Selective Search for Object Recognition

Uijlings et al.

Schuyler Smith

Overview

- **Introduction**
- Object Recognition
- Selective Search
  - Similarity Metrics
- Results

Object Recognition

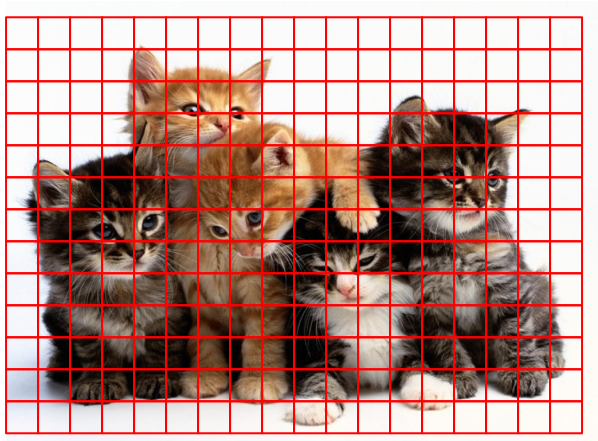
Goal:

Problem: Where do we look in the image for the object?

One Solution

Idea: Exhaustively search for objects.

Problem: Extremely slow, must process tens of thousands of candidate objects.

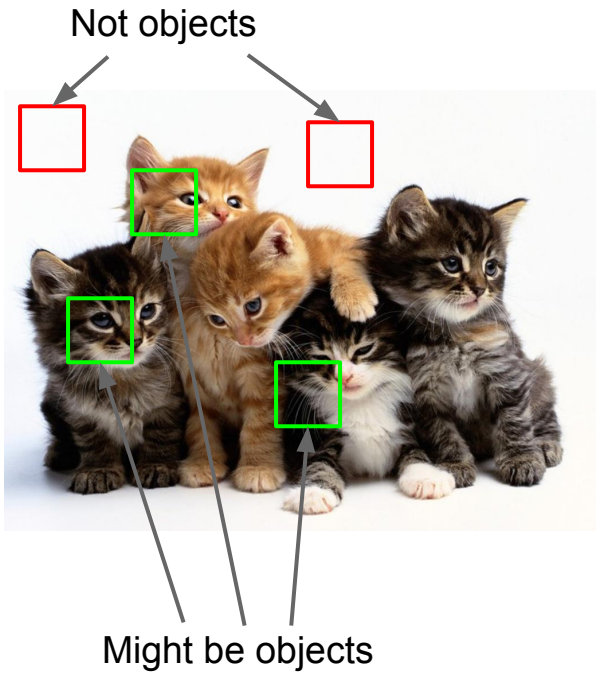


[N. Dalal and B. Triggs. “Histograms of oriented gradients for human detection.” In CVPR, 2005.]

One Solution

Idea: Running a scanning **detector** is cheaper than running a recognizer, so do that first.

1. Exhaustively search for candidate objects with a generic detector.
2. Run recognition algorithm only on candidate objects.

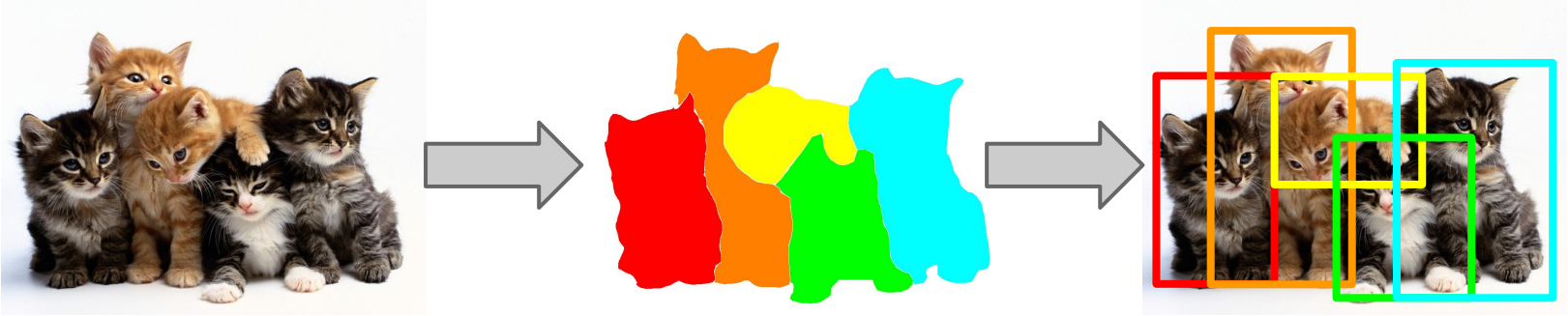


Problem: What about oddly-shaped objects? Will we need to scan with windows of many different shapes?

[B. Alexe, T. Deselaers, and V. Ferrari. “Measuring the objectness of image windows.” IEEE transactions on Pattern Analysis and Machine Intelligence, 2012.]

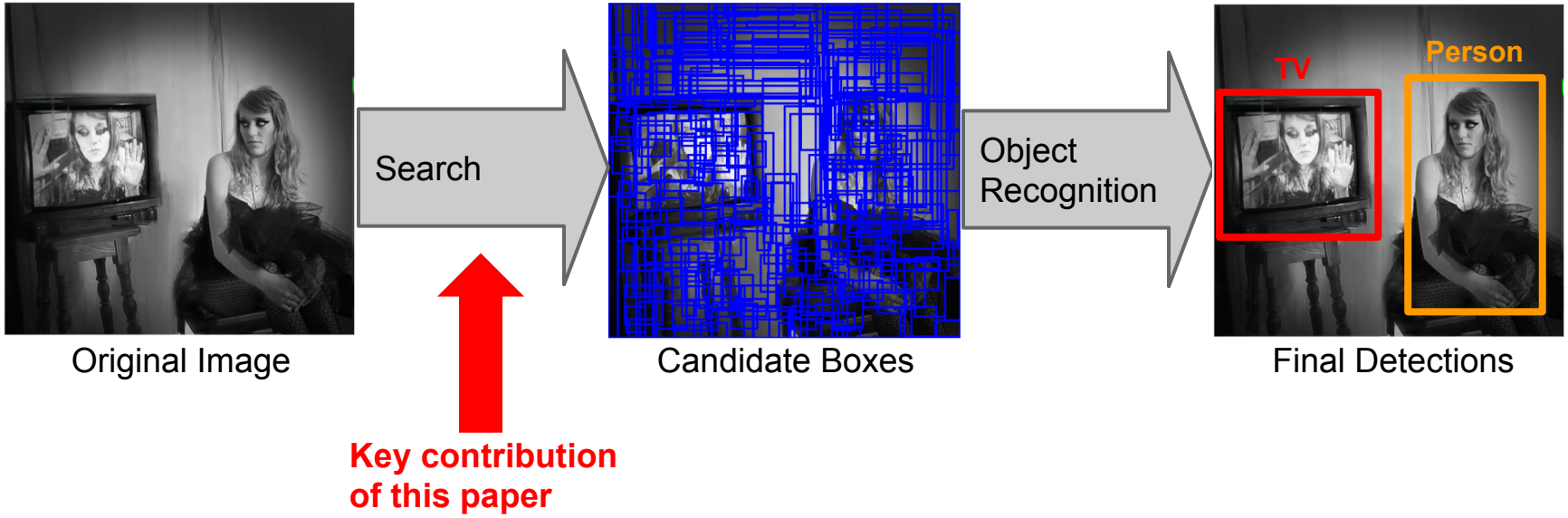
Segmentation

Idea: If we correctly segment the image before running object recognition, we can use our segmentations as candidate objects.



Advantages: Can be efficient, makes no assumptions about object sizes or shapes.

General Approach



Overview

- Introduction
- **Object Recognition**
- Selective Search
  - Similarity Metrics
- Results

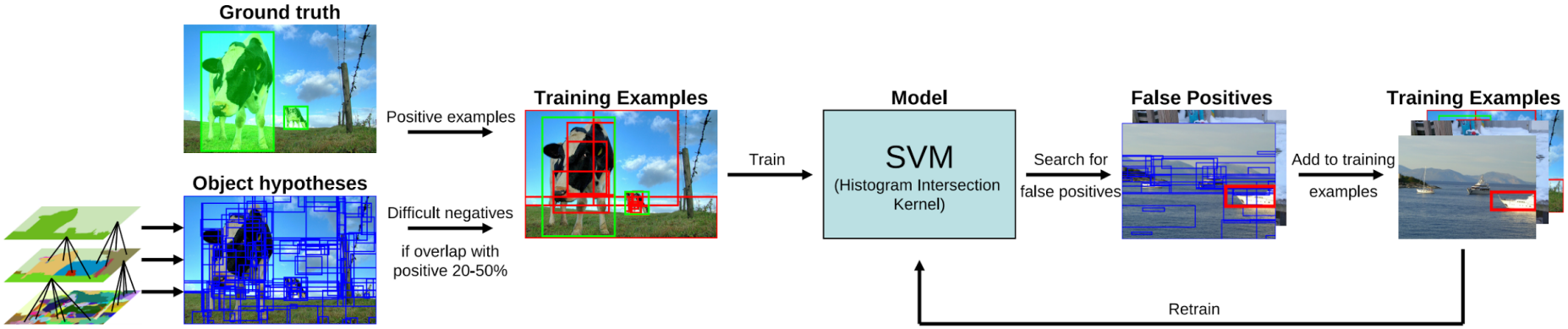
Recognition Algorithm

Basic approach:

- Bag of words model, with SIFT-based feature descriptors
- Spatial pyramid with four levels to encode some spatial information
- SVM for classification

Object Recognition

Training:

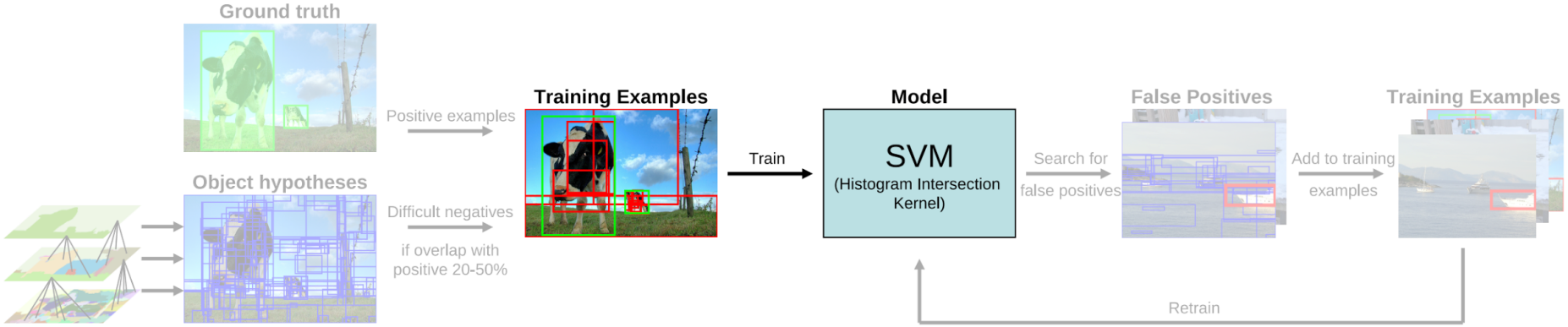


Object Recognition

Step 1: Train Initial Model

Positive Examples: From ground truth.

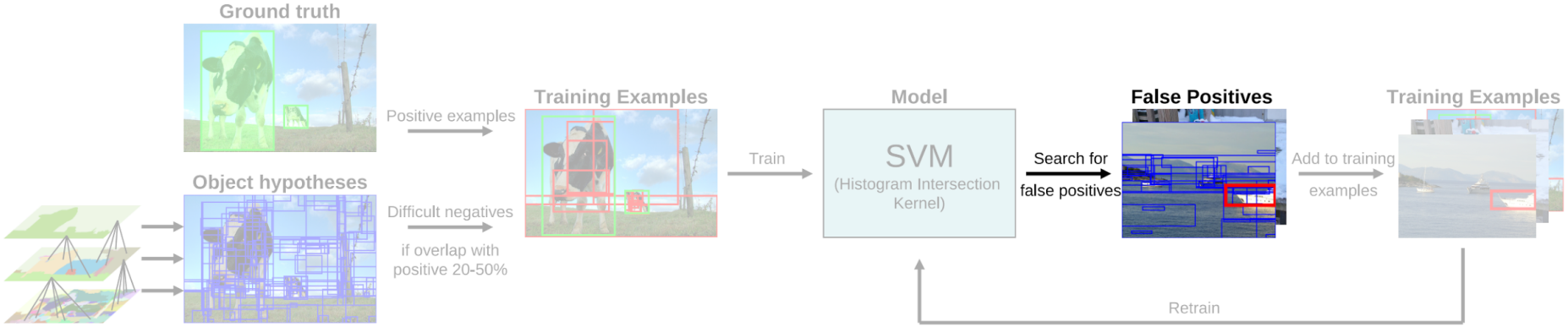
Negative Examples: Sample hypotheses that overlap 20-50% with ground truth.



Object Recognition

Step 2: Search for False Positives

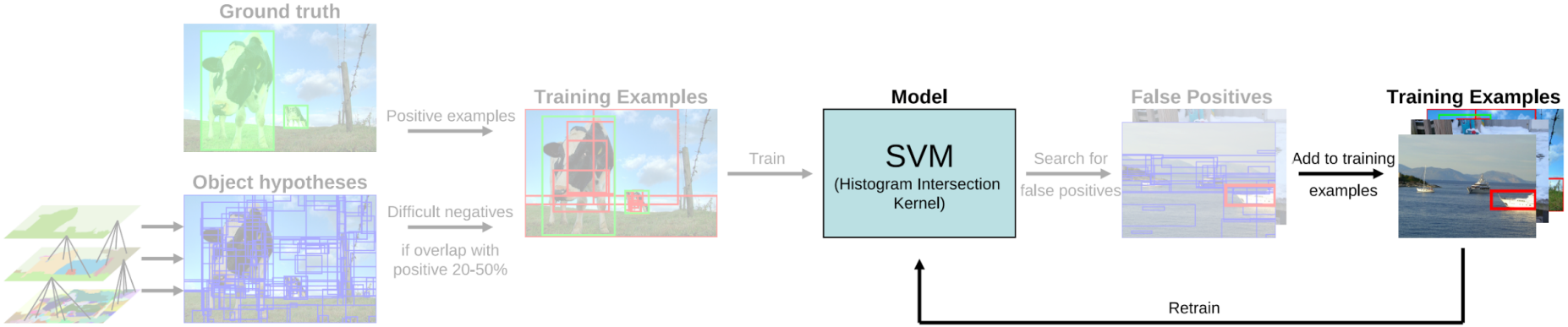
Run model on image and collect mistakes.



Object Recognition

Step 3: Retrain Model

Add **false positives** as new **negative examples**, retrain.



Overview

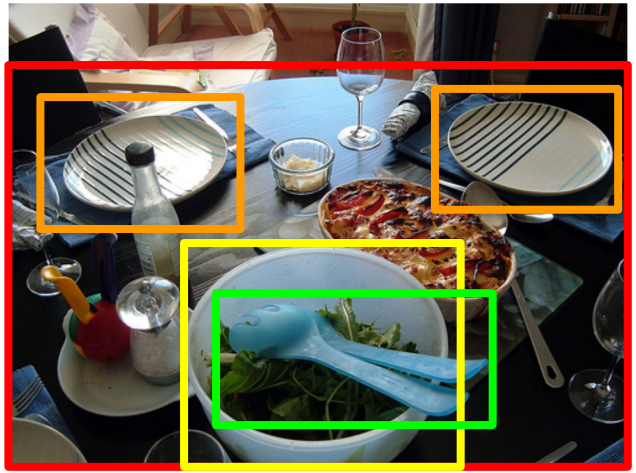
- Introduction
- Object Recognition
- **Selective Search**
  - Similarity Metrics
- Results

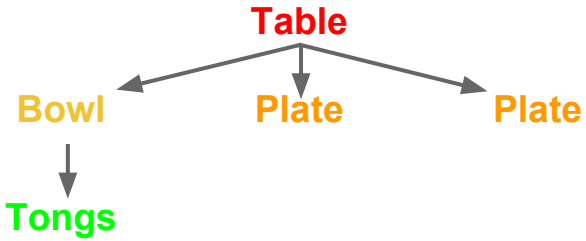
Hierarchical Image Representation

Images are actually 2D representations of a 3D world.

Objects can be on top of, behind, or parts of other objects.

We can encode this with an object/segment **hierarchy**.





Segmentation is Hard

As we saw in Project 1, it’s not always clear what separates an object.

Kittens are distinguishable by color (sort of), but not texture.

Chameleon is distinguishable by texture, but not color.

Segmentation is Hard

As we saw in Project 1, it’s not always clear what separates an object.

Wheels are part of the car, but not similar in color or texture.

How do we recognize that the head and body/sweater are the same “person”?

Selective Search

Goals:

1. Detect objects at any scale.
    a. Hierarchical algorithms are good at this.
2. Consider multiple grouping criteria.
    a. Detect differences in color, texture, brightness, etc.
3. Be fast.

Idea: Use bottom-up grouping of image regions to generate a hierarchy of small to large regions.

Selective Search

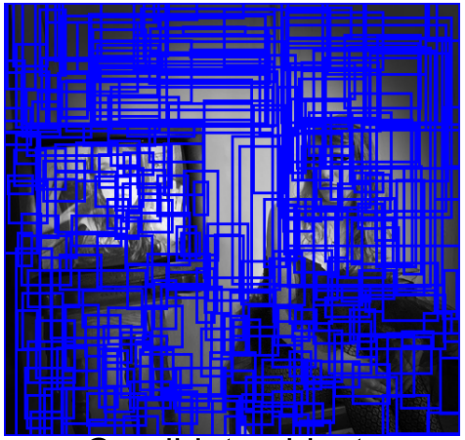
Step 1: Generate initial sub-segmentation

Goal: Generate many regions, each of which belongs to at most one object.

Using the method described by Felzenszwalb et al. from week 1 works well.

Input Image

Segmentation



Candidate objects

[P. F. Felzenszwalb and D. P. Huttenlocher. “Efficient Graph-Based Image Segmentation.” IJCV, 59:167–181, 2004.]

Selective Search

Step 2: Recursively combine similar regions into larger ones.

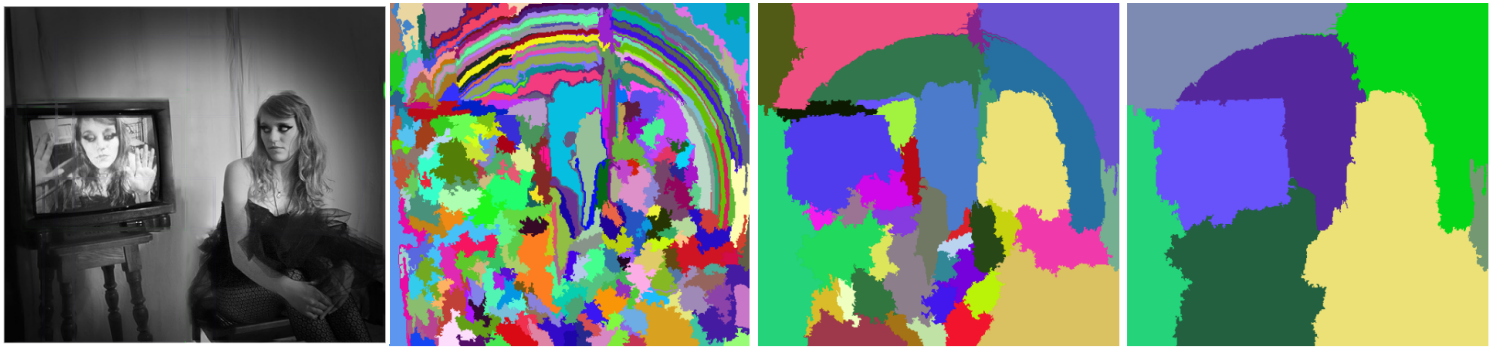
Greedy algorithm:

1. From set of regions, choose two that are most similar.
2. Combine them into a single, larger region.
3. Repeat until only one region remains.

This yields a hierarchy of successively larger regions, just like we want.

Selective Search

Step 2: Recursively combine similar regions into larger ones.



Input Image | Initial Segmentation | After some iterations | After more iterations

Selective Search

Step 3: Use the generated regions to produce candidate object locations.

Input Image

Overview

- Introduction
- Object Recognition
- Selective Search
  - **Similarity Metrics**
- Results

Similarity

What do we mean by “**similarity**”?

Goals:

1. Use multiple grouping criteria.
2. Lead to a balanced hierarchy of small to large objects.
3. Be efficient to compute: should be able to quickly combine measurements in two regions.

Similarity

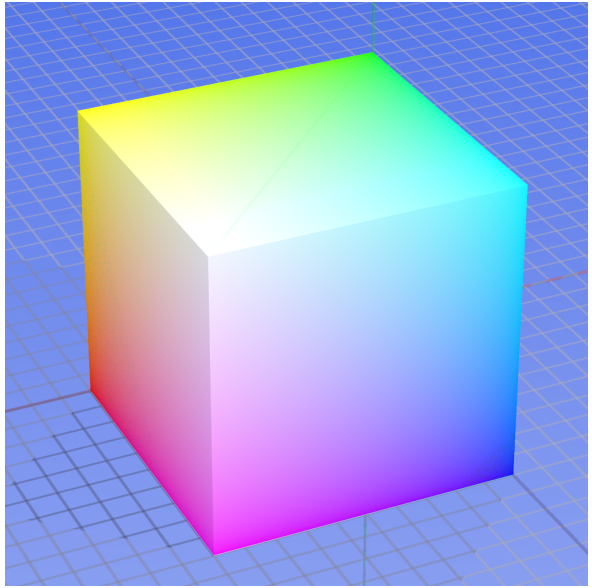
What do we mean by “**similarity**”?

Two-pronged approach:

1. Choose a **color space** that captures interesting things.
    a. Different color spaces have different invariants, and different responses to changes in color.
2. Choose a **similarity metric** for that space that captures everything we’re interested: color, texture, size, and shape.

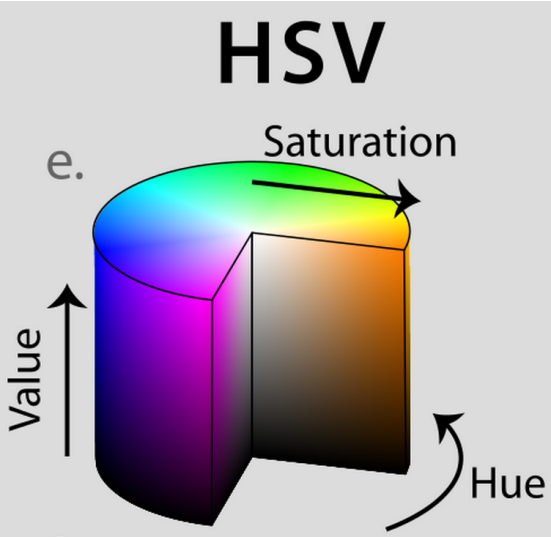
Similarity

RGB (red, green, blue) is a good baseline, but changes in illumination (shadows, light intensity) affect all three channels.



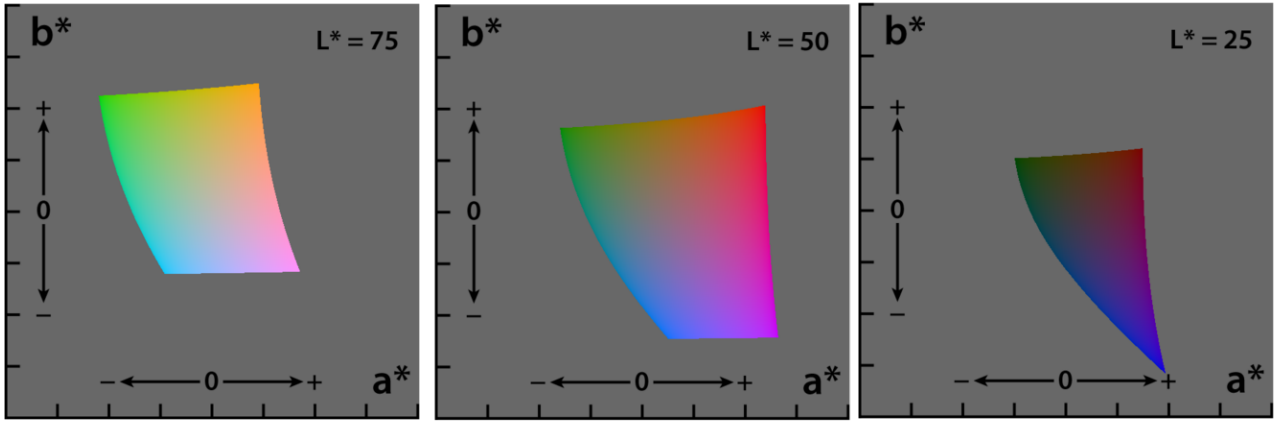
Similarity

HSV (hue, saturation, value) encodes color information in the hue channel, which is invariant to changes in lighting. Additionally, saturation is insensitive to shadows, and value is insensitive to brightness changes.



Similarity

Lab uses a lightness channel and two color channels (a and b). It's calibrated to be *perceptually uniform*. Like HSV, it's also somewhat invariant to changes in brightness and shadow.



Similarity

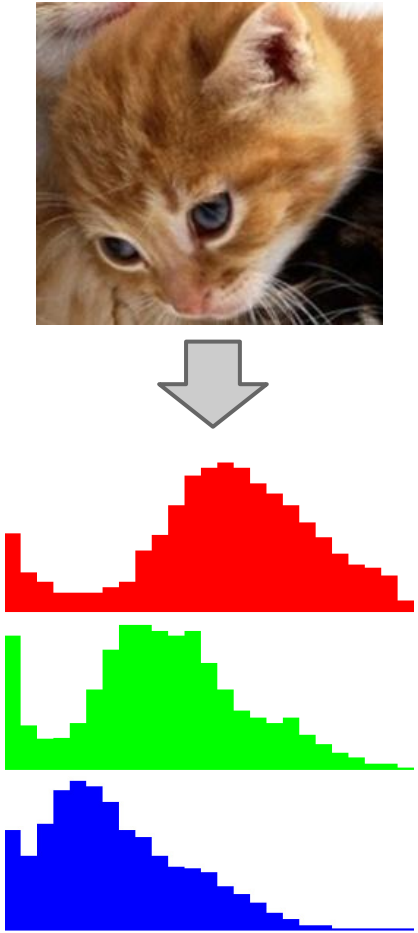
Similarity Measures: Color Similarity

Create a color histogram *C* for each channel in region *r*.

In the paper, 25 bins were used, for 75 total dimensions.

We can measure similarity with histogram intersection:

$$s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$

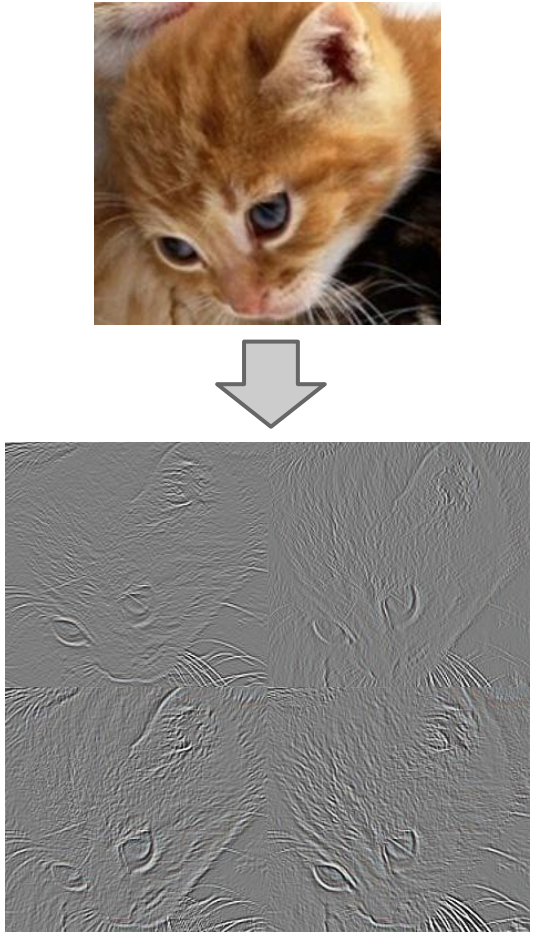


Similarity

Similarity Measures: Texture Similarity

Can measure textures with a HOG-like feature:

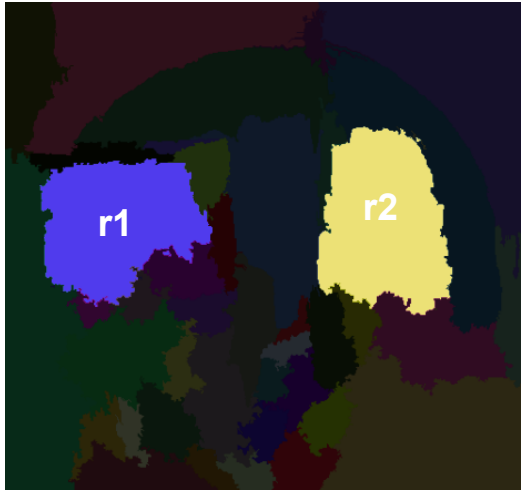
1. Extract gaussian derivatives of the image in 8 directions and for each channel.
2. Construct a 10-bin histogram for each, resulting in a 240-dimensional descriptor.



Similarity

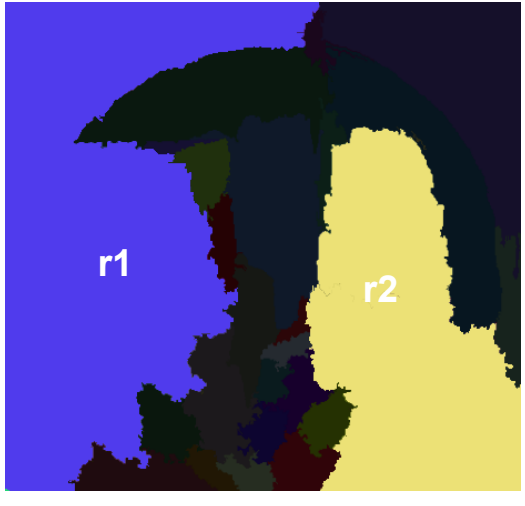
Similarity Measures: Size Similarity

We want small regions to merge into larger ones, to create a balanced hierarchy.



Solution: Add a size component to our similarity metric, that ensures small regions are more similar to each other.

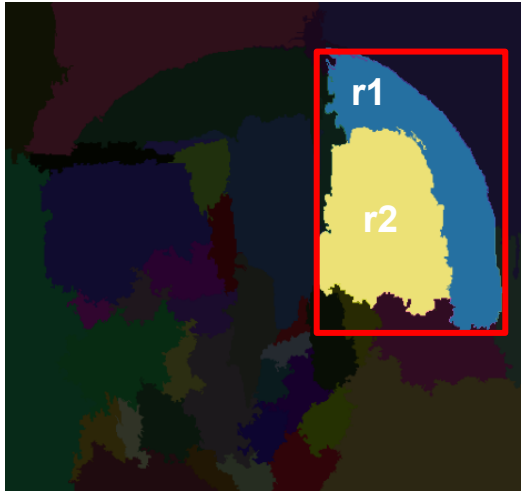
$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(\text{r}_\text{j})}{\text{size}(im)}$$



Similarity

Similarity Measures: Shape Compatibility

We also want our merged regions to be cohesive, so we can add a measure of how well two regions “fit together”.



$$fill(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_i)}{\text{size}(im)}$$

Similarity

Final similarity metric:

We measure the similarity between two patches as a **linear combination** of the four given metrics:

$$
\begin{aligned}
s(r_i, r_j) \quad = \quad & a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + \\
& a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j),
\end{aligned}
$$

Then, we can create a diverse collection of region-merging strategies by considering different weighted combinations in different color spaces.

Overview

- Introduction
- Object Recognition
- Selective Search
  - Similarity Metrics
- **Results**

Evaluation

Measuring box quality:

We introduce a metric called **Average Best Overlap**:

$$\mathrm{ABO} = \frac{1}{|G^c|} \sum_{g_i^c \in G^c} \max_{l_j \in L} \mathrm{Overlap}(g_i^c, l_j)$$

Overlap between ground truth and best selected box.

Average of “best overlaps” across all images.

Segmentation Results

Note that HSV, Lab, and rgl do noticeably better than RGB.

Texture on its own performs worse than the color, size, and fill similarity metrics.

Similarities	MABO	# box
C	0.635	356
T	0.581	303
S	0.640	466
F	0.634	449
C+T	0.635	346
C+S	0.660	383
C+F	0.660	389
T+S	0.650	406
T+F	0.638	400
S+F	0.638	449
C+T+S	0.662	377
C+T+F	0.659	381
C+S+F	0.674	401
T+S+F	0.655	427
C+T+S+F	0.676	395

Colours	MABO	# box
HSV	0.693	463
I	0.670	399
RGB	0.676	395
rgI	0.693	362
Lab	0.690	328
H	0.644	322
rgb	0.647	207
C	0.615	125

Thresholds	MABO	# box
50	0.676	395
100	0.671	239
150	0.668	168
250	0.647	102
500	0.585	46
1000	0.477	19

The best similarity measure overall uses all four metrics.

Segmentation Results

Combining strategies improves performance even more:

Version	Diversification Strategies	MABO	# win	# strategies	time (s)
Single Strategy	HSV C+T+S+F $k = 100$	0.693	362	1	0.71
Selective Search Fast	HSV, Lab C+T+S+F, T+S+F $k = 50, 100$	0.799	2147	8	3.79
Selective Search Quality	HSV, Lab, rgI, H, I C+T+S+F, T+S+F, F, S $k = 50, 100, 150, 300$	0.878	10,108	80	17.15

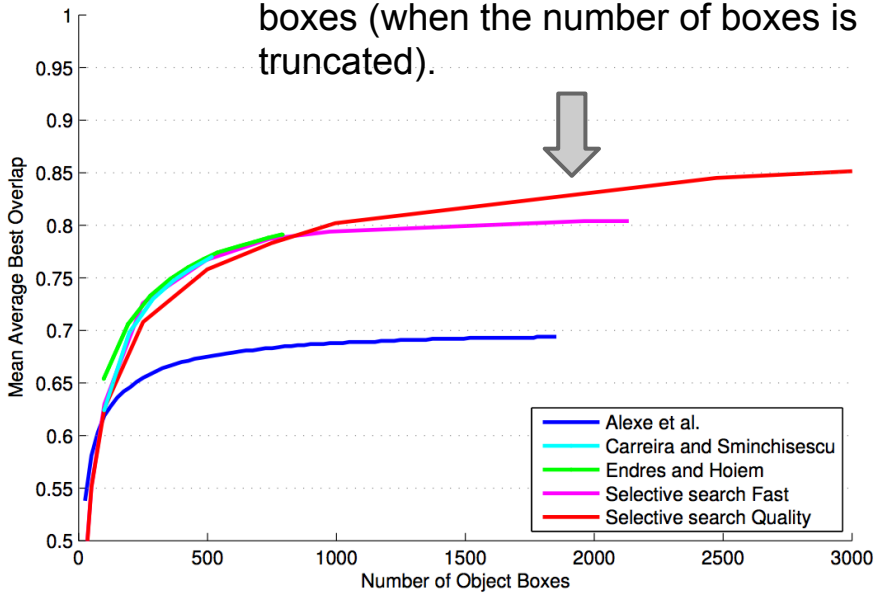
Using an ensemble greatly improves performance, at the cost of runtime (more candidate windows to check).

Segmentation Results

method	recall	MABO	# windows
Arbelaez *et al.* [3]	0.752	0.649 ± 0.193	418
Alexe *et al.* [2]	0.944	0.694 ± 0.111	1,853
Harzallah *et al.* [16]	0.830	-	200 per class
Carreira and Sminchisescu [4]	0.879	0.770 ± 0.084	517
Endres and Hoiem [9]	0.912	0.791 ± 0.082	790
Felzenszwalb *et al.* [12]	0.933	0.829 ± 0.052	100,352 per class
Vedaldi *et al.* [34]	0.940	-	10,000 per class
Single Strategy	0.840	0.690 ± 0.171	289
Selective search "Fast"	0.980	0.804 ± 0.046	2,134
Selective search "Quality"	0.991	0.879 ± 0.039	10,097

Excellent performance with fewer boxes than previous algorithms, which speeds up recognition.

"Quality" can outperform "Fast" even when returning the same number of boxes (when the number of boxes is truncated).



Segmentation Results

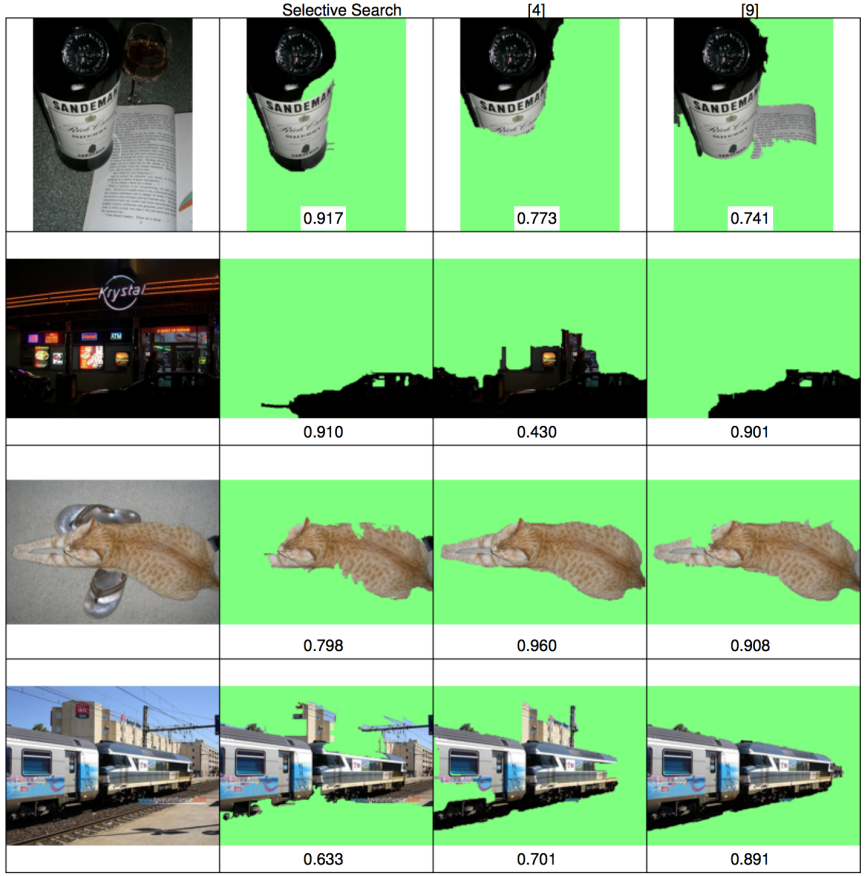
(a) Bike: 0.863

(b) Cow: 0.874

(c) Chair: 0.884

(d) Person: 0.882

Segmentation Results



[4] [9]

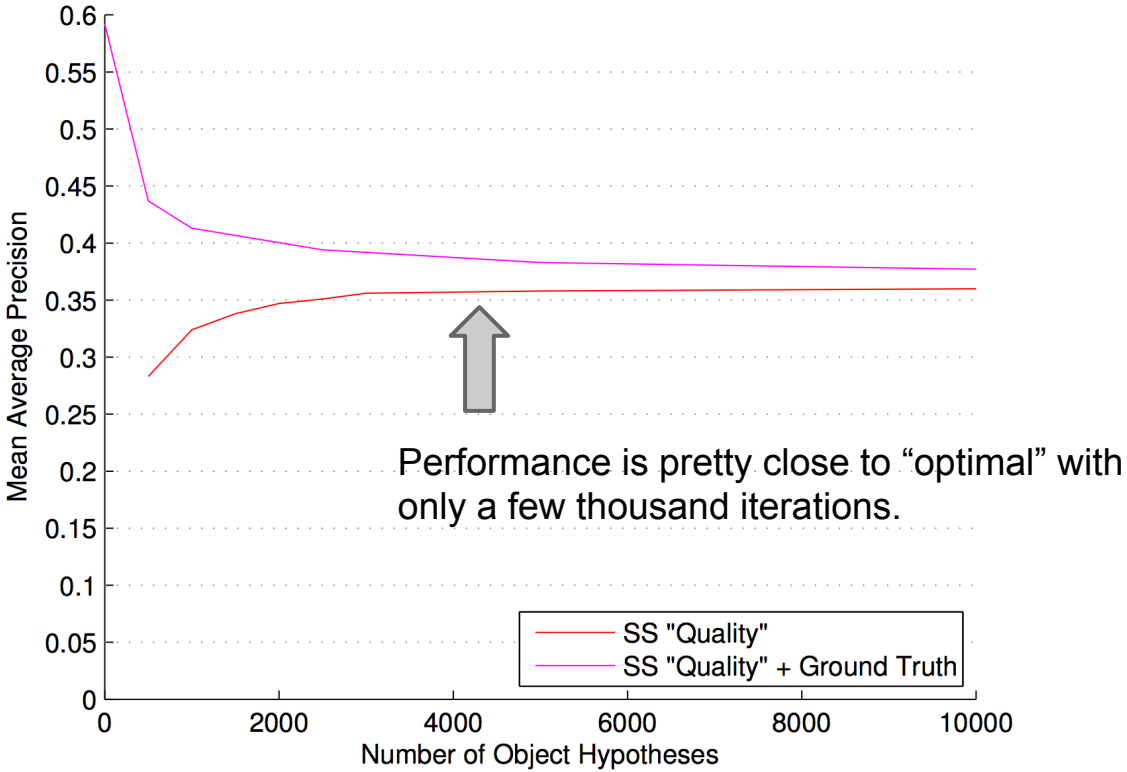
Recognition Results

Object recognition performance (average precision per class on Pascal VOC 2010):

System	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv
NLPR	.533	**.553**	**.192**	**.210**	.300	.544	.467	.412	**.200**	.315	.207	.303	.486	.553	.465	.102	.344	.265	**.503**	.403
MIT UCLA [38]	.542	.485	.157	.192	.292	**.555**	.435	.417	.169	.285	.267	.309	.483	.550	.417	.097	.358	.308	.472	.408
NUS	.491	.524	.178	.120	.306	.535	.328	.373	.177	.306	.277	.295	**.519**	**.563**	.442	.096	.148	.279	.495	.384
UoCTTI [12]	.524	.543	.130	.156	**.351**	.542	**.491**	.318	.155	.262	.135	.215	.454	.516	**.475**	.091	.351	.194	.466	.380
This paper	**.562**	.424	.153	.126	.218	.493	.368	**.461**	.129	**.321**	**.300**	**.365**	.435	.529	.329	**.153**	**.411**	**.318**	.470	**.448**

A couple of notable misses compared to other techniques, but best on about half, and best on average.

Effect of Location Quality



Summary

- We can speed up object recognition by applying a segmentation algorithm first, to help select object locations.

- Selective Search is a flexible hierarchical segmentation algorithm for this purpose.

- Performance is improved by using a diverse set of segmentation criteria.

- The performance of Selective Search and the complete object recognition pipeline are both very competitive with other appraoches.

Questions?