Name: Harsh Vijay Mamania

CS 5330: Week 5 Homework

Question #1

Give an intuitive definition of the meaning of eigenvectors in the context of image analysis (think about the faces example).

Solution-

- Eigenvectors represent the main ways faces in the dataset differ from each other.
- In the faces example, each eigenvector corresponds to a kind of “face pattern”:
    - One might capture hair style changes.
    - Another might capture face shape variation.
    - Another might reflect expression differences.
- We can think of eigenvectors as the basic building blocks of faces.
- Any face can be approximately reconstructed by combining these building blocks with different weights.
- Intuitively, they define the most important directions of variation in the dataset.

In General Image Analysis

- Eigenvectors capture the dominant patterns of variation across the image dataset
- Each eigenvector represents a fundamental "feature template" - edges, textures, color gradients, or spatial patterns that frequently appear
- Any image can be reconstructed as a weighted combination of these principal patterns

---

Question #2

You have the set of eigenvalues 0.3, 0.25, 0.24, 0.12, 0.06, 0.02, 0.01

A. What is the dimensionality of original data (the number of significant dimensions)?
B. How many dimensions of the data would you need to keep to represent 75% of the variation?
C. How many dimensions of the data would you need to keep to represent 90% of the variation?
D. If you wanted to see best visualization of the data in 2-D, how would you do it?

Solution-

A. 7 dimensions (7 eigenvalues, one per original dimension)

B. Need 3 dimensions to capture at least 75% of variation [0.3 + 0.25 + 0.24 = 0.79 (79%)]

C. Need 4 dimensions to capture at least 90% of variation [0.3 + 0.25 + 0.24 + 0.12 = 0.91 (91%)]

D.

- Project data onto the first two eigenvectors (0.3, 0.25)
- These capture the two directions of maximum variance (0.3 + 0.25 = 55% of total variation)
- Each data point becomes a 2-D coordinate: (projection on eigenvector 1, projection on eigenvector 2)

Question #3

How does aliasing affect building histograms? What is a solution to avoid aliasing when building a histogram?

Solution-

How Aliasing Affects Histograms

- Bin boundary artifacts: Values near bin edges get hard-assigned to one bin or another, creating artificial discontinuities
- Sensitivity to small changes: A pixel value of 127 vs 128 might fall into completely different bins, even though they're nearly identical
- Loss of smoothness: Similar colors/intensities that span bin boundaries don't contribute to neighboring bins, creating jagged histogram distributions
- Example: In a 16-bin color histogram, RGB values (63, 0, 0) and (64, 0, 0) fall into different bins despite being perceptually almost identical

Solution 1: Low-Pass Filtering Before Histogram

- Apply Gaussian blur or other smoothing filter to image first
- Reduces high-frequency noise and sharp transitions
- Creates smoother value distributions that span bin boundaries more naturally
- Example: Blur image slightly before computing color histogram - neighboring pixels influence each other, reducing abrupt intensity changes

Solution 2: Soft Binning (Interpolation)

- Instead of hard-assigning a value to one bin, distribute its contribution across multiple neighboring bins
- Use weighted voting based on distance to bin centers
- Example: A pixel value of 127.5 between bins [120-128] and [128-136] contributes 0.5 to each bin instead of 1.0 to just one
- Common approaches:
    - Linear interpolation between adjacent bins
    - Trilinear interpolation for 3D color histograms (RGB)
    - Gaussian weighting for smoother distributions

Question #4

What are the three types of aliasing we might encounter in computer vision? Give an example of each one.

Solution-

1. Spatial Aliasing
    - Occurs when image resolution is too low to capture fine spatial details
    - High-frequency spatial patterns appear as lower-frequency artifacts
    - Example: Photographing a brick wall or chain-link fence - regular patterns create moiré effects or jagged edges because pixel spacing is too coarse to properly sample the fine details

2. Spectral (Color) Aliasing
    - Happens when color/wavelength sampling is insufficient
    - Incorrect color reconstruction or false colors appear
    - Example: Demosaicing artifacts in Bayer-pattern cameras - a single red pixel trying to represent a region creates color fringing at edges. Or capturing a rainbow with too few color channels, losing subtle hue transitions

3. Temporal Aliasing
    - Results from frame rate being too low to capture motion
    - Fast movements appear jerky, slowed, or reversed
    - Example: Wagon wheel effect in video - spinning wheels appear to rotate backward or freeze because the rotation frequency exceeds half the frame rate (violates temporal Nyquist limit)

Question #5

Consider a training set of images of doors. Each image is 128x128 = 2^14 pixels. You decide to build an image eigenspace of reduced dimension using your data set. You find that 32 eigenvectors represents over 90% of the variation in the data set and decide to use those eigenvectors as your embedding space.

A. What is the process for projecting a new image of a door into the door eigenspace?
B. How many dimensions will the embedding of the new image have? What is the compression factor?
C. How big is each eigenvector?
D. What is the process for re-creating the original image from the embedding?
E. If you project a picture of a frog into the door eigenspace and then try recreating the original image, what will the re-creation look like?

Solution-

1. Process for Projecting New Door Image
   - Flatten the 128×128 image into a 16,384-dimensional vector
   - Subtract the mean image (computed from training set) to center the data
   - Compute dot product with each of the 32 eigenvectors
   - Result is a 32-dimensional vector of coefficients (the embedding)

2. Embedding Dimensions & Compression Factor
   - Embedding has 32 dimensions
   - Compression factor: 16,384 / 32 = 512× compression

3. Size of Each Eigenvector
   - Each eigenvector is 16,384 dimensions (same as flattened image size: 128×128)

4. Recreating Original Image
   - Multiply each of the 32 eigenvectors by its corresponding coefficient from the embedding
   - Sum all 32 weighted eigenvectors together
   - Add back the mean image
   - Reshape from 16,384-D vector back to 128×128 image

5. Projecting a Frog Image
   - The reconstruction will look door-like or ghostly/blurry
   - Since eigenvectors only capture door variations, the frog will be forced into "door space"
   - You'll lose frog-specific features and see a vague door-shaped blob
   - It's like trying to describe a frog using only vocabulary designed for doors

---