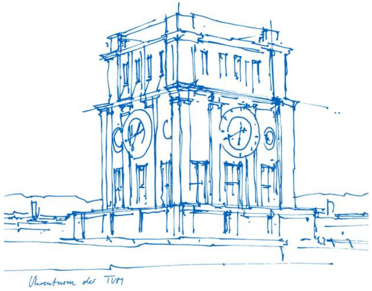
Computer Vision II: Multiple View Geometry (IN2228)

Chapter 05 Correspondence Estimation
(Part 3 Clarification and SIFT)

Dr. Haoang Li

31 May 2023 12:00-13:30



Announcement before Class

➢ Exam

✓ Cheat sheet

- We will hold a lecture for knowledge review in July (highlighting knowledge important for our exam). The review scope will be narrowed down. Accordingly, we tentatively do not allow the cheat sheet in our exam.

✓ Document for reviewing Chapters 01--05

- We uploaded a document to highlight important knowledge in Chapters 00—05. It will be highly relevant to the final exam. If you want, you can start to review Chapters 00-05 from now on. Please download this document from our course website or Moodle.
- The other pages related to the highlighted knowledge should be also reviewed. I will give you a more precise scope in the future review class.
- This document is subject to “slight” change since our exam questions have not been finalized.

Announcement before Class

➢ Updated Lecture Schedule

For updates, slides, and additional materials:
https://cvg.cit.tum.de/teaching/ss2023/cv2

90-minute course; 45-minute course

Foundation

Wed 19.04.2023	Chapter 00: Introduction
Thu 20.04.2023	Chapter 01: Mathematical Backgrounds
Wed 26.04.2023	Chapter 02: Motion and Scene Representation (Part 1)
Thu 27.04.2023	Chapter 02: Motion and Scene Representation (Part 2)
Wed 03.05.2023	Chapter 03: Image Formation (Part 1)
Thu 04.05.2023	Chapter 03: Image Formation (Part 2)
Wed 10.05.2023	Chapter 04: Camera Calibration
Thu 11.05.2023	Chapter 05: Correspondence Estimation (Part 1)
Wed 17.05.2023	Chapter 05: Correspondence Estimation (Part 2)
Thu 18.05.2023	No lecture (Public Holiday)
Wed 24.05.2023	No lecture (Conference)
Thu 25.05.2023	No lecture (Conference)

} Videos and reading materials about the combination of deep learning and multi-view geometry

Wed 31.05.2023	Chapter 05: Correspondence Estimation (Part 3)

Core part

Thu 01.06.2023	Chapter 06: 2D-2D Geometry (Part 1)
Wed 07.06.2023	Chapter 06: 2D-2D Geometry (Part 2)
Thu 08.06.2023	No lecture (Public Holiday)
Wed 14.06.2023	Chapter 06: 2D-2D Geometry (Part 3)
Thu 15.06.2023	Chapter 07: 3D-2D Geometry (Part 1)
Wed 21.06.2023	Chapter 07: 3D-2D Geometry (Part 2)
Thu 22.06.2023	Chapter 08: 3D-3D Geometry
Wed 28.06.2023	Chapter 09: Single-view Geometry (Part 1)
Thu 29.06.2023	Chapter 09: Single-view Geometry (Part 2)

Advanced topics and high-level task

Wed 05.07.2023	Chapter 10: Photometric Error (Direct Method)
Thu 06.07.2023	Chapter 11: Bundle Adjustment and Optimization
Wed 12.07.2023	Chapter 12: Robot Estimation
Thu 13.07.2023	Knowledge Review
Wed 19.07.2023	Chapter 13: SLAM and SFM (Part 2)
Thu 20.07.2023	Chapter 13: SLAM and SFM (Part 1)

Today’s Outline

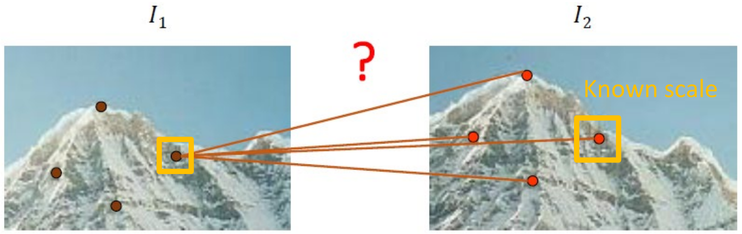
- Recap on Feature Matching Problem
- Recap on Patch Descriptor-based Method
- Response to Frequently-asked Questions
- A More Effective Method: SIFT
- Other Methods

Recap on Feature Matching Problem

➢ Problem Formulation

✓ Given a detected point in I_1, how to find the best match in I_2?
Here, we assume that **detected points (Harris)**, **points' descriptions (scale, rotation etc.)** are both known. A naive matching strategy is **brute force matching**.

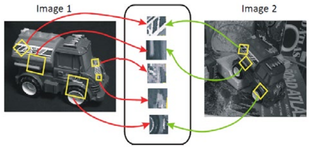
- Based on descriptor similarity, compare each feature in I_1 against all the features in I_2 (N^2 comparisons, where N is the number of features in each image). Select the point pair with the minimum distance.
- How to define descriptor? How to measure similarity?



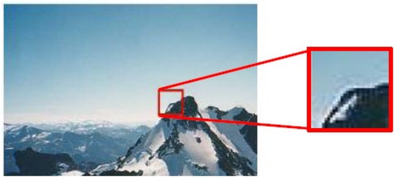
Recap on Feature Matching Problem

➢ Two types of descriptions

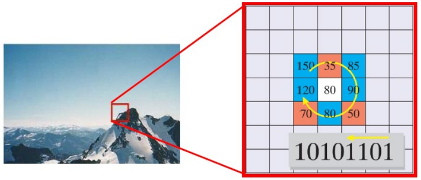
✓ Patch descriptor (i.e., patch of **intensity values**)
- Introduced in our last class
- Patch need to be warped to the canonical space



Canonical space

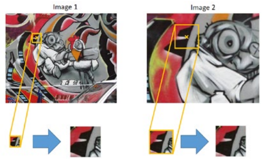


✓ Census descriptor (a vector with integer/float values)
- Introduced today



Recap on Feature Matching Problem

➢ Descriptor similarity measurement (2D patch)



2D patches

✓ **Sum of Squared Differences (SSD):** always ≥ 0. It's exactly 0 only if H and F are identical

$$SSD = \sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(H(u,v) - F(u,v)\right)^2$$

✓ Sum of Absolute Differences (SAD): always ≥ 0. It's 0 only if H and F are identical

H and F denote left and right patches respectively

$$SAD = \sum_{u=-k}^{k}\sum_{v=-k}^{k}\left|H(u,v) - F(u,v)\right|$$

✓ To account for the difference in the average intensity of two images (typically caused by additive illumination changes), we subtract the mean value of each image:

$$\mu_H = \frac{1}{n}\sum_{u=-k}^{k}\sum_{v=-k}^{k}H(u,v) \qquad \mu_F = \frac{1}{n}\sum_{u=-k}^{k}\sum_{v=-k}^{k}F(u,v) \quad \Rightarrow \quad ZSSD = \sum_{u=-k}^{k}\sum_{v=-k}^{k}\left(\left(H(u,v) - \boxed{\mu_H}\right) - \left(F(u,v) - \boxed{\mu_F}\right)\right)^2$$

Zero-mean

Recap on Feature Matching Problem

➢ Descriptor Similarity measurement (1D census vector)

✓ Normalized Cross Correlation (NCC): ranges between -1 and +1 and is exactly 1 if H and F are identical

(1,1) and (1,1):
NCC = 1

$$NCC = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}}$$

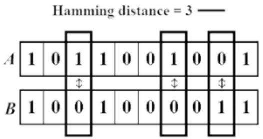
(1,1) and (-1,-1):
NCC = -1

1D descriptors

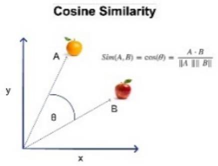
✓ Cosine Euclidean, or Hamming distance are also applicable

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Euclidean distance



A	1	0	1	1	0	0	1	0	0	1
			↕				↕		↕	
B	1	0	0	1	0	0	0	0	1	1



Recap on Feature Matching Problem

➢ Structure of Knowledge Introduction

✓ Two types of methods

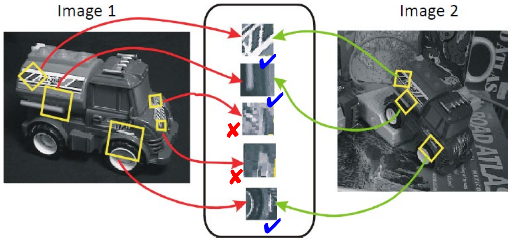
Feature matching
(we only talk about brute force matching strategy)

- **Patch descriptor-based** method
(introduce in the last lecture;
Let us review it first)
  - Exhaustive/straightforward search of scale
(two images)
  - Automatically scale determination
(a single image)
- **Census descriptor-based** method
(today)
  - Automatically scale determination
(a single image)

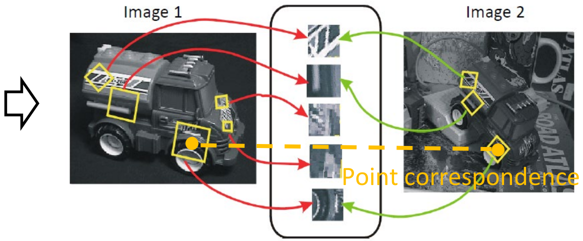
Recap on Patch Descriptor-based Method

➢ General pipeline

✓ It is based on the patch descriptor (scale, rotation, etc.).
✓ Warp each patch into a canonical patch.
✓ Establish point correspondences based on similarity (SSD) of the warped patch.



Canonical space



Recap on Patch Descriptor-based Method

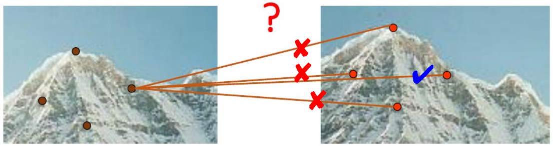
➢ Properties of Patch Descriptor

✓ Distinctiveness of a feature descriptor

- A descriptor is a “description” of the pixel information **around** a feature.
- “Distinctiveness” means that the descriptor can uniquely distinguish a feature from the other features **without ambiguity**.

✓ Robustness to geometric changes

- **Scale**-invariant (for zooming)
- **Rotation**-invariant
- View point-invariant (for perspective changes)



Distinctiveness

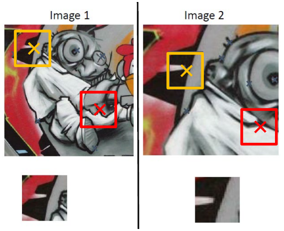
Geometric changes

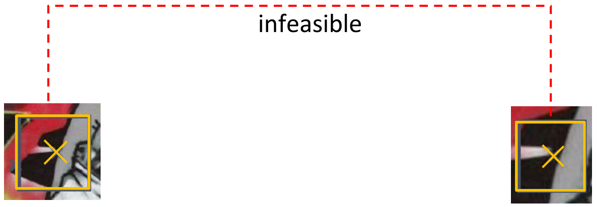
Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Straightforward but inefficient patch scale search (relying on two images)

- Assume that we have independently detected some key points in the left and right *images with different scales* (here, we only consider scale).
- We aim to use **brute force matching** to match points based on sum of squared distances (SSD) of their associated patch features.
- Problem: Two patches with **the same size have different appearances due to different scales** (**even if they are corresponding patches**), and thus we cannot **directly** use SSD (see below).



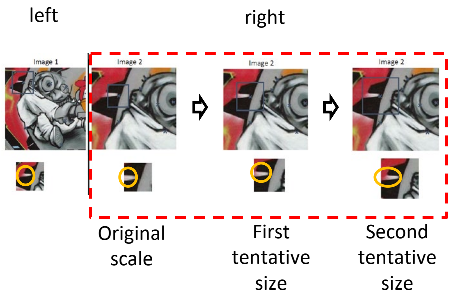


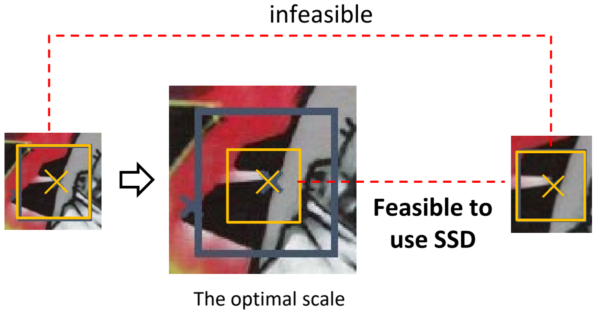
Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Straightforward but inefficient patch scale search (relying on two images)

- A straightforward solution is to keep the **left patch unchanged, and resize** the **right patch** with different tentative sizes.



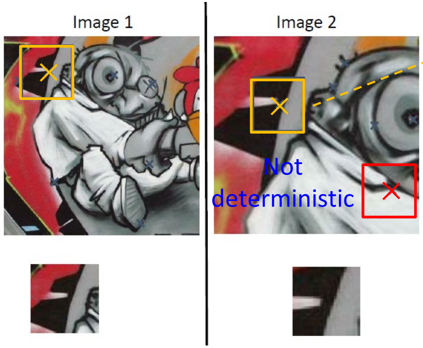


Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Straightforward but inefficient patch scale search (relying on two images)

- In practice, for **each** left patch, we have to validate **all the** right patches using **all the** tentative scales.
- We take a patch in the left image for example.



Scale 1	SSD is large (inappropriate scale)
Scale 2	**SSD is small (appropriate scale)**
…	
Scale N	SSD is large (inappropriate scale)

Scale 1
Scale 2
…
Scale N

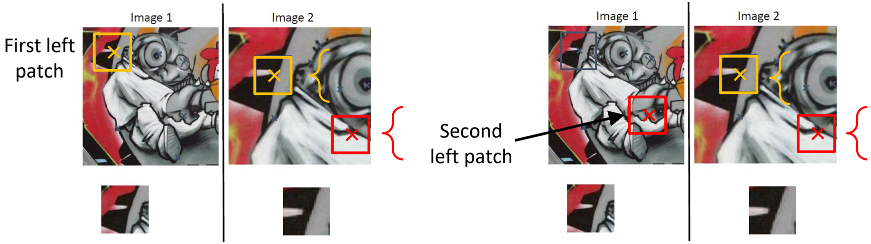
All the SSD are large (they are not point correspondence!)

Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Straightforward but inefficient patch scale search (relying on two images)

- Drawback 1: Scale determination **depends on** tentative matching.
  Algorithm complexity is N^2S assuming N features per image and S tentative sizes per feature.
  First N: number of left patches; Second N: number of right patches
- Drawback 2: We fix the scale of left patch, but cannot guarantee this scale is optimal (distinctive enough).

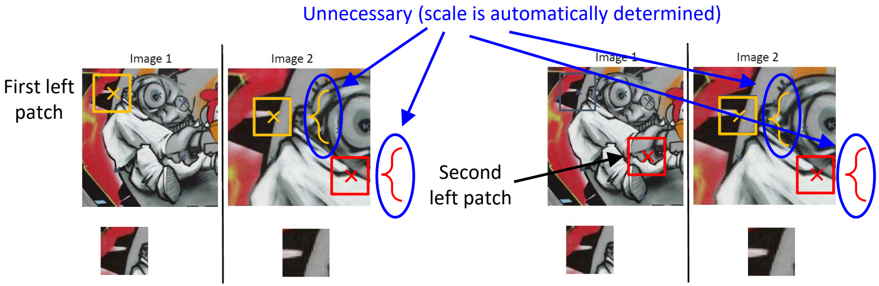


Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Straightforward but inefficient patch scale search (relying on two images)

- Relationship with brute force matching with **known** scales (introduced before): In brute force matching, we assume that scale of each **patch is known a priori**. So, the complexity is only N^2
- It would be great if we can determine the scale before matching (**independent of matching**). How to achieve this? (How to **automatically** determine the scale in **a single image**?)

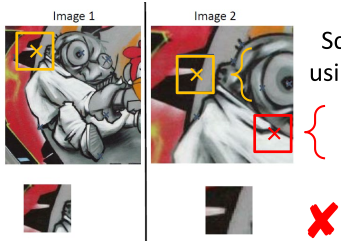


Recap on Patch Descriptor-based Method

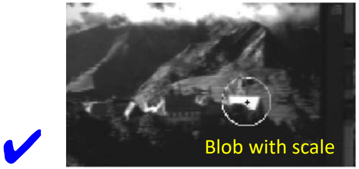
➢ Scale of Patch Descriptor

✓ Automatic scale determination

- We aim to "**automatically" assign each patch (both left and right) its own size**.
- In other words, we assign scale based on **a single image**. Our scale assignment **is independent of tentative matching**.



Scale is determined using **a pair of** images



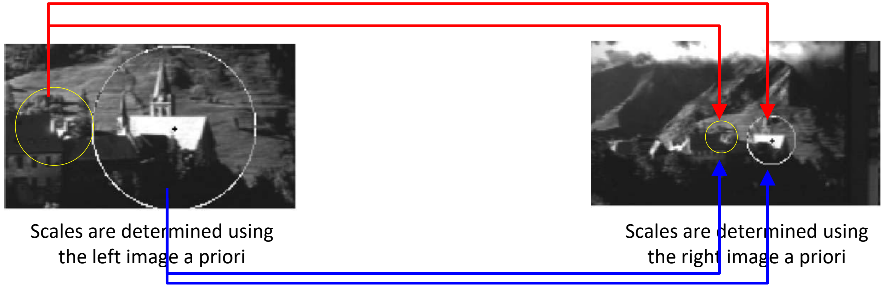
Scale is determined using **a single** image

Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ Automatic scale determination

- If we achieve the automatic scale determination, the straightforward but inefficient method introduced before **degenerate into** the brute force matching with known scales (no tentative scale test).
- This is just an overview. I will review **details of automatic scale determination** later.

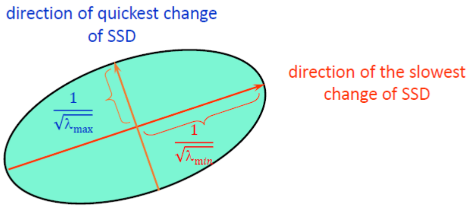

Scales are determined using the left image a priori
Scales are determined using the right image a priori

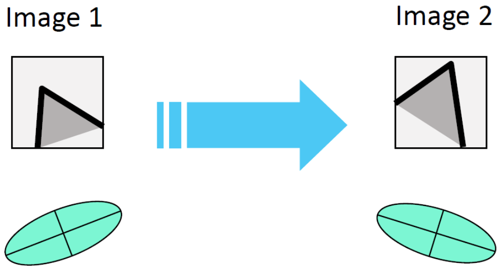
Explanations for FAQ

➢ Patch Descriptor about Rotating

✓ Two strategies to rotate patch (first)

- The Harris detector is rotation invariant
- Eigenvectors of M matrix correspond to the directions of quickest and slowest changes of SSD


direction of quickest change of SSD
$\frac{1}{\sqrt{\lambda_{\max}}}$
direction of the slowest change of SSD
$\frac{1}{\sqrt{\lambda_{min}}}$



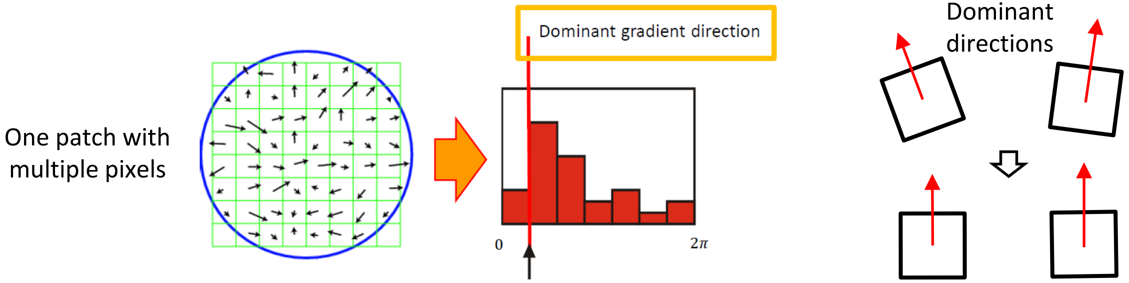
Ellipse rotates but its shape (i.e., eigenvalues of M) remains the same.

Response to FAQ

➢ Patch Descriptor about Rotating

✓ Two strategies to rotate patch (second)

- Compute gradients vectors **at each pixel** within a patch.
- Build a histogram of gradient orientations, weighted by the gradient magnitudes (norm of vector).
- Extract all local maxima in the histogram: each local maximum above a threshold is a candidate dominant orientation. (Typically, we have up to three directions.)

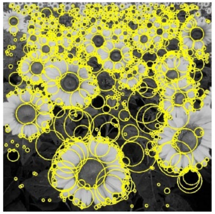


Response to FAQ

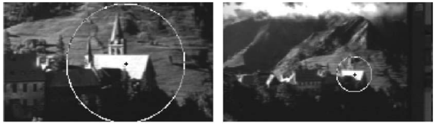
➢ Understanding Blob with Scale

✓ Formal definition of blob (introduced in our previous class)

- A blob is a group of connected pixels in an image that share some common property (e.g, grayscale value).
- In the image below, the colored regions are blobs. Given **a single image**, we aim to detect blobs and **"automatically" assign them "appropriate" scales**.



"**Automatically**" obtained scale in a single image



Different blob scales in two images

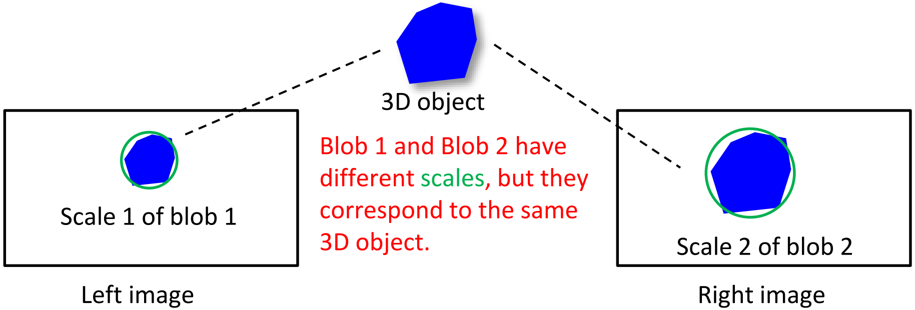
Appropriate: Circles correspond to the same 3D area

Response to FAQ

➢ Understanding Blob with Scale

✓ Reason for blob detection/marking

- In feature **matching** problem, a blob inherently encodes the **scale** information (corner can hardly encode this information). Accordingly, we can directly resize the two patches and evaluate them by SSD.
- Mathematically, scale can be expressed by the radius of circle.



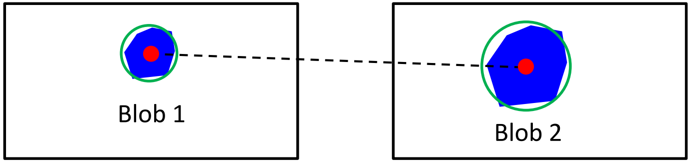
Blob 1 and Blob 2 have different scales, but they correspond to the same 3D object.

Response to FAQ

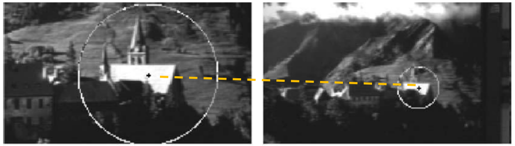
➢ Understanding Blob with Scale

✓ From blob matching to point correspondence

- Given a pair of matched blobs, their centers constitute a point correspondence.
- A small drawback: Blob centers may not be very precise, compared with corners



Left image   Right image



Left image   Right image

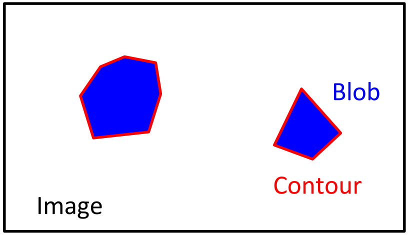
Blob centers constitute a point correspondence

Response to FAQ

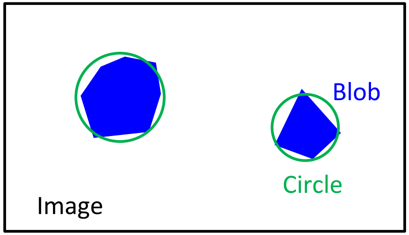
➢ Understanding Blob with Scale

✓ First type of misunderstanding

- Some students may misunderstand that we aim to find the precise contours to tightly enclose blobs with different shapes (see below).
- Instead, in our context, we focus on finding a **circle** to appropriately **mark** each blob (circle is unnecessarily an inscribed or circumscribed circle).



Misunderstanding



Our goal

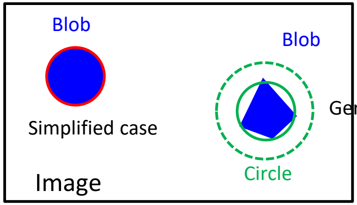
Response to FAQ

➢ Understanding Blob with Scale

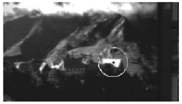
✓ Second type of misunderstanding

- Some students may misunderstand that we can only handle circular blob. It is not the case. In practice, we can deal with blobs with arbitrary shapes.

- When talking about scale computation, I consider the **circular blob** for illustration.
- I originally want to use this simplified case to help you understand the overall pipeline without going into too many details. However, it seems that it may mislead you. In the following, I will clarify this issue and then use another way to introduce knowledge.



General case

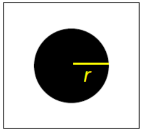


Response to FAQ

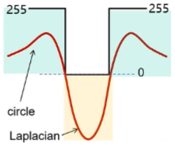
➢ Understanding Blob with Scale

✓ Simplified but not general case (introduced in the last class)

- Assume that we have detected a circular blob. Which scale will lead to the maximum convolution response?



A known circular blob



Configuration for maximum response

Simplified case

A conclusion introduced in the class. Relationship between scale σ and radius r is $\boxed{\sigma = r/\sqrt{2}.}$

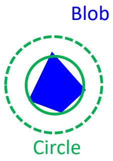
$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right]$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$(x^2+y^2-2\sigma^2) \cdot exp(-\frac{(x^2+y^2)}{2\sigma^2})$$

$$r^2 - 2\sigma^2 = 0$$



General case

- In practice, we do not know circular blob a priori. So, 1) how can we detect a blob with an arbitrary shape? 2) Can we still use a circle to mark such a blob? 3) What's the practicality of the above conclusion?

Derivation of conclusion will not be asked in the exam.

Response to FAQ

➢ Understanding Blob with Scale

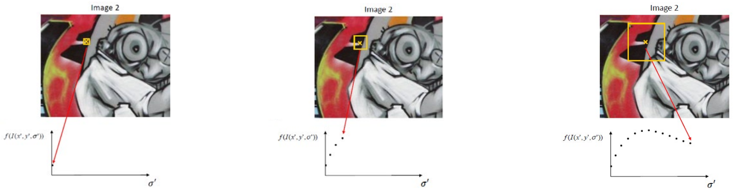
✓ How to detect a blob and determine its optimal scale σ?

- Main idea: we have to try a set of candidates.
- Only a single image is enough. We take a patch for example.
- Briefly, we apply a kernel (Laplacian of Gaussian) w.r.t. a single parameter σ to **an image patch**.
- We validate a set of **candidate** scales σ. A scale leading to the **maximum response** is the optimal scale of patch (**characteristic scale**).

$$f = \text{Kernel} * \text{Image}$$

Function (response) ↓ Patch

Laplacian of Gaussian (LoG) w.r.t. a single unknown parameter σ



Function extremum

Response to FAQ

➢ Understanding Blob with Scale

$$f = \text{Kernel} * \text{Image}$$

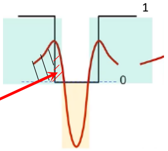
Function (response)

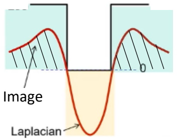
Patch

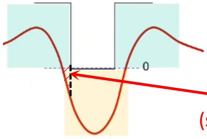
Laplacian of Gaussian (LoG) w.r.t. a single unknown parameter σ

✓ How to detect a blob and determine its optimal scale σ?

- Intuitive illustration of convolution between patch and LoG







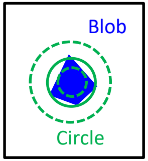
Zero-valued convolution (since pixel intensity=0; LoG>0)

Negative convolution (since LoG <0; pixel intensity>0)

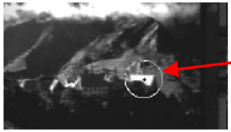
Smaller response | Maximum response | Smaller response

- From optimal scale to circle that is used to mark blob

Image



General case



We draw circles based on the conclusion

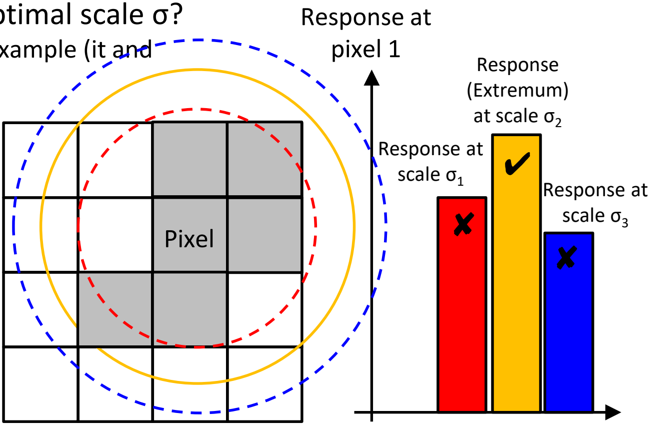
$$\sigma = r / \sqrt{2}.$$

Response to FAQ

➢ Understanding Blob with Scale

✓ How to detect a blob and determine its optimal scale σ?

- A more detailed explanation: We take a pixel for example (it and its neighbors have similar brightness)
- We try several candidate scales
- Save the scale achieving the extremum


Response at pixel 1
Response (Extremum) at scale σ_2
Response at scale σ_1
Response at scale σ_3
Pixel

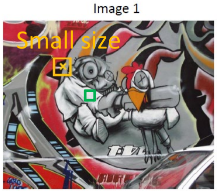
Find the optimal scale at a certain pixel

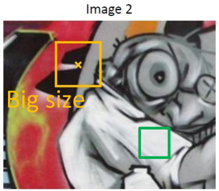
Response to FAQ

➢ Understanding Blob with Scale

✓ Intuitive illustration of reason for selecting LoG kernel

- Human can easily determine the scale of a patch with sufficient textures (significant gradient).
- We extract **edge** based on Laplace.
- By extension, we detect **blob** with scale based on LoG.

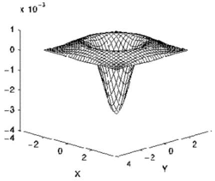


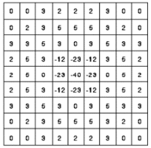


Discontinuity

Smoothness

Human can determine the scale by perceiving the change of patch appearance.





0	0	3	2	2	2	3	0	0
0	2	3	5	5	5	3	2	0
3	3	5	3	0	3	5	3	3
2	5	3	-12	-23	-12	3	5	2
2	5	0	-23	-40	-23	0	5	2
2	5	3	-12	-23	-12	3	5	2
3	3	5	3	0	3	5	3	3
0	2	3	5	5	5	3	2	0
0	0	3	2	2	2	3	0	0

σ = 1.4 (adjustable)

$$LoG = -\frac{1}{\pi\sigma^4}\left[1-\frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

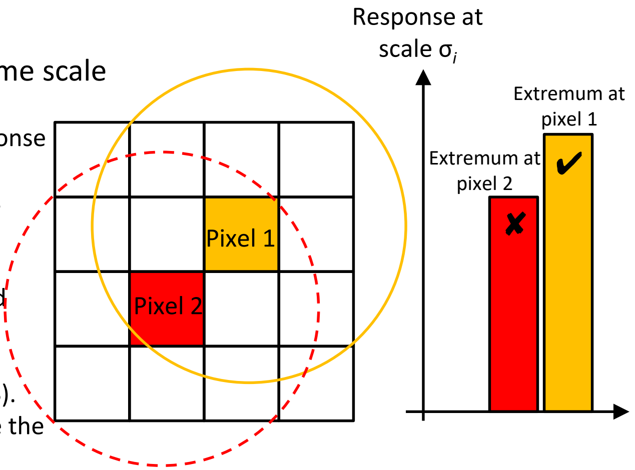
Laplacian of Gaussian (LoG) w.r.t. a single unknown parameter σ

Response to FAQ

➢ Understanding Blob with Scale

✓ Discarding too close patches with the same scale

- Assume that pixel 1 achieves the maximum response at a certain candidate scale σ_i. At the same time, pixels 2 also achieves the maximum response at this scale.
- If pixel 1 and pixel 2 are too close, how to discard one of them?
- We can exploit non-maximum suppression (NMS). We compare their respective extrema and only save the Largest extremum.



Non-maximum suppression **at a certain scale**

Response to FAQ

➢ Understanding Blob with Scale

✓ Summary of Algorithm to detect blobs with scales in a single image

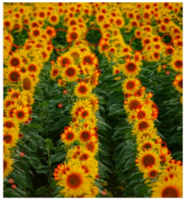
1. Build a Laplacian scale space, starting with some initial scale and going for n iterations:

1.1. Generate a (scale-normalized) Laplacian of Gaussian (LoG) filter at a given scale “sigma”.
1.2. Filter image with the LoG kernel.
1.3. Save square of Laplacian filter response for current level of scale space.
1.4. Increase scale by a factor k.

2. Perform non-maximum suppression in scale space.

3. Display resulting circles at their characteristic scales.

Input


Result of candidate scale validation and NMS

Response to FAQ

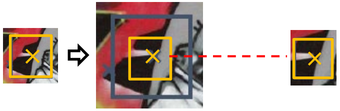
See also the previous slide “Straightforward scale search” vs. “automatic scale determination”

➢ Understanding Blob with Scale

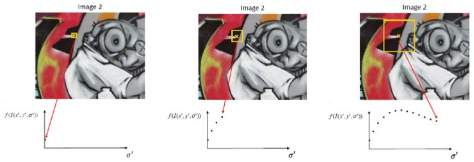
✓ Clarification of a statement about “known” scale (introduced in the last class)

- My statement: Before matching, we have known the scale of each patch a priori. So, we **do NOT need to try several candidate scales** **when matching** (like straightforward method with complexity N^2S).
- My original meaning: We can first assume that scales of both left and right patch has been obtained. So, we can directly rescale patches and evaluate their difference. (I conceal the detail of scale determination on purpose).

- However, to determine the scale **before matching**, we **still have to try a set of candidate scales.** So, what is the total complexity?



Determine scale **when** matching

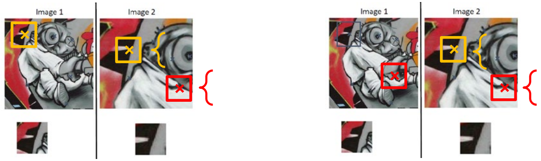


Scale determination **before** matching

Recap on Patch Descriptor-based Method

➢ Scale of Patch Descriptor

✓ "Straightforward scale search" vs. "automatic scale determination"

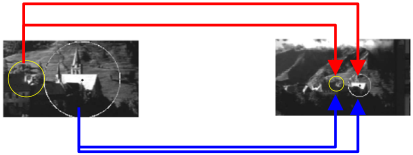


Simultaneously find the optimal patch correspondence and optimal scale

Complexity: $N^2 \bullet S$

Brute force matching with **straightforward** scale search (one-step method)

First step: Scale determination



Second step: brute force matching based on rescaled patches

Complexity: $N \bullet S + N \bullet S + N^2$

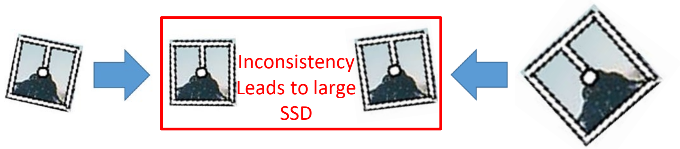
Brute force matching with **automatic** scale determination (two-step method)

Scale-invariant Feature Transform (SIFT)

- ➢ Motivation

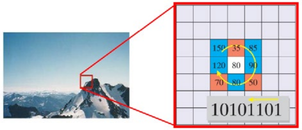
- ➢ Disadvantages of patch descriptor-based method

- First: If the warping is not estimated accurately, very small errors in rotation, scale, and view point will affect matching score based on SSD.
- An alternative strategy: census descriptor-based method (to generate a descriptor, we still need to warp patch. However, instead of directly comparing patches by SSD, we compare their associated vector descriptors—less sensitive to noise.)



Left warped patch

Right warped patch



$$NCC = \frac{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)F(u,v)}{\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)^2}\sqrt{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F(u,v)^2}}$$

1D descriptors

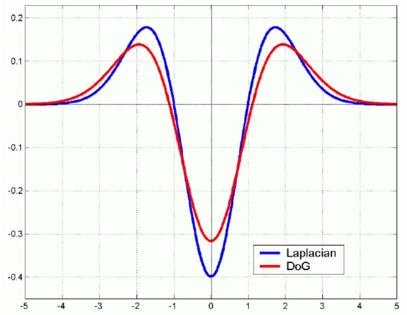
Scale-invariant Feature Transform (SIFT)

➢ Motivation

➢ Disadvantages of patch descriptor-based method

- Second: Laplacian of Gaussian (LoG) is relative inefficient.
- An alternative strategy: difference of Gaussian (DoG) kernel*

$$LOG(x,y) \approx DoG(x,y) = \underbrace{G_{k\sigma}(x,y) - G_{\sigma}(x,y)}_{\text{Gaussian at different scales}}$$



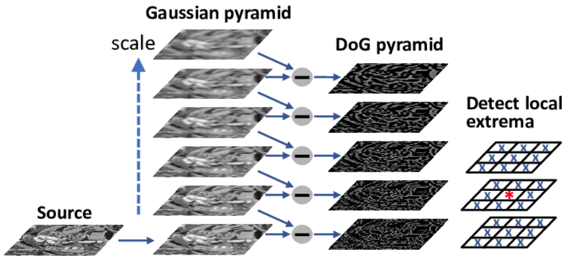
*The proof that LoG can be approximated by a difference of Gaussian comes from the Heat Equation: $\frac{\partial G_\sigma}{\partial \sigma} = \sigma \nabla^2 G_\sigma$

Such a proof will not be asked in the exam.

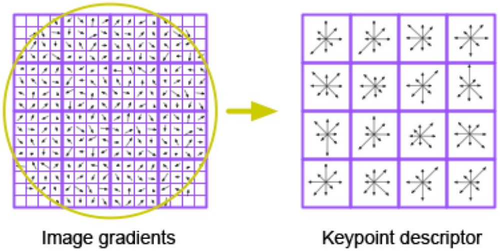
Scale-invariant Feature Transform (SIFT)

➢ Overview

✓ Step 1: Key point extraction based on extreme detection using **DoG**
✓ Step 2: **Census descriptor** assignment by Histogram of Oriented Gradients



First step (Dog instead of LoG)



Second step

Lowe, "Distinctive Image Features from Scale Invariant Keypoints", Internal Journal of Computer Vision, 2004.

Scale-invariant Feature Transform (SIFT)

➢ Key Point Extraction

✓ Image blurring based on Gaussian kernel

- This operation blurs the image, but maintains the image size.
- This operation is the **first of two steps** to generate DoG for validation of a set of candidate scales.

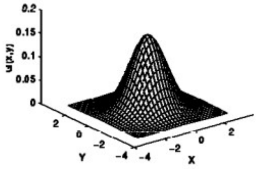
First step of DoG generation

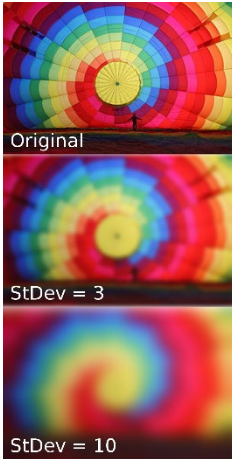
$$LOG(x,y) \approx DoG(x,y) = G_{k\sigma}(x,y) - G_{\sigma}(x,y)$$

Second step of DoG generation

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

A graphical representation of the 2D Gaussian distribution with mean(0,0) and σ = 1 is shown to the right.



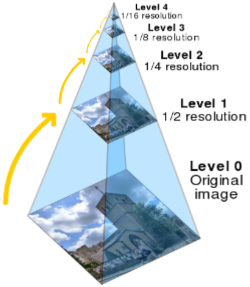


Scale-invariant Feature Transform (SIFT)

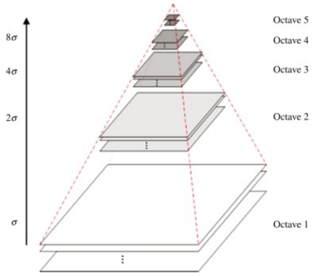
➢ Key Point Extraction

✓ Image down-sampling

- This operation keeps the sharpness, but reduces the image size
- This step is **not a compulsory** step. Even if we only have a single octave, we can still detect key points. This step can be used to find more key points (at each octave, we can find some).



Down-sampled images



Space-scale pyramid

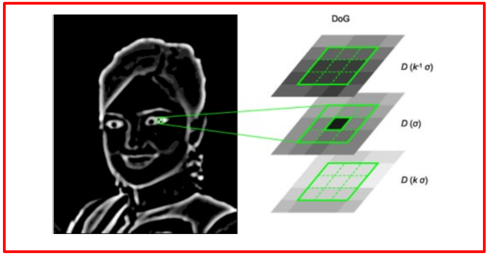
Blurred images

Scale-invariant Feature Transform (SIFT)

➢ Key Point Extraction

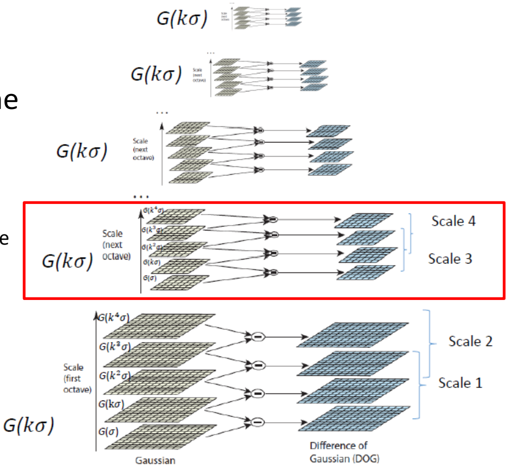
✓ Building a space-scale pyramid

Adjacent blurred images are subtracted to produce the Difference of Gaussian (DoG) images.



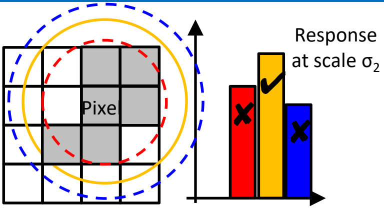
These images are similar to the images convolved by a set of LoG with different scales

We take one octave for example



Scale-invariant Feature Transform (SIFT)

➢ Key Point Extraction



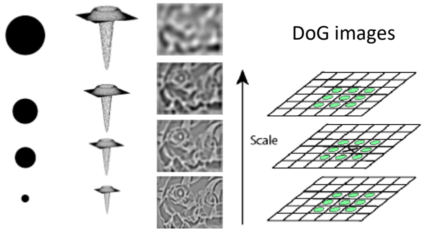
Scale determination by LoG (**change window size in a single image**)

✓ Scale space extrema detection

SIFT key points: local extrema in DoG images

- Each pixel is compared to 26 neighbors (below in green): its **8 neighbors in the current image (**NMS**)** + **9 neighbors** in the adjacent **upper scale** + **9 neighbors** in the adjacent **lower scale** (9+9: local extremum at different scales—**keep window size unchanged on multiple images**)
- If the pixel is **an extremum with respect to its 26 neighbors** then it is selected as SIFT **feature**.

We take one octave for example



For each extrema, the output of the SIFT detector is the **location** (x, y) and the **scale** s

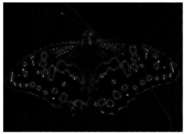
Scale-invariant Feature Transform (SIFT)

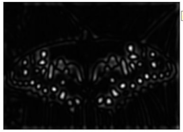
➢ Key Point Extraction

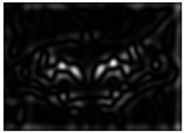
✓ Representative result

Input image

DoG Images at different octaves (different resolutions)

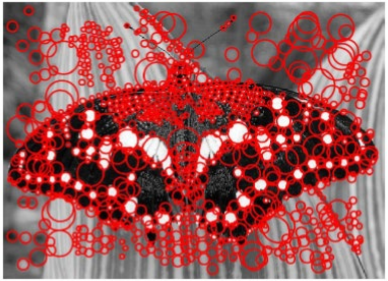

Magnitude of $(G(k\sigma) - G(\sigma))$ | $s = 4; \sigma = 1.6$ |


Magnitude of $(G(k^6\sigma) - G(k^5\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)


Magnitude of $(G(k^9\sigma) - G(k^8\sigma))$ | $s = 4; \sigma = 1.6$ |
(third octave shown at the input resolution for convenience)

Multiple octaves can be used to find more key points (at each octave, we can find some).

Size of circle represents scale

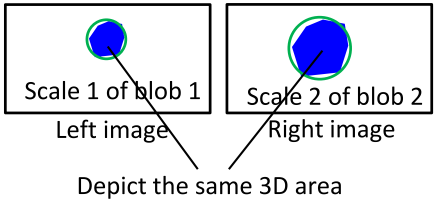

Local extrema of DoG images across **Scale** (different σ of Gaussian) and **Space** (different resolutions)

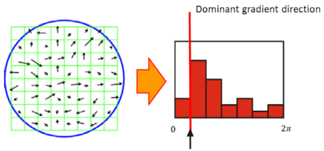
Scale-invariant Feature Transform (SIFT)

➢ Descriptor Computation

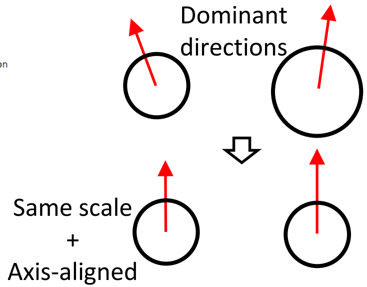
✓ Preprocessing step

- For a blob, consider a circular region around it. The radius of circle is determined by scale of this blob. (**scale-invariant**)
- Compute dominant orientation, and de-rotate the patch. (**rotation-invariant**)
- Reason for **adaptive circle radius** and **de-rotation**: regardless of type of descriptor (patch or census), only **two axis-aligned patches with the same scale** can be compared.





Dominant direction obtaining

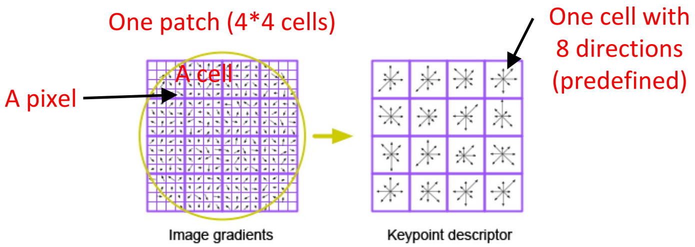


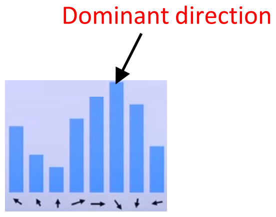
Scale-invariant Feature Transform (SIFT)

➢ Descriptor Computation

✓ Compute **histogram of oriented gradients descriptor** (census descriptor)

- Input: a de-rotated patch
- Divide patch into 4 × 4 cells
- For each cell, generate an 8-bin histograms (i.e., 8 directions)





8 directions

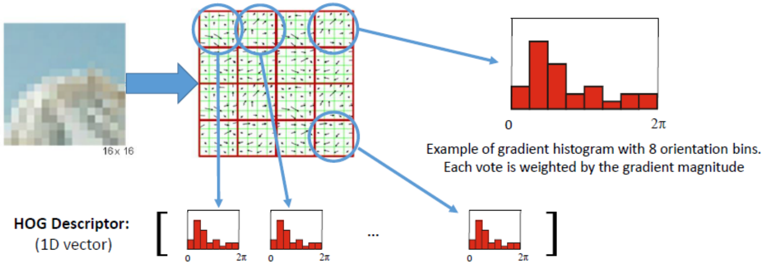
Scale-invariant Feature Transform (SIFT)

➢ Descriptor Computation

✓ Compute **histogram of oriented gradients** descriptor (census descriptor)

- Concatenate all histograms into a single 1D vector, resulting SIFT descriptor: $\underbrace{4 \times 4}_{\text{16 cells}} \times \underbrace{8}_{\text{8 bins}}$ = 128 values

16*16 pixels



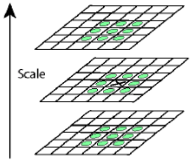
Example of gradient histogram with 8 orientation bins.
Each vote is weighted by the gradient magnitude

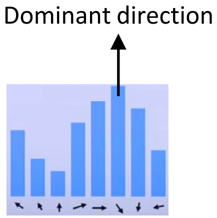
Scale-invariant Feature Transform (SIFT)

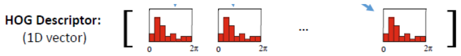
➢ Output of Algorithm

✓ Location (pixel coordinates of the center of the patch): 2D vector
✓ Scale (i.e., size) of the patch: 1 scalar value (high scale corresponds to high blur in the space scale pyramid)
✓ Orientation (dominant direction): 1 scalar value (i.e., angle of the patch)

✓ Descriptor: 4x4x8 = 128 element 1D vector



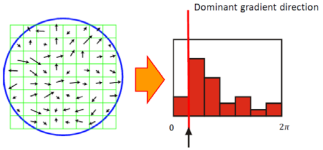




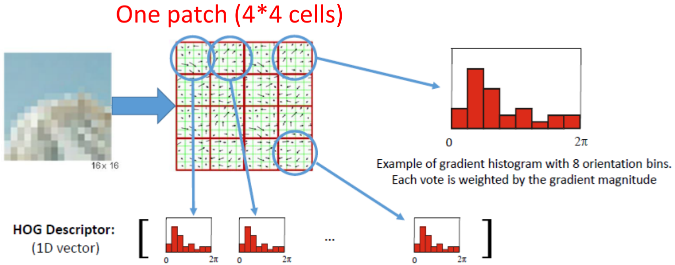
Scale-invariant Feature Transform (SIFT)

➢ Dominant Direction Determination vs. Census Descriptor Generation

✓ Dominant direction determination: pixel-level
✓ Descriptor generation: we should first de-rotate the patch based on dominant direction. Then we generate descriptors on cell-level



Pixel-level


One patch (4*4 cells)
16x 16
0
2π
Example of gradient histogram with 8 orientation bins.
Each vote is weighted by the gradient magnitude
HOG Descriptor:
(1D vector)
...

Cell-level

Other Descriptors and Detectors

➢ Overview

- ✓ FAST: **F**eatures from **A**ccelerated **S**egment **T**est
- ✓ SURF: **S**peeded **U**p **R**obust **F**eatures
- ✓ BRIEF: **B**inary **R**obust **I**ndependent **E**lementary **F**eatures
- ✓ ORB: **O**riented FAST and **R**otated **B**RIEF
- ✓ BRISK: **B**inary **R**obust **I**nvariant **S**calable **K**eypoints
- ✓ SuperPoint: Deep learning-based method

Other Descriptors and Detectors

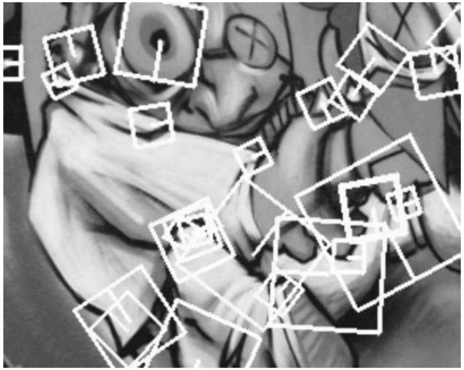
➢ “SURF” Blob Detector & Descriptor

✓ SURF: Speeded Up Robust Features

✓ Similar to SIFT but much faster

✓ Results comparable with SIFT:

- Faster computation
- Generally shorter descriptors



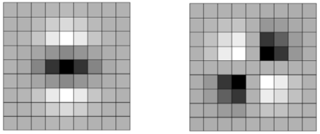
Bay, Tuytelaars, Van Gool, "Speeded Up Robust Features", European Conference on Computer Vision (ECCV), 2006.

Other Descriptors and Detectors

➢ “SURF” Blob Detector & Descriptor

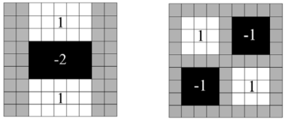
✓ Basic idea: approximate Gaussian and DoG filters using box filters

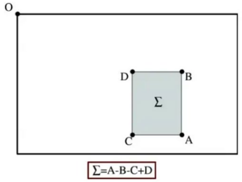
Original second order partial derivatives of a Gaussian



$\frac{\partial^2 G(x,y)}{\partial y^2}$ $\frac{\partial^2 G(x,y)}{\partial x \partial y}$

SURF Approximation using **box filters**





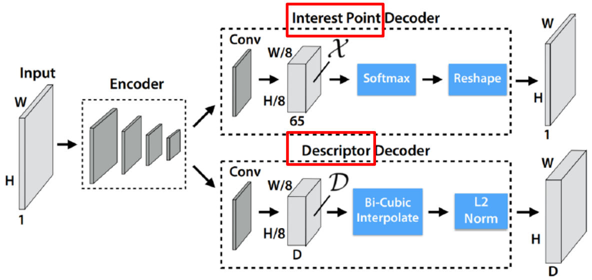
Integral image

Surf will not be asked in the exam.

Other Descriptors and Detectors

➢ “SuperPoint”: A Deep Learning-based Method

✓ Joint regression of keypoint location and descriptor.

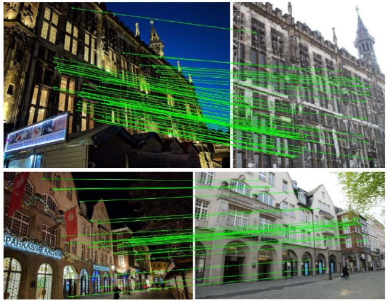


Detone , Malisiewicz , Rabinovich . SuperPoint : Self Supervised Interest Point Detection and Description. CVPRW 2018.

Other Descriptors and Detectors

➢ "SuperPoint": A Deep Learning-based Method

✓ Trained on synthetic images and refined on homographies of real images.
✓ Detector is less accurate than SIFT, but descriptor outperforms SIFT.
✓ For efficiency, it is slower than SIFT.



Summary

- Recap on Feature Matching Problem
- Recap on Patch Descriptor-based Method
- Response to Frequently-asked Questions
- A More Effective Method: SIFT
- Other Methods

Thank you for your listening!

If you have any questions, please come to me :-)