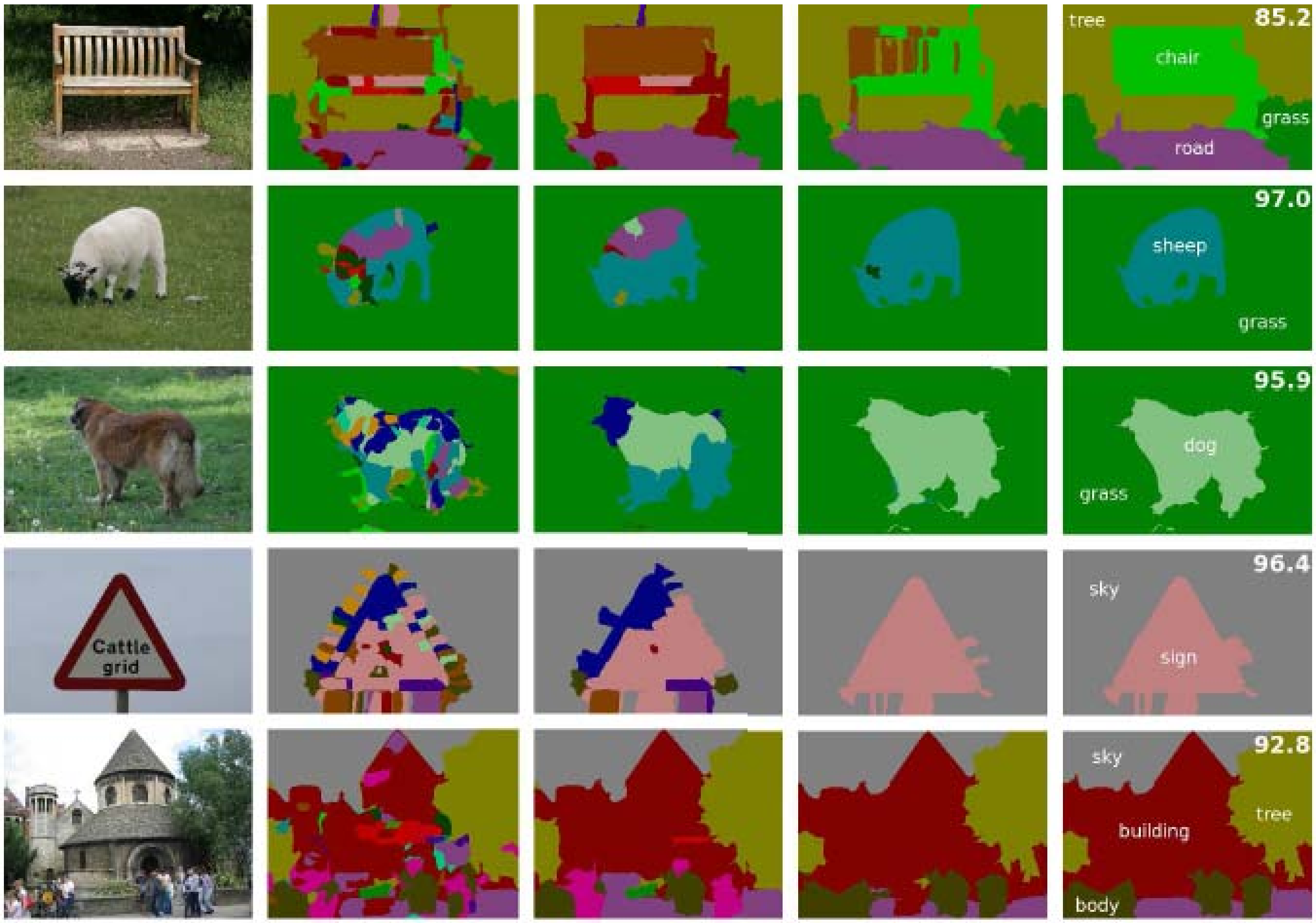
* For the other works, results are only reported on a single fold

Robustness to Over-segmentation

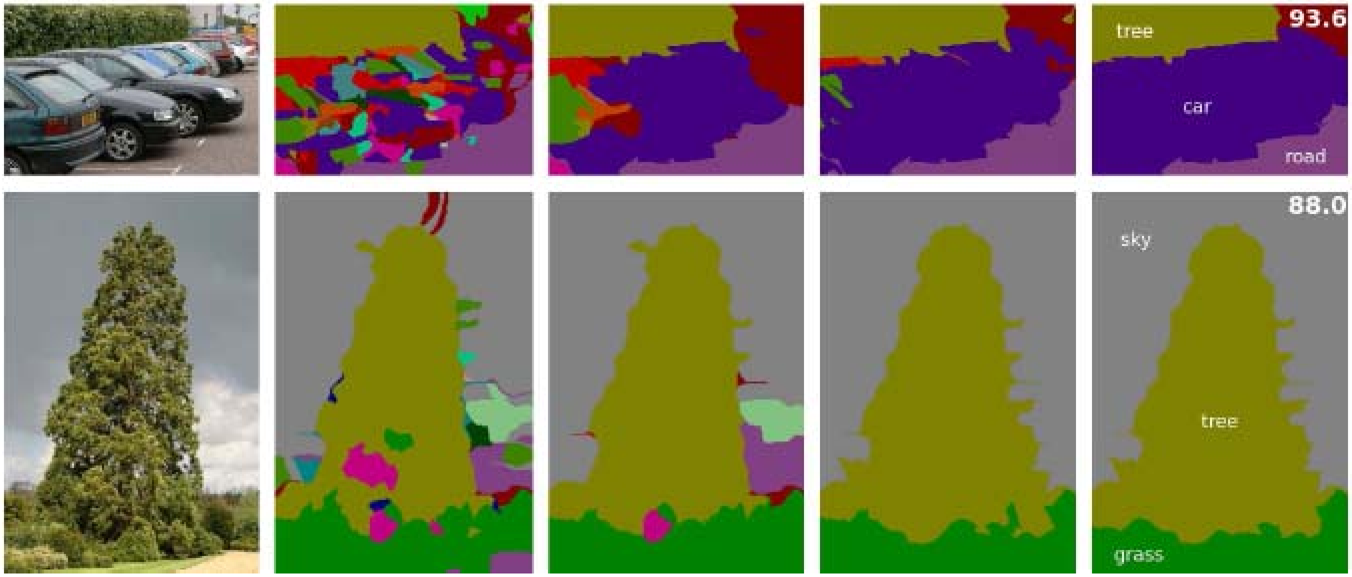


- Using different number of superpixels
- After approx 200 superpixels, the accurracy of algorithm insensitive to change in number of superpixels

Quality assessment

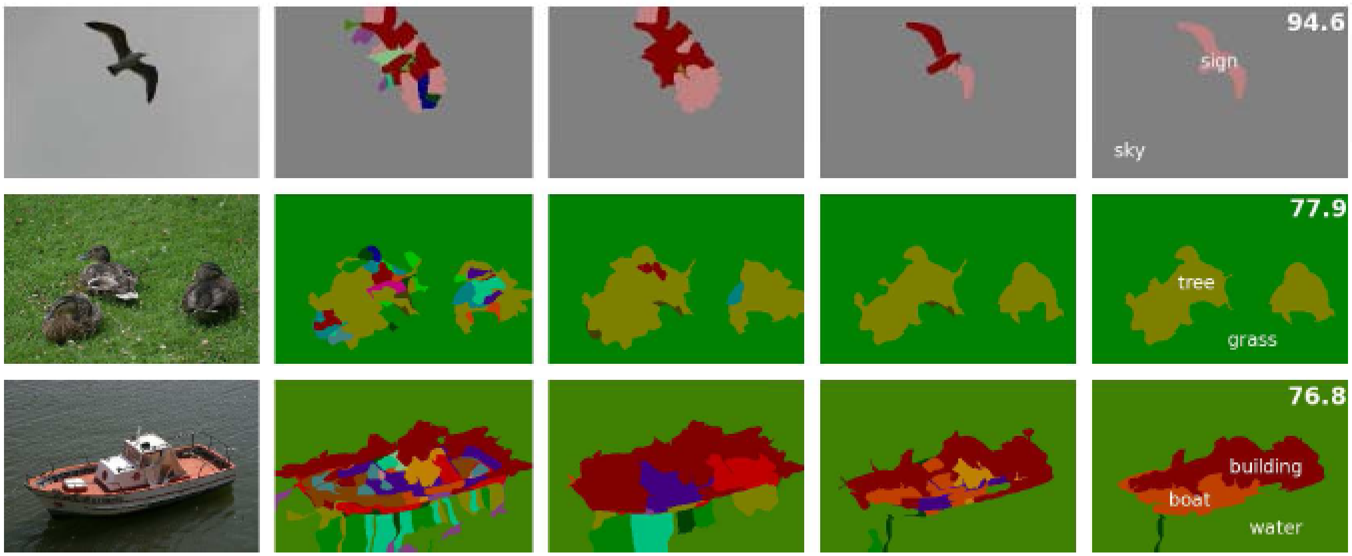


Quality assessment



(a) Image (b) Baseline Logistic (c) Baseline CRF (d) Rel. Loc. Logistic (e) Rel. Loc. CRF

Quality assessment



(a) Image (b) Baseline Logistic (c) Baseline CRF (d) Rel. Loc. Logistic (e) Rel. Loc. CRF

The End

- Thank you….