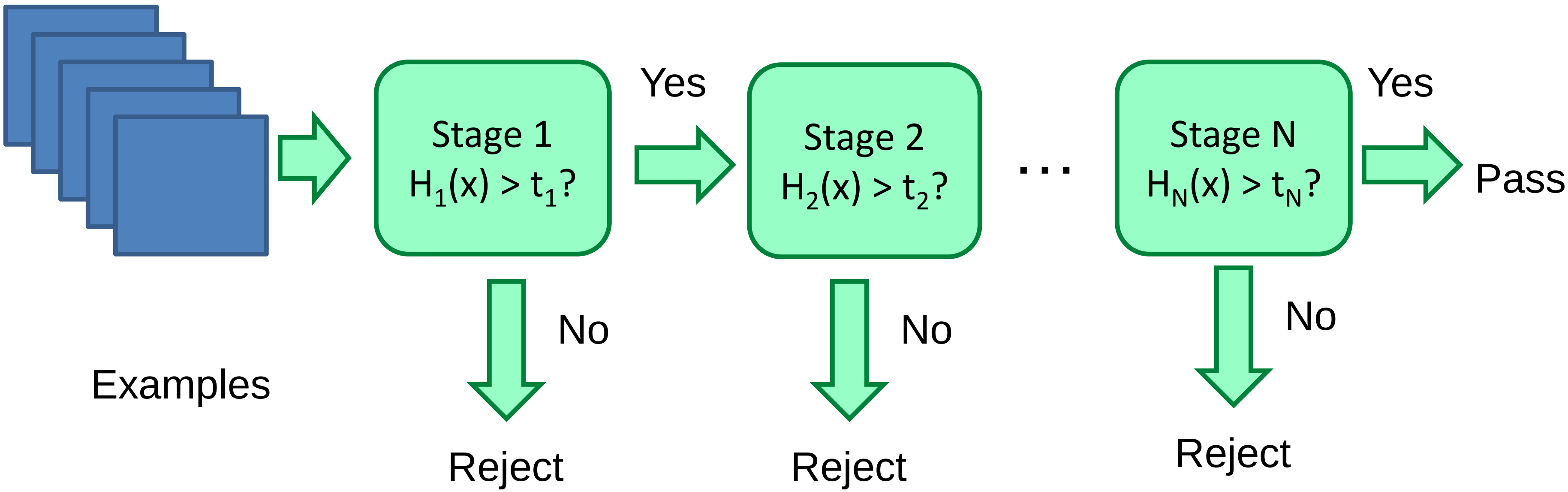
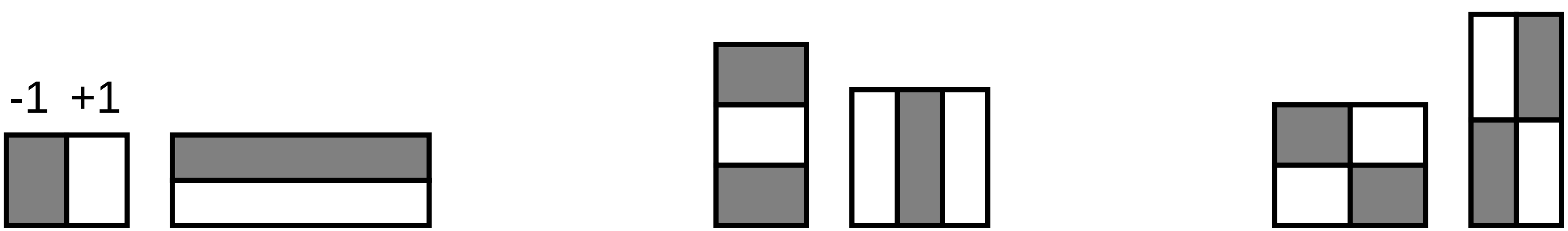
Cascade for Fast Detection



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



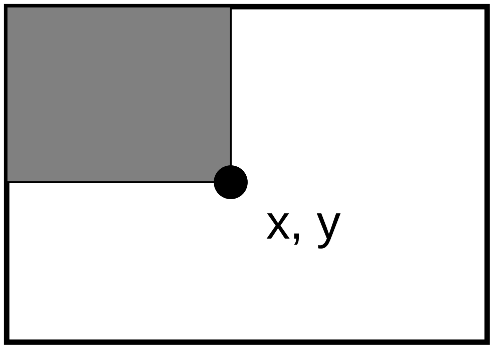
Two-rectangle features

Three-rectangle features

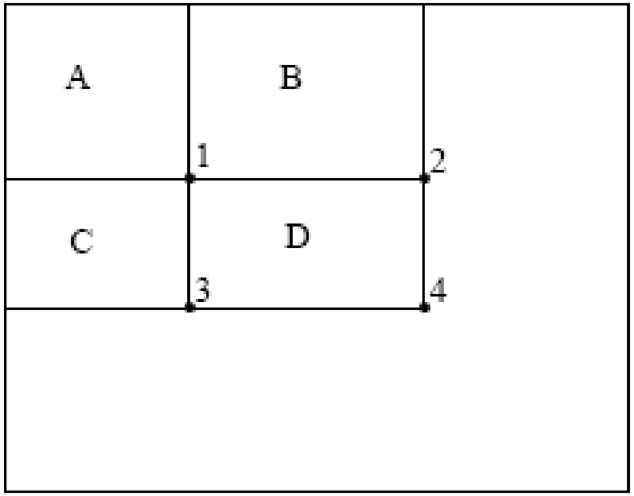
Etc.

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 Adaboost

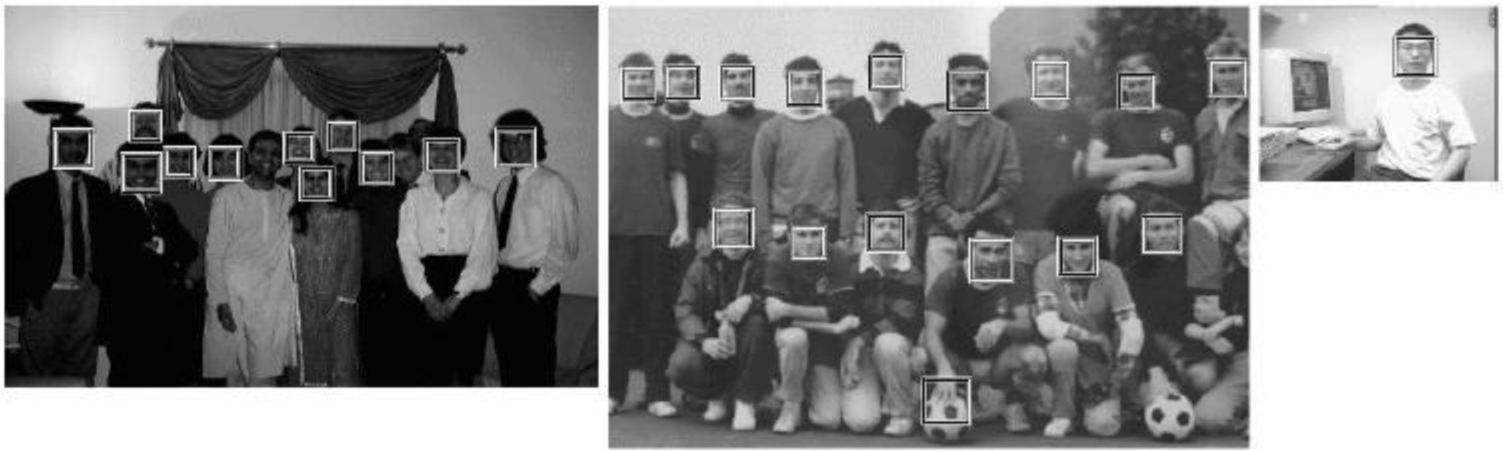
- Create a large pool of features (180K)
- Select features that are discriminative and work well together
  - “Weak learner” = feature + threshold + parity

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

  - Choose weak learner that minimizes error on the weighted training set
  - Reweight

Viola Jones Results

Speed = 15 FPS (in 2001)



Detector \ False detections	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

Object Detection

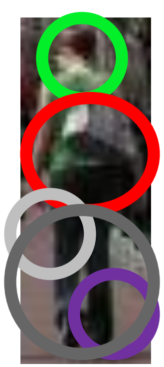
- Overview
- Viola-Jones
- Dalal-Triggs
- Deformable models
- Deep learning

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



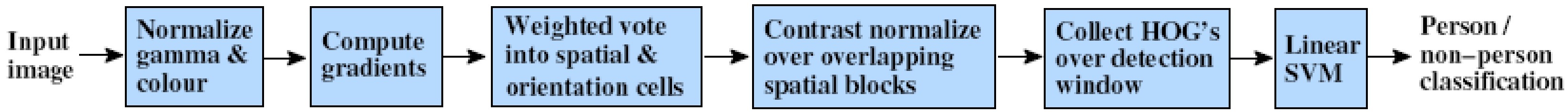
$$+4 + 1 + 0.5 + 3 + 0.5 = 10.5 \overset{?}{>} 7.5$$

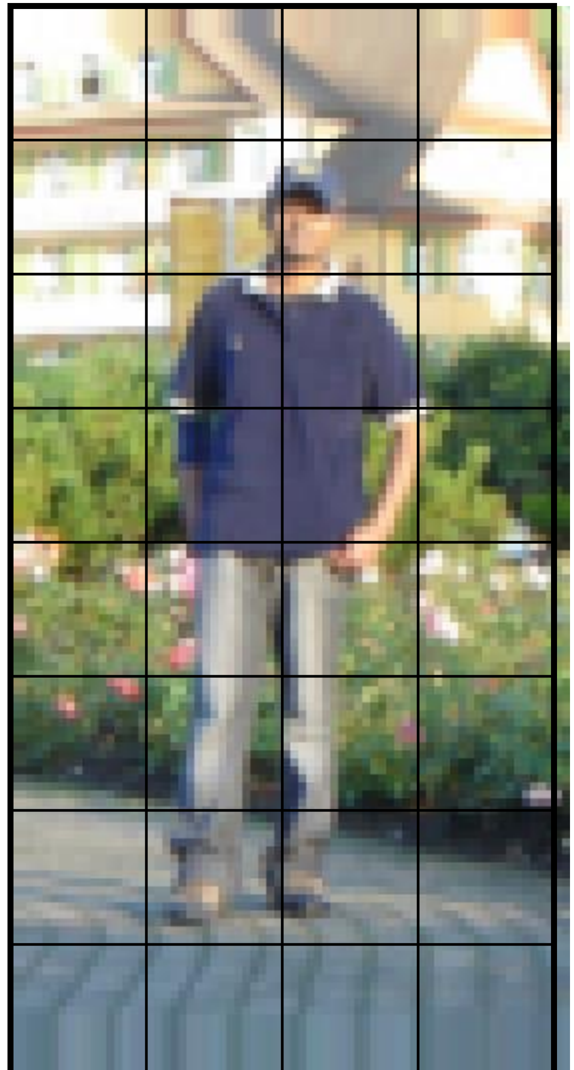
Object

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

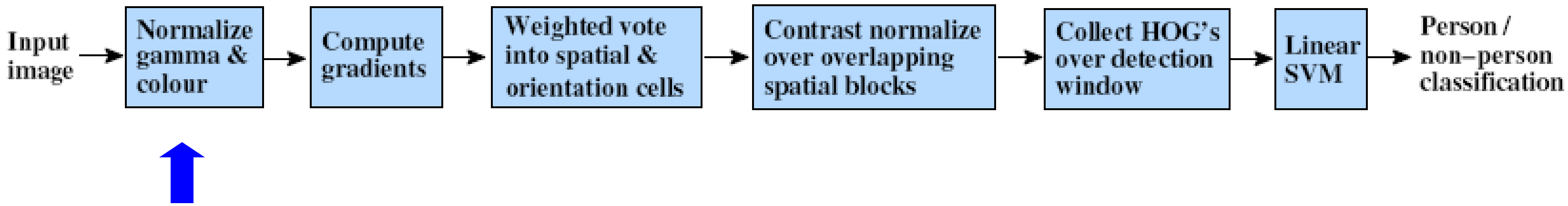
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05





Slides by Pete Barnum

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

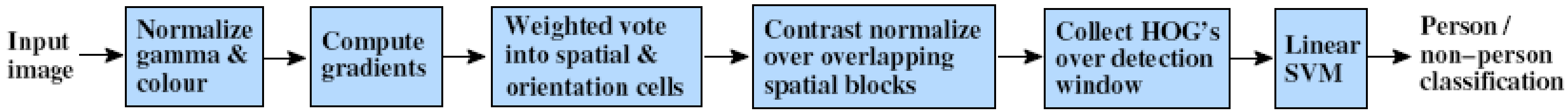


- Tested with
  - RGB
  - LAB
  - Grayscale

Slightly better performance vs. grayscale

- Gamma Normalization and Compression
  - Square root
  - Log

Very slightly better performance vs. no adjustment



Outperforms

-1	0	1

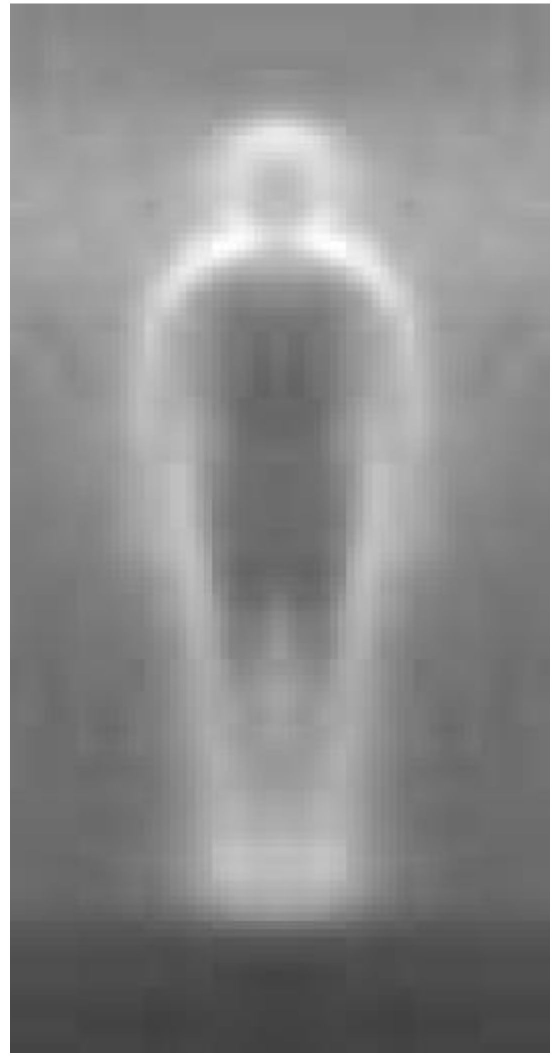
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected



0	1
-1	0

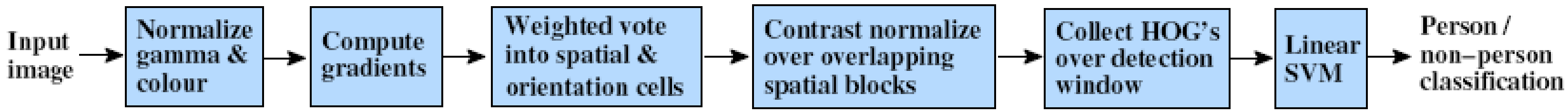
diagonal

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

Sobel

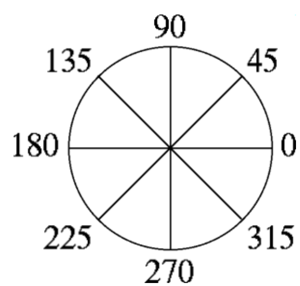
Slides by Pete Barnum

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

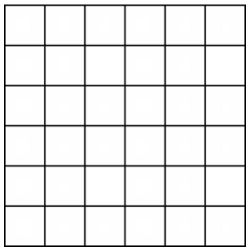


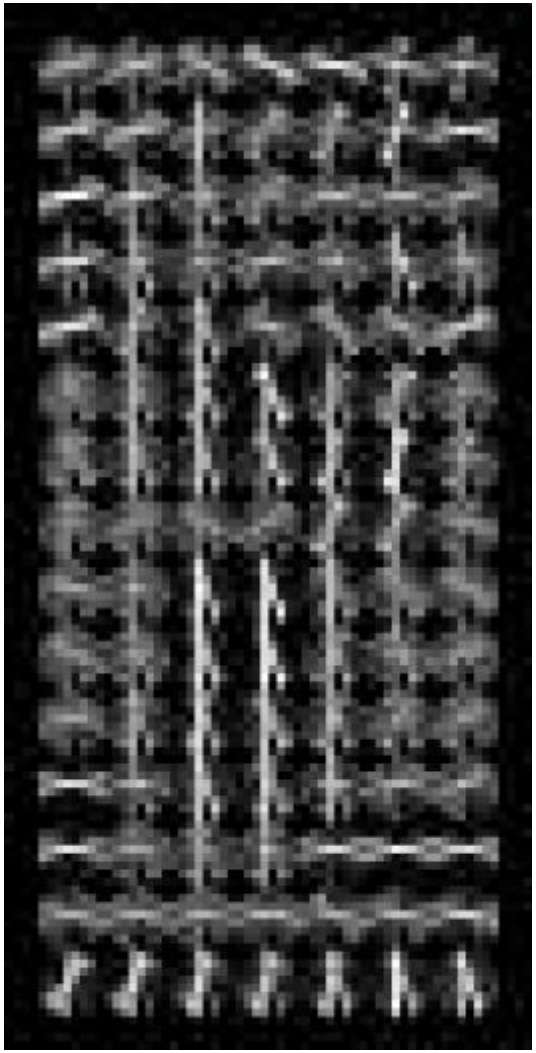
- Histogram of gradient orientations

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



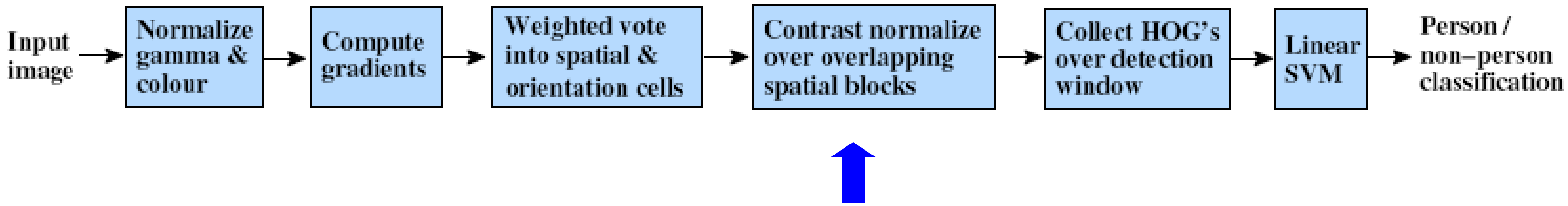
Histograms in k x k pixel cells



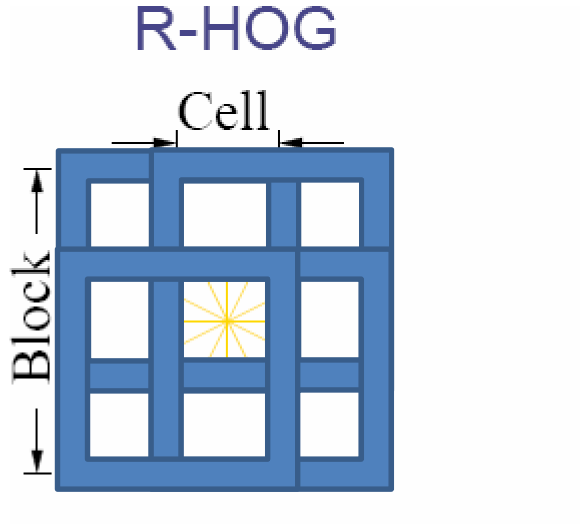


  – Votes weighted by magnitude
  – Bilinear interpolation between cells

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

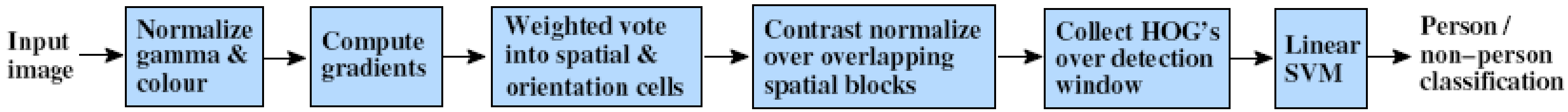


Normalize with respect to surrounding cells

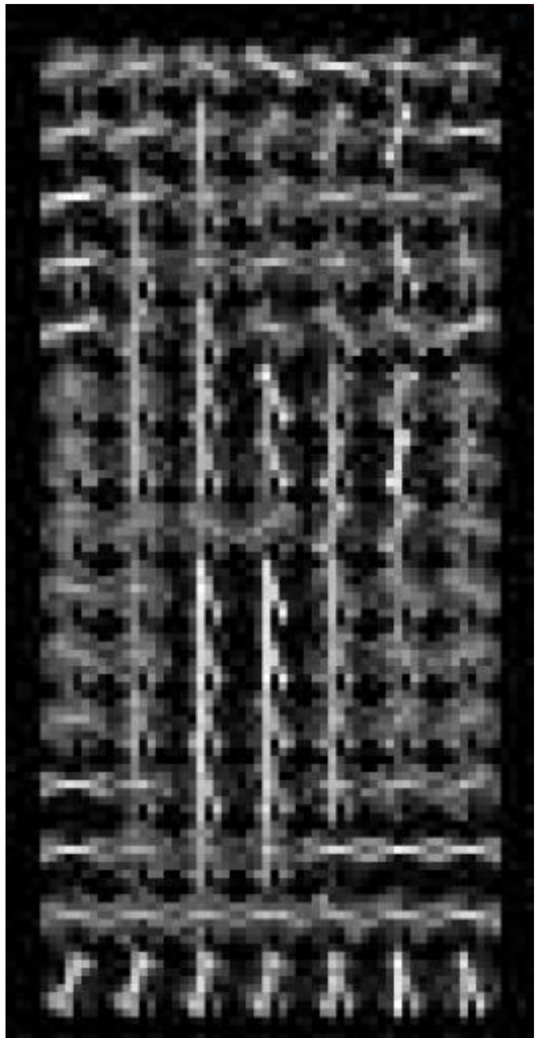


$$L2-norm : v \longrightarrow v/\sqrt{||v||_2^2+\epsilon^2}$$

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



X=



Original Formulation

orientations

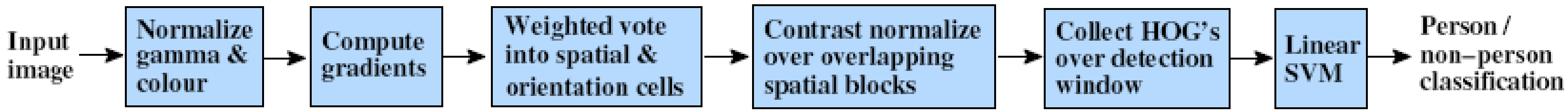
features = 15 x 7 x 9 x 4 = 3780

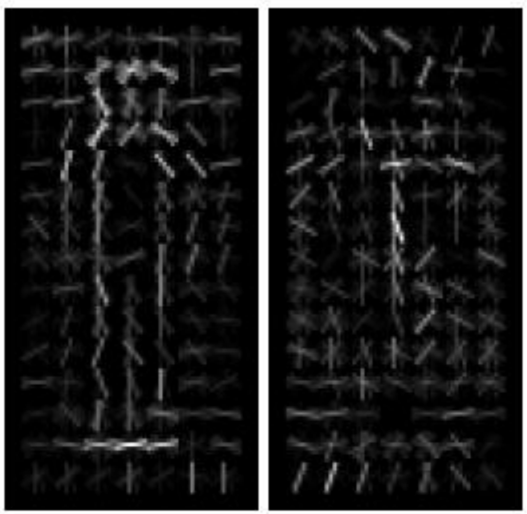
cells

normalizations by neighboring cells

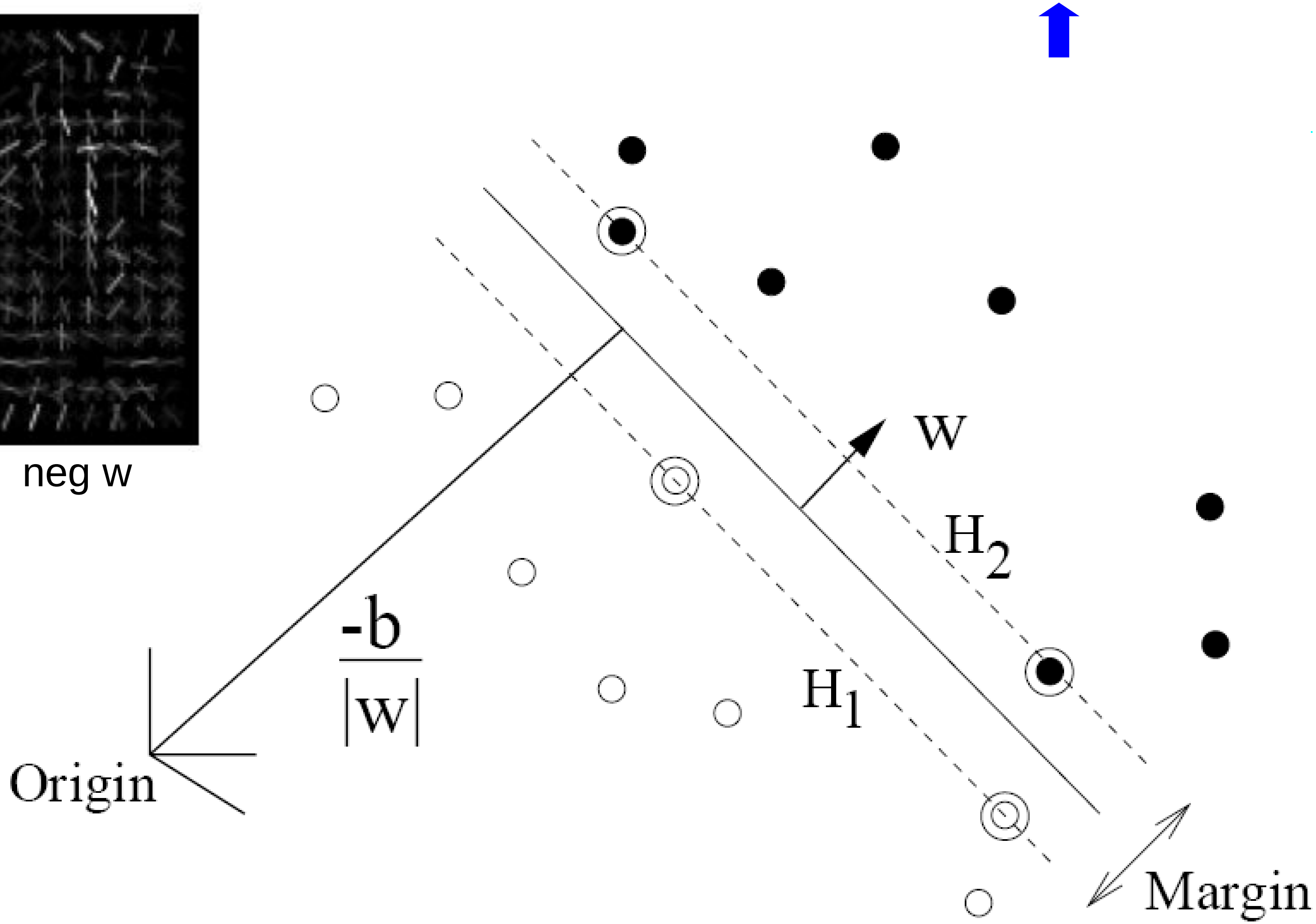
Slides by Pete Barnum

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



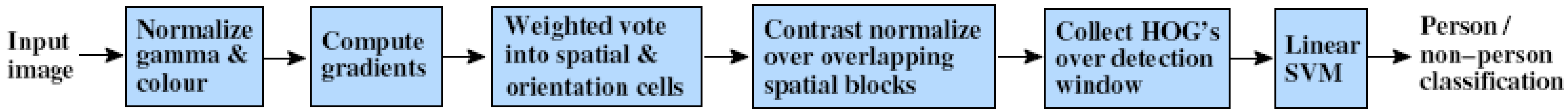


pos w neg w



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

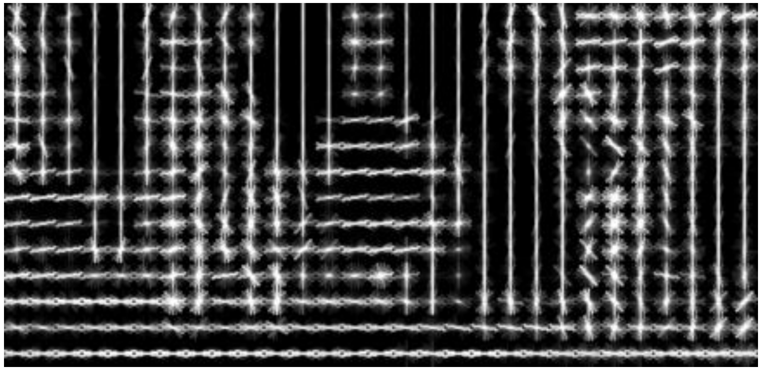
Slides by Pete Barnum

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

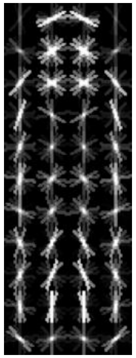
Pedestrian detection with HOG

- Train a pedestrian template using a linear 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*

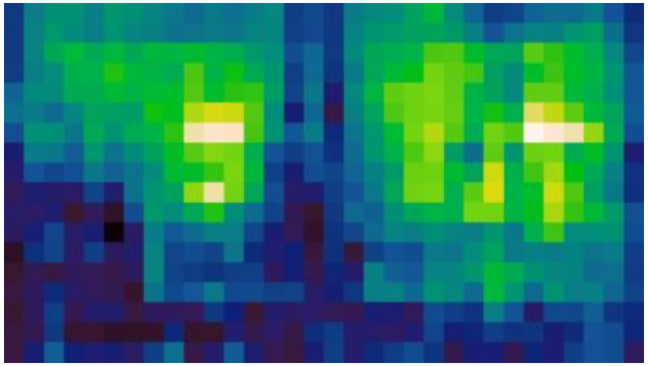
HOG feature map



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 and Weaknesses of Statistical Template Approach

Strengths

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

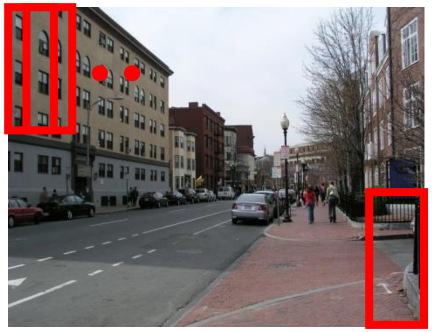
Weaknesses

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

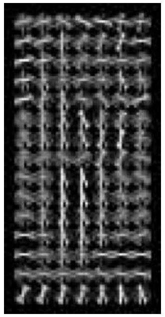
Tricks of the trade

- 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 detectable object
- “Jittering” 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

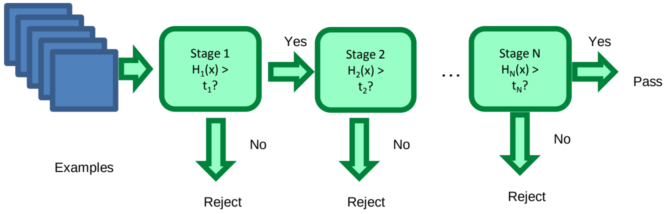
Things to remember



- Sliding window for search
- Features based on differences of intensity (gradient, wavelet, etc.)
  - Excellent results require careful feature design



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



- Bootstrapping to deal with many, many negative examples