CS 83: Computer Vision	Winter 2024

Quiz 2 — 01/24/2024

Prof. Pediredla — ***Student: Amittai Siavava***

Credit Statement

I discussed solution ideas with:

1. Ivy (Aiwei) Zhang (Q1, Q2)
2. Angelic McPherson (Q1, Q2)

However, all the typed work is my own, with reference to class notes especially on convolutions and correlations.

Problem 1.

In class, we discussed how, given a windowing function $w(s,t)$, we can use the following *covariance* metric:

$$E_w(u,v;x,y) = \sum_{s,t} w(s,t)\left[I(x-s+u, y-t+v) - I(x-s, y-t)\right]^2, \tag{1.1}$$

in order to identify whether an image patch centered at (x,y) looks like a corner. In particular, large values of E_w for all possible displacements (u,v) of the window indicate that the patch is a corner.

Assuming that the displacements u and v are small, show that the metric of Equation (1.1) can be approximated as:

$$E_w(u,v;x,y) \approx [u,v] \cdot \mathcal{M}_w(x,y) \cdot [u,v]^T, \tag{1.2}$$

where $\mathcal{M}_w(x,y)$ is the *covariance matrix*:

$$\mathcal{M}_w(x,y) = \begin{bmatrix} \sum_{s,t} w(s,t) I_x(x-s,y-t) I_x(x-s,y-t) & \sum_{s,t} w(s,t) I_x(x-s,y-t) I_y(x-s,y-t) \\ \sum_{s,t} w(s,t) I_y(x-s,y-t) I_x(x-s,y-t) & \sum_{s,t} w(s,t) I_y(x-s,y-t) I_y(x-s,y-t) \end{bmatrix}. \tag{1.3}$$

Since the displacements of u and v are small, consider the first-order Taylor approximation of $I(x-s+u, y-t+v)$ around the point $(x-s, y-t)$. Following the expansion

$$f(x,y) \approx f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b),$$

we have:

$$I(x-s+u, y-t+v) \approx I(x-s, y-t) + I_x(x-s, y-t)u + I_y(x-s, y-t)v \tag{1.4}$$

Plugging equation (1.4) into equation (1.1) and simplifying, we have:

$$\begin{aligned}
E_w(u,v;x,y) &= \sum_{s,t} w(s,t)\left[I(x-s+u, y-t+v) - I(x-s,y-t)\right]^2 \\
&\approx \sum_{s,t} w(s,t)\left[I(x-s,y-t) + I_x(x-s,y-t)u + I_y(x-s,y-t)v - I(x-s,y-t)\right]^2 \\
&\approx \sum_{s,t} w(s,t)\left[I_x(x-s,y-t)u + I_y(x-s,y-t)v\right]^2 \\
&\approx \sum_{s,t} w(s,t)\Big[(I_x(x-s,y-t)u)^2 + 2\left(I_x(x-s,y-t)\cdot I_y(x-s,y-t)\cdot uv\right) + (I_y(x-s,y-t)v)^2\Big] \\
&\approx \sum_{s,t} w(s,t)[u,v]\begin{bmatrix} I_x(x-s,y-t)^2 & I_x(x-s,y-t)I_y(x-s,y-t) \\ I_x(x-s,y-t)I_y(x-s,y-t) & I_y(x-s,y-t)^2 \end{bmatrix}[u,v]^T \\
&\approx \sum_{s,t}[u,v]\begin{bmatrix} w(s,t)I_x(x-s,y-t)^2 & w(s,t)I_x(x-s,y-t)I_y(x-s,y-t) \\ w(s,t)I_x(x-s,y-t)I_y(x-s,y-t) & w(s,t)I_y(x-s,y-t)^2 \end{bmatrix}[u,v]^T \\
&\approx \sum_{s,t}[u,v]\mathcal{M}_w(x,y)[u,v]^T
\end{aligned} \tag{1.5}$$

Problem 2.

Show that the covariance matrix can be written equivalently as:

$$\mathcal{M}_w(x,y) = w(x,y) * \begin{bmatrix} I_x(x,y)I_x(x,y) & I_x(x,y)I_y(x,y) \\ I_y(x,y)I_x(x,y) & I_y(x,y)I_y(x,y) \end{bmatrix}, \tag{2.1}$$

where $*$ indicates convolution of the windowing function $w(x,y)$ with each element of the matrix.

Covariance matrix:

$$\mathcal{M}_w(x,y) =$$
$$\begin{bmatrix} \sum_{s,t} w(s,t)I_x(x-s,y-t)I_x(x-s,y-t) & \sum_{s,t} w(s,t)I_x(x-s,y-t)I_y(x-s,y-t) \\ \sum_{s,t} w(s,t)I_y(x-s,y-t)I_x(x-s,y-t) & \sum_{s,t} w(s,t)I_y(x-s,y-t)I_y(x-s,y-t) \end{bmatrix} \tag{2.2}$$

These are the elements in the covariance matrix:

$$\sum_{s,t} w(s,t)I_x(x-s,y-t)I_x(x-s,y-t) \tag{2.3}$$

$$\sum_{s,t} w(s,t)I_x(x-s,y-t)I_y(x-s,y-t) \tag{2.4}$$

$$\sum_{s,t} w(s,t)I_y(x-s,y-t)I_x(x-s,y-t) \tag{2.5}$$

$$\sum_{s,t} w(s,t)I_y(x-s,y-t)I_y(x-s,y-t) \tag{2.6}$$

Recall discrete function convolution:

$$(f * g)(x,y) = \sum_{s,t} f(s,t)g(x-s,y-t) \tag{2.7}$$

We notice that:

$$\sum_{s,t} w(s,t)I_x(x-s,y-t)I_x(x-s,y-t) = w(x,y) * I_x(x,y)I_x(x,y)$$
$$\sum_{s,t} w(s,t)I_x(x-s,y-t)I_y(x-s,y-t) = w(x,y) * I_x(x,y)I_y(x,y)$$
$$\sum_{s,t} w(s,t)I_y(x-s,y-t)I_x(x-s,y-t) = w(x,y) * I_x(x,y)I_y(x,y)$$
$$\sum_{s,t} w(s,t)I_y(x-s,y-t)I_y(x-s,y-t) = w(x,y) * I_y(x,y)I_y(x,y)$$

Therefore, we can express the covariance matrix as:

$$\mathcal{M}_w(x,y) = \begin{bmatrix} w(x,y) * I_x(x,y)I_x(x,y) & w(x,y) * I_x(x,y)I_y(x,y) \\ w(x,y) * I_y(x,y)I_x(x,y) & w(x,y) * I_y(x,y)I_y(x,y) \end{bmatrix} \tag{2.8}$$

If we define $*$ when applied to a matrix to be the convolution of a function with each element in the matrix, we have:

$$\mathcal{M}_w(x,y) = w(x,y) * \begin{bmatrix} I_x(x,y)I_x(x,y) & I_x(x,y)I_y(x,y) \\ I_y(x,y)I_x(x,y) & I_y(x,y)I_y(x,y) \end{bmatrix}, \tag{2.9}$$

Problem 3.

As we discussed in class, we can derive various "cornerness" metrics that take the form of functionals of *only* the product and sum of the eigenvalues of the covariance matrix.

Pick your favorite one (or propose your own), and explain how you would compute this metric efficiently for the entire image, using only convolutions and element-wise operations between images without explicitly computing eigenvalues. You can explain this either verbally, or using pseudocode.

I chose to implement the Harris corner metric, which is defined as:

$$R(x, y) = \mathbf{det}\, \mathcal{M}_w(x, y) - \kappa \cdot (\mathbf{tr}\, \mathcal{M}_w(x, y))^2 \tag{3.1}$$

where κ is a constant.

The main idea is that the product of a matrix's eigenvalues is equal to its determinant, and the sum of a matrix's eigenvalues is equal to its trace (the sum of elements along its diagonal).

Algorithm 1 Compute the Harris Corner Metric for an Image

procedure HARRIS($I,\ w,\ \kappa$)

$G^x_\sigma \leftarrow \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ ▷ Sobel derivative filter.

$I_x \leftarrow G^x_\sigma * I$

$I_{x^2} \leftarrow I_x \times I_x$

$I_y \leftarrow (G^x_\sigma)^T * I$

$I_{y^2} \leftarrow I_y \times I_y$

$I_{xy} \leftarrow I_x \times I_y$

$R \leftarrow [[0$ **for** $y \in I]$ **for** $x \in I]$ ▷ Initialize corner metric to zero for each pixel.

for $x \in I$ **do**

for $y \in I$ **do**

$M \leftarrow w(x,y) * \begin{bmatrix} I_{x^2}[x,y] & I_{xy}[x,y] \\ I_{xy}[x,y] & I_{y^2}[x,y] \end{bmatrix}$ ▷ Covariance matrix for current pixel.

$R[x,y] \leftarrow \mathbf{det}\, M - \kappa \times (\mathbf{tr}\, M)^2$ ▷ Corner metric for current pixel.

end for

end for

for $x \in I$ **do**

for $y \in I$ **do**

for $(i,j) \in \{-1,0,1\} \times \{-1,0,1\}$ **do**

if $R[x+i, y+j] > R[x,y]$ **then**

$R[x,y] \leftarrow 0$ ▷ Non-maximum suppression.

break

end if

end for

end for

end for

return R

end procedure