Dao Anh Khoa 1/18/2026

Code

Convolution Correlation

Correlation operation is done mainly by:

```
res[y, x] = img[y: y + h_mask, x: x + w_mask].flatten().dot(flatten_mask)
```

Convolution is correlation with rotated kernel by 180°

```
r_mask = mask[::-1, ::-1]
return Convolution_Correlation.correlation(img, r_mask)
```

Gaussian

Each entry (x, y) in the matrix is calculated from the gauss function in 2D. Then matrix is normalized to have summation of 1 and thus the intergral constant normalization unit can be ignored:

```
res[y, x] = _gauss_function(x, y, offset, offset, sigma)
...
return res/res.sum()
```

Where `_gauss_function` is the gaussian function in 2D:

$$G(x, y) = \exp\left(-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}\right)$$

Compare convolution with correlation by Gaussian kernel

The image I is created 1 in the middle and 0 everywhere else (delta image), size 100 × 100

Kernel K is 2D Gaussian, size 21×21, $\sigma = 5$

$$\begin{aligned}\text{Mean}|I * K - I \star K| = 0.0,\\ \text{Mean}|K * I - K \star I| = 0.0004947591.\end{aligned}$$

where, $*$ denotes convolution, and $\star$ denotes correlation.

Explaination:

The 0 absolute differences shows that convolution and correlation give the same result for the image. This happens when the kernel is symmetric, so flipping the kernel does not change it. While 0.0004947591 migth be error from the even shape of the image does not have the true middle input the uneven padding becuse of the odd shape of kernel. Nevertheless, it is safe to ignore in practice.

Gaussian Pyramid

Given an input image, first apply convolution with kernel filter, then scale down the image by 2:

```
res = Convolution_Correlation.convolution(img, kernel)
return res[::2, ::2]
```

The pyramid is recursively applying the above for each scaled down of the image.

Fourier Transform

The kernel is first padded to have the same shape like the image, then apply transformation `np.fft.fft2` to both the image and padded kernel, then multiply them.

High pass - Band pass Gaussian

A Gaussian kernel acts as a low-pass filter, preserving low-frequency components while suppressing high-frequency details. To construct a high-pass Gaussian, we subtract the Gaussian low-pass response from an impulse (Dirac delta) centered at the kernel origin. This allows low-frequency content to be attenuated and high-frequency components (edges, fine details) to be emphasized.

```
gauss = GaussianMask.mask_gauss(sigma, size)
delta = np.zeros_like(gauss)
delta[center, center] = 1.0
high_pass = delta - gauss
```

A band-pass Gaussian filter preserves frequencies within a specific range while suppressing both low and high extremes. It can be constructed by taking the difference of two Gaussian low-pass filters with different standard deviations.

```
g_low = GaussianMask.mask_gauss(sigma_low, size)
g_high = GaussianMask.mask_gauss(sigma_high, size)
band_pass = g_low - g_high
```

Steerable Gaussian

First, two partial derivative G_x and G_y is constructed by

$$\frac{\partial G}{\partial x} = -\frac{x - c_x}{\sigma^2} G(x, y) \qquad \frac{\partial G}{\partial y} = -\frac{y - c_y}{\sigma^2} G(x, y)$$

```
gauss_x[y, x] = -(x - cx) / sigma**2 * gauss[y, x]
gauss_y[y, x] = -(y - cy) / sigma**2 * gauss[y, x]
```

First, the oriented kernel is synthesized and then correlated with the image

```
G_theta = np.cos(theta) * Gx + np.sin(theta) * Gy
response = Convolution_Correlation.correlation(image, G_theta)
```

Second, the image is correlated separately with the basis filters G_x and G_y, and the responses are then linearly combined:

```
Rx = Convolution_Correlation.correlation(image, Gx)
Ry = Convolution_Correlation.correlation(image, Gy)
steered_response = np.cos(theta) * Rx + np.sin(theta) * Ry
```

Mean of absolute differences between two result is `-6.203254299891965 4e-09`

Theory

Question:
What is the minimum memory capacity required to store a Gaussian pyramid?

Answer:
Let the input image have width W and height H, with total area

$$A = W \times H$$

A Gaussian pyramid is constructed by repeatedly downsampling the image by a factor of 2 in both width and height at each level.

Thus, the area at level i is: $A_(t + 1) = \frac{A_t}{4}$

The total memory required to store the entire pyramid is:

$$\sum_{i=0}^{\infty} \frac{A}{4^i} = A \cdot \frac{1}{1 - \frac{1}{4}} = \frac{4}{3} A$$

Conclusion:
The smallest memory capacity required to store a Gaussian pyramid is: $\frac{4}{3} WH$

Steerable Filters

Question: How to prove the steerability of a filter?

Answer

A steerable filter is an orientation-selective filter that can be computationally rotated to any direction. Rather than designing a new filter for each orientation, a steerable filter is synthesized from a linear combination of a small, fixed set of "basis filters" that is independent of angle θ:

$$f(x,y,\theta) = \sum_{j=1}^{M} k_j(\theta) f_j(x,y).$$

Question Prove that derivative of Gaussian is a steerable filter

Answer:

Rotation of a function $f(x,y)$ is performed by multiplying the coordinate system with a rotation matrix paramaterized by angle θ.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

Without loss of generalization, assume that $x_0 = y_0 = 0$ and $\sigma_x = \sigma_y = 1$, and ingore the constants. Then the gaussian function is:

$$G(x,y) = e^{-x^2-y^2}$$

It is straightforward to see that Gaussian function is isotropic: $G(x,y) = G(x',y')$. Next, the partial derivative of Gaussian to each axis:

$$\begin{aligned} \frac{\partial G}{\partial x'} &= G_{x'} = -2x'e^{-(x'^2+y'^2)} = -2(x\cos\theta + y\sin\theta)\, e^{-(x^2+y^2)} \\ \frac{\partial G}{\partial y'} &= G_{y'} = -2y'e^{-(x'^2+y'^2)} = -2(-x\sin\theta + y\cos\theta)\, e^{-(x^2+y^2)} \end{aligned}$$

It is now straightforward to see that:

$$\nabla_{x'y'} G = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \nabla_{x,y} G$$

Furthermore, if we only care about the response of a rotated gaussian filter along the x-axis:

$$\frac{\partial G}{\partial x'} = [\cos\theta \quad \sin\theta] \nabla_{x,y} G = \cos\theta G_x + \sin\theta G_y.$$

Question: How many basis?

Answer

It is easy to see that $\partial_y G$ is $\partial_x G$ rotated by $90°$, which are orthogonal and thus constitue two basis:

$$G_x^{90^\circ} = -2(x\cos 90^\circ + y\sin 90^\circ)e^{-x^2-y^2} = -2ye^{-x^2-y^2}$$

$$G_x^{\theta^\circ} = \cos\theta G_x^{0^\circ} + \sin\theta G_x^{90^\circ}$$

Question Prove second derivative of gaussian is also steerable and has no less than 3 basis.

Answer

$$\begin{aligned} G_{x'x'} &= \frac{\partial^2 G}{\partial x'^2} \\ &= \left(\cos\theta \frac{\partial}{\partial x} + \sin\theta \frac{\partial}{\partial y}\right)^2 G \\ &= \cos^2\theta G_{xx} + 2\sin\theta\cos\theta G_{xy} + \sin^2\theta G_{yy}. \end{aligned}$$

With G independent of angle θ. Because the second-order derivative produces polynomials of degree 2 in $\cos\theta$ and $\sin\theta$, there shall be no less than 3 basis.

Question: What about third derivative of gaussian

Answer

The third-order directional derivative can be written as a linear combination of basis filters with coefficients that depend linearly on the angular monomials. There should be at least 4 basis, since the angular dependence consists of monomials of degree 3 has number of linearly indepdendent is 4:

$$G_{x'x'x'} = \cos^3\theta G_{xxx} + 3\cos^2\theta \sin\theta G_{xxy} + 3\cos\theta \sin^2\theta G_{xyy} + \sin^3\theta G_{yyy}.$$