Object Detection using Haar-like Features

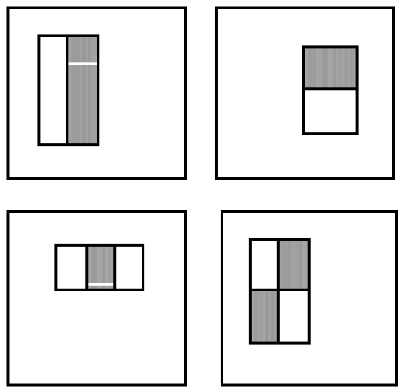
CS 395T: Visual Recognition and Search

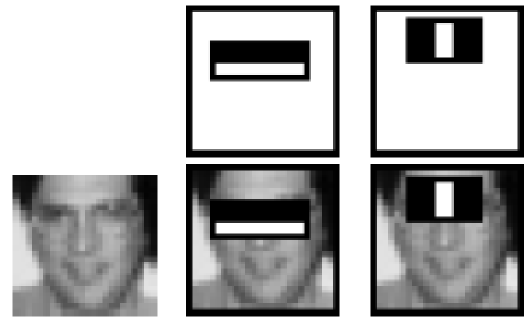
Harshdeep Singh

The Detector

- Using boosted cascades of Haar-like features
- Proposed by [Viola, Jones 2001]
- Implementation available in OpenCV

Haar-like features





- feature = w_1 x RecSum(r_1) + w_2 x RecSum(r_2)
- Weights can be positive or negative
- Weights are directly proportional to the area
- Calculated at every point and scale

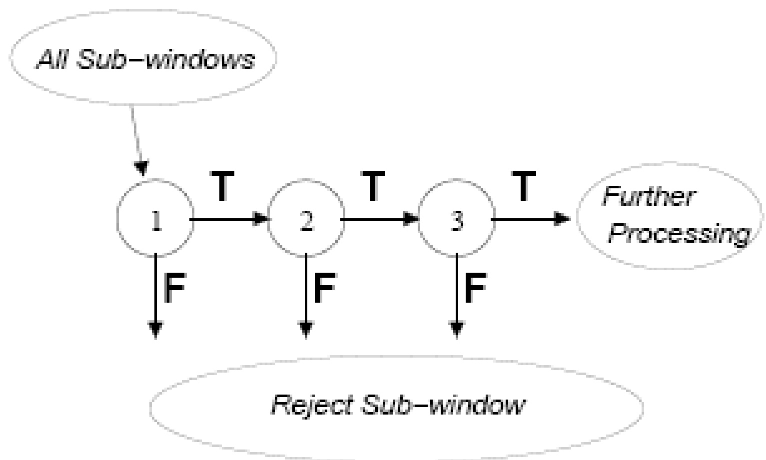
Weak Classifier

- A **weak classifier** (h(x, f, p, ϑ)) consists of
  - feature (f)
  - threshold (ϑ)
  - polarity (p), such that

$$h(x, f, p, \theta) = \begin{cases} 1 & if\ pf(x) < p\theta \\ 0 & otherwise \end{cases}$$

- Requirement
  - Should perform better than random chance

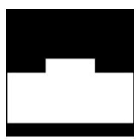
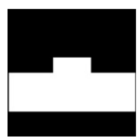
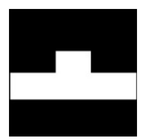
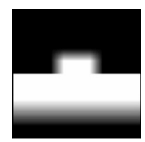
Attentional Cascade



- Initial stages have less features (faster computation)
- More time spent on evaluating more promising sub-windows

Cascade Creation - Walkthrough

- Input:
  - f = Maximum acceptable false positive rate per layer (0.5)
  - d = Minimum acceptable detection rate per layer (0.995)
  - F_{target} = Target overall false positive rate
    - Or maximum number of stages in the cascade
    - For nStages = 14, $F_{target} = f^{nStages} = 6.1$ e-5
  - P = Set of positive examples
    - 200 distorted versions of a synthetic image

  - N = Set of negative examples
    - 100 images from BACKGROUND_Google category of Caltech 101 dataset



Cascade Creation - Walkthrough

$F_0 = 1$
i = 0
while $F_i > F_{target}$ and i < nStages
    i = i + 1
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Cascade Creation - Walkthrough

F_i = False alarm rate of the cascade with i stages

$F_0 = 1$
i = 0
while $F_i > F_{target}$ and i < nStages
    i = i + 1
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Cascade Creation - Walkthrough

F_i = False alarm rate of the cascade with i stages

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i < nStages$
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i <$ nStages
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Weight for each

positive sample	0.5/m
negative sample	0.5/n

m – number of positive samples (200)
n – number of negative samples (100)

Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i <$ nStages
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Weight for each
    positive sample   0.5/m
    negative sample   0.5/n

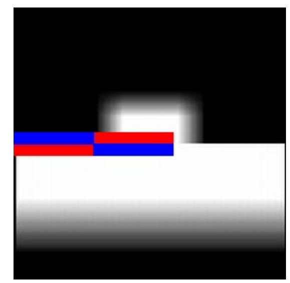
m – number of positive samples (200)
n – number of negative samples (100)

Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i <$ nStages
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

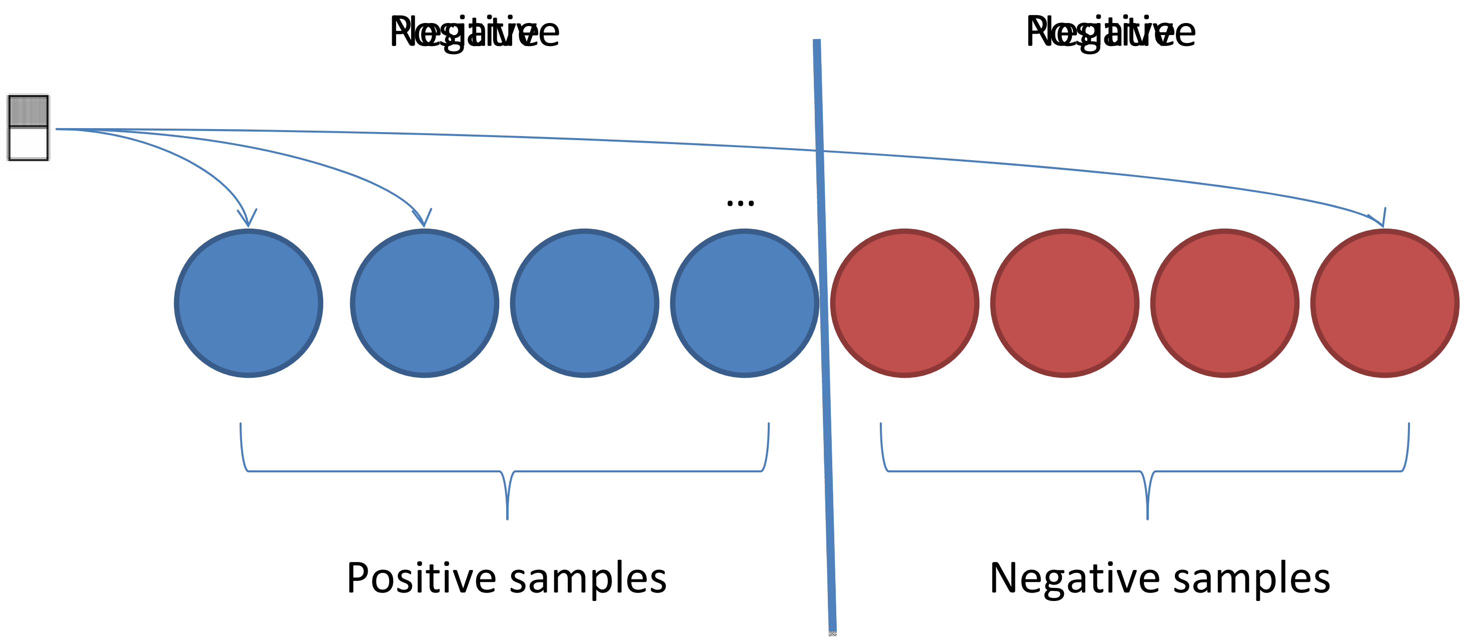
The one with minimum error

$$\epsilon_t = min_{f,p,\theta} \sum_i w_i \,|h(x_i, f, p, \theta) - y_i|$$



$\epsilon_t = 0.005$

Error minimization



T^+: Total sum of weights of positive examples
T^-: Total sum of weights of negative examples
S^+: Total sum of weights of positive examples below the current one
S^-: Total sum of weights of negative examples below the current one

$e_1 = S^+ + (T^- - S^-)$
$e_2 = S^- + (T^+ - S^+)$

$\mathbf{e = \min(e_1, e_2)}$

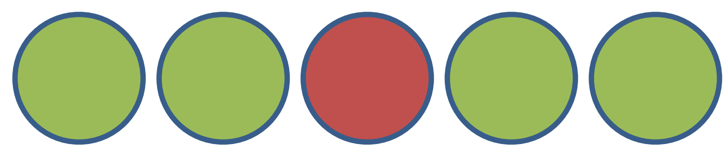
Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i <$ nStages
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

$$w_{t+1,i} = w_{t,i}\,\beta_t^{1-e_i}$$

$e_i = 0$, if example x_i is classified correctly
$e_i = 1$, otherwise

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$$



Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i < nStages$
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

f_i = number of negative samples that were detected by this stage/ total number of negative samples
= 1/100

Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i < nStages$
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

How far will you go to get down to f?

Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i < nStages$
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & otherwise \end{cases}$$

$$\alpha_t = \log \frac{1}{\beta_t} \qquad \beta_t = \frac{\epsilon_t}{1 - \epsilon_t}$$

Weight is inversely proportional to the training error

Paper
Decrease threshold until the classifier has a detection rate of at least d

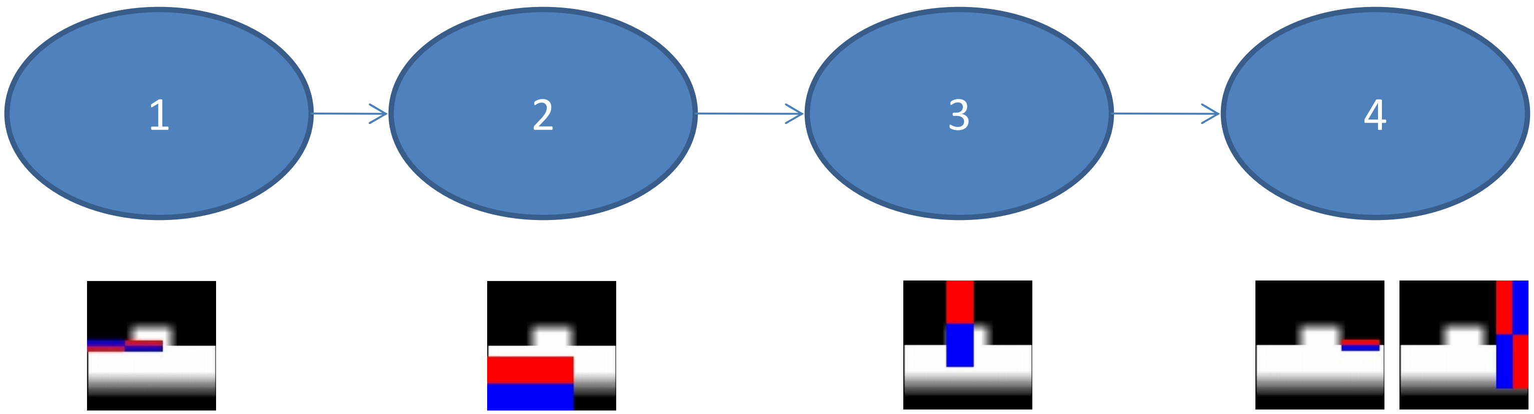
OpenCV
1.For each positive sample, find the weighted sum of all features
2.Sort these values
3.Set threshold = sorted_values[(1-d) * |P|]

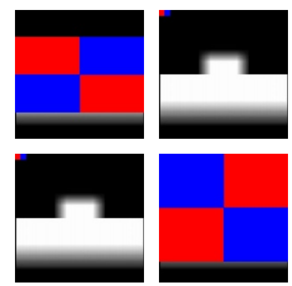
Cascade Creation - Walkthrough

$F_0 = 1$
$i = 0$
while $F_i > F_{target}$ and $i < nStages$
    $i = i + 1$
    Train Classifier for stage i
        Initialize Weights
        Normalize Weights
        Pick the (next) best weak classifier
        Update Weights
        Evaluate f_i
        if $f_i > f$
            go back to Normalize Weights
        Combine weak classifiers to form the strong stage classifier
        Evaluate F_i

Add another stage?

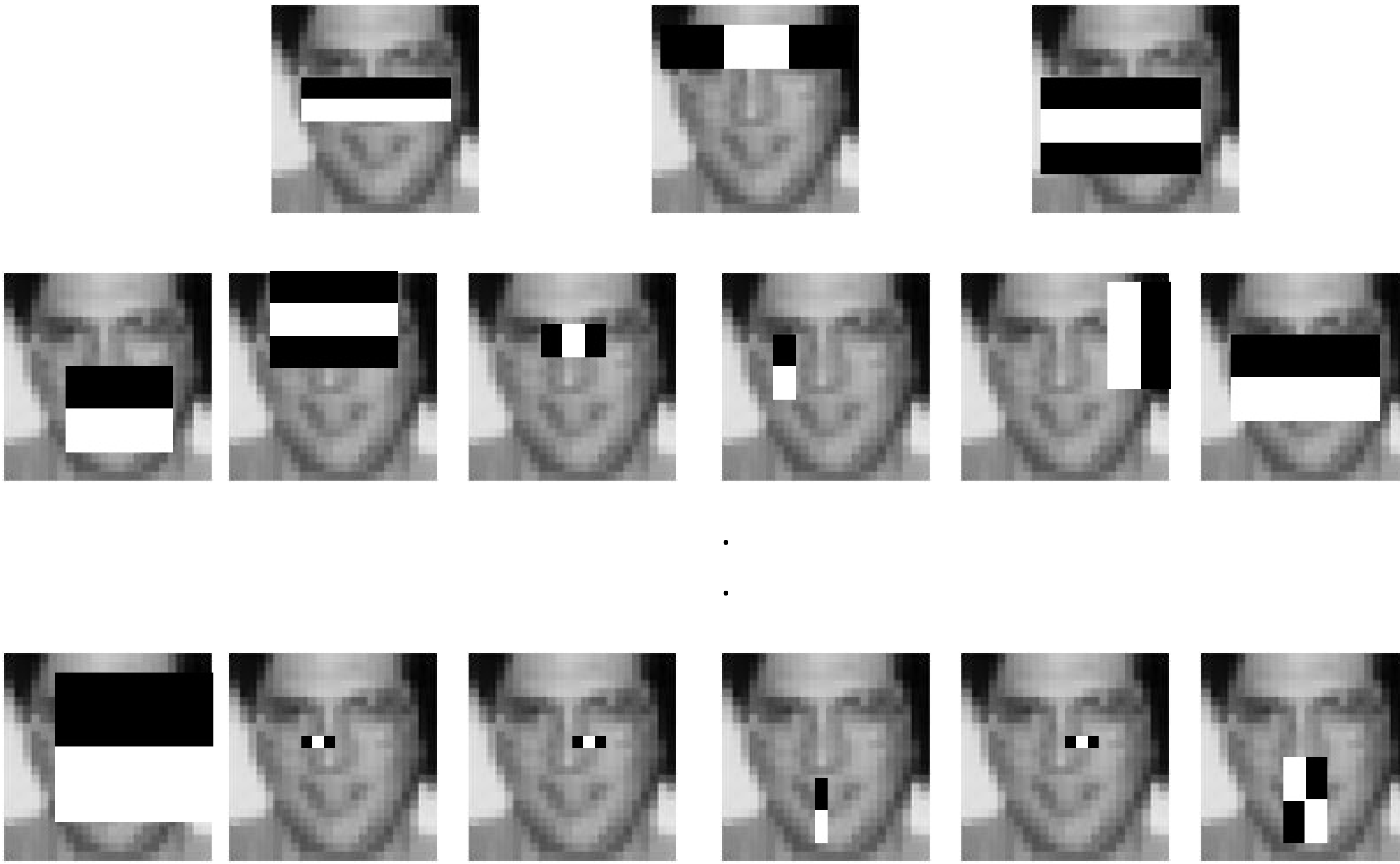
Resulting Cascade





If f (maximum false alarm rate) is increased from 0.5 to 0.7, a cascade with only the first two stages is created

Which features actually get selected?

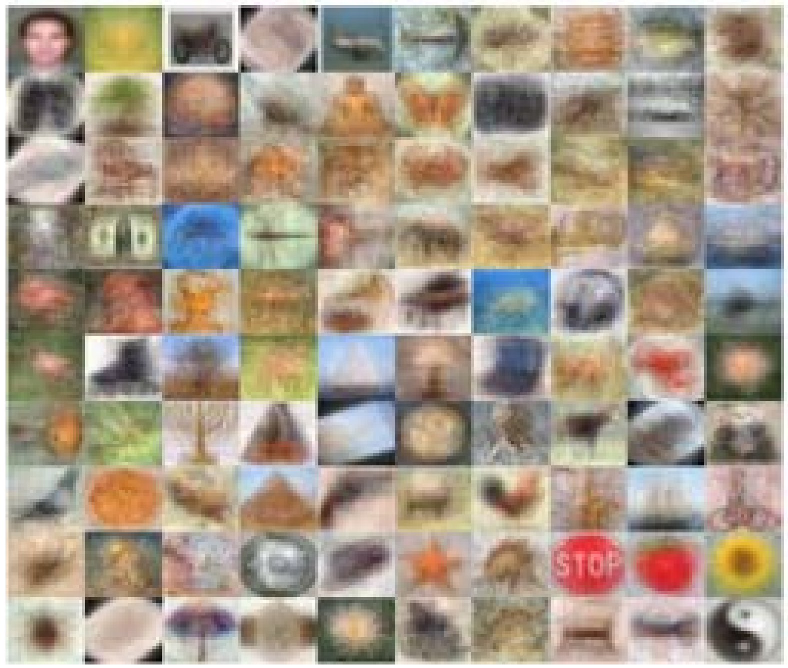


Stage 0

Stage 1 … 10 more

Stage 21 … 206 more

Other Objects?



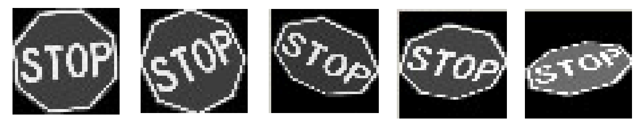
Caltech 101 dataset

“Most images have little or no clutter. The objects tend to be centered in each image. Most objects are presented in a stereotypical pose.”

Training

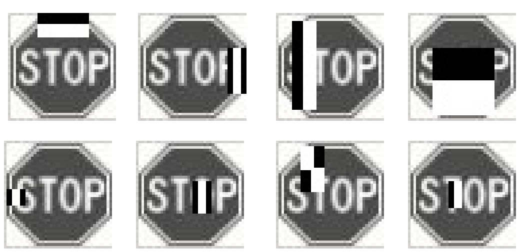
Hand label ROI in 40/64 images

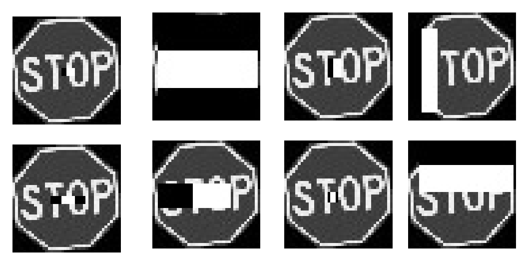
Generate 1000 random distortions of a representative image



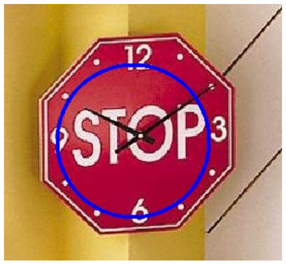
Negative samples taken from *BACKGROUND_Google* category of Caltech 101

Some features that get selected



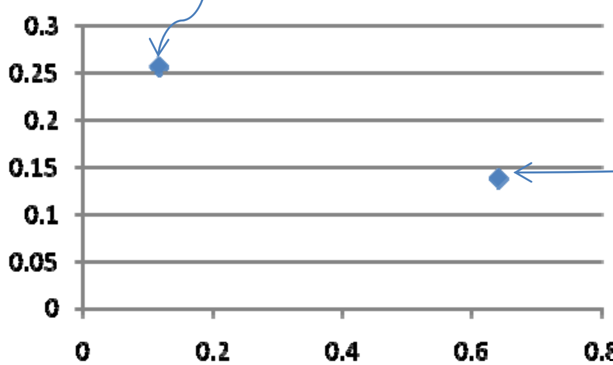


Performance



Hand label ROI

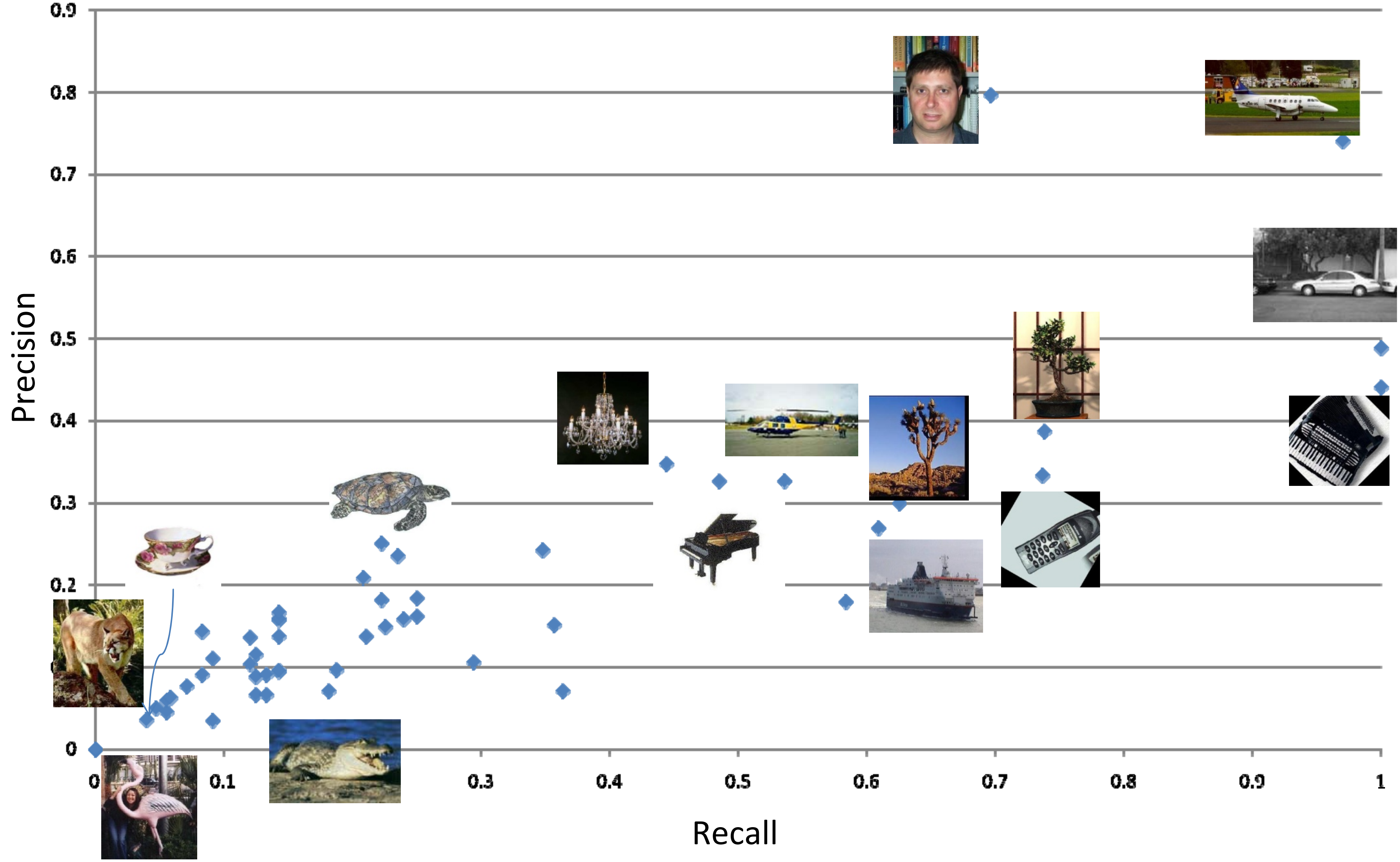
Hand label ROI



Random distortions

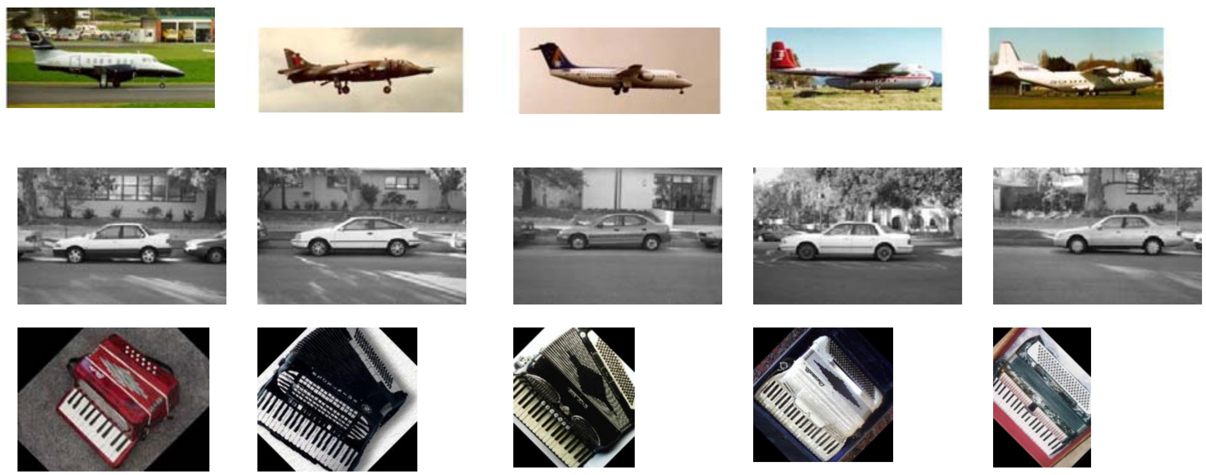
Random distortions

Other Categories

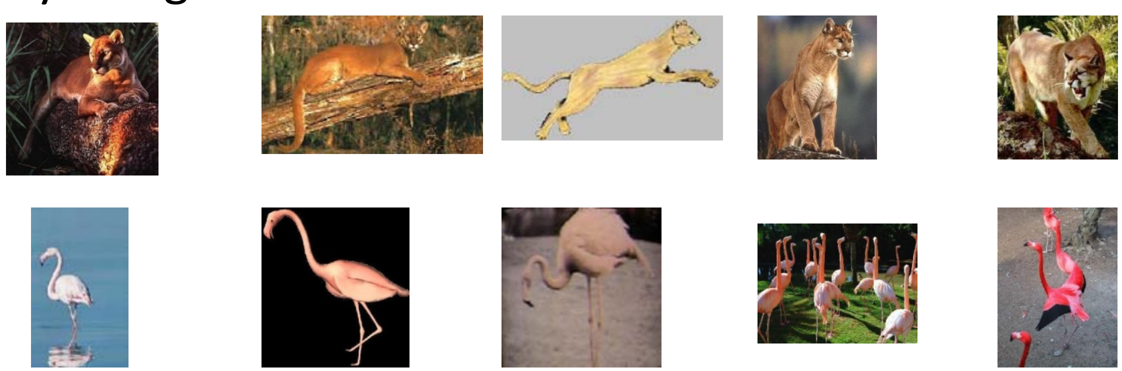


Variation in Training Images

High accuracy categories

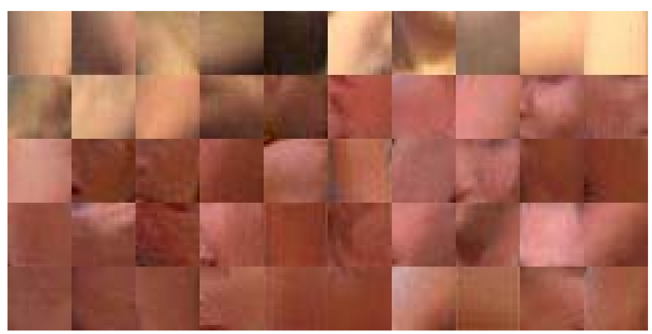


Low accuracy categories

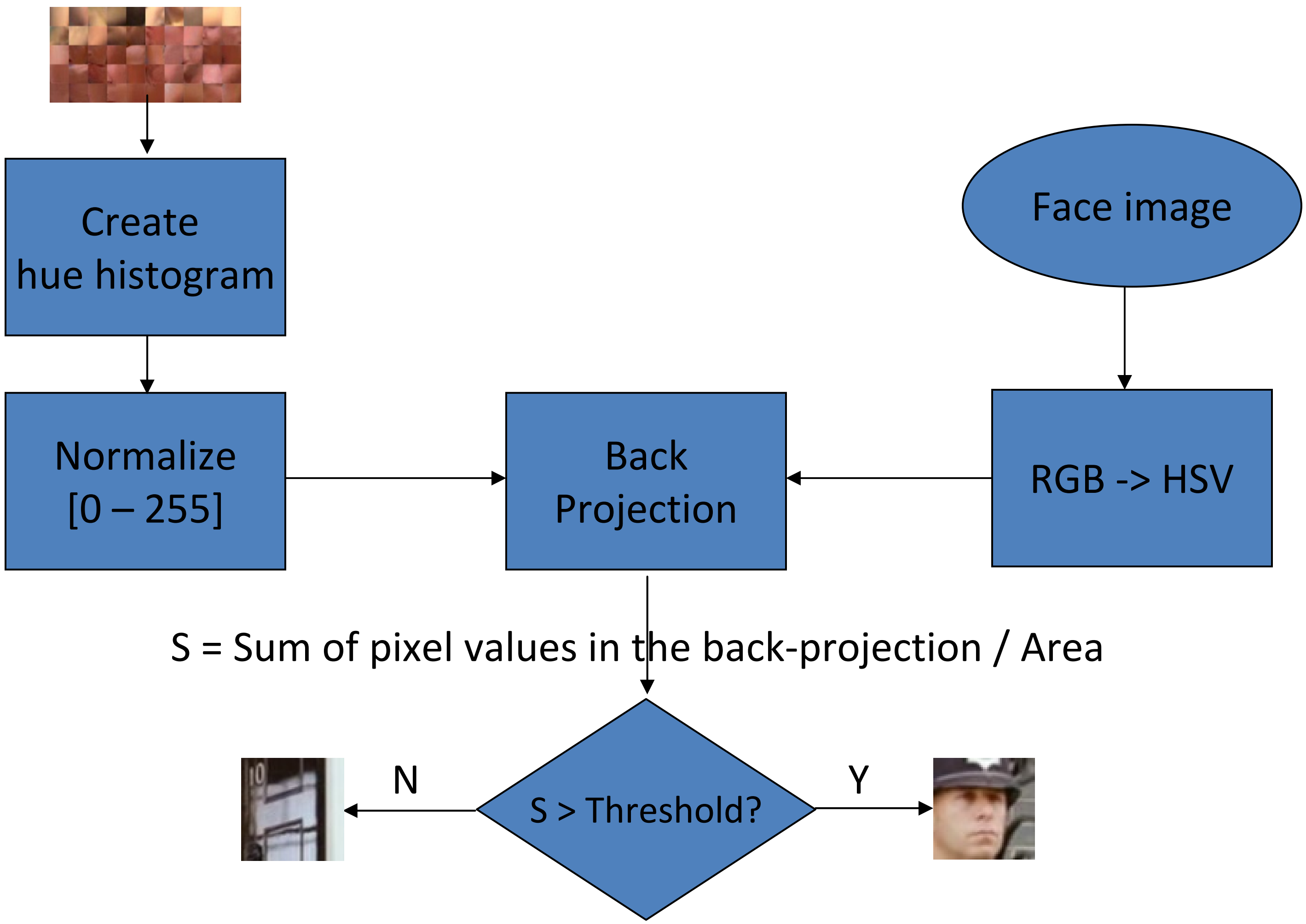


Skin Color Approximation

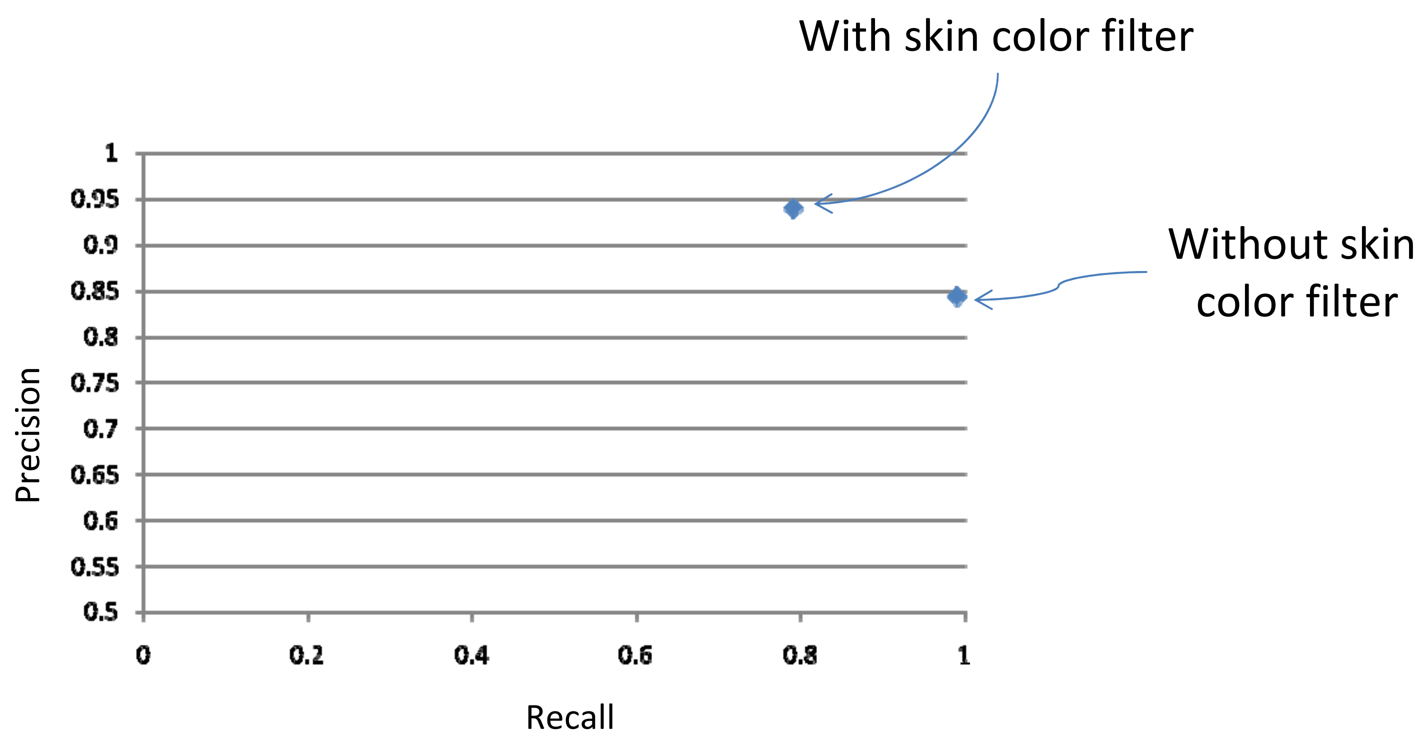
- To filter results of face detector
- Derived from [Bradsky 1998]
- Template Image
  - Patches of faces of different subjects under varying lighting conditions



Skin Color Approximation

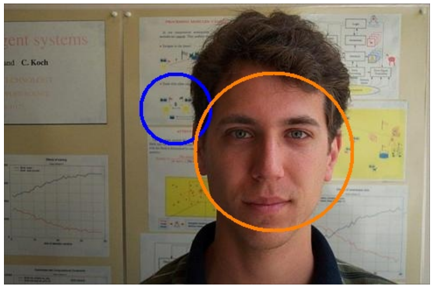


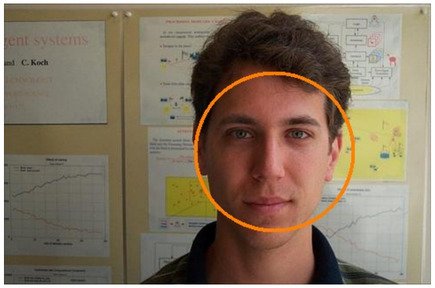
Result

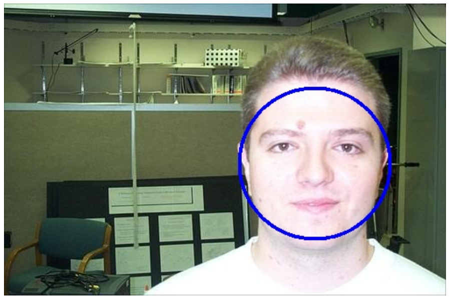


Evaluated on 435 face images in the Caltech 101 dataset

When does it help?



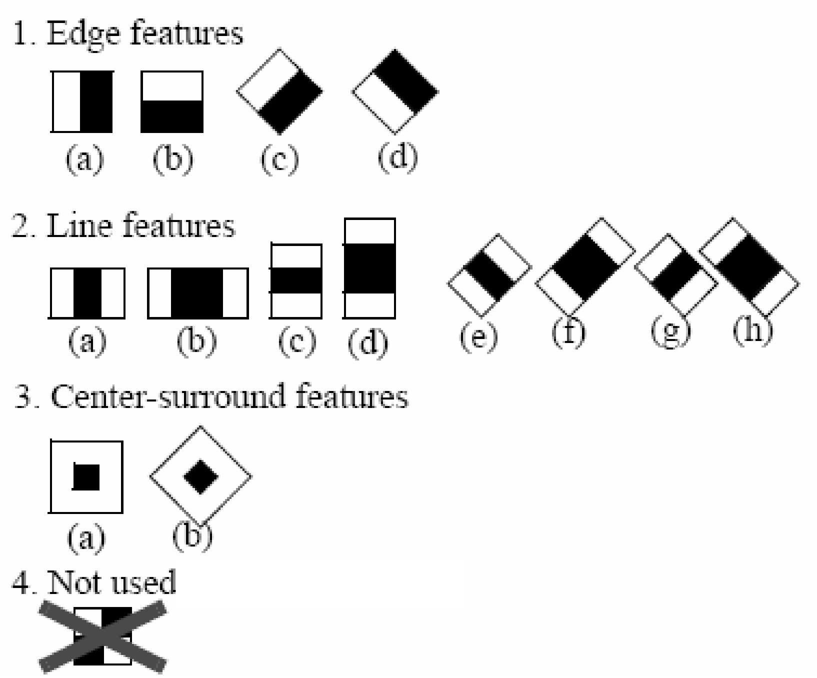




Without skin filter

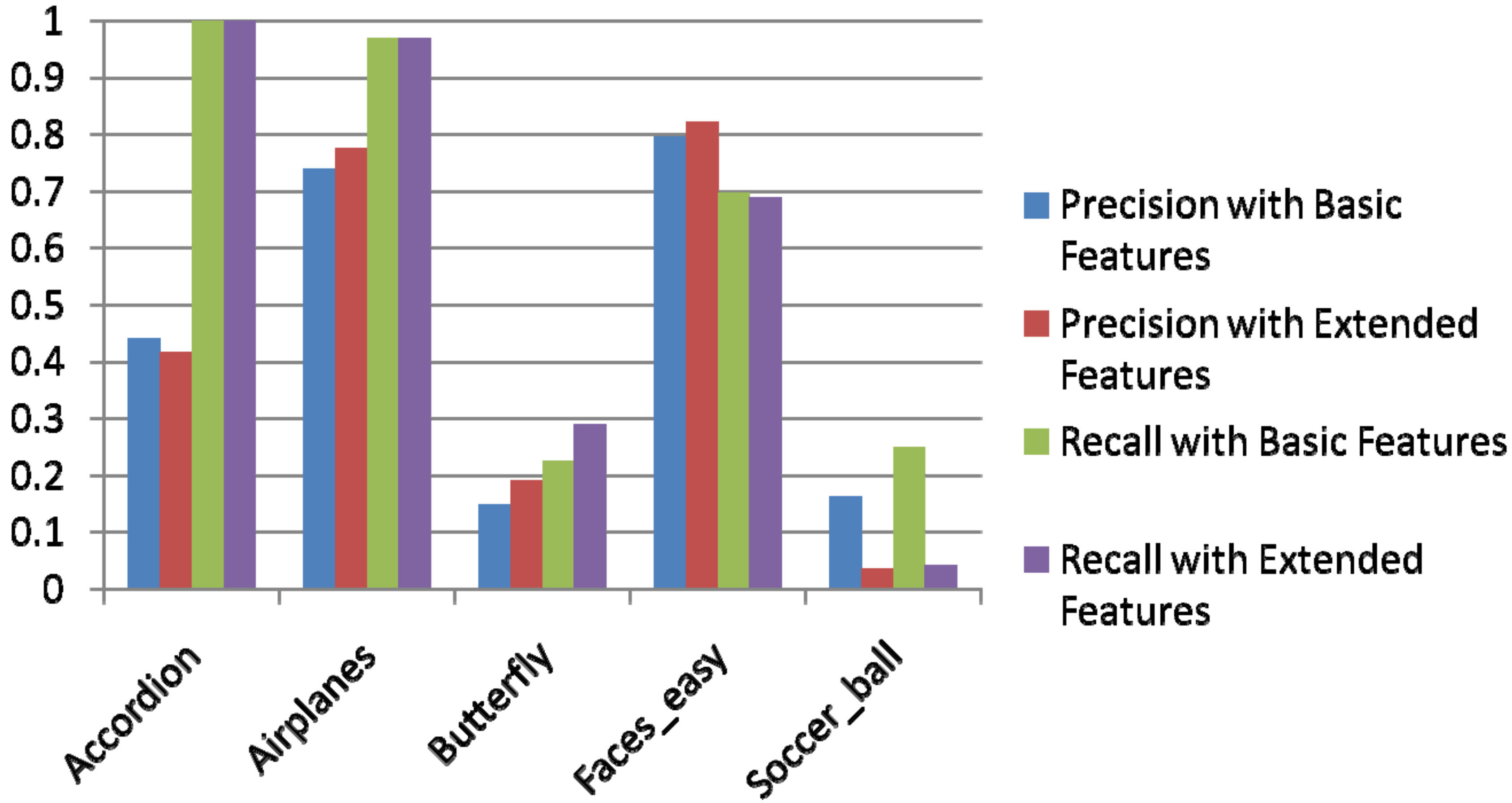
With skin filter

Rotated Features



An Extended Set of Haar-like Features for Rapid Object Detection, Lienhart and Maydt

Results



1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Accordion
Airplanes
Butterfly
Faces_easy
Soccer_ball
Precision with Basic Features
Precision with Extended Features
Recall with Basic Features
Recall with Extended Features

Lessons

1. Viola Jones' technique worked pretty well for faces and some other categories like airplanes and car_sides.
2. Did not work well with many other categories. A large number of false positives.
3. Accuracy depends largely on the amount of variation in training and test images.
4. In some cases, the training algorithm is not able to go below the maximum false alarm rate of a layer, even with a very large number of features.
5. Selected features for the first few stages are more "intuitive" than the later ones.
6. Skin color can be used to increase the precision of face detection at the cost of recall. Dependent on illumination.
7. Using rotated features can increase accuracy but not too much.
8. Training classifiers is slow! Let OpenCV use as much memory as you have.