Structure-from-Motion (SfM)

A lot of slides borrowed from Noah Snavely + Shree Nayar's YT series: First principals of Computer Vision

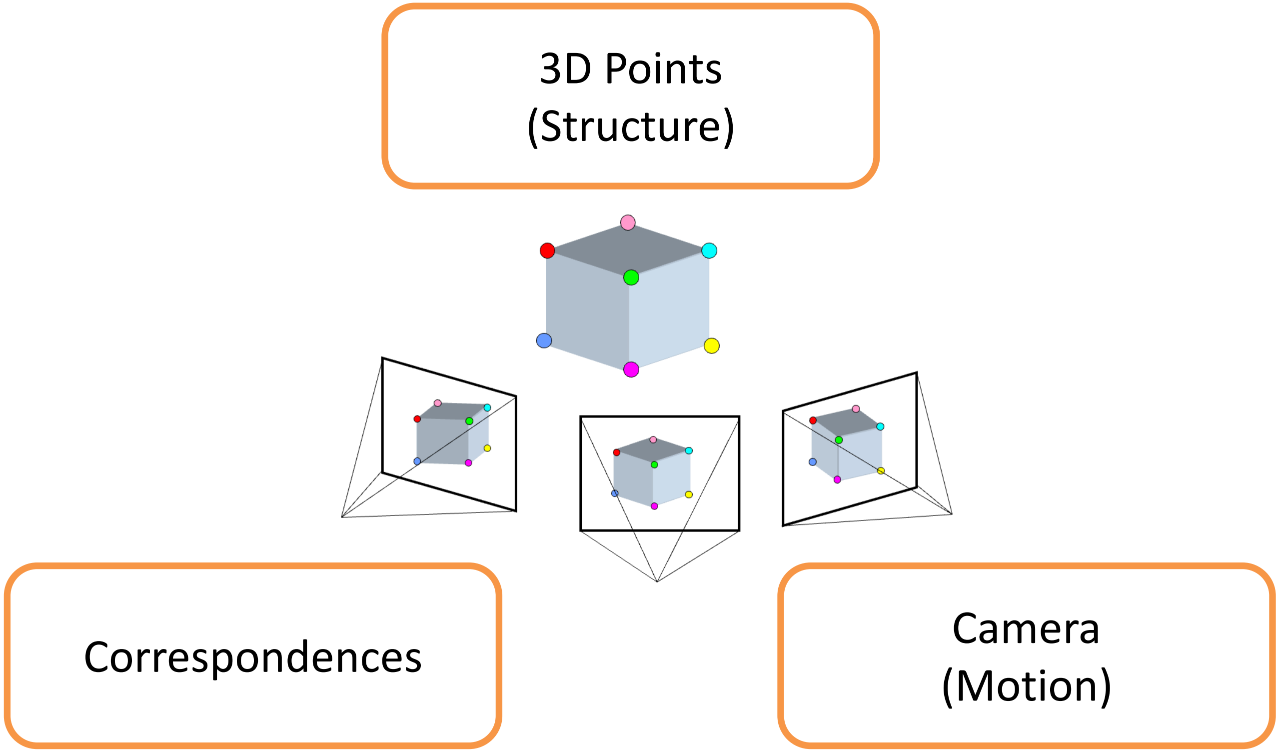
CS194: Intro to Computer Vision and Comp. Photo
Alexei Efros, UC Berkeley, Fall 2024

Recall: Camera calibration & triangulation

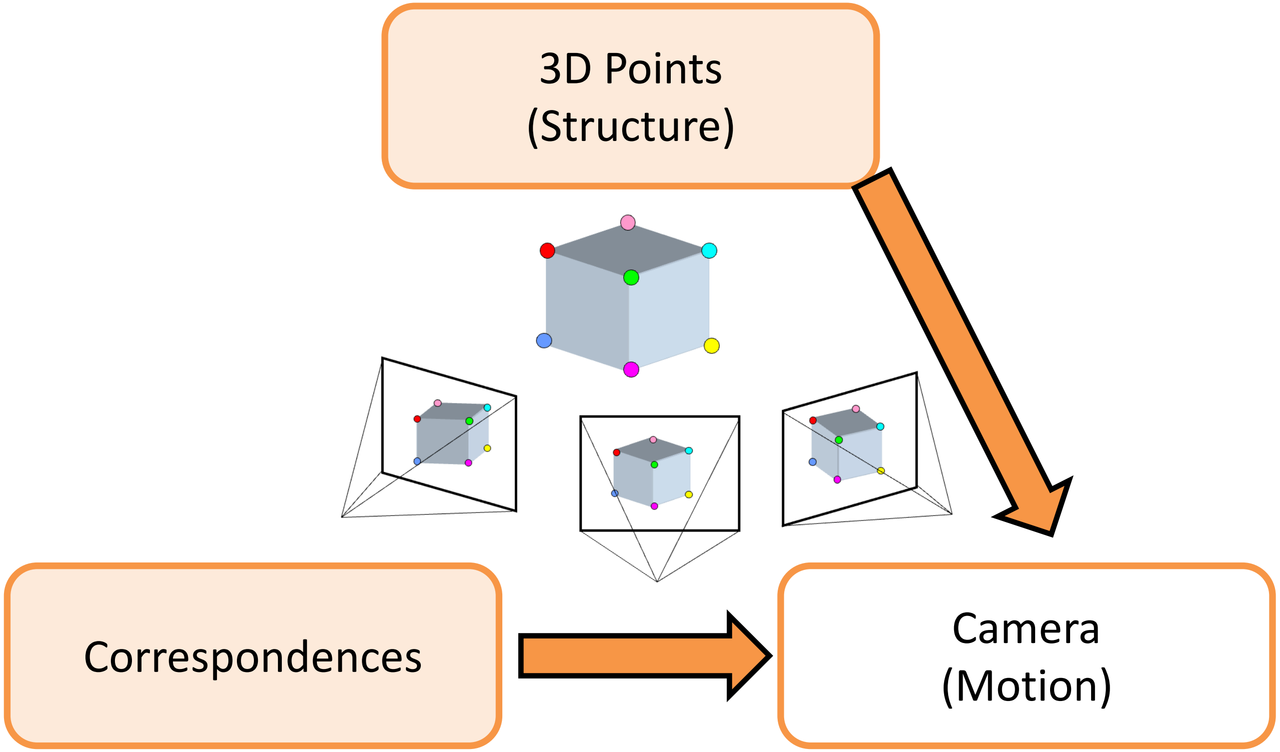
- Suppose we know **3D points** and their **matches** in an image
  - How can we compute the **camera parameters**?

- Suppose we know **camera parameters** for multiple cameras, each observing a point
  - How can we compute the **3D location** of that point?

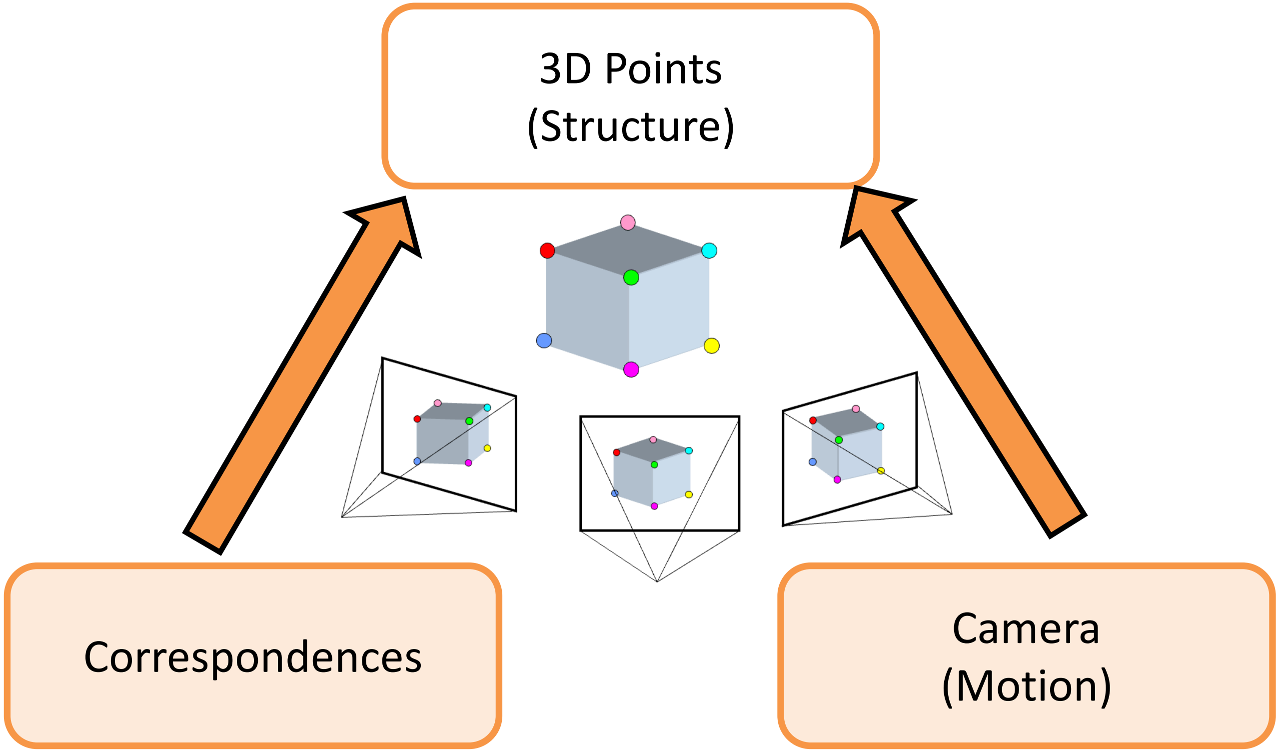
if you know 2 you get the other:



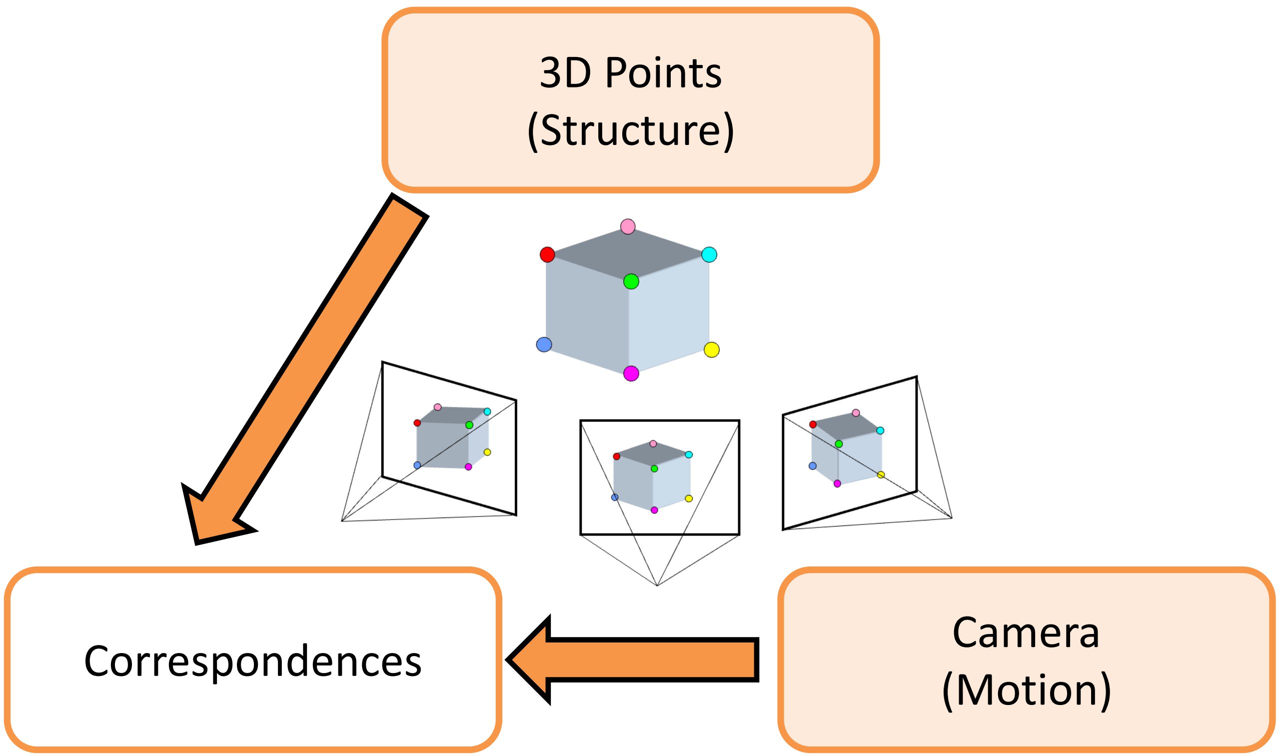
Camera Calibration; aka Perspective-n-Point



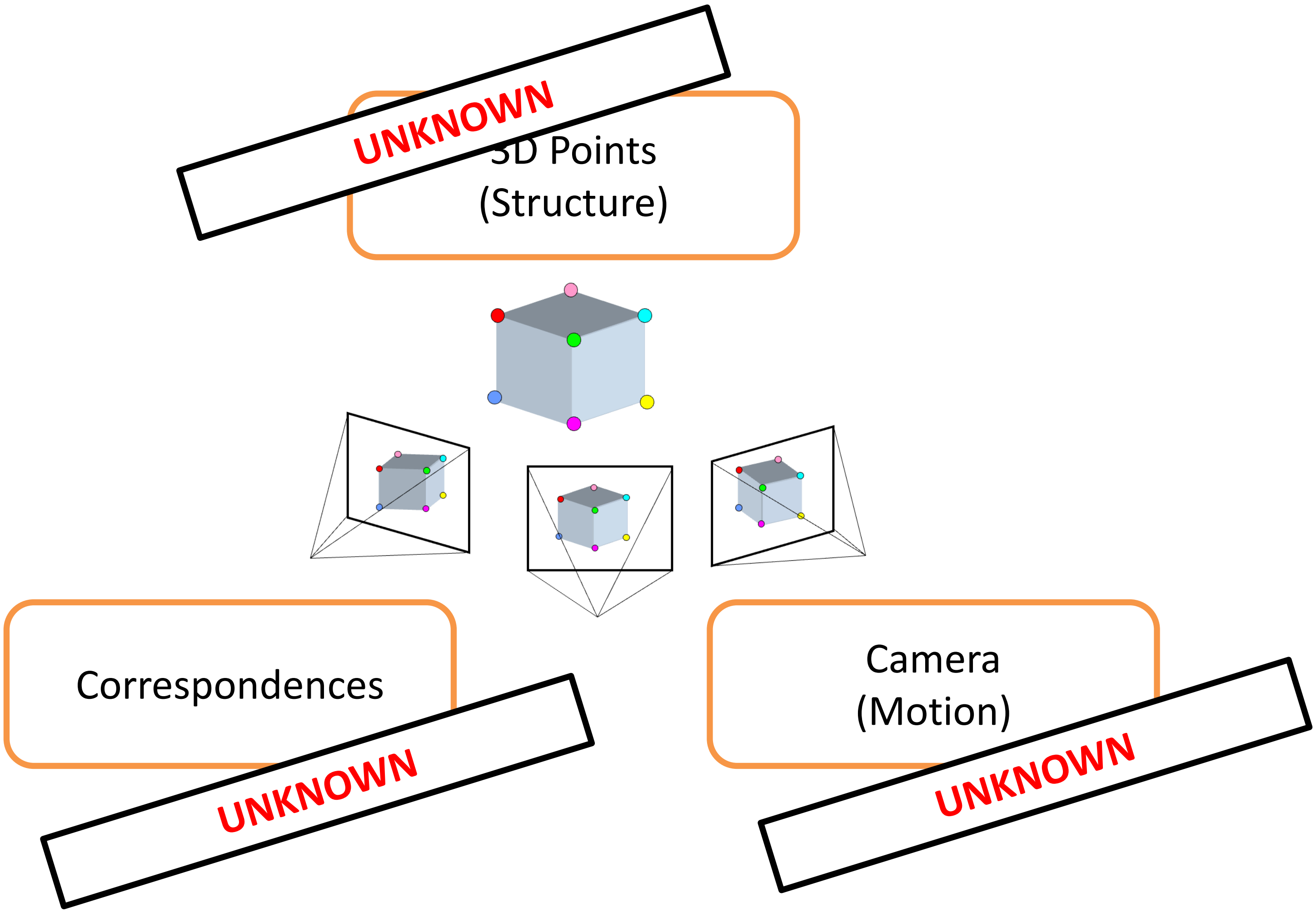
Stereo (w/2 cameras); aka Triangulation



?


3D Points
(Structure)
Correspondences
Camera
(Motion)

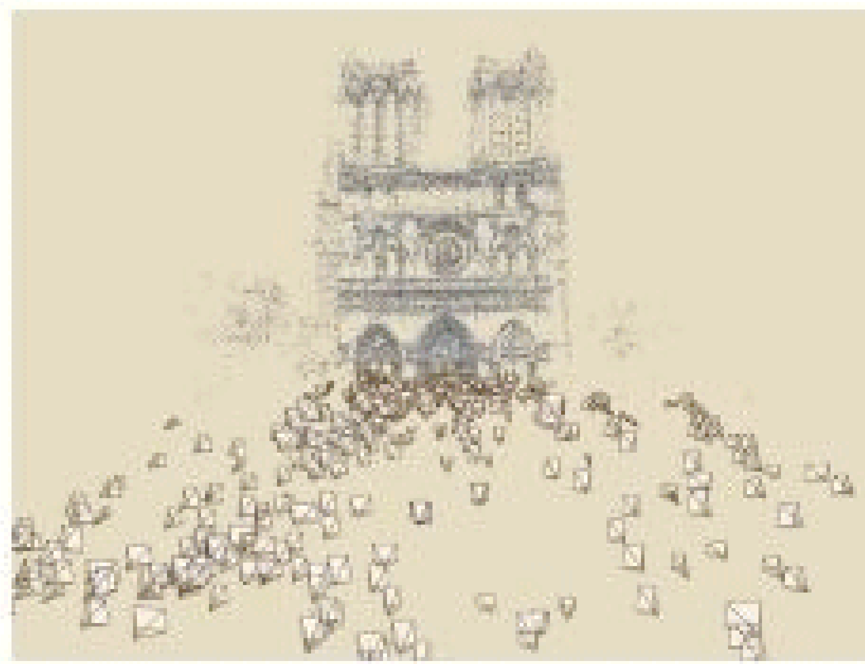
Ultimate: Structure-from-Motion



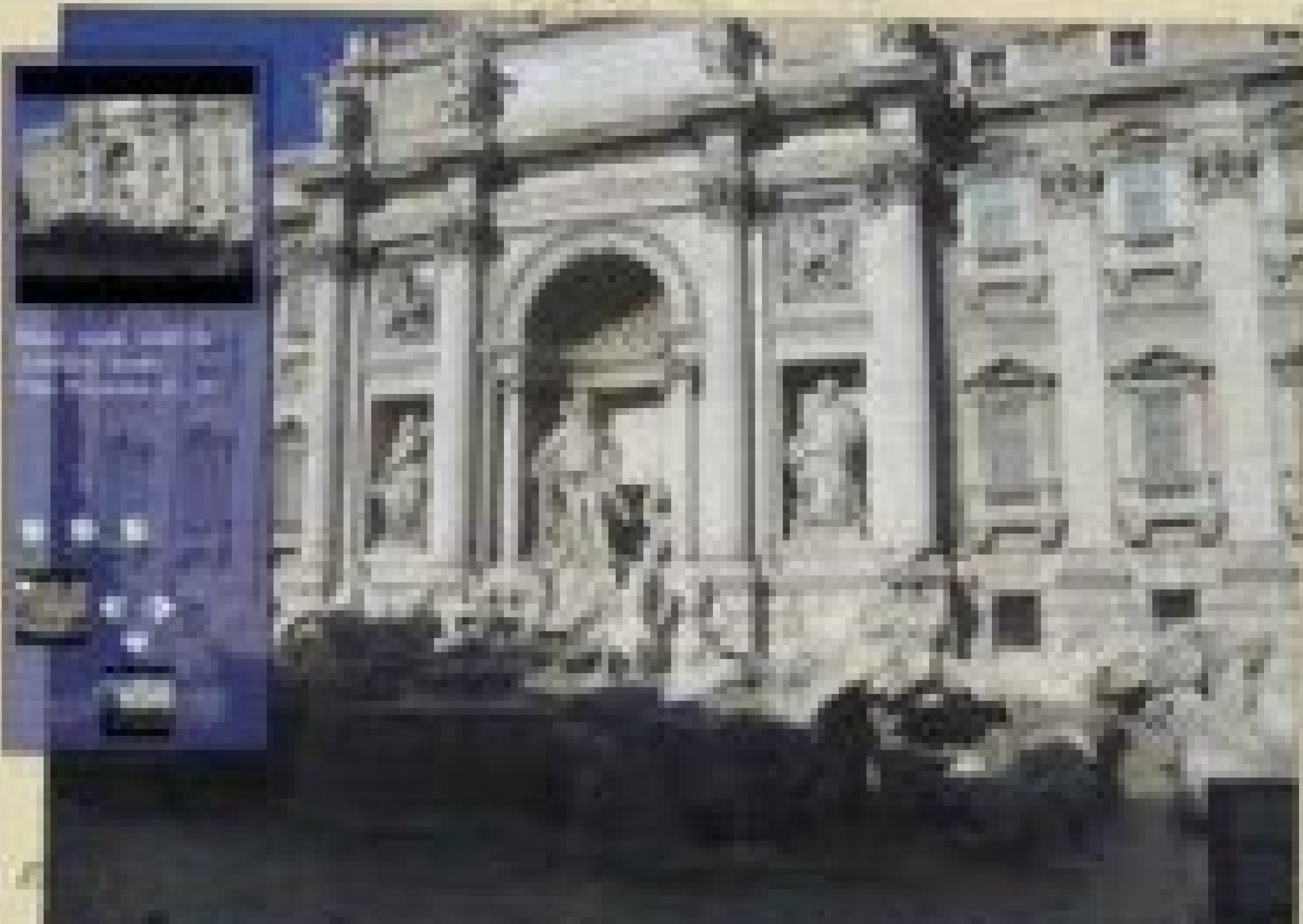
Start from nothing known (except maybe intrinsics), exploit the relationship to slowly get the right answer

Photo Tourism

Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," SIGGRAPH 2006

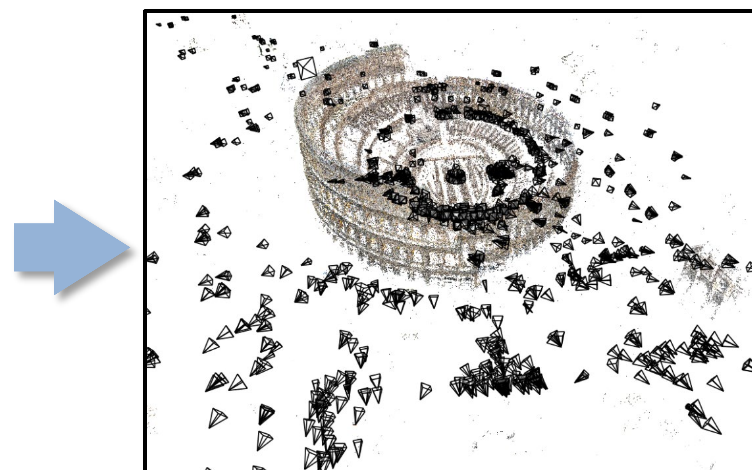


https://youtu.be/mTBPGuPLI5Y



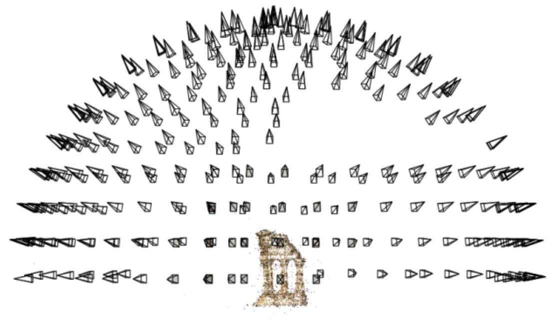
Structure from Motion (SfM)

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

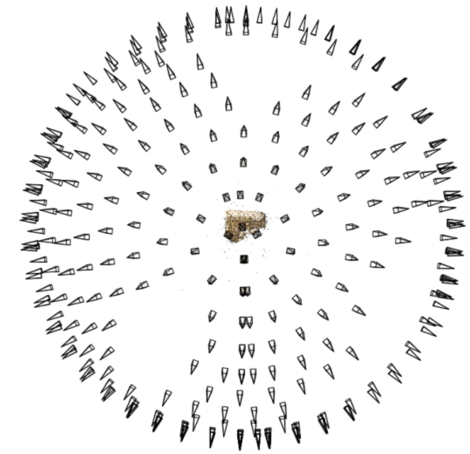


This is (roughly) the **structure from motion** problem

Structure from motion



Reconstruction (side)



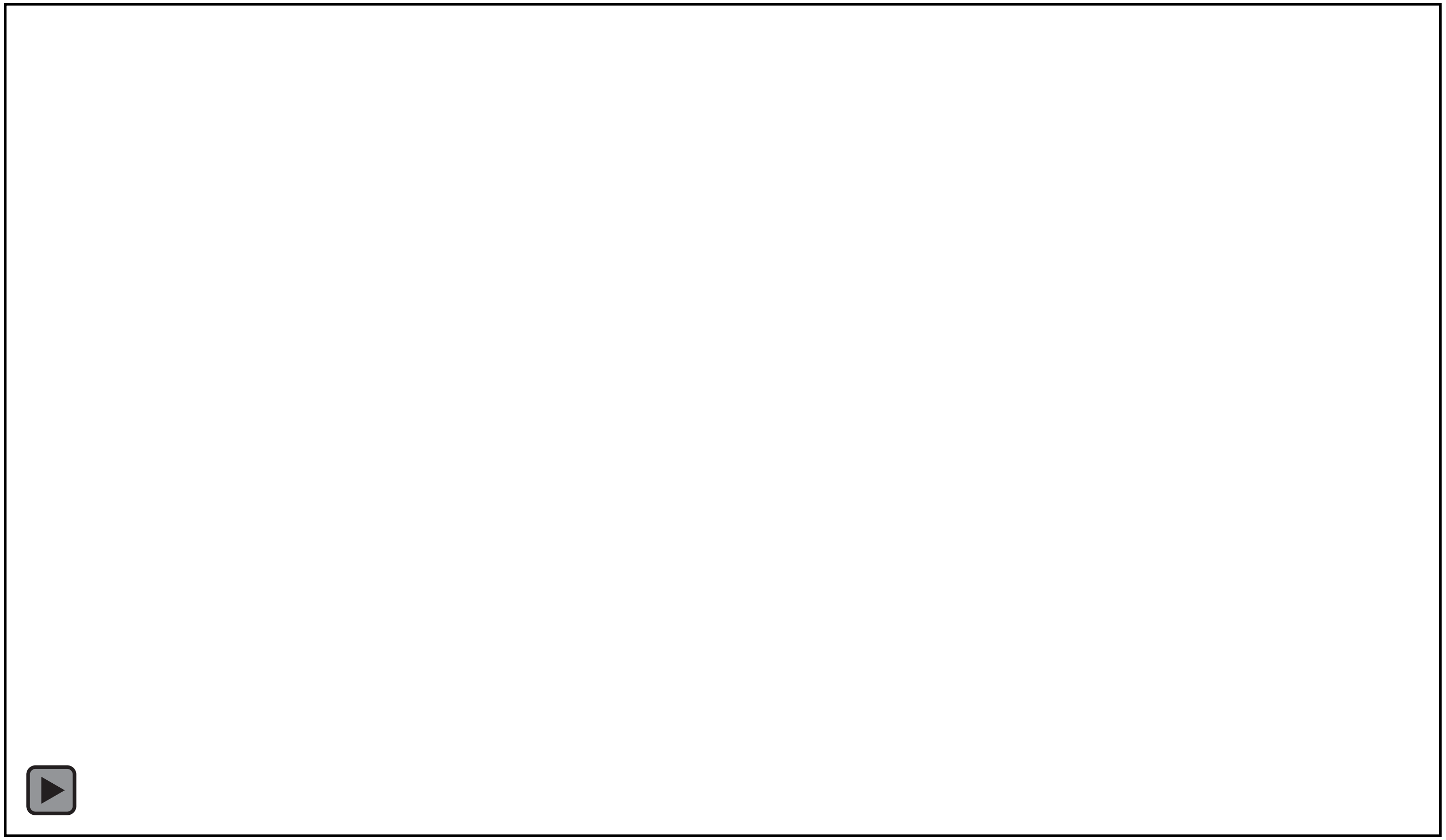
(top)

- 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*

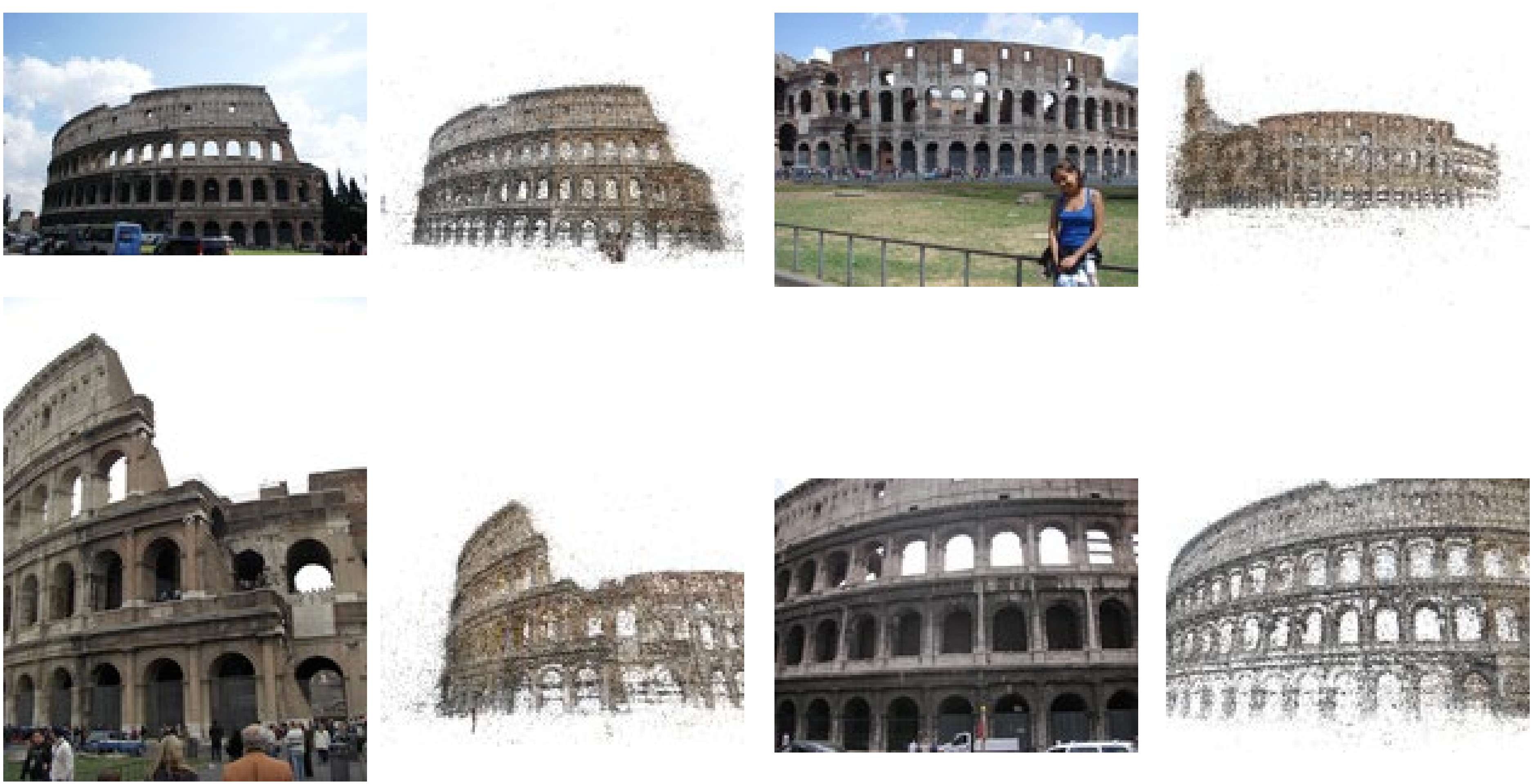
Large-scale structure from motion



Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

Building Rome in a Day, Agarwal et al. ICCV 2009

Large-scale structure from motion



Rome’s Colosseum

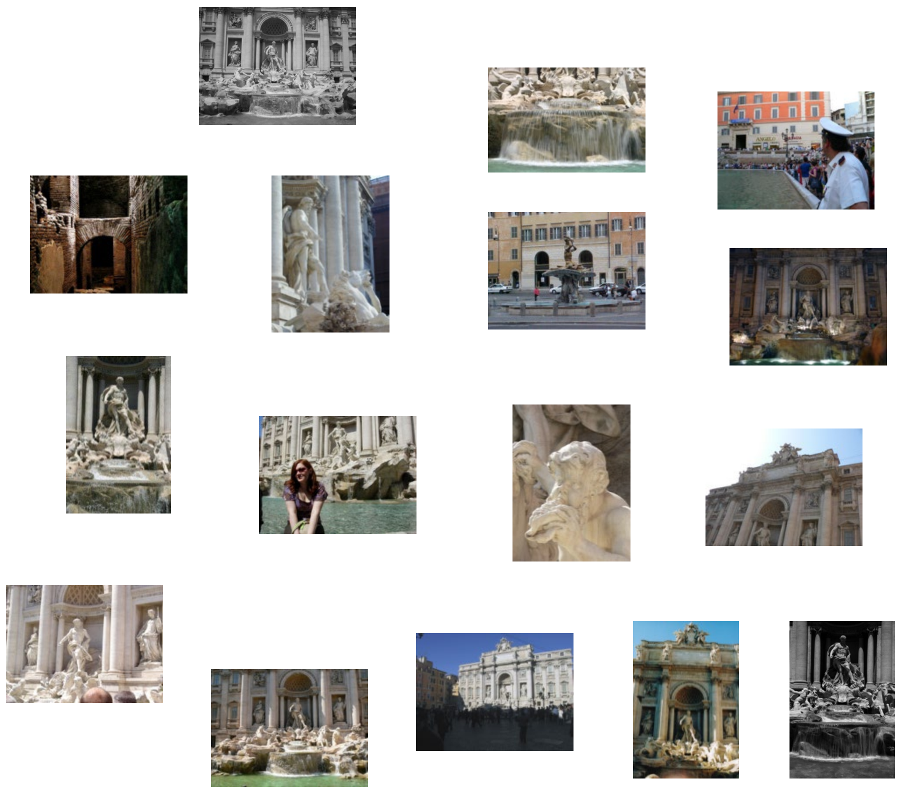
Building Rome in a Day, Agarwal et al. ICCV 2009

First step: Correspondence

- Feature detection and matching

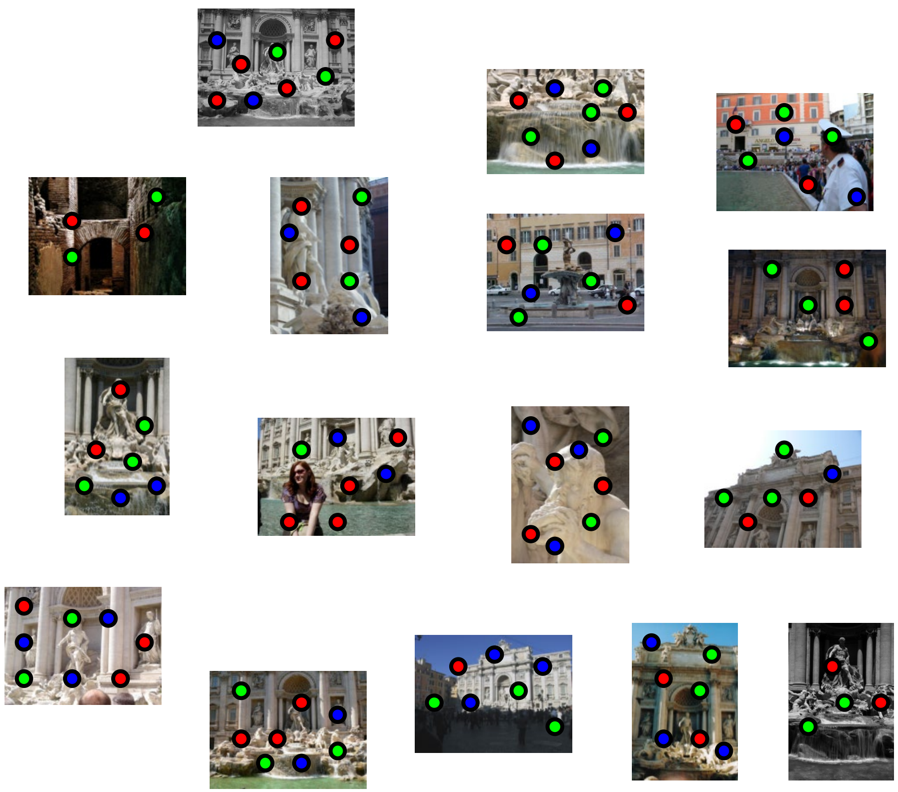
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



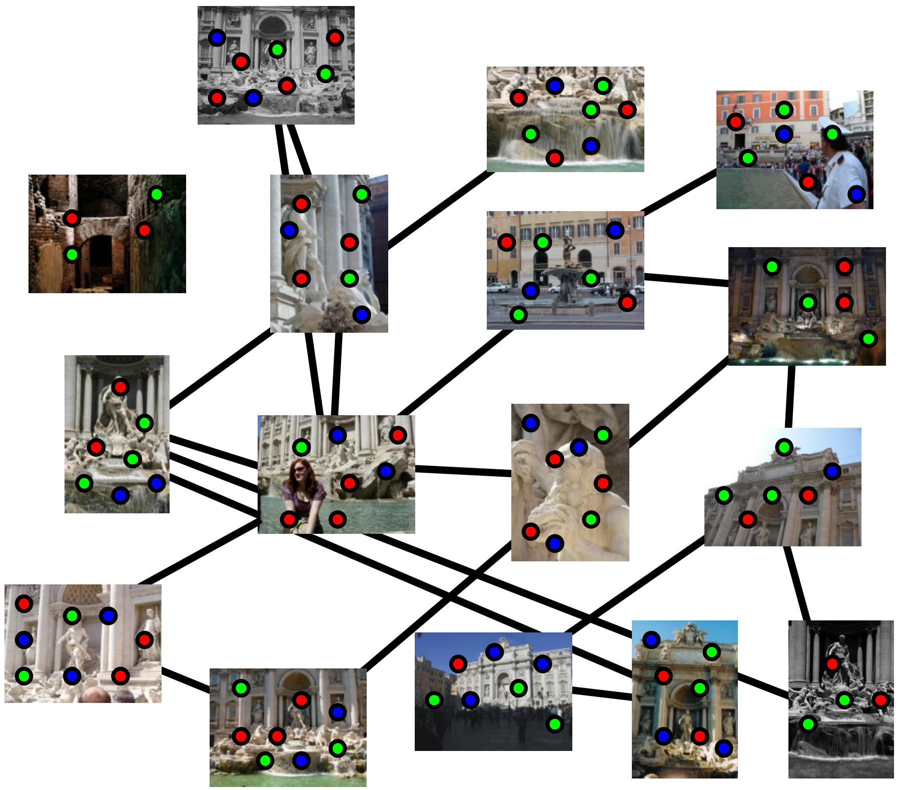
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



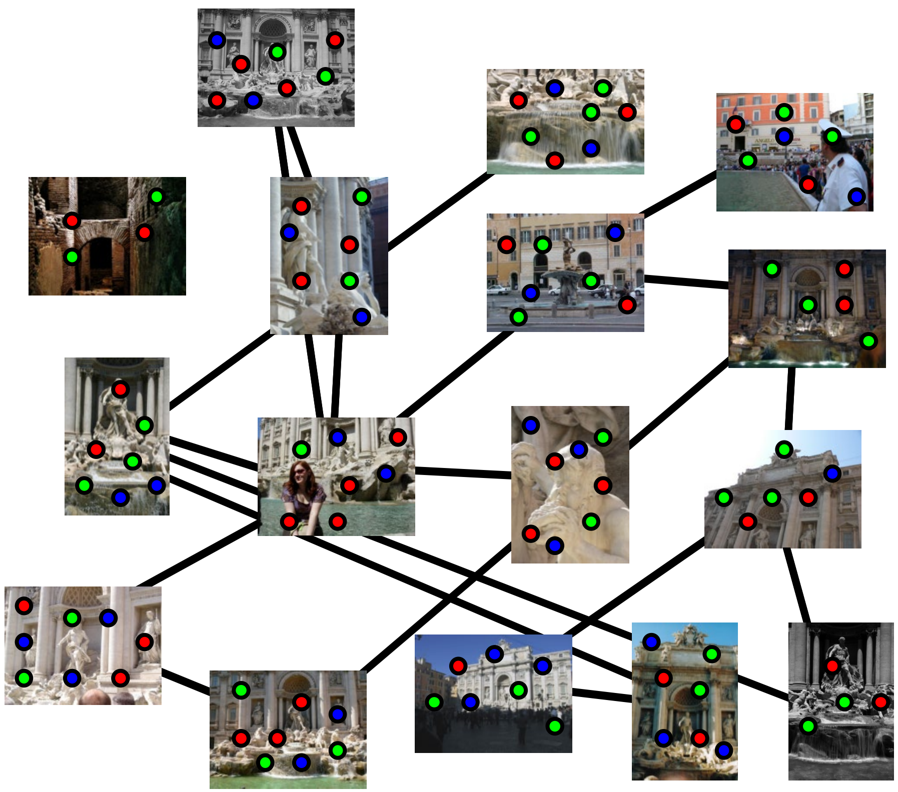
Feature matching

Match features between each pair of images



Feature matching

Refine matching using RANSAC to estimate fundamental matrix between each pair



Correspondence estimation

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

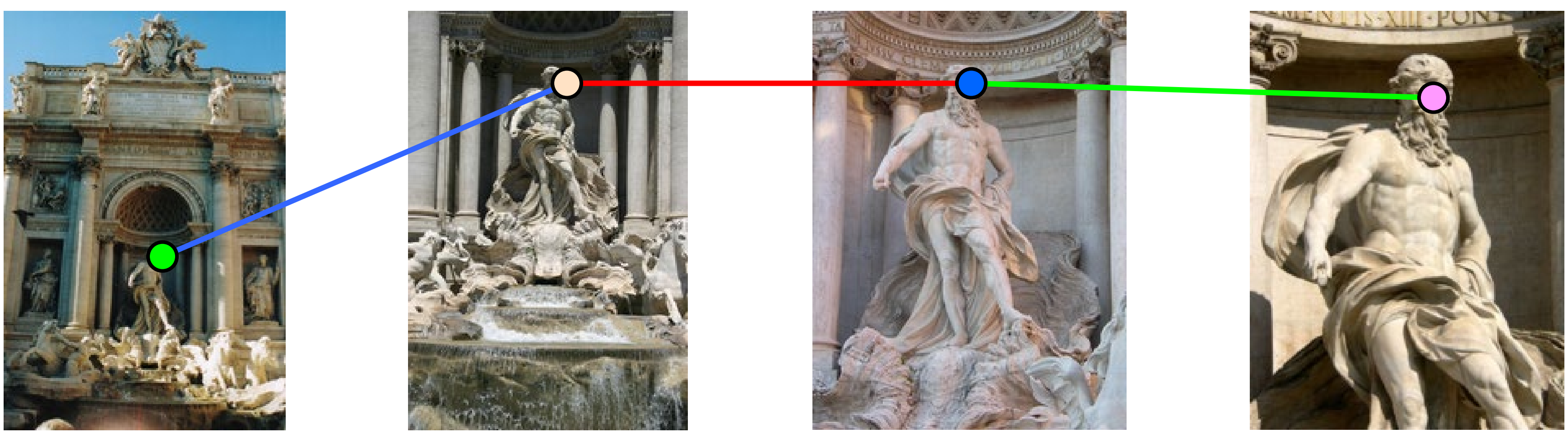
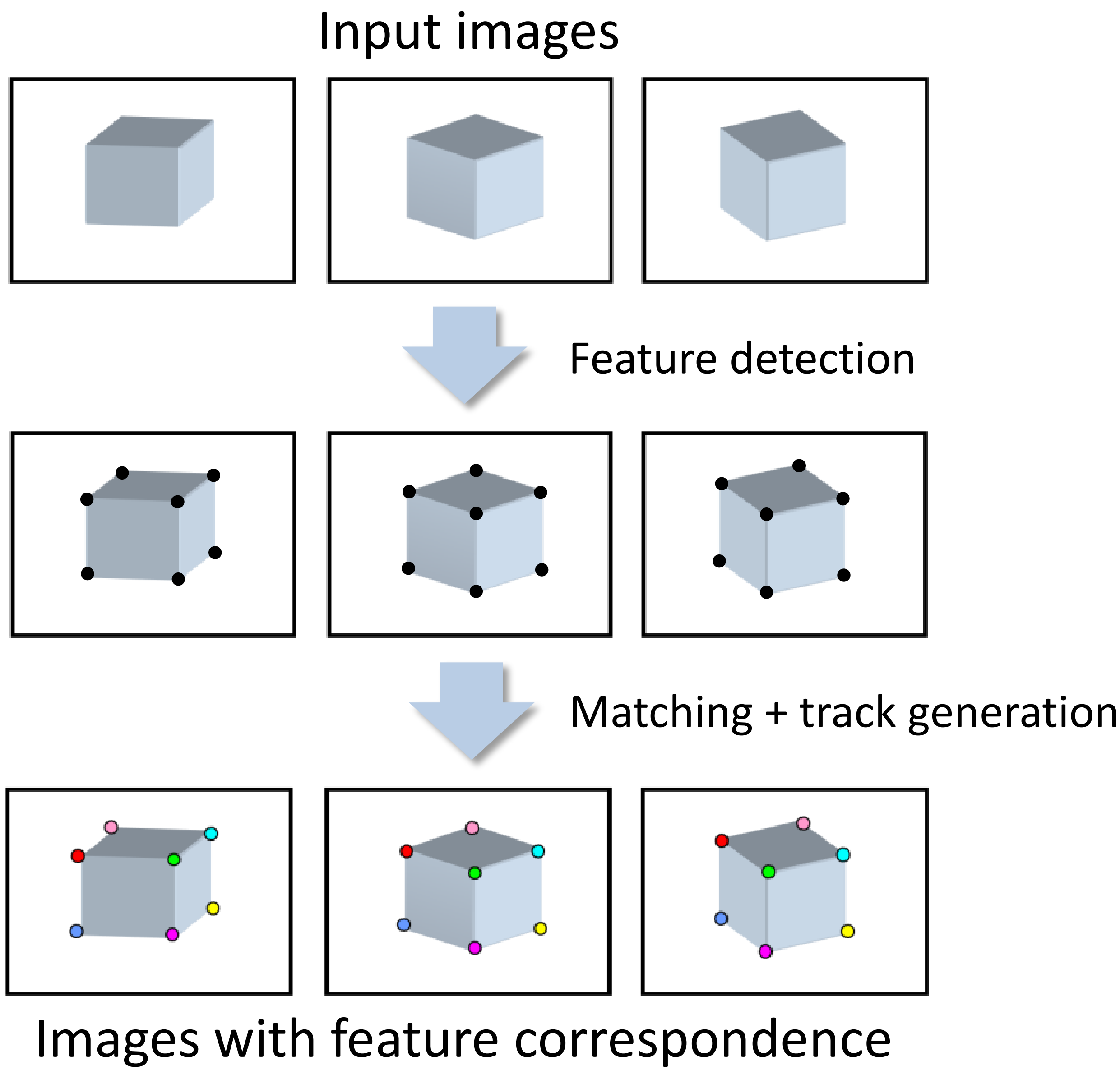
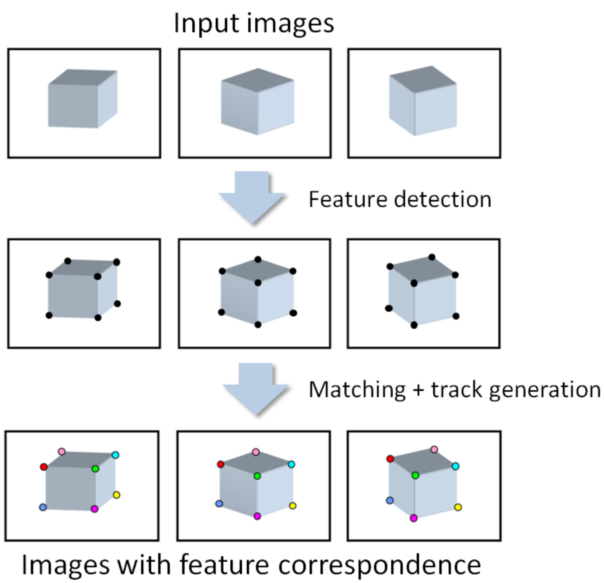


Image 1 Image 2 Image 3 Image 4

The story so far…



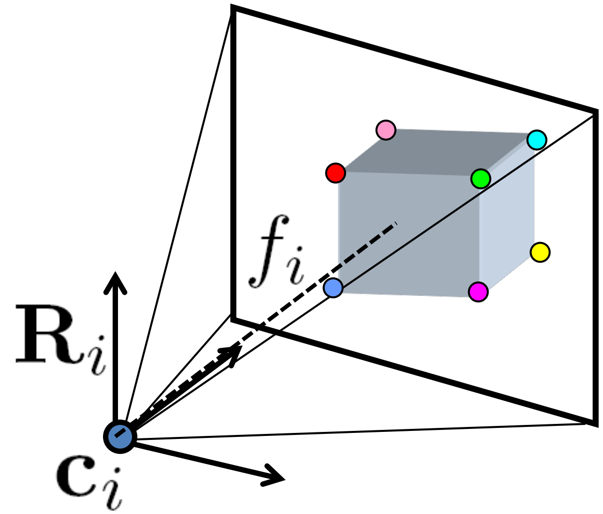
The story so far…



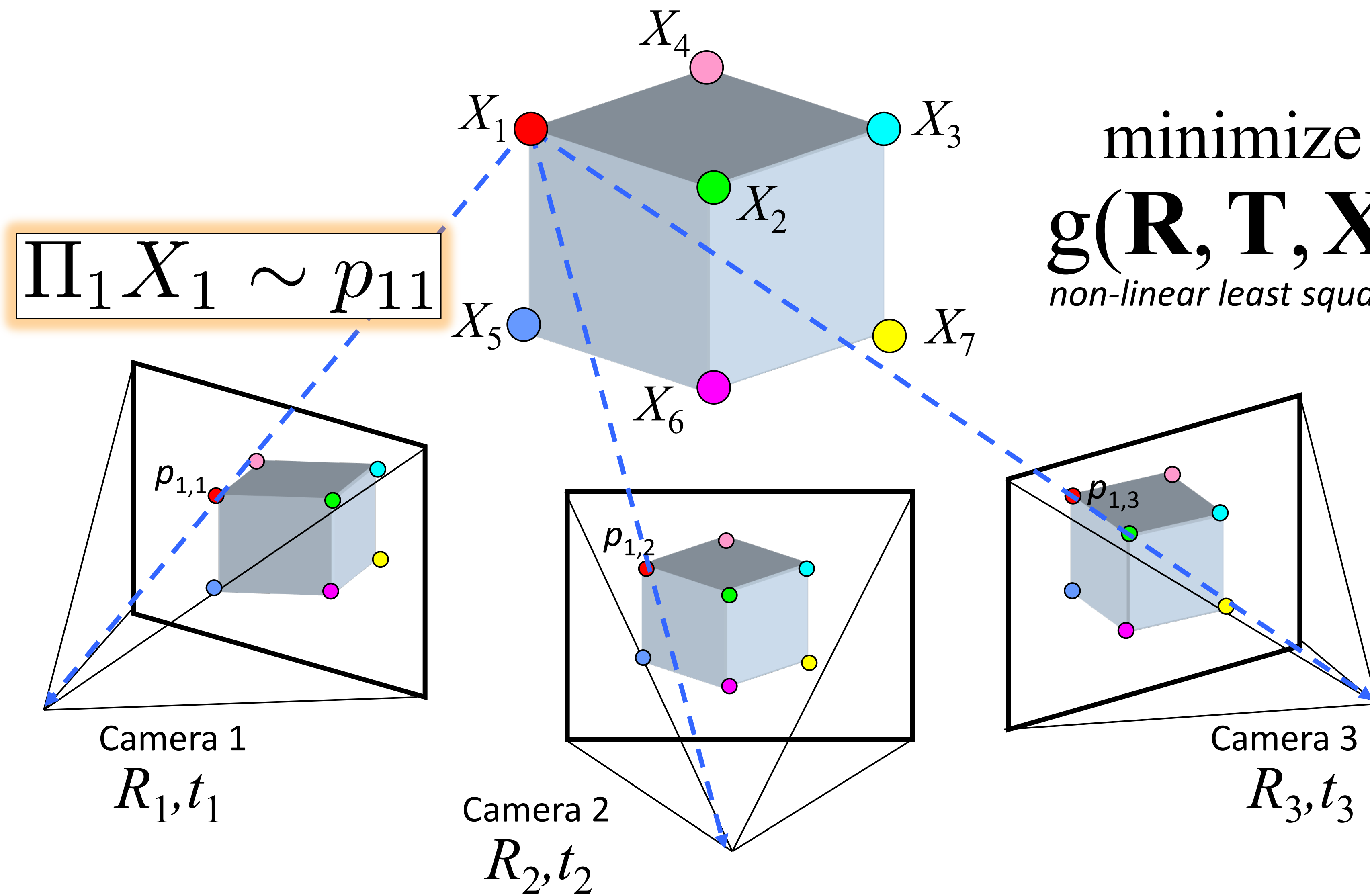
- Next step:
  - Use structure from motion to solve for geometry (cameras and points)
- First: what are cameras and points?

Review: Points and cameras

- Point: 3D position in space ($\mathbf{X}_j$)
- Camera (C_i):
  - A 3D position ($\mathbf{c}_i$)
  - A 3D orientation ($\mathbf{R}_i$)
  - Intrinsic parameters (**focal length**, aspect ratio, ...)
  - 7 parameters (3+3+1) in total



Structure from motion



minimize

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

non-linear least squares

Structure from motion

- Minimize sum of squared reprojection errors:

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

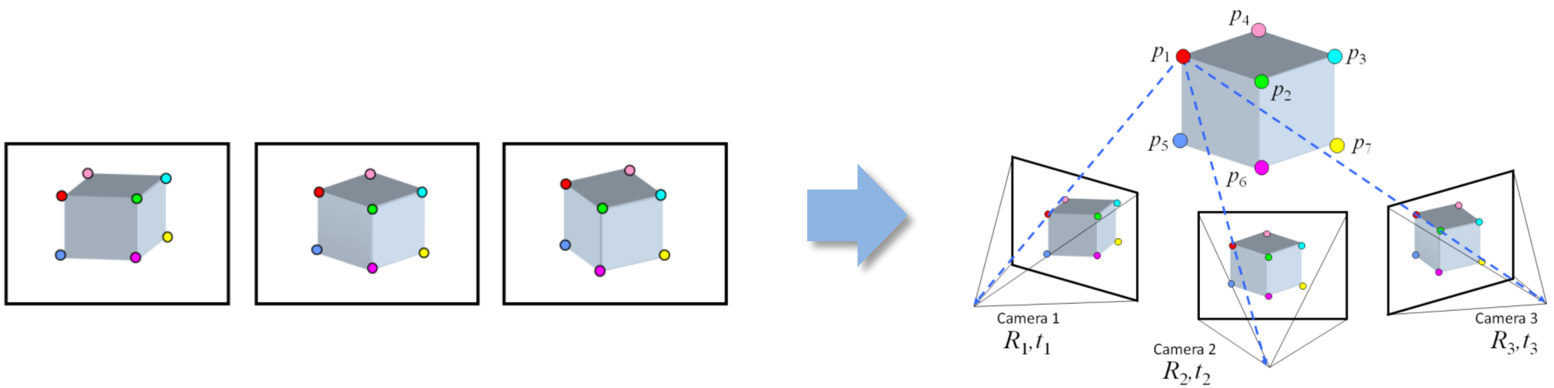
predicted
image location

observed
image location

indicator variable:
is point *i* visible in image *j* ?

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

Solving structure from motion

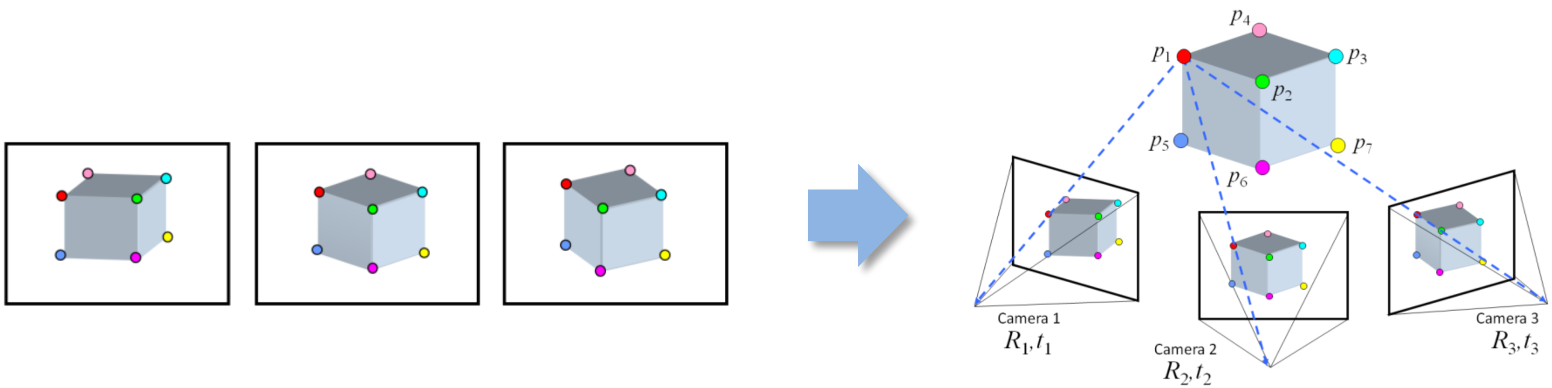


Inputs: feature tracks

Outputs: 3D cameras and points

- Challenges:
  - Large number of parameters (1000's of cameras, millions of points)
  - Very non-linear objective function

Solving structure from motion

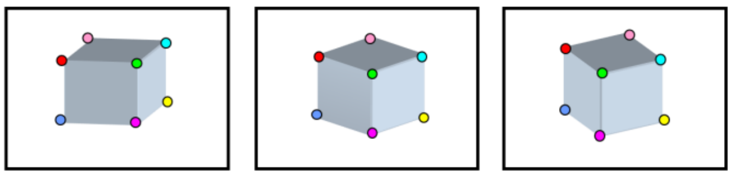


Inputs: feature tracks

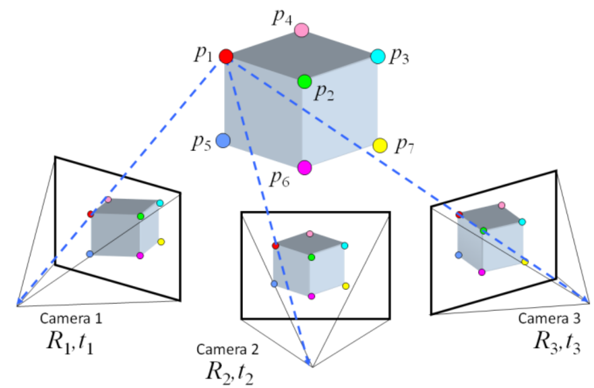
Outputs: 3D cameras and points

- Important tool: Bundle Adjustment [Triggs *et al.* '00]
  - Joint non-linear optimization of both cameras and points
  - Very powerful, elegant tool
- The bad news:
  - Starting from a random initialization is very likely to give the wrong answer
  - Difficult to initialize all the cameras at once

Solving structure from motion



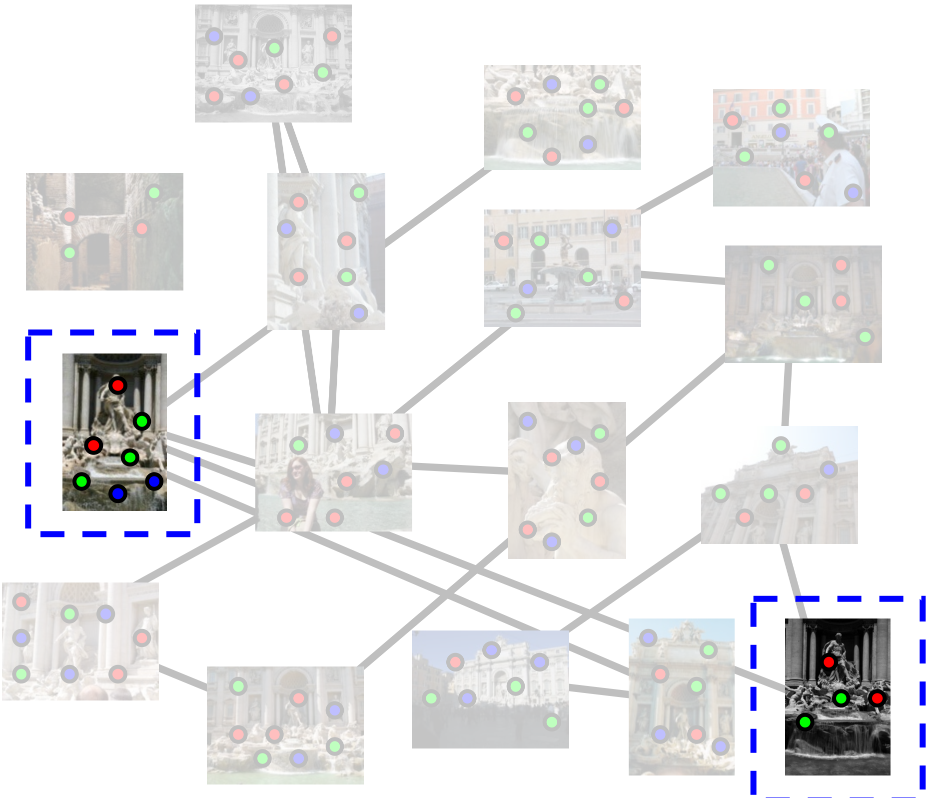
Inputs: feature tracks



Outputs: 3D cameras and points

- The good news:
  - Structure from motion with two cameras is (relatively) easy
  - Once we have an initial model, it’s easy to add new cameras
- Idea:
  - Start with a small seed reconstruction, and grow

Incremental SfM



- Automatically select an initial pair of images

1. Picking the initial pair

- We want a pair with many matches, but which has as large a baseline as possible

✔ lots of matches
✘ small baseline

✔ large baseline
✘ very few matches

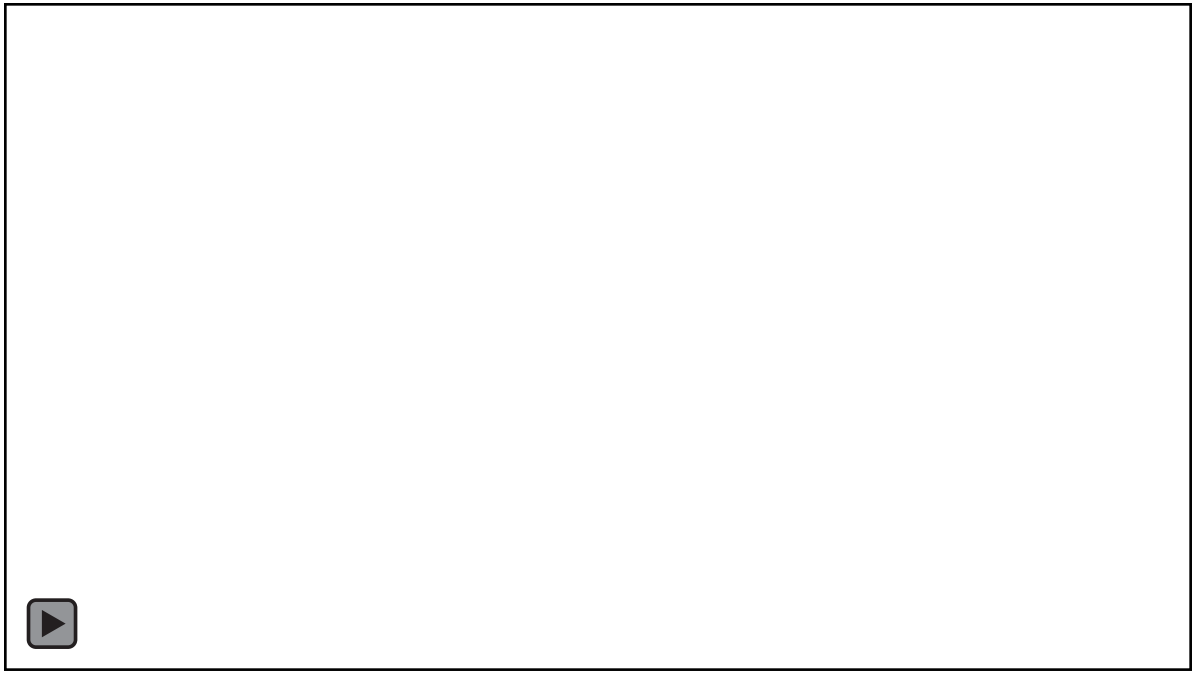
✔ large baseline
✔ lots of matches

Incremental SfM: Algorithm

1. Pick a strong initial pair of images
2. Initialize the model using two-frame SfM
3. While there are connected images remaining:
    a. Pick the image which sees the most existing 3D points
    b. Estimate the pose of that camera
    c. Triangulate any new points
    d. Run bundle adjustment

Visual Simultaneous Localization and Mapping (V-SLAM)

- Main differences with SfM:
  - Continuous visual input from sensor(s) over time
  - Gives rise to problems such as loop closure
  - Often the goal is to be online / real-time



Video from Daniel Cremer's Lab