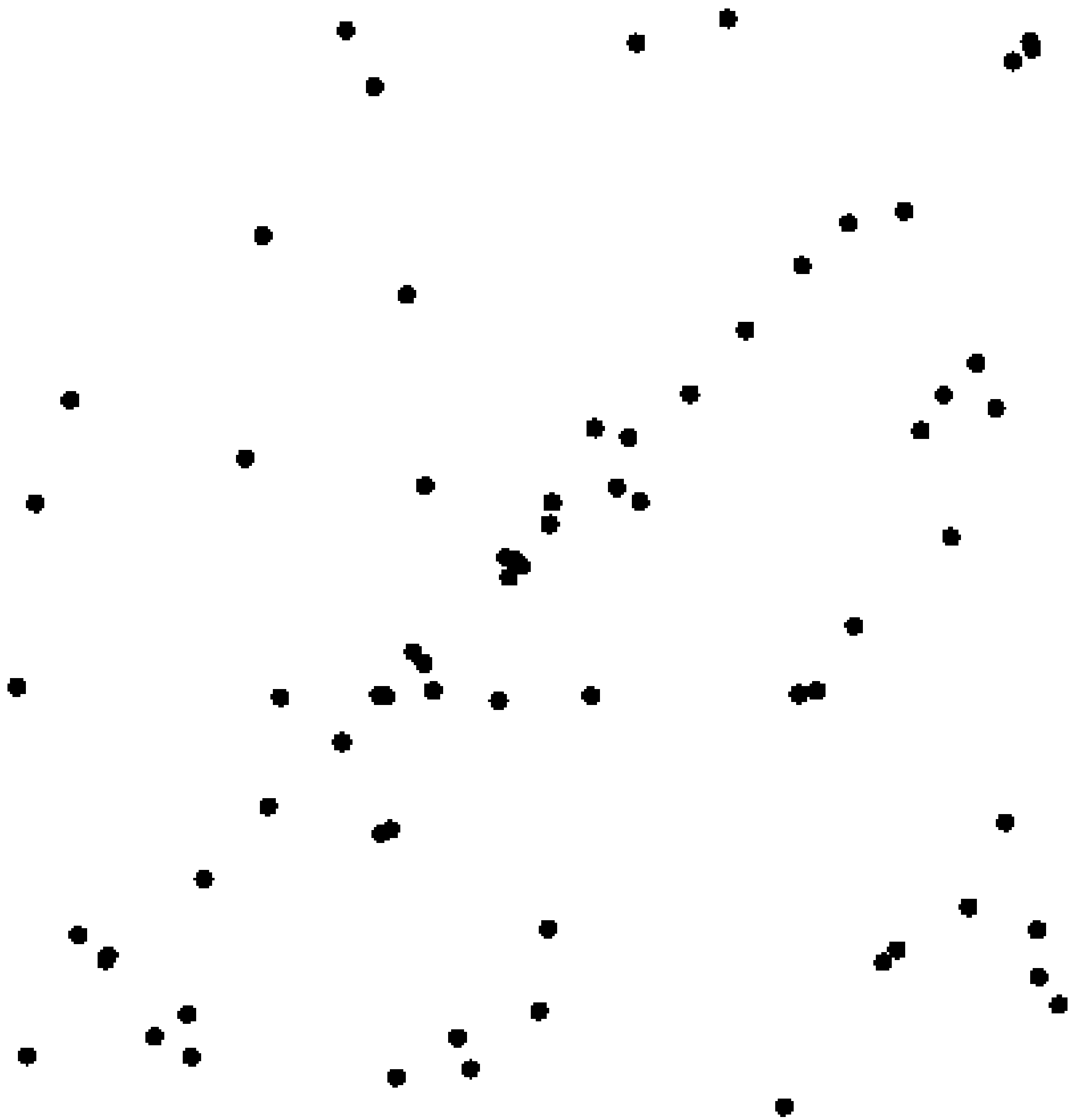
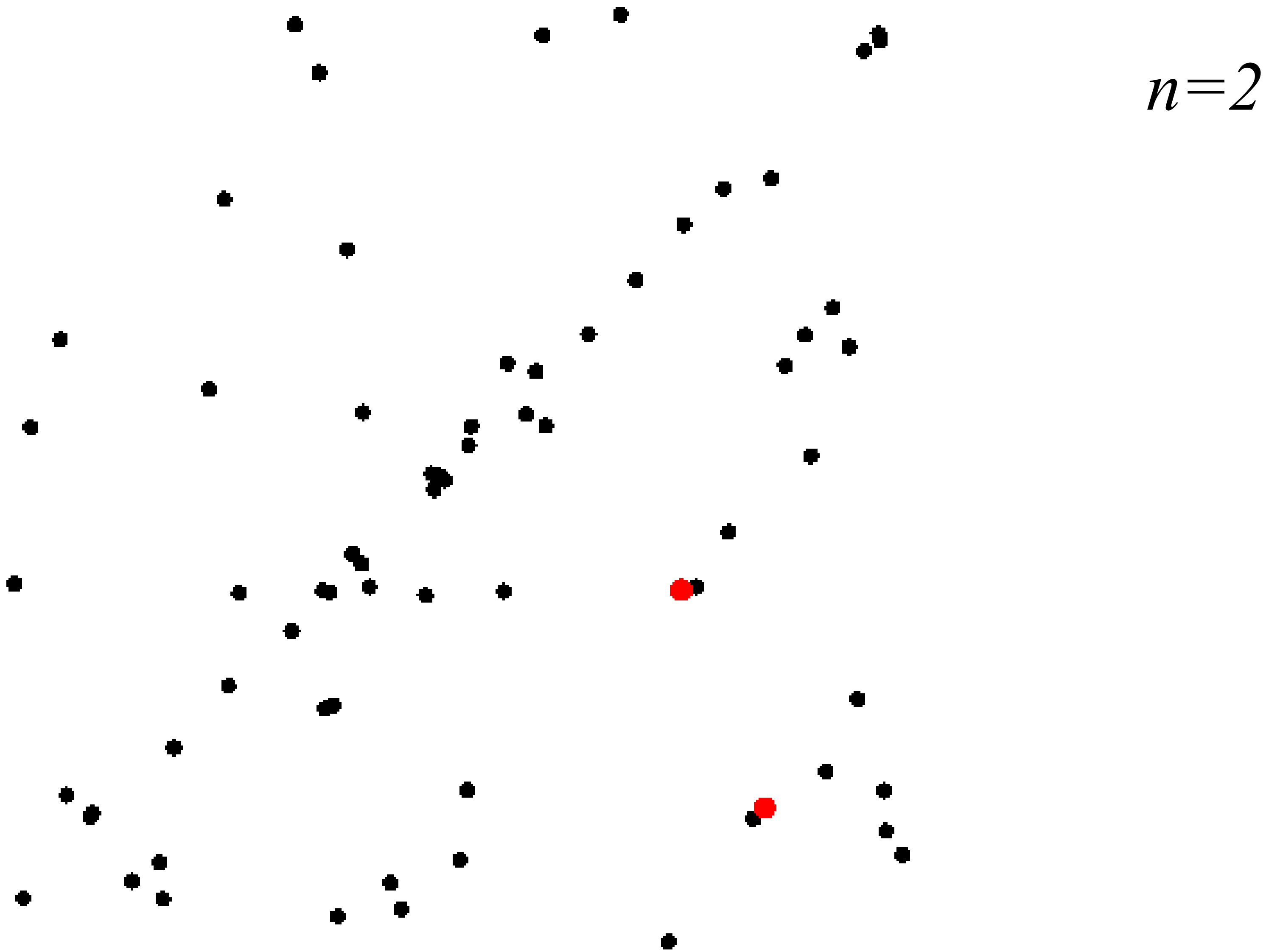
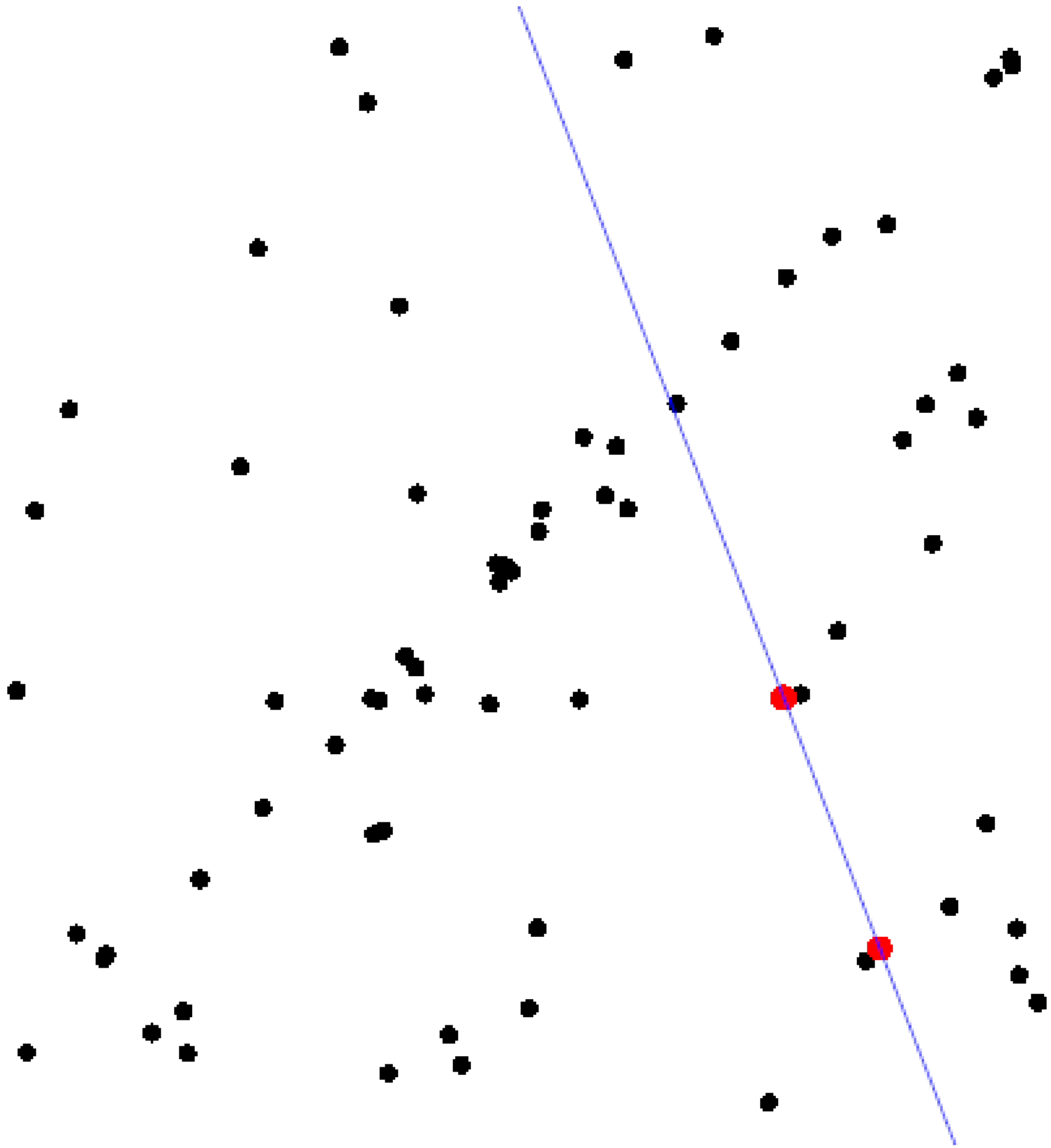
# Example: line fitting



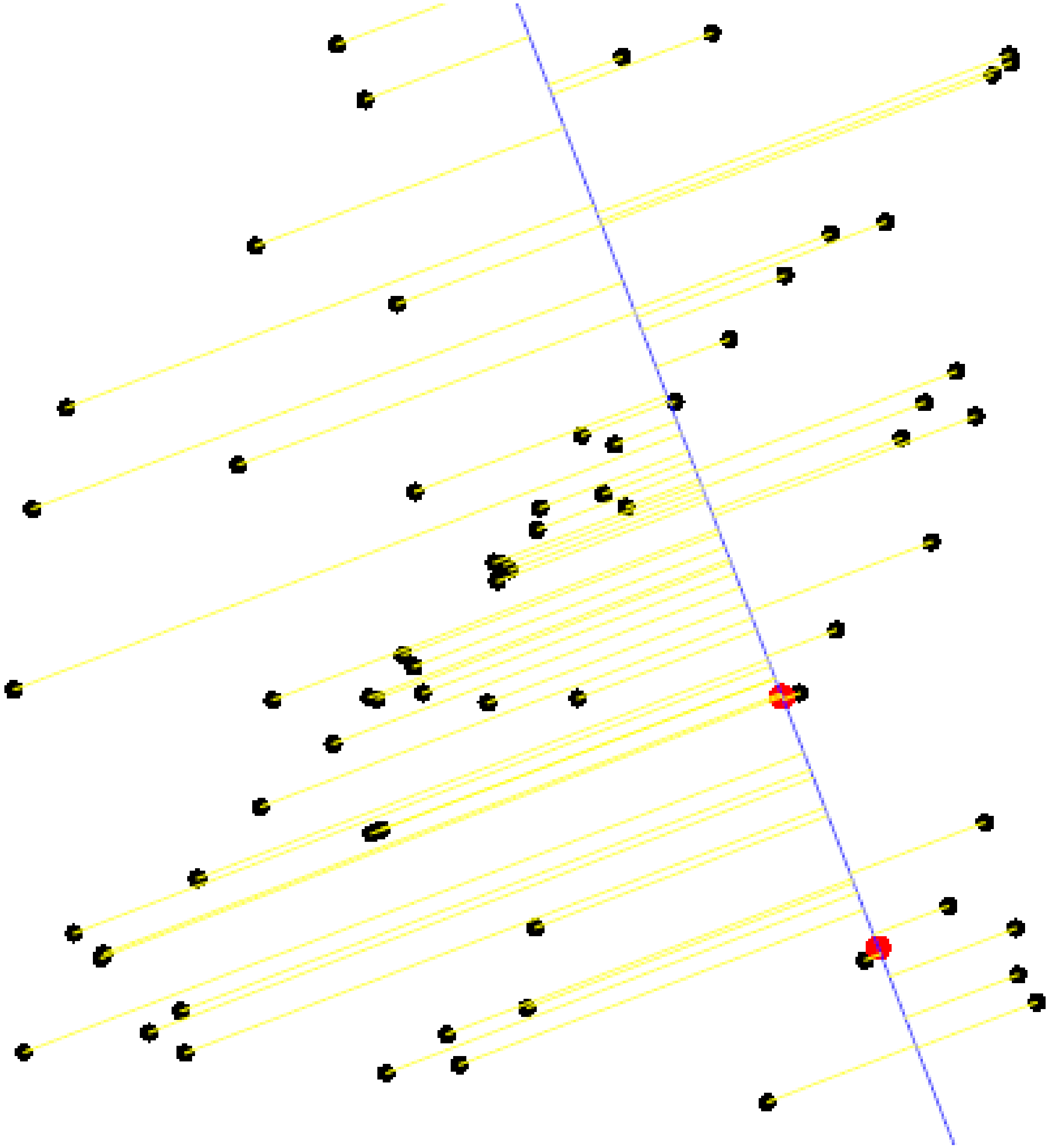
# Example: line fitting



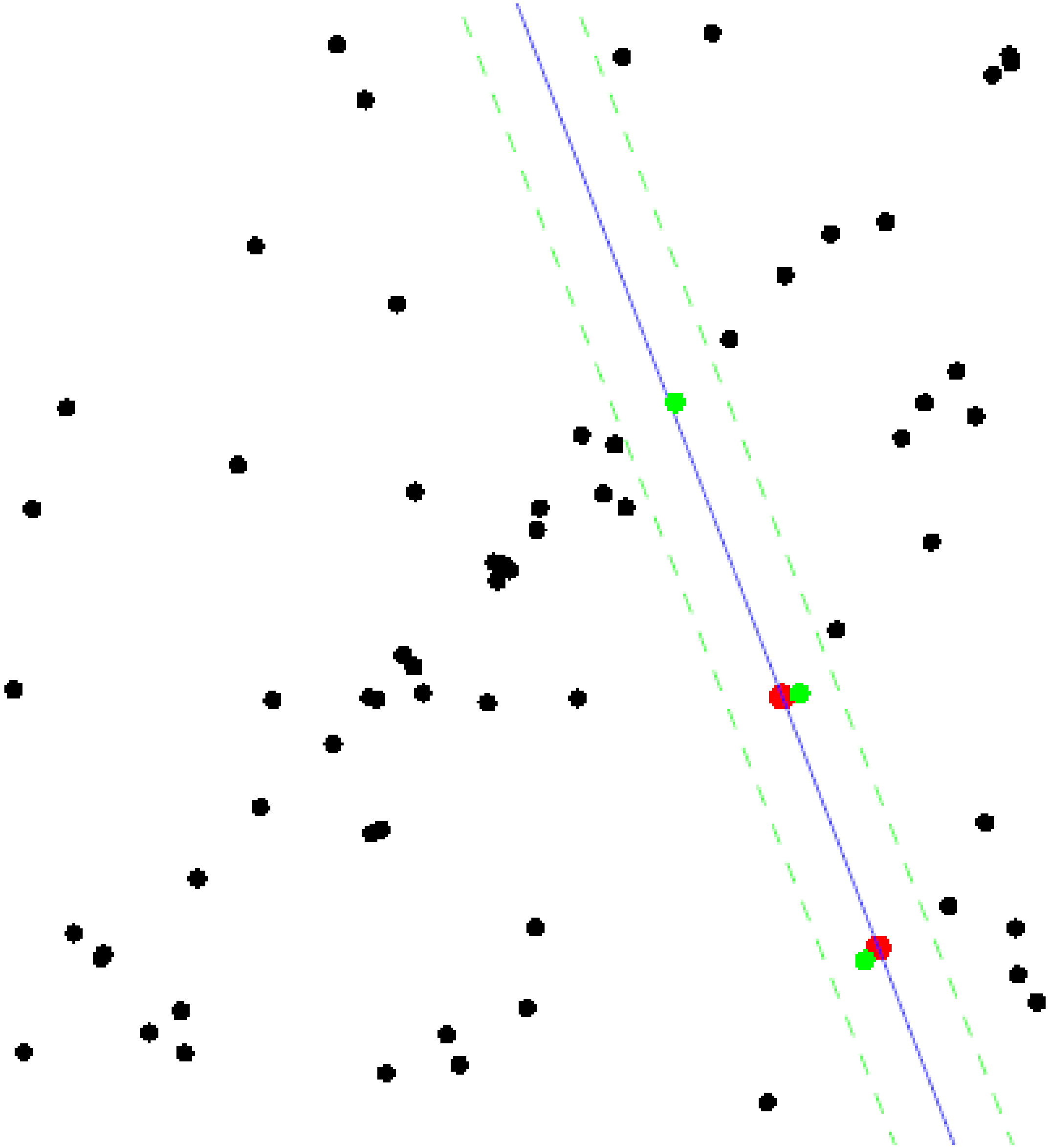
# Model fitting



# Measure distances

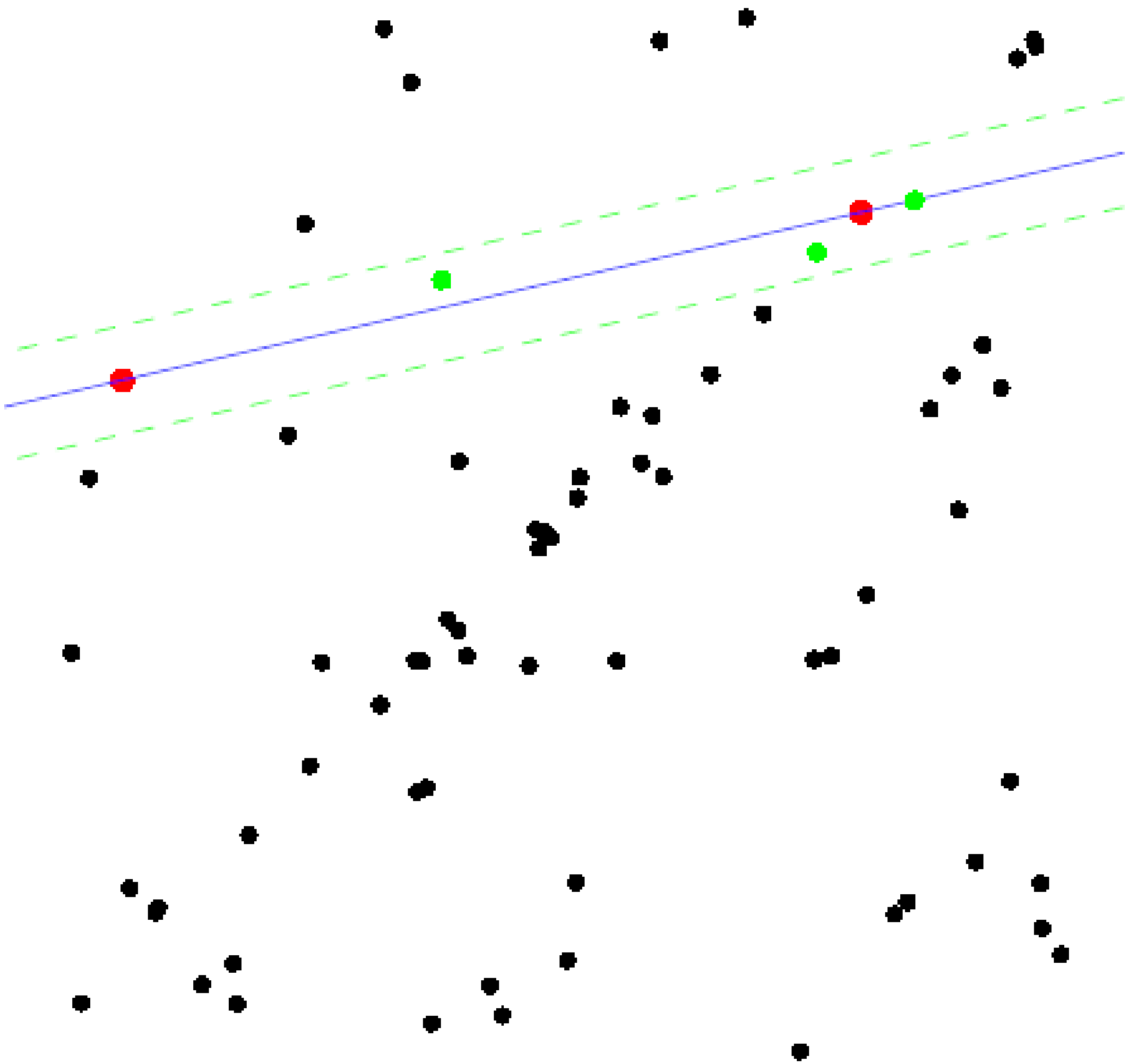


# Count inliers



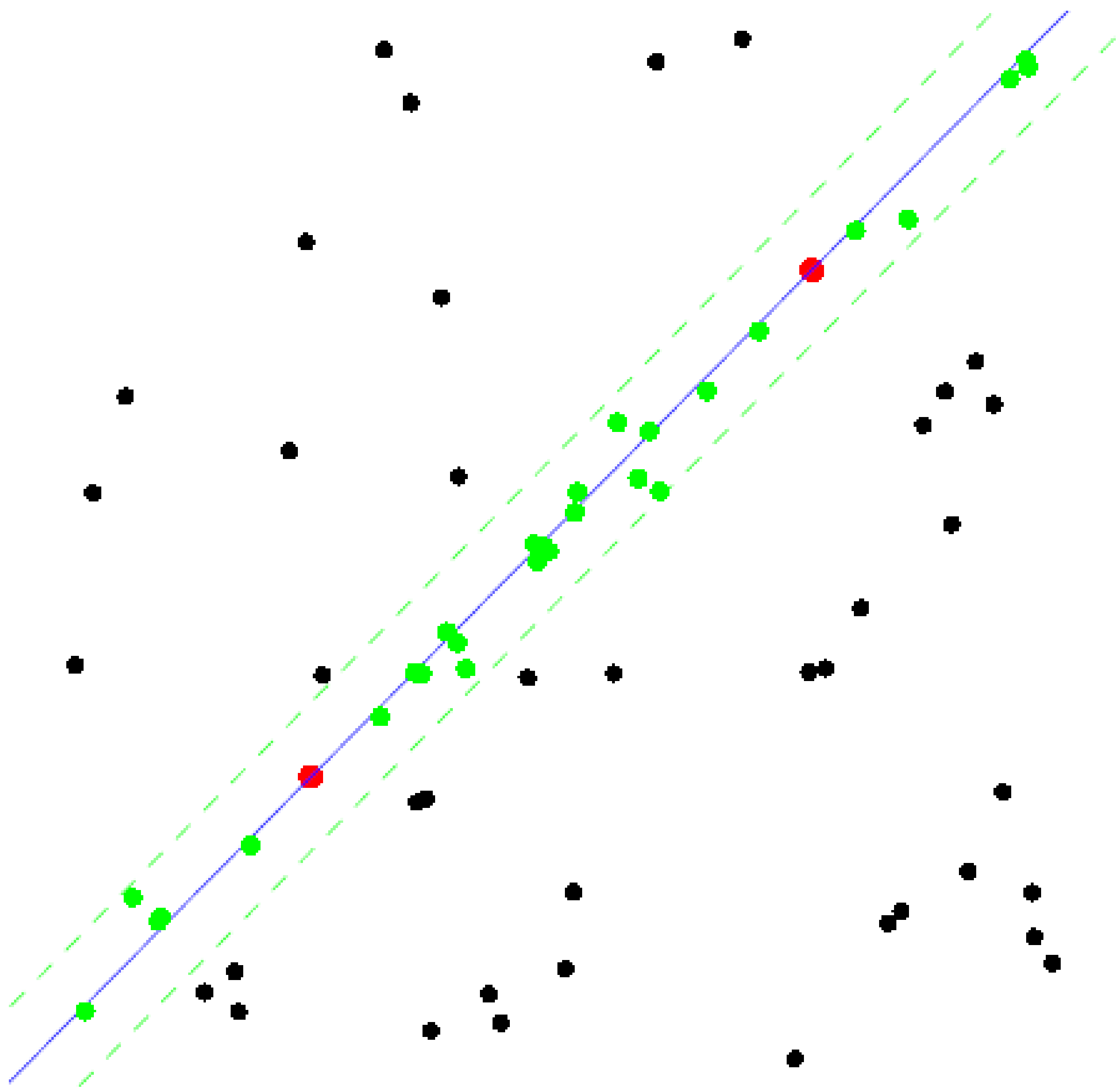
$c=3$

# Another trial



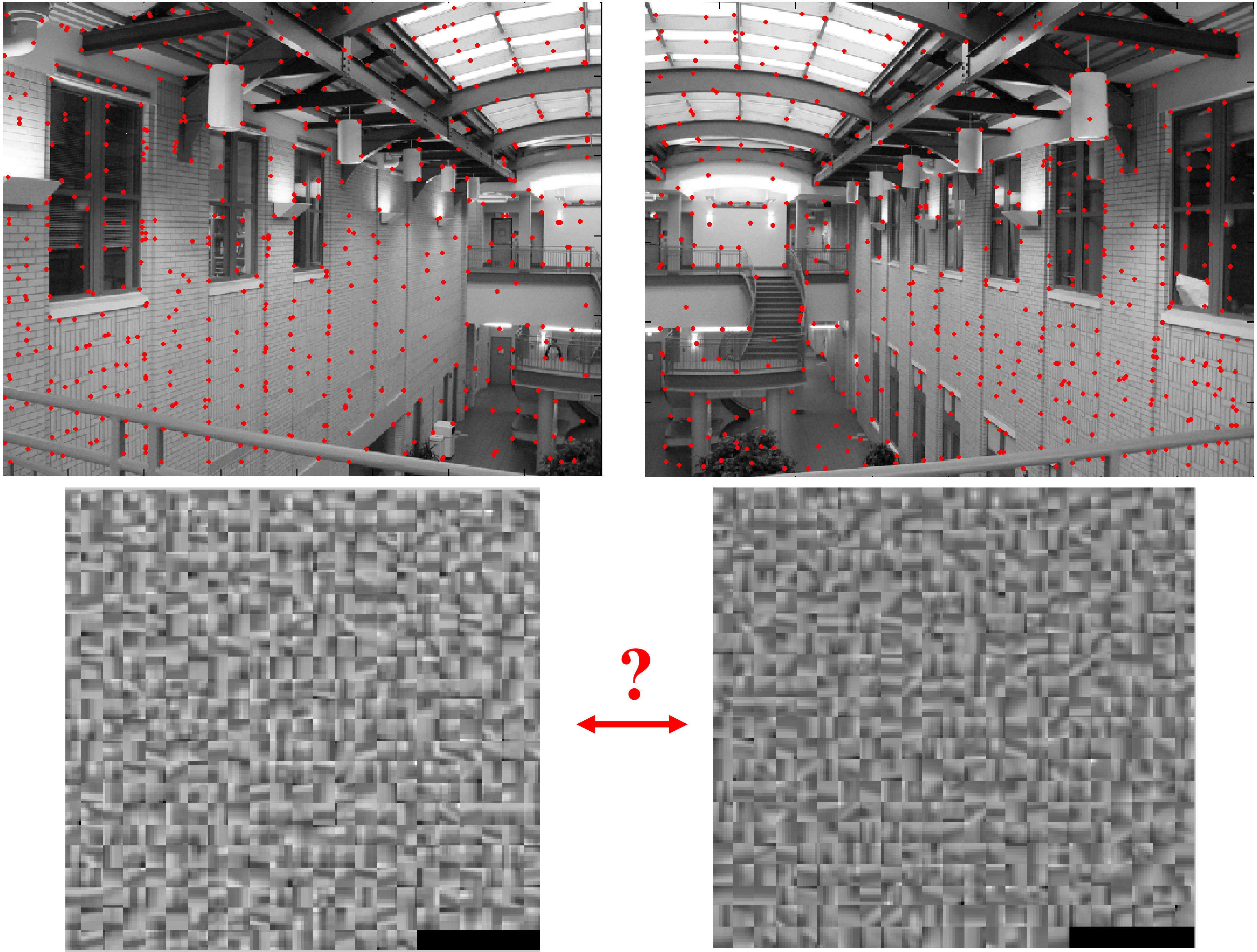
$c=3$

# The best model



$c=15$

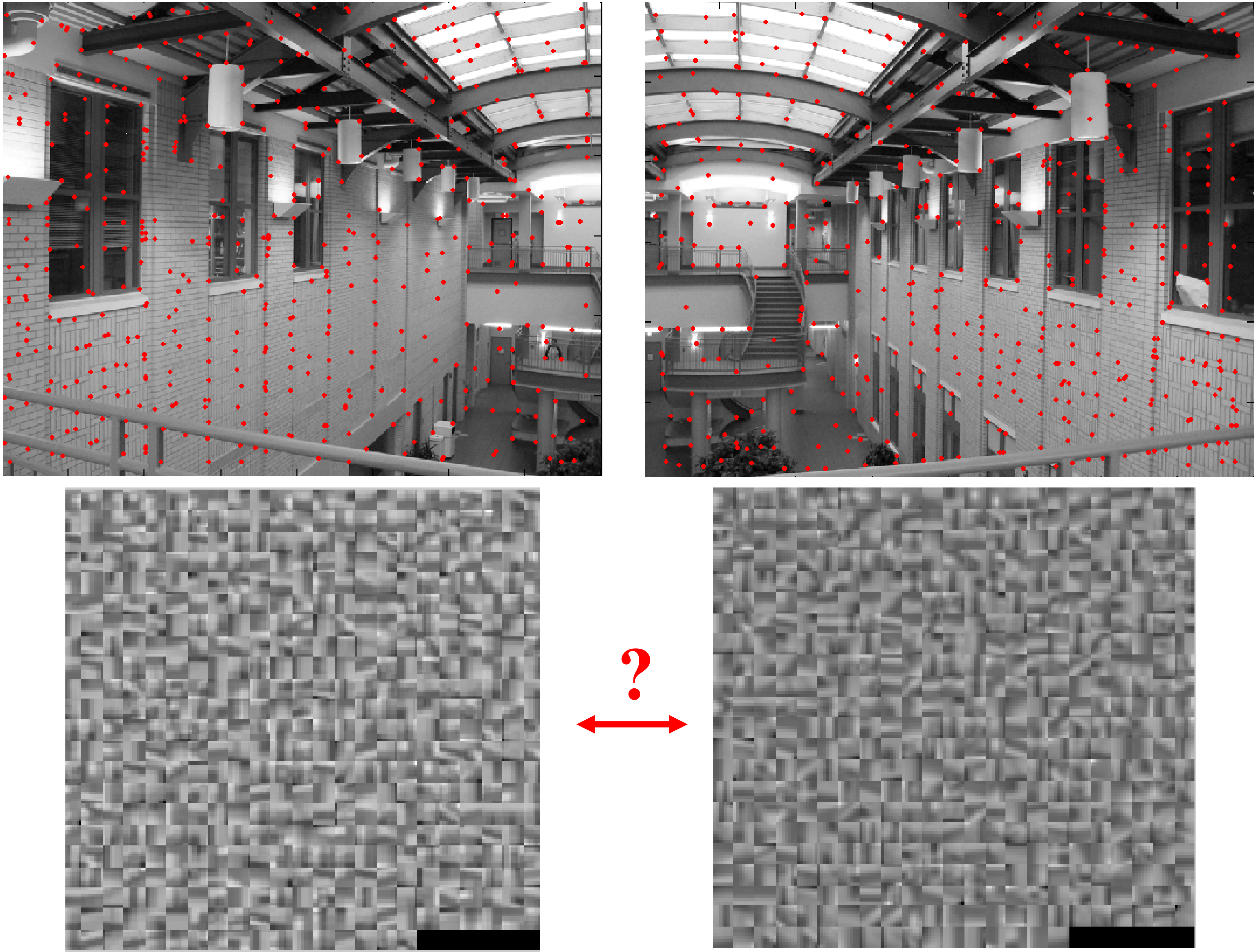
# Feature matching



# Feature matching

- Exhaustive search
  - for each feature in one image, look at *all* the other features in the other image(s)
- Hashing
  - compute a short descriptor from each feature vector, or hash longer descriptors (randomly)
- Nearest neighbor techniques
  - *k*-trees and their variants

# What about outliers?

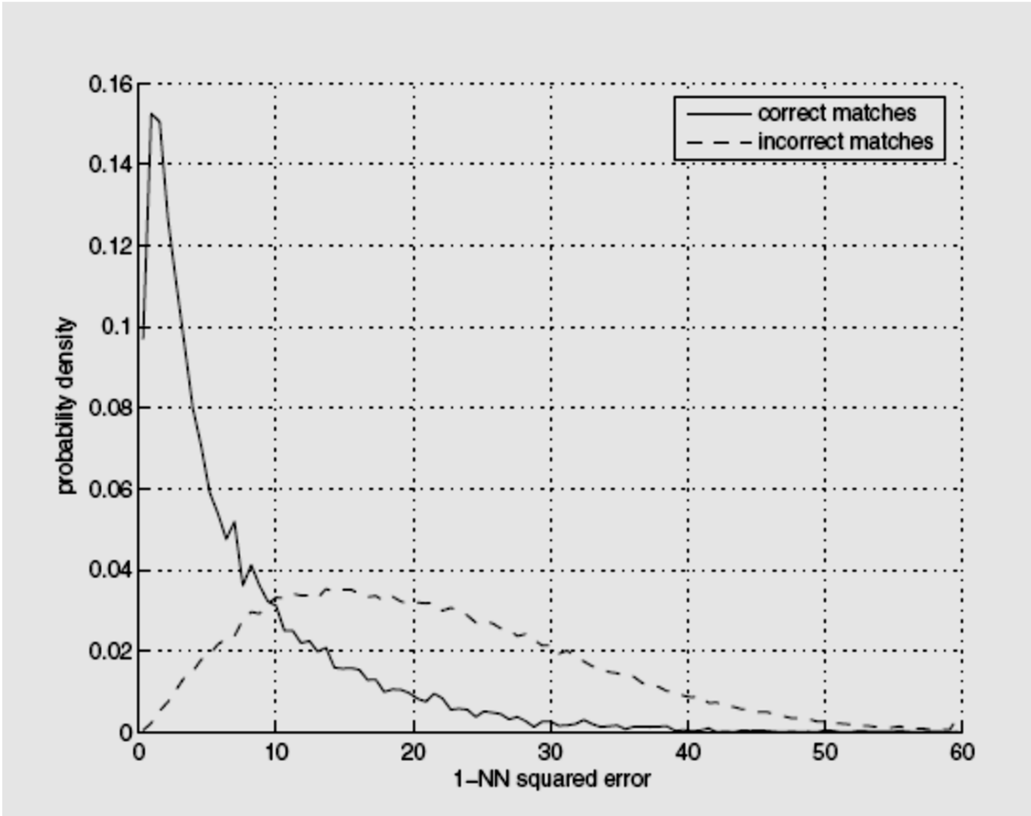


# Feature-space outlier rejection

Let's not match all features, but only these that have "similar enough" matches?

How can we do it?

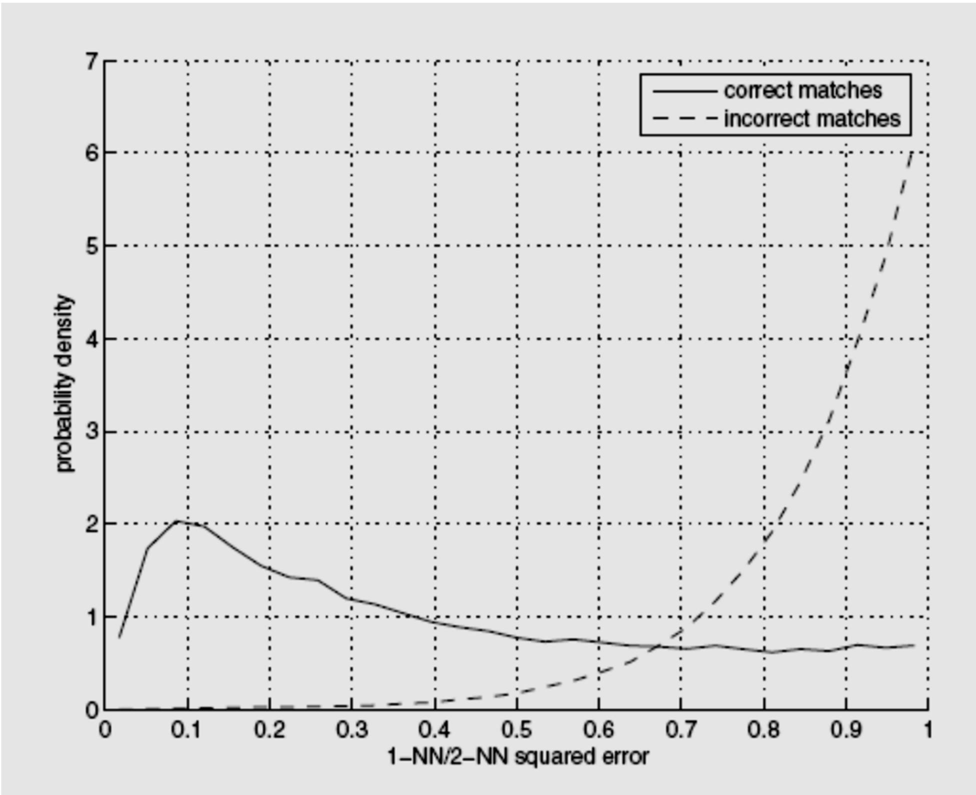
- SSD(patch1,patch2) < threshold
- How to set threshold?



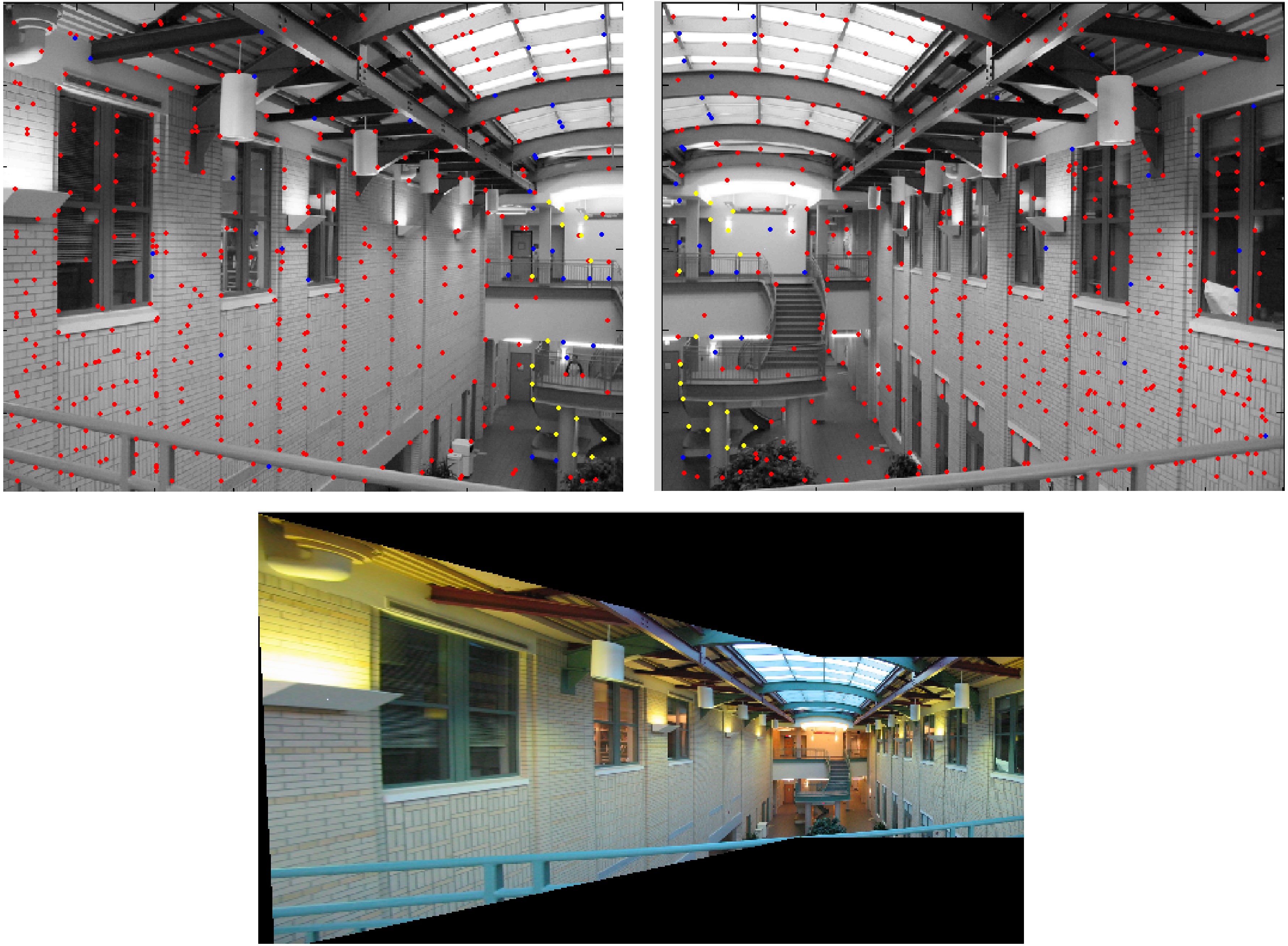
# Feature-space outlier rejection

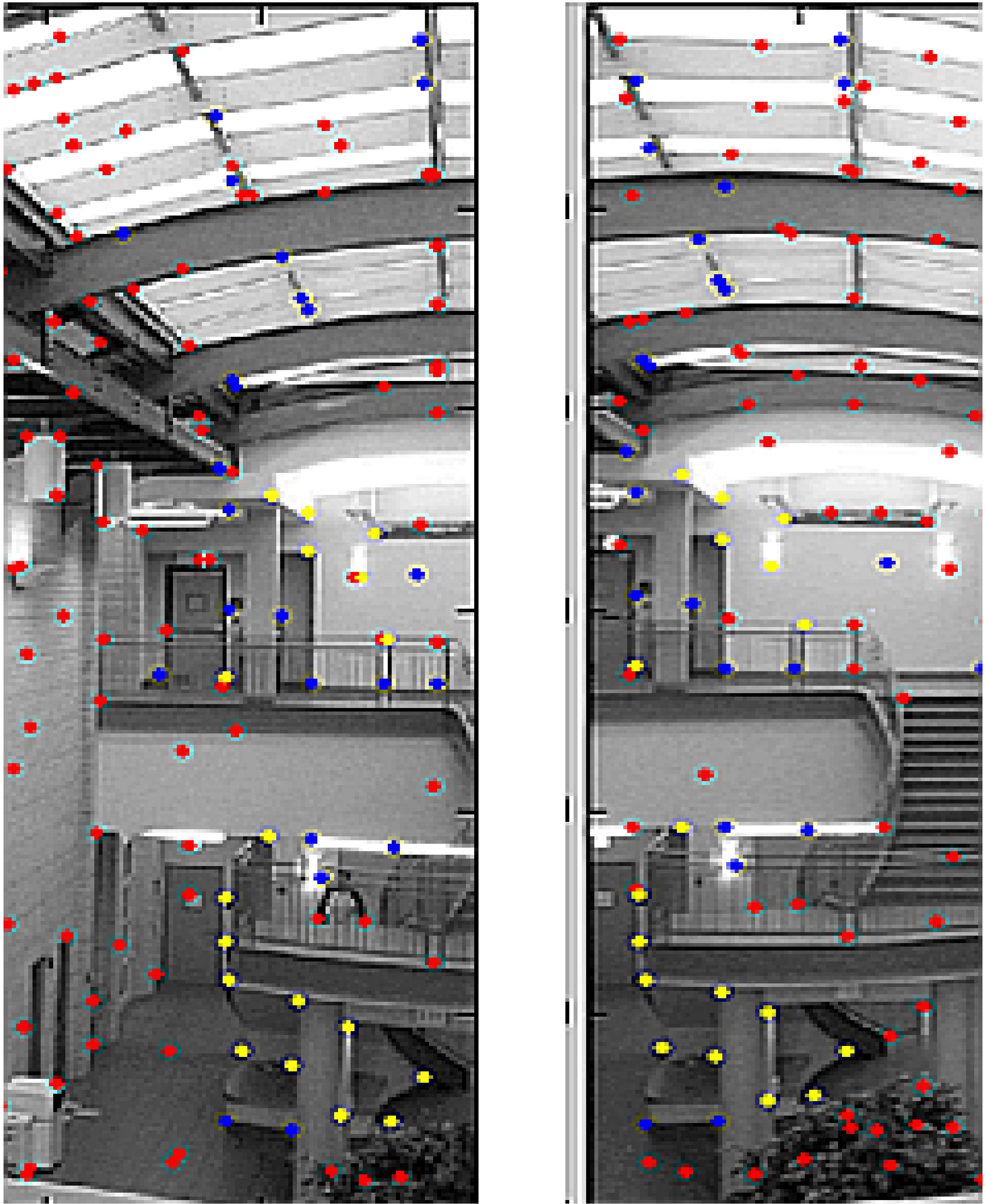
A better way [Lowe, 1999]:

- 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?

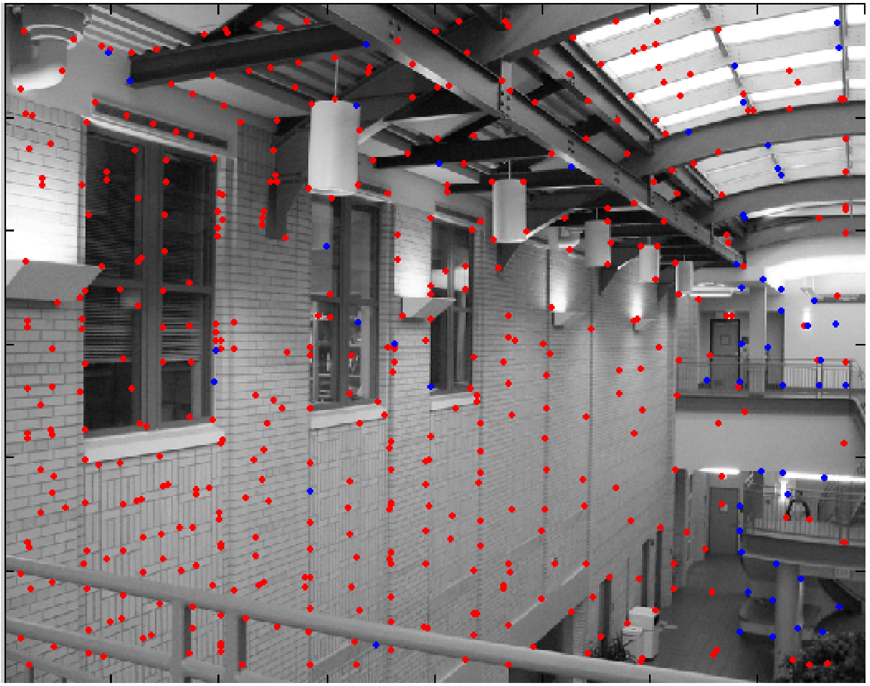


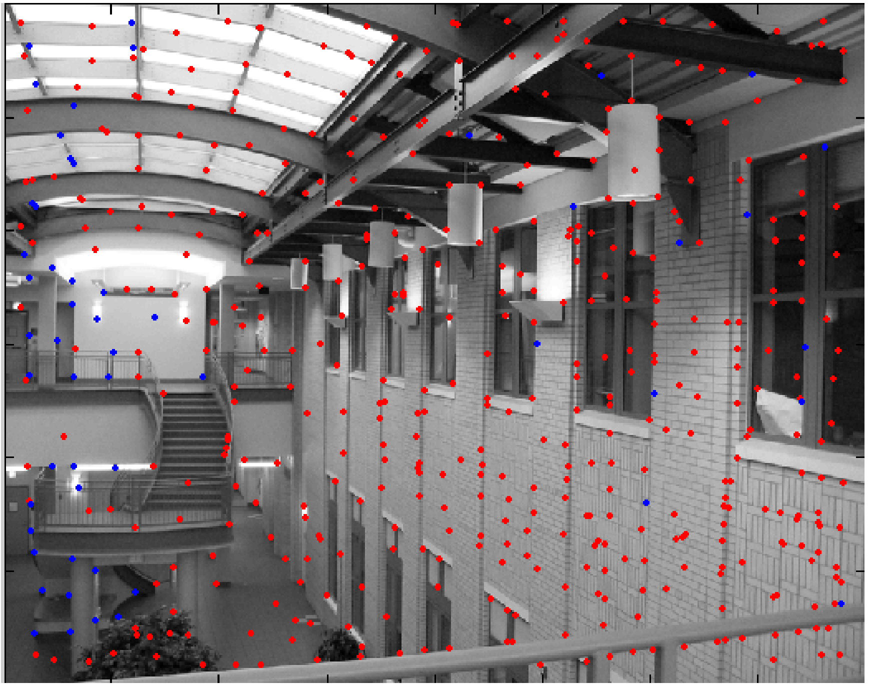
# RANSAC





# Feature-space outliner rejection

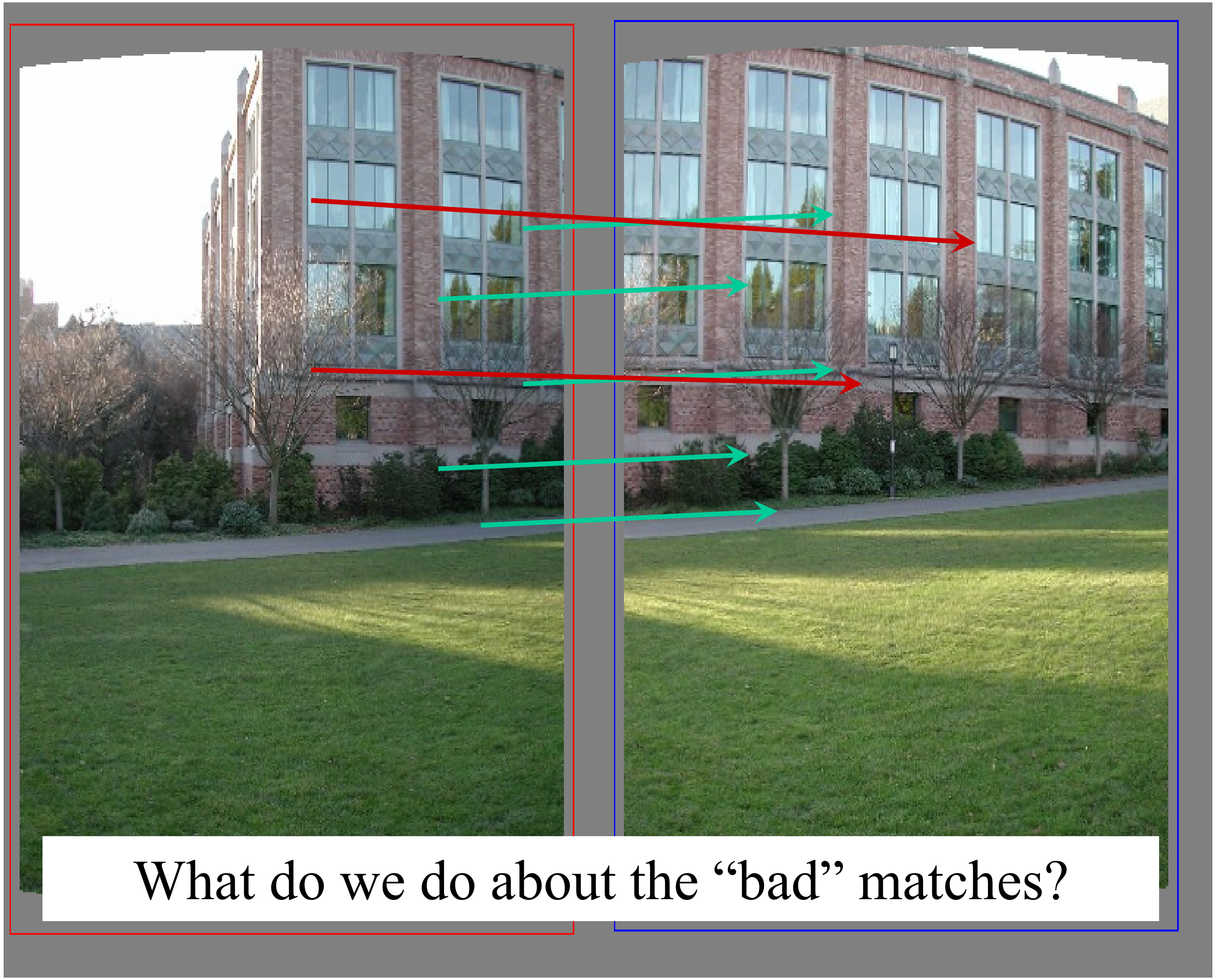




Can we now compute H from the blue points?

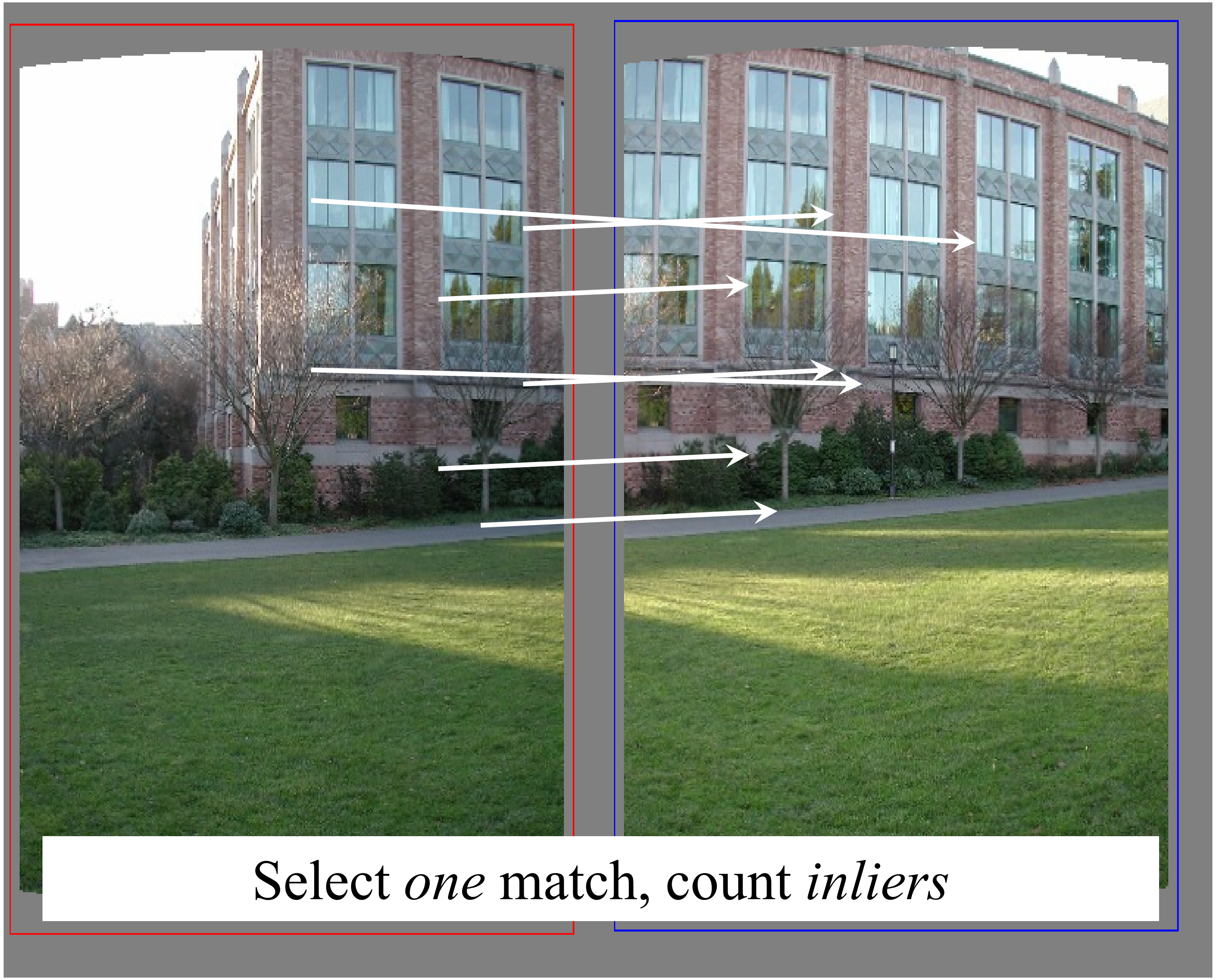
- No!  Still too many outliers…
- What can we do?

# Matching features

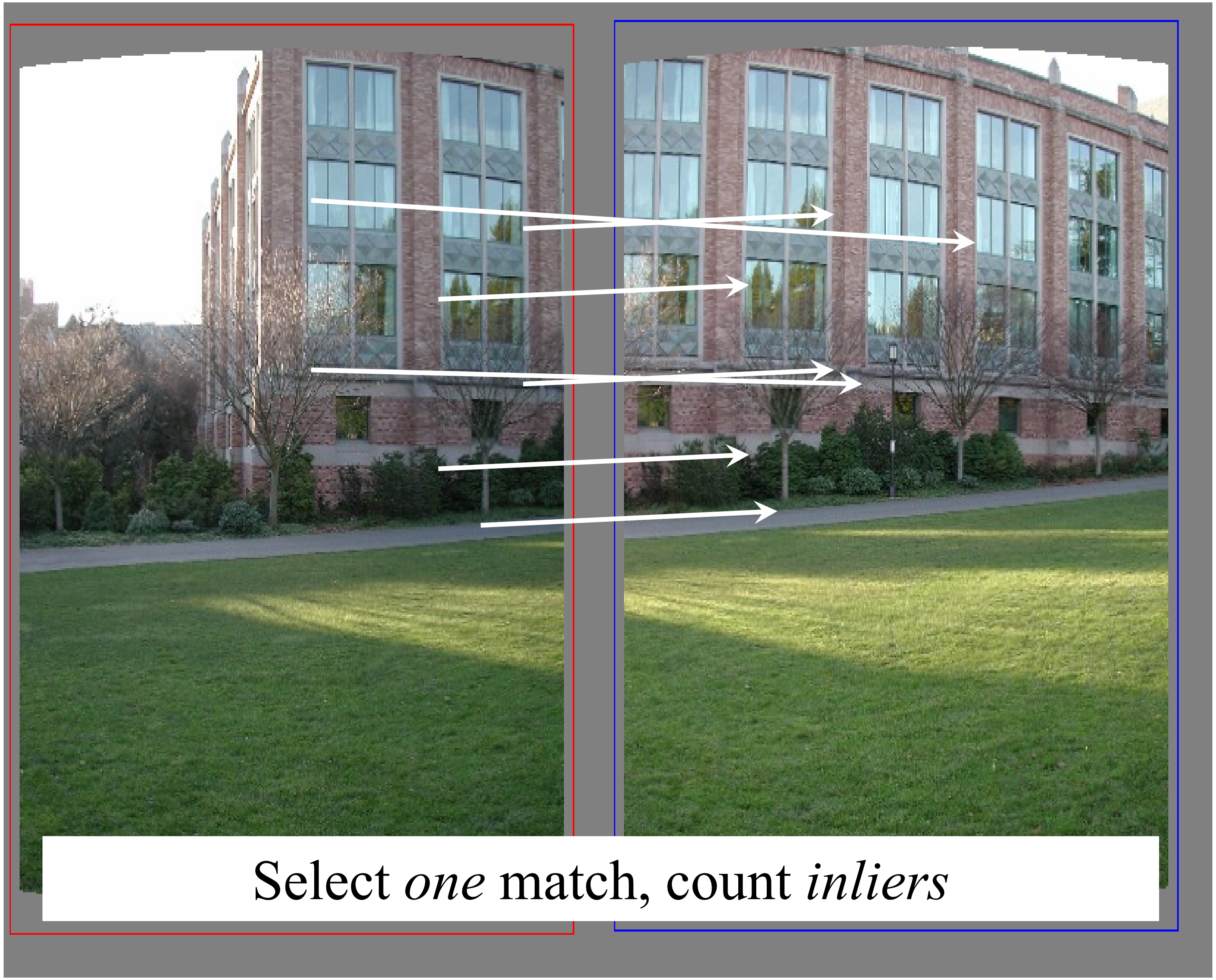


What do we do about the “bad” matches?

# RAndom SAmple Consensus

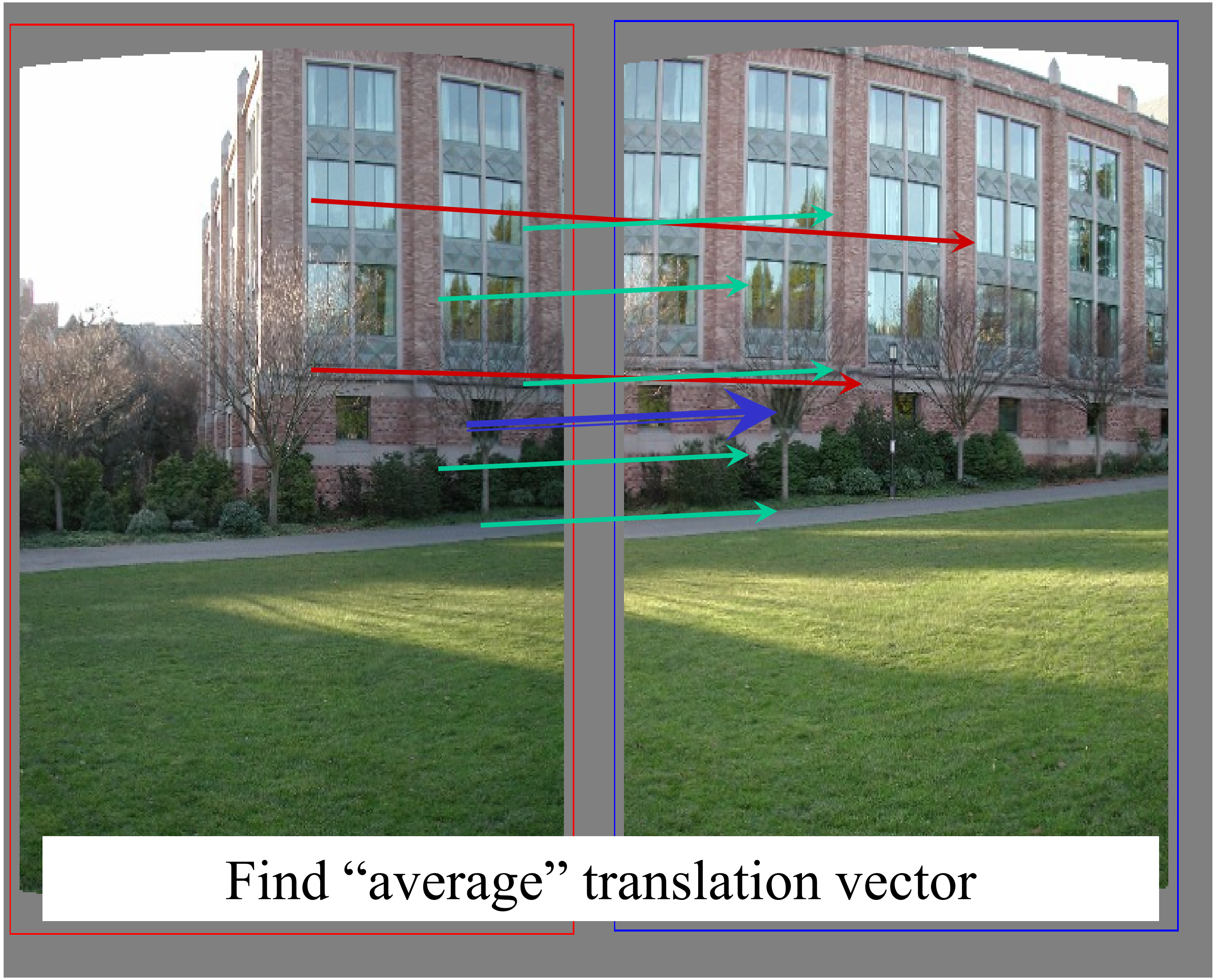


# RAndom SAmple Consensus



Select *one* match, count *inliers*

# Least squares fit



# RANSAC for estimating homography

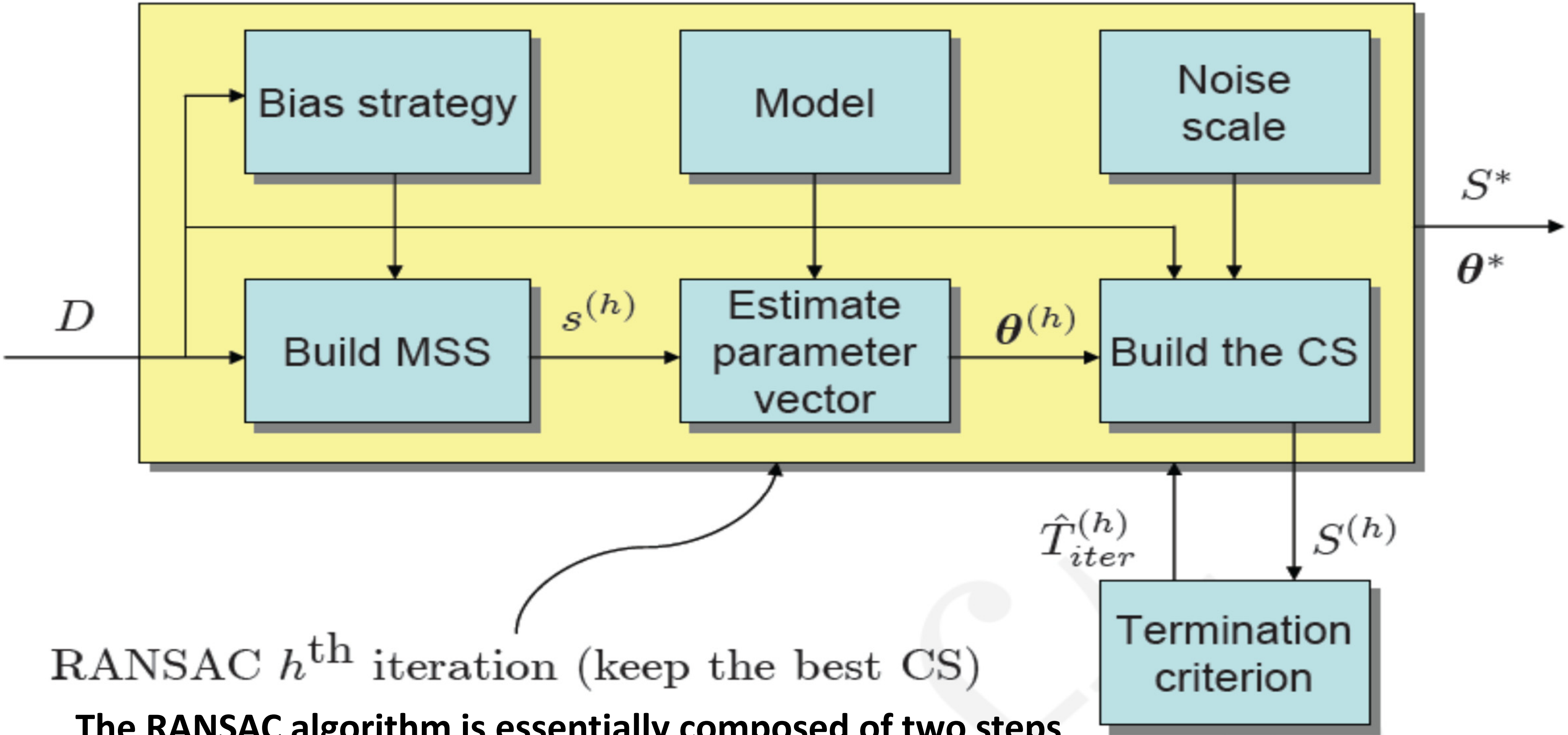
- RANSAC loop:

1. Select four feature pairs (at random)
2. Compute homography H (exact – DLT ?)
3. Compute *inliers* where $SSD(p_i', \boldsymbol{H}\, p_i) < \varepsilon$
4. Record the largest set of inliers so far
5. Re-compute least-squares H estimate on the largest set of the inliers

# RANSAC in general

- RANSAC = Random Sample Consensus
- an algorithm for robust fitting of models in the presence of many data outliers
- Compare to robust statistics

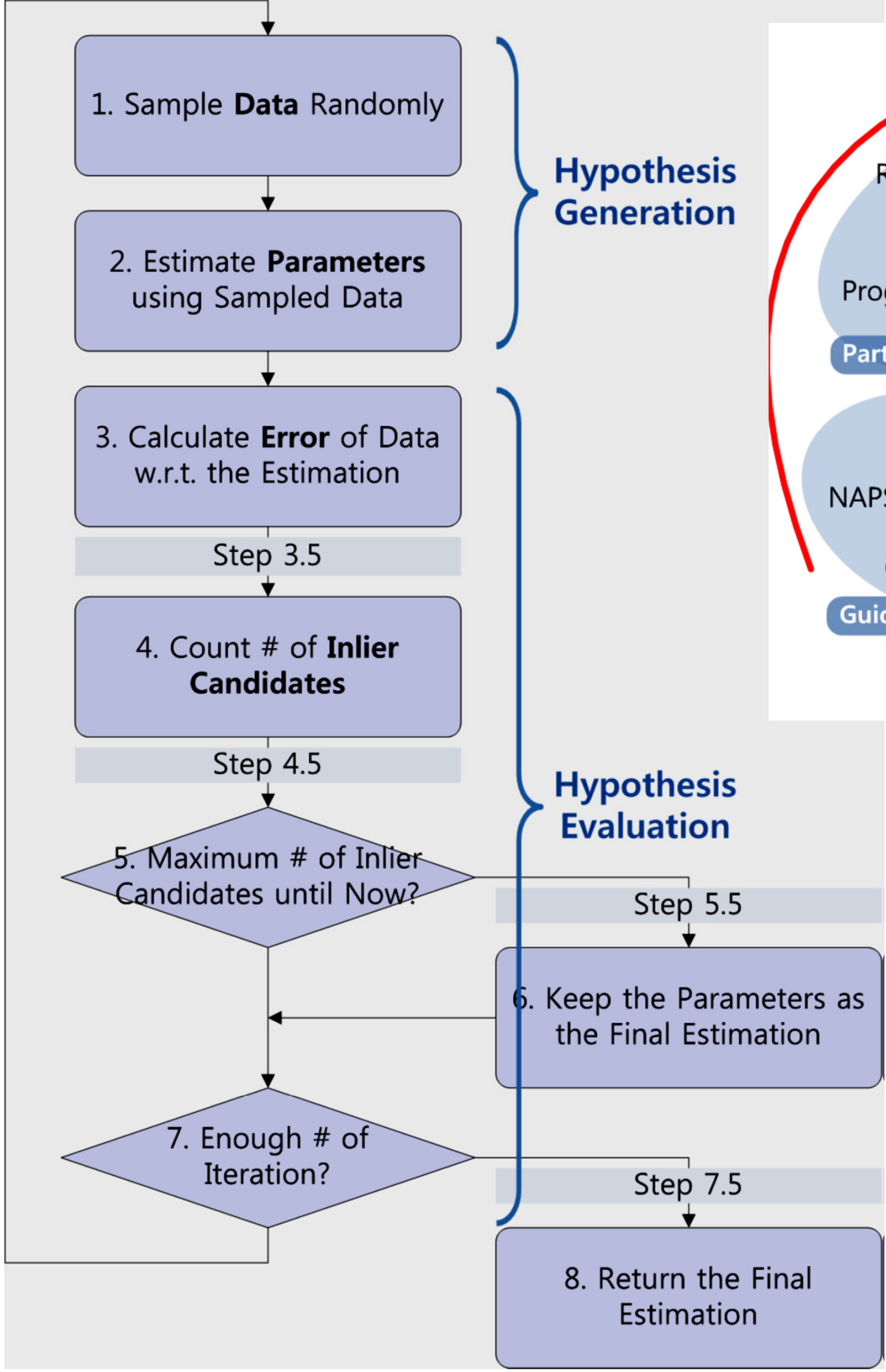
- Given $N$ data points $x_i$, assume that majority of them are generated from a model with parameters $\Theta$, try to recover $\Theta$.

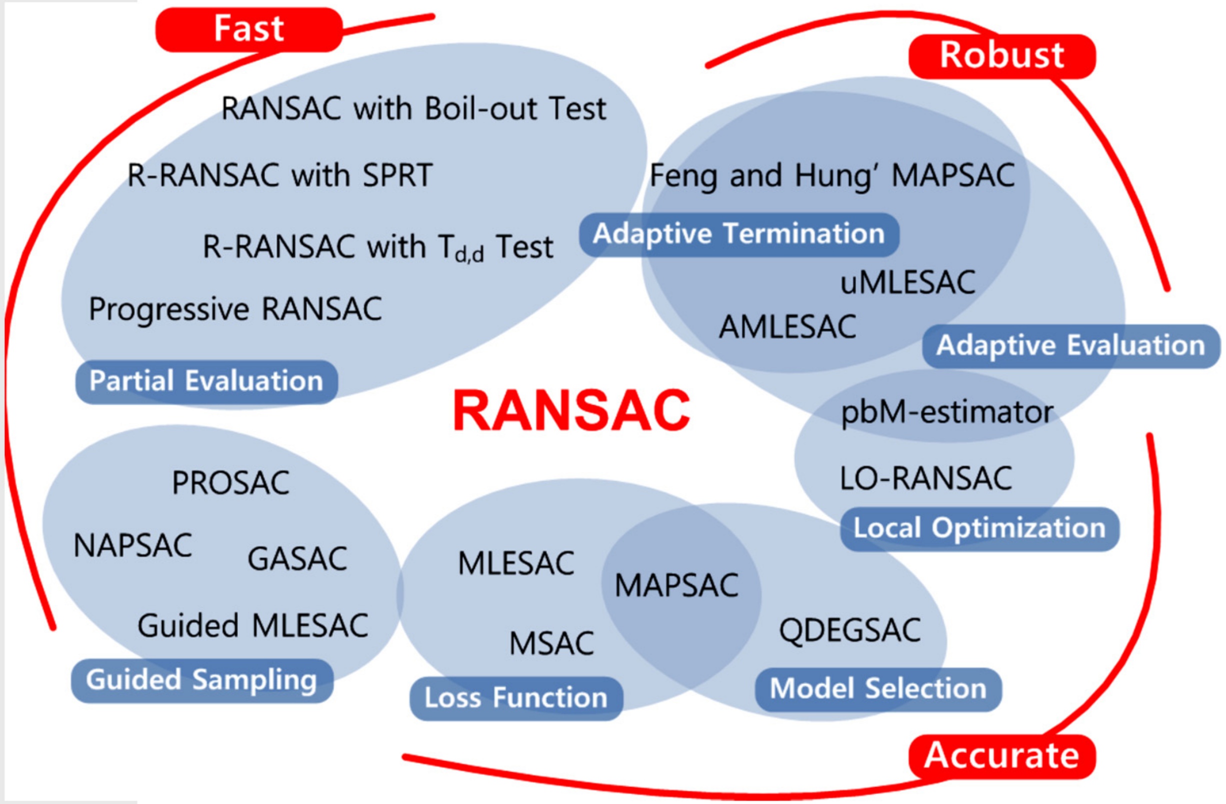


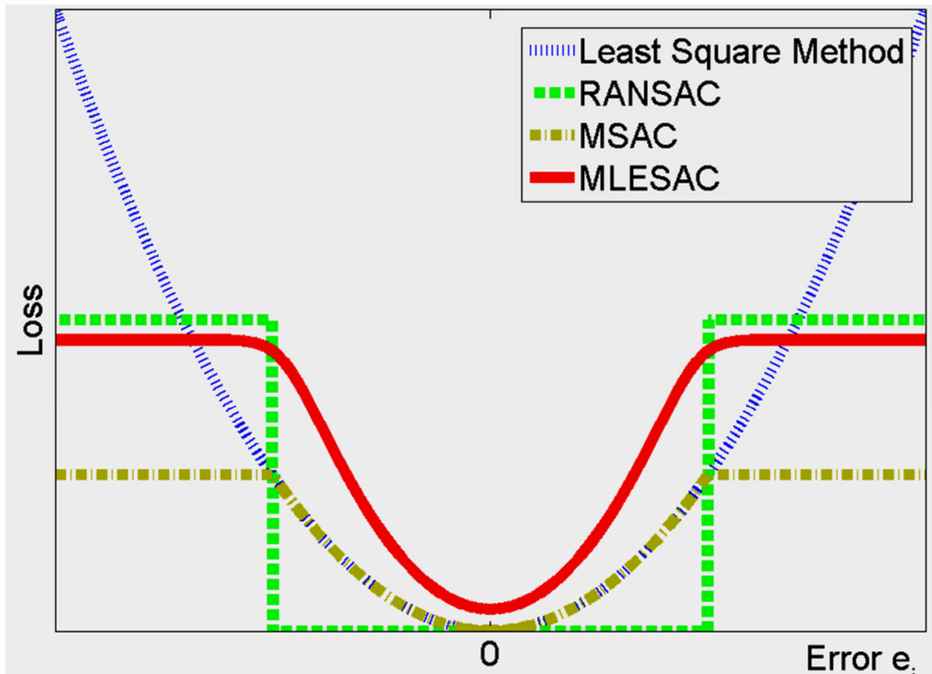

The RANSAC algorithm is essentially composed of two steps that are repeated in an iterative fashion (hypothesize{and{test framework):

• **Hypothesize.** First **minimal sample sets (MSSs)** are randomly selected from the input dataset and the model parameters are computed using only the elements of the MSS. The cardinality of the MSS is the smallest sufficient to determine the model parameters (as opposed to other approaches, such as least squares, where the parameters are estimated using all the data available, possibly with appropriate weights).

• **Test.** In the second step RANSAC checks which elements of the entire dataset are consistent with the model instantiated with the parameters estimated in the rst step. The set of such elements is called **consensus set (CS).**




**Loss Functions**

RANSAC converts a estimation problem in the continuous domain into a selection problem in the discrete domain. For example, there are 200 points to find a line and least square method uses 2 points. There are ${}_{200}\mathrm{C}_2 = 19,900$ available pairs. The problem is now to select the most suitable pair among huge number of pairs.

## 2.2 Hypothesis Evaluation

RANSAC finally chooses the most probable hypothesis, which is supported by the most inlier candidates (Step 5 and 6). A datum is recognized as the inlier candidate, whose error from a hypothesis is within a predefined threshold (Step 4). In case of line fitting, error can be geometric distance from the datum to the estimated line. The threshold is the second tuning variable, which is highly related with magnitude of noise which contaminates inliers (shortly *the magnitude of noise*). However, the magnitude of noise is also unknown in almost all application.

RANSAC solves the selection problem as an optimization problem. It is formulated as

$$\hat{M} = \arg\min_{M} \left\{ \sum_{d \in \mathscr{D}} \mathrm{Loss}\Big(\mathrm{Err}(d;M)\Big) \right\}, \tag{2}$$

where $\mathscr{D}$ is data, Loss is a loss function, and Err is a error function such as geometric distance. The loss function of least square method is represented as $\mathrm{Loss}(e) = e^2$. In contrast, RANSAC uses

$$\mathrm{Loss}(e) = \begin{cases} 0 & |\, e \,| < c \\ \mathrm{const} & \text{otherwise} \end{cases}, \tag{3}$$

where $c$ is the threshold. Figure 3 shows difference of two loss functions. RANSAC has constant loss at large error while least square method has huge loss. Outliers disturb least squares because they usually have large error.

# RANSAC algorithm

Run $k$ times: How many times?

(1) draw $n$ samples randomly How big? Smaller is better

(2) fit parameters $\Theta$ with these $n$ samples

(3) for each of other $N$-$n$ points, calculate its distance to the fitted model, count the number of inlier points, $c$

Output $\Theta$ with the largest $c$

How to define? Depends on the problem.

# How to determine k

$n$: number of samples drawn each iteration

$p$: probability of real inliers

$P$: probability of at least 1 success after k trials

$$P = 1 - \underbrace{\underbrace{(1 - \underbrace{p^n}_{\text{n samples are all inliers}})}_{\text{a failure}}{}^k}_{\text{failure after k trials}}$$

$$k = \frac{\log(1-P)}{\log(1-p^n)}$$

for $P=0.99$

| $n$ | $p$ | $k$ |
|---|---|---|
| 3 | 0.5 | 35 |
| 6 | 0.6 | 97 |
| 6 | 0.5 | 293 |

# RANSAC Method for computing F:

**(i) Interest points: Compute interest points in each image.**

**(ii) Putative correspondences: Compute a set of interest point matches based on proximity and similarity of their intensity neighbourhood;**

**(iii) RANSAC robust estimation: Repeat for *N* samples:**

**(a) Select a random sample of 7 (or 8) correspondences and compute the fundamental matrix F (Algebraic Min. or DLT).**

**(b) the solution with most inliers is retained; i.e. Choose the F with the largest number of *inliers*;**

**Repeat the following two steps, until stability:**

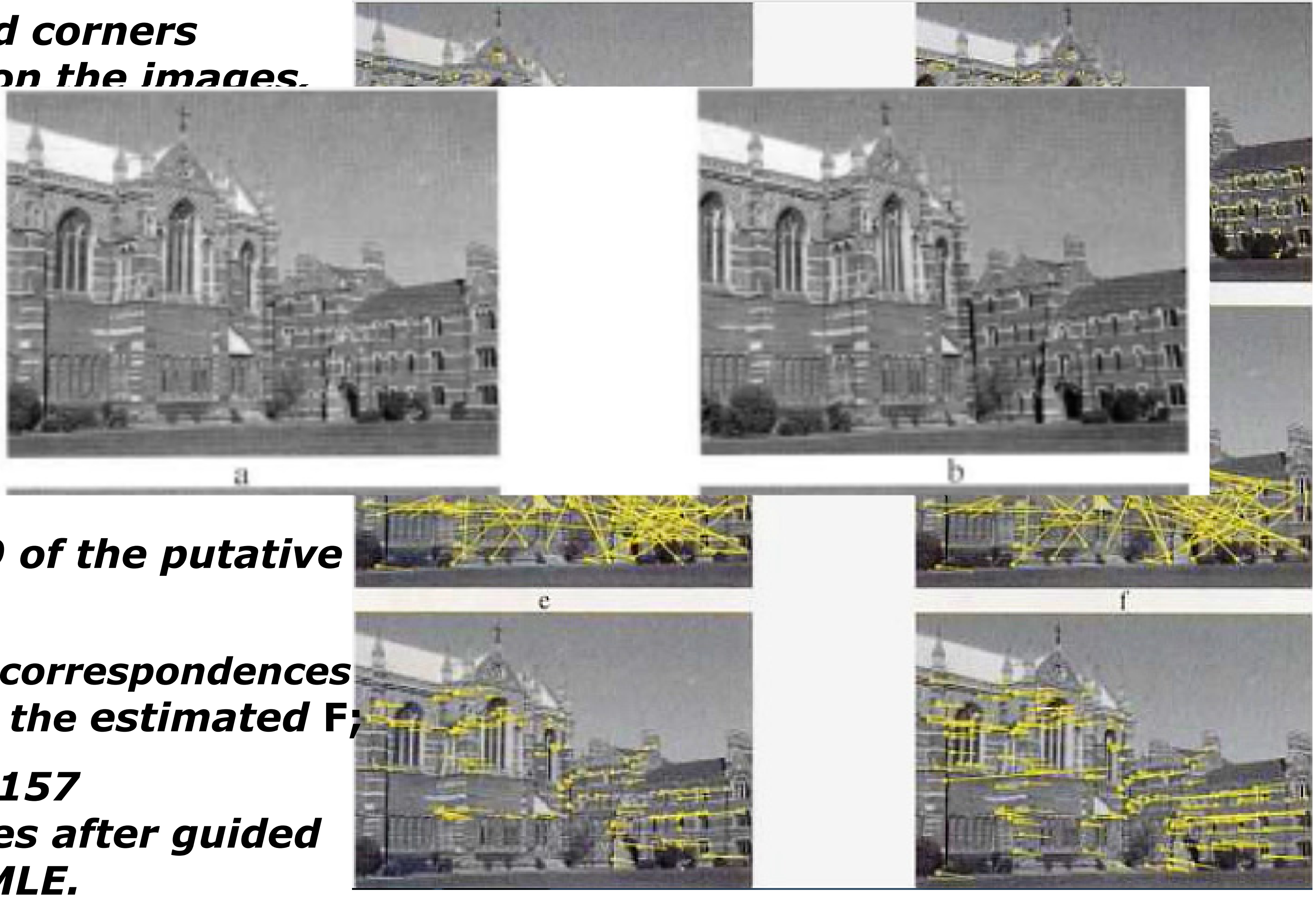
**(iv) Non-linear estimation: re-estimate F from all correspondences classified as *inliers* by minimizing a cost function, using the Levenberg-Marquardt (LM) algorithm.**

**(v) Guided matching: Further interest point correspondences are now determined using the estimated F to define a search strip about the epipolar line.**

**Other methods – Gold-standard (MLE); Sampson Distance (cost) function;**

*(c) (d) detected corners superimposed on the images. There are appr... corners on eac...*

*The following r... superimposed ... image: (e) 188 ... matches shown ... linking corners, ... mismatches;*




*(f) outliers - 89 of the putative matches,*

*(g) inliers - 99 correspondences consistent with the estimated* **F***;*

*(h) final set of 157 correspondences after guided matching and MLE.*

**Both the fundamental and essential matrices could completely describe the geometric relationship between corresponding points of a stereo pair of cameras.**

**The only difference between the two is that the fundamental matrix deals with uncalibrated cameras, while the essential matrix deals with calibrated cameras.**

# Applications

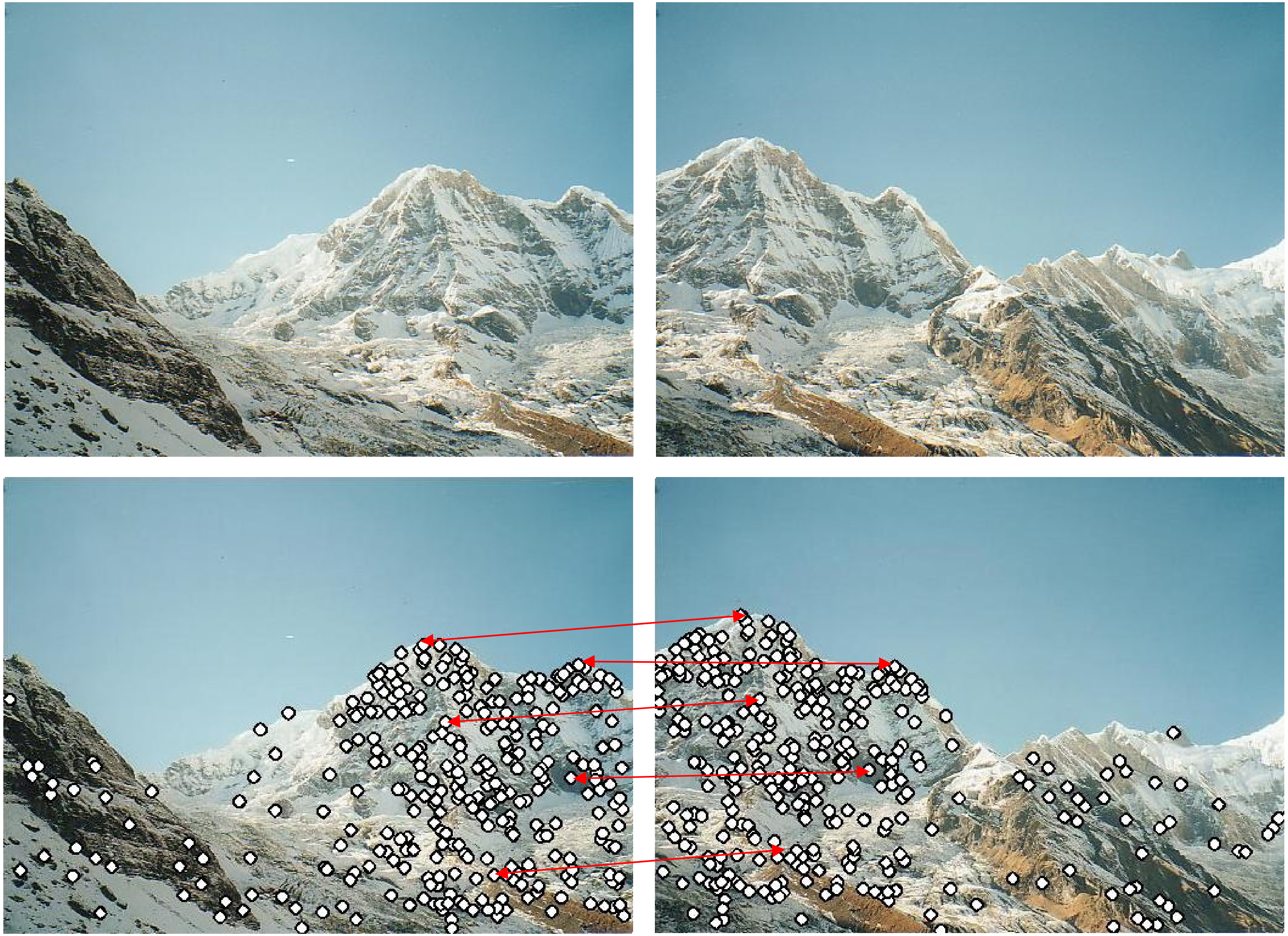
# Feature Matching and RANSAC

© **Krister Parmstrand**

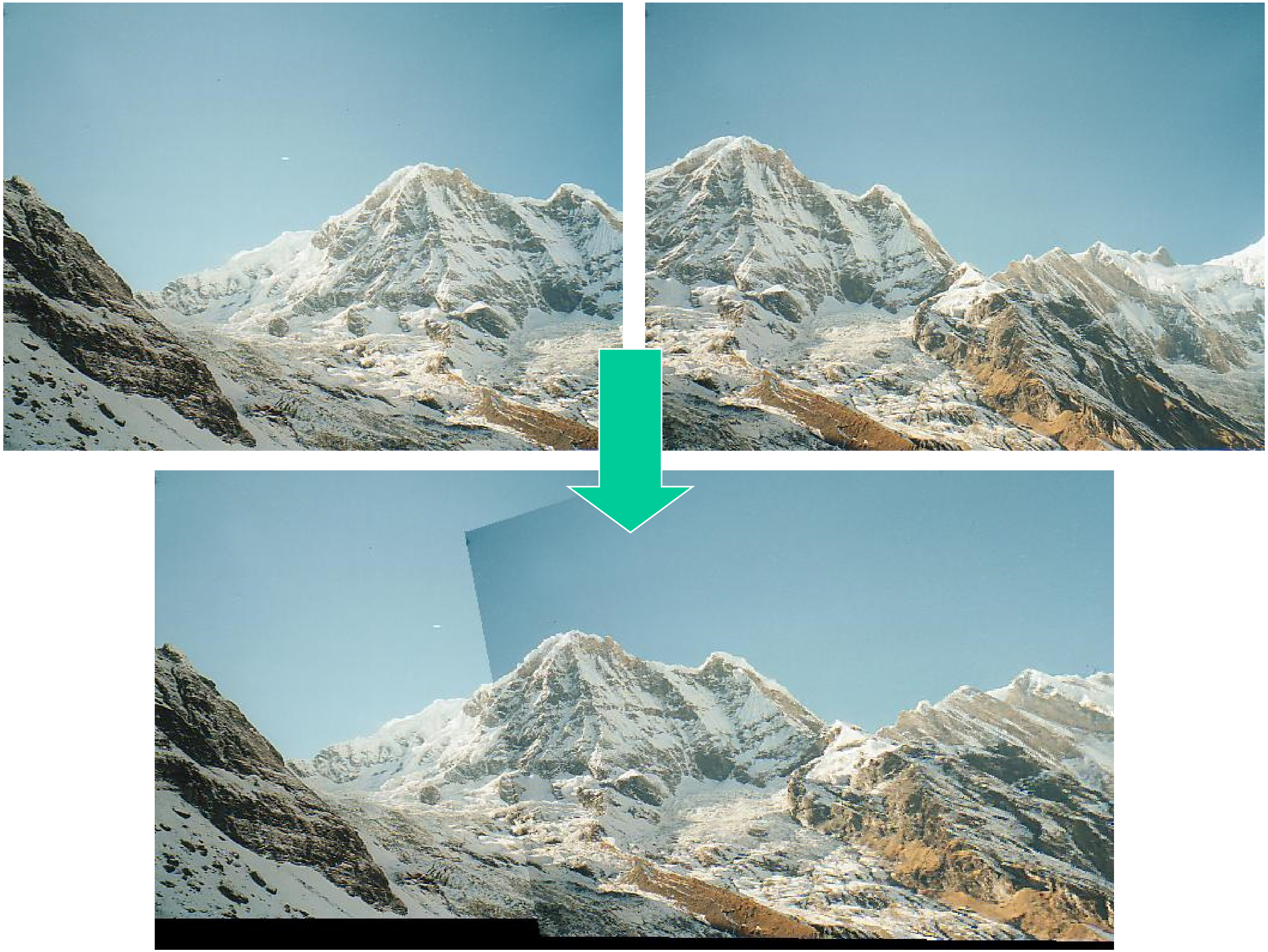
*with a lot of slides stolen from Steve Seitz and Rick Szeliski*

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

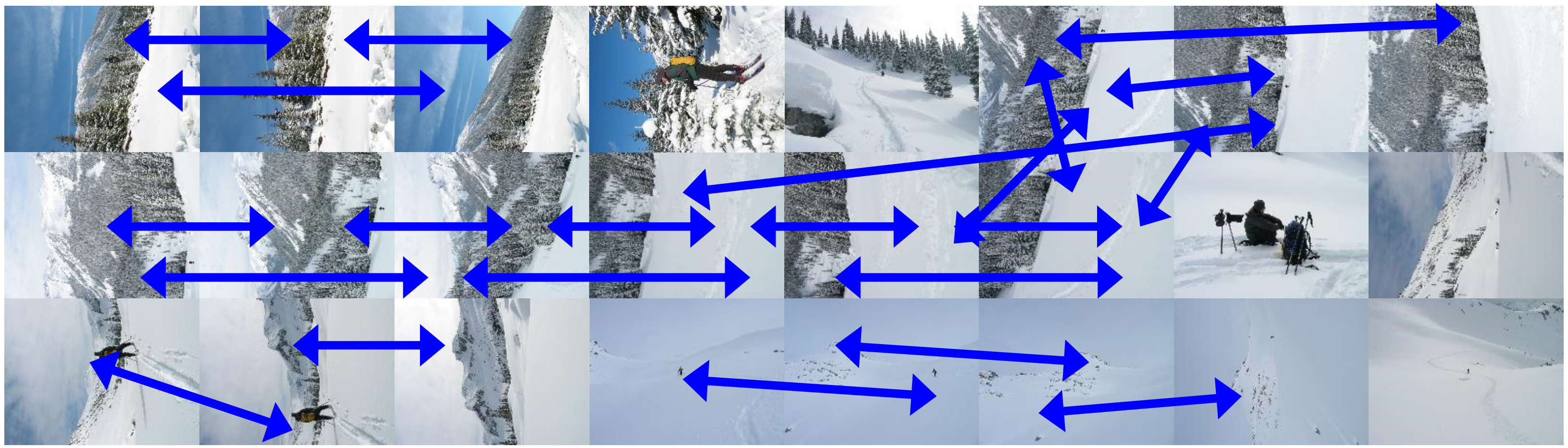
# Automatic image stitching



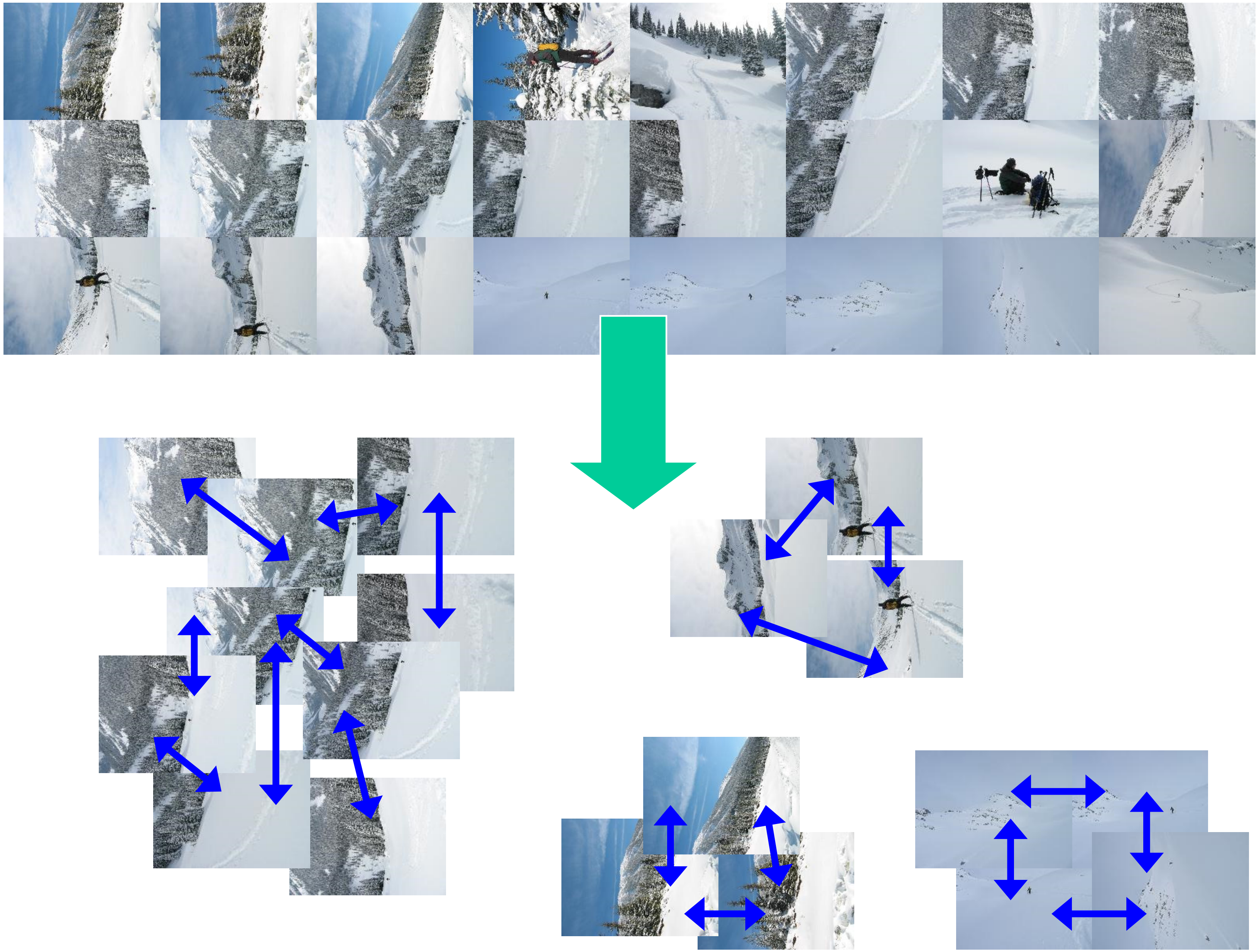
# Automatic image stitching



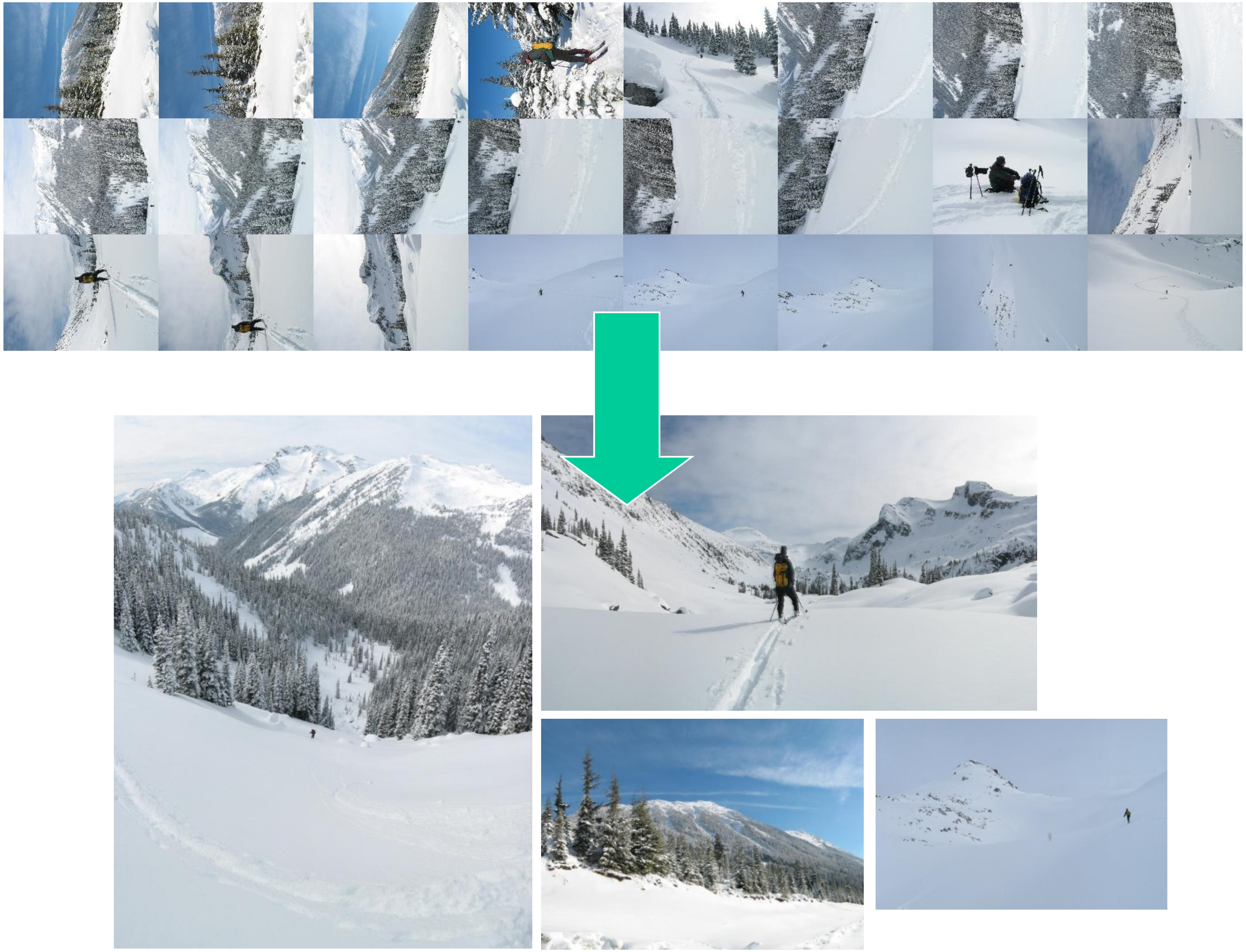
# Automatic image stitching



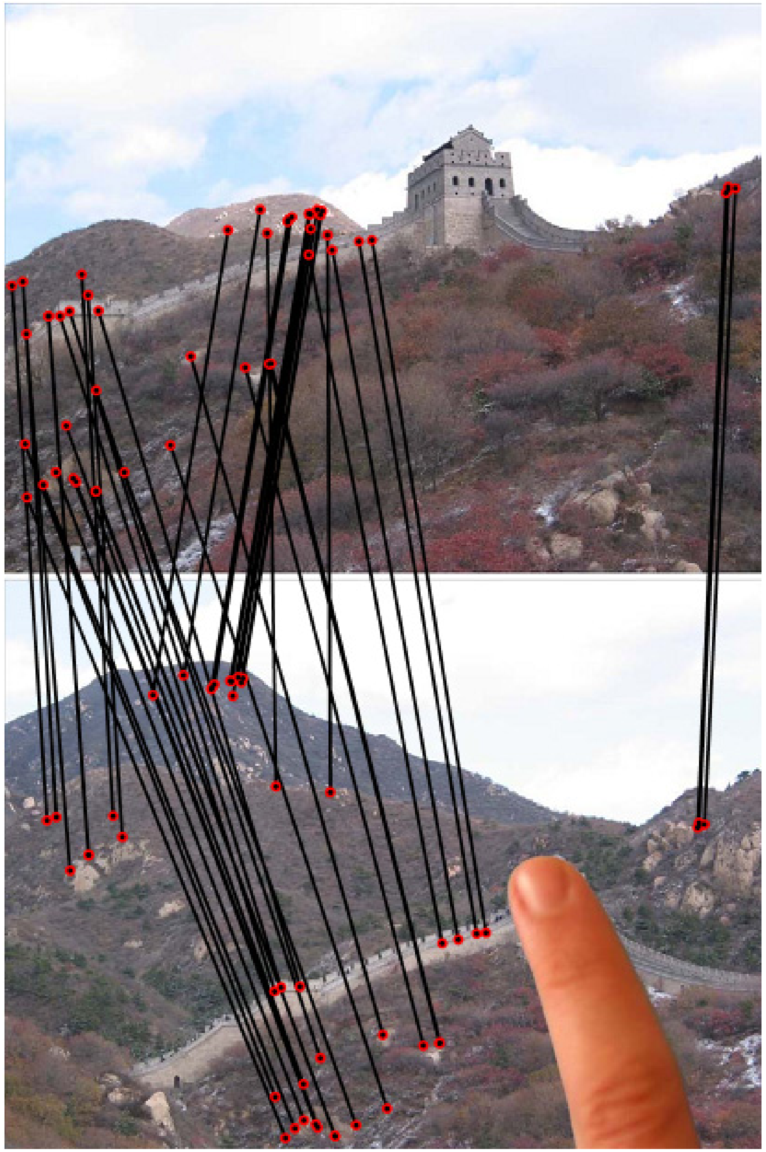
# Automatic image stitching



# Automatic image stitching

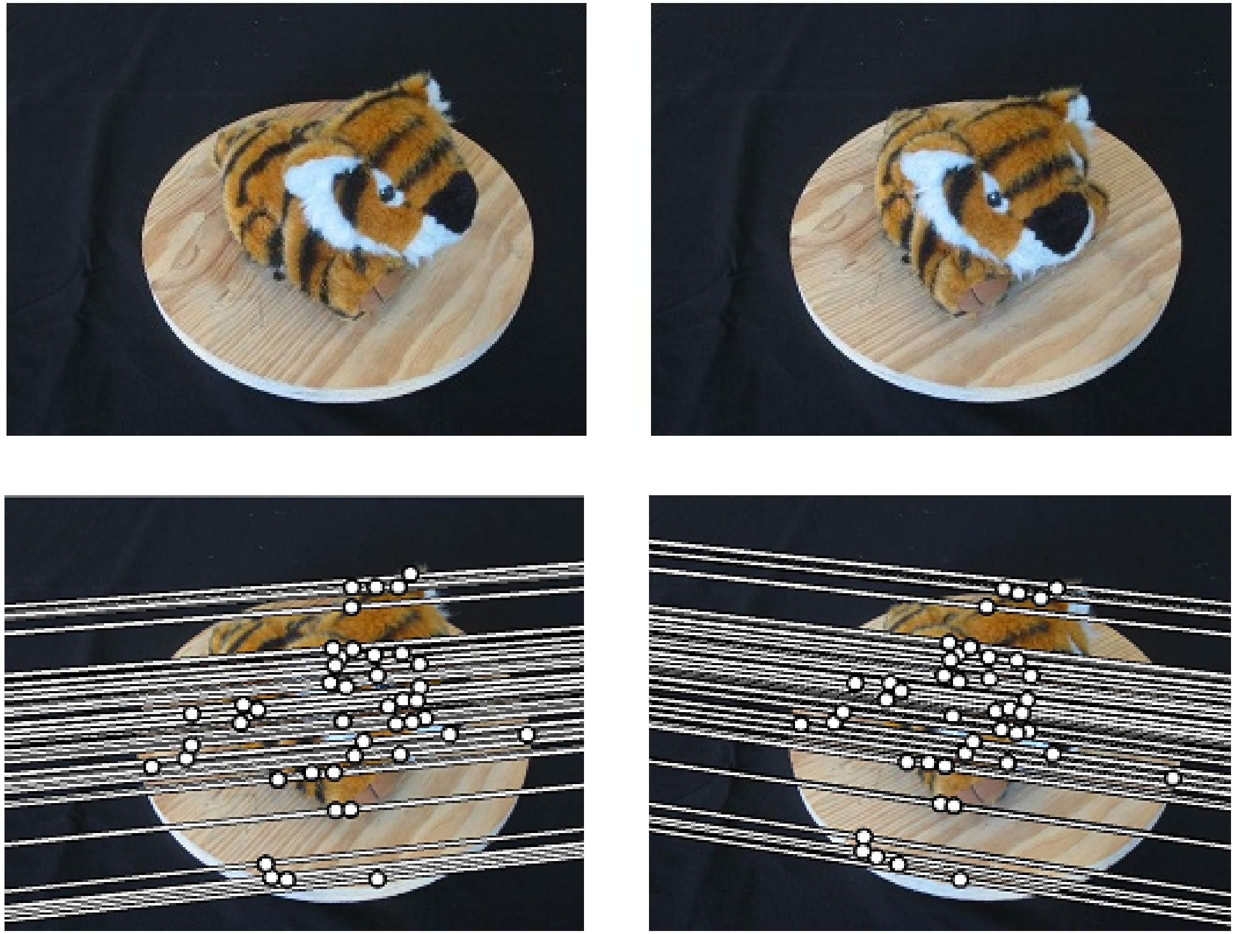


# Correspondence Results



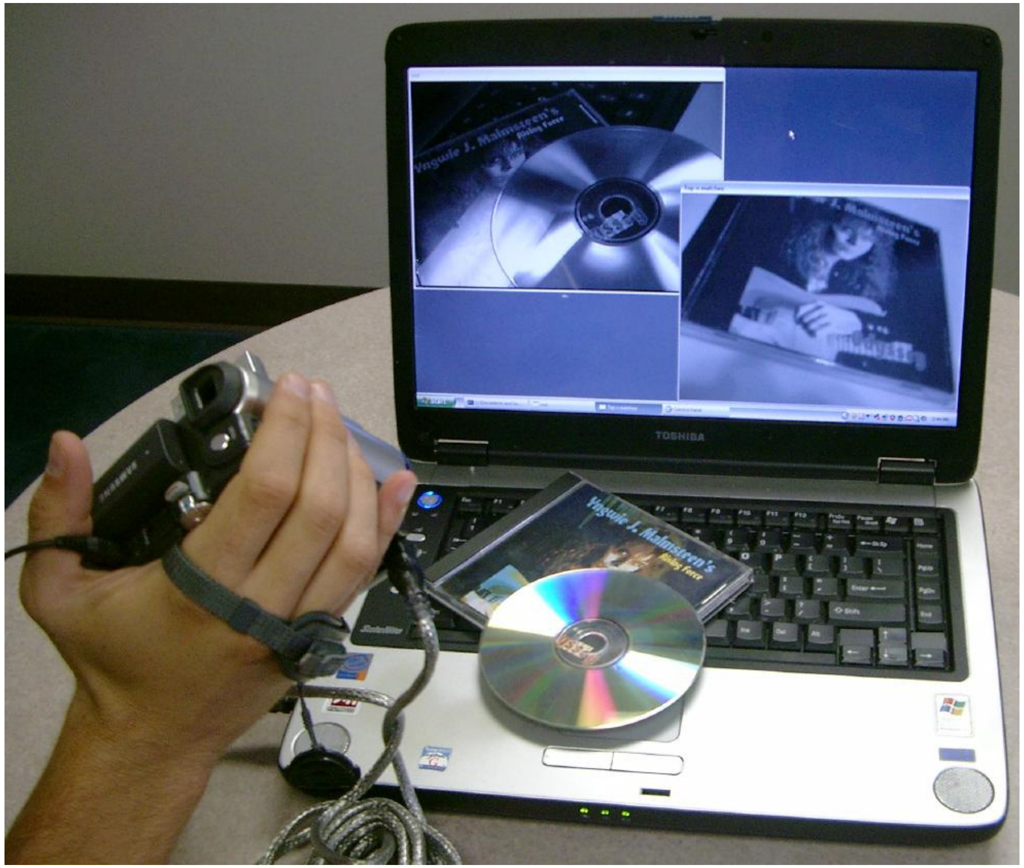
Chum & Matas 2005

# Object Recognition Results



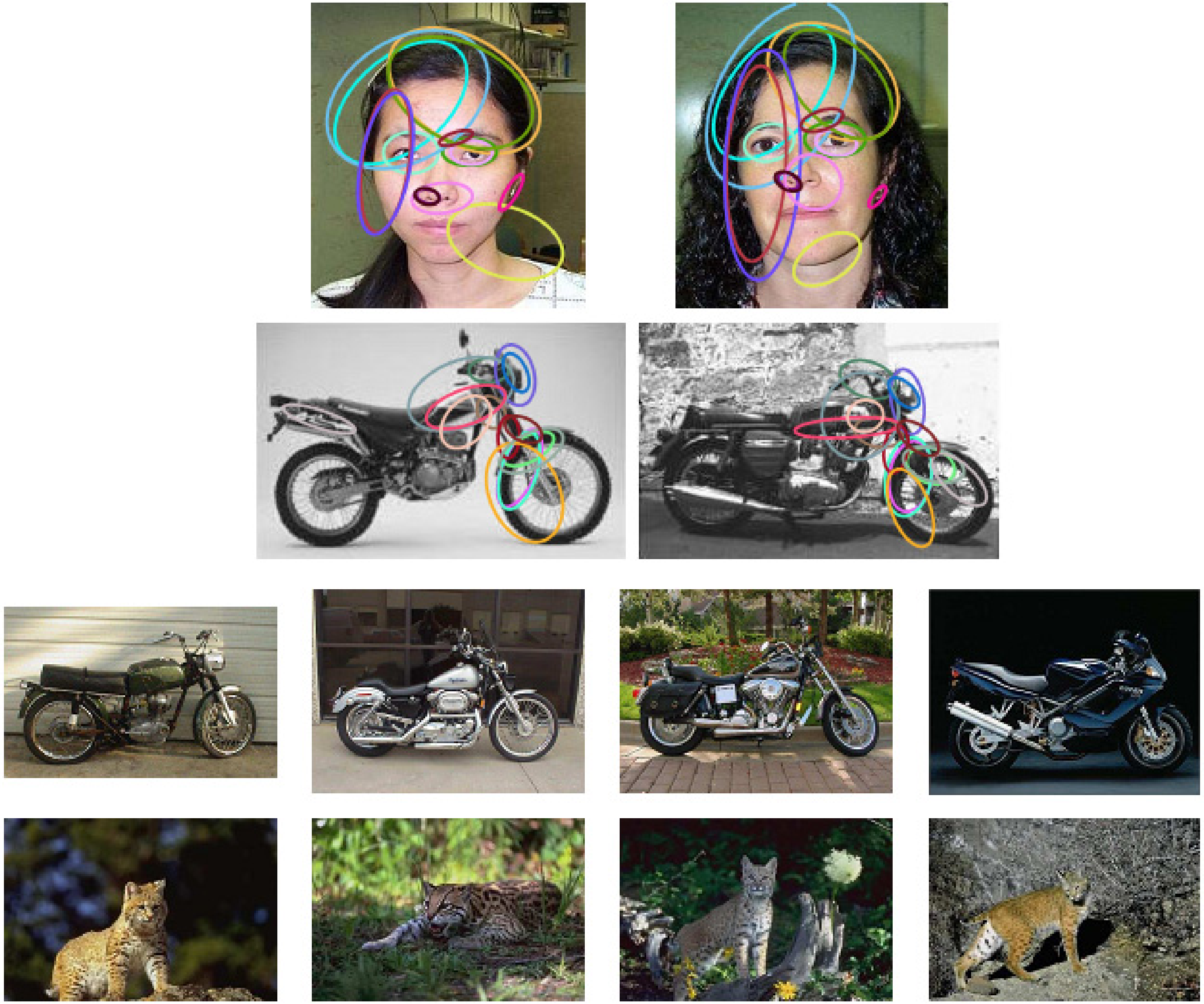
Brown & Lowe 2005

# Object Recognition Results



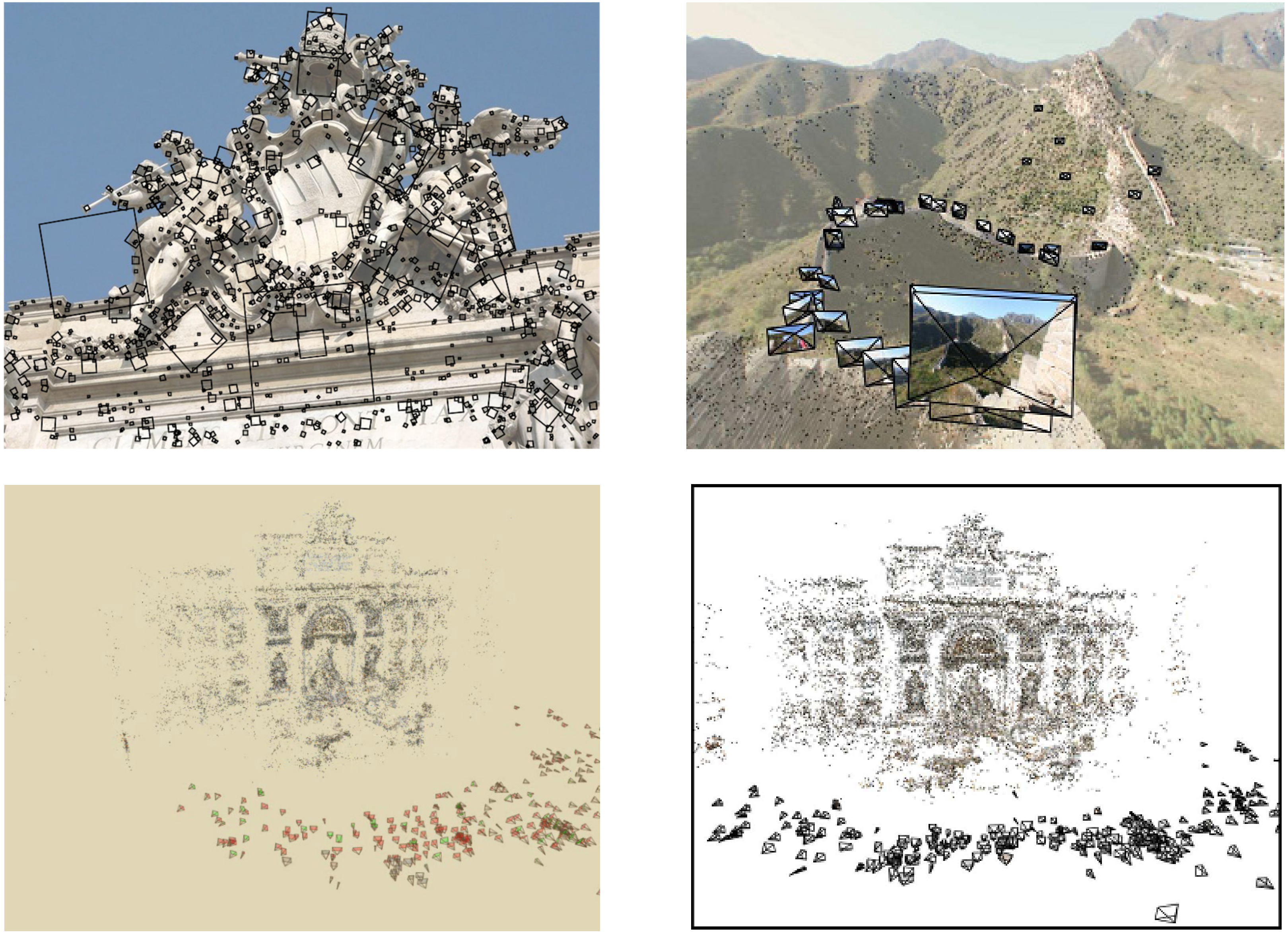

Nister & Stewenius 2006

# Object Classification Results



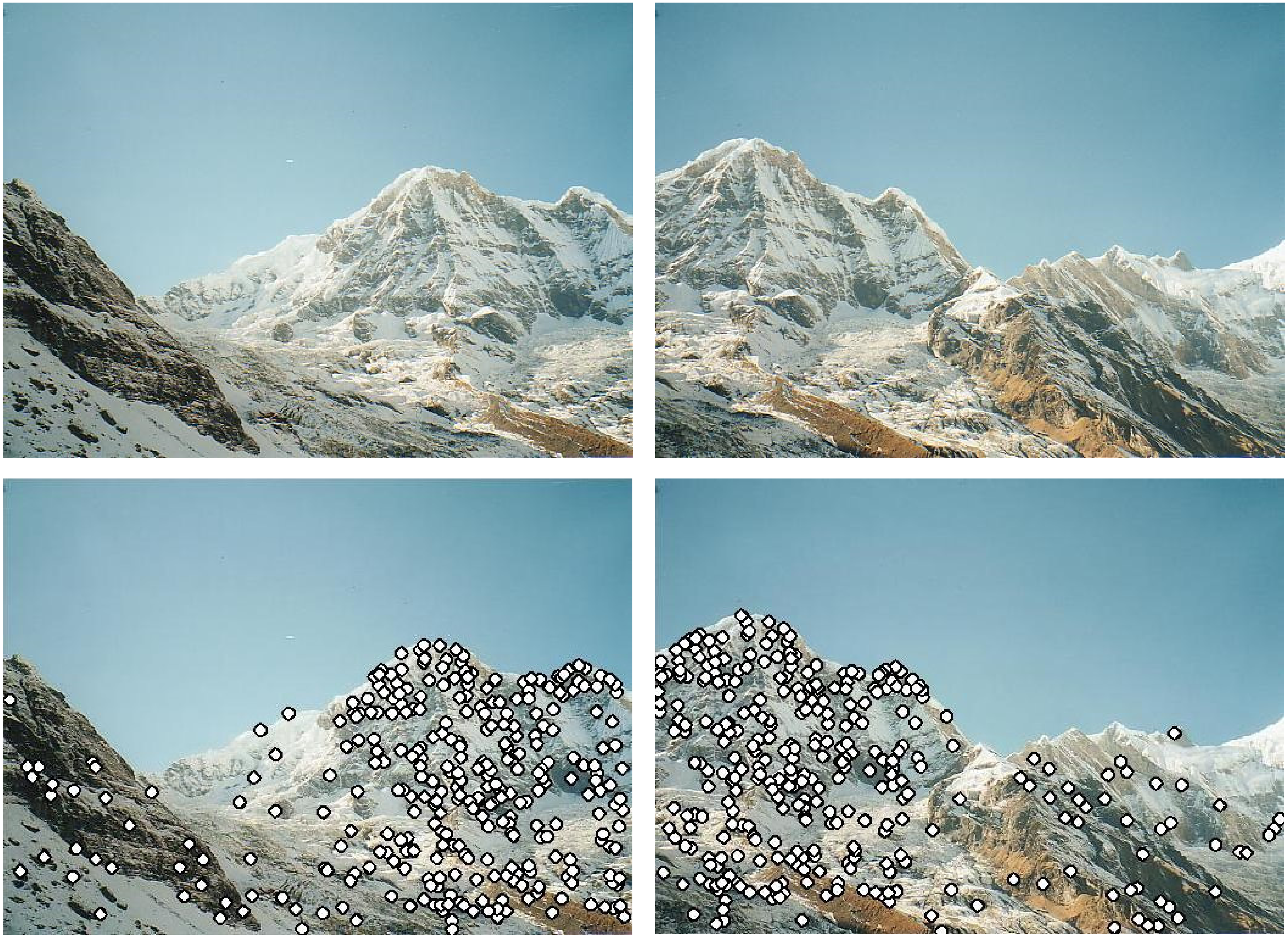
Grauman & Darrell 2006, Dorko & Schmid 2004

# Geometry Estimation Results

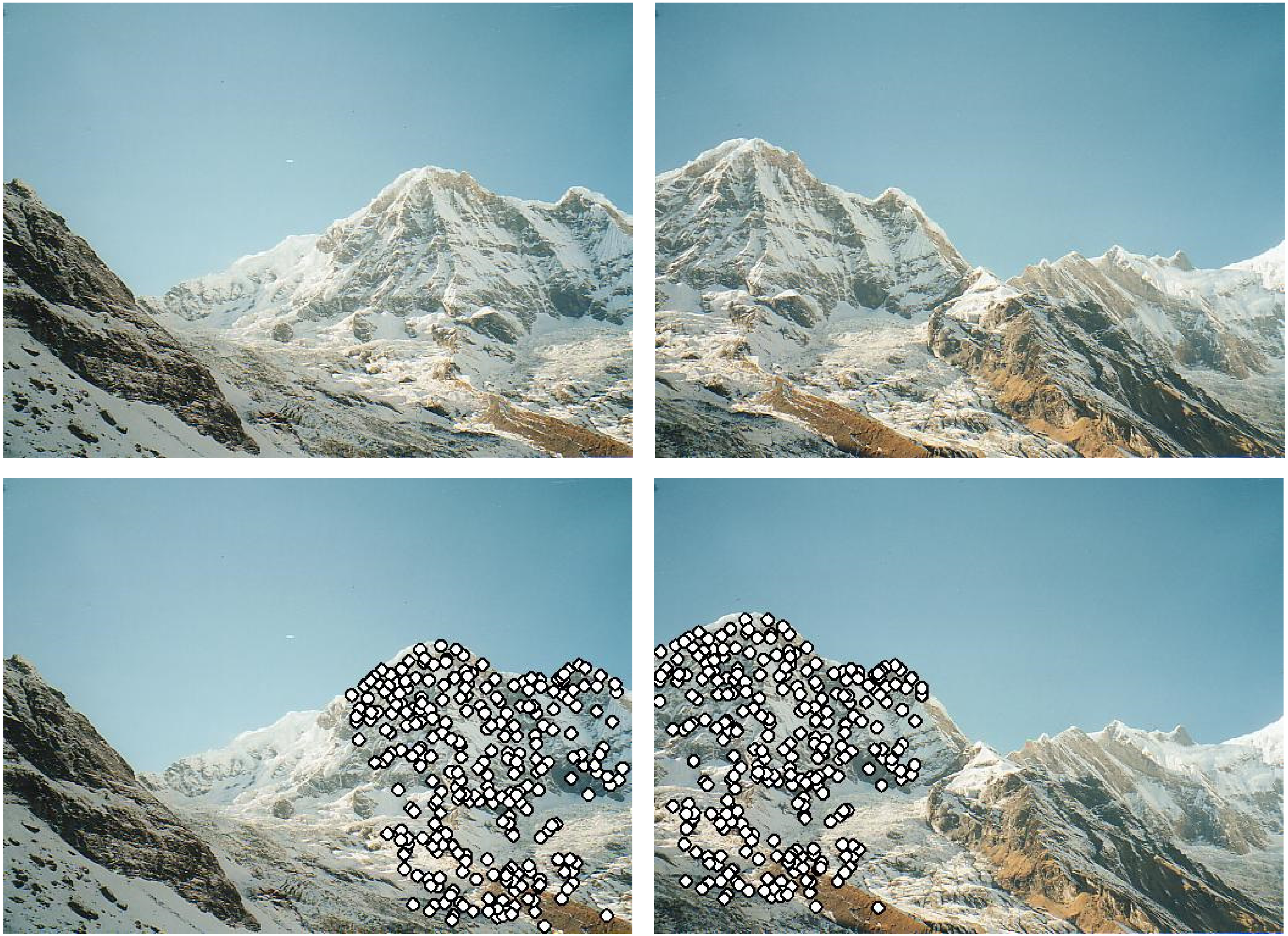


Snavely, Seitz, & Szeliski 2006

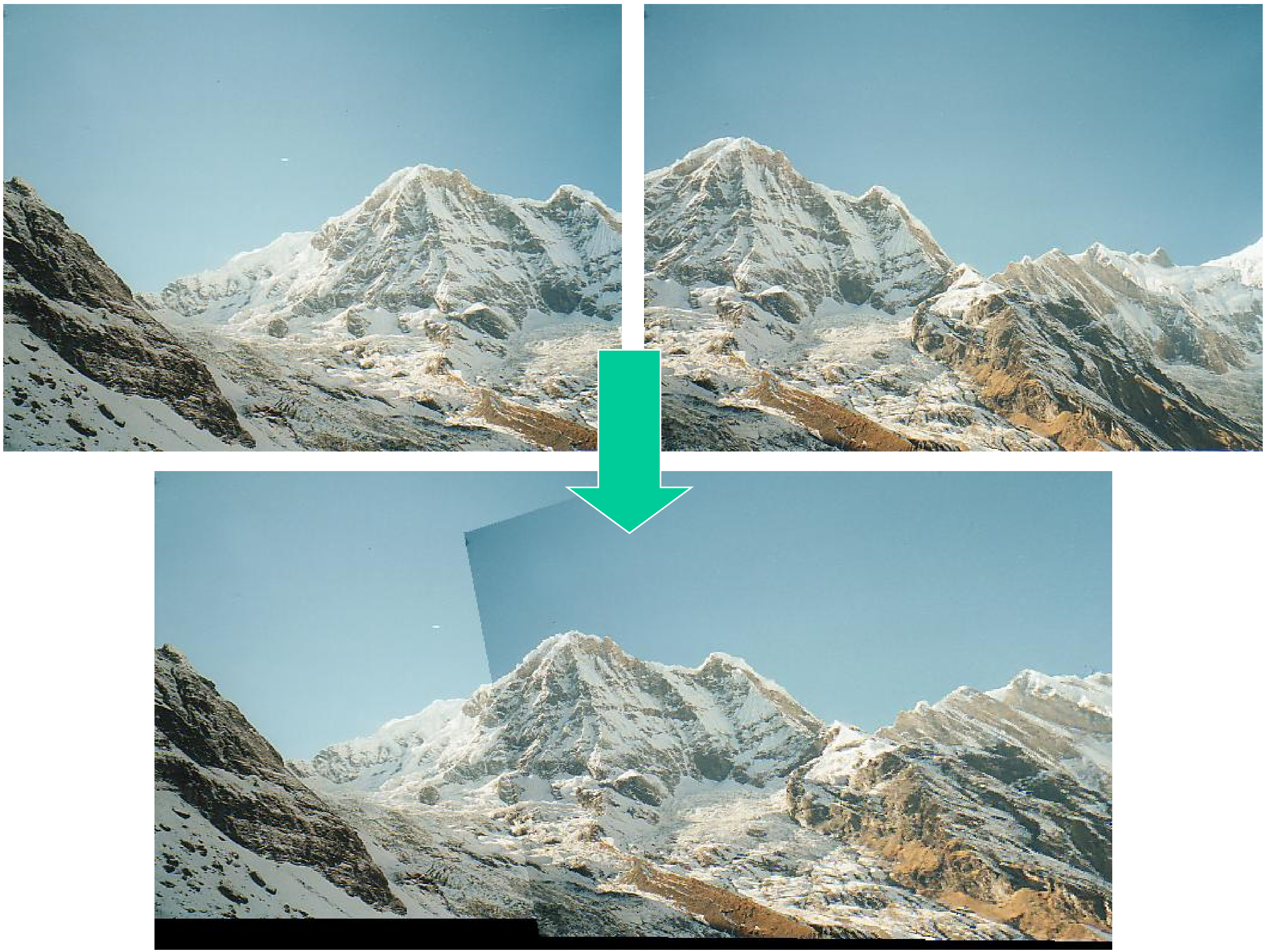
# RANSAC for Homography



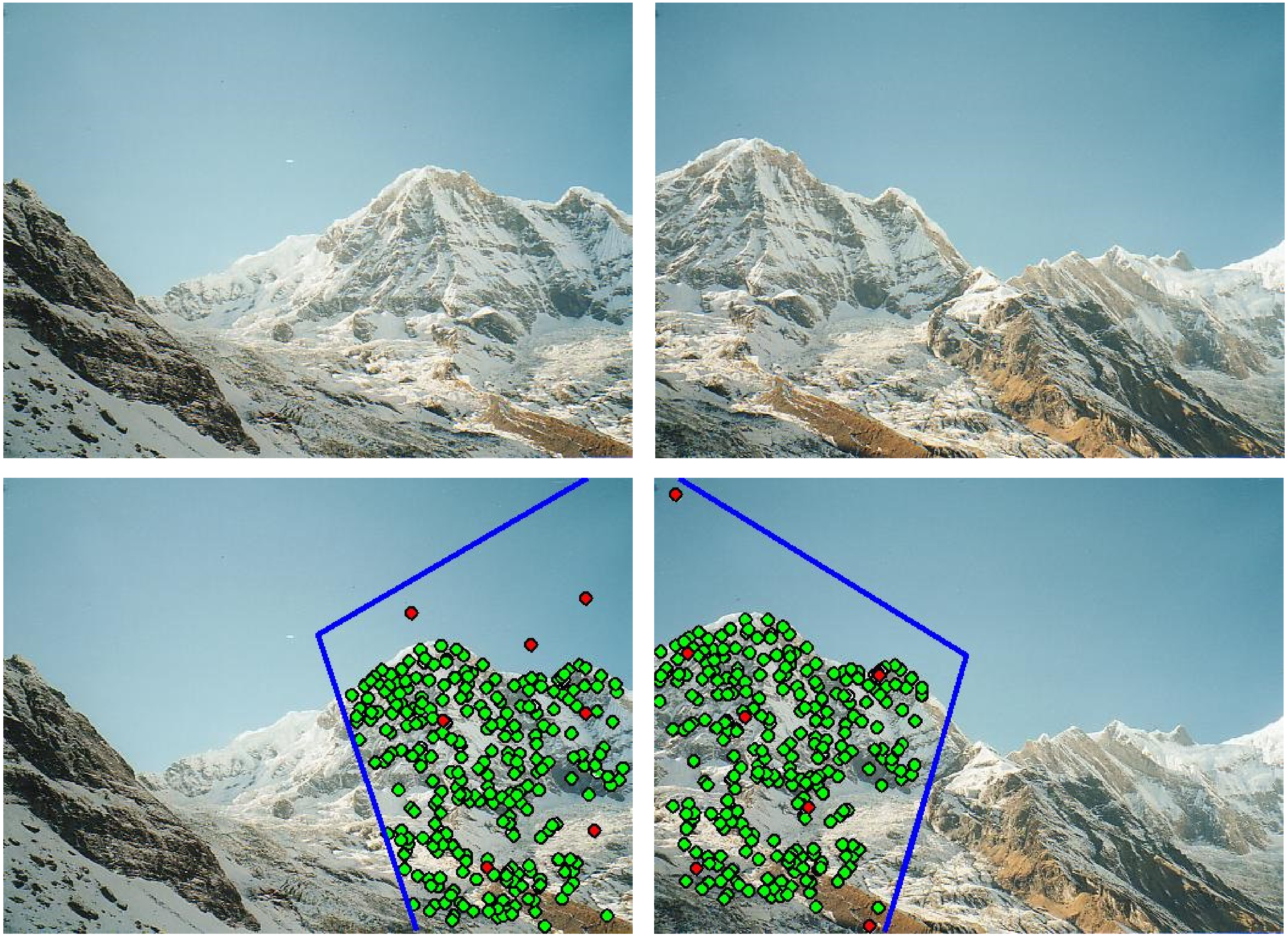
# RANSAC for Homography



# RANSAC for Homography

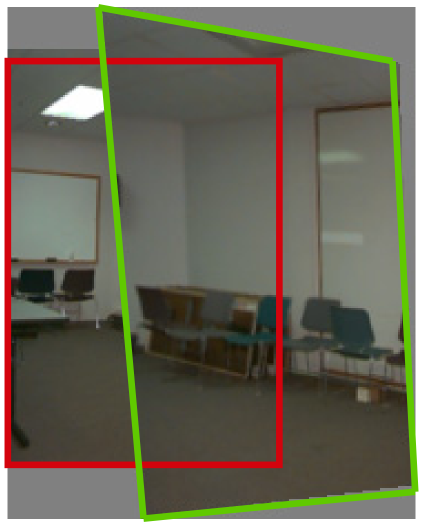


# Probabilistic model for verification



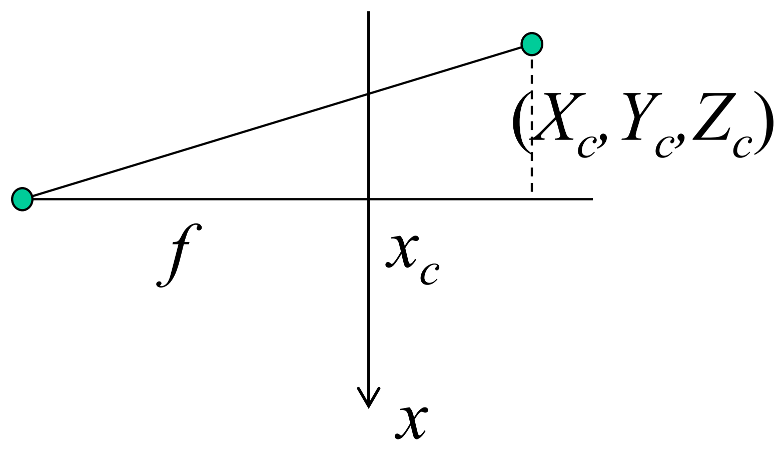
# Plane perspective mosaics

– 8-parameter generalization of affine motion
  - works for pure rotation or planar surfaces
– Limitations:
  - local minima
  - slow convergence



# Revisit Homography

$$\begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

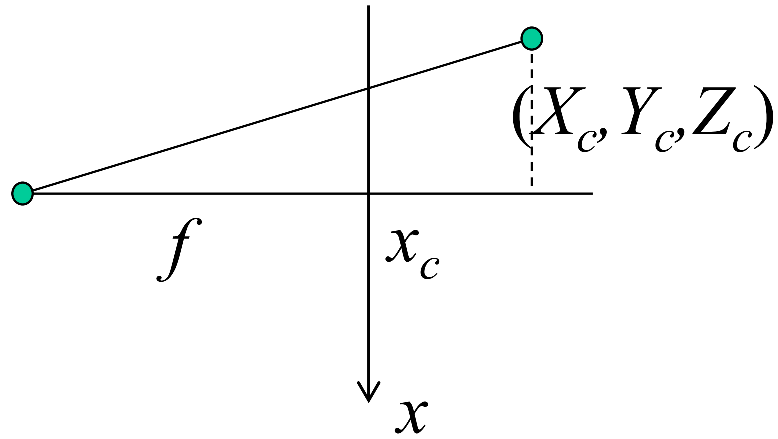



$$\begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$\mathbf{KRK}^{-1}\mathbf{x}_1 \sim \mathbf{x}_2$$

# Estimate f from H?

$$\begin{pmatrix} x_1 - x_c \\ y_1 - y_c \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_1 & 0 & 0 \\ 0 & f_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$




$$\begin{pmatrix} x_2 - x_c \\ y_2 - y_c \\ 1 \end{pmatrix} \sim \begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

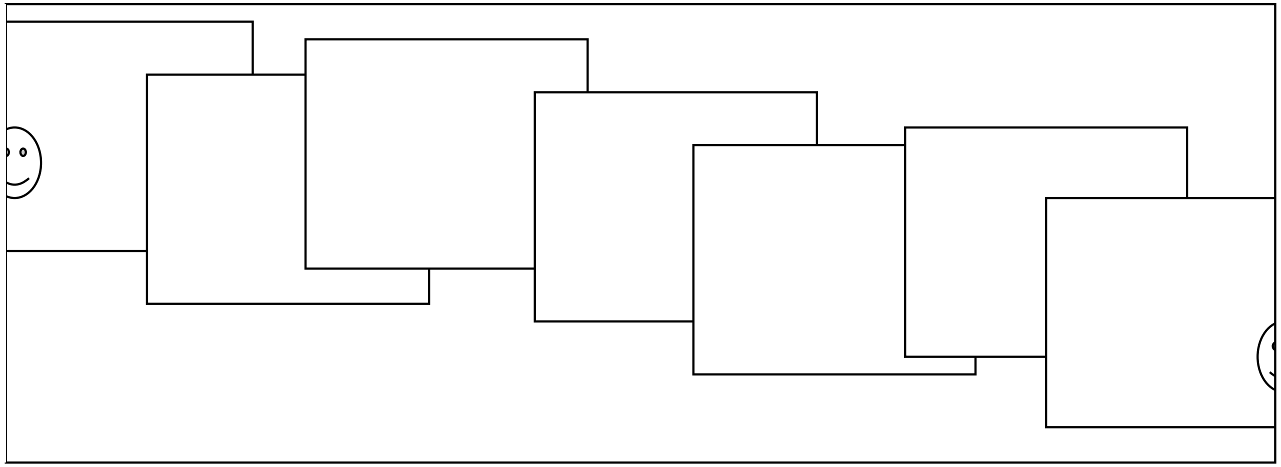
$$\mathbf{R} \sim \mathbf{K}_2^{-1} \mathbf{H} \mathbf{K}_1$$

$$= \begin{bmatrix} a & b & c / f_1 \\ d & e & g / f_1 \\ h * f_2 & i * f_2 & j * \dfrac{f_2}{f_1} \end{bmatrix}$$

$$\underbrace{(\mathbf{K}_2 \mathbf{R} \mathbf{K}_1^{-1})}_{\mathbf{H}} \mathbf{x}_1 \sim \mathbf{x}_2$$

$$f_1 = ?, \quad f_2 = ?$$

# The drifting problem



- Error accumulation
  - small errors accumulate over time

# Bundle Adjustment

Associate each image i with $\mathbf{K}_i \quad \mathbf{R}_i$

Each image i has features :$\mathbf{p}_{ij}$
matched with that say j-th feature in m-th frame

Trying to minimize total matching residuals

$$E(\text{all } f_i \text{ and } \mathbf{R}_i) = \sum_{(i,m)} \sum_j \left\| \mathbf{p}_{ij} \sim \mathbf{K}_i \mathbf{R}_i \mathbf{R}_m^{-1} \mathbf{K}_m^{-1} \mathbf{p}_{mj} \right\|^2$$

*Derive the above, from fundamentals (eqns. 2 slides back).*

# Rotations

- How do we represent rotation matrices?

1. Axis / angle $(\boldsymbol{n},\theta)$
   $\boldsymbol{R} = \boldsymbol{I} + \sin\theta\, [\boldsymbol{n}]_\times + (1 - \cos\theta)\, [\boldsymbol{n}]_\times^2$
   (Rodriguez Formula), with
   $[\boldsymbol{n}]_\times$ be the cross product matrix.

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

# Incremental rotation update

1. Small angle approximation

   $\Delta \boldsymbol{R} = \boldsymbol{I} + \sin\theta\, [\boldsymbol{n}]_\times + (1 - \cos\theta)\, [\boldsymbol{n}]_\times^2$

   $\approx \boldsymbol{I} + \theta\, [\boldsymbol{n}]_\times = \boldsymbol{I} + [\boldsymbol{\omega}]_\times$

   *linear in* $\boldsymbol{\omega} = \theta \boldsymbol{n}$

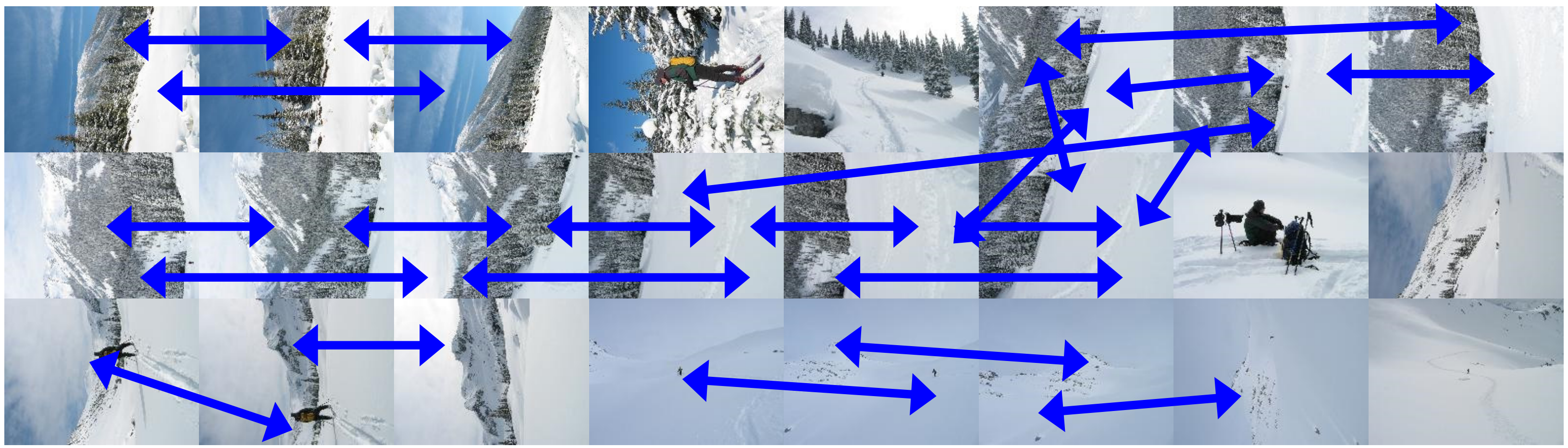
2. Update original $\boldsymbol{R}$ matrix

   $\boldsymbol{R} \leftarrow \boldsymbol{R}\, \Delta \boldsymbol{R}$

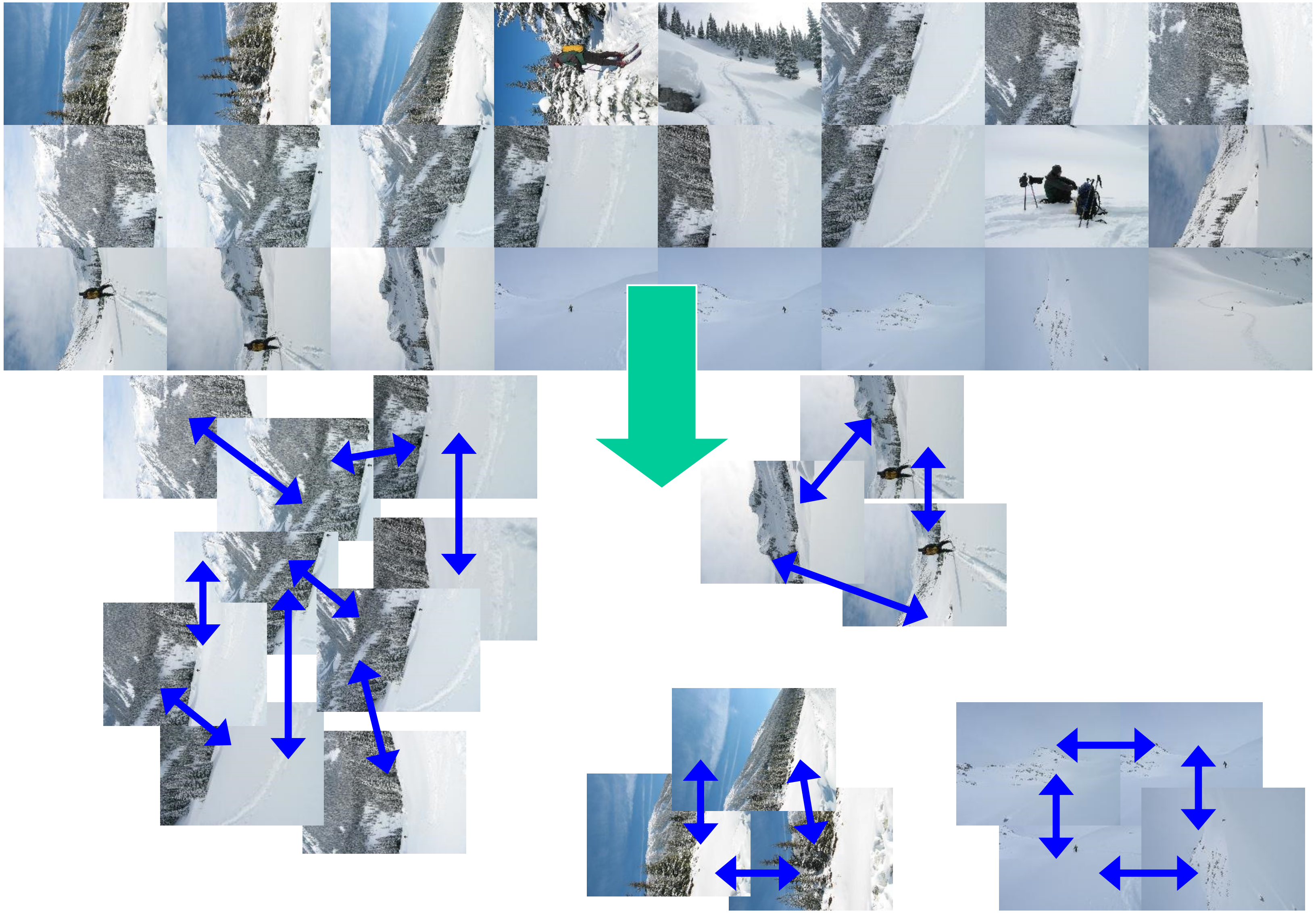
# Recognizing Panoramas

[Brown & Lowe, ICCV'03]

# Finding the panoramas



# Finding the panoramas



# Algorithm overview

**Algorithm: Panoramic Recognition**

**Input:** $n$ unordered images

# Algorithm overview

**Algorithm: Panoramic Recognition**

**Input:** $n$ unordered images

I. Extract SIFT features from all $n$ images

# Algorithm overview

**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

# Algorithm overview

**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)

# Algorithm overview

**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

# Algorithm overview

**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:

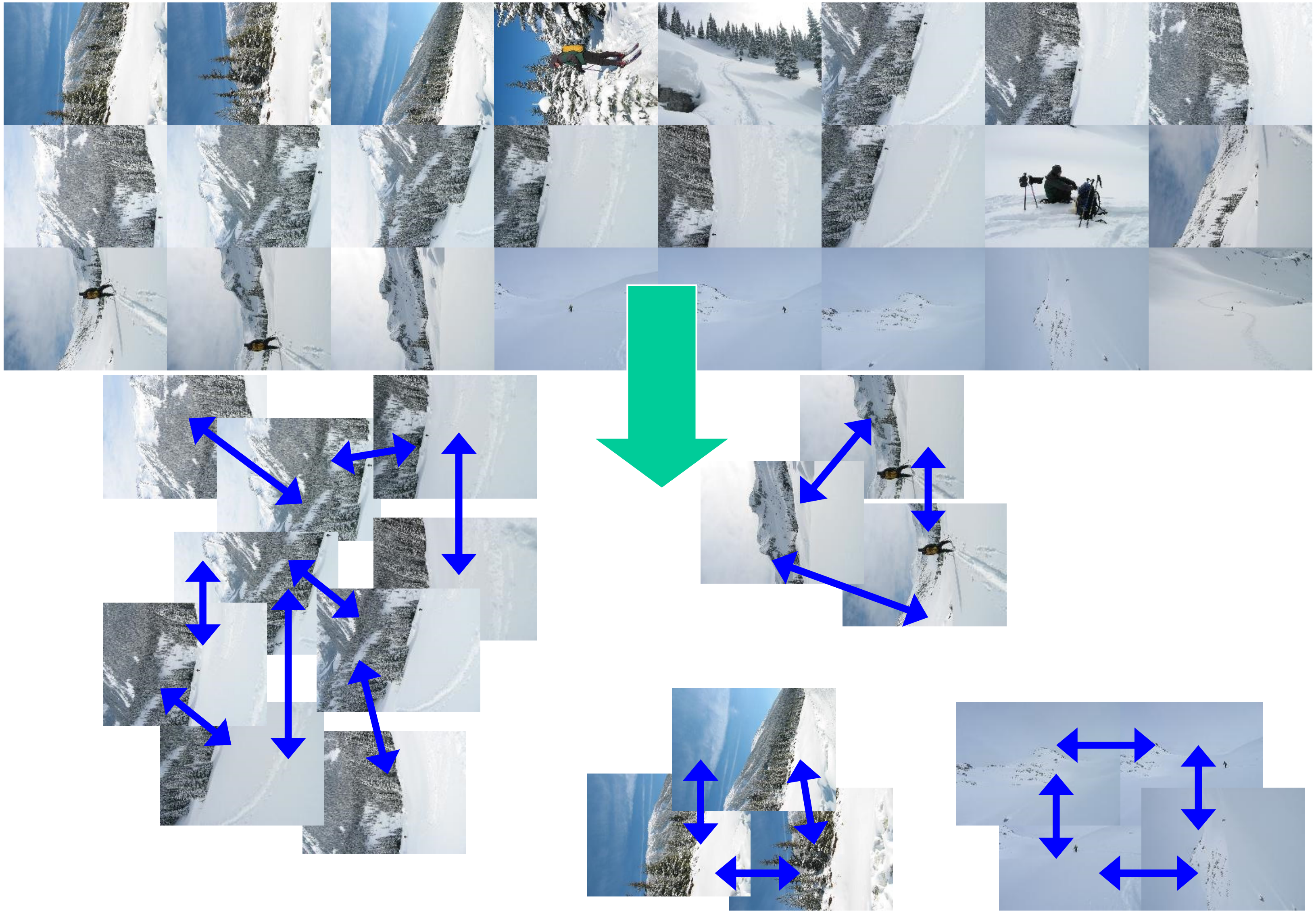
   (i) Select $m$ candidate matching images (with the maximum number of feature matches to this image)

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

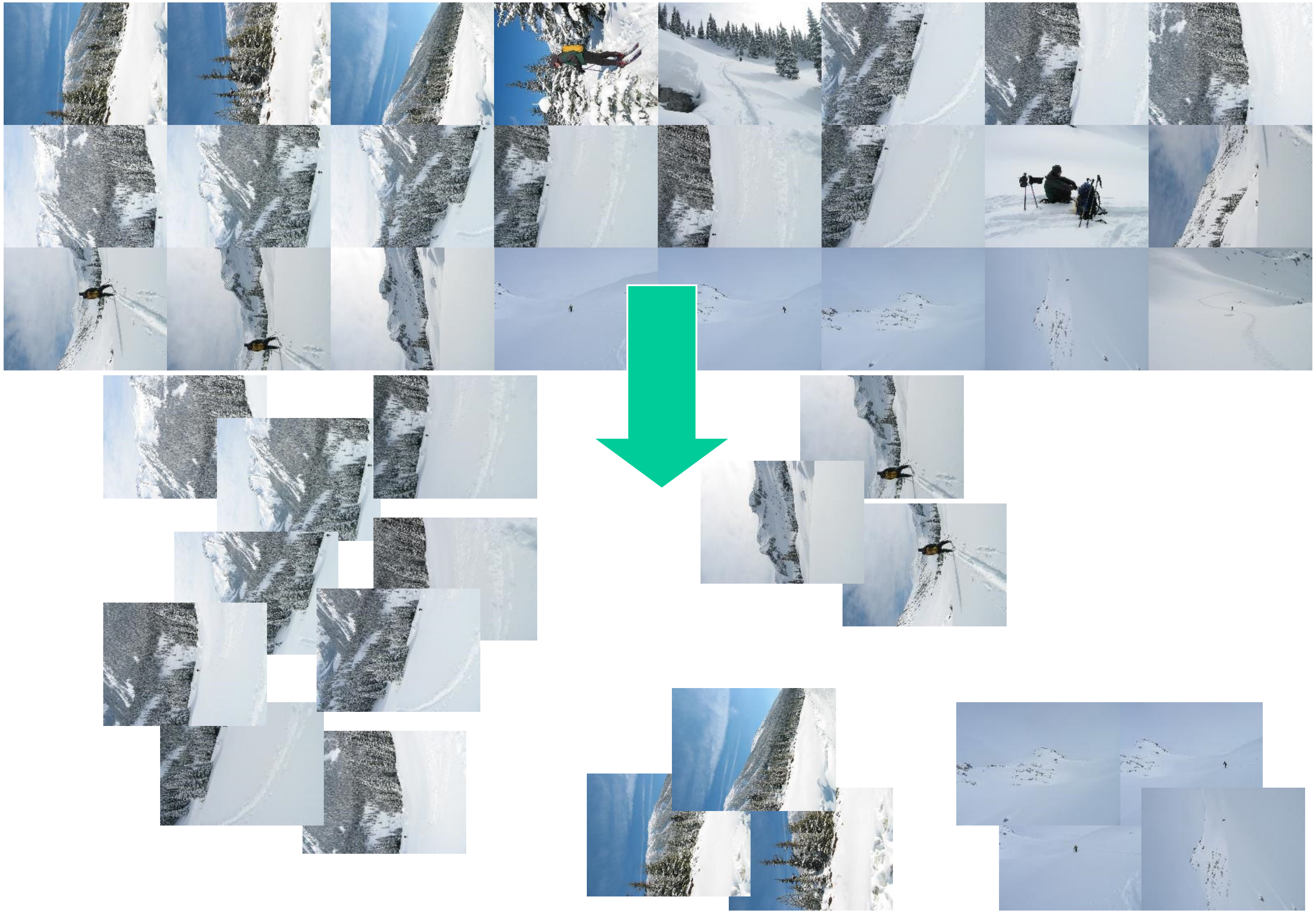
   (iii) Verify image matches using probabilistic model

IV. Find connected components of image matches

# Finding the panoramas



# Finding the panoramas



# Algorithm overview

**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

# Algorithm overview

**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

# Algorithm overview

**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)

# Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images

# Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images

# Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images

- 2D Rotations ($\theta$, $\phi$)
  - Ordering $\nRightarrow$ matching images

# Why “Recognising Panoramas”?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images

- 2D Rotations ($\theta$, $\phi$)
  - Ordering $\nRightarrow$ matching images

# Why "Recognising Panoramas"?

1D Rotations ($\theta$)

- Ordering $\Rightarrow$ matching images

- 2D Rotations ($\theta$, $\phi$)
  - Ordering $\nRightarrow$ matching images

# Homography for Rotation

Parameterise each camera by rotation and focal length

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \ [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This gives pairwise

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j, \ \mathbf{H}_{ij} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^T\mathbf{K}_j^{-1}$$

# Bundle Adjustment

New images initialised with rotation, focal length of best matching image

# Bundle Adjustment

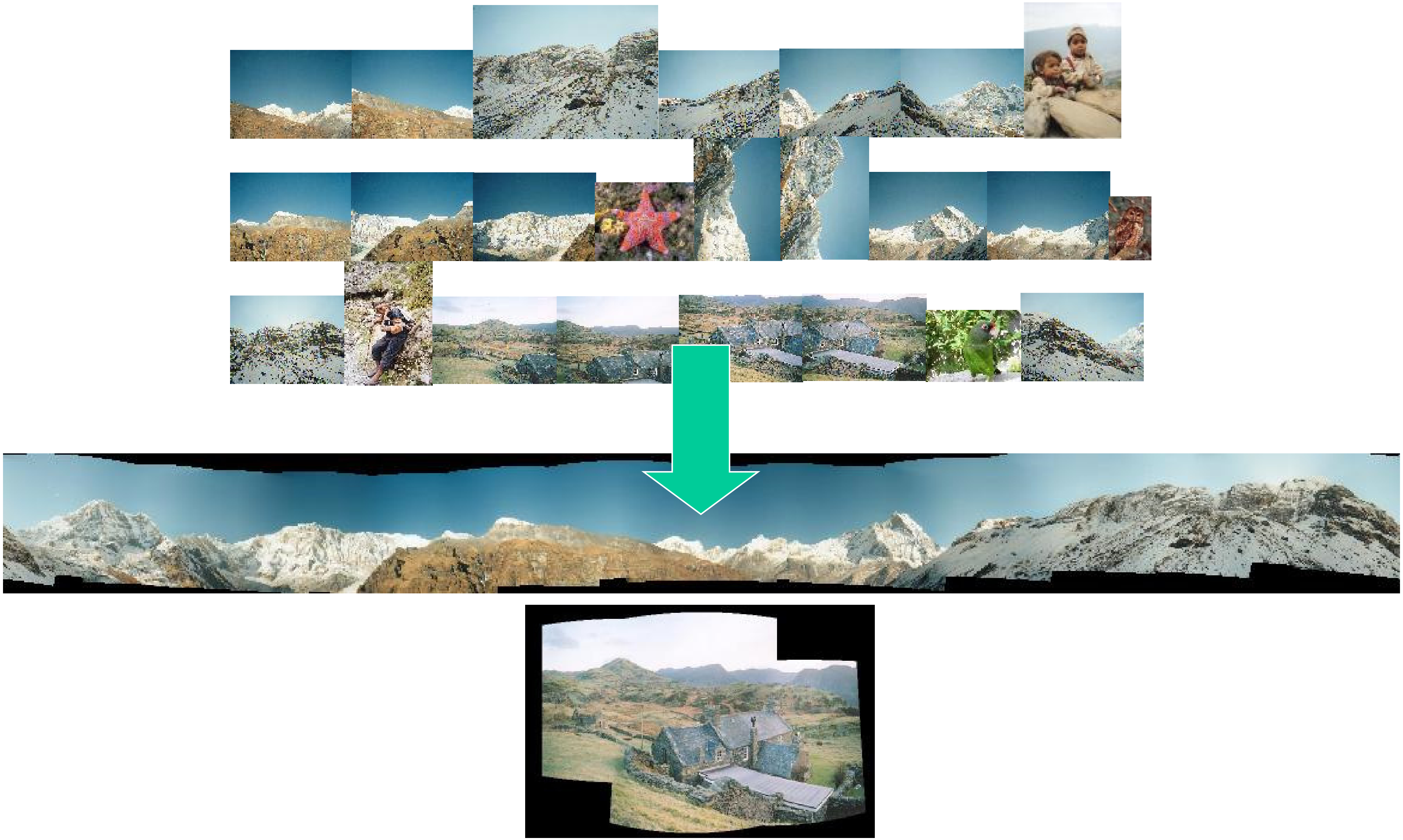
New images initialised with rotation, focal length of best matching image

# Multi-band Blending

Burt & Adelson 1983

- Blend frequency bands over range $\propto \lambda$

# Results



# Matching Mistakes

- Accidental alignment
  - repeated / similar regions
- Failed alignments
  - moving objects / parallax
  - low overlap
  - “feature-less” regions

- No 100% reliable algorithm?