Introduction to Computer Vision

Alvin Lin

August 2018 - December 2018

Linear Filters

We can think of an image as a function f from $\mathbb{R}^2$ to $\mathbb{R}$ where $f(x, y)$ gives the intensity at position (x, y). Realistically, we expect the image only to be defined over a rectange with a finite range. A color image is just three functions pasted together, which can be represented as:

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

An image contains a discrete number of pixels, which contain data according to the color space being used. In computer vision, we operate on digital (discrete) images. By sampling the 2D space on a regular grid and quantizing each sample by rounding to the nearest integer, we can represent an image as a matrix of integer values.

Filtering

Filtering is the process of forming a new image whose pixels are a combination of the original pixel values. The goal is to extract useful information from the images (features such as edges, corners, or blobs) or modify/enhance image properties (super-resolution, de-noising, etc).

Noise Reduction

We can measure noise in multiple images of the same static scene. Types of noise include:

- **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

For our first attempt at a solution, we will replace each pixel with an average of all values in its neighborhood. We assume pixels to be like their neighbors and noise processes to be independent from pixel to pixel. We can use a moving average to compute the new pixel values, and add weights to the moving average to scale the amount of change. In 2D, we average all the pixels in a 2D neighborhood around the target pixel. Say the averaging window size if $2k+1 \times 2k_1$, then the new pixel value is:

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

The $\frac{1}{(2k+1)^2}$ scaling factor attributes a uniform weight to each pixel and the summations accumulate the summed intensity of all pixels in the $2k+1 \times 2k+1$ window. We can generalize this to allow different weights depending on the neighboring pixel's relative position:

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

where $H[u,v]$ is the weighting function. This is known as cross-correlation and denoted $G = H \otimes F$. Essentially, filtering an image is replacing a 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.

Gaussian Filters

An example Gaussian filter:

$$H[u,v] = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Using this as a filter mask prioritizes the nearest pixels and gives them the most influence over the output. Gaussian filters are most affected by the size of the kernel or mask and the variance of the Gaussian, which determines the extent of smoothing.

Boundary Issues

The filter window falls off the edge of the image and we need to extrapolate. We can either clip the filter, wrap around the images, copy the original image edge, or reflect across the edge.

Filtering an Impulse Signal (Correlation)

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

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \otimes \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & i & h & g & 0 & 0 \\ 0 & 0 & f & e & d & 0 & 0 \\ 0 & 0 & c & b & a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Convolution

If we instead flip the filter in both dimensions and apply cross-correlation by using this formula instead:

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

then the filter orientation is preserved when applied on an impulse signal.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \star \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & a & b & c & 0 & 0 \\ 0 & 0 & d & e & f & 0 & 0 \\ 0 & 0 & g & h & i & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

This operation is denoted using the convolution operator ($\star$).

Correlation vs Convolution

Cross-correlation:

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

Convolution:

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

Note that if the filter is symmetric, then there is no difference between correlation and convolution. Using a filter like:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

will shift the image to the right if using convolution, and to the left if using correlation.

Shift Invariant Linear System

A shift invariant operator behaves the same everywhere. The value of the output depends on the pattern in the image neighborhood and not the position of the neighborhood.

- Superposition: $h \star (f_1 + f_2) = h \star f_1 + h \star f_2$
- Scaling: $h \star (kf) = k \star (hf)$

Convolution is linear and shift invariant, commutative, associative, and has the property of identity when applied on a unit impulse e, $(f \star e = f)$. The property of differentiation also holds:

$$\frac{\mathrm{d}}{\mathrm{d}x}(f \star g) = \frac{\mathrm{d}f}{\mathrm{d}x} \star g$$

Median Filter

A median filter replaces a pixel with the median of its neighboring values. Unlike a mean filter, it is edge preserving and does not add additional blur. No new pixel values are introduced because only existing values are used to replace outlier values.

Template Matching

Filters look like the effects they are intended to find. We can use filters and normalized cross-correlation to find image features. Correlation mask:

$$C_{fg} = \sum_{[i,j] \in \mathbb{R}} f(i,j) g(i,j)$$

The brightness of the underlying image can skew the correlation score, so we normalize by subtracting the mean value of the template. This works better, but still not well. The most popular matching score is using the sum of squared differences.

$$C_{fg} = \sum_{[i,j] \in \mathbb{R}} (f(i,j) - g(i,j))^2$$

When an image is take by different sensors, there may be intensity differences, which we can resolve by normalizing the images before doing correlation.

$$f_{norm} = \frac{f - \bar{f}}{\sqrt{\sum (f - \bar{f})^2}}$$
$$g_{norm} = \frac{g - \bar{g}}{\sqrt{\sum (g - \bar{g})^2}}$$

You can find all my notes at `http://omgimanerd.tech/notes`. If you have any questions, comments, or concerns, please contact me at alvin@omgimanerd.tech