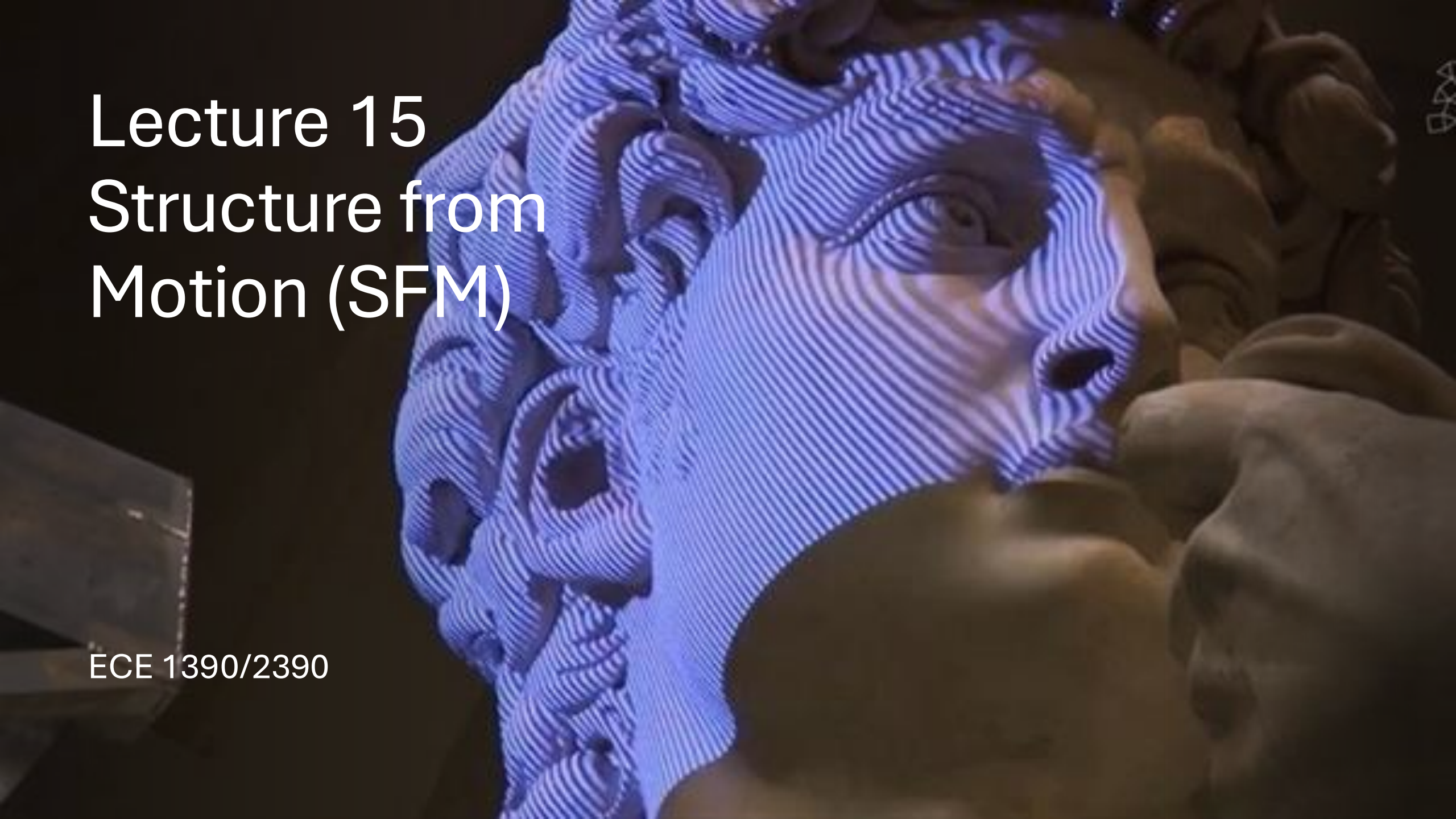
Lecture 15
Structure from Motion (SFM)



ECE 1390/2390

Requirements:

Everyone

- Written in Python using OpenCV
- Code compliant with MIT use license
- Documented and maintained on GitHub
- Include demonstration/example code

ECE 1390 Students

- Load/Save and process still images

- include an image enhancement method
- Include an image filtering method
- Include an edge detection method
- Demonstrate segmentation
- Include some object recognition

- Incorporate 1 additional methods from class code

ECE 2390 Students

- Load/Save and process still images
- Process video feed
- include an image enhancement method
- Include an image filtering method
- Include an edge detection method
- Demonstrate segmentation
- Include some object recognition
- Include some object tracking
- Incorporate 3 additional methods from class code

Syllabus

Week	Date	Topic	In class work	Due date
8	10/14/2024	FALL Break [No Class]		
	10/16/2024	Camera Calibration	Python problems (HW5)	Due 10/23
9	10/21/2024	Depth estimation	Group Project work	
	10/23/2024	3D Reconstruction and Pose estimation	Python problems (HW6)	Due 10/30
10	10/28/2024	Haar Cascade Classifiers	Python problems (HW7)	Due 11/4
	10/30/2024	HOG and Custom Detectors	Project updates	
11	11/4/2024	Object Tracking	Project updates	
	11/6/2024	OCR Text Detection	Python problems (HW8)	Due 11/13
12	11/11/2024	MediaPipe/SLAM	Group Project work	
	11/13/2024	OpenCV DNN	Python problems (HW9)	Due 11/20
13	11/18/2024	Super Resolution	Python problems (HW10)	Due 12/4
	11/20/2024	Image compression	Python problems	
14	11/25/2024	Thanksgiving recess		
	11/27/2024	Thanksgiving recess		
15	12/2/2024	Group Project work	Group Project work	
	12/4/2024	Group Project work	Group Project work	Final project commit
16	12/9/2024	Final Projects	Project presentations	Final group ratings

Project Updates (10/30 & 11/4)

10/30

- BokehSwap
- DC not DC
- AutonomousDriving
- RepVision
- FaceEmojiSwap

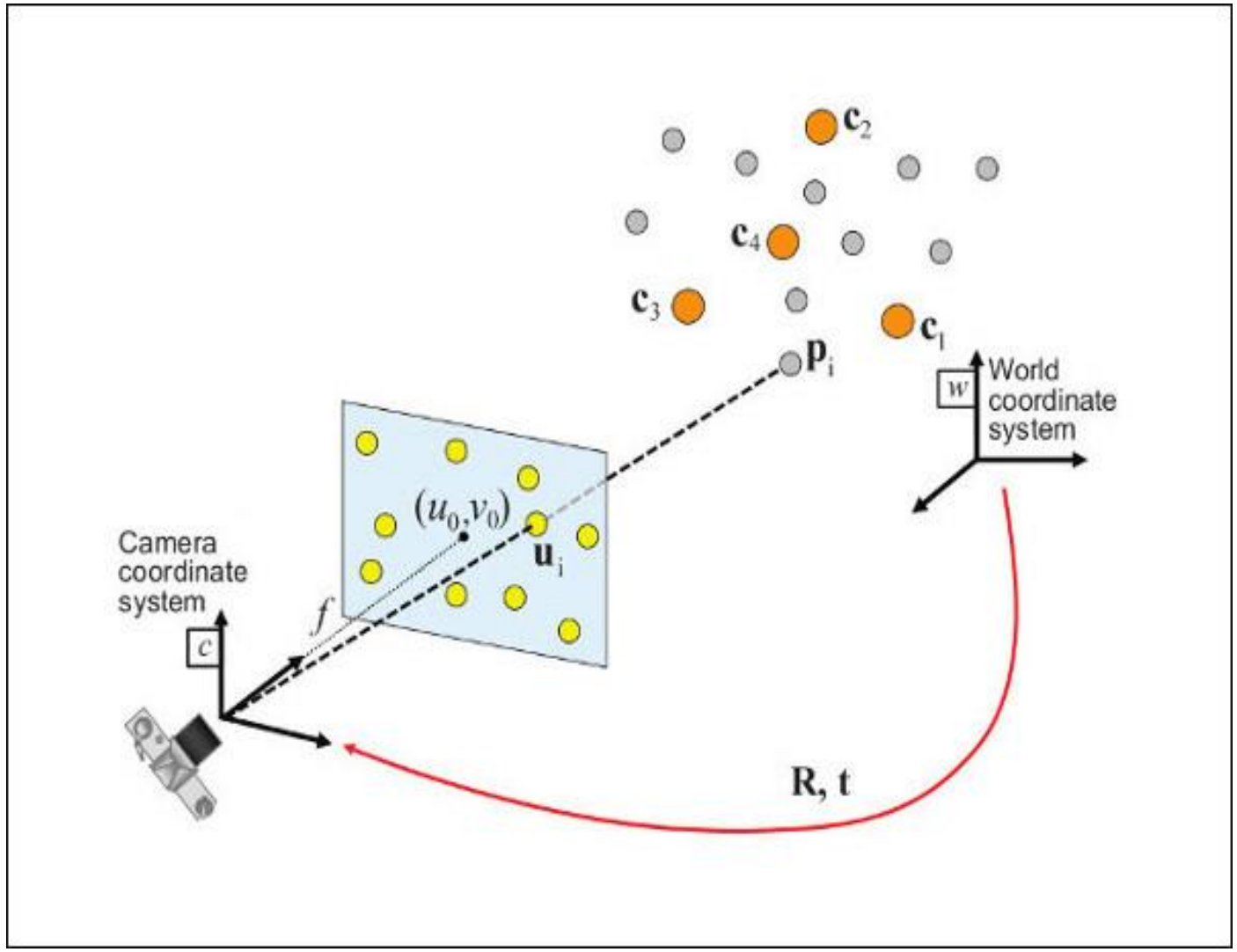
11/4

- Team Sebastian, Timothy, Jake, & Tyler
- Shopkeepr
- The Riddlers
- Touchfree

What do I expect?

- Progress update
- 10min informal presentation
- Can use your own laptop or give me slides

- Show off what you have done so far
- Have you tried things that didn't work?
- What class lessons have you incorporated into the project? Any insights?
- Have your objectives changed?
- Has there been any unanticipated issues?



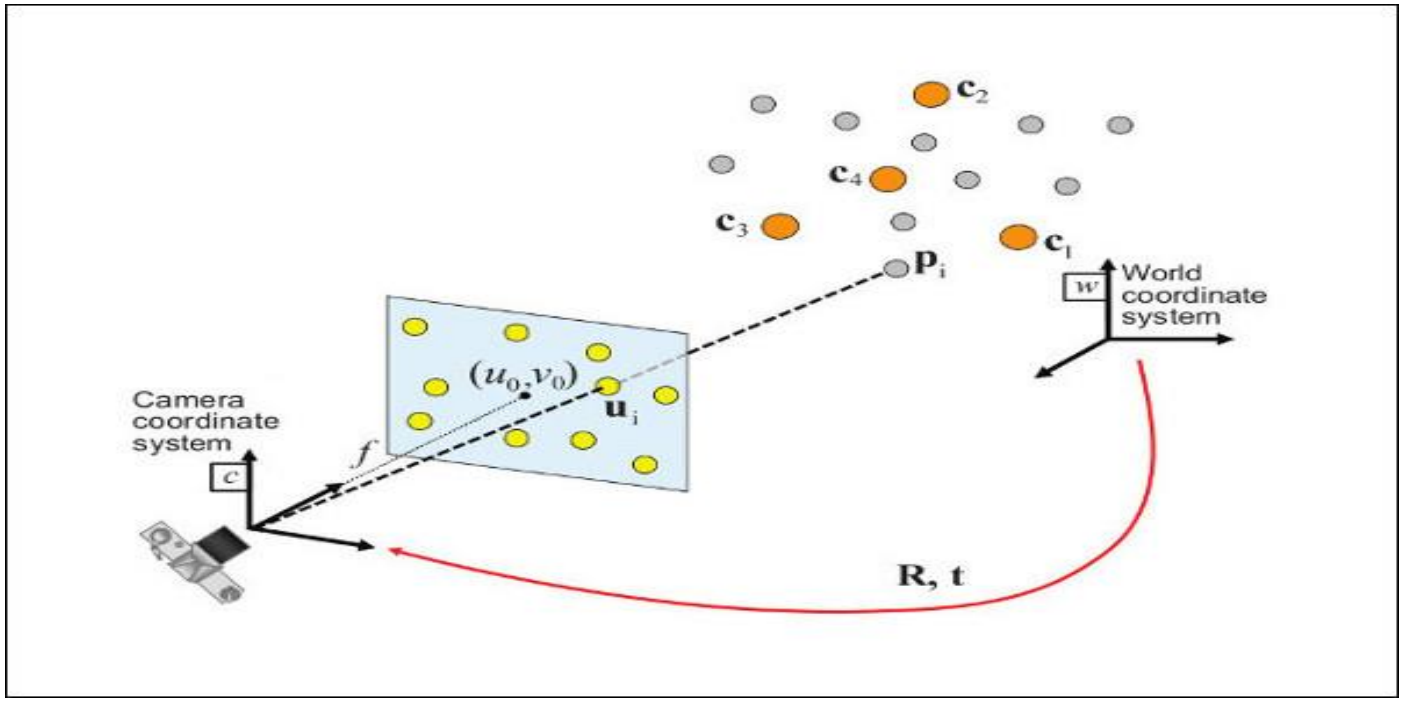
$$\begin{bmatrix} B_x \\ B_y \\ B_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & \Delta X \\ r_{1,0} & r_{1,1} & r_{1,2} & \Delta Y \\ r_{2,0} & r_{2,1} & r_{2,2} & \Delta Z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} A_x \\ A_y \\ A_z \\ 1 \end{bmatrix}$$

3D to 2D projection

$$\begin{bmatrix} u_{ideal} \\ v_{ideal} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \\ 1 \end{bmatrix}$$

Intrinsic Camera distortion (K)

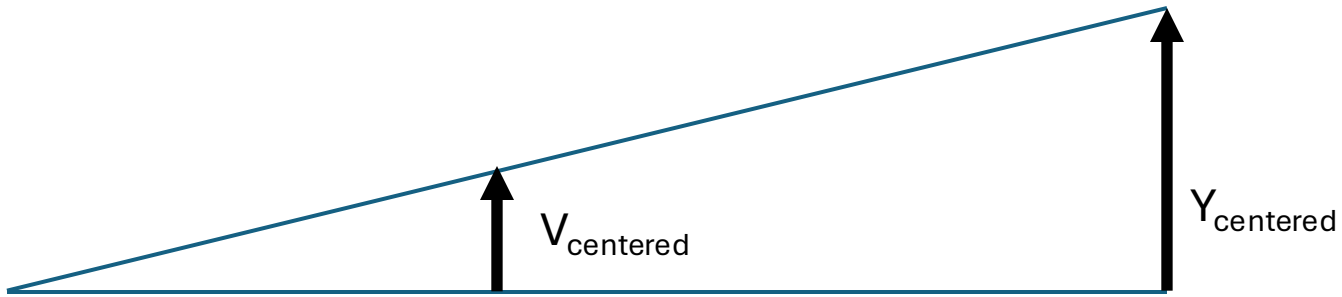
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{ideal} \\ v_{ideal} \\ 1 \end{bmatrix}$$



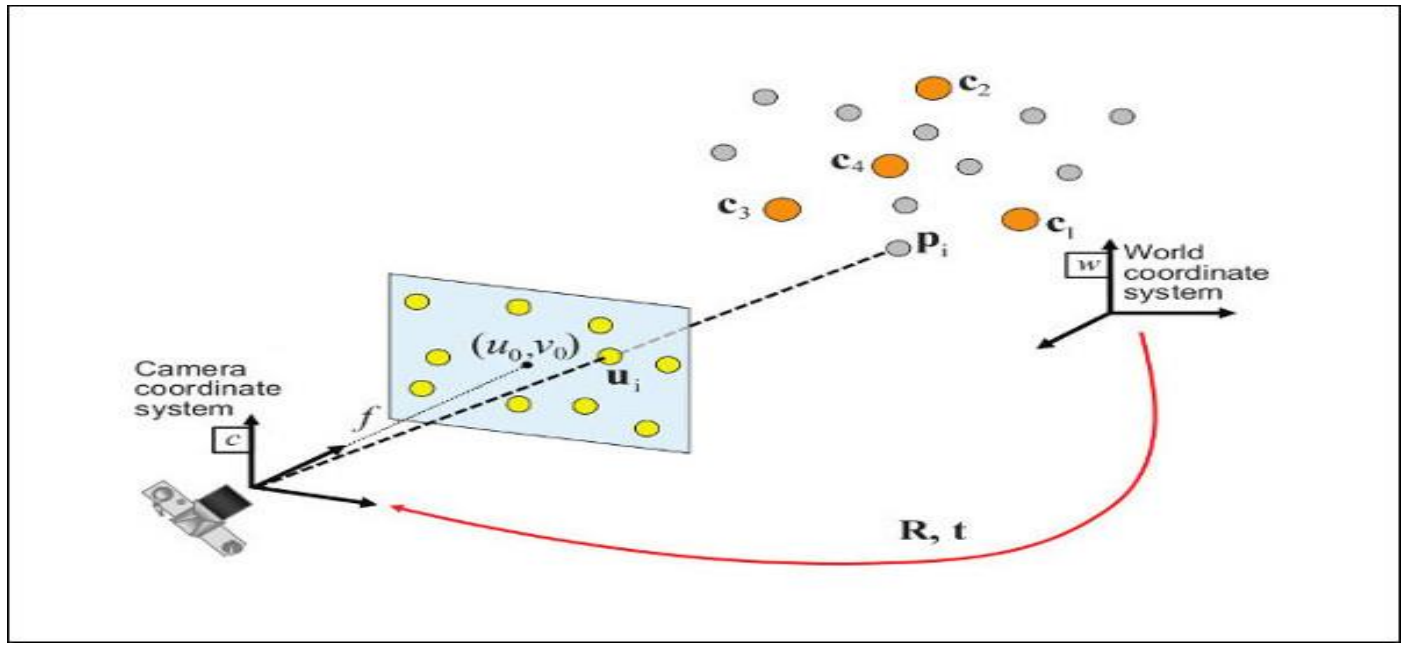
$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \frac{1}{n}\sum \overrightarrow{X_{world}}$$

$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \vec{t}$$

$$\overrightarrow{U_{image|Centered}} = S * \overrightarrow{X_{world|Centered}}$$



$$\overrightarrow{U_{image|Centered}} = S * \overrightarrow{X_{world|Centered}}$$



$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \frac{1}{n}\sum \overrightarrow{X_{world}}$$

$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \vec{t}$$

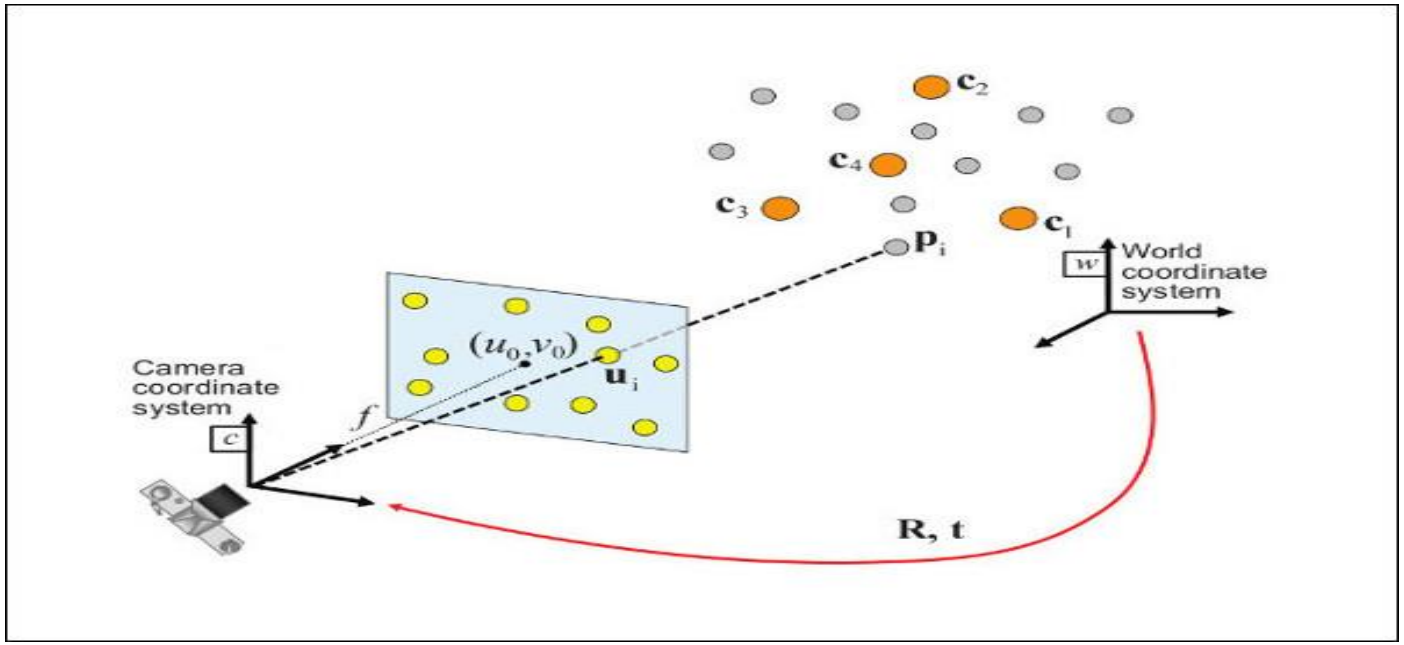
$$\begin{bmatrix} B_x \\ B_y \\ B_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} & \Delta X \\ r_{1,0} & r_{1,1} & r_{1,2} & \Delta Y \\ r_{2,0} & r_{2,1} & r_{2,2} & \Delta Z \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} A_x \\ A_y \\ A_z \\ 1 \end{bmatrix}$$

$$\overrightarrow{U_{image|Centered}} = S * \overrightarrow{X_{world|Centered}}$$

$$\begin{bmatrix} u_{ideal} \\ v_{ideal} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} B_x \\ B_y \\ B_z \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} u_c \\ v_c \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{0,0} & r_{0,1} & r_{0,2} \\ r_{1,0} & r_{1,1} & r_{1,2} \\ r_{2,0} & r_{2,1} & r_{2,2} \end{bmatrix} * \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{ideal} \\ v_{ideal} \\ 1 \end{bmatrix}$$



$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \frac{1}{n}\sum \overrightarrow{X_{world}}$$

$$\overrightarrow{X_{world|Centered}} = \overrightarrow{X_{world}} - \vec{t}$$

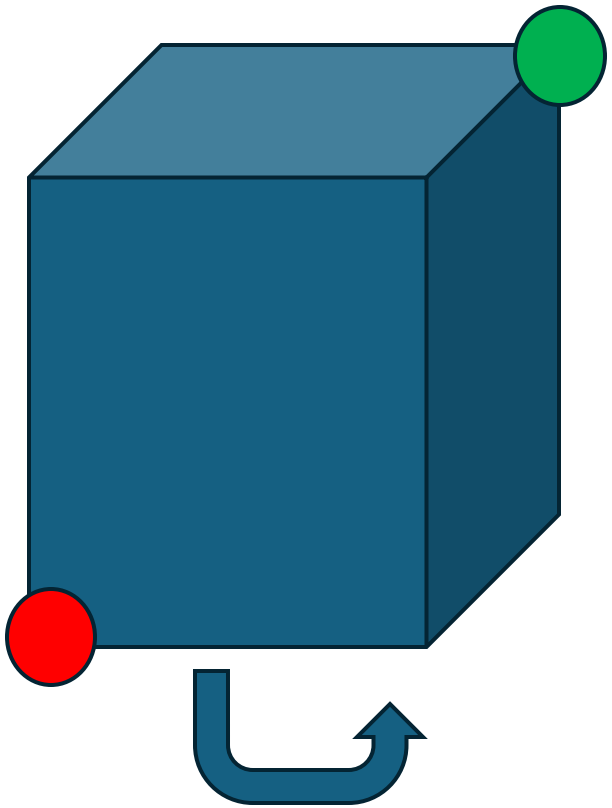
$$\overrightarrow{U_{image|Centered}} = S * \overrightarrow{X_{world|Centered}}$$

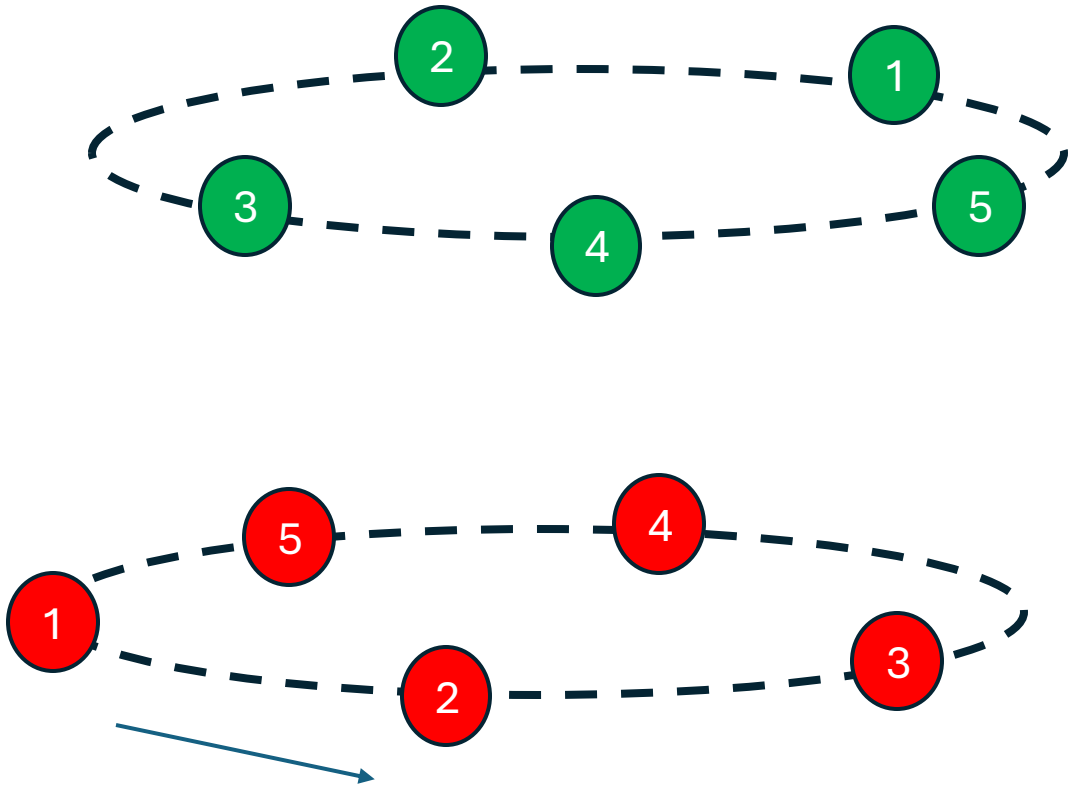
$$R=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\sin\theta_x\\ 0 & \sin\theta_x & \cos\theta_x\end{bmatrix} * \begin{bmatrix}\cos\theta_Y & 0 & \sin\theta_Y\\ 0 & 1 & 0\\ -\sin\theta_Y & 0 & \cos\theta_Y\end{bmatrix} * \begin{bmatrix}\cos\theta_Z & -\sin\theta_Z & 0\\ \sin\theta_Z & \cos\theta_Z & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$\begin{bmatrix}u_c\\ v_c\end{bmatrix} = \rho * \begin{bmatrix}X_c\\ Y_c\\ Z_c\end{bmatrix}$$

$$\rho = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix}\begin{bmatrix}r_{0,0} & r_{0,1} & r_{0,2}\\ r_{1,0} & r_{1,1} & r_{1,2}\\ r_{2,0} & r_{2,1} & r_{2,2}\end{bmatrix} = \begin{bmatrix}r_{0,0} & r_{0,1} & r_{0,2}\\ r_{1,0} & r_{1,1} & r_{1,2}\end{bmatrix}$$

Tomasi-Kanade Factorization



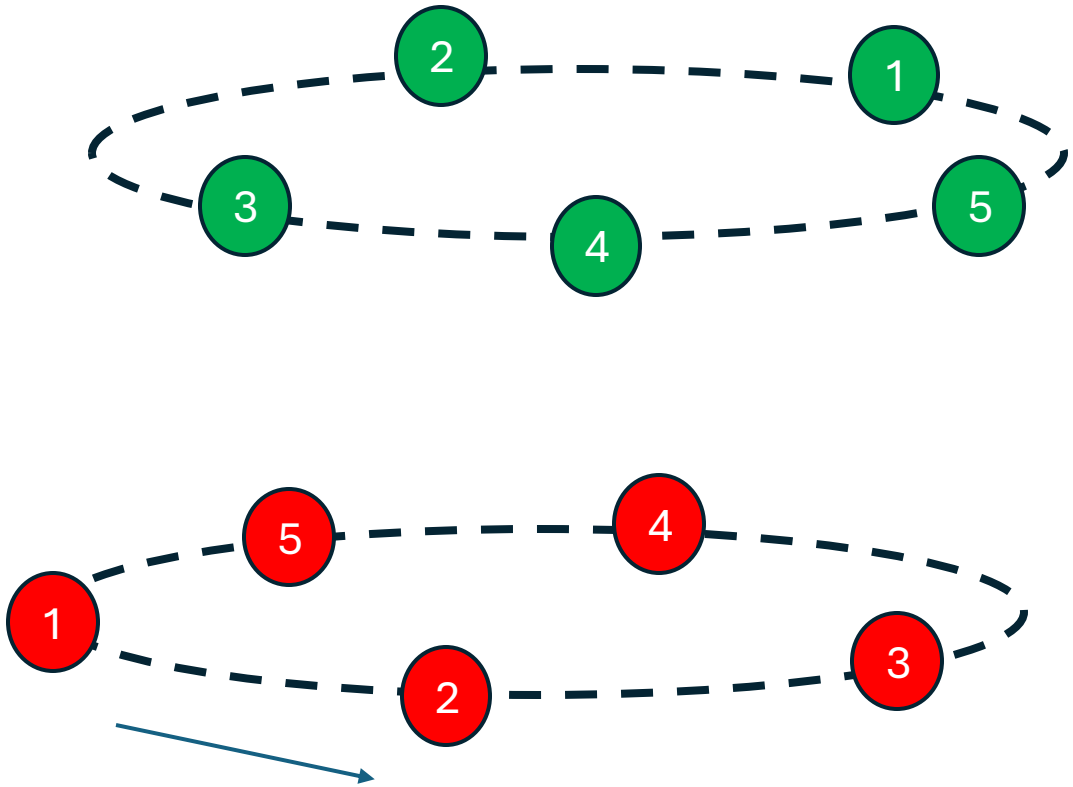


Tomasi-Kanade Factorization

Number of tracked points (centered)

$$W = \begin{bmatrix} U_{1,A} & U_{1,B} & \cdots \\ U_{2,A} & U_{2,B} & \\ U_{3,A} & U_{3,B} & \\ U_{4,A} & U_{4,B} & \\ U_{5,A} & U_{5,B} & \\ & \vdots & \\ V_{1,A} & V_{1,B} & \cdots \\ V_{2,A} & V_{2,B} & \\ V_{3,A} & V_{3,B} & \\ V_{4,A} & V_{4,B} & \\ V_{5,A} & V_{5,B} & \\ & \vdots & \end{bmatrix}$$

Number of frames (x2)



Tomasi-Kanade Factorization

Number of tracked points (centered) [M]

$$W = \begin{bmatrix} U_{1,A} & U_{1,B} & \cdots \\ U_{2,A} & U_{2,B} & \\ U_{3,A} & U_{3,B} & \\ U_{4,A} & U_{4,B} & \\ U_{5,A} & U_{5,B} & \\ & \vdots & \\ V_{1,A} & V_{1,B} & \cdots \\ V_{2,A} & V_{2,B} & \\ V_{3,A} & V_{3,B} & \\ V_{4,A} & V_{4,B} & \\ V_{5,A} & V_{5,B} & \\ & \vdots & \end{bmatrix}$$

Number of frames (N x 2)

$$W^T = \mathrm{P} * X^T$$

W = < M x 2 N >

Sub-matrix that depends on movement (rho)
< 2N x 3 >

Sub-matrix that depends on shape (X)
< M x 3 >

Tomasi-Kanade Factorization

Number of tracked points (centered)

$$W = \begin{bmatrix} U_{1,A} & U_{1,B} & \cdots \\ U_{2,A} & U_{2,B} & \\ U_{3,A} & U_{3,B} & \\ U_{4,A} & U_{4,B} & \\ U_{5,A} & U_{5,B} & \\ & \vdots & \\ V_{1,A} & V_{1,B} & \cdots \\ V_{2,A} & V_{2,B} & \\ V_{3,A} & V_{3,B} & \\ V_{4,A} & V_{4,B} & \\ V_{5,A} & V_{5,B} & \\ & \vdots & \end{bmatrix}$$

Number of frames (x2)

$$W^T = \mathrm{P} * X^T$$

$$SVD(W^T) \rightarrow WT = \mathrm{U} * \mathrm{S} * \mathrm{V}^\mathsf{T}$$

P is rank 3

X is rank 3

Therefore, W is also (ideally) rank 3

$$P = U_{<2Nx3>} * S^{1/2}_{<3x3>}$$

$$X = V_{<Mx3>} * S^{1/2}_{<3x3>}$$

Tomasi-Kanade Factorization

$$W = \begin{bmatrix} U_{1,A} & U_{1,B} & \cdots \\ U_{2,A} & U_{2,B} & \\ U_{3,A} & U_{3,B} & \\ U_{4,A} & U_{4,B} & \\ U_{5,A} & U_{5,B} & \\ & \vdots & \\ V_{1,A} & V_{1,B} & \cdots \\ V_{2,A} & V_{2,B} & \\ V_{3,A} & V_{3,B} & \\ V_{4,A} & V_{4,B} & \\ V_{5,A} & V_{5,B} & \\ & \vdots & \end{bmatrix}$$

$$W^T = (\mathrm{P} * Q) * (Q^{-1} * X^T)$$

However, this is not a unique solution:

Metric constraints:
We want a rotation matrix that uses orthogonal axes

For i in range(1:N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(N+1:2*N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(1:N)

$$(P_i * Q) * (P_{i+N} * Q)^T = 0$$

Tomasi-Kanade Factorization

$$W = \begin{bmatrix} U_{1,A} & U_{1,B} & \cdots \\ U_{2,A} & U_{2,B} & \\ U_{3,A} & U_{3,B} & \\ U_{4,A} & U_{4,B} & \\ U_{5,A} & U_{5,B} & \\ & \vdots & \\ V_{1,A} & V_{1,B} & \cdots \\ V_{2,A} & V_{2,B} & \\ V_{3,A} & V_{3,B} & \\ V_{4,A} & V_{4,B} & \\ V_{5,A} & V_{5,B} & \\ & \vdots & \end{bmatrix}$$

$$W^T = (\mathrm{P} * Q) * (Q^{-1} * X^T)$$

However, this is not a unique solution:

Metric constraints:
We want a rotation matrix that uses orthogonal axes

For i in range(1:N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(N+1:2*N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(1:N)

$$(P_i * Q) * (P_{i+N} * Q)^T = 0$$

Tomasi-Kanade Factorization

$$P_i * Q * QT * Pi^T = 1$$

$$P_i * Q * QT * P_{N+i}^T = 0$$

$$\begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{2,1} & q_{2,2} & q_{2,3} \\ q_{3,1} & q_{3,2} & q_{3,3} \end{bmatrix} = Q * QT$$

Metric constraints:

We want a rotation matrix that uses orthogonal axes

For i in range(1:N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(N+1:2*N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(1:N)

$$(P_i * Q) * (P_{i+N} * Q)^T = 0$$

$$\begin{bmatrix} P_{i,1} * q_{1,1} + P_{i,2} * q_{2,1} + P_{i,3} * q_{3,1} \\ P_{i,1} * q_{1,2} + P_{i,2} * q_{2,2} + P_{i,3} * q_{3,2} \\ P_{i,1} * q_{1,3} + P_{i,2} * q_{2,3} + P_{i,3} * q_{3,3} \end{bmatrix} * [P_{i,1} \quad P_{i,2} \quad P_{i,3}] =$$

$$P_{i,1} * (P_{i,1} * q_{1,1} + P_{i,2} * q_{2,1} + P_{i,3} * q_{3,1}) +$$
$$P_{i,2} * (P_{i,1} * q_{1,2} + P_{i,2} * q_{2,2} + P_{i,3} * q_{3,2}) +$$
$$P_{i31} * (P_{i,1} * q_{1,3} + P_{i,2} * q_{2,3} + P_{i,3} * q_{3,3}) = 1$$

Tomasi-Kanade Factorization

$P_{i,1} * (P_{i,1} * q_{1,1} + P_{i,2} * q_{2,1} + P_{i,3} * q_{3,1}) +$
$P_{i,2} *(P_{i,1} * q_{1,2} + P_{i,2} * q_{2,2} + P_{i,3} * q_{3,2}) +$
$P_{i,3} *(P_{i,1} * q_{1,3} + P_{i,2} * q_{2,3} + P_{i,3} * q_{3,3}) = 1$

$P_{N+i,1} * (P_{i,1} * q_{1,1} + P_{i,2} * q_{2,1} + P_{i,3} * q_{3,1}) +$
$P_{N+i,2} *(P_{i,1} * q_{1,2} + P_{i,2} * q_{2,2} + P_{i,3} * q_{3,2}) +$
$P_{N+,3} *(P_{i,1} * q_{1,3} + P_{i,2} * q_{2,3} + P_{i,3} * q_{3,3}) = 0$

$$\text{Chol} \begin{bmatrix} q_{1,1} & q_{1,2} & q_{1,3} \\ q_{2,1} & q_{2,2} & q_{2,3} \\ q_{3,1} & q_{3,2} & q_{3,3} \end{bmatrix} = Q$$

Metric constraints:
We want a rotation matrix that uses orthogonal axes

For i in range(1:N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(N+1:2*N)

$$(P_i * Q) * (P_i * Q)^T = 1$$

For i in range(1:N)

$$(P_i * Q) * (P_{i+N} * Q)^T = 0$$

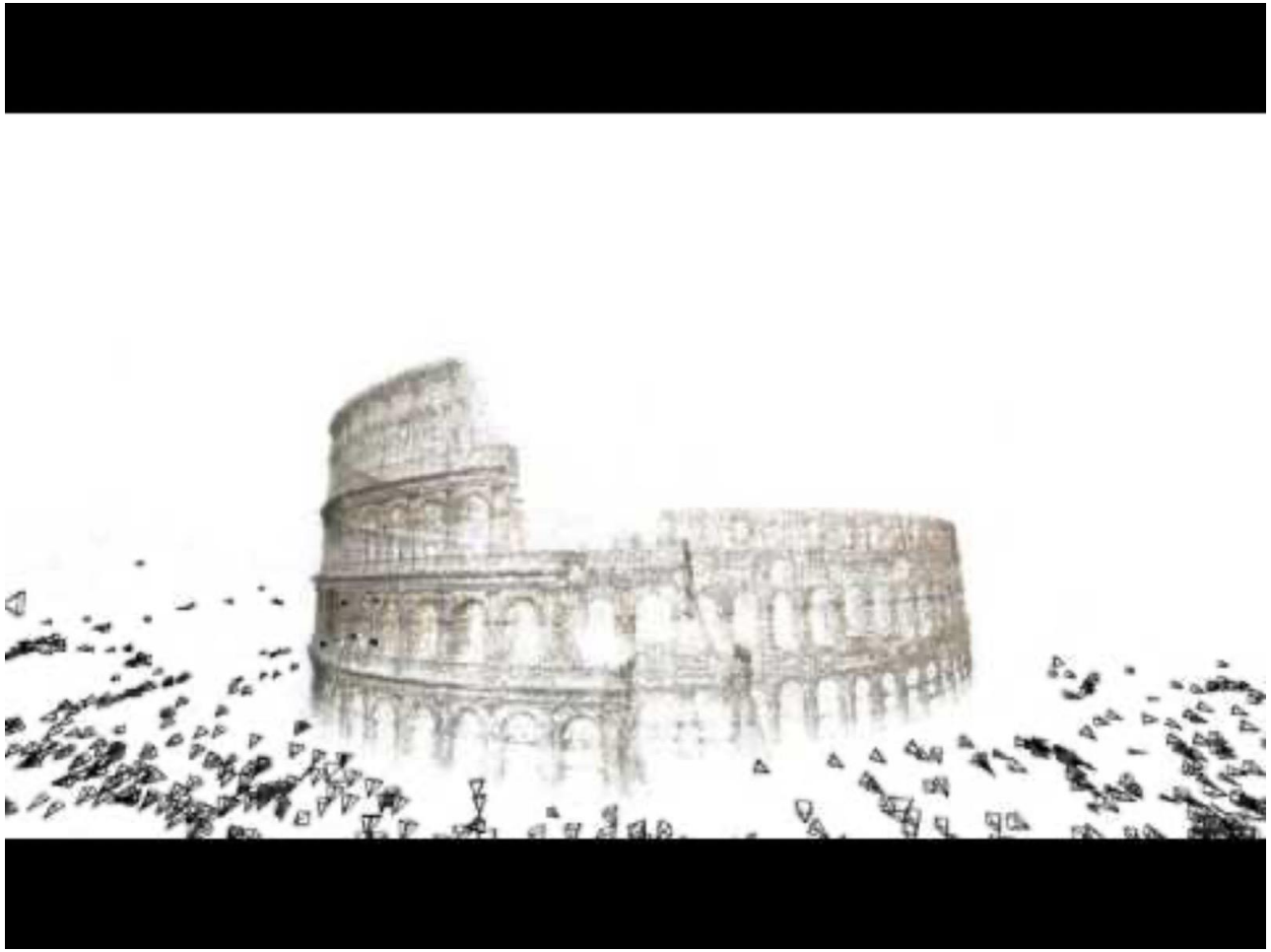
Tomasi-Kanade Factorization

Steps:

1. Find matching points (M) across frames (N)
2. Construct the **W** matrix < M x 2*N >
3. SVD on **W -> U*S*V^T.** Keep only top 3 terms
4. P’ = U *$S^{1/2}$ *(non-unique P)*
5. X ‘= V *$S^{1/2}$ *(non-unique X)*
6. Use P’ to find Q^TQ
7. Chol(Q^TQ) = Q
8. P = P’*Q. *(unique P)*
9. $X^T = Q^{-1}*X'^T$ *(unique X)*

COLMAP

Building Rome in a Day
Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski
International Conference on Computer Vision, 2009, Kyoto, Japan.

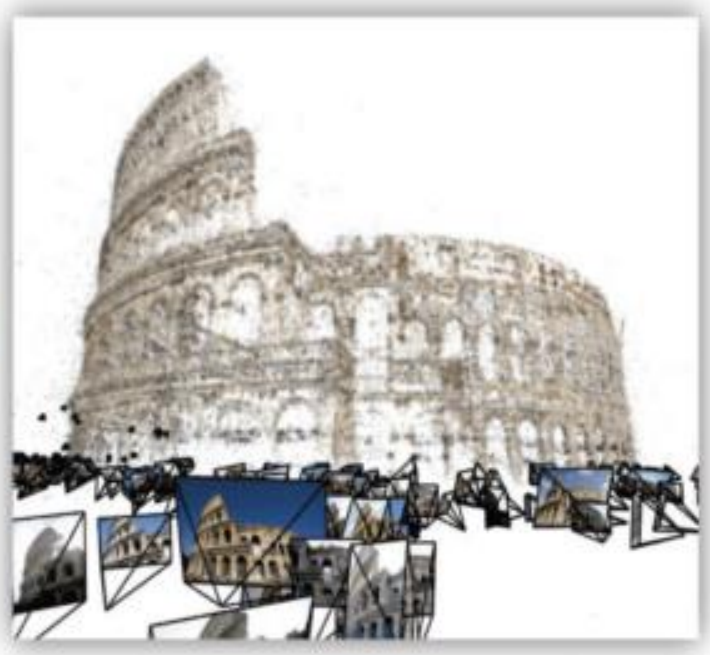


COLMAP

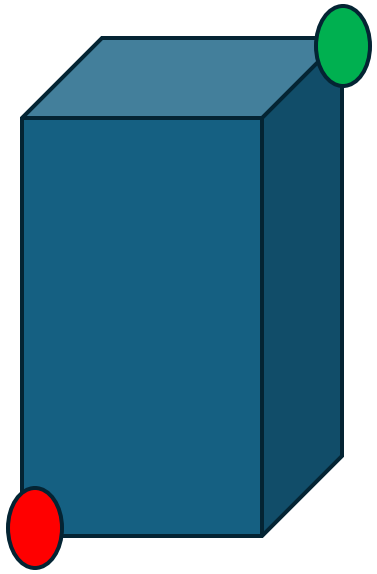
Building Rome in a Day
Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski
International Conference on Computer Vision, 2009, Kyoto, Japan.

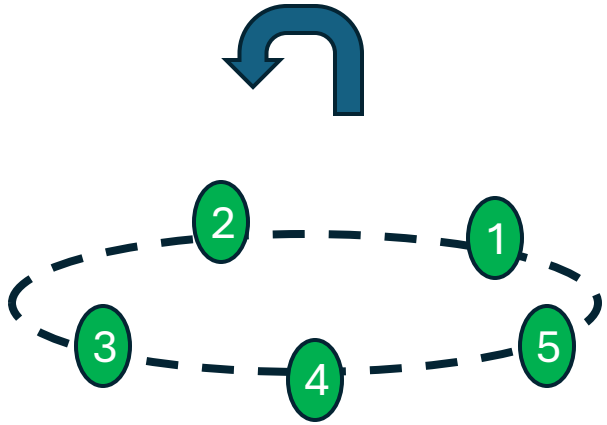
Structure from Motion

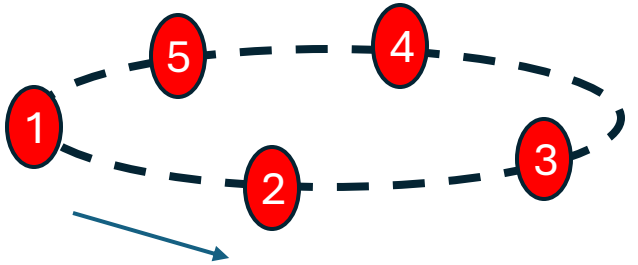
Stereo Fusion

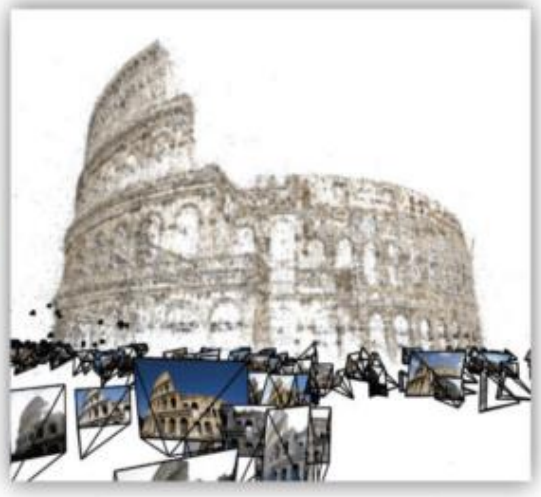


COLMAP









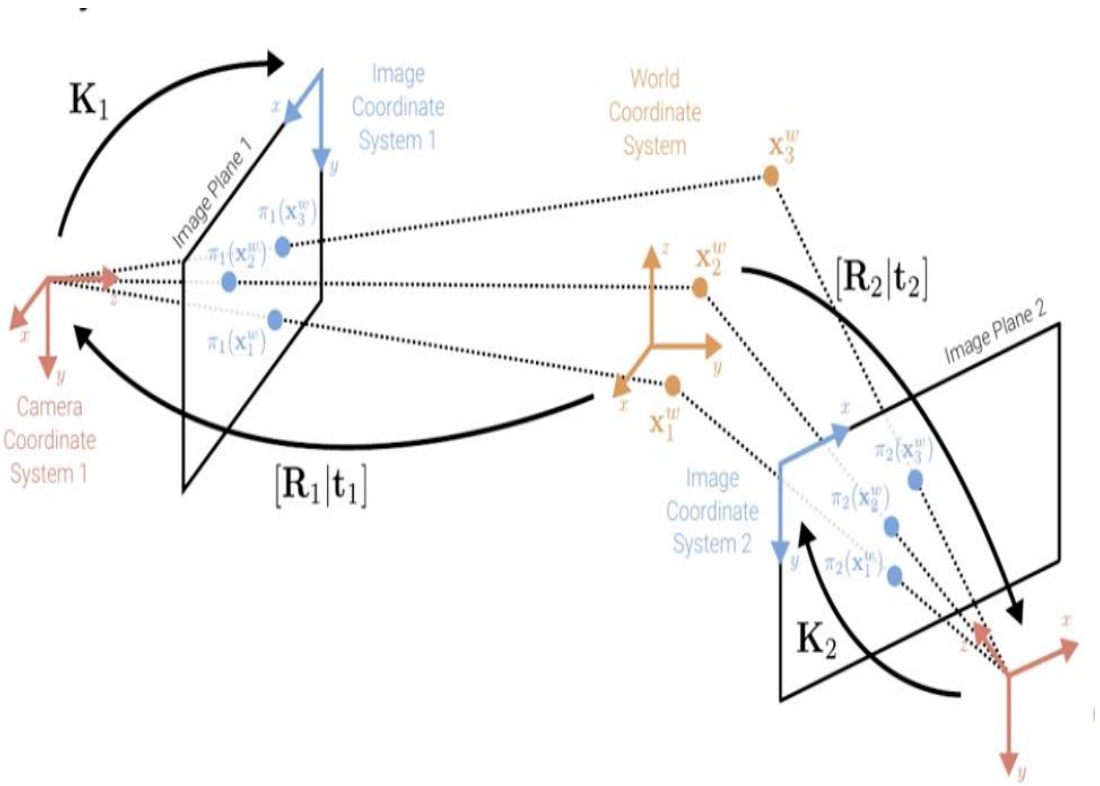
Moving the object is the same as moving the camera

COLMAP

$\Pi = \{\pi_i\}$: Intrinsic (K_i) and Extrinsic (R_i, T_i) properties for N cameras

$X = \{x_j\}$: Set of P 3D points in the world

$U = \{u_j\}$: Set of P 2D points in the image



Masking variable

$$\{\Pi, X\} = \arg\min \sum_{i=1}^{N} \sum_{j=1}^{P} w_{i,j} \left\| u_j - \pi_i\,(x_j) \right\|^2$$

Image measured point

Image projected point

Camera projector function

$$\pi_i\,(x_j) = K_i\left(R_i x_j + t_i\right)$$

COLMAP

$\Pi = \{\pi_i\}$: Intrinsic (K_i) and Extrinsic (R_i, T_i) properties for N cameras

$X = \{x_j\}$: Set of P 3D points in the world

$U = \{u_j\}$: Set of P 2D points in the image

- Large scale minimization
- Not convex (sensitive to local minima)
- Requires good initial guess
- Computationally intense

- Solution is sparse (make use of sparse solvers e.g. Google's Ceres)

Masking variable

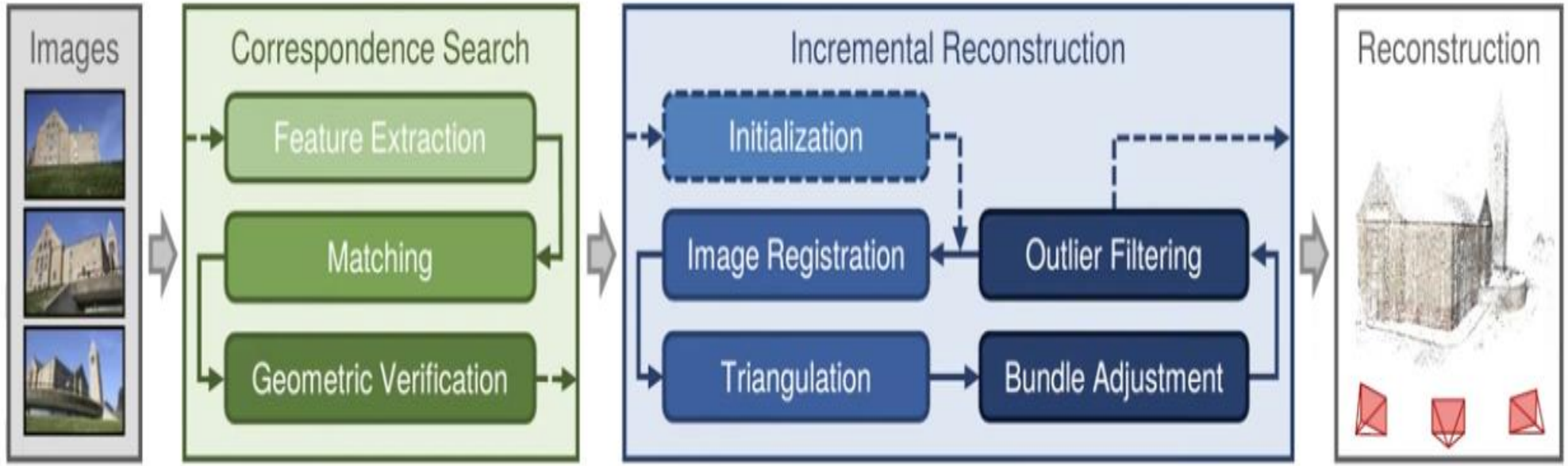
$$\{\Pi, X\} = \arg\min \sum_{i=1}^{N} \sum_{j=1}^{P} w_{i,j} \left\| u_j - \pi_i\ (x_j) \right\|^2$$

Image measured point

Image projected point

Camera projector function

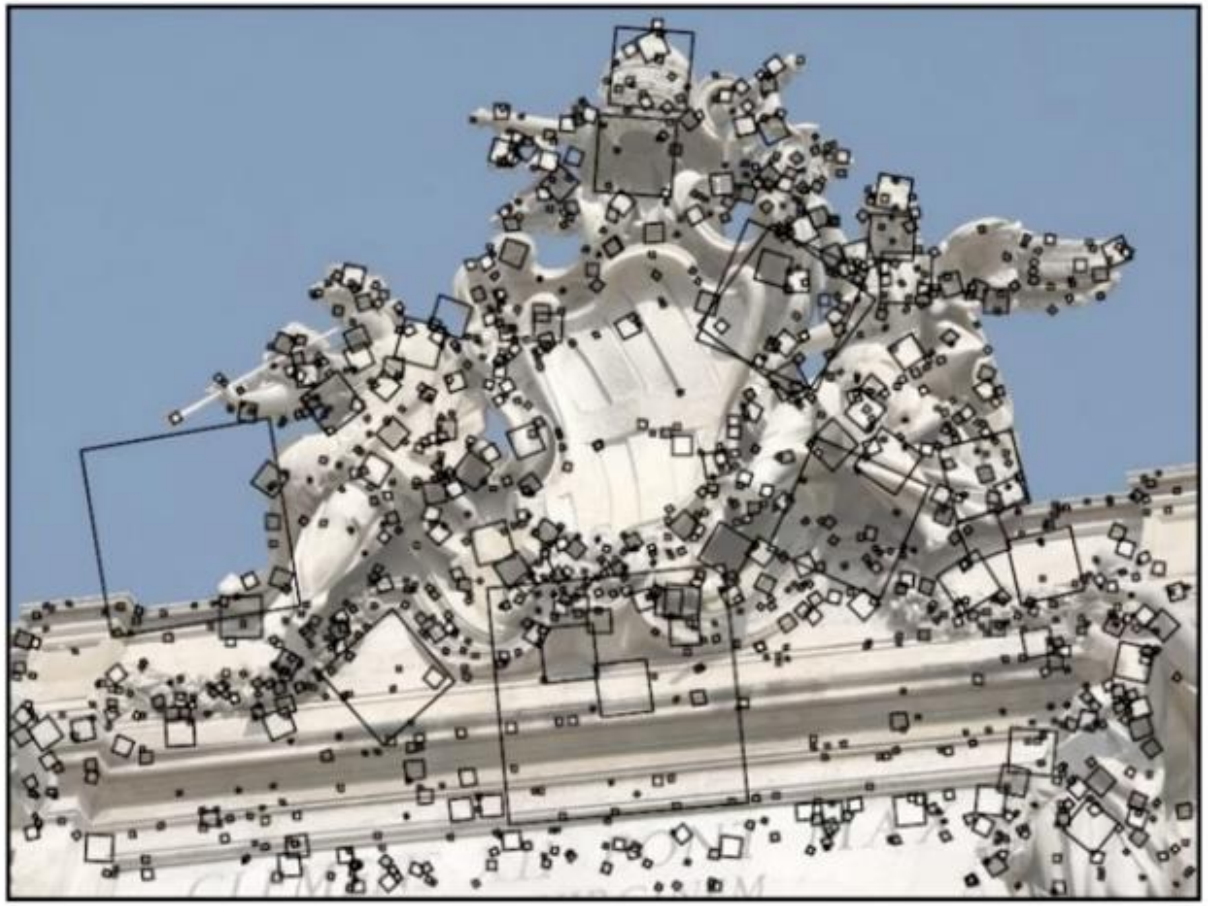
$$\pi_i\ (x_j) = K_i (R_i x_j + t_i)$$



Feature matching and reconstruction pipeline (COLMAP):

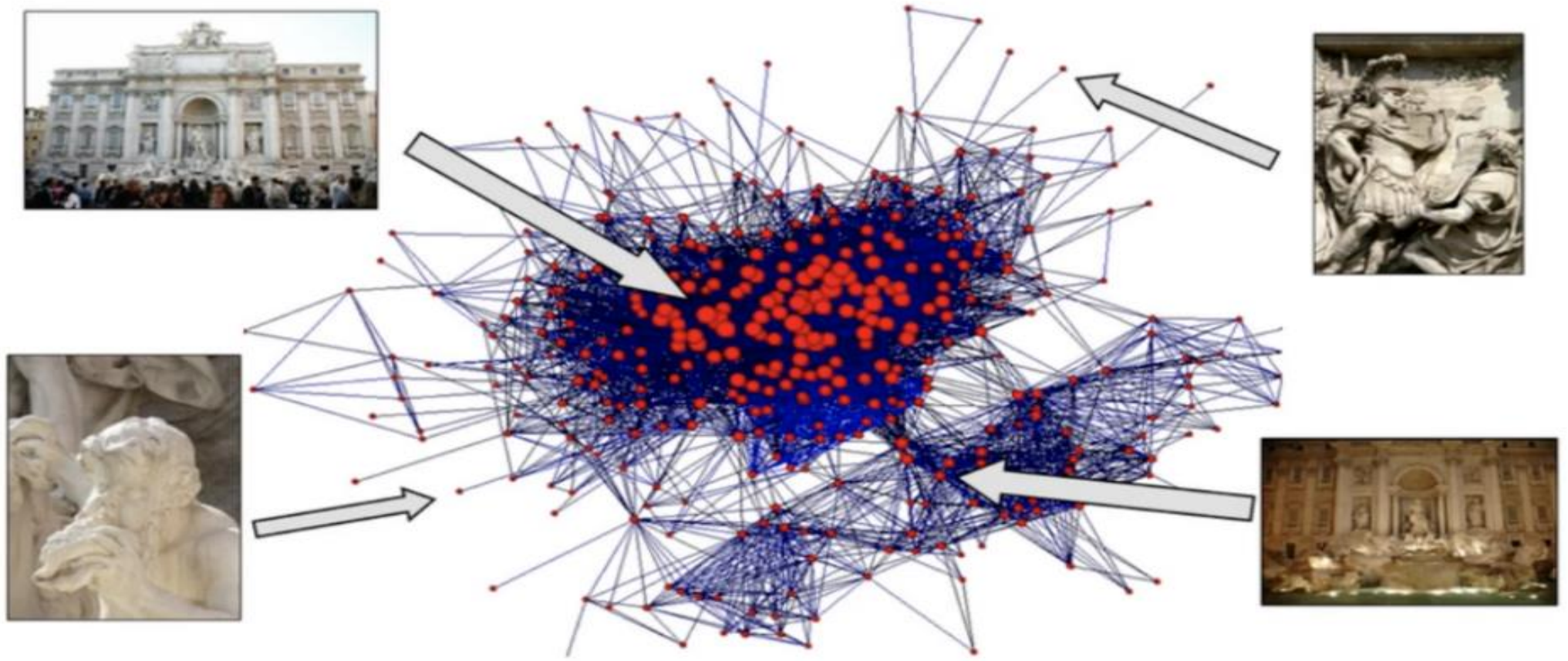
- **Correspondence Search:** Find and match robust 2D features acroess images
- **Incremental Reconstruction:** Start with 2 views, incrementally add cameras

Step 1. Find the features for all images



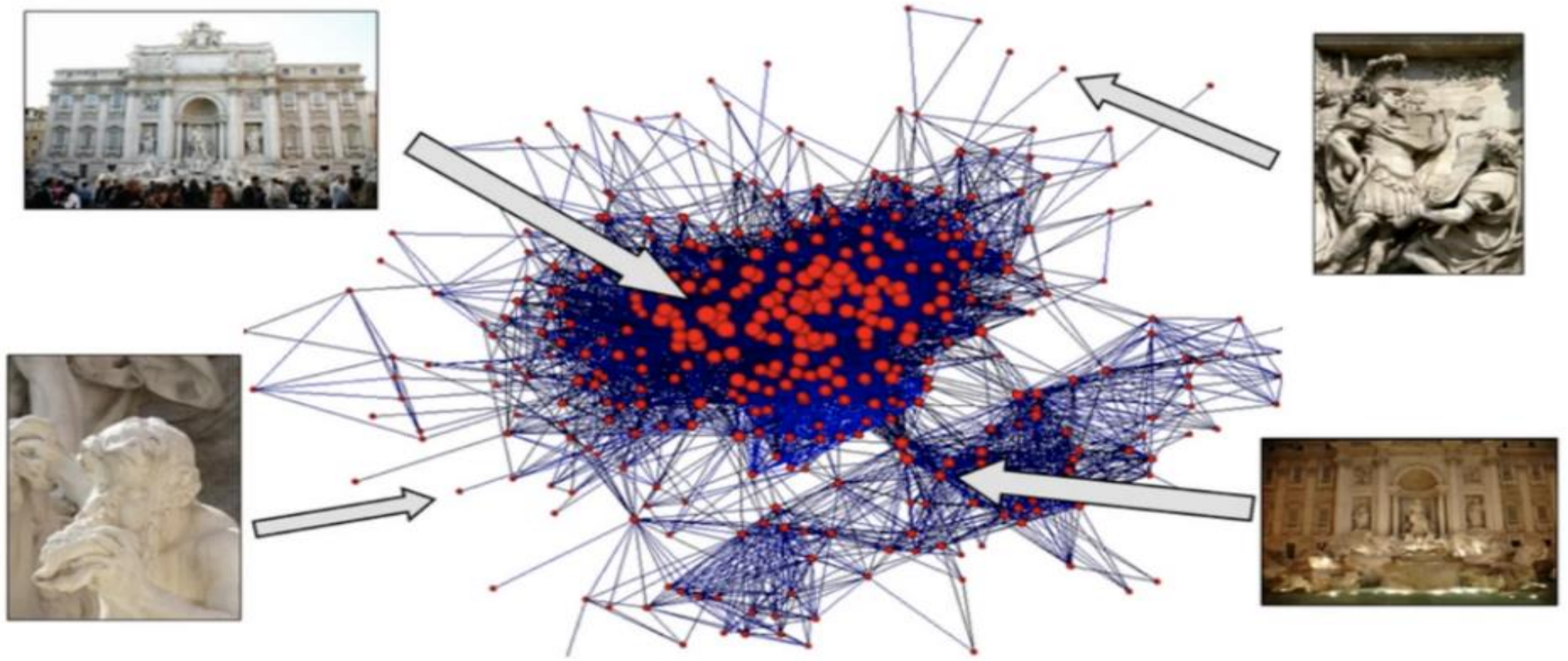
- **Detect features** (eg, SIFT, SURF, BRISK) in all input images

Step 2. Create graph of all-to-all image matches



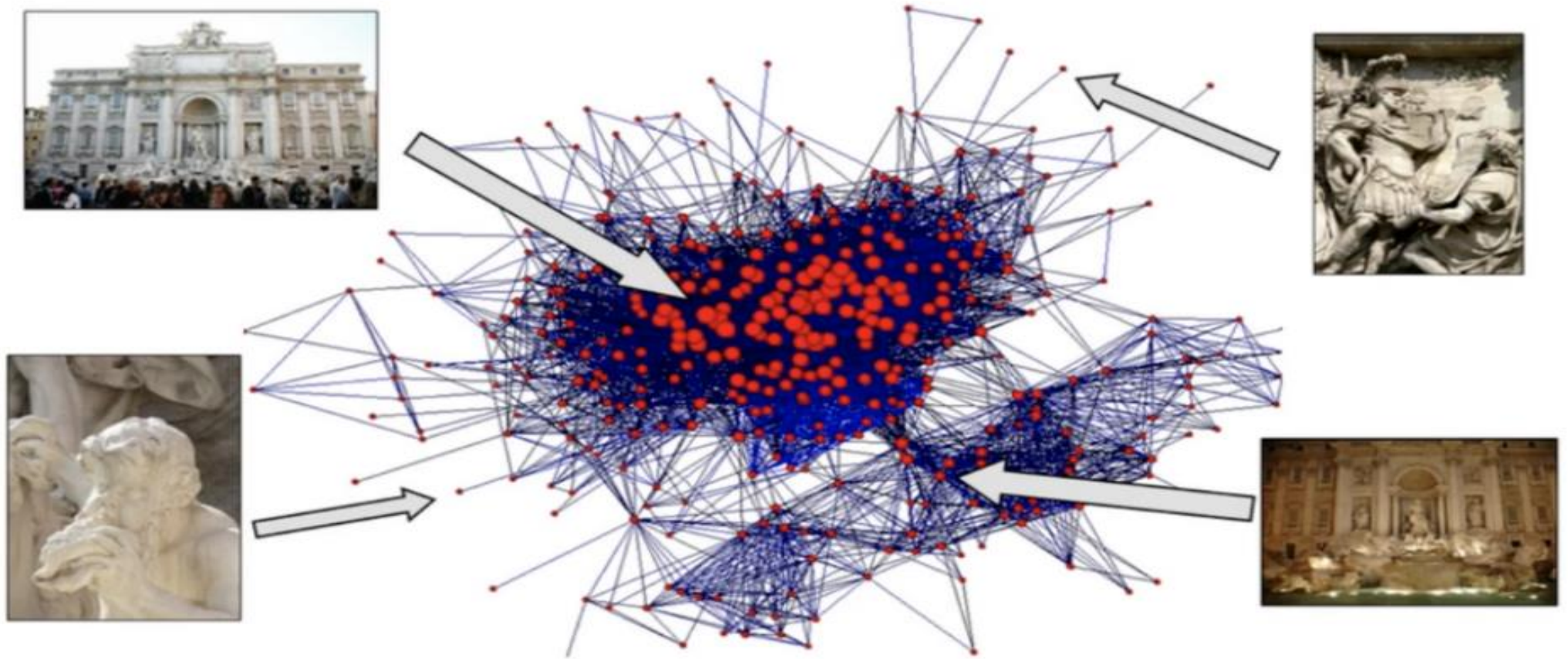
- Find **overlapping image pairs** and **associated feature correspondences**

Step 3. Prune matches (geometric verification)



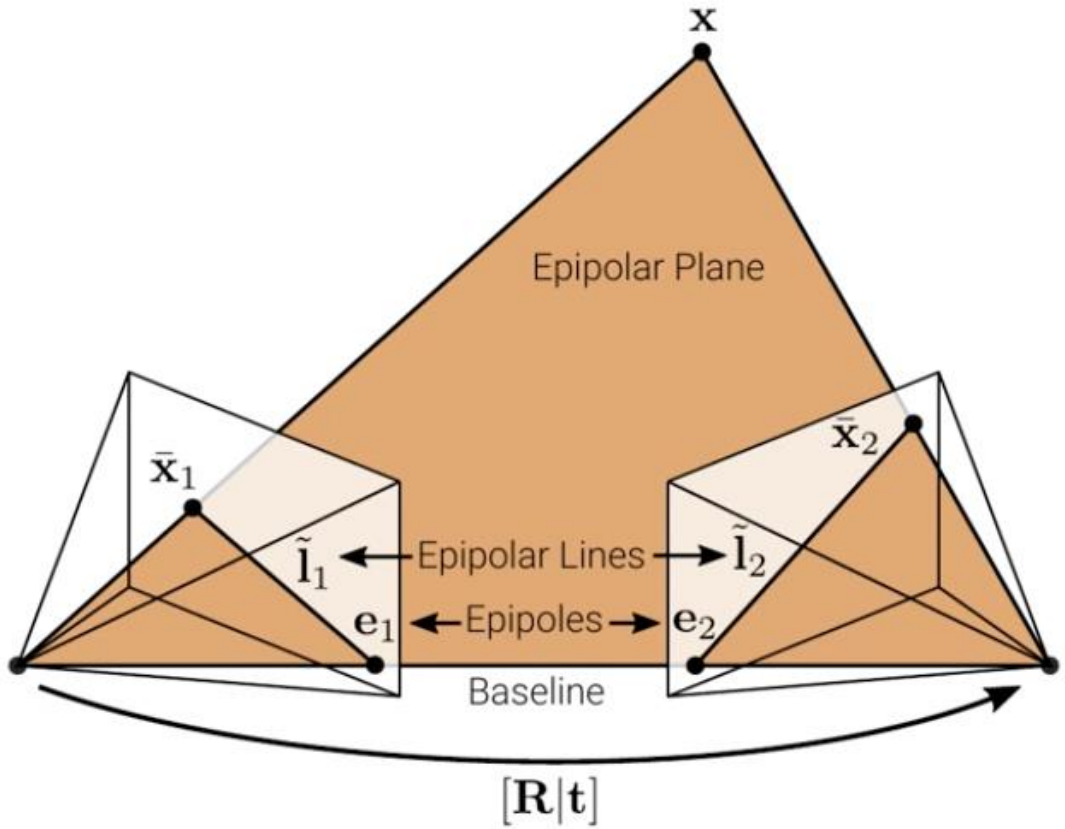
- Find **overlapping image pairs** and **associated feature correspondences**

Step 4. Find most centrally connected pair of images



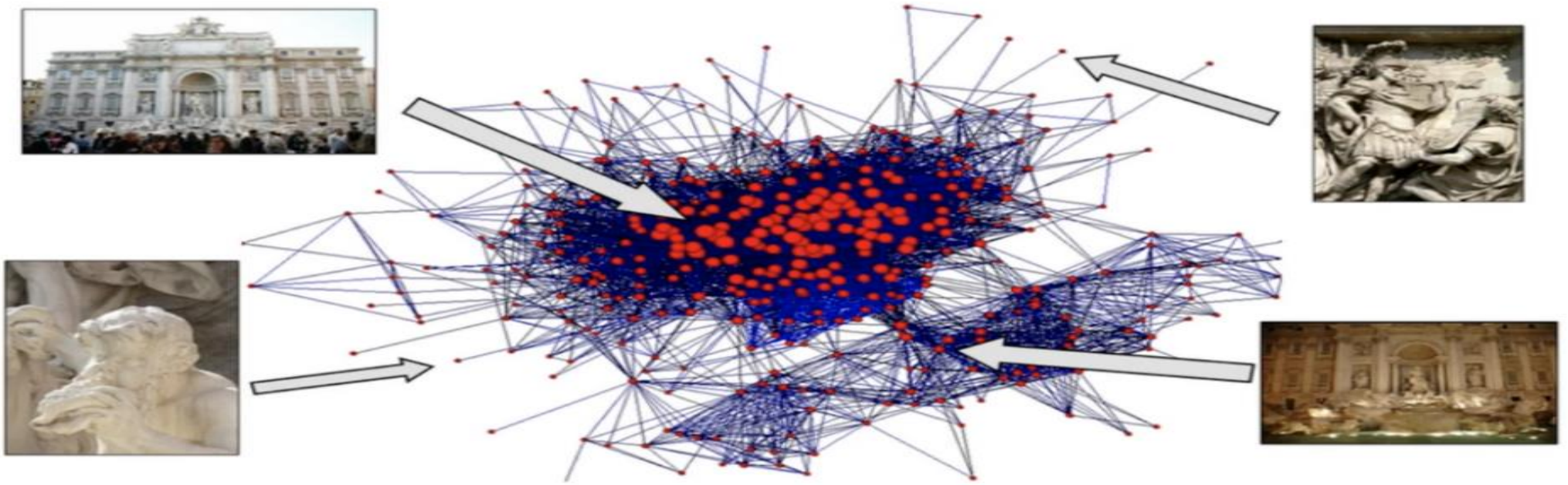
- Find **overlapping image pairs** and **associated feature correspondences**

Step 5. Two image initialization



- Select **two views** with many correspondences and **estimate their geometry**

Step 6. Add new image

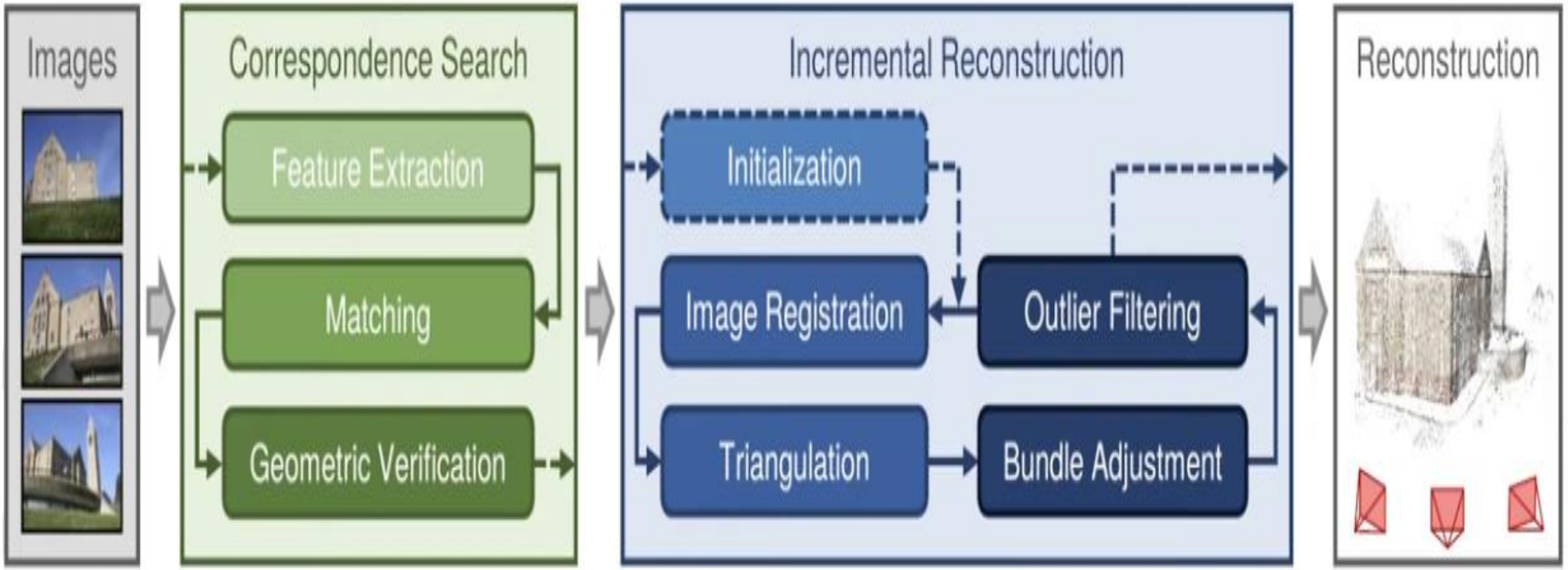


Already know this from previous step

$$\{\Pi, X\} = \arg\min \sum_{i=1}^{N} \sum_{j=1}^{P} w_{i,j} \left\| u_j - \pi_i\,(x_j) \right\|^2$$

Solve this for the new image

$$\pi_i\,(x_j) = K_i\left(R_i x_j + t_i\right)$$

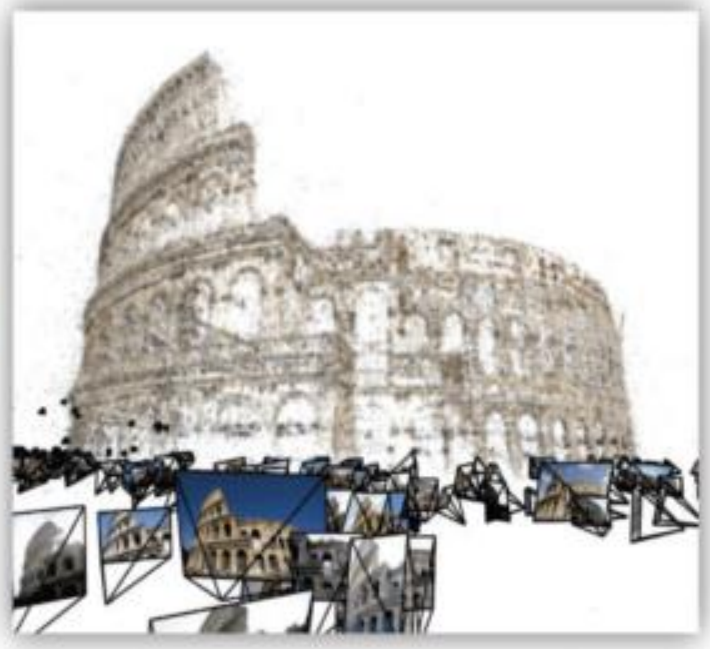


RANSAC:
Random Sampling Consensus

1) Grab small subsets of the data to solve the problem (e.g. features to solve the homography)
2) Based on that solution, compute for all other points if they are outliers or inliers
3) Repeat 1-2
4) Remove points that produce the lowest number inliers

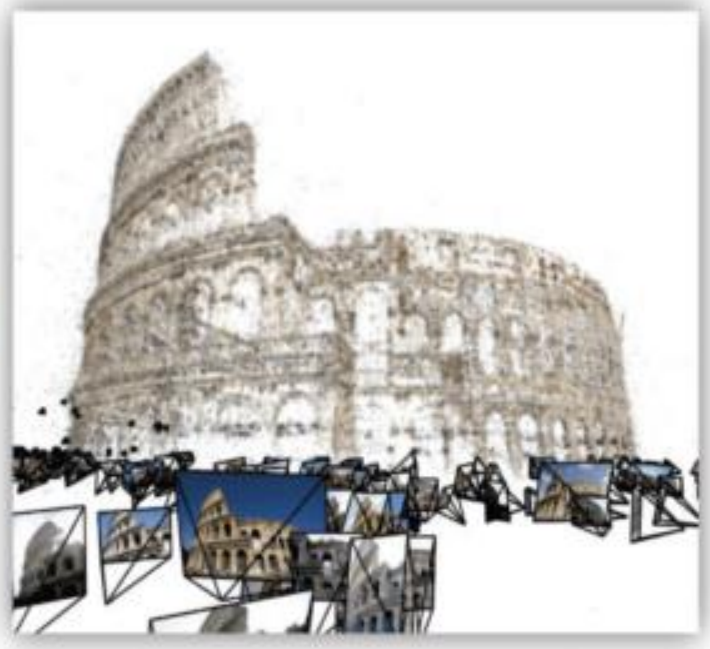
Sparse Reconstruction

Dense Reconstruction



Sparse Reconstruction

Dense Reconstruction

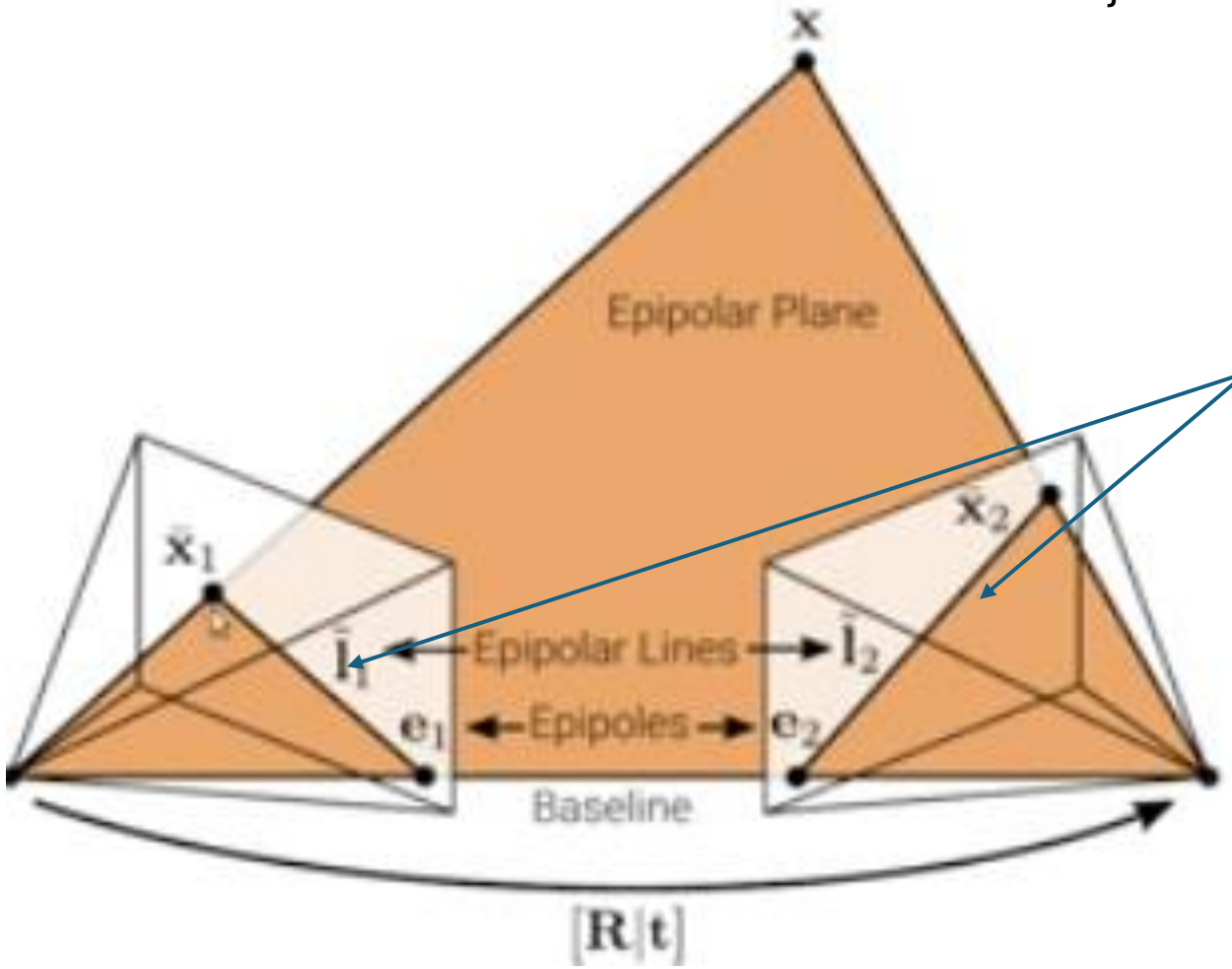


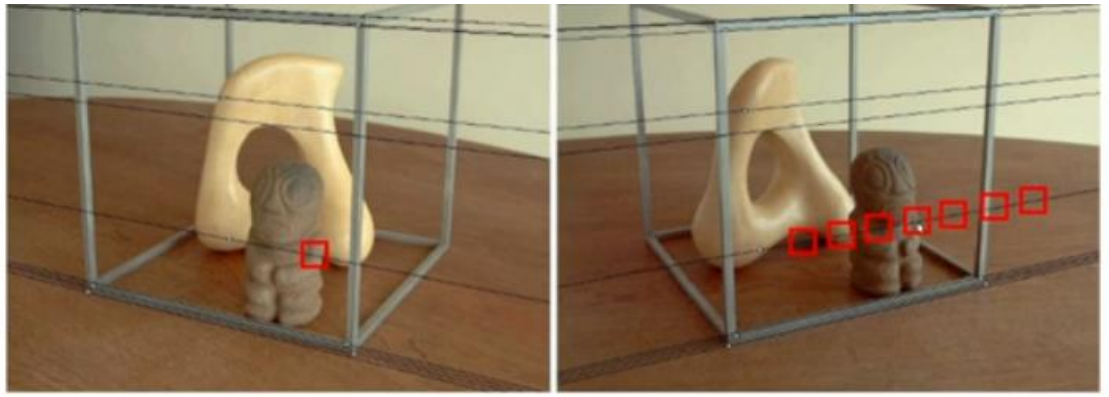
Epipolar coordinates

Projection by the Essential matrix produces a (epipolar) line

$$I_2 = E * u_1'$$

$$I_1 = E^T * u_2'$$





- E, mask = cv2.findEssentialMat(pts1, pts2, K, method=cv2.RANSAC, prob=0.999, threshold=1.0)
- _, R, t, _ = cv2.recoverPose(E, pts1, pts2, K, mask=mask)