HOG In this assignment, a variant of HOG proposed by Dalal and Triggs (2005) is implemented. We follow the Algorithm 1 (`extract_hog`) for computing the HOG descriptor:

(1) `get_differential_filter`: Get the sobel differential filters:

$$\texttt{filter\_x} = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}, \quad \texttt{filter\_y} = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}.$$

(2) `filter_image`: Compute the gradient along x and y directions. Zeros are padded on the boundary on the input image to get the same size filtered image.

(3) `get_gradient`: Compute the magnitude and angle of the gradient. The range of the angle is $[0, \pi)$, i.e., for $\theta \in [-\pi, 0)$, set $\theta = \theta + \pi$; for $\theta = \pi$, set $\theta = 0$.

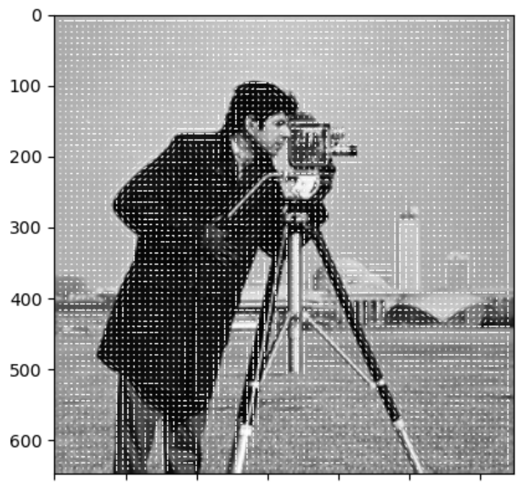
(4) `build_histogram`: Divide the image to cells based on the `cell_size` (typical is 8). Build the histogram of oriented gradients for each cell with the 6 bins given by $\theta_1 = [165, 180) \cup [0, 15)$, $\theta_2 = [15, 45)$, $\theta_3 = [45, 75)$, $\theta_4 = [75, 105)$, $\theta_5 = [105, 135)$, $\theta_6 = [135, 165)$.

(5) `get_block_descriptor`: HOG is locally normalized to account for changes in illumination and contrast. Group the cells together into larger, spatially connected blocks (use `block_size`=2) and apply L_2 normalization to the descriptor ($e = 0.001$).

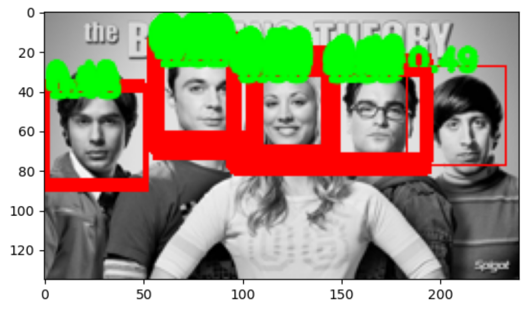
Figure 1a shows the figure of visualizing the HOG discriptor of the '`cameraman.tif`' image using the `visualize_hog` function.

Application: Face Detection By using the HOG descriptor, we can use a simgle template image '`template.png`' (50×50) to detect faces in the target image '`target.png`' (135×240). First, measure the **normalized cross-corrlation (NCC)** between the HOG descriptors from the template and target image patches. Then **thresholding** on NCC scores using threshold value 0.48, i.e., all the bounding boxes with NCC scores < 0.48 will be deleted. As thresholding will produce many overlapping bouding boxes, we use **non-maximum suppression** with IoU 50% to detect correct bounding boxes for faces.

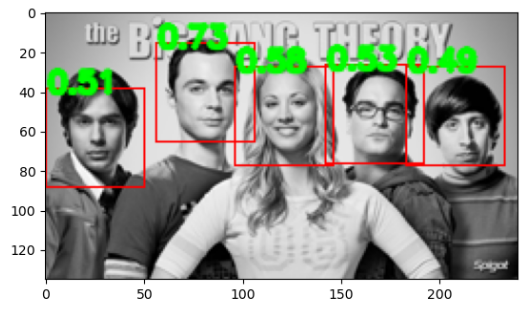
We need to compute the HOG descriptors of all target patches, which is $(135-50+1)*(240-50+1) = 16426$ patches in total. It took around 10 minutes to run the `face_recognition` function. Figure 1b shows the plot of visualizing the bounding boxes after thresholding on NCC scores. We can see many overlapping bounding boxes. Figure 1c shows the plot of bounding boxes obtained by non-maximum suppression. We can see that all the five faces are correctly detected and each face is detected only once.



(a)



(b)



(c)