E1.216 : COMPUTER VISION

LECTURE 10: 3D RECONSTRUCTION FROM IMAGES

Structure from Motion

Venu Madhav Govindu
Department of Electrical Engineering
Indian Institute of Science, Bengaluru

2023

In this lecture we shall look at **3D Reconstruction** from images

Structure and Motion Estimation

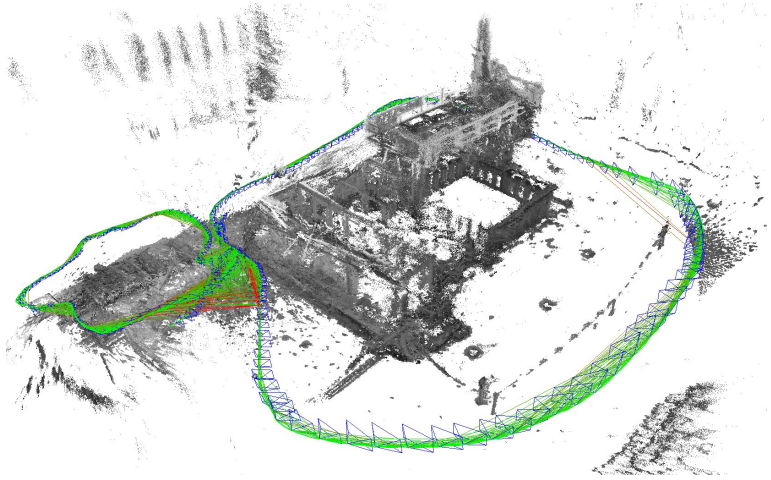
N-view camera geometry

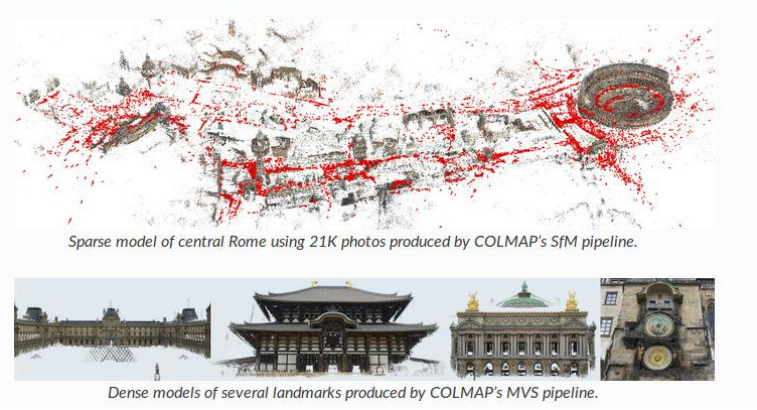
- Previous Lecture: Two-view or Epipolar Geometry
- Multiview idea extensible to 3 and 4 views
- Results in *trifocal* or *quadrifocal* tensor
- No more such tensorial forms available beyond 4
- Historically, other solutions were developed earlier
- Tensorial representation arose relatively recently
- General problem: 3D reconstruction from many images

Structure and Motion Estimation

Camera geometry of multiple views

- The general problem is that of using multiple views of a scene
- Scene is assumed to be static
- Solve for camera *motion* and geometry of *3D structure*
- Significant advances in the engineering of solutions
- Results in solutions for both **structure** and **motion**
- Often called **structure from motion**
- In robotics: SLAM for **simultaneous localisation and mapping**





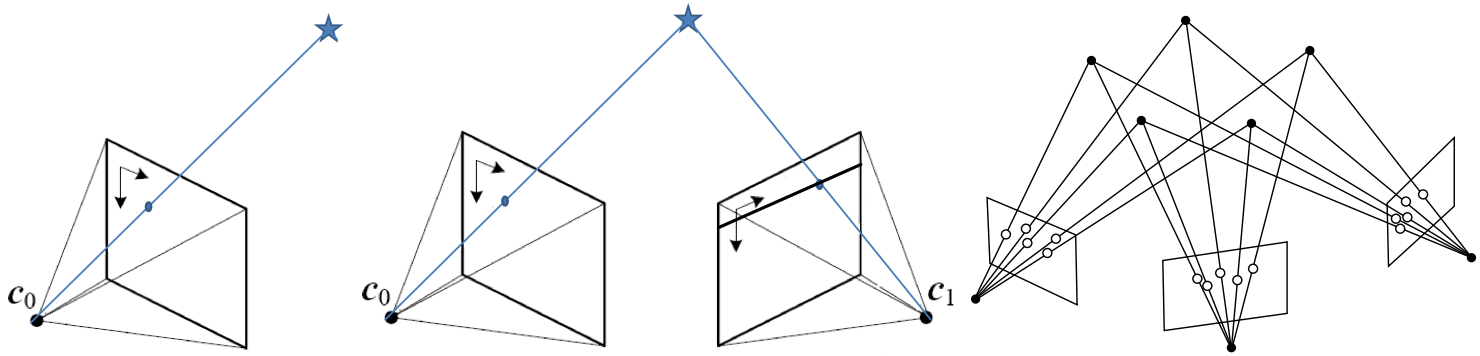
Sparse model of central Rome using 21K photos produced by COLMAP's SfM pipeline.

Dense models of several landmarks produced by COLMAP's MVS pipeline.

Geometric Problems in Computer Vision

- Simultaneous Localisation and Mapping (SLAM)
- Structure from Motion (SfM)
- Common to both approaches:
  - 3D Reconstruction from multiple images
  - Geometry induced by pinhole camera
- How are SfM and SLAM different?

Engel *et al.* 'LSD-SLAM'; COLMAP


c_0
c_0
c_1

Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Ideal Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

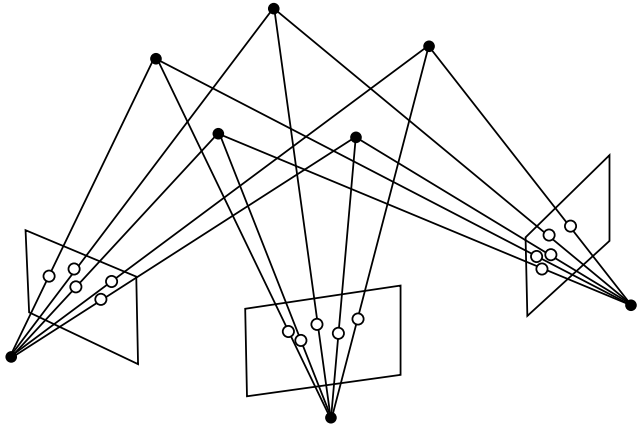
Projective Geometry

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Taking a Picture

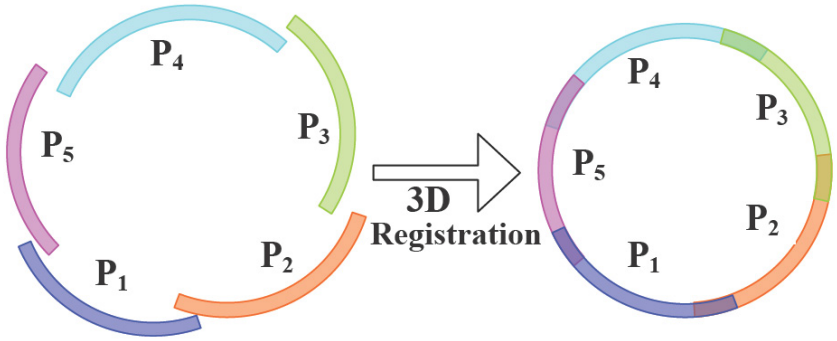
- 3D Point in Homogeneous Form
- Rigid 3D Motion
- Ideal Pinhole Camera
- Image Projection
- $\mathbb{P}^3 \rightarrow \mathbb{P}^2$

Introduction



$$\min_{\boldsymbol{R},\boldsymbol{T},\boldsymbol{S}} \sum d^2(x_j^i, \hat{x}_j^i) + d^2(y_j^i, \hat{y}_j^i)$$

Minimise Reprojection Error



$$\min_{\boldsymbol{R},\boldsymbol{T}} \sum_{i=1}^{N} \|\boldsymbol{R}\boldsymbol{P}_1(i) + \boldsymbol{T} - \boldsymbol{P}_2(i)\|^2$$

3D Registration + Correspondences

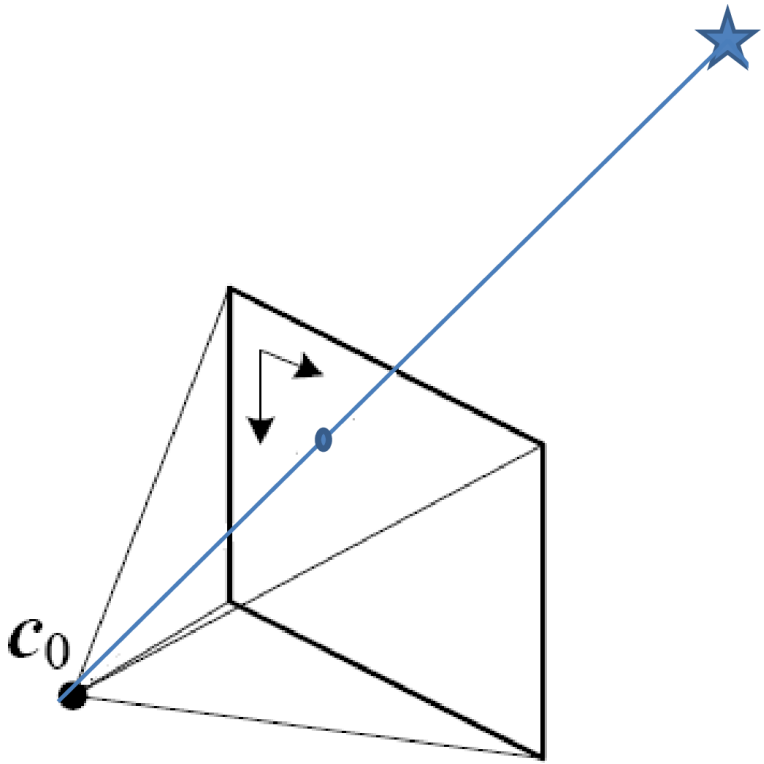
Pin-hole Camera Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{I} \mid \mathbf{0} \end{bmatrix} \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \hline \mathbf{0} & 1 \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

3D Registration

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \hline \mathbf{0} & 1 \end{bmatrix}}_{\boldsymbol{M}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

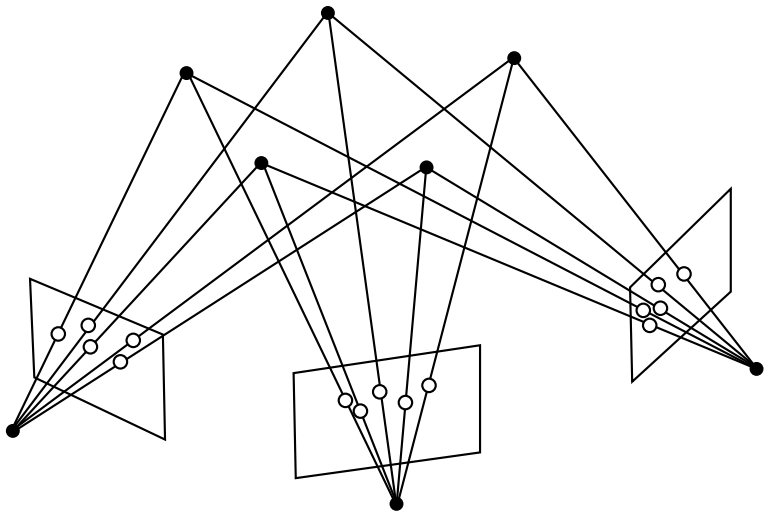
Structure and Motion Estimation



$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \boldsymbol{K} \begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{T} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Structure and Motion Estimation



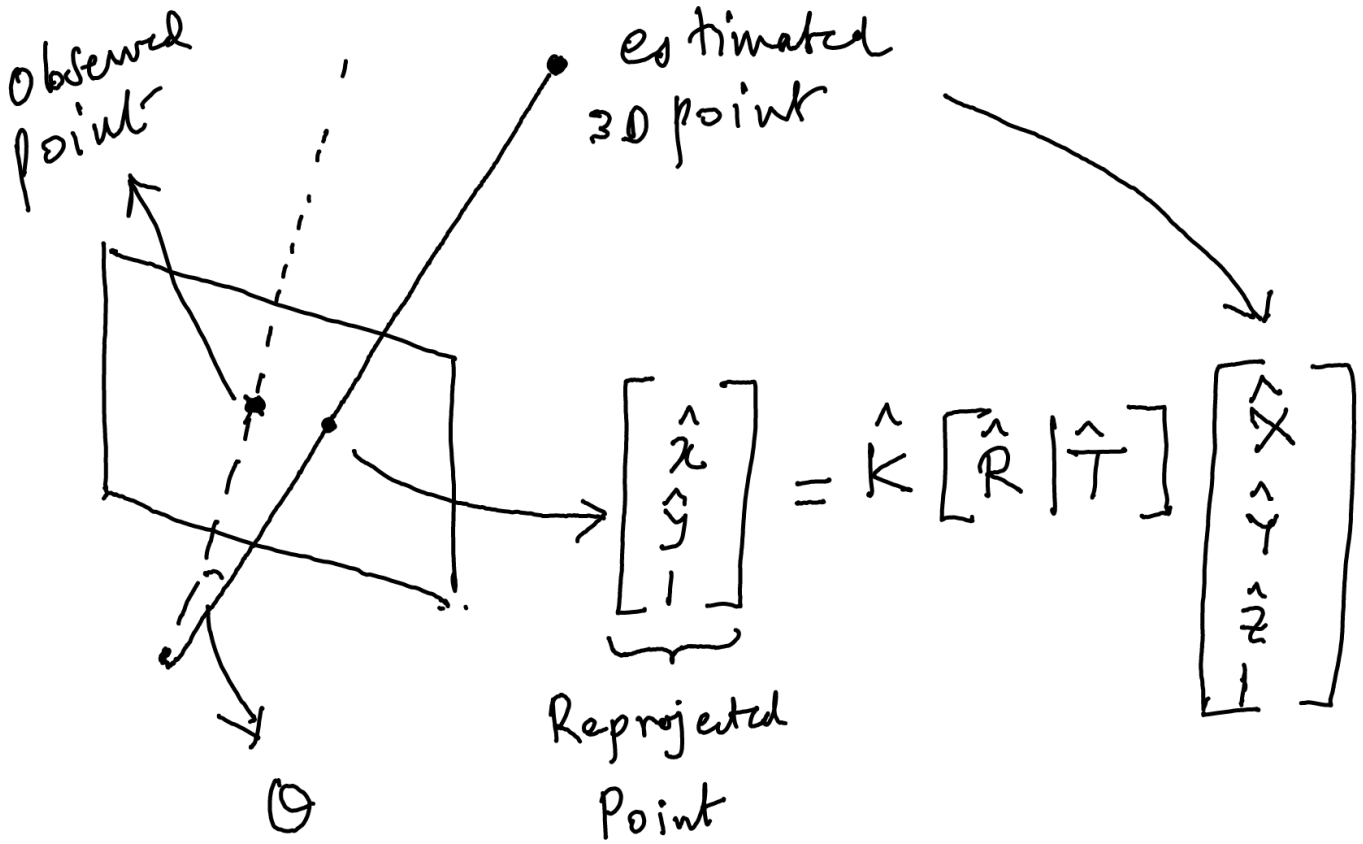
$$\begin{bmatrix} x_j^i \\ y_j^i \\ 1 \end{bmatrix} = \boldsymbol{P}^i \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix}$$

$$= \boldsymbol{K}^i \left[\begin{array}{c|c} \boldsymbol{R}^i & \boldsymbol{T}^i \end{array} \right] \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix}$$

Multiple Cameras

- i-th camera and j-th point
- Let observed point be $\boldsymbol{x}_j^i$
- Assume estimates for structure and motion
- Estimated projection is $\hat{\boldsymbol{x}}_j^i$ (function of structure and motion variables)
- Measure discrepancy $d(\boldsymbol{x}_j^i, \hat{\boldsymbol{x}}_j^i) = ||\boldsymbol{x}_j^i - \hat{\boldsymbol{x}}_j^i||$
  - **Bundle adjustment** of estimated rays
  - Distance on image plane (**reprojection error**)

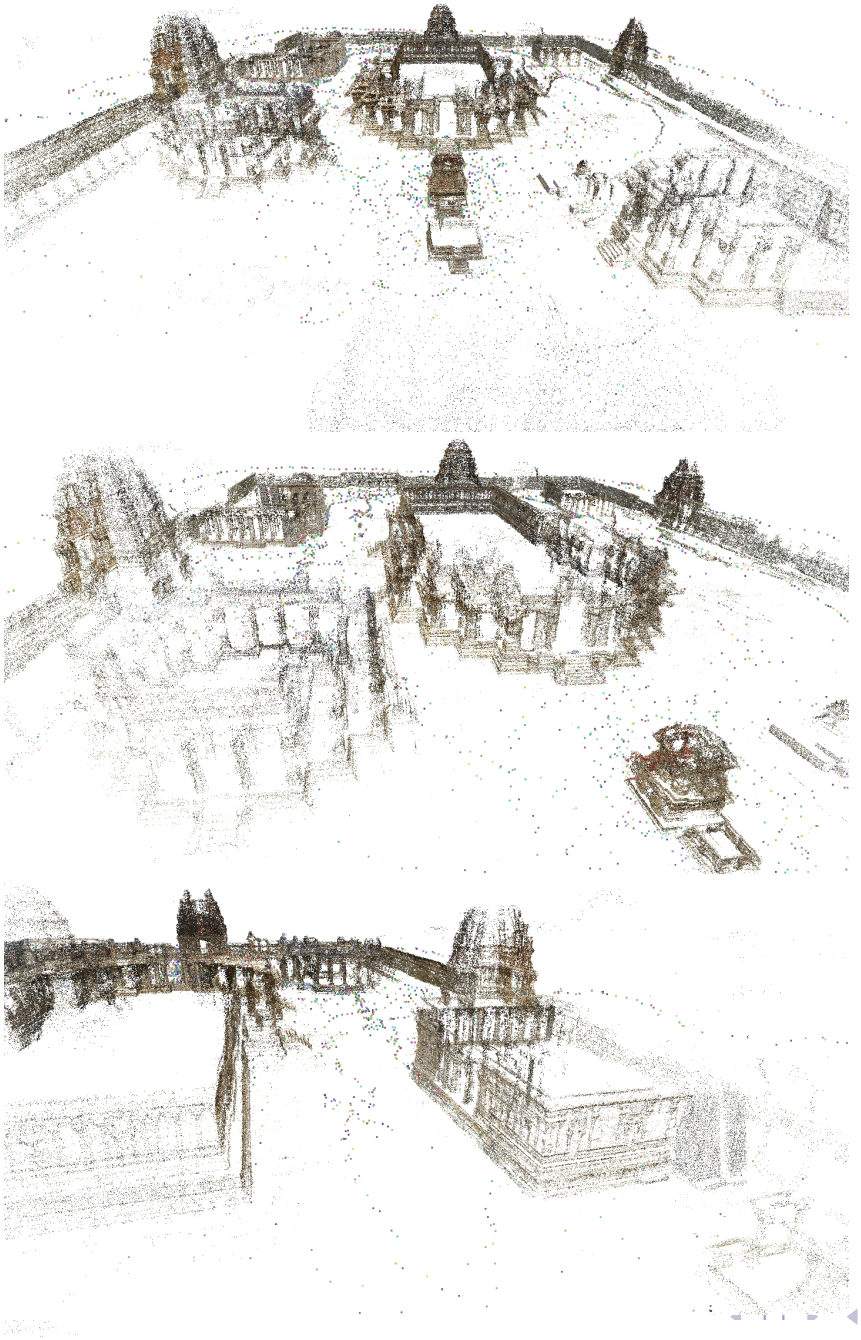
Structure and Motion Estimation



Minimisation of Reprojection Error

$$\min_{\boldsymbol{K},\boldsymbol{R},\boldsymbol{T},X,Y,Z} \sum_i \sum_j ||\boldsymbol{x}_j^i - \hat{\boldsymbol{x}}_j^i||^2$$

Vitthala Temple, Hampi



Structure and Motion Estimation

$$\min_{\hat{P}^i, \hat{X}_j} \sum_{ij} d(\hat{P}^i \hat{X}_j, x_j^i)^2 = \sum_{ij} ||Proj(\hat{P}^i \hat{X}_j) - x_j^i||^2$$

Here P is 3×4 camera matrix
Can also parametrise as a physical camera $P = K[R|t]$

Error measure

- Require projective invariance to choice of basis
- Measure error in the image plane
- Error measure is defined in terms of "reprojection error"
- Measure distance from point to its reprojected image
- Take a least-squares approach
- Lsq justified as MLE given Gaussian noise in image point location

Bundle Adjustment

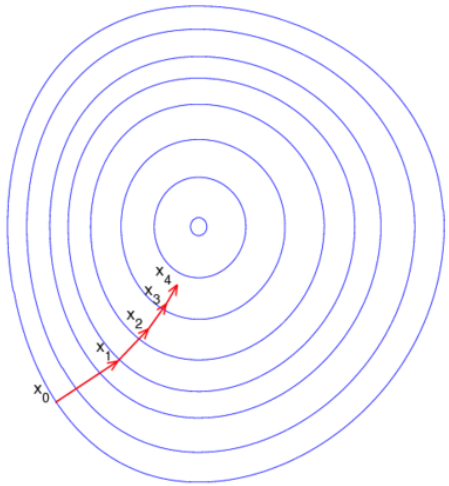
$$\min_{\hat{P}^i, \hat{X}_j} \sum_{ij} d(\hat{P}^i \hat{X}_j, x_j^i)^2 = \sum_{ij} ||Proj(\hat{P}^i \hat{X}_j) - x_j^i||^2$$

- Adjust bundle of rays between each camera centre and 3D points
- Can tolerate missing data (i.e. matches not visible in some images)
- Provides a true MLE despite missing data
- $d(.,.)$ can be modified to incorporate error covariances
- Minimisation of cost function is a complicated problem
- Use general non-linear least-squares minimisation methods
- Levenberg-Marquardt minimisation is the method of choice
- Works very well in practice
- Needs a good initialisation to avoid getting stuck in local minima

Before we arrive at the Levenberg-Marquardt method,
let's very quickly review

- Gradient Descent
- Newton algorithm

Then we can see how Levenberg-Marquardt skillfully blends two different methods while retaining their advantages



Gradient Descent

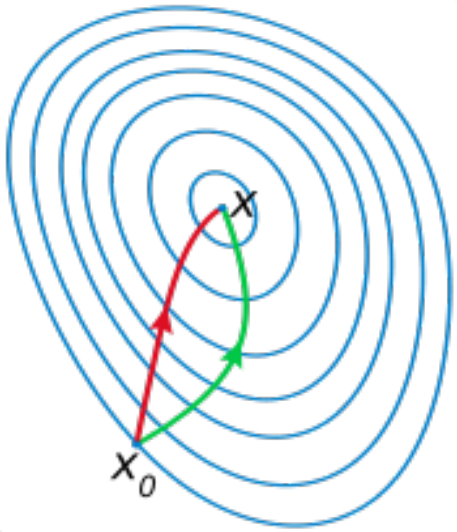
- Given a multivariate function $F(\boldsymbol{x})$
- Start at a point $\boldsymbol{x}_0$
- Move along the direction of the gradient $\nabla F(\boldsymbol{x})$
- Update equation $\boldsymbol{x} \leftarrow \boldsymbol{x} - \lambda \nabla F(\boldsymbol{x})$
- Repeat till convergence (hopefully!)

Picture taken from www.wikipedia.org

Limitations of Gradient Descent

- Gradient Descent is simple and easy to implement
- Many iterations to converge if curvature varies in different directions
- Figuring out optimal step-size λ is time consuming
- Small λ controls convergence but is slow
- Large λ results in speed-up but can overshoot
- Step-size is not constant but dependent on gradient magnitude
- Shallow areas result in small steps (bad)
- Conversely, in steep areas one could overshoot
- Consider long valley

Structure and Motion Estimation



$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

Newton Algorithm

- Can use *second order* information
- Use curvature along with gradient direction information
- Consider $F(\boldsymbol{x} + \triangle \boldsymbol{x}) = F(\boldsymbol{x}) + F^{'}(\boldsymbol{x})^{T} \triangle \boldsymbol{x} + \triangle \boldsymbol{x}^{T} F^{''}(\boldsymbol{x}) \triangle \boldsymbol{x}$
- Minimisation done by solving for $\triangle \boldsymbol{x}$
- Update is $\boldsymbol{x} \leftarrow \boldsymbol{x} - \boldsymbol{H}^{-1} \nabla F(\boldsymbol{x})$
- $\boldsymbol{H}$ is the Hessian of the function evaluated at current $\boldsymbol{x}$

Comparison

$$\begin{aligned} \boldsymbol{x} &\leftarrow \boldsymbol{x} - \nabla F \\ \boldsymbol{x} &\leftarrow \boldsymbol{x} - \boldsymbol{H}^{-1} \nabla F \end{aligned}$$

What does this imply ?

- We approximate $F(\boldsymbol{x})$ locally as quadratic function
- Can jump to local optimum
- If close to solution, we can guess minimum well
- If approximation is good, we converge faster than steepest descent

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1} \nabla F$$

Levenberg's Algorithm

- Can blend Newton's method with steepest descent
- For large λ, rule is $\boldsymbol{x} \leftarrow \boldsymbol{x} - \frac{1}{\lambda} \nabla F$ (s.d.)
- For small λ, rule is the Newton update
- Start with steepest descent and gradually move towards quadratic rule
- Control the shift from one regime to another using error improvement

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1} \nabla F$$

Steps in Levenberg's Algorithm

- Carry out an update using rule
- Evaluate error measure at new location and compare with previous position's error
- If error *increases*, go back to previous position and *increase* λ
- If error *decreases*, keep update and *decrease* λ
- Scaling of λ at each step done using a fixed constant, say 10

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1} \nabla F$$

Justification for Levenberg's Algorithm

- If error increases after step, quadratic approximation is poor
- Need to move towards steepest descent as likely far from minima
- Done by increasing λ
- If error decreases, approximation is good
- Can converge well using quadratic approximation
- Move away from steepest descent approach by decreasing λ

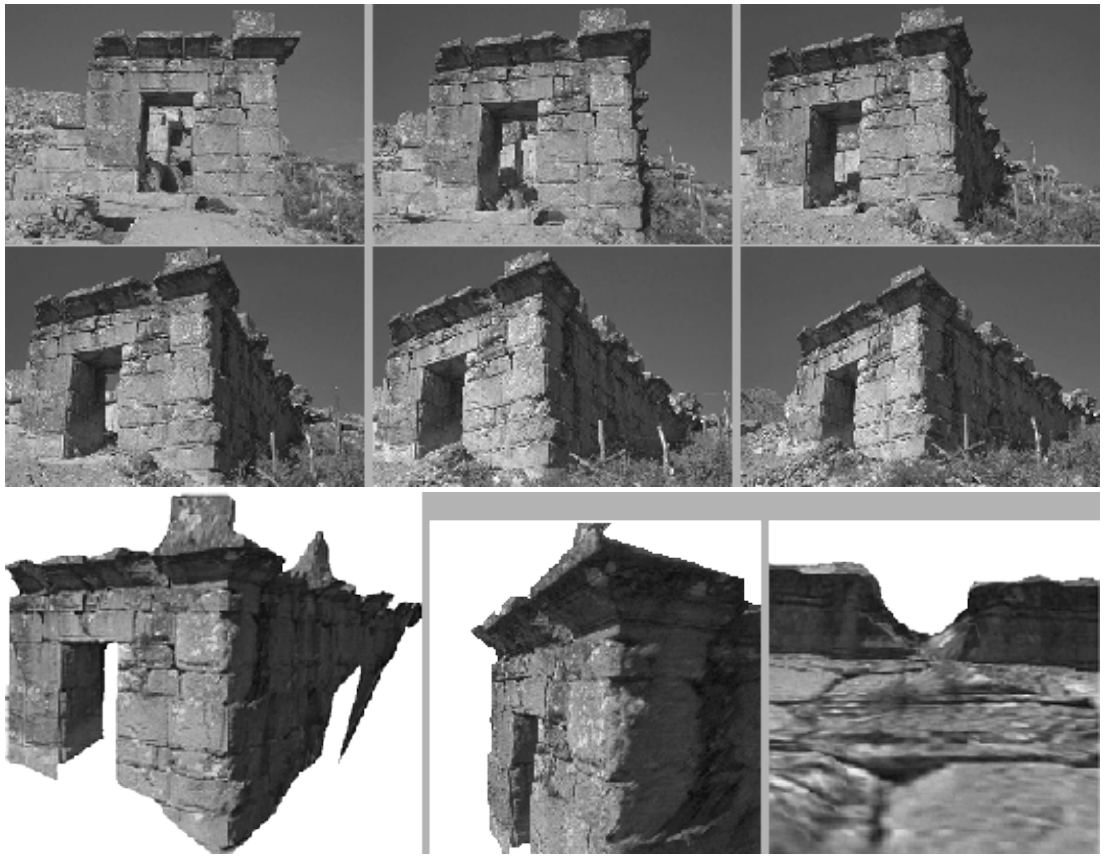
Comparison

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1} \nabla F \quad \textit{Levenberg}$$
$$\boldsymbol{x} \leftarrow \boldsymbol{x} - (\boldsymbol{H} + \lambda \textit{diag}(\boldsymbol{H}))^{-1} \nabla F \quad \textit{Marquardt}$$

What did Marquardt add to this ?

- Insight that incorporated local curvature information
- When λ is high, doing gradient descent
- Can do better than Levenberg by using Hessian information
- Move further in the direction where gradient is small
- Improves convergence in this process
- Combination known as **Levenberg-Marquardt algorithm**

Structure and Motion Estimation

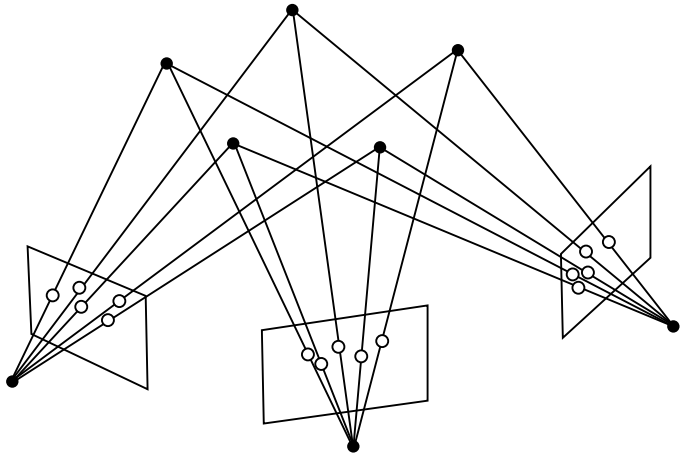


Results of Marc Pollefeys

Structure and Motion Estimation

Results of Marc Pollefeys

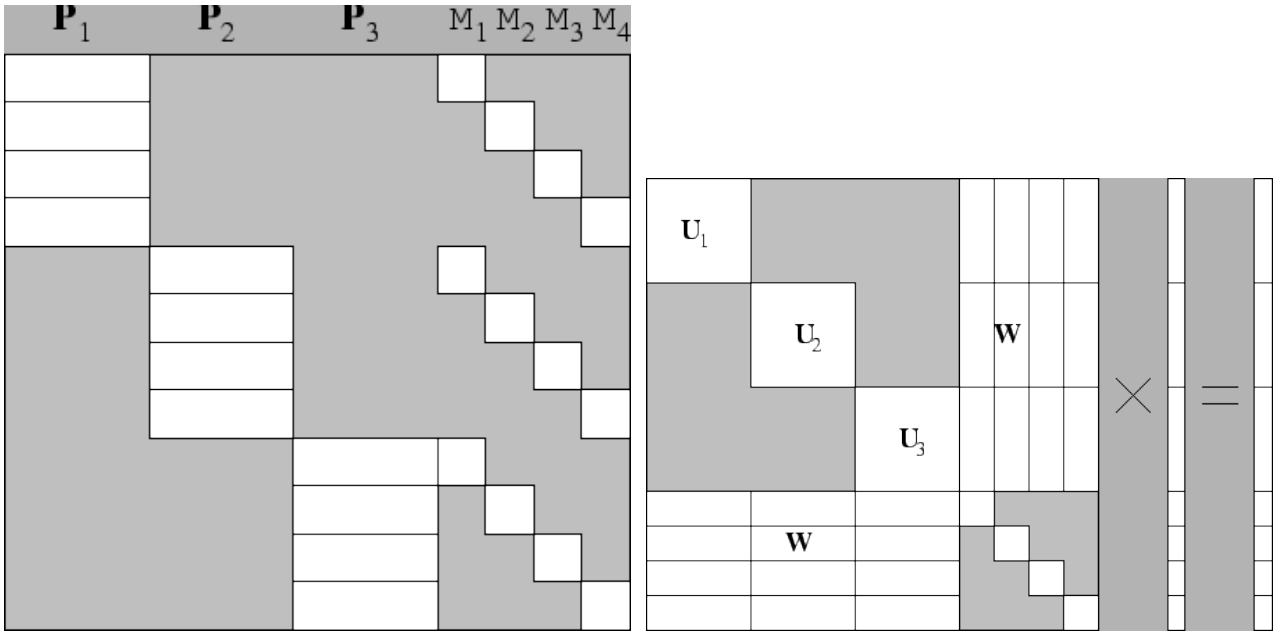
Structure and Motion Estimation



Issues with Bundle Adjustment

- **Too many points**
  - Do not include all views or all points at the same time
- **Interleave Speed-up**
  - Alternately adjust structure and motion
  - Reduces size of Hessian to be inverted to max of 11 $\times$ 11
- **Sparse Methods**
  - Effectively exploit the sparsity of the Jacobian involved
  - Jacobian form is used to approximate the Hessian
  - Approximation of form $(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T$

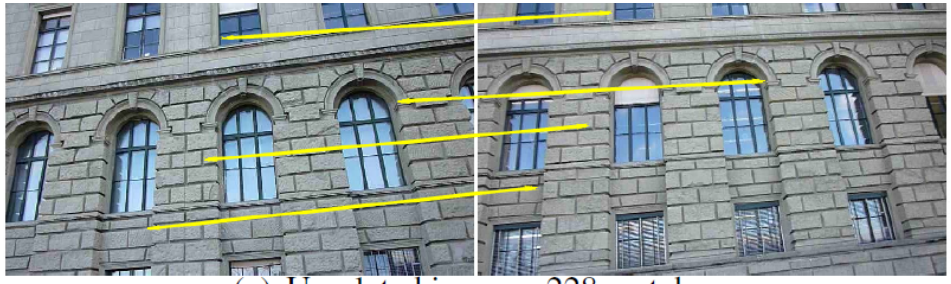
Structure and Motion Estimation



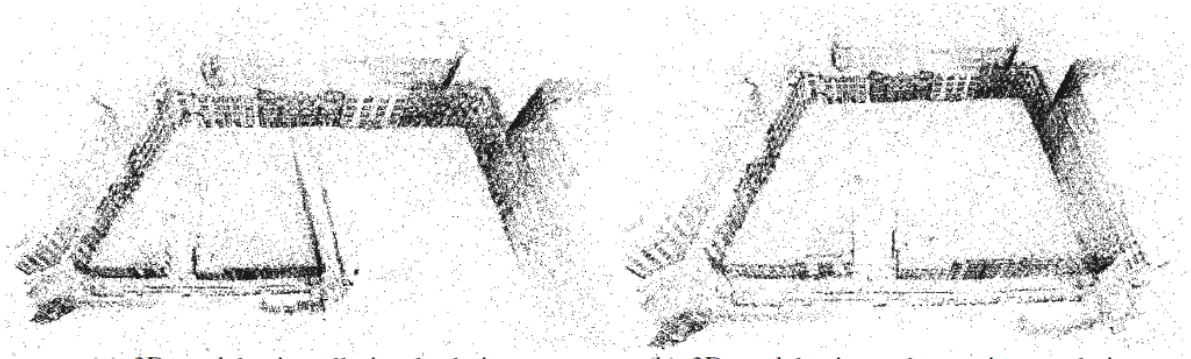
Exploiting Sparse Structure

- Jacobian has a sparse structure
- Separates out relationship between point structure and cameras
- Can exploit this in solving the normal equations
- Results in significant speed-up
- Makes bundle adjustment quite attractive

Robustness



(a) Unrelated images, 228 matches



(a) 3D model using all visual relations

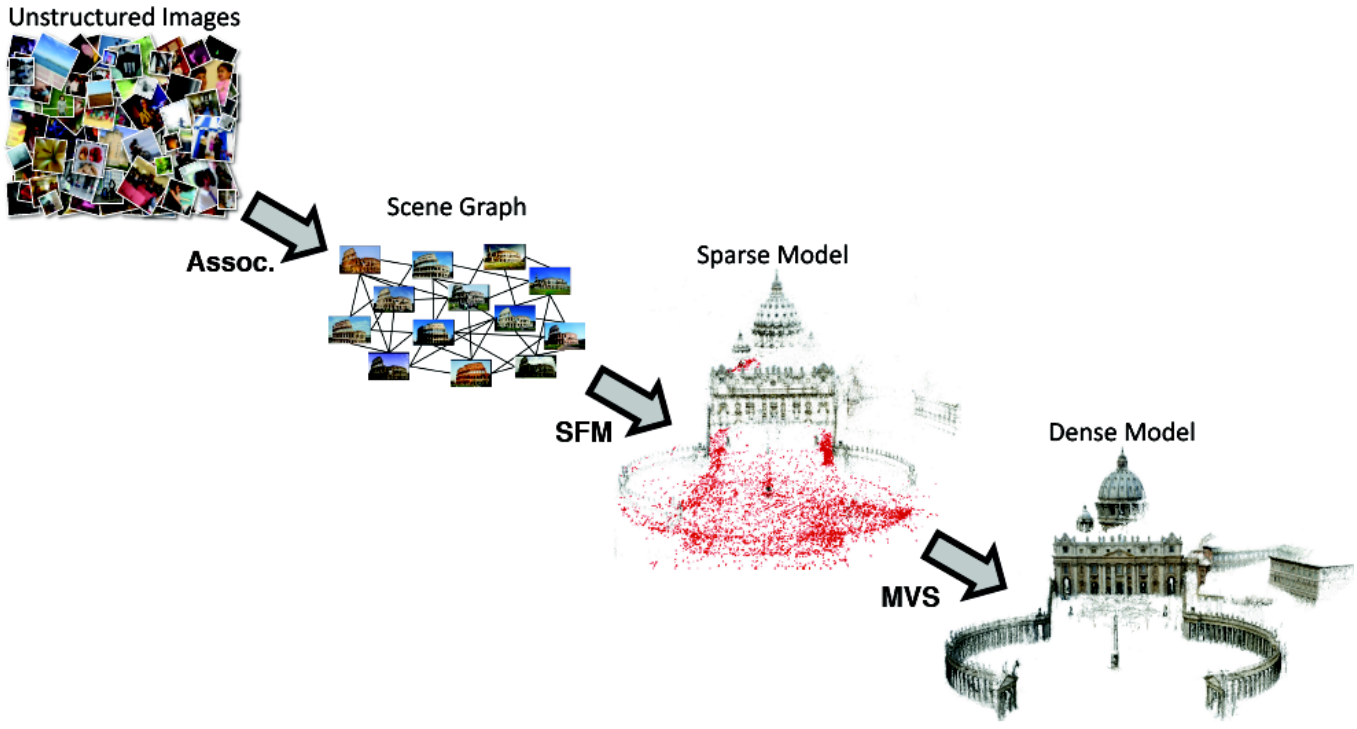
(b) 3D model using only consistent relations

Outliers in data

- Outliers are ubiquitous in real data
- Robustness
  - Identify and remove outliers (RANSAC)
  - Estimation in presence of outliers (M-estimation)

Figure from Zach *et al.* "Disambiguating Visual Relations Using Loop Constraints"

Structure and Motion Estimation



Large-scale SfM

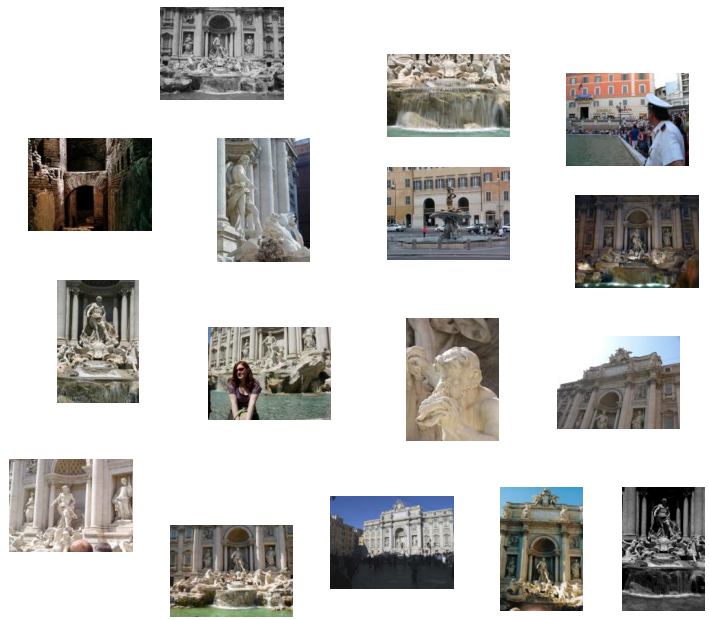
- Major advances in all aspects
- Can handle 10000+ images
- Many heuristics for unstructured data
- Key ingredients: Geometry, Optimization, Data Structures

Picture from tutorial 'Large-scale 3D Modeling from Crowdsourced Data', CVPR 2017

Following slides are borrowed from Noah Snavely's lecture

Feature detection

Detect features using SIFT [Lowe, IJCV 2004]

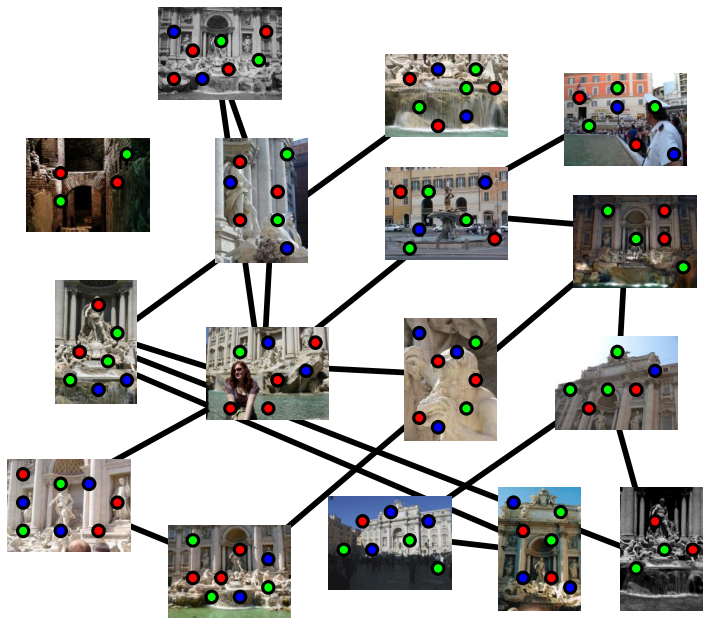


Feature detection

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

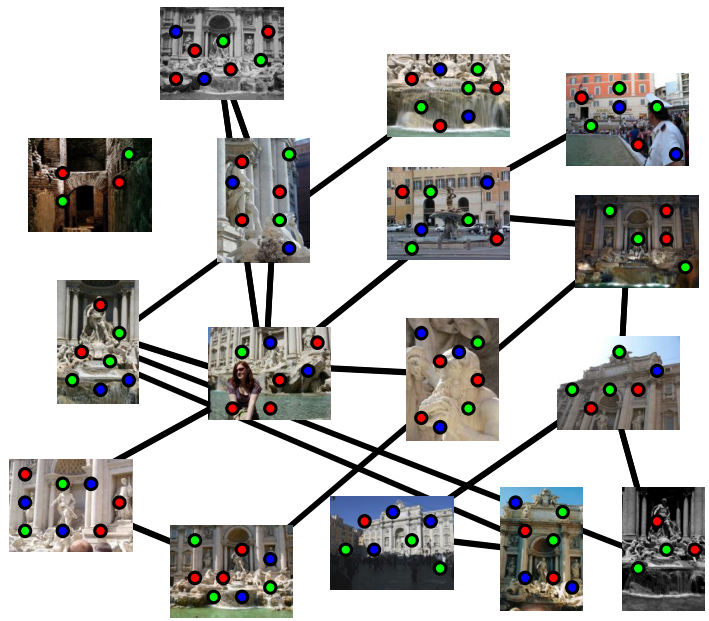
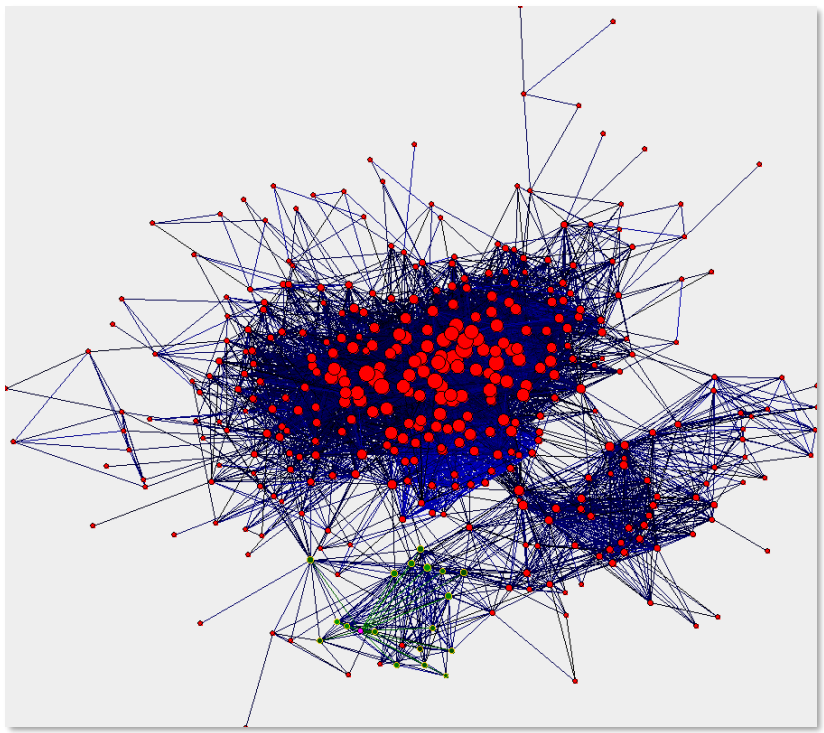


Image connectivity graph



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

Correspondence estimation

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

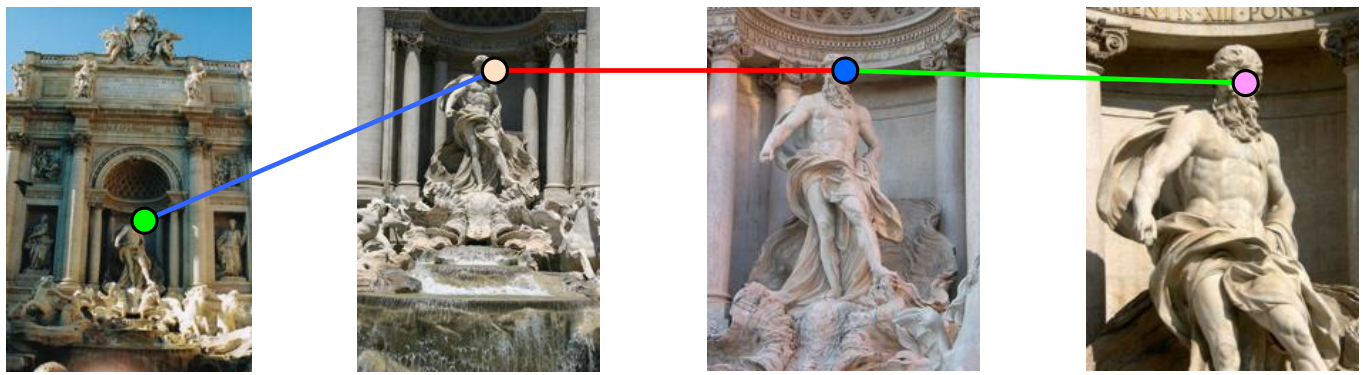
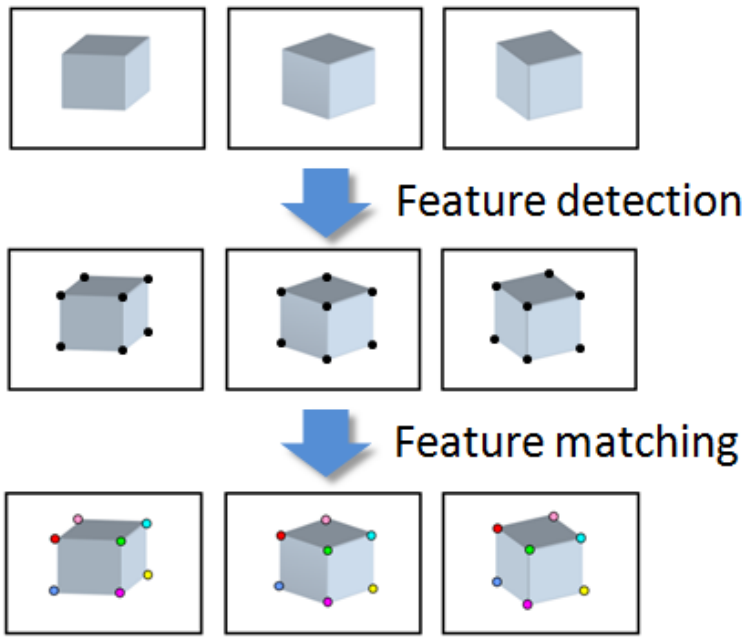
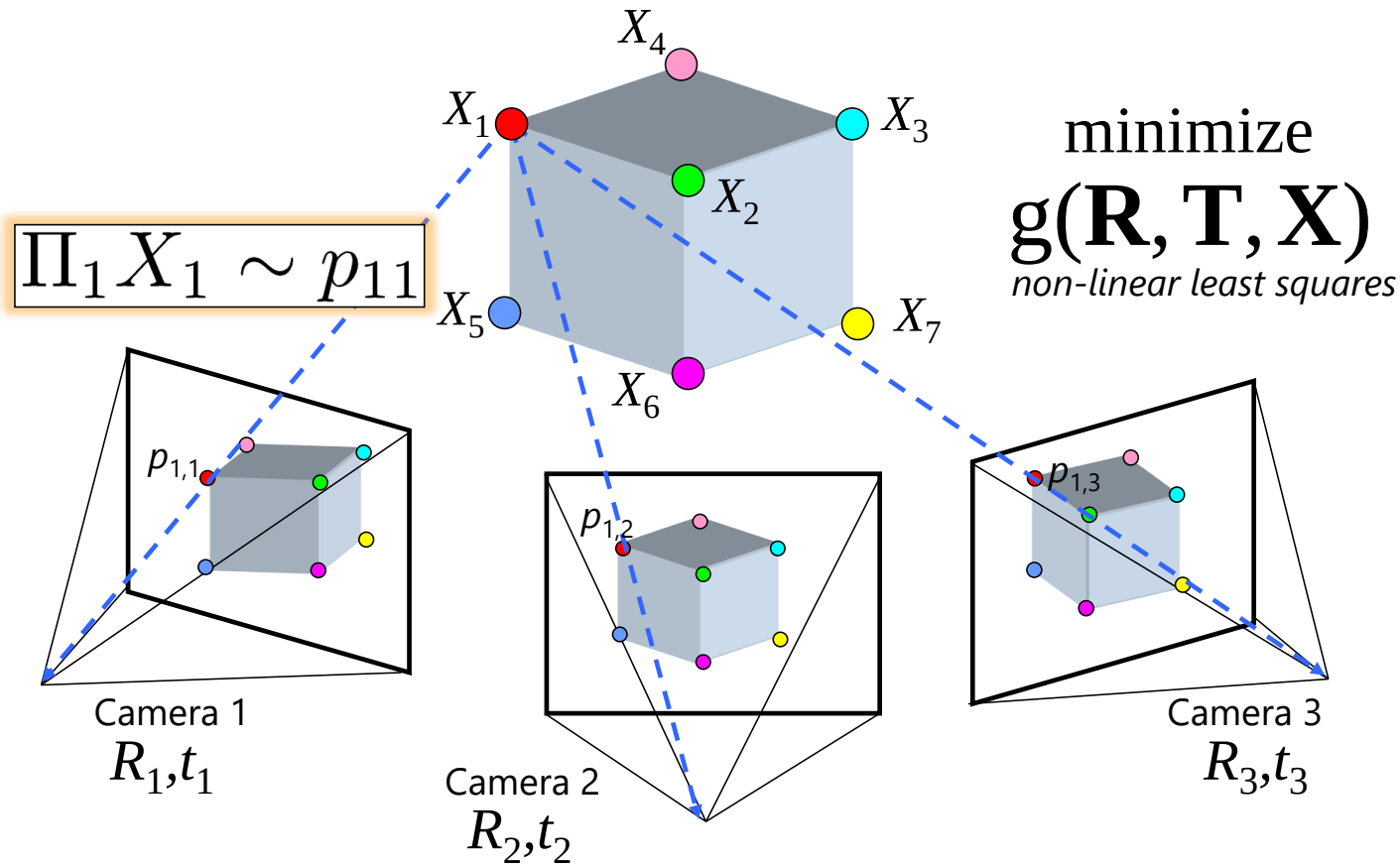


Image 1 Image 2 Image 3 Image 4

Input to Structure from Motion



Structure from motion



Problem size

- What are the variables?
  - Cameras and points
- How many variables per camera?
  - 6 (if calibrated), more if uncalibrated
- How many variables per point?
  - 3

- Trevi Fountain collection

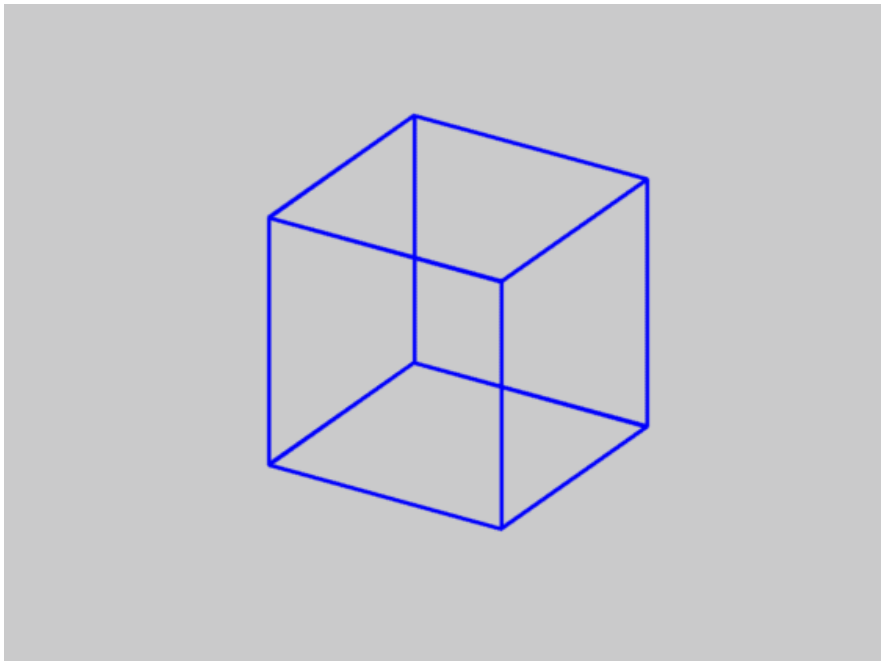
  466 input photos

  \+ > 100,000 3D points

  = very large optimization problem

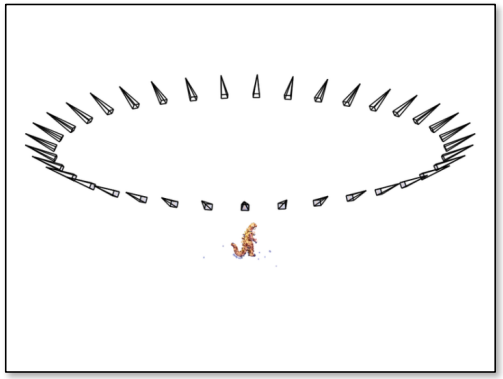
Is SfM always uniquely solvable?

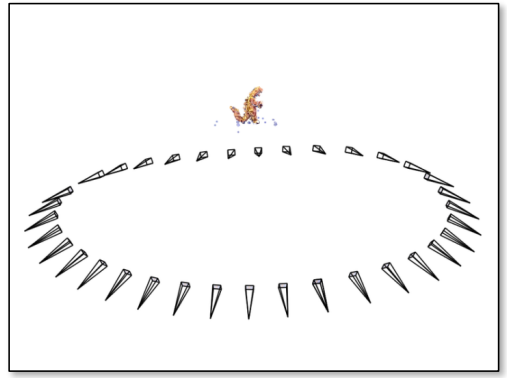
- No...



SfM – Failure cases

- Necker reversal

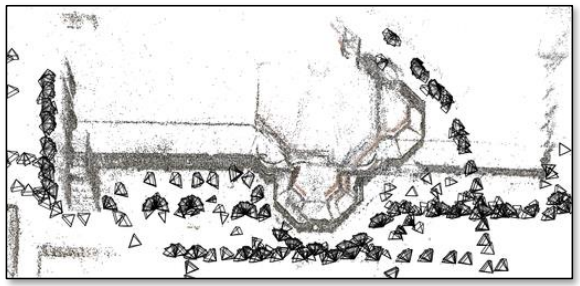




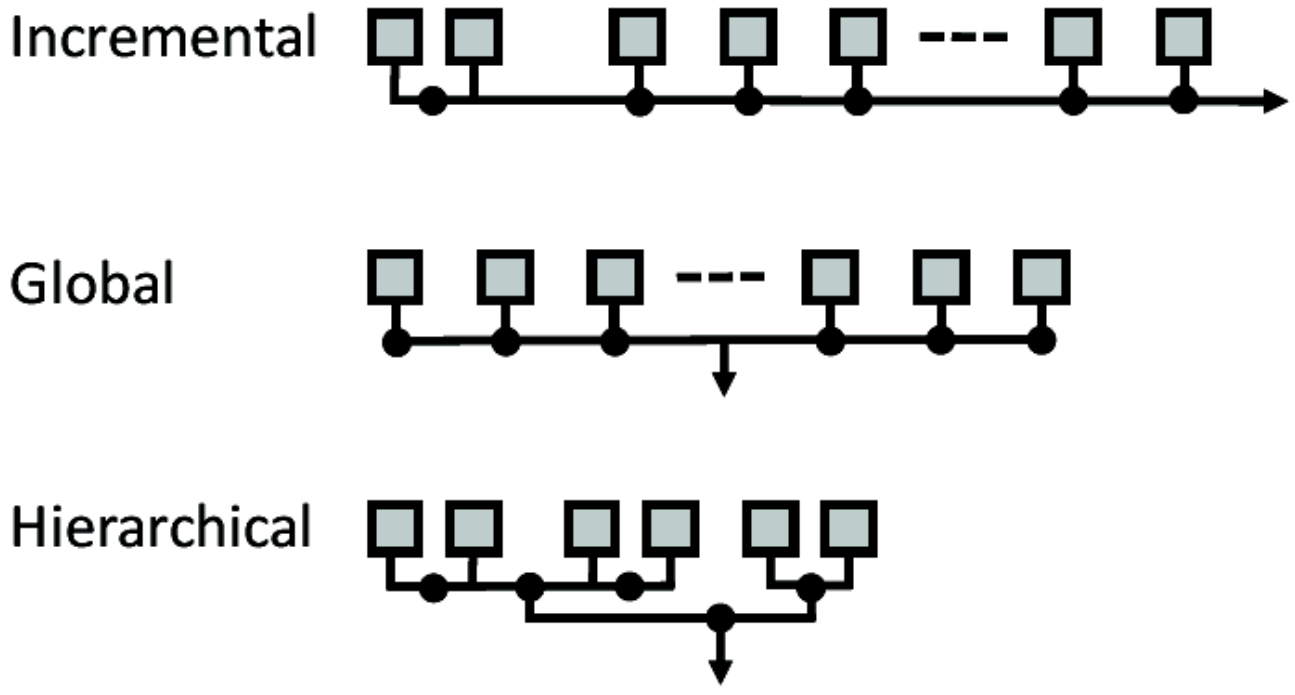
Structure from Motion – Failure cases

- Repetitive structures: Symmetries in man-made scenes






Structure and Motion Estimation

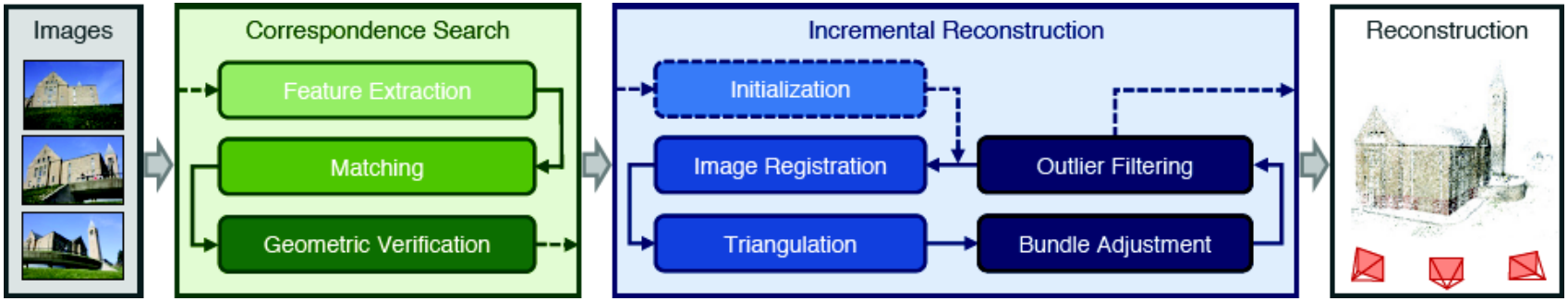


SfM Paradigms

- Most pipelines are incremental
- Global and hierarchical methods increasingly popular

Picture from tutorial 'Large-scale 3D Modeling from Crowdsourced Data', CVPR 2017

Structure and Motion Estimation

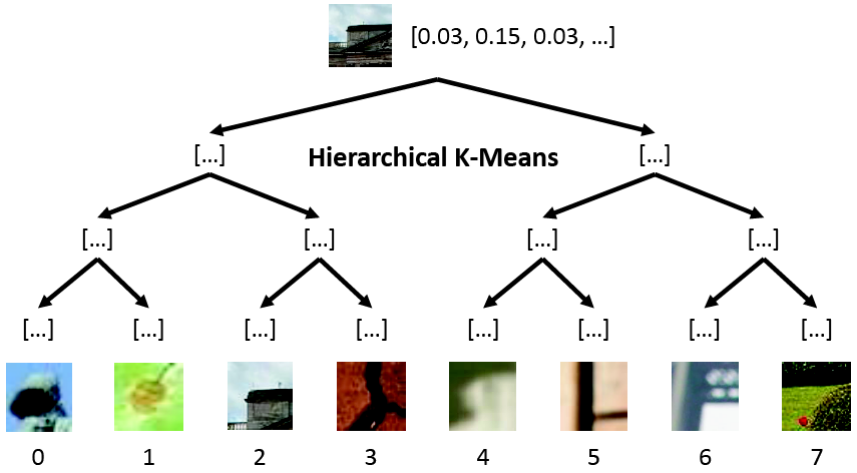


Incremental SfM

- Feature correspondence most expensive
- Geometric verification is vital
- Need robustness and stable estimation
- Large number of heuristics for unstructured data
- Impressive results on large datasets

Picture from tutorial 'Large-scale 3D Modeling from Crowdsourced Data', CVPR 2017

Point Features



Matching Images

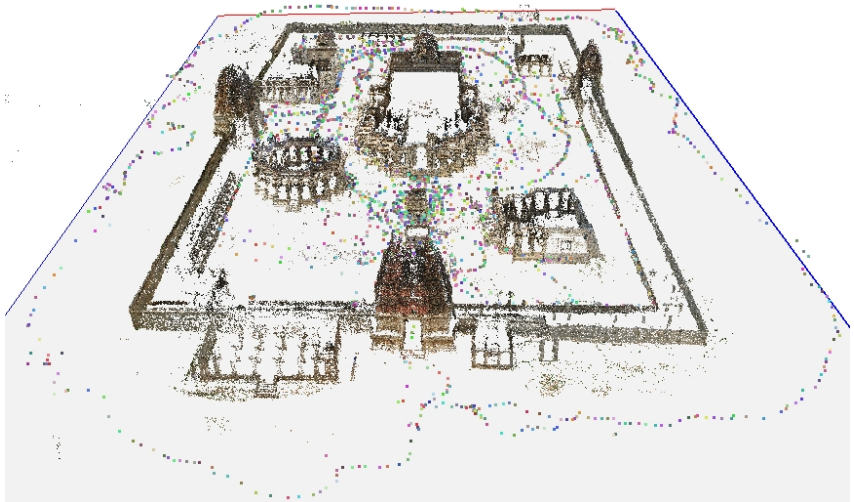
- Test for matching image pairs
- Brute force prohibitively expensive
- High dimensional search
- Vocabulary tree representation used
- Inverted table
- Represent image by 'bag-of-words'
- Major computational bottleneck
- Need geometric verification

Picture from tutorial 'Large-scale 3D Modeling from Crowdsourced Data', CVPR 2017

3D Reconstruction from Images

- Take lots of pictures
- We took around 2500 of them!
- Solve the *structure-from-motion* problem (invert image formation)
- And the result is $\cdots$

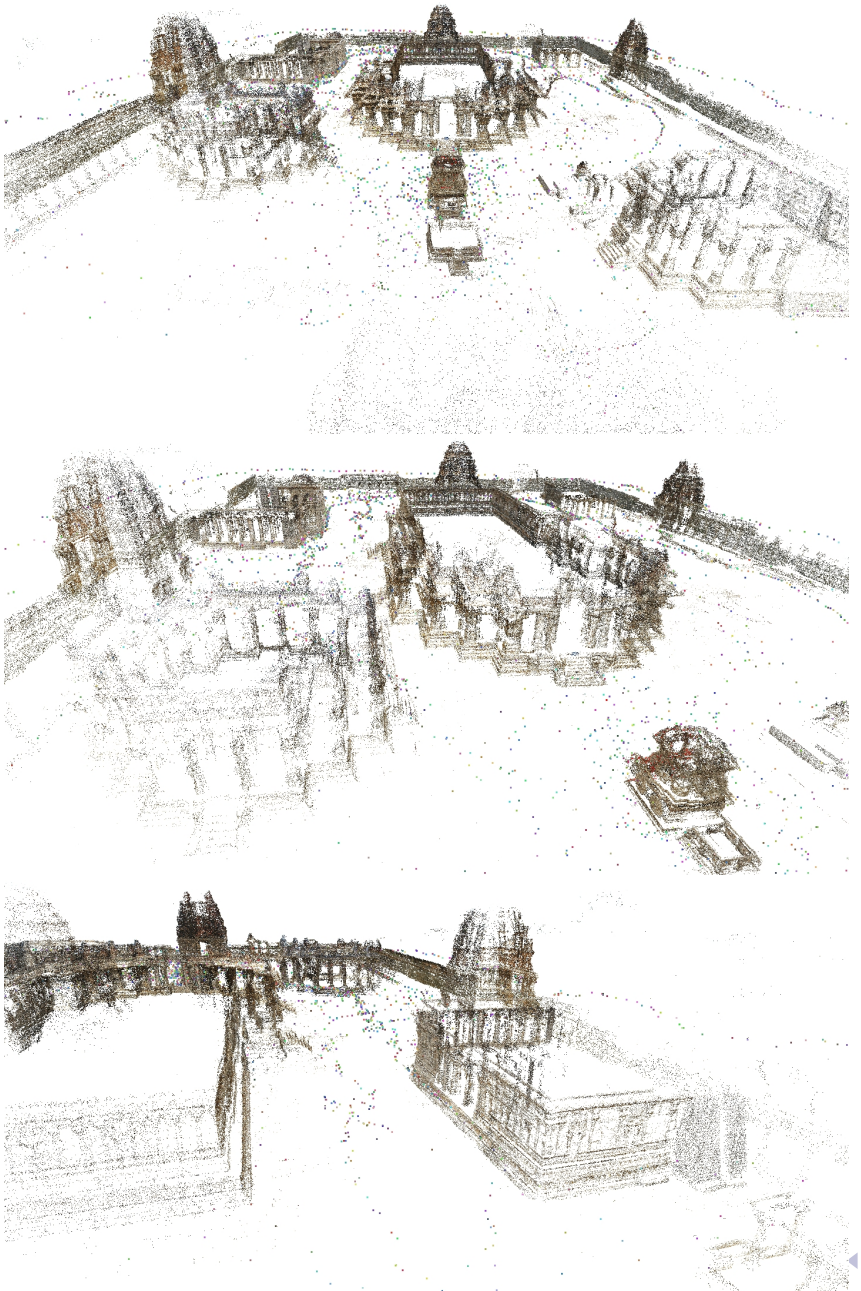
3D Reconstruction from Images



- Take lots of pictures
- We took around 2500 of them!
- Solve the *structure-from-motion* problem (invert image formation)
- And the result is $\cdots$

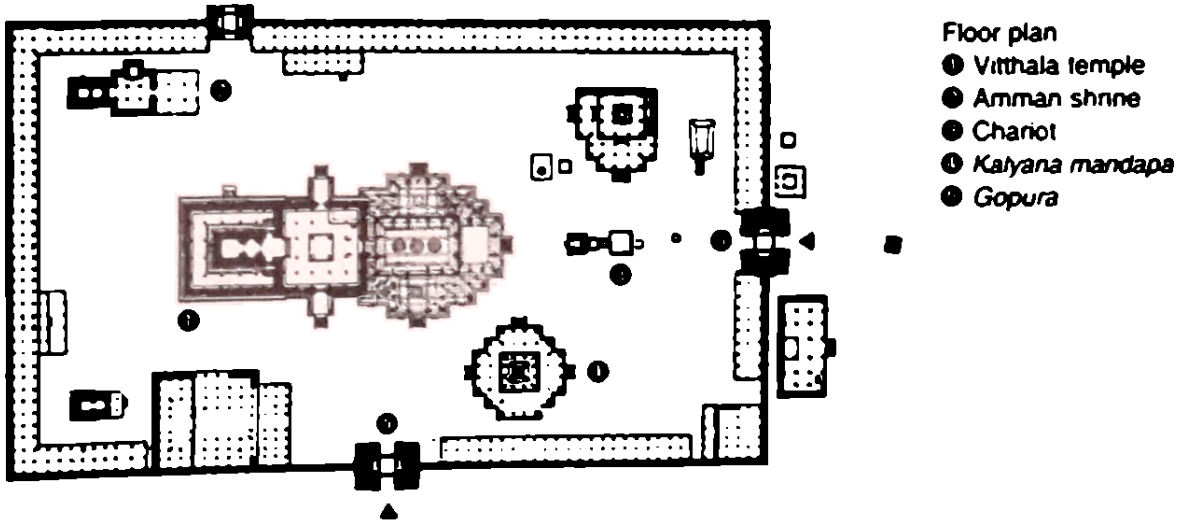
Vitthala Temple, Hampi

Rendering of our 3D model

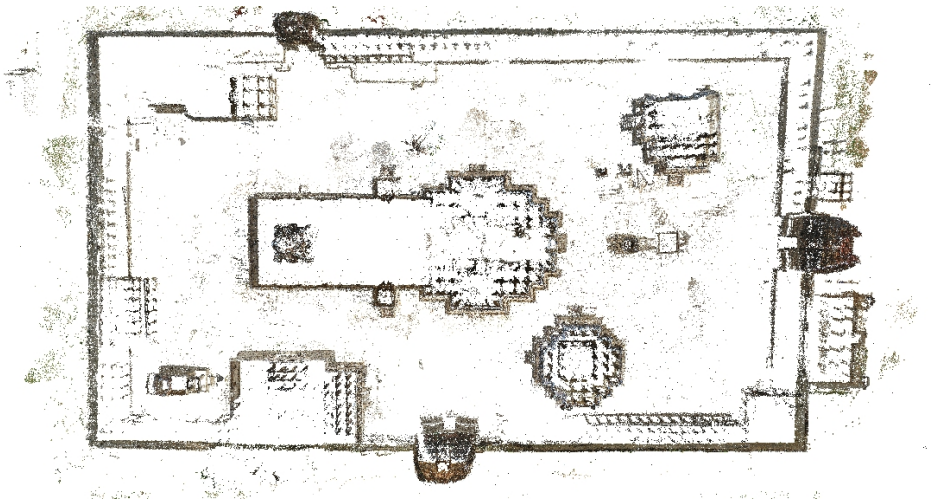


Vitthala Temple, Hampi

Comparison of Reconstruction



(a) Architectural Drawing



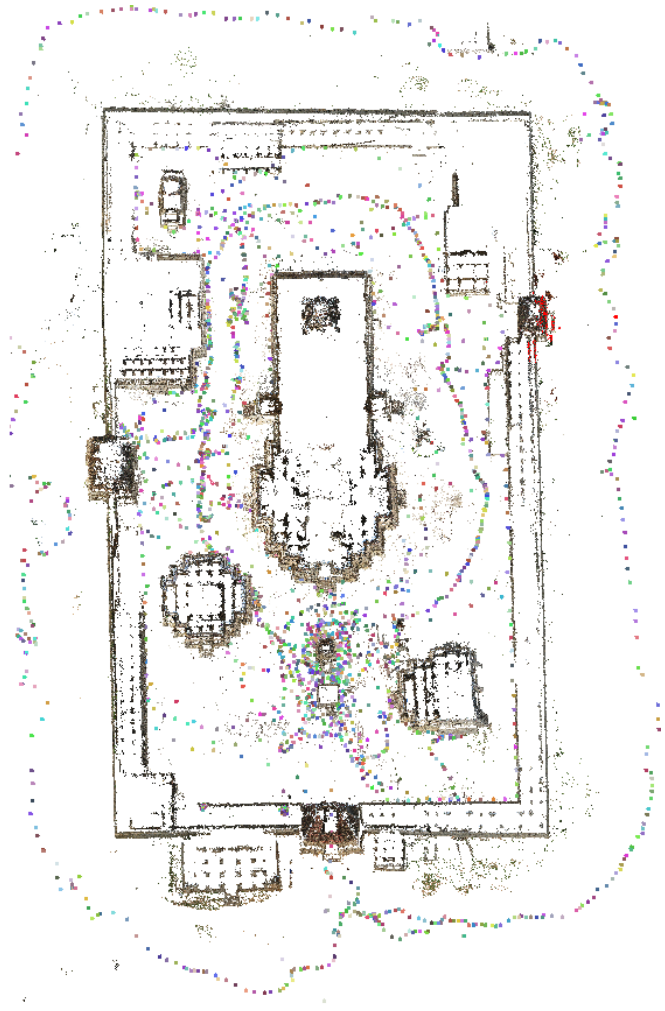
(b) Reconstruction

Incremental SfM

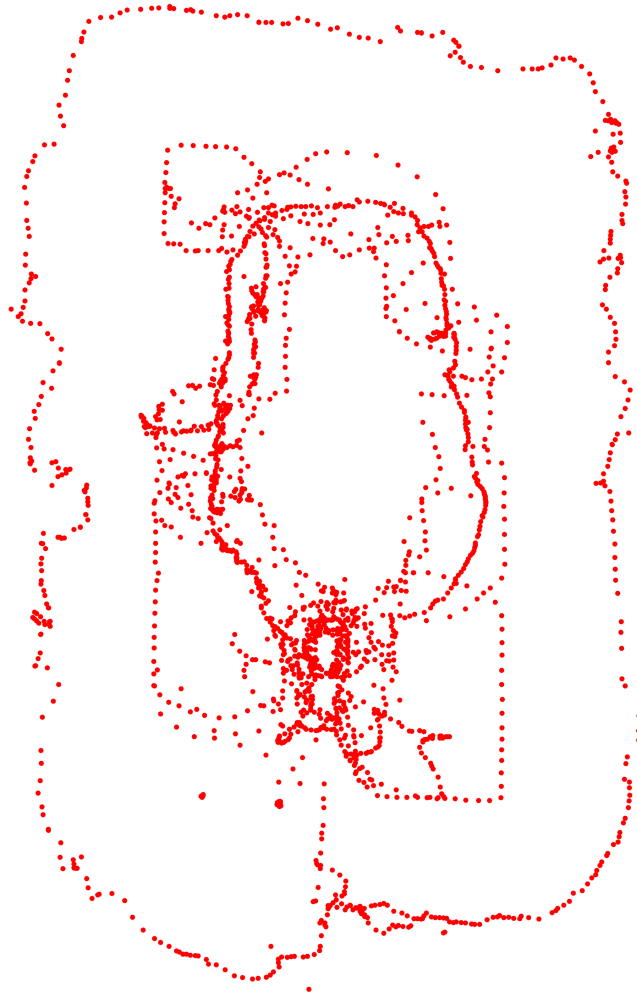
Pipelines

- Start with pair of images
- Add individual cameras, verify, refine by BA
- Discard problematic images/features (robustness)
- Relatively efficient, no recover from errors
- Many heuristics and implementation issues
- Bundler, VSFM, COLMAP, Theia, OpenMVG

Figure: SfM reconstruction of the Vitthala temple complex in Hampi.

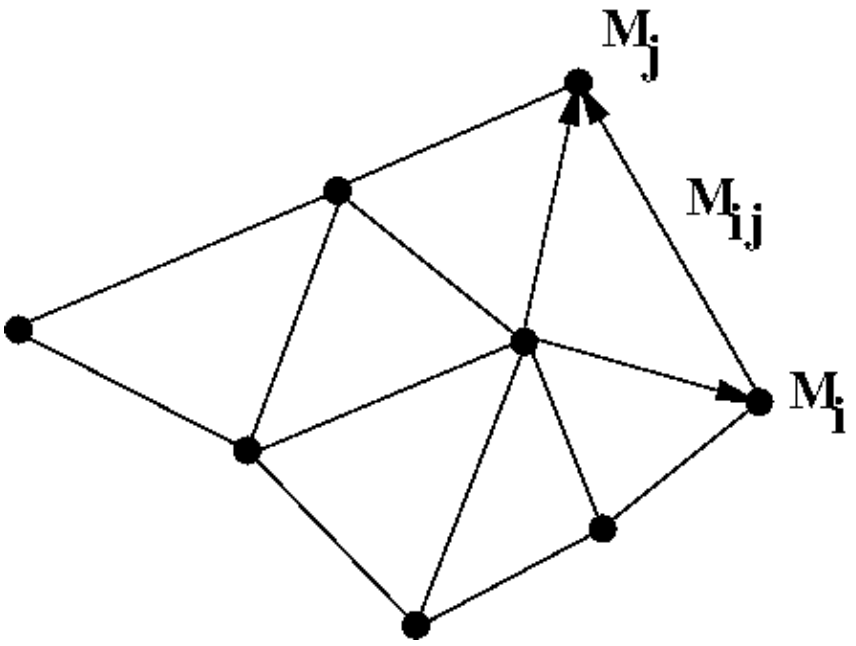


(a) Plan View of 3D Reconstruction



(b) Cameras in the viewgraph

Global Methods



$$\boldsymbol{M}_{ij} = \boldsymbol{M}_j \boldsymbol{M}_i^{-1}$$

Analogous to vector form

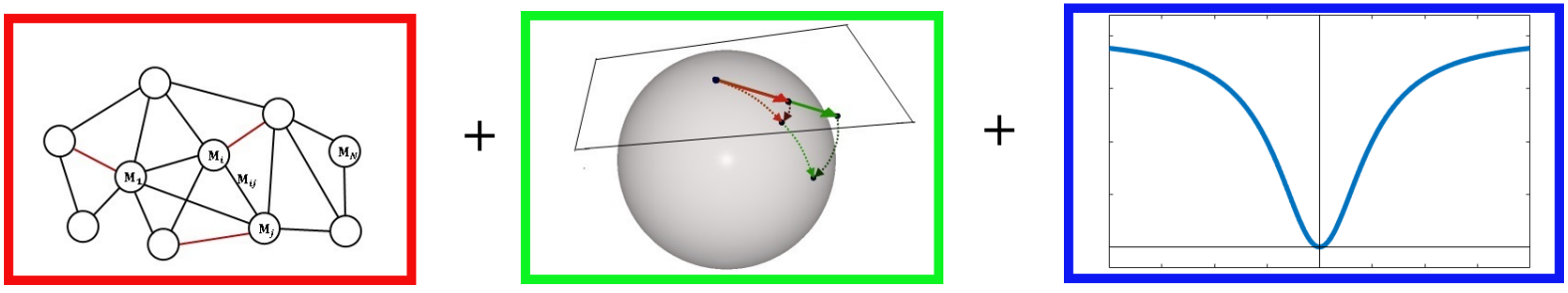
$$\boldsymbol{m}_{ij} = \boldsymbol{m}_j - \boldsymbol{m}_i$$

Motion Averaging

- Solve for all cameras at once
- Factor out 3D structure variables
- Pairwise motion estimates are cheap
- Redundancy in camera viewgraph
- Efficient, accurate and robust
- Standard approach in recent years

Motion Averaging

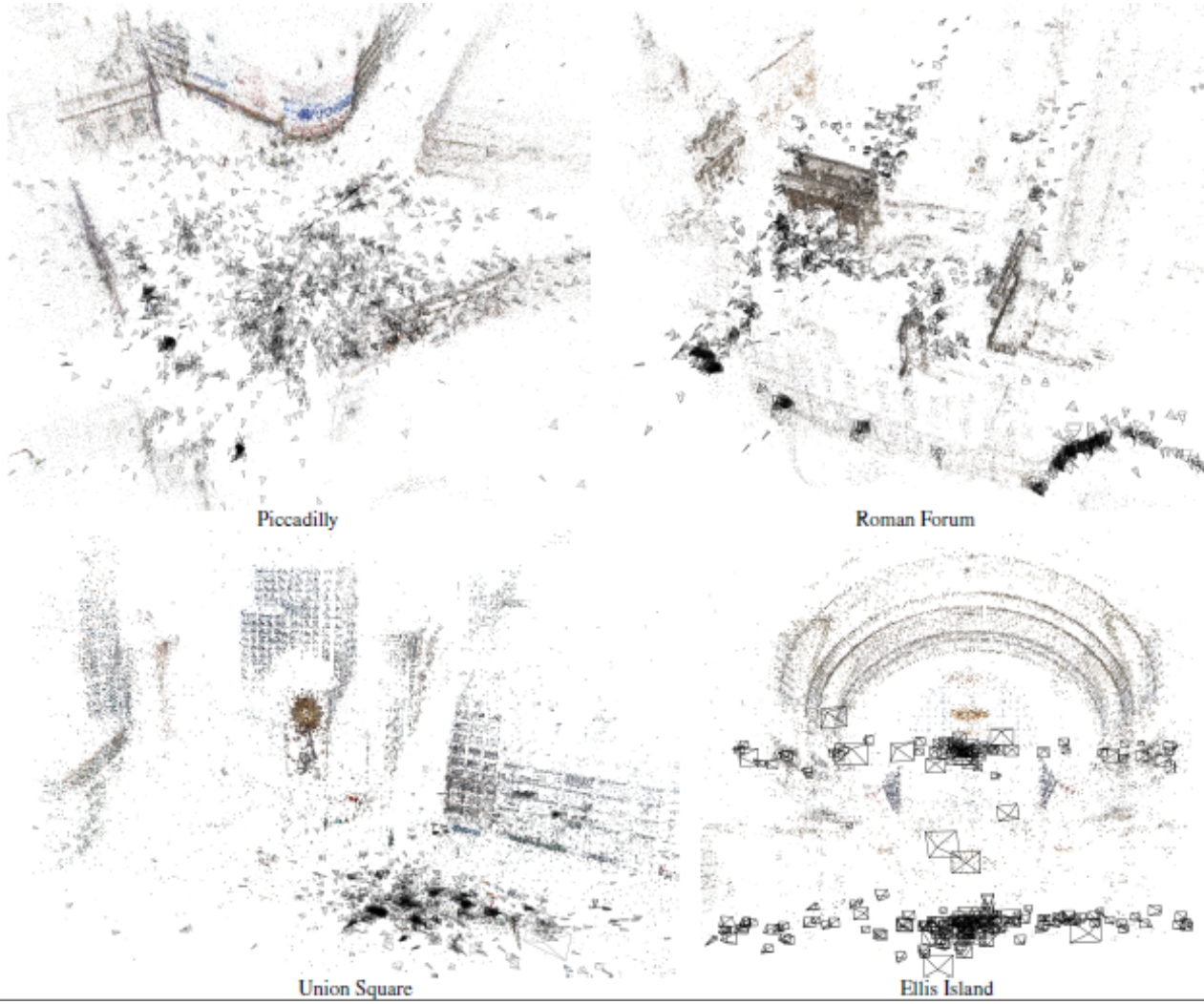
$$\sum_{\mathcal{E}} \rho\left(d\left(\boldsymbol{R}_{ij}, \boldsymbol{R}_j \boldsymbol{R}_i^{-1}\right)\right)$$



Global Methods: Motion Averaging

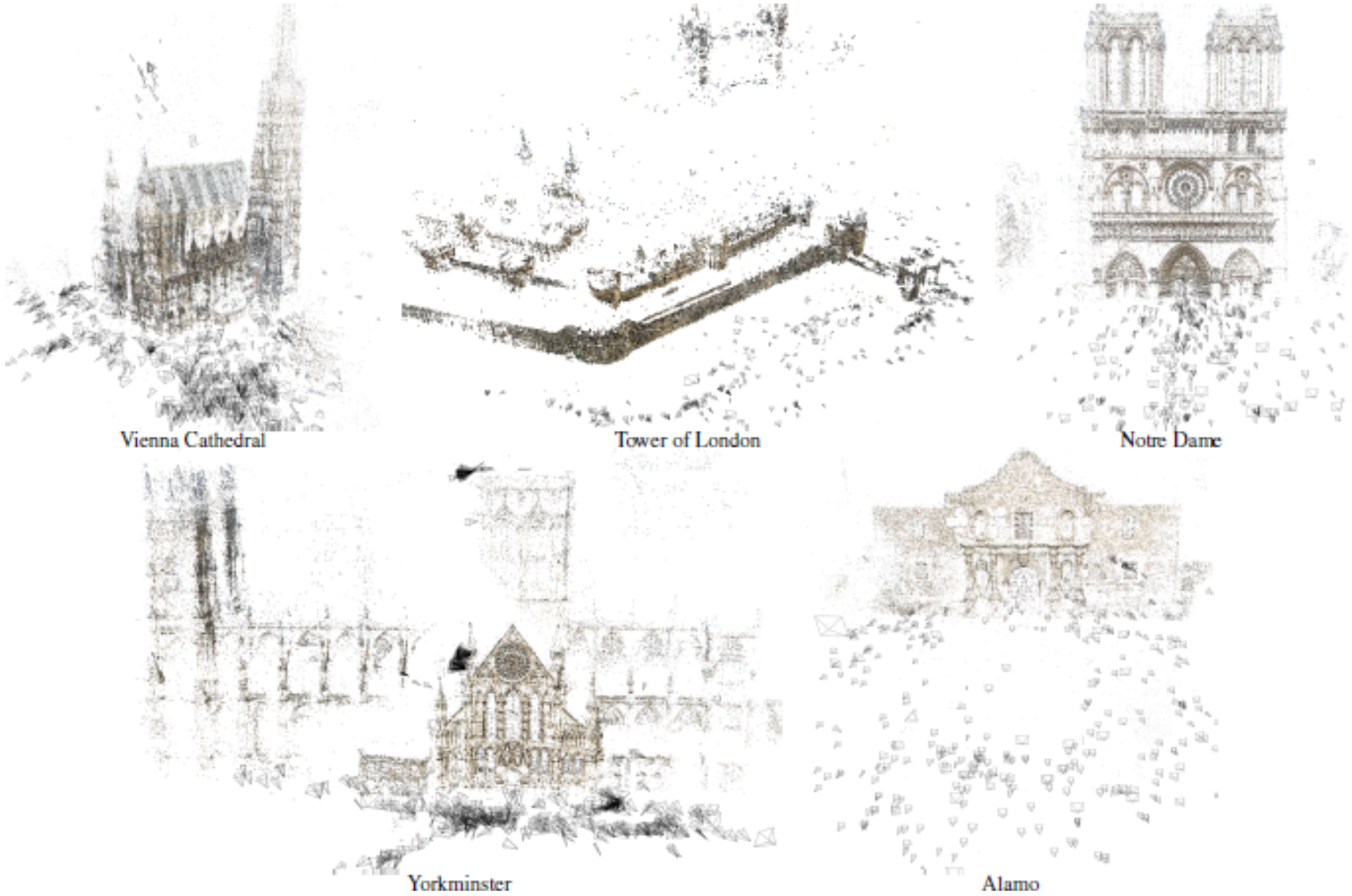
- Key ingredients
  - Graph structure $\Leftrightarrow$ camera-camera relationships
  - Riemannian Manifold (Camera motions are Lie Groups)
  - Robustness (Real data is always corrupted)

Motion Averaging



Results from Wilson *et al.* “Robust Global Translation with 1DSfM”

Motion Averaging



Results from Wilson *et al.* "Robust Global Translation with 1DSfM"

Hierarchical SfM

(a) Architectural plan view of the Vitthala temple, Hampi.

(b) Reconstruction of Vitthala temple, Hampi

(c) Reconstruction overlaid on Google map.

Figure 9: Reconstruction of Vitthala temple, Hampi.

Results from Bhowmick *et al.* CVIU 2017

Hierarchical SfM

Results from Bhowmick *et al.* CVIU 2017