UNIVERSITÀ DI PARMA

Feature Matching

Summary

UNIVERSITÀ DI PARMA

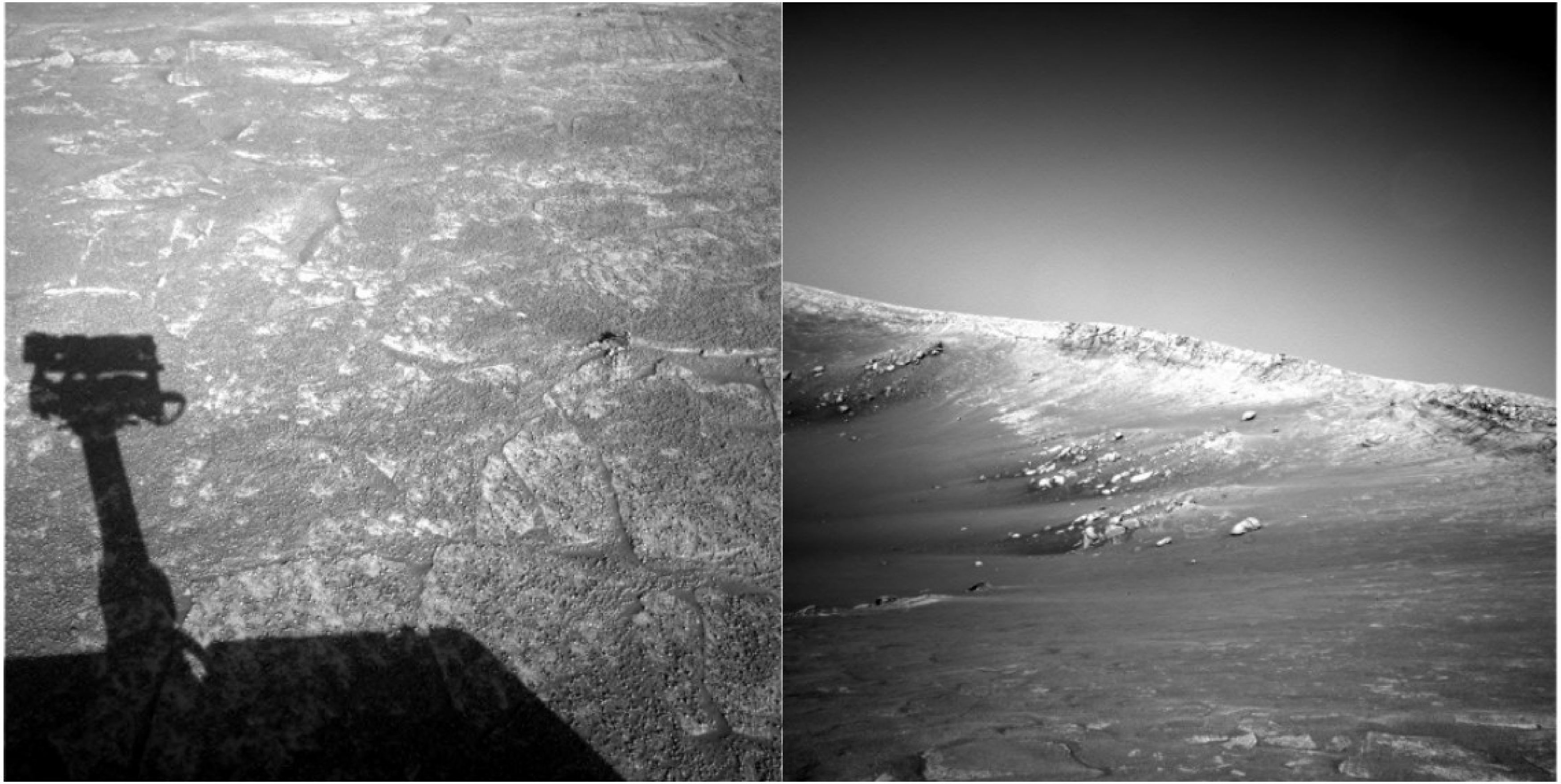
- Keypoints descriptor
  - SIFT
  - BRIEF
  - BRISK
- Performance
- Feature Matching

References

- [FP] D. A. Forsyth and J. Ponce. **Computer Vision: A Modern Approach** (2nd Edition). Prentice Hall, 2011.

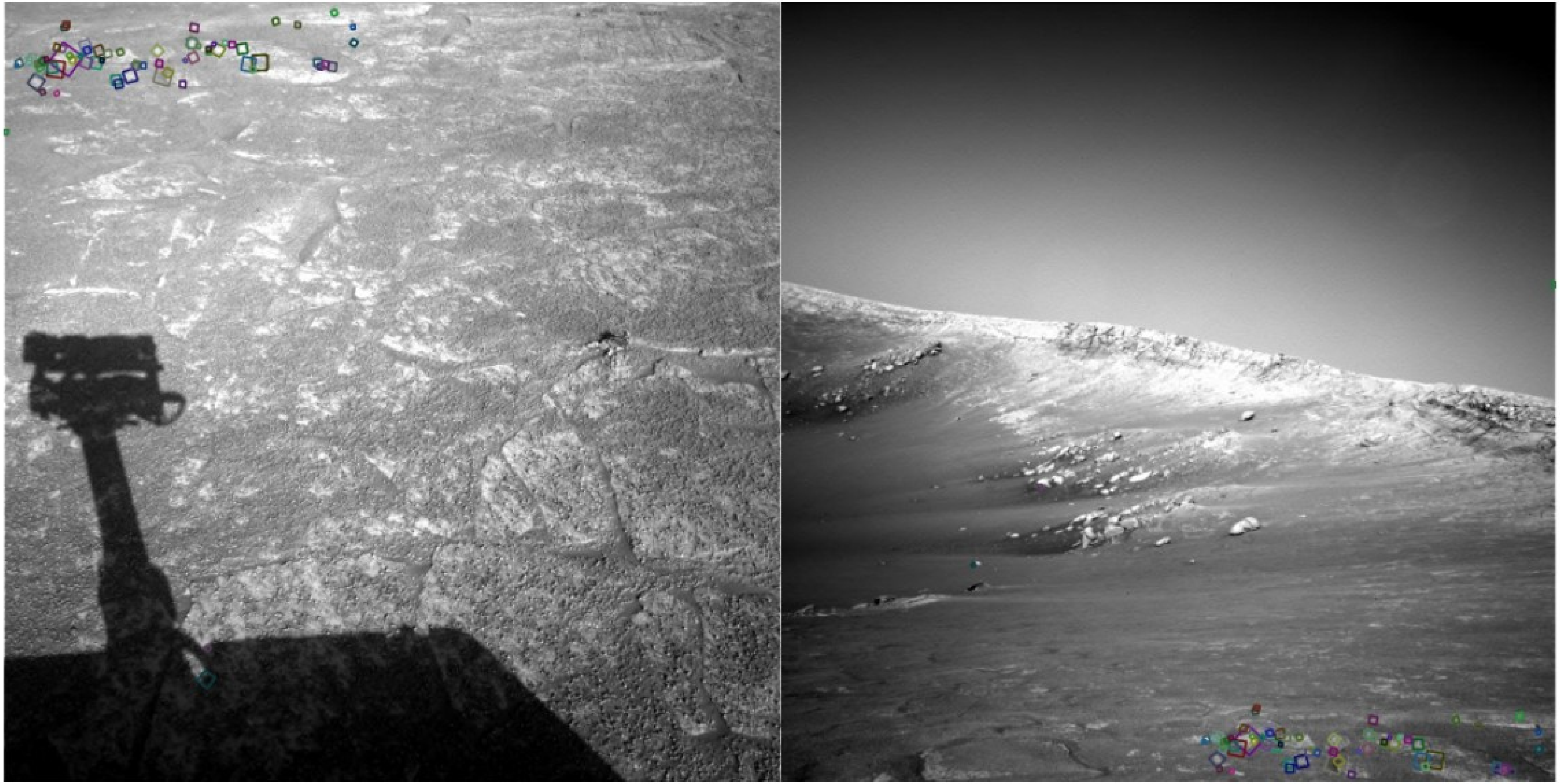
- **CS231A · Computer Vision: from 3D reconstruction to recognition**, Prof. Silvio Savarese – Stanford University
- **CS131 · Computer Vision: Foundations and Applications,** Prof. Fei-Fei Li – Stanford University
- **TTI Chicago: Computer Vision**, Raquel Urtasun, University of Toronto
- **Elementi di Analisi per Visione Artificiale**
  - Paolo Medici http://www.ce.unipr.it/people/medici

Beyond Stereo Vision



NASA Mars Rover images

Beyond Stereo Vision



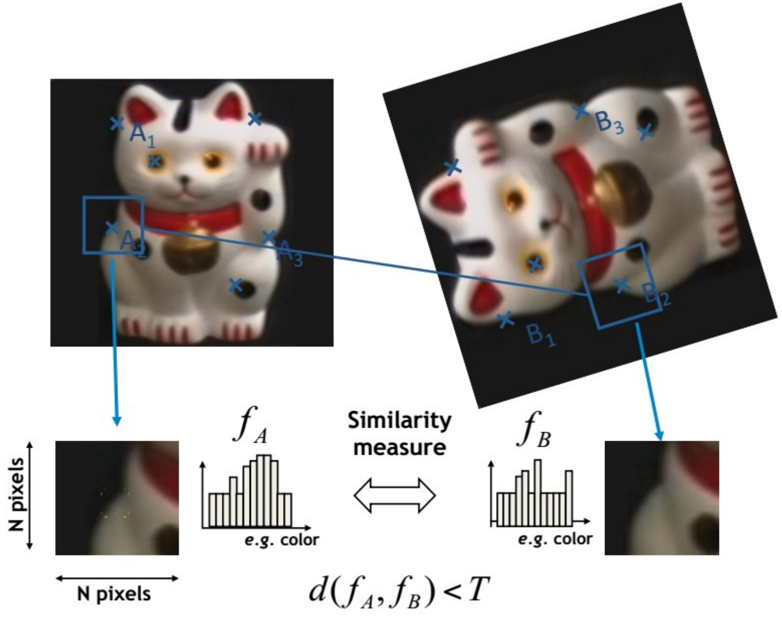
NASA Mars Rover images with SIFT feature matches

Characteristics of a good feature

- **Repeatability:** to be able to find the same features on even very different images
- **Saliency:** the feature contains information
- **Locality:** relatively small area of the image, namely more robust wrt occlusions and clutter

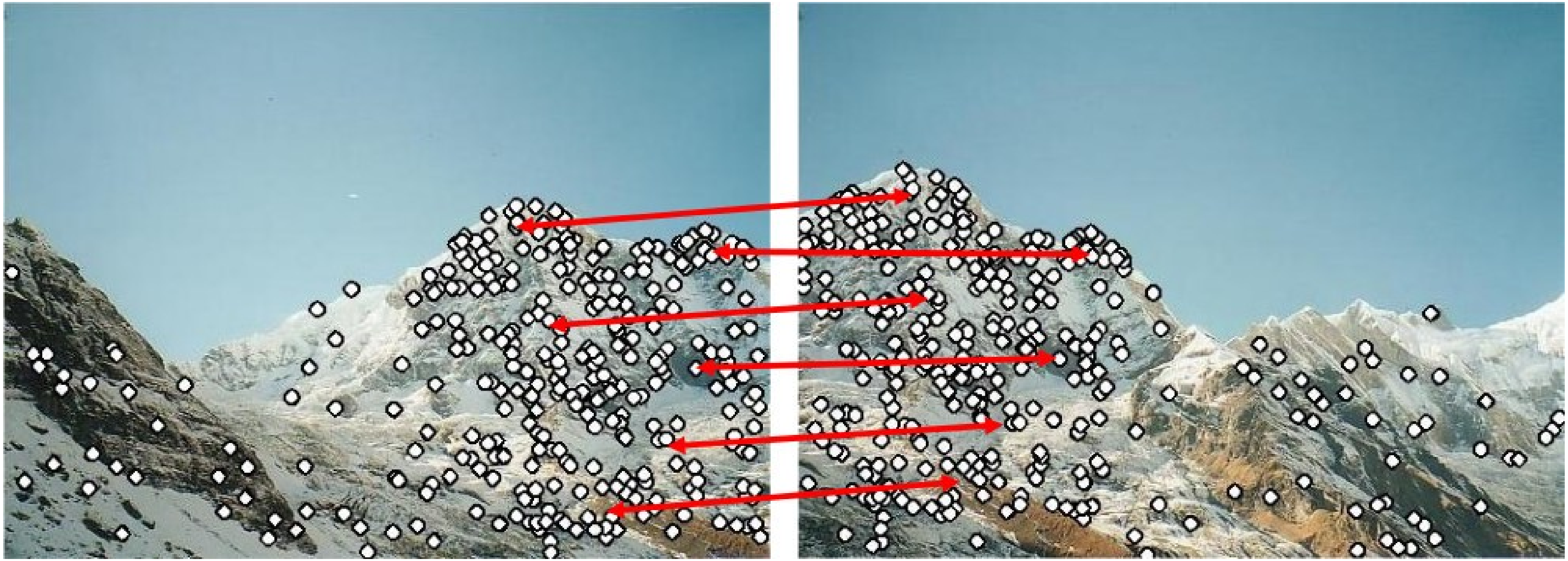
Example recap

- Select specific points → keypoints or features → **features extraction**
- Find potential correspondences → **features matching**
- Application dependent step
  - Matching
  - Indexing
  - Detection
  - Image alignement

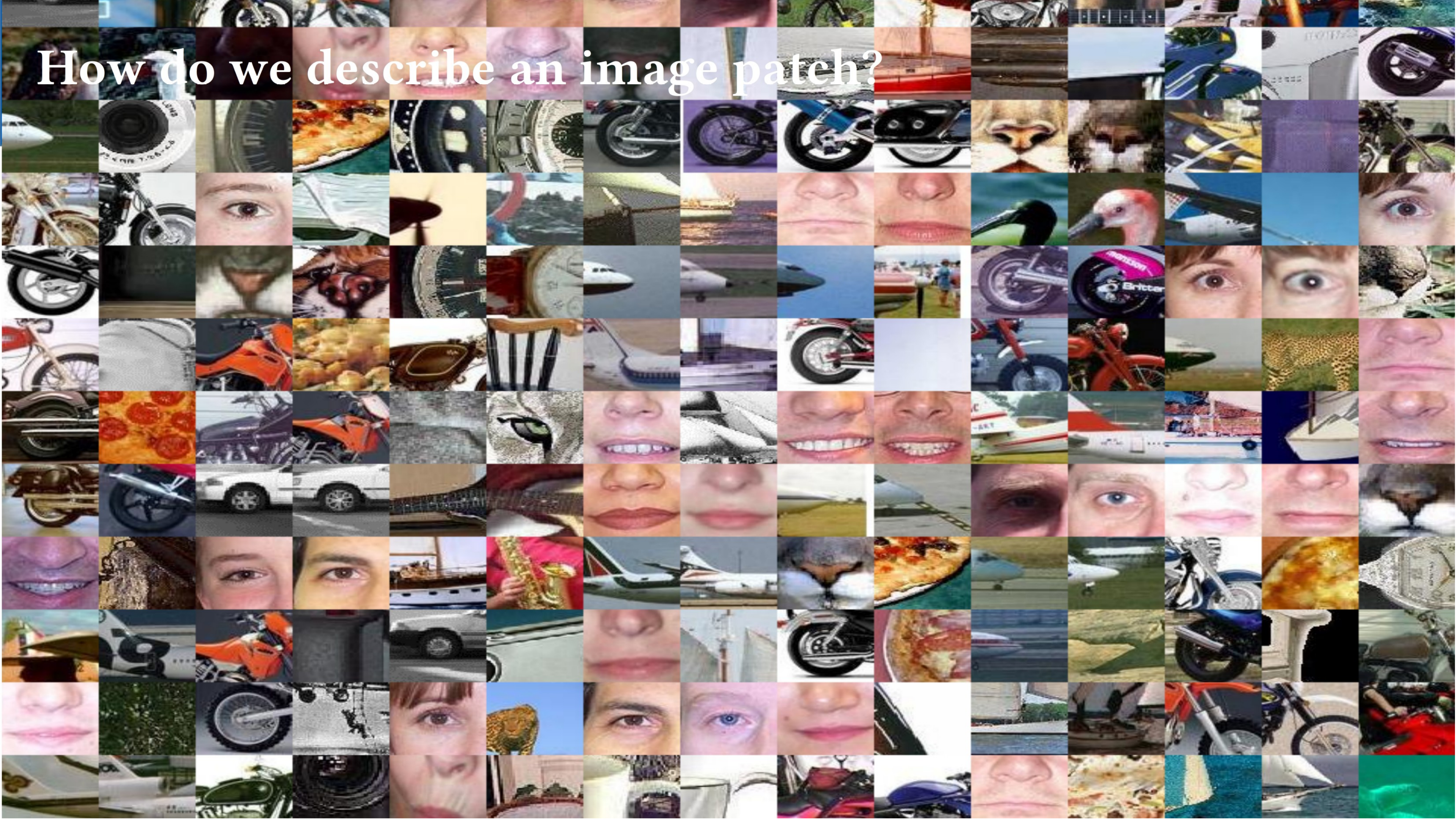


Example

- How to describe keypoints for matching?
  - Keypoints themselves does not work → feature descriptor



How do we describe an image patch?



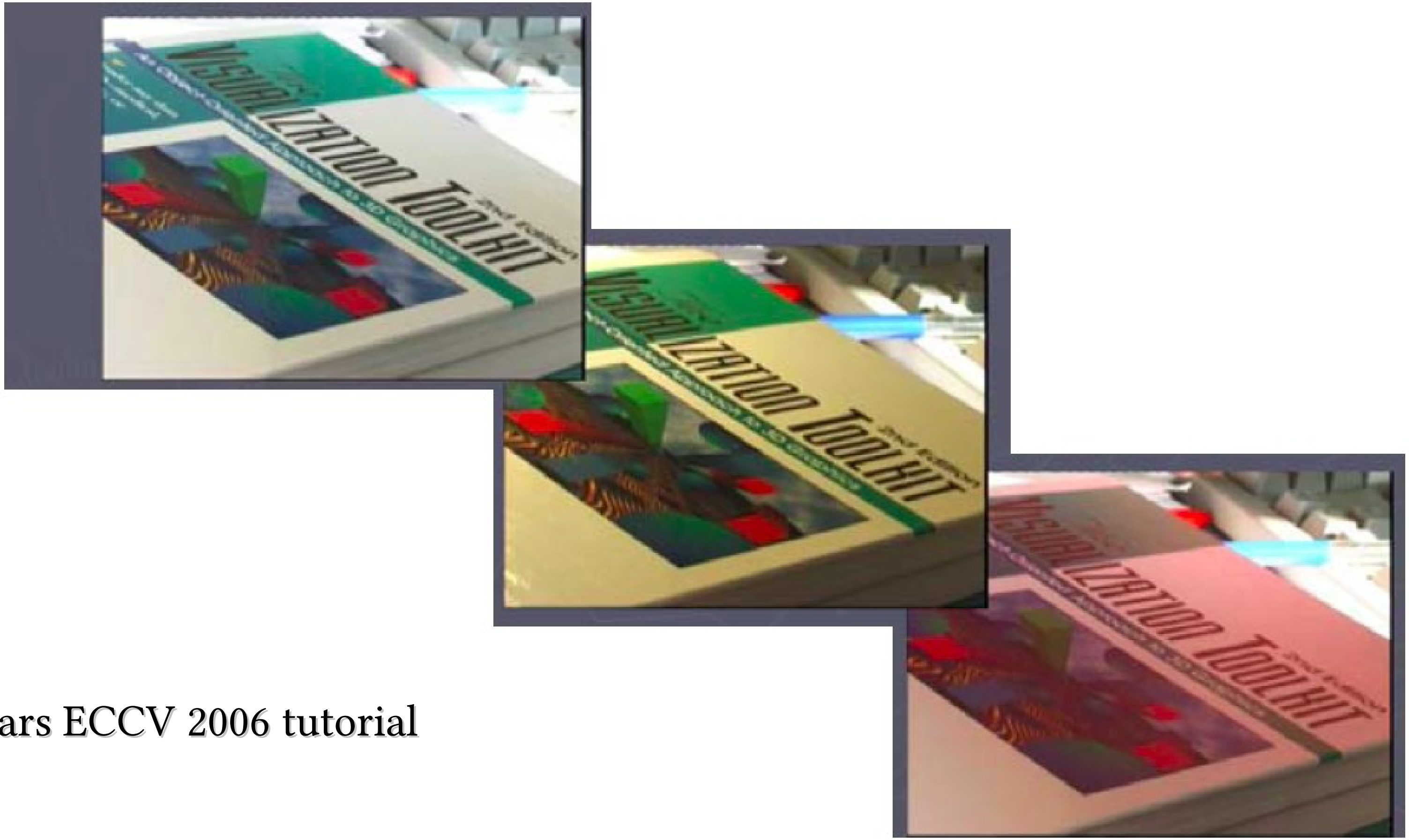
Feature descriptor

- Match should be as much as unique as possible
- Issues
  - Geometrical transformations
    - Translation
    - Scale
    - Rotation
    - …
  - Photometrical transofrmations
    - Illuminations
    - Color
    - …

Orientation, scale (and translation)

e.g. scale,
translation,
rotation

Illumination & color



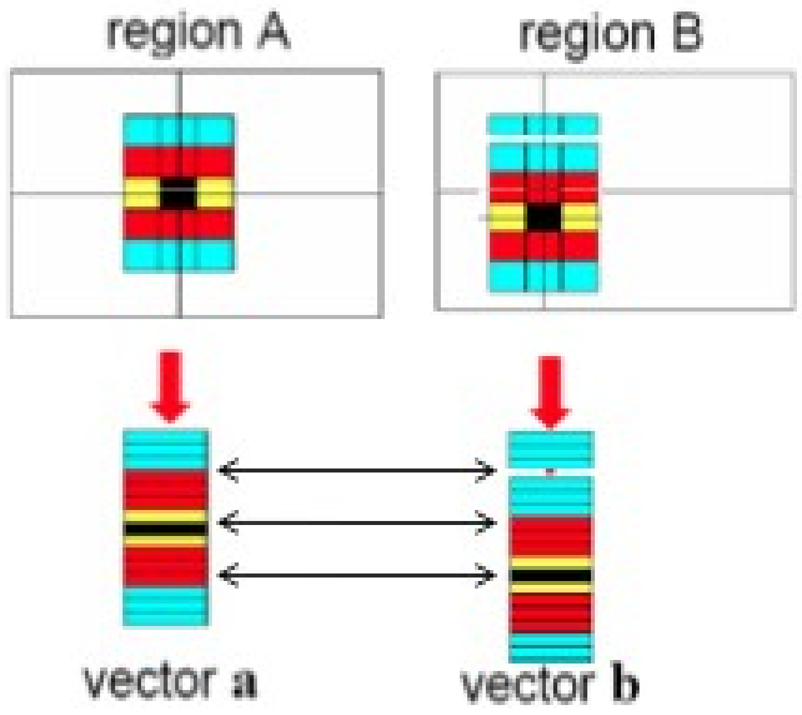
T. Tuytelaars ECCV 2006 tutorial

Feature descriptor

- To match patches we do not use patches themselves
- What is a keypoint descriptor?
  - Keypoint “measure” that can be effectively used for matching
- Properties
  - Invariant/Robust
  - Distinctive
  - Compact
  - Efficient

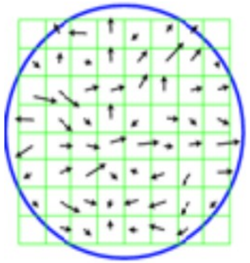
Raw Patches

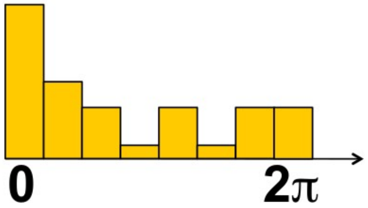
- Create a feature vector, and normalize it
  - Normalization for mean and variance reduce matching problems
- Very sensitive to even small shifts, rotations and any affine transformation

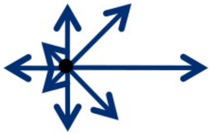


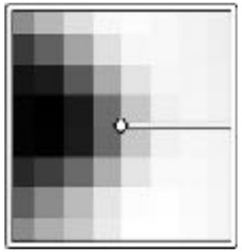
Rotation invariance

- Compute gradients and a histogram of gradients
  - Rotate the patch according to dominant maxima or average







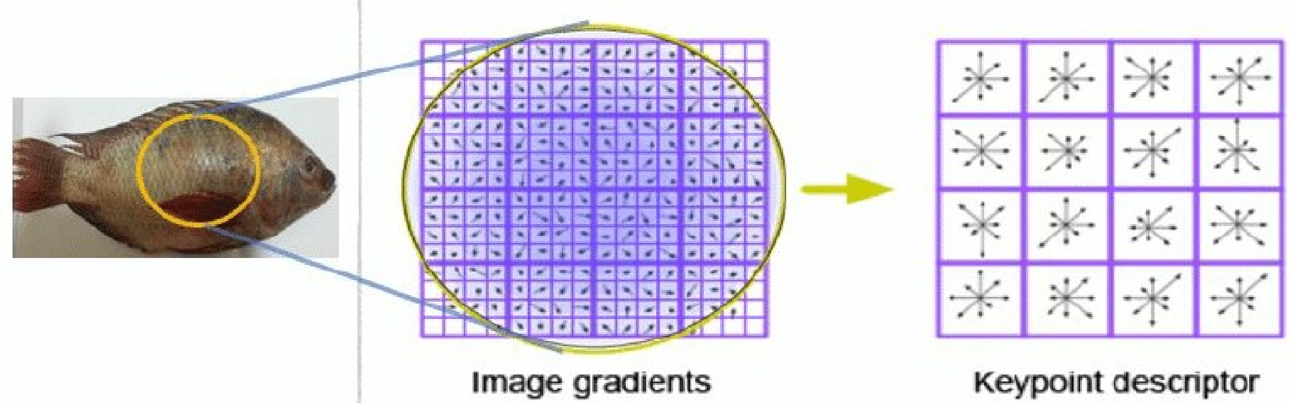


Scale invariance

- For Harris keypoints we used a pyramidal approach
  - i.e. a multiscale window

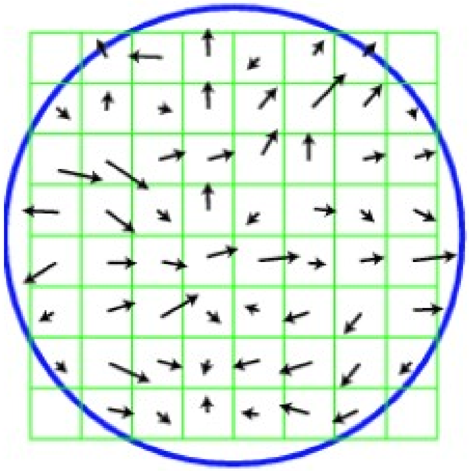
SIFT – Scale Invariant Feature Transform

- Lowe 2004
  - Also keypoints extraction
- Robust wrt geometric transformations
  - Scale and Rotation
- Anyway robust wrt luminance variations
- Unfortunately slow



SIFT – Scale Invariant Feature Transform

- For each keypoint
  - Compute gradient in a specific window around the keypoint
  - Gradient is also downweight using a Gaussian filtering



SIFT – Scale Invariant Feature Transform

- For each keypoint
  - Compute gradient in a specific window around the keypoint
  - Gradient is also downweight using a Gaussian filtering

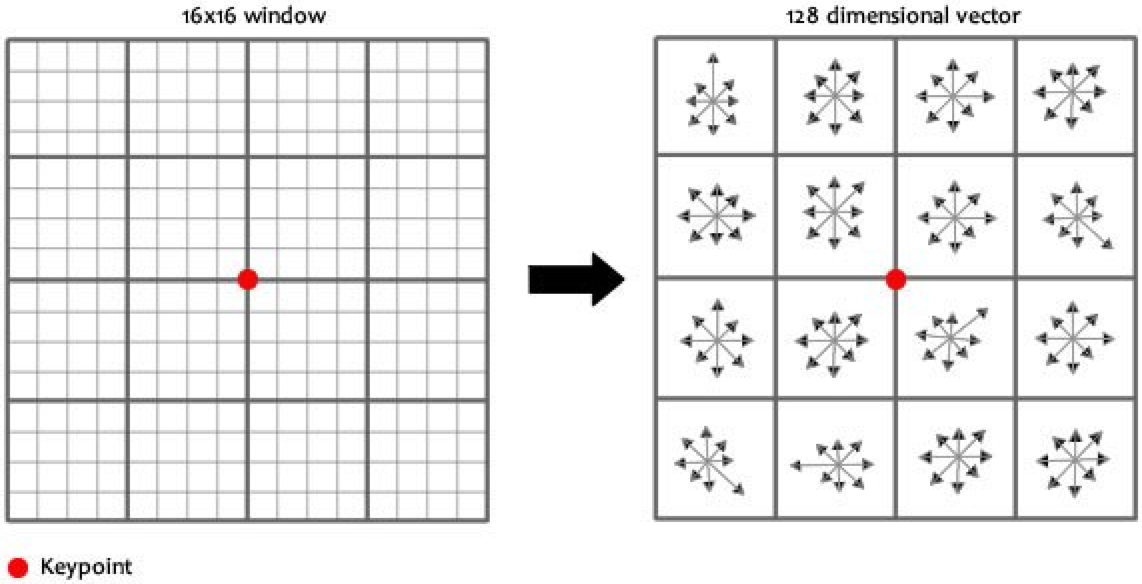
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y)))$$

SIFT – Scale Invariant Feature Transform

- In the gradient window:
  - Compute a gradient orientation in the 4×4 quadrant



SIFT – Scale Invariant Feature Transform

example 8x8 locations 2x2
histogram array

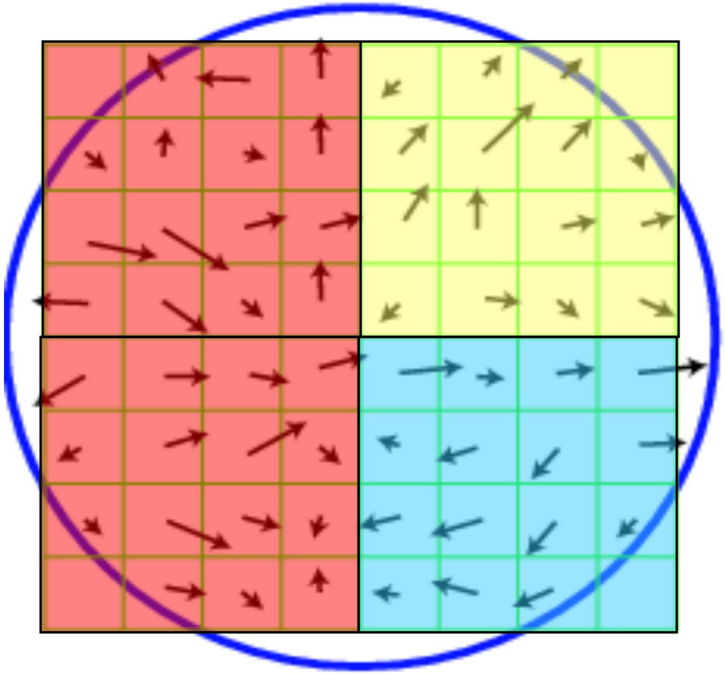
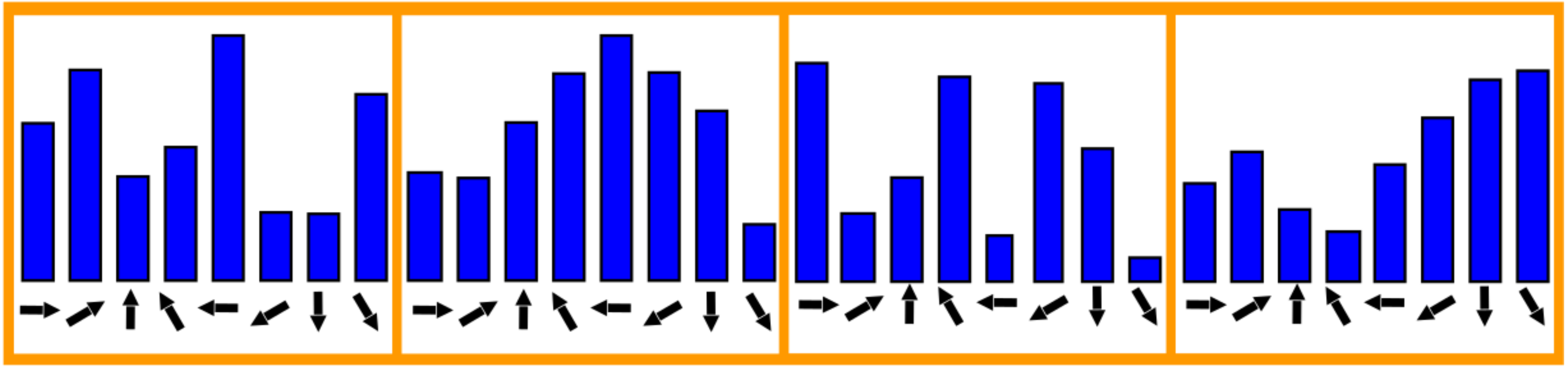


Image gradients

Keypoint descriptor

SIFT – Scale Invariant Feature Transform

- In the gradient window:
  - Compute a gradient orientation in each 4×4 quadrant
  - Accumulate results in histogram bins (8 in the figure)

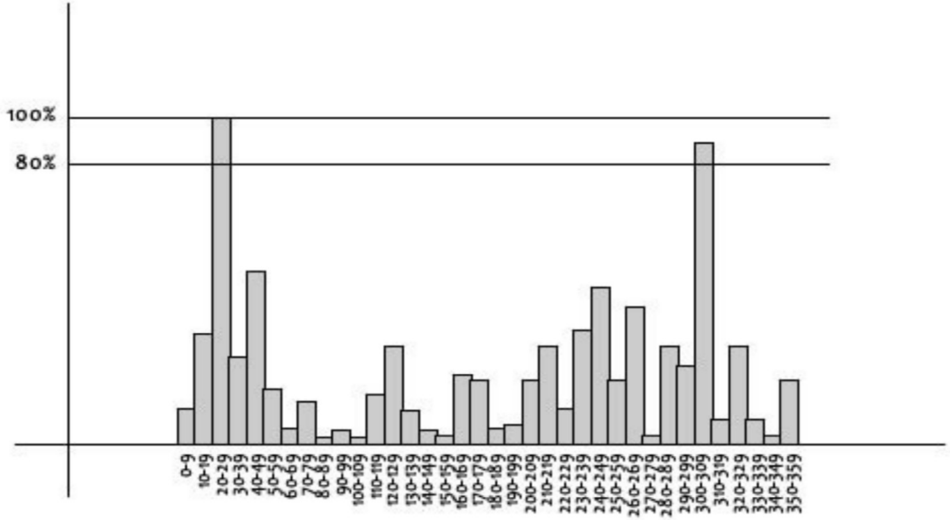


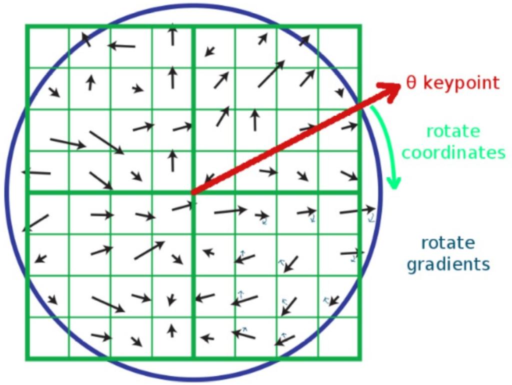
SIFT – Scale Invariant Feature Transform

- We showed different situations!
  - Where is the truth?
- Assume that we bin "o" orientations over "$k \times k$" quadrants, where each quadrant is "$s \times s$" pixels
- What's final dimensionality of descriptor?
  - Common implementation: o=8, k=4, s=4 $\longrightarrow$ 128

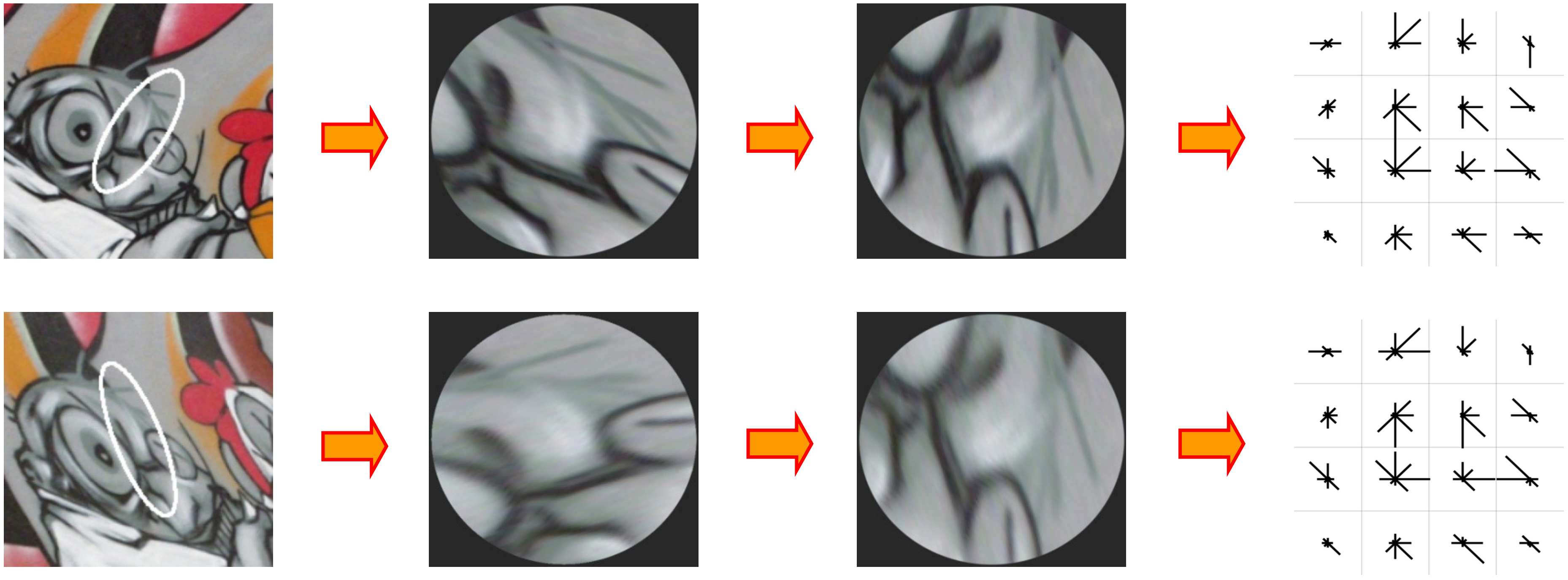
SIFT – Scale Invariant Feature Transform

- Maximum gaves us info about keypoint orientation
  - Optionally we also consider other potential orientation when we have other values close to maximum
    - Namely we can have more than one descriptor





SIFT – Scale Invariant Feature Transform



SIFT – Scale Invariant Feature Transform

- How to obtain invariance wrt contrast?
  - Rescale to have unit norm ⟶ $x = \frac{x}{||x||} \quad x \in \mathbb{R}^{128}$
- How to obtain invariance wrt luminance?
  - Do you remember we used gradients?
  - Anyway values are “clipped” to 0.2 to improve robustness to photometric variations
- How to obtain invariance wrt rotation?
  - We used the whole patch orientation to adjust x

SIFT – Scale Invariant Feature Transform

- Result is then a 128 dim vector
  - Float values
- How to match them?
  - Euclidean distance can be effectively used
- See OpenCV example

SIFT – Scale Invariant Feature Transform

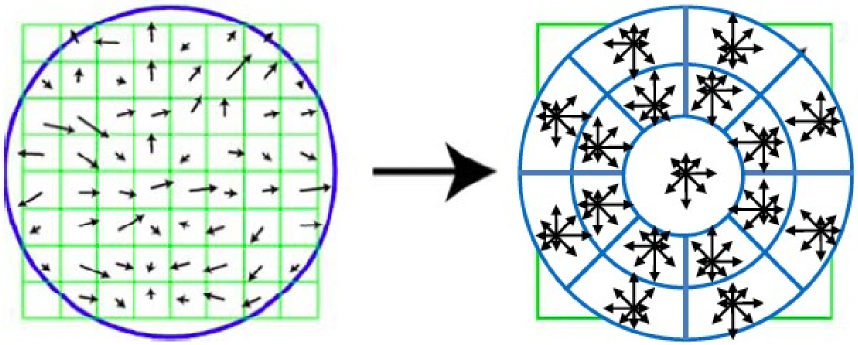
- Summary
  - Extraordinarily robust matching technique
  - Sometimes matches night vs day
  - Lots of code available

PCA SIFT

- Principal Component Analysis
  - Reduce the dimensionality of a data set
  - From 128 to 10 or so
  - A linear transformation (eigendecomposition) is used to extract principal components
  - Needs training!

GLOH

- Gradient location-orientation histogram
  - Uses a log-polar binning structure instead of quadrants.
  - Exploits 17 spatial bins and 16 orientation bins.
  - PCA (trained on a large dataset) is used to reduce the 272 space to a 128 one



(a) image gradients (b) keypoint descriptor

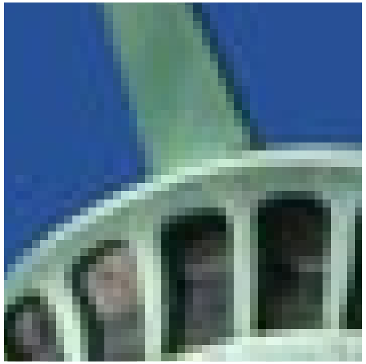
SURF

- Speeded Up Robust Feature
  - Again similar to SIFT
  - 20×20 patch subdivided in 4×4 quadrants
  - 64D descriptor
  - As stable as SIFT but several times faster than SIFT

Both SIFT and SURF are patented algorithms!

Binary Descriptors

- Using SIFT, SURF… we obtain “floating point” values
  - Match have to cope with this
  - Binary values can be faster to be matched



[0, 0, 1, 0, 1, 1, 0, 1, …]

Binary Descriptors

- BRIEF
  - Binary Robust Independent Elementary Features
- FREAK
  - Fast Retina Keypoint
- BRISK
  - Binary Robust Invariant Scalable Keypoints
- ORB
  - Oriented FAST and Rotated BRIEF

BRIEF

- After a gaussian smoothing of patch, several intensity comparisons between pair of pixels are performed
- Each pair is selected according to a specific pattern
  - Anyway the same pattern is used for all patches

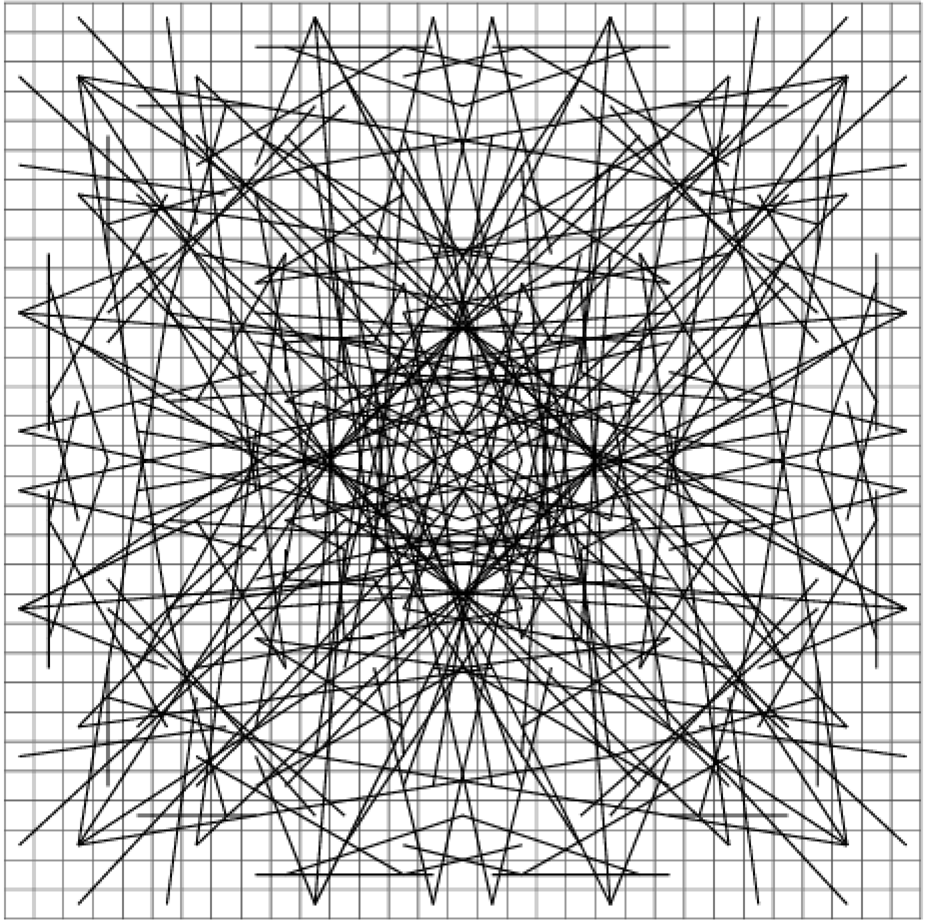
Binary test

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

BRIEF descriptor

$$f_{n_d}(p) := \sum_{1 \le i \le n_d} 2^{i-1} \tau(p; x_i, y_j)$$

BRIEF: pattern example



BRIEF

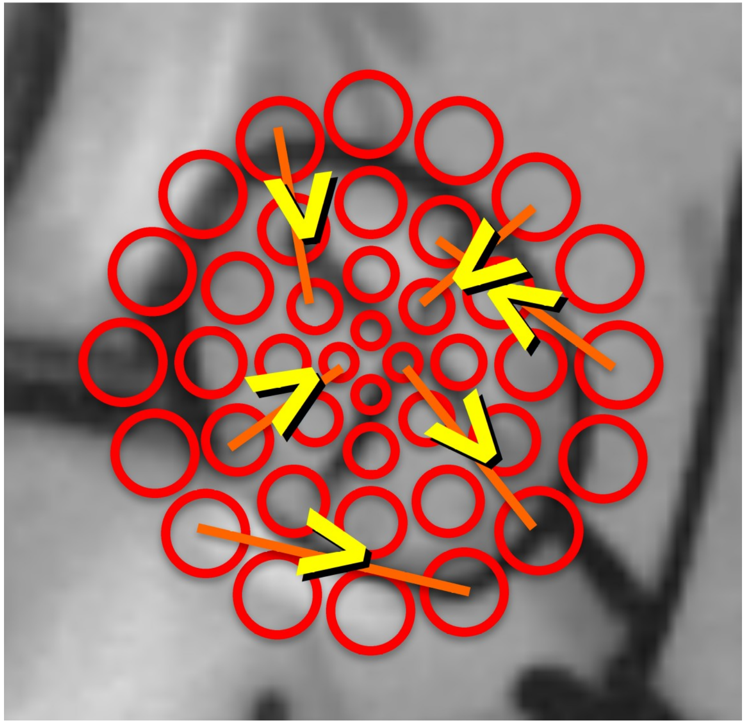
- Different strategies for sampling geometry
  - Random
  - Uniform: random pairs having anyway a fixed distance
  - Gaussian: gaussian distributions around the keypoint
  - Gaussian 2: first point selected as previous strategy, but second one as gaussian distribution centered on the former
  - Polar: simmetry wrt keypoint

BRIEF

- Pro
  - Simple intensity comparison
  - Definitely fast
  - Good results
- Cons
  - Not very reliable against rotations
- Solution $\longrightarrow$ ORB
  - Oriented FAST and rotated BRIEF

BRISK

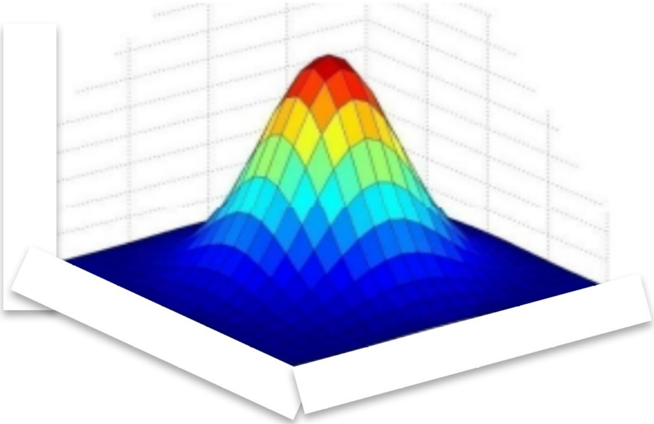
- Binary Robust Scalable Keypoints

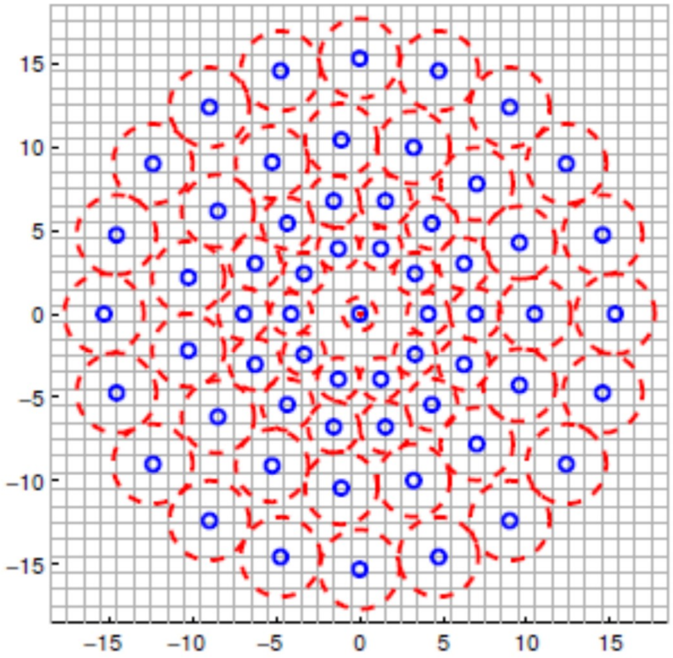


BRISK

- Same approach as BRIEF but given pattern

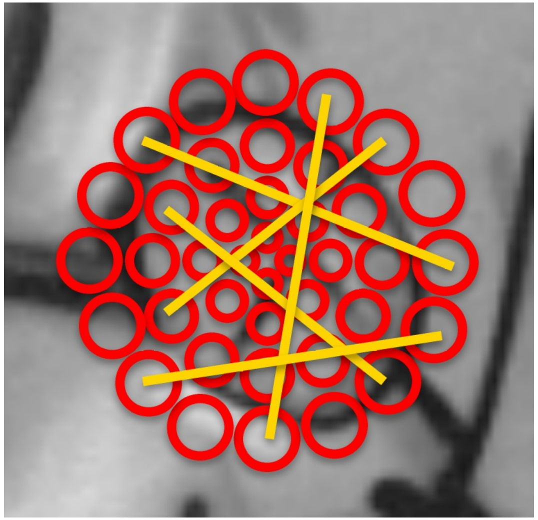
2D Gaussian around each sampling point





BRISK

- A number of sampling pairs is selected
- For each pair the gradient is computed betwwen the 2 smoothed intensity
- Different pairs are used for computing descriptor and patch orientation



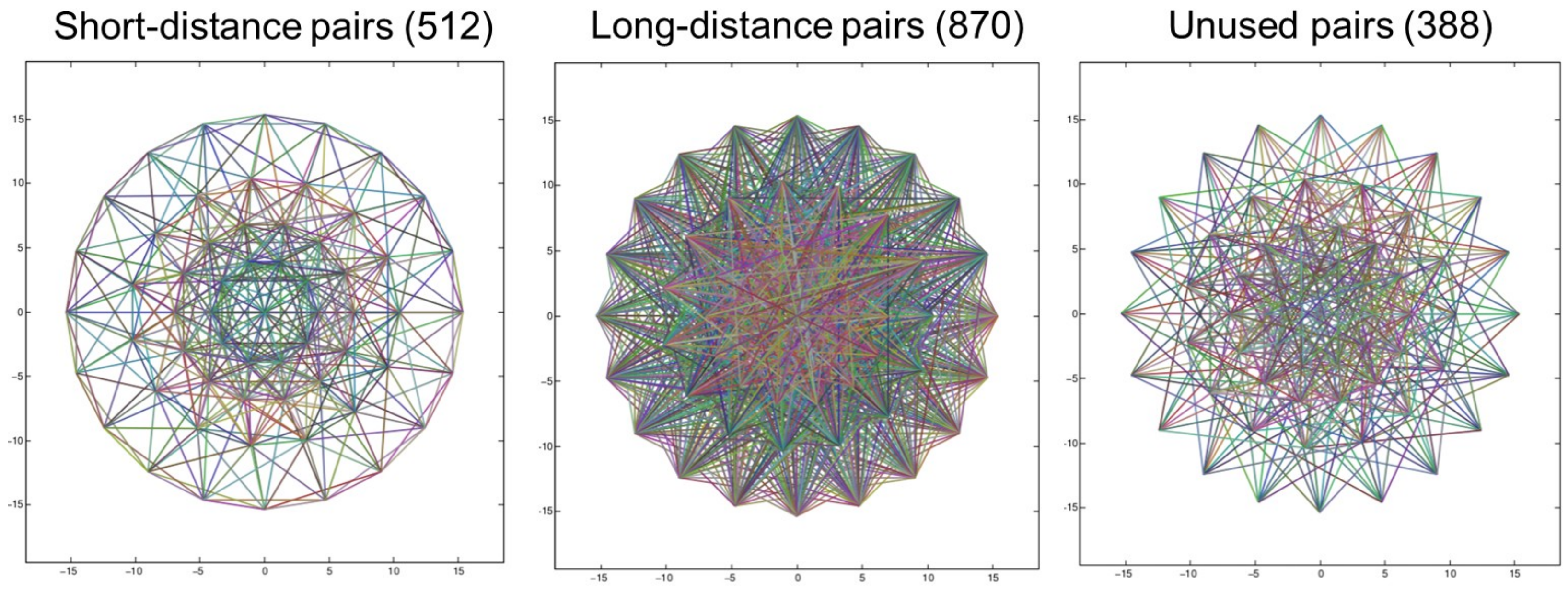
$$\boldsymbol{g}(\boldsymbol{p}_i, \boldsymbol{p}_j) = \underbrace{\frac{(\boldsymbol{p}_j - \boldsymbol{p}_i)}{\|\boldsymbol{p}_j - \boldsymbol{p}_i\|}}_{\text{unit vector}} \cdot \underbrace{\frac{I(\boldsymbol{p}_j, \sigma_j) - I(\boldsymbol{p}_i, \sigma_i)}{\|\boldsymbol{p}_j - \boldsymbol{p}_i\|}}_{\text{gradient magnitude}}$$

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L} \sum_{p_i, p_j \in L} g(p_i, p_j)$$

BRISK

UNIVERSITÀ DI PARMA

- Short distance pairs → descriptor → $I(p_j,\sigma_j) > I(p_i,\sigma_i)$
- Long distance pairs → orientation → $\arctan2(g_y,g_x)$



BRISK Repeatability

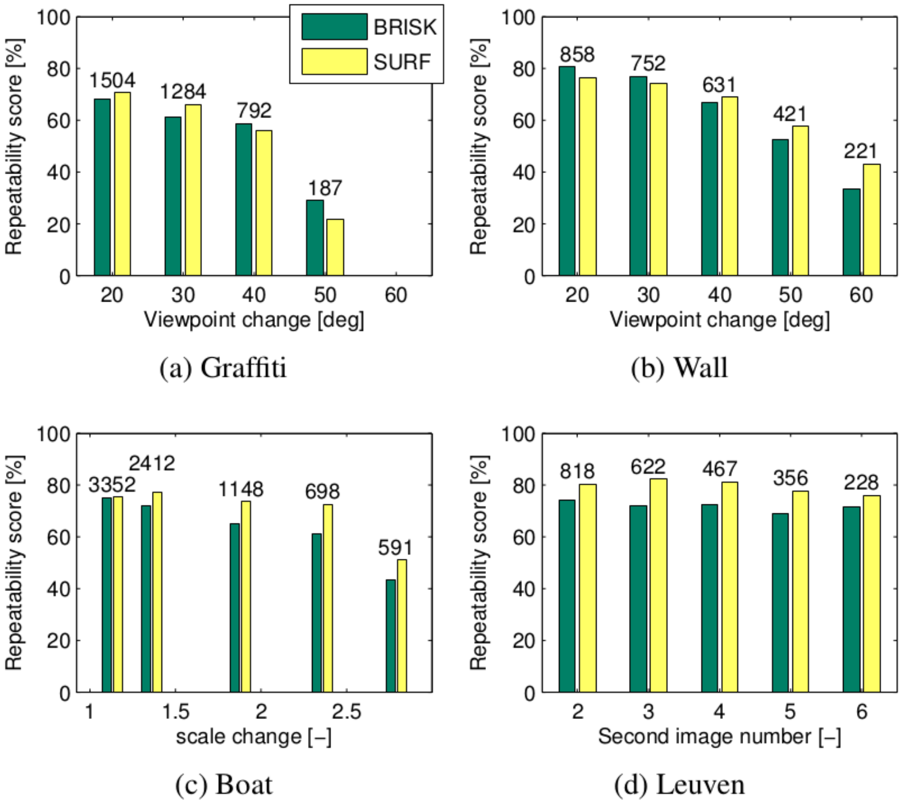
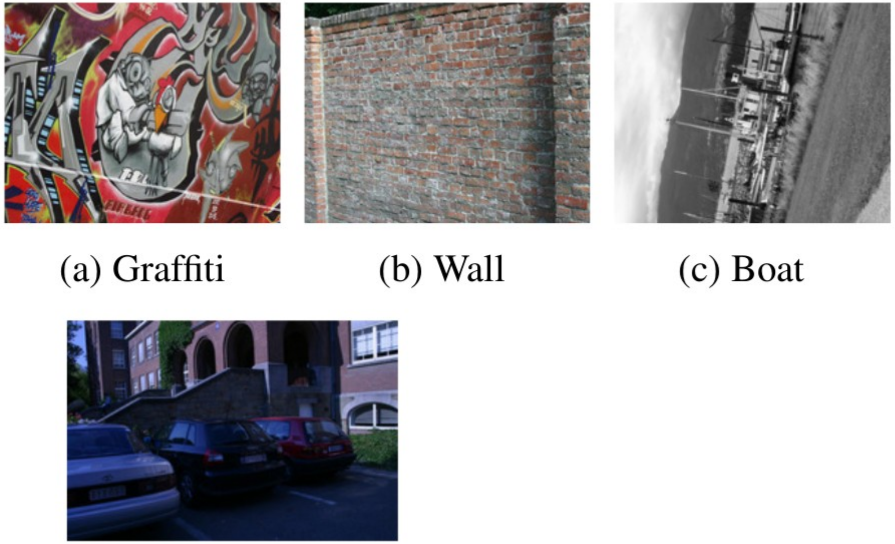


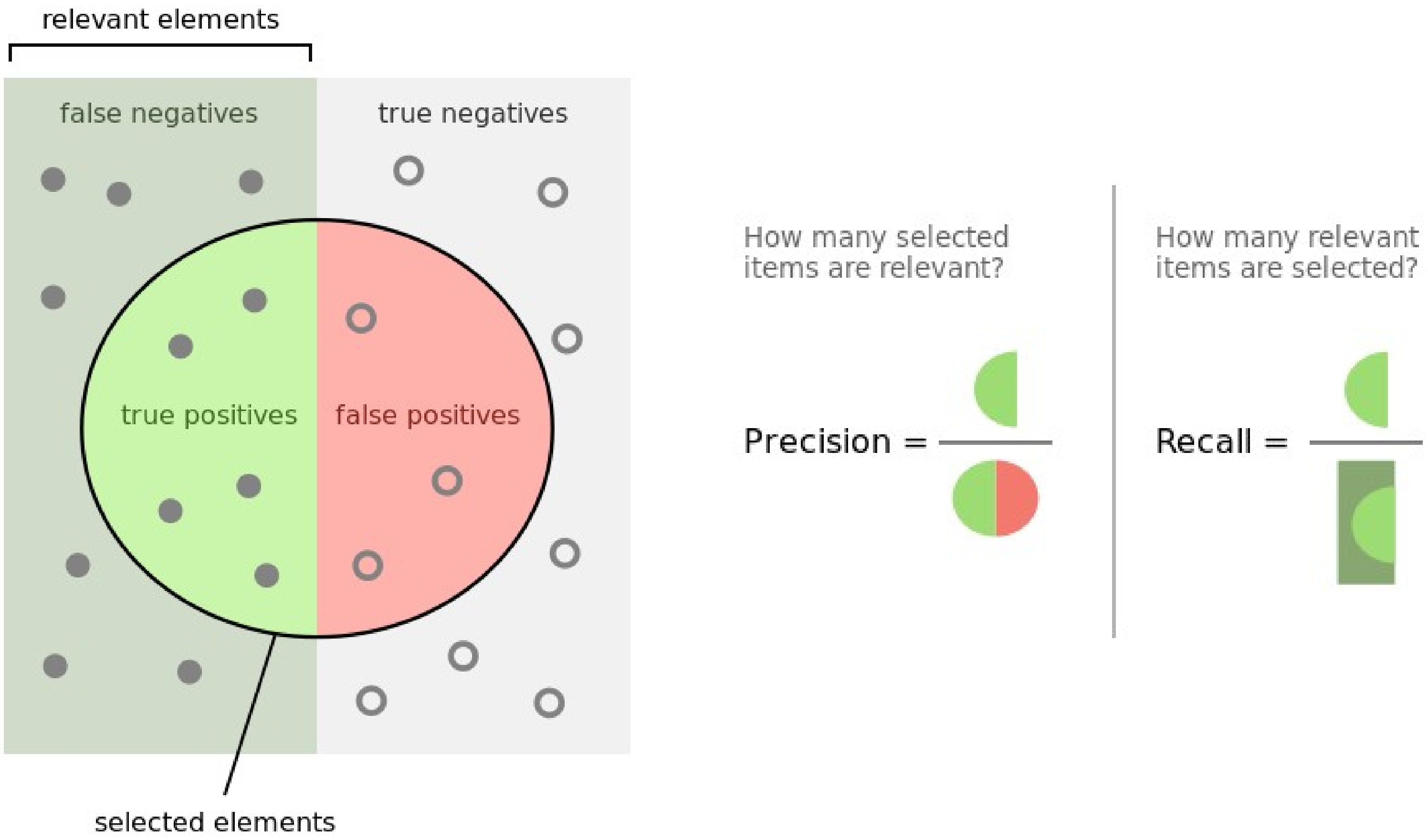
Figure 5. Repeatability scores for 50% overlap error of the BRISK and the SURF detector. The resulting similarity correspondences (approximately matched between the detectors) are given as numbers above the bars.



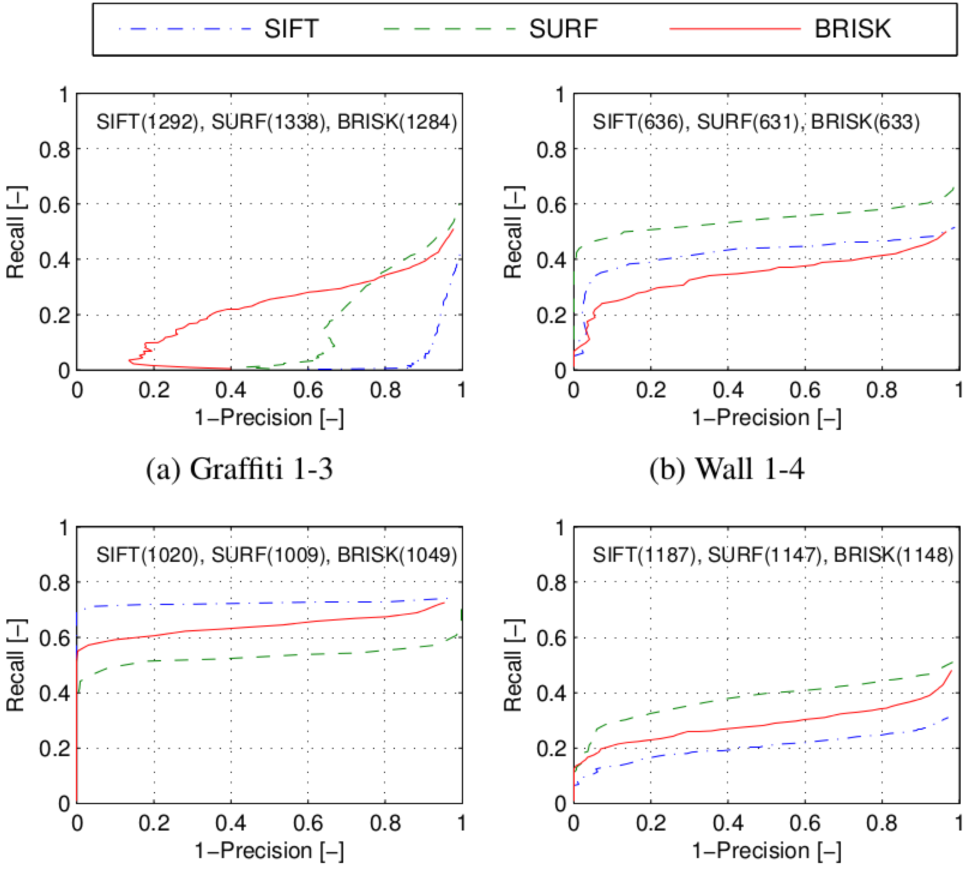
(a) Graffiti (b) Wall (c) Boat

Leuven

Precision-Recall



Overall Algorithm Results



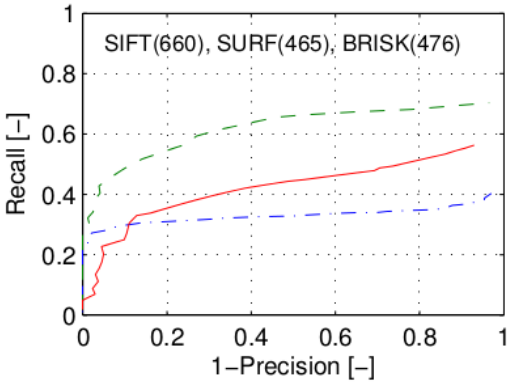
(a) Graffiti 1-3

(b) Wall 1-4

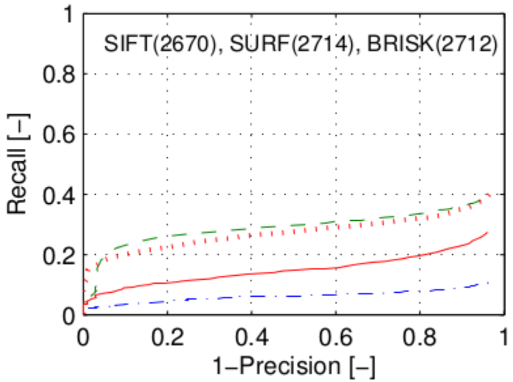
(c) Image Rotation of 60° on Wall 1.

(d) Boat 1-4

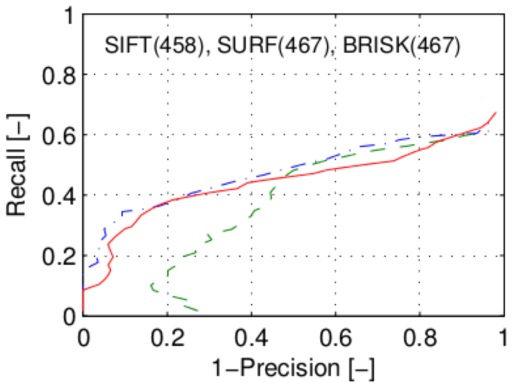
Overall Algorithm Results



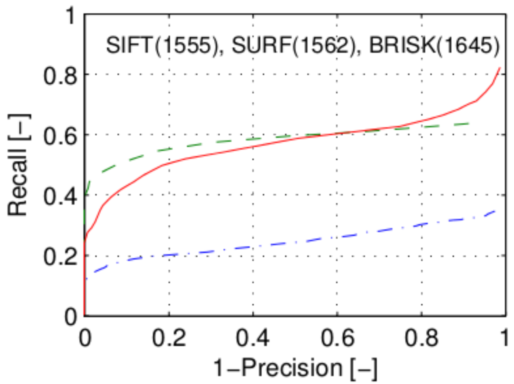
(e) Bikes 1-4



(f) Trees 1-4



(g) Leuven 1-4



(h) Ubc 1-4

Timings

UNIVERSITÀ DI PARMA

- Scalable for faster execution by
  - Reducing number of sampling pairs
  - Examining only one scale
  - Omitting pattern rotation step

	SIFT	SURF	BRISK
Detection threshold	4.4	45700	67
Number of points	1851	1557	1051
Detection time [ms]	1611	107.9	17.20
Description time [ms]	9784	559.1	22.08
Total time [ms]	11395	667.0	39.28
Time per point (ms)	**6.156**	**0.4284**	**0.03737**

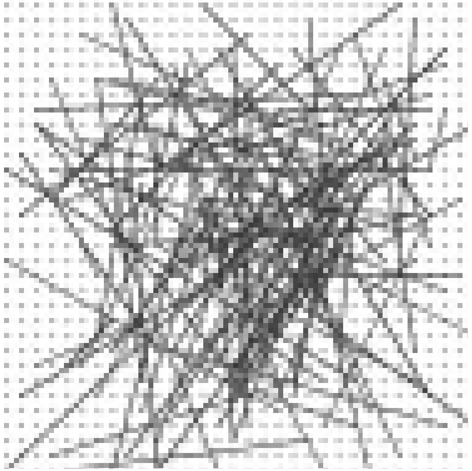
Table 1. Detection and extraction timings for the first image in the Graffiti sequence (size: 800×640 pixels).

	SIFT	SURF	BRISK
Points in first image	1851	1557	1051
Points in second image	2347	1888	1385
Total time [ms]	291.6	194.6	29.92
Time per comparison [ns]	**67.12**	**66.20**	**20.55**

Table 2. Matching timings for the Graffiti image 1 and 3 setup.

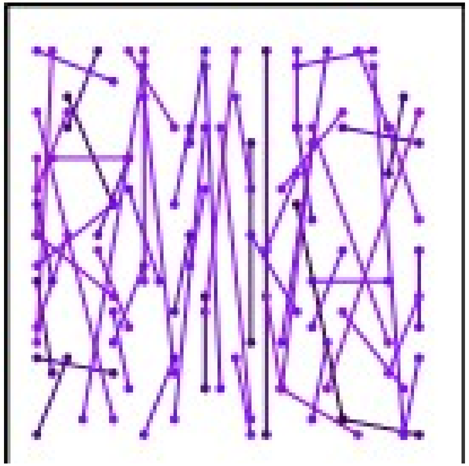
Binary Descriptors Summary

BRIEF



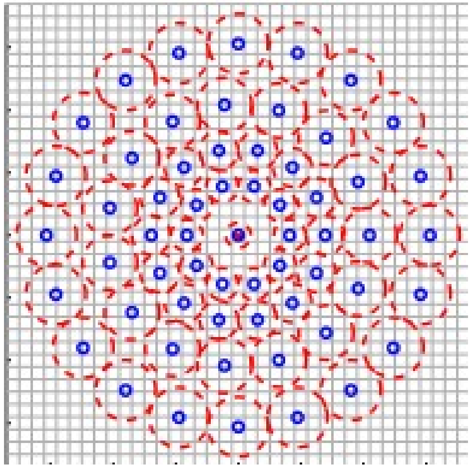
- Efficient

ORB

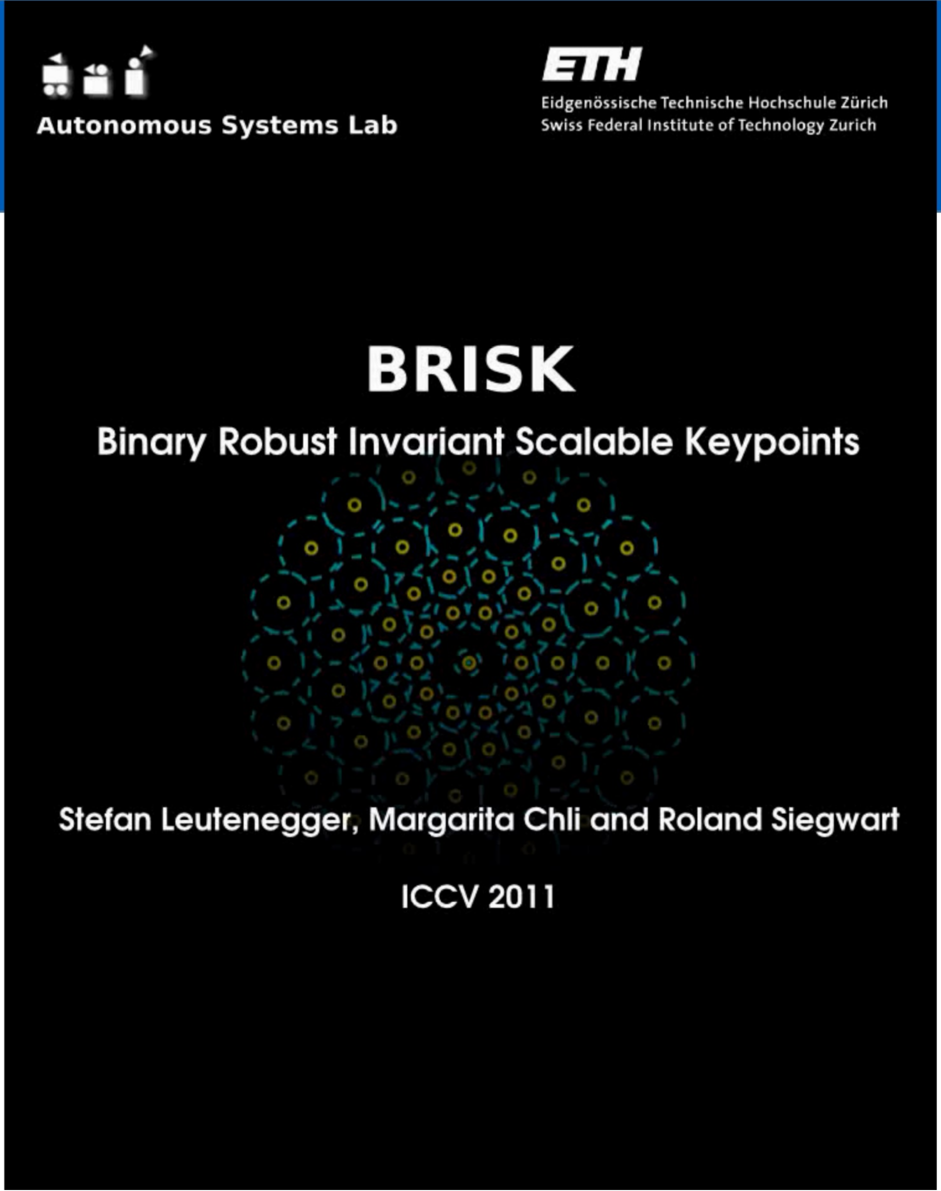


- Efficient
- Rotation

BRISK

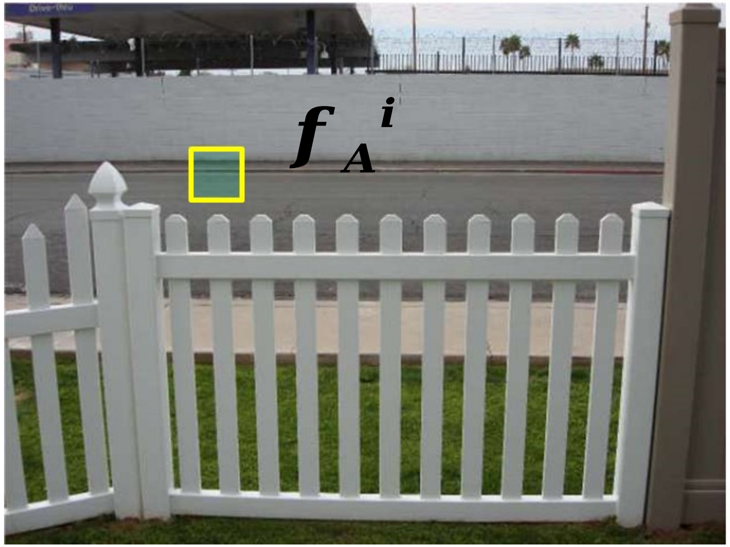


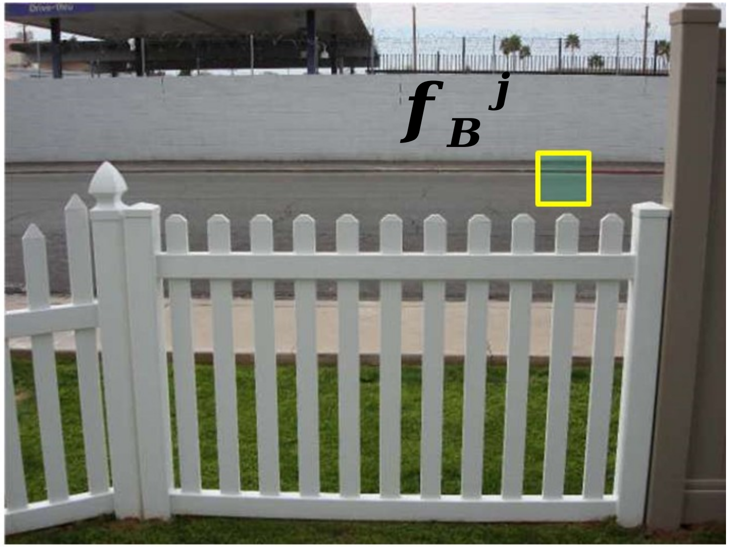
- Efficient
- Rotation
- Scale
- Noise


Autonomous Systems Lab
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
BRISK
Binary Robust Invariant Scalable Keypoints
Stefan Leutenegger, Margarita Chli and Roland Siegwart
ICCV 2011

Matching

- We found keypoints
- We compute descriptors
- And now?





Matching

- For each descriptor i of image A
  - Measure "distance" with each descriptor j of image B
  - Get the <u>two</u> best matches
  - Compare them against a threshold
    - Above ⟶ no match
    - Below ⟶ compute the ratio distance

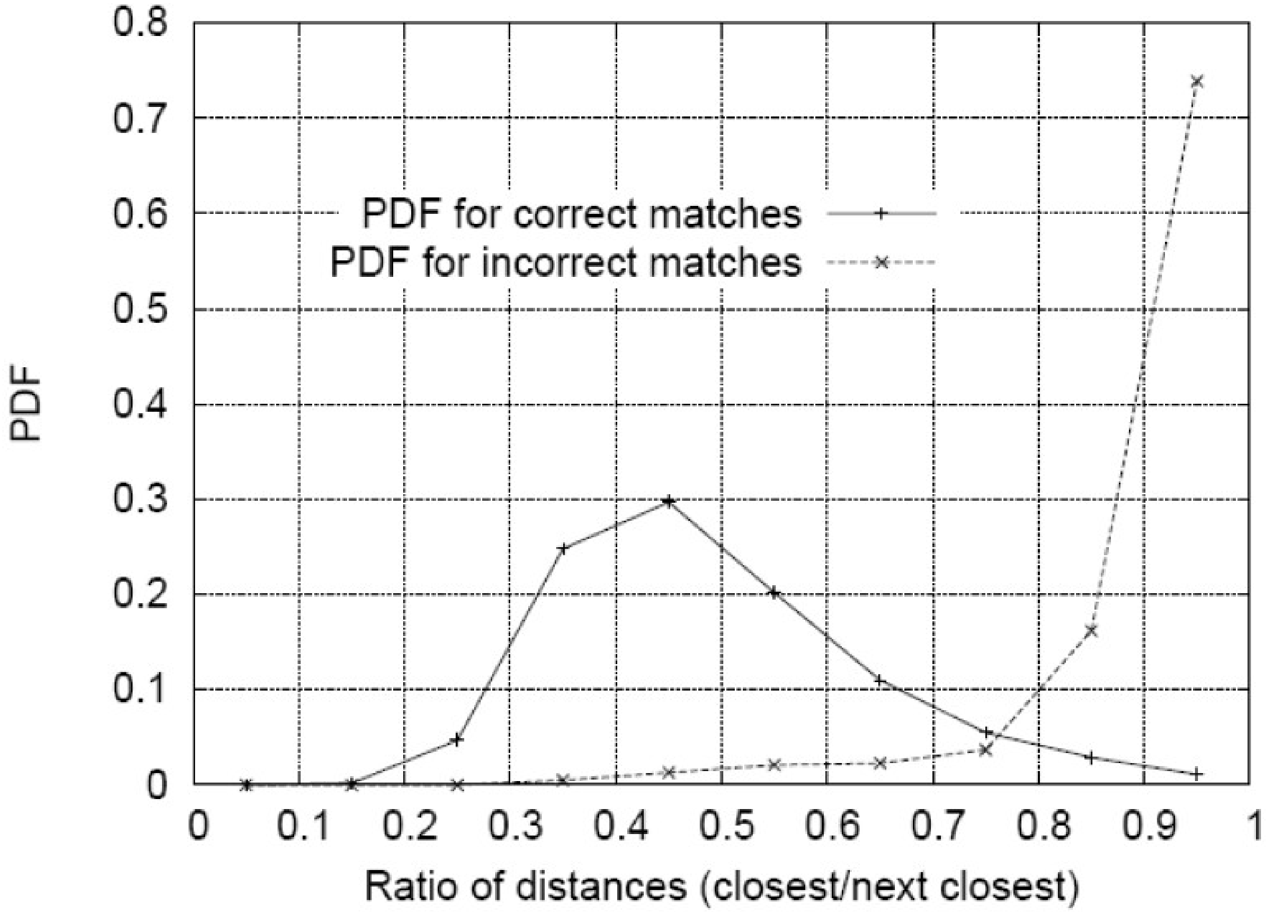
$$\frac{dist\left(f_A^i, f_B^{best}\right)}{dist\left(f_A^i, f_B^{2^{nd}\,best}\right)}$$

Matching

- When ratio distance ~ 1 ⟶ no match
  - Matches are too similar ⟶ ambiguity
- When ratio distance << 1 ⟶ we have a match
  - First match outperforms 2^{nd} one (and others)

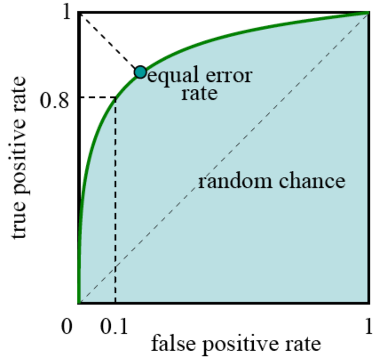
$$\frac{dist\left(f_A^i, f_B^{best}\right)}{dist\left(f_A^i, f_B^{2^{nd}\,best}\right)}$$

Matching (Probability Density Function)



Matching

- Other thresholdes have to be selected, how?
  - precision/recall analysis
  - empirically



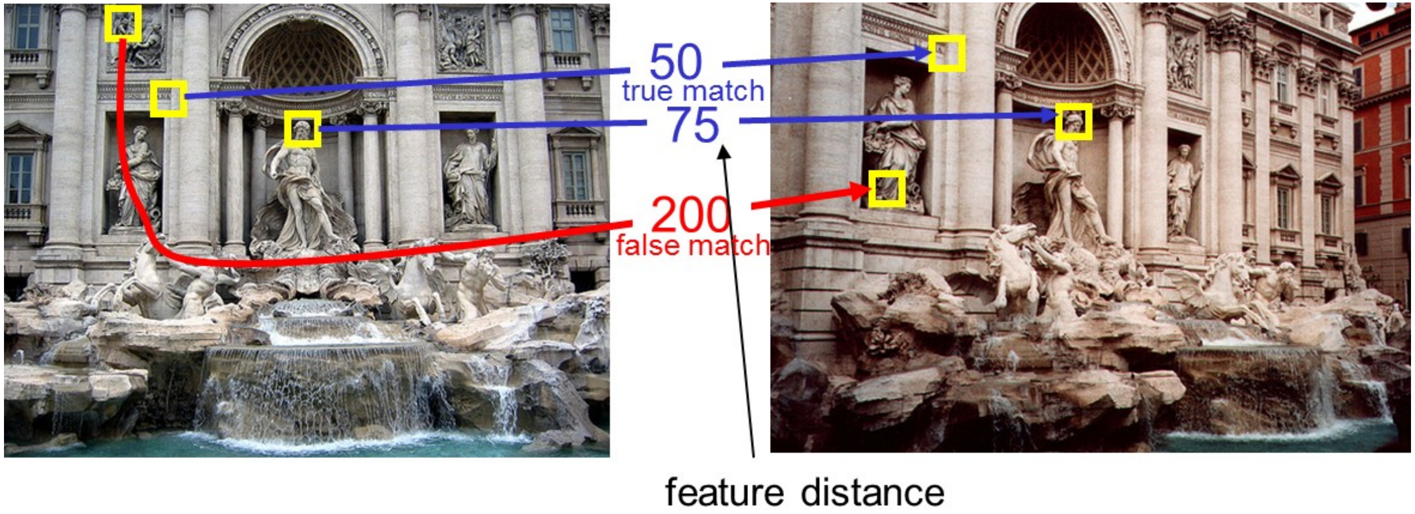


Image Stitching

UNIVERSITÀ DI PARMA

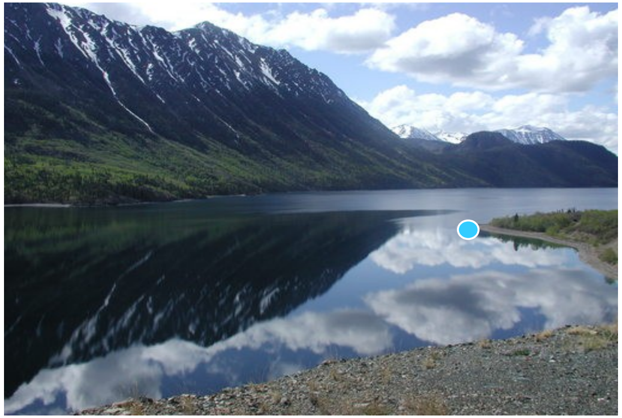
- Combine multiple images in a single view

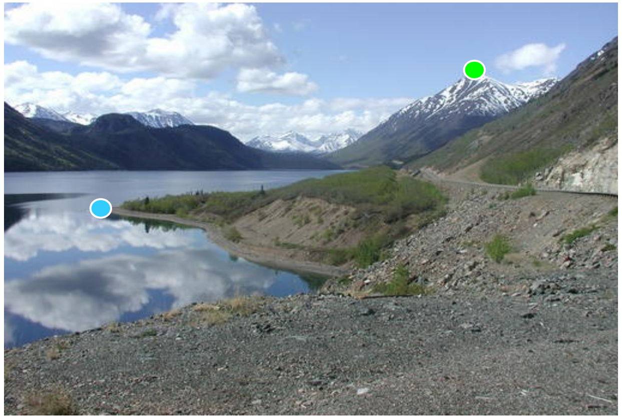
Image Stitching

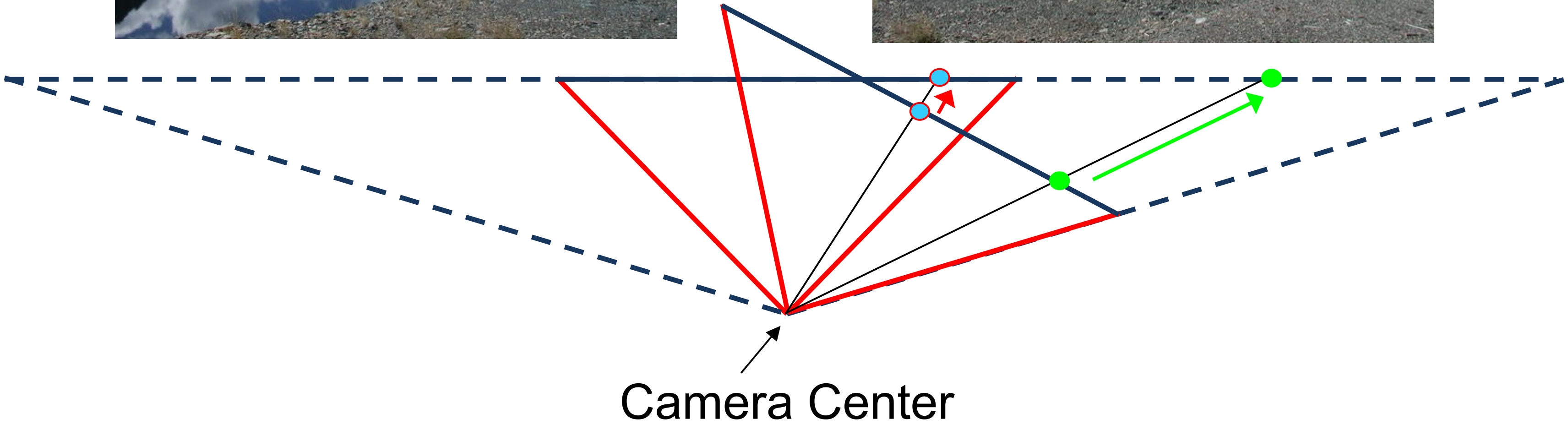
- Main steps
  - Acquire multiple images
  - Get transformation from first image to following one
  - Reproject that image onto the previous one
  - Blend them in the result
  - Until we have images → repeat

Image Stitching

- Main steps
  - Acquire multiple images
  - Get transformation from first image to following one
  - Reproject that image onto the previous one
  - Blend them in the result
  - Until we have images → repeat

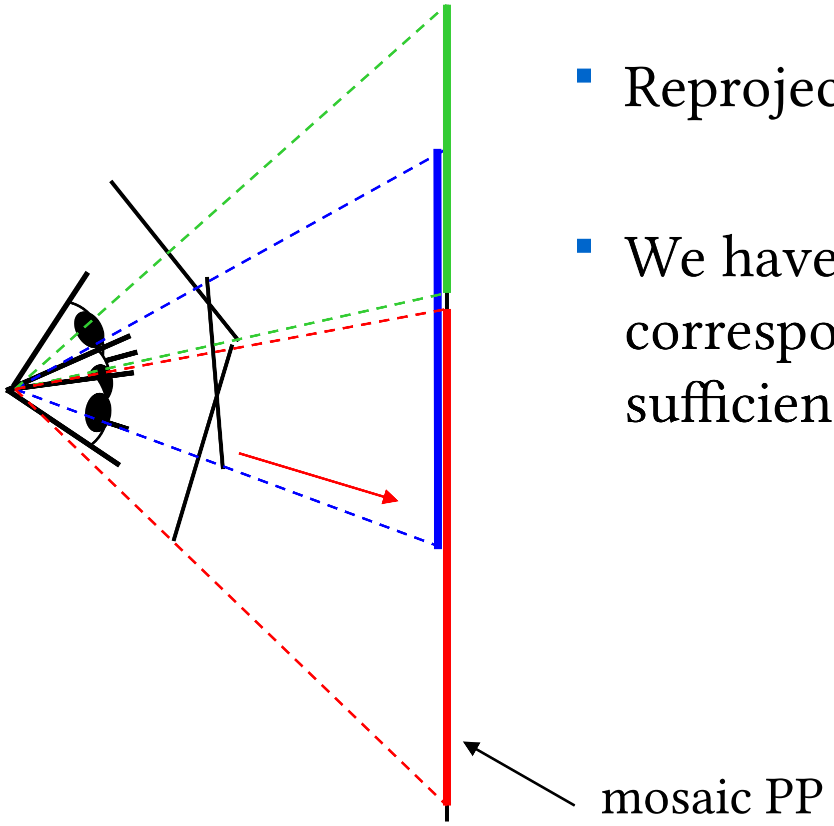






Camera Center

Image Stitching



- Reproject images on a common plane
- We have a “virtual” camera that correspondes to that plane with a FOV sufficiently large to include all other views

Homography

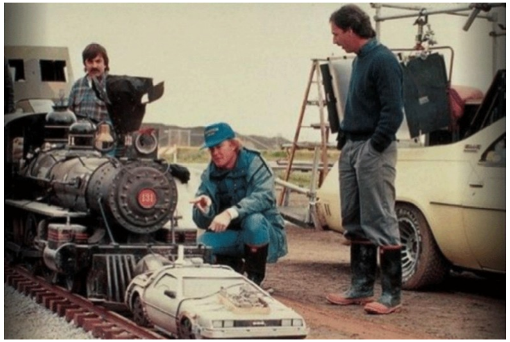
- The relation under a projection is an homography

$$\overline{x_2} = \underbrace{\begin{bmatrix} h_{11} & h_{13} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}}_{H} \cdot \overline{x_1}$$

Homography

- How many matches we need?
  - 9 unknowns? $\longrightarrow$ no, we will never be able to solve scale
  - 8 unknowns $\longrightarrow$ yes!
- Each match $\longrightarrow$ 2 equations
- We need (at least) 4 correspondencies!

$$h = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}$$



Homography

UNIVERSITÀ DI PARMA

- We can solve using:

$$\begin{cases} x_2 = h_{11}x_1 + h_{12}y_1 + h_{13} \\ y_2 = h_{21}x_1 + h_{22}y_1 + h_{23} \\ z_2 = h_{31}x_1 + h_{32}y_1 + h_{33} \end{cases}$$

- For Euclidean coordinates

$$\begin{cases} x_2' = \dfrac{x_2}{z_2} = \dfrac{h_{11}x_1 + h_{12}y_1 + h_{13}}{h_{31}x_1 + h_{32}y_1 + h_{33}} \\ y_2' = \dfrac{y_2}{z_2} = \dfrac{h_{21}x_1 + h_{22}y_1 + h_{23}}{h_{31}x_1 + h_{32}y_1 + h_{33}} \end{cases}$$

Homography

- 2nd step

$$\begin{cases} x_2'(h_{31}x_1 + h_{32}y_1 + h_{33}) = h_{11}x_1 + h_{12}y_1 + h_{13} \\ y_2'(h_{31}x_1 + h_{32}y_1 + h_{33}) = h_{21}x_1 + h_{22}y_1 + h_{23} \end{cases}$$

- Ordering

$$\begin{cases} -h_{11}x_1 - h_{12}y_1 - h_{13} + h_{31}x_1x_2' + h_{32}y_1x_2' + h_{33}x_2' = 0 \\ -h_{21}x_1 - h_{22}y_1 - h_{23} + h_{31}x_1y_2' + h_{32}y_1y_2' + h_{33}y_2' = 0 \end{cases}$$

Homography

UNIVERSITÀ DI PARMA

- This for a single match

$$\begin{cases} -h_{11}x_1 - h_{12}y_1 - h_{13} + h_{31}x_1{x_2}' + h_{32}y_1{x_2}' + h_{33}{x_2}' = 0 \\ -h_{21}x_1 - h_{22}y_1 - h_{23} + h_{31}x_1{y_2}' + h_{32}y_1{y_2}' + h_{33}{y_2}' = 0 \end{cases}$$

- For all

$$A * h = 0$$

Homography

UNIVERSITÀ DI PARMA

- This for a single match

$$\begin{cases} -h_{11}x_1 - h_{12}y_1 - h_{13} + h_{31}x_1{x_2}' + h_{32}y_1{x_2}' + h_{33}{x_2}' = 0 \\ -h_{21}x_1 - h_{22}y_1 - h_{23} + h_{31}x_1{y_2}' + h_{32}y_1{y_2}' + h_{33}{y_2}' = 0 \end{cases}$$

- For all

$$A = \begin{bmatrix} -{x_1}^{(1)} & -{y_1}^{(1)} & -1 & 0 & 0 & 0 & x_1{x_2}'^{(1)} & y_1{x_2}'^{(1)} & {x_2}'^{(1)} \\ 0 & 0 & 0 & -{x_1}^{(1)} & -{y_1}^{(1)} & -1 & x_1{y_2}'^{(1)} & y_1{y_2}'^{(1)} & {y_2}'^{(1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -{x_1}^{(n)} & -{y_1}^{(n)} & -1 & 0 & 0 & 0 & x_1{x_2}'^{(n)} & y_1{x_2}'^{(n)} & {x_2}'^{(n)} \\ 0 & 0 & 0 & -{x_1}^{(n)} & -{y_1}^{(n)} & -1 & x_1{y_2}'^{(n)} & y_1{y_2}'^{(n)} & {y_2}'^{(n)} \end{bmatrix}$$

Homography

- Again we can use SVD decomposition to find best fit

$$A = UDV^T$$

- Last column of V is the solution

SVD

UNIVERSITÀ DI PARMA

- Both OpenCV and Eigen provide a SVD solver:

  cv::SVD::compute(const Mat & A, Mat & D, Mat & U, Mat & Vt)

- Not enough we need to normalize result

- Main steps
  - Acquire multiple images
  - Get transformation from first image to following one
  - Reproject that image onto the previous one
  - Blend them in the result
  - Until we have images $\longrightarrow$ repeat

Reprojection

- We now have H
- Simply use

$$\overline{x_2} = H \cdot \overline{x_1}$$

- Basically we fill plane of first image adding other pixels

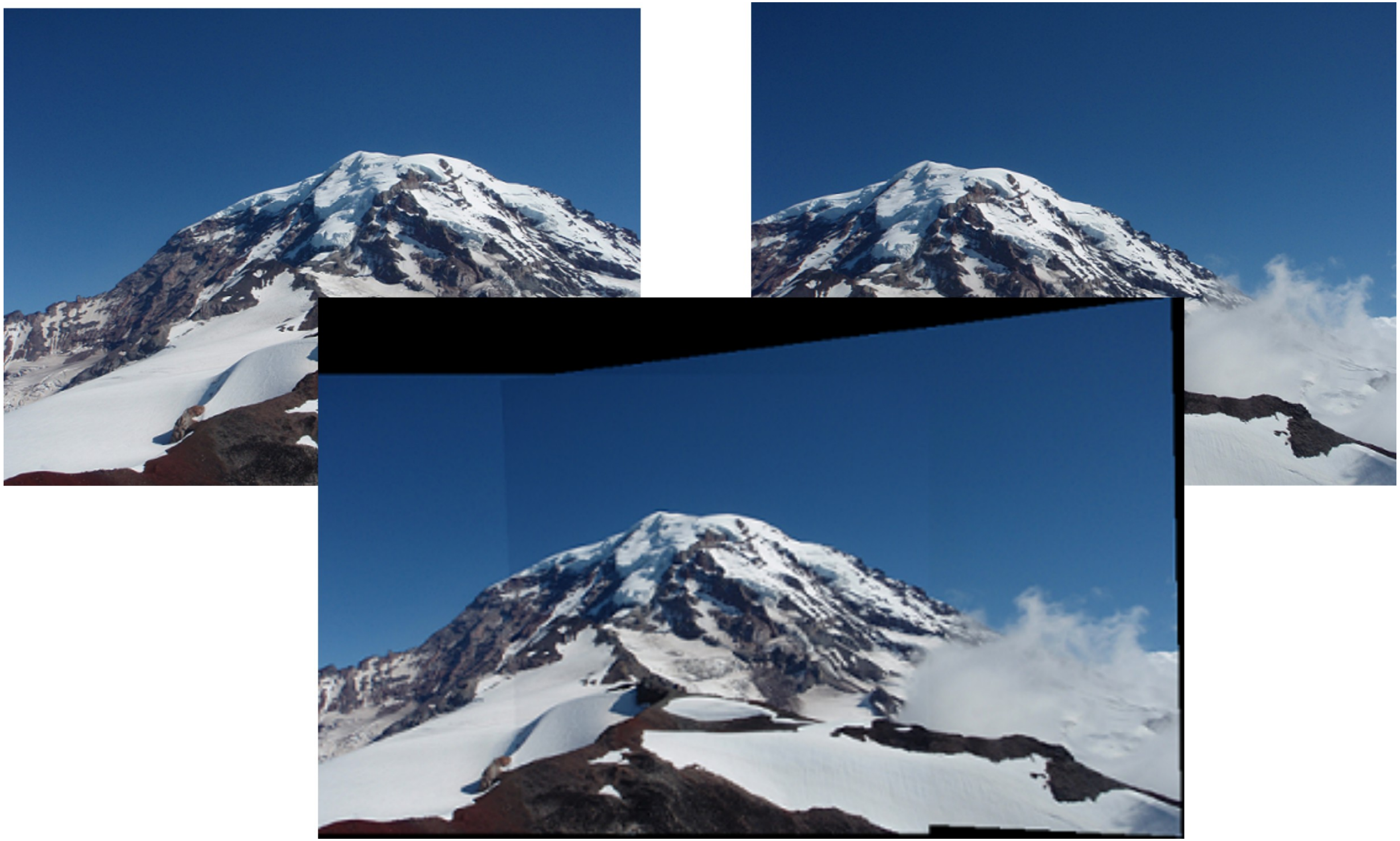
Image Stitching

- Main steps
  - Acquire multiple images
  - Get transformation from first image to following one
  - Reproject that image onto the previous one
  - Blend them in the result
  - Until we have images $\longrightarrow$ repeat

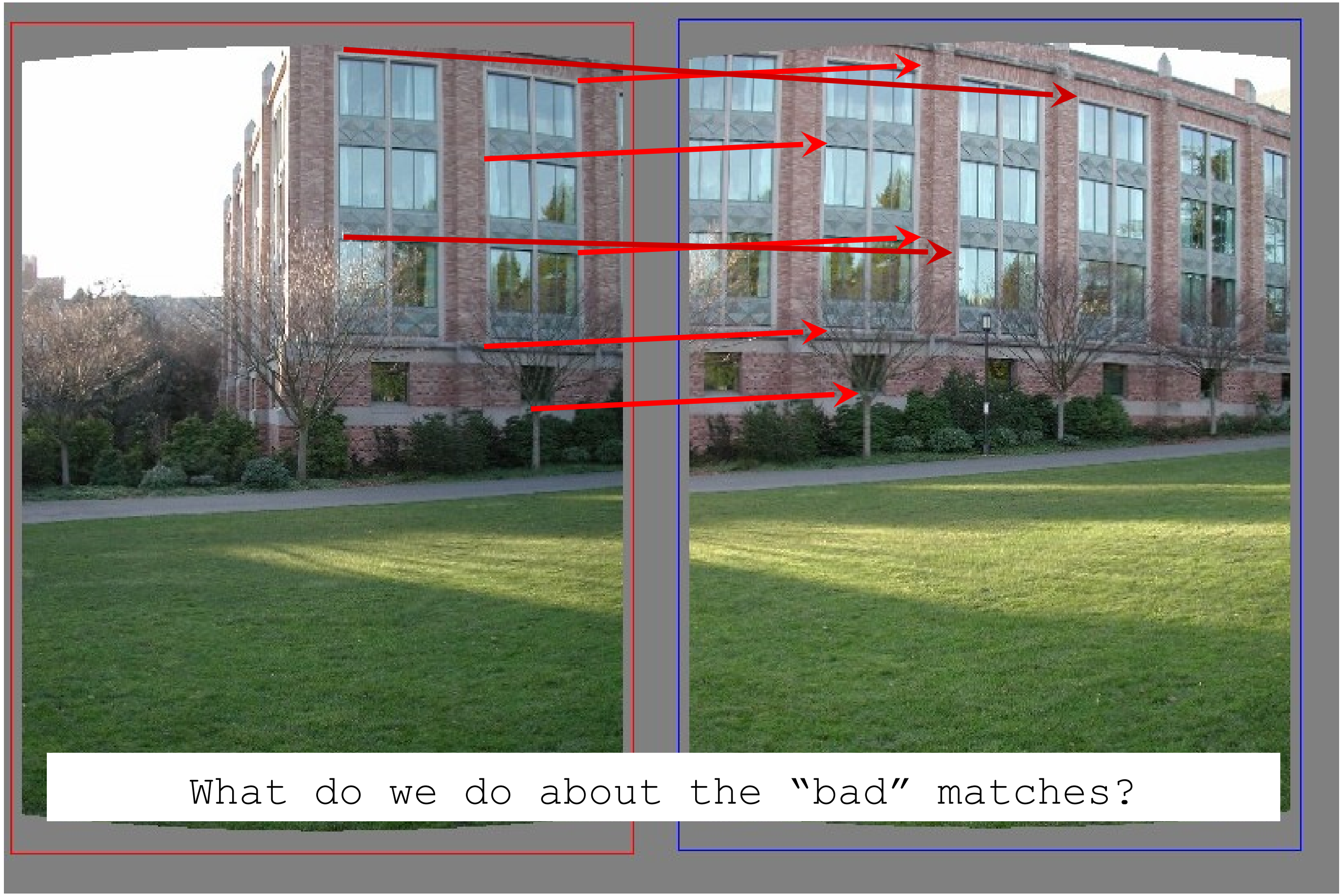
Image blending

- Consider bilinear interpolation
  - Previous equation gives floating point results
- When we have multiple points from different images use an average

Image stitching

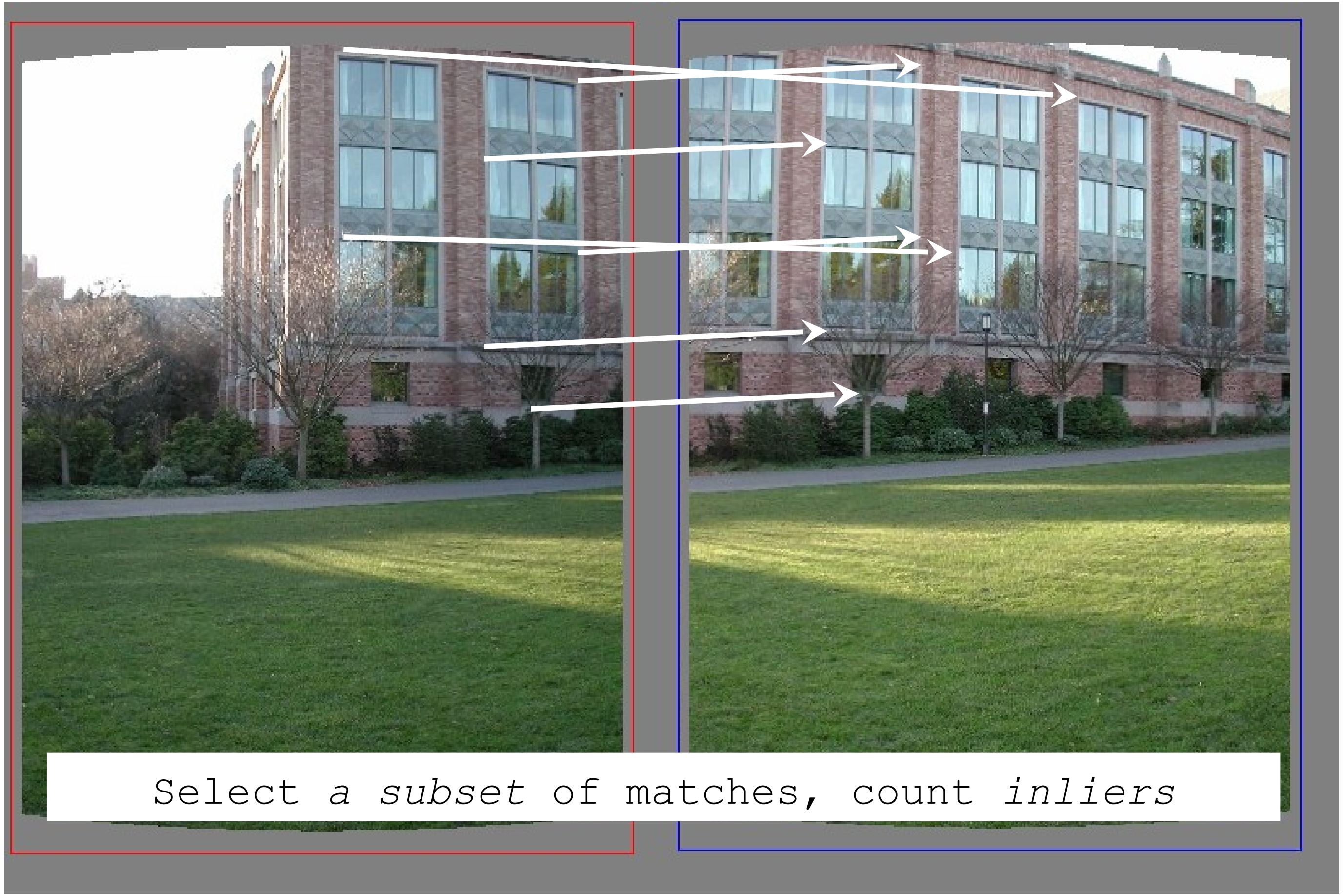


Issues

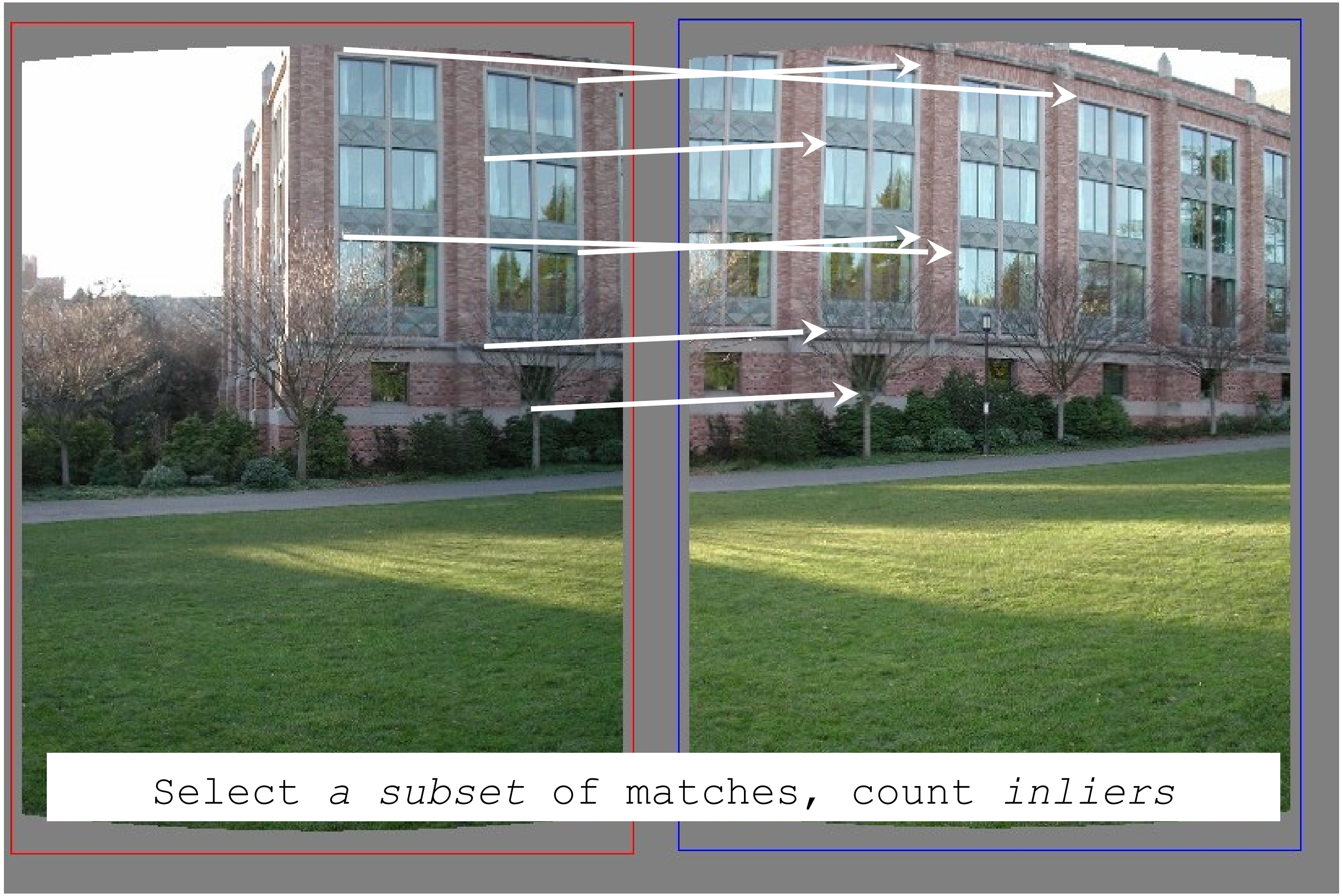


Issues

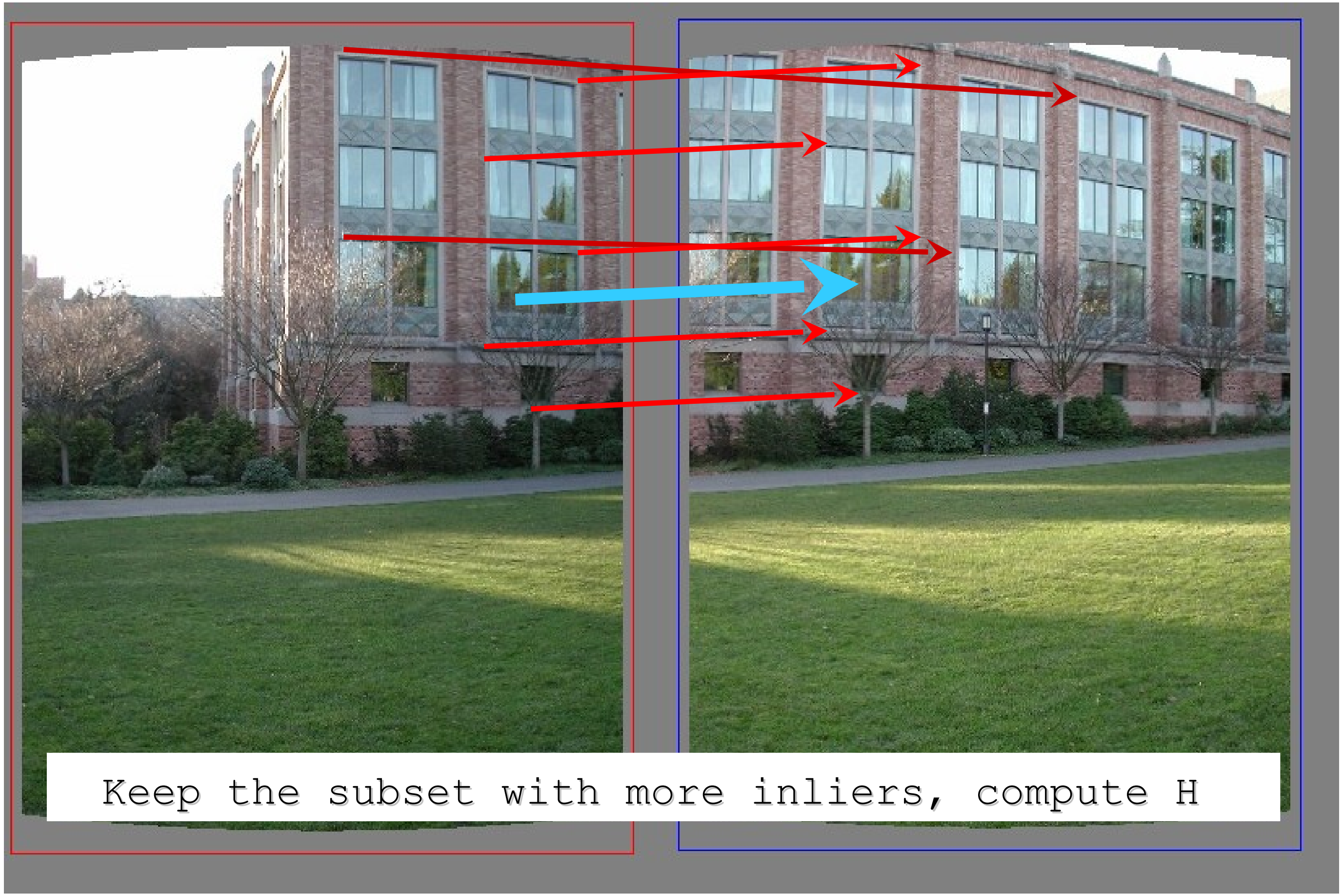
UNIVERSITÀ DI PARMA



Issues



Issues



RANSAC

UNIVERSITÀ DI PARMA

- Randomly select 4 matches
- Compute H from those matches
- Count how many inliers are obtained assuming

  $||p_2', H\, p_i|| < \varepsilon$
- Repeat N times
- At the end recompute H using all inliers of the best match!

UNIVERSITÀ DI PARMA

Feature Matching

Question time!