SPATIAL REASONING (1)

SCANNING AND MAPPING

Dr TIAN Jing

tianjing@nus.edu.sg

Module objective

Knowledge and understanding

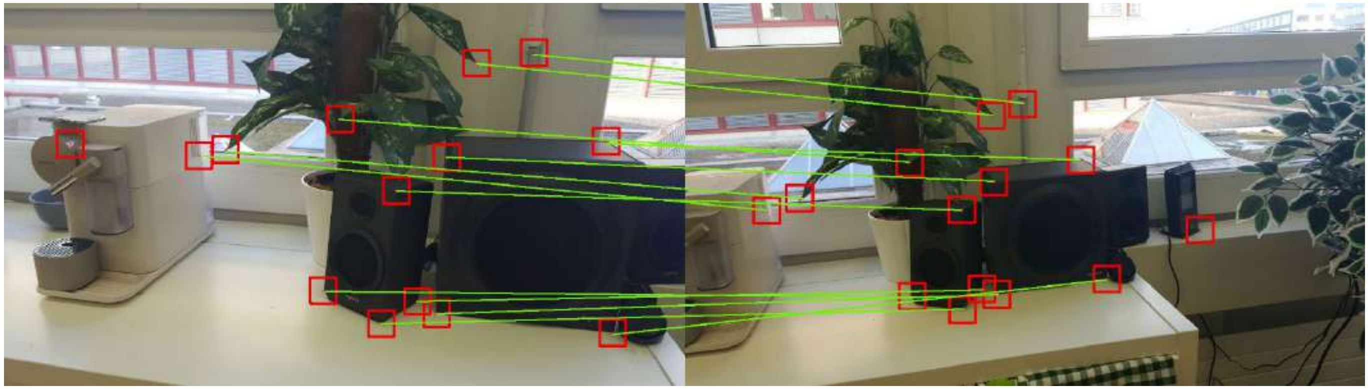
- Understand the fundamentals of spatial reasoning, including feature extraction and matching from multi-view images, 3D mapping.

Key skills

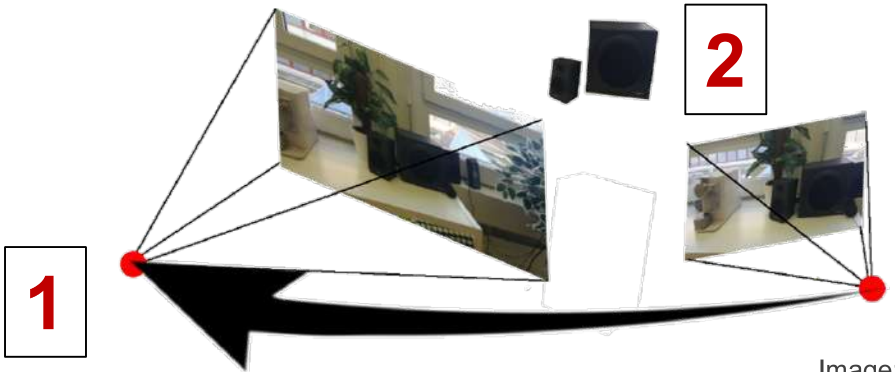
- Construct 3D scene map based on image/video captured by the camera

Recap of roadmap

Input data (multiple images)



Methods focused today



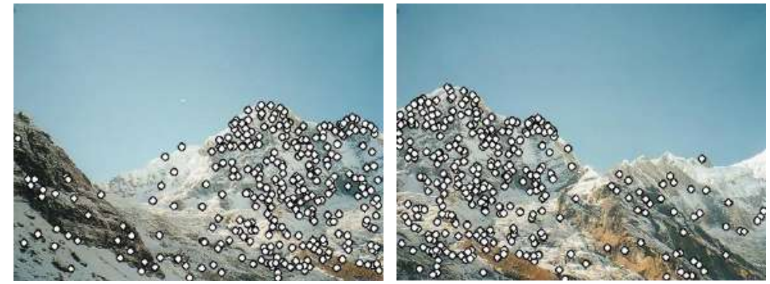
Deliverables

1. Transformation between input images
2. 3D position in the physical world

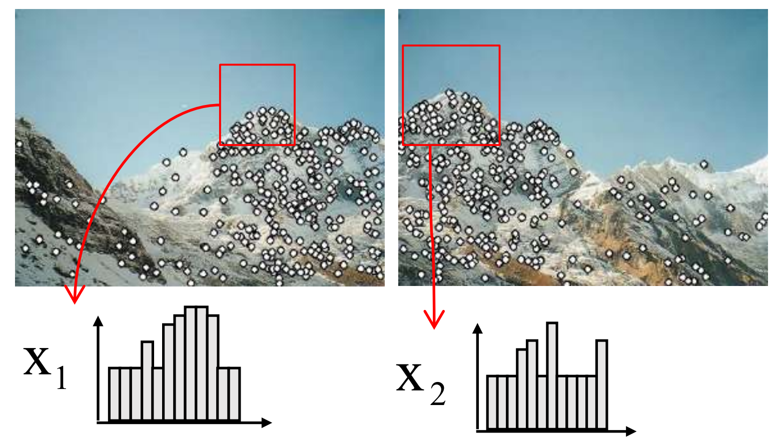
Image: http://rpg.ifi.uzh.ch/docs/teaching/2019/01_introduction.pdf

Feature matching

1) Detection:
   Find a set of distinctive key points.

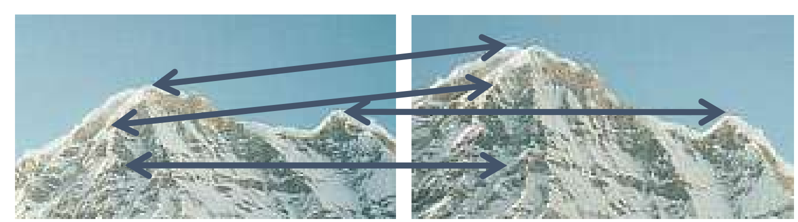


2) Description:
   Extract feature descriptor around each interest point as vector.

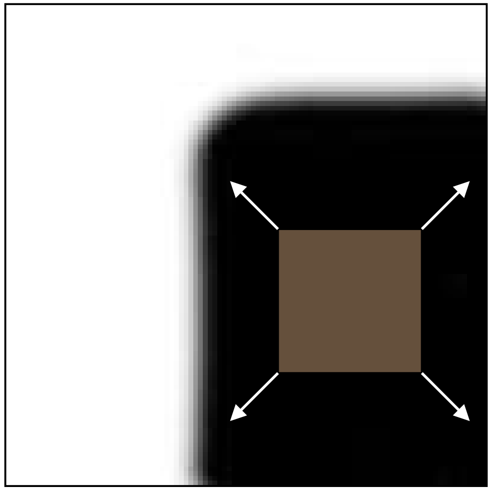


3) Matching:
   Compute distance between feature vectors to find correspondence.

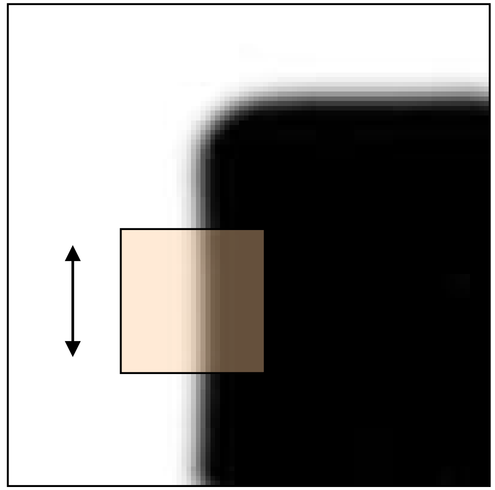
$$d(\mathbf{X}_1, \mathbf{X}_2) < T$$



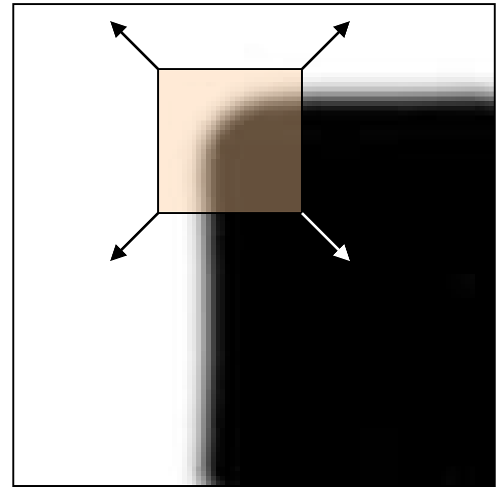
Recap: Harris detector



Flat: no change in all directions



Edge: no change along the edge direction



Corner: significant change in all directions

Objective: Find patches ($\Omega(x, y)$) that generate a large variation when it is moved around with a shift value (u, v).

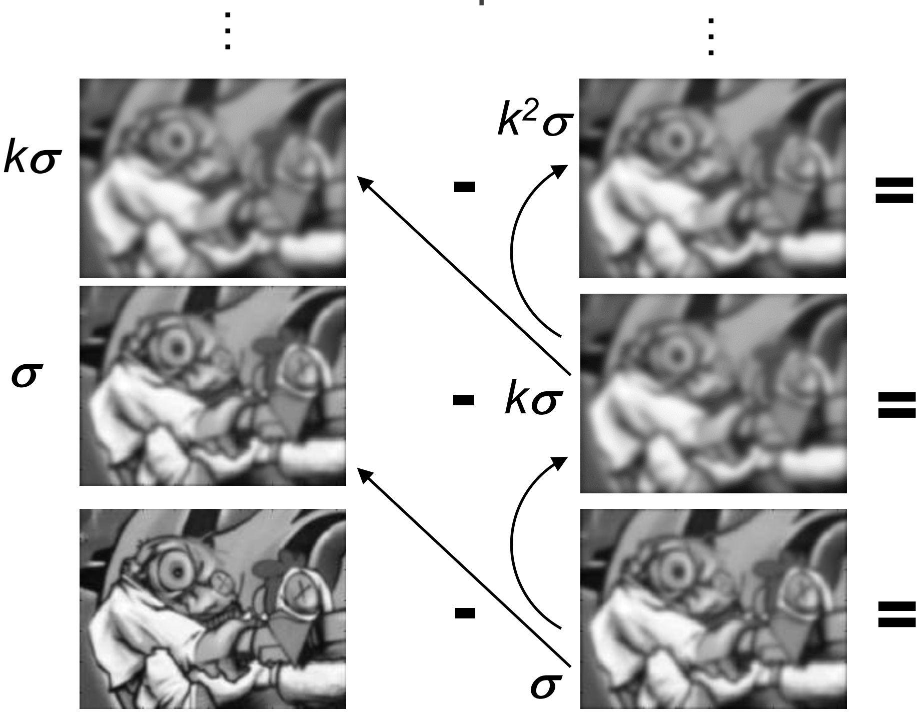
$$E(u, v) = \sum_{\Omega(x,y)} w(x, y)(I(x + u, y + v) - I(x, y))^2$$

- E is the difference between the original and the moved window.
- (u, v) are the window's displacements in the x, y directions, respectively.
- $w(x, y)$ is the mask function at position (x, y).
- $I(x + u, y + v)$ is the intensity of the moved window.
- $I(x, y)$ is the intensity of the original image.

SIFT (scale-invariant feature transform): Keypoint detection

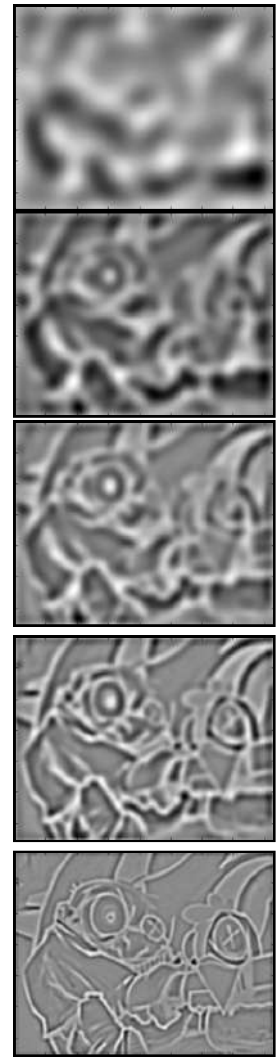
The step-by-step tutorial is available at http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/

Image pyramid with Gaussian filter ($k^s\sigma$) for s-th scale, σ is used in Gaussian filter, k is the user-defined parameter.



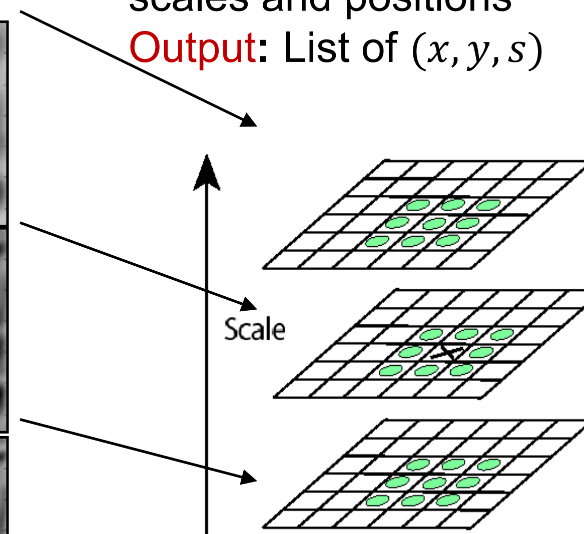
Input image

Filtered image



Find maxima across scales and positions

Output: List of (x, y, s)



Details: For each filtered image (with different scales) we compare the central pixel to its 9+8+9 neighbours (green locations in above figure) on the higher and the lower level. When the pixel is a maximum of this 9+8+9 blob, it is identified as a SIFT keypoint.

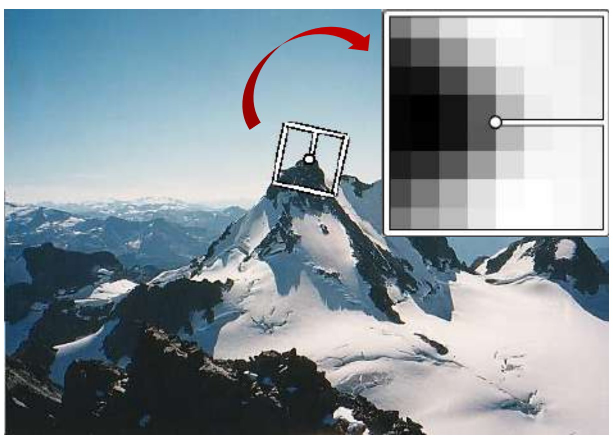
SIFT (scale-invariant feature transform): Feature extraction

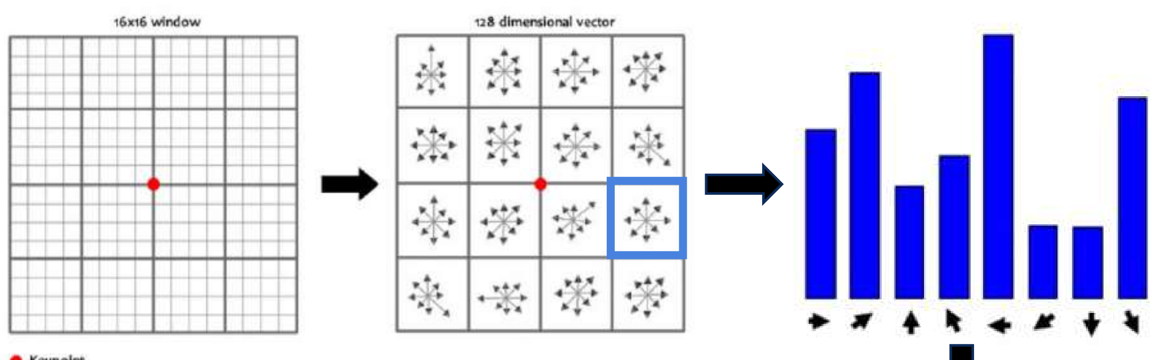
version:4.4.0 *July, 2020*

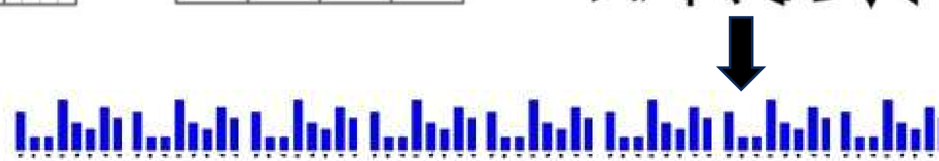
SIFT patent has been expired to be included into OpenCV 4.4 released on 18 July 2020

SIFT (Scale-Invariant Feature Transform) algorithm has been moved to the main repository (patent on SIFT is expired)

- Detect keypoints (see previous slide). For each keypoint (at specific location and scale), warp the region around it to canonical orientation and resize the region to 16×16 pixels.
- Divide the region into 4×4 squares (totally 16). Each square has 4×4 pixels.
- For each square, compute gradients for each pixels, then compute gradient direction histogram over 8 directions (bins). Concatenate the histograms computed from 16 squares to obtain a 128 ($16 \times 8 = 128$) dimensional feature.







Reference:

- https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/
- D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2, 2004, pp. 91-110

SURF: Speeded up robust features

Each sub-region (i.e., squares used in SIFT) has a four-dimensional descriptor vector for its underlying intensity structure $\mathbf{V} = (\sum dx, \sum|dx|, \sum dy, \sum|dy|)$. This results in a descriptor vector for all 16 sub-regions of length $64 = 16 \times 4$. (In SIFT, each descriptor has 128 dimensions).

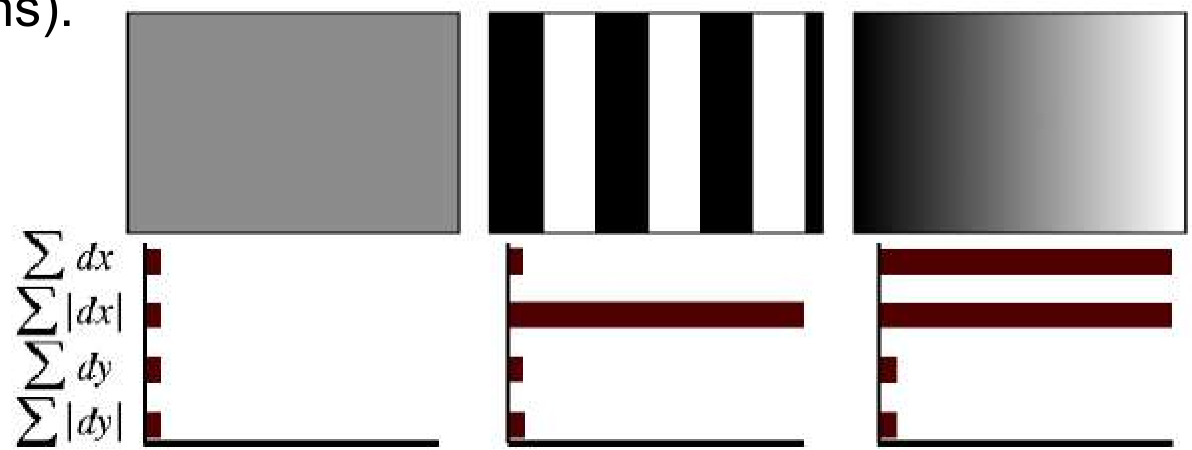


Fig. 3. The descriptor entries of a sub-region represent the nature of the underlying intensity pattern. Left: In case of a homogeneous region, all values are relatively low. Middle: In presence of frequencies in x direction, the value of $\sum |d_x|$ is high, but all others remain low. If the intensity is gradually increasing in x direction, both values $\sum d_x$ and $\sum |d_x|$ are high.

Reference:

- https://medium.com/@deepanshut041/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e
- H. Bay, T. Tuytelaars, L. V. Gool, SURF: Speeded Up Robust Features, ECCV 2006, pp. 404-417.

Feature matching

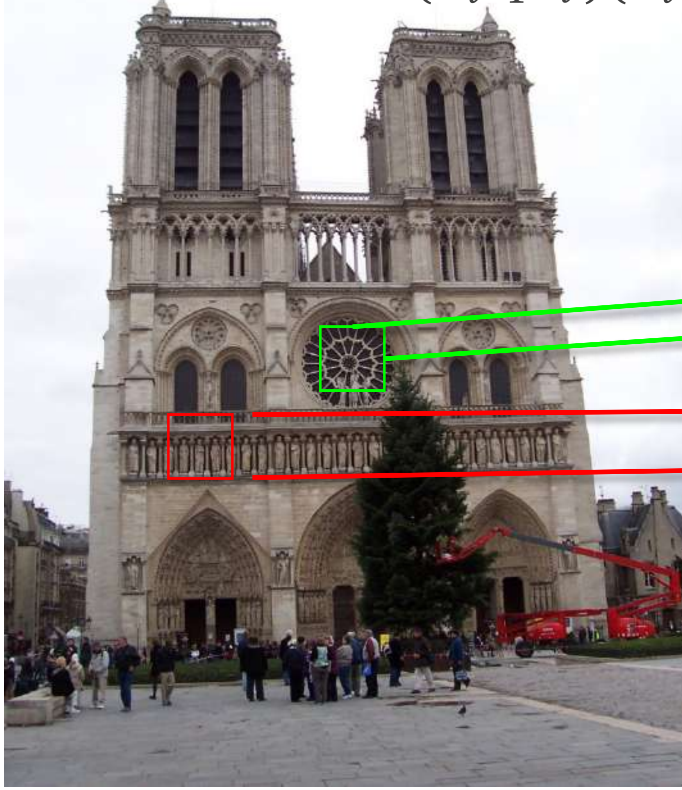
Given features $\mathbf{p}$ and $\mathbf{q}$ that are illustrated as squares in left/right images, respectively.

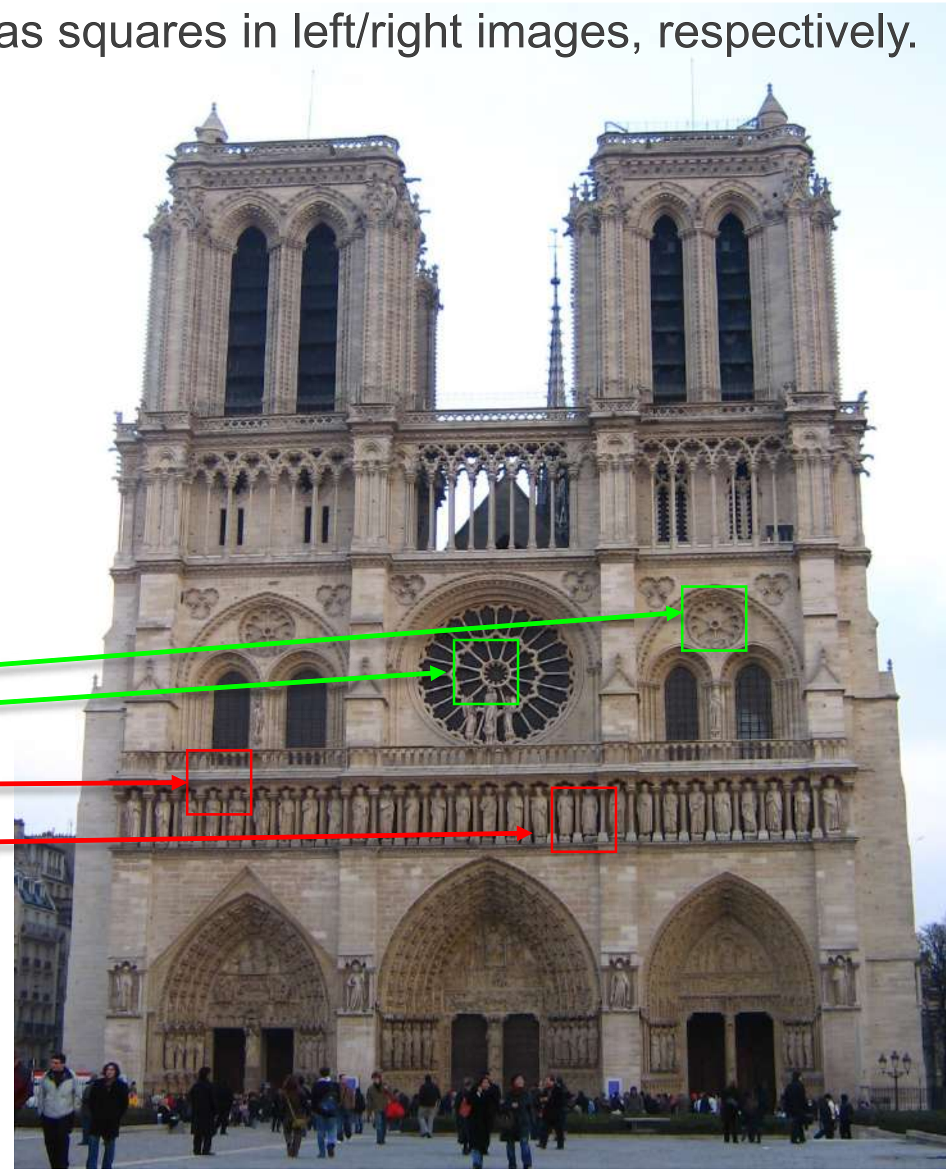
- Euclidean distance

$$d(\mathbf{p},\mathbf{q}) = \sqrt{(p_1 - q_1)^2 + \cdots + (p_n - q_n)^2}$$

- Cosine similarity

$$d(\mathbf{p},\mathbf{q}) = \frac{p \cdot q}{p^T p q^T q} = \frac{\sum_{i=1}^{n}(p_i q_i)}{\left(\sum_{i=1}^{n} p_i^2\right)\left(\sum_{i=1}^{n} q_i^2\right)}$$





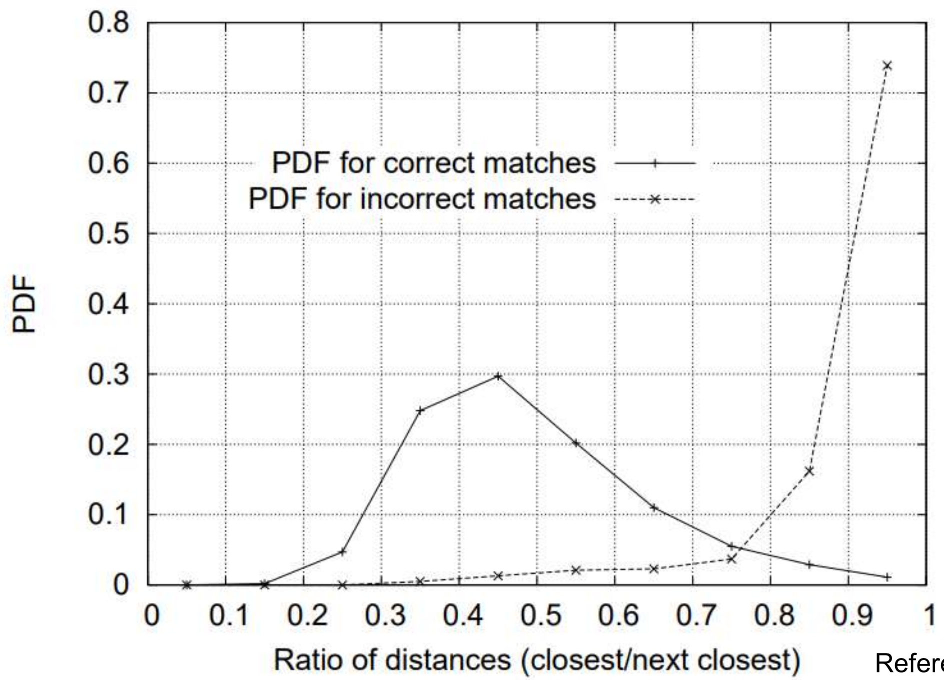
Feature matching

- Compare distance of *the closest (NN1)* and the *second-closest (NN2)* feature vector neighbor.

$$\text{If } NN1 \approx NN2 \quad Ratio\ \frac{NN1}{NN2} \approx 1 \quad \text{Ambiguity matches}$$

$$\text{If } NN1 \ll NN2 \quad Ratio\ \frac{NN1}{NN2} \to 0 \quad \text{Good match}$$

- Sort matches in the order of this ratio, then choose a threshold.

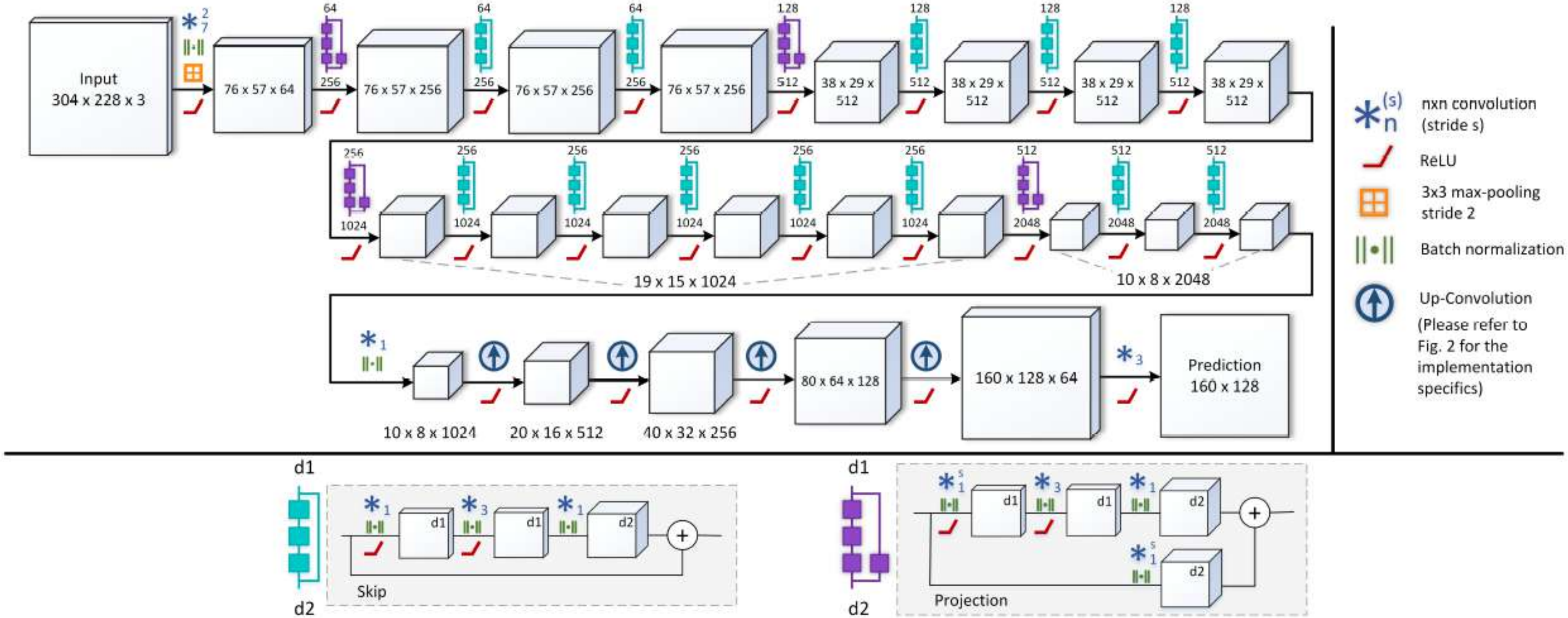


The probability that a match is correct can be determined by taking the ratio of distance from the closest neighbor to the distance of the second closest. Using a database of 40,000 keypoints, the solid line shows the *probability density function* (PDF) of this ratio for correct matches, while the dotted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

Reference: D. G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2, 2004, pp. 91-110.

End-to-end solution using machine learning

Objective: A fully convolutional architecture to model the ambiguous mapping between monocular images and depth maps.



Reference: I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, N. Navab, Deeper Depth Prediction with Fully Convolutional Residual Networks, https://arxiv.org/abs/1606.00373, code is available at https://github.com/Mhaiyang/Depth-Prediction

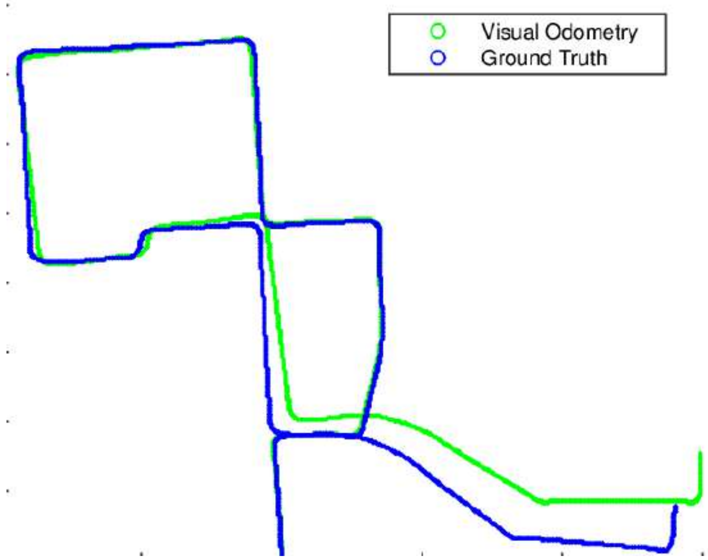
Application: Visual odometry for robotic path planning

Objective: Visual odometry incrementally estimates the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras

Dataset: Visual Odometry / SLAM Evaluation (full dataset, 22 GB), http://www.cvlibs.net/datasets/kitti/eval_odometry.php

Reference:

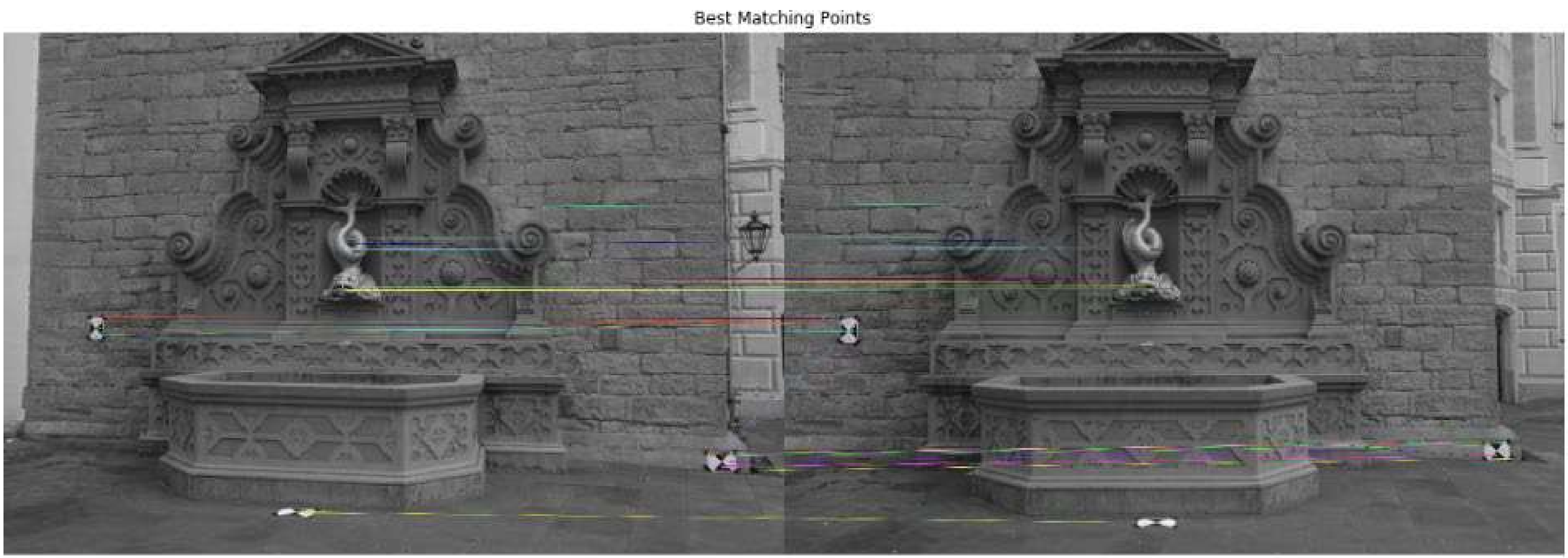
- Monocular visual odometry, https://github.com/avisingh599/mono-vo
- Stereo visual odometry, https://github.com/utkarshj1303/Stereo-Visual-Odometry-Plotting-An-Objects-Trajectory-Using-A-Sequence-Of-Images



Source: Davide Scaramuzza, *Tutorial on Visual Odometry*, available at https://sites.google.com/site/scarabotix/tutorial-on-visual-odometry

Workshop 3D sensor data representation and modelling

- Task: Feature extraction and matching from multiple view images
- Dataset: SfM Camera trajectory quality evaluation, https://github.com/openMVG/SfM_quality_evaluation



What we have learnt

Knowledge	• Camera model: Four reference systems, stereo image model.
Application	• Estimate depth from stereo camera • Construct point cloud (points in the world reference system) based on a set of images. • Visual odometry

Thank you!

Dr TIAN Jing
Email: tianjing@nus.edu.sg