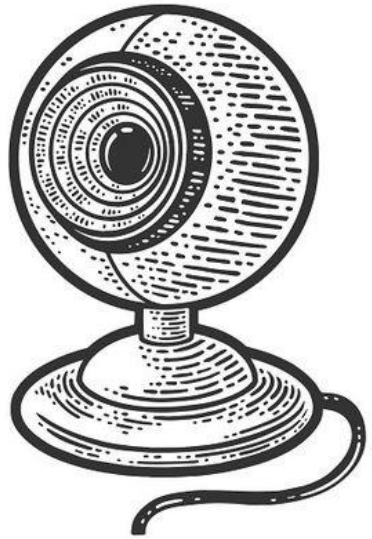
Webcam Image Processing – Final Project Presentation (PDF)

- **Project Name:** Webcam Image Processing Final Project
- **Student Name:** Alaa Abdelkader
- **ID:** 223101266
- **Instructor Name:** Dr. Mahmoud Zaki
- **Course:** Image Processing
- **Date:** 2025



1. Introduction

- “This project demonstrates real-time image processing using a webcam, including mirrored video feed, original color display, Canny edge detection with adjustable thresholds, and frequency domain filtering (low-pass, high-pass, band-pass, and band-reject).

  applications:
  image enhancement, edge detection, real-time filtering.

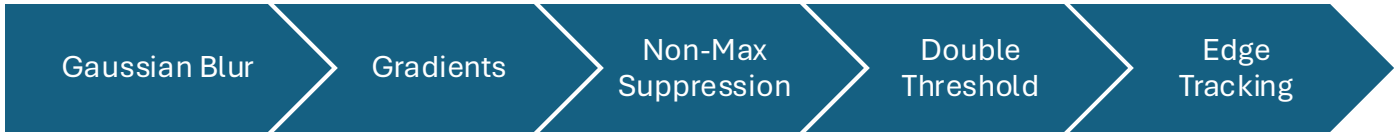
2. Libraries Used

library	
cv2	Video capture, image processing, DFT/IDFT
numpy	Array and matrix operations, mathematical calculations
matplotlib	(Optional) plotting or visualizing frequency spectra

3. System Workflow

Webcam → Mirror → Grayscale → Canny / Frequency Filters → Display

Canny Pipeline:



Frequency Filters:

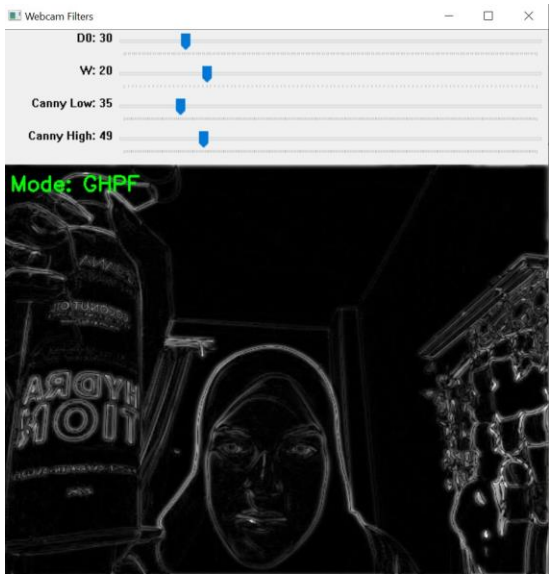
4. Features Implemented

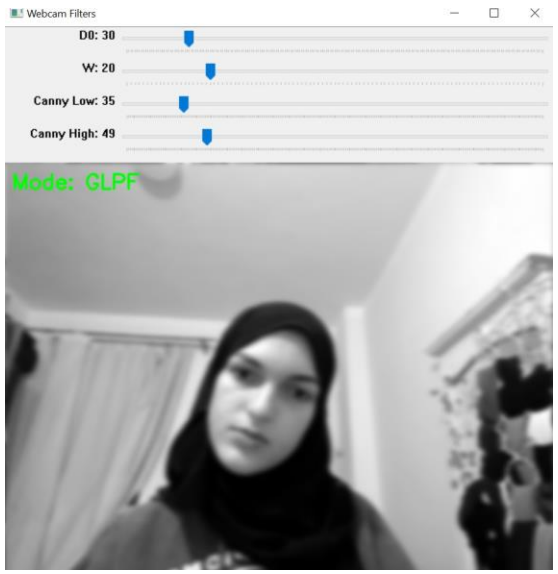
- **A. Real-Time Webcam Feed**
- Mirrored video for natural user experience.
- Keyboard shortcuts to switch between modes.
- **B. Canny Edge Detection**
- Adjustable thresholds via trackbars.
- Custom implementation demonstrates full pipeline.
- Optional: overlay of edge detection steps for explanation.
- **C. Frequency Domain Filtering**
- **Filters Implemented:**
  - Ideal, Gaussian, Butterworth (Low-Pass / High-Pass)
  - Band-Pass / Band-Reject
- Cutoff frequency (D0) and bandwidth (W) adjustable in real time.

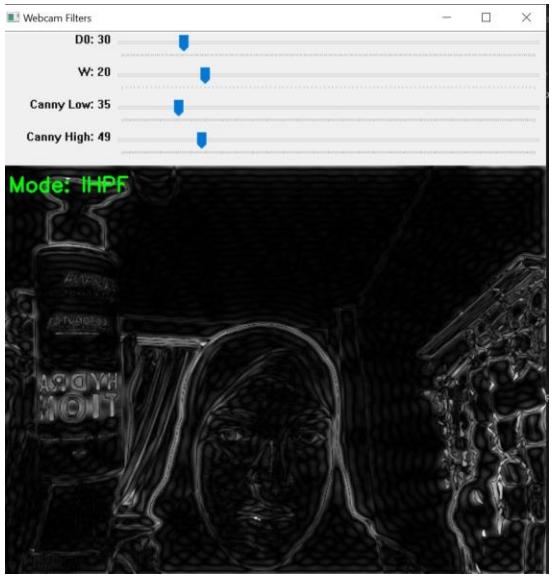
D. User Interaction

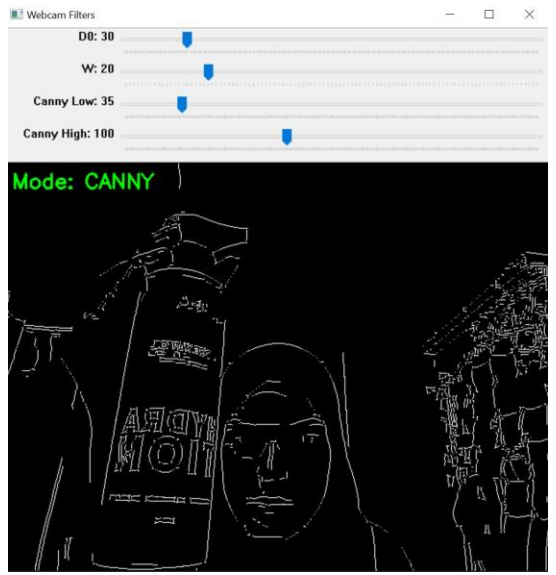
- Trackbars: D0, W, Canny thresholds
- Keyboard controls:

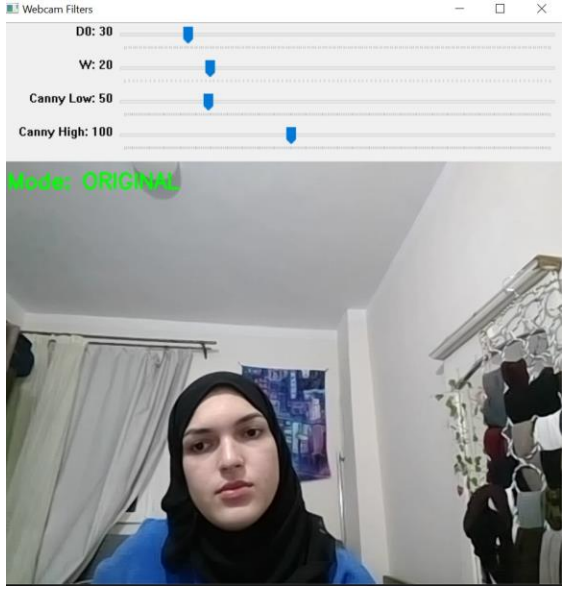
```
o: Original
c: Canny
1-8: Frequency filters
q: Quit
```





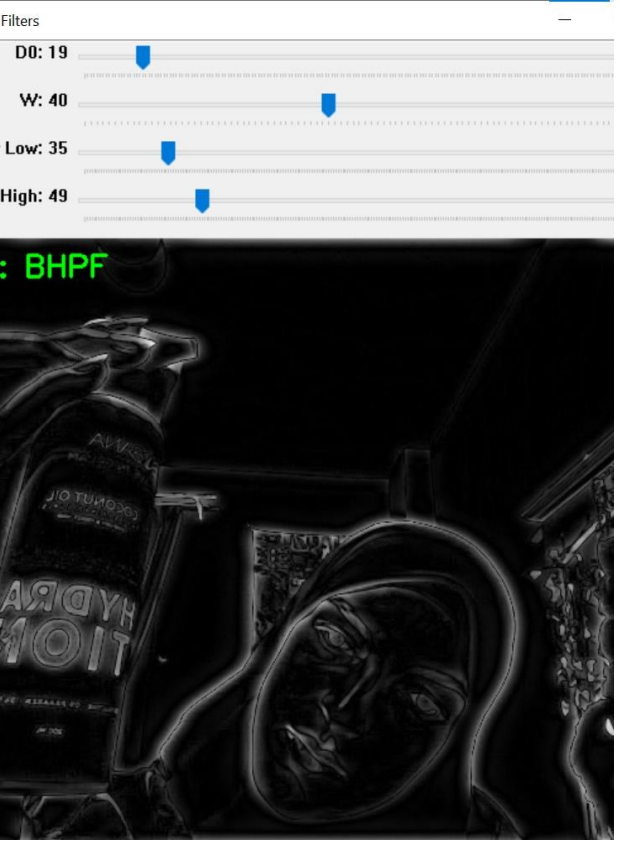


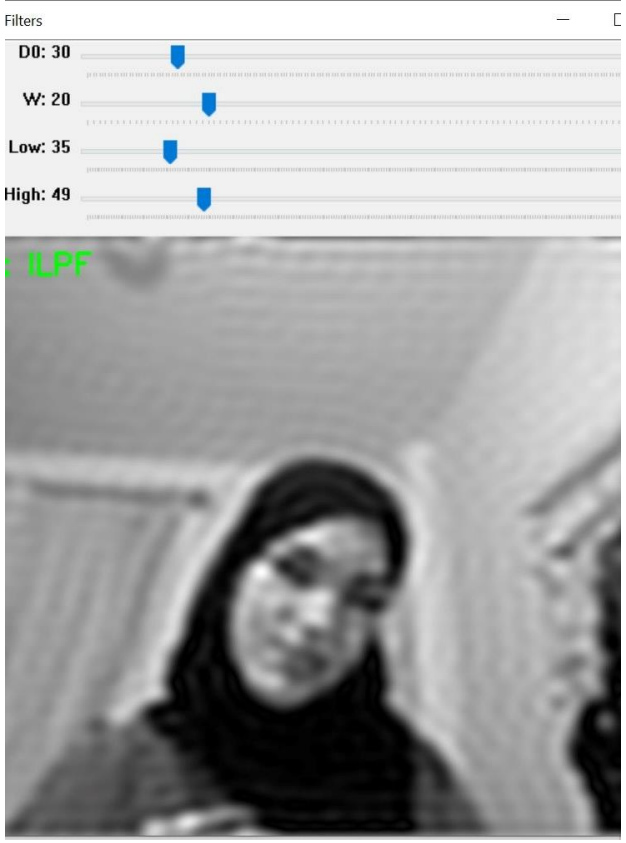


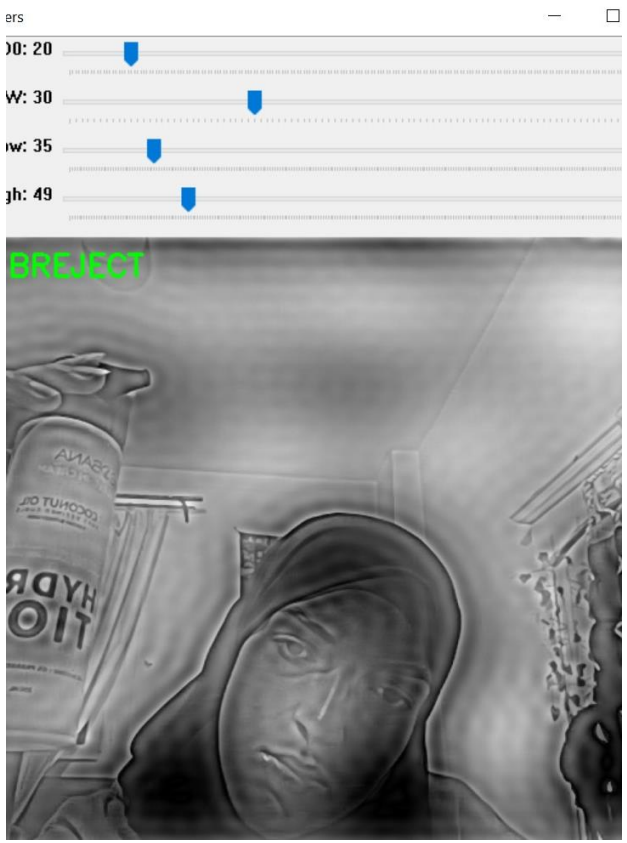


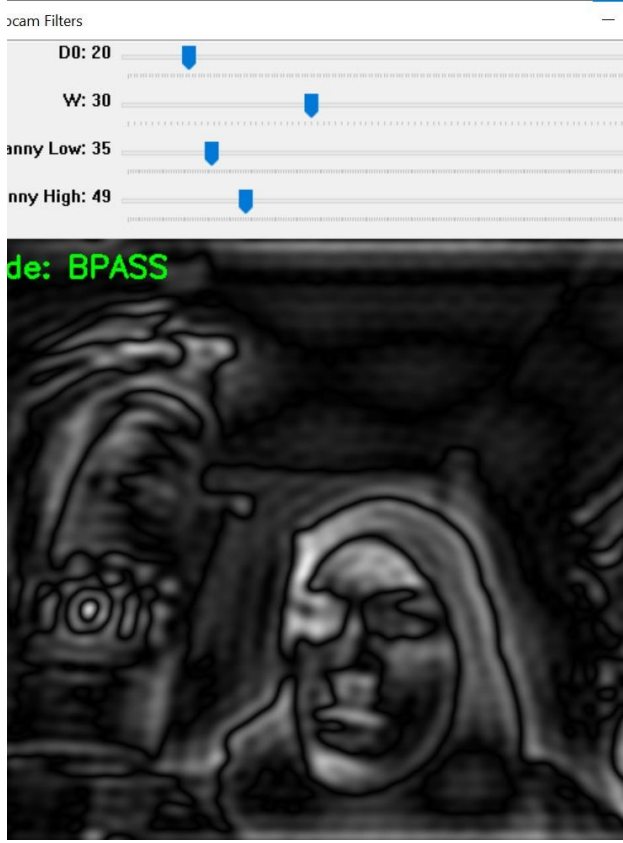
5. Visualizations

- Original
- Canny
- Gaussian blur(hp,lp)
- Ideal (hp)
- Butterworth(lp)









visualization

- Rest of functions

7. Summary

- Real-time webcam processing pipeline implemented successfully.
- Demonstrates both spatial domain (Canny) and frequency domain (DFT-based filters) techniques.
- Adjustable parameters allow interactive observation.
- Modular code facilitates future extensions.

Line-by-Line Code Documentation (Mapped to Project Presentation)

This section documents the implementation of the Webcam Image Processing project. Each line of code is explained in relation to the system workflow, Canny edge detection pipeline, and frequency-domain filtering techniques described in the presentation.

Code: import cv2
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: import numpy as np
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Gaussian Blur --- #Smooths the image to *reduce noise* before edge detection.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def gaussian_blur(image, ksize=5):#Defines a function gaussian_blur that takes an image and a kernel size ksize; larger values produce more smoothing.(ksize)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return cv2.GaussianBlur(image, (ksize, ksize), 0) #defines the Gaussian window size (odd number required). The last parameter 0 means OpenCV automatically calculates the Gaussian sigma.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Gradient / Canny Functions ---
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def compute_gradients(image): #Defines a function to compute gradient magnitudes and directions.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: grad_x = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3) #applies Sobel operator to detect horizontal intensity changes.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: grad_y = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3) #applies Sobel operator to detect vertical intensity changes.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: grad_mag = np.sqrt(grad_x**2 + grad_y**2) #computes gradient magnitude using Euclidean distance formula.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: grad_angle = np.arctan2(grad_y, grad_x)#calculates angle of gradient using arctan2() (range $-\pi$ to π).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return grad_mag, grad_angle #Returns both magnitude and angle for later steps.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def non_maximum_suppression(grad_mag, grad_angle): #Thins edges by *suppressing non-maximal pixels* along the gradient direction.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: M, N = grad_mag.shape #Extracts image dimensions M (height) and N (width).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: Z = np.zeros((M, N), dtype=np.float32) #Creates an empty array Z to hold the thinned edges

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: angle = grad_angle * 180. / np.pi #Converts gradient angles from radians to degrees.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: angle[angle < 0] += 180 #Ensures angles lie between *0° and 180°*.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: for i in range(1, M-1):

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: for j in range(1, N-1): # Compare current pixel with neighbors in gradient direction Loops through the image, avoiding borders (to prevent index errors).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: q = r = 255 # Default comparison values if no direction matches.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if (0 <= angle[i,j] < 22.5) or (157.5 <= angle[i,j] <= 180): #Horizontal edge → compare left and right.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: q,r = grad_mag[i, j+1], grad_mag[i, j-1]
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif 22.5 <= angle[i,j] < 67.5:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: q,r = grad_mag[i+1, j-1], grad_mag[i-1, j+1] #Diagonal edge (down-right/up-left).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif 67.5 <= angle[i,j] < 112.5:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: q,r = grad_mag[i+1, j], grad_mag[i-1, j] #Vertical edge → compare above and below.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif 112.5 <= angle[i,j] < 157.5:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: q,r = grad_mag[i-1, j-1], grad_mag[i+1, j+1] #Other diagonal.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: Z[i,j] = grad_mag[i,j] if grad_mag[i,j] >= q and grad_mag[i,j] >= r else 0 #Keeps the pixel only if it is *the strongest* in its gradient direction. Else, sets it to 0 (non-edge).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return Z
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def double_threshold(image, low, high):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: strong, weak = 255, 75 #Defines intensity values for strong and weak edges.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: strong_edges = np.zeros_like(image, dtype=np.uint8) #pixels above the high threshold
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: weak_edges = np.zeros_like(image, dtype=np.uint8) #pixels between low and high threshold
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: strong_edges[image >= high] = strong
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: weak_edges[(image >= low) & (image < high)] = weak #Classifies edges by intensity.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return strong_edges, weak_edges
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def edge_tracking(strong_edges, weak_edges): # Weak edges connected to strong edges become strong

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: M, N = strong_edges.shape
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: result = np.copy(strong_edges) # Starts the final edge map as a copy of strong edges.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: weak, strong = 75, 255 #Defines pixel values for convenience.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: for i in range(1, M-1):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: for j in range(1, N-1):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if weak_edges[i,j] == weak: #Loops through all weak edge pixels.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if np.any(result[i-1:i+2, j-1:j+2] == strong):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: result[i,j] = strong # Looks at the 3x3 neighborhood.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Weak becomes strong if connected to a strong edge.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: else:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: result[i,j] = 0 #Otherwise, removed
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return result

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Final step of *Canny edge detection*.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Frequency Domain Filter Masks ---

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def ideal_low_pass(shape, D0): #Pass low frequencies, block high frequencies. (D0)->Cutoff frequency, adjustable via trackbar.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # shape: the spatial resolution (rows, cols).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # D0: cutoff frequency (radius of the pass region).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: rows, cols = shape #Unpacks the shape into number of rows and columns.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: crow, ccol = rows//2, cols//2 #* Computes the center of the frequency domain (DC component).

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: u = np.arange(rows) - crow #vertical frequencies

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: v = np.arange(cols) - ccol #horizontal frequencies

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: U, V = np.meshgrid(u, v, indexing='ij') #Creates a 2D grid of frequency coordinates:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: D = np.sqrt(U**2 + V**2) #Computes Euclidean distance from the center for each frequency bin.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = (D <= D0).astype(np.float32) #Sets mask = 1 inside the circle of radius D0, 0 outside. sharp cuttoff
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return mask[:, :, np.newaxis] #* Expands mask into 3D to match image channels (H × W × 1).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def ideal_high_pass(shape, D0): # Pass high frequencies, block low frequencies. (D0)->Cutoff frequency, adjustable via trackbar.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return 1 - ideal_low_pass(shape, D0) #High-pass is simply the complement of low-pass.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def gaussian_low_pass(shape, D0): #Defines a Gaussian low-pass mask.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: rows, cols = shape
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: crow, ccol = rows//2, cols//2
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: u = np.arange(rows) - crow
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: v = np.arange(cols) - ccol
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: U, V = np.meshgrid(u, v, indexing='ij')
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: D = np.sqrt(U**2 + V**2)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = np.exp(-(D**2) / (2*D0**2)) #Gaussian attenuation formula:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Smooth, no sharp edges
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Strong values near center
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Gradually falling
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return mask[:, :, np.newaxis].astype(np.float32) #Output as float32, with channel dimension.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def gaussian_high_pass(shape, D0): #Complement of Gaussian low-pass.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return 1 - gaussian_low_pass(shape, D0)

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def butterworth_low_pass(shape, D0, n=2):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: rows, cols = shape
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: crow, ccol = rows//2, cols//2
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: u = np.arange(rows) - crow
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: v = np.arange(cols) - ccol
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: U, V = np.meshgrid(u, v, indexing='ij')
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: D = np.sqrt(U**2 + V**2)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = 1 / (1 + (D/D0)**(2*n)) #* Butterworth formula:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Smoother than ideal
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Adjustable roll-off via n

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return mask[:, :, np.newaxis].astype(np.float32)

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def butterworth_high_pass(shape, D0, n=2): #n: filter order controlling steepness.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return 1 - butterworth_low_pass(shape, D0, n) #Complement of low-pass.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def band_reject(shape, D0, W): #Creates a band-reject filter (also called notch reject). D0: center frequency of rejection band. W: width of rejected band.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Coordinate grid:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: rows, cols = shape

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: crow, ccol = rows//2, cols//2

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: u = np.arange(rows) - crow

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: v = np.arange(cols) - ccol
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: U, V = np.meshgrid(u, v, indexing='ij')
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: D = np.sqrt(U**2 + V**2)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = np.ones((rows, cols), dtype=np.float32) #* Start with full pass (all ones).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask[(D >= D0 - W/2) & (D <= D0 + W/2)] = 0 #Zero-out the frequencies inside the rejected band.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return mask[:, :, np.newaxis] #* Add channel axis.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def band_pass(shape, D0, W): #* Inverse of band-reject: keeps only the frequencies inside the band.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: return 1 - band_reject(shape, D0, W)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Band filters use D0 and W (bandwidth) to selectively pass/reject frequency ranges.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Trackbar Callback ---
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: def nothing(x): #* It does nothing because you only need the trackbar values, not an active callback.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: pass
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Webcam Setup ---
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cap = cv2.VideoCapture(0) #* Opens the default webcam (device index 0).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'original' #Initializes display mode
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.namedWindow('Webcam Filters') #Creates a window where the webcam output will be shown.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.createTrackbar('D0', 'Webcam Filters', 30, 200, nothing) #Trackbar for cutoff frequency *D0* (used by LPF, HPF, Gaussian, Butterworth, etc.).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.createTrackbar('W', 'Webcam Filters', 20, 100, nothing) #Trackbar for *band width* (used in band-pass / band-reject).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.createTrackbar('Canny Low', 'Webcam Filters', 50, 255, nothing)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.createTrackbar('Canny High', 'Webcam Filters', 100, 255, nothing)

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Trackbars for Canny low and high thresholds.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: print("Keys: o=Original, c=Canny, 1=ILPF, 2=GLPF, 3=BLPF, 4=IHPF, 5=GHPF, 6=BHPF, 7=Band-Reject, 8=Band-Pass, q=Quit")

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Shows keyboard controls for switching between filter modes

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: while True:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: ret, frame = cap.read() #Reads one frame from the webcam.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if not ret:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: break #Breaks loop if camera fails.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # Runs forever until user presses a quit key.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: # --- Mirror the frame ---

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: frame = cv2.flip(frame, 1) #Flips horizontally to create a mirror effect.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY) # Converts frame to grayscale for processing.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: D0 = cv2.getTrackbarPos('D0', 'Webcam Filters') #Gets low-pass/high-pass cutoff frequency.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: W = max(cv2.getTrackbarPos('W', 'Webcam Filters'), 1) #* max(..., 1) ensures W is never zero.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: canny_low = cv2.getTrackbarPos('Canny Low', 'Webcam Filters') #* Gets Canny thresholds from UI.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: canny_high = cv2.getTrackbarPos('Canny High', 'Webcam Filters')
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if mode == 'original':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: display = frame.copy() #* No processing; show the raw mirrored frame.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'canny': #* Executes your custom Canny implementation (not OpenCV's).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: blurred = gaussian_blur(gray) #Smooth image to remove noise
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: grad_mag, grad_angle = compute_gradients(blurred) #Compute Sobel gradients.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: nms = non_maximum_suppression(grad_mag, grad_angle) #Thin edges to 1-pixel thickness.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: strong, weak = double_threshold(nms, canny_low, canny_high) #Identify strong and weak edges using thresholds from trackbars
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: edges = edge_tracking(strong, weak) #Track weak edges only if connected to strong ones.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: display = cv2.cvtColor(edges, cv2.COLOR_GRAY2BGR) #Convert edges to BGR to display in color window.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: else: #* Any mode besides 'original' and 'canny' must be a frequency filter.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Computes the 2-channel DFT (real + imaginary).
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: dft = cv2.dft(np.float32(gray), flags=cv2.DFT_COMPLEX_OUTPUT)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: dft_shift = np.fft.fftshift(dft) #Moves low frequencies to center of the spectrum.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if mode == 'ilpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = ideal_low_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'glpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = gaussian_low_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'blpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = butterworth_low_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'ihpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = ideal_high_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'ghpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = gaussian_high_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'bhpf':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = butterworth_high_pass(gray.shape, D0)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'breject':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = band_reject(gray.shape, D0, W)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif mode == 'bpass':
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mask = band_pass(gray.shape, D0, W)
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #Multiplies Fourier spectrum with the selected filter mask.This suppresses or preserves certain frequencies.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: fshift = dft_shift * mask
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: f_ishift = np.fft.ifftshift(fshift) #Undo shift to prepare for inverse DFT.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: img_back = cv2.idft(f_ishift) #Converts filtered spectrum back to spatial domain.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: output = cv2.magnitude(img_back[:,:,0], img_back[:,:,1]) #Computes magnitude of complex result
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: output = cv2.normalize(output, None, 0, 255, cv2.NORM_MINMAX).astype(np.uint8) #Normalize values so they display correctly as an image.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: display = cv2.cvtColor(output, cv2.COLOR_GRAY2BGR) #* Convert grayscale result to BGR for window display.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.putText(display, f"Mode: {mode.upper()}", (10,30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,255,0), 2) #* Writes text on the displayed frame showing the active mode.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.imshow('Webcam Filters', display) #Shows the processed frame
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: key = cv2.waitKey(1) & 0xFF #Reads key input every frame.
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: if key == ord('q'): #quit
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: break #Each key sets a different mode
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('o'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'original'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('c'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'canny'

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('1'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'ilpf'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('2'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'glpf'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('3'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'blpf'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('4'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'ihpf'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('5'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'ghpf'
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('6'):
Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny

method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'bhpf'

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('7'):

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'breject'

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: elif key == ord('8'):

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: mode = 'bpass'

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code:

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cap.release()

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: cv2.destroyAllWindows()

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.

Code: #* Releases webcam and closes all windows.

Explanation: This instruction contributes to the real-time webcam image processing pipeline. It supports frame acquisition, mirroring, grayscale conversion, edge detection using the Canny method, or frequency-domain filtering using DFT/IDFT as outlined in the project workflow.