GEOS 639 – INSAR AND ITS APPLICATIONS

GEODETIC IMAGING AND ITS APPLICATIONS IN THE GEOSCIENCES

Lecturer:

Franz J Meyer, Geophysical Institute, University of Alaska Fairbanks, Fairbanks; fjmeyer@alaska.edu

Lecture 5: Stereo Photogrammetry – Processing Details and Transition to Structure-from-Motion

Stereo-Photogrammetry: Image Matching Techniques

Part 1: Normalized Cross Correlation

Feature Extraction and Correspondence Matching

- When generating DEMs from stereo pairs, we are applying **automatic procedures** during the **Relative Orientation** and the final **3-D Coordinate Estimation steps**:
  - Identification of clearly recognizable features with clearly defined position → **interest point detection**
  - Finding the corresponding feature in image 2 based on a feature found in image 1 → **feature matching**
- **Today, we will discuss traditional and advanced techniques for these tasks**
- **Selected Methods:**
  - **Correlation**
  - Moravec Operator
  - Harris Operator
  - **SIFT**
  - SURF

We will focus on these (one traditional and one advanced technique)

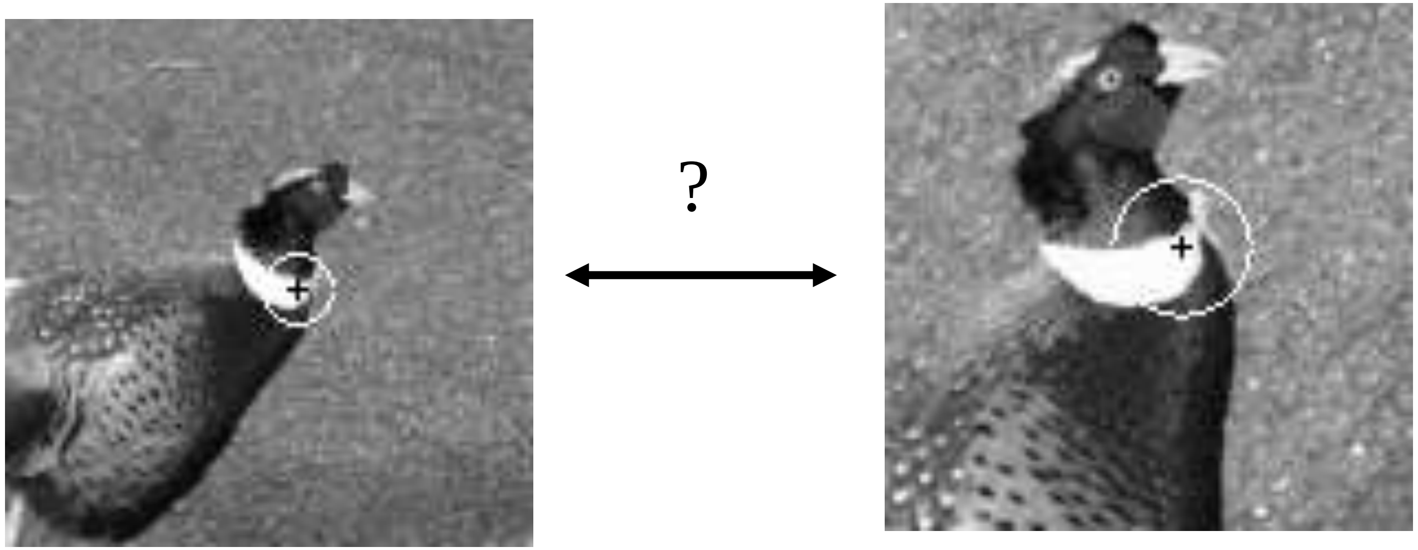
- **In the next lectures, we will discuss one more method that will increase the robustness of image matching (RANSAC)**

Relevant Work on Interest Point Detection and Image Matching

- **Corner-based local interest points**
  - Moravec (1981), Harris (1992)
- **Descriptors**
  - **Correlation window around each corner**
    - Zhang (1995)
  - **Local, rotationally invariant**
    - Schmid & Mohr (1997)
  - **Scale-invariance:**
    - Crowley & Parker (1984),
    - Shokoufandeh et. al. (1999),
    - Lindeberg (1993,1994),
    - Mikolakczyk & Schmid (2002).
- **Maximally-Stable Extremal Regions (MSER)**
  - Matas (2002)
- **Scale Invariant Feature Transform (SIFT)**
  - Lowe (1999, 2004)
- **Speeded Up Robust Features (SURF)**
  - Bay,Tuytelaars, Van Gool (2006)

The Feature Matching Problem

- Given we identified an area of interest in an image, what information should we use for finding (matching) corresponding regions in different images?



Feature Matching Methods

Simplest Approach: Cross Correlation

- Directly compare intensities using “sum of squared differences” or “normalized cross-correlation”

Feature Matching Methods

Simplest Approach: Normalized Cross Correlation

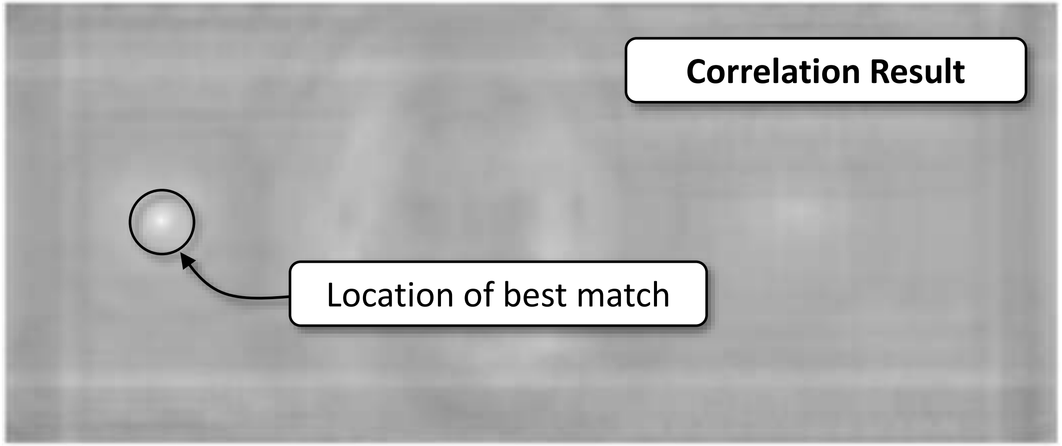
- Correlation is oldest technique for finding correspondence between image pixels
- Normalized Cross Correlation is a pattern matching techniques that uses an image template to find corresponding gray value patterns in a second image (see example below)

- Assumptions:
  - **Scene point must have same intensity in each image**
  - **Cross-correlation is not rotation invariant – so images should not be rotated relative to each other**

Cross-Correlation matching example:

- Find Federal Reserve stamp on $100 bill

Template for correlation matching



Feature Matching Methods

Simplest Approach: Normalized Cross Correlation

- **Input information needed for cross-correlation:**
  - Pair of input images (u_1 and u_2)
  - Define window of size W to be used for cross correlation analysis (how large do you want your search template to be?)
  - Define a search region $R(P_1)$ in image 2 associated with a pixel P_1 in image 1

- For each image location at distance $d = [d_i \quad d_k]^T$ from pixel P_1, **calculate correlation coefficient**

$$cc(d) = \frac{\left|\sum_W u_1[i,k] \cdot u_2[i,k]\right|}{\sqrt{\sum_W |u_1[i,k]|^2 \cdot \sum_W |u_2[i,k]|^2}}$$

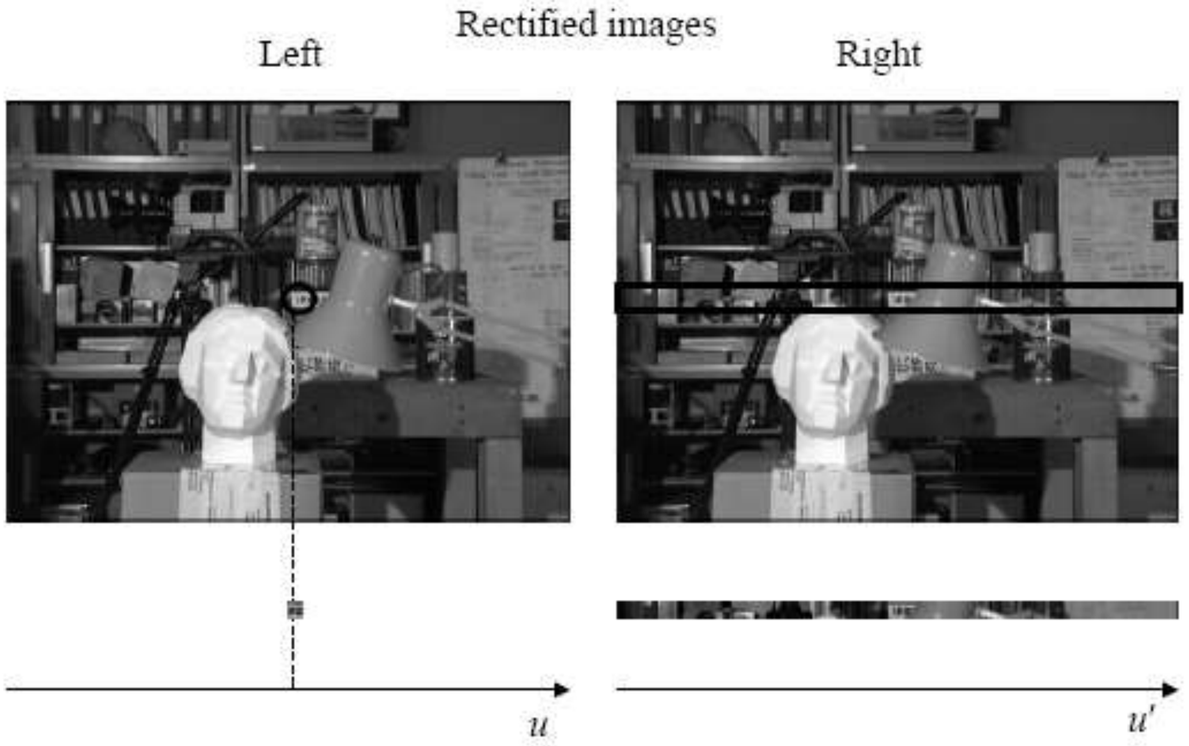
- **The disparity** for P_r is the vector $\bar{d} = [\bar{d}_i \quad \bar{d}_k]^T$ that maximizes $cc(d)$ within $R(P_1)$

$$\bar{d} = argmax_{d \in R(P_1)}\{cc(d)\}$$

Feature Matching Methods

Simplest Approach: Cross Correlation

- Works satisfactorily when we matching corresponding regions related mostly by translation.
  - e.g., stereo pairs, video sequence assuming small camera motion



Feature Matching Methods

Simplest Approach: Normalized Cross Correlation

- **Problems – Part 1:**
  - **What if image contrast and image noise levels differ between images**
    - → Different maximum and minimum intensities
    - → **Simple intensity matching degrades quickly**
  - **What if Images were acquired from different directions and scatter differently in these directions?**
    - → Scene objects are not perfect Lambertian scatterers
    - → **Matching quality reduced**

- **Solution to Problems – Part 1:**
  - **Do not use image intensity values but use intensity gradients instead!**
  - **Possible approach:**
    - Compute the gradient magnitude at each pixel in the two images without smoothing
    - Map the gradient magnitude values into three values: -1, 0, 1 (by thresholding the gradient magnitude)
    - → **Amplitude independent and more sensitive correlations**

Feature Matching Methods

Simplest Approach: Cross Correlation

- **Problems – Part 2:**
  - **Cross-correlation does not allow for the following variations between images:**
    - Relative image rotations
    - Variation of pose (due to different image locations)
    - Variations in image scale
  - **Matching success strongly depends on image structure relative to search window size.**
    - too small a window → may not capture enough image structure and → many false matches
    - too large a window → matching less sensitive to noise (desired) but also decreases precision (blurs disparity map)
- Hence, we will discuss a more powerful descriptor called **SIFT**

Stereo-Photogrammetry: Image Matching Techniques

Part 2: Scale Invariant Feature Transform (SIFT)

SIFT – A Powerful Feature Extraction and Matching Method

- **SIFT** = **S**cale **I**nvariant **F**eature **T**ransform
- **Properties**:
  - Extracts & describes large amount of features to recognize objects in images
  - Robust identification of objects even among clutter and under partial occlusion,
  - **Invariant** to uniform scaling, changing orientation, and partially invariant to affine distortion and illumination changes.
- **An Image matching approach based on SIFT includes the following steps:**
  1. **Scale-space feature extraction and formulating a descriptor for these features**
     - Extract scale and rotation invariant interest points (i.e., keypoints).
  2. **Matching features in one image to the same features in other images**
     - Find features in other images.
  3. **Combine features to identify the location of objects**
     - This is done using an algorithm called the Hough transform.
  4. **Model verification and outlier removal**
     - Use least squares methods to verify solutions and remove outliers.

Almost all of SIFT is hidden here!

D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", **International Journal of Computer Vision**, 60(2): 91-110, 2004. **Cited 45,696 times (3/2018) – 55,920 times (3/2020)**

Think – Pair – Share

Feature identification and matching

Q1: In Stereo photogrammetry we use automatic methods such as **cross-correlation** to detect and match-up interest points in the two images of a stereo pair.

What is the goal of automatically finding and matching many points in the two images? What is the product we are trying to create?

Q2: Cross-correlation has limitations for feature detection and matching.

What are the main limitations of cross-correlation and for which stereo constellations (1) do we and (2) do we not care about these limitations

SIFT – A Powerful Feature Extraction and Matching Method

- **Let's reiterate the main problems in feature matching:**
  - Find most robust features that describe a scene the best and can be easily recognized in other images
  - Formulate a descriptor that can robustly describe what your feature looks like
  - Make sure your descriptors are robust even if (1) image scale, (2) image orientation, (3) image illumination, and (4) image noise are changing

- **Hence: SIFT performance goal check list**
  - Scale Invariance
  - Rotation Invariance
  - Illumination invariance
  - Viewpoint invariance

The SIFT Algorithm

Overview of the Approach

1. Create Scale Space (multi-scale Image Pyramid)
by applying larger and larger Gaussian blurring filters to the original image

2. Create "Differences of Gaussian" Image Pyramid
by taking difference between consecutive pyramid images

3. Find Local Extrema in the "Difference of Gaussian" Scale Space

4. Choose Candidate Keypoints from the Identified Extrema

5. Calculate and Assign Keypoint Orientation

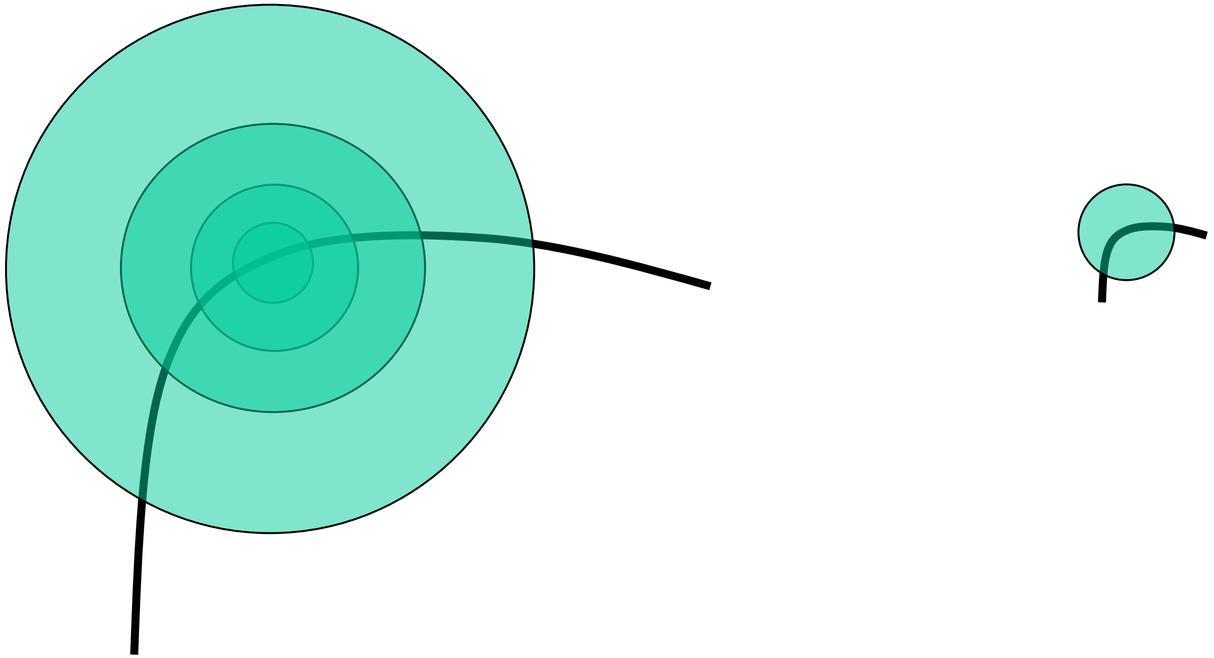
6. Build Keypoint Descriptor
by calculating gradient directions within a 16x16 window around keypoint

Match Keypoints found in different images

SIFT – Scale Invariant Feature Detection

Step 1: Construct Scale Space

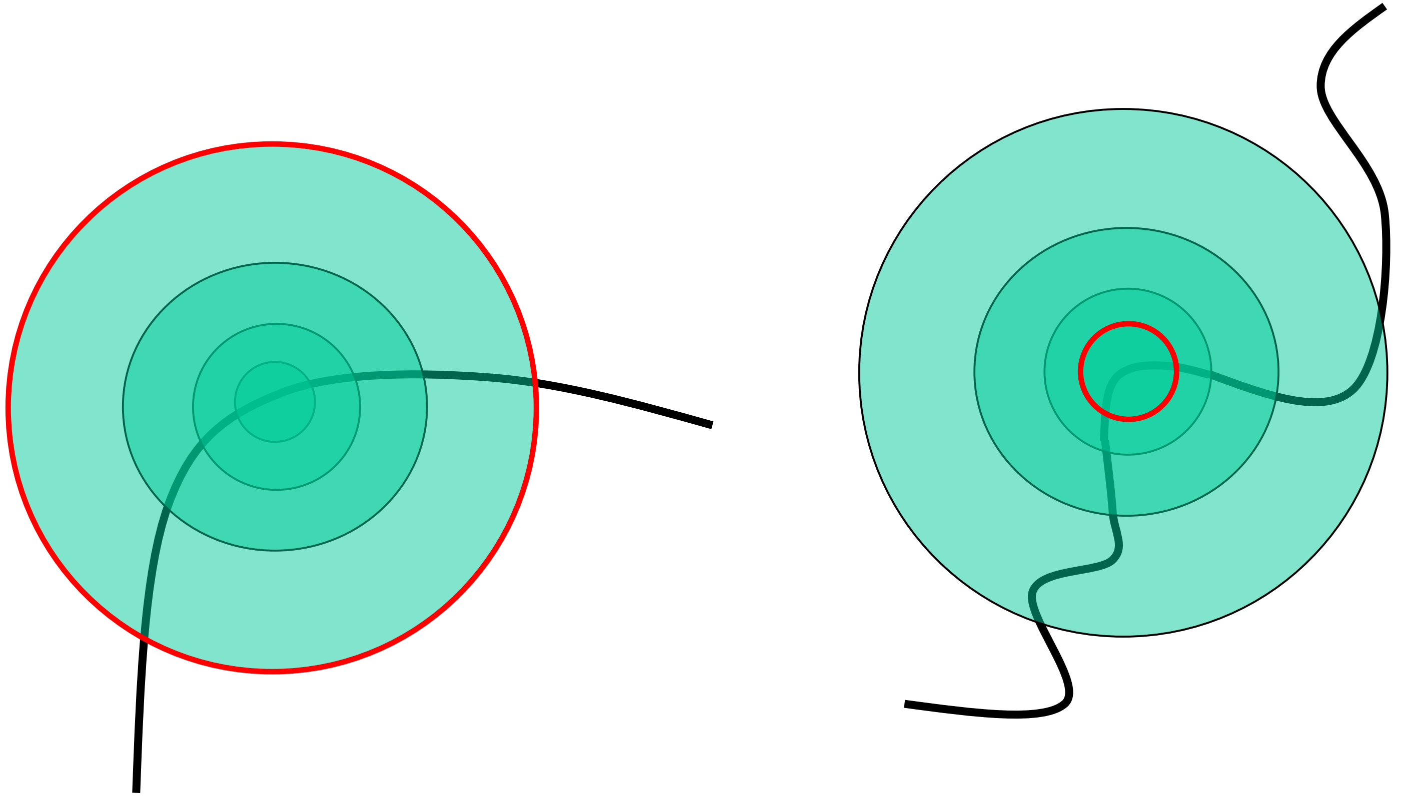
- **Why use various scales?**
  - Consider regions (e.g. circles) of different sizes around a point
  - Regions of corresponding **relative** sizes will look the same in both images



SIFT – Scale Invariant Feature Detection

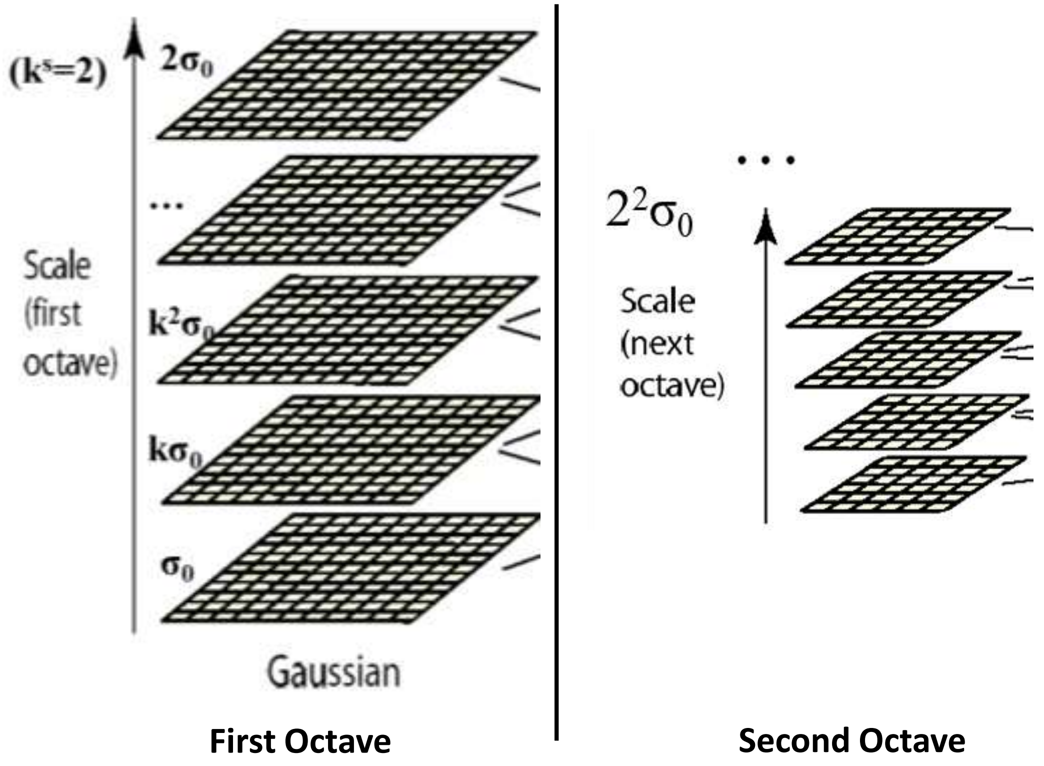
Step 1: Construct Scale Space

- **Why use various scales?**
  - Consider regions (e.g. circles) of different sizes around a point
  - Regions of corresponding **relative** sizes will look the same in both images
  - **Problem**: How to choose corresponding circle sizes **automatically and independently** in each image?



SIFT – Scale Invariant Feature Detection
Step 1: Construct Scale Space

- **Solution**: We construct multiple resolution scales for each image using Gaussian Scale Space Processing



Gaussian Scale Space Concept

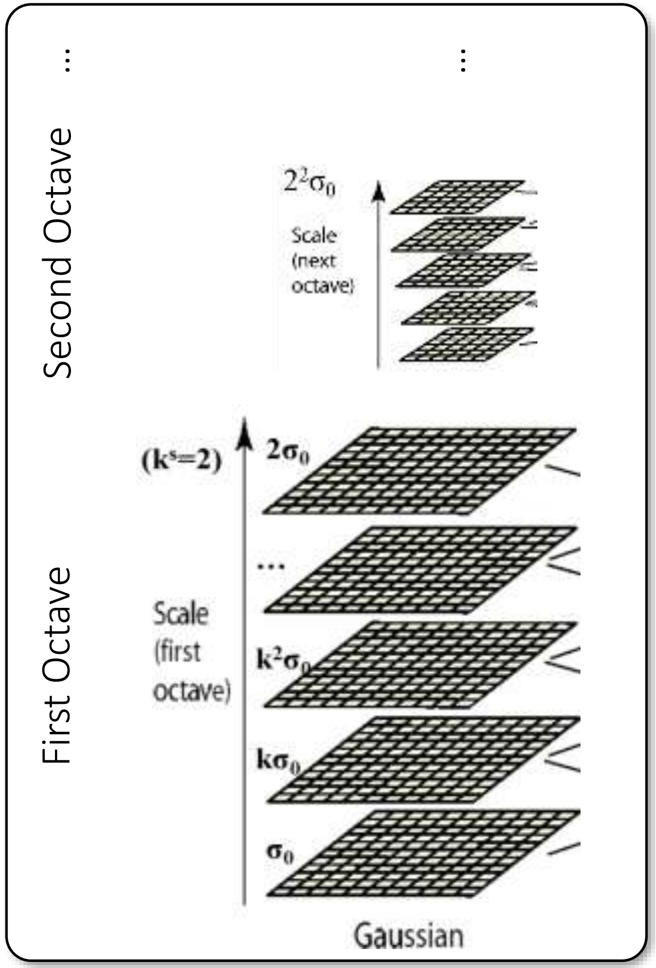
- **Gaussian Scale Space:**
  - **Created by convolving image u with Gaussian smoothing kernels of progressively larger standard deviation σ**
  - **An image pyramid is created**,
    - each pyramid level is called octave o and contains a total of s sublevels
    - Each sublevel is separated by constant factor k such that the standard deviation scales as

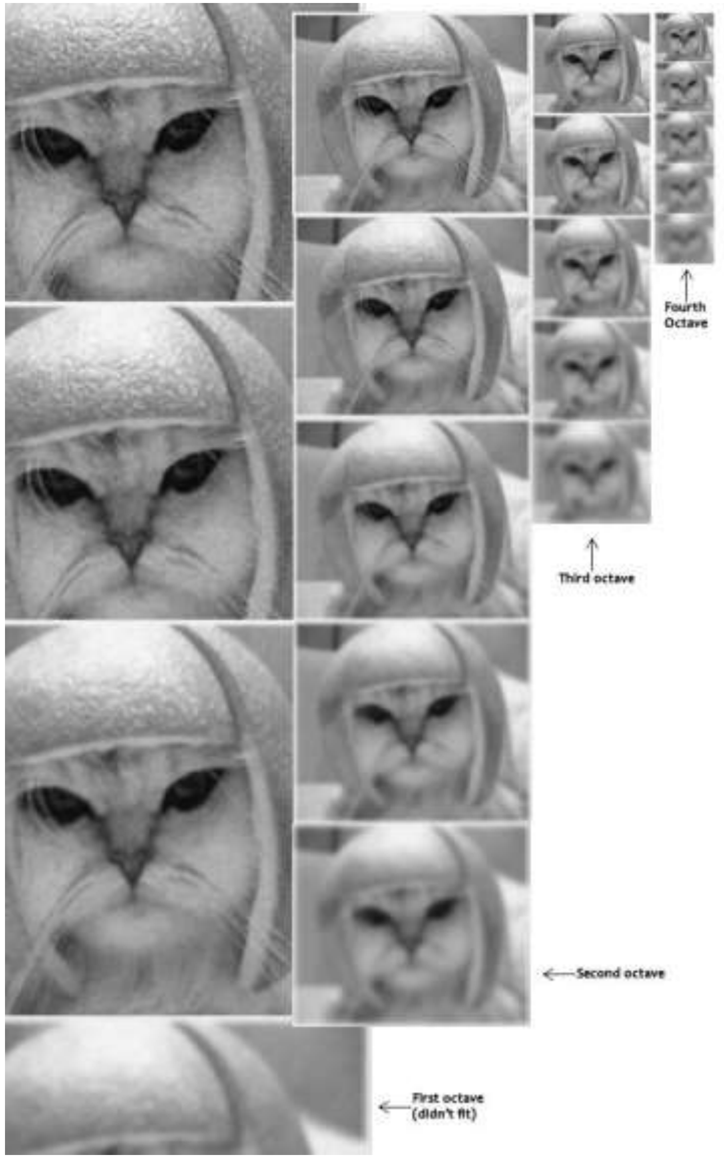
$$\sigma(s, o) = \sigma_0 2^{(o+(k/s))}$$

SIFT – Scale Invariant Feature Detection

Step 1: Construct Scale Space

- Example of a Gaussian Scale Space composed of four octaves





The SIFT Algorithm

Overview of the Approach

1. Create Scale Space (multi-scale Image Pyramid)
by applying larger and larger Gaussian blurring filters to the original image

2. Create “Differences of Gaussian” Image Pyramid
by taking difference between consecutive pyramid images

3. Find Local Extrema in the “Difference of Gaussian” Scale Space

4. Choose Candidate Keypoints from the Identified Extrema

5. Calculate and Assign Keypoint Orientation

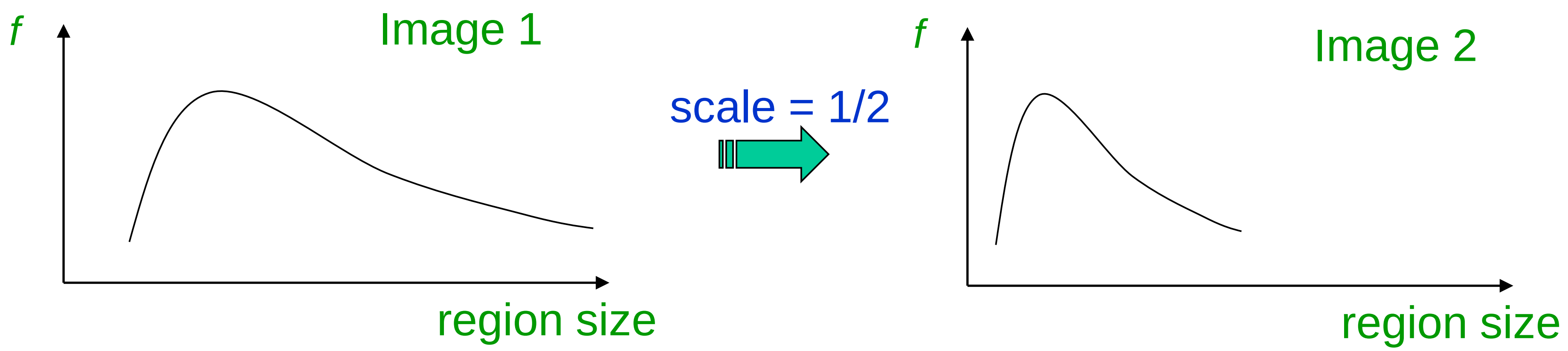
6. Build Keypoint Descriptor
by calculating gradient directions within a 16x16 window around keypoint

Match Keypoints found in different images

SIFT – Scale Invariant Feature Detection

Step 2: Take Difference of Gaussians

- **Problem 2:** Now that we have developed a scale space, **we need to identify a descriptor that can help us identify scale invariant features**
- **Solution:**
  - Find a function within a region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)
  - **Example**: average intensity [for corresponding regions (even of different sizes) it will be the same]

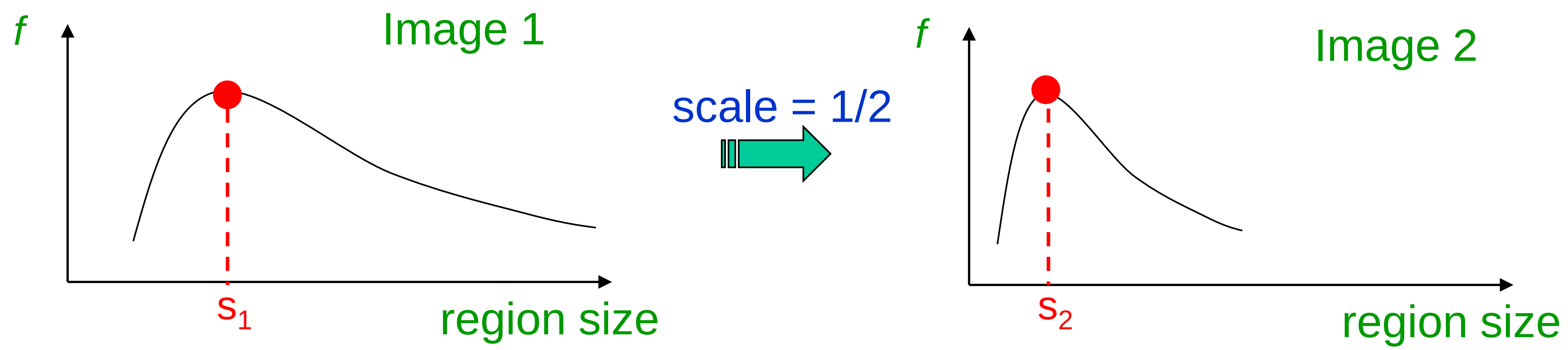


Example of an image function that looks the same at different scales (scale invariant)

SIFT – Scale Invariant Feature Detection

Step 2: Take Difference of Gaussians

- **Problem 2:** Now that we have developed a scale space, **we need to identify a descriptor that can help us identify scale invariant features**
- **Solution:**
  - To make detection of invariant features robust we usually **take a local maximum of this function**
  - **Observation**: region size, for which the maximum is achieved, should be invariant to image scale

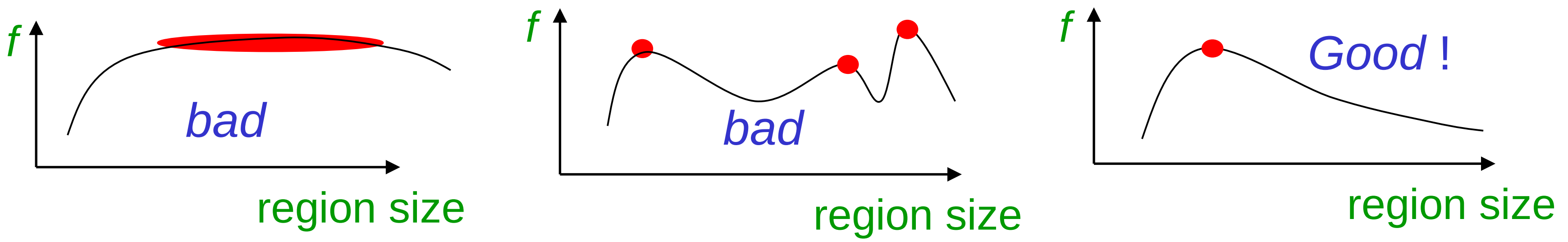


Example of an image function that looks the same at different scales (scale invariant)

SIFT – Scale Invariant Feature Detection

Step 2: Take Difference of Gaussians

- **Problem 2:** Now that we have developed a scale space, **we need to identify a descriptor that can help us identify scale invariant features**
- **Solution:**
  - **Finally**, To identify scale accurately, we want a function that has one sharp peak



Examples of image functions with varying peak quality

→ A good function would be one which responds to local contrast (sharp local intensity change)

SIFT – Scale Invariant Feature Detection

Step 2: Take Difference of Gaussians

- **Difference of Gaussians** is a good scale invariant feature producing a clear peak at a feature's spatial scale

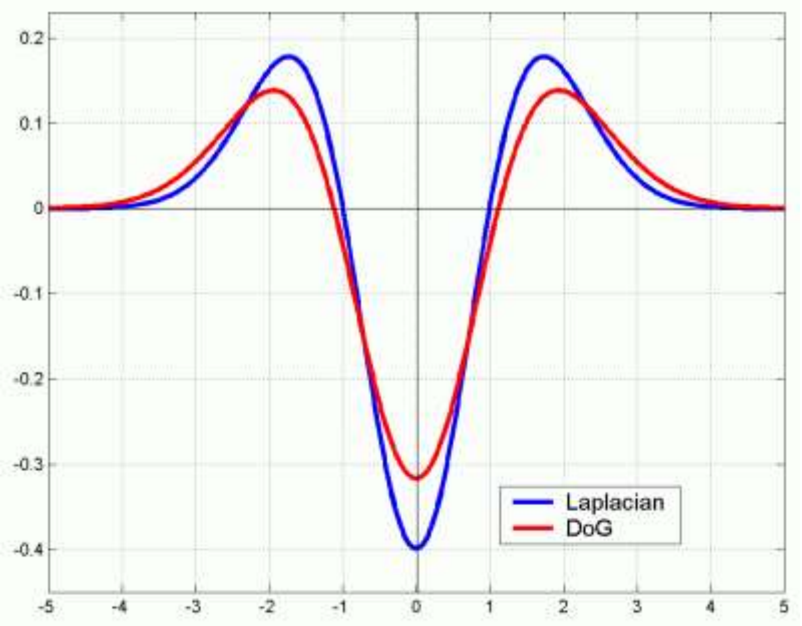
$$DoG = G(x, y, k\sigma) - G(x, y, \sigma)$$

  - With Gaussian

$$G(x, y, \sigma) = \frac{1}{\sqrt{2\pi} \cdot \sigma} exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

- DoG is related to the Laplacian L that some of you may know from digital image processing
  - Advantage of DoG is that it is faster to calculate

$$L = \sigma^2 \left(G_{xx}(x, y, \sigma) + G_{yy}(x, y, \sigma)\right)$$

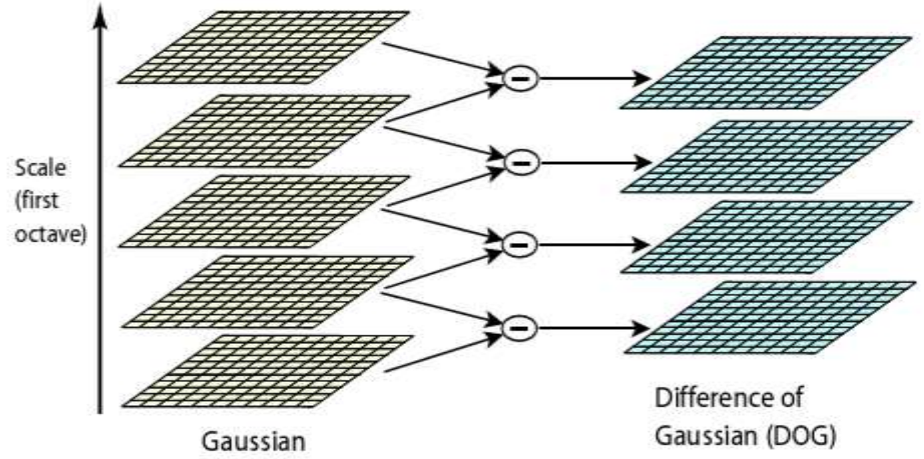


L or DoG kernel is a matching filter. It finds blob-like structure. It turns out to be also successful in getting characteristic scale of other structures, such as corner regions.

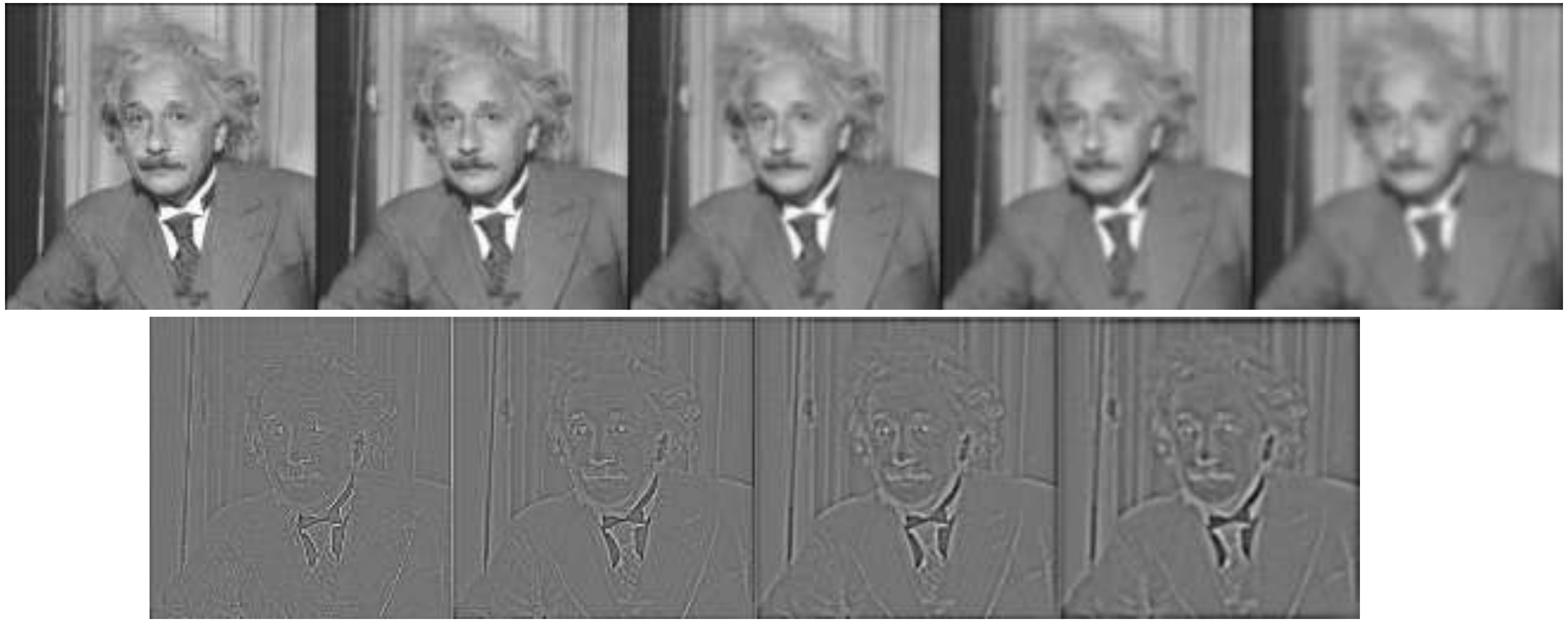
SIFT – Scale Invariant Feature Detection
Step 2: Take Difference of Gaussians

- **Example of DoG images through a Gaussian Scale Space**

Concept



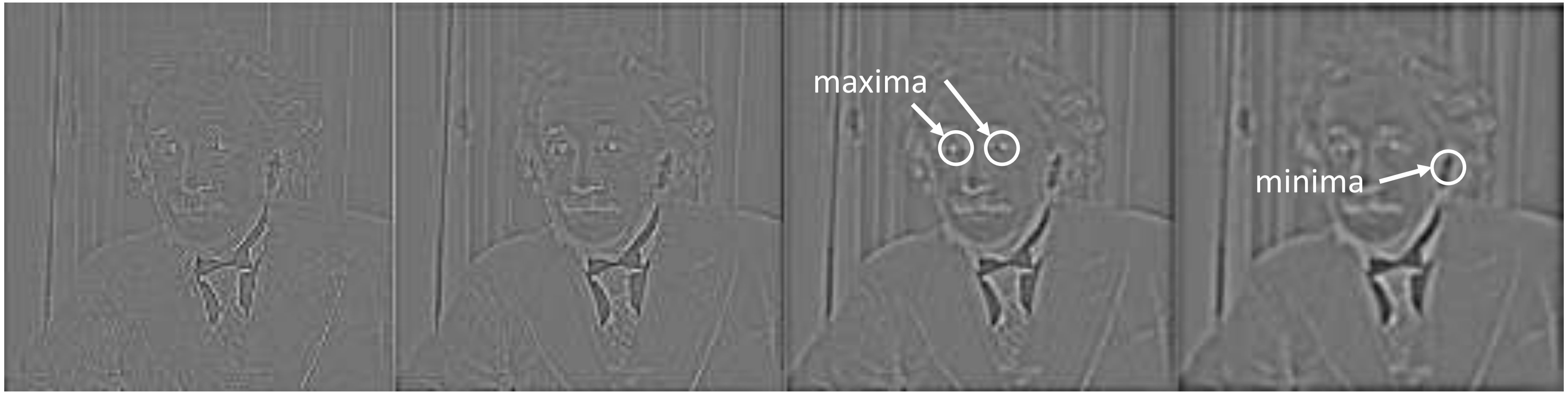
Example



Why the Difference of Gaussian is Useful

- Difference of Gaussian extrema identify and locate “blobs” in images → blob-type features are great for measuring the location of that feature in the image
  - Maxima = dark blobs on light background
  - Minima = light blobs on dark background

- The sigma parameter of the DoG filter can help determining the scale of a detected blob feature



The SIFT Algorithm

Overview of the Approach

1. Create Scale Space (multi-scale Image Pyramid)
by applying larger and larger Gaussian blurring filters to the original image

2. Create "Differences of Gaussian" Image Pyramid
by taking difference between consecutive pyramid images

3. Find Local Extrema in the "Difference of Gaussian" Scale Space

4. Choose Candidate Keypoints from the Identified Extrema

5. Calculate and Assign Keypoint Orientation

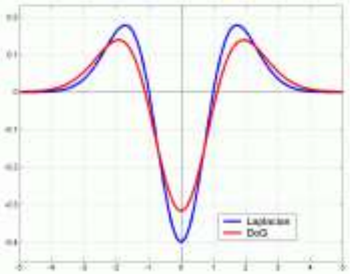
6. Build Keypoint Descriptor
by calculating gradient directions within a 16x16 window around keypoint

Match Keypoints found in different images

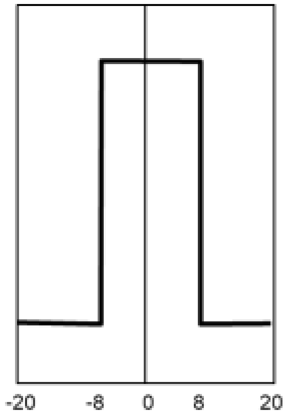
SIFT – Scale Invariant Feature Detection

Step 3 & 4: Automatic Scale Selection

- Automatic Scale Selection

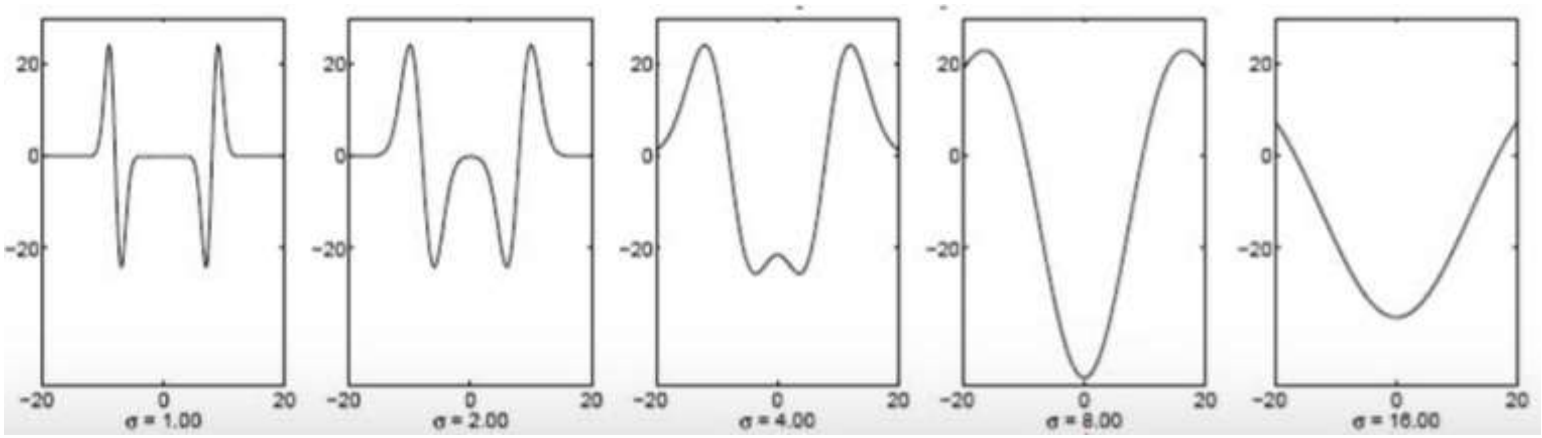


Original Signal



Radius = 8

DoG Responses



Increasing sigma parameter

Extremum

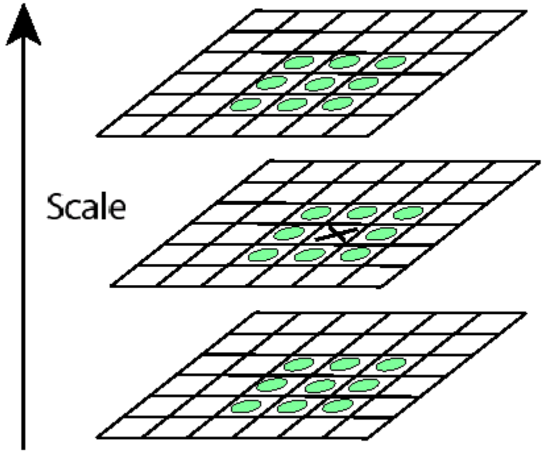
→ optimal scale at which feature can be detected in the image

→ location of feature in the image

SIFT – Scale Invariant Feature Detection

Step 3 & 4: Sub-pixel Local Spatial DoG Maximum

- **Detect DoG Extrema in the scale space by 3D Curve Fitting to DoG shape**



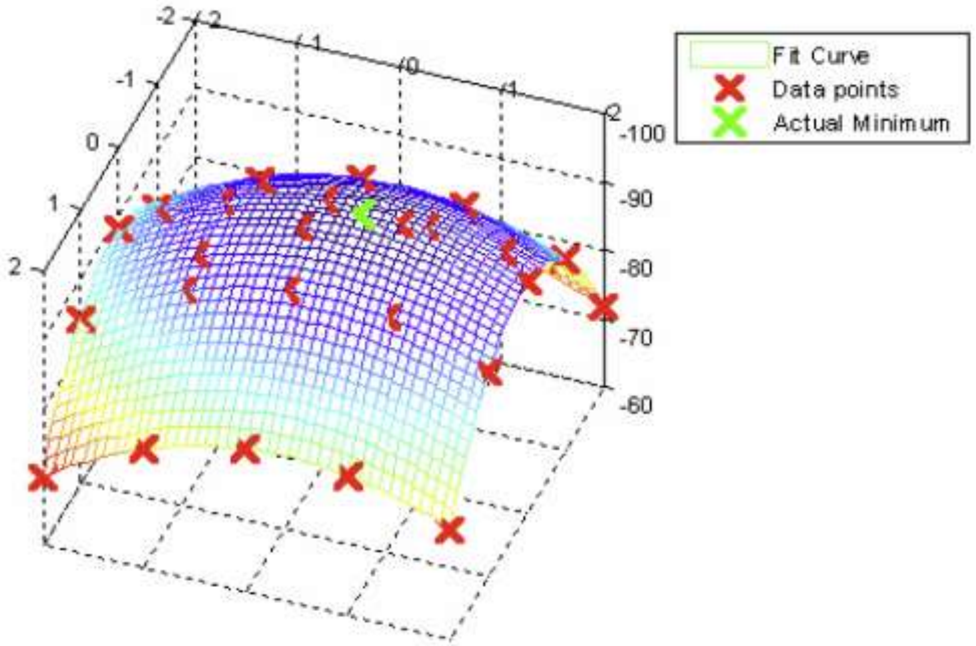
- **Then, Taylor Series expansion of fitted curve:**

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

  – Differentiate and set to 0

$$\hat{x} = -\frac{\partial^2 D}{\partial x^2}^{-1} \frac{\partial D}{\partial x}$$

  to get subpixel maximum locations.



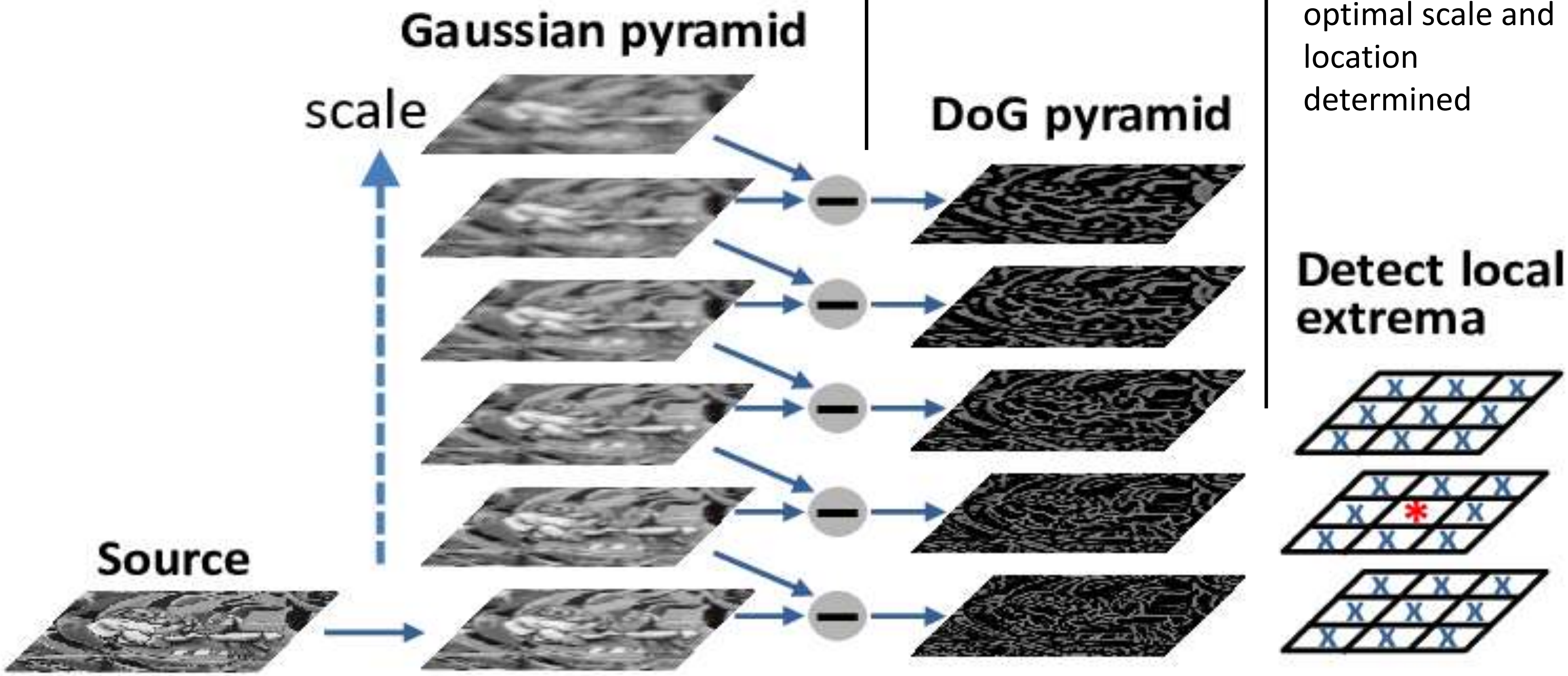
We achieved scale invariance !!
We achieved accurate feature location !!

SIFT Workflow so Far

Step 1: Image Pyramid
Facilitate scale invariance

Step 2: DoG Pyramids
Finds blob-like features (keypoints)

Step 3: Scale Selection
Keypoints detected at optimal scale and location determined



The SIFT Algorithm

Overview of the Approach

1. Create Scale Space (multi-scale Image Pyramid)
by applying larger and larger Gaussian blurring filters to the original image

2. Create "Differences of Gaussian" Image Pyramid
by taking difference between consecutive pyramid images

3. Find Local Extrema in the "Difference of Gaussian" Scale Space

4. Choose Candidate Keypoints from the Identified Extrema

5. Calculate and Assign Keypoint Orientation

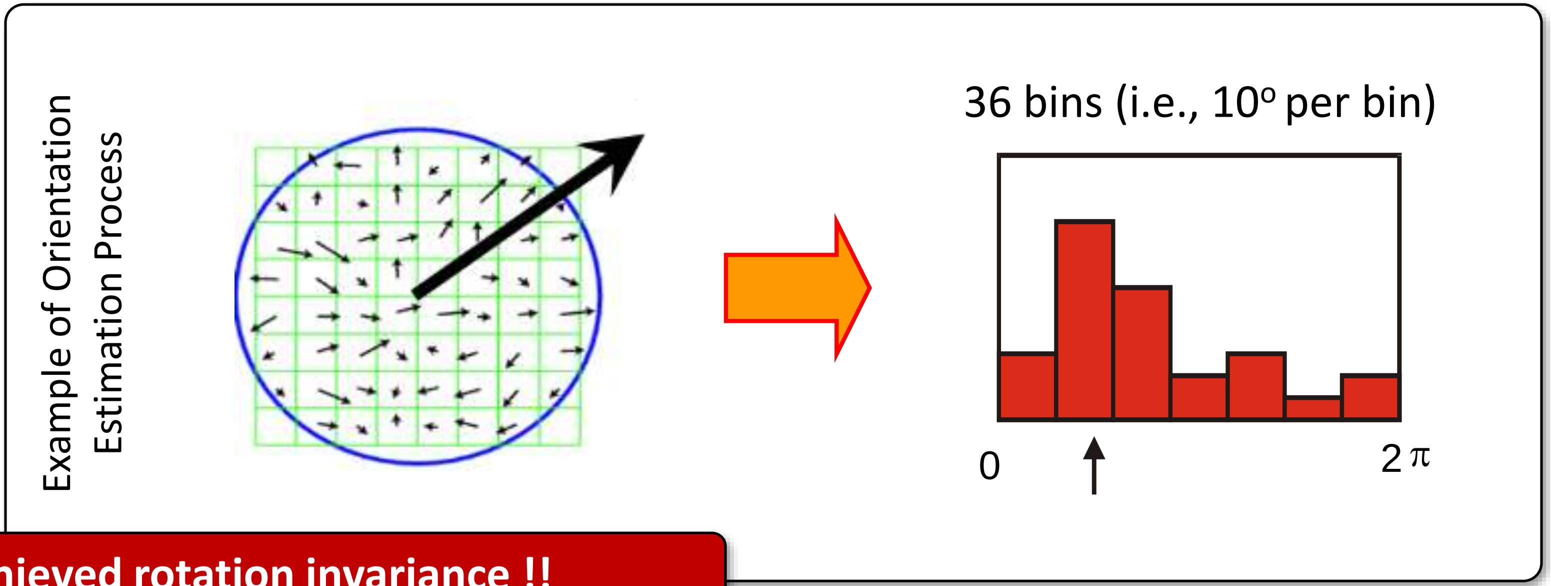
6. Build Keypoint Descriptor
by calculating gradient directions within a 16x16 window around keypoint

Match Keypoints found in different images

SIFT – Scale Invariant Feature Detection

Step 5: Assign Keypoint Orientations

- **Knowing the orientation of identified features makes matching more robust**
- **Compute gradients** for at the optimal scale of the detected keypoint
- **For 8x8 region around keypoint**
  - Calculate $8 \cdot 8 = 36$ gradients and create directional histogram
  - Weight each point by distance from keypoint with Gaussian window of 1.5σ
  - **Maximum of directional histogram is keypoint orientation**



We achieved rotation invariance !!

The SIFT Algorithm

Overview of the Approach

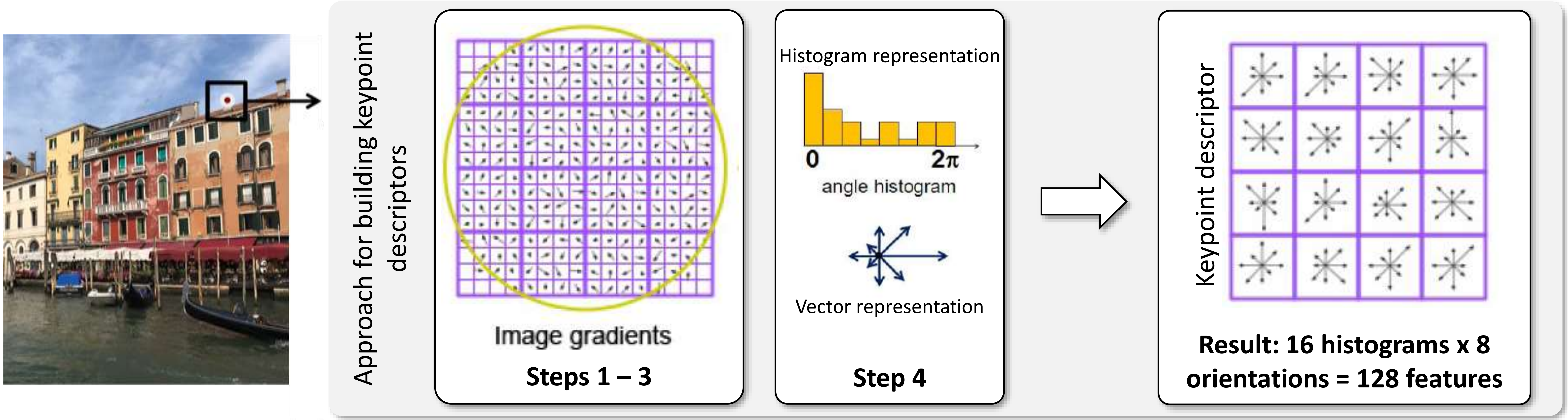
1. **Create Scale Space (multi-scale Image Pyramid)**
   by applying larger and larger Gaussian blurring filters to the original image
2. **Create "Differences of Gaussian" Image Pyramid**
   by taking difference between consecutive pyramid images
3. **Find Local Extrema in the "Difference of Gaussian" Scale Space**
4. **Choose Candidate Keypoints from the Identified Extrema**
5. **Calculate and Assign Keypoint Orientation**
6. **Build Keypoint Descriptor**
   by calculating gradient directions within a 16x16 window around keypoint

Match Keypoints found in different images

SIFT – Scale Invariant Feature Detection

Step 6: Build Keypoint descriptors

- **Goal:** Extract as much information as possible for each keypoint to improve matching quality and reduce amount of mis-matches between images – **we use local image gradients as a descriptor**
- **Approach per keypoint:**
  1. Find the blurred image of closest scale
  2. Take a 16 x16 window around detected keypoint
  3. Divide into a 4x4 grid of cells.
  4. Compute histogram in each cell [typically 8 bin histogram]



SIFT – Scale Invariant Feature Detection

Back to Performance Check

- **Criterion 1:** Scale Invariance ✓
  - Scale Space usage
- **Criterion 2:** Rotation Invariance ✓
  - Align with largest gradient
- **Criterion 3:** Illumination Invariance ✓
  - Gradient based rather than illumination based
- **Criterion 4:** Viewpoint Invariance (✓)
  - For small viewpoint changes –Check (mostly)

The SIFT Algorithm
Overview of the Approach

1. Create Scale Space (multi-scale Image Pyramid)
by applying larger and larger Gaussian blurring filters to the original image

2. Create "Differences of Gaussian" Image Pyramid
by taking difference between consecutive pyramid images

3. Find Local Extrema in the "Difference of Gaussian" Scale Space

4. Choose Candidate Keypoints from the Identified Extrema

5. Calculate and Assign Keypoint Orientation

6. Build Keypoint Descriptor
by calculating gradient directions within a 16x16 window around keypoint

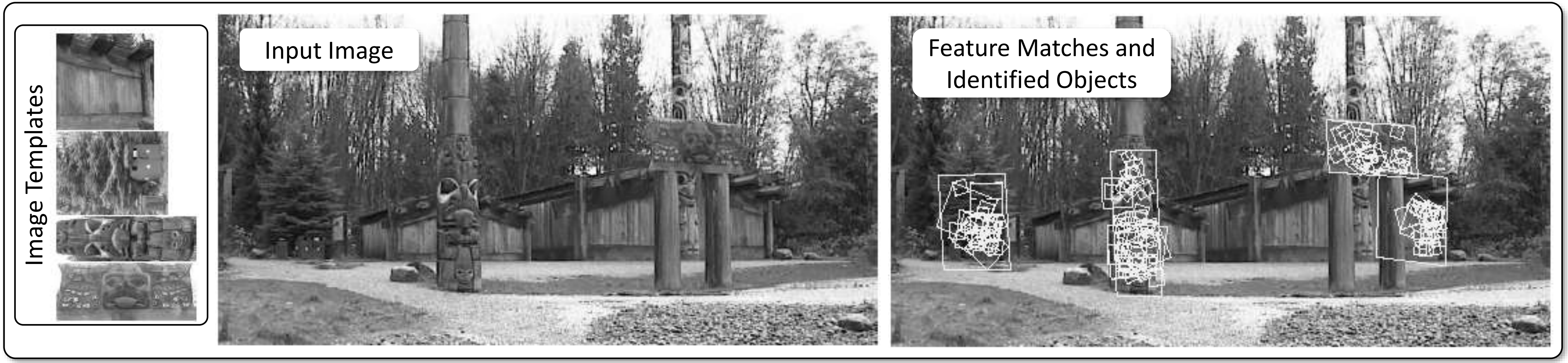
Match Keypoints found in different images

SIFT – Scale Invariant Feature Detection

Identifying and Matching Objects Based on SIFT Features

- **Can be done with as few as 3 SIFT features.**
- **Use Hough transform to cluster features belonging to one object**
- Clusters can be identified in Hough space after 3 entries
- More details in the next slide (hidden in this presentation but available in the online slide deck)

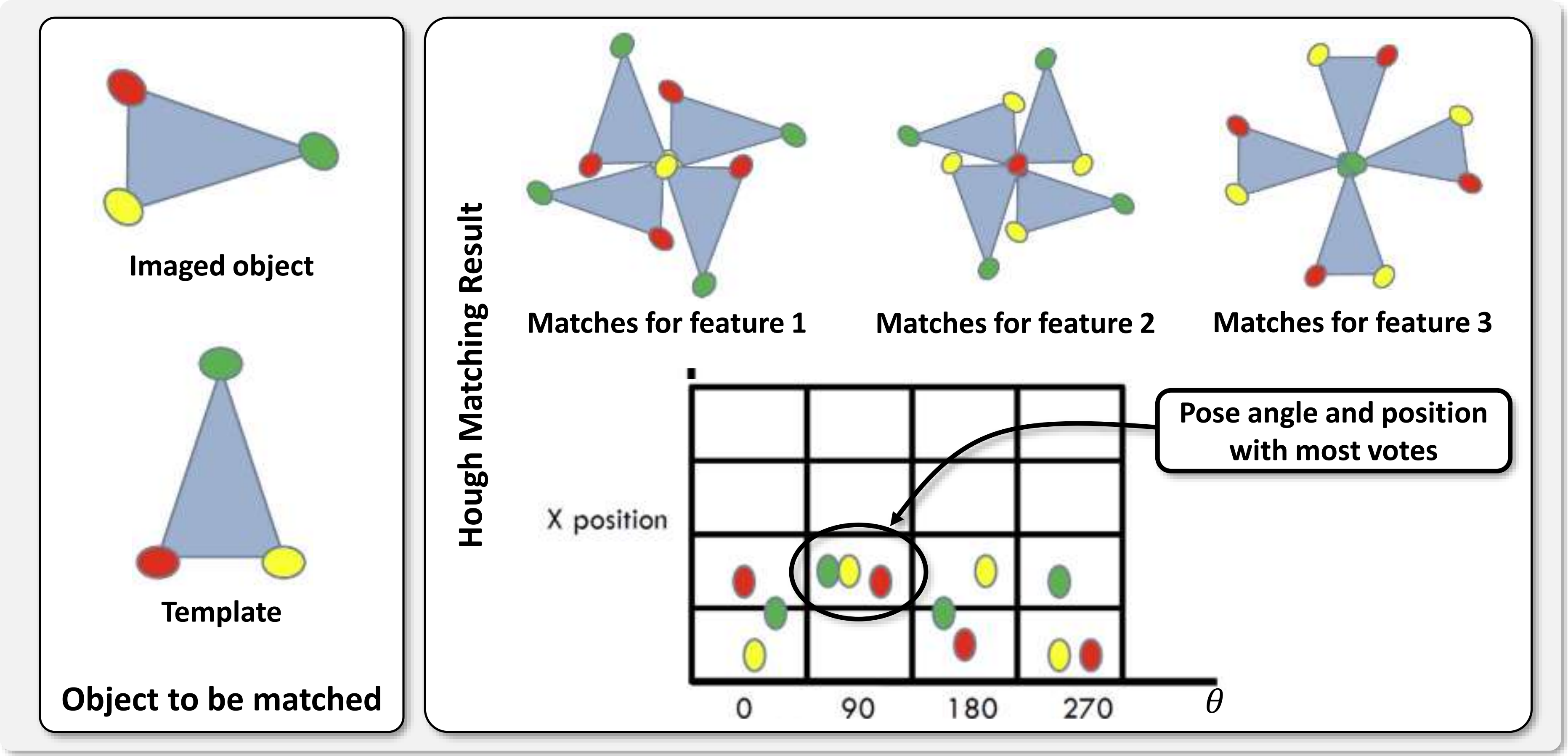
- **Example problem to solve:** Identifying Objects (templates) in this outdoor scene



SIFT – Scale Invariant Feature Detection

Hough Transform Matching Concept

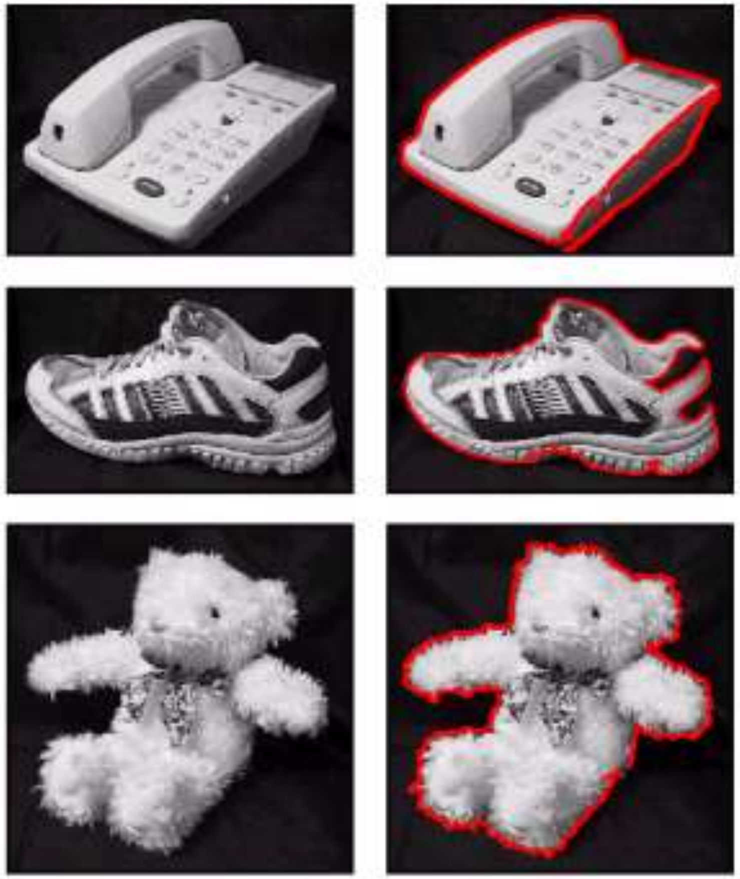
- For the Current View, color feature match with template image
- Take each feature and align template image at that feature → vote for the x position of the object center and the θ of the object based on all aligning poses



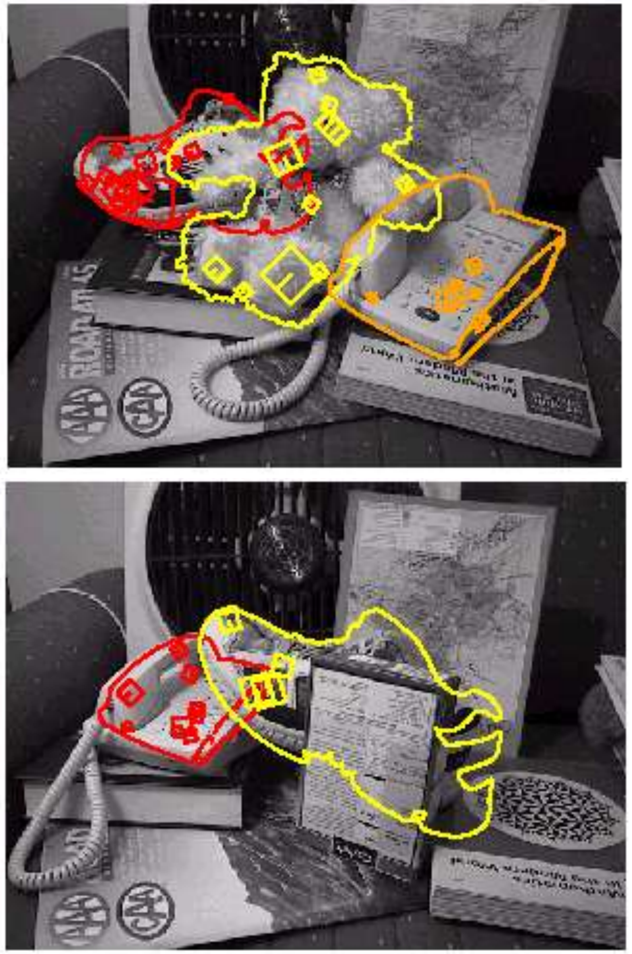
SIFT – Scale Invariant Feature Detection
Matching Examples

- **Identifying objects (even if partially obstructed)**

Object Models

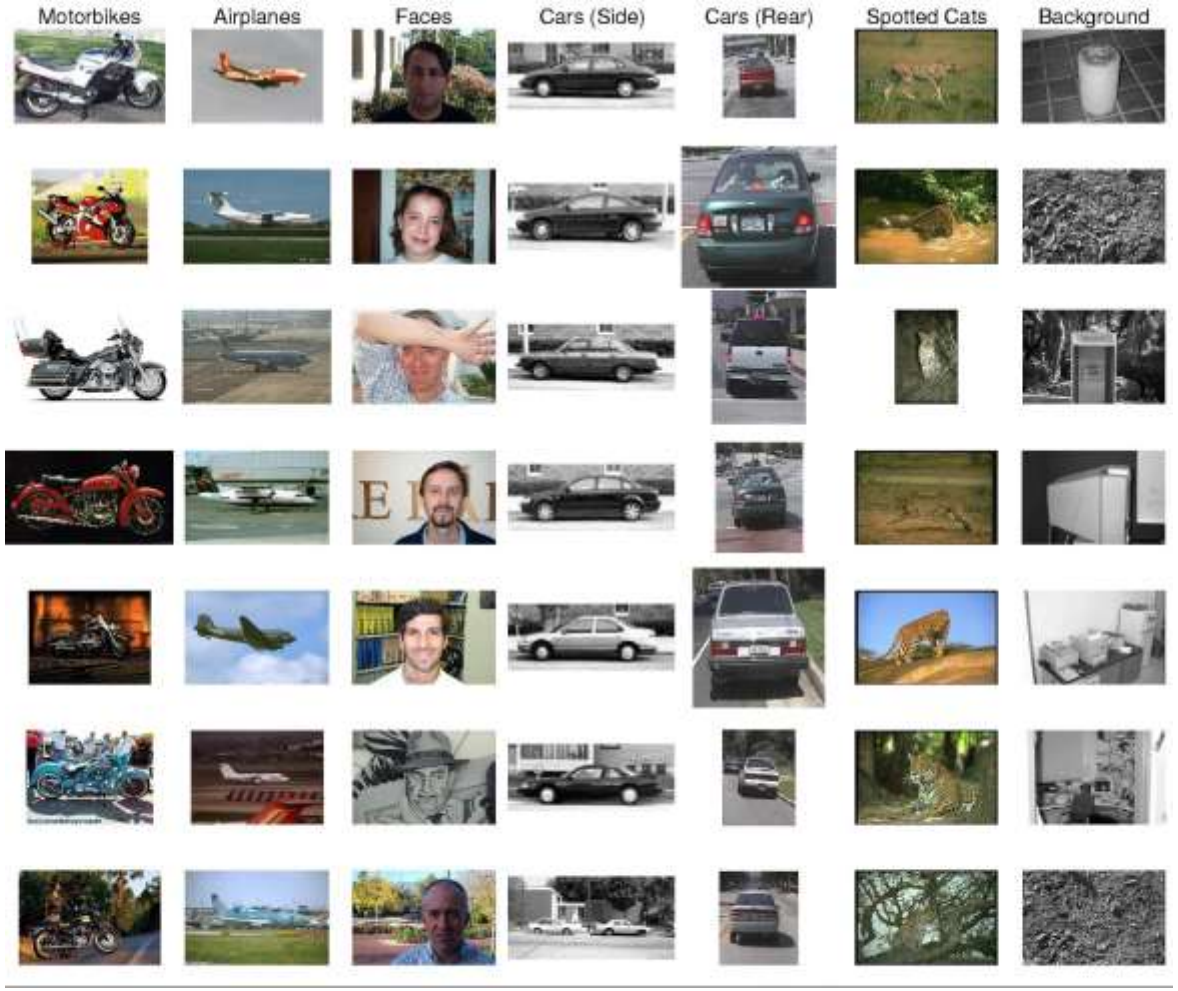


Recognition under occlusion



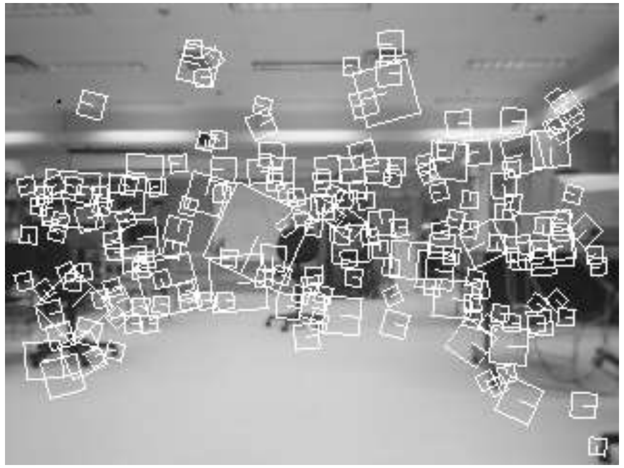
SIFT – Scale Invariant Feature Detection
Other SIFT Applications: Automatic Image Categorization

- **Flickr images automatically categorized based on SIFT features**



SIFT – Scale Invariant Feature Detection
Other SIFT Applications: Robot Trajectory Estimation

- **Automatic tracking of trajectory of robot in a room based on matched SIFT features**



SIFT – Scale Invariant Feature Detection

Other SIFT Applications: Automatic Image Retrieval from Data Bases

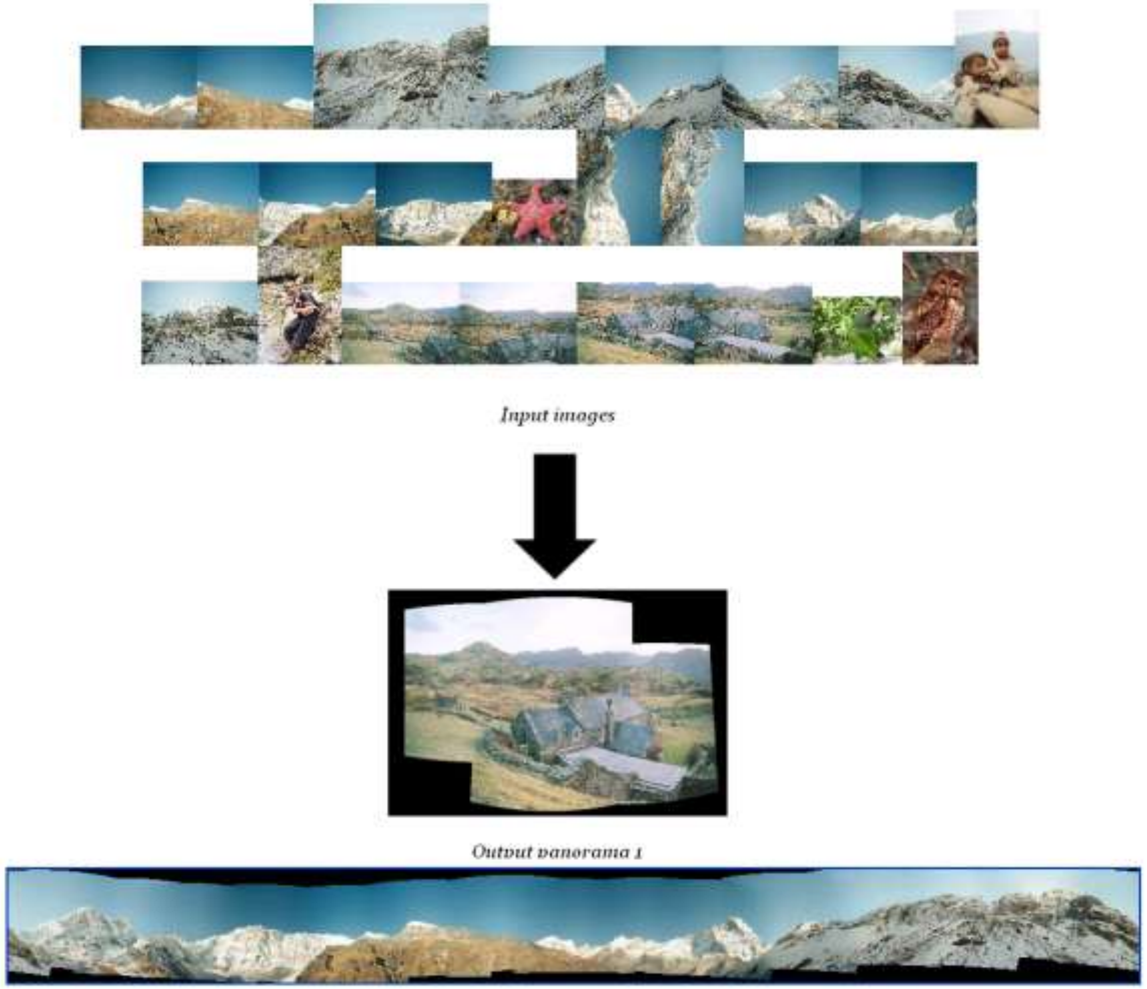
- **Automatic retrieval of matching images from the Flickr data base**

Template image

SIFT – Scale Invariant Feature Detection

Other SIFT Applications: Image panoramas from an unordered image set

- **Automatic mosaicking of unsorted image sets**



Cool Stuff Powered by SIFT

What's Next

- **Next Lecture:**
  - Structure-from-Motion Processing to generate Digital Elevation Models

- **After that:**
  - **Lab:** Structure-from-Motion for DEM processing