Indian Institute of Science Education and Research
Bhopal

Computer Vision(DSE-312/EECS-320)

Assignment-2

Name: Jiya Sinha
Roll No.: 22161

Time of submission: Marks Obtained:

Please follow the instructions given in the assignment carefully.

Please provide your detailed answers and any explanations or diagrams directly below each question in the 'Answers' section.

1. Question 1

   Answer:

Algorithm 1 SIFT (Scale-Invariant Feature Transform)

Require: Input image I, number of octaves n_o, images per octave n_i, initial σ, contrast threshold t_c, edge response threshold R_{th}, window size w.

Ensure: Keypoints and descriptors.

procedure SIFT(I)

Initialize n_o, n_i, σ, t_c, R_{th}, w.

Convert input image I to grayscale.

Initialize *Gaussian Pyramid* and *Difference of Gaussians (DoG) Pyramid*.

for $o = 1$ *to* n_o **do**

Generate $n_i + 3$ Gaussian-blurred images using

$$k = 2^{\frac{1}{n_i}}, \quad \sigma_i = \sigma \cdot k^i$$

Compute DoG images for each pair of consecutive Gaussian images:

$$\text{DoG} = G_{\sigma_{i+1}} - G_{\sigma_i}$$

Add Gaussian and DoG images to their respective pyramids.

Initialize *keypoints* and *descriptors.* **for *each*** *DoG octave in DoG Pyramid* **do**

Identify candidate keypoints by finding local extrema in a 3D neighborhood. **for *each*** *candidate keypoint* (x, y, s) **do**

Localize keypoints by solving:

$$\mathbf{J} = -\mathbf{H}^{-1}\mathbf{d}, \quad \mathbf{H} = \begin{bmatrix} \partial_x^2 & \partial_{xy}^2 & \partial_{xs}^2 \\ \partial_{xy}^2 & \partial_y^2 & \partial_{ys}^2 \\ \partial_{xs}^2 & \partial_{ys}^2 & \partial_s^2 \end{bmatrix}$$

Compute contrast $C = D(x, y, s) + 0.5 \cdot \mathbf{J}^T \cdot \mathbf{d}$. **if** $|C| < t_c$ *or edge response exceeds* R_{th} **then**

Discard keypoint. **else**

Compute descriptor using gradient magnitude and orientation within a neighborhood.

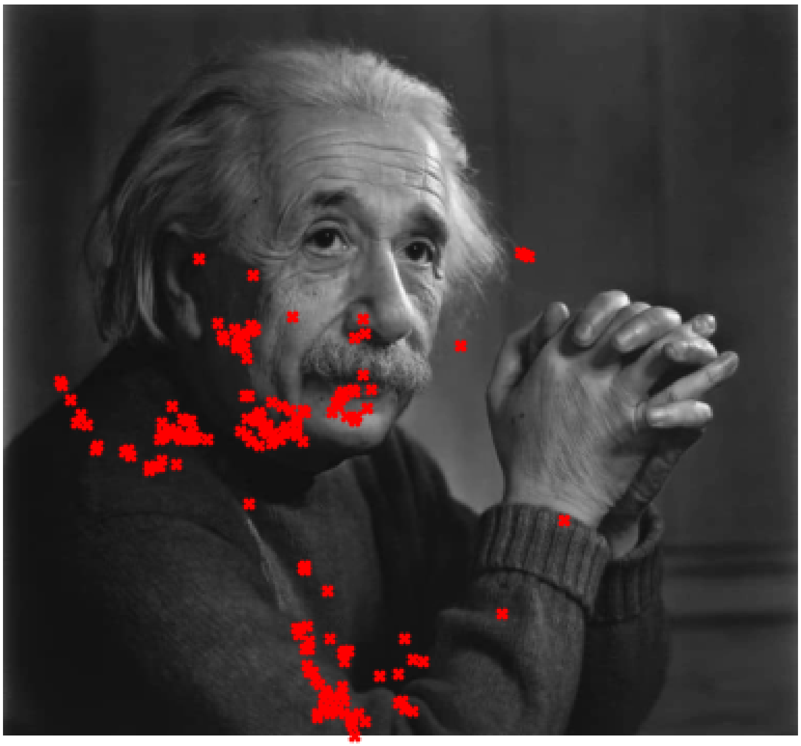
Normalize descriptor for scale invariance.

Add keypoint and descriptor to final list.

Visualize keypoints overlaid on the input image.

return Keypoints and Descriptors.

end procedure



Algorithm 2 Bag of Visual Words Algorithm for an Image

Input: Image I, number of clusters k

Output: Normalized histogram H, cluster centers C

Step 1: Extract SIFT Keypoints and Descriptors Detect keypoints in the image I using SIFT Compute descriptors for each keypoint, $D = \{d_1, d_2, \ldots, d_n\}$

Step 2: Cluster Descriptors using K-Means Apply K-Means clustering to D with k clusters Obtain cluster centers $C = \{c_1, c_2, \ldots, c_k\}$ and assign each descriptor d_i to its nearest cluster center

Step 3: Construct a Histogram Initialize a histogram H with k bins: $H[j] = 0$ for $j = 1, 2, \ldots, k$ **foreach** *descriptor* d_i **do**

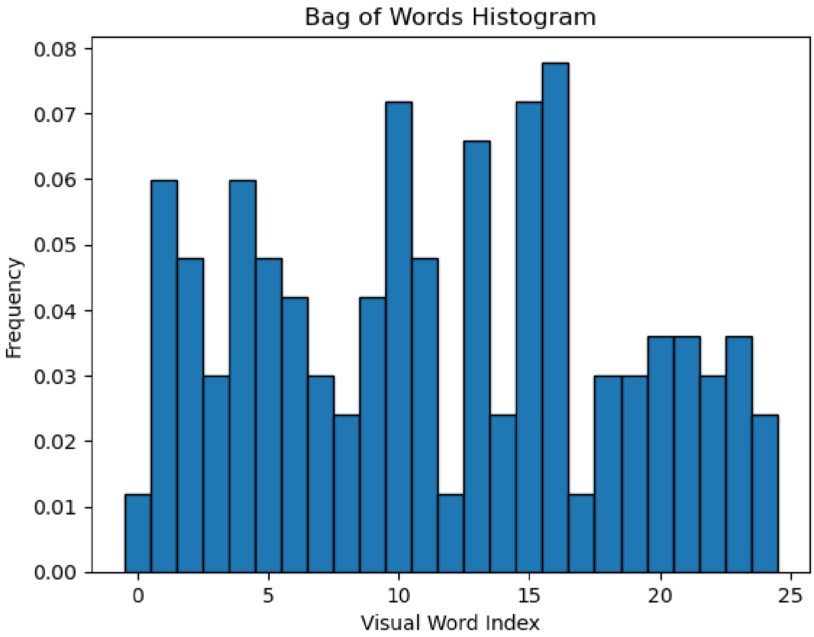
    Determine the cluster index j for d_i Increment $H[j] \leftarrow H[j] + 1$

end

Normalize H so that $\sum_{j=1}^{k} H[j] = 1$

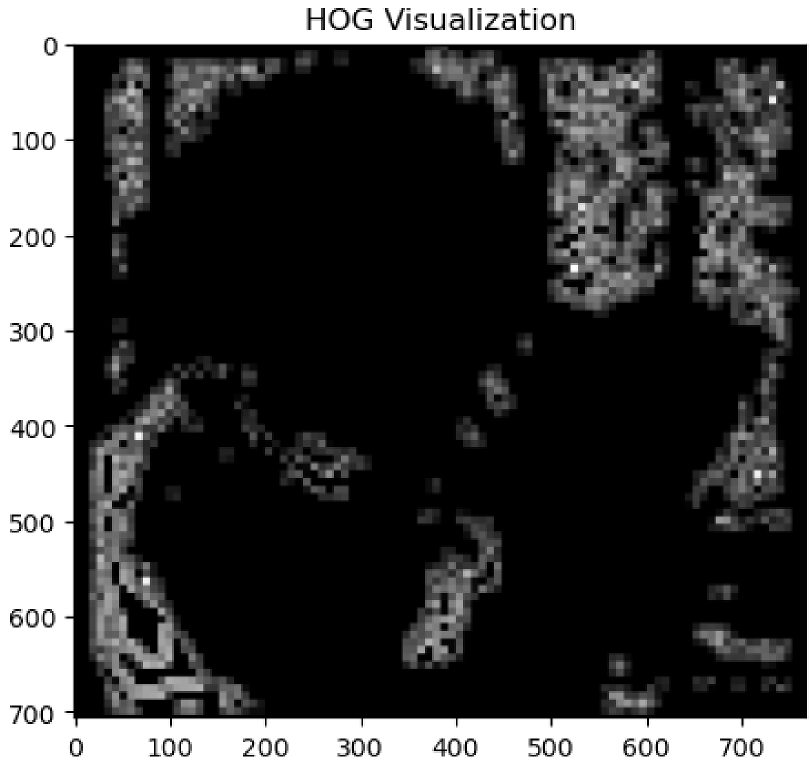
Step 4: Visualize the Histogram Plot H as a bar graph for visualization

return H, C



Algorithm 3 Histogram of Oriented Gradients (HOG) Algorithm

Input: Image I, Cell size (h_c, w_c), Block size (h_b, w_b), Number of bins n
Output: HOG descriptor H
Step 1: Preprocess Image **if** *I is a color image* **then**
  Convert I to grayscale by averaging RGB channels
end
Resize I so that its dimensions are divisible by the cell size (h_c, w_c)
Step 2: Compute Gradients Use Sobel filters to compute gradients ∇_x and ∇_y Calculate magnitude $M = \sqrt{\nabla_x^2 + \nabla_y^2}$ and angle $\theta = \arctan 2(\nabla_y, \nabla_x)$
Step 3: Compute Cell Histograms Initialize histogram H_c for each cell **foreach** *cell (i, j) of size (h_c, w_c)* **do**
  Extract gradients within the cell Quantize angles θ into n bins (e.g., $0°$ to $360°$ divided equally) Populate the histogram using gradient magnitudes M as weights
end
Step 4: Normalize Block Histograms Group neighboring cells into overlapping blocks of size (h_b, w_b) **foreach** *block (i, j)* **do**
  Concatenate histograms of cells in the block Normalize the block histogram using L_2 normalization
end
Step 5: Construct Final Descriptor Flatten all normalized block histograms into a single vector H
return H



2. Question 2

   Answer: Optical Flow Computation using Lucas-Kanade Method

Algorithm 4 Optical Flow Computation using Lucas-Kanade Method

Input: Video frames $\{I_t\}$, background model, window size w, number of background frames
Output: Optical flow vectors (u, v) representing horizontal and vertical velocities
procedure Initialize Background
    Read first n frames from the video
    Convert frames to grayscale
    Compute the median of the frames to initialize background model
end procedure
procedure Gradient Computation
    For each pixel in the previous and current frame:
    Calculate I_x (gradient in x-direction), I_y (gradient in y-direction), and I_t (time gradient) **for** *each pixel* (i, j) *in the frame* **do**
        $I_x[i,j] = \frac{I[i,j+1]-I[i,j-1]}{2}$ ▷ Gradient in x-direction
    $I_y[i,j] = \frac{I[i+1,j]-I[i-1,j]}{2}$ ▷ Gradient in y-direction
    $I_t[i,j] = I_{current}[i,j] - I_{previous}[i,j]$ ▷ Time gradient

end procedure
procedure Lucas-Kanade Optical Flow
    For each pixel (i, j) in the frame, apply the following steps:
    Construct A and b matrices for the optical flow equation:

$$A = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}, \quad b = \begin{pmatrix} -I_x I_t \\ -I_y I_t \end{pmatrix}$$

    Solve the optical flow equation $A \cdot v = b$ to get velocity components:

$$v = \begin{pmatrix} u \\ v \end{pmatrix} = A^{-1} \cdot b$$

    If A is invertible (i.e., $\det(A) \neq 0$), compute the horizontal (u) and vertical (v) velocities for the pixel
    If A is not invertible, set $u[i,j] = 0$ and $v[i,j] = 0$ (handle singular matrix case)
end procedure
procedure Foreground Detection
    Compute foreground mask by subtracting the background from the current frame:

$$\text{fg\_mask} = |I_{background} - I_{current}|$$

    Threshold the foreground mask to highlight moving objects
    Create an overlay where the foreground pixels are highlighted (e.g., green color)
end procedure
procedure Overlay and Visualize
    Combine the original frame with the foreground mask overlay
    Draw optical flow vectors (arrows) representing the direction and speed of movement
end procedure
procedure Process Video
    Initialize background using Initialize Background() **for** *each frame in the video* **do**
        Convert current frame to grayscale
    Perform gradient computation using Gradient Computation()
    Compute optical flow using Lucas-Kanade Optical Flow()
    Detect foreground using Foreground Detection()
    Create overlay with foreground mask using Overlay and Visualize()

Final Output The output video includes:

- A foreground mask highlighting the moving objects with a green overlay