EBU7240

Computer Vision

- Detection1: Pedestrian detection -

Semester 1, 2021

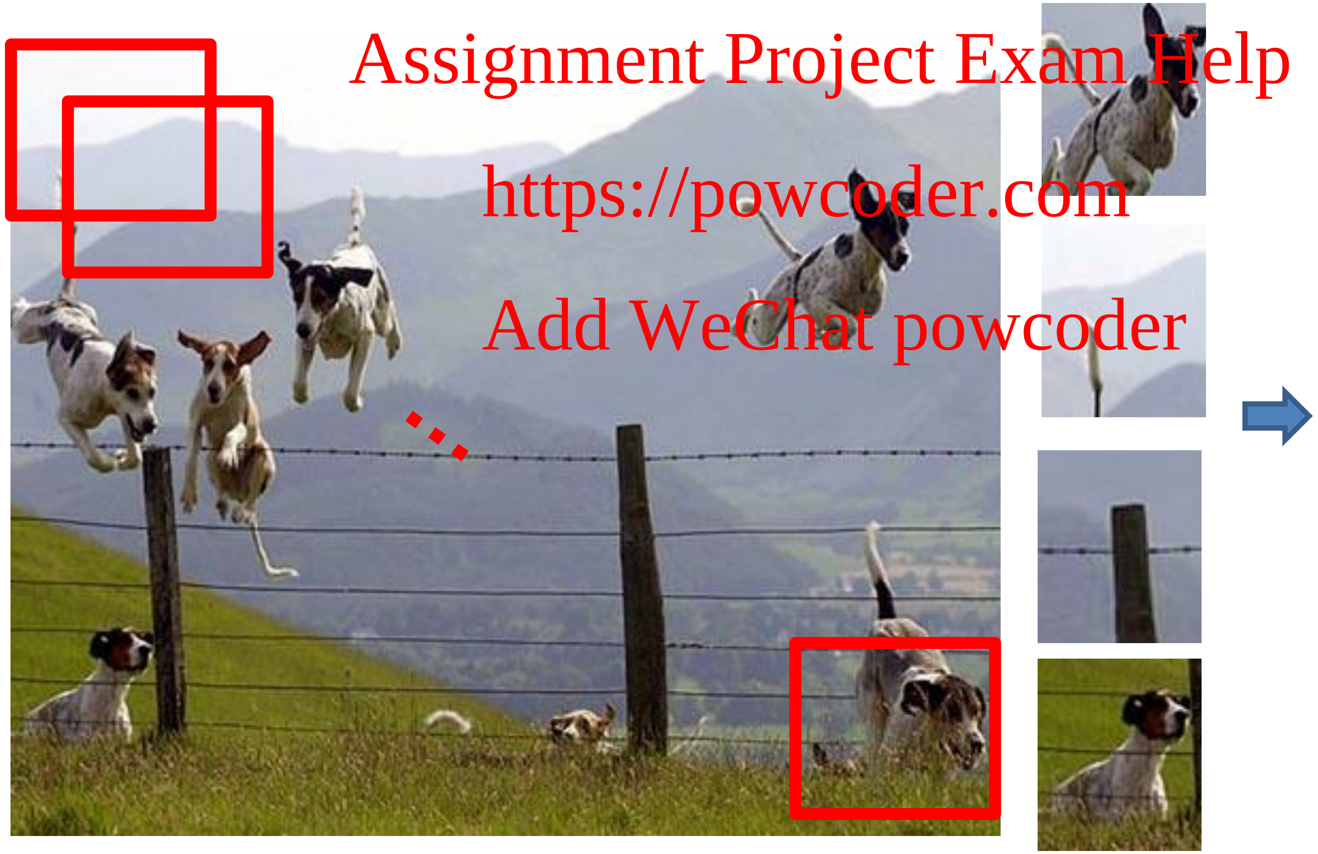
Changjae Oh

Outline

- **Overview**
- **Dalal-Triggs (pedestrian detection)**
  - Histogram of Oriented Gradients
  - Learning with SVM

Object Detection

- Focus on **object search**: "Where is it?"
- Build templates that differentiate **object patch** from **background patch**



Object or
Non-Object?

J.Hays

Challenges in modeling the object class

Illumination

Object pose

'Clutter'

Occlusions

Intra-class appearance

Viewpoint

[K. Grauman, B. Leibe]

Challenges in modeling the non-object class

True
Detections

Bad
Localization

Confused with
Similar Object

Misc. Background

Confused with
Dissimilar Objects

[Hays]

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

General Process of Object Detection

Specify Object Model

What are the object parameters?

↓

Generate Hypotheses

↓

Score Hypotheses

↓

Resolve Detections

Specifying an object model

1. **Statistical Template in Bounding Box**
   - Object is some (x,y,w,h) in image
   - Features defined with respect to bounding box coordinates

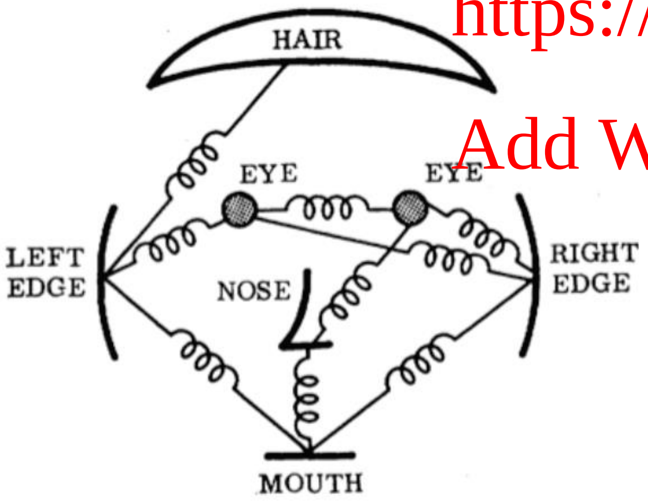
Image

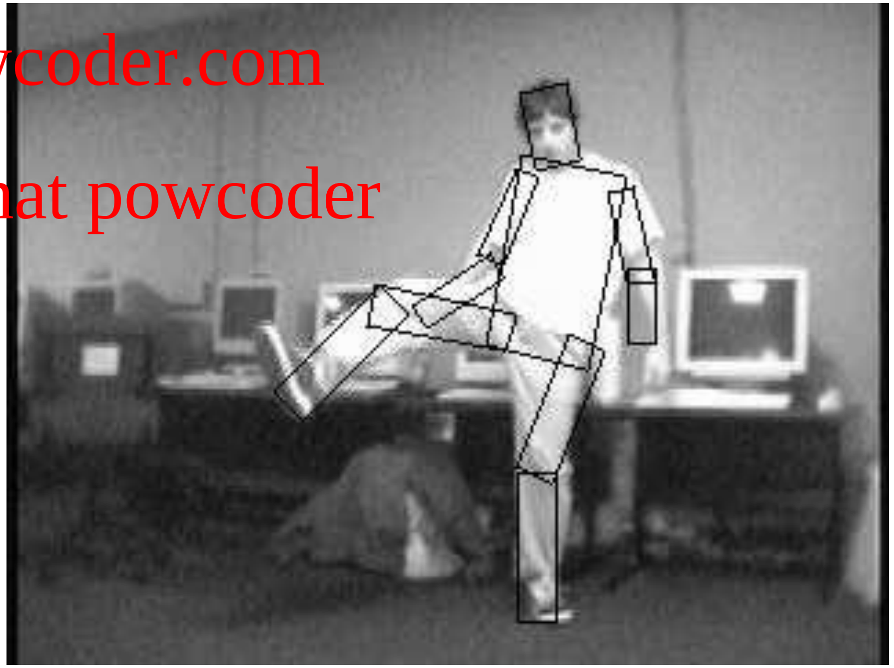
Template Visualization

Specifying an object model

2. Articulated parts model

- Object is configuration of parts
- Each part is detectable



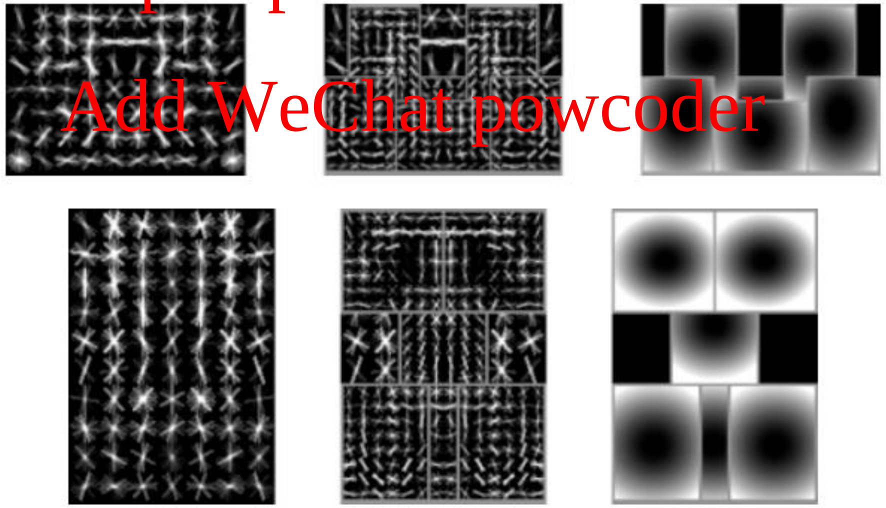


Specifying an object model

3. Hybrid template/parts model

Detections

Template Visualization



root filters
coarse resolution

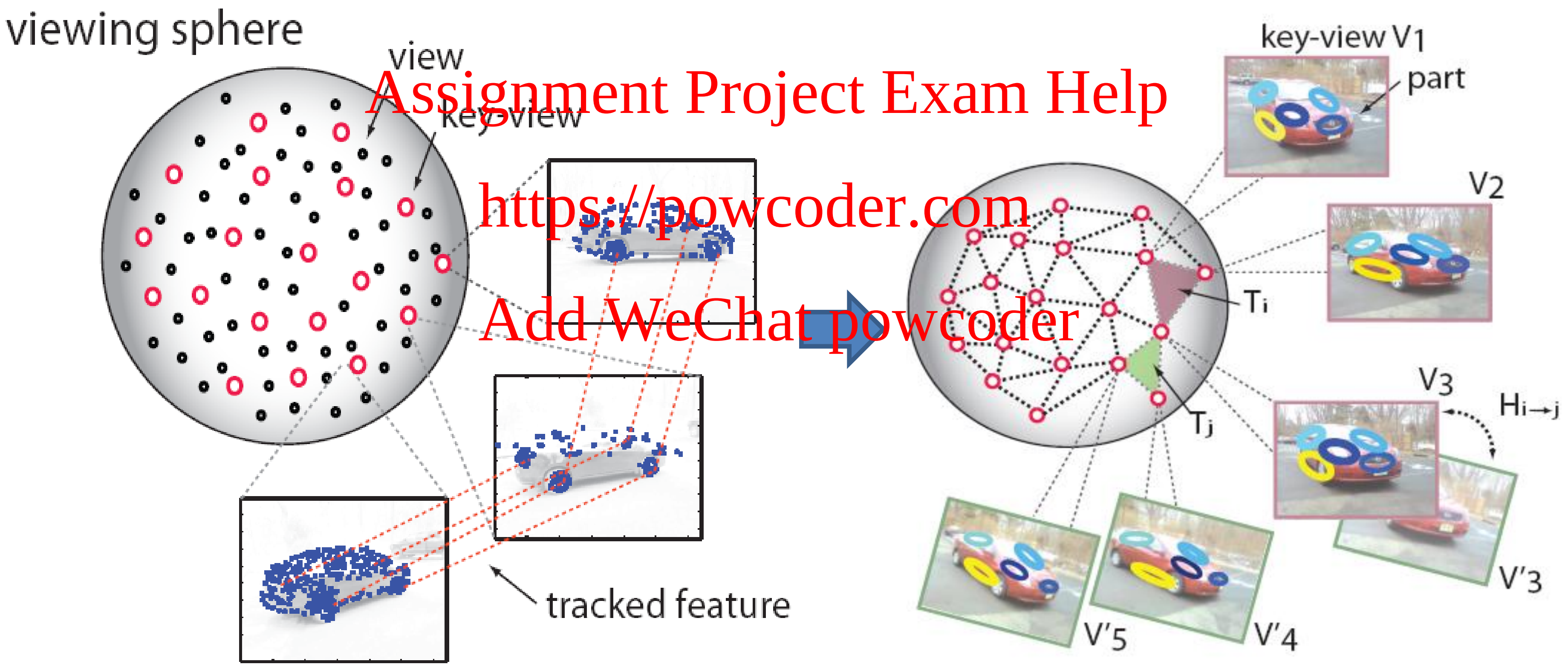
part filters
finer resolution

deformation
models

Specifying an object model

4. 3D-ish model

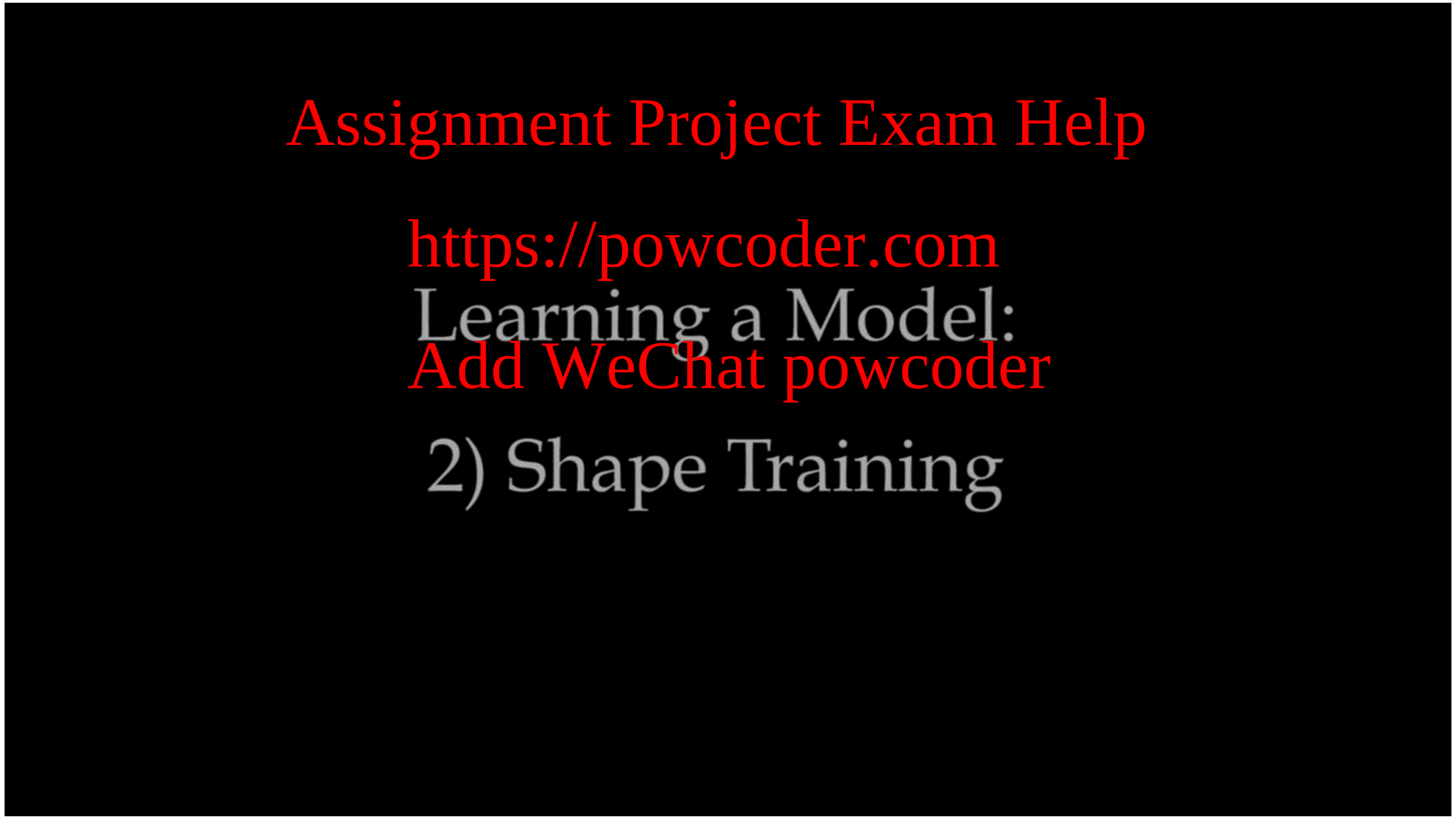
– Object is collection of 3D planar patches under affine transformation



Specifying an object model

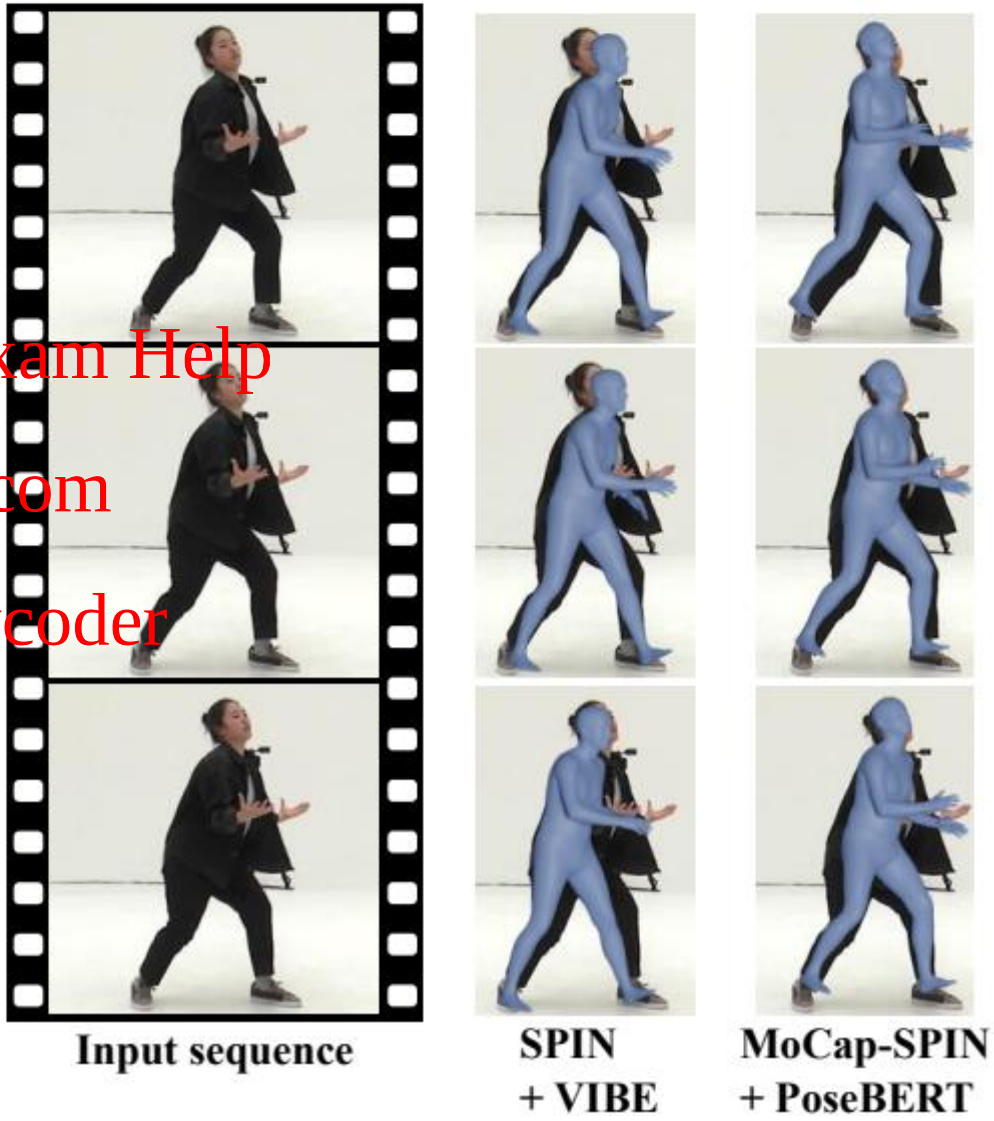
4. Deformable 3D model

– Object is a parameterized space of shape/pose/deformation of class of 3D object



Why not just pick the most complex model?

- **Inference is harder**
  - More parameters
  - Harder to 'fit' (infer / optimize fit)
  - Longer computation



Fabien Baradel, Leveraging MoCap Data for Human Mesh Recovery, Arxiv 2021

General Process of Object Detection

Specify Object Model

↓

Generate Hypotheses

Propose an alignment of the model to the image

↓

Score Hypotheses

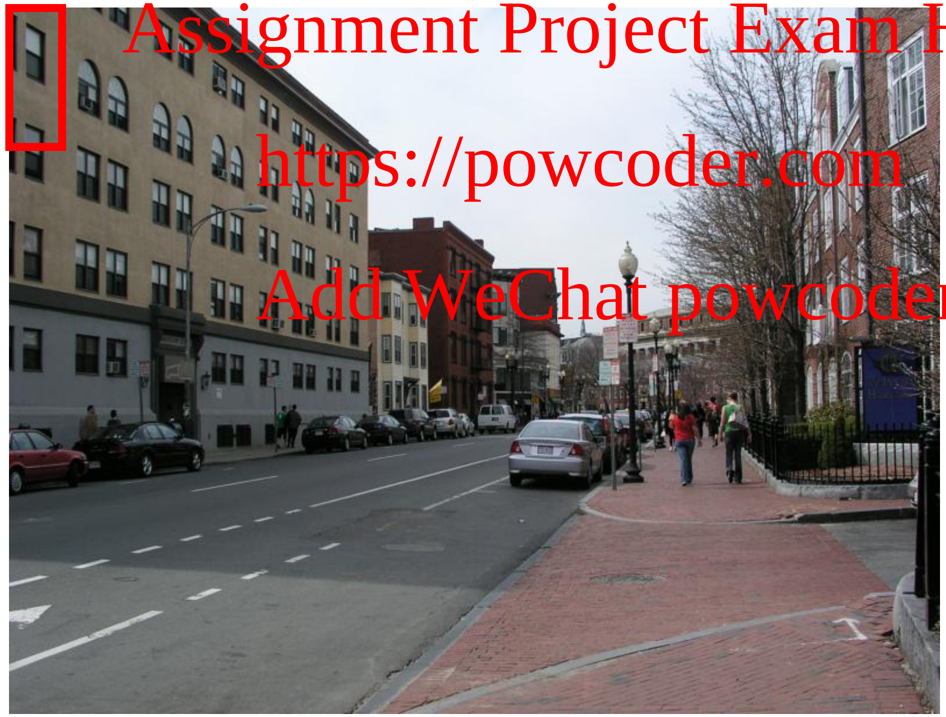
↓

Resolve Detections

James Hays

Generating hypotheses

1. **2D template model / sliding window**
   – Test patch at each location and scale



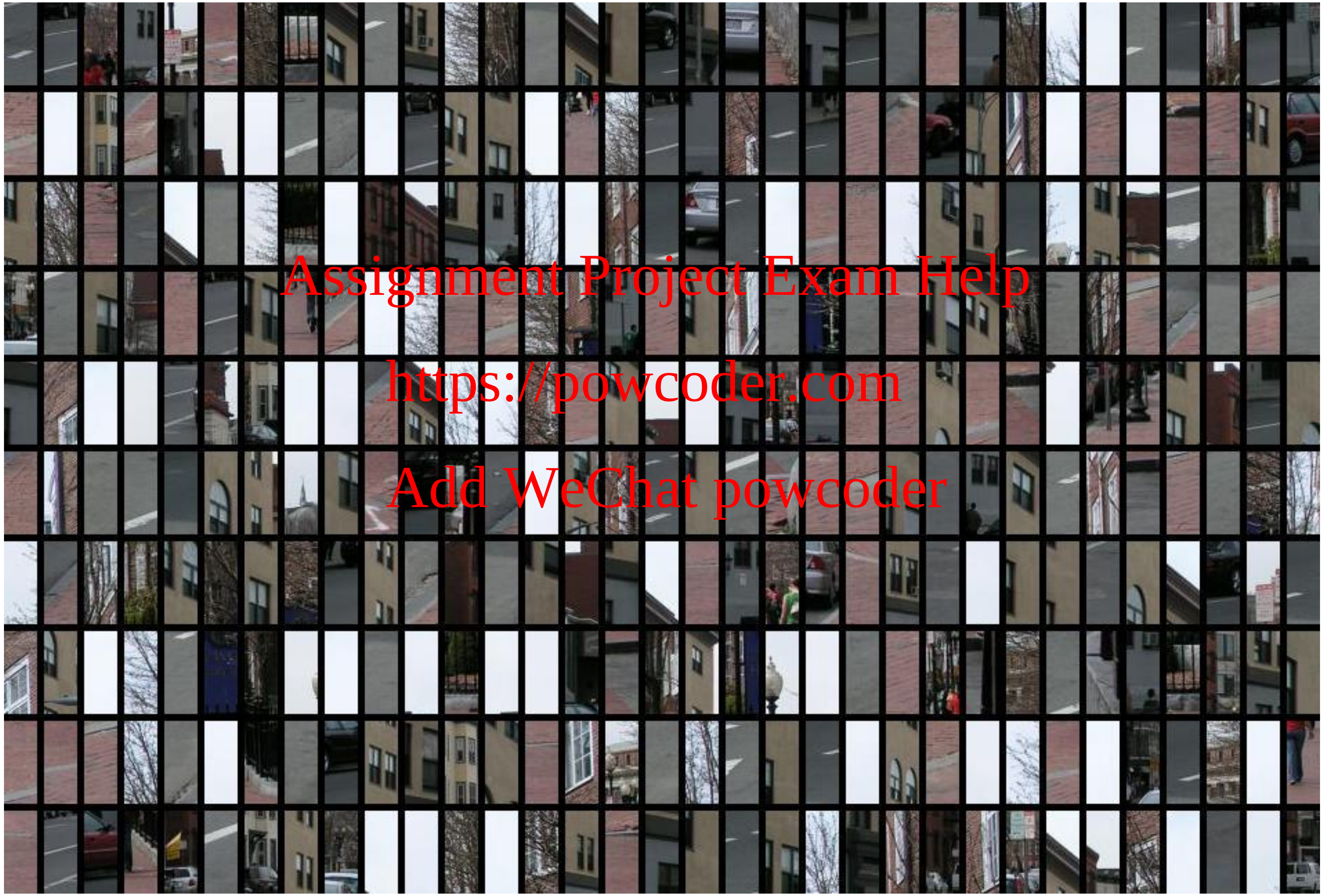
James Hays

Generating hypotheses

1. **2D template model / sliding window**
   – Test patch at each location and scale

Note – Template did not change size

Each window is separately classified



Generating hypotheses

2. **Region-based proposal**
   – Arbitrary bounding box + image 'cut' segmentation

Endres Hoiem 2010

General Process of Object Detection

Specify Object Model

↓

Generate Hypotheses

↓

Score Hypotheses

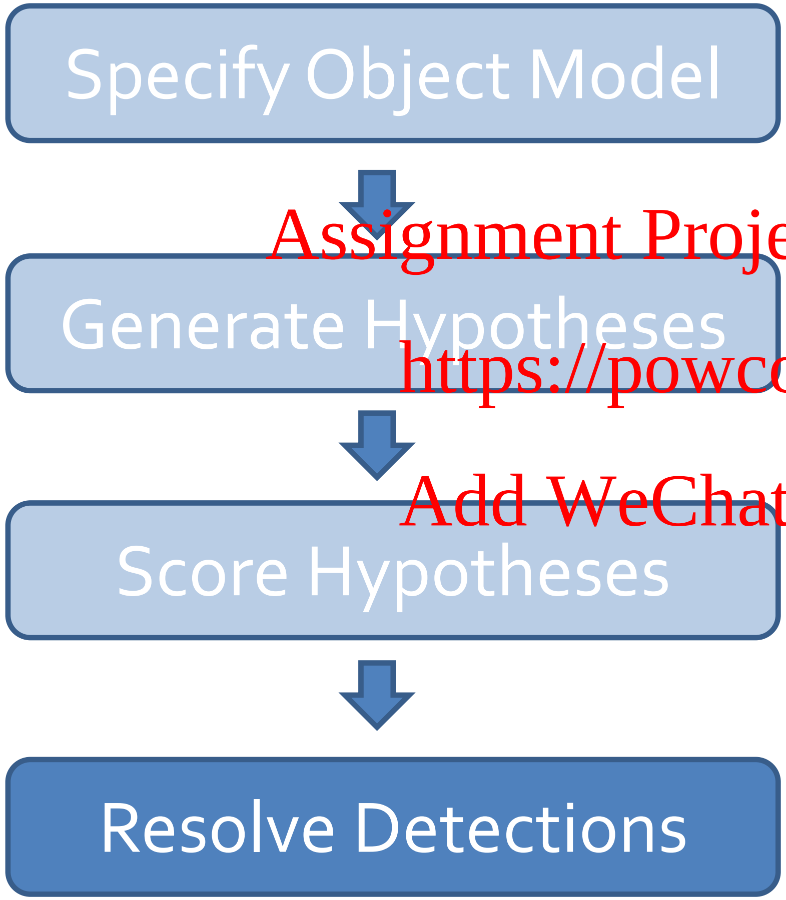
↓

Resolve Detections

Mainly gradient-based features, usually based on summary representation, many classifiers.

James Hays

General Process of Object Detection

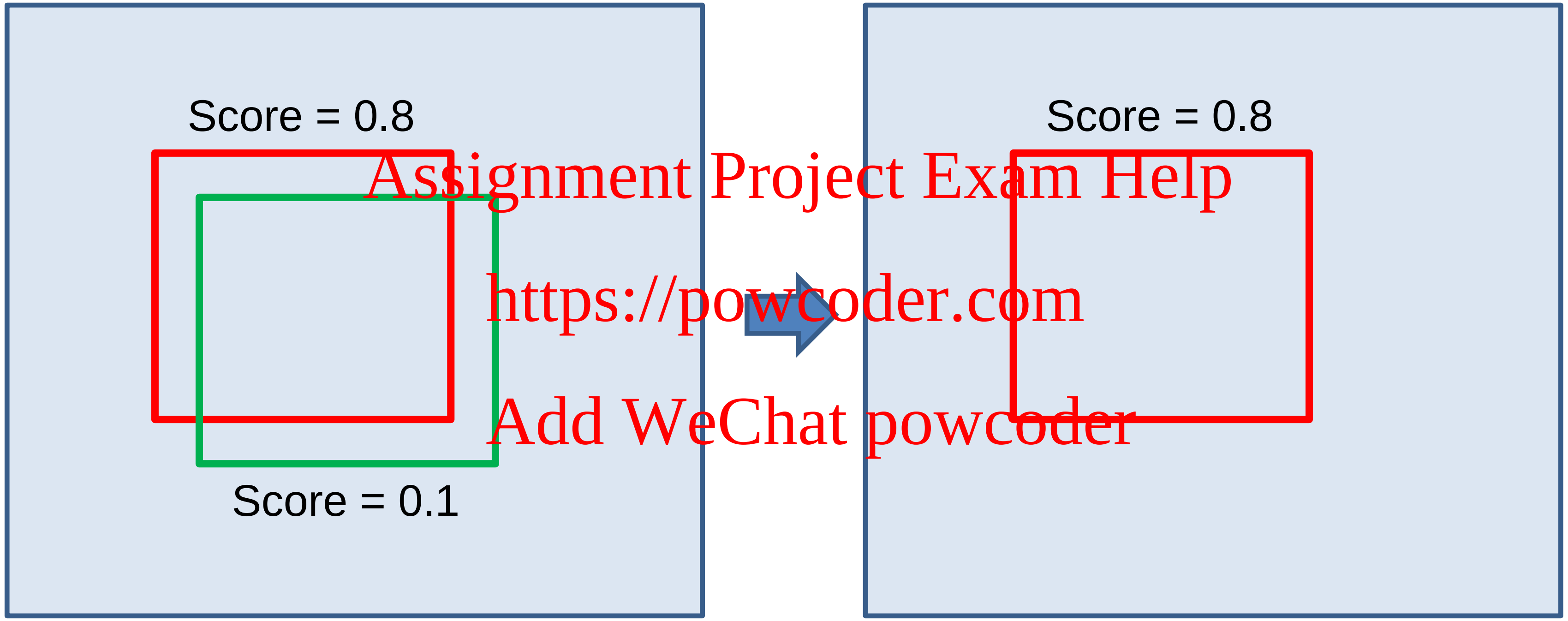


Rescore each proposed object based on the whole set

James Hays

Resolving detection scores

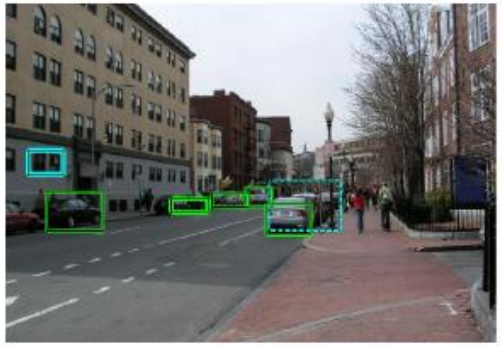
1. **Non-max suppression**



Resolving detection scores

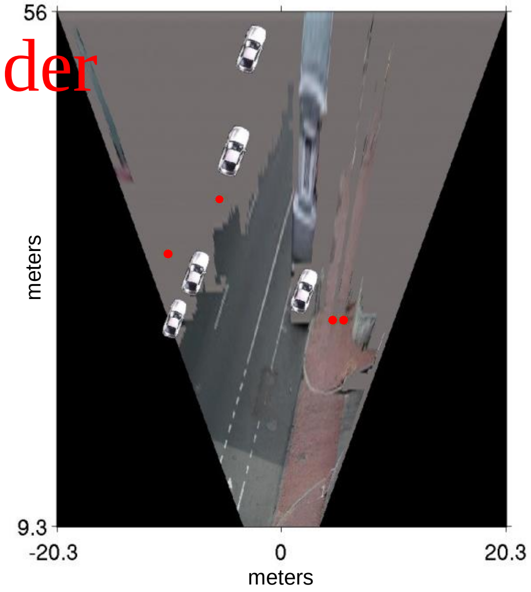
2. Context/reasoning

- Via geometry
- Via known information or prior distributions
- Non-max suppression



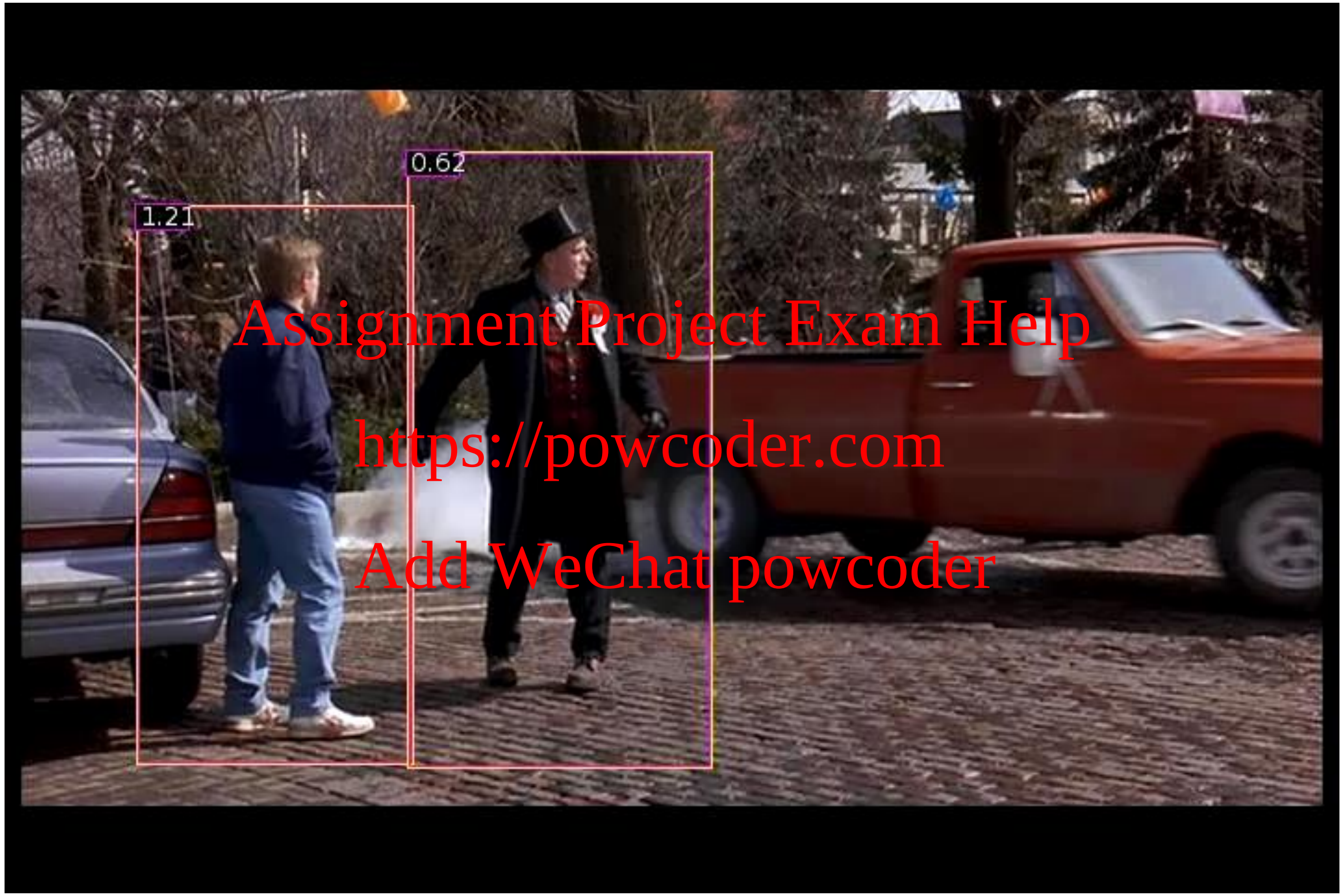
(g) Car Detections: Local

(h) Ped Detections: Local



Hoiem et al. 2006

Dalal Triggs: Person detection with HOG & linear SVM



Histograms of Oriented Gradients for Human Detection, Navneet Dalal, Bill Triggs, International Conference on Computer Vision & Pattern Recognition - June 2005

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

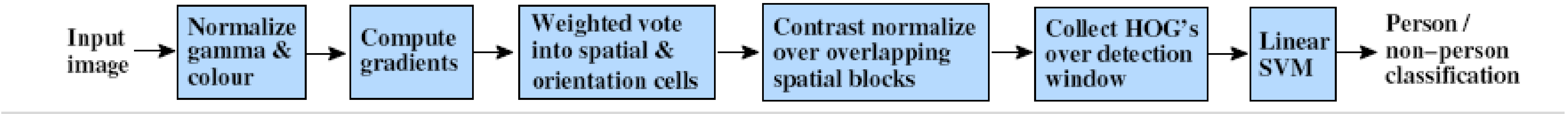
Example: Dalal-Triggs pedestrian detector

1. Extract fixed-sized (64x128 pixel) window at each position and scale

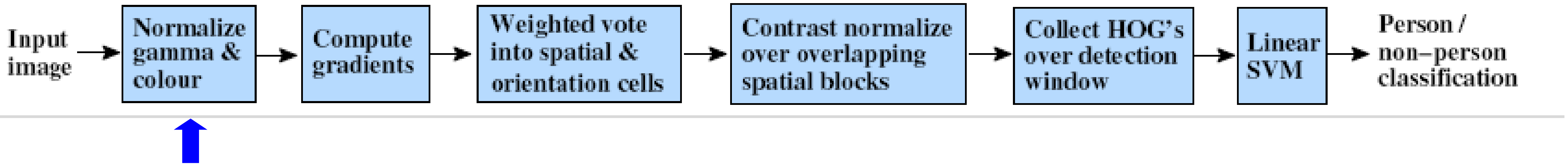
2. Compute HOG (histogram of gradient) features within each window
3. Score the window with a linear SVM classifier

4. Perform non-maxima suppression to remove overlapping detections w ith lower scores

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



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

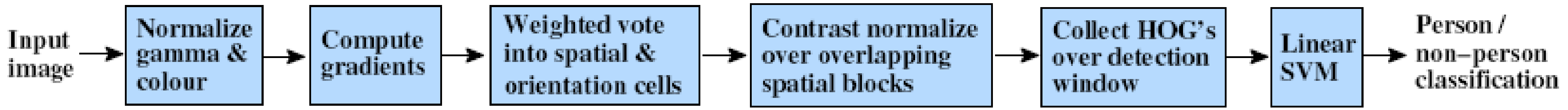


- **Tested with**
  - RGB
  - LAB
  - Grayscale

  Slightly better performance vs. grayscale (RGB, LAB)

- **Gamma Normalization and Compression**
  - Square root
  - Log

  Very slightly better performance vs. no adjustment (Square root, Log)



Outperforms

-1	0	1

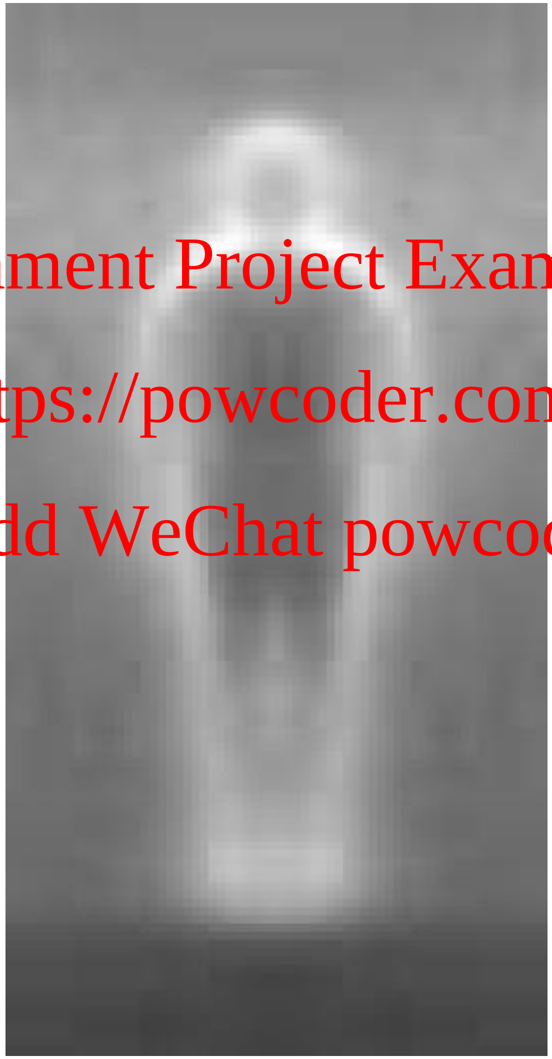
centered

-1	1

uncentered

1	-8	0	8	-1

cubic-corrected

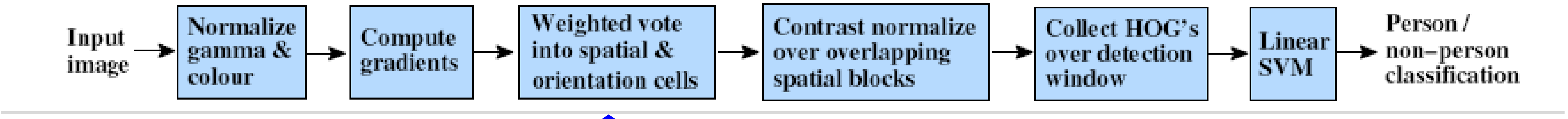


0	1
-1	0

diagonal

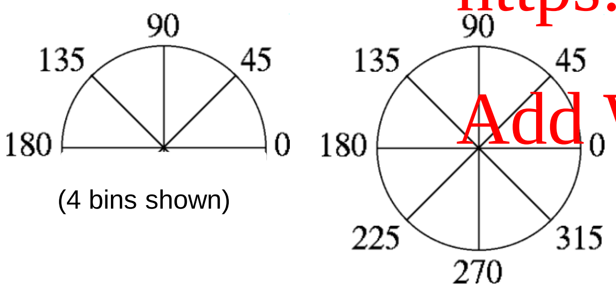
-1	0	1
-2	0	2
-1	0	1

Sobel

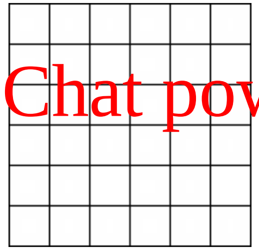


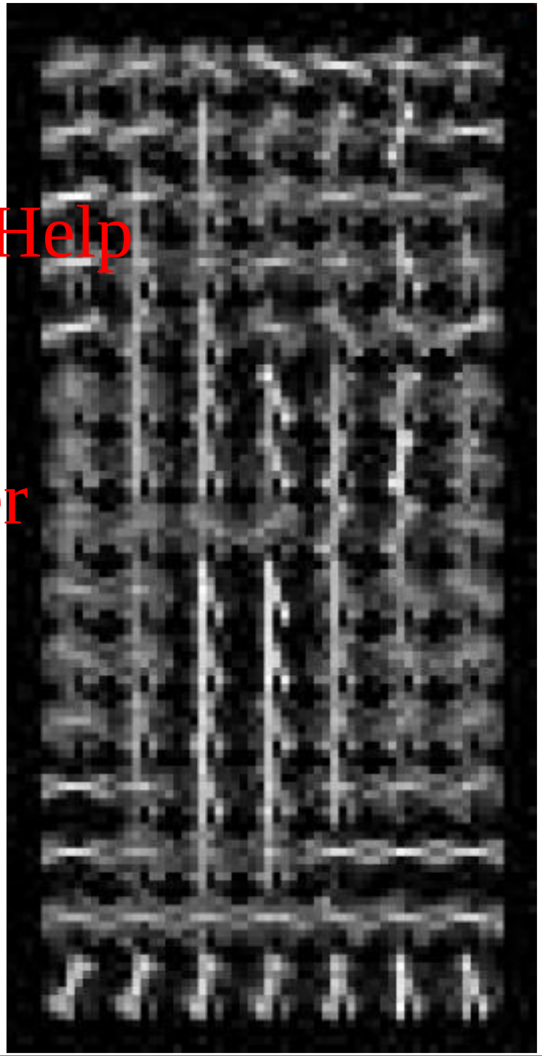
Histogram of Oriented Gradients

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



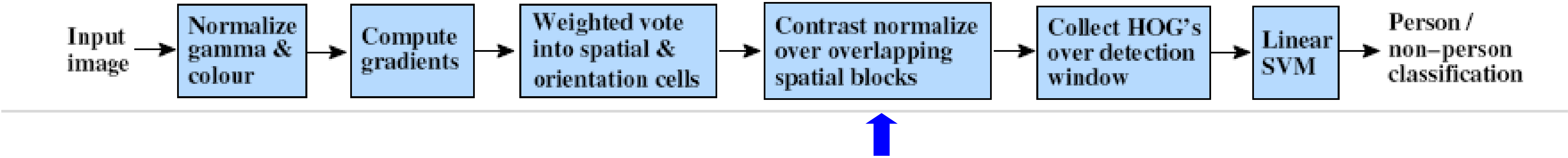
Histograms over k x k pixel cells





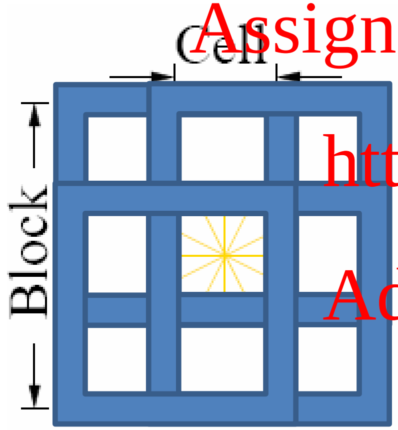
– Votes weighted by magnitude
– Bilinear interpolation between cells

Slides by Pete Barnum



Normalize with respect to surrounding cells

Rectangular HOG (R-HOG)

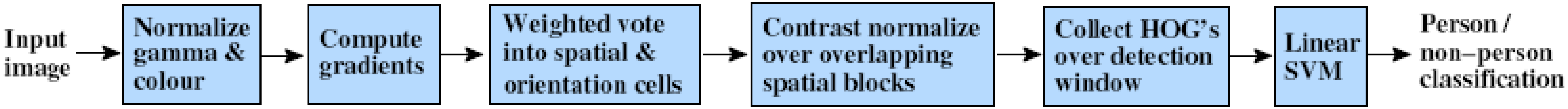


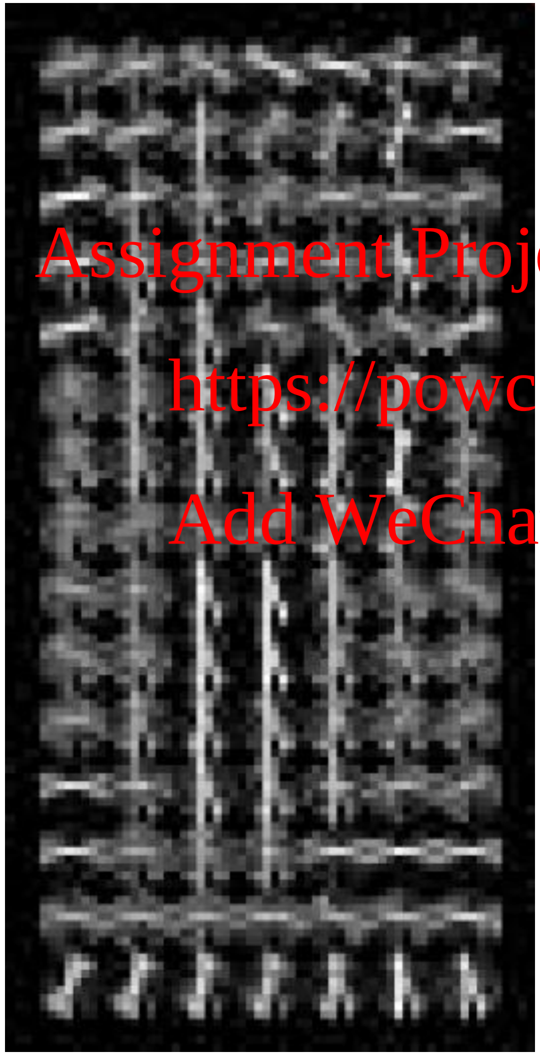
How to normalize?

- Concatenate all cell responses from neighboring blocks into vector.
- Normalize vector.
- Extract responses from cell of interest.
- Do this 4x for each neighbor set in 2x2.

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

e is a small constant
(to remove div. by zero on empty bins)



X= 

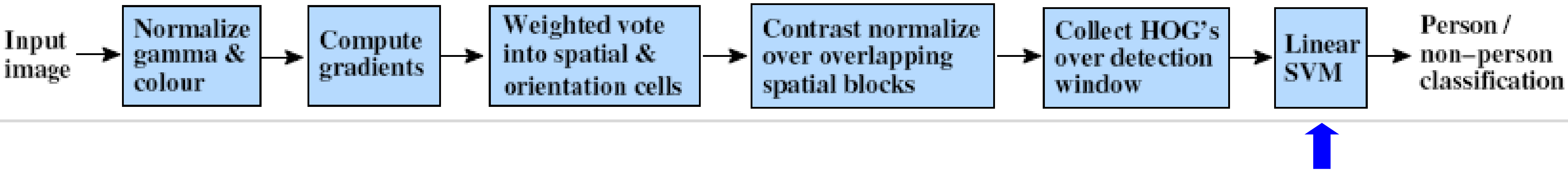
orientations

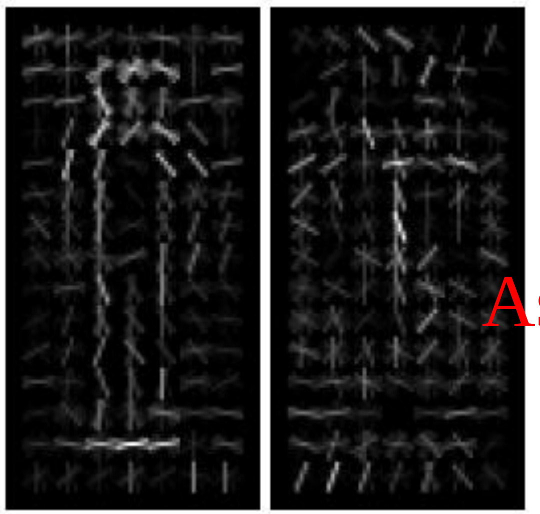
features = 15 x 7 x 9 x 4 = 3780

cells

normalizations by neighboring cells

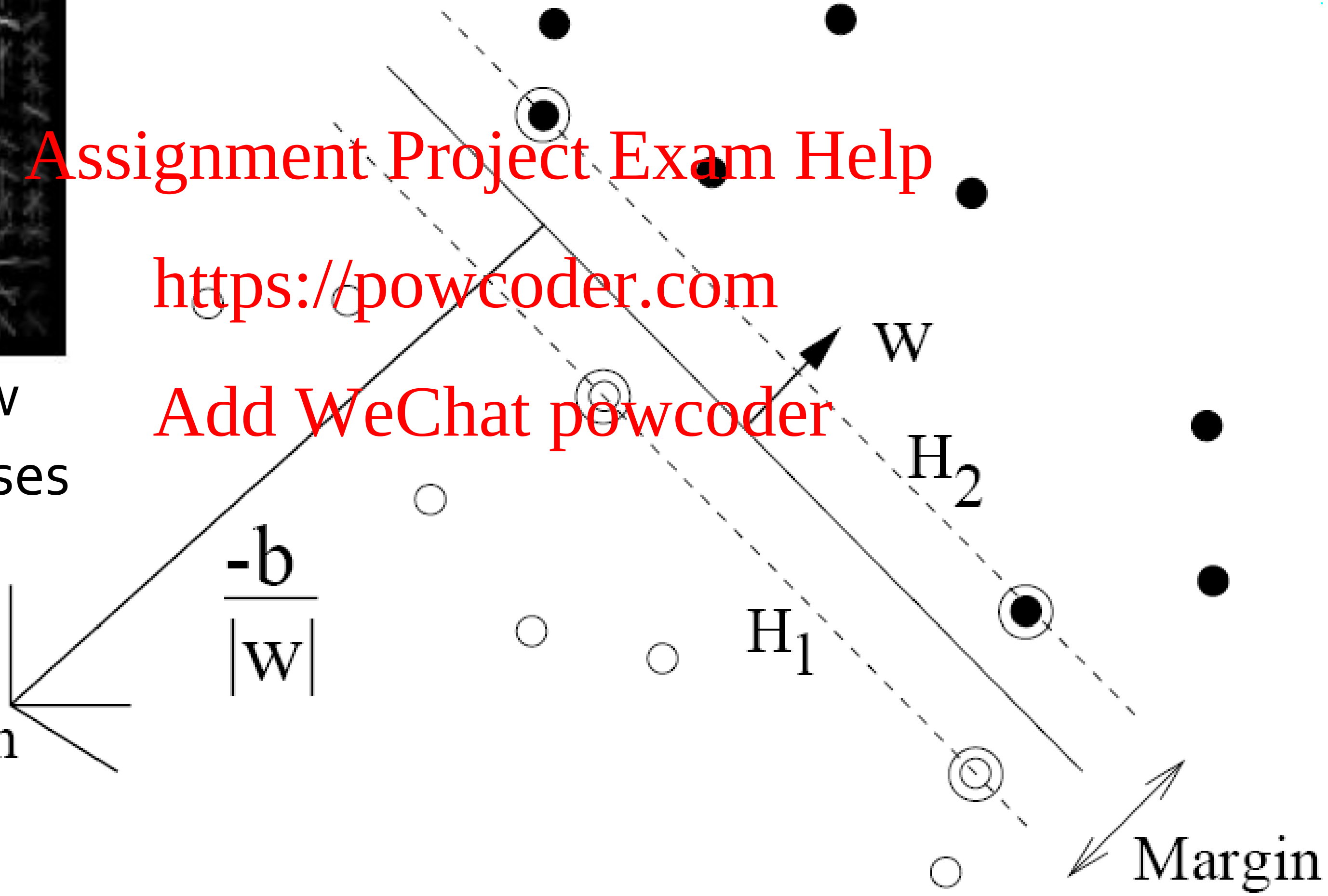
Slides by Pete Barnum

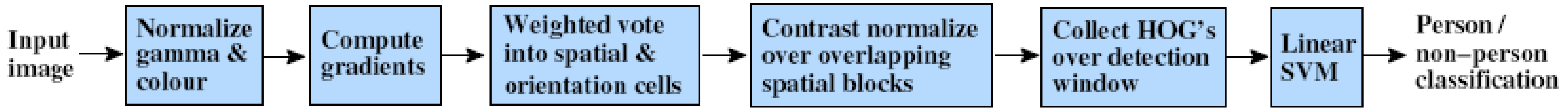




pos w neg w

Training data classes





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

$$sign(0.16) = 1$$

$=>$ pedestrian

Strengths/Weaknesses of Statistical Template Approach

- **Strengths**
  - Works very well for non-deformable objects with canonical orientations: faces, cars, p edestrians
  - Fast detection

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

EBU7240

Computer Vision

- Detection2: Face detection -

Semester 1, 2021

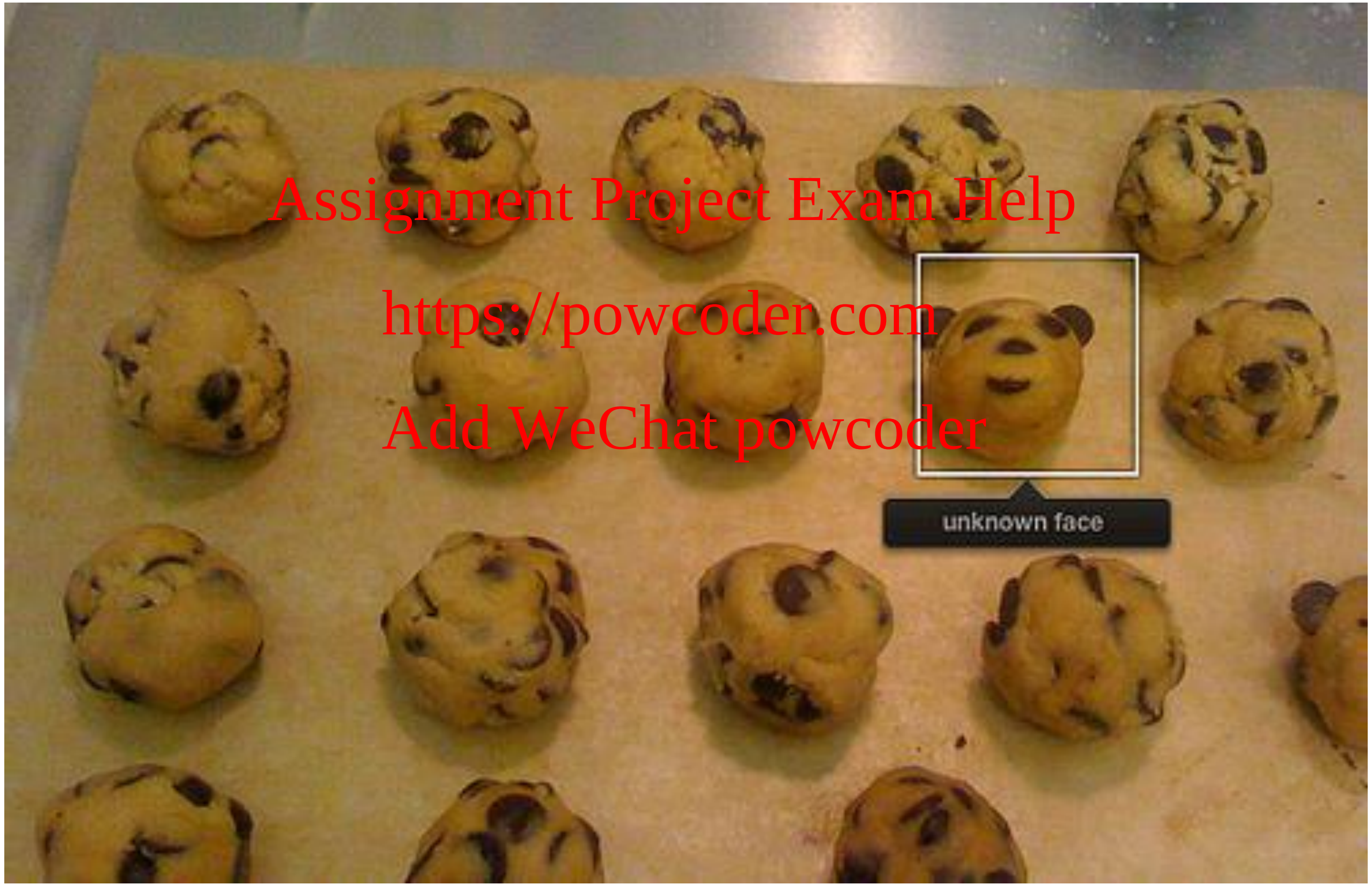
Changjae Oh

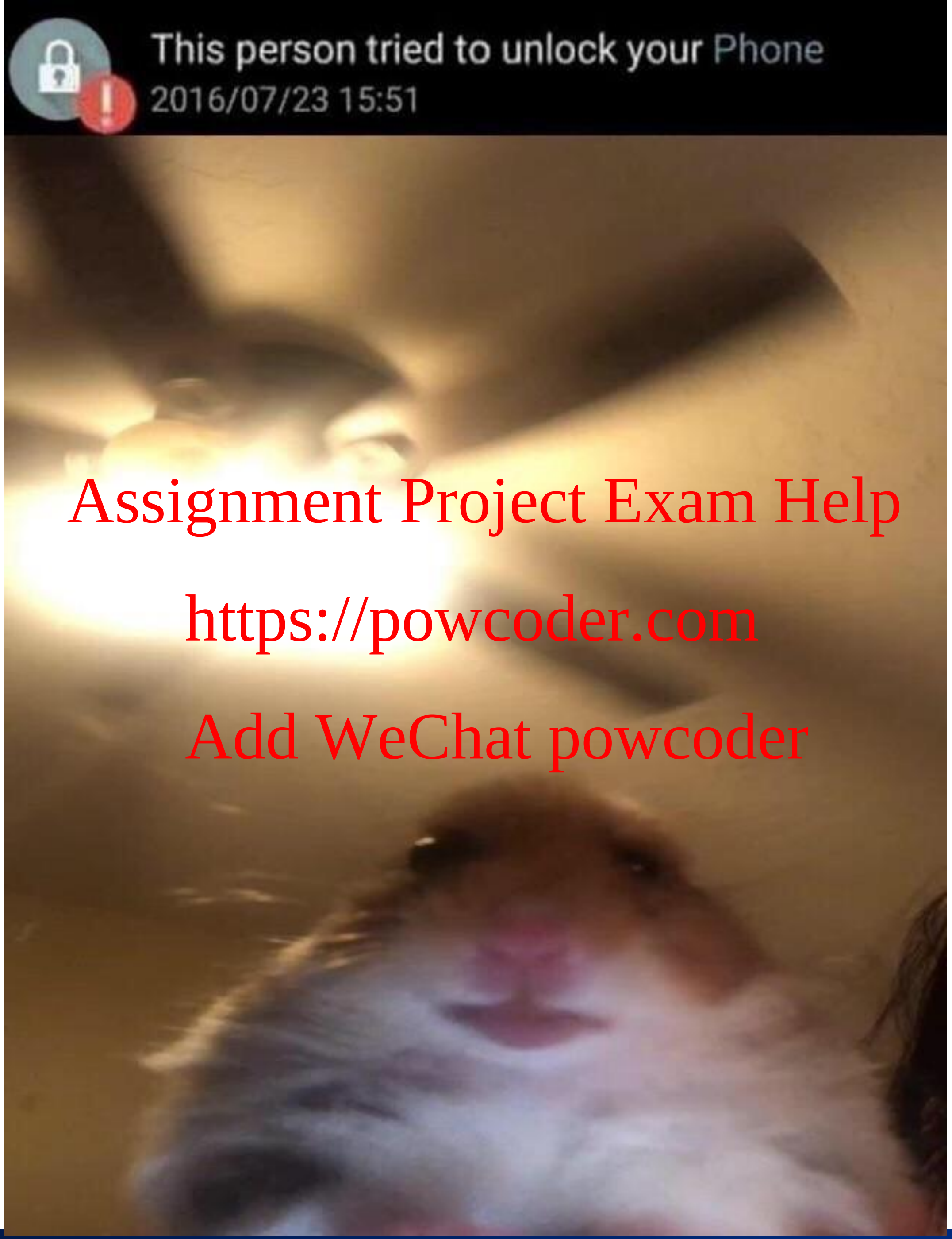
Outline

- **Overview**
- **Viola-Jones (face detection)**
  - Boosting for learning
  - Decision trees

Consumer application: Apple iPhoto

- [Things iPhoto thinks are faces]




This person tried to unlock your Phone
2016/07/23 15:51

Challenges of face detection

- **Sliding window = tens of thousands of location/scale evaluations**
  - One megapixel image has ~10^6 pixels, and a comparable number of candidate face lo cations

- **Faces are rare: 0–10 per image**

  - For computational efficiency, spend as little time as possible on the non-face windows

- **For 1 Mpix, to avoid having a false positive in every image, our false positive rate has to be less than 10^{-6}**

The Viola/Jones Face Detector

- **A seminal approach to real-time object detection.**
- **Training is slow, but detection is very fast**

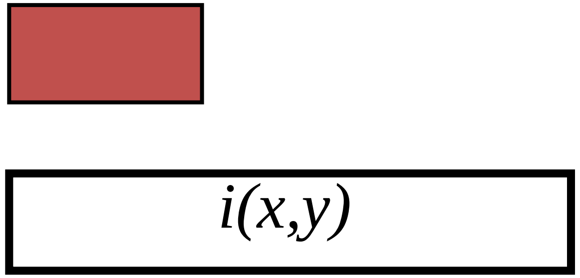
- **Key ideas:**
  1. Integral images for fast feature evaluation
  2. Boosting for feature selection
  3. Attentional cascade for fast non-face window rejection

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

1. Integral images for fast feature evaluation

- The integral image computes a value at each pixel (x,y) that is the sum of all pixel values above and to the left of (x,y), inclusive.

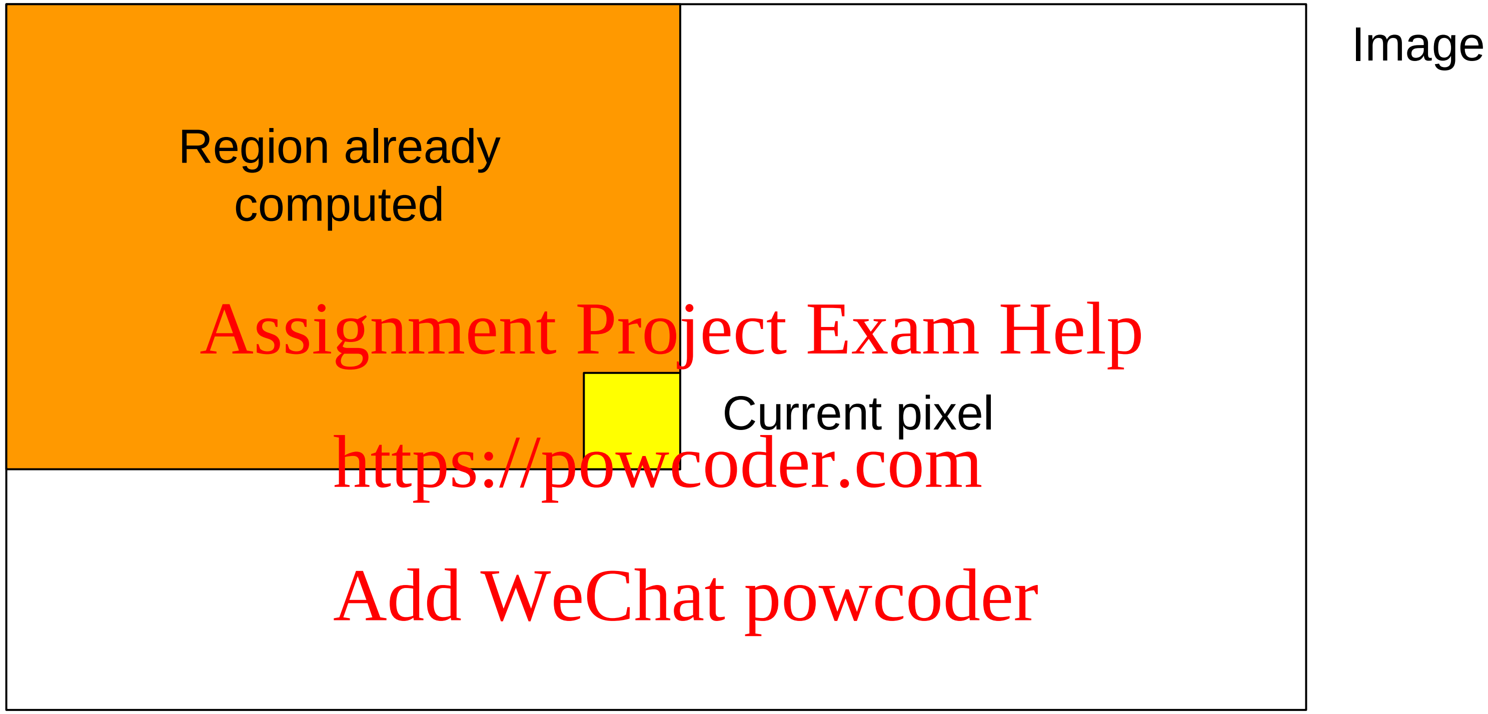


- This can quickly be computed in one pass through the image.

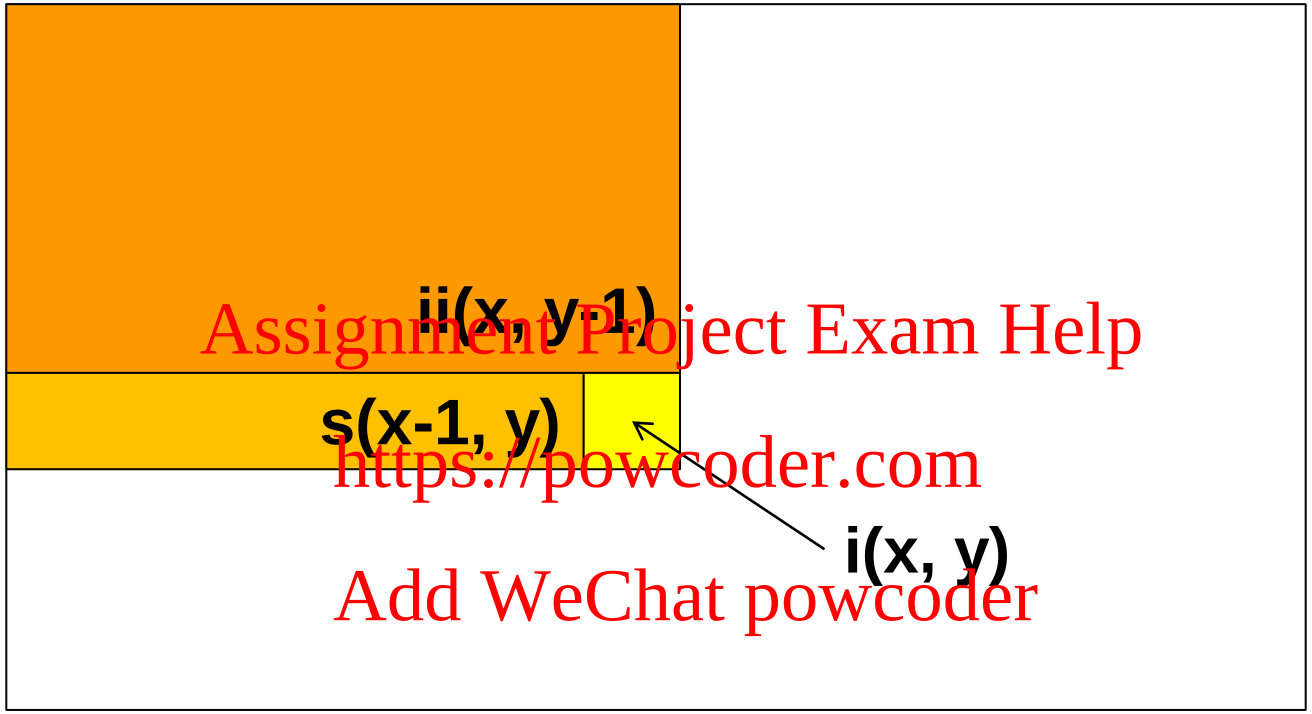
$$I_{\Sigma}(x,y) = \sum_{\substack{x' \le x \\ y' \le y}} i(x',y')$$

- 'Summed area table'

Computing the integral image



Computing the integral image

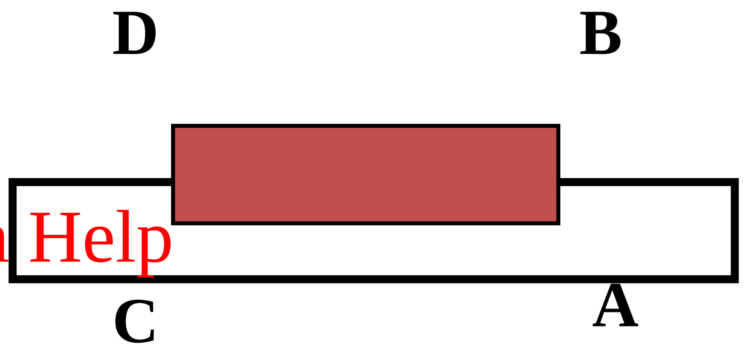


- **Cumulative row sum: $s(x, y) = s(x-1, y) + i(x, y)$**
- **Integral image: $ii(x, y) = ii(x, y-1) + s(x, y)$**

```
Python: ii = np.cumsum(i)
```

Computing sum within a rectangle

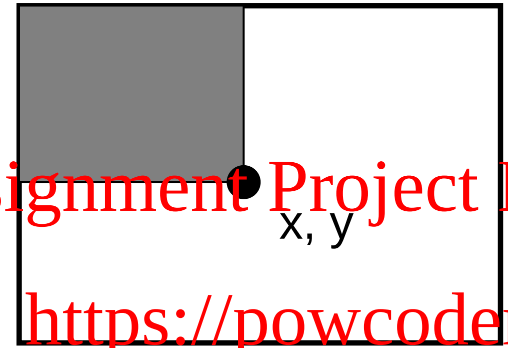
- Let A,B,C,D be the values of the integra l image at the corners of a rectangle
- The sum of original image values within the rectangle can be computed as:
  - sum = A – B – C + D



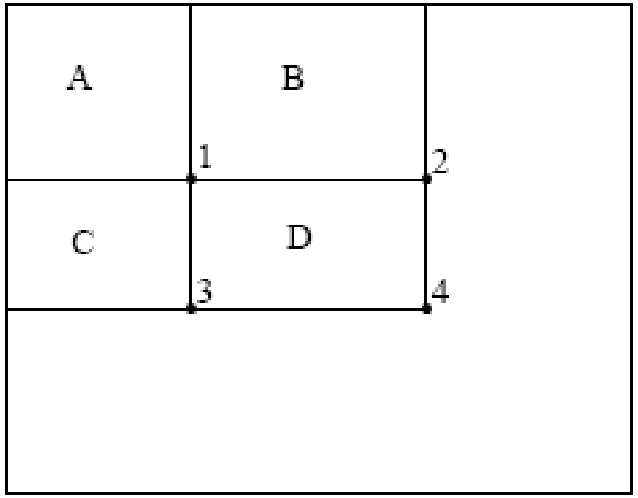
Only 3 additions are required for any size of rectangle!

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)

Integral Images- example

- **Find the integral image of the figure below and computer the sum of pixels in the grey region based on the integral image.**

1	2	3	4	5
1	1	1	2	6
1	6	0	5	2
1	3	6	5	1
1	2	0	1	3

1	3	6	10	15
2	5	9	15	26
3	12	16	27	40
4	16	26	42	56
5	19	29	46	63

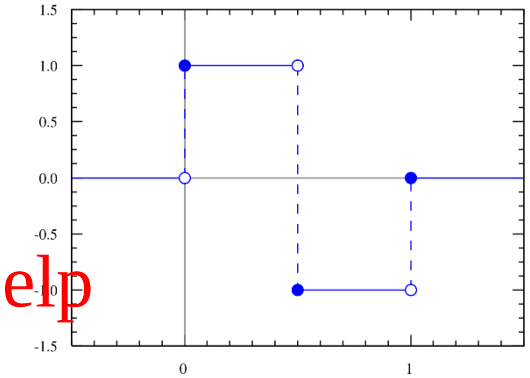
ii(4) - ii(2) - ii(3) + ii(1)

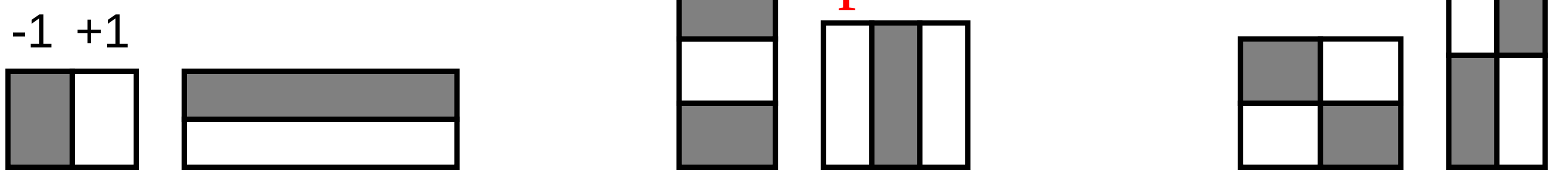
= 42 – 10 – 4 + 1

Features that are fast to compute

- **“Haar-like features”**
  - Differences of sums of intensity
  - Computed at different positions and scales within sliding window

Haar wavelet

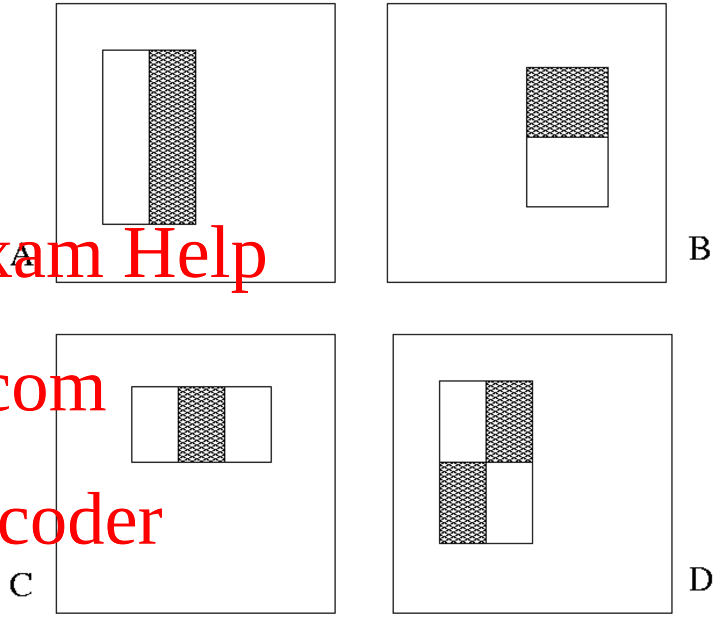




Two-rectangle features   Three-rectangle features   Etc.

Image Features

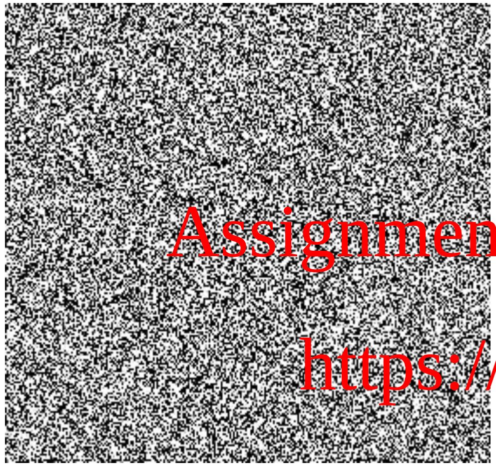
"Rectangle filters"



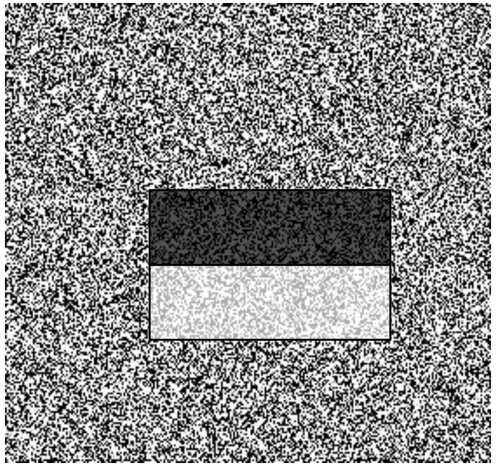
$$\textit{Value} = \sum(\textit{pixels in white area}) - \sum(\textit{pixels in black area})$$

Example

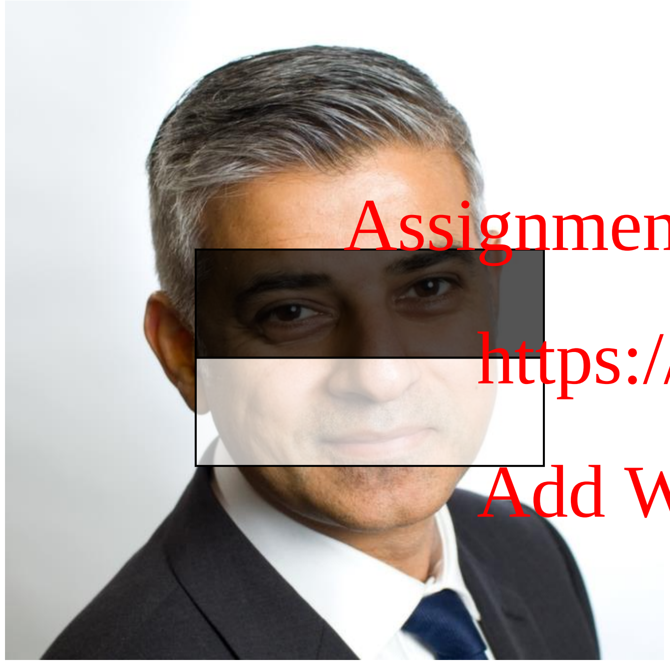
Source

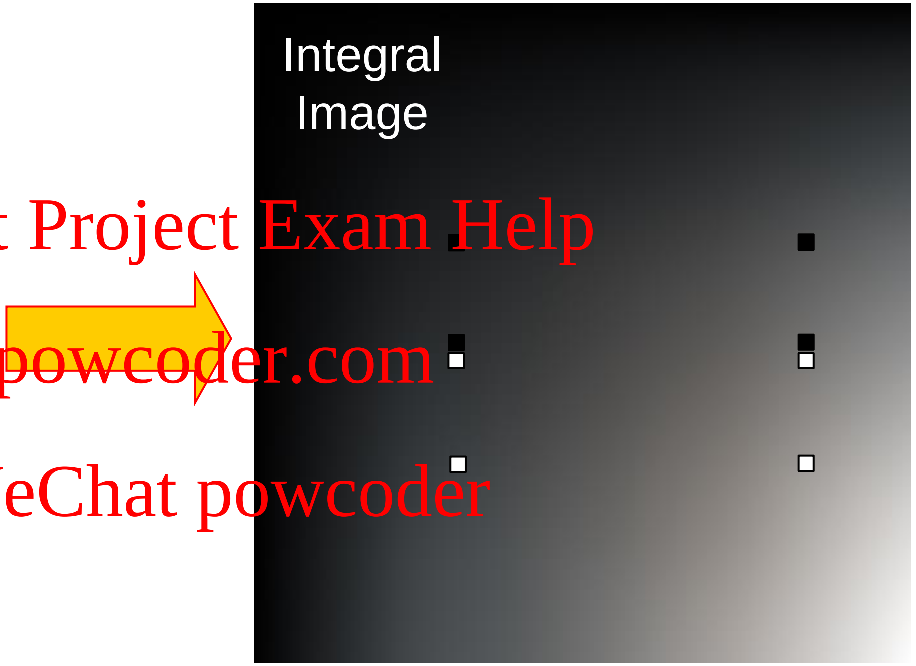


Result



Computing a rectangle feature



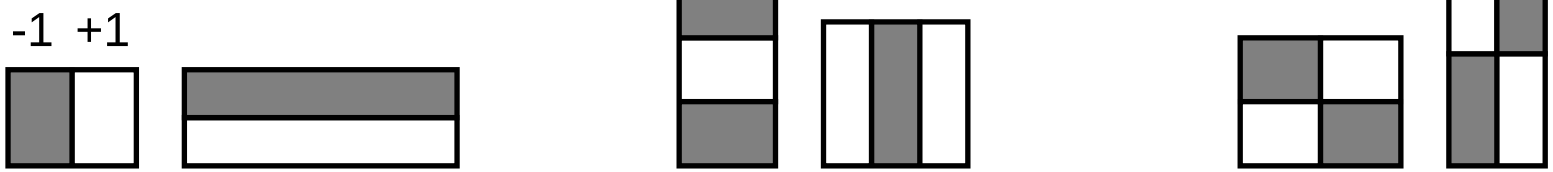


But these features are rubbish…!

Yes, individually they are ‘weak classifiers’

•*Jargon: ‘feature’ and ‘classifier’ are used interchangeably here. Also with ‘learner’.*

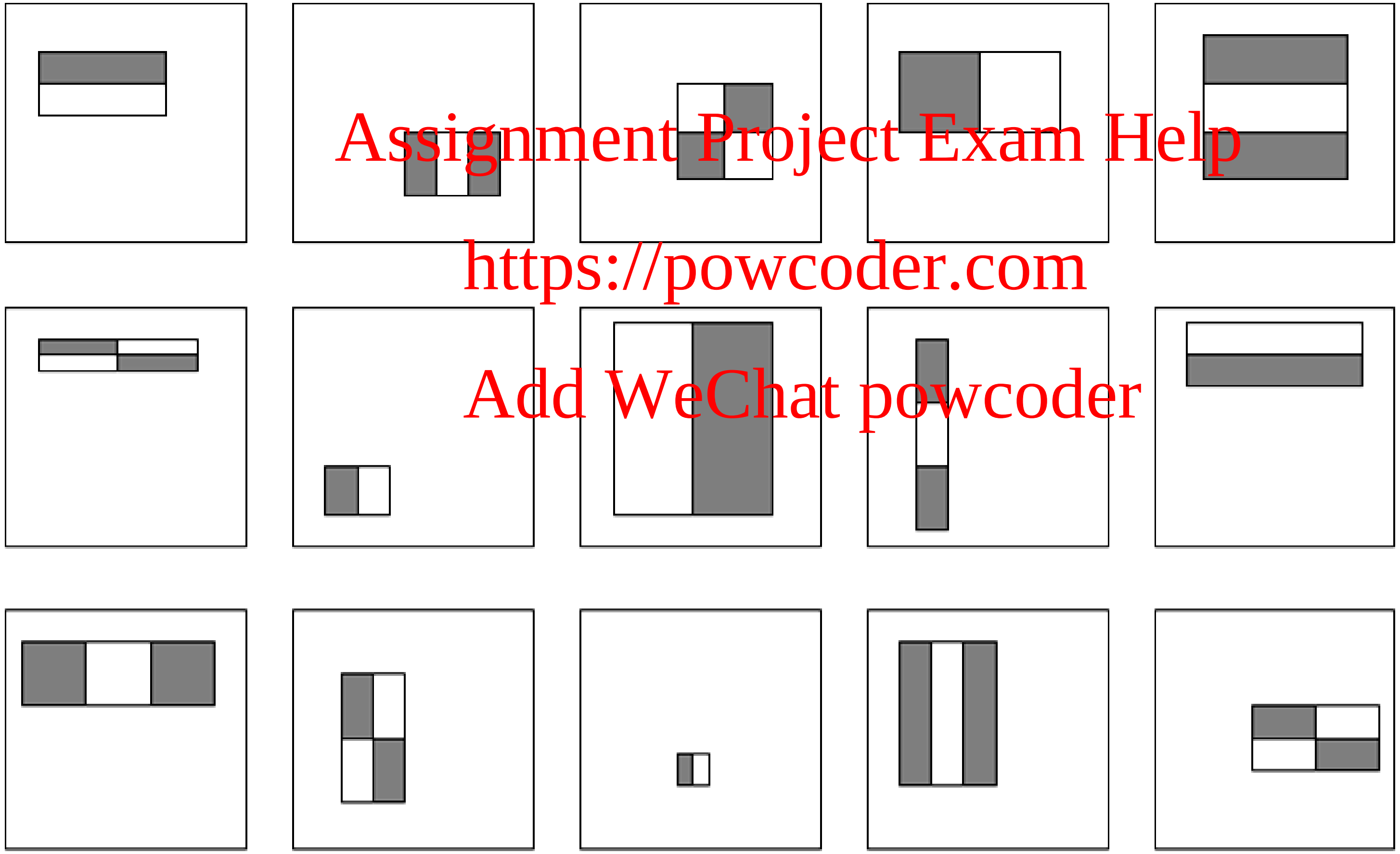
But, what if we combine *thousands* of them…



Two-rectangle features   Three-rectangle features   Etc.

How many features are there?

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!



How many features are there?

- For a 24x24 detection region, the number of possible rectangle features is ~160,000!

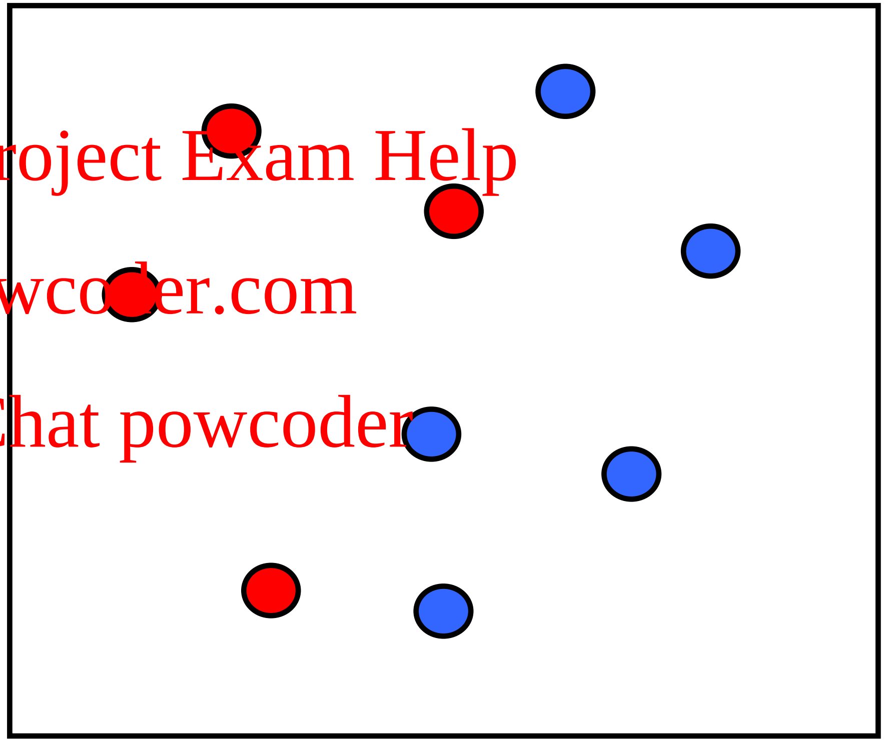
- At test time, it is impractical to evaluate the entire feature set.

- Can we learn a 'strong classifier' using just a small subset of all possible f eatures?

2. *Boosting* for feature selection

Initially, weight each training example equally.

Weight = size of point



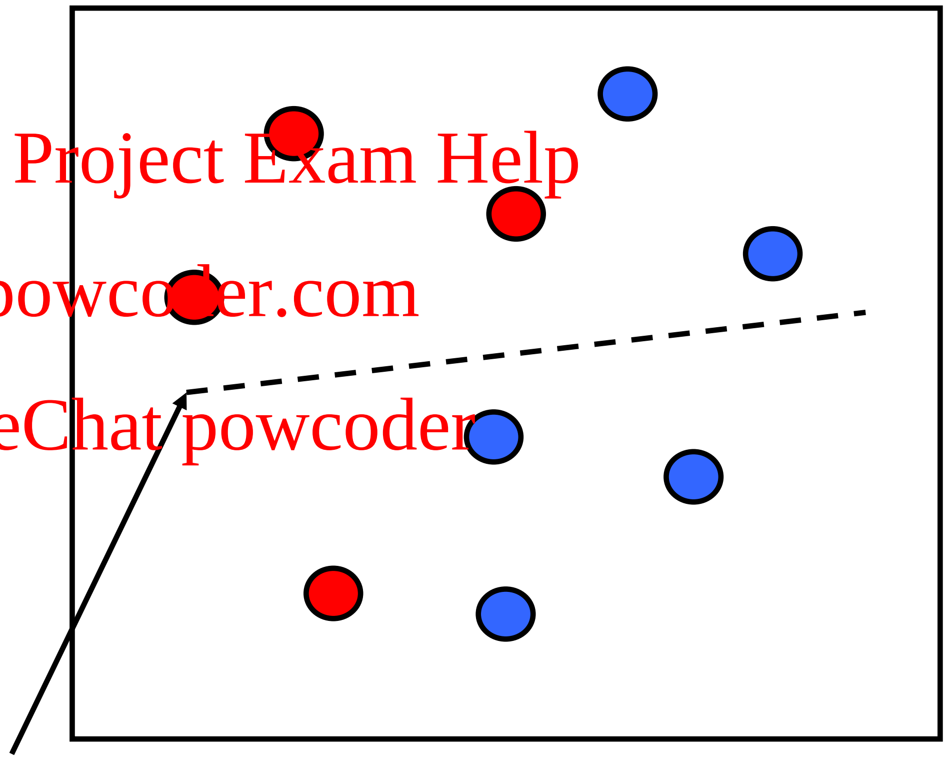
Slide credit: Paul Viola

2. *Boosting* for feature selection

In each boosting round:

Find the weak classifier, trained for each feature, that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.



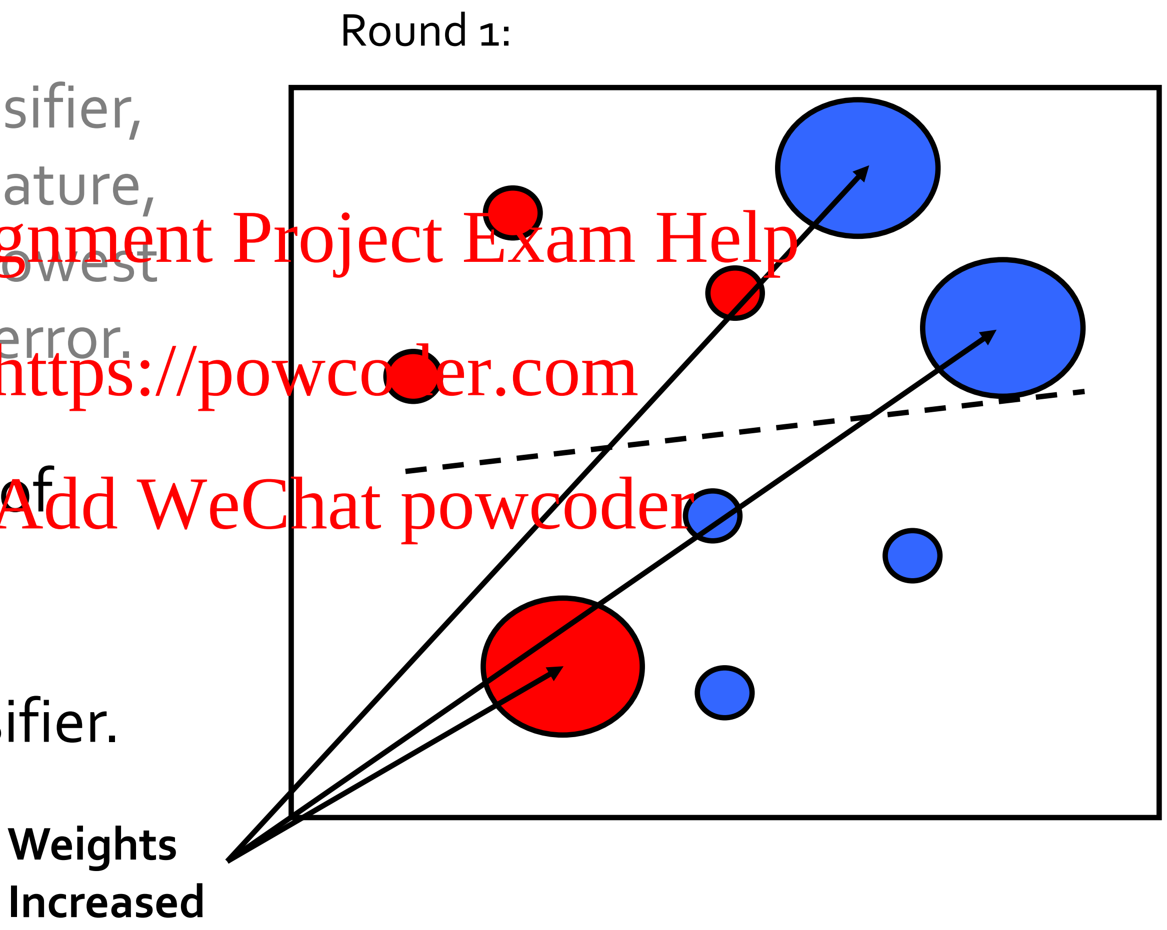
Slide credit: Paul Viola

Boosting illustration

In each boosting round:

Find the weak classifier, trained for each feature, that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.

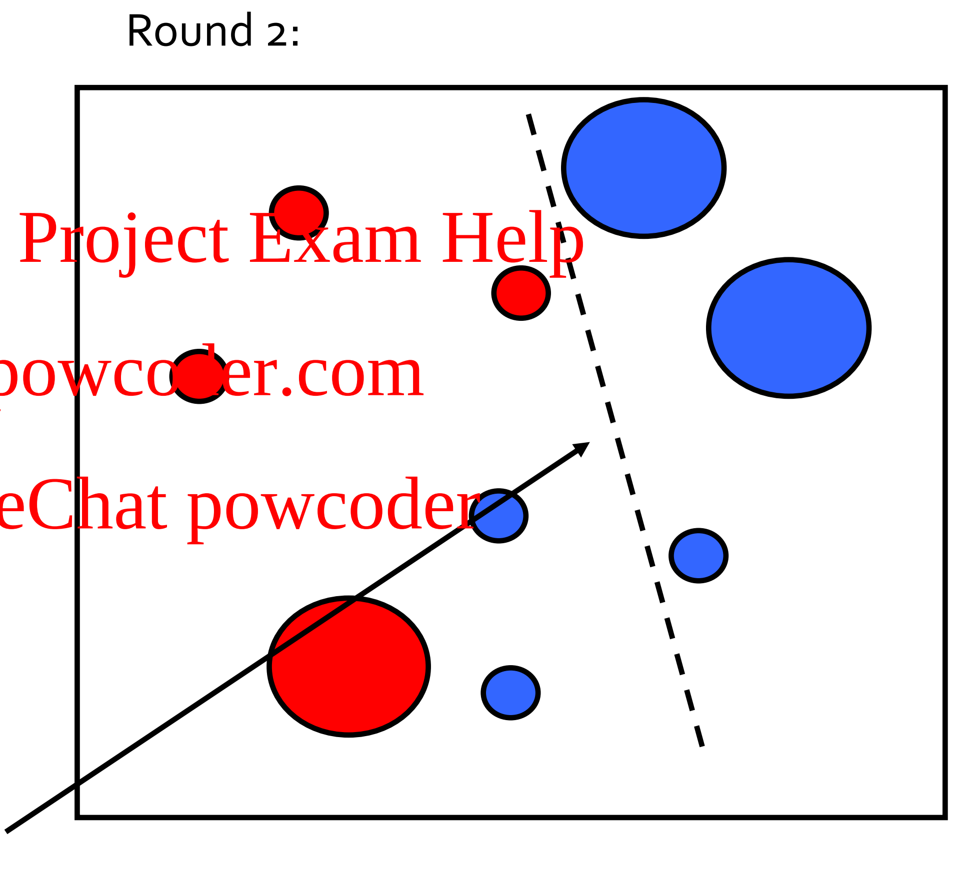


Boosting illustration

In each boosting round:

Find the weak classifier, trained for each feature, that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.



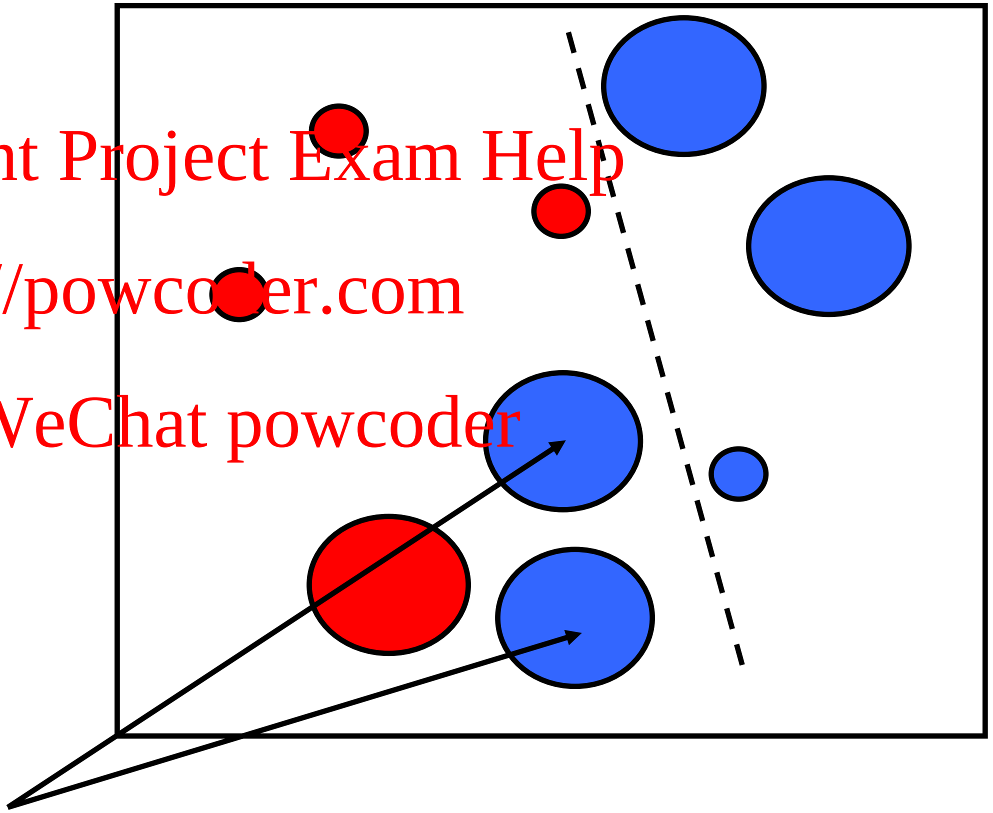
Boosting illustration

In each boosting round:

Find the weak classifier, trained for each feature, that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.

Round 2:



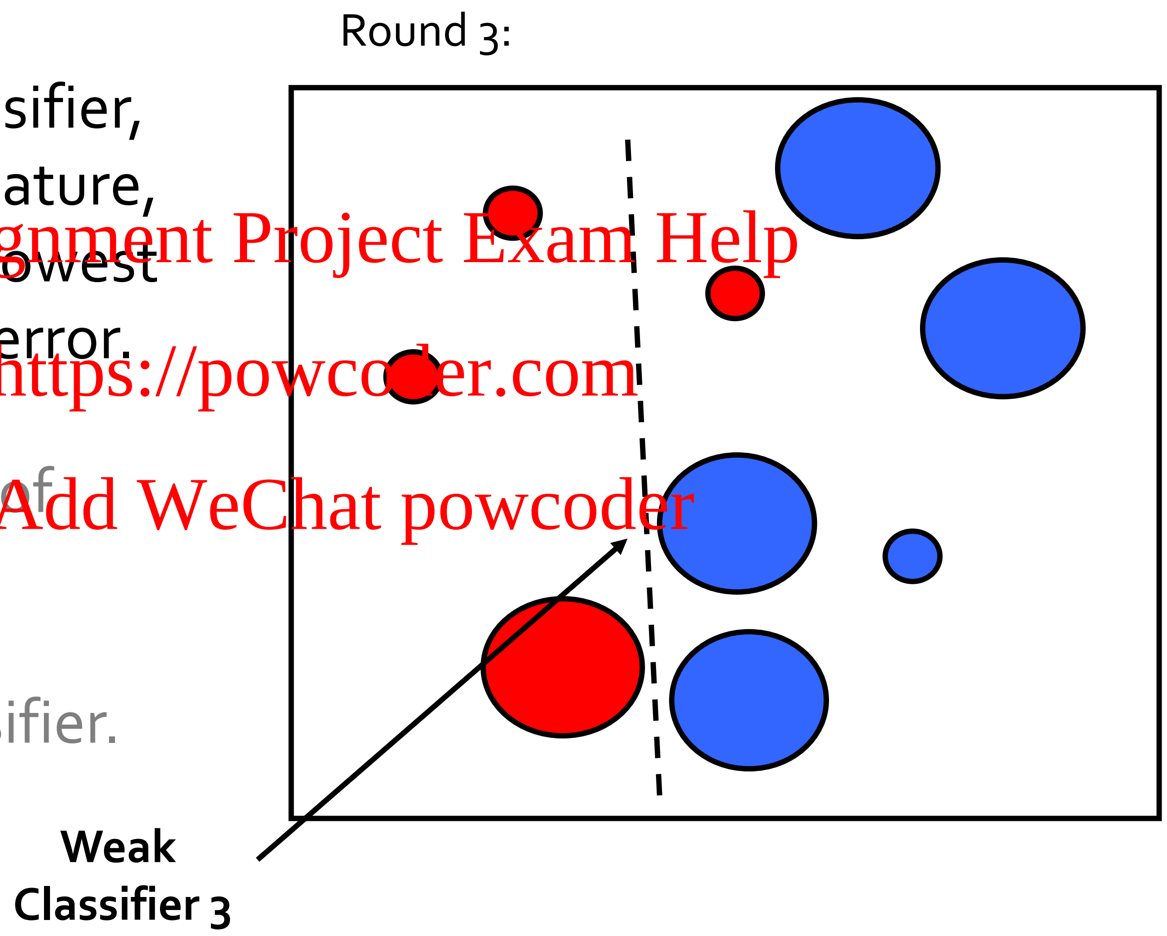
Weights Increased

Boosting illustration

In each boosting round:

Find the weak classifier, trained for each feature, that achieves the lowest *weighted* training error.

Raise the weights of training examples misclassified by current weak classifier.

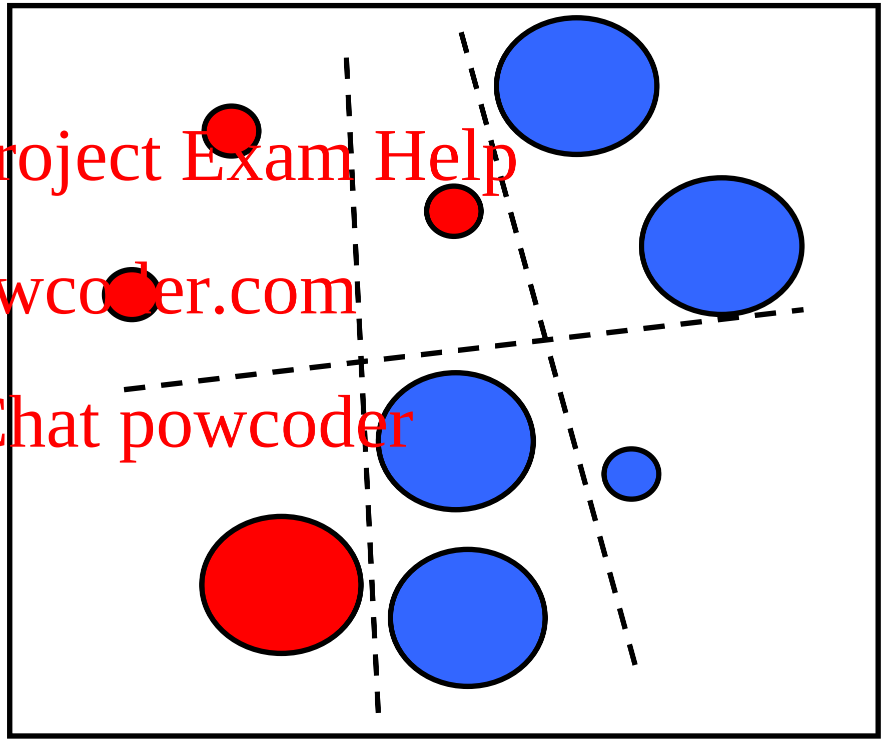


Boosting illustration

Compute final classifier as linear combination of all weak classifier.

Weight of each classifier is directly proportional to its accuracy.

Round 3:



Exact formulas for re-weighting and combining weak learners depend on the particular boosting scheme (e.g., AdaBoost).

Y. Freund and R. Schapire, A short introduction to boosting, *Journal of Japanese Society for Artificial Intelligence*, 14(5):771-780, September, 1999.

Boosting for face detection

- **First two features selected by boosting:**
  - The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks.
  - The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

The two features are shown in the top row and then overlayed on a typical training face in the bottom row.

Feature selection with boosting

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

Final strong learner

Weak learner

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

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' = black or white region flip → $s_j \in \pm 1$

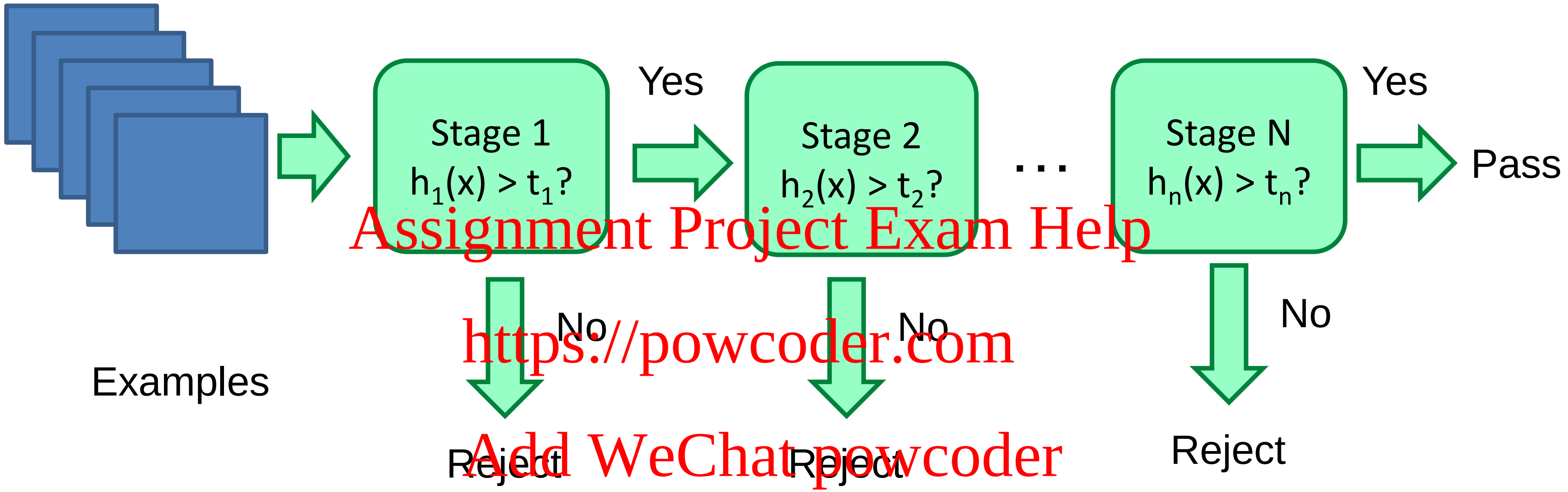
- Choose weak learner that minimizes error on the weighted training set, then reweight

Boosting: Pros and Cons

- **Advantages of boosting**
  - Integrates classifier training with feature selection
  - Complexity of training is linear instead of quadratic in the number of training examples
  - Flexibility in the choice of weak learners, boosting scheme
  - Testing is fast

- **Disadvantages**
  - Needs many training examples
  - Training is slow

Cascade for Fast Detection

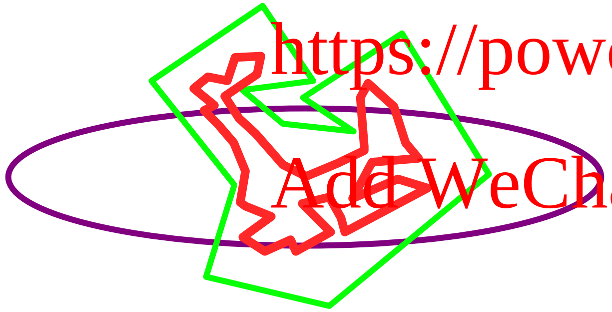


Fast classifiers early in cascade which reject many negative examples but detect almost all positive examples.

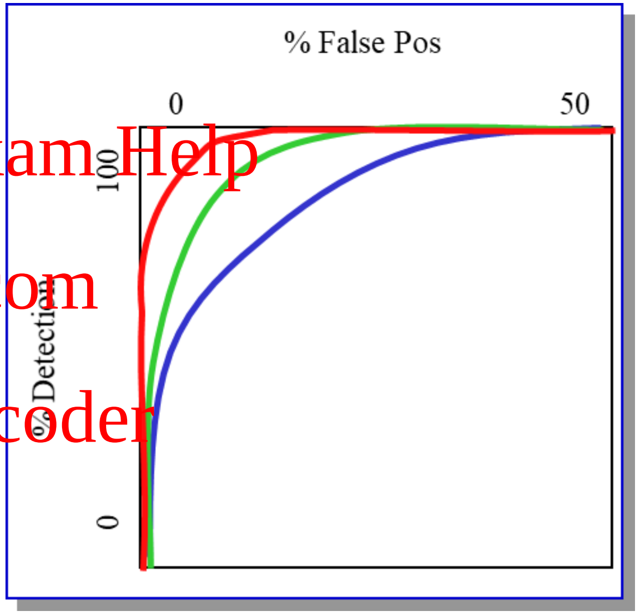
Slow classifiers later, but most examples don't get there.

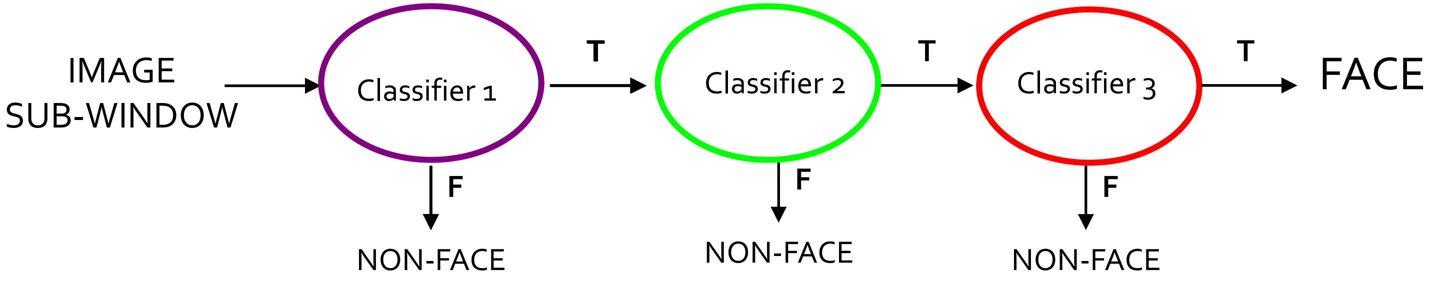
Attentional cascade

- **Chain classifiers that are progressively more complex and have lower fal se positive rates:**



Receiver operating characteristic

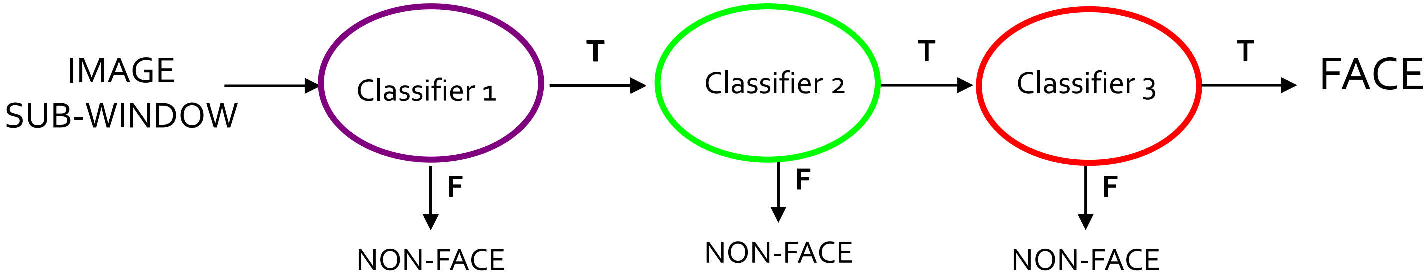




Attentional cascade

- The detection rate and the false positive rate of the cascade are found by multiplying the respective rates of the individual stages

- A detection rate of 0.9 and a false positive rate on the order of 10^{-6} can be achieved by a 10-stage cascade if each stage has a detection rate of 0.99 ($0.99^{10} \approx 0.9$) and a false positive rate of about 0.30 (0.310 ≈ 6×10^{-6})



Training the cascade

- Set target detection and false positive rates for each stage
- Keep adding features to the current stage until its target rates have been met
  - Need to lower boosting threshold to maximize detection (as opposed to minimizing total classification error)
  - Test on a validation set
- If the overall false positive rate is not low enough, then add another stage
- Use false positives from current stage as the negative training examples f or the next stage

The implemented system

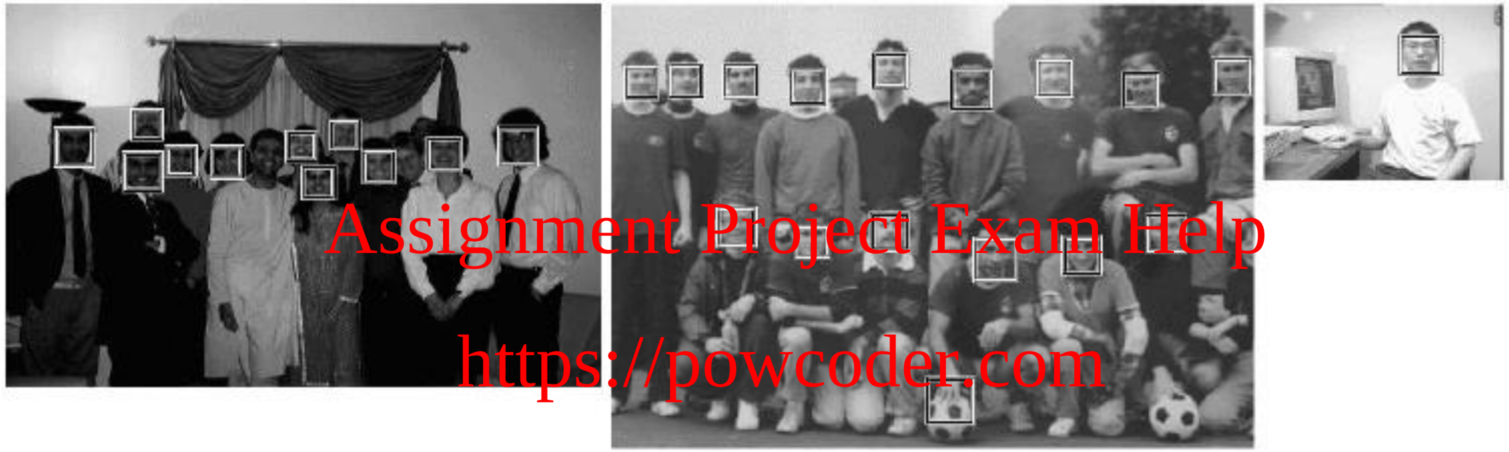
- **Training Data**
  - 5000 faces
    - All frontal, rescaled to 24x24 pixels
  - 300 million non-faces
    - 9500 non-face images
  - Faces are normalized
    - Scale, translation
- **Many variations**
  - Across individuals
  - Illumination
  - Pose

Viola-Jones details

- **38 stages with 1, 10, 25, 50 … features**
  - 6061 total used out of 180K candidates
  - 10 features evaluated on average
- **Training Examples**
  - 4916 positive examples
  - 10000 negative examples collected after each stage
- **Scanning**
  - Scale detector rather than image
  - Scale steps = 1.25 (factor between two consecutive scales)
  - Translation 1*scale (# pixels between two consecutive windows)
- **Non-max suppression: average coordinates of overlapping boxes**
- **Train 3 classifiers and take vote**

Viola-Jones Results

Speed = 15 FPS (in 2001)



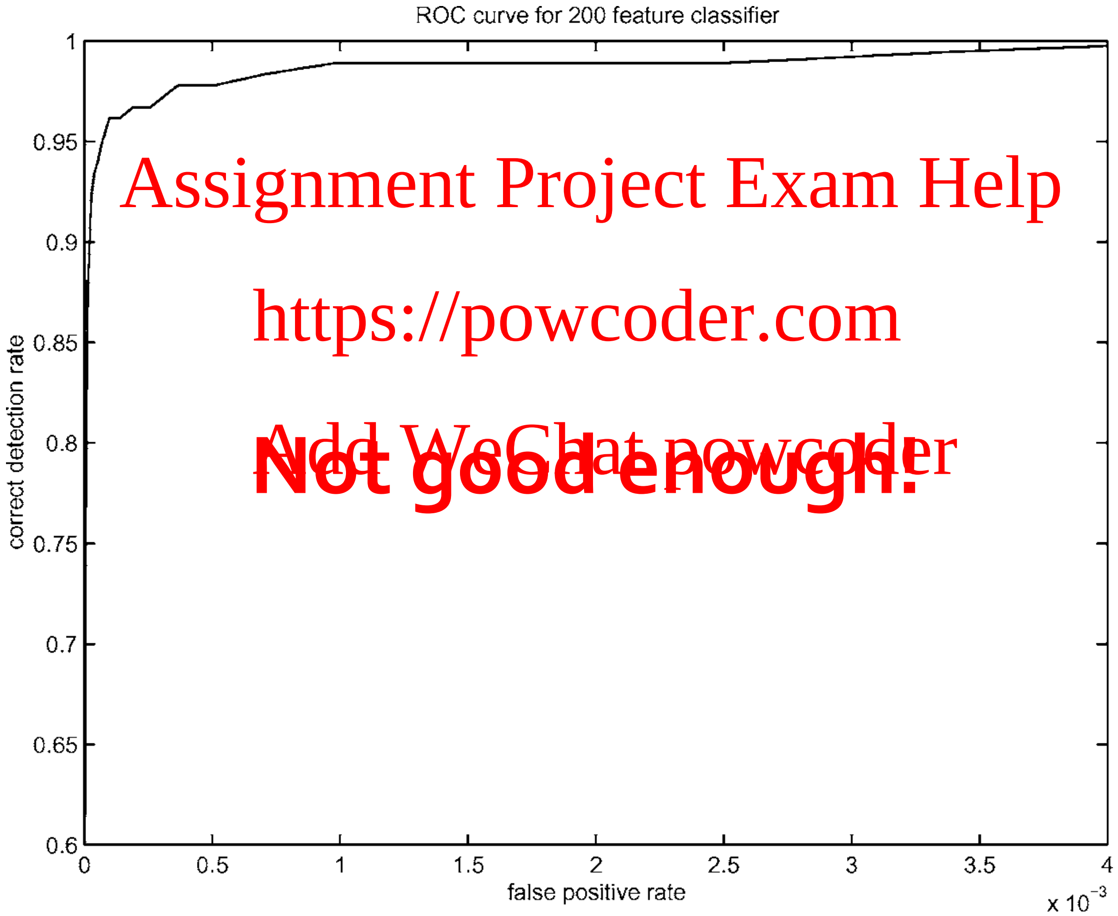
Detector \ False detections	10	31	50	65	78	95	167
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%
Viola-Jones (voting)	81.1%	89.7%	92.1%	93.1%	93.1%	93.2 %	93.7%
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Roth-Yang-Ahuja	-	-	-	-	(94.8%)	-	-

MIT + CMU face dataset

Boosting for face detection

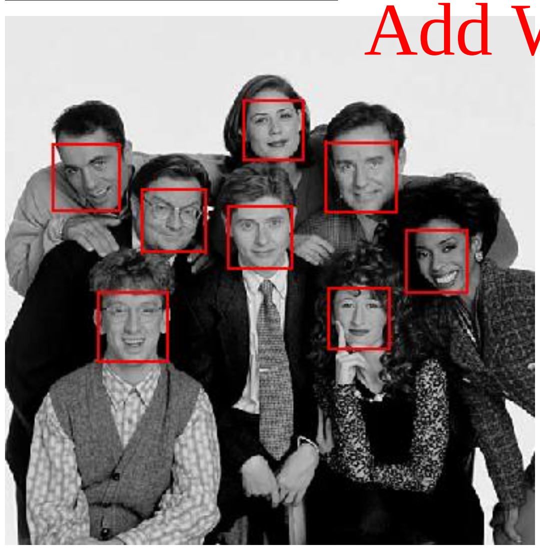
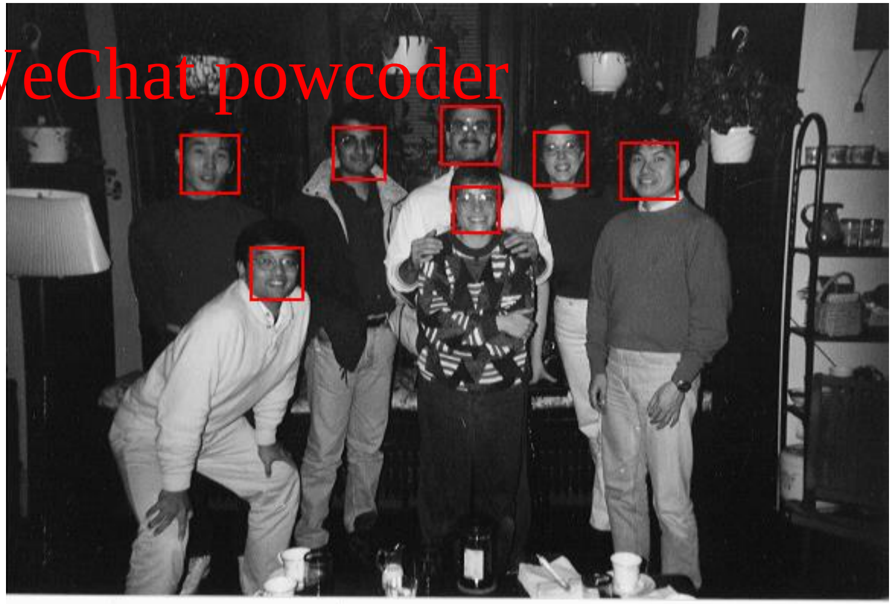
- A 200-feature classifier can yield 95% detection rat e and a false positive rate of 1 in 14084

True positive rate



Receiver operating characteristic (ROC) curve

Output of Face Detector on Test Images

Summary: Viola/Jones detector

- Rectangle features
- Integral images for fast computation
- Boosting for feature selection
- Attentional cascade for fast rejection of negative windows