CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

1 Submission

- Assignment due: Feb 15 (11:55pm)
- Individual assignment
- 1 page summary write-up with resulting visualization (more than 1 page assignment will be automatically returned.).
- Submission through Canvas.
- List of submission codes:
  - `HOG.m`
  - `GetDifferentialFilter.m`
  - `GetGradient.m`
  - `BuildHistogram.m`
  - `GetBlockDescriptor.m`
- The function that does not comply with its specification will not be graded.
- You are not allowed to use any high level MATLAB built-in function of image processing and computer vision, e.g., `imfilter` and `conv2` except for IO functions such as `imread`, `imwrite`, and `imagesc`. Please consult with TA if you are not sure about the list of allowed functions.

CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

2 HOG

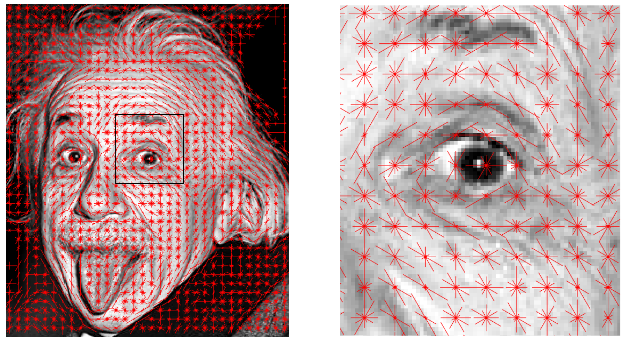


Figure 1: Histogram of oriented gradients. HOG feature is extracted and visualized for (a) the entire image and (b) zoom-in image. The orientation and magnitude of the red lines represents the gradient components in a local cell.

In this assignment, you will implement a variant of HOG (Histogram of Oriented Gradients) in MATLAB proposed by Dalal and Trigg [1] (2015 Longuet-Higgins Prize Winner). It had been long standing top representation (until deep learning) for the object detection task with a deformable part model by combining with a SVM classifier [2]. Given an input image, your algorithm will compute the HOG feature and visualize as shown in Figure 1 (the line directions are perpendicular to the gradient to show edge alignment). The orientation and magnitude of the red lines represents the gradient components in a local cell.

`function [hog] = HOG(im)`
Input: input gray-scale image with `uint8` format.
Output: HOG descriptor.
Description: You will compute the HOG descriptor of input image `im`. The pseudocode can be found below:

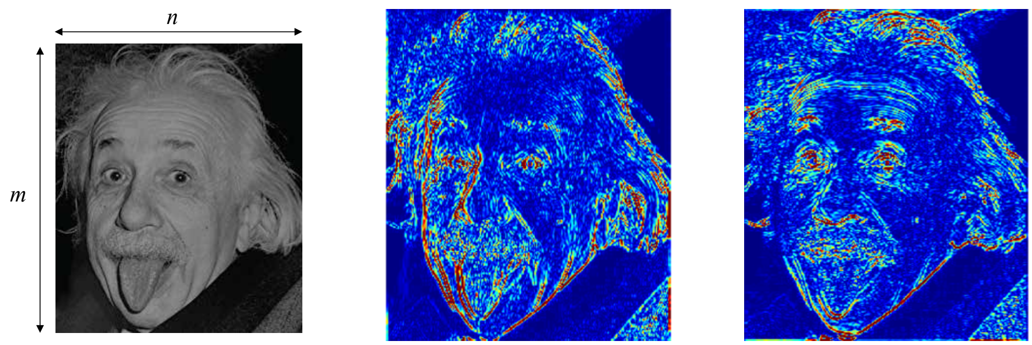
Algorithm 1 HOG

1: Convert the gray-scale image to `double` format.
2: Get differential images using `GetDifferentialFilter` and `FilterImage`
3: Compute the gradients using `GetGradient`
4: Build the histogram of oriented gradients for all cells using `BuildHistogram`
5: Build the descriptor of all blocks with normalization using `GetBlockDescriptor`
6: Return a long vector (`hog`) by concatenating all block descriptors.

CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

2.1 Image filtering



(a) Input image

(b) Differential along x direction

(c) Differential along y direction

Figure 2: (a) Input image dimension. (b-c) Differential image along x and y directions.

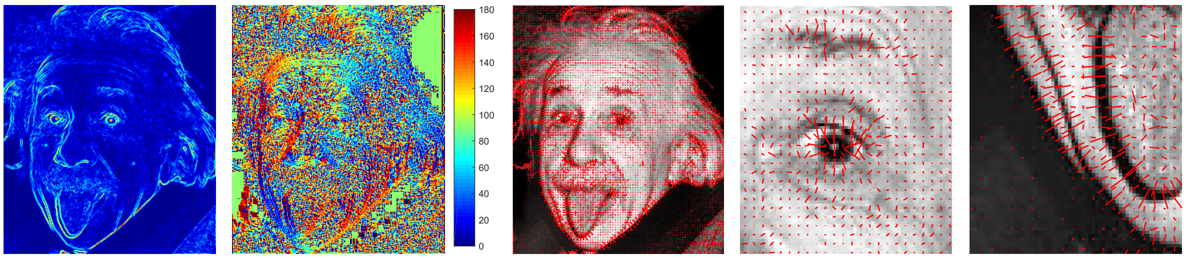
`function [filter_x, filter_y] = GetDifferentialFilter()`
Input: none.
Output: `filter_x` and `filter_y` are 3×3 filters that differentiate along x and y directions, respectively.
Description: You will compute the gradient by differentiating the image along x and y directions. This code will output the differential filters.

`function [im_filtered] = FilterImage(im, filter)`
Input: `im` is the gray scale $m \times n$ image (Figure 2(a)) converted to double (refer to `im2double` built-in function); `filter` is a filter ($k \times k$ matrix)
Output: `im_filtered` is $m \times n$ filtered image. You may need to pad zeros on the boundary on the input image to get the same size filtered image.
Description: Given an image and filter, you will compute the filtered image. Given the two functions above, you can generate differential images by visualizing the magnitude of the filter response as shown in Figure 2(b) and 2(c).

CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

2.2 Gradient Computation



(a) Magnitude (b) Angle (c) Gradient (d) Zoomed eye (e) Zoomed neck

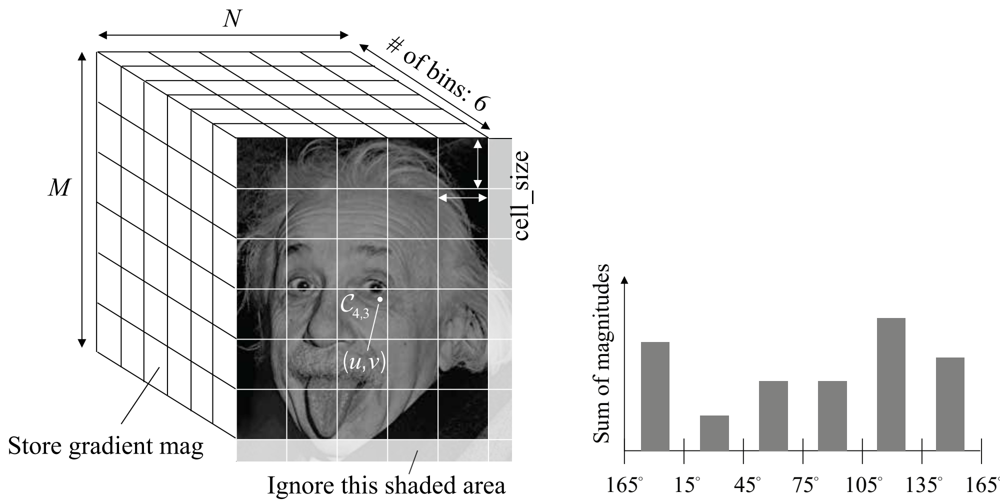
Figure 3: Visualization of (a) magnitude and (b) orientation of image gradients. (c-e) Visualization of gradients at every 3rd pixel (the magnitudes are re-scaled for illustrative purpose.).

`function [grad_mag, grad_angle] = GetGradient(im_dx, im_dy)`
Input: `im_dx` and `im_dy` are the x and y differential images (size: $m \times n$).
Output: `grad_mag` and `grad_angle` are the magnitude and orientation of the gradient images (size: $m \times n$). Note that the range of the angle should be $[0, \pi)$, i.e., unsigned angle ($\theta == \theta + \pi$).
Description: Given the differential images, you will compute the magnitude and angle of the gradient. Using the gradients, you can visualize and have some sense with the image, i.e., the magnitude of the gradient is proportional to the contrast (edge) of the local patch and the orientation is perpendicular to the edge direction as shown in Figure 3.

CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

2.3 Orientation Binning



(a) ori_histo (b) Histogram per cell

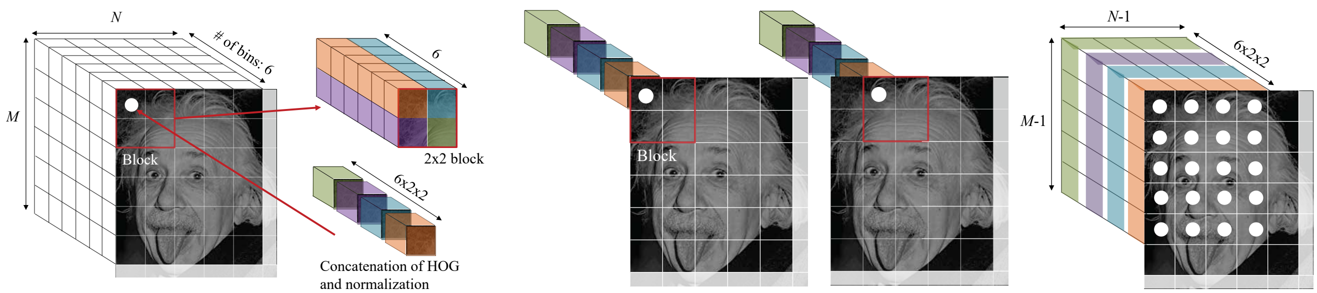
Figure 4: (a) Histogram of oriented gradients can be built by (b) binning the gradients to corresponding bin.

`function ori_histo = BuildHistogram(grad_mag, grad_angle, cell_size)`
Input: `grad_mag` and `grad_angle` are the magnitude and orientation of the gradient images (size: $m \times n$); `cell_size` is the size of each cell, which is a positive integer.
Output: `ori_histo` is a 3D tensor with size $M \times N \times 6$ where M and N are the number of cells along y and x axes, respectively, i.e., $M = \lfloor m/\texttt{cell\_size} \rfloor$ and $N = \lfloor n/\texttt{cell\_size} \rfloor$ where $\lfloor \cdot \rfloor$ is the round-off operation as shown in Figure 4(a).
Description: Given the magnitude and orientation of the gradients per pixel, you can build the histogram of oriented gradients for each cell.

$$\texttt{ori\_histo}(i, j, k) = \sum_{(u,v)\in\mathcal{C}_{i,j}} \texttt{grad\_mag}(u, v) \quad \text{if } \texttt{grad\_angle}(u, v) \in \theta_k \tag{1}$$

where $\mathcal{C}_{i,j}$ is a set of x and y coordinates within the (i, j) cell, and θ_k is the angle range of each bin, e.g., $\theta_1 = [165°, 180°) \cup [0°, 15°)$, $\theta_2 = [15°, 45°)$, $\theta_3 = [45°, 75°)$, $\theta_4 = [75°, 105°)$, $\theta_5 = [105°, 135°)$, and $\theta_6 = [135°, 165°)$. Therefore, `ori_histo(i,j,:)` returns the histogram of the oriented gradients at (i, j) cell as shown in Figure 4(b). Using the `ori_histo`, you can visualize HOG per cell where the magnitude of the line proportional to the histogram as shown in Figure 1. Typical `cell_size` is 8.

2.4 Block Normalization



(a) Block descriptor (b) Block overlap with stride 1

Figure 5: HOG is normalized to account illumination and contrast to form a descriptor for a block. (a) HOG within (1,1) block is concatenated and normalized to form a long vector of size 24. (b) This applies to the rest block with overlap and stride 1 to form the normalized HOG.

`function ori_histo_normalized = GetBlockDescriptor(ori_histo, block_size)`
Input: `ori_histo` is the histogram of oriented gradients without normalization. `block_size` is the size of each block (e.g., the number of cells in each row/column), which is a positive integer.
Output: `ori_histo_normalized` is the normalized histogram (size: $(m-1) \times (n-1) \times (6 \times \texttt{block\_size}^2)$.
Description: To account for changes in illumination and contrast, the gradient strengths must be locally normalized, which requires grouping the cells together into larger, spatially connected blocks (adjacent cells). Given the histogram of oriented gradients, you apply L_2 normalization as follow:

1. Build a descriptor of the first block by concatenating the HOG within the block. You can use `block_size`=2, i.e., 2×2 block will contain $2 \times 2 \times 6$ entries that will be concatenated to form one long vector as shown in Figure 5(a).
2. Normalize the descriptor as follow:
$$\hat{h}_i = \frac{h_i}{\sqrt{\sum_i h_i^2 + e^2}} \tag{2}$$
where h_i is the i^{th} element of the histogram and $\hat{h}_i$ is the normalized histogram. e is the normalization constant to prevent division by zero (e.g., $e = 0.001$).
3. Assign the normalized histogram to `ori_histo_normalized`(1,1) (white dot location in Figure 5(a)).
4. Move to the next block `ori_histo_normalized`(1,2) with the stride 1 and iterate 1-3 steps above.

The resulting `ori_histo_normalized` will have the size of $(m-1) \times (n-1) \times 24$.

CSCI 5561: Assignment #1

Histogram of Oriented Gradients (HOG)

References

[1] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *CVPR*, 2005.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *TPAMI*, 2010.