n View Geometry

Computer Vision II

CSE 252B

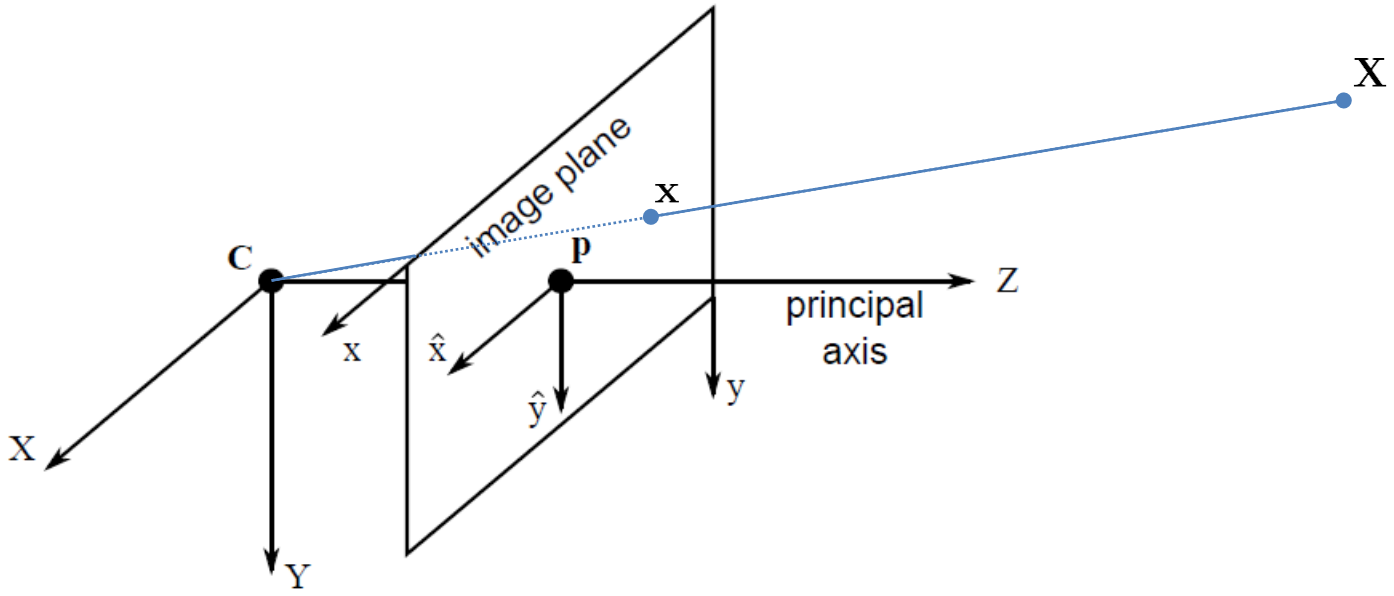
Lecture 17

Announcements

- Final exam, Mar 16, 7:00 PM-9:59 PM
  - Only for CSE graduate students taking the course-hosted comprehensive exam
- Assignment 5 is due Mar 18, 11:59 PM
- Please complete course and TA evaluations
  - Must be completed by Mar 14, 8:00 AM
- Reading
  - Sections 12.1, 12.2, 12.3, and 18.1

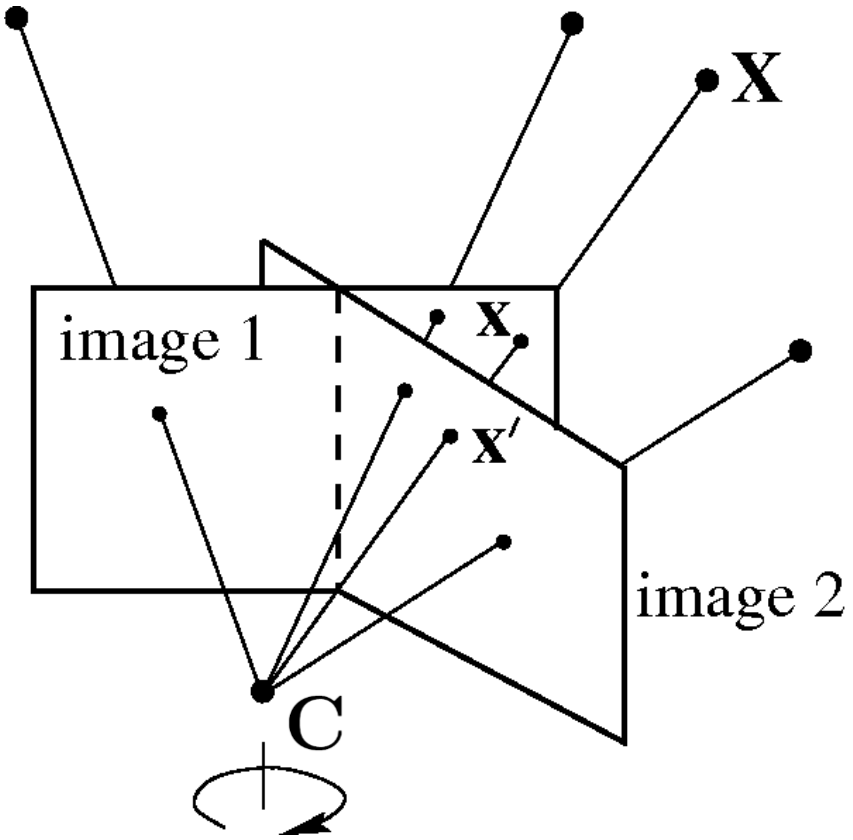
Single view geometry

- Imaging a scene



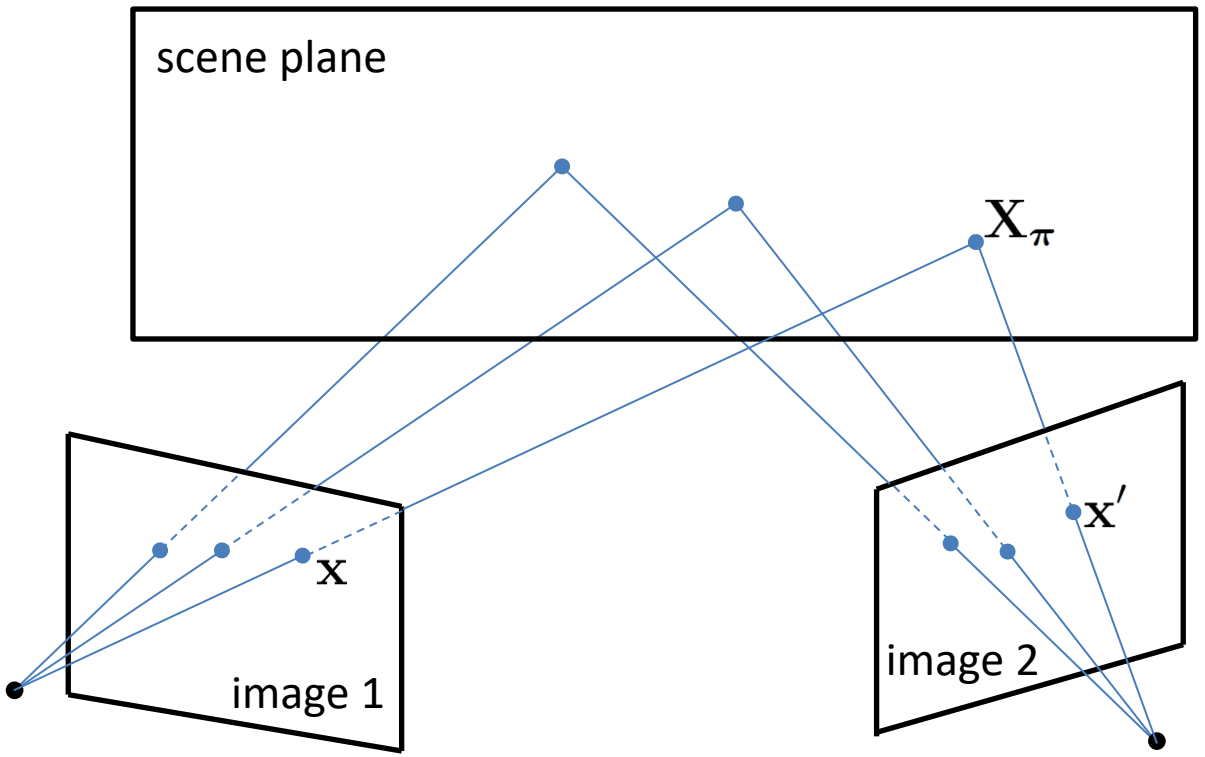
Two view geometry

- Rotation about the same camera center



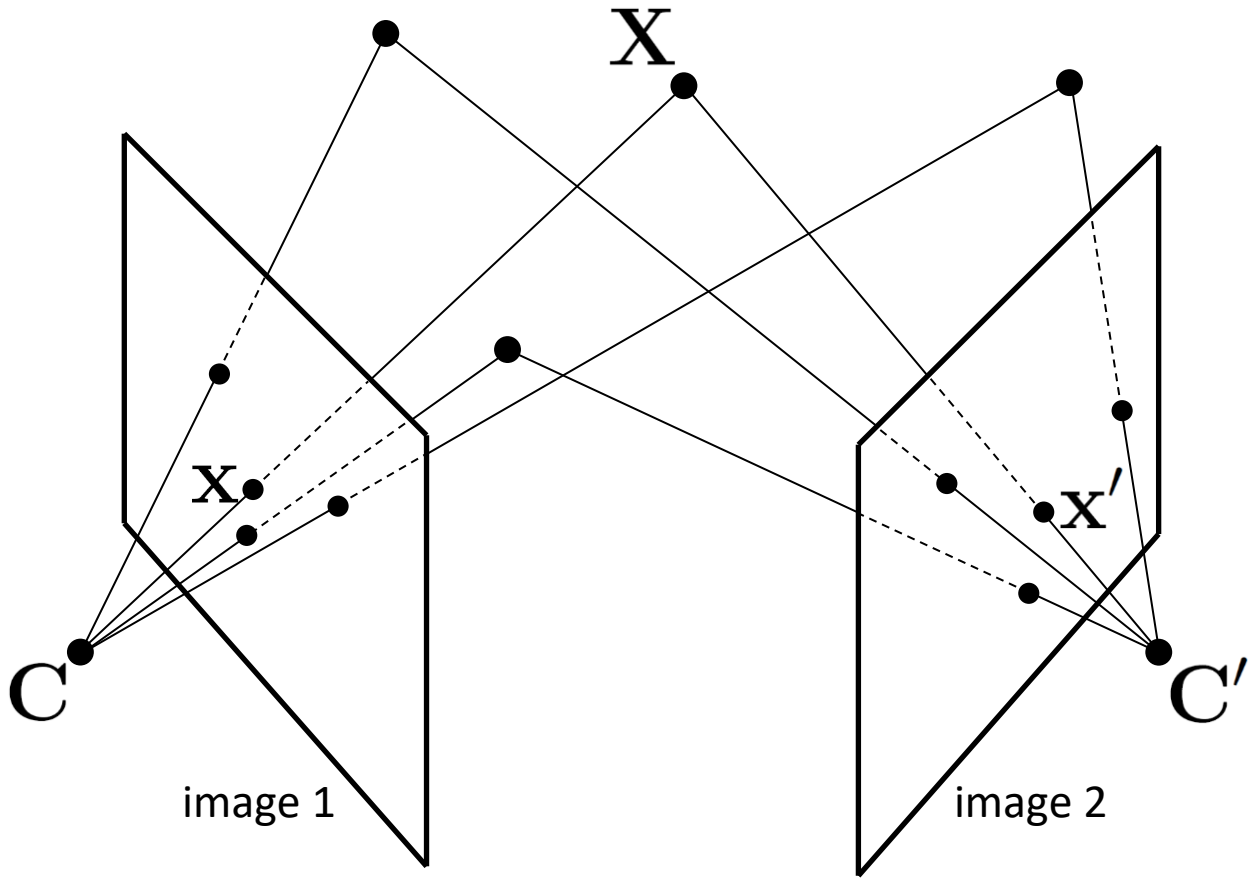
Two view geometry

- Imaging a plane



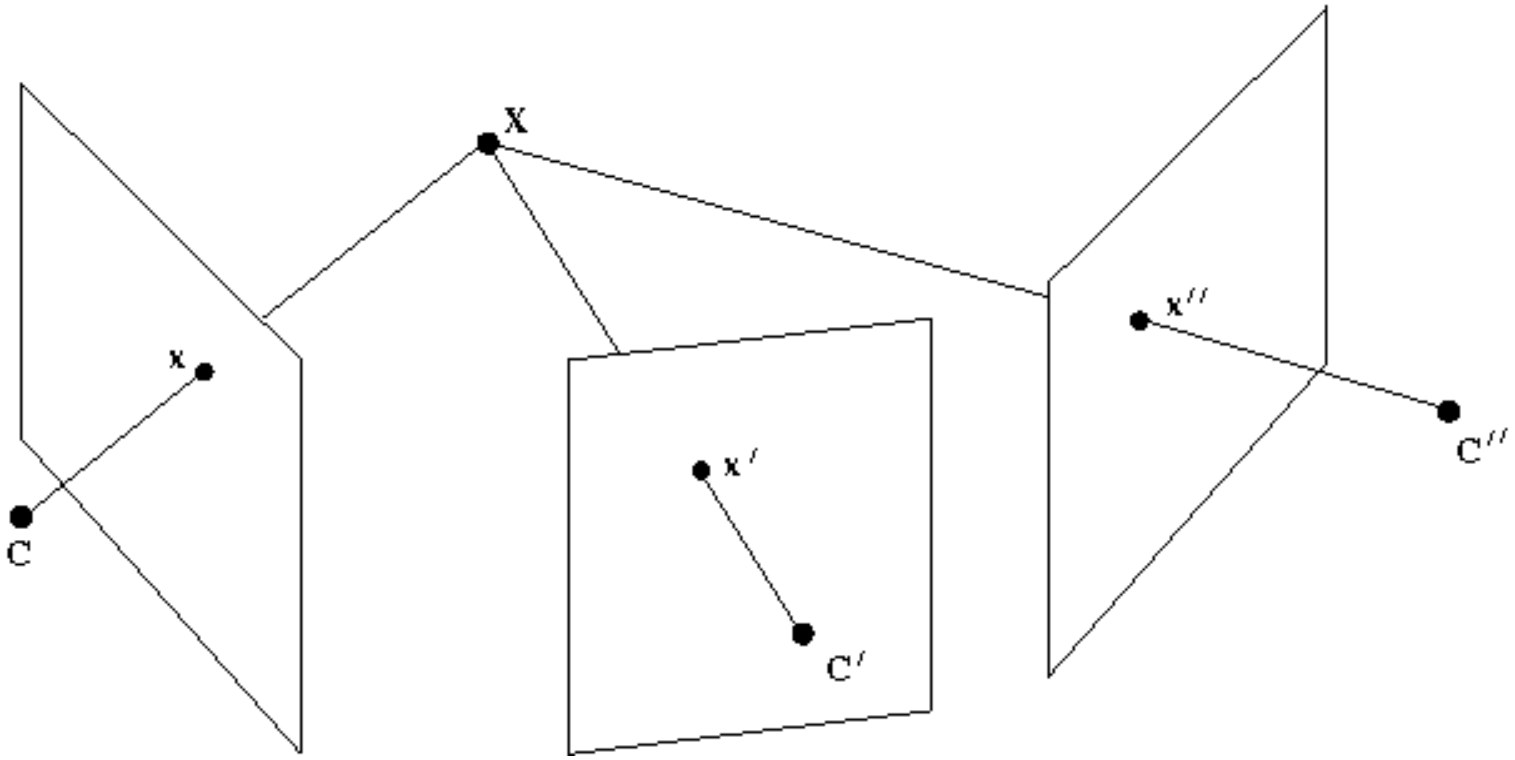
Two view geometry

- Imaging a 3D scene



Three view geometry

- Imaging a scene



Imaging geometry models

- Single view
  - Uncalibrated: camera projection matrix
    - 12 elements, 11 degrees of freedom
  - Calibrated: normalized camera projection matrix
    - 12 elements, 6 degrees of freedom
- Two views
  - Rotation about the same camera center
    - Uncalibrated: 2D projective transformation matrix
      - 9 elements, 8 degrees of freedom
    - Calibrated: 3D rotation matrix
      - 9 elements, 3 degrees of freedom

Imaging geometry models

- Two views
  - Imaging a plane
    - Uncalibrated: 2D projective transformation matrix
      - 9 elements, 8 degrees of freedom
    - Calibrated: 2D projective transformation matrix
      - 9 elements, 8 degrees of freedom
  - Imaging a 3D scene
    - Uncalibrated: fundamental matrix
      - 9 elements, 7 degrees of freedom
    - Calibrated: essential matrix
      - 9 elements, 5 degrees of freedom

Imaging geometry models

- Three views
  - Uncalibrated: trifocal tensor
    - 27 elements, 18 degrees of freedom
  - Calibrated: calibrated trifocal tensor
    - 27 elements, 11 degrees of freedom

Not covered in CSE 252B

Imaging geometry models

- Four views
  - Uncalibrated
    - Quadrifocal tensor ($3 \times 3 \times 3 \times 3$ tensor)
      - 81 elements, 29 degrees of freedom
    - Four camera projection matrices
      - 48 elements, 29 degrees of freedom
  - Calibrated
    - Calibrated quadrifocal tensor ($3 \times 3 \times 3 \times 3$ tensor)
      - 81 elements, 17 degrees of freedom
    - Four normalized camera projection matrices
      - 48 elements, 17 degrees of freedom

Not covered in CSE 252B

Not covered in CSE 252B

In practice, due to the additional elements, the quadrifocal tensor is not used

Imaging geometry models

- n views
  - Uncalibrated
    - n camera projection matrices
      - $12n$ elements, $11n$ - 15 degrees of freedom

3D projective transformation

  - Calibrated
    - n normalized camera projection matrices
      - $12n$ elements, $6n$ - 7 degrees of freedom

3D similarity transformation

n views, uncalibrated, estimation

- Parameters
  - n camera projection matrices
    - $12n$ elements, $11n$ - 15 parameters
  - m 3D scene points
    - $4m$ elements, $3m$ parameters
- Measurements
  - m 2D point correspondences over n views
    - $2mn$ measurements
- Solution when $3m + 11n - 15 \leq 2mn$

3D projective transformation

Example: 4 views requires at least 6 point correspondences, $47 \leq 48$

n views, calibrated, estimation

- Parameters
  - n normalized camera projection matrices
    - $12n$ elements, $6n$ - 7 parameters
  - m 3D scene points
    - $4m$ elements, $3m$ parameters
- Measurements
  - m 2D point correspondences over n views
    - $2mn$ measurements
- Solution when $3m + 6n - 7 \leq 2mn$

3D similarity transformation

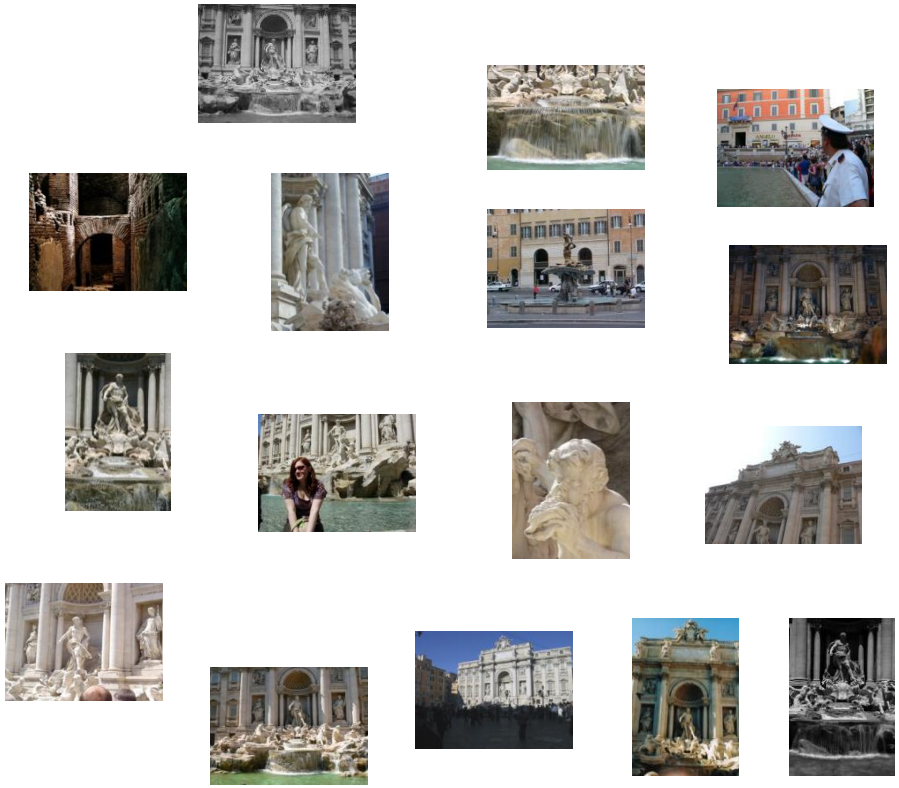
Example: 4 views requires at least 4 point correspondences, $29 \leq 32$

Approaches for n view reconstruction of scene structure and cameras

1. Factorization See textbook for details
   - Requires **all** m 3D scene points are visible in **all** n views
2. Alternative, general approach
   - Establish 2D point correspondences between images or video frames
   - Compute an initial reconstruction
   - Nonlinear optimization

n view reconstruction of scene structure and cameras

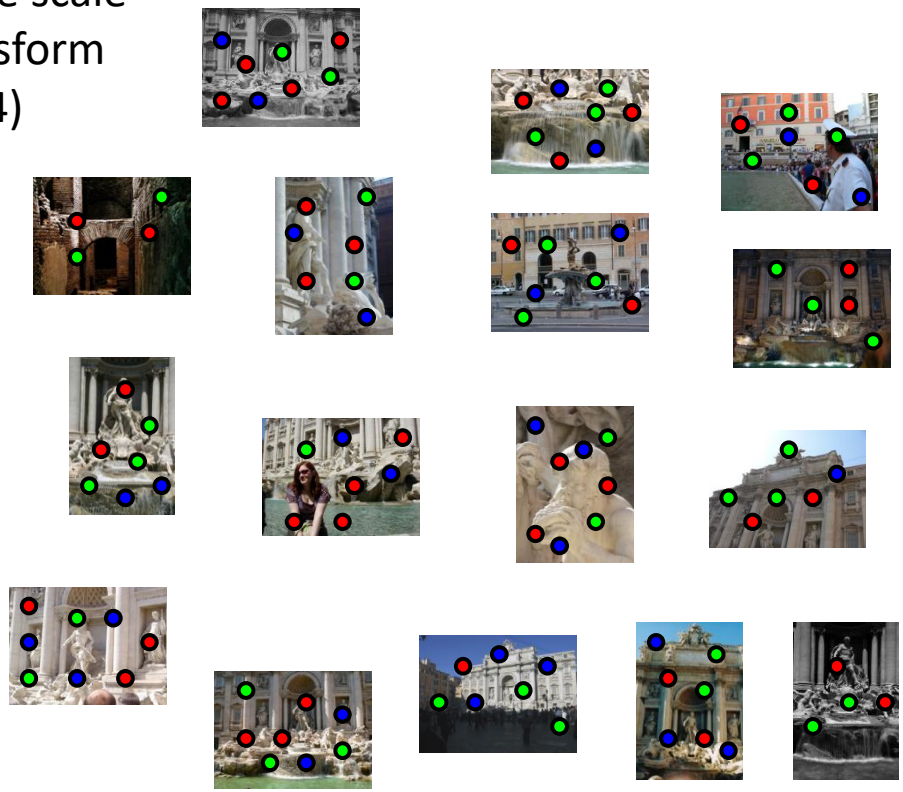
- Given a set of images of a scene, following is one approach (of many)



n view reconstruction of scene structure and cameras

- Feature detection and feature descriptors

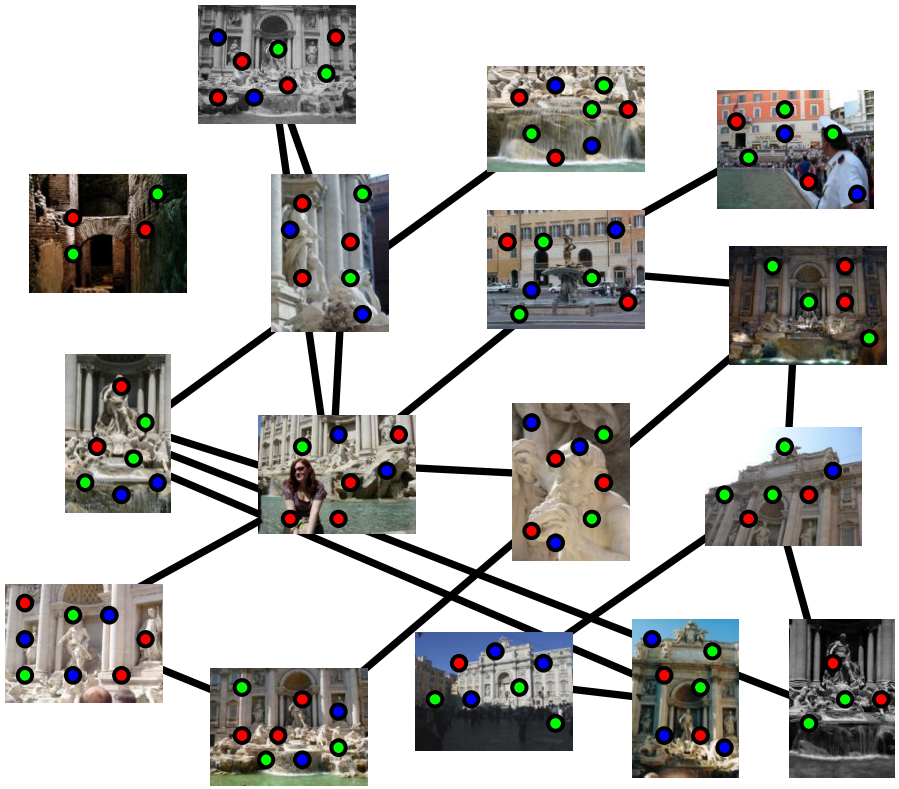
For example, using the scale invariant feature transform (SIFT; Lowe, IJCV 2004)



n view reconstruction of scene structure and cameras

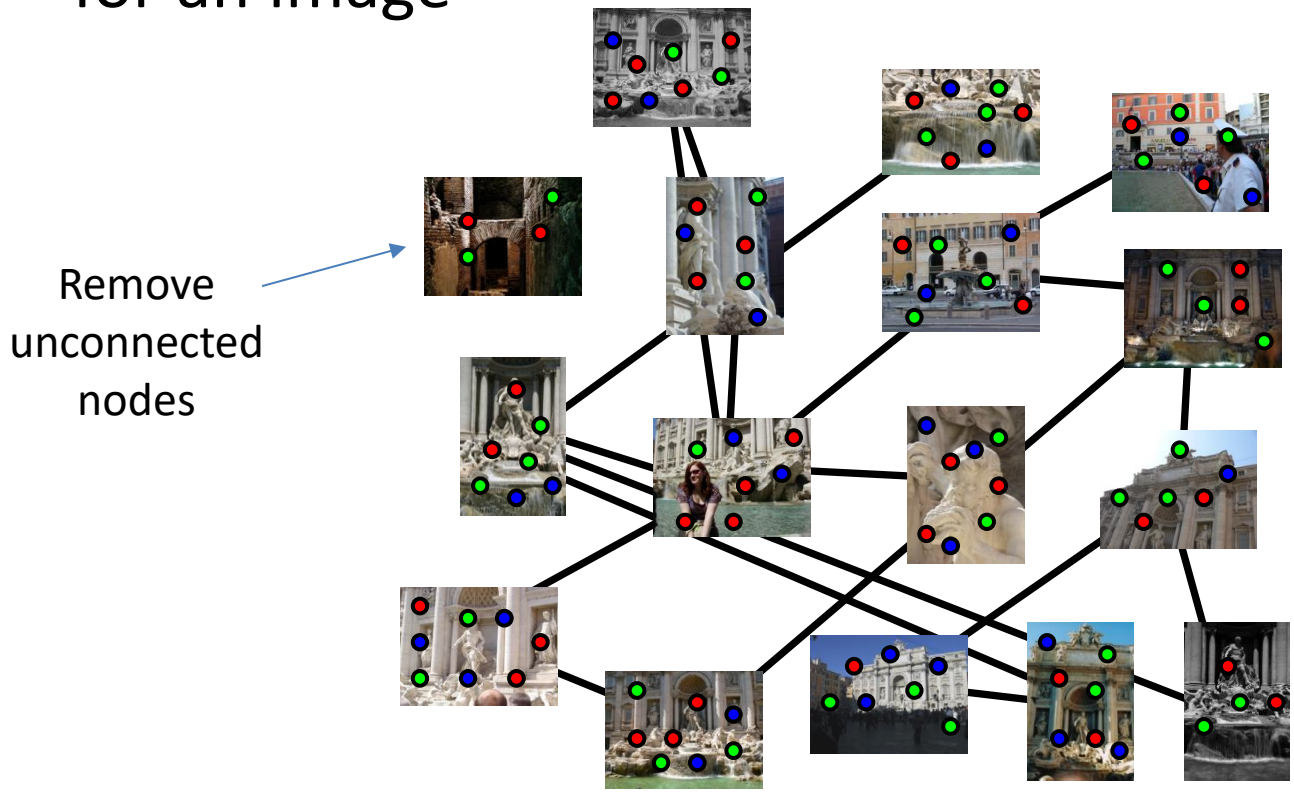
- Match feature between each pair or (better) triplet of images

Use MSAC



n view reconstruction of scene structure and cameras

- Create a graph where each node is the set of points for an image



n view reconstruction of scene structure and cameras

- Given a video of a scene, following is one approach (of many)
- Detect and track features over time
  - Use MSAC for outlier rejection over consecutive triplets of video frames



n view reconstruction of scene structure and cameras

- Create a graph where each node is the set of points for an image (or video frame)
- Determine the minimum spanning tree (MST), preferring edges with more feature correspondences
- Determine relative imaging geometry between the MST node pairs (fundamental or essential matrix) or (better) triplets (trifocal tensor or calibrated trifocal tensor)
- Transform resulting scene points and cameras such that they are all in the same (relative) coordinate frame (e.g., relative to one of the cameras)

n view reconstruction of scene structure and cameras

- Resection(ing): (linear) estimate each camera (camera projection matrix or normalized camera projection matrix) associated with a node **not** in the MST
- Triangulation: (linear) estimate initial 3D scene points using all 2D image point correspondences and their associated cameras (estimated above)
- Bundle adjustment: nonlinear optimization, simultaneously adjusting all parameters of all cameras and all 3D scene points

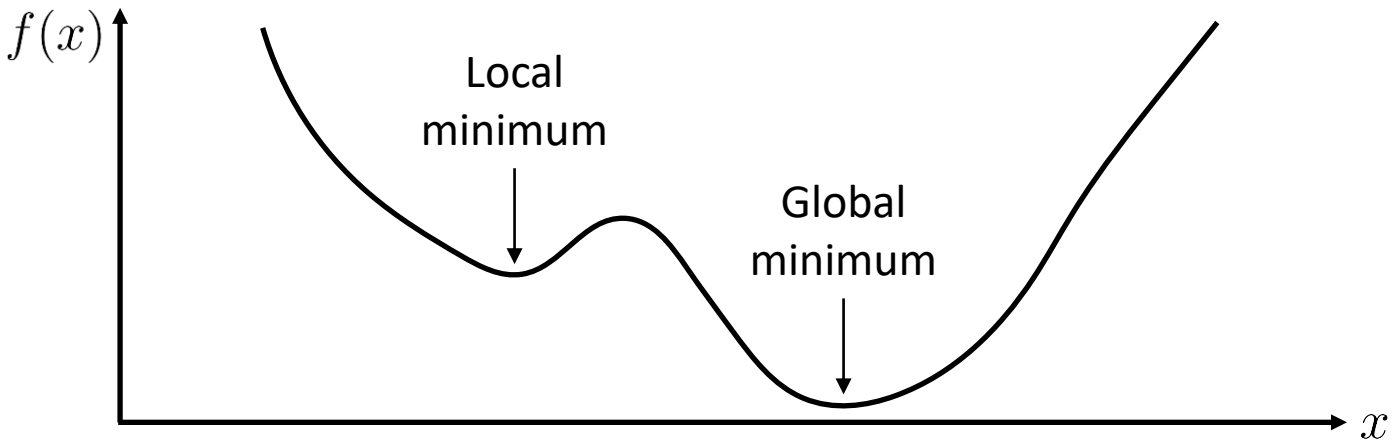
Triangulation

Triangulation

for uncalibrated cameras, but easily modified for calibrated cameras

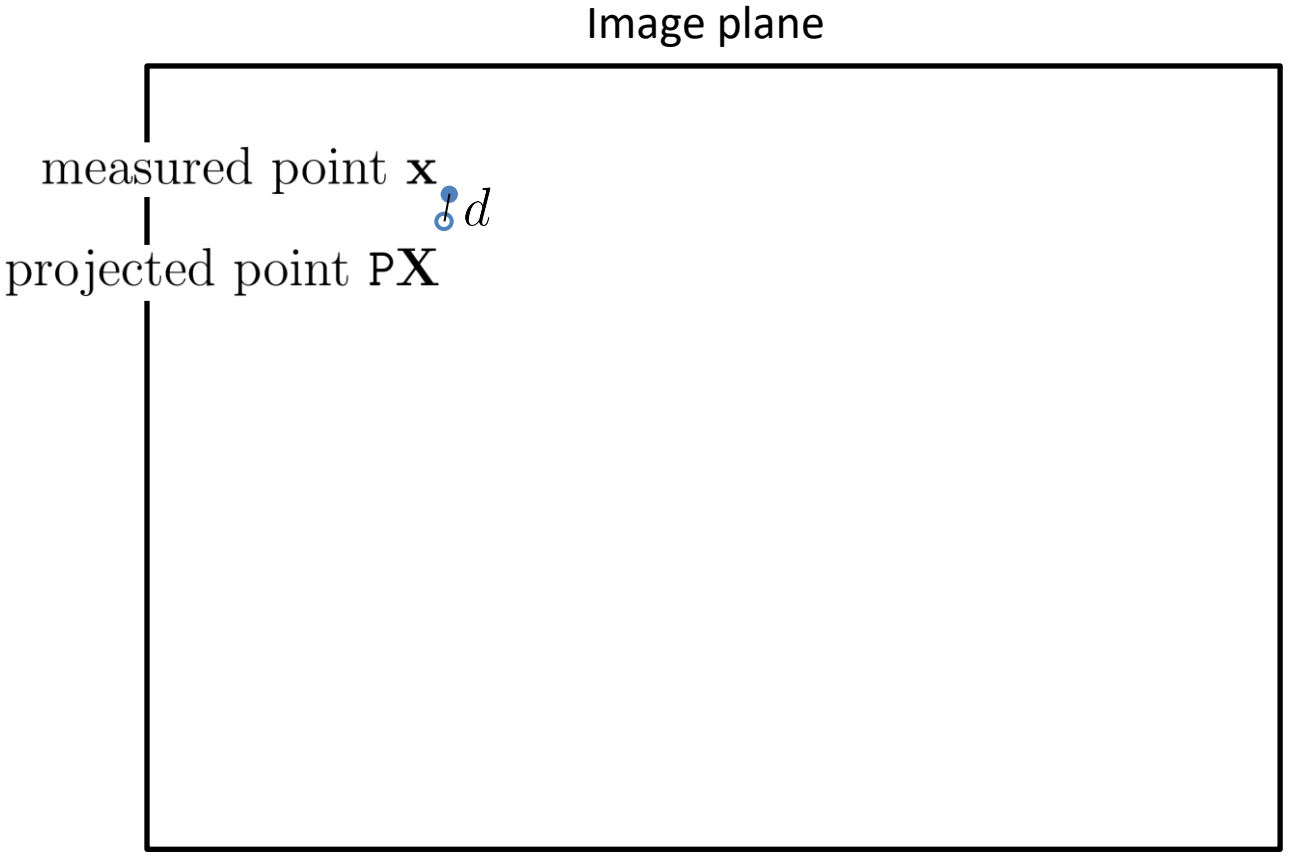
$$\mathbf{x}^{(j)} = \mathrm{P}^{(j)}\mathbf{X} \,\forall\, j \text{ where } \mathbf{X} \text{ is visible, solve for } \mathbf{X}$$

- Given image point correspondences over multiple views
- Minimize geometric error $\sum_j d(\mathbf{x}^{(j)}, \mathrm{P}^{(j)}\mathbf{X})^2$
- A nonlinear mapping requires a nonlinear optimization problem solver
  - Use linear estimation for initial estimate (to mitigate converging to local optimum)
  - Iterative process to determine global optimum



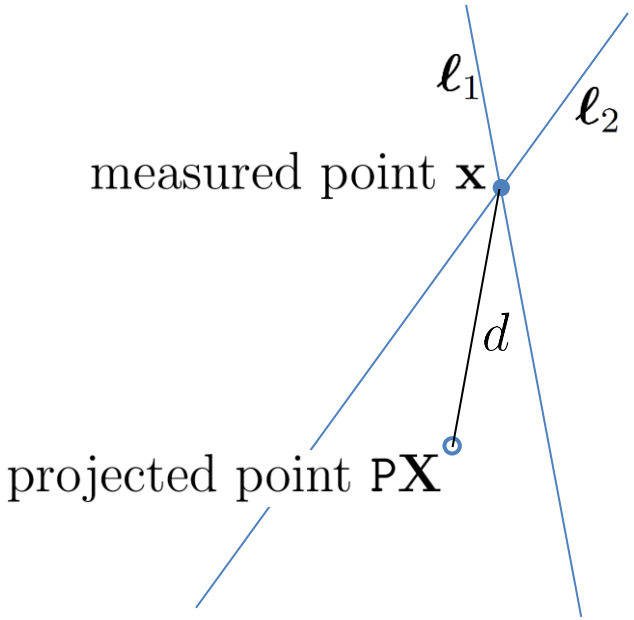
Triangulation

- Error in image point measurements



Linear estimation of 3D point using the direct linear transformation (DLT) algorithm

- Minimize algebraic error between the projected point and two lines intersecting at the measured point



Points and hyperplanes

- Two 2D lines define a 2D point

2D point at intersection of two 2D lines in general position is (right) null space of matrix

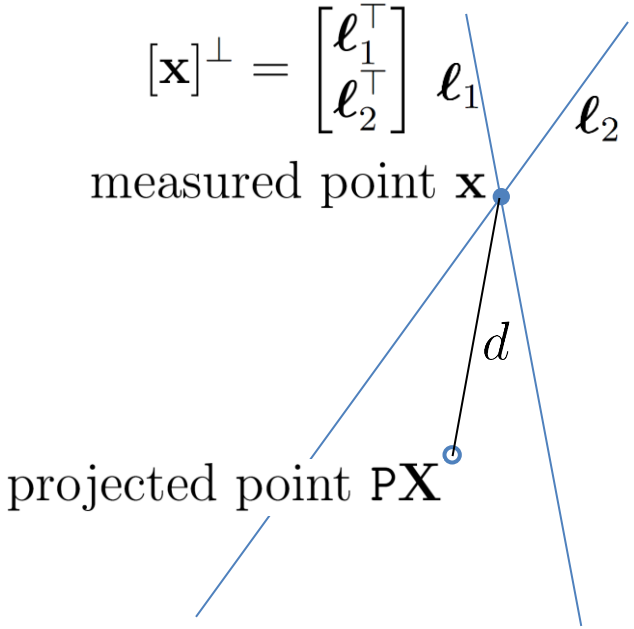
$$\begin{bmatrix} \boldsymbol{\ell}_1^\top \\ \boldsymbol{\ell}_2^\top \end{bmatrix} \mathbf{x} = \mathbf{0}$$

Two 2D lines intersecting at a 2D point is the left null space of 2D point

$$[\mathbf{x}]^\perp = \begin{bmatrix} \boldsymbol{\ell}_1^\top \\ \boldsymbol{\ell}_2^\top \end{bmatrix}$$

Linear estimation of 3D point using the direct linear transformation (DLT) algorithm

- Minimize algebraic error between the projected point and two lines intersecting at the measured point



$$[\mathbf{x}]^\perp \mathbf{x} = \mathbf{0} \qquad \mathbf{x} = \mathtt{P}\mathbf{X}$$
$$[\mathbf{x}]^\perp \mathtt{P}\mathbf{X} = \mathbf{0}$$

Linear estimation of 3D point using the direct linear transformation (DLT) algorithm

Given $n \geq 2$ point correspondences over n views where $\mathbf{X}$ is visible

$$\mathbf{x}^{(j)} = \mathtt{P}^{(j)}\mathbf{X} \,\forall\, j \text{ where } \mathbf{X} \text{ is visible, solve for } \mathbf{X}$$

$$[\mathbf{x}^{(j)}]^{\perp}\mathtt{P}^{(j)}\mathbf{X} = \mathbf{0} \,\forall\, j$$

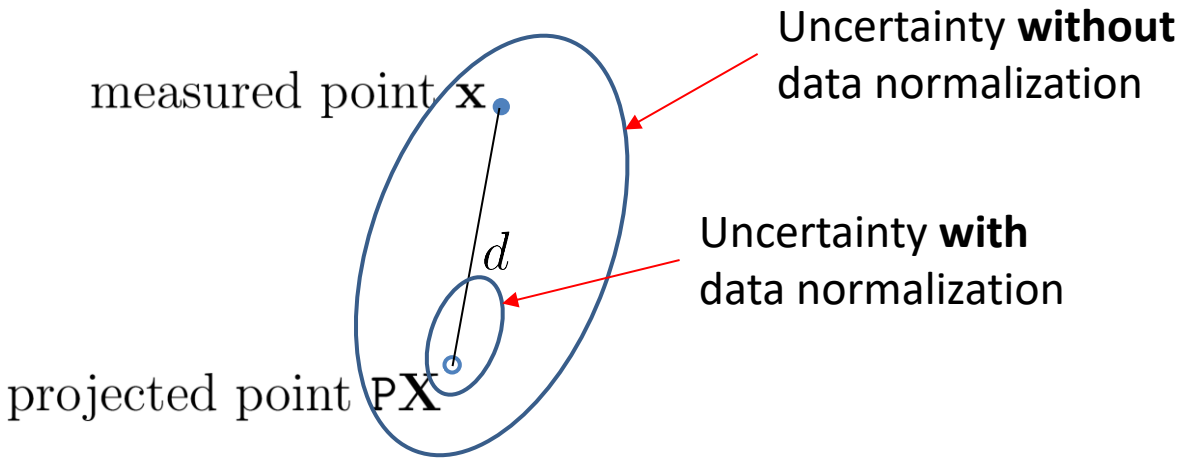
$$\begin{bmatrix} [\mathbf{x}^{(1)}]^{\perp}\mathtt{P}^{(1)} \\ \vdots \\ [\mathbf{x}^{(n)}]^{\perp}\mathtt{P}^{(n)} \end{bmatrix} \mathbf{X} = \mathbf{0}$$

$$\mathtt{A}\mathbf{X} = \mathbf{0}, \text{ where } \mathtt{A} = \begin{bmatrix} [\mathbf{x}^{(1)}]^{\perp}\mathtt{P}^{(1)} \\ \vdots \\ [\mathbf{x}^{(n)}]^{\perp}\mathtt{P}^{(n)} \end{bmatrix}$$

But, data normalization must be used

Data normalization

- Data normalization is required to reduce the propagated uncertainty under the nonlinear projection



Data normalizing transformations

- 2D

Determine the similarity transformation $\mathtt{T}$ such that the mean (i.e., centroid) of the transformed points is at the origin and their standard deviation from the origin is $\sqrt{2}$.

$$\mathtt{T} = \begin{bmatrix} s & 0 & -s\mu_{\tilde{x}} \\ 0 & s & -s\mu_{\tilde{y}} \\ 0 & 0 & 1 \end{bmatrix}, \text{ where } s = \sqrt{\frac{2}{\sigma_{\tilde{x}}^2 + \sigma_{\tilde{y}}^2}}$$

where $\mu_{\tilde{x}}$ and $\sigma_{\tilde{x}}^2$, and $\mu_{\tilde{y}}$ and $\sigma_{\tilde{y}}^2$ are the mean and variance of the $\tilde{x}$ and $\tilde{y}$ coordinates, respectively.

Linear estimation of 3D point using the direct linear transformation (DLT) algorithm

- Algorithm

1. Data normalize the points

$$\mathbf{x}_{\mathrm{DN}}^{(j)} = \mathtt{T}\mathbf{x}^{(j)} \; \forall \, j$$

   and camera projection matrices

$$\mathtt{P}_{\mathrm{DN}}^{(j)} = \mathtt{T}\mathtt{P}^{(j)}$$

2. Estimate the 3D point $\mathbf{X}$ from the data normalized points and data normalized camera projection matrices. (No data denormalization.)

Nonlinear estimation using the Levenberg-Marquardt algorithm

Given

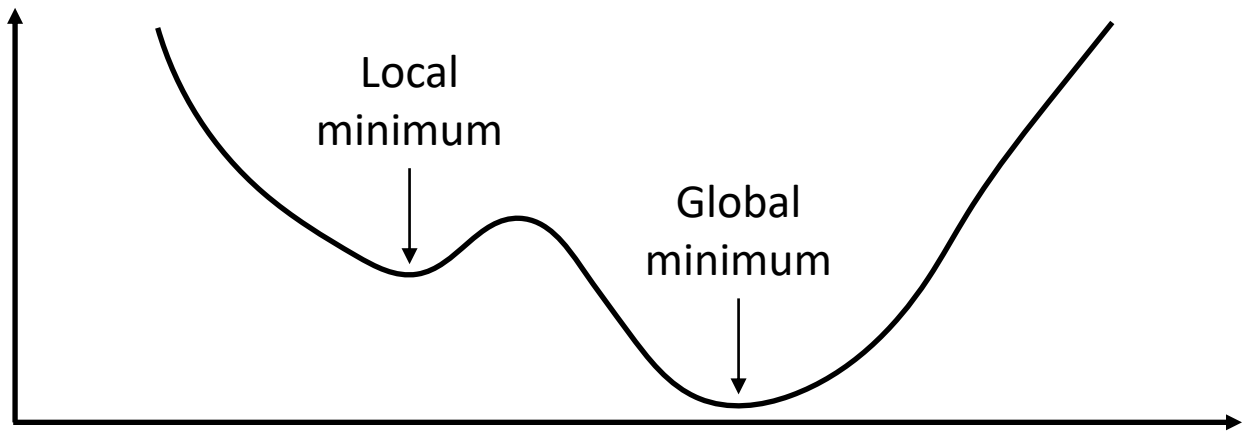
Measurement vector $\mathbf{X}$ with associated covariance matrix $\Sigma_{\mathbf{X}}$

Parameter vector $\hat{\mathbf{P}}$ (initial estimate)

Nonlinear function f, where $\hat{\mathbf{X}} = f(\hat{\mathbf{P}})$

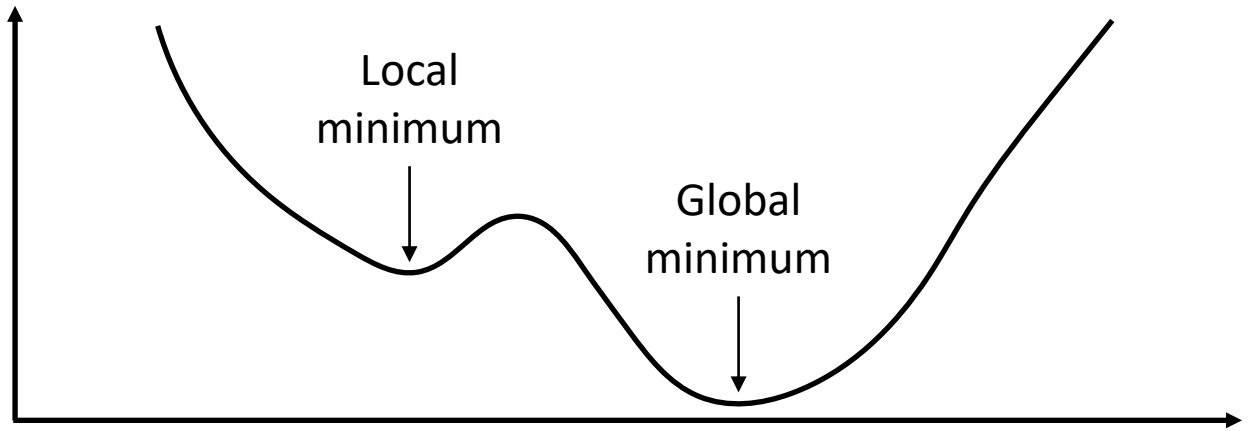
Objective

Find $\hat{\mathbf{P}}$ that minimizes $\boldsymbol{\epsilon}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}}$



Parameter vector

- May get stuck in local minimum of high dimensional space
- **Use minimal parameterizations towards mitigating this**



Parameterization of a homogeneous vector

Let the homogeneous vector $\bar{\mathbf{v}} = (a, \mathbf{b}^\top)^\top \in \mathbb{R}^n$, where $\|\bar{\mathbf{v}}\| = 1$ (i.e., $\bar{\mathbf{v}}$ is a unit vector), be parameterized as

$$\mathbf{v} = \frac{2}{\operatorname{sinc}(\cos^{-1}(a))}\mathbf{b} \in \mathbb{R}^{n-1}$$

then, if $\|\mathbf{v}\| > \pi$, normalized by

$$\mathbf{v} = \left(1 - \frac{2\pi}{\|\mathbf{v}\|}\left\lceil \frac{\|\mathbf{v}\| - \pi}{2\pi} \right\rceil\right)\mathbf{v}$$

Always use the parameterized homogeneous vector, not the input homogeneous vector or matrix

Parameterization of a homogeneous vector

The parameterized homogeneous vector $\mathbf{v}$ is deparameterized as the homogeneous vector

$$\bar{\mathbf{v}} = \left(\cos\left(\frac{\|\mathbf{v}\|}{2}\right), \frac{\operatorname{sinc}\left(\frac{\|\mathbf{v}\|}{2}\right)}{2}\mathbf{v}^\top \right)^\top \in \mathbb{R}^n$$

Unit vector with nonnegative first element

$$\bar{\mathbf{v}} = \left(a, \mathbf{b}^\top\right)^\top, \text{ where } a = \cos\left(\frac{\|\mathbf{v}\|}{2}\right) \text{ and } \mathbf{b} = \frac{\operatorname{sinc}\left(\frac{\|\mathbf{v}\|}{2}\right)}{2}\mathbf{v}$$

where $\|\bar{\mathbf{v}}\| = 1$ and a is nonnegative. For the deparameterization,

$$\frac{\partial \bar{\mathbf{v}}}{\partial \mathbf{v}} = \frac{\partial (a, \mathbf{b}^\top)}{\partial \mathbf{v}} = \begin{bmatrix} \frac{\mathrm{d}a}{\partial \mathbf{v}} \\ \frac{\partial \mathbf{b}}{\partial \mathbf{v}} \end{bmatrix} \in \mathbb{R}^{n \times (n-1)}$$

where

$$\frac{\mathrm{d}a}{\partial \mathbf{v}} = \begin{cases} \mathbf{0}^\top & \text{if } \|\mathbf{v}\| = 0 \\ -\frac{1}{2}\mathbf{b}^\top & \text{otherwise} \end{cases}$$

and

$$\frac{\partial \mathbf{b}}{\partial \mathbf{v}} = \begin{cases} \frac{1}{2}\mathtt{I} & \text{if } \|\mathbf{v}\| = 0 \\ \frac{\operatorname{sinc}\left(\frac{\|\mathbf{v}\|}{2}\right)}{2}\mathtt{I} + \frac{1}{4\|\mathbf{v}\|}\frac{\mathrm{d}\operatorname{sinc}\left(\frac{\|\mathbf{v}\|}{2}\right)}{\mathrm{d}\frac{\|\mathbf{v}\|}{2}}\mathbf{v}\mathbf{v}^\top & \text{otherwise} \end{cases}$$

Parameterization of a homogeneous vector

- Sinc function

The sinc function

$$\operatorname{sinc}(x) = \begin{cases} 1 & \text{if } x = 0 \\ \frac{\sin(x)}{x} & \text{otherwise} \end{cases}$$

The derivative is given by

$$\frac{\mathrm{d}\operatorname{sinc}(x)}{\mathrm{d}x} = \begin{cases} 0 & \text{if } x = 0 \\ \frac{\cos(x)}{x} - \frac{\sin(x)}{x^2} & \text{otherwise} \end{cases}$$

Nonlinear estimation of 3D point using the Levenberg-Marquardt algorithm

Parameter vector

3×1

$\hat{\mathbf{X}}$

Parameterization of homogeneous vector

Always use the parameterized homogeneous vector, not the input homogeneous vector or matrix

Mapping

$$\hat{\mathbf{x}}^{(j)} = \mathtt{P}^{(j)}\hat{\mathbf{X}} \; \forall \, j$$

Measurement vector

$2n \times 1$

$$\left(\tilde{\mathbf{x}}^{(1)\top}, \ldots, \tilde{\mathbf{x}}^{(n)\top}\right)^{\top} \text{ with associated covariance matrices } \Sigma_{\tilde{\mathbf{x}}^{(1)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}^{(n)}}$$

Cost

$$\sum_j \boldsymbol{\epsilon}_j^{\top} \Sigma_{\tilde{\mathbf{x}}^{(j)}}^{-1} \boldsymbol{\epsilon}_j, \text{ where } \boldsymbol{\epsilon}_j = \tilde{\mathbf{x}}^{(j)} - \hat{\tilde{\mathbf{x}}}^{(j)}$$

Jacobian

$$\mathtt{J} = \begin{bmatrix} \mathtt{A}_1 \\ \vdots \\ \mathtt{A}_n \end{bmatrix}, \text{ where } \mathtt{A}_j = \frac{\partial \hat{\tilde{\mathbf{x}}}^{(j)}}{\partial \hat{\mathbf{X}}} = \frac{\partial \hat{\tilde{\mathbf{x}}}^{(j)}}{\partial \bar{\mathbf{X}}} \frac{\partial \bar{\mathbf{X}}}{\partial \hat{\mathbf{X}}}$$

$2n \times 3$

2×4 $\quad$ 4×3

But, use data normalization

Projection of a point under the camera projection matrix

The homogeneous 3D point $\mathbf{X}$ is projected to the homogeneous 2D point $\mathbf{x}$ under the (homogeneous) camera projection matrix $\mathtt{P}$ by

$$\mathbf{x} = \mathtt{P}\mathbf{X}$$

Dehomogenizing the 2D point results in the mapping $\mathbf{X} \mapsto \tilde{\mathbf{x}}$. For this mapping

$$\frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{p}} = \frac{1}{w}\begin{bmatrix} \mathbf{X}^\top & \mathbf{0}^\top & -\tilde{x}\mathbf{X}^\top \\ \mathbf{0}^\top & \mathbf{X}^\top & -\tilde{y}\mathbf{X}^\top \end{bmatrix} \qquad 2 \text{ x } 12$$

and

$$\frac{\partial \tilde{\mathbf{x}}}{\partial \mathbf{X}} = \frac{1}{w}\begin{bmatrix} \mathbf{p}^{1\top} - \tilde{x}\mathbf{p}^{3\top} \\ \mathbf{p}^{2\top} - \tilde{y}\mathbf{p}^{3\top} \end{bmatrix} \qquad 2 \text{ x } 4$$

where $w = \mathbf{p}^{3\top}\mathbf{X}$ and $\mathbf{p}^{i\top}$ is the ith row of $\mathtt{P}$.

Nonlinear estimation of 3D point using the Levenberg-Marquardt algorithm

Normal equations matrix

$$\mathtt{U} = \sum_j \mathtt{A}_j^\top \Sigma_{\tilde{\mathbf{x}}^{(j)}}^{-1} \mathtt{A}_j \qquad 3 \text{ x } 3$$

Normal equations vector

$$3 \text{ x } 1 \quad \boldsymbol{\epsilon}_\mathbf{a} = \sum_j \mathtt{A}_j^\top \Sigma_{\tilde{\mathbf{x}}^{(j)}}^{-1} \boldsymbol{\epsilon}_j, \text{ where } \boldsymbol{\epsilon}_j = \tilde{\mathbf{x}}^{(j)} - \hat{\tilde{\mathbf{x}}}^{(j)}$$

Augmented normal equations

$$\mathtt{S} = \mathtt{U}^* = \mathtt{U} + \lambda \mathtt{I} \qquad 3 \text{ x } 3$$

$$\mathbf{e} = \boldsymbol{\epsilon}_\mathbf{a} \qquad 3 \text{ x } 1$$

$$\mathtt{S}\boldsymbol{\delta}_\mathbf{a} = \mathbf{e}, \text{ solve for } \boldsymbol{\delta}_\mathbf{a} \qquad 3 \text{ x } 1$$

Candidate parameter vector

$$\hat{\mathbf{X}}_0 = \hat{\mathbf{X}} + \boldsymbol{\delta}_\mathbf{a}$$

Adjustment

Normalize parameterization of a homogeneous vector after adjustment

Data normalization with covariance propagation

Determine the similarity transformation $\mathtt{T}$ such that the mean (i.e., centroid) of the transformed points is at the origin and their standard deviation from the origin is $\sqrt{2}$.

$$\mathtt{T} = \begin{bmatrix} s & 0 & -s\mu_{\tilde{x}} \\ 0 & s & -s\mu_{\tilde{y}} \\ 0 & 0 & 1 \end{bmatrix}, \text{ where } s = \sqrt{\frac{2}{\sigma_{\tilde{x}}^2 + \sigma_{\tilde{y}}^2}}$$

where $\mu_{\tilde{x}}$ and $\sigma_{\tilde{x}}^2$, and $\mu_{\tilde{y}}$ and $\sigma_{\tilde{y}}^2$ are the mean and variance of the $\tilde{x}$ and $\tilde{y}$ coordinates, respectively.

$$\mathbf{x}_{\text{DN}} = \mathtt{T}\mathbf{x}$$

$$\begin{bmatrix} \tilde{\mathbf{x}}_{\text{DN}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathtt{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}} \\ 1 \end{bmatrix}, \text{ where } \mathtt{A} = s\mathtt{I} \text{ and } \mathbf{t} = \begin{bmatrix} -s\mu_{\tilde{x}} \\ -s\mu_{\tilde{y}} \end{bmatrix}$$

$$\begin{bmatrix} \tilde{\mathbf{x}}_{\text{DN}} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathtt{A}\tilde{\mathbf{x}} + \mathbf{t} \\ 1 \end{bmatrix}$$

$$\tilde{\mathbf{x}}_{\text{DN}} = \mathtt{A}\tilde{\mathbf{x}} + \mathbf{t}$$

If unknown, then assume to be identity

Covariance propagation

$$\Sigma_{\tilde{\mathbf{x}}_{\text{DN}}} = \mathtt{J}\Sigma_{\tilde{\mathbf{x}}}\mathtt{J}^\top, \text{ where } \mathtt{J} = \frac{\partial \tilde{\mathbf{x}}_{\text{DN}}}{\partial \tilde{\mathbf{x}}} = \mathtt{A} = s\mathtt{I}$$

Nonlinear estimation of 3D point using the Levenberg-Marquardt algorithm

- Algorithm

1. Data normalize the points

$$\tilde{\mathbf{x}}_{\mathrm{DN}}^{(j)} = \mathtt{A}\tilde{\mathbf{x}}^{(j)} + \mathbf{t} \text{ and associated covariance matrices } \Sigma_{\tilde{\mathbf{x}}_{\mathrm{DN}}^{(j)}} = s^2 \Sigma_{\tilde{\mathbf{x}}^{(j)}}$$

and camera projection matrices

$$\mathtt{P}_{\mathrm{DN}}^{(j)} = \mathtt{T}\mathtt{P}^{(j)}$$

2. Estimate the 3D point $\mathbf{X}$ from the data normalized points and data normalized camera projection matrices. (No data denormalization.)

Bundle adjustment
(sparse Levenberg-Marquardt)

Nonlinear estimation using the Levenberg-Marquardt algorithm

Given

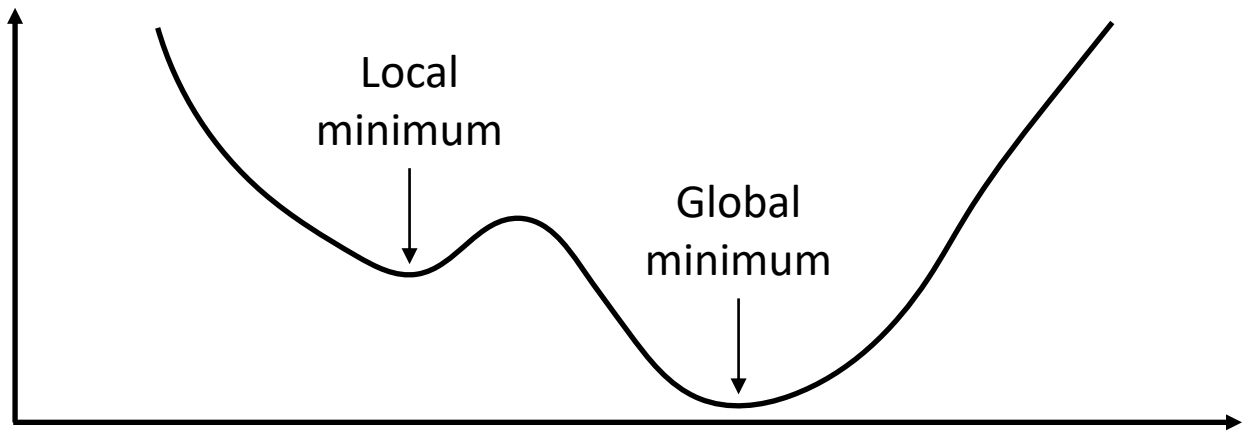
Measurement vector $\mathbf{X}$ with associated covariance matrix $\Sigma_{\mathbf{X}}$

Parameter vector $\hat{\mathbf{P}}$ (initial estimate)

Nonlinear function f, where $\hat{\mathbf{X}} = f(\hat{\mathbf{P}})$

Objective

Find $\hat{\mathbf{P}}$ that minimizes $\boldsymbol{\epsilon}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} = \mathbf{X} - \hat{\mathbf{X}}$



Sparse Levenberg-Marquardt

Parameter vector

$$\hat{\mathbf{P}} = \left(\mathbf{a}^{(1)\top}, \ldots, \mathbf{a}^{(m)\top}, \mathbf{b}_1^\top, \ldots, \mathbf{b}_n^\top\right)^\top$$

Mapping under nonlinear function f

$$\hat{\mathbf{X}} = f(\hat{\mathbf{P}})$$

Measurement vector

$$\mathbf{X} = \left(\mathbf{X}_1^{(1)\top}, \ldots, \mathbf{X}_n^{(1)\top}, \ldots, \mathbf{X}_1^{(m)\top}, \ldots, \mathbf{X}_n^{(m)\top}\right)$$

with associated covariance matrices $\Sigma_{\mathbf{X}_1^{(1)}}, \ldots, \Sigma_{\mathbf{X}_n^{(1)}}, \ldots, \Sigma_{\mathbf{X}_1^{(m)}}, \ldots, \Sigma_{\mathbf{X}_n^{(m)}}$

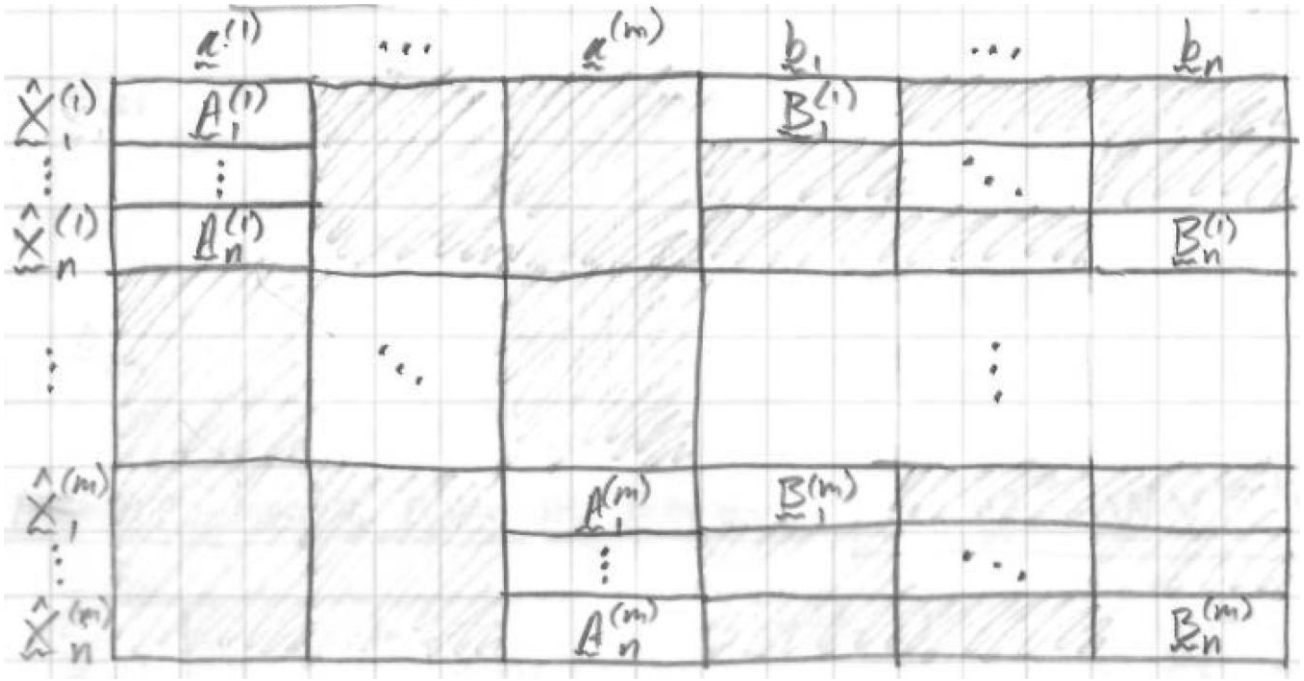
Cost

$$\sum_i \sum_j \boldsymbol{\epsilon}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \boldsymbol{\epsilon}_i^{(j)}, \text{ where } \boldsymbol{\epsilon}_i^{(j)} = \mathbf{X}_i^{(j)} - \hat{\mathbf{X}}_i^{(j)}$$

For missing data $\mathbf{X}_i^{(j)}$, omit $\Sigma_{\mathbf{X}_i^{(j)}}$ and $\boldsymbol{\epsilon}_i^{(j)}$

Sparse Levenberg-Marquardt

- Jacobian



$$\mathtt{A}_i^{(j)} = \frac{\partial \hat{\mathbf{X}}_i^{(j)}}{\partial \mathbf{a}^{(j)}}$$

$$\mathtt{B}_i^{(j)} = \frac{\partial \hat{\mathbf{X}}_i^{(j)}}{\partial \mathbf{b}_i}$$

For missing data $\mathbf{X}_i^{(j)}$, omit $\mathtt{A}_i^{(j)}$ and $\mathtt{B}_i^{(j)}$

Sparse Levenberg-Marquardt

- Normal equations

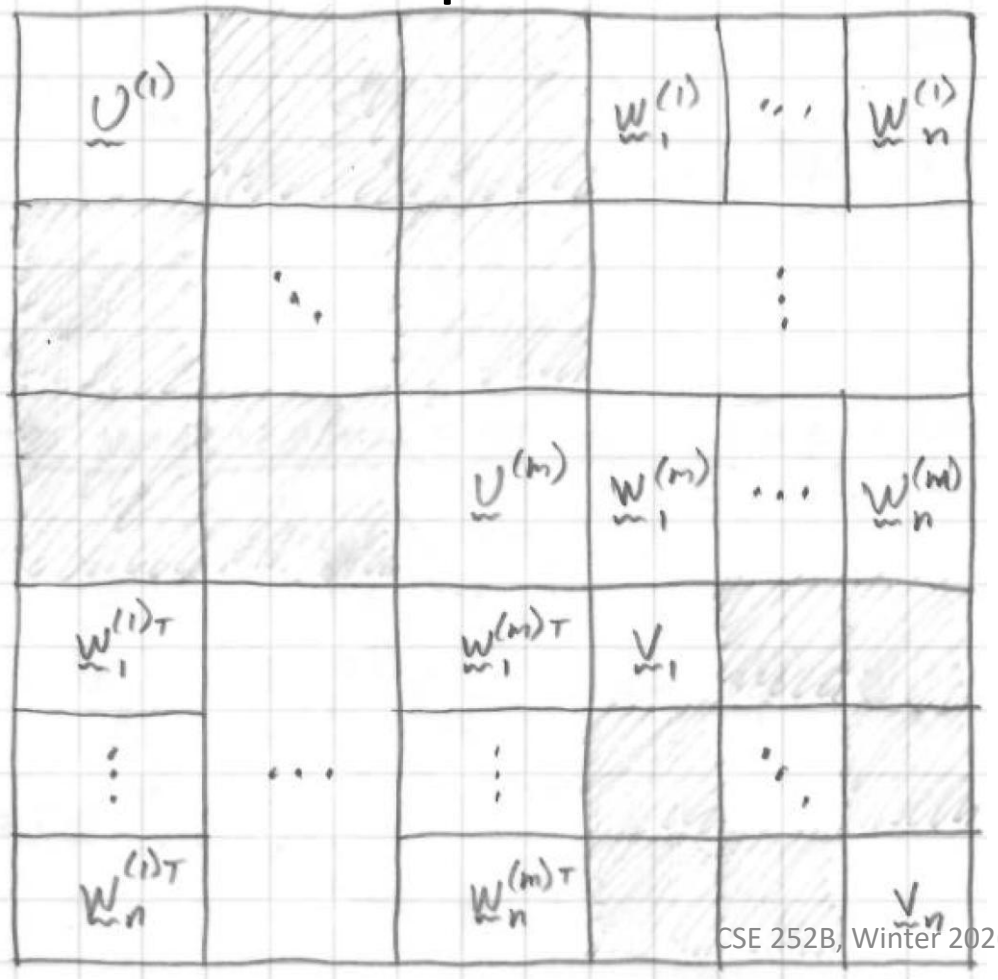
$$\left(\mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \mathtt{J}\right) \boldsymbol{\delta} = \mathtt{J}^\top \Sigma_{\mathbf{X}}^{-1} \boldsymbol{\epsilon}$$

$$\begin{bmatrix} \mathtt{U} & \mathtt{W}' \\ \mathtt{W}'^\top & \mathtt{V} \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

Sparse Levenberg-Marquardt

For missing data $\mathbf{X}_i^{(j)}$, omit $\mathtt{A}_i^{(j)}$, $\mathtt{B}_i^{(j)}$, $\Sigma_{\mathbf{X}_i^{(j)}}$, and $\mathtt{W}_i^{(j)}$

- Normal equations matrix $\begin{bmatrix} \mathtt{U} & \mathtt{W}' \\ \mathtt{W}'^{\top} & \mathtt{V} \end{bmatrix}$



$$\mathtt{U}^{(j)} = \sum_i \mathtt{A}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \mathtt{A}_i^{(j)}$$

$$\mathtt{V}_i = \sum_j \mathtt{B}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \mathtt{B}_i^{(j)}$$

$$\mathtt{W}_i^{(j)} = \mathtt{A}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \mathtt{B}_i^{(j)}$$

Sparse Levenberg-Marquardt

- Normal equations vector $\begin{bmatrix} \boldsymbol{\epsilon}_{\mathbf{a}} \\ \boldsymbol{\epsilon}_{\mathbf{b}} \end{bmatrix}$

$$\left(\boldsymbol{\epsilon}_{\mathbf{a}^{(1)}}^{\top}, \ldots, \boldsymbol{\epsilon}_{\mathbf{a}^{(m)}}^{\top}, \boldsymbol{\epsilon}_{\mathbf{b}_1}^{\top}, \ldots, \boldsymbol{\epsilon}_{\mathbf{b}_n}^{\top}\right)^{\top}$$

where

$$\boldsymbol{\epsilon}_{\mathbf{a}^{(j)}} = \sum_i \mathtt{A}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \boldsymbol{\epsilon}_i^{(j)}$$

$$\boldsymbol{\epsilon}_{\mathbf{b}_i} = \sum_j \mathtt{B}_i^{(j)\top} \Sigma_{\mathbf{X}_i^{(j)}}^{-1} \boldsymbol{\epsilon}_i^{(j)}$$

where $\boldsymbol{\epsilon}_i^{(j)} = \mathbf{X}_i^{(j)} - \hat{\mathbf{X}}_i^{(j)}$

For missing data $\mathbf{X}_i^{(j)}$, omit $\mathtt{A}_i^{(j)}$, $\mathtt{B}_i^{(j)}$, $\Sigma_{\mathbf{X}_i^{(j)}}$, and $\boldsymbol{\epsilon}_i^{(j)}$

Sparse Levenberg-Marquardt

Augmented normal equations

$$\begin{bmatrix} \mathtt{U}^* & \mathtt{W} \\ \mathtt{W}^\top & \mathtt{V}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

where $\mathtt{U}^{(j)*} = \mathtt{U}^{(j)} + \lambda \mathtt{I}$ and $\mathtt{V}_i^* = \mathtt{V}_i + \lambda \mathtt{I}$

Solve for $\left(\boldsymbol{\delta}_\mathbf{a}^\top, \boldsymbol{\delta}_\mathbf{b}^\top\right)^\top$ using blockwise methods

$$\begin{bmatrix} \mathtt{I} & -\mathtt{W}\mathtt{V}^{*-1} \\ \mathtt{0} & \mathtt{I} \end{bmatrix} \begin{bmatrix} \mathtt{U}^* & \mathtt{W} \\ \mathtt{W}^\top & \mathtt{V}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \mathtt{I} & -\mathtt{W}\mathtt{V}^{*-1} \\ \mathtt{0} & \mathtt{I} \end{bmatrix} \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

$$\begin{bmatrix} \mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top & \mathtt{0} \\ \mathtt{W}^\top & \mathtt{V}^* \end{bmatrix} \begin{bmatrix} \boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} - \mathtt{W}\mathtt{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

$$\begin{bmatrix} (\mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top)\boldsymbol{\delta}_\mathbf{a} \\ \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a} + \mathtt{V}^*\boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} - \mathtt{W}\mathtt{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

Sparse Levenberg-Marquardt

$$\begin{bmatrix} (\mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top)\boldsymbol{\delta}_\mathbf{a} \\ \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a} + \mathtt{V}^*\boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} - \mathtt{W}\mathtt{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

1. Solve for $\boldsymbol{\delta}_\mathbf{a}$

$$(\mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top)\boldsymbol{\delta}_\mathbf{a} = \boldsymbol{\epsilon}_\mathbf{a} - \mathtt{W}\mathtt{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b}$$

$$\mathtt{S}\boldsymbol{\delta}_\mathbf{a} = \mathbf{e}, \text{ solve for } \boldsymbol{\delta}_\mathbf{a} = (\boldsymbol{\delta}_{\mathbf{a}(1)}^\top, \ldots, \boldsymbol{\delta}_{\mathbf{a}(m)}^\top)^\top$$

where

$$\mathtt{S} = \begin{bmatrix} \mathtt{S}^{(1,1)} & \cdots & \mathtt{S}^{(1,m)} \\ \vdots & \ddots & \vdots \\ \mathtt{S}^{(m,1)} & \cdots & \mathtt{S}^{(m,m)} \end{bmatrix}$$

where

$$\mathtt{S}^{(j,k)} = \begin{cases} \mathtt{U}^{(j)*} - \sum_i \mathtt{W}_i^{(j)}\mathtt{V}_i^{*-1}\mathtt{W}_i^{(k)\top} & \text{if } j = k \\ -\sum_i \mathtt{W}_i^{(j)}\mathtt{V}_i^{*-1}\mathtt{W}_i^{(k)\top} & \text{if } j \neq k \end{cases}$$

and

$$\mathbf{e} = (\mathbf{e}^{(1)\top}, \ldots, \mathbf{e}^{(m)\top})^\top$$

where

$$\mathbf{e}^{(j)} = \boldsymbol{\epsilon}_{\mathbf{a}(j)} - \sum_i \mathtt{W}_i^{(j)}\mathtt{V}_i^{*-1}\boldsymbol{\epsilon}_{\mathbf{b}_i}$$

For missing data $\mathbf{X}_i^{(j)}$, omit $\mathtt{W}_i^{(j)}$

Sparse Levenberg-Marquardt

$$\begin{bmatrix} (\mathtt{U}^* - \mathtt{W}\mathtt{V}^{*-1}\mathtt{W}^\top)\boldsymbol{\delta}_\mathbf{a} \\ \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a} + \mathtt{V}^*\boldsymbol{\delta}_\mathbf{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\epsilon}_\mathbf{a} - \mathtt{W}\mathtt{V}^{*-1}\boldsymbol{\epsilon}_\mathbf{b} \\ \boldsymbol{\epsilon}_\mathbf{b} \end{bmatrix}$$

2. Solve for $\boldsymbol{\delta}_\mathbf{b}$

$$\begin{aligned} \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a} + \mathtt{V}^*\boldsymbol{\delta}_\mathbf{b} &= \boldsymbol{\epsilon}_\mathbf{b} \\ \mathtt{V}^*\boldsymbol{\delta}_\mathbf{b} &= \boldsymbol{\epsilon}_\mathbf{b} - \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a} \\ \boldsymbol{\delta}_\mathbf{b} &= \mathtt{V}^{*-1}\left(\boldsymbol{\epsilon}_\mathbf{b} - \mathtt{W}^\top\boldsymbol{\delta}_\mathbf{a}\right) \end{aligned}$$

where

$$\boldsymbol{\delta}_{\mathbf{b}_i} = \mathtt{V}_i^{*-1}\left(\boldsymbol{\epsilon}_{\mathbf{b}_i} - \sum_j \mathtt{W}_i^{(j)\top}\boldsymbol{\delta}_{\mathbf{a}^{(j)}}\right)$$

For missing data $\mathbf{X}_i^{(j)}$, omit $\mathtt{W}_i^{(j)}$

Candidate parameter vector

Adjustment

$$\begin{aligned} \mathbf{a}_0^{(j)} &= \mathbf{a}^{(j)} + \boldsymbol{\delta}_{\mathbf{a}^{(j)}} \\ \mathbf{b}_{i_0} &= \mathbf{b}_i + \boldsymbol{\delta}_{\mathbf{b}_i} \end{aligned}$$

Bundle adjustment, uncalibrated, using the Levenberg-Marquardt algorithm

Parameter vector

$$\left(\hat{\mathbf{p}}^{(1)\top}, \ldots, \hat{\mathbf{p}}^{(m)\top}, \hat{\mathbf{X}}_1^\top, \ldots, \hat{\mathbf{X}}_n^\top\right)^\top$$

Mapping

$$\hat{\mathbf{x}}_i^{(j)} = \hat{\mathrm{P}}^{(j)} \hat{\mathbf{X}}_i \, \forall \, i, j$$

Measurement vector

$$\left(\tilde{\mathbf{x}}_1^{(1)\top}, \ldots, \tilde{\mathbf{x}}_n^{(1)\top}, \ldots, \tilde{\mathbf{x}}_1^{(m)\top}, \ldots, \tilde{\mathbf{x}}_n^{(m)\top}\right)^\top$$

with associated covariance matrices $\Sigma_{\tilde{\mathbf{x}}_1^{(1)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_n^{(1)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_1^{(m)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_n^{(m)}}$

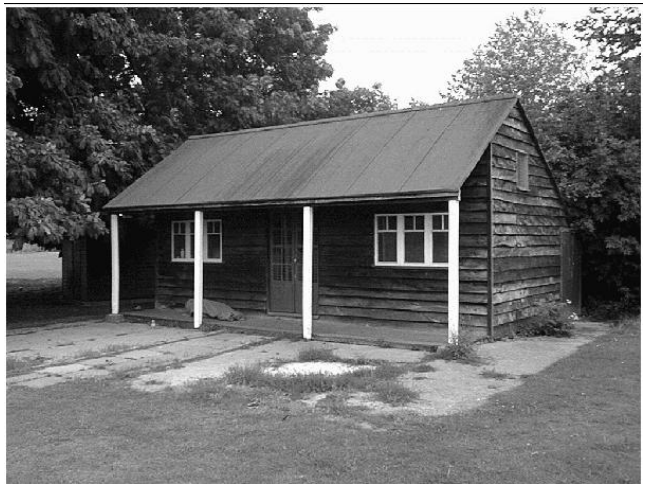
Cost

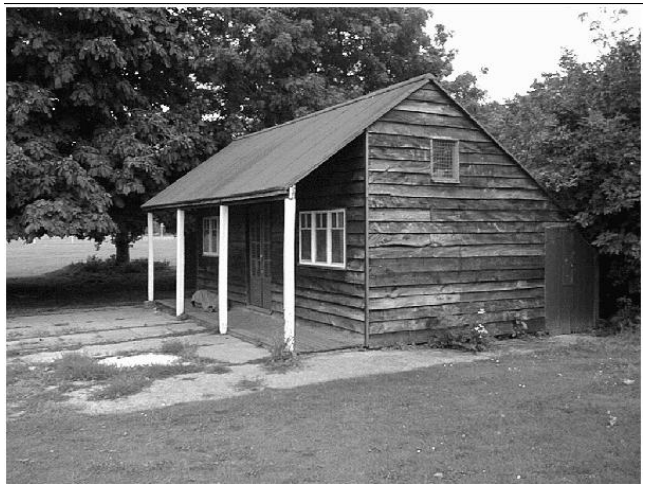
$$\sum_i \sum_j \boldsymbol{\epsilon}_i^{(j)\top} \Sigma_{\tilde{\mathbf{x}}_i^{(j)}}^{-1} \boldsymbol{\epsilon}_i^{(j)}, \text{ where } \boldsymbol{\epsilon}_i^{(j)} = \tilde{\mathbf{x}}_i^{(j)} - \hat{\tilde{\mathbf{x}}}_i^{(j)}$$

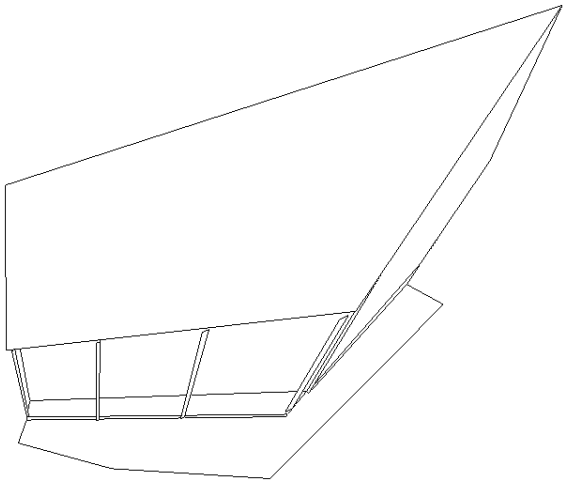
For missing data $\tilde{\mathbf{x}}_i^{(j)}$, omit $\Sigma_{\tilde{\mathbf{x}}_i^{(j)}}$ and $\boldsymbol{\epsilon}_i^{(j)}$

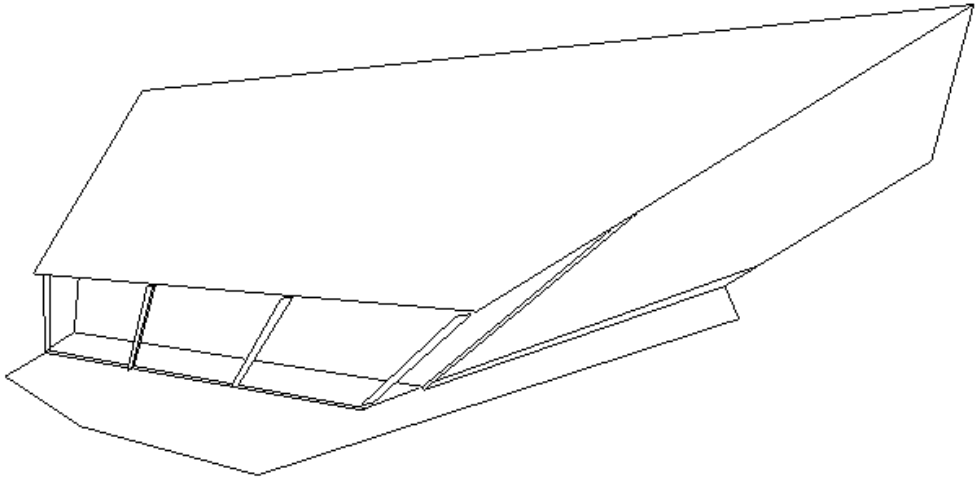
But, use data normalization

Uncalibrated: up to 3D projective transformation



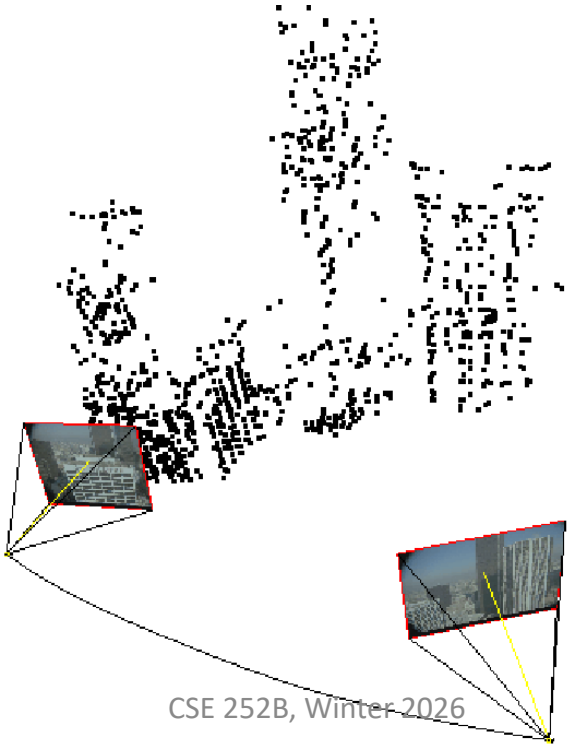






n view reconstruction of scene structure and cameras





After autocalibration

Bundle adjustment, calibrated, using the Levenberg-Marquardt algorithm

Parameter vector

$$\left(\hat{\boldsymbol{\omega}}^{(1)\top}, \hat{\mathbf{t}}^{(1)\top}, \ldots, \hat{\boldsymbol{\omega}}^{(m)\top}, \hat{\mathbf{t}}^{(m)\top}, \hat{\mathbf{X}}_1^\top, \ldots, \hat{\mathbf{X}}_n^\top\right)^\top$$

Mapping

$$\hat{\hat{\mathbf{x}}}_i^{(j)} = [\exp([\hat{\boldsymbol{\omega}}^{(j)}]_\times) \,|\, \hat{\mathbf{t}}^{(j)}]\hat{\mathbf{X}}_i \,\forall\, i, j$$

Measurement vector

$$\left(\hat{\hat{\mathbf{x}}}_1^{(1)\top}, \ldots, \hat{\hat{\mathbf{x}}}_n^{(1)\top}, \ldots, \hat{\hat{\mathbf{x}}}_1^{(m)\top}, \ldots, \hat{\hat{\mathbf{x}}}_n^{(m)\top}\right)^\top$$

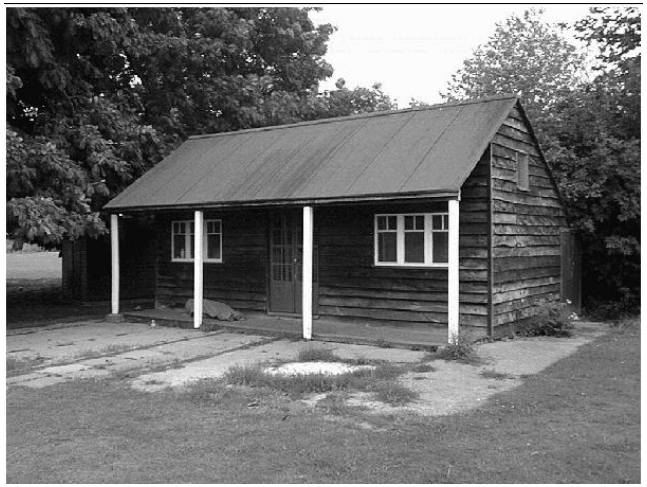
with associated covariance matrices $\Sigma_{\hat{\hat{\mathbf{x}}}_1^{(1)}}, \ldots, \Sigma_{\hat{\hat{\mathbf{x}}}_n^{(1)}}, \ldots, \Sigma_{\hat{\hat{\mathbf{x}}}_1^{(m)}}, \ldots, \Sigma_{\hat{\hat{\mathbf{x}}}_n^{(m)}}$

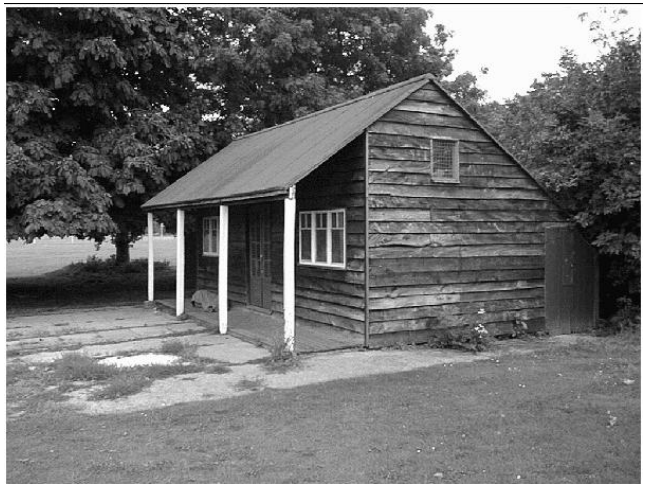
Cost

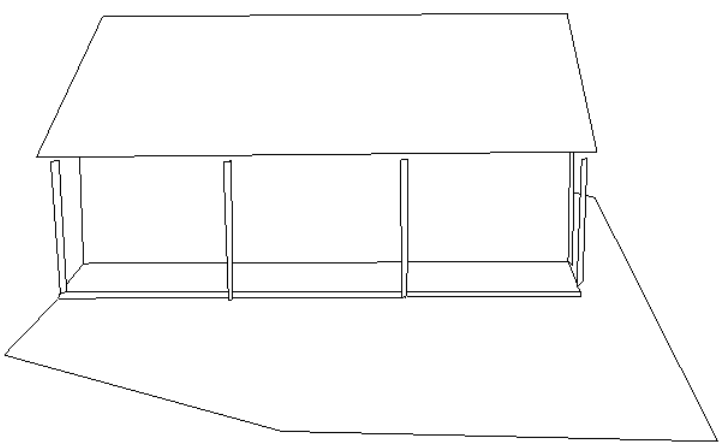
$$\sum_i \sum_j \boldsymbol{\epsilon}_i^{(j)\top} \Sigma^{-1}_{\hat{\hat{\mathbf{x}}}_i^{(j)}} \boldsymbol{\epsilon}_i^{(j)}, \text{ where } \boldsymbol{\epsilon}_i^{(j)} = \hat{\hat{\mathbf{x}}}_i^{(j)} - \hat{\hat{\hat{\mathbf{x}}}}_i^{(j)}$$

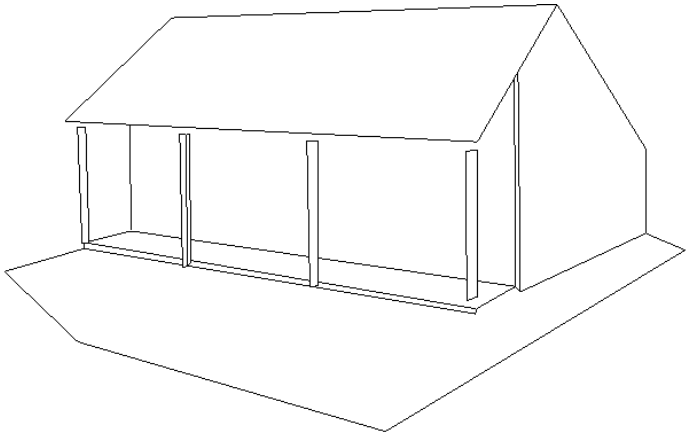
For missing data $\hat{\hat{\mathbf{x}}}_i^{(j)}$, omit $\Sigma_{\hat{\hat{\mathbf{x}}}_i^{(j)}}$ and $\boldsymbol{\epsilon}_i^{(j)}$

Calibrated: up to 3D similarity transformation

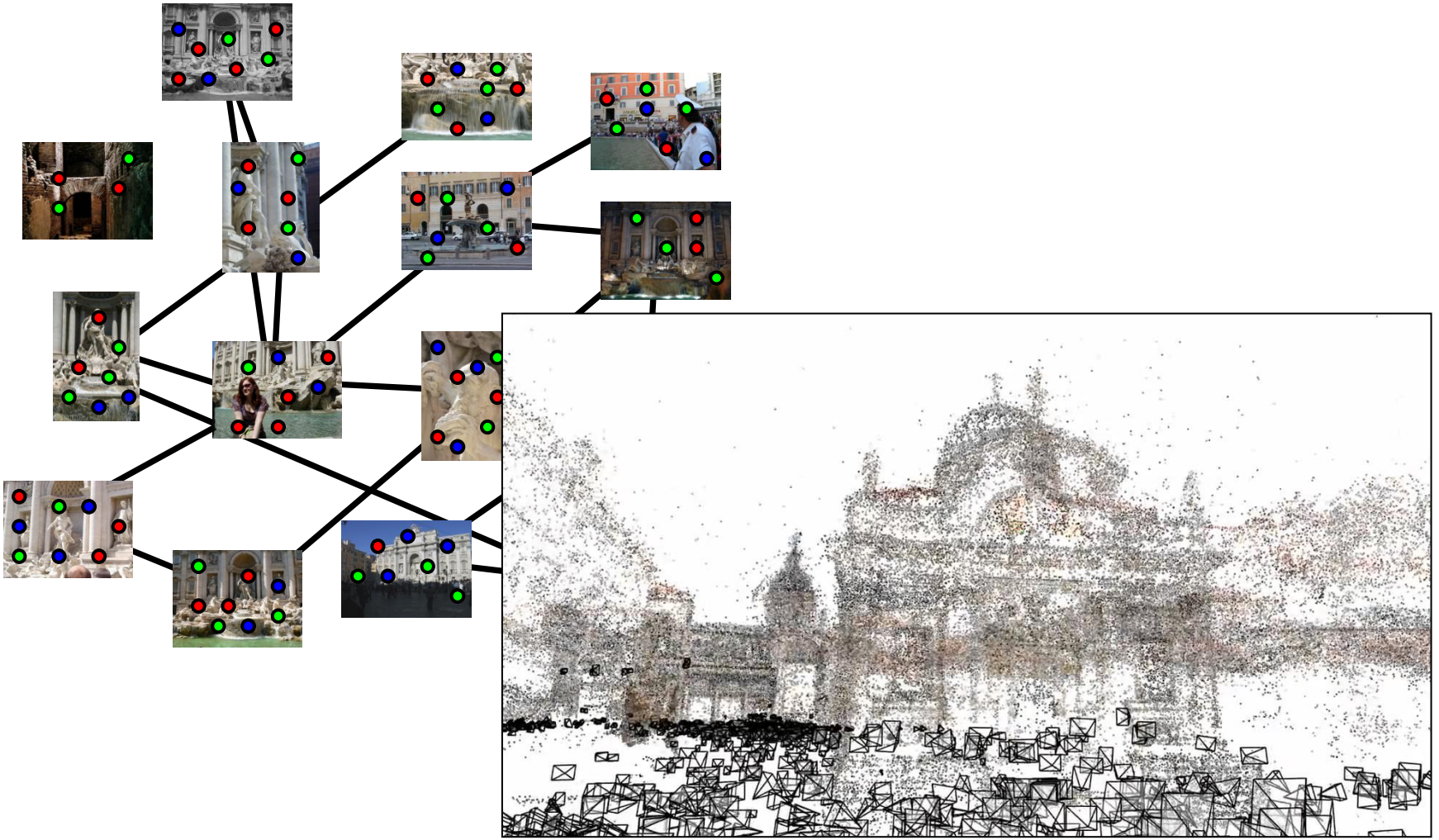








n view reconstruction of scene structure and cameras



2D block adjustment, uncalibrated, using the Levenberg-Marquardt algorithm

Parameter vector

$$\left(\hat{\mathbf{h}}^{(1)\top}, \ldots, \hat{\mathbf{h}}^{(m)\top}, \hat{\mathbf{x}}_{\text{scene},1}^{\top}, \ldots, \hat{\mathbf{x}}_{\text{scene},n}\right)^{\top}$$

Mapping

$$\hat{\mathbf{x}}_i^{(j)} = \hat{\mathtt{H}}^{(j)} \hat{\mathbf{x}}_{\text{scene},i} \, \forall \, i, j$$

Measurement vector

$$\left(\tilde{\mathbf{x}}_1^{(1)\top}, \ldots, \tilde{\mathbf{x}}_n^{(1)\top}, \ldots, \tilde{\mathbf{x}}_1^{(m)\top}, \ldots, \tilde{\mathbf{x}}_n^{(m)\top}\right)^{\top}$$

with associated covariance matrices $\Sigma_{\tilde{\mathbf{x}}_1^{(1)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_n^{(1)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_1^{(m)}}, \ldots, \Sigma_{\tilde{\mathbf{x}}_n^{(m)}}$

Cost

$$\sum_i \sum_j \boldsymbol{\epsilon}_i^{(j)\top} \Sigma_{\tilde{\mathbf{x}}_i^{(j)}}^{-1} \boldsymbol{\epsilon}_i^{(j)}, \text{ where } \boldsymbol{\epsilon}_i^{(j)} = \tilde{\mathbf{x}}_i^{(j)} - \hat{\tilde{\mathbf{x}}}_i^{(j)}$$

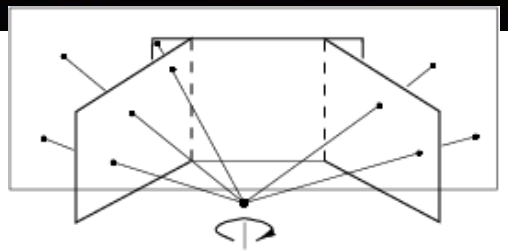
For missing data $\tilde{\mathbf{x}}_i^{(j)}$, omit $\Sigma_{\tilde{\mathbf{x}}_i^{(j)}}$ and $\boldsymbol{\epsilon}_i^{(j)}$

But, use data normalization

n view reconstruction of scene structure and cameras

- Mosaic construction from rotating camera





2D projective transformations
2D scene points



Next lecture

- Additional topics
- Reading
  - Sections 2.7, 7.4, 8.5, 8.6, 8.7, 8.8, and A6.8, and chapter 19