CS-GY 6643 **E. K. Wong**
Project 2: HOG Feature **Fall 2015**
(Revised 12-15-2015)

Total points = 100.

Project Description. In this project, you will implement the *Histograms of Oriented Gradients (HOG)* feature for detecting human in 2D images. Assume that the input image *I* to your program is of size *R* (row) × *C* (column) pixels, with *R* and *C* multiples of 8. First compute the gradient angle and normalized edge magnitude as in project 1. Use the image display coordinate system with *r* pointing downward and *c* pointing to the right and compute the horizontal and vertical gradients by using templates [-1, 0, 1] and $[-1, 0, 1]^T$. Use the center pixel (with value 0) as reference point in both templates. The gradient angle can be computed by the formula

$$\theta = \arctan\left(\frac{I_r}{I_c}\right)$$

with I_r and I_c representing the vertical and horizontal gradients, respectively. Next negate the computed angle (i.e., $\theta' = -\theta$) so θ'will be with respect to the (*x, y*) coordinate system with *x* points to the right and *y* points upward. The normalized edge magnitude is computed by the formula

$$M(r,c) = \frac{1}{\sqrt{2}}\sqrt{I_r^2 + I_c^2}$$

and then rounded off to the nearest integer. For those locations when the templates go outside of the boundary of the image, you can omit the computation of gradients and simply assign a value of 0 for edge magnitude and let the gradient angle value be *undefined.* If both I_r and I_c are zero, let the magnitude equals zero and let the gradient angle be undefined. Refer to lecture notes on how to compute the HOG feature. Do not include undefined values when forming the histogram for the cells in the HOG feature. Use the following parameter values in your implementation: *cell size* = 8 x 8 pixels, *block size* = 16 x 16 pixels (or 2 x 2 cells), *block overlap step size* = 8 pixels (or 1 cell.) Use *L2* norm for block normalization. Leave the histogram and final descriptor values as floating point numbers. No need to round off to integers.

A set of **20** training images (cut from detection windows) containing 10 positive and 10 negative samples will be provided. A set of 10 test images containing 5 positive and 5 negative samples will also be provided to test your classifier (described below.)

- Compute the HOG descriptor for each of the training and test images. Output the HOG descriptor for the following 4 training images (2 positive and 2 negative): crop001030c.bmp, crop001034b.bmp, 00000003a_cut.bmp, and 00000057a_cut.bmp. Also, output the HOG descriptor for the following test images (1 positive and 1 negative): crop001008b.bmp and 00000053a_cut.bmp.
- Compute the mean descriptor for the positive training images and the mean descriptor for the negative training images. Next, compute the Euclidean distance from each positive sample to the mean descriptor of the positive samples, and compute the Euclidian distance from each negative sample to the mean descriptor of the negative samples. Output the mean descriptor for the positive samples and the mean descriptor for the negative samples. Also, output the Euclidean distance of each training sample to its respective mean.
- Start with some arbitrary *W* value, use the Error-Correcting procedure you have learned in class to train a 2-class linear classifier that will classify an input HOG descriptor to *human* or *non-human.* Output the initial *W*, the learning rate α, order of training samples,

the number of iterations through the training samples, and the classification results (either positive or negative) for each test image.

Put the computed Euclidian distances between training samples and their respective mean onto a MS Words or PDF document. Also, put the initial W, the learning rate α, order of training samples, the number of iterations through the training samples, and the classification results in the same Words or PDF document. Output the HOG descriptors and the mean descriptors as separate files. Use one *.txt* or text file for each descriptor. Output the histogram values for each block on a separate line, starting with the upper-left corner block, then go left-to-right and top-to-bottom. Since there are 105 blocks, your file should contain 105 lines with each line containing 36 floating-point numbers separated by blank space. Round off the floating-point numbers to 2 digits after the decimal when writing to the text file.

Hand in the source code (with full comments and documentation), executable code, and output files as described above. Indicate on the Words or PDF file which compiler was used and provide instructions on how to compile and run your program. **Points will be taken off if any of these are missing.** **C**/C++, Matlab, Python, and Java are the recommended languages. If you would like to use other languages, send me an e-mail first. Except for reading/writing images, **you are not allowed to use any built-in library functions for any of the steps you are required to implement (including functions to compute convolutions or cross-correlations.)** Save a backup copy of the files you hand in.