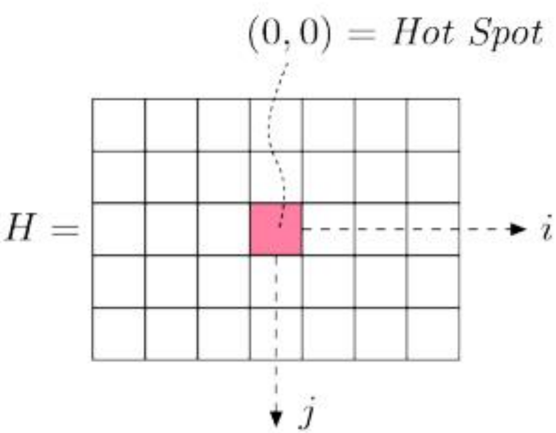
CSE185
Introduction to Computer Vision
Lab 07: Harris Corner Detection

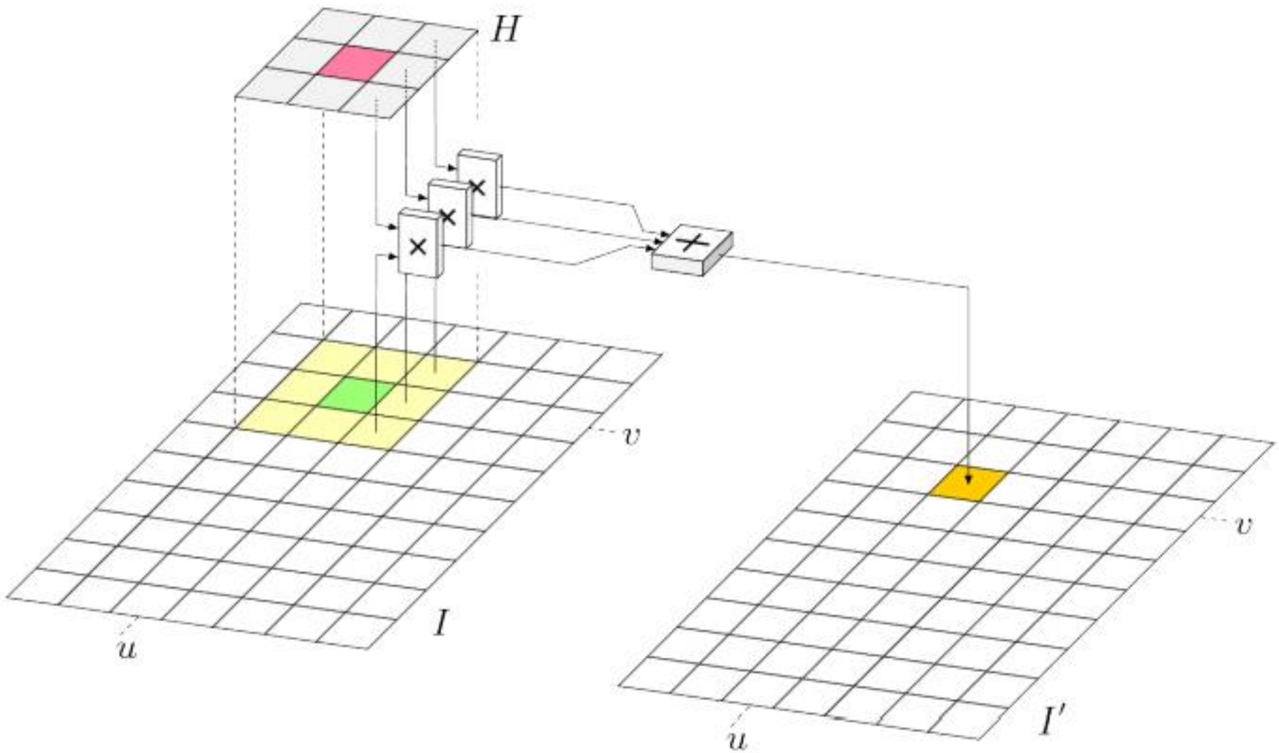
Instructor: Daniel Leung
TA: Mohammadkazem Ebrahimpour
Xueqing Deng

Spatial Filtering

- Filtering: sliding inner product

$$I'(u,v) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} I(u+i, v+j) \cdot H(i,j)$$





Spatial Filtering

- Filtering: sliding inner product

$$I'(u,v) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} I(u+i, v+j) \cdot H(i,j)$$

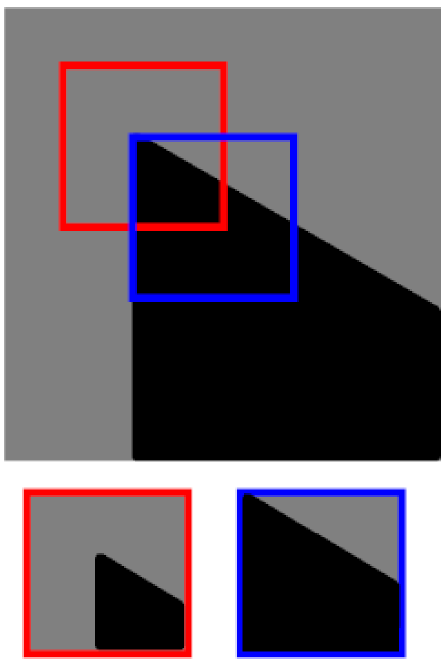
- Notation: $I' = I * H$ or $I' = I \otimes H$

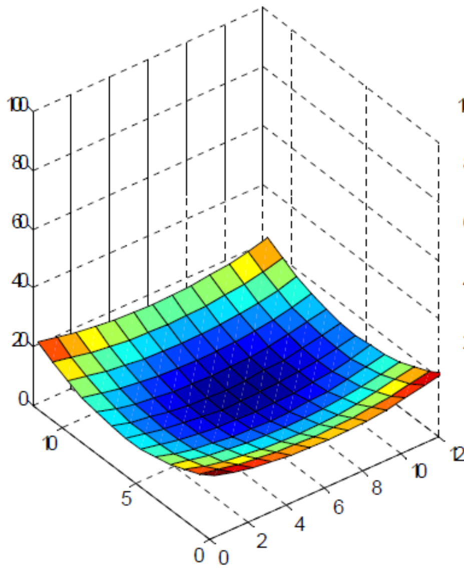
- In MATLAB, we use `imfilter(I, H)` to compute spatial filtering
  - use `imfilter(I, H, 'replicate')` to pad boundaries

Harris Corner Detection

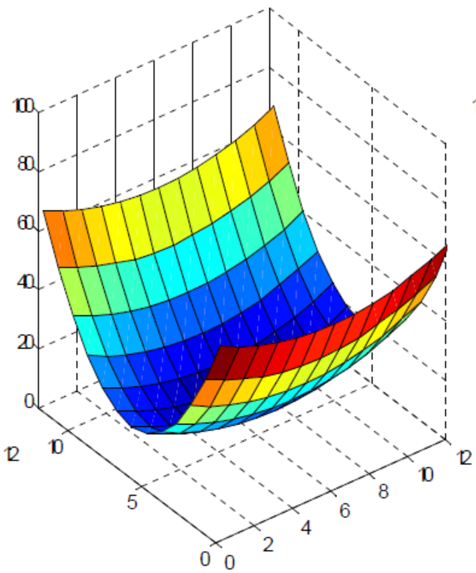
- Analysis the change of intensity for the shift $[u, v]$:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2$$

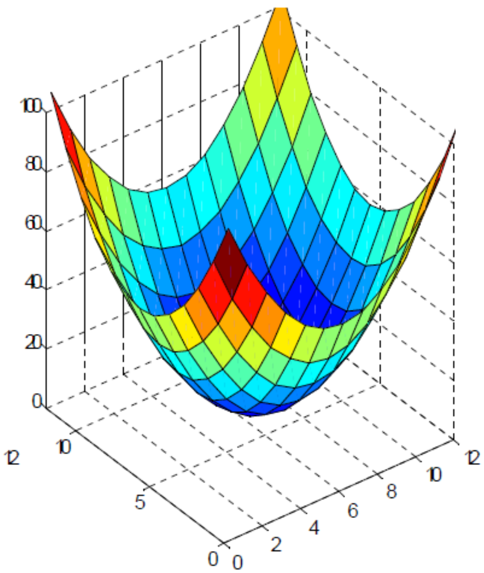




flat



edge



corner

Harris Corner Detection

- Approximate $E(u, v)$ with Taylor's expansion:

$$E(u, v) \cong [u \quad v] M \begin{bmatrix} u \\ v \end{bmatrix}$$

  – M is a 2×2 matrix computed from image gradients:

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

- Compute the corner response:

$$R = \det(M) - \alpha\big(\mathrm{trace}(M)\big)^2$$

determinant

Step 1: Image Gradients

- Use derivative of Gaussian (DoG) to compute image gradients:

$$I_x = \frac{\partial G}{\partial x} \otimes I$$
$$I_y = \frac{\partial G}{\partial y} \otimes I$$

  - G: Gaussian function/kernel

- Derivative can be calculated by convolution/filtering:
  - Sobel: $D_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$ and $D_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$
  - simple one: $D_x = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}$ and $D_y = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$

Step 1: Image Gradients

- Use derivative of Gaussian (DoG) to compute image gradients:

$$I_x = D_x \otimes G \otimes I$$

$$I_y = D_y \otimes G \otimes I$$

- Filtering is commutative and associative:

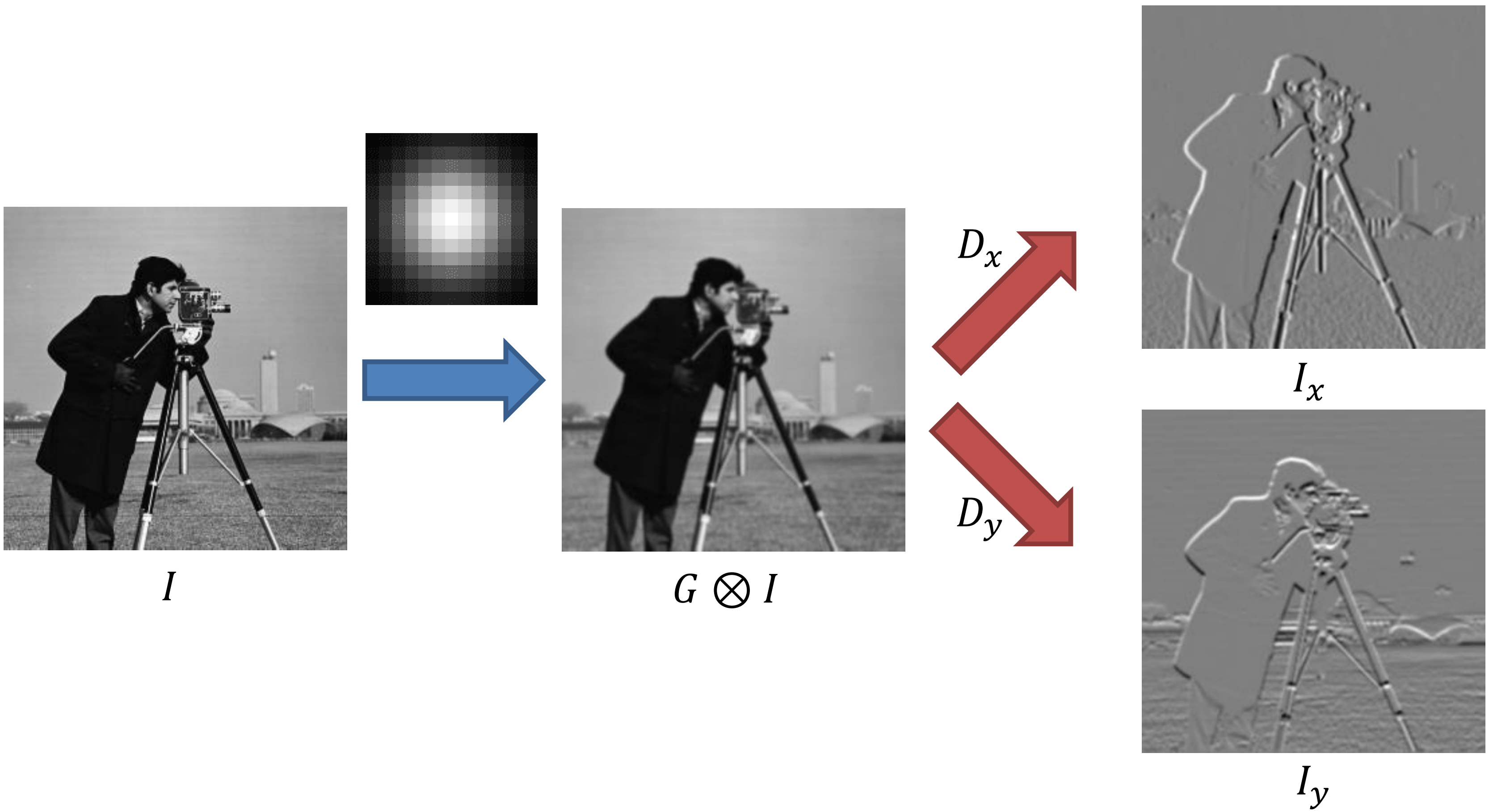
$$I_x = D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I = G \otimes (D_x \otimes I)$$

$$I_y = D_y \otimes (G \otimes I) = \left(D_y \otimes G\right) \otimes I = G \otimes \left(D_y \otimes I\right)$$

Step 1: Image Gradients

- Method 1: Apply Gaussian filtering to image, and then compute image gradients

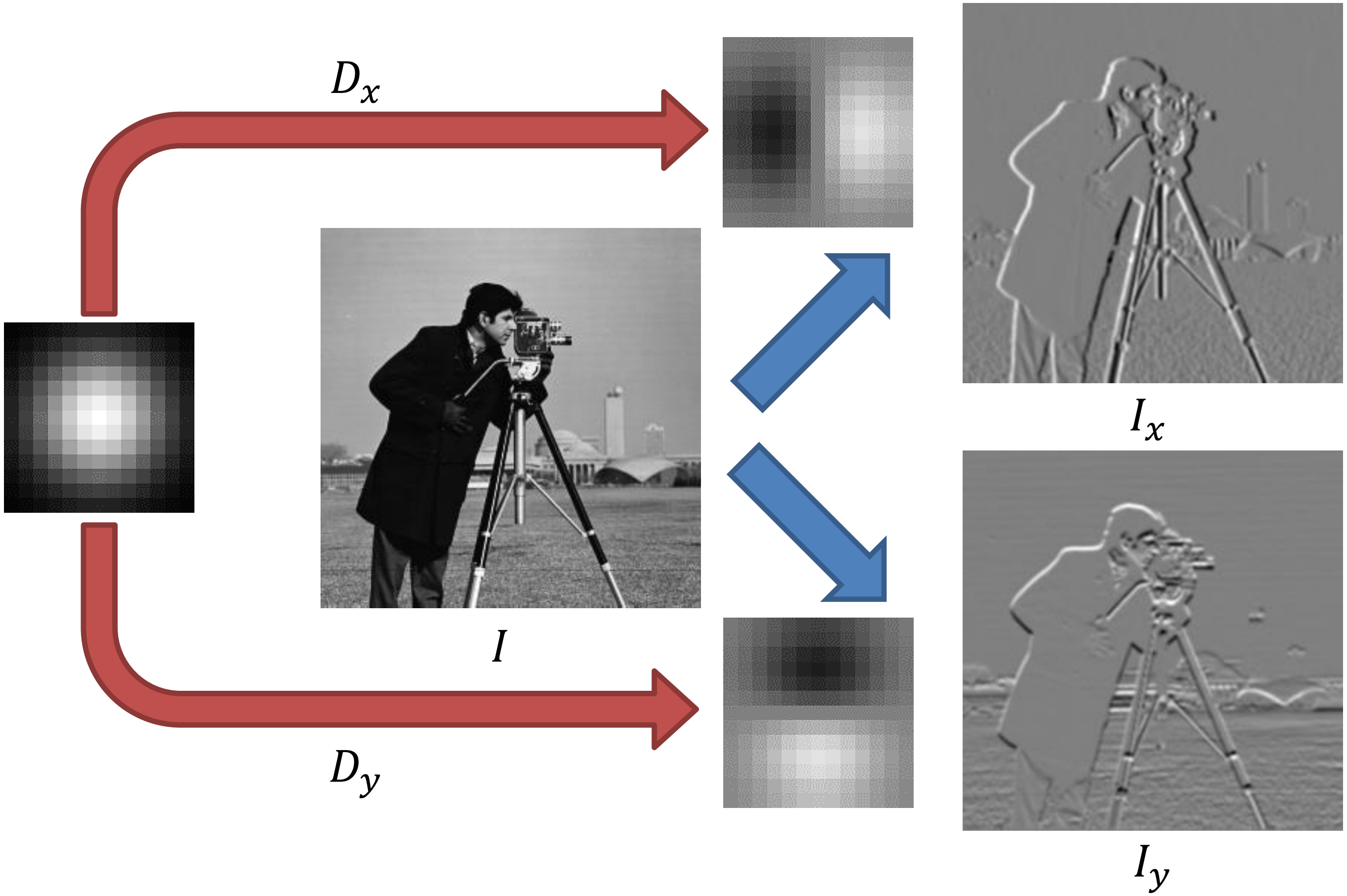
$$I_x = D_x \otimes (G \otimes I) \text{ and } I_y = D_y \otimes (G \otimes I)$$



Step 1: Image Gradients

- Method 2: Compute the derivative of Gaussian, and then apply filtering to image

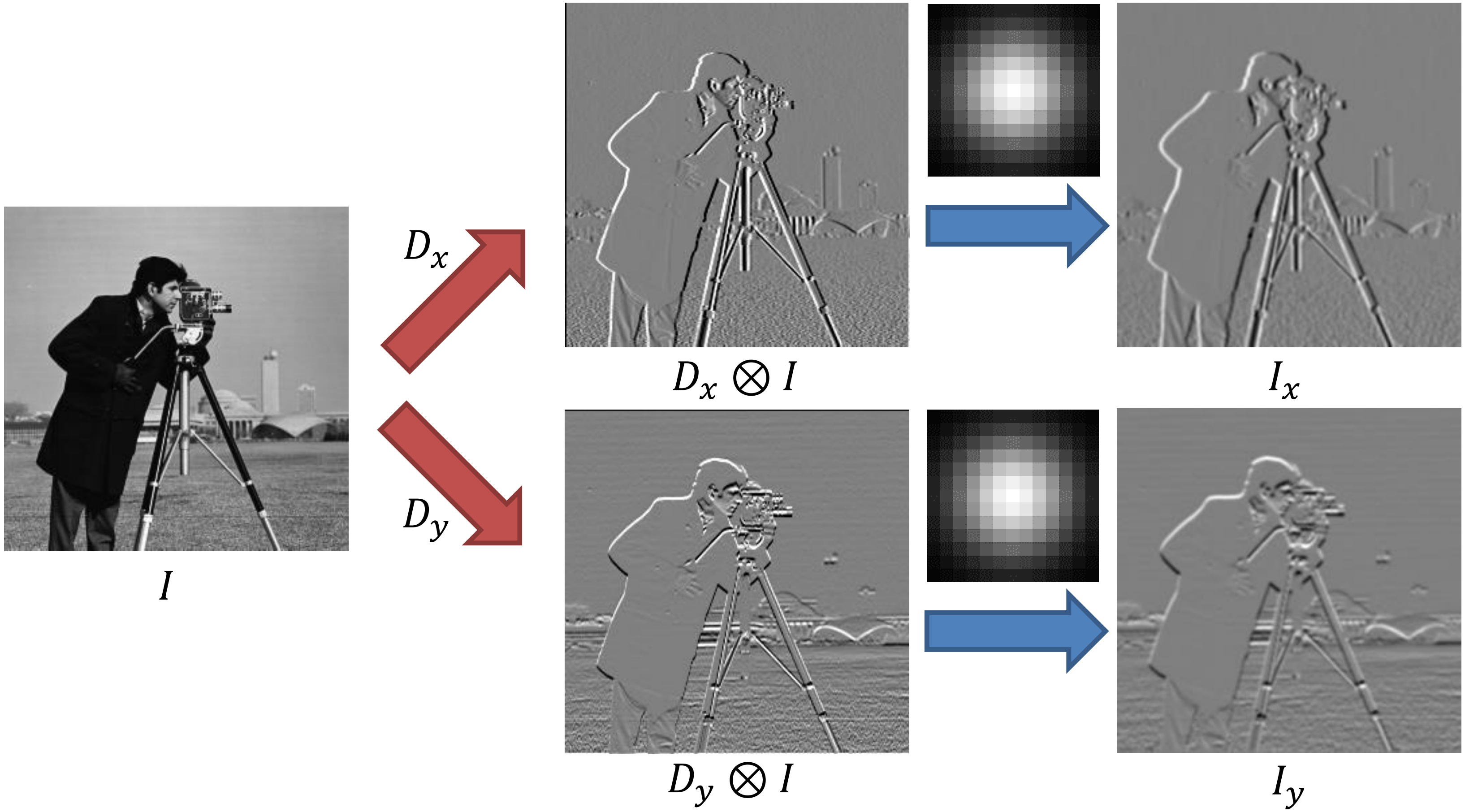
$$I_x = (D_x \otimes G) \otimes I \quad \text{and} \quad I_y = \left(D_y \otimes G\right) \otimes I$$



Step 1: Image Gradients

- Method 3: Compute the gradient of image, and then apply Gaussian filtering

$$I_x = G \otimes (D_x \otimes I) \text{ and } I_y = G \otimes (D_y \otimes I)$$



Step 2: Products of Gradients

- Compute products of gradients at every pixel

$$I_{xx} = I_x \cdot I_x$$

$$I_{yy} = I_y \cdot I_y$$

$$I_{xy} = I_x \cdot I_y$$

Use element-wise multiplication

Step 3: Matrix M

- Use Gaussian filtering to compute the sum of products of gradients at every pixel

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} G \otimes I_x^2 & G \otimes I_x I_y \\ G \otimes I_x I_y & G \otimes I_y^2 \end{bmatrix}$$

$$S_{xx} = G \otimes I_{xx}$$

$$S_{yy} = G \otimes I_{yy}$$

$$S_{xy} = G \otimes I_{xy}$$

Step 4: Corner Response

- Compute the determinant and the trace of M:

$$M = \begin{bmatrix} S_{xx} & S_{xy} \\ S_{xy} & S_{yy} \end{bmatrix}$$

element-wise
`.*`

$$\det(M) = S_{xx} \cdot S_{yy} - S_{xy} \cdot S_{xy}$$

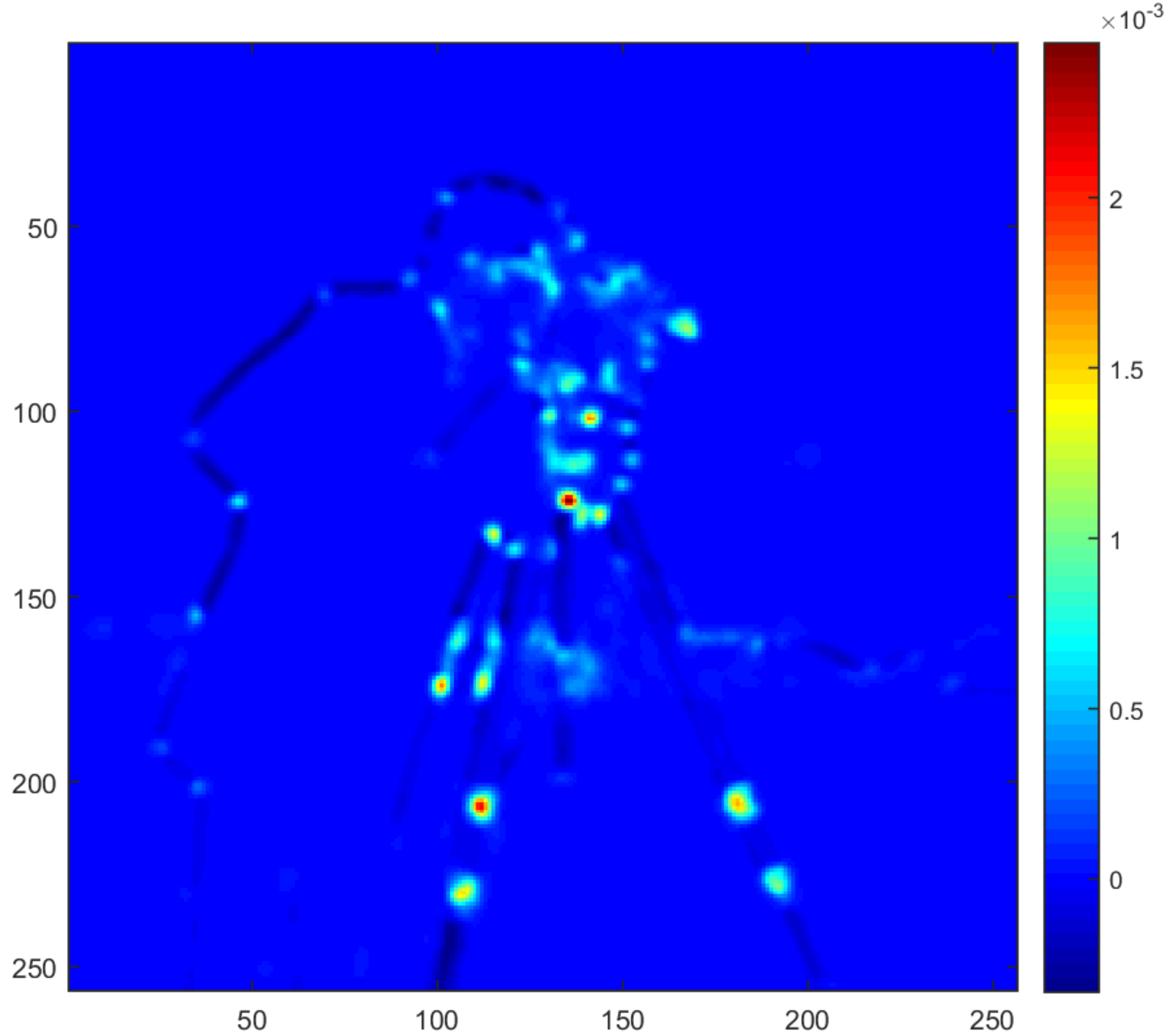
$$\mathrm{trace}(M) = S_{xx} + S_{yy}$$

element-wise
`.^2`

- Compute the corner response:

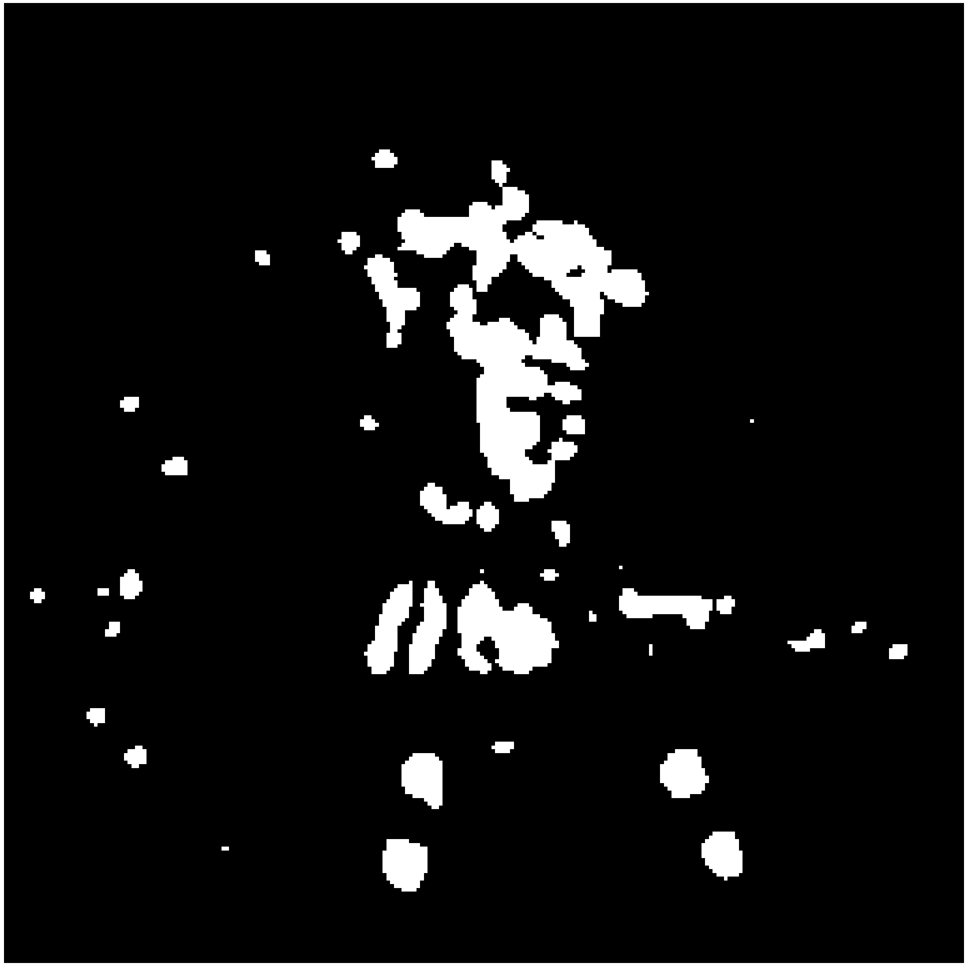
$$R = \det(M) - \alpha\left(\mathrm{trace}(M)\right)^2$$

Step 4: Corner Response



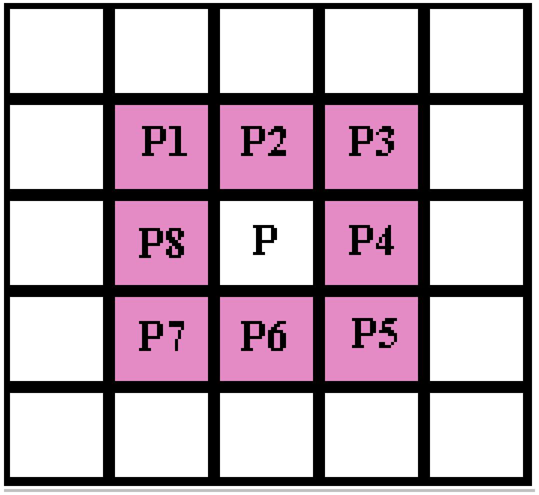
Step 4: Corner Response

- Apply thresholding on R: $R > R_{threshold}$



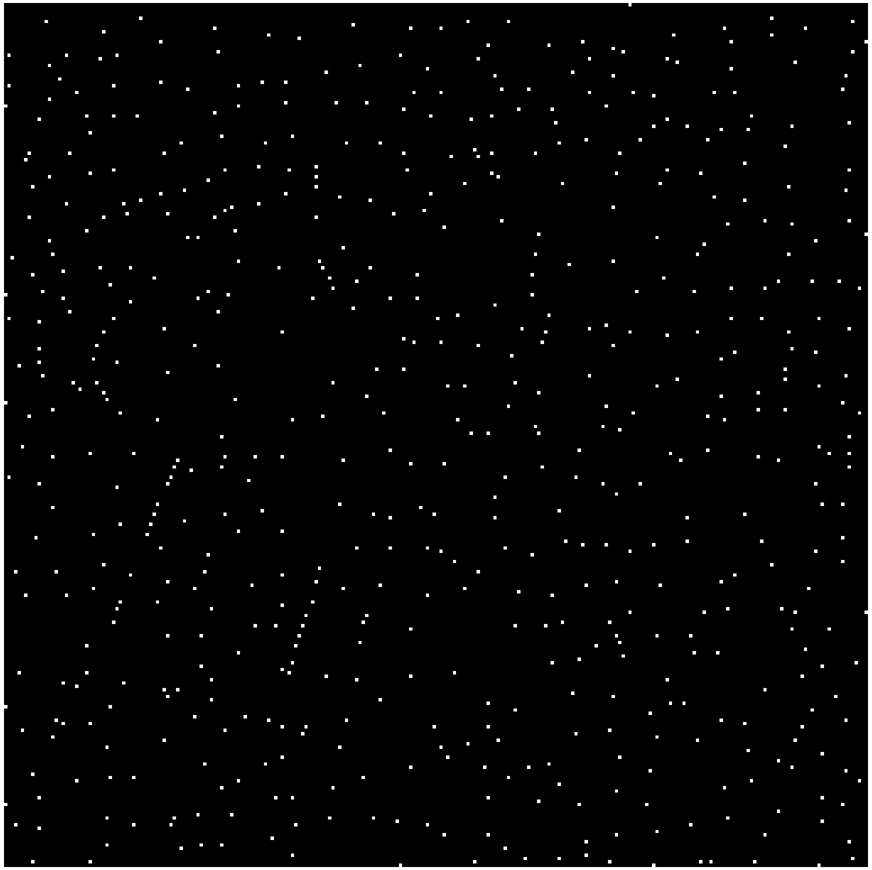
Step 5: Non-maxima Suppression

- Local maxima point should have *R* value greater than its all 8 neighbors
  - $R(x, y) > R(x-1, y-1)$
  - $R(x, y) > R(x-1, y)$
  - $R(x, y) > R(x-1, y+1)$
  - $R(x, y) > R(x, y-1)$
  - $R(x, y) > R(x, y+1)$
  - $R(x, y) > R(x+1, y-1)$
  - $R(x, y) > R(x+1, y)$
  - $R(x, y) > R(x+1, y+1)$



Step 5: Non-maxima Suppression

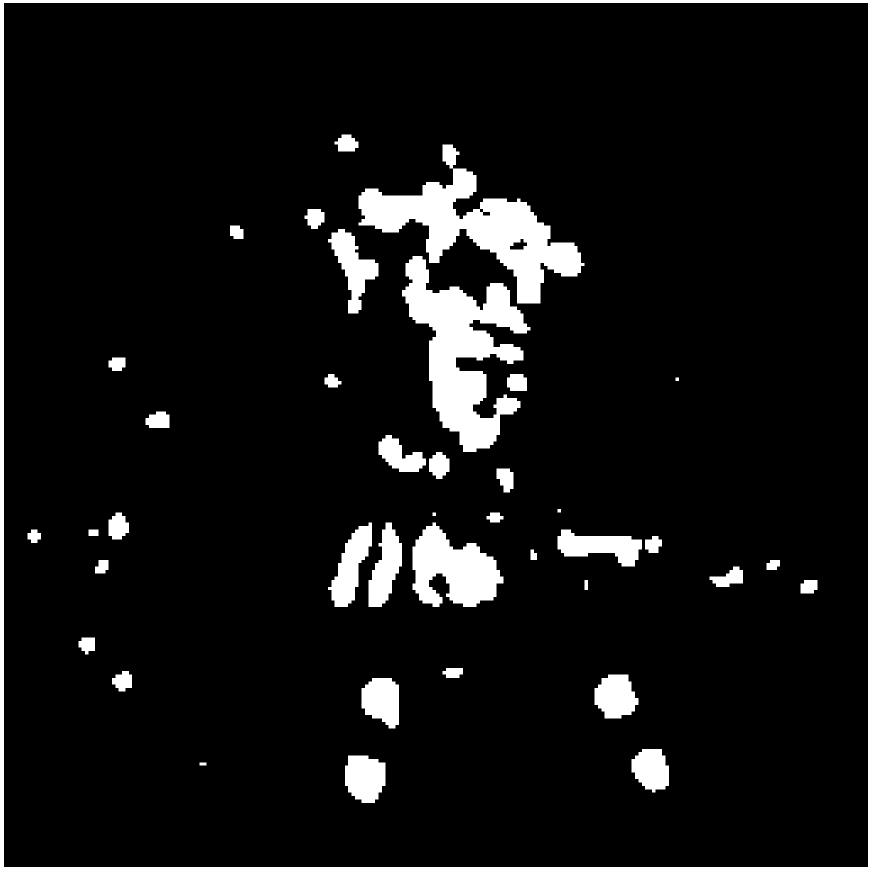
- In MATLAB, use `imregionalmax(R)`
- Bonus: implement your own function



Local Maxima

Step 5: Non-maxima Suppression

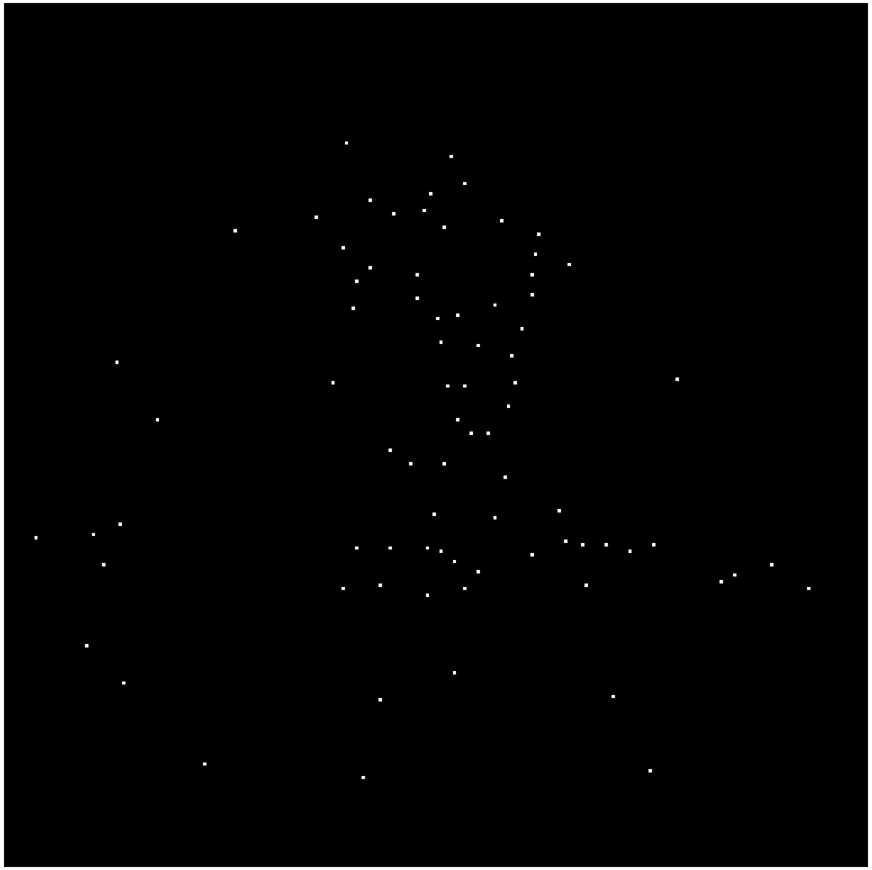
- Apply AND (&) to your corner map and local-maxima



Corner Map

Step 5: Non-maxima Suppression

- Apply AND (&) to your corner map and local-maxima



Final Corner Map

Step 6: Extract corner points and plot

- Use find to extract (x, y) of corner points:

```
[corner_y, corner_x] = find(final_corner_map);
```

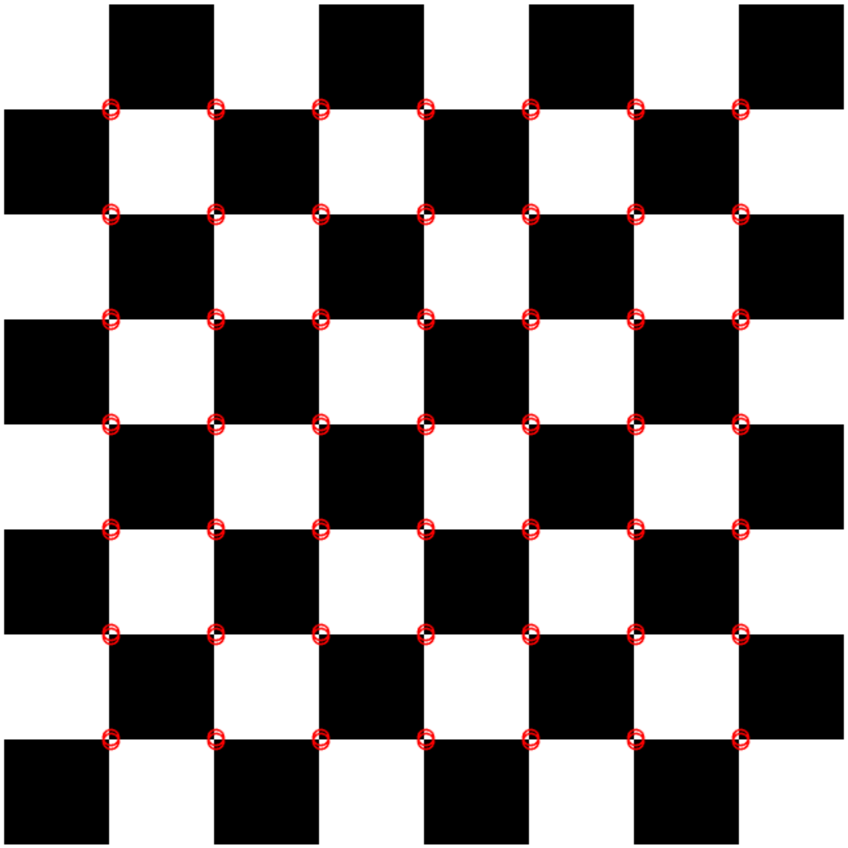
- Plot corners:

```
%% visualize results
figure, imshow(I); hold on;
plot(corner_x, corner_y, 'ro');
```

Results: cameraman

Results: baboon

Results: checkboard



Summary

1. Compute x and y gradients:

$$I_x = D_x \otimes (G_1 \otimes I) \text{ and } I_y = D_y \otimes (G_1 \otimes I)$$

2. Compute products of gradients

$$I_{xx} = I_x \cdot I_x \text{ and } I_{yy} = I_y \cdot I_y \text{ and } I_{xy} = I_x \cdot I_y$$

Different Gaussians

3. Apply Gaussian filtering

$$S_{xx} = G_2 \otimes I_{xx} \text{ and } S_{yy} = G_2 \otimes I_{yy} \text{ and } S_{xy} = G_2 \otimes I_{xy}$$

4. Compute corner response and apply thresholding

$$R = \det(M) - \alpha\big(\mathrm{trace}(M)\big)^2$$

5. Non-maxima suppression

TODO

- Implement Harris_corner_detector.m (20pt)
- Bonus: implement non-maxima suppression (20pt)
- Upload lab07.m, Harris_corner_detector.m and all output images