Handwriting recognition

**Handwriting recognition** (HWR) is the ability of a computer to receive and interpret intelligible [handwritten](https://en.wikipedia.org/wiki/Handwriting) input from sources such as [paper](https://en.wikipedia.org/wiki/Paper" \o "Paper)documents, [photographs](https://en.wikipedia.org/wiki/Photograph), [touch-screens](https://en.wikipedia.org/wiki/Touch-screen) and other devices. The image of the written text may be sensed "off line" from a piece of paper by optical scanning ([optical character recognition](https://en.wikipedia.org/wiki/Optical_character_recognition)) or [intelligent word recognition](https://en.wikipedia.org/wiki/Intelligent_word_recognition). Alternatively, the movements of the pen tip may be sensed "on line", for example by a pen-based computer screen surface, a generally easier task as there are more clues available.

Object tracking (process involved)

1. Taking an initial set of object detections (such as an input set of bounding box coordinates)
2. Creating a unique ID for each of the initial detections
3. And then tracking each of the objects as they move around frames in a video, maintaining the assignment of unique IDs.

Furthermore, object tracking allows us to apply a unique ID to each tracked object, making it possible for us to count unique objects in a video. Object tracking is paramount to building a person counter (which we’ll do later in this series).

Steps in detecting the object are:

Step #1: Detect the presence of a specific colored object(ball) using computer vision techniques.

Step #2: Track the ball as it moves around in the video frames, drawing its previous positions as it moves.

Detailed procedure :

🡪We start object tracking by Capture video frames from webcam

🡪Define the lower and upper bandwidth of color you want to track

🡪Create an empty queue

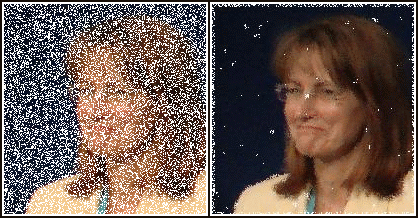
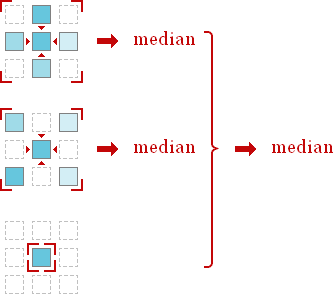
🡪 Flip the image that we are capturing

🡪 Then convert the image from BGR to HSV format

* *We convert so to separate image luminance from color information .HGV is used in situations where colors description plays an integral role*
* **

🡪 Threshold the HSV image for a range of chosen color by masking within that range of bandwidth

🡪 Smoothen the image to reduce the noise in the frame (in this project we median\_blur and GaussianBlur)

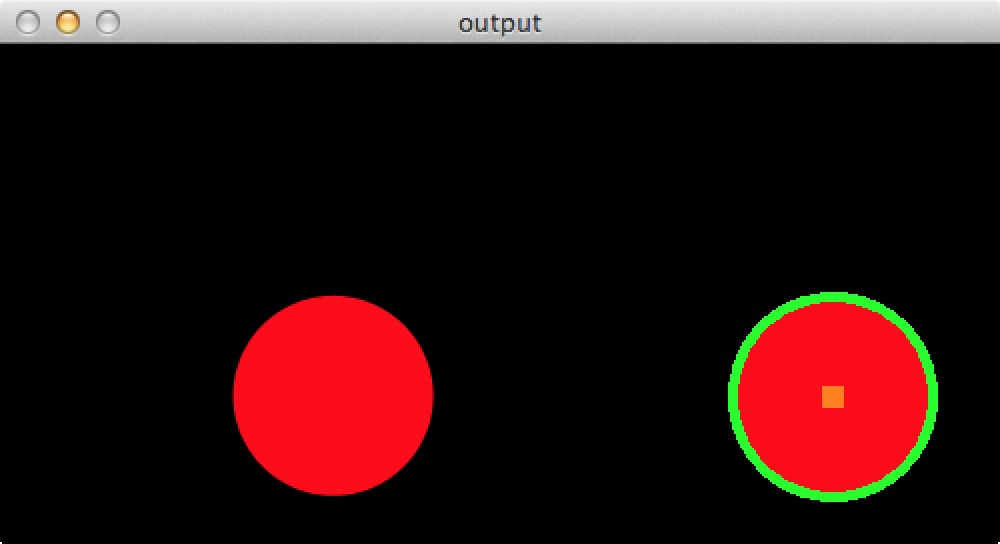
 GaussianBlur

🡪 Define an threshold to set the range of operation

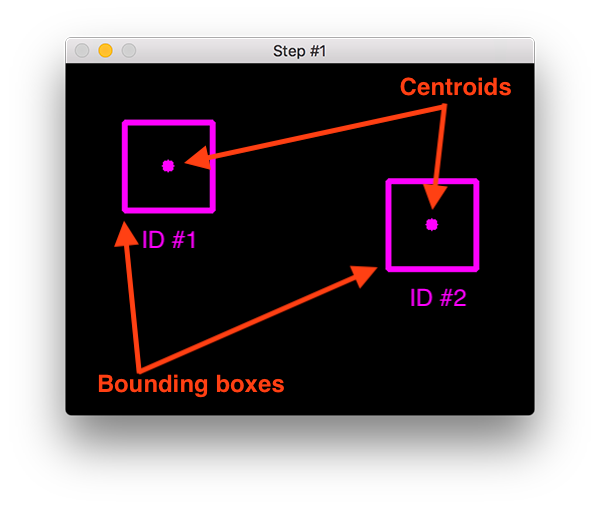
🡪 Then find contour (edges ) in frame to separate object



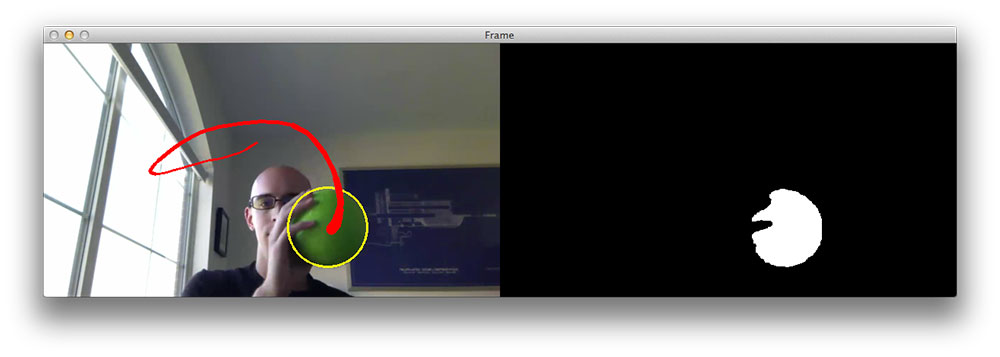
🡪 Set an limit for object size if the object size is greater than prescribed value then start searching for the object with descired color In the frame

* *Then encircle the object with minimum dia circle that fits it*
* **

🡪Then check for the change in the moment of circle from the initial position

🡪

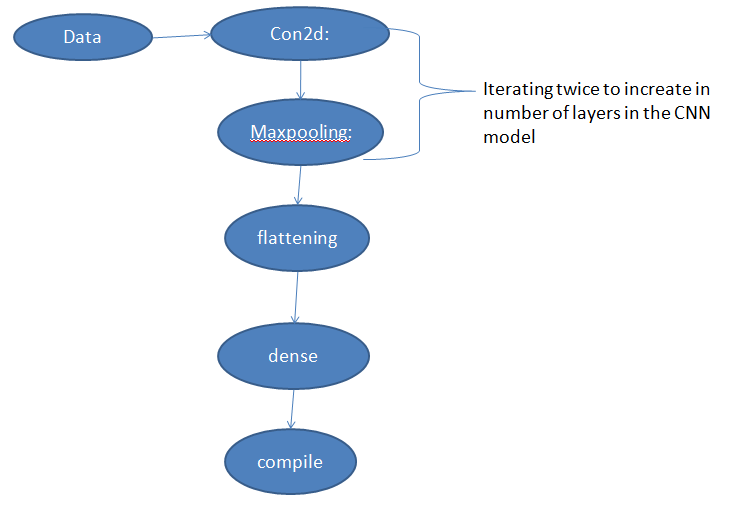
🡪If there is any change then we append the new center to the empty queue we created along with initial center

* *This queue is used to track the object ,we then draw a line along the path it traces*
* *This line is then displayed on the screen*
* **

🡪Based on the path the object traces *MODEL* predicts which character resembles the path it’s tracing

MODEL BUILDING:

* We start our model by importing the required modules we need
* We use keras to build an CNN network for the model that uses those lines object was tracing to predict its output
* Data for the model is the Hindi character data set, which we can download
* ***Architeture used in the model***



* We save the model with .h5 format
* This .h5 file is used in the application part to predict the model