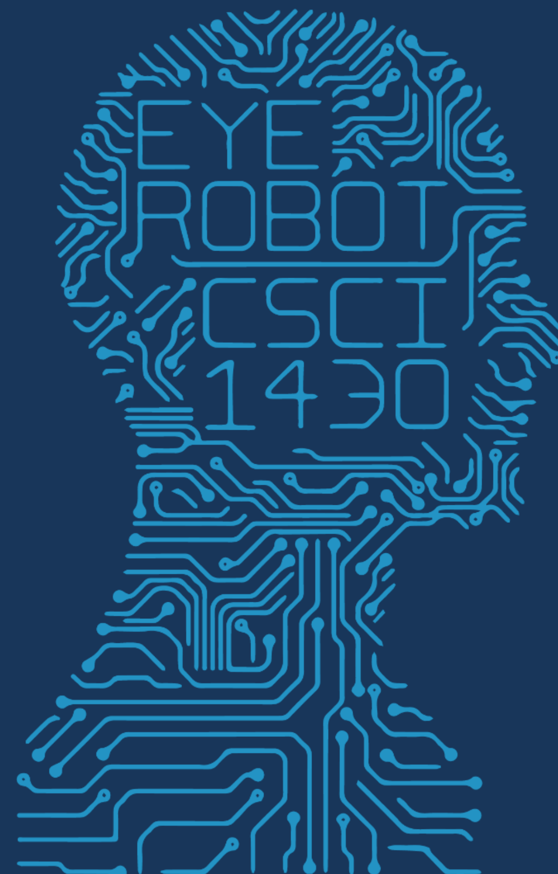
1950

Future Vision



2017 MWF 1pm 368

Computer Vision


note:
black & white

Object Detection Design challenges

- How to efficiently search for likely objects
  - Even simple models require searching hundreds of thousands of positions and scales
- Feature design and scoring
  - How should appearance be modeled?
    What features correspond to the object?
- How to deal with different viewpoints?
  - Often train different models for a few different viewpoints

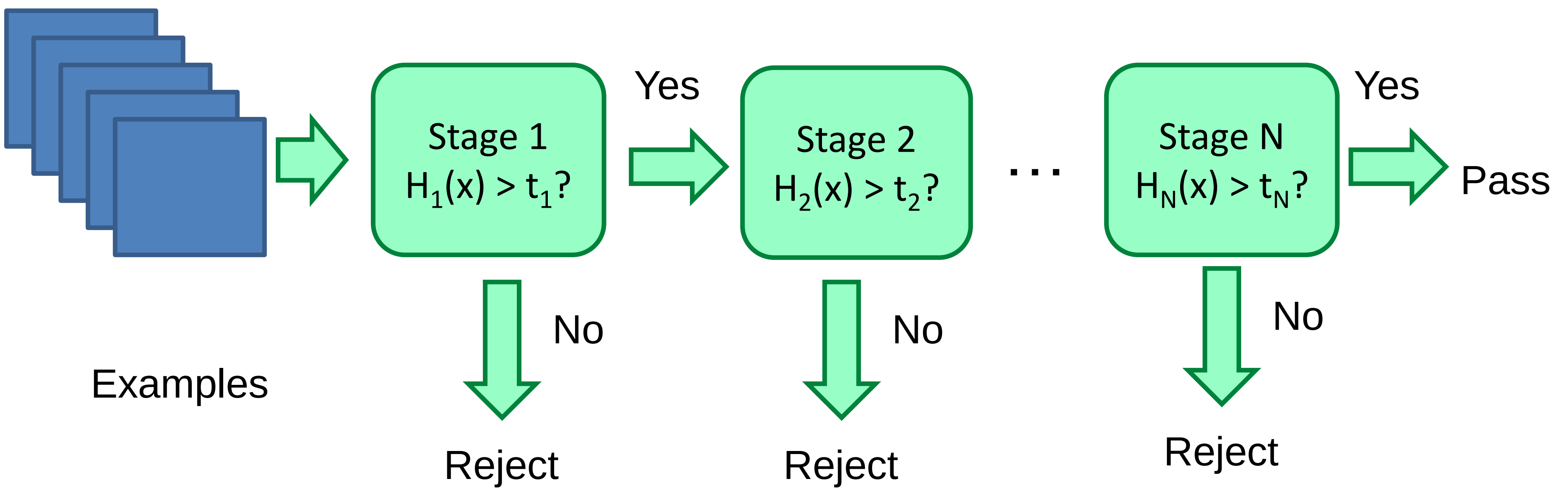
Recap: Viola-Jones sliding window detector

Fast detection through two mechanisms

- Quickly eliminate unlikely windows
- Use features that are fast to compute

Viola and Jones. Rapid Object Detection using a Boosted Cascade of Simple Features (2001).

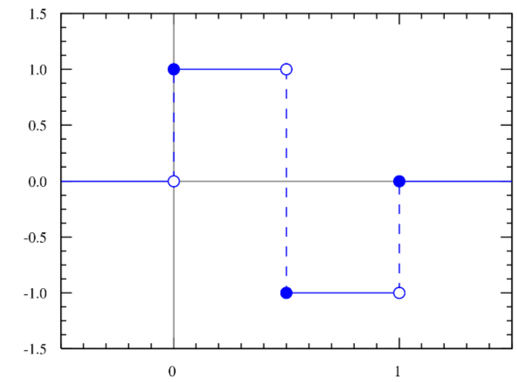
Cascade for Fast Detection



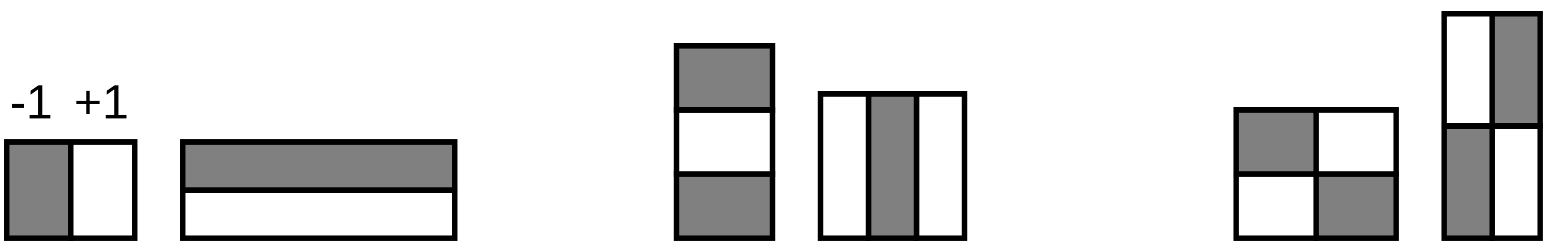
- Choose threshold for low false negative rate
- Fast classifiers early in cascade
- Slow classifiers later, but most examples don’t get there

Features that are fast to compute

Haar wavelet



- "Haar-like features"
  - Differences of sums of intensity
  - Thousands, computed at various positions and scales within detection window



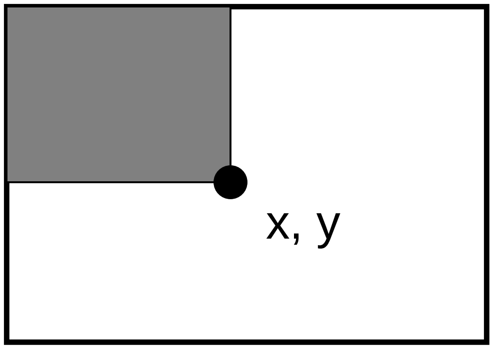
Two-rectangle features

Three-rectangle features

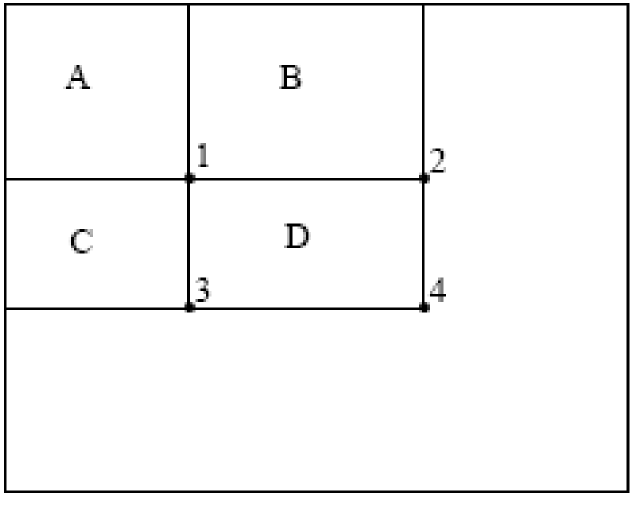
Etc.

Integral Images

- `ii = cumsum(cumsum(im, 1), 2)`



ii(x,y) = Sum of the values in the grey region



SUM within Rectangle D is
ii(4) - ii(2) - ii(3) + ii(1)

Feature selection with boosting

- Create a large pool of features (180K)
- Select discriminative features that work well together

Final strong learner

Weak learner

$$h(\mathbf{x}) = \operatorname{sign}\left(\sum_{j=1}^{M} \alpha_j h_j(\mathbf{x})\right)$$

window

Learner weight

  - "Weak learner" = feature + threshold + 'polarity'

value of rectangle feature

$$h_j(\mathbf{x}) = \begin{cases} -s_j & \text{if } f_j < \theta_j \\ s_j & \text{otherwise} \end{cases}$$

threshold

'polarity' $\longrightarrow$ $s_j \in \pm 1$

  - Choose weak learner that minimizes error on the weighted training set, then reweight

Adaboost pseudocode Szeliski p665

1. Input the positive and negative training examples along with their labels $\{(\boldsymbol{x}_i, y_i)\}$, where $y_i = 1$ for positive (face) examples and $y_i = -1$ for negative examples.

2. Initialize all the weights to $w_{i,1} \leftarrow \frac{1}{N}$, where N is the number of training examples. (Viola and Jones (2004) use a separate N_1 and N_2 for positive and negative examples.)

3. For each training stage $j = 1 \dots M$:

   (a) Renormalize the weights so that they sum up to 1 (divide them by their sum).

   (b) Select the best classifier $h_j(\boldsymbol{x}; f_j, \theta_j, s_j)$ by finding the one that minimizes the weighted classification error

   $$e_j = \sum_{i=0}^{N-1} w_{i,j} e_{i,j}, \tag{14.3}$$

   $$e_{i,j} = 1 - \delta(y_i, h_j(\boldsymbol{x}_i; f_j, \theta_j, s_j)). \tag{14.4}$$

   For any given f_j function, the optimal values of (θ_j, s_j) can be found in linear time using a variant of weighted median computation (Exercise 14.2).

   (c) Compute the modified error rate β_j and classifier weight α_j,

   $$\beta_j = \frac{e_j}{1 - e_j} \quad \text{and} \quad \alpha_j = -\log \beta_j. \tag{14.5}$$

   (d) Update the weights according to the classification errors $e_{i,j}$

   $$w_{i,j+1} \leftarrow w_{i,j} \beta_j^{1-e_{i,j}}, \tag{14.6}$$

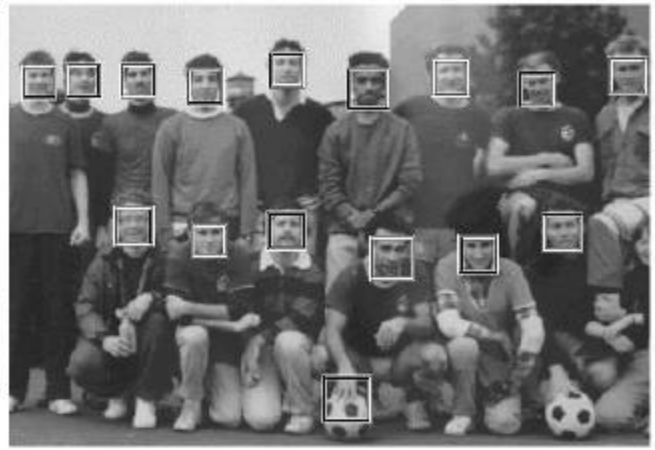
   i.e., downweight the training samples that were correctly classified in proportion to the overall classification error.

4. Set the final classifier to

$$h(\boldsymbol{x}) = \text{sign}\left[\sum_{j=0}^{m-1} \alpha_j h_j(\boldsymbol{x})\right]. \tag{14.7}$$

Viola Jones Results

Speed = 15 FPS (in 2001)


False detections / Detector	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset

- Viola-Jones has a very large space of simple weak ‘edge- or pattern-like’ classifiers.
- Learn importance/spatial layout of these edges for a particular class.

- Can we use a known layout?

Object Detection

- Overview
- Viola-Jones
- Dalal-Triggs
- Deformable models
- Deep learning

Person detection with HoG's & linear SVM's

- Histograms of Oriented Gradients for Human Detection, Navneet Dalal, Bill Triggs, International Conference on Computer Vision & Pattern Recognition - June 2005
- http://lear.inrialpes.fr/pubs/2005/DT05/

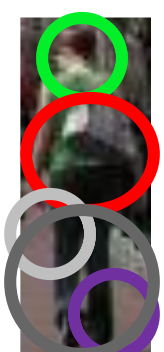
Statistical Template

Object model =

sum of scores of features at fixed positions

$$+3 +2 -2 -1 -2.5 = -0.5 \overset{?}{>} 7.5$$

Non-object



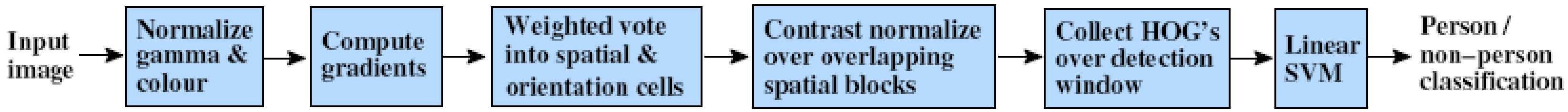
$$+4 +1 +0.5 +3 +0.5 = 10.5 \overset{?}{>} 7.5$$

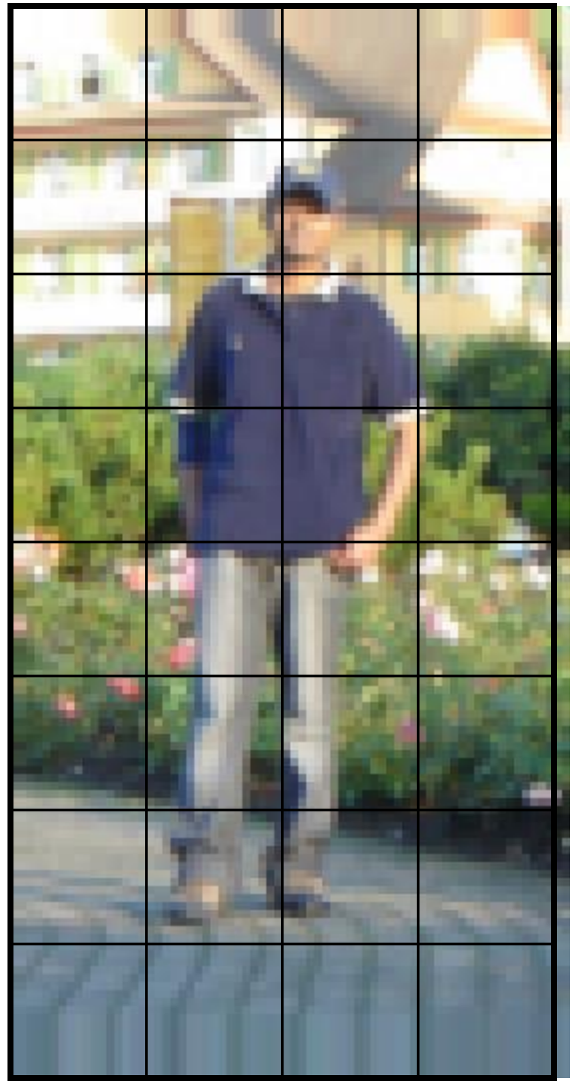
Object

Example: Dalal-Triggs pedestrian detector

1. Extract fixed-sized (64x128 pixel) window at each position and scale
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier
4. Perform non-maxima suppression to remove overlapping detections with lower scores

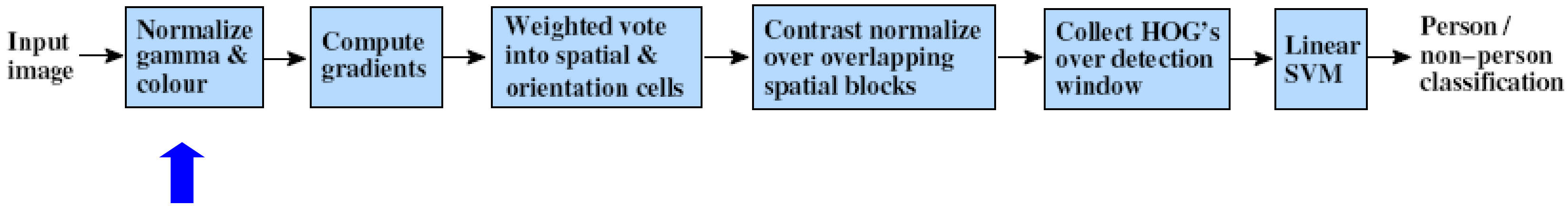
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

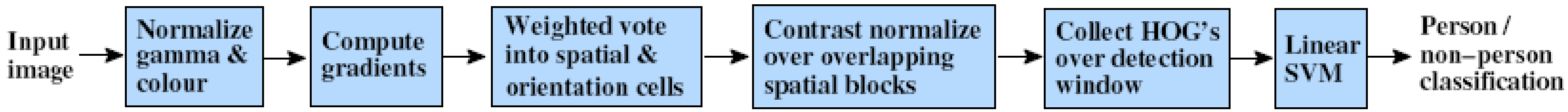


- Tested with
  - RGB
  - LAB
  - Grayscale

Slightly better performance vs. grayscale

- Gamma Normalization and Compression
  - Square root
  - Log

Very slightly better performance vs. no adjustment



Outperforms

-1	0	1

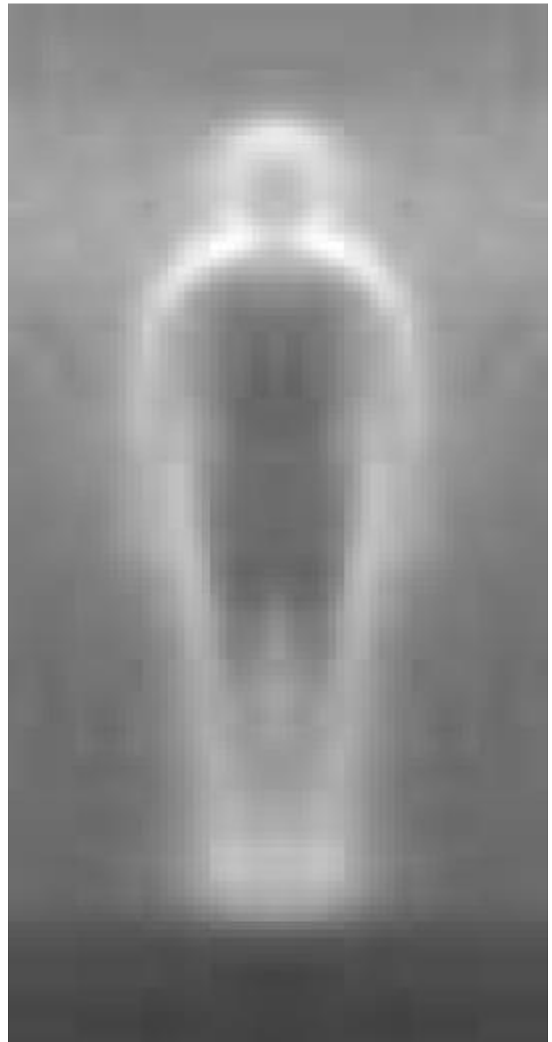
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

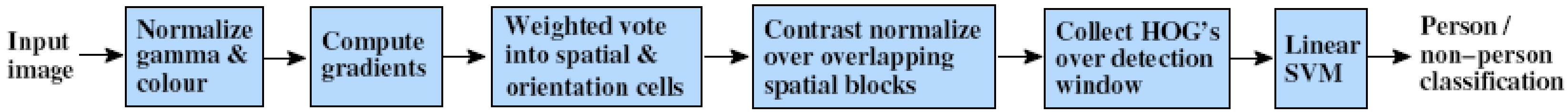
diagonal

-1	0	1
-2	0	2
-1	0	1

Sobel

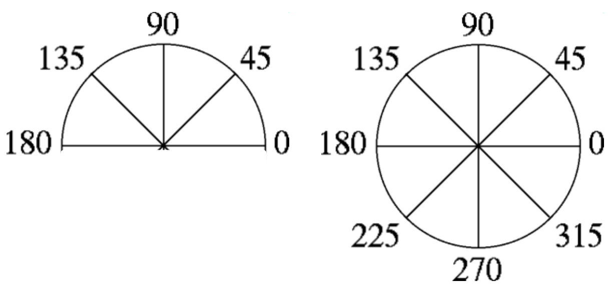
Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

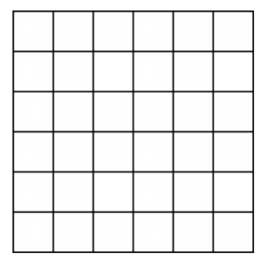


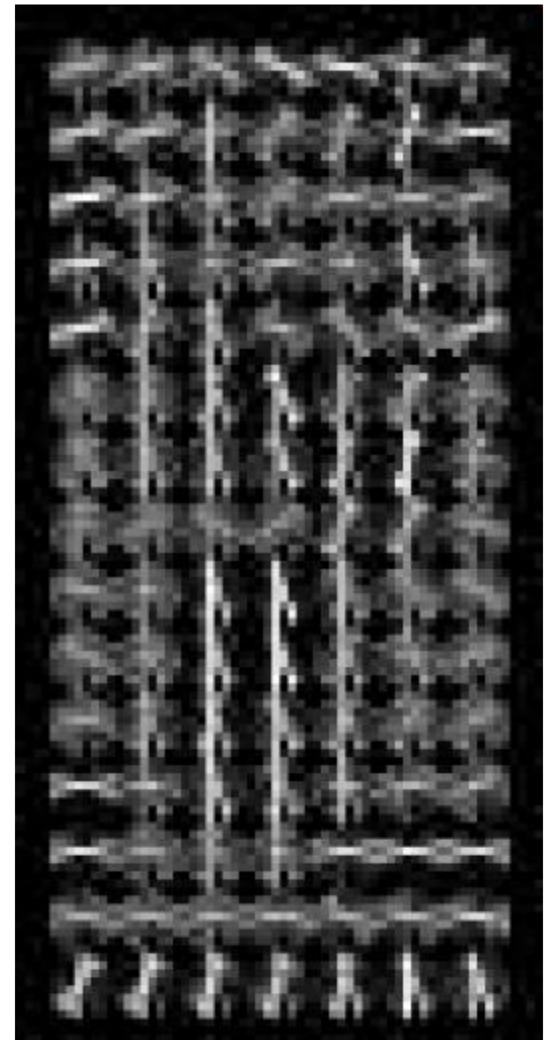
- Histogram of Oriented Gradients

Orientation: 9 bins (for unsigned angles 0 -180)



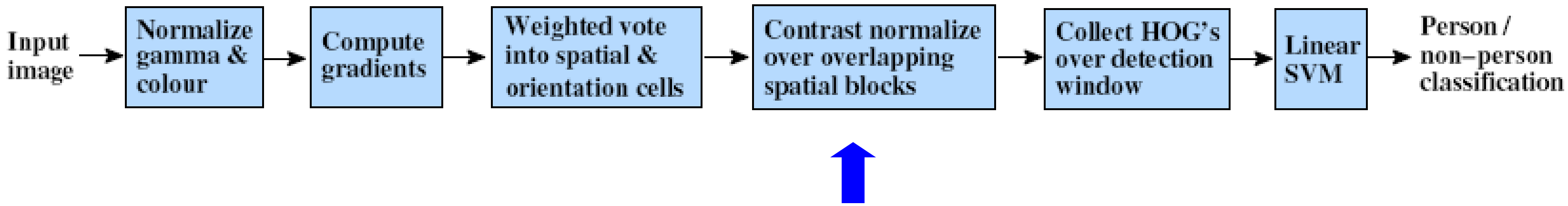
Histograms in k x k pixel cells

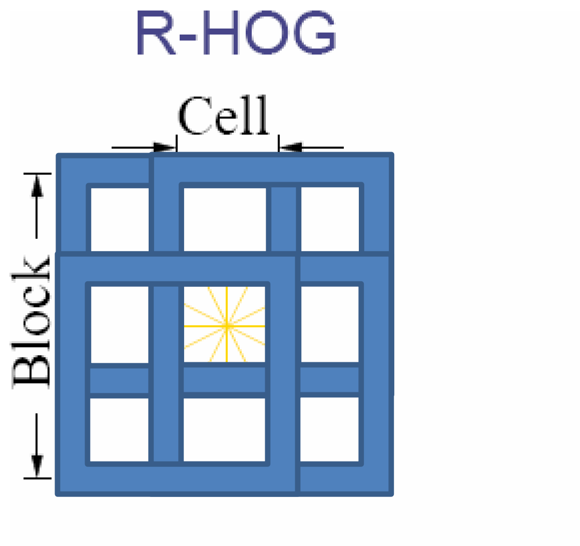




  - Votes weighted by magnitude
  - Bilinear interpolation between cells

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





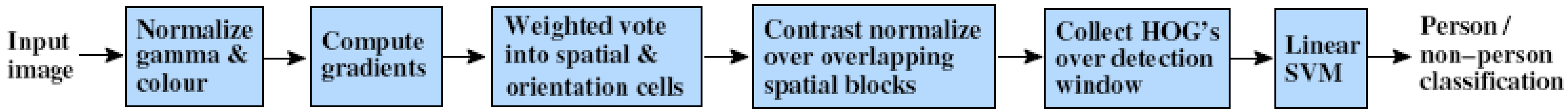
Normalize with respect to surrounding cells

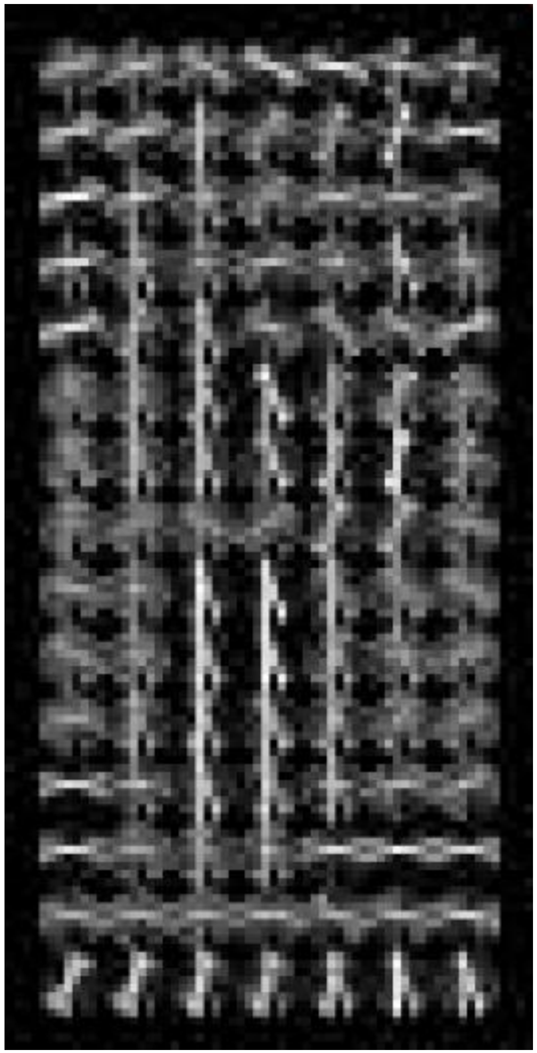
$$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

e is a small constant

Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



X= 

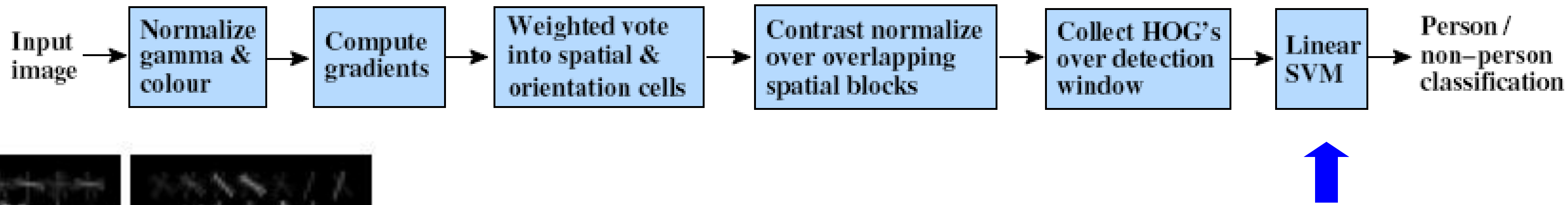
orientations

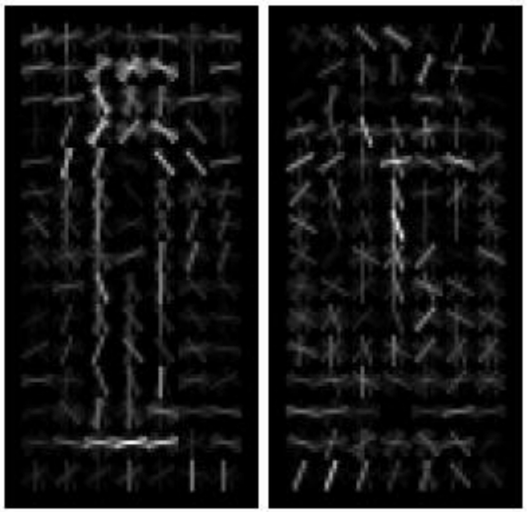
features = 15 x 7 x 9 x 4 = 3780

cells

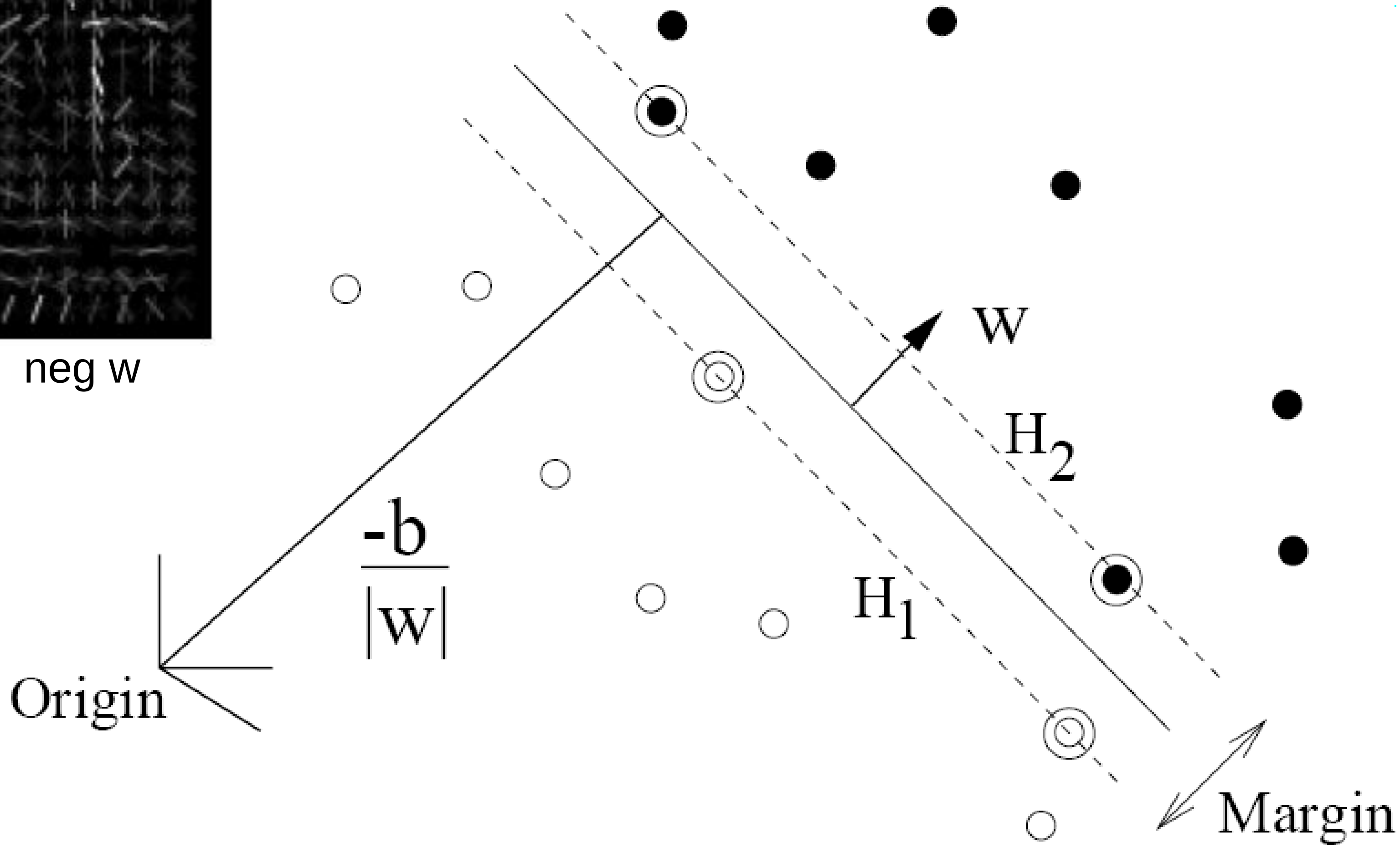
normalizations by neighboring cells

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



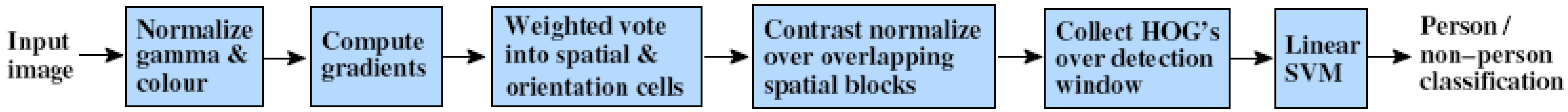


pos w neg w



Slides by Pete Barnum

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05



$$0.16 = w^T x - b$$

$$sign(0.16) = 1$$

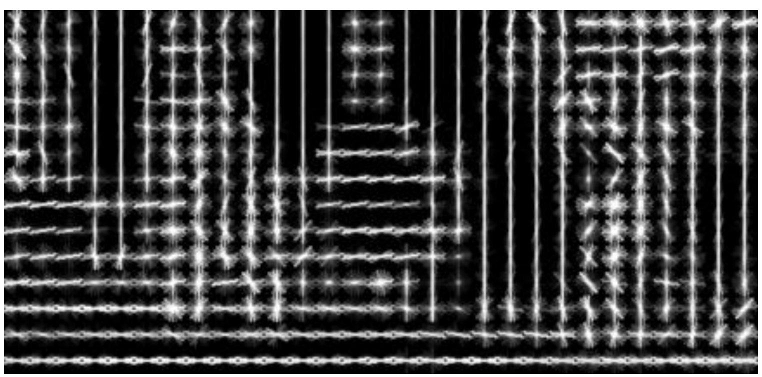
$=>$ pedestrian

Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05

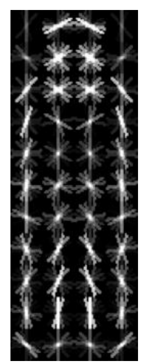
Pedestrian detection with HOG

- Learn a pedestrian template using a support vector machine
- At test time, convolve feature map with template
- Find local maxima of response
- For multi-scale detection, repeat over multiple levels of a HOG *pyramid*

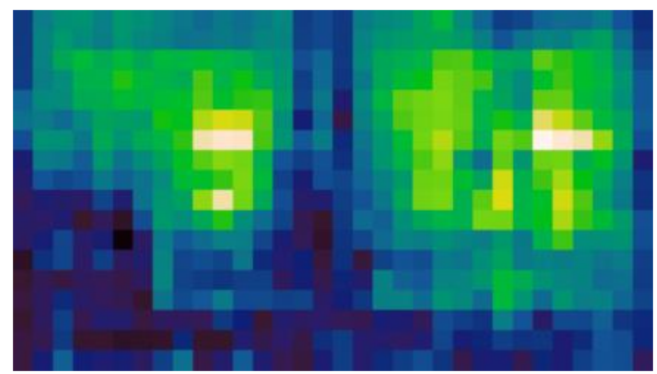
HOG feature map



Template



Detector response map



N. Dalal and B. Triggs, [Histograms of Oriented Gradients for Human Detection](), CVPR 2005

Something to think about…

- Sliding window detectors work
  - *very well* for faces
  - *fairly well* for cars and pedestrians
  - *badly* for cats and dogs
- Why are some classes easier than others?

Strengths/Weaknesses of Statistical Template Approach

Strengths

- Works very well for non-deformable objects with canonical orientations: faces, cars, pedestrians
- Fast detection

Weaknesses

- Not so well for highly deformable objects or “stuff”
- Not robust to occlusion
- Requires lots of training data

Tricks of the trade

- Details in feature computation really matter
  - E.g., normalization in Dalal-Triggs improves detection rate by 27% at fixed false positive rate

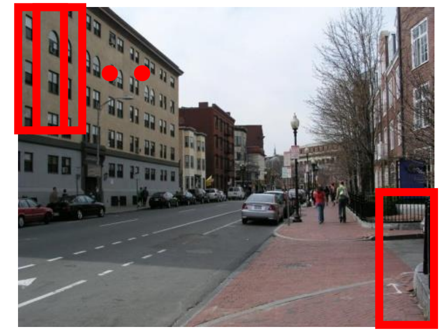
- Template size
  - Typical choice is size of smallest expected detectable object

- “Jittering” or “augmenting” to create synthetic positive examples
  - Create slightly rotated, translated, scaled, mirrored versions as extra positive examples.

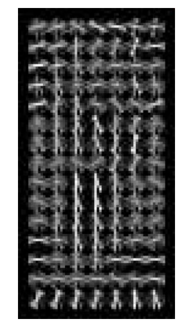
- Bootstrapping to get hard negative examples
  1. Randomly sample negative examples
  2. Train detector
  3. Sample negative examples that score > -1
  4. Repeat until all high-scoring negative examples fit in memory

Things to remember

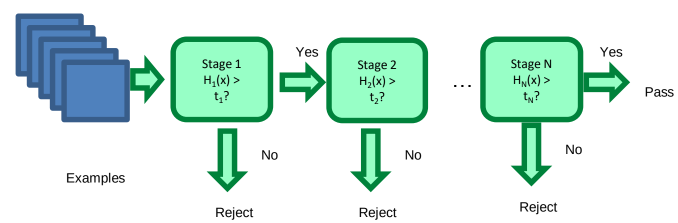
- Sliding window for search



- Features based on differences of intensity (gradient, wavelet, etc.)
  - Excellent results = careful feature design



- Boosting for feature selection
- Integral images, cascade for speed



- Bootstrapping to deal with many, many negative examples

Project 5

- Train Dalal-Triggs model for faces
- Classify examples

- We need some test photographs…