HOG “histogram of gradients”

It generalizes the object representation in a way that the output representation for the same object is wouldn’t undergo much change under different conditions (like light or other noise), this is called feature descriptor.

It takes input image of object (face in our case) and produces Output vector representing histograms of gradients

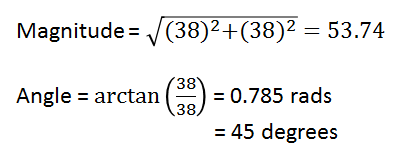
it calculates gradient vector for each pixel by measuring change in pixel value along x-direction and y-direction.

For example:

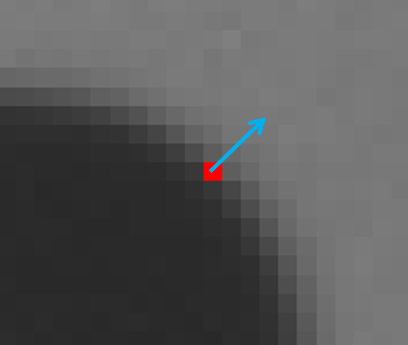
This is a grayscale image, so the pixel values just range from 0(black) – 255(white), we comoute the change in x-direction as the difference in color values between pixels to left and right of current pixel, so in this case it would be 38 (94 – 56), note that the sbtraction can be from left to right or from right to left but which ever order we take we must use it for the entire image, similarly the change along y-direction is computed as 38(93 – 55), and Now we have our gradient vector:

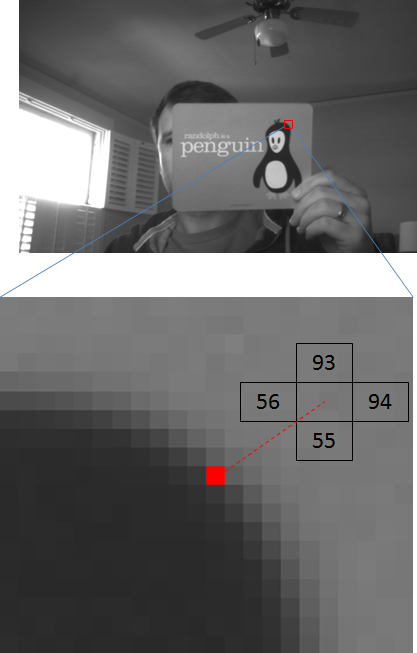


the gradient vector for current pixel can be represented with the given magnitude and direction

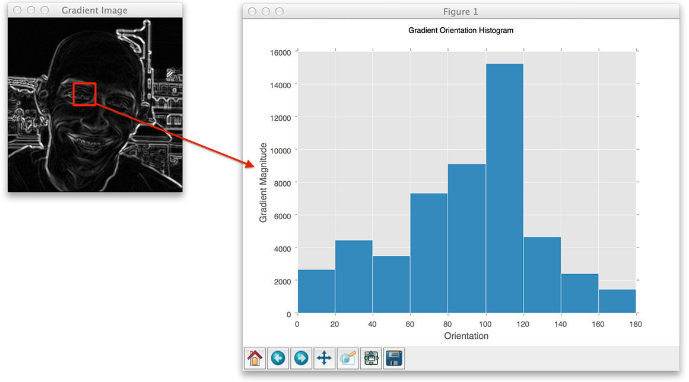


* the angles are between 0 and 180 degrees instead of 0 to 360 degrees. These are called **“unsigned” gradients** because a gradient and its negative are represented by the same numbers, we don’t use the 0 – 360 degrees because Empirically unsigned gradients work better than signed gradients.
* We can now draw this vector as an arrow on the image and this operation is applied to each pixel in the image.

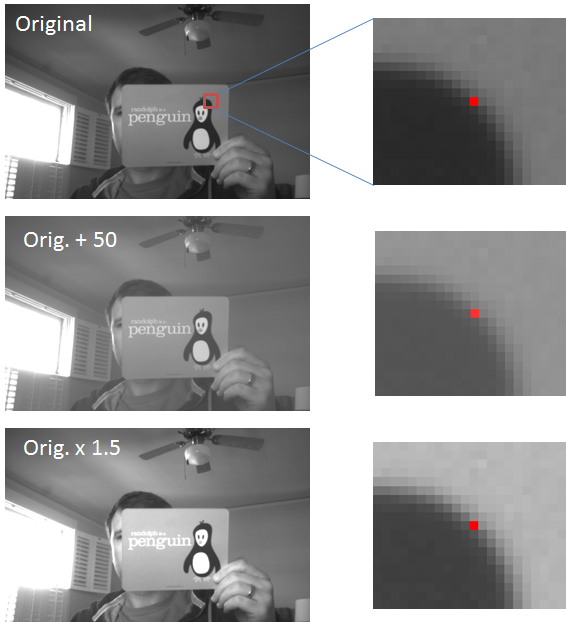




* Combining histogram.
* We distribute the gradient vectors on n bins to contruct a histogram (n can be either 8, 9, 12, …) and we choose n which give us better accuracy.
* If we have 9 bins, they will be carry the angles (0, 20, 40, 60, 80, 100, 120, 140, 160)
* We use histogram with n = 8 according to this paper.
* The histogram will be formed with angles on horizontal axis and magnitude on vertical axis.
* For each gradient vector, its contribution to the histogram is given by the magnitude of the vector (so stronger gradients have a bigger impact on the histogram), We split the contribution between the two closest bins, for example, if a gradient vector has an angle of 85 degrees, then we add ¼ of its magnitude to the bin centred at 70 degrees, and ¾ of its magnitude to the bin centred at 90.



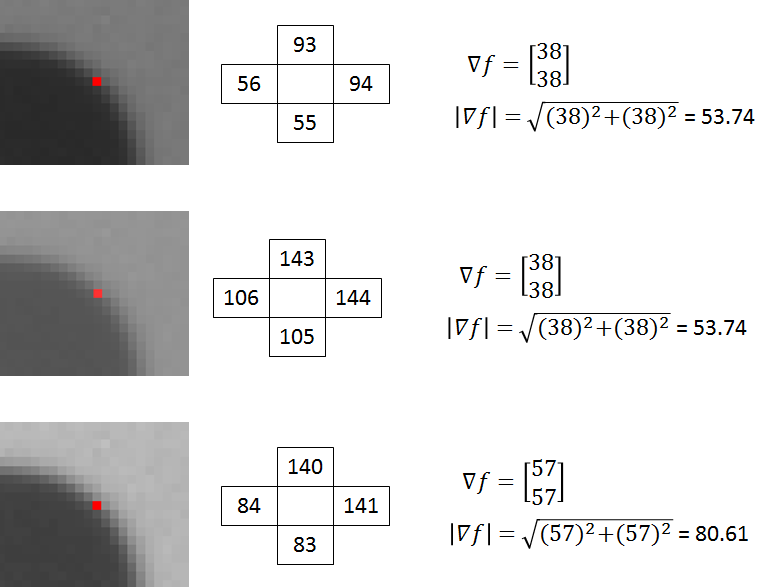
* How it’ll be applied on the image?
* We divide the image into cells of (1X1, 4X4, …..) and we choose the number of pixels per cell according to which give us the best accuracy.
* We use 12X12 pixels per cell.
* Now we have multiple histogram for a single image we combine them sequentially in a vector.
* Normalization
* Let’s take a moment to first look at the effect of normalizing gradient vectors in general.
* By normalizing your gradient vectors, you can make them invariant to *multiplications* of the pixel values (different illumination conditions)
* Ex:



(caption)[The first image shows a pixel, highlighted in red, the second image, all pixel values have been increased by 50, the third image, all pixel values in the original image have been multiplied by 1.5]

Notice how the third image displays an increase in contrast. The effect of the multiplication is that bright pixels became much brighter while dark pixels only became a little brighter, thereby increasing the contrast between the light and dark parts of the image.

Let’s look at the actual pixel values and how the gradient vector changes in these three images. The numbers in the boxes below represent the values of the pixels surrounding the pixel marked in red.



* The gradient vectors are equivalent in the first and second images, but in the third, the gradient vector magnitude has increased by a factor of 1.5.

If you divide all three vectors by their respective magnitudes, you get the same result for all three vectors: [0.71  0.71]’.

* By dividing the gradient vectors by their magnitude we can make them invariant (or at least more robust) to changes in contrast. (normalizing to unit length)
* Normalizing a vector does not affect its orientation, only the magnitude.
* **Histogram normalization**

Rather than normalize each histogram individually, the cells are first grouped into blocks and normalized based on all histograms in the block, Each block consists of nXn (4X4 in our case) cells 50% overlapping, The effect of the block overlap is that each cell will appear multiple times in the final descriptor but normalized by a different set of neighbouring cells.

* Ex :



[caption](2X2, 50% overlapping)

* All chosen parameters according to this paper :

<https://link.springer.com/content/pdf/10.1007%2F978-3-642-02172-5_24.pdf>