Structure-from-Motion (SfM), Multi-View Stereo (MVS), and the coming 3D learning revolution

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

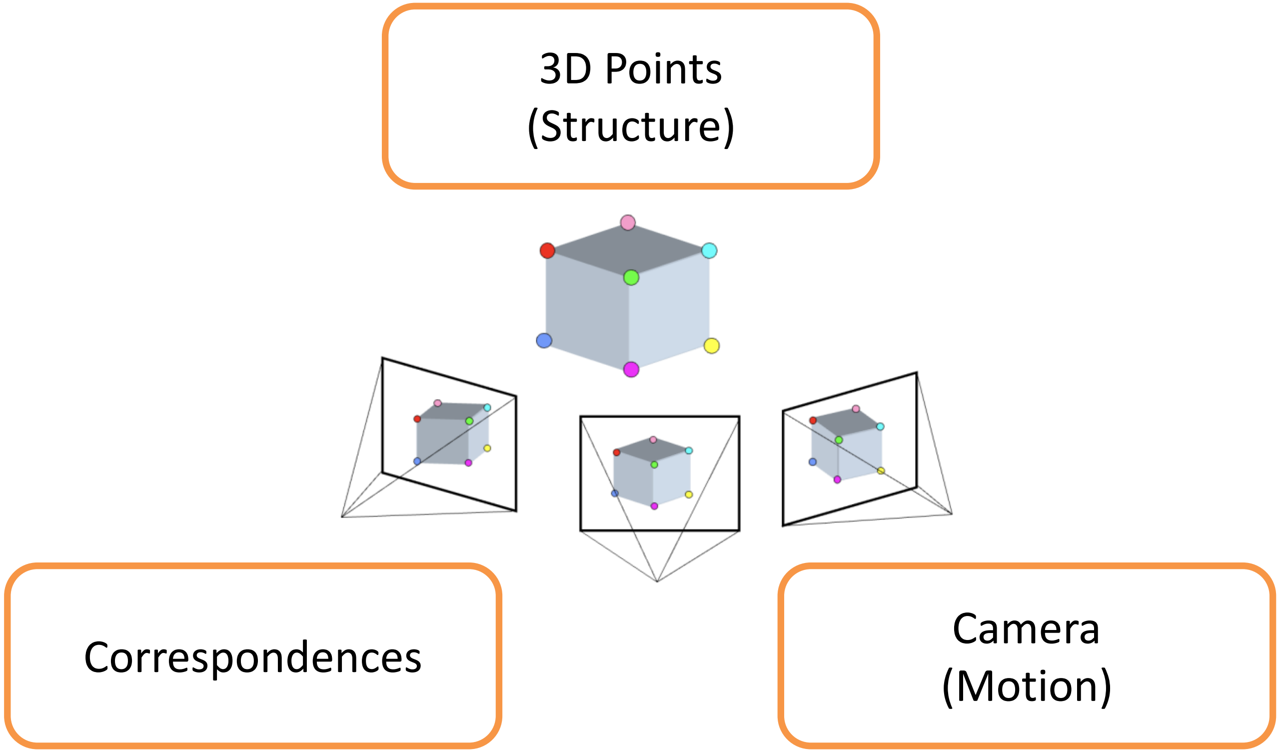
CS280: Computer Vision
Alexei Efros, UC Berkeley, Spring 2024

Recall: Camera calibration & triangulation

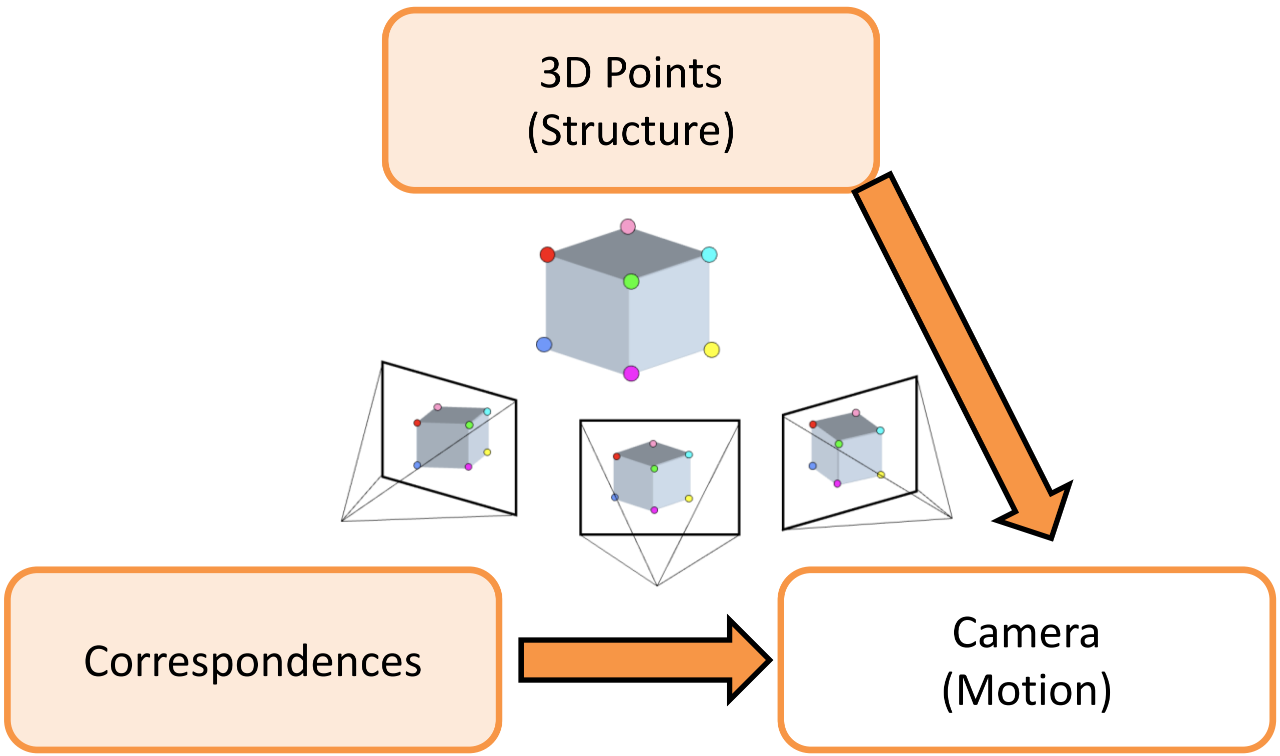
- Suppose we know **3D points** and their **matches** in an image
  - How can we compute the **camera parameters**?

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

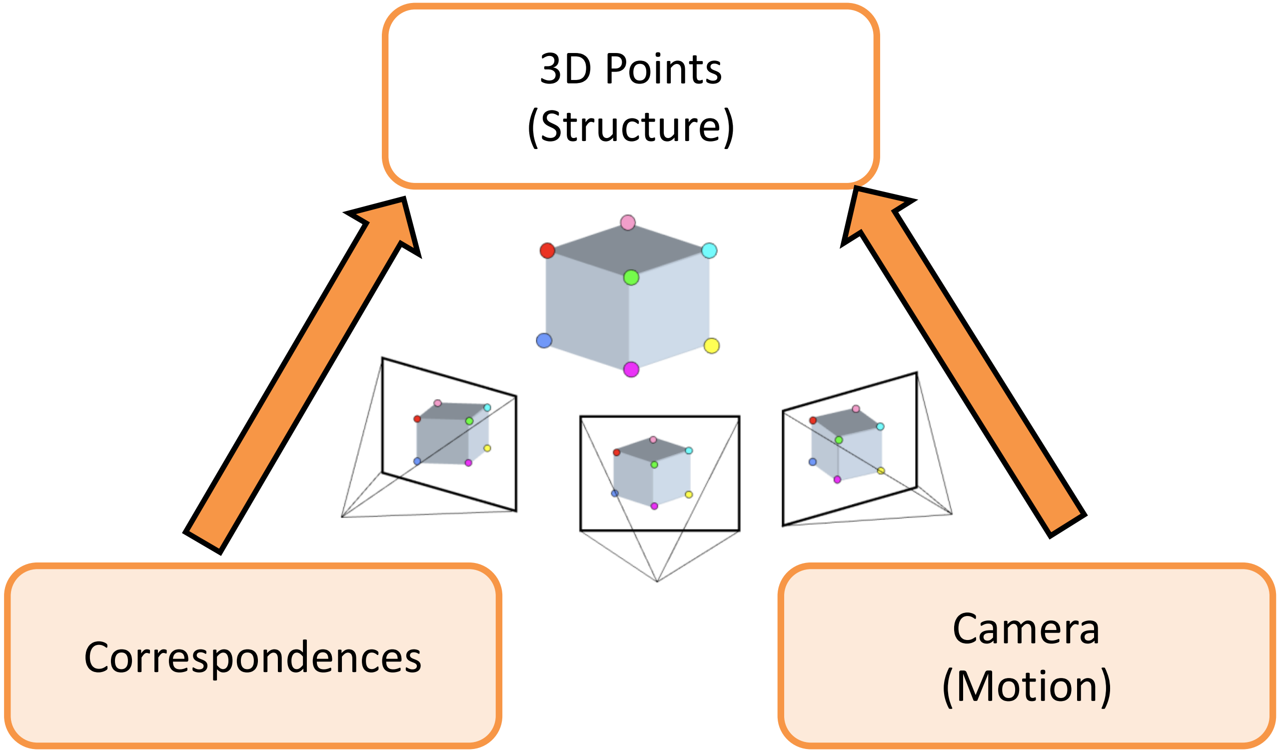
if you know 2 you get the other:



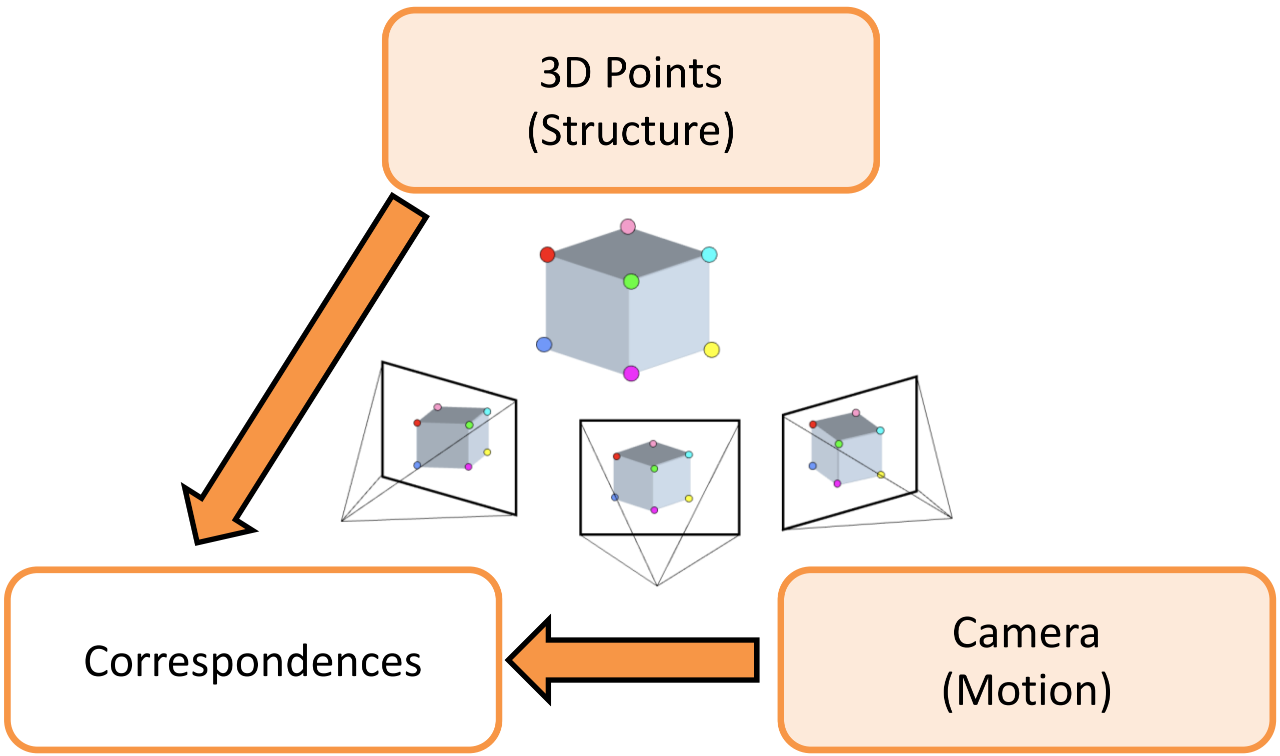
Camera Calibration; aka Perspective-n-Point



Stereo (w/2 cameras); aka Triangulation

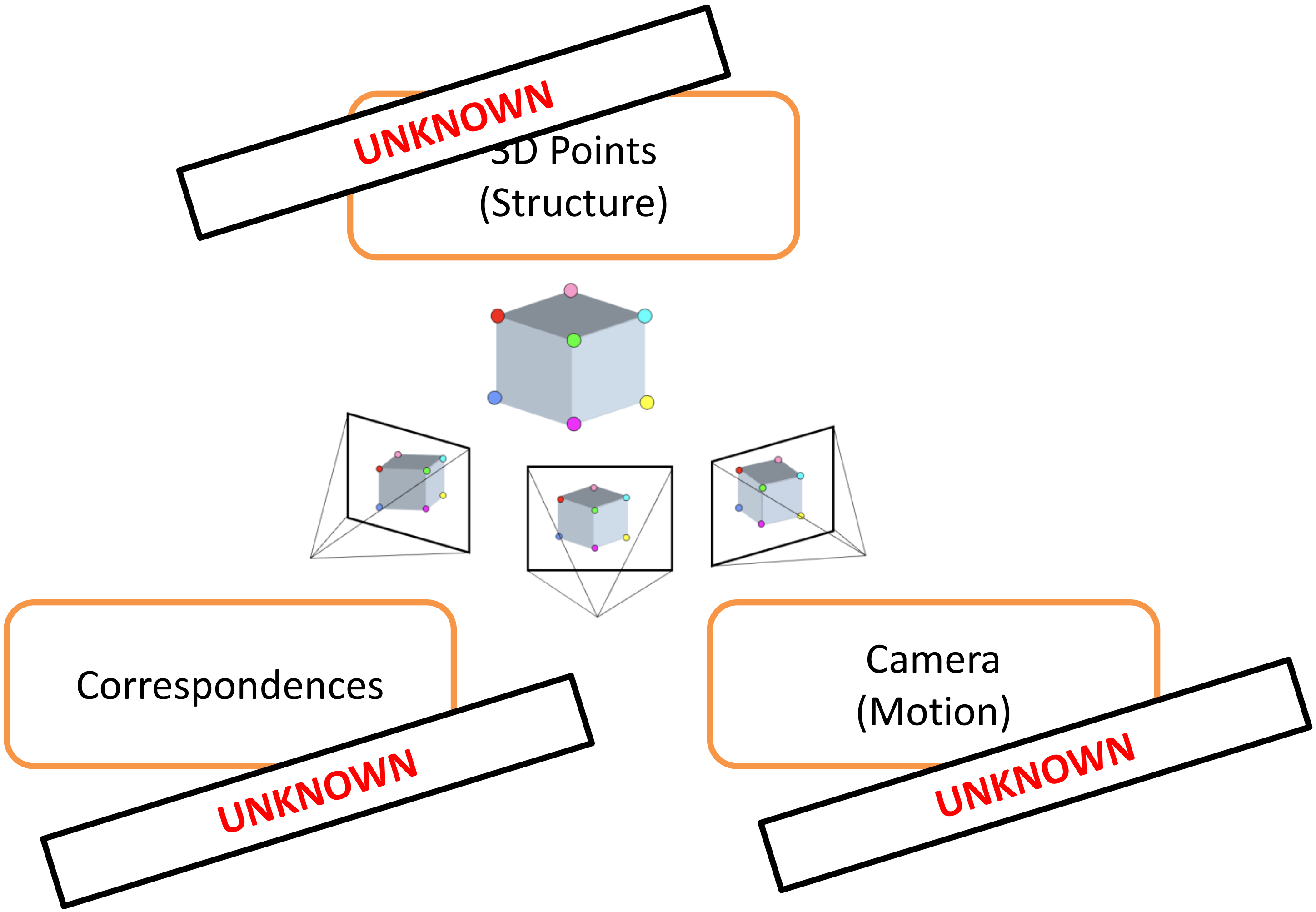


?



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

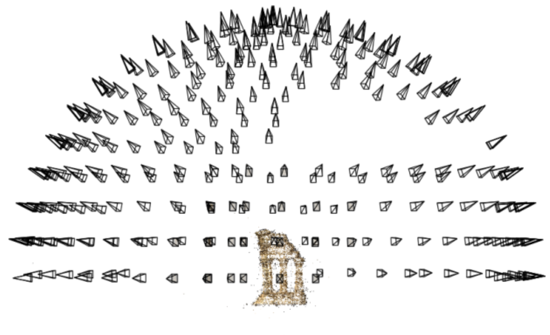
https://youtu.be/mTBPGuPLI5Y

Structure from Motion (SfM)

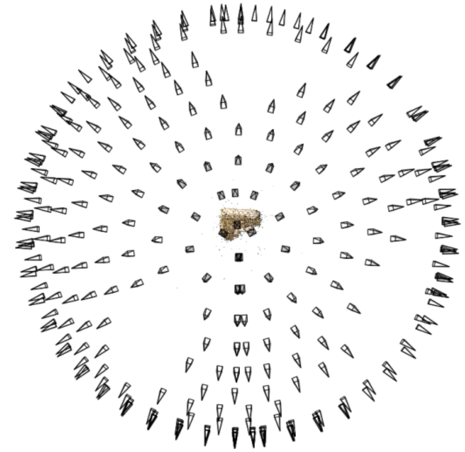
- Given many images, how can we
  a) figure out where they were all taken from?
  b) build a 3D model of the scene?

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 $\mathbf{K}_j$

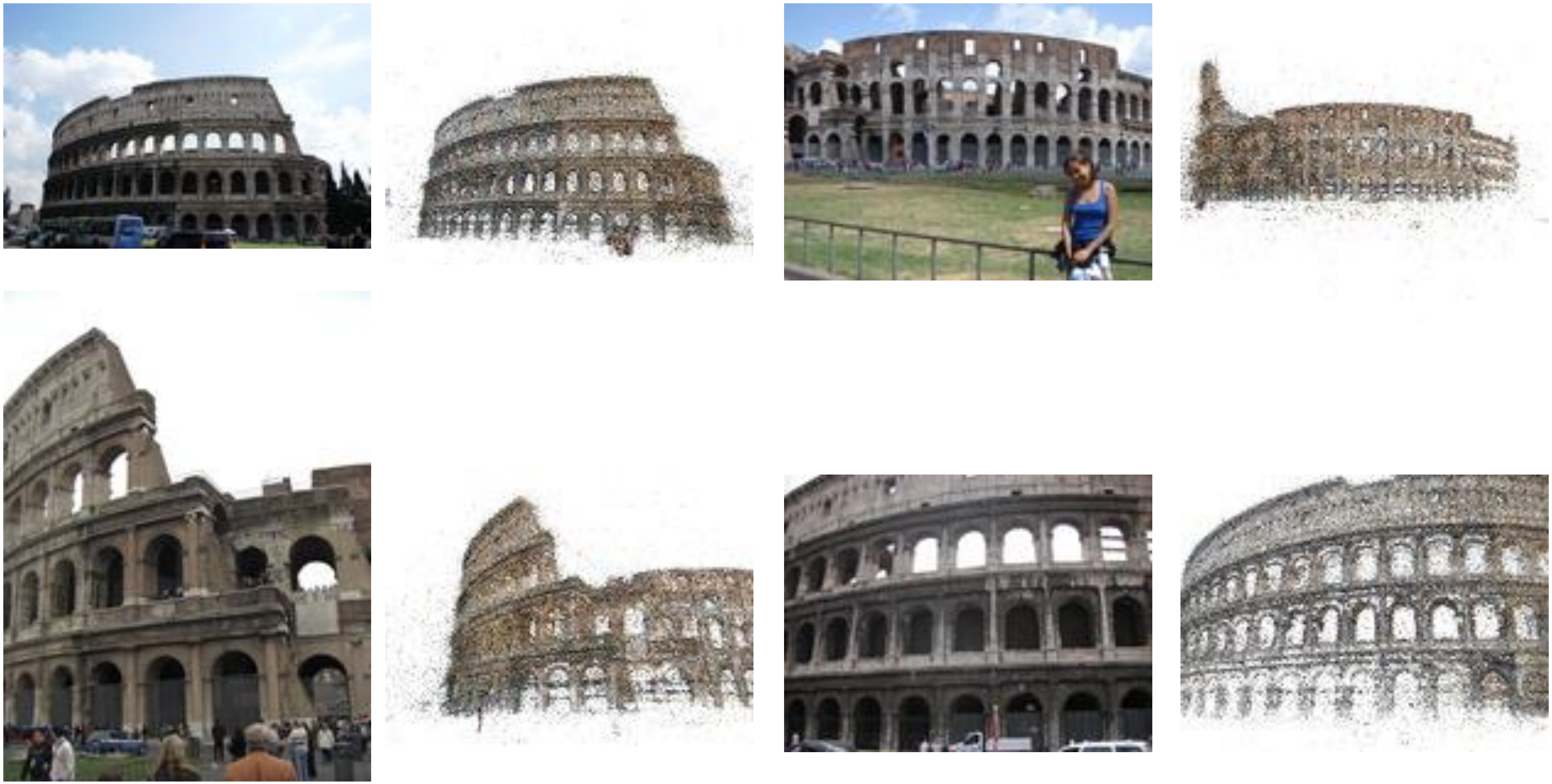
- Objective function: minimize *reprojection error*

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

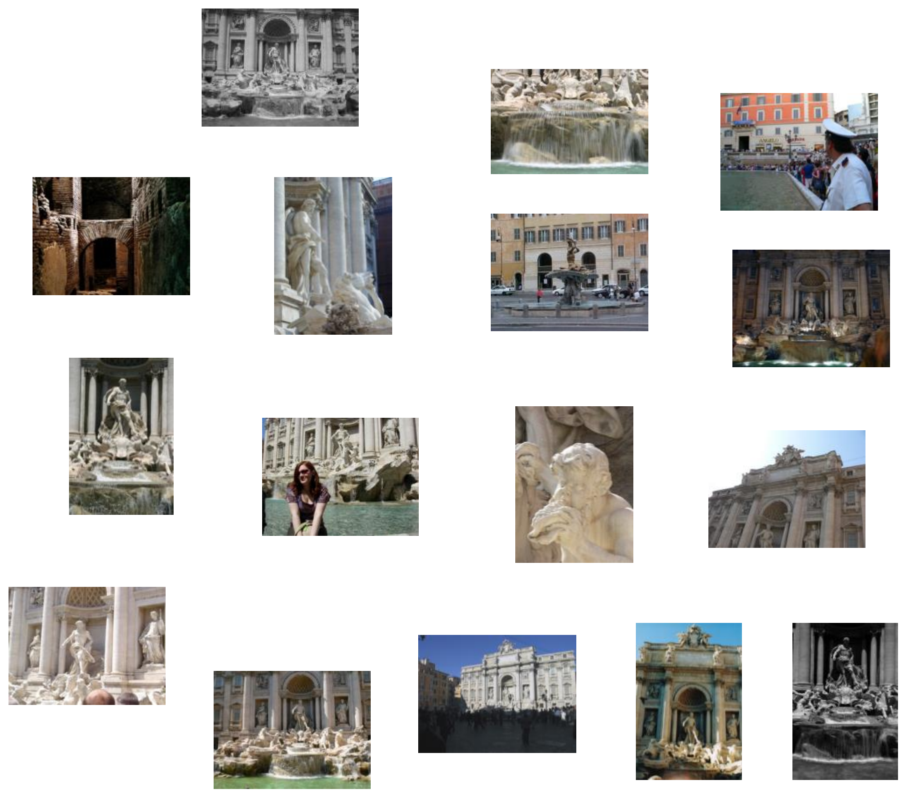
Building Rome in a Day, Agarwal et al. ICCV 2009

First step: Correspondence

- Feature detection and matching

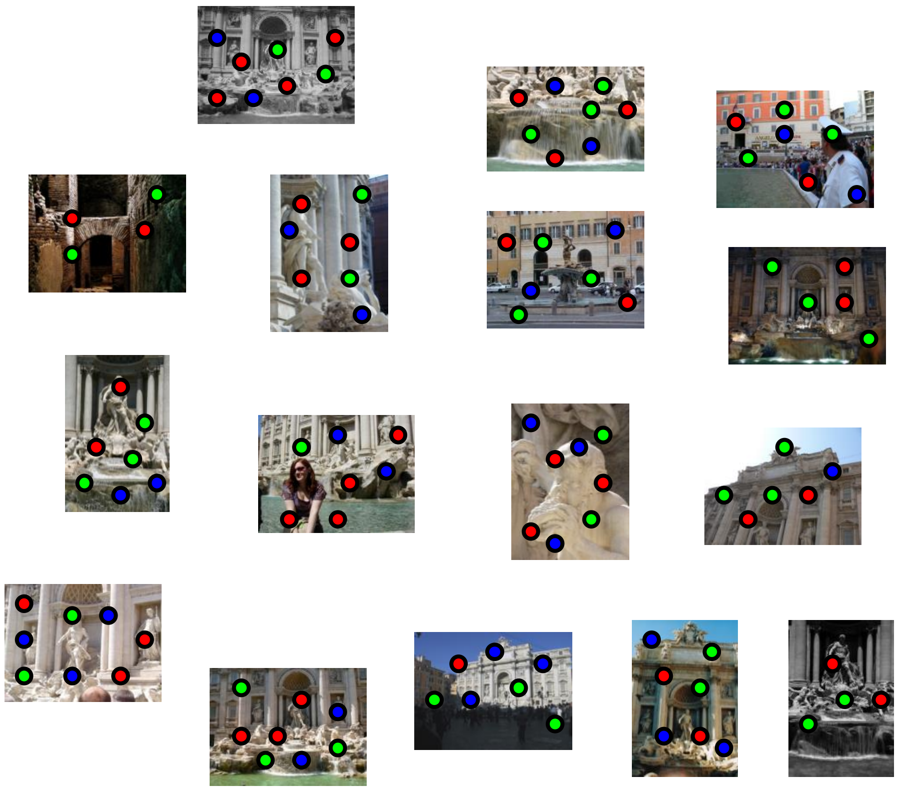
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



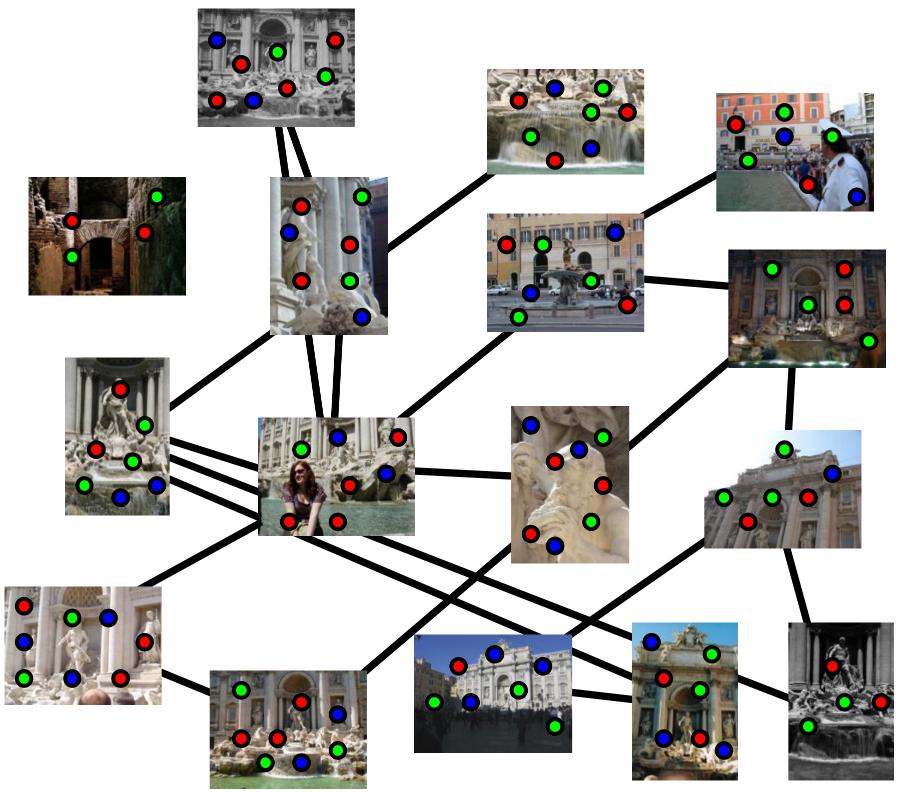
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



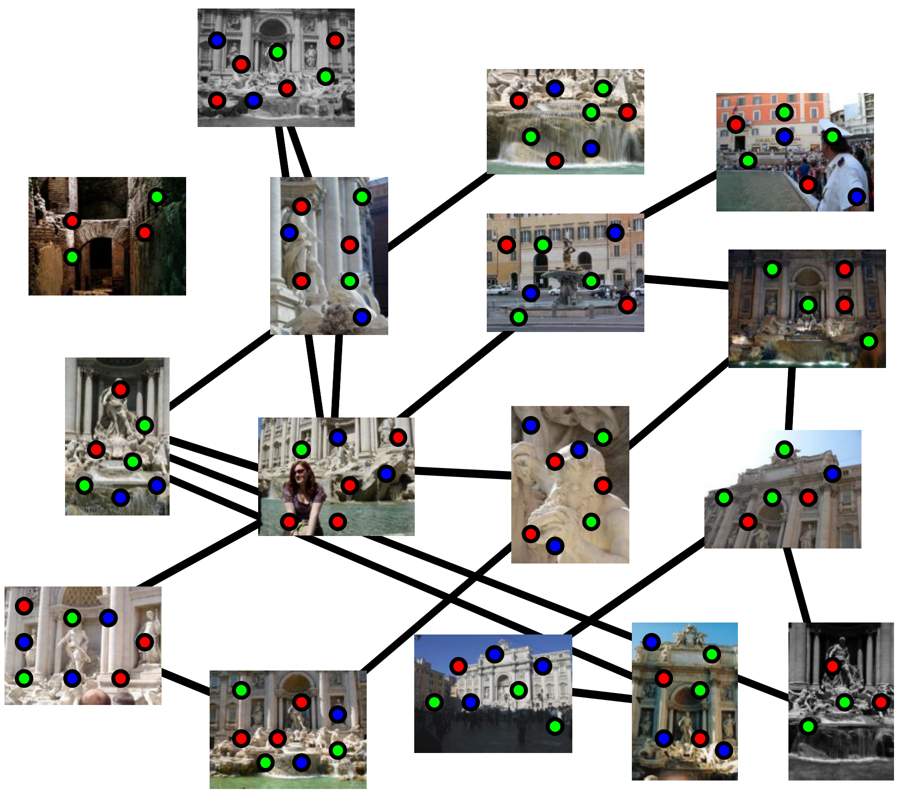
Feature matching

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

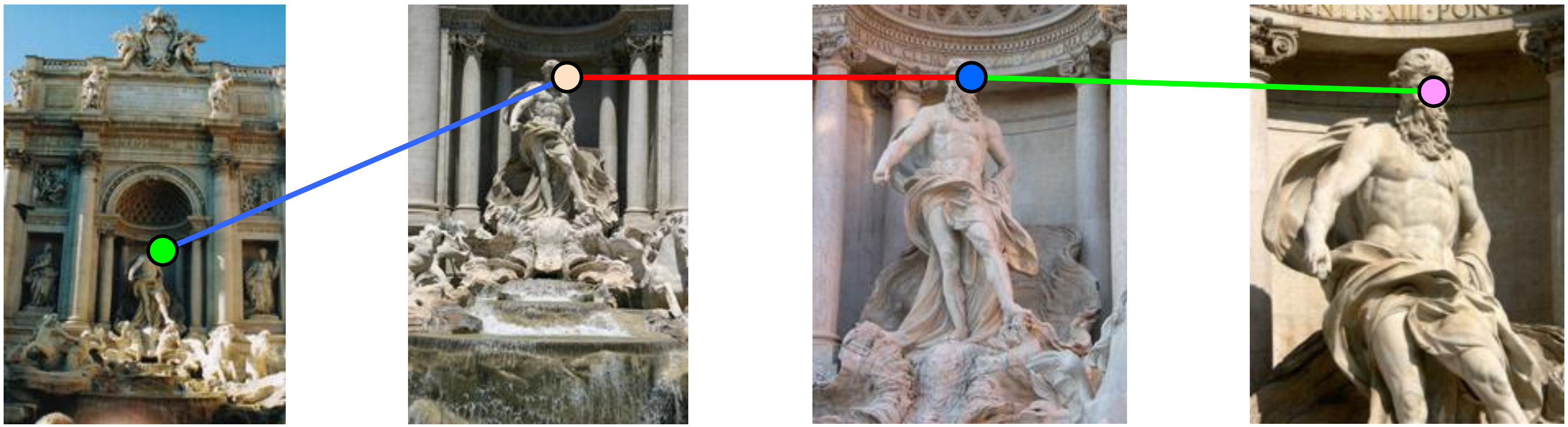
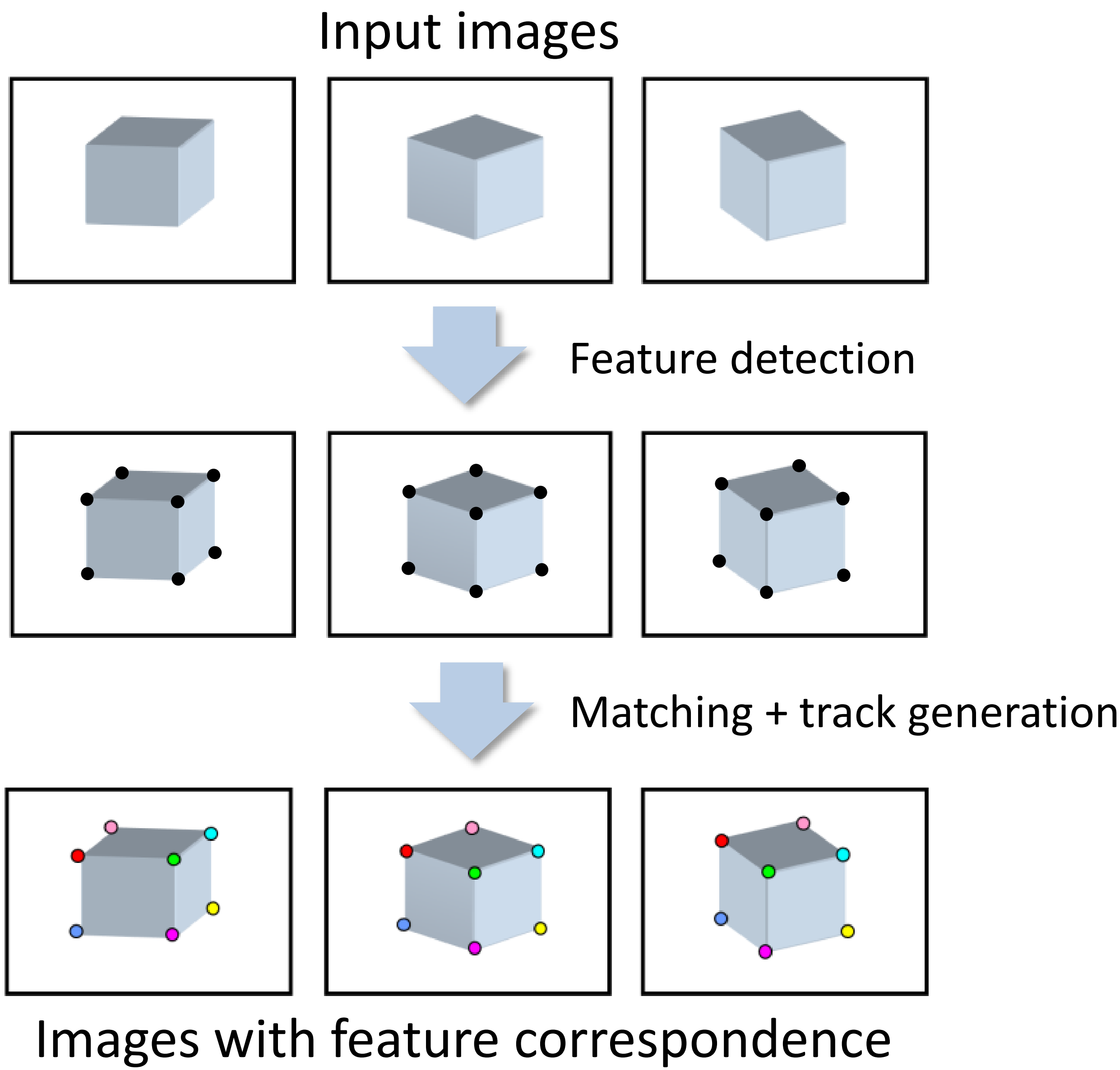
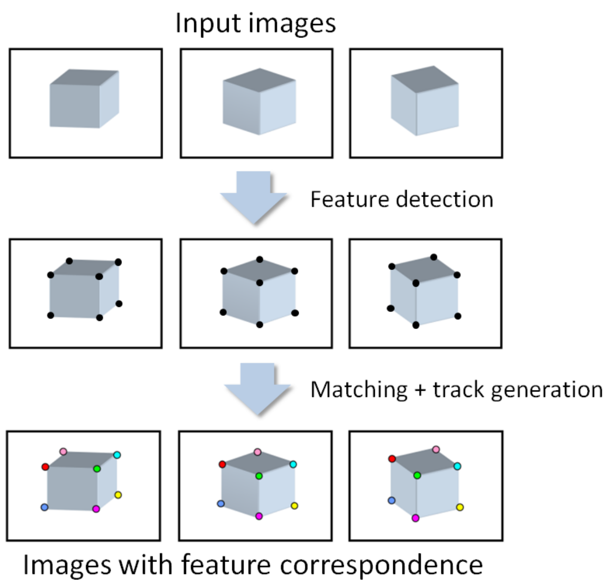


Image 1 Image 2 Image 3 Image 4

The story so far…



The story so far...

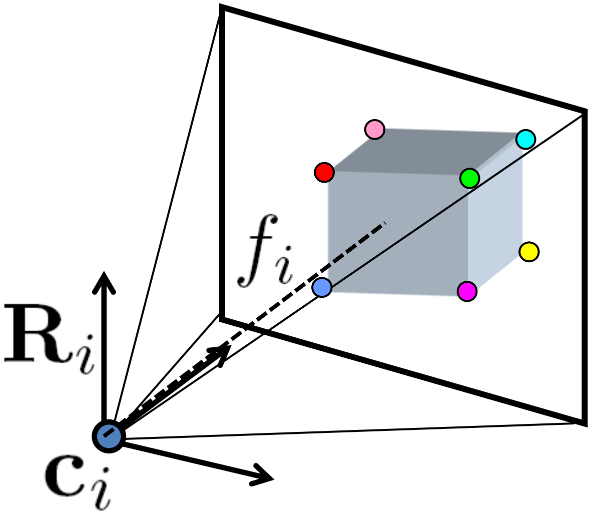


- Next step:
  - Use structure from motion to solve for geometry (cameras and points)
- First: what are cameras and points?

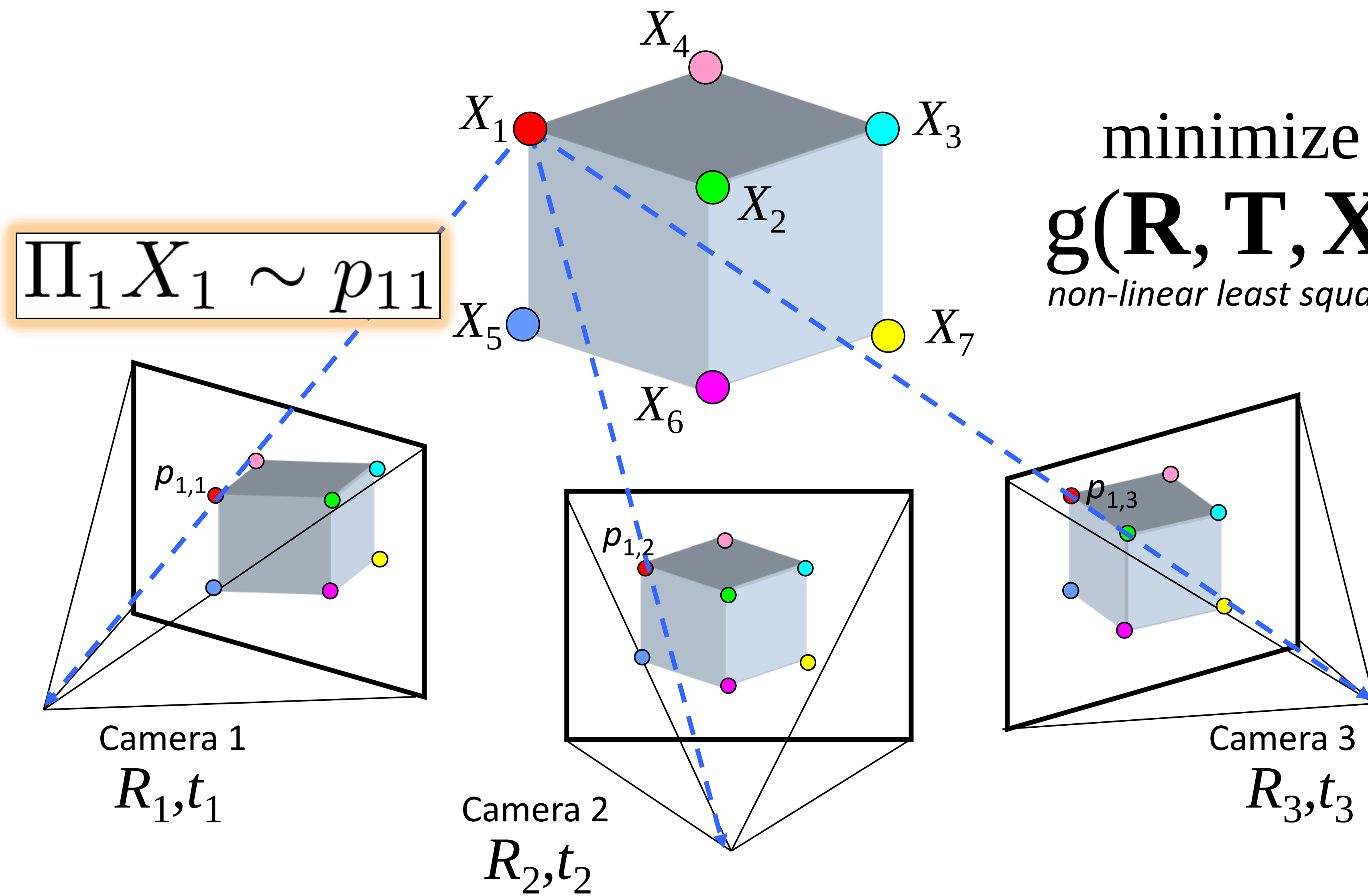
Review: Points and cameras

- 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

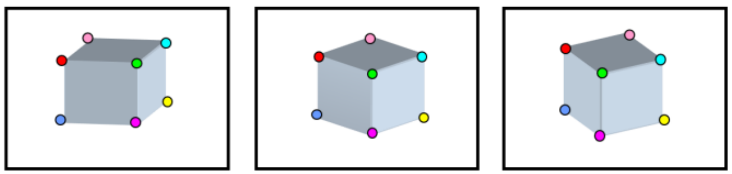
Structure from motion

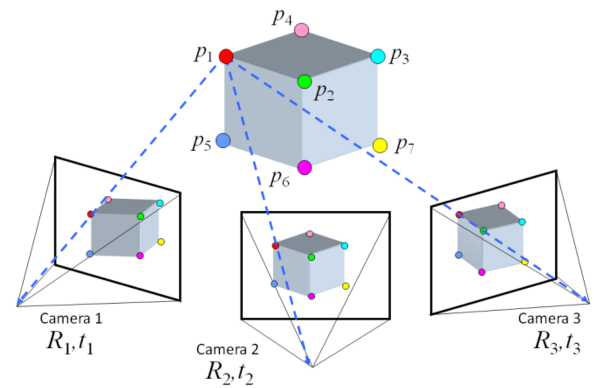
- Minimize sum of squared reprojection errors:

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

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

Solving structure from motion



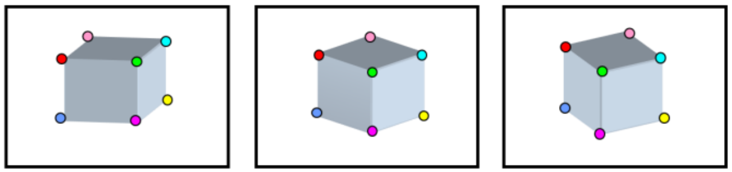


Inputs: feature tracks

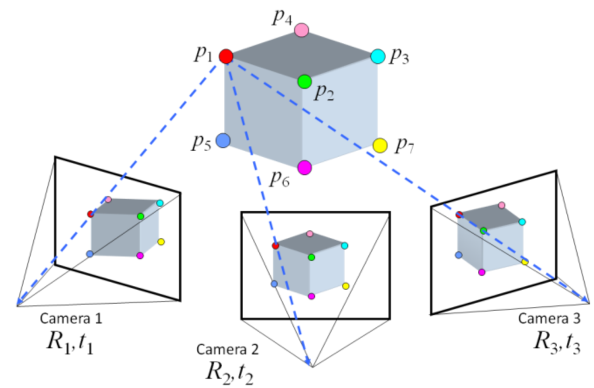
Outputs: 3D cameras and points

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

Solving structure from motion



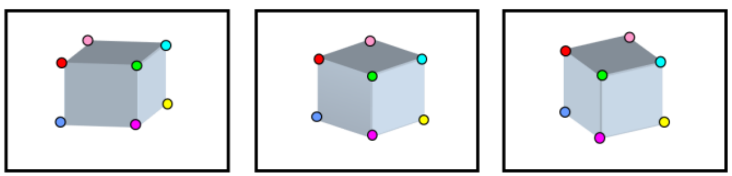
Inputs: feature tracks



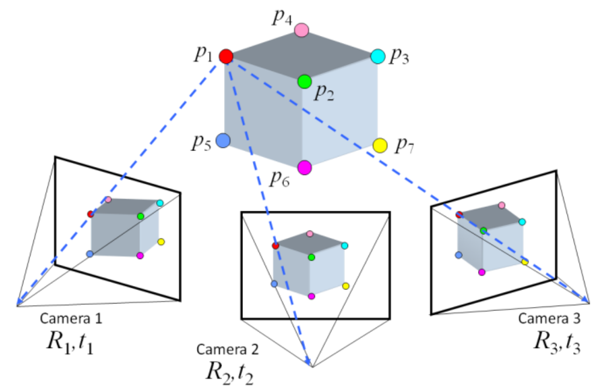
Outputs: 3D cameras and points

- 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



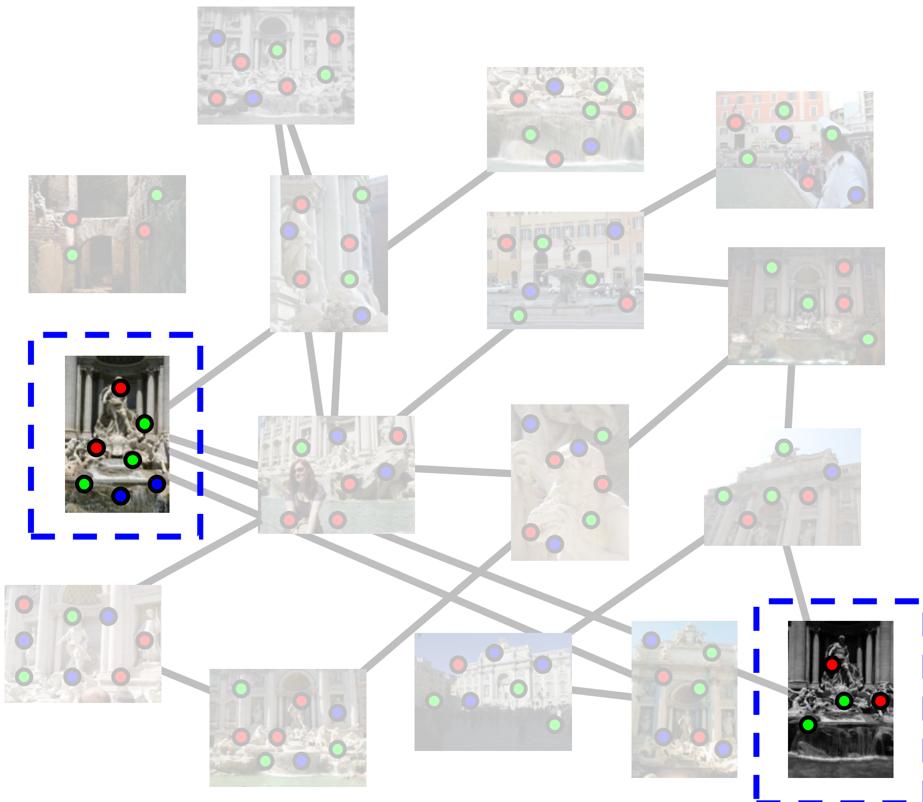
Inputs: feature tracks



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

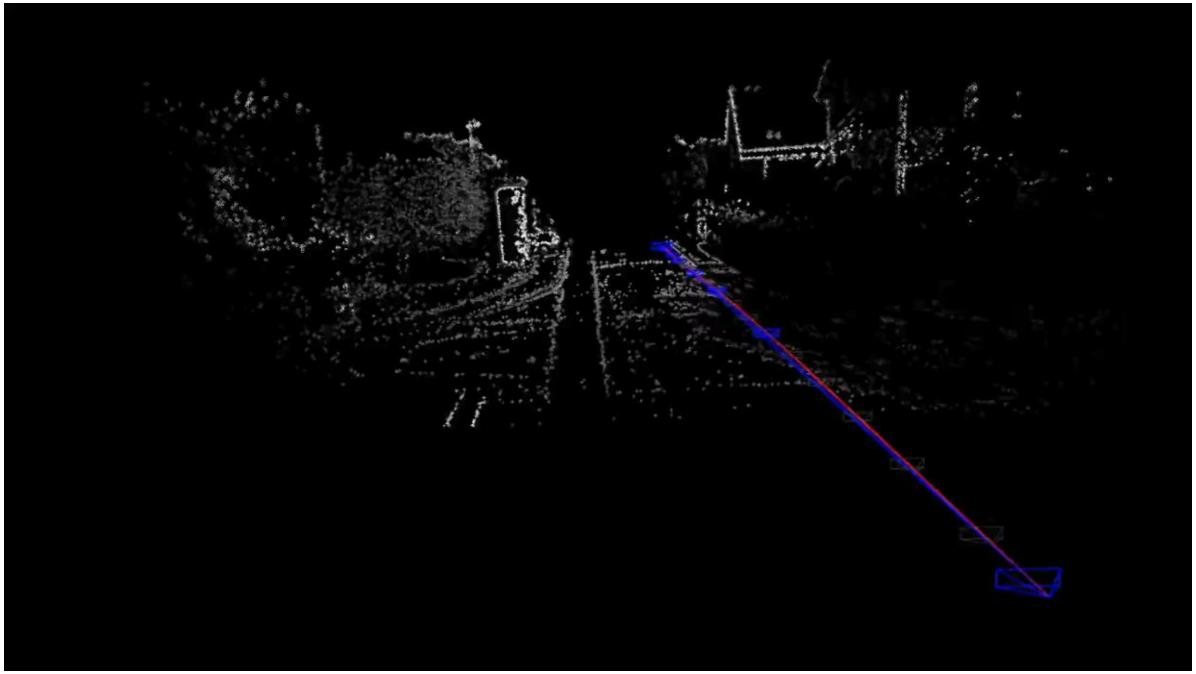
✓ large baseline
✓ lots of matches

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

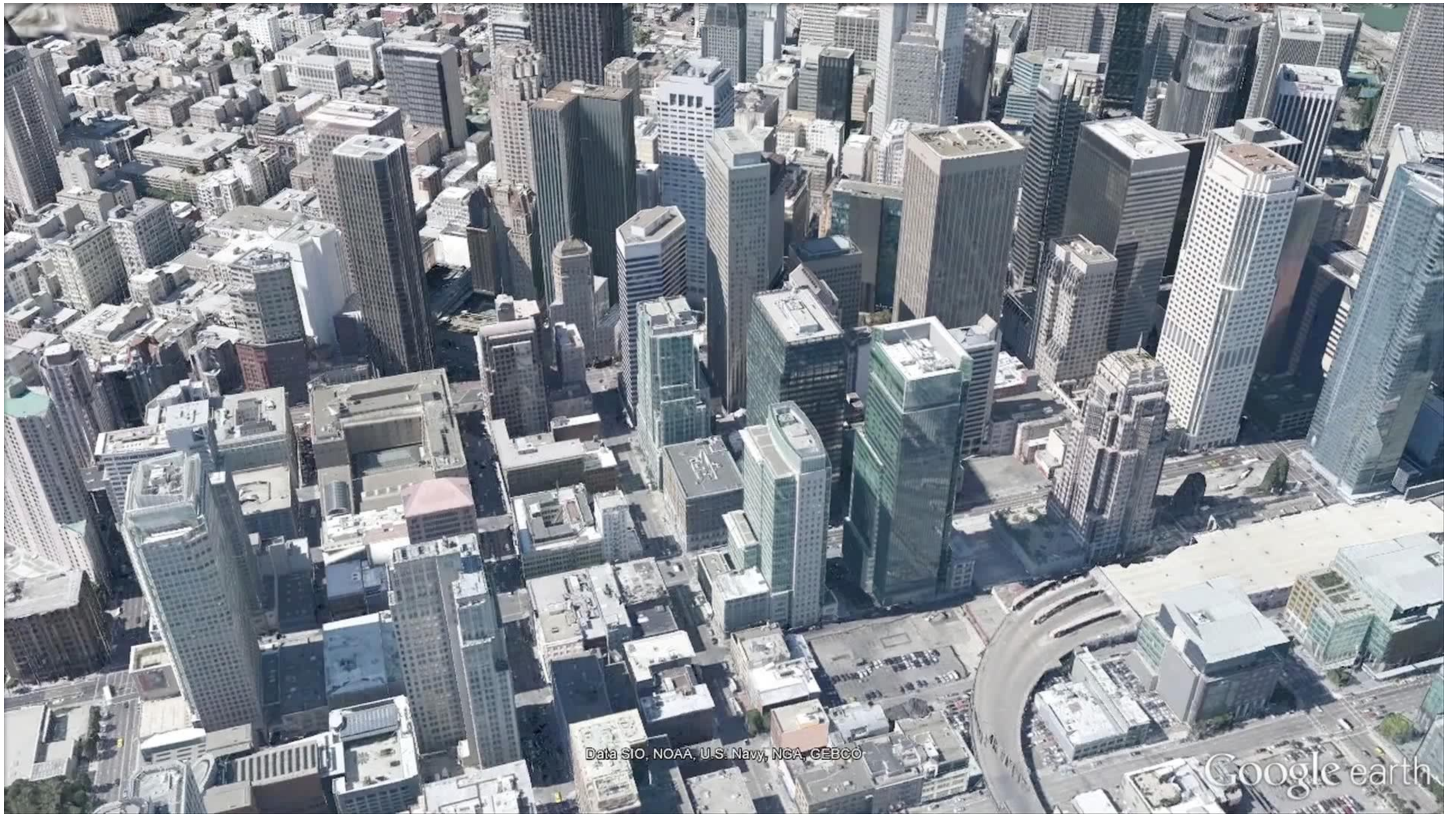
Visual Simultaneous Localization and Mapping (V-SLAM)

- 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

What if we want solid models?

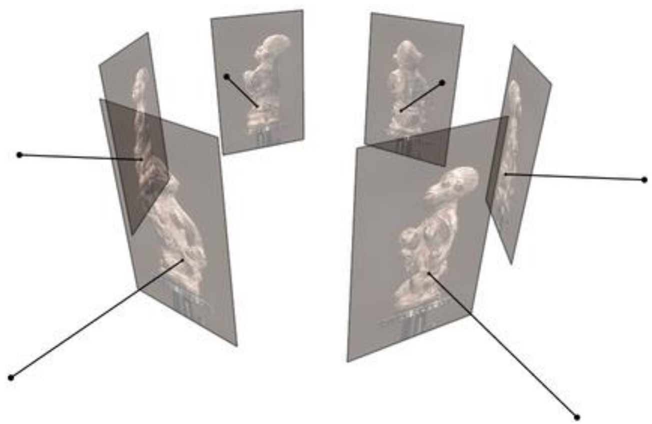


Slide credit: Noah Snavely

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

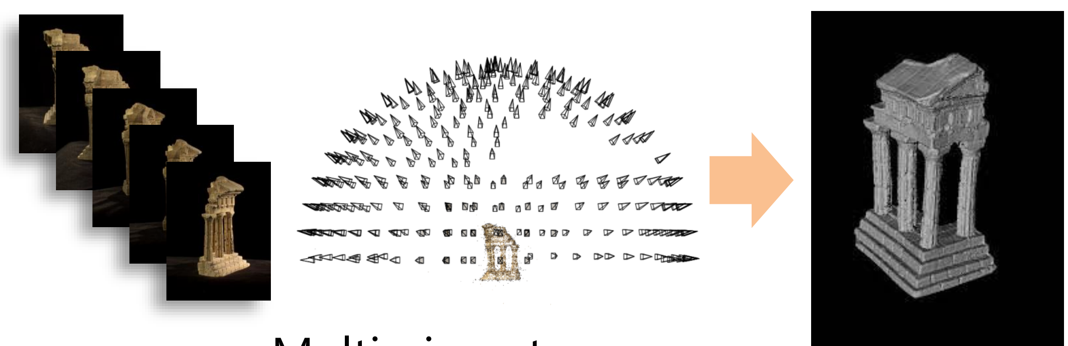
Slide credit: Noah Snavely

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

Multi-view Stereo

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

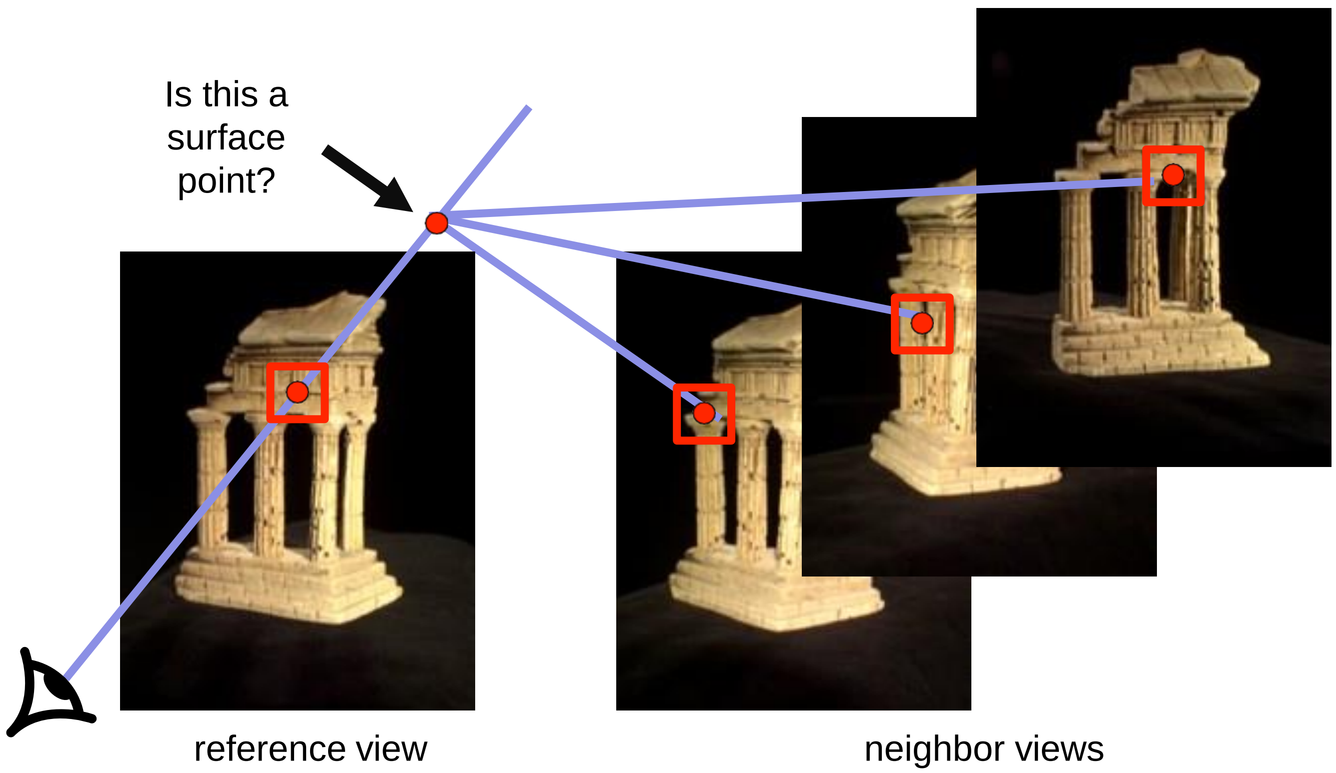
Binocular Stereo



Multi-view stereo

Slide credit: Noah Snavely

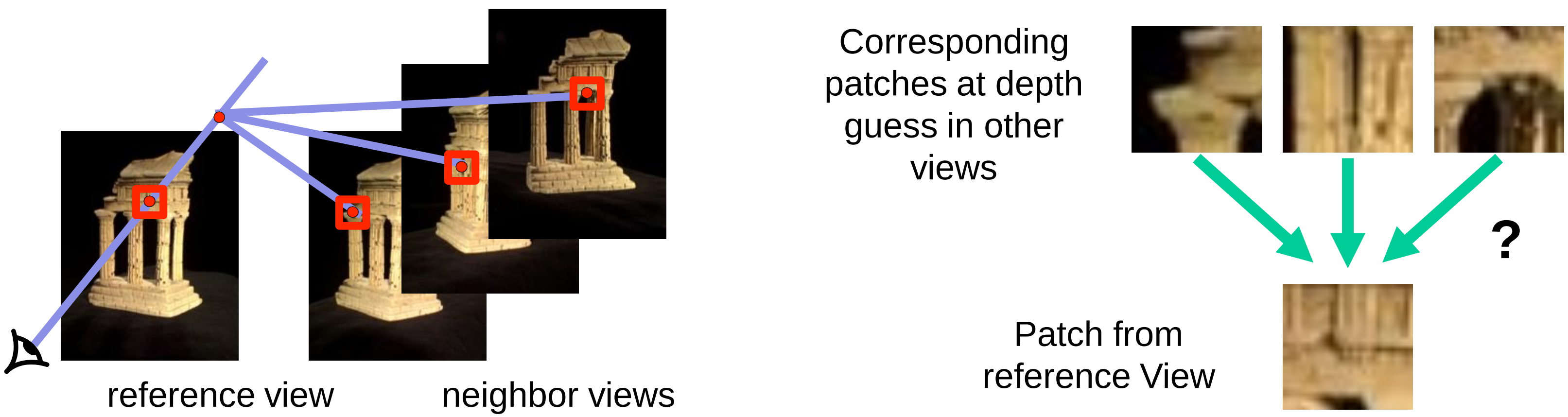
Multi-view stereo: Basic idea



Source: Y. Furukawa

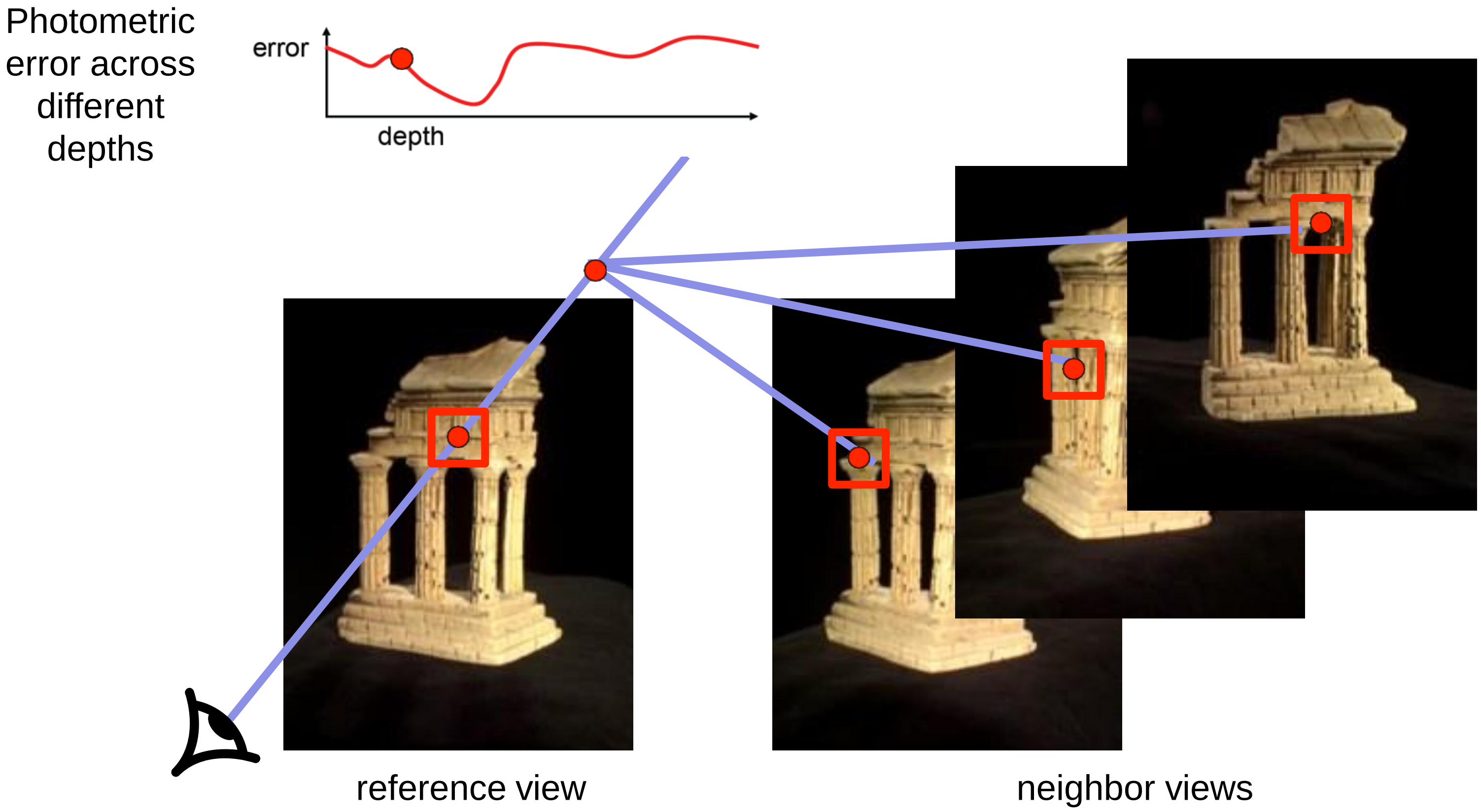
Multi-view stereo: Basic idea

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



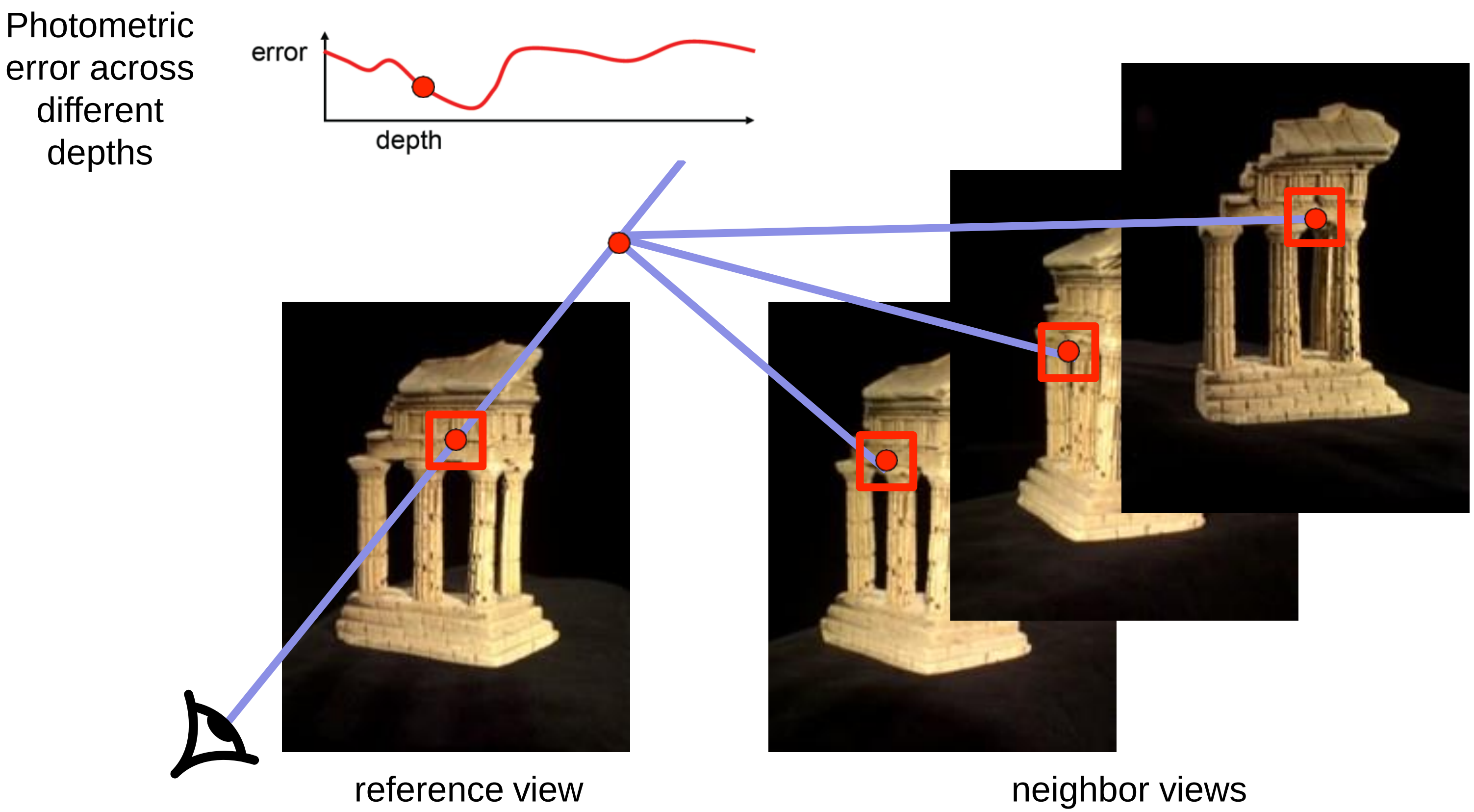
Source: Y. Furukawa

Multi-view stereo: Basic idea



Source: Y. Furukawa

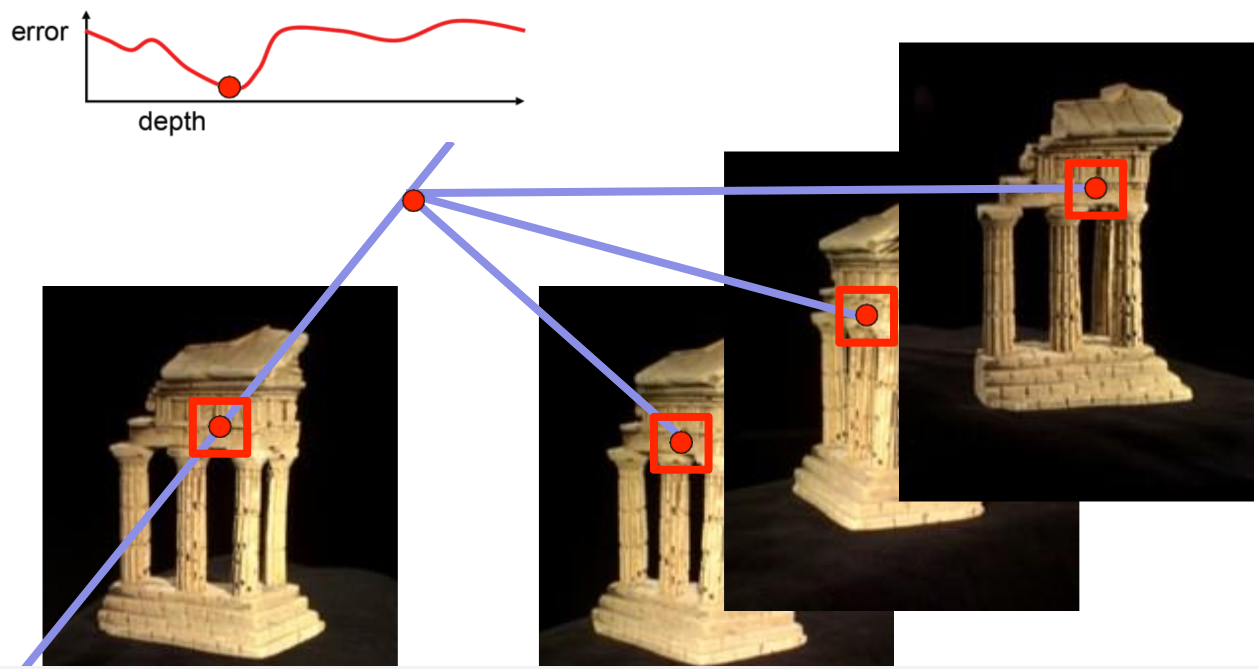
Multi-view stereo: Basic idea



Source: Y. Furukawa

Multi-view stereo: Basic idea

Photometric error across different depths



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

Multi-view stereo: advantages

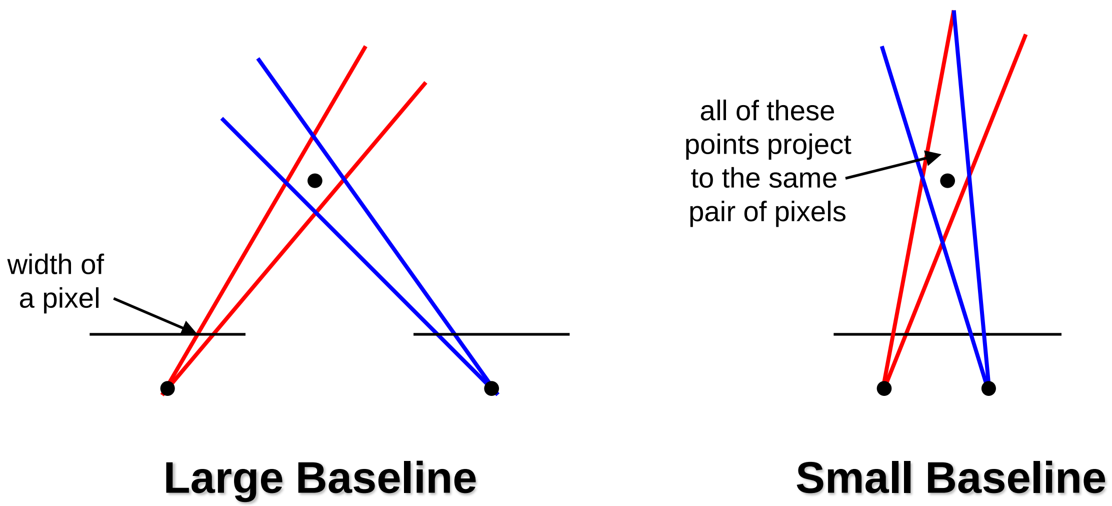
Can match windows using more than 1 other image, giving a **stronger match signal**

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

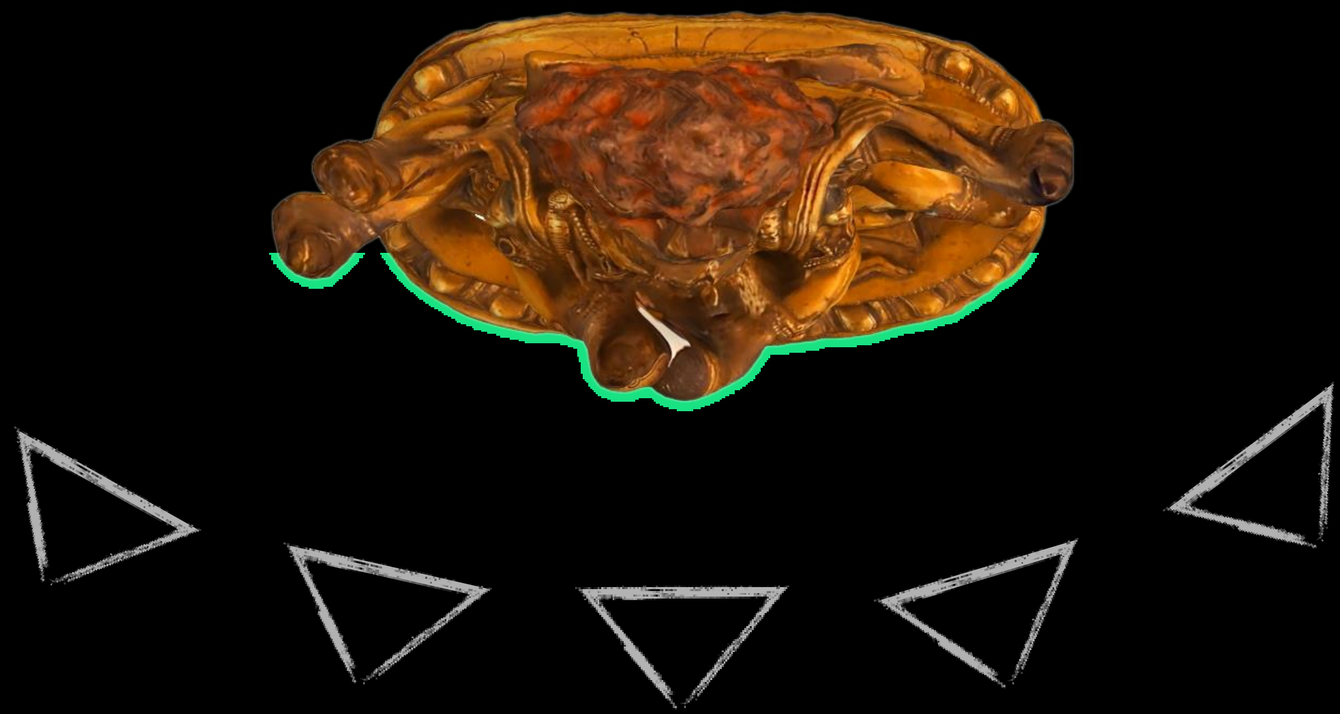


What's the optimal baseline?

- Too small: large depth error
- Too large: difficult search problem

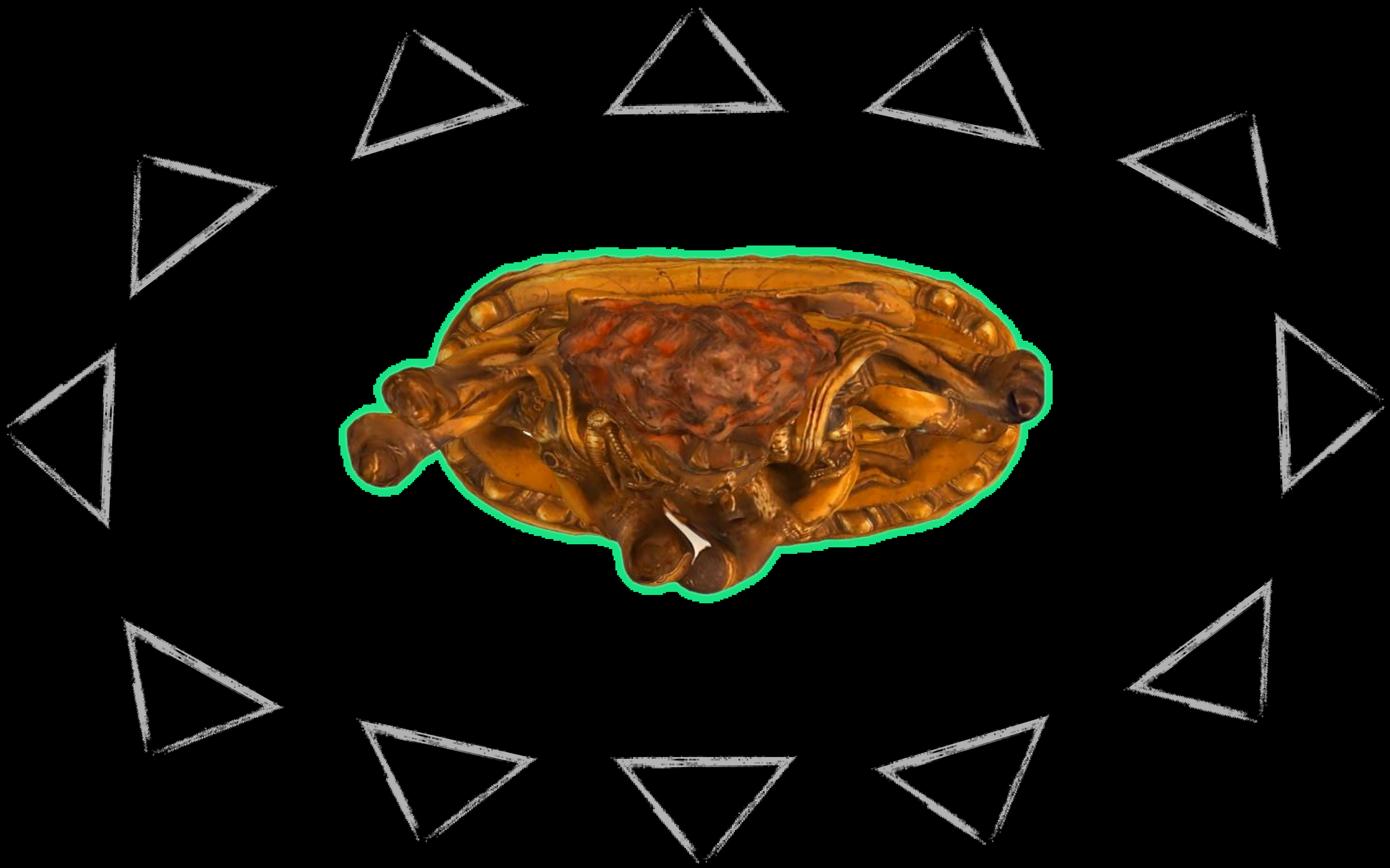
Slide credit: Noah Snavely

Single depth map often isn't enough



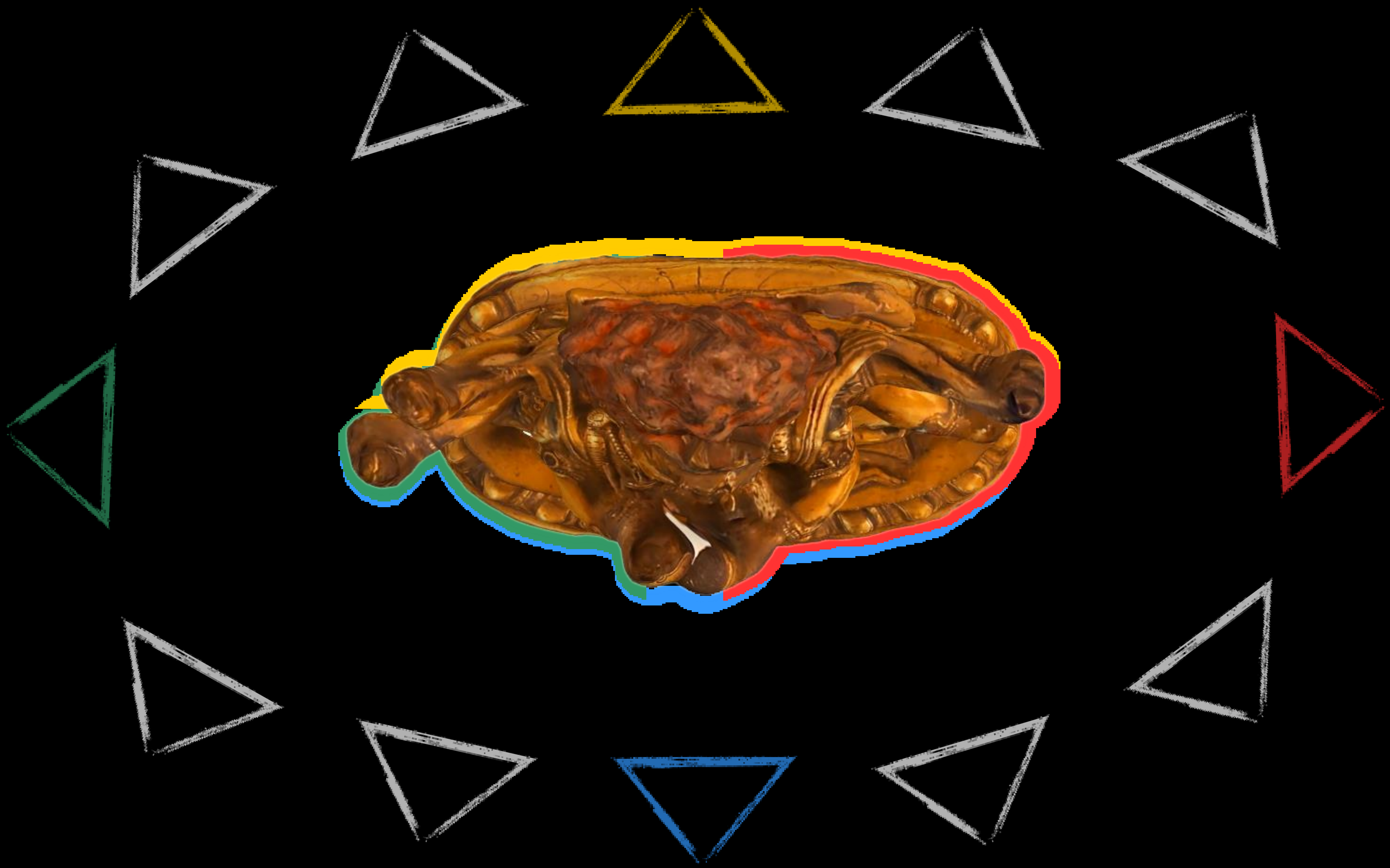
Source: N. Snavely

Really want full coverage



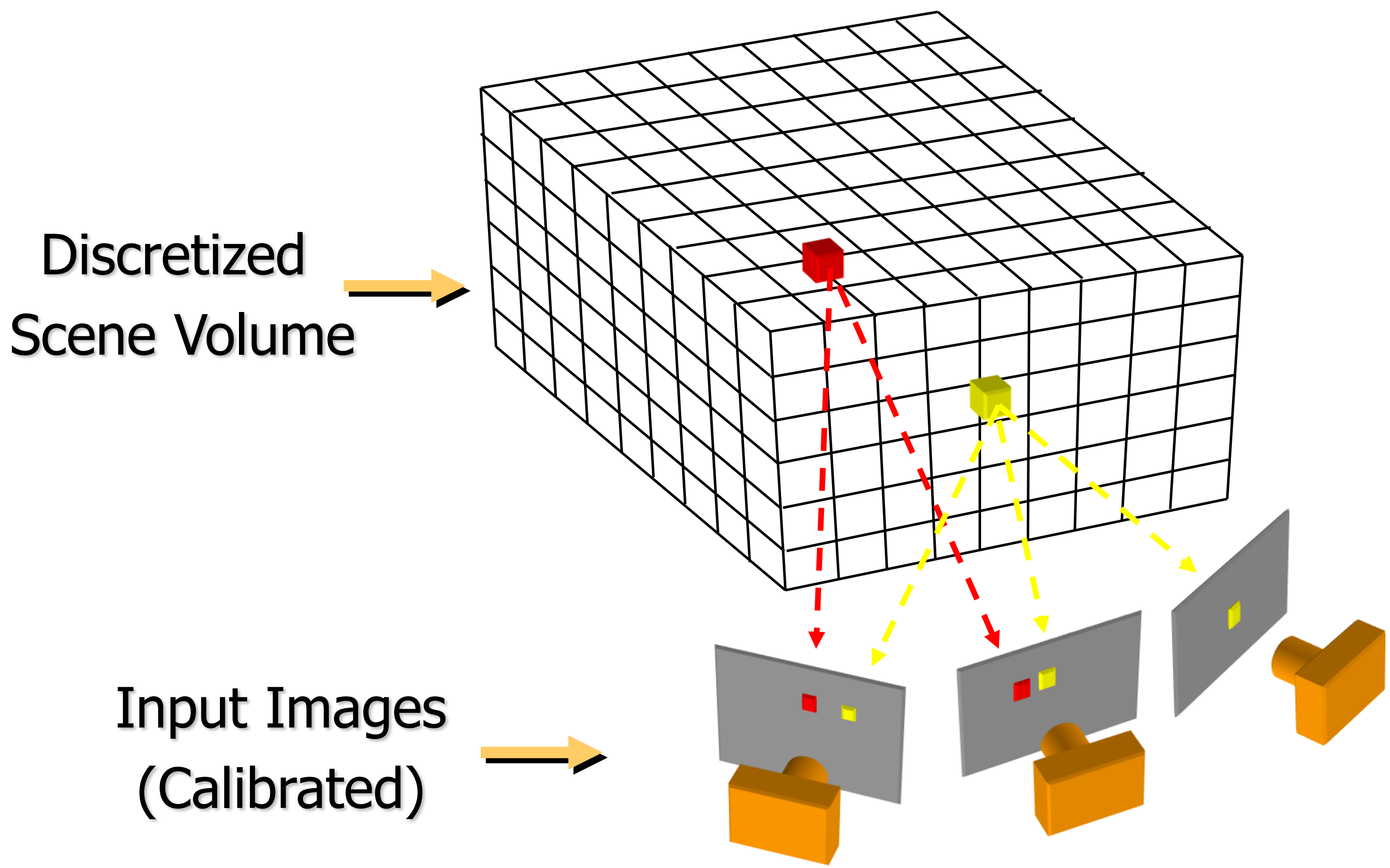
Source: N. Snavely

Idea: Combine many depth maps



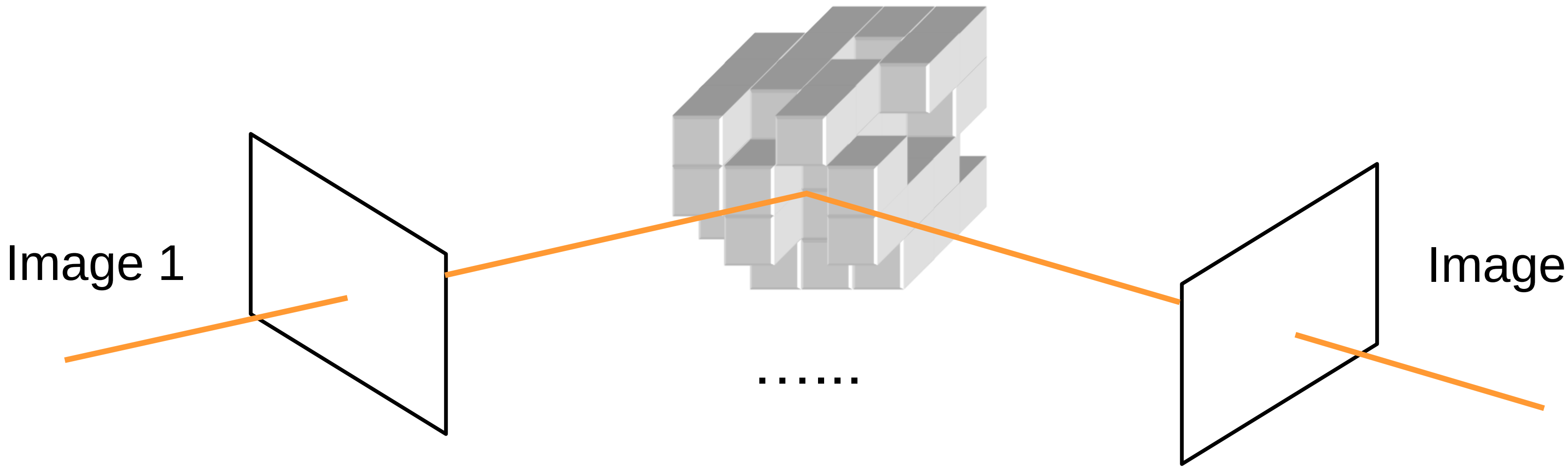
Source: N. Snavely

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

Space Carving Results

Input Image
(1 of 100)

Reconstruction

Source: S. Seitz

Tool for you: COLMAP

https://github.com/colmap/colmap

A general SfM + MVS pipeline

The Deep Learning Lesson, revisited

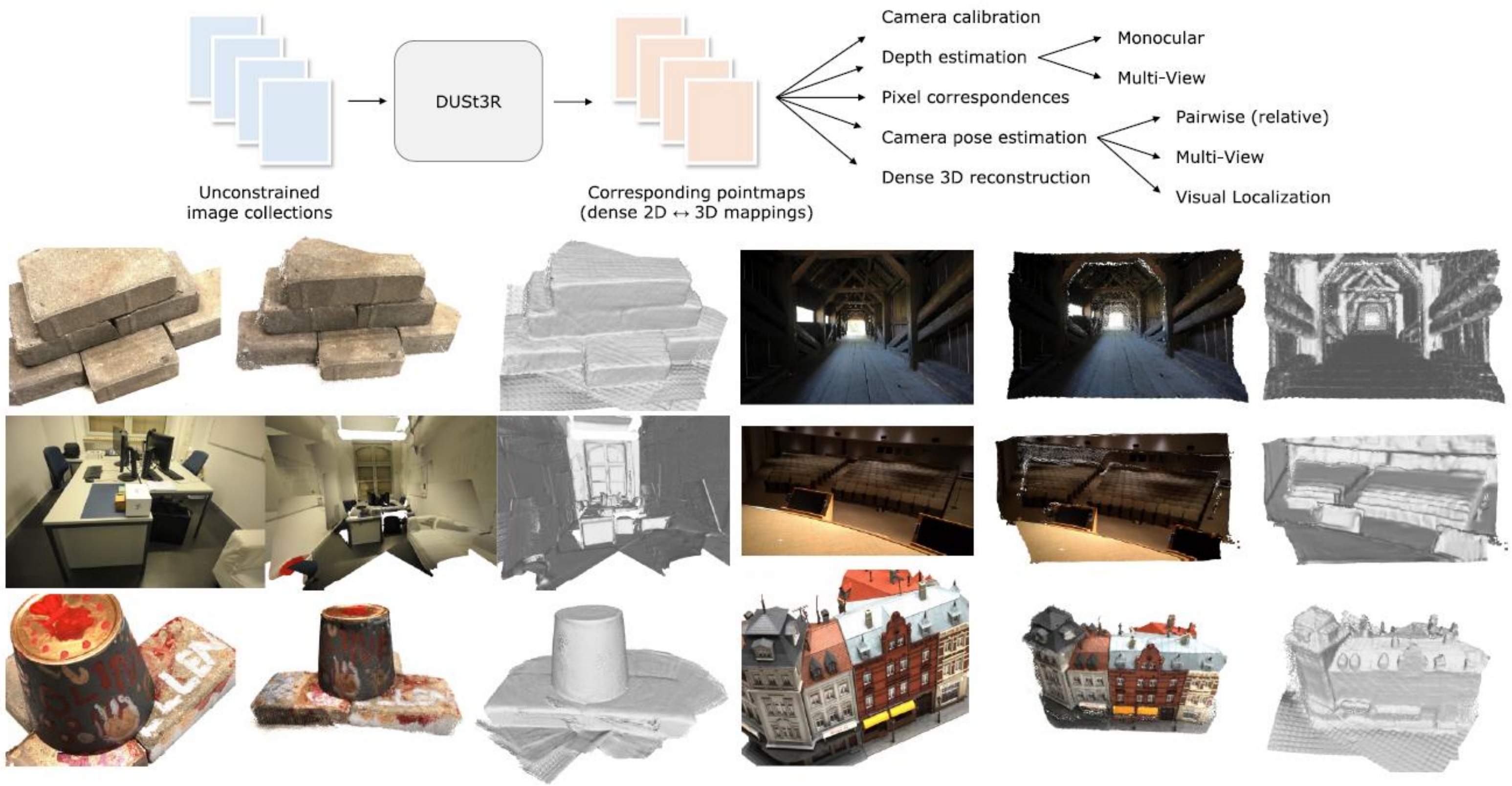
- Old-school Recognition Pipeline:
  1. Detect Features
  2. Find Regions
  3. Segmentation
  4. Recognition
- New Recognition:
  - End-to-End Learning!

- Old-school 3D Pipeline:
  1. Detect Features
  2. Calibrate Cameras
  3. Run Structure-from-Motion
  4. Run Multi-View Stereo
- New 3D?!

New 3D Vision revolution coming…

DUSt3R: Geometric 3D Vision Made Easy

Shuzhe Wang*, Vincent Leroy†, Yohann Cabon†, Boris Chidlovskii† and Jerome Revaud†
*Aalto University †Naver Labs Europe
shuzhe.wang@aalto.fi firstname.lastname@naverlabs.com



CVPR 2024

Forget (almost) everything you learned!

- No Pipelines
- No Camera Projection
- No Reprojection Error
- No Explicit Triangulation
- No Epipolar constraints
- Etc.

DUSt3R: Dense Uncalibrated Stereo 3D Reconstruction

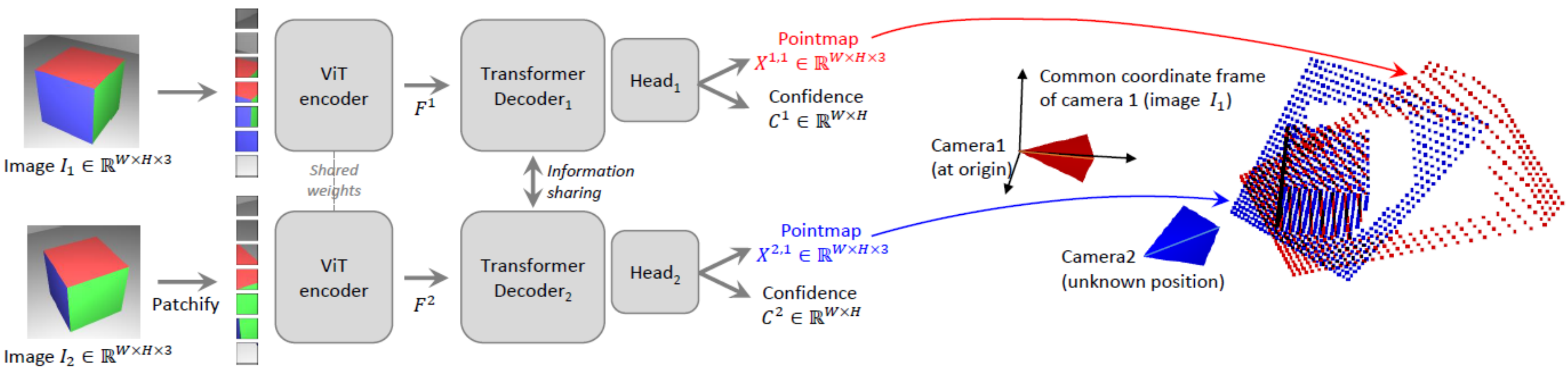


Figure 2. **Architecture of the network $\mathcal{F}$.** Two views of a scene (I^1, I^2) are first encoded in a Siamese manner with a shared ViT encoder. The resulting token representations F^1 and F^2 are then passed to two transformer decoders that constantly exchange information via cross-attention. Finally, two regression heads output the two corresponding pointmaps and associated confidence maps. Importantly, the two pointmaps are expressed in the same coordinate frame of the first image I^1. The network $\mathcal{F}$ is trained using a simple regression loss (Eq. (4))

- End-to-end Stereo Reconstruction
- Operates on “3D pointmaps”
- Very simple supervised learning setup:
  - Get lots of 3D datasets
  - Generate 3D pointmaps for image pairs
  - Loss is just L2 on 3D pointmaps

What can you do with this?

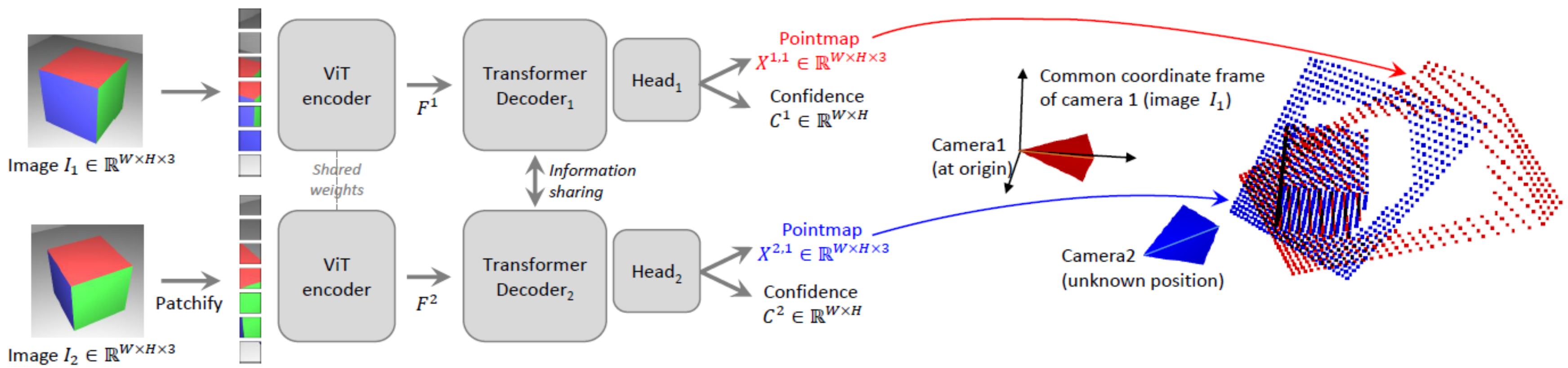


Figure 2. **Architecture of the network $\mathcal{F}$.** Two views of a scene (I^1, I^2) are first encoded in a Siamese manner with a shared ViT encoder. The resulting token representations F^1 and F^2 are then passed to two transformer decoders that constantly exchange information via cross-attention. Finally, two regression heads output the two corresponding pointmaps and associated confidence maps. Importantly, the two pointmaps are expressed in the same coordinate frame of the first image I^1. The network $\mathcal{F}$ is trained using a simple regression loss (Eq. (4))

- Point Matching:
  - Simple Nearest Neighbors between 3D points
- Recovering Intrinsics:
  - Simple optimization between 2D points and corresponding 3D points for each camera
- Relative Pose Estimation:
  - Through the Essential Matrix
  - Or directly with Procrustes algorithm

2-view to Multi-view

Figure 3. **Reconstruction examples** on two scenes never seen during training. From left to right: RGB, depth map, confidence map, reconstruction. The left scene shows the raw result output from $\mathcal{F}(I^1, I^2)$. The right scene shows the outcome of global alignment (Sec. 3.4).

- Multiview Global Alignment:
  - Setup pairwise graph (like in SfM)
  - Optimize for consistency

Results

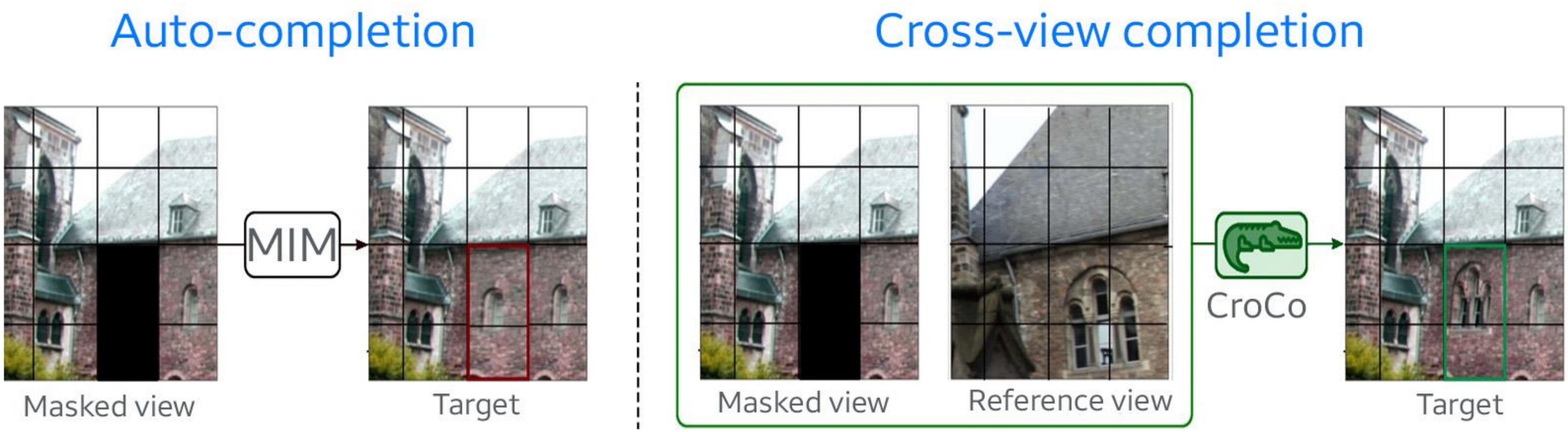
DUSt3R:
Geometric 3D Vision Made Easy

S. Wang [1], V. Leroy [2], Y. Cabon [2], B. Chidlovskii [2] and J. Revaud [2]

[1] Aalto University

2 Naver Labs Europe

CroCo: MAE for Cross-view Learning



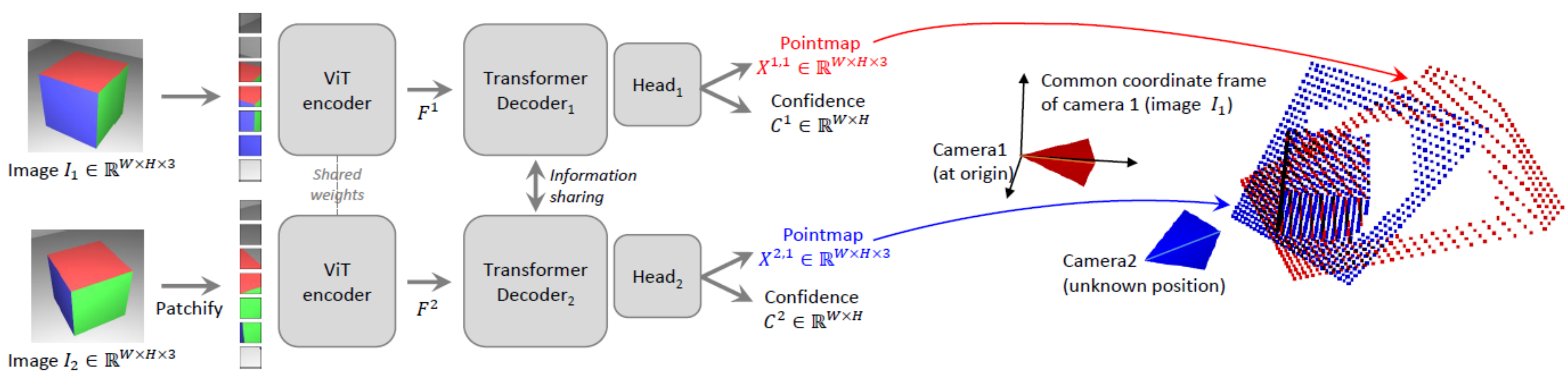


Figure 2. **Architecture of the network $\mathcal{F}$.** Two views of a scene (I^1, I^2) are first encoded in a Siamese manner with a shared ViT encoder. The resulting token representations F^1 and F^2 are then passed to two transformer decoders that constantly exchange information via cross-attention. Finally, two regression heads output the two corresponding pointmaps and associated confidence maps. Importantly, the two pointmaps are expressed in the same coordinate frame of the first image I^1. The network $\mathcal{F}$ is trained using a simple regression loss (Eq. (4))