SIFT Descriptor

- **S**cale **I**nvariant **F**eature **T**ransform
- Invented by David Lowe in 2004

David Lowe

FOLLOW

Professor Emeritus, Computer Science Dept., University of British Columbia
Verified email at cs.ubc.ca - Homepage

Computer Vision Object Recognition

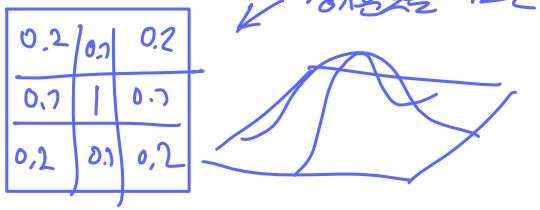
TITLE	CITED BY	YEAR
Distinctive image features from scale-invariant keypoints DG Lowe International journal of computer vision 60 (2), 91-110	68297	2004

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

SIFT Descriptor

Descriptor computation:

- Consider a $\mathbf{16 \times 16}$ **pixel patch**
- Multiply the patch by a **Gaussian filter,** compute **dominant orientation,** and **de-rotate patch**
- Compute **HOG descriptor**
  - Divide patch into **4×4 cells**
  - Use **8 bin histograms** (, i.e., 8 directions)
  - **Concatenate all histograms** into a single 1D vector
  - Resulting SIFT descriptor: **4×4×8 = 128 values**
- Descriptor Matching: SSD (i.e., Euclidean-distance)
- Why 4×4 cells and why 8 bins? See later



compute similarity b/w local patches.

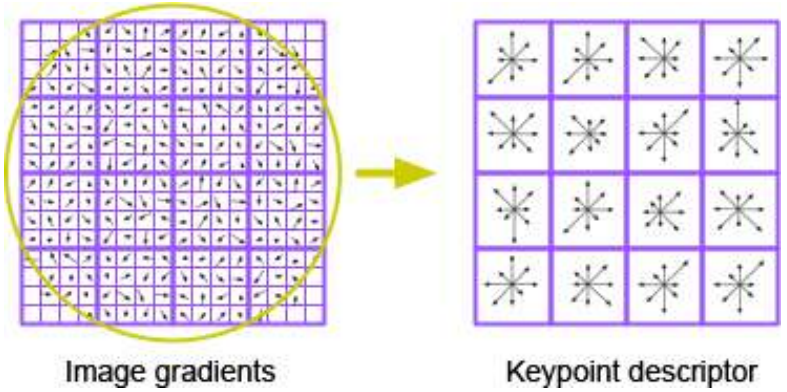


Image gradients

Keypoint descriptor

Descriptor Normalization

- The HOG descriptor is already **invariant to additive illumination** because it is based on gradients
  - Adding the same pixel offset to the local patch gives the same gradients
- To make it **invariant affine illumination changes**, the descriptor vector $\boldsymbol{v}$ is then normalized such that its L_2 norm is 1:
  - High-contrast photos could linearly increase gradients -> make the descriptor invariant to such changes by normalizing it

$$\overline{\boldsymbol{v}} = \frac{\boldsymbol{v}}{\sqrt{\sum_i^n v_i^2}}$$

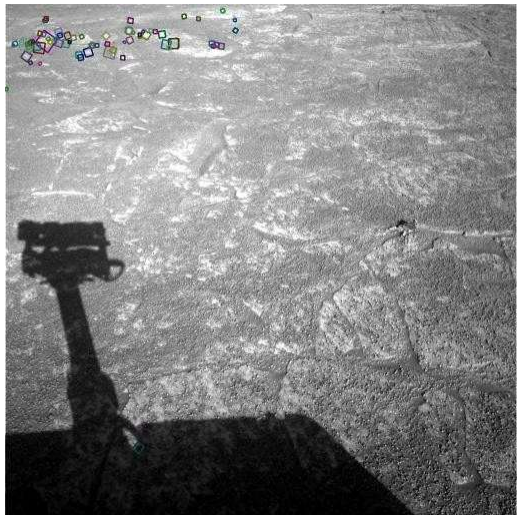
$\alpha I + \beta$

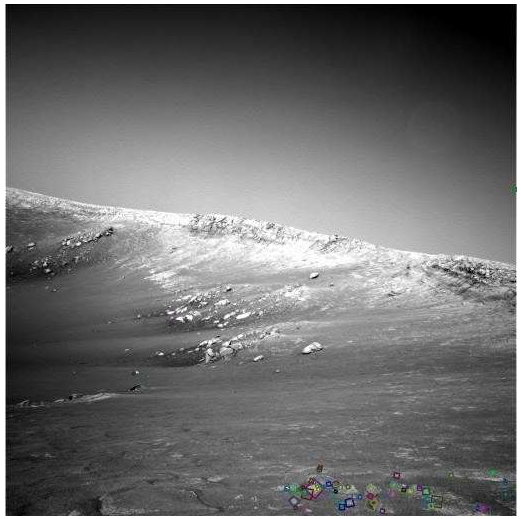
- We can conclude that the **SIFT descriptor is invariant to affine illumination changes**

SIFT Matching Robustness

based on experimental result.

- Can handle severe **viewpoint changes** (**up to 50 degree** out-of-plane rotation)
- Can handle even **severe non affine changes in illumination** (low to bright scenes)
- Original SIFT binary files: http://people.cs.ubc.ca/~lowe/keypoints
- OpenCV C/C++ implementation: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html



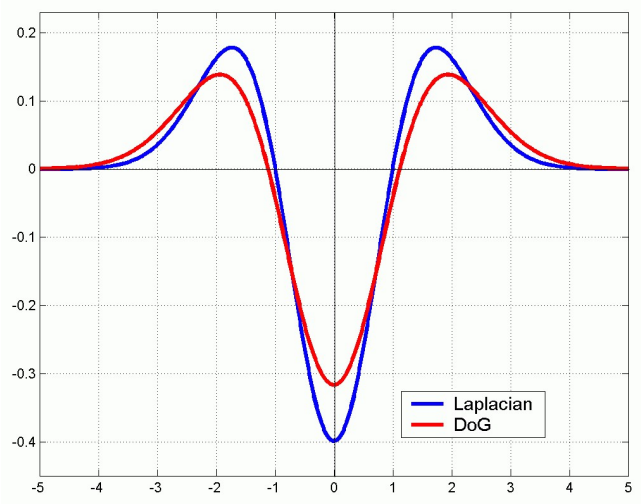


SIFT Detector

- SIFT uses the **Difference of Gaussian (DoG) kernel** instead of Laplacian of Gaussian (LoG) because computationally cheaper

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

only pixel-wise difference

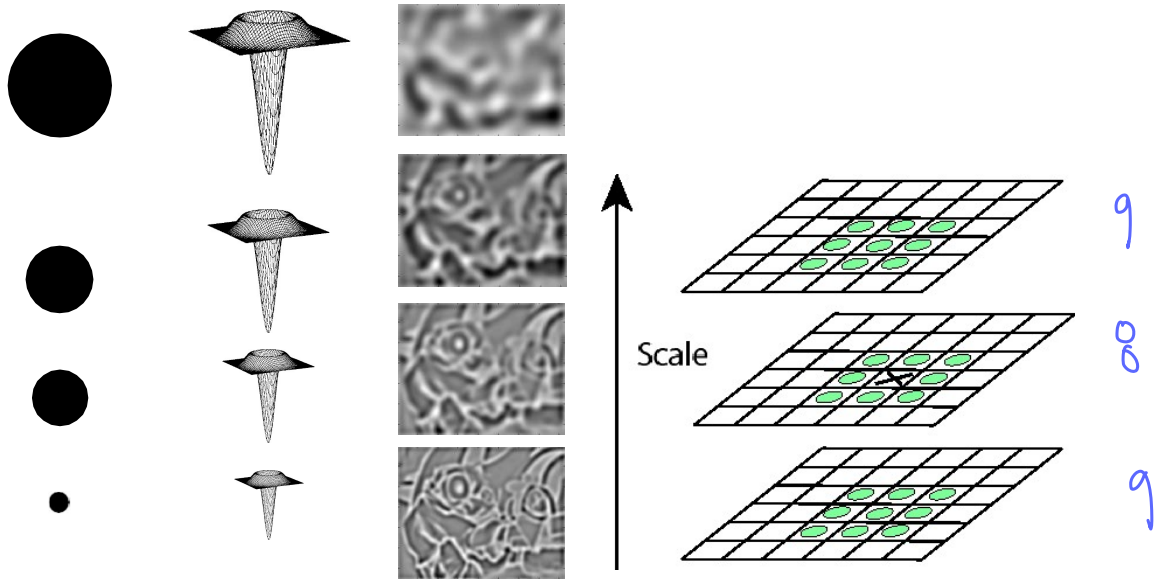


- The proof that LoG can be approximated by a difference of Gaussian comes from the Heat Equation: $\frac{\partial G_\sigma}{\partial \sigma} = \sigma \nabla^2 G_\sigma$

SIFT Detector (location + scale)

SIFT keypoints: **local extrema** in **both space and scale** of the **DoG images**

- **Each pixel is compared to 26 neighbors** (below in green): its 8 neighbors in the current image + 9 neighbors in the adjacent upper scale + 9 neighbors in the adjacent lower scale
- If the pixel is a global maximum or minimum (i.e., extrema) with respect to its 26 neighbors then it is selected as SIFT feature

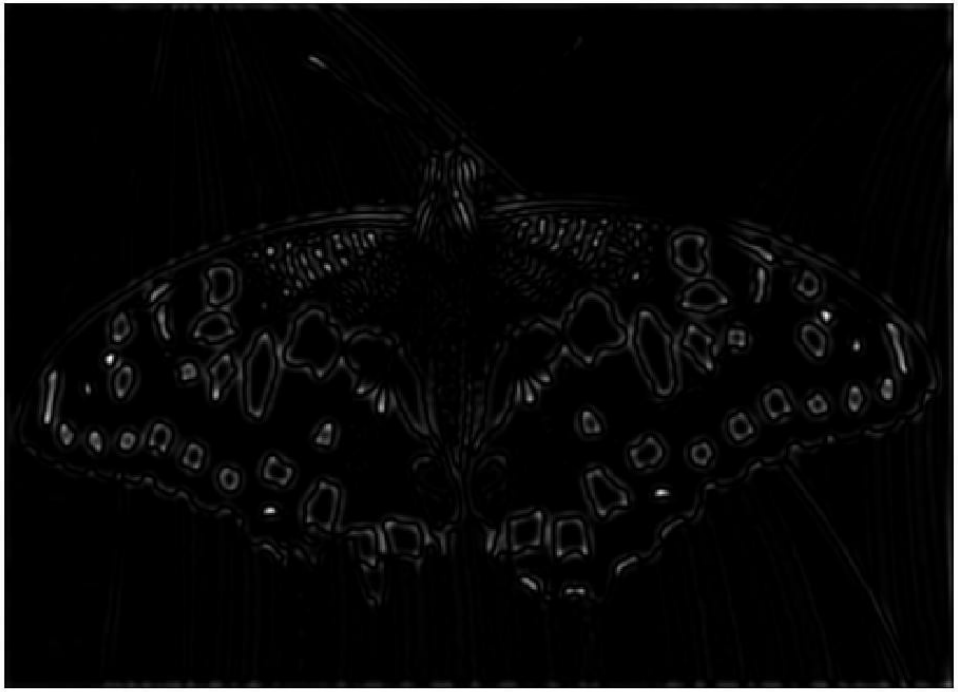


For each extrema, the output of the SIFT detector is the **location** (x, y) and the **scale** s

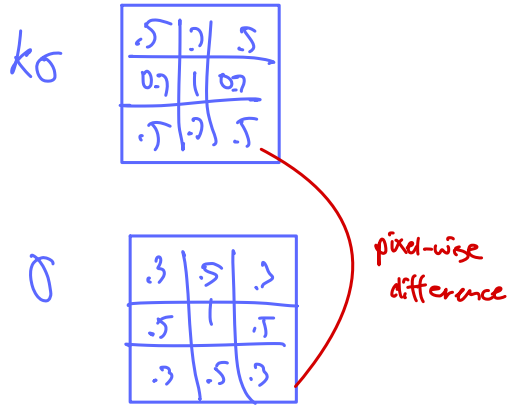
+ rotation θ

Example

DoG Images example



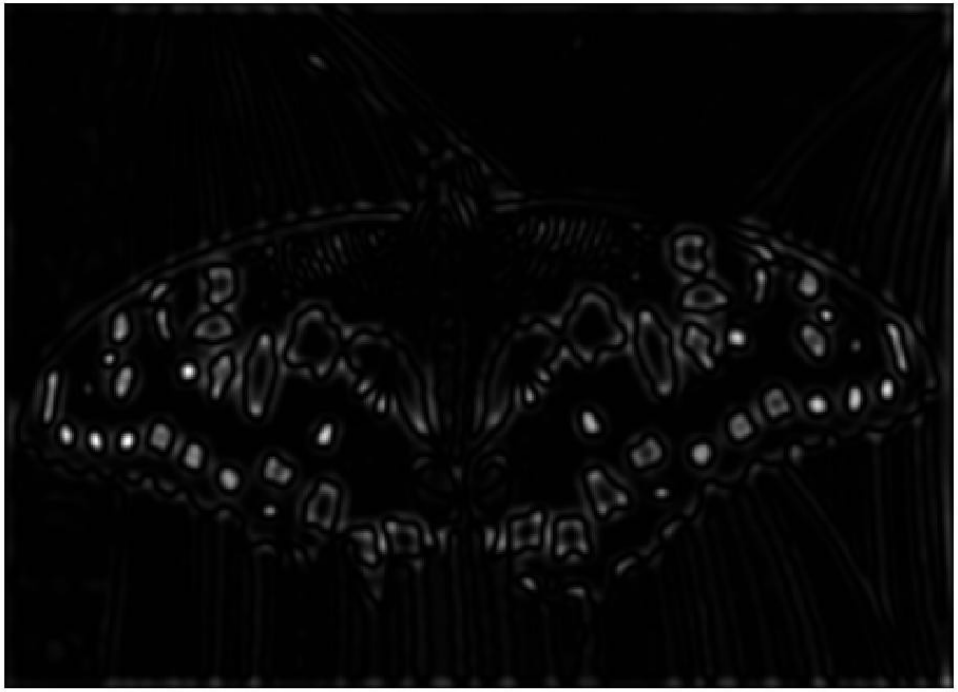
Magnitude of $(G(k\sigma) - G(\sigma))$ | $s = 4; \sigma = 1.6$ |



DoG Images example

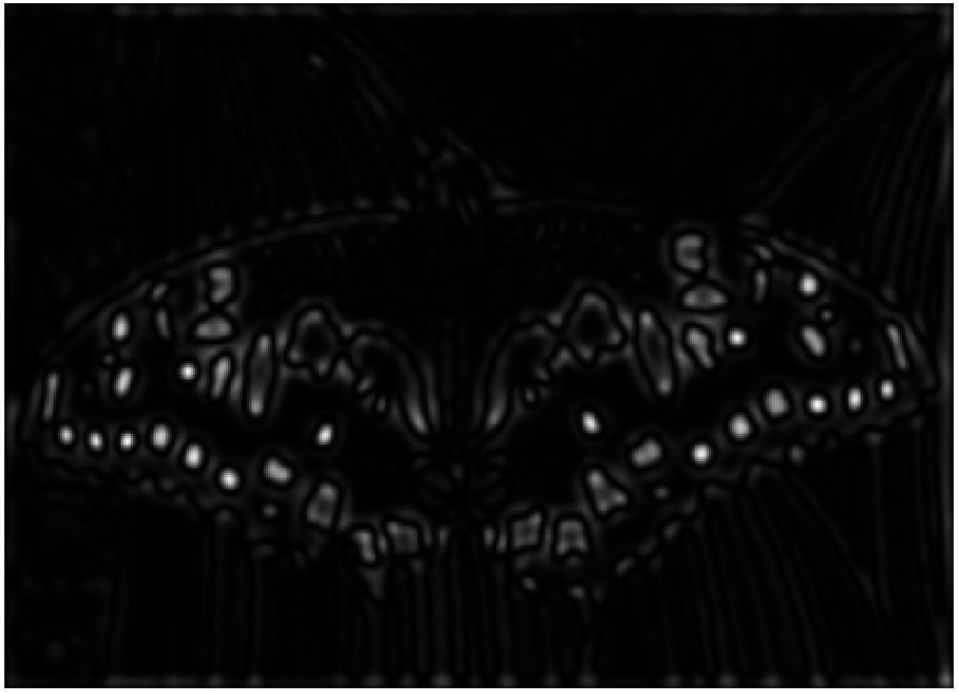
Magnitude of $(G(k^2\sigma) - G(k\sigma))|\ s = 4; \sigma = 1.6\ |$

DoG Images example



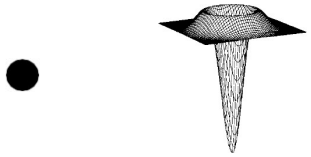
Magnitude of $(G(k^3\sigma) - G(k^2\sigma))|\ s = 4; \sigma = 1.6\ |$

DoG Images example



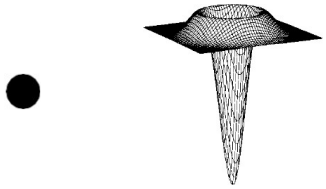
Magnitude of $(G(k^4\sigma) - G(k^3\sigma))$ | $s = 4; \sigma = 1.6$ |

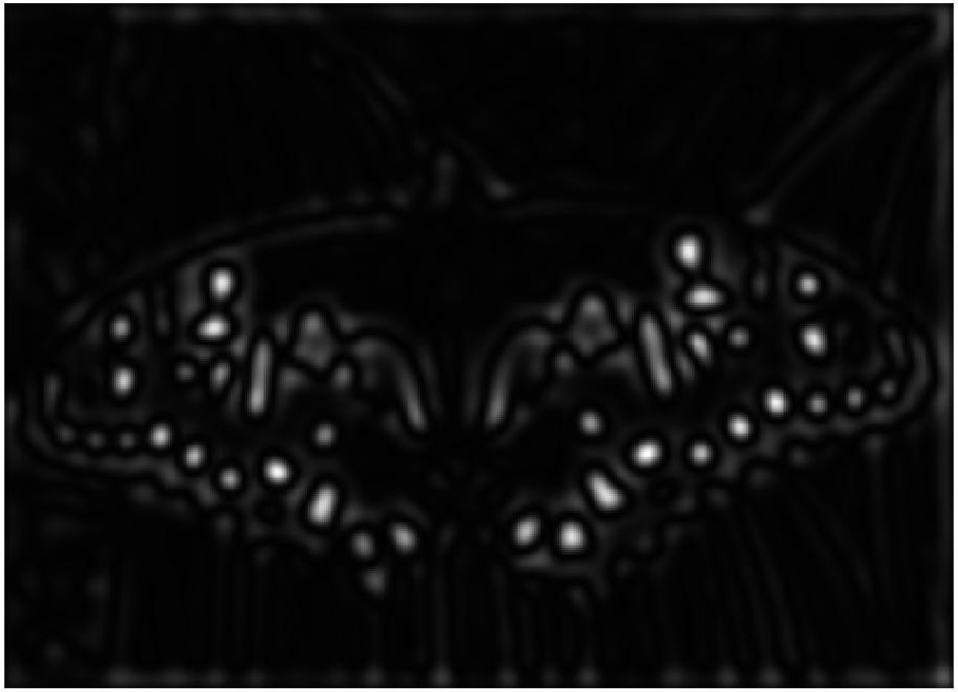
DoG Images example



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

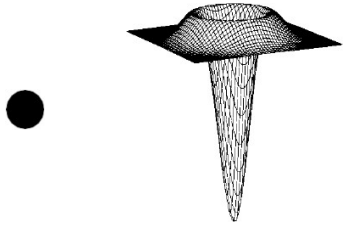
DoG Images example





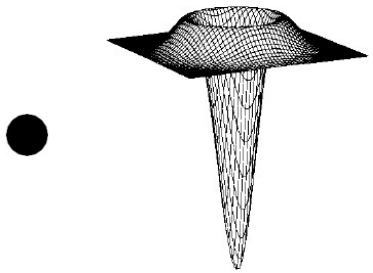
Magnitude of $(G(k^6\sigma) - G(k^5\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example



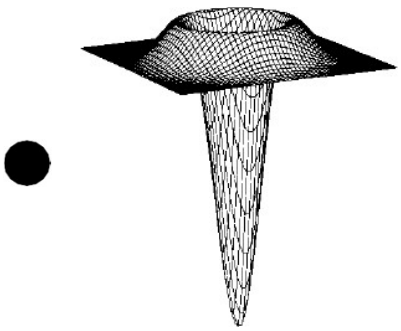
Magnitude of $(G(k^7\sigma) - G(k^6\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example



Magnitude of $(G(k^8\sigma) - G(k^7\sigma))$ | $s = 4; \sigma = 1.6$ |
(second octave shown at the input resolution for convenience)

DoG Images example

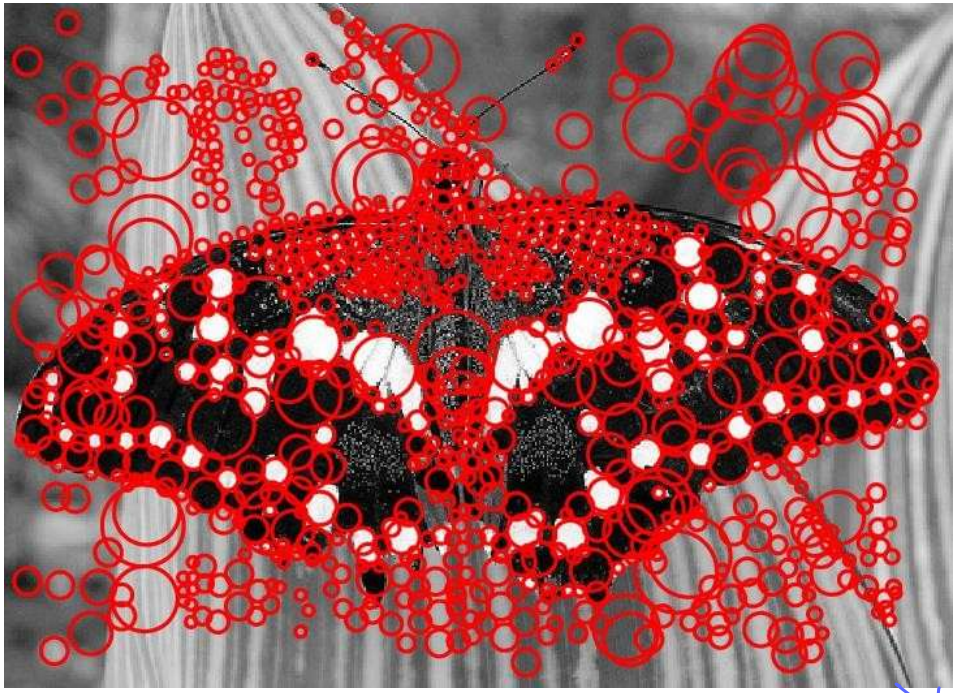


Magnitude of $(G(k^9\sigma) - G(k^8\sigma))$ | $s = 4; \sigma = 1.6$ |
(third octave shown at the input resolution for convenience)

Scale 커지면 detail pattern을 찾을 수 없게됨.

Local extrema of DoG images across Scale and Space

- Some *distinct* points can be extracted from DoG (differences of Gaussians)
    - Local patches without distinct textures should not have big differences
- For the visualization, draw a circle at the position of the local extrema where the radius of the circle is from selected scale (dominant rotations are not included for the visualization)



Center of circle
= position (x, y)
radius : σ
↑
std of Gaussian filter
+
rotation : θ

automatically captured local features.

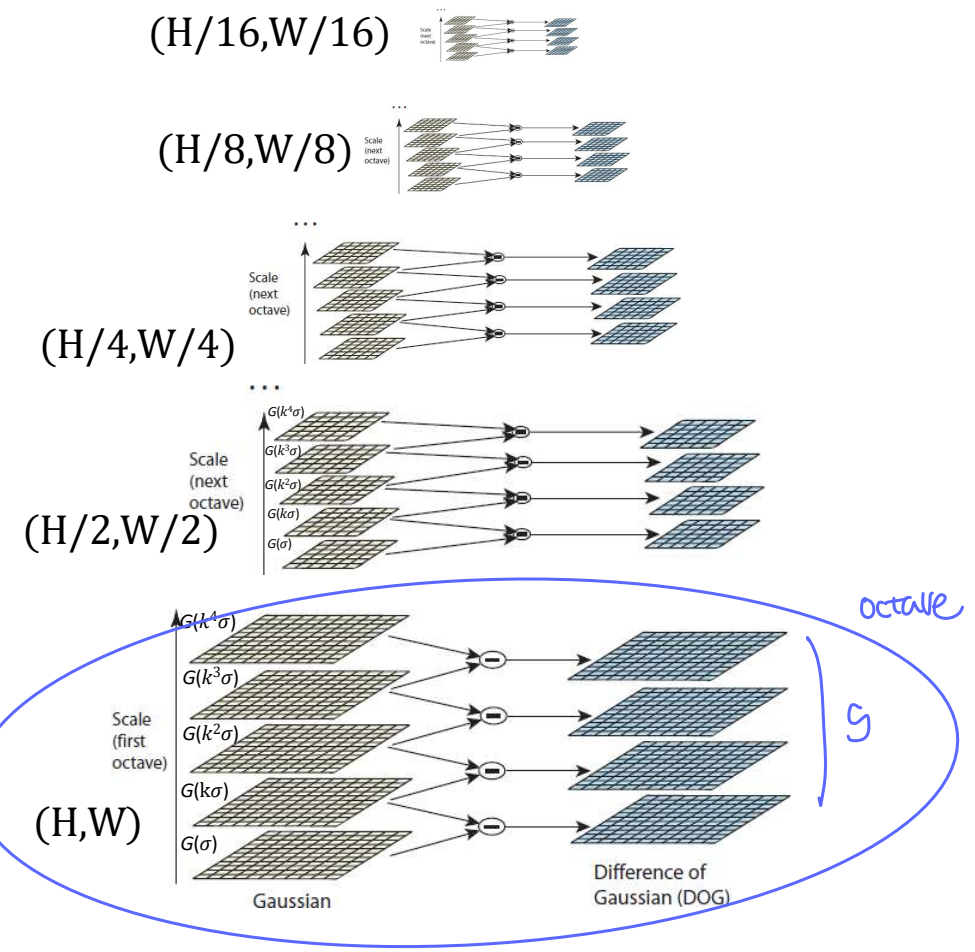
How it is implemented in practice

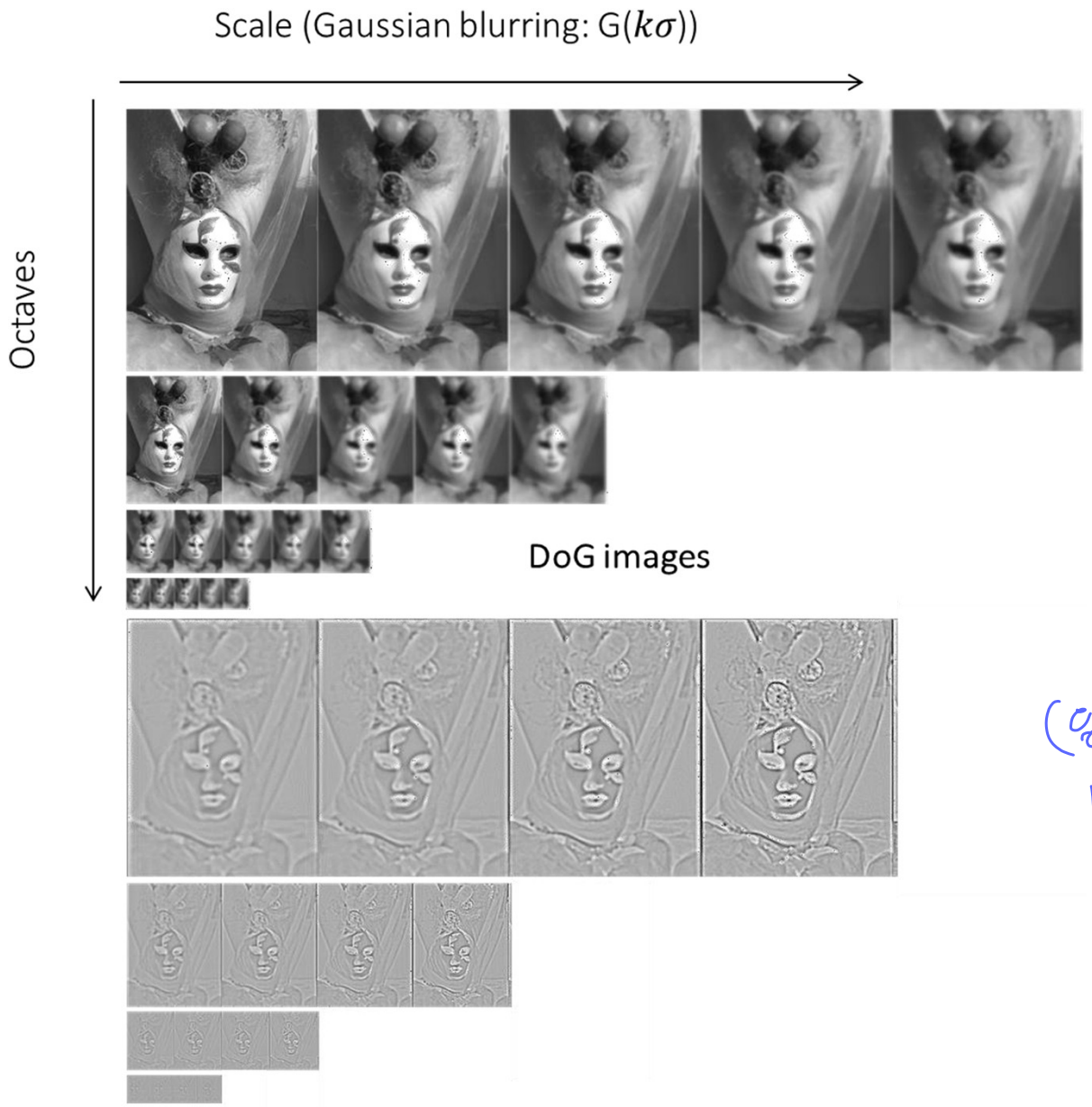
1. Build a Space-Scale Pyramid:

- The initial image is incrementally convolved with Gaussians $G(k^i\sigma)$ to produce blurred images separated by a constant factor k in scale space (shown stacked in the left column).
  - The initial Gaussian G(σ) has σ=1.6
  - k is chosen: $k = 2^{1/s}$, where s is the number of intervals into which each octave of scale space is divided
  - Each octave consists of s images, blurred with different stds.
- Adjacent blurred images are then subtracted to produce the **Difference-of-Gaussian (DoG)** images

2. Scale-Space extrema detection

- Detect local maxima and minima in space-scales





Images in same octave have same resolutions.

LoG를 계산이 빠르다 (양사이는) 그래서 SIFT는 DoG를 쓸 수 밖에 없었음.

SIFT: Recap

try to find some distinctive points.

- SIFT: **Scale Invariant** Feature Transform
- An approach to **detect and describe** regions of interest in an image.
  - SIFT detector = **DoG detector**
- SIFT features are **invariant to 2D rotation**, and reasonably invariant to **rescaling**, **viewpoint changes** (up to 50 degrees), and **illumination**

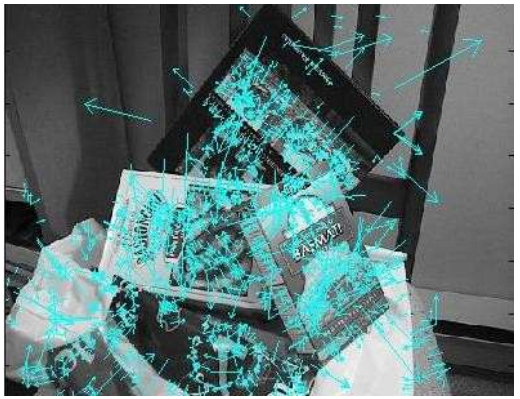
empirical finding.

- It runs in real-time but **expensive**
  - The expensive steps are the **scale detection and descriptor extraction**

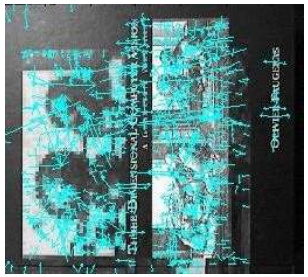
당시는 CPU로 계산했기 때문.

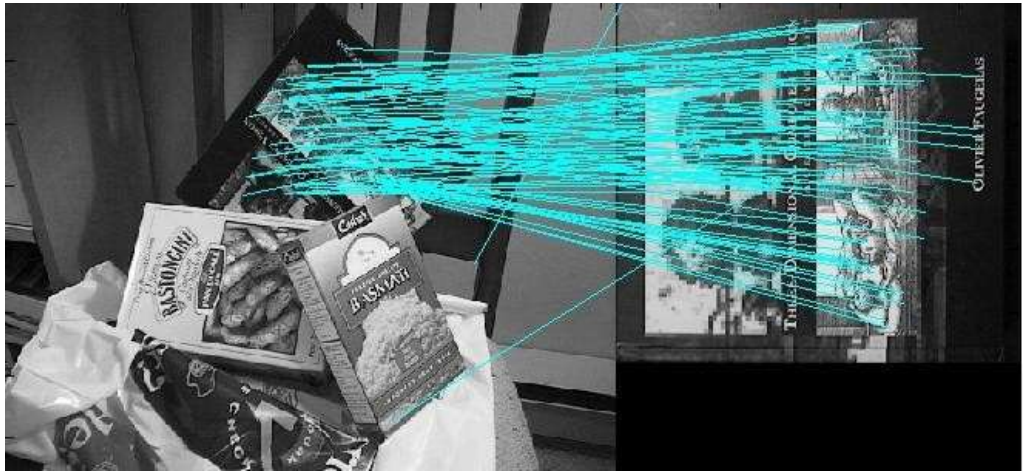
Original SIFT Demo by David Lowe

Download original SIFT binaries and Matlab function from :
http://people.cs.ubc.ca/~lowe/keypoints



```
>>[image1, descriptor1s, locs1] = sift('scene.pgm');
>>showkeys(image1, locs1);
>>[image2, descriptors2, locs2] = sift('book.pgm');
>>showkeys(image2, locs2);
>>match('scene.pgm','book.pgm');
```





What's the output of SIFT?

↗ information of local patch

128 + 2 + 1 + 1 → 132 values.

- **Descriptor**: 4x4x8 = 128-element 1D vector
- **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** (i.e., angle of the patch): 1 scalar value (dominant rotation from the rotation histogram)

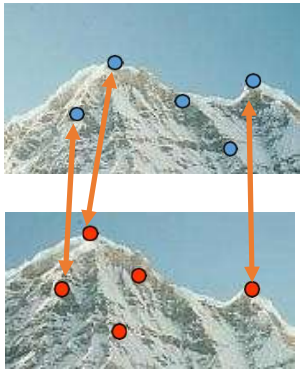
Histogram of Gradient : HoG

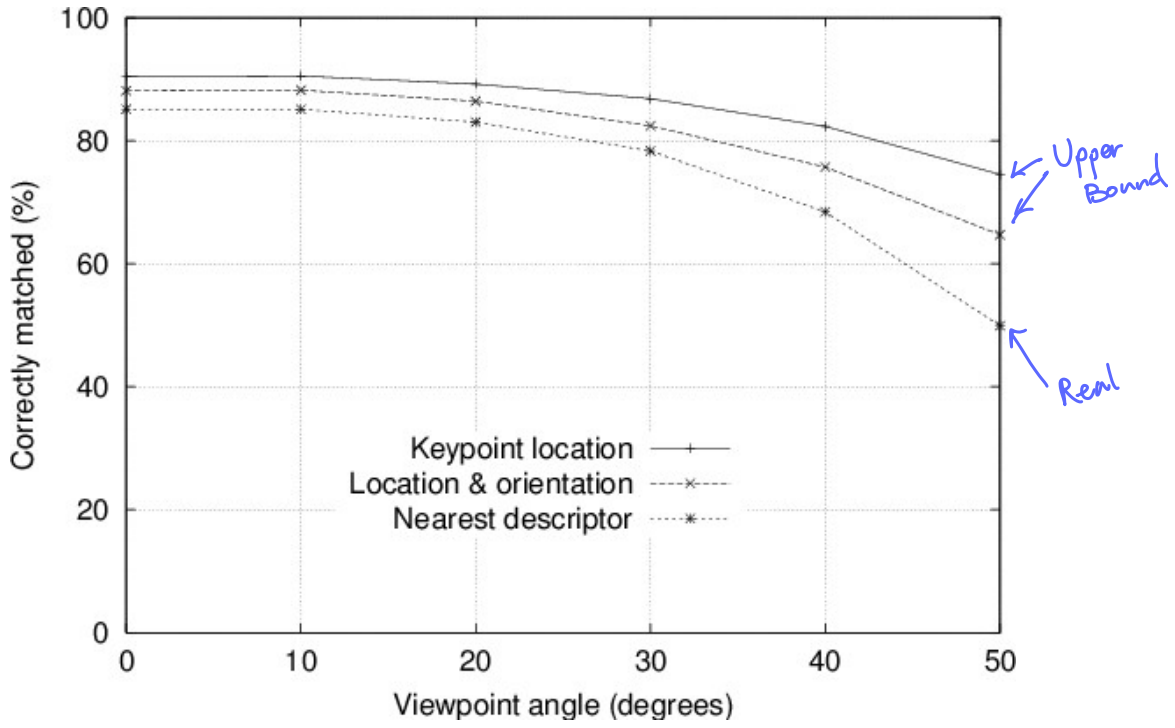
SIFT Repeatability with Viewpoint Changes

- **Keypoint location**: Evaluates only whether the same spatial position is detected (requires ground truth, not usable in practice).
- **Location + orientation**: Adds dominant orientation consistency, providing rotation invariance (requires ground truth, not usable in practice).
- **Nearest descriptor**: Uses the full 128-D descriptor for matching, enabling correspondence without ground truth.
- High repeatability of *location* alone reflects an easier condition, but only *descriptors* make real matching feasible.

Repeatability=

$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$





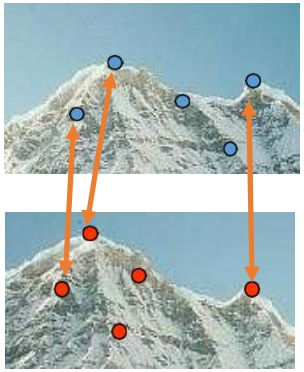
Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", Internal Journal of Computer Vision, 2004. PDF

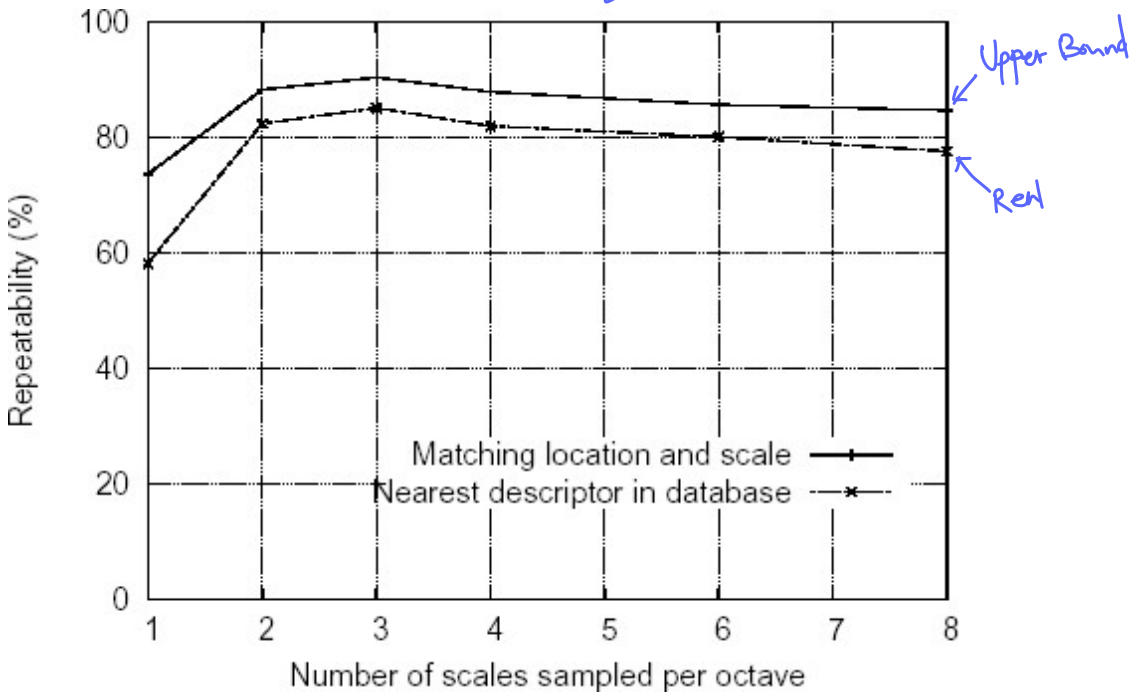
SIFT Repeatability with Number of Scales per Octave

- **Matching location & scale**: Requires ground truth correspondences, used only for detector evaluation.
- **Nearest descriptor**: Does not require ground truth, reflects practical matching performance.

Repeatability=

$$\frac{\text{\# correspondences detected}}{\text{\# correspondences present}}$$





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

Influence of Number of Orientations and Number of Sub-patches

The graph shows that a single orientation histogram (n = 1) is very poor at discriminating.
The results improve with a 4x4 array of histograms with 8 orientations.

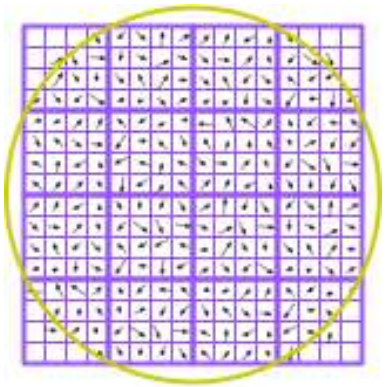
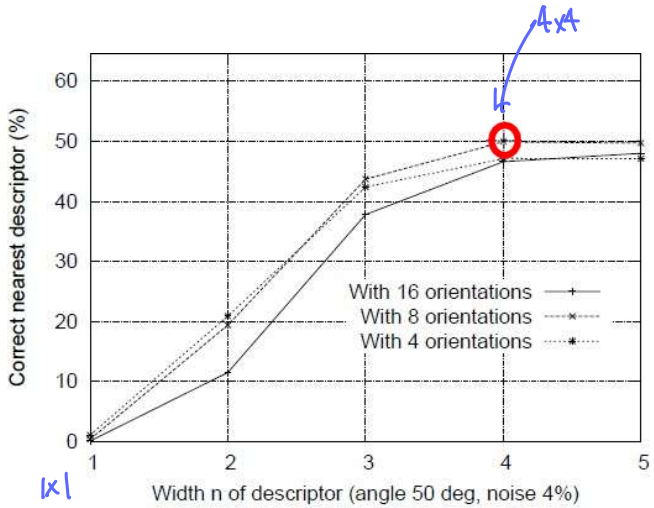


Image gradients

4x4 HOGs



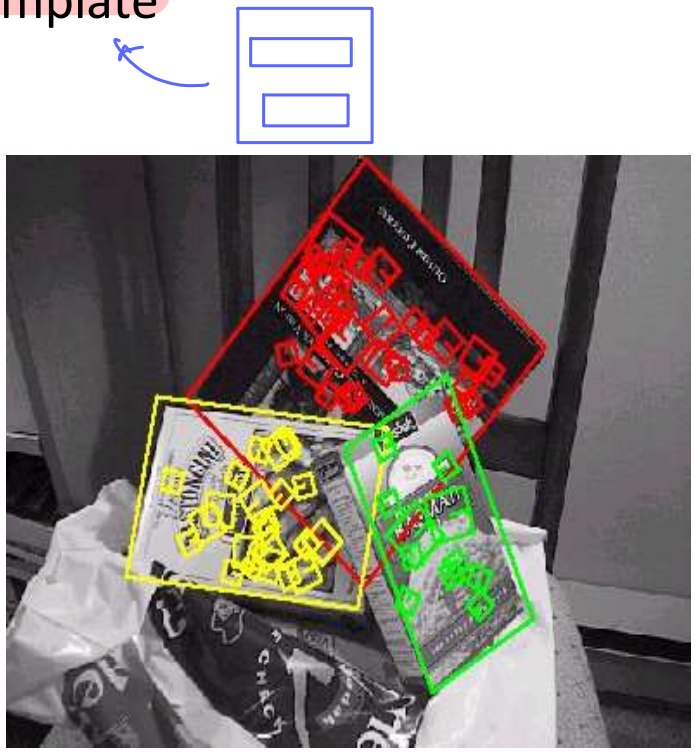
4, 4, 8 로 정해진 이유?
실험결과 가장 Good!

Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and addition of 4% noise.

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

Application of SIFT to Object recognition

- Can be implemented easily by returning object with the largest number of correspondences with the template



Application of SIFT to Panorama Stitching

AutoStitch: http://matthewalunbrown.com/autostitch/autostitch.html
M. Brown and D. G. Lowe. Recognising Panoramas, International Conference on Computer Vision (ICCV), 2003. PDF.

Feature Matching

For each point, how to **match** its **corresponding point** in the other image?

- **Brute-force Matching: compare each feature** descriptor **of Image 1 against** the descriptor of **each feature in Image 2** and assign as correspondence the feature with **closest descriptor** (e.g., minimum of SSD). If each image contains N features, we need to perform N^2 **comparisons**.

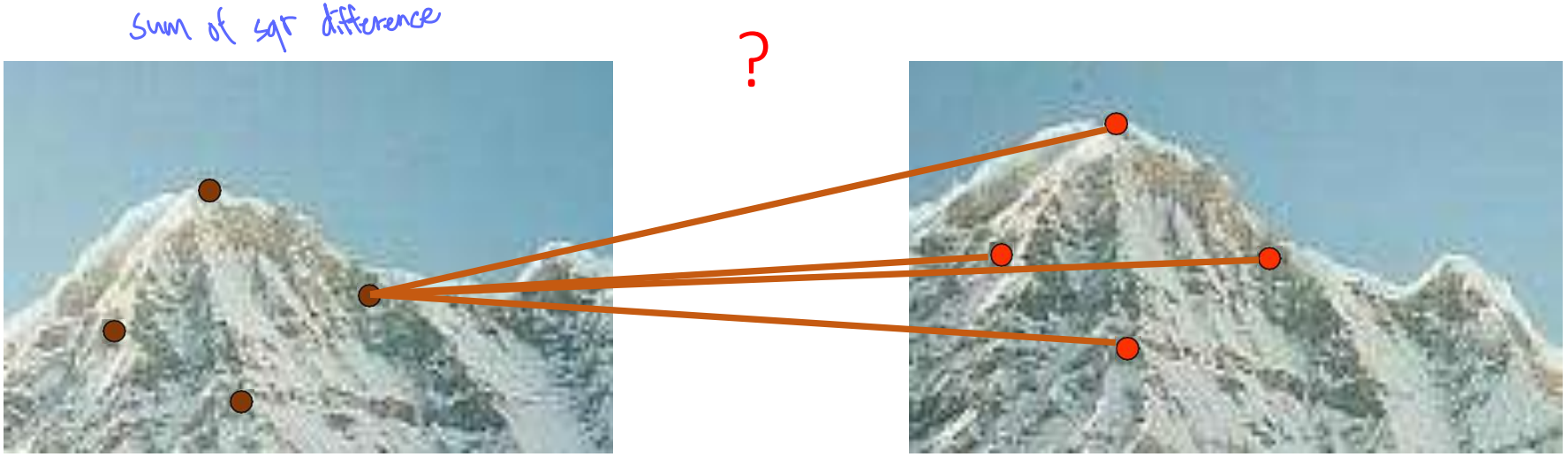
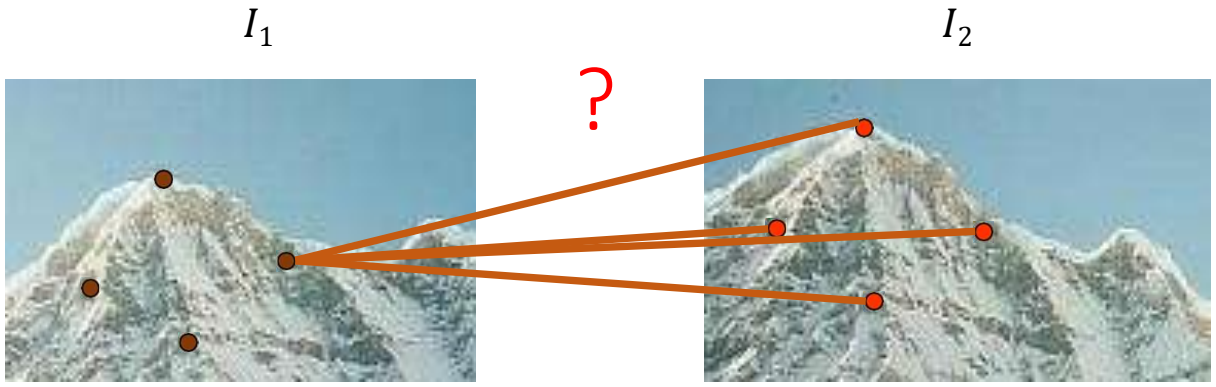


Image 1

Image 2

Feature Matching

- Given a feature in I_1, how to find the best match in I_2?
  1. **Define distance function** that compares two descriptors ((Z)SSD, (Z)SAD, (Z)NCC or Hamming distance for binary descriptors (e.g., Census, HOG, ORB, BRIEF, BRISK, FREAK)
  2. **Brute-force matching:**
     1. 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)
     2. Take the one at minimum distance, i.e., the **closest descriptor**



Distance Ratio

- In SIFT, the nearest neighbor is the descriptor with the smallest Euclidean distance.
- To avoid false matches, compare the closest and second-closest distances.
- A match is kept only if the closest is significantly better (distance ratio < threshold).
- This reduces false matches caused by feature ambiguity and high-dimensional noise.

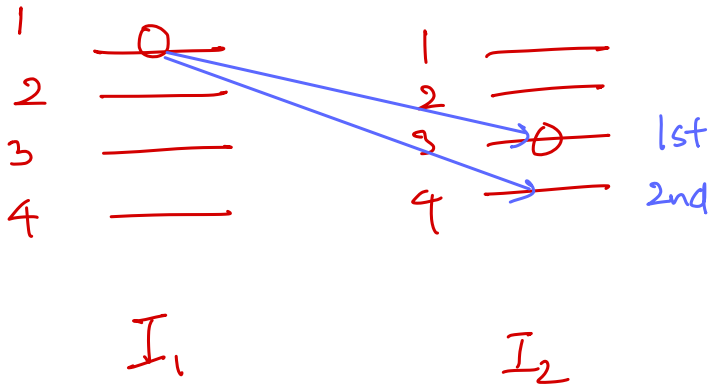
Distance Ratio

- **Issues with closest descriptor**: can occasionally return good scores for false matches
- **Better approach**: compute ratio of distances to 1st to 2nd closest descriptor

$$\frac{d_1}{d_2} < Threshold\ (usually\ 0.8)$$

where:

d_1 is the distance from the closest descriptor
d_2 is the distance of the 2nd closest descriptor



SIFT Feature Matching: Distance Ratio

The SIFT paper recommends to use a threshold on 0.8. Where does this come from?

*"**A threshold of 0.8 eliminates 90% of the incorrect matches** **while discarding less than 5% of the correct matches.**"*

"This figure was generated by matching images following random scale and orientation change, with viewpoint change of 30 degrees, and addition of 2% image noise, against a database of 40,000 keypoints."

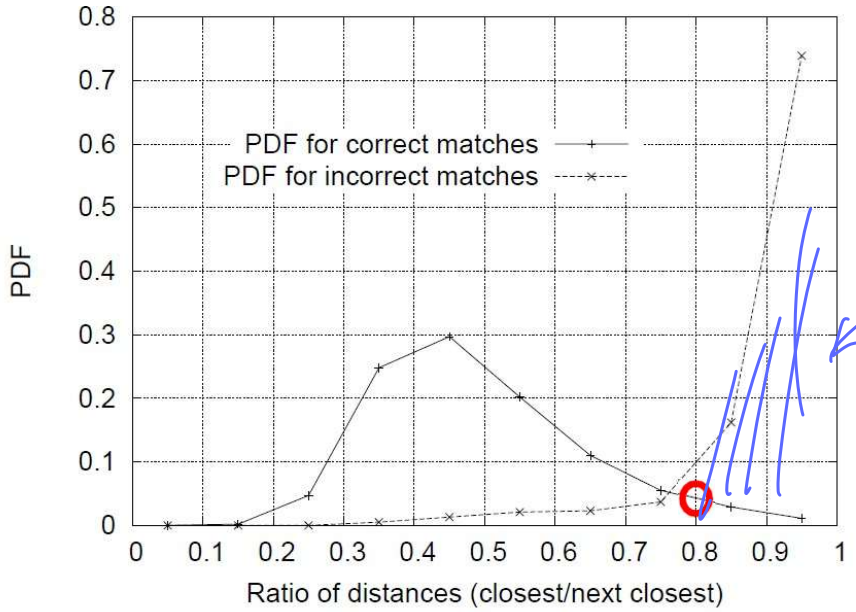
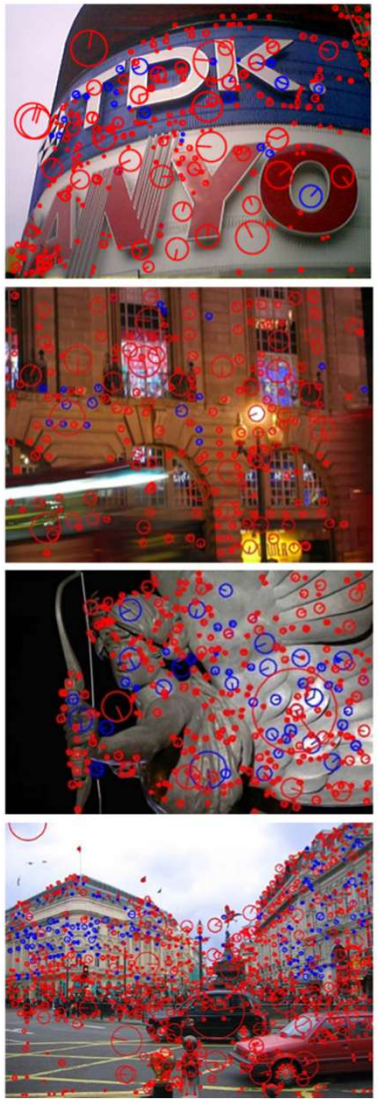
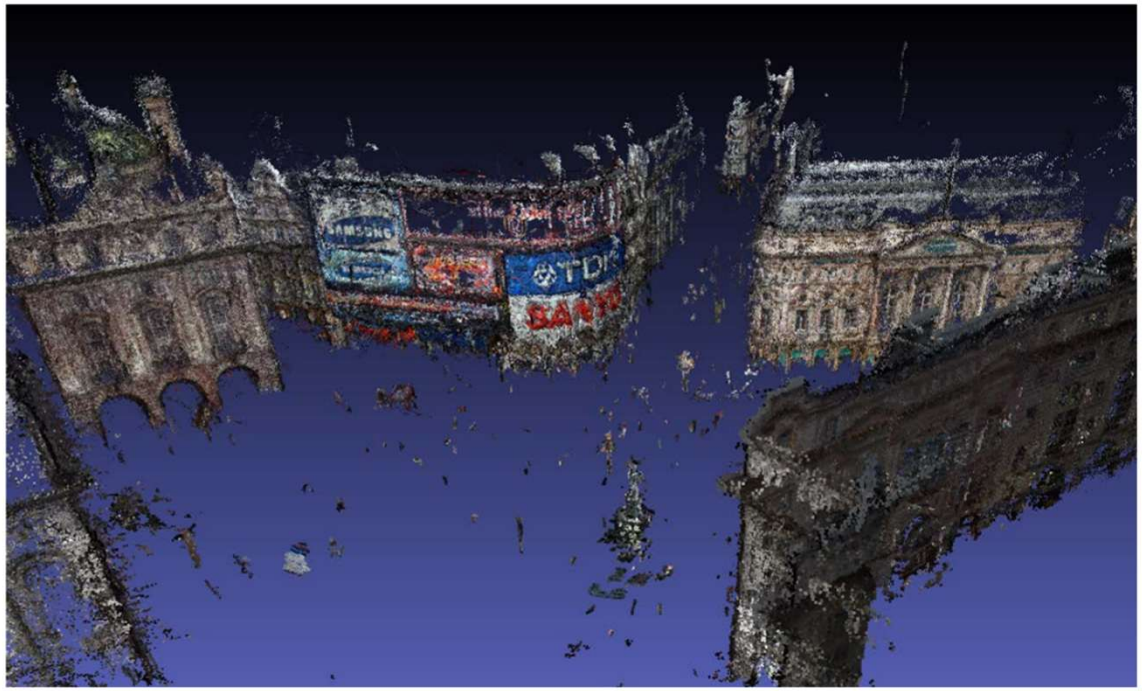


Figure 11: The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the PDF of this ratio for correct matches, while the dotted line is for matches that were incorrect.

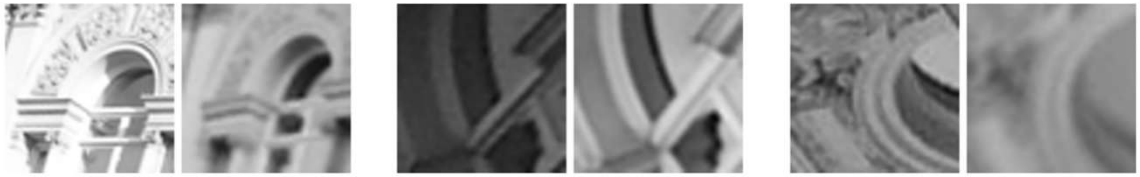
Applications: Structure-from-Motion (SfM)



Training set #1:



3k images, 59k unique points, 380k

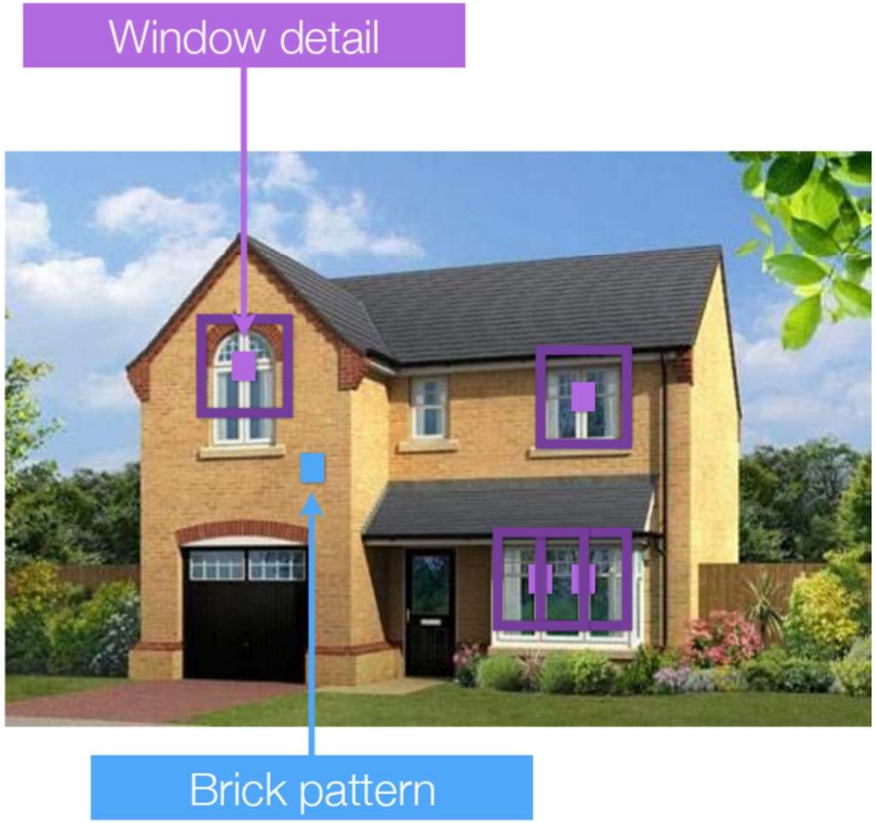


Failure cases (Lack of global context)

Repetitive structures cause problems for feature matching

Multiple locations in an image provide good matches and have similar matching scores

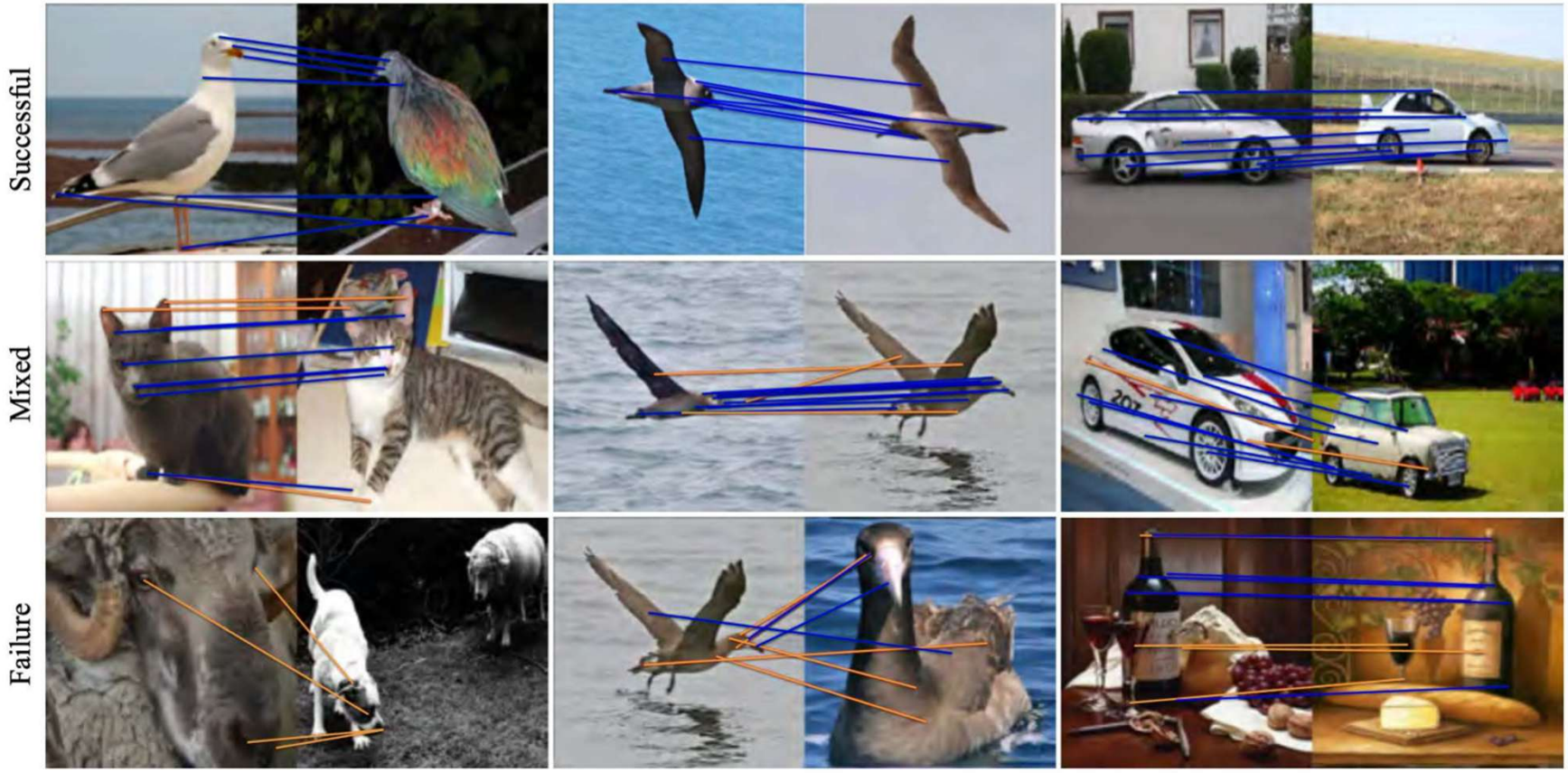
They are particularly common in man-made environments



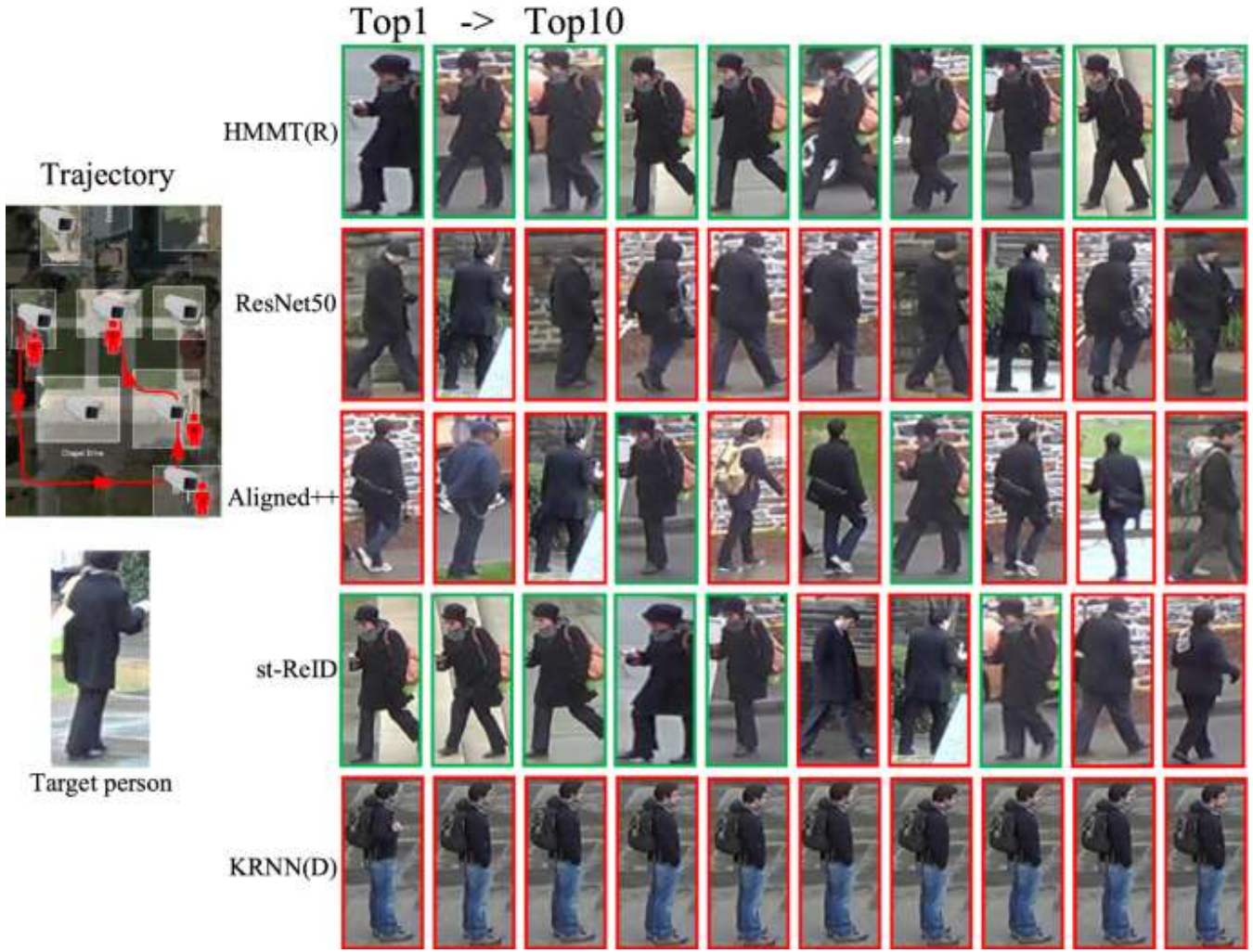
비슷하게 생긴 창문이라
다른 창문끼리 correspondence
높게 나옴. (local patch 만 보니까.)

find semantically matching points.

Recent works (semantic matching)



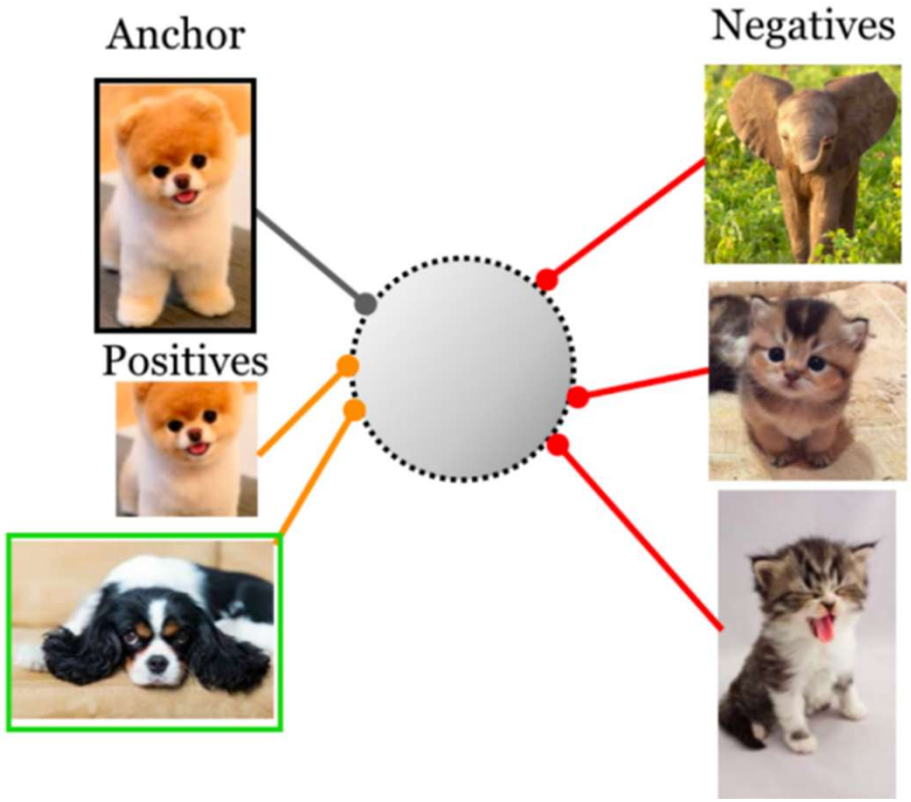
Recent works (person re-identification)



A little bit different recent works (self-supervised visual feature extractor)

contrastive learning.

- Train visual feature extractor without labels
- Embedding similar-looking images closer in the embedding space



A little bit different recent works (self-supervised visual feature extractor (DINO v3))

- With massive amount of training data (without any labels), the visual feature extractor can match semantically same parts