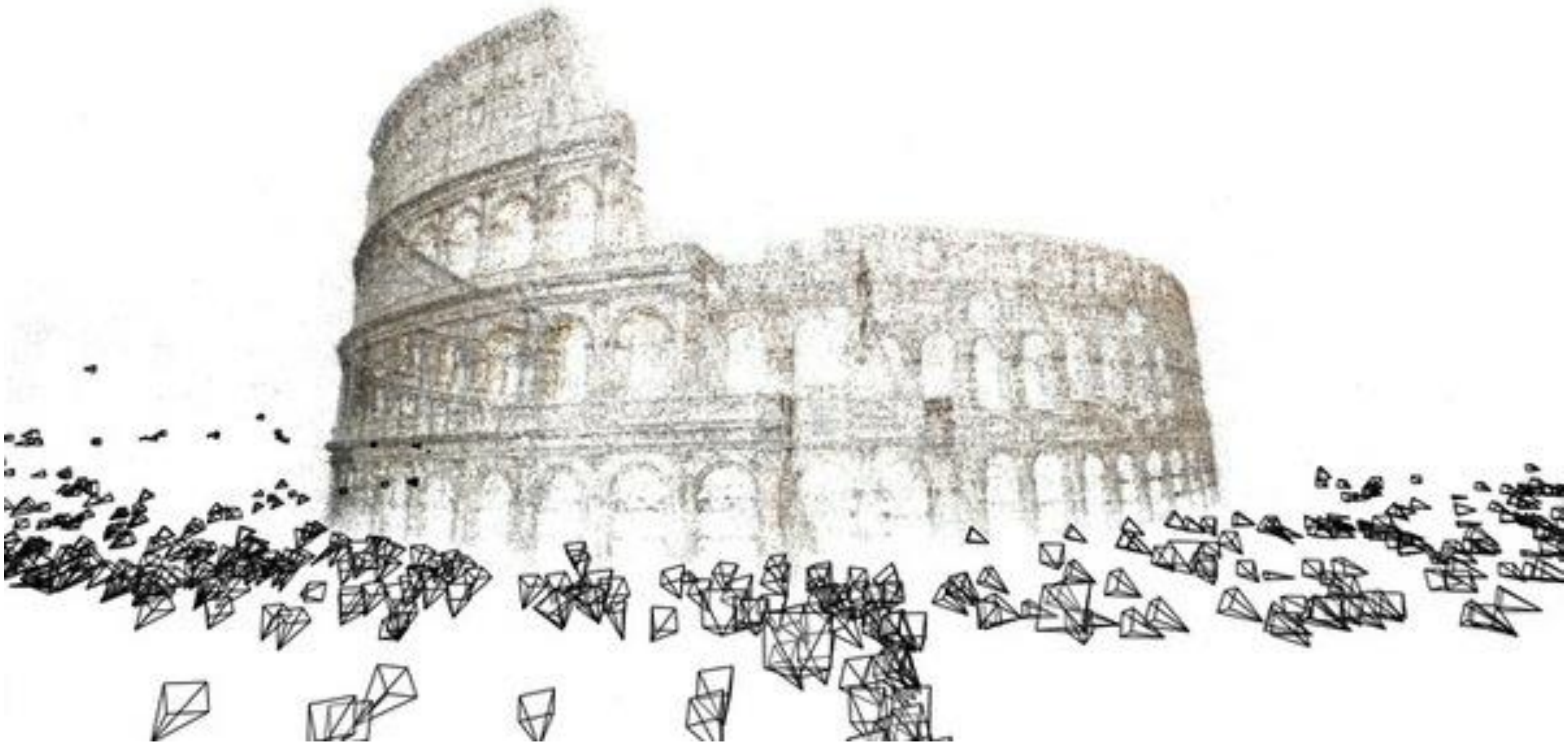
Image correspondences and structure from motion



15-463, 15-663, 15-862
Computational Photography
Fall 2017, Lecture 20

http://graphics.cs.cmu.edu/courses/15-463

Course announcements

- Homework 5 posted.
  - It is a merged version of the original HW5 and the planned HW6.
  - You will need cameras for second part that one as well, so keep the ones you picked up for HW4 (or use your phone cameras.
  - Start in the first week or else you won't finish it ☺ .
- Homework 4 has been graded.
  - Mean: 76:68.
  - Median: 80.
  - Tonemapped (i.e., LDR) images should be stored as .PNG, not .HDR.
- Next guest lecture on Wednesday: Suren Jayasuriya.
  - Will talk about computational sensors.

Overview of today's lecture

- The image correspondence pipeline.
- Describing interest points.
- Matching interest points and RANSAC.
- Structure from motion.

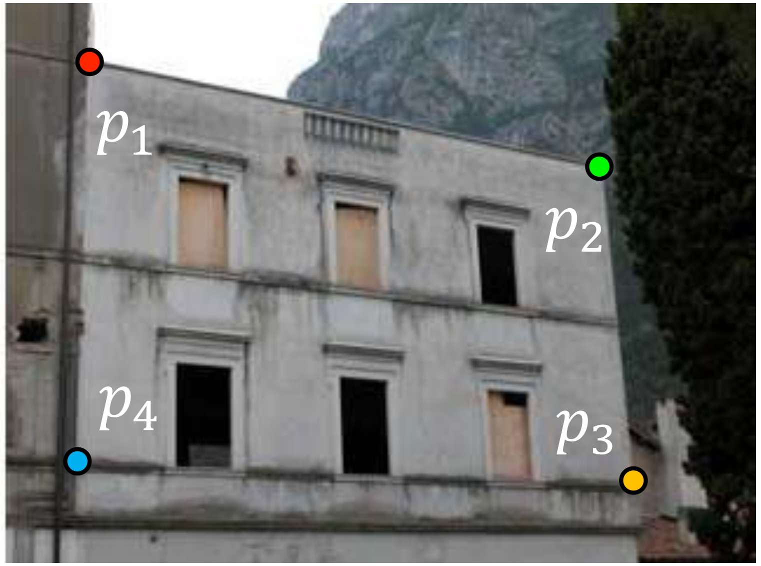
Slide credits

Most of these slides were adapted from:

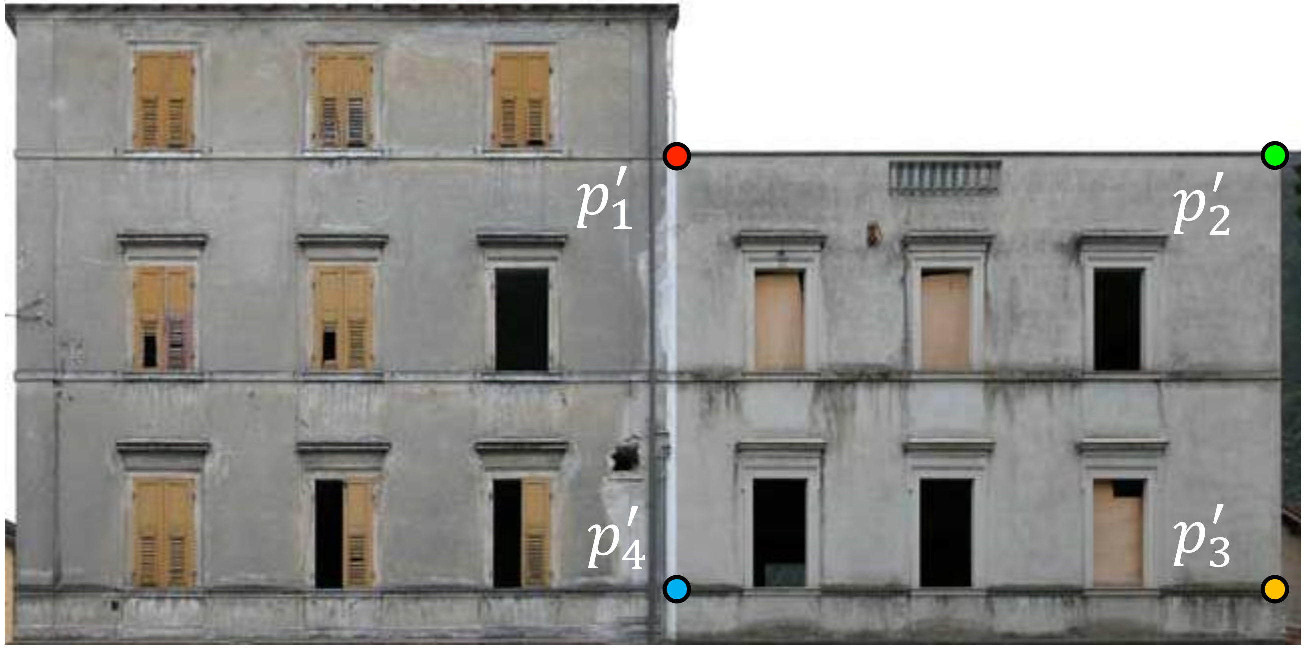
- Kris Kitani (15-463, Fall 2016).
- Noah Snavely (Cornell).

The image correspondence pipeline

Create point correspondences



original image



target image

Can we automate this step?

The image correspondence pipeline

1. Feature point detection
    - Detect corners using the Harris corner detector.

2. Feature point description

3. Feature matching and homography estimation

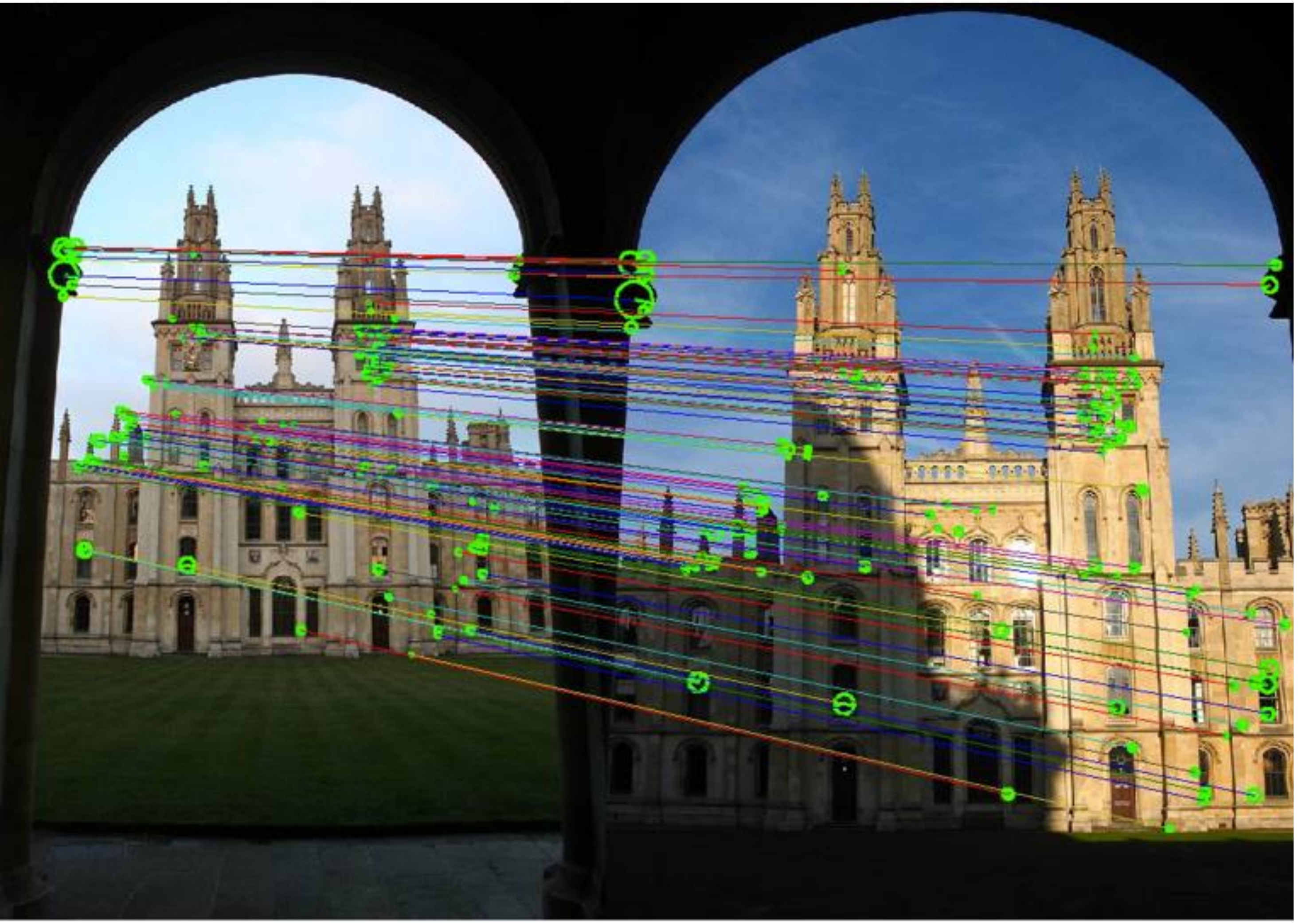
The image correspondence pipeline

1. Feature point detection
    - Detect corners using the Harris corner detector.

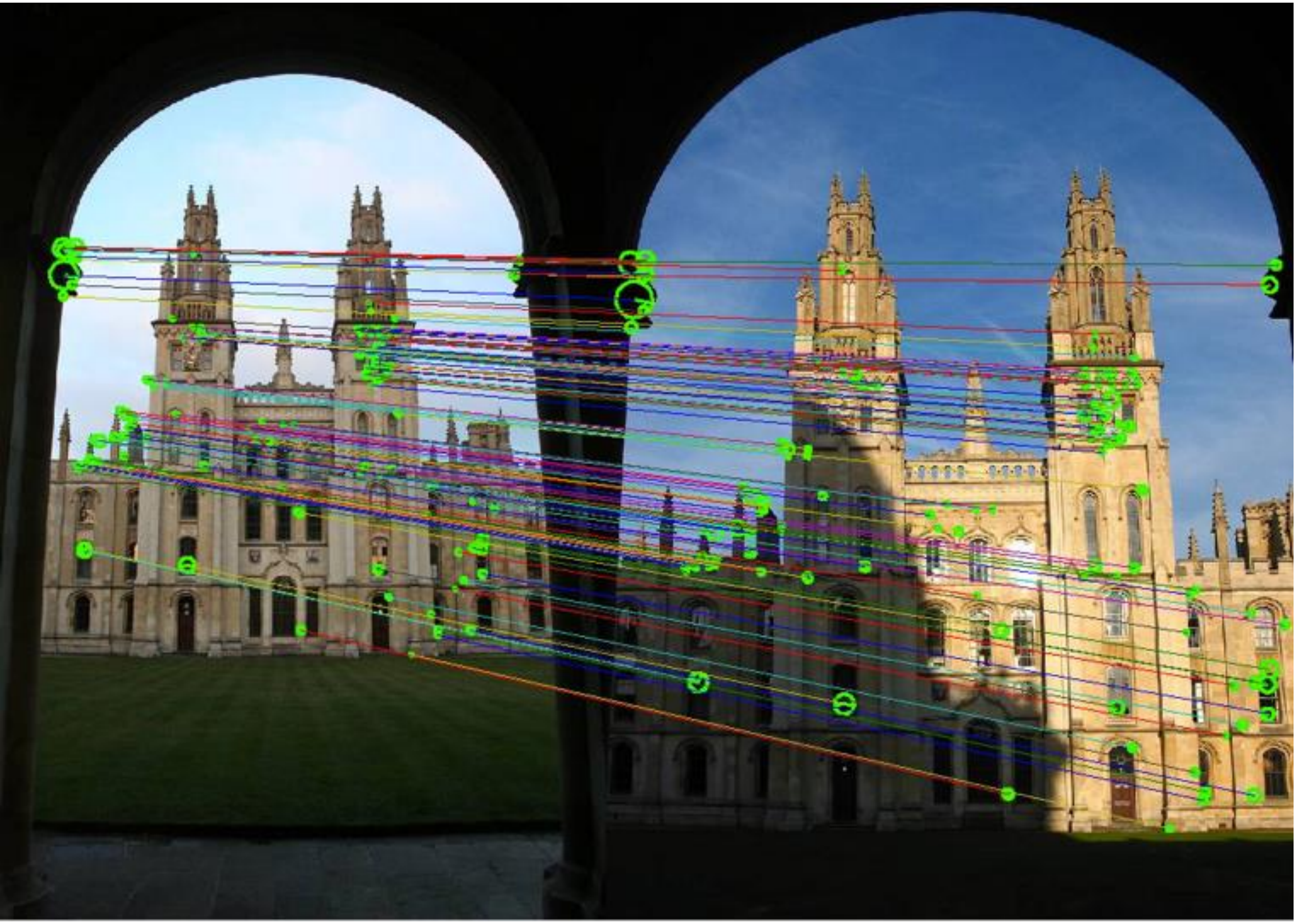
2. Feature point description

3. Feature matching and homography estimation

How do we match features robustly?



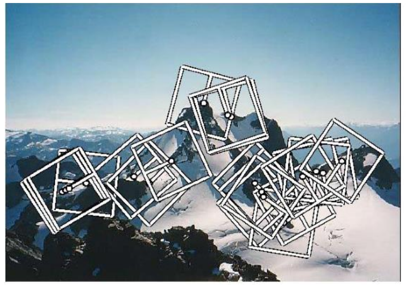
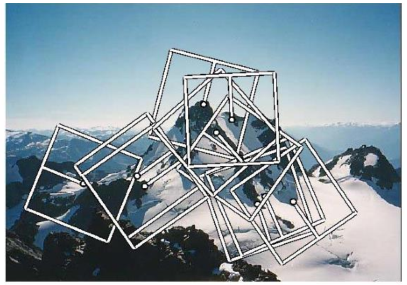
How do we match features robustly?

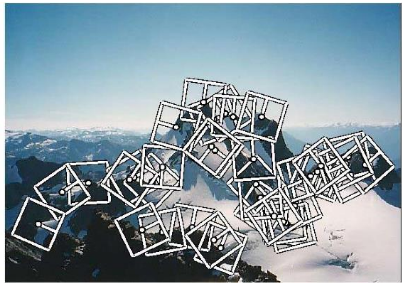


- We need a way to *describe* regions around each feature.
- How do you account for changes in viewpoint or scale?
- Tradeoff between discriminative power and invariance to appearance changes.

Multi-scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

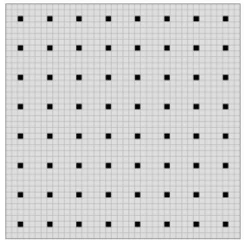


Multi-scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample every 5th pixel
(*what's the purpose of this step?*)

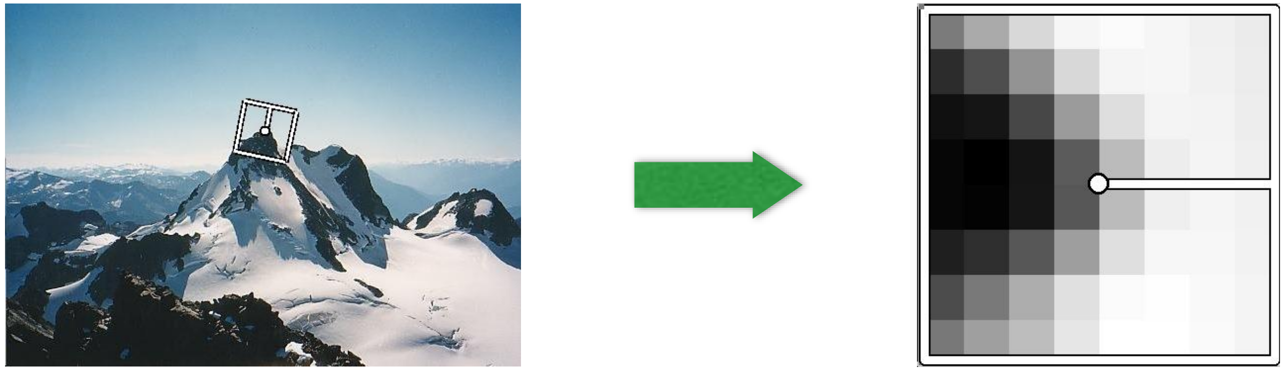


Subtract the mean, divide by standard deviation
(*what's the purpose of this step?*)

Haar Wavelet Transform
(*what's the purpose of this step?*)

Orientation normalization

Use the dominant image gradient direction to
normalize the orientation of the patch



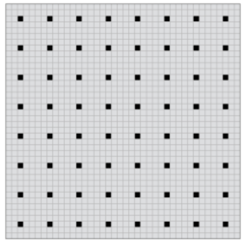
This is how you compute the theta of the feature

Multi-scale Oriented Patches (MOPS)

Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample every 5th pixel
(*what's the purpose of this step?*)



Subtract the mean, divide by standard deviation
(*what's the purpose of this step?*)

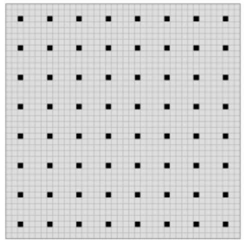
Haar Wavelet Transform
(*what's the purpose of this step?*)

Multi-scale Oriented Patches (MOPS)

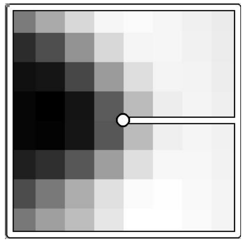
Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

Get 40 x 40 image patch, subsample every 5th pixel
(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard deviation
(*what's the purpose of this step?*)



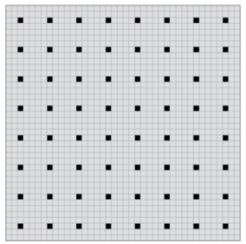
Haar Wavelet Transform
(*what's the purpose of this step?*)

Multi-scale Oriented Patches (MOPS)

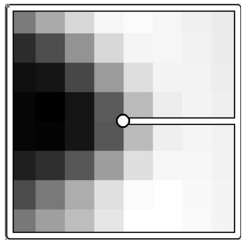
Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

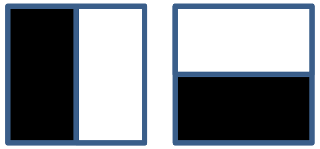
Get 40 x 40 image patch, subsample every 5th pixel
(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard deviation
(removes bias and gain)



Haar Wavelet Transform
(*what's the purpose of this step?*)

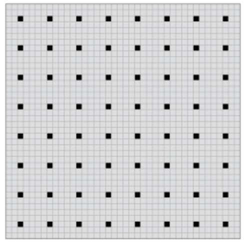


Multi-scale Oriented Patches (MOPS)

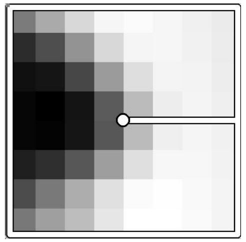
Multi-Image Matching using Multi-Scale Oriented Patches. M. Brown, R. Szeliski and S. Winder.
International Conference on Computer Vision and Pattern Recognition (CVPR2005). pages 510-517

Given a feature (x, y, s, θ)

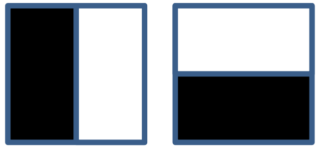
Get 40 x 40 image patch, subsample every 5th pixel
(low frequency filtering, absorbs localization errors)



Subtract the mean, divide by standard deviation
(removes bias and gain)



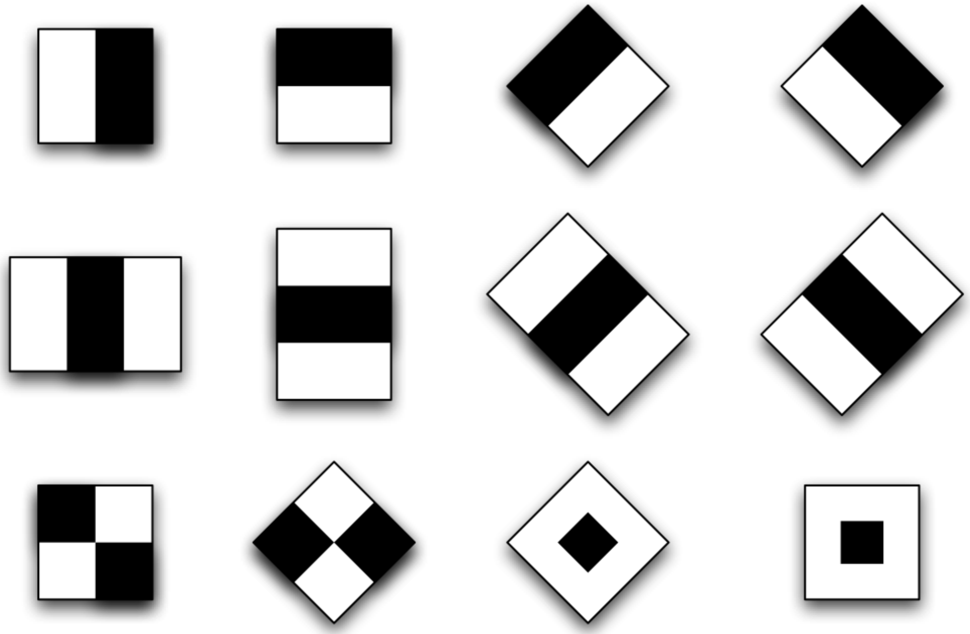
Haar Wavelet Transform
(low frequency projection)



Haar Wavelets

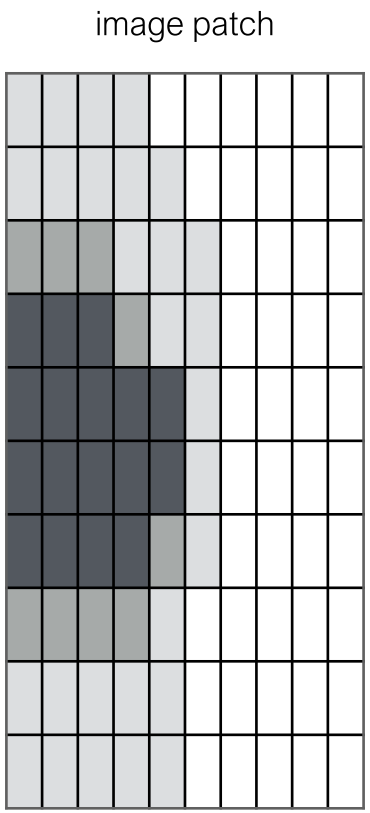
(actually, Haar-like features)

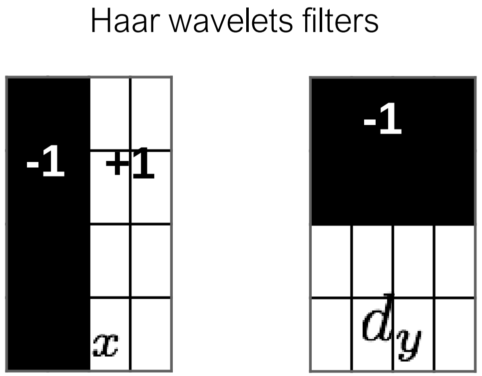
Use responses of a bank of filters as a descriptor



Computing Haar wavelet responses

1. Haar wavelet responses can be computed with filtering.



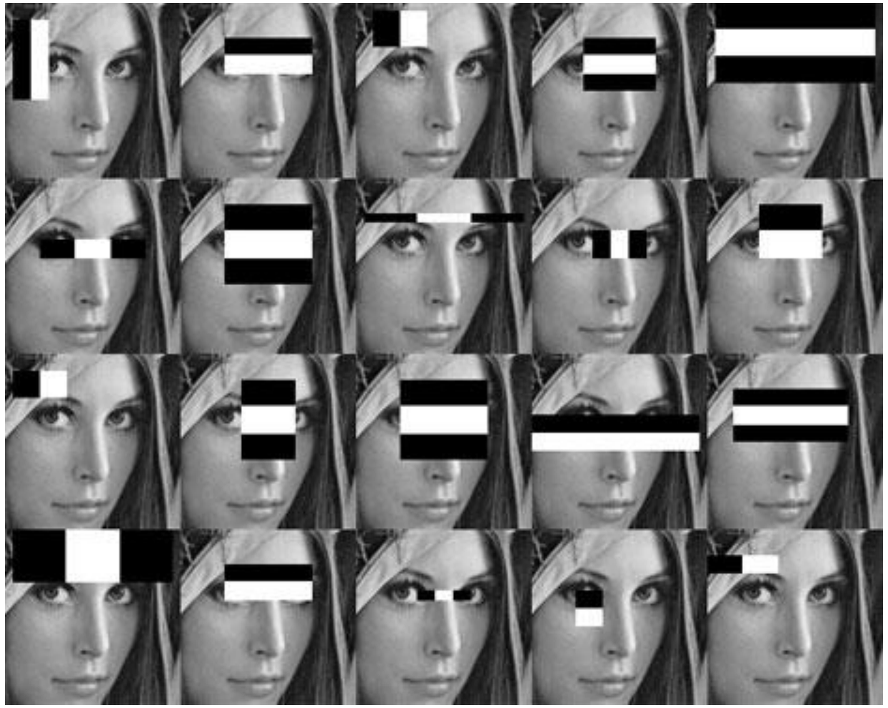


2. Haar wavelet responses can be computed efficiently (in constant time) with integral images.

Computing Haar wavelet responses

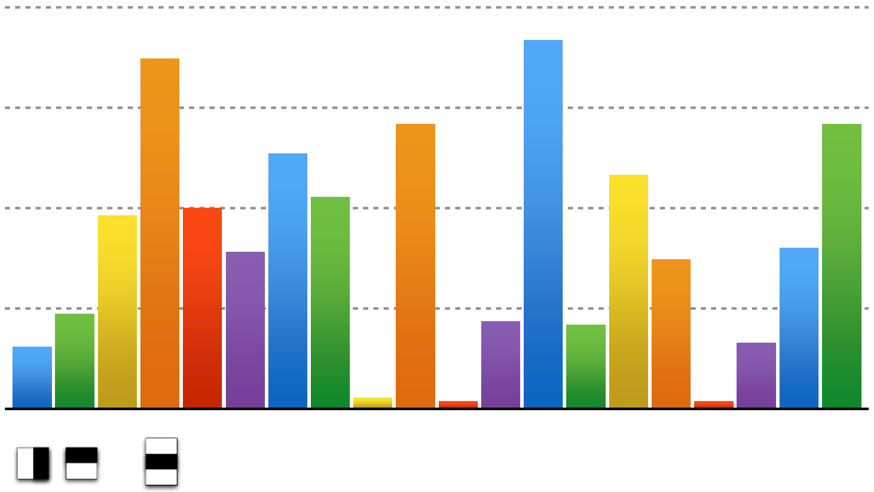
Given an image patch, compute filter responses

filter bank (20 Haar wavelet filters)



Responses are usually computed at specified location as a face patch descriptor

vector of filter responses



The image correspondence pipeline

1. Feature point detection
    - Detect corners using the Harris corner detector.

2. Feature point description
    - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching and homography estimation

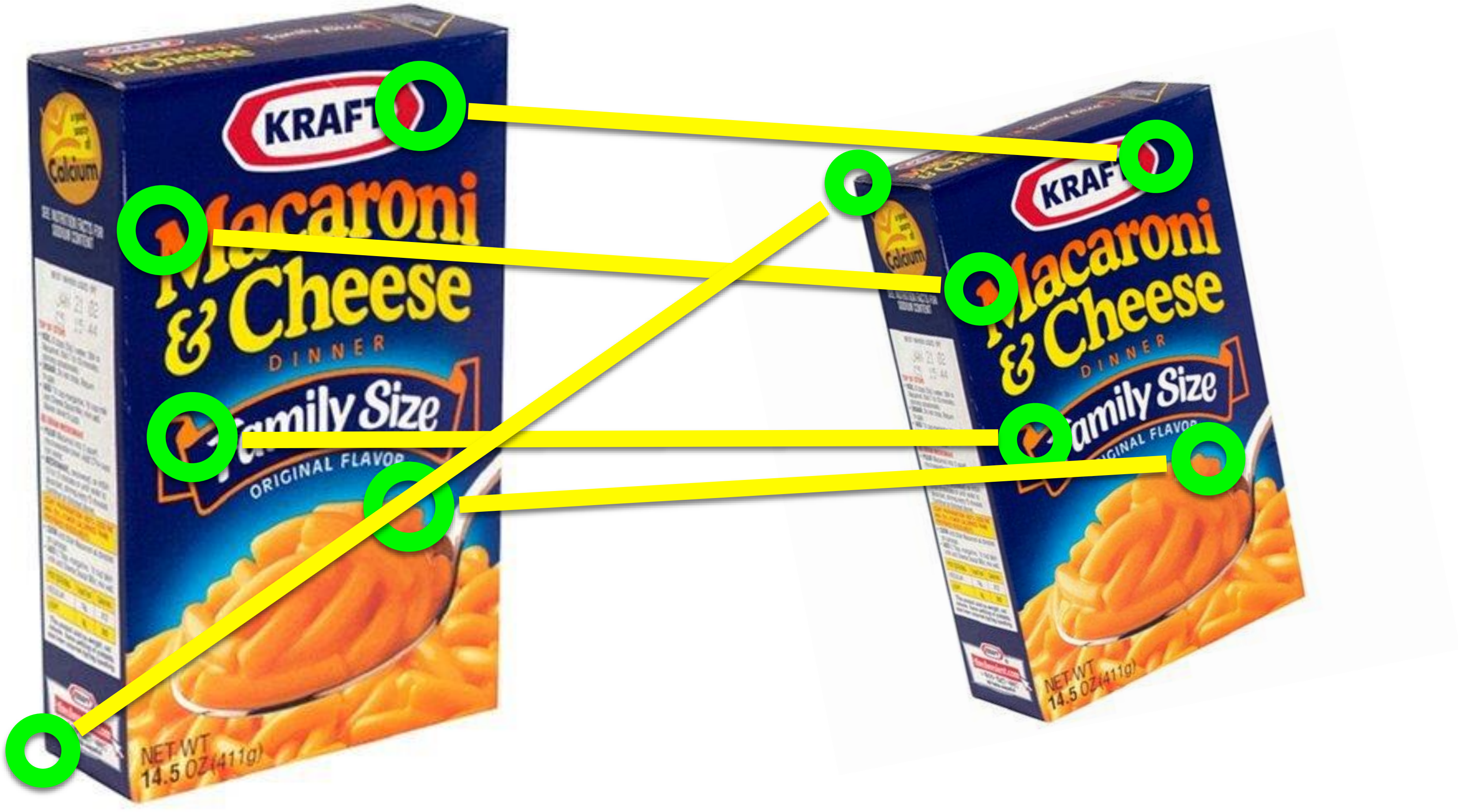
The image correspondence pipeline

1. Feature point detection
    - Detect corners using the Harris corner detector.

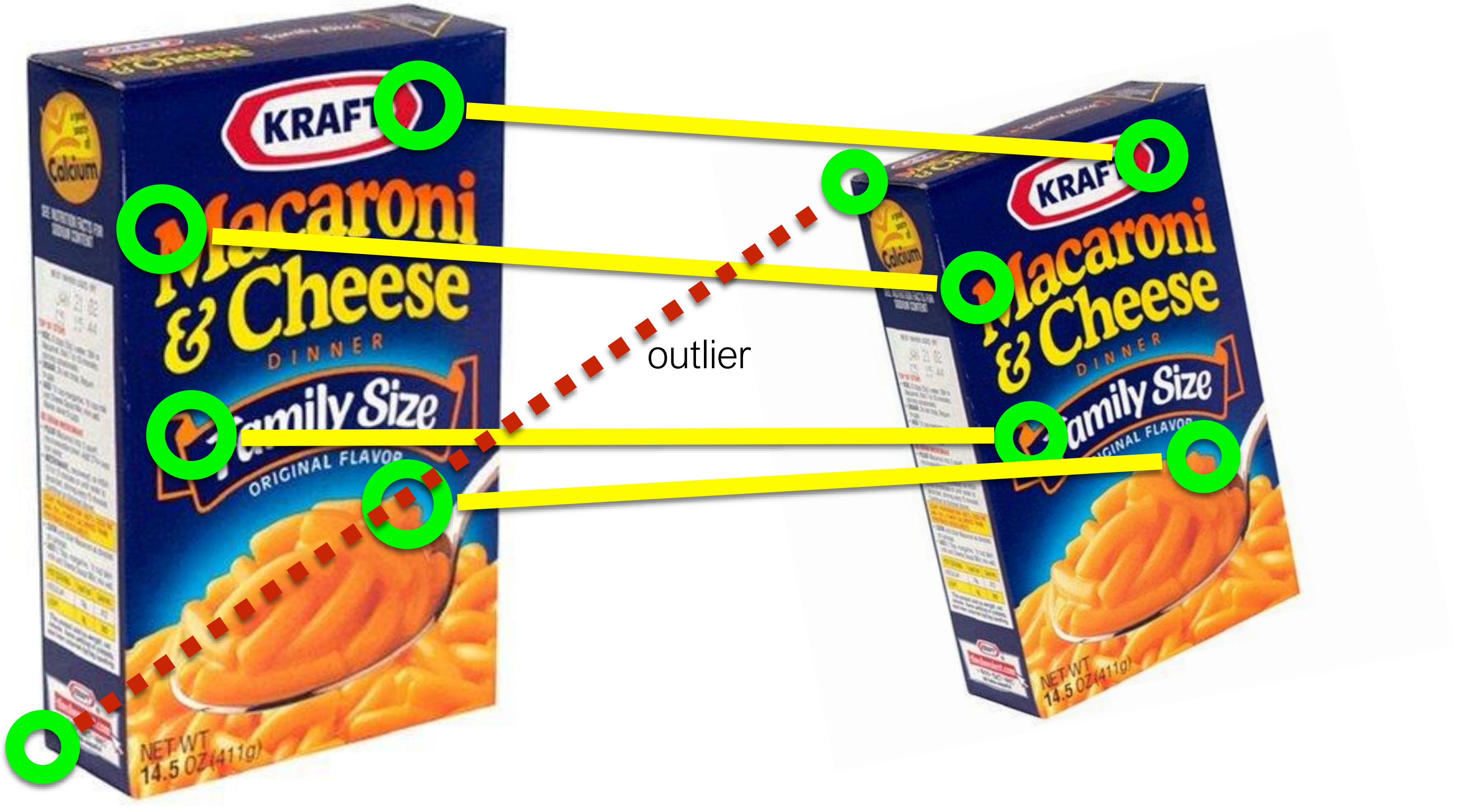
2. Feature point description
    - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching and homography estimation

Up to now, we’ve assumed correct correspondences



What if there are mismatches?



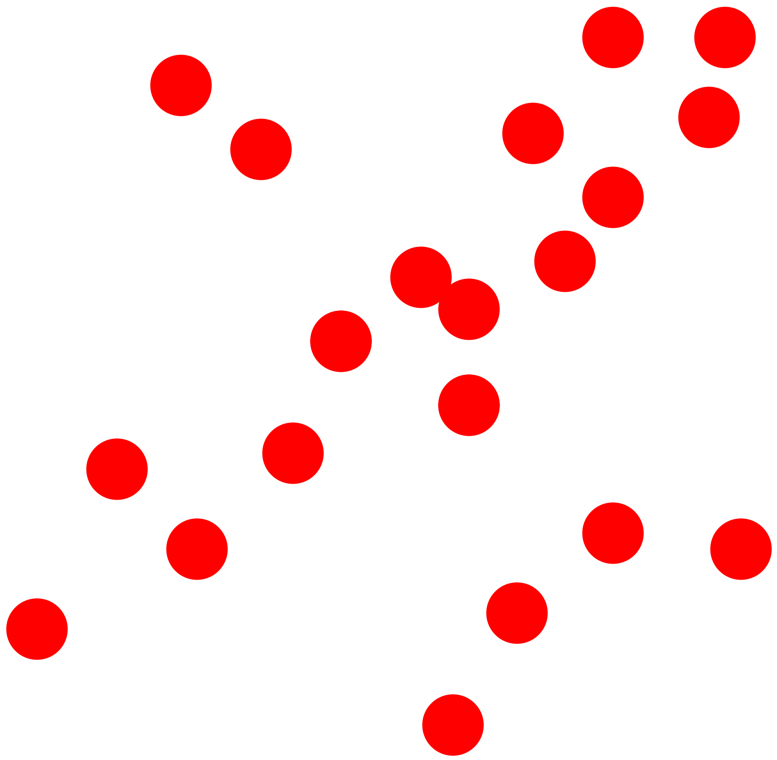
How would you find just the inliers?

RANSAC

RANdom **SA**mple **C**onsensus

[Fischler & Bolles in ’81]

Fitting lines
(with outliers)

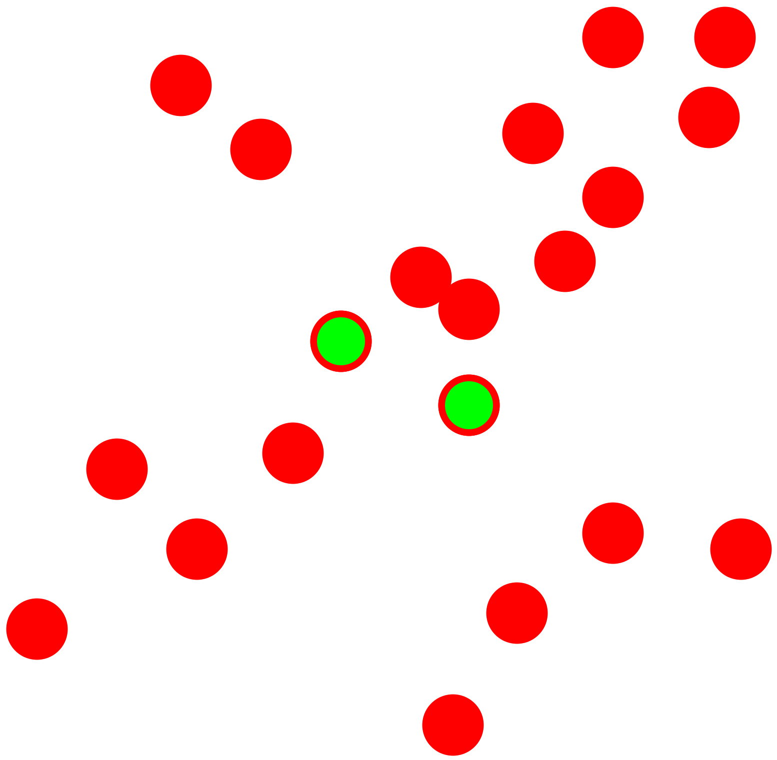


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

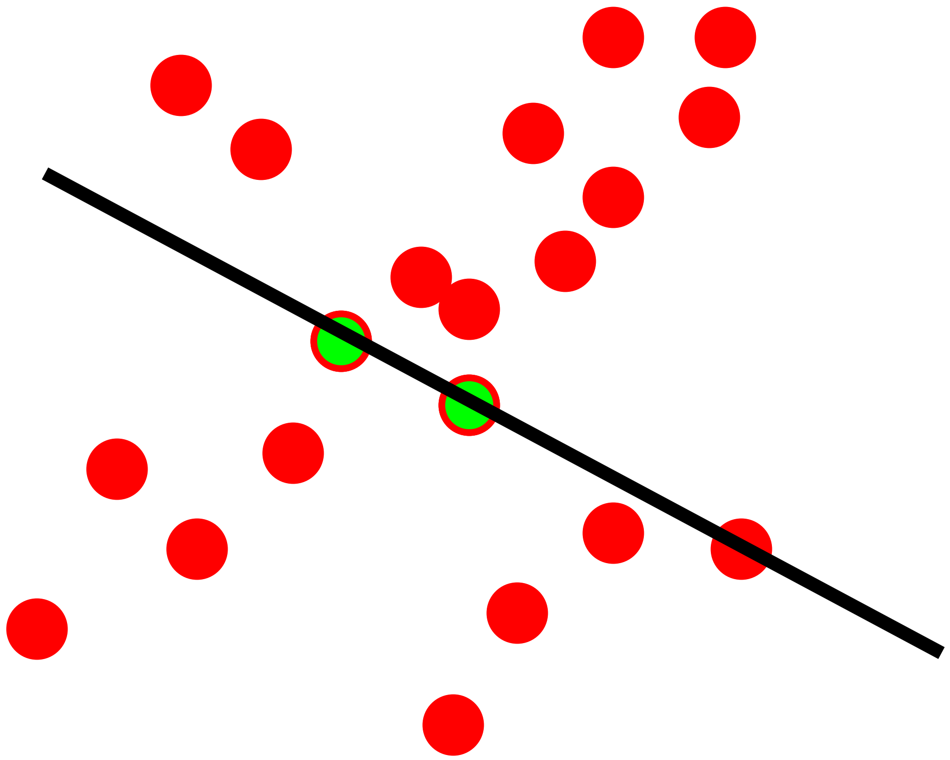


Algorithm:

1. **Sample (randomly) the number of points required to fit the model**
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



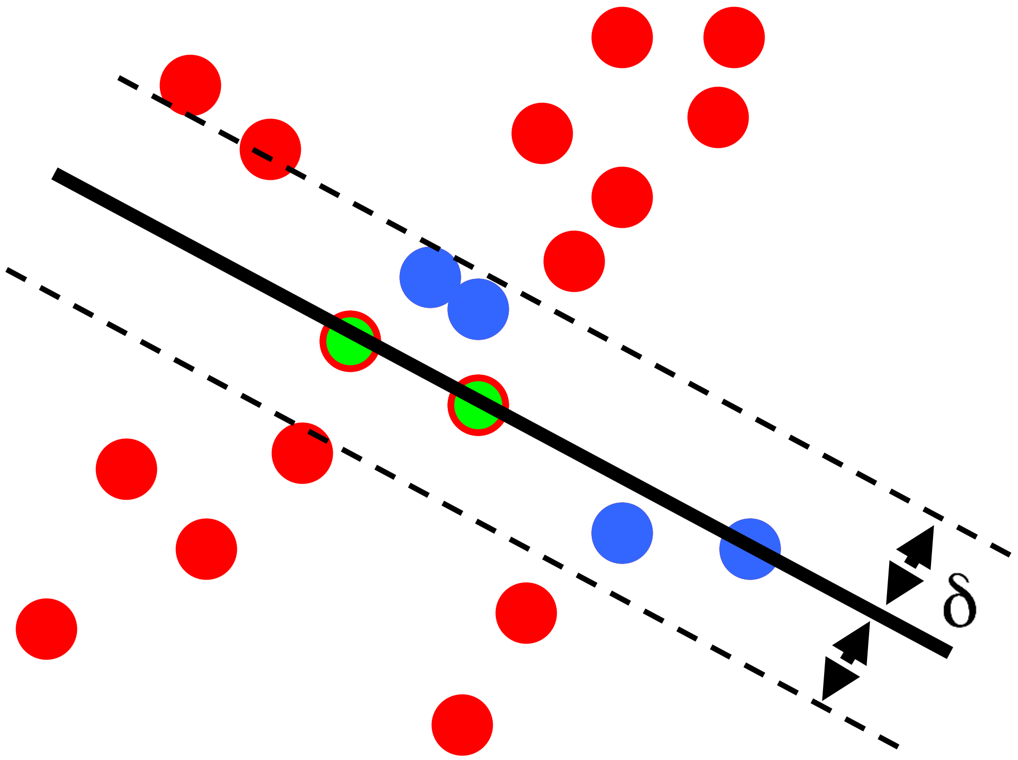
Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. **Solve for model parameters using samples**
3. Score by the fraction of inliers within a preset threshold of the model

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)

$N_I = 6$

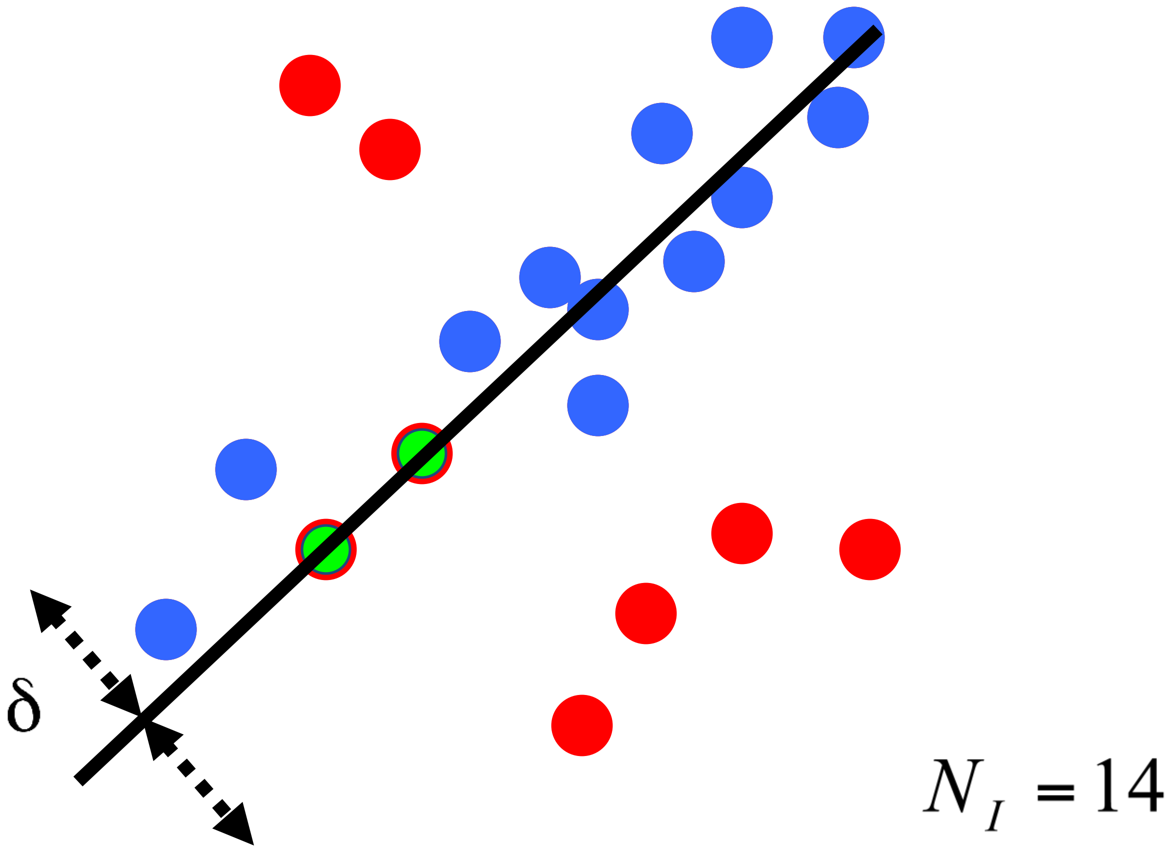


Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. **Score by the fraction of inliers within a preset threshold of the model**

Repeat 1-3 until the best model is found with high confidence

Fitting lines
(with outliers)



Algorithm:

1. Sample (randomly) the number of points required to fit the model
2. Solve for model parameters using samples
3. Score by the fraction of inliers within a preset threshold of the model

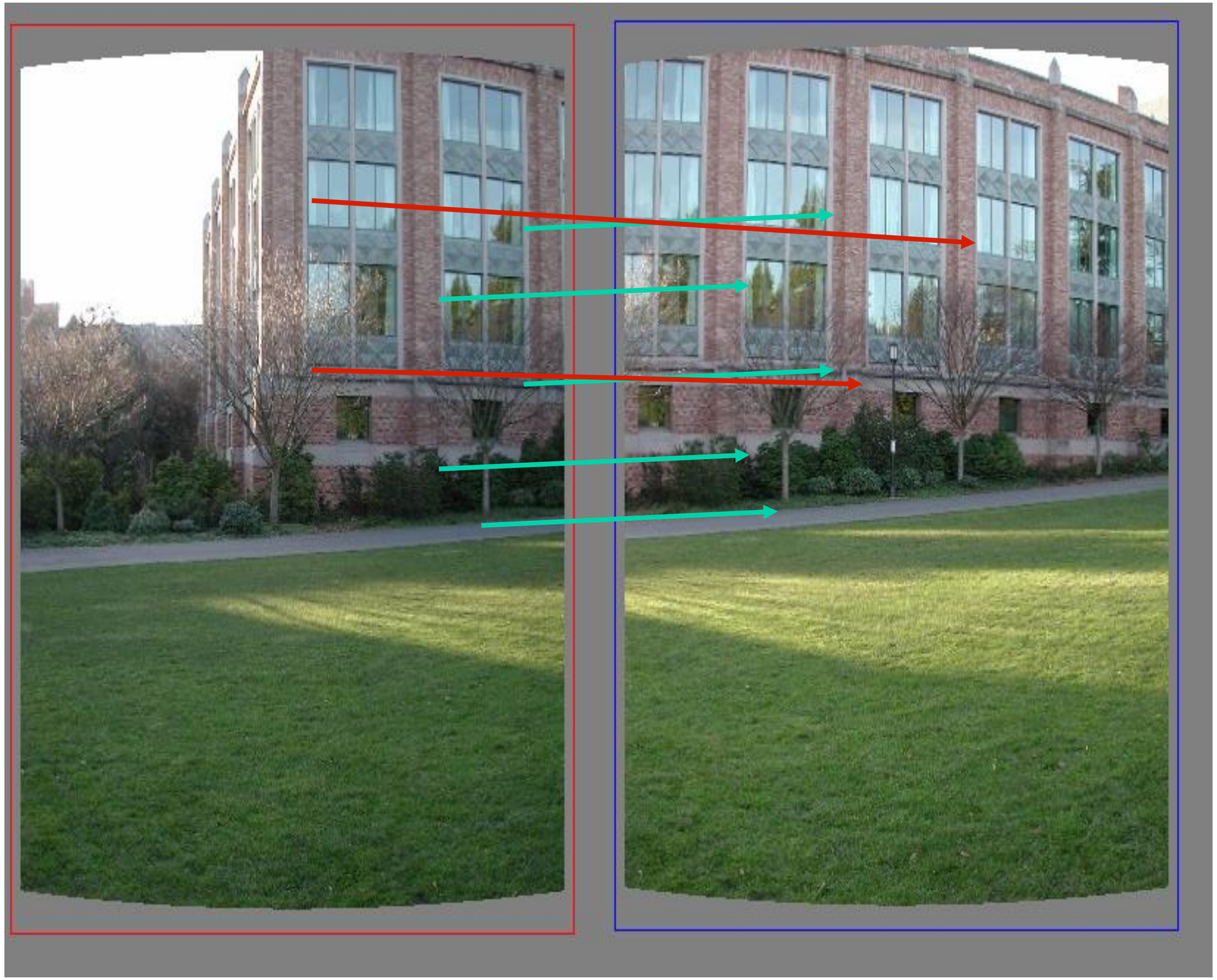
Repeat 1-3 until the best model is found with high confidence

How to choose parameters?

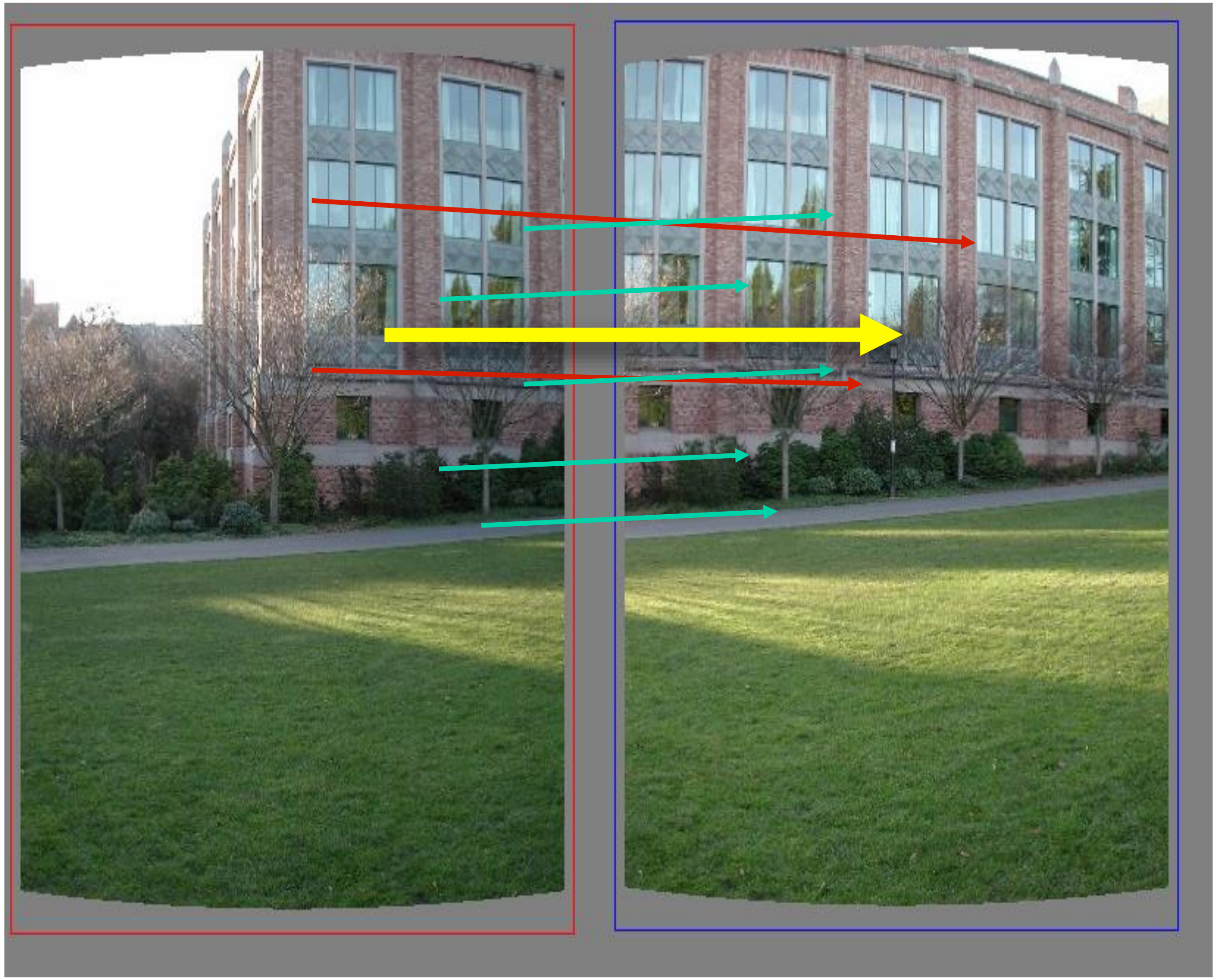
- Number of samples N
  - Choose N so that, with probability p, at least one random sample is free from outliers (e.g. p=0.99) (outlier ratio: e)
- Number of sampled points s
  - Minimum number needed to fit the model
- Distance threshold δ
  - Choose δ so that a good point with noise is likely (e.g., prob=0.95) within threshold
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$

$$N = \frac{\log(1-p)}{\log\left(1-(1-e)^s\right)}$$

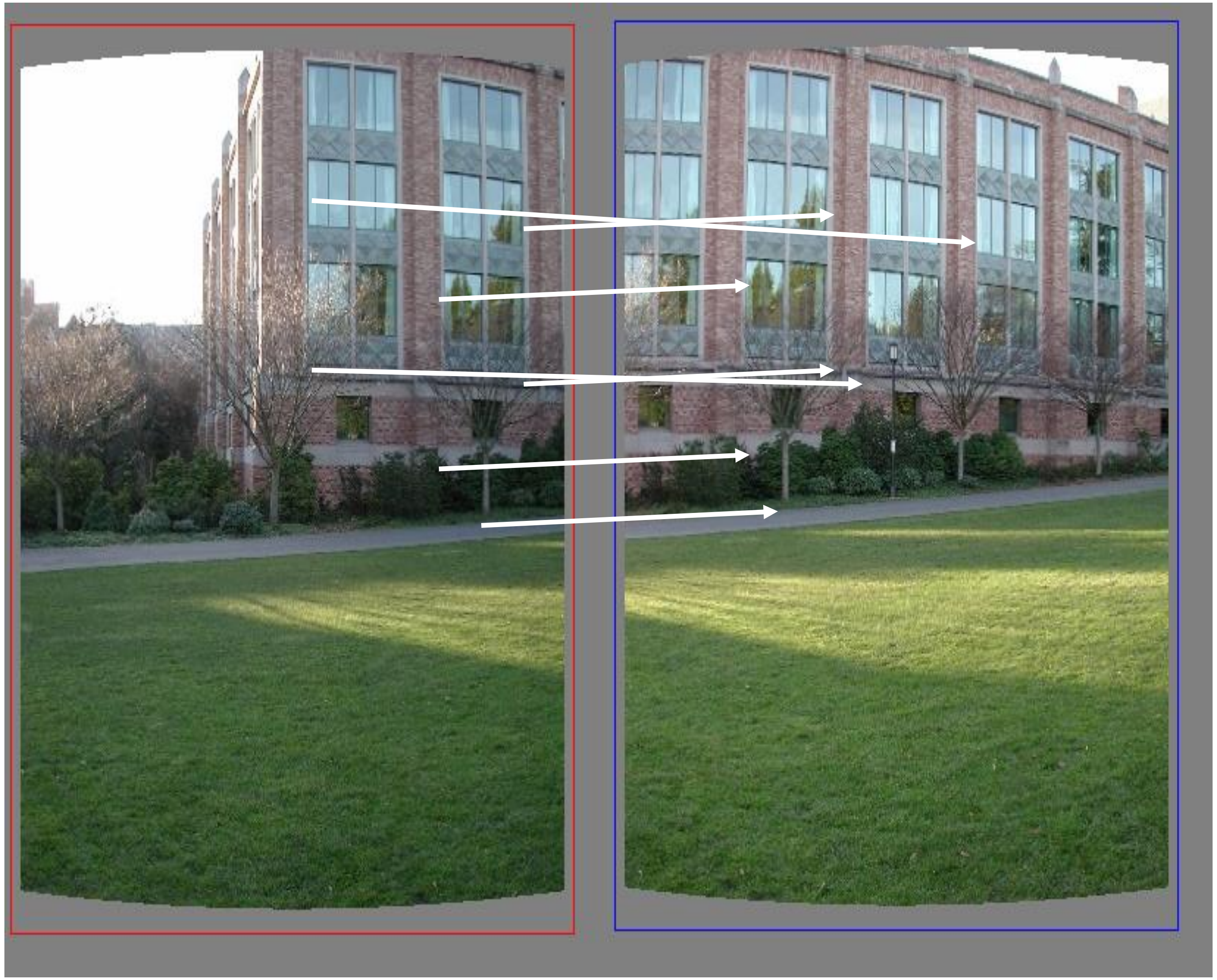
	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177



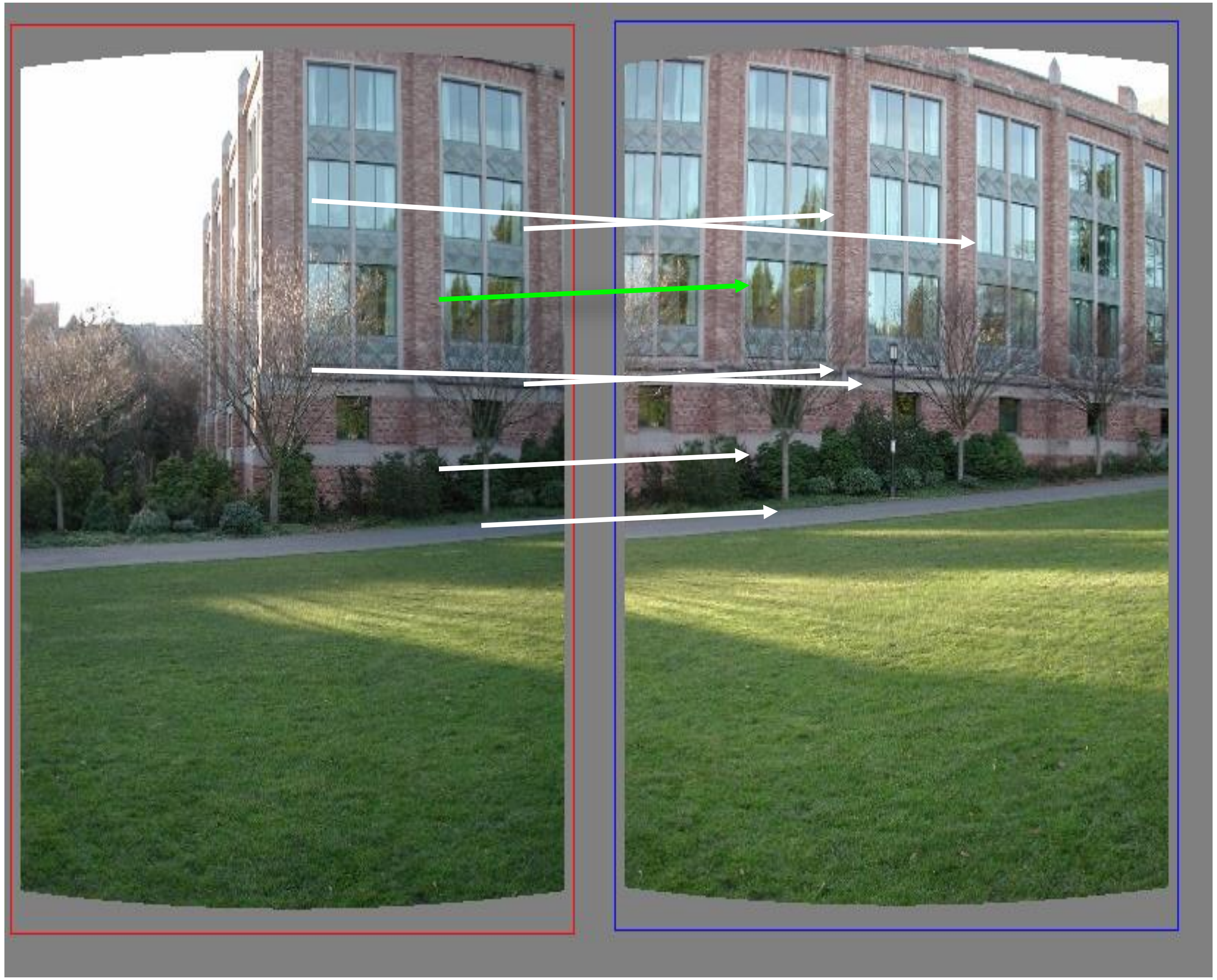
Matched points



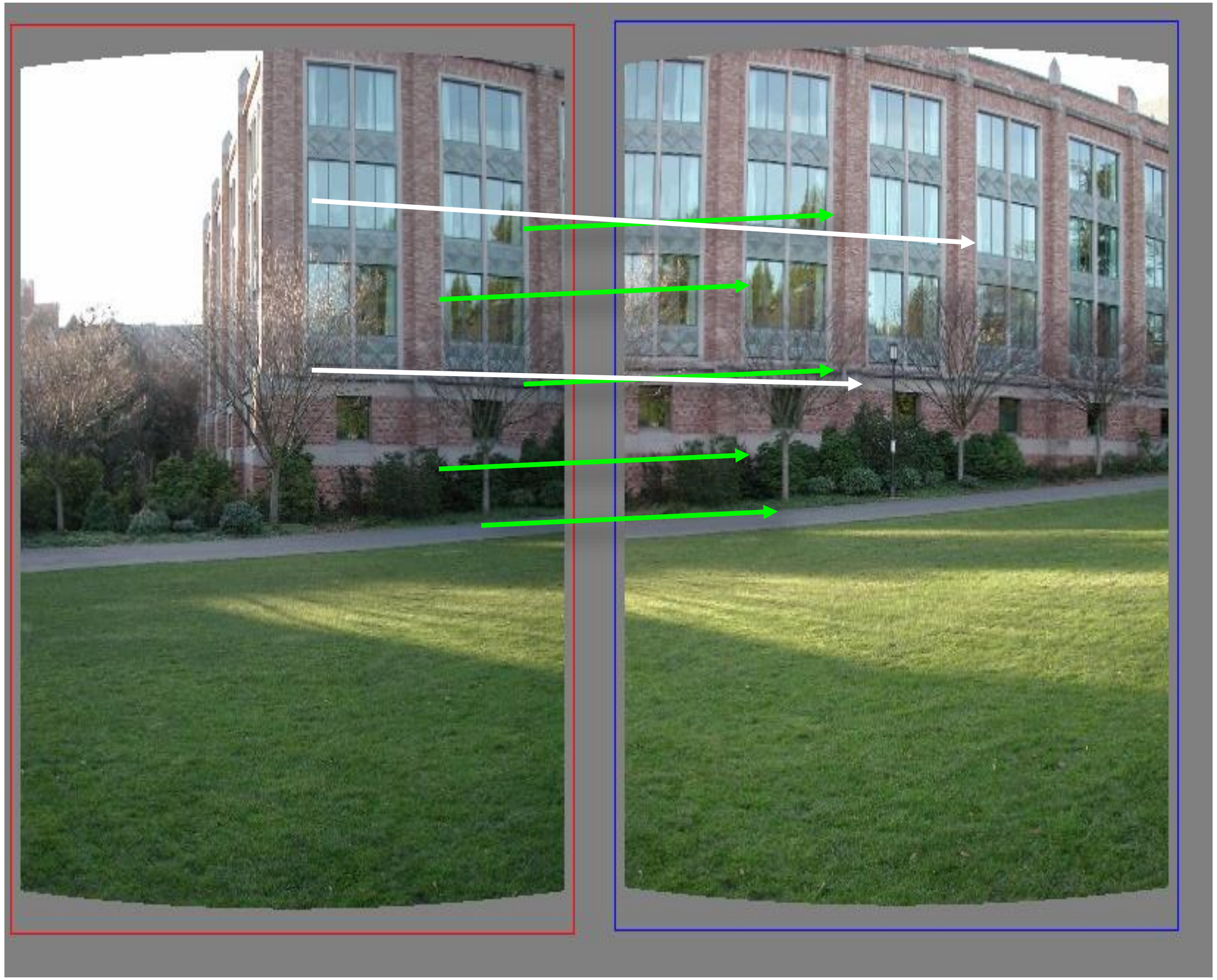
Least Square fit finds the ‘average’ transform



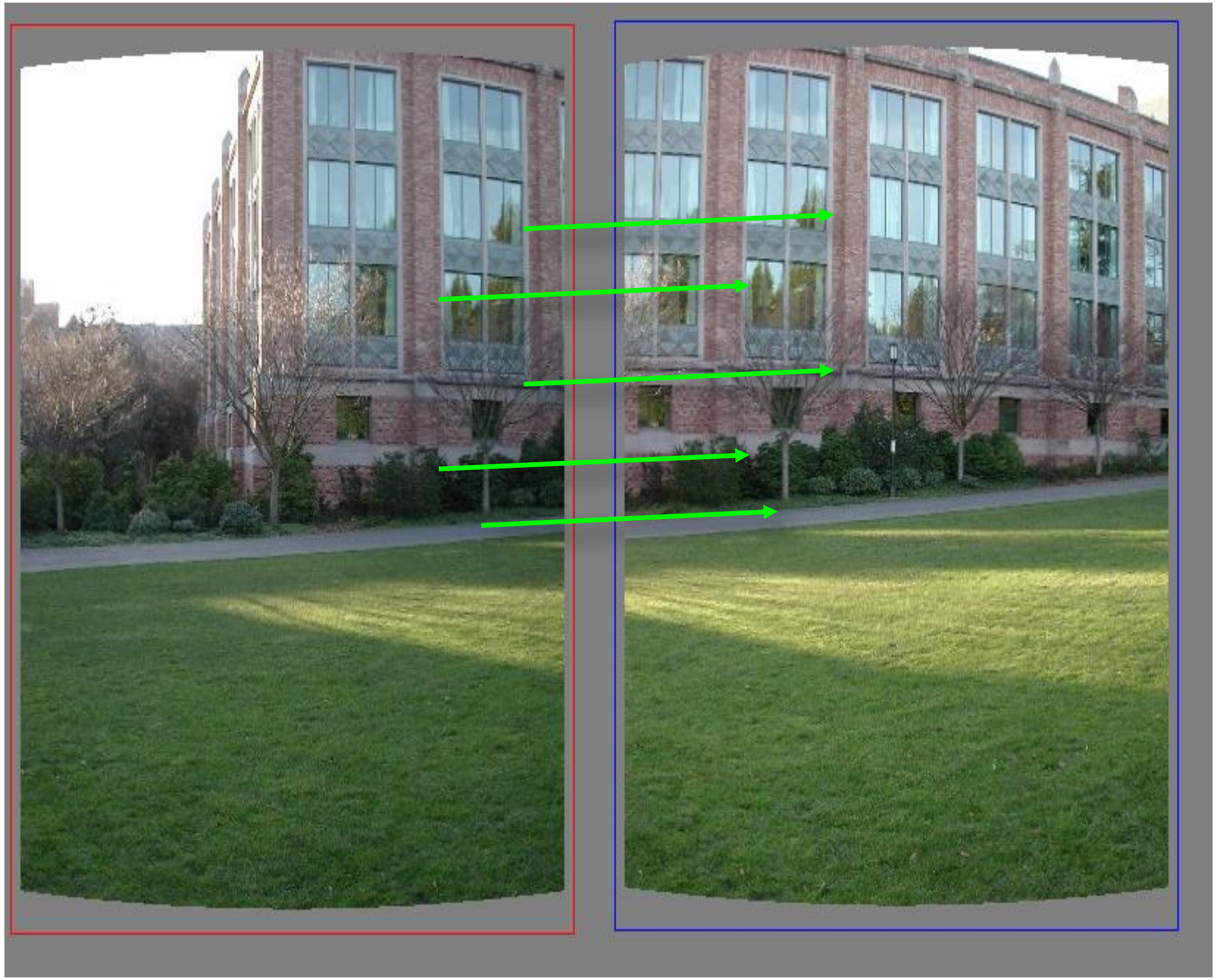
RANSAC: Use one correspondence, find inliers



RANSAC: **Use one correspondence**, find inliers



RANSAC: Use one correspondence, **find inliers**



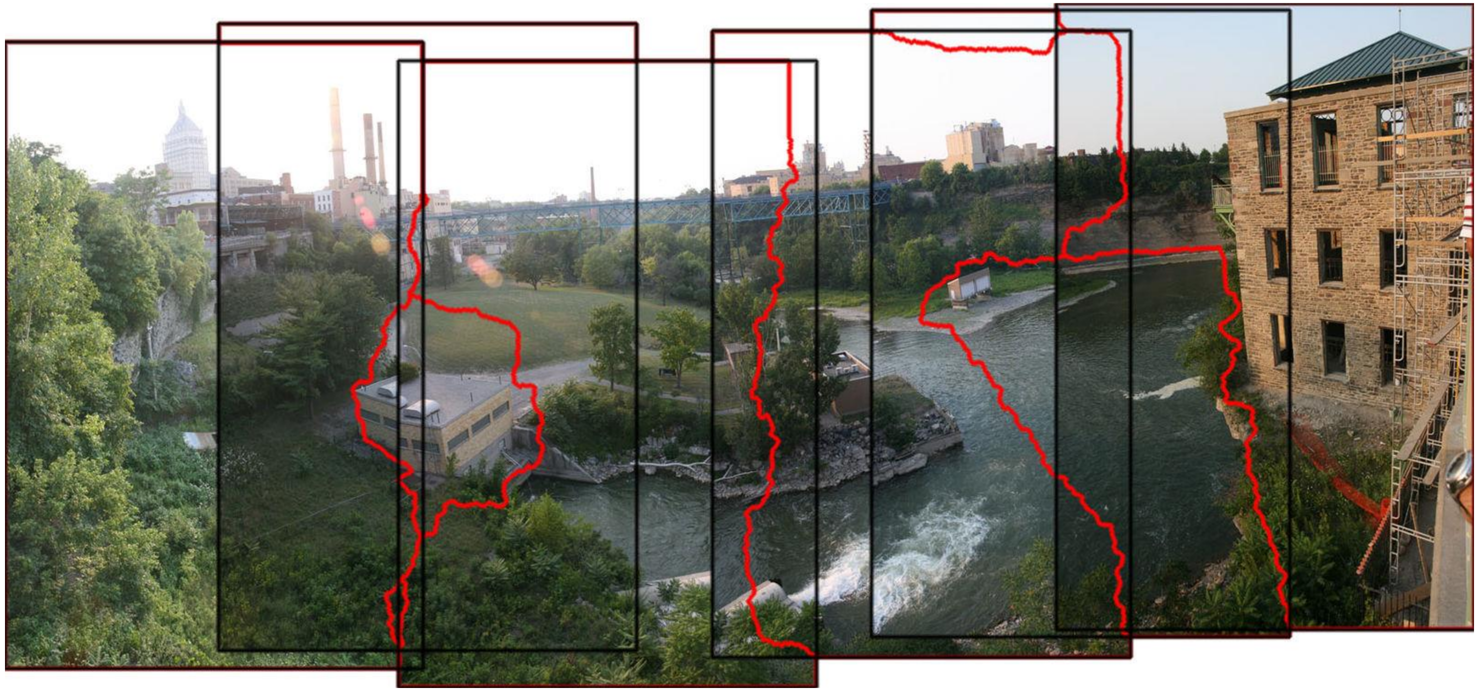
RANSAC: Use one correspondence, **find inliers**

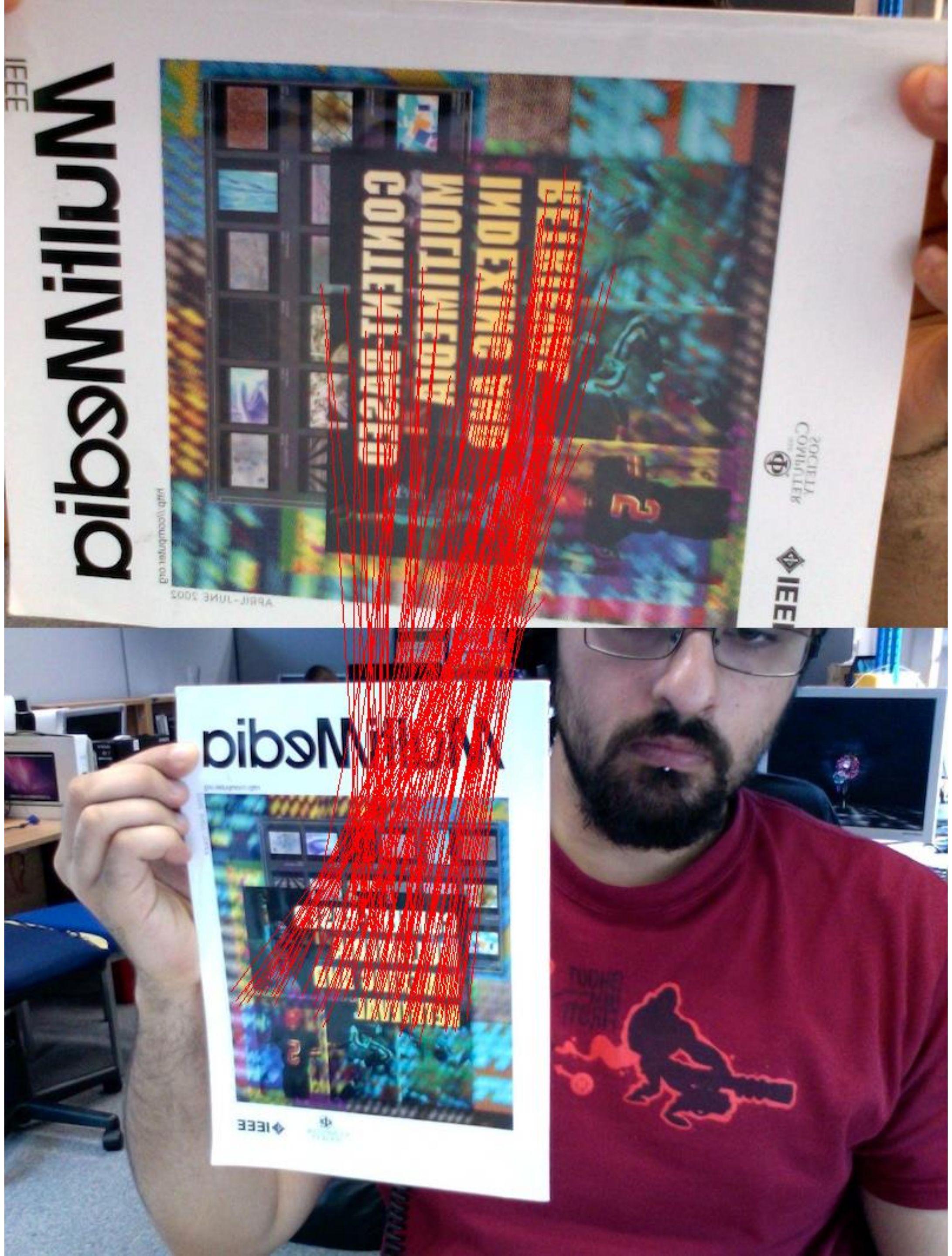
Estimating homography using RANSAC

- RANSAC loop
  1. Get four point correspondences (randomly)
  2. Compute homography H (DLT)
  3. Count inliers
  4. Keep if largest number of inliers
- Recompute H using all inliers

Why four point correspondences?

Useful for…





The image correspondence pipeline

1. Feature point detection
   - Detect corners using the Harris corner detector.

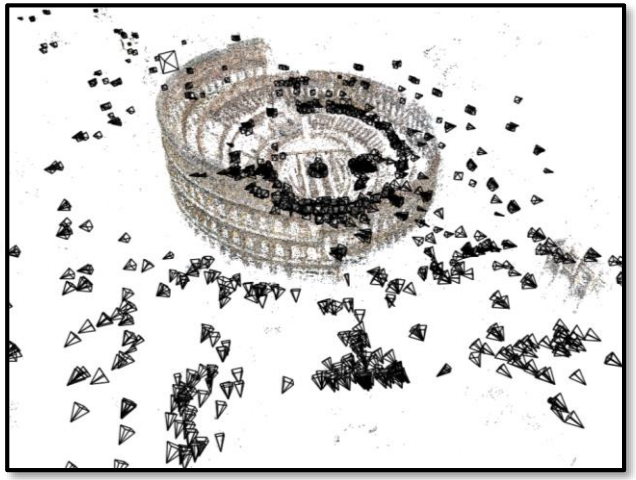
2. Feature point description
   - Describe features using the Multi-scale oriented patch descriptor.

3. Feature matching and homography estimation
   - Do both simultaneously using RANSAC.

Structure from motion

Structure from motion

- Given many images, how can we
  a) figure out where they were all taken from?
  b) build a 3D model of the scene?



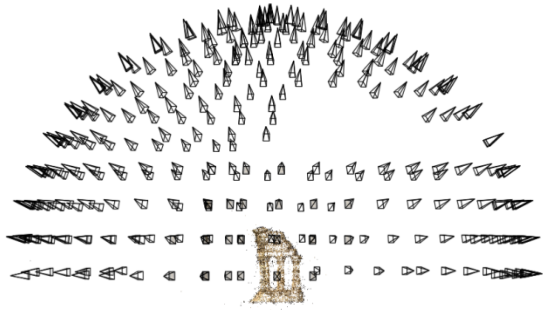
This is (roughly) the **structure from motion** problem

Structure from motion

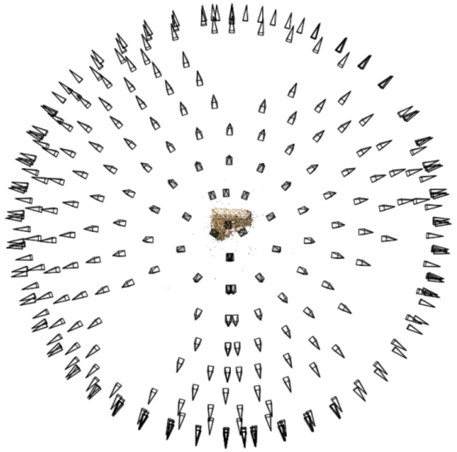
- Input: images with points in correspondence

  $p_{i,j} = (u_{i,j}, v_{i,j})$

Structure from motion



Reconstruction (side)



(top)

- Input: images with points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$

- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$ possibly $\mathbf{K}_j$

- Objective function: minimize *reprojection error*

Camera calibration & triangulation

- Suppose we know 3D points
  - And have matches between these points and an image
  - How can we compute the camera parameters?

- Suppose we have know camera parameters, each of which observes a point
  - How can we compute the 3D location of that point?

Structure from motion

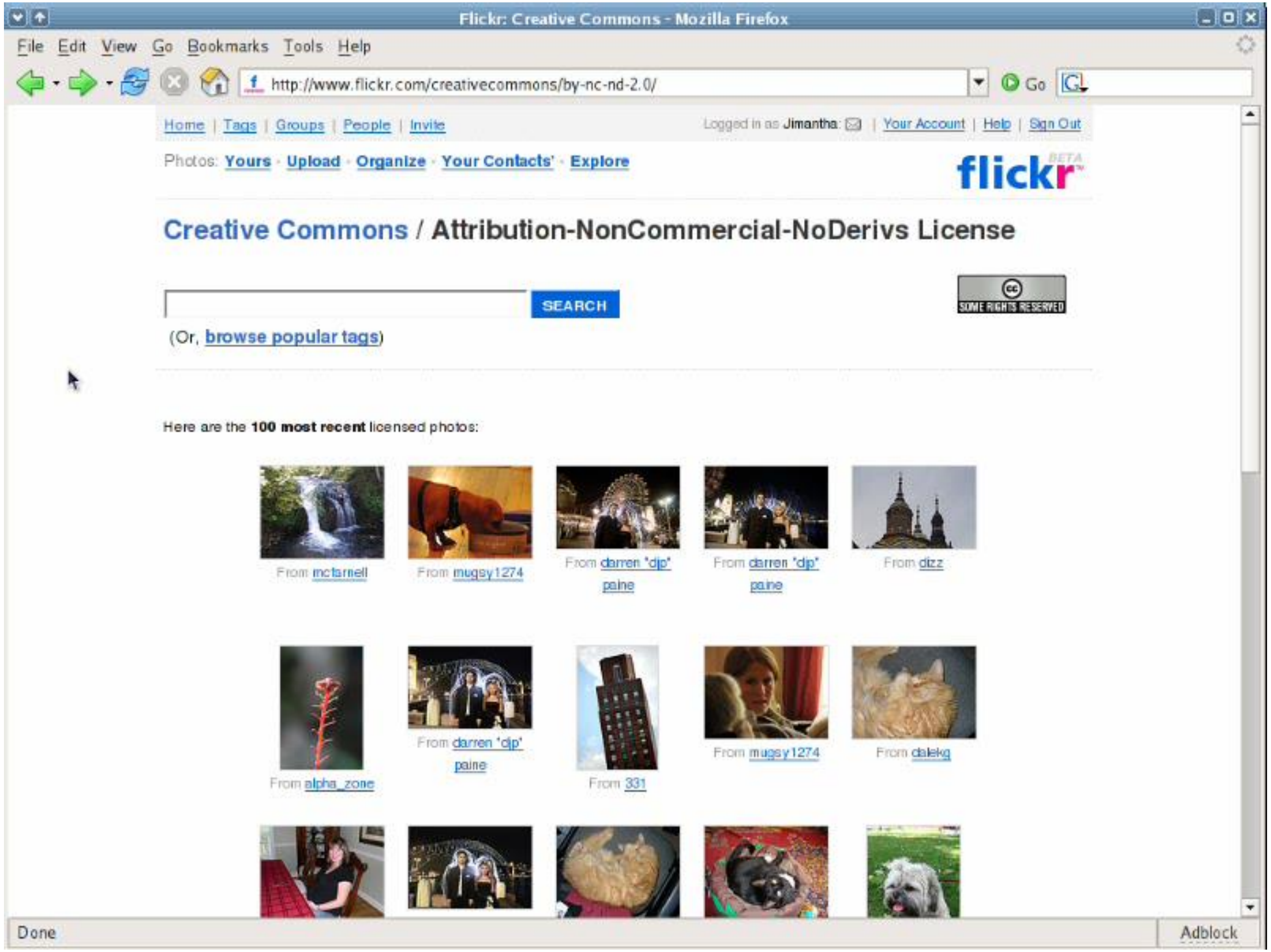
- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
  - (but solvable)

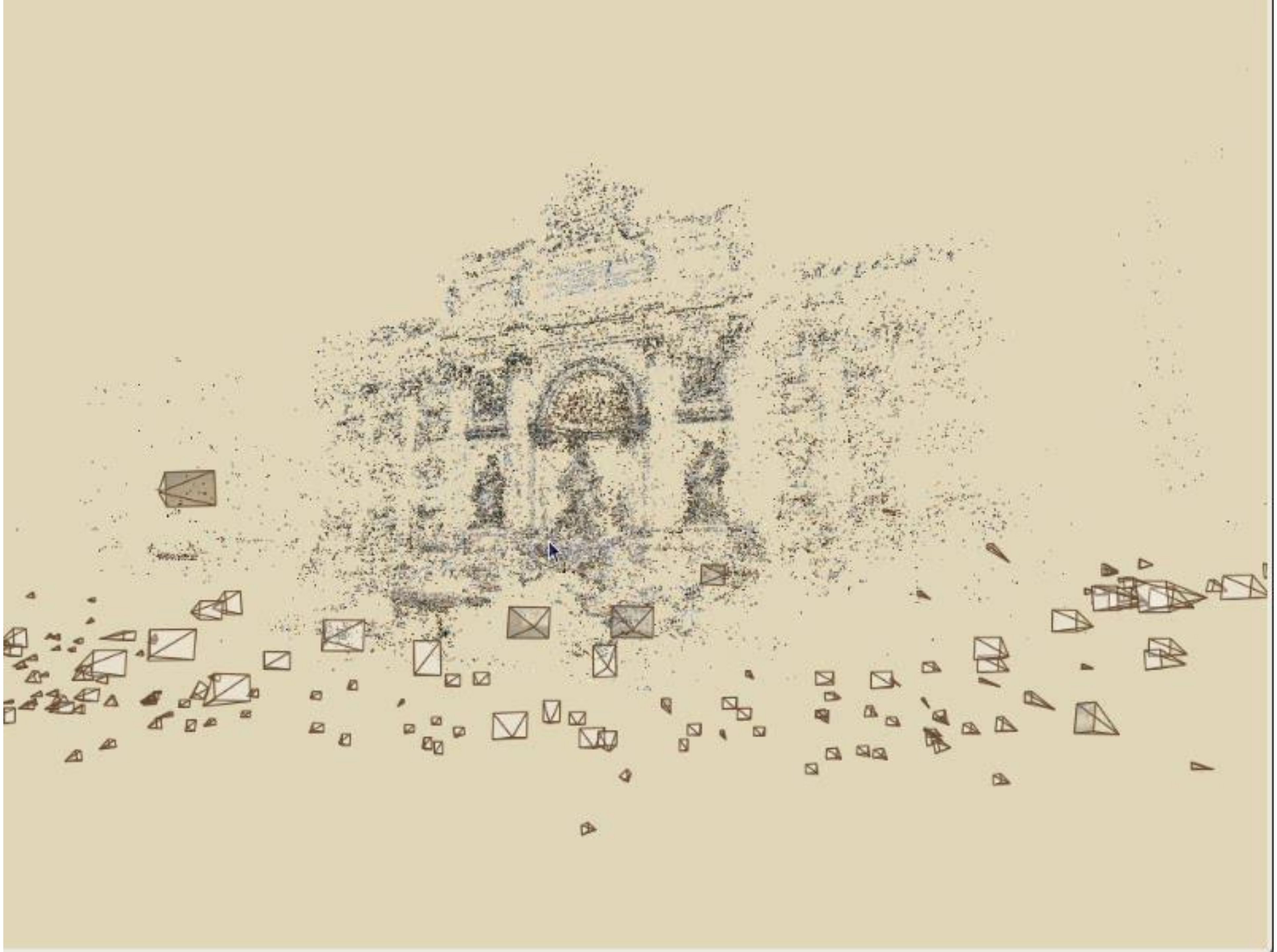
15,464

37,383

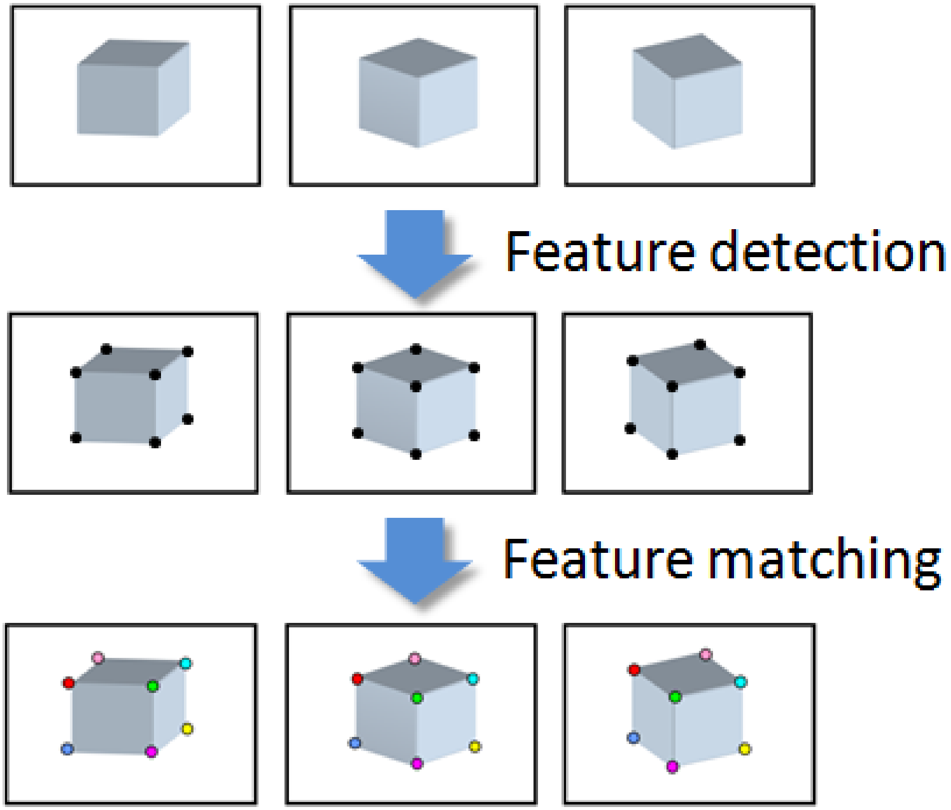
76,389

Standard way to view photos



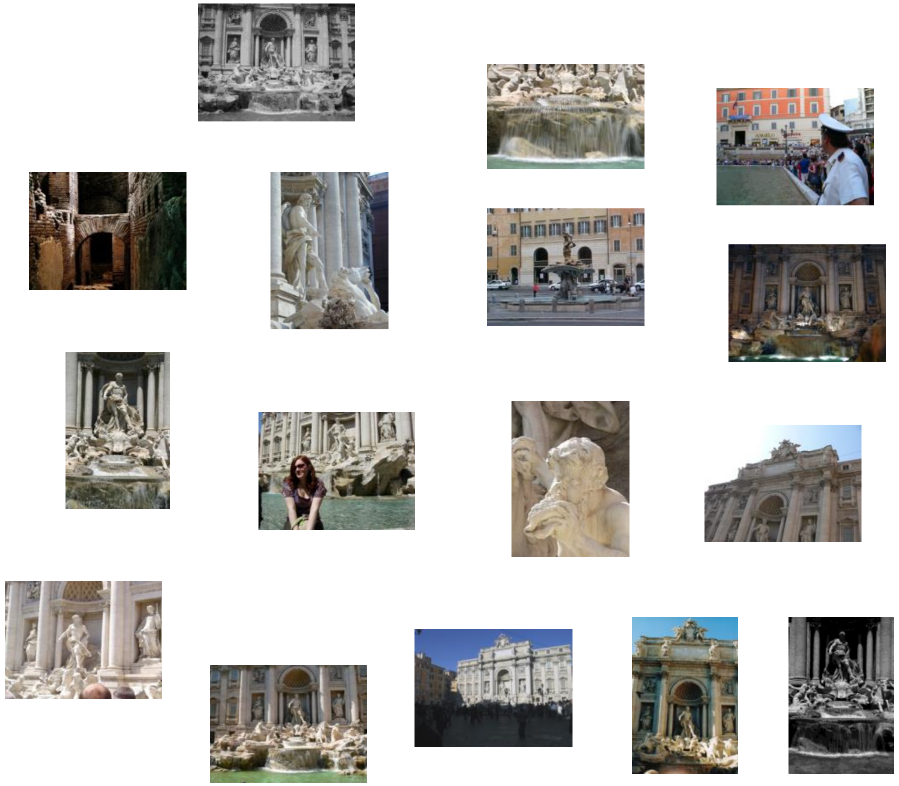


Input: Point correspondences



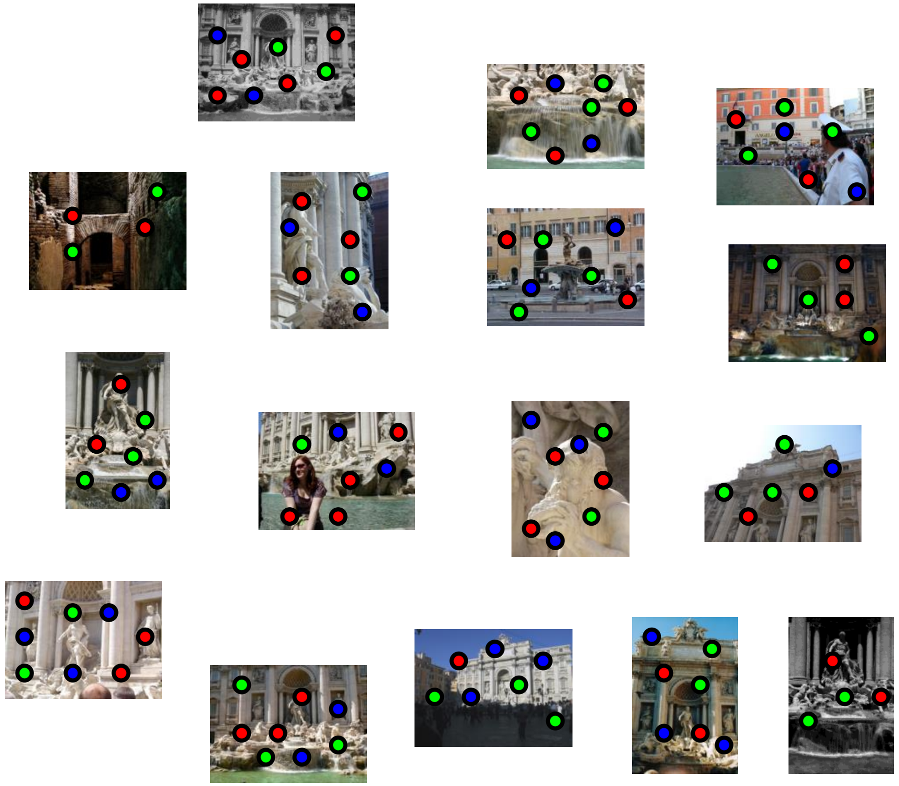
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



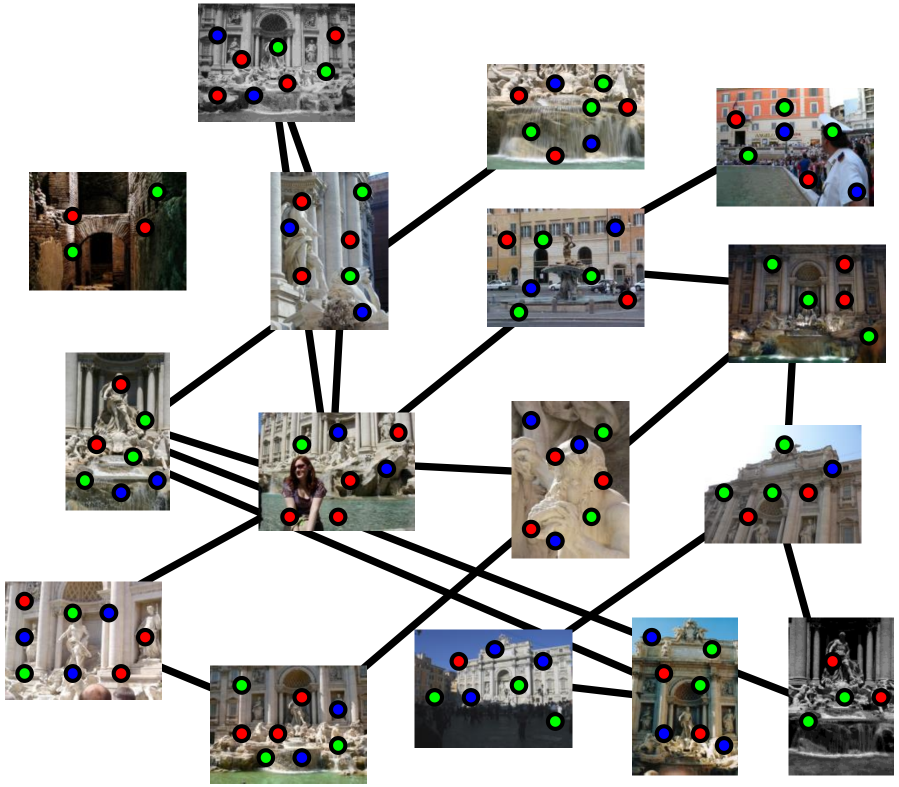
Feature description

Describe features using SIFT [Lowe, IJCV 2004]



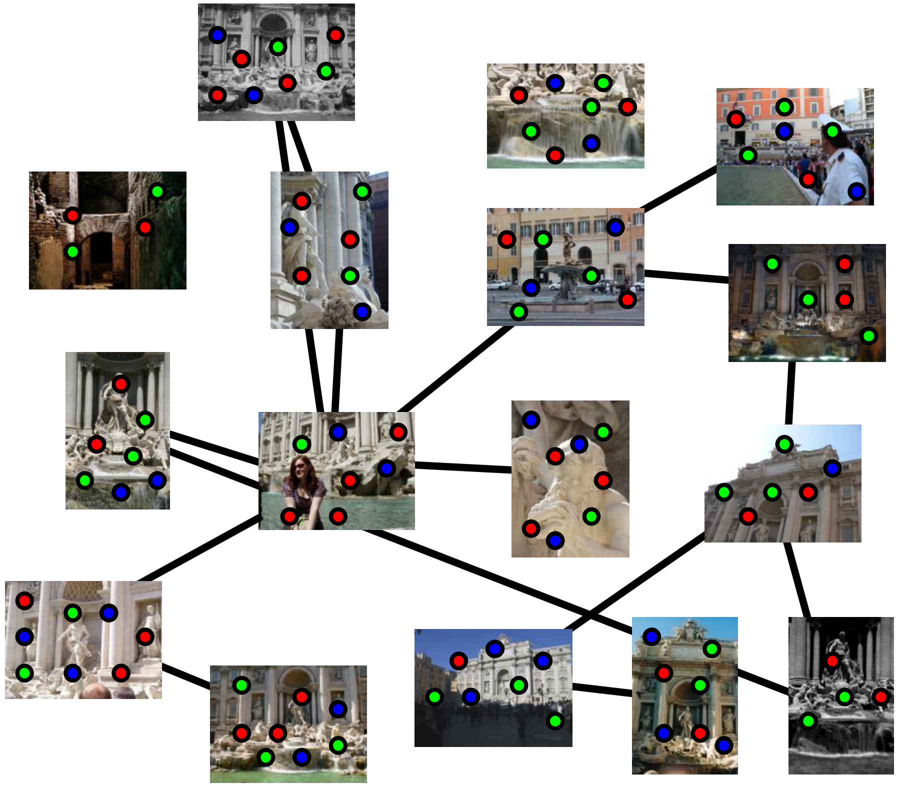
Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair



Correspondence estimation

- Link up pairwise matches to form connected components of matches across several images

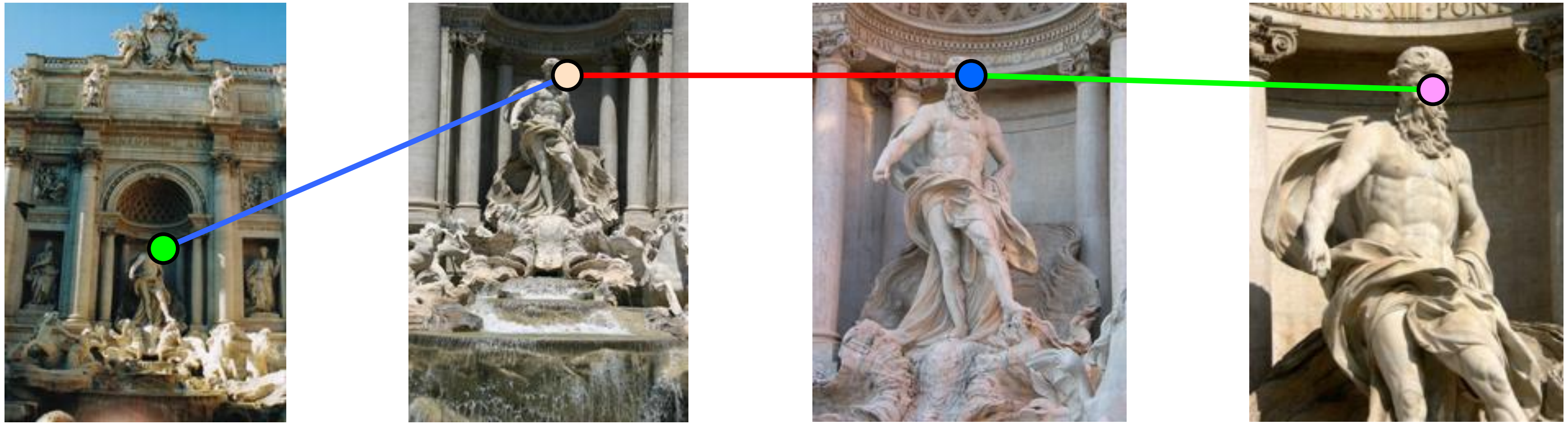
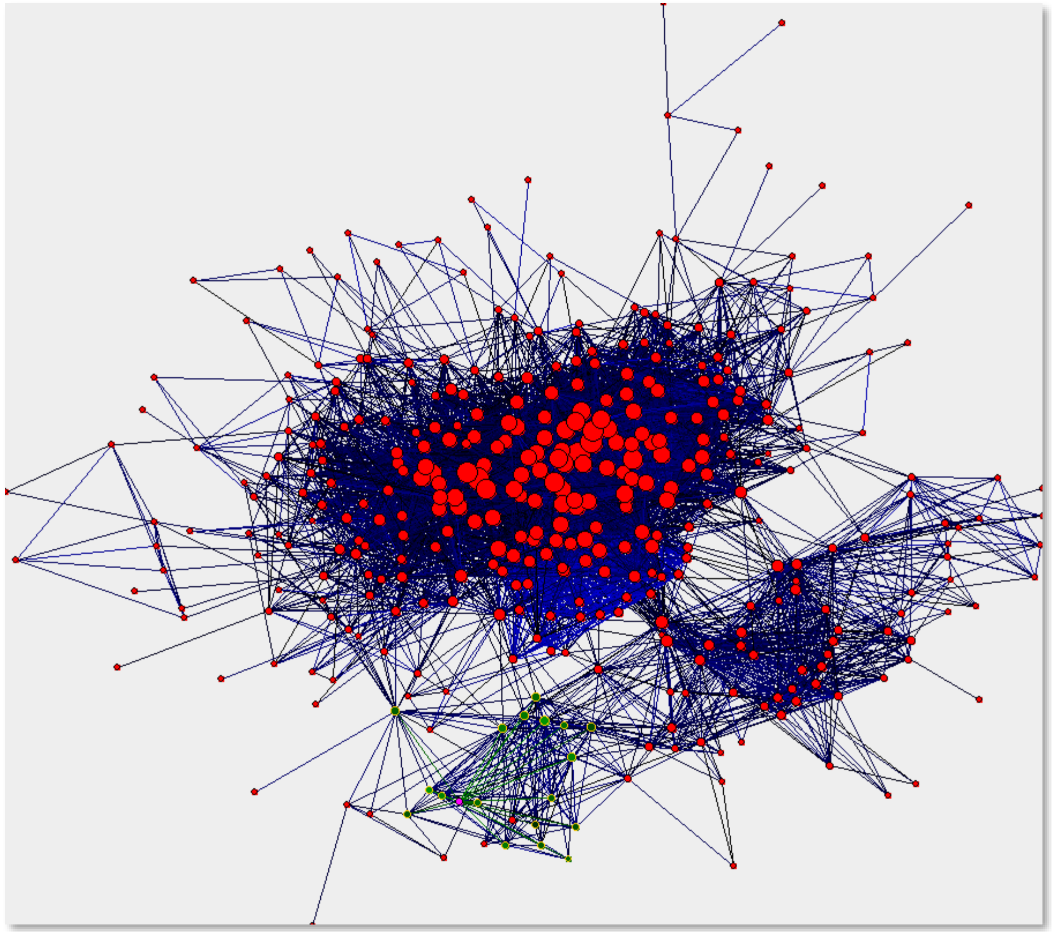


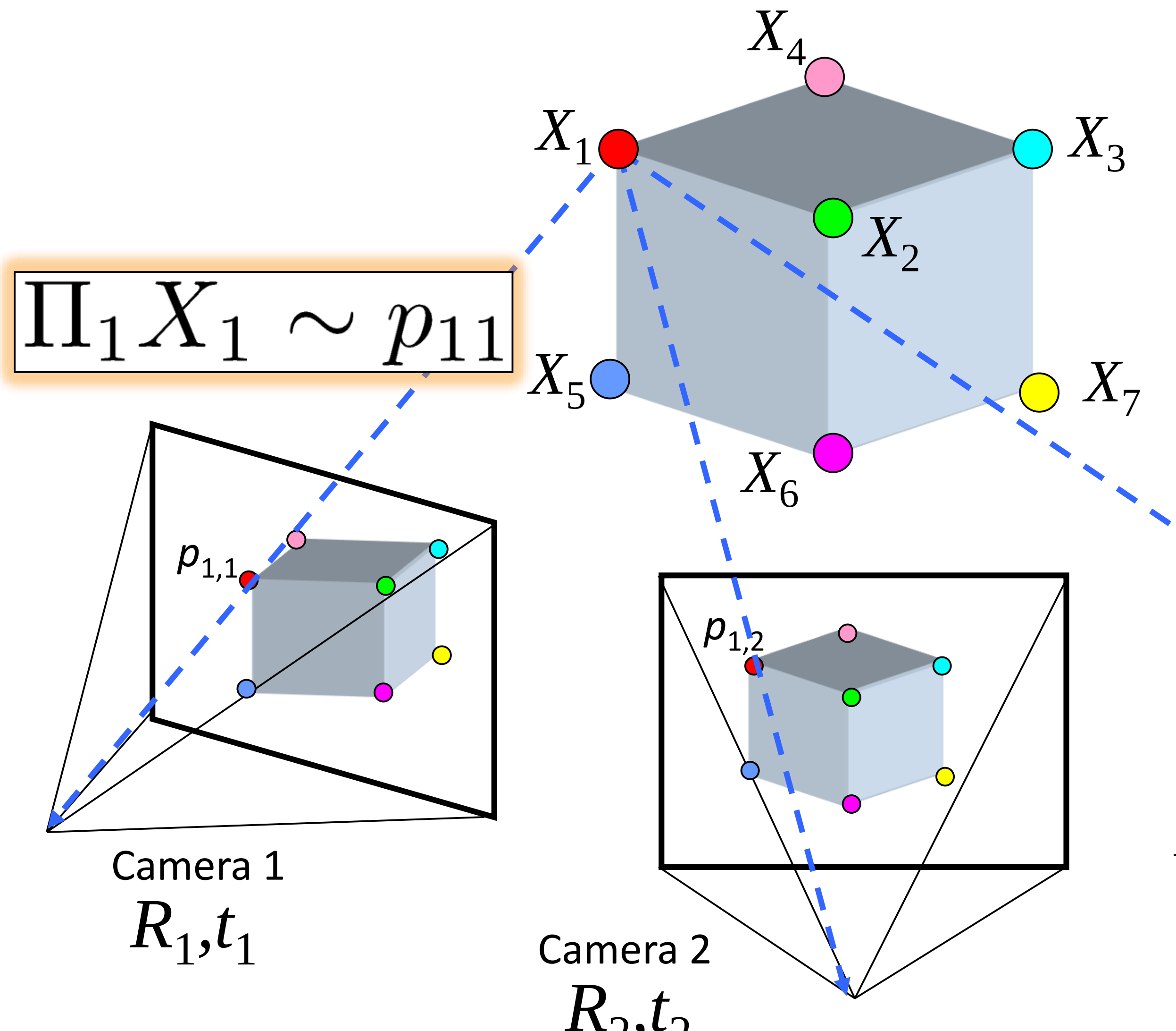
Image 1 Image 2 Image 3 Image 4

Image connectivity graph



(graph layout produced using the Graphviz toolkit: http://www.graphviz.org/)

Structure from motion

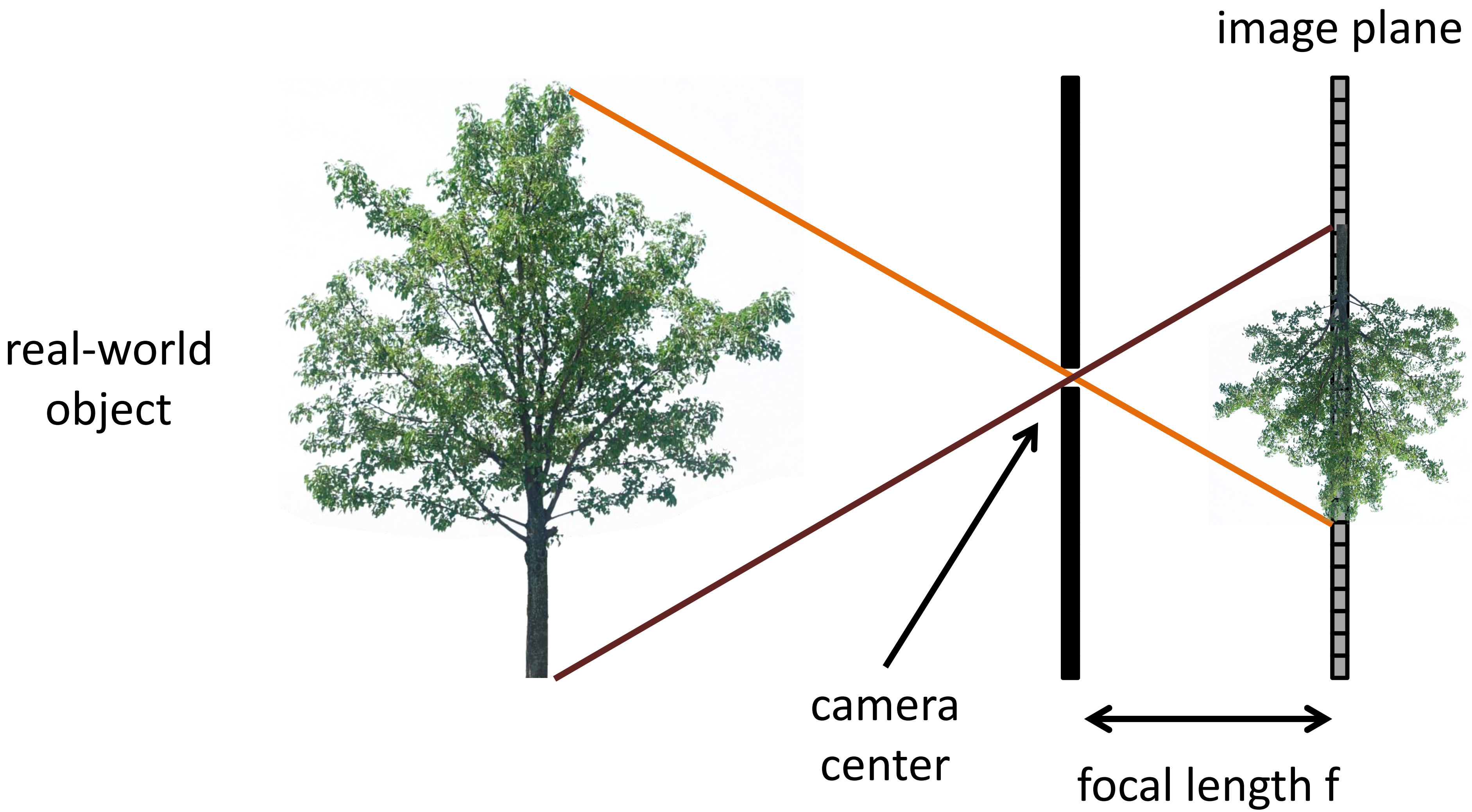


minimize

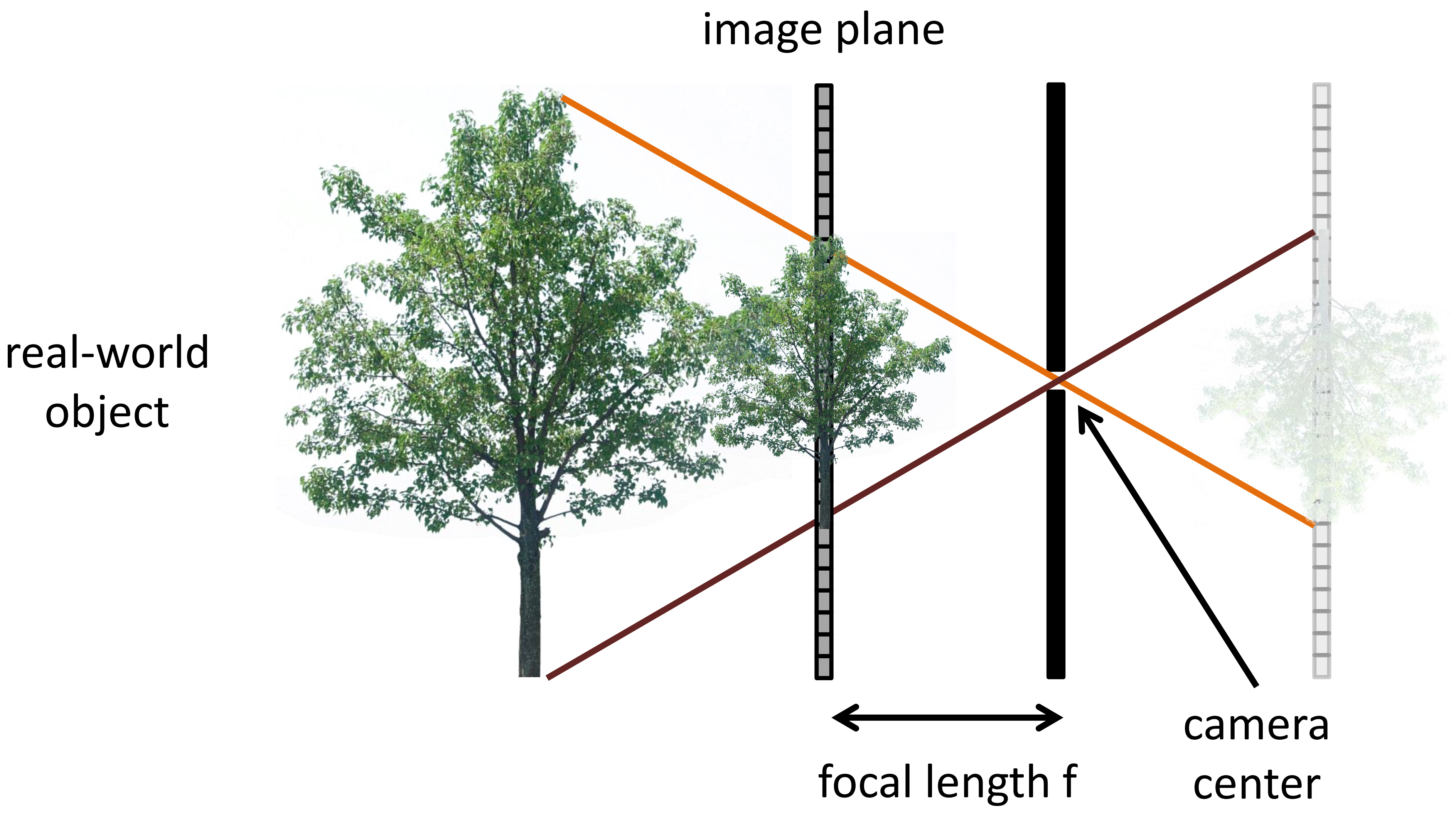
$$g(\mathbf{R}, \mathbf{T}, \mathbf{X})$$

non-linear least squares

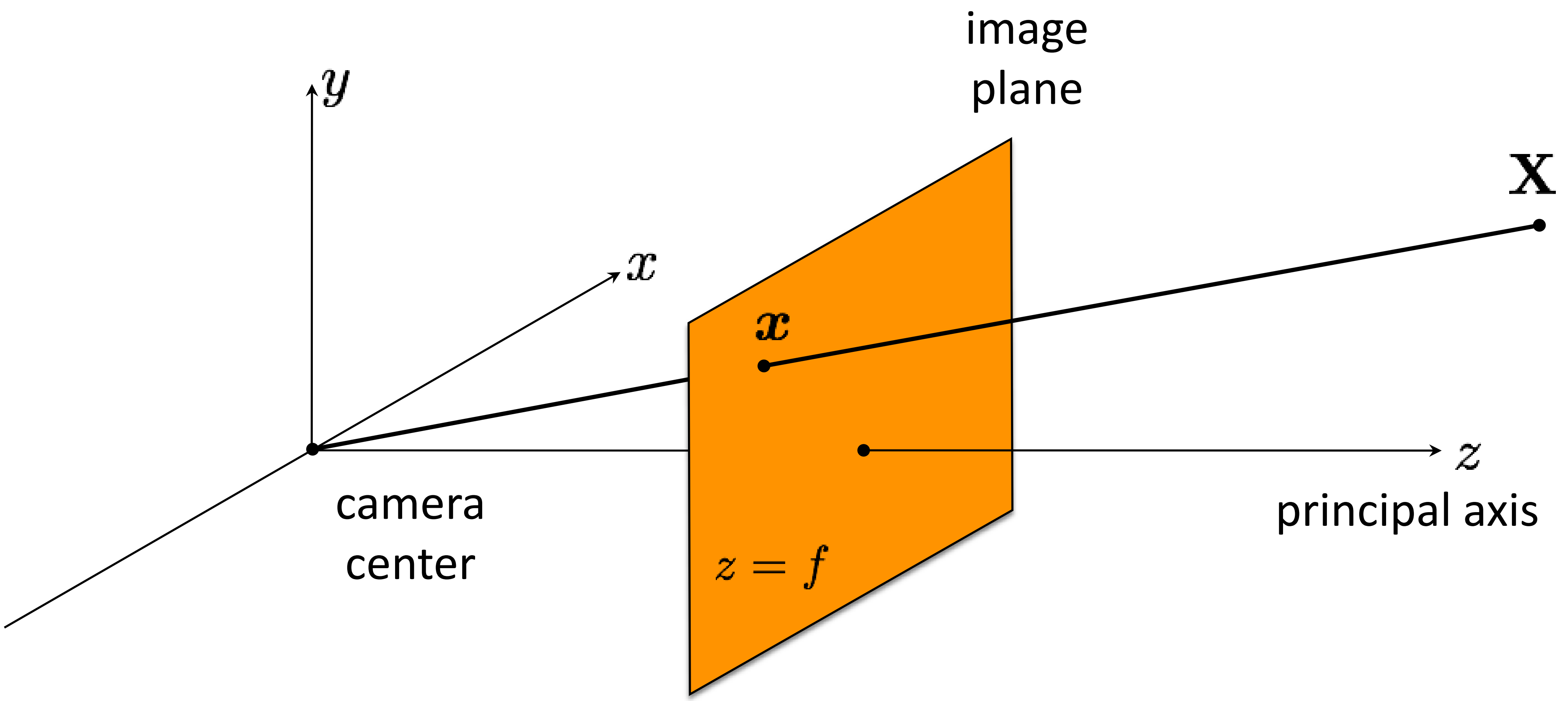
The pinhole camera



The (rearranged) pinhole camera



The (rearranged) pinhole camera



$$\boldsymbol{x} = \mathbf{PX}$$

Camera projection matrix

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}| - \mathbf{C}]$$

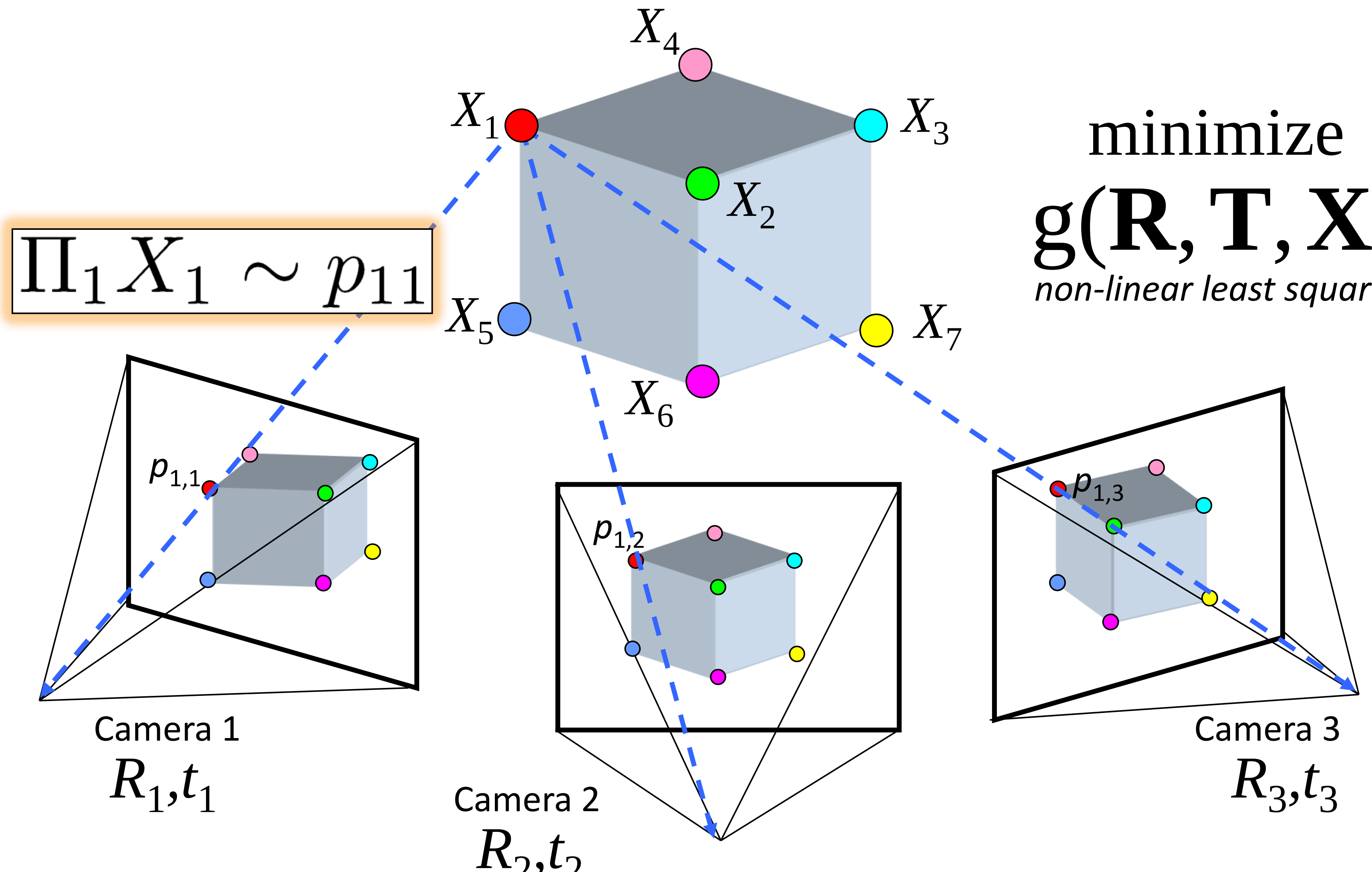
3x3 intrinsics	3x3 3D rotation	3x3 identity	3x1 3D translation

Another way to write the mapping:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

$$\mathbf{t} = -\mathbf{R}\mathbf{C}$$

Structure from motion



minimize

$$g(\mathbf{R}, \mathbf{T}, \mathbf{X})$$

non-linear least squares

Structure from motion

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \underbrace{w_{ij}}_{\substack{\textit{indicator variable}: \\ \text{is point } i \text{ visible in image } j\,?}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\textit{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\textit{observed} \\ \text{image location}}} \right\|^2$$

- Minimizing this function is called *bundle adjustment*
  - Optimized using non-linear least squares, e.g. Levenberg-Marquardt

Problem size

- What are the variables?
- How many variables per camera?
- How many variables per point?

- Trevi Fountain collection

466 input photos

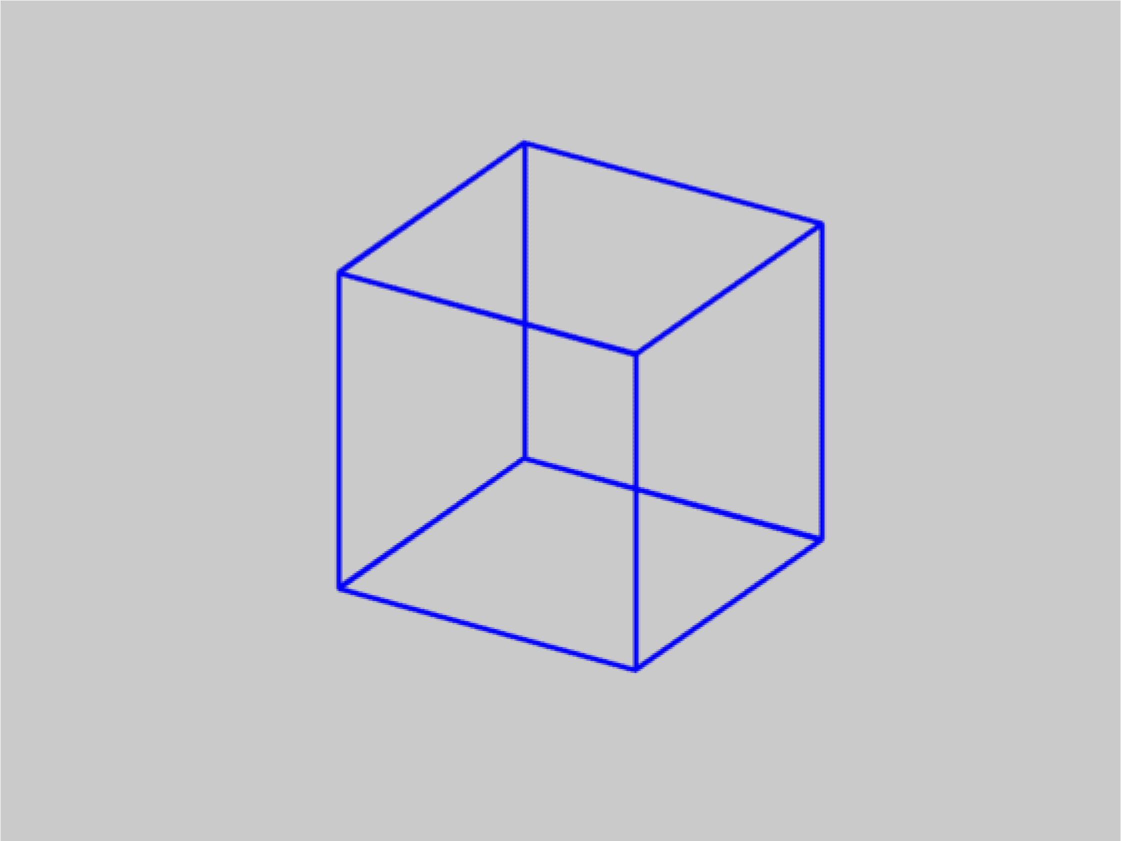
+ > 100,000 3D points

= very large optimization problem

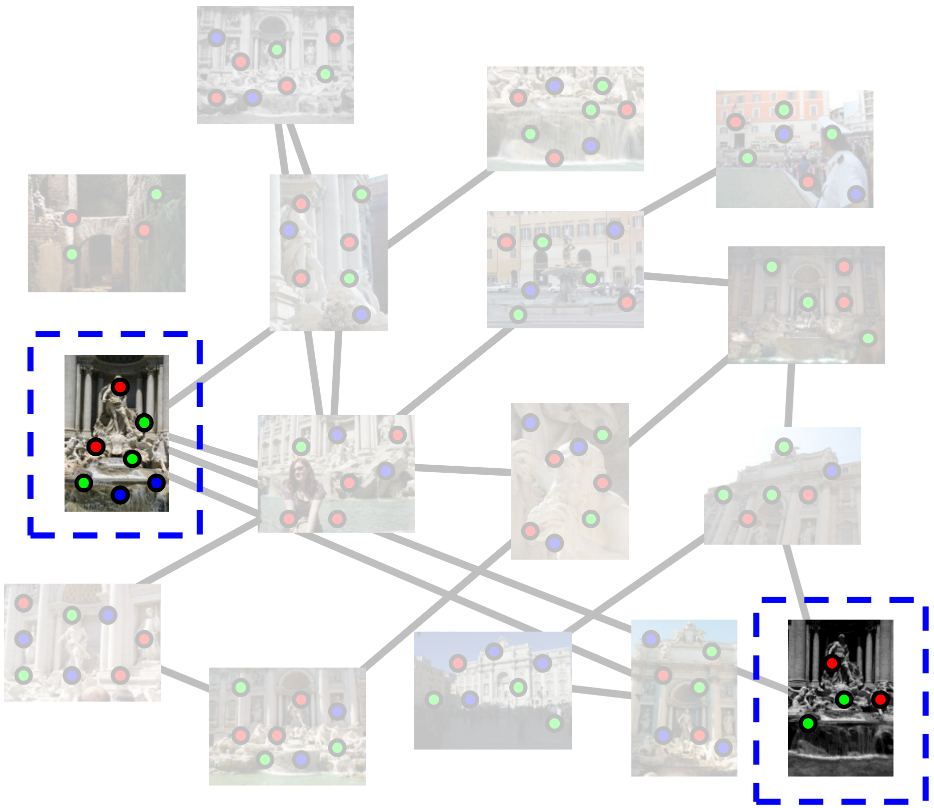
Is SfM always uniquely solvable?

Is SfM always uniquely solvable?

- No…



Incremental structure from motion



Incremental structure from motion

Final reconstruction

Even larger scale SfM

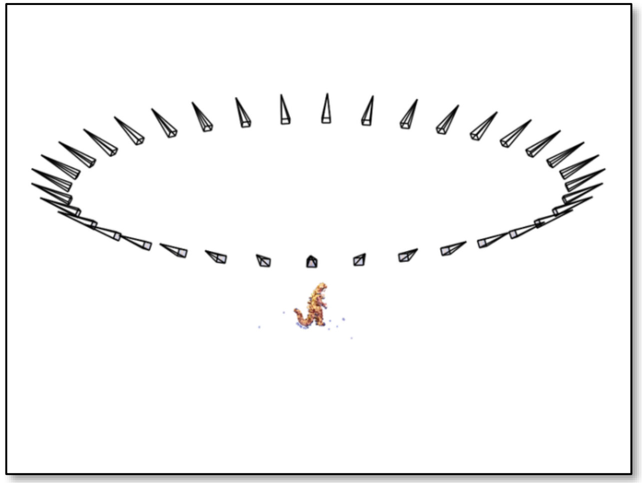
City-scale structure from motion

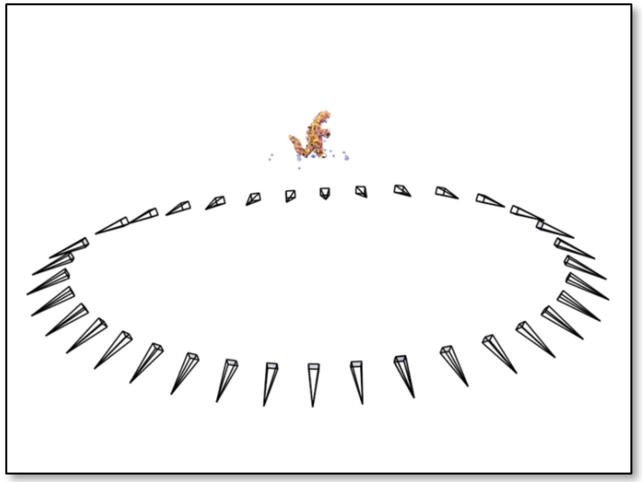
- “Building Rome in a day”

http://grail.cs.washington.edu/projects/rome/

SfM – Failure cases

- Necker reversal

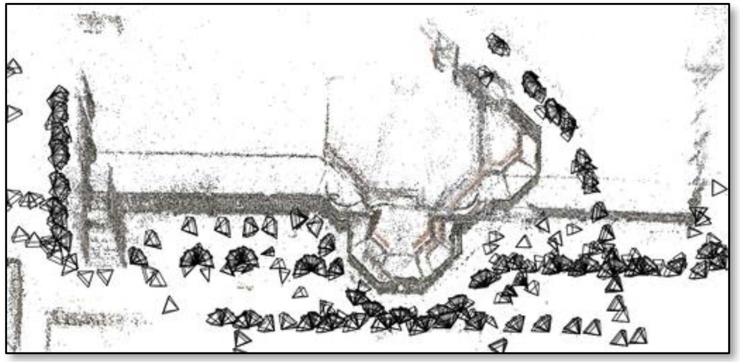




Structure from Motion – Failure cases

- Repetitive structures

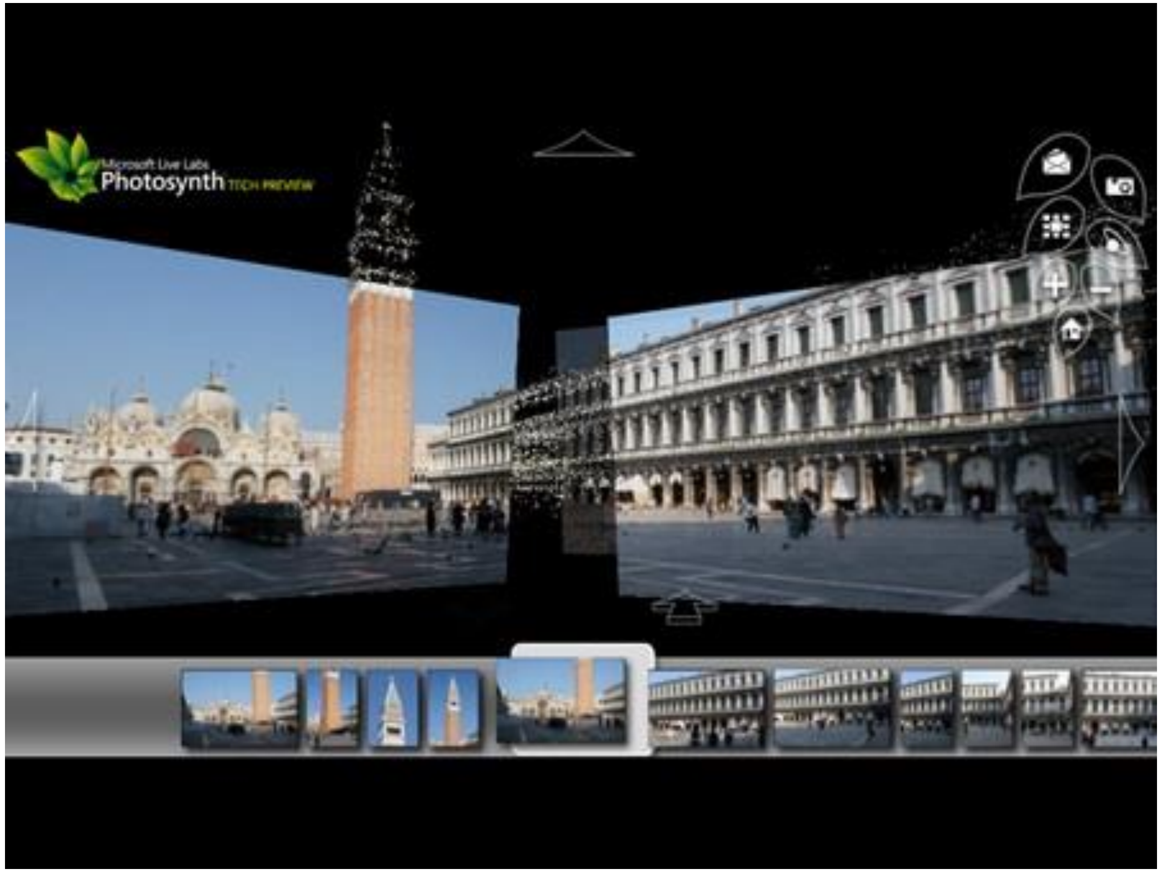





SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking
- Visual effects ("Match moving")
  - https://www.youtube.com/watch?v=RdYWp70P_kY

Applications – Photosynth



Applications – Hyperlapse

https://www.youtube.com/watch?v=SOpwHaQnRSY

References

Basic reading:

- Szeliski textbook, Sections 4, 6.1.4, 7.

Additional reading:

- Hartley and Zisserman, “Multiple View Geometry,” Cambridge University Press 2003.
  as usual when it comes to geometry and vision, this book is the best reference; Sections 10, 11, and 14 in particular discuss everything about structure from motion.
- Snavely et al., “Photo tourism: Exploring photo collections in 3D,” SIGGRAPH 2006.
- Snavely et al., “Finding Paths through the World's Photos,” SIGGRAPH 2008.
- Snavely et al., “Modeling the world from Internet photo collections,” IJCV 2008.
  the series of papers on developing Photo Tourism and large scale structure from motion.
- Agarwal et al., “Building Rome in a Day,” ICCV 2009.
  a follow-up on even larger-scale structure from motion (using “city-scale” photo collections).