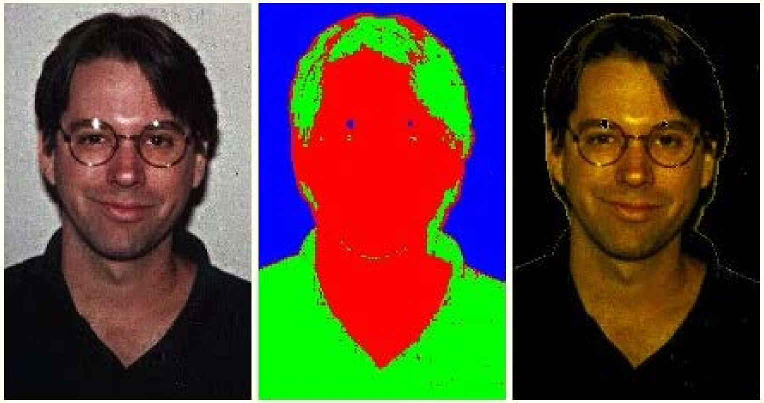
EE 596 Machine Vision HW 4
Assigned: October 12, 2015
Due: October 25, 2015 at 11:59pm

Skin Detection



face image segmented by color skin pixels highlighted

Details: Skin Classification

- r = R/(R+G+B)
- g = G/(R+G+B)
- Do this for both (R,G,B) space and (r,g) space.

- Start with the face training image set. Run the K-means algorithm on the face training set to get K clusters with small K, ie K < 9, represented by average (r,g) [(R,B,G)].

- Represent each cluster by its mean in color space, ie (r_{mean},g_{mean}) [(R_{mean},G_{mean},B_{mean})].

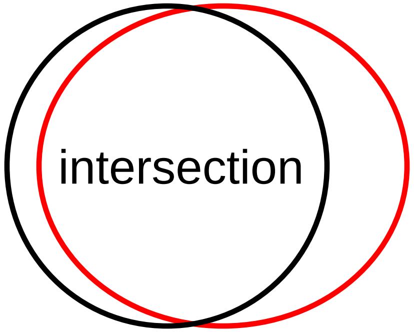
- Use the groundtruth images to assign the true label (skin or not) of each cluster. Majority of pixels wins.

Continued

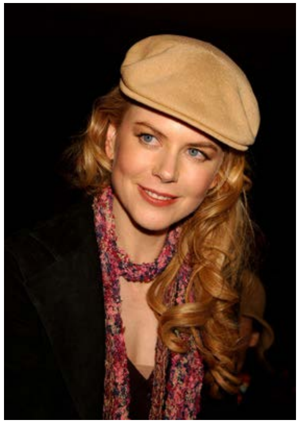
- Train a classifier to learn skin vs. nonskin color in both color spaces. Your training vectors will have centroid plus class for each of the clusters in the training set. Try at least the Naive Bayes and Random Forest classifiers.

- Run your skin finder on images from both the training and testing set, feeding it cluster centroid vectors to be classified.

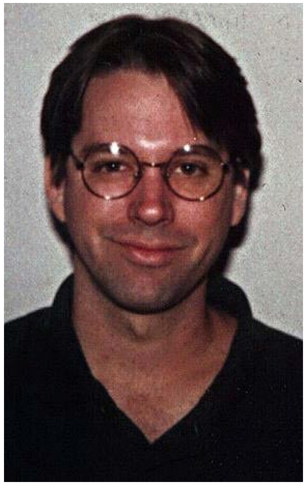
- Report on its performance: Jaccard index plus images. Jaccard index is TP/(TP+FP+FN) or intersection/union.



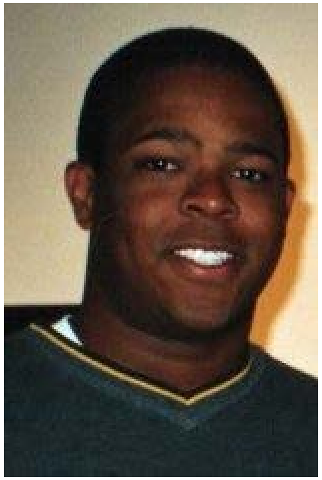
Required Test Images



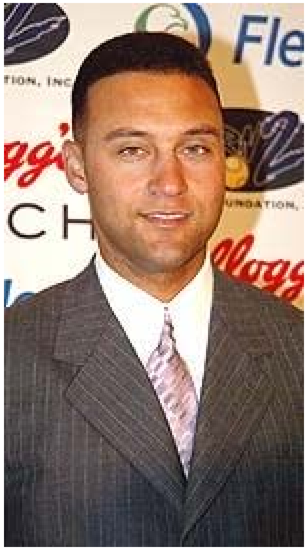
face1.png



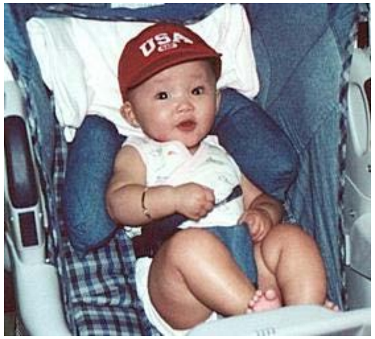
face4.png



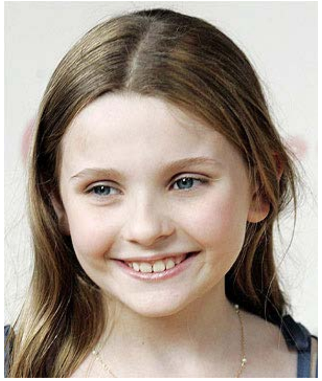
face5.png



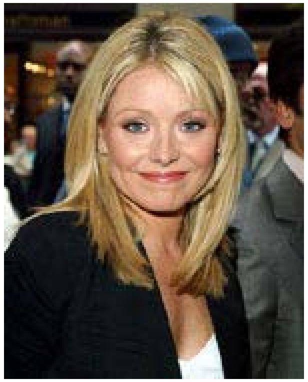
face8.png



face10.png



face23.png



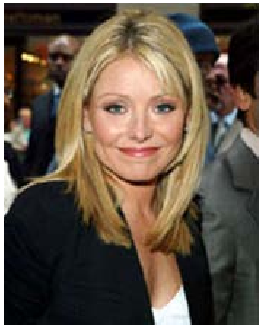
face28.png

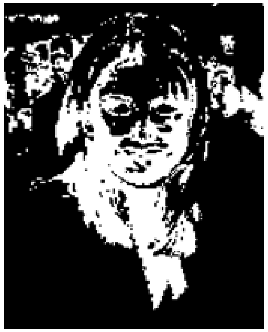
Continued

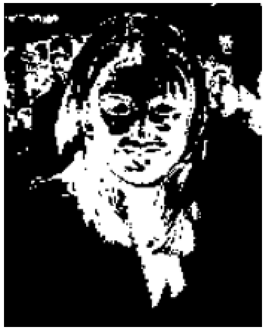
- In the report, show results like those below. See report template.

(1) Name: face28

Original Image NB Skin-Labeled Image (r,g) NB Skin (RGB)







Naïve Bayes Jaccard (r,g) Naïve Bayes Jaccard (RGB)

Random Forest Jaccard (r,g) Random Forest Jaccard (RGB)

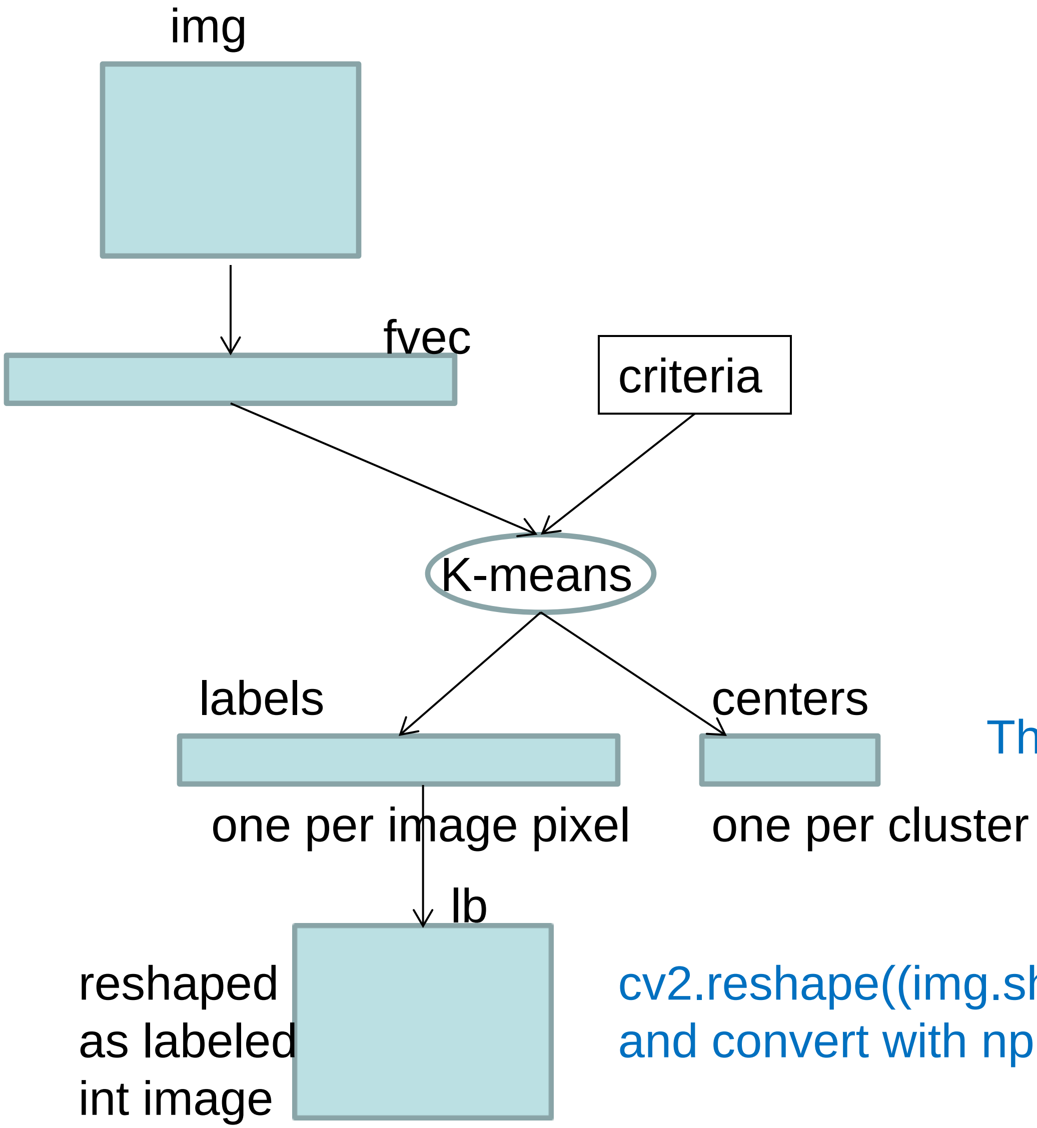
Modules and Data Structures

1. for K-means (for RGB)
   - img = imread(filename) reads the image file
   - ivec = im.reshape((-1, 3)) converts to a vector
   - fvec = np.float32(ivec) converts to float
   - criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, max_iter, epsilon) stops after max_iter iterations or when accuracy is epsilon.

   ```
   ret, labels, centers = cv2.kmeans(fvec, K, criteria,
               attempts, cv2.KMEANS_RANDOM_CENTERS)
   ```

   - attemps: times to try with different random centers

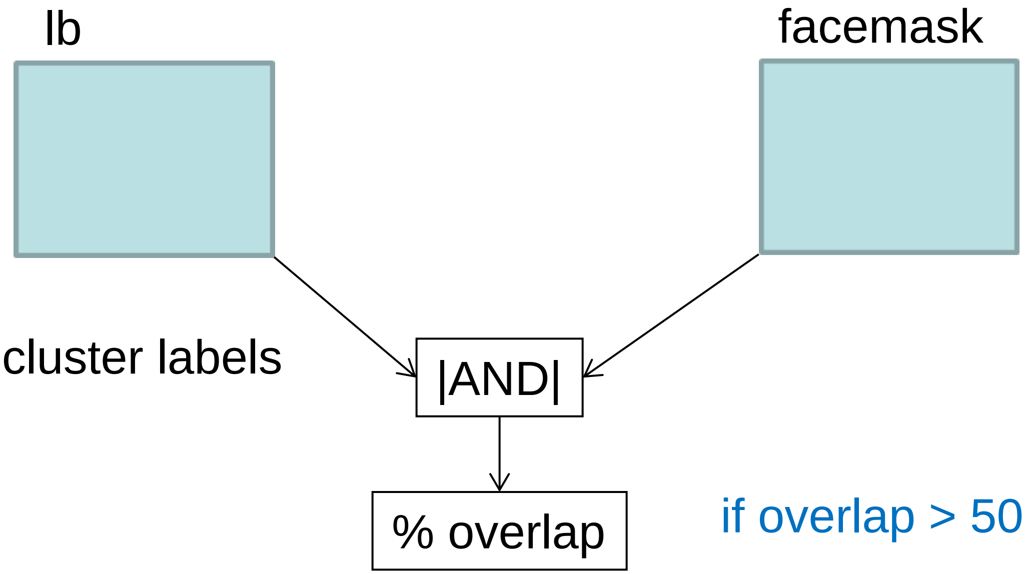
K-means



These are floats.

cv2.reshape((img.shape[0],img.shape[1])
and convert with np.uint8

Setup for Classifying



if overlap > 50%, positive class, else negative

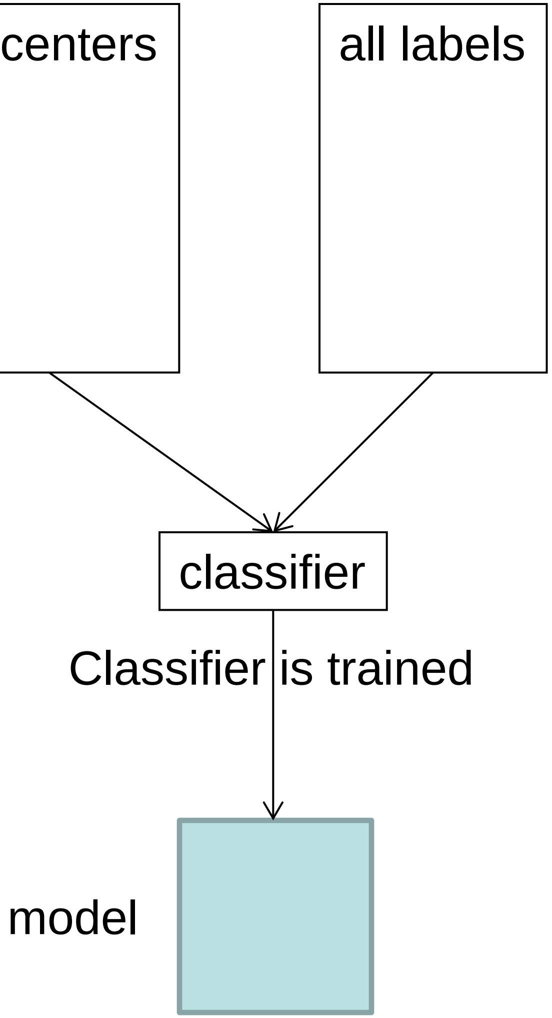
positive centers		
R1	G1	B1
R2	G2	B2

plabels
1
1
1

negative centers		
R1	G1	B1
R2	G2	B2

nlabels
-1
-1
-1

TRAINING



all centers

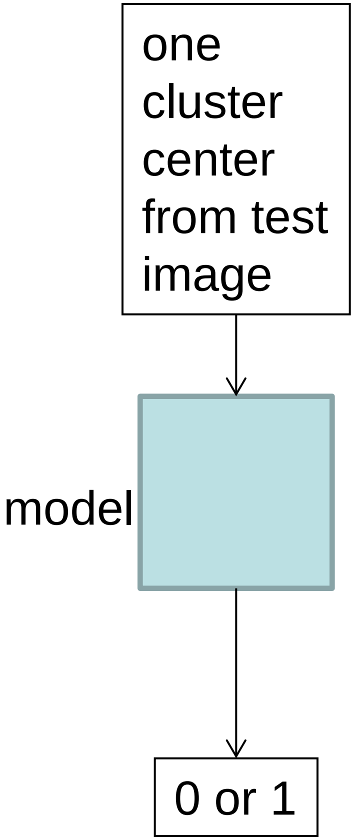
all labels

classifier

Classifier is trained

model

TESTING

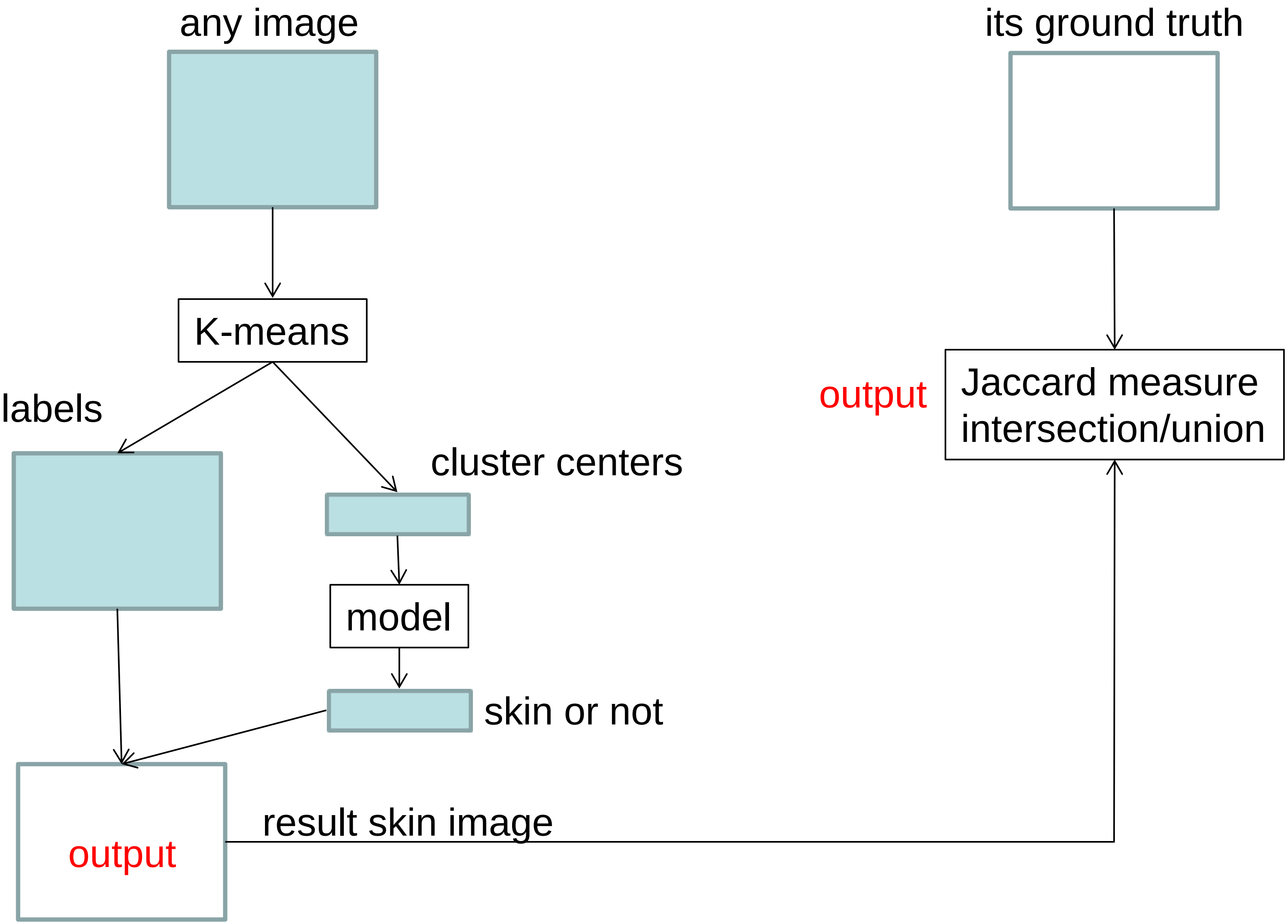


one cluster center from test image

model

0 or 1

Evaluating


any image
its ground truth
K-means
labels
cluster centers
model
skin or not
output
result skin image
output
Jaccard measure
intersection/union

Calling Classifiers

```
model = cv2.NormalBayesClassifier()
or
model = cv2.RTrees()
#
# train the model with the samples and their labels
#
model.train(samples, cv2.CV_ROW_SAMPLE, labels)
#
# use the model to predict the label for one (new) center (float)
#
p = model.predict(centers_test[i])
```