Guess Who App

Description

by Marinos Galiatsatos

Contents

Preface

In this app we want to detect the shirt's and hair' color of a person and also we want to tell if someone wears or not eyeglasses and hat. The person must be at approximately 2 meters from the camera, upright and it is highly recommended a white background behind him, in order to avoid any wrong detected faces or wrong color values. The shirts' colors that we support at the moment are red, green , blue, white and black. For hair we support brown, blonde and black colors. Finally, for eyeglasses we support almost all types of eyeglasses and for hats we use a specific type of hat in order to avoid any wrong detections.

For all of these we have used OpenCV 2.4.9 for Python and the machine learning package for python mlpy. We have also used a HP 1080p webcam. In our setup we had not any fixed lighting conditions but we had a white background behind the person.

Description

In order to accomplish good detections, we set four visible rectangles into the frames of the video. Each of these rectangles represent a Region of Interest (ROI), a specific region that we have set before for shirt, hair, eyeglasses and hat.

Face roi

In order to detect all our ROIs, we want something for reference. So we use the haar cascade face classifier for face detection in order to:

- find if a person exists in the video
- having something for reference to draw our four ROIs

The classifier comes trained with the OpenCV package. We simply draw the face rectangle, which the dimensions are provided from the face classifier.

Shirt roi

We draw a rectangle on a person's shirt and we divide it into six smaller regions with the same size. In this way we want to collect more features for better training. We use RGB and HSV Histograms for feature extraction and we store them in a csv file. Every csv file contains feature vectors from 100 frames of the video. For training, we have used the libsvm algorithm for every color. So for every color we have one libsvm and we have trained them with positive and negative values for

every color. For example, for red color, the red libsvm has been trained with positive values of red features and negative values from the rest of the color. More specifically the csv files with the positive red values contain the red features and also they have an extra column at the end of the feature vectors with 1s. The negative csv files contain the green, blue, white and black features with an extra column at the end of the feature vectors with -1s.

Hair roi

We draw a rectangle above the face rectangle that we use as reference and we divide it into six smaller regions in order to collect more features for better training. We use RGB Histograms for feature extraction and we store them in a csv file. Every csv file contains features from 100 frames of the video. For training we use the libsvm algorithm for every hair' color. As we mentioned in the shirt roi section, we train our svms with positive and negative values from every color. Important note: due to lack of hair samples the algorithms are not trained, so the hair' color detection cannot be done. But, on the other hand, we provide all the tools and scripts for feature extraction and hair libsvms training.

Eyeglass roi

As previously mentioned, we use the face rectangle as reference, in order to draw our eyes rectangle. For feature extraction we use Canny Edges. A Canny Edge can draw the outline of the eyeglasses if the person wears eyeglasses and the outline of the eyes if the person does not wear. We divide the region into 6 smaller region with the same size and we collect the edge values/features from the edge function. Next, we create csv files with the edge values for every one of the 100 frames of the video. For training we use the Perceprton algorithm and we feed them with positive and negative csvs. The positive csvs contains edge features for someone who wears eyeglasses and the negative csvs contains edge features from someone who does not wear eyeglasses.

Hat roi

Using again the face rectangle as reference, we draw a bigger rectangle than the one of hair, above the face rectangle. We divide the rectangle into six smaller rectangles with the same size. For feature extraction we use Canny Edges and FloodFill. We thought that we can extract the edges from the hat and get the values from it, but if someone has thick hair, the classifier may detect that the person wears hat. In order to reduce the wrong detection we use FloodFill. The function floodFill() fills with white color the area that has another color from the background. In our setup the background is white, so a hat that has a different color than the background can be detected. If someone wears a hat, the filled area will be larger than the filled area from just the hair. So we take 1 edge feature and 1 FloodFill feature for every sub hat roi and for 1 frame we have 6+6 features for hat roi. For training we use the Perceptron algorithm and we feed it with positive and negative edge and FloodFill values/features. The positive csvs contains edge and FloodFilled features for someone who wears hat and the negative csvs contains edge and FloodFilled features from someone who does not wear hat. Important note: our training data (hats) were a black and a purple hat. We tried with a white hat but the results were not so good.

Data/Feature Collection

For feature collection we use the *collect_all.py* script, which combines all the rois, that we have mentioned before, and with the suitable functions collects the features and saves them in csv files. If all the rectangles are appeared correctly, the script shows the message 'OK!'. The user has the

ability to start extracting/collecting the features by pressing the 'ESC' key, if the 'OK!' message has been appeared.

Person's Detection

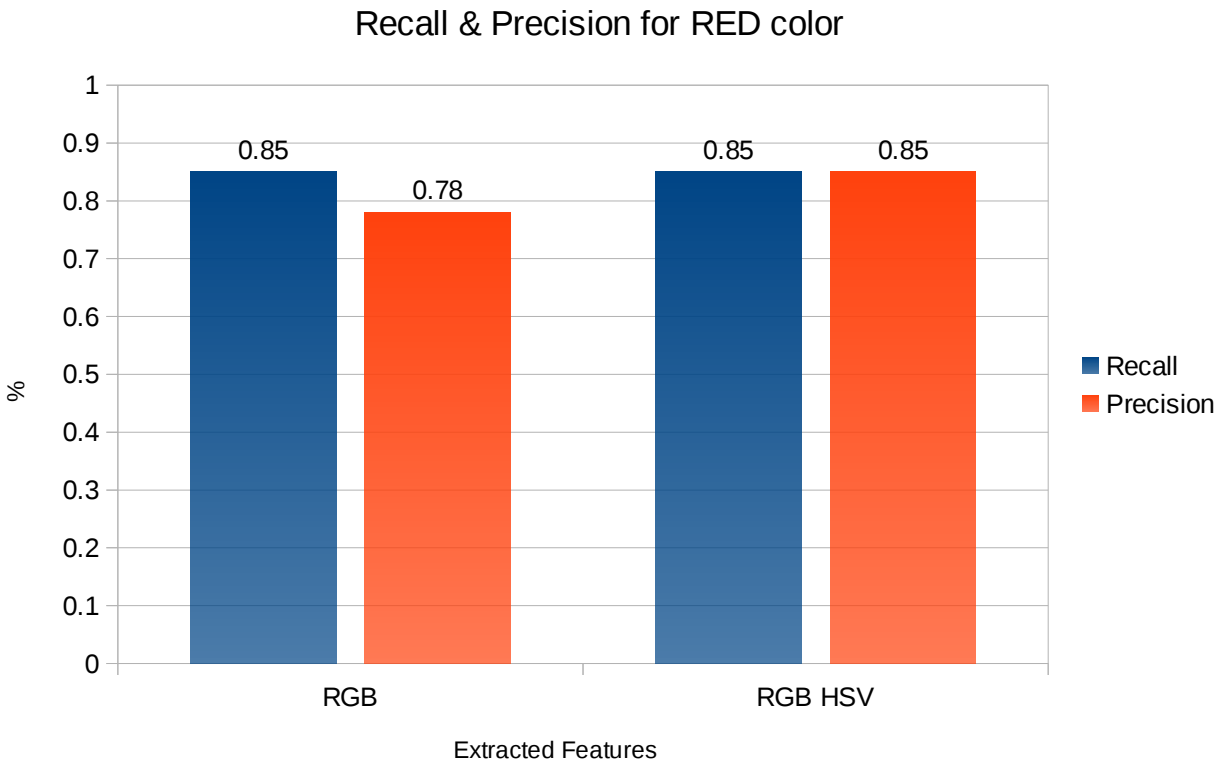
For a person's detection we use the *detection.py* script, which can detect in real time a person. Also, has the ability to detect a person from a prerecorded video. It uses the same way that we mentioned in the Data/Feature Collection section. It loads the trained Libsvms and Perceptrons for detection. For shirt detection, we use a heuristic algorithm in order to decide the shirt's color.

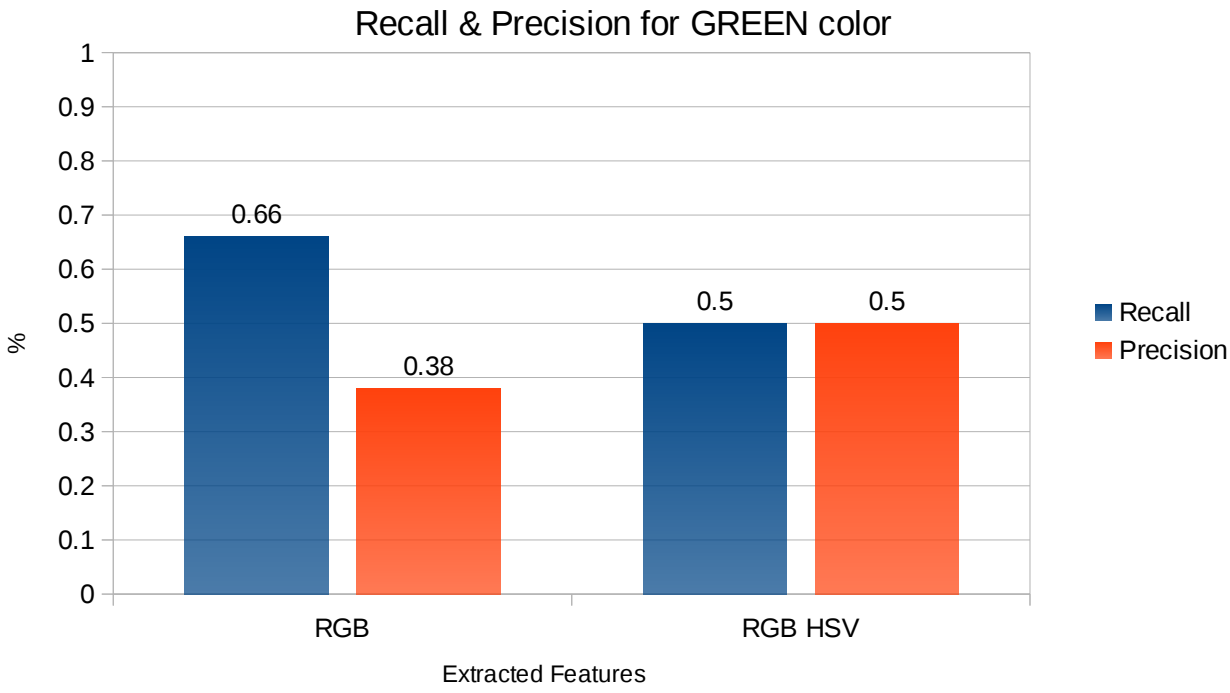
Results

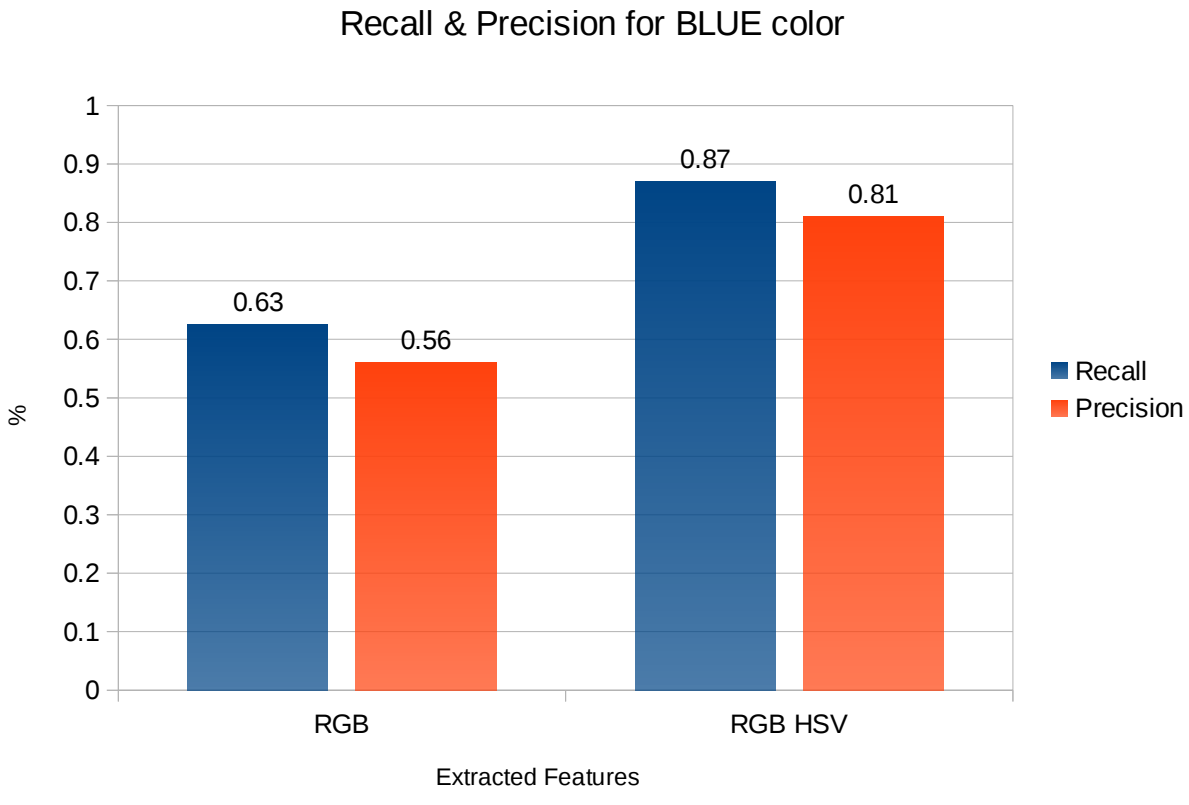
Shirts

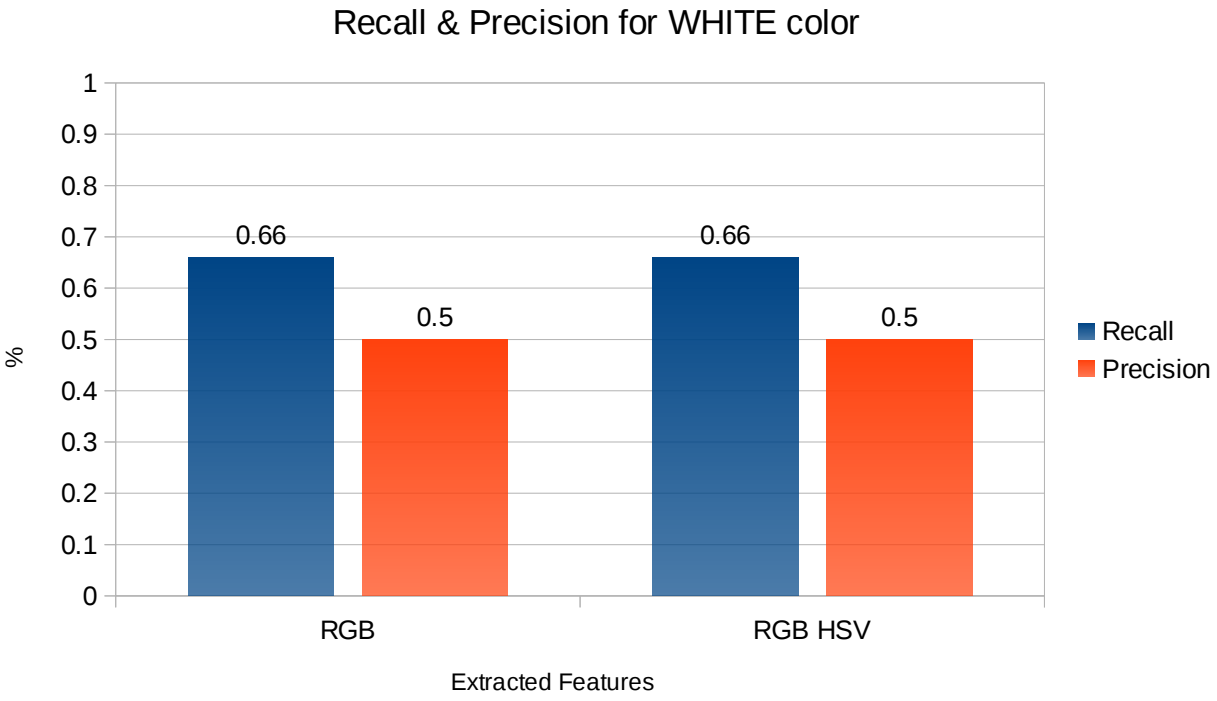
We gave our attention to the detection of shirt's color, due to the fact that the eyeglasses and hat detections can be improved, as described below.

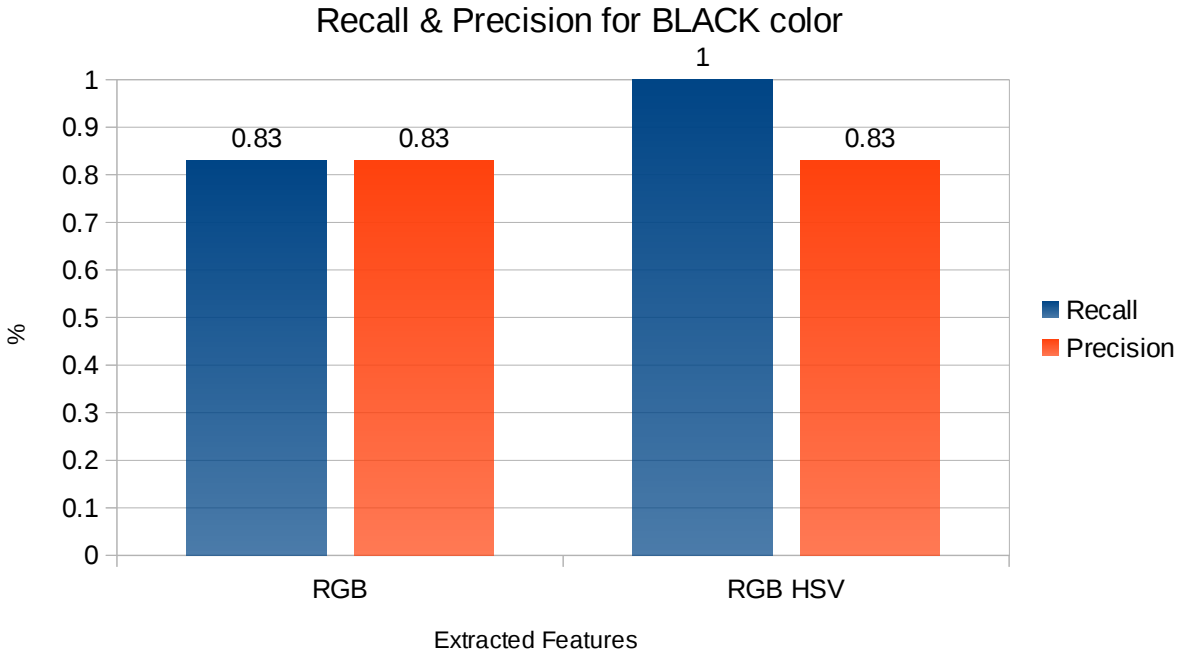
So we calculated the Recall and Precision for every test set for each color (red,green,blue,white,black) and we found the overall Recall and Precision. The datagrams below show the results:




Recall & Precision for GREEN color
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
%
0.66
0.38
0.5
0.5
RGB
RGB HSV
Extracted Features
Recall
Precision


Recall & Precision for BLUE color
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
%
0.63
0.56
0.87
0.81
RGB
RGB HSV
Extracted Features
Recall
Precision


Recall & Precision for WHITE color
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
%
0.66
0.5
0.66
0.5
RGB
RGB HSV
Extracted Features
Recall
Precision


Recall & Precision for BLACK color
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
%
0.83
0.83
1
0.83
RGB
RGB HSV
Extracted Features
Recall
Precision

Thoughts about Shirt's recall and precision

As we can see, the RGB and the RGB HSV recall and precision have similar values. In some colors the one way is better than the other, but we do not have a clear winner. If we had more test sets, maybe we could make a safe decision. In our app we use both ways for shirt's color decision and if they have different results, we display both.

Future Work

For future work, we could do some things to improve the App:

1. First, we could have a fixed setup, with standard lighting and white background for collection and detection. Then we can re-train our svms with new data, in order to avoid wrong detections.

2. Second, we could train a Neural Network for eyeglasses detection. This is the most safe way for eyeglasses detection and then we will know for sure if a person wears eyeglasses or not.

3. Third, we could train a Neural Network for hat detection. With this training we could train and other types of hats, so our App will have the ability to recognize all types of hats.

4. Fourth, with this App we can easily create the “Guess Who” game by adding questions and voice to our App.

Home

End of Description