Topic 10: Feature Extraction

Dr. V Masilamani

masila@iiitdm.ac.in

Department of Computer Science and Engineering
IIITDM Kancheepuram
Chennai-127

Overview

1. What is a feature of an object or image?
2. Feature Extraction
3. Feature Selection
4. Corner Point Detection
5. Corner Point Detection
   - Harris corner point detection
6. Harris detector: Algorithm
7. Harris detector: Workflow

What is a feature?

- A feature is a property of an object. Eg: Size of an object, Shape of an object.
- Features of Image may be of any of the following types
  - Feature of an image may be given in terms of features of pixels
  - Some properties of image such as
    - Histogram of the image
    - Fourier Transform(FT) of image or some properties of FT
    - Edge map of the image
    - **Features of some important points in image**
  - Some properties of blocks of image
    - Local(block) average
    - Local(block) histogram
  - Some properties of objects in the image
    - Area, diameter, perimeter texture etc.

Feature Extraction

- Feature Extraction: Process of extracting features of image(object) such that
  - feature values of images of different types of scenes(objects) are very different, but very close for similar scenes(objects)
- Need of feature extraction
  - Using raw data(image) for Image matching or classification is not effective as
    - Images of the same scene taken at different time or different cameras may be different
    - Raw data may have more redundant or irrelevant data for classification
    - Raw data is generally large, and hence computation time will be high

Feature Selection

- Suppose an object or an image is represented by a feature vector $f = (f_1, f_2, ..., f_n)$, where each f_i is a feature
- The process of dropping some of the features in f such that matching or classification accuracy improves
- Some feature selection methods
  - if Similarity(f_i, f_j) is high then drop f_i or f_j
  - Some important similarity metrics
    - Normalized Cross Correlation(NCC) (more the NCC, more the similarity)
    - Mutual Information(MI)(more the MI, More the similarity)
    - Mean Squared Difference(MSD)(Lesser the MSD, more the similarity)

- Consider the problem: Given two images, check if they are of the same scene
- We can solve this problem,
  If we can find some points in one image that can be
  - Found in another image as well
  - Found precisely – well localized
  - Found reliably – well matched(the features of the corresponding points are matched)

  when both images are of the same scene
  - Such points are called as important points

Other uses of such important points

- Want to compute a fundamental matrix to recover geometry(To do 3D reconstruction).
- Robotics/Vision: To find how a bunch of points move from one frame to another.
- Object tracking
- Build a panorama...

Suppose you want to build a panorama

How do we build panorama?

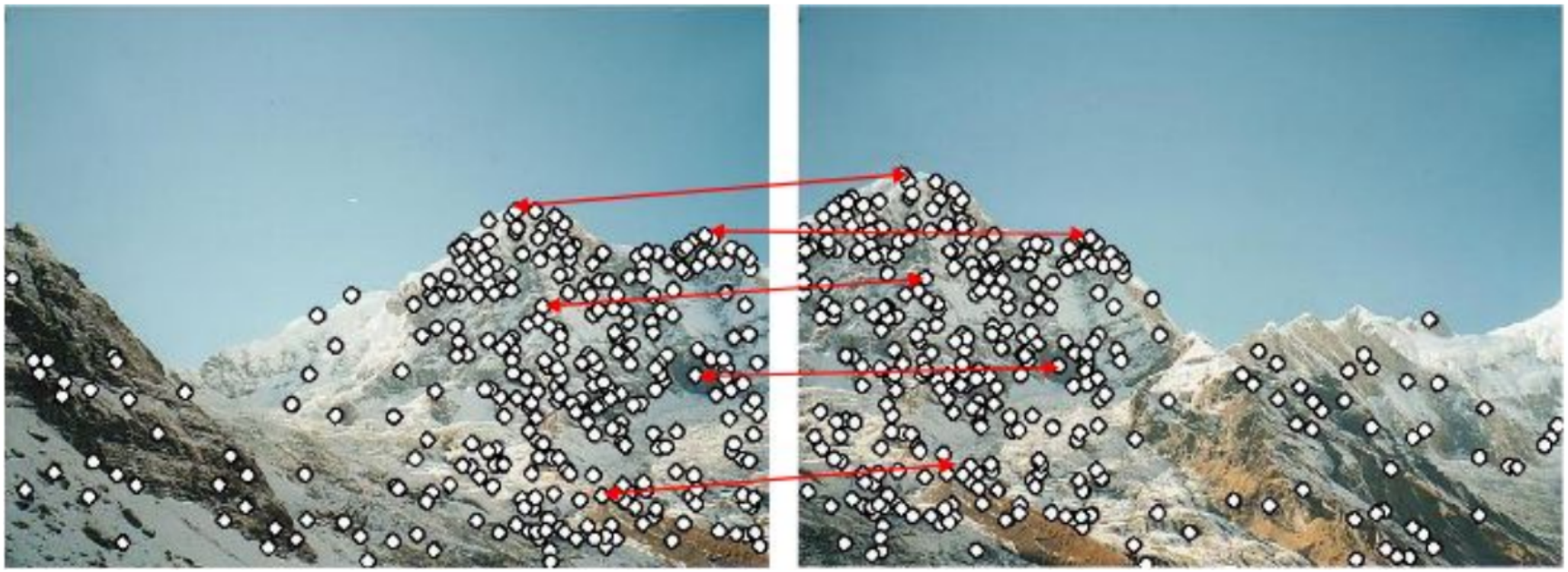
We need to match(align) the images.

Matching with Features

Detect features (feature points) in both images.

Matching with Features

- Detect features (feature points) in both images.
- Match features- find corresponding pairs.



Matching with Features

- Detect features (feature points) in both images.
- Match features- find corresponding pairs.
- Use these pairs to align images

Matching with Features

Problem 1:

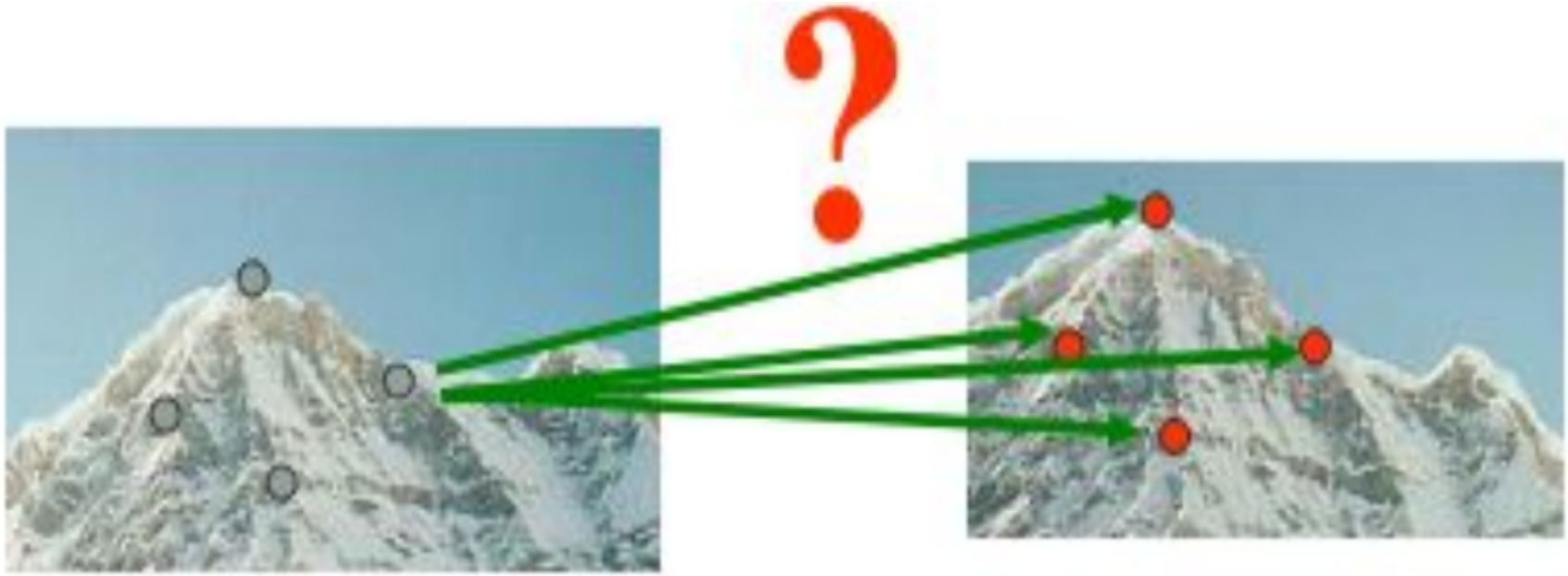
- Detect the same point independently in both the images.

no chance to match!

Matching with Features

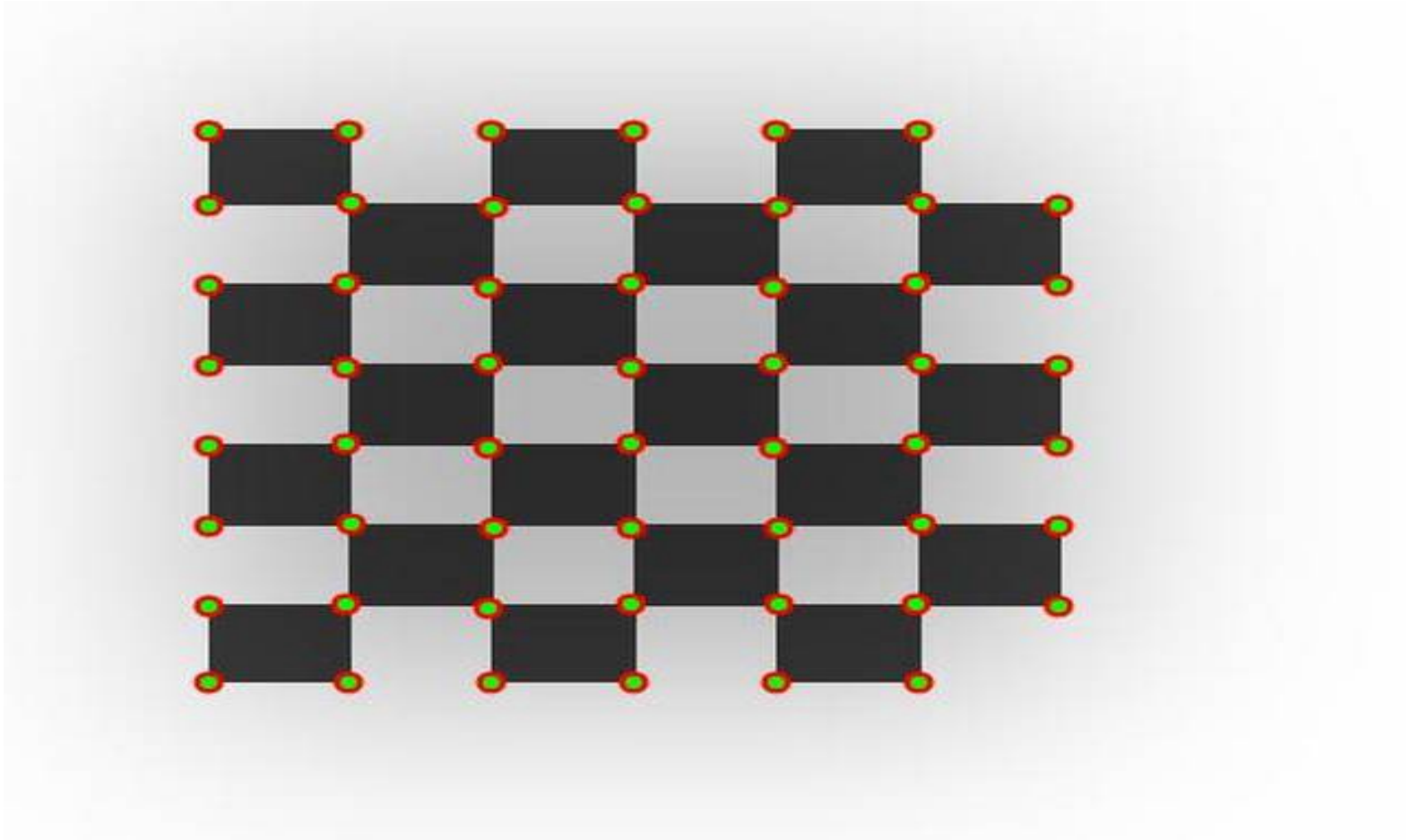
Problem 2:

- For each point correctly recognize the corresponding one

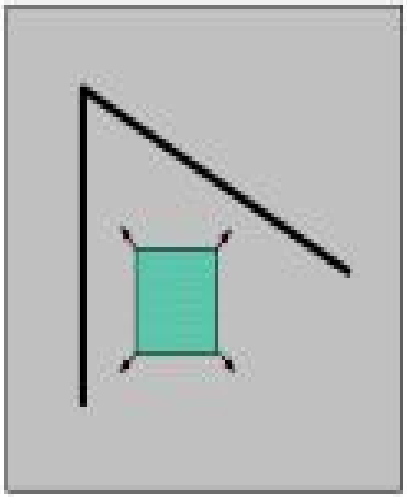


Corner points -important points

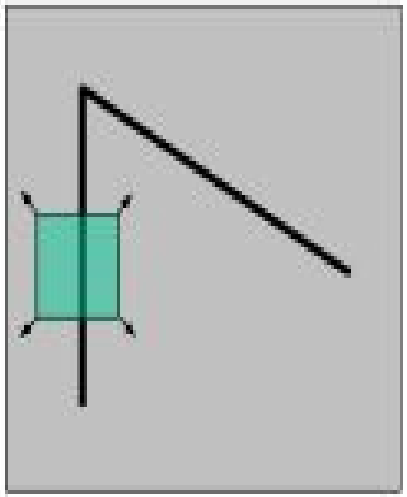
What is a corner point



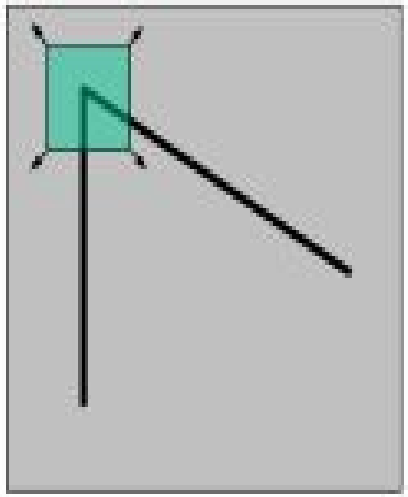
Corner points -important points (cont.)



"flat" region:
no change in all directions

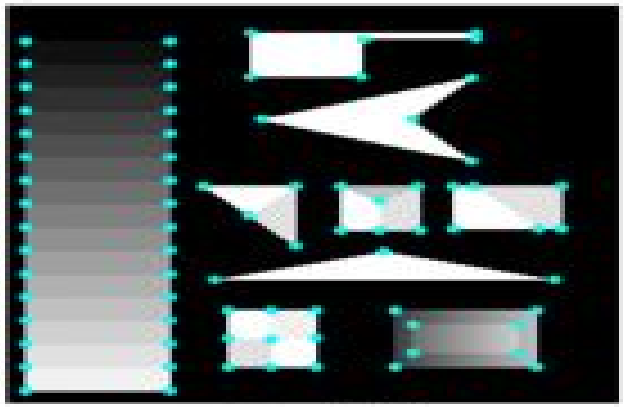


"edge":
no change along the edge direction

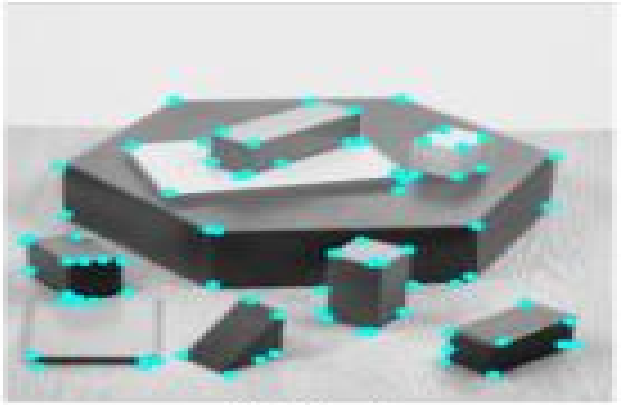


"corner":
significant change in all directions

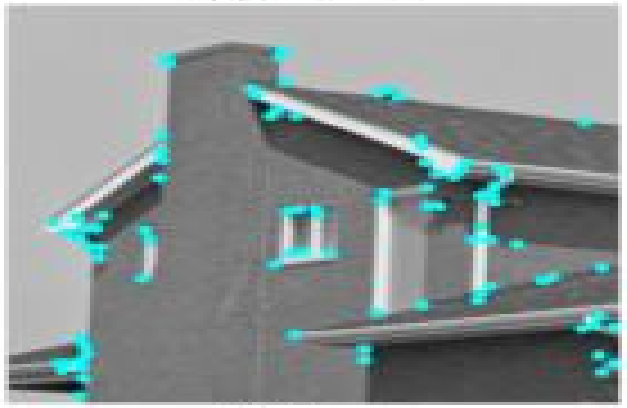
Corner points -important points (cont.)



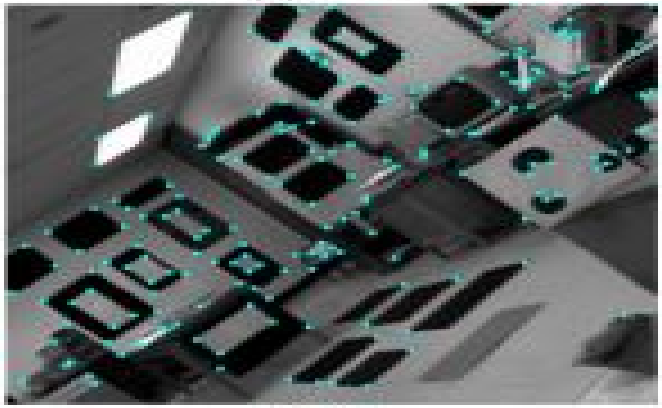
(a) Artificial



(b) Block



(c) House



(d) Lab

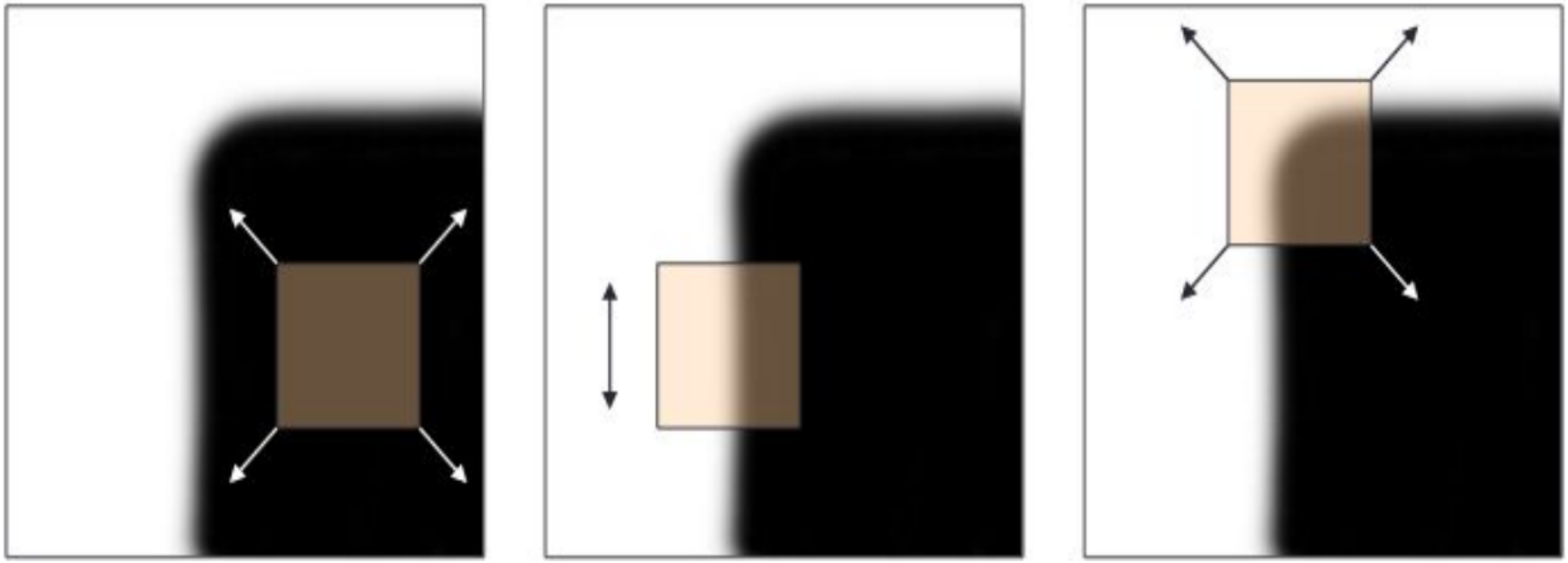
Corner points -important points (cont.)

- Key property: in the region around a corner, image gradient has two or more dominant directions.

How to model corner point

- Consider the window centered at (p, q) with size $a \times b$ in the image, say W_1
- Move the window with displacement vector (u, v) the resultant window as W_2
- If $||W_2 - W_1||^2$ is large for two or more displacement vectors (u, v)s than (p, q) is called as a corner point
- Note that (u, v) provide direction to move the window

Corner points -important points (cont.)



Harris Corner point Detection

- How to find whether or not the difference is large in two or more directions efficiently?
- To do that efficiently,the difference between two windows separated by the displacement vector (u, v) is defined as:

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

Window function - $w(x, y)$

Shifted Intensity - $I(x + u, y + v)$

Intensity - $I(x, y)$

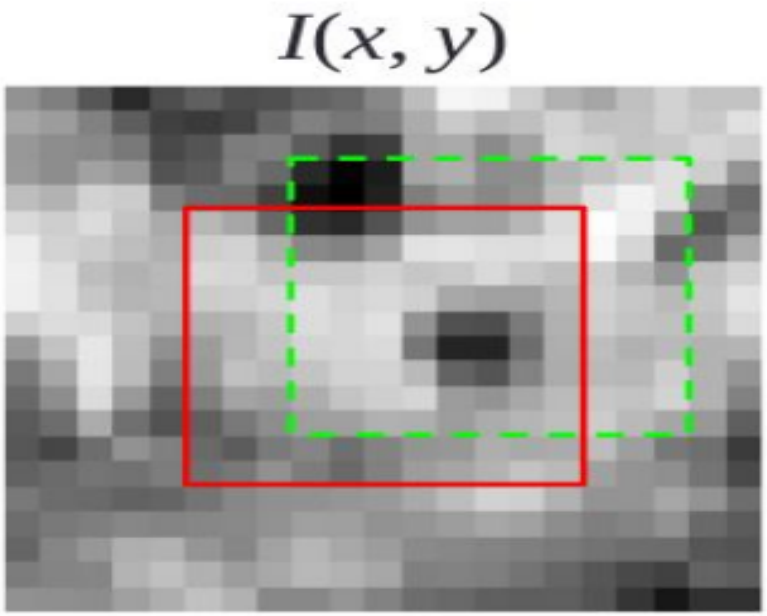
Objective:

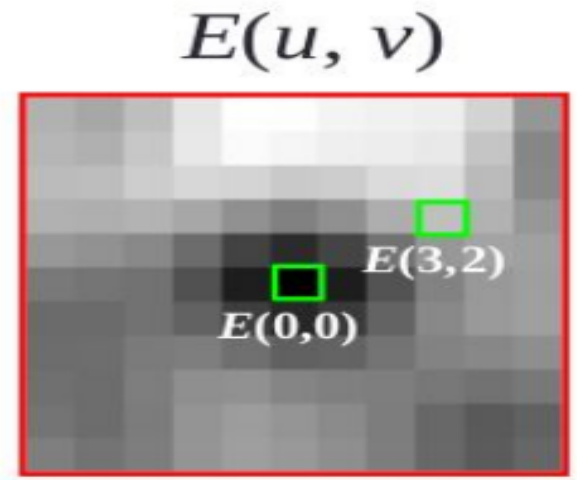
- Find each window that produce a large E value for two or more (u, v) (ie for two or more directions).

Harris Corner point Detection (cont.)

Change in appearance for the shift [u,v]

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





Taylor Series Expansion Approximation up to first derivative for 2D:

- $f(x+u, y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y)$
- Applying Taylor series approximation for $I(x+u, y+v)$ in $E(u,v)$ we get,

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

- $E(u,v) = \sum_{x,y} w(x,y) \, [\, uI_x(x,y) + vI_y(x,y) \,]^2$
- $E(u,v) =$
  $\sum_{x,y} w(x,y) \, [\, u^2 I_x^2(x,y) + v^2 I_y^2(x,y) + 2uvI_x(x,y)I_y(x,y) \,]$

Harris Corner point Detection (cont.)

we can rewrite as matrix equation:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2(x,y) & I_x(x,y)I_y(x,y) \\ I_x(x,y)I_y(x,y) & I_y^2(x,y) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

Written in a simple form as:

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

Where

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

Harris Corner point Detection (cont.)

- Decompose M as $M = Q^T A Q$, where
  - A is the diagonal matrix with eigen values of M in the diagonal: $A = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$
  - Each column of Q is an eigen vector of M
- Sub M in E(u,v), we get

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} Q^T A Q \begin{bmatrix} u \\ v \end{bmatrix}$$

-

$$E(u,v) \approx (Q \begin{bmatrix} u & v \end{bmatrix}^T)^T A Q \begin{bmatrix} u \\ v \end{bmatrix}$$

Harris Corner point Detection (cont.)

Interpretation of $Q(u,v)^T$

- $(u,v)^T$ is transformed into a new coordinate system with eigen vectors as axes, say $(u',v')^T$
- Hence

$$E(u,v) \approx (\begin{bmatrix} u' & v' \end{bmatrix} A \begin{bmatrix} u' \\ v' \end{bmatrix}$$

-

$$E(u,v) \approx (\begin{bmatrix} u' & v' \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix}$$

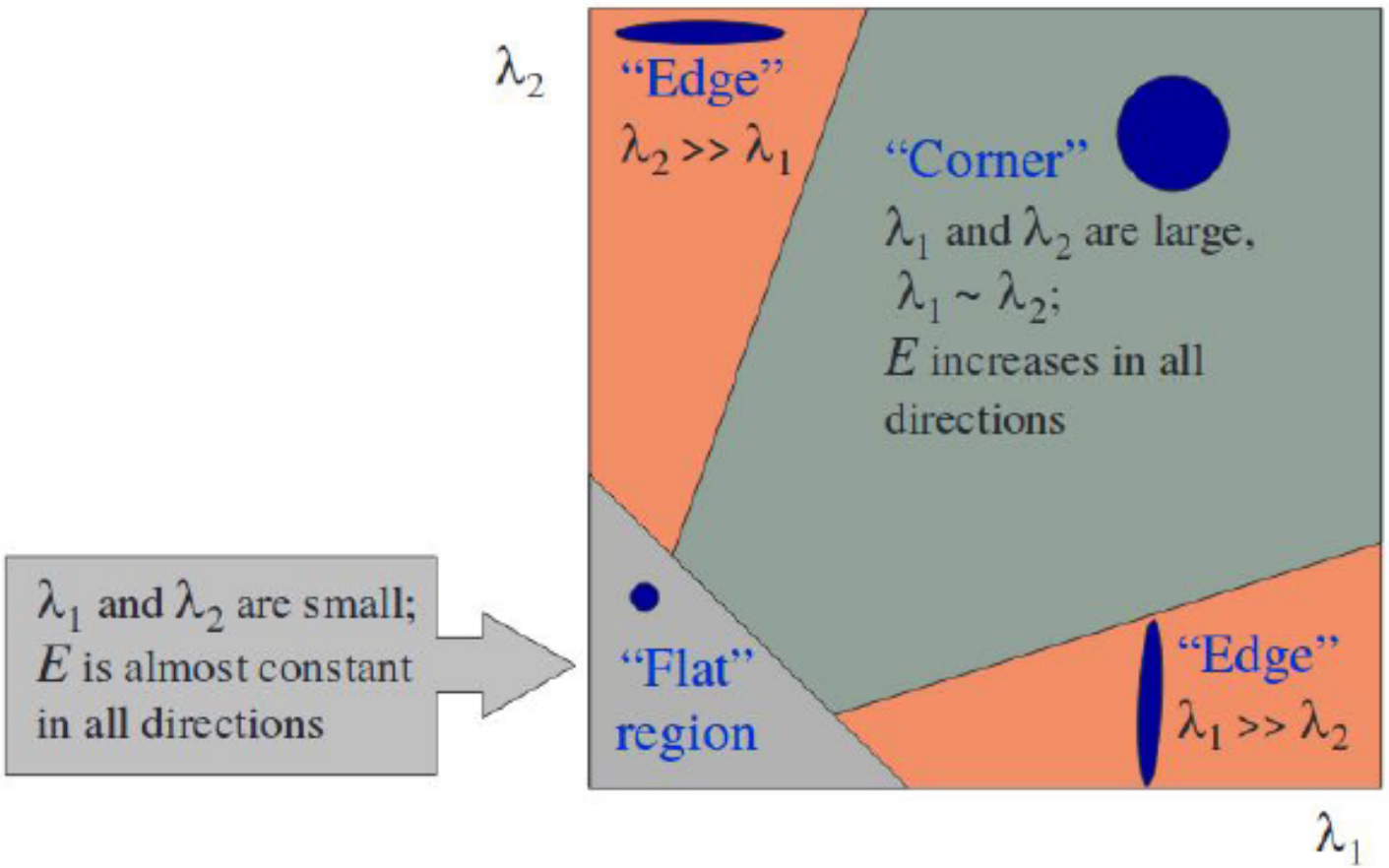
- $E(u,v) = (u'^2\lambda_1 + v'^2\lambda_2)$

Harris Corner point Detection (cont.)

- **Observation:** $E(u, v)$ is large when λ_1 and λ_2 are large
- As (u, v) is a directional vector, (u', v') is also a directional vector in eigen space
- If we consider $(u', 0)$ and $(0, v')$ as two directions, E is large when λ_1(length in $(u', 0)$) and λ_2(length in $(0, v')$)

Interpreting the eigenvalues

Classification of image points using eigenvalues of M:

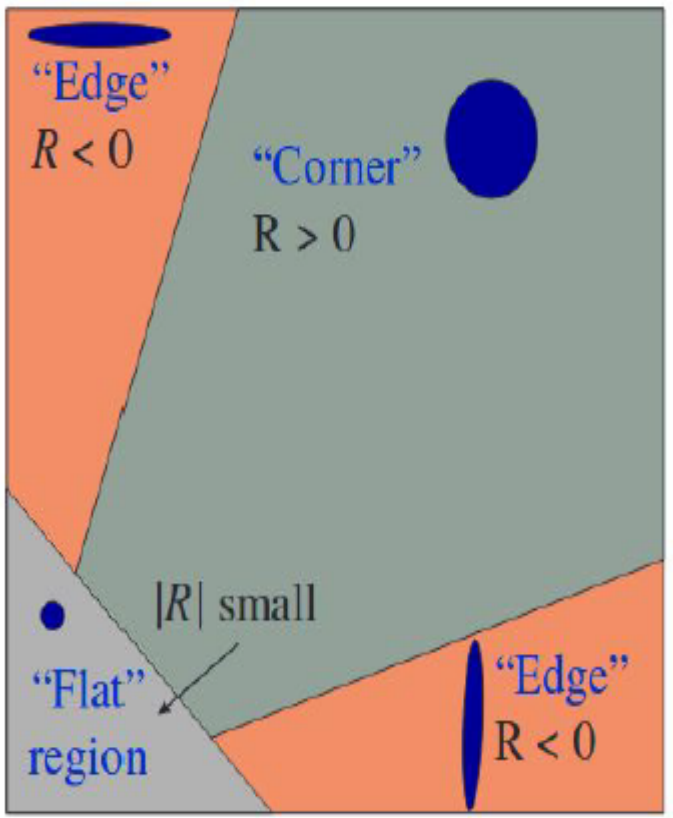


Harris corner response function

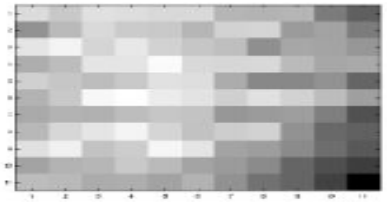
For computational efficiency, use the following

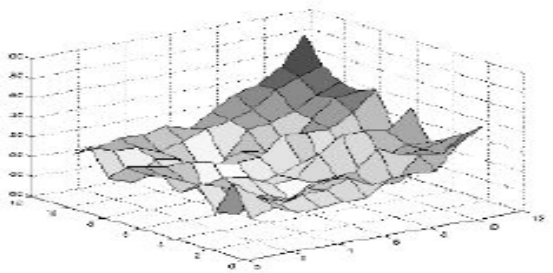
$$R = det(M) - \alpha trace(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

- R is large for a corner
- R is negative with large magnitude for an edge
- $| R |$ is small for a flat region



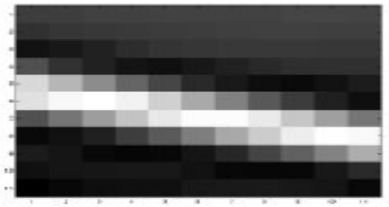
Low texture region

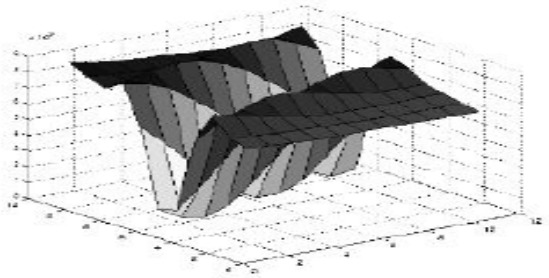




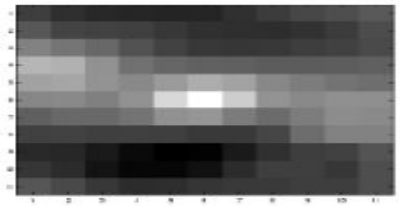
- gradients have small magnitude
- small λ_1, small λ_2

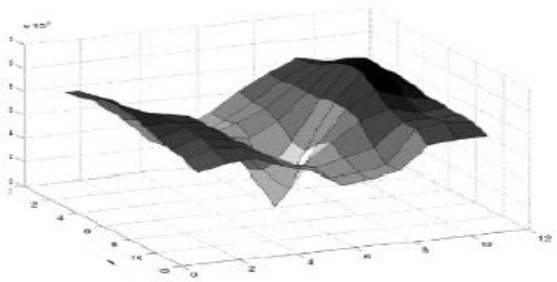
Edge





Edge





$$\sum \nabla I (\nabla I)^T$$

Harris corner point detection algorithm

Input: I
Output: Set of all corner points in I
Steps:

- Find derivative image I in x direction, and call it as I_x
- Find derivative image I in y direction, and call it as I_y
- Compute the products of derivatives at each pixel:
  - $I_{x2} = I_x I_x$;
  - $I_{y2} = I_y I_y$;
  - $I_{xy} = I_x I_y$;
- Compute sums of products of derivatives at each pixel
  - $S_{x2} = Conv(I_{x2}, h)$;
  - $S_{y2} = Conv(I_{y2}, h)$;
  - $S_{xy} = Conv(I_{xy}, h)$

Harris corner point detection algorithm (cont.)

where h is the sum filter

- Compute at for each pixel (x, y)

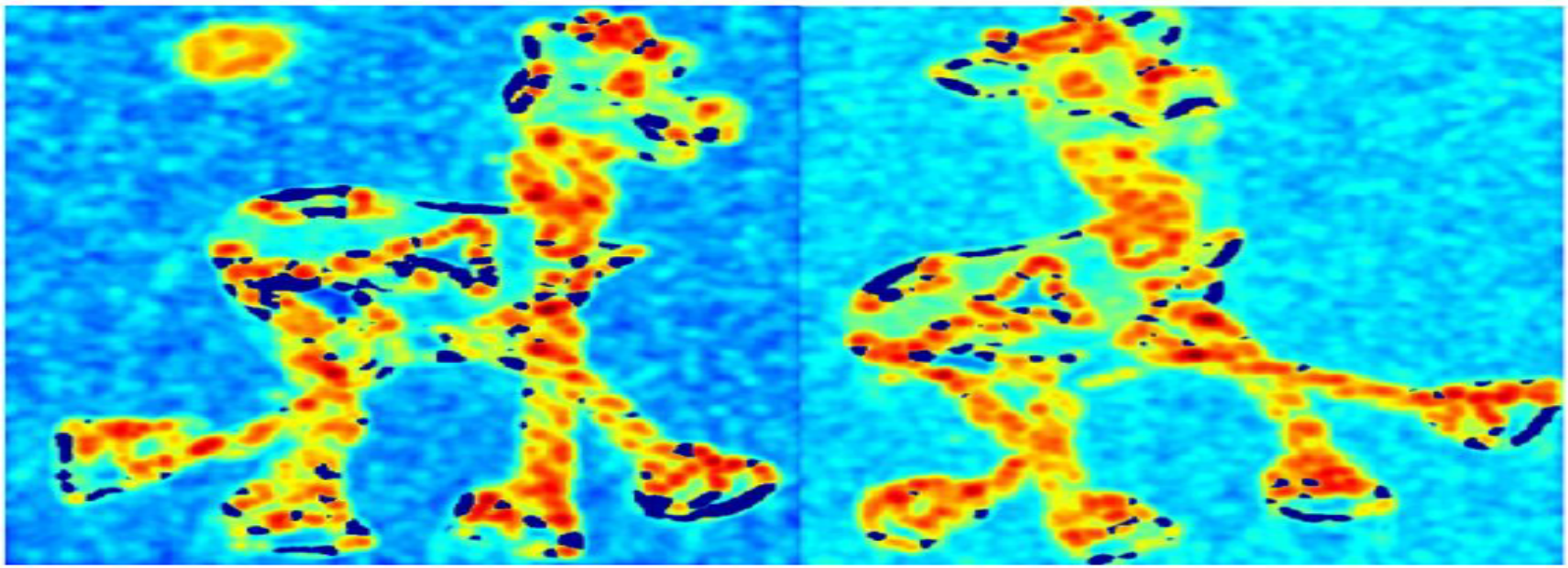
$$H(x,y) = \begin{bmatrix} S_{x2}(x,y) & S_{xy(x,y)} \\ S_{xy}(x,y) & S_{y2(x,y)} \end{bmatrix}$$

- Compute corner response function R at each pixel:
  $R(x,y) = Det(H(x,y)) - k(Trace(H(x,y)))^2$
- If $R(x,y) > T$ then output (x,y) as key point

Harris Detector: Workflow

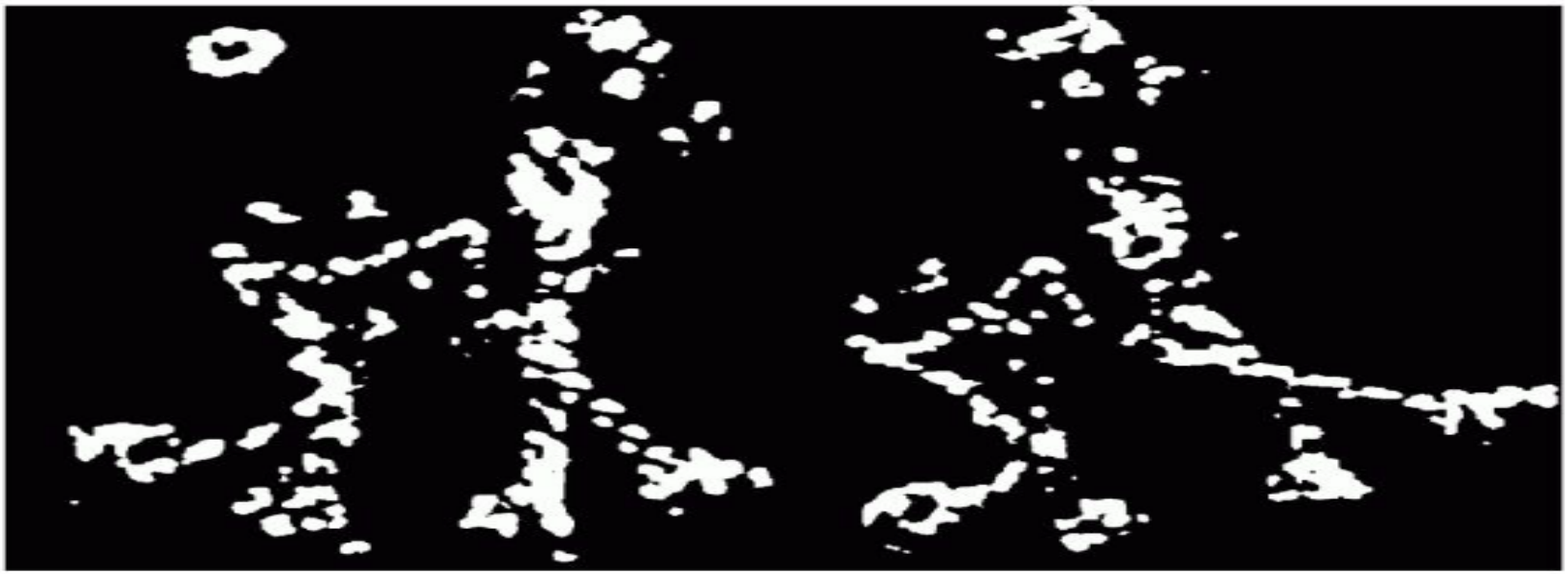
Harris Detector: Workflow (cont.)

Compute corner response R



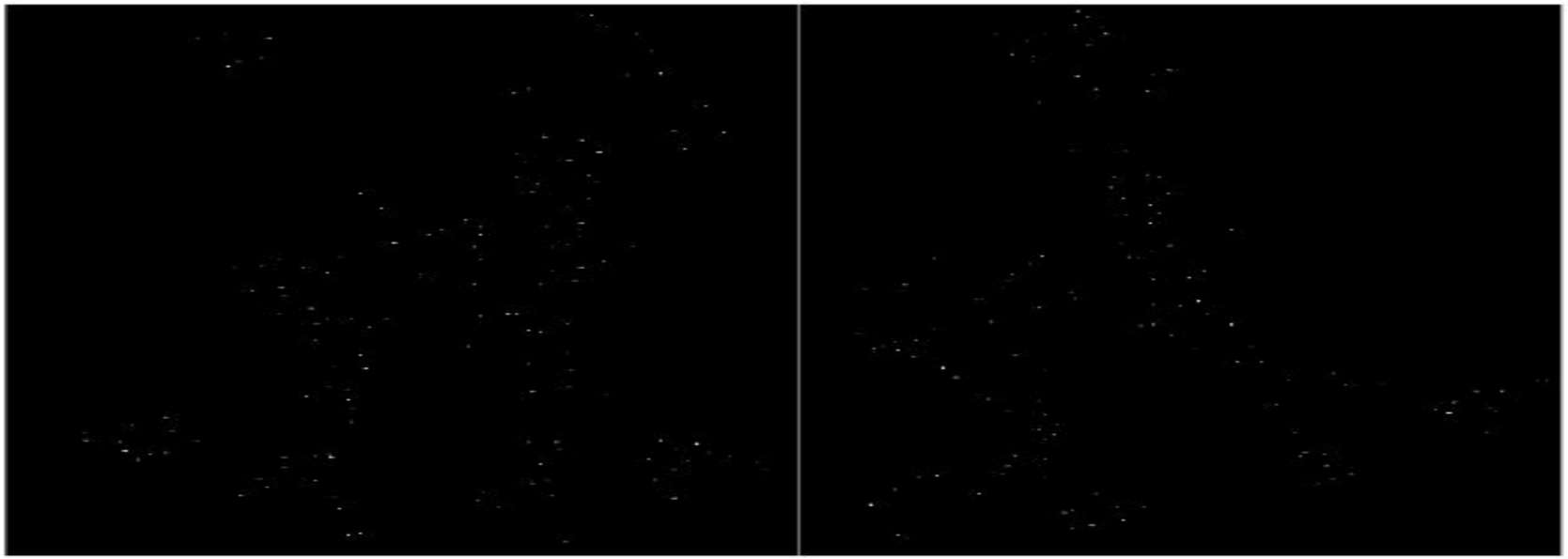
Harris Detector: Workflow (cont.)

Find the points with large corner response: $R > threshold$



Harris Detector: Workflow (cont.)

Take only the points of local maxima of R



Harris Detector: Workflow (cont.)