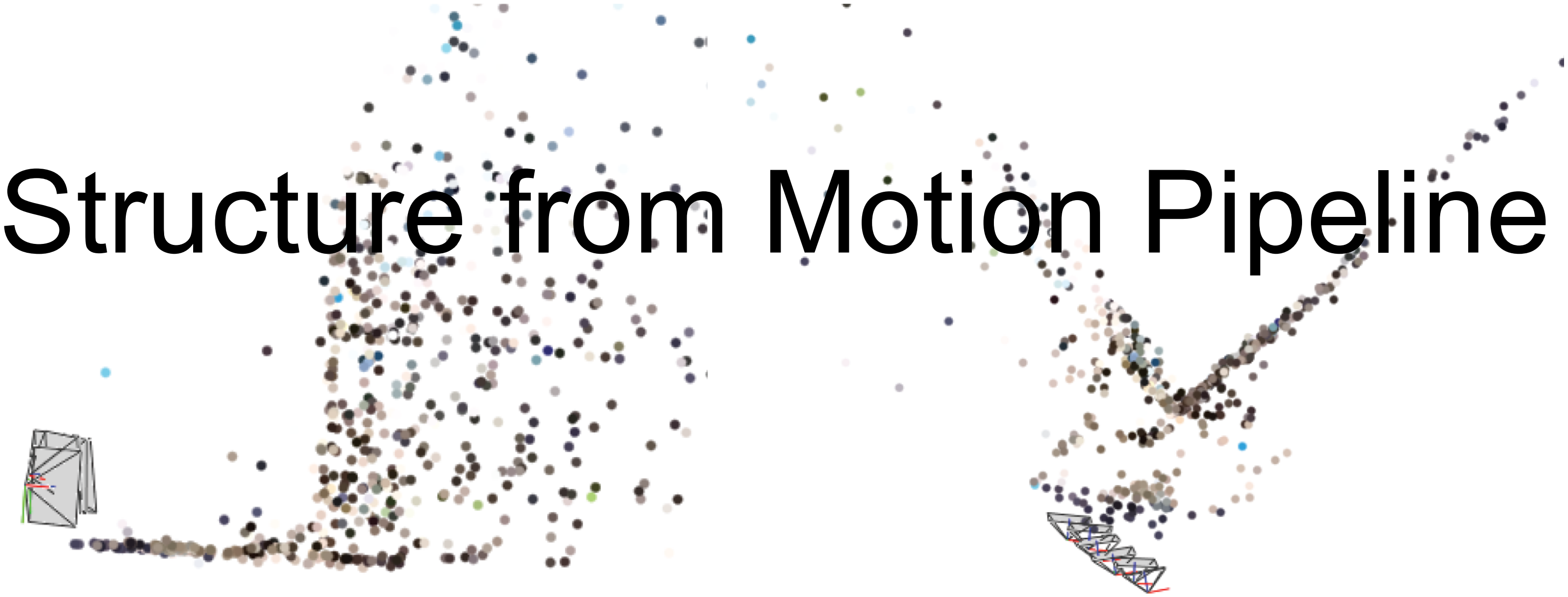
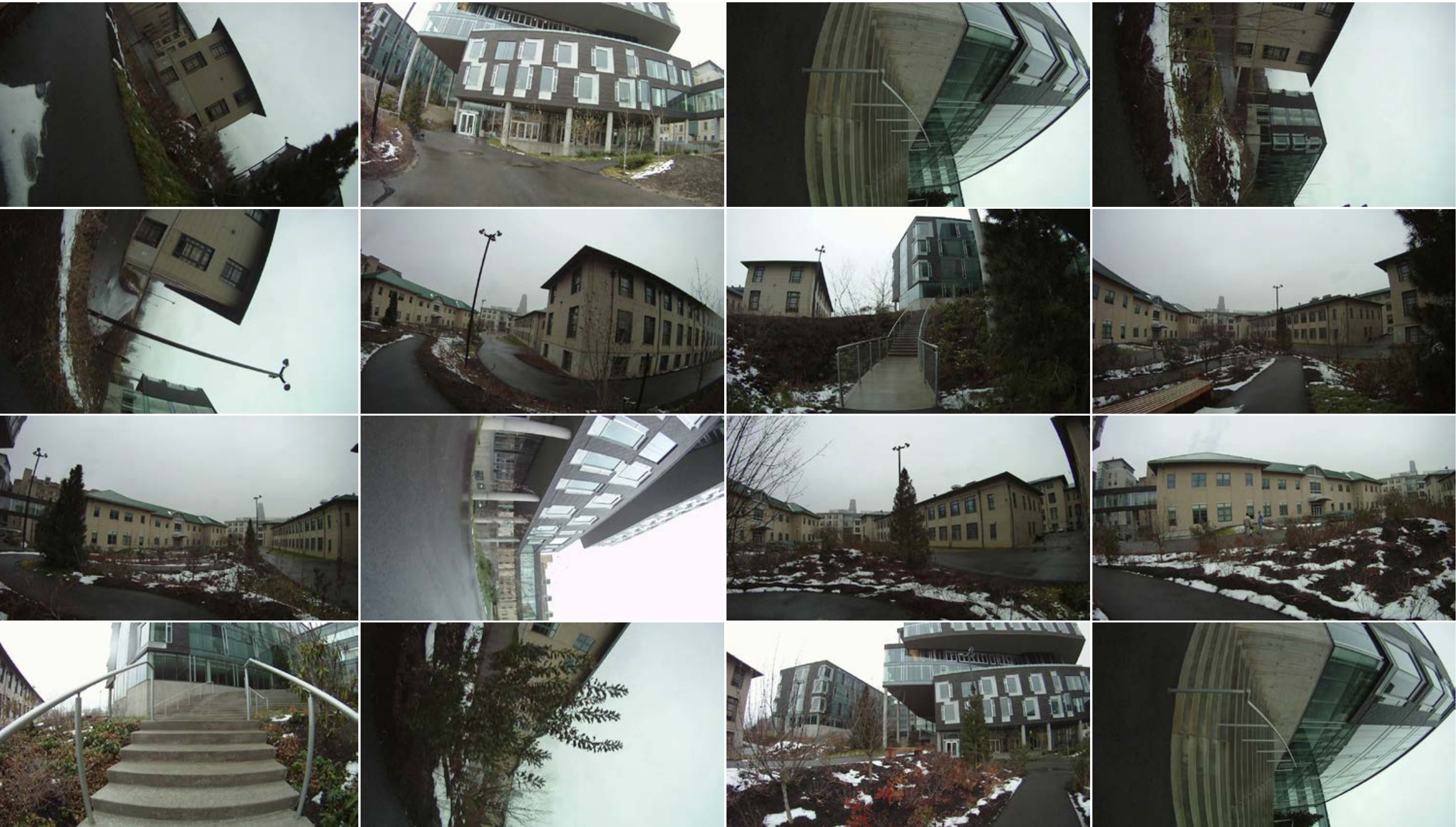
Structure from Motion Pipeline

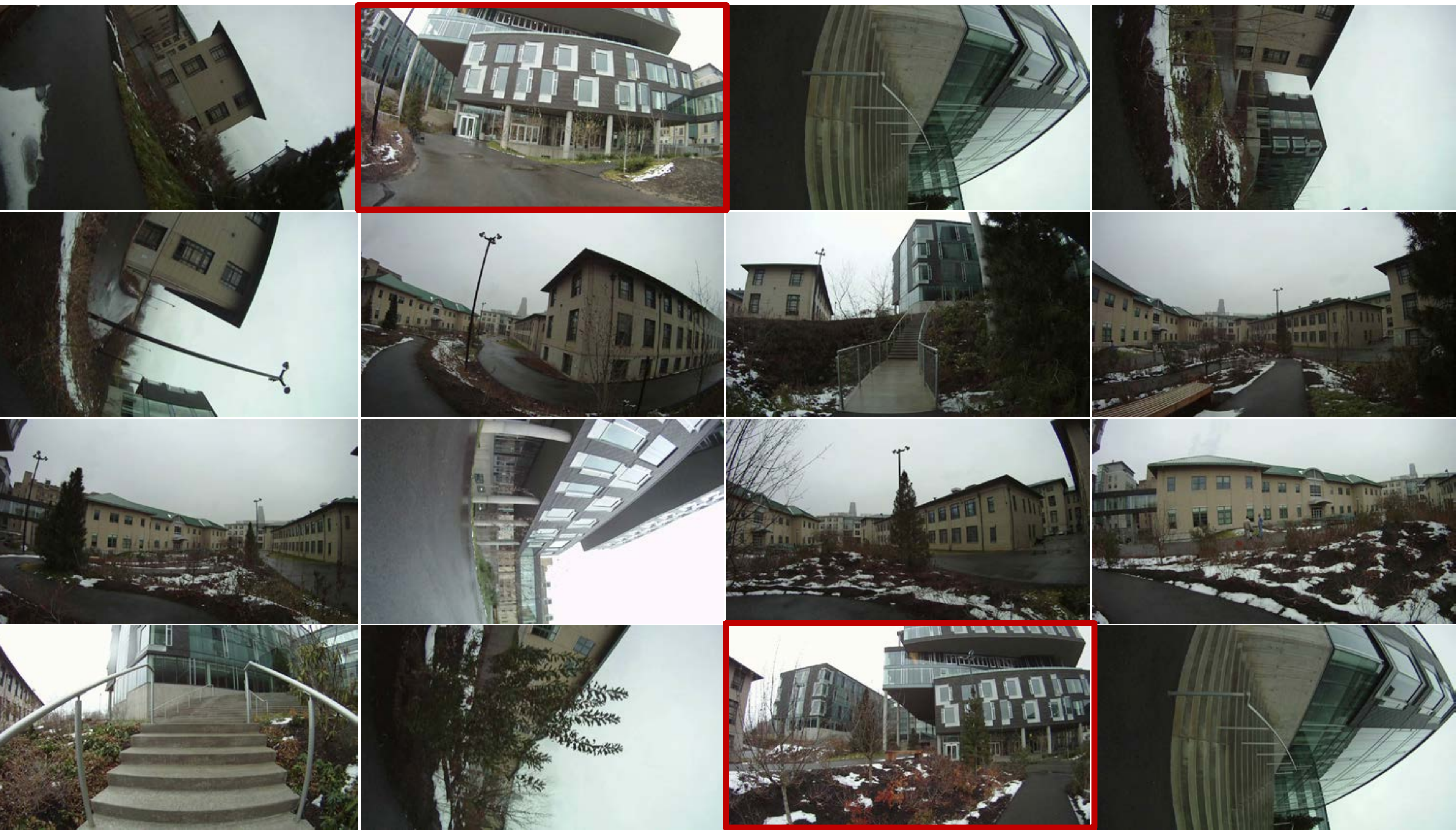


Where Am I (camera)?
Where are they (points)?

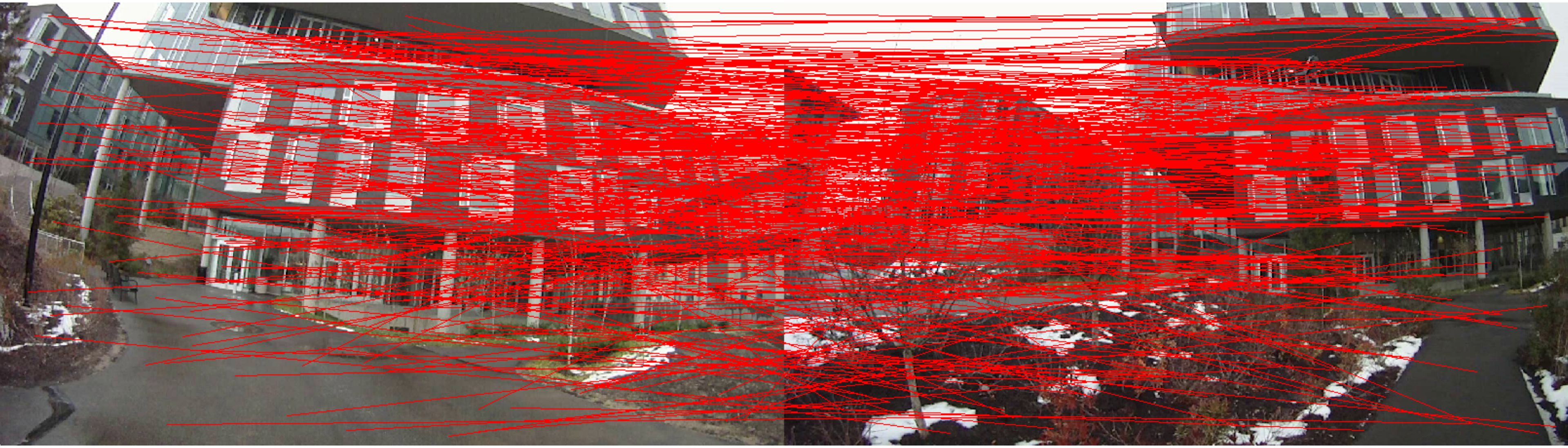
Input Images

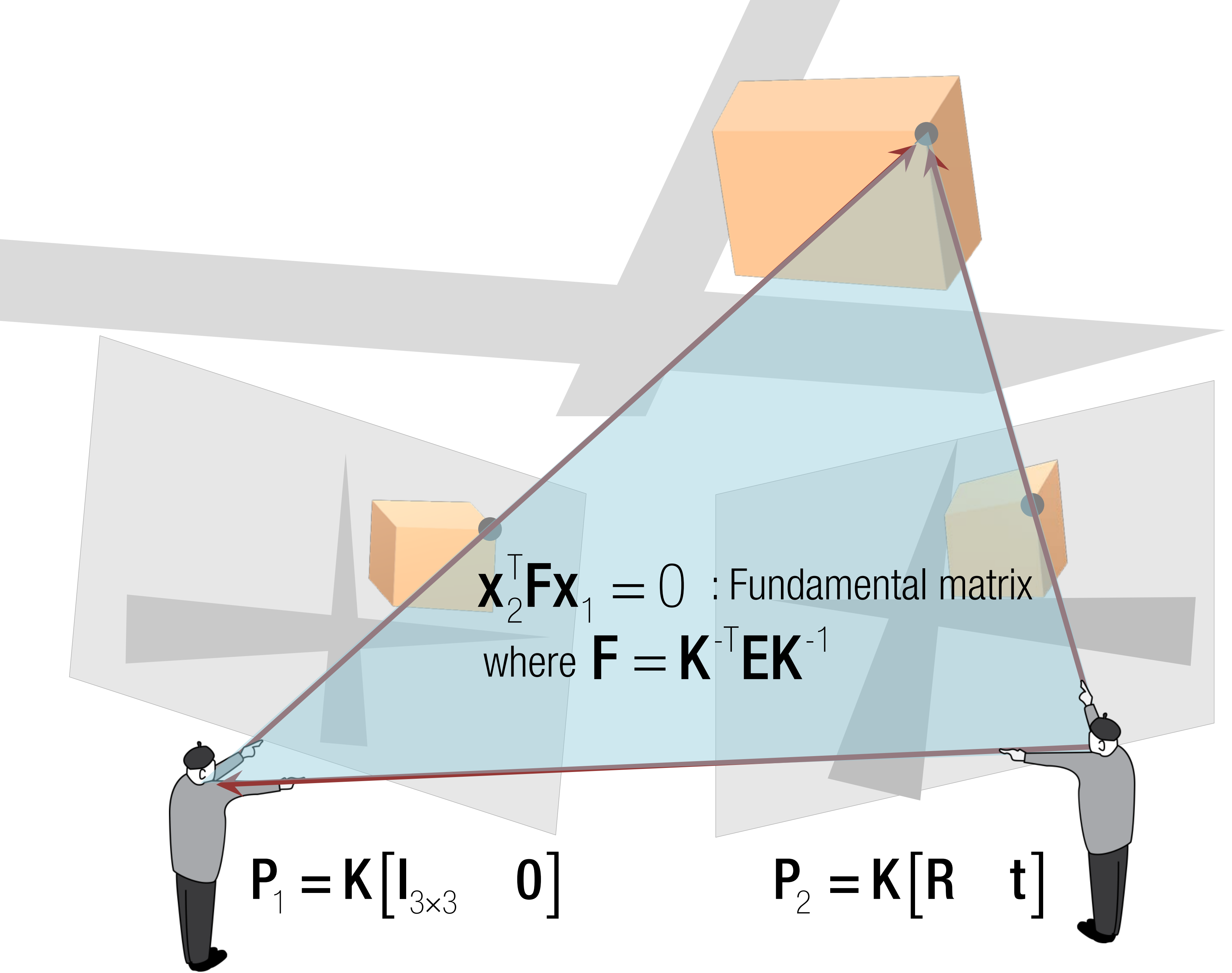


Initial Pair Images



1. Pairwise Image Feature Matching





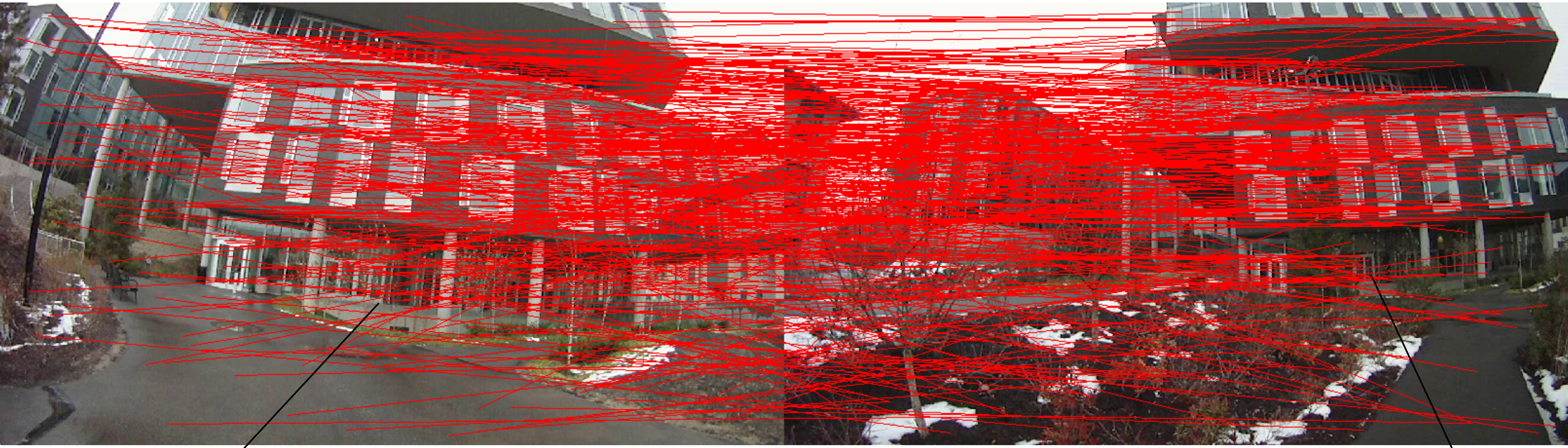
$$x_2^{T}\mathbf{F}x_1 = 0$$: Fundamental matrix

where $\mathbf{F} = \mathbf{K}^{-T}\mathbf{E}\mathbf{K}^{-1}$

$$\mathbf{P}_1 = \mathbf{K}\begin{bmatrix}\mathbf{I}_{3\times3} & \mathbf{0}\end{bmatrix}$$

$$\mathbf{P}_2 = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$$

1. Pairwise Image Feature Matching



$\mathbf{x}_1$

$\mathbf{x}_2$

$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0$$

Epipolar constraint

# of unknowns: 8

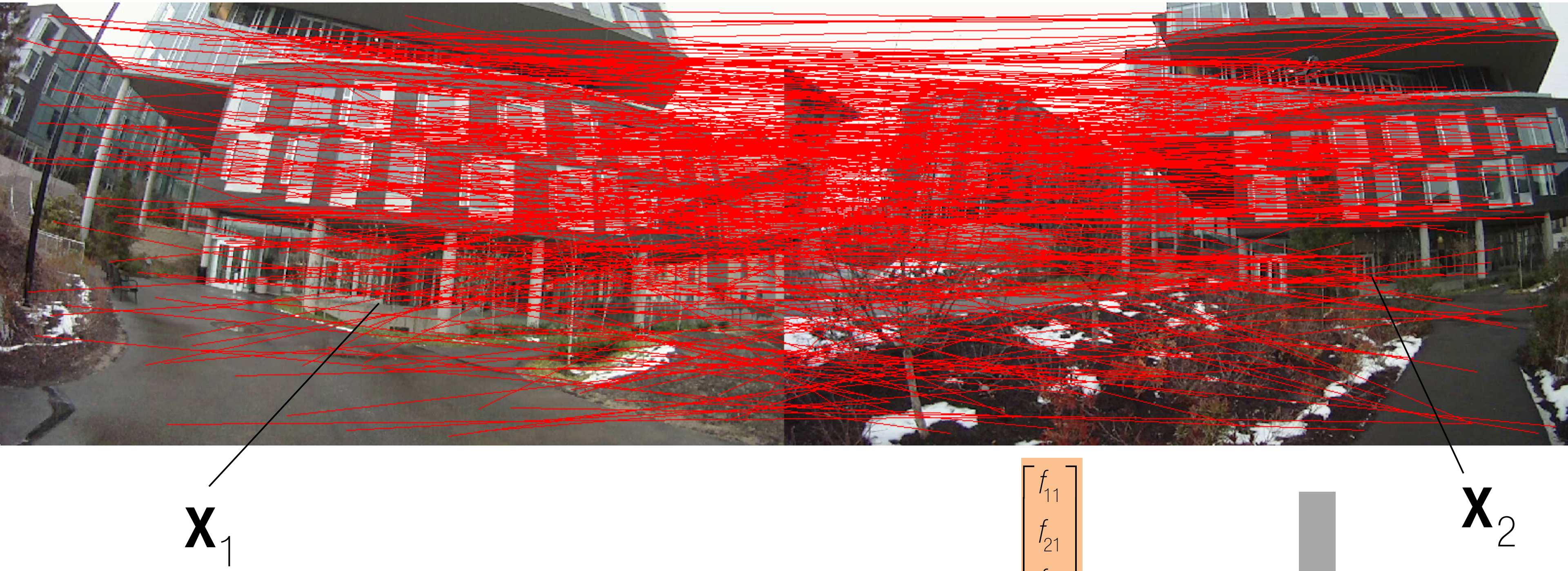
# of required equations: 8

$$\mathbf{x}_{2,1}^{\mathrm{T}}\mathbf{F}\mathbf{x}_{1,1} = 0$$

$$\vdots$$

$$\mathbf{x}_{2,8}^{\mathrm{T}}\mathbf{F}\mathbf{x}_{1,8} = 0$$

1. Pairwise Image Feature Matching



$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0$$

Epipolar constraint

$$\underbrace{\begin{bmatrix} u_1^1u_1^2 & u_1^1v_1^2 & u_1^1 & v_1^1u_1^2 & v_1^1v_1^2 & v_1^1 & u_1^2 & v_1^2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_8^1u_8^2 & u_8^1v_8^2 & u_8^1 & v_8^1u_8^2 & v_8^1v_8^2 & v_8^1 & u_8^2 & v_8^2 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}}_{\mathbf{x}} = \mathbf{0}$$

$=0$

8 Point Algorithm

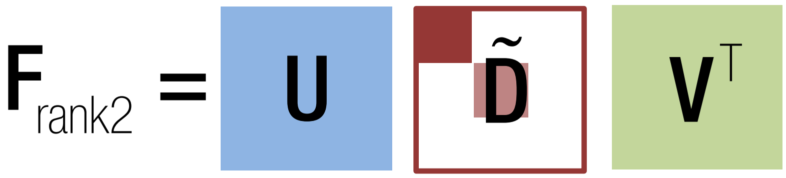
- Construct 8x9 matrix **A**.
- Solving linear homogeneous equations via SVD:

$$\mathbf{x} = \mathbf{V}_{:,8} \quad \text{where} \quad \mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$

$$\mathbf{F} = \mathrm{reshape}(\mathbf{x}, 3, 3)\text{: constructing matrix from vector.}$$

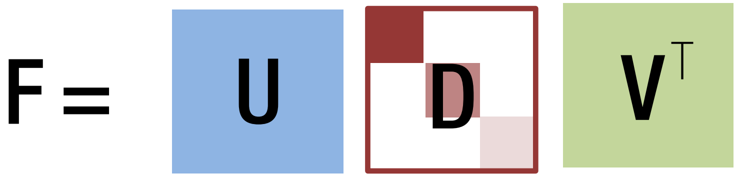
- Applying rank constraint, i.e., $\mathrm{rank}(\mathbf{F}) = 2$.

$$\mathbf{F}_{\mathrm{rank2}} = \mathbf{U}\tilde{\mathbf{D}}\mathbf{V}^{\mathsf{T}} \quad \text{where} \quad \tilde{\mathbf{D}} \; :\mathbf{D} \quad \text{with the last element zero.}$$



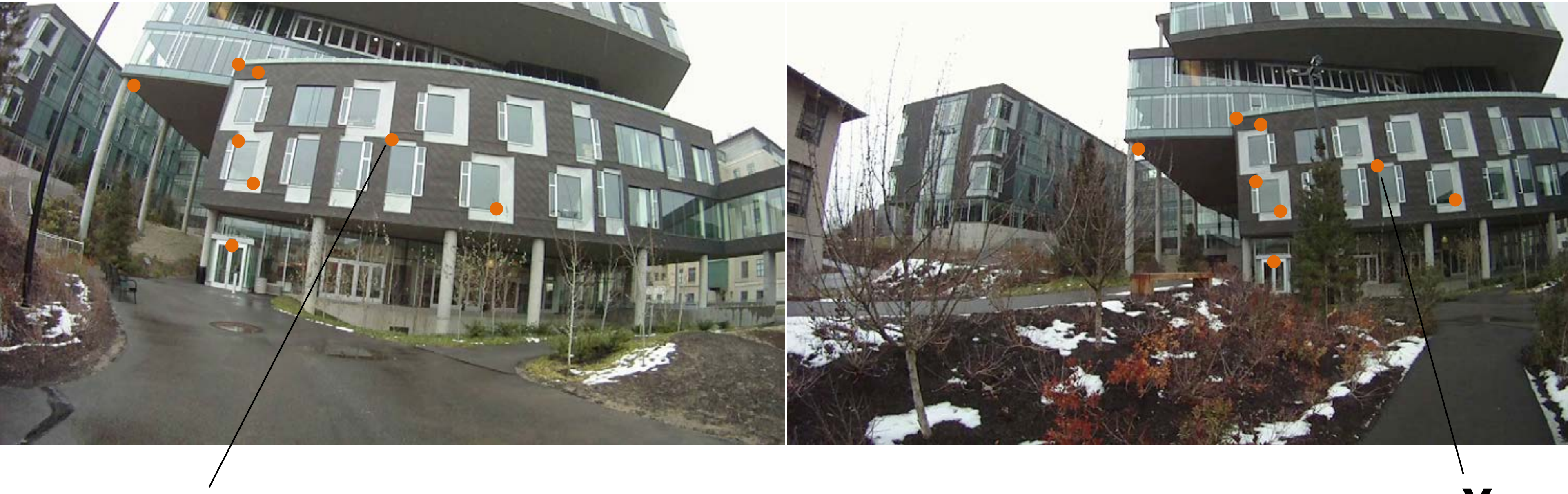
$$\mathbf{F}_{\mathrm{rank2}} = \mathbf{U}\;\tilde{\mathbf{D}}\;\mathbf{V}^{\mathsf{T}}$$

SVD cleanup



$$\mathbf{F} = \mathbf{U}\;\mathbf{D}\;\mathbf{V}^{\mathsf{T}}$$

2. Outlier Rejection via RANSAC



$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0$$

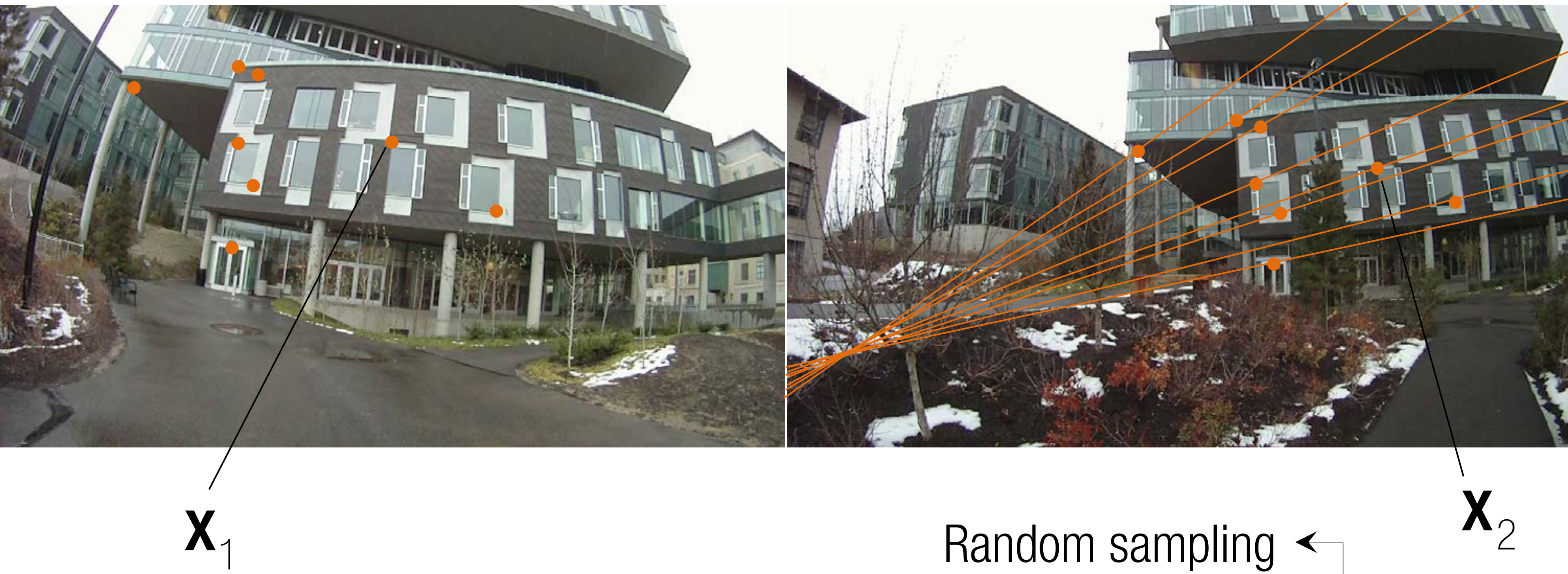
Random sampling

Model building

Thresholding

Inlier counting

2. Outlier Rejection via RANSAC



$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0$$

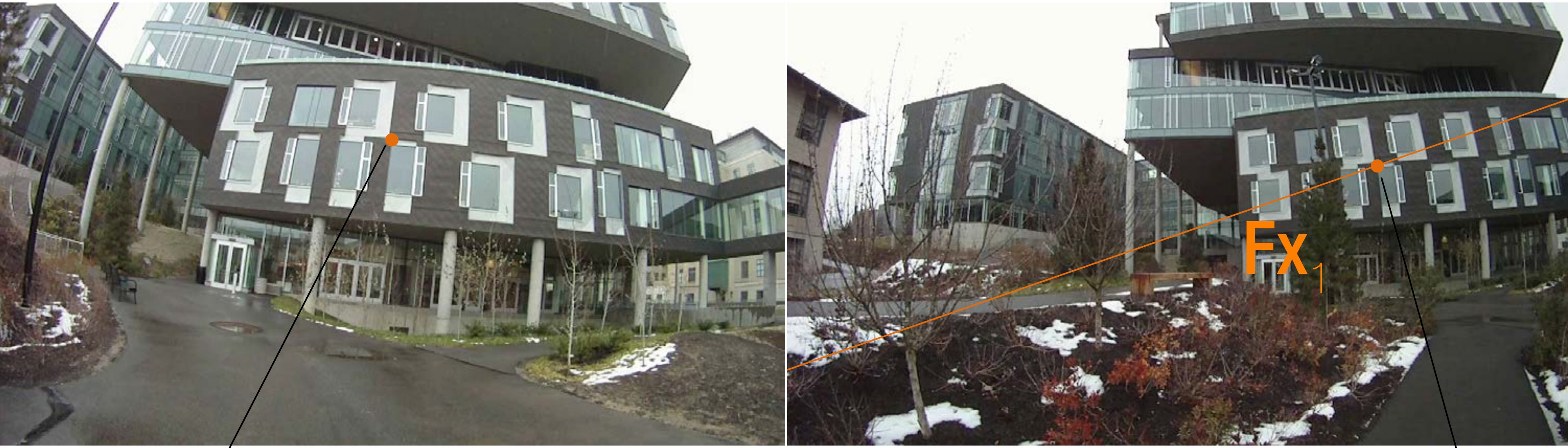
Random sampling

Model building

Thresholding

Inlier counting

2. Outlier Rejection via RANSAC



$$\mathbf{x}_2^{\mathrm{T}}\mathbf{F}\mathbf{x}_1 = 0$$

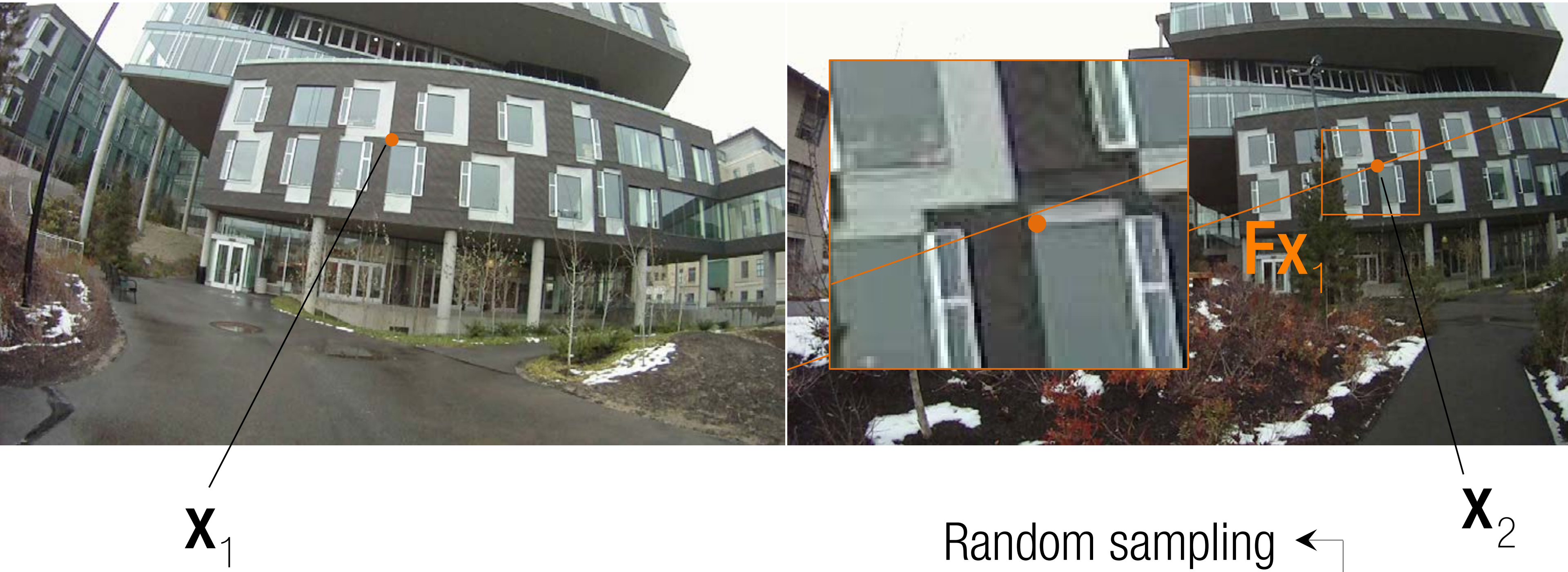
Random sampling

Model building

Thresholding

Inlier counting

2. Outlier Rejection via RANSAC



$$\mathbf{x}_2^{\mathsf{T}}\mathbf{F}\mathbf{x}_1 = 0$$

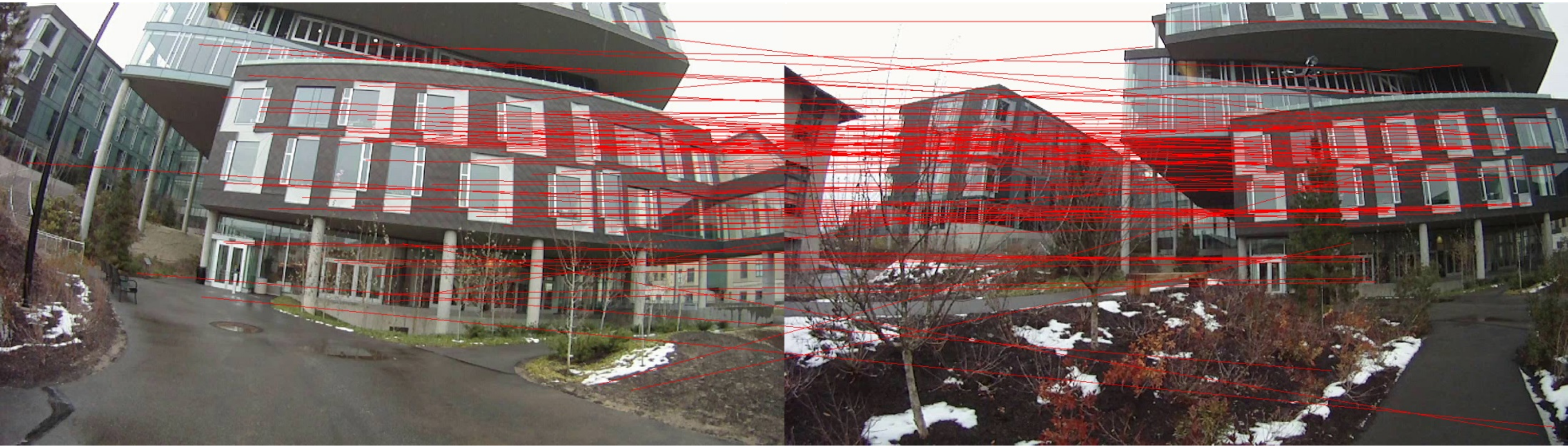
Random sampling

Model building

Thresholding

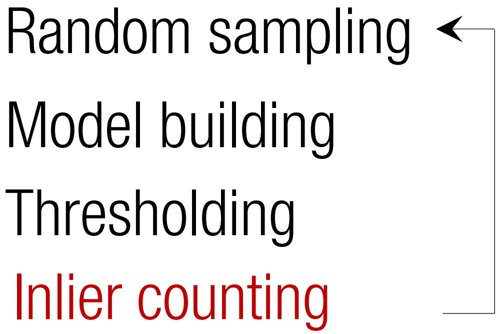
Inlier counting

2. Outlier Rejection via RANSAC



$$\mathbf{x}_2^T \mathbf{F}\mathbf{x}_1 = 0$$

# of inliers: 253

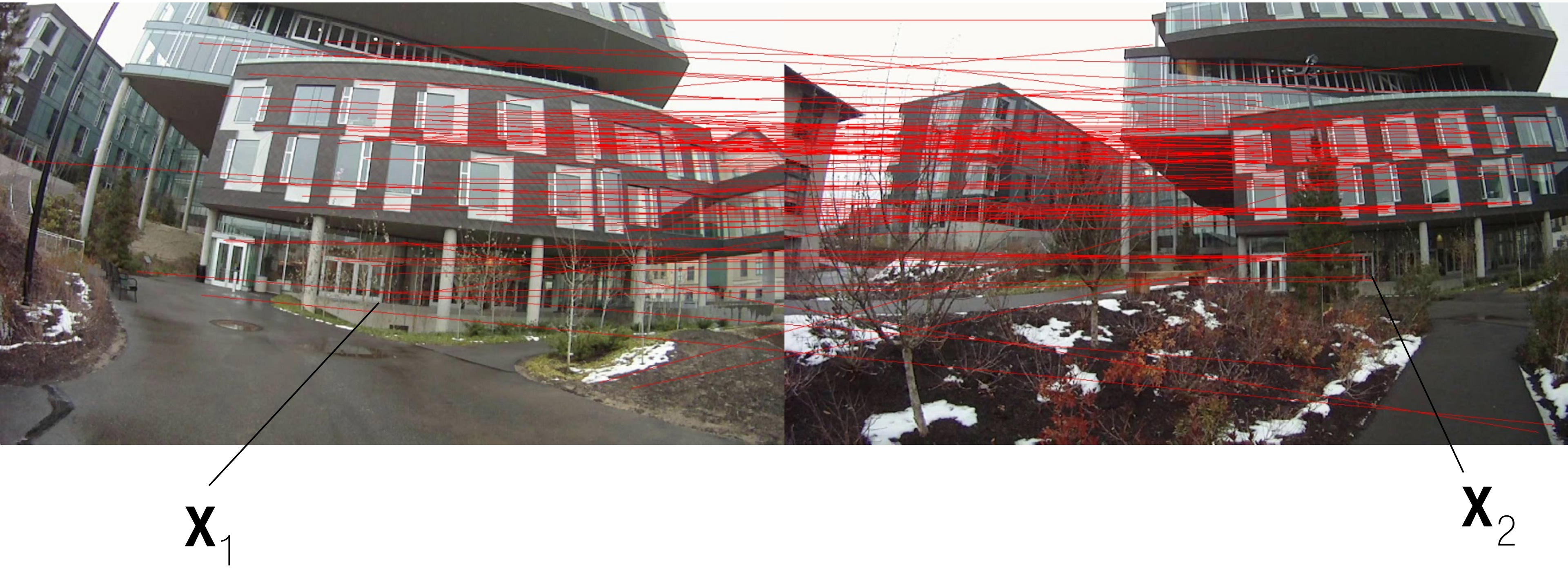

Random sampling

Model building

Thresholding

Inlier counting

3. Essential Matrix Computation

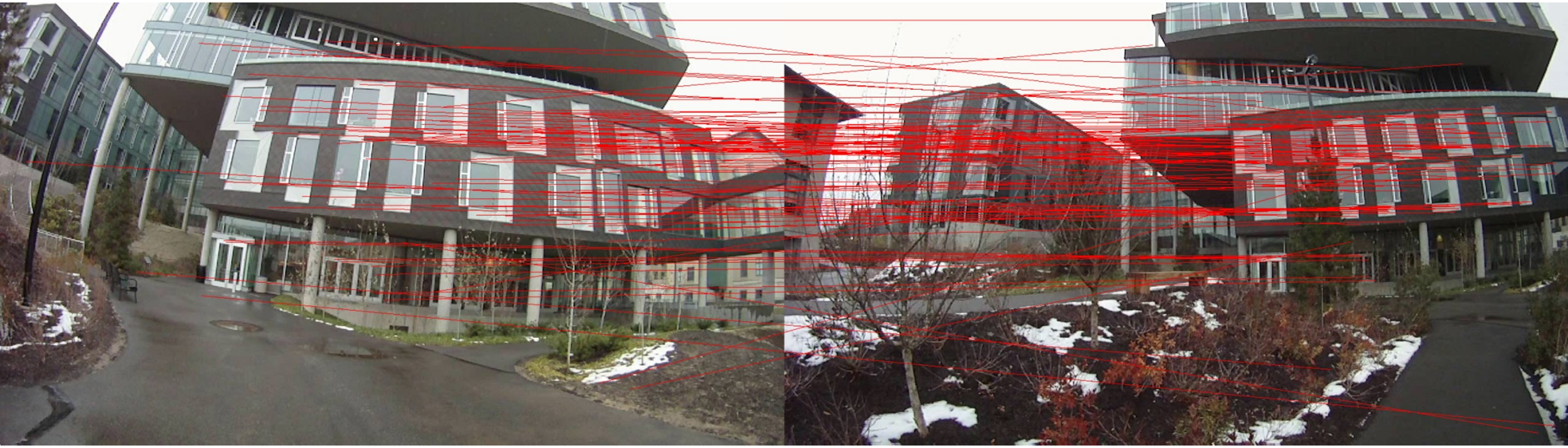


$$\mathbf{x}_2^{\top}\mathbf{K}^{-\top}\mathbf{E}\mathbf{K}^{-1}\mathbf{x}_1 = 0$$

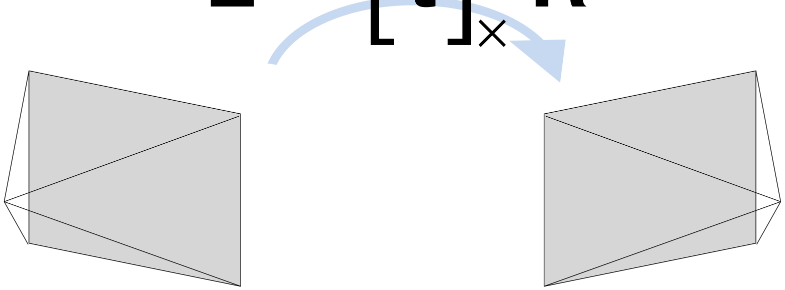
$$\mathbf{x}_2^{\top}\mathbf{F}\mathbf{x}_1 = 0$$

$$\longrightarrow \quad \mathbf{K}^{\top}\mathbf{F}\mathbf{K} = \mathbf{E}$$

4. Relative Transform from Essential Matrix



$$E = \left[t \right]_{\times} R$$



$$P_1 = \left[I_{3\times3} \mid 0_3 \right] \qquad P_2 = \left[R \mid t \right]$$

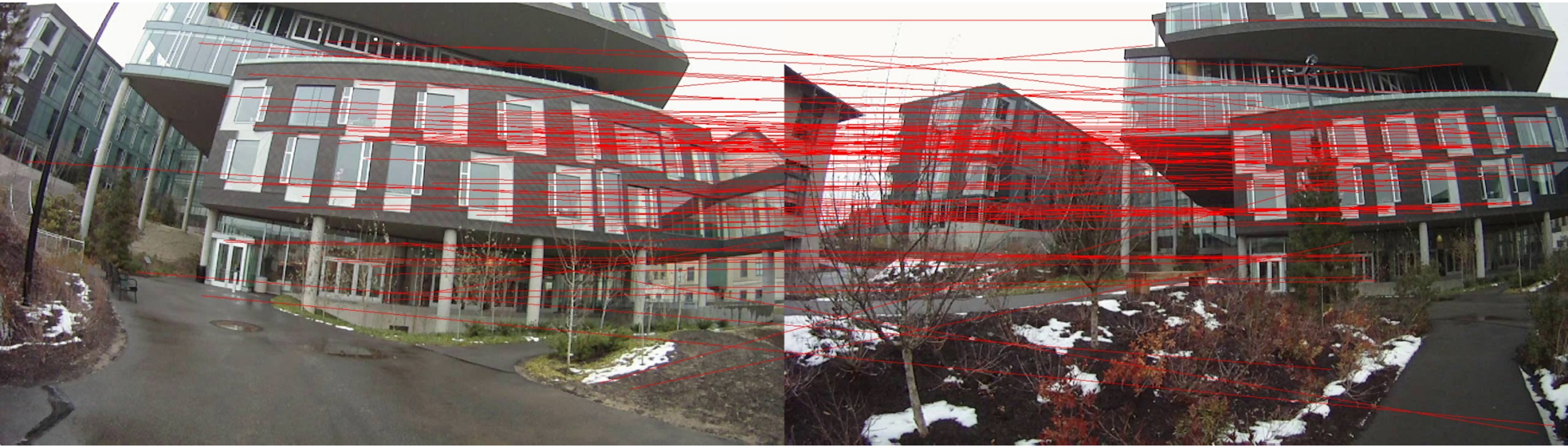
$$P_2 = \left[UYV^{\top} \mid u_3 \right]$$

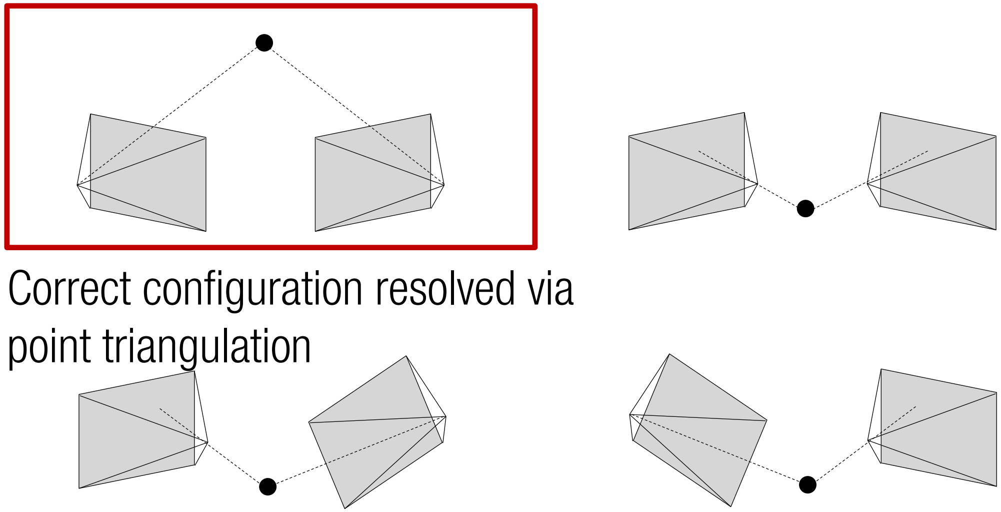
$$\left[UY^{\top}V^{\top} \mid u_3 \right]$$

$$\left[UYV^{\top} \mid -u_3 \right]$$

$$\left[UY^{\top}V^{\top} \mid -u_3 \right]$$

4. Relative Transform from Essential Matrix





Correct configuration resolved via point triangulation

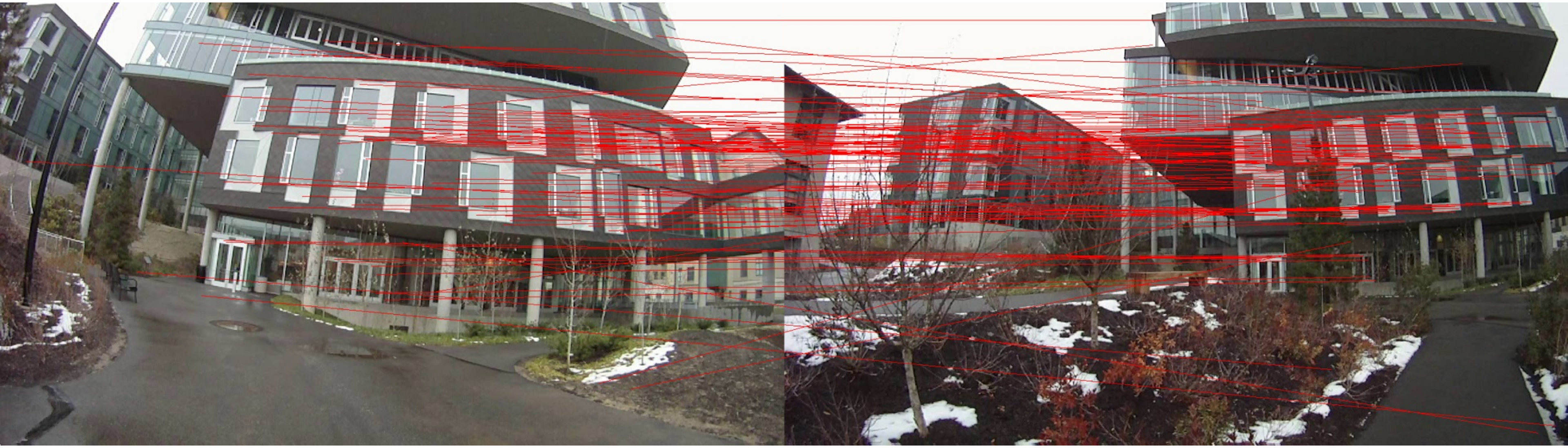
$$P_2 = \left[UYV^{\top} \mid u_3 \right]$$

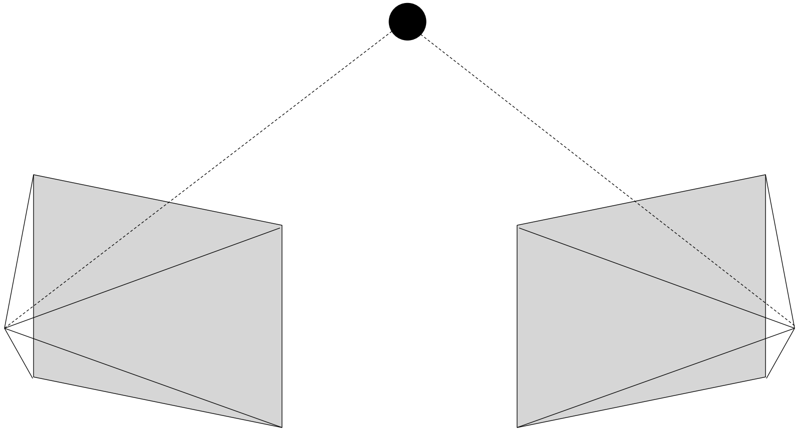
$$\left[UY^{\top}V^{\top} \mid u_3 \right]$$

$$\left[UYV^{\top} \mid -u_3 \right]$$

$$\left[UY^{\top}V^{\top} \mid -u_3 \right]$$

5. Point Triangulation

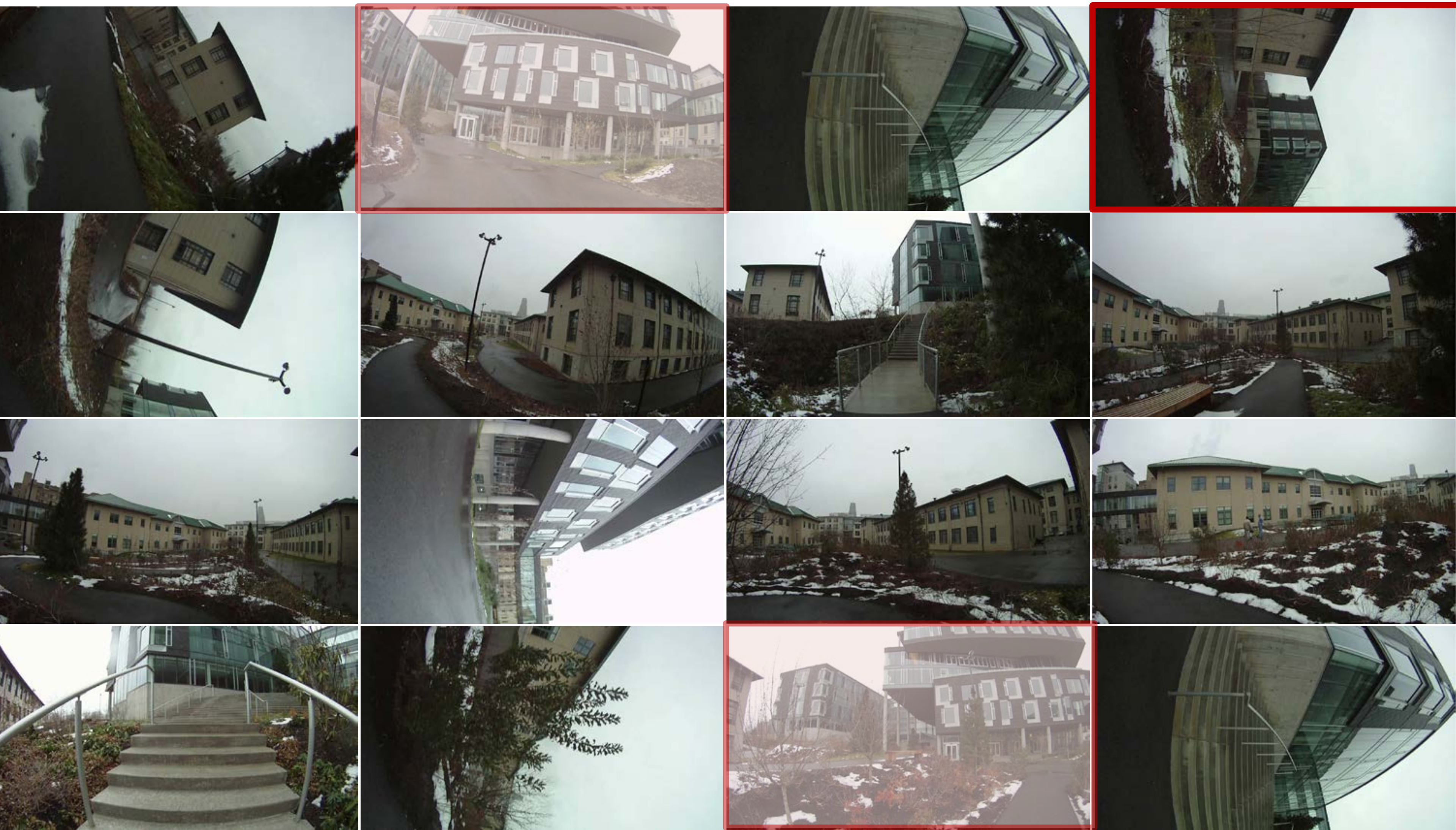




$$\begin{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ 1 \end{bmatrix}_\times \mathbf{P}_1 \\ \begin{bmatrix} \mathbf{x}_2 \\ 1 \end{bmatrix}_\times \mathbf{P}_2 \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \mathbf{0}$$

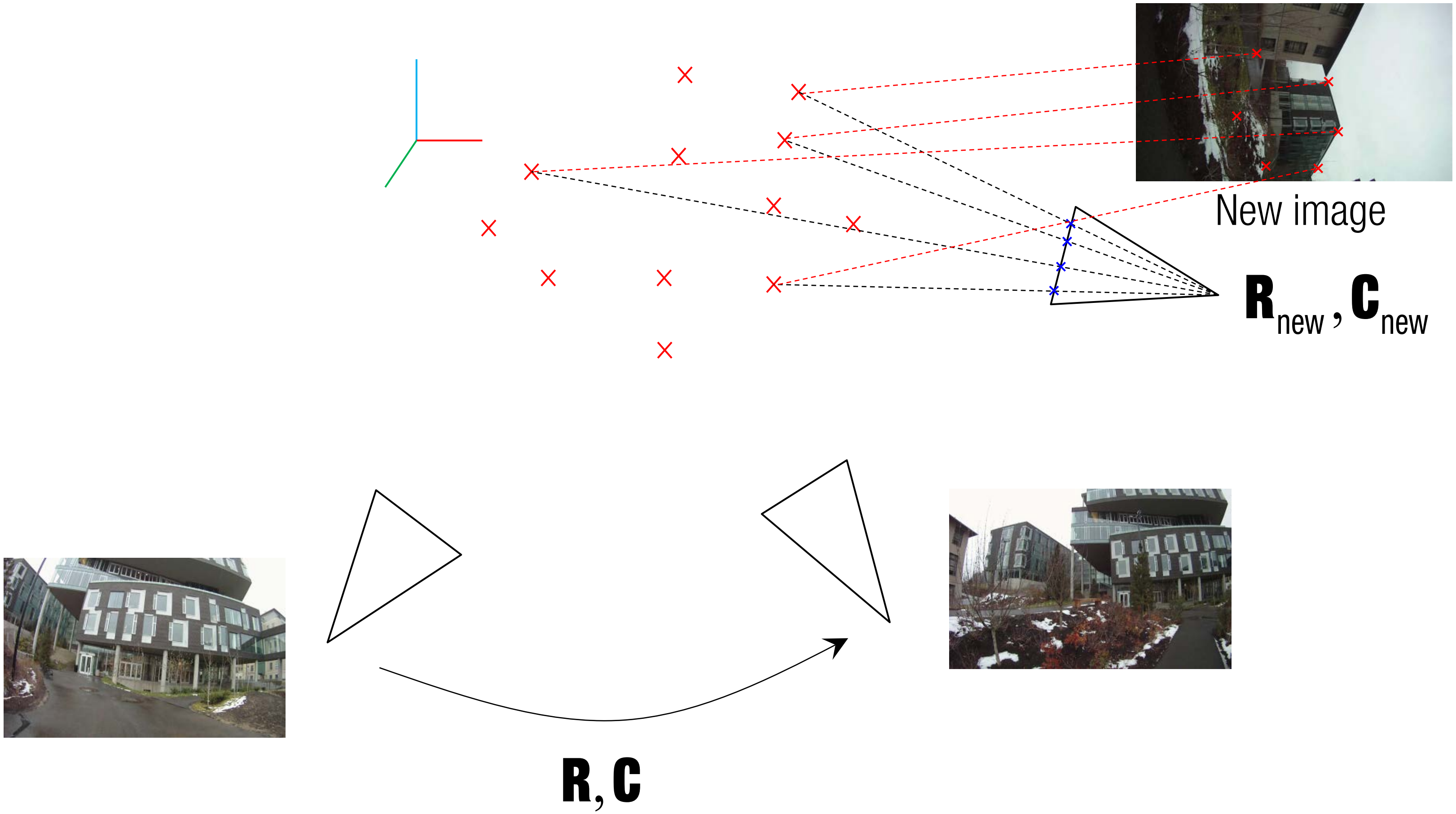
Initial Pair Images

Adding New Image



6. New Camera Registration

Perspective-n-point



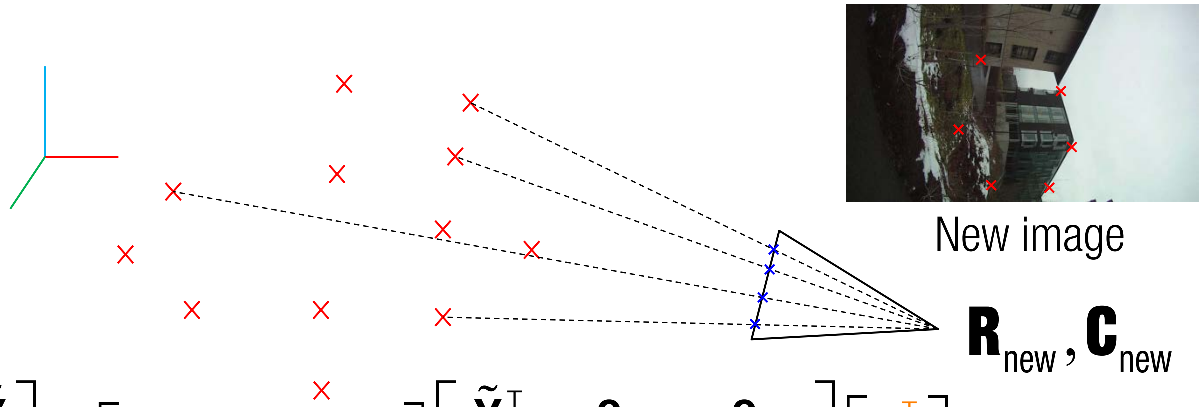
New image

$\mathbf{R}_{new}$, $\mathbf{C}_{new}$

$\mathbf{R}, \mathbf{C}$

6. New Camera Registration

Perspective-n-point

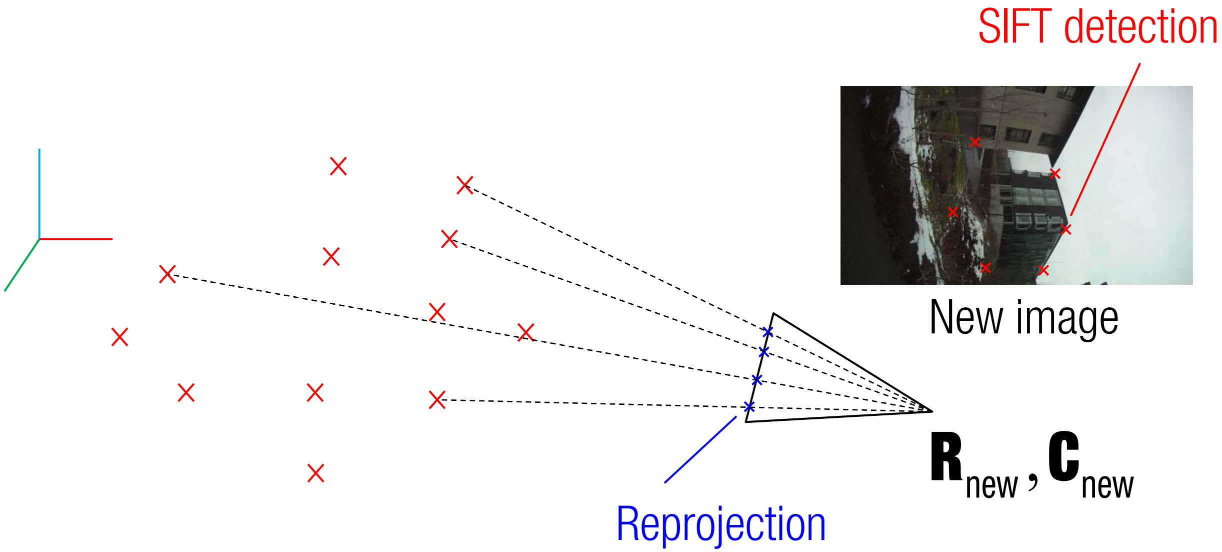


$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_\times \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} \tilde{\mathbf{X}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}_\times \begin{bmatrix} \mathbf{P}_1\tilde{\mathbf{X}} \\ \mathbf{P}_2\tilde{\mathbf{X}} \\ \mathbf{P}_3\tilde{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} 0 & -1 & v \\ 1 & 0 & -u \\ -v & u & 0 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{X}}^\mathsf{T} & \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \tilde{\mathbf{X}}^\mathsf{T} & \mathbf{0}_{1\times4} \\ \mathbf{0}_{1\times4} & \mathbf{0}_{1\times4} & \tilde{\mathbf{X}}^\mathsf{T} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1^\mathsf{T} \\ \mathbf{P}_2^\mathsf{T} \\ \mathbf{P}_3^\mathsf{T} \end{bmatrix} =$$

$$\begin{bmatrix} \mathbf{0}_{1\times4} & -\tilde{\mathbf{X}}^\mathsf{T} & v\tilde{\mathbf{X}}^\mathsf{T} \\ \tilde{\mathbf{X}}^\mathsf{T} & \mathbf{0}_{1\times4} & -u\tilde{\mathbf{X}}^\mathsf{T} \\ -v\tilde{\mathbf{X}}^\mathsf{T} & u\tilde{\mathbf{X}}^\mathsf{T} & \mathbf{0}_{1\times4} \end{bmatrix} \begin{bmatrix} \mathbf{P}_1^\mathsf{T} \\ \mathbf{P}_2^\mathsf{T} \\ \mathbf{P}_3^\mathsf{T} \end{bmatrix} = \mathbf{0}$$

3x12 matrix

7. Bundle Adjustment



SIFT detection

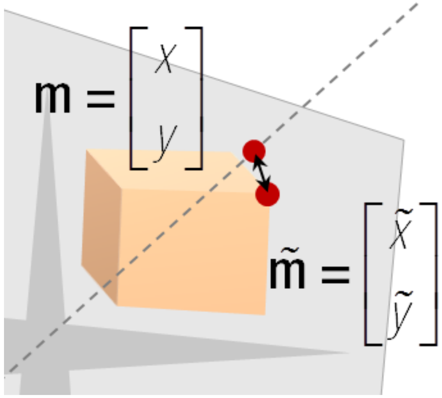
New image

$\mathbf{R}_{\text{new}}, \mathbf{C}_{\text{new}}$

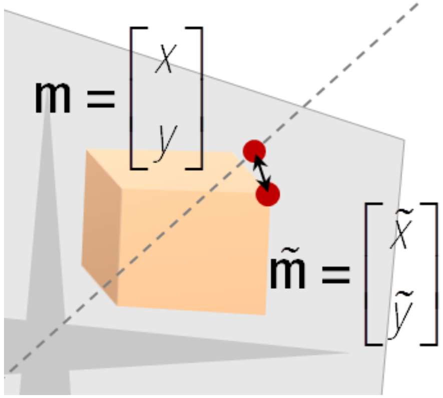
Reprojection

Reprojection error

$$e = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u/w \\ v/w \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{R}\left[\mathbf{I}_{3\times 3} \quad \mathbf{-C} \right] \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$


$\mathbf{m} = \begin{bmatrix} x \\ y \end{bmatrix}$
$\tilde{\mathbf{m}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$

Reprojection error


$\mathbf{m} = \begin{bmatrix} x \\ y \end{bmatrix}$
$\tilde{\mathbf{m}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix}$

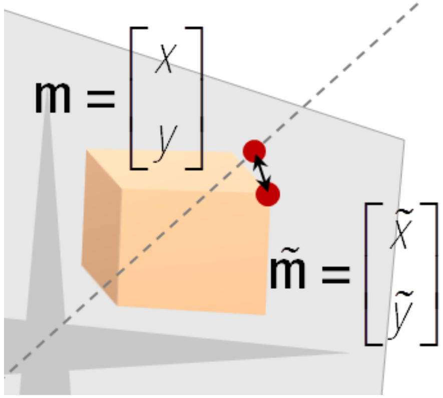
$$e = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u/w \\ v/w \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{P}\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{R}\begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix}\begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$\underset{\mathbf{q},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) / w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \\ v(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) / w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \end{bmatrix} \right\|^2 = \underset{\mathbf{q},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \mathbf{b} - \mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \right\|^2$$

$$\mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) = \begin{bmatrix} u/w \\ u/w \end{bmatrix}$$

Reprojection error

$$e = \begin{bmatrix}\tilde{x}\\ \tilde{y}\end{bmatrix} - \begin{bmatrix} u/w \\ v/w\end{bmatrix} \quad \text{where} \quad \begin{bmatrix}u\\ v\\ w\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{X}\\ 1\end{bmatrix} = \mathbf{K}\mathbf{R}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ 1\end{bmatrix}$$



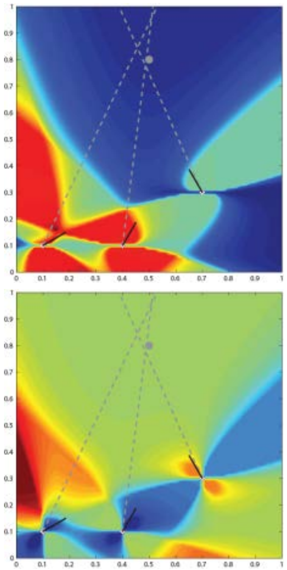
$\mathbf{m} = \begin{bmatrix}x\\ y\end{bmatrix}$
$\tilde{\mathbf{m}} = \begin{bmatrix}\tilde{x}\\ \tilde{y}\end{bmatrix}$

$$\underset{\mathbf{q},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \begin{bmatrix}\tilde{x}\\ \tilde{y}\end{bmatrix} - \begin{bmatrix} u(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})/w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \\ v(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})/w(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})\end{bmatrix} \right\|^2 = \underset{\mathbf{q},\mathbf{C},\mathbf{X}}{\text{minimize}} \left\| \mathbf{b} - \mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) \right\|^2$$

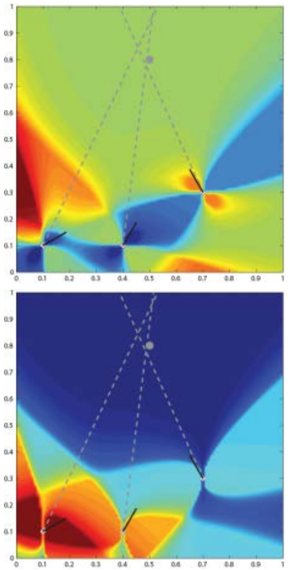
$$\mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X}) = \begin{bmatrix} u/w \\ u/w \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \underbrace{\frac{\partial \mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})}{\partial \mathbf{R}}}_{2\text{x}9} \underbrace{\frac{\partial \mathbf{R}}{\partial \mathbf{q}}}_{9\text{x}4} & \underbrace{\frac{\partial \mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})}{\partial \mathbf{C}}}_{2\text{x}3} & \underbrace{\frac{\partial \mathbf{f}(\mathbf{R}(\mathbf{q}),\mathbf{C},\mathbf{X})}{\partial \mathbf{X}}}_{2\text{x}3} \end{bmatrix}$$

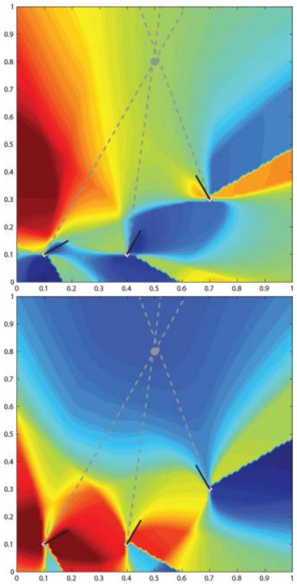
$$\mathbf{J}^{\top}\mathbf{J}\Delta\mathbf{x} = \mathbf{J}^{\top}\left(\mathbf{b} - \mathbf{f}(\mathbf{x})\right)$$



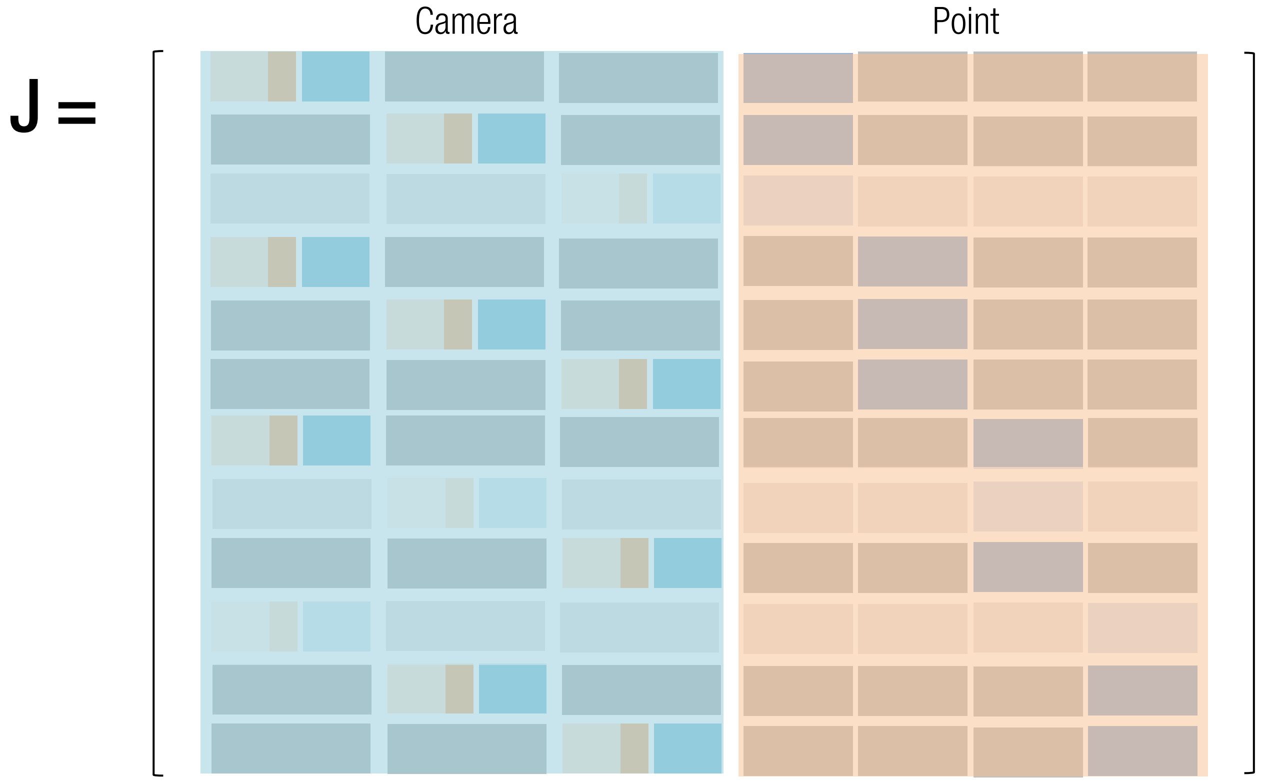
$\Delta X +$

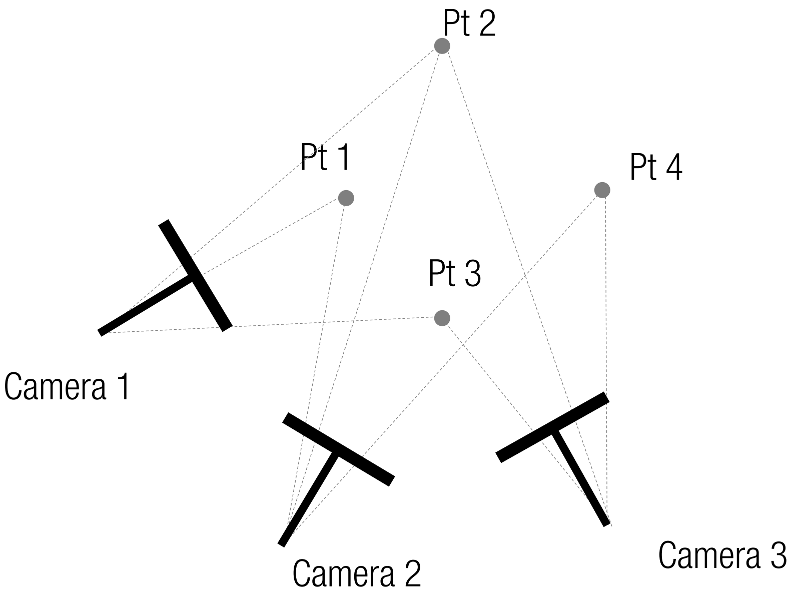


$\Delta Y =$



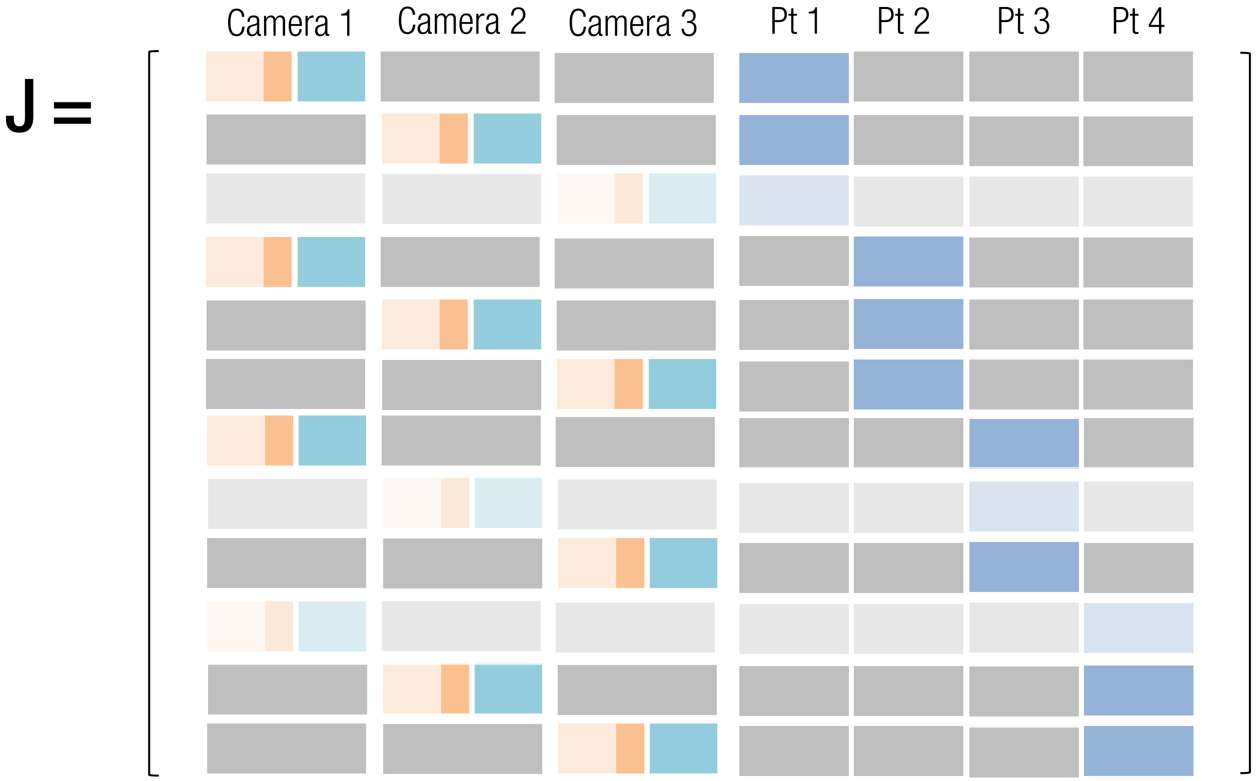
where $\Delta\mathbf{x} = \begin{bmatrix}\Delta X \\ \Delta Y\end{bmatrix}$

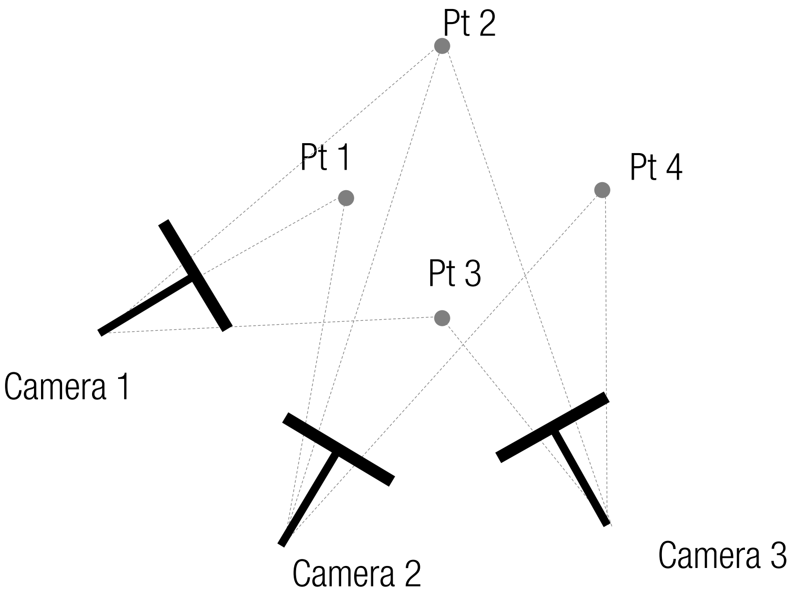




$$\Delta \mathbf{x} = \underline{\left(\mathbf{J}^{T}\mathbf{J}\right)}^{-1}\mathbf{J}^{T}\left(\mathbf{b}-\mathbf{f}(\mathbf{x})\right)$$

Main computational bottle neck

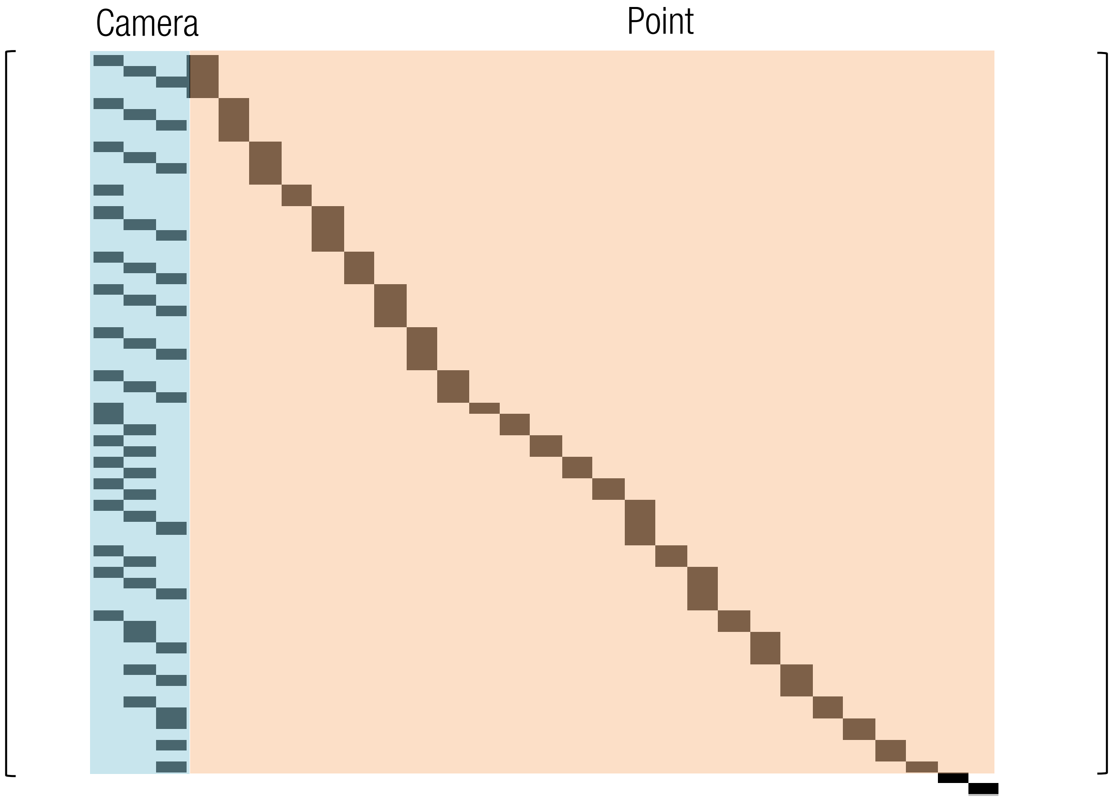


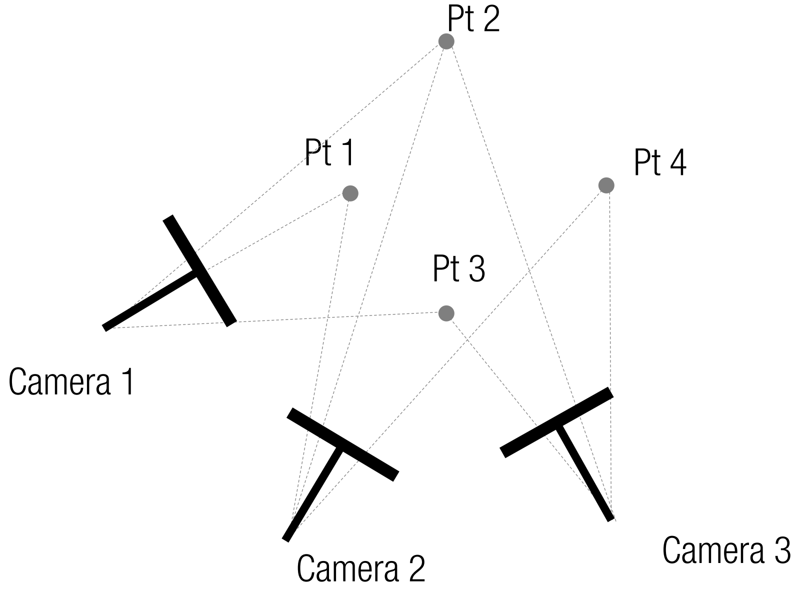


$$\Delta \mathbf{x} = \underline{\left(\mathbf{J}^{T}\mathbf{J}\right)}^{-1} \mathbf{J}^{T}\left(\mathbf{b} - \mathbf{f}(\mathbf{x})\right)$$

Main computational bottle neck

$\mathbf{J} =$

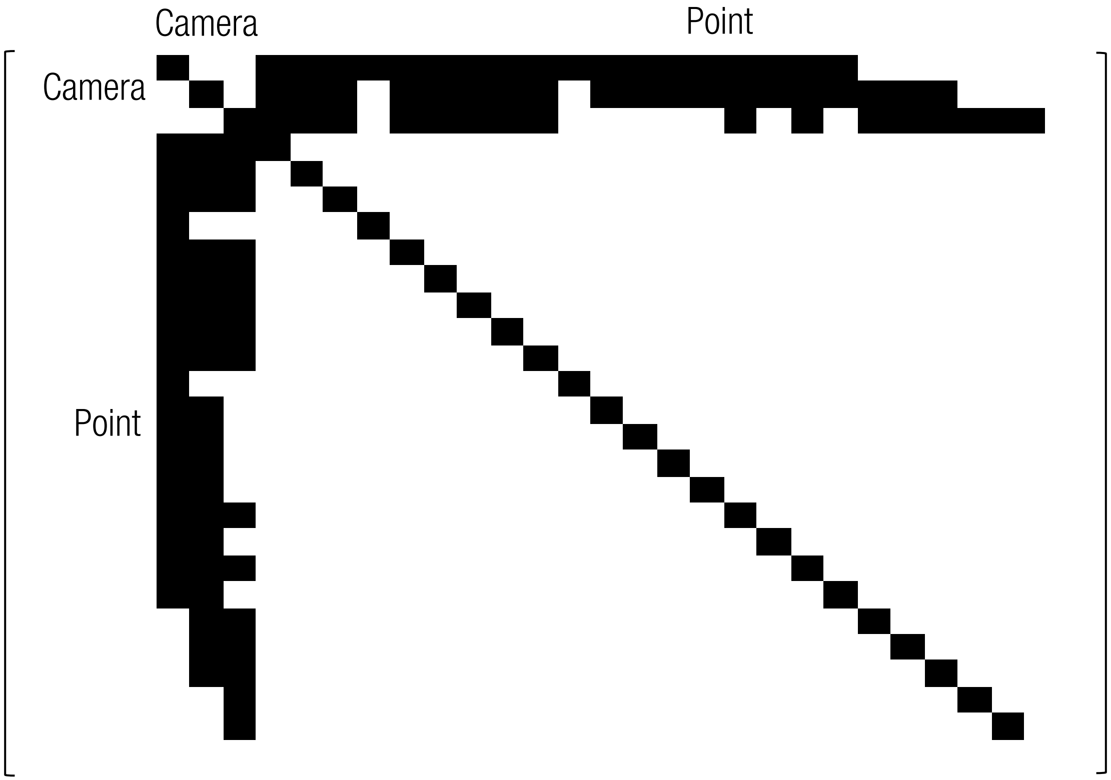


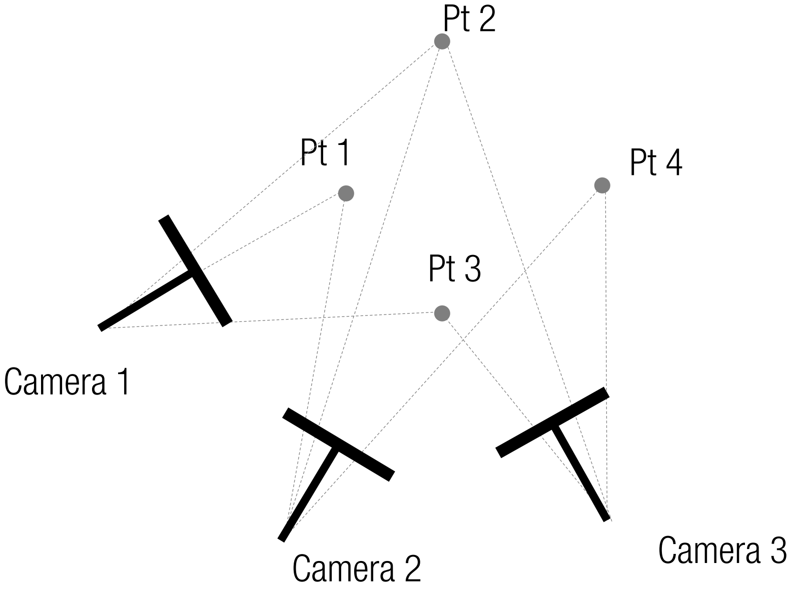


$$\Delta \mathbf{x} = \underline{\left(\mathbf{J}^{T}\mathbf{J}\right)}^{-1}\mathbf{J}^{T}\left(\mathbf{b} - \mathbf{f}(\mathbf{x})\right)$$

Main computational bottle neck

$\mathbf{J}^{T}\mathbf{J} =$

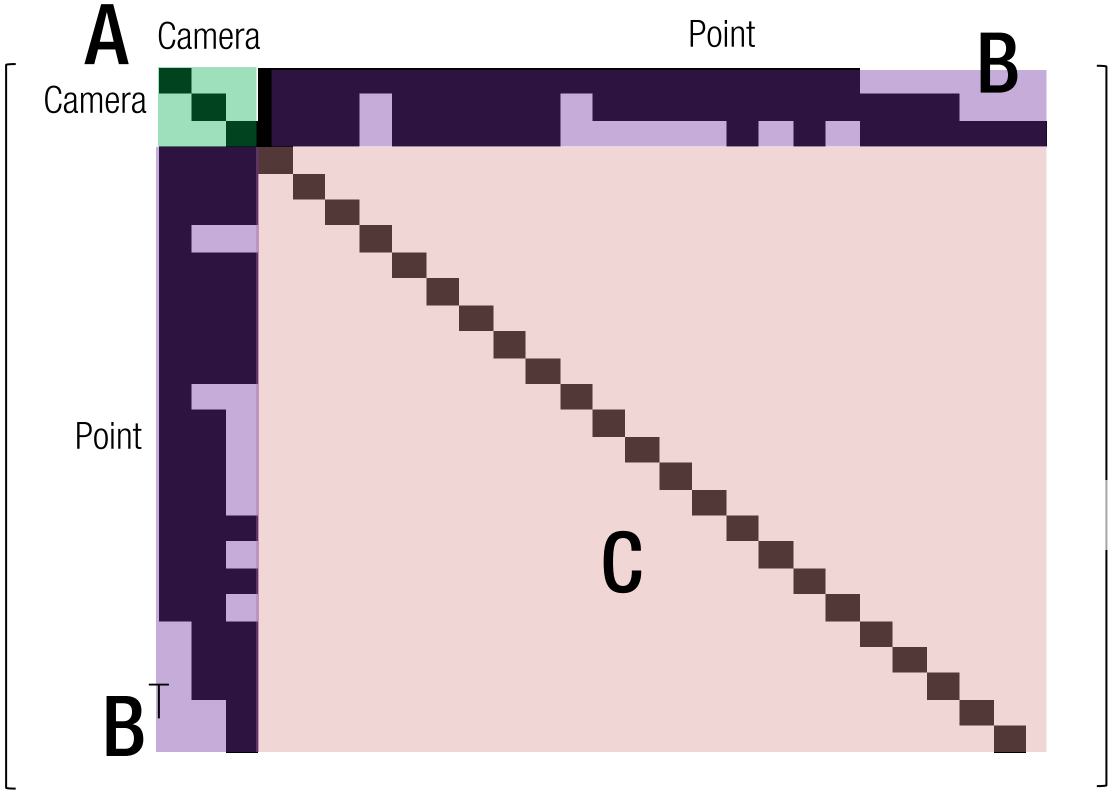




$$\Delta \mathbf{x} = \underline{\left(\mathbf{J}^{T}\mathbf{J}\right)^{-1}}\ \mathbf{J}^{T}\left(\mathbf{b} - \mathbf{f}(\mathbf{x})\right)$$

Main computational bottle neck

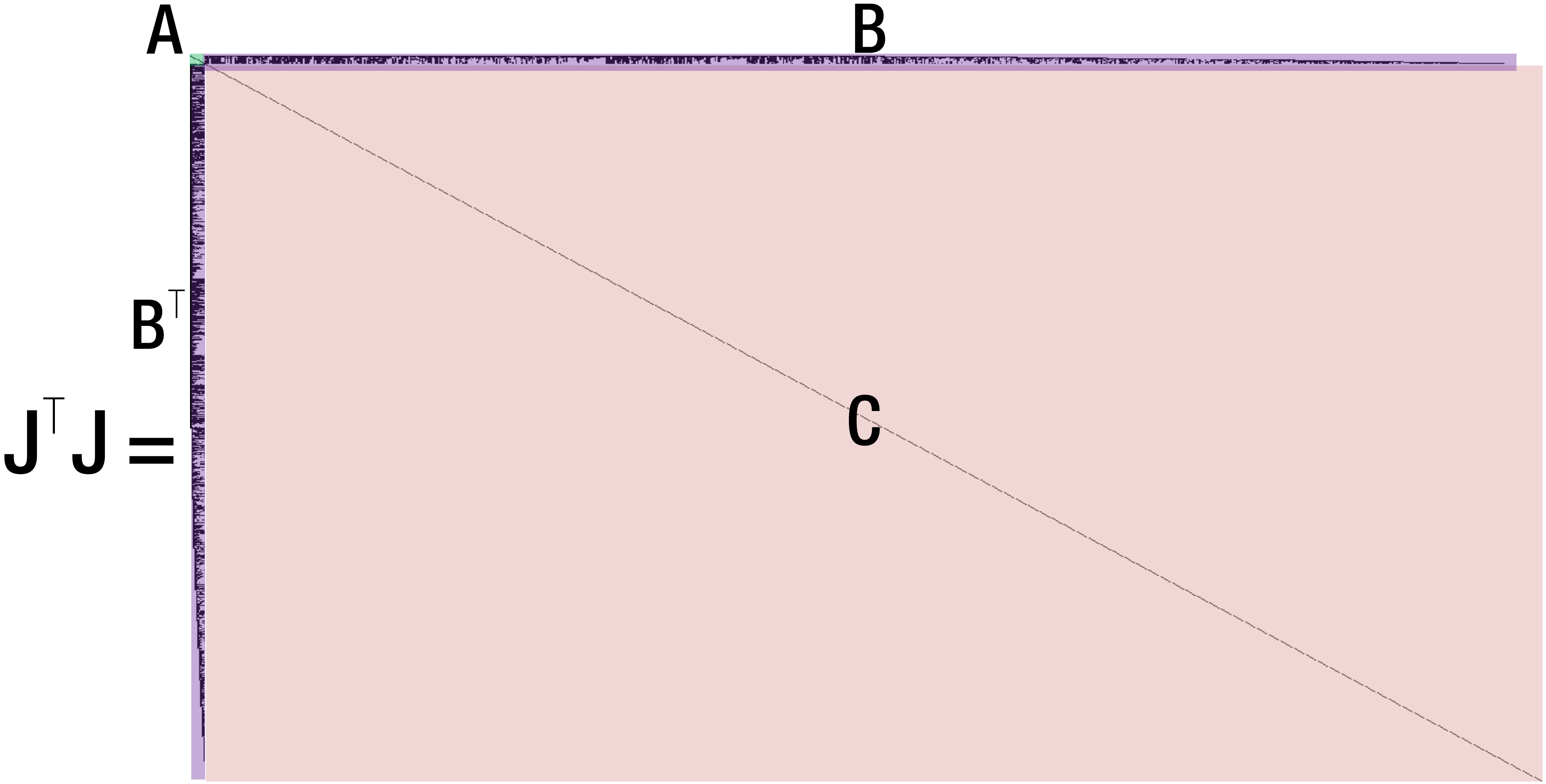
$\mathbf{J}^T\mathbf{J} =$



$$\Delta\mathbf{x} = \underbrace{\left(\mathbf{J}^T\mathbf{J}\right)^{-1}}_{\text{Main computational bottle neck}} \mathbf{J}^T\left(\mathbf{b}-\mathbf{f}(\mathbf{x})\right)$$

$$\left(\mathbf{A}-\mathbf{B}\mathbf{C}^{-1}\mathbf{B}^T\right)\Delta\mathbf{x}_c = \mathbf{e}_c - \mathbf{B}\mathbf{C}^{-1}\mathbf{e}_p$$: Reduced system

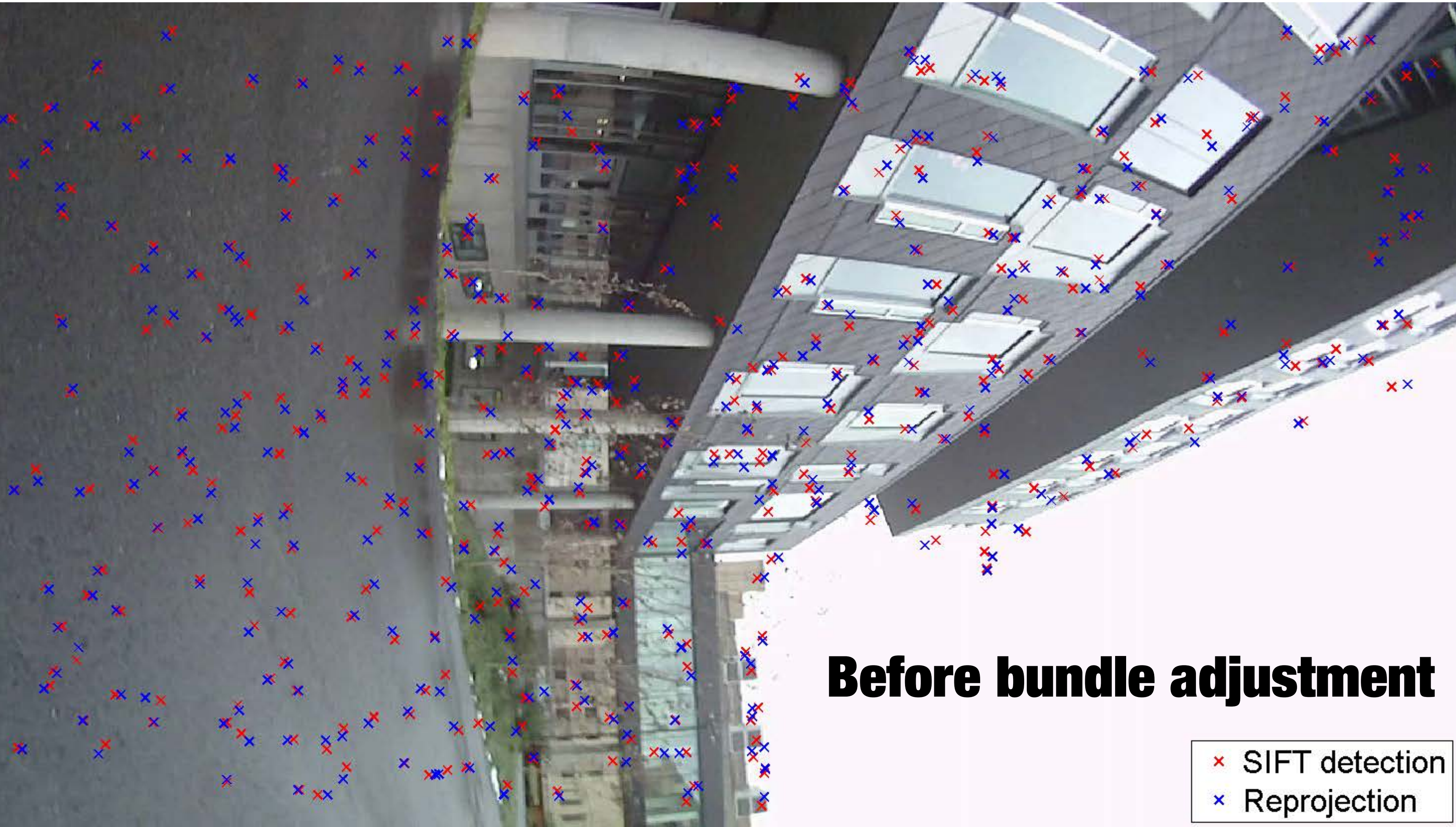
$$\mathbf{C}\Delta\mathbf{x}_p = \mathbf{e}_p - \mathbf{B}^T\Delta\mathbf{x}_c$$: Back substitution



$$\left(\mathbf{A} - \mathbf{B}\mathbf{C}^{-1}\mathbf{B}^{T}\right)\Delta\mathbf{x}_{c} = \mathbf{e}_{c} - \mathbf{B}\mathbf{C}^{-1}\mathbf{e}_{p}$$: Reduced system

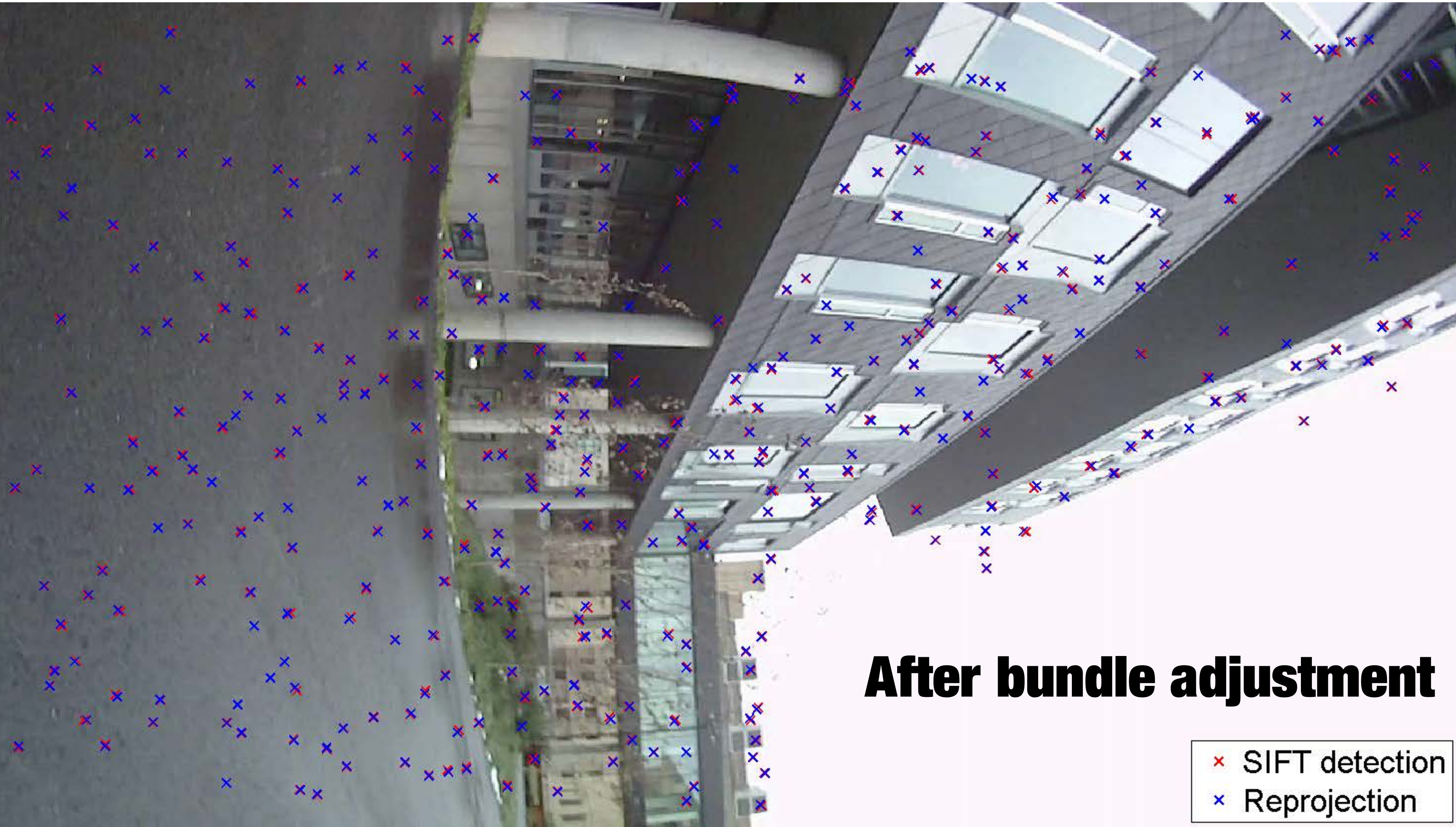
$$\mathbf{C}\Delta\mathbf{x}_{p} = \mathbf{e}_{p} - \mathbf{B}^{T}\Delta\mathbf{x}_{c}$$: Back substitution

7. Bundle Adjustment



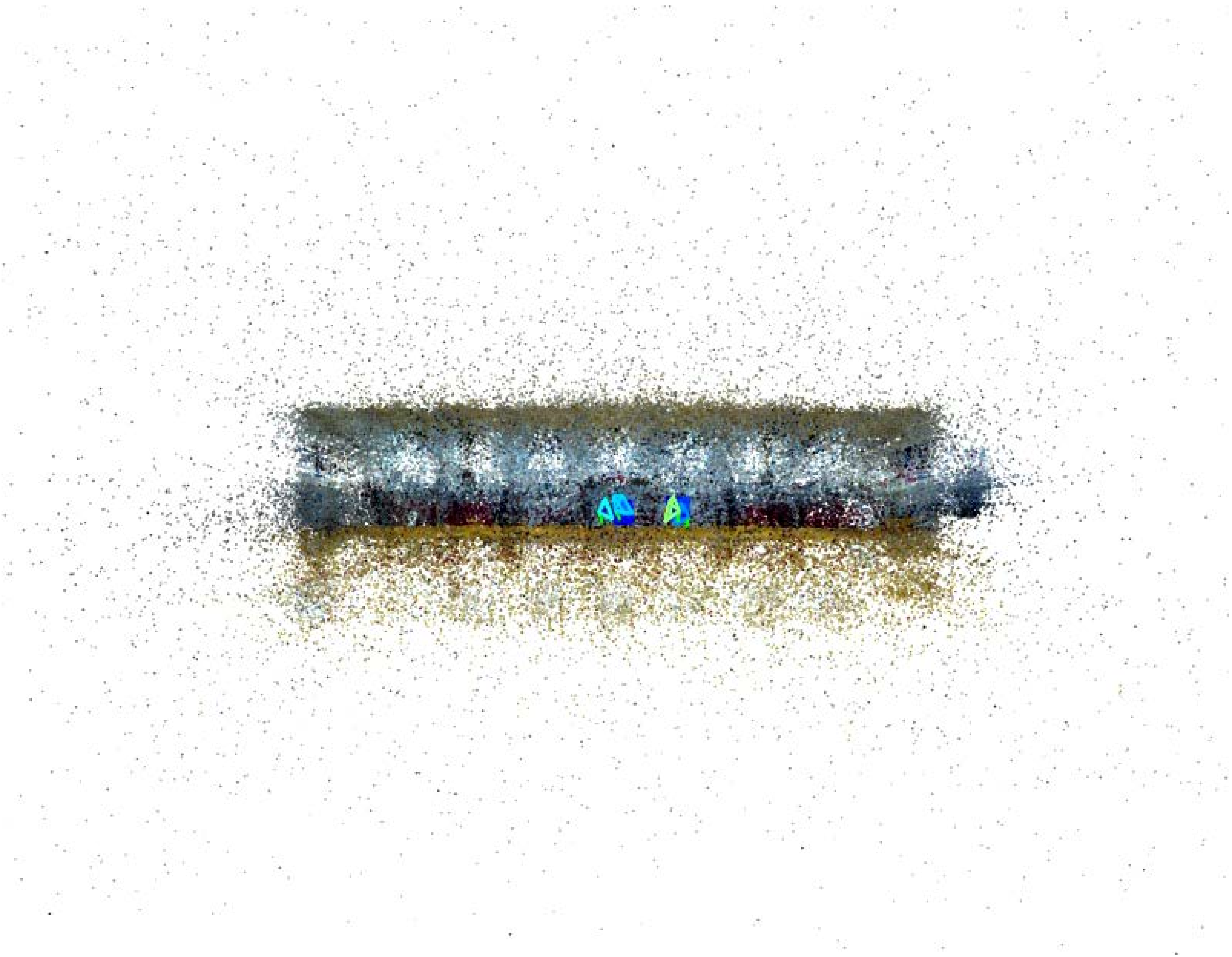
Before bundle adjustment

7. Bundle Adjustment



After bundle adjustment

GO SPARTANS

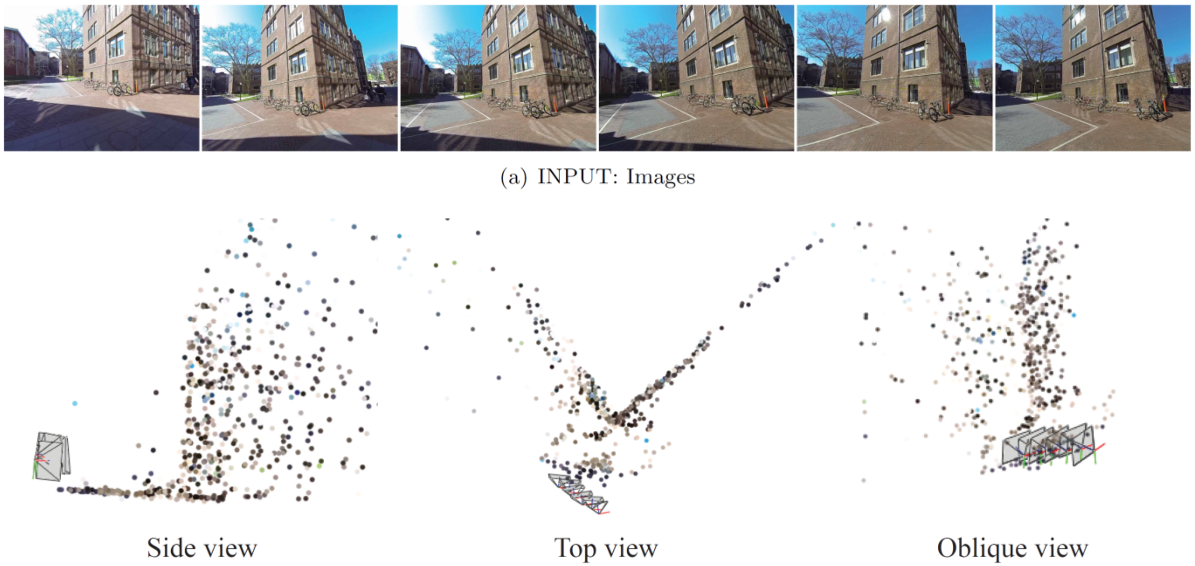


Structure from Motion Project

Project 2: Structure from Motion

CIS 580, Machine Perception, Spring 2015
Preliminary report due: 2015.04.20. 11:59AM
Final Due: 2015.05.04. 11:59AM

This project aims to reconstruct a 3D point cloud and camera poses of 6 images as shown in Figure 1. Your task is to implement the full pipeline of structure from motion including two view reconstruction, triangulation, PnP, and bundle adjustment. For nonlinear optimization parts, you are free to choose an optimizer such as built-in functions in MATLAB or Sparse Bundle Adjustment package (http://users.ics.forth.gr/∼lourakis/sba/). Input images are taken by a GoPro Hero 3 camera (Black Edition) and fisheye lens distortion is corrected. We also provide correspondences between all possible pairs of images, i.e, $\mathcal{I}_i \leftrightarrow \mathcal{I}_j$ for $\forall i, j$ where $\mathcal{I}_i$ is the i^{th} image. In Figure 1(b), 6 cameras and 1459 points are reconstructed in 3D.



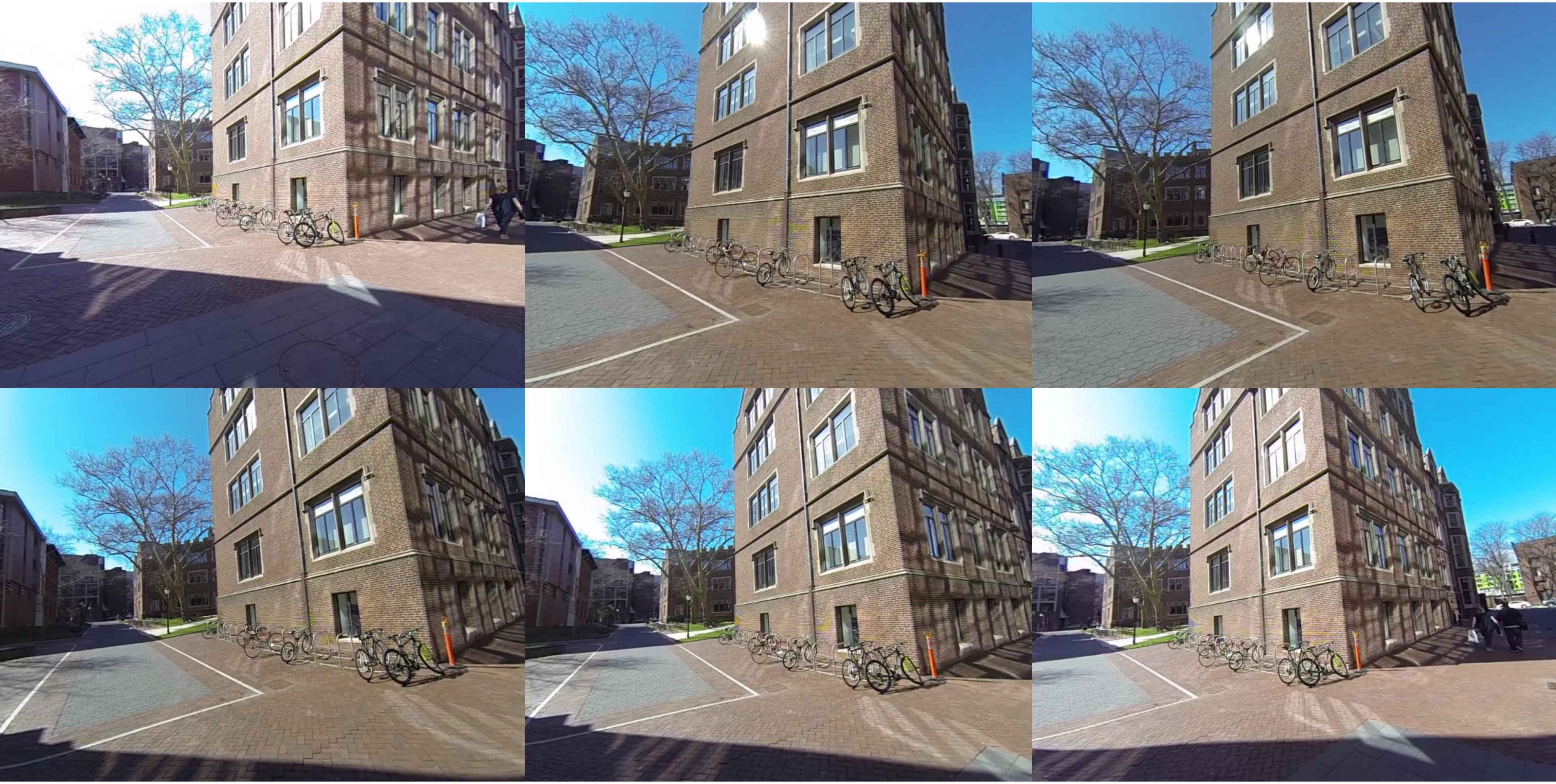
(a) INPUT: Images

(b) OUTPUT: 3D reconstruction

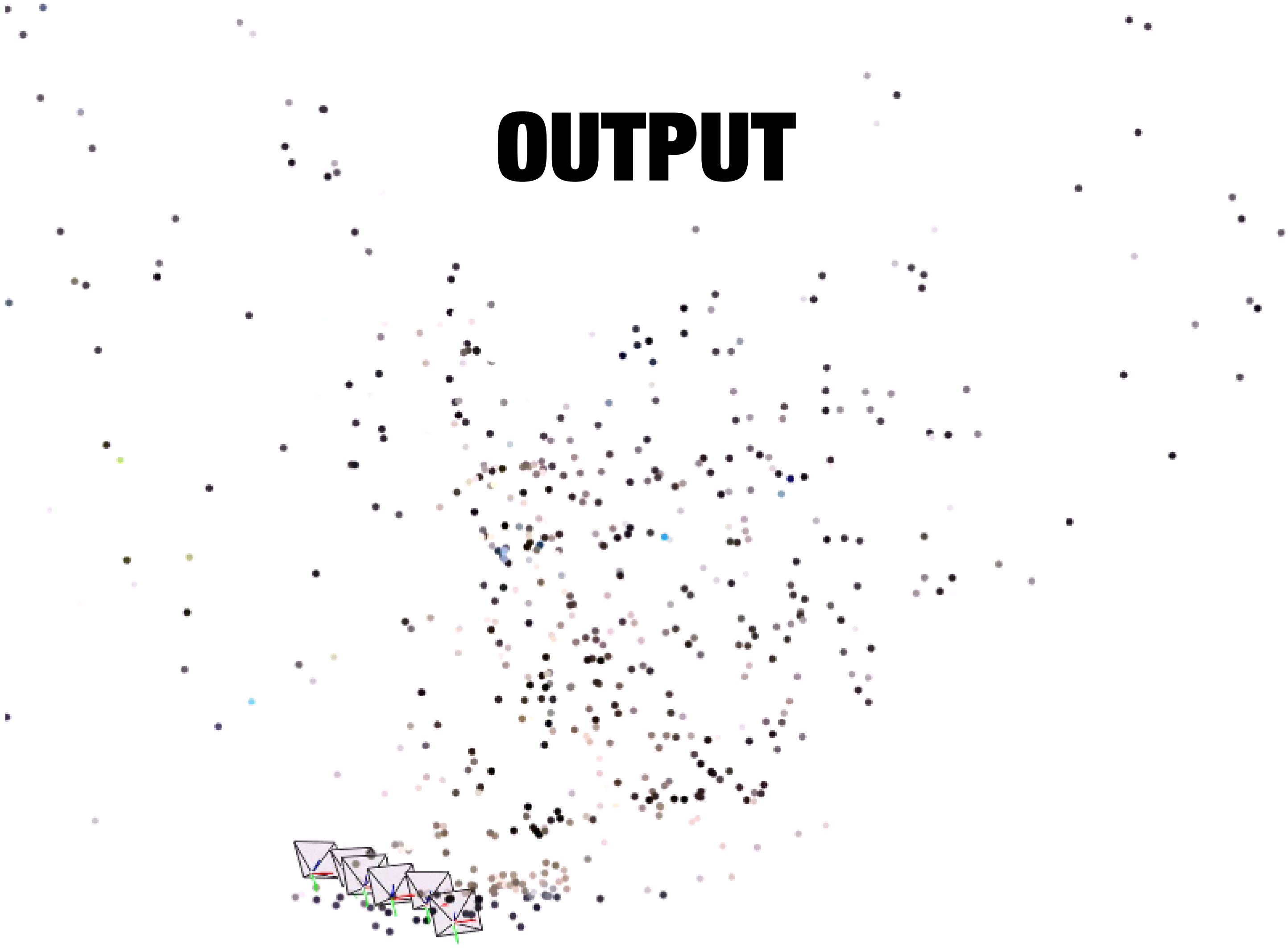
Figure 1: (a) Given 6 images of space in front of Levine Hall, (b) reconstruct 3D point cloud and camera poses.

Data repository: `cis.upenn.edu/~cis580/Spring2015/Projects/proj2/SfMProjectData.zip`
Submission: Submit your complete code to turnin

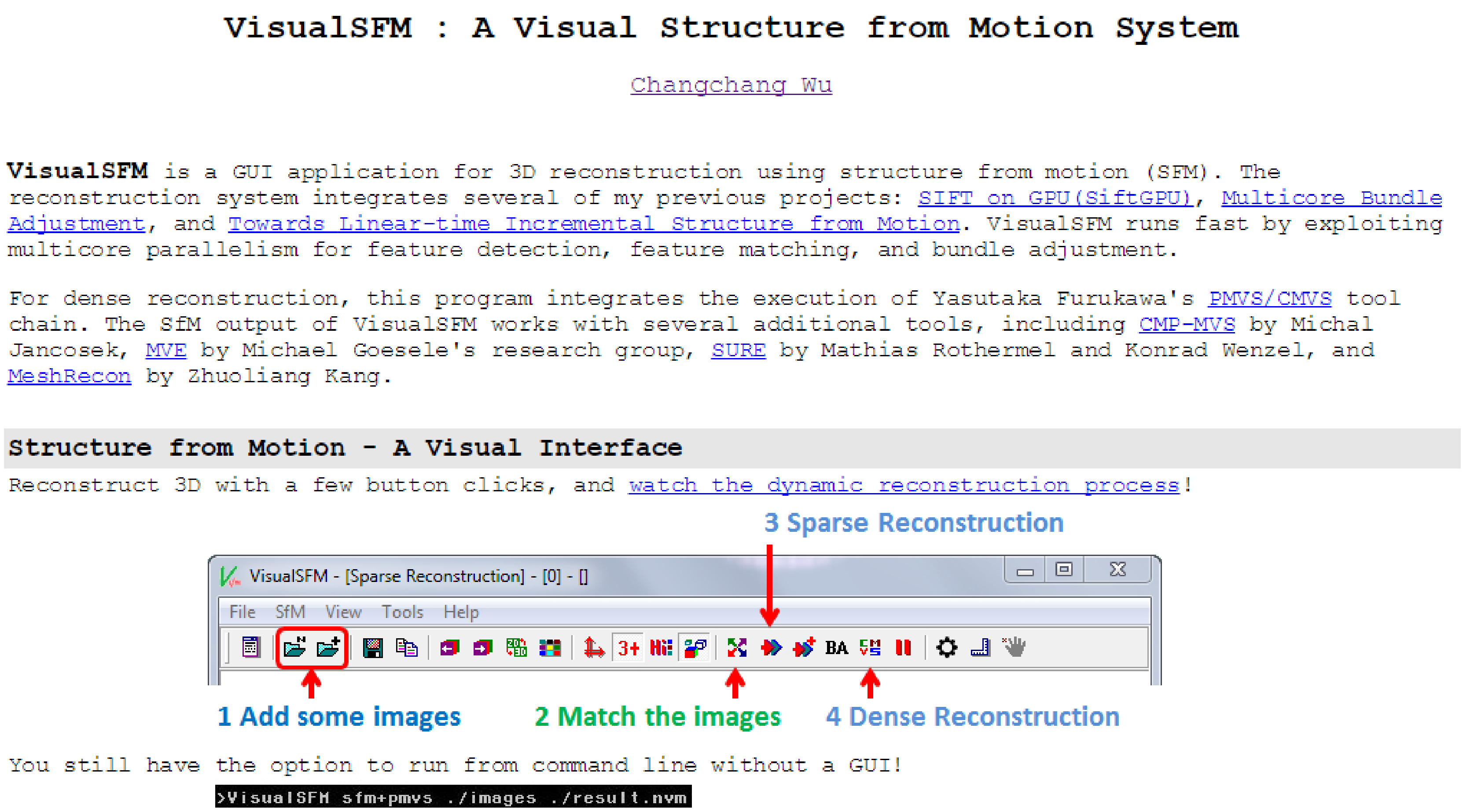
INPUT



OUTPUT



Comparison with VisualSfM



VisualSFM : A Visual Structure from Motion System

Changchang Wu

VisualSFM is a GUI application for 3D reconstruction using structure from motion (SFM). The reconstruction system integrates several of my previous projects: SIFT on GPU(SiftGPU), Multicore Bundle Adjustment, and Towards Linear-time Incremental Structure from Motion. VisualSFM runs fast by exploiting multicore parallelism for feature detection, feature matching, and bundle adjustment.

For dense reconstruction, this program integrates the execution of Yasutaka Furukawa's PMVS/CMVS tool chain. The SfM output of VisualSFM works with several additional tools, including CMP-MVS by Michal Jancosek, MVE by Michael Goesele's research group, SURE by Mathias Rothermel and Konrad Wenzel, and MeshRecon by Zhuoliang Kang.

Structure from Motion - A Visual Interface

Reconstruct 3D with a few button clicks, and watch the dynamic reconstruction process!

You still have the option to run from command line without a GUI!

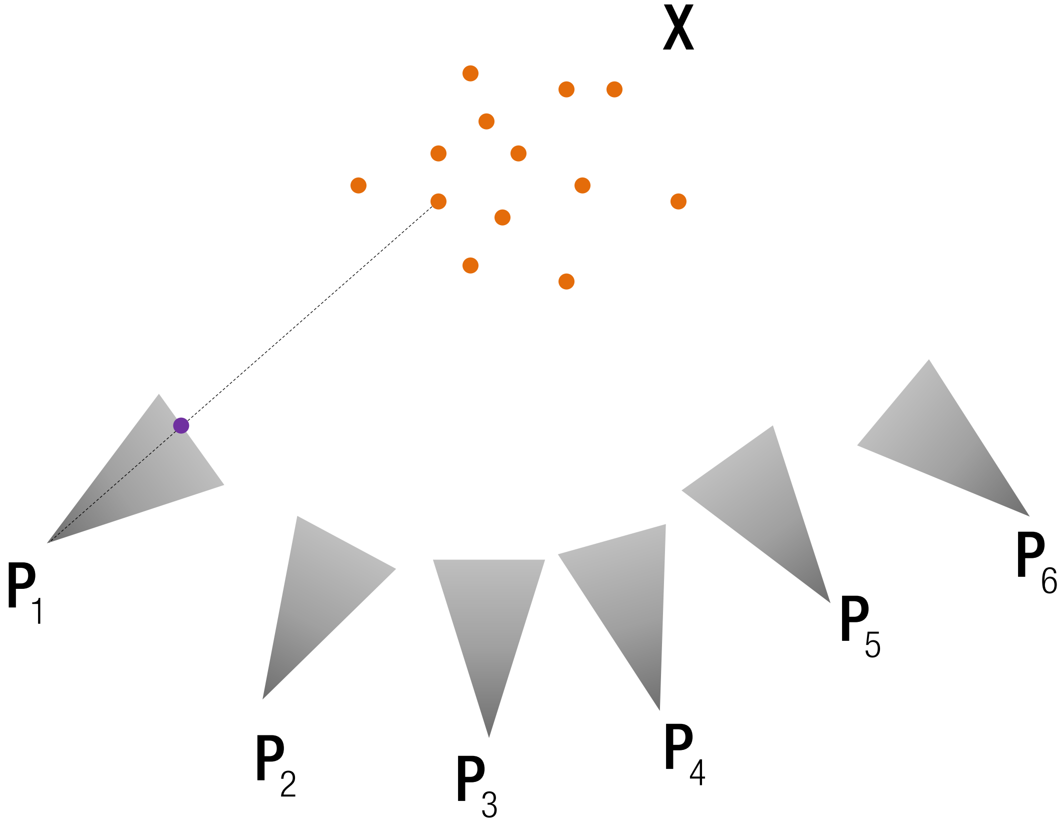
```
>VisualSFM sfm+pmvs ./images ./result.nvm
```

http://ccwu.me/vsfm/

Unknowns

Parameters to Estimate

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



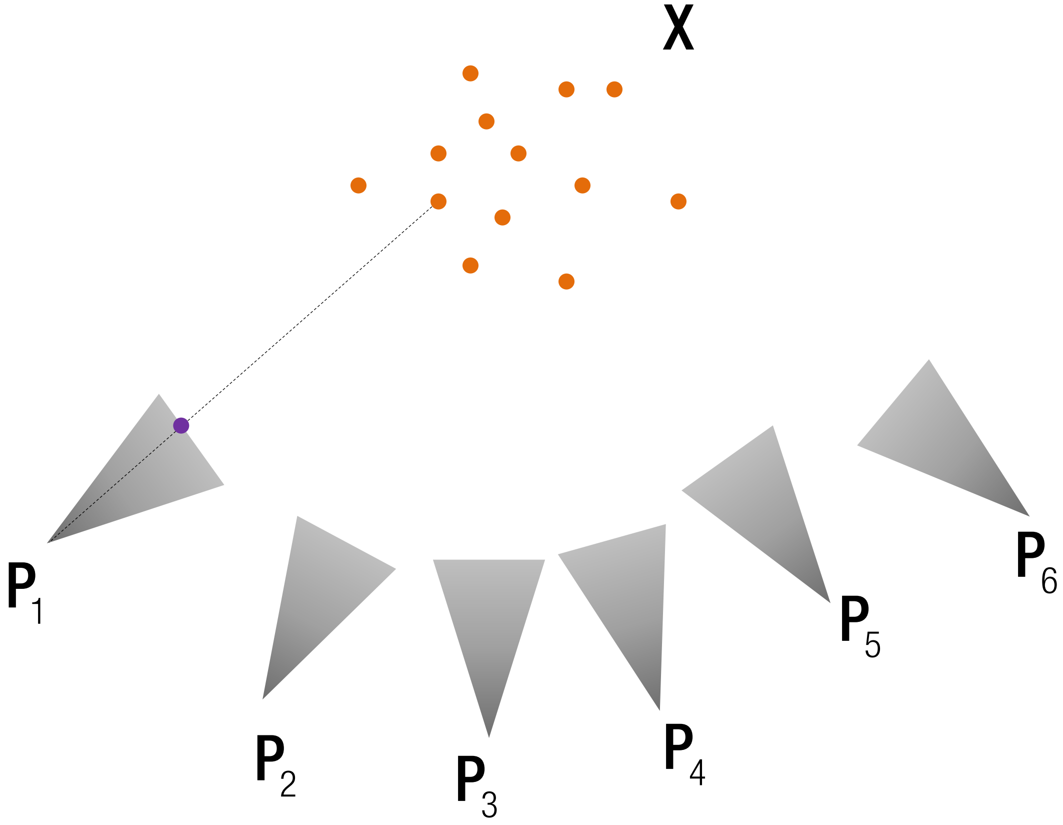
$$\begin{aligned} \mathbf{P} &= \mathbf{K}\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \\ &= \mathbf{K}\mathbf{R}\begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{C} \end{bmatrix} \end{aligned}$$

Unknowns

Parameters to Estimate

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



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

$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{C} \end{bmatrix}$$

Unknowns

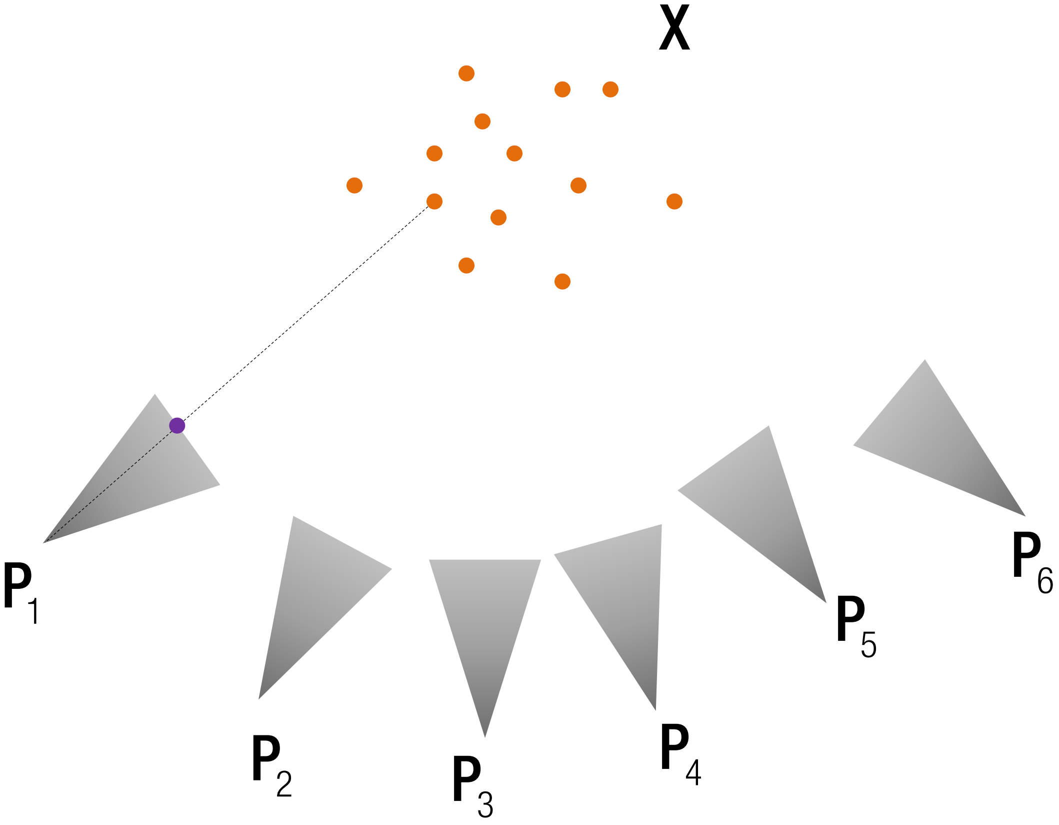
Parameters to Estimate

$$\lambda\begin{bmatrix}u\\v\\1\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix}$$

$$= \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix}$$

$$= \mathbf{K}\mathbf{R}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\1\end{bmatrix}$$

Unknowns: **R**, **C**, and **X**



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

$$= \mathbf{K}\mathbf{R}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}$$

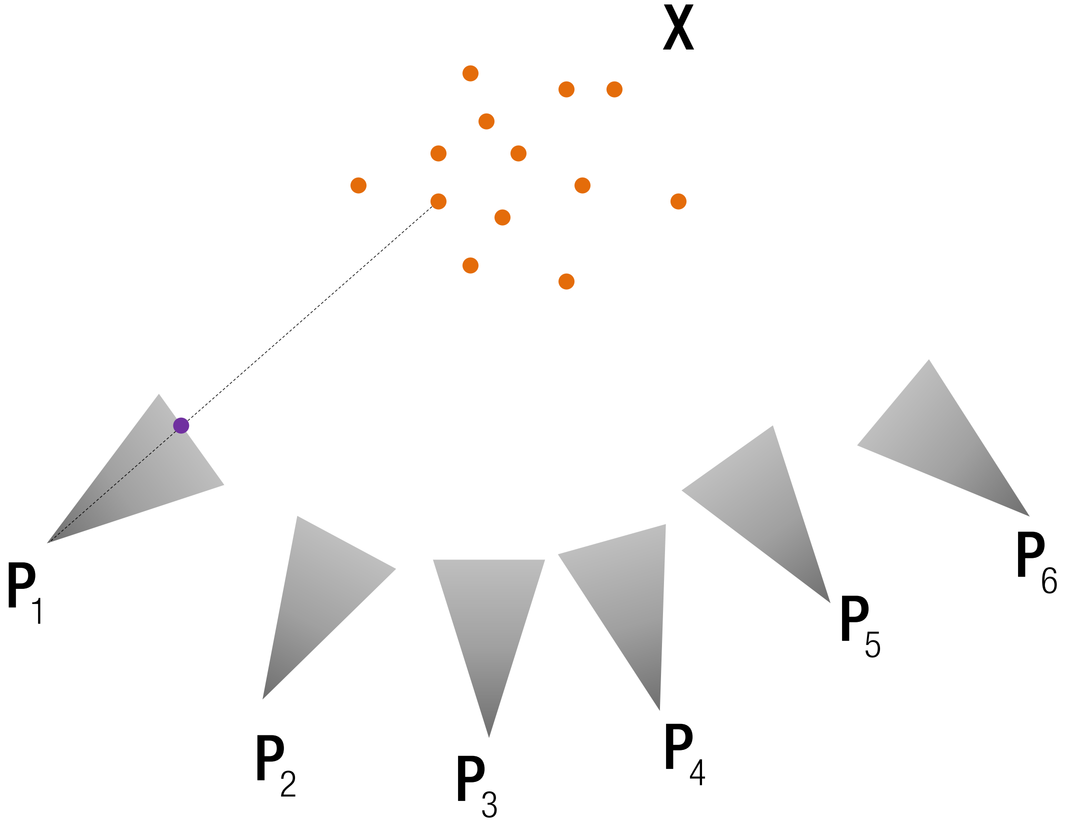
Knowns

Measurements

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



Unknowns: **R**, **C**, and **X**

Knowns: *u*, *v*, and **K**

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

$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix}$$

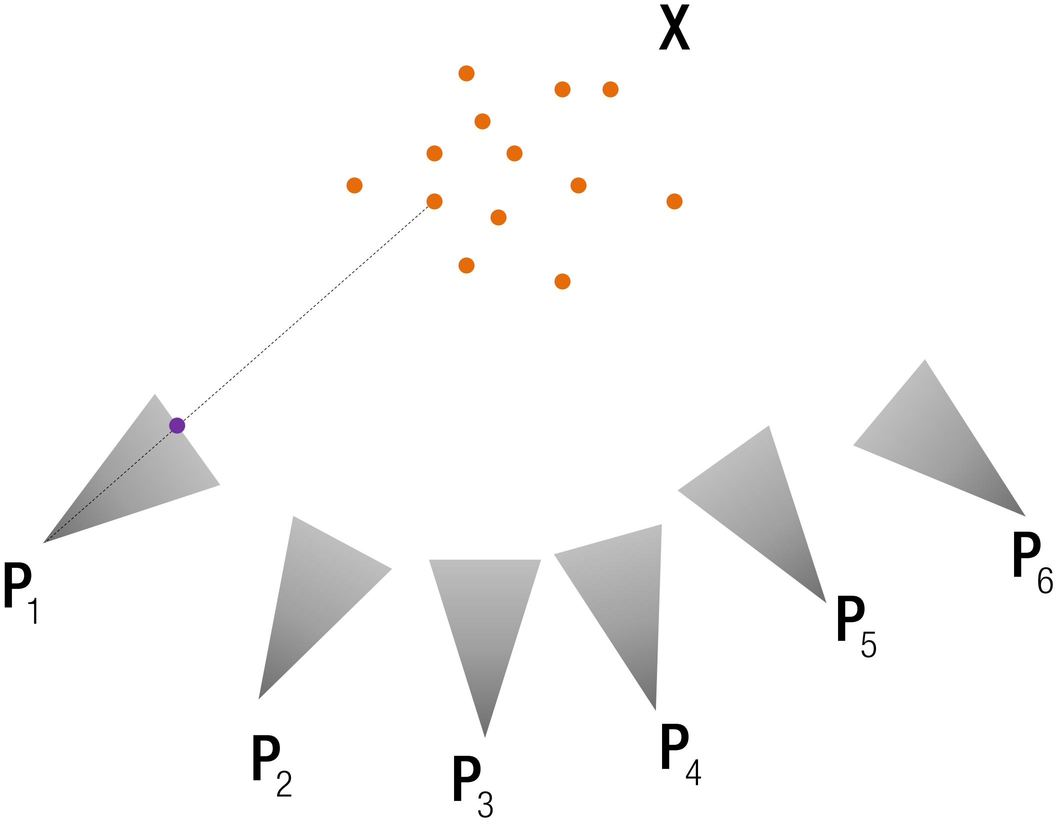
Knowns

Intrinsic Parameter

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \underbrace{\mathbf{K}}_{\text{precalibrated}}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



calibration.txt: intrinsic parameter

K = [568.996140852 0 643.21055941;
0 568.988362396 477.982801038;
0 0 1]

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

$$= \mathbf{K}\mathbf{R}\begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix}$$

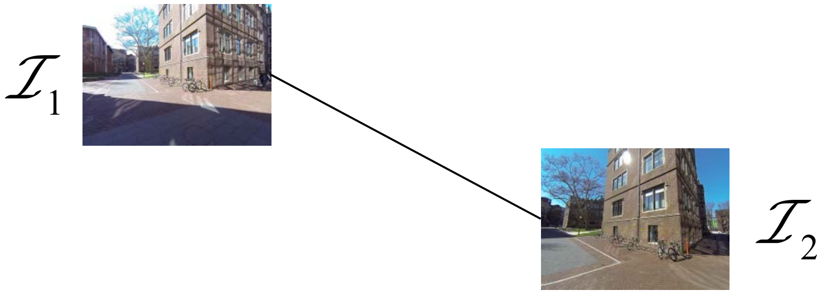
Knowns

Matching Across Images

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

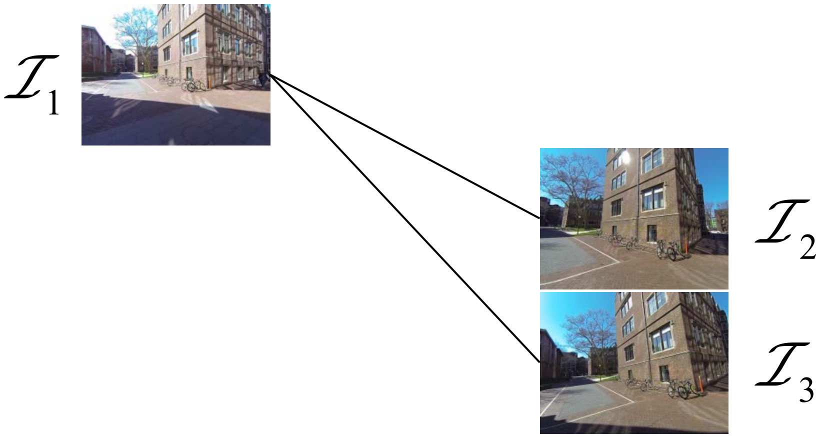
$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



Knowns

Matching Across Images

$$
\begin{aligned}
\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \\
&= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \\
&= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times 3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}
\end{aligned}
$$



Knowns

Matching Across Images

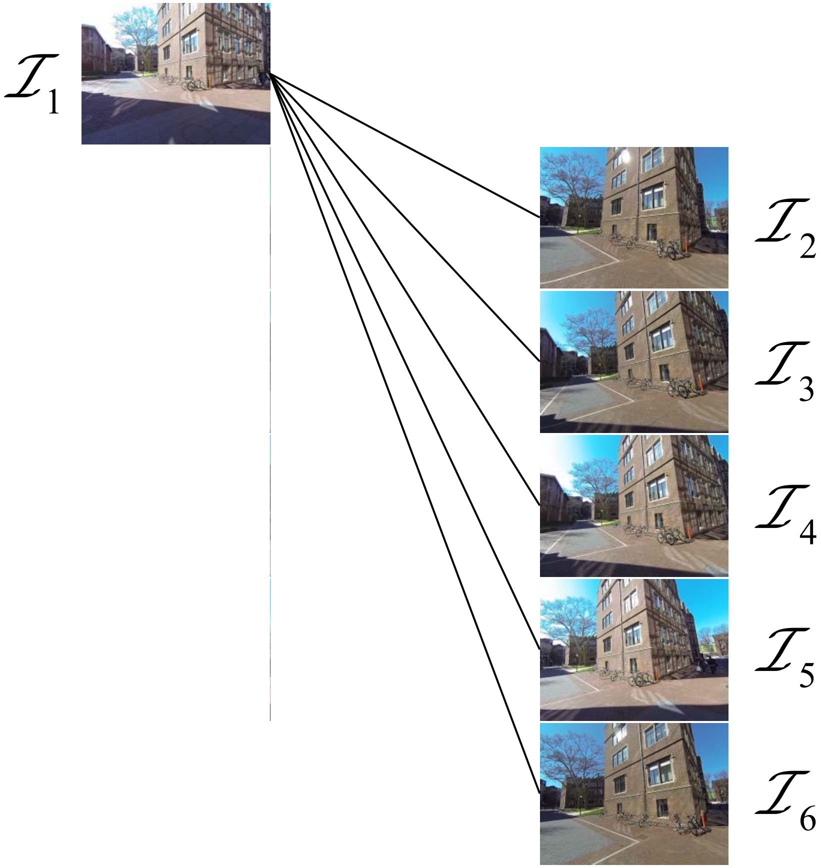
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Matching data:

matching1.txt



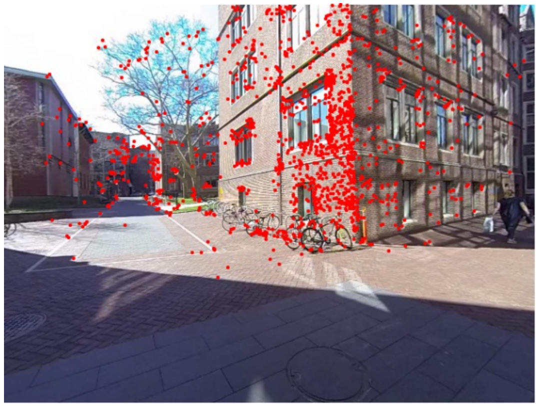
Knowns

Matching File Format

Number of feature points in image1

matching1.txt

nFeatures: 2002



$\mathcal{I}_1$

Knowns

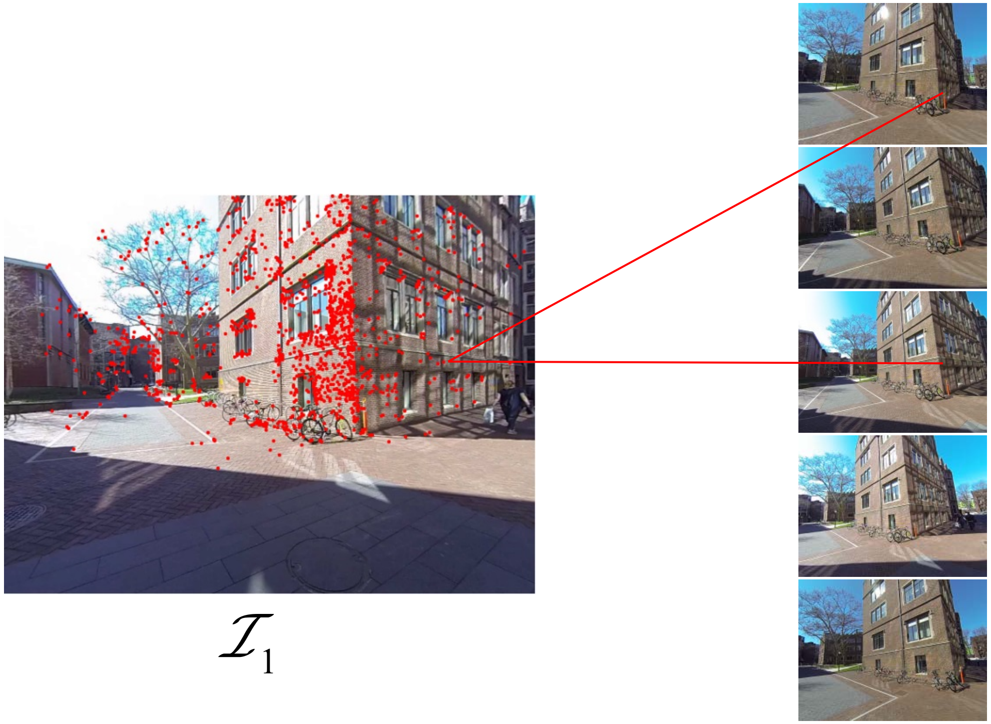
Matching File Format

Number of feature points in image1

matching1.txt

```
nFeatures: 2002
3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000
```

Number of matches across images



Knowns

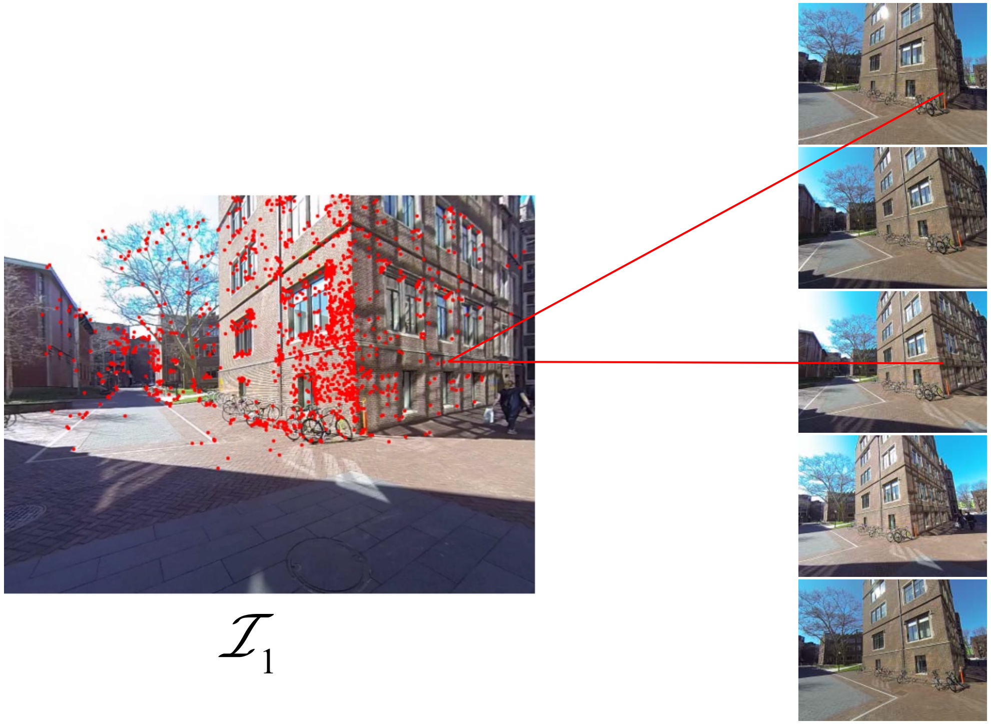
Matching File Format

Number of feature points in image1

matching1.txt

```
nFeatures: 2002
3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000
```

RGB values



$\mathcal{I}_1$

Knowns

Matching File Format

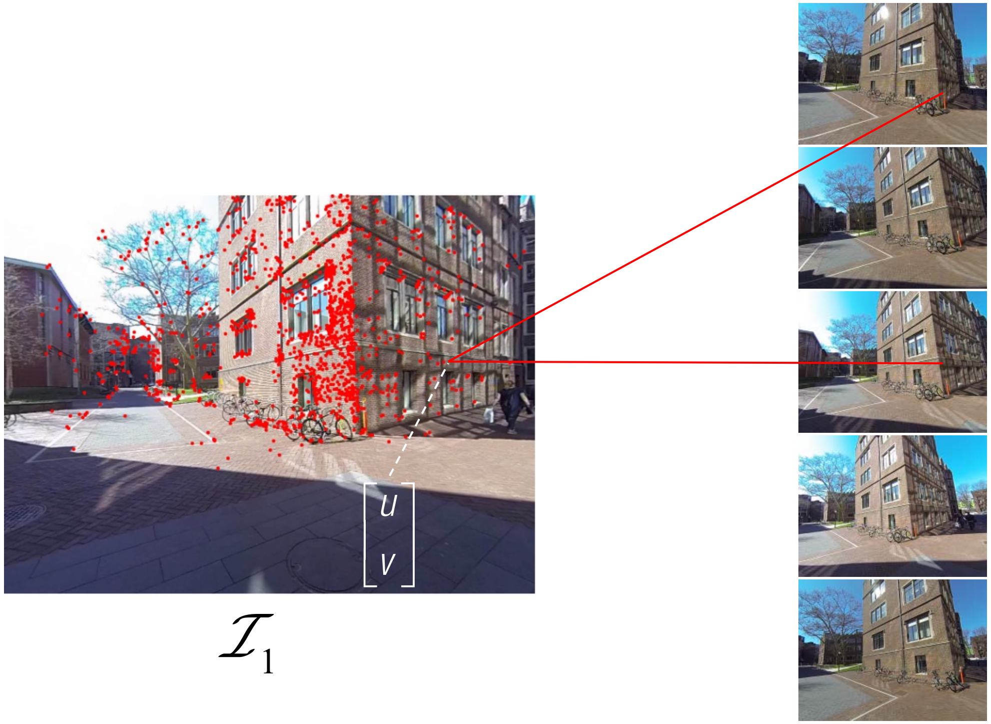
Number of feature points in image1

matching1.txt

nFeatures: 2002

3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000

Feature location (u,v)



Knowns

Matching File Format

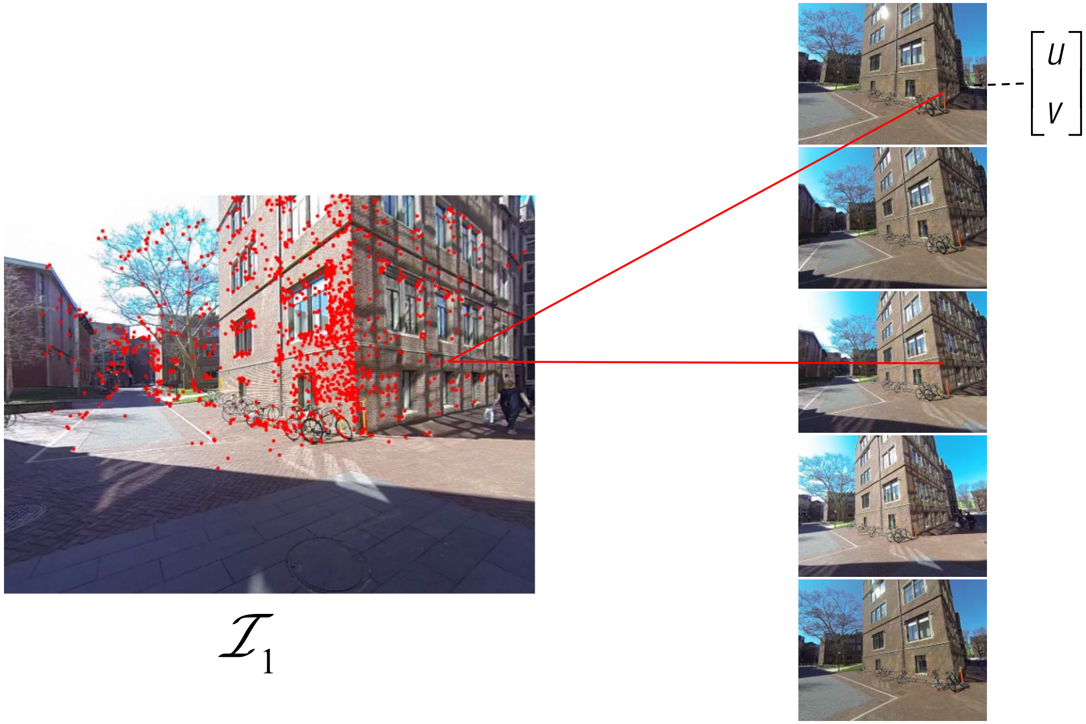
Number of feature points in image1

matching1.txt

nFeatures: 2002

3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000

Feature location (u,v) in the second image



Knowns

Matching File Format

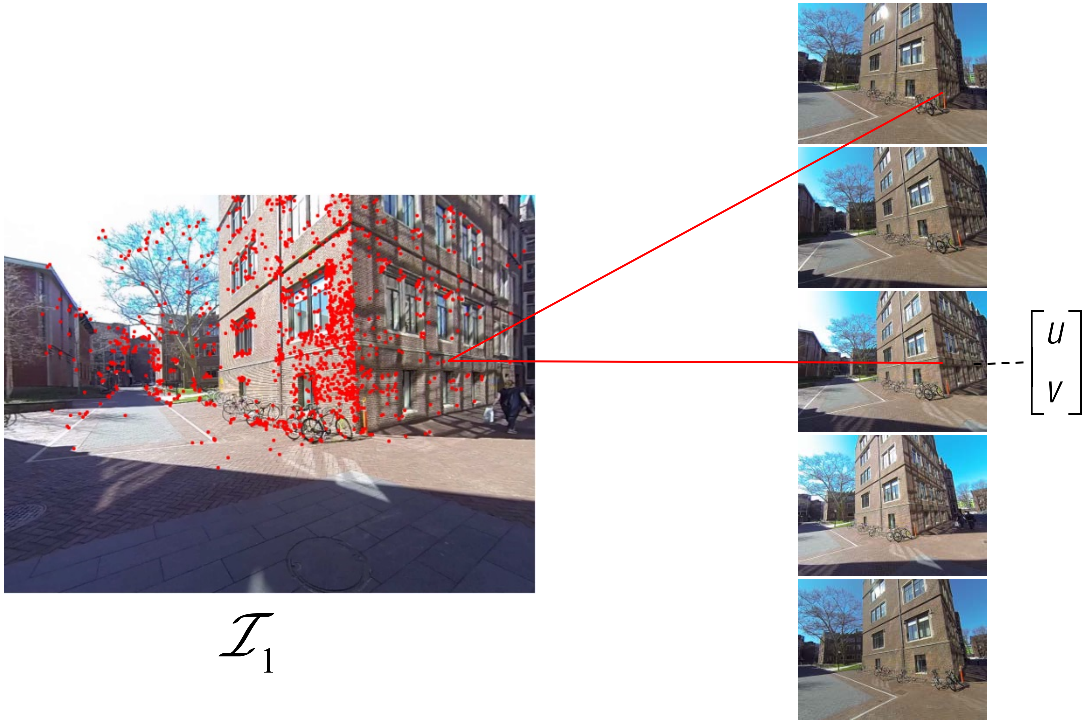
Number of feature points in image1

matching1.txt

nFeatures: 2002

3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000

Feature location (u,v) in the fourth image



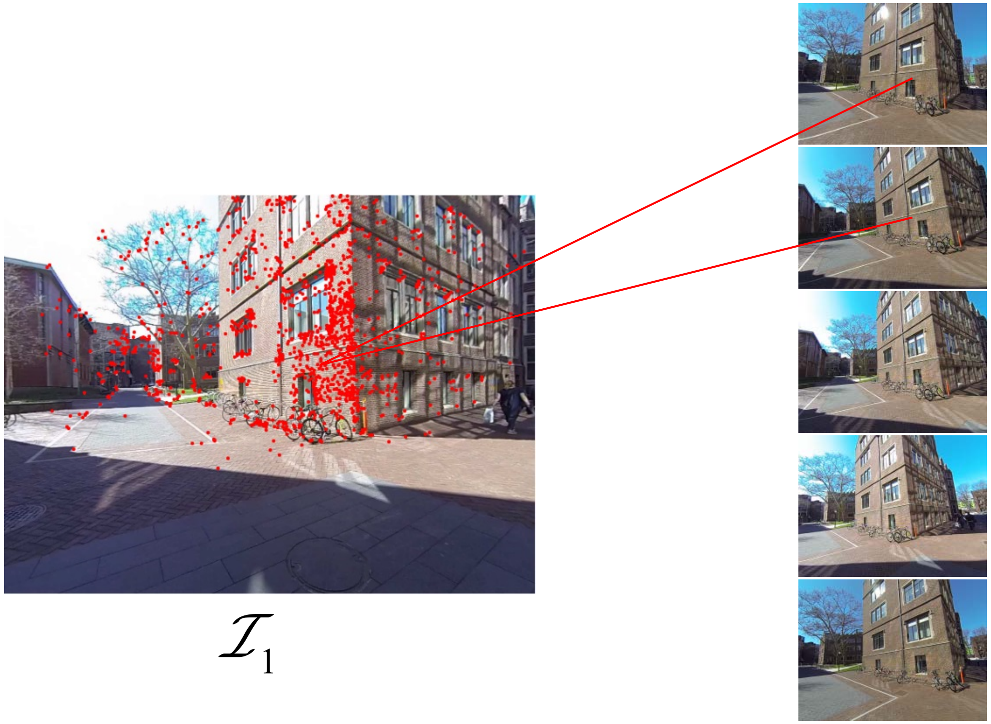
Knowns

Matching File Format

Number of feature points in image1

matching1.txt

```
nFeatures: 2002
3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000
3 229 212 202 634.560000 276.740000 2 520.960000 383.180000 3 672.890000 372.240000
```



$\mathcal{I}_1$

Knowns

Matching File Format

matching1.txt

```
nFeatures: 2002
3 137 128 105 454.740000 392.370000 2 308.570000 500.320000 4 447.580000 479.360000
3 229 212 202 634.560000 276.740000 2 520.960000 383.180000 3 672.890000 372.240000
3 90 81 59 456.250000 391.990000 2 308.570000 500.320000 4 447.580000 479.360000
3 90 81 59 456.250000 391.990000 2 308.570000 500.320000 4 447.580000 479.360000
3 183 158 140 804.630000 467.750000 2 717.520000 576.130000 4 978.600000 539.230000
2 241 225 215 797.950000 242.530000 2 693.880000 356.820000
2 58 40 37 754.440000 573.210000 2 684.400000 699.970000
3 229 212 202 634.560000 276.740000 2 520.960000 383.180000 3 672.890000 372.240000
2 142 135 129 960.800000 262.520000 2 836.090000 375.660000
4 172 203 238 466.340000 202.980000 2 321.130000 304.500000 3 465.910000 315.030000 4 437.970000 292.91
2 227 212 205 627.700000 169.440000 2 508.360000 278.100000
2 227 212 205 627.700000 169.440000 2 508.360000 278.100000
3 238 222 206 538.520000 455.990000 2 410.690000 572.820000 3 560.580000 567.890000
3 71 52 41 863.520000 414.110000 2 776.100000 520.160000 4 1047.040000 469.630000
```

Knowns

Matching Across Images

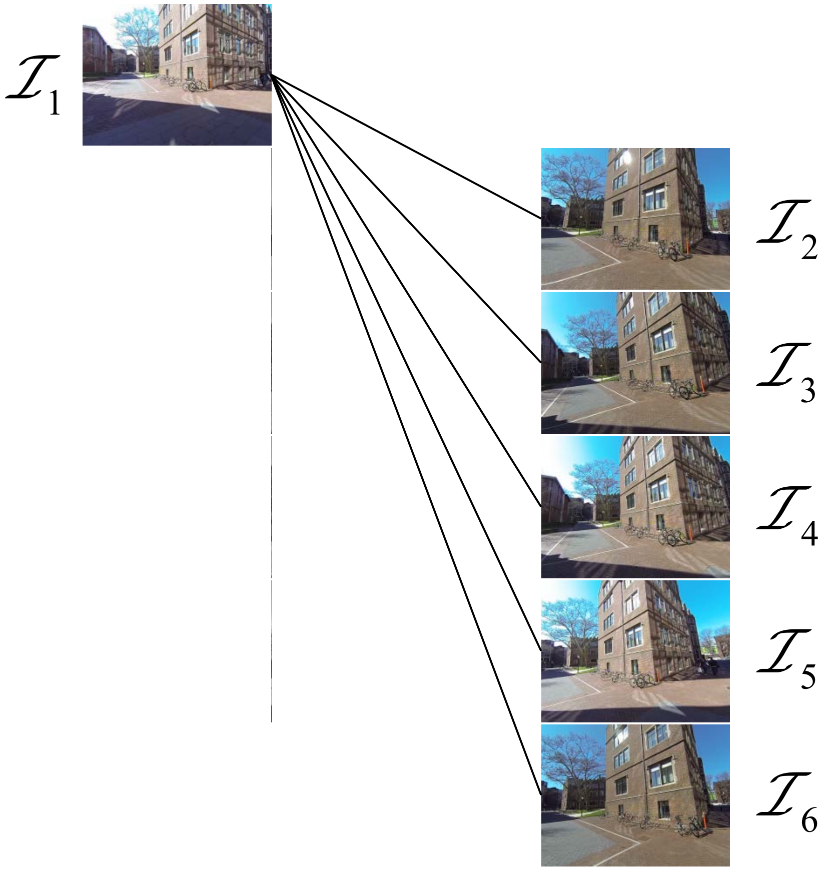
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{X} \\ 1\end{bmatrix}$$

$$= \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\begin{bmatrix}\mathbf{X} \\ 1\end{bmatrix}$$

$$= \mathbf{KR}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{X} \\ 1\end{bmatrix}$$

Matching data:

matching1.txt



Knowns

Matching Across Images

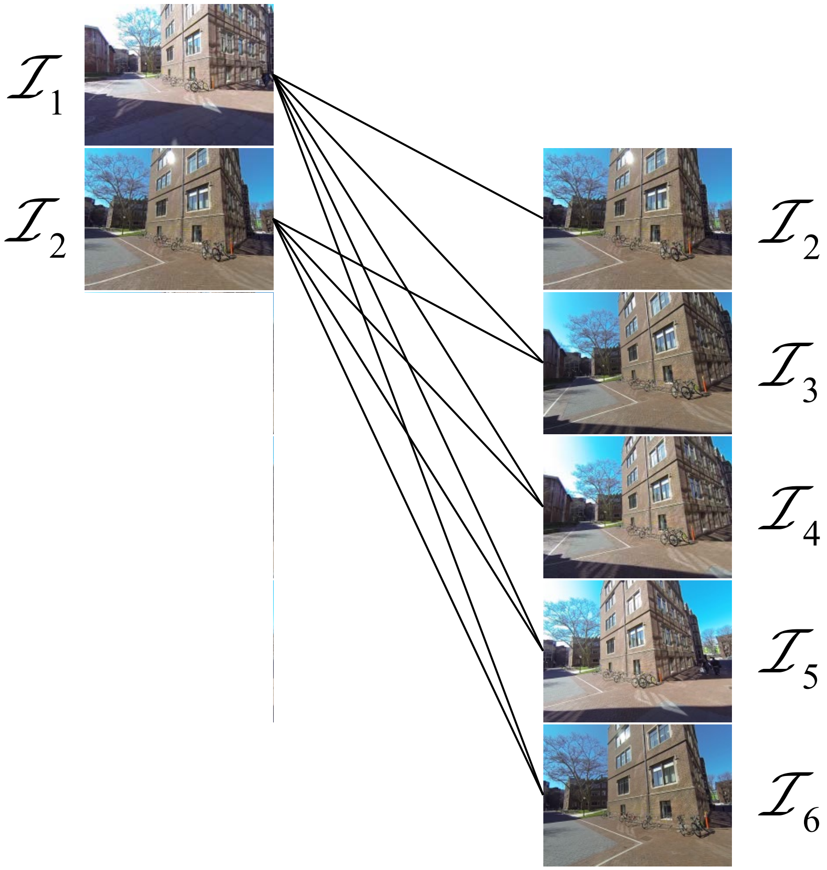
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{X}\\ 1\end{bmatrix}$$

$$= \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ 1\end{bmatrix}$$

$$= \mathbf{K}\mathbf{R}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{X}\\ 1\end{bmatrix}$$

Matching data:

matching1.txt, matching2.txt



Knowns

Matching Across Images

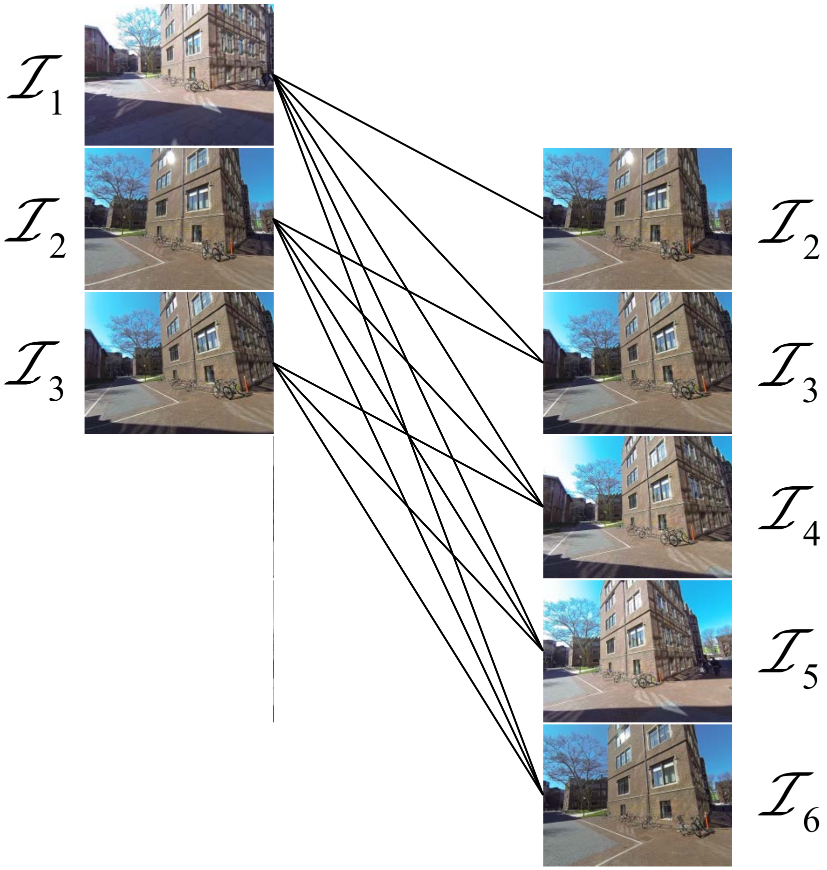
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

Matching data:

matching1.txt, matching2.txt, matching3.txt



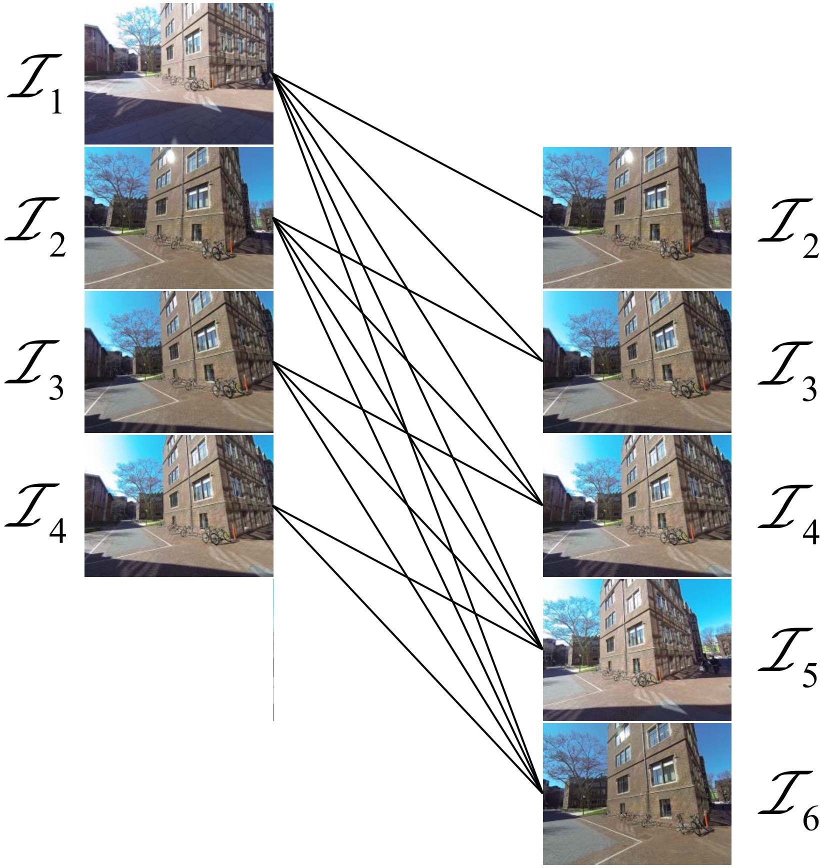
Knowns

Matching Across Images

$$
\begin{aligned}
\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \\
&= \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \\
&= \mathbf{K}\mathbf{R} \begin{bmatrix} \mathbf{I}_{3\times3} & -\mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}
\end{aligned}
$$

Matching data:

matching1.txt, matching2.txt, matching3.txt, matching4.txt



Knowns

Matching Across Images

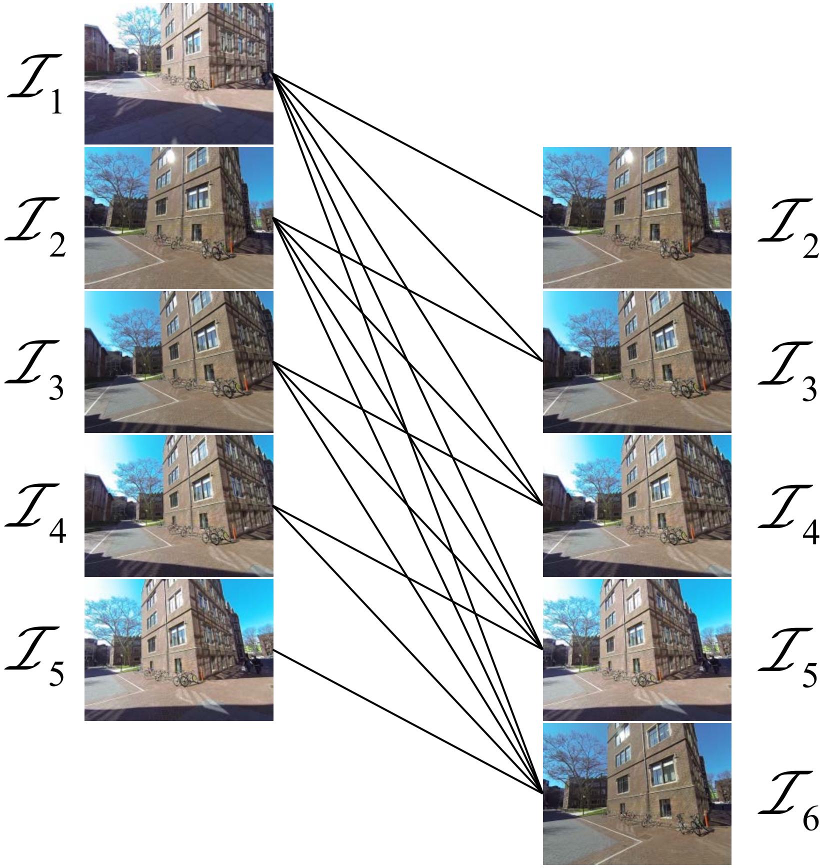
$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$

$$= \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\begin{bmatrix}\mathbf{X} \\ 1\end{bmatrix}$$

$$= \mathbf{KR}\begin{bmatrix}\mathbf{I}_{3\times3} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{X} \\ 1\end{bmatrix}$$

Matching data:

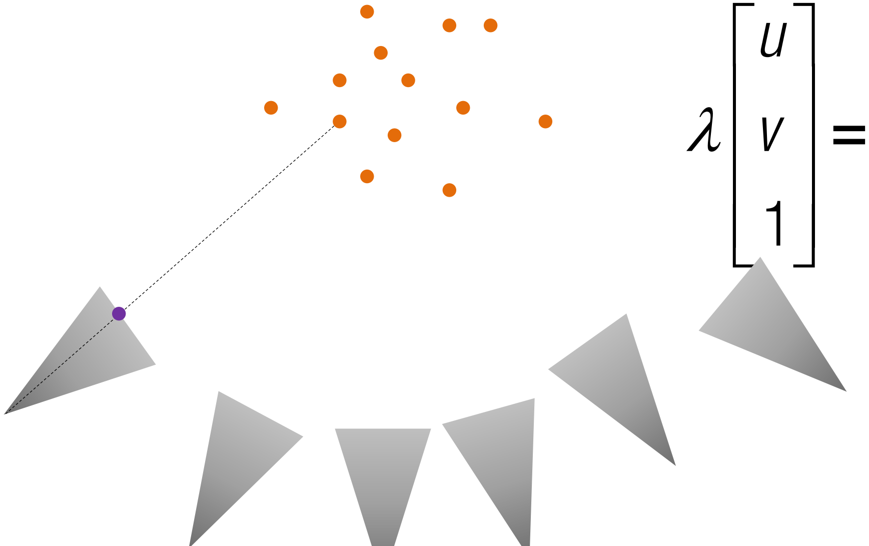
matching1.txt, matching2.txt, matching3.txt, matching4.txt, matching5.txt



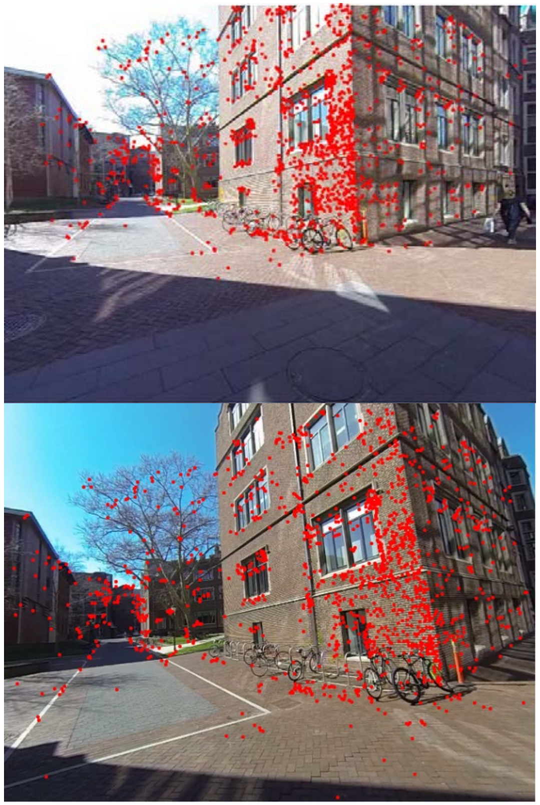
Additional Data

image0000001.bmp – image0000006.bmp

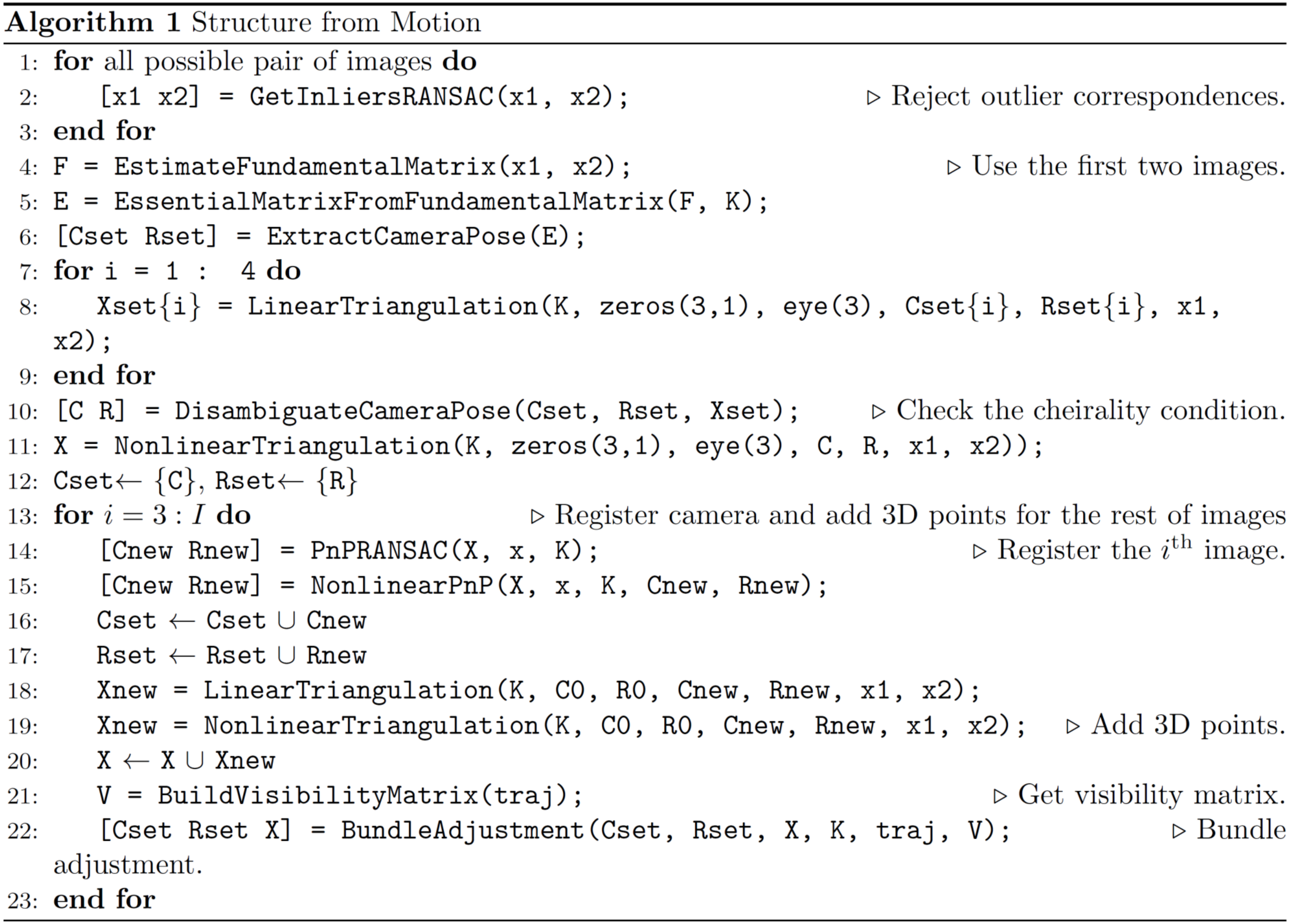
Images are useful to debug your result:
To project 3D points to an image to see alignment with the image.



$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$$



Structure from Motion Pipeline



Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);                ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);                 ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 : 4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1, x2);
end for
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);      ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset← {C}, Rset← {R}
for $i = 3 : I$ do                                     ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                  ▷ Register the $i^{th}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset ← Cset ∪ Cnew
    Rset ← Rset ∪ Rnew
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);   ▷ Add 3D points.
    X ← X ∪ Xnew
    V = BuildVisibilityMatrix(traj);                   ▷ Get visibility matrix.
    [Cset Rset X] = BundleAdjustment(Cset, Rset, X, K, traj, V);   ▷ Bundle adjustment.
end for
```

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);        ▷ Reject outlier correspondences.
end for
```

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);    ▷ Reject outlier correspondences.
end for
```

Matches filtered by a fundamental matrix

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);        ▷ Reject outlier correspondences.
end for
```

Algorithm 2 GetInliersRANSAC

```
$n \leftarrow 0$
for $i = 1 : M$ do
    Choose 8 correspondences, $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$, randomly
    F = EstimateFundamentalMatrix($\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$)
    $\mathcal{S} \leftarrow \emptyset$
    for $j = 1 : N$ do
        if $|\mathbf{x}_{2j}^{\mathsf{T}} \mathbf{F} \mathbf{x}_{1j}| < \epsilon$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
        end if
    end for
    if $n < |\mathcal{S}|$ then
        $n \leftarrow |\mathcal{S}|$
        $\mathcal{S}_{in} \leftarrow \mathcal{S}$
    end if
end for
```

$$\begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix}^{\mathsf{T}} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} = 0$$

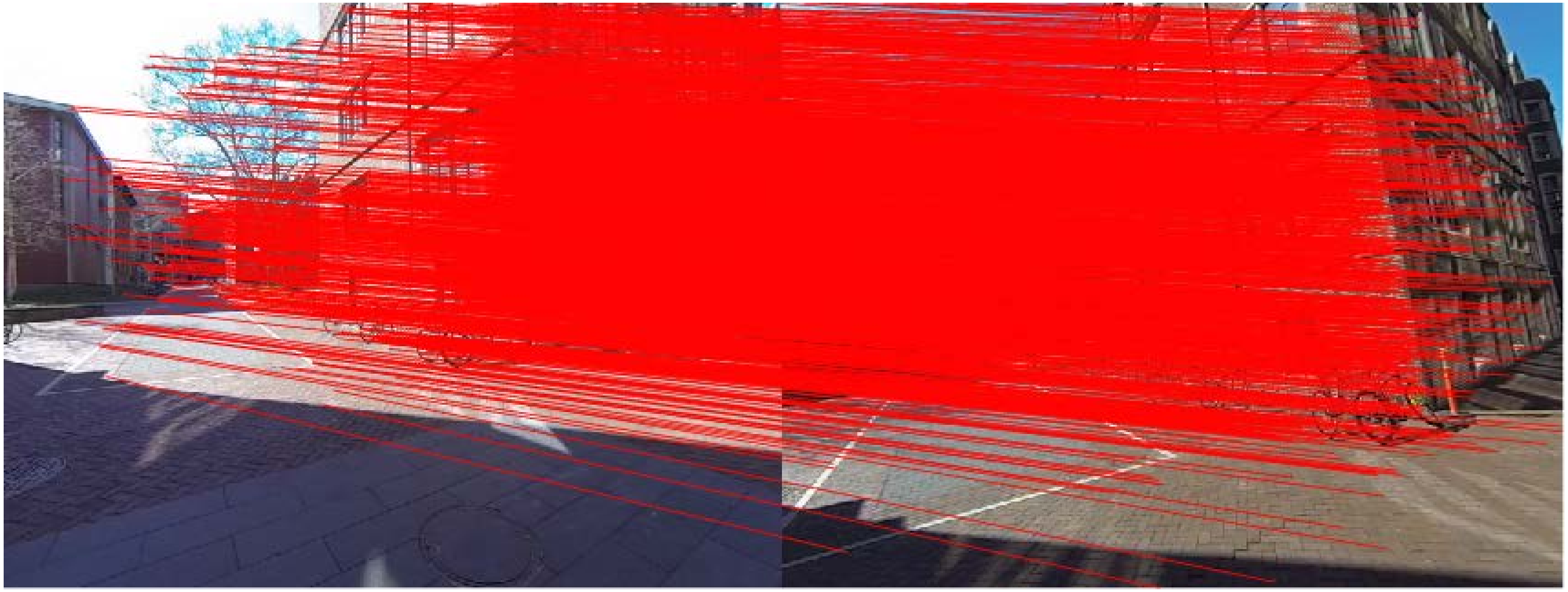
$$\begin{bmatrix} u_2u_1 & u_2v_1 & u_2 & v_2u_1 & v_2v_1 & v_2 & u_1 & v_1 & 1 \\ \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \mathbf{0}$$

$$\textbf{rank(F)} = 2 \quad \longrightarrow \quad \mathbf{F} = \mathbf{UDV}^{\mathsf{T}} \quad \text{where } \mathbf{D} = \begin{bmatrix} d_1 & & \\ & d_2 & \\ & & 0 \end{bmatrix}$$

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);          ▷ Reject outlier correspondences.
end for
```



Matches filtered by a fundamental matrix

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);        ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);         ▷ Use the first two images.
```

$\mathcal{I}_1$

$$\mathbf{x}_2^{\mathsf{T}}\mathbf{F}\mathbf{x}_1 = 0$$

$\mathcal{I}_2$

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);            ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);             ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
```

$\mathcal{I}_1$

$\mathcal{I}_2$

$$\mathbf{E} = \mathbf{K}^\mathsf{T}\mathbf{F}\mathbf{K}$$

An essential matrix must have (1,1,0) singular values.

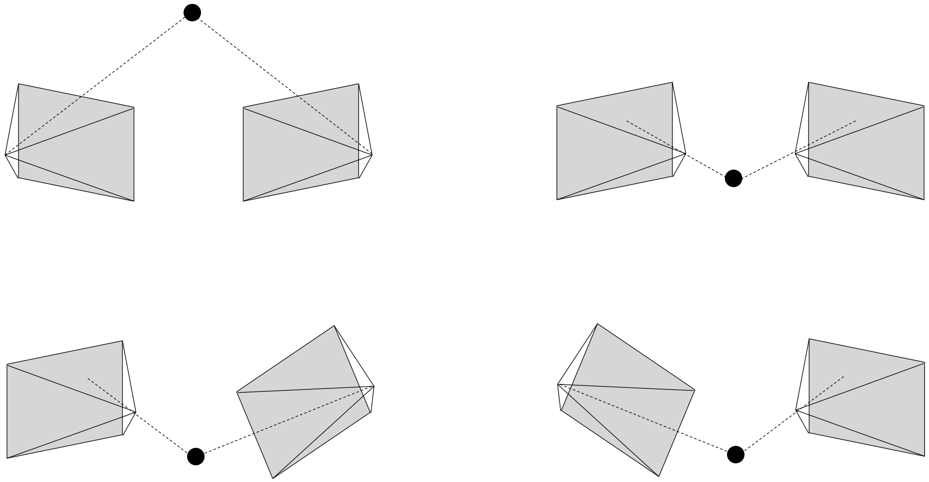
$$\mathbf{E} = \mathbf{U}\mathbf{D}\mathbf{V}^\mathsf{T} \quad \text{where} \quad \mathbf{D} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix}$$

Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);          ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);           ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
```

Given an essential matrix, there are 4 camera pose configurations.



$$P_2 = \left[UYV^{\top} \;\middle|\; u_3 \right]$$

$$\left[UY^{\top}V^{\top} \;\middle|\; u_3 \right]$$

$$\left[UYV^{\top} \;\middle|\; -u_3 \right]$$

$$\left[UY^{\top}V^{\top} \;\middle|\; -u_3 \right]$$

$$\text{where } \mathbf{Y} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

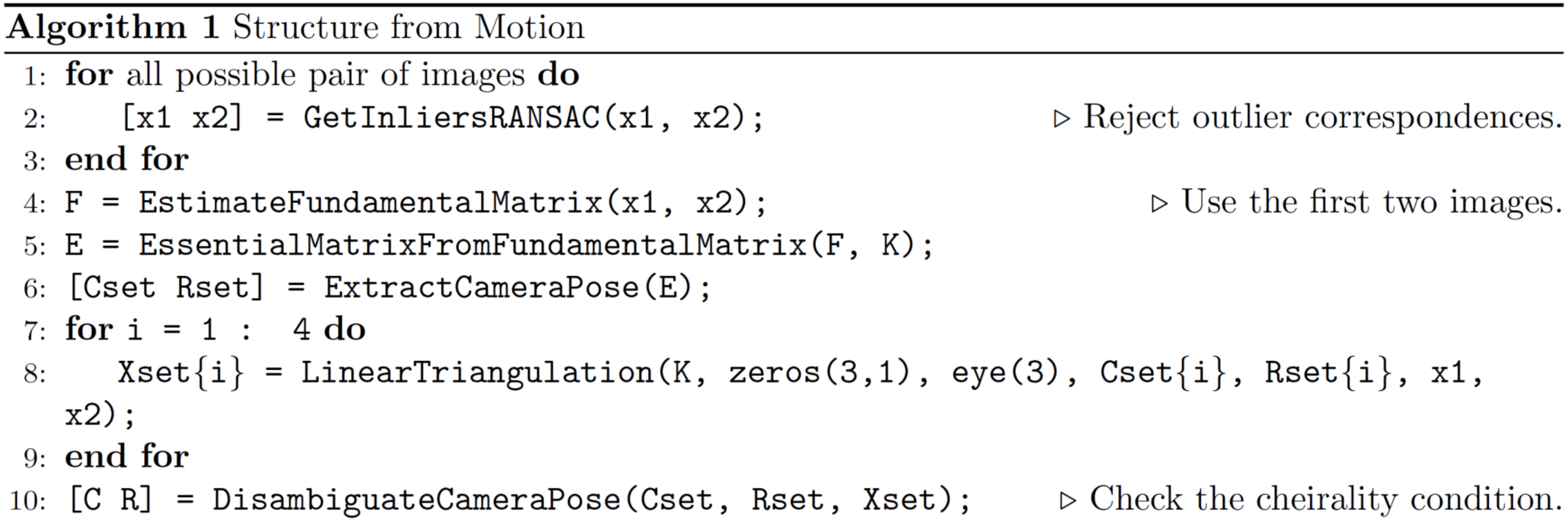
Structure from Motion Pipeline

Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);          ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);           ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 : 4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1, x2);
end for
```

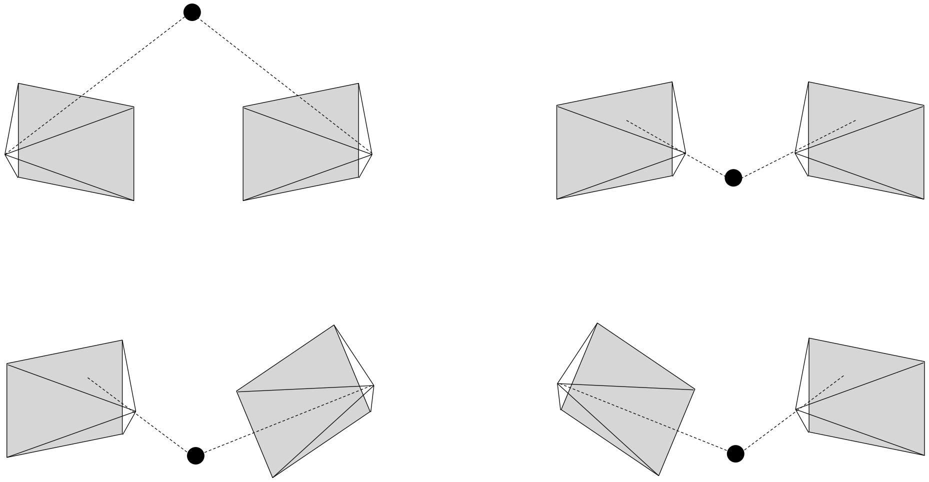
$$\begin{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ 1 \end{bmatrix}_\times \mathbf{P}_1 \\ \begin{bmatrix} \mathbf{x}_2 \\ 1 \end{bmatrix}_\times \mathbf{P}_2 \\ \vdots \\ \begin{bmatrix} \mathbf{x}_F \\ 1 \end{bmatrix}_\times \mathbf{P}_F \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} = \mathbf{0}$$

Structure from Motion Pipeline



Algorithm 1 Structure from Motion

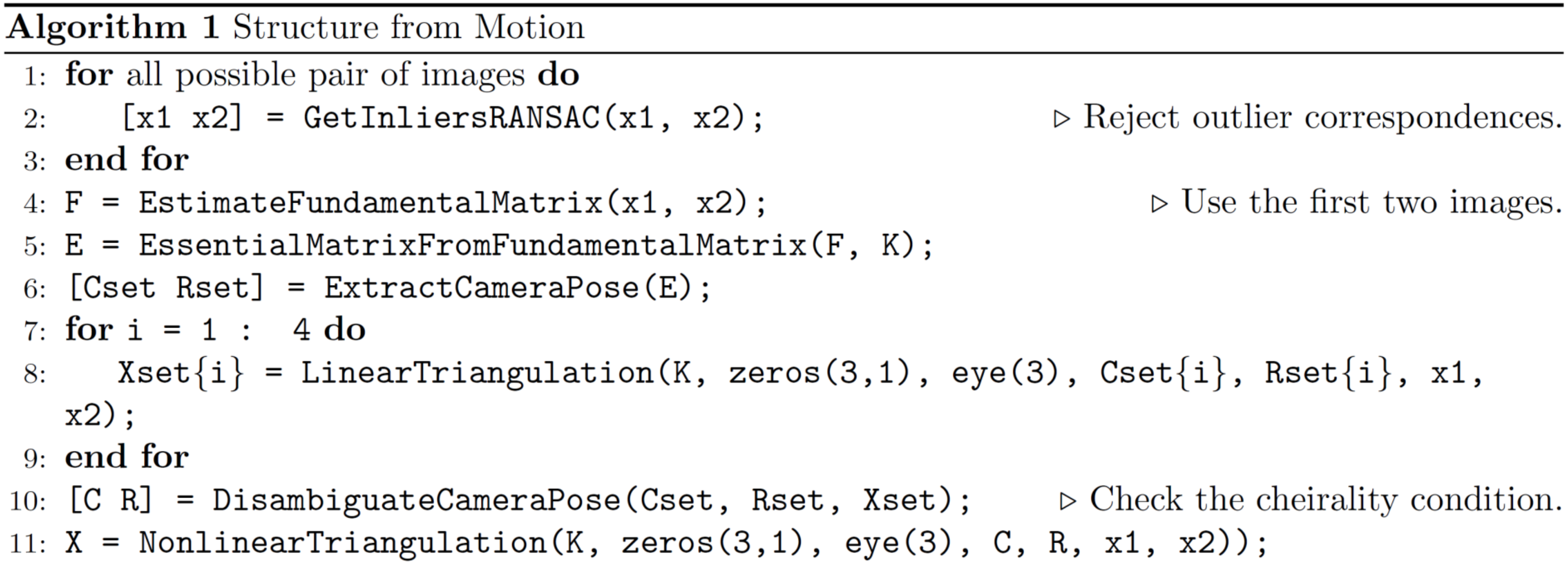
```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);          ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);            ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 :  4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1,
x2);
end for
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);   ▷ Check the cheirality condition.
```



$$\begin{bmatrix}0 & 0 & 1\end{bmatrix}\mathbf{X} \geq 0$$

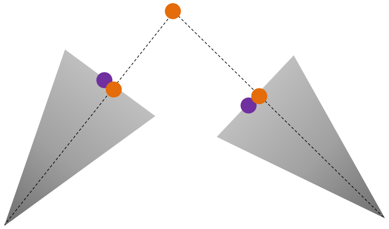
$$\mathbf{r}_3^{\mathsf{T}}\left(\mathbf{X}-\mathbf{C}\right) \geq 0$$

Structure from Motion Pipeline



Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);                         ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);                           ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 : 4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1,
x2);
end for
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);              ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
```



$$\min_{\mathbf{X}} \sum_{i=1}^{2} \left(u_i - \frac{\mathbf{P}_i^1 \tilde{\mathbf{X}}}{\mathbf{P}_i^3 \tilde{\mathbf{X}}} \right)^2 + \left(v_i - \frac{\mathbf{P}_i^2 \tilde{\mathbf{X}}}{\mathbf{P}_i^3 \tilde{\mathbf{X}}} \right)^2$$

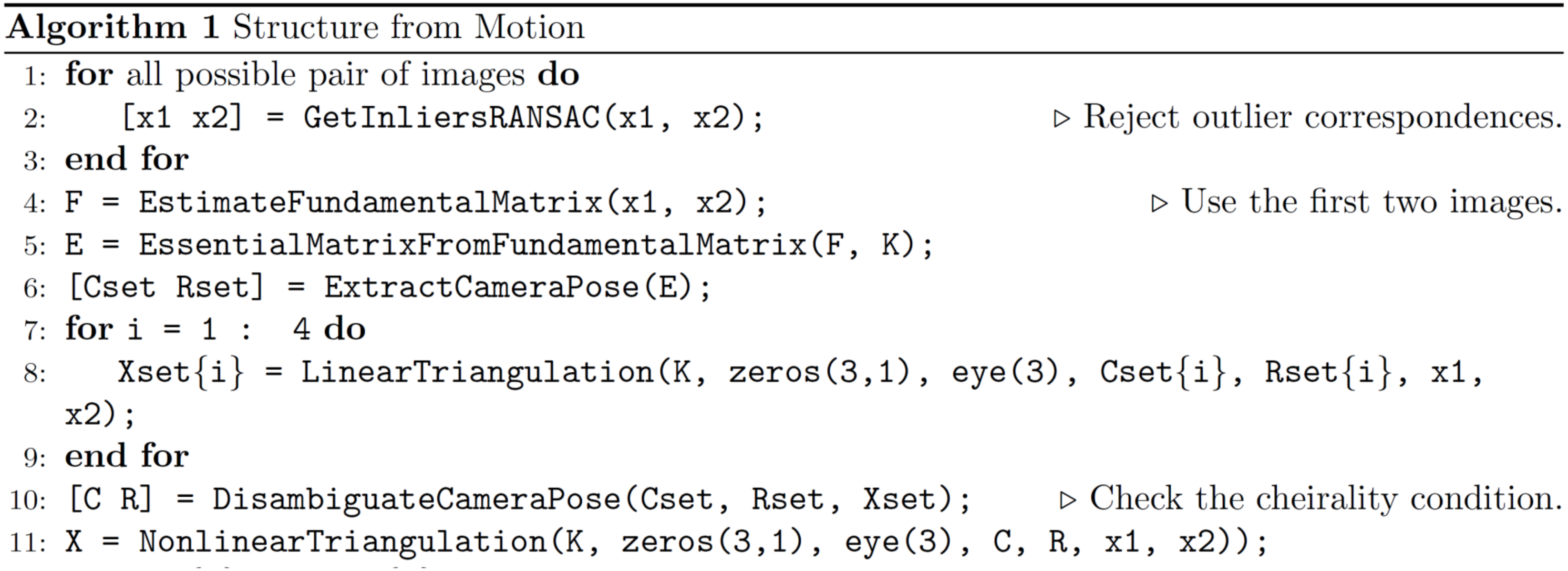
Reprojection error minimization

● (u, v)

● $(u_{\text{repro}}, v_{\text{repro}}) = \left(\frac{\mathbf{P}^1 \tilde{\mathbf{X}}}{\mathbf{P}^3 \tilde{\mathbf{X}}}, \frac{\mathbf{P}^2 \tilde{\mathbf{X}}}{\mathbf{P}^3 \tilde{\mathbf{X}}} \right)$

where $\mathbf{P} = \begin{bmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \mathbf{P}^3 \end{bmatrix}$ and $\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix}$

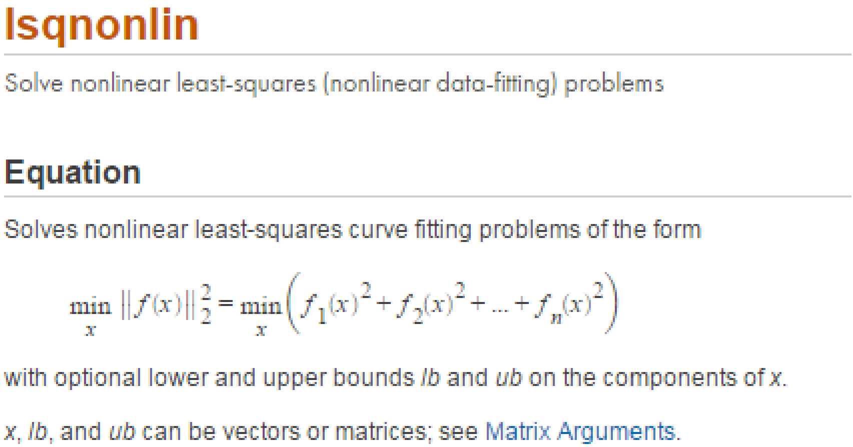
Structure from Motion Pipeline



Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);                         ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);                              ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 :  4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1, x2);
end for
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);         ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
```

Use MATLAB built-in function such as



lsqnonlin

Solve nonlinear least-squares (nonlinear data-fitting) problems

Equation

Solves nonlinear least-squares curve fitting problems of the form

$$\min_{x} \|f(x)\|_2^2 = \min_{x} \left(f_1(x)^2 + f_2(x)^2 + ... + f_n(x)^2 \right)$$

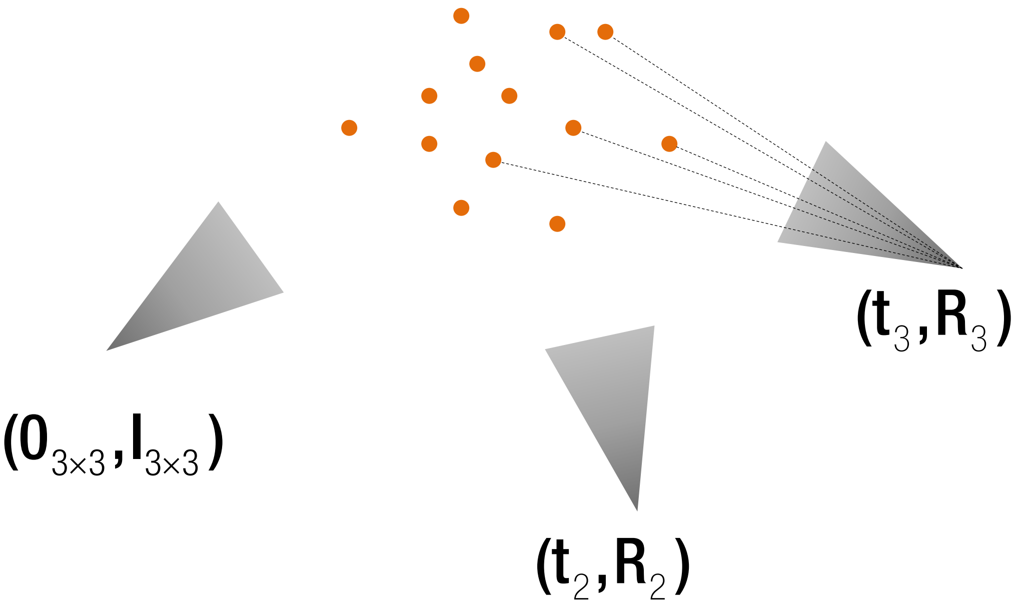
with optional lower and upper bounds *lb* and *ub* on the components of *x*.

x, *lb*, and *ub* can be vectors or matrices; see Matrix Arguments.

Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset$\leftarrow$ {C}, Rset$\leftarrow$ {R}
for $i = 3 : I$ do                      ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                          ▷ Register the $i^{th}$ image.
```

Given the reconstructed camera poses and points from the first two images, register a new image in the same coordinate system.



Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset ← {C}, Rset ← {R}
for $i = 3 : I$ do                  ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                       ▷ Register the $i^{\text{th}}$ image.
```

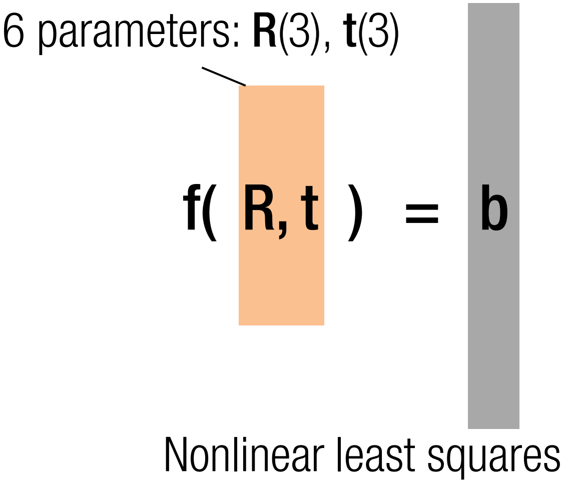
Algorithm 3 PnPRANSAC

```
$n \leftarrow 0$
for $i = 1 : M$ do
    Choose 6 correspondences, $\hat{\mathbf{X}}$ and $\hat{\mathbf{x}}$, randomly
    [C R] = LinearPnP($\hat{\mathbf{X}}$, $\hat{\mathbf{x}}$, K)
    $\mathcal{S} \leftarrow \emptyset$
    for $j = 1 : N$ do
        $e = \left(u - \frac{\mathbf{P}_1^\top \tilde{\mathbf{X}}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}}\right)^2 + \left(v - \frac{\mathbf{P}_2^\top \tilde{\mathbf{X}}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}}\right)^2$        ▷ Measure reprojection error.
        if $e < \epsilon_r$ then
            $\mathcal{S} \leftarrow \mathcal{S} \cup \{j\}$
        end if
    end for
    if $n < |\mathcal{S}|$ then
        $n \leftarrow |\mathcal{S}|$
        $\mathcal{S}_{in} \leftarrow \mathcal{S}$
    end if
end for
```

$$\begin{bmatrix}\mathbf{x}_1\\1\end{bmatrix}_\times \mathbf{P}\begin{bmatrix}\mathbf{X}_1\\1\end{bmatrix} = \mathbf{0} \longrightarrow \begin{bmatrix}\mathbf{0}_{1\times 4} & -\tilde{\mathbf{X}}_1^\mathsf{T} & v_1\tilde{\mathbf{X}}_1^\mathsf{T}\\ \tilde{\mathbf{X}}_1^\mathsf{T} & \mathbf{0}_{1\times 4} & -u_1\tilde{\mathbf{X}}_1^\mathsf{T}\\ -v_1\tilde{\mathbf{X}}_1^\mathsf{T} & u_1\tilde{\mathbf{X}}_1^\mathsf{T} & \mathbf{0}_{1\times 4}\end{bmatrix}\begin{bmatrix}\mathbf{P}_1^\mathsf{T}\\ \mathbf{P}_2^\mathsf{T}\\ \mathbf{P}_3^\mathsf{T}\end{bmatrix} = \mathbf{0}$$

Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset ← {C}, Rset ← {R}
for $i = 3 : I$ do                              ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                              ▷ Register the $i^{\text{th}}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
```



6 parameters: **R**(3), **t**(3)

f(R, t) = b

Nonlinear least squares

$$\underset{\mathbf{C},\mathbf{R}}{\text{minimize}} \sum_{j=1}^{J} \left(\left(u_j - \frac{\mathbf{P}_1^\top \tilde{\mathbf{X}}_\mathbf{j}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}_\mathbf{j}} \right)^2 + \left(v_j - \frac{\mathbf{P}_2^\top \tilde{\mathbf{X}}_\mathbf{j}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}_\mathbf{j}} \right)^2 \right)$$

$$\underset{\mathbf{C},\mathbf{q}}{\text{minimize}} \sum_{j=1}^{J} \left(\left(u_j - \frac{\mathbf{P}_1^\top \tilde{\mathbf{X}}_\mathbf{j}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}_\mathbf{j}} \right)^2 + \left(v_j - \frac{\mathbf{P}_2^\top \tilde{\mathbf{X}}_\mathbf{j}}{\mathbf{P}_3^\top \tilde{\mathbf{X}}_\mathbf{j}} \right)^2 \right)$$

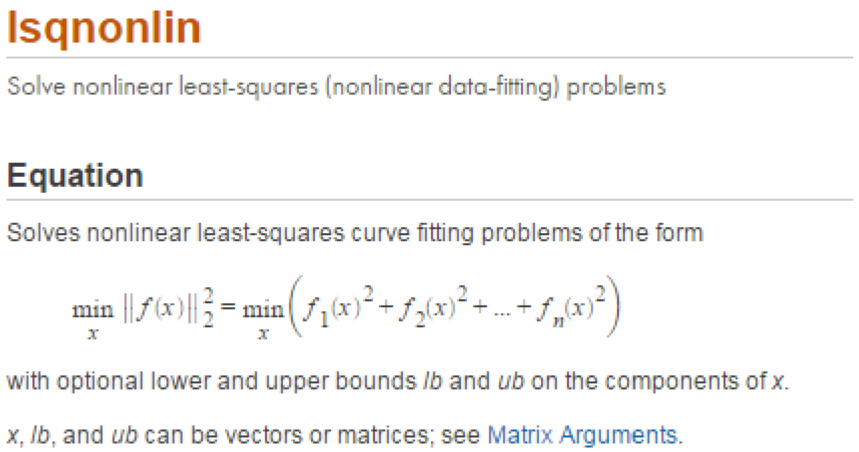
Quaternion parameterization to enforce SO(3)

$$\mathbf{R} = \begin{bmatrix} 1-2q_zq_z-2q_yq_y & -2q_zq_w+2q_yq_x & 2q_yq_w+2q_zq_x \\ 2q_xq_y+2q_wq_z & 1-2^*q_zq_z-2q_xq_x & 2q_zq_y-2q_xq_w \\ 2q_xq_z-2q_wq_y & 2q_yq_z+2q_wq_x & 1-2q_yq_y-2q_xq_x \end{bmatrix}$$

Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset ← {C}, Rset ← {R}
for $i = 3 : I$ do                         ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                          ▷ Register the $i^{\text{th}}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
```

Use MATLAB built-in function such as



lsqnonlin

Solve nonlinear least-squares (nonlinear data-fitting) problems

Equation

Solves nonlinear least-squares curve fitting problems of the form

$$\min_x \|f(x)\|_2^2 = \min_x \left(f_1(x)^2 + f_2(x)^2 + \ldots + f_n(x)^2\right)$$

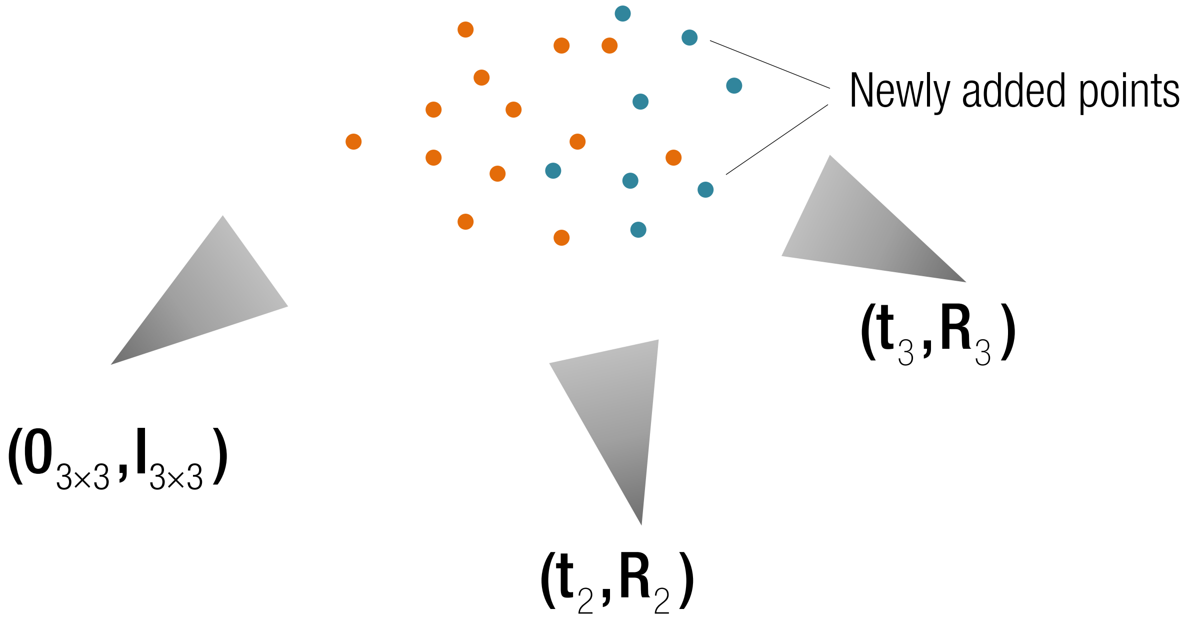
with optional lower and upper bounds *lb* and *ub* on the components of *x*.

x, *lb*, and *ub* can be vectors or matrices; see Matrix Arguments.

Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset← {C}, Rset← {R}
for $i = 3 : I$ do                         ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                           ▷ Register the $i^{\mathrm{th}}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset ← Cset ∪ Cnew
    Rset ← Rset ∪ Rnew
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);   ▷ Add 3D points.
    X ← X ∪ Xnew
```

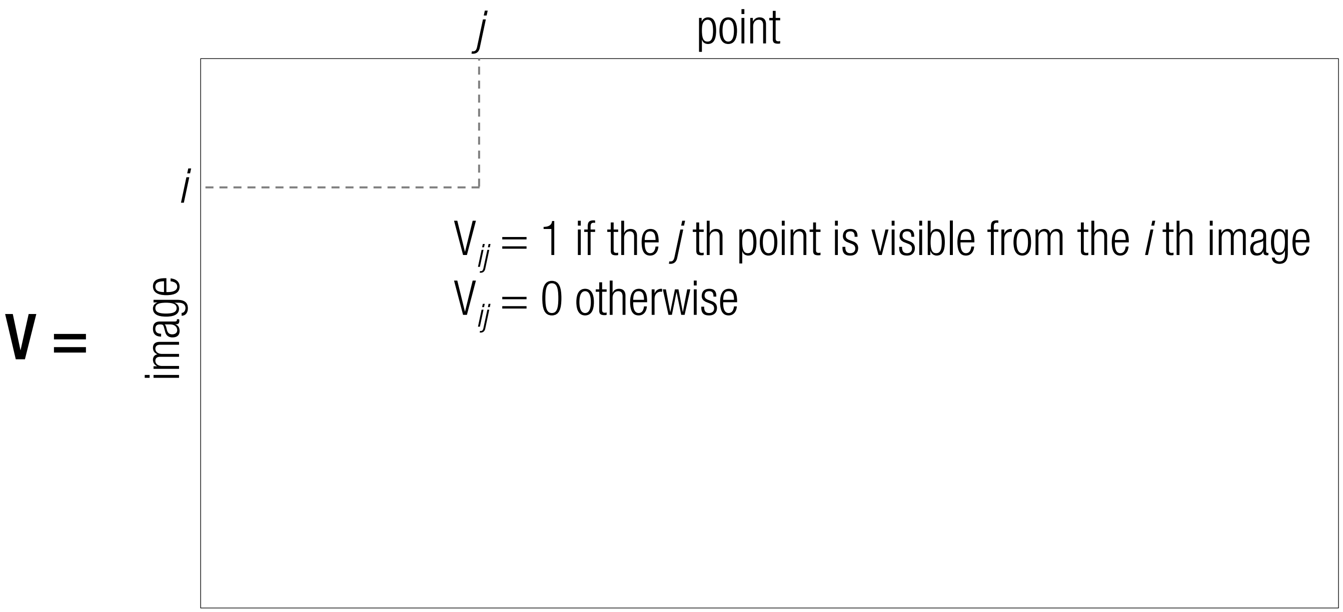
Triangulate new 3D points that were not reconstructed yet.



Structure from Motion Pipeline

```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset← {C}, Rset← {R}
for $i = 3 : I$ do                     ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                        ▷ Register the $i^{\text{th}}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset ← Cset ∪ Cnew
    Rset ← Rset ∪ Rnew
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);   ▷ Add 3D points.
    X ← X ∪ Xnew
    V = BuildVisibilityMatrix(traj);                         ▷ Get visibility matrix.
```

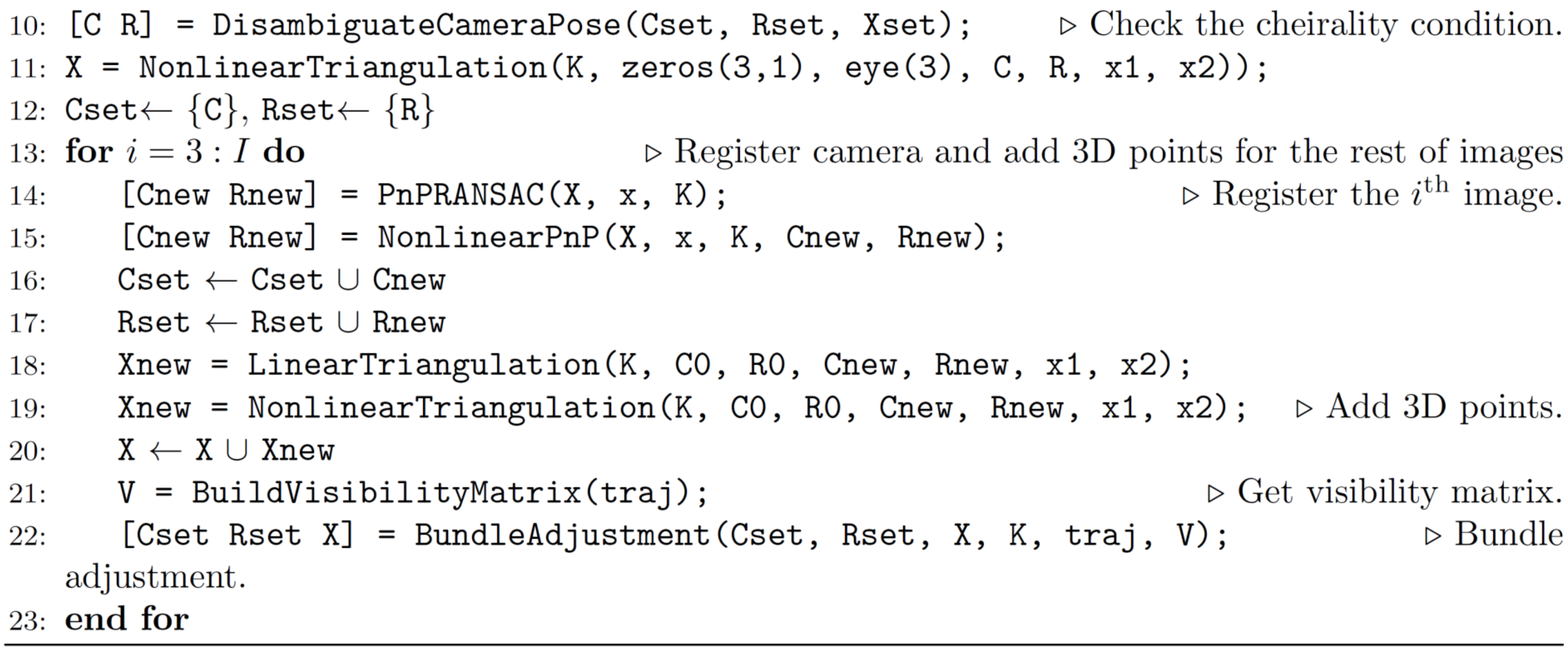
Get a visibility matrix to indicate that a point is visible from images.



$V_{ij} = 1$ if the *j* th point is visible from the *i* th image

$V_{ij} = 0$ otherwise

Structure from Motion Pipeline

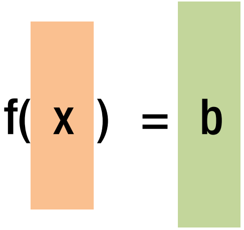


```
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);        ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset $\leftarrow$ {C}, Rset $\leftarrow$ {R}
for $i = 3 : I$ do                ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                     ▷ Register the $i^{\text{th}}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset $\leftarrow$ Cset $\cup$ Cnew
    Rset $\leftarrow$ Rset $\cup$ Rnew
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);   ▷ Add 3D points.
    X $\leftarrow$ X $\cup$ Xnew
    V = BuildVisibilityMatrix(traj);                        ▷ Get visibility matrix.
    [Cset Rset X] = BundleAdjustment(Cset, Rset, X, K, traj, V);   ▷ Bundle adjustment.
end for
```

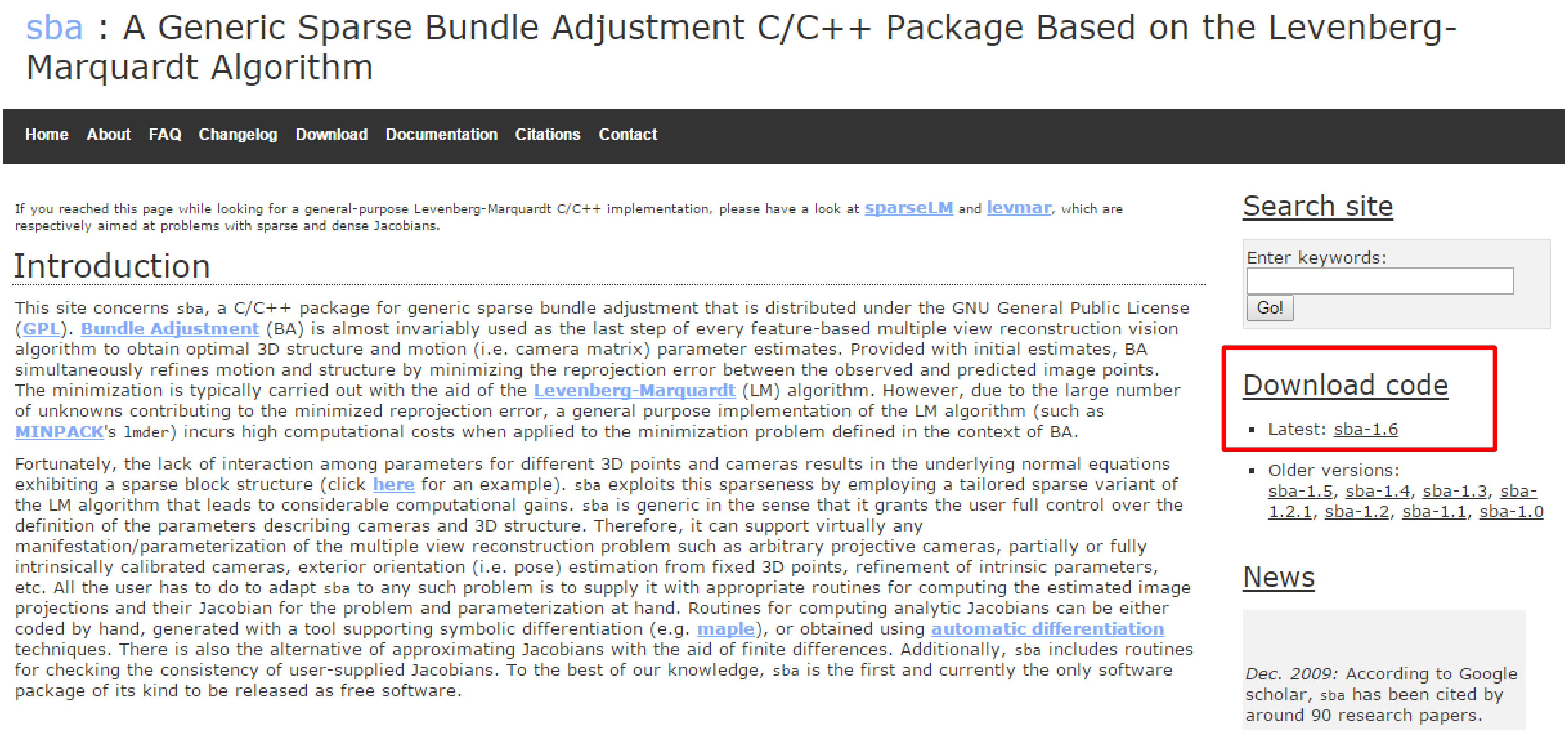
Refine camera poses and 3D points by minimizing reprojection error:

$$\underset{\{\mathbf{C}_i,\mathbf{q}_i\}_{i=1}^{I},\{\mathbf{X}\}_{j=1}^{J}}{\text{minimize}} \quad \sum_{i=1}^{I}\sum_{j=1}^{J}\mathbf{V}_{ij}\left(\left(u_{ij}-\frac{\mathbf{P}_{i1}^{\mathsf{T}}\widetilde{\mathbf{X}}_{\mathbf{j}}}{\mathbf{P}_{i3}^{\mathsf{T}}\widetilde{\mathbf{X}}_{\mathbf{j}}}\right)^2+\left(v_{ij}-\frac{\mathbf{P}_{i2}^{\mathsf{T}}\widetilde{\mathbf{X}}_{\mathbf{j}}}{\mathbf{P}_{i3}^{\mathsf{T}}\widetilde{\mathbf{X}}_{\mathbf{j}}}\right)^2\right)$$

Large scale optimization (6 images and 1000 3D points)
Number of unknowns: 6x7+1000x3 = 3042
Number of equations: 2x(nonzeros in **V**)



Sparse Bundle Adjustment Package



sba : A Generic Sparse Bundle Adjustment C/C++ Package Based on the Levenberg-Marquardt Algorithm

If you reached this page while looking for a general-purpose Levenberg-Marquardt C/C++ implementation, please have a look at sparseLM and levmar, which are respectively aimed at problems with sparse and dense Jacobians.

Introduction

This site concerns sba, a C/C++ package for generic sparse bundle adjustment that is distributed under the GNU General Public License (GPL). Bundle Adjustment (BA) is almost invariably used as the last step of every feature-based multiple view reconstruction vision algorithm to obtain optimal 3D structure and motion (i.e. camera matrix) parameter estimates. Provided with initial estimates, BA simultaneously refines motion and structure by minimizing the reprojection error between the observed and predicted image points. The minimization is typically carried out with the aid of the Levenberg-Marquardt (LM) algorithm. However, due to the large number of unknowns contributing to the minimized reprojection error, a general purpose implementation of the LM algorithm (such as MINPACK's lmder) incurs high computational costs when applied to the minimization problem defined in the context of BA.

Fortunately, the lack of interaction among parameters for different 3D points and cameras results in the underlying normal equations exhibiting a sparse block structure (click here for an example). sba exploits this sparseness by employing a tailored sparse variant of the LM algorithm that leads to considerable computational gains. sba is generic in the sense that it grants the user full control over the definition of the parameters describing cameras and 3D structure. Therefore, it can support virtually any manifestation/parameterization of the multiple view reconstruction problem such as arbitrary projective cameras, partially or fully intrinsically calibrated cameras, exterior orientation (i.e. pose) estimation from fixed 3D points, refinement of intrinsic parameters, etc. All the user has to do to adapt sba to any such problem is to supply it with appropriate routines for computing the estimated image projections and their Jacobian for the problem and parameterization at hand. Routines for computing analytic Jacobians can be either coded by hand, generated with a tool supporting symbolic differentiation (e.g. maple), or obtained using automatic differentiation techniques. There is also the alternative of approximating Jacobians with the aid of finite differences. Additionally, sba includes routines for checking the consistency of user-supplied Jacobians. To the best of our knowledge, sba is the first and currently the only software package of its kind to be released as free software.

Search site

Enter keywords:

Download code

- Latest: sba-1.6
- Older versions: sba-1.5, sba-1.4, sba-1.3, sba-1.2.1, sba-1.2, sba-1.1, sba-1.0

News

Dec. 2009: According to Google scholar, sba has been cited by around 90 research papers.

http://users.ics.forth.gr/~lourakis/sba/

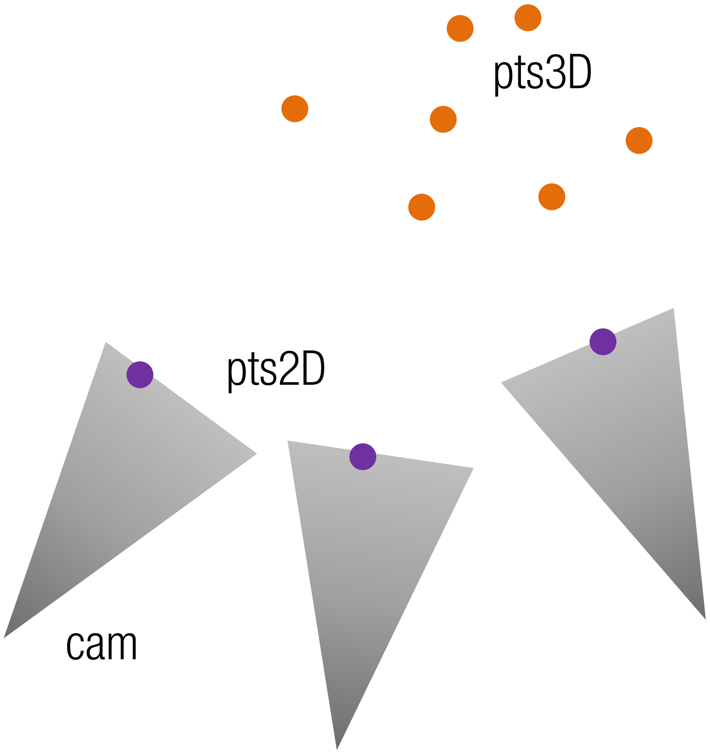
SBA Wrapper

```
function [cP, X] = sba_wrapper(measurements, cP, X, K)

r0 = [];    cams = [];  pts2D = [];
spmask=sparse([], [], [], 1, nFrames);

% Concatenate all camera poses into cams vector
cam
% Concatenate all 2D measurements into pts2D
pts2D
% Set 3D points
pts3D = X;

% Calibration parameters
cal = [K(1,1) K(1,2), K(1,3), K(2,2), K(2,3)];

opts=[1E-1, 0, 0, 1E-5, 0.0];
p0=[Mat2Vec(cams)' Mat2Vec(pts3D)'];

[ret, p, info]=sba(nFeatures, 0, nFrames, 1, spmask, p0, 7, 3, pts2D, 2, 'projection', 1e+2, 1, opts, 'motstr', r0, cal);

% Retrieve paramters
```

SBA Wrapper

```
function [cP, X] = sba_wrapper(measurements, cP, X, K)

r0 = [];    cams = [];  pts2D = [];
spmask=sparse([], [], [], 1, nFrames);

% Concatenate all camera poses into cams vector
cam
% Concatenate all 2D measurements into pts2D
pts2D
% Set 3D points
pts3D = X;

% Calibration parameters
cal = [K(1,1) K(1,2), K(1,3), K(2,2), K(2,3)];

opts=[1E-1, 0, 0, 1E-5, 0.0];
p0=[Mat2Vec(cams)' Mat2Vec(pts3D)'];

[ret, p, info]=sba(nFeatures, 0, nFrames, 1, spmask, p0, 7, 3, pts2D, 2, 'projection', 1e+2, 1, opts, 'motstr', r0, cal);

% Retrieve paramters
```

Function to fill

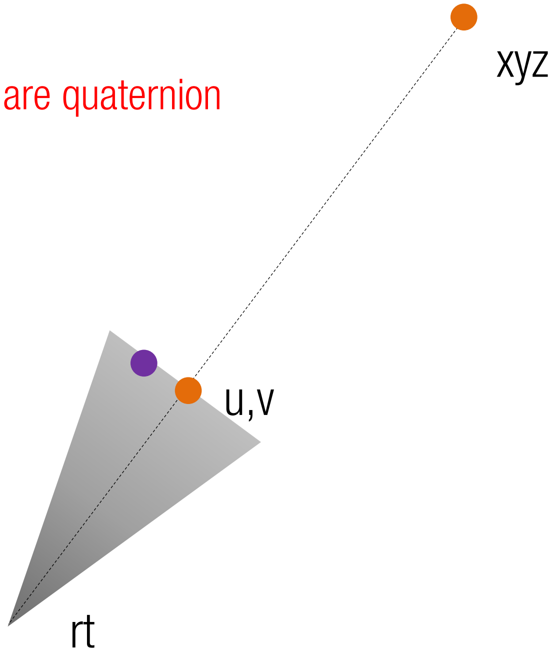


Reprojection

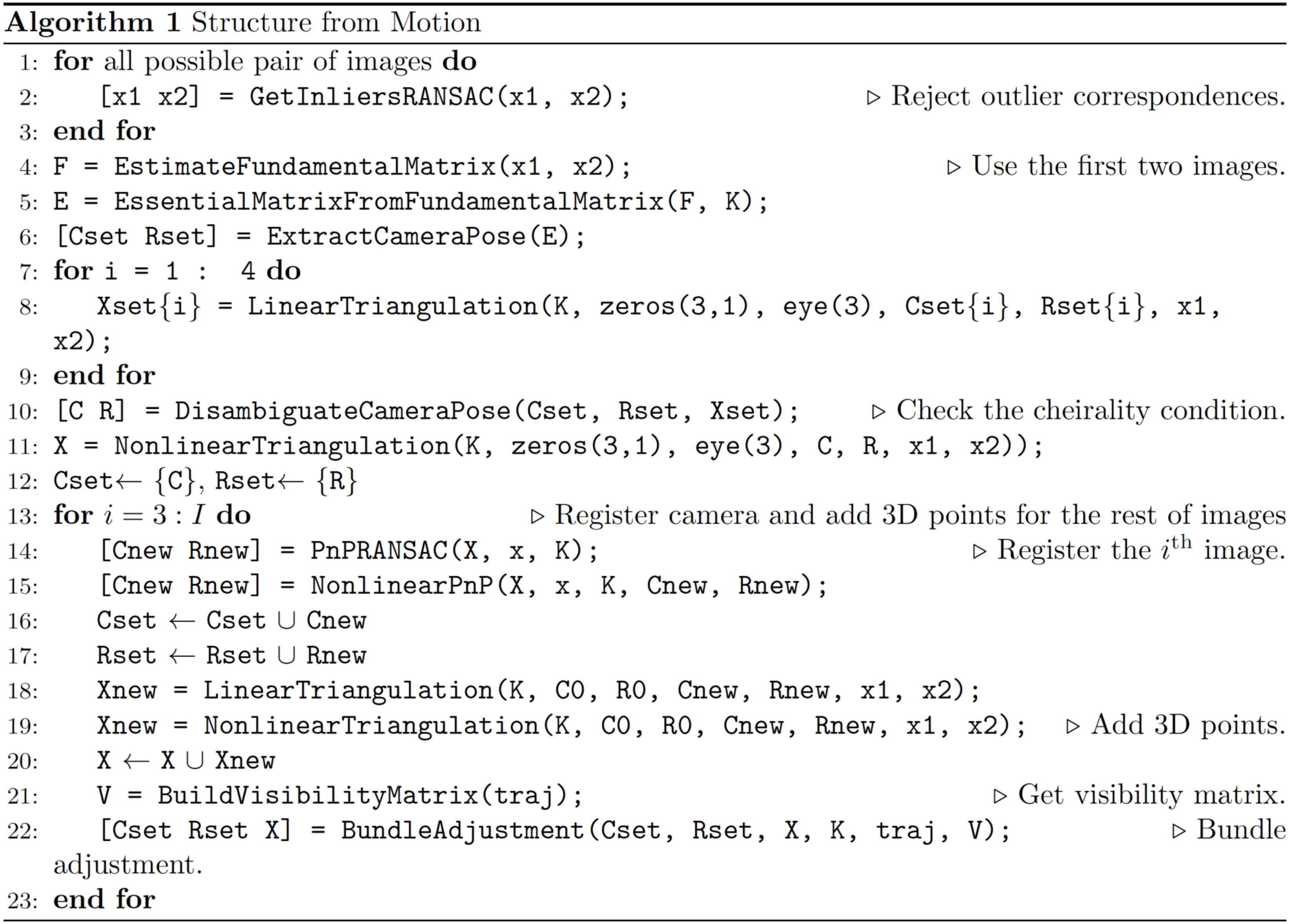
```
function m = projection(j, i, rt, xyz, r0, a)

K = [a(1) a(2) a(3); 0 a(4) a(5); 0 0 1];
%% Code to fill
% Build P matrix from rt
% rt is 7 dimensional vector-the first three are camera center and the rest are quaternion

%% Get xyzX = xyz';

%% Code to fill
% Project the 3D point to the camera to produce (u,v)
% Return reprojection

m(1) = u;
m(2) = v;
```

Reprojection

```
function m = projection(j, i, rt, xyz, r0, a)

K = [a(1) a(2) a(3); 0 a(4) a(5); 0 0 1];
%% Code to fill
% Build P matrix from rt
% rt is 7 dimensional vector-the first three are camera center and the rest are quaternion

%% Get xyzX = xyz';

%% Code to fill
% Project the 3D point to the camera to produce (u,v)
% Return reprojection

m(1) = u;
m(2) = v;
```



Structure from Motion Pipeline



Algorithm 1 Structure from Motion

```
for all possible pair of images do
    [x1 x2] = GetInliersRANSAC(x1, x2);                          ▷ Reject outlier correspondences.
end for
F = EstimateFundamentalMatrix(x1, x2);                            ▷ Use the first two images.
E = EssentialMatrixFromFundamentalMatrix(F, K);
[Cset Rset] = ExtractCameraPose(E);
for i = 1 :  4 do
    Xset{i} = LinearTriangulation(K, zeros(3,1), eye(3), Cset{i}, Rset{i}, x1,
x2);
end for
[C R] = DisambiguateCameraPose(Cset, Rset, Xset);                 ▷ Check the cheirality condition.
X = NonlinearTriangulation(K, zeros(3,1), eye(3), C, R, x1, x2));
Cset← {C}, Rset← {R}
for $i = 3 : I$ do                    ▷ Register camera and add 3D points for the rest of images
    [Cnew Rnew] = PnPRANSAC(X, x, K);                             ▷ Register the $i^{th}$ image.
    [Cnew Rnew] = NonlinearPnP(X, x, K, Cnew, Rnew);
    Cset ← Cset ∪ Cnew
    Rset ← Rset ∪ Rnew
    Xnew = LinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);
    Xnew = NonlinearTriangulation(K, C0, R0, Cnew, Rnew, x1, x2);   ▷ Add 3D points.
    X ← X ∪ Xnew
    V = BuildVisibilityMatrix(traj);                              ▷ Get visibility matrix.
    [Cset Rset X] = BundleAdjustment(Cset, Rset, X, K, traj, V);      ▷ Bundle
adjustment.
end for
```

Tips

- Start early.
- Play with VisualSfM or Sparse Bundle Adjustment package.
- Debug by visualizing points and images in 3D.
- Debug by projecting 3D points to an image.
- Change the initial pair of images if the first and second images do not work for you.
- Change the order of camera registration if the sequential order of images does not work for you.

Desired Result