Questions

1. This question and the next one are more about image filtering than they are about edge detection.

   A 2D filter $f(x, y)$ is said to be *separable* if it can be written as a product of two 1D filters. Show that the following filters are separable. For the first three, note you will need to multiply a 3×1 vector with a 1×3 vector.

   (a) The Prewitt filter (for x direction)

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

   (b) The Prewitt filter (for y direction)

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

   (c) the Sobel filter (for x direction)

   $$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

   (d) the 2D Gaussian (for continuous case) $G(x, y)$.
   Try to remember the definition, rather than looking it up. Assume the mean is $(0, 0)$.

2. (a) Write out pseudocode for the convolution of a 2D image `I` and a 2D filter `f`, using nested `for` loops. For simplicity, assume that the arrays are indexed from 0 to `size - 1` in each dimension.

   (b) Separability allows you to speed up the convolution by treating it as two 1D convolutions. Explain how.

3. This exercise deals with *steerable filters*. Specifically we look at how to take the derivatives of a smoothed image in an arbitrary direction. We set up the problem by supposing that you smooth an image $I(x, y)$ with a Gaussian and then you compute the derivative in the x and y directions. This is equivalent to convolving the image with the two filters $\frac{\partial}{\partial x}G(x, y)$ and $\frac{\partial}{\partial y}G(x, y)$.

   (a) Let $G^1_\theta(x, y)$ the (directional) first derivative of the smoothed image *in some arbitrary direction* $(\cos\theta, \sin\theta)$. Show using basic Calculus that this directional derivative is:

   $$G^1_\theta(x, y) \;=\; \cos\theta \, \frac{\partial}{\partial x}G(x, y) + \sin\theta \, \frac{\partial}{\partial y}G(x, y)$$

   So, the directional derivative is a weighted sum of the derivatives in the x and y directions.

(b) Suppose that we would like to take the *second* derivative of the smoothed image in direction $(\cos\theta, \sin\theta)$. In particular, how could we do so by combining derivatives in the x and y directions ?

4. Canny's edge detector requires setting thresholds. What mistakes can occur if the threshold τ_{high} is set to too small a value ? What mistakes can occur if the threshold value is too high? What mistakes can occur if too much blur is used (σ too big) or not enough blur is used (σ too small) ?

5. The non-maximum suppression step is used the thin edges. As described in the slides, only a 3×3 neighborhood of each above-threshold point was considered. Does this imply that the method cannot thin edges whose thickness is greater than 3 pixels ?

Answers

1. (a) The Prewitt filter (for x direction)

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

   (b) The Prewitt filter (for y direction)

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

   (c) the Sobel filter (for x direction)

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

   (d)

$$G(x, y; \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} = (\frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}) \; (\frac{1}{\sqrt{2\pi}\sigma} e^{-y^2/2\sigma^2})$$

2. (a) Suppose we have an image `I` of size of size $N \times N$ and filter `f` of size $M \times M$. We want to compute the following. Note that we want to compute this for each (x, y).

$$I(x, y) * f(x, y) \equiv \sum_{u,v} I(u, v) \; f(x - u, y - v)$$

   For the pseudocode below, when the indices of `I(x-u, y-v)` are not valid, we are essentially assuming the value is 0.

```
Initialize a 2D matrix A() of size N+M-1 x N+M-1 to be 0.
for each column index x in the output array A  (0 to N+M-2)
   for each row index y in the output array A  (0 to N+M-2)
      for each column index v in the image I  (0 to N-1)
         for each row index u in the image I  (0 to N-1)
             if there is no index out of bounds error in I below
                 A(x,y) += I(u,v) f(x-u, y-v)
```

   (b) The pseudocode above would take N^2M^2 multiplications and additions.
   But if we used the separability property, then for each `(x,y)` in the accumulator array, we can compute the value of `A(x,y)` as two 1D convolutions:

$$I(x, y) * f(x, y) = I(x, y) * f_1(x) * f_2(y)$$

where $f_1(x)$ is a row vector and $f_2(y)$ is a column vector. So we could first compute a 1D convolution within each row y:

$$I(x,y) * f_1(x) = \sum_u I(u,y) f_1(x-u)$$

which would involve N^2M operations and yield a matrix of size N rows and $N+M-1$ columns. Then we would compute

$$(I(x,y) * f_1(x)) * f_2(y) = \sum_v f_2(y-v) I(x,v)$$

where the convolutions would be done within each column, and this would be $N(N+M-1)M$ additions and multiplications.

So in total, we would need $N^2M + N(N+M-1)M$ which is $O(N^2M)$ when $N > M$.

If $M = 2$, then we don't save any time. But if M is much larger than 2, then it really is faster.

3. (a) The definition of a directional derivative (from Calculus) is:

$$G^1_\theta(x,y) \equiv lim_{s \to 0} \frac{1}{s}(G(x + s\cos\theta, y + s\sin\theta) - G(x,y))$$

Taking the limit gives the answer. (Use the chain rule with partial derivatives.)

(b) Let $G^2_\theta(x,y)$ denote the second derivative in the direction θ. This is the first derivative of the first derivative in that direction, so applying the definition of a derivative from Calculus:

$$G^2_\theta(x,y) \equiv lim_{s \to 0} \frac{1}{s}(G^1_\theta(x + s\cos\theta, y + s\sin\theta) - G^1_\theta(x,y))$$

Substituting from part (a) i.e.

$$G^1_\theta(x,y) \;=\; \cos\theta \, \frac{\partial}{\partial x} G(x,y) + \sin\theta \, \frac{\partial}{\partial y} G(x,y)$$

and taking the limit (again using the chain rule with partial derivatives) gives:

$$G^2_\theta(x,y) = \cos^2\theta \, \frac{\partial^2}{\partial x^2} \, G(x,y) + \sin^2\theta \, \frac{\partial^2}{\partial y^2} G(x,y) \;+\; 2\cos\theta\sin\theta \frac{\partial^2}{\partial x \partial y} G(x,y)$$

Notice that the second derivative in direction θ depends on the second derivatives in x and in y as well as on the cross partial derivatives. Another way to write the above is:

$$G^2_\theta(x,y) = \begin{bmatrix} \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} \frac{\partial^2}{\partial x^2} G(x,y) & \frac{\partial^2}{\partial x \partial y} G(x,y) \\ \frac{\partial^2}{\partial x \partial y} G(x,y) & \frac{\partial^2}{\partial y^2} G(x,y) \end{bmatrix} \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}$$

where the 2×2 matrix is called the *Hessian* of $G(x,y)$.

4. The Canny edge detector identifies points as being edges or not edges. If the threshold is too low, then it will accept pixels (and their gradient orientations) incorrectly as edges. This is called a *false positive*. If the threshold is too high, then it will reject pixels that actually are edges. This is called a *false negative*. If σ is too low, then the noise will not be blurred enough, relative to the signal (edges). In this case, we may have false positives and the true edges that are detected might have position and orientation errors. If the σ is too high, then the signal will be blurred too much. This can cause different edges to blur together, which can distort the position and orientation of the edges.

5. Using a 3×3 neighborhood is not necessarily a problem: edges that are four or more pixels thick can still be thinned. For example, if the magnitude of the gradient is above threshold over $n > 4$ pixels along some direction, but it has a local maximum (peak) at only one of those n pixels, then the method will still remove (non-maxima suppression) all candidate pixels except the local max one, since by definition those pixels are not local maxima.

