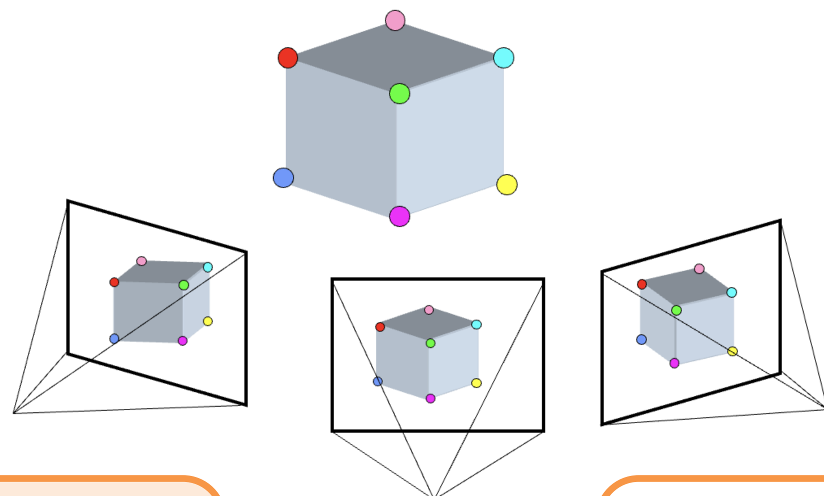
SfM / MVS / NeRF..

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

CS180: Intro to Computer Vision and Comp. Photo
Angjoo Kanazawa & Alexei Efros, UC Berkeley, Fall 2025

End of last lecture: Camera from Corresp

3D Points
(Structure)



Correspondences

Camera
(Motion)

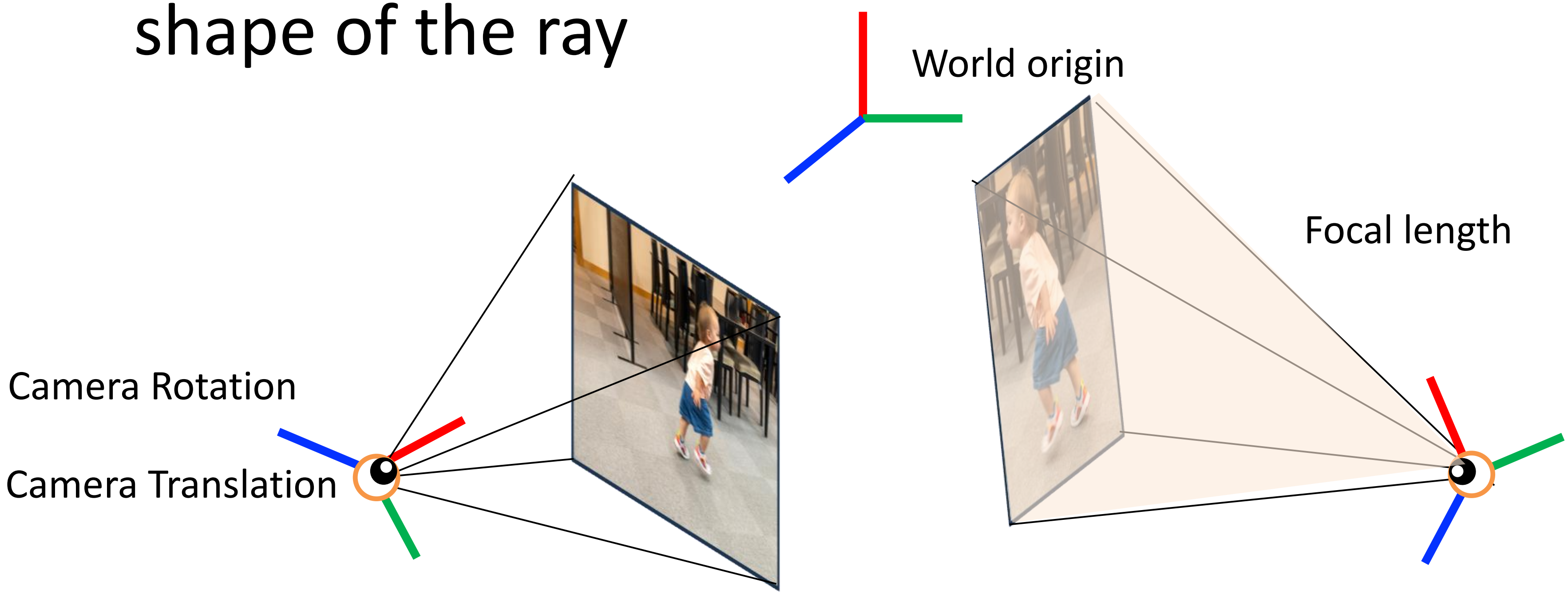
How to estimate the camera?

- 1. Estimate the fundamental/essential matrix with correspondences!

- 2. Another method: Calibration

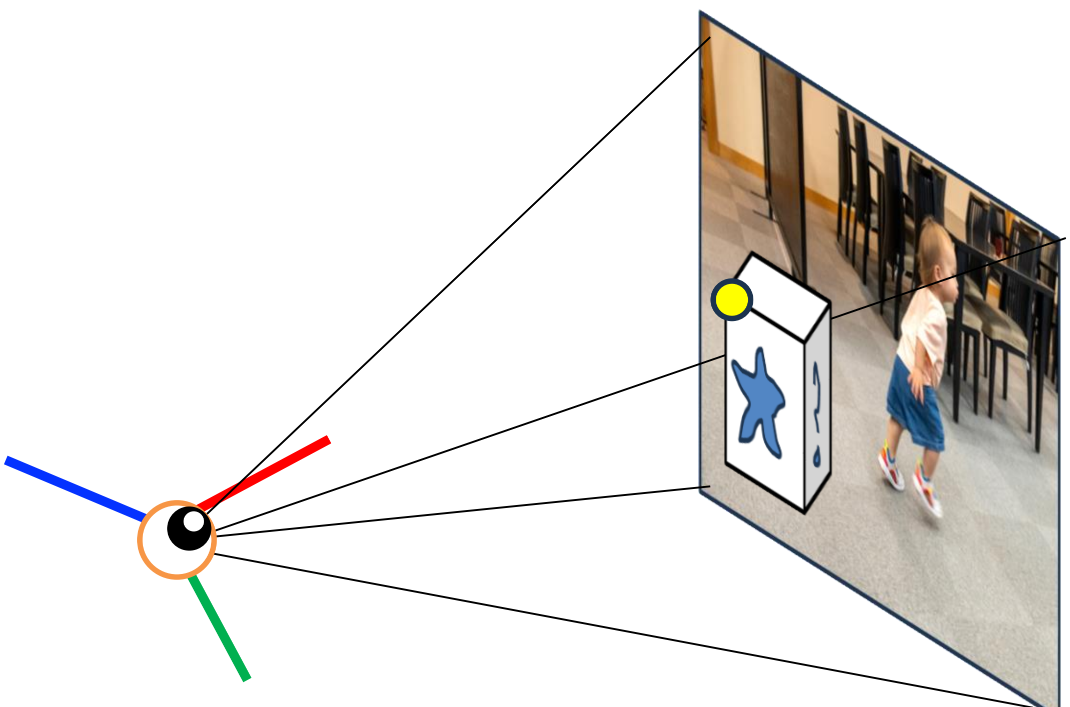
Problem: Solve for the camera

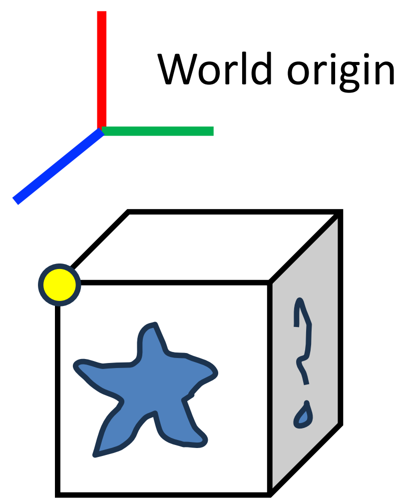
- What are the camerea parameters?
  - Extrinsics (R, T)
  - Intrinsics (K)
- How am I situated in the world + what is the shape of the ray



Calibration

- Definition: Solve for camera using a known 3D structure + where it is in the image
- Invasive / active
- Can’t be done on existing pictures





Only have to do it once if the cameras are static

How to calibrate the camera?

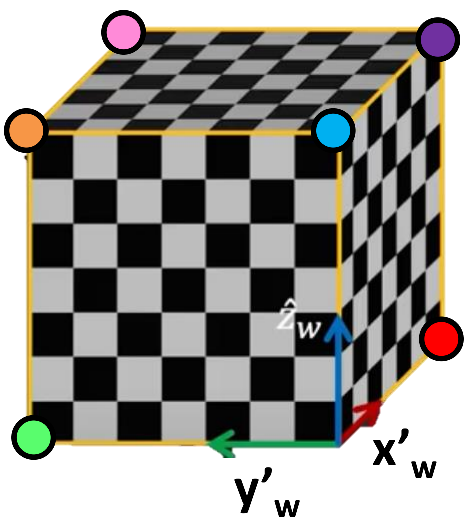
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

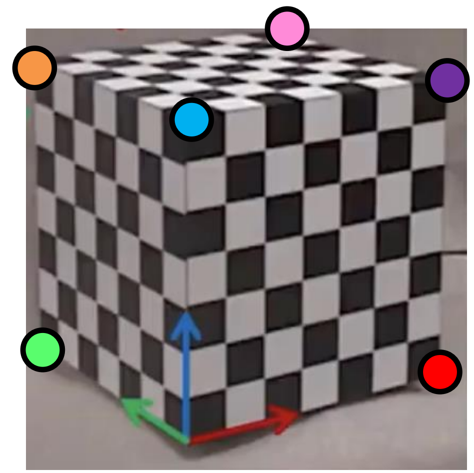
$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

If we know the points in 3D we can estimate the camera!!

Step 1: With a known 3D object

1. Take a picture of an object with known 3D geometry

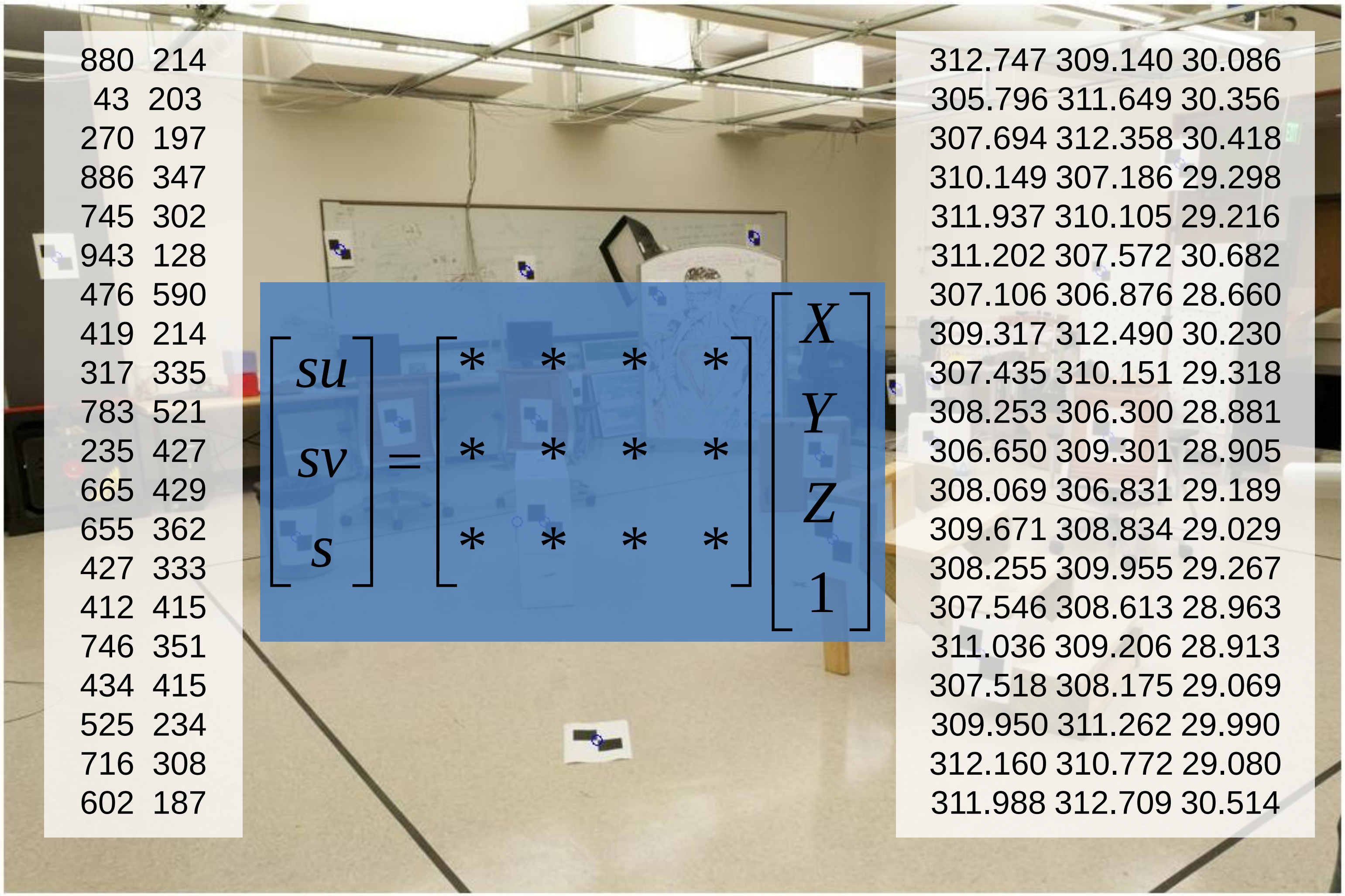




2. Identify correspondences

Slide from Shree Nayar

How do we calibrate a camera?



880	214
43	203
270	197
886	347
745	302
943	128
476	590
419	214
317	335
783	521
235	427
665	429
655	362
427	333
412	415
746	351
434	415
525	234
716	308
602	187

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

312.747	309.140	30.086
305.796	311.649	30.356
307.694	312.358	30.418
310.149	307.186	29.298
311.937	310.105	29.216
311.202	307.572	30.682
307.106	306.876	28.660
309.317	312.490	30.230
307.435	310.151	29.318
308.253	306.300	28.881
306.650	309.301	28.905
308.069	306.831	29.189
309.671	308.834	29.029
308.255	309.955	29.267
307.546	308.613	28.963
311.036	309.206	28.913
307.518	308.175	29.069
309.950	311.262	29.990
312.160	310.772	29.080
311.988	312.709	30.514

Method: Set up a linear system

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Solve for m's entries using linear least squares

Ax=0 form

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1X_1 & -u_1Y_1 & -u_1Z_1 & -u_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1X_1 & -v_1Y_1 & -v_1Z_1 & -v_1 \\ & & & & & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_nX_n & -u_nY_n & -u_nZ_n & -u_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_nX_n & -v_nY_n & -v_nZ_n & -v_n \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{14} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{24} \\ m_{31} \\ m_{32} \\ m_{33} \\ m_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \\ 0 \\ 0 \end{bmatrix}$$

Similar to how you solved for homography!

Can we factorize M back to K [R | T]?

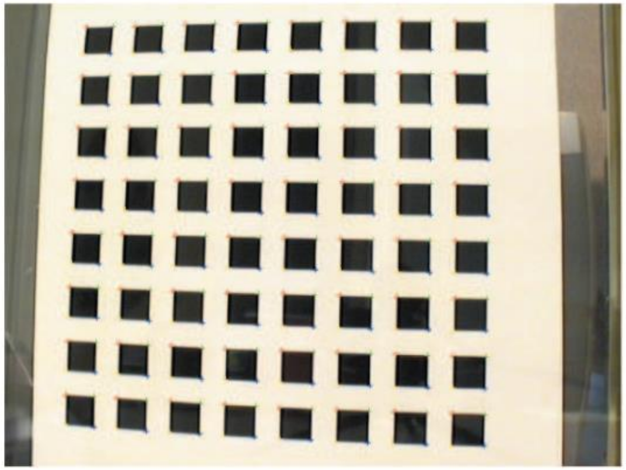
- Yes.
- Why? because K and R have a very special form:

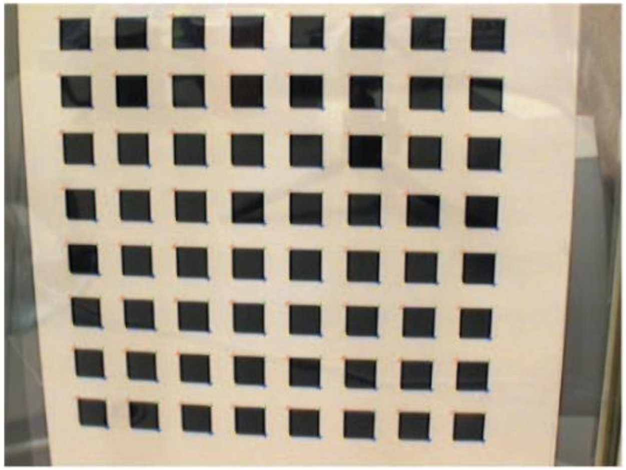
$$\begin{bmatrix} f_x & s & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

- QR decomposition
- Practically, use camera calibration packages (there is a good one in OpenCV)

Inserting a 3D known object…

- Also called "Tsai'scalibration" requires non-coplanar 3D points, is not very practical…
- Modern day calibration uses a planar calibration target





- Developed in 2000 by Zhang at Microsoft research

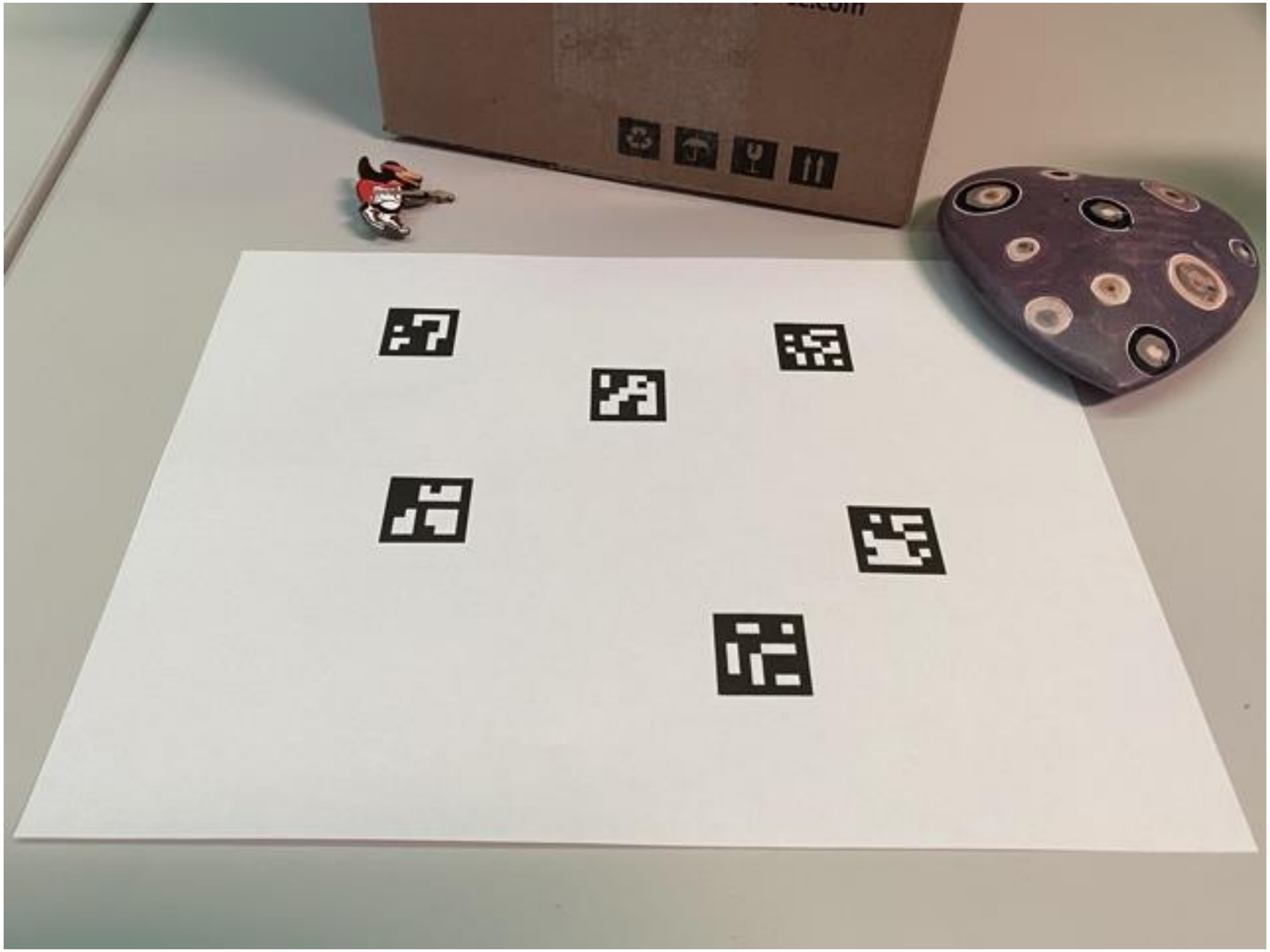
Zhang, A flexible new technique for camera calibration, IEEE Transaction
on Pattern Analysis and Machine Intelligence, 2000

Doesn't plane give you homography?

- Yes! If it's a plane, it's only a homography, so instead of recovering 3x4 matrix, you will recover 3x3 in Zhang's method
- The 3x3 gives first two columns of R and T

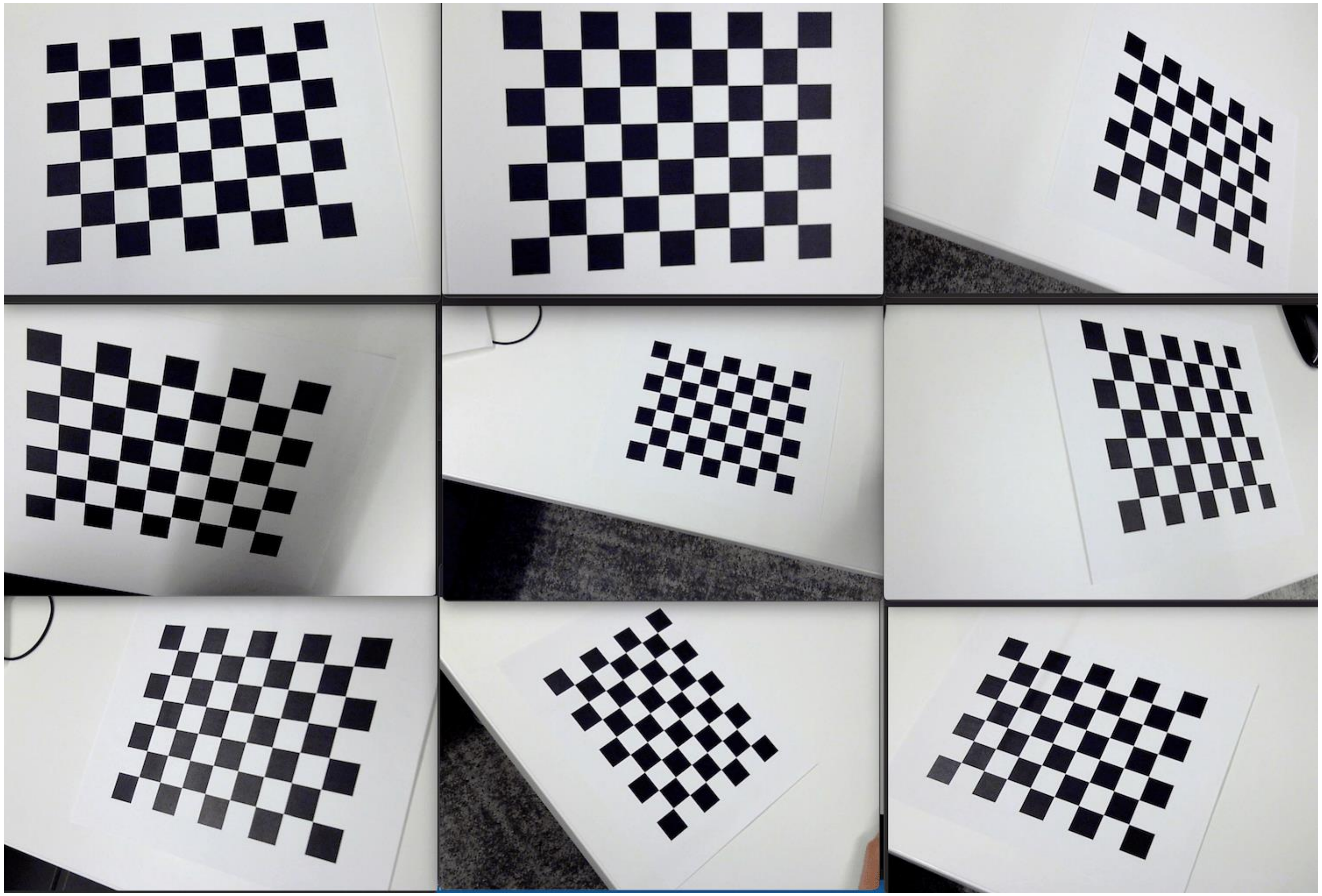
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} & r_{22} & t_2 \\ r_{31} & r_{32} & t_3 \end{bmatrix}$$

You will use Aruco tags



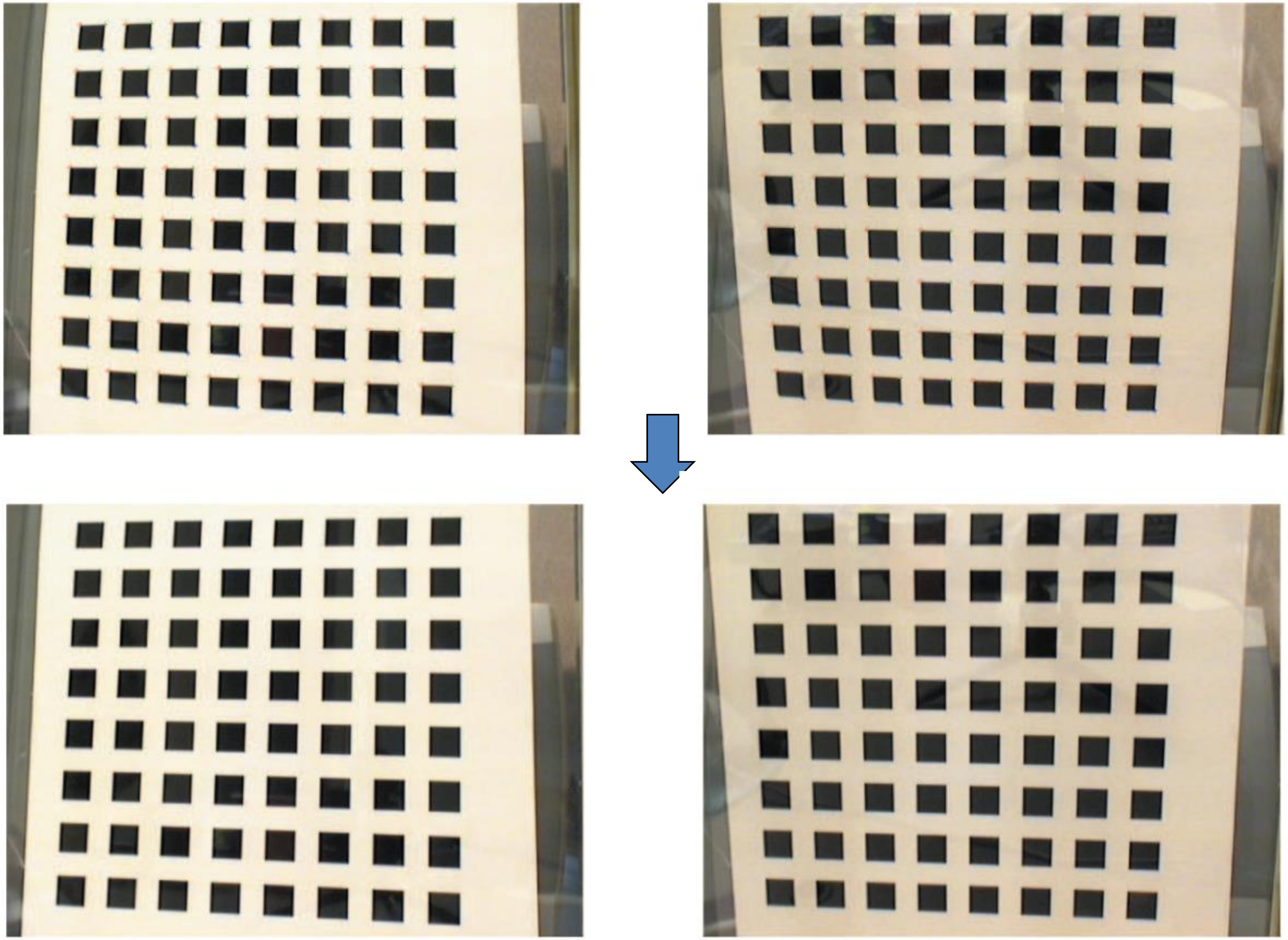
In practice: Step 0

- Calibrate your intrinsics first, also estimates lens distortion (cv2.calibrateCamera)



Cv2.undistort()

Step 1: Undistort your image



Step 2: Estimate camera with PnP

cv2.solvePnP()

- PnP – "Perspective-n-Point" problem:
- Estimate extrinsic parameters given n correspondences

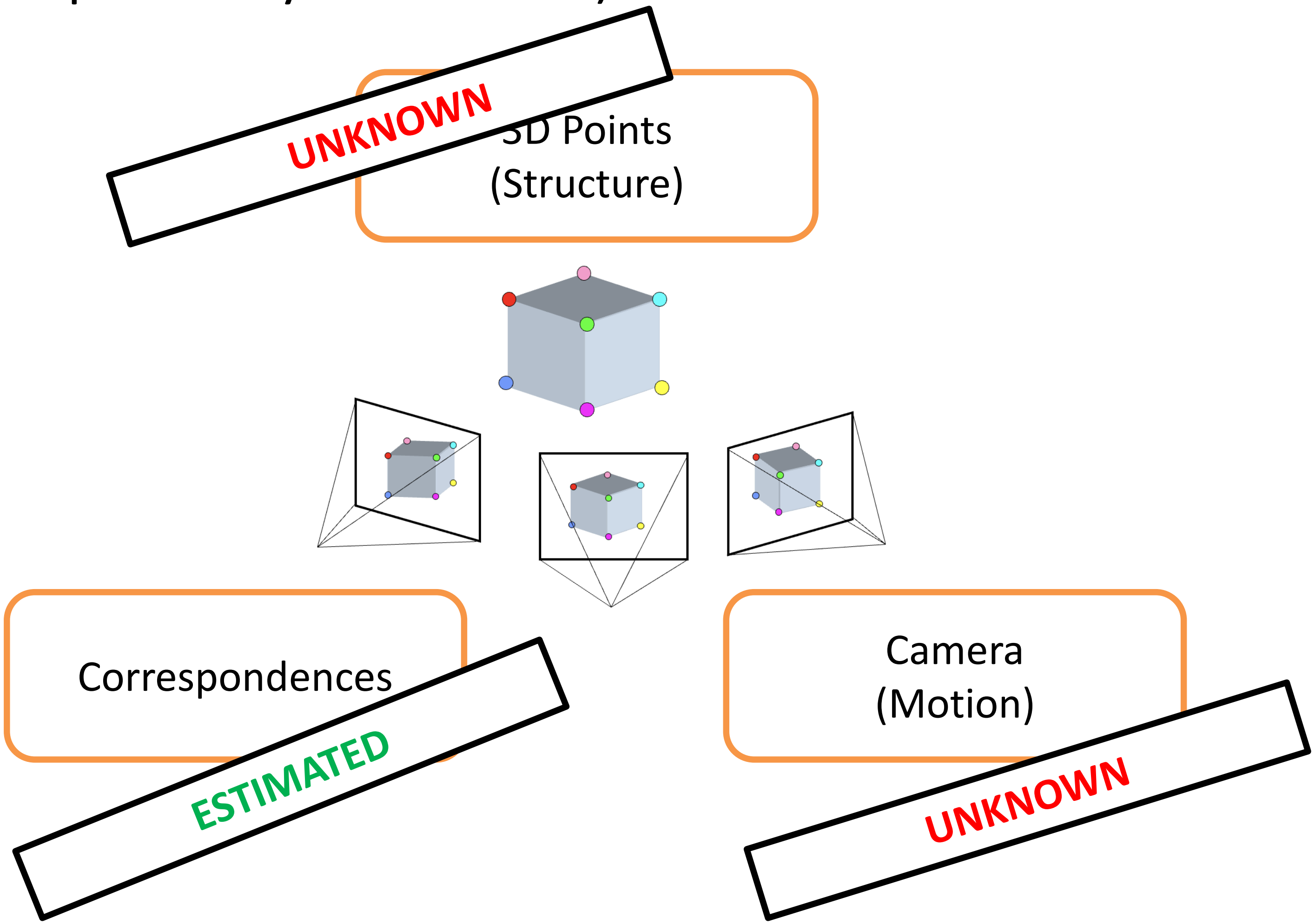
Minimize reprojection loss with non-linear least squares

$$\min_{R,T} \sum \|x_i - K[R\ T]X\|^2$$

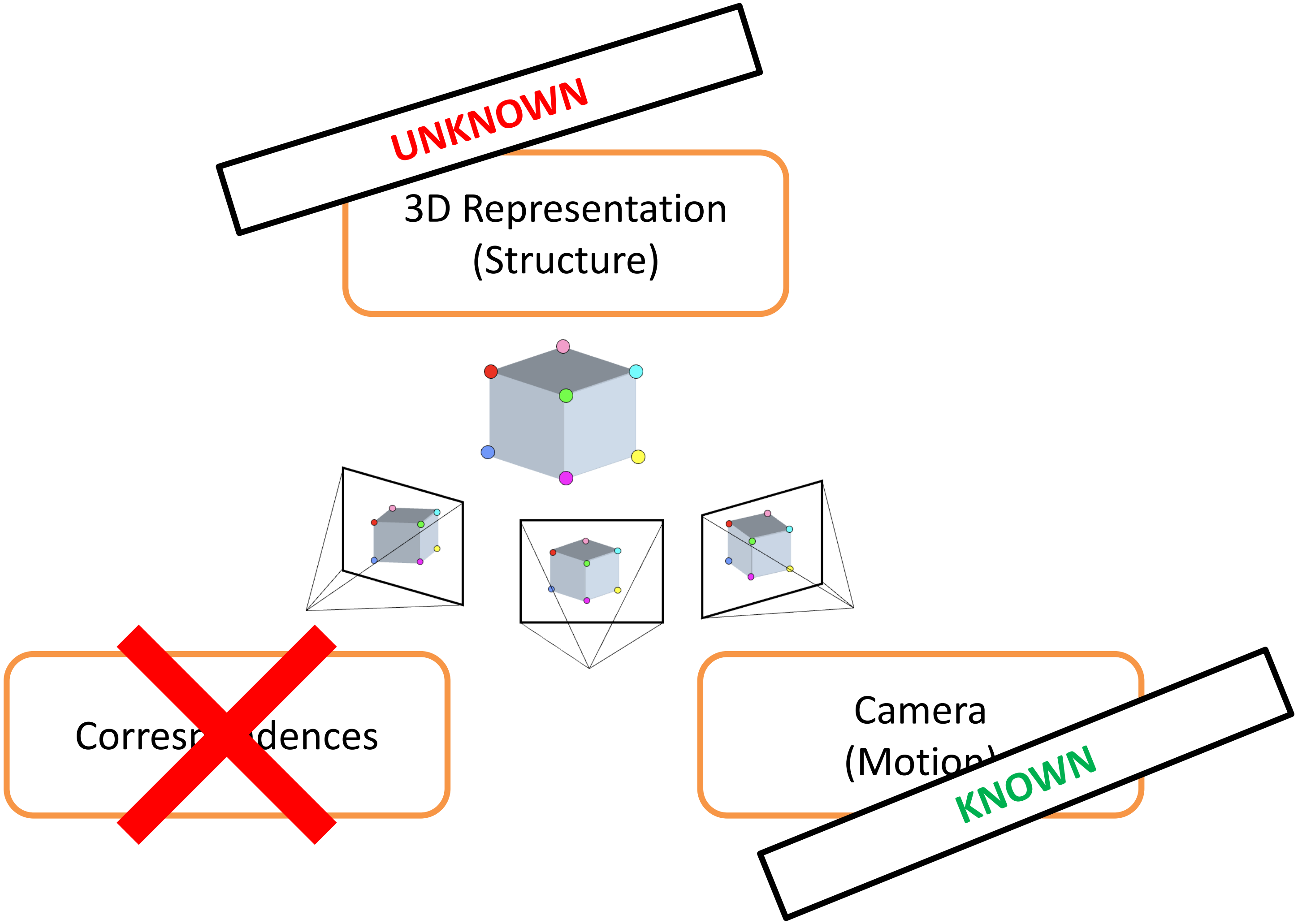
In general you do DLT first (Ax=0), then use that as initialization, or do other algorithms like Efficient PnP

Putting it all together

- Structure-from-Motion: You know nothing!
  (except ok maybe intrinsics)

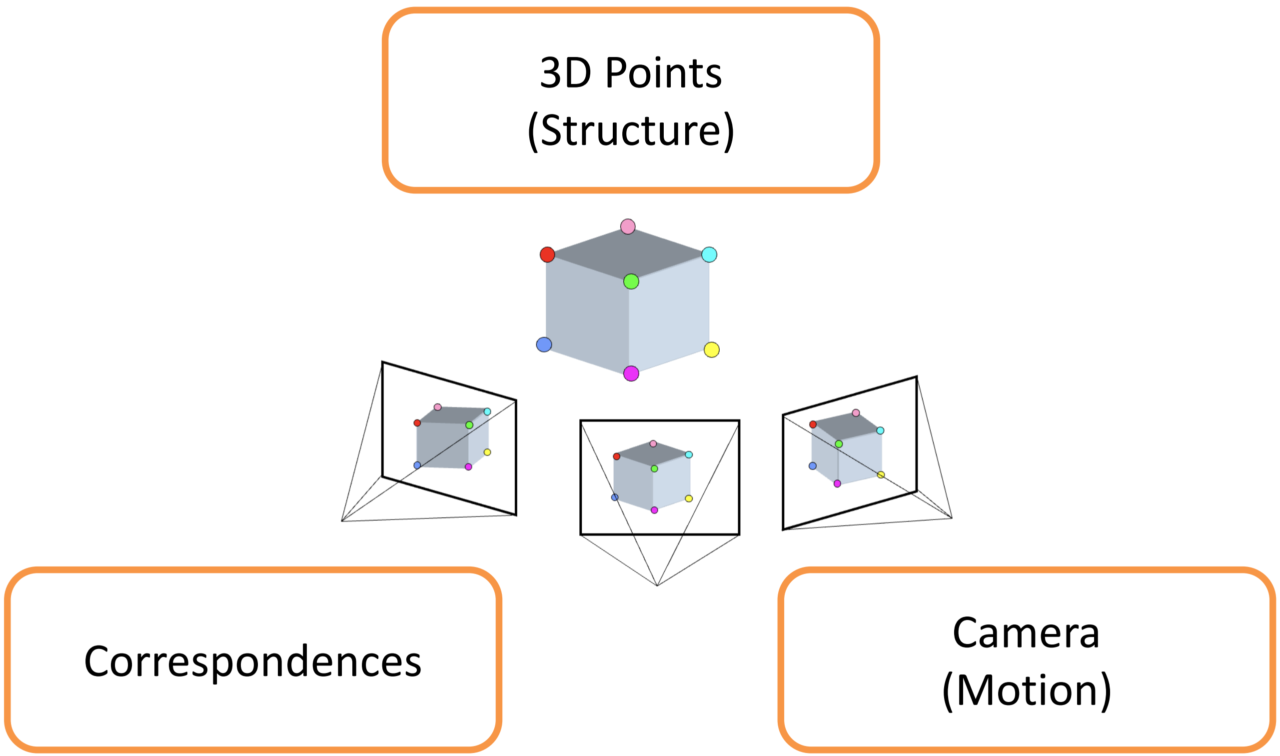


(after that): Neural Rendering

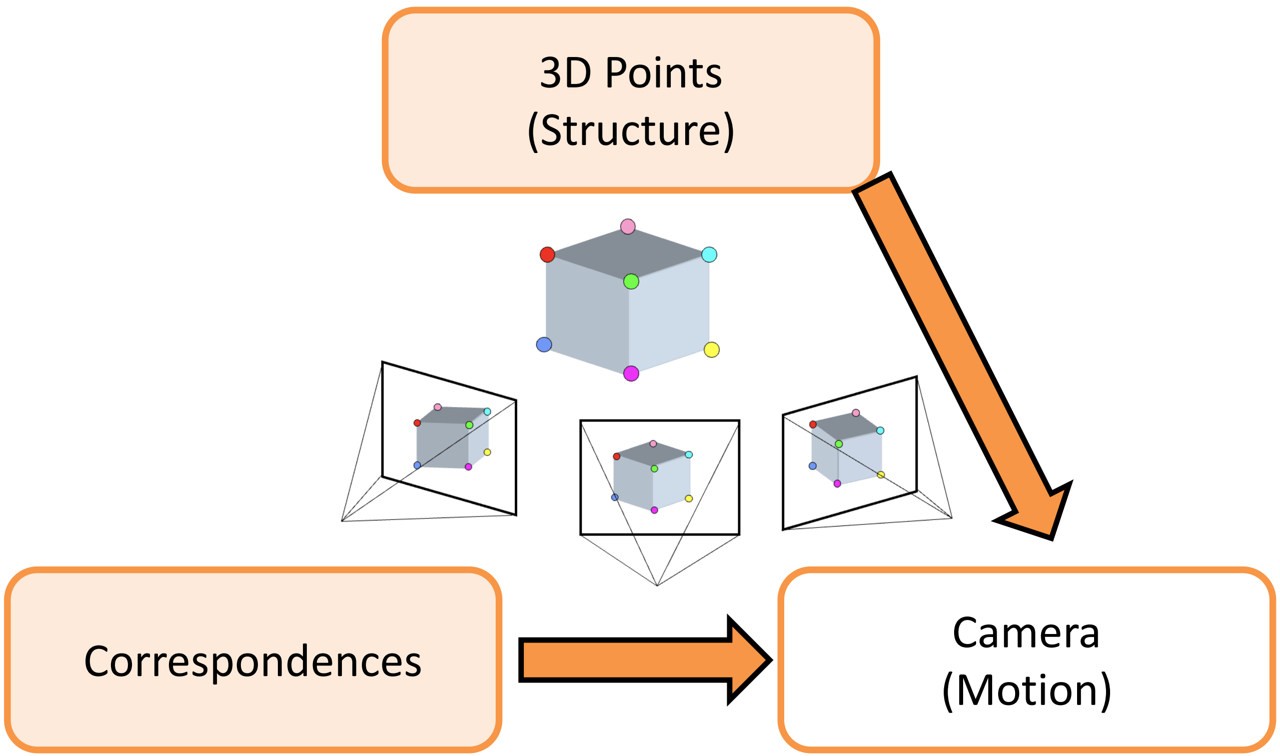


A form of multi-view stereo, more on this in the NeRF lecture.

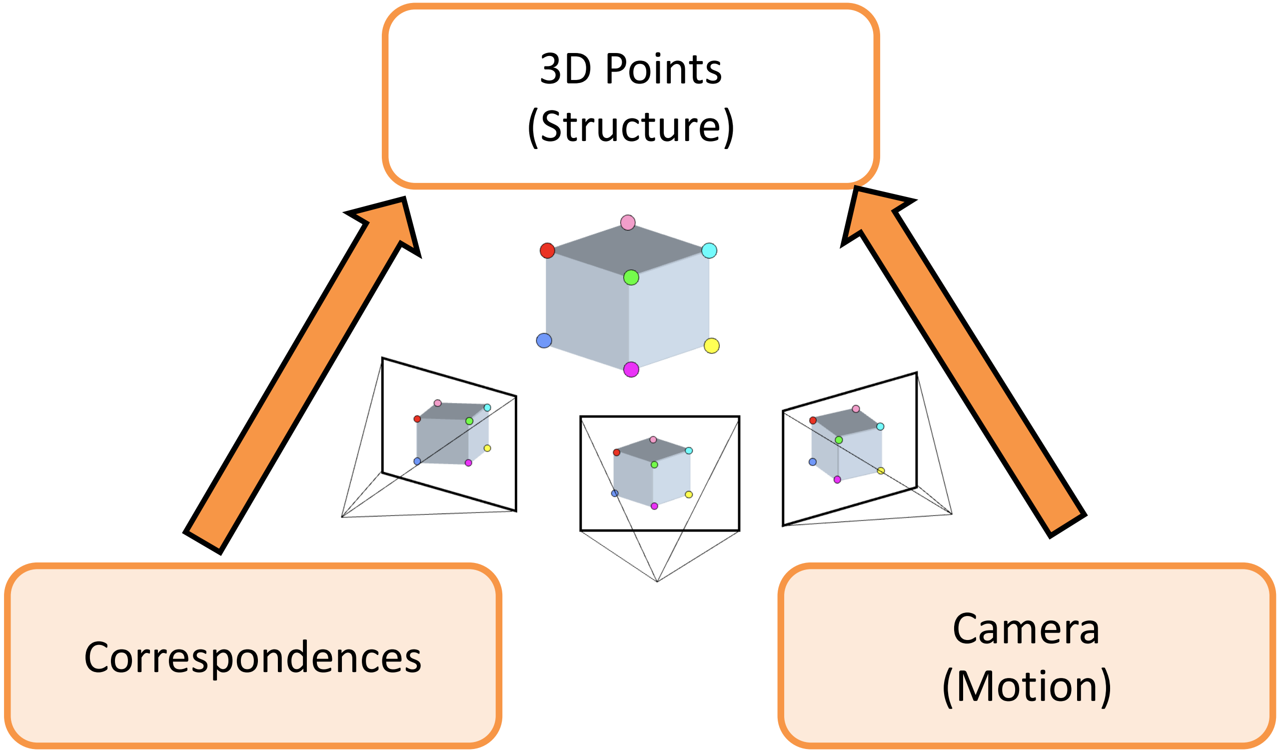
if you know 2 you get the other:



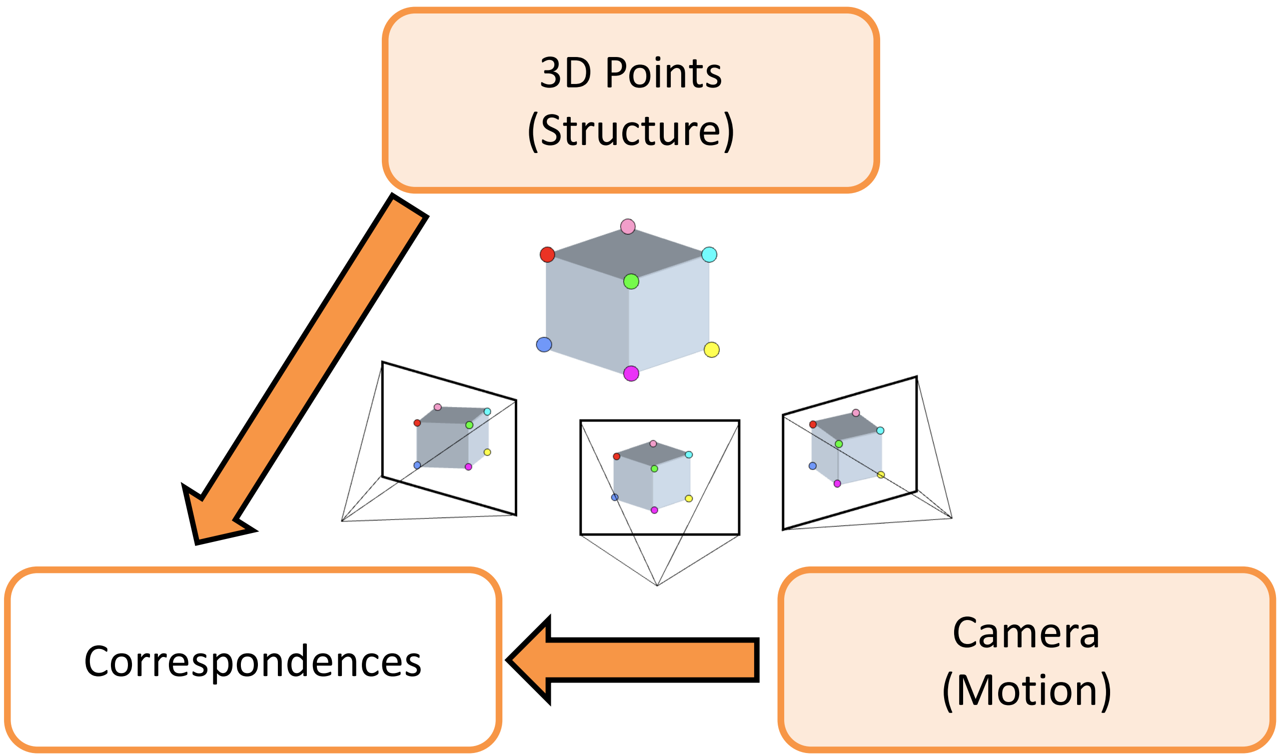
Camera Calibration; aka Perspective-n-Point



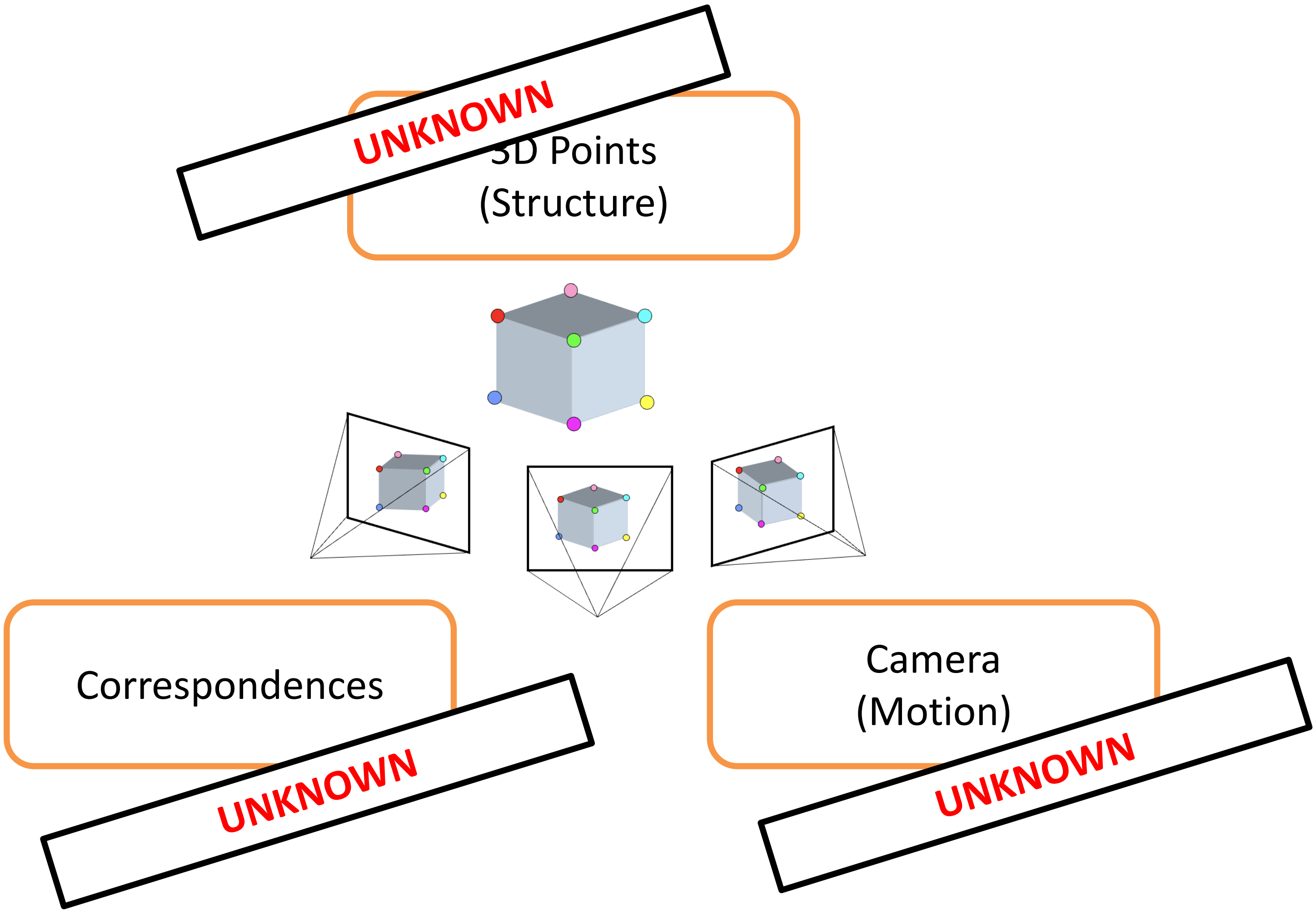
Stereo (w/2 cameras); aka Triangulation



You can easily get correspondence via projection from 3D points + Camera



Ultimate: Structure-from-Motion



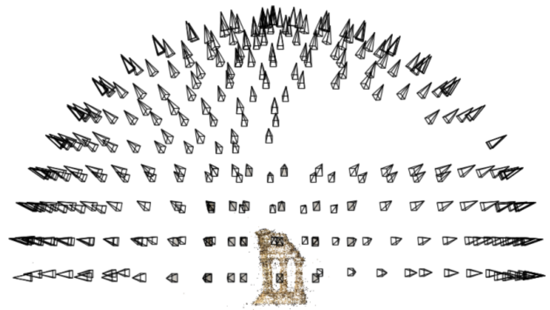
Start from nothing known (except maybe intrinsics), exploit the relationship to slowly get the right answer

Photo Tourism

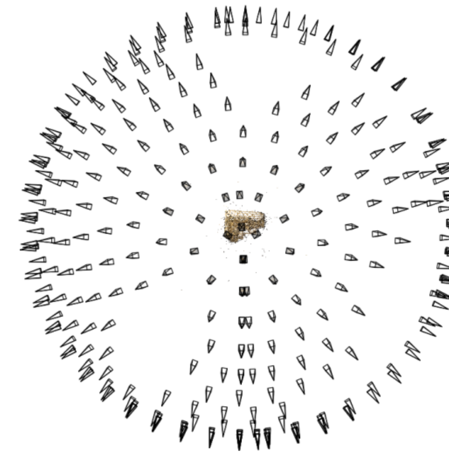
Noah Snavely, Steven M. Seitz, Richard Szeliski, "Photo tourism: Exploring photo collections in 3D," SIGGRAPH 2006

https://youtu.be/mTBPGuPLI5Y

Structure from motion



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 $\mathbf{K}_j$

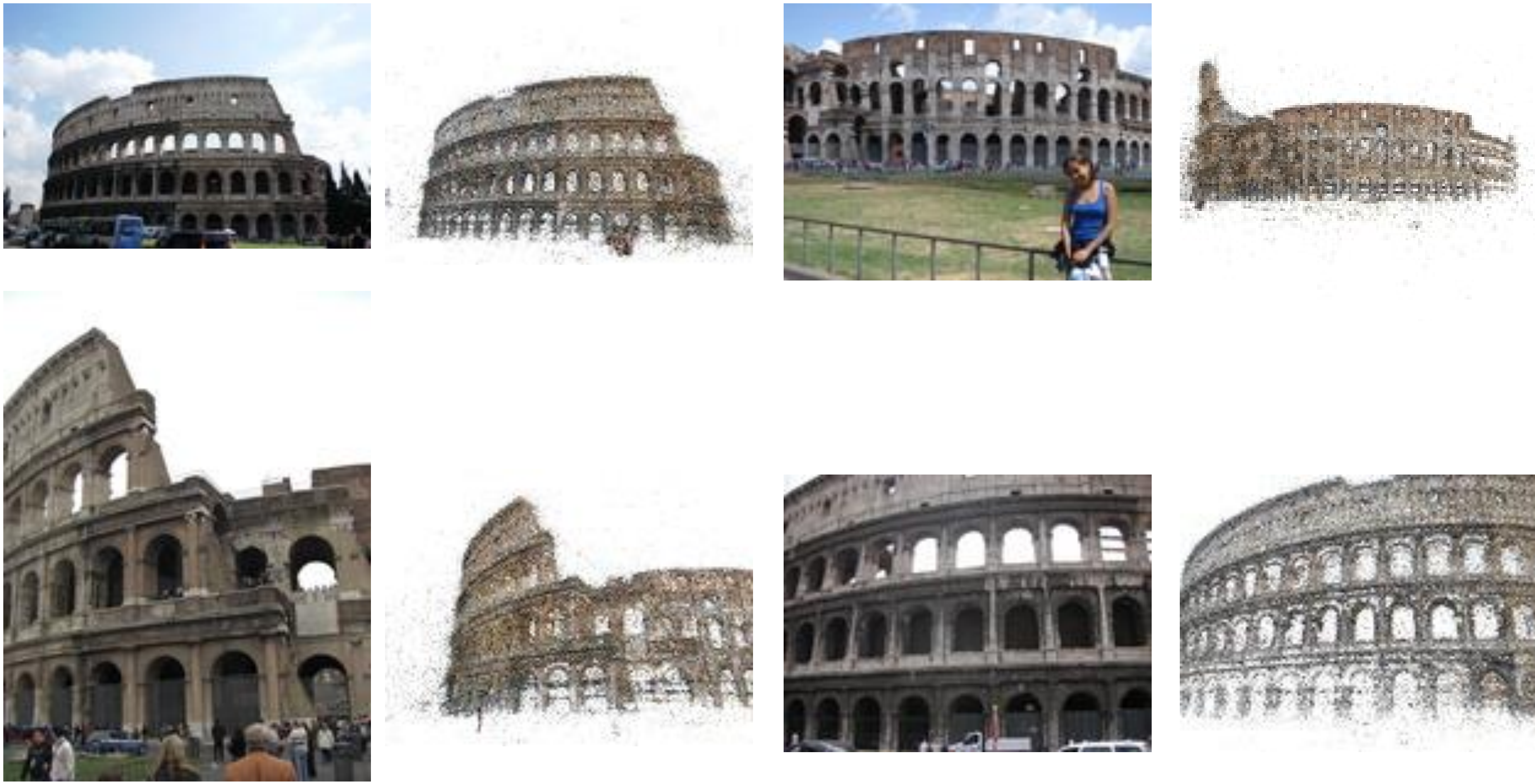
- Objective function: minimize *reprojection error*

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

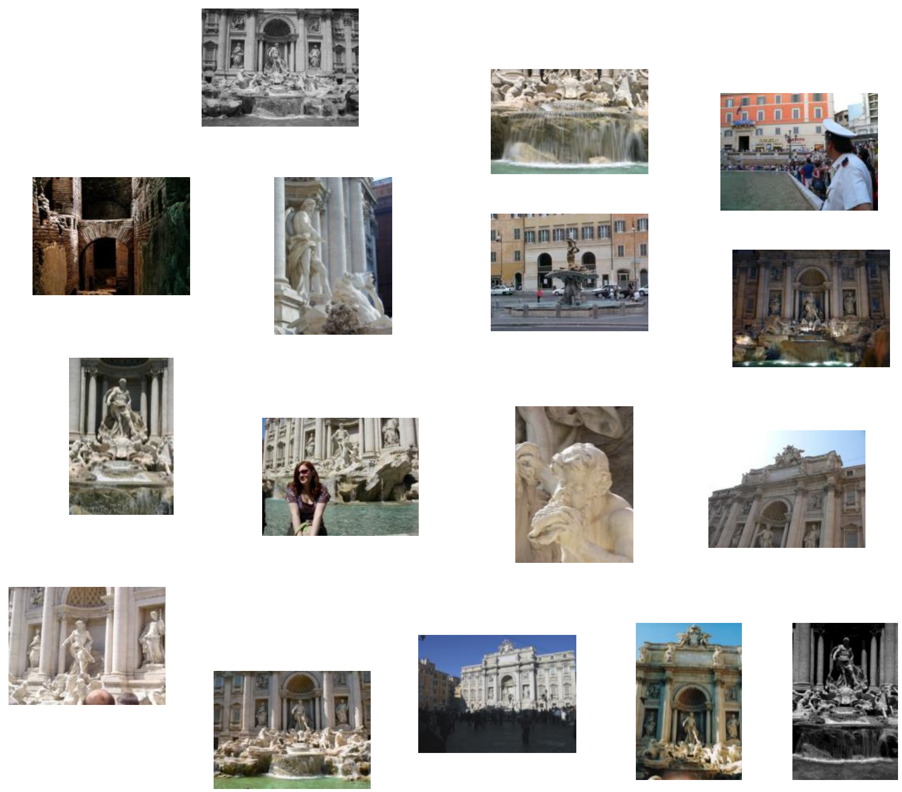
Building Rome in a Day, Agarwal et al. ICCV 2009

First step: Correspondence

- Feature detection and matching

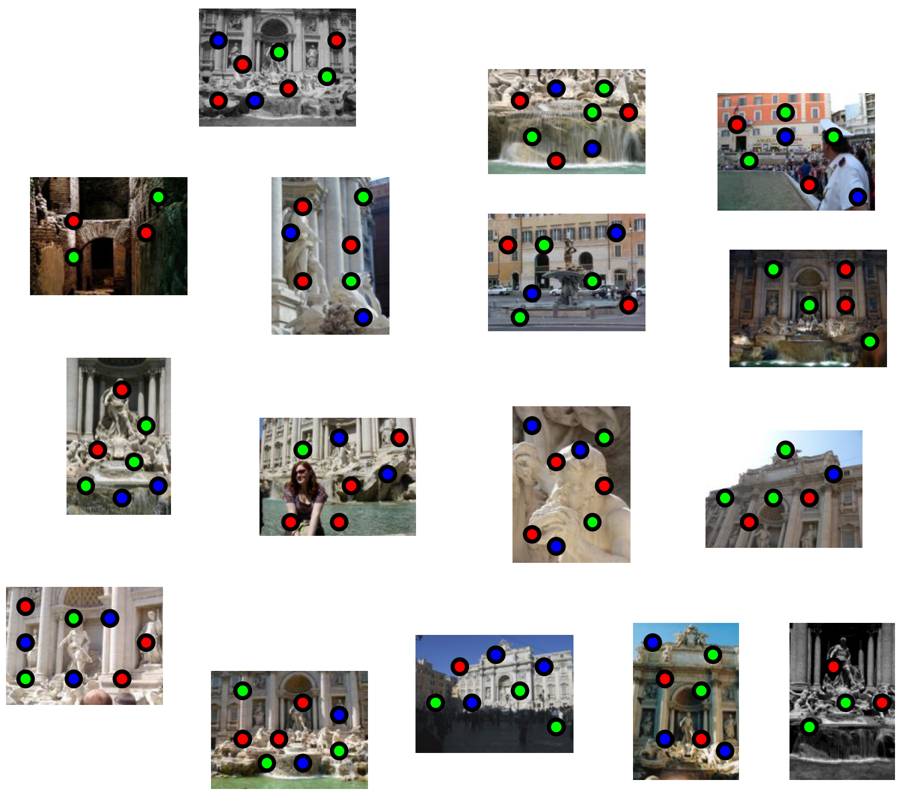
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



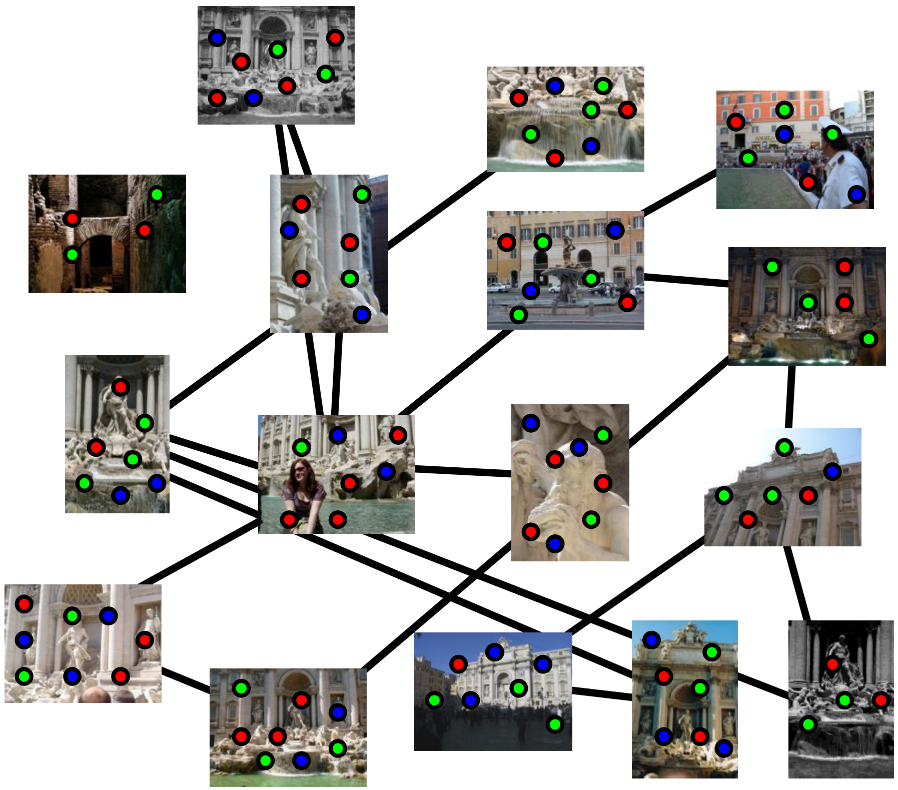
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



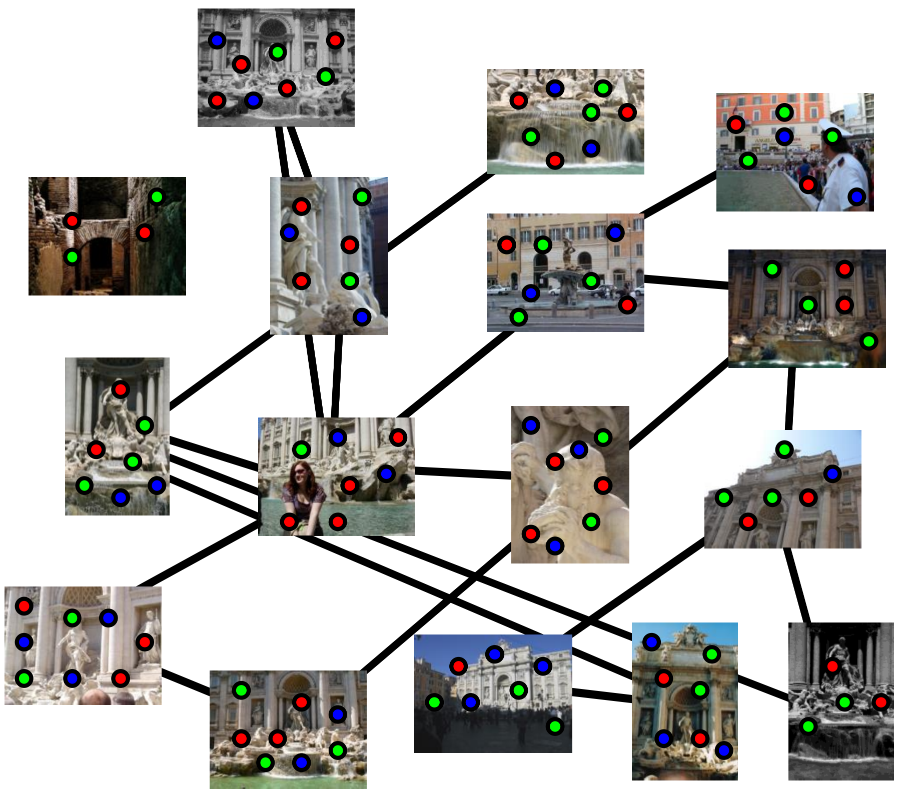
Feature matching

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

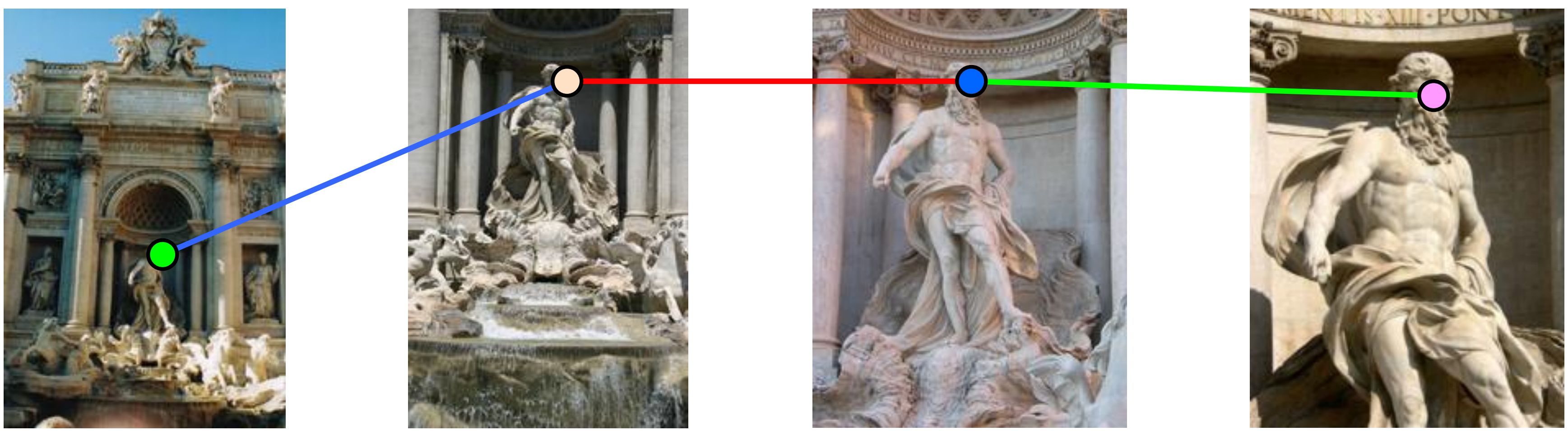
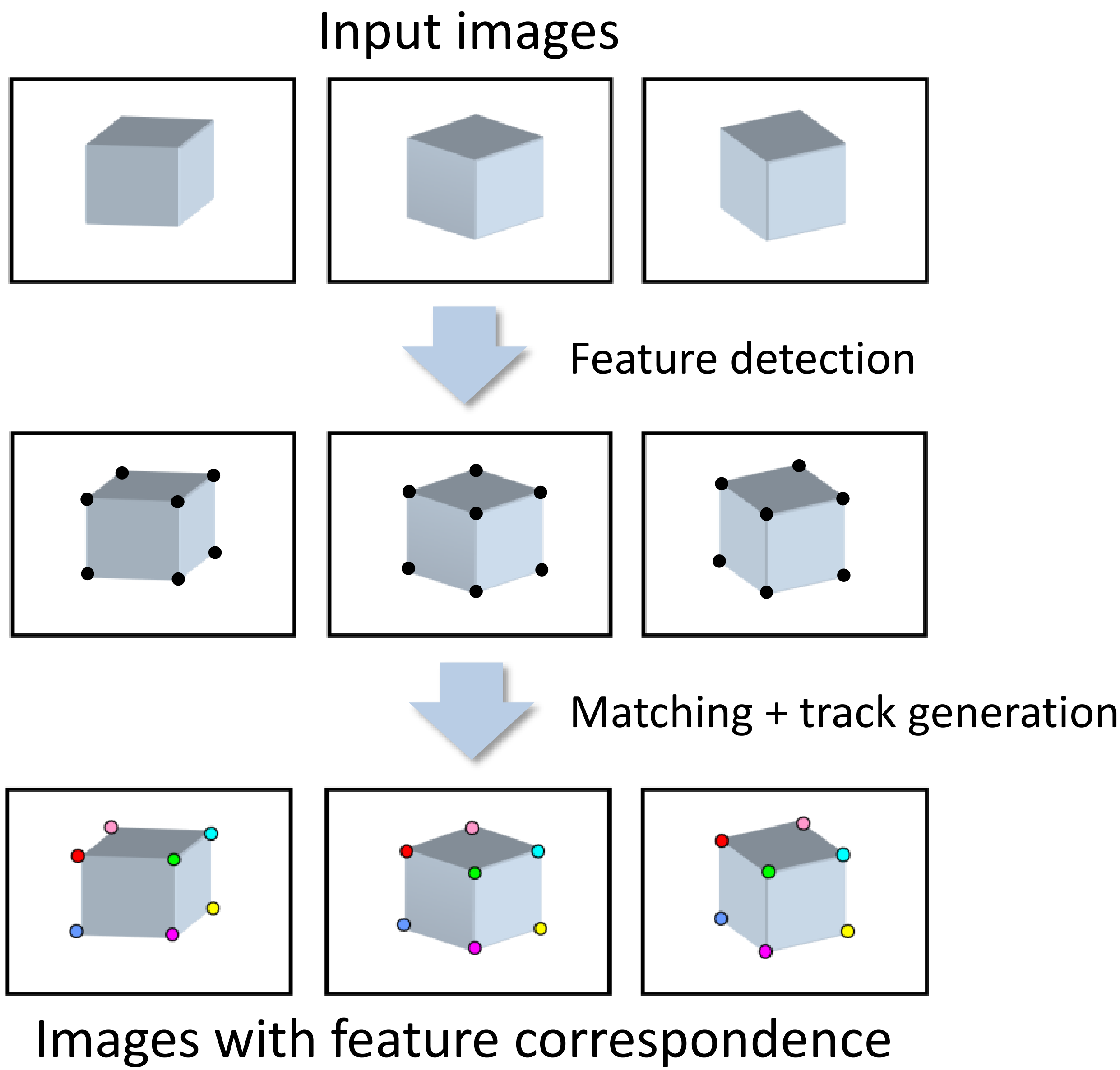


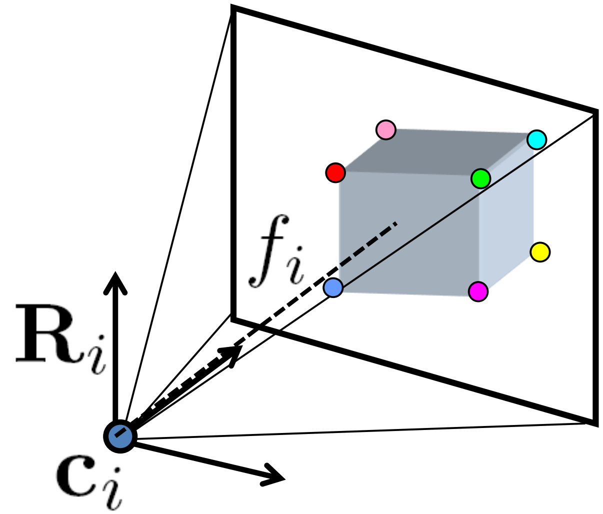
Image 1 Image 2 Image 3 Image 4

The story so far…

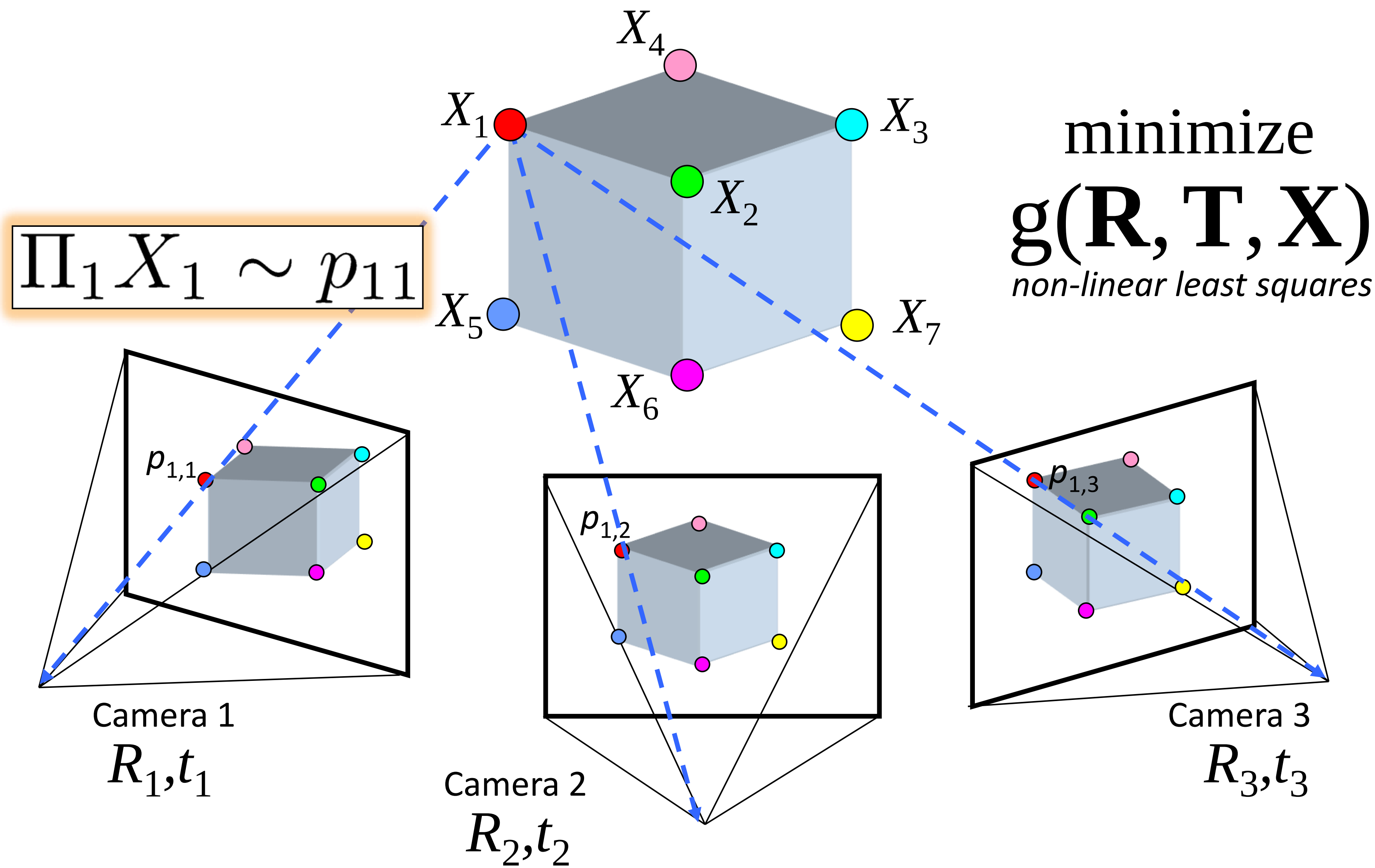


Review: Points and cameras

- 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



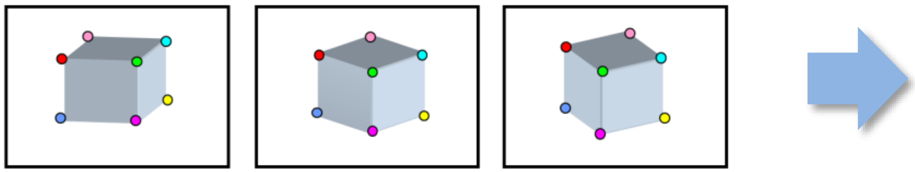
Structure from motion

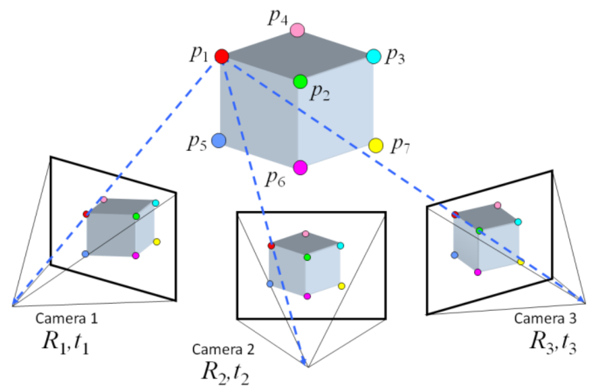
- Minimize sum of squared reprojection errors:

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

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

Solving structure from motion



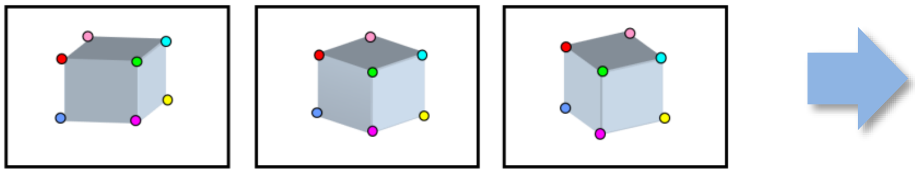


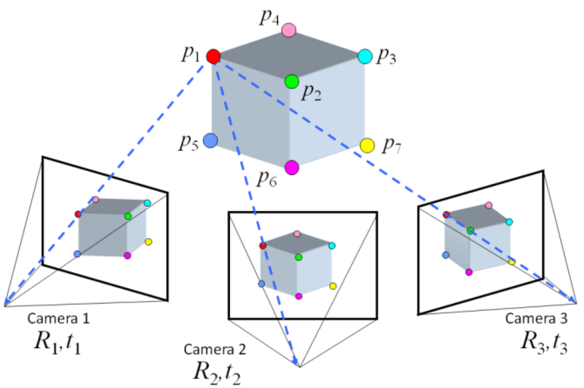
Inputs: feature tracks

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



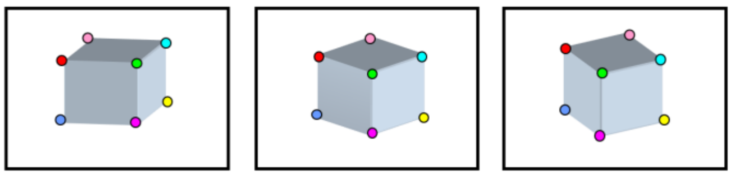


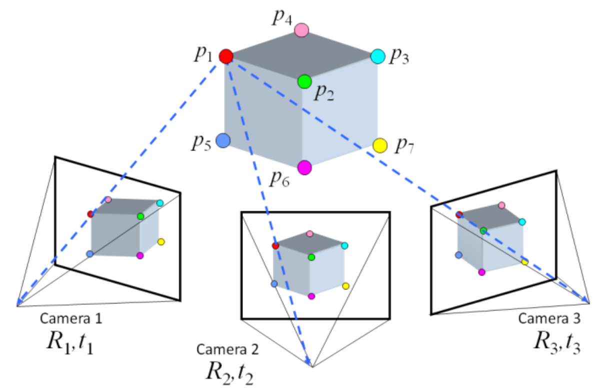
Inputs: feature tracks

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



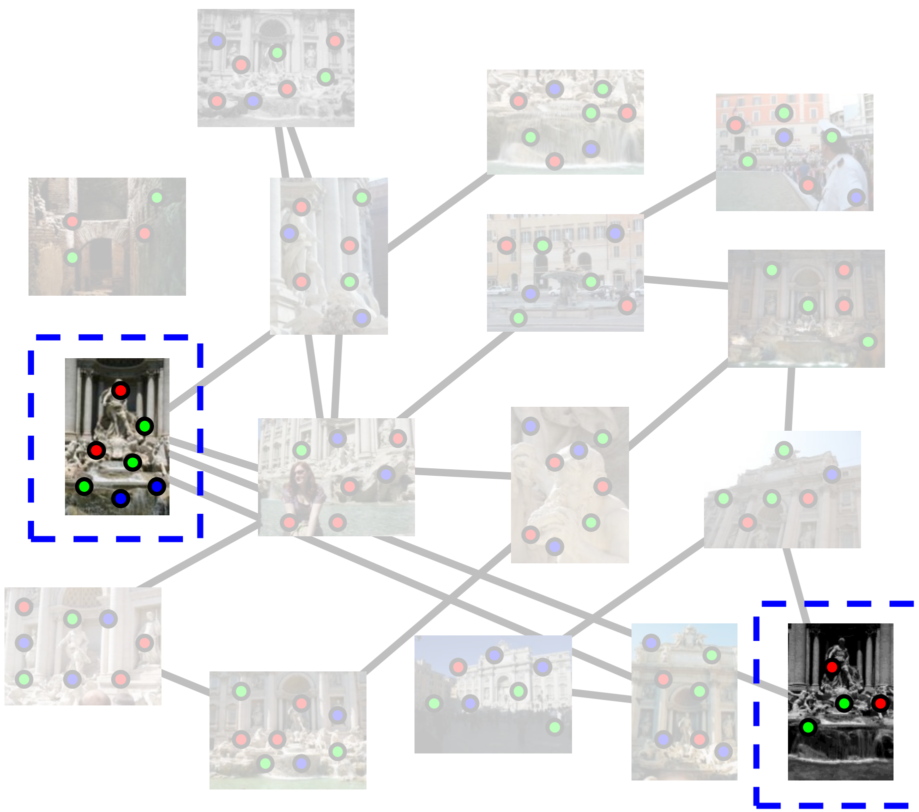


Inputs: feature tracks

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

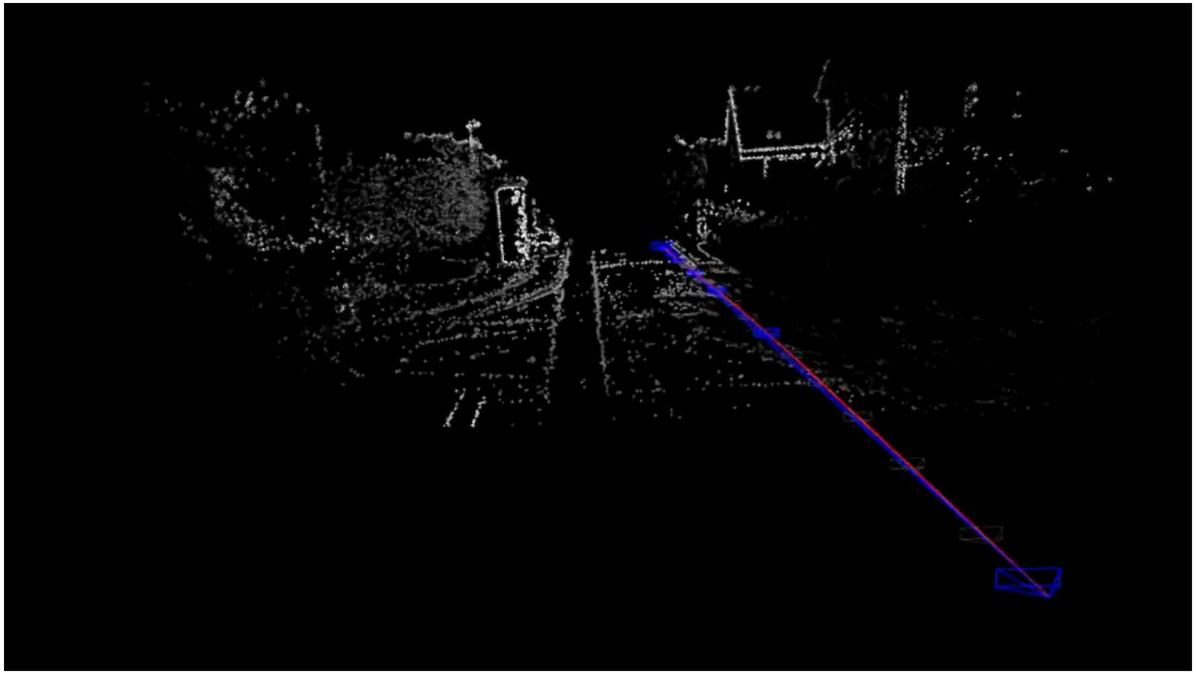

✓ large baseline
✓ lots of matches

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

Now what, are we done?

- What does SfM give you

Sparse points!!! Why?

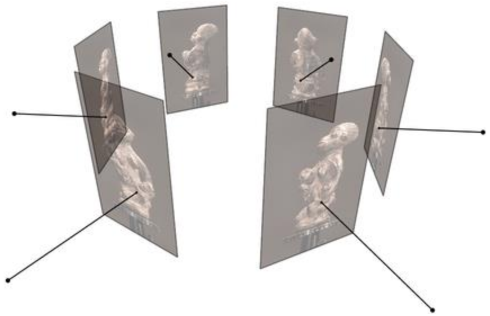
What if we want solid models?

Slide credit: Noah Snavely

Multi-View Stereo
(after SfM), i.e. known camera

Multi-view Stereo (Lots of calibrated images)

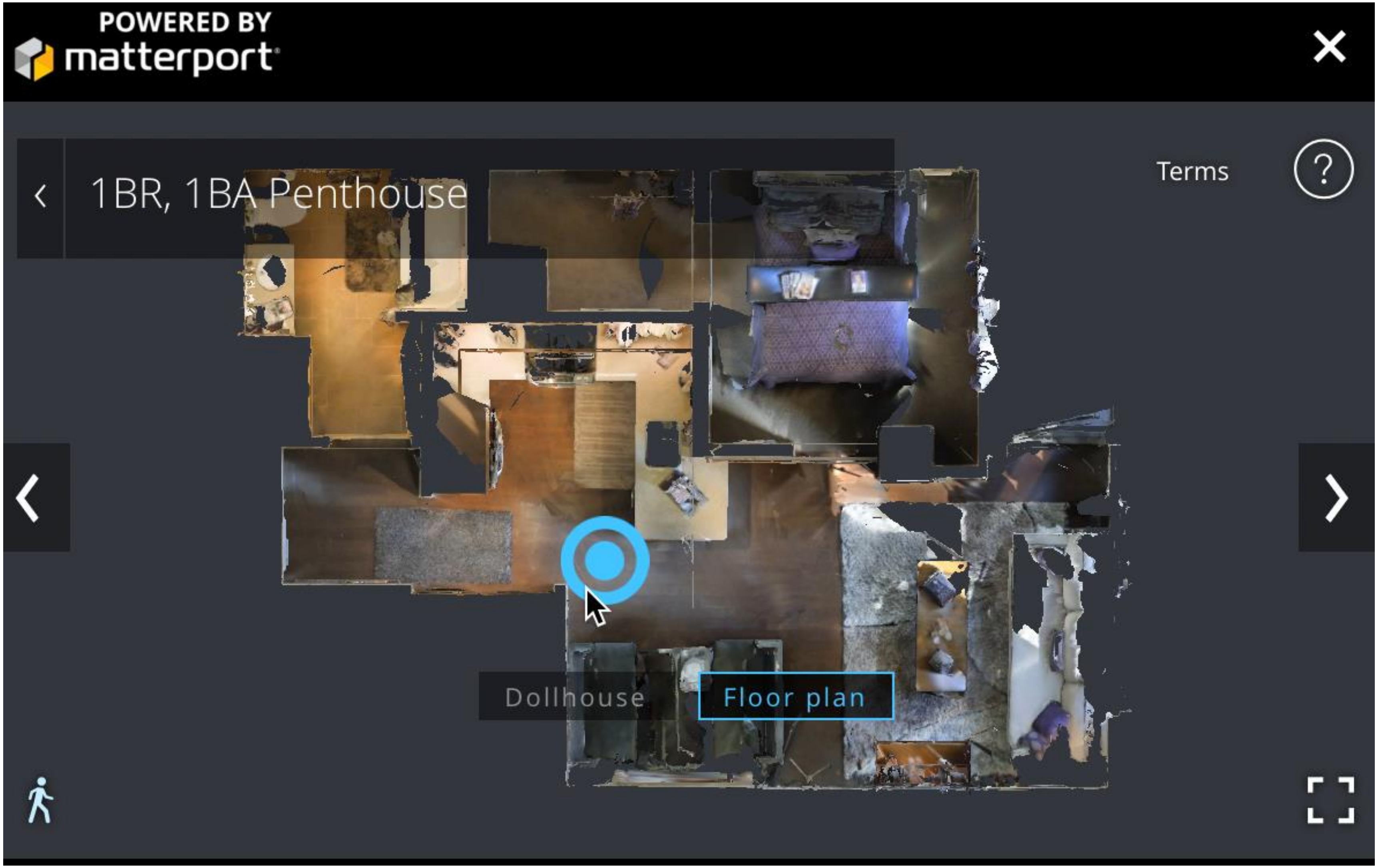
- Input: calibrated images from several viewpoints (known camera: intrinsics and extrinsics)
- Output: 3D Model



Figures by Carlos Hernandez

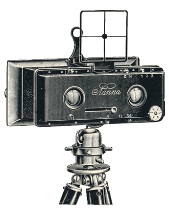
In general, conducted in a controlled environment with multi-camera setup that are all calibrated

Slide credit: Noah Snave

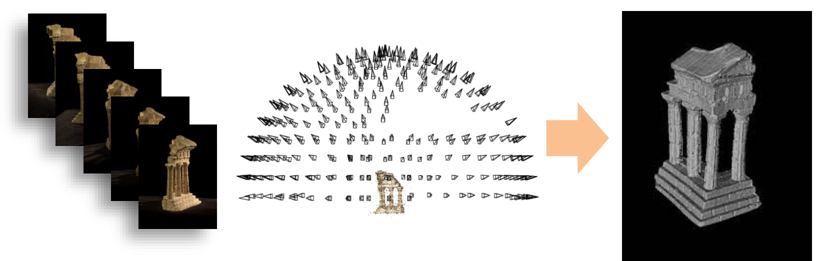

POWERED BY
matterport
1BR, 1BA Penthouse
Terms
Dollhouse
Floor plan

Multi-view Stereo

Problem formulation: given several images of the same object or scene, compute a representation of its 3D shape



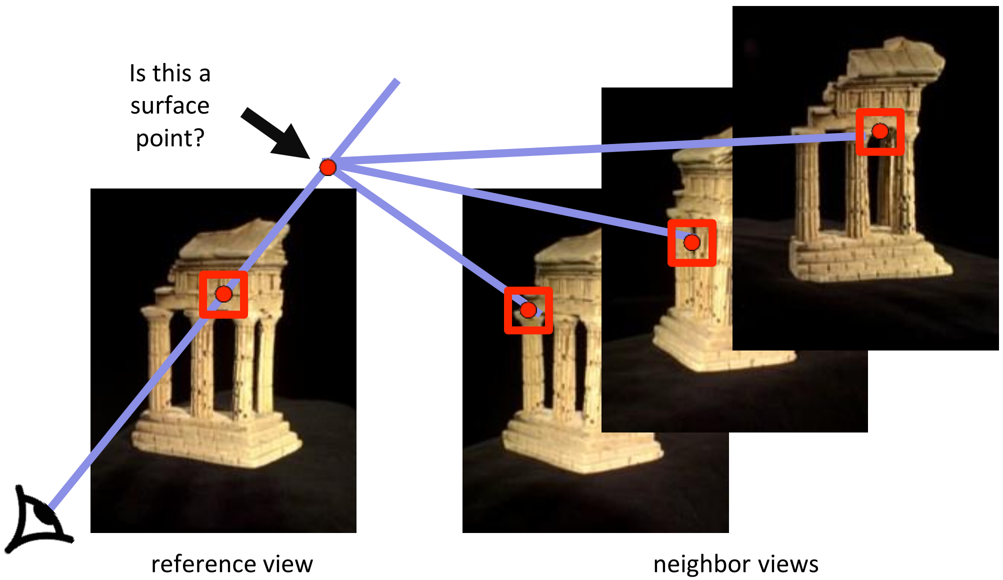
Binocular Stereo



Multi-view stereo

Slide credit: Noah Snavely

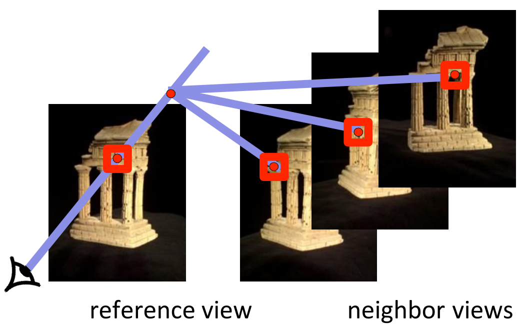
Multi-view stereo: Basic idea

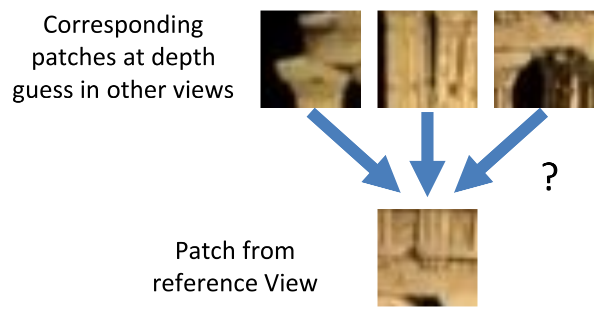


Source: Y. Furukawa

Multi-view stereo: Basic idea

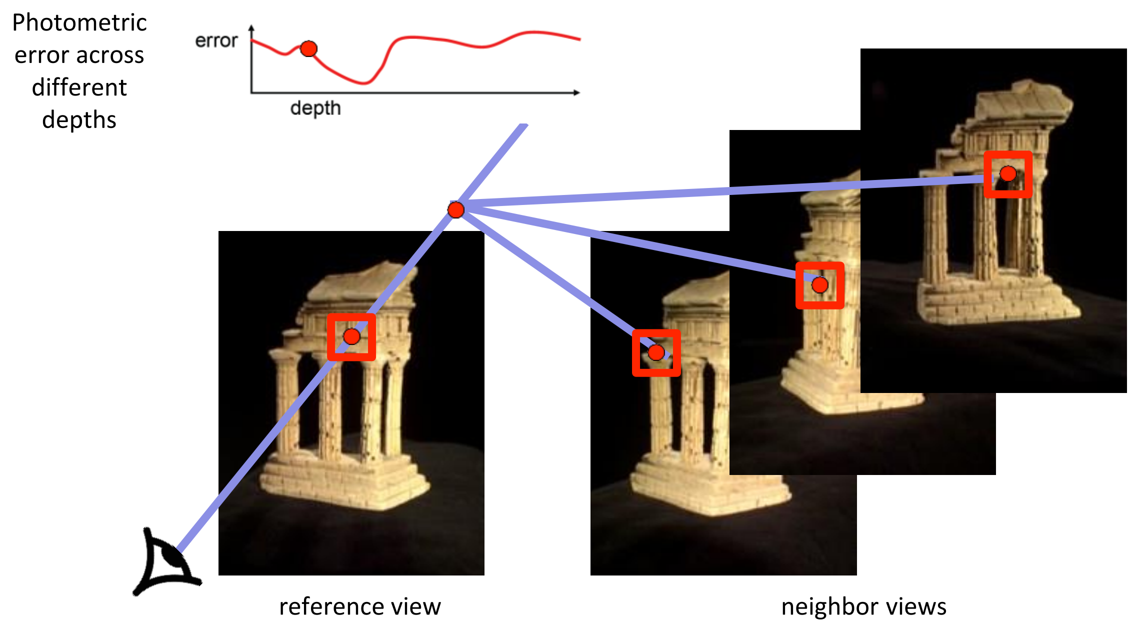
Evaluate the likelihood of geometry at a particular depth for a particular reference patch:





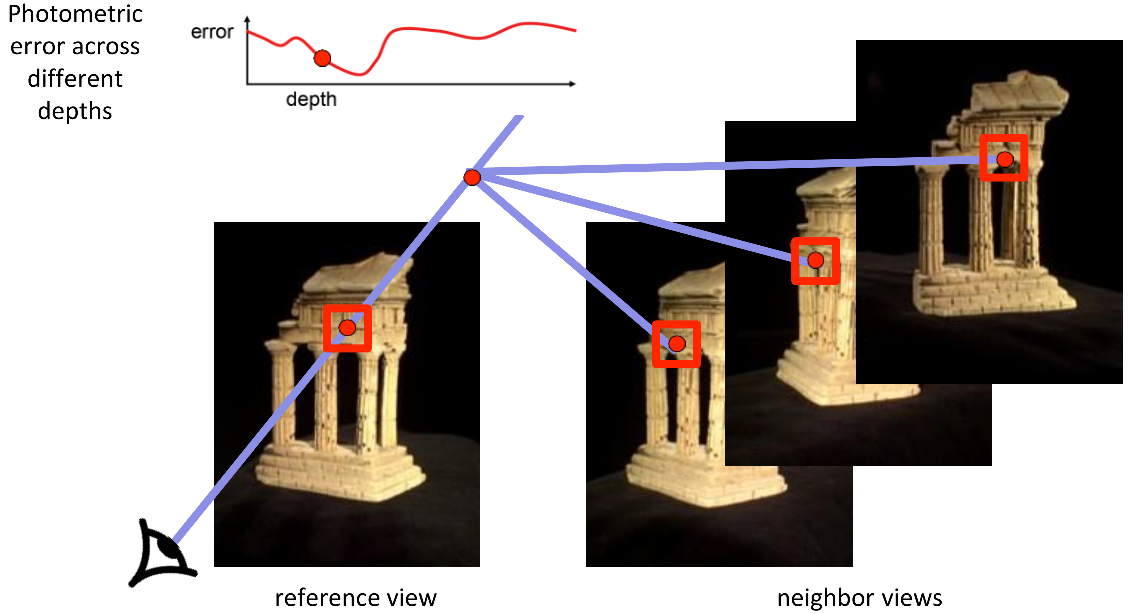
Source: Y. Furukawa

Multi-view stereo: Basic idea



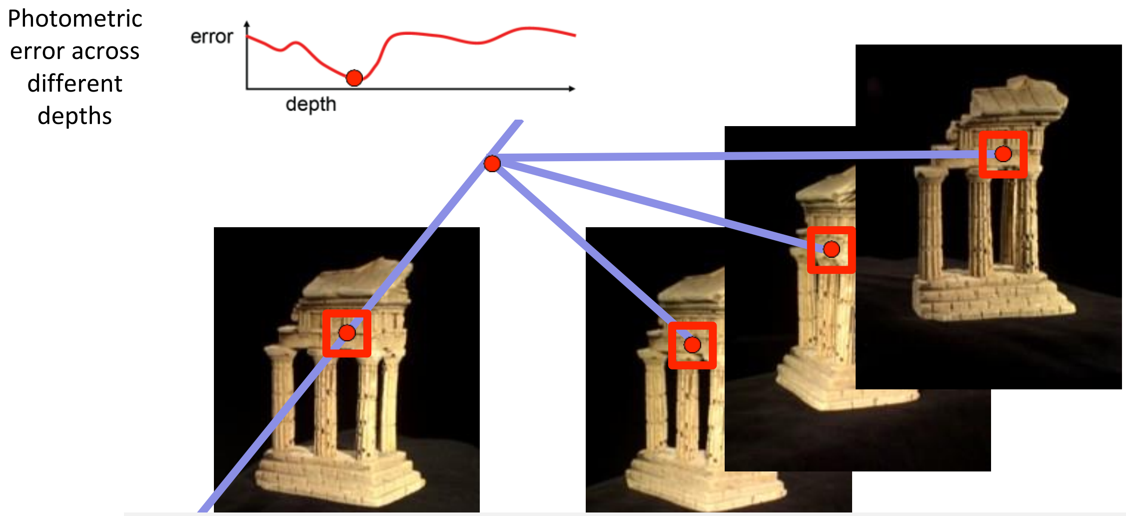
Source: Y. Furukawa

Multi-view stereo: Basic idea



Source: Y. Furukawa

Multi-view stereo: Basic idea



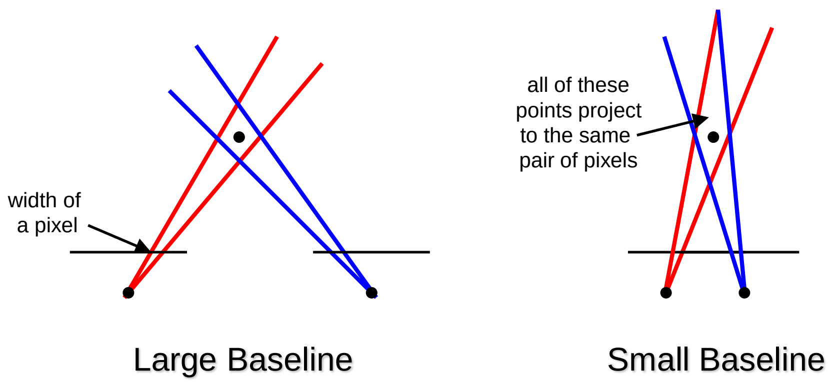
In this manner, solve for a depth map over the whole reference view

Multi-view stereo: advantages over 2 view

- Can match windows using more than 1 other image, giving a **stronger match signal**
- If you have lots of potential images, can **choose the best subset** of images to match per reference image
- Can reconstruct a depth map for each reference frame, and the merge into a **complete 3D model**

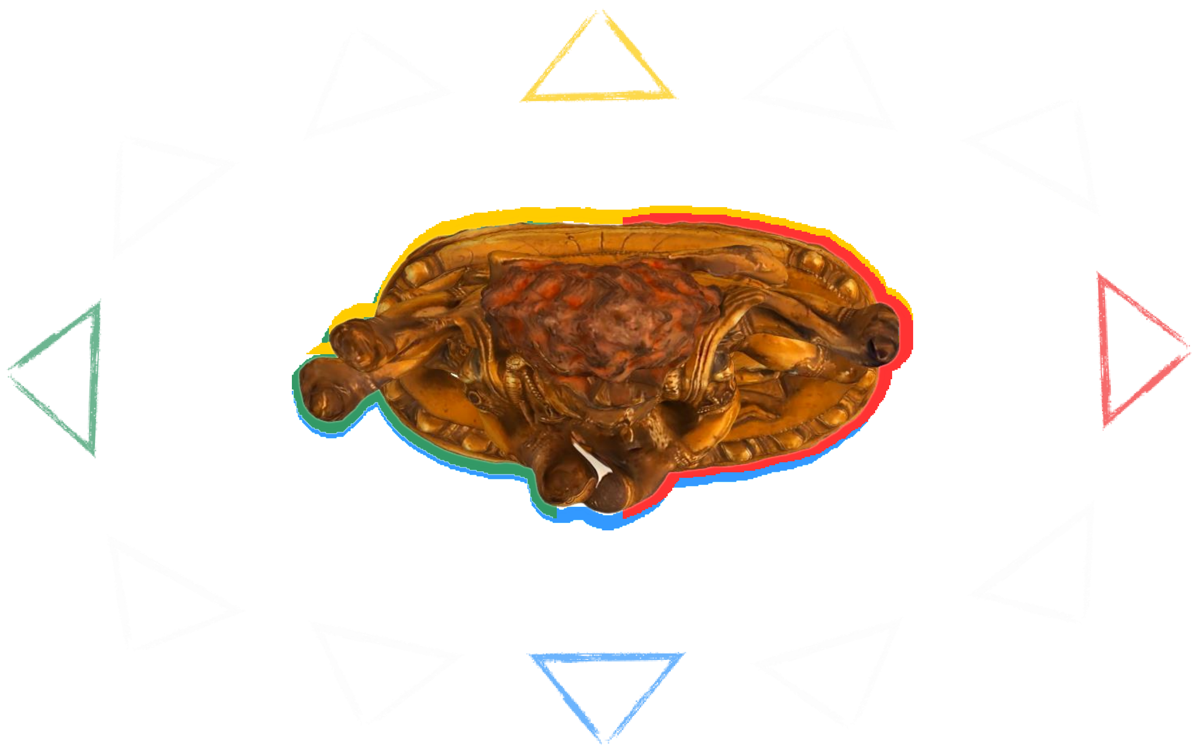
Source: Y. Furukawa

Choosing the baseline

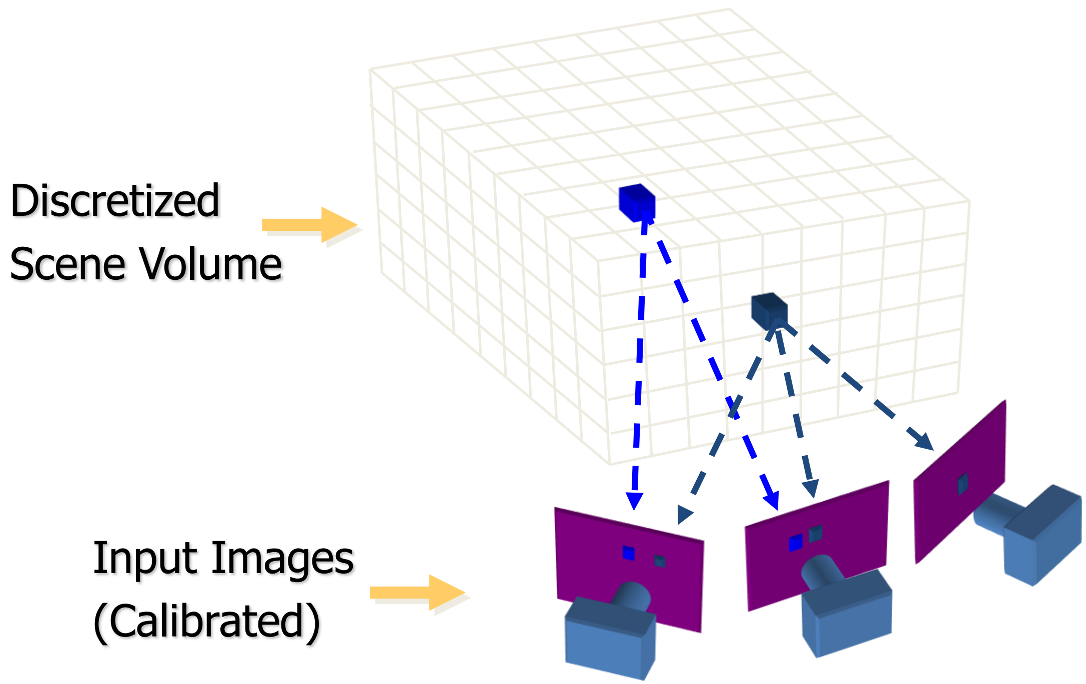


- What's the optimal baseline?
  - Too small: large depth error
  - Too large: difficult search problem

Slide credit: Noah Snavely



Volumetric stereo



Goal: Assign RGB values to voxels in V
photo-consistent with images

Space Carving

- Space Carving Algorithm
  - Initialize to a volume V containing the true scene
  - Choose a voxel on the outside of the volume
  - Project to visible input images
  - Carve if not photo-consistent
  - Repeat until convergence

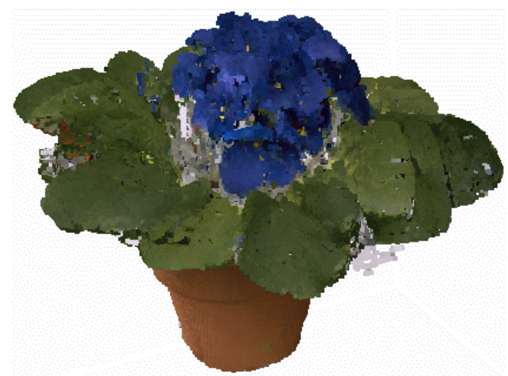
K. N. Kutulakos and S. M. Seitz, [A Theory of Shape by Space Carving](), *ICCV* 1999

Space Carving Results

Input Image (1 of 45)

Reconstruction

Reconstruction



Reconstruction

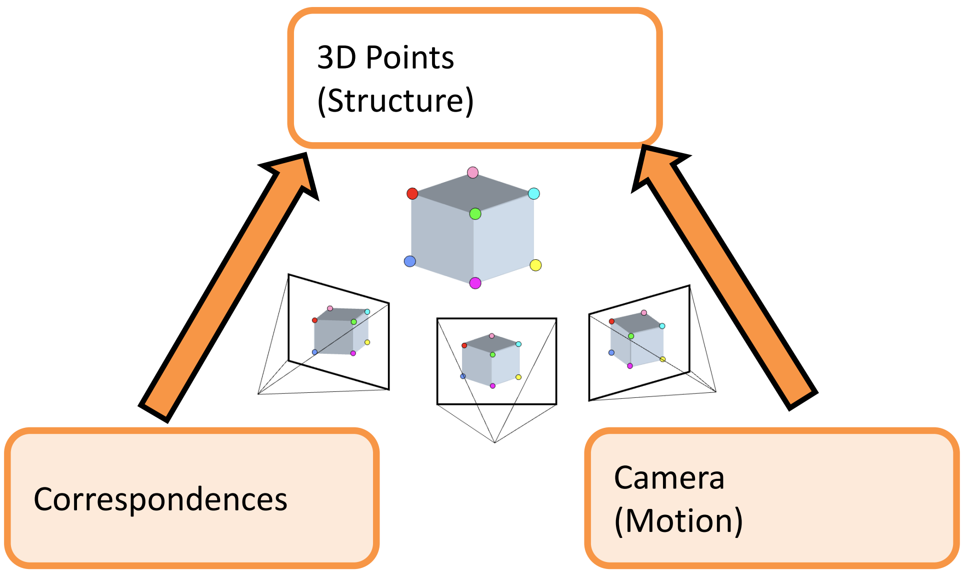
Source: S. Seitz

Tool for you: COLMAP

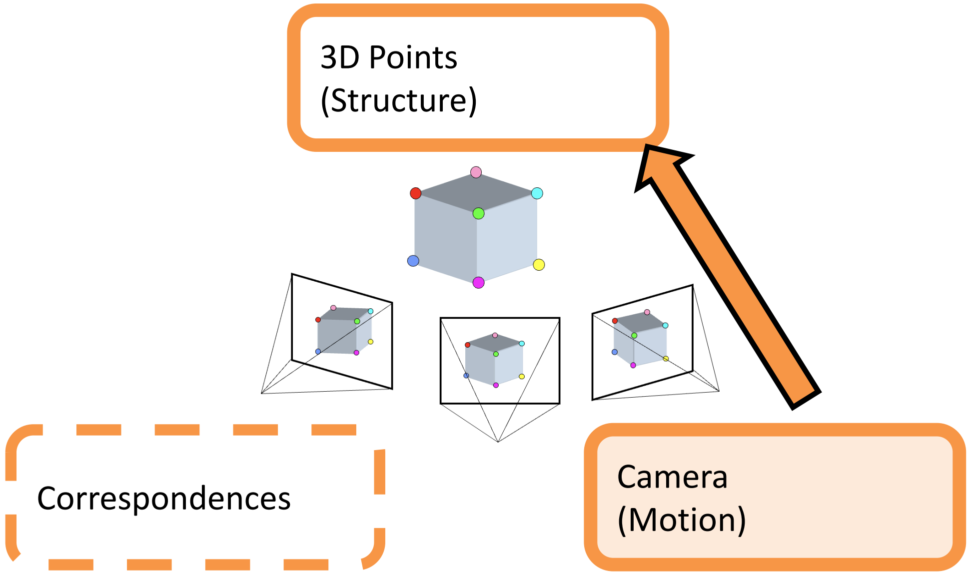
https://github.com/colmap/colmap

A general SfM + MVS pipeline

Multi-View Stereo



Volumetric “Neural” Rendering



Does not use explicit correspondences,
relies on reconstruction loss (Analysis-by-Synthesis)

Neural Radiance Fields

Video from the original ECCV'20 paper