SCC.366: Media Coding & Processing

2022-2023

Week 06 – Lecture 2

Advanced Edge Detectors

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- What is an Edge?
- Why do we care about edges?
- Origin of edges
- First order edge detectors
- **Canny edge detector**
- Edge detection using Laplacian
- Summary of the main points

Review - Image Derivative in 1D

- Forward, backward and central difference equations:

Forward difference $$\frac{df}{dx} = f(x+1) - f(x)$$

Backward difference $$\frac{df}{dx} = f(x) - f(x-1)$$

Central difference $$\frac{df}{dx} = f(x+1) - f(x-1)$$

Review - Edge Detection Operators

- Sobel edge detector

Kernel to calculate $\frac{\partial f}{\partial x}$

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

Kernel to calculate $\frac{\partial f}{\partial y}$

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

- Prewitt edge detector

Kernel to calculate $\frac{\partial f}{\partial x}$

-1	0	1
-1	0	1
-1	0	1

Kernel to calculate $\frac{\partial f}{\partial y}$

-1	-1	-1
0	0	0
1	1	1

- Roberts edge detector

1	0
0	-1

0	1
-1	0

Review - Gradient, Direction, Strength

- The *gradient* of an image at each pixel is a vector:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The edge strength is given by the gradient magnitude:

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

OR

$$||\nabla f|| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

More efficient approximation of the gradient magnitude

- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Review - Image Gradient (Visualisation)

Original image

Partial derivative in the x direction $\frac{\partial f}{\partial x}$

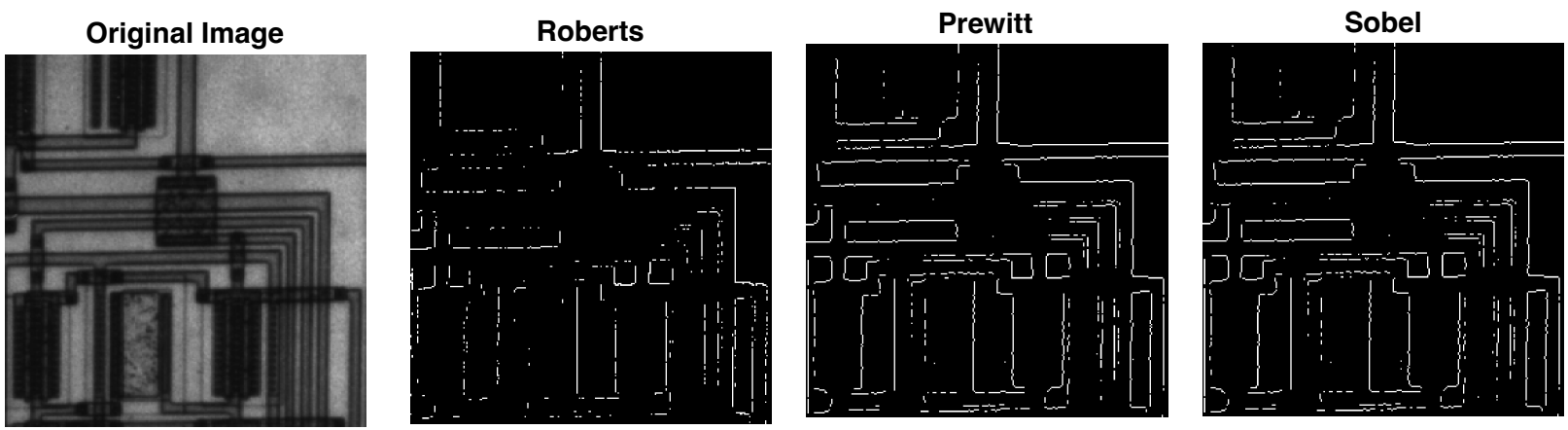
Partial derivative in the y direction $\frac{\partial f}{\partial y}$

The gradient magnitude $||\nabla f||$

Review - Edge Detection

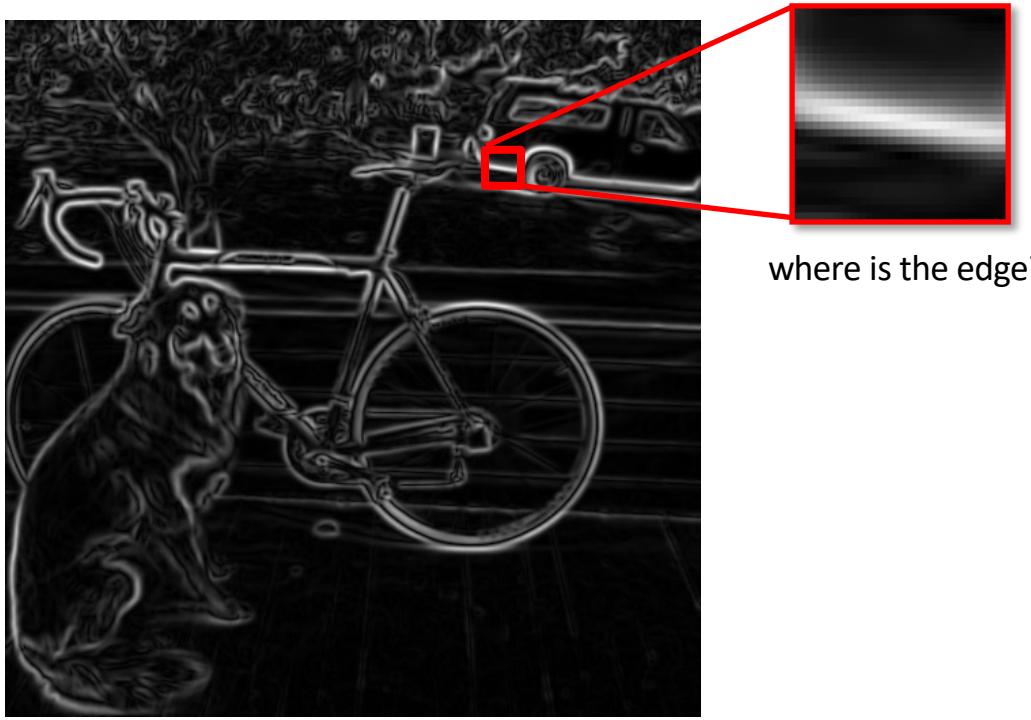
- **Thresholding:** Pixels with gradient magnitudes greater or equal to a threshold value are selected as edges
  - For example, threshold could be set to 33% of the maximum magnitude

$$B = \begin{cases} 0, & \left|\left|\nabla f\right|\right| < threshold \\ 1, & \left|\left|\nabla f\right|\right| \geq threshold \end{cases}$$



Where is the edge?

- **Edges are Thick**



Outline

- What is an Edge?
- Why do we care about edges?
- Origin of edges
- First order edge detectors
- **Canny edge detector**
- Edge detection using Laplacian
- Summary of the main points

Canny Edge Detector

- The Canny algorithm was introduced to further enhance edge detection
- The Canny algorithm can be summarized in five steps:
  1. Blurring the grayscale image
  2. Calculating the gradient magnitude and direction (Sobel)
  3. Non-maxima suppression
  4. Double thresholding
  5. Edge tracking by hysteresis

Canny Edge Detector (Step 1)

- Blurring the grayscale image
  - A Gaussian filter

Filter (5,5)

$$\frac{1}{273} \times \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Filter (3,3)

$$\frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Original Image

Blurred Image

Canny Edge Detector (Step 2)

- Calculating the gradient magnitude and direction
  - Any edge detectors such as: Sobel, Prewitt

Derivatives in x direction

Derivatives in y direction

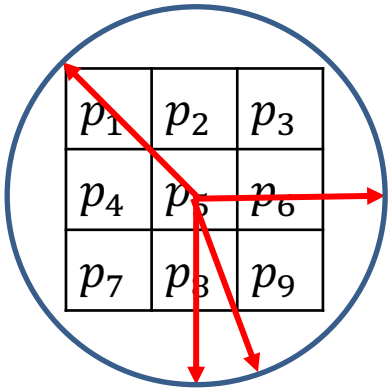
Gradient magnitudes

Canny Edge Detector (Step 3)

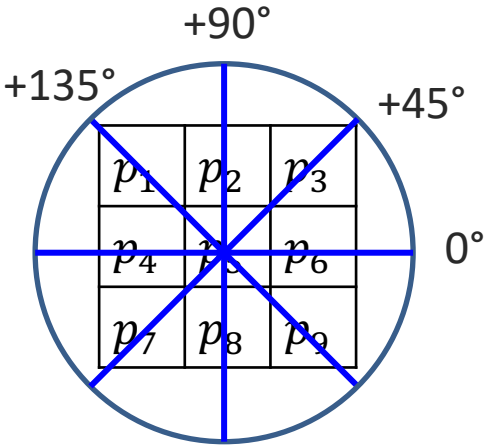
- Non-maxima suppression
  - This step is also known as edge thinning because it preserves the sharpest gradients and discards the others.
  - The image magnitude produced results in thick edges. Ideally, the final image should have thin edges.
  - The gradient magnitude of the current pixel is compared with the magnitude of the neighbouring pixels that point in the same direction.
  - The direction is rounded to four angles: 0, 45, 90, and 135.
    - For example, the angle is rounded to zero if it is in the range [0°,22.5°] or [157.5°, 180°].

Canny Edge Detector (Step 3)

- The gradient magnitude of the current pixel is compared with the magnitude of the neighbouring pixels that point in the same direction.
- The direction is rounded to four angles: 0, 45, 90, and 135.
  - For example, the angle is rounded to zero if it is in the range [0°,22.5°] or [157.5°, 180°].



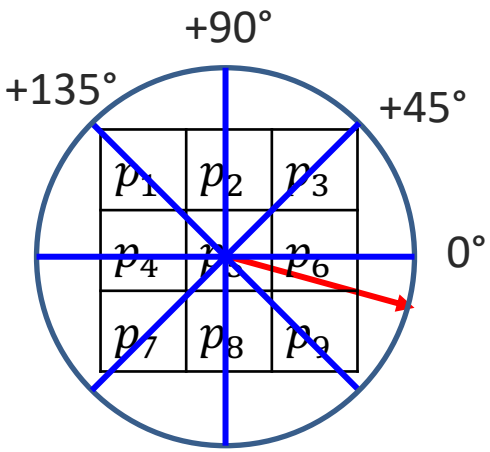
Gradient direction



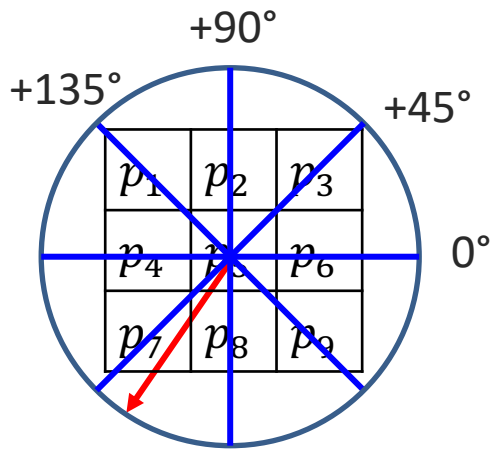
Quantized direction

Canny Edge Detector (Step 3)

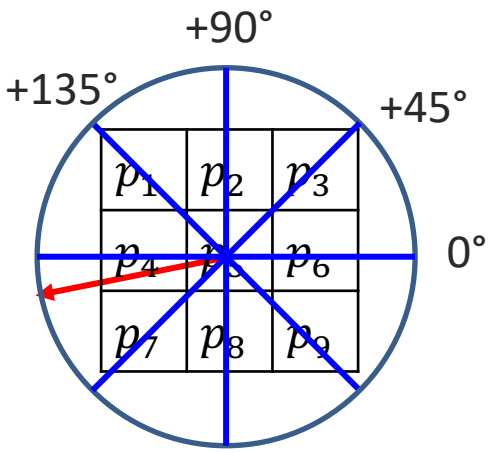
- The direction is rounded to four angles: 0, 45, 90, and 135.
  - For example, the angle is rounded to zero if it is in the range [0°,22.5°] or [157.5°, 180°].



Gradient direction is rounded to 0°



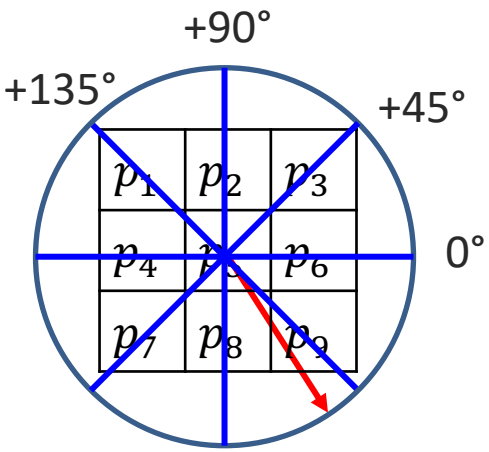
Gradient direction is rounded to 45°



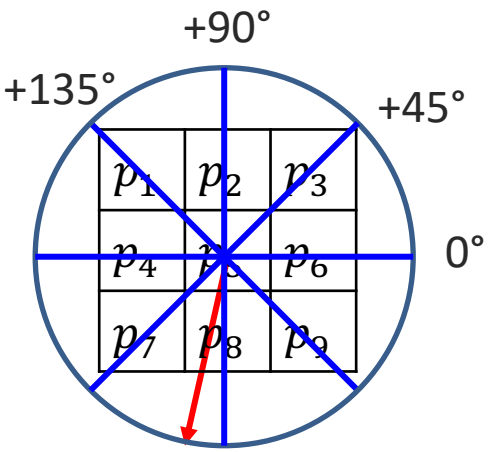
Gradient direction is rounded to 0°

Canny Edge Detector (Step 3)

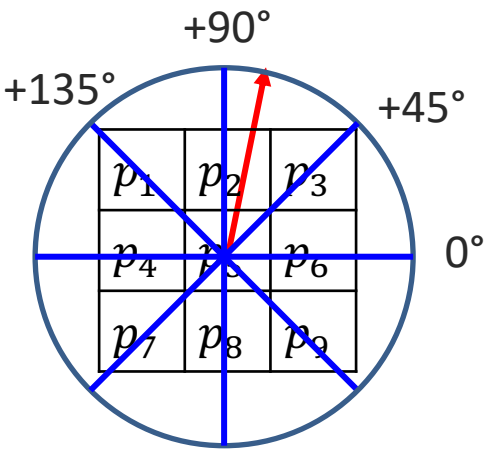
- The direction is rounded to four angles: 0, 45, 90, and 135.
  - For example, the angle is rounded to zero if it is in the range [0°,22.5°] or [157.5°, 180°].



Gradient direction is rounded to 135°



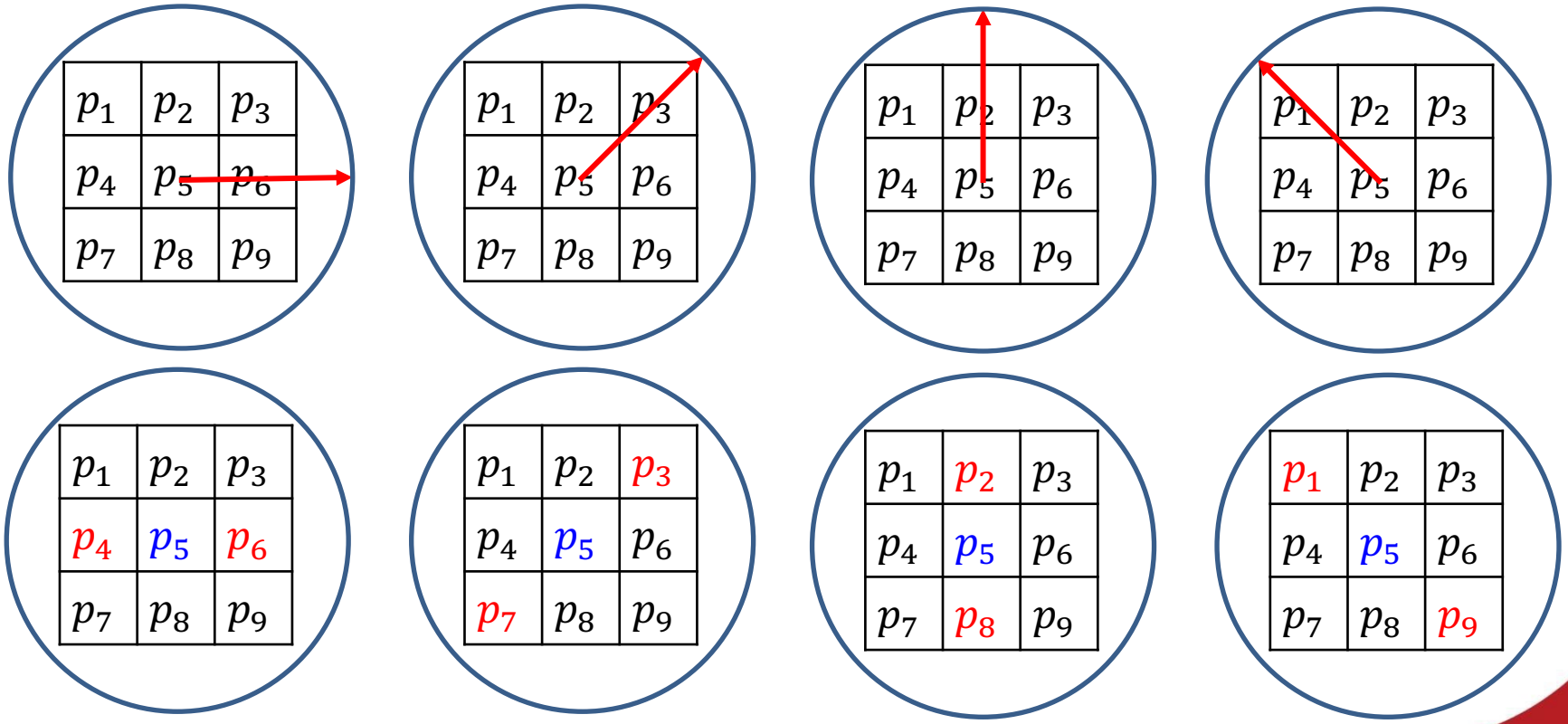
Gradient direction is rounded to 90°



Gradient direction is rounded to 90°

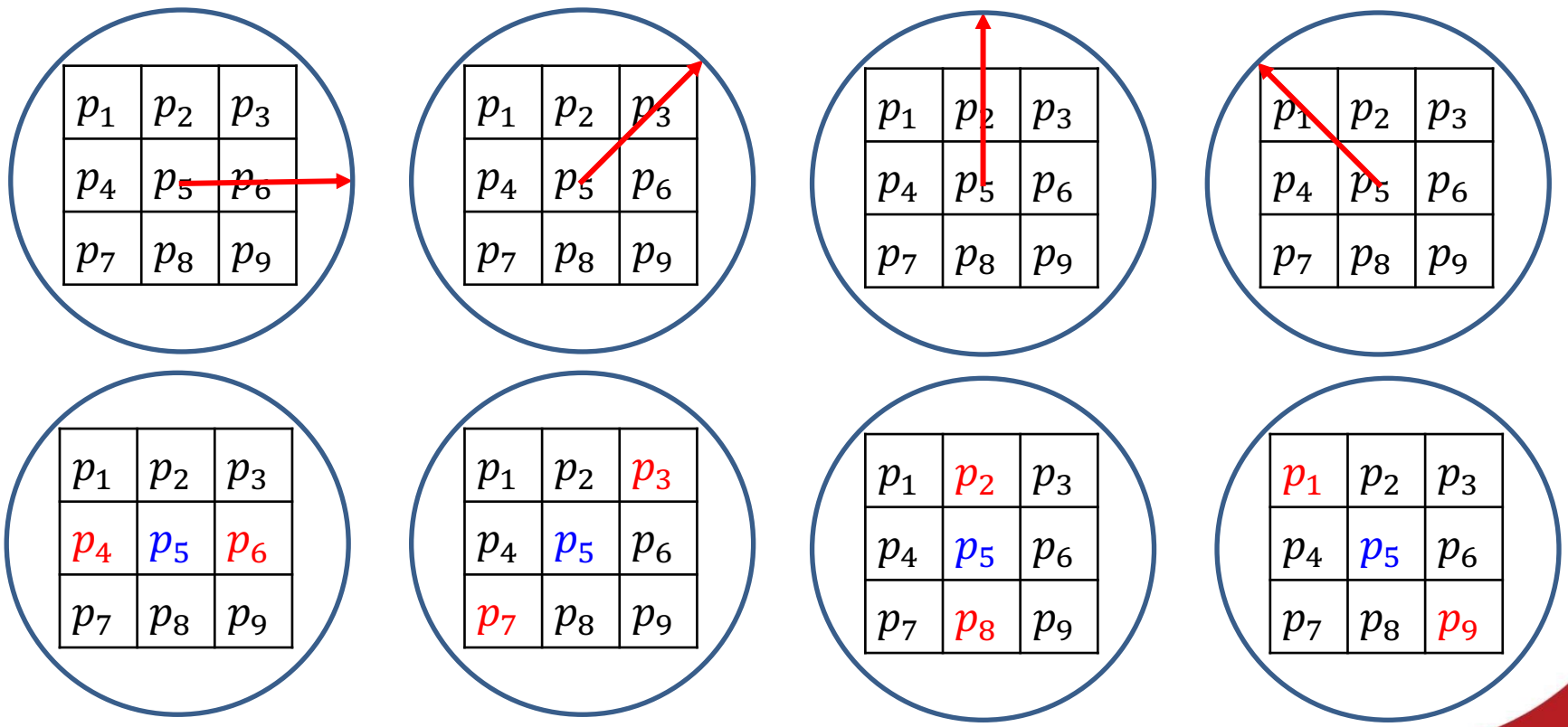
Canny Edge Detector (Step 3)

- The gradient magnitude of the current pixel is compared with the magnitude of the neighbouring pixels that point in the same direction.



Canny Edge Detector (Step 3)

- If the gradient magnitude of p_5 is less than of the gradient magnitude at one or both neighbours along quantized gradient direction, the gradient magnitude of p_5 is set to 0



Canny Edge Detector (Step 3)

- Non-maxima suppression

Gradient magnitudes

Non-maxima suppression

Canny Edge Detector (Step 4)

- Double thresholding
  - The algorithm classifies the magnitude in three categories strong, weak, and non-edge pixels
    - Pixels with gradient magnitudes less than the low threshold are discarded, i.e their values are set to zero.
    - Pixels with gradient magnitudes higher than the high threshold are chosen to be strong edges and they appear in the final result.
    - If the gradient magnitude of a pixel falls in between the two thresholds, it is considered to be a weak edge.

Canny Edge Detector (Step 4)

- Double thresholding

Non-maxima suppression

Double Thresholding

Canny Edge Detector (Step 5)

- Edge tracking by hysteresis
  - Weak edges are either taken to the strong edges group or to the non-edge group.
  - Weak edges could be meaningful, or, they are just caused by noise.
  - Normally, true edges are connected to the strong edges while the one that is caused by noise is not connected. Based on that, the algorithm is as follows:
    - For each of the weak edges, compare the magnitude of the pixel being processed with its eight surrounding pixels. If at least one of these pixels belongs to the strong edges group, the value of the pixel being processed is preserved. Otherwise, it is discarded.

Canny Edge Detector (Step 5)

- Edge tracking by hysteresis

Double Thresholding

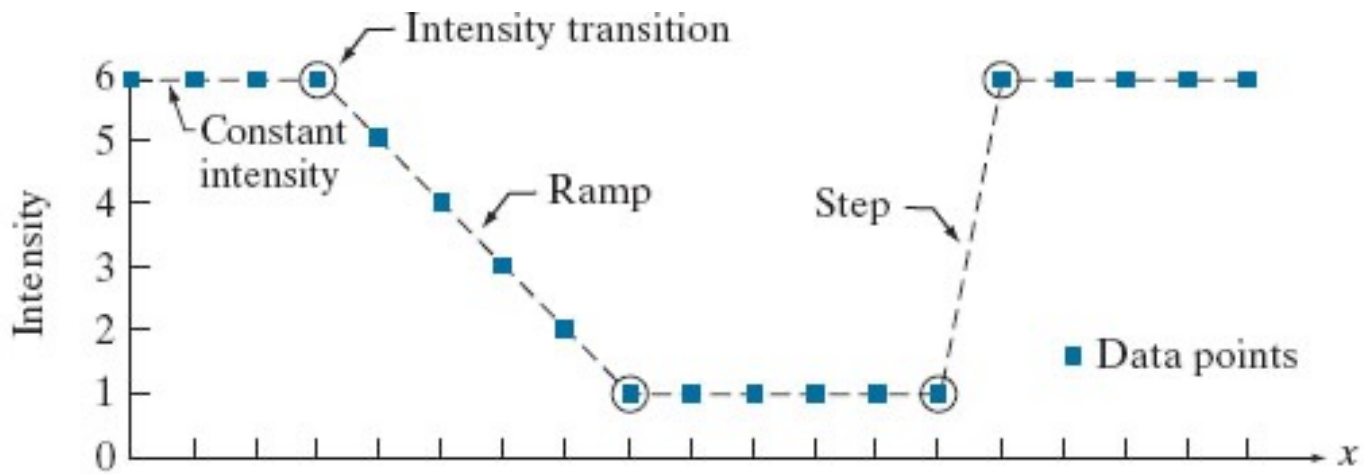
Edge Tracking

Outline

- What is an Edge?
- Why do we care about edges?
- Origin of edges
- First order edge detectors
- Canny edge detector
- **Edge detection using Laplacian**
- Summary of the main points

First Order vs. Second Order

- First derivative is positive at the onset of the ramp and at points on the ramp, and it is zero in areas of constant intensity.



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	

Forward difference $\frac{df}{dx} = f(x+1) - f(x)$

How to Calculate Second Derivative?

- We use forward difference equation to calculate 2nd derivative

Forward difference $\frac{df}{dx} = f(x+1) - f(x)$

$$\frac{d^2f}{dx^2} = f'(x+1) - f'(x)$$

Using backward difference

$$f'(x+1) = f(x+1) - f(x)$$

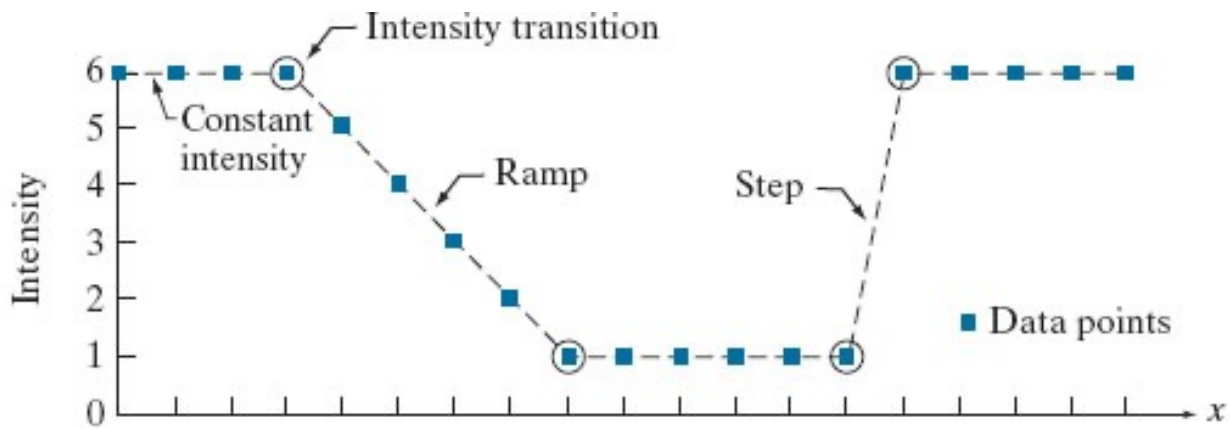
$$f'(x) = f(x) - f(x-1)$$

$$\frac{d^2f}{dx^2} = [f(x+1) - f(x)] - [f(x) - f(x-1)]$$

$$\boxed{\frac{d^2f}{dx^2} = f(x+1) - 2f(x) - f(x-1)}$$

First Order vs. Second Order

- Second derivative is positive at the beginning of the ramp, negative at the end of the ramp, zero at points on the ramp, and zero at points of constant intensity.



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

$$\frac{d^2 f}{dx^2} = f(x+1) - 2f(x) + f(x-1)$$

Laplacian Edge Detector

- The Laplacian edge detectors vary from the previously discussed edge detectors.
- This method uses **only one filter**
- The Laplacian $L(x, y)$ of an image with pixel intensity values $f(x, y)$ is given by:

$$L(x, y) = \frac{d^2 f}{dx^2} + \frac{d^2 f}{dy^2}$$

1	1	1
1	-8	1
1	1	1

0	1	0
1	-4	1
0	1	0

Common Laplacian edge detection filters

Laplacian Edge Detector

- In a single pass, Laplacian detection performs **second-order derivatives** and hence are sensitive to noise.
- To avoid this sensitivity to noise, before applying this method, **Gaussian smoothing is performed** on the image.
- In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result.

Laplacian Edge Detector

- In fact, since the convolution operation is associative, we can convolve the Gaussian smoothing filter with the Laplacian filter first of all, and then convolve this hybrid filter with the image to achieve the required result.
- **Advantage:**
  - The LoG (Laplacian of Gaussian) kernel can be precalculated in advance so only one convolution needs to be performed at run-time on the image

0.4	0.8	0.4
0.8	-8	0.8
0.4	0.8	0.4

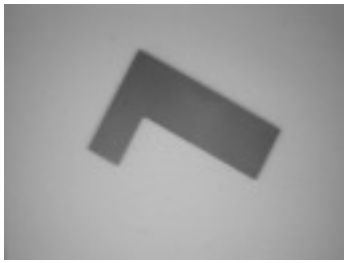
Laplacian Edge Detector

- The LoG operator calculates the second spatial derivative of an image. This means that
  - In areas where the image has a constant intensity (i.e. where the intensity gradient is zero), the LoG response will be zero.
  - At a reasonably sharp edge between two regions of uniform but different intensities, the LoG response will be:
    - Zero at a long distance from the edge
    - Positive just to one side of the edge,
    - Negative just to the other side of the edge
    - Zero at some point in between, on the edge itself

Laplacian Edge Detector

- Example:

Image



LoG

- Note that the output contains negative and non-integer values, so for display purposes the image has been normalized to the range 0 - 255.

Image Sharpening

- What will happen if the LoG filtered image is added to the original image? then the result will be to make any edges in the original image much sharper and give them more contrast.

Image

+

LoG

Image Sharpening

- What will happen if the LoG filtered image is added to the original image?
  - The original image becomes much sharper and give them more contrast

Image

+

LoG

Sharpened Image

Summary

- Canny Edge detector further enhances edge detection
- The Canny algorithm can be summarized in five steps:
  - Blurring the grayscale image
  - Calculating the gradient magnitude and direction (Sobel)
  - Non-maxima suppression
  - Double thresholding
  - Edge tracking by hysteresis
- Laplacian edge detector uses 2nd order derivatives and is more sensitive to noise than first order edge detector
- Laplacian operator can be used in increase image contrast

SCC.366: Media Coding & Processing

2022-2023

Week 07 – Lecture 1

Image Matching

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- What is Image matching?
- Applications of image matching
- Challenges in image matching
- Holistic vs. local image matching
- Holistic image matching an its limitations
  - Intensity based descriptor
  - Feature based descriptor e.g., HOG
- Local image matching
  - Interest point detection
  - Interest point description
  - Matching process
- Summary of the main points

Image Matching (Definition)

- "The process of bringing two images geometrically into agreement so that corresponding pixels in the two images correspond to the same physical region of the scene being imaged" (Dai & Lu, 1999)

- In short, image matching is the process of establishing a correspondence between a pair of images

Image Matching (Definition)

- Image matching is the process of establishing a correspondence between a pair of images

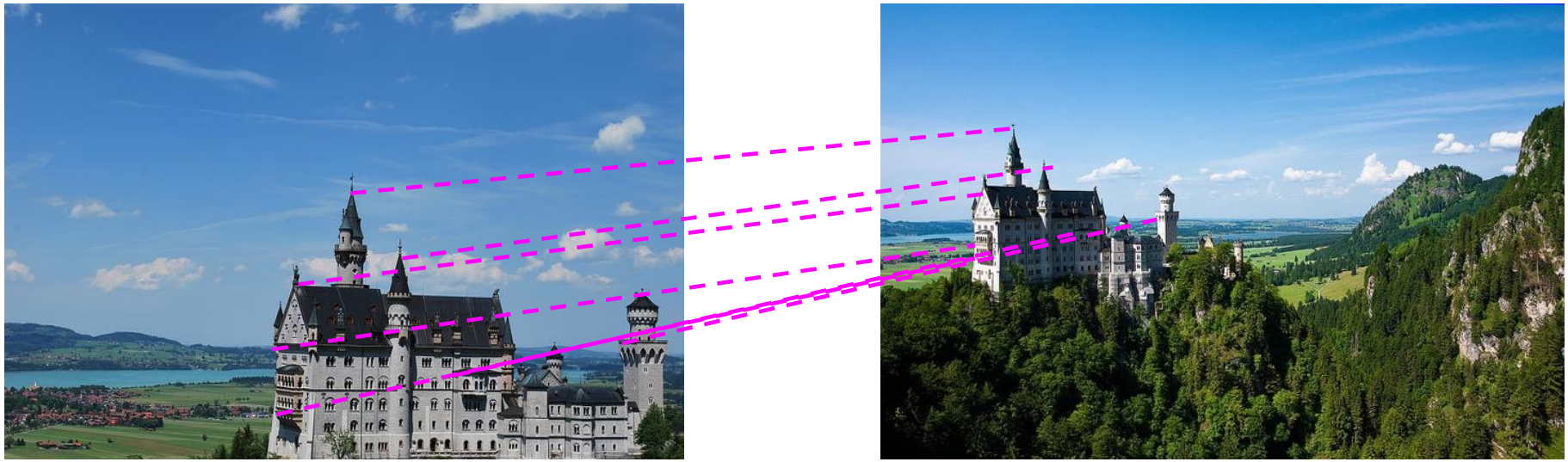
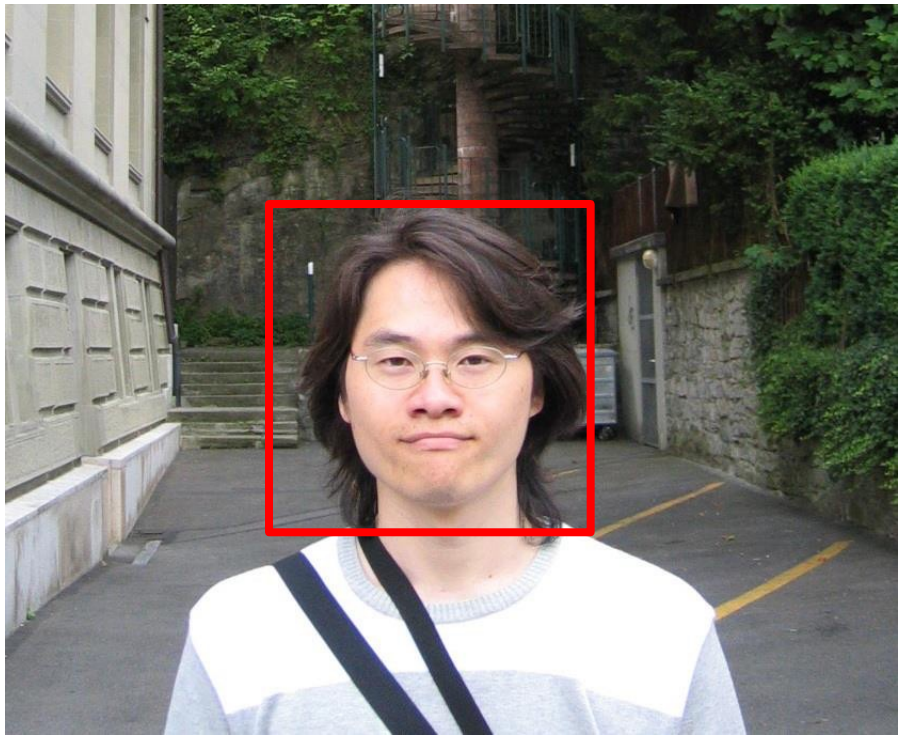


Image Matching Applications

- Face recognition
- Image retrieval
- Image stitching (panorama)
- Object tracking
- 3D Rreconstruction

Image Matching Applications

- Face recognition



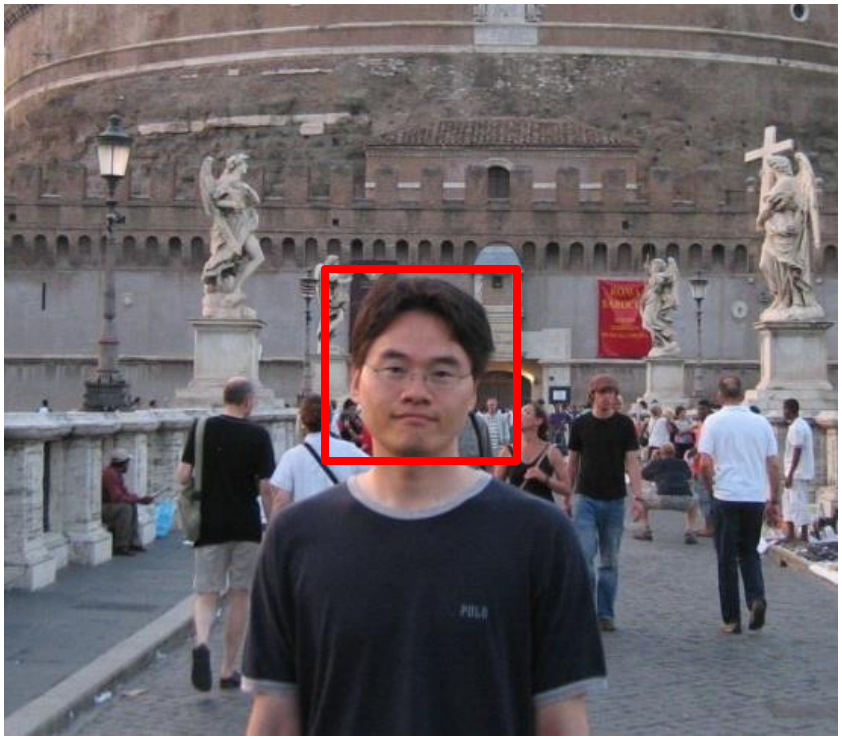


Image Matching Applications

- Image retrieval

Query image

Image database X

Image Matching Applications

- Image stitching for panorama

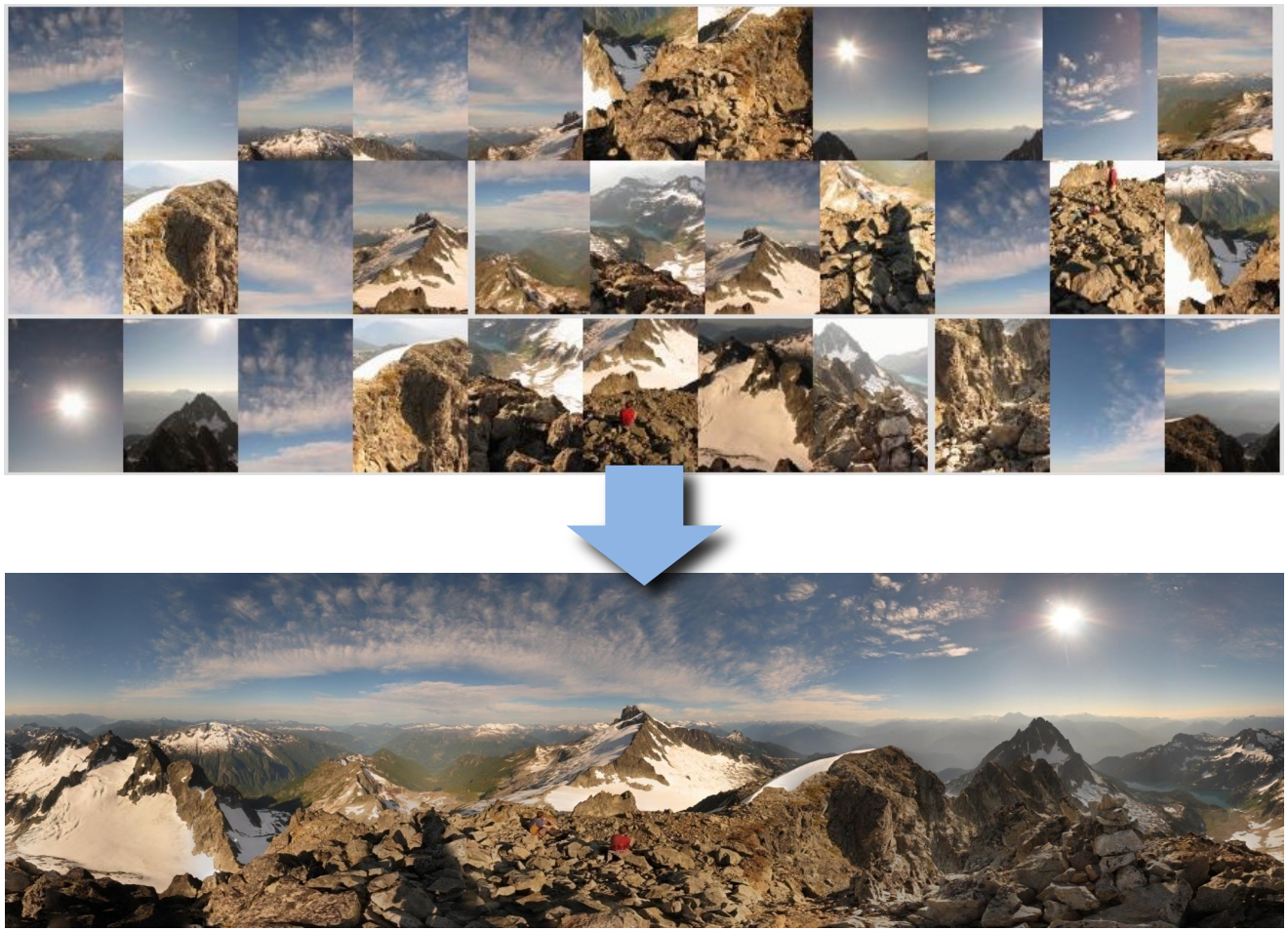


Image Matching Applications

- 3D reconstruction

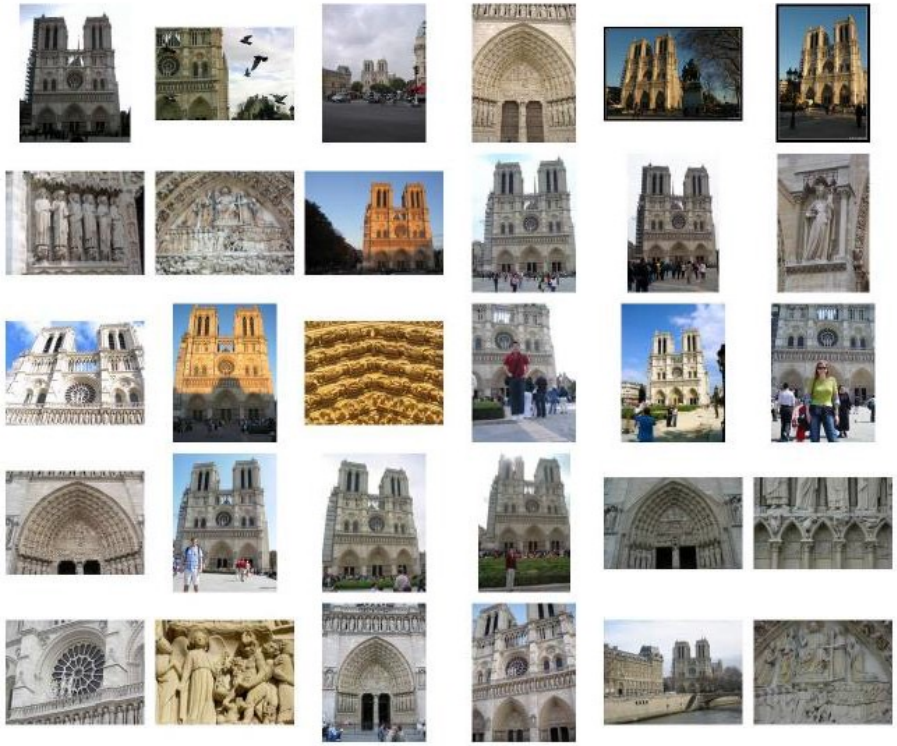


Image Matching Applications

- Object tracking

Challenges in Image Matching

- Illumination variation
- Scale variation
- Rotation and Perspective Variation

Challenges in Image Matching

- Illumination variation

Challenges in Image Matching

- Scale variation

Challenges in Image Matching

- Rotation and perspective variation

Challenges in Image Matching

- A hard example (NASA Mars Rover images)

Challenges in Image Matching

- A hard example (NASA Mars Rover images)

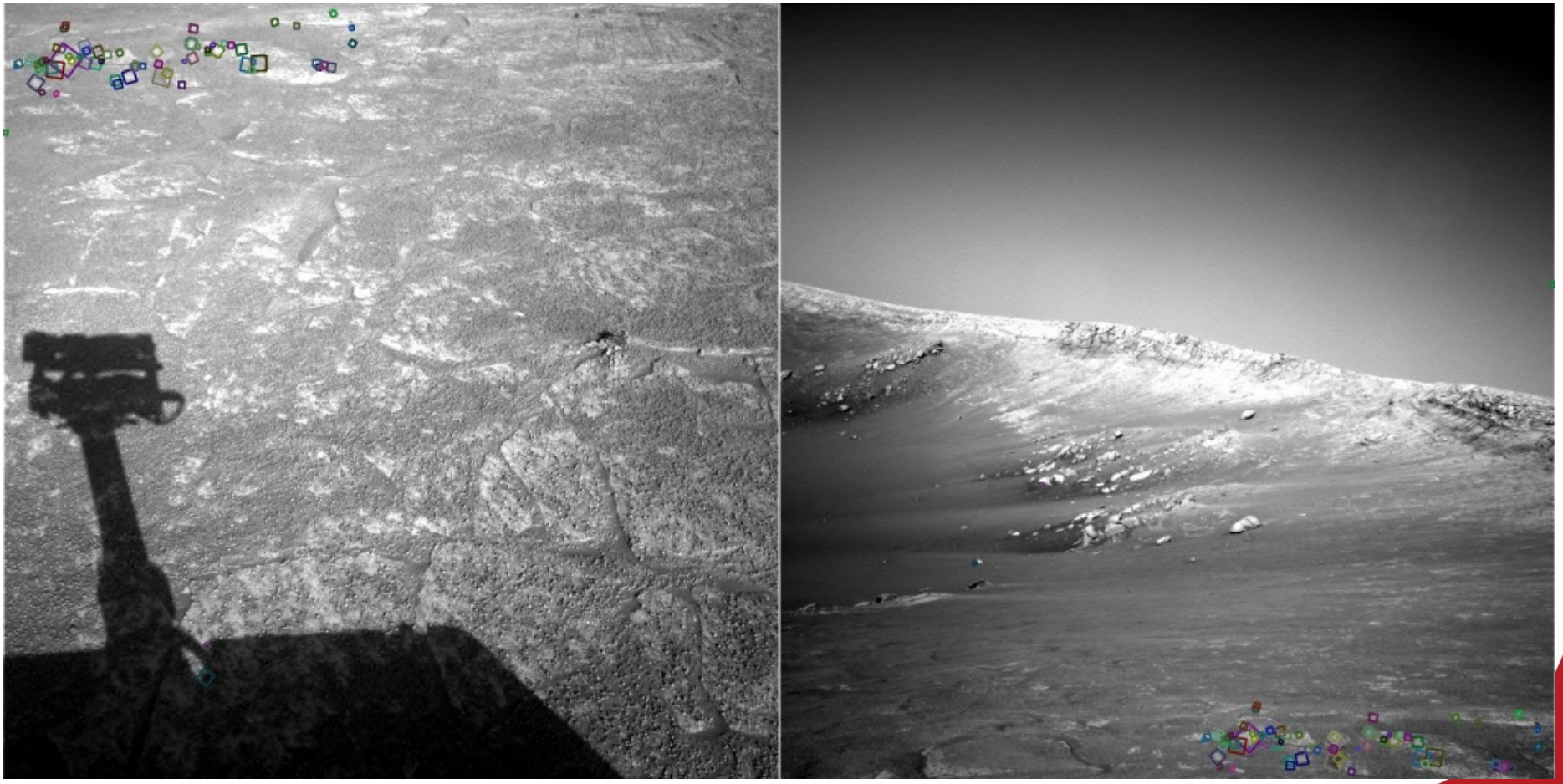


Image Matching (Holistic)

- The entire image is used to calculate similarity

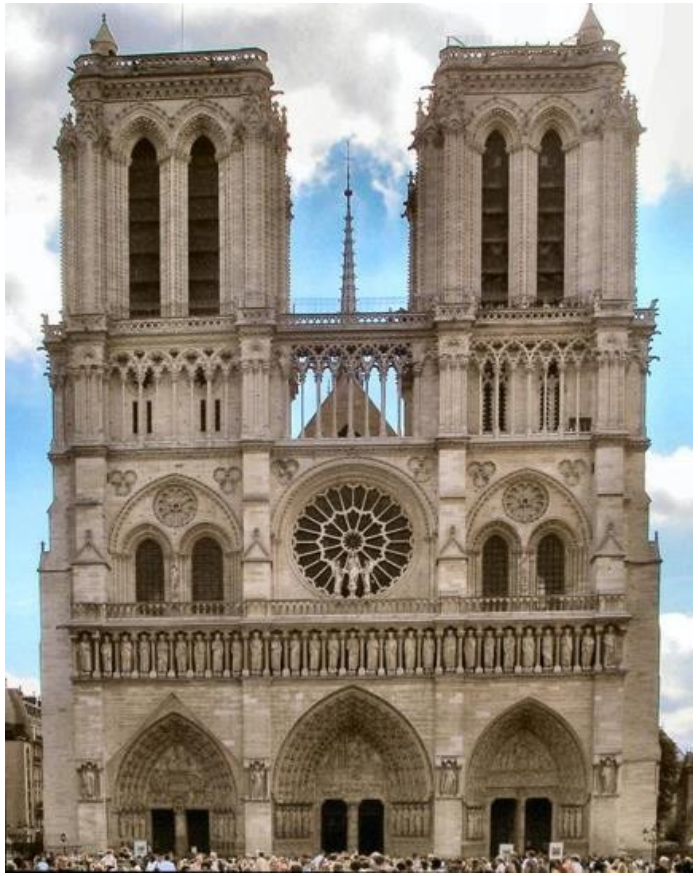
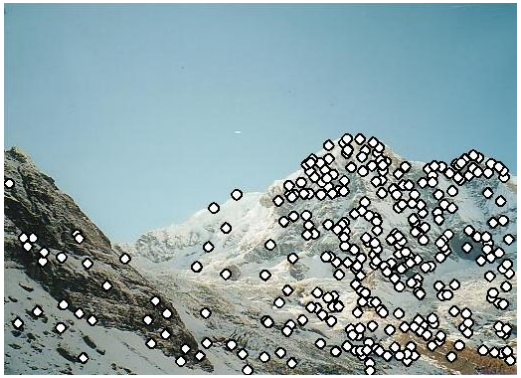


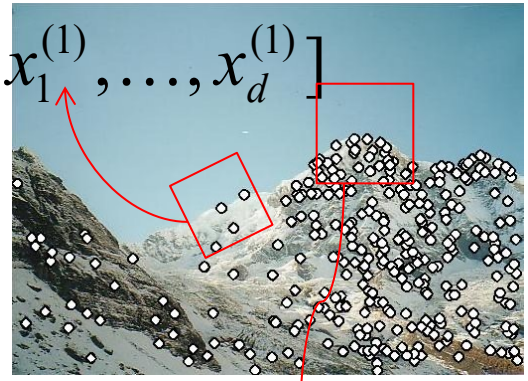
Image Matching (Local)

1. Interest Point Detection



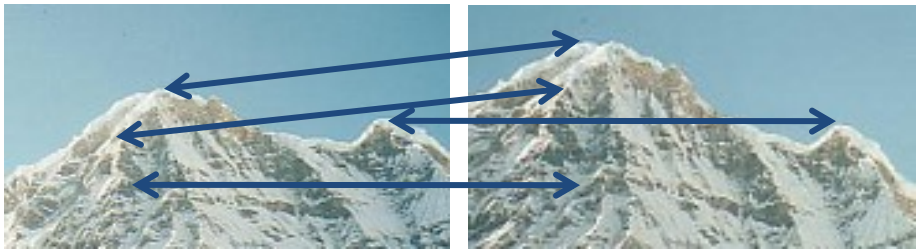
2. Local Feature Description

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



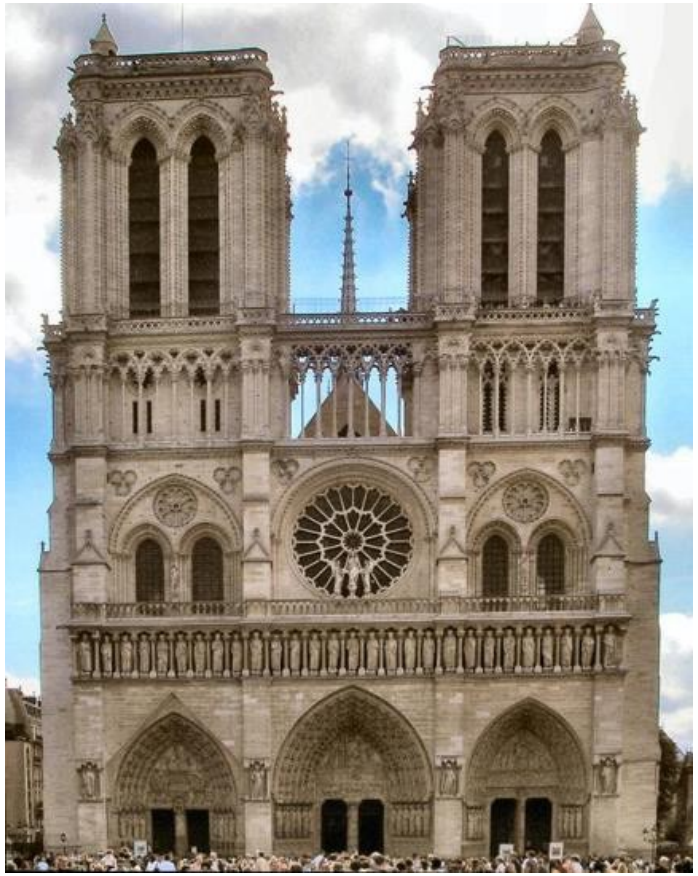
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3. Feature Matching



Holistic Image Matching

- Measures how the images overlap



Holistic Image Matching: Problem Setup

- To build a distance function *d* that receives a pair of images and outputs a value indicating **how different** the two input images are.
- Ideally, *d* should output 0 for matching image pairs and an infinite (or very high) value for non-matching image pairs.

d(,) = 0.5

d(,) =80

Holistic Image Matching: Problem Setup

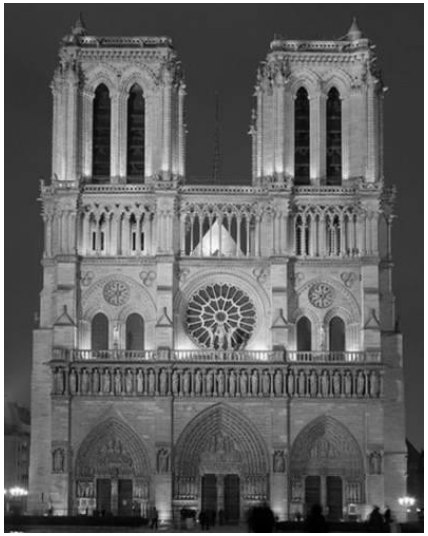
- For a given image pair A and B, check if d(A, B) is larger or smaller than a given threshold τ (e.g., τ =3).
  - If τ is small: some matching images will be labeled as *non-match*.
  - If τ is large: some non-matching images will be labeled as *match*.

$d($, $) = 0.5$

$d($, $) = 80$

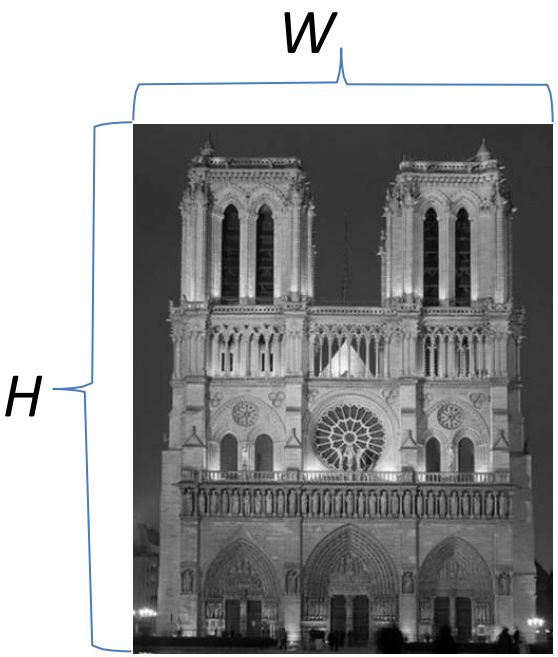
Holistic Image Matching: Problem Setup

- We will assume that all images are in gray-level.
- Color images are processed by either
  - Converting them to gray images, or
  - Measuring distance for each color channel (e.g., R, G, and B) and calculate the average.



Holistic Image Matching: Problem Setup

- A gray level bitmap image can be represented as
  - a two-dimensional matrix **A** (of size $H \times W$), or
  - a vector **a** (of size $n \times 1$; $n = H \times W$).



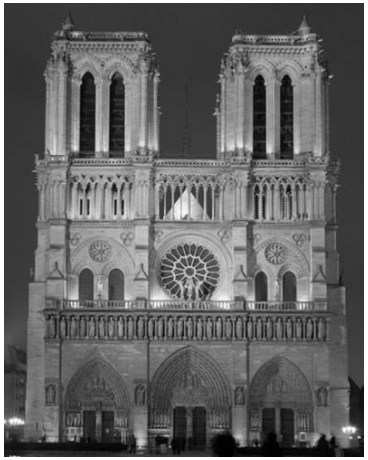
$$\mathbf{A} = \begin{pmatrix} a^{11} & \cdots & a^{1W} \\ \vdots & \ddots & \vdots \\ a^{H1} & \cdots & a^{HW} \end{pmatrix}$$

$$\mathbf{a} = \begin{pmatrix} a^{11} \\ \vdots \\ a^{H1} \\ \vdots \\ a^{1W} \\ \vdots \\ a^{HW} \end{pmatrix} = \begin{pmatrix} a^{1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ a^{n} \end{pmatrix}$$

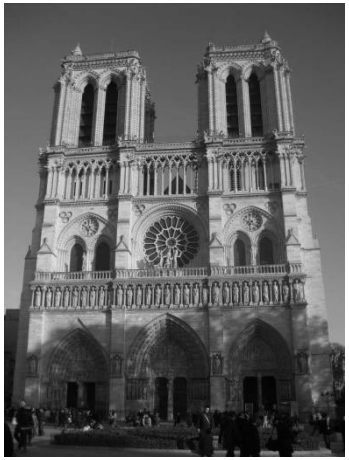
Sum of Squared Differences (SSD)

- SSD of two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is:
  - the sum of squared differences between the elements of $\boldsymbol{a}$ and $\boldsymbol{b}$

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} (a^i - b^i)^2$$



$$\mathbf{a} = \begin{pmatrix} a^1 \\ \vdots \\ a^n \end{pmatrix}$$



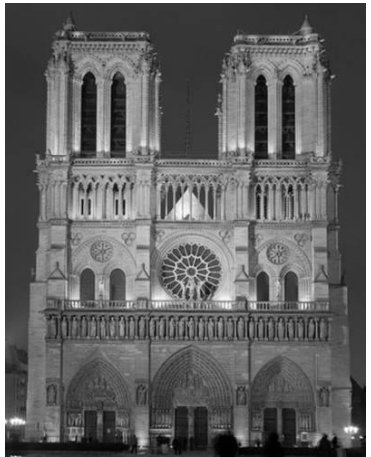
$$\mathbf{b} = \begin{pmatrix} b^1 \\ \vdots \\ b^n \end{pmatrix}$$

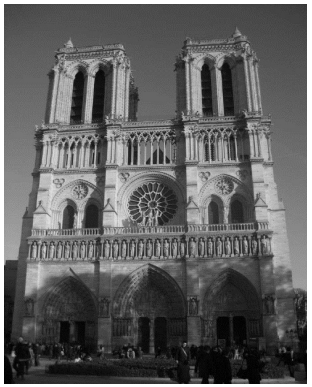
Sum of Squared Differences (SSD)

- SSD of two vectors **a** and **b** is:
  - the sum of squared differences between the elements of **a** and **b**

$$d(\boldsymbol{a}, \boldsymbol{b}) = \sum_{i=1}^{n} (a^i - b^i)^2$$

- It is applied only when **a** and **b** are the same size.
- If not, images can be resized before calculating the SSD.





Application: image retrieval

Query image **z**

Image database X

Image retrieval: problem formulation

- Given a database of images X = {$\mathbf{x}_1$, …, $\mathbf{x}_I$} and a query image **z**, find images in X that matches **z**.
- All images are the same size: $n \times 1$ ($n = H \times W$).

Query image **z**

Image database X = {$\mathbf{x}_1$, …, $\mathbf{x}_I$}

Image retrieval

- Problem
  - Given a database of images $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a query image $\mathbf{z}$, to find images in X that matches $\mathbf{z}$.
- Our strategy
  - Calculate the distance (SSD) between $\mathbf{z}$ and each element of X.
  - Pick a threshold τ.
  - Output all images in X whose distances from $\mathbf{z}$ is smaller than τ as retrieved results.

Image retrieval

Query image **z**

Image database X = {$\mathbf{x}_1$, …, $\mathbf{x}_l$}

$$\mathbf{z} = \begin{pmatrix} z^1 \\ \vdots \\ z^n \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \cdots & x_l^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_l^n \end{pmatrix}$$

$\mathbf{x}_1$ $\mathbf{x}_l$

Image retrieval

$$\mathbf{z} = \begin{pmatrix} z^1 \\ \vdots \\ z^n \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} x_1^1 & \cdots & x_l^1 \\ \vdots & \ddots & \vdots \\ x_1^n & \cdots & x_l^n \end{pmatrix}$$

$\mathbf{x}_1$ $\mathbf{x}_l$

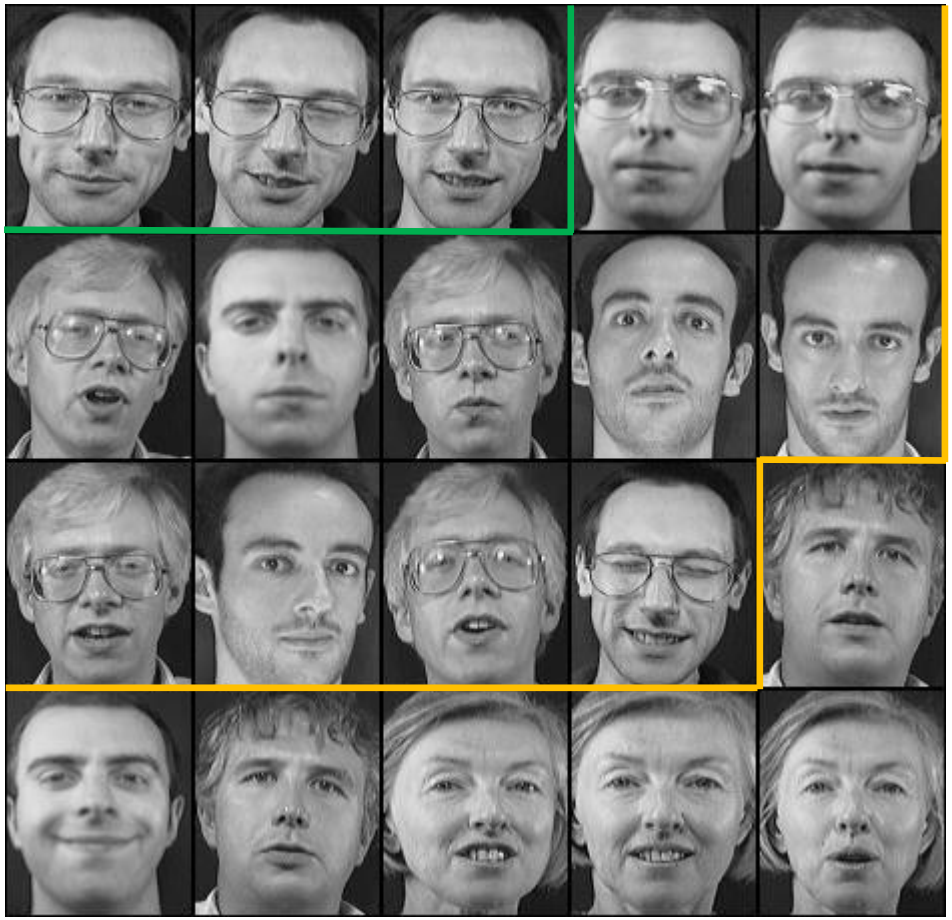
$$d(\boldsymbol{z}, \boldsymbol{x_1}) = \sum_{i=1}^{n} (z^i - x_1^i)^2$$

$$\vdots \qquad \vdots$$

$$d(\boldsymbol{z}, \boldsymbol{x_l}) = \sum_{i=1}^{n} (z^i - x_l^i)^2$$

Image retrieval

Query image **z**



Retrieved images arranged in increasing SSD from **z**.

SSD *d*:

6,459,606
10,275,183
11,195,272 τ
19,300,439
19,651,553
19,669,056
20,223,637
21,459,433
21,746,212
22,053,120
22,176,936
22,270,539
22,668,506
22,885,634 τ
23,276,550
23,282,331
23,387,770
24,104,343
24,214,001
24,244,896

Holistic image matching

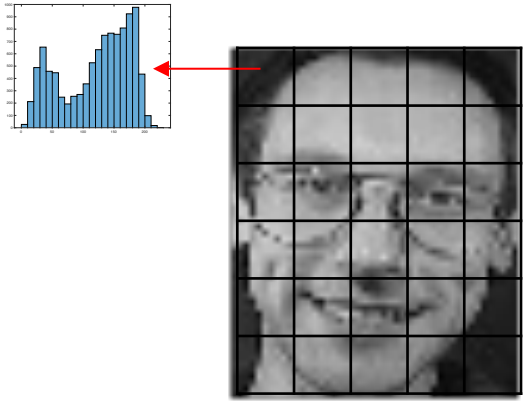
- Holistic matching does not always work!
- For example, if the image is shifted a bit

Query image **z**

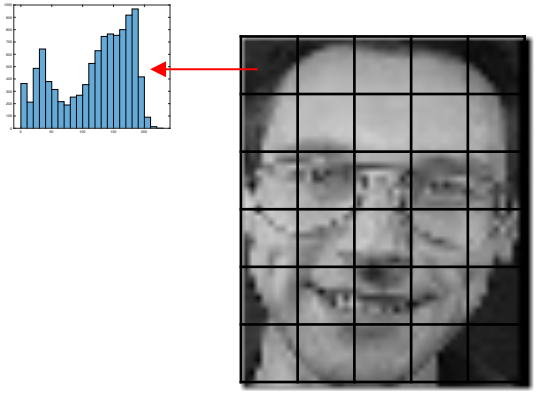
Query image **z**
Shifted by 5 pixels

Local Image Histogram

- One approach to mitigate this issue is:
  - Dividing image into small partitions
  - Calculating histogram of intensity or color



Query image **z**



Query image **z**

Shifted by 5 pixels

Holistic image matching

- Holistic matching does not always work!
- For example, if the illumination is changed

Query image **z**

Query image **z**

Illumination changed by -50

Query image **z**

Illumination changed by +50

HOG: Histogram of Oriented Gradient

- **A feature descriptor** is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information.
- **Histogram of oriented gradients (HOG)** is a feature descriptor used to detect objects in computer vision and image processing.

Histogram of Oriented Gradient (Steps)

1. Calculate magnitude and direction of gradient at each pixel in the image
2. Divide the image into 8×8 cells
3. Calculate histogram of gradients in each cell
4. Block Normalization
5. Form HOG feature vector

Steps to calculate HOG (Step 1)

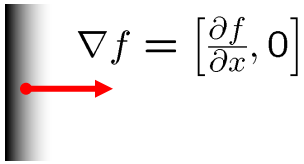
1. **Calculate magnitude and direction of gradient at each pixel in the image**
2. Divide the image into 8×8 cells
3. Calculate histogram of gradients in each cell
4. Block Normalization
5. Form HOG feature vector

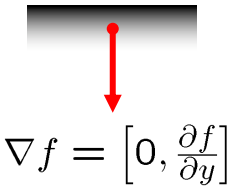
Review: Image Gradient

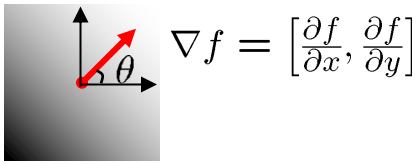
- The *gradient* of an image at each pixel is a vector:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- The gradient points in the direction of most rapid increase in intensity







Review: Image Gradient

- How to calculate $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$?

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- We can use any edge detector e.g., Sobel

$\frac{\partial f}{\partial x}$

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

$\frac{\partial f}{\partial y}$

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

Review: Image Gradient

- The edge strength is given by the gradient magnitude:

$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

OR

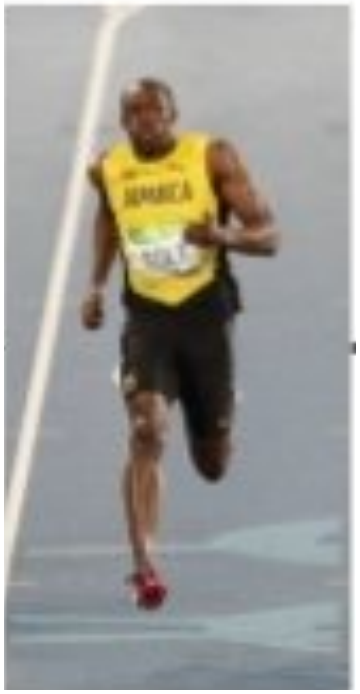
$$||\nabla f|| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

More efficient approximation of the gradient magnitude

- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

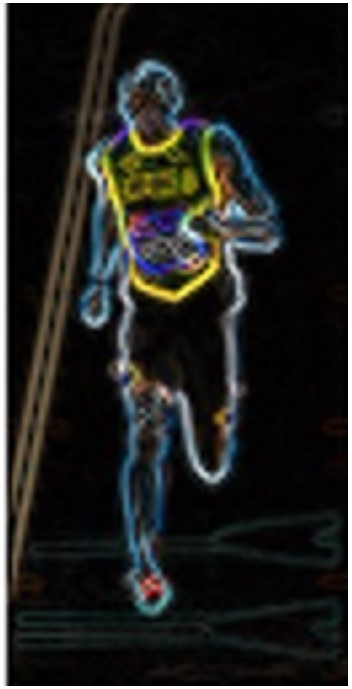
Steps to calculate HOG (Step 1)



Image

$\frac{\partial f}{\partial x}$

$\frac{\partial f}{\partial y}$



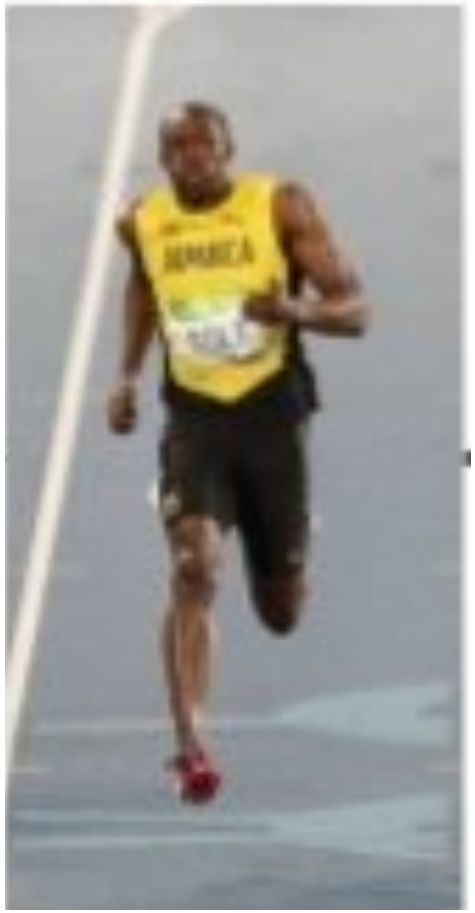
$||\nabla f||$

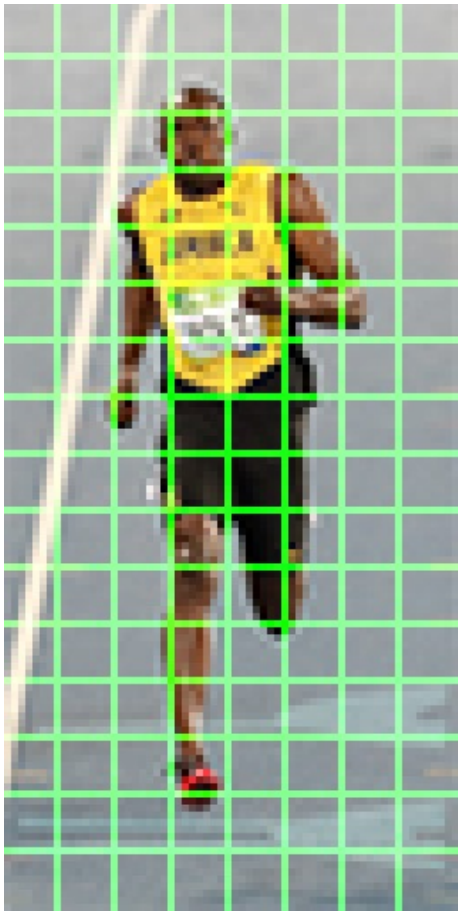
Steps to calculate HOG (Step 2)

1. Calculate magnitude and direction of gradient at each pixel in the image
2. **Divide the image into 8×8 cells**
3. Calculate histogram of gradients in each cell
4. Block Normalization
5. Form HOG feature vector

Steps to calculate HOG (Step 2)

- The image is divided into 8*8 cells



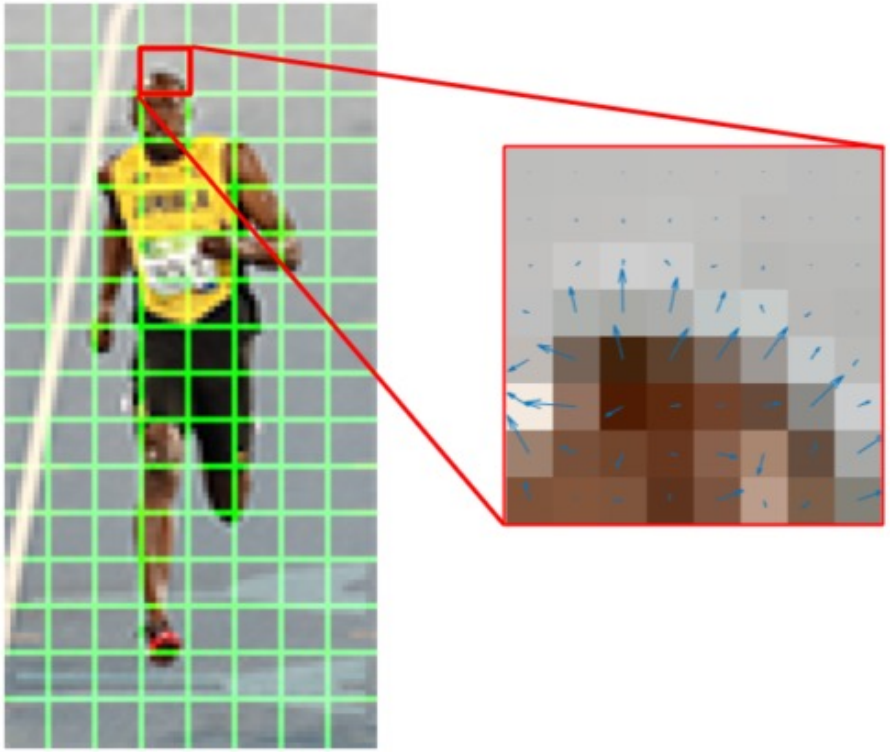


Steps to calculate HOG (Step 3)

1. Calculate magnitude and direction of gradient at each pixel in the image
2. Divide the image into 8×8 cells
3. **Calculate histogram of gradients in each cell**
4. Block Normalization
5. Form HOG feature vector

Steps to calculate HOG (Step 3)

- The gradient of an 8×8 patch contains 2 values (magnitude and direction) per pixel which adds up to 8x8x2 = 128 numbers



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

$||\nabla f||$

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

θ

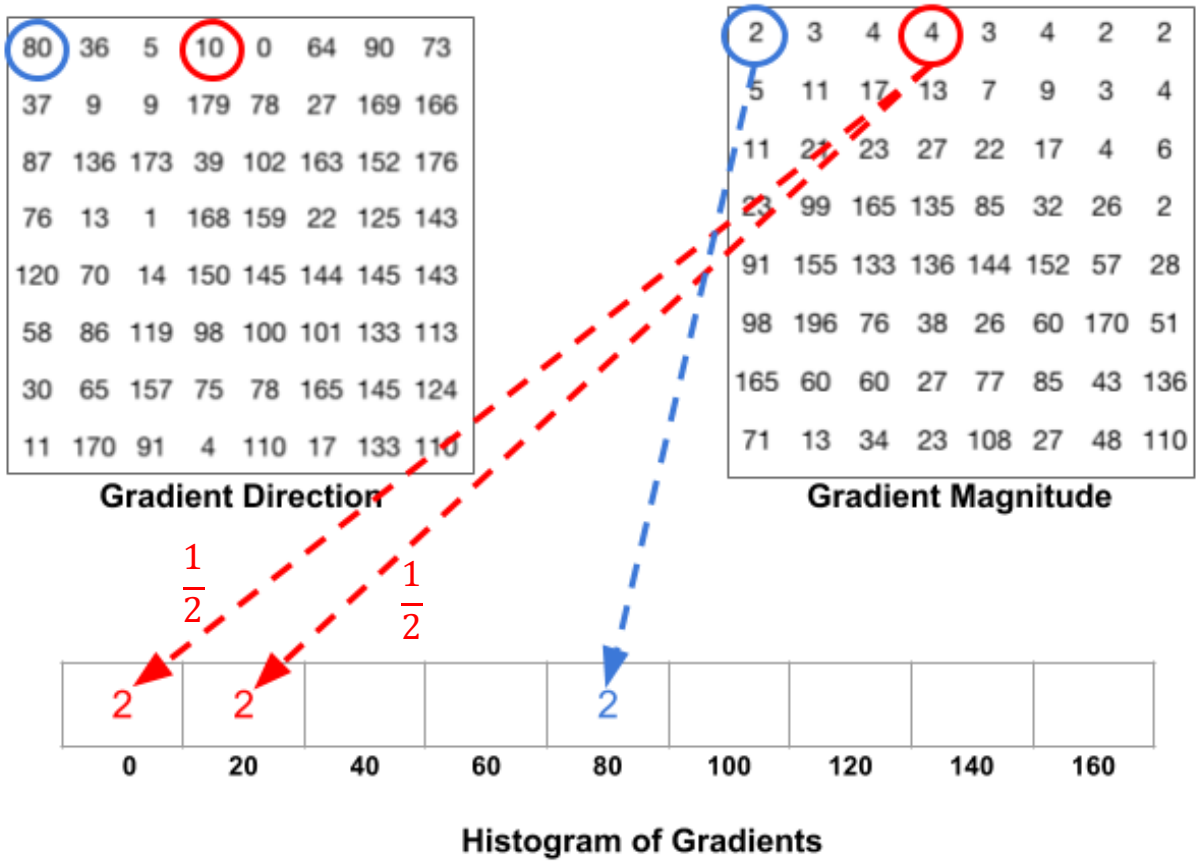
Steps to calculate HOG (Step 3)

- For each cell, a 9-point histogram is calculated.
- A 9-point histogram develops a histogram with 9 bins and each bin has an angle range of 20 degrees.

Value									
Bins	0	20	40	60	80	100	120	140	160

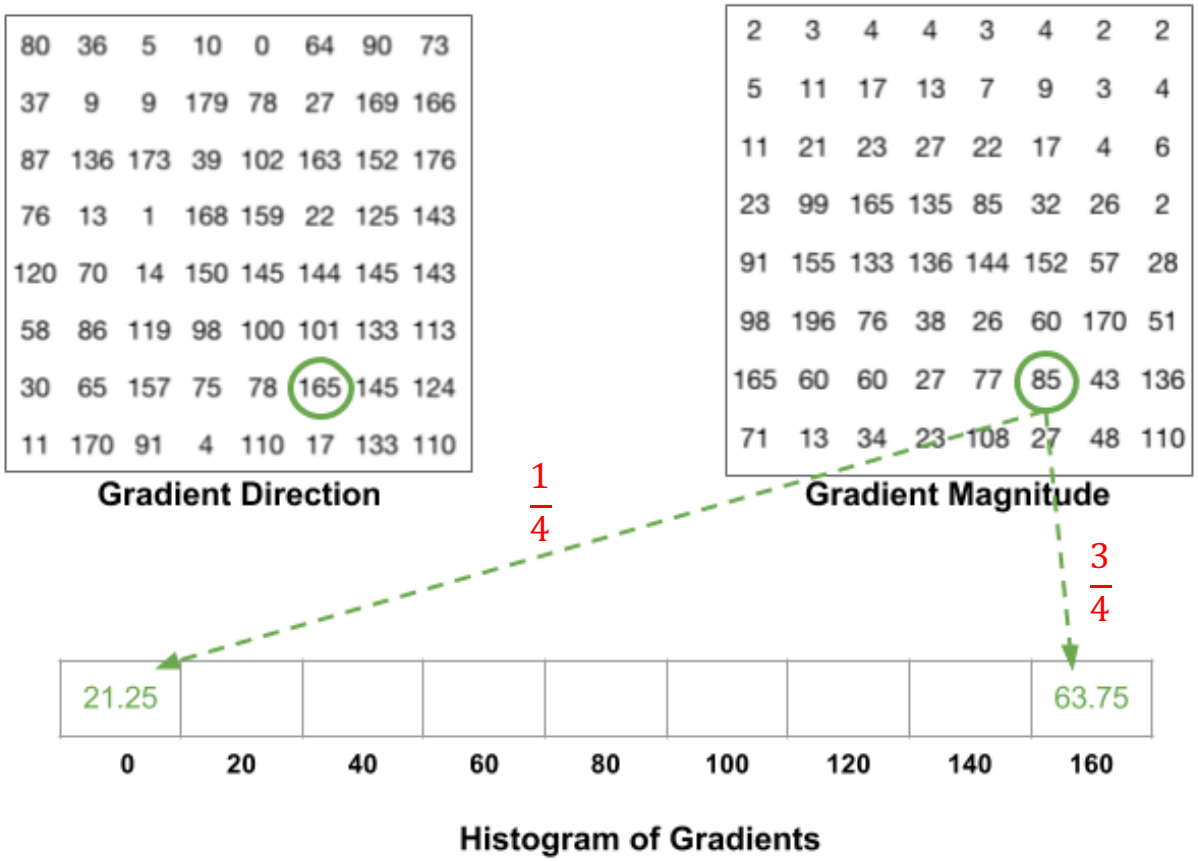
Steps to calculate HOG (Step 3)

- A histogram of gradient orientation is created by quantization



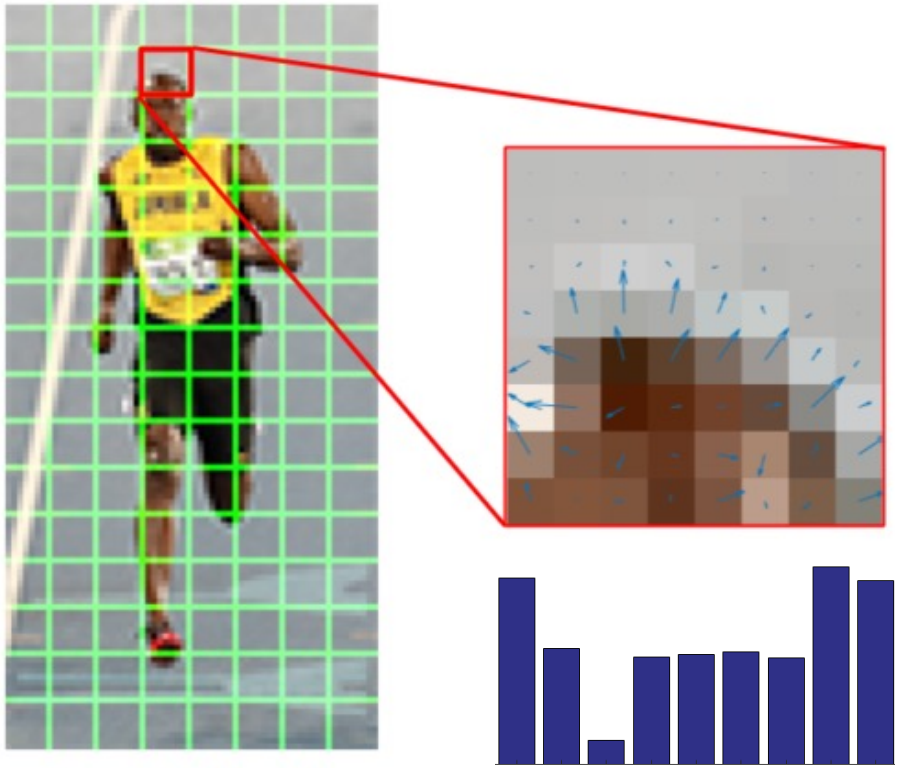
Steps to calculate HOG (Step 3)

- A histogram of gradient orientation is created by quantization



Steps to calculate HOG (Step 3)

- The histogram of a cell is as below



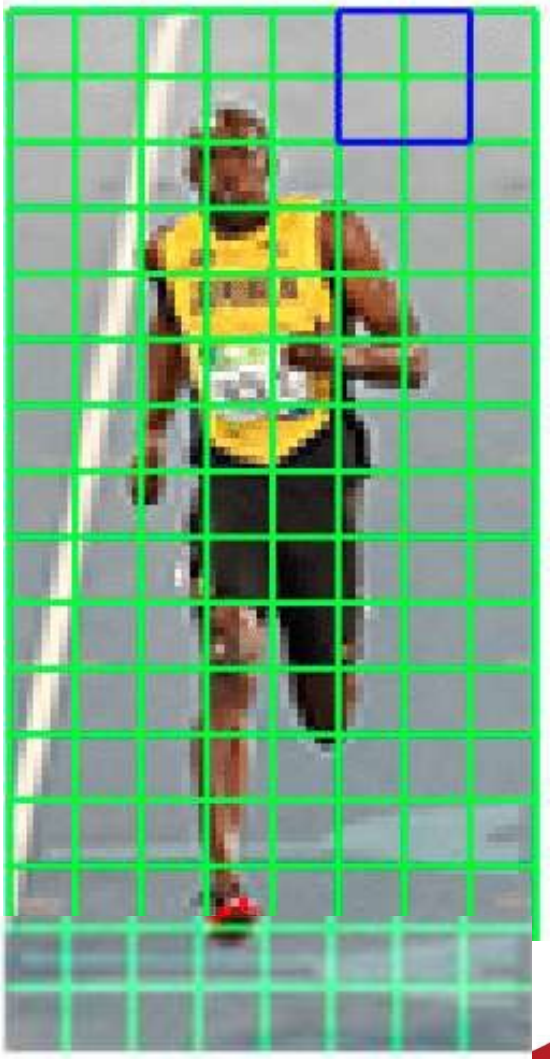
9-bin histogram

Steps to calculate HOG (Step 4)

1. Calculate magnitude and direction of gradient at each pixel in the image
2. Divide the image into 8×8 cells
3. Calculate histogram of gradients in each cell
4. **Block Normalization**
5. Form HOG feature vector

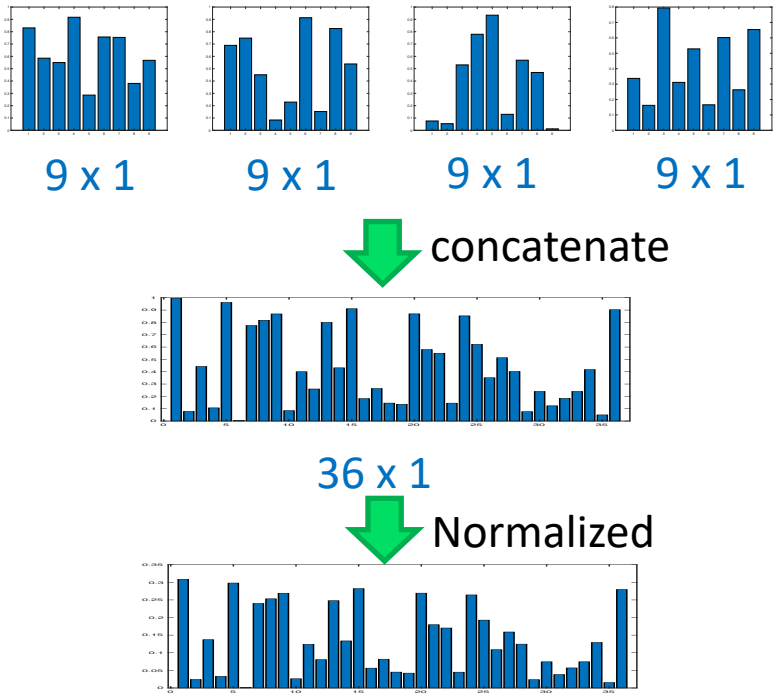
Steps to calculate HOG (Step 4)

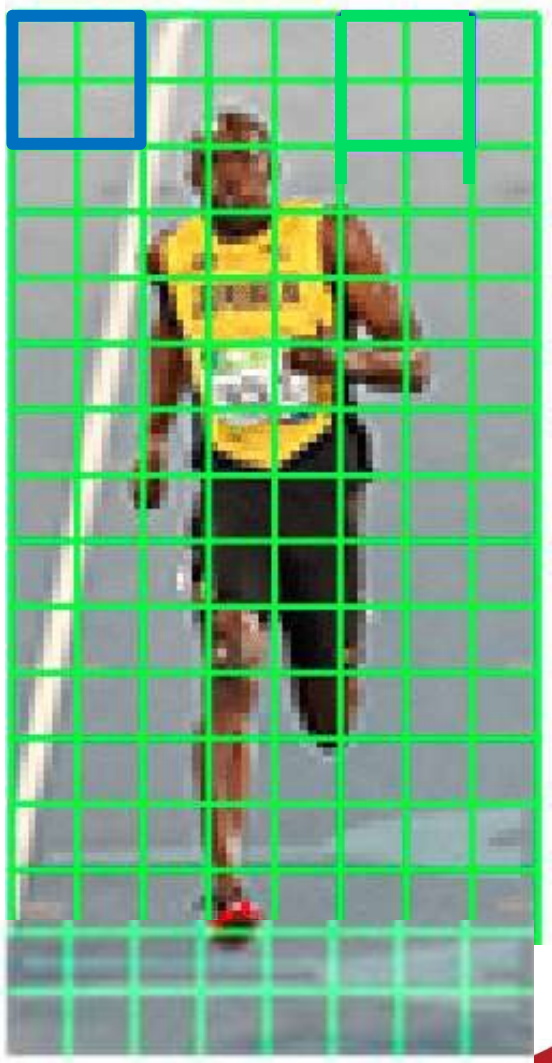
- 4 cells are merged to form a block (2x2 cells). This merging process is done in an overlapping manner with a stride of one cell.
- A 16×16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector
- Normalize the histogram so they are not affected by lighting variations



Steps to calculate HOG (Step 4)

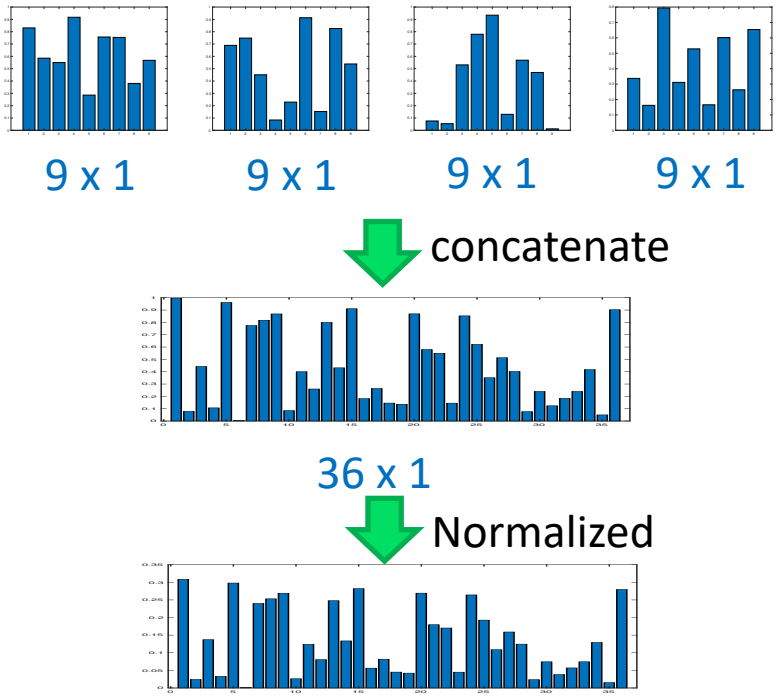
- This merging process is done in an overlapping manner with a stride of one cell.

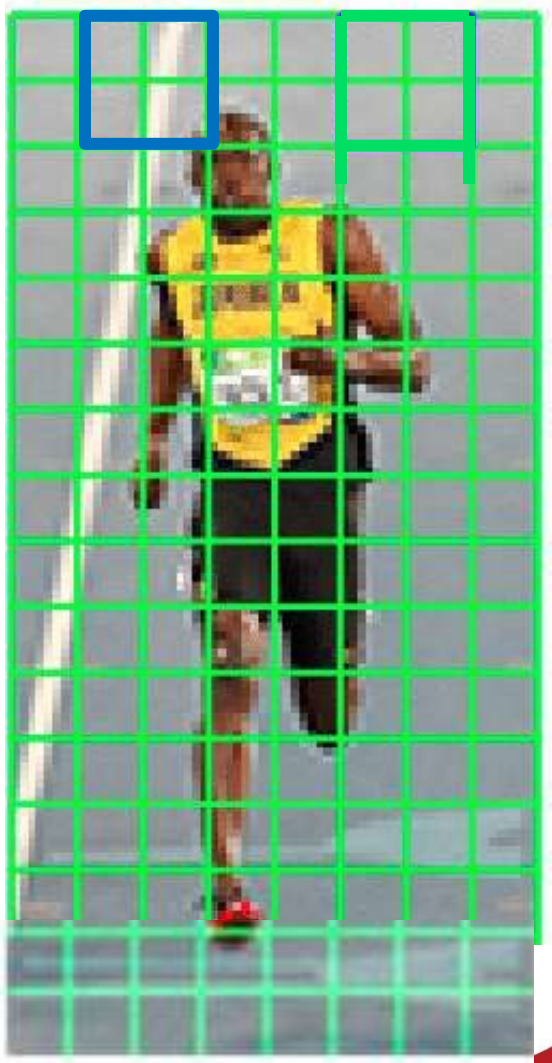




Steps to calculate HOG (Step 4)

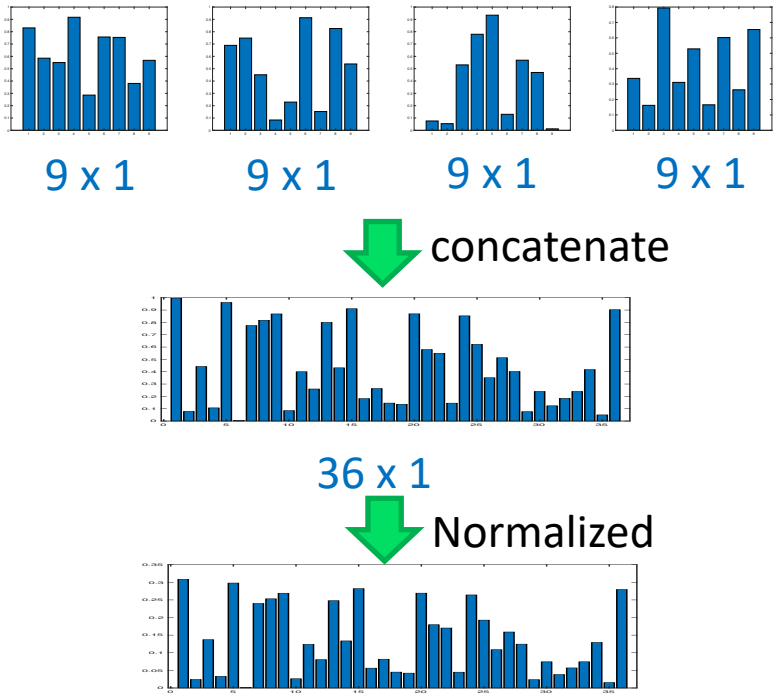
- This merging process is done in an overlapping manner with a stride of one cell.

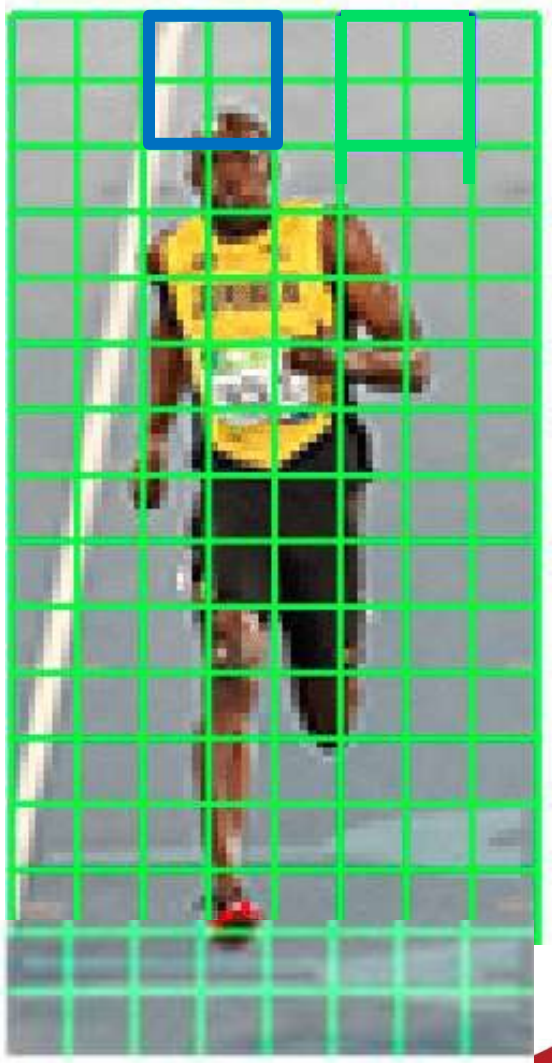




Steps to calculate HOG (Step 4)

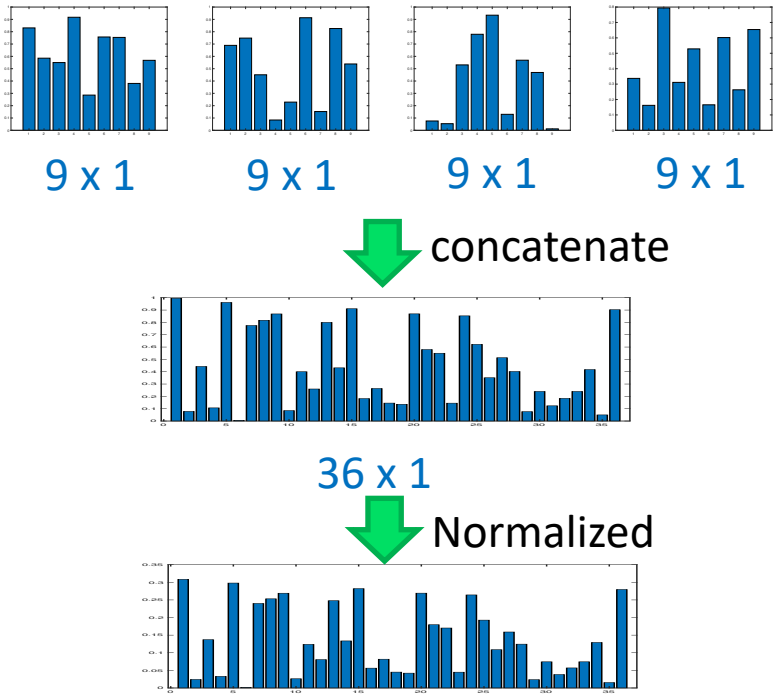
- This merging process is done in an overlapping manner with a stride of one cell.

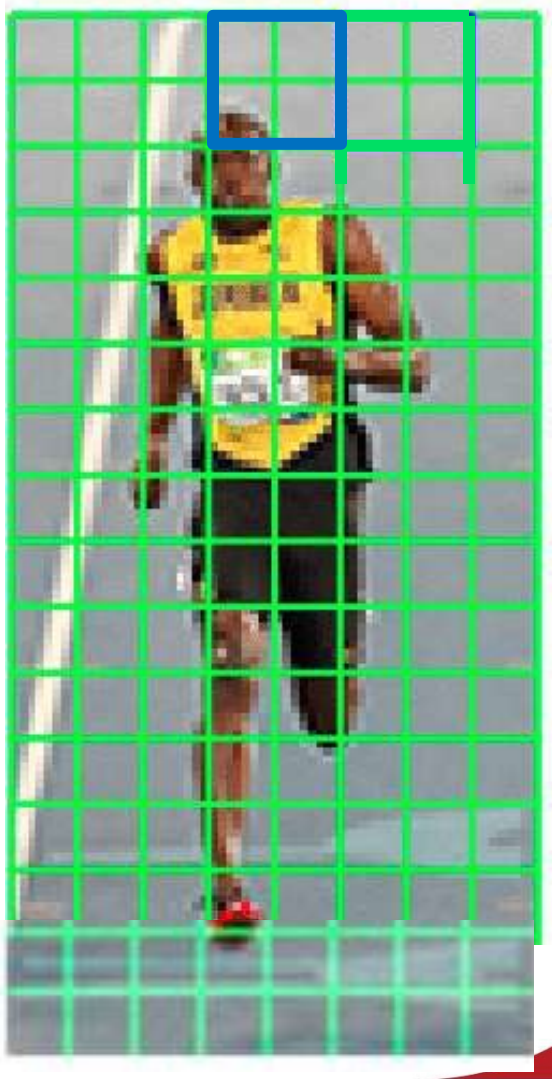




Steps to calculate HOG (Step 4)

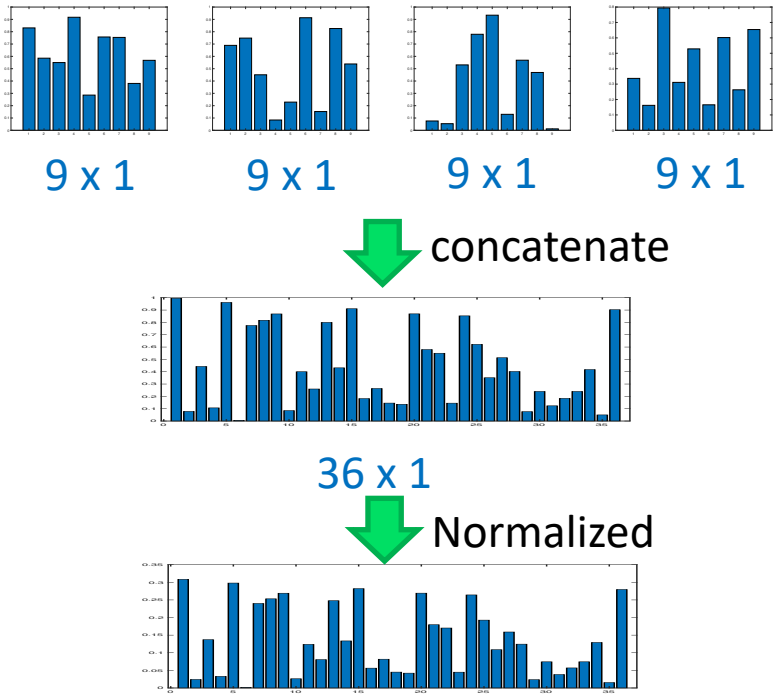
- This merging process is done in an overlapping manner with a stride of one cell.

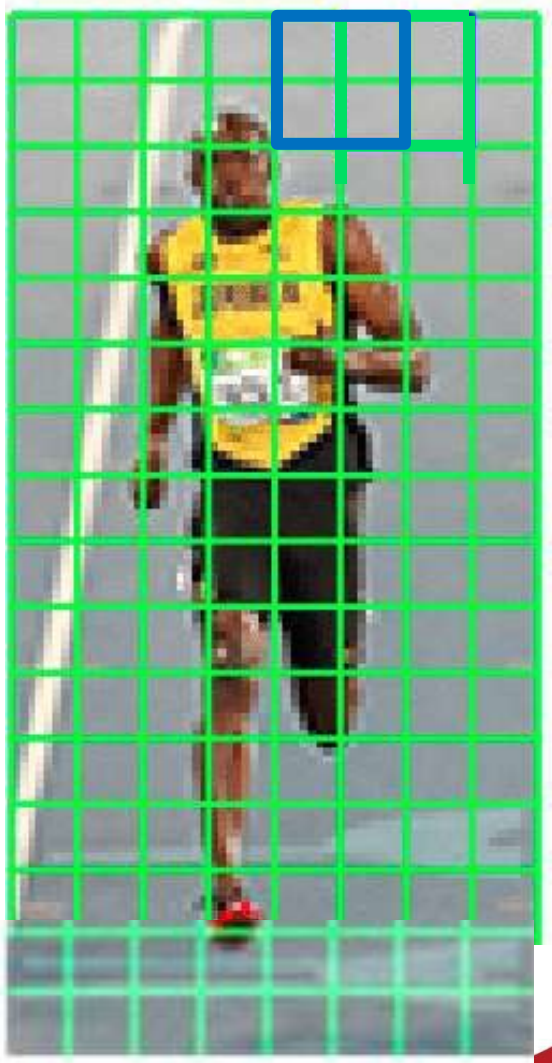




Steps to calculate HOG (Step 4)

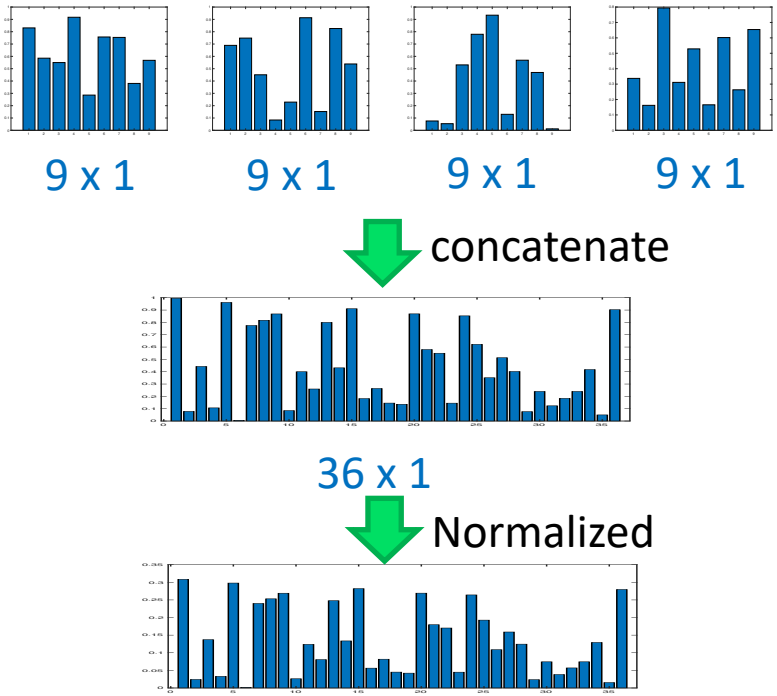
- This merging process is done in an overlapping manner with a stride of one cell.

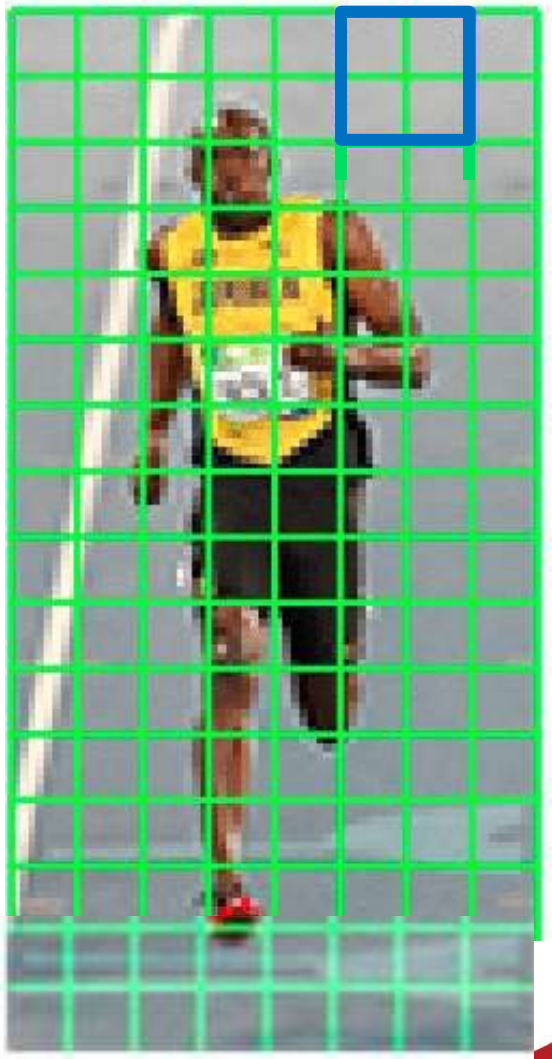




Steps to calculate HOG (Step 4)

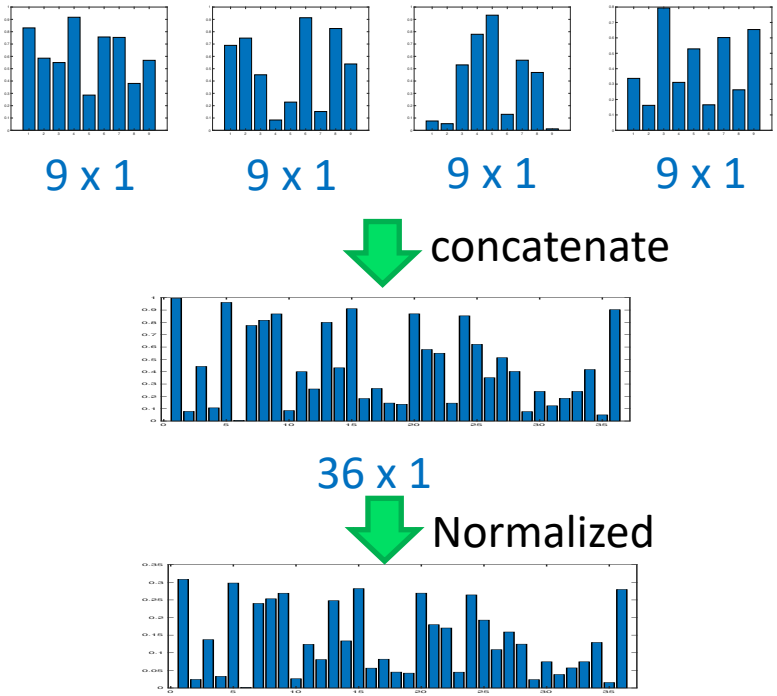
- This merging process is done in an overlapping manner with a stride of one cell.

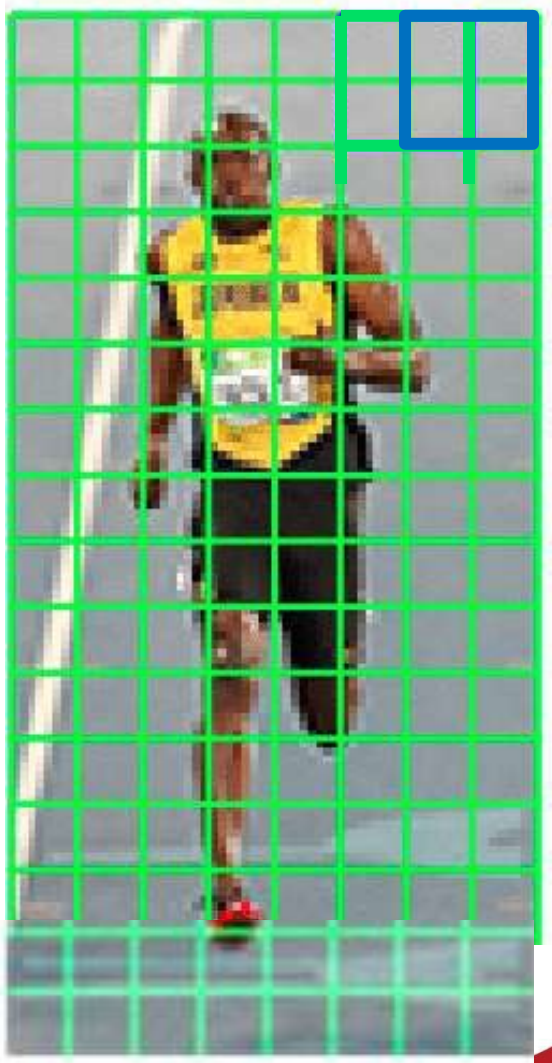




Steps to calculate HOG (Step 4)

- This merging process is done in an overlapping manner with a stride of one cell.





Steps to calculate HOG (Step 5)

1. Calculate magnitude and direction of gradient at each pixel in the image
2. Divide the image into 8×8 cells
3. Calculate histogram of gradients in each cell
4. Block Normalization
5. **Form HOG feature vector**

Steps to calculate HOG (Step 5)

- To calculate the final feature vector for the entire image, the 36×1 vectors are concatenated into one giant vector,
  - e.g., 36×105 = 3780-dimensional vector for this example.

Calculate HOG for Color Images

- The description of HOG we discussed so far is for gray-level images.

- Color images can be processed in two different ways:

  - Converting them to grey-level image, or

  - The gradient can be computed in each color band, and the gradient with greatest magnitude at each pixel can be used to compute the HOG feature

Holistic image matching

- Holistic matching does not always work!

Holistic image matching

- Holistic matching does not always work!

SCC.366: Media Coding & Processing

2022-2023

Week 07 – Lecture 2

Local Image Matching

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- What is Image matching?
- Applications of image matching
- Challenges in image matching
- Holistic vs. local image matching
- Holistic image matching an its limitations
  - Intensity based descriptor
  - Feature based descriptor e.g., HOG
- Local image matching (This Lecture)
  - Interest point detection
  - Interest point description
  - Matching process
- Summary of the main points

Questionnaire – Measuring Temperature Adaptability

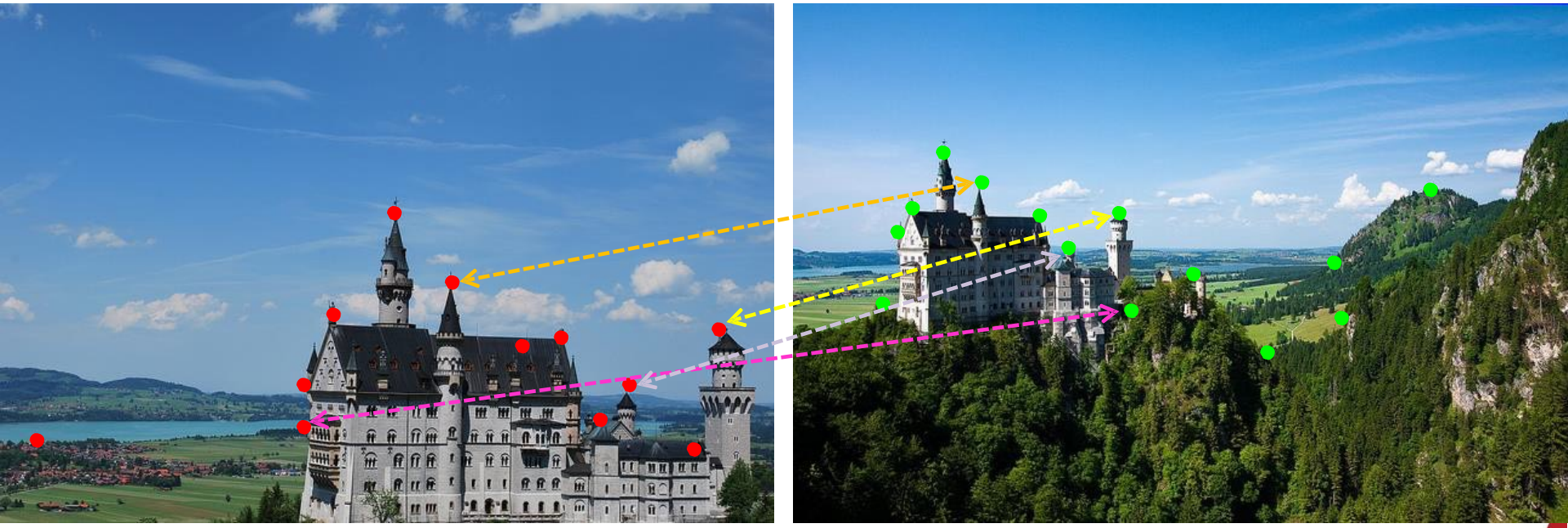
- **Jasper Clively TYP - General Questionnaire.**

  What makes temperature control biased based on Gender, Clothing and Activity Level.

- https://forms.office.com/r/sNkrWPWC7Y

Local Image Matching (Definition)

- Image matching is the process of establishing a correspondence between a pair of images

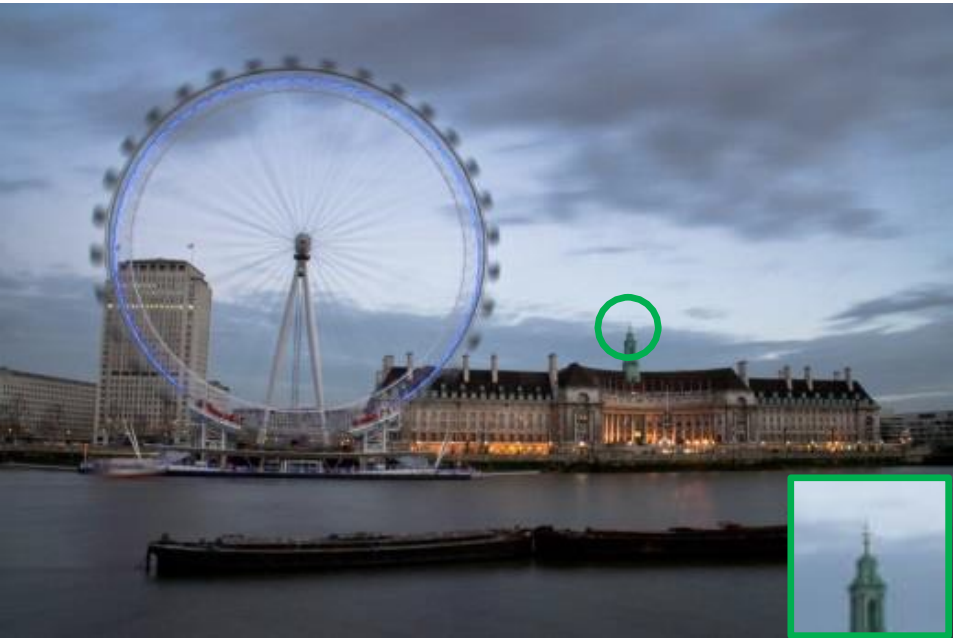


Global Image Matching

- Global/Holistic features cannot deal with occlusions, clutter, viewpoint changes

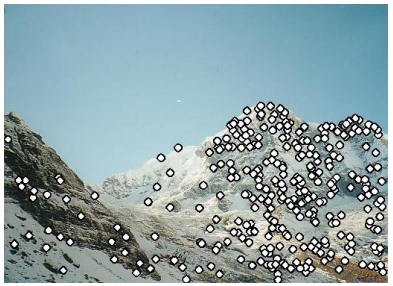
Local Image Matching

- Advantages of local features:
  - Can deal with occlusions
  - Can deal with clutter
    - (a lot of objects in the image and thus it is difficult for an observer to focus their mind on any particular object)
  - More invariant to image transformations
  - More robust to noise
  - Sparse representation of the image



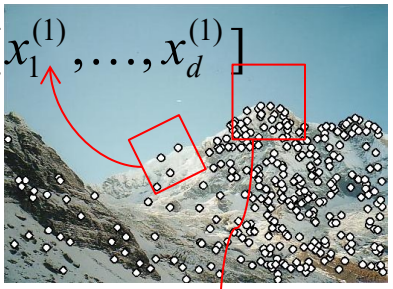
Local Image Matching (main components)

1) **Detection**: Identify the interest points, e.g., Harris corner and SIFT



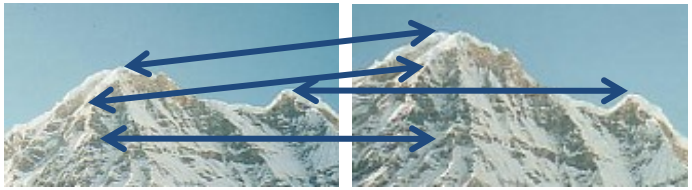
2) **Description**: Extract feature descriptor surrounding each interest point, e.g., HOG, SIFT

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching**: Determine correspondence between descriptors in two views

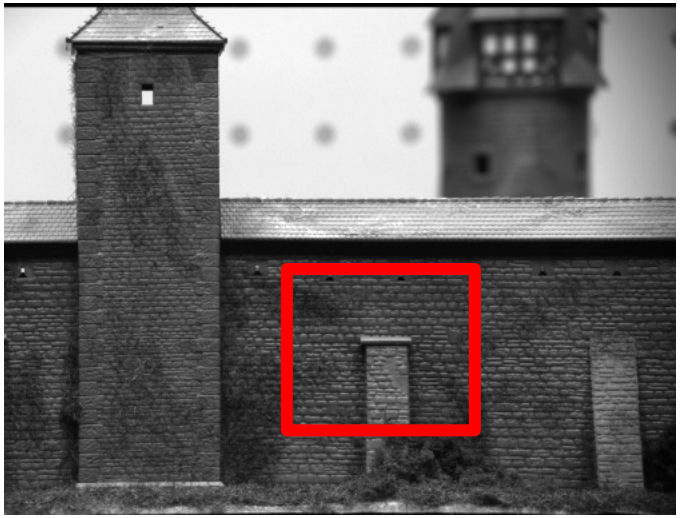


A Good Local Feature

- Accurate and repeatable localization of the feature points
- Invariance to translation, rotation, scale, viewpoint
- Robustness to noise, lighting conditions, compression and blur
- Distinctiveness of descriptor

Motivation: Patch Matching

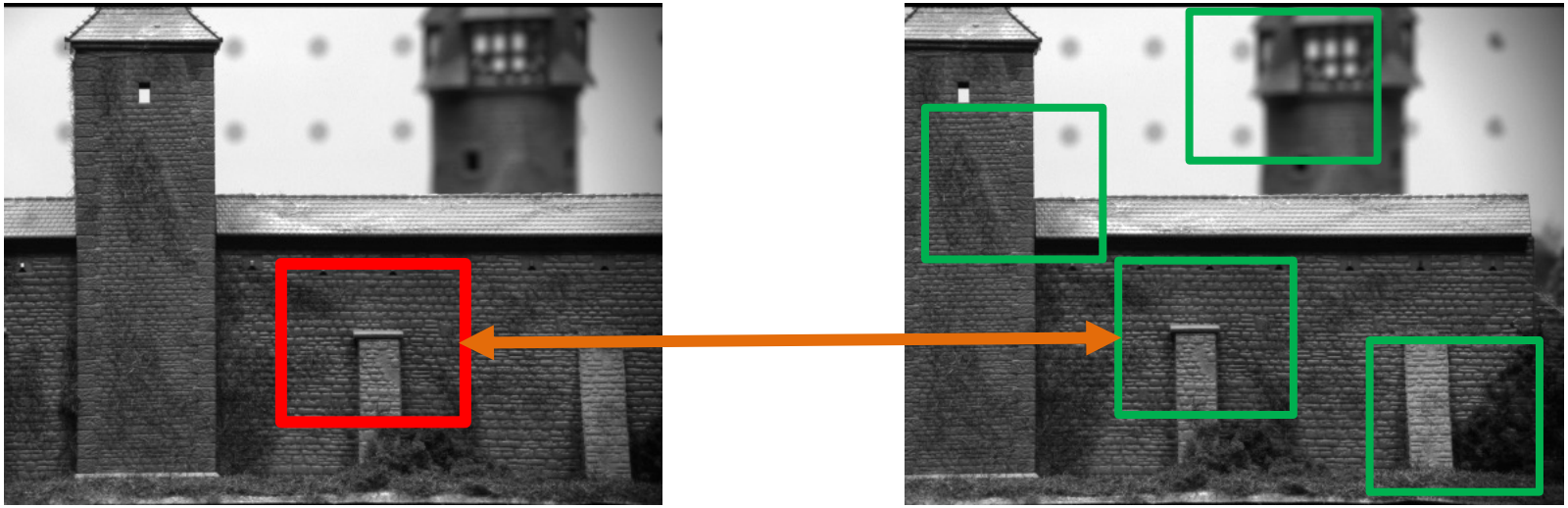
- Elements to be matched are image patches of fixed size



- **Question:** What is the best (most similar) patch in the second image?

A Good Local Feature

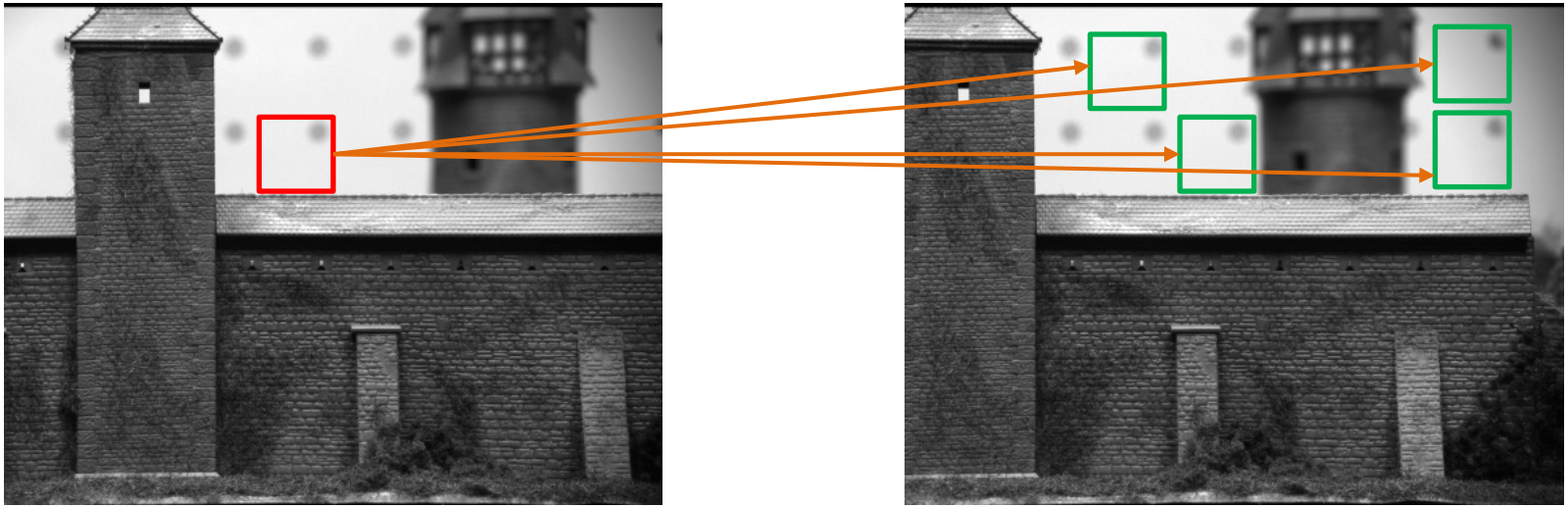
- **Question:** What is the best (most similar) patch in the second image?



- Intuition: this would be a good patch for matching, since it is very distinctive (there is only one patch in the second image that looks similar).

Not all Patches are Distinctive

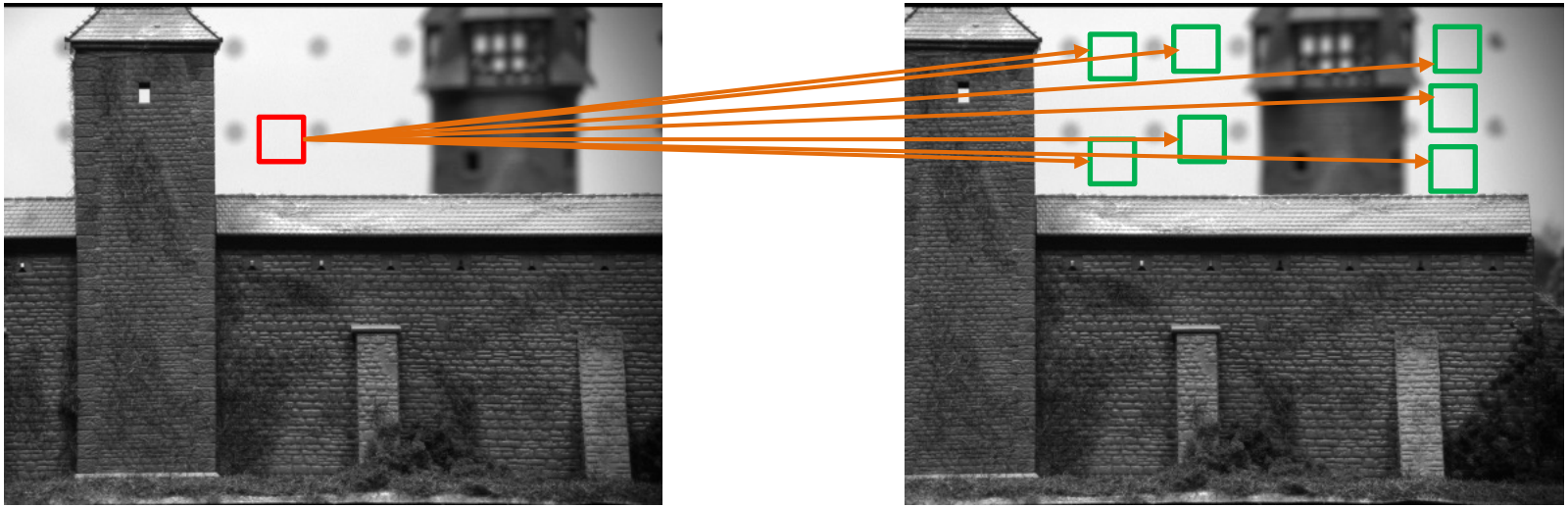
- **Question:** What is the best (most similar) patch in the second image?



Intuition: this would be a BAD patch for matching, since it is not very distinctive (there are a few similar patches in the second image).

A very Bad Local Feature

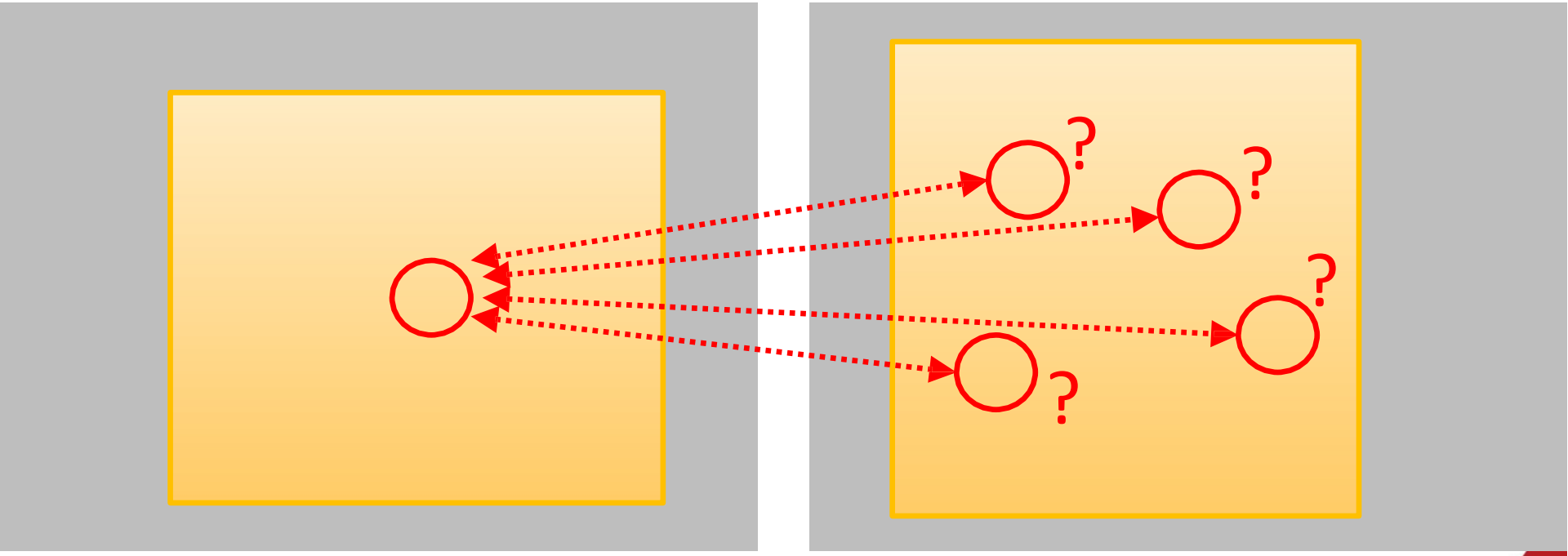
- **Question:** What is the best (most similar) patch in the second image?



Intuition: this would be a very BAD patch for matching, since there are many similar patches in the second image.

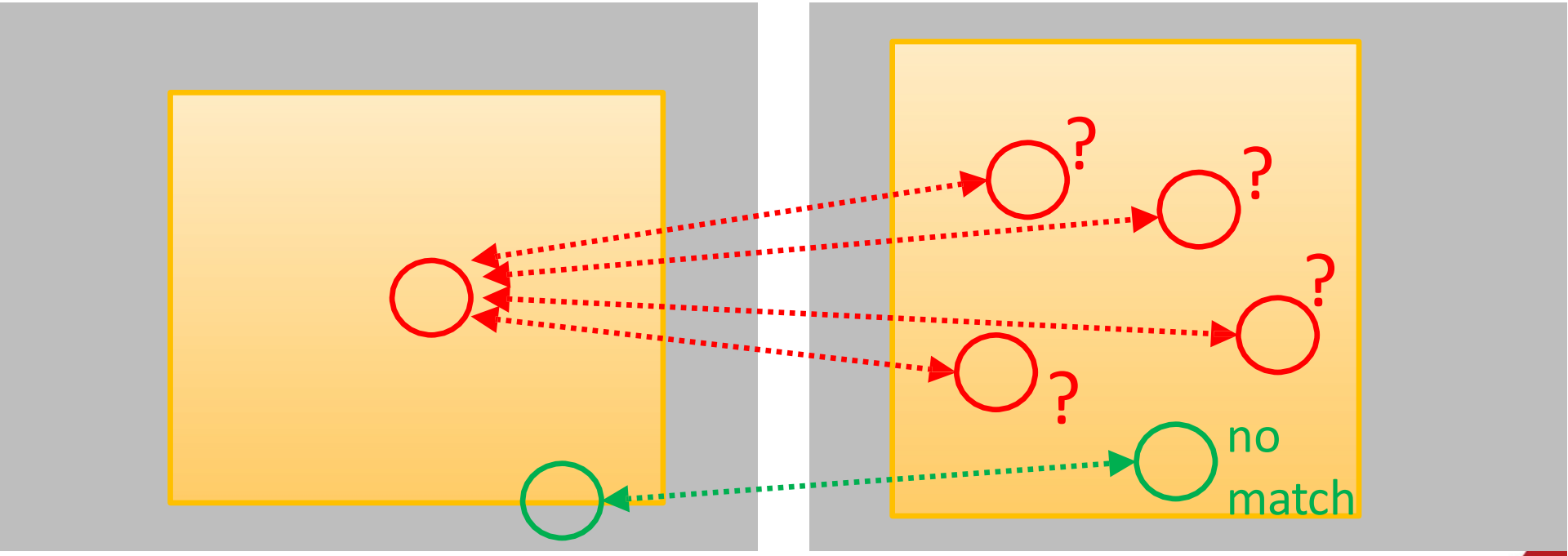
Harris Corner Detector (Intuition)

- Flat regions are not distinctive



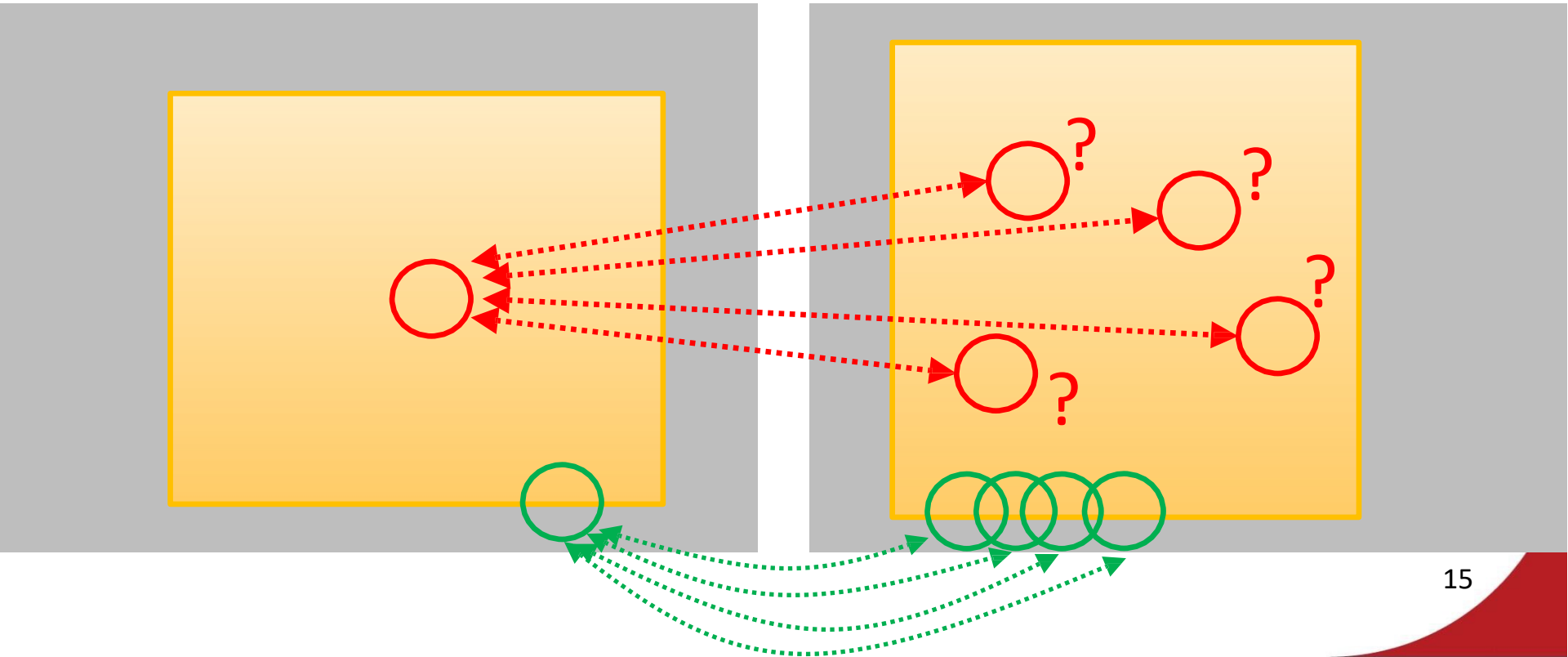
Harris Corner Detector (Intuition)

- Edges are more distinctive than flat regions



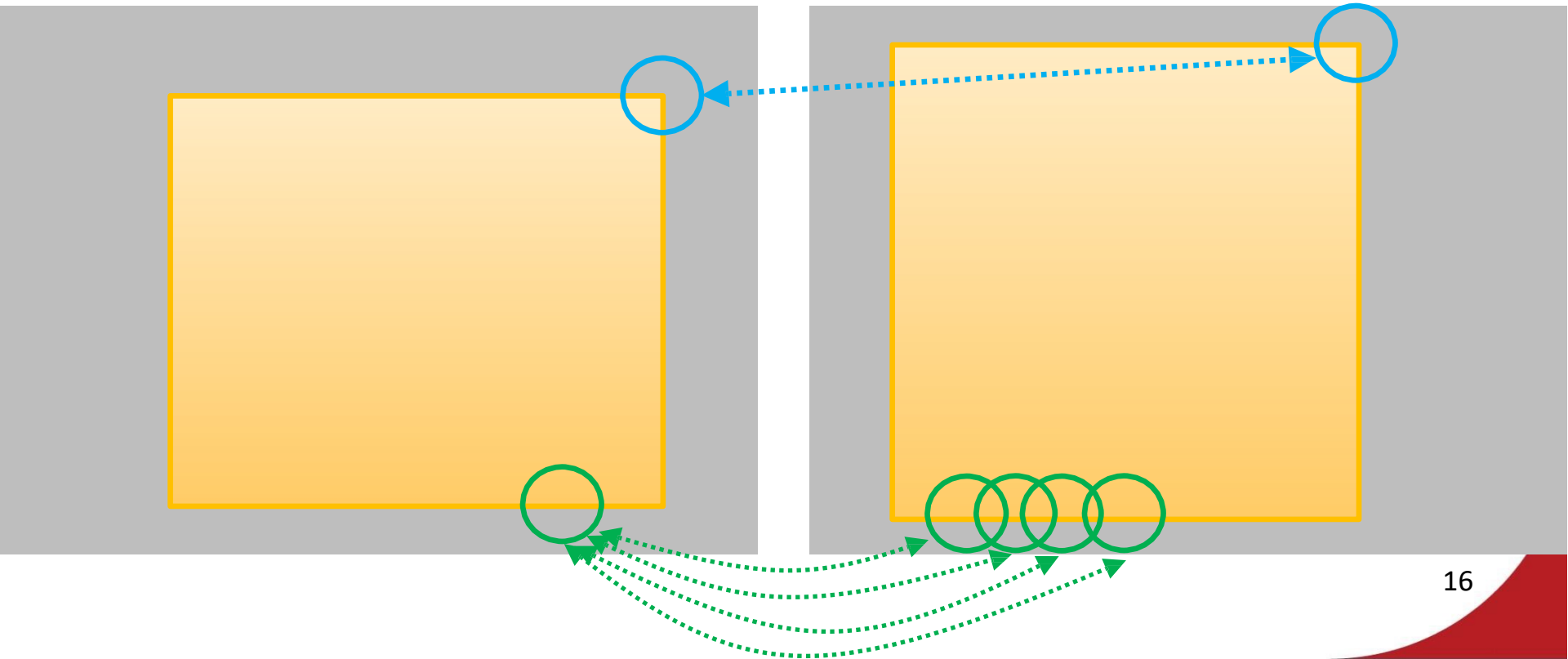
Harris Corner Detector (Intuition)

- But still …



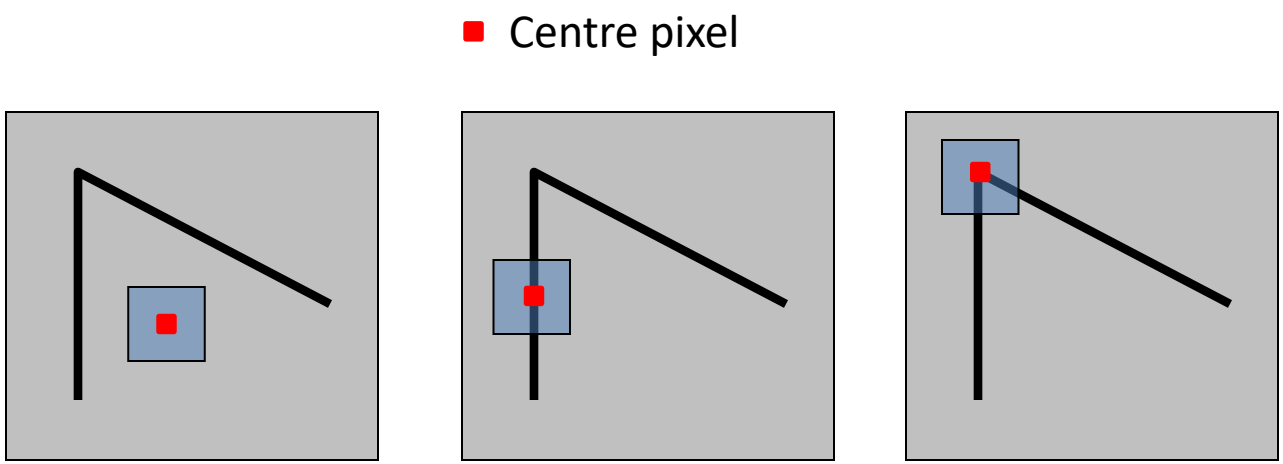
Harris Corner Detector (Intuition)

- Corners are even more distinctive than edges



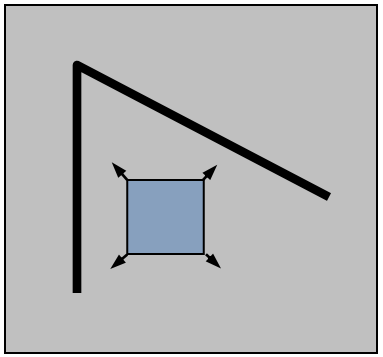
Harris Corner Detector

- Intuition: Find points that different from their neighborhood
- What defines whether a point/pixel is a good or bad candidate?



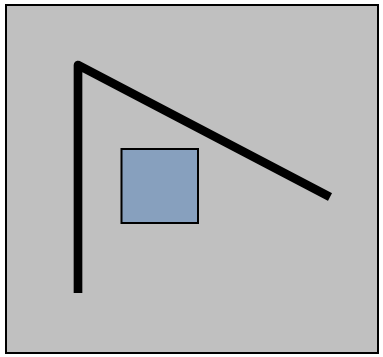
How to find corners in an image?

- Find points that different from their neighborhood by shifting the window in all possible directions

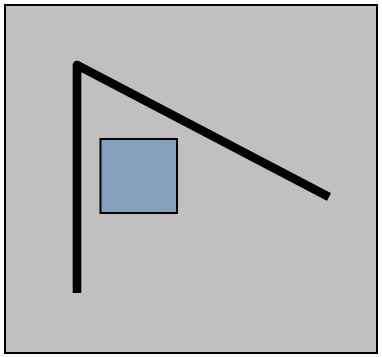


"flat" region

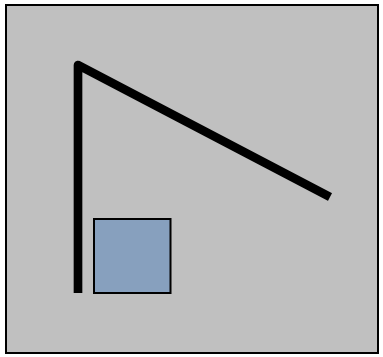
no change in all directions



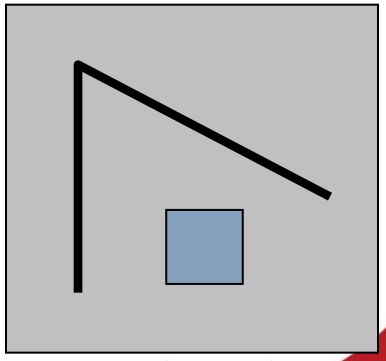
Not Changed



Not Changed



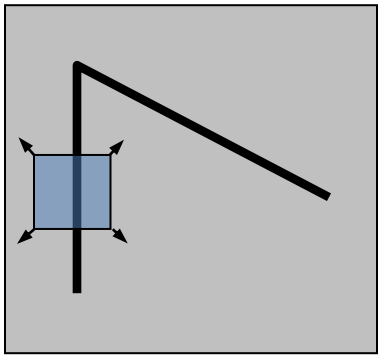
Not changed



Not changed

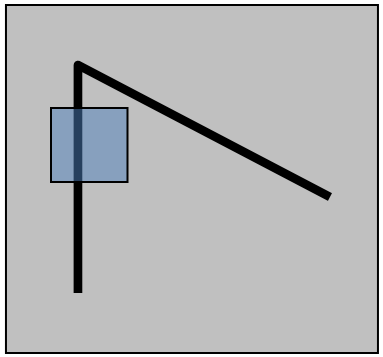
How to find corners in an image?

- Find points that different from their neighborhood by shifting the window in all possible directions

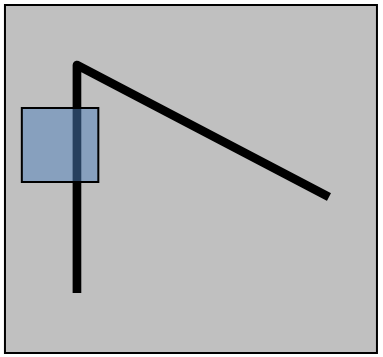


"edge"

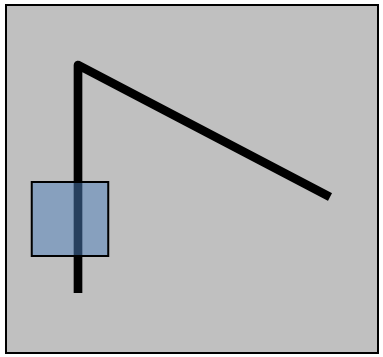
no change along the edge direction



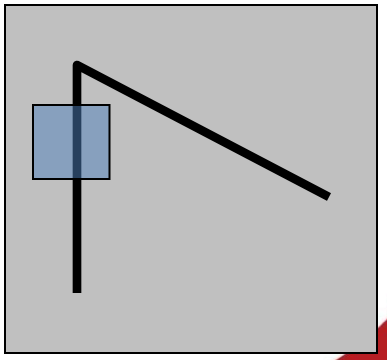
Changed



Changed



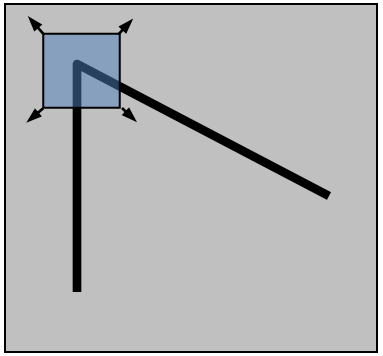
Not changed



Not changed

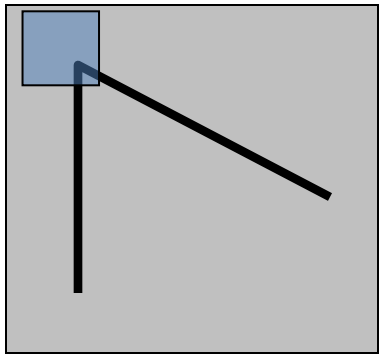
How to find corners in an image?

- Find points that different from their neighborhood by shifting the window in all possible directions

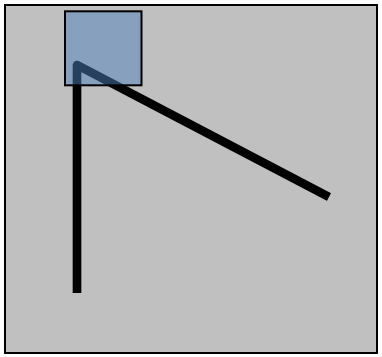


"corner"

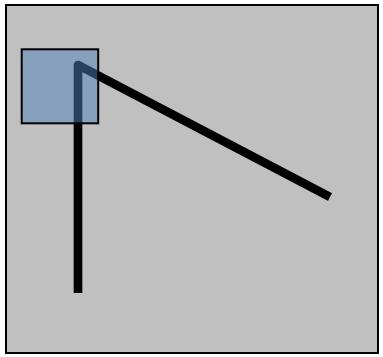
significant change in all directions



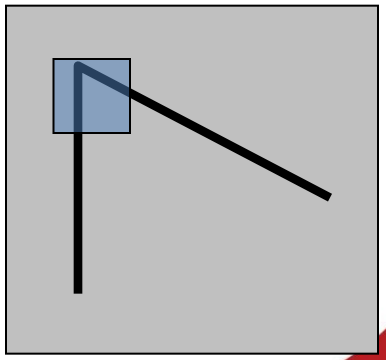
Changed



Changed



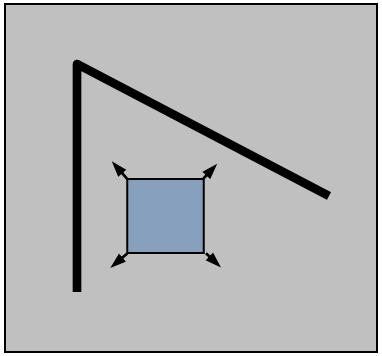
Changed



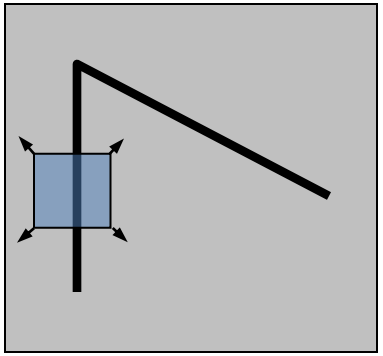
Changed

How to find corners in an image?

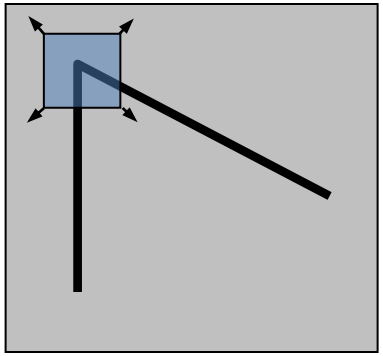
- Find points that different from their neighborhood by shifting the window in all possible directions



"flat" region
no change in all directions



"edge"
no change along the edge direction

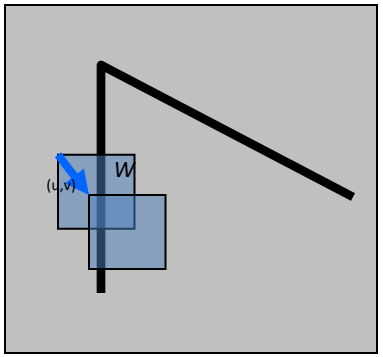


"corner"
significant change in all directions

Harris Corner Detection (the math)

Consider shifting the window *W* by (*u,v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" $E(u,v)$:



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

SSD Error

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

$$E(u,v) = \sum_{i=1}^{3}\sum_{j=1}^{3}\left[I\left(x_i+u, y_j+v\right) - I(x_i, y_j)\right]^2$$

$I(x_1, y_1)$	$I(x_2, y_1)$	$I(x_3, y_1)$
$I(x_1, y_2)$	$I(x_2, y_2)$	$I(x_3, y_2)$
$I(x_1, y_3)$	$I(x_2, y_3)$	$I(x_3, y_3)$

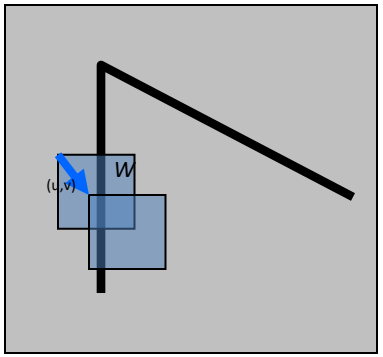
(u,v)

$I(x_1+u, y_1+v)$	$I(x_2+u, y_1+v)$	$I(x_3+u, y_1+v)$
$I(x_1+u, y_2+v)$	$I(x_2+u, y_2+v)$	$I(x_3+u, y_2+v)$
$I(x_1+u, y_3+v)$	$I(x_2+u, y_3+v)$	$I(x_3+u, y_3+v)$

Harris Corner Detection (the math)

Consider shifting the window *W* by (*u,v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" $E(u,v)$:



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

- For nearly constant patches, $E(u,v)$ will be near 0.
- For very distinctive patches, $E(u,v)$ will be larger.
- Hence... we want patches where $E(u,v)$ is LARGE.
- Slow to compute exactly for each pixel and each offset (u,v)

Small Motion Assumption

Taylor Series expansion of *I*:

$$I(x+u, y+v) = I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion (*u*, *v*) is small, then first order approximation is good

$$\begin{aligned} I(x+u, y+v) &\approx I(x,y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v \\ &\approx I(x,y) + I_x u + I_y v \\ &\approx I(x,y) + [I_x \;\; I_y] \begin{bmatrix} u \\ v \end{bmatrix} \end{aligned}$$

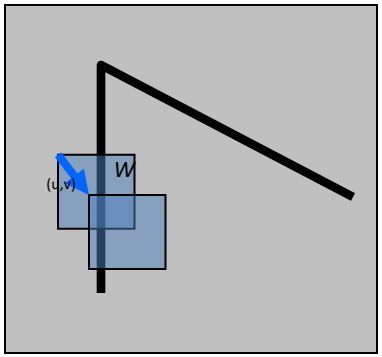
shorthand: $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide…

Harris Corner Detection (the math)

Consider shifting the window *W* by (*u,v*)

- define an SSD "error" *E(u,v)*:



$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u, y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

Harris Corner Detection (the math)

$$
\begin{aligned}
E(u,v) &= \sum_{(x,y)\in W} [I(x+u,y+v) - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I(x,y) + I_x u + I_y v - I(x,y)]^2 \\
&\approx \sum_{(x,y)\in W} [I_x u + I_y v]^2
\end{aligned}
$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

$$\approx \sum_{(x,y)\in W} (I_x u)^2 + (I_y v)^2 + 2 I_x I_y uv$$

$$\approx \sum_{(x,y)\in W} {I_x}^2 u^2 + {I_y}^2 v^2 + 2 I_x I_y uv$$

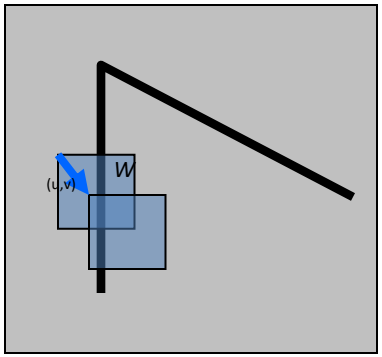
$$\approx u^2 \sum_{(x,y)\in W} {I_x}^2 + v^2 \sum_{(x,y)\in W} {I_y}^2 + 2uv \sum_{(x,y)\in W} I_x I_y$$

Harris Corner Detection (the math)

$$E(u,v) \approx u^2 \underbrace{\sum_{(x,y)\in W} {I_x}^2}_{A} + v^2 \underbrace{\sum_{(x,y)\in W} {I_y}^2}_{C} + 2uv \underbrace{\sum_{(x,y)\in W} I_x I_y}_{B}$$

$$E(u,v) \approx u^2 A + v^2 C + 2uvB$$

Harris Corner Detection (the math)



Consider shifting the window *W* by (*u,v*)

- define an SSD "error" $E(u,v)$:

$$E(u,v) \approx \sum_{(x,y)\in W} [I_x u + I_y v]^2$$

$$\approx Au^2 + 2Buv + Cv^2$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

The Second Moment Matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

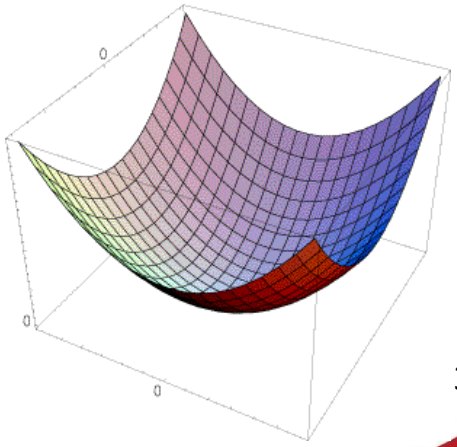
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Note: these are just products of components of the gradient, I_x, I_y

shorthand: $I_x = \frac{\partial I}{\partial x}$

Let's try to understand its shape.



The Second Moment Matrix (for Edges)

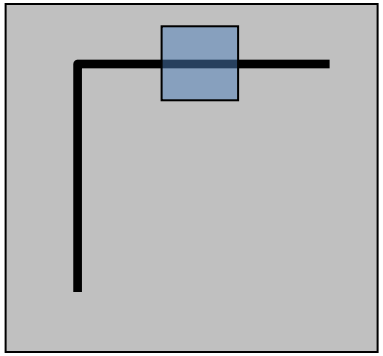
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

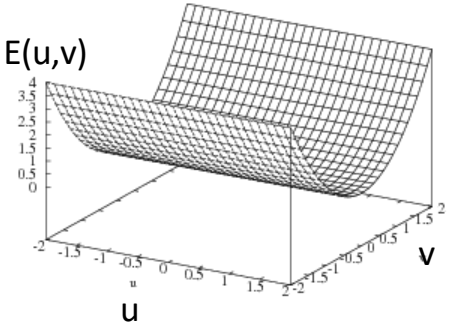
$$C = \sum_{(x,y)\in W} I_y^2$$

shorthand: $I_x = \frac{\partial I}{\partial x}$



Horizontal edge: $I_x = 0$

$$H = \begin{bmatrix} 0 & 0 \\ 0 & C \end{bmatrix}$$



The Second Moment Matrix (for Edges)

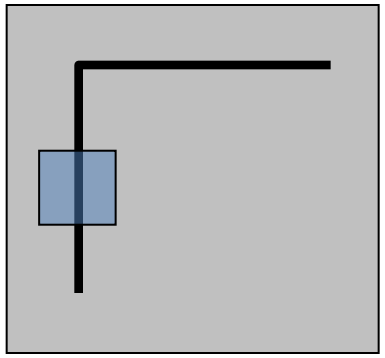
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

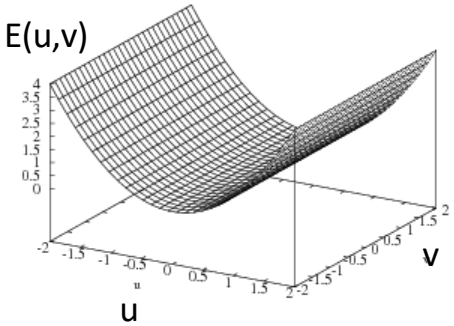
$$C = \sum_{(x,y)\in W} I_y^2$$

shorthand: $I_x = \frac{\partial I}{\partial x}$



Vertical edge: $I_y = 0$

$$H = \begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$$



Intuitive Way to Understand Harris

- Treat gradient vectors as a set of (I_x, I_y) points with a centre of mass defined as being at (0,0).

- Fit an ellipse to that set of points via scatter matrix

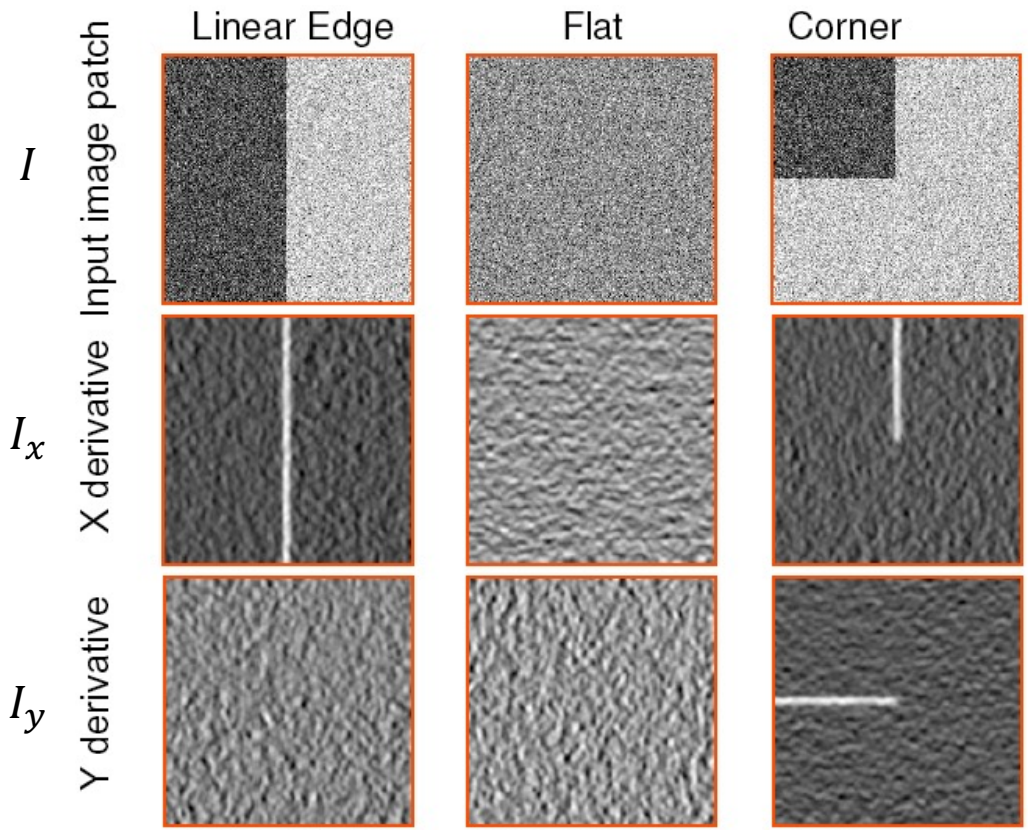
- Analyse ellipse parameters for varying cases…

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

Harris Corner Detection (Intuition)

- Let's calculate the derivatives of these images along x and y directions, e.g., Sobel edge detector

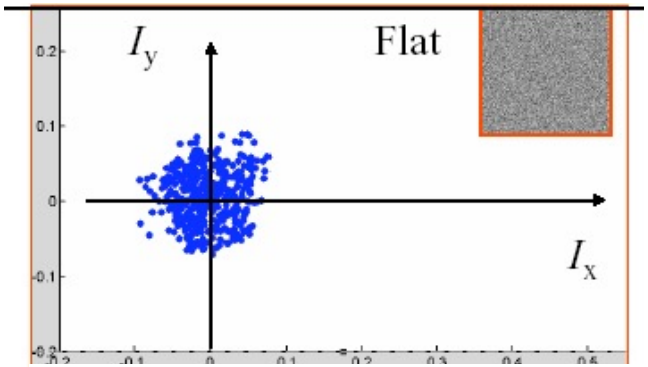


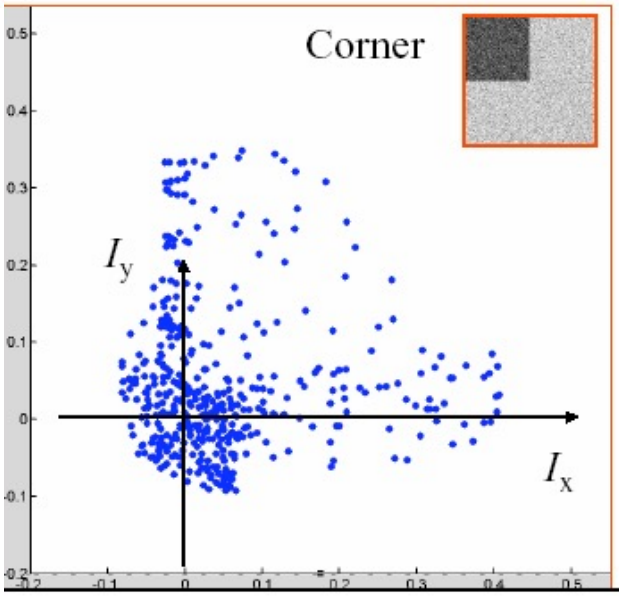
-1	-2	-1
0	0	0
1	2	1

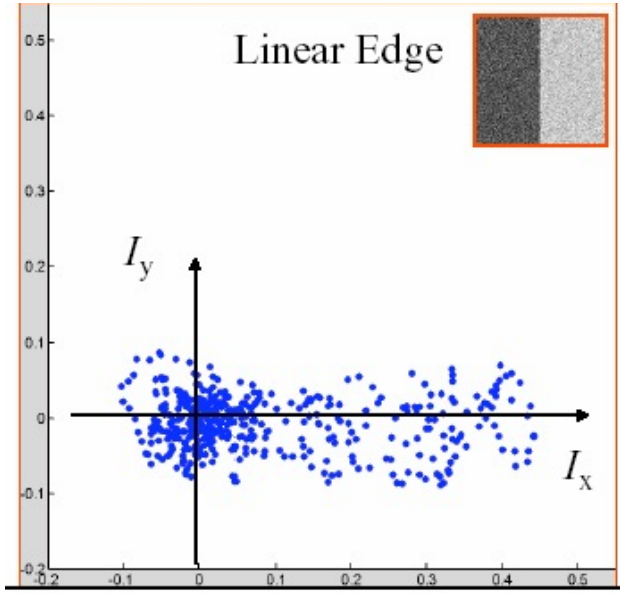
-1	0	1
-2	0	2
-1	0	1

Harris Corner Detection (Intuition)

- Let's treat gradient vectors as a set of (I_x, I_y) points



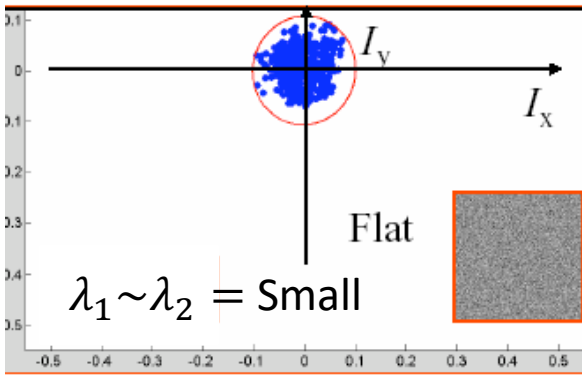


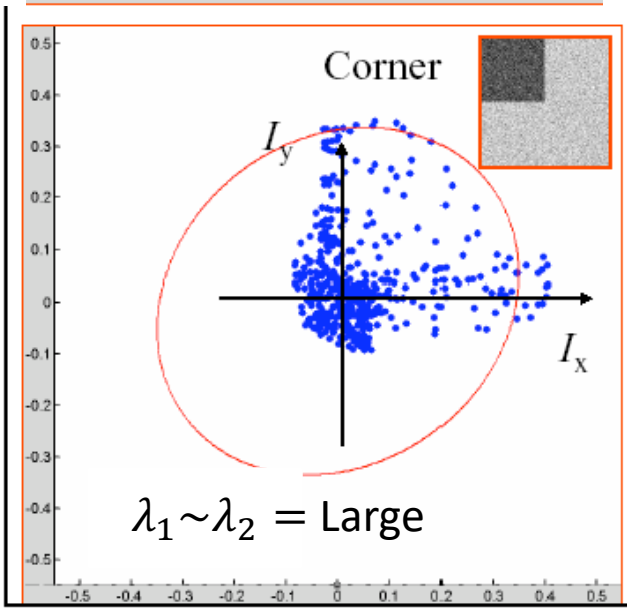


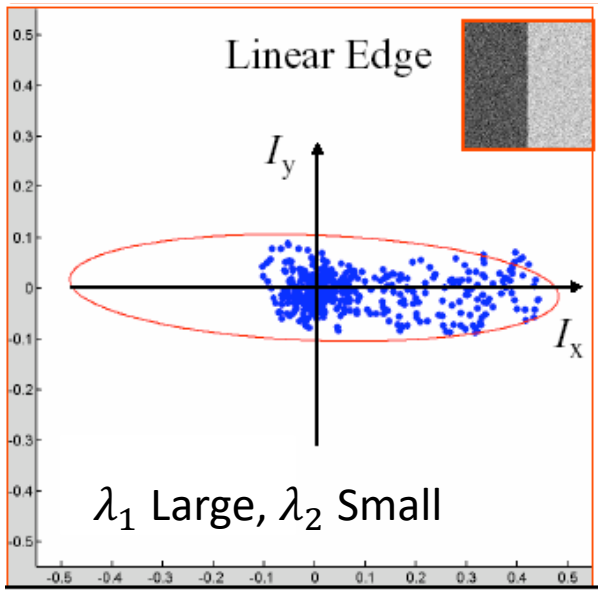
Note: The distribution of points is very different for these three types of patches

Harris Corner Detection (Intuition)

- Let's treat gradient vectors as a set of (I_x, I_y) points







The distribution of points can be categorised by fitting an ellipse to data points and analysing major axis and minor axis of the ellipse.

SCC.366: Media Coding & Processing

2022-2023

Week 08 – Lecture 1

Local Image Matching Cont.

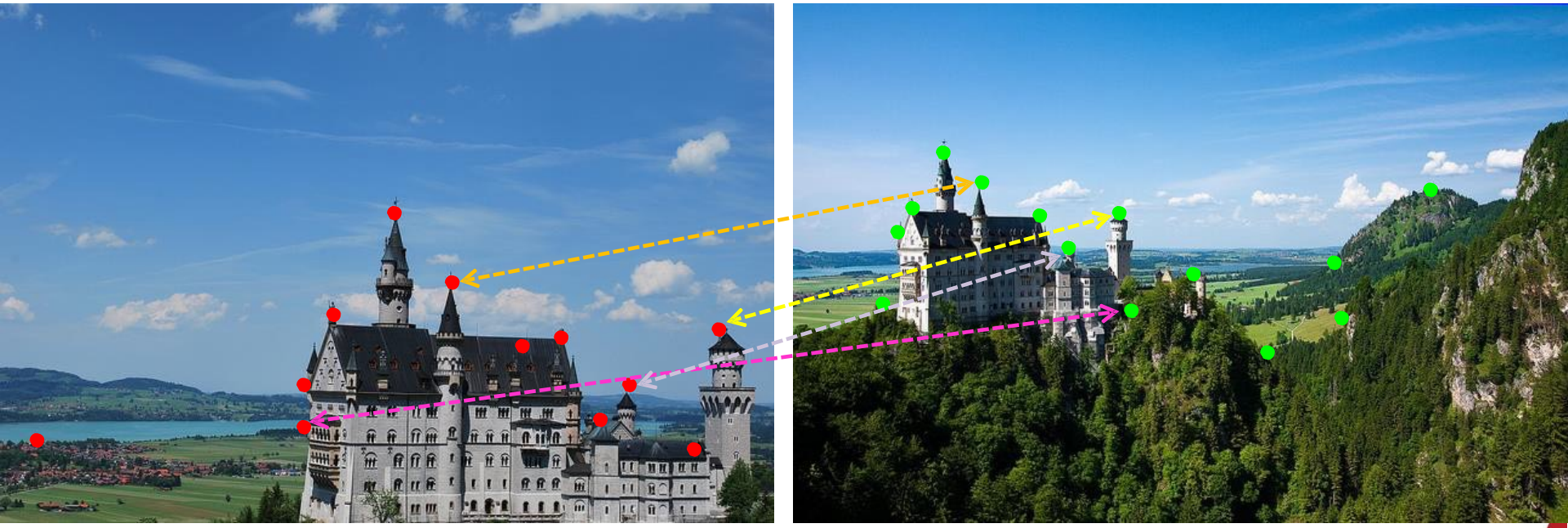
Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- What is Image matching?
- Applications of image matching
- Challenges in image matching
- Holistic vs. local image matching
- Holistic image matching an its limitations
  - Intensity based descriptor
  - Feature based descriptor e.g., HOG
- Local image matching (This Lecture)
  - Interest point detection
  - Interest point description
  - Matching process
- Summary of the main points

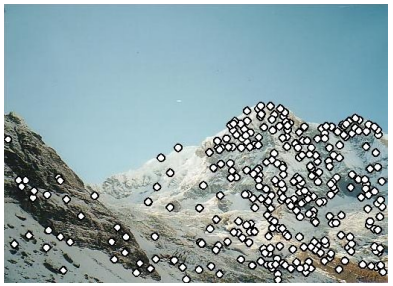
Local Image Matching (Definition)

- Image matching is the process of establishing a correspondence between a pair of images



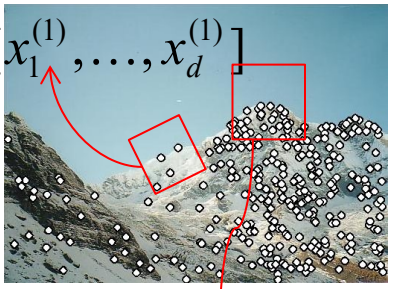
Local Image Matching (main components)

1) **Detection**: Identify the interest points, e.g., Harris corner and SIFT



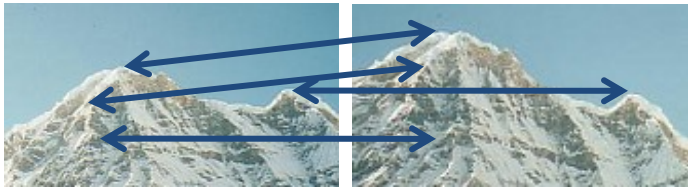
2) **Description**: Extract feature descriptor surrounding each interest point, e.g., HOG, SIFT

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

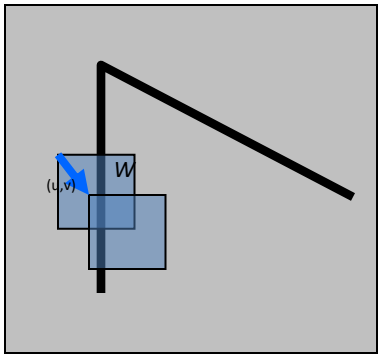
3) **Matching**: Determine correspondence between descriptors in two views



Harris Corner Detection (the math)

Consider shifting the window *W* by (*u,v*)

- how do the pixels in W change?
- compare each pixel before and after by summing up the squared differences (SSD)
- this defines an SSD "error" $E(u,v)$:



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

- For nearly constant patches, $E(u,v)$ will be near 0.
- For very distinctive patches, $E(u,v)$ will be larger.
- Hence... we want patches where $E(u,v)$ is LARGE.
- Slow to compute exactly for each pixel and each offset (u,v)

The Second Moment Matrix

The surface $E(u,v)$ is locally approximated by a quadratic form.

$$E(u,v) \approx Au^2 + 2Buv + Cv^2$$

$$\approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

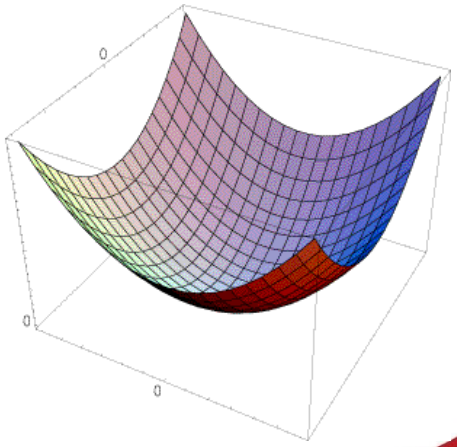
$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

Note: these are just products of components of the gradient, I_x, I_y

shorthand: $I_x = \frac{\partial I}{\partial x}$



Harris Corner Detection (the math)

$$A = \sum_{(x,y)\in W} I_x^2$$

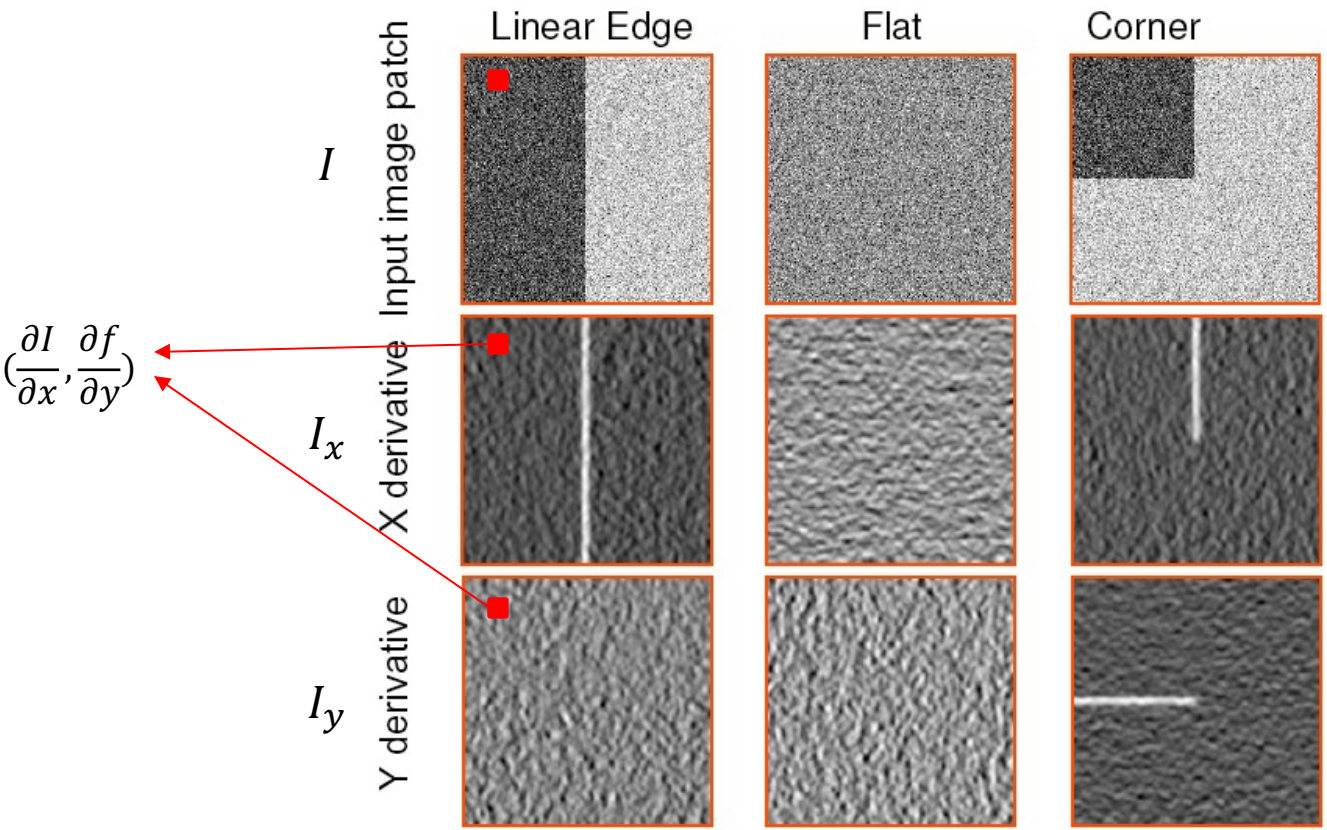
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

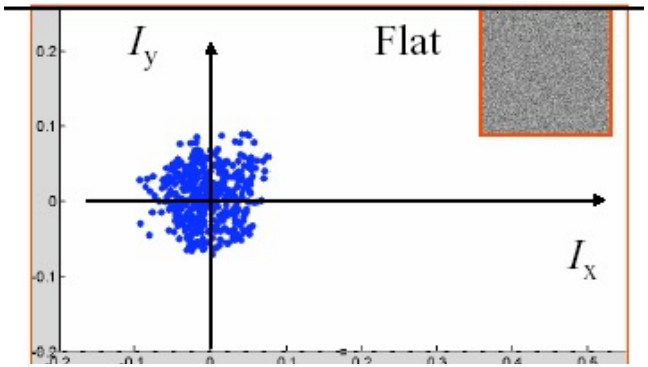
Harris Corner Detection (Intuition)

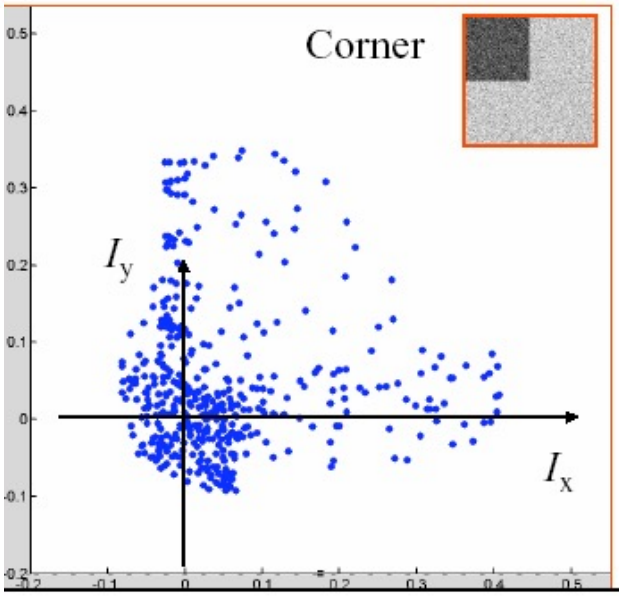
- Let's calculate the derivatives of these images along x and y directions, e.g., Sobel edge detector

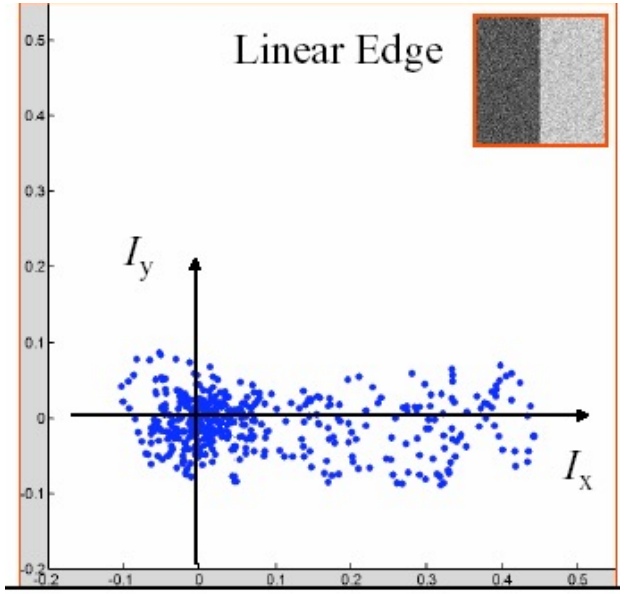


Harris Corner Detection (Intuition)

- Let's treat gradient vectors as a set of (I_x, I_y) points



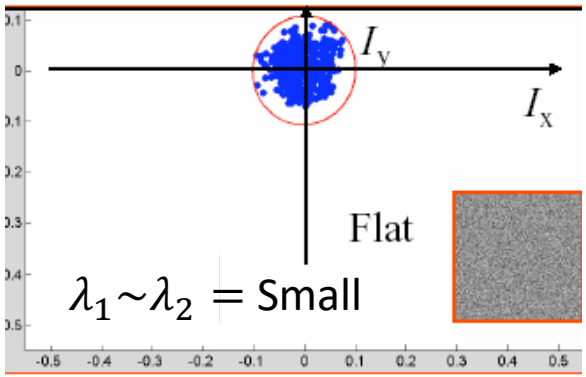


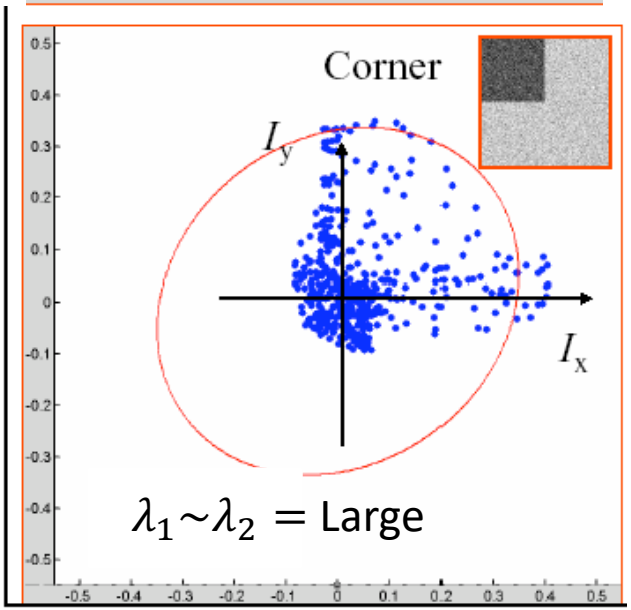


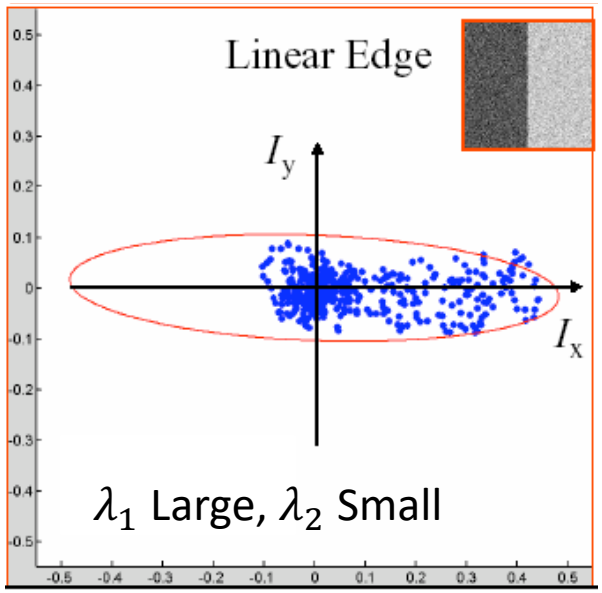
Note: The distribution of points is very different for these three types of patches

Harris Corner Detection (Intuition)

- Let's treat gradient vectors as a set of (I_x, I_y) points







The distribution of points can be categorised by fitting an ellipse to data points and analysing major axis and minor axis of the ellipse.

Harris Corner Detection (the math)

The surface $E(u,v)$ is locally approximated by a quadratic form.

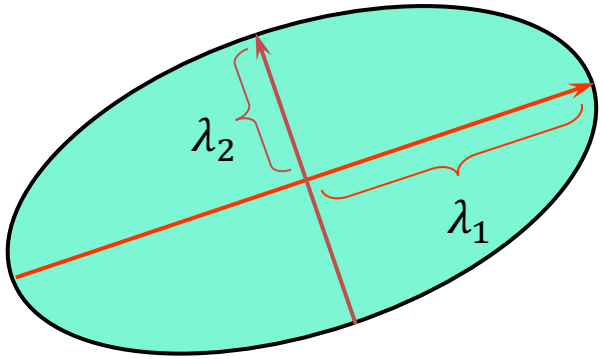
$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

shorthand: $I_x = \frac{\partial I}{\partial x}$

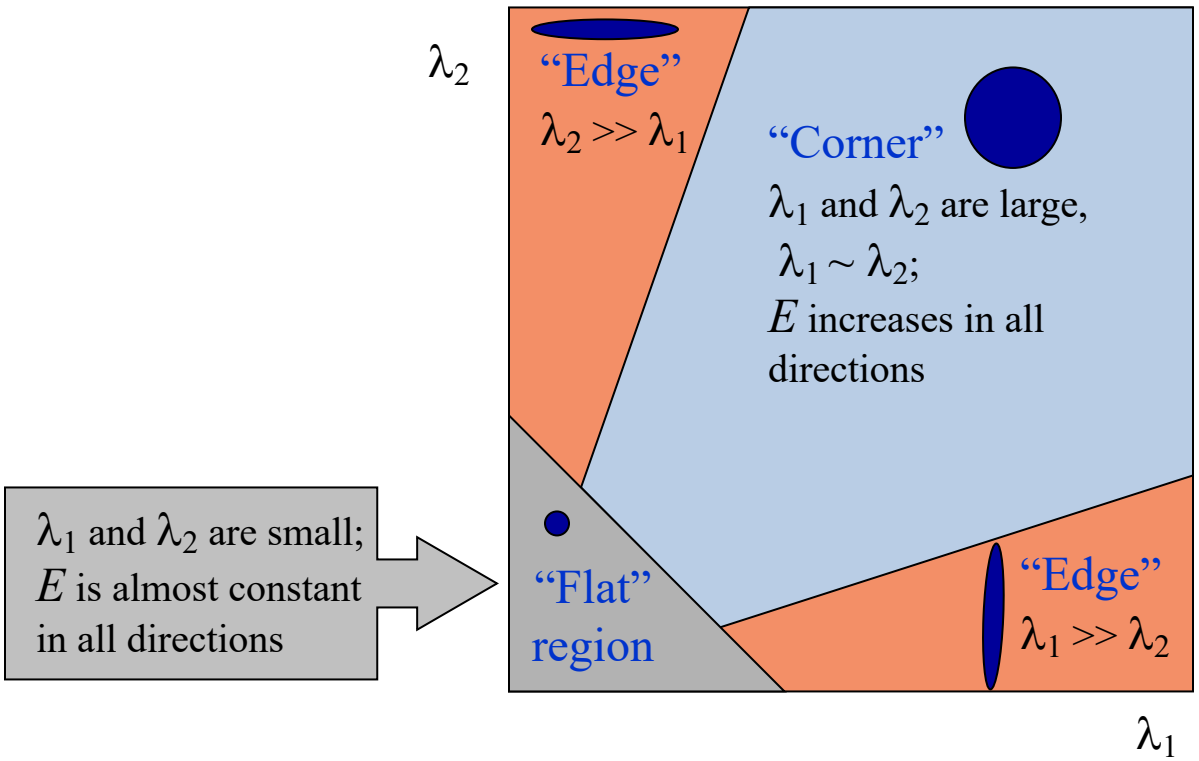


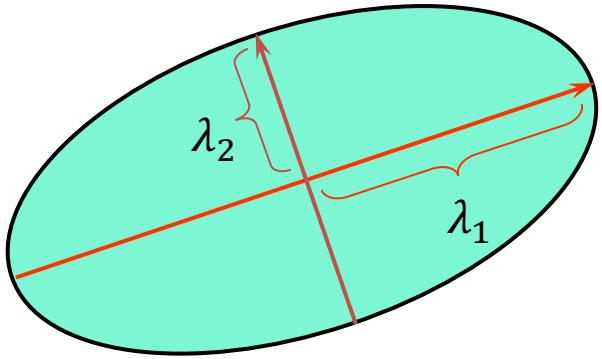
The major semi axis: $\lambda_1 = \frac{1}{2}\left[A + C + \sqrt{B^2 + (A-C)^2}\right]$

The minor semi axis: $\lambda_2 = \frac{1}{2}\left[A + C - \sqrt{B^2 + (A-C)^2}\right]$

Interpreting major and minor axis of ellipse

Classification of image points using λ_1 and λ_2.





Corner Response Measure

How to define a single metric R?

Measure of corner response is defined as:

$$R = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2} = \frac{det(H)}{tr(H)}$$

OR

$$\boxed{\begin{aligned} R &= \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \\ &= \det(H) - k * tr(H)^2 \end{aligned}}$$

λ_1 major semi axis
λ_2 minor semi axis
Both should be large

- K is empirically determined constant; $k = 0.04 - 0.06$

Determinant and Trace of a Matrix

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

$$det(H) = A \times C - B \times B$$

$$tr(H) = A + C$$

Corner Response Measure

Measure of corner response is defined as:

$$A = \sum_{(x,y)\in W} I_x^2$$

$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} I_y^2$$

e.g., W is 7×7

$$R = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2$$
$$= \det(H) - k * tr(H)^2$$
$$= A\times C - B\times B - k\times(A+C)^2$$

- K is empirically determined constant; $k = 0.04 - 0.06$

Non-Maxima Suppression

- Corner response R is calculated for all pixels
- Step 1: Generate corner candidates
  - The (x, y)-th pixel becomes a corner candidate if its corner score is high enough, e.g.,

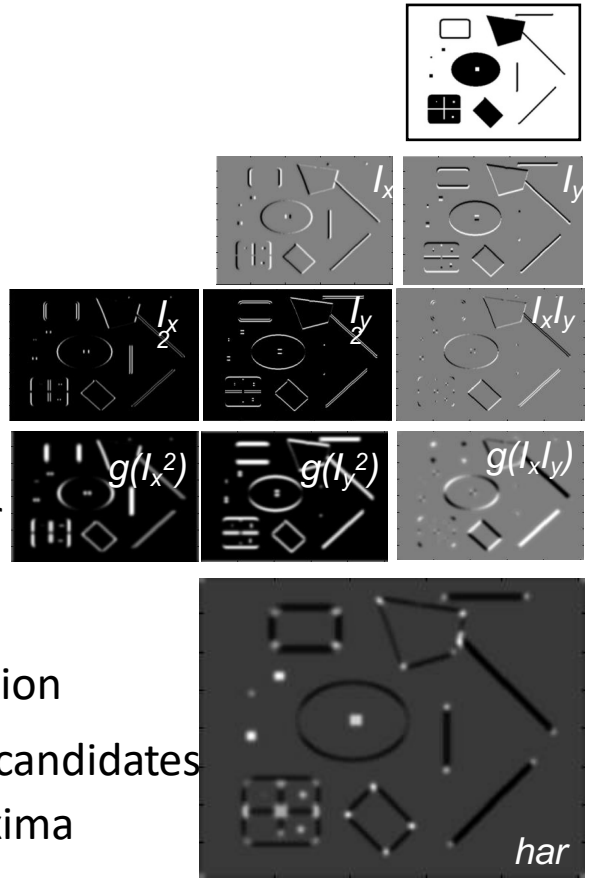
$$R(x, y) > 0.1 \times \max(R)$$

Maximum R in the image

- Step 2: Choose local maxima
  - The candidate pixel (x, y) becomes a corner if $R(x, y)$ is a local maximum in a 3×3 window, i.e., if
    - $R(x, y) > R(x-1, y-1)$
    - $R(x, y) > R(x-1, y)$
    - $R(x, y) > R(x-1, y+1)$
    - $R(x, y) > R(x, y-1)$
    - $R(x, y) > R(x, y+1)$
    - $R(x, y) > R(x+1, y-1)$
    - $R(x, y) > R(x+1, y)$
    - $R(x, y) > R(x+1, y+1)$

Harris Detector (Summary)

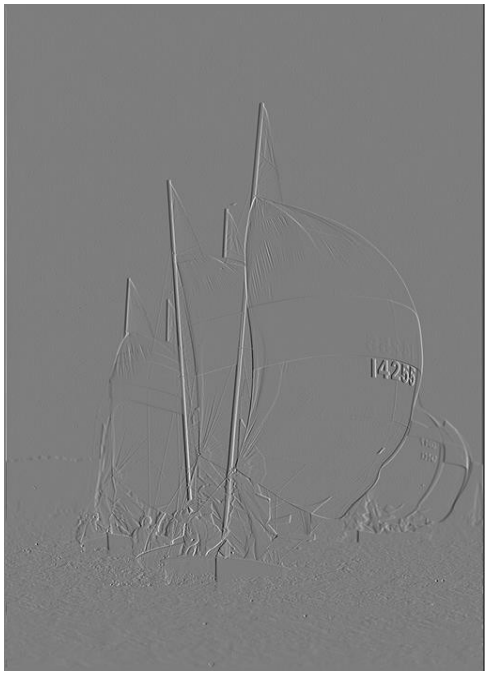
1. Image derivatives
2. Square of derivatives
3. Gaussian filter
4. Cornerness function
5. Generate corner candidates
6. Choose local maxima



Harris Detector (Step 1)

1) Generate two derivative images (e.g., Sobel)

I

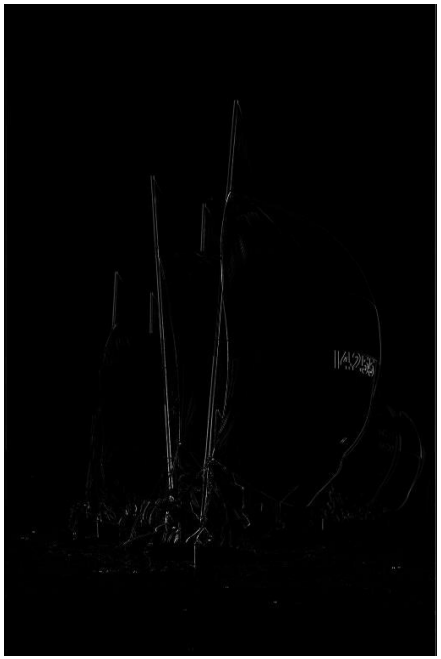


$$I_x = \frac{\partial I}{\partial x}$$

$$I_y = \frac{\partial I}{\partial y}$$

Harris Detector (Step 2)

2) Generate three products of derivative images



$I_x{}^2$

$I_x I_y$

$I_y{}^2$

Harris Detector (Step 3)

3) Smooth products of derivative images

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \underbrace{\begin{bmatrix} A & B \\ B & C \end{bmatrix}}_{H} \begin{bmatrix} u \\ v \end{bmatrix}$$

$$A = \sum_{(x,y)\in W} I_x^2 \qquad B = \sum_{(x,y)\in W} I_x I_y \qquad C = \sum_{(x,y)\in W} I_y^2$$

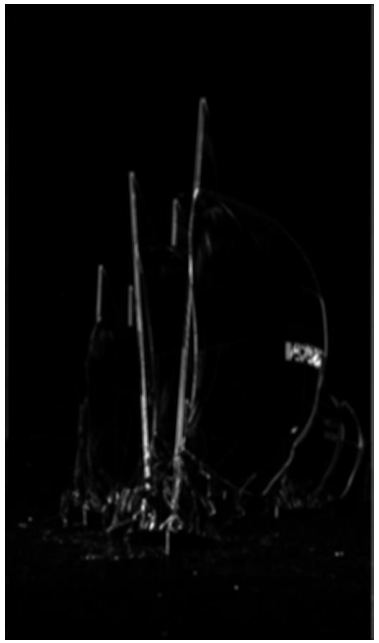
W is 7×7

Obtained by applying smoothing filters such as Averaging or Gaussian

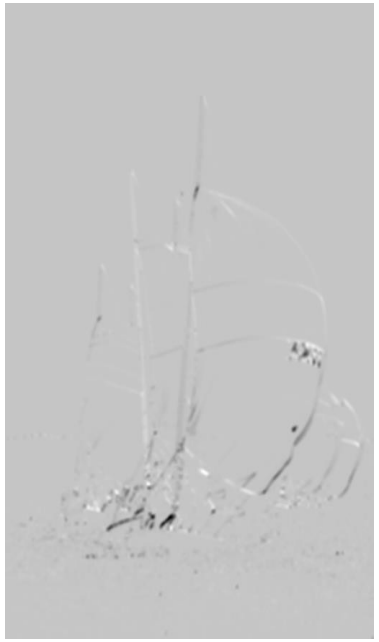
In practice we use Gaussian filtering

Harris Detector (Step 3)

3) Smooth products of derivative images



$$A = \sum_{(x,y)\in W} {I_x}^2$$



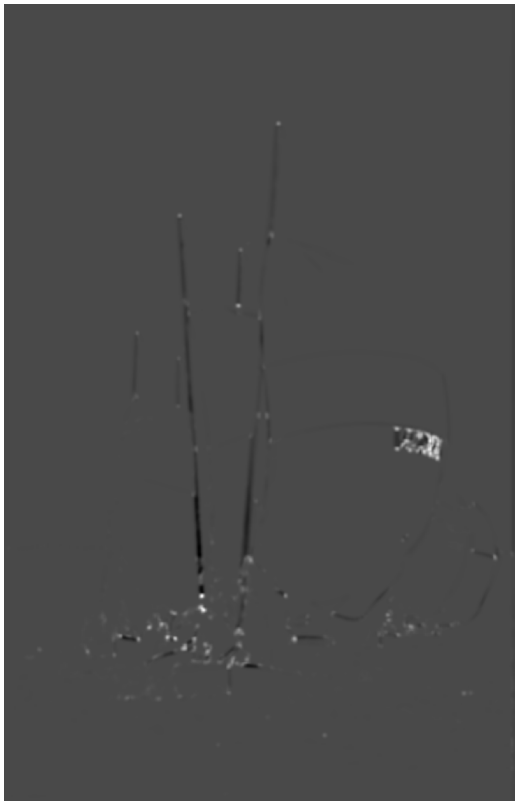
$$B = \sum_{(x,y)\in W} I_x I_y$$

$$C = \sum_{(x,y)\in W} {I_y}^2$$

Obtained by applying Gaussian smoothing

Harris Detector (Step 4)

4) Generate a corner score image



$$R = A \times C - B \times B - k \times (A + C)^2$$

$$A = \sum_{(x,y) \in W} {I_x}^2$$

$$B = \sum_{(x,y) \in W} I_x I_y$$

$$C = \sum_{(x,y) \in W} {I_y}^2$$

Harris Detector (Step 5)

5) Generate corner candidates

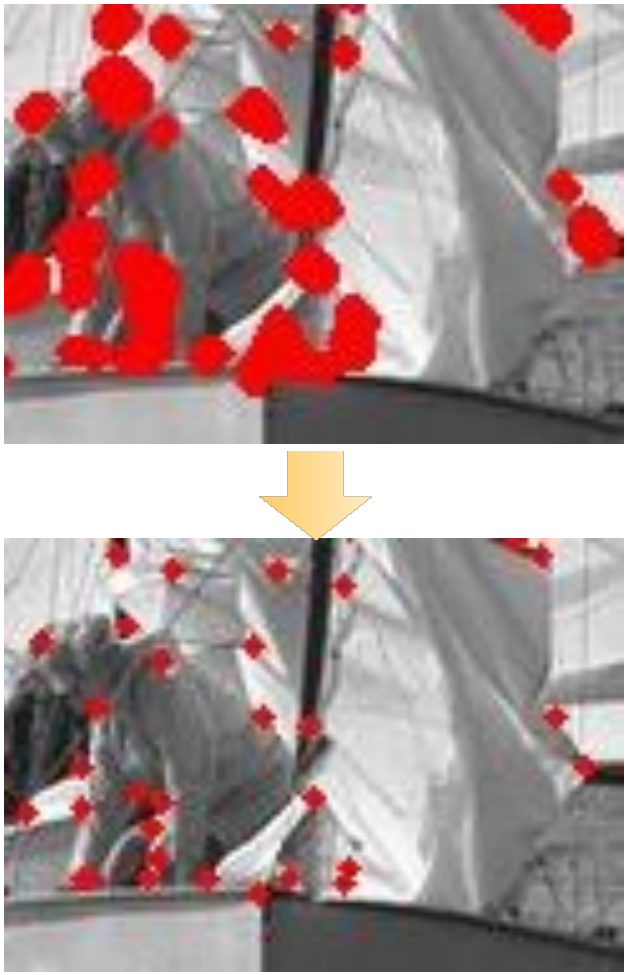
R

The (x, y)-th pixel becomes a corner candidate (marked as a red dot in the left image) if its corner score is *high enough*:

$$R(x, y) > 0.1 \times \max(R)$$

Harris Detector (Step 6)

6) Choose local maxima

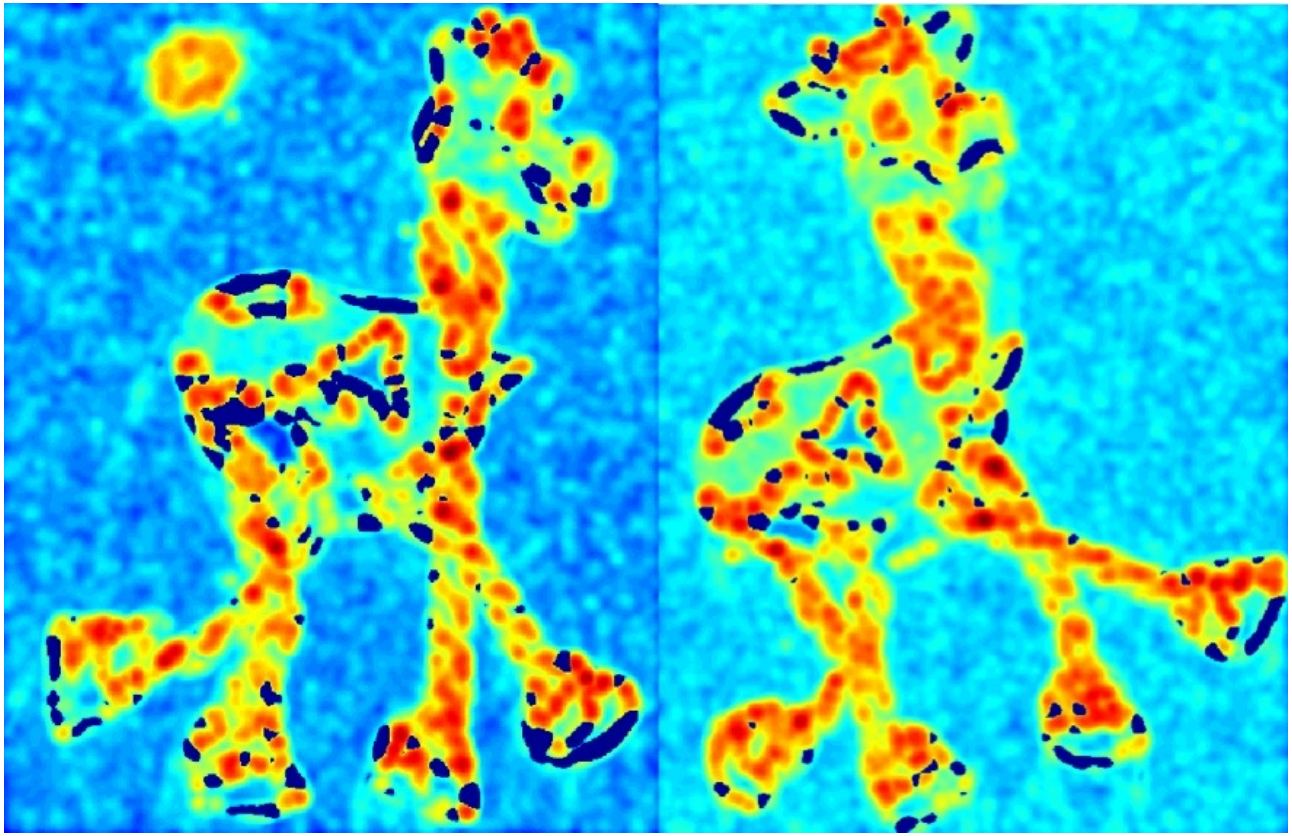


The candidate pixel (x,y) becomes a corner if $R(x,y)$ is a local maximum in a 3×3 window, i.e., if

- $R(x,y) > R(x-1,y-1)$
- $R(x,y) > R(x-1,y)$
- $R(x,y) > R(x-1,y+1)$
- $R(x,y) > R(x,y-1)$
- $R(x,y) > R(x,y+1)$
- $R(x,y) > R(x+1,y-1)$
- $R(x,y) > R(x+1,y)$
- $R(x,y) > R(x+1,y+1)$

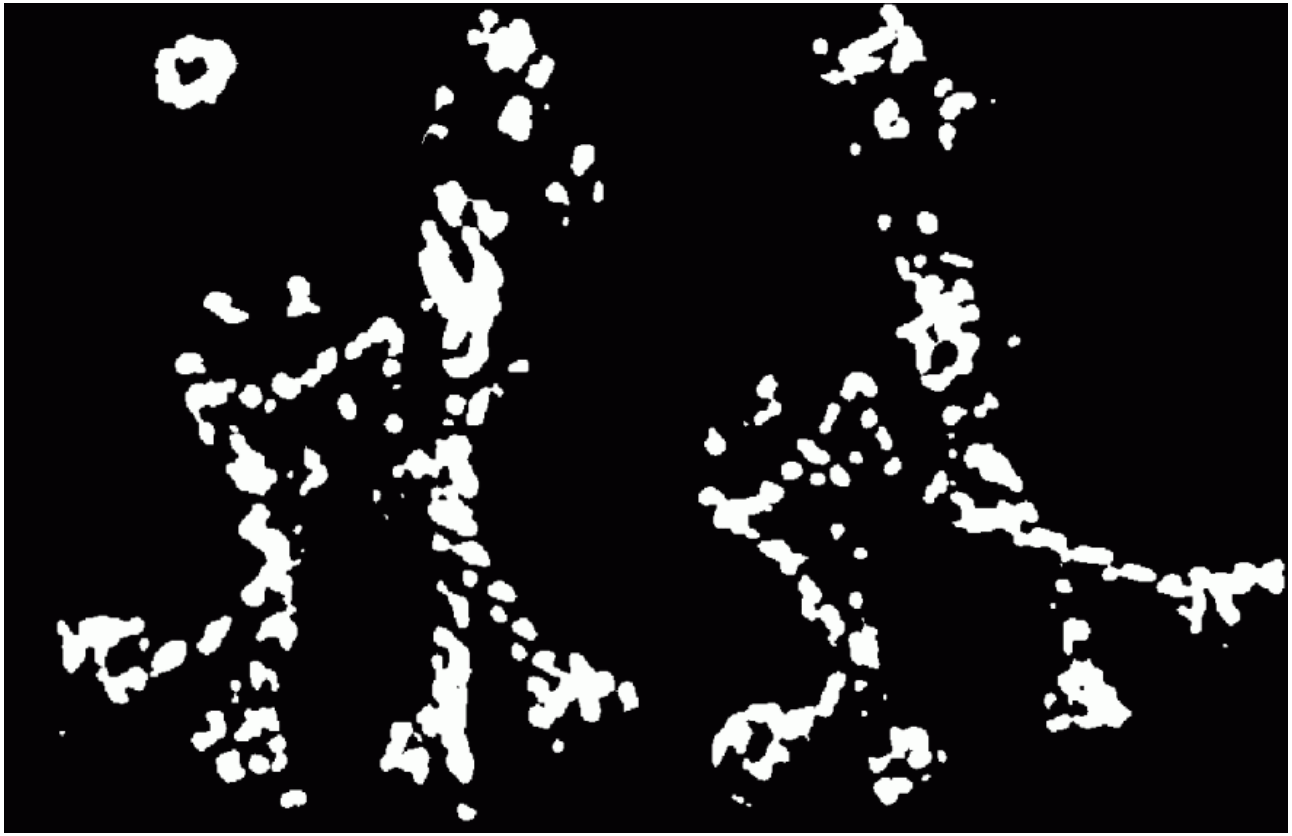
Harris Detector (Example)

R Value



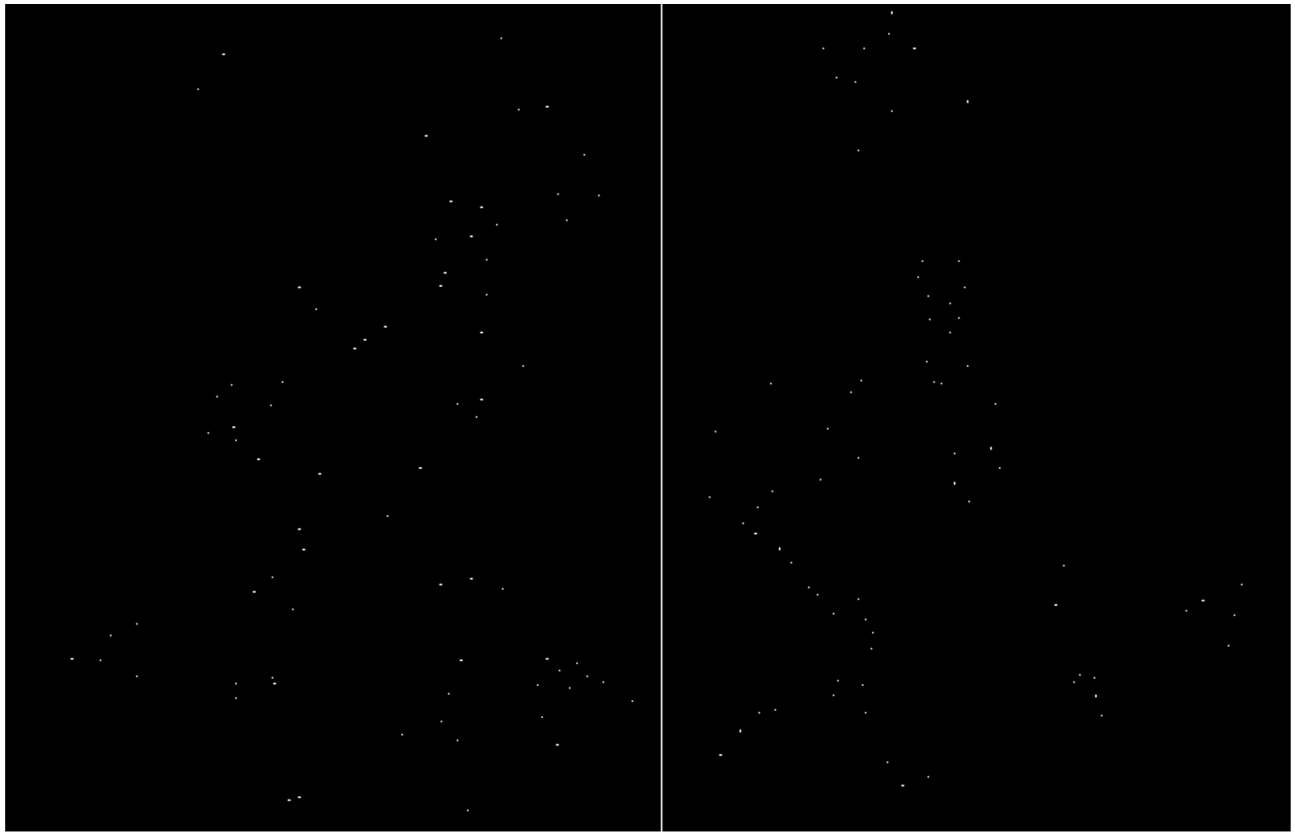
RED: high R value,
BLUE: low R value

Thresholding



$$R(x, y) > 0.1 \times \max(R)$$

Non-Maxima Suppression



Harris Interest Points (in red)

Detected interest points are shown in RED color

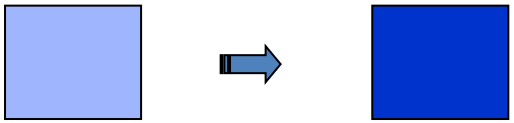
A Good Corner Detector

- Accurate and repeatable localization of the feature points
- Invariance to translation, rotation, scale, viewpoint
- Robustness to noise, lighting conditions, compression and blur

- Let's see if Harris interest point detector is robust to such variations

Robustness to Intensity Shift

- Let's see if Harris interest point detector is robust to intensity shift

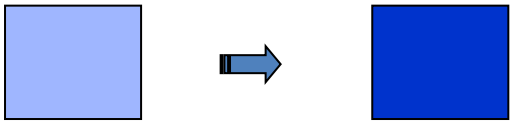


Invariance to intensity shift $I \rightarrow I + b$

Second moment ellipse remains the same as only derivatives are used

Robustness to Intensity Scaling (Homework)

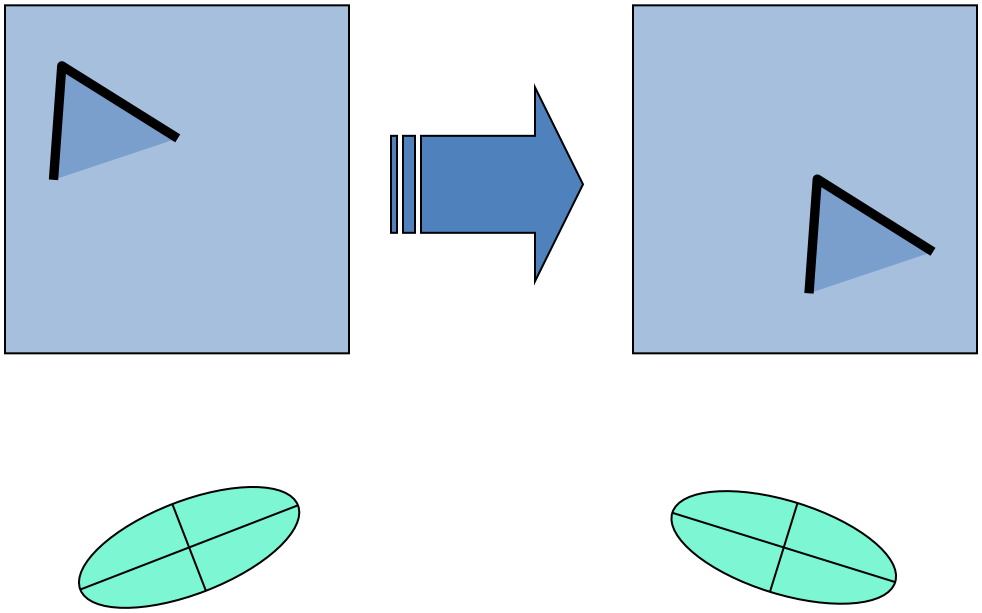
- Is Harris interest point detector robust to intensity scaling?



$I \rightarrow \mathrm{a}I$

Robustness to Image Translation (Homework)

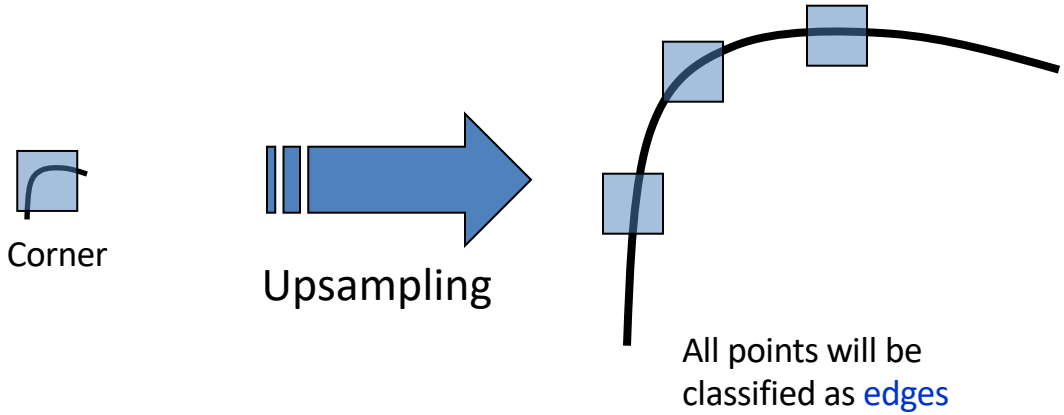
- Let's see if Harris interest point detector is robust to image translation



Note: Second moment ellipse rotates but the lengths of both the semi major axis and semi minor axis remain the same

Robustness to Spatial Scaling

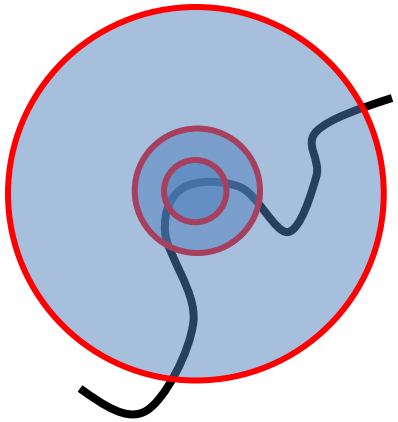
- Let's see if Harris interest point detector is robust to spatial scaling



- How to resolve this issue?

Robustness to Spatial Scaling

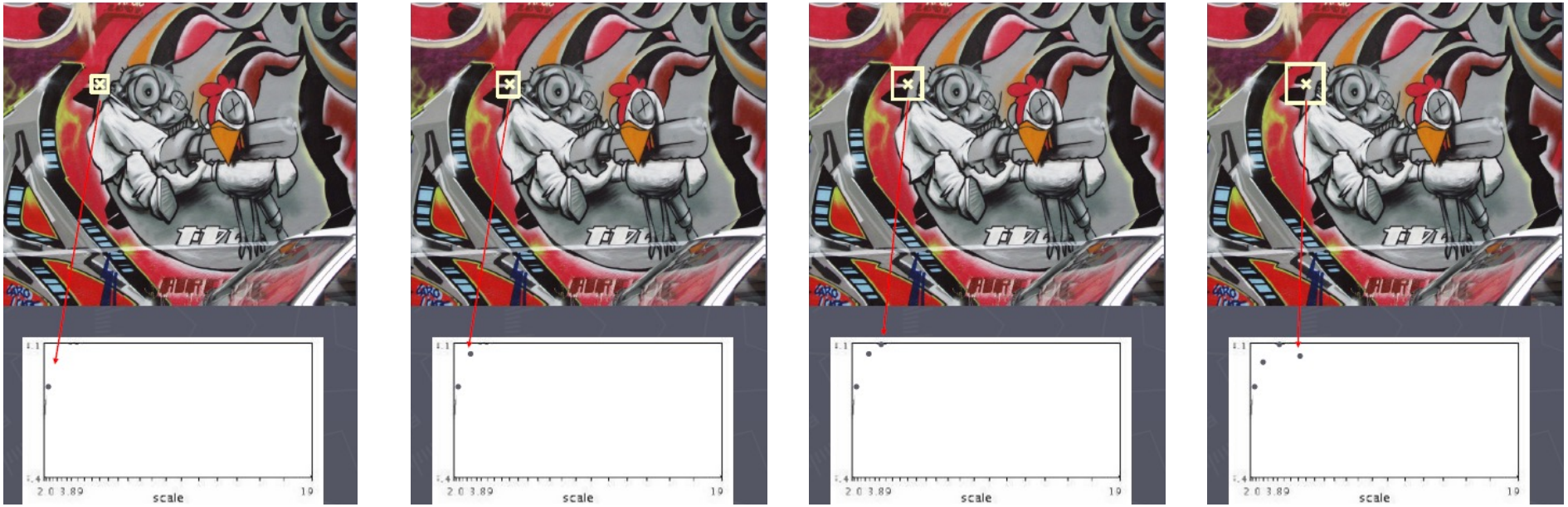
- Suppose you're looking for corners in the below image



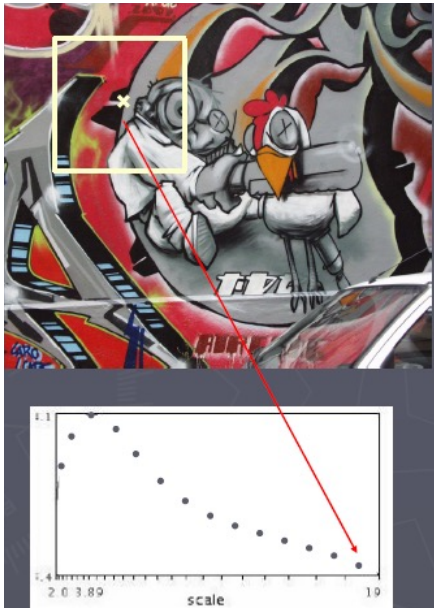
Key idea: find scale that gives local maximum of f in both position and scale

One definition of f: the Harris operator

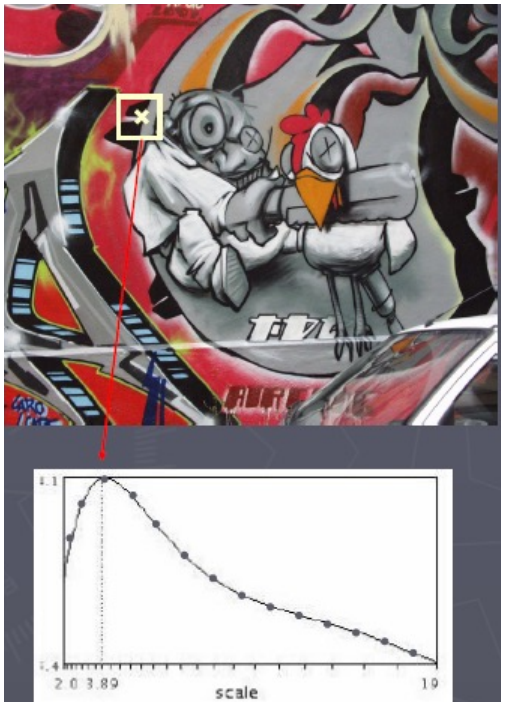
Robustness to Spatial Scaling (Example)

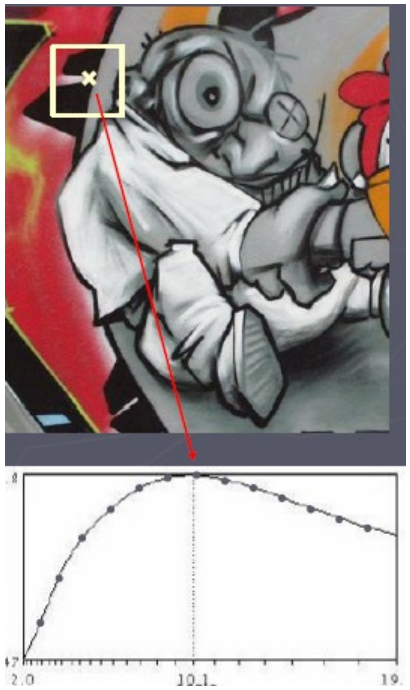


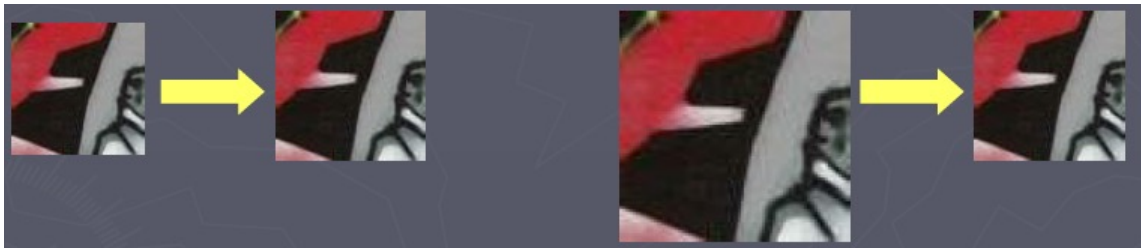
Robustness to Spatial Scaling (Example)



Robustness to Spatial Scaling (Example)

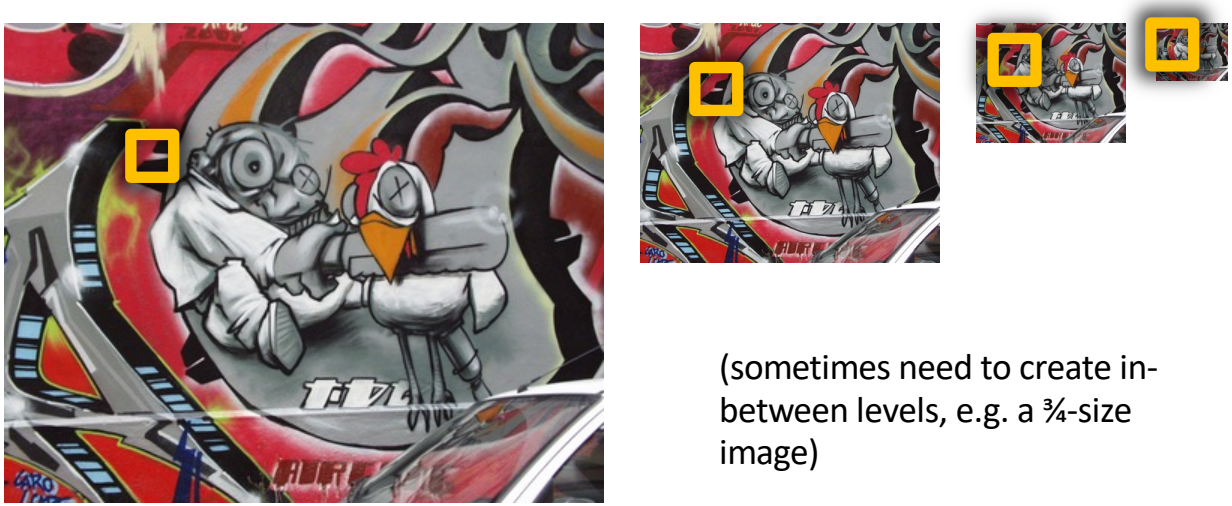






Implementation

Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid



(sometimes need to create in-between levels, e.g. a ¾-size image)

SCC.366: Media Coding & Processing

2022-2023

Week 08 – Lecture 2

Local Image Matching Cont.

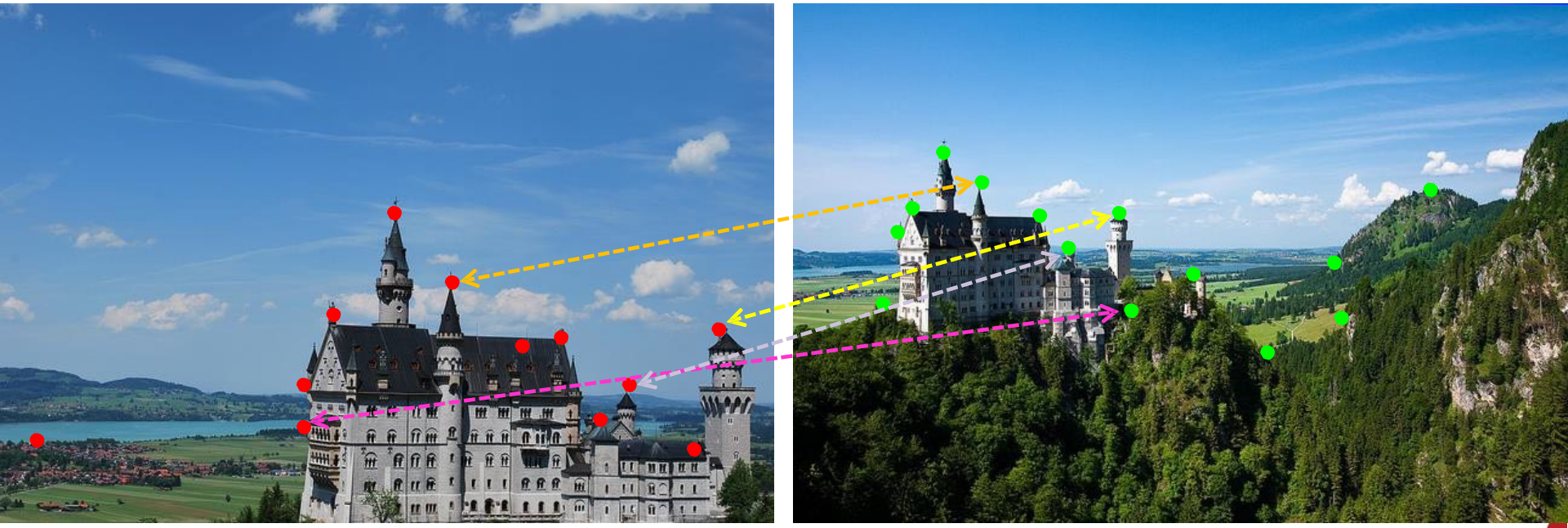
Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- What is Image matching?
- Applications of image matching
- Challenges in image matching
- Holistic vs. local image matching
- Holistic image matching an its limitations
  - Intensity based descriptor
  - Feature based descriptor e.g., HOG
- Local image matching (This Lecture)
  - Interest point detection
  - Interest point description
  - Matching process
- Summary of the main points

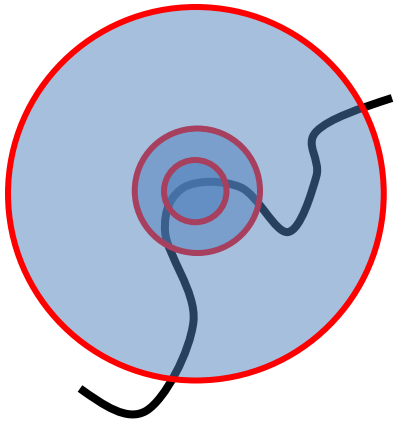
Local Image Matching (Definition)

- Image matching is the process of establishing a correspondence between a pair of images



Robustness to Spatial Scaling

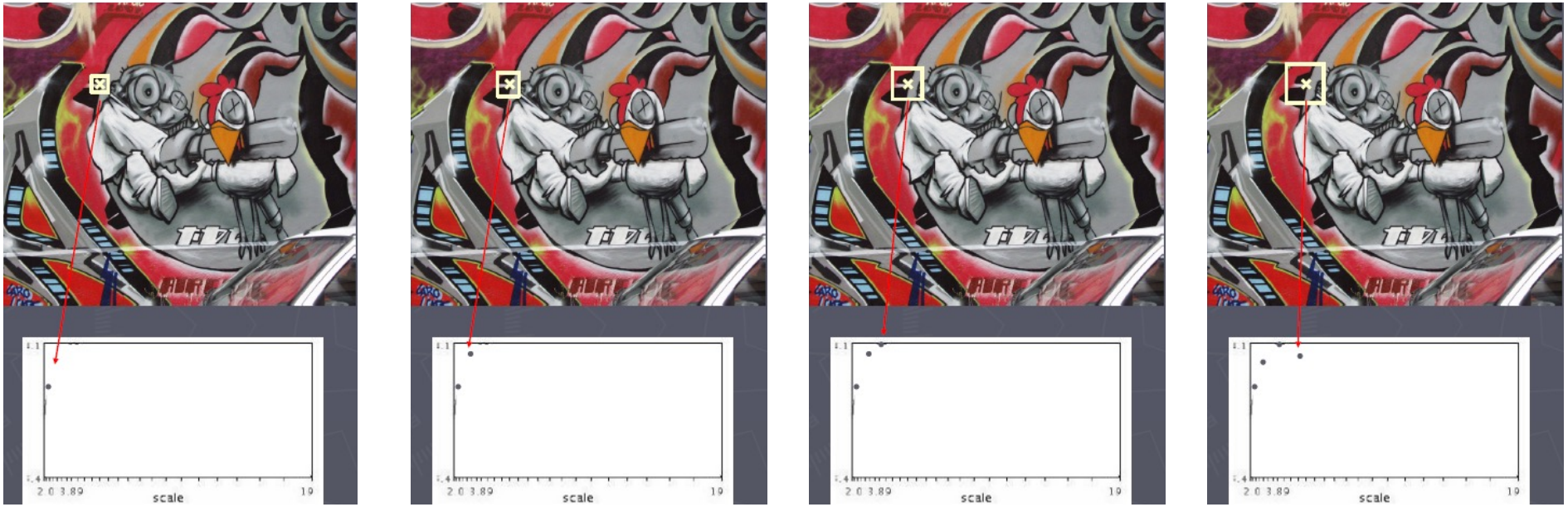
- Suppose you're looking for corners in the below image



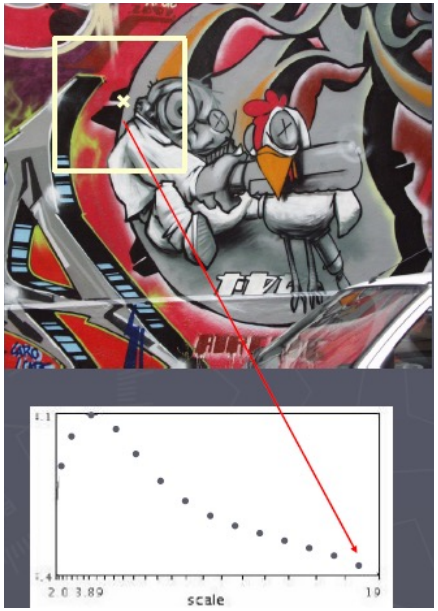
Key idea: find scale that gives local maximum of f in both position and scale

One definition of f: the Harris operator

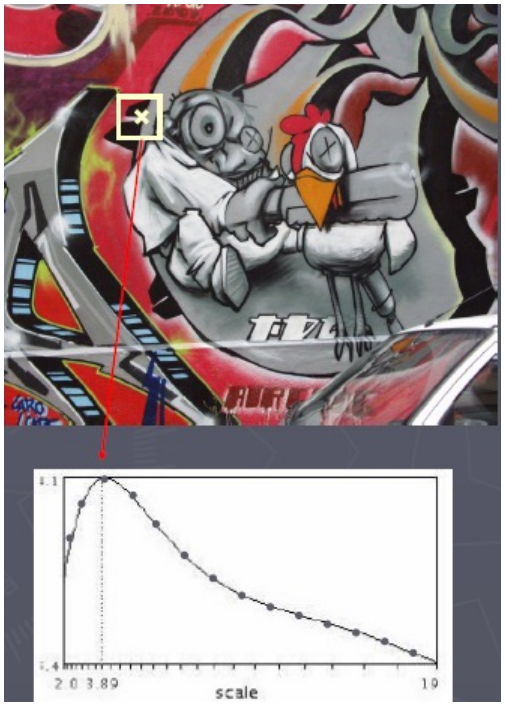
Robustness to Spatial Scaling (Example)

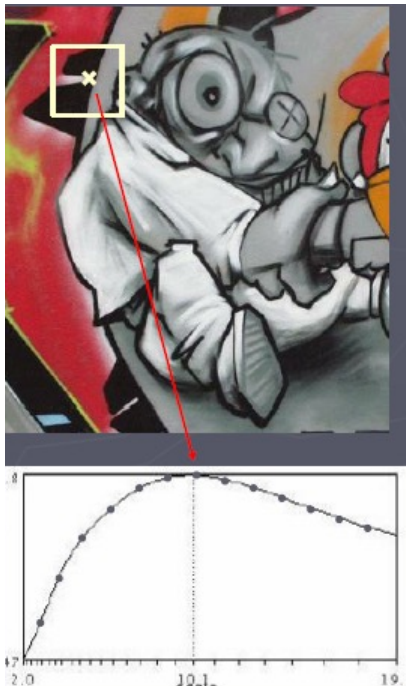


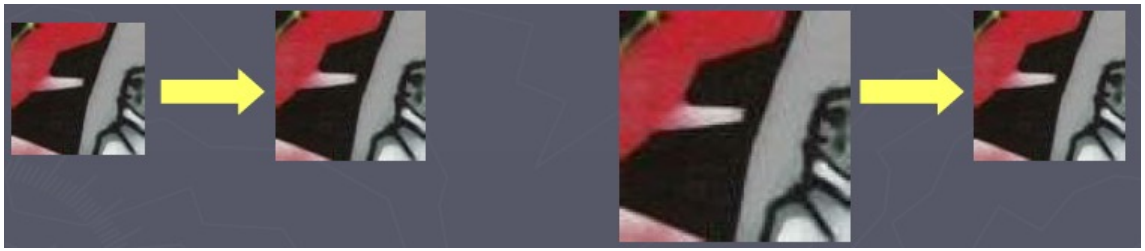
Robustness to Spatial Scaling (Example)



Robustness to Spatial Scaling (Example)

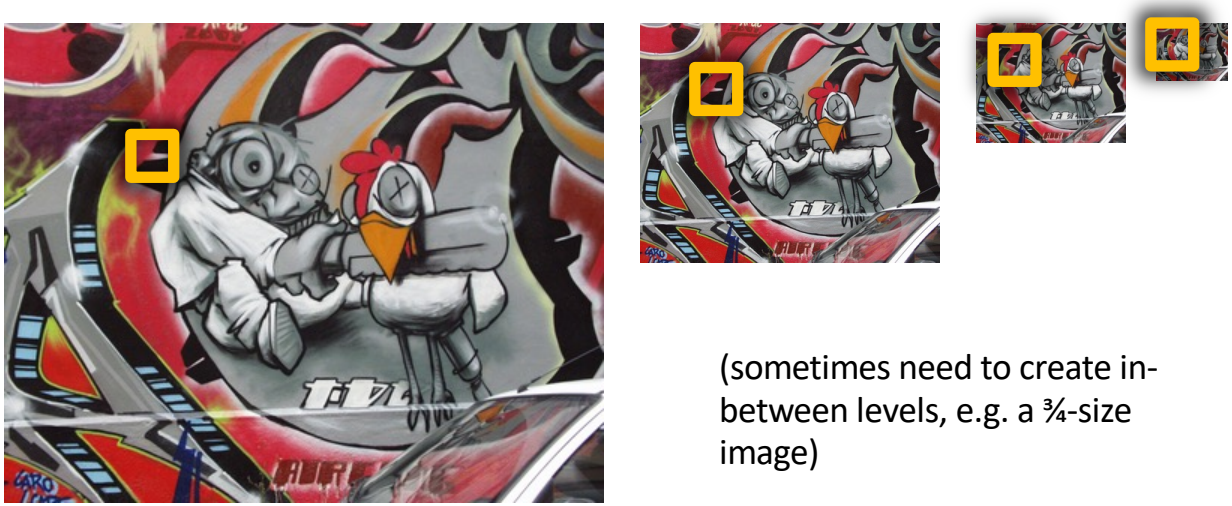






Implementation

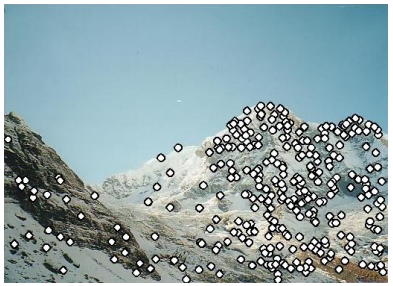
Instead of computing *f* for larger and larger windows, we can implement using a fixed window size with a Gaussian pyramid



(sometimes need to create in-between levels, e.g. a ¾-size image)

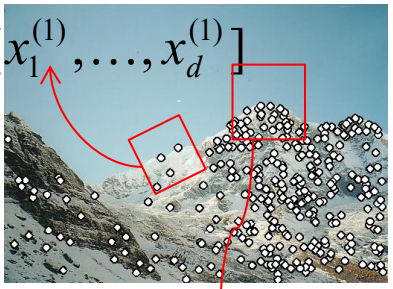
Local Image Matching (main components)

1) **Detection**: Identify the interest points, e.g., Harris corner and SIFT



2) **Description**: Extract feature descriptor surrounding each interest point, e.g., HOG, SIFT

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching**: Determine correspondence between descriptors in two views

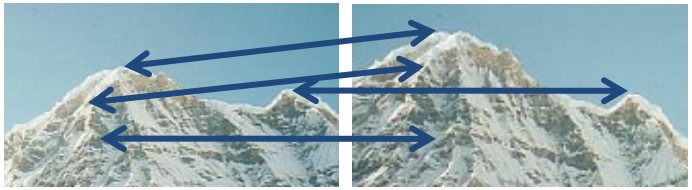
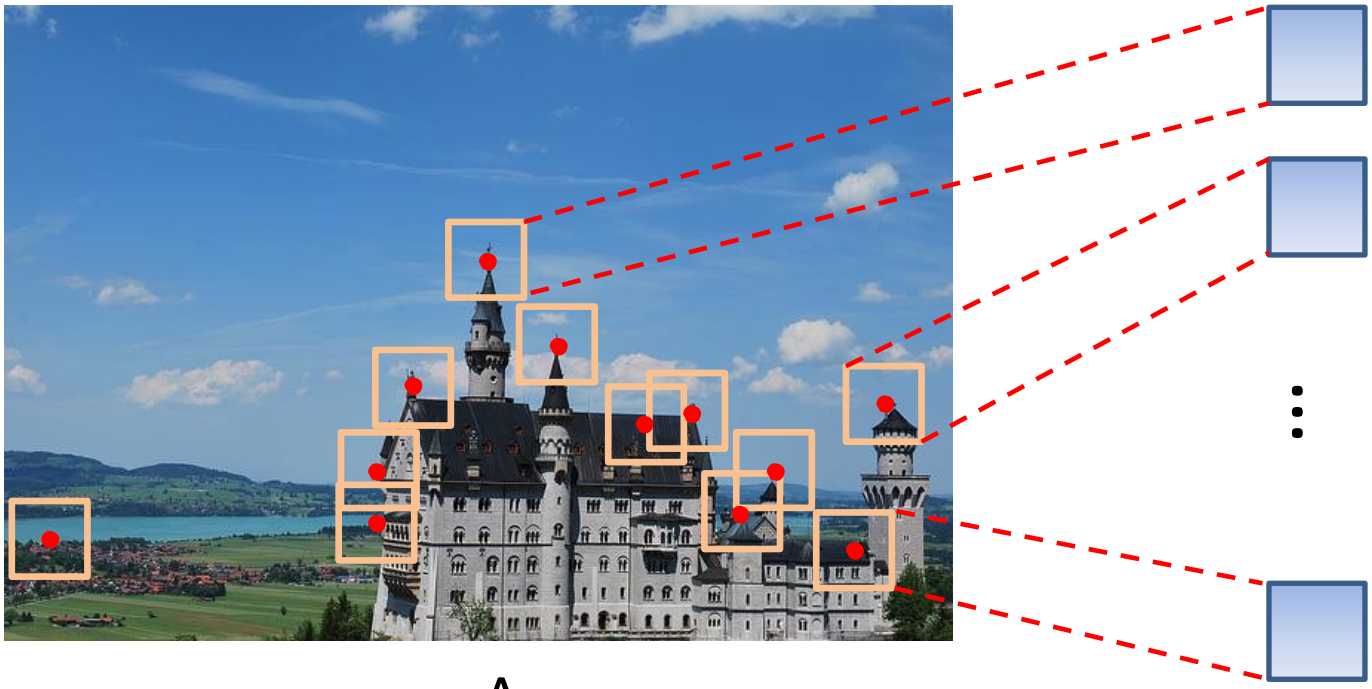


Image matching with scale variation

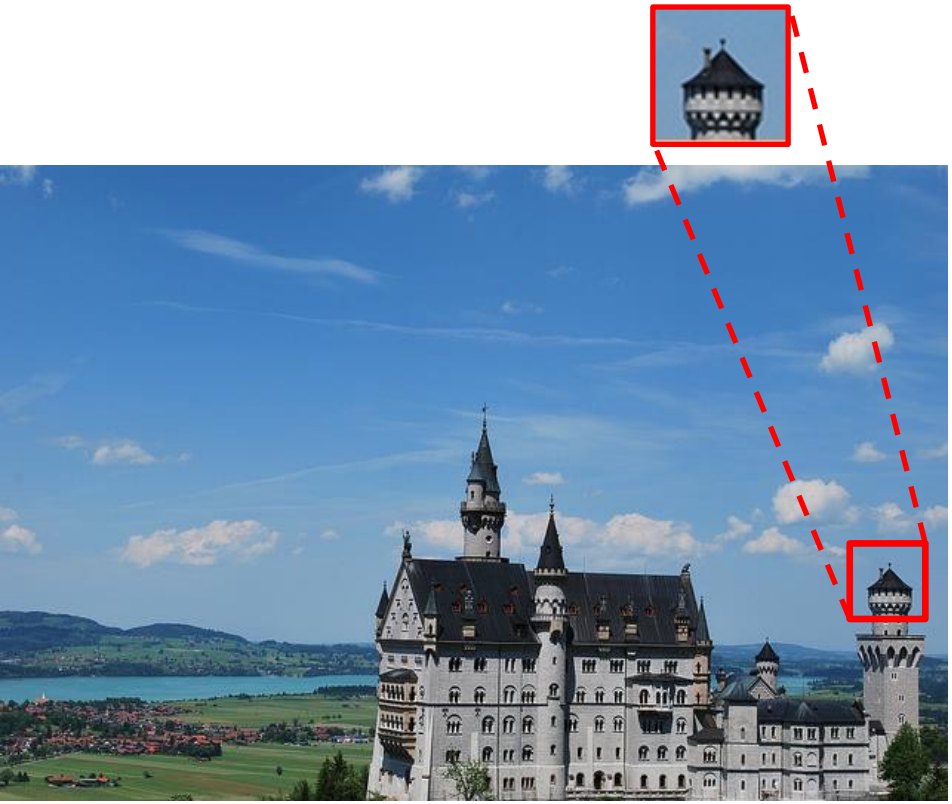
- Extracting a fixed-size (e.g., 16×16) image patch from each feature location.
- Extract HoG from each patch as a descriptor



A

Image matching with scale variation

- This won’t apply if the scales of the objects of interest in two images are different.



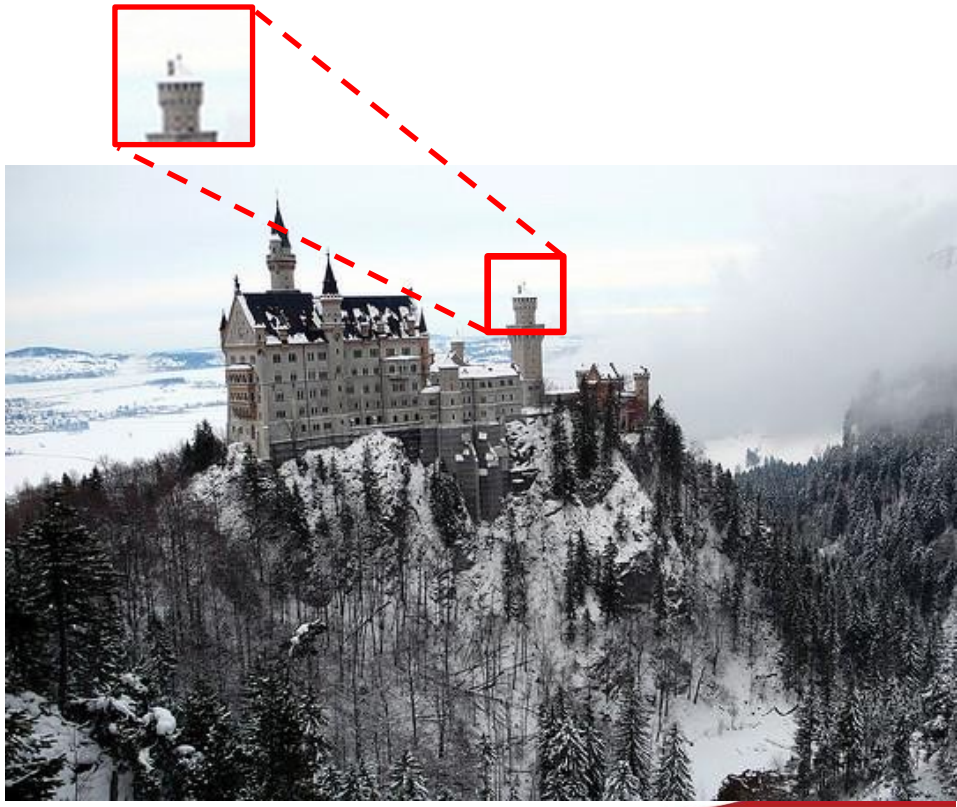


Image matching with scale variation

- Instead, we can extract a bigger patch from the left image and scale it down.

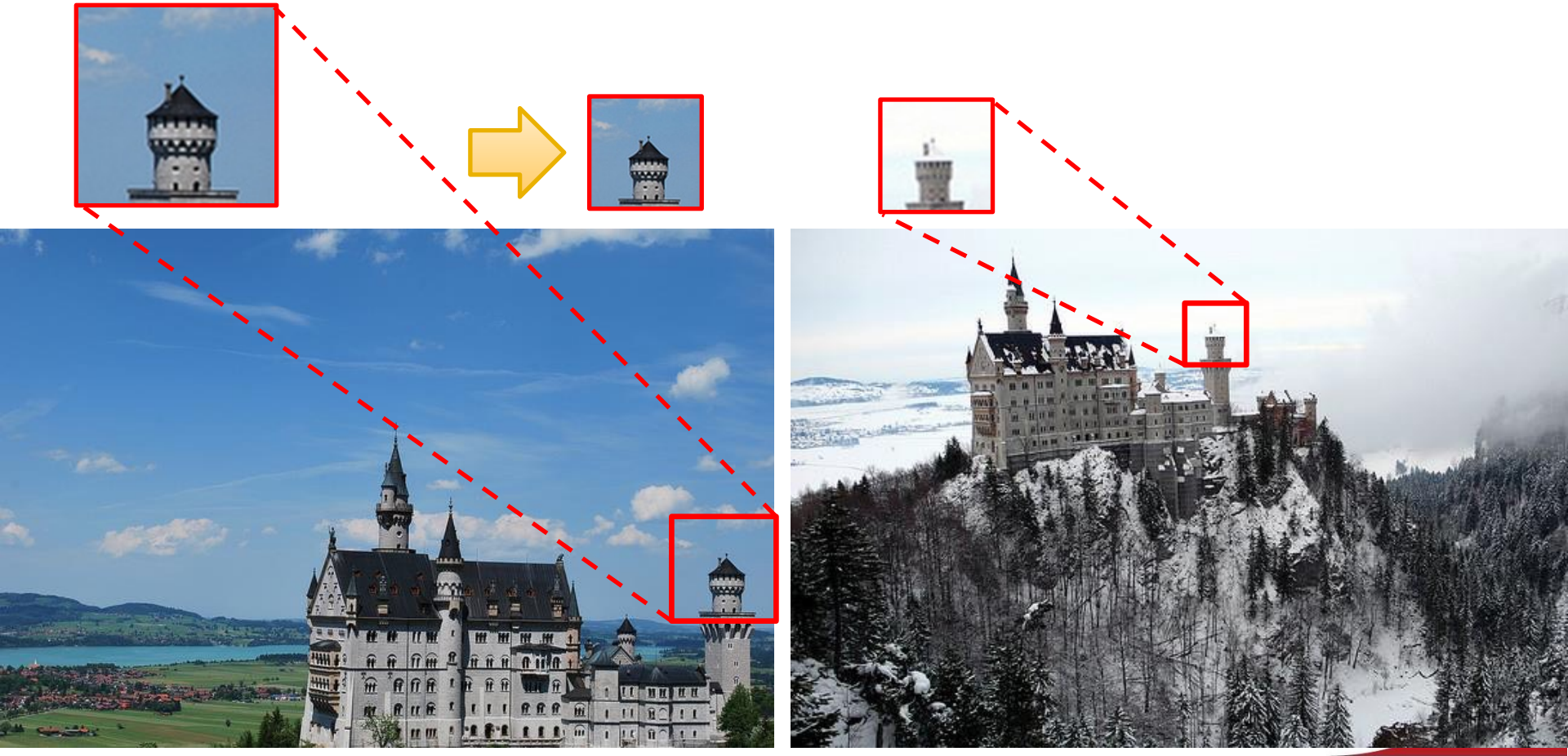


Image matching with scale variation

- This requires determining a proper patch size for each feature.

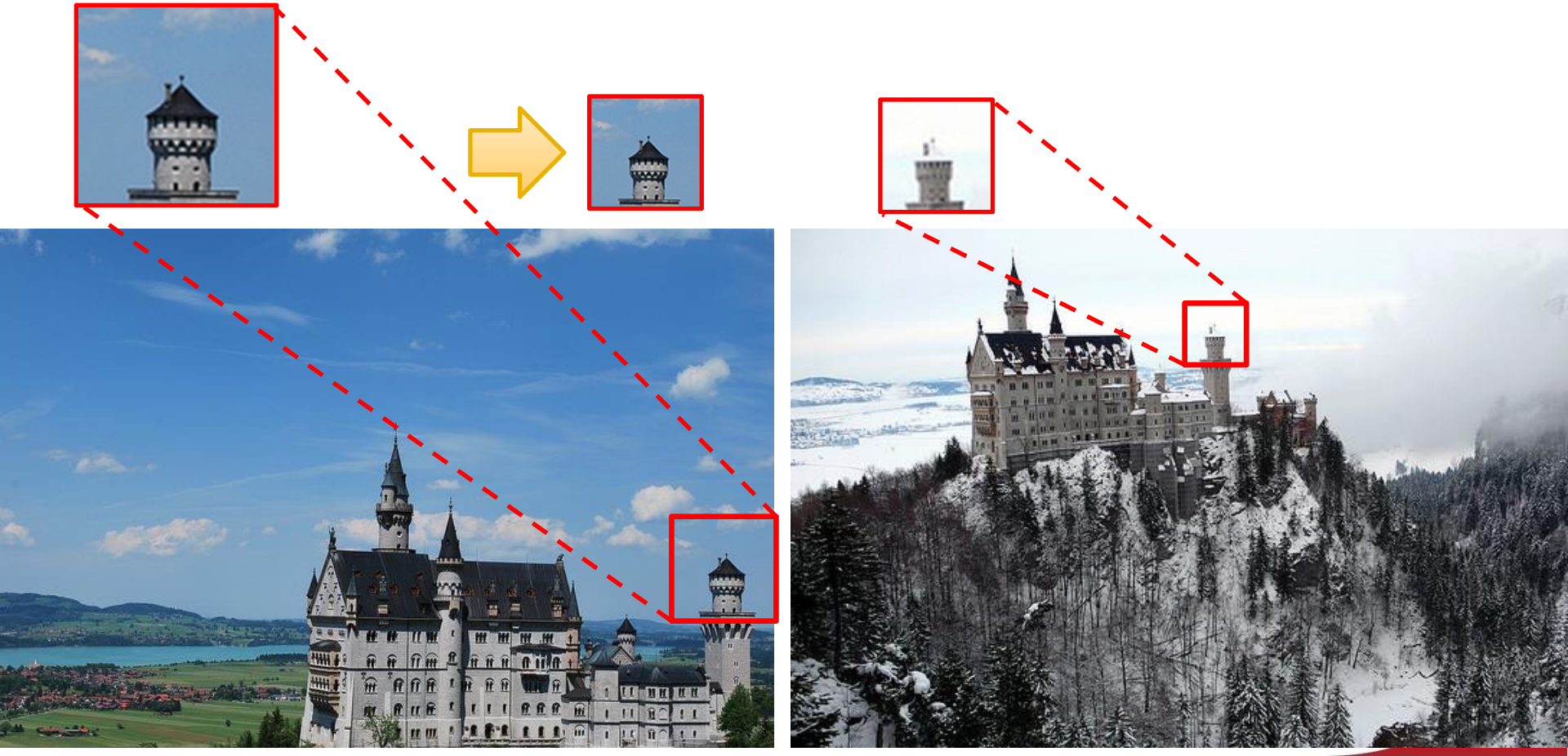
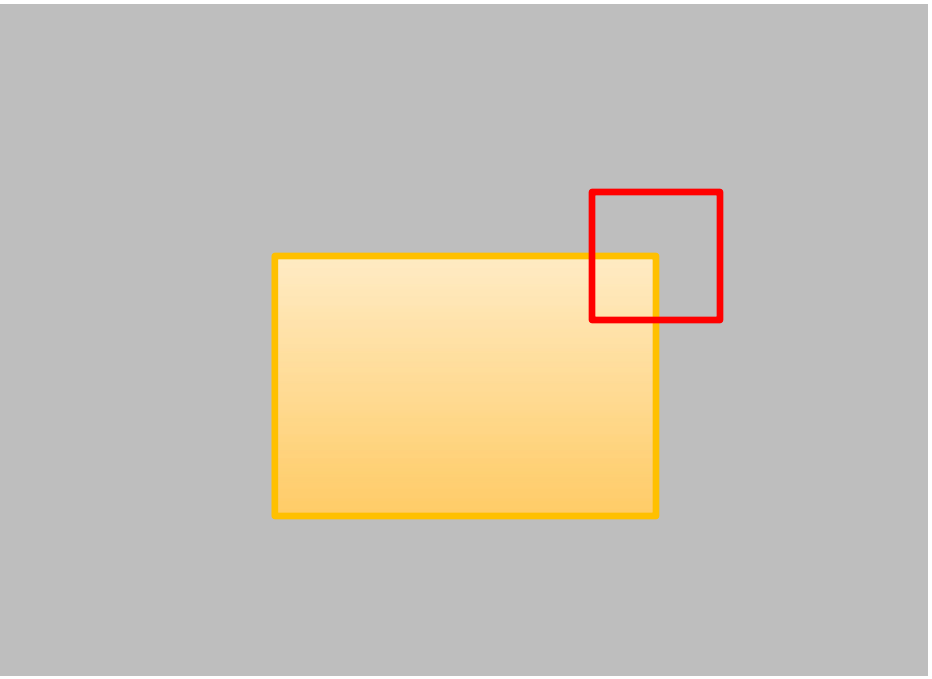


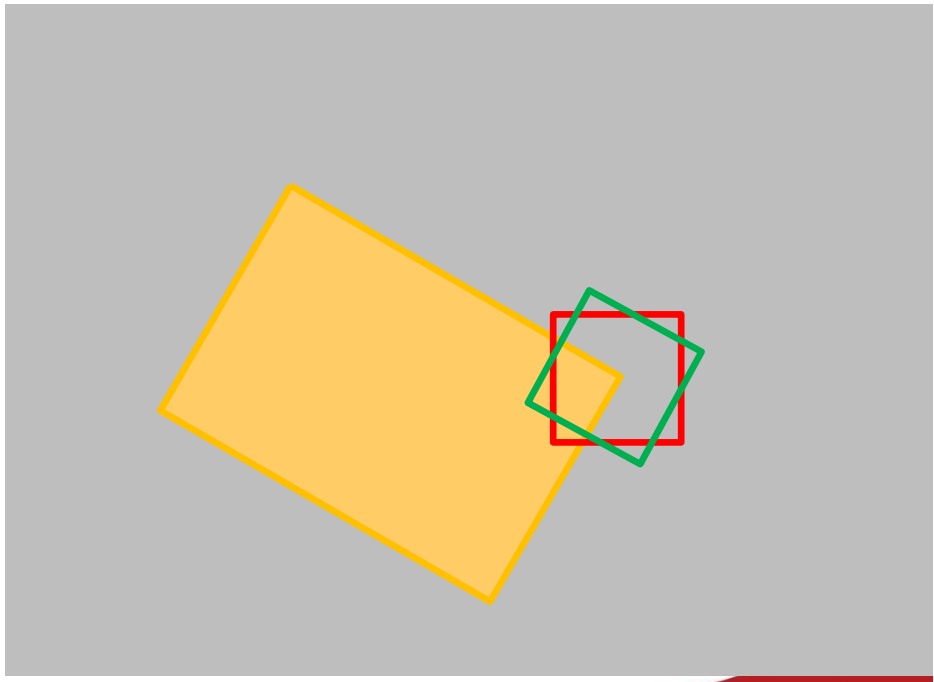
Image matching with orientation variation

- So far, we assumed that the orientations of the objects in images are roughly fixed.
- This may not be true in practice.

Image matching with orientation variation

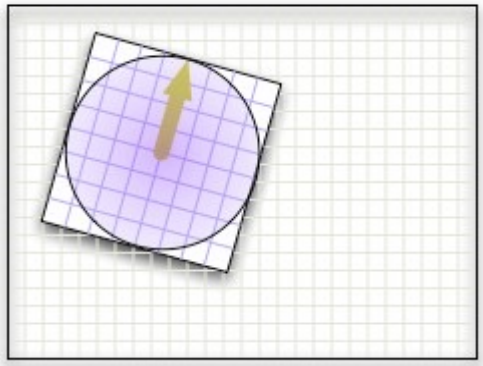
- Similarly to image scaling, we can determine proper orientations and align each patch orientation before matching.

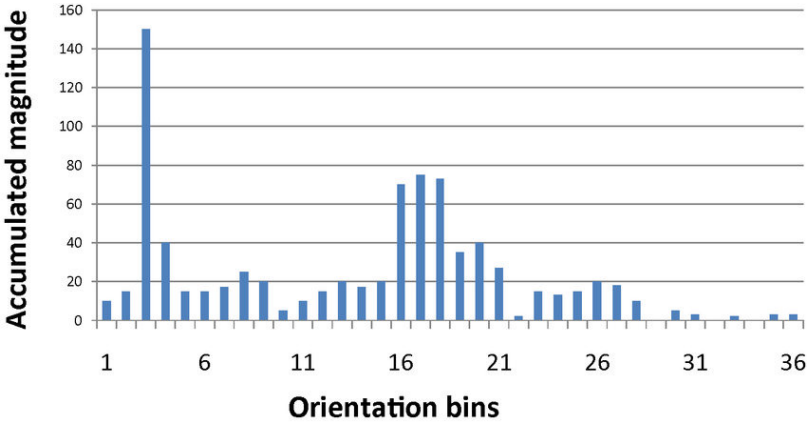




SIFT Descriptor

- An orientation histogram is formed from the gradient orientations of sample points within a region around the keypoint
- The orientation histogram has 36 bins covering the 360 degree range of orientations
- Each sample added to the histogram is weighted by its gradient magnitude
- Peaks in the orientation histogram correspond to dominant directions of local gradients
  - The highest peak in the histogram is detected, and then any other local peak that is within 80% of the highest peak is used to also create a keypoint with that orientation.
  - Therefore, for locations with multiple peaks of similar magnitude, there will be multiple keypoints created at the same location and scale but different orientations.





SIFT Descriptor

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

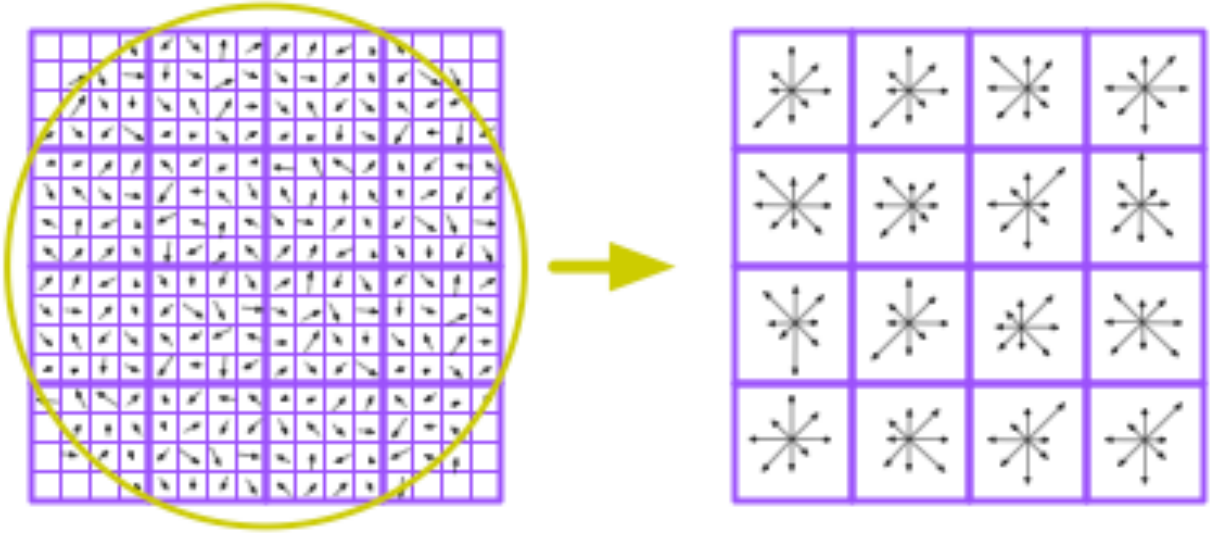
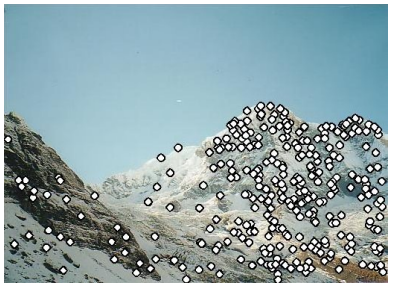


Image gradients

Keypoint descriptor

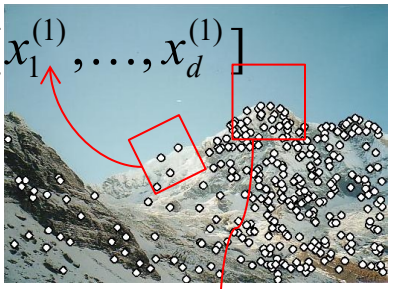
Local Image Matching (main components)

1) **Detection**: Identify the interest points, e.g., Harris corner and SIFT



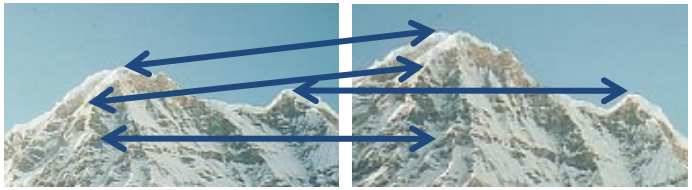
2) **Description**: Extract feature descriptor surrounding each interest point, e.g., HOG, SIFT

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



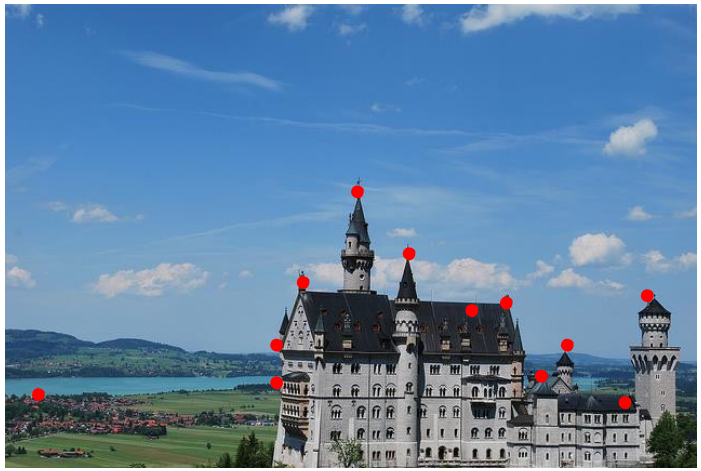
$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

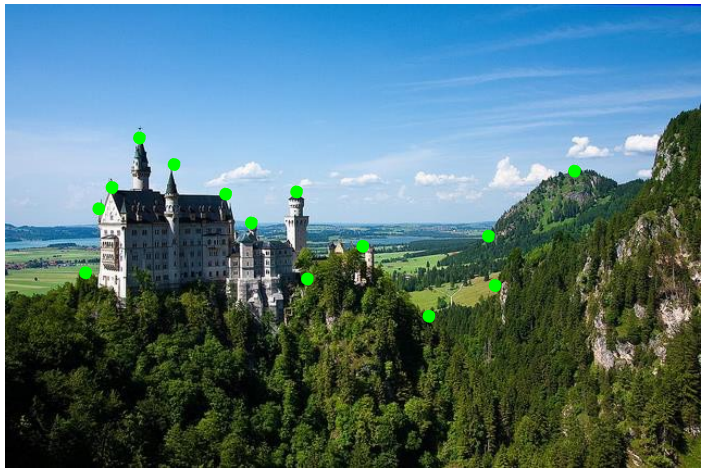
3) **Matching**: Determine correspondence between descriptors in two views



Feature-based Image matching

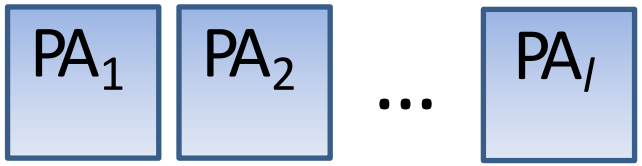
- Each input image is converted to a set of small (patch) images

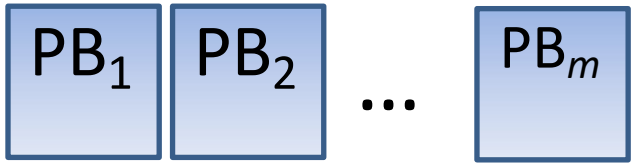




A

B





l keypoints

m keypoints

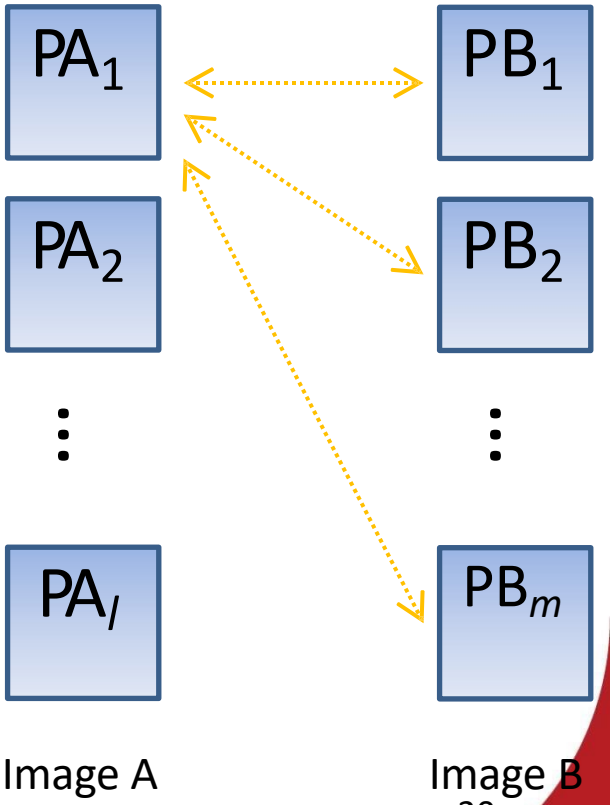
Feature-based Image matching

- Calculate the SSDs of PA_1 and all patches of B.
- Choose the best and the second best matching patches (say PB_i and PB_j): PB_i and PB_j have minimum distances to PA_1.
- Check if
  - $d(PA_1,PB_i) < 0.8 \times d(PA_1,PB_j)$
  - $d(PA_1,PB_i) < \tau$
- If true, PA_1 and Pb_i are declared as match.

$$d(PA_1, PB_1)$$

$$d(PA_1, PB_2)$$

$$d(PA_1, PB_m)$$



τ: user-defined parameter

SCC.366: Media Coding & Processing

2022-2023

Week 08 – Lecture 2

Image Pyramid

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- Image down-sampling
- Aliasing
- Gaussian image pyramid
- Laplacian image pyramid
- Applications of image pyramid
- Summary of the main points

Image down-sampling

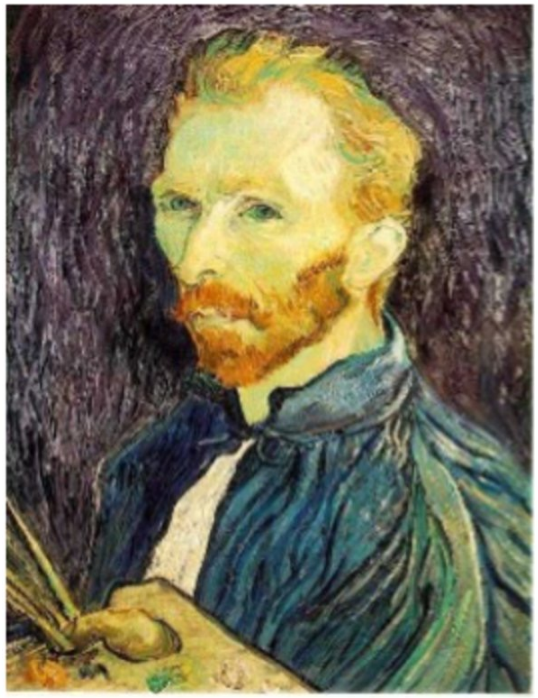
Naïve Image down-sampling

- Assume that the image is too big for practical use:
  - Requires too much memory
  - Time consuming to process
  - Too big for the screen
  - …

- A smaller image can be obtained by image subsampling

Naïve Image down-sampling

- Throw away half the rows and columns



1/2

Delete even rows
Delete even columns

1/4

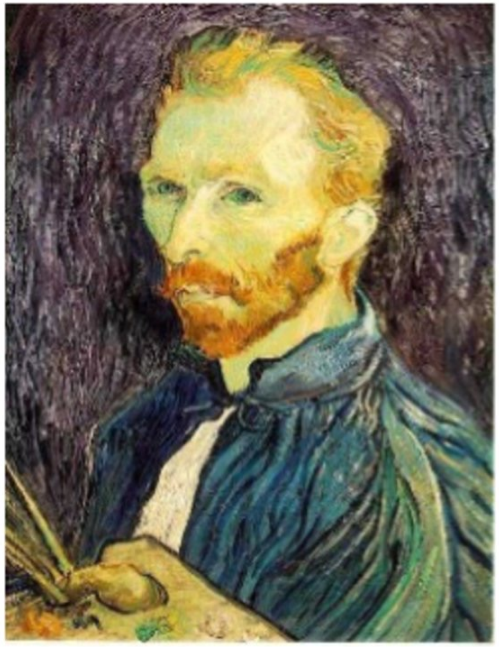
Delete even rows
Delete even columns

1/8

- What is the problem with this approach?

Naïve Image down-sampling

- The 1/8 image is so pixelated (this artifact also called **aliasing**)



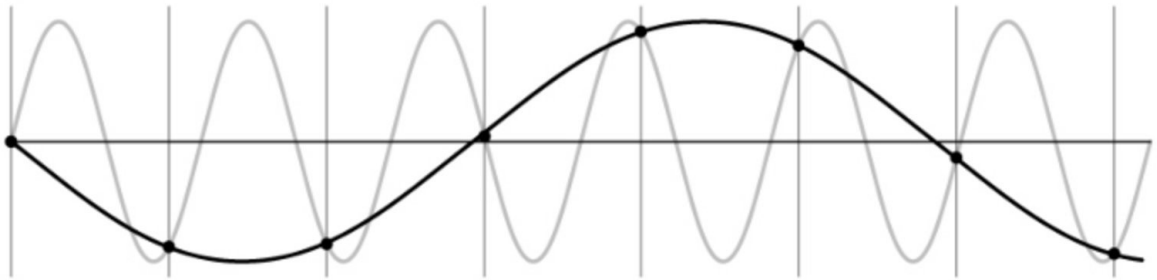
1/2

1/4 (2x zoom)

1/8 (4x zoom)

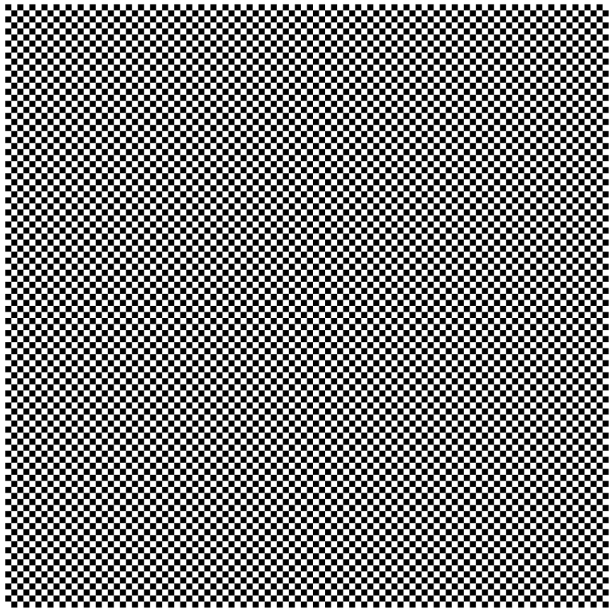
Aliasing

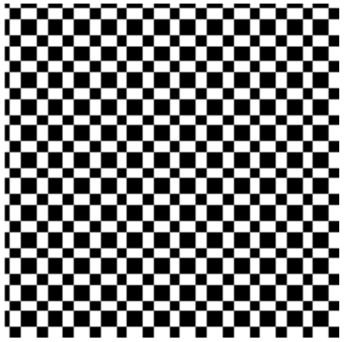
- Aliasing occurs when the (spatial) sampling rate is not high enough to capture the details in the image
- High frequencies are transformed to lower frequencies (i.e. aliases)
- To avoid aliasing the sampling rate must be at least two times the maximum frequency in the image (at least two samples per cycle)
- This minimum sampling rate is called the Nyquist rate



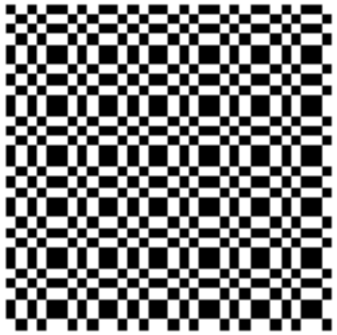
Aliasing: Spatial Under-sampling

- High frequencies are transformed to lower frequencies (this artifact is called **aliasing**)



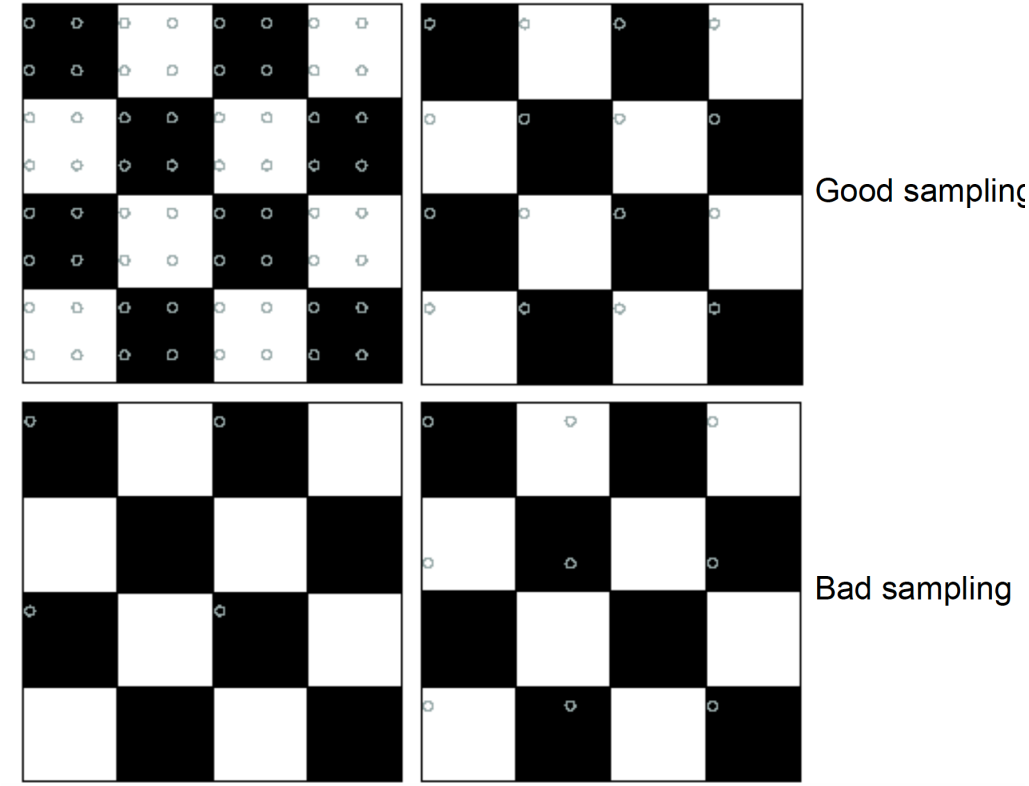


1/10

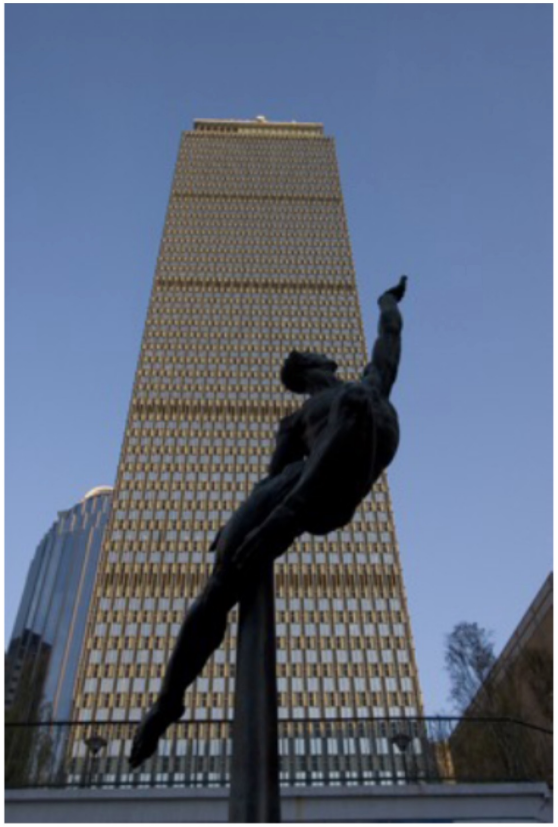


1/16

Aliasing: Spatial Under-sampling

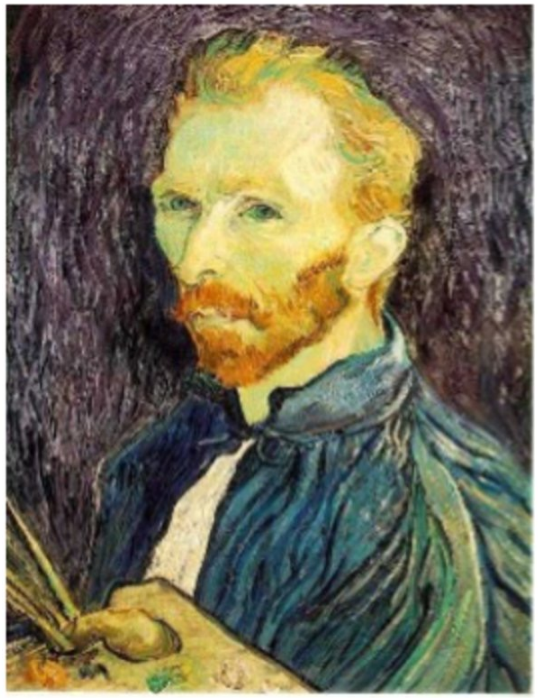


Aliasing: Spatial Under-sampling



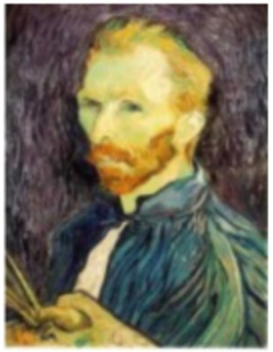
Better Image down-sampling

- Apply a smoothing filter first (e.g., Gaussian filter), then throw away half the rows and columns



1/2

Gaussian filter
Delete even rows
Delete even columns



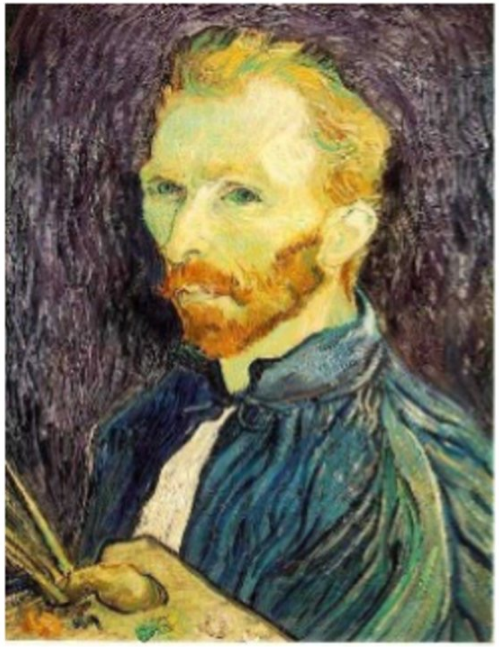
1/4

Gaussian filter
Delete even rows
Delete even columns

1/8

Better Image down-sampling

- The 1/8 image is much better than the Naïve one



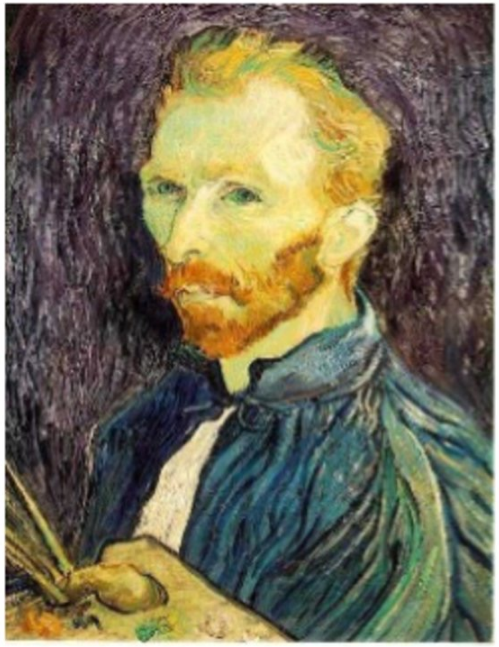
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Naïve Image down-sampling

- The 1/8 image is so pixelated



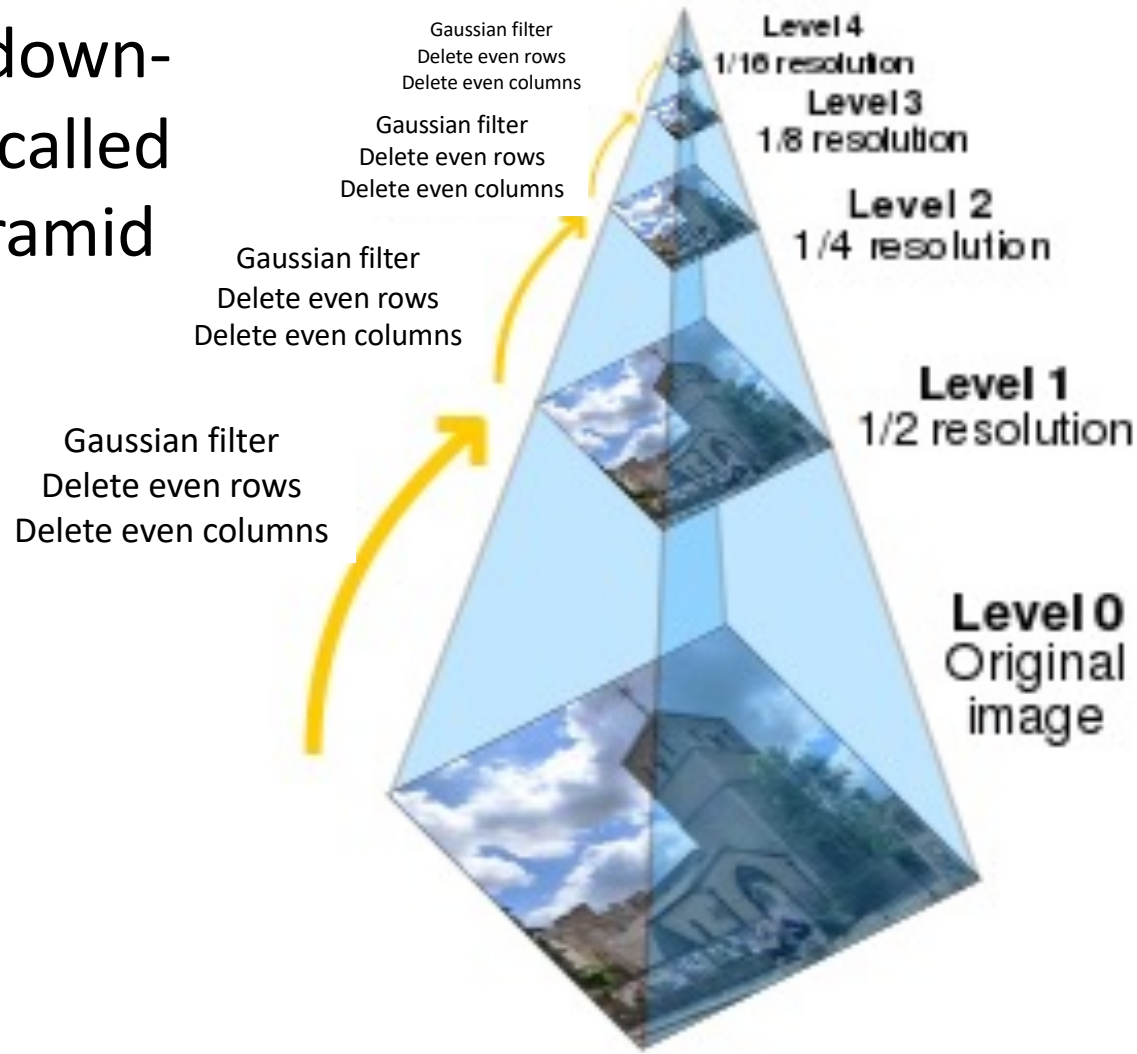
1/2

1/4 (2x zoom)

1/8 (4x zoom)

Gaussian Image Pyramid

- The sequence of down-sampled images is called Gaussian Image Pyramid



Properties of Gaussian Image Pyramid

What happens to the details of the image?

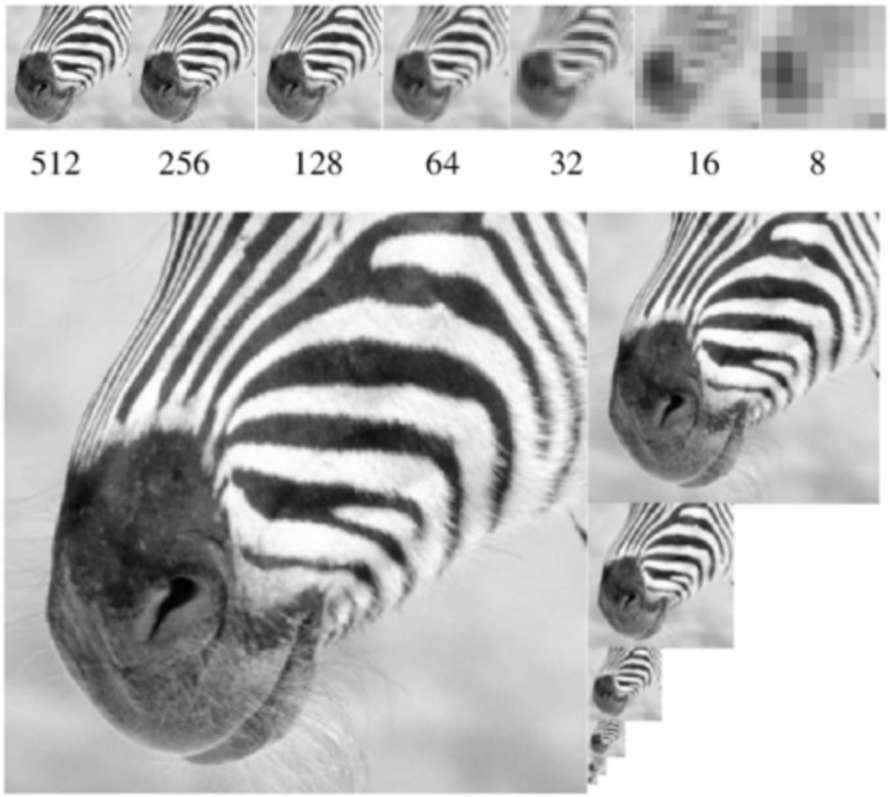
- They get smoothed out as we move to higher levels

What is preserved at the higher levels?

- Mostly large uniform regions in the original image

How would you reconstruct the original image from the image at the upper level?

- That is not possible



Blurring is Lossy

- What does the residual look like?

level 0

level 1 (before downsampling)

Blurring is Lossy

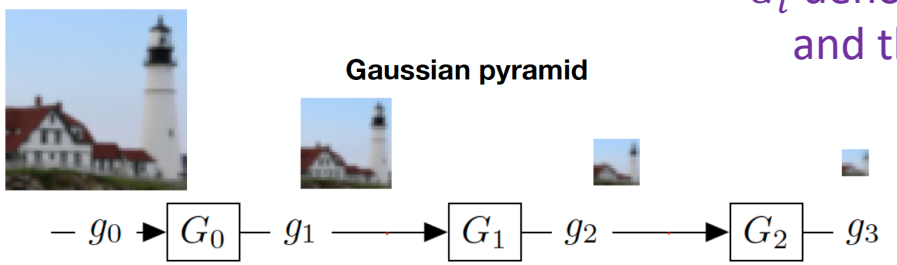
- What does the residual look like?

level 0 - level 1 (before downsampling) = residual

From Gaussian to Laplacian Image Pyramid

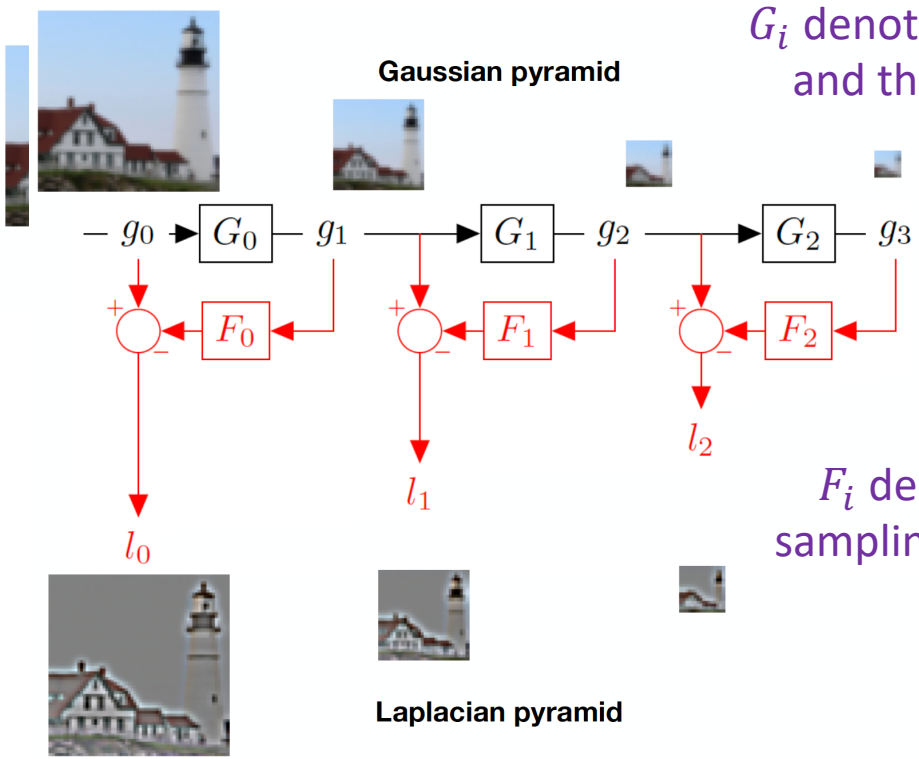
- At each level, retain the residuals instead of the blurred images themselves

G_i denotes applying Gaussian and then down-sampling



Laplacian Image Pyramid

- At each level, retain the residuals instead of the blurred images themselves

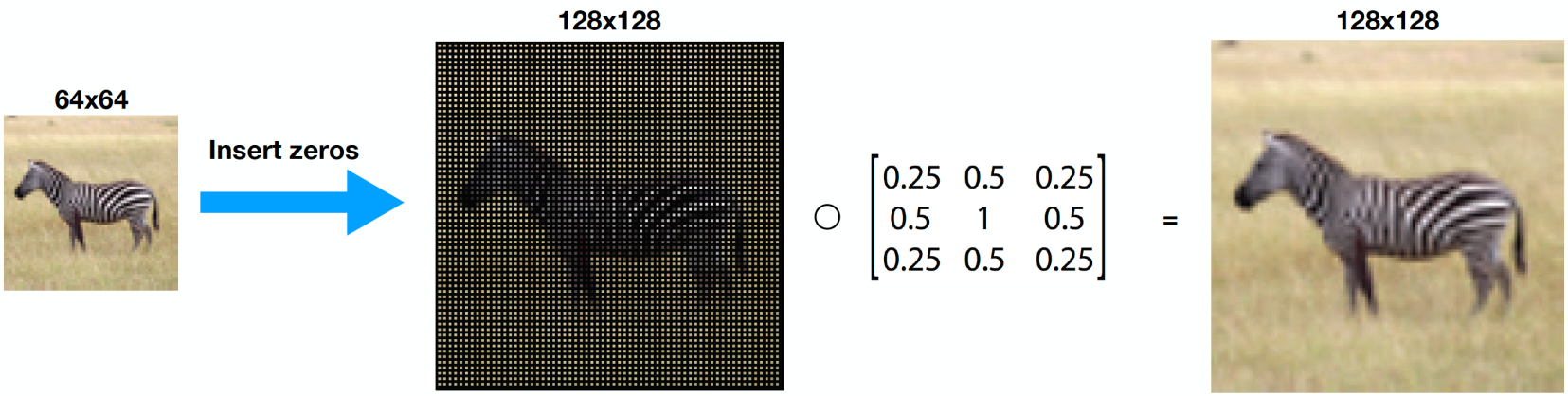


G_i denotes applying Gaussian and then down-sampling

F_i denotes applying up-sampling and then Gaussian

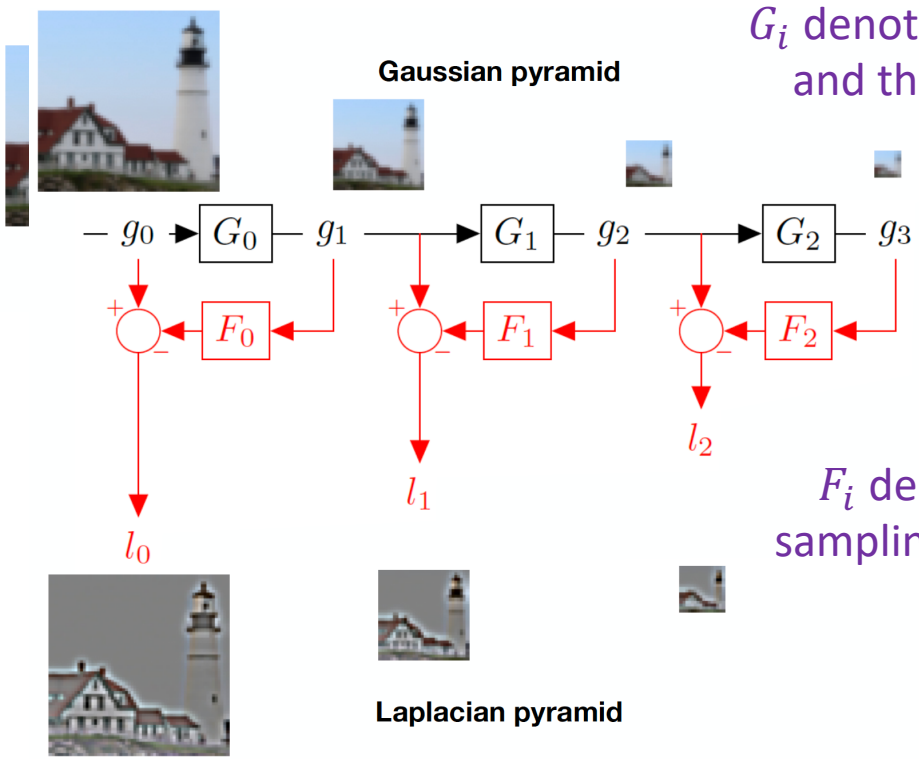
Up-sampling and Gaussian

- At each level, retain the residuals instead of the blurred images themselves



Laplacian Image Pyramid

- What do we need to store to be able to reconstruct the original image?

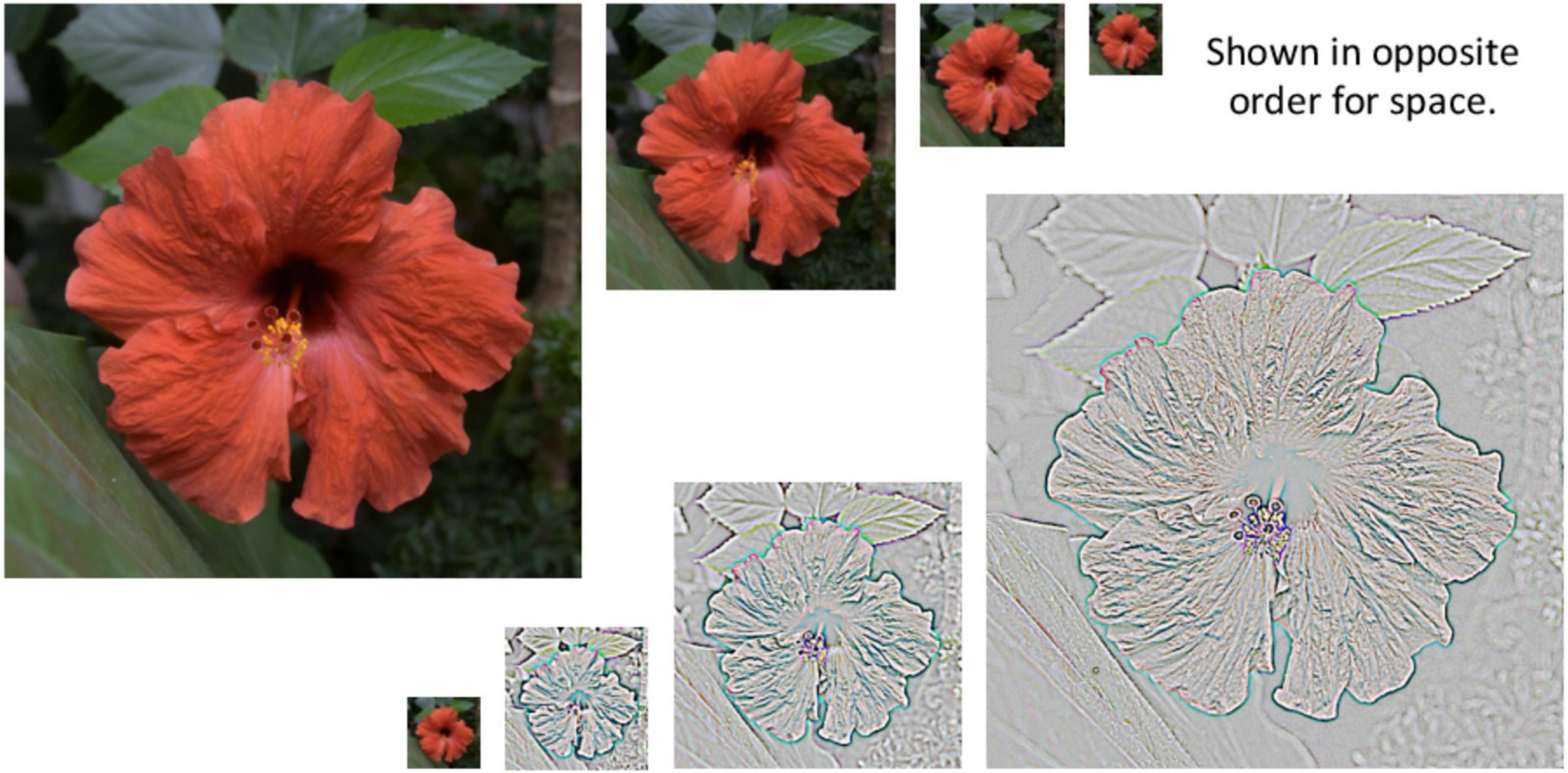


G_i denotes applying Gaussian and then down-sampling

F_i denotes applying up-sampling and then Gaussian

Gaussian vs Laplacian

- Which one takes more space to store?



Applications of Image Pyramid

image compression

multi-scale texture mapping

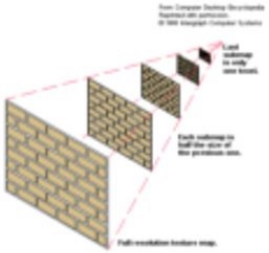


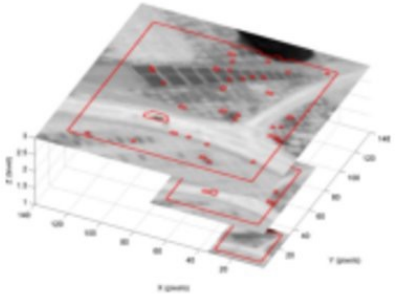
image blending

focal stack compositing

denoising

multi-scale detection

multi-scale registration



SCC.366: Media Coding & Processing

2022-2023

Week 09 - Lecture 1

Optical Flow

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

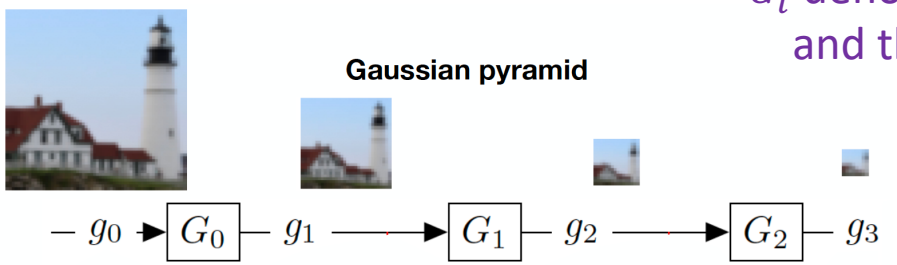
Outline – Last week's lecture

- Gaussian image pyramid
- Laplacian image pyramid

From Gaussian to Laplacian Image Pyramid

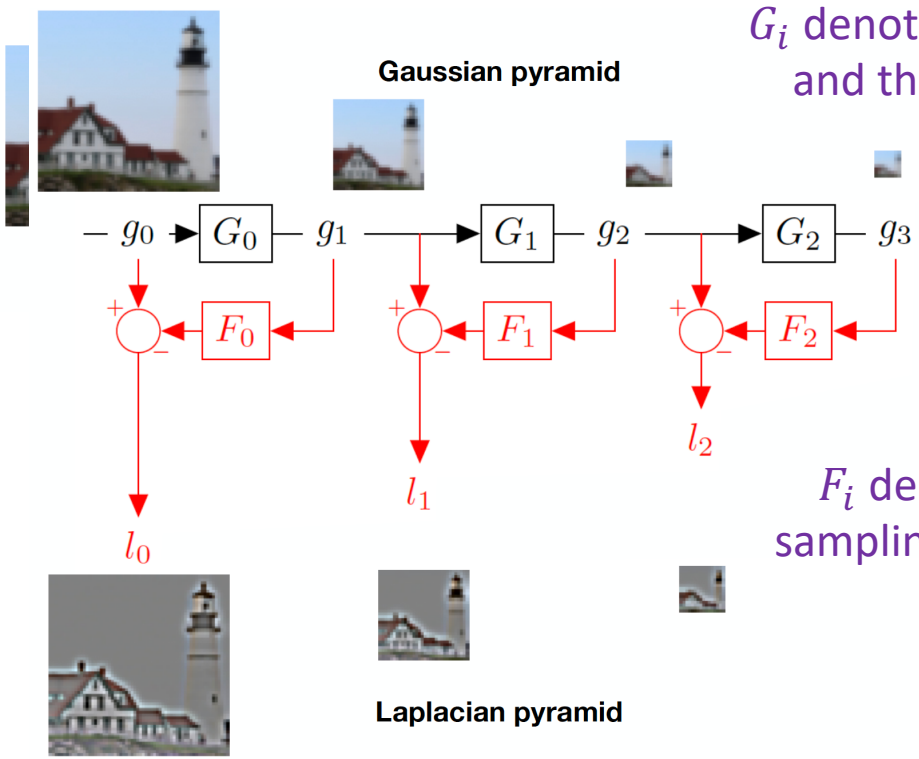
- At each level, retain the residuals instead of the blurred images themselves

G_i denotes applying Gaussian and then down-sampling



Laplacian Image Pyramid

- At each level, retain the residuals instead of the blurred images themselves

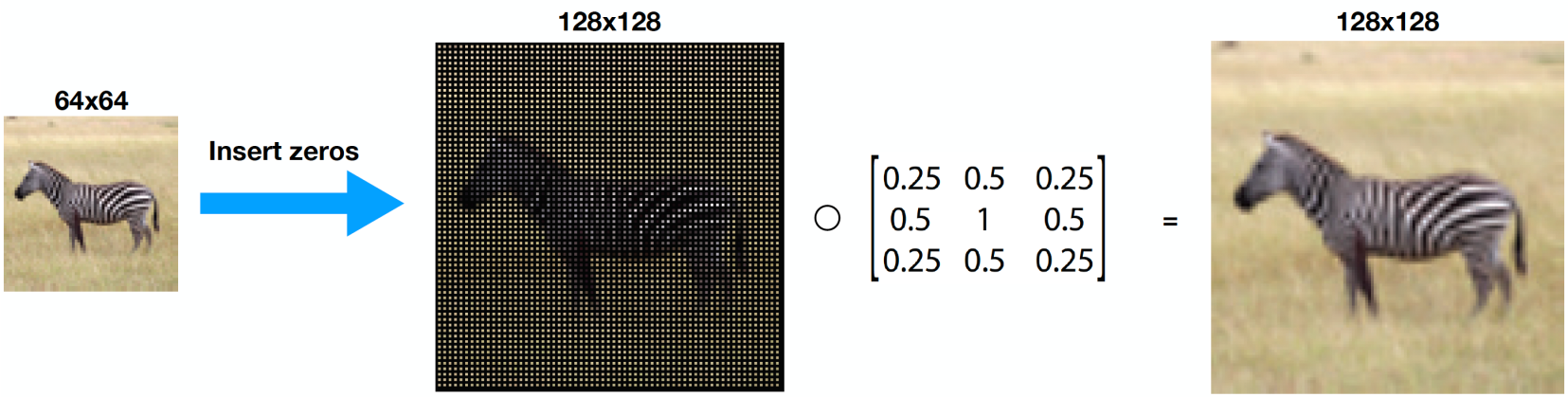


G_i denotes applying Gaussian and then down-sampling

F_i denotes applying up-sampling and then Gaussian

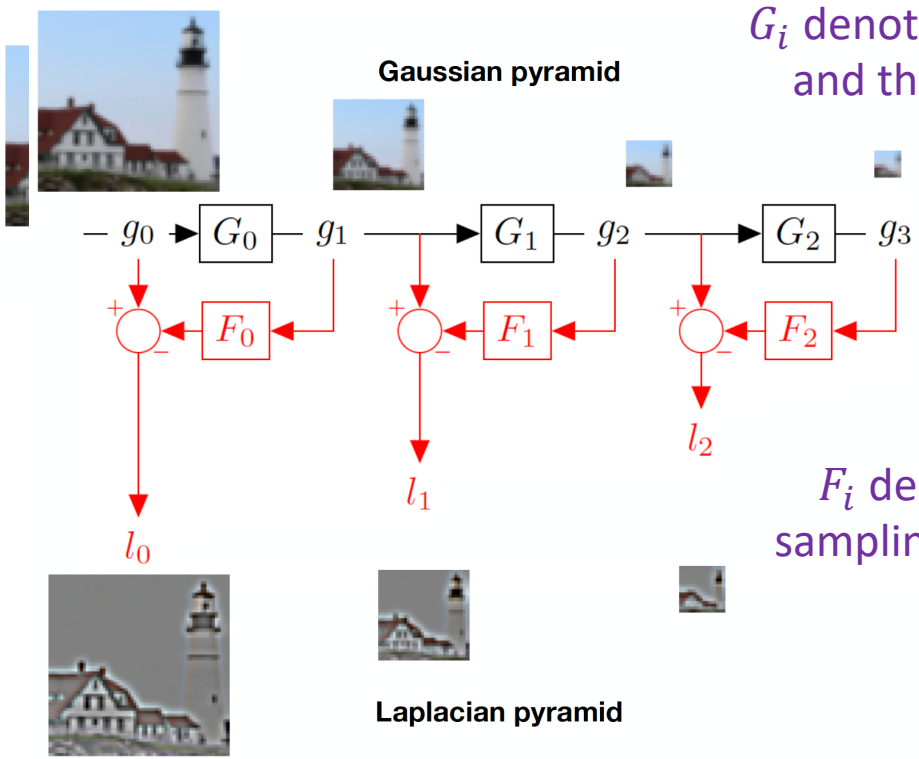
Up-sampling and Gaussian

- At each level, retain the residuals instead of the blurred images themselves



Laplacian Image Pyramid

- What do we need to store to be able to reconstruct the original image?

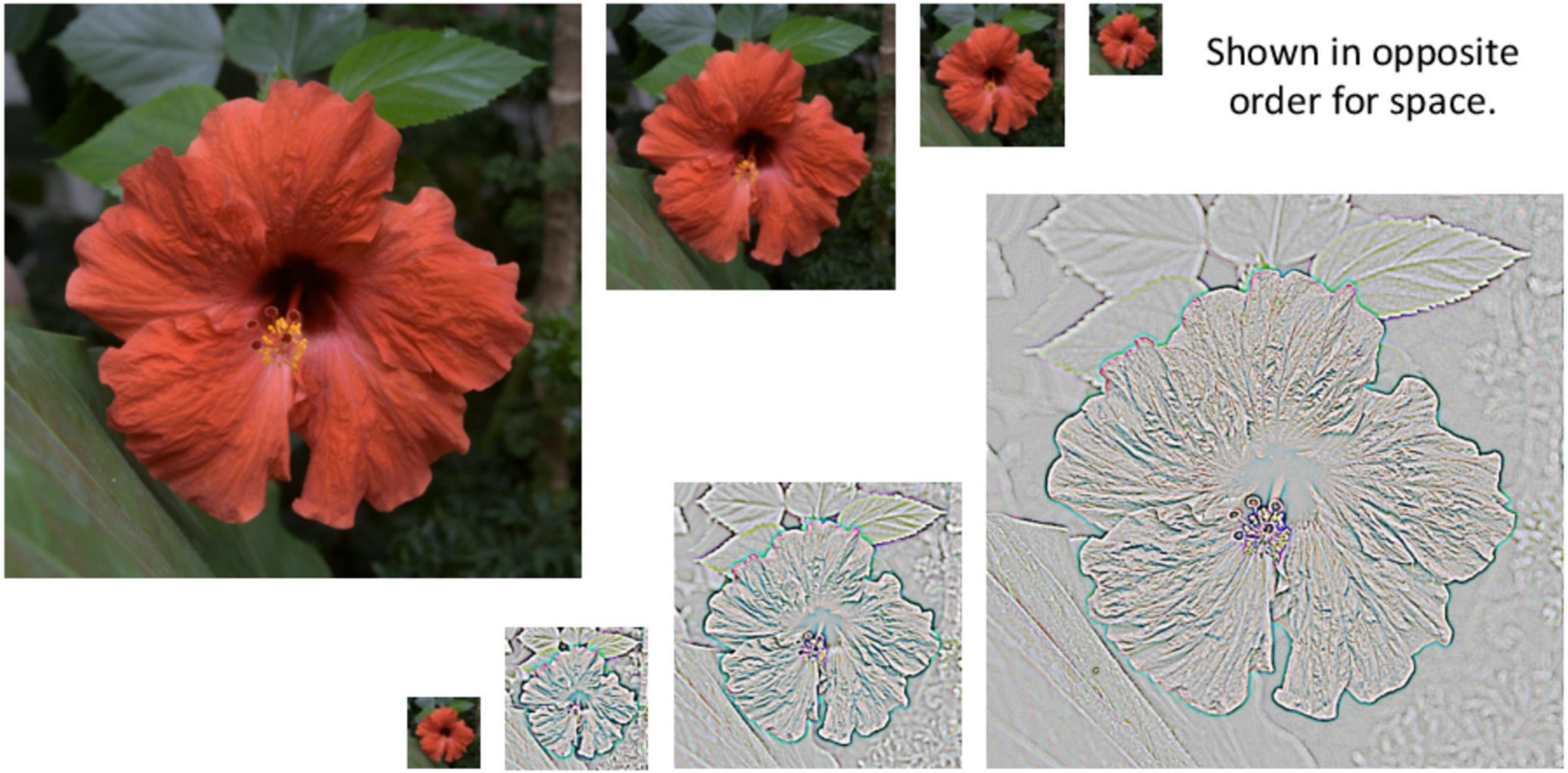


G_i denotes applying Gaussian and then down-sampling

F_i denotes applying up-sampling and then Gaussian

Gaussian vs Laplacian

- Which one takes more space to store?



Applications of Image Pyramid

image compression

multi-scale texture mapping

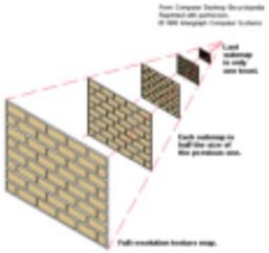


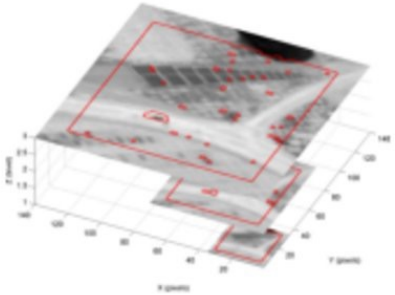
image blending

focal stack compositing

denoising

multi-scale detection

multi-scale registration

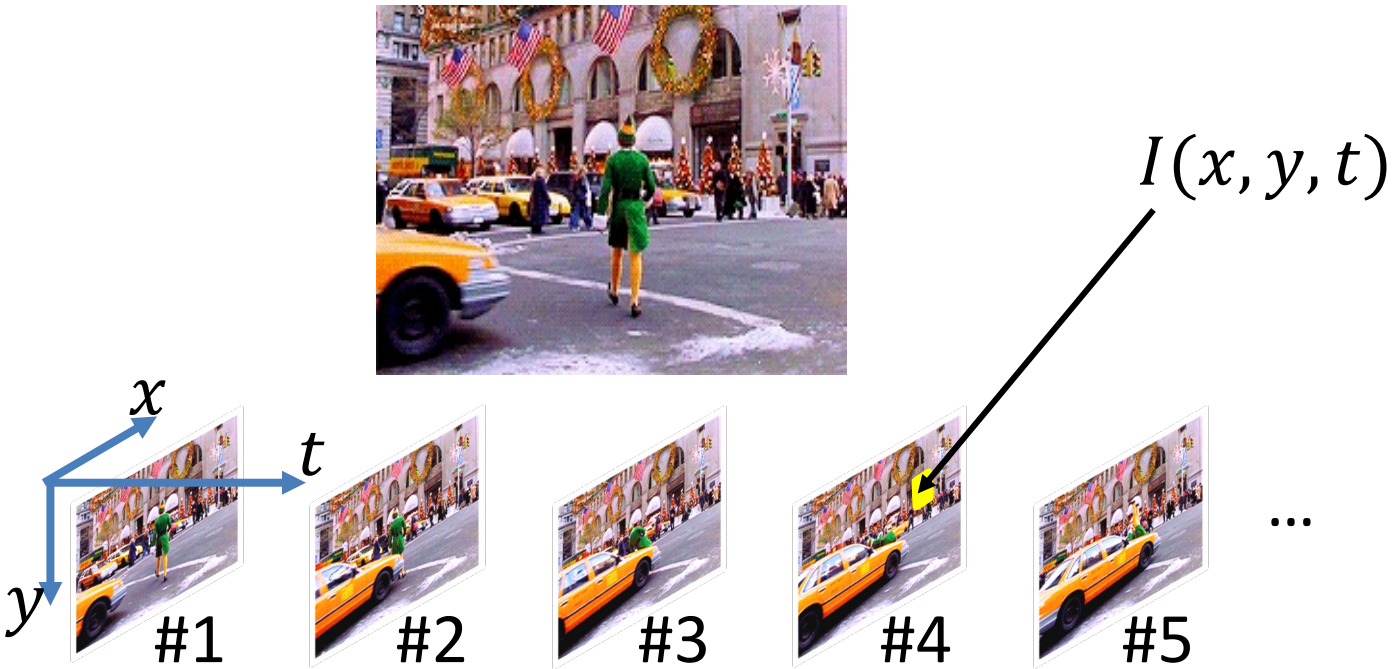


Outline – Optical Flow

- Definition and example
- Applications
- Computational methods
- Aperture problem
- Summary of the main points

Video

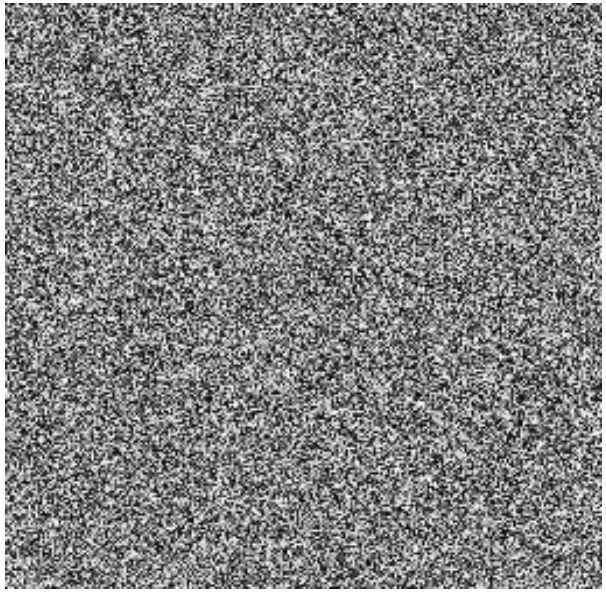
- A video is composed of a sequence of frames over time.
- Image data in a video thus becomes a function of space (x, y) and time (t).



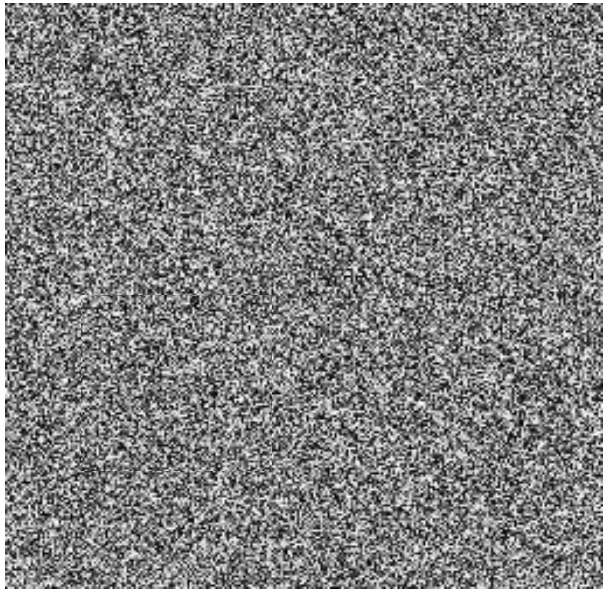
Video Frames

Adjacent three frames in a video

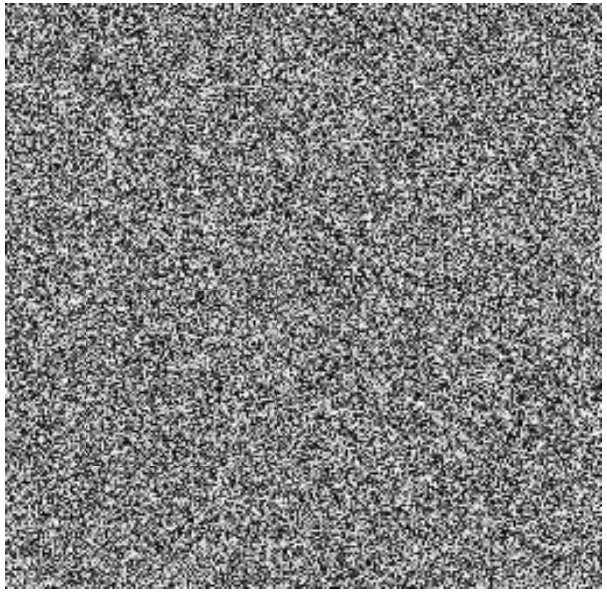
Can you read some information from this video?



Frame #1



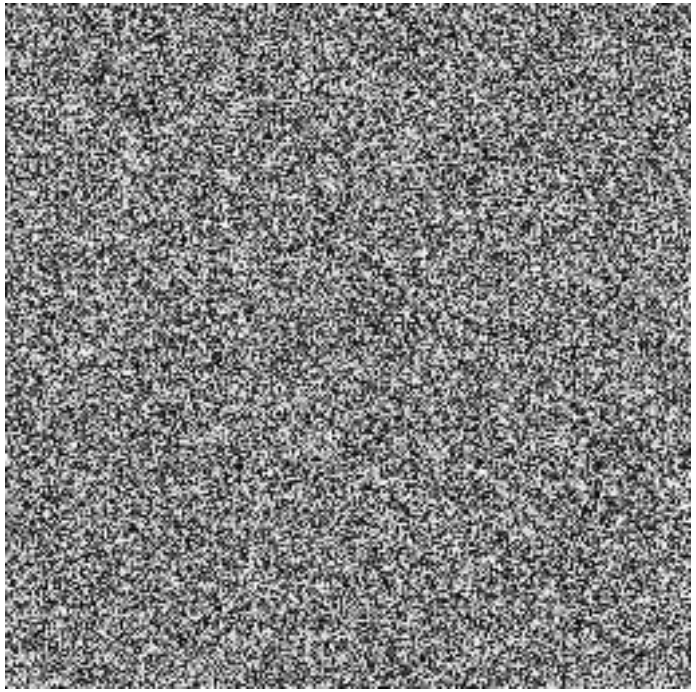
Frame #2



Frame #3

Motion

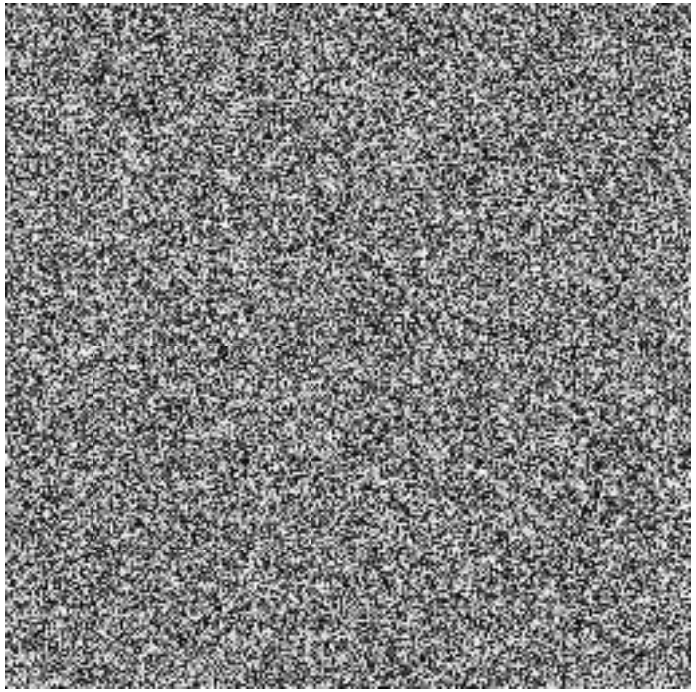
- Motion may be the only clue in some cases



Frame #1

Motion

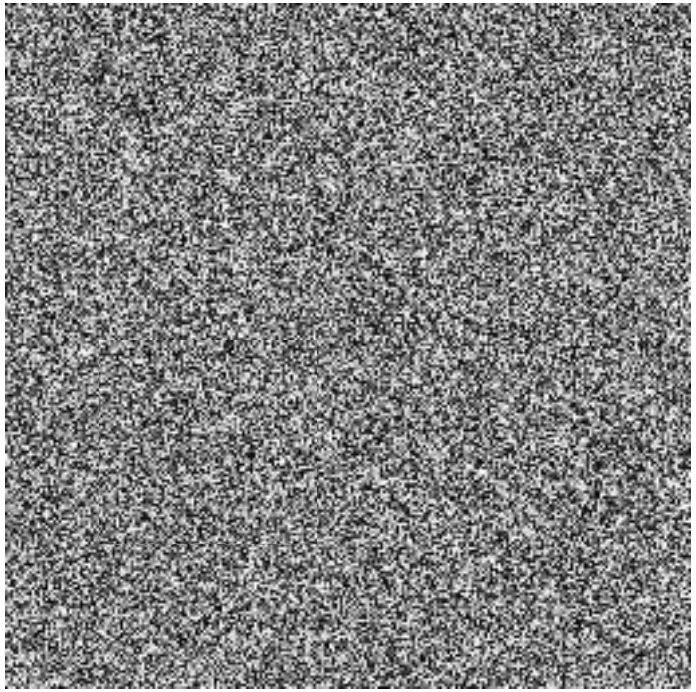
- Motion may be the only clue in some cases



Frame #2

Motion

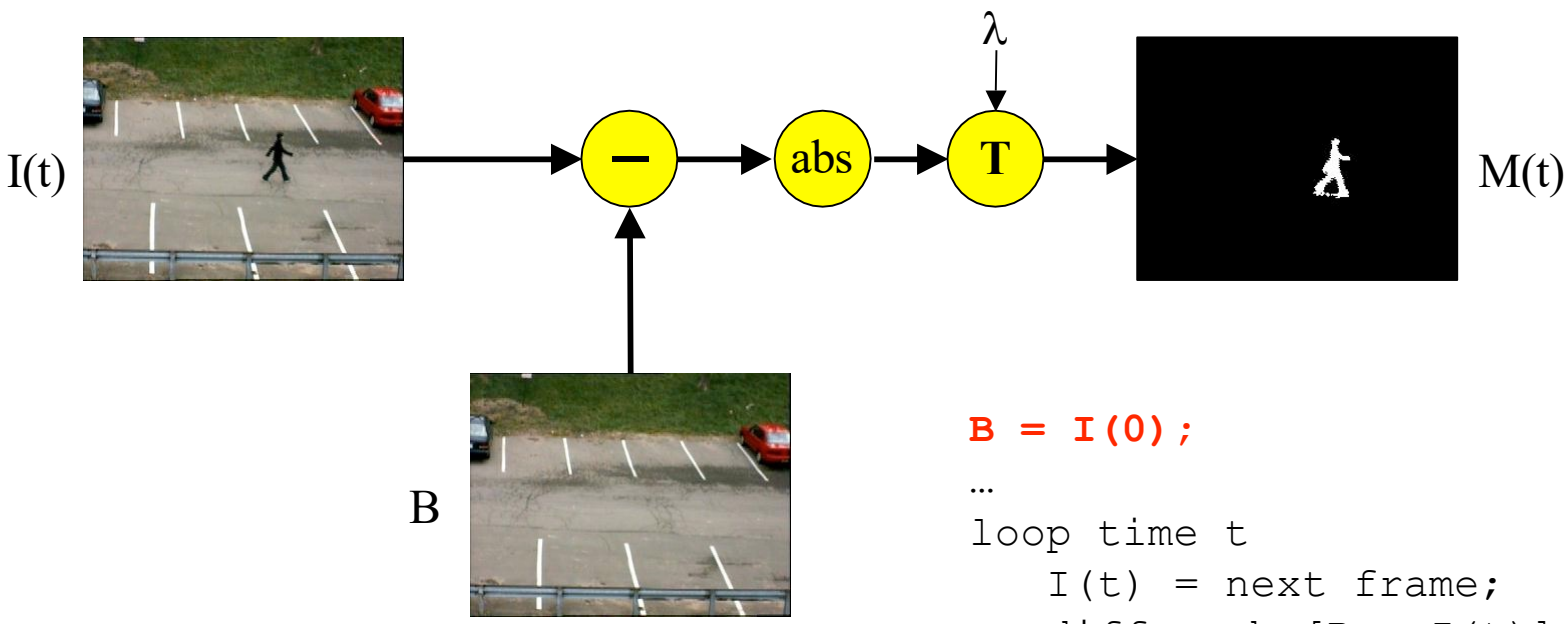
- Motion may be the only clue in some cases



Frame #3

Describe Motion - Background Subtraction

- Background model is a static image
- Pixels are labeled as object (1) or not object (0) based on thresholding the absolute intensity difference between current frame and background.

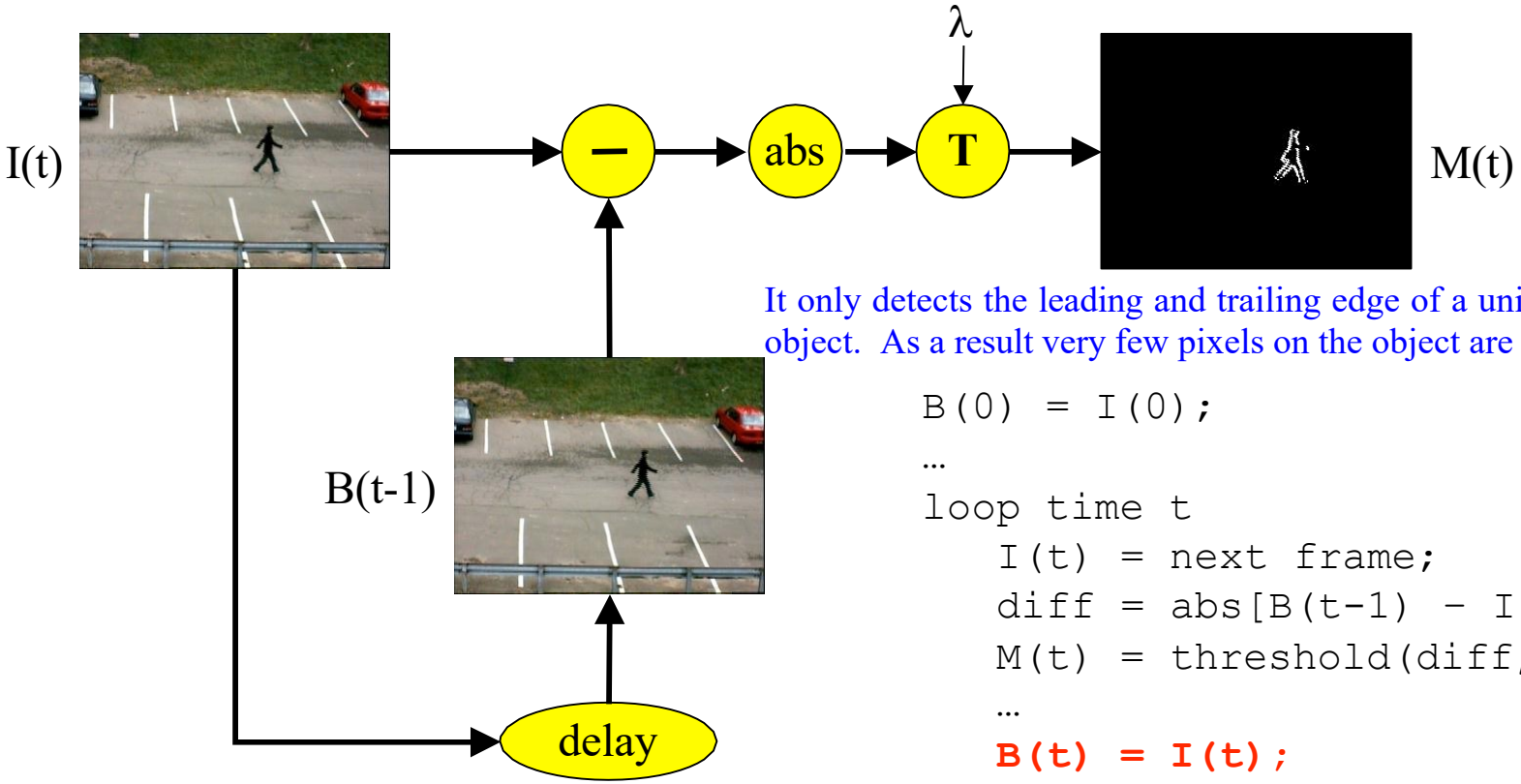



```
B = I(0);
…
loop time t
    I(t) = next frame;
    diff = abs[B - I(t)];
    M(t) = threshold(diff,λ);
    …
end
```

Objects that enter the scene and stop continue to be detected, making it difficult to detect new objects that pass in front of them.

Describe Motion - Frame Differencing

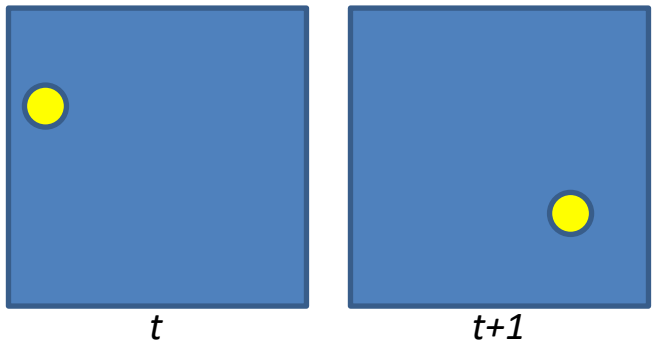
- Background model is a static image
- Pixels are labelled as object (1) or not object (0) based on thresholding the absolute intensity difference between current frame and background.



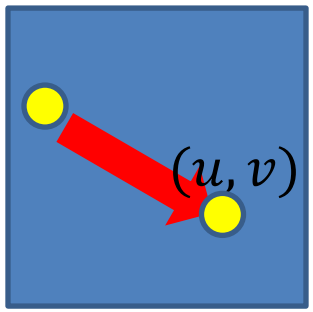
It only detects the leading and trailing edge of a uniformly colored object. As a result very few pixels on the object are labelled.

```
B(0) = I(0);
…
loop time t
    I(t) = next frame;
    diff = abs[B(t-1) - I(t)];
    M(t) = threshold(diff,λ);
    …
    B(t) = I(t);
end
```

Describe Motion - Optical Flow



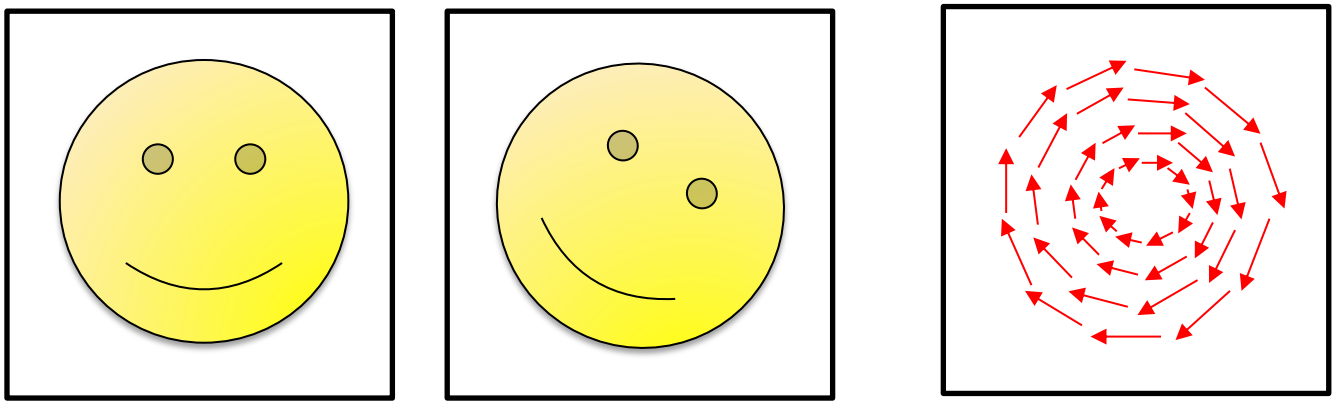
Point in adjacent video frames



The displacement vector (*magnitude* and *direction*)

Optical Flow

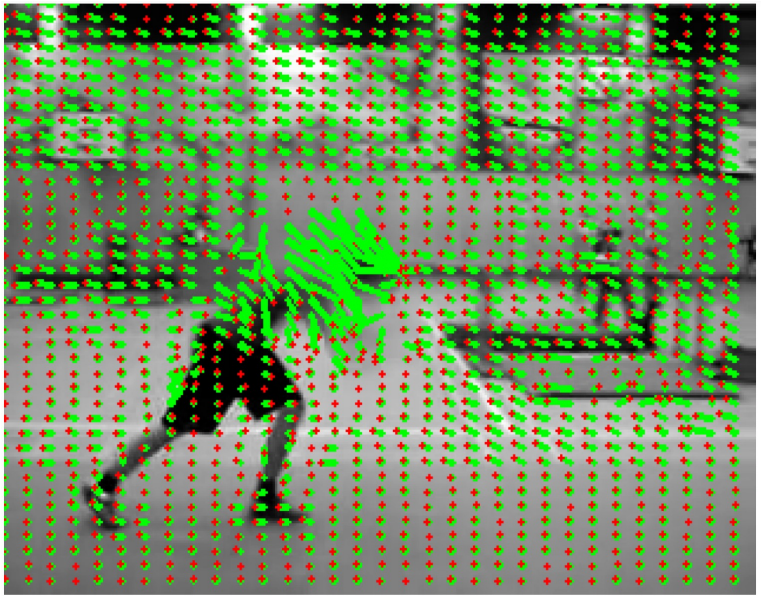
- The optical flow technique aims to estimate the displacement vectors of all pixels in the image sequence.
- In other words, we need to get the motion (*magnitude* and *direction*) at each pixel.



t *t+1* Optical Flow (OF)

Optical Flow

- The optical flow technique aims to estimate the displacement vectors of all pixels in the image sequence.
- In other words, we need to get the motion (*magnitude* and *direction*) at each pixel.

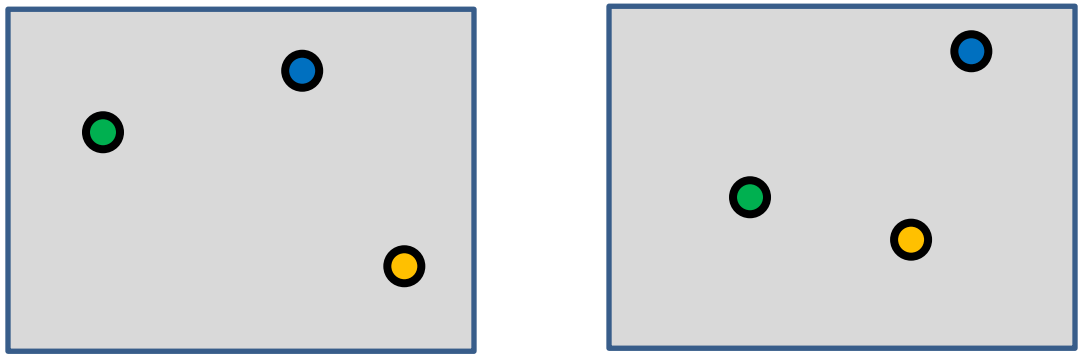


https://www.irisa.fr/texmex/people/jain/w-Flow/motion_cvpr13.pdf

Applications

- Motion-based segmentation
- Structure from motion (3D shape and motion)
- Alignment (Global motion compensation)
  - UAV video analysis
  - Camcorder video stabilization
- Video compression (MPEG)
- Object tracking
- …

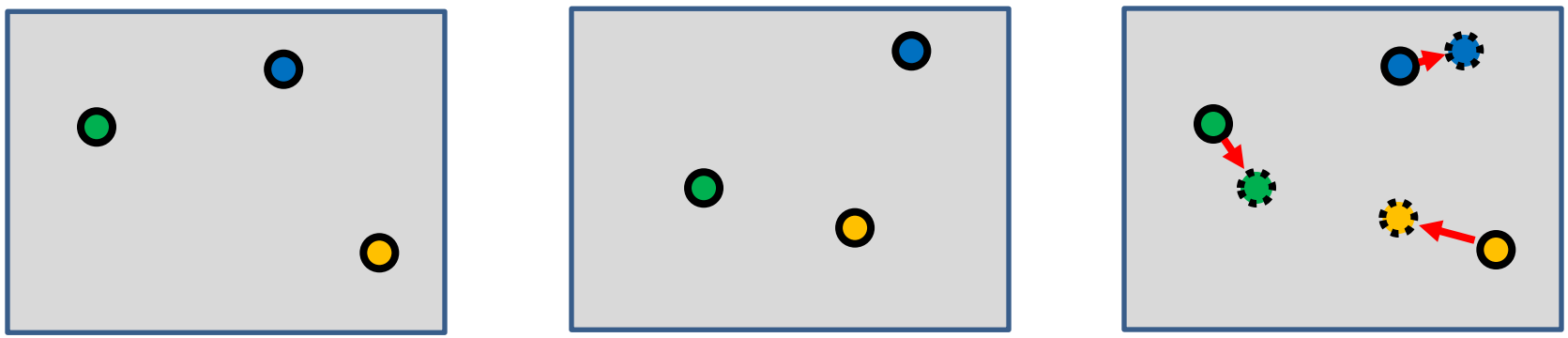
Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t+1)$

Given two adjacent frames, we need to estimate the pixels' **motions** between these two frames.

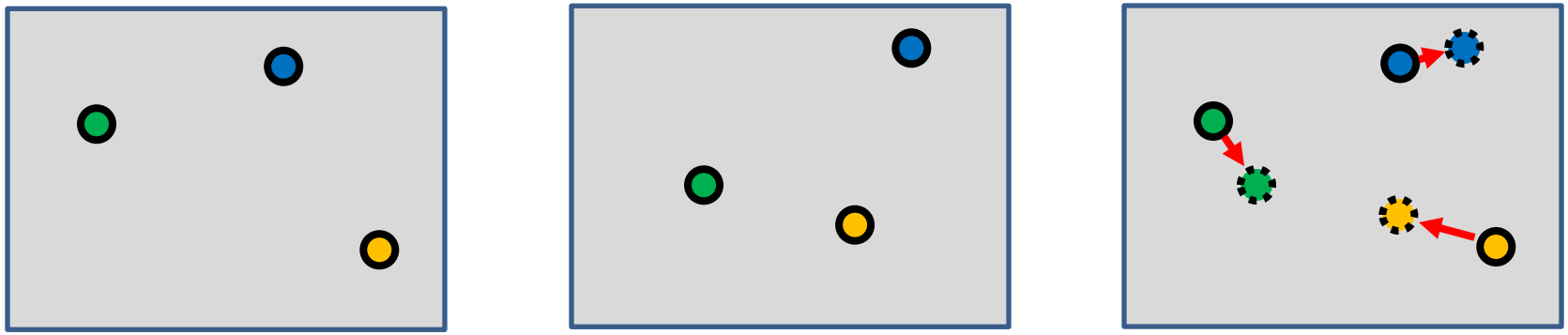
Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t+1)$ $Motions$

Given two adjacent frames, we need to estimate the pixels' **motions** between these two frames.

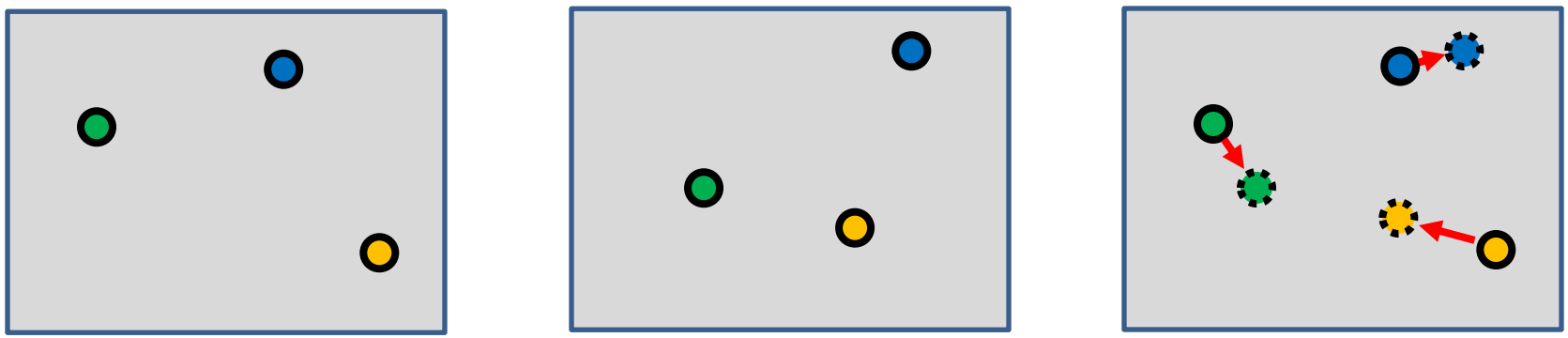
Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t+1)$ $Motions$

How to estimate pixels' motions between these two frames?

Optical Flow Estimation



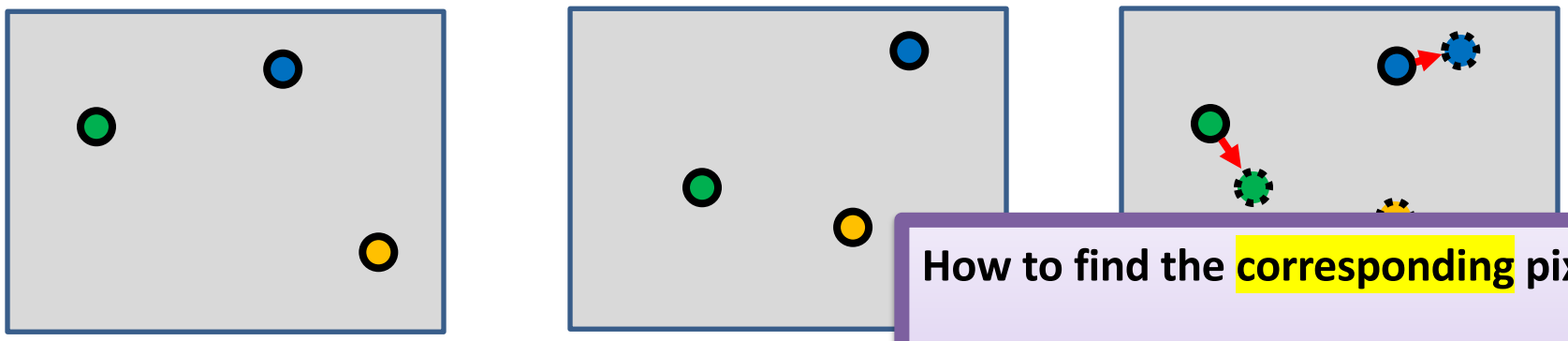
$I(x, y, t)$ $I(x, y, t+1)$ $Motions$

How to estimate pixels' motions between these two frames?

Solve the pixel correspondence problem!

- For each pixel in $I(x, y, t)$, we can look for its corresponding pixel in the next frame $I(x, y, t+1)$.

Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t +$

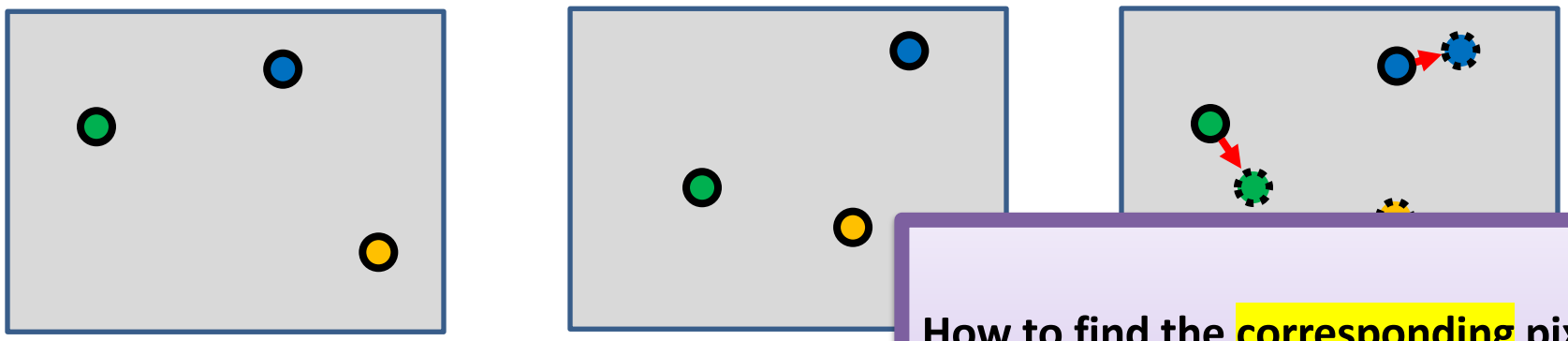
How to find the corresponding pixel?

How to estimate pixels' motions bet

Solve the pixel correspondence problem!

- For each pixel in $I(x, y, t)$, we can look for its corresponding pixel in the next frame $I(x, y, t + 1)$.

Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t +$

How to estimate pixels' motions bet

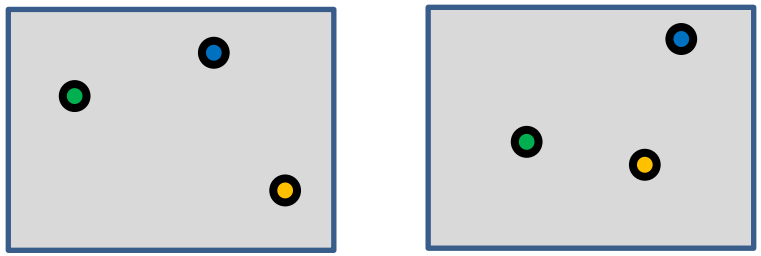
How to find the corresponding pixel?

✓ Optical Flow Estimation Method (*Use some manually specified assumptions to find the correspondence*).

Solve the pixel correspondence problem!

- For each pixel in $I(x, y, t)$, we can look for its corresponding pixel in the next frame $I(x, y, t + 1)$.

Traditional Optical Flow Estimation



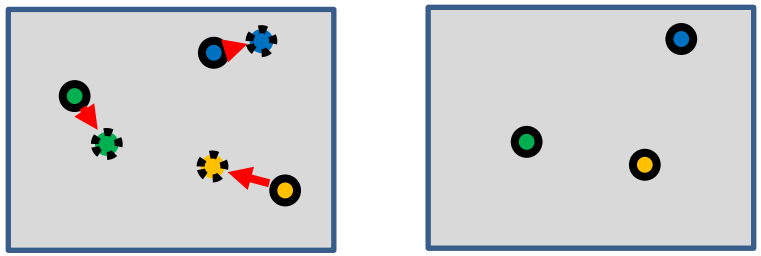
$I(x, y, t)$ $I(x, y, t+1)$

Color Image

Key Assumptions of the optical flow problem:

- **Assumption 1:** Color constancy:
  - ✓ The same pixel in $I(x, y, t)$ looks the same (*i.e.,* same color) as in $I(x, y, t+1)$.

Traditional Optical Flow Estimation



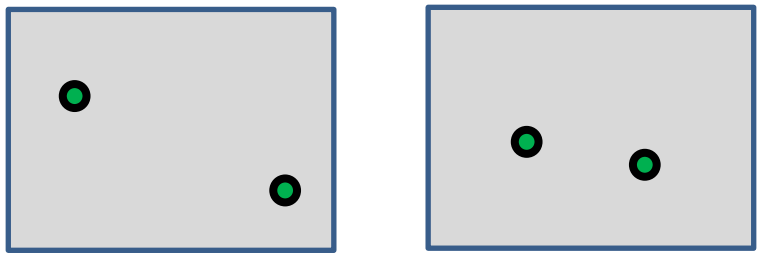
$I(x, y, t)$ $I(x, y, t+1)$

Color Image

Key Assumptions of the optical flow problem:

- **Assumption 1:** Color constancy:
  - ✓ The same pixel in $I(x, y, t)$ looks the same (*i.e.,* same color) as in $I(x, y, t+1)$.

Traditional Optical Flow Estimation

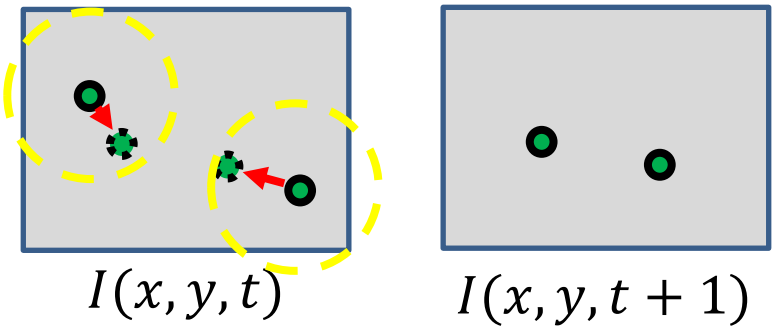


$I(x, y, t)$ $I(x, y, t+1)$

Key Assumptions of the optical flow problem:

- **Assumption 2:** Small motion - pixels do not move too far.

Traditional Optical Flow Estimation



Key Assumptions of the optical flow problem:

- **Assumption 2:** Small motion - pixels do not move too far.
  - Thus, we only need to search in a small region for each pixel.

Traditional Optical Flow Estimation

$I(x, y, t)$

$I(x, y, t+1)$

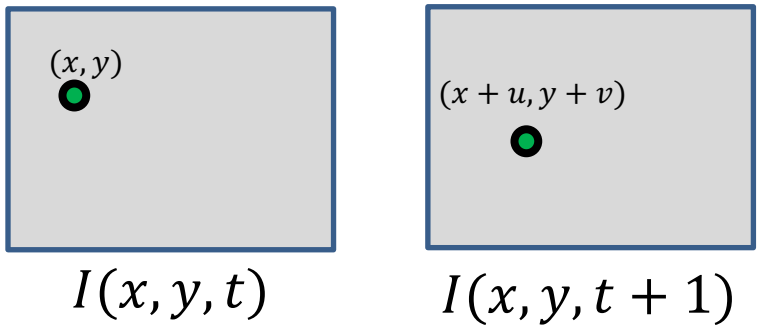
Key Assumptions of the optical flow problem:

- **Assumption 3:** Neighbors move together:
  - ✓ Assume the neighboring patch of each pixel has the same motion vector (u,v). Thus, we can find correspondence at the patch level, instead of at single pixel level.

Color Constancy Constraint (Assumption 1)

- Color Constancy Equation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$



Color Constancy Constraint (Assumption 2)

- Color Constancy Equation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

- Take Taylor expansion of $I(x + u, y + v, t + 1)$ at (x, y, t) to linearize the right side:

$$I(x + u, y + v, t + 1) \approx I(x, y, t) + \frac{\partial I}{\partial x}.u + \frac{\partial I}{\partial y}.v + \frac{\partial I}{\partial t} + \cdots$$

- Assuming **small motion**, we can write:

$$I(x + u, y + v, t + 1) = I(x, y, t) + \frac{\partial I}{\partial x}.u + \frac{\partial I}{\partial y}.v + \frac{\partial I}{\partial t}$$

Color Constancy Constraint (Assumption 2)

- Assuming small motion, we can write:

Image derivative along x

Image derivative along t

$$I(x+u, y+v, t+1) = I(x, y, t) + \boxed{\frac{\partial I}{\partial x}}.u + \boxed{\frac{\partial I}{\partial y}}.v + \boxed{\frac{\partial I}{\partial t}}$$

Image derivative along y

- Let $\frac{\partial I}{\partial x} = I_x, \frac{\partial I}{\partial y} = I_y$ and $\frac{\partial I}{\partial t} = I_t$, we can write:

$$I(x+u, y+v, t+1) = I(x, y, t) + I_x.u + I_y.v + I_t$$

Color Constancy Equation

- Color Constancy Equation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

- Assuming small motion, we have:

$$I(x + u, y + v, t + 1) = I(x, y, t) + I_x . u + I_y . v + I_t$$

- And thus:

$$I(x, y, t) = I(x, y, t) + I_x . u + I_y . v + I_t$$

Color Constancy Equation

$$I(x, y, t) = I(x, y, t) + I_x . u + I_y . v + I_t$$

- We have $I(x, y, t)$ at both sides and thus we can cancel them:

$$\cancel{I(x, y, t)} = \cancel{I(x, y, t)} + I_x . u + I_y . v + I_t$$

- Color (or Brightness) Constancy Equation is:

$$I_x . u + I_y . v + I_t = 0$$

Color Constancy Equation

- What do the terms of the color constancy equation represent?

flow velocities

$$I_x . u + I_y . v + I_t = 0$$

Image gradients
(at a point p)

temporal gradient

How do you compute these terms?

SCC.366: Media Coding & Processing

2022-2023

Week 09 - Lecture 2

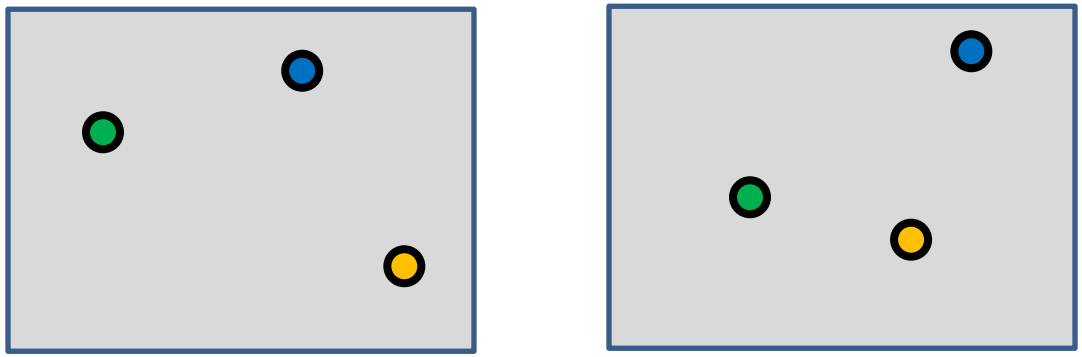
Optical Flow Cont.

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline – Optical Flow

- Definition and example
- Applications
- Computational methods
- Aperture problem
- Summary of the main points

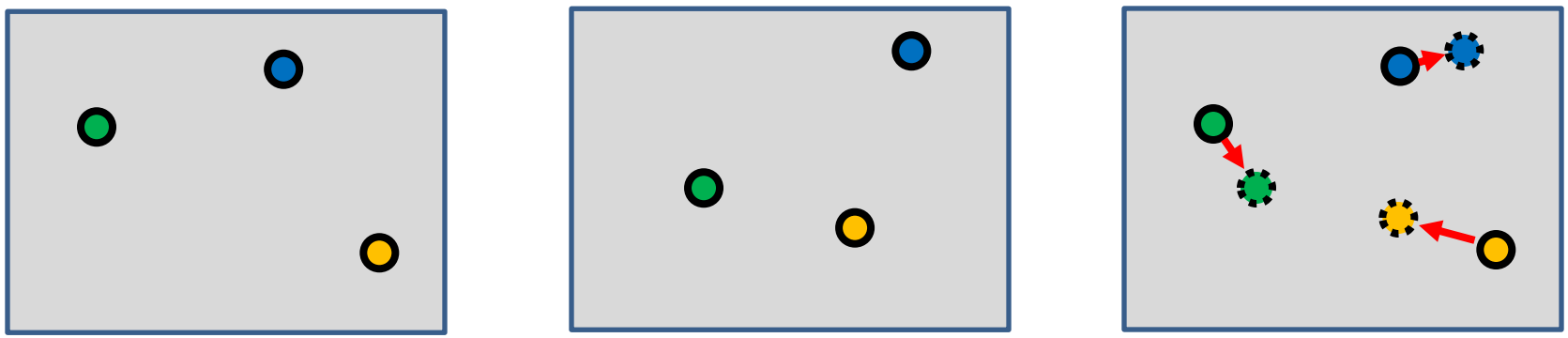
Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t+1)$

Given two adjacent frames, we need to estimate the pixels' **motions** between these two frames.

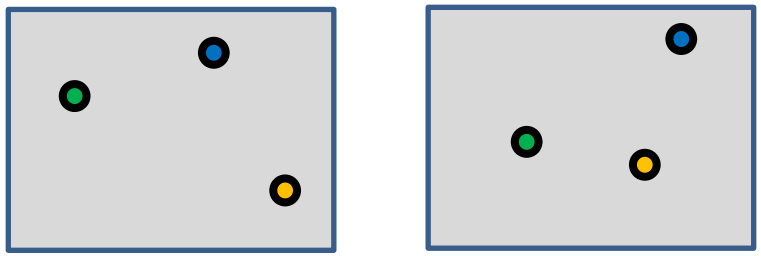
Optical Flow Estimation



$I(x, y, t)$ $I(x, y, t+1)$ $Motions$

Given two adjacent frames, we need to estimate the pixels' **motions** between these two frames.

Traditional Optical Flow Estimation



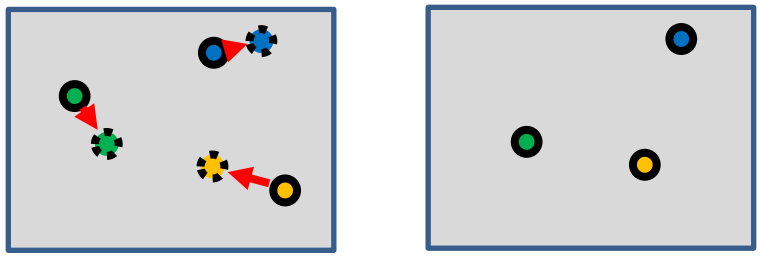
$I(x, y, t)$ $I(x, y, t+1)$

Color Image

Key Assumptions of the optical flow problem:

- **Assumption 1:** Color constancy:
  - ✓ The same pixel in $I(x, y, t)$ looks the same (*i.e.,* same color) as in $I(x, y, t+1)$.

Traditional Optical Flow Estimation



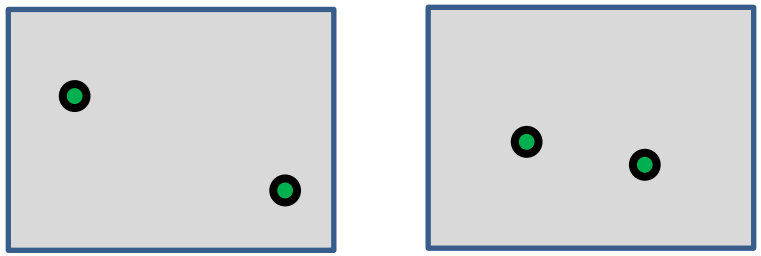
$I(x, y, t)$ $I(x, y, t+1)$

Color Image

Key Assumptions of the optical flow problem:

- **Assumption 1:** Color constancy:
  - ✓ The same pixel in $I(x, y, t)$ looks the same (*i.e.,* same color) as in $I(x, y, t+1)$.

Traditional Optical Flow Estimation

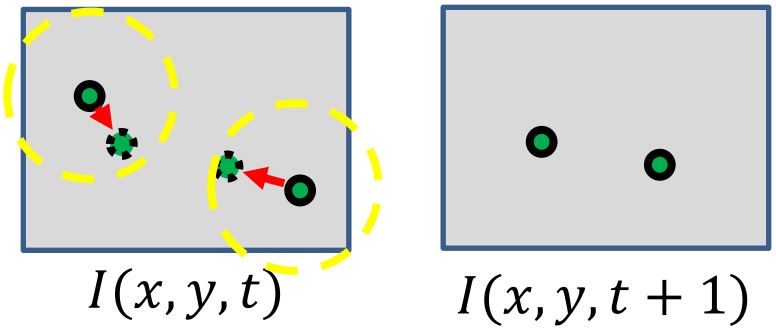


$I(x, y, t)$ $I(x, y, t+1)$

Key Assumptions of the optical flow problem:

- **Assumption 2:** Small motion - pixels do not move too far.

Traditional Optical Flow Estimation



Key Assumptions of the optical flow problem:

- **Assumption 2:** Small motion - pixels do not move too far.
  - Thus, we only need to search in a small region for each pixel.

Traditional Optical Flow Estimation

$I(x, y, t)$

$I(x, y, t+1)$

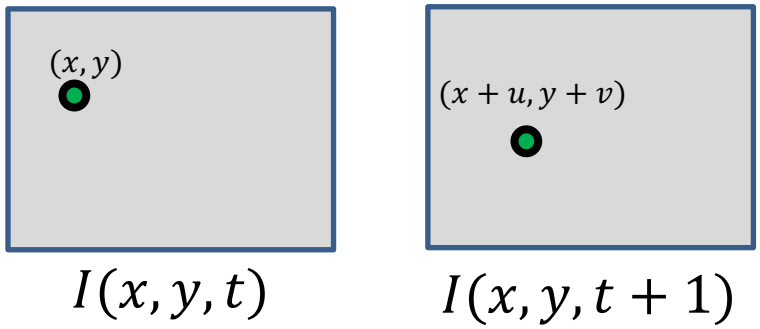
Key Assumptions of the optical flow problem:

- **Assumption 3:** Neighbors move together:
  - ✓ Assume the neighboring patch of each pixel has the same motion vector (u,v). Thus, we can find correspondence at the patch level, instead of at single pixel level.

Color Constancy Constraint (Assumption 1)

- Color Constancy Equation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$



Color Constancy Equation

- Color Constancy Equation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

- Assuming small motion, we have:

$$I(x + u, y + v, t + 1) = I(x, y, t) + I_x . u + I_y . v + I_t$$

- And thus:

$$I(x, y, t) = I(x, y, t) + I_x . u + I_y . v + I_t$$

Color Constancy Equation

- What do the terms of the color constancy equation represent?

flow velocities

$$I_x . u + I_y . v + I_t = 0$$

Image gradients
(at a point p)

temporal gradient

How do you compute these terms?

Color Constancy Equation

- Image gradients $I_x = \frac{\partial I}{\partial x}$ and $I_y = \frac{\partial I}{\partial y}$ can be calculated using several approaches such as forward difference, Sobel edge detector or any other edge detectors.
- Temporal gradient $I_t = \frac{\partial I}{\partial x}$ is calculated by frame differencing

flow velocities

$$I_x . u + I_y . v + I_t = 0$$

Image gradients
(at a point p)

temporal gradients

Color Constancy Equation (Example)

t

1	1	1	1	1
1	1	1	1	1
1	**10**	**10**	**10**	**10**
1	**10**	**10**	**10**	**10**
1	**10**	**10**	**10**	**10**
1	**10**	**10**	**10**	**10**

$t+1$

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	**10**	**10**	**10**
1	1	**10**	**10**	**10**
1	1	**10**	**10**	**10**

$I_x = \frac{\partial I}{\partial x}$ Using filter -1 0 1

-	0	0	0	-
-	0	0	0	-
-	**9**	0	0	-
-	**9**	0	0	-
-	**9**	0	0	-
-	**9**	0	0	-

$I_y = \frac{\partial I}{\partial y}$ Using filter -1 0 1

-	-	-	-	-
0	**9**	**9**	**9**	**9**
0	**9**	**9**	**9**	**9**
-	0	0	0	0
0	0	0	0	0
-	-	-	-	-

$I_t = \frac{\partial I}{\partial t}$

0	0	0	0	0
0	0	0	0	0
0	**9**	**9**	**9**	**9**
0	**9**	0	0	0
0	**9**	0	0	0
0	**9**	0	0	0

Color Constancy Equation - Underconstrained

- How to compute flow velocities u and v?
- Color Constancy Equation is one equation with 2 unknown terms

flow velocities
(unknown)

$$I_x . u + I_y . v + I_t = 0$$

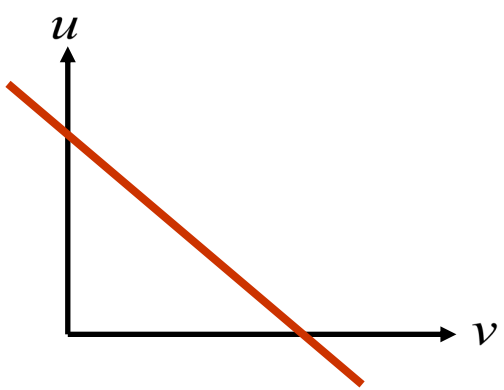
Image gradients
(at a point p)

temporal gradients

Color Constancy Equation - Underconstrained

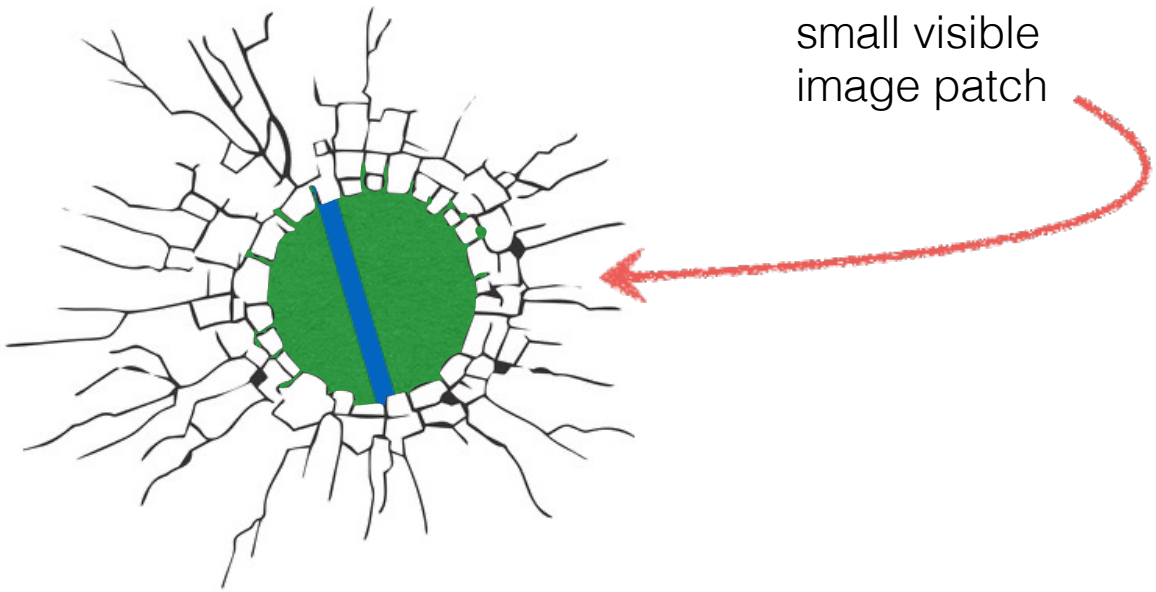
- Color Constancy Equation is one equation with 2 unknown terms and thus flow velocities u and v cannot be found uniquely with a single constraint (i.e., a single pixel)

$$I_x.u + I_y.v + I_t = 0$$



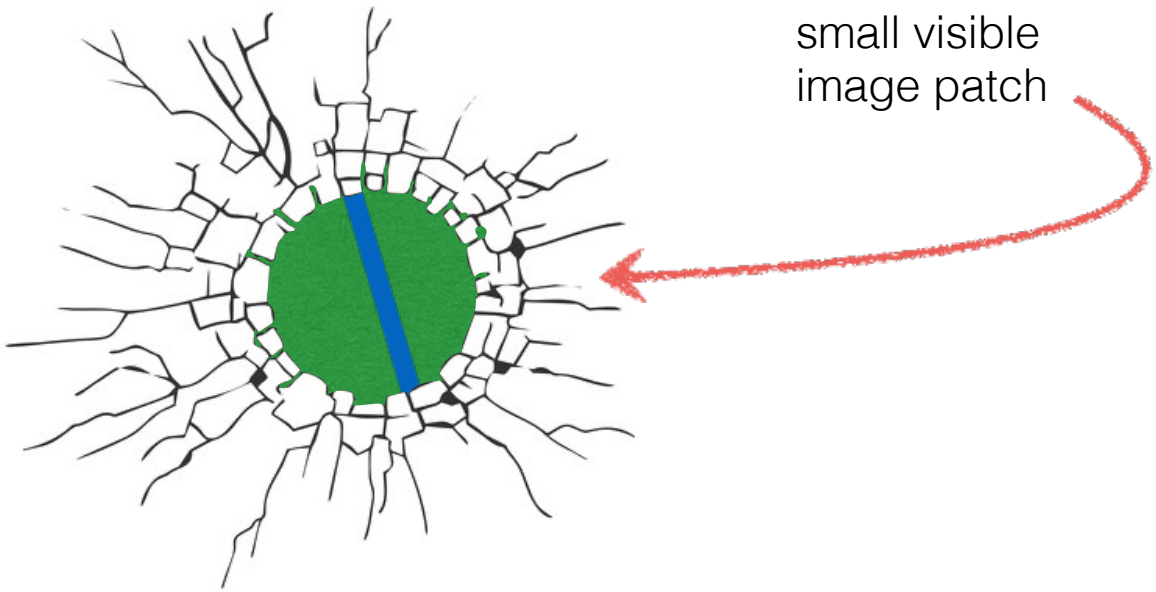
This issue is called **Aperture Problem**

Aperture Problem



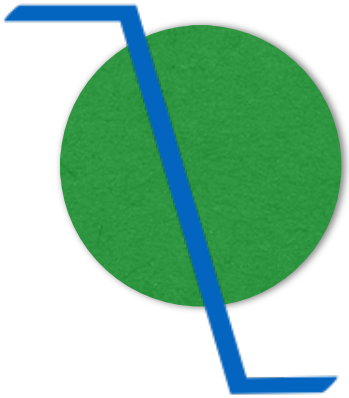
In which direction is the line moving?

Aperture Problem

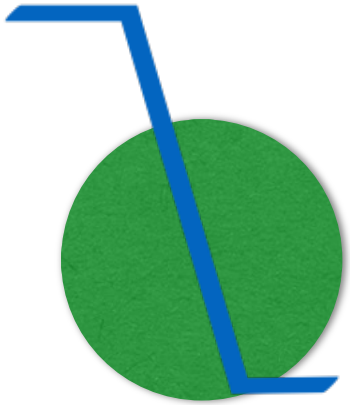


In which direction is the line moving?

Aperture Problem



Aperture Problem



Working around the aperture problem

- There are several approaches to estimate flow velocities u and v, such as:
  - Horn-Schunck Optical Flow (1981)
  - Lucas-Kanade Optical Flow (1981)

$$I_x . u + I_y . v + I_t = 0$$

Lucas-Kanade Optical Flow

$$I_x . u + I_y . v + I_t = 0$$

- Assume that the surrounding patch (say 5x5) has 'constant flow'
- Assumption is:
  - Neighboring pixels have same displacement, i.e., (u, v) (Assumption 3)

- Using a 5 x 5 image patch, gives us ? equations

Lucas-Kanade Optical Flow

$$I_x.u + I_y.v + I_t = 0$$

- Assume that the surrounding patch (say 5x5) has 'constant flow'
- Assumption is:
  - Neighboring pixels have same displacement, i.e., (u, v) (Assumption 3)

- Using a 5 x 5 image patch, gives us 25 equations

$$I_x(p_1).u + I_y(p_1).v + I_t(p_1) = 0$$
$$I_x(p_2).u + I_y(p_2).v + I_t(p_2) = 0$$
$$I_x(p_3).u + I_y(p_3).v + I_t(p_3) = 0$$

$$I_x(p_{25}).u + I_y(p_{25}).v + I_t(p_{25}) = 0$$

Lucas-Kanade Optical Flow

$$I_x . u + I_y . v + I_t = 0$$

- Assume that the surrounding patch (say 5x5) has 'constant flow'
- Assumption is:
  - Neighboring pixels have same displacement, i.e., (u, v) (Assumption 3)

- Using a 5 x 5 image patch, gives us 25 equations

$$\begin{bmatrix} I_x(\mathrm{p}_1) & I_y(\mathrm{p}_1) \\ I_x(\mathrm{p}_2) & I_y(\mathrm{p}_2) \\ \vdots & \vdots \\ I_x(\mathrm{p}_{25}) & I_y(\mathrm{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathrm{p}_1) \\ I_t(\mathrm{p}_2) \\ \vdots \\ I_t(\mathrm{p}_{25}) \end{bmatrix}$$

Lucas-Kanade Optical Flow

$$I_x . u + I_y . v + I_t = 0$$

- Assume that the surrounding patch (say 5x5) has 'constant flow'
- Assumption is:
  - Neighboring pixels have same displacement, i.e., (u, v) (Assumption 3)

- Using a 5 x 5 image patch, gives us 25 equations

$$\begin{bmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_{25}) & I_y(\mathbf{p}_{25}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_{25}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Lucas-Kanade Optical Flow

$$\begin{bmatrix} I_x(\mathbf{p_1}) & I_y(\mathbf{p_1}) \\ I_x(\mathbf{p_2}) & I_y(\mathbf{p_2}) \\ \vdots & \vdots \\ I_x(\mathbf{p_{25}}) & I_y(\mathbf{p_{25}}) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} I_t(\mathbf{p_1}) \\ I_t(\mathbf{p_2}) \\ \vdots \\ I_t(\mathbf{p_{25}}) \end{bmatrix}$$

$$\underset{25\text{x}2}{A} \quad \underset{2\text{x}1}{d} = \underset{25\text{x}1}{b}$$

Least squares solution for *d* given by $(A^T A)\ d = A^T b$

$$\underbrace{\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix}}_{A^T A} \begin{bmatrix} u \\ v \end{bmatrix} = - \underbrace{\begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}}_{A^T b}$$

Lucas-Kanade Optical Flow

$$\begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$$

$$A^T A \qquad\qquad\qquad\qquad A^T b$$

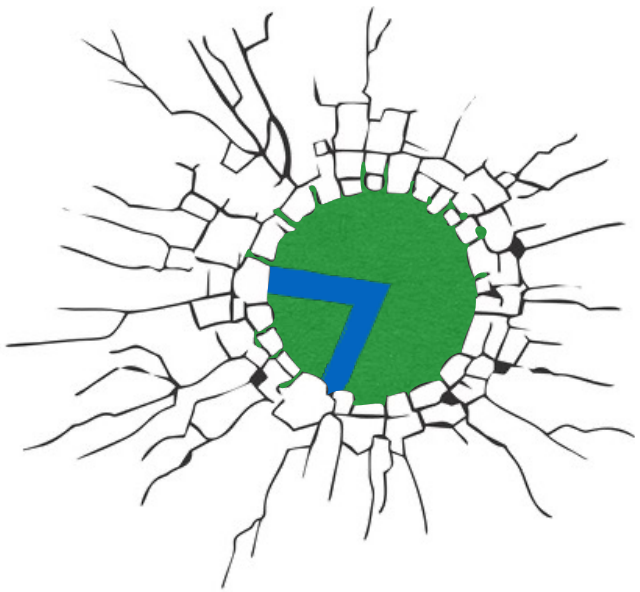
When is this solvable? I.e., what are good points to track?

- $\mathbf{A^TA}$ should be invertible $\lambda_1 \geq \lambda_2 > 0$
- λ_2 should not be too small
- λ_1 / λ_2 should not be too large (λ_1 = larger eigenvalue)

Does this remind you of anything?

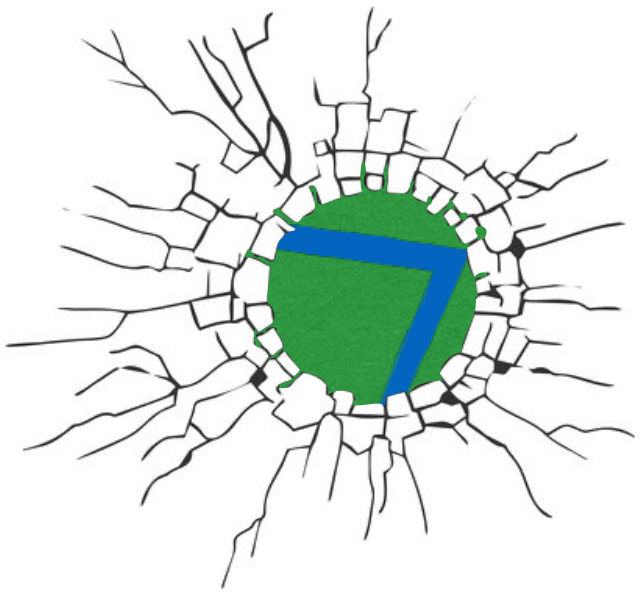
Criteria for Harris corner detector

Lucas-Kanade Optical Flow



Want patches with different gradients to the avoid aperture problem

Lucas-Kanade Optical Flow



Want patches with different gradients to the avoid aperture problem

Summary of Optical Flow Estimation Assumptions

- **Assumption 1:** Color constancy.
- **Assumption 2:** Small motion - pixels do not move far.
- **Assumption 3:** Neighbors move together.

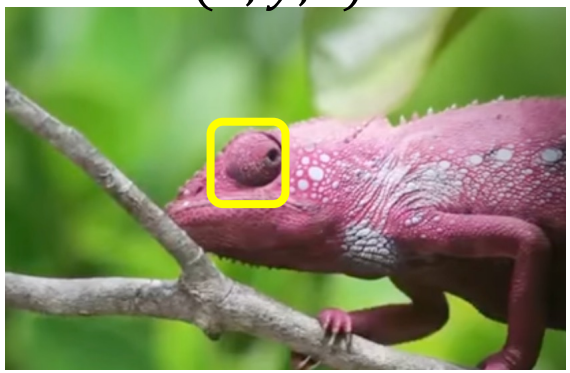
Any issues within these assumptions?

Assumption 1: Color Constancy Assumption

However, color can change over time!

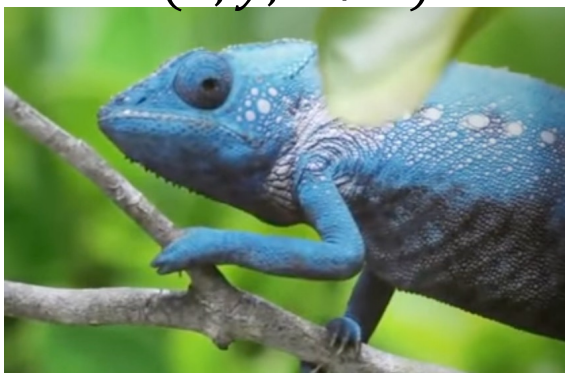
How to handle this issue?

$I(x, y, t)$



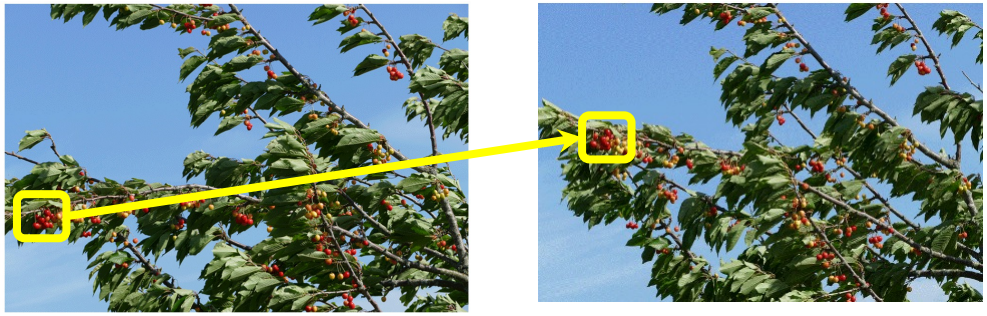
chameleon

$I(x, y, t+1)$



Assumption 2: Small Motion Assumption

Video



Frame t Frame t+1

- Is the motion small enough?

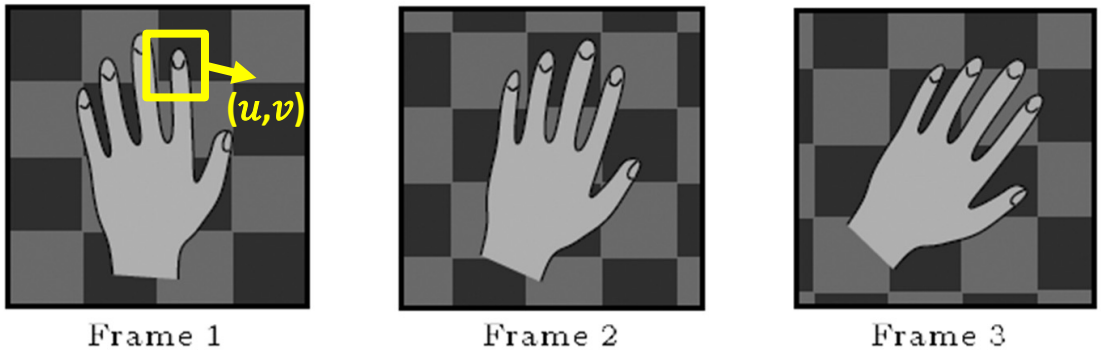
  Probably NOT.
- How to handle this problem?

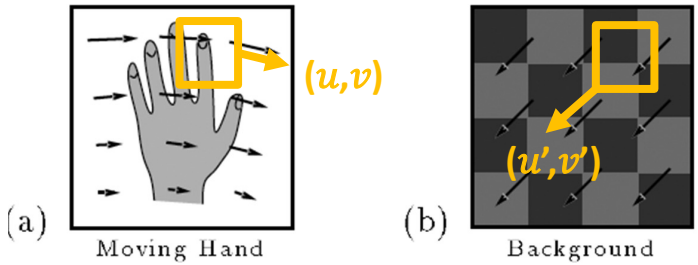
Assumption 3: Neighboring pixels move together

However, pixels may not move like their neighbors.

How to handle this issue?

- Motion segmentation – Multiple Layers.





Foreground and background are both moving, but move differently.

Foreground Layer Background Layer

J. Wang and E. Adelson. Layered Representation for Motion Analysis.

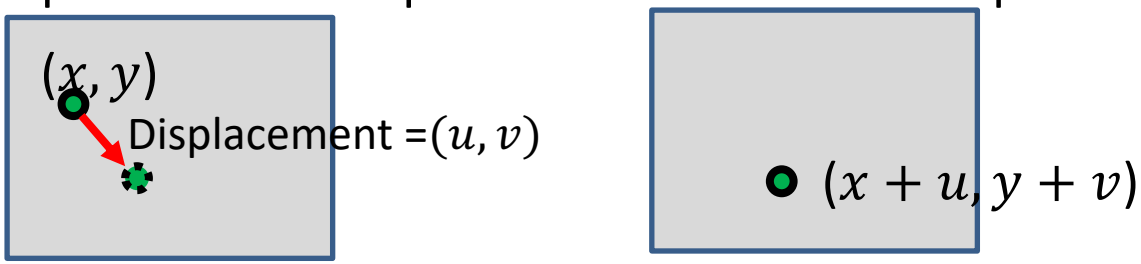
Motion of textureless regions

Question: How will the optical flow estimation behave?

Optical Flow Representation

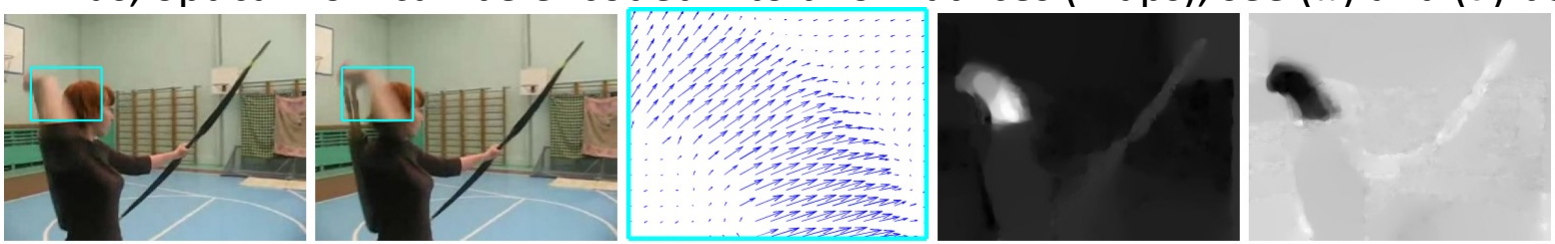
Question: How can we store the optical flow estimation result?

- Optical flow represents the displacement vector of each pixel.



Frame t Frame $t+1$

- The displacement vector <u>at each pixel</u> contains two numbers (u, v).
- Thus, optical flow can be encoded into two matrices (maps), see (d) and (e) below.



(a) (b) (c) (d) (e)

(a),(b): a pair of consecutive video frames.
(c): a close-up of optical flow in the outlined area;
(d): horizontal component (u) of the optical flow (*higher intensity corresponds to positive values, and lower intensity to negative values*).
(e): vertical component (v).

SCC.366: Media Coding & Processing

2022-2023

Week 10 Lecture 1

Discrete Cosine Transform (DCT)

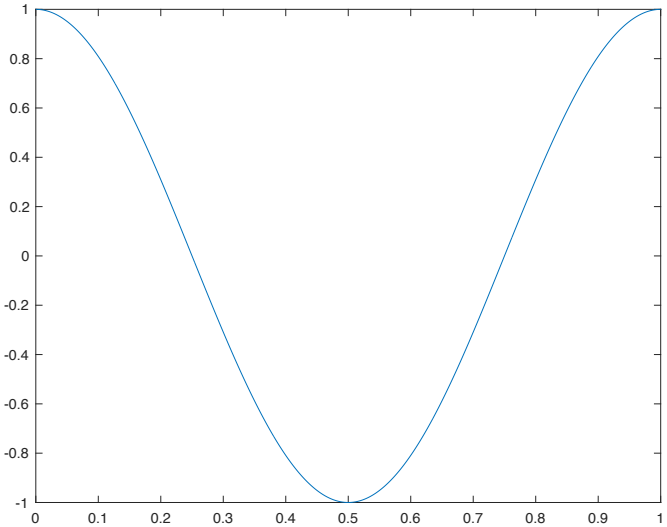
Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

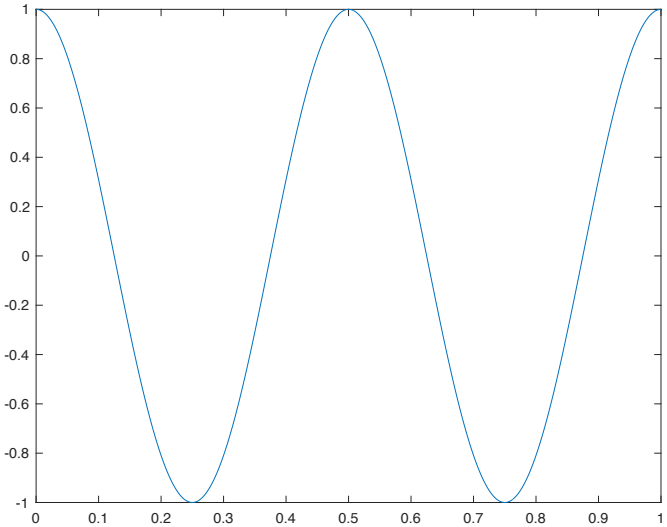
- Discrete Cosine Transform (DCT)
  - 1D DCT
  - 2D DCT
- DCT Applications, e.g., compression, data hiding
- JPEG Compression
- Summary of the main points

1D DCT

- The way that DCT works, is we take some data, e.g., image, and we try to represent it as a sum of lots of cosine waves



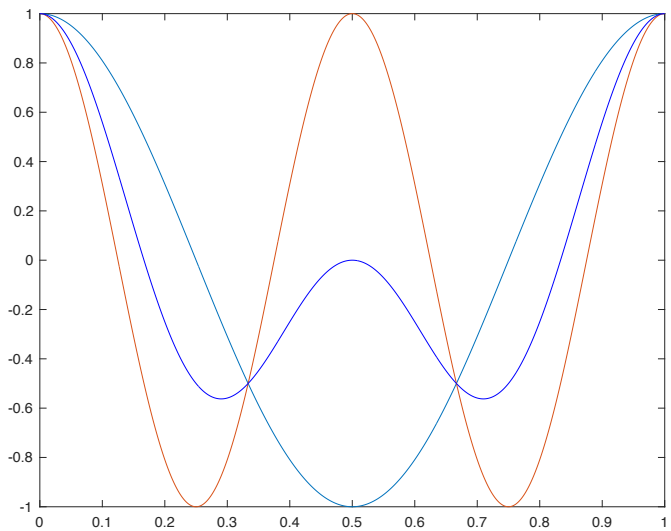
Standard Cosine Wave



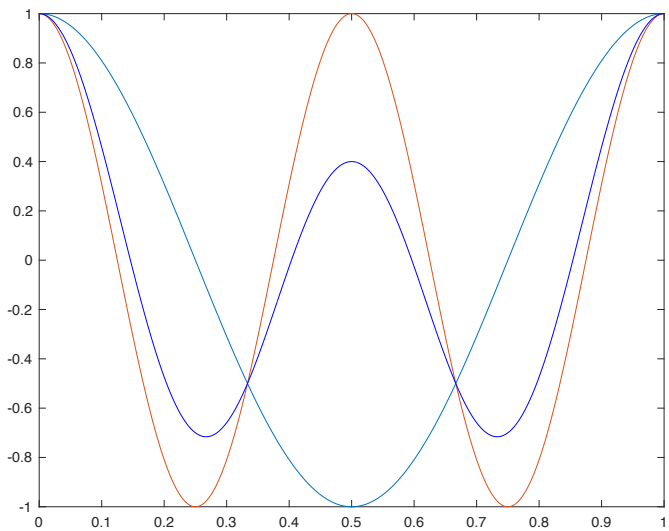
High Frequency Cosine Wave

1D DCT

- We can create a more complex shaped wave by adding these two cosine waves
- In practice, adding two waves together results in a wave much taller than the input. So, we weight both waves, e.g., taking an average or weighted averaging



Averaging



Weighted Averaging

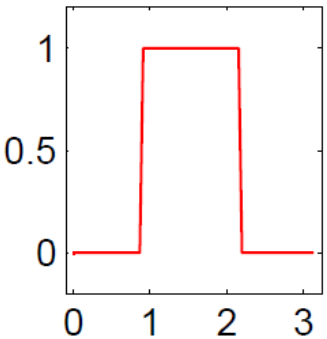
0.7 weight for standard cosine

0.3 weight for the other cosine

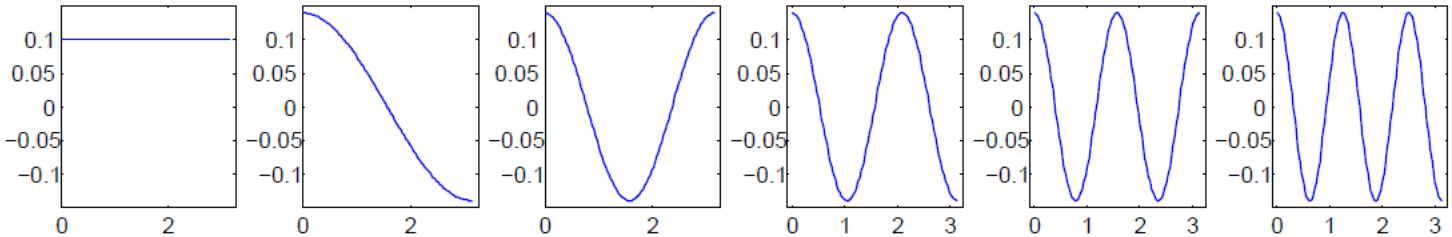
1D DCT

- By increasing the number of cosine waves, we can increase the number of possible shapes of waves we can produce

Reconstruction by Cosine Waves



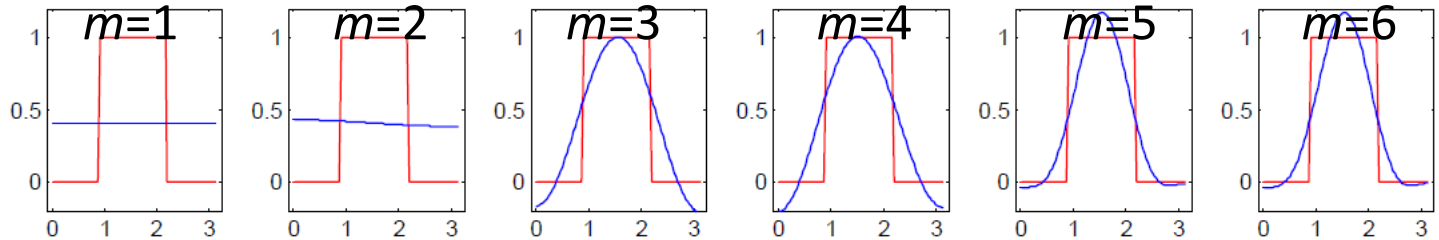
x



$\mathbf{b}_1$ $\mathbf{b}_2$ $\mathbf{b}_3$ $\mathbf{b}_4$ $\mathbf{b}_5$ $\mathbf{b}_6$

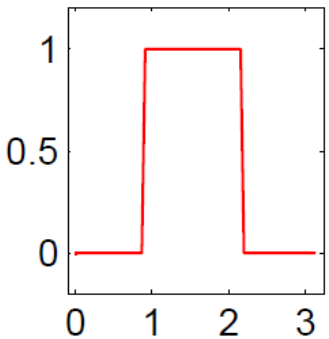
$$\mathbf{x} \approx \tilde{\mathbf{x}} = \sum_{k=1}^{m} x'^{k} \mathbf{b}_k$$

:



Now, let's reconstruct a signal **x** as a combination of selected cosine functions.
[Note: We have infinitely many cosine waves.]

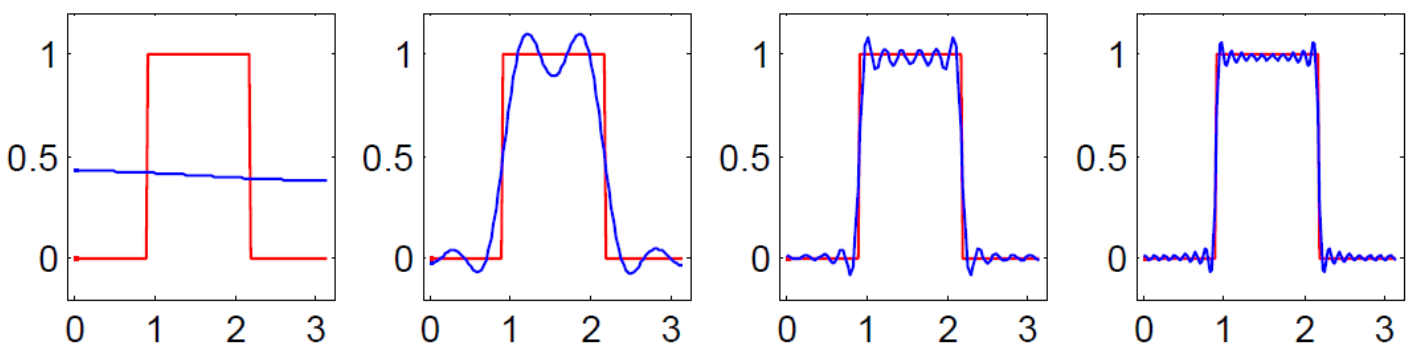
Reconstruction by Cosine Waves



x

- Cosine waves facilitate coarse-to-fine reconstruction.
  - Low-frequency (small # oscillations) components contribute to the overall shape.
  - High-frequency components add fine details.
  - Contribution of very high-frequency components are negligible.

$$\mathbf{x} \approx \tilde{\mathbf{x}} = \sum_{k=1}^{m} x'^{k} \mathbf{b}_k :$$

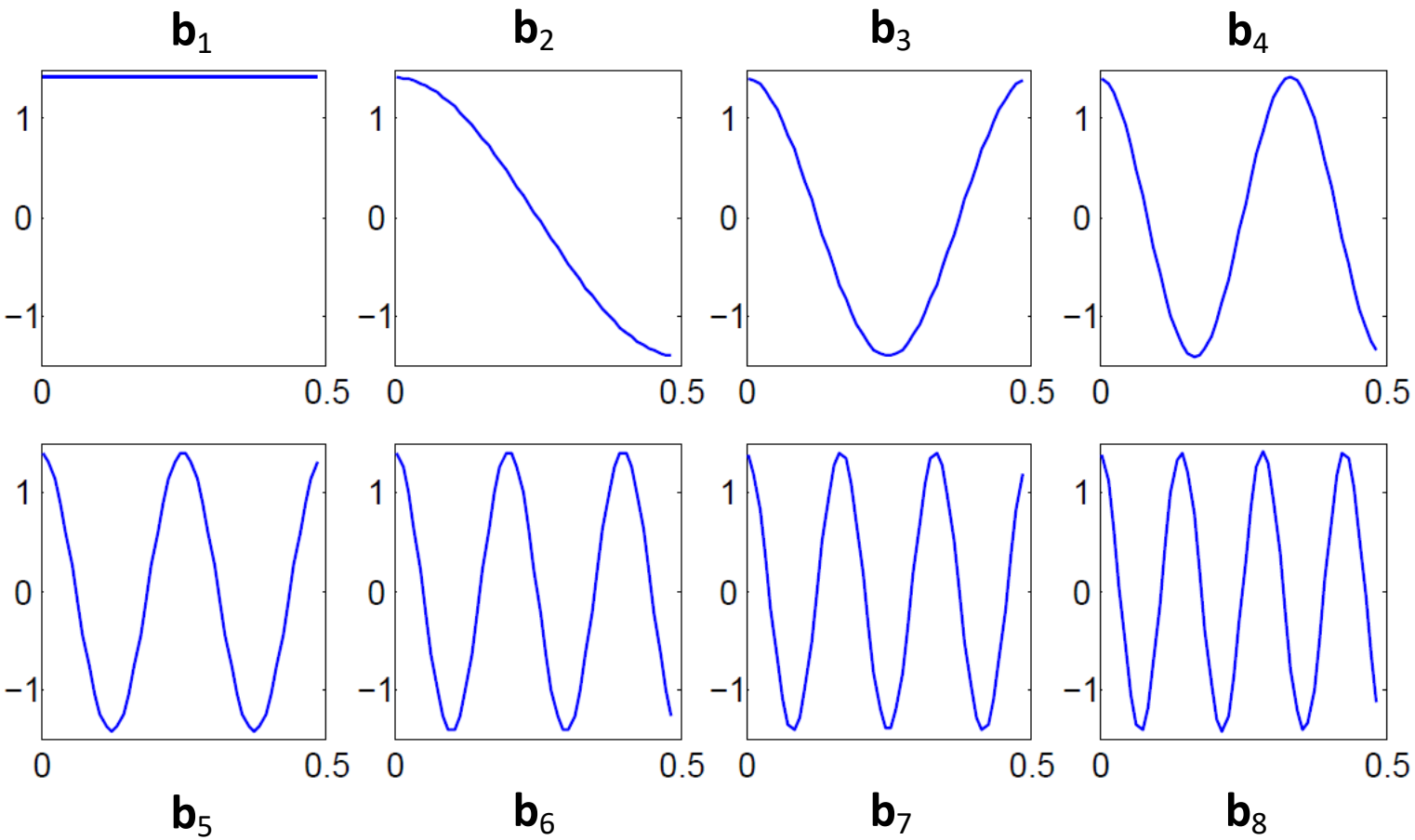


m=2 *m*=10 *m*=30 *m*=50

1D DCT

- The way that the mathematics work, is if we have a signal that 8 long (i.e., a vector of 8 real values), then we can represent it using 8 cosine waves of different frequencies.

1D DCT (8 Cosine Waves)



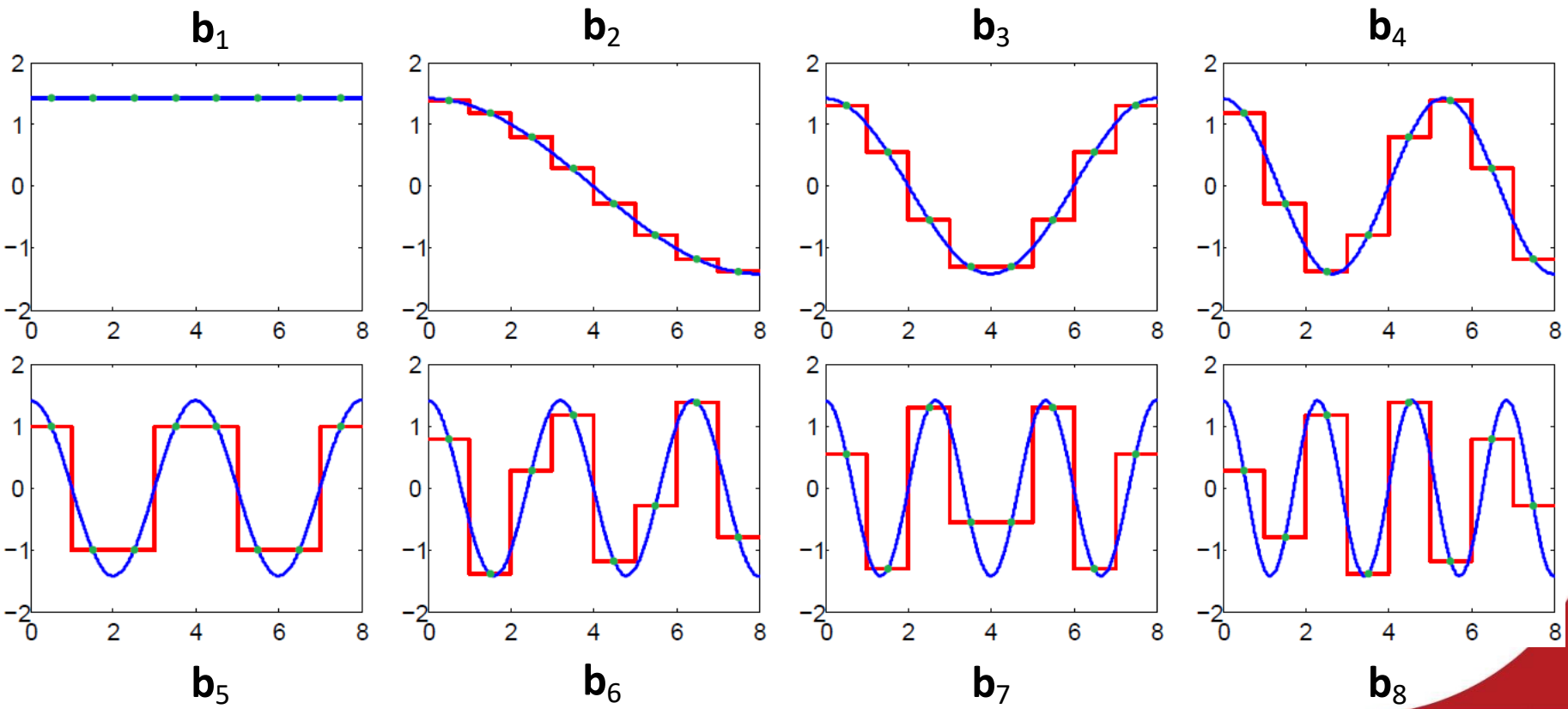
1D DCT

- The way that the mathematics work, is if we have a signal that 8 long (i.e., a vector of 8 real values), then we can represent it using 8 cosine waves of different frequencies.
- Since we deal with discrete signal, we need to digitised the continuous cosine waves by sampling

1D DCT Basis Vectors

- Each 1D DCT basis vector is obtained by sampling a continuous cosine wave in a regular grid.

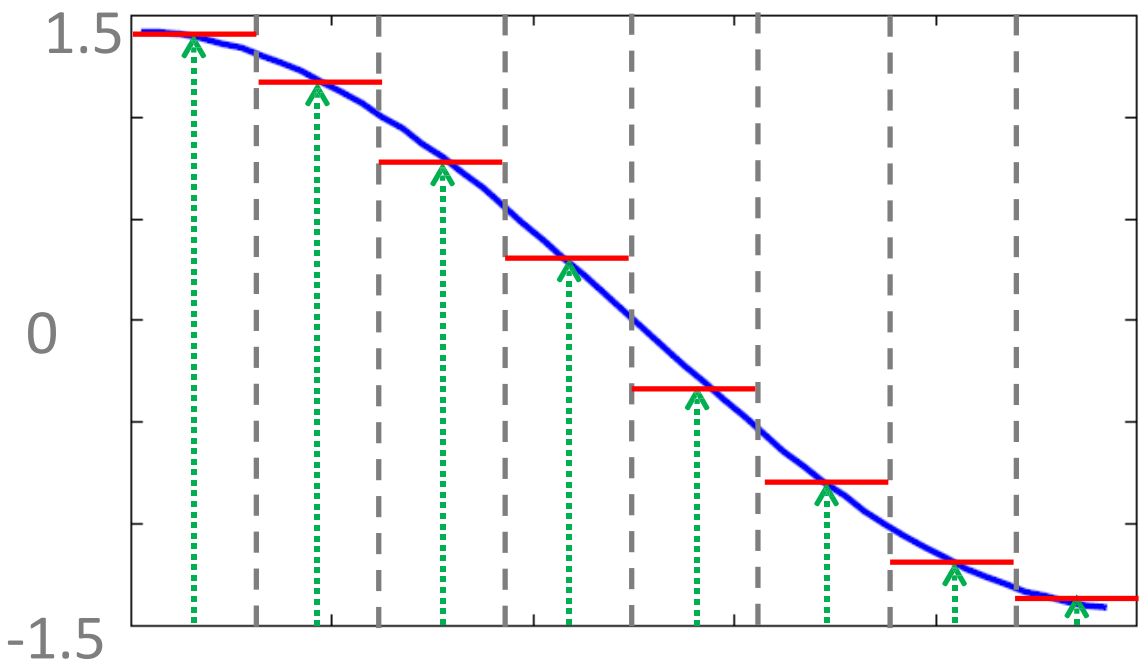
Red curves: original sinusoidal waves.
Blue curves: sampled 1d DCT basis vectors.



Sampling a Continuous Cosine Wave

Each 1D DCT basis vector is obtained by sampling a cosine wave to 8 bins:

[1.39, 1.18, 0.79, 0.28, -0.28, -0.79, -1.18, -1.39]



and normalizing the result: $\rightarrow$ $\mathbf{b}_2$ = [0.49, 0.42, 0.28, 0.10, -0.10, -0.28, -0.42, -0.49]

1D DCT Basis Vectors

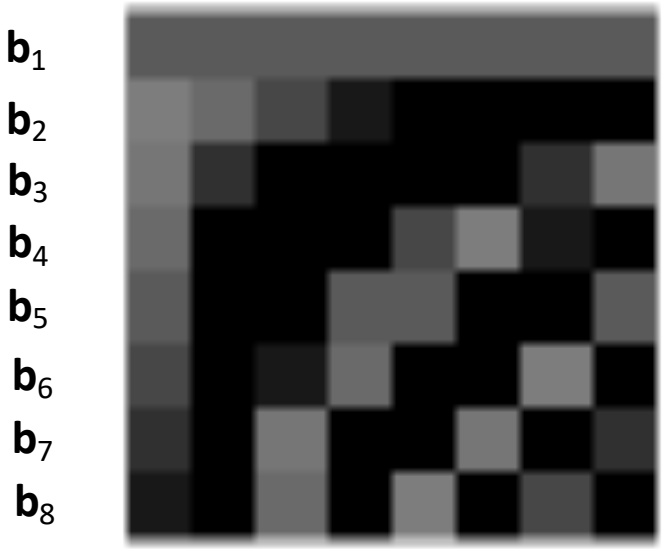
- Each 1D DCT basis vector is obtained by sampling a continuous sinusoidal wave in a regular grid.

$\mathbf{b}_1$	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
$\mathbf{b}_2$	0.49	0.42	0.28	0.10	−0.10	−0.28	−0.42	−0.49
	0.46	0.19	−0.19	−0.46	−0.46	−0.19	0.19	0.46
⋮	0.42	−0.10	−0.49	−0.28	0.28	0.49	0.10	−0.42
	0.36	−0.36	−0.36	0.36	0.36	−0.36	−0.36	0.36
	0.28	−0.49	0.10	0.42	−0.42	−0.10	0.49	−0.28
	0.19	−0.46	0.46	−0.19	−0.19	0.46	−0.46	0.19
$\mathbf{b}_8$	0.10	−0.28	0.42	−0.49	0.49	−0.42	0.28	−0.10

B

1D DCT Basis Vectors (Visualisation)

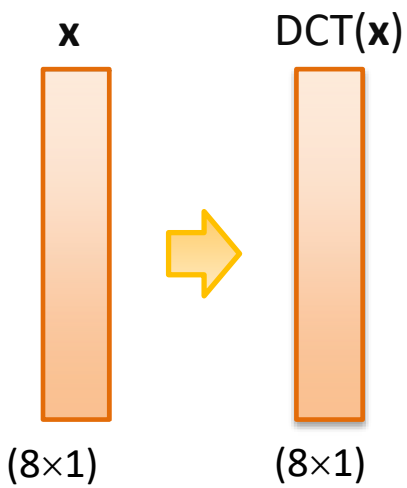
- Each 1D DCT basis vector is obtained by sampling a continuous sinusoidal wave in a regular grid.



B

1D DCT

- The way that the mathematics work, is if we have a signal that 8 long (i.e., a vector of 8 real values), then we can represent it using 8 cosine waves of different frequencies.



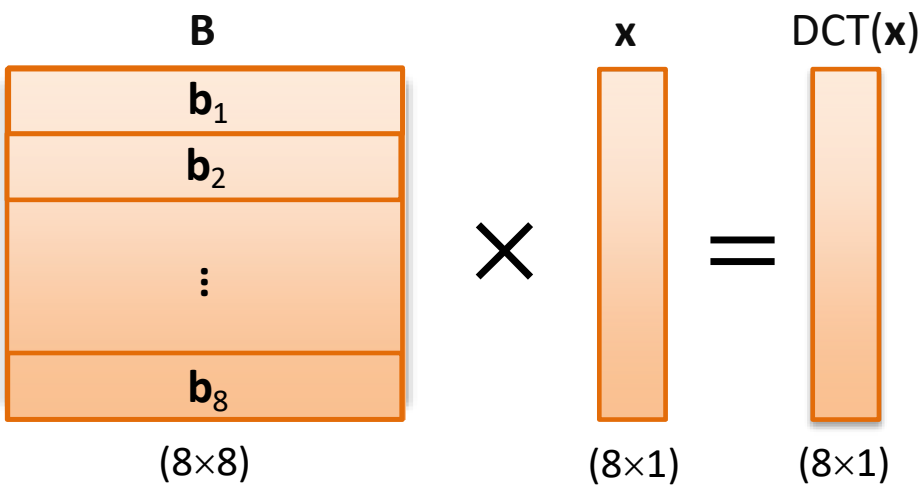
$$\mathbf{x} = (x^1, x^2, x^3, x^4, x^5, x^6, x^7, x^8)^T,$$

$$\mathrm{DCT}(\mathbf{x}) = (x'^1, x'^2, x'^3, x'^4, x'^5, x'^6, x'^7, x'^8)^T.$$

$$\mathbf{x} = x'^1\mathbf{b}_1 + x'^2\mathbf{b}_2 + x'^3\mathbf{b}_3 + x'^4\mathbf{b}_4 + x'^5\mathbf{b}_5 + x'^6\mathbf{b}_6 + x'^7\mathbf{b}_7 + x'^8\mathbf{b}_8$$

How to Calculate 1D DCT Coefficients?

- The way that the mathematics work, is if we have a signal that 8 long (i.e., a vector of 8 real values), then we can represent it using 8 cosine waves of different frequencies.



$$\mathbf{Bx} = \mathrm{DCT}(\mathbf{x})$$

Inverse 1D DCT

- How to reconstruct the original signal from 1D DCT coefficients, i.e., inverse 1D DCT?

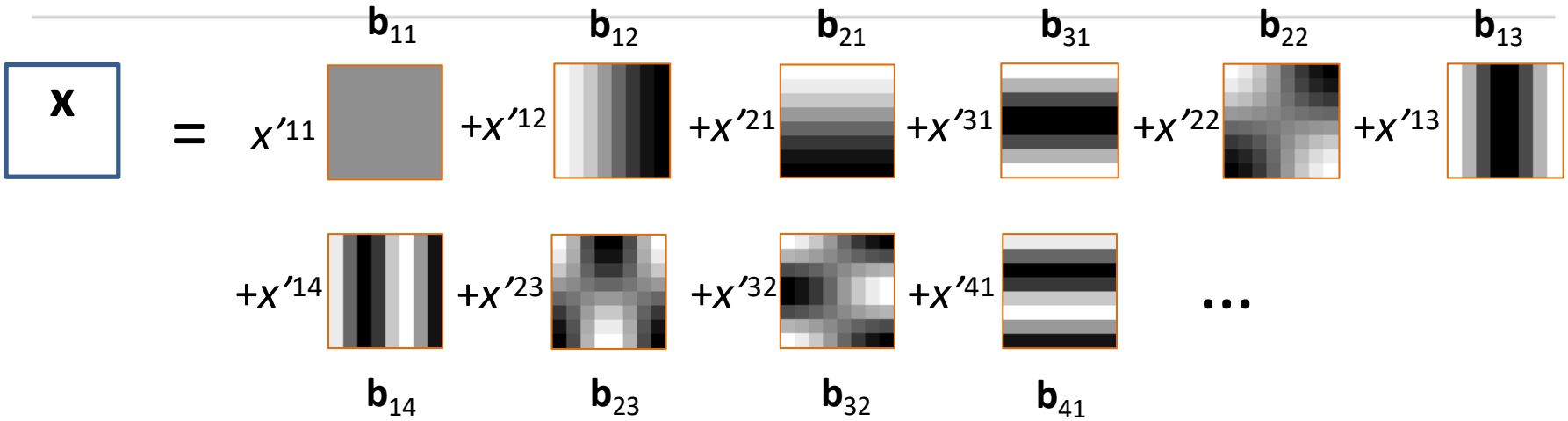
$$\mathbf{Bx} = \mathrm{DCT}(\mathbf{x})$$

$$\boldsymbol{B}^{-1}\mathbf{Bx} = \boldsymbol{B}^{-1}\mathrm{DCT}(\mathbf{x})$$

- Since B is an orthogonal matrix, $\boldsymbol{B}^{-1} = \boldsymbol{B}^{T}$

$$\mathbf{x} = \boldsymbol{B}^{T}\mathrm{DCT}(\mathbf{x})$$

2D DCT



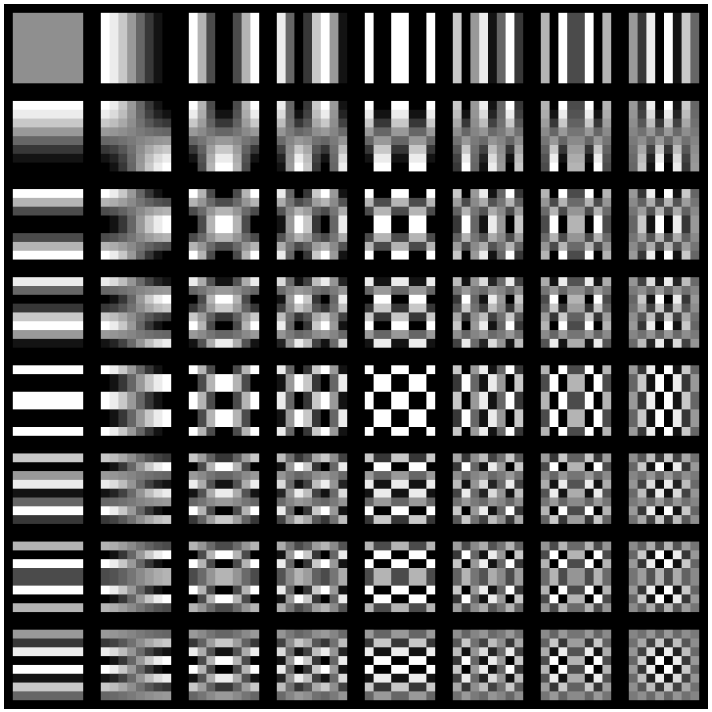
DCT (**x**) =

x'^{11}	x'^{12}	x'^{13}	x'^{14}	x'^{15}	x'^{16}	x'^{17}	x'^{18}
x'^{21}	x'^{22}	x'^{23}	x'^{24}	x'^{25}	x'^{26}	x'^{27}	x'^{28}
x'^{31}	x'^{32}	x'^{33}	x'^{34}	x'^{35}	x'^{36}	x'^{37}	x'^{38}
x'^{41}	x'^{42}	x'^{43}	x'^{44}	x'^{45}	x'^{46}	x'^{47}	x'^{48}
x'^{51}	x'^{52}	x'^{53}	x'^{54}	x'^{55}	x'^{56}	x'^{57}	x'^{58}
x'^{61}	x'^{62}	x'^{63}	x'^{64}	x'^{65}	x'^{66}	x'^{67}	x'^{68}
x'^{71}	x'^{72}	x'^{73}	x'^{74}	x'^{75}	x'^{76}	x'^{77}	x'^{78}
x'^{81}	x'^{82}	x'^{83}	x'^{84}	x'^{85}	x'^{86}	x'^{87}	x'^{88}

- An image (block) x is represented as a weighted combination of the DCT *basis images* {$\mathbf{b}_{11}$, … , $\mathbf{b}_{88}$}.
- DCT(**x**) is given as the combination weights of **x** {x'^{11}, … , x'^{88}}.

2D DCT Basis Vectors (Visualisation)

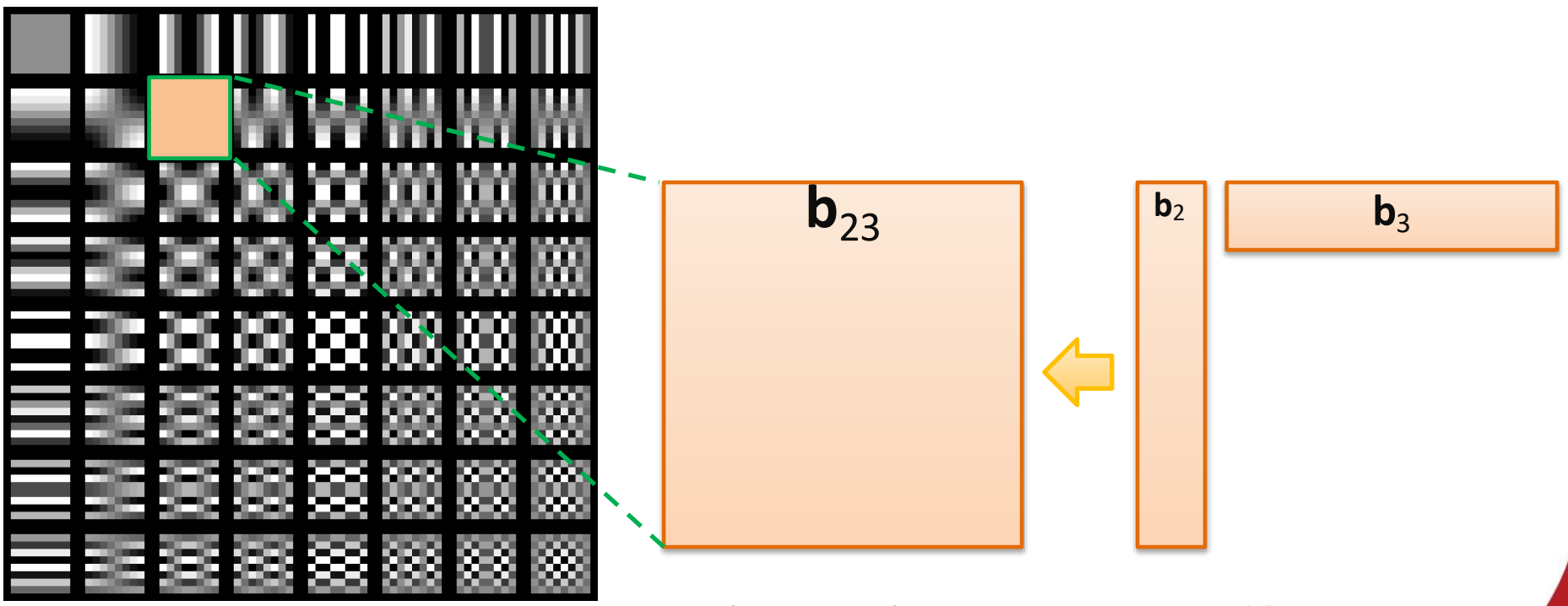
- Obtained as combinations of 1D basis vectors



How to construct the 2D DCT basis vectors?

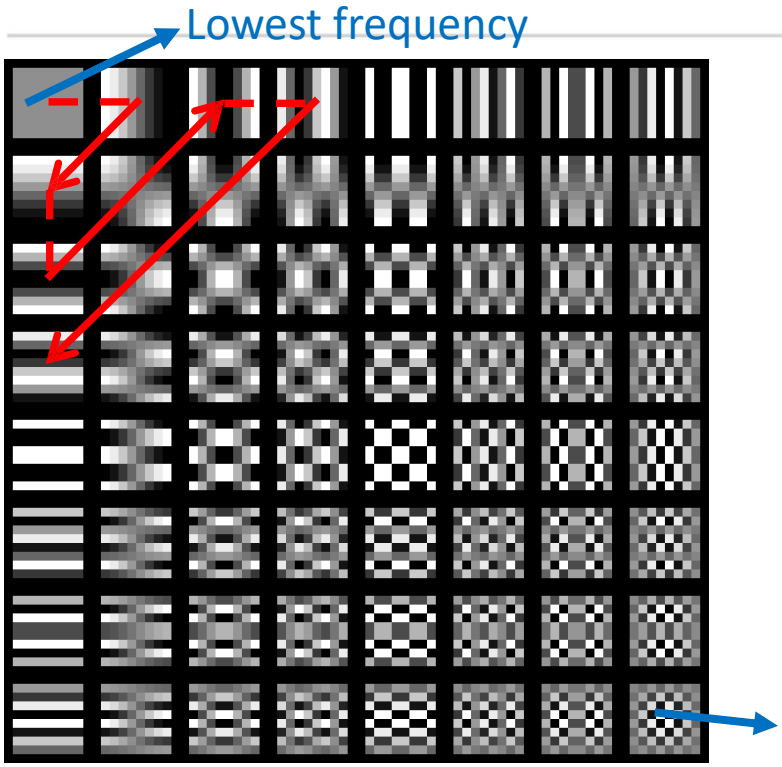
Obtained as combinations of 1d basis vectors:

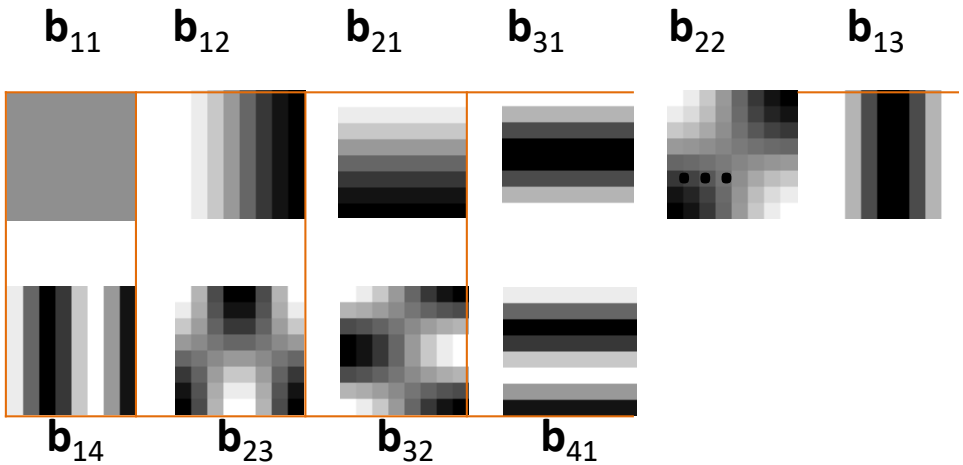
$$\mathbf{b}_{jk} = \mathbf{b}_j \mathbf{b}_k^T$$



Each 2D DCT basis matrix is *separable*: product of two vectors.

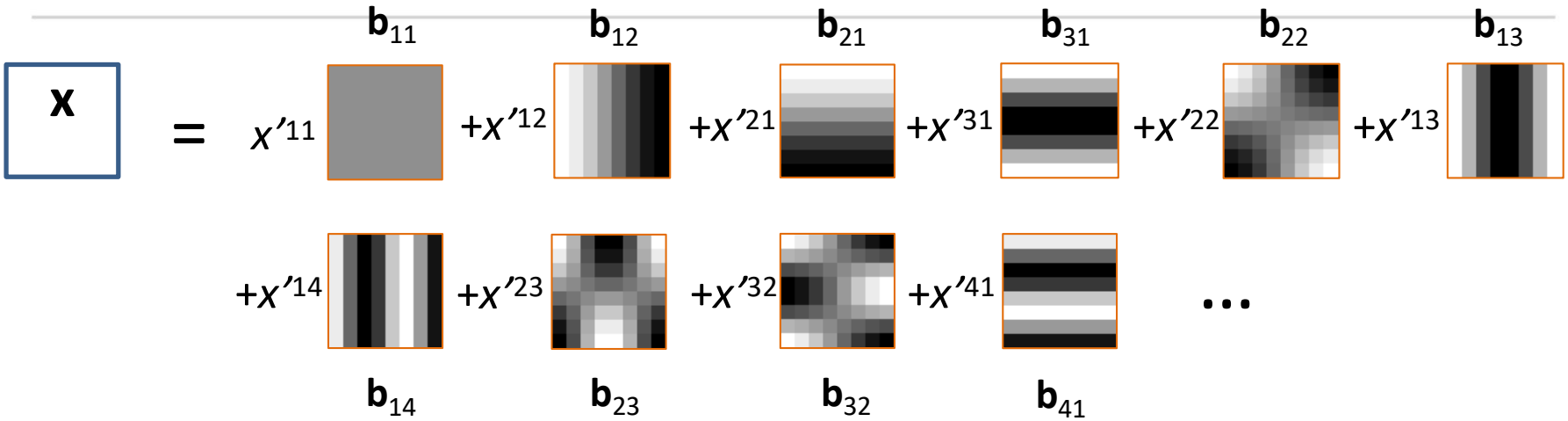
Image reconstruction using DCT





- This figure shows 64 images of size 8×8
- Each image is a 2D cosine wave pattern
- In a zig-zag order, images are in increasing frequency (# oscillations)
- These images are called DCT basis images

2D DCT

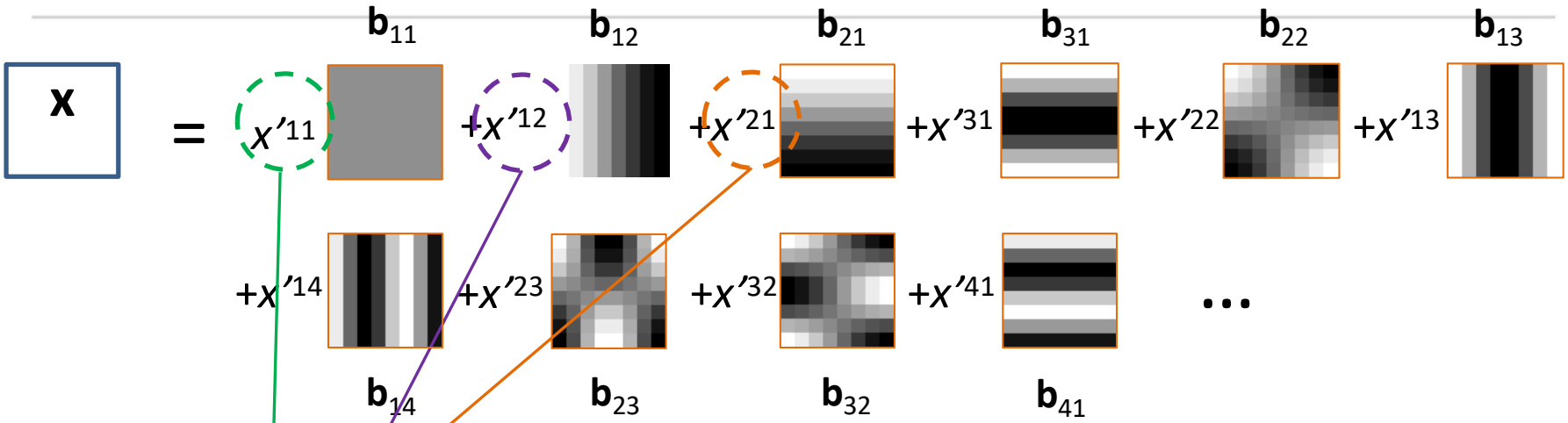


DCT (**x**) =

x'_{11}	x'_{12}	x'_{13}	x'_{14}	x'_{15}	x'_{16}	x'_{17}	x'_{18}
x'_{21}	x'_{22}	x'_{23}	x'_{24}	x'_{25}	x'_{26}	x'_{27}	x'_{28}
x'_{31}	x'_{32}	x'_{33}	x'_{34}	x'_{35}	x'_{36}	x'_{37}	x'_{38}
x'_{41}	x'_{42}	x'_{43}	x'_{44}	x'_{45}	x'_{46}	x'_{47}	x'_{48}
x'_{51}	x'_{52}	x'_{53}	x'_{54}	x'_{55}	x'_{56}	x'_{57}	x'_{58}
x'_{61}	x'_{62}	x'_{63}	x'_{64}	x'_{65}	x'_{66}	x'_{67}	x'_{68}
x'_{71}	x'_{72}	x'_{73}	x'_{74}	x'_{75}	x'_{76}	x'_{77}	x'_{78}
x'_{81}	x'_{82}	x'_{83}	x'_{84}	x'_{85}	x'_{86}	x'_{87}	x'_{88}

- An image (block) x is represented as a weighted combination of the DCT *basis images* {$\mathbf{b}_{11}$, … , $\mathbf{b}_{88}$}.
- DCT(**x**) is given as the combination weights of **x** {x'_{11}, … , x'_{88}}.

2D DCT



DCT (**x**) =

x'_{11}	x'_{12}	x'_{13}	x'_{14}	x'_{15}	x'_{16}	x'_{17}	x'_{18}
x'_{21}	x'_{22}	x'_{23}	x'_{24}	x'_{25}	x'_{26}	x'_{27}	x'_{28}
x'_{31}	x'_{32}	x'_{33}	x'_{34}	x'_{35}	x'_{36}	x'_{37}	x'_{38}
x'_{41}	x'_{42}	x'_{43}	x'_{44}	x'_{45}	x'_{46}	x'_{47}	x'_{48}
x'_{51}	x'_{52}	x'_{53}	x'_{54}	x'_{55}	x'_{56}	x'_{57}	x'_{58}
x'_{61}	x'_{62}	x'_{63}	x'_{64}	x'_{65}	x'_{66}	x'_{67}	x'_{68}
x'_{71}	x'_{72}	x'_{73}	x'_{74}	x'_{75}	x'_{76}	x'_{77}	x'_{78}
x'_{81}	x'_{82}	x'_{83}	x'_{84}	x'_{85}	x'_{86}	x'_{87}	x'_{88}

- An image (block) x is represented as a weighted combination of the DCT *basis images* $\{\mathbf{b}_{11}, \ldots, \mathbf{b}_{88}\}$.
- DCT(**x**) is given as the combination weights of **x** $\{x'_{11}, \ldots, x'_{88}\}$.

DCT Applications

- JPEG Compression
- Data Hiding
- Image Watermarking
- Etc.

SCC.366: Media Coding & Processing

2022-2023

Week 10 Lecture 2

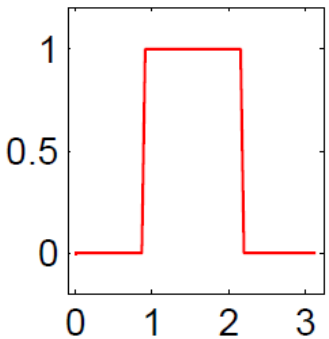
JPEG Compression

Dr. Hossein Rahmani
Senior Lecturer in Data Science
Email and Team: h.rahmani@lancaster.ac.uk

Outline

- Discrete Cosine Transform (DCT)
  - 1D DCT
  - 2D DCT
- DCT Applications, e.g., compression, data hiding
- JPEG Compression
- Summary of the main points

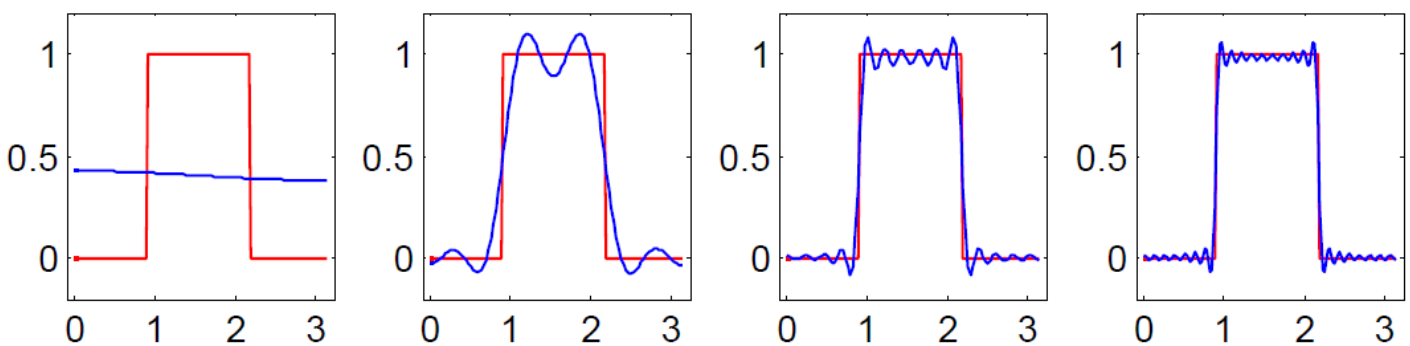
Reconstruction by Cosine Waves



x

- Cosine waves facilitate coarse-to-fine reconstruction.
  - Low-frequency (small # oscillations) components contribute to the overall shape.
  - High-frequency components add fine details.
  - Contribution of very high-frequency components are negligible.

$$\mathbf{x} \approx \tilde{\mathbf{x}} = \sum_{k=1}^{m} x'^{k} \mathbf{b}_k :$$



m=2 *m*=10 *m*=30 *m*=50

1D DCT

- The way that the mathematics work, is if we have a signal that 8 long (i.e., a vector of 8 real values), then we can represent it using 8 cosine waves of different frequencies.
- Since we deal with discrete signal, we need to digitised the continuous cosine waves by sampling

1D DCT Basis Vectors

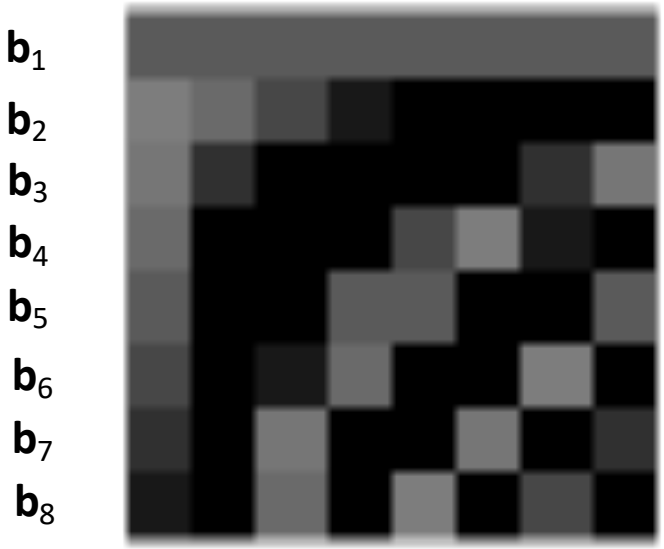
- Each 1D DCT basis vector is obtained by sampling a continuous sinusoidal wave in a regular grid.

$\mathbf{b}_1$	0.36	0.36	0.36	0.36	0.36	0.36	0.36	0.36
$\mathbf{b}_2$	0.49	0.42	0.28	0.10	−0.10	−0.28	−0.42	−0.49
	0.46	0.19	−0.19	−0.46	−0.46	−0.19	0.19	0.46
⋮	0.42	−0.10	−0.49	−0.28	0.28	0.49	0.10	−0.42
	0.36	−0.36	−0.36	0.36	0.36	−0.36	−0.36	0.36
	0.28	−0.49	0.10	0.42	−0.42	−0.10	0.49	−0.28
	0.19	−0.46	0.46	−0.19	−0.19	0.46	−0.46	0.19
$\mathbf{b}_8$	0.10	−0.28	0.42	−0.49	0.49	−0.42	0.28	−0.10

B

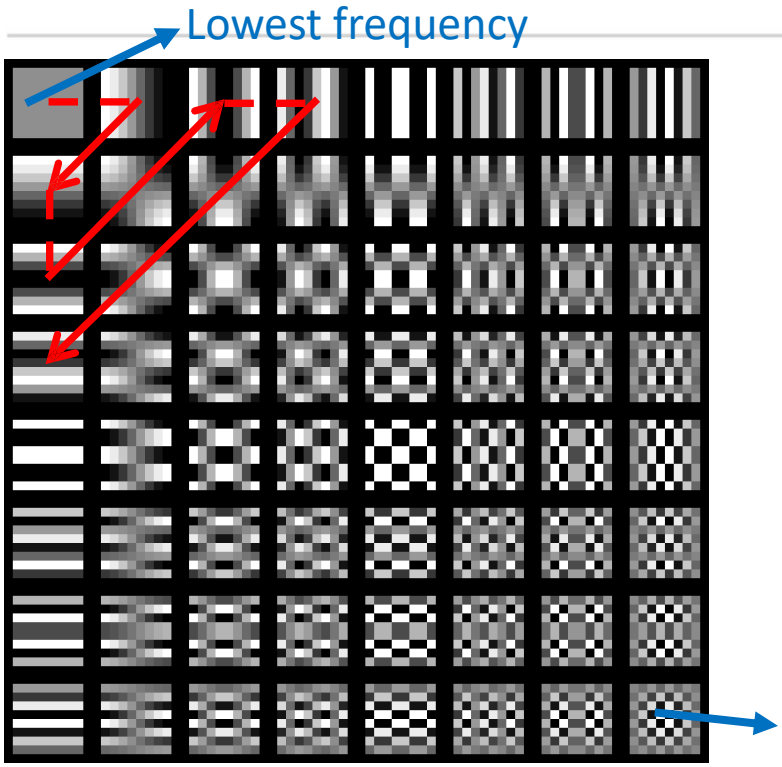
1D DCT Basis Vectors (Visualisation)

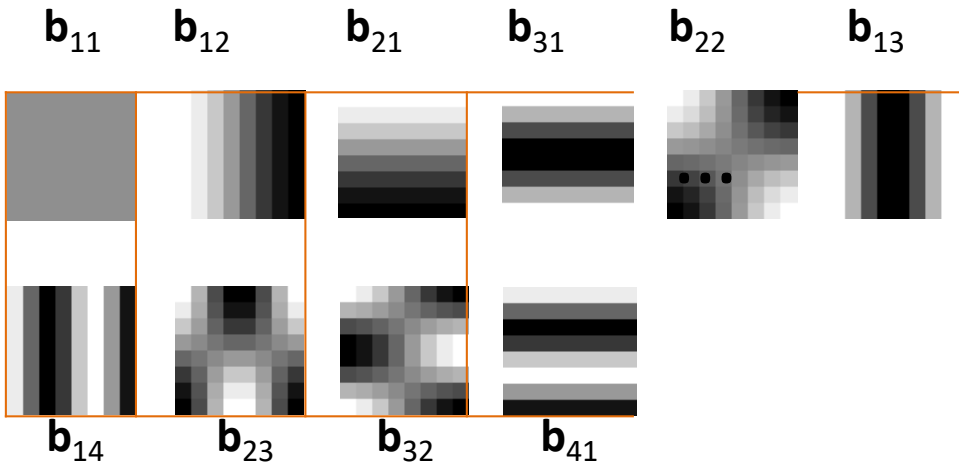
- Each 1D DCT basis vector is obtained by sampling a continuous sinusoidal wave in a regular grid.



B

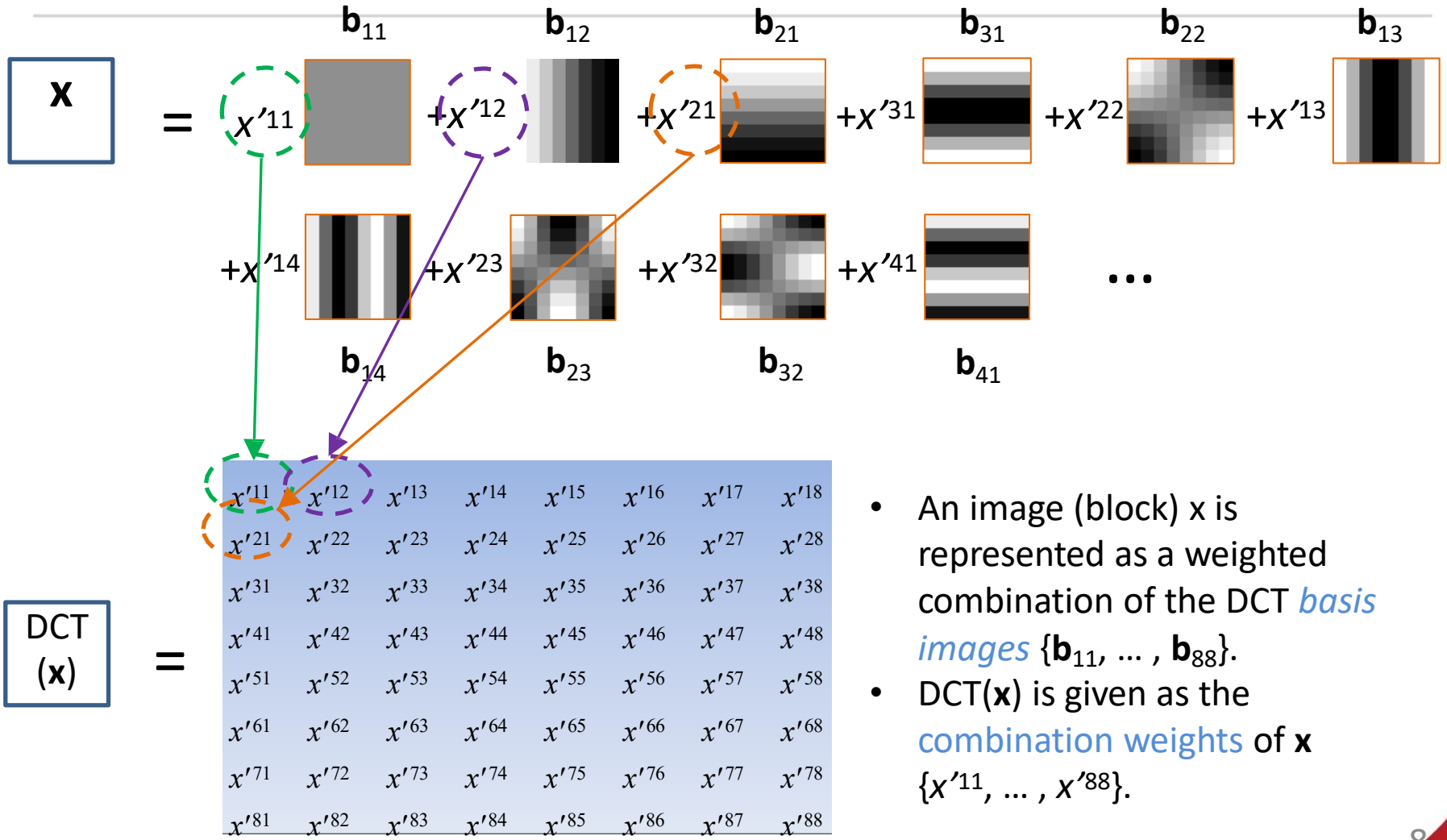
Image reconstruction using DCT





- This figure shows 64 images of size 8×8
- Each image is a 2D cosine wave pattern
- In a zig-zag order, images are in increasing frequency (# oscillations)
- These images are called DCT basis images

2D DCT



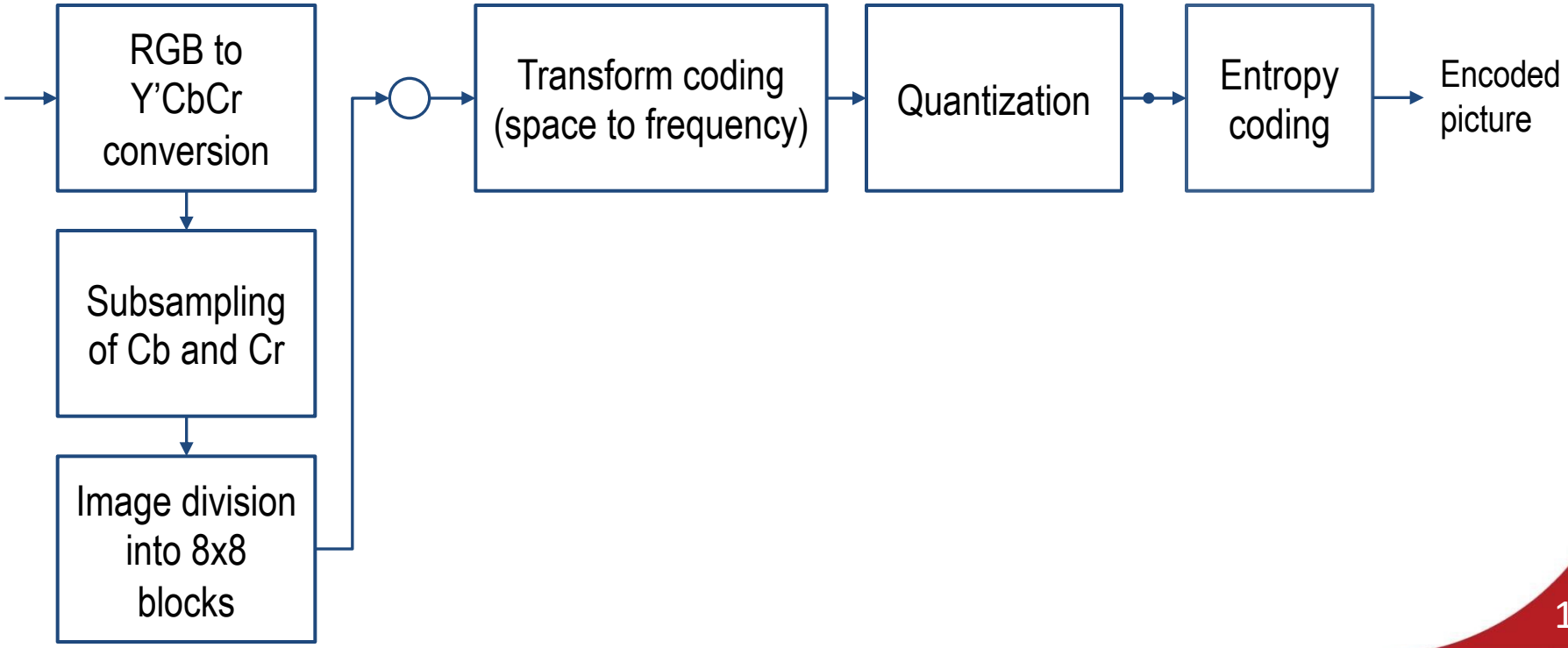
- An image (block) x is represented as a weighted combination of the DCT *basis images* $\{\mathbf{b}_{11}, \ldots, \mathbf{b}_{88}\}$.
- DCT(**x**) is given as the combination weights of **x** $\{x'_{11}, \ldots, x'_{88}\}$.

DCT Applications

- JPEG Compression
- Data Hiding
- Image Watermarking
- Etc.

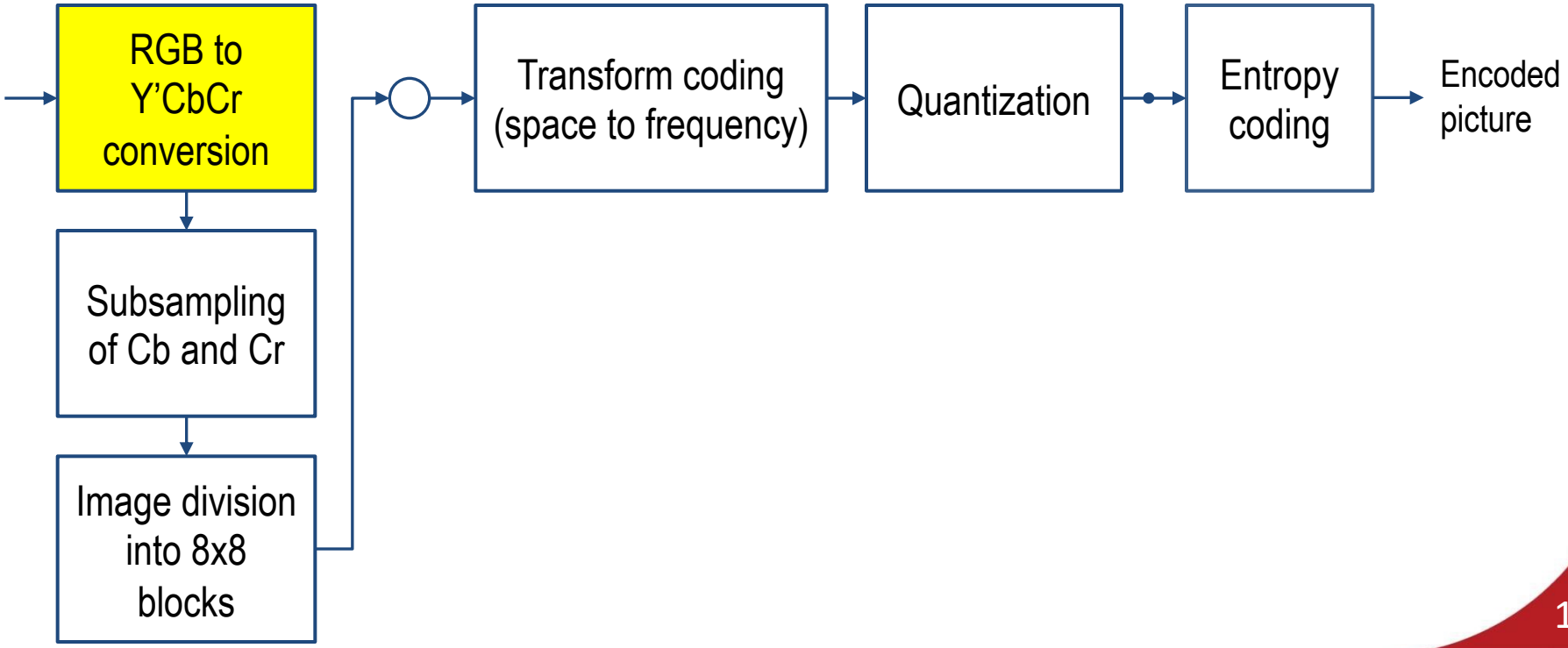
JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding

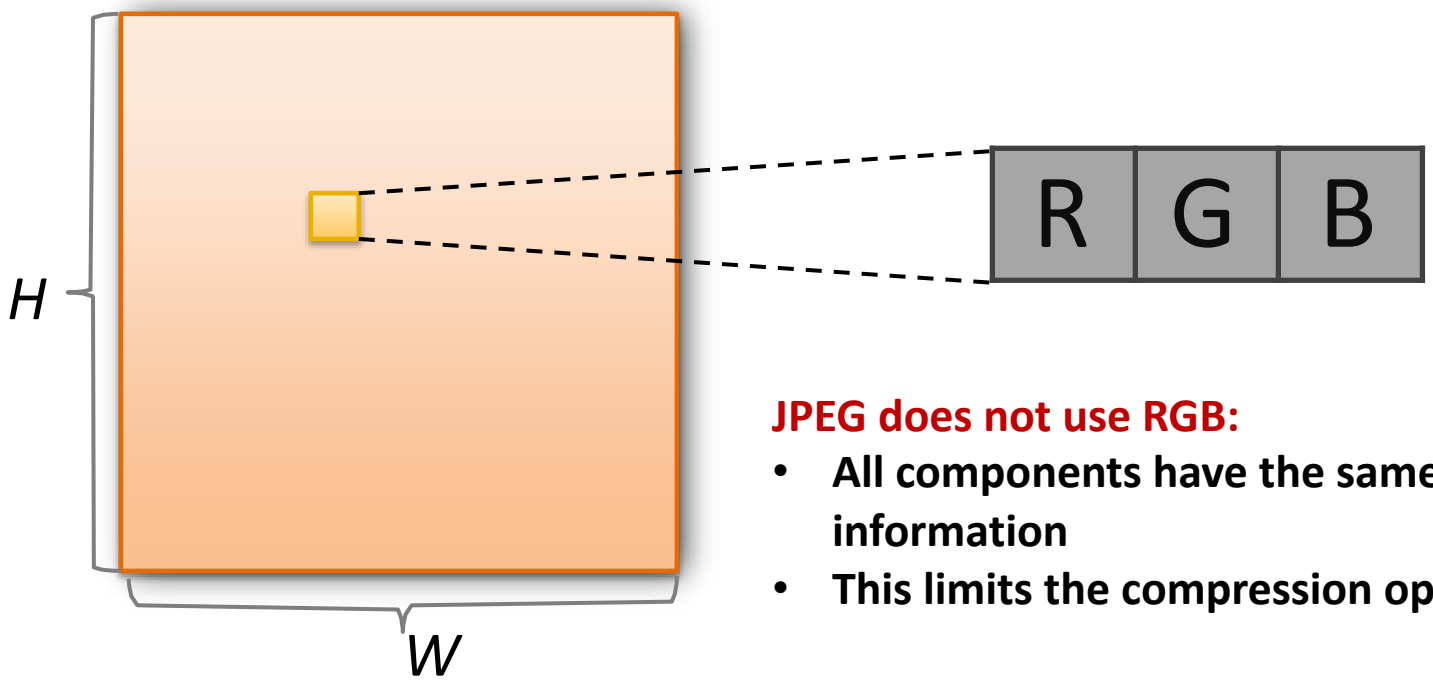


JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding



RGB vs. YCbCr



JPEG does not use RGB:

- **All components have the same level of information**
- **This limits the compression opportunity**

- The downside is that R, G and B components are all EQUALLY IMPORTANT
- Equal Importance leaves a little room for compression
- We need a new mapping which provides three components with un-equal importance

Color Model in JPEG: YC_bC_r

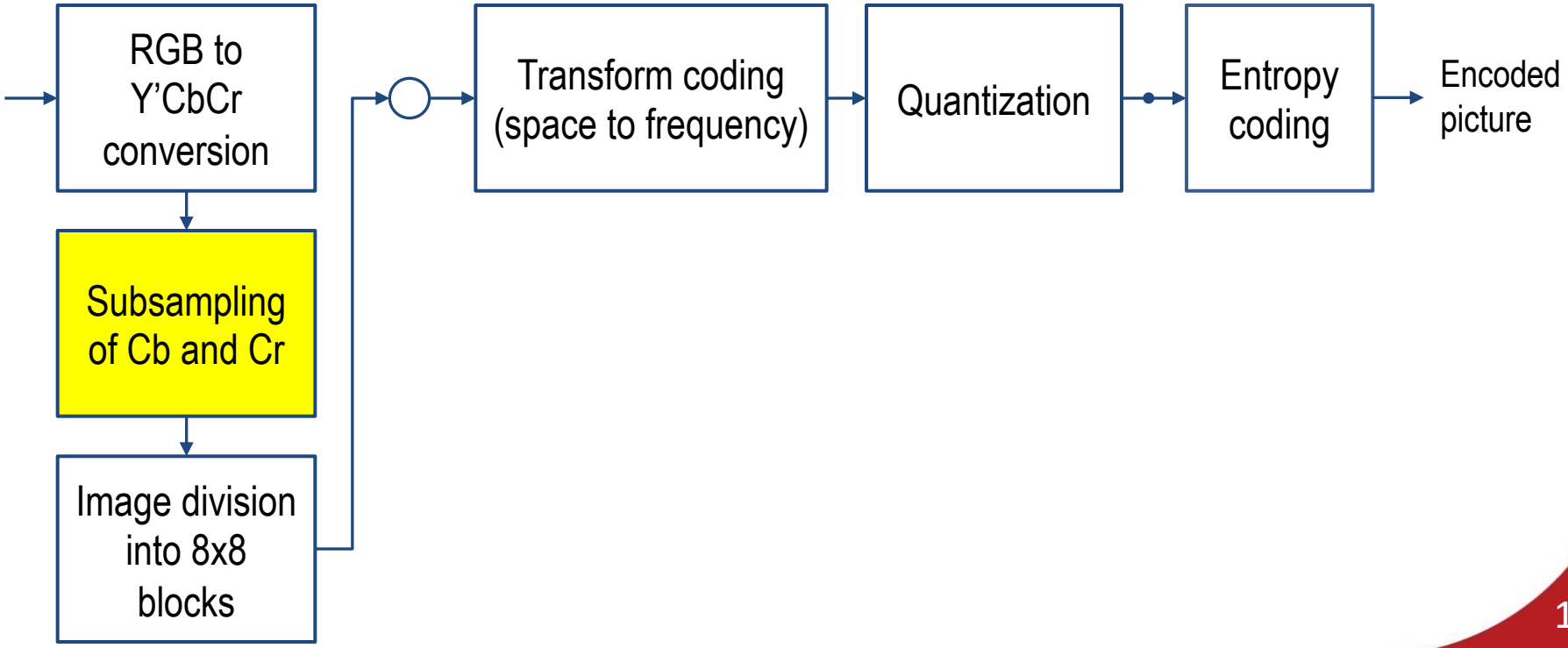
$$Y = 0.299R + 0.587G + 0.114B$$
$$C_b = -0.168R - 0.331G + 0.500B + 128$$
$$C_r = 0.500R - 0.419G - 0.081B + 128$$

Y C_b C_r

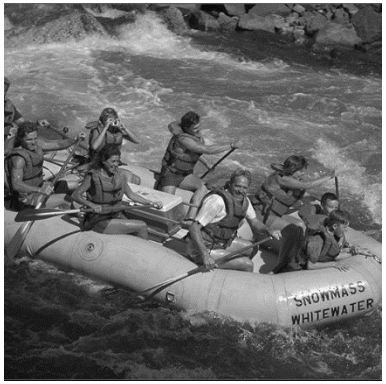
- Roughly, YC_bC_r model separates brightness (Y) from chromaticity (C_bC_r).
- The human is more sensitive to the luminosity (brightness) variations than color (chromaticity) variations.

JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding



Sub-sampling Color Information in YC_bC_r



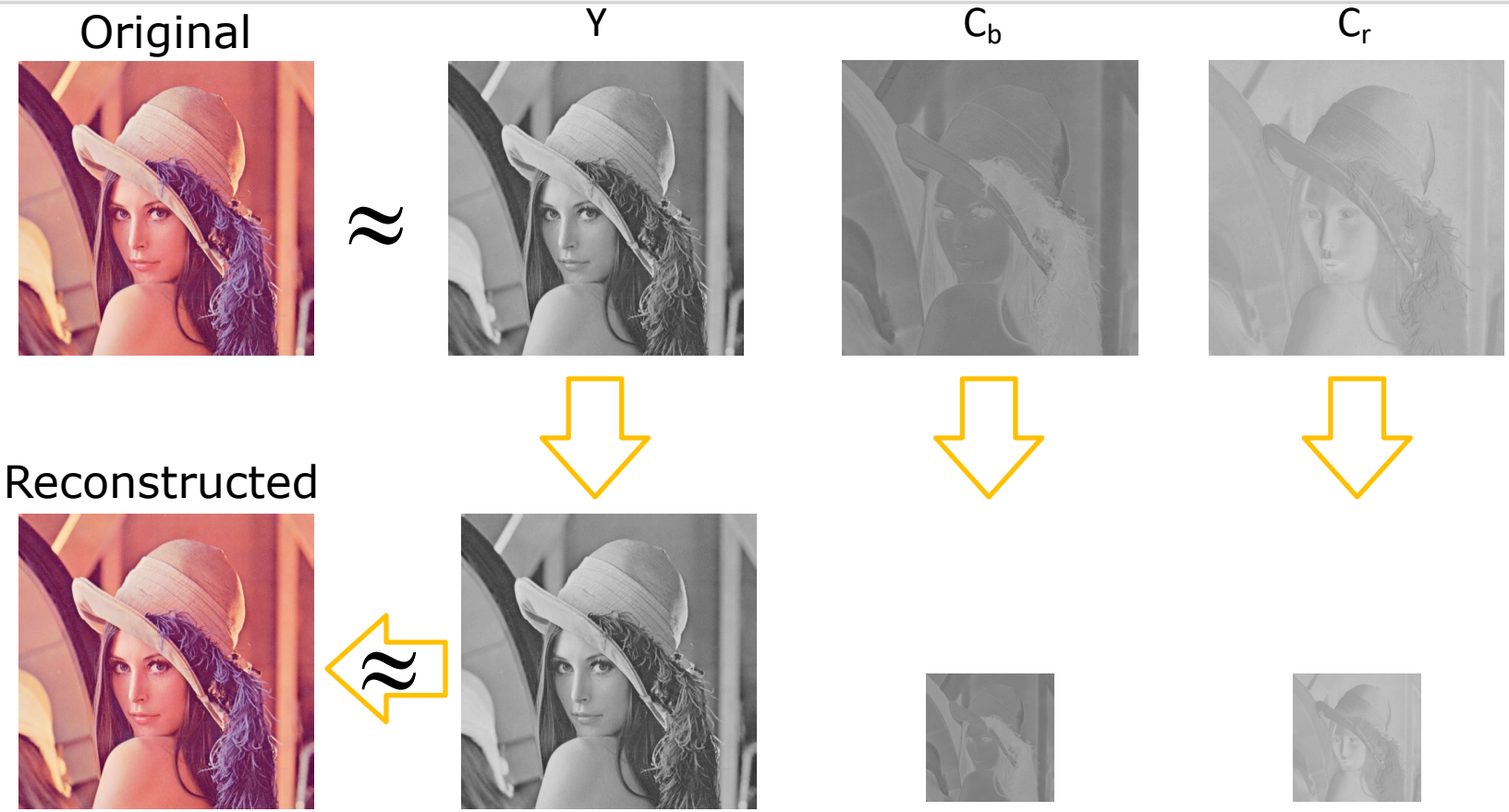
Y

C_b

C_r

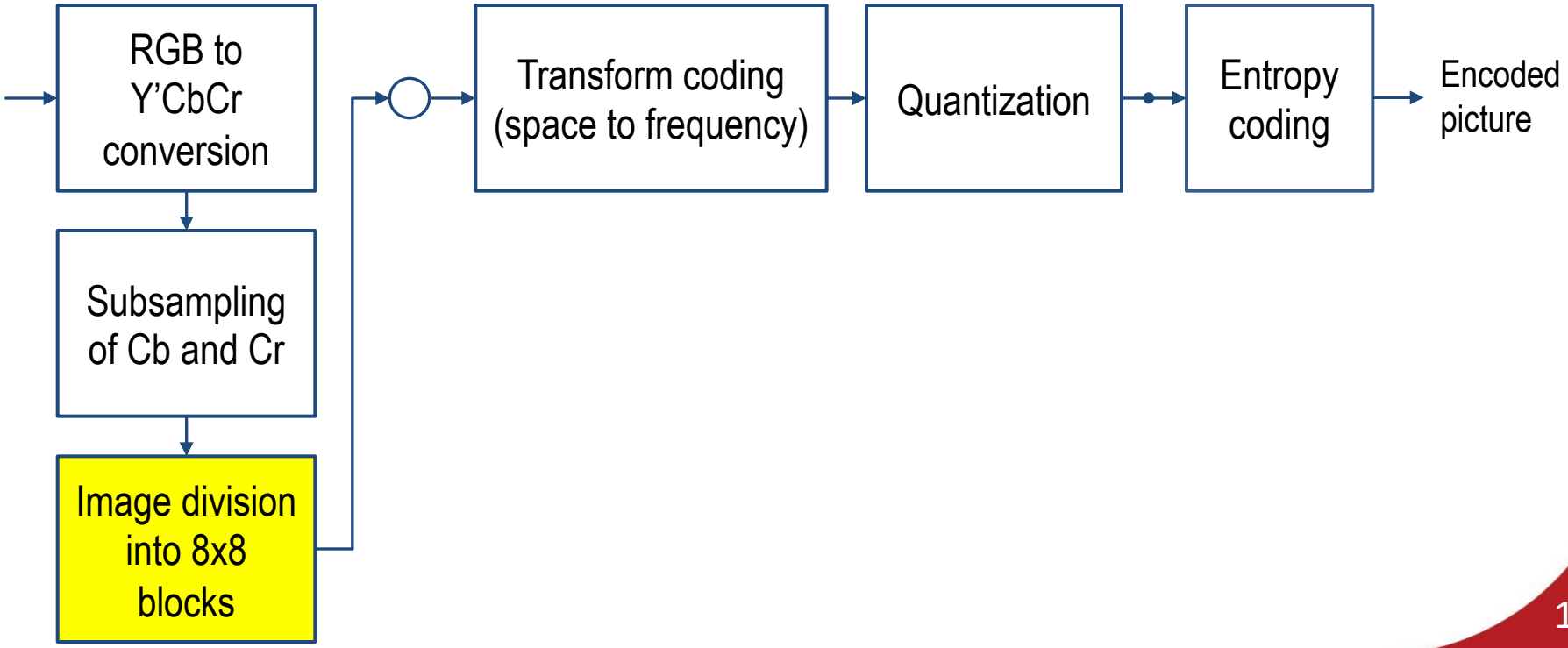
- JPEG encodes each color channel separately
- Chromaticity components (C_b and C_r) are sub-sampled before encoding

Sub-sampling Color Information in YC_bC_r



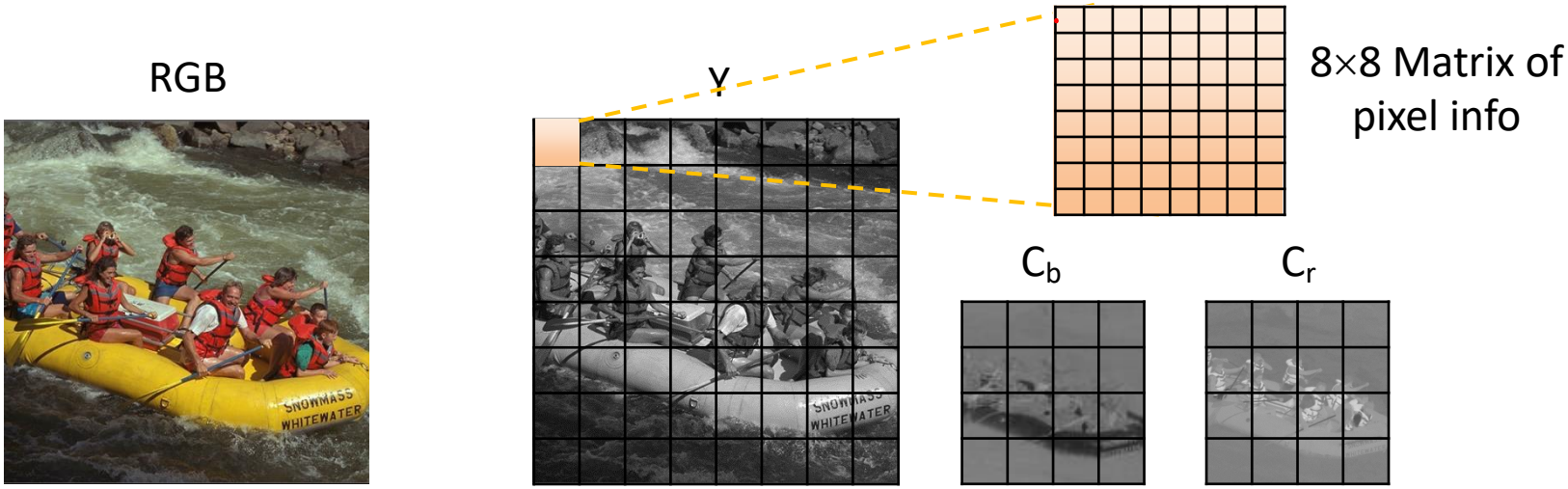
JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding



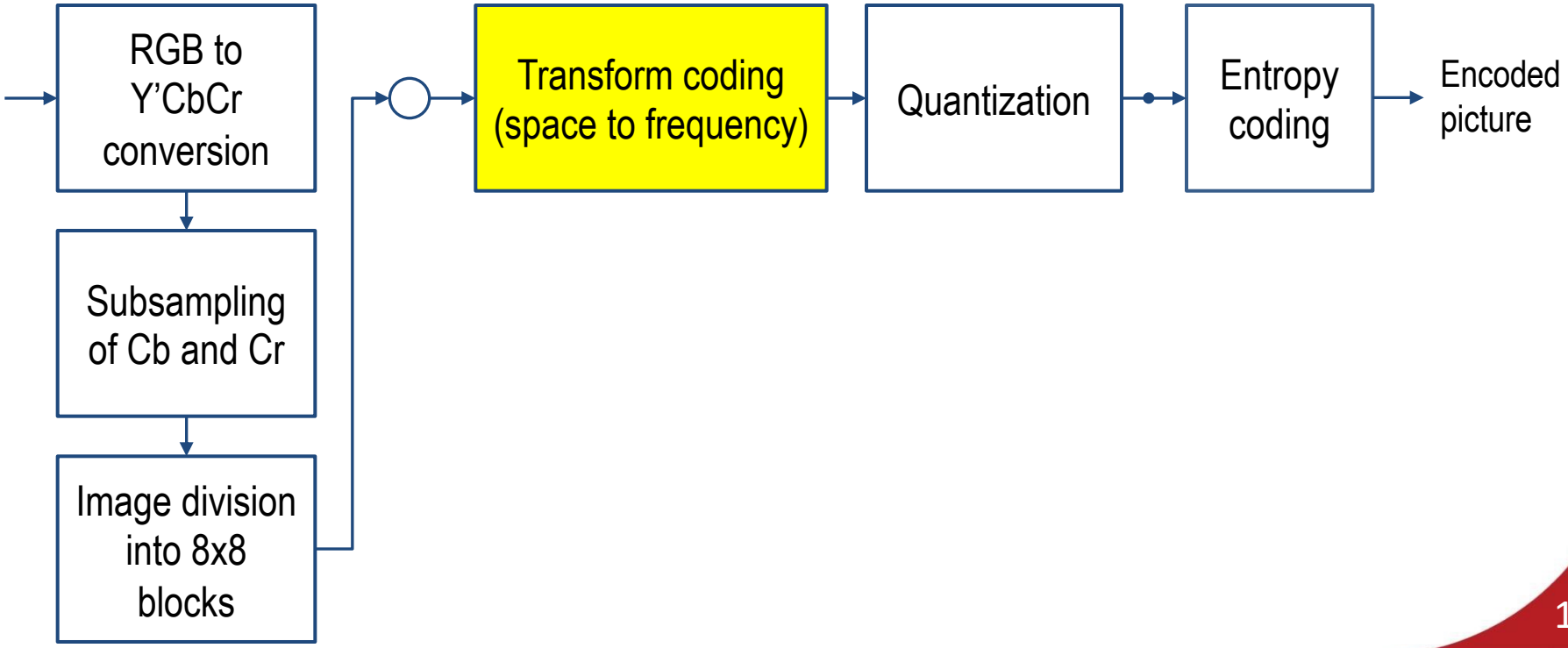
Dividing Image into 8*8 Blocks

- In general, an (big) image is decomposed into 8×8 patches (sub-windows)
- Each channel (e.g., Y) is divided into non-overlapping 8×8-sized blocks
- Each patch is processed/encoded independently

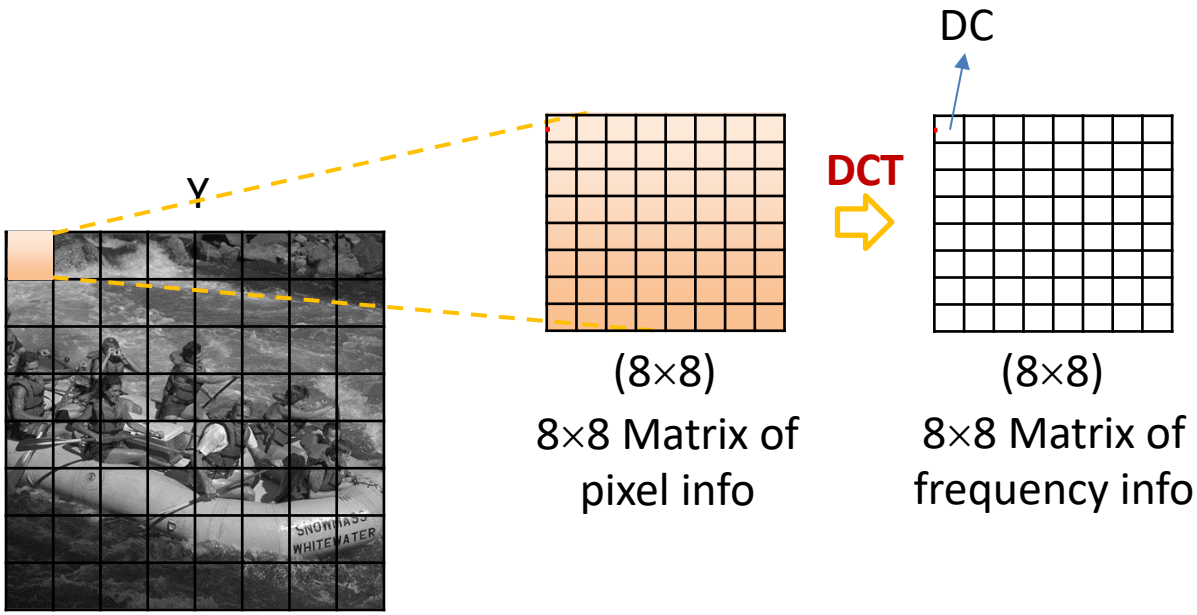


JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding

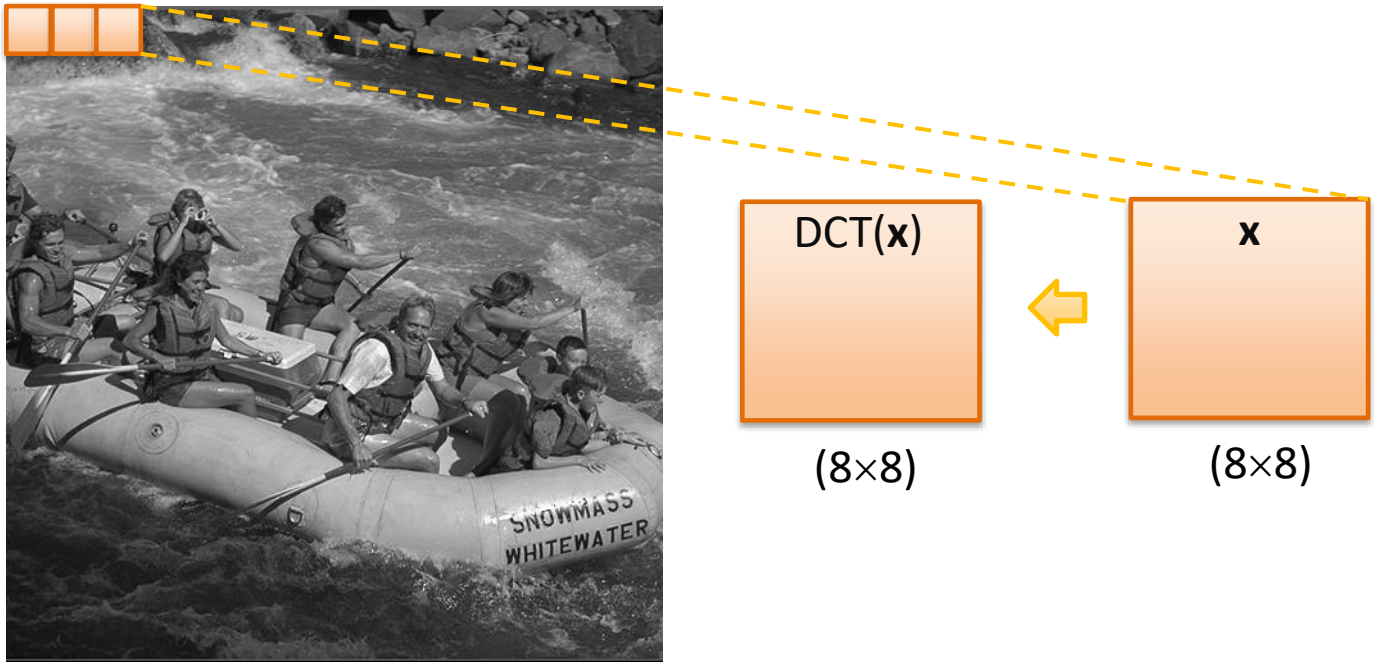


Discrete Cosine Transform (DCT)



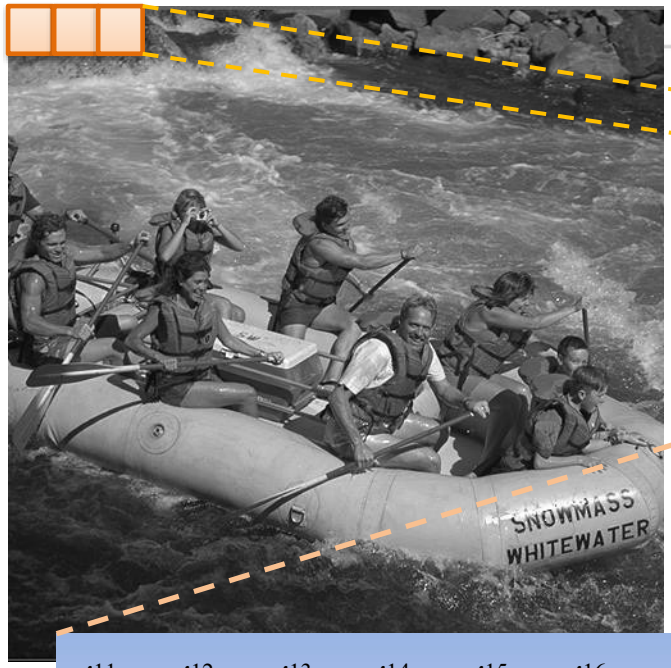
- All components in the resultant matrix are equally important, which makes it hard to do the compression
  - How to change the game?
  - Transformation to a new coordinate where components have different meaning
- Each block is independently mapped by the discrete cosine transform (DCT)
- Its inverse is called “inverse DCT” or “iDCT”

Block-wise encoding



- Each channel (e.g., Y) is divided into non-overlapping 8×8-sized blocks
  - Computational complexity gain
  - Applies to images of any size
  - Block artifacts at high-compression ratios (shown later)
- Each block is independently mapped by the discrete cosine transform (DCT)

Block-wise encoding



DCT(**x**)

(8×8)

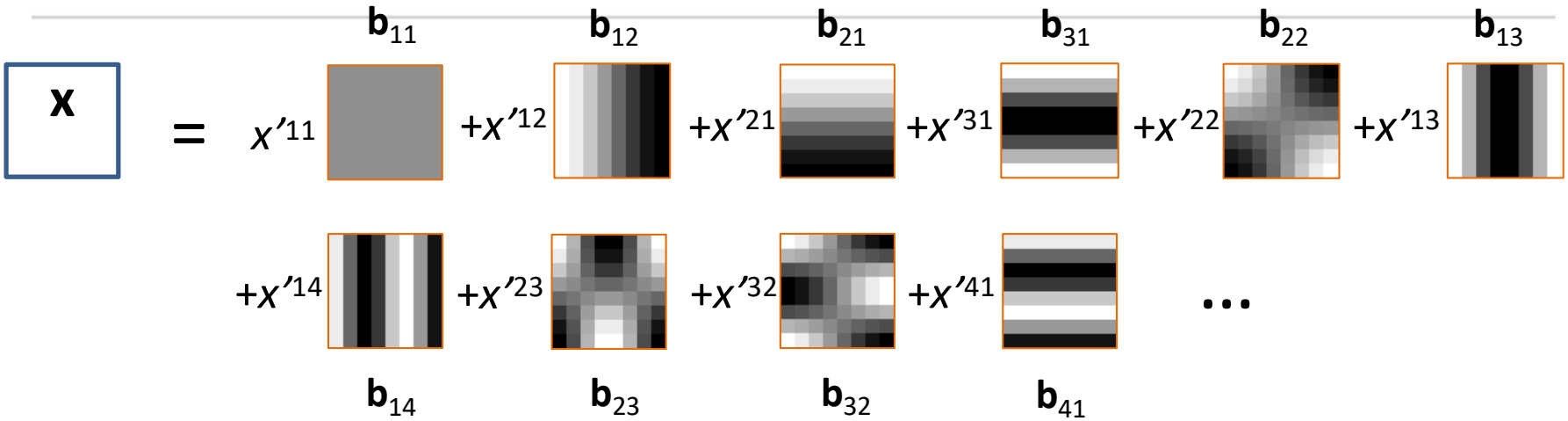
x

(8×8)

x'^{11}	x'^{12}	x'^{13}	x'^{14}	x'^{15}	x'^{16}	x'^{17}	x'^{18}
x'^{21}	x'^{22}	x'^{23}	x'^{24}	x'^{25}	x'^{26}	x'^{27}	x'^{28}
x'^{31}	x'^{32}	x'^{33}	x'^{34}	x'^{35}	x'^{36}	x'^{37}	x'^{38}
x'^{41}	x'^{42}	x'^{43}	x'^{44}	x'^{45}	x'^{46}	x'^{47}	x'^{48}
x'^{51}	x'^{52}	x'^{53}	x'^{54}	x'^{55}	x'^{56}	x'^{57}	x'^{58}
x'^{61}	x'^{62}	x'^{63}	x'^{64}	x'^{65}	x'^{66}	x'^{67}	x'^{68}
x'^{71}	x'^{72}	x'^{73}	x'^{74}	x'^{75}	x'^{76}	x'^{77}	x'^{78}
x'^{81}	x'^{82}	x'^{83}	x'^{84}	x'^{85}	x'^{86}	x'^{87}	x'^{88}

x^{11}	x^{12}	x^{13}	x^{14}	x^{15}	x^{16}	x^{17}	x^{18}
x^{21}	x^{22}	x^{23}	x^{24}	x^{25}	x^{26}	x^{27}	x^{28}
x^{31}	x^{32}	x^{33}	x^{34}	x^{35}	x^{36}	x^{37}	x^{38}
x^{41}	x^{42}	x^{43}	x^{44}	x^{45}	x^{46}	x^{47}	x^{48}
x^{51}	x^{52}	x^{53}	x^{54}	x^{55}	x^{56}	x^{57}	x^{58}
x^{61}	x^{62}	x^{63}	x^{64}	x^{65}	x^{66}	x^{67}	x^{68}
x^{71}	x^{72}	x^{73}	x^{74}	x^{75}	x^{76}	x^{77}	x^{78}
x^{81}	x^{82}	x^{83}	x^{84}	x^{85}	x^{86}	x^{87}	x^{88}

Discrete Cosine Transform (DCT)



DCT (**x**) =

x'_{11}	x'_{12}	x'_{13}	x'_{14}	x'_{15}	x'_{16}	x'_{17}	x'_{18}
x'_{21}	x'_{22}	x'_{23}	x'_{24}	x'_{25}	x'_{26}	x'_{27}	x'_{28}
x'_{31}	x'_{32}	x'_{33}	x'_{34}	x'_{35}	x'_{36}	x'_{37}	x'_{38}
x'_{41}	x'_{42}	x'_{43}	x'_{44}	x'_{45}	x'_{46}	x'_{47}	x'_{48}
x'_{51}	x'_{52}	x'_{53}	x'_{54}	x'_{55}	x'_{56}	x'_{57}	x'_{58}
x'_{61}	x'_{62}	x'_{63}	x'_{64}	x'_{65}	x'_{66}	x'_{67}	x'_{68}
x'_{71}	x'_{72}	x'_{73}	x'_{74}	x'_{75}	x'_{76}	x'_{77}	x'_{78}
x'_{81}	x'_{82}	x'_{83}	x'_{84}	x'_{85}	x'_{86}	x'_{87}	x'_{88}

- An image (block) x is represented as a weighted combination of the DCT *basis images* {$\mathbf{b}_{11}$, … , $\mathbf{b}_{88}$}.
- DCT(**x**) is given as the combination weights of **x** {x'_{11}, … , x'_{88}}.

DCT example

147	145	137	194	238	221	200	197
149	145	138	207	232	217	206	198
148	144	147	211	234	231	205	188
148	145	145	221	253	243	197	189
149	142	152	236	253	235	186	188
146	138	168	253	253	229	188	184
145	137	186	253	250	221	189	180
144	137	216	252	239	217	182	180

15.31	−1.74	−208	0.70	0.84	0.04	−0.06	−0.26
−0.30	−0.35	0.62	0.42	−0.07	−0.24	−0.26	−0.02
−0.07	0.12	0.07	−0.28	−0.09	0.08	0.07	0.17
−0.02	−0.04	−0.04	0.07	0.15	−0.07	−0.07	−0.05
0.01	0.02	−0.01	0.03	−0.02	−0.11	0.08	0.02
0.02	−0.04	−0.03	0.09	−0.03	−0.03	−0.01	0.02
0.00	0.02	−0.02	−0.02	−0.05	0.02	0.05	0.00
−0.04	0.02	0.02	0.00	−0.02	0.01	0.01	0.01

DCT(**x**)

DCT example

15.31	−1.74	−208	0.70	0.84	0.04	−0.06	−0.26
−0.30	−0.35	0.62	0.42	−0.07	−0.24	−0.26	−0.02
−0.07	0.12	0.07	−0.28	−0.09	0.08	0.07	0.17
−0.02	−0.04	−0.04	0.07	0.15	−0.07	−0.07	−0.05
0.01	0.02	−0.01	0.03	−0.02	−0.11	0.08	0.02
0.02	−0.04	−0.03	0.09	−0.03	−0.03	−0.01	0.02
0.00	0.02	−0.02	−0.02	−0.05	0.02	0.05	0.00
−0.04	0.02	0.02	0.00	−0.02	0.01	0.01	0.01

DCT(**x**)

DCT(**x**)

Removing some high-frequency components does not severely reduce the perceived quality of the image.

15.31	−1.74	−208	0.70	0.84	0.04	−0.06	−0.26
−0.30	−0.35	0.62	0.42	−0.07	−0.24	−0.26	0
−0.07	0.12	0.07	−0.28	−0.09	0.08	0	0
−0.02	−0.04	−0.04	0.07	0.15	0	0	0
0.01	0.02	−0.01	0.03	−0.02	0	0	0
0.02	−0.04	−0.03	0.09	0	0	0	0
0.00	0.02	−0.02	0	0	0	0	0
−0.04	0.02	0	0	0	0	0	0

x ≈ $\underline{\mathbf{x}}_{40}$

Inverse DCT

= $\mathrm{DCT}^{-1}(\underline{\mathrm{DCT}(\mathbf{x})})$: Recon. based on 40 components.

$\underline{\mathrm{DCT}(\mathbf{x})}$

DCT example

x

≈

$\underline{\mathbf{x}}_{40}$

DCT^{-1}

15.31	−1.74	−208	0.70	0.84	0.04	−0.06	−0.26
−0.30	−0.35	0.62	0.42	−0.07	−0.24	−0.26	0
−0.07	0.12	0.07	−0.28	−0.09	0.08	0	0
−0.02	−0.04	−0.04	0.07	0.15	0	0	0
0.01	0.02	−0.01	0.03	−0.02	0	0	0
0.02	−0.04	−0.03	0.09	0	0	0	0
0.00	0.02	−0.02	0	0	0	0	0
−0.04	0.02	0	0	0	0	0	0

$\underline{\mathrm{DCT}(\mathbf{x})}$

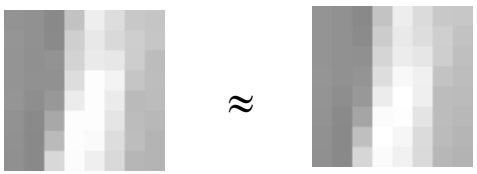
$\underline{\mathbf{x}}_{20}$

DCT^{-1}

15.31	−1.74	−208	0.70	0.84	0.04	0	0
−0.30	−0.35	0.62	0.42	−0.07	0	0	0
−0.07	0.12	0.07	−0.28	0	0	0	0
−0.02	−0.04	−0.04	0	0	0	0	0
0.01	0.02	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\underline{\mathrm{DCT}(\mathbf{x})}$

DCT example



$\mathbf{x} \approx \underline{\mathbf{x}}_{40}$

DCT^{-1}

15.31	−1.74	−208	0.70	0.84	0.04	−0.06	−0.26
−0.30	−0.35	0.62	0.42	−0.07	−0.24	−0.26	0
−0.07	0.12	0.07	−0.28	−0.09	0.08	0	0
−0.02	−0.04	−0.04	0.07	0.15	0	0	0
0.01	0.02	−0.01	0.03	−0.02	0	0	0
0.02	−0.04	−0.03	0.09	0	0	0	0
0.00	0.02	−0.02	0	0	0	0	0
−0.04	0.02	0	0	0	0	0	0

DCT^{-1}

0	0	−208	0.70	0.84	0.04	−0.06	−0.26
0	0	0.62	0.42	−0.07	−0.24	−0.26	−0.02
0	0.12	0.07	−0.28	−0.09	0.08	0.07	0.17
−0.02	−0.04	−0.04	0.07	0.15	−0.07	−0.07	−0.05
0.01	0.02	−0.01	0.03	−0.02	−0.11	0.08	0.02
0.02	−0.04	−0.03	0.09	−0.03	−0.03	−0.01	0.02
0.00	0.02	−0.02	−0.02	−0.05	0.02	0.05	0.00
−0.04	0.02	0.02	0.00	−0.02	0.01	0.01	0.01

- Low-frequency components are important: Removing only the first 5 components (in zig-zag order) lead to a significantly different image.

DCT Summary

$$\boxed{\begin{matrix}\text{DCT}\\(\mathbf{x})\end{matrix}} = \begin{bmatrix} x'^{11} & x'^{12} & x'^{13} & x'^{14} & x'^{15} & x'^{16} & x'^{17} & x'^{18} \\ x'^{21} & x'^{22} & x'^{23} & x'^{24} & x'^{25} & x'^{26} & x'^{27} & x'^{28} \\ x'^{31} & x'^{32} & x'^{33} & x'^{34} & x'^{35} & x'^{36} & x'^{37} & x'^{38} \\ x'^{41} & x'^{42} & x'^{43} & x'^{44} & x'^{45} & x'^{46} & x'^{47} & x'^{48} \\ x'^{51} & x'^{52} & x'^{53} & x'^{54} & x'^{55} & x'^{56} & x'^{57} & x'^{58} \\ x'^{61} & x'^{62} & x'^{63} & x'^{64} & x'^{65} & x'^{66} & x'^{67} & x'^{68} \\ x'^{71} & x'^{72} & x'^{73} & x'^{74} & x'^{75} & x'^{76} & x'^{77} & x'^{78} \\ x'^{81} & x'^{82} & x'^{83} & x'^{84} & x'^{85} & x'^{86} & x'^{87} & x'^{88} \end{bmatrix}$$

- In zig-zag order
  - Left top: low-frequency (small # oscillations).
  - Right bottom: high-frequency.
- High-frequency components add fine details.
- Low-frequency components contribute to the global description.

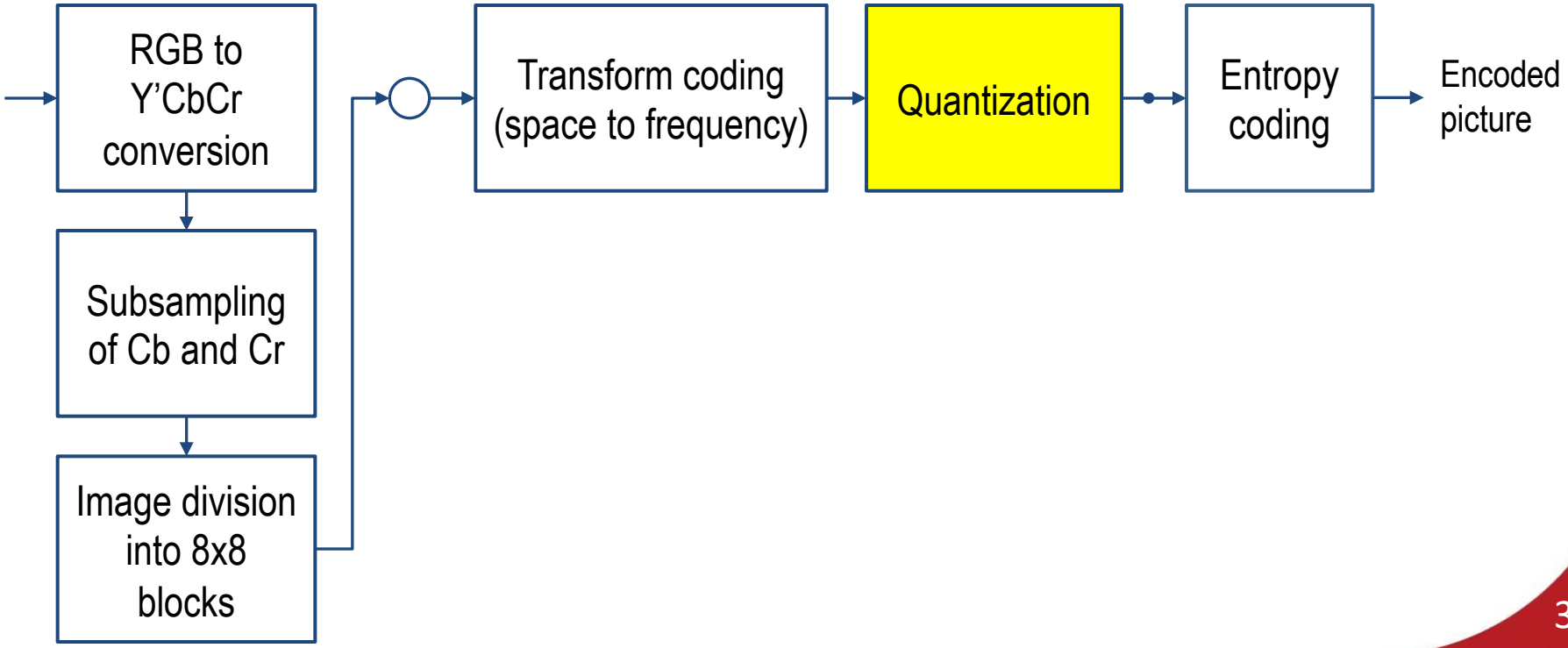
DCT Summary

$$\boxed{\begin{matrix}\text{DCT}\\(\mathbf{x})\end{matrix}} = \begin{matrix} x'^{11} & x'^{12} & x'^{13} & x'^{14} & x'^{15} & x'^{16} & x'^{17} & x'^{18} \\ x'^{21} & x'^{22} & x'^{23} & x'^{24} & x'^{25} & x'^{26} & x'^{27} & x'^{28} \\ x'^{31} & x'^{32} & x'^{33} & x'^{34} & x'^{35} & x'^{36} & x'^{37} & x'^{38} \\ x'^{41} & x'^{42} & x'^{43} & x'^{44} & x'^{45} & x'^{46} & x'^{47} & x'^{48} \\ x'^{51} & x'^{52} & x'^{53} & x'^{54} & x'^{55} & x'^{56} & x'^{57} & x'^{58} \\ x'^{61} & x'^{62} & x'^{63} & x'^{64} & x'^{65} & x'^{66} & x'^{67} & x'^{68} \\ x'^{71} & x'^{72} & x'^{73} & x'^{74} & x'^{75} & x'^{76} & x'^{77} & x'^{78} \\ x'^{81} & x'^{82} & x'^{83} & x'^{84} & x'^{85} & x'^{86} & x'^{87} & x'^{88} \end{matrix}$$

- DCT does not directly compress data (reduce the number of bits); It orders data in increasing frequency
- In the subsequent quantization step, only approximations of the DCT components are stored:
  - High-frequency contribute less to the perceived visual quality
    → High-frequency components are roughly approximated

JPEG – Main Blocks

- The diagram below shows the main blocks of the JPEG encoder for image coding



DCT and Quantization

- Because DCT is designed to work on pixel values ranging from -128 to 127, the original block is levelled off by subtracting 128 from each entry.

$$\begin{bmatrix} 154 & 123 & 123 & 123 & 123 & 123 & 123 & 136 \\ 192 & 180 & 136 & 154 & 154 & 154 & 136 & 110 \\ 254 & 198 & 154 & 154 & 180 & 154 & 123 & 123 \\ 239 & 180 & 136 & 180 & 180 & 166 & 123 & 123 \\ 180 & 154 & 136 & 167 & 166 & 149 & 136 & 136 \\ 128 & 136 & 123 & 136 & 154 & 180 & 198 & 154 \\ 123 & 105 & 110 & 149 & 136 & 136 & 180 & 166 \\ 110 & 136 & 123 & 123 & 123 & 136 & 154 & 136 \end{bmatrix}$$

Subtracting 128

$$\begin{bmatrix} 26 & -5 & -5 & -5 & -5 & -5 & -5 & 8 \\ 64 & 52 & 8 & 26 & 26 & 26 & 8 & -18 \\ 126 & 70 & 26 & 26 & 52 & 26 & -5 & -5 \\ 111 & 52 & 8 & 52 & 52 & 38 & -5 & -5 \\ 52 & 26 & 8 & 39 & 38 & 21 & 8 & 8 \\ 0 & 8 & -5 & 8 & 26 & 52 & 70 & 26 \\ -5 & -23 & -18 & 21 & 8 & 8 & 52 & 38 \\ -18 & 8 & -5 & -5 & -5 & 8 & 26 & 8 \end{bmatrix}$$

DCT and Quantization

- Because DCT is designed to work on pixel values ranging from -128 to 127, the original block is levelled off by subtracting 128 from each entry.
- Then 2D DCT is applied

$$\begin{bmatrix} 26 & -5 & -5 & -5 & -5 & -5 & -5 & 8 \\ 64 & 52 & 8 & 26 & 26 & 26 & 8 & -18 \\ 126 & 70 & 26 & 26 & 52 & 26 & -5 & -5 \\ 111 & 52 & 8 & 52 & 52 & 38 & -5 & -5 \\ 52 & 26 & 8 & 39 & 38 & 21 & 8 & 8 \\ 0 & 8 & -5 & 8 & 26 & 52 & 70 & 26 \\ -5 & -23 & -18 & 21 & 8 & 8 & 52 & 38 \\ -18 & 8 & -5 & -5 & -5 & 8 & 26 & 8 \end{bmatrix}$$

2D DCT

$$\begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix}$$

DCT and Quantization

- Because DCT is designed to work on pixel values ranging from -128 to 127, the original block is levelled off by subtracting 128 from each entry.
- Then 2D DCT is applied
- The DCT coefficients are now compressed by quantization

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Quantization Table

DCT and Quantization

- The DCT coefficients are now compressed by quantization

$$\begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$Q_{i,j}$: Quantization Table

$D_{i,j}$: DCT values

$$\begin{bmatrix} 162.3 & 40.6 & 20.0 & 72.3 & 30.3 & 12.5 & -19.7 & -11.5 \\ 30.5 & 108.4 & 10.5 & 32.3 & 27.7 & -15.5 & 18.4 & -2.0 \\ -94.1 & -60.1 & 12.3 & -43.4 & -31.3 & 6.1 & -3.3 & 7.1 \\ -38.6 & -83.4 & -5.4 & -22.2 & -13.5 & 15.5 & -1.3 & 3.5 \\ -31.3 & 17.9 & -5.5 & -12.4 & 14.3 & -6.0 & 11.5 & -6.0 \\ -0.9 & -11.8 & 12.8 & 0.2 & 28.1 & 12.6 & 8.4 & 2.9 \\ 4.6 & -2.4 & 12.2 & 6.6 & -18.7 & -12.8 & 7.7 & 12.0 \\ -10.0 & 11.2 & 7.8 & -16.3 & 21.5 & 0.0 & 5.9 & 10.7 \end{bmatrix}$$

Quantization

$$C_{i,j} = round\left(\frac{D_{i,j}}{Q_{i,j}}\right)$$

$$\begin{bmatrix} 10 & 4 & 2 & 5 & 1 & 0 & 0 & 0 \\ 3 & 9 & 1 & 2 & 1 & 0 & 0 & 0 \\ -7 & -5 & 1 & -2 & -1 & 0 & 0 & 0 \\ -3 & -5 & 0 & -1 & 0 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$C_{i,j}$: Quantized DCT values

Different Quantization Tables

- To yield quality levels of 10 to 90

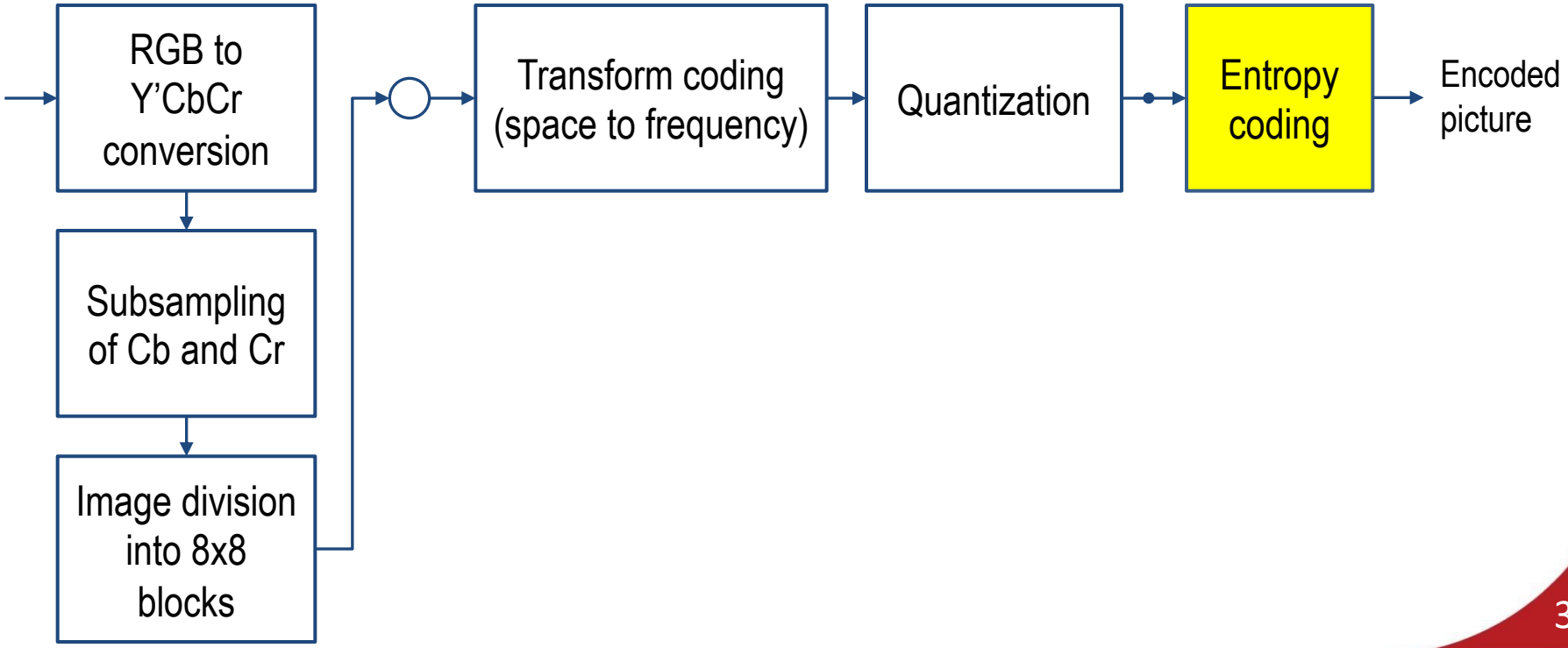
$$Q_{10} = \begin{bmatrix} 80 & 60 & 50 & 80 & 120 & 200 & 255 & 255 \\ 55 & 60 & 70 & 95 & 130 & 255 & 255 & 255 \\ 70 & 65 & 80 & 120 & 200 & 255 & 255 & 255 \\ 70 & 85 & 110 & 145 & 255 & 255 & 255 & 255 \\ 90 & 110 & 185 & 255 & 255 & 255 & 255 & 255 \\ 120 & 175 & 255 & 255 & 255 & 255 & 255 & 255 \\ 245 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \\ 255 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

$$Q_{90} = \begin{bmatrix} 3 & 2 & 2 & 3 & 5 & 8 & 10 & 12 \\ 2 & 2 & 3 & 4 & 5 & 12 & 12 & 11 \\ 3 & 3 & 3 & 5 & 8 & 11 & 14 & 11 \\ 3 & 3 & 4 & 6 & 10 & 17 & 16 & 12 \\ 4 & 4 & 7 & 11 & 14 & 22 & 21 & 15 \\ 5 & 7 & 11 & 13 & 16 & 12 & 23 & 18 \\ 10 & 13 & 16 & 17 & 21 & 24 & 24 & 21 \\ 14 & 18 & 19 & 20 & 22 & 20 & 20 & 20 \end{bmatrix}$$

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

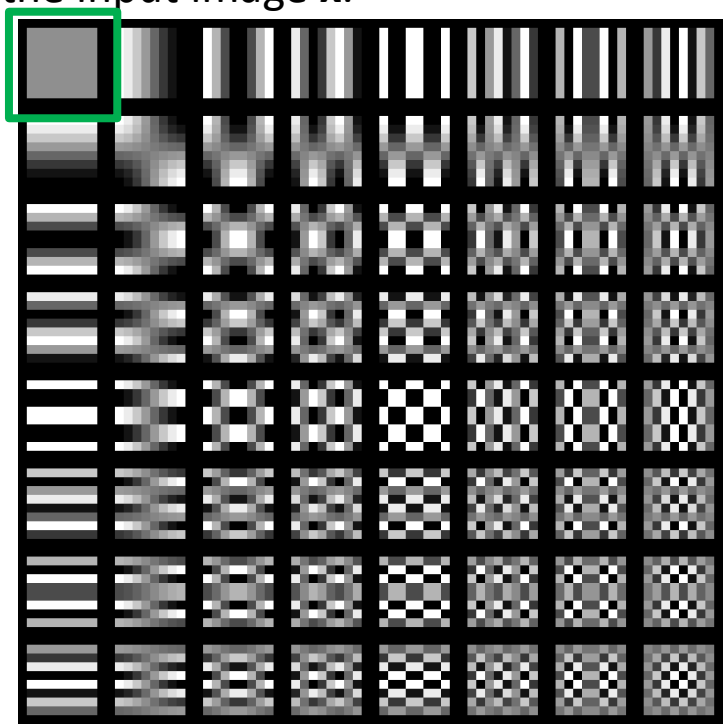
JPEG – Main Blocks

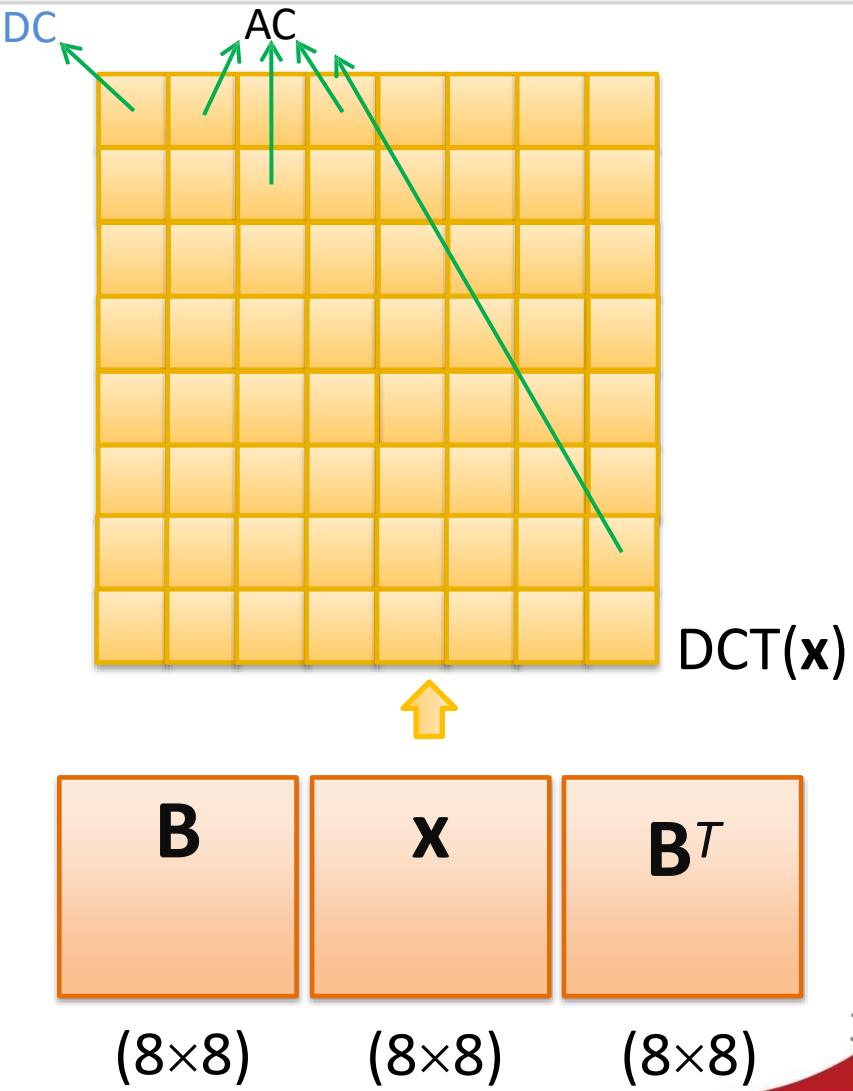
- The diagram below shows the main blocks of the JPEG encoder for image coding



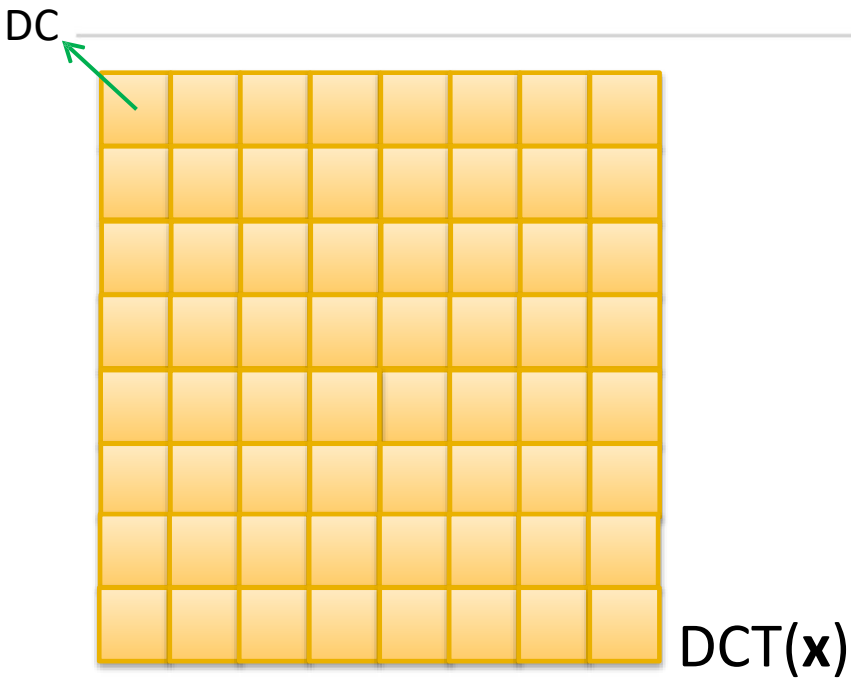
Symbol encoding: DC components

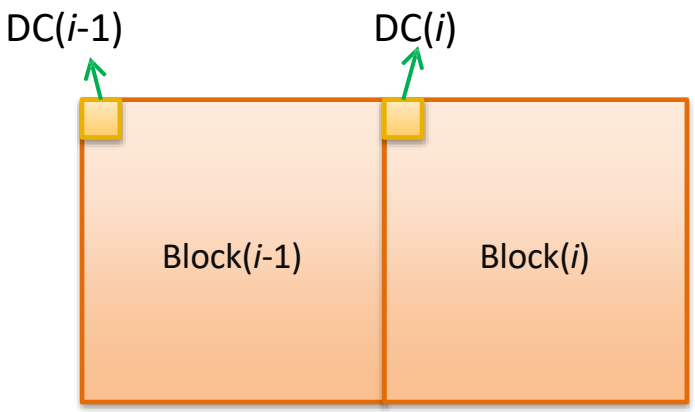
DC component (the first DCT coefficient) contains the average brightness value of the input image **x**.





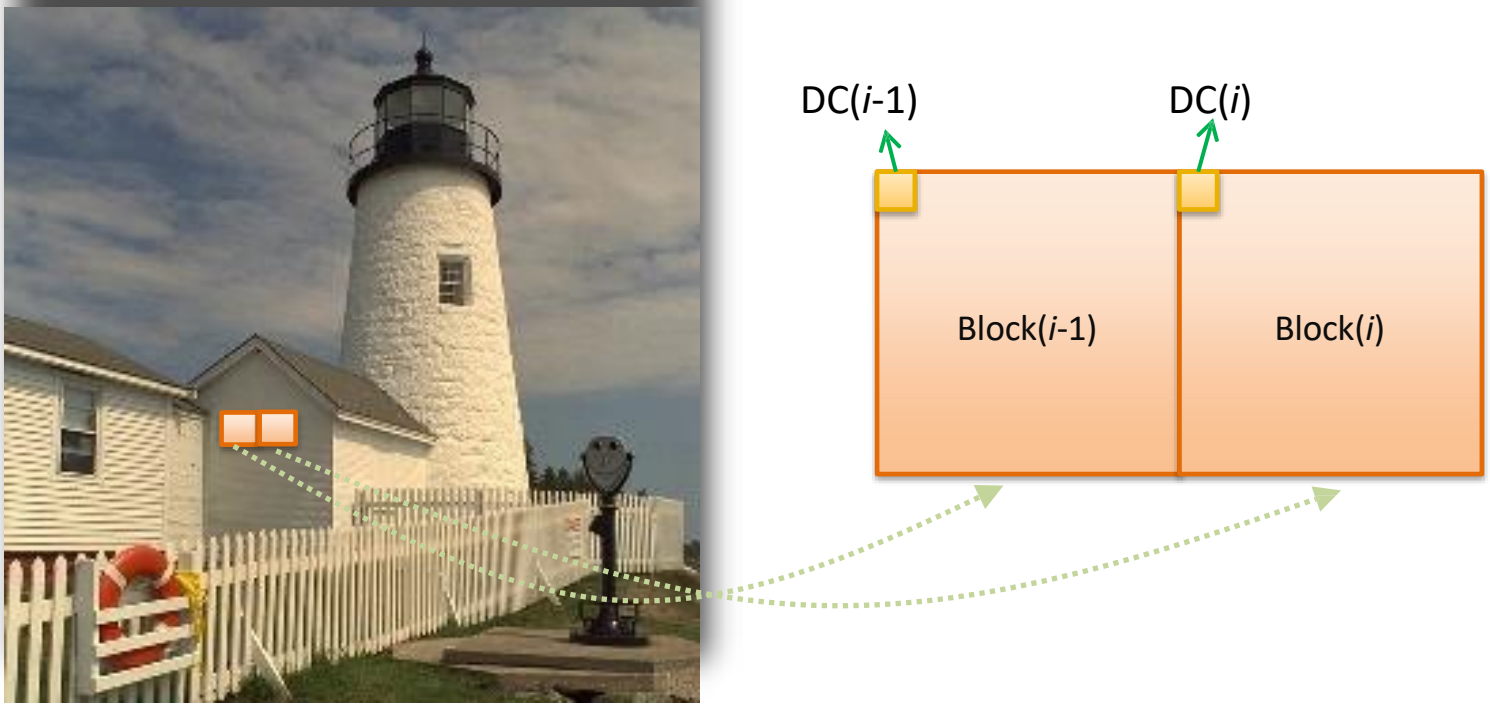
Symbol encoding: DC components





- The DC component contains the average color value of the block.
- Neighboring blocks are likely to have the same or similar average colors.
  $\rightarrow$ Differential pulse code modulation (DPCM): exploits correlation between the DC values.

Symbol encoding: DC components



- The DC component contains the average color value of the block.
- Neighboring blocks are likely to have the same or similar average colors.
  $\rightarrow$ Differential pulse code modulation (DPCM): exploits correlation between the DC values.

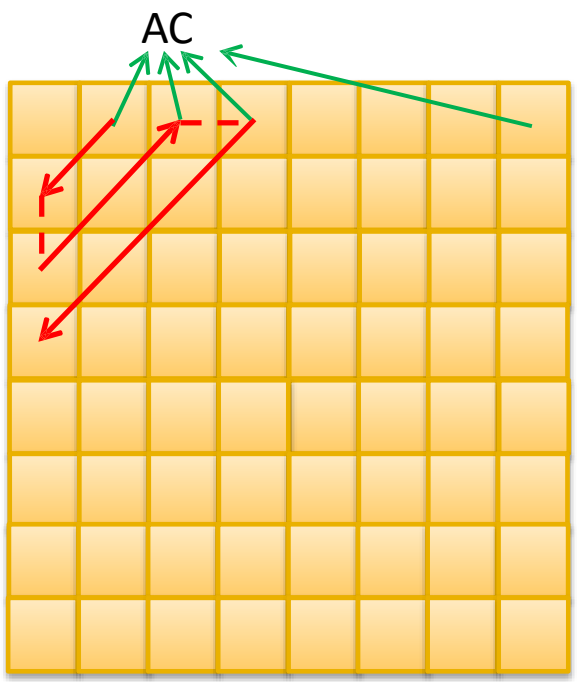
Differential pulse code modulation (DPCM)

T= 1, 1, 0.9, 0.9, 0.7, 0.9, 0.8, ….

T'= 1 0 -0.1 0 -0.2 0.2 -0.1

For a numerical sequence, DPCM stores the difference between each symbol (a number) and its predecessor

Symbol encoding: AC components



−96	−66	20	0	0	0	0	0
72	0	−14	−19	−26	0	0	0
−70	0	0	24	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized DCT coefficients

After quantization, most AC components are zero

Symbol encoding: AC components

-66 72 -70 0 20 0 -14 0 0 0 0 0
-19 0 0 -26 24 0 0 0 0 0 0 …… 0

RLE

-66(0) 72(0) -70(0) 20(1) -14(1)
-19(5) -26(2) 24(0) …

A(B): the number B of zeros observed before *A*

−96	−66	20	0	0	0	0	0
72	0	−14	−19	−26	0	0	0
−70	0	0	24	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized DCT coefficients

After quantization, most AC components are zero
→ Many consecutive zeros in zig-zag ordering
→ Run-length encoding (RLE) of 0 values