Panorama

Sheng Li

sheng.li@seu.edu.cn

School of Cyber Science and Engineering
Southeast University

July 25, 2022

Overview

- Research objectives and contents
- Features
- SIFT
  - Scale-space extrema detection
  - Keypoint localization
  - Orientation assignment
  - Keypoint descriptor
- BRIEF
  - Method
  - Comparison
- Object recognition
  - Keypoint matching
  - Computing homograhy

Overview (cont.)

- Robust fitting with RANSAC
- Image stitching
  - Linear transformation
  - Alpha blending

Research objective and contents

- Objective
  In this project, we will implement a keypoint detector, a feature descriptor and an image stitching tool based on feature matching and homography.
- Contents
  Keypoint detectors find particularly salient points in an image. We can then extract a feature descriptor that helps describe the region around each of the interest points. Once we have extracted the interest points, we can use feature descriptors to find correspondences between images to do panorama stitching.

Requirements for features

no chance to match!

We need a repeatable detector

Figure 1: Detect the **same** points independently in both images

Requirements for features (cont.)

We need a reliable and distinctive descriptor

Figure 2: For each point correctly recognize the corresponding one

What is an Interesting Feature?

- Has sufficient image content (brightness/color/texture variation, etc.) within the local window
- Has a well-defined position in the image
- Has well-defined descriptions for matching with other points
- Invariant to image rotation and scaling

Are edges interesting?

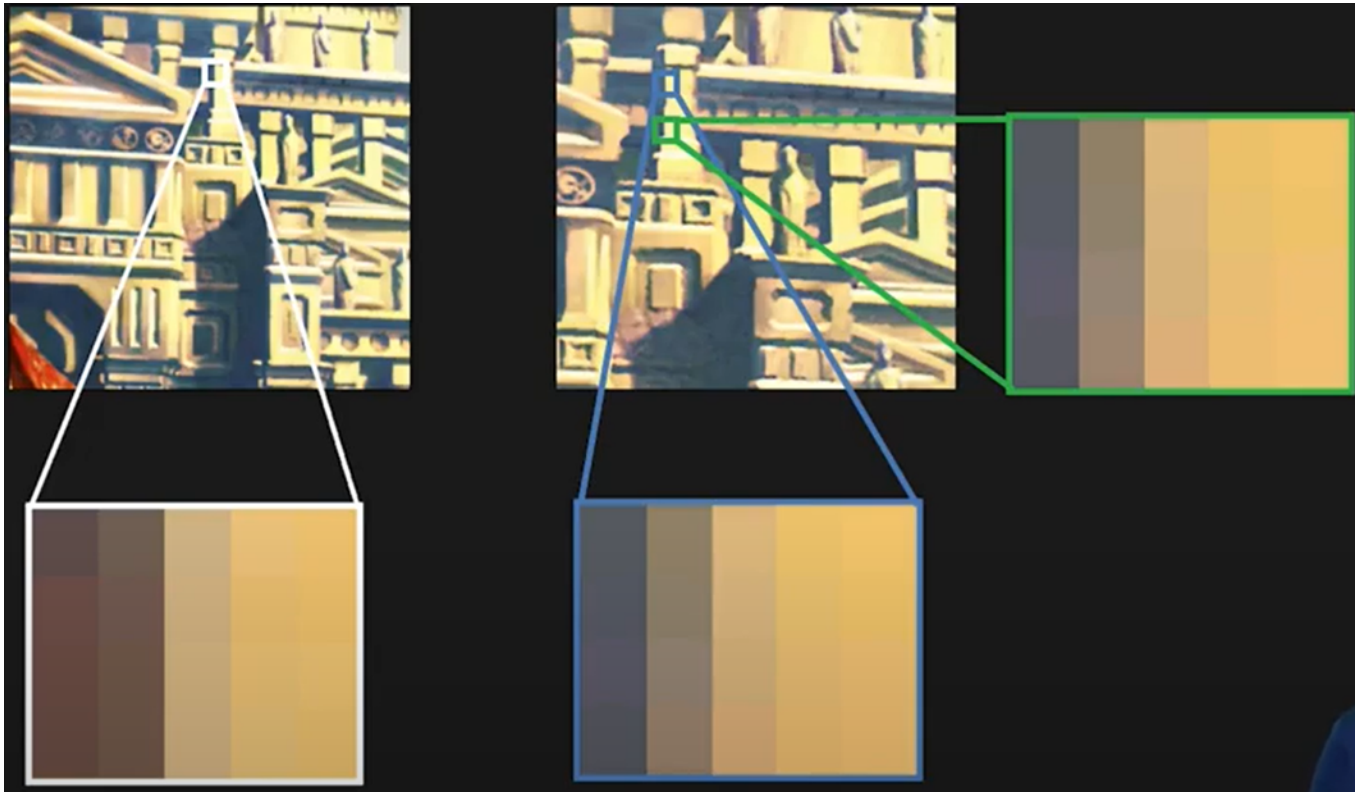

Figure 3: Not distinctive to **localize** a edge!

Are corners interesting?

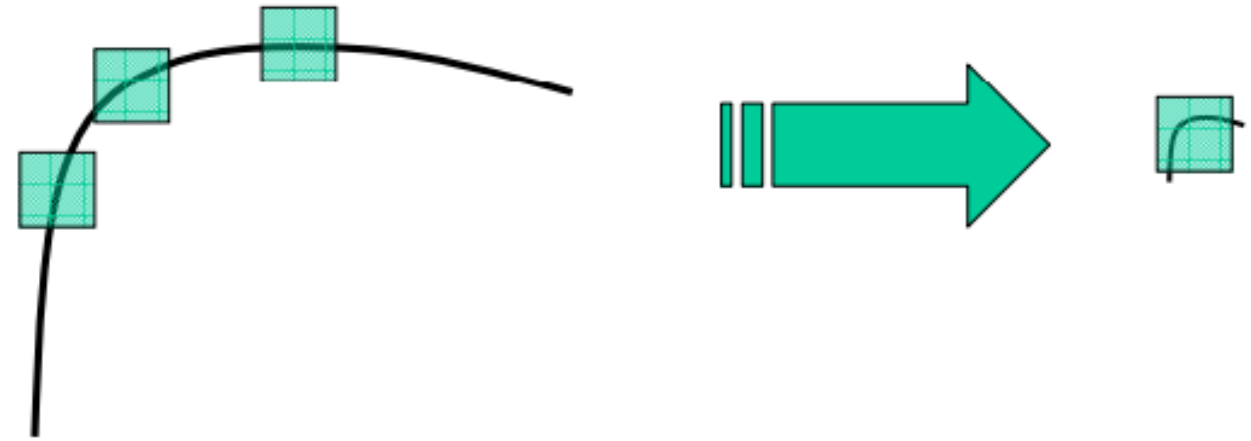

Figure 4: Not invariant to scaling!

Blobs as interest points

For a Blob-like Feature to be useful, we need to

- Locate the blob
- Determine its size
- Determine its orientation
- Formulate a description that is independent of size and orientation

Blobs as interest points (cont.)

Informally, a blob is a region of an image in which some properties are constant or approximately constant. One of the first and also most common blob detectors is based on the Laplacian of the Gaussian (LoG)[1].

Given an input image $f(x, y)$, this image is convolved by a Gaussian kernel at a certain scale to give a scale space representation as discussed in the previous frame. Then, the result of applying the Laplacian operator is computed.

Blobs as detection

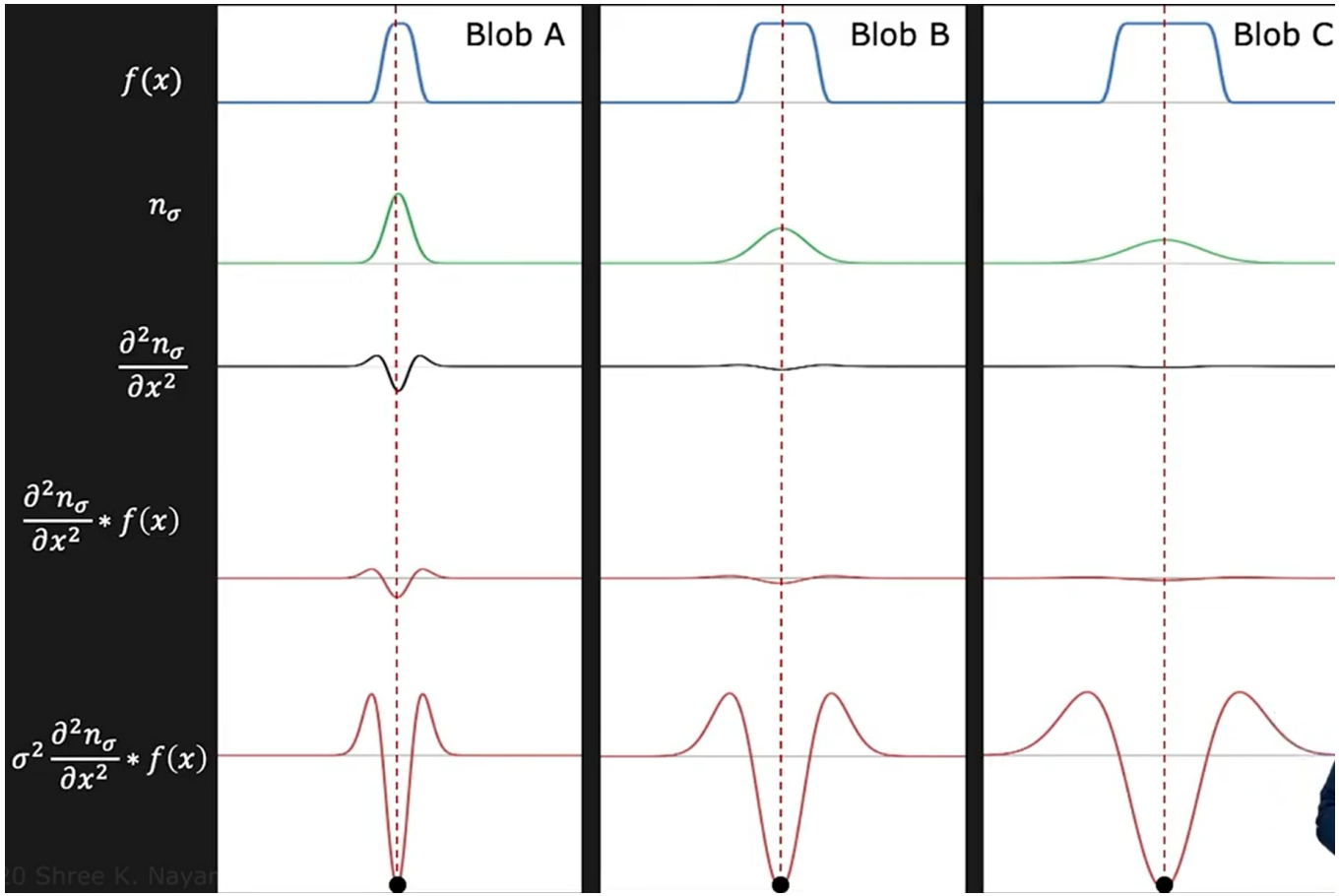

Figure 5: Automatic scale selection with Laplacian of Gaussian

Local extrema in (x, σ) space represent blobs, where σ is called characteristic scale.

Blobs as detection (cont.)

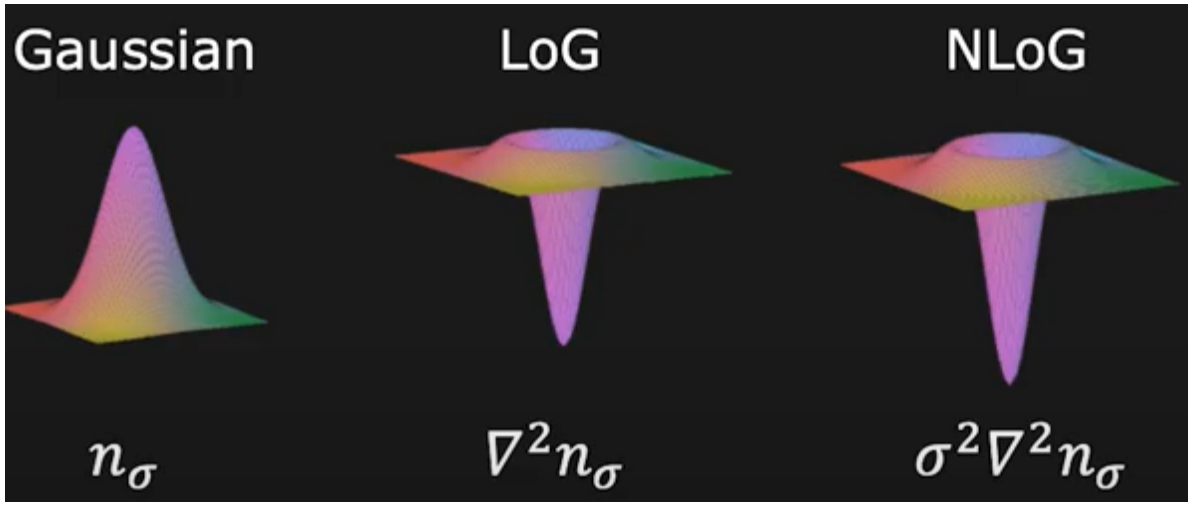

Figure 6: Normalized Laplacian of Gaussian

Characteristic Scale ($\propto$ size of blob): the σ at which σ-normalized second derivative attains its extreme value.
Locations of blobs are given by **local extrema** after applying Normalized Laplacian of Gaussian at many scales.

Introduction to SIFT

SIFT, aka scale invariant feature transform is a method for extracting **distinctive** invariant features from images that can be used to perform reliable matching between different views of an object or scene.

- Invariant to **image scale and rotation**
- Robust matching across a a substantial range of **affine distortion, change in 3D viewpoint, addition of noise, and change in illumination**
- Match with **high probability** against a large database of features from many images

Workflow

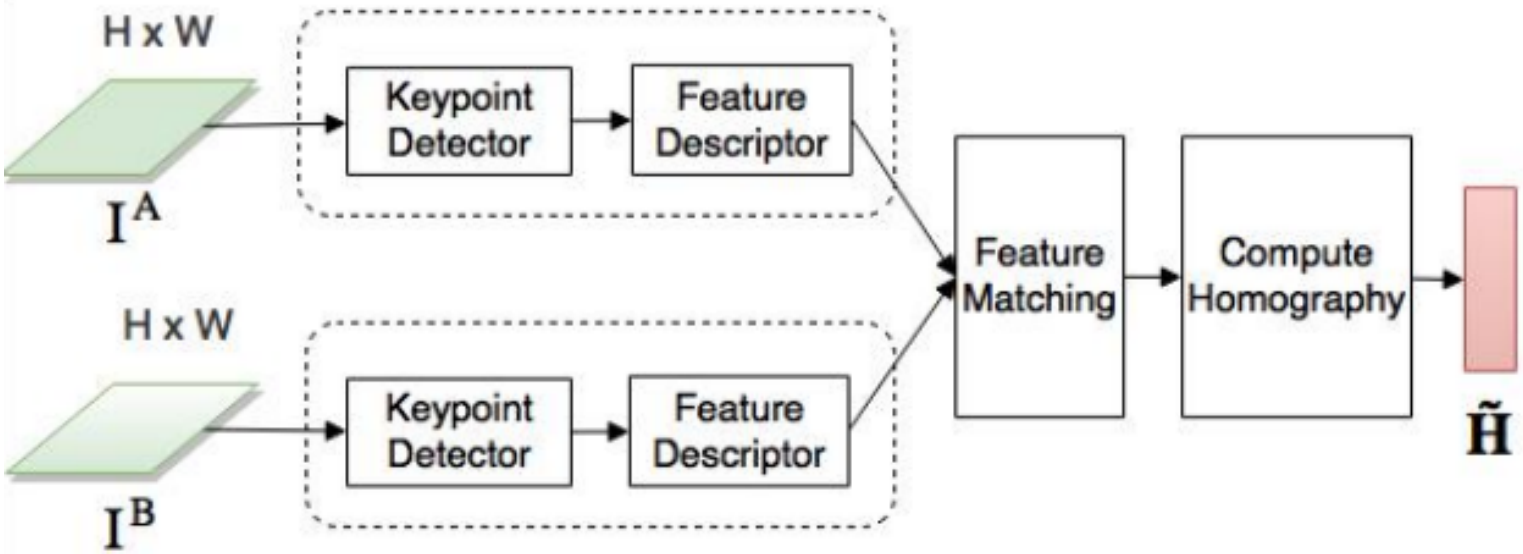

Figure 7: Generic Feature-based Approach

1. Find a set of distinctive keypoints
2. Define a region around each keypoint
3. Compute a local descriptor from the region
4. Match descriptors

Overall picture of keypoint detection

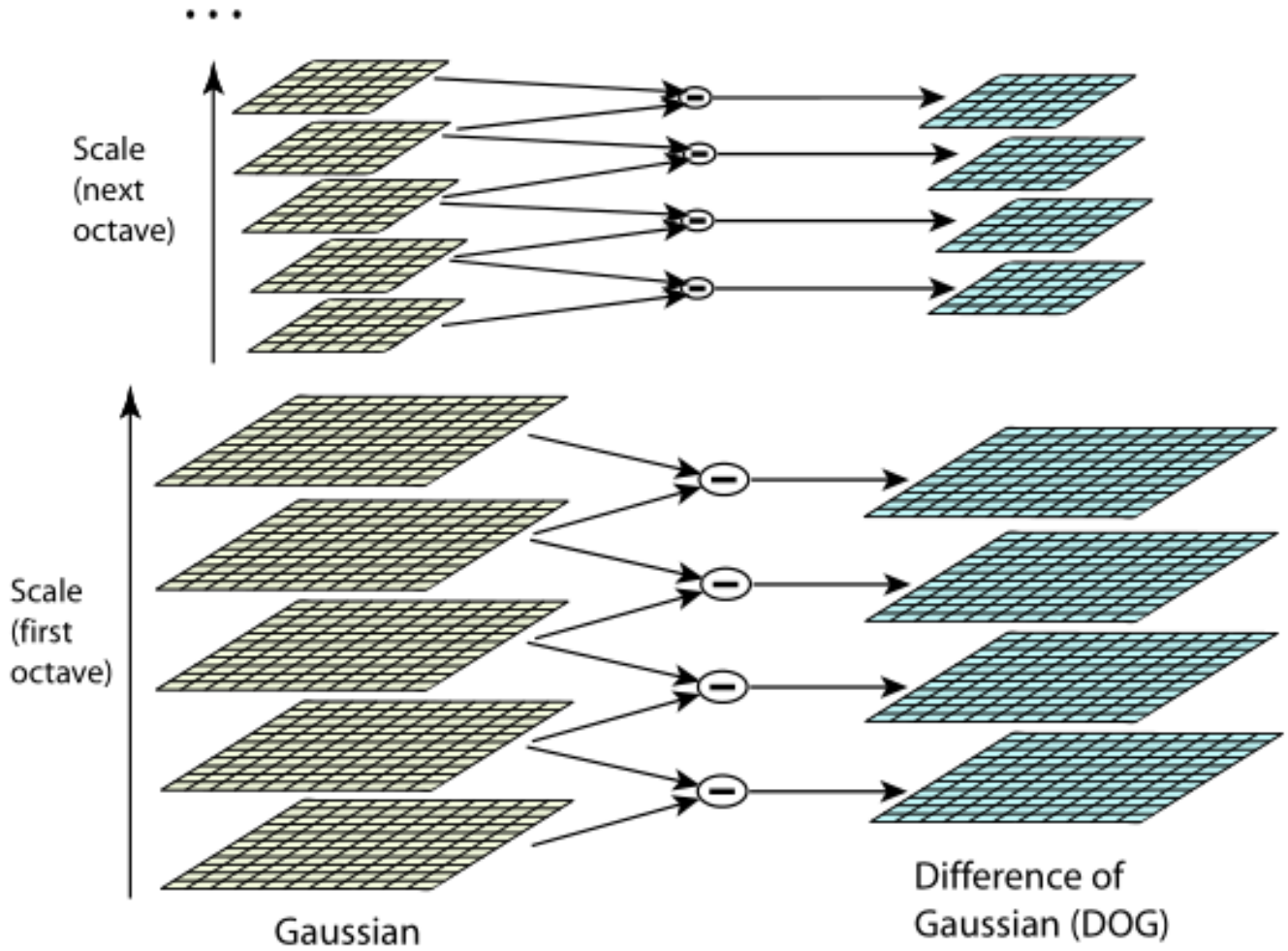

Figure 8: Scale space construction [2]

Gaussian pyramid

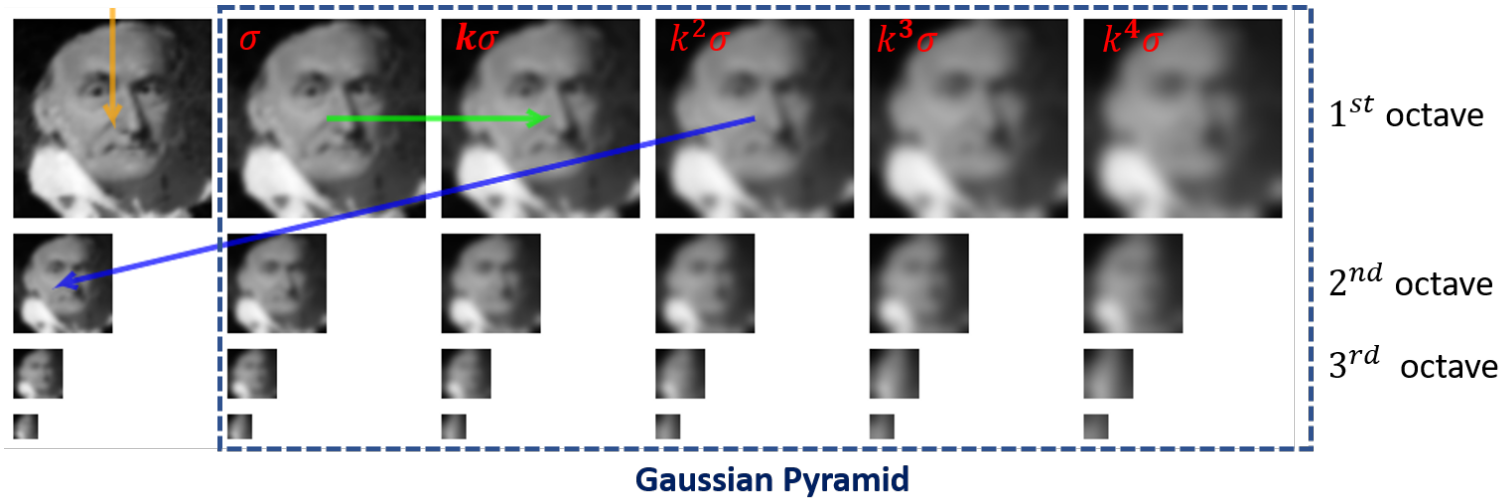

Figure 9: Gaussian pyramid construction

Convolve input image I with Gaussian G of various scale σ, following the green arrow.

$$L(x, y, k^l\sigma) = G(x, y, k^l\sigma) * I(x, y)$$

where $k = \sqrt{2}$ and $l \in 0, 1, 2, 3, 4$

After each octave, the Gaussian image is down-sampled by a factor of 2, following the blue arrow.

Approximation to LoG

The relationship between D and $\nabla^2 G$ can be understood from the heat diffusion equation:

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

$\nabla^2 G$ can be computed from the finite difference approximation to $\frac{\partial G}{\partial \sigma}$, using the difference of nearby scales at $k\sigma$ and σ:

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

and therefore,

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G$$

This shows that when the DoG function has scales differing by a constant factor, it already incorporates the σ^2 scale normalization required for the scale-invariant Laplacian.

Difference of Gaussian

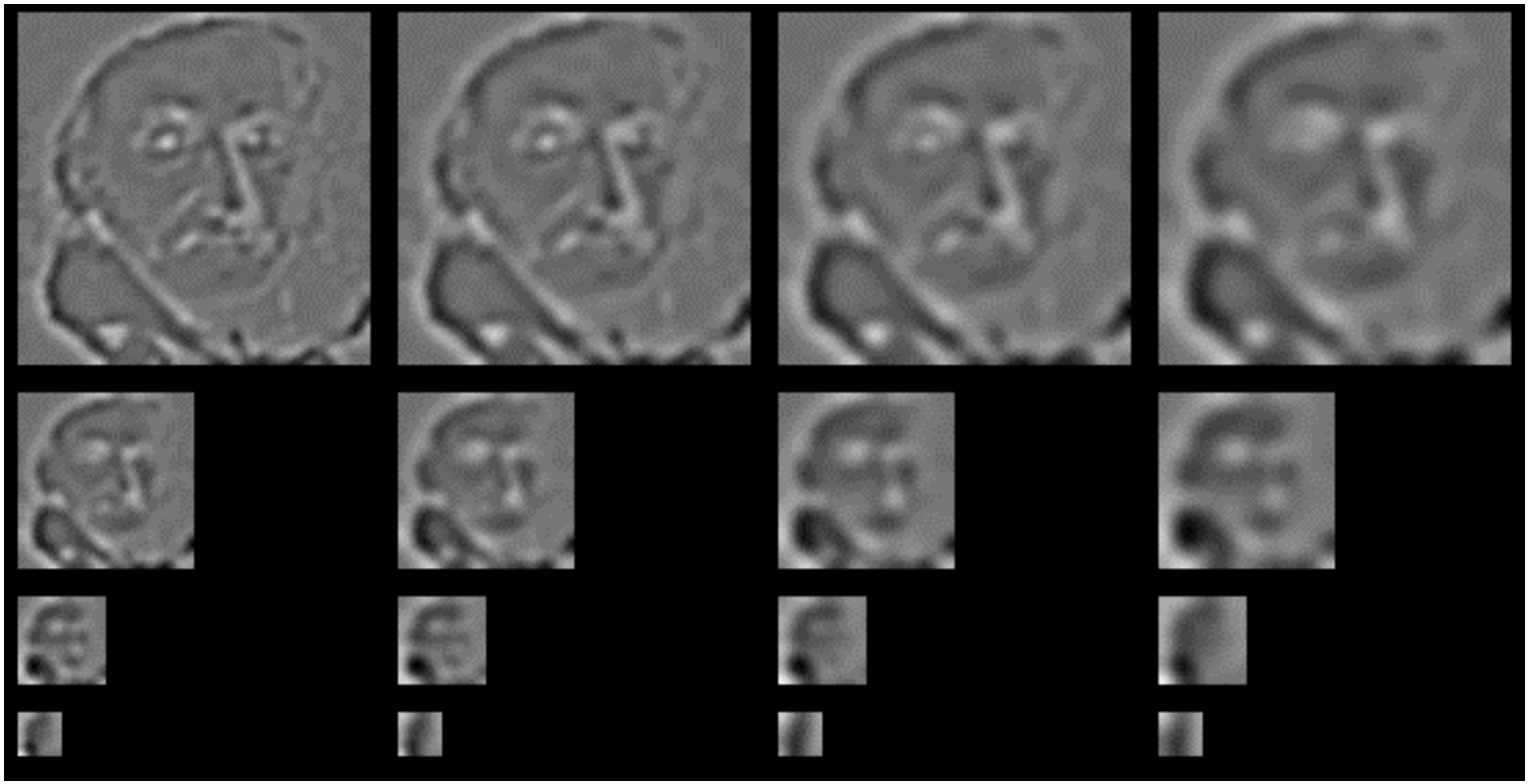

Figure 10: DoG pyramid construction

To detect stable keypoint, convolve image I with difference of Gaussian:

$$D(x, y, k^l\sigma) = L(x, y, k^{l+1}\sigma) - L(x, y, k^l\sigma)$$

where $k = \sqrt{2}$ and $l \in 0, 1, 2, 3$

Local extrema detection

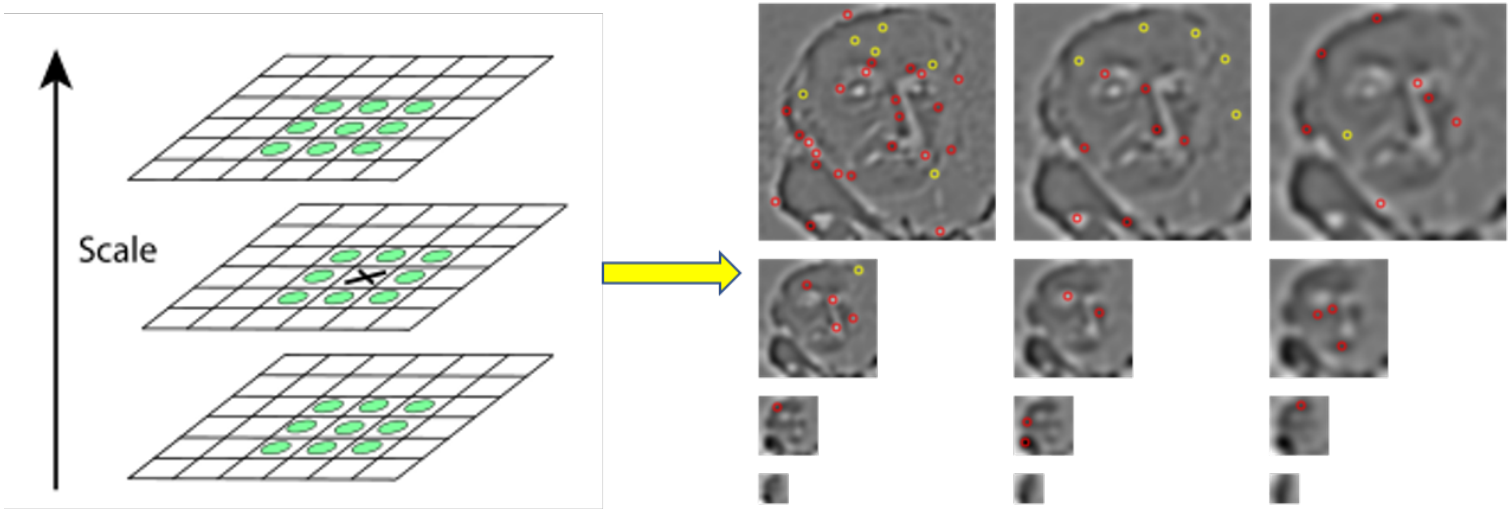

Figure 11: 26 comparisons in 3x3 regions at the current and adjacent scales

Each sample point is compared to its 8 neighbors in the current image and 9 neighbors in the scale above and below.

To **remove weak extrema**: check against $|D(x, y, \sigma)| > 0.03$

Principal curvature ratio

The principal curvatures can be computed from a 2x2 Hessian matrix [3]:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

Let α be the larger eigenvalue and β be the smaller one($\alpha = r\beta$):

$$\mathrm{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$
$$\mathrm{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Curvature ratio R for the corresponding point in the DoG pyramid:

$$R = \frac{\mathrm{Tr}(H)^2}{\mathrm{Det}(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r+1)^2}{r}$$

To **remove edge points**, check against a threshold

$$R > 12(r = 10)$$

Principal orientation

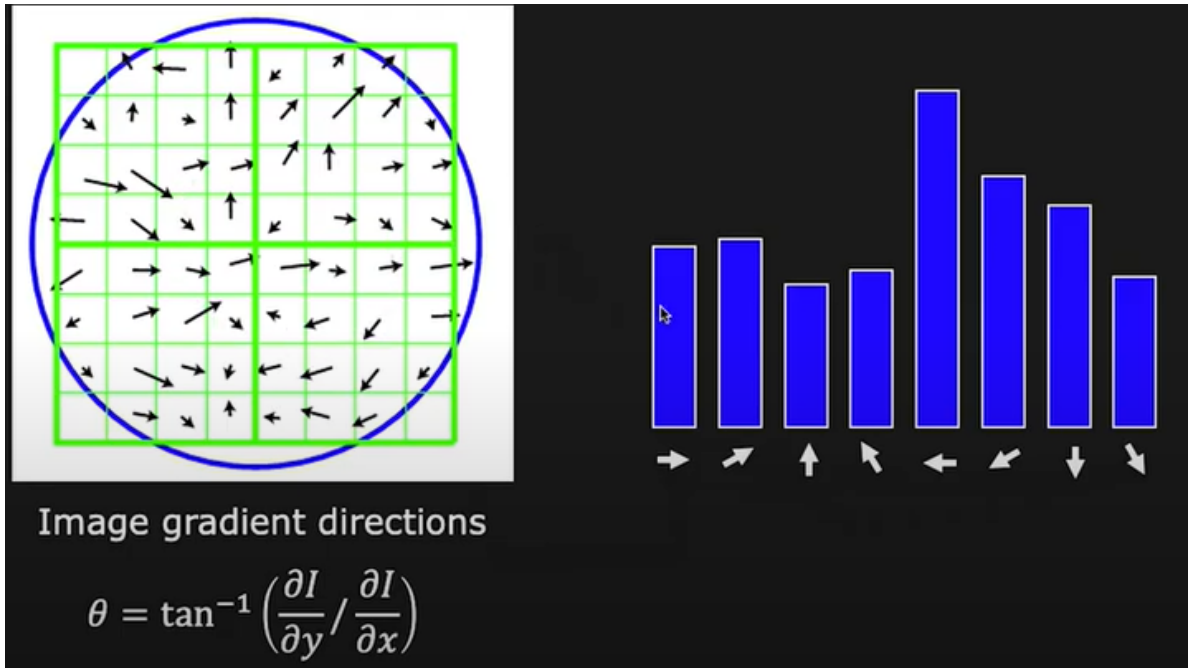

Figure 12: Use the histogram of gradient directions

The orientation histogram has 36 bins covering the 360 degree range of orientations. Each sample added to the histogram is weighted by its gradient magnitude and by a Gaussian-weighted circular window with **a** σ **that is 1.5 times that of the scale of the keypoint**. Peaks in the orientation histogram correspond to dominant directions of local gradients.

Descriptor representation

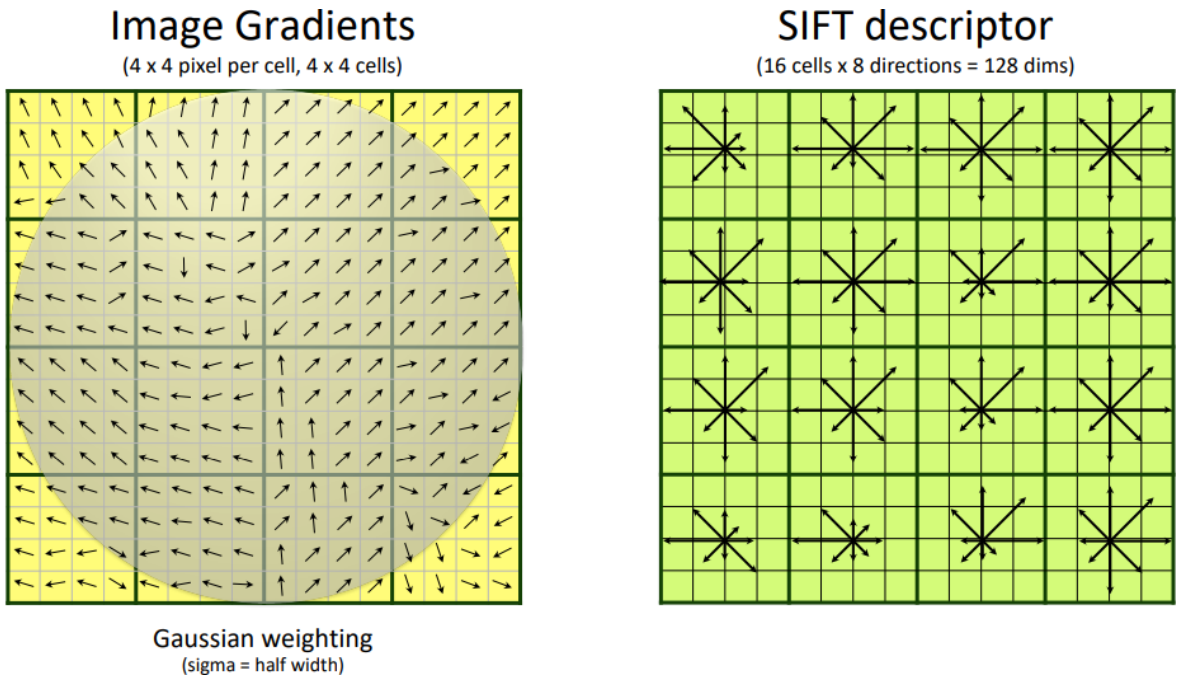

Figure 13: Descriptor representation

- Rotation invariance: relate with the keypoint principal orientation
- Collect into 4×4 orientation histograms with 8 orientation bins
- Bin value = sum of gradient magnitudes near that orientation
- Normalize feature vector to unit length to reduce effect of linear illumination change.

BRIEF

BRIEF[4]: Binary Robust Independent Elementary Features uses binary strings as an efficient feature point descriptor. It is highly discriminative even when using relatively few bits and can be computed using simple intensity difference tests. Furthermore, the descriptor similarity can be evaluated using the Hamming distance, which is very efficient to compute, instead of the L_2 norm as is usually done.

Pairwise intensity comparisons

The descriptor itself is a vector that is n-bits long, where each bit is the result of the following simple test:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) := \begin{cases} 1, & \text{if } \mathbf{p}[\mathbf{x}] < \mathbf{p}[\mathbf{y}] \\ 0, & \text{otherwise} \end{cases}$$

where $\mathbf{p(x)}$ is the pixel intensity in the image of $\mathbf{p}$ at $\mathbf{x} = (u, v)$. There is no need to encode the test results as actual bits. It is fine to encode them as a 256 element vector by setting n to 256 bits.

Spatial arrangement of the binary tests

Generating a length 256-bit vector leaves many options for selecting the nd test locations (x_i, y_i) in a patch of size $S \times S$.

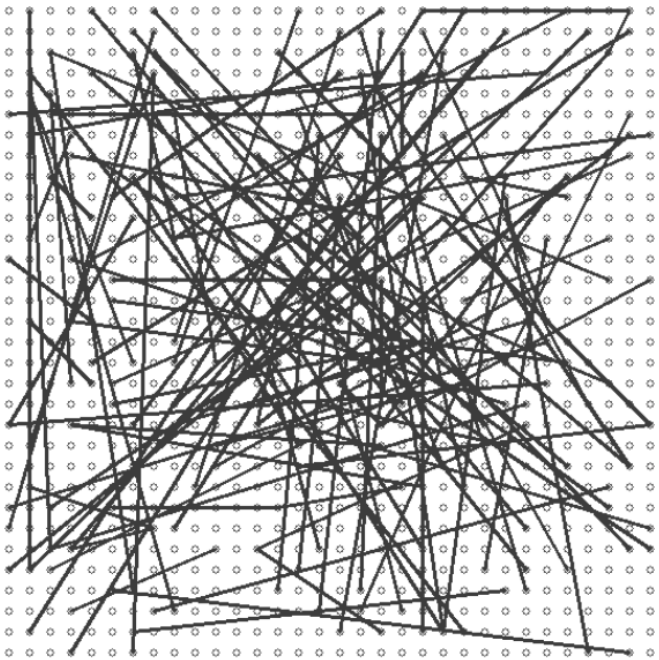

Figure 14: Approach to choosing the test locations

$(\mathbf{X}, \mathbf{Y}) \sim$ i.i.d. Uniform(- $\frac{patchWidth}{2}$, $\frac{patchWidth}{2}$): The (u_i, y_i) locations are evenly distributed over the patch and tests can lie close to the patch border.

Matching metric

The distance metric used to compute the similarity between two descriptors is critical. For BRIEF, this distance metric is the Hamming distance, which is simply the number of bits in two descriptors that differ.

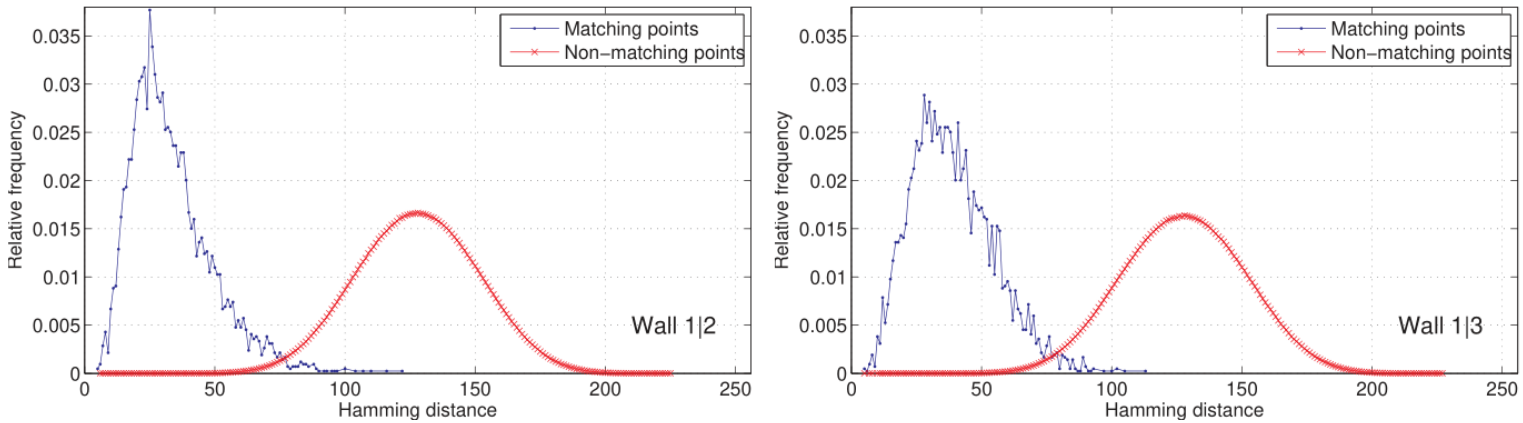

Figure 15: Distributions of Hamming distances for matching pairs of points

For each image pair, Figure 15 shows the normalized histograms, or distributions, of Hamming distances between corre- sponding points (in blue) and non-corresponding points (in red).

Orientation sensitivity

BRIEF is not designed to be rotationally invariant. Nevertheless, as shown by our results on the 5 test data sets, it tolerates small amounts of rotation. Up to 10 to 15 degrees, there is little degradation of recognition rate followed by a precipitous drop.

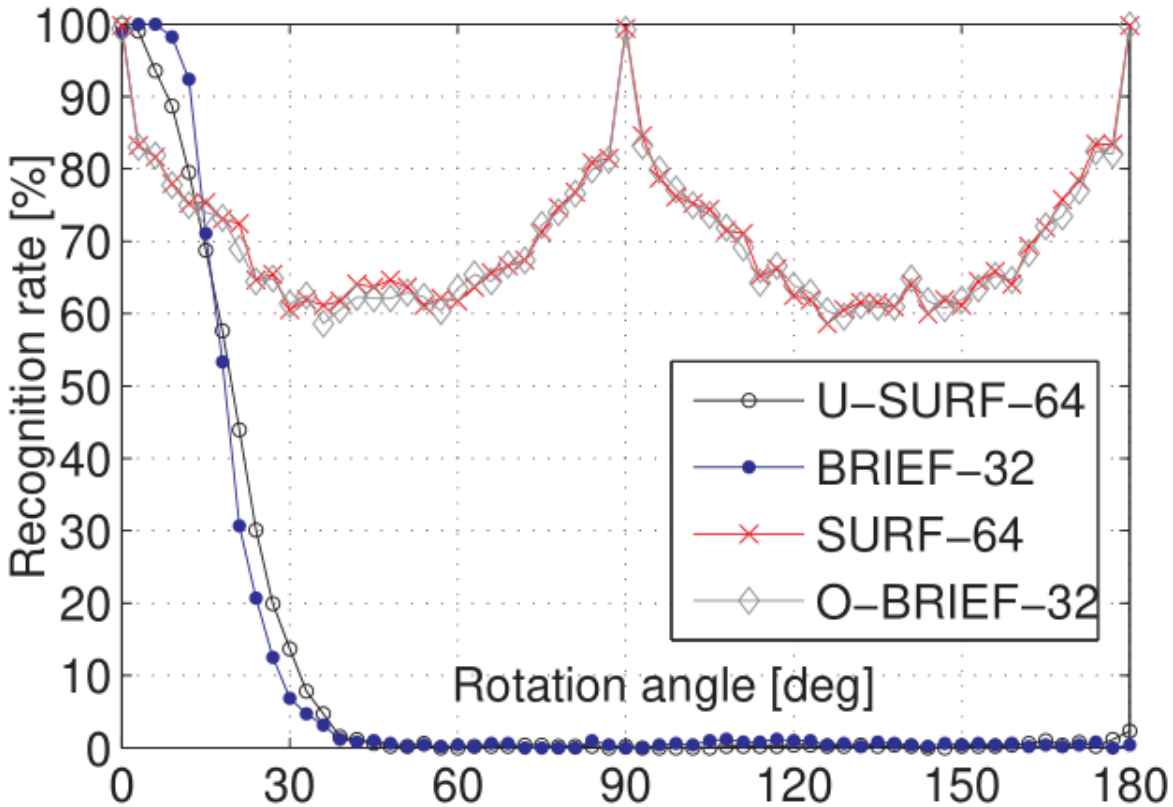

Figure 16: Recognition rate as a function of the rotation angle

Comparison with SIFT

Our experiments show that only 256 bits often suffice to obtain very good matching results. BRIEF is therefore very **efficient both to compute and to store in memory**. Furthermore, comparing strings can be done by computing the Hamming distance, which can be done extremely **fast on modern CPUs** that often provide a specific instruction to perform a XOR or bit count operation.
Future work will incorporate **orientation and scale invariance** into BRIEF so that it can compete with SIFT in a wider set of situations.

Object recognition

Object recognition is performed by first matching each keypoint independently to the database of keypoints extracted from training images. Many of these initial matches will be incorrect due to ambiguous features or features that arise from background clutter. Therefore, clusters of at least 3 features are first identified that agree on an object and its pose, as these clusters have a much higher probability of being correct than individual feature matches [5]. Then, each cluster is checked by performing a detailed geometric fit to the model, and the result is used to accept or reject the interpretation.

Keypoints matching

1. The **best candidate** match for each keypoint is found by identifying its **nearest neighbor** in the database of keypoints from training images.
2. The nearest neighbor is defined as the keypoint with minimum Euclidean distance between feature vectors:

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

3. Efficient nearest neighbor indexing with k-d tree [6].

Normal equation

The affine transformation of $[x\ y]^T$ to $[u\ v]^T$ can be written as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

where the model translation is $[t_x\ t_y]^T$.

To solve for the transformation parameters, the equation above can be rewritten to gather the unknowns into a column vector:

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \\ & & \cdots & & & \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_x \\ t_y \end{bmatrix} = \begin{bmatrix} u \\ v \\ \vdots \end{bmatrix}$$

$$\mathbf{Ax} = \mathbf{b}$$

x can be determined by solving the normal equations,

$$\mathbf{x} = \left[\mathbf{A}^\mathbf{T}\mathbf{A}\right]^{-1}\mathbf{A}^\mathbf{T}\mathbf{b}$$

Estimating homography using RANSAC

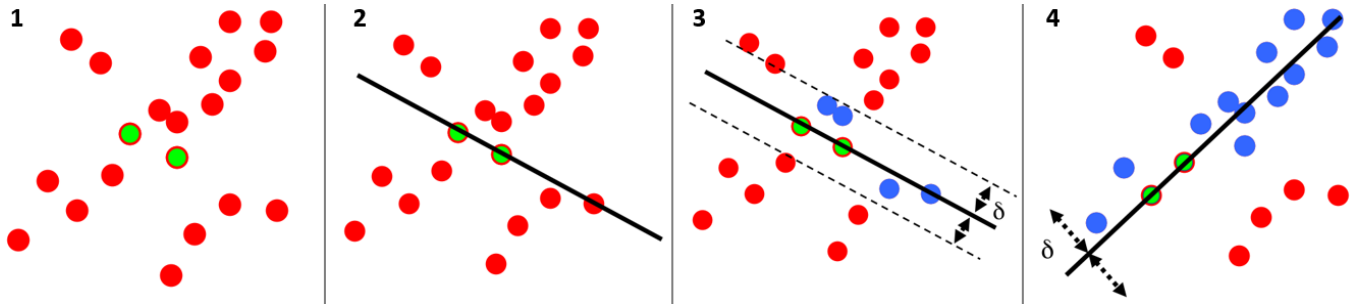

Figure 17: RANSAC algorithm[7] for fitting lines

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples, i.e. compute H.
3. Count inliers that fit the model within a measure of error ε.
4. Repeat Steps N times.
5. Choose the model that has the largest number inliers and recompute H using all inliers.

Where $s = 4$ pairs of feature points and ε is acceptable alignment error in pixels for homography.

Transform into the same frame

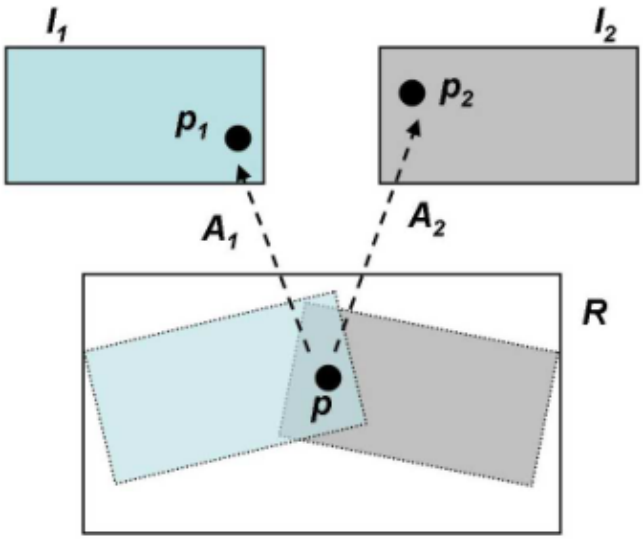

Figure 18: Combine overlapping images into single image

For each pixel **p** in R, compute:

$$p_1 = A_1 p \text{ and } p_2 = A_2 p.$$

- Both p_1 and p_2 fall outside I_1 and I_2, then $R(p) =$ black.
- Both p_1 and p_2 fall inside I_1 and I_2, then $R(p) =$ blending of $I_1(p_1)$ and I2(p2).
- Otherwise, only one of p_1 or p_2 falls inside I_1 or I_2. So, R(p) = I1(p1) or I2(p2), as appropriate.

Alpha blending

Usually, the images to be stitched together have different overall intensity and contrast. The stitched image has an apparent seam as circled in Figure 19.

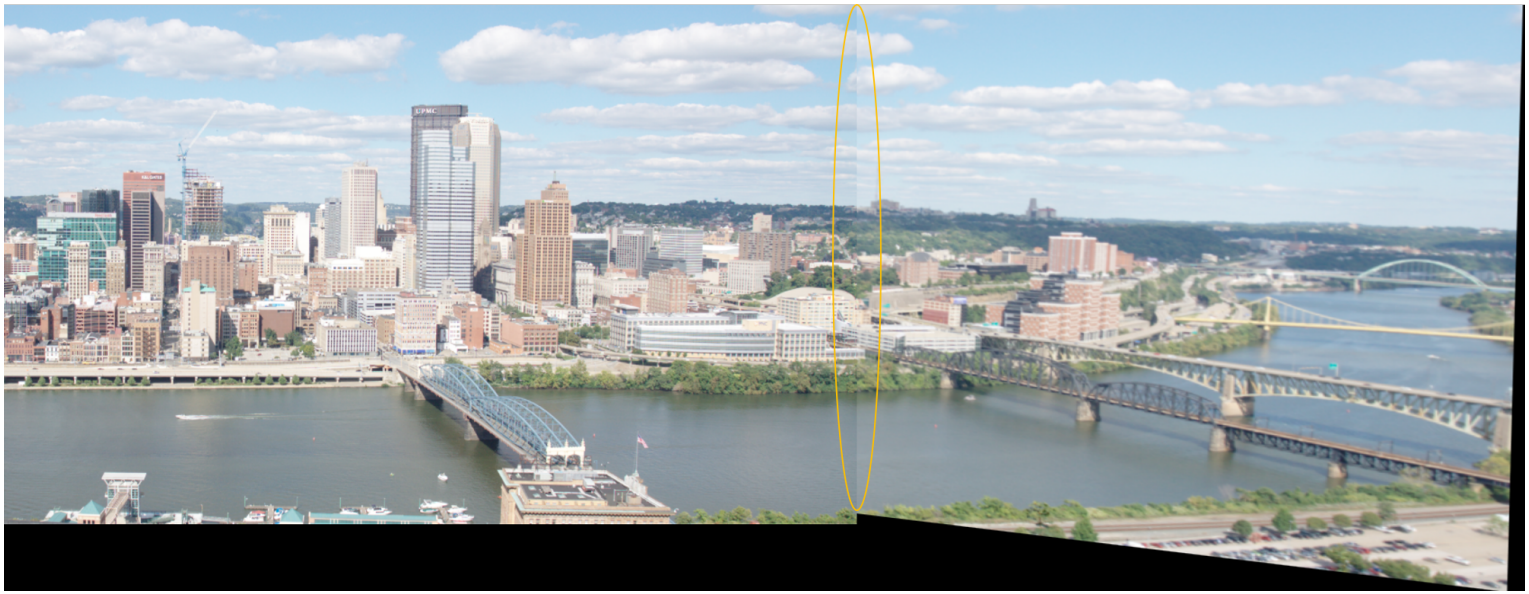

Figure 19: Apparent seam in the stitched image

Alpha blending (cont.)

The basic idea of **alpha blending** is introduced as follows:

- Let the color in the overlapping regions change smoothly from the color in one image to the color in the other image.
- Let $C_1(p)$ denote color of pixel p in image 1.
- Let $C_2(p)$ denote color of pixel p in image 2.
- Then, color $C(p)$ of blended image is given by

$$C(p) = \alpha C_1(p) + (1 - \alpha) C_2(p)$$

- where α is related to the distances to the overlapping boundaries

$$\alpha = \frac{d_1}{d_1 + d_2}$$

Bibliography

[1] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," in *Readings in computer vision*, Elsevier, 1987, pp. 671–679.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[3] C. Harris, M. Stephens, *et al.*, "A combined corner and edge detector," in *Alvey vision conference*, Citeseer, vol. 15, 1988, pp. 10–5244.

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*, Springer, 2010, pp. 778–792.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157.

Bibliography (cont.)

[6] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Transactions on Mathematical Software (TOMS)*, vol. 3, no. 3, pp. 209–226, 1977.

[7] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.