MIT CSAIL

6.869: Advances in Computer Vision

MIT COMPUTER VISION

Image Features, Homographies, RANSAC and Panoramas

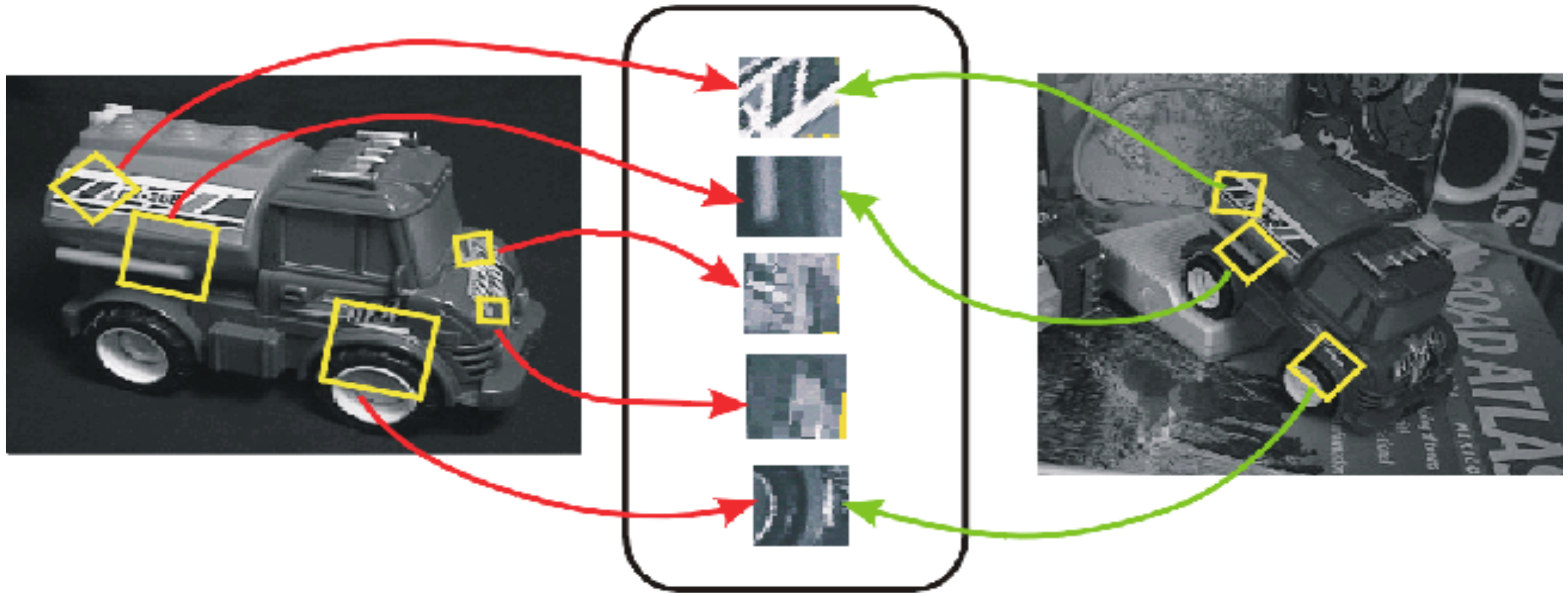
Lecture 13

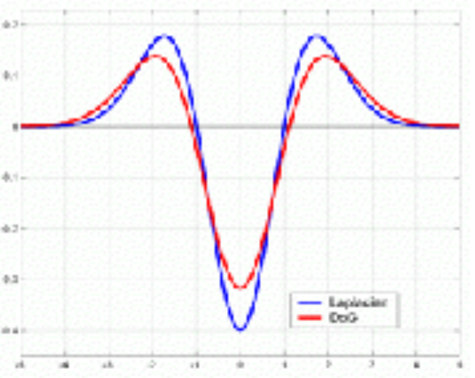
Tali Dekel

Scale and Rotation Invariant Detection: Recap

- **Given:** two images of the same scene with a large scale difference and/or rotation between them
- **Goal:** find **the same** interest points **independently** in each image
- **Solution:** search for **maxima** of suitable functions in **scale** and in **space** (over the image).

  » finding a characteristic scale

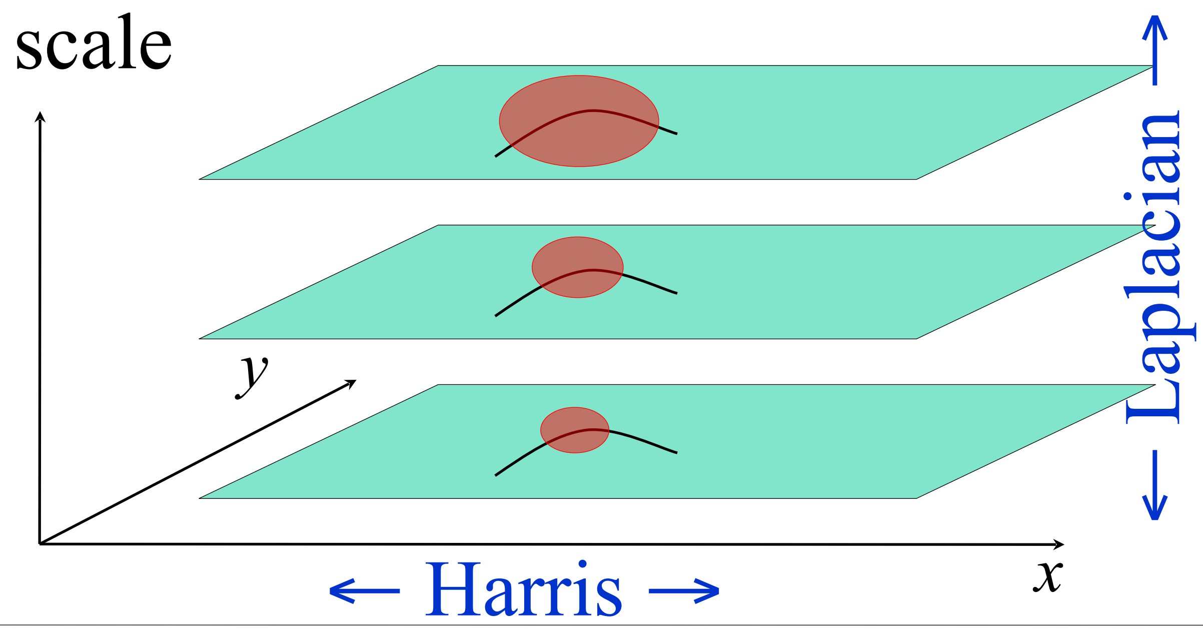




Scale Invariant Detectors

- Harris-Laplacian[1]

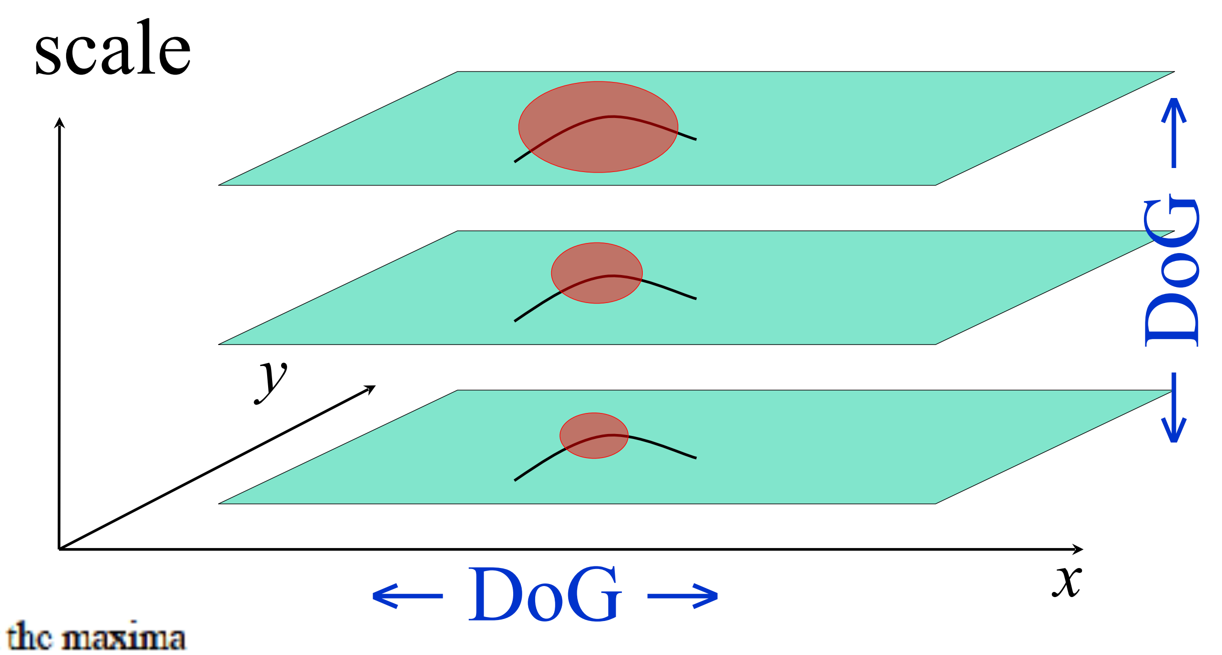
  Find local maximum of:
  - Harris corner detector in space (image coordinates)
  - Laplacian in scale

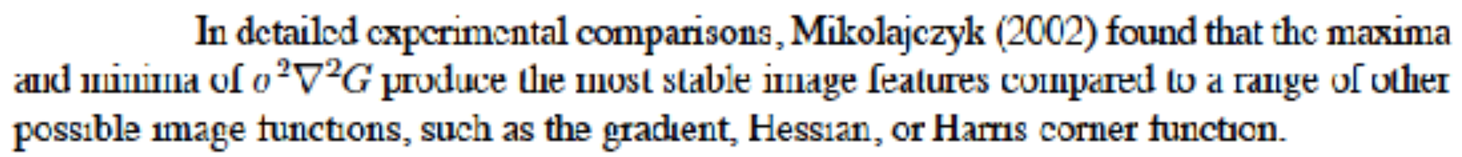


- SIFT (Lowe)[2]

  Find local maximum (minimum) of:

  – Difference of Gaussians in space and scale

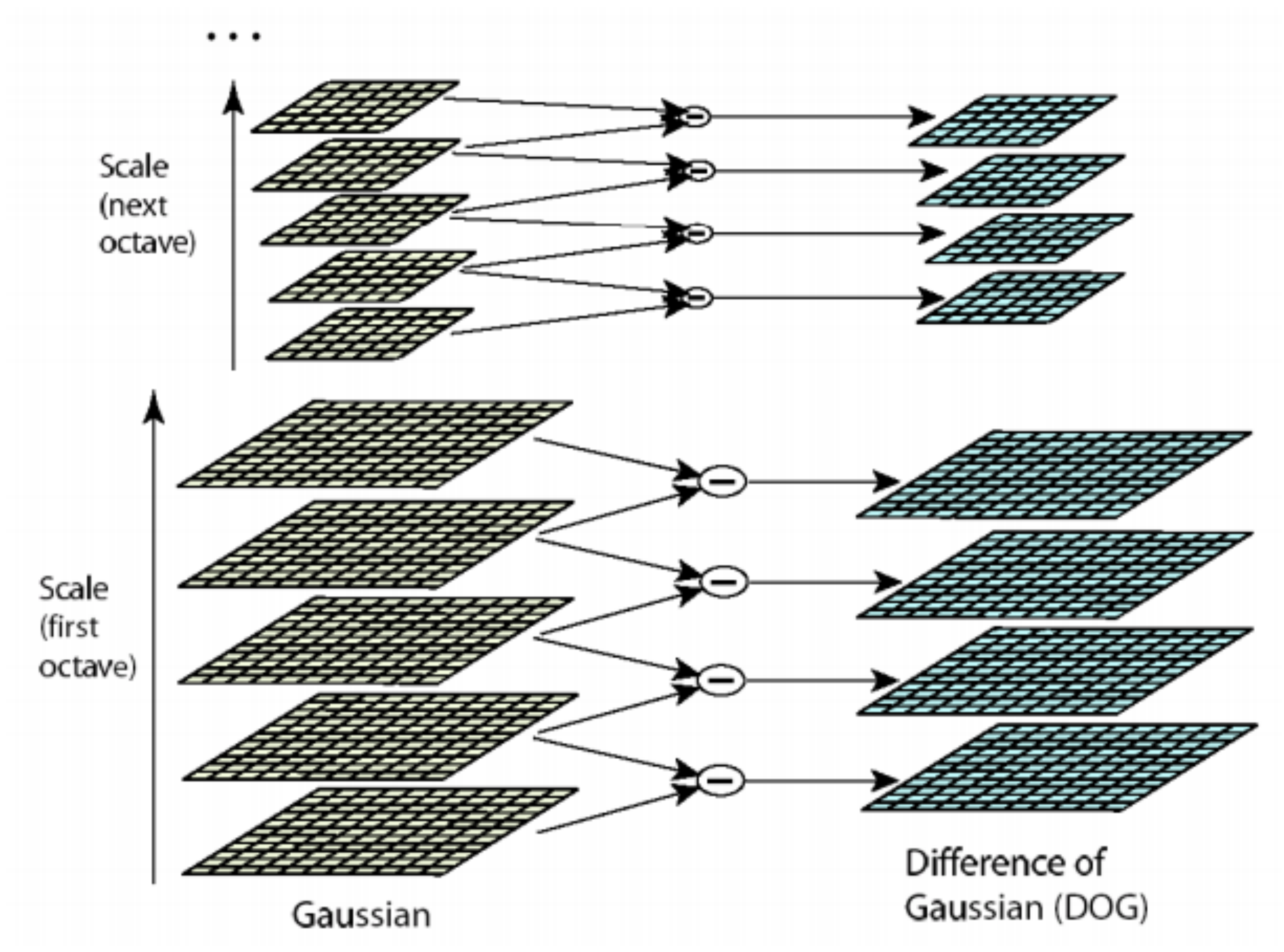



In detailed experimental comparisons, Mikolajczyk (2002) found that the maxima and minima of $\sigma^2\nabla^2 G$ produce the most stable image features compared to a range of other possible image functions, such as the gradient, Hessian, or Harris corner function.

[1] *K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001*

[2] *D.Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". Accepted to IJCV 2004*

DOG Scale Space (Lowe 2004)

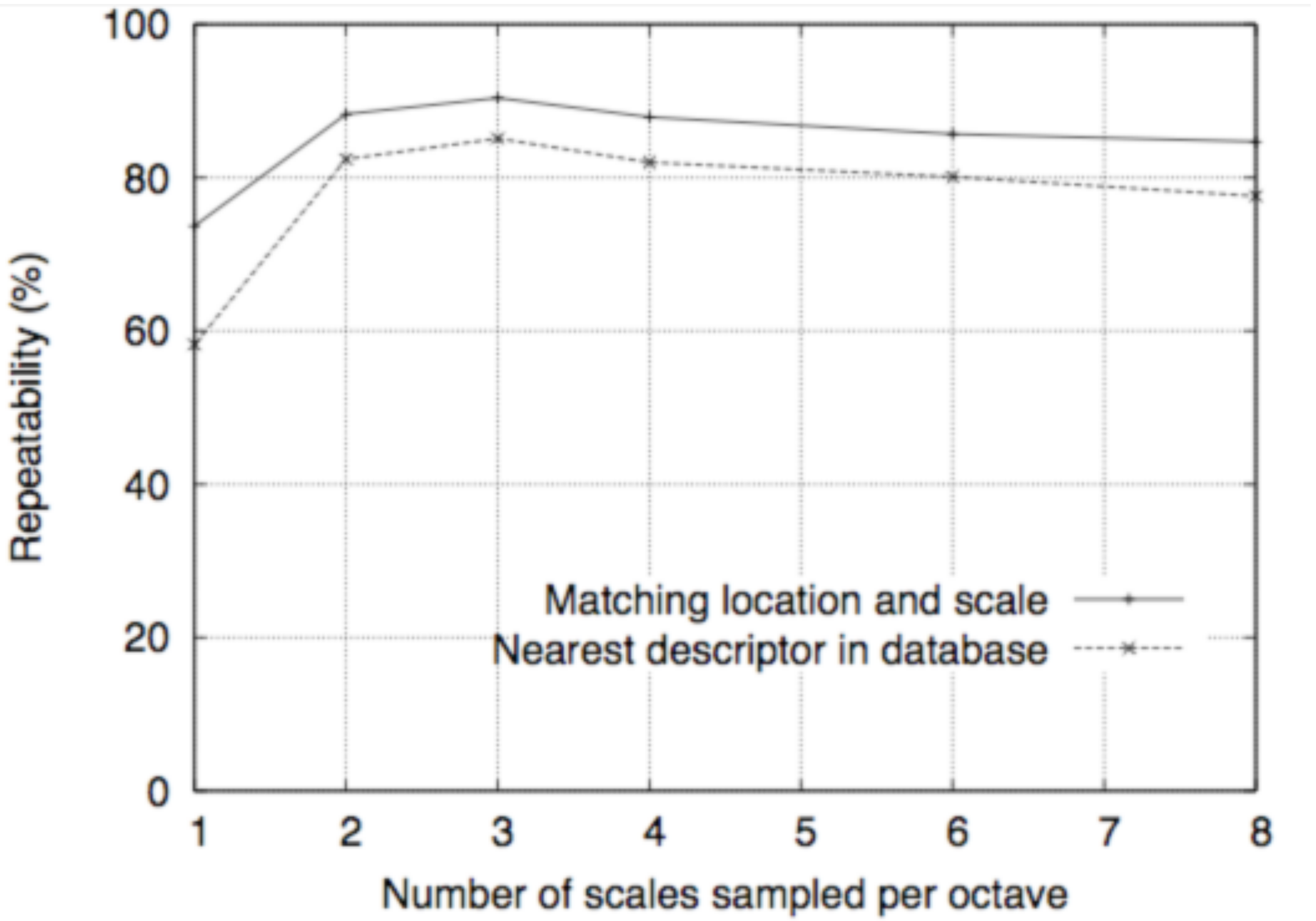


Each octave is doubling of scale – Halve image dimensions

Within octave several scales – Same dimension for all images

Figure 1: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Repeatability vs number of scales sampled per octave



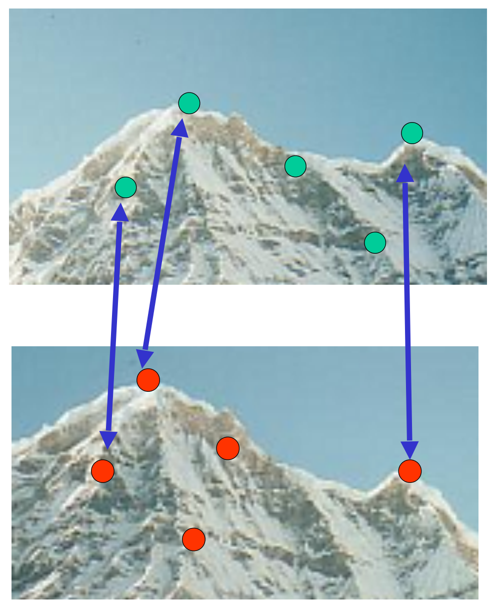
David G. Lowe, "Distinctive image features from scale-invariant keypoints," International Journal of Computer Vision, 60, 2 (2004), pp. 91-110

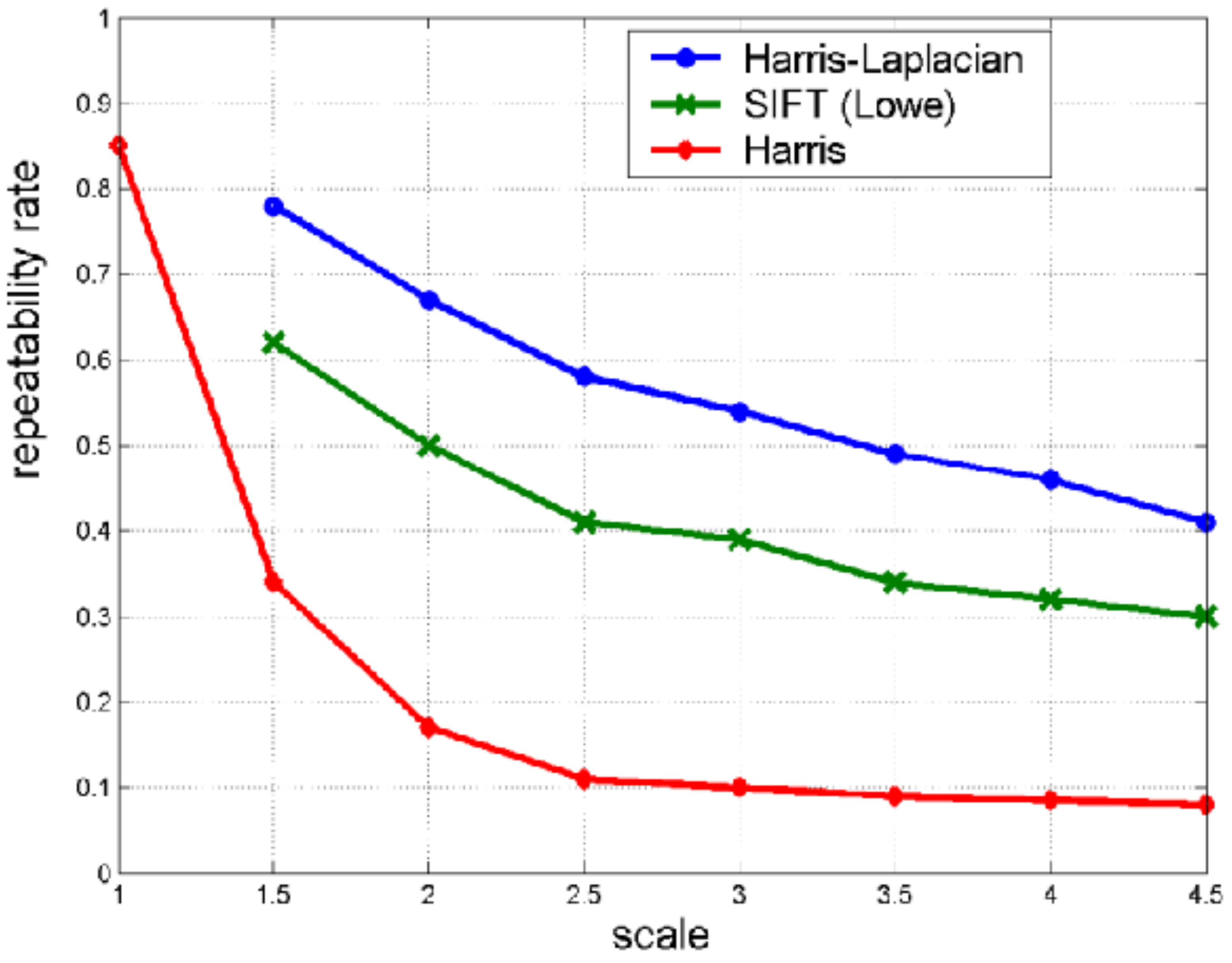
Scale Invariant Detectors

- Experimental evaluation of detectors w.r.t. scale change

Repeatability rate:

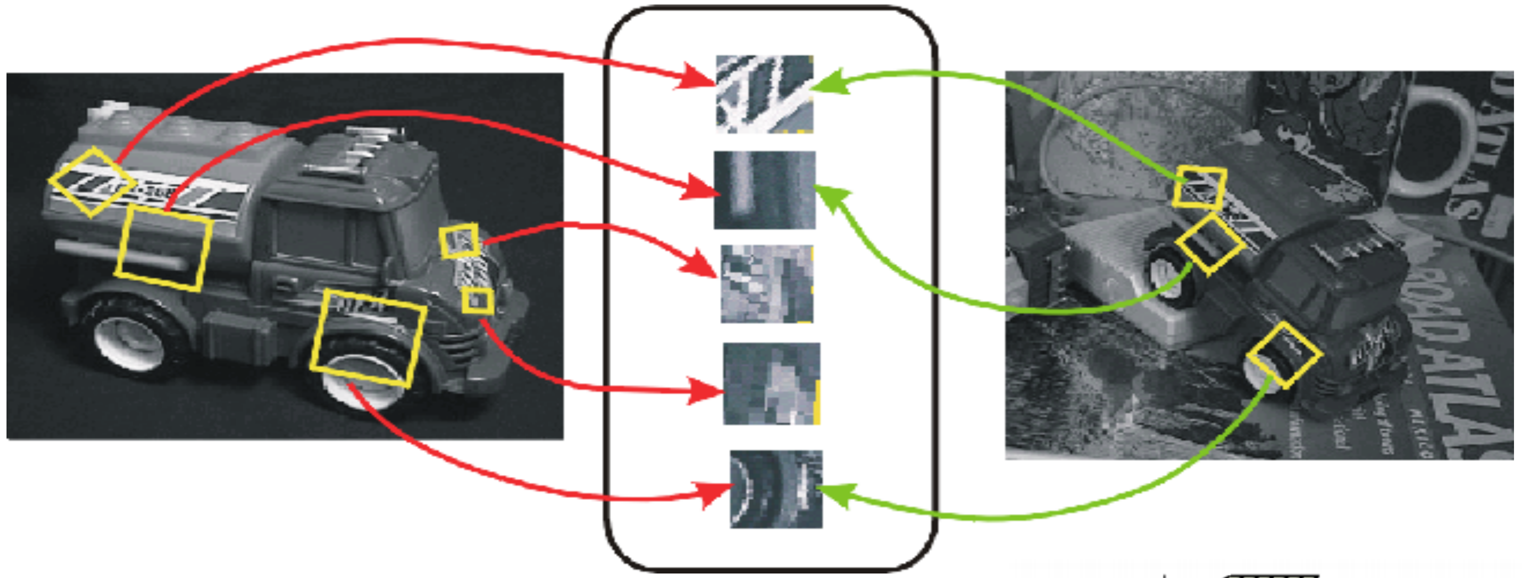
$$\frac{\#\text{ correspondences}}{\#\text{ possible correspondences}}$$



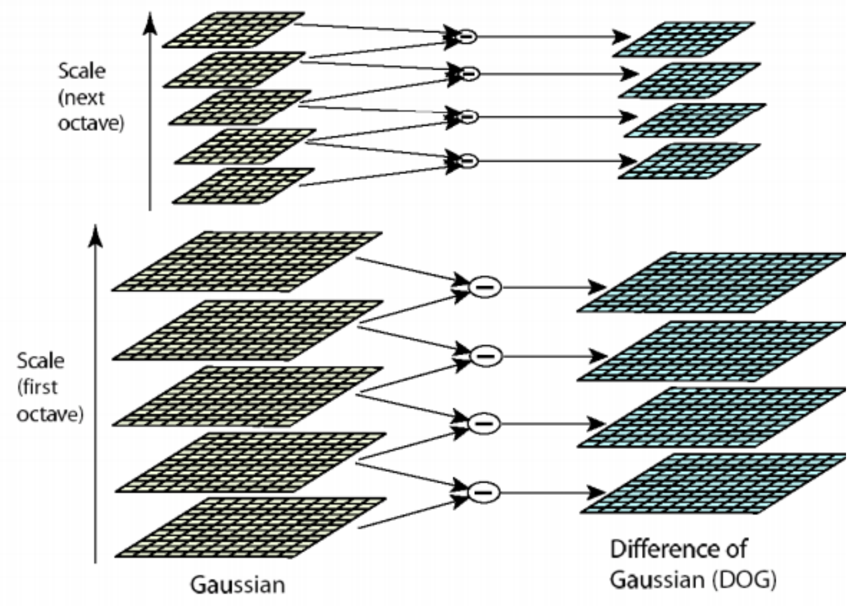


K.Mikolajczyk, C.Schmid. "Indexing Based on Scale Invariant Interest Points". ICCV 2001

SIFT— Orientation Assignment



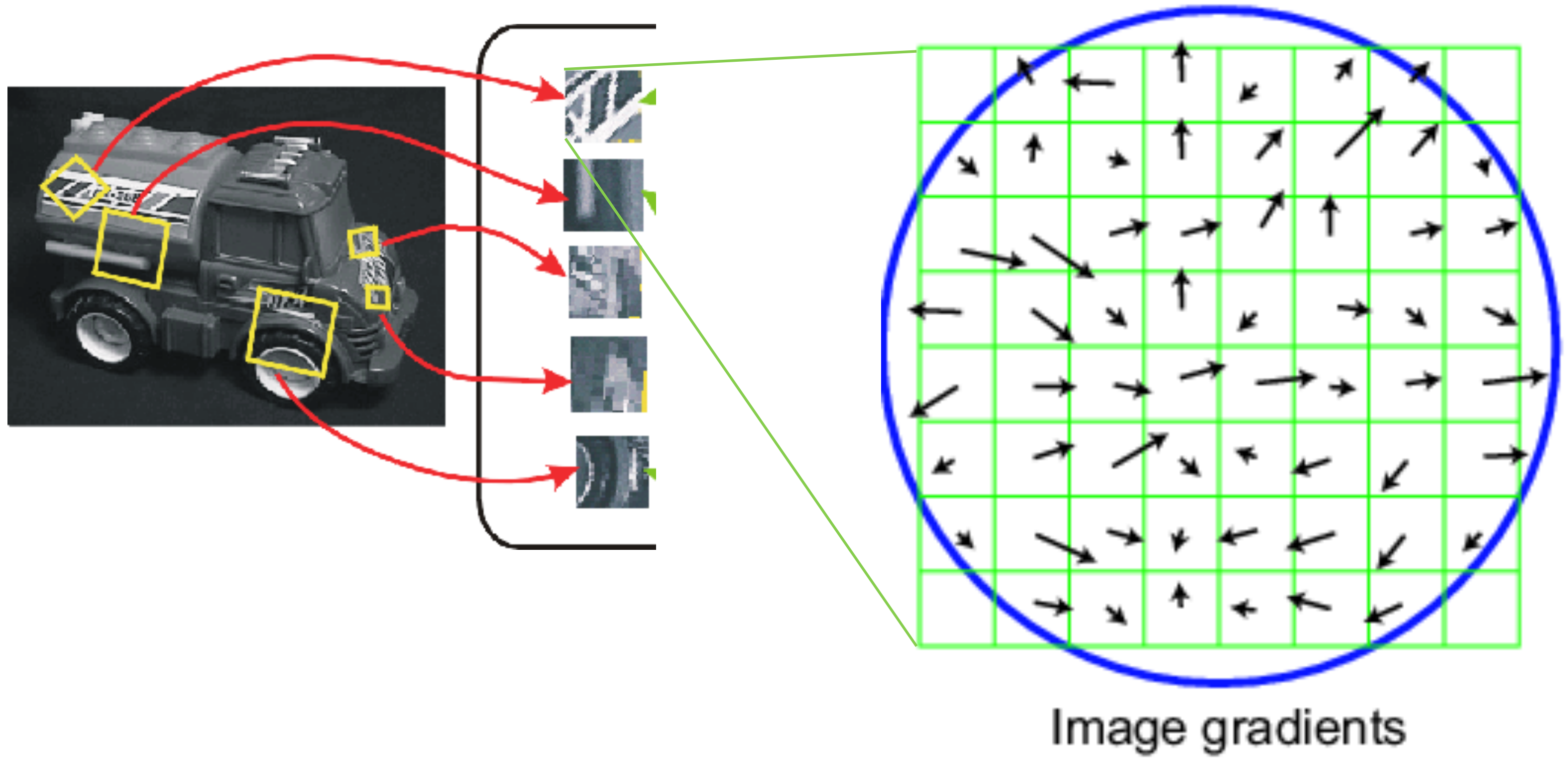
- Use the scale of the key point to grab smoothed image L
- Compute gradient magnitude and orientation:



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

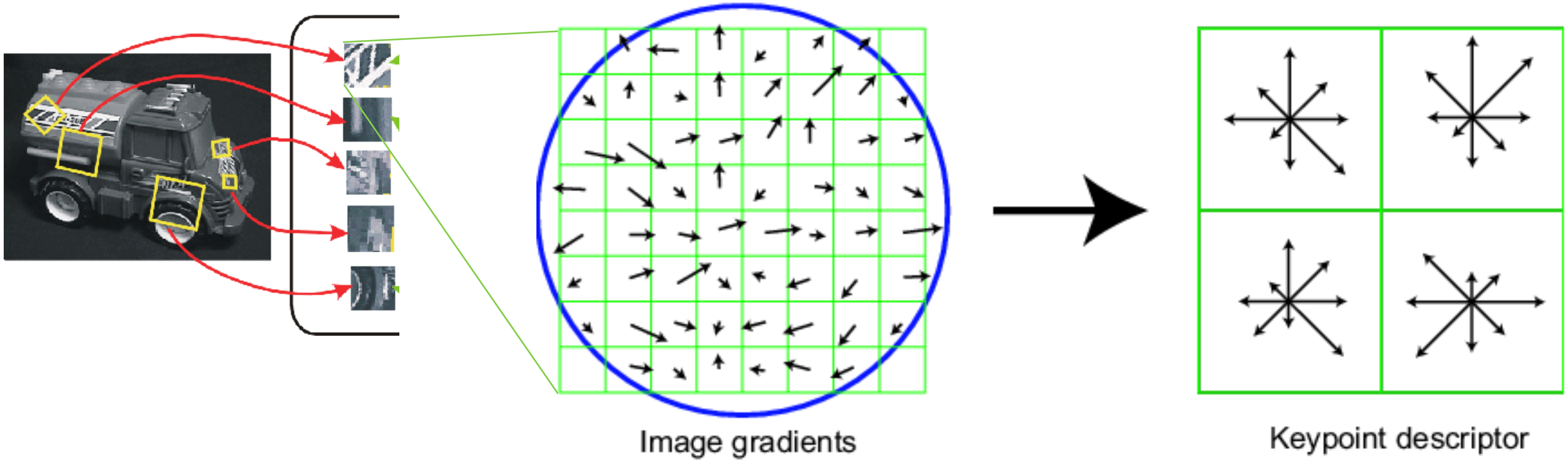
SIFT — Vector Formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian of variance half the window (for smooth falloff)



SIFT — Vector Formation

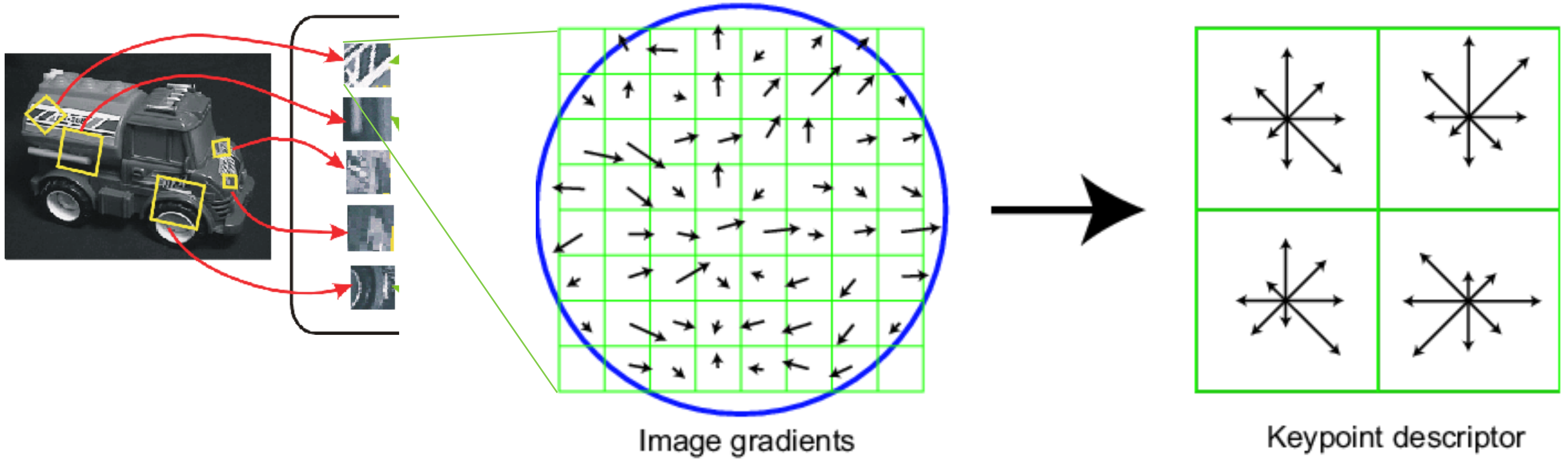
- 4x4 array of gradient orientation histograms
  - not really histogram, weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much.



showing only 2x2 here but is 4x4

Reduce Effect of Illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after normalization, clamp gradients >0.2
  - renormalize



showing only 2x2 here but is 4x4

Tuning and evaluating the SIFT descriptors

Database images were subjected to:

rotation, scaling, affine stretch, brightness and contrast changes, and added noise.

Feature point detectors and descriptors were compared before and after the distortions, and evaluated for:

- Sensitivity to number of histogram orientations and subregions.
- Stability to noise.
- Stability to affine change.
- Feature distinctiveness

Sensitivity to number of histogram orientations and subregions (n)

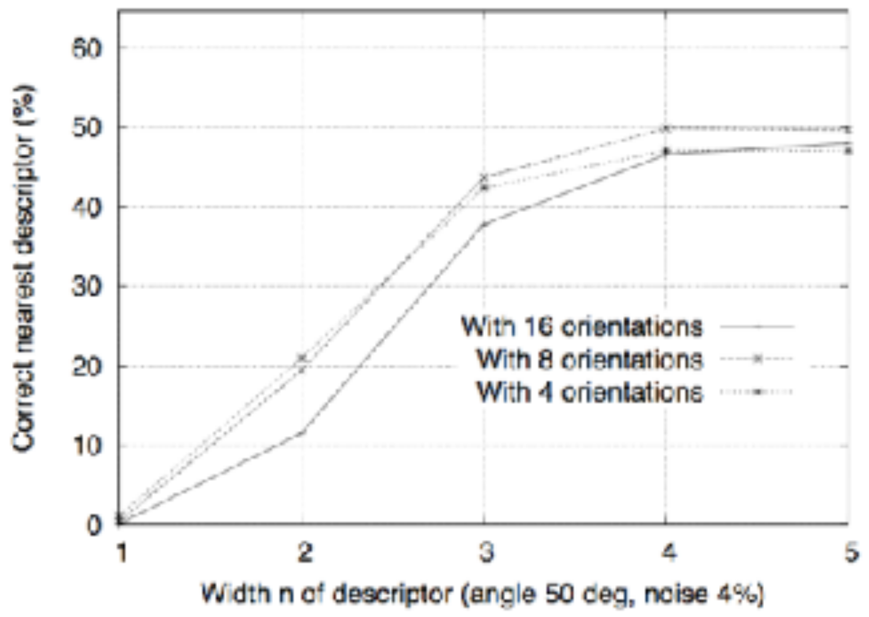
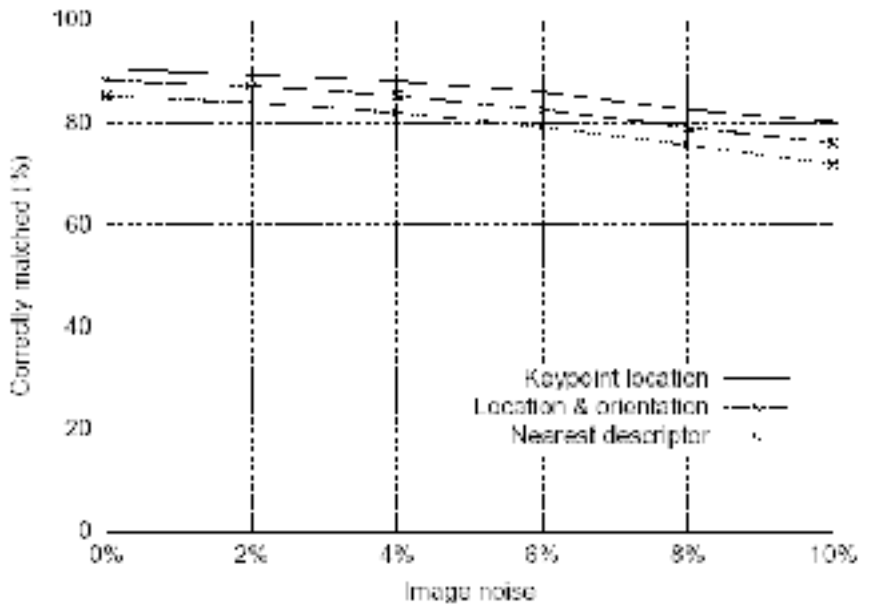


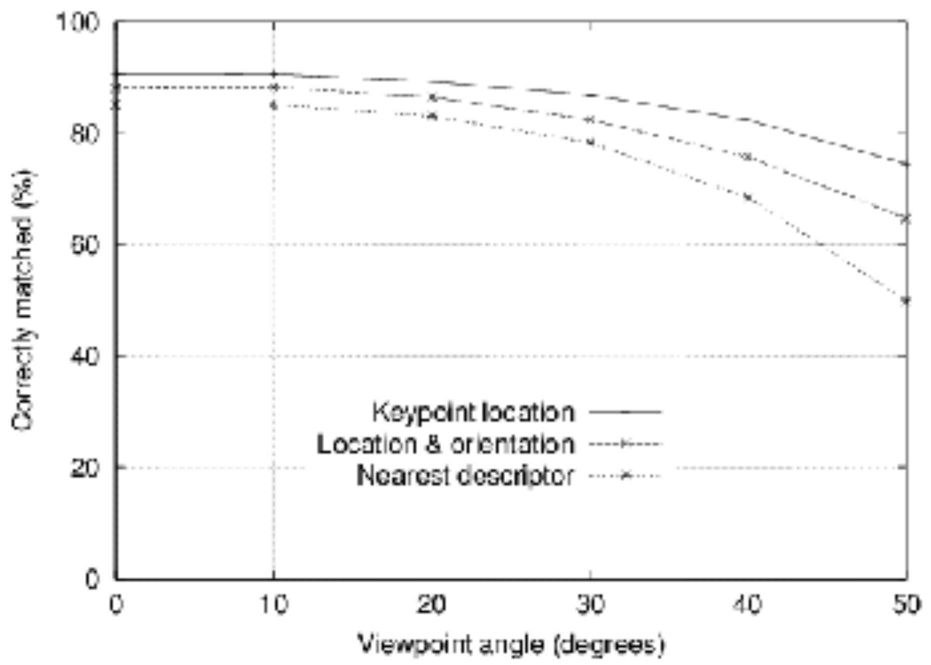
Figure 8: This graph shows the percent of keypoints giving the correct match to a database of 40,000 keypoints as a function of width of the $n \times n$ keypoint descriptor and the number of orientations in each histogram. The graph is computed for images with affine viewpoint change of 50 degrees and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

Feature stability to noise



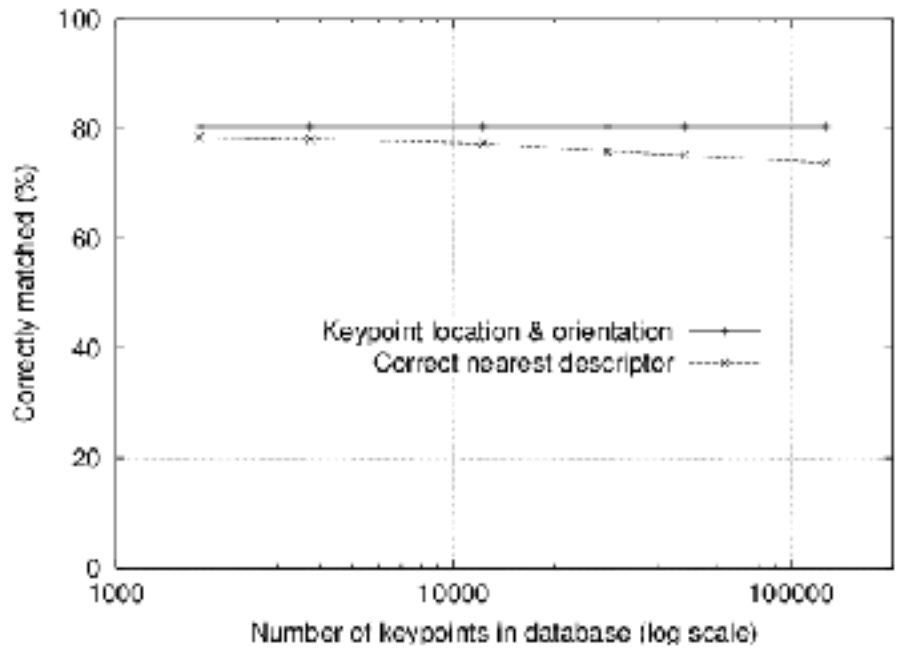
- Match features after random change in image scale & orientation, with differing levels of image noise
- Find nearest neighbor in database of 30,000 features

Feature stability to affine change



- Match features after random change in image scale & orientation, with 2% image noise, and affine distortion
- Find nearest neighbor in database of 30,000 features

Distinctiveness of features

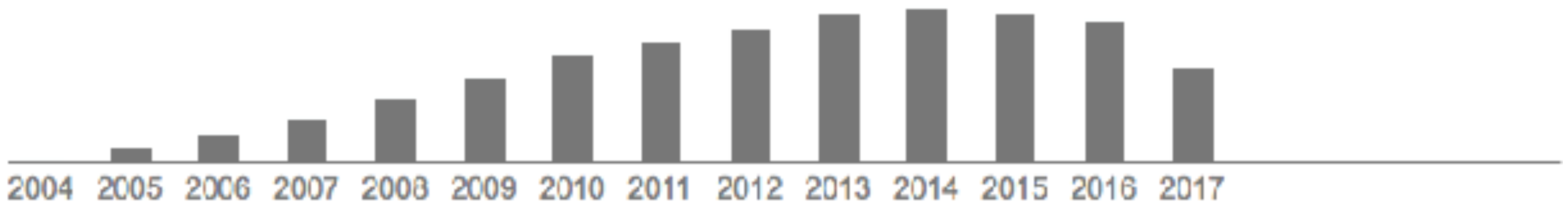


- Vary size of database of features, with 30 degree affine change, 2% image noise
- Measure % correct for single nearest neighbor match

SIFT Impact

Distinctive image features from scale-invariant keypoints

Authors	David G Lowe
Publication date	2004/11/1
Journal	International journal of computer vision
Volume	60
Issue	2
Pages	91-110
Publisher	Springer Netherlands
Description	This paper presents 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. The features are invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from ...
Total citations	Cited by 43944



A good SIFT features tutorial:

http://www.cs.toronto.edu/~jepson/csc2503/tutSIFT04.pdf

By Estrada, Jepson, and Fleet.

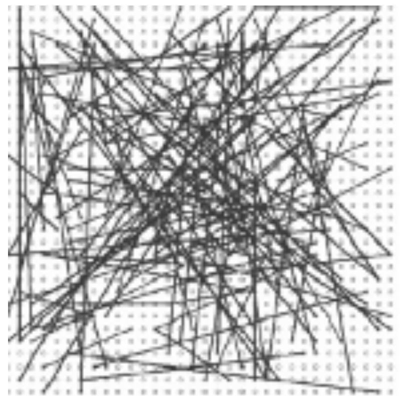
The original SIFT paper:

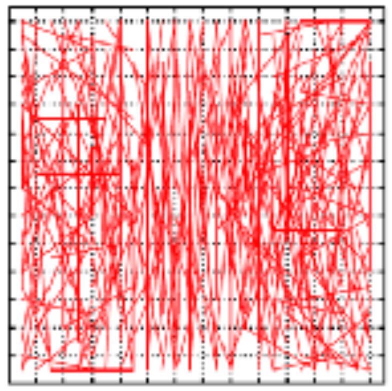
http://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf

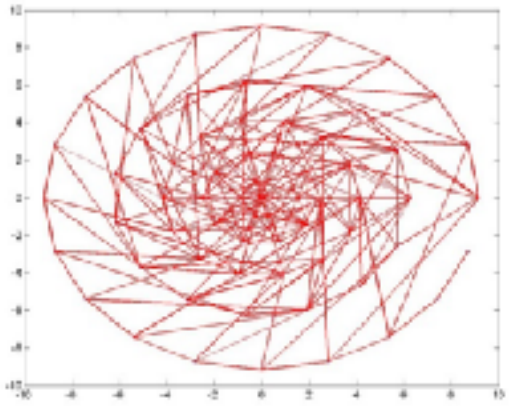
Binary Descriptors

BRIEF, BRISK, ORB, FREAK

- Extremely efficient computation and comparison
- Encode a patch as a binary string using only pairwise intensity comparisons
  - Sampling pattern around each key point
  - Sampling pairs
  - Descriptor is given by a binary string:







$$F = \sum_{0 \le a \le N} 2^a T(P_a)$$

$$T(P_a) = \begin{cases} 1 & \text{if } I(P_a^{r1}) > I(P_a^{r2}) \\ 0 & \text{otherwise} \end{cases}$$

- Matching using Hamming distance: $L = \sum_{0 \le a \le N} XOR(F_a^1, F_a^2)$

Time per keypoint	SIFT	SURF	BRISK	FREAK
Description in [ms]	2.5	1.4	0.031	0.018
Matching time in [ns]	1014	566	36	25

Table 1: Computation time on 800x600 images where approximately 1500 keypoints are detected per image. The computation times correspond to the description and matching of all keypoints.

A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. In IEEE Conference on Computer Vision and Pattern Recognition,

Stitching a pair of image

We have:

- Well-localized features
- Distinctive descriptor

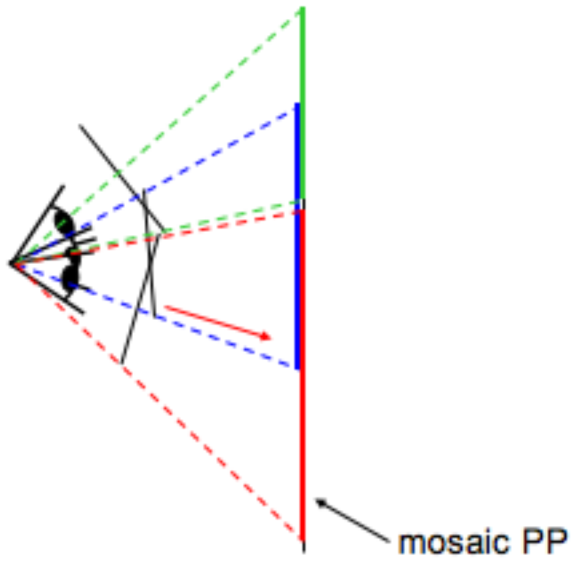
Now we need to:

- Match pairs of feature points in different images
- Robustly compute homographies
  (in the presence of errors/outliers)

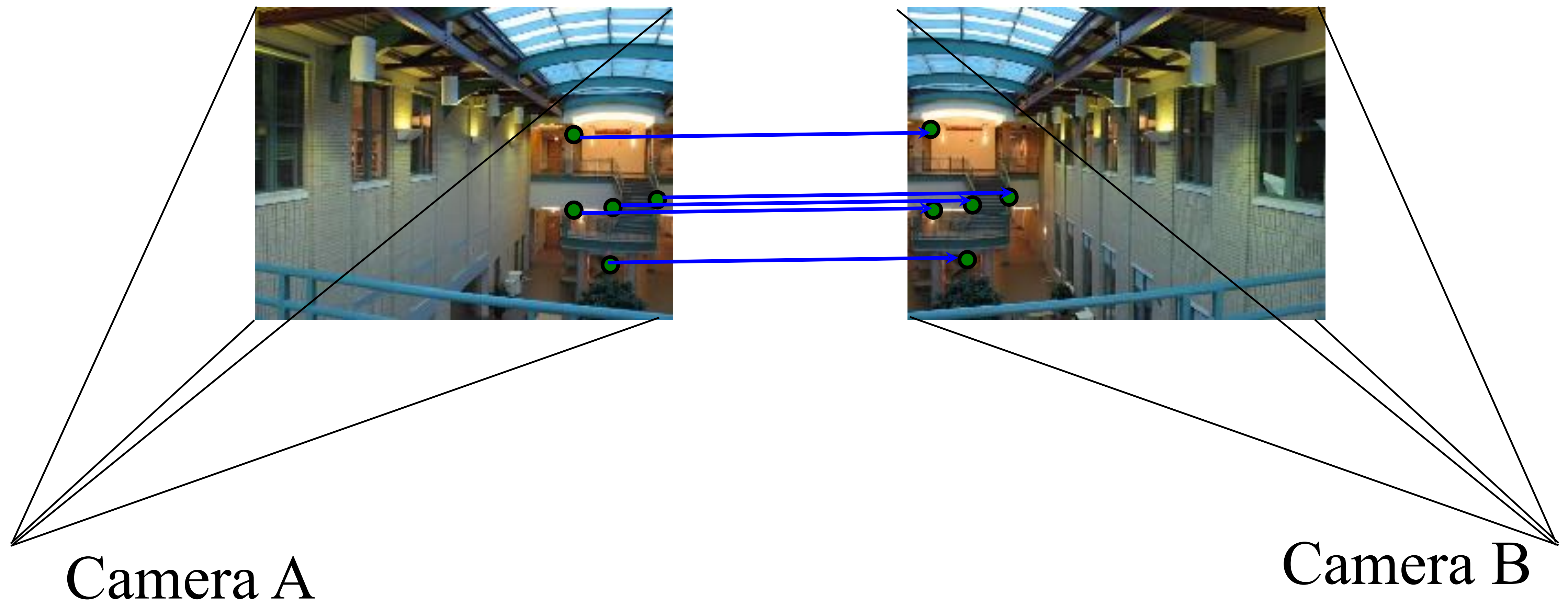
Building a Panorama

A 3D interpretation:

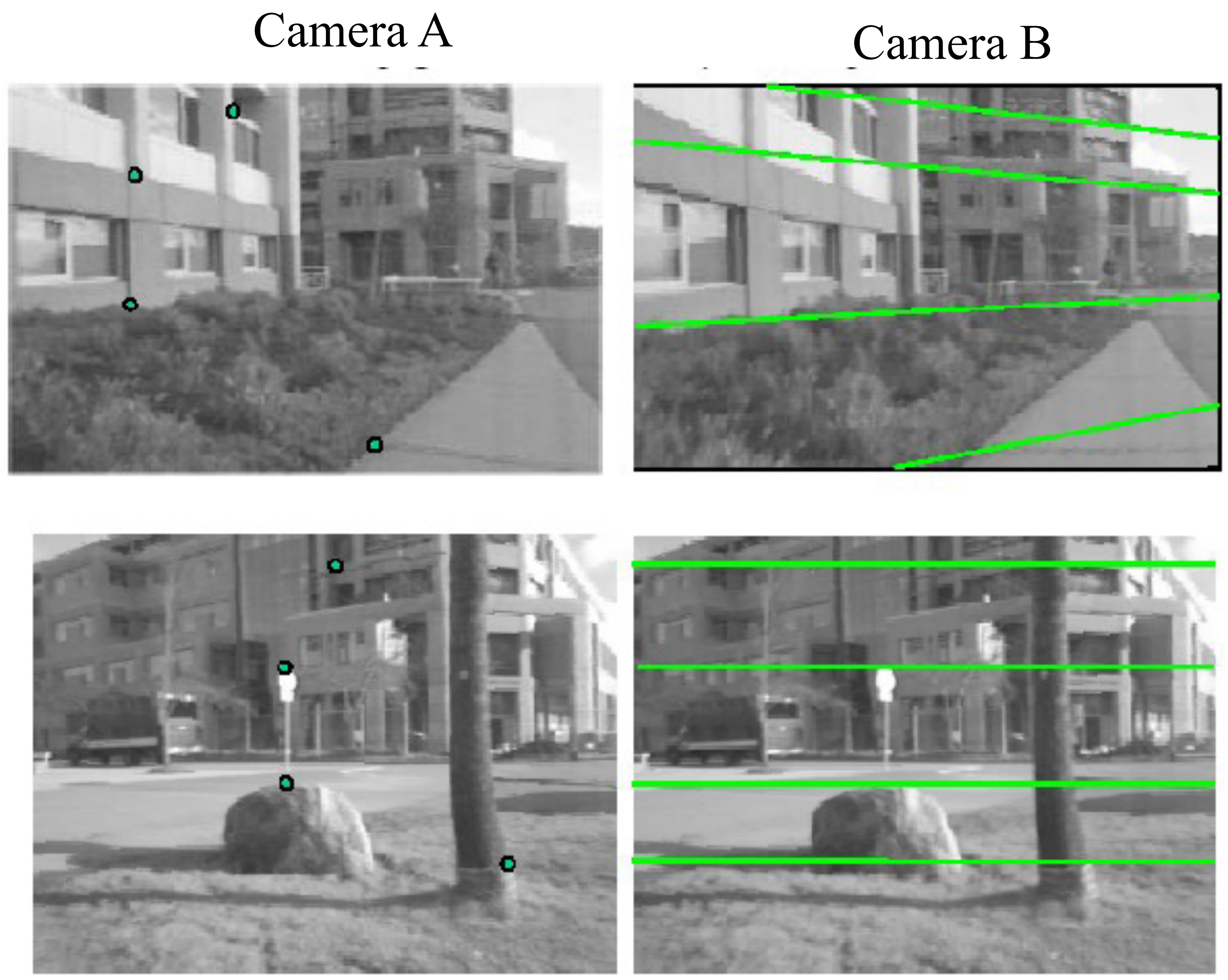
- Build a synthetically wide-angle camera
- Reproject all images onto a common plan
- The mosaic is formed on this plane



Under what conditions can we know where to translate each point of image A to where it would appear in camera B (with calibrated cameras), knowing nothing about image depths?

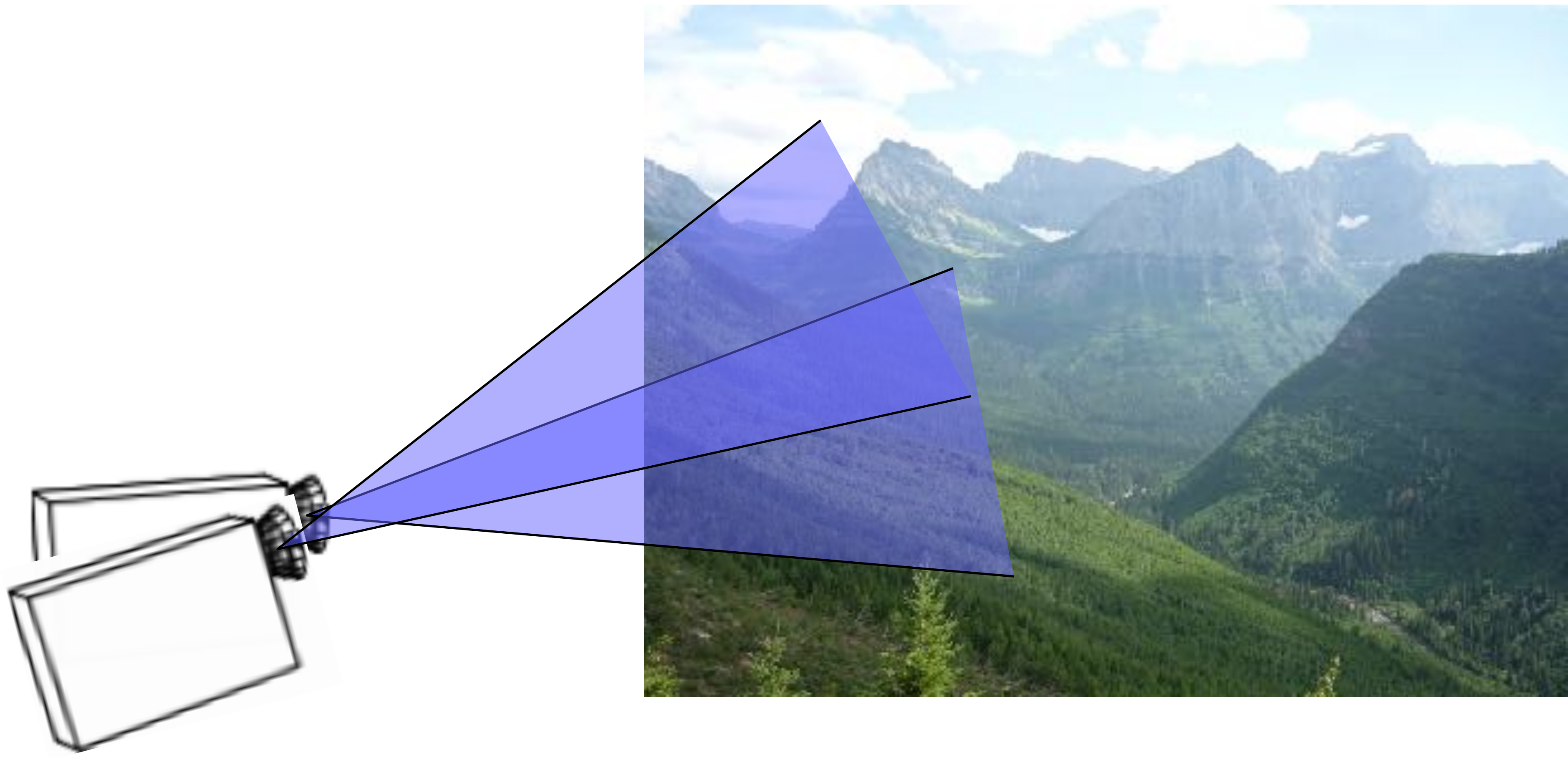


Depth-based ambiguity of position

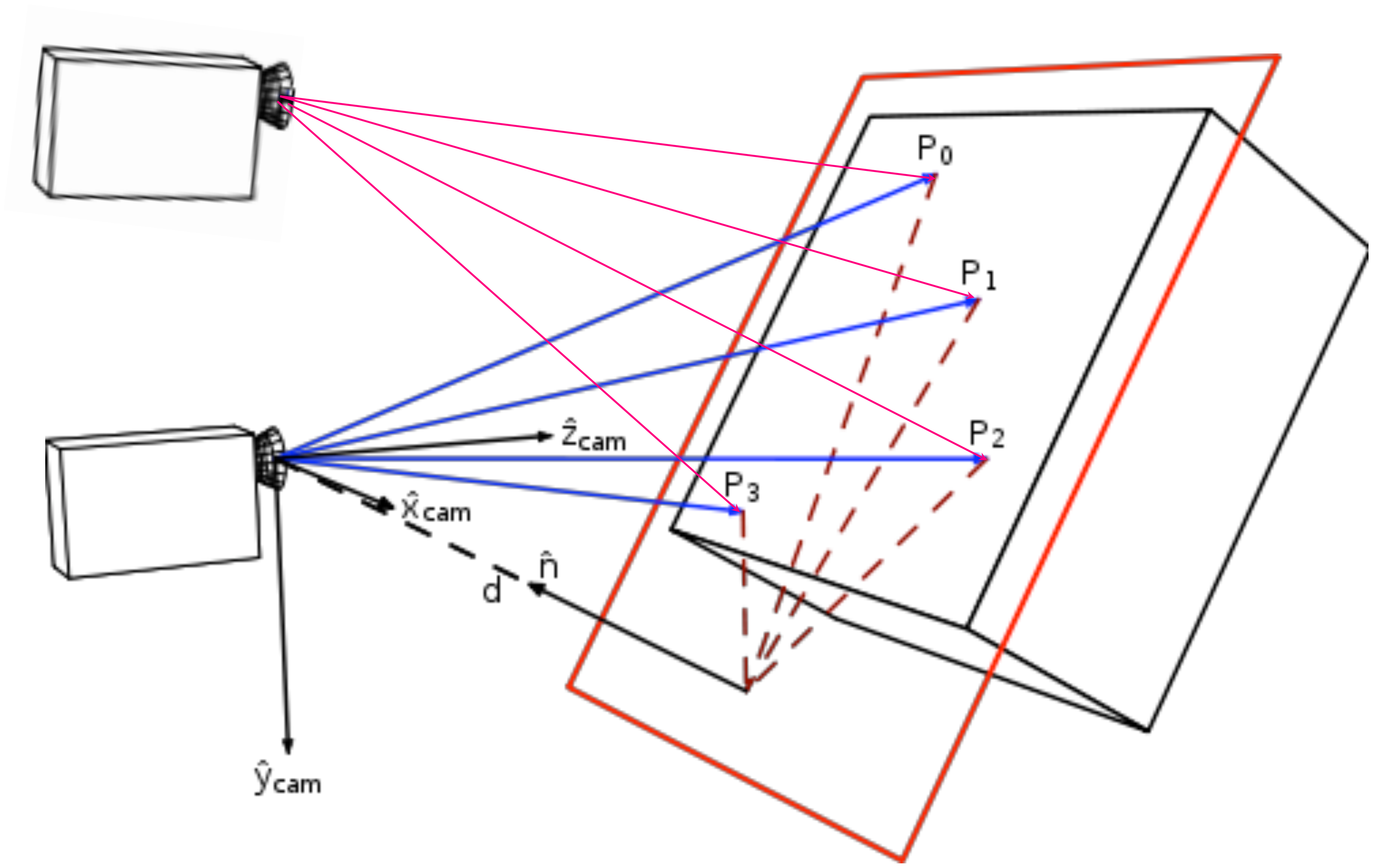


In general, matches are constrained to lie on the epipolar lines, but… that’s it?, there are no more constraints?

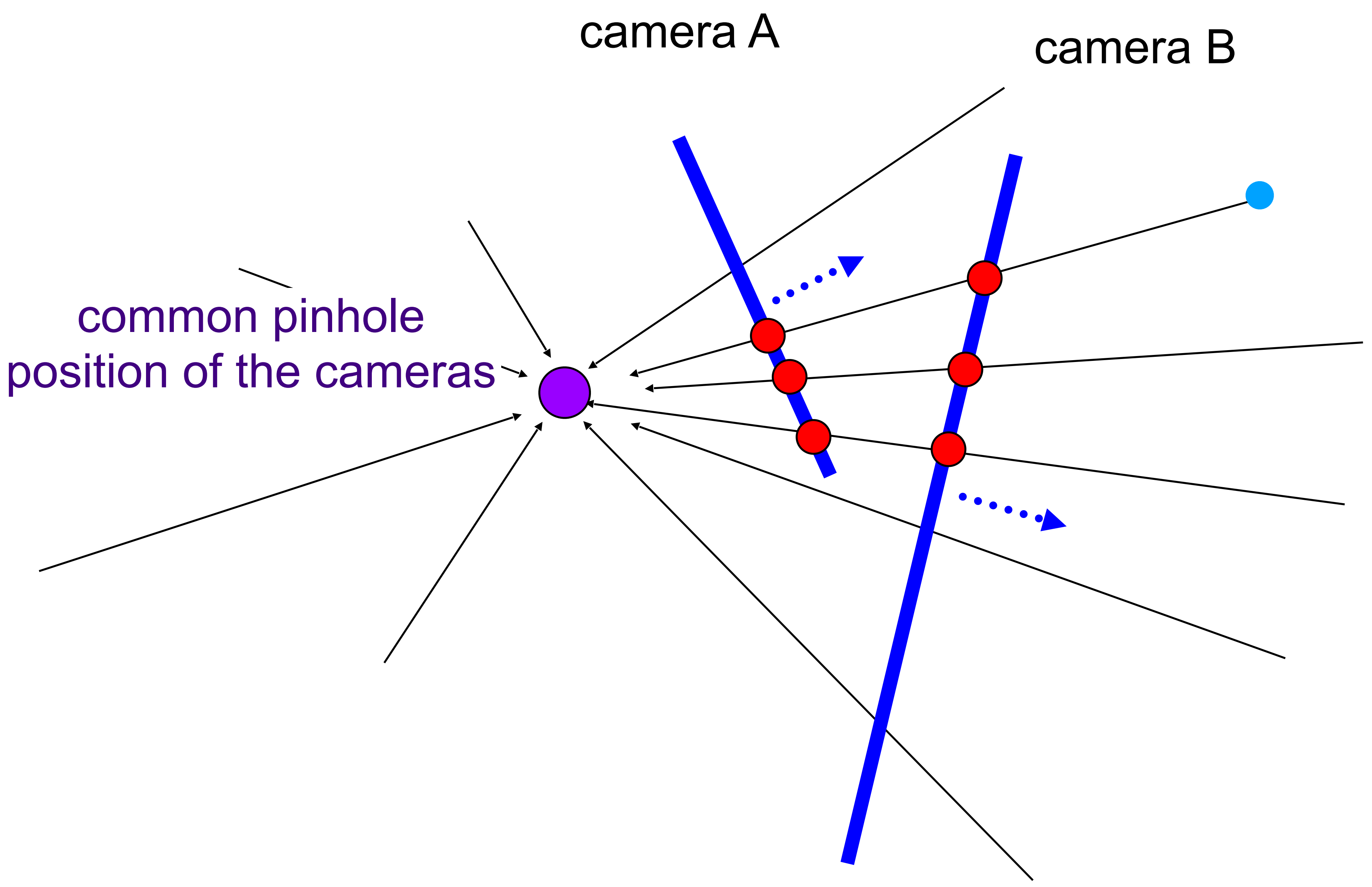
(a) Pure camera rotation



and (b) imaging a planar surface

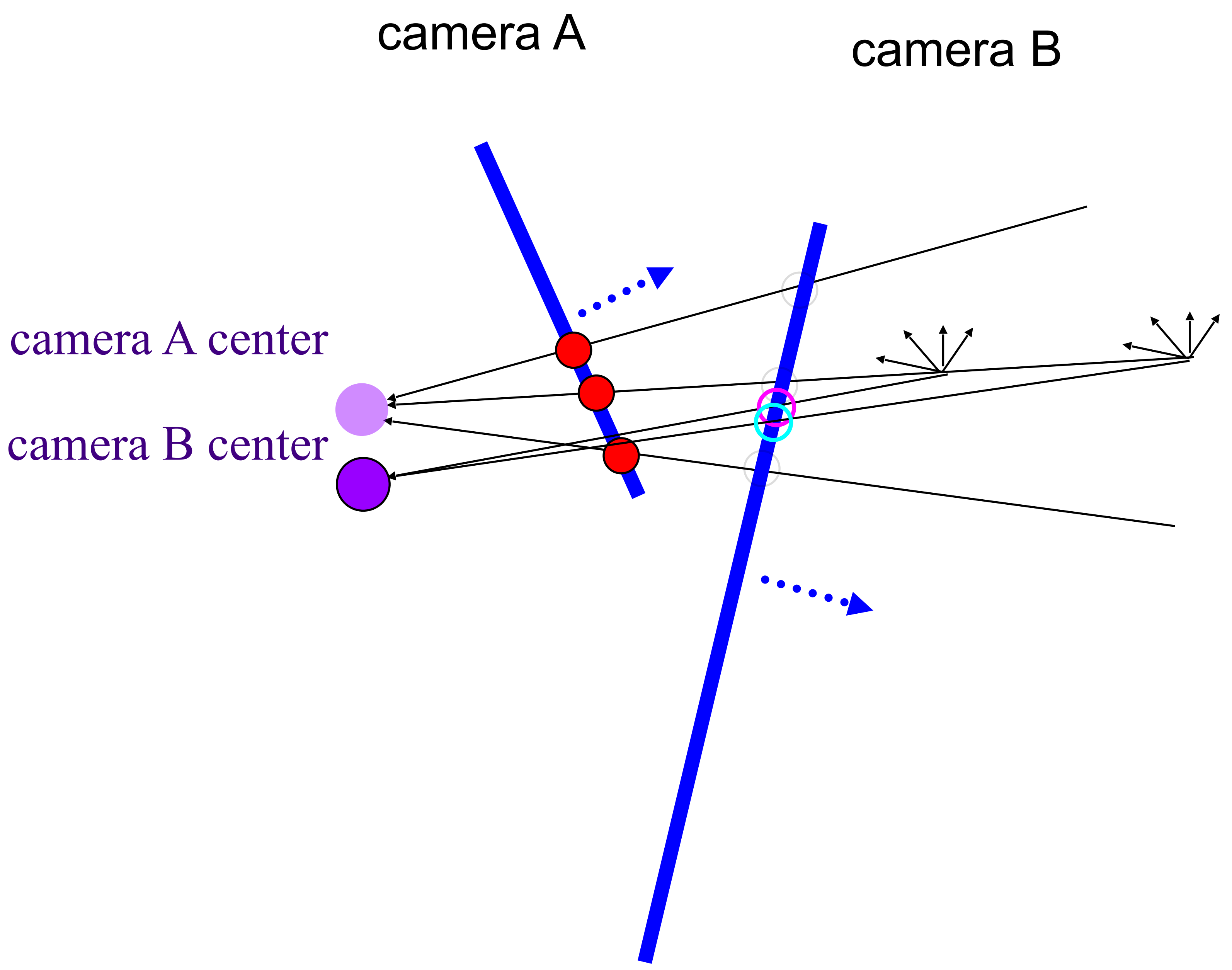


Two cameras with the same center of projection



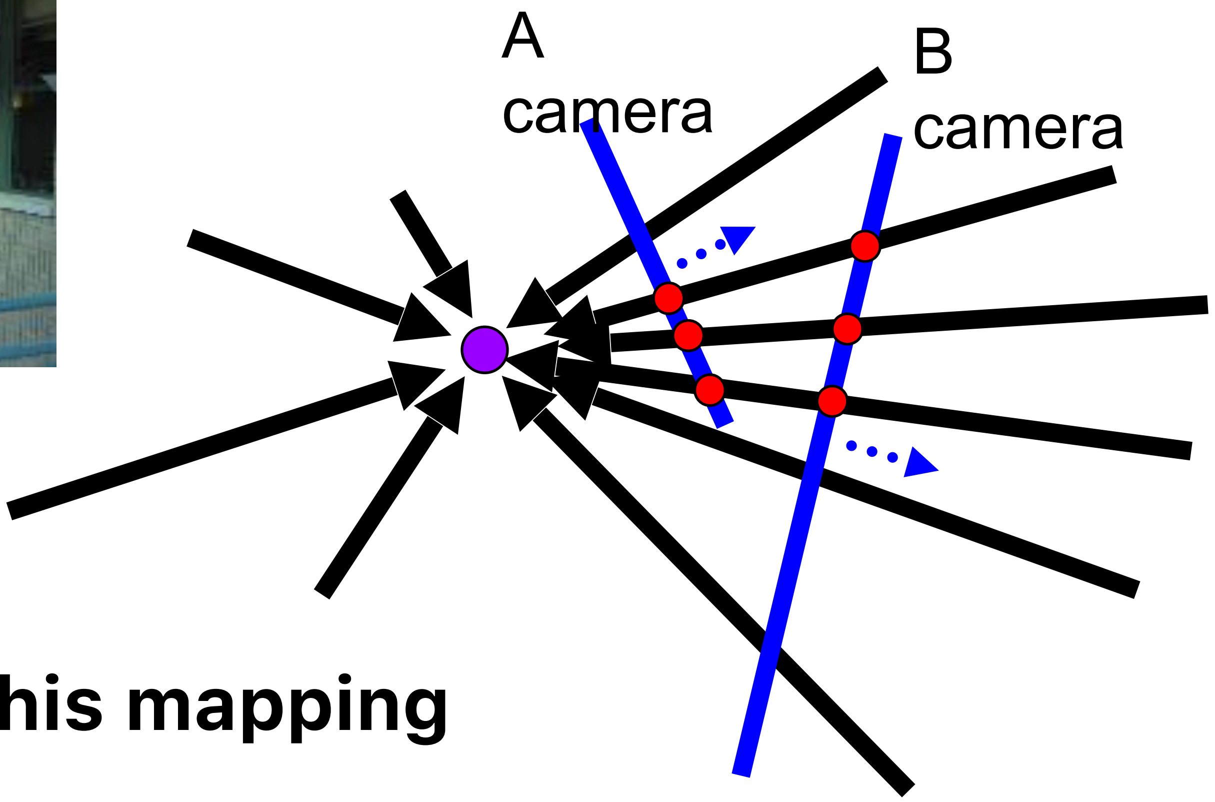
Can generate any synthetic camera view
as long as it has **the same center of projection**!

Two cameras with offset centers of projection



Recap

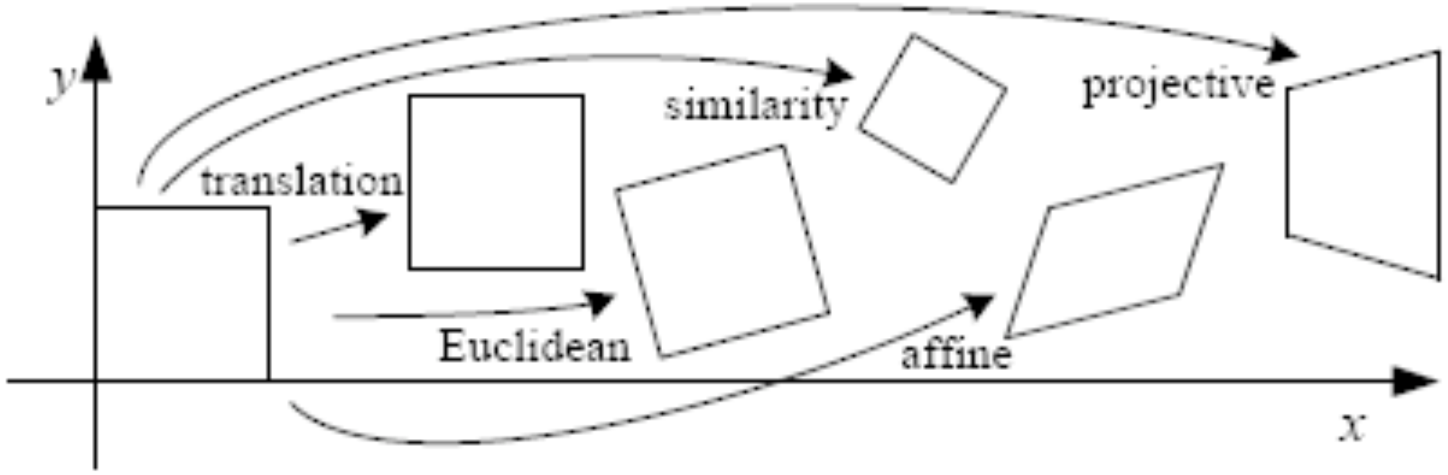
- **When we only rotate the camera depth does not matter**
- **It only performs a 2D warp**
  - one-to-one mapping of the 2D plane
  - plus of course reveals stuff that was outside the field of view



- **Now we just need to figure out this mapping**

Aligning images

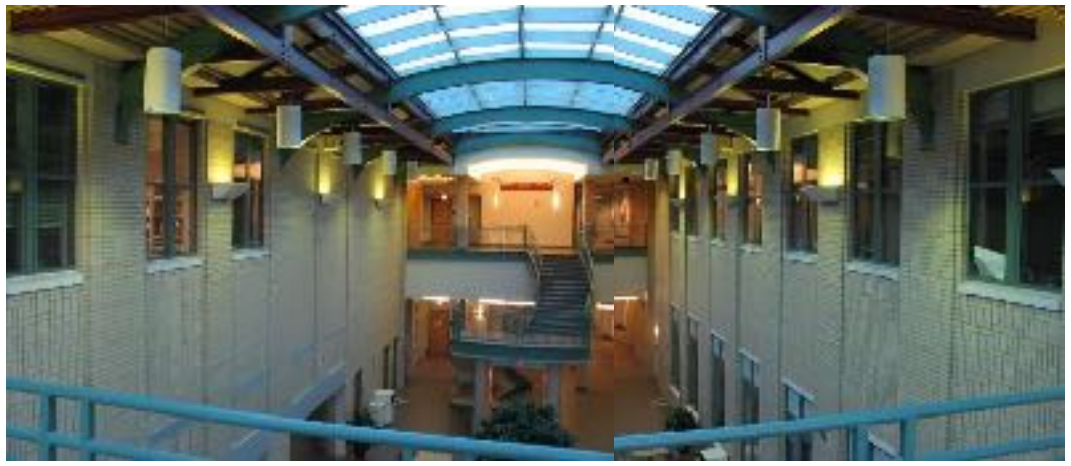
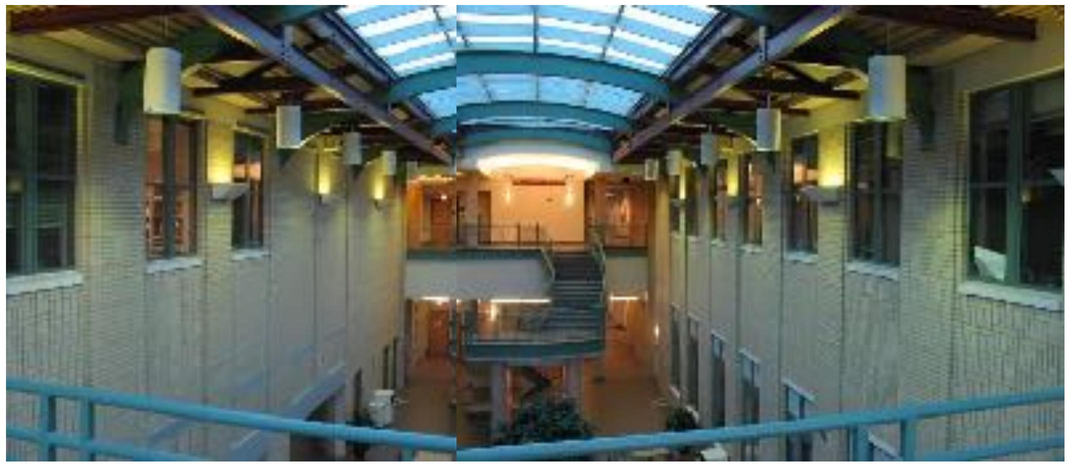
- We have established that pairs of images from the same viewpoint can be aligned through a simple 2D spatial transformation (warp).
- What kind of transformation?



Aligning images: translation?

left on top

right on top

Translations are not enough to align the images

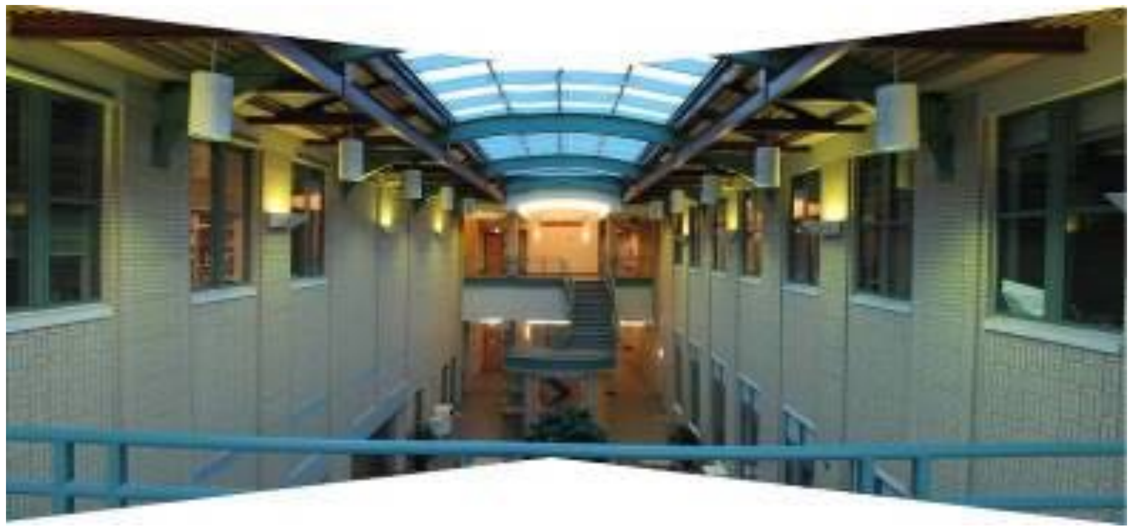


Image Warping

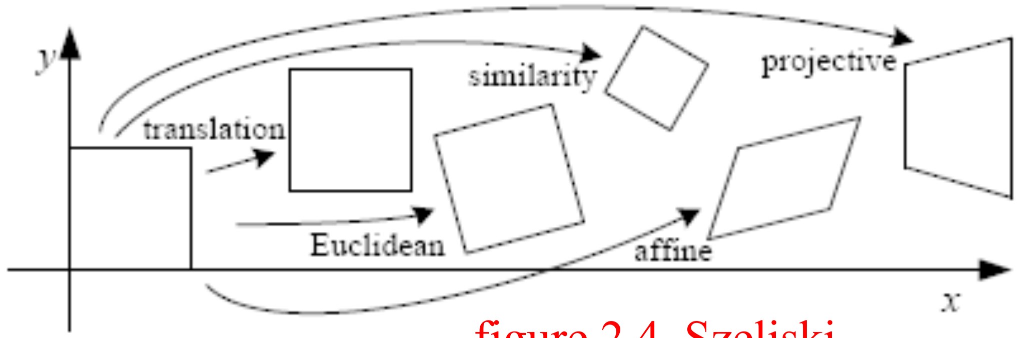
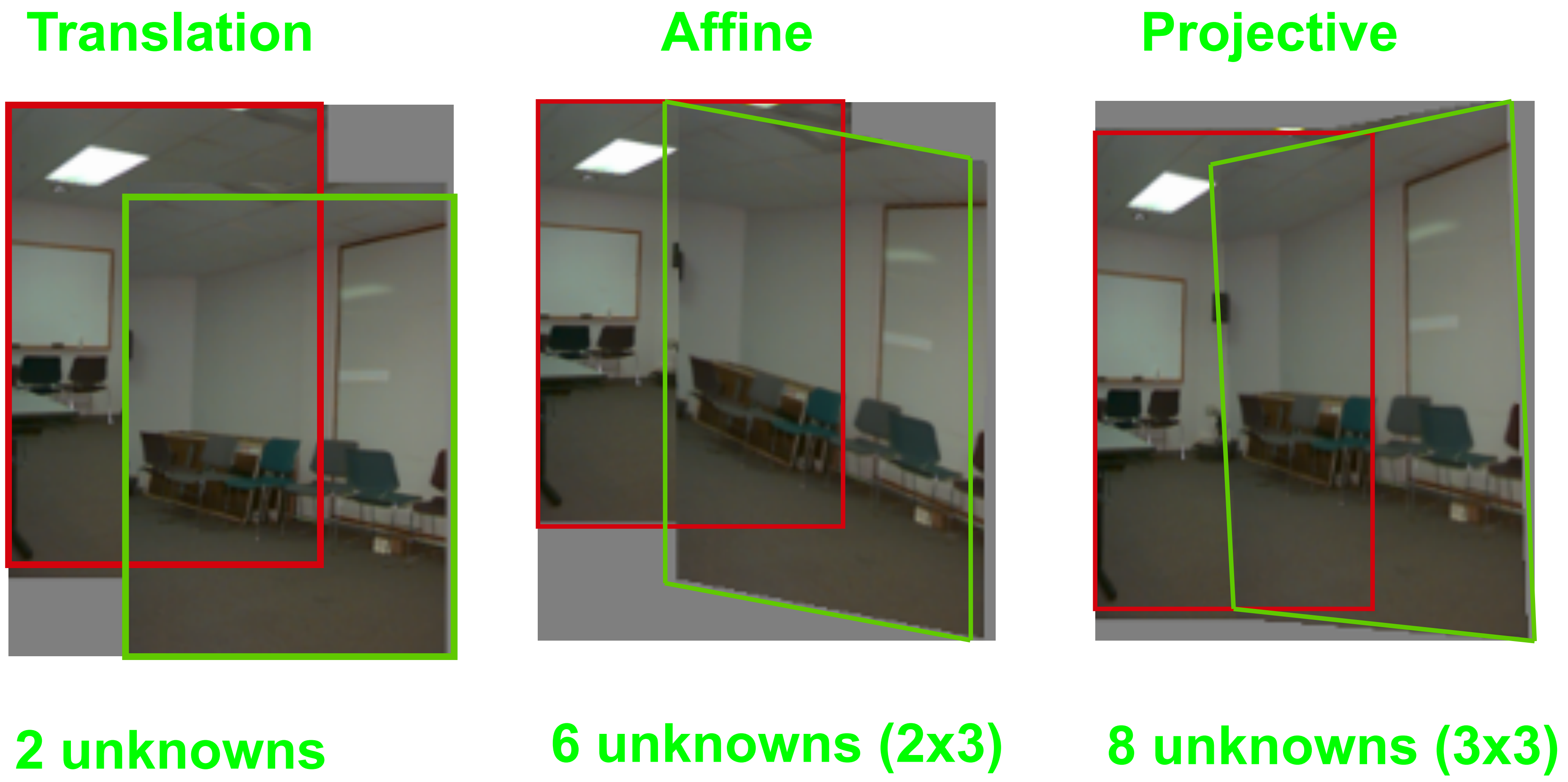


figure 2.4, Szeliski



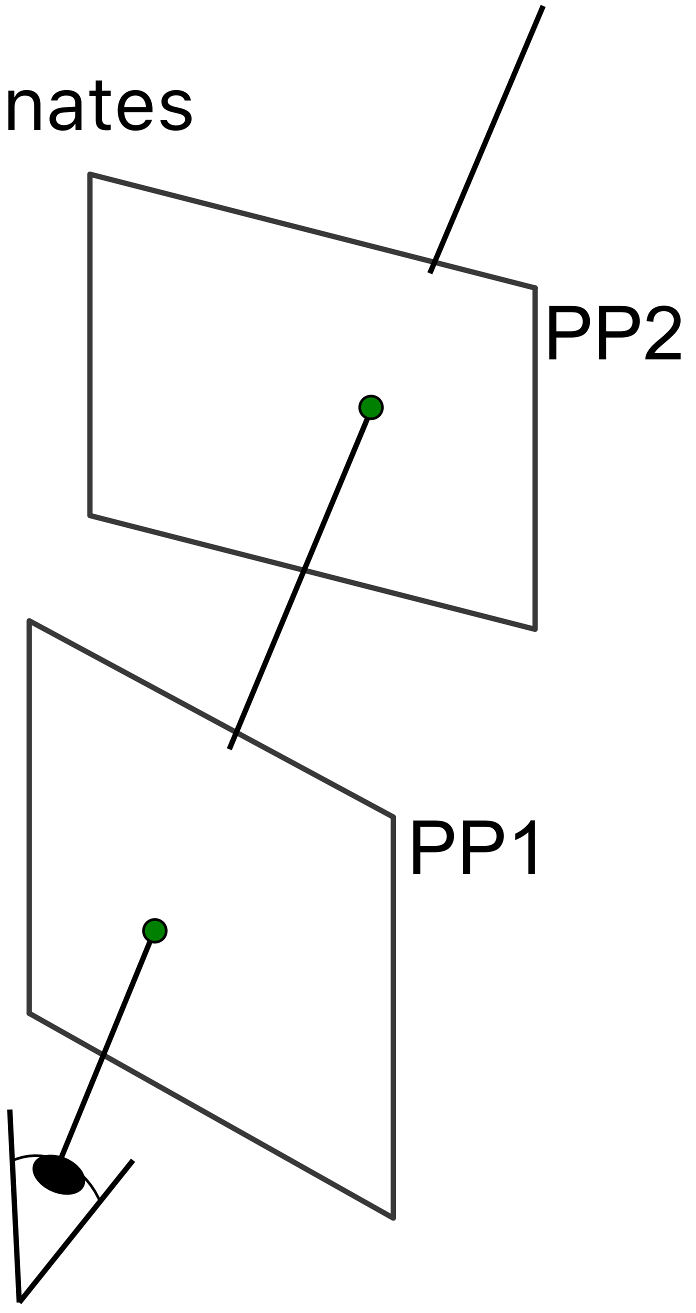
Homography

See Szeliski Sect 2.1.5, Mapping from one camera to another.

- Perspective transform – mapping between any two projection planes with the same center of projection called ***Homography***

- Represented as 3x3 matrix in **homogenous** coordinates

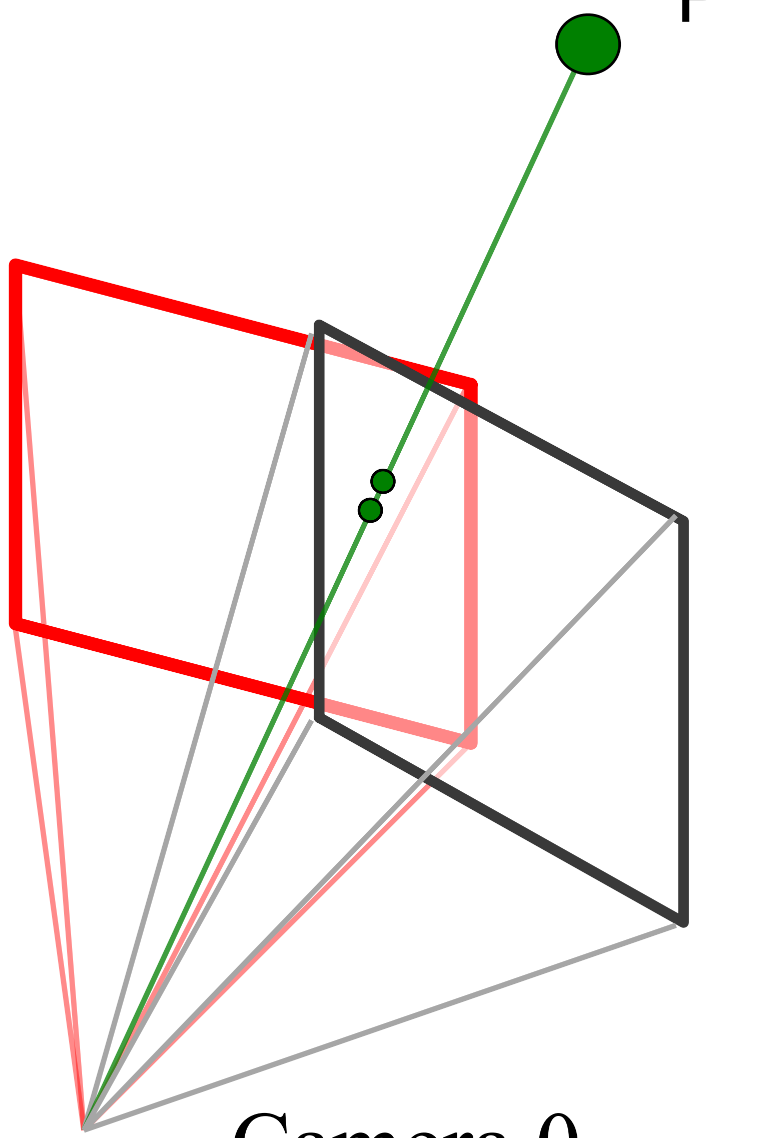
$$\underbrace{\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}}_{\mathbf{p}'} = \underbrace{\begin{bmatrix} * & * & * \\ * & * & * \\ * & * & * \end{bmatrix}}_{\mathbf{H}} \underbrace{\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}}_{\mathbf{p}}$$



To apply a homography H

- Compute w**p′ = Hp** (regular matrix multiply)
- Convert p′ from homogeneous to image coordinates (divide by w)

Homography



Camera 1
parameters: A_1

Camera 0
parameters: A_0

$$\tilde{x}_0 = A_0 \tilde{P} = A_0 \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\tilde{x}_1 = A_1 \tilde{P}$$

$$A_0 = K[I|0] = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$A_1 = K[R \mid 0]$$

Two cameras with the same center of projection

$$\tilde{x}_0 = A_0 \tilde{P} = A_0 \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad A_0 = K[I|0] = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\tilde{x}_0 = KP \qquad P = (X, Y, Z)^T$$

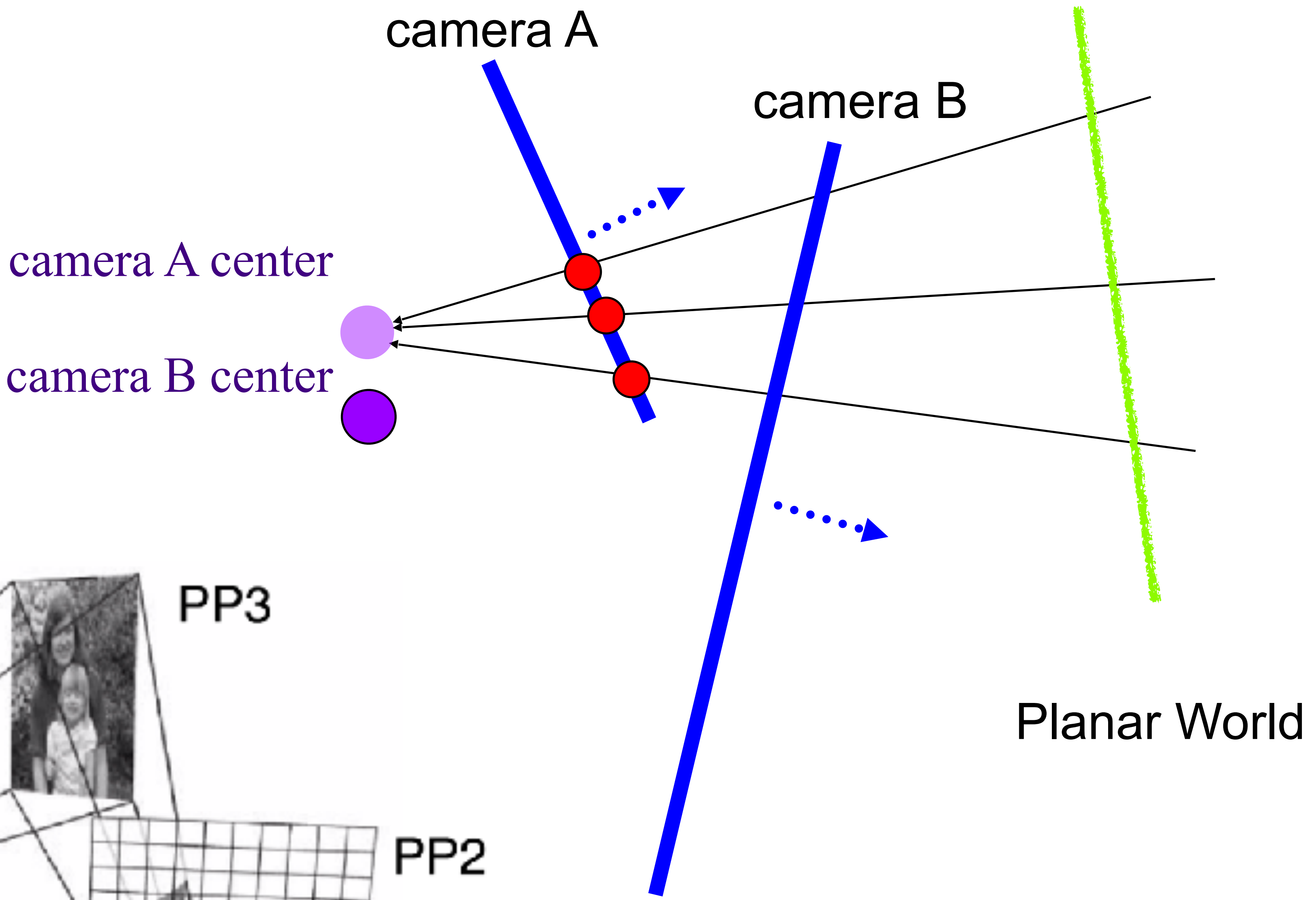
$$\tilde{x}_1 = KRP$$

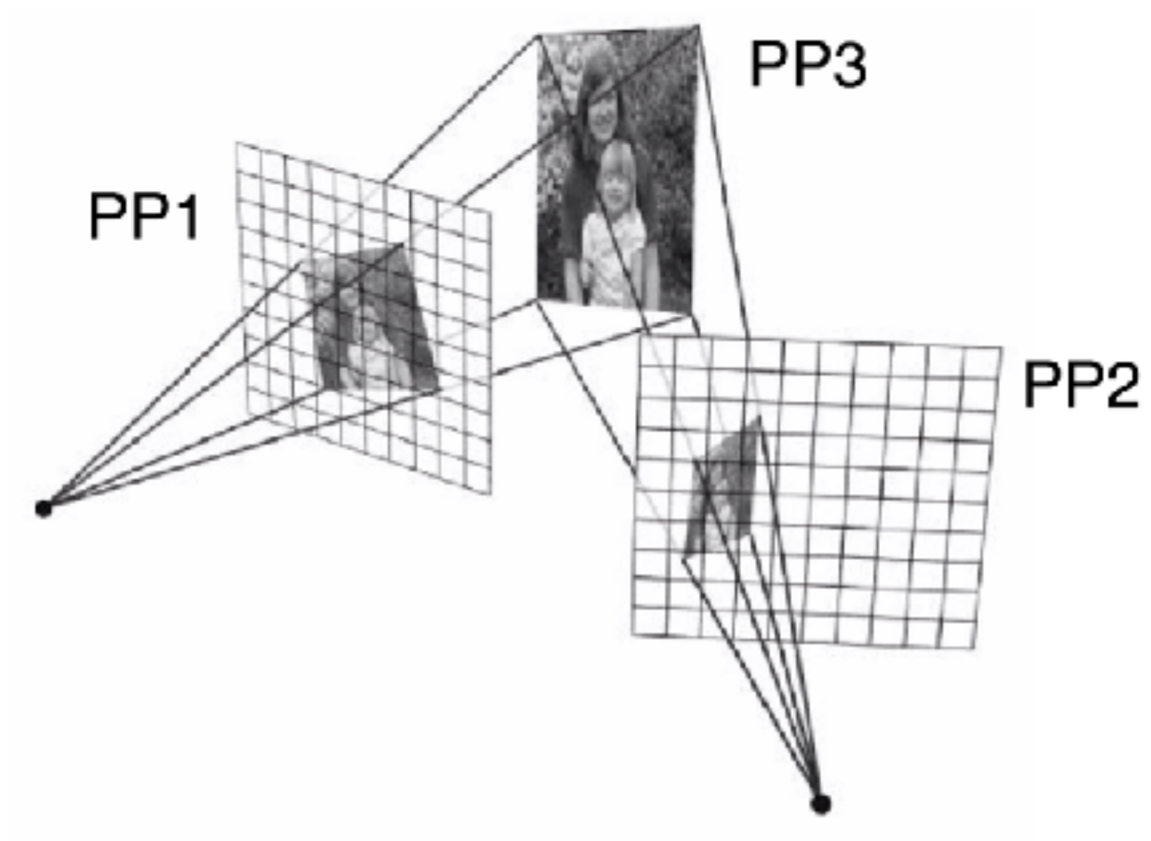
- We seek for a mapping such that: $\tilde{x}_0 = M_{10} \tilde{x}_1$

$$\tilde{x}_0 = KR^{-1}K^{-1}\tilde{x}_1$$

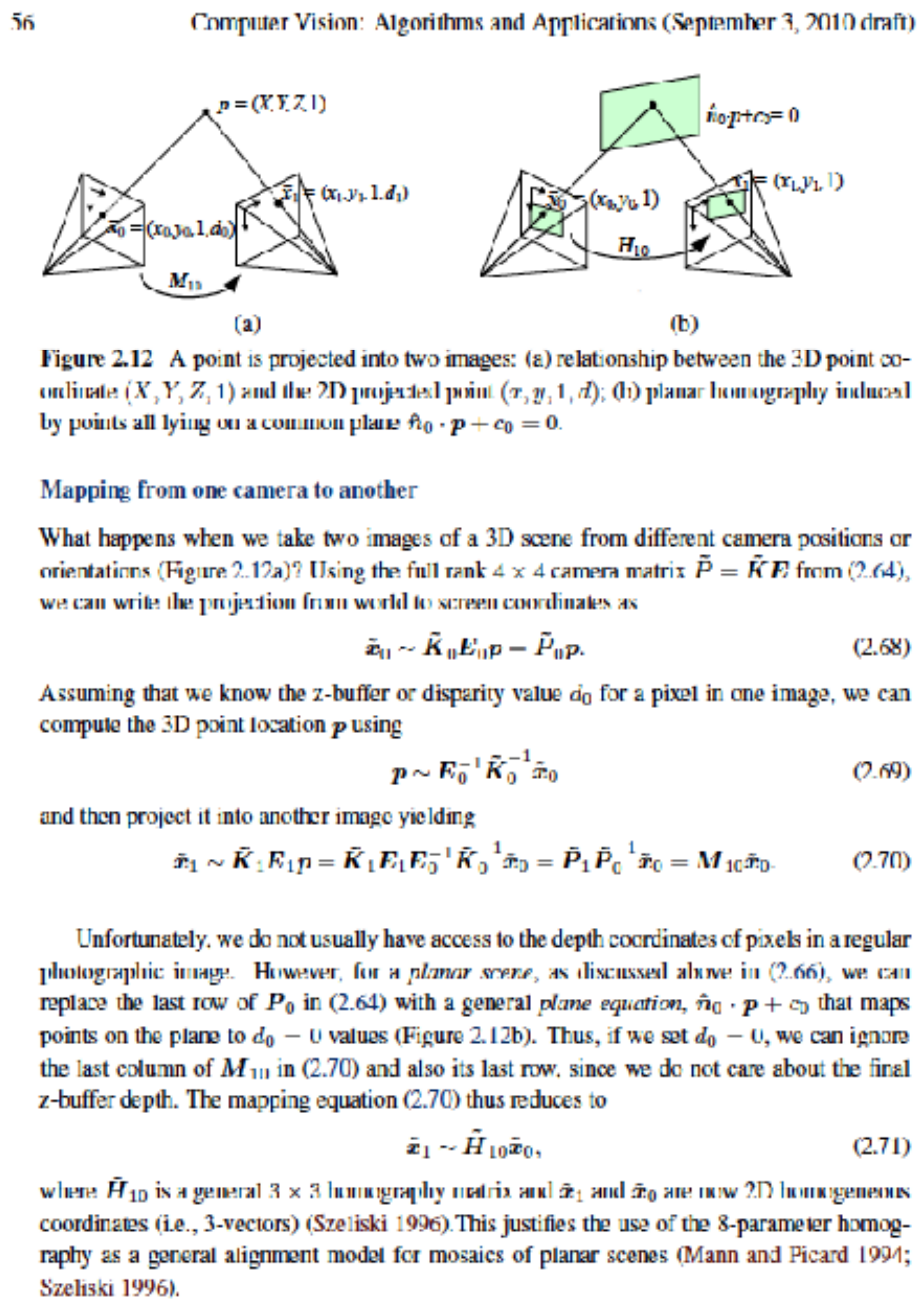
How many pairs of points does it take to specify M_10?

Planar objects

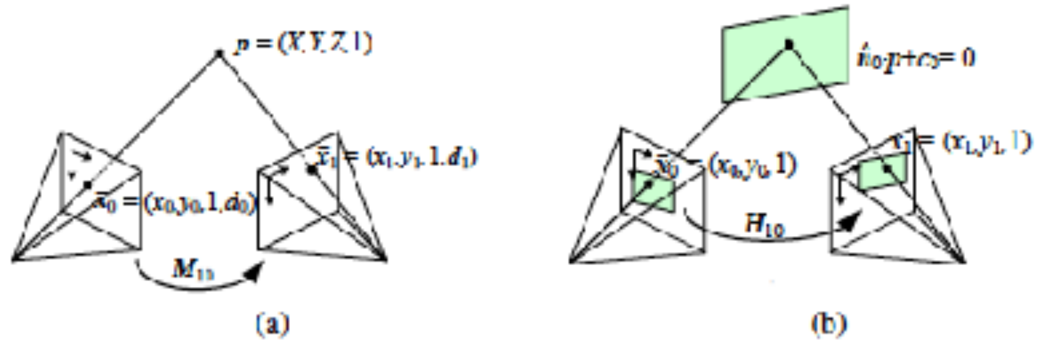




Planar objects



56 Computer Vision: Algorithms and Applications (September 3, 2010 draft)



$p = (X,Y,Z,1)$
$\tilde{x}_0 = (x_0, y_0, 1, d_0)$
$\tilde{x}_1 = (x_1, y_1, 1, d_1)$
M_{10}
(a)
$\hat{n}_0 \cdot p + c_0 = 0$
$\tilde{x}_0\ (x_0, y_0, 1)$
$\tilde{x}_1 = (x_1, y_1, 1)$
$\tilde{H}_{10}$
(b)

Figure 2.12 A point is projected into two images: (a) relationship between the 3D point coordinate $(X, Y, Z, 1)$ and the 2D projected point $(x, y, 1, d)$; (b) planar homography induced by points all lying on a common plane $\hat{n}_0 \cdot p + c_0 = 0$.

Mapping from one camera to another

What happens when we take two images of a 3D scene from different camera positions or orientations (Figure 2.12a)? Using the full rank 4×4 camera matrix $\tilde{P} = \tilde{K}E$ from (2.64), we can write the projection from world to screen coordinates as

$$\tilde{x}_0 \sim \tilde{K}_0 E_0 p = \tilde{P}_0 p. \tag{2.68}$$

Assuming that we know the z-buffer or disparity value d_0 for a pixel in one image, we can compute the 3D point location p using

$$p \sim E_0^{-1} \tilde{K}_0^{-1} \tilde{x}_0 \tag{2.69}$$

and then project it into another image yielding

$$\tilde{x}_1 \sim \tilde{K}_1 E_1 p = \tilde{K}_1 E_1 E_0^{-1} \tilde{K}_0^{-1} \tilde{x}_0 = \tilde{P}_1 \tilde{P}_0^{-1} \tilde{x}_0 = M_{10} \tilde{x}_0. \tag{2.70}$$

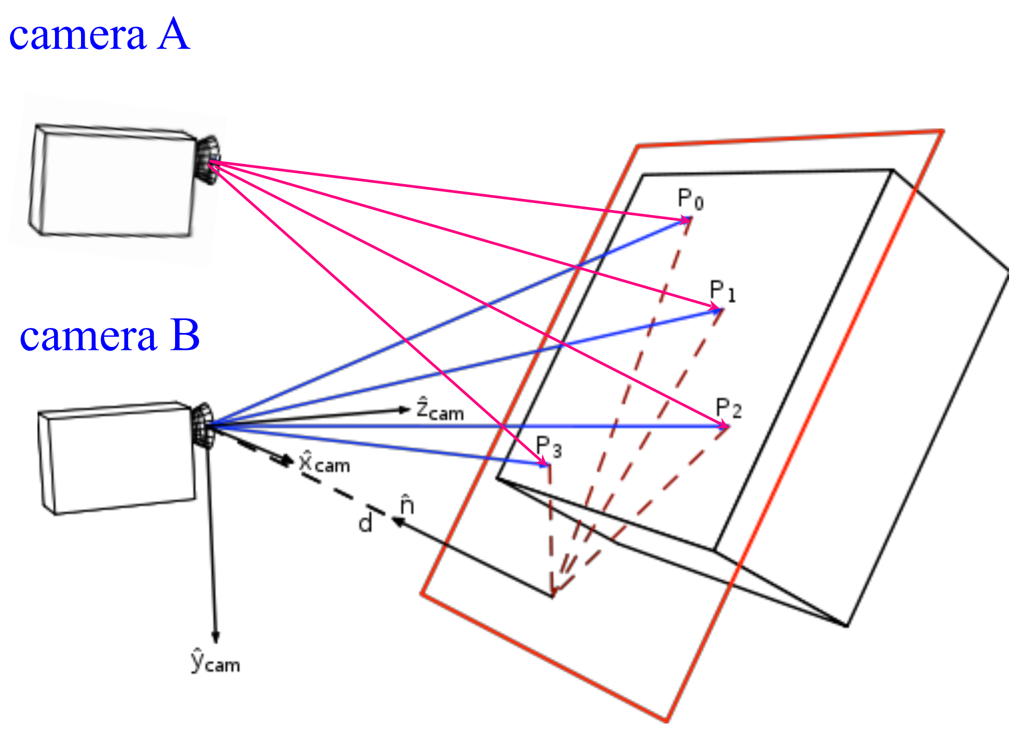
Unfortunately, we do not usually have access to the depth coordinates of pixels in a regular photographic image. However, for a *planar scene*, as discussed above in (2.66), we can replace the last row of P_0 in (2.64) with a general *plane equation*, $\hat{n}_0 \cdot p + c_0$ that maps points on the plane to $d_0 = 0$ values (Figure 2.12b). Thus, if we set $d_0 = 0$, we can ignore the last column of M_{10} in (2.70) and also its last row, since we do not care about the final z-buffer depth. The mapping equation (2.70) thus reduces to

$$\tilde{x}_1 \sim \tilde{H}_{10} \tilde{x}_0, \tag{2.71}$$

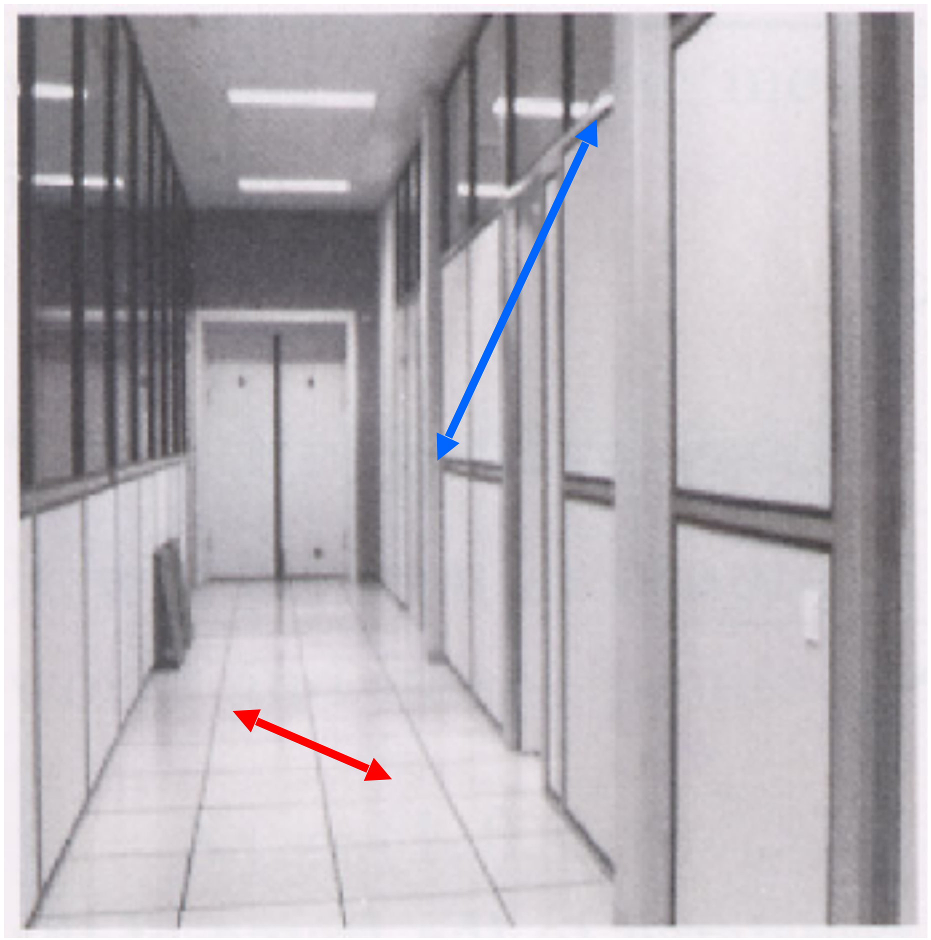
where $\tilde{H}_{10}$ is a general 3×3 homography matrix and $\tilde{x}_1$ and $\tilde{x}_0$ are now 2D homogeneous coordinates (i.e., 3-vectors) (Szeliski 1996). This justifies the use of the 8-parameter homography as a general alignment model for mosaics of planar scenes (Mann and Picard 1994; Szeliski 1996).

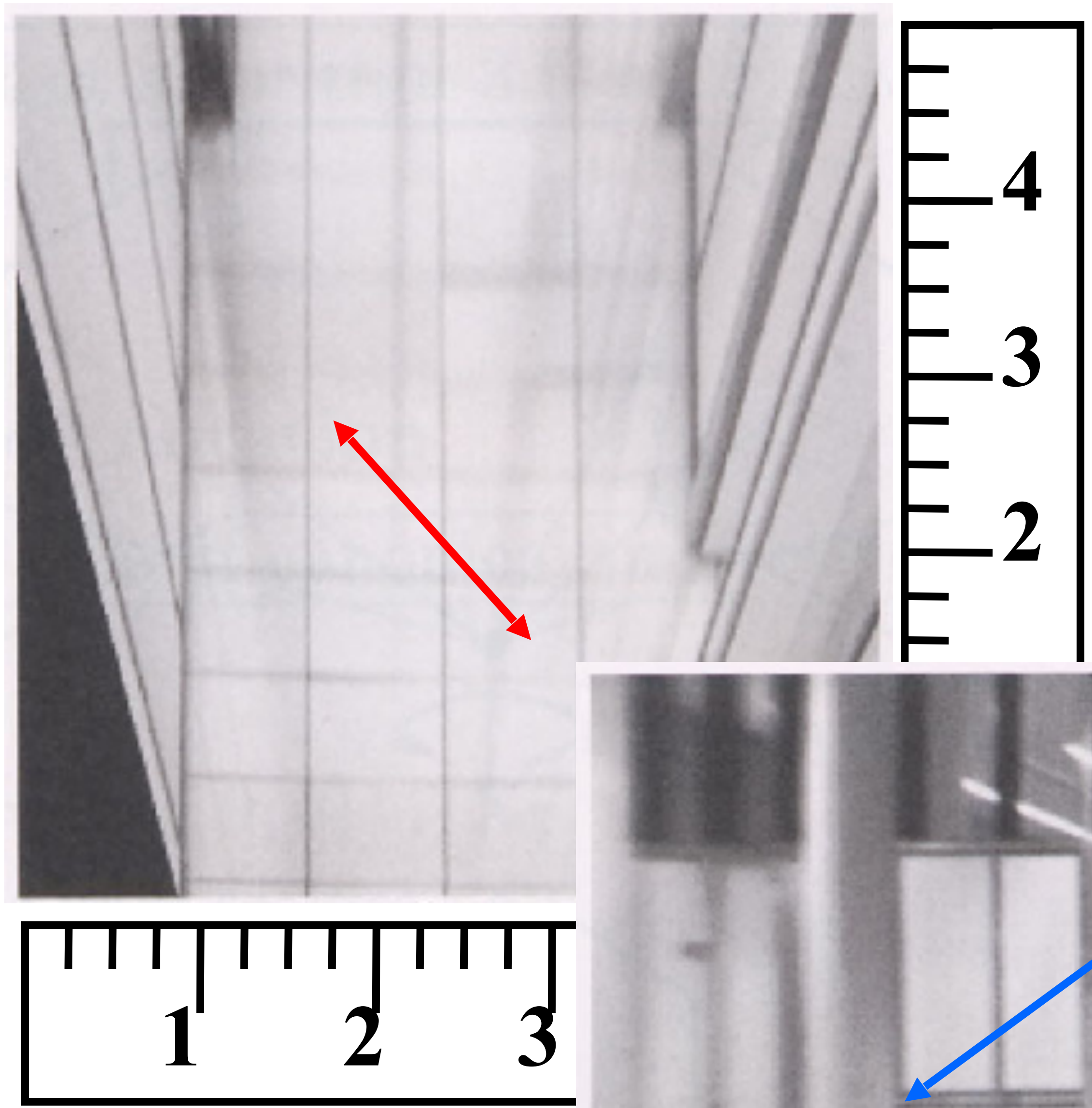
From Szeliski book

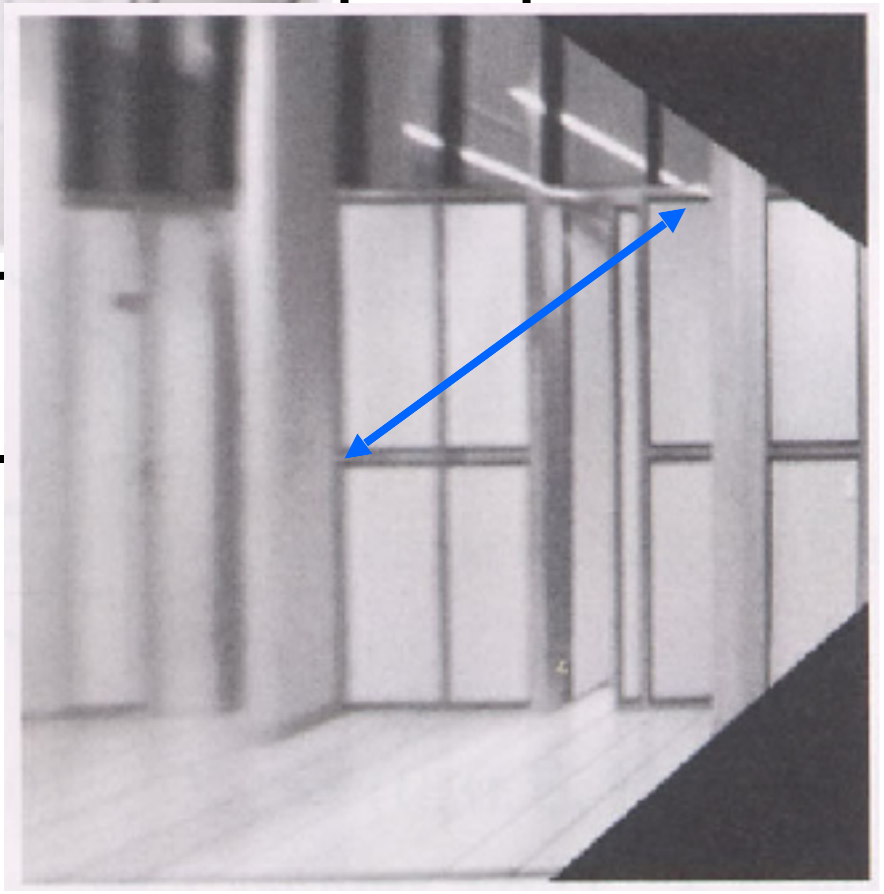
Images of planar objects, taken by generically offset cameras, are also related by a homography.



Measurements on planes



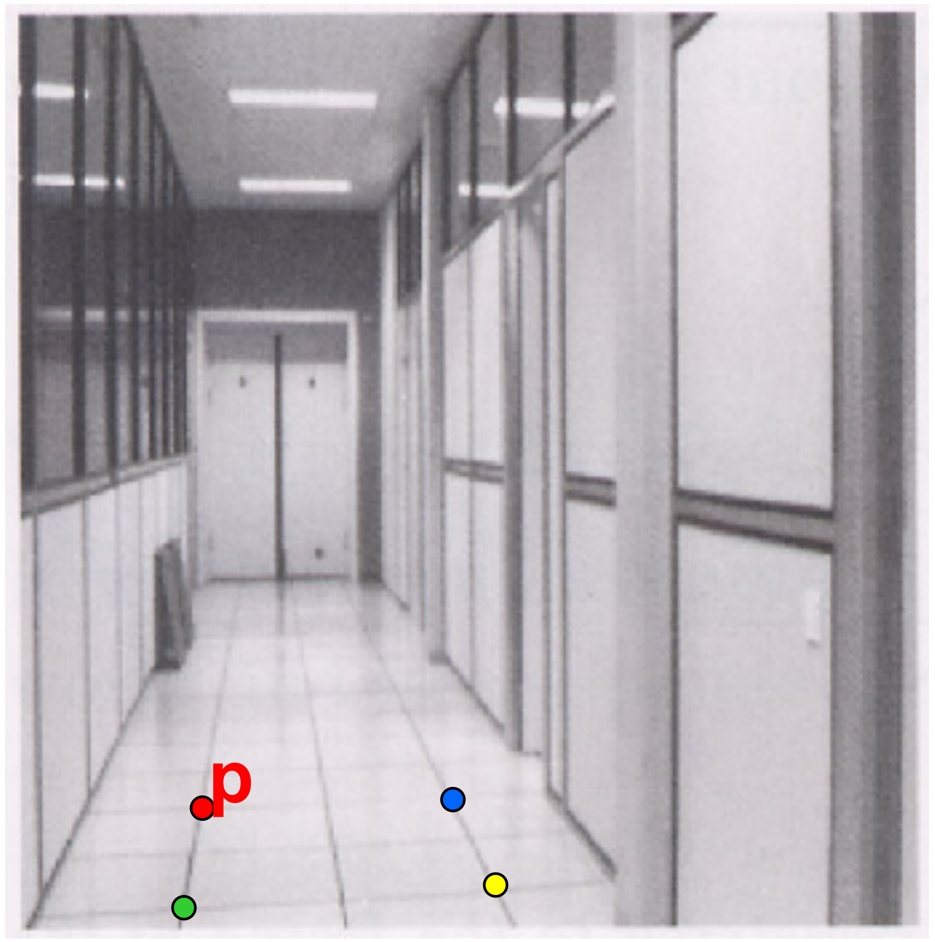


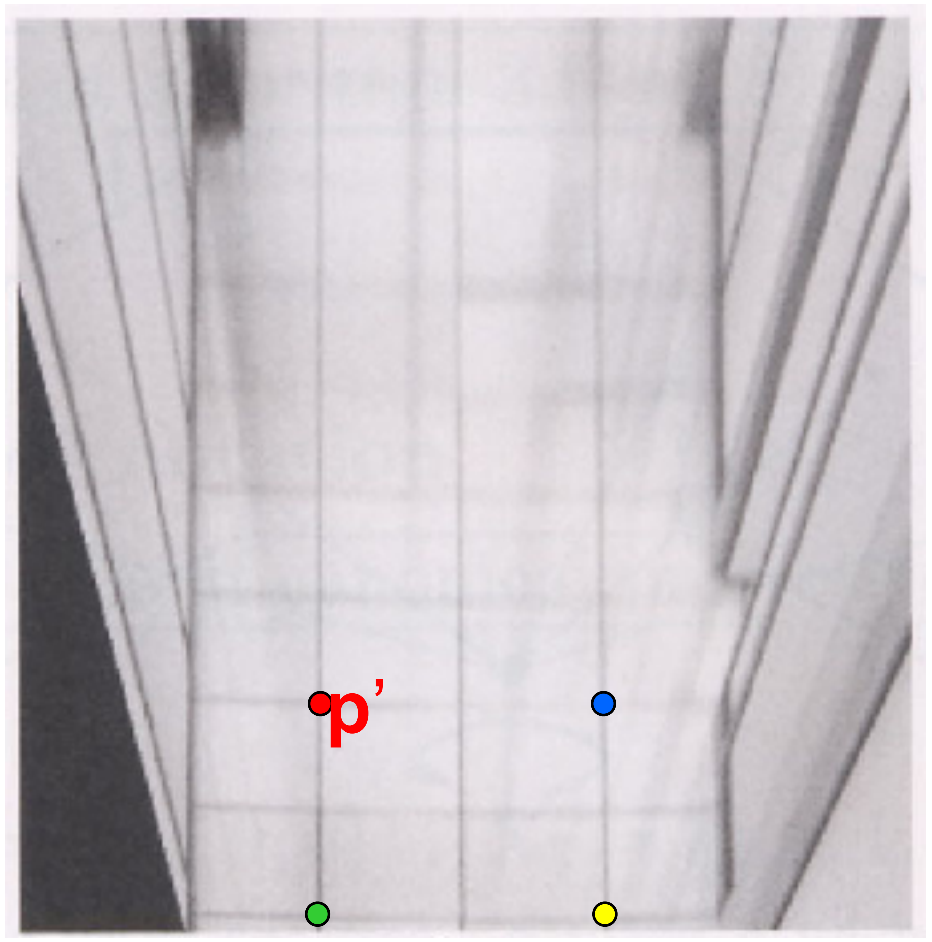


Approach: unwarp then measure

How to unwarp?

Image rectification





To unwarp (rectify) an image

- solve for homography **H** given **p** and **p**'
- solve equations of the form: w**p**' = **Hp**

– linear in unknowns

– H is defined up to an arbitrary scale factor

– how many points are necessary to solve for **H**?

Solving for homographies

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \cong \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\begin{aligned} x'_i &= \frac{h_{00}x_i + h_{01}y_i + h_{02}}{h_{20}x_i + h_{21}y_i + h_{22}} \\ y'_i &= \frac{h_{10}x_i + h_{11}y_i + h_{12}}{h_{20}x_i + h_{21}y_i + h_{22}} \end{aligned}$$

$$\begin{aligned} x'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{00}x_i + h_{01}y_i + h_{02} \\ y'_i(h_{20}x_i + h_{21}y_i + h_{22}) &= h_{10}x_i + h_{11}y_i + h_{12} \end{aligned}$$

$$\begin{bmatrix} x_i & y_i & 1 & 0 & 0 & 0 & -x'_i x_i & -x'_i y_i & -x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & -y'_i x_i & -y'_i y_i & -y'_i \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

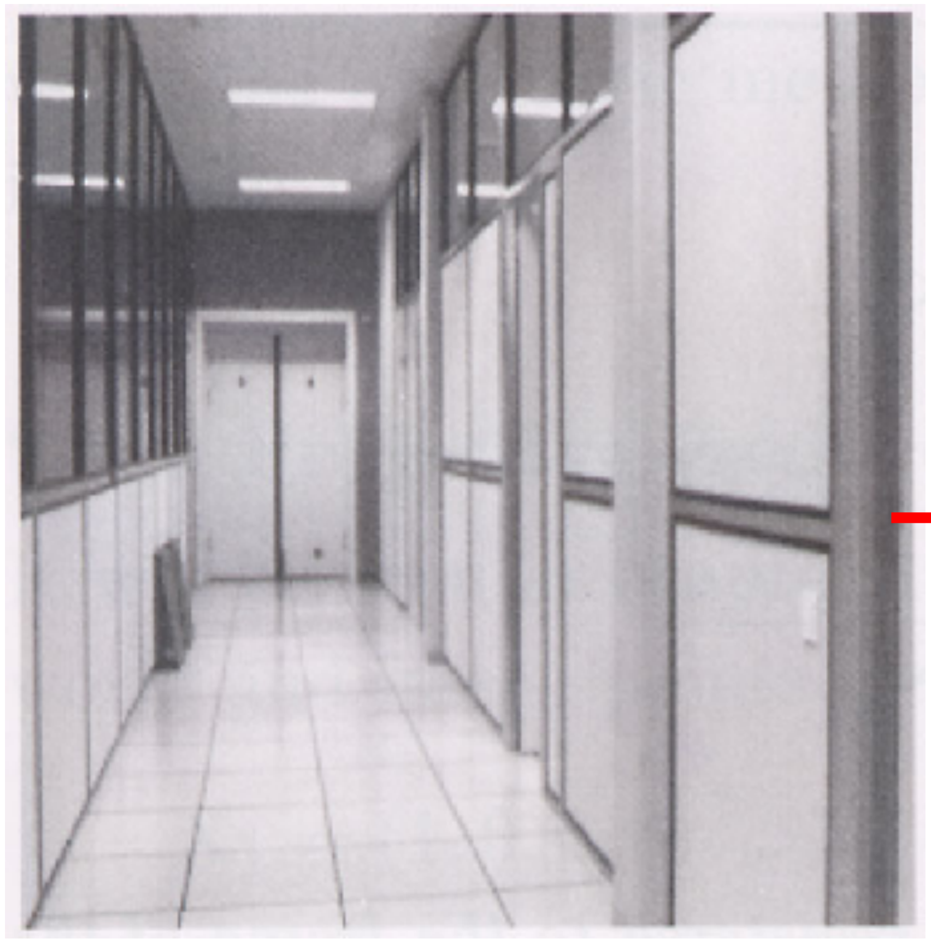
Solving for homographies

$$\underbrace{\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\ & & & & \vdots & & & & \\ x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n \end{bmatrix}}_{\substack{\mathbf{A} \\ 2n \times 9}} \underbrace{\begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix}}_{\substack{\mathbf{h} \\ 9}} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}}_{\substack{\mathbf{0} \\ 2n}}$$

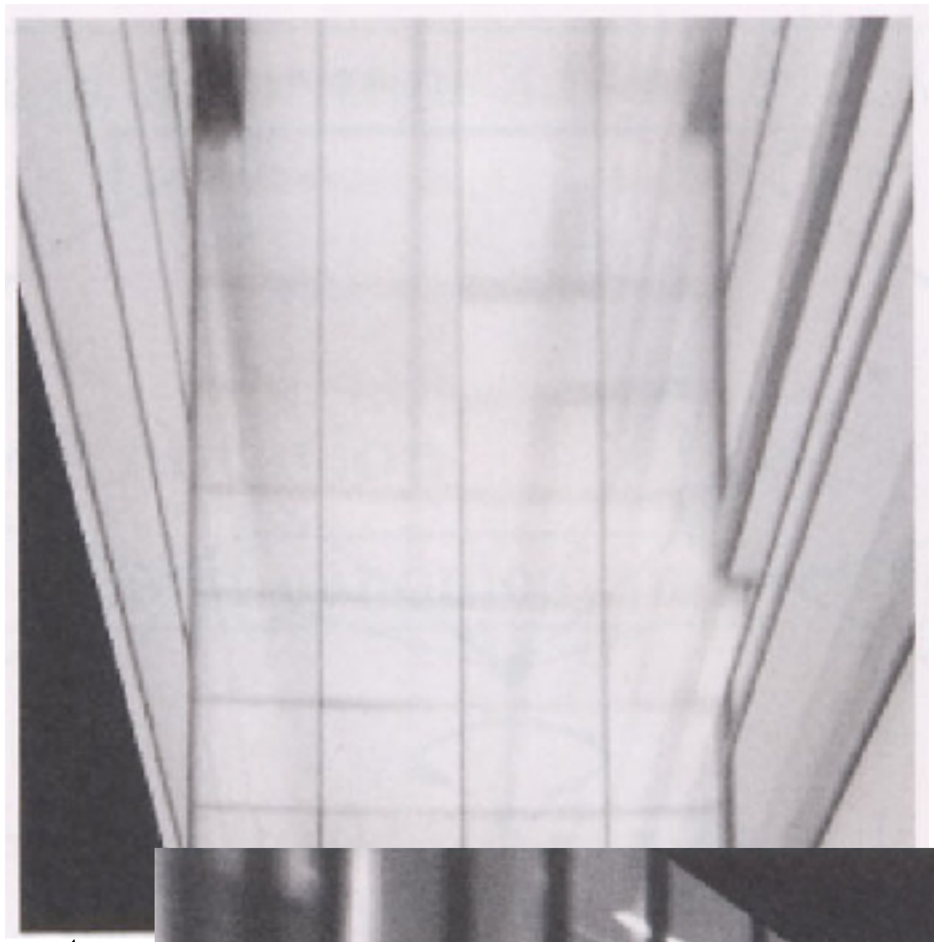
Defines a least squares problem: $\text{minimize } \|\mathbf{Ah} - \mathbf{0}\|^2$

- Since **h** is only defined up to scale, solve for unit vector $\hat{\mathbf{h}}$
- Solution: $\hat{\mathbf{h}}$ = eigenvector of $\mathbf{A}^T\mathbf{A}$ with smallest eigenvalue
- Works with 4 or more points

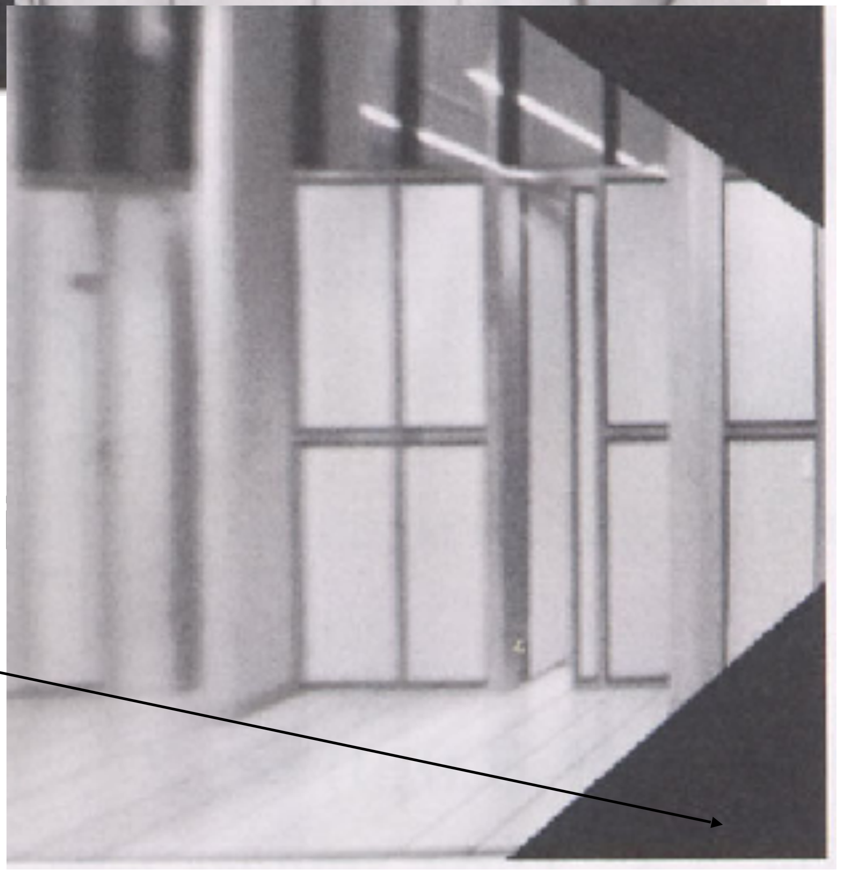
Image warping with homographies



homography so that image is parallel to floor



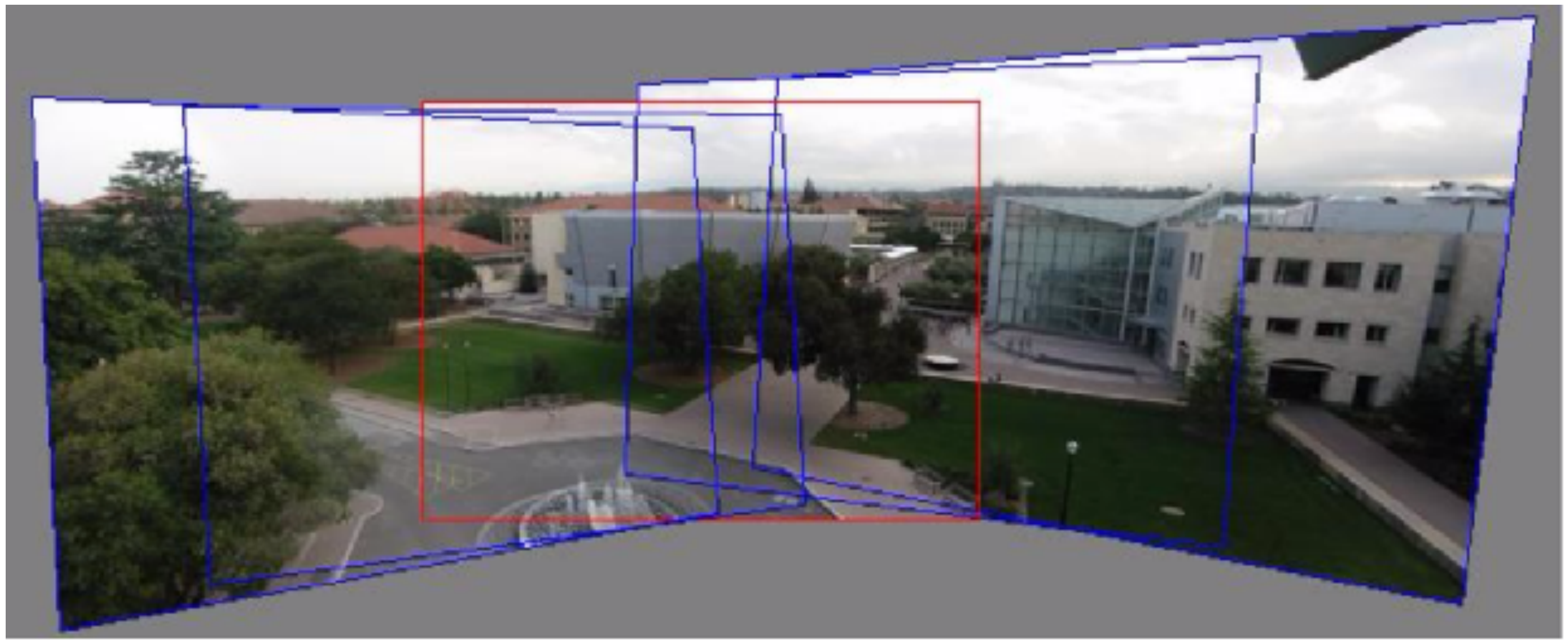
homography so that image is parallel to right wall



black area where no pixel maps to

Automatic image mosaicing

- **Basic Procedure**
  - Take a sequence of images **from the same position.**
    - Rotate the camera about its optical center (entrance pupil).
  - Robustly compute the homography transformation between second image and first.
  - Transform (warp) the second image to overlap with first.
  - Blend the two together to create a mosaic.
  - If there are more images, repeat.



Robust feature matching through RANSAC

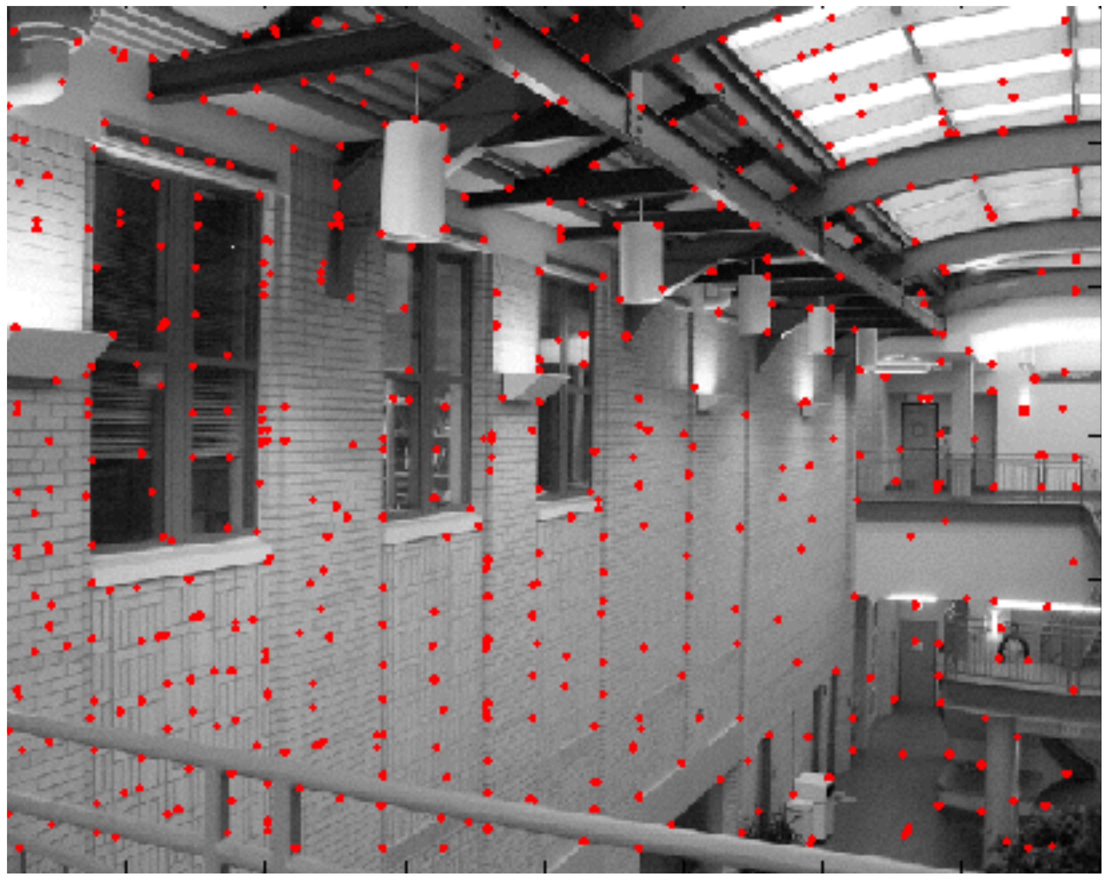
© **Krister Parmstrand**

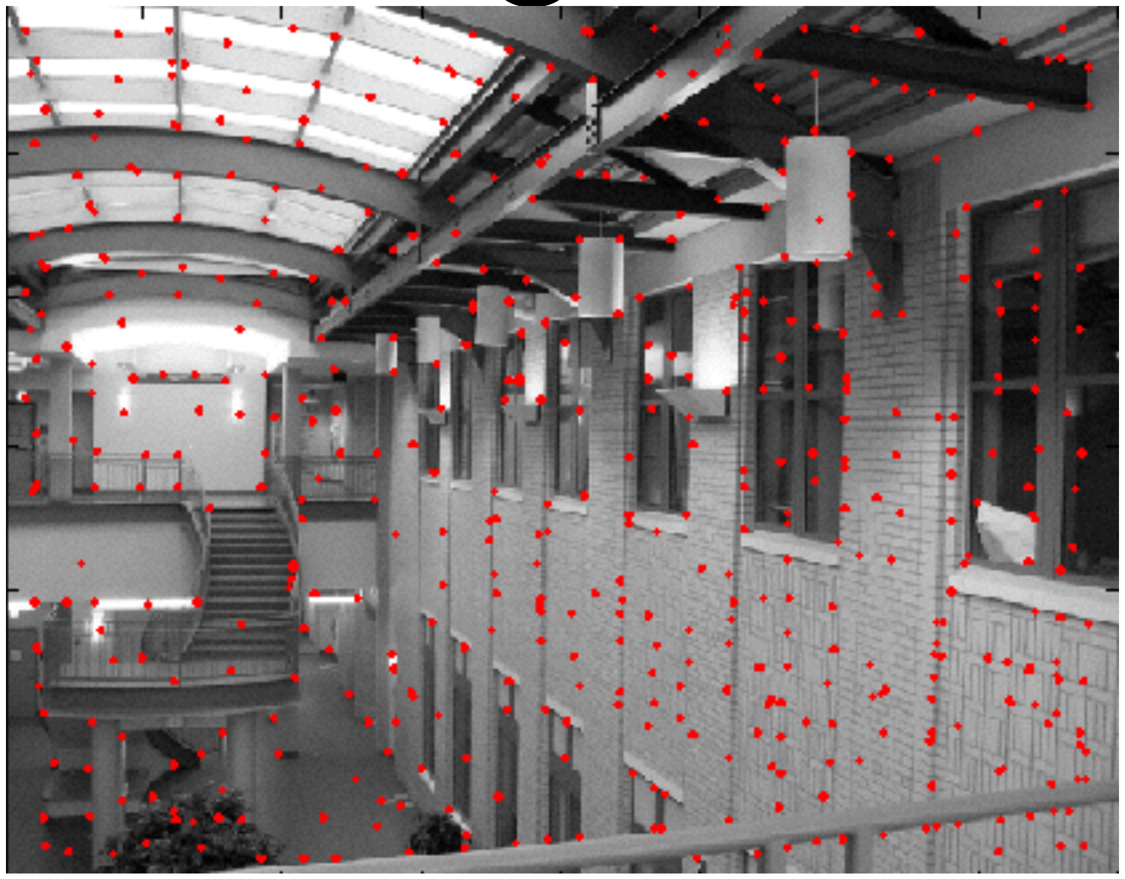
Nikon D70. Stitched Panorama. The sky has been retouched. No other image manipulation.

with a lot of slides stolen from
Steve Seitz and Rick Szeliski

15-463: Computational Photography
Alexei Efros, CMU, Fall 2005

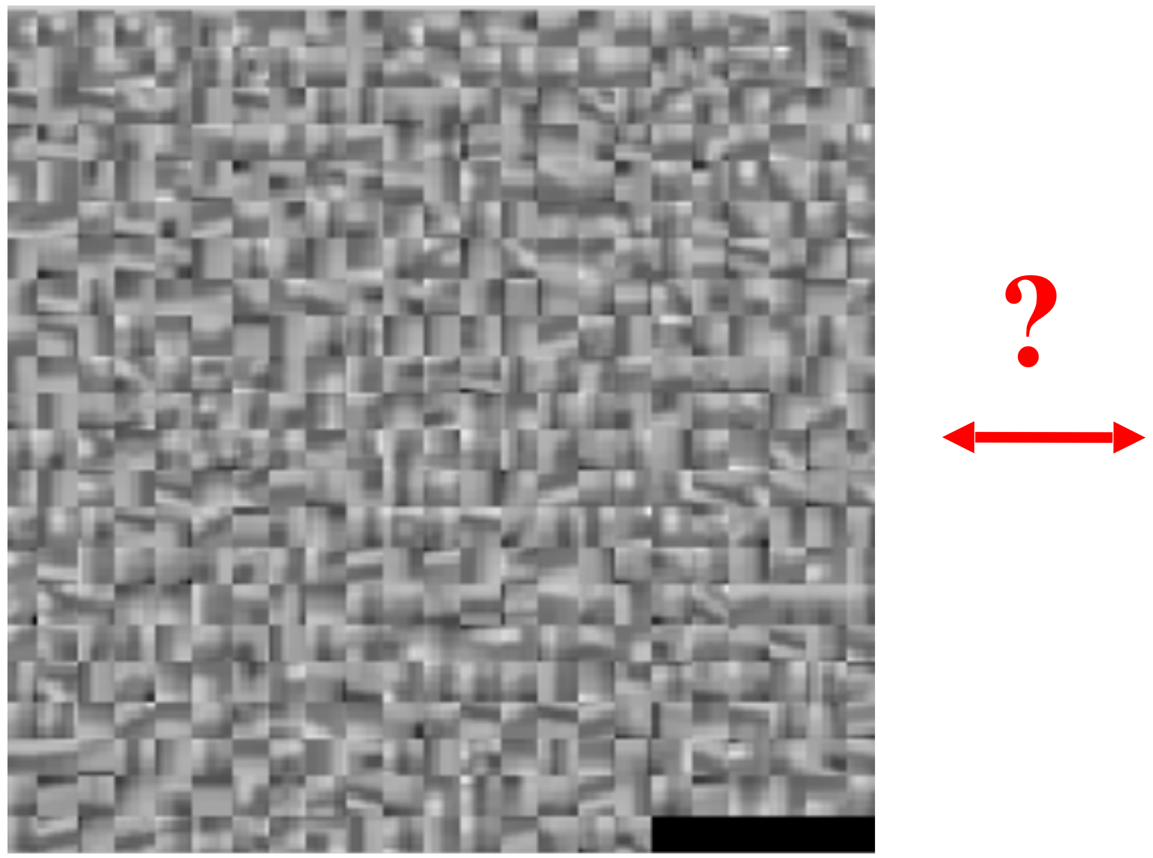
Feature matching

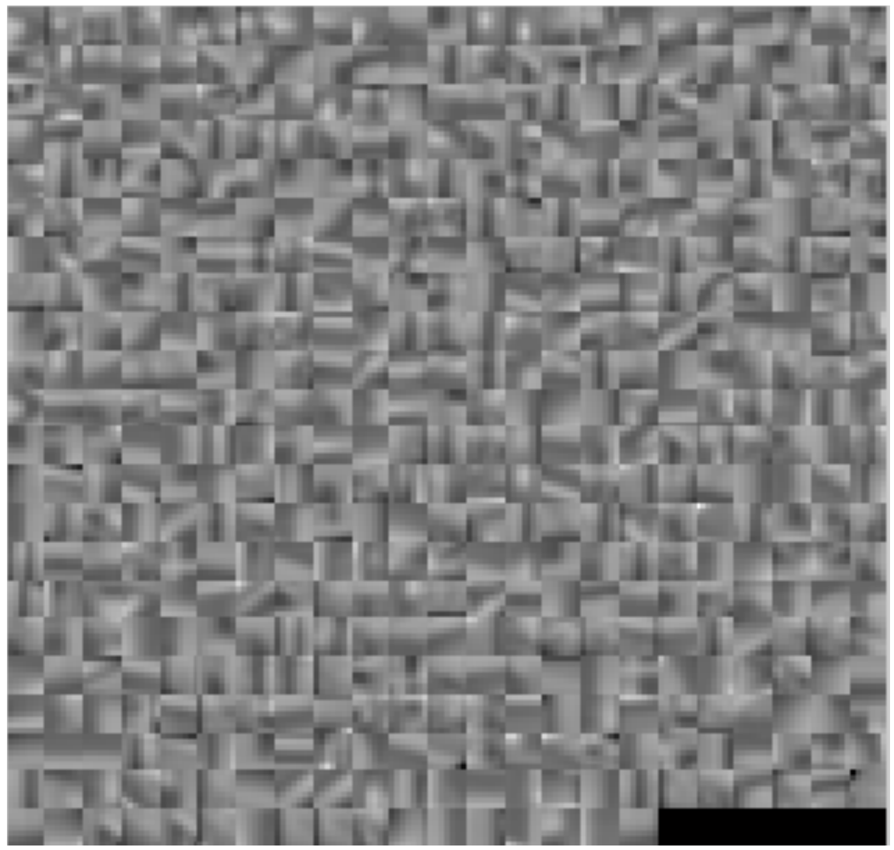




descriptors for left image feature points

descriptors for right image feature points





Strategies to match images robustly

(a) Working with individual features: For each feature point, find most similar point in other image (SIFT distance)

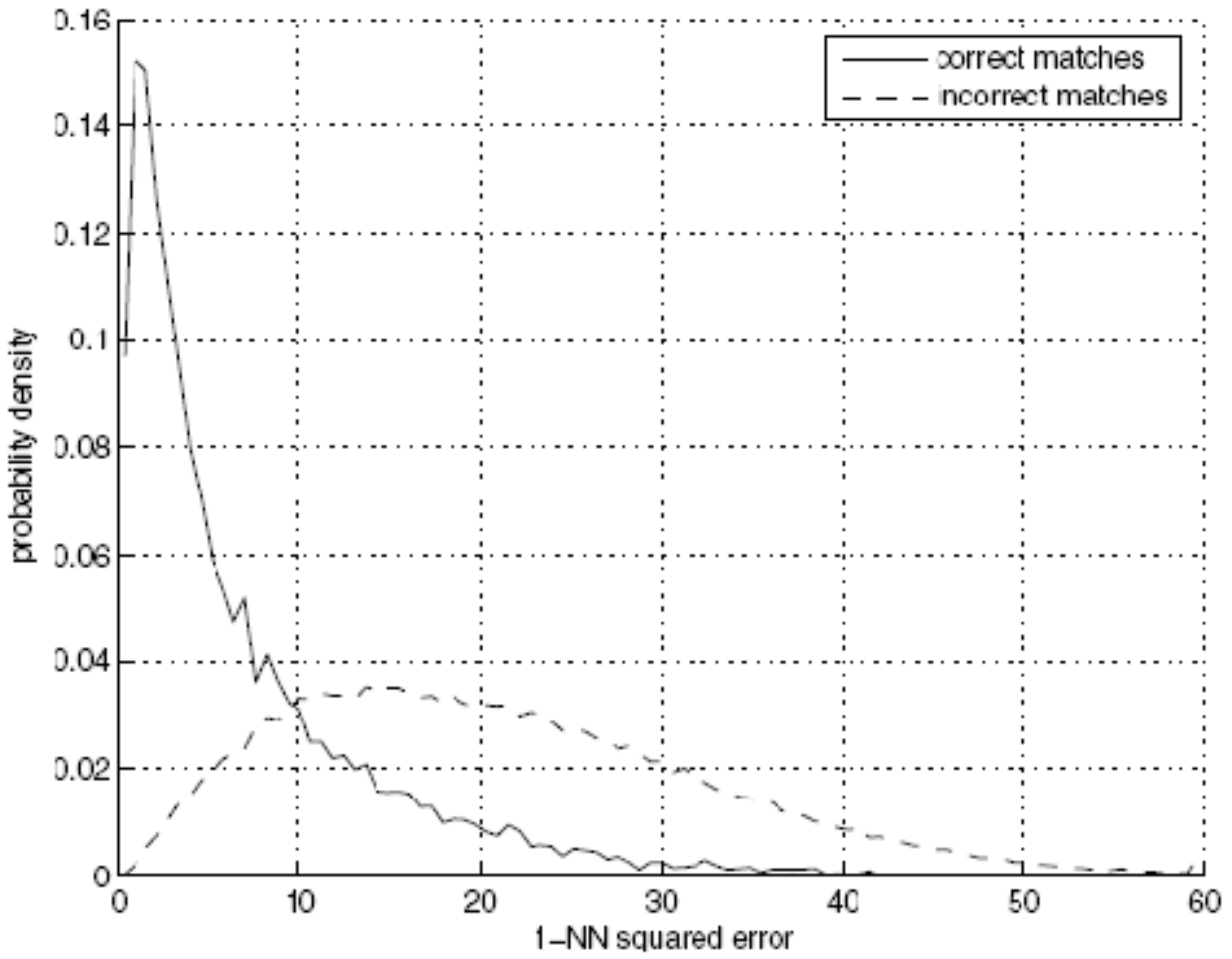
Reject ambiguous matches where there are too many similar points

(b) Working with all the features: Given some good feature matches, look for possible homographies relating the two images

Reject homographies that don’t have many feature matches.

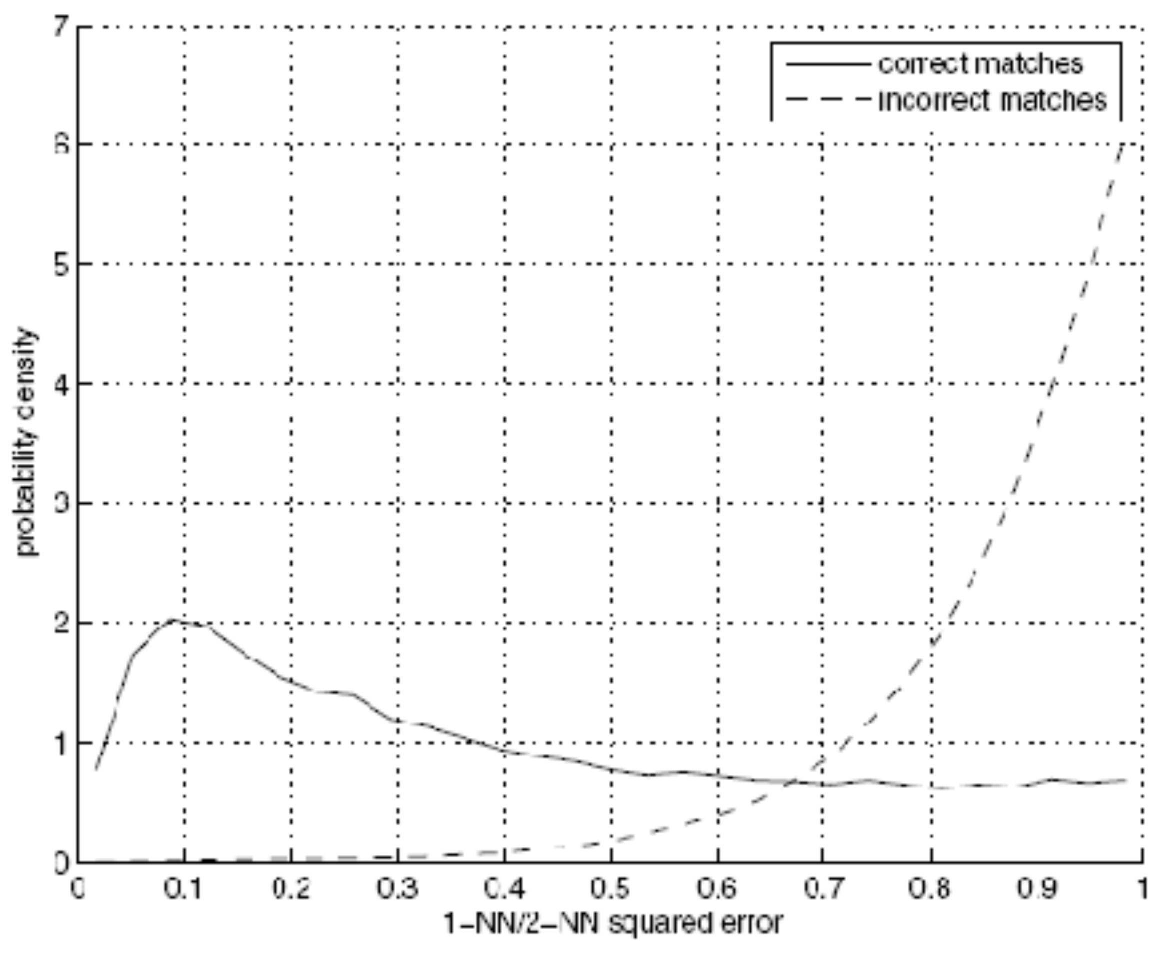
(a) Feature-space outlier rejection

- Let's not match all features, but only these that have "similar enough" matches?
- How can we do it?
  - dist(patch1,patch2) < threshold
  - How to set threshold?
    Not so easy.



Feature-space outlier rejection

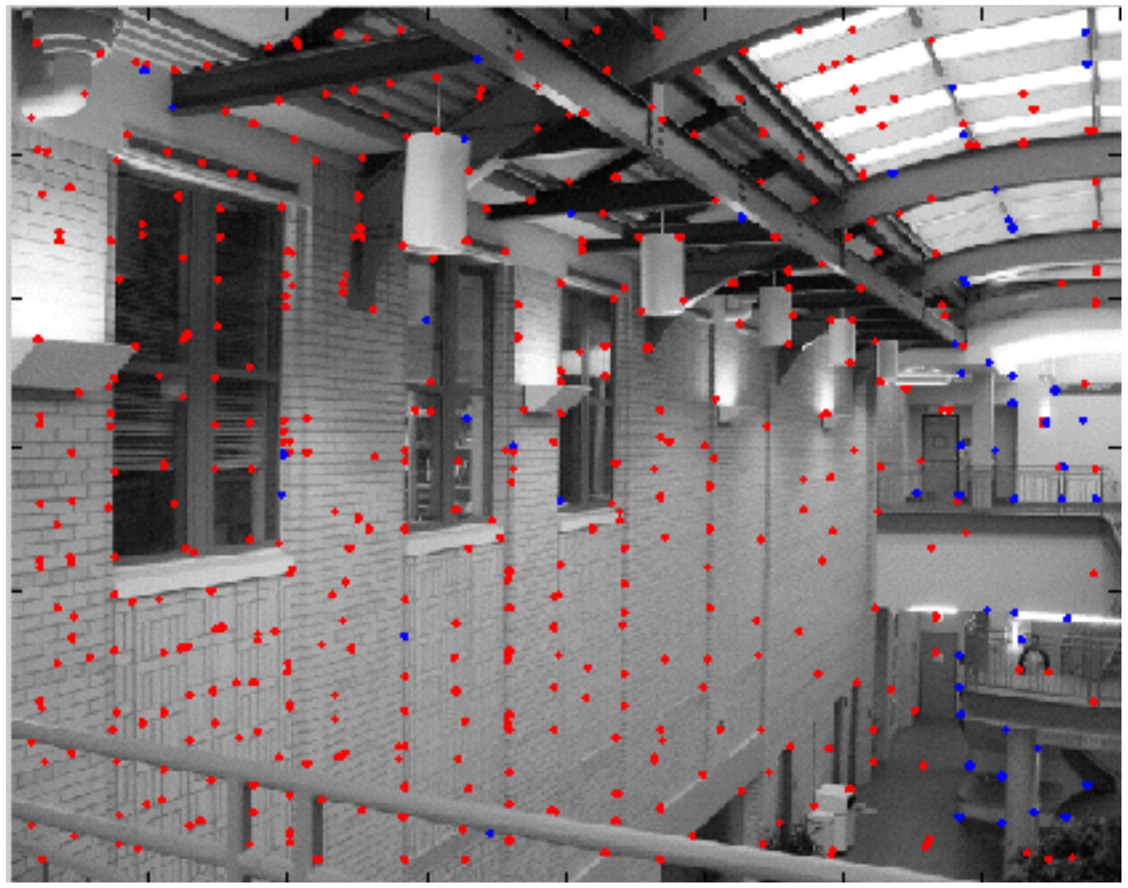
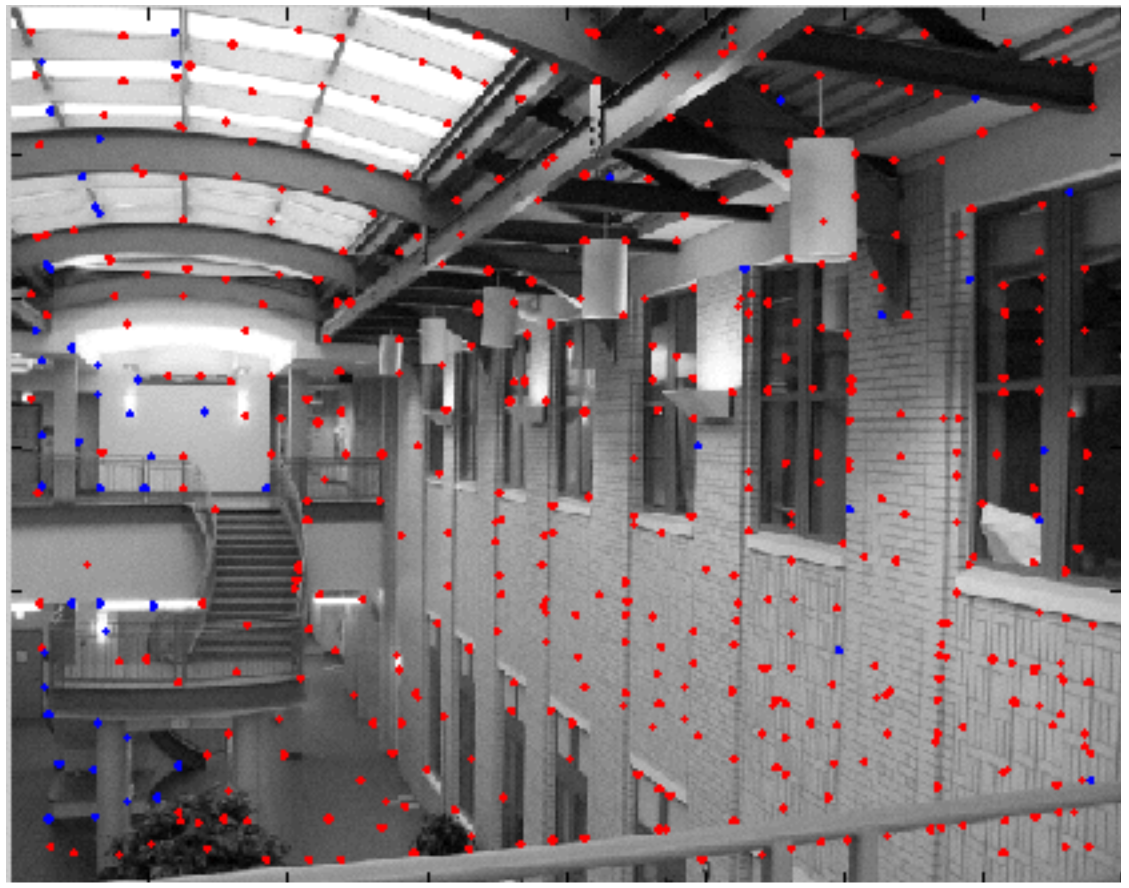
- A better way [Lowe, 1999]:
  - 1-NN: SSD of the closest match
  - 2-NN: SSD of the <u>second-closest</u> match
  - Look at how much better 1-NN is than 2-NN, e.g. 1-NN/2-NN
  - That is, is our best match so much better than the rest?



Feature matching

- Exhaustive search
  - for each feature in one image, look at **all** the other features in the other image(s)
  - Usually not so bad
- Hashing
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
  - *k*-trees and their variants (Best Bin First)

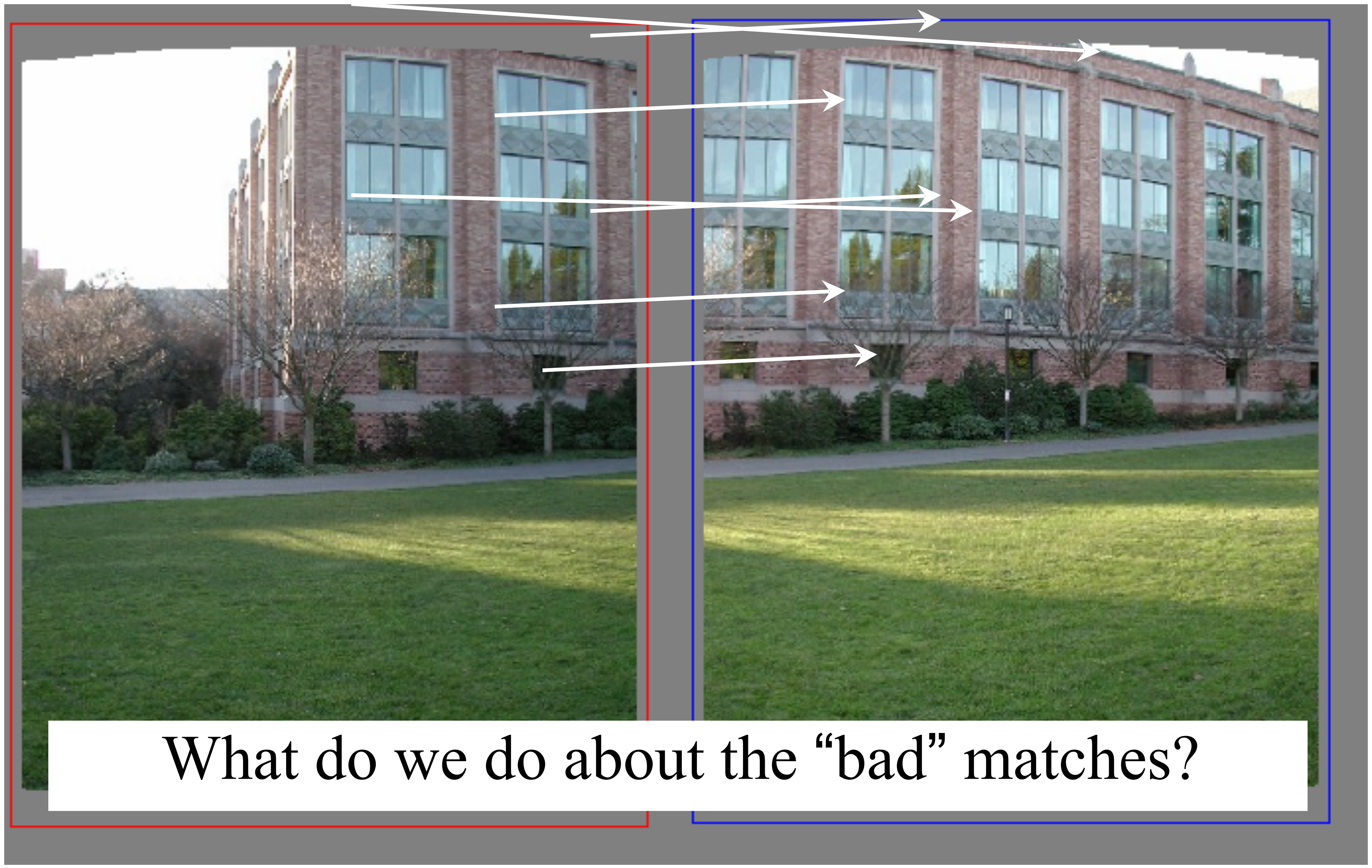
Feature-space outlier rejection

- Can we now compute H from the blue points?
  - No! Still too many outliers…
  - What can we do?

(b) Matching many features — looking for a good homography

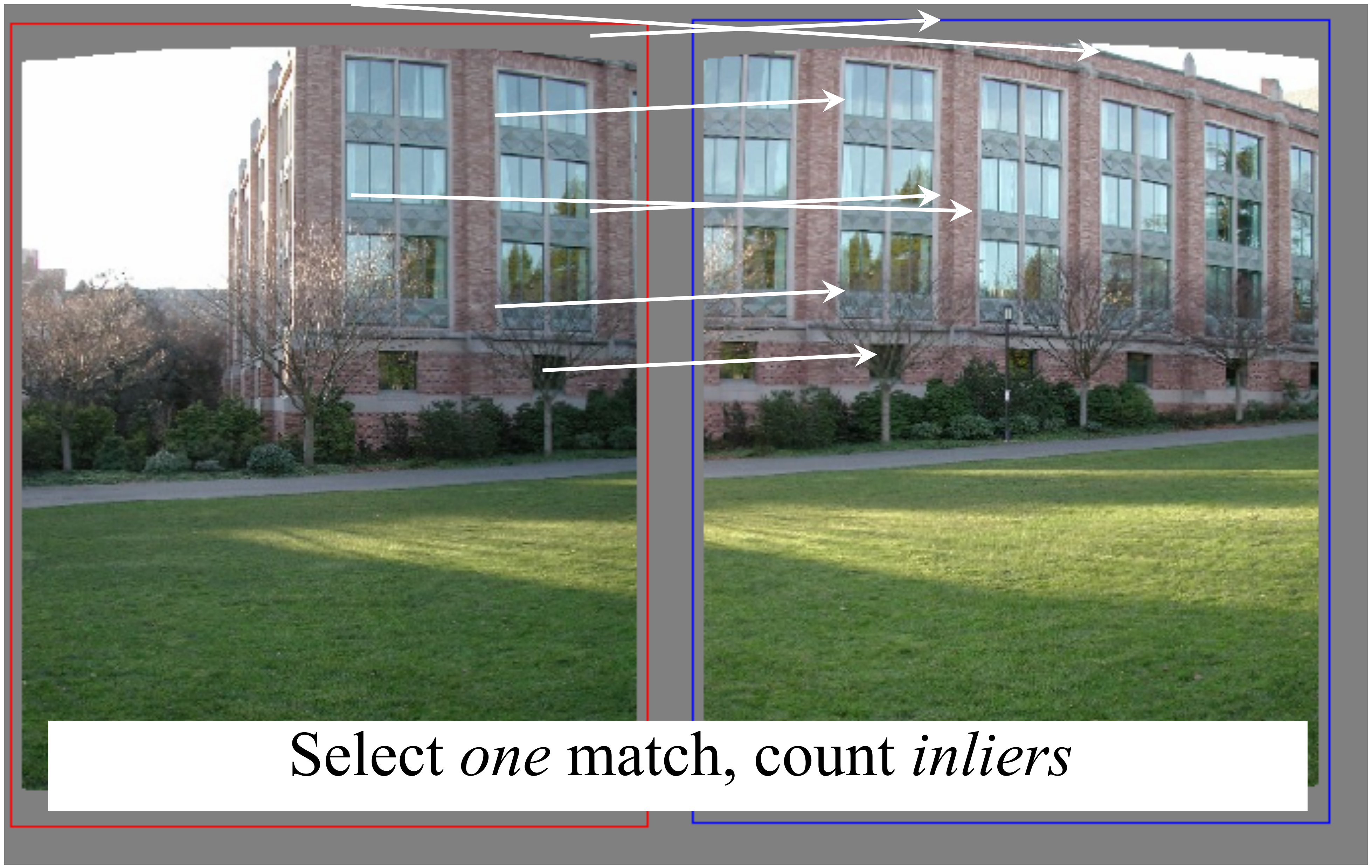
Simplified illustration with translation instead of homography



What do we do about the “bad” matches?

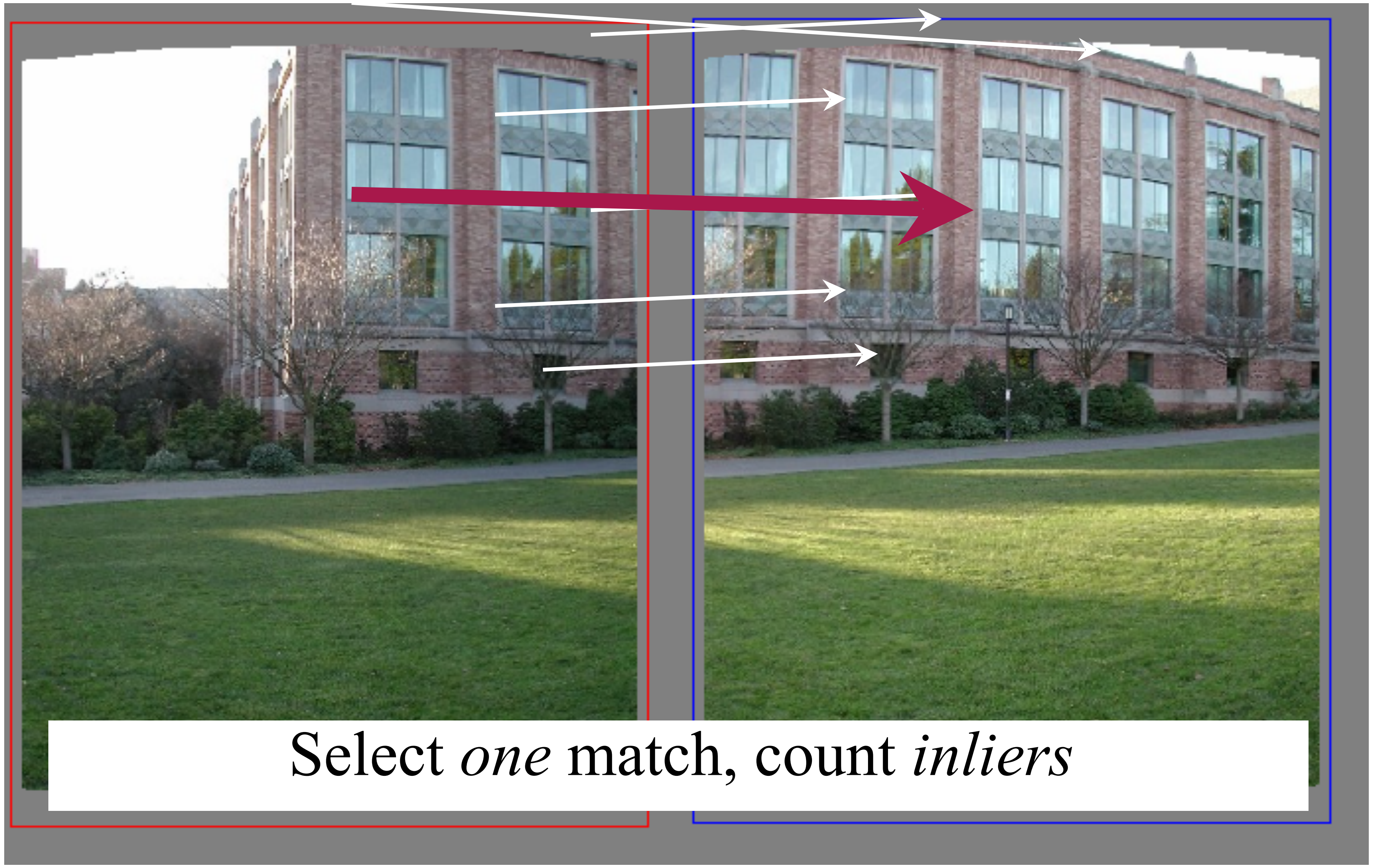
Note: at this point we don’t know which ones are good/bad

RAndom SAmple Consensus



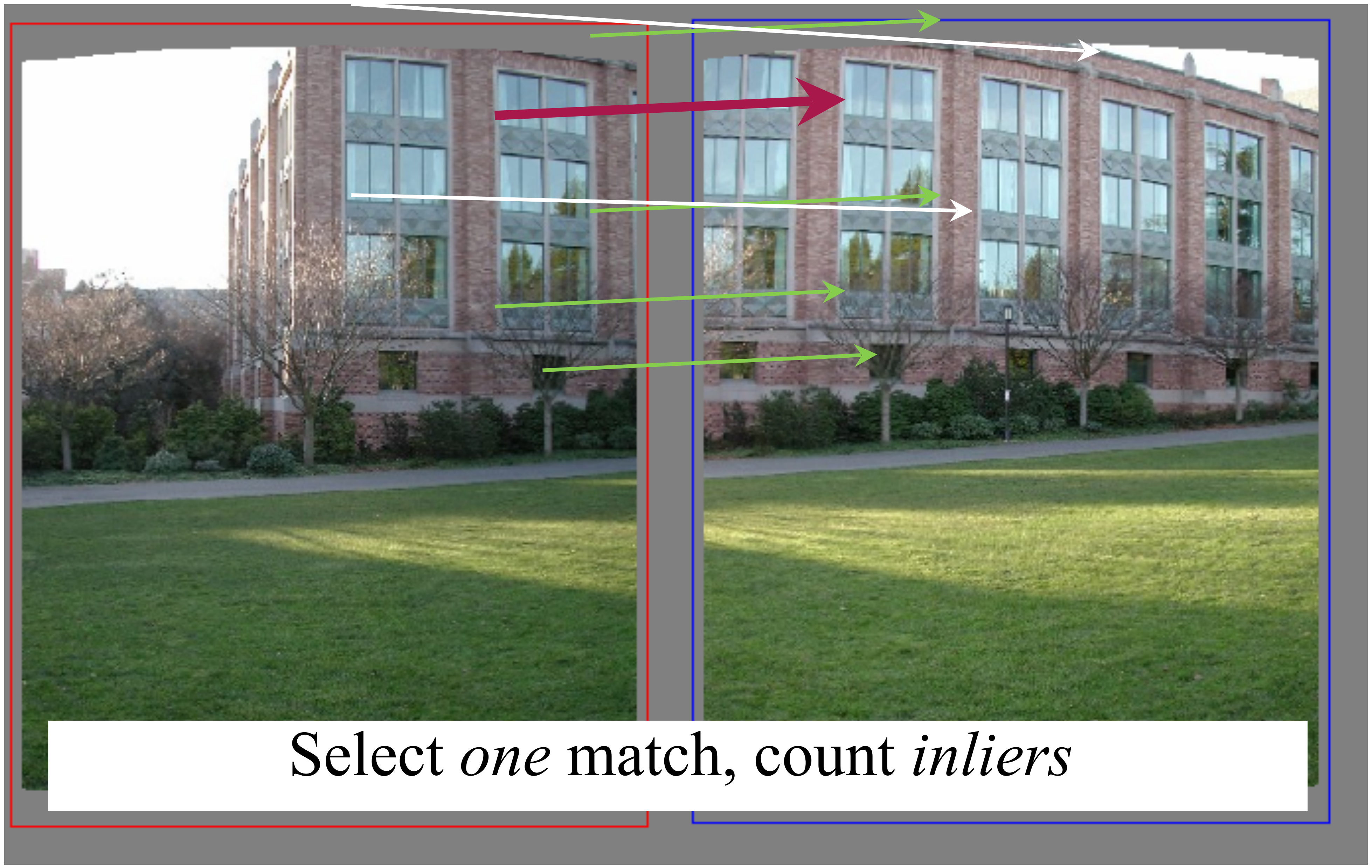
Select *one* match, count *inliers*

RAndom SAmple Consensus



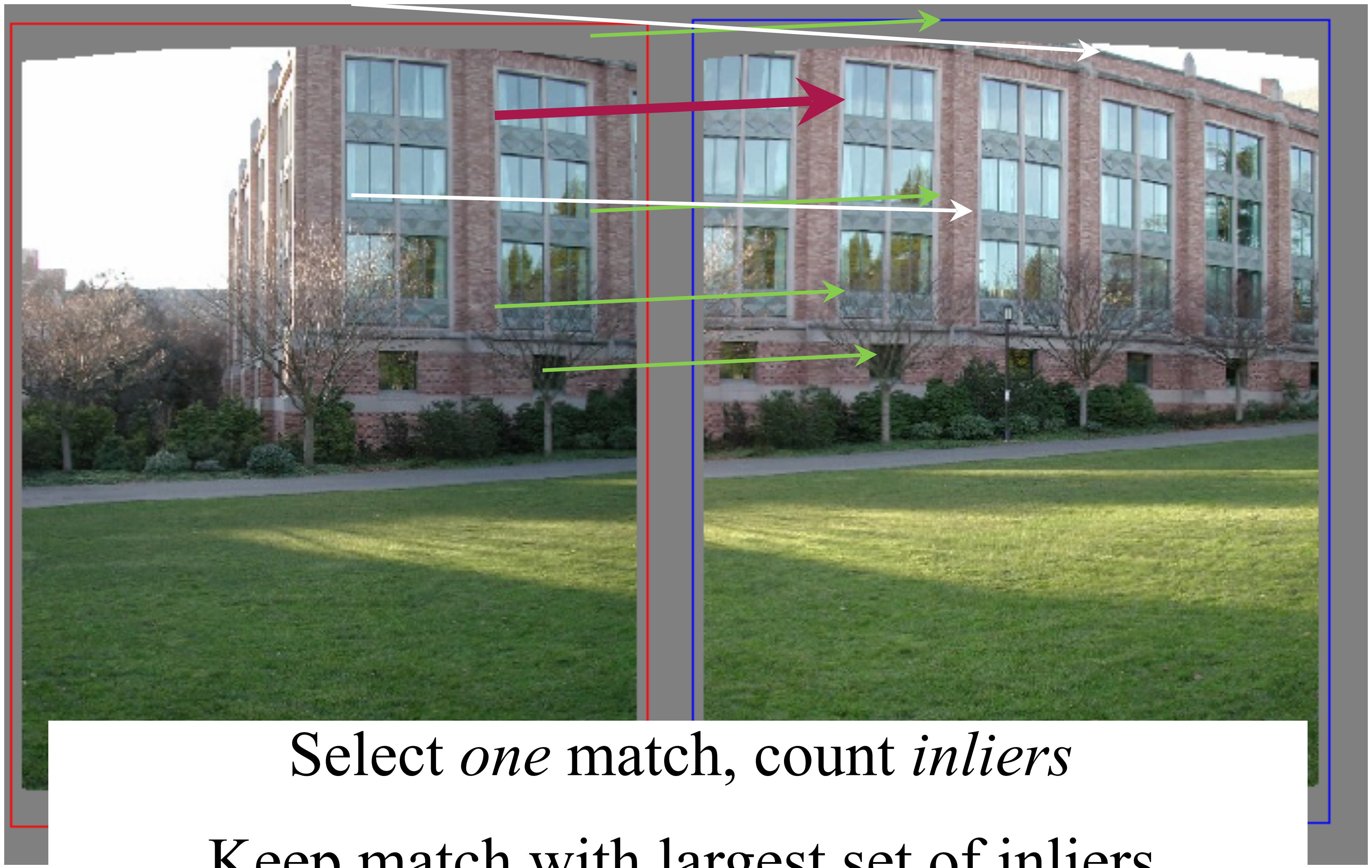
0 inliers

RAndom SAmple Consensus


Select *one* match, count *inliers*

4 inliers

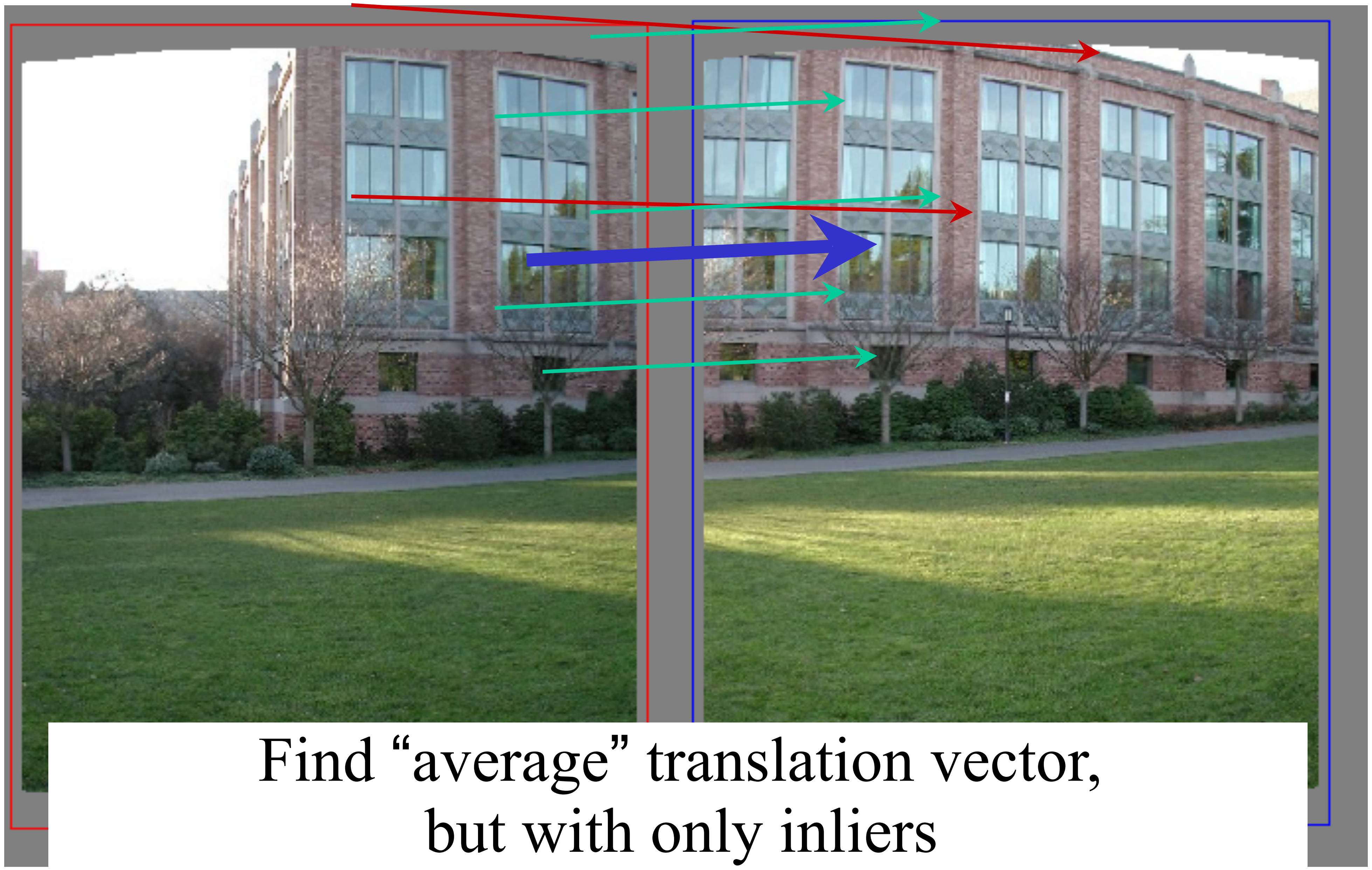
RAndom SAmple Consensus



Select *one* match, count *inliers*

Keep match with largest set of inliers

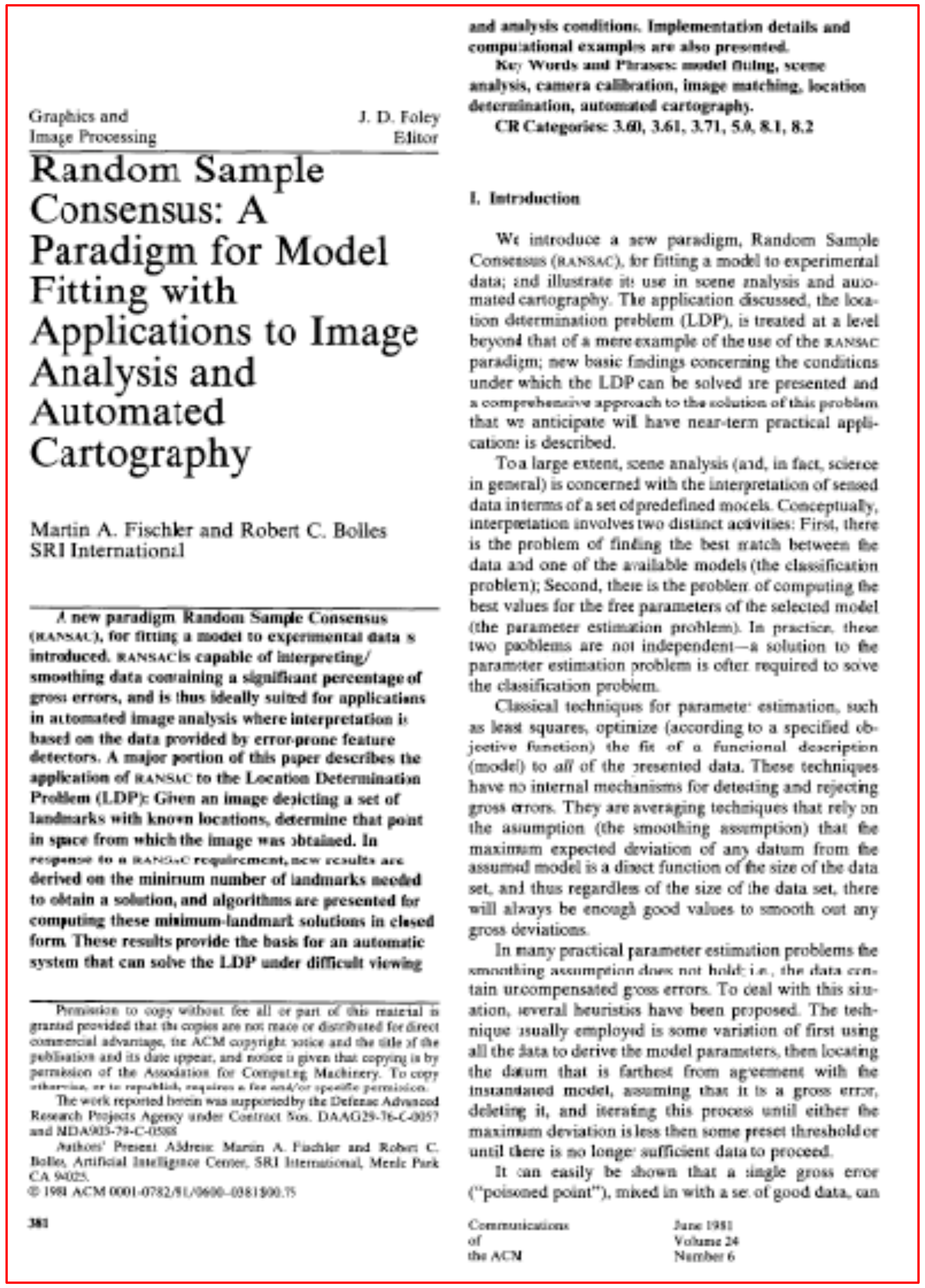
At the end: Least squares fit



Find “average” translation vector, but with only inliers

Reference

- M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.
- http://portal.acm.org/citation.cfm?id=358692



Graphics and Image Processing — J. D. Foley, Editor

Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography

Martin A. Fischler and Robert C. Bolles
SRI International

A new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data is introduced. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is thus ideally suited for applications in automated image analysis where interpretation is based on the data provided by error-prone feature detectors. A major portion of this paper describes the application of RANSAC to the Location Determination Problem (LDP): Given an image depicting a set of landmarks with known locations, determine that point in space from which the image was obtained. In response to a RANSAC requirement, new results are derived on the minimum number of landmarks needed to obtain a solution, and algorithms are presented for computing these minimum-landmark solutions in closed form. These results provide the basis for an automatic system that can solve the LDP under difficult viewing and analysis conditions. Implementation details and computational examples are also presented.

Key Words and Phrases: model fitting, scene analysis, camera calibration, image matching, location determination, automated cartography.
CR Categories: 3.60, 3.61, 3.71, 5.0, 8.1, 8.2

The work reported herein was supported by the Defense Advanced Research Projects Agency under Contract Nos. DAAG29-76-C-0057 and MDA903-79-C-0588.

Authors' Present Address: Martin A. Fischler and Robert C. Bolles, Artificial Intelligence Center, SRI International, Menlo Park CA 94025.

I. Introduction

We introduce a new paradigm, Random Sample Consensus (RANSAC), for fitting a model to experimental data; and illustrate its use in scene analysis and automated cartography. The application discussed, the location determination problem (LDP), is treated at a level beyond that of a mere example of the use of the RANSAC paradigm; new basic findings concerning the conditions under which the LDP can be solved are presented and a comprehensive approach to the solution of this problem that we anticipate will have near-term practical applications is described.

To a large extent, scene analysis (and, in fact, science in general) is concerned with the interpretation of sensed data in terms of a set of predefined models. Conceptually, interpretation involves two distinct activities: First, there is the problem of finding the best match between the data and one of the available models (the classification problem); Second, there is the problem of computing the best values for the free parameters of the selected model (the parameter estimation problem). In practice, these two problems are not independent—a solution to the parameter estimation problem is often required to solve the classification problem.

Classical techniques for parameter estimation, such as least squares, optimize (according to a specified objective function) the fit of a functional description (model) to *all* of the presented data. These techniques have no internal mechanisms for detecting and rejecting gross errors. They are averaging techniques that rely on the assumption (the smoothing assumption) that the maximum expected deviation of any datum from the assumed model is a direct function of the size of the data set, and thus regardless of the size of the data set, there will always be enough good values to smooth out any gross deviations.

In many practical parameter estimation problems the smoothing assumption does not hold; i.e., the data contain uncompensated gross errors. To deal with this situation, several heuristics have been proposed. The technique usually employed is some variation of first using all the data to derive the model parameters, then locating the datum that is farthest from agreement with the instantiated model, assuming that it is a gross error, deleting it, and iterating this process until either the maximum deviation is less then some preset threshold or until there is no longer sufficient data to proceed.

It can easily be shown that a single gross error ("poisoned point"), mixed in with a set of good data, can

RANSAC for Homography

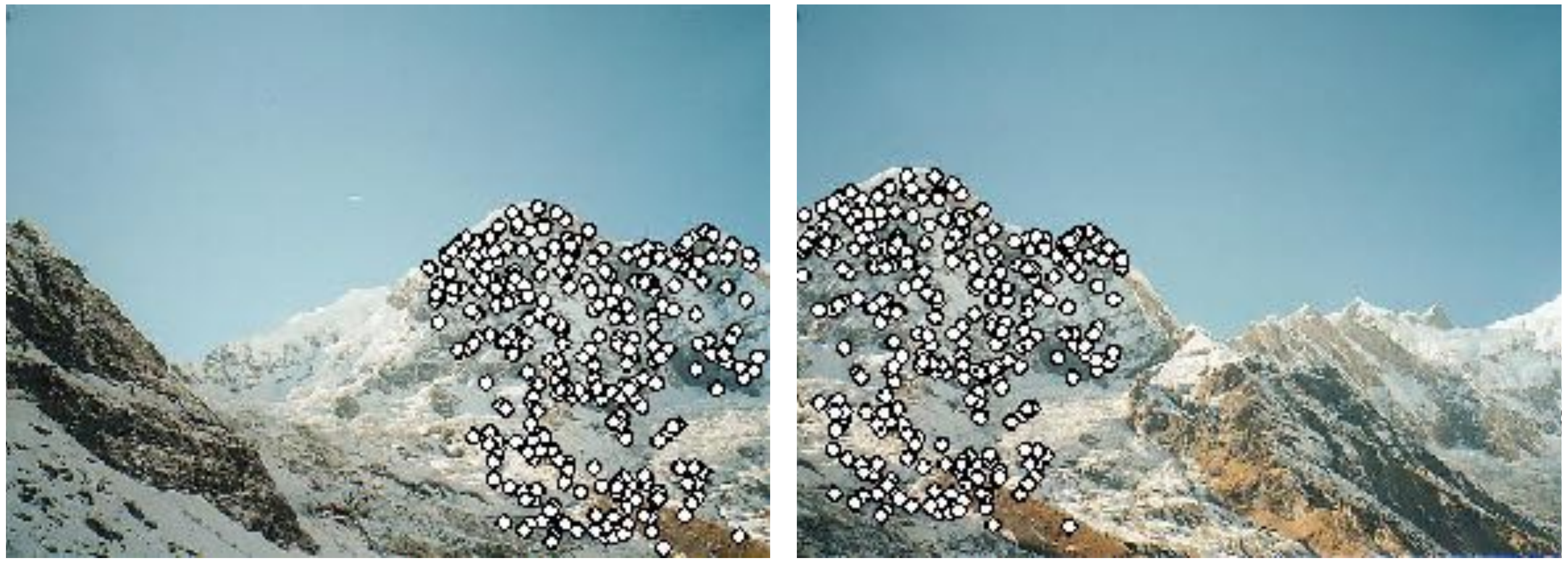
Repeat N times:

- Select a random set of feature matches (4 pairs)
- Fit an homography
- Compute inliers: apply the transformation to all the features and compute the error (distance between matching points after the transformation), $||p_i', \boldsymbol{H}\, p_i|| < \varepsilon$

  Count number of inliers

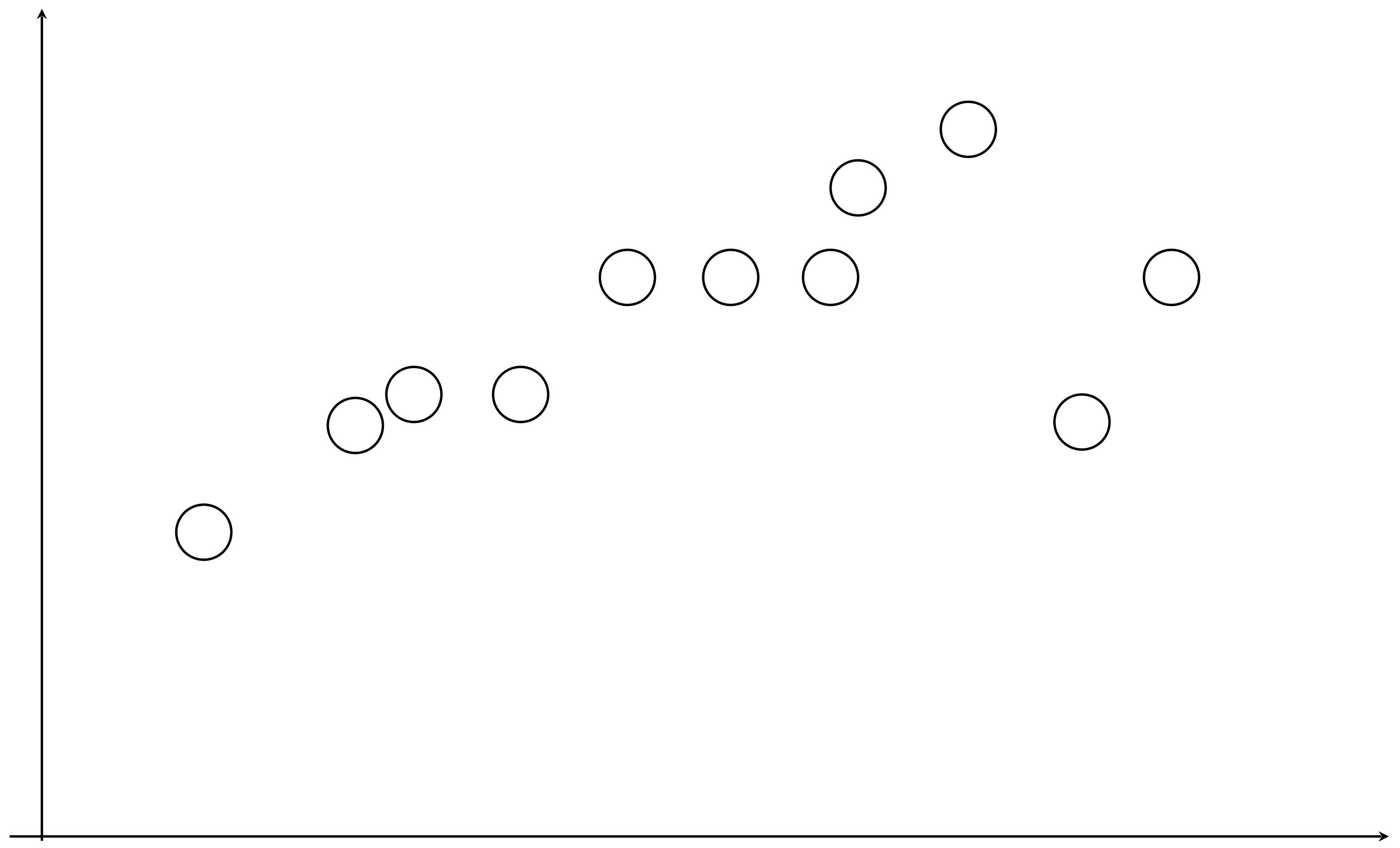
Select homography with largest number of inliers,

Re-compute least-squares H estimate using all of the inliers



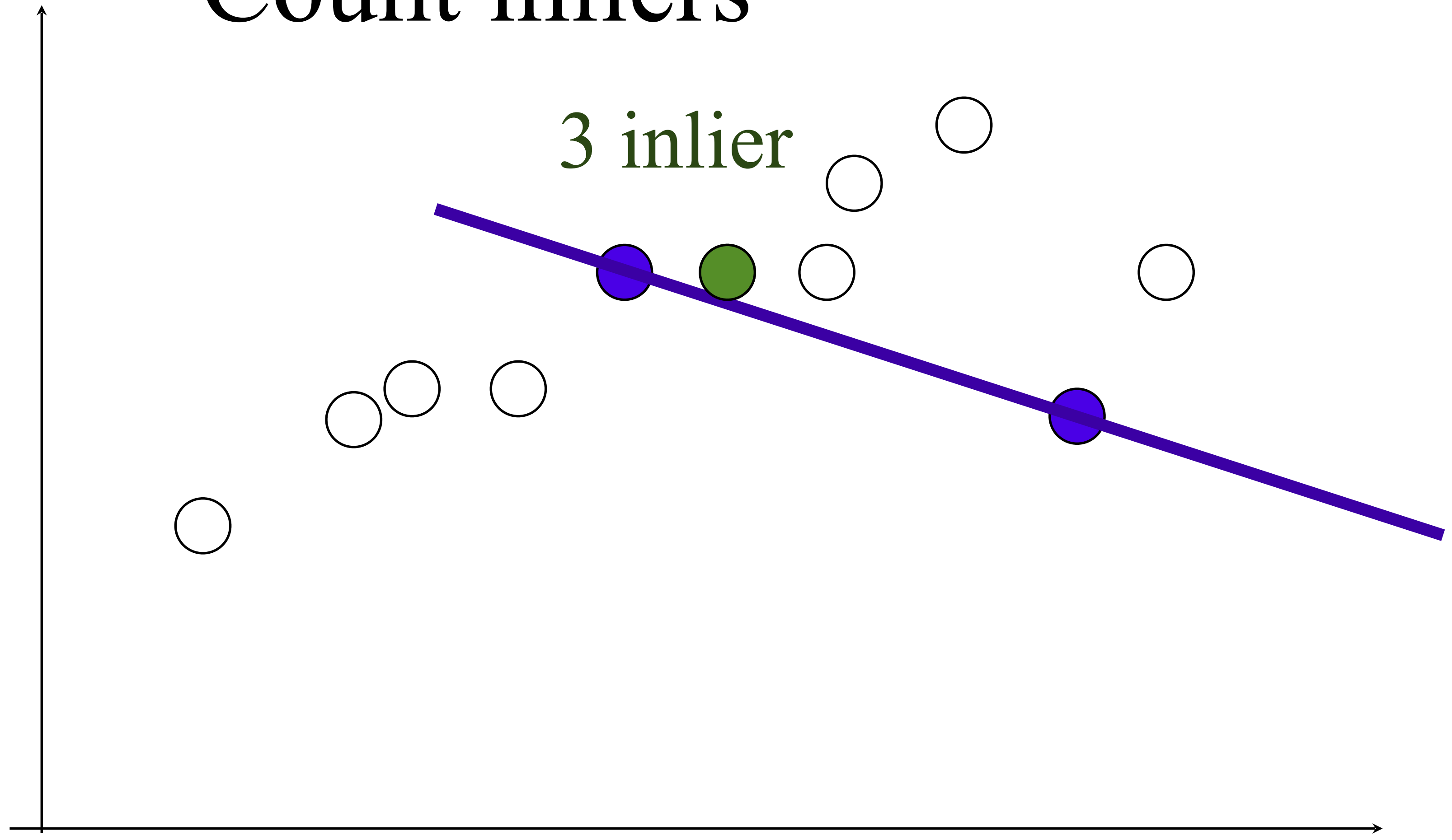
Simple example: fit a line

- Rather than homography H (8 numbers) fit y=ax+b (2 numbers a, b) to 2D pairs



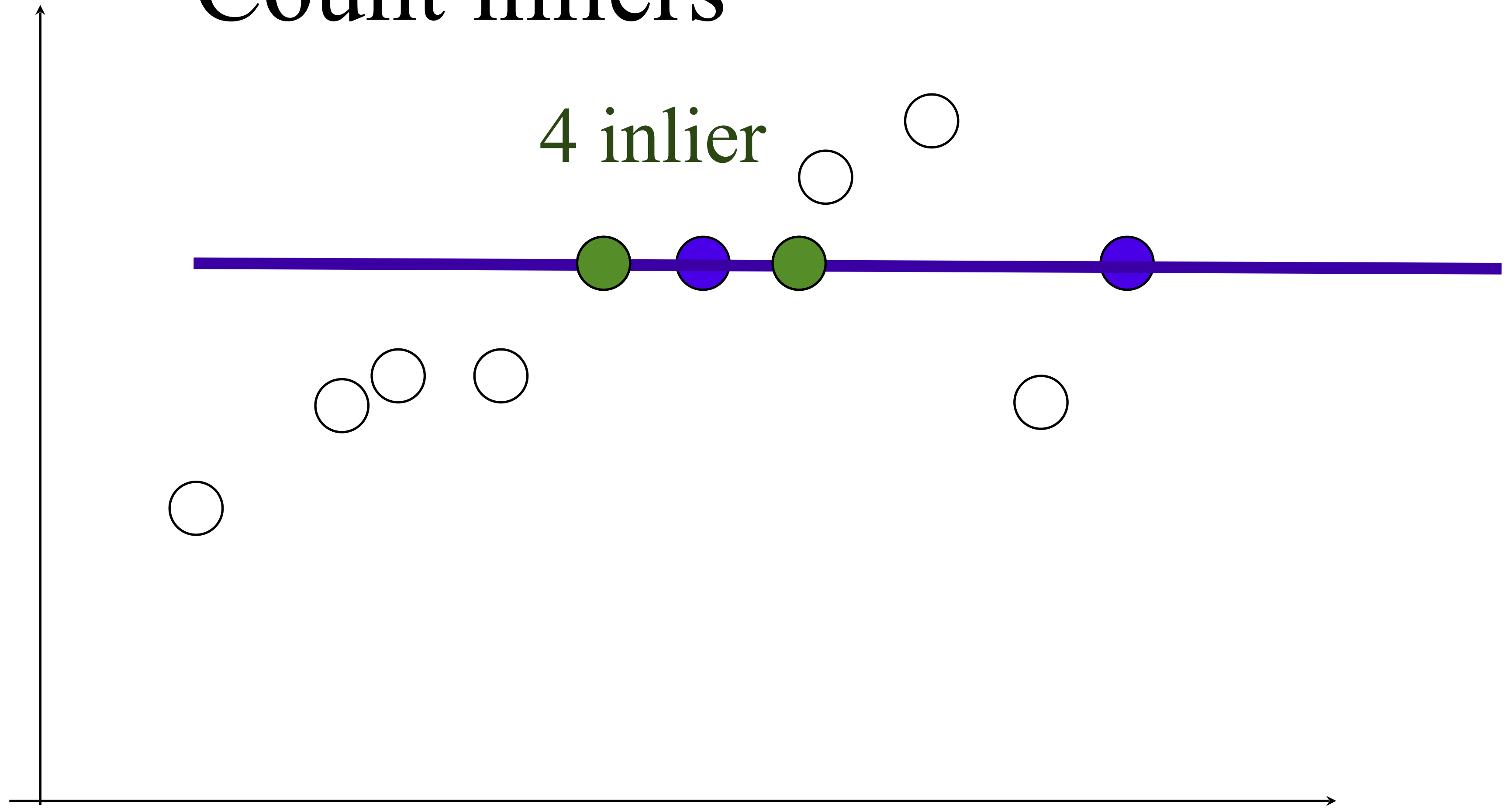
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



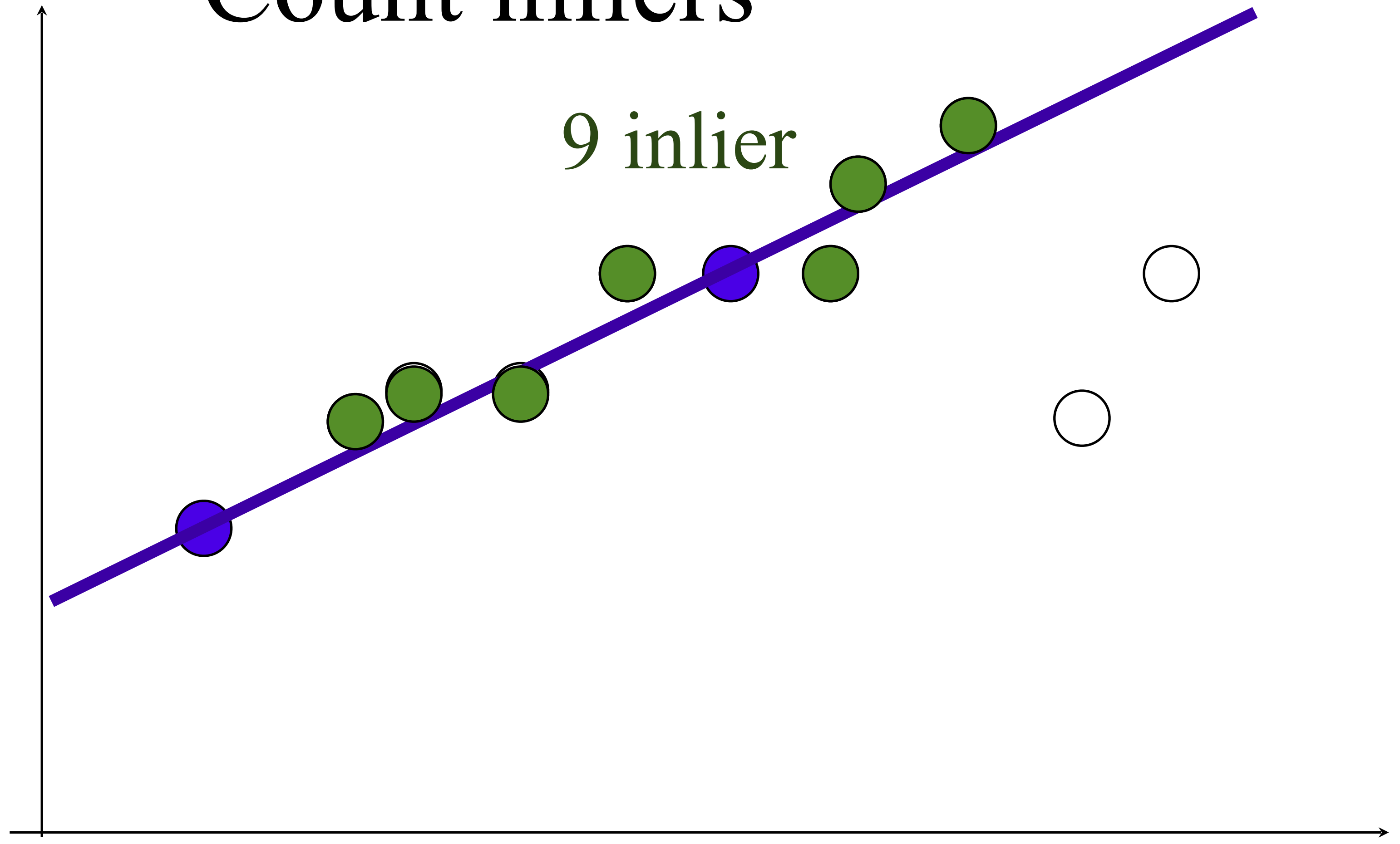
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



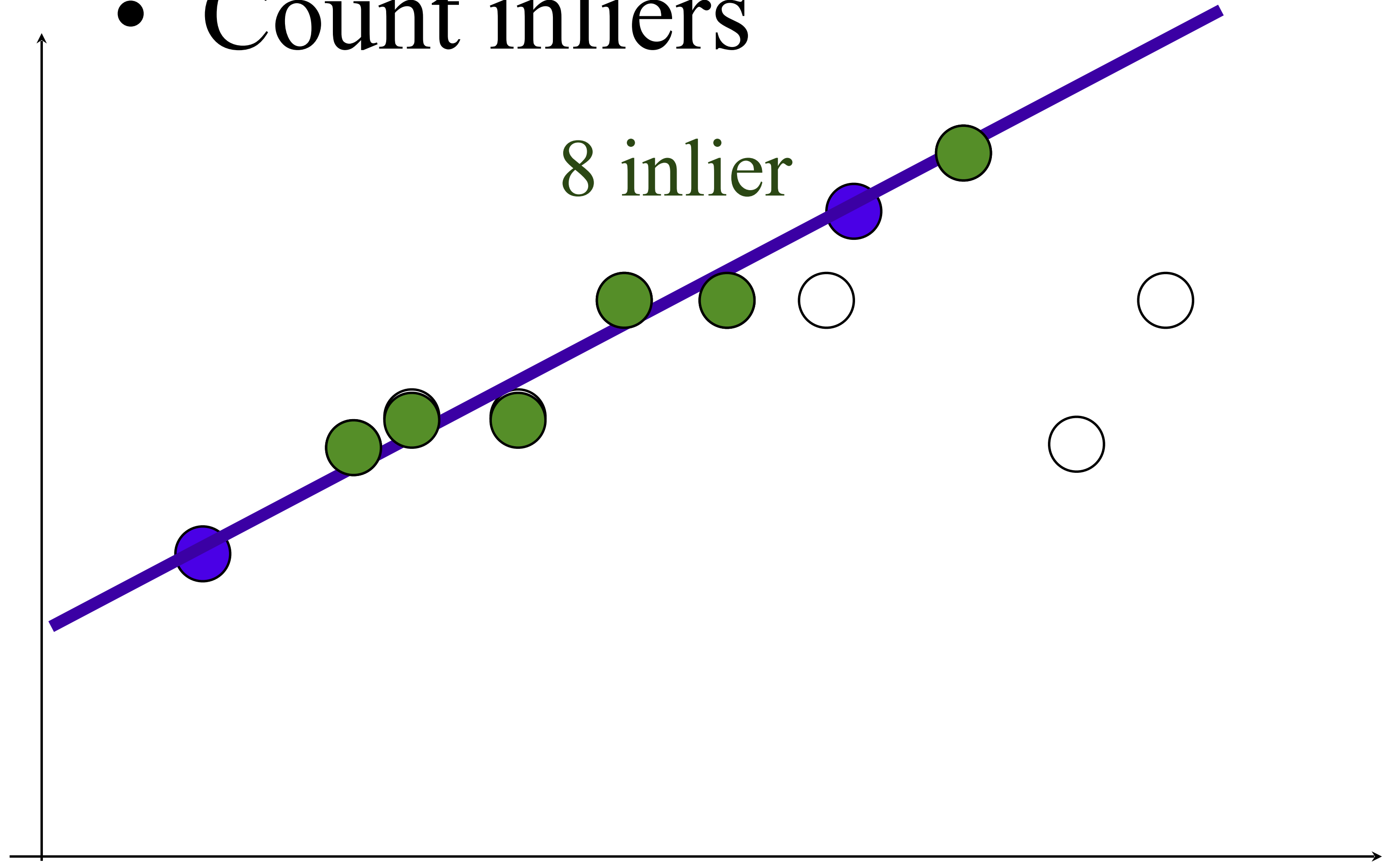
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers



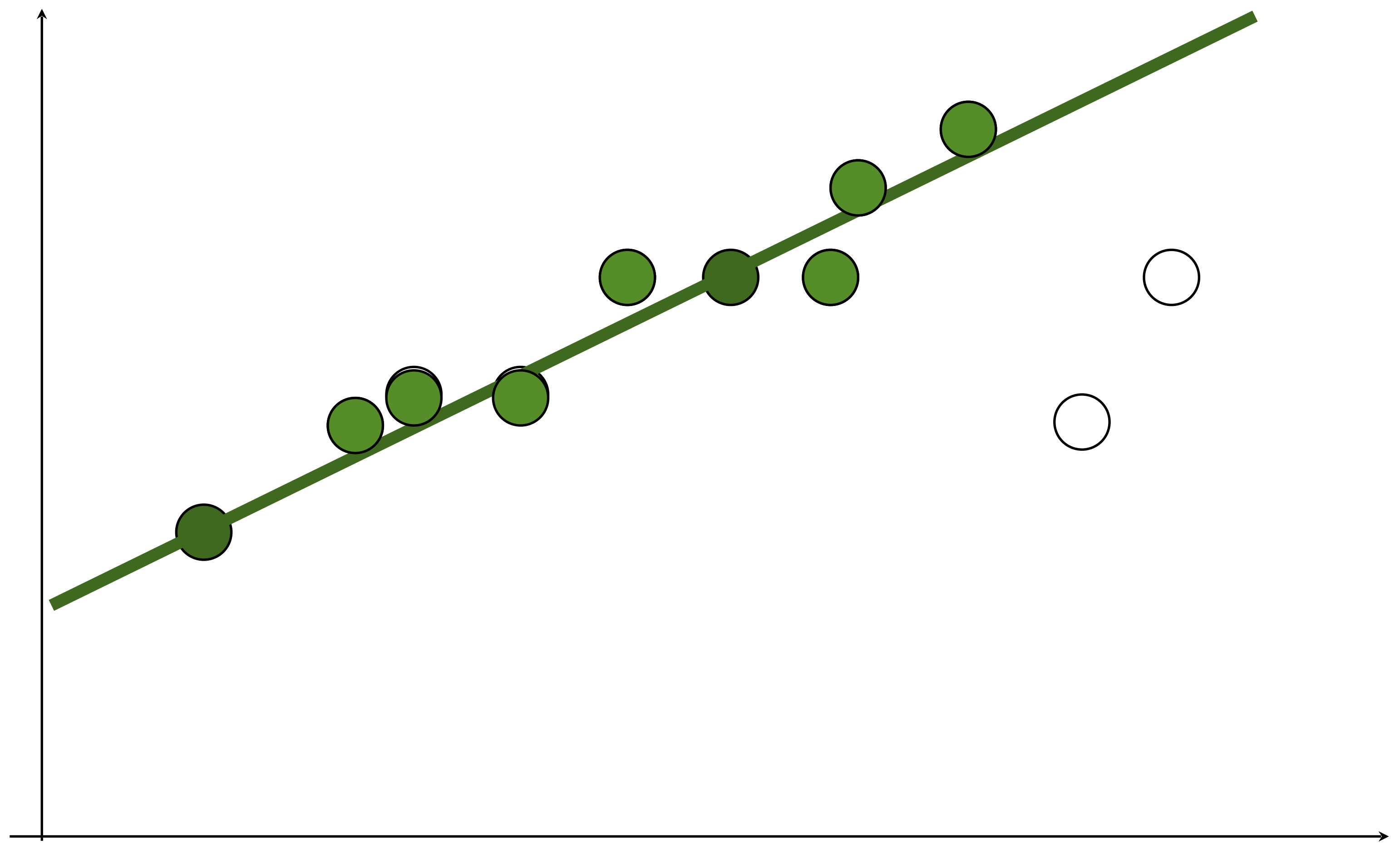
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

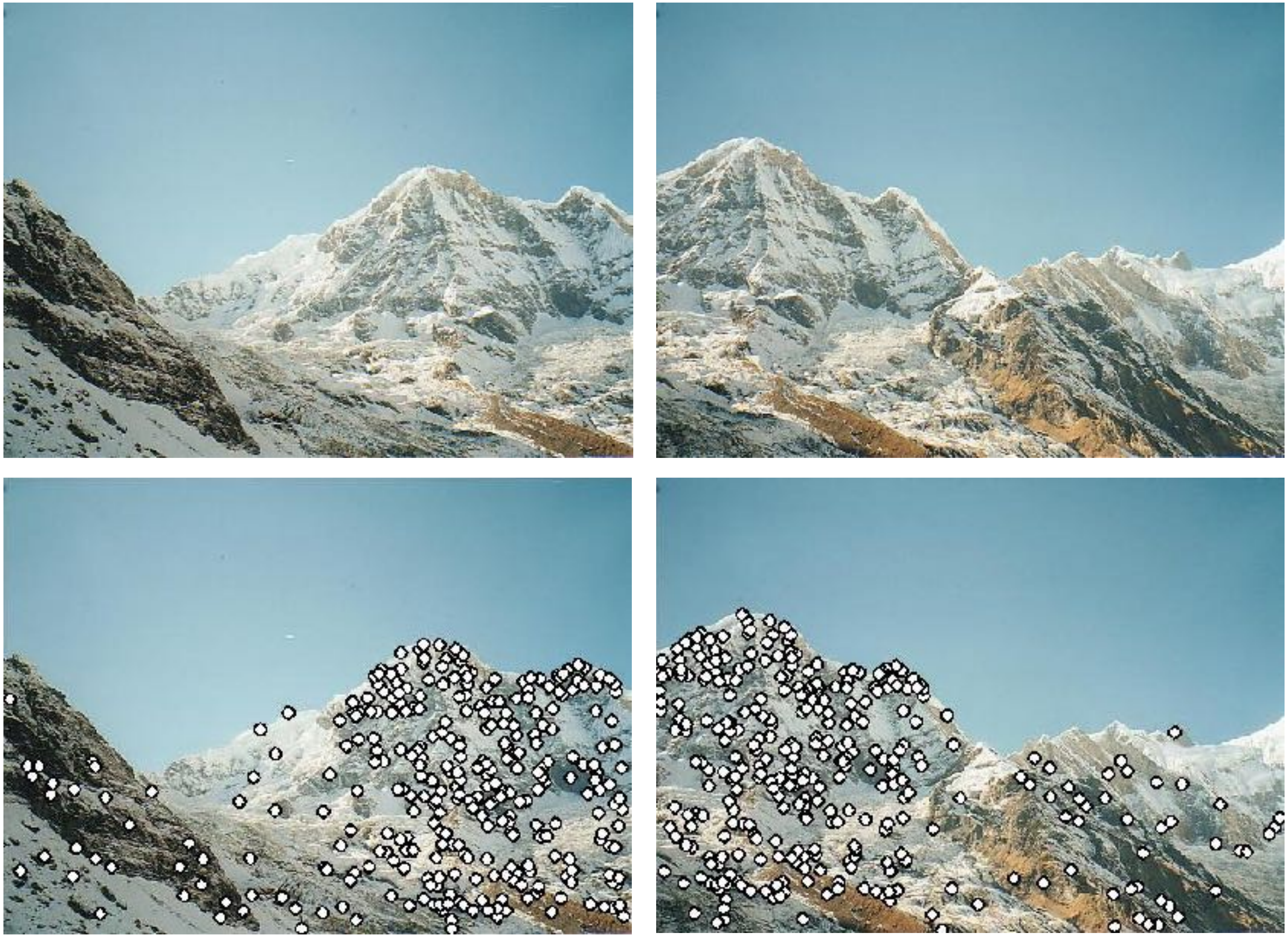


Simple example: fit a line

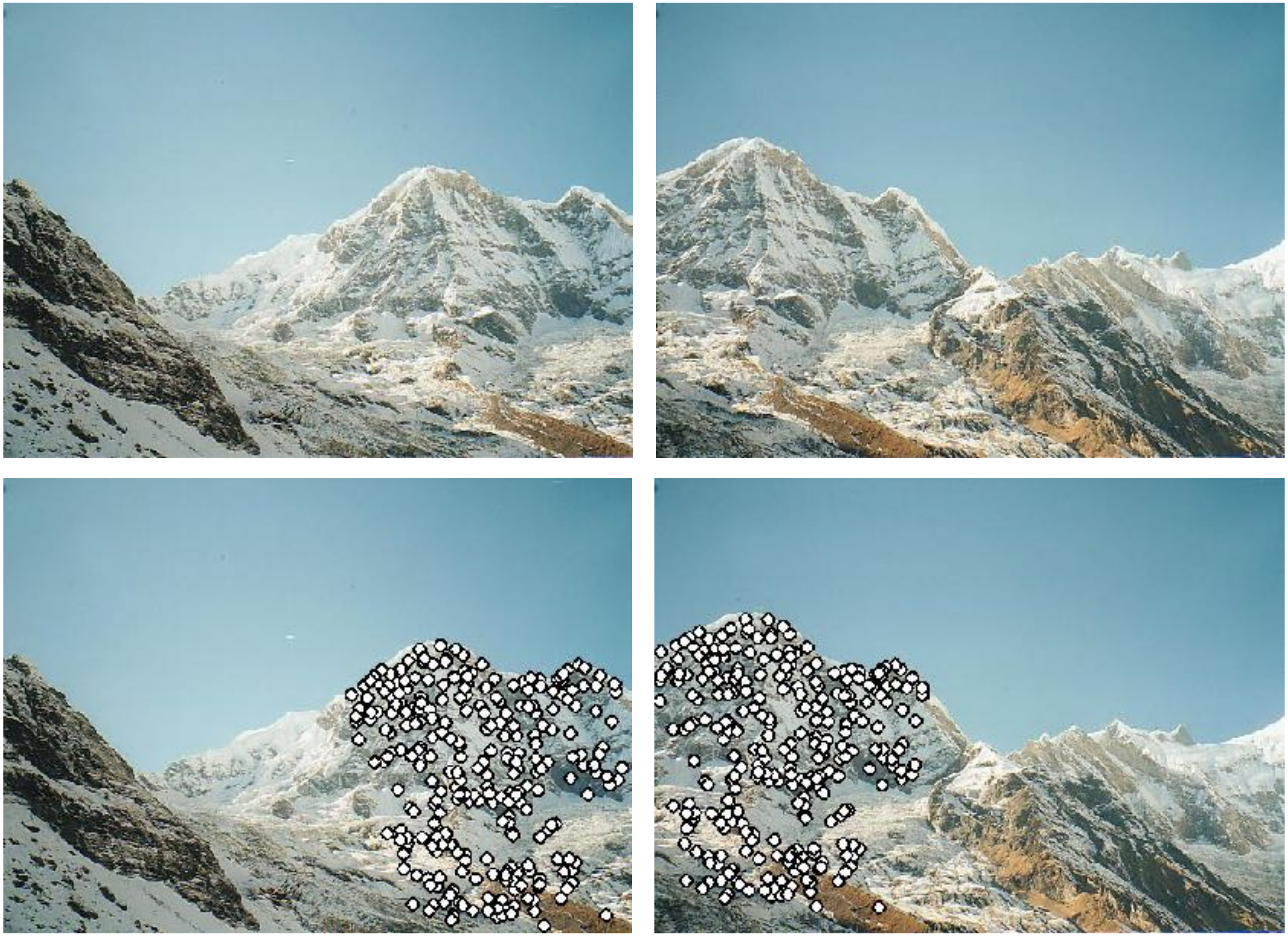
- Use biggest set of inliers
- Do least-square fit



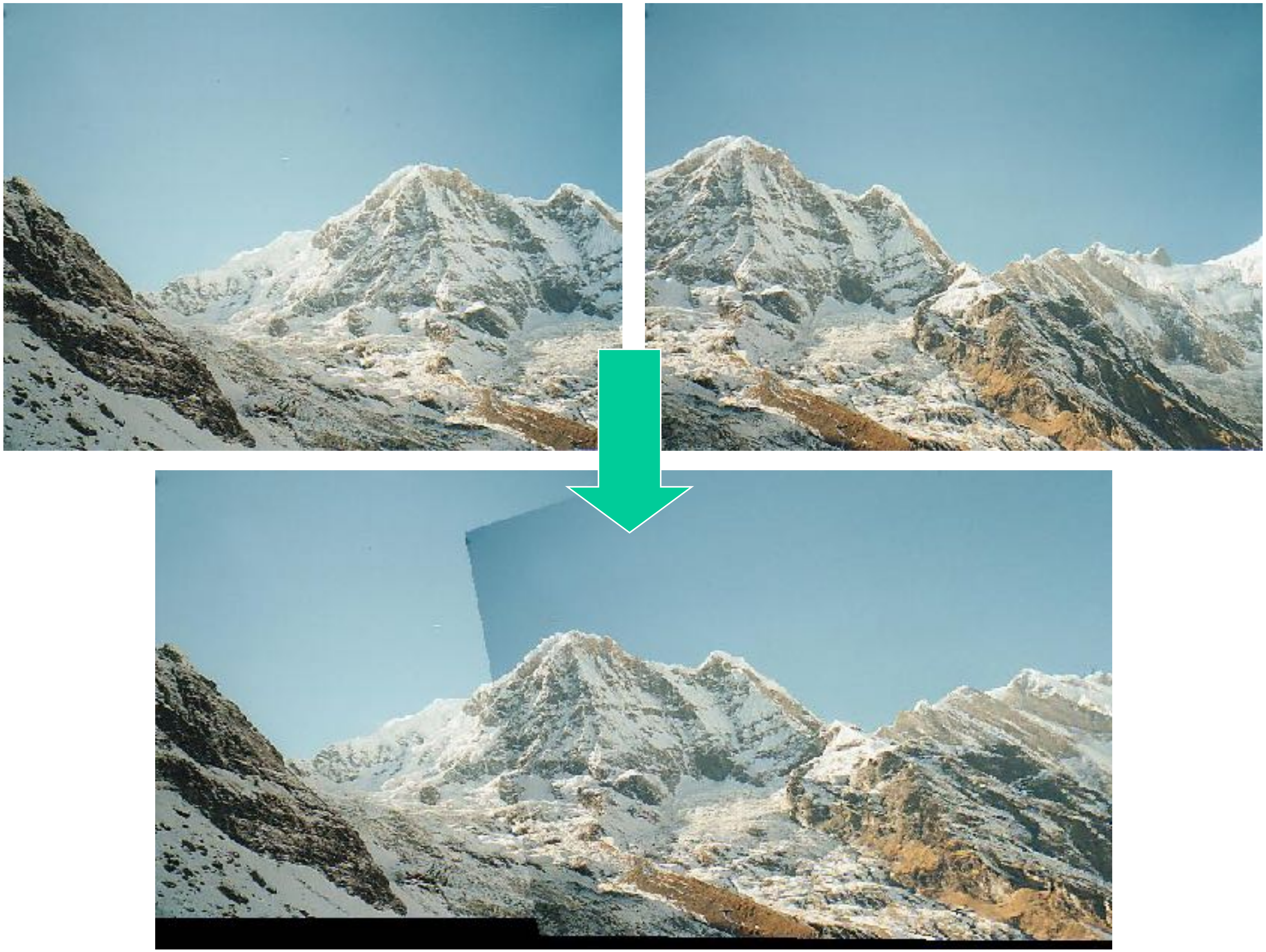
RANSAC for Homography



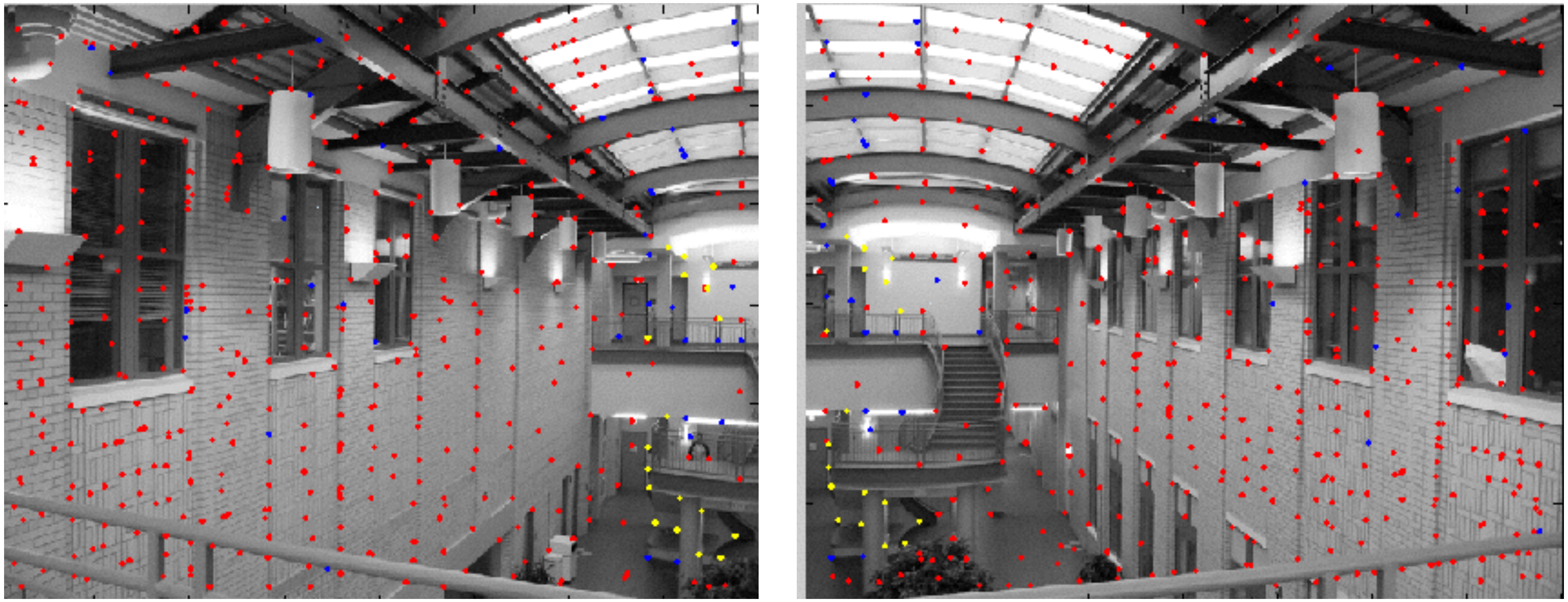
RANSAC for Homography



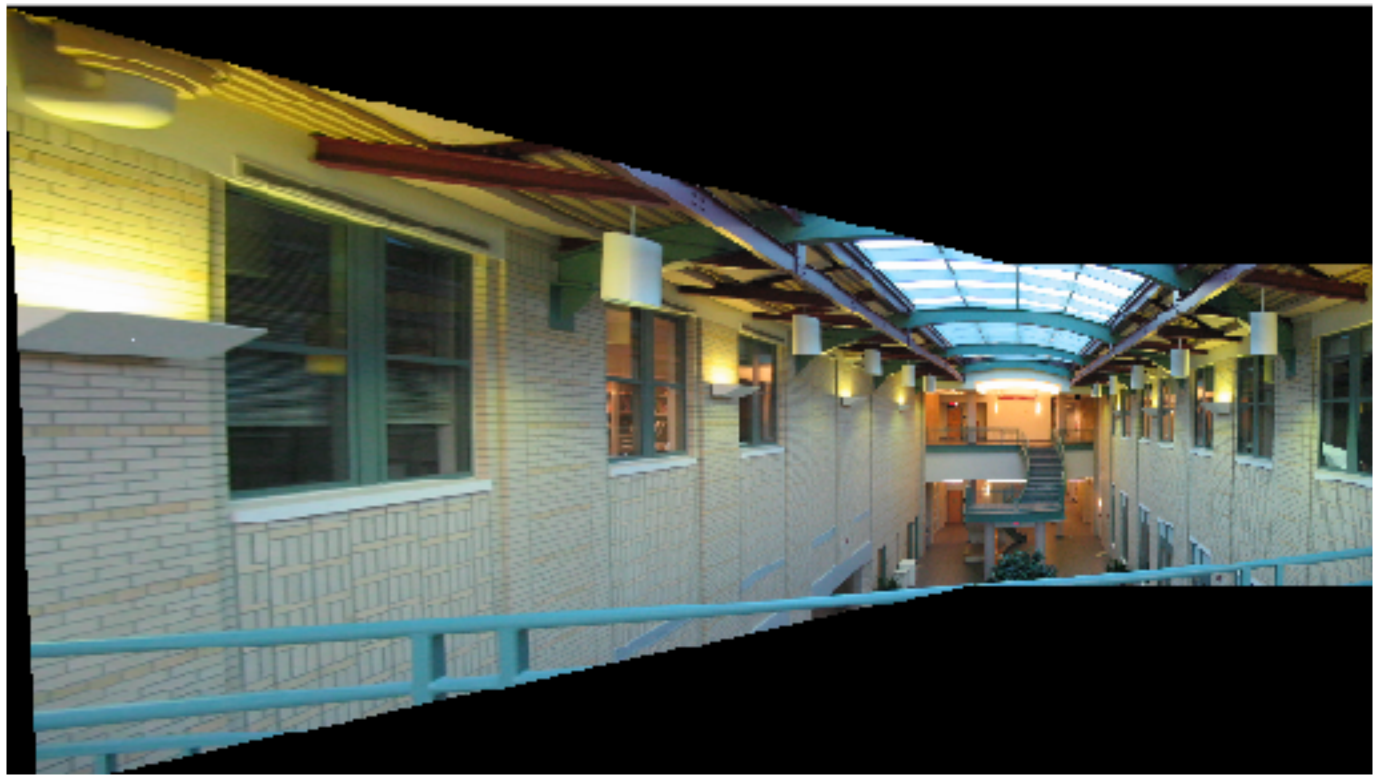
RANSAC for Homography



RANSAC



Red: Rejected by 2nd nearest neighbor criterion
Blue: RANSAC outliers
Yellow: RANSAC inliers



Robustness

- Proportion of inliers in our pairs is G (for “good”)
- Our model needs P pairs

  P=4 for homography
- Probability that we pick P inliers?

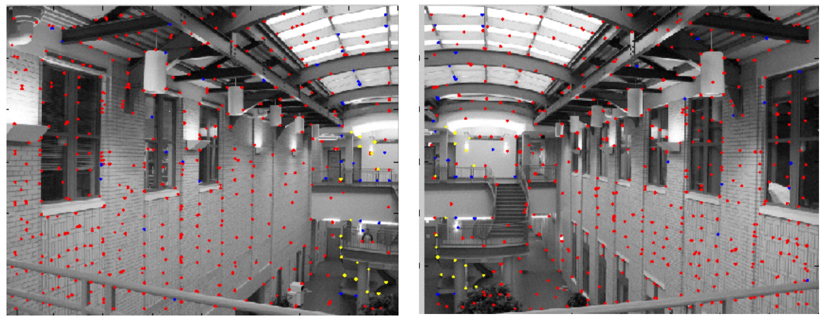
  G^P
- Probability that after N RANSAC iterations we have **not** picked a set of inliers?

  $(1-G^P)^N$

Robustness: example

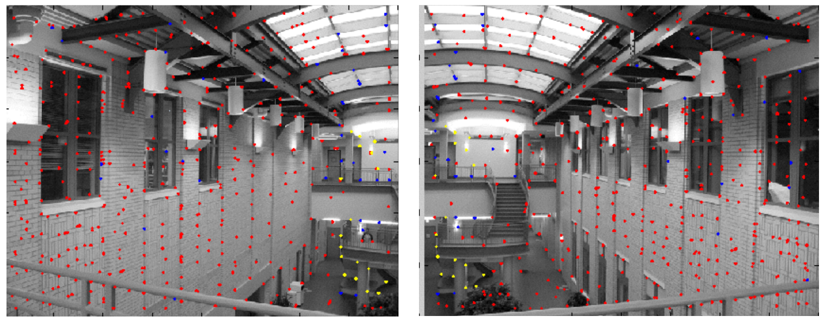
- Proportion of inliers **G=0.5**
- Probability that we pick P=4 inliers?
  - 0.5^4**=**0.0625 (6% chance)
- Probability that we have **not** picked a set of inliers?
  - N=100 iterations:
    $(1-0.5^4)^{100}$=0.00157 (1 chance in 600)
  - N=1000 iterations:
    1 chance in 1e28

Robustness: example



- Proportion of inliers **G=0.3**
- Probability that we pick P=4 inliers?
  - 0.3^4**=**0.0081 (0.8% chance)
- Probability that we have **not** picked a set of inliers?
  - N=100 iterations:
    $(1-0.3^4)^{100}$=0.44 (1 chance in 2)
  - N=1000 iterations:
    1 chance in 3400

Robustness: example



- Proportion of inliers **G=0.1**
- Probability that we pick P=4 inliers?
  - 0.1^4**=**0.0001 (0.01% chances, 1 in 10,000)
- Probability that we have **not** picked a set of inliers?
  - N=100 iterations: $(1-0.1^4)^{100}=0.99$
  - N=1000 iterations: 90%
  - N=10,000: 36%
  - N=100,000: 1 in 22,000

Robustness: conclusions

$(1-G^P)^N$

- Effect of number of parameters of model/number of necessary pairs
  - Bad exponential
- Effect of percentage of inliers
  - Base of the exponential
- Effect of number of iterations
  - Good exponential

Example: Recognising Panoramas

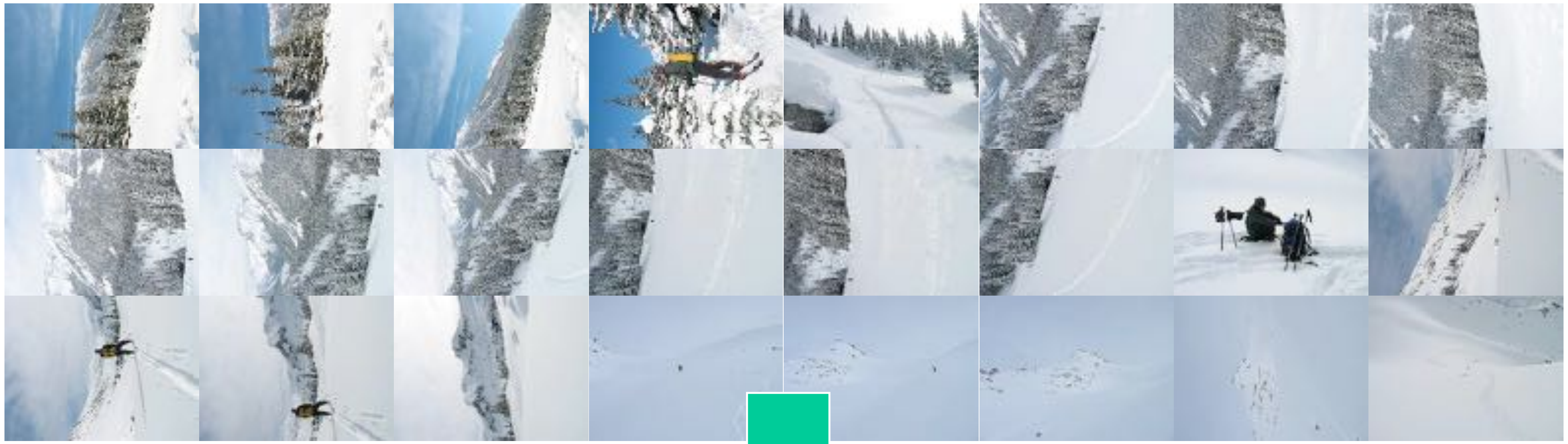
M. Brown and D. Lowe,

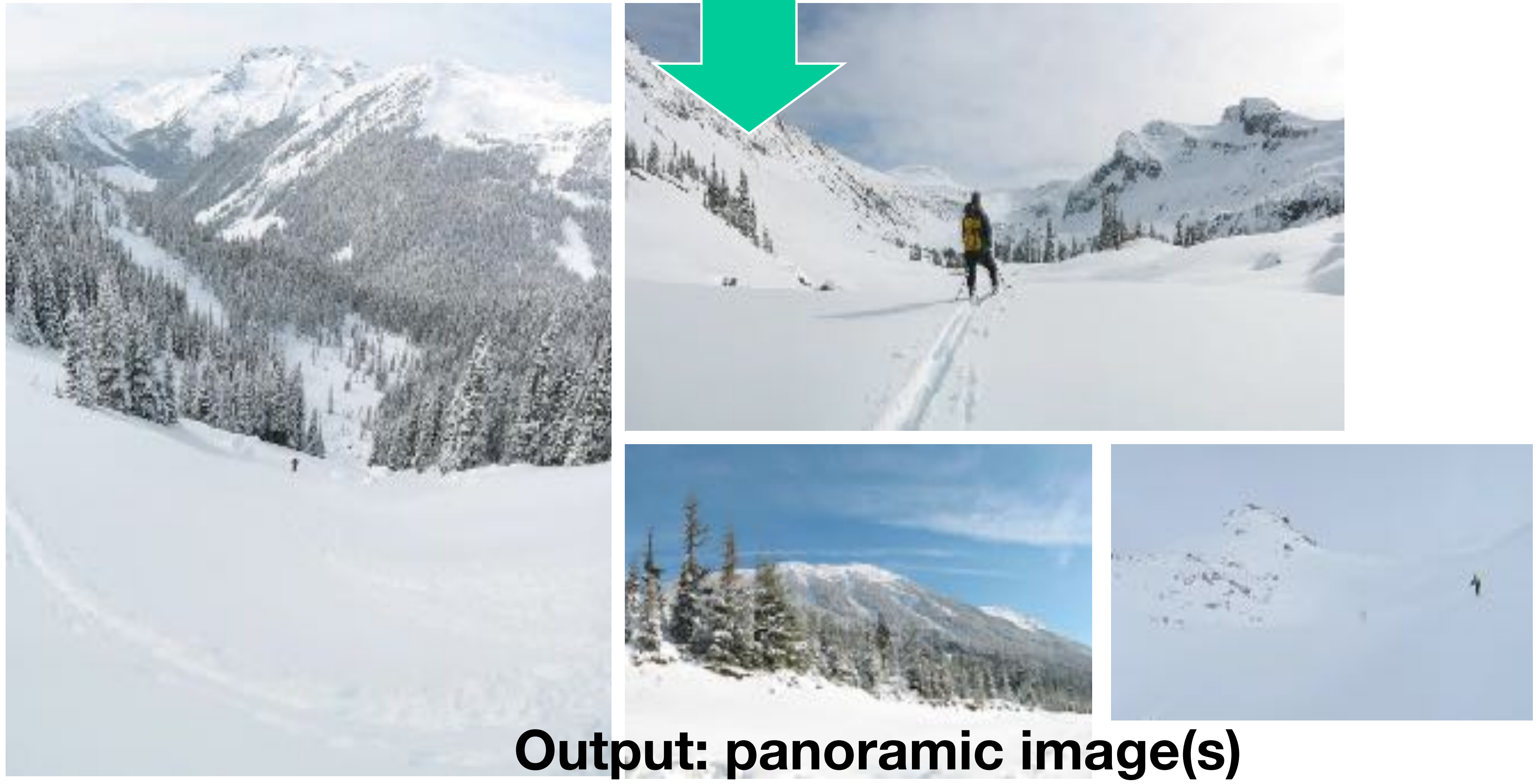
University of British Columbia

* M. Brown and D. Lowe. Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision, 74(1), pages 59-73, 2007 (pdf 3.5Mb | bib) * M. Brown and D. G. Lowe. Recognising Panoramas. In Proceedings of the 9th International Conference on Computer Vision (ICCV2003), pages 1218-1225, Nice, France, 2003 (pdf 820kb | ppt | bib)

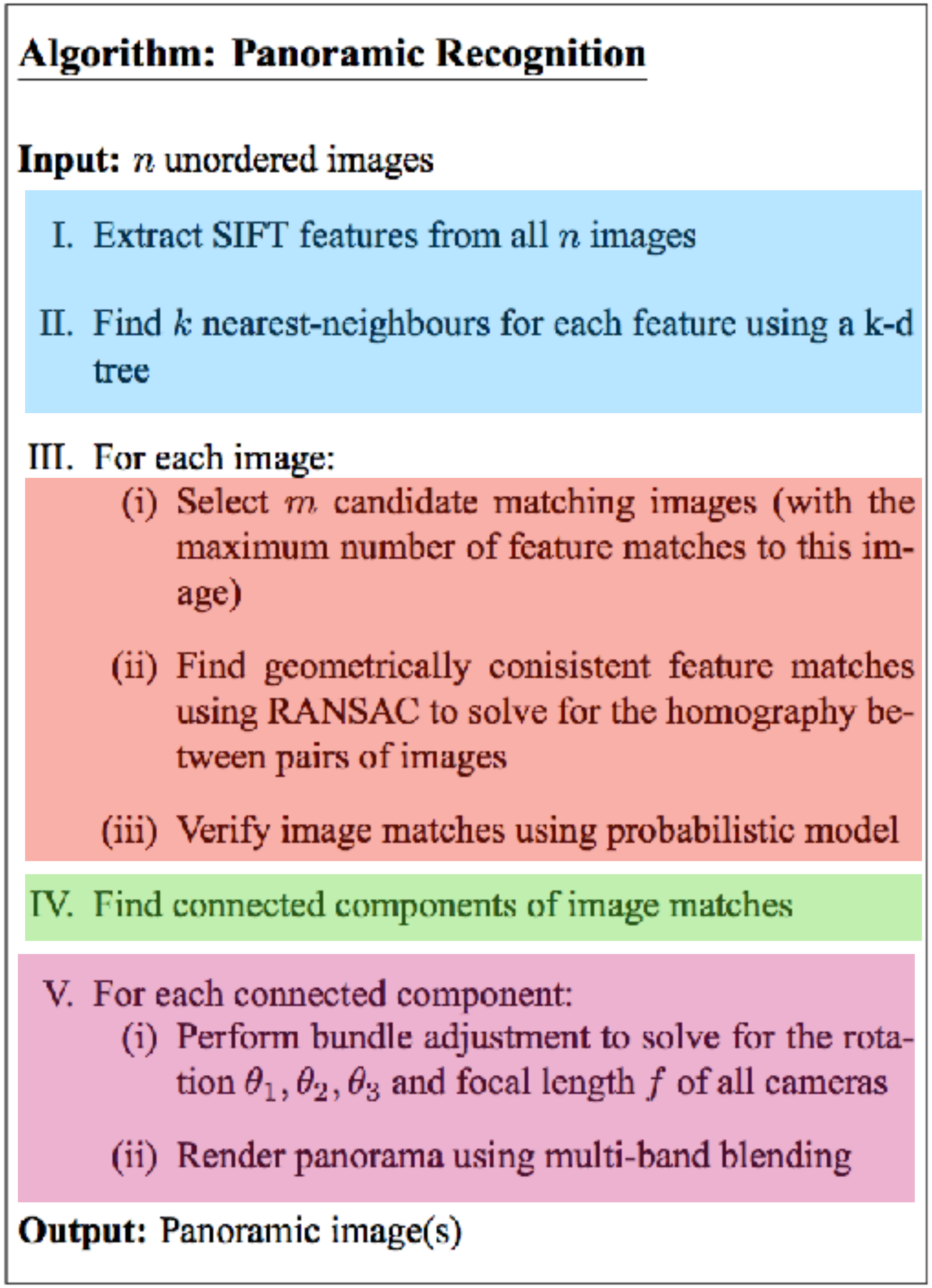
Recognising Panoramas

Input: unordered set of images





Output: panoramic image(s)



Algorithm: Panoramic Recognition

```
Input: $n$ unordered images
I. Extract SIFT features from all $n$ images
II. Find $k$ nearest-neighbours for each feature using a k-d tree
III. For each image:
    (i) Select $m$ candidate matching images (with the maximum number of feature matches to this image)
    (ii) Find geometrically conisistent feature matches using RANSAC to solve for the homography between pairs of images
    (iii) Verify image matches using probabilistic model
IV. Find connected components of image matches
V. For each connected component:
    (i) Perform bundle adjustment to solve for the rotation $\theta_1, \theta_2, \theta_3$ and focal length $f$ of all cameras
    (ii) Render panorama using multi-band blending
Output: Panoramic image(s)
```

Feature Matching

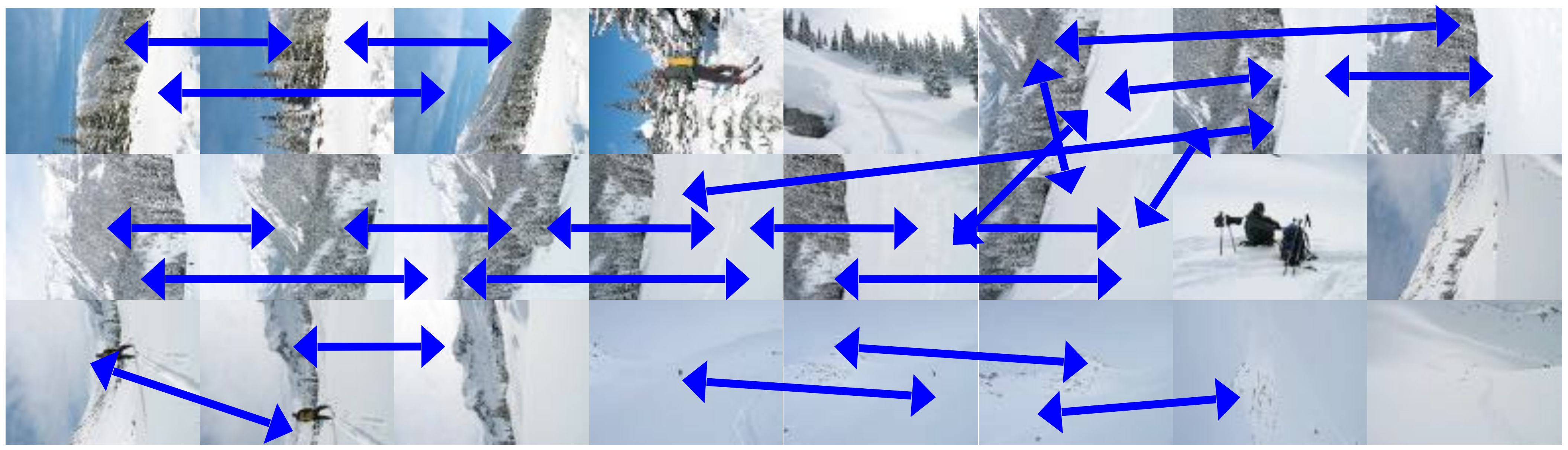
O(nlog(n))

Image (Geometric) Matching

Finding Panoramas

Global Optimization

Finding the panoramas

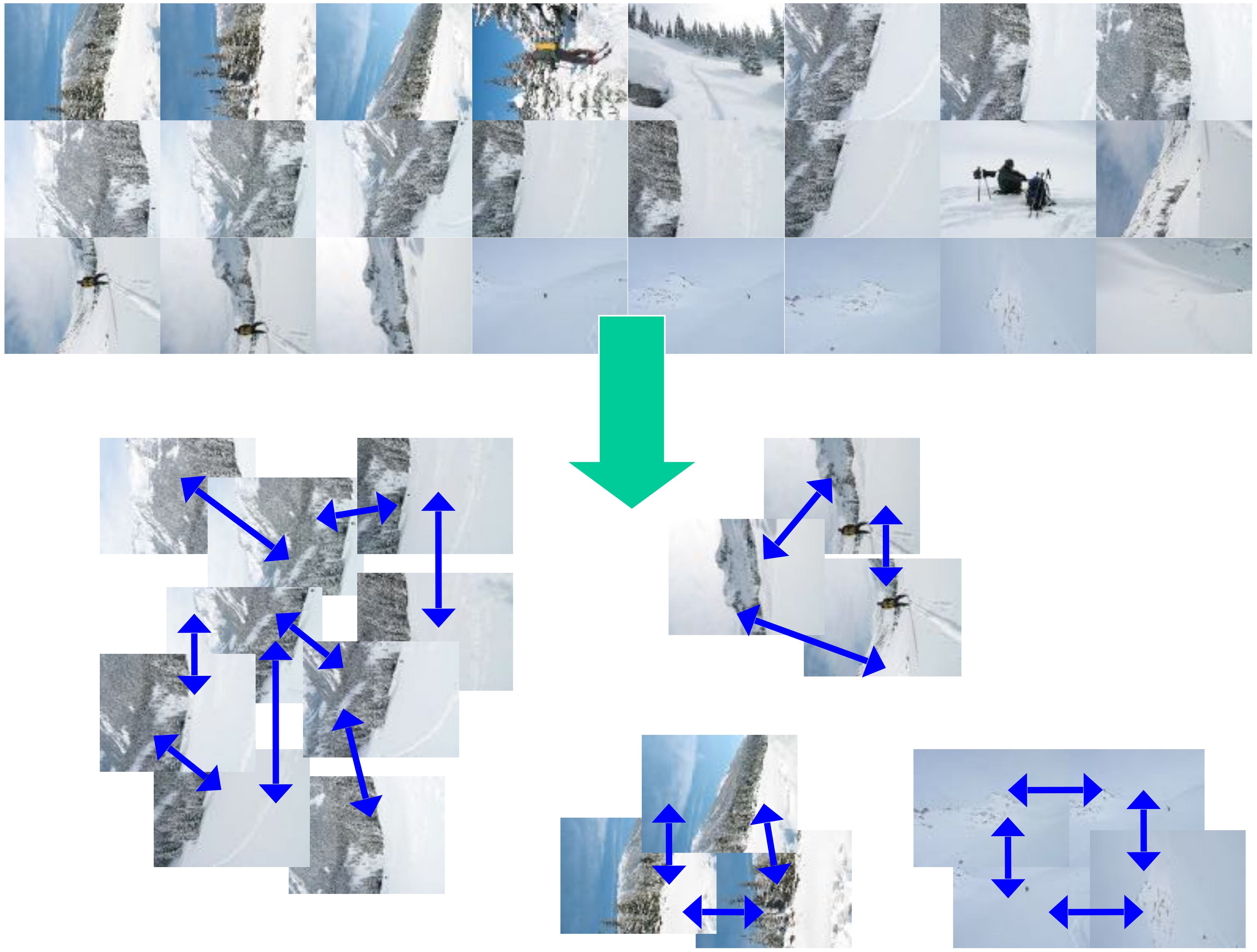


(i) Select m candidate matching images (with the maximum number of feature matches to this image)

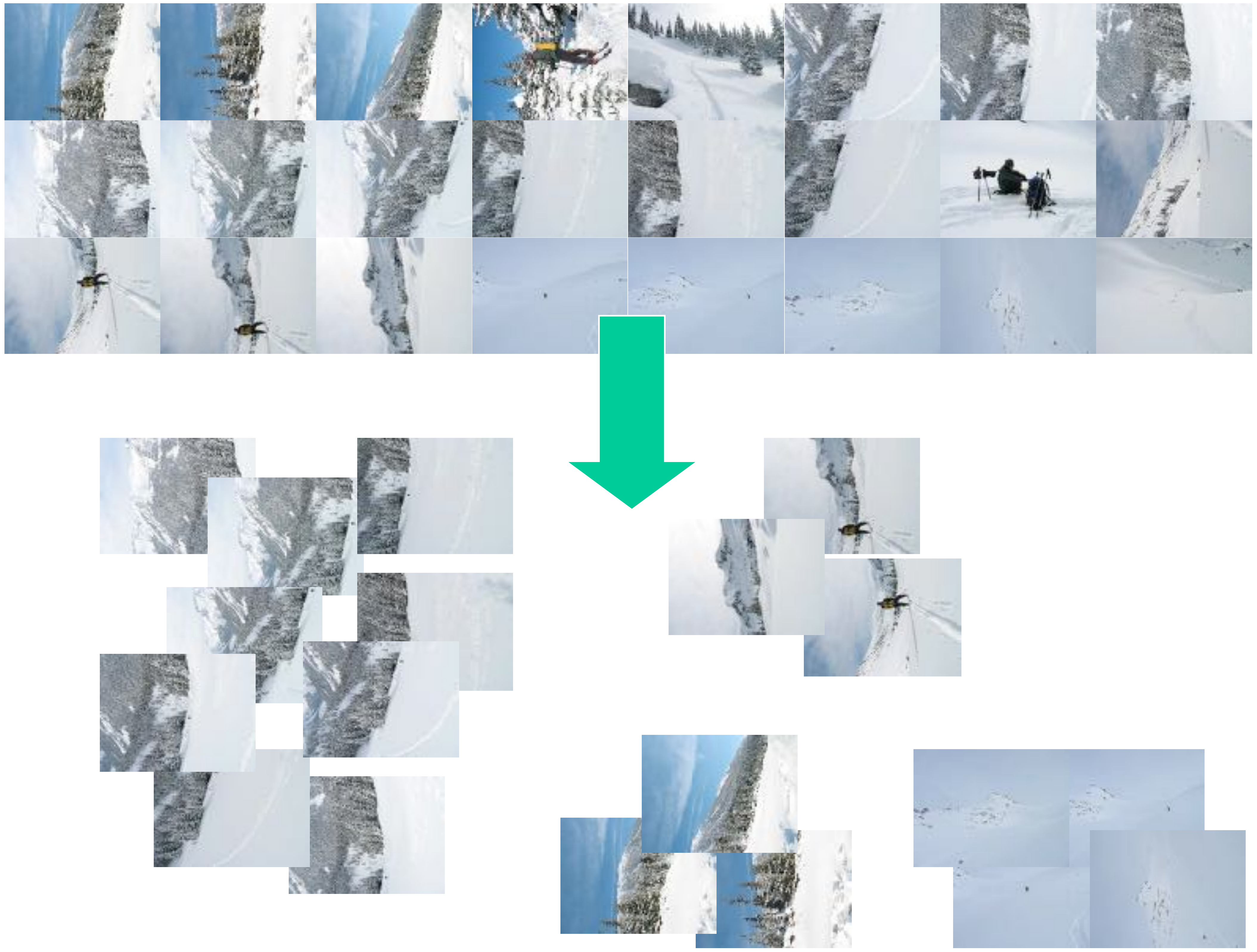
(ii) Find geometrically conisistent feature matches using RANSAC to solve for the homography between pairs of images

(iii) Verify image matches using probabilistic model

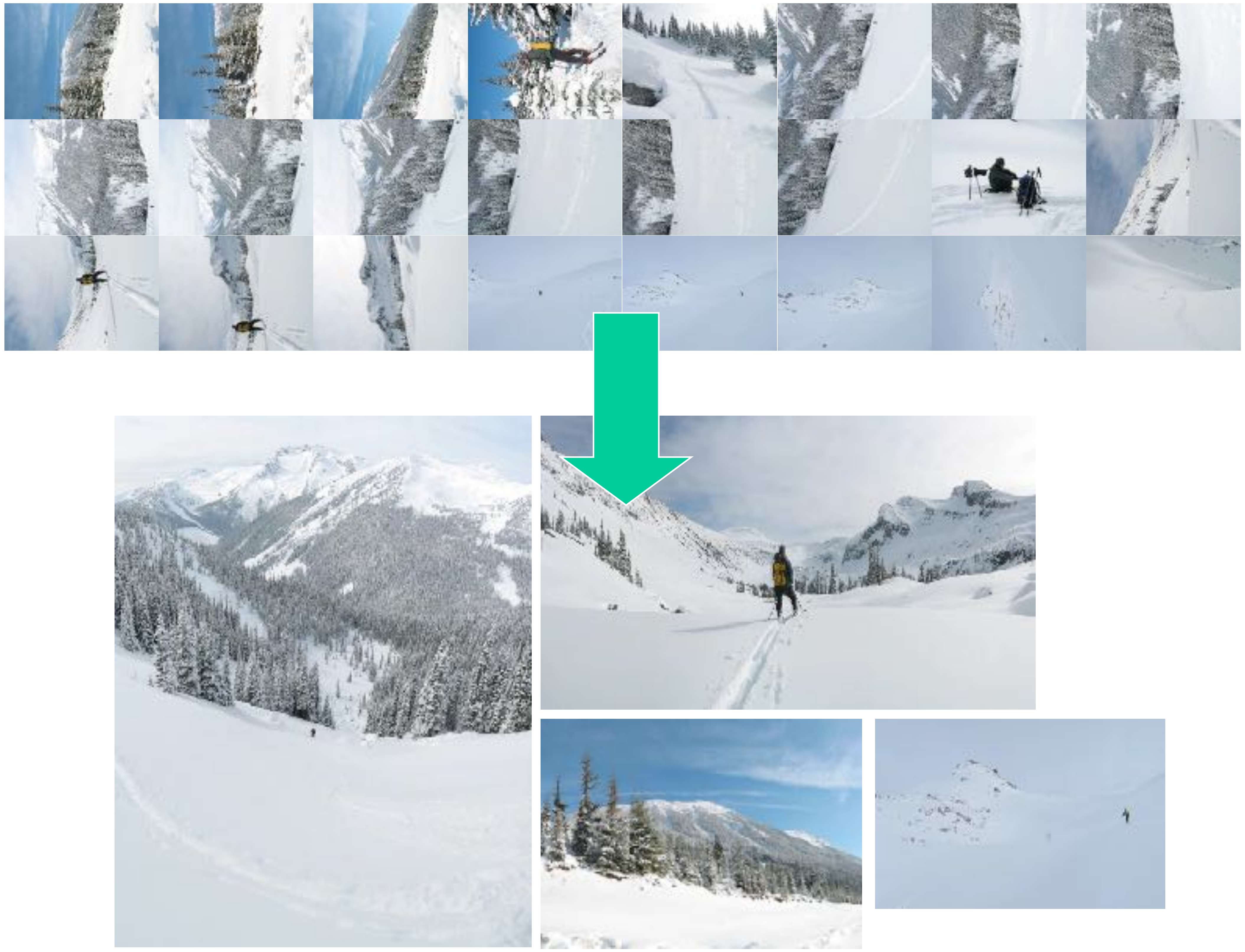
Finding the panoramas



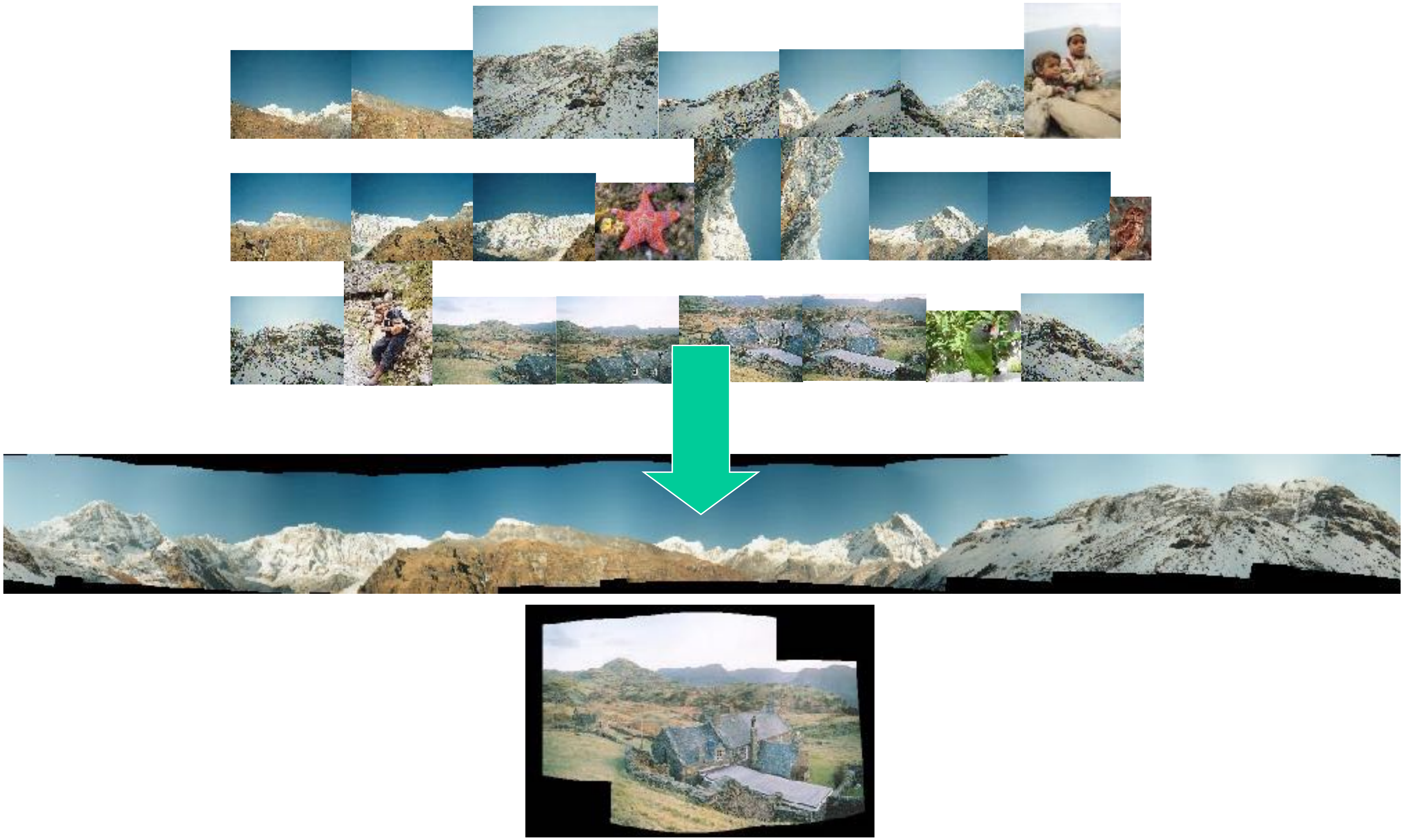
Finding the panoramas



Finding the panoramas



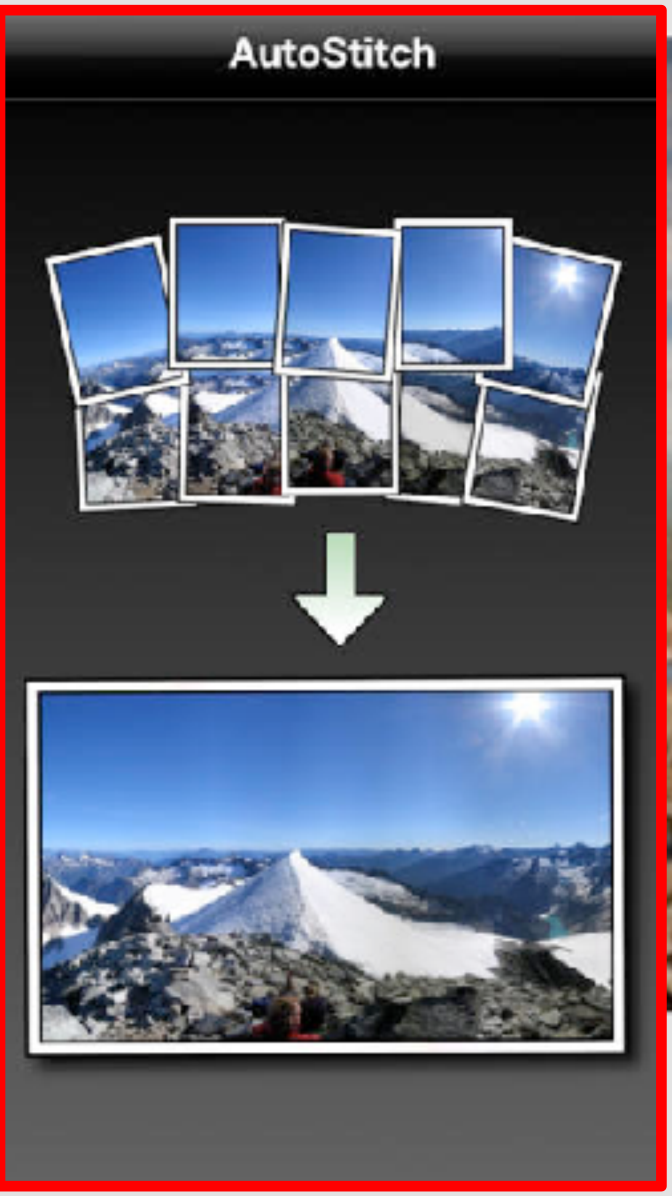
Results



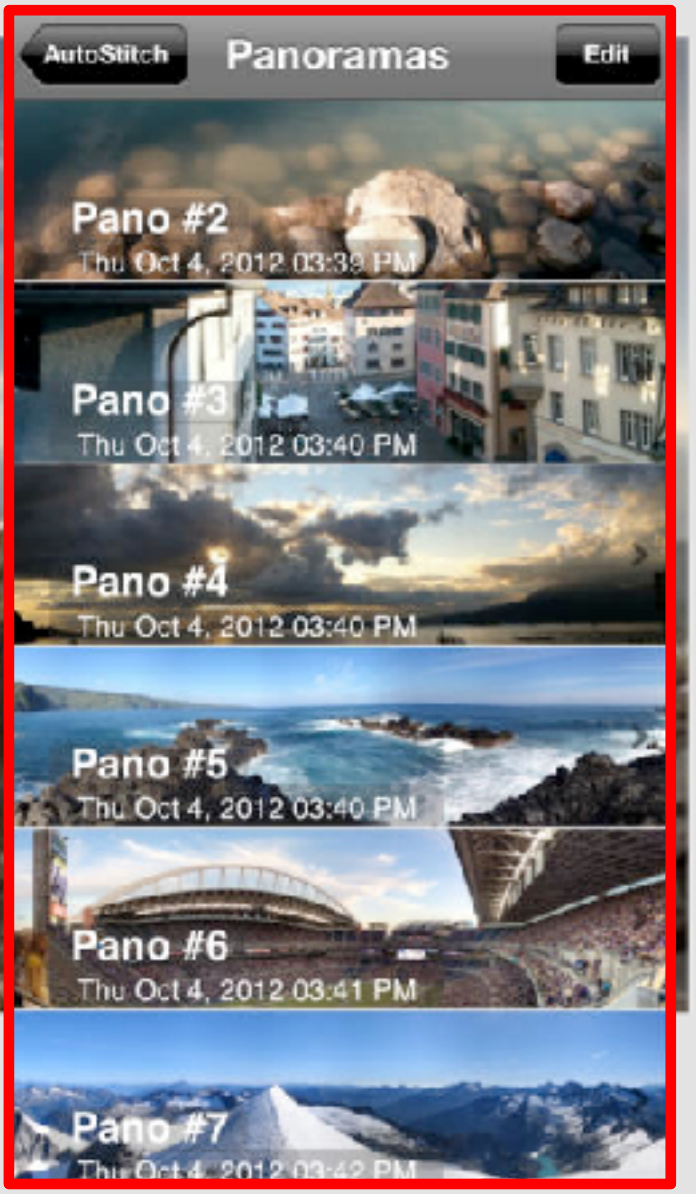
AUTOSTITCH

AutoStitch | Gallery | **Download (Windows demo)** | Buy Autopano | Licensing | Press | FAQ | Publications

AutoStitch :: a new dimension in automatic image stitching

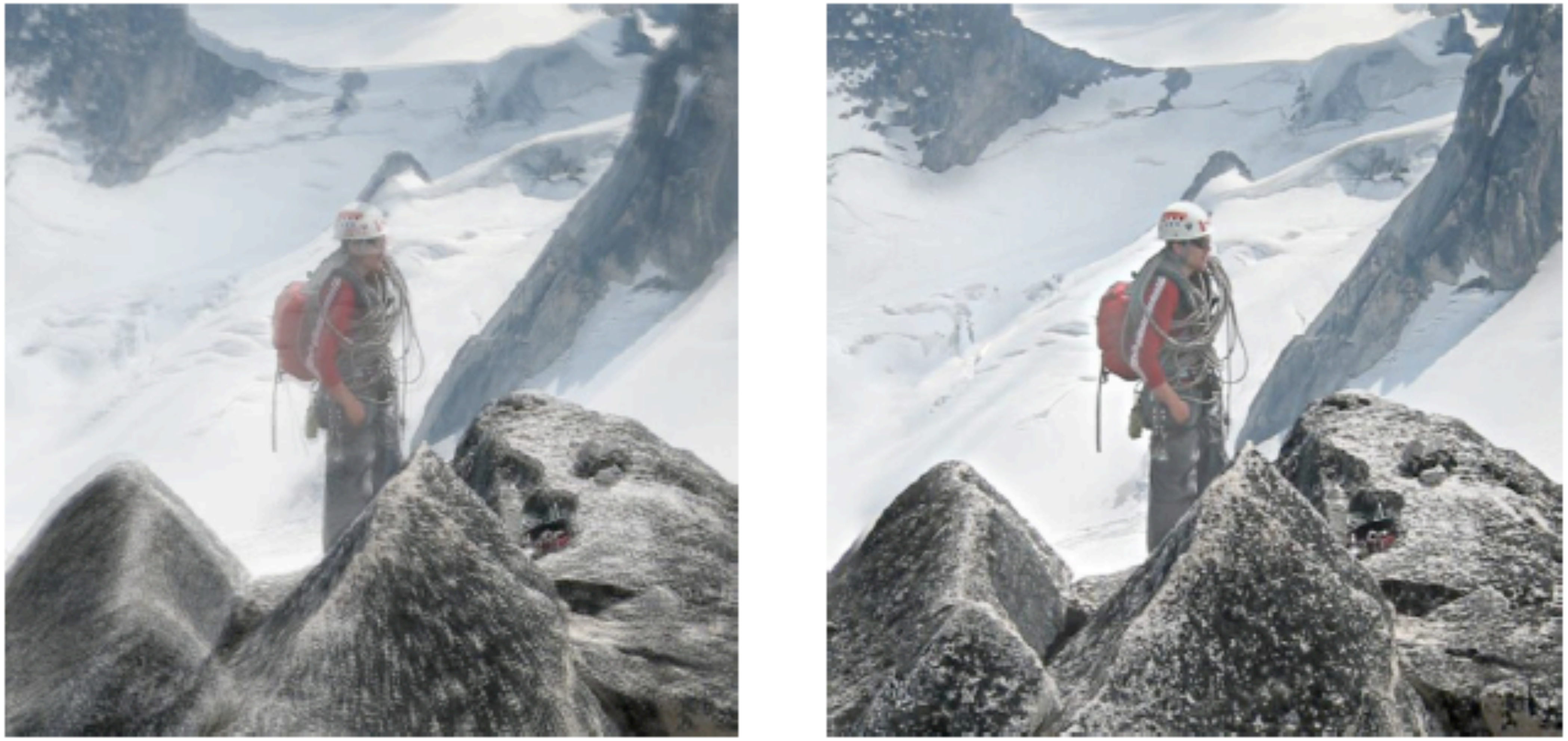


Serratus



Welcome to AutoStitch. If you have an iPhone, please check out our new iPhone version of AutoStitch below! If you're looking for the Windows demo version, you can download it using the link above, or read on to find out more about AutoStitch. Thanks for visiting!

Benefits of Laplacian image compositing



(a) Linear blending (b) Multi-band blending

Figure 7. Comparison of linear and multi-band blending. The image on the right was blended using multi-band blending using 5 bands and $\sigma = 5$ pixels. The image on the left was linearly blended. In this case matches on the moving person have caused small misregistrations between the images, which cause blurring in the linearly blended result, but the multi-band blended image is clear.

M. Brown and D. Lowe. Automatic Panoramic Image Stitching using Invariant Features. International Journal of Computer Vision, 74(1), pages 59-73, 2007

Photo Tourism:
Exploring Photo Collections in 3D

Noah Snavely

Steven M. Seitz

University of Washington

Richard Szeliski

Microsoft Research

15,464

37,383

76,389

Photo Tourism
Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski
University of Washington *Microsoft Research*

SIGGRAPH 2006

Rendering

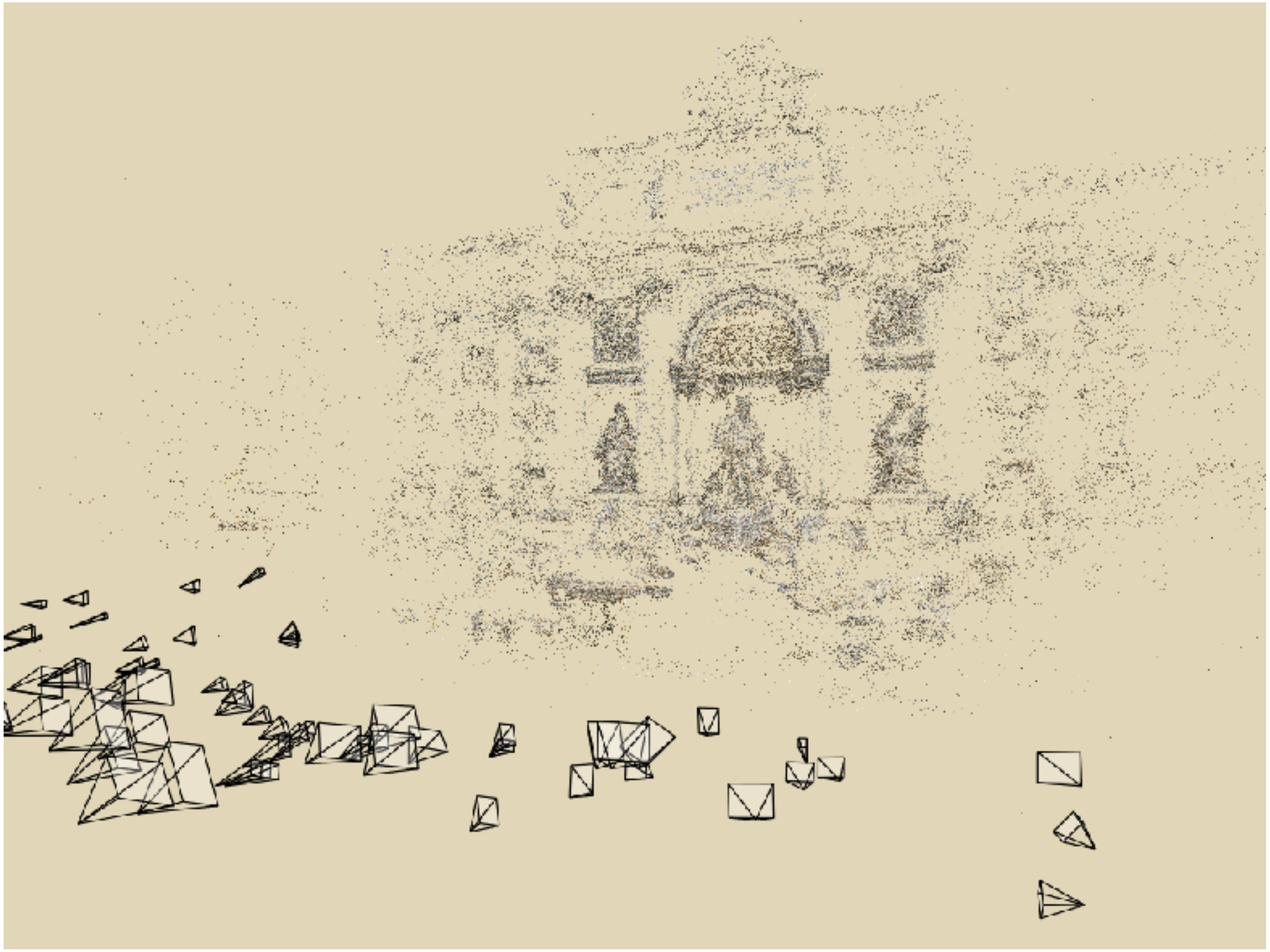
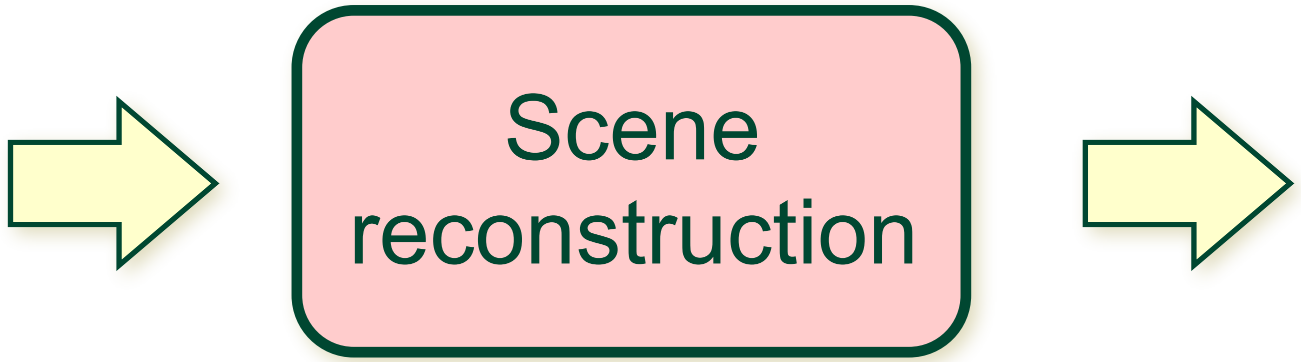
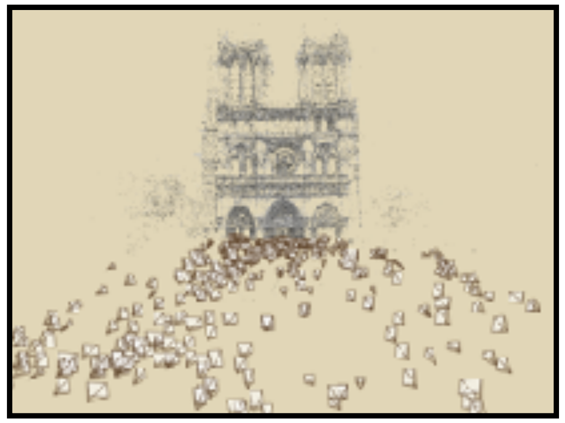


Photo Tourism overview

Input photographs





Relative camera positions and orientations

Point cloud

Sparse correspondence

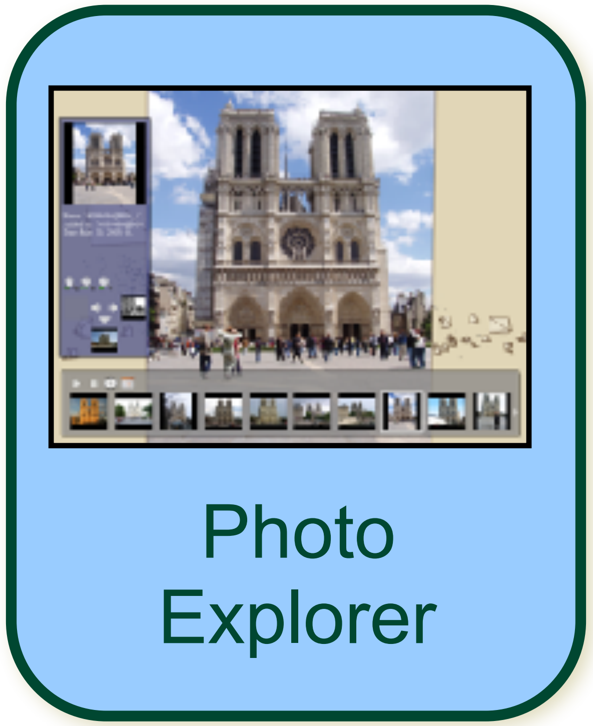
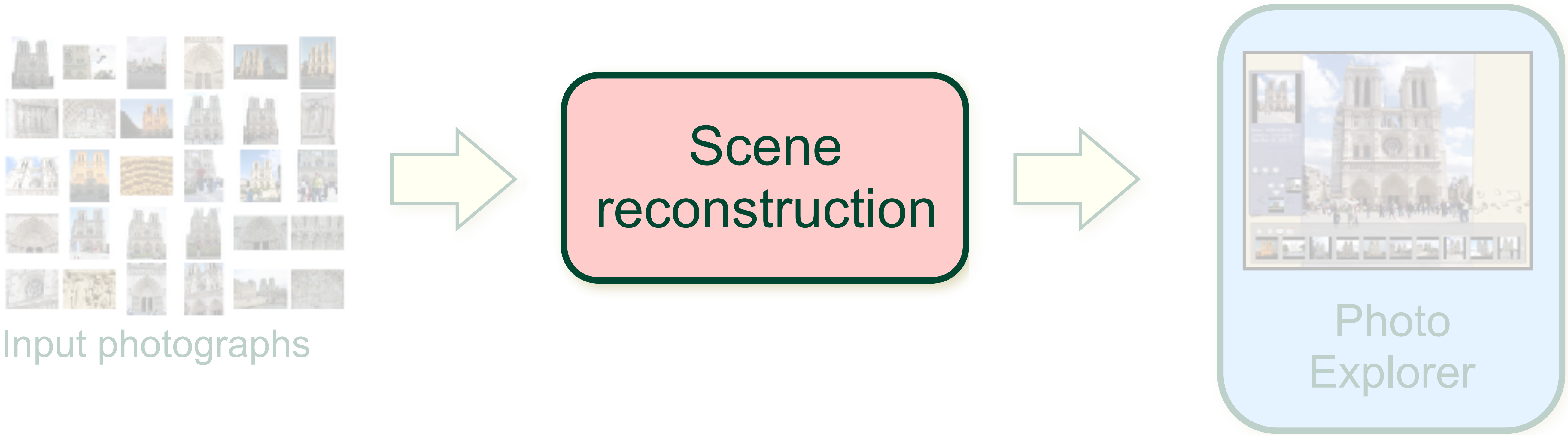
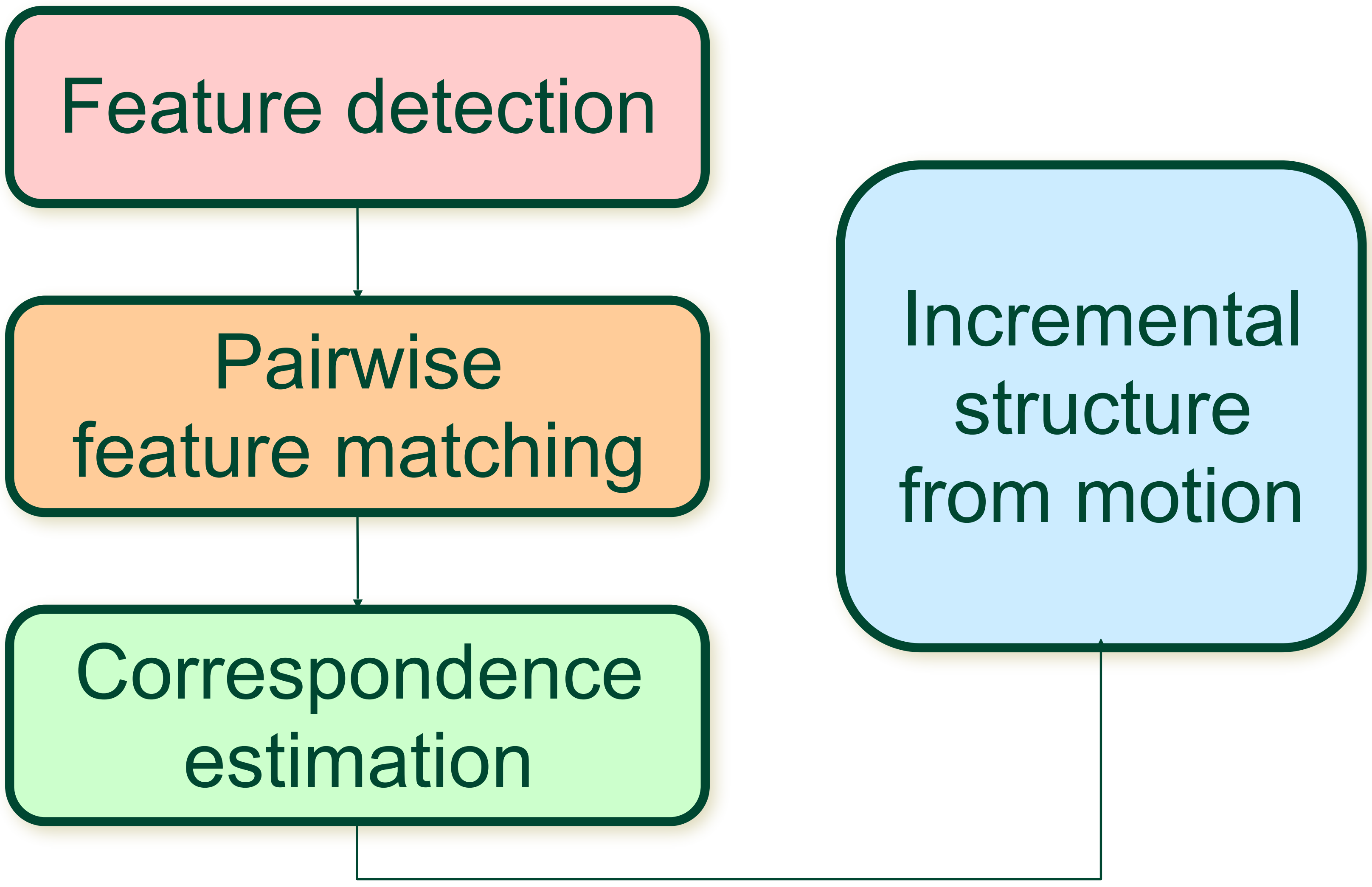


Photo Tourism overview



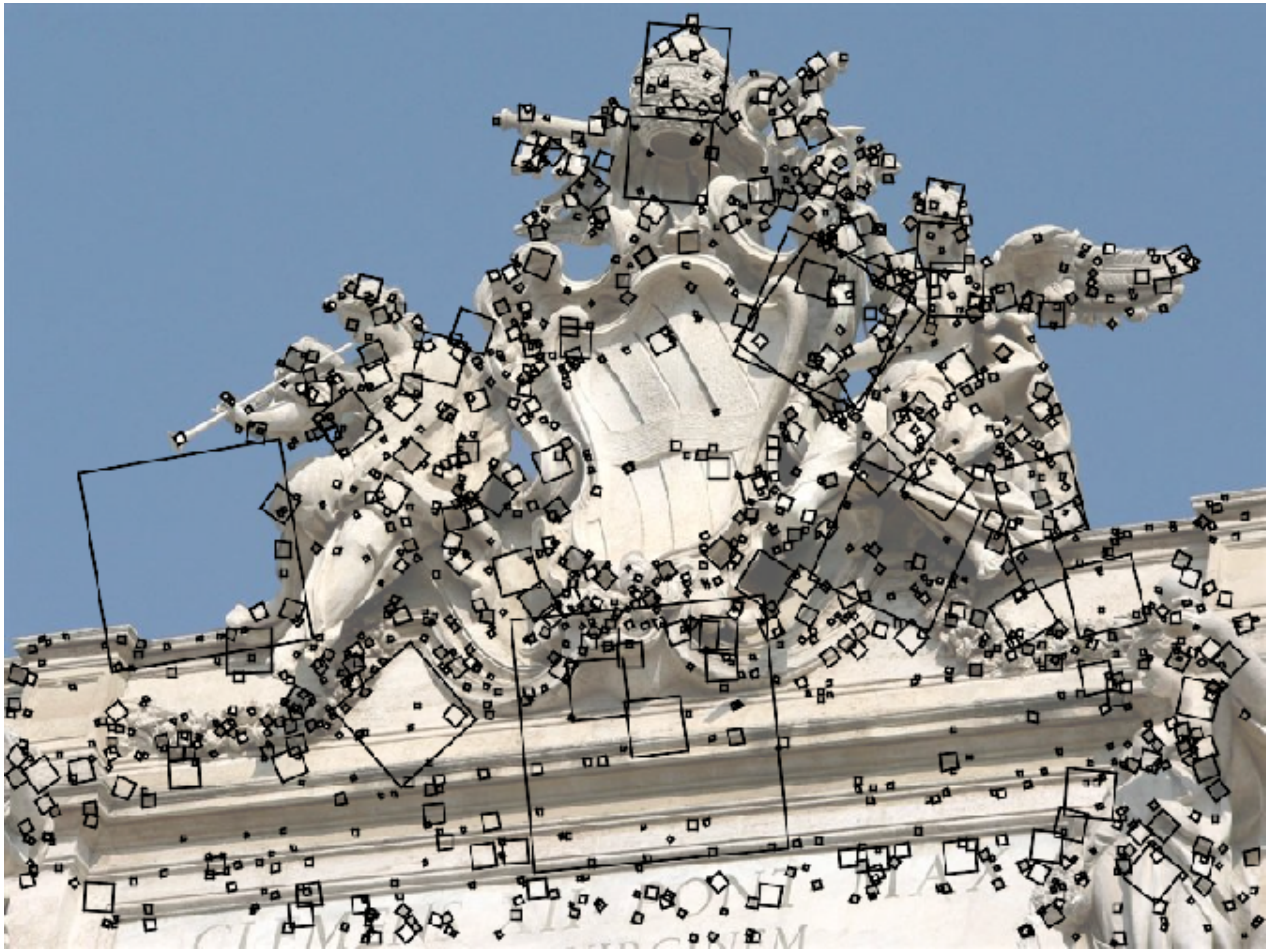
Scene reconstruction

- Automatically estimate
  - position, orientation, and focal length of cameras
  - 3D positions of feature points


Feature detection
Pairwise feature matching
Correspondence estimation
Incremental structure from motion

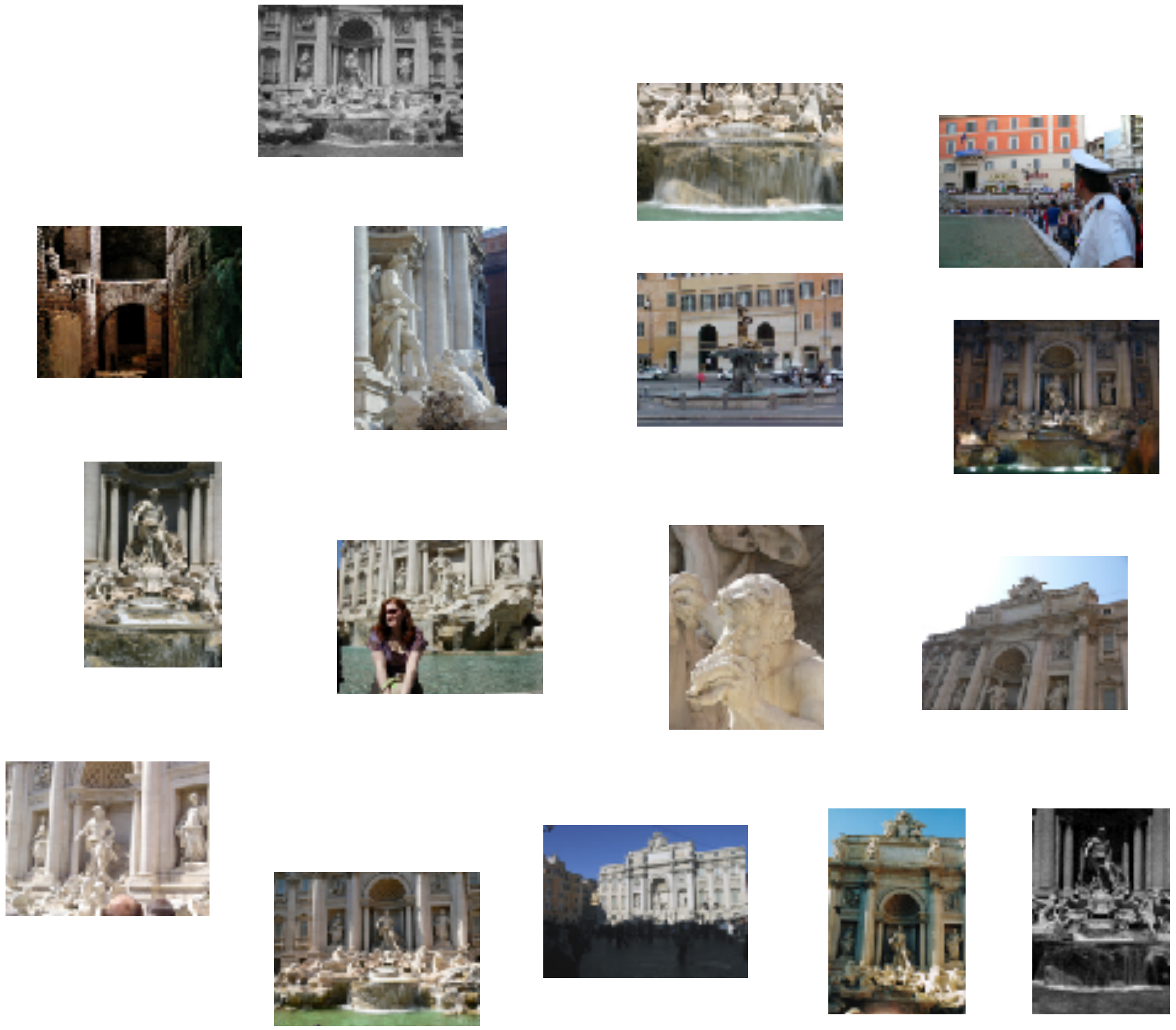
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

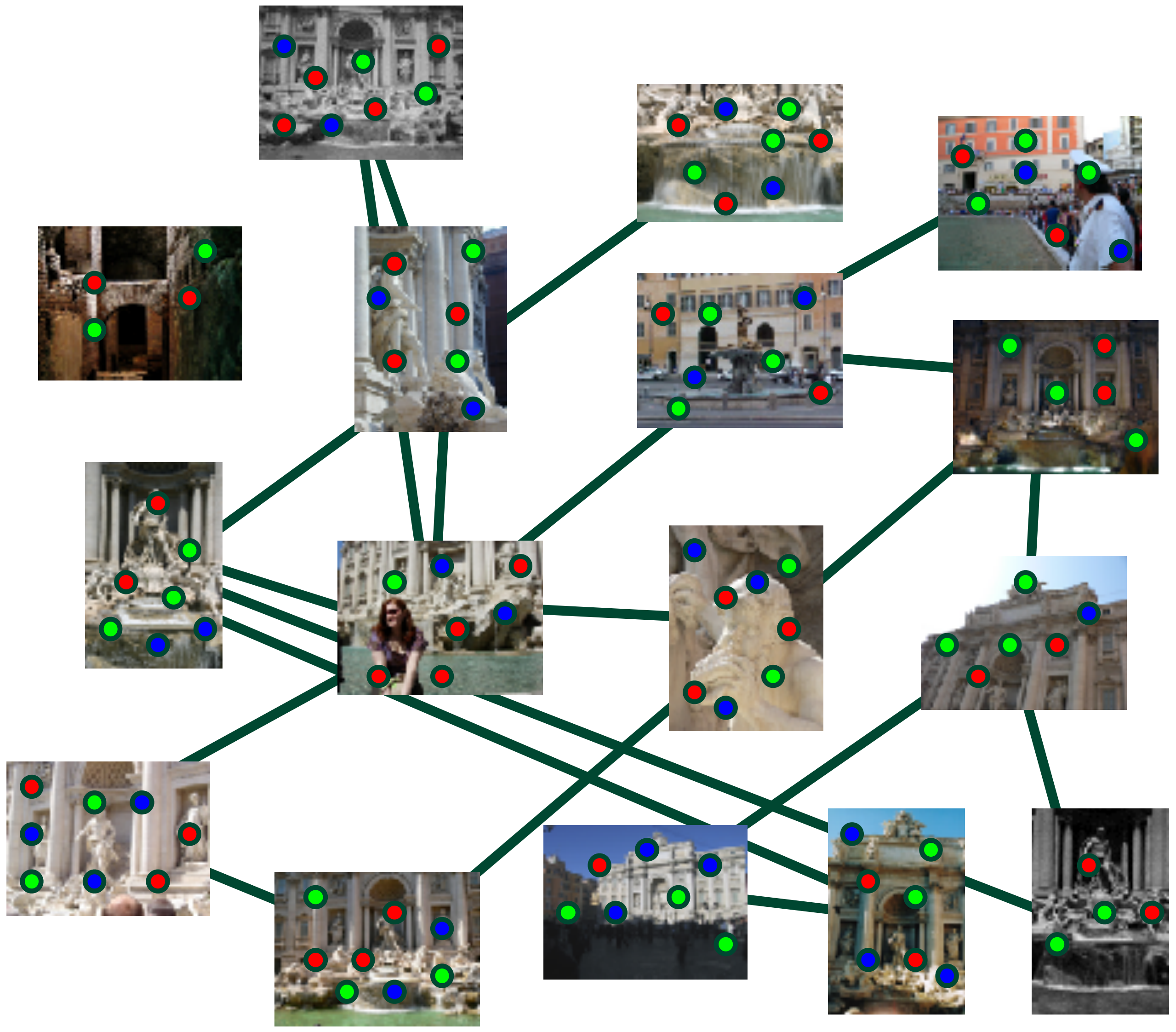


Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

Feature matching

Match features between each pair of images

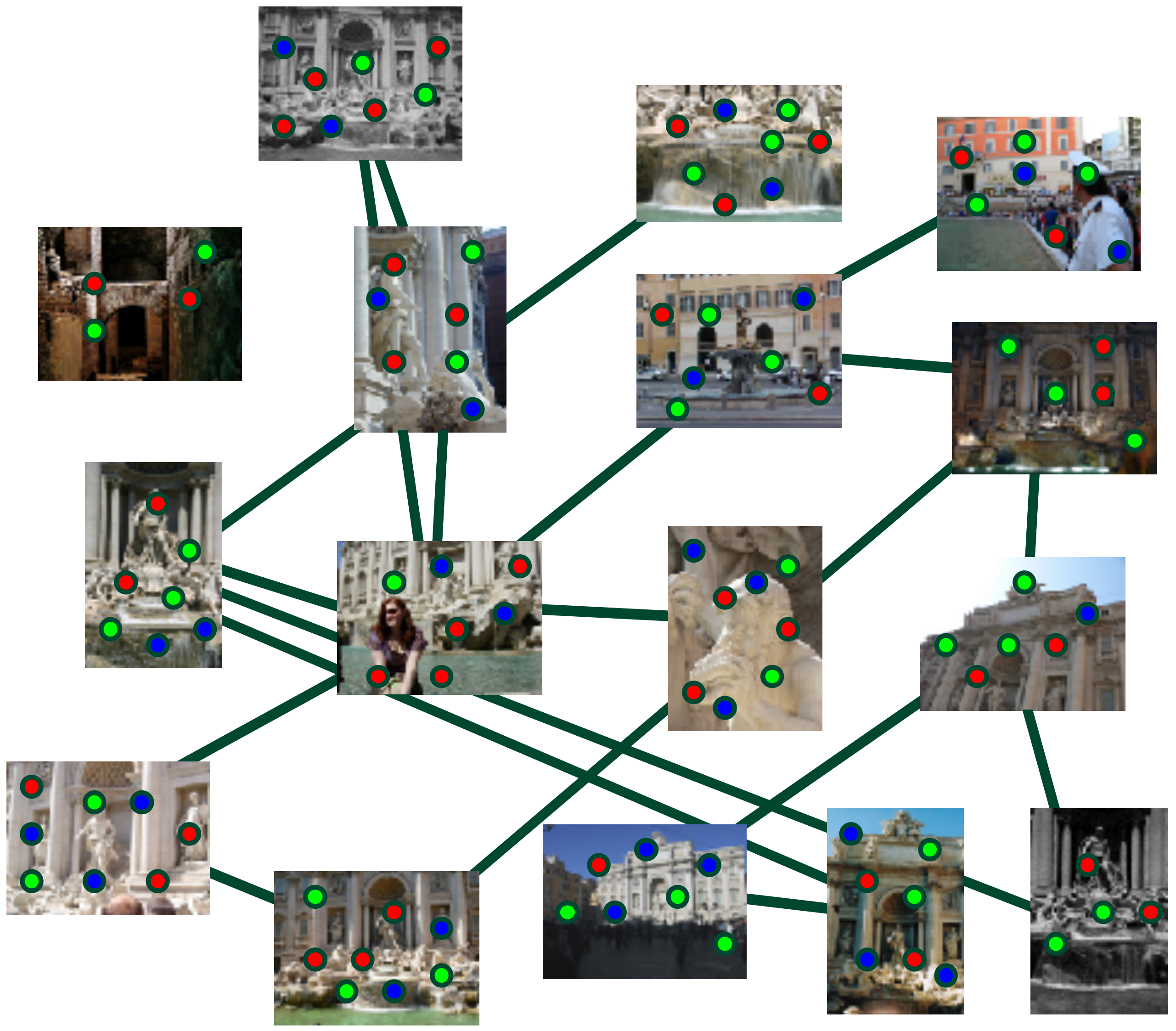


Feature matching

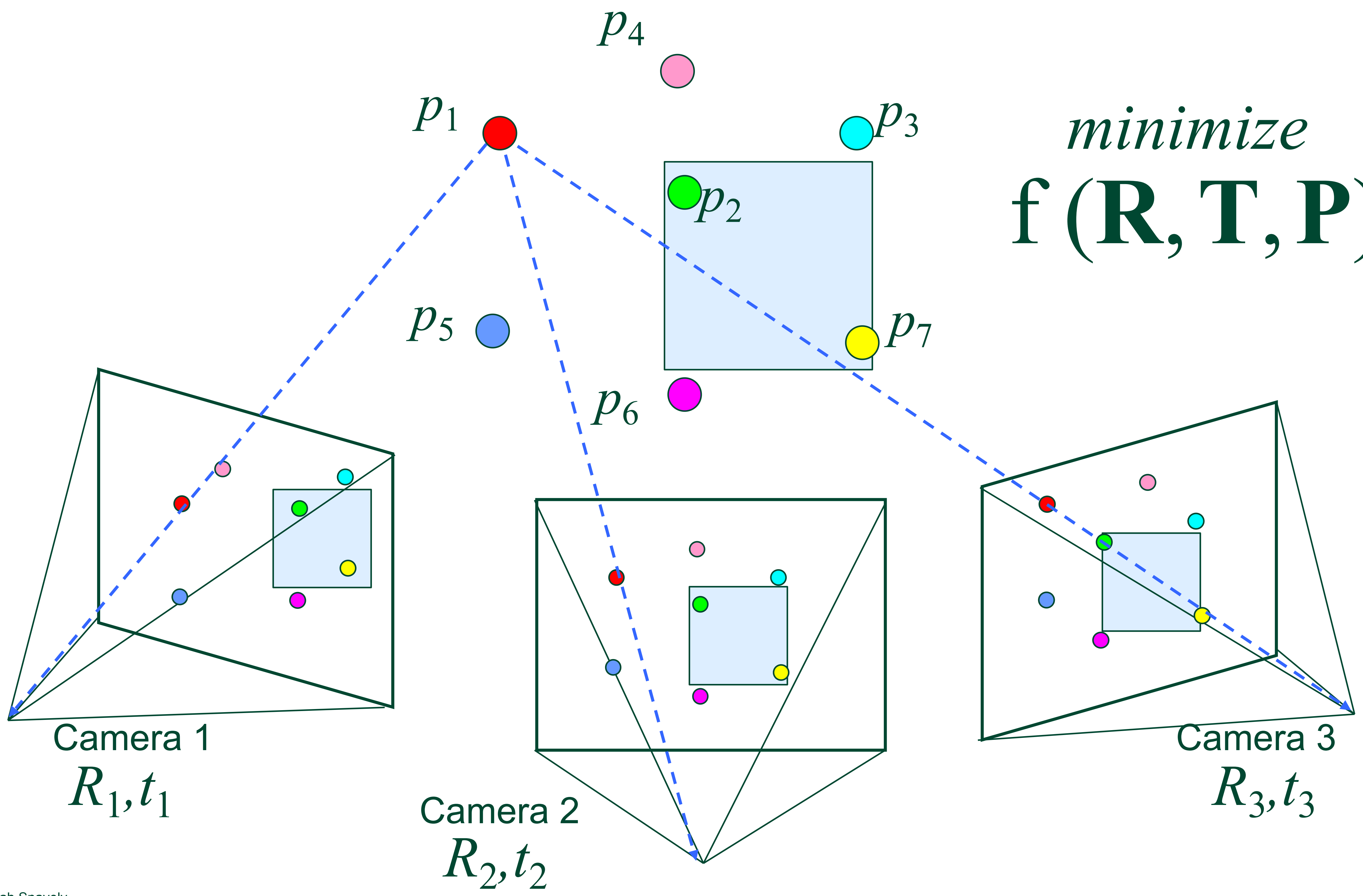
Refine matching using RANSAC [Fischler & Bolles 1987] to estimate fundamental matrices between pairs

(See 6.801/6.866 for fundamental matrix, or Hartley and Zisserman, Multi-View Geometry.

See also the fundamental matrix song: http://danielwedge.com/fmatrix/)



Structure from motion



minimize

$$\mathrm{f}\,(\mathbf{R}, \mathbf{T}, \mathbf{P})$$

Links

- Code available: http://phototour.cs.washington.edu/bundler/
- http://phototour.cs.washington.edu/
- http://livelabs.com/photosynth/
- http://www.cs.cornell.edu/~snavely/