Type of Approaches

Different approaches tackle detection differently. They can roughly be categorized into three main types:

- Find **interest points**, followed by Hough voting
- **Sliding windows**: "slide" a box around image and classify each image crop inside a box (contains object or not?)
- Generate **region (object) proposals**, and classify each region $\leftarrow$ **We have looked at R-CNN a little bit. Today we'll focus on how to group pixels in super pixels, and regions. Once you have a region, you just attack it with a Neural Network.**

Segmentation
(in 15 minutes)

How Can We Get Semantics Quickly?

- Each image has about 1 to 4 **million pixels**.
- That's a lot.

How Can We Get Semantics Quickly?

- Each image has about 1 to 4 **million pixels**.
- That's a lot.
- A real robotics system needs to recognize things really fast if it wants to react to the world around it in real-time

How Can We Get Semantics Quickly?

- Each image has about 1 to 4 **million pixels**.
- That's a lot.
- A real robotics system needs to recognize things really fast if it wants to react to the world around it in real-time
- So the main question really is...

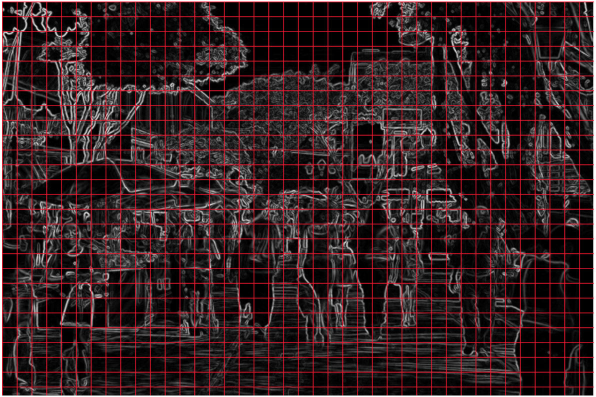
How Can We Get Semantics Quickly?

- Each image has about 1 to 4 **million pixels**.
- That's a lot.
- A real robotics system needs to recognize things really fast if it wants to react to the world around it in real-time
- So the main question really is...

How can we find this cookie before it's gone?

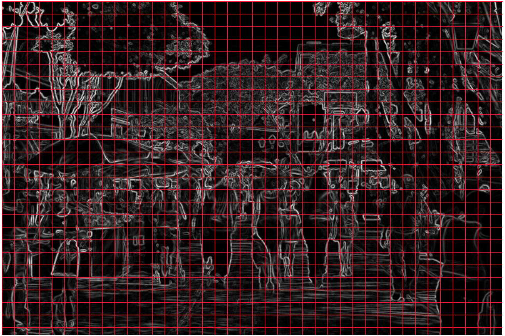
Remember HOG?

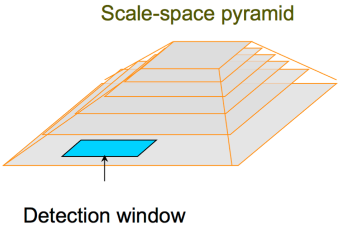
- HOG divides an image into cells. There is 64-times less cells than pixels. Two orders of magnitude less. Yet still, there is a lot of them.



Remember HOG?

- HOG divides an image into cells. There is 64-times less cells than pixels. Two orders of magnitude less. Yet still, there is a lot of them.
- And let's not forget we still need to run the detector, not only tones of locations, but also tones of scales.



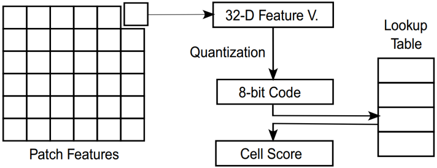


Remember DPM?

- I could tell you that with really cool tricks you can run DPM in real-time: *MA Sadeghi, D Forsyth, 30hz object detection with dpm v5, ECCV 2014*
- But that would ruin my point...
- (Also the AP of DPM is only 30%, while state-of-the-art is at 63%.)

Original HOG [Vondrick et al., MIT] | 8-bit Quantization | 4-bit Quantization

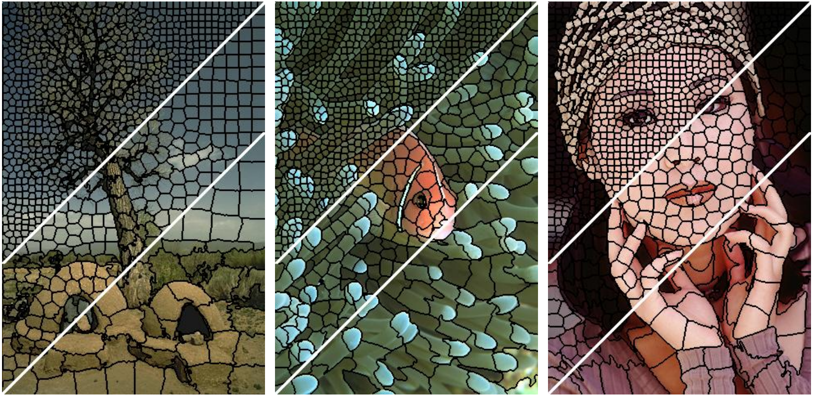
Vector Quantization For Speedup



- 32 multiplications and 31 additions vs. one table look-up
  - + Over 25x speedup
  - + Random-access score computation
  - - Introduction of error

Superpixels

- One of the ideas (which I like quite a lot) is to merge similar pixels into (way less) "superpixels".
- **First question** to ask: How does that help us?
- **Second question**: What properties should superpixels (or any regions) satisfy in order to be useful?

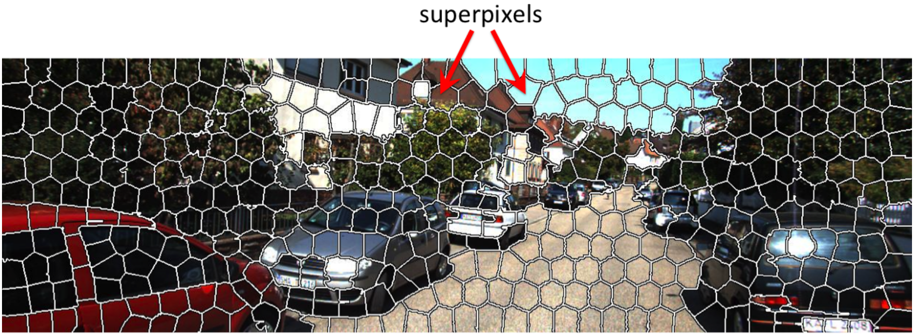


Why Are Superpixels Useful?

- Example 1: How can we find all *road* pixels in this image?

Why Are Superpixels Useful?

- Compute superpixels. A 4million pixel image is converted into only 500 superpixels. And now?

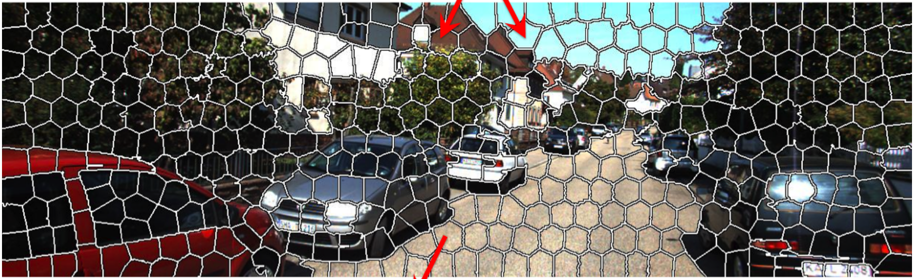


[Superpixels computed by Jian Yao, PhD student at UofT. Thanks Jian!]

Why Are Superpixels Useful?

- Compute features for each superpixel. (make sure to normalize them, e.g, to norm or max value 1; classifiers will work better)
- Different superpixels have different number of pixels. How can I compute my feature vector (has to have the same dimension for each superpixel)?

superpixels



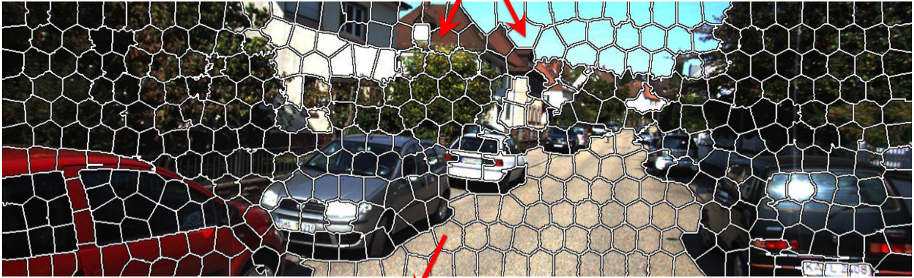
$$\mathbf{f} = [\text{color}, \text{gradients}, x, y, \ldots]^T$$

How?

Why Are Superpixels Useful?

- Average or better: **histogram** (typically the more dimensions that the feature vector has, the better the classifiers work). Histograms allow you to inflate a feature to multiple dimensions. And now?

superpixels

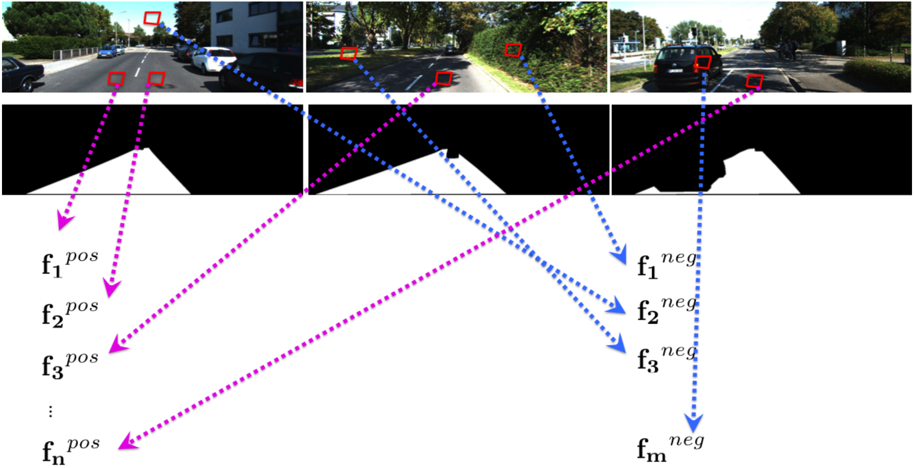


$$\mathbf{f} = [\text{color}, \text{gradients}, x, y, \ldots]^T$$

Can be: 1) average across pixels in superpixel, 2) a histogram, 3) a histogram of visual words

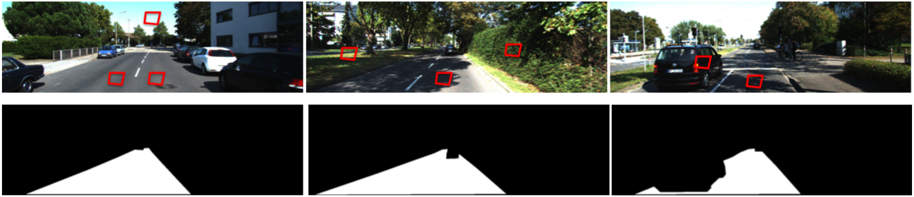
Why Are Superpixels Useful?

- Collect (randomly sample) a bunch of positive features (from regions annotated as *road*) and negative features (from regions of non-road).



Why Are Superpixels Useful?

- Train your favorite classifier. And at test time?



Train classifier

$\mathbf{f_1}^{pos}$

$\mathbf{f_2}^{pos}$

$\mathbf{f_3}^{pos}$

$\vdots$

$\mathbf{f_n}^{pos}$

$\mathbf{w}^T \cdot \mathbf{f} + b = 0$

$\mathbf{f_1}^{neg}$

$\mathbf{f_2}^{neg}$

$\mathbf{f_3}^{neg}$

$\mathbf{f_m}^{neg}$

Why Are Superpixels Useful?

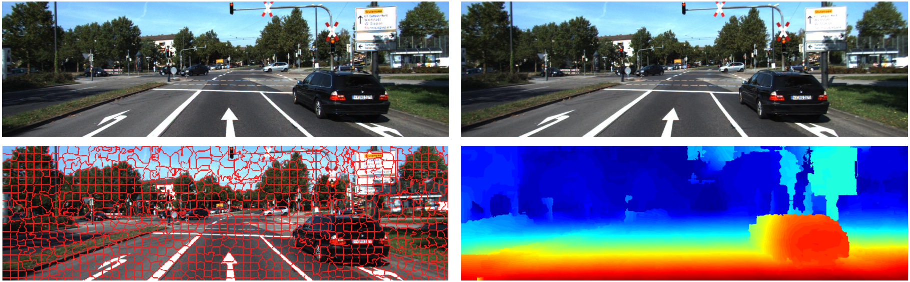
- Some problems can emerge, e.g., shadows, or grayish things on buildings that can confuse the classifier

Shadows look like anything else (car, dark tree, etc). Prediction will be bad.
Can we do something about it?

Why Are Superpixels Useful?

- We can make use of **depth** (e.g, road is typically not 1m above our eye level). Which brings us to:
- Example 2: We can use superpixels also for de-noising stereo results

(a) left image (b) right image

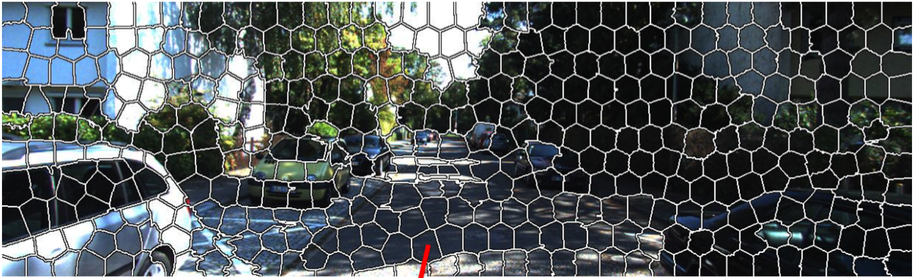


(c) superpixels (d) estimated disparity

Figure: `http://ttic.uchicago.edu/~dmcallester/SPS/index.html`

Why Are Superpixels Useful?

- (back to) Example 1: Now just simply augment superpixel features with 3D features.
- And hope for the best.



$$\mathbf{f} = [\text{color}, \text{gradients}, x, y, Y, \text{3d features (gradients in 3D?)}, \ldots]$$

Add depth informed features

Why Are Superpixels Useful?

- Example 3: You can also train superpixel classifiers to predict **surface labels** from a **monocular image**

Figure: http://web.engr.illinois.edu/~dhoiem/projects/context/

Why Are Superpixels Useful?

- Example 4: You can also train superpixel classifiers to predict a variety of **semantic classes**

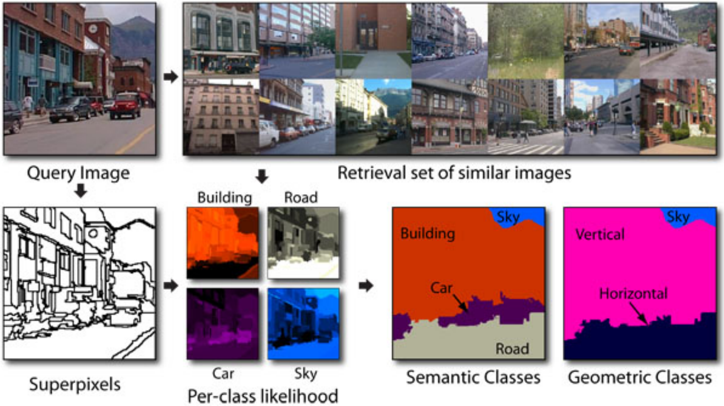


Figure: https://www.youtube.com/watch?v=UZ_NTYCFlHk

[Results from: J. Tighe and S. Lazebnik, SuperParsing: Scalable Nonparametric Image Parsing

Why Are Superpixels Useful?

- Remember the RGB-D image in Assignment 3?

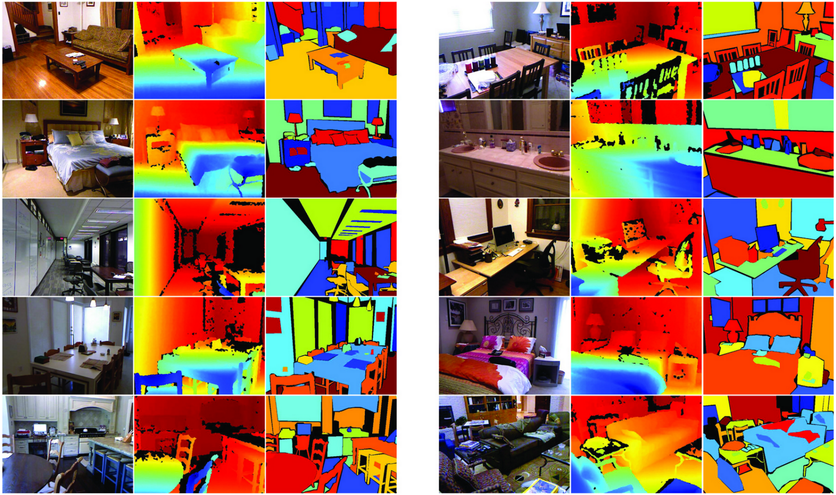


Figure: `http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html`

Why Are Superpixels Useful?

- Example 5: You can also train superpixel classifiers with RGB-D features to predict a variety of **semantic classes**

(a) Color Image (b) Depth (c) Contours (d) Semantic Segmentation

Figure: http://www.cs.berkeley.edu/~sgupta/pdf/GuptaArbelaezMalikCVPR13.pdf

Why Are Superpixels Useful?

- Example 6: And of course also to find our cookie.

[Superpixels computed by Jian Yao, PhD student at UofT. Thanks Jian!]

Why Are Superpixels Useful?

- Example 7: We can use them for tracking. And many more things.

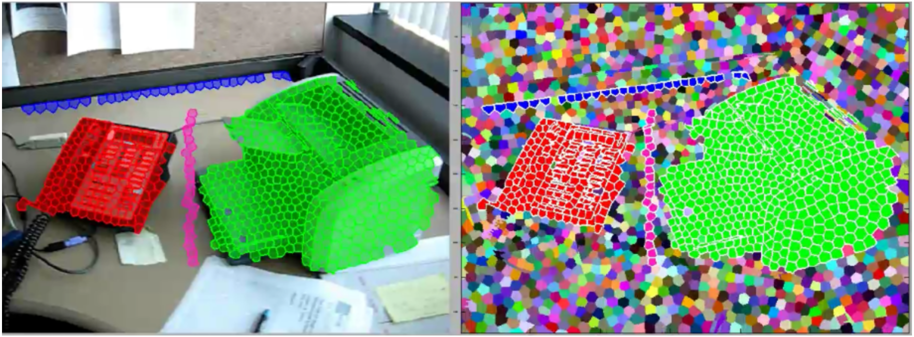


Figure:
http://groups.csail.mit.edu/vision/sli/projects.php?name=temporal_superpixels

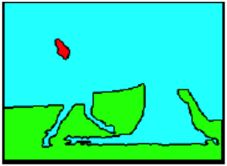
Segmentation

- Question 2: What properties should our segmentation have?

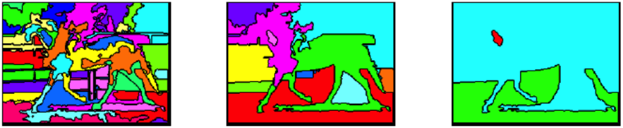
- Should be fast to compute
- Should not merge different objects (under segmentation)

Types of segmentations

Oversegmentation



Undersegmentation



Multiple Segmentations

Slide by D. Hoiem

Segmentation Algorithms

There are a lot of them out there

- Felzenswalb and Huttenlocher's Graph-based Segmentation
- SLIC
- SEEDS
- Shi and Malik's Normalized Cuts
- Malik's group: Probability of boundary (gPb)
- Grundmann et al: Hierarchical Graph-based Video Segmentation
- Chang et al., Temporal Superpixels

For most the code is available!

Segmentation Algorithms

There are a lot of them out there

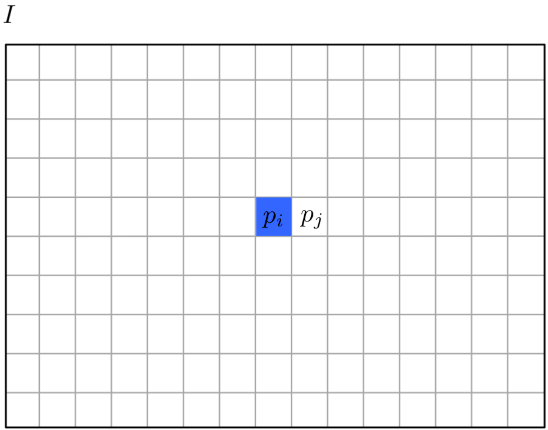
- Felzenswalb and Huttenlocher's Graph-based Segmentation $\rightarrow$ Let's quickly look at this one
- SLIC
- SEEDS
- Shi and Malik's Normalized Cuts
- Malik's group: Probability of boundary (gPb)
- Grundmann et al: Hierarchical Graph-based Video Segmentation
- Chang et al., Temporal Superpixels

For most the code is available!

Felzenswalb and Huttenlocher's Graph-based Segmentation:
`http://cs.brown.edu/~pff/segment/`

Felzenswalb & Huttenlocher

- Take an image.

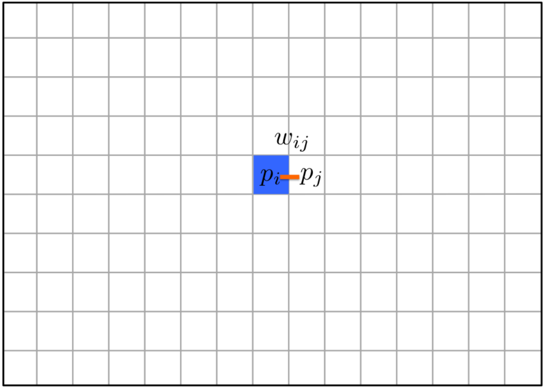


Felzenswalb & Huttenlocher

- Compute a weight between a pair of neighboring pixels. A weight reflects **dissimilarity**.

I

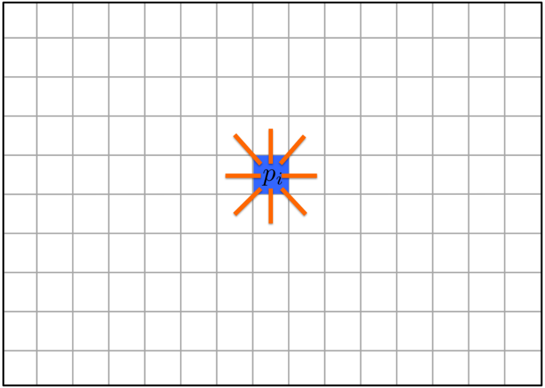
$$w_{ij} = |I(p_i) - I(p_j)|$$



Felzenswalb & Huttenlocher

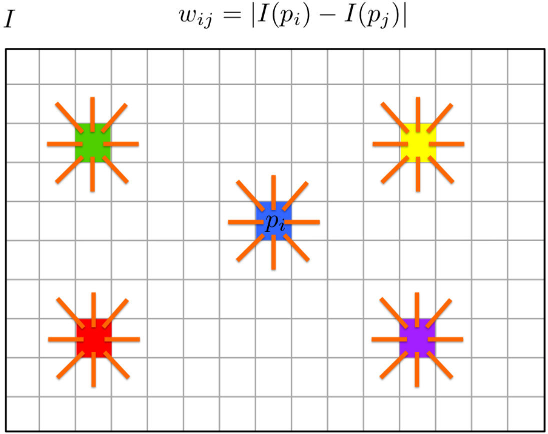
- Compute weights between a pixel and all 8 of its neighbors.

I $$w_{ij} = |I(p_i) - I(p_j)|$$



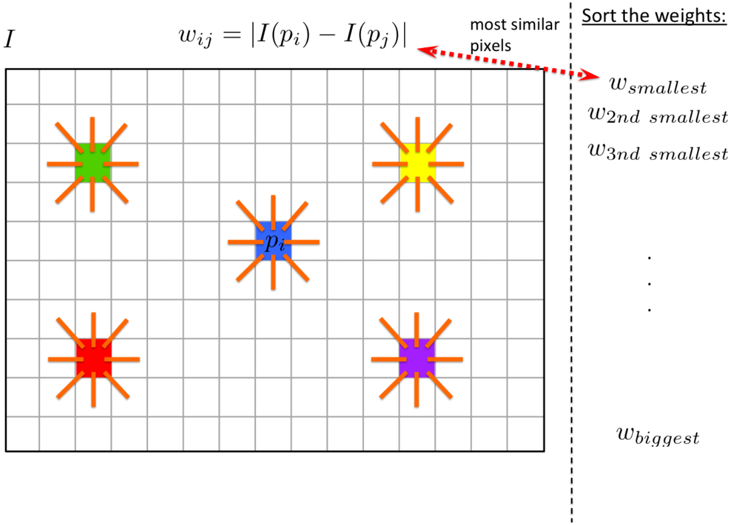
Felzenswalb & Huttenlocher

- And do that for **all pixels** in the image.



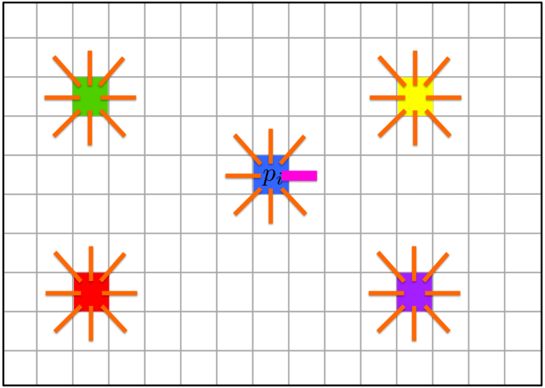
Felzenswalb & Huttenlocher

- Now sort all the weights in the image by non-decreasing values.



Felzenswalb & Huttenlocher

- Pick the weight from the top of the lest (most similar two neighboring pixels)



Sort the weights:

$w_{smallest}$

$w_{2nd\ smallest}$

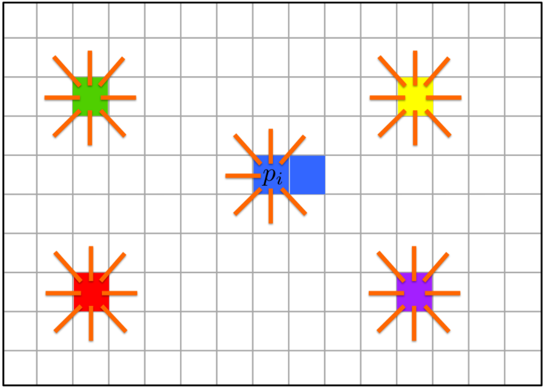
$w_{3nd\ smallest}$

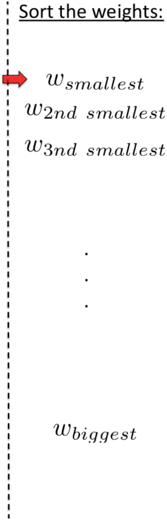
.
.
.

$w_{biggest}$

Felzenswalb & Huttenlocher

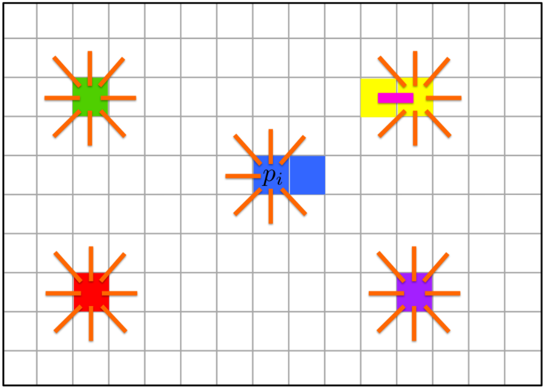
- If the weights is smaller than the internal dissimilarities (plus a threshold), **merge** the pixels





Felzenswalb & Huttenlocher

- Take the second weight in the list and do the same.



Sort the weights:

$w_{smallest}$

$w_{2nd\ smallest}$

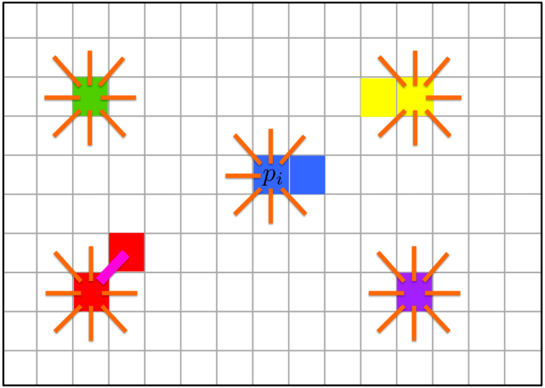
$w_{3nd\ smallest}$

.
.
.

$w_{biggest}$

Felzenswalb & Huttenlocher

- And the third one. Repeat until the end of list, or until the weight is higher than the dissimilarity between the pair of components.



Sort the weights:

$w_{smallest}$

$w_{2nd\ smallest}$

$w_{3nd\ smallest}$

.
.
.

$w_{biggest}$

Felzenswalb & Huttenlocher

- Result. The algorithm runs real-time.

Segmentation Algorithms

There are a lot of them out there

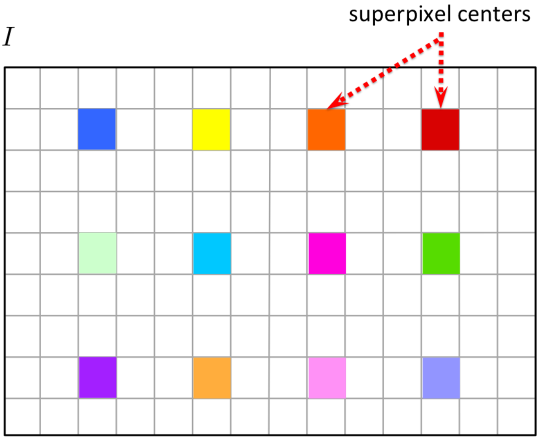
- Felzenswalb and Huttenlocher's Graph-based Segmentation
- SLIC $\rightarrow$ Let's quickly look at this one too
- SEEDS
- Shi and Malik's Normalized Cuts
- Malik's group: Probability of boundary (gPb)
- Grundmann et al: Hierarchical Graph-based Video Segmentation
- Chang et al., Temporal Superpixels

For most the code is available!

SLIC: `http://ivrg.epfl.ch/research/superpixels`

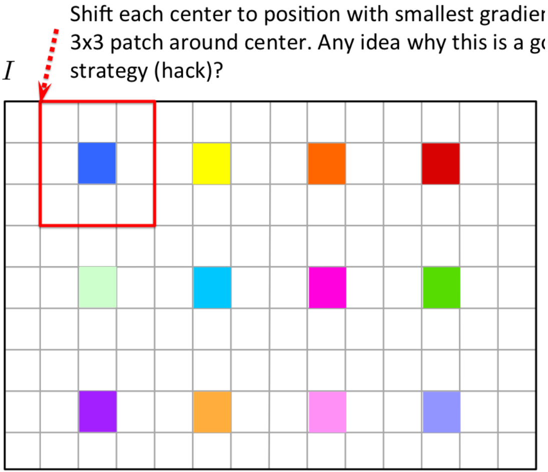
SLIC Superpixels

- Ask the user for the number of superpixels.
- Initialize the superpixel **centers** in a regular grid.



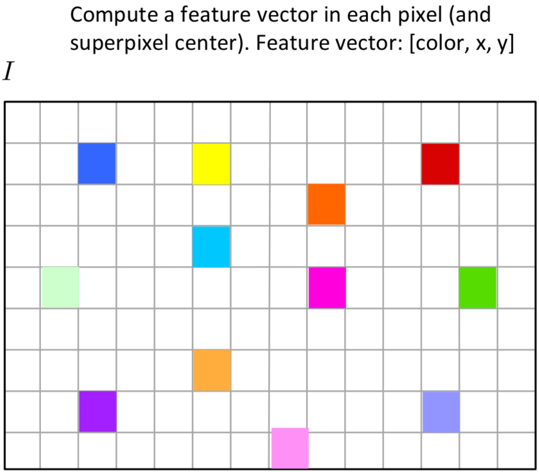
SLIC Superpixels

- Allow each center to shift in a 3×3 window to the location with the smallest gradient.



SLIC Superpixels

- Computer feature vectors for each pixel.

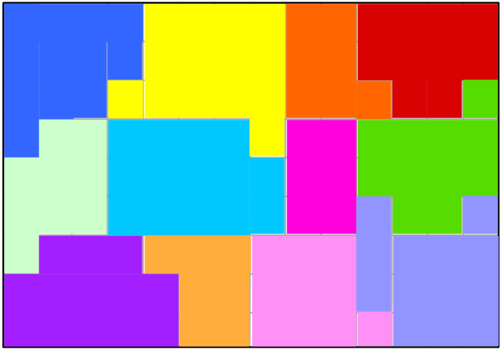


SLIC Superpixels

- Assign each pixel to the closest superpixel center.

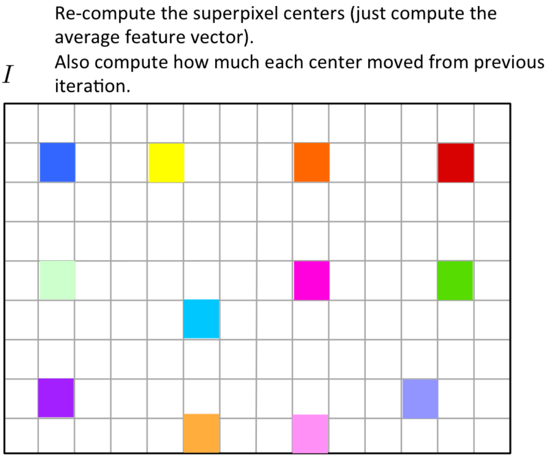
Assign all other pixels to the closest center, where ``closest'' is looking at a distance between the feature vectors

I



SLIC Superpixels

- Re-compute the centers (just average the features across pixels in each superpixel).



SLIC Superpixels

- Assign again. Iterate until the centers become stable (don't move from the previous iteration).

Results

- Results

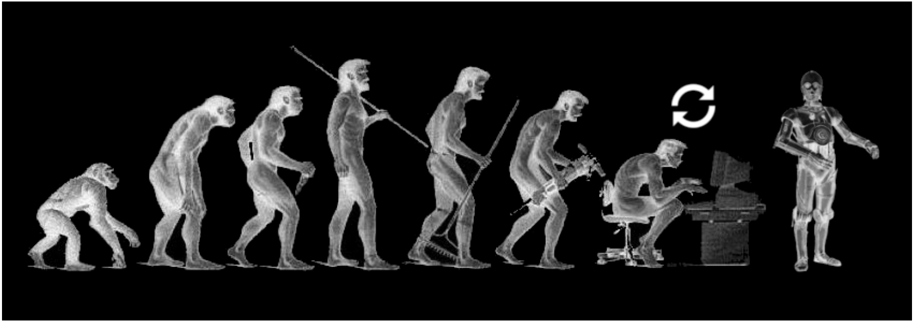
Results

- Results

Figure: https://www.youtube.com/watch?v=nUnAQUbeymQ

So... Singularity?

- So, you've seen some of the best of computer vision. What do you think, are we approaching singularity?



So... Singularity?

- Not just yet...

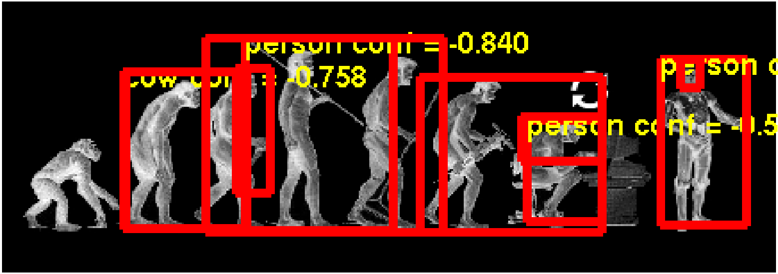


Figure: Results of RCNN

That's It For CSC420... But There Is Much More of Computer Vision For Those Interested!