Name: Harsh Vijay Mamania

CS 5330: Week 2 Homework

Question #1

Give a couple of subjective explanations of what a filter can do.

Solution-

Couple of things that a filter can do:

Feature Detection

- A filter acts as a feature detector that searches for specific patterns within an image.
- It convolves across the image, computing similarity measures between the filter template and local image regions.
- Strong pattern matches produce high response values, while weak matches yield low responses.
- This enables detection of visual characteristics such as edges, corners, and textures.

Image Transformation

- A filter functions as a local image transformer that modifies pixel values based on their spatial neighborhood.
- It applies weighted combinations of neighboring pixel values to compute new pixel intensities.
- Depending on the filter coefficients, this operation can blur details, sharpen features, reduce noise, or emphasize specific image aspects.
- The filter weights determine the nature and magnitude of the transformation applied to the image.

Question #2

Give an example of a separable filter and explain the primary advantages of using them.

Solution-

Gaussian separable filter:

- A 2D Gaussian filter can be expressed as the convolution of two 1D filters:
  - A vertical filter $g = [1\ 2\ 1]^T$
  - A horizontal filter $h = [1\ 2\ 1]$

- Performing the convolution $g \otimes h$ produces the 3 × 3 kernel:
  - $\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
  - This kernel is a discrete approximation of a Gaussian.

- Because convolution is associative:
  - $f \otimes (g \otimes h) = (f \otimes g) \otimes h$
  - The filtering can be applied in two stages instead of one full 2D convolution.

- In practice:
  - First convolve the image f with the 1D vertical filter g
  - Then convolve the intermediate result with the 1D horizontal filter h
  - The final output is identical to convolving with the full 2D Gaussian kernel.

Reduction in number of operations per pixel

- Direct 2D convolution using a 3 × 3 kernel:
  - 9 multiplications
  - 8 additions
  - 1 normalization (division)
  - Total: 18 basic operations per pixel

- Separable implementation using two 1D convolutions:
  - First pass (3 × 1 filter):
    - 3 multiplications + 2 additions + 1 division = 6 operations
  - Second pass (1 × 3 filter):
    - 3 multiplications + 2 additions + 1 division = 6 operations
  - Total: 12 basic operations per pixel

- This shows that separability reduces computation while producing the same result.

Scaling to larger filters (as discussed in class)

- A 16 × 16 2D filter requires:
  - 256 multiplications

    - 255 additions
    - 1 division
    - Approximately 512 operations per pixel

- A separable version requires:
    - One 16 × 1 pass and one 1 × 16 pass
    - Effective filter size reduced to 32
    - Approximately 64 operations per pixel

- The computational savings become increasingly significant for larger kernels and high-resolution images.

Advantages of using separable filters

- *Improved computational efficiency*:
  Separable filters reduce the computational cost of convolution from $O(k^2)$ operations per pixel for a k × k kernel to O(2k) operations by applying two 1D convolutions.
- *Faster execution*:
  The reduced number of arithmetic operations leads to significantly faster filtering, especially for large kernel sizes and high-resolution images.
- *Reduced memory requirements*:
  Only the 1D filter kernels need to be stored instead of the full 2D kernel, lowering memory usage.
- *No loss of accuracy*:
  When a filter is truly separable, applying successive 1D convolutions produces the same result as applying the original 2D filter.

Question #3

What does it mean if a filter is an edge-preserving filter? Give an example of one.

Solution-

- An edge-preserving filter is one that smooths/reduces noise in an image while maintaining sharp boundaries between different regions.
- Traditional filters (like Gaussian blur) smooth everything uniformly - edges get blurred along with noise
- Edge-preserving filters are "adaptive" - they smooth within regions but avoid smoothing across edges
- They detect where significant intensity changes occur (edges) and preserve those boundaries
- These filters consider both - spatial distance AND intensity similarity

“Median” and “Bilateral” Filters:

- A **median filter** is a common edge-preserving filter used when smoothing an image while retaining large edge information.
    - Given an N × N neighborhood, the central pixel is replaced by the median value of the pixels within the mask.
    - Unlike Gaussian filtering, the median filter does not average across strong intensity changes, allowing edges to remain sharp.
    - However, the median filter cannot be expressed as a convolution or frequency-domain operation. It is an algorithmic process that requires sorting, typically an N log N operation at each pixel, and involves data-dependent conditionals, making it computationally expensive on modern processors.

- The **bilateral filter** is the most widely used edge-preserving filter and has many applications.
    - It smooths images by averaging pixels that are both spatially close and similar in intensity or color.
    - This is achieved by combining a spatial Gaussian with a second Gaussian that depends on the intensity difference between the central pixel and its neighbors.
    - As a result, pixels that differ significantly in intensity receive very low weights, even if they are spatially close, which prevents smoothing across edges.
    - While the bilateral filter is computationally expensive in its basic form, several fast approximations exist that produce high-quality results.

Question #4

What is an example of a high-pass filter, and why would we want to use one?

Solution-

- A high-pass filter preserves high-frequency information (rapid changes/details) while attenuating or removing low-frequency information (slow variations/smooth regions)
- High frequencies correspond to edges, textures, and fine details
- Low frequencies correspond to gradual intensity changes and smooth areas

Example: Sobel Filter

A **Sobel filter** is a common high-pass filter used for edge detection. It computes the gradient of image intensity at each pixel, emphasizing regions of high spatial frequency (edges).

The Sobel operator uses two 3×3 kernels to detect edges in horizontal and vertical directions:

Horizontal (Gx):
[-1 0 +1]
[-2 0 +2]
[-1 0 +1]

Vertical (Gy):
[-1 -2 -1]
[0 0 0]
[+1 +2 +1]

The gradient magnitude is computed as: $G = \sqrt{(Gx^2 + Gy^2)}$

Why use a high-pass filter?

Edge detection:
- Highlights boundaries and transitions between objects.
- Essential for feature detection and object recognition.

Image sharpening:
- Enhances fine details and textures.
- Makes images appear crisper (by emphasizing edges).

Noise visualization:
- Can reveal high-frequency noise in images.
- Useful for quality assessment.

Preprocessing for vision tasks:
- Many computer vision algorithms rely on edge information.
- High-pass filtering extracts the relevant features.

Unsharp masking (image enhancement):

- Add the high-pass filtered result back to the original.
- Result = Original + α × (Original - Blurred).
- Selectively enhances edges while maintaining overall appearance.

Question #5

What are a couple of ways of handling boundary issues (the edges of the image) when applying a filter?

Solution-

1. Zero Padding (Pad with zeros)

- Assume all pixels outside the image boundary have a value of zero (black)
- Simple to implement but can create artificial dark edges in the filtered result

2. Constant Padding

- Pad with a constant value (could be zero, or another chosen value)
- Variation of zero padding with more control

3. Replicate/Clamp (Extend edge pixels)

- Replicate the nearest edge pixel outward to fill the padding region
- The boundary pixels are repeated beyond the edge
- Avoids introducing artificial values and maintains better continuity

4. Reflect/Mirror Padding

- Mirror the image pixels across the boundary
- Pixels are reflected to create padding values
- Maintains continuity better than simple replication

5. Wrap/Periodic Padding

- Treat the image as if it tiles or wraps around
- Left edge connects to right edge, top edge to bottom edge
- Particularly useful for certain frequency domain operations

6. Ignore/Crop

- Don't compute filter output for pixels near boundaries where the mask extends beyond the image
- Results in a smaller output image but avoids boundary artifacts
- Valid approach when boundary information can be sacrificed

Question #6

You take the Fourier transform of an image and organize it so the DC and low frequencies are in the center of the visualization, and the high frequencies are around the outside of the image. If you were to set the values in a small circle at the center of the Fourier transform to zero and then convert the image back to the regular spatial domain, what would be the effect on the image?

Solution-

Setting the values in a small circle at the center of the Fourier transform to zero removes the DC component and low-frequency information from the image.

This operation effectively applies a **high-pass filter** to the image.

Effects on the spatial domain image:

- **Smooth regions are suppressed:** Areas with gradual intensity variations and smooth transitions are removed or flattened, as these correspond to low frequencies
- **Edges and fine details are preserved:** High-frequency content remains intact, causing sharp transitions and textures to be emphasized
- **Loss of overall brightness structure:** Removing the DC component (the very center pixel representing average intensity) shifts the mean intensity of the image, affecting overall brightness and contrast
- **Edge-map appearance:** The resulting image resembles an edge detection result, showing primarily boundaries, textures, and rapid intensity changes while suppressing uniform regions

---

Question #7

You have an image that consists of vertical and horizontal lines. After taking the Fourier transform of the image you set a centered horizontal bar of the Fourier image to zero, all except the central DC signal. After taking the inverse Fourier Transform, how do you think the resulting image would look?

Solution-

The Setup:

- Original image: vertical and horizontal lines
- Fourier transform: DC component at center, frequencies spread around
- Operation: Zero out a centered horizontal bar in frequency domain (except DC)
- Then inverse transform back to spatial domain

Frequency domain orientation:

- Vertical structures in spatial domain → horizontal frequencies in Fourier domain
- Horizontal structures in spatial domain → vertical frequencies in Fourier domain
- This is a 90° relationship

What we're removing:

- A horizontal bar in the Fourier domain (keeping only DC)
- Since horizontal patterns in frequency domain correspond to vertical patterns in spatial domain...
- We're removing the frequencies that represent vertical lines in the original image

Effect on the spatial domain image:

- Vertical lines would be removed/suppressed - their corresponding frequencies (horizontal bar in Fourier domain) have been zeroed
- Horizontal lines would remain - their frequencies (vertical in Fourier domain) are untouched
- DC component preserved - overall average intensity is maintained

Result: **The image would show only the horizontal lines, with the vertical lines largely removed or significantly attenuated.**

(“directional filtering” // This is essentially an “oriented band-stop filter”)
