Gaze-tracking and moving the cursor with gaze direction

- **Chethan Hebbar**

Face-Detection(First step):

The method:

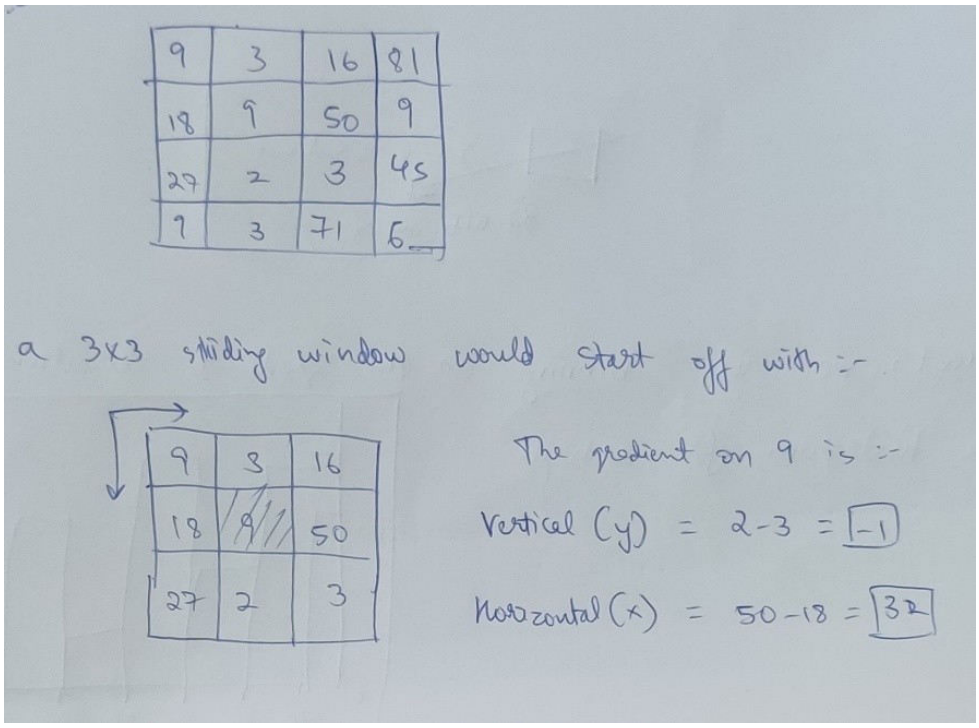
1) We read the image/frame and convert it to grayscale.

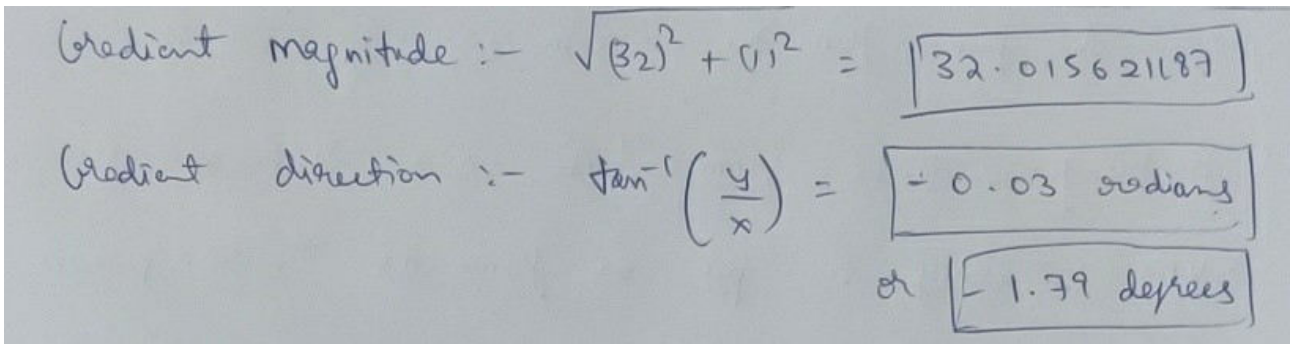
2) “get_frontal_face_detector” from dlib library uses HOG and then a linear SVM to classify the images as “with face” or “without face”

3) Feeding the grayscale image to this detector, let us look at it’s working briefly

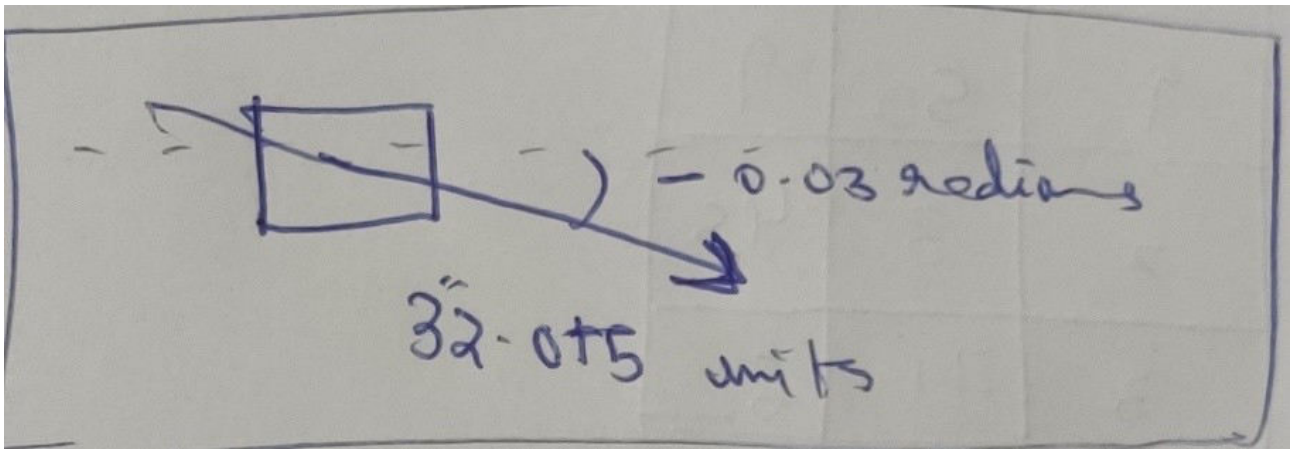
4) Histogram Oriented Gradients(HOG):

* HOG uses a sliding-window approach on the grayscale image and calculates the gradient values for the pixels. Consider a 4x4 image.

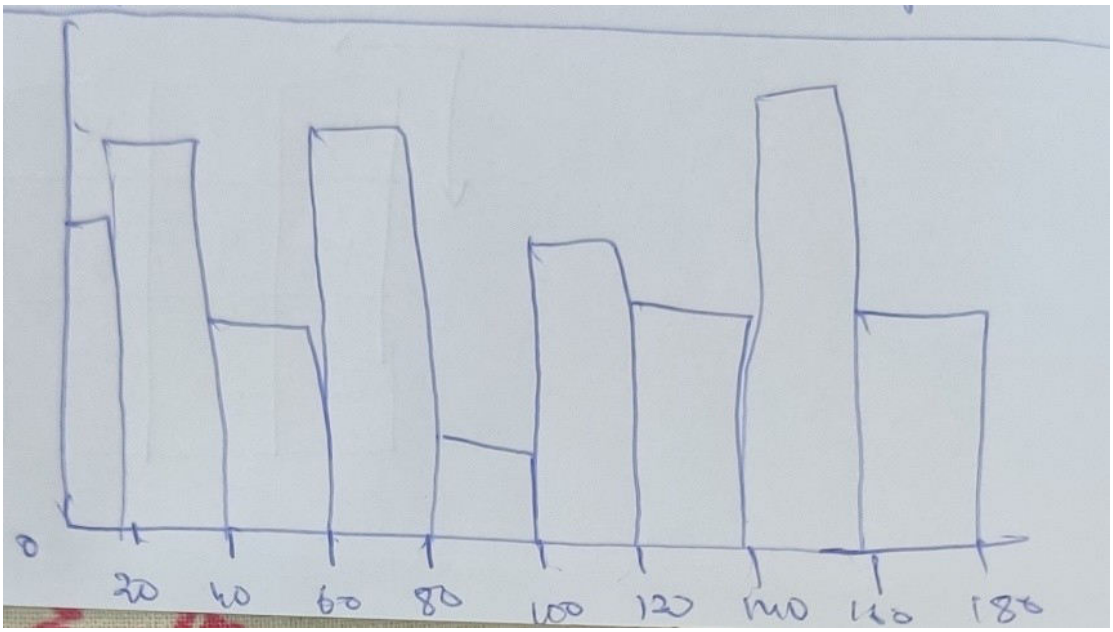




* Now, after calculating the gradient direction for all the pixels using the sliding window, we have a set of gradient vectors at the pixels, the pixel looks something like this:



* Next, the model performs binning on the gradient directions and we get a histogram, for example:

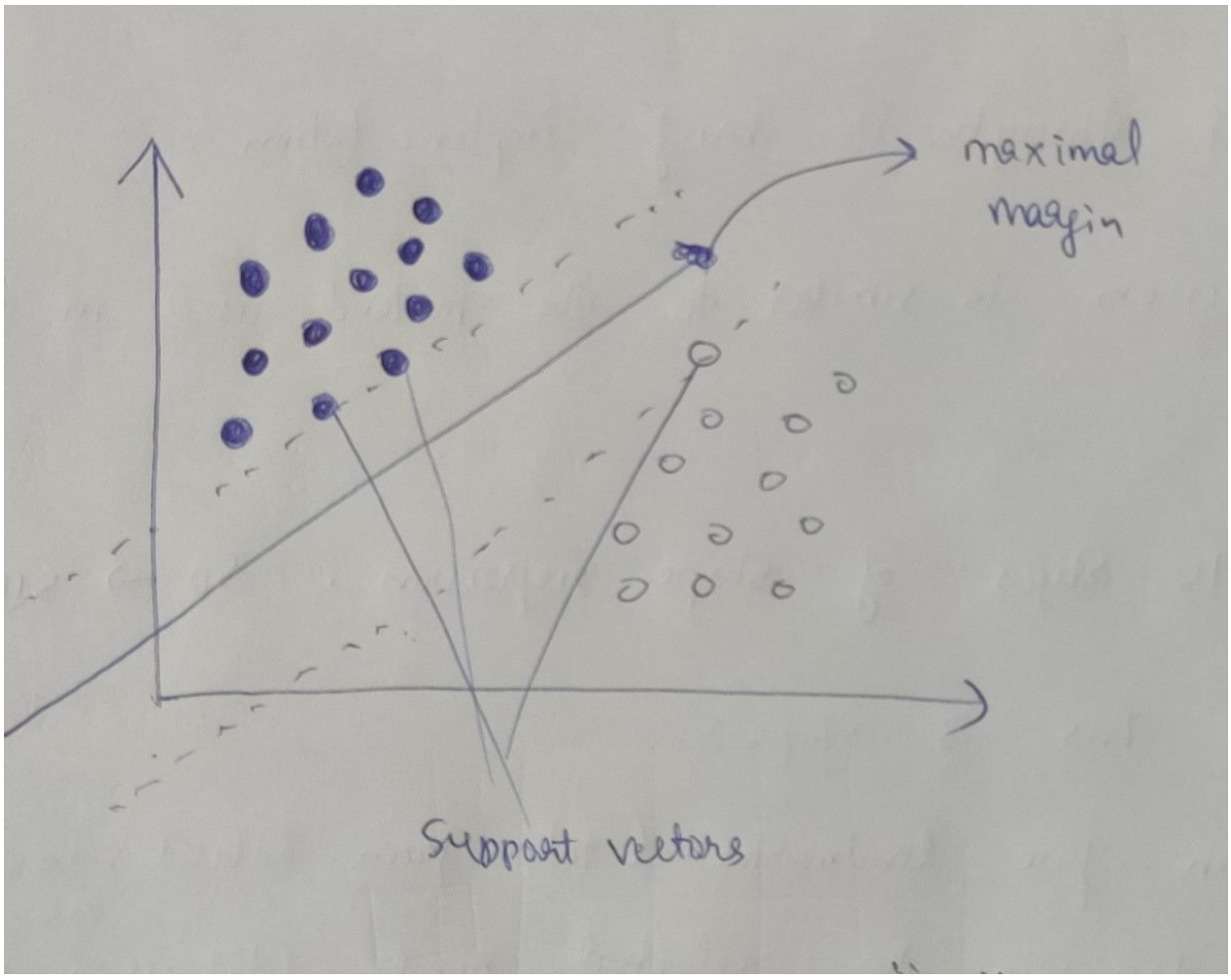


5) After normalisation, the datapoints are sent to the Linear SVM to be classified:

* Given n-datapoints and we have 2 classes, i.e “with face”(1) and “without face”(0).

* The SVM performs “dimension-expansion” => n + 1 dimensions using “kernel functions” [f(xi) where xi is a n-dimensional vector, aka data point]

* A hyperplane is the used to best-fit the points, a hyperplane has the equation [w^T * xi – B = 0] where w is the normal vector to the hyperplane.



* A datapoint on which prediction is to be made, is passed through the kernel function and then classified based on which side of the hyperplane it lies on.

6) Next the model applies a bounding box on the face and returns a tuple of the diagnol of the box, i.e (p1, p2) where p1 p2 are ends of the diagnol.

Landmark Detection(shape prediction):

1) We pass the rectangle coordinates and the gray frame/image to the shape-predictor-68 pretrained model. It has already been trained to produce state-of-the-art results.

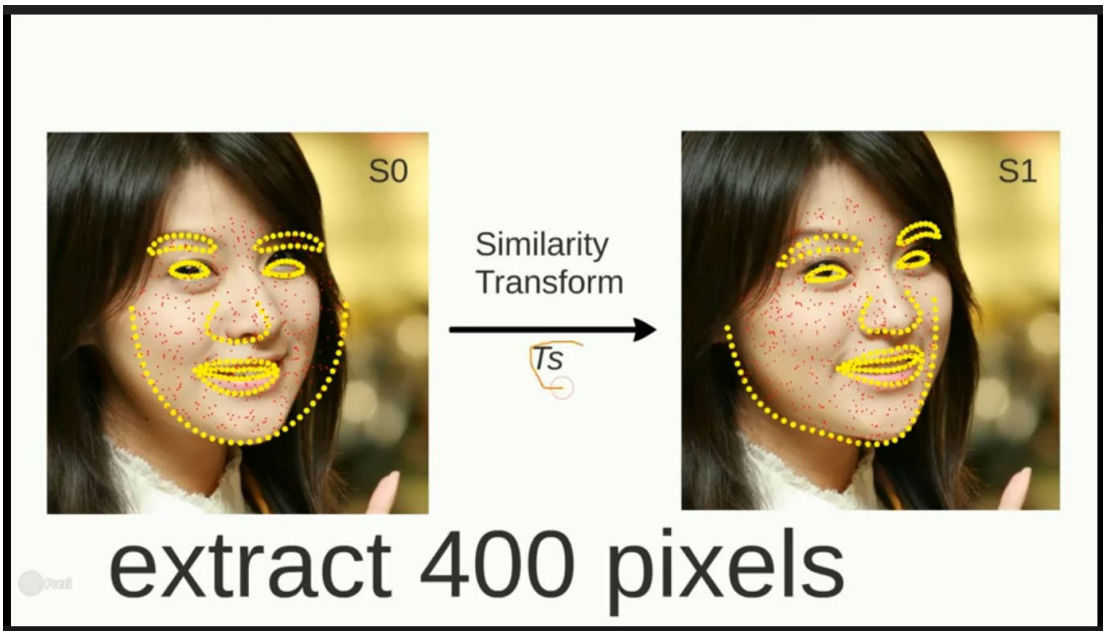
2) Let us go through the implementation briefly.

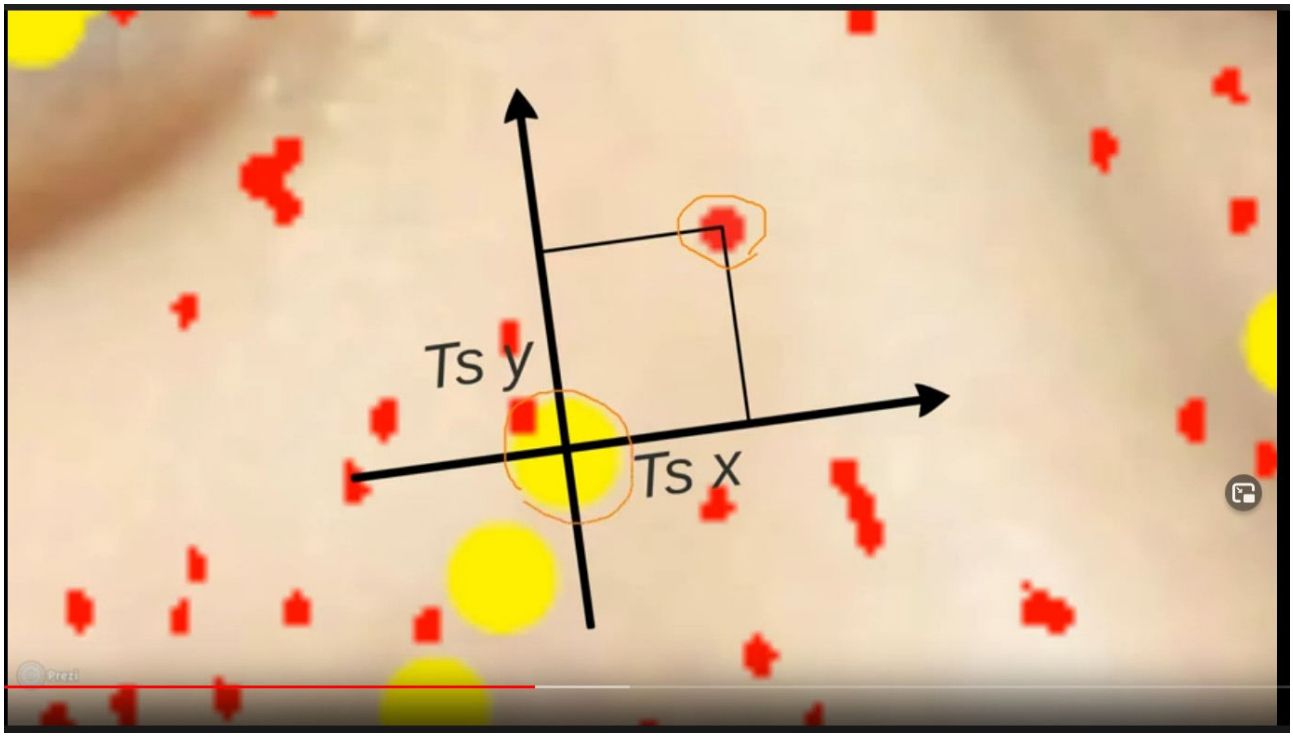
* Shape regression is similar to the method used in the pre-trained model.

* Let us say that the model uses 10 stages of shape-regression, going from S(0) to S(10).

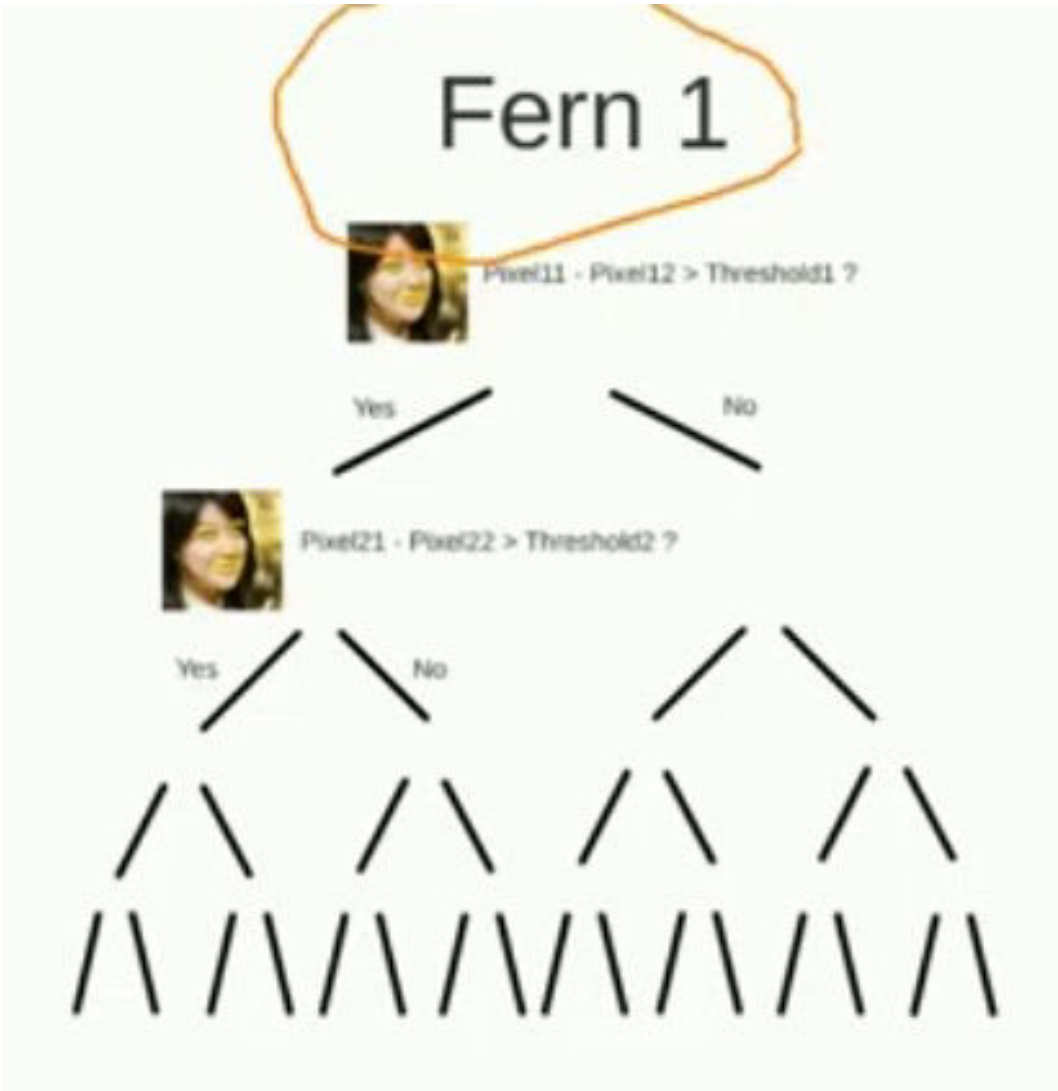
* Every stage uses basically the same startegy:
> S(0) uses a mean-landmark from the previous datasets while others use the previous landmarks, i.e S(I) uses S(I-1)'s landmarks

> The landmarks are then 'transformed' along their axes





> Ferns: a fern can be thought of as a constrained binary tree where the same conditional check(done in decision trees) is done. It helps to split data into groups.



> The DELTA(landmarks) is the loss on which the model is trained, landmarks after applying the transform are slightly changed.

> For this case let us take an example of 400 pixels on which transofrmation is applied and 500 ferns per stage.

* Testing algorithm:

> Map the mean facial landmarks to the rectangle sent(i.e the face)

> for stages 1 through 10:

> Retrieve and map the locations of the 400 sample pixels with respect to image using the similarity transform S(mean) to Sc.

> for j from 1 to 500:

> Traverse the jth fern by comparing the pixel difference feature with the threshold at each level. Reach one of the

16 leaves and obtain the DELTA at that leaf and set Sc = Sc + DELTA.

Controlling the mouse using HOUG circle detection:

1) Now that we have our landmarks(68 coordinates), we first perform a little preprocessing.

2) Using a mask and joining the landmarks of the eye as a polygon, we get the eye image.

3) Reducing noise in the eye image by using motion blur or gaussian blur.

4) Next we pass the image to HOUGE gradient based algorithm to detect circles in the image.

6) Ok, so basically after detecting the circle(majorly the iris of the eye), we use the distance-vector between the center of the detected circle and the center of the landmarks(resting position of the pupil) to control our mouse.

7) The results is not great, the cursor is not stable.

Next steps and another good repo used:

1) Want to use CNNs to classify the eye-image as looking center, left, right, top or bottom.

2) One challenge I had with this is creating the dataset(I could not find dataset online, although I did’nt look very closely)

3) Training the CNN can be done given time.

4) I used a well documented repo to get better results, the repo is linked in the README on this repository.

5) It turned out that the HOUGH-Circles method gave better results, but have not implemented CNN.

6) The tool is quite slow on my machine, running it remotely or on a GPU might increase performance significantly.

References:

Paper on HOG ::http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

Paper on Ensemble of regression trees:
https://www.csc.kth.se/~vahidk/papers/KazemiCVPR14.pdf