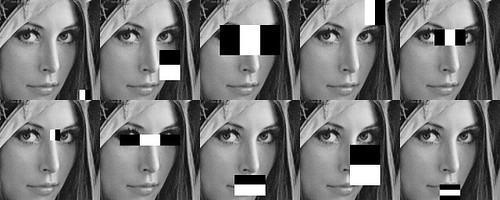
**Overview**

The Purpose of this project is to build a web based, mobile responsive camera usinf AI and machine learning to identify skin tones, undertones and skin textures.

**Harr Cascade**

Haar-like features are digital image features used in object recognition. They owe their name to their intuitive similarity with Haar wavelets and were used in the first real-time face detector. Historically, working with only image intensities made the task of feature calculation computationally expensive

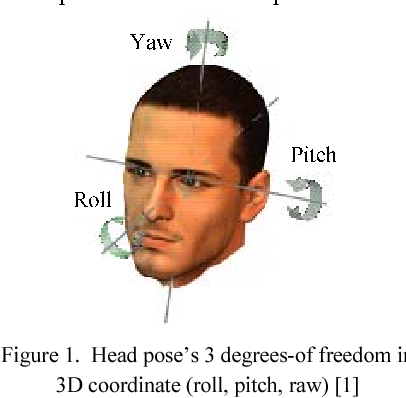


**Face Angle- Head pose estimation**

There are three major steps:

1. Face detection. A face detector is adopted to provide a face box containing a human face. Then the face box is expanded and transformed to a square to suit the needs of later steps.
2. Facial landmark detection. A custom trained facial landmark detector based on TensorFlow is responsible for output 68 facial landmarks.
3. Pose estimation. Once we got the 68 facial landmarks, a mutual PnP algorithms is adopted to calculate the pose.

The marks is detected frame by frame, which result in small variance between adjacent frames. This makes the pose unstable. A Kalman filter is used to solve this problem, you can draw the original pose to observe the difference.



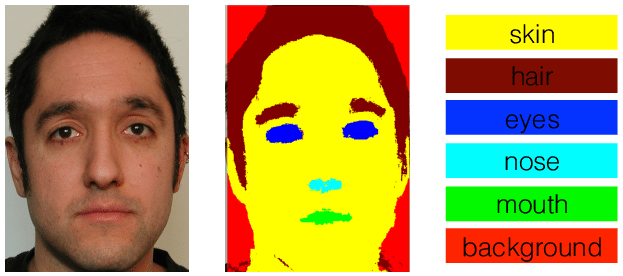
**OpenCV:**

**OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source BSD license.**  Find different properties of images like brightness, contrast, color distribution etc which will be helpful to make color and tone detection better. We will be using Opencv contrast operation to contrast the images for better visualization.

## **Region-based Segmentation**

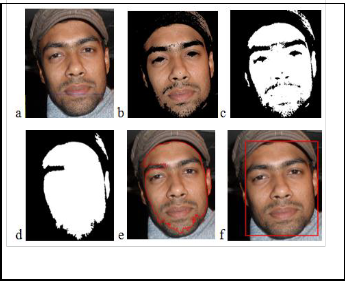
One simple way to segment different objects could be to use their pixel values. An important point to note – the pixel values will be different for the objects and the image’s background if there’s a sharp contrast between them.

In this case, we can set a threshold value. The pixel values falling below or above that threshold can be classified accordingly (as an object or the background). This technique is known as **Threshold Segmentation**. An image will have other regions beside the face . Face segmentation is required to automatically crop out face from the image and detection can be carried out only in that part. For this we will be using ROI pixel segmentation technique of Opencv to get the desired result.



## **Color Detection**

For detecting faces in color images using skin color model algorithm combined with skin likely-hood, skin Segmentation, Morphological operation and Template matching. Color images with skin color in the chromatic and pure color space YCrCb, which separates luminance and chrominance components. A Gaussian probability density is estimated from skin samples, collected from different ethnic groups, via the maximum-likelihood criterion. Adaptive thresholding for segmentation to localize the faces within the detected skin regions. Then, mathematical morphological operators are used to remove noisy regions and fill holes in the skin-color region, so we can extract candidate human face regions



**Camera Server Interface and Web Architecture**

For one client all cameras will be connected with a system which will be further connected with the cloud using SSH.

* Picture will be either taken from mobile camera,
* Comparing the pixes values with grey card inorder to white balancing
* Upload the picture
* Picture will be sent to the model
* Generated result will sent to webURL using Flask as an API
* Compare the result with database provided by client
* Recommend 2 other choices closest to detected color
* Store the output
* SHow the output

**Milestones**

(i) face Detection (5-7 hrs):

* Configuring software binaries:
  + Anaconda environment
  + Numpy
  + OpenCV
  + Python Jupyter Server Setup
  + MySQL or any other database.
  + Visualization and misc. libraries: Matplotlib, Seaborn and others.
* Building haarcascade model,(publically avaialable but we have to build in a way that it should run on web)
* Train it for faces

(ii) Segmentation and color inferencing and Data storage(10-12 hrs):

* Segment the face areas.
* Finding the correct angle.
* Skin tone detection.
* Getting YCB, Hex, RGB value
* Store it to database

(iii) UI design - Verification and development (10-15 hrs):

Web base GUi, deploying everything on web portal

(iv) Reporting System (1 hr):

* Show the output
* Compare with 110 Pantone Scale
* Final result
* Database storage