# Applied Computer Vision

David Vernon
Carnegie Mellon University Africa

vernon@cmu.edu
www.vernon.eu

# Lecture 11

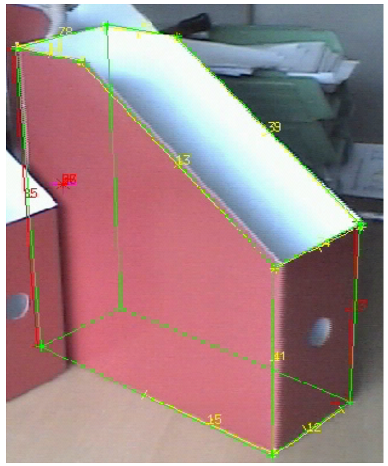
# Image Features

Harris and interest point operator

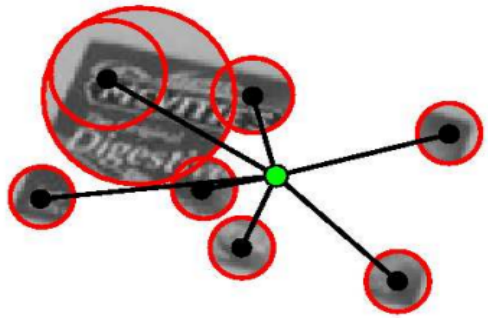
# Approaches to Object Recognition

- Generic Gestalt Principles
  - The world is structured, extract features
  - perceptual grouping

- Model based
  - CAD model of object
  - Geometric features
  - Locate features and their arrangement



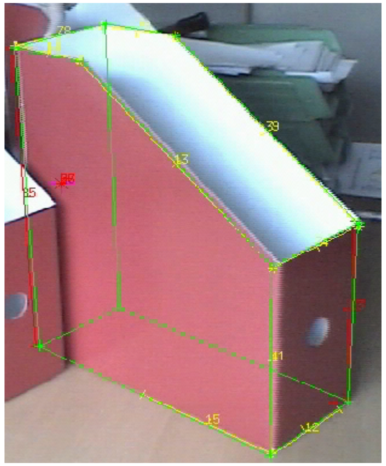
- Appearance based
  - Interest points / point features
  - or “whole” object

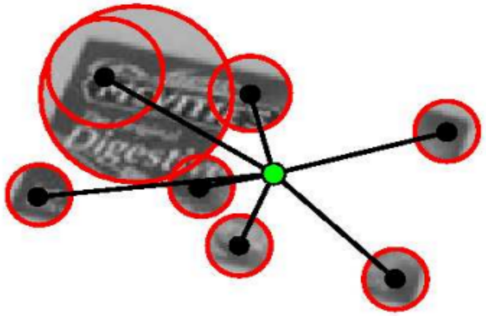


Credit: Markus Vincze, Technische Universität Wien

# Approaches to Object Recognition

- Point features
  - Key point features
  - Interest points
  - Corners
- Find features in one image and track
- Find features in all images and match





Credit: Markus Vincze, Technische Universität Wien

# Objects and Interest Points (IPs)

1. Feature detection
   Extract interest points
   (unique image regions)

2. Feature description
   Calculate local
   (invariant) descriptors

3. Feature matching / feature tracking
   Find correspondences

4. Find similar image regions/objects

Credit: Markus Vincze, Technische Universität Wien

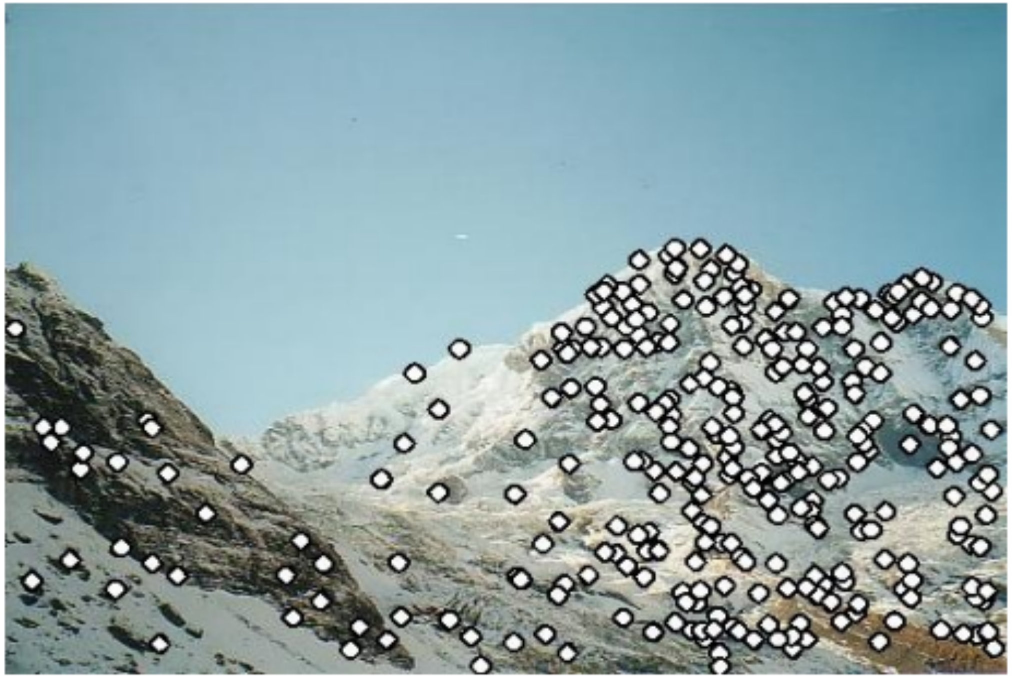
# Example: Object Recognition for Image Stitching

- How to recognize the same objects or parts of objects in different images?
- Example: Panorama – putting many images together

Credit: Markus Vincze, Technische Universität Wien

# Image Stitching: Interest Points (IPs)

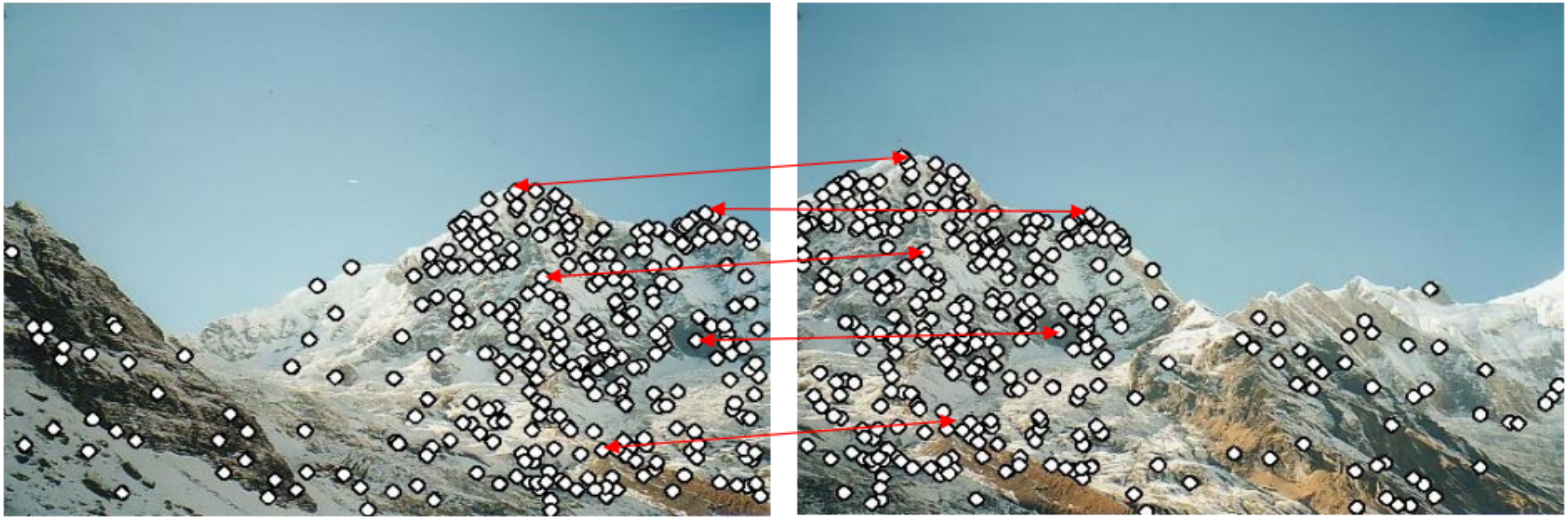
- Recognition of features in both images
- Finding of corresponding point pairs



Credit: Markus Vincze, Technische Universität Wien

# Image Stitching: Matching

- Recognition of features in both images
- Finding of corresponding point pairs
- Use points to put image together → panorama



Credit: Markus Vincze, Technische Universität Wien

# Image Stitching: Matching

- Recognition of features in both images
- Finding of corresponding point pairs
- Use points to put image together → panorama

Credit: Markus Vincze, Technische Universität Wien

# Matching with Features

Problem 1:

Detection of the same point pair independently in two images

- Here: not the same points, no match
- Need: reliable and distinctive point descriptor

Credit: Markus Vincze, Technische Universität Wien

# Matching with Features

Problem 2:
For every point in the image, find the corresponding point in the other image

- Here: which point on the right is correct?
- Need: reliable and distinctive point descriptor


Credit: Markus Vincze, Technische Universität Wien

# Interest Point Detection

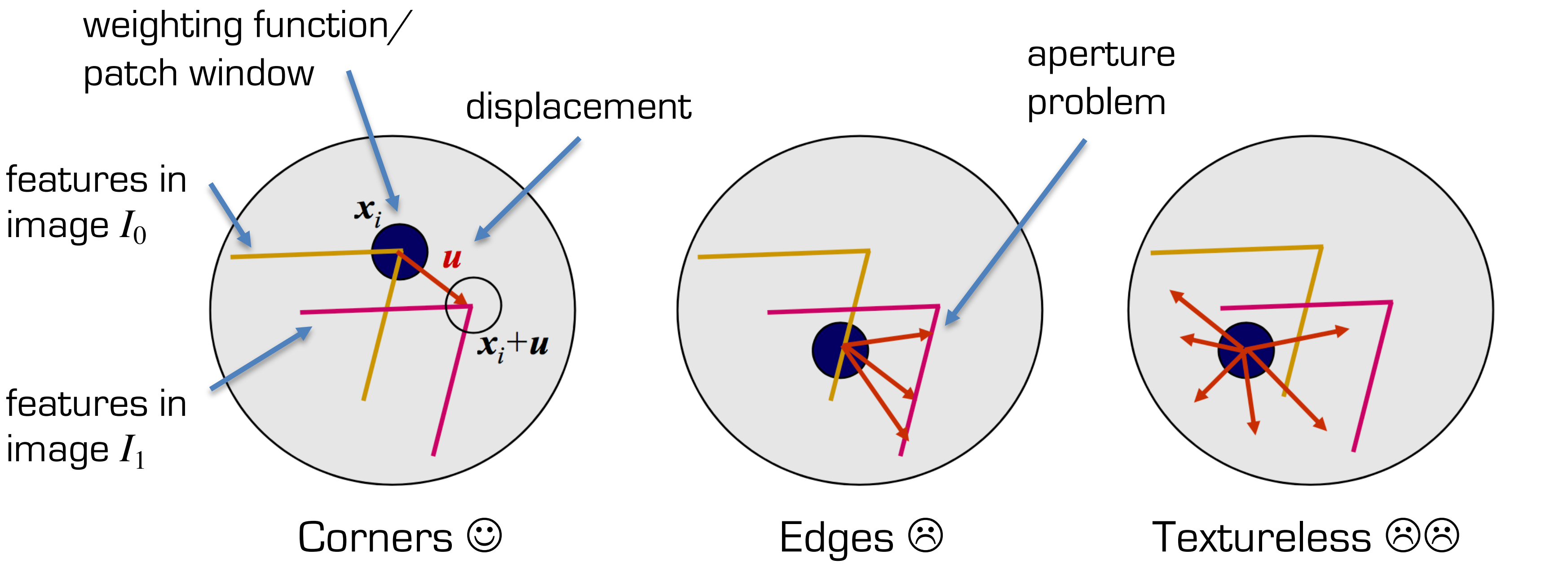
Many different approaches

- Corner detector: Harris (1988), Hessian
- Multi-scale corner detector with scale selection
  - Scale invariant Harris and Hessian corners
  - Difference of Gaussian (DoG) (Lowe 2004)
- Affine covariant Regions
  - Harris-Affine (Mikolajczyk, Schmid ‘02, Schaffalitzky, Zisserman ’02)
  - Hessian-Affine (Mikolajczyk and Schmid ’02)
  - Maximally stable extremal regions (MSER) (Matas et al. ’02)
  - Intensity based regions (IBR) (Tuytelaars and Van Gool ’00)
  - Edge based regions (EBR) (Tuytelaars and Van Gool ’00)
  - Entropy-based regions (salient regions) (Kadir et al. ’04)
  - Features from accelerated segment test (FAST) (Rosten et al. '05)

Credit: Markus Vincze, Technische Universität Wien

# Interest Point Detection

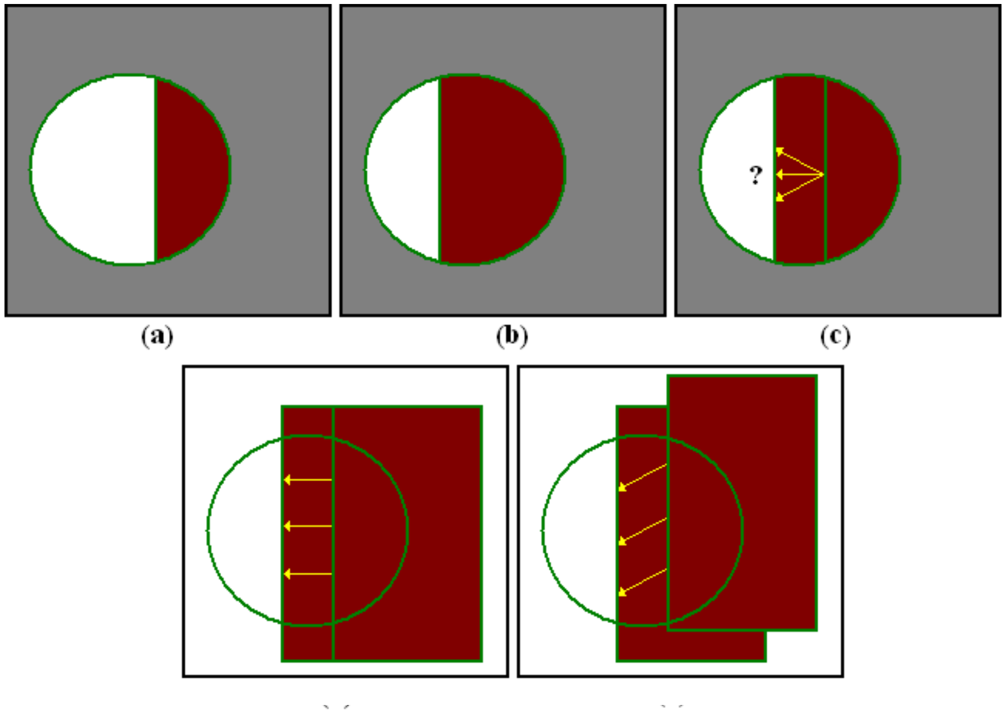
- Textureless patches are almost impossible to localize
- Patches with high contrast (gradient) are easier to localize
- Straight-line segments suffer from the aperture problem




Credit: Szelisky 2010

# Interest Point Detection

- Aperture problem with edges
- Given two images (a) and (b) taken at different times, determine the movement of edge points from frame 1 to frame 2 (c) …

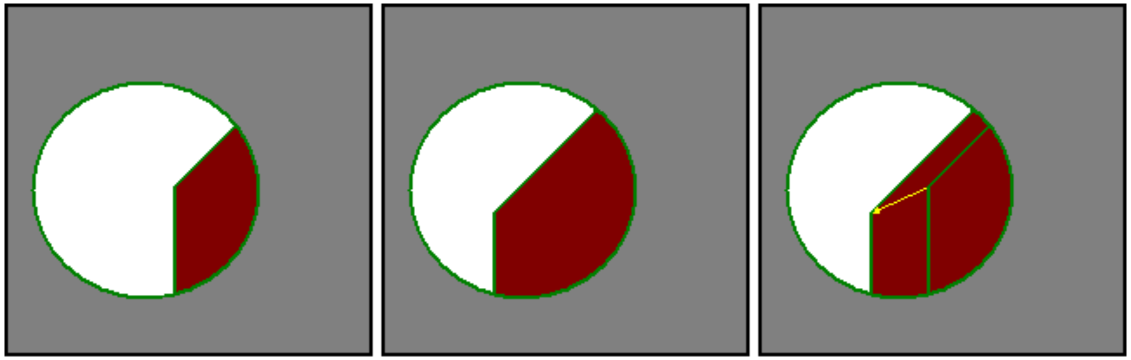



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

# Interest Point Detection

Use corners / point features / interest points

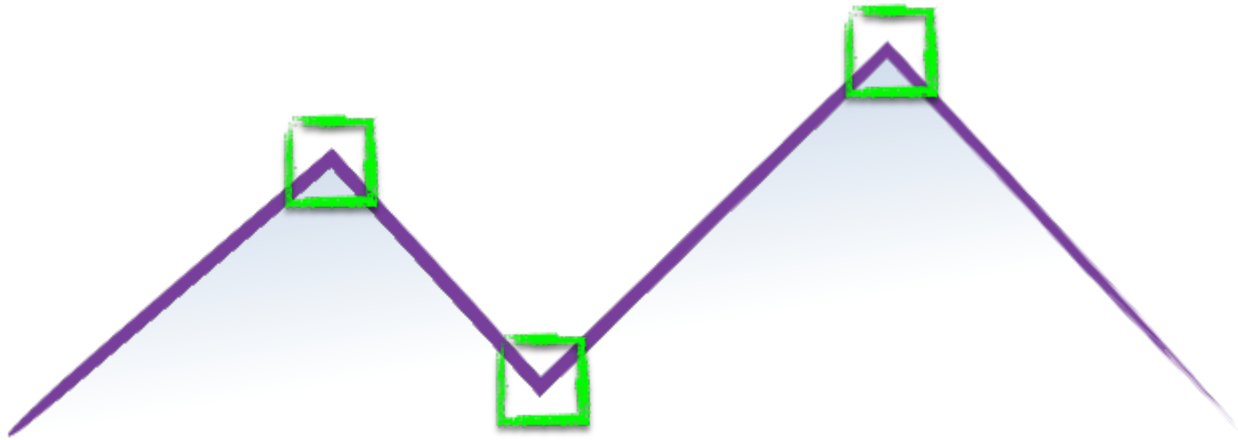
- corner … intersection of two edges
- interest point … any feature that can be robustly detected



Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

# How do you find a corner?

[Moravec 1980]



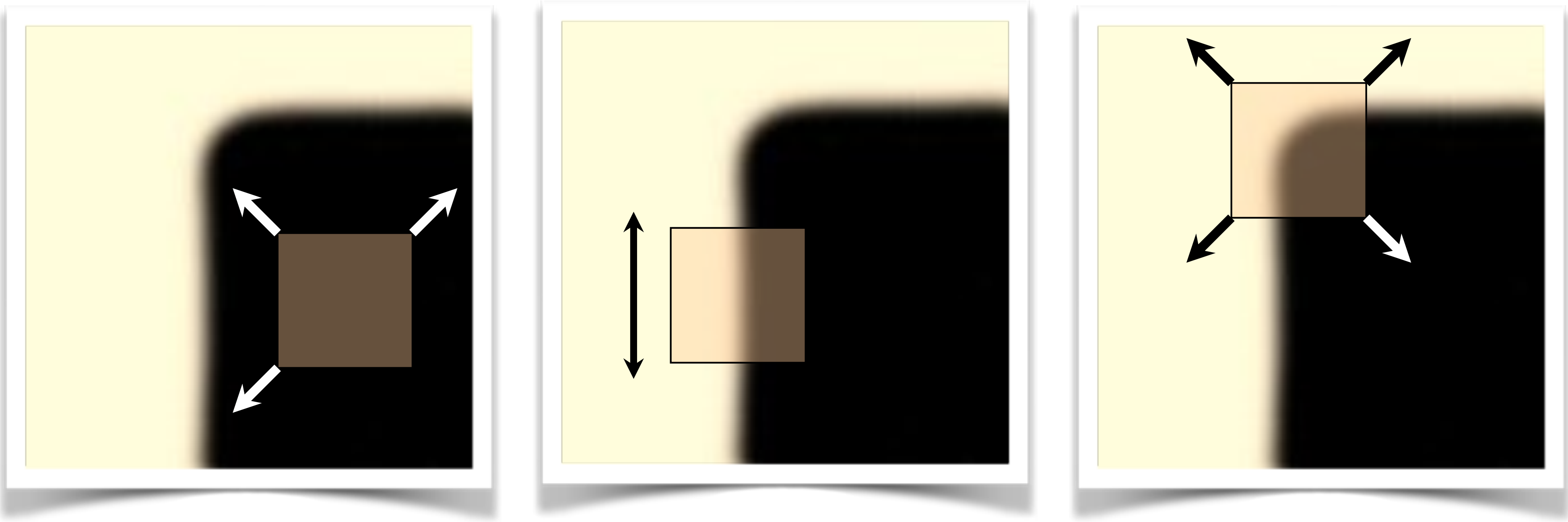
Easily recognized by looking through a small window

Shifting (displacing) the window should give large change in intensity

Credit: Kris Kitani, Carnegie Mellon University

Easily recognized by looking through a small window

Shifting (displacing) the window should give large change in intensity



"flat" region:
no change in all directions

"edge":
no change along the edge direction

"corner":
significant change in all directions

Credit: Kris Kitani, Carnegie Mellon University

[Moravec 1980]

# Interest Point Detection

## Autocorrelation function

How well an image patch matches itself as a function of a small displacement
(sum of squared differences in a region window)

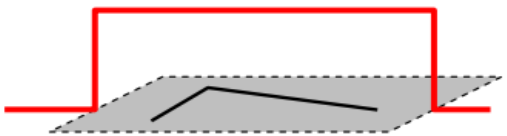
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u, y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

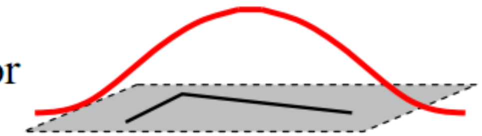
Intensity

Window function $w(x,y) =$



1 in window, 0 outside

or



Gaussian

Credit: Markus Vincze, Technische Universität Wien

# Interest Point Detection

For small shifts $[u, v]$ we have a bilinear approximation

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

where M is the $2{\times}2$ autocorrelation matrix computed from image derivatives $I_x$ and $I_x$

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

Also known as the covariance matrix

To see why, let's consider the following

Credit: Markus Vincze, Technische Universität Wien

# Interest Point Detection

Consider the intensity variation for a displacement $(\Delta i, \Delta j)$ as sum of squared differences $SSD$

assuming a box weighting / windowing function $w$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j) \in W} \left(f(i,j) - f(i - \Delta i, j - \Delta j)\right)^2$$

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

# Interest Point Detection

Consider the intensity variation for a displacement $(\Delta i, \Delta j)$ as sum of squared differences $SSD$

Approximating the displaced image as follows

$$f(i - \Delta i, j - \Delta j) \approx f(i,j) + \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix}$$

displacement in $i$ and $j$ directions

rate of change of image in $i$ and $j$ directions

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

# Interest Point Detection

Consider the intensity variation for a displacement $(\Delta i, \Delta j)$ as sum of squared differences $SSD$

Substituting terms and simplifying

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left( f(i,j) - f(i,j) - \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)^2$$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left( \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)^2$$

$$SSD_W(\Delta i, \Delta j) = \sum_{(i,j)\in W} \left( \begin{bmatrix} \Delta i & \Delta j \end{bmatrix} \left( \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} \\ \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \begin{bmatrix} \frac{\delta f(i,j)}{\delta i} & \frac{\delta f(i,j)}{\delta j} \end{bmatrix} \right) \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix} \right)$$

# Interest Point Detection

Consider the intensity variation for a displacement $(\Delta i, \Delta j)$ as sum of squared differences $SSD$

Rearranging

$$SSD_W(\Delta i, \Delta j) = [\Delta i \quad \Delta j] \begin{bmatrix} \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta i}\right)^2 & \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i}\frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i}\frac{\delta f(i,j)}{\delta j} & \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta j}\right)^2 \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix}$$

Credit: Kenneth Dawson-Howe, A Practical Introduction to Computer Vision with OpenCV, © Wiley & Sons Inc. 2014

# Interest Point Detection

Consider the intensity variation for a displacement $(\Delta i, \Delta j)$ as sum of squared differences $SSD$

Comparing

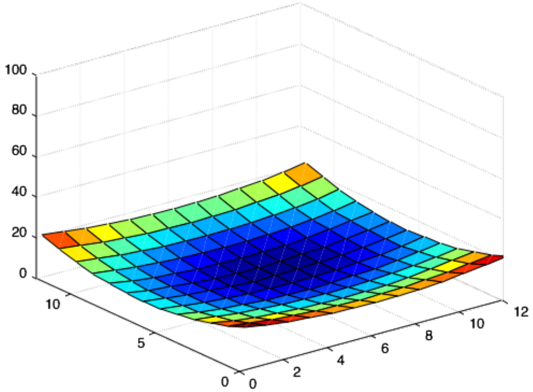
$$E(u,v) \cong [u,v] \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

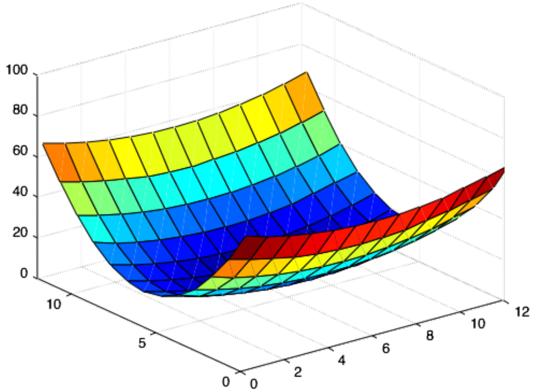
$$SSD_W(\Delta i, \Delta j) = [\Delta i \quad \Delta j] \begin{bmatrix} \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta i}\right)^2 & \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i}\frac{\delta f(i,j)}{\delta j} \\ \sum_{(i,j)\in W} \frac{\delta f(i,j)}{\delta i}\frac{\delta f(i,j)}{\delta j} & \sum_{(i,j)\in W} \left(\frac{\delta f(i,j)}{\delta j}\right)^2 \end{bmatrix} \begin{bmatrix} \Delta i \\ \Delta j \end{bmatrix}$$

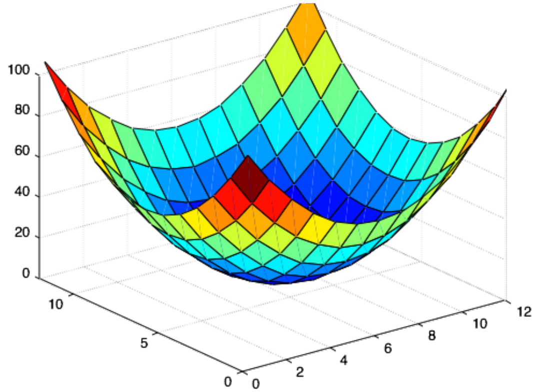
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

# Interest Point Detection

Which error surface indicates a good image feature?



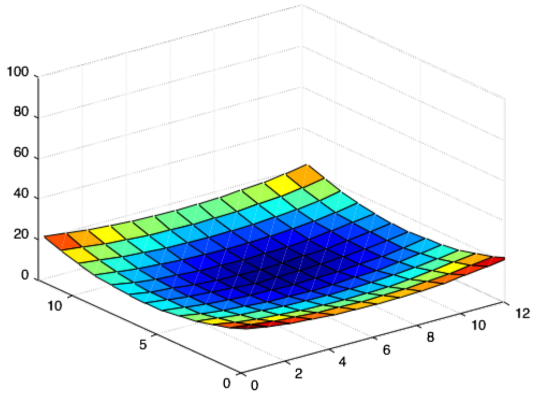



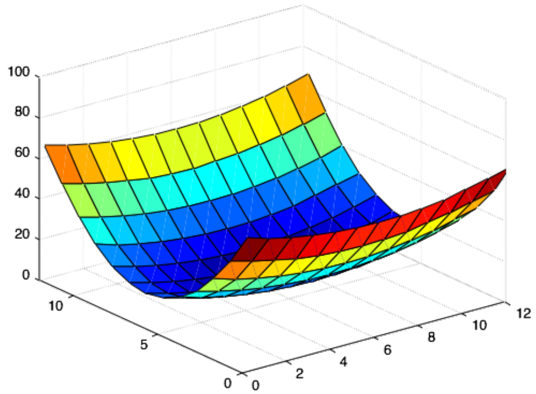

What kind of image patch do these surfaces represent?

Credit: Kris Kitani, Carnegie Mellon University

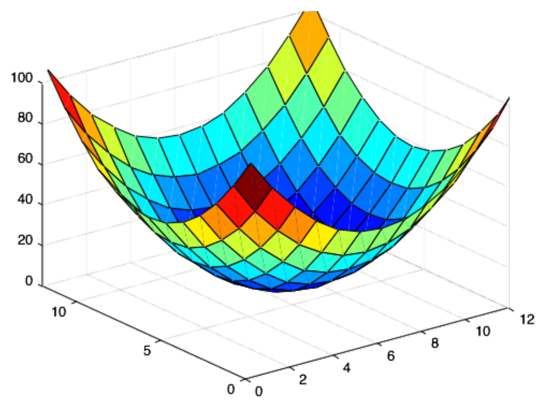
# Interest Point Detection




flat



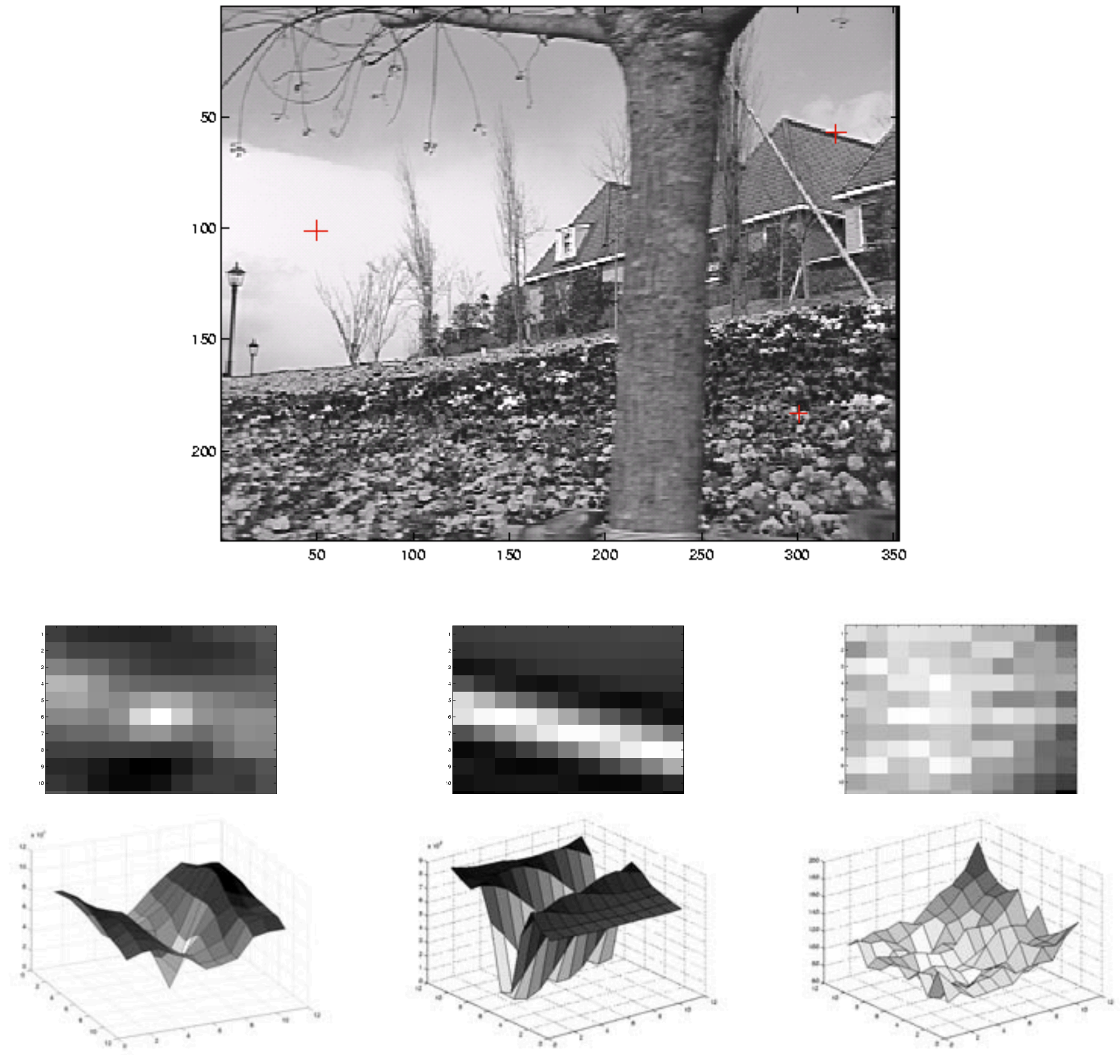

edge
‘line’




corner
‘dot’

Credit: Kris Kitani, Carnegie Mellon University

# Interest Point Detection



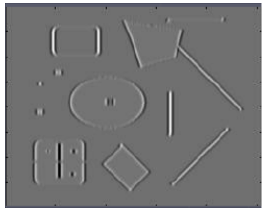

Autocorrelation $E(u, v)$ in neighbourhood of a point

Credit: Szeliski 2010

# Interest Point Detection

1. Compute image gradients over small region

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



2. Subtract mean from each image gradient

3. Compute the covariance matrix

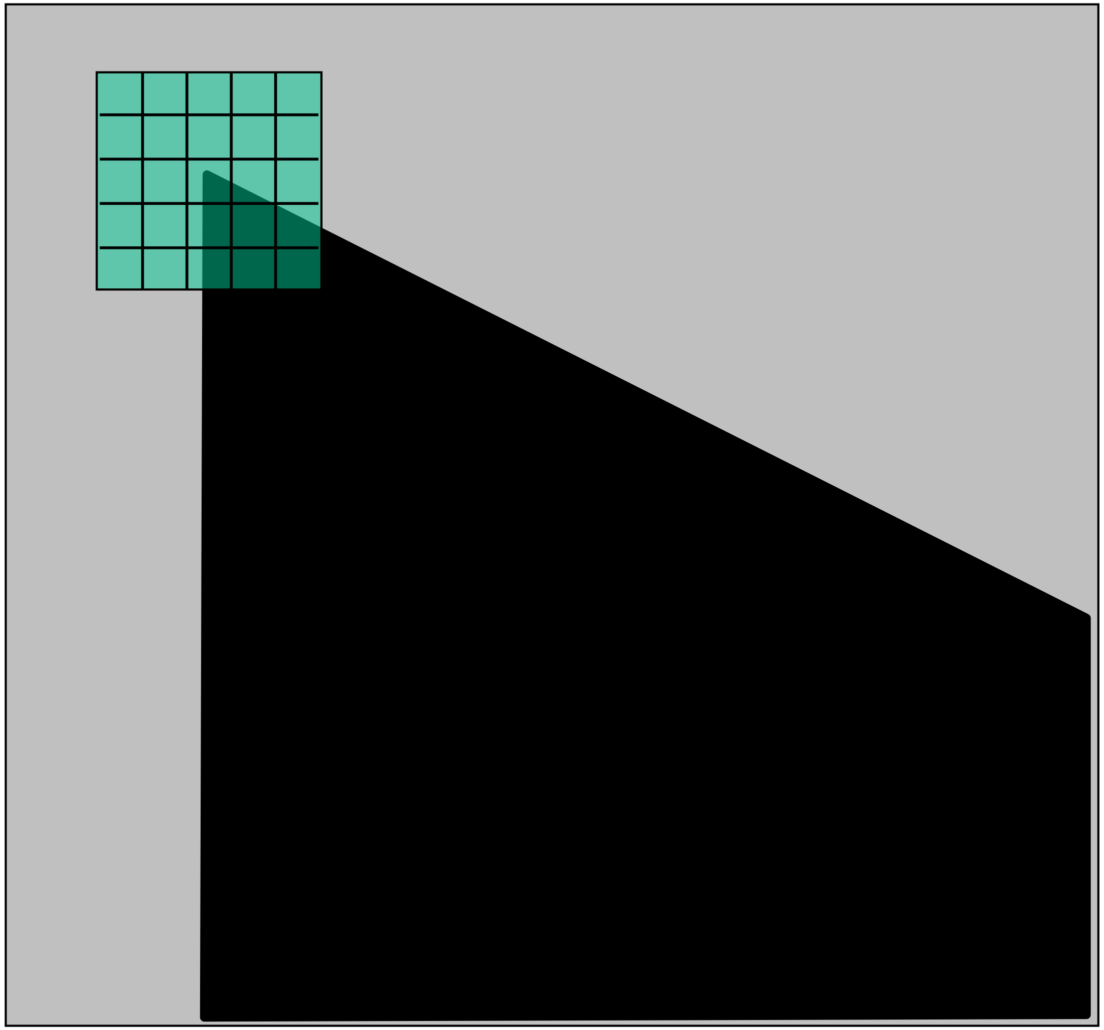
$$\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$$

4. Compute eigenvectors and eigenvalues

5. Use threshold on eigenvalues to detect corners

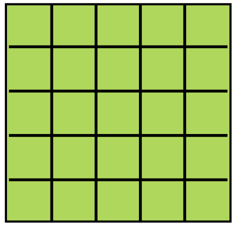
Credit: Kris Kitani, Carnegie Mellon University

1. Compute image gradients over a small region (not just a single pixel)



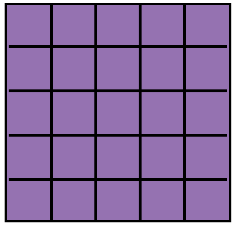
array of x gradients

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



array of y gradients

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

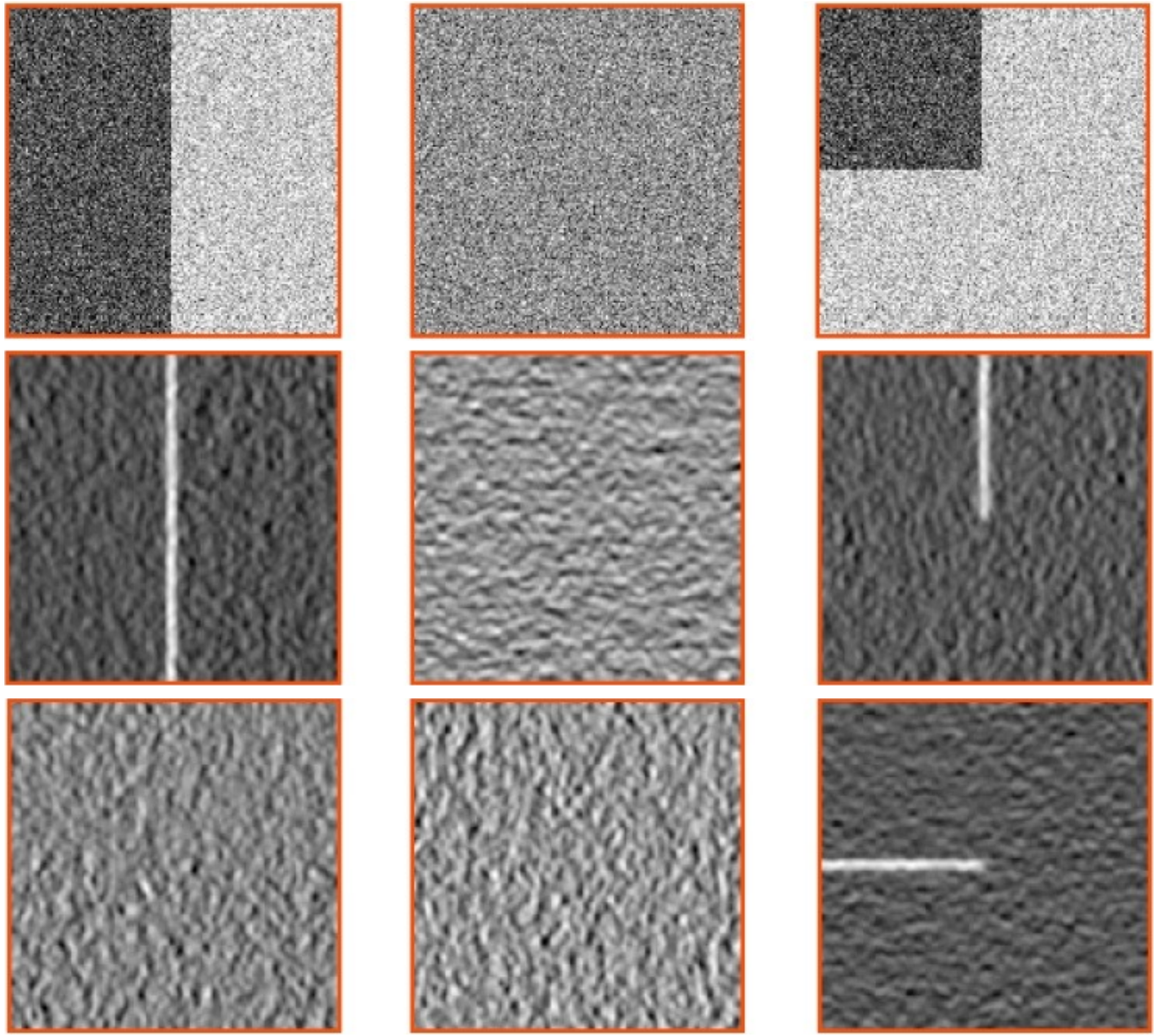


Credit: Kris Kitani, Carnegie Mellon University

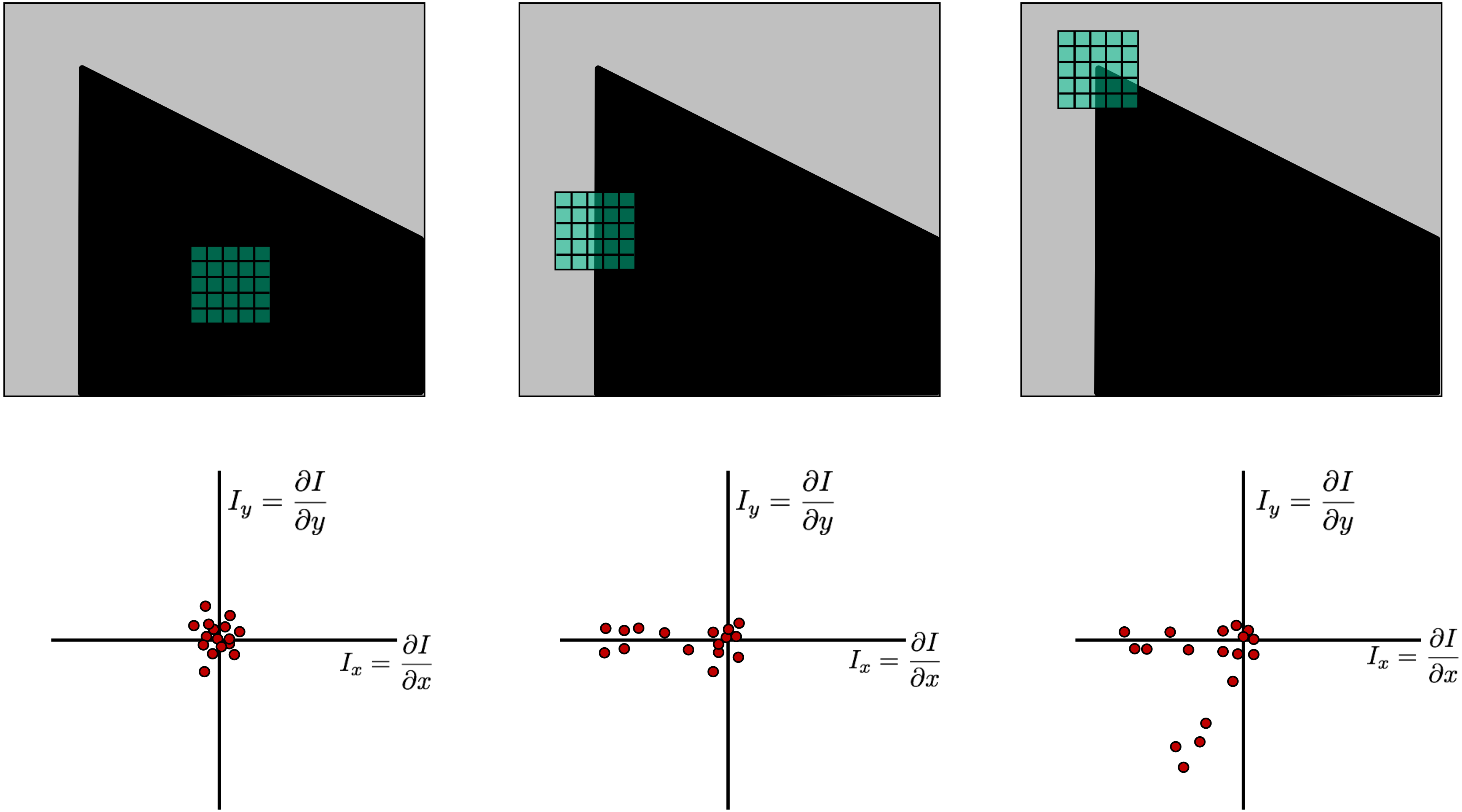
image

X derivative

Y derivative

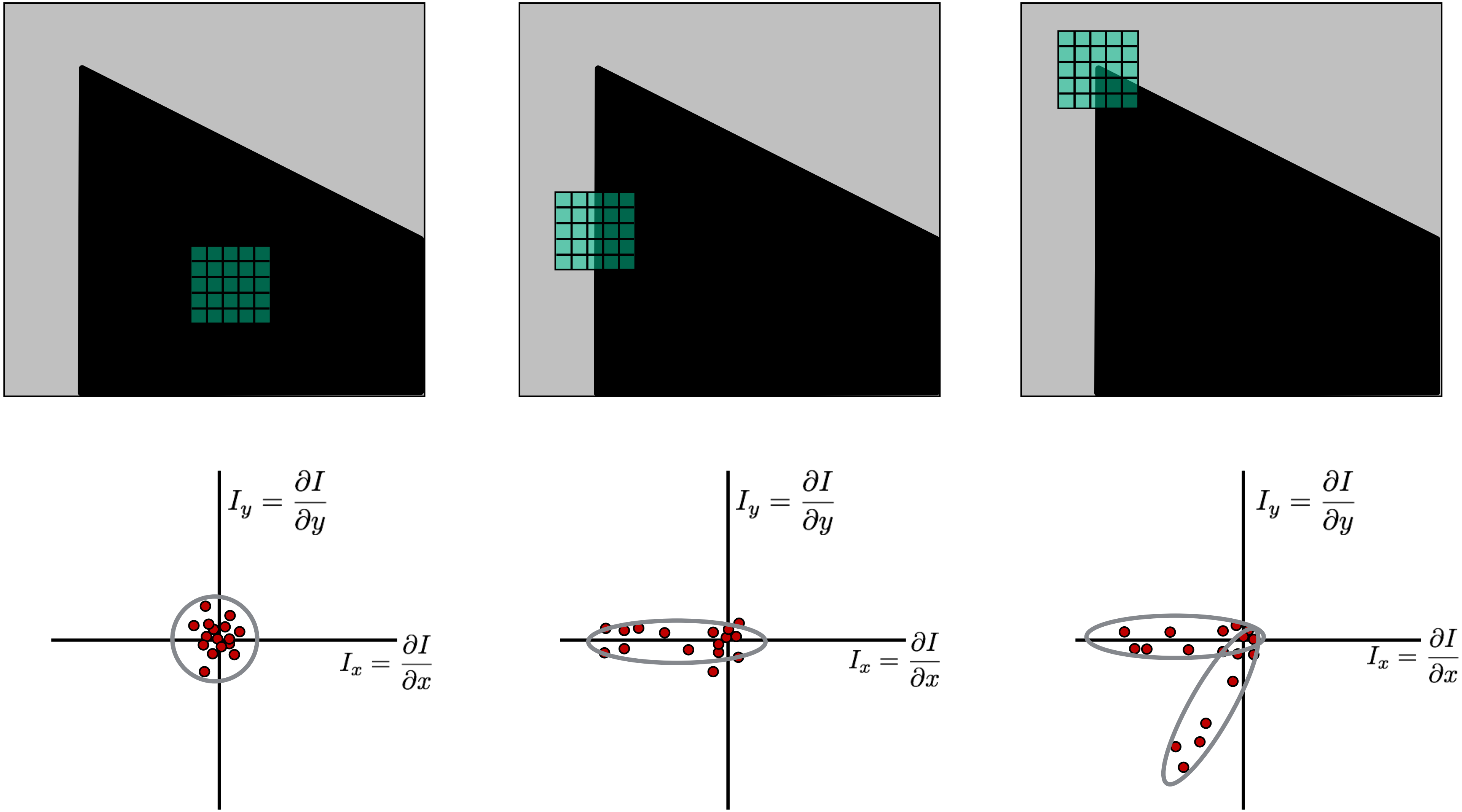


Credit: Kris Kitani, Carnegie Mellon University



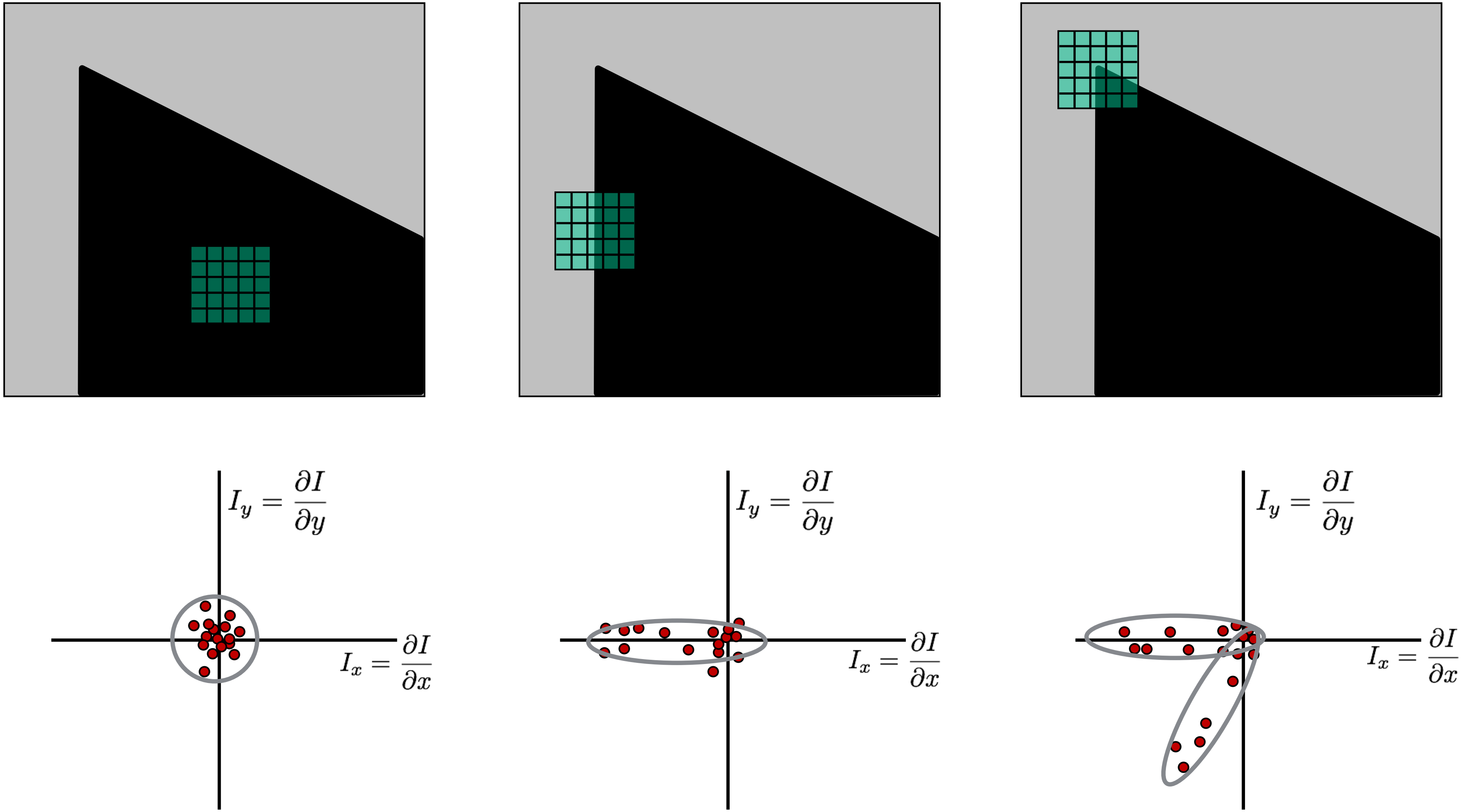

## What does the distribution tell you about the region?

Credit: Kris Kitani, Carnegie Mellon University




# Distribution reveals edge orientation and magnitude

Credit: Kris Kitani, Carnegie Mellon University

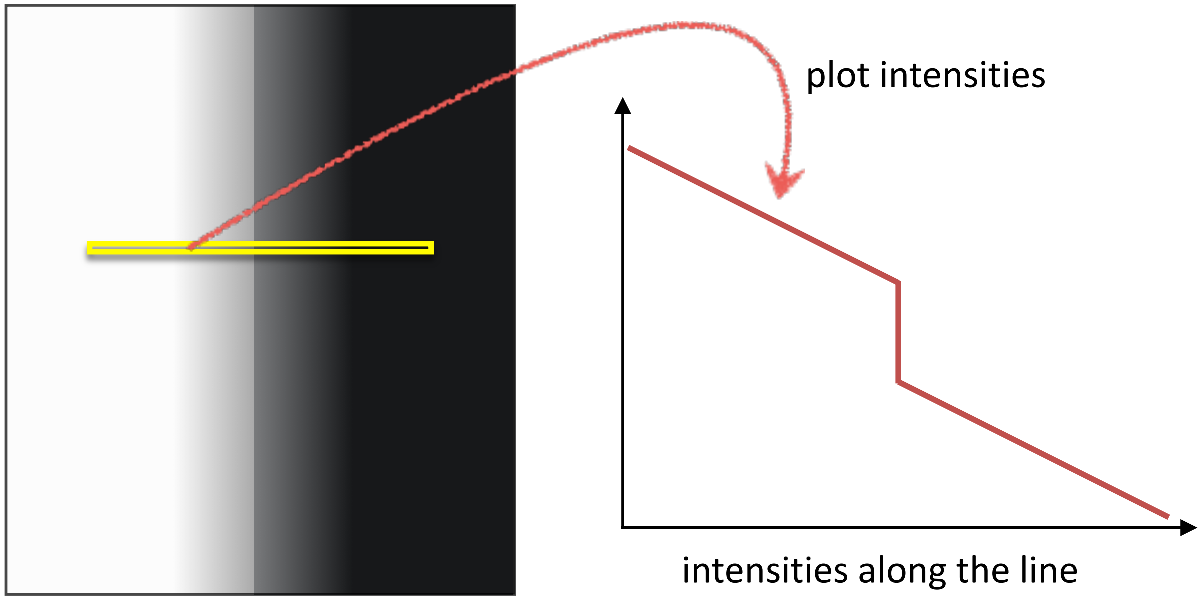



## How do you quantify orientation and magnitude?

Credit: Kris Kitani, Carnegie Mellon University

# 2. Subtract the mean from each image gradient

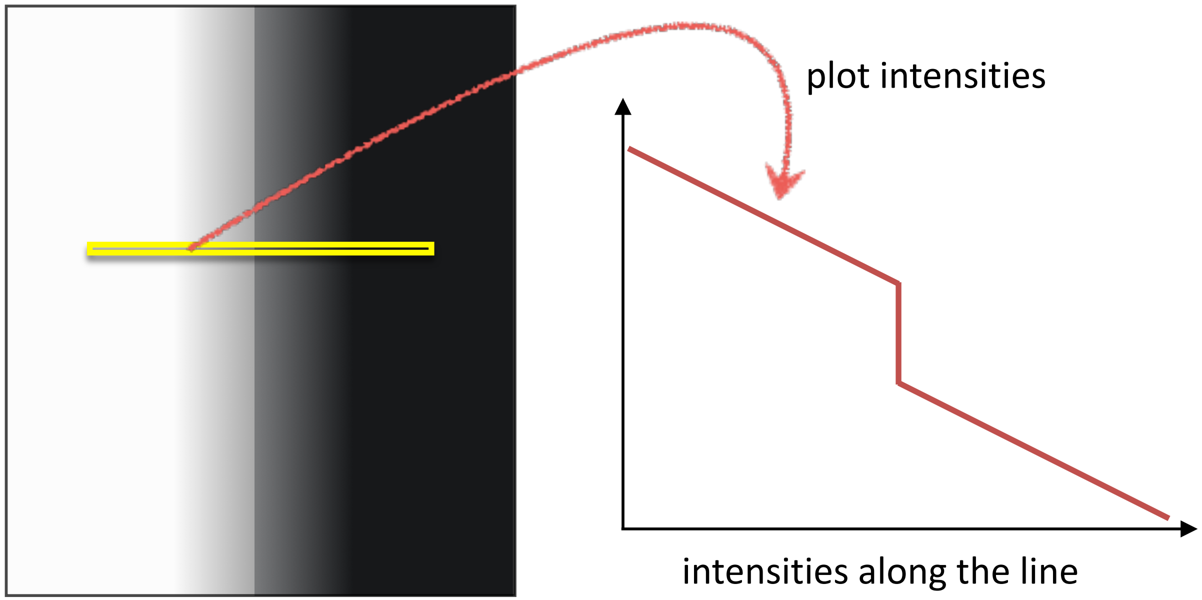
constant intensity gradient

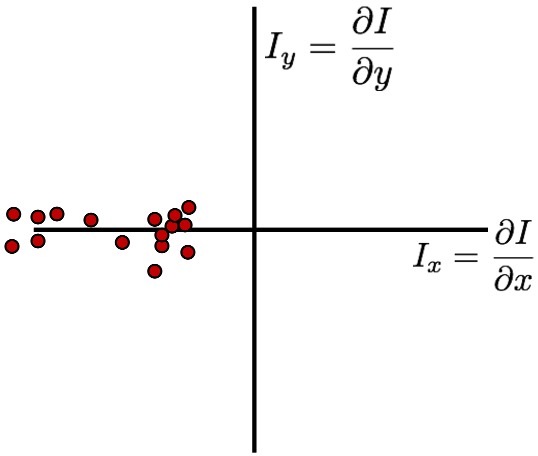



Credit: Kris Kitani, Carnegie Mellon University

# 2. Subtract the mean from each image gradient

constant intensity gradient



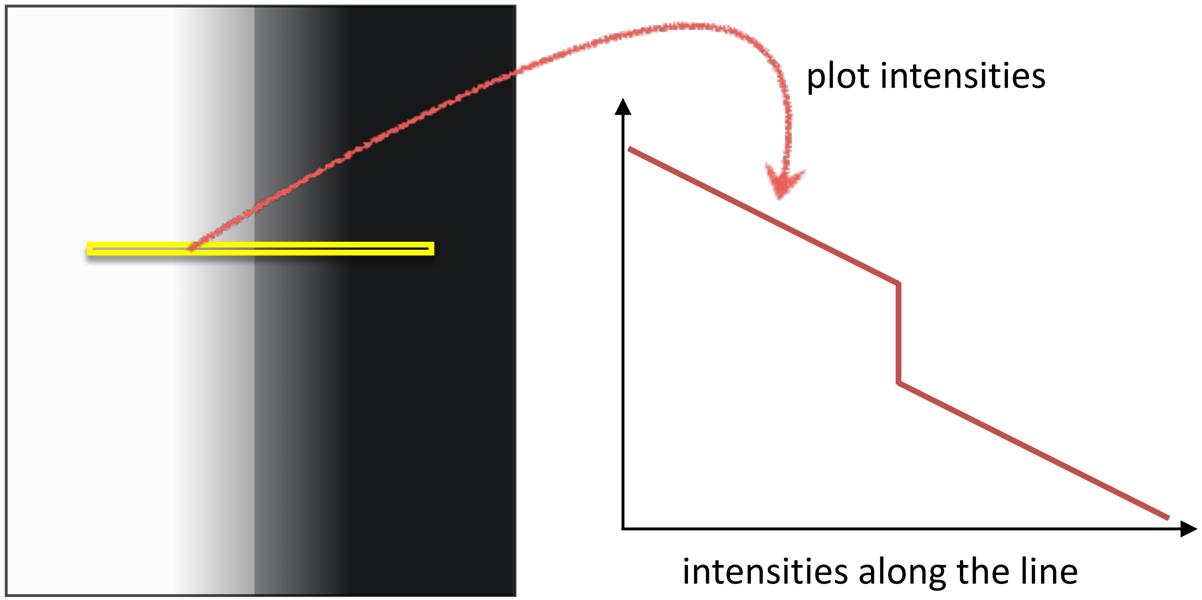

subtract mean

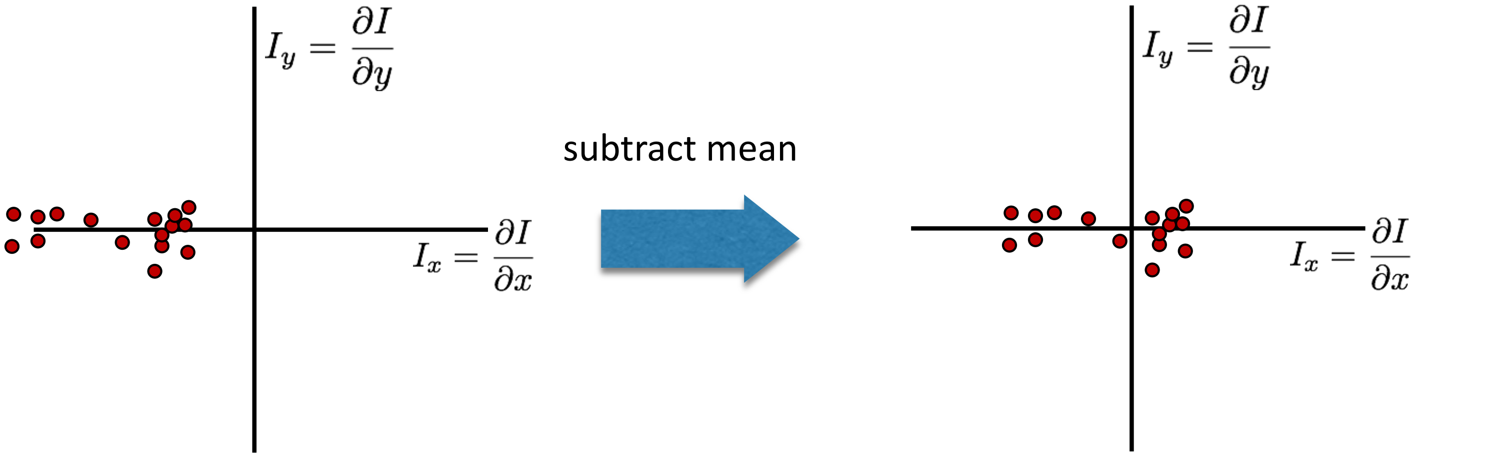
plot of image gradients

Credit: Kris Kitani, Carnegie Mellon University

# 2. Subtract the mean from each image gradient

constant intensity gradient




plot of image gradients

data is centered
('DC' offset is removed)

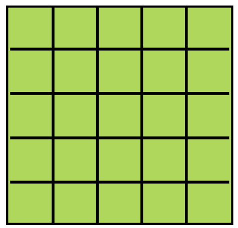
Credit: Kris Kitani, Carnegie Mellon University

# 3. Compute the covariance matrix

$$\begin{bmatrix} \sum\limits_{p \in P} I_x I_x & \sum\limits_{p \in P} I_x I_y \\ \sum\limits_{p \in P} I_y I_x & \sum\limits_{p \in P} I_y I_y \end{bmatrix}$$

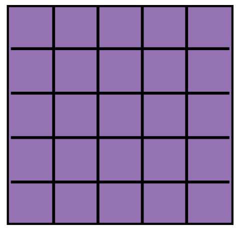
$$I_x = \frac{\partial I}{\partial x} \qquad I_y = \frac{\partial I}{\partial y}$$

$\sum\limits_{p \in P} I_x I_y$ =sum(



array of x gradients

.*



array of y gradients

)

Where does this covariance matrix come from?

Credit: Kris Kitani, Carnegie Mellon University

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$Me = \lambda e$$

eigenvector

$$(M - \lambda I)e = 0$$

Credit: Kris Kitani, Carnegie Mellon University

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$M\boldsymbol{e} = \lambda \boldsymbol{e}$$

eigenvector

$$(M - \lambda I)\boldsymbol{e} = 0$$

1. Compute the determinant of $M - \lambda I$ (returns a polynomial)

Credit: Kris Kitani, Carnegie Mellon University

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$\mathbf{M}\boldsymbol{e} = \lambda\boldsymbol{e}$$

eigenvector

$$(\boldsymbol{M} - \lambda \boldsymbol{I})\boldsymbol{e} = 0$$

1. Compute the determinant of (returns a polynomial) $M - \lambda I$

2. Find the roots of polynomial (returns eigenvalues) $\det(M - \lambda I) = 0$

Credit: Kris Kitani, Carnegie Mellon University

# 4. Compute eigenvalues and eigenvectors

eigenvalue

$$Me = \lambda e$$

eigenvector

$$(M - \lambda I)e = 0$$

1. Compute the determinant of (returns a polynomial) $M - \lambda I$

2. Find the roots of polynomial (returns eigenvalues) $\det(M - \lambda I) = 0$

3. For each eigenvalue, solve (returns eigenvectors) $(M - \lambda I)e = 0$

Credit: Kris Kitani, Carnegie Mellon University

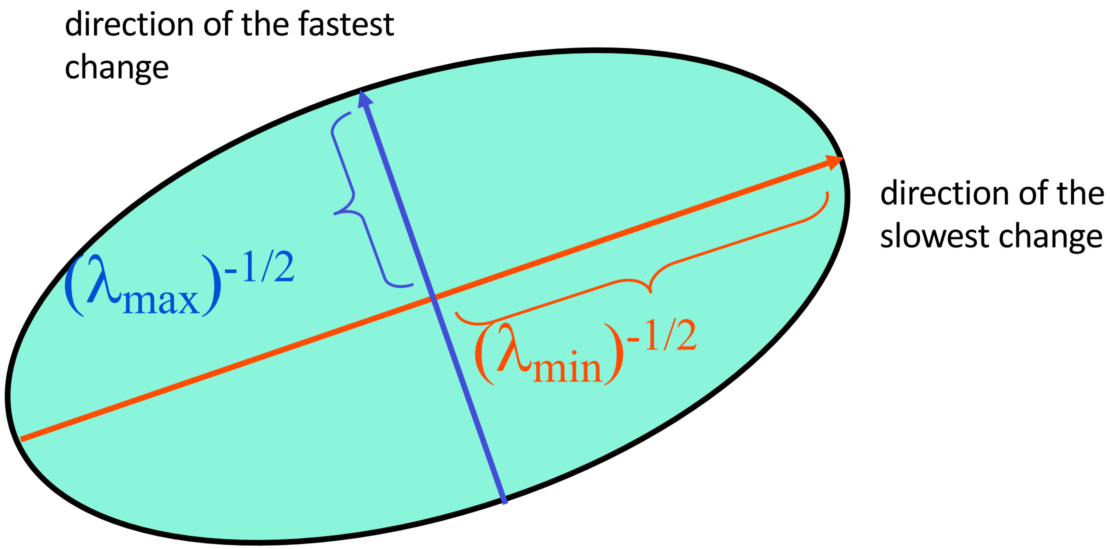
Since $M$ is symmetric, we have

$$M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

We can visualize $M$ as an ellipse with axis lengths determined by the eigenvalues and orientation determined by $R$

Ellipse equation:

$$[u \;\; v] \; M \begin{bmatrix} u \\ v \end{bmatrix} = \text{const}$$

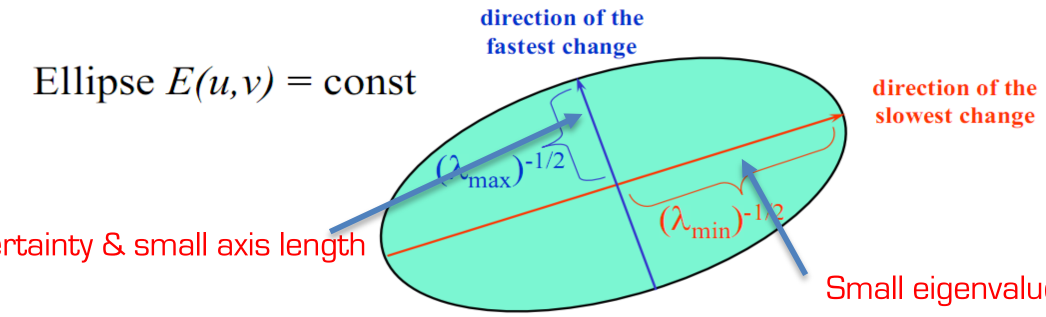



Credit: Kris Kitani, Carnegie Mellon University

## Eigenvalue analysis of $M$

- Eigenvalues $\lambda_1$ and $\lambda_2$ of $M$ indicate maximum and minimum directions of gradient, respectively

- Larger uncertainty depends on the smaller eigenvalue
  - Find points where the value of the smaller eigenvalue is large (i.e. where both eigenvalues are large)
  - These indicate good features to track , i.e. corners

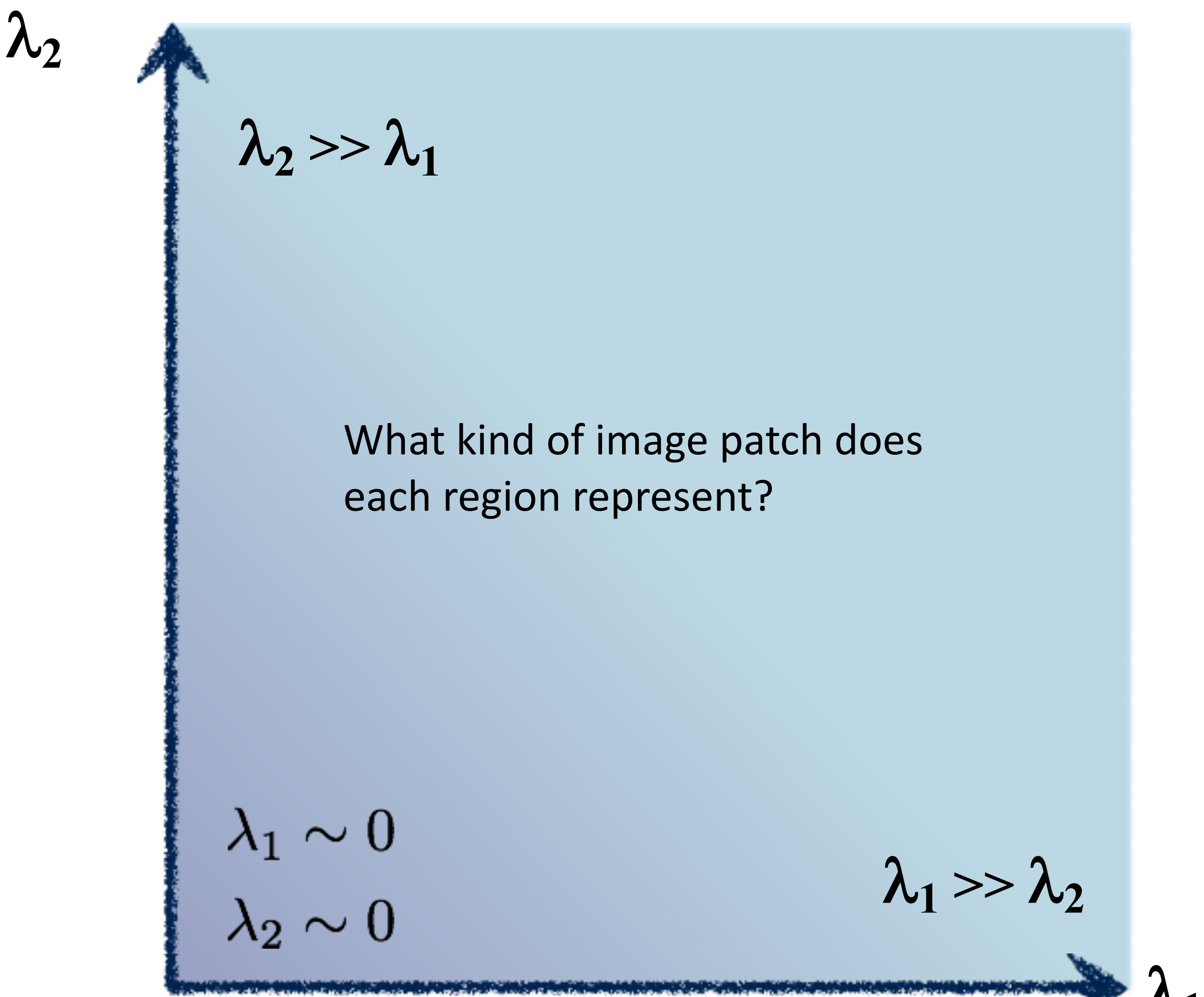



Large eigenvalue: small uncertainty & small axis length

Small eigenvalue: big uncertainty & big axis length

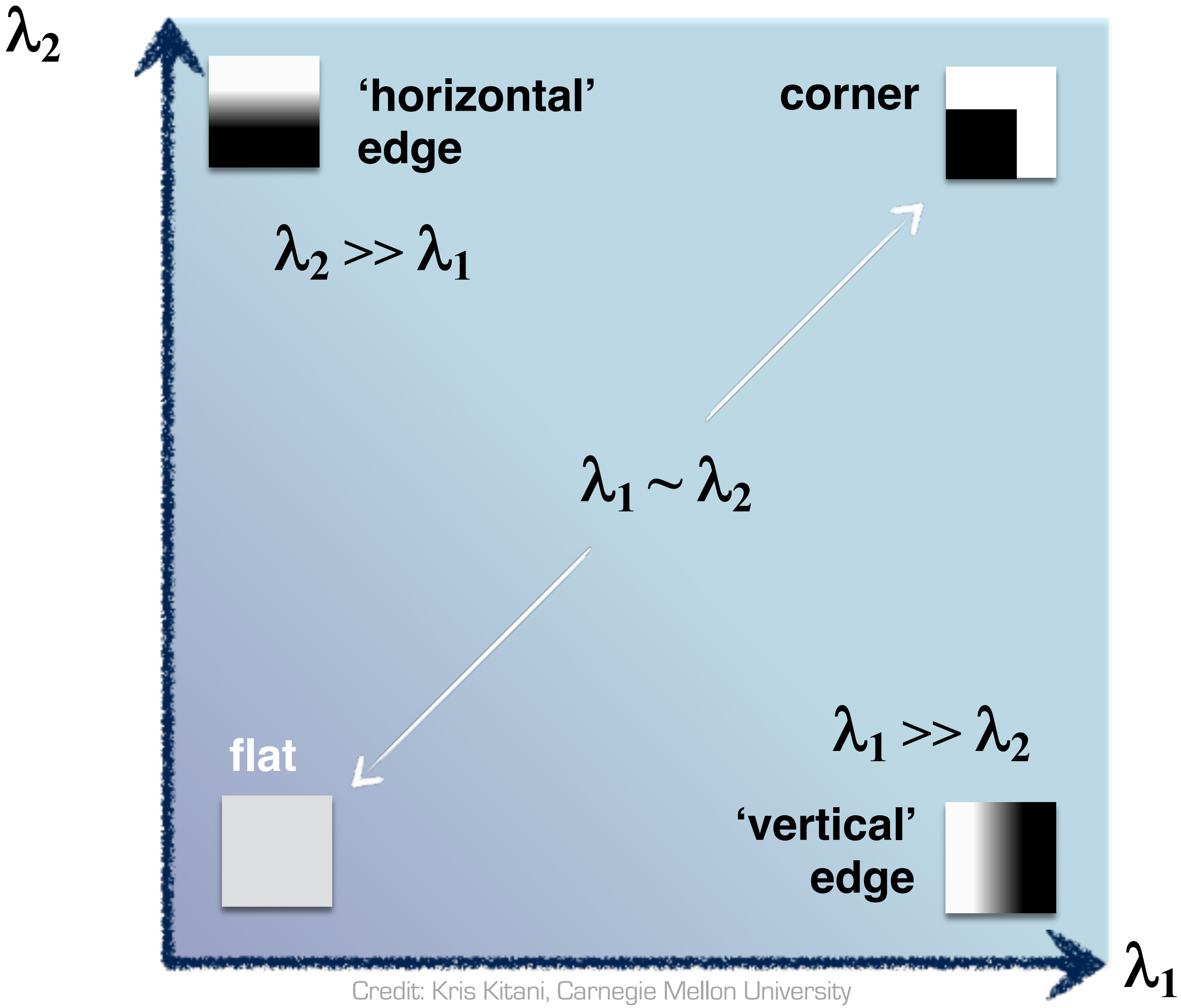
Credit: Markus Vincze, Technische Universität Wien

# Interpreting eigenvalues



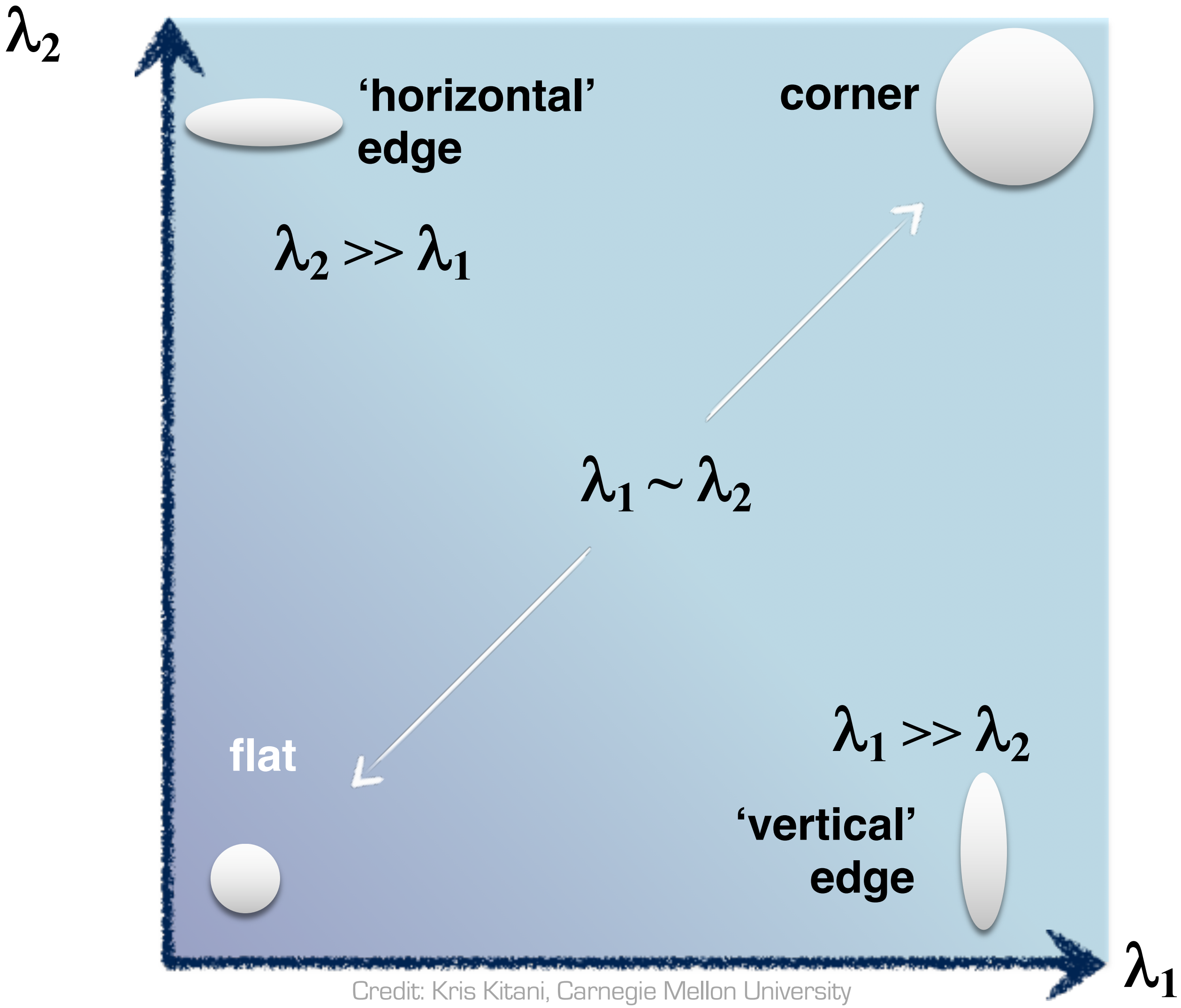

Credit: Kris Kitani, Carnegie Mellon University

# Interpreting eigenvalues



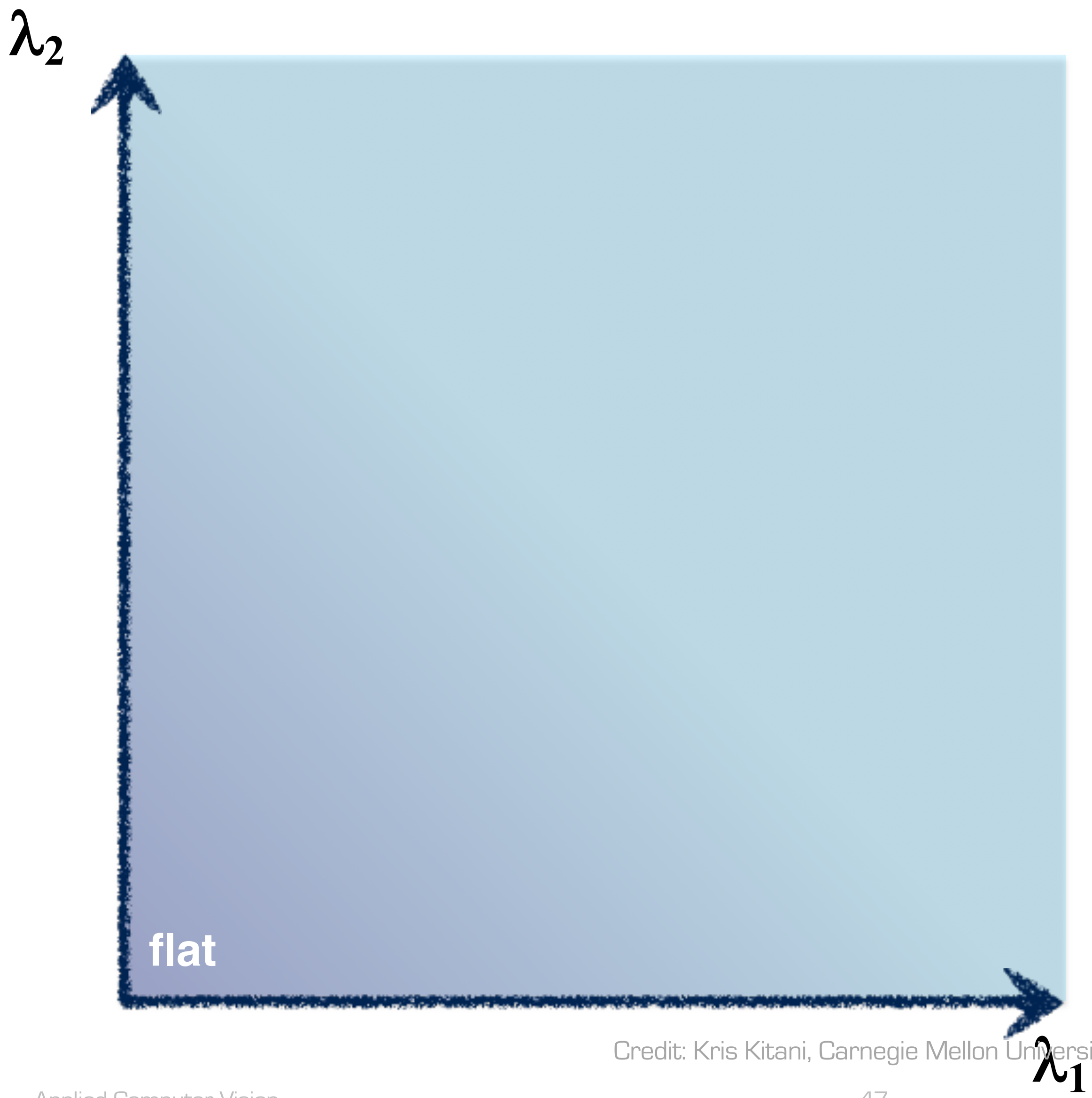

Credit: Kris Kitani, Carnegie Mellon University

# Interpreting eigenvalues




Credit: Kris Kitani, Carnegie Mellon University

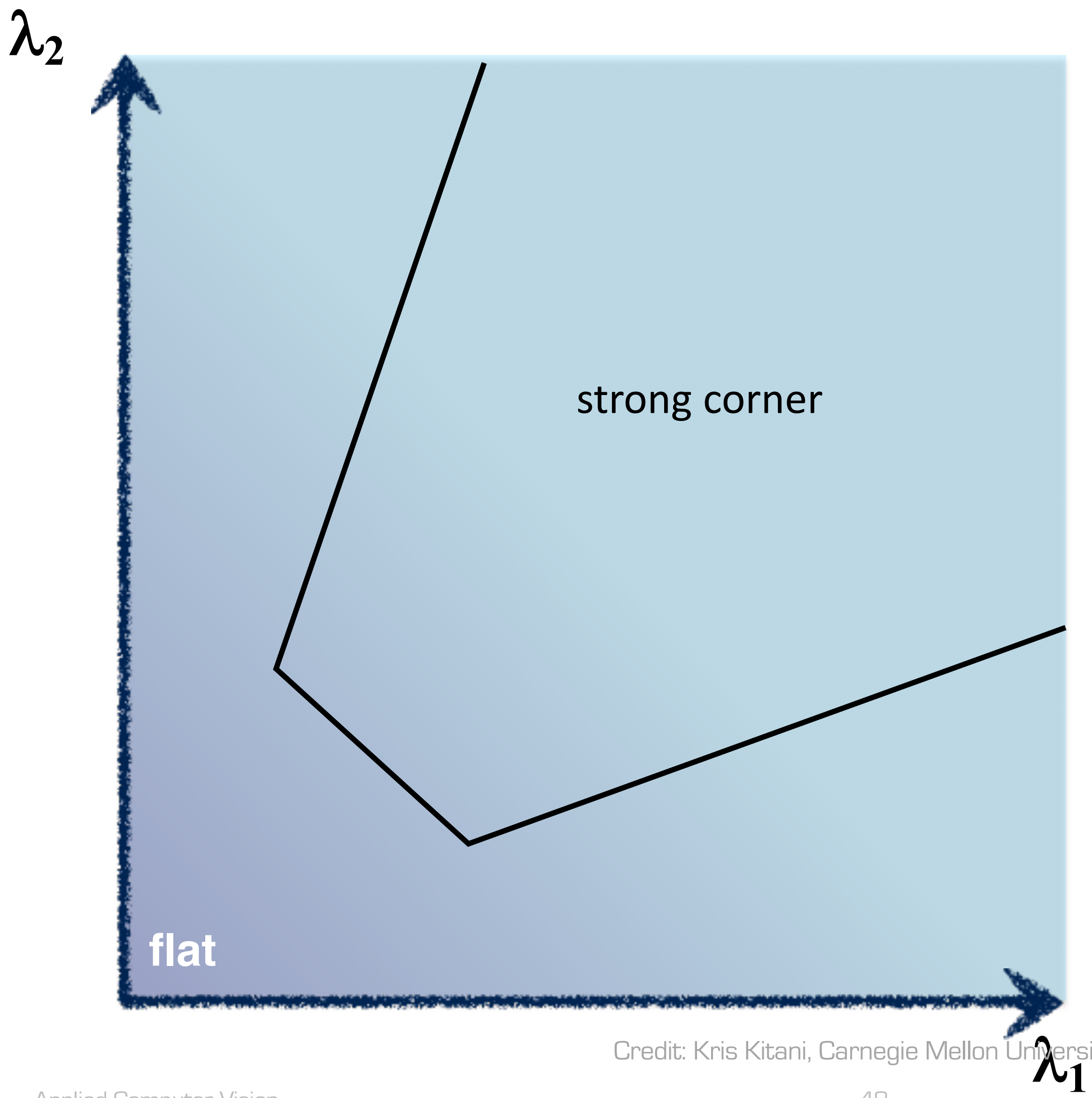
# 5. Use threshold on eigenvalues to detect corners




Think of a function to score 'cornerness'

Credit: Kris Kitani, Carnegie Mellon University

# 5. Use threshold on eigenvalues to detect corners

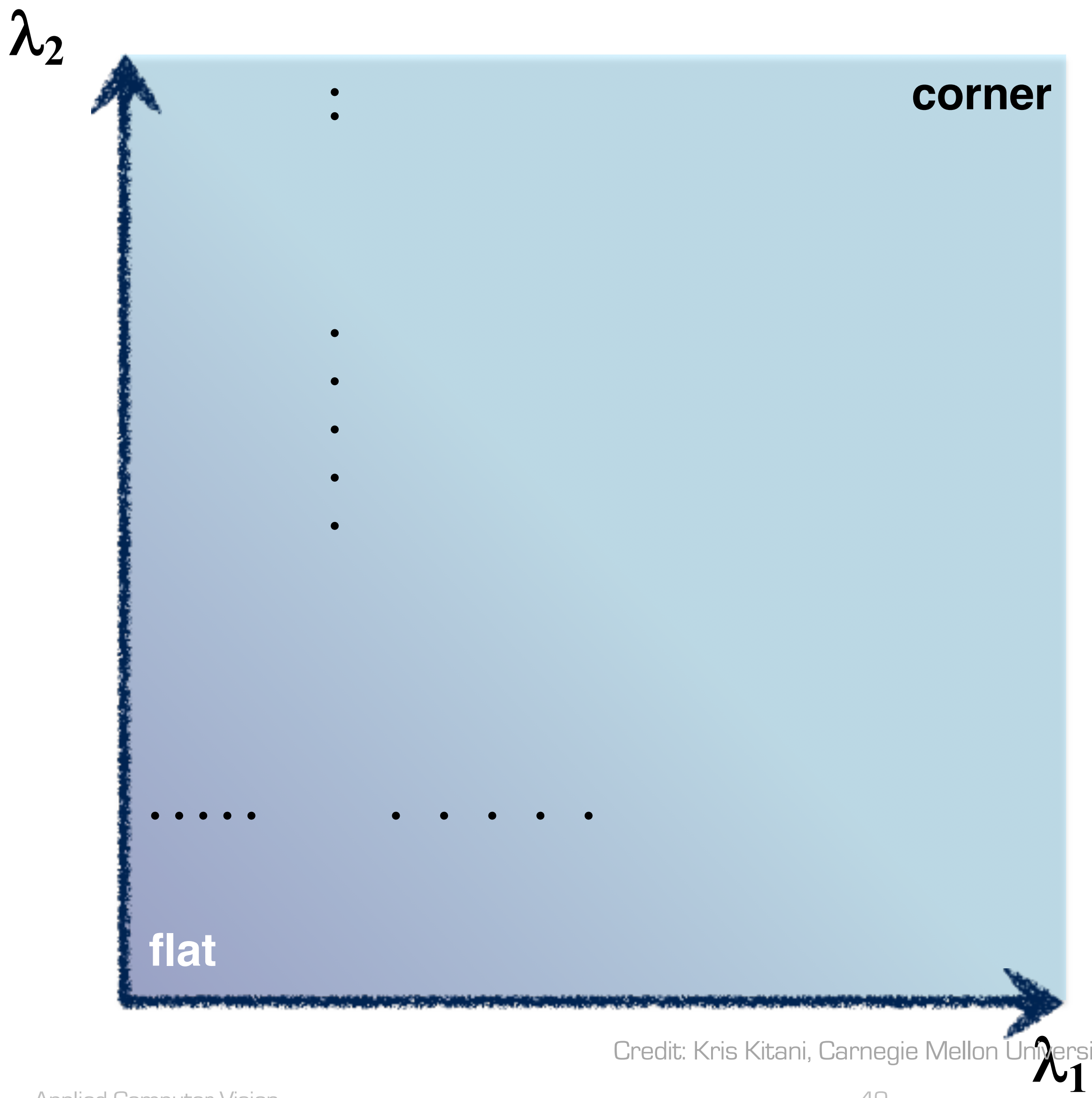



Think of a function to score 'cornerness'

Credit: Kris Kitani, Carnegie Mellon University

# 5. Use threshold on eigenvalues to detect corners

(a function of )



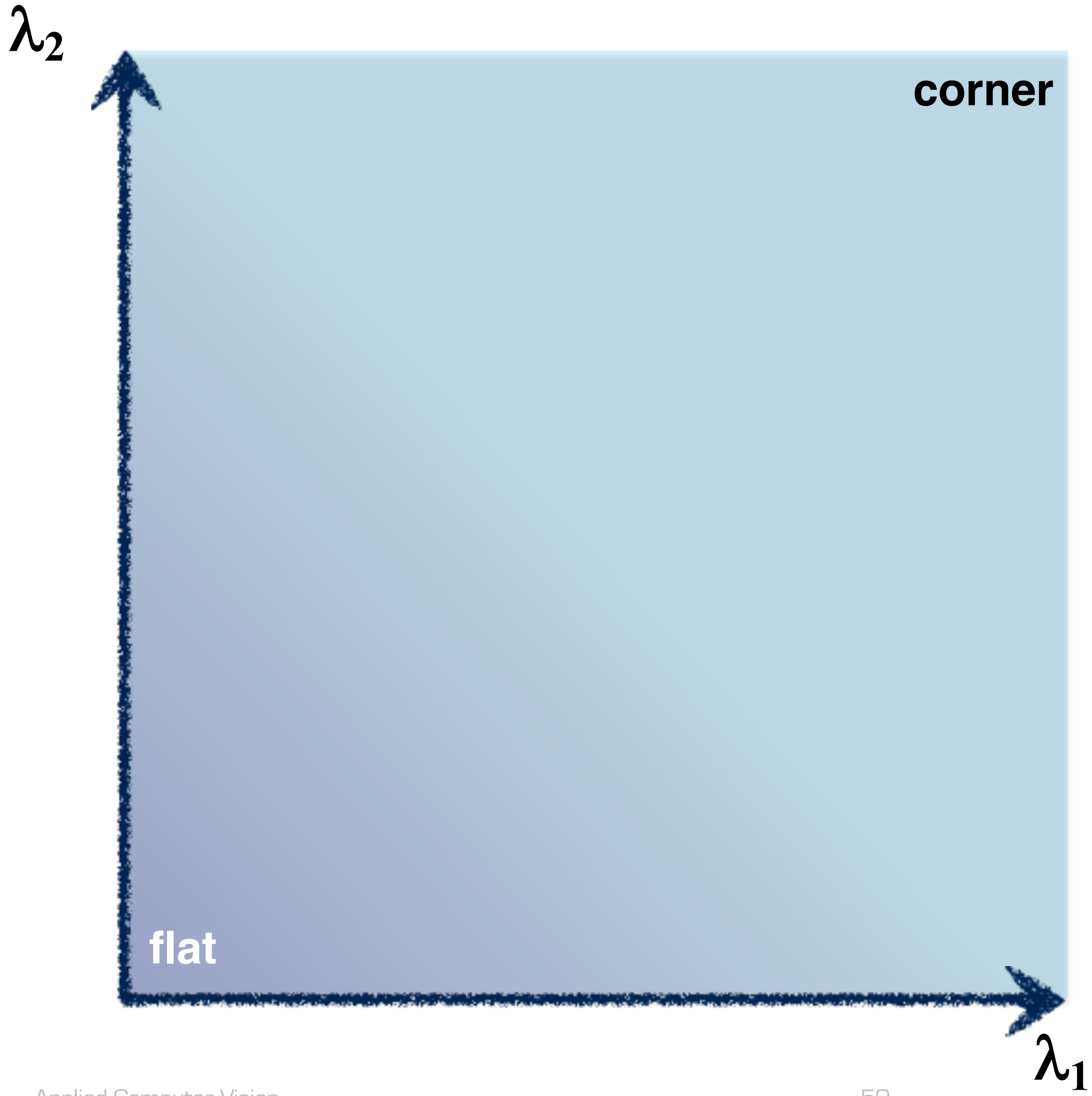

Use the smallest eigenvalue as the response function

$$R = \min(\lambda_1, \lambda_2)$$

Credit: Kris Kitani, Carnegie Mellon University

# 5. Use threshold on eigenvalues to detect corners

(a function of )



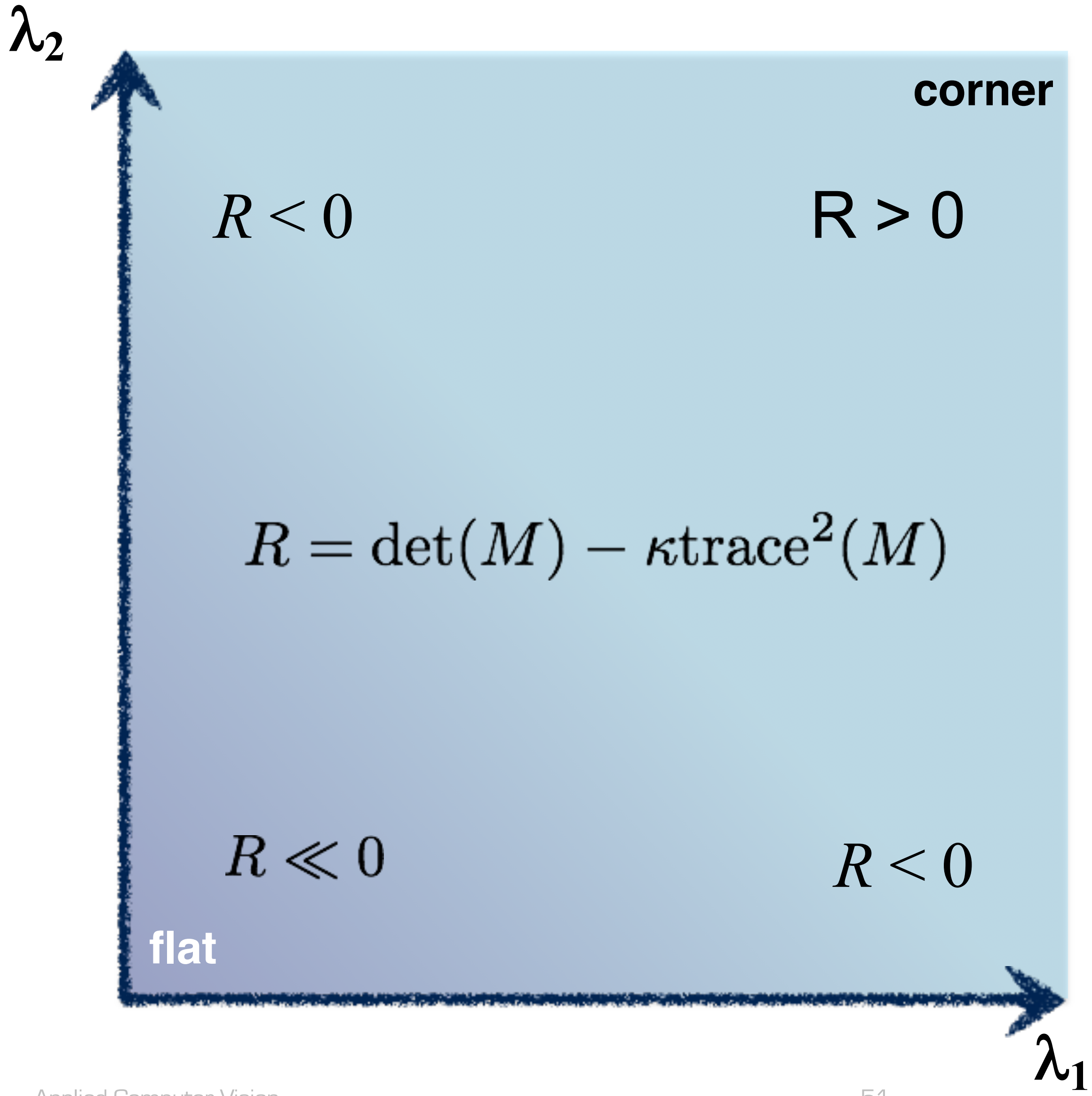

Eigenvalues need to be bigger than one.

$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Can compute this more efficiently…

# 5. Use threshold on eigenvalues to detect corners

(a function of )




$$\det M = \lambda_1 \lambda_2$$

$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

$$det\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = ad - bc$$

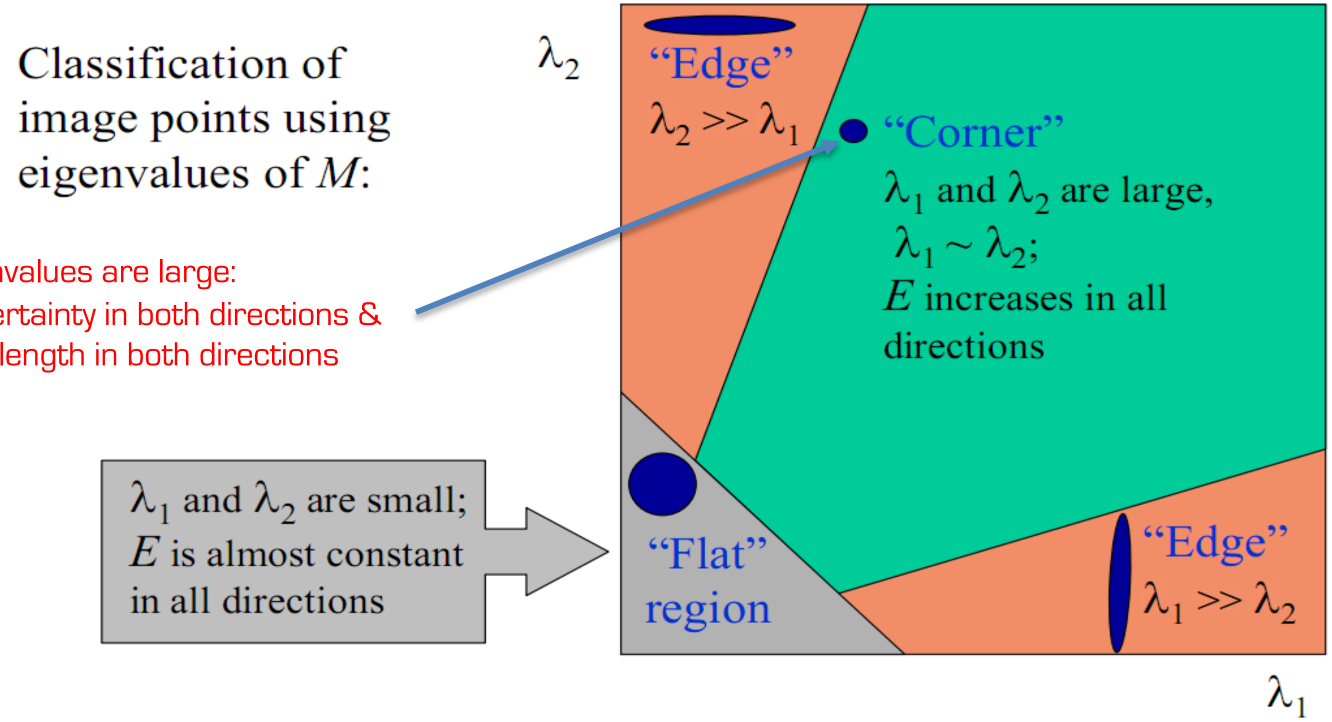
$$trace\left(\begin{bmatrix} a & b \\ c & d \end{bmatrix}\right) = a + d$$

# Summary

Intensity change in shifting window: eigenvalue analysis

Eigenvalues. $\lambda_1$ and $\lambda_2$ of $M$ indicate max and min directions of gradient

Both eigenvalues are large:
small uncertainty in both directions &
small axis length in both directions




Credit: Markus Vincze, Technische Universität Wien

# Summary

Measure of “cornerness” without computing eigenvalues explicitly:

Maximize $\det M$

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

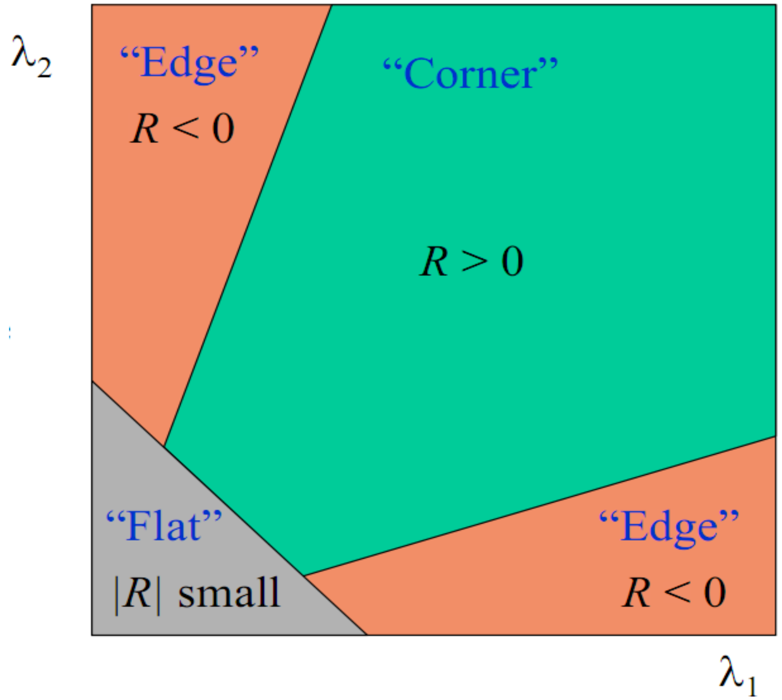
$$\begin{aligned} \det M &= \lambda_1 \lambda_2 \\ \operatorname{trace} M &= \lambda_1 + \lambda_2 \end{aligned}$$

($k$ – empirical constant, $k = 0.04$-$0.06$)

Credit: Markus Vincze, Technische Universität Wien

# Summary

- $R$ is negative for edges
- $R$ is small for flat regions
- Find points with large measure of cornerness ($R >$ threshold)
- Take points that are local maxima in $R$




Credit: Markus Vincze, Technische Universität Wien

Harris & Stephens (1988)

$$R = \det(M) - \kappa \operatorname{trace}^2(M)$$

Kanade & Tomasi (1994)

$$R = \min(\lambda_1, \lambda_2)$$

Nobel (1998)

$$R = \frac{\det(M)}{\operatorname{trace}(M) + \epsilon}$$

Credit: Kris Kitani, Carnegie Mellon University

# Harris Detector

C.Harris and M.Stephens. "A Combined Corner and Edge Detector." 1988.

1. Compute x and y derivatives of image

$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$

2. Compute products of derivatives at every pixel

$$I_{x^2} = I_x \cdot I_x \qquad I_{y^2} = I_y \cdot I_y \qquad I_{xy} = I_x \cdot I_y$$

3. Compute the sums of the products of derivatives at each pixel

$$S_{x^2} = G_{\sigma'} * I_{x^2} \qquad S_{y^2} = G_{\sigma'} * I_{y^2} \qquad S_{xy} = G_{\sigma'} * I_{xy}$$

Credit: Kris Kitani, Carnegie Mellon University

# Harris Detector

C.Harris and M.Stephens. "A Combined Corner and Edge Detector."1988.

4. Define the matrix at each pixel

$$M(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$

5. Compute the response of the detector at each pixel

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

6. Threshold on value of R; compute non-max suppression.

Credit: Kris Kitani, Carnegie Mellon University

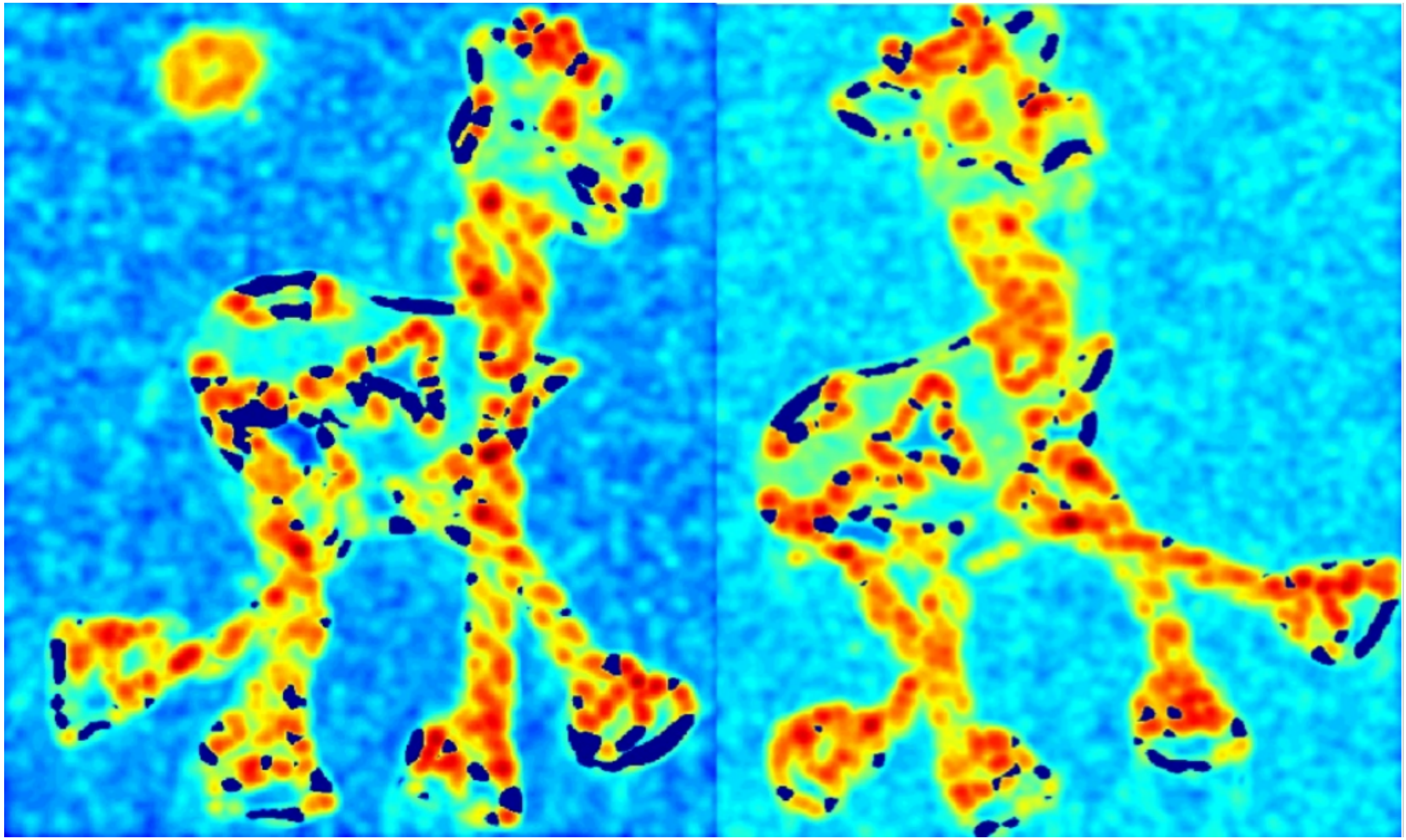
# Harris Corner Detector

Same object with different illumination and pose

Credit: Markus Vincze, Technische Universität Wien

# Harris Corner Detector

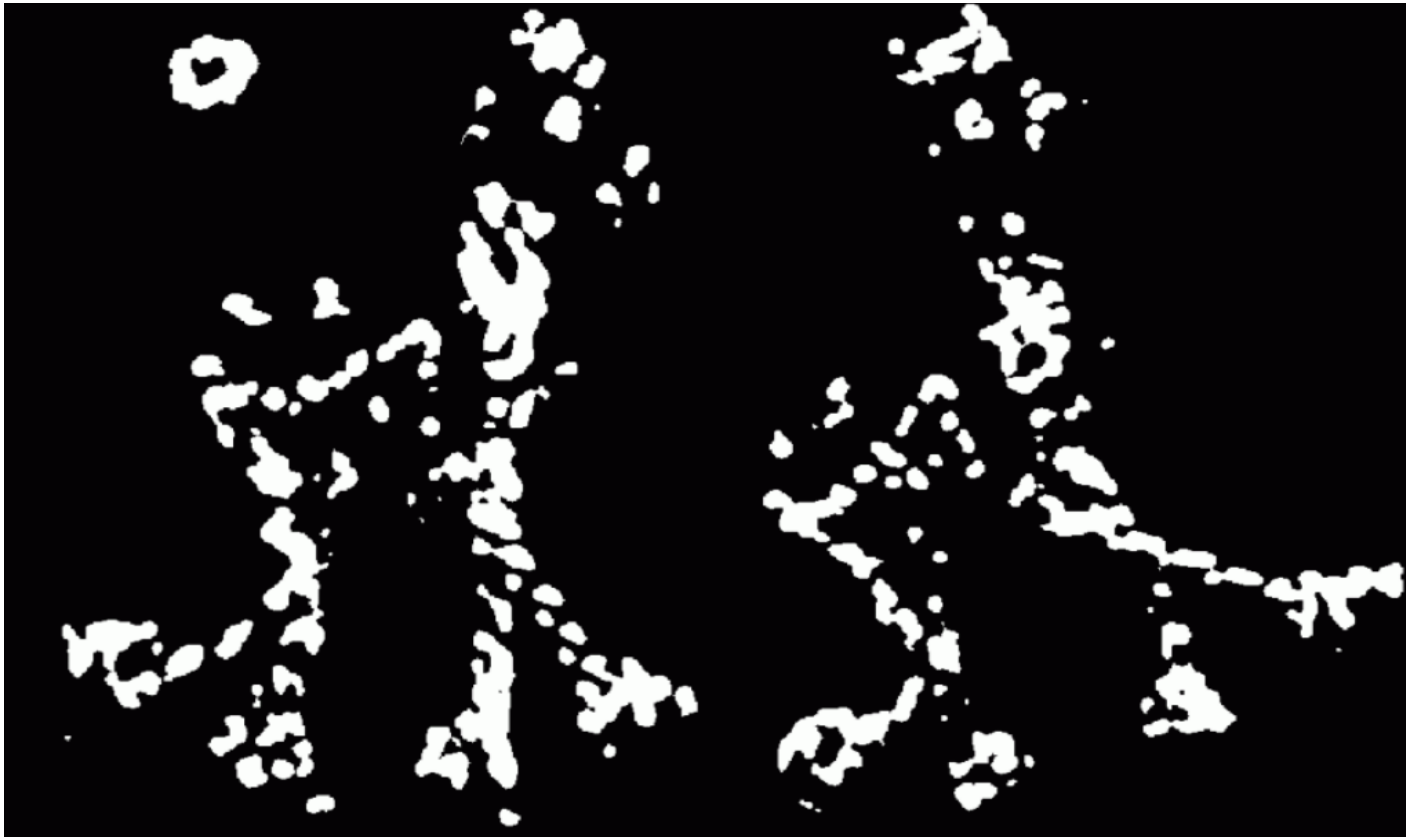
$R$ values



Credit: Markus Vincze, Technische Universität Wien

# Harris Corner Detector

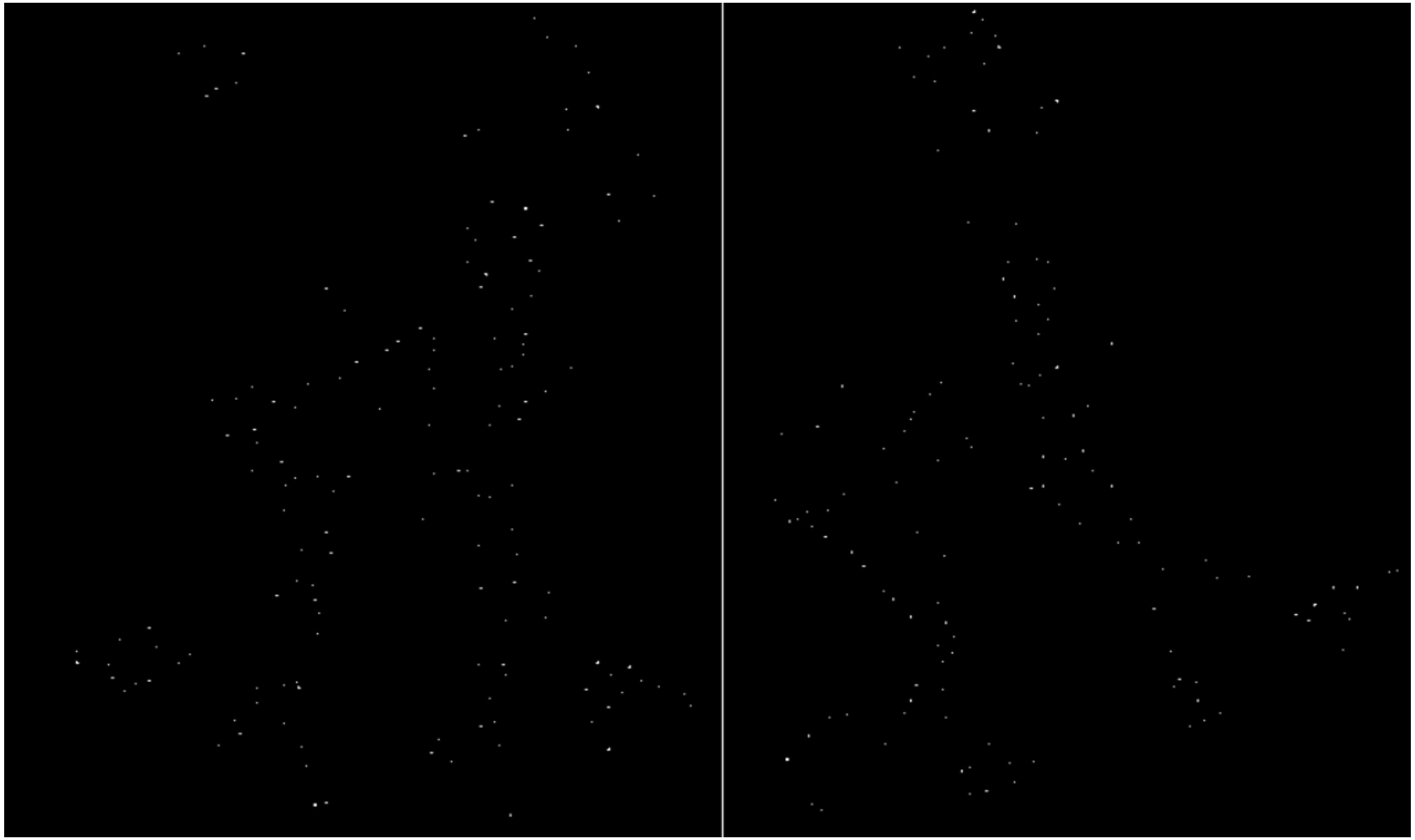
Points (regions) with $R$ larger than a threshold



Credit: Markus Vincze, Technische Universität Wien

# Harris Corner Detector

## Local maxima



Credit: Markus Vincze, Technische Universität Wien

# Harris Corner Detector

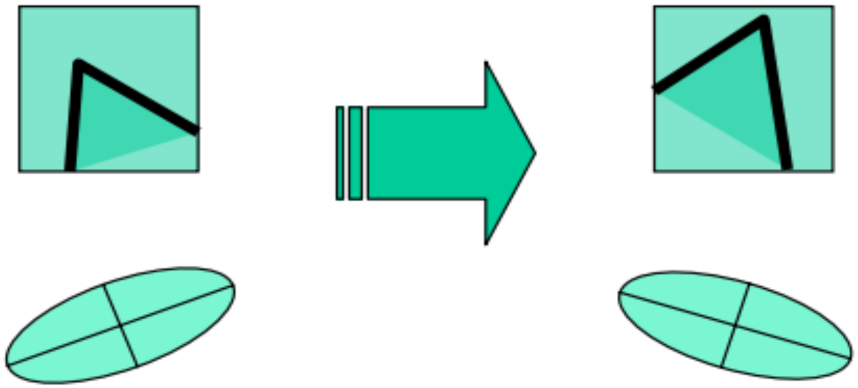
Detected corner points

Credit: Markus Vincze, Technische Universität Wien

# Harris Corner Detector

Properties

- Rotation invariant: corner response is invariant to image rotation
- Ellipse rotates with the corner but the shape and size (i.e. the eigenvalues) remain unchanged



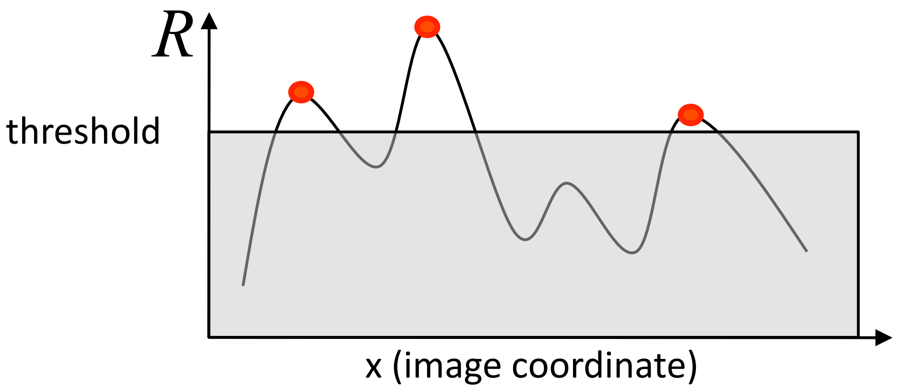
Credit: Markus Vincze, Technische Universität Wien

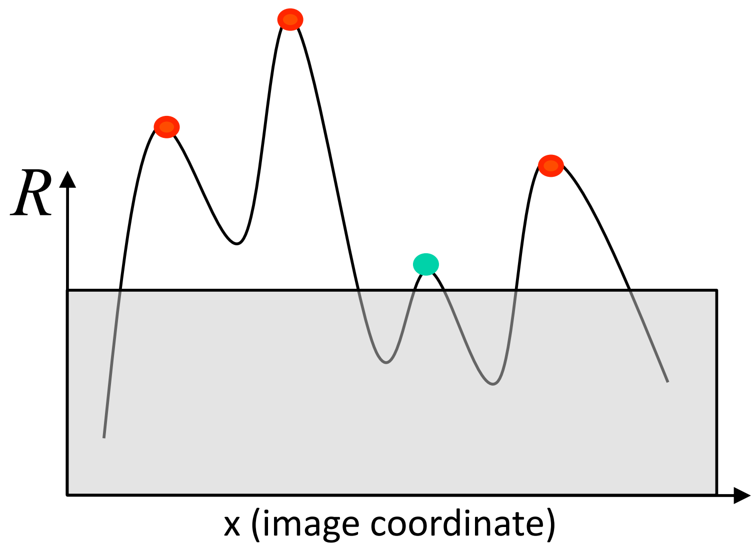
# Harris Corner Detector

Partial invariance to *affine intensity* change

**Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$**

**Intensity scale: $I \rightarrow a\,I$**




Credit: Kris Kitani, Carnegie Mellon University

The Harris corner detector not invariant to changes in …

# Demos

The following code is taken from the harrisCornerDetection project in the lectures directory of the ACV repository

See:

```
harrisCornerDetection.h
harrisCornerDetectionImplementation.cpp
harrisCornerDetectionApplication.cpp
```

```
/*
  Example use of openCV to find interest point features using the Harris corner detector
  -------------------------------------------------------------------------------------
  Implementation file

  see  http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=cornerharris#cornerharris
  and  http://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.html#goodfeaturestotrackdetector

  David Vernon
  24 June 2017
*/

#include "harrisCornerDetection.h"

/*
 * function harrisCornerDetection
 * Trackbar callback - set maximum number of corners / interest points
 * Trackbar callback - set minimum distance between matched points
 * Trackbar callback - set block size over which to compute gradients
*/

void harrisCornerDetection(int, void*) {

   extern Mat   src;
   extern int   num_corners;
   extern int   quality_level;
   extern int   min_distance;
   extern int   block_size;
   extern char* feature_location_window_name;
   extern char* feature_magnitude_window_name;
```

```
   Mat cornerness_image; // image of the response to the Harris corner detector
   Mat corner_image;     // image showing the selected Harris interest points
   Mat image_grey;

   vector<KeyPoint> keypoints;

   cvtColor(src, image_grey, CV_BGR2GRAY);  // Harris operates on grey-scale images

   if (num_corners < 1)  num_corners    = 1;// can't define trackbar lower limit so enforce it here
   if (block_size < 2)    block_size    = 2; // can't define trackbar lower limit so enforce it here
   if (min_distance < 1)  min_distance  = 2; // can't define trackbar lower limit so enforce it here


   /* see  http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html?highlight=cornerharris#cornerharris */

   cornerHarris(image_grey,           // input
                cornerness_image,     // output
                block_size,           // blocksize, i.e. the size of region over which to compute the autocorrelation matrix
                3,                    // the neighbourhood over which the partial derivatives are computed (using the Sobel operator)
                0.04);                // the Trace multiplier used in the Harris forumula

   /* see  http://docs.opencv.org/2.4/modules/features2d/doc/common_interfaces_of_feature_detectors.html#goodfeaturestotrackdetector */

   GoodFeaturesToTrackDetector harris_detector(num_corners,                // maximum number of corners to detect
                                               0.01,                       // quality level
                                               min_distance,               // minimum distance between matched points
                                               block_size,                 // block size over which to compute the autocorrelation matrix
                                               true,                       // true to use Harris; false to use minimum eigenvalue
                                               0.04);                      // the Trace multiplier used in the Harris forumula

   harris_detector.detect(image_grey,keypoints);


   drawKeypoints(src, keypoints, corner_image, Scalar( 0, 0, 255 ) );

   Mat cornerness_display_image = convert_32bit_image_for_display(cornerness_image);

   imshow(feature_magnitude_window_name, cornerness_display_image);
   imshow(feature_location_window_name, corner_image);
}
```

# Reading

R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

Section 4.1 Points and Patches

Section 4.1.1 Feature Detectors