Linear Filters: Part 1
Tuesday, Sept 9

Announcements

• Please put your name in problem set files.

• Updated office hours
  – Me: CSA 114
    • Wed 1:15 pm – 2:15 pm
    • Thurs 2 pm – 3 pm
  – Harshdeep: TAY CS Lab
    • Mon 1 pm – 2 pm
    • Fri 2 pm – 3 pm

Image neighborhoods

• Q: What happens if we reshuffle all pixels within the images?

  • A: Its histogram won’t change.
    Point-wise processing unaffected.

  • Need to measure properties relative to small neighborhoods of pixels

Images as functions

• We can think of an image as a function, f, from \( \mathbb{R}^2 \) to \( \mathbb{R} \):
  • \( f(x, y) \) gives the intensity at position \((x, y)\)
  • Realistically, we expect the image only to be defined over a rectangle, with a finite range:
    
    \[
    - f : [a, b] \times [c, d] \rightarrow [0, 1.0]
    \]

• A color image is just three functions pasted together. We can write this as a “vector-valued” function:

  \[
  f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}
  \]

Digital images

• In computer vision we operate on digital (discrete) images:
  • Sample the 2D space on a regular grid
  • Quantize each sample (round to nearest integer)
  • Image thus represented as a matrix of integer values.
Motivation: noise reduction

- In Pset 0 we measured noise in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?

Common types of noise

- **Salt and pepper noise**: random occurrences of black and white pixels
- **Impulse noise**: random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Original  | Salt and pepper noise
Impulse noise | Gaussian noise

Source: S. Seitz

Gaussian noise

```
>> noise = randn(size(im)).*sigma;
>> output = im + noise;
```

Effect of sigma on Gaussian noise:
Image shows the noise values themselves.
Effect of sigma on Gaussian noise:
This shows the noise values added to the raw intensities of an image.

Effect of sigma on Gaussian noise
This shows the noise values added to the raw intensities of an image.

Motivation: noise reduction
- In Pset 0 we measured noise in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?
- What if there’s only one image?

First attempt at a solution
- Let’s replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

First attempt at a solution
- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

Weighted Moving Average
Can add weights to our moving average
Weights \([1, 1, 1, 1] / 5\)
Weighted Moving Average
Non-uniform weights [1, 4, 6, 4, 1] / 16

Source: S. Marschner

Moving Average In 2D

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz

Moving Average In 2D

Source: S. Seitz
Moving Average In 2D

\[ F[x, y] \quad G[x, y] \]

Correlation filtering

Say the averaging window size is \(2k+1 \times 2k+1\):

\[
G[i, j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]
\]

Loop over all pixels in neighborhood around image pixel \(F[i, j]\)

Attribute uniform weight to each pixel

Now generalize to allow different weights depending on neighboring pixel's relative position:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

Non-uniform weights

Correlation filtering

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

This is called cross-correlation, denoted \(G = H \otimes F\)

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" \(H[u, v]\) is the prescription for the weights in the linear combination.

Averaging filter

\[
F[x, y] \quad \otimes \quad H[u, v] \quad G[x, y]
\]

Original filtered

"box filter"

\[
G = H \otimes F
\]

Gaussian filter

\[
H[u, v] = \frac{1}{2\pi \sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}}
\]

This kernel is an approximation of a Gaussian function:

\[
h(u, v) = \frac{1}{2\pi \sigma^2} e^{-\frac{u^2 + v^2}{2\sigma^2}}
\]

Smoothing by averaging

Smoothing by averaging depicts box filter:

white = high value, black = low value

Source: S. Seitz
Smoothing with a Gaussian

Gaussian filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note: Gaussian function has infinite support, but discrete filters use finite kernels

![Gaussian filter with size](image)

- Variance of Gaussian: determines extent of smoothing

![Gaussian filter with variance](image)

Matlab

```matlab
>> hsize = 10;
>> sigma = 5;
>> h = fspecial('gaussian', hsize, sigma);
>> mesh(h);
>> imagesc(h);
>> outim = imfilter(im, h);
>> imshow(outim);
```

Boundary issues

What is the size of the output?
- MATLAB: filter2(g, f, shape)
  - `shape = 'full'`: output size is sum of sizes of f and g
  - `shape = 'same'`: output size is same as f
  - `shape = 'valid'`: output size is difference of sizes of f and g

![Boundary issues](image)
Boundary issues

What about near the edge?
• the filter window falls off the edge of the image
• need to extrapolate
• methods:
  – clip filter (black)
  – wrap around
  – copy edge
  – reflect across edge

Source: S. Marschner

Filtering an impulse signal

What is the result of filtering the impulse signal (image) $F$ with the arbitrary kernel $H$?

$F[x, y] \otimes H[u, v] \rightarrow G[x, y]$