# CSE 152 Discussion Week 3

1/21/20

~~April 15, 2019~~

By: Yu-Ying Yeh

Narrated By: Steve Guerin

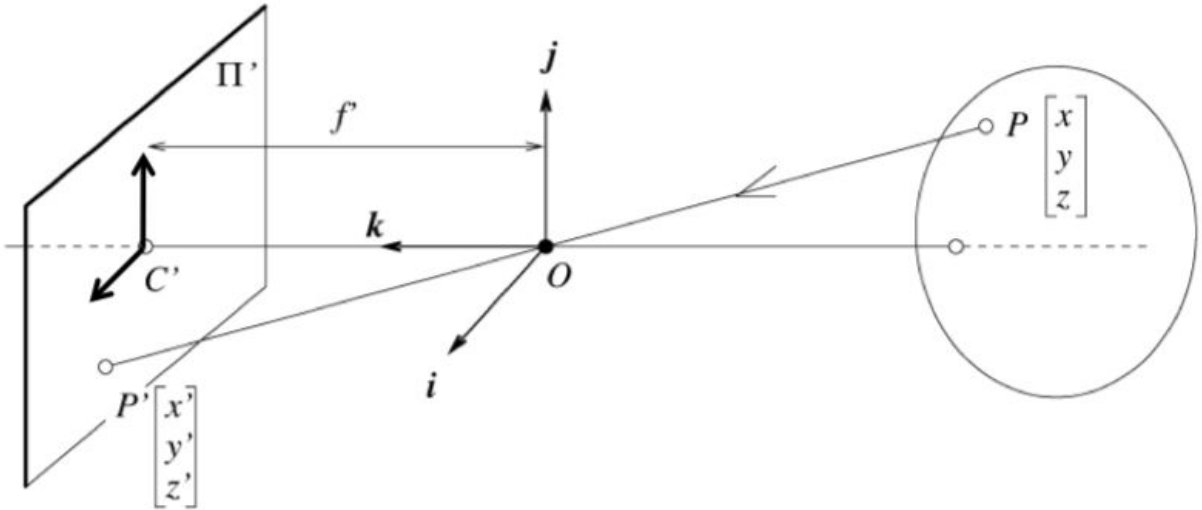
- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Outline

- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Geometric Image Formation

- Pinhole perspective projection

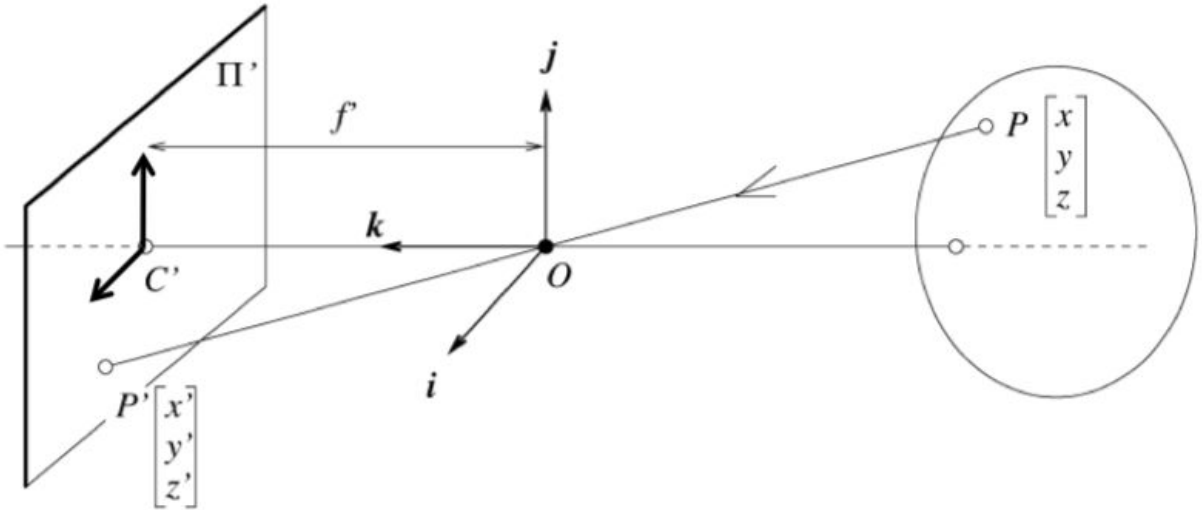



$$(x, y, z) \rightarrow (f' \frac{x}{z}, f' \frac{y}{z})$$

Not linear transformation

# Geometric Image Formation

- Pinhole perspective projection




$$(x, y, z) \rightarrow (f' \frac{x}{z}, f' \frac{y}{z})$$

Not linear transformation

Perspective projection become “linear”
In Homogeneous Coordinates

# Geometric Image Formation

- Project 4 points in 3D space onto camera 2D image under different cases

# Geometric Image Formation

- Project 4 points in 3D space onto camera 2D image under different cases
  - Transform 3D point from Euclidean coordinates to Homogeneous coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

# Geometric Image Formation

- Project 4 points in 3D space onto camera 2D image under different cases
  - Transform 3D point from Euclidean coordinates to Homogeneous coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - Camera Projection

$$M = K\Pi_w^c T = \underbrace{\begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters } 3 \times 3} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Projection } 3 \times 4} \underbrace{\begin{bmatrix} {}_w^c R & {}^c O_w \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\text{Extrinsic Parameters } 4 \times 4}$$

Rigid Transformation

$${}^B P = {}_A^B R {}^A P + {}^B O_A$$

Remember:

Inverse $R^{-1} = R^T$

# Geometric Image Formation

- Project 4 points in 3D space onto camera 2D image under different cases
  - Transform 3D point from Euclidean coordinates to Homogeneous coordinates

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

  - Camera Projection

$$M = K\Pi_w^c T = \underbrace{\begin{bmatrix} f & s & c_x \\ 0 & \alpha f & c_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Intrinsic Parameters } 3 \times 3} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\text{Projection } 3 \times 4} \underbrace{\begin{bmatrix} {}_w^c R & {}^c O_w \\ \mathbf{0}^T & 1 \end{bmatrix}}_{\text{Extrinsic Parameters } 4 \times 4}$$

Rigid Transformation

$${}^B P = {}_A^B R {}^A P + {}^B O_A$$

Remember:

Inverse $R^{-1} = R^T$

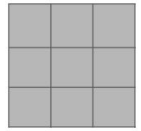
  - Transform 2D point from Homogeneous coordinates to Euclidean coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

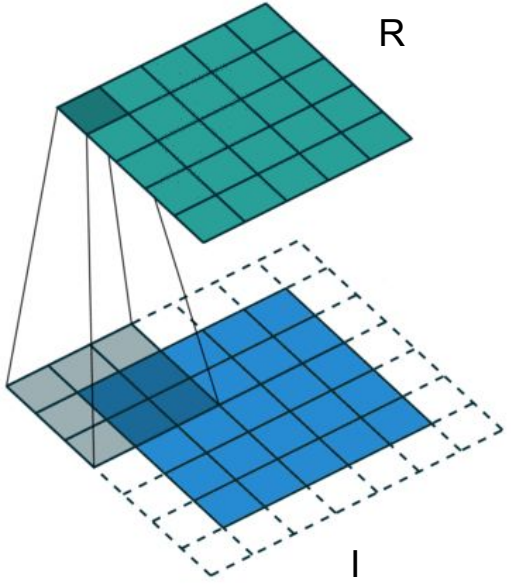
# Outline

- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Recap: Filtering



K



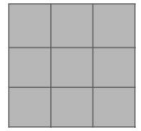

- 2D Convolution: Apply linear filters on image
  - Kernel is **flipped** over both axes

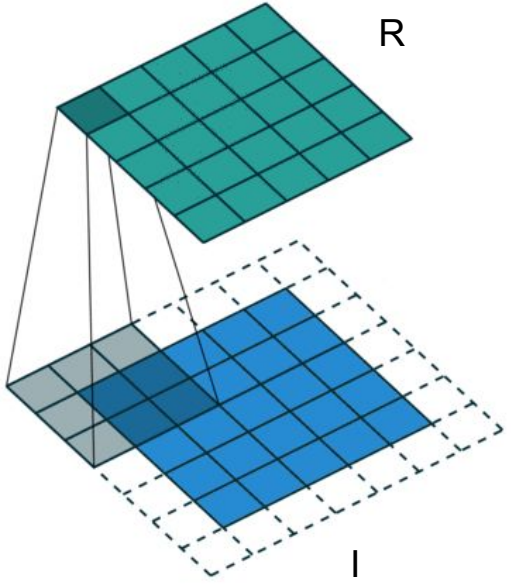
$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

- 2D Correlation:
  - Kernel is **not flipped**

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i+h, j+k)$$

# Recap: Filtering




- 2D Convolution: Apply linear filters on image
  - Kernel is **flipped** over both axes

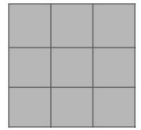
$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

- 2D Correlation:
  - Kernel is **not flipped**

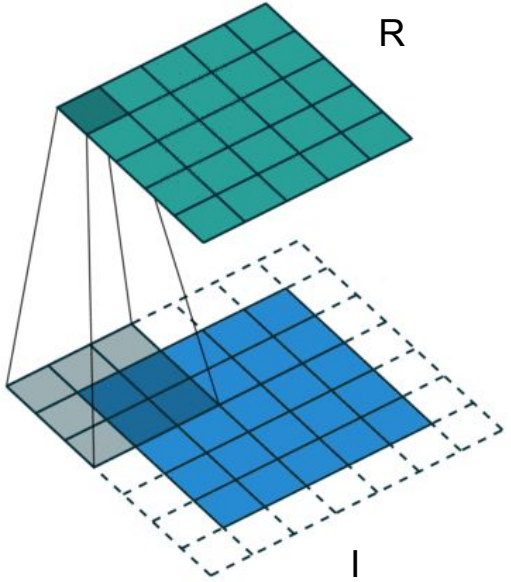
$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i+h, j+k)$$

- Zero-padding:
  - Pad zeros outside the original image
  - Question: how many pixels should we pad to maintain the same size after convolution?

# Recap: Filtering



K




- 2D Convolution: Apply linear filters on image
  - Kernel is **flipped** over both axes

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i-h, j-k)$$

- 2D Correlation:
  - Kernel is **not flipped**

$$R(i,j) = \sum_{h=-m/2}^{m/2} \sum_{k=-m/2}^{m/2} K(h,k) I(i+h, j+k)$$

- Zero-padding:
  - Pad zeros outside the original image
  - Question: how many pixels should we pad to maintain the same size after convolution?
  - Ans: $\left\lfloor \frac{m}{2} \right\rfloor$ on each side, m is kernel size

# Recap: Filtering

original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

Pixel offset

Filtered
(no change)

original

$\frac{1}{9}$

| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

BOX filter

Blurred (filter applied in both dimensions).

original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 0 |

Pixel offset

Shifted one
Pixel to the left

Original

| | | |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 2 | 0 |
| 0 | 0 | 0 |

$-\ \frac{1}{9}$

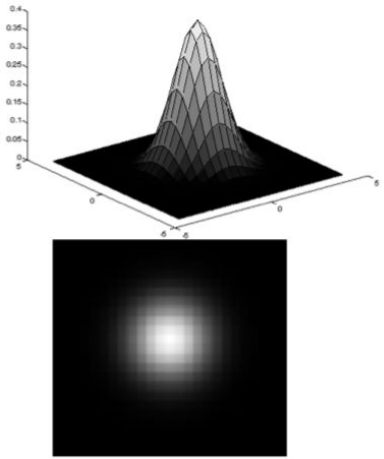
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 1 | 1 |
| 1 | 1 | 1 |

Sharpening filter
- Accentuates differences with local average

# Smoothing

- Gaussian Smoothing

## An Isotropic Gaussian



- Circularly symmetric Gaussian with variance $\sigma^2$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

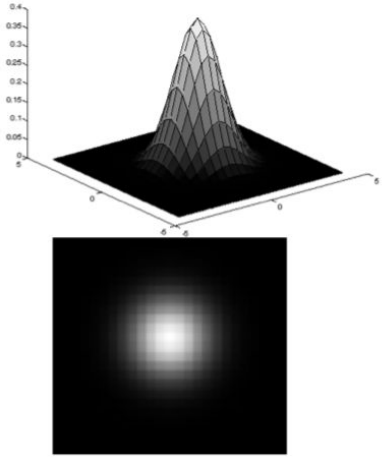
$\frac{1}{273}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

# Smoothing

- Gaussian Smoothing

An Isotropic Gaussian



- Circularly symmetric Gaussian with variance $\sigma^2$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

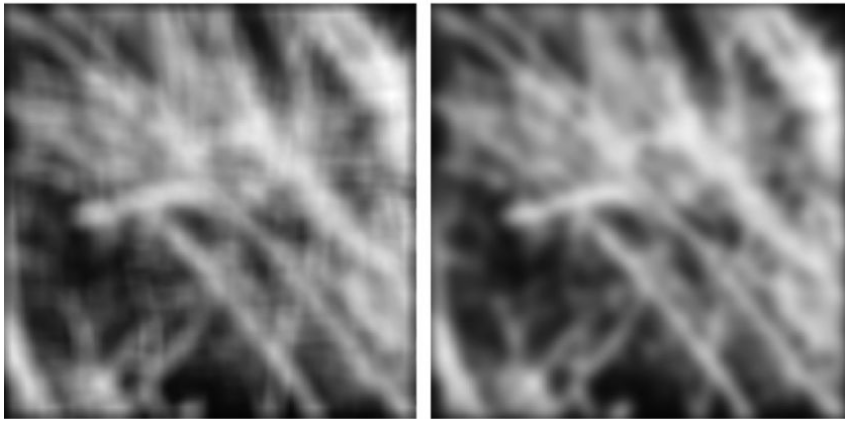
$\frac{1}{273}$

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 7 | 4 | 1 |
| 4 | 16 | 26 | 16 | 4 |
| 7 | 26 | 41 | 26 | 7 |
| 4 | 16 | 26 | 16 | 4 |
| 1 | 4 | 7 | 4 | 1 |

- Smoothing: BOX v.s. Gaussian

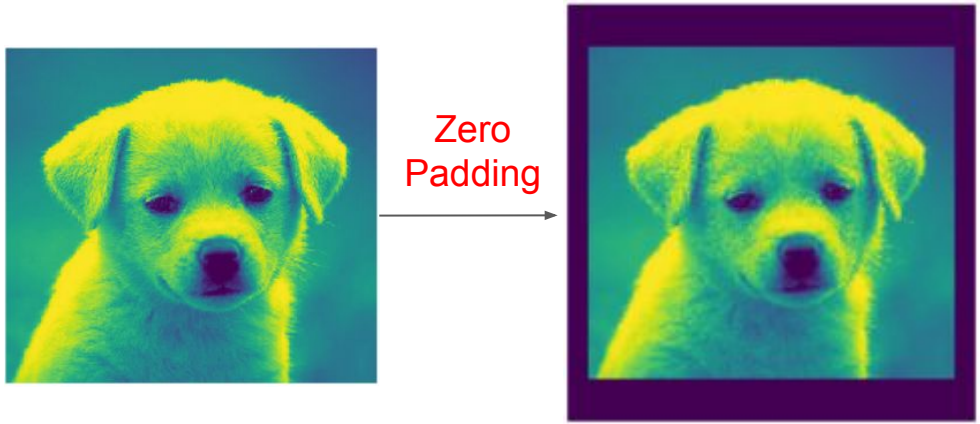
Box filter

Gaussian filter



# Filtering

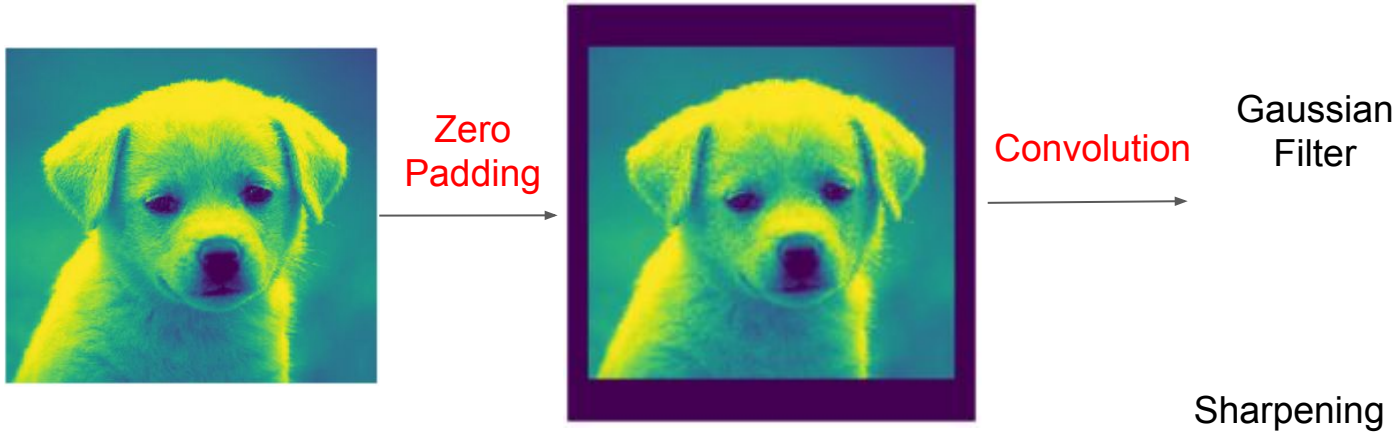
# Filtering



# Filtering

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

Sobel Filter

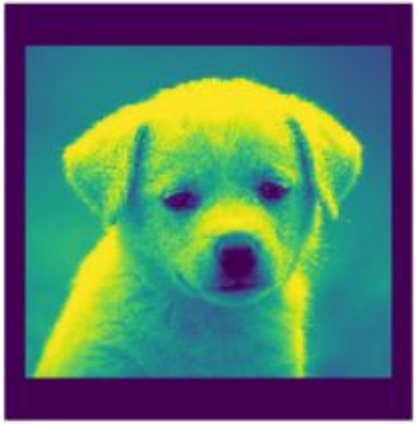



Gaussian Filter

Sharpening Filter

# Filtering

Zero Padding

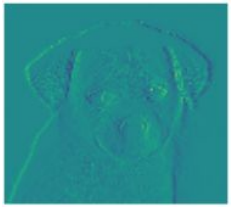


Convolution

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 2 | 0 | -2 |
| 1 | 0 | -1 |

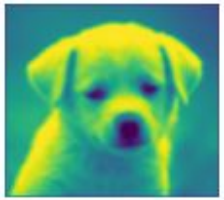
Sobel Filter

Image Gradient



Gaussian Filter

Blurred



Sharpening Filter

Sharpened

Try other filters yourself …
E.g., median filter, BOX filter ...

# Outline

- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Template Matching

- Template Image 
- Find the location with maximum response from the larger image

# Template Matching

- Template Image 
- Find the location with maximum response from the larger image

- Find the correlation instead of convolution
  - flip back template then apply convolution
- Subtract off the mean value of the image or template to make result not biased toward higher-intensity (white) regions

# Outline

- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Corner Detection

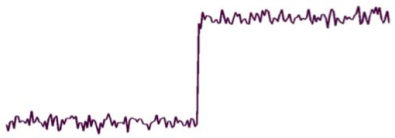
1. Filter image with a Gaussian
   a. Convolution with Gaussian filter
2. Compute the gradient everywhere
   a. Convolution with [1 0 -1] and [1 0 -1].T to get $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$
3. Compute C(x,y) over the window at every point
   a. Sum the $I_x^2$, $I_x I_y$ and $I_y^2$ over the window by convolution with a window of ones
4. Find $\lambda_1$ and $\lambda_2$ at each location by calculating eigenvalues of C
5. Detect corner if both $\lambda_1$ and $\lambda_2$ are large
   a. r(x,y) = min ( $\lambda_1(x,y)$, $\lambda_2(x,y)$ )
   b. Non-Maximum Suppression
      i. Detect corner if r(x,y) is local maximum within a window
      ii. Set response map of other non-maximum pixels to zeros

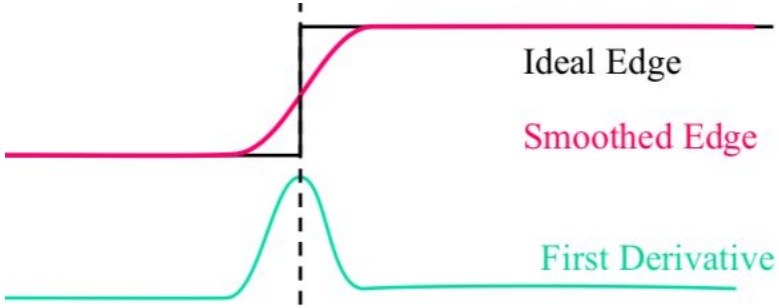
$$C(x,y) = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

# Corner Detection

1. Filter image with a Gaussian
   a. Convolution with Gaussian filter (from previous question)
2. Compute the gradient everywhere
   a. Convolution with [1 0 -1] and [1 0 -1].T to get $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$

Noisy step edge






$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h}$$

First Derivative: [1 0 -1]

# Corner Detection

3. Compute C(x,y) over the window at every point
   a. Sum the $I_x^2$, $I_x I_y$ and $I_y^2$ over the window by convolution with a window of ones

$I_x$

$I_y$

$$C(x,y) = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

# Corner Detection

3. Compute C(x,y) over the window at every point
   a. Sum the $I_x^2$, $I_x I_y$ and $I_y^2$ over the window by convolution with a window of ones

$$I_x \xrightarrow{\text{Elementwise Multiplication}} I_x^2$$

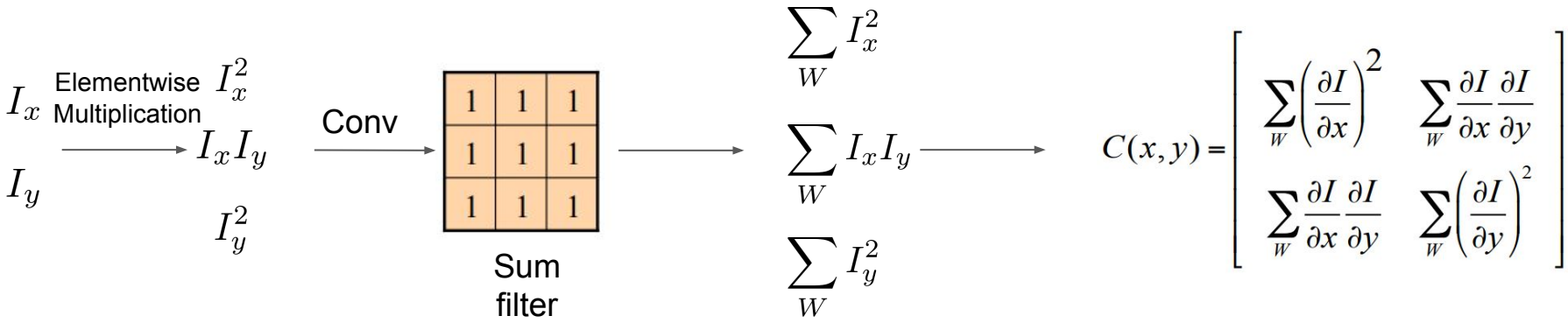
$$I_x, I_y \longrightarrow I_x I_y$$

$$I_y \longrightarrow I_y^2$$

$$C(x,y) = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

# Corner Detection

3. Compute C(x,y) over the window at every point
   a. Sum the $I_x^2$, $I_x I_y$ and $I_y^2$ over the window by convolution with a window of ones



# Corner Detection

4. Find $\lambda_1$ and $\lambda_2$ at each location by calculating eigenvalues of C

Because C is a symmetric positive definite matrix, it can be factored as:

$$C = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where R is a 2x2 rotation matrix. $\lambda_1$, $\lambda_2$ are non-negative and are Eigenvalues of C.

$$C(x,y) = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

# Corner Detection

4. Find $\lambda_1$ and $\lambda_2$ at each location by calculating eigenvalues of C

Because C is a symmetric positive definite matrix, it can be factored as:

$$C = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where R is a 2x2 rotation matrix. $\lambda_1$, $\lambda_2$ are non-negative and are Eigenvalues of C.

Solve $\boldsymbol{Av} = \lambda \boldsymbol{v}$

Is equivalent to solve $(\boldsymbol{A} - \lambda \boldsymbol{I})\,\boldsymbol{v} = \boldsymbol{0}$

When the linear system has non-trivial solutions,

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0$$

# Corner Detection

4. Find $\lambda_1$ and $\lambda_2$ at each location by calculating eigenvalues of C

Because C is a symmetric positive definite matrix, it can be factored as:

$$C = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where R is a 2x2 rotation matrix. $\lambda_1$, $\lambda_2$ are non-negative and are Eigenvalues of C.

Solve $\boldsymbol{Av} = \lambda \boldsymbol{v}$

Is equivalent to solve $(\boldsymbol{A} - \lambda \boldsymbol{I})\,\boldsymbol{v} = \boldsymbol{0}$

When the linear system has non-trivial solutions,

$$\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0$$

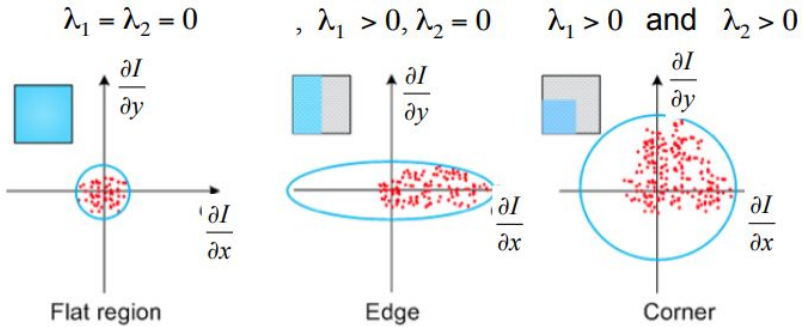
**Not necessarily need for loop until now**

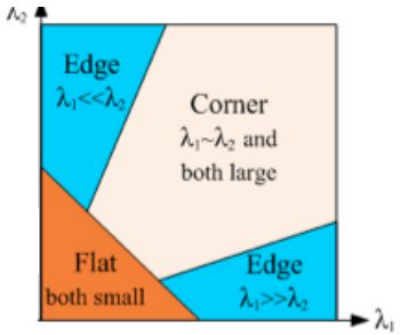
# Corner Detection

5. Detect corner if both $\lambda_1$ and $\lambda_2$ are large
   a. r(x,y) = min ( $\lambda_1(x,y)$, $\lambda_2(x,y)$ )
   b. Non-Maximum Suppression (NMS)
      i. Detect corner if r(x,y) is local maximum within a window
      ii. Set response map of other non-maximum pixels to zeros

$$C(x,y) = \begin{bmatrix} \sum_W \left(\frac{\partial I}{\partial x}\right)^2 & \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \\ \sum_W \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} & \sum_W \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix}$$

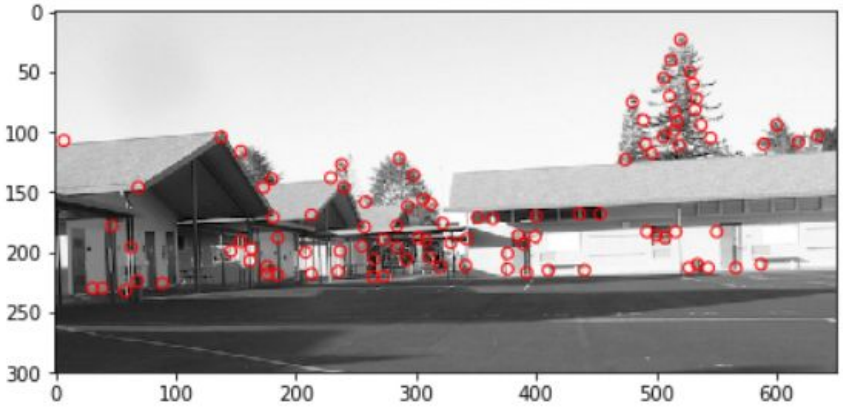
$$C = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

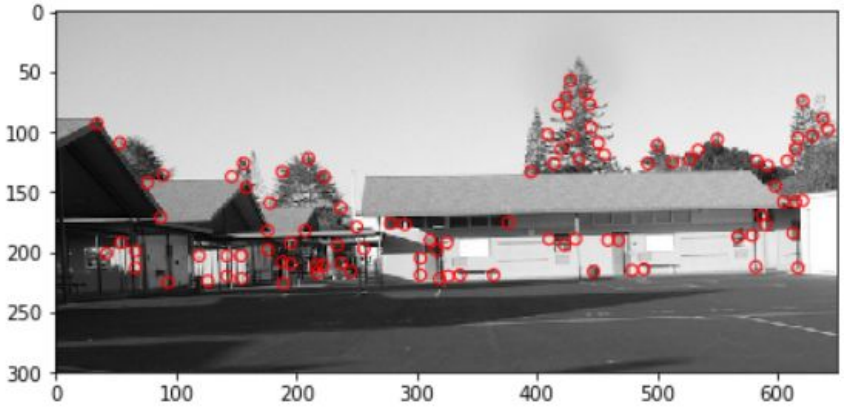


# Corner Detection

- Sample results



# Outline

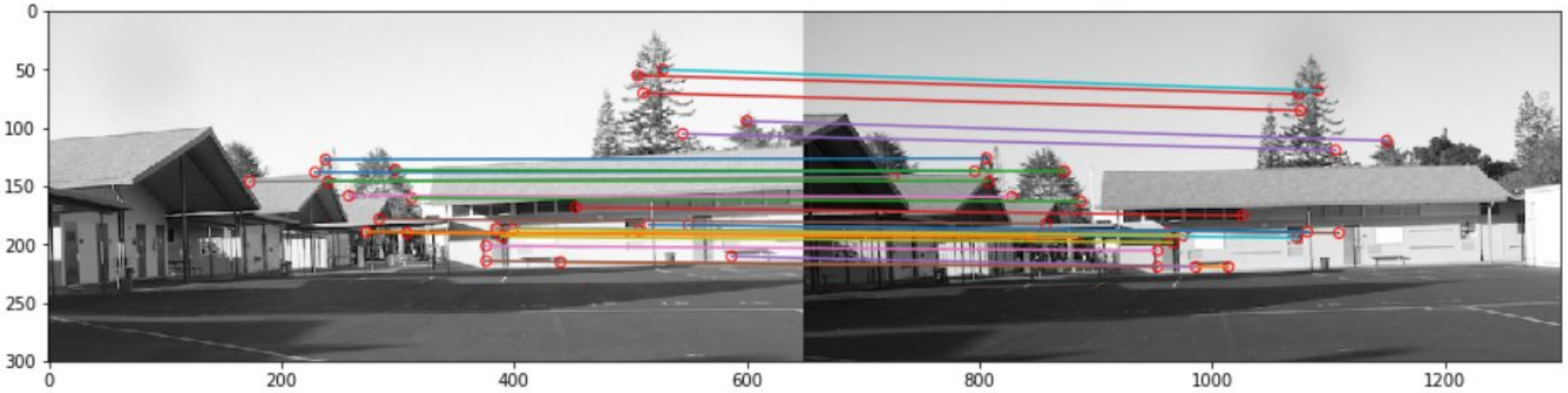
- Geometric Image Formation
- Filtering
- Template Matching
- Corner Detection
- Feature Matching using SIFT

# Feature Matching using SIFT

- SIFT feature extractor -> 128-d descriptor
- Distance: SSD (Sum of Squares Differences)

$$\mathrm{SSD}(u,v) = \sum_i (u_i - v_i)^2$$

- Accept matching if best match's distance is small enough (SSD threshold) and $\frac{best\ match's\ distance}{second\ best\ match's\ distance} \leq 0.3$ (Nearest Neighbor threshold)



# Q & A