Feature Detection and Matching: Part 1

Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

Last updated: 2021-02-22

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Two Major Challenges in Computer Vision

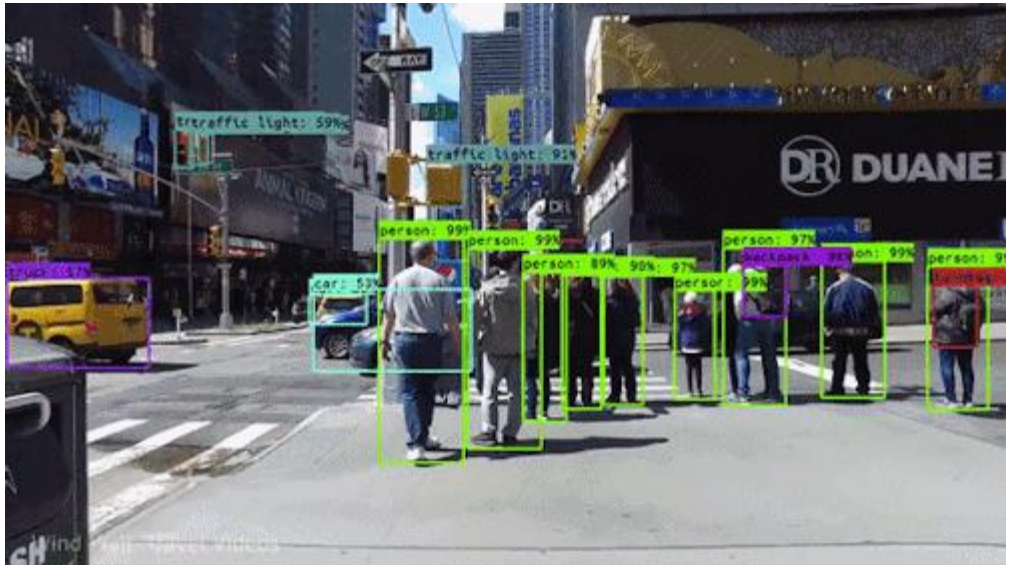
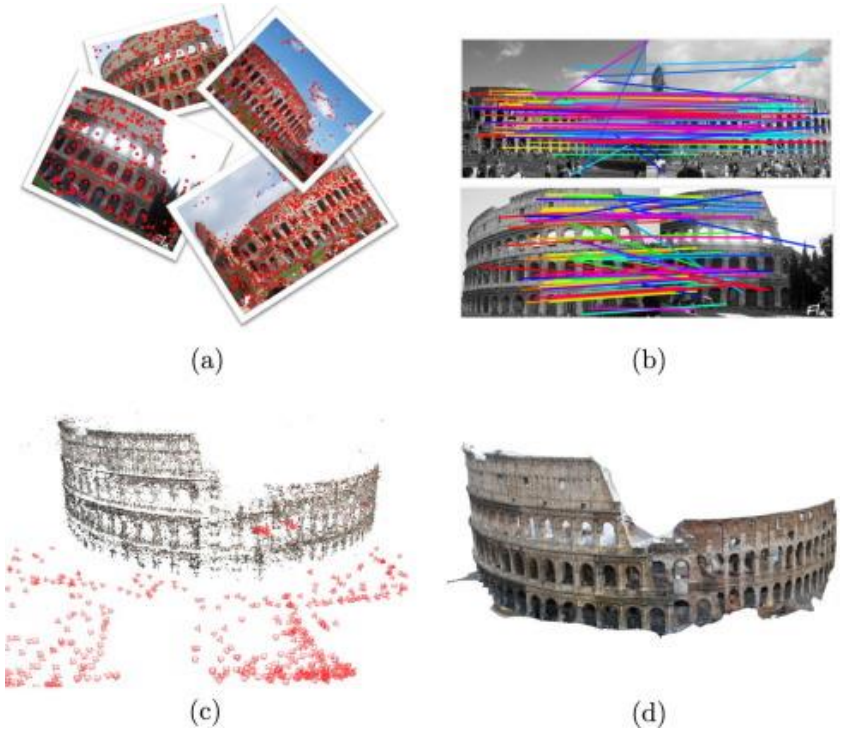


Image recognition
(interpretation)

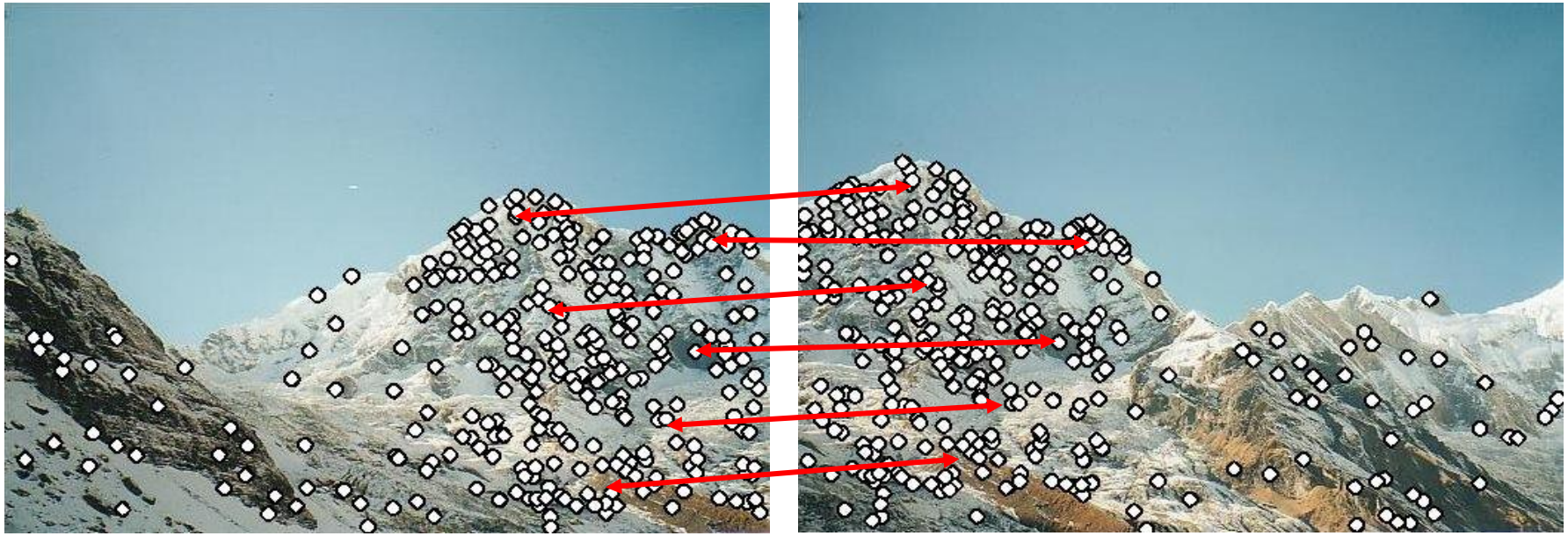


Feature matching
(3D reconstruction)

Why Do We Extract Features?

- We have two images
- How do we combine or stitch them together?

Why We Extract Features? Extracting Features (Continue)



What are Features (Keypoints)?

- A feature is a prominent point that is selected based on a certain criteria, such as edge, corner, or blob.
- This is represented in terms of the coordinates of the image points by pixel or sub-pixel.
- The feature likely contain and preserve the distinctive **local** regional information.
- Note: “interest points” = “keypoints”, also sometimes called “features”

Many applications:

- Object/motion tracking: which points are good to track?
- 3D scene reconstruction: find correspondences across different views
- Object recognition: find patches likely to tell us something about object category

Example: Automatic Panoramas

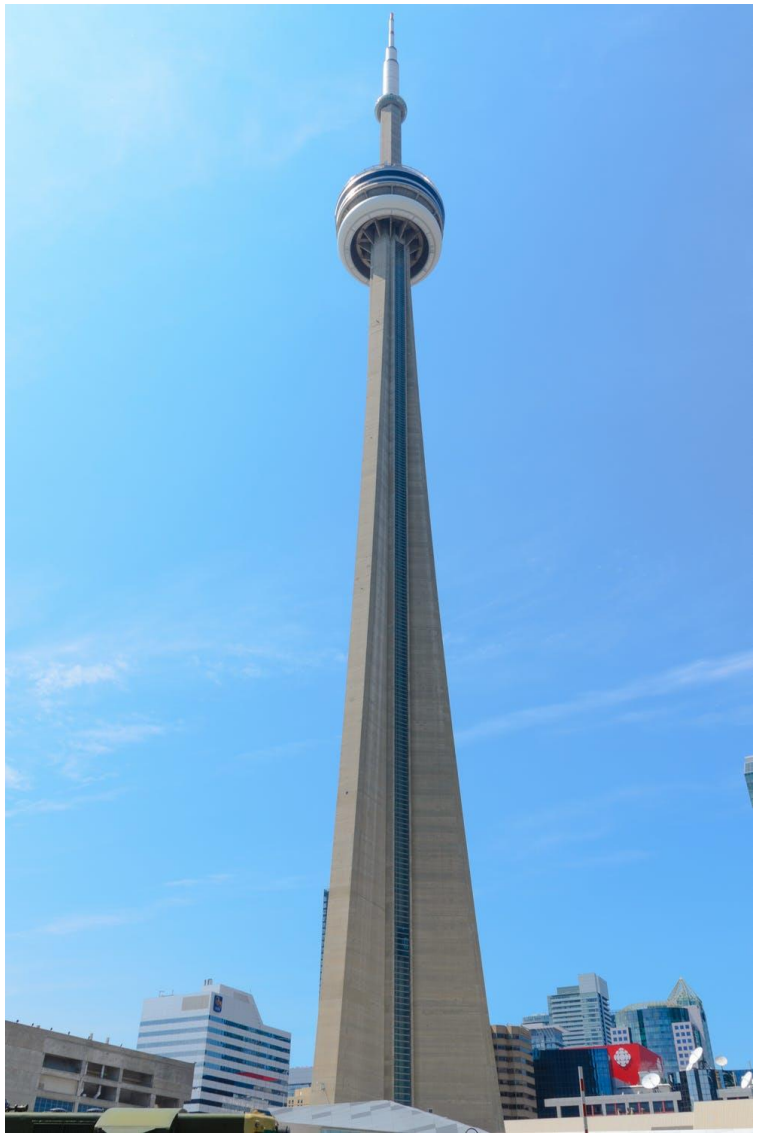
Structure from motion software (Pix 4D)

A large volume collected images from drone

Orthophoto(10,000 × 3,656) geometrically connected to each collected images

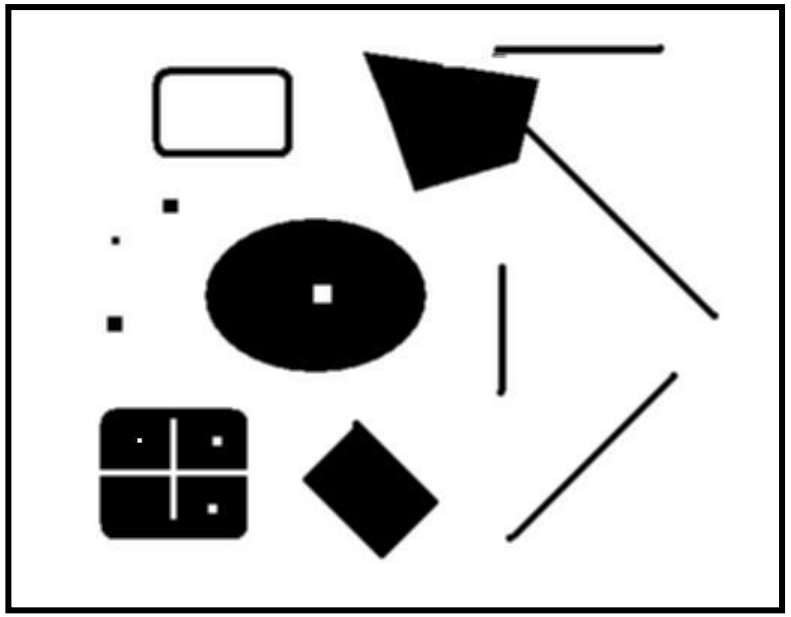
Jongseong Choi, Chul Min Yeum, Shirley J. Dyke, and Mohammad R. Jahanshahi, "Computer-Aided Approach for Rapid Post-Event Visual Evaluation of a Building Façade," Sensors, 18, 3017 (2018).

Image or Feature Matching Is a Challenging Task

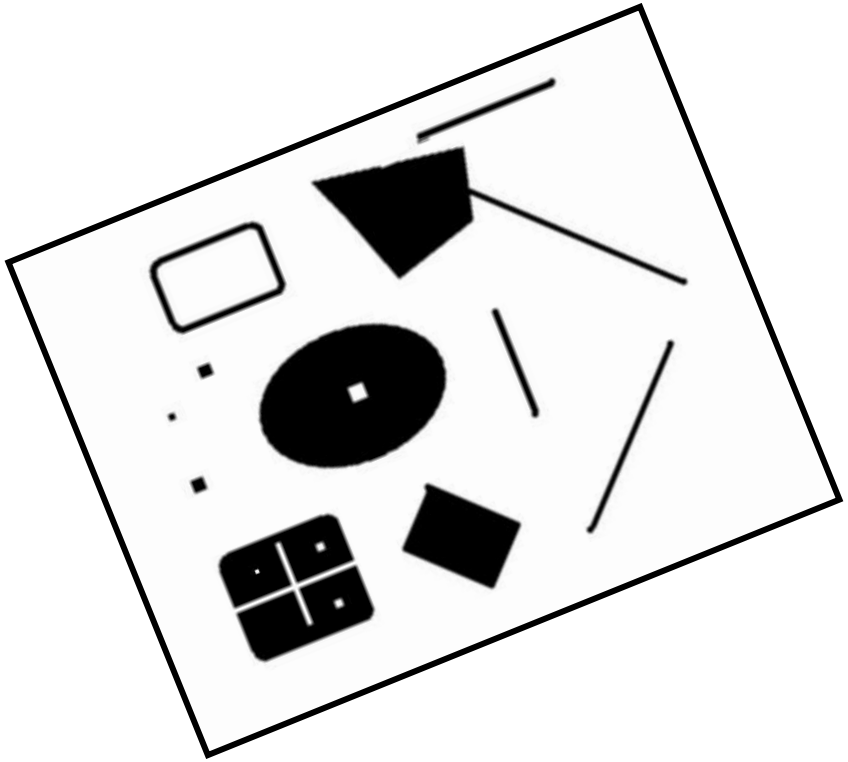


Example: Keypoints/Features

Q. Suppose you have to click on some point to find out how the image is deformed. Which points would you choose?

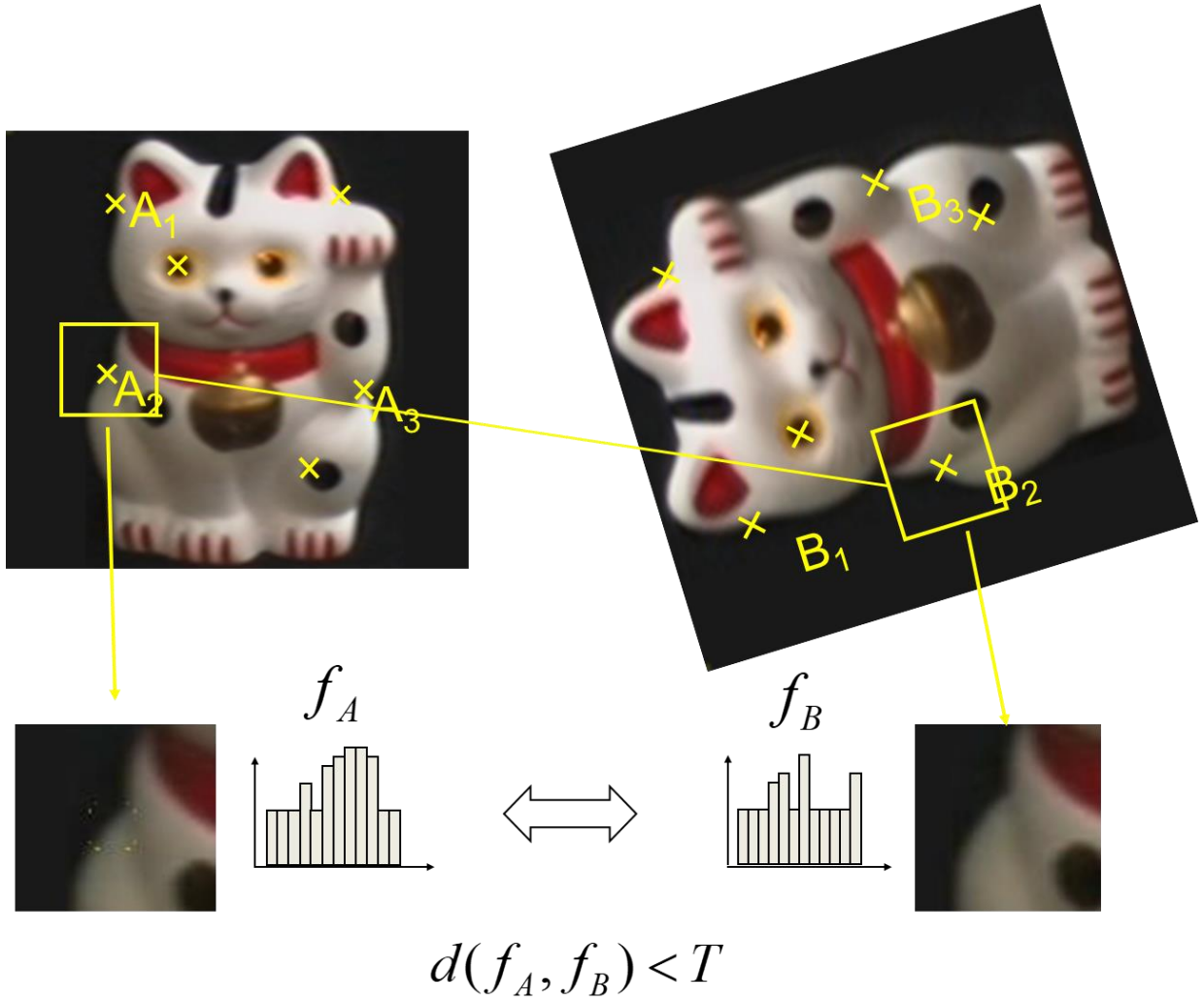


original



deformed

Overview of Feature Matching



Step 1. Find a set of distinctive key-points

Step 2. Define a region around each keypoint

Step 3. Extract and normalize the region content

Step 4. Compute a local descriptor from the normalized region

Step 5. Match local descriptors

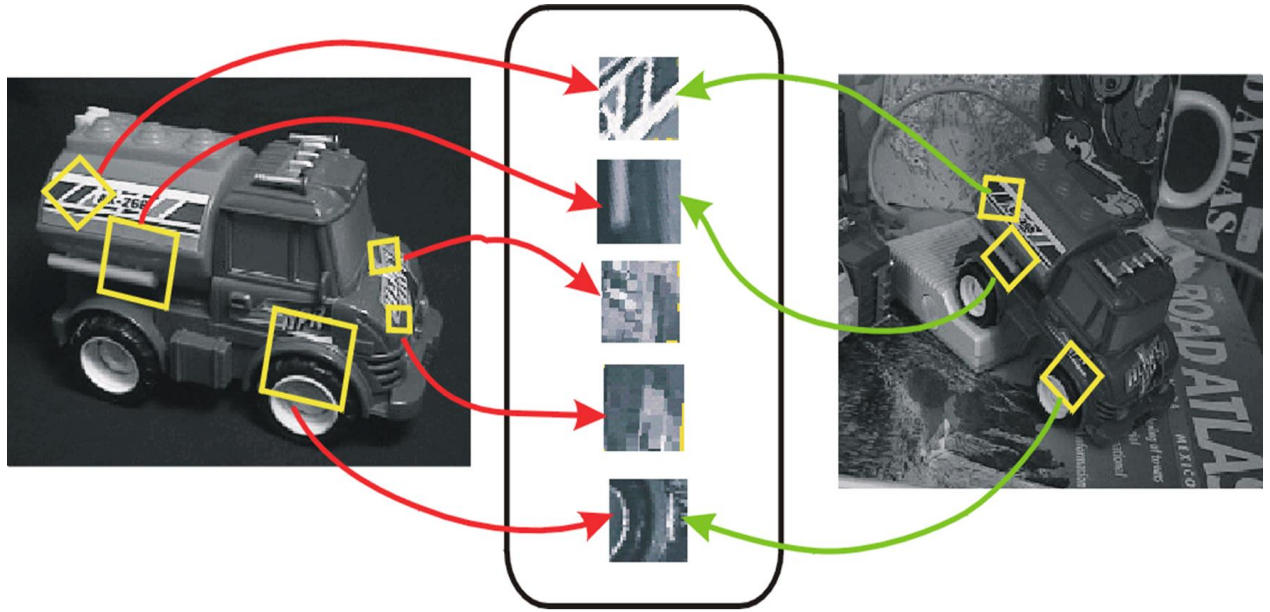
Characteristic of Good Features

Repeatability: The same feature can be found in several images despite geometric and photometric transformations

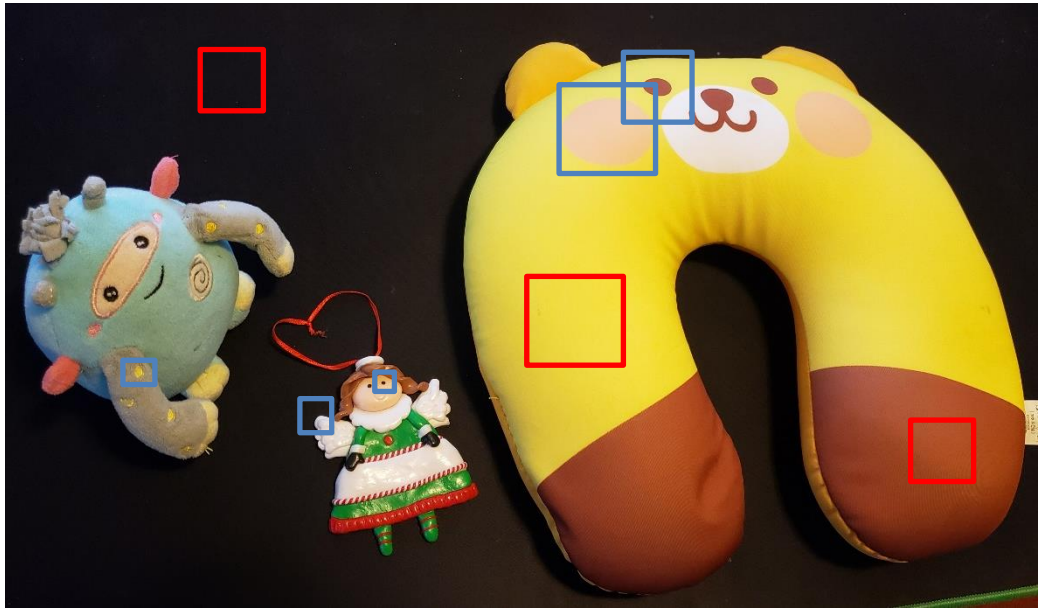
Saliency: Each feature is distinctive

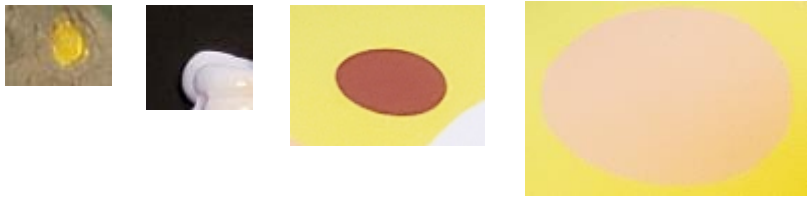
Compactness and efficiency: Many fewer features than image pixels

Locality: A feature occupies a relatively small area of the image; robust to clutter and occlusion



Example: Step 1. Find a Set of Distinctive Keypoints





Good keypoints

Not good keypoints

Step 1. Find a set of distinctive key-points

Step 2. Define a region around each keypoint

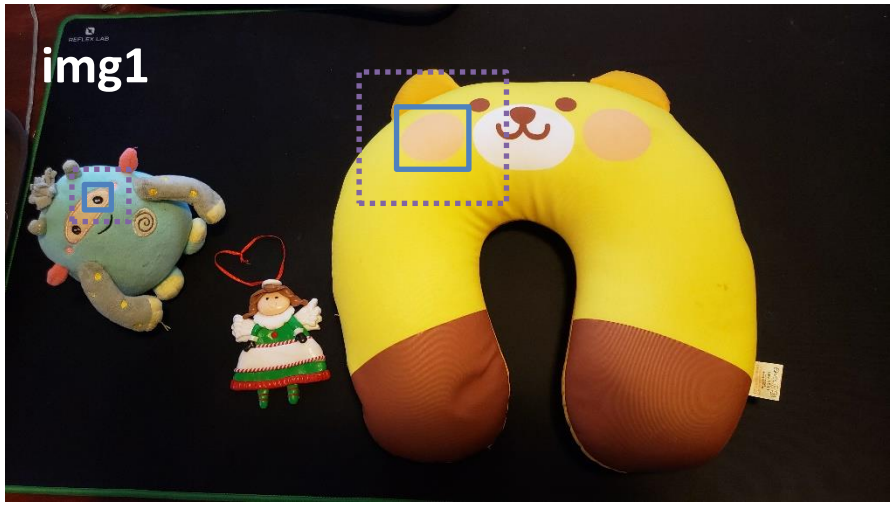
Step 3. Extract and normalize the region content

Step 4. Compute a local descriptor from the normalized region

Step 5. Match local descriptors

Q. How to match the extracted features?

Example: Step 2. Define a Region Around Each Keypoint



How to describe the features?

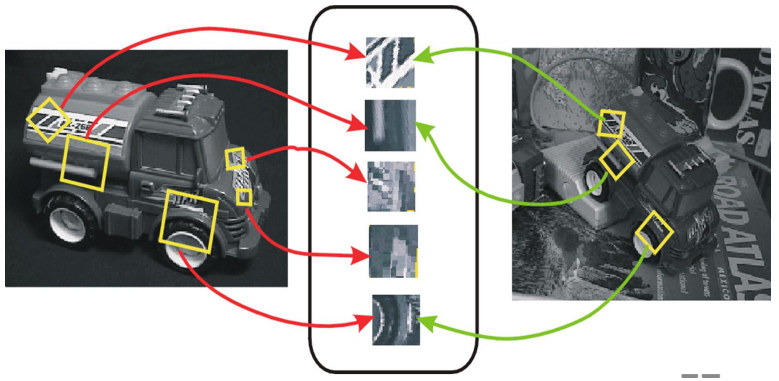
Step 1. Find a set of distinctive key-points

Step 2. Define a region around each keypoint

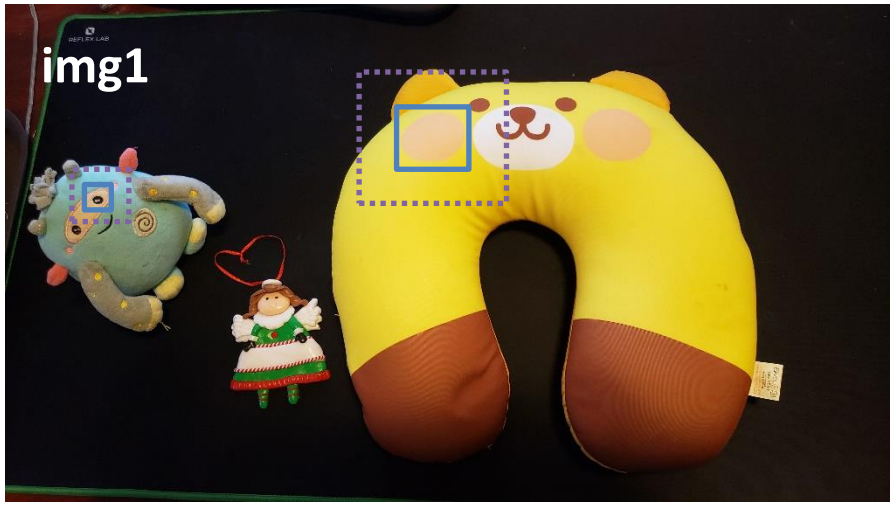
Step 3. Extract and normalize the region content

Step 4. Compute a local descriptor from the normalized region

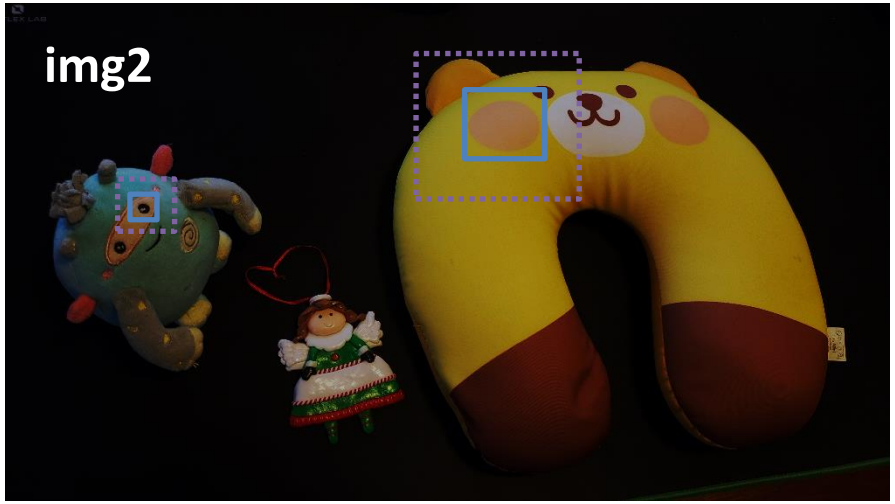
Step 5. Match local descriptors



Example: Step 3. Extract and Normalize the Region Content



How to make the "region" insensitive to color or light variations



Step 1. Find a set of distinctive key-points

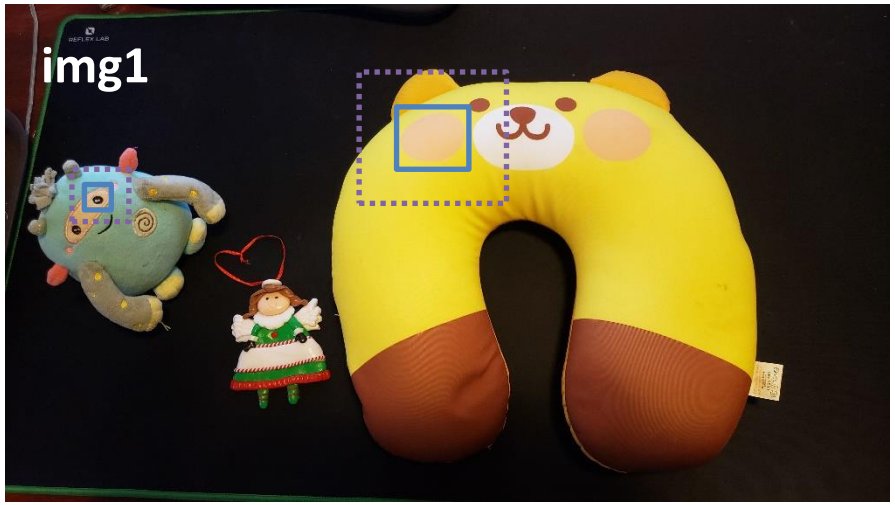
Step 2. Define a region around each keypoint

Step 3. Extract and normalize the region content

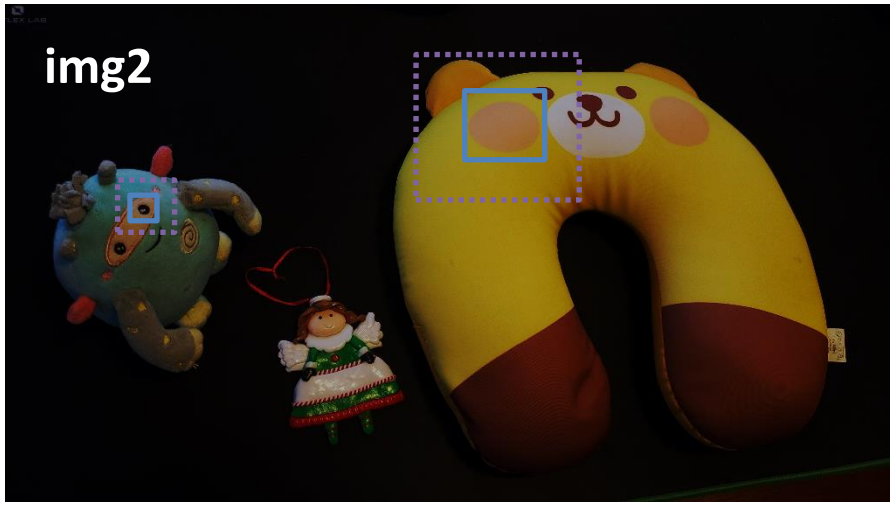
Step 4. Compute a local descriptor from the normalized region

Step 5. Match local descriptors

Example: Step 4. Compute a Local Descriptor from the Normalized Region



How to represent the region using a fixed size vector



10
8
⋮
5
6

4
2
⋮
9
10

10
8
⋮
4
6

4
2
⋮
9
11

Step 1. Find a set of distinctive key-points

Step 2. Define a region around each keypoint

Step 3. Extract and normalize the region content

Step 4. Compute a local descriptor from the normalized region

Step 5. Match local descriptors

The descriptor is a fingerprint that can uniquely represent the features.

Example: Step 5. Match Local Descriptors

img1

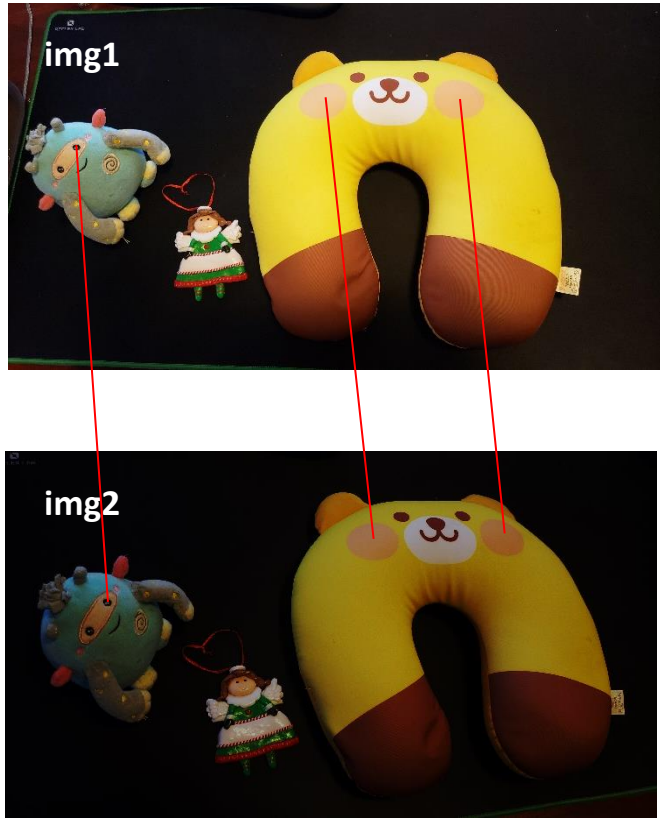
10
8
⋮
5
6

feature 1

...

4
2
⋮
9
10

feature N



Matching the descriptors

img2

3
4
⋮
1
0

feature 1

...

5
3
⋮
11
12

feature M

Step 1. Find a set of distinctive key-points

Step 2. Define a region around each keypoint

Step 3. Extract and normalize the region content

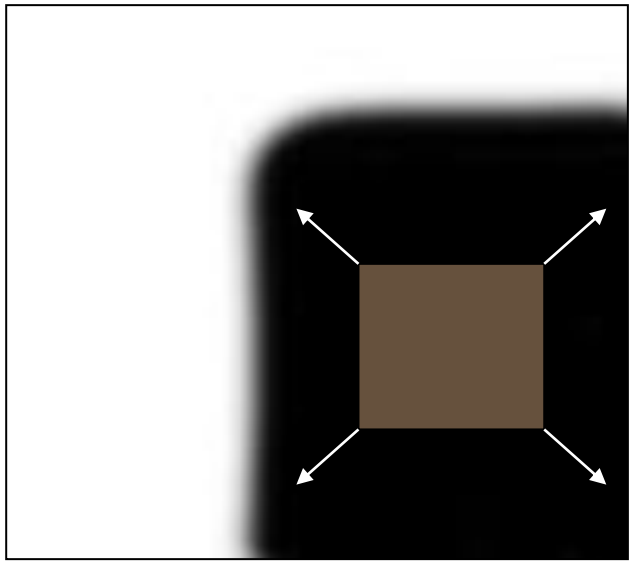
Step 4. Compute a local descriptor from the normalized region

Step 5. Match local descriptors

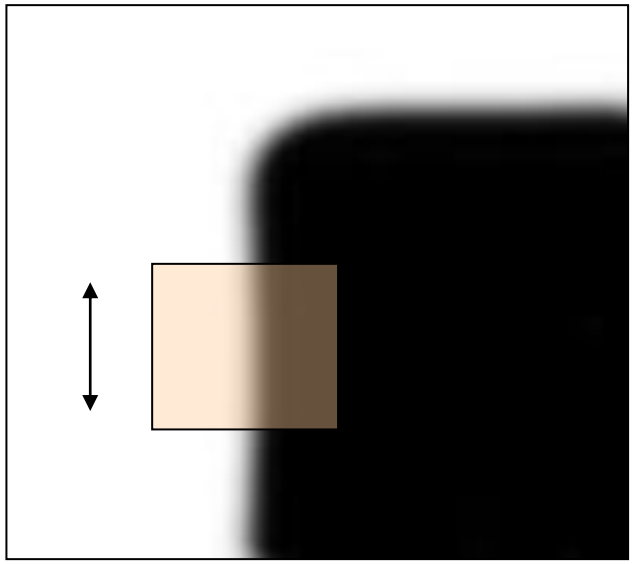
What Locations Would You Choose as Features ?

Corner Detection: Basic Idea

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity

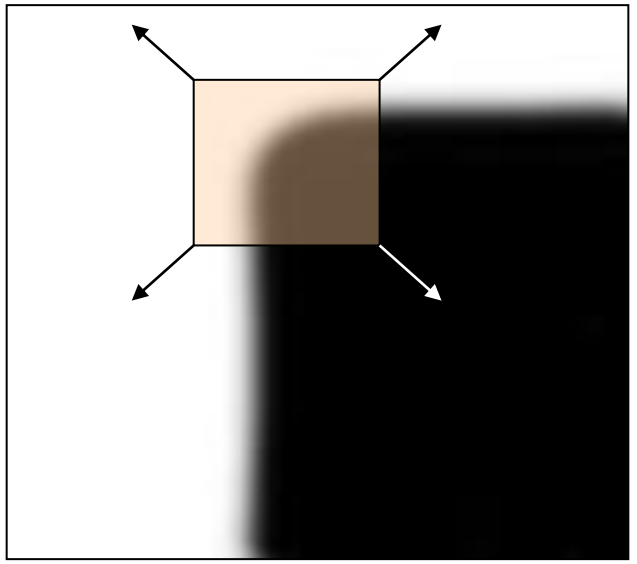


Flat region
no change in all directions



Edge
no change along the edge direction

Distinctive but not locality



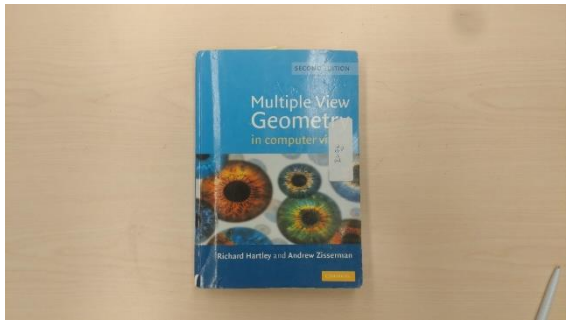
Corner
significant change in all directions

Distinctive and locality

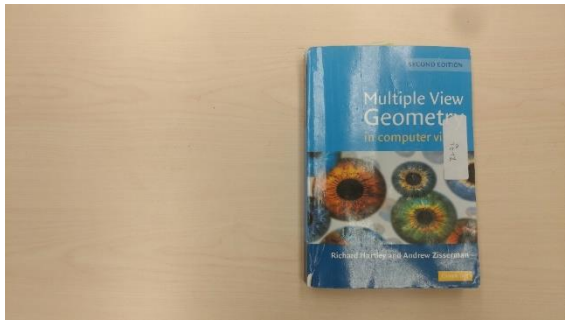
Invariance and Covariance of Features

Invariance: images are transformed, and feature locations do not change

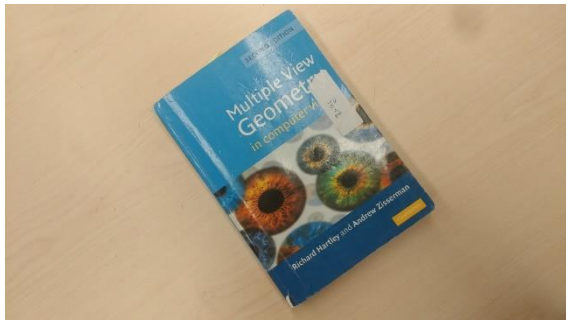
Covariance: if we have two transformed versions of the same image, features should be detected in corresponding locations



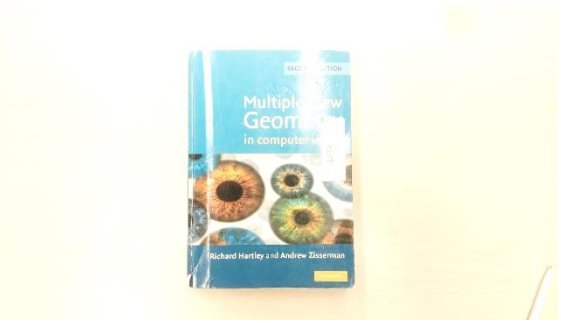
Original



Translation



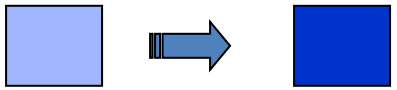
Rotation



Intensity

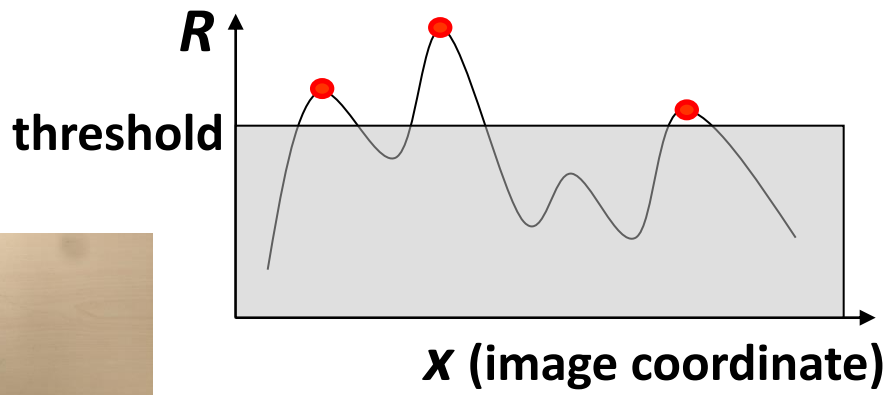
Scale

Intensity Change

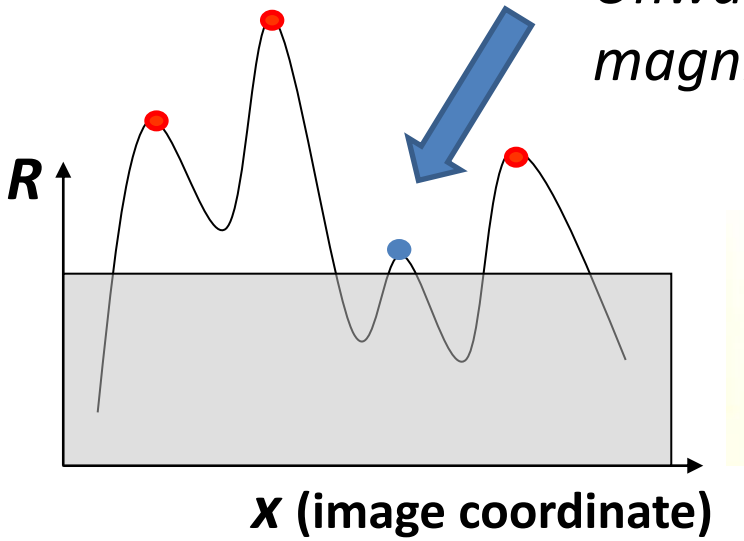


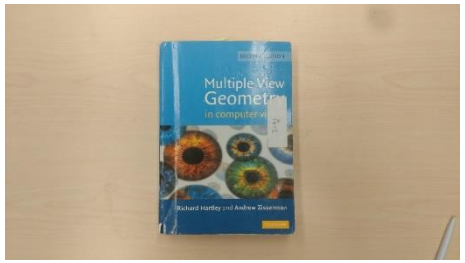
$I \rightarrow a\,I + b$

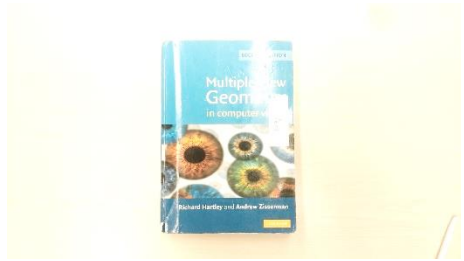
- Only derivatives are used for edge or corner detection => invariance to intensity shift $I \rightarrow I + b$
- Intensity scaling: $I \rightarrow a\,I$



Unwanted magnification

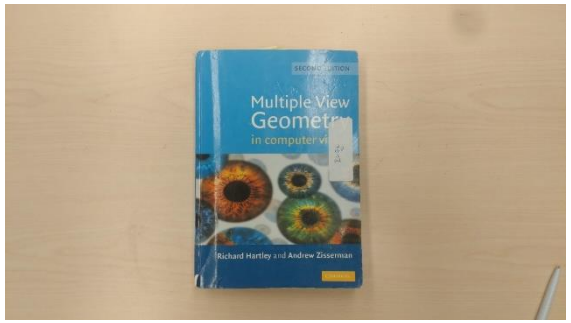


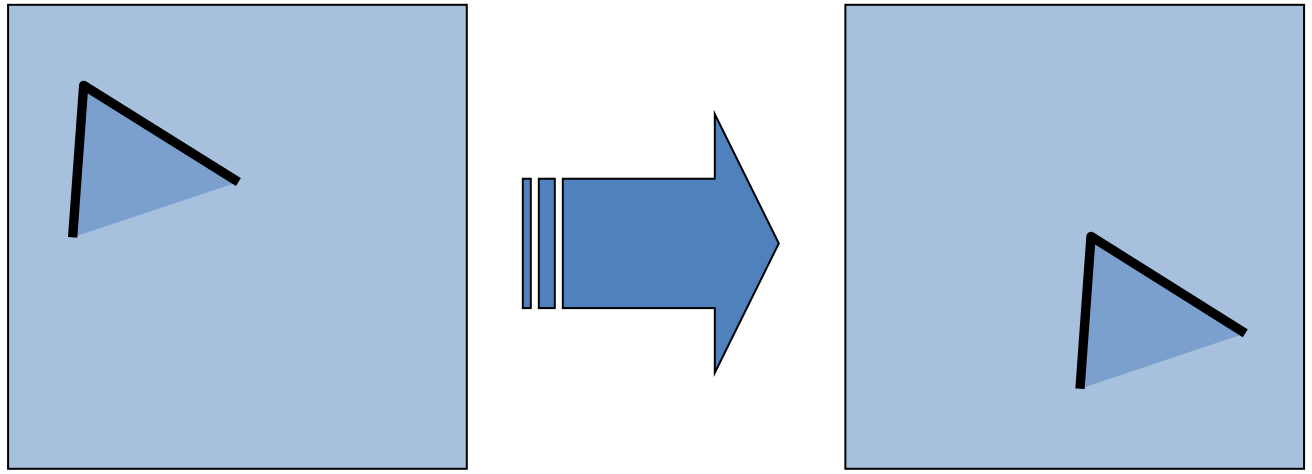


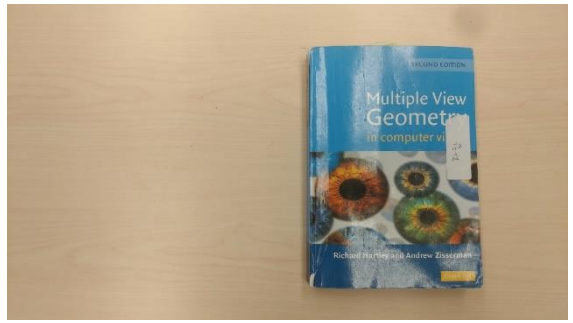


***Partially invariant* to intensity change**

Image Translation



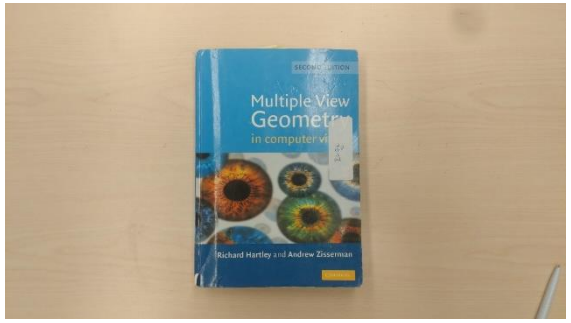


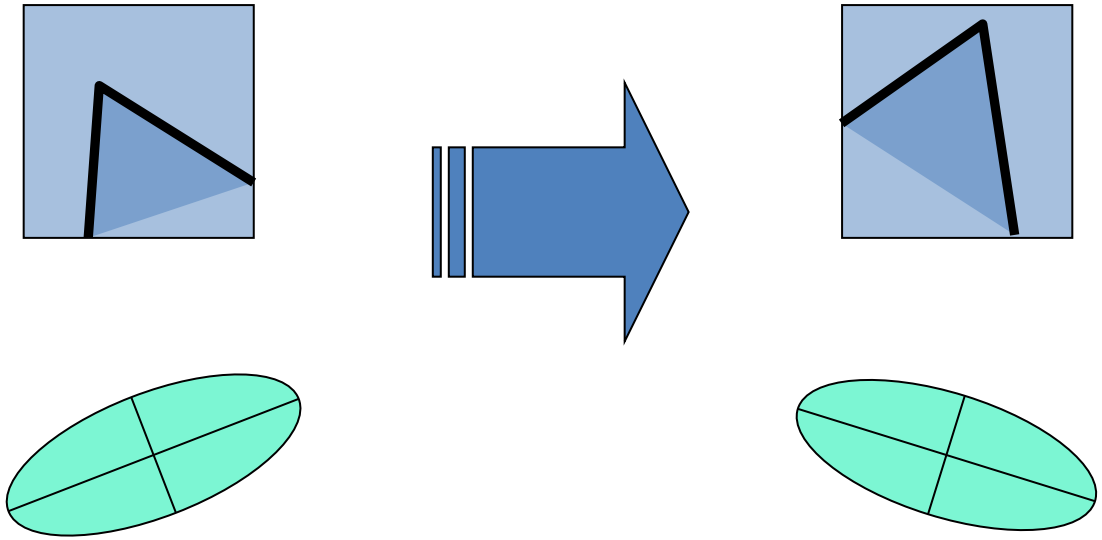


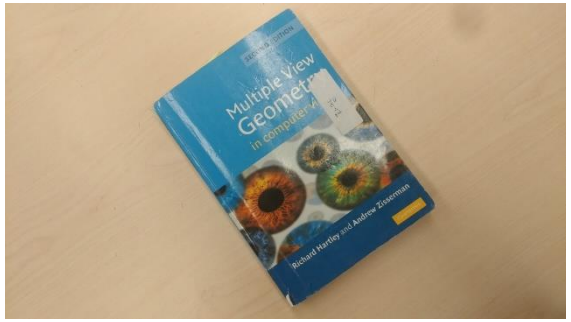
- **Derivatives and window function are shift-invariant**

Corner locations are covariant to image translation

Image Rotation



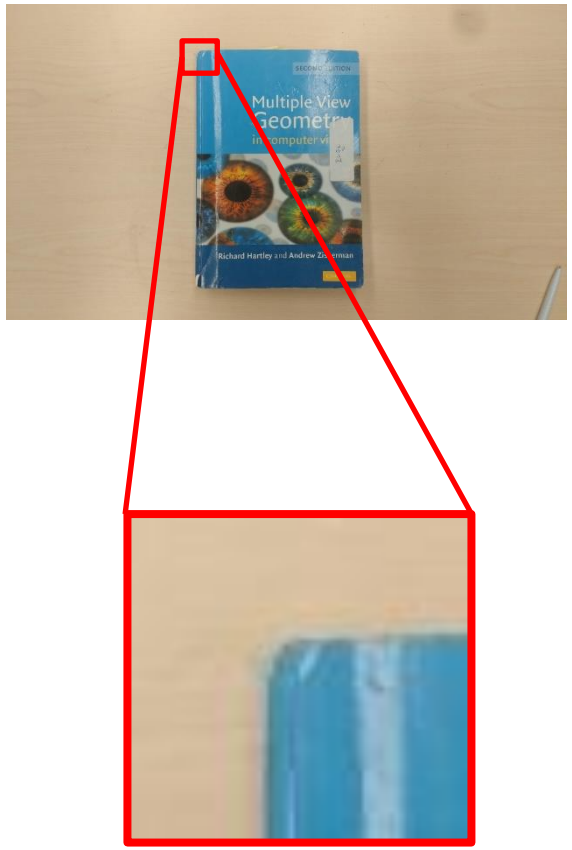


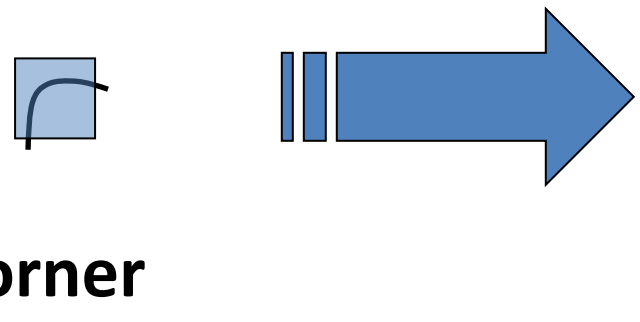


Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

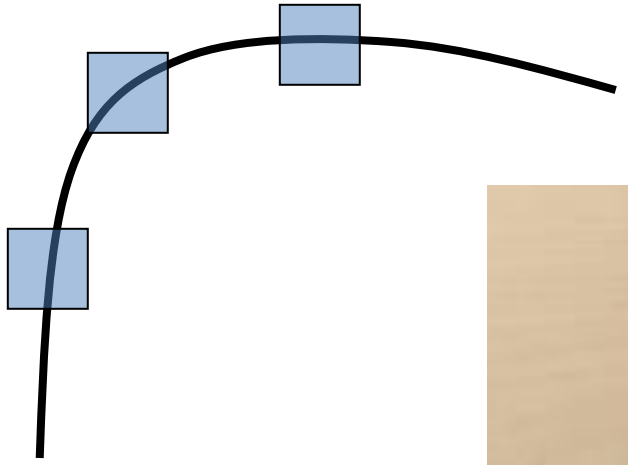
Corner locations are covariant to image rotation

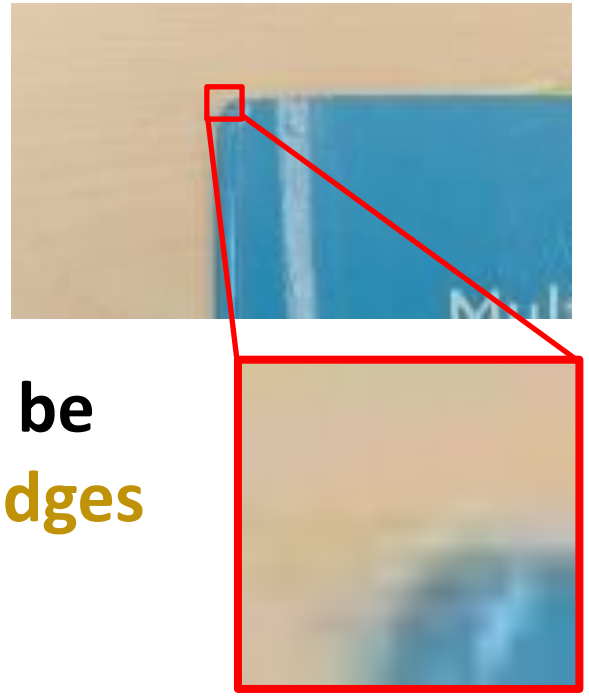
Scaling





Corner



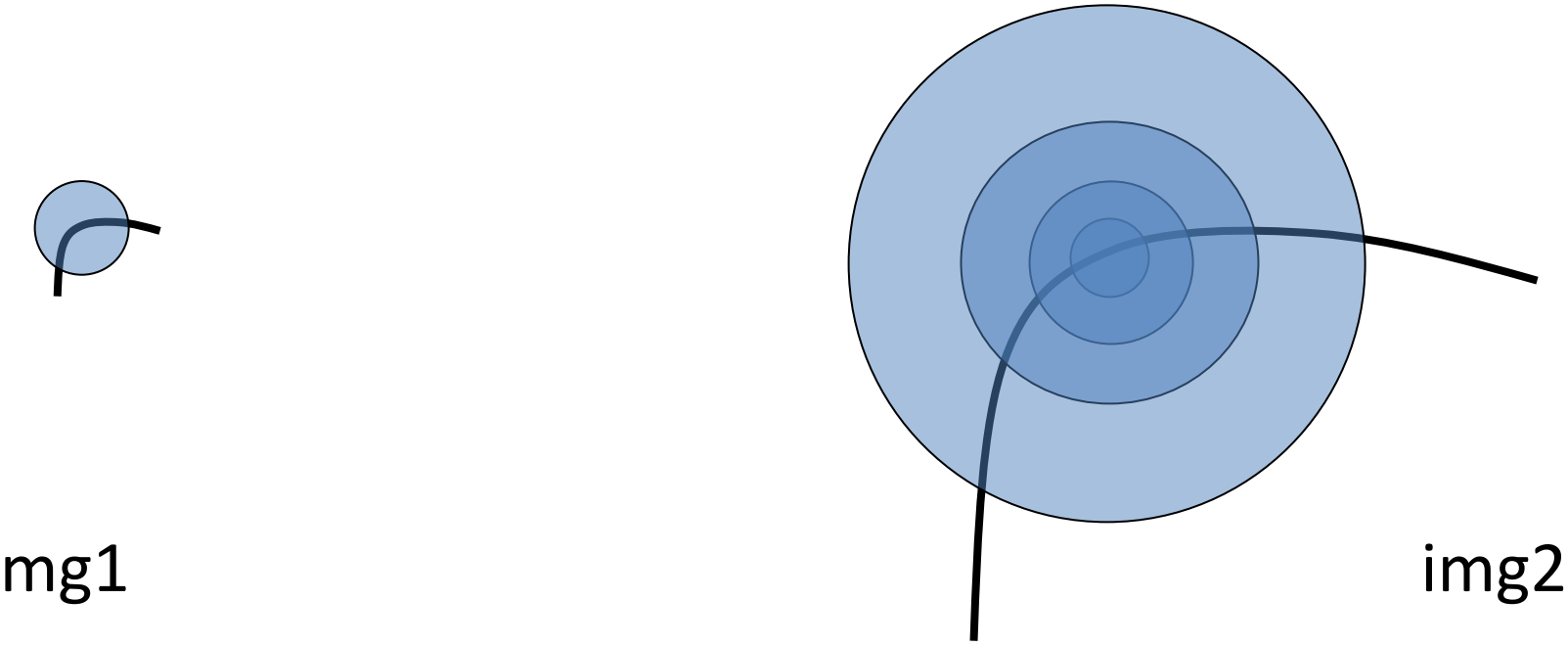


All points will be classified as edges

Corner location is **NOT** invariant to image scale!

Scale Invariant Detection

- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will be the same in both images

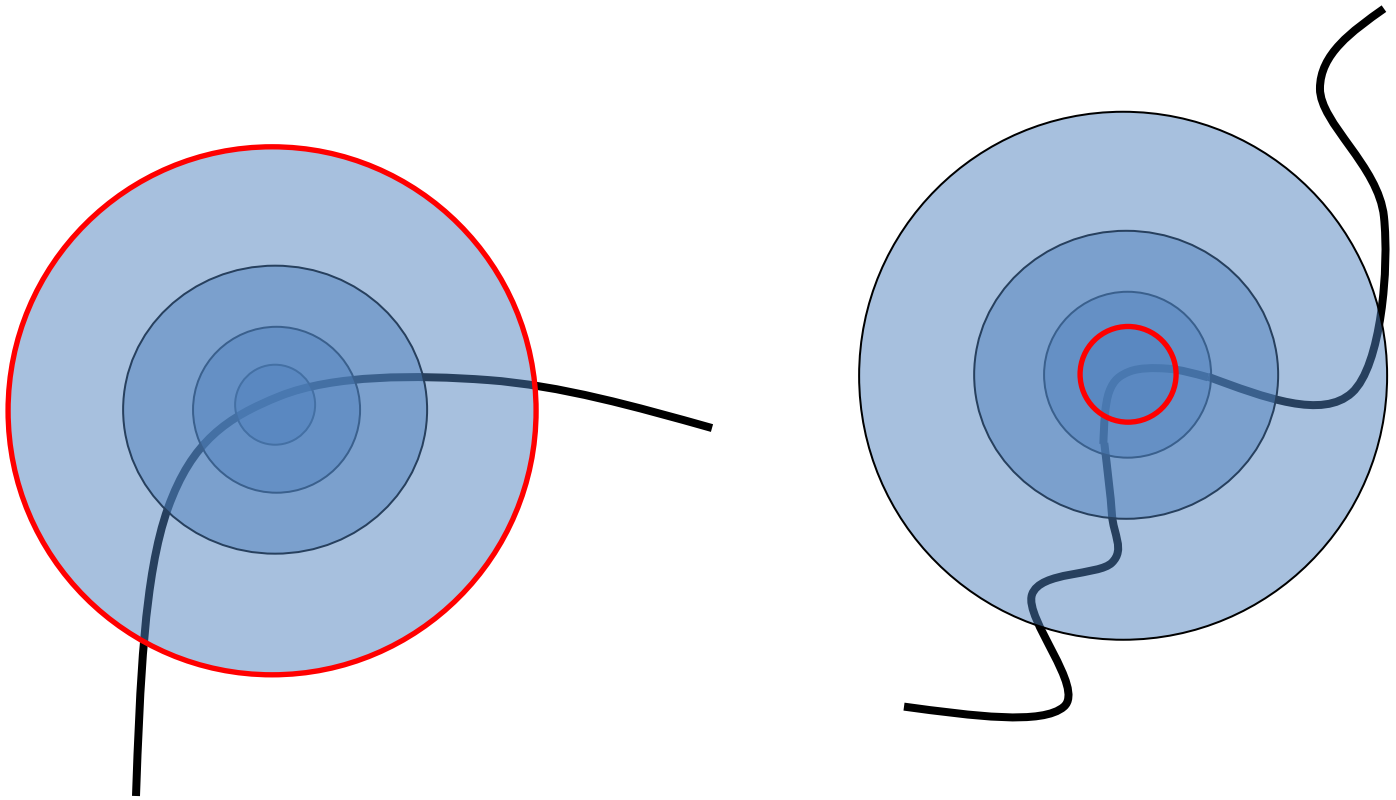


img1

img2

Scale Invariant Detection (Continue)

- To detect the corners in both image, we need to know how to choose corresponding circles ***independently*** in each image?
- Choose the scale of the “best” corner



Example: Scale Invariance

Slide Credits and References

- Lecture notes: Gordon Wetzstein
- Lecture notes: Mohammad Jahanshahi
- Lecture notes: Noah Snavely
- Lecture notes: L. Fei-Fei
- Lecture notes: D. Frosyth
- Lecture notes: James Hayes
- Lecture notes: Yacov Hel-Or
- Lecture notes: K. Grauman, B. Leibe