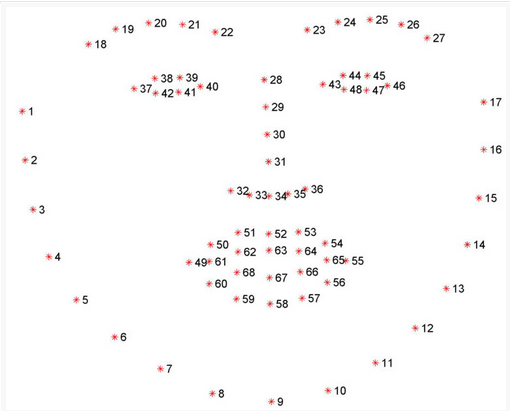
**The drowsiness detector algorithm**

* First, we’ll setup a camera that monitors a stream for faces
* If a face is found, we apply facial landmark detection and extract the eye regions
* We will be using “**haarcascade\_frontalface\_default**” classifier and “**dlib’s shape\_predictor\_68\_face\_landmarks**”.

### Understanding dlib’s facial landmark detector

* The pre-trained facial landmark detector inside the dlib library is used to estimate the location of ***68 (x, y)-coordinates*** that map to facial structures on the face.

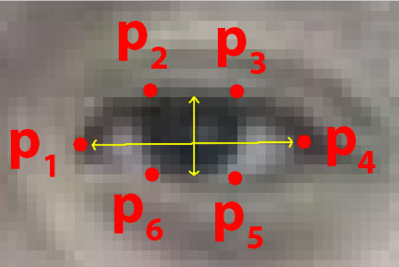


### Detecting facial landmarks with dlib, OpenCV, and Python

* Initializes dlib’s pre-trained face detector based on a modification to the standard **[Histogram of Oriented Gradients + Linear SVM method](https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/" \t "_blank)** for object detection.

**In terms of blink detection, we are only interested in two sets of facial structures — the eyes.**

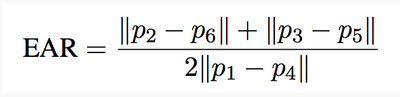
Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:



Based on this image, we should take away on key point:

**There is a relation between the width and the height of these coordinates.**

We can then derive an equation that reflects this relation called the eye aspect ratio (EAR) (*for drowsiness detection)*



To improve our blink detector:

* Computing the eye aspect ratio for the N-th frame, along with the eye aspect ratios for N – 6 and N + 6 frames, then concatenating these eye aspect ratios to form a 13 dimensional feature vector.
* Training a HaarCaascade on these feature vectors.