Automatic Panoramic Image Stitching

Dr. Matthew Brown,
University of Bath

AutoStitch iPhone

"Create gorgeous panoramic photos on your iPhone"
- Cult of Mac

"Raises the bar on iPhone panoramas"
- TUAW

"Magically combines the resulting shots"
- New York Times

Available on the iPhone App Store

4F12 class of '99

Case study – Image mosaicing

Any two images of a general scene with the same camera centre are related by a planar projective transformation given by:

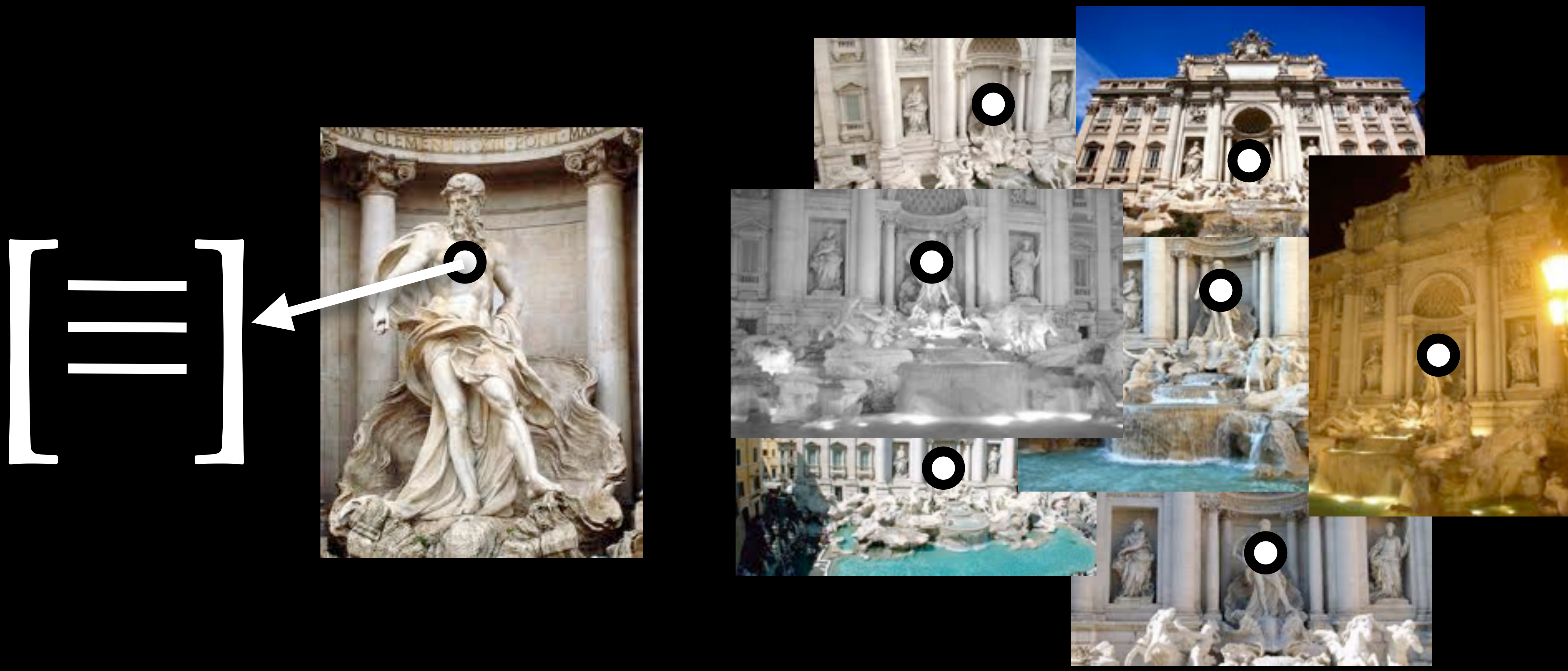
$$\tilde{\mathbf{w}}' = \mathbf{K} \mathrm{R} \mathbf{K}^{-1} \tilde{\mathbf{w}}$$

where $\mathbf{K}$ represents the camera calibration matrix and R is the rotation between the views.

This projective transformation is also known as the homography induced by the plane at infinity. A minimum of four image correspondences can be used to estimate the homography and to warp the images onto a common image plane. This is known as **mosaicing**.

Local Feature Matching

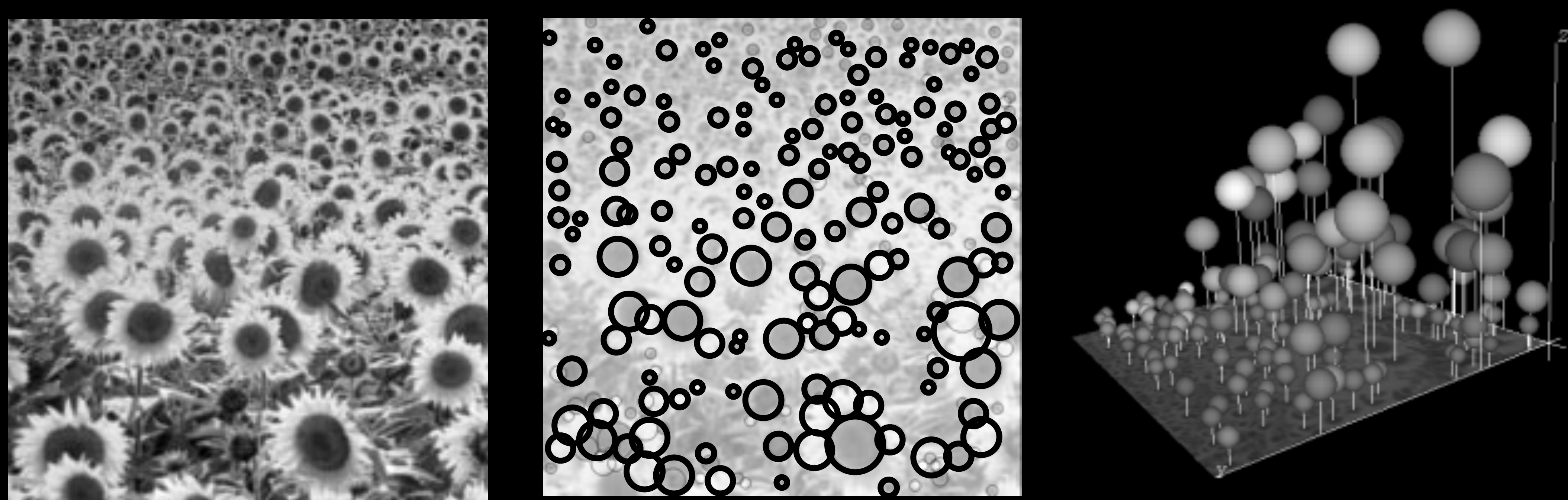
- Given a point in the world...



...compute a description of that point
that can be easily found in other images

Scale Invariant Feature Transform

- Start by detecting points of interest (blobs)



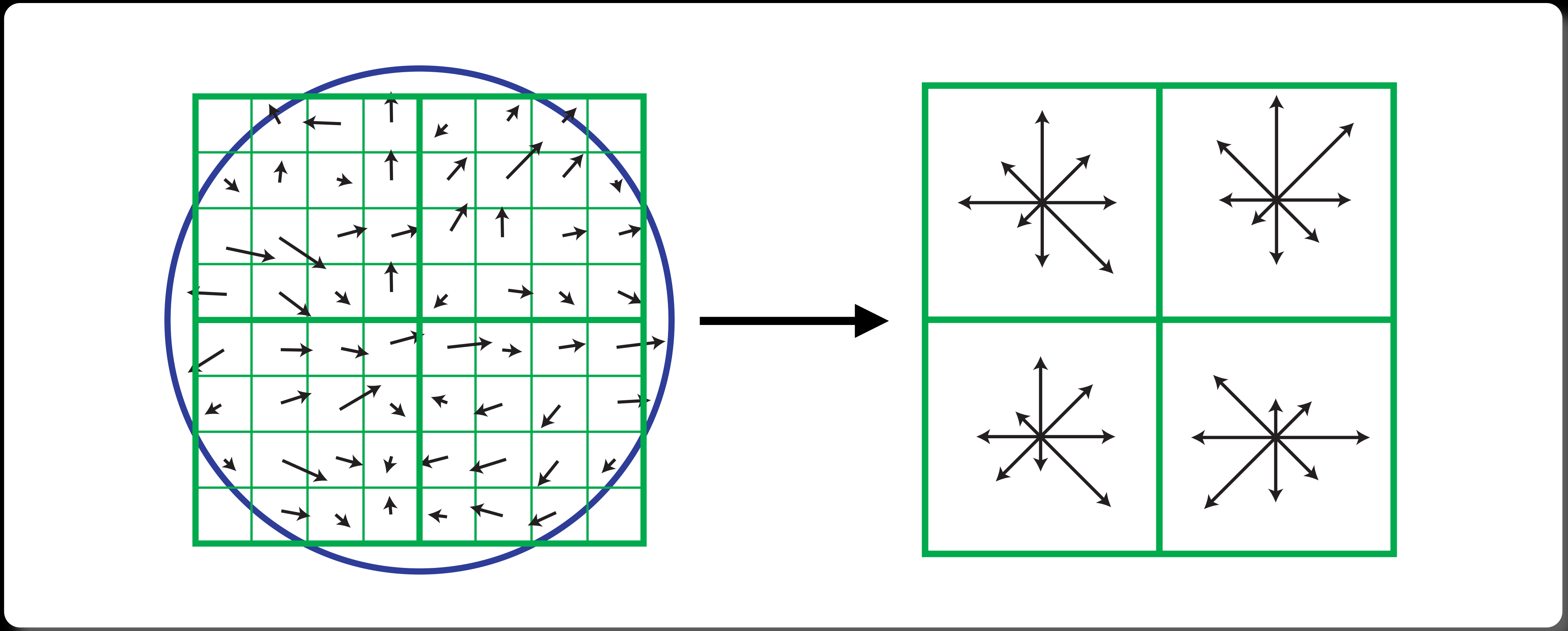
- Find maxima of image Laplacian over scale and space

$$L(I(\mathbf{x})) = \nabla.\nabla I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

[T. Lindeberg]

Scale Invariant Feature Transform

- Describe local region by distribution (over angle) of gradients



- Each descriptor: 4 x 4 grid x 8 orientations = 128 dimensions

Scale Invariant Feature Transform

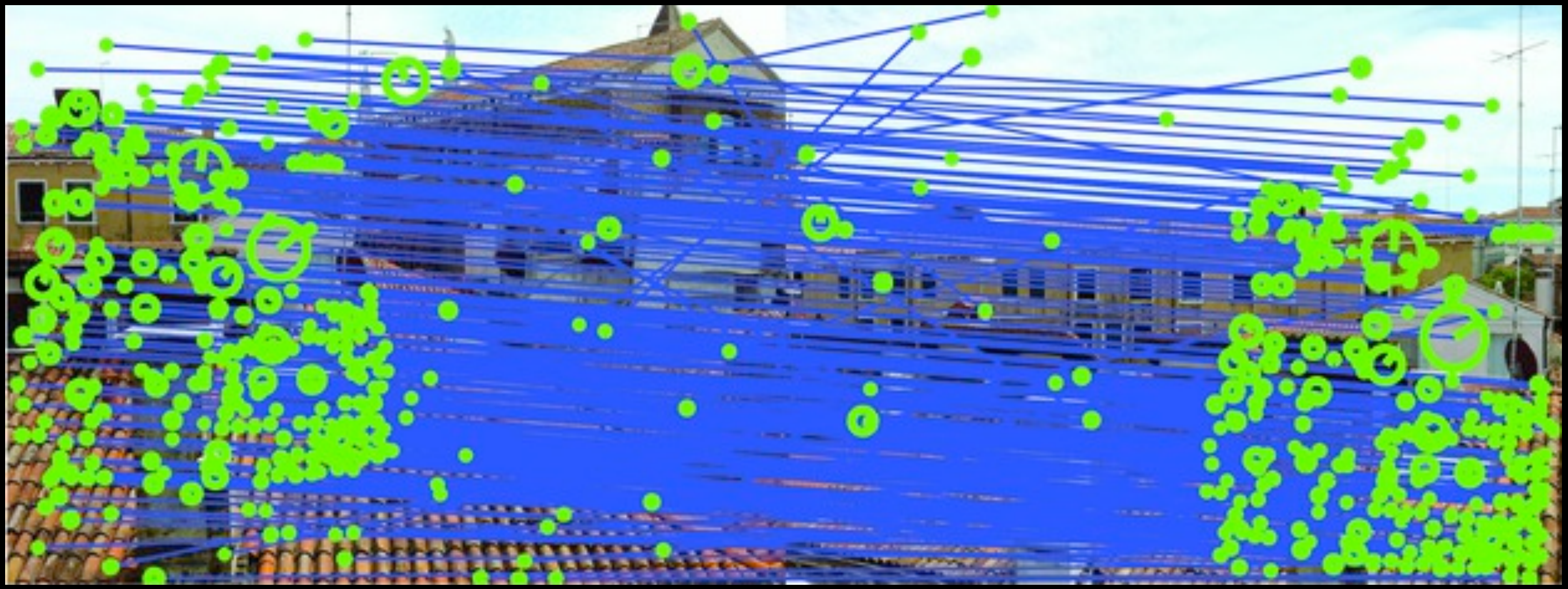
- Extract SIFT features from an image

- Each image might generate 100's or 1000's of SIFT descriptors

Feature Matching

- Goal: Find all correspondences between a pair of images

- Extract and match all SIFT descriptors from both images



[A. Vedaldi]

Feature Matching

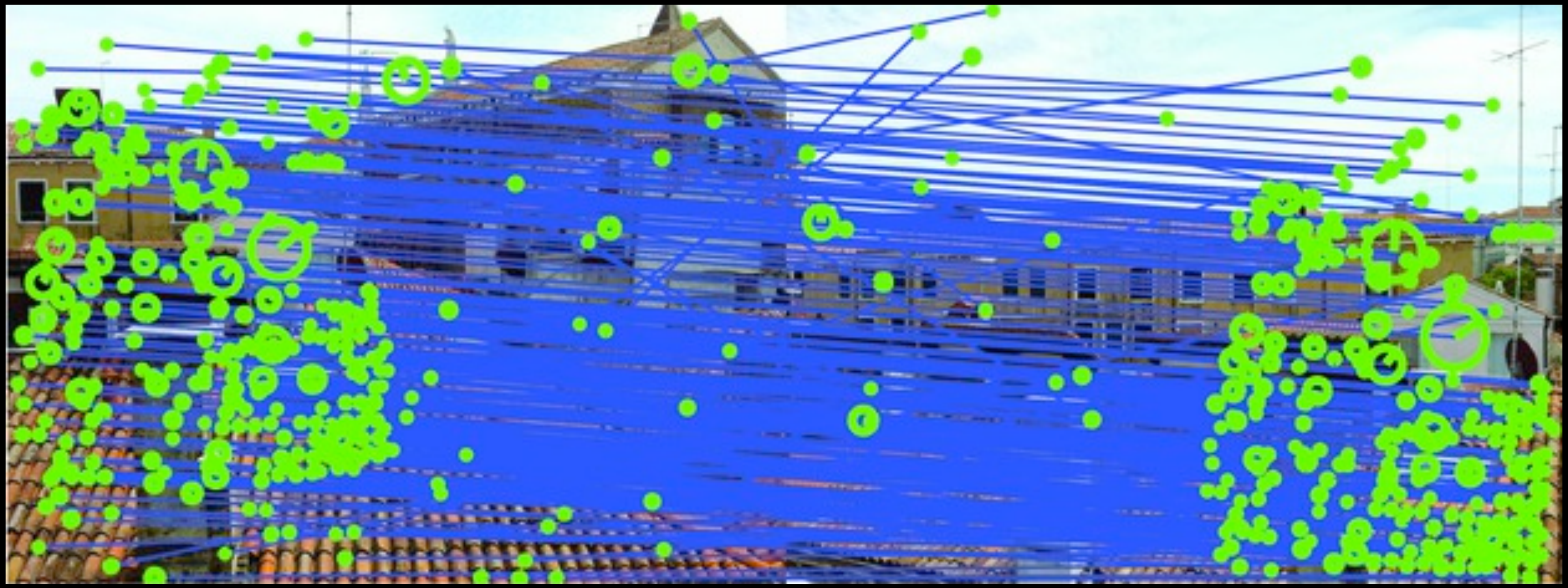
- Each SIFT feature is represented by 128 numbers
- Feature matching becomes task of finding a nearby 128-d vector
- All nearest neighbours:

$$\forall j \;\; NN(j) = \arg\min_i ||\mathbf{x}_i - \mathbf{x}_j||, \;\; i \neq j$$

- Solving this exactly is $O(n^2)$, but good approximate algorithms exist
- e.g., [Beis, Lowe '97] Best-bin first k-d tree
- Construct a binary tree in 128-d, splitting on the coordinate dimensions
- Find approximate nearest neighbours by successively exploring nearby branches of the tree

2-view Rotational Geometry

- Feature matching returns a set of noisy correspondences
- To get further, we will have to understand something about the **geometry** of the setup



2-view Rotational Geometry

- Recall the projection equation for a pinhole camera

$$\tilde{\mathbf{u}} = \begin{bmatrix} & \mathbf{K} & \end{bmatrix} \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \end{array} \right] \tilde{\mathbf{X}}$$

$\tilde{\mathbf{u}} \sim [u,\ v,\ 1]^T$: Homogeneous image position

$\tilde{\mathbf{X}} \sim [X,\ Y,\ Z,\ 1]^T$: Homogeneous world coordinates

$\mathbf{K}\ (3 \times 3)$: Intrinsic (calibration) matrix

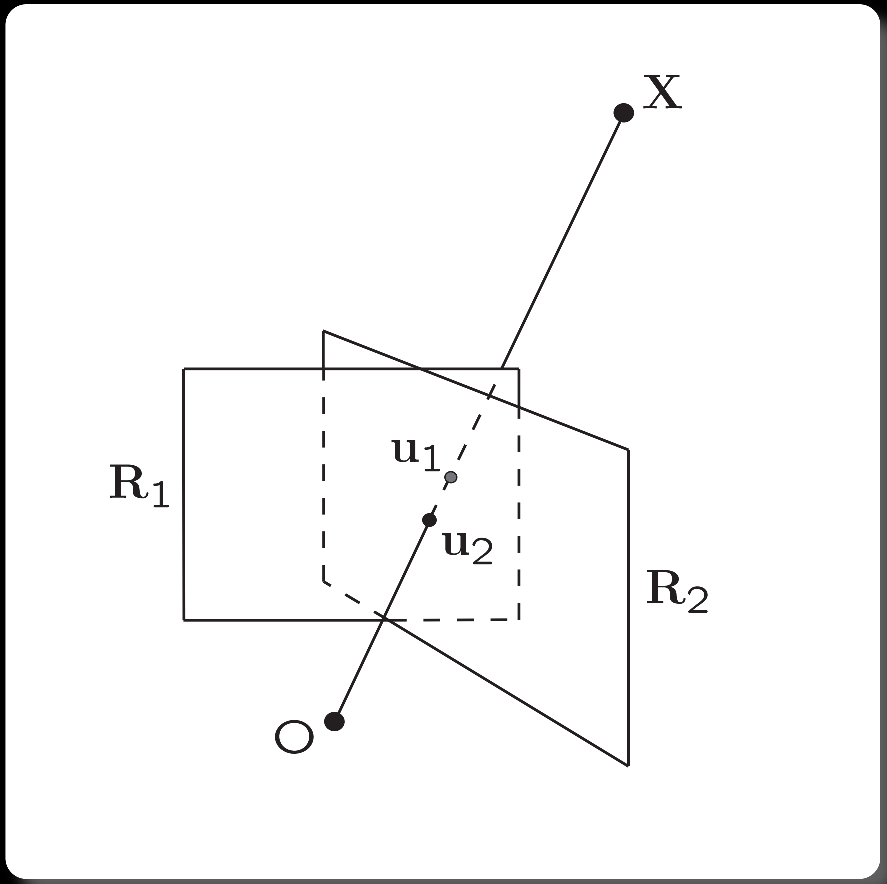
$\mathbf{R}\ (3 \times 3)$: Rotation matrix

$\mathbf{t}\ (3 \times 1)$: Translation vector

2-view Rotational Geometry

- Consider two cameras at the same position (translation)
- WLOG we can put the origin of coordinates there

$$\tilde{\mathbf{u}}_1 = \mathbf{K}_1[\,\mathbf{R}_1 \mid \mathbf{t}_1\,]\,\tilde{\mathbf{X}}$$



- Set translation to 0

$$\tilde{\mathbf{u}}_1 = \mathbf{K}_1[\,\mathbf{R}_1 \mid \mathbf{0}\,]\,\tilde{\mathbf{X}}$$

- Remember $\tilde{\mathbf{X}} \sim [X,\ Y,\ Z,\ 1]^T$ so

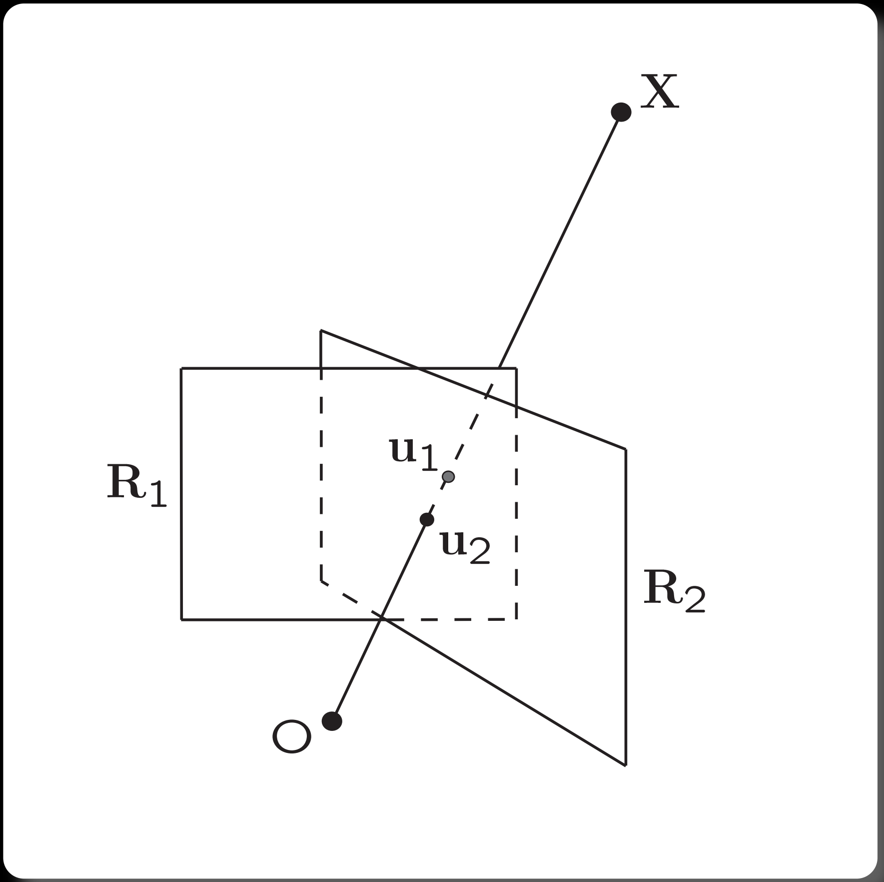
$$\underline{\tilde{\mathbf{u}}_1 = \mathbf{K}_1\mathbf{R}_1\mathbf{X}}$$

(where $\mathbf{X} = [X,\ Y,\ Z]^T$)

2-view Rotational Geometry

- Add a second camera (same translation but different rotation and intrinsic matrix)

$$\tilde{\mathbf{u}}_1 = \mathbf{K}_1\mathbf{R}_1\mathbf{X}$$
$$\tilde{\mathbf{u}}_2 = \mathbf{K}_2\mathbf{R}_2\mathbf{X}$$



- Now eliminate **X**

$$\mathbf{X} = \mathbf{R}_1^T\mathbf{K}_1^{-1}\tilde{\mathbf{u}}_1$$

- Substitute in equation 1

$$\tilde{\mathbf{u}}_2 = \mathbf{K}_2\mathbf{R}_2\mathbf{R}_1^T\mathbf{K}_1^{-1}\tilde{\mathbf{u}}_1$$

This is a 3x3 matrix -- a (special form) of **homography**

Computing H: Quiz

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

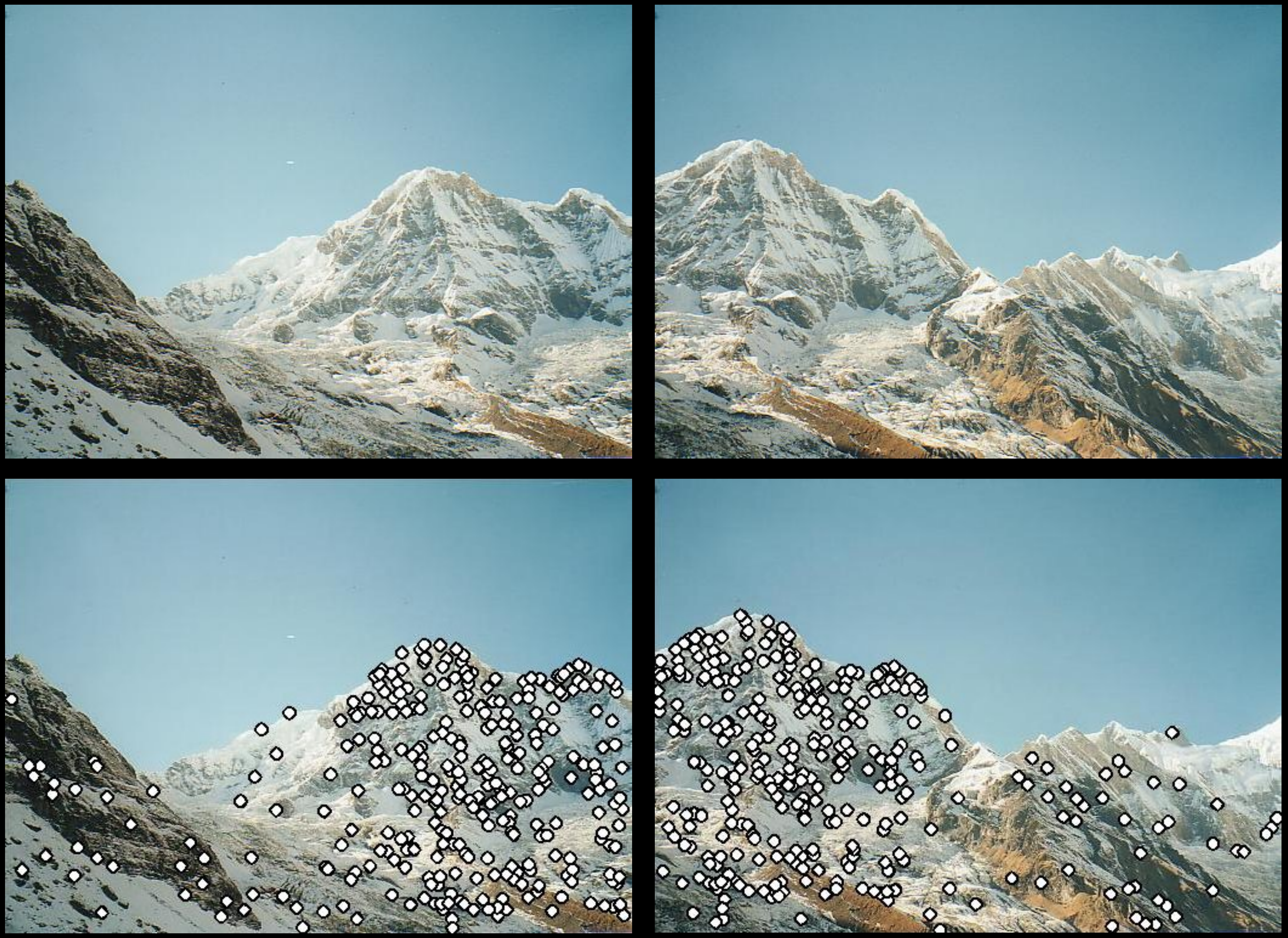
- Each correspondence between 2 images generates _____ equations
- A homography has _____ degrees of freedom
- _____ point correspondences are needed to compute the homography
- Rearranging to make H the subject leads to an equation of the form

$$\mathbf{Mh} = \mathbf{0}$$

- This can be solved by _____

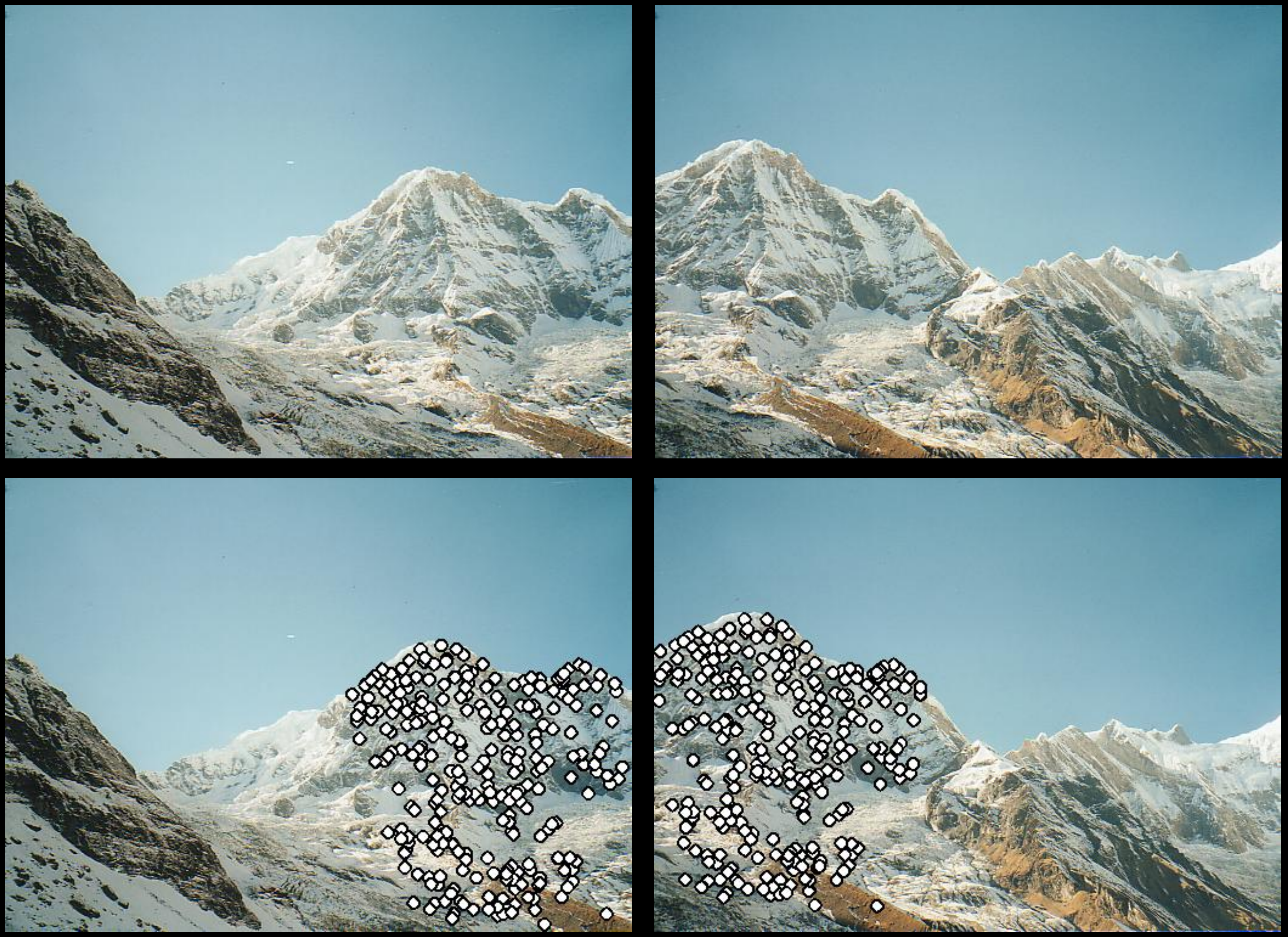
Finding Consistent Matches

- Raw SIFT correspondences (contains **outliers**)



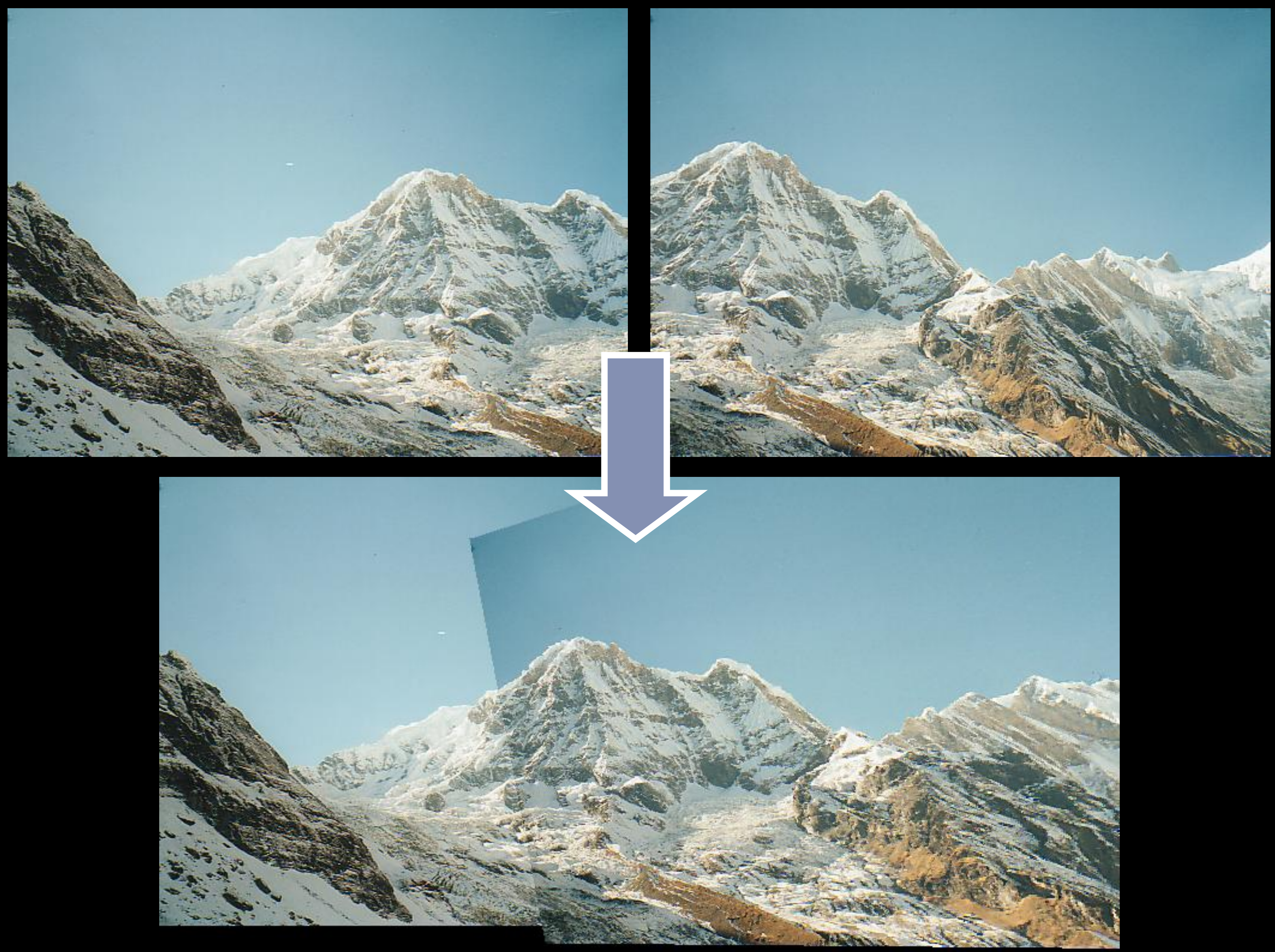
Finding Consistent Matches

- SIFT matches consistent with a rotational **homography**



Finding Consistent Matches

- Warp images to common coordinate frame



RANSAC

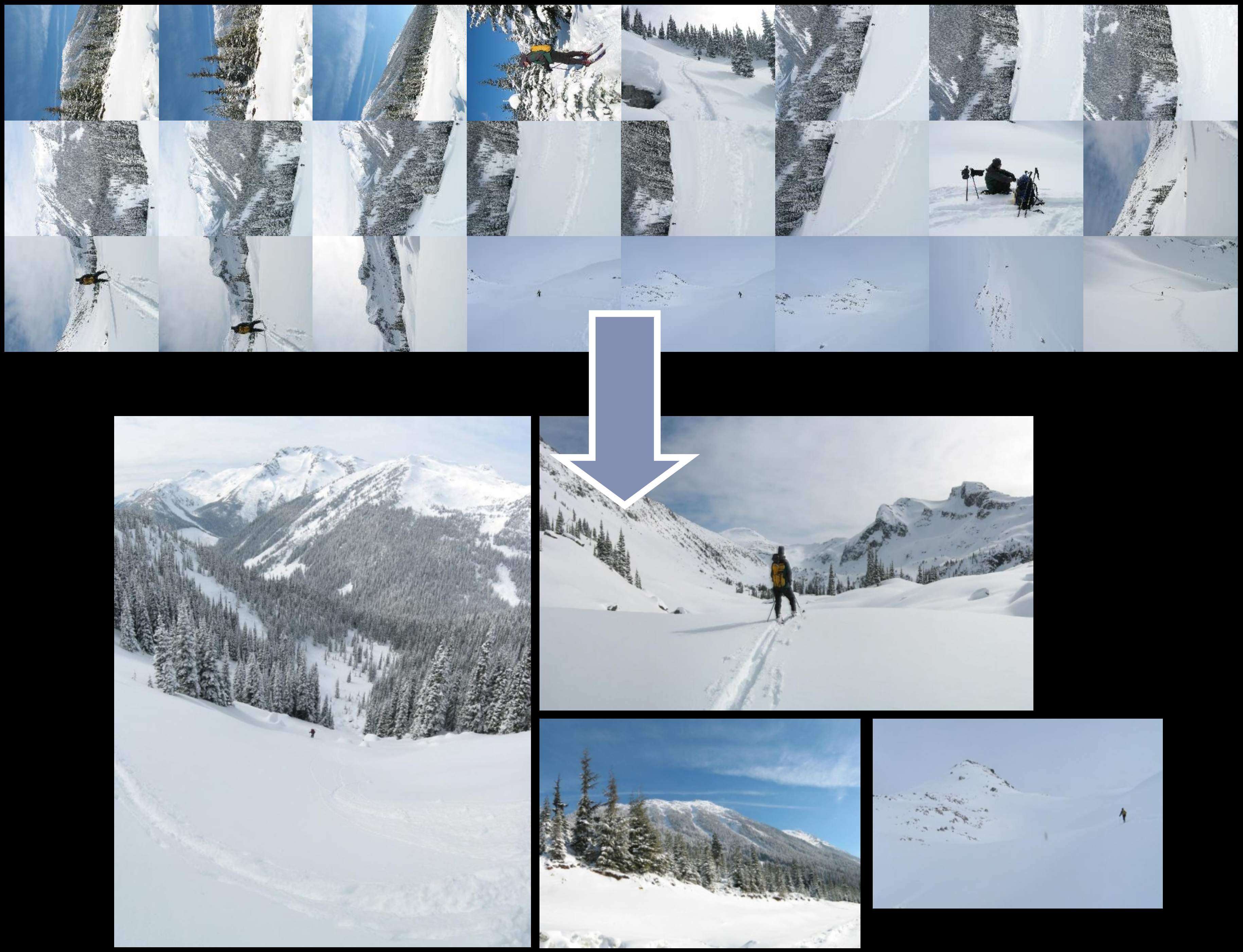
- **RA**ndom **SA**mple **C**onsensus [Fischler-Bolles ’81]
- Allows us to robustly estimate the best fitting homography despite noisy correspondences
- **Basic principle:** select the smallest random subset that can be used to compute H
- Calculate the support for this hypothesis, by counting the number of **inliers** to the transformation
- Repeat sampling, choosing H that maximises # inliers

RANSAC

```
H = eye(3,3); nBest = 0;

for (int i = 0; i < nIterations; i++)
{
    P4 = SelectRandomSubset(P);
    Hi = ComputeHomography(P4);
    nInliers = ComputeInliers(Hi);
    if (nInliers > nBest)
    {
        H = Hi;
        nBest = nInliers;
    }
}
```

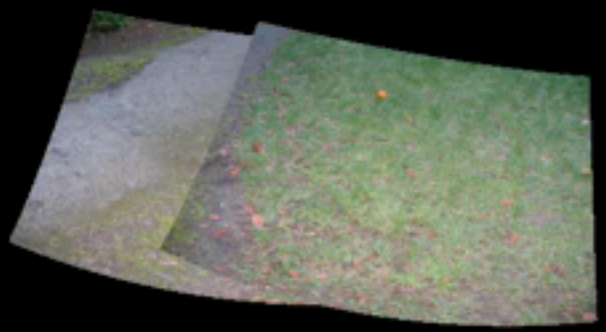
Recognising Panoramas



[Brown, Lowe ICCV’03]

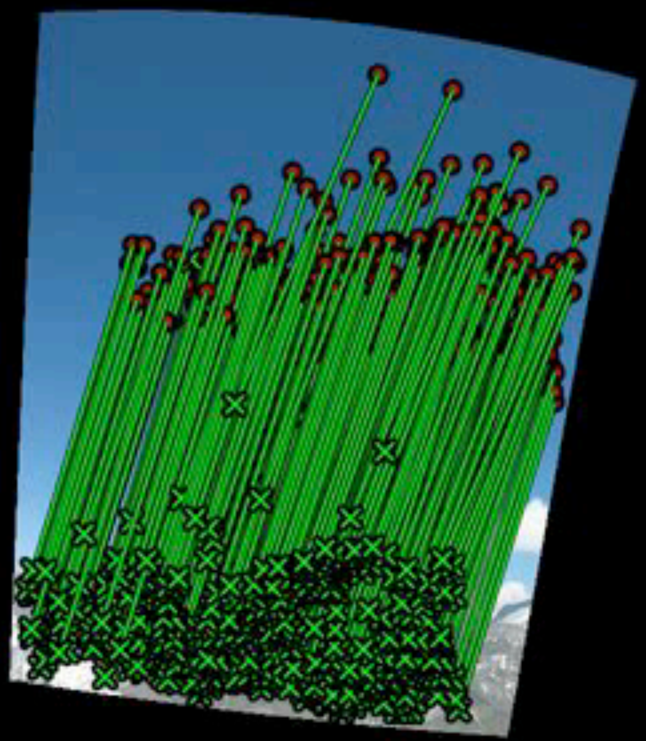
Global Alignment

- The pairwise image relationships are given by **homographies**
- But over time multiple pairwise mappings will accumulate errors
- Notice: gap in panorama before it is closed...



Gap Closing

Bundle Adjustment



Bundle Adjustment

- Minimise sum of robustified residuals

$$\mathbf{u}_{ij} \qquad - \mathbf{m}_{ij}$$

  - $\mathbf{u}_{ij}$ = projected position of point i in image j
  - $\mathbf{m}_{ij}$ = measured position of point i in image j
  - $\mathcal{V}(i)$ = set of images where point i is visible
  - n_p = # points/tracks (mutual feature matches across images)
  - $\mathbf{\Theta}$ = camera parameters

- Robust error function (Huber)

$$f(\mathbf{x}) = \begin{cases} |\mathbf{x}|^2, & |\mathbf{x}| < \sigma \\ 2\sigma|\mathbf{x}| - \sigma^2, & |\mathbf{x}| \geq \sigma \end{cases}$$