# 3D Reconstruction Pipelines

## GVG 2022 - Lecture 13

CZECH INSTITUTE
OF INFORMATICS
ROBOTICS AND
CYBERNETICS
CTU IN PRAGUE

Torsten Sattler

# GVG - Brief Summary

- Previously in the GVG lecture:
  - Absolute camera pose estimation
  - Homograph estimation
  - Fundamental matrix estimation
  - Reconstruction from two views
  - Essential matrix estimation
- **This lecture**: Putting things together for full 3D reconstruction

# Structure-from-Motion (SfM)

Input: images

Output: (sparse) 3D point cloud, camera poses

model computed using Colmap

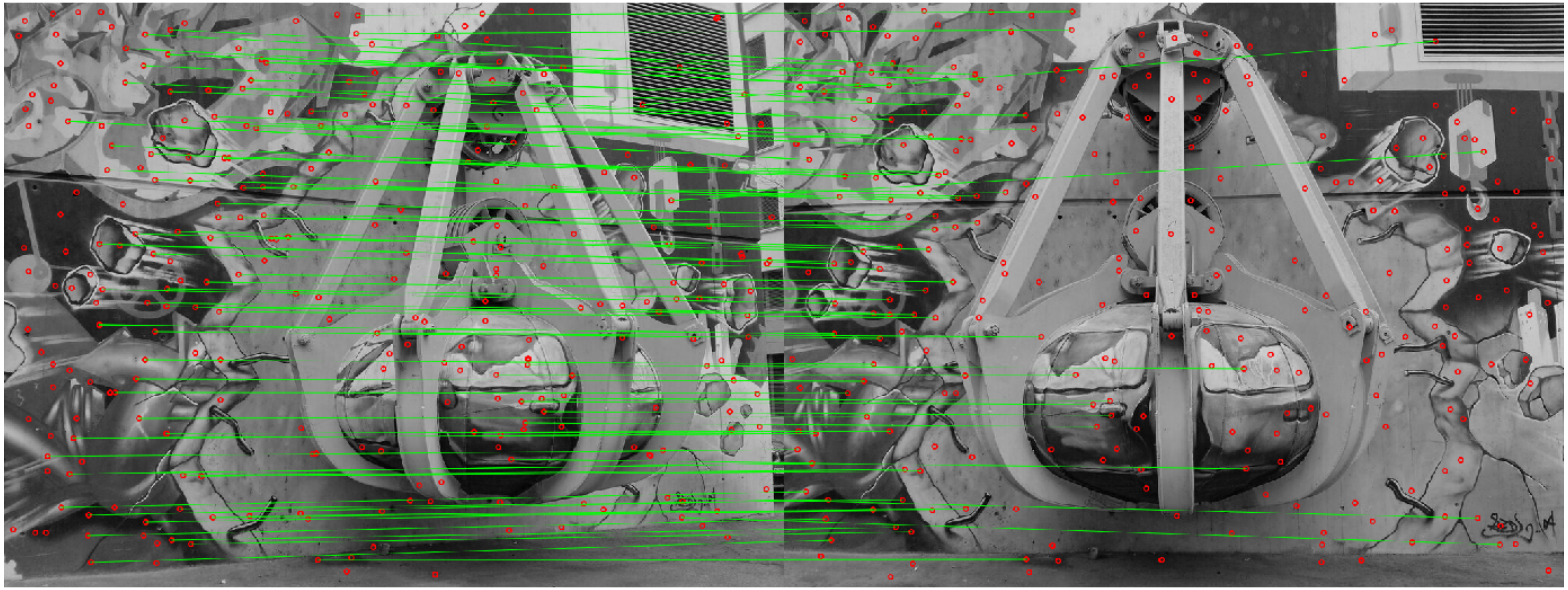
# Sequential / Incremental SfM

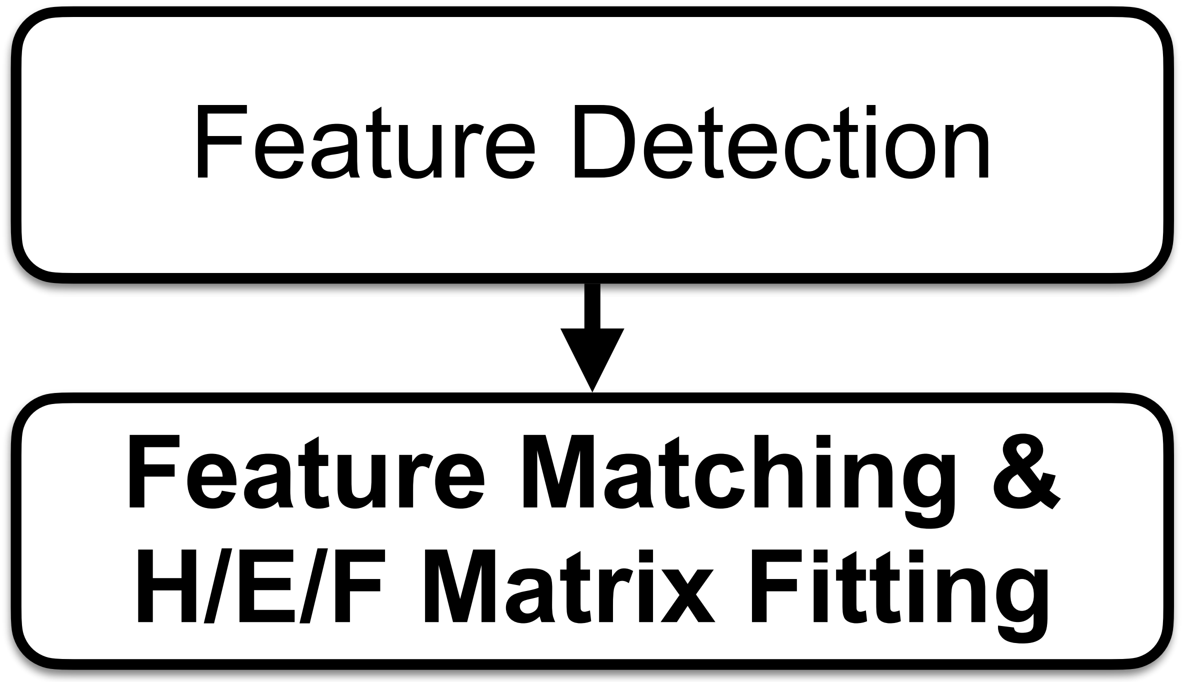
# Sequential / Incremental SfM

**Feature Detection**

Detect interest points and extract descriptors for them, e.g., SIFT features (see lecture 06)

# Sequential / Incremental SfM






- Nearest neighbor search in descriptor space to establish feature matches
- Robust model fitting via **RANSAC**

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model $>\eta$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

CZECH INSTITUTE OF INFORMATICS ROBOTICS AND CYBERNETICS CTU IN PRAGUE

Torsten Sattler

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support **inliers** of model

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >$\eta$

  Estimate model from *n* random data points

  Estimate support **inliers** of model

  If new best model

    update best model, $\eta$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >$\eta$

Estimate model from *n* random data points

Estimate support **inliers** of model

If new best model

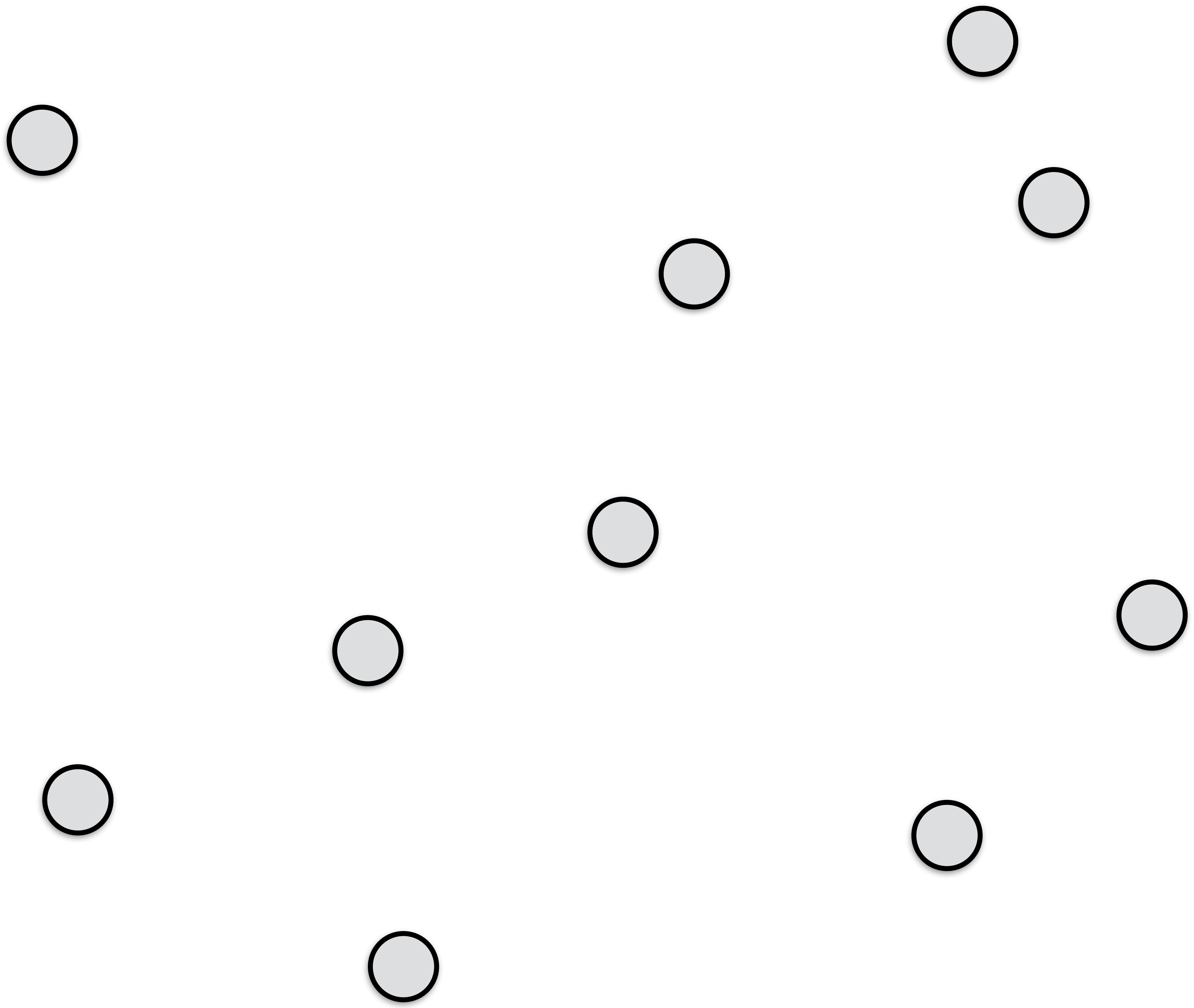
update best model, $\eta$

**Return**: Model with most inliers

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

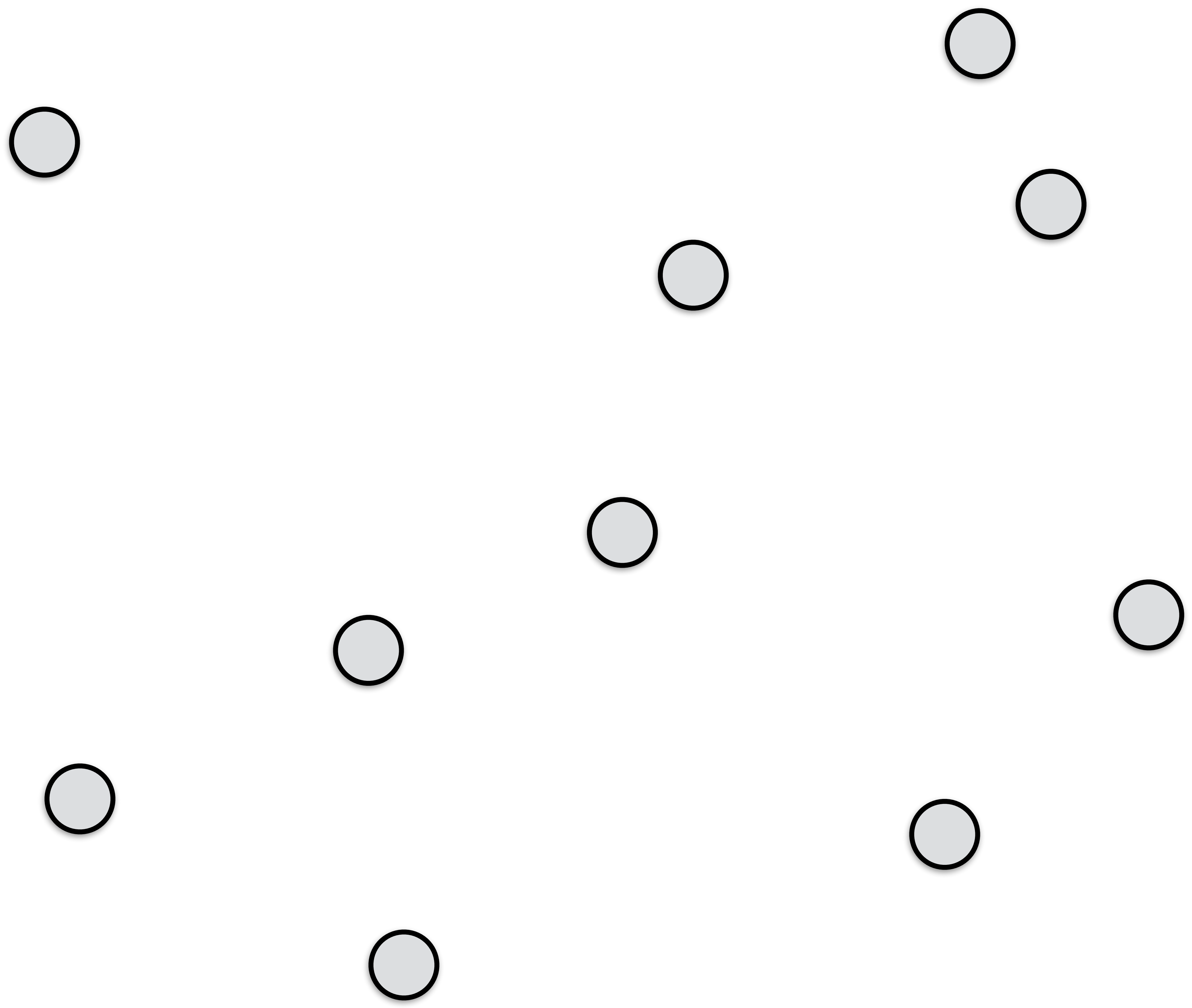
## 2D line fitting example



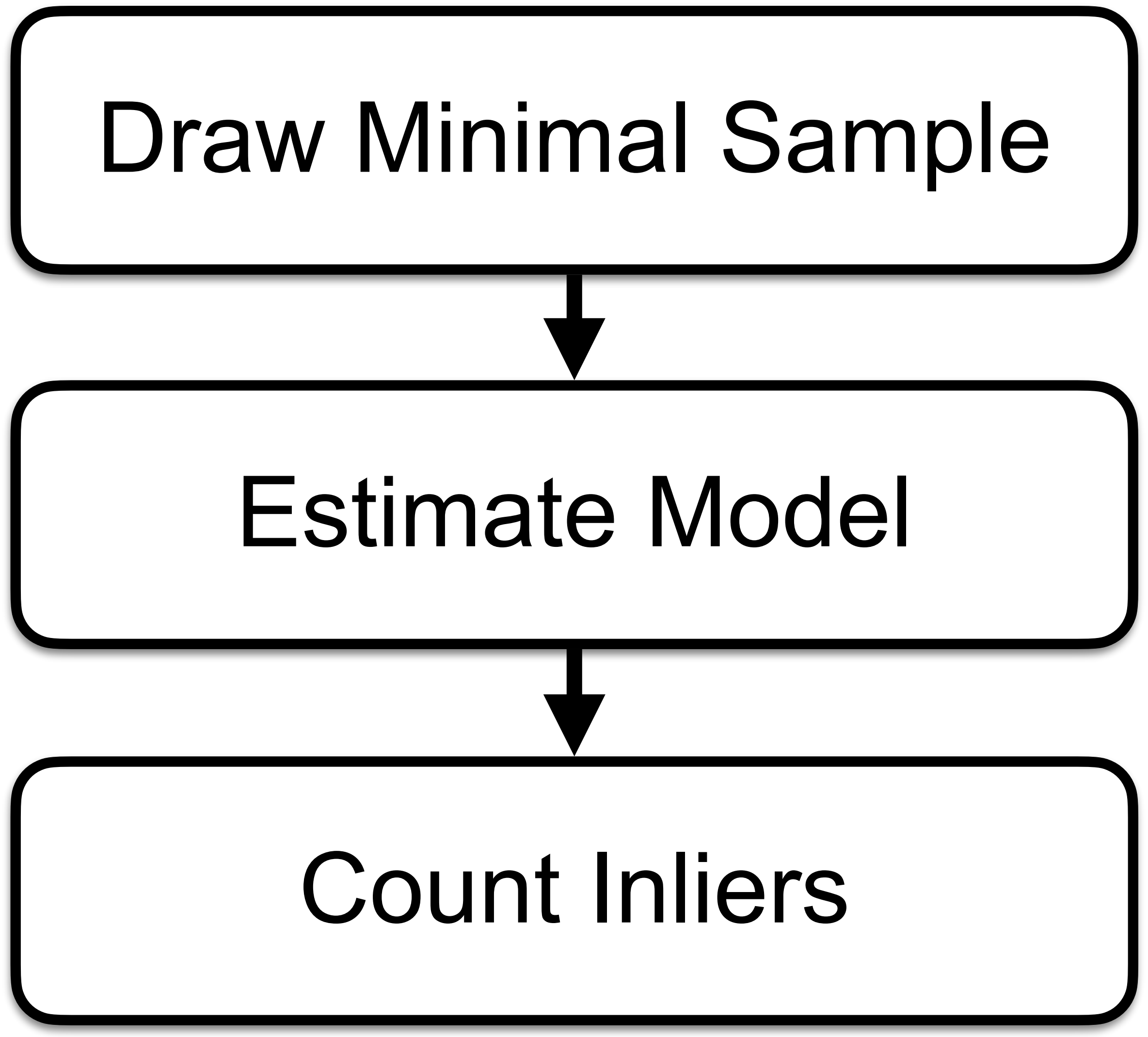
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



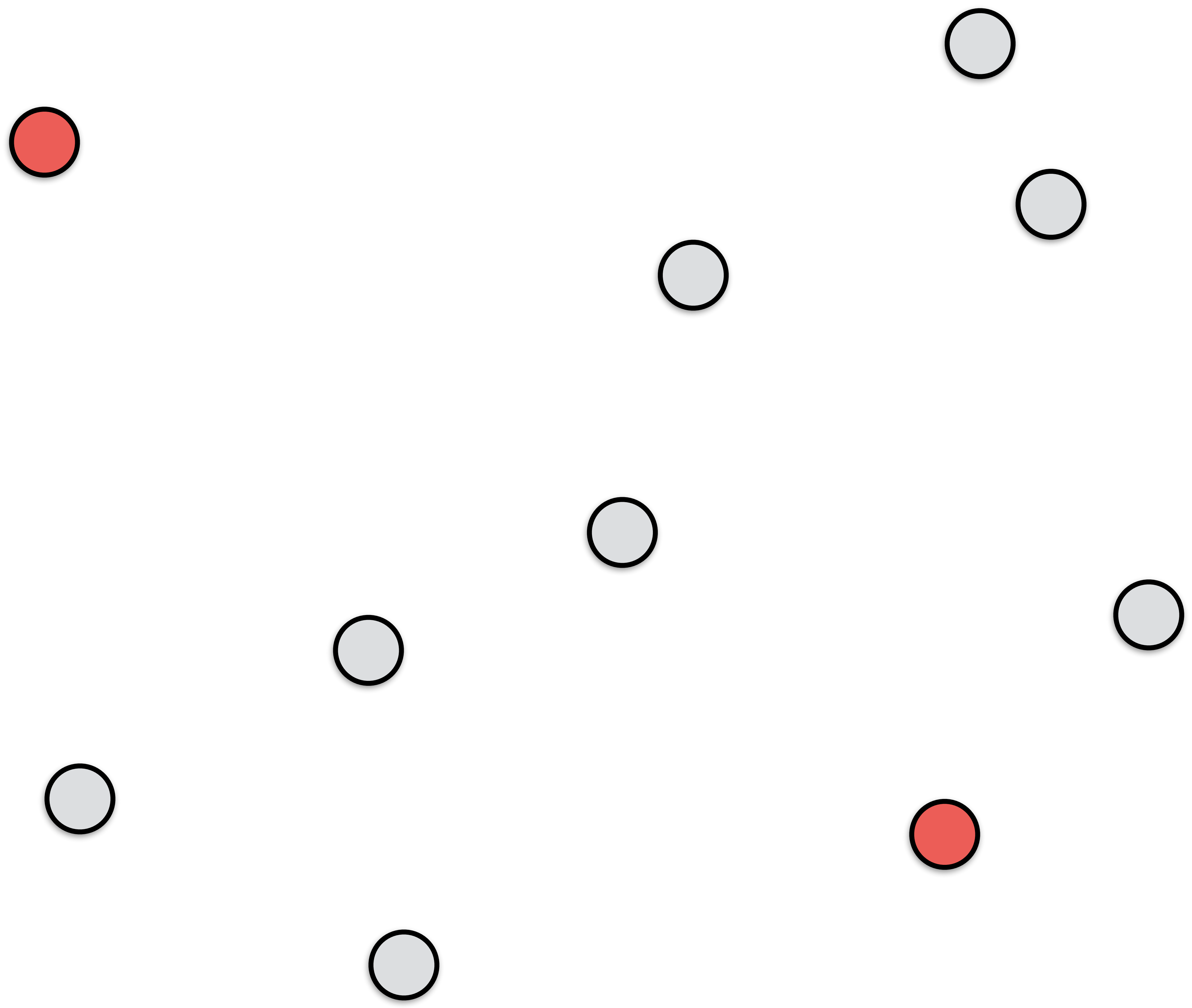
Repeat:



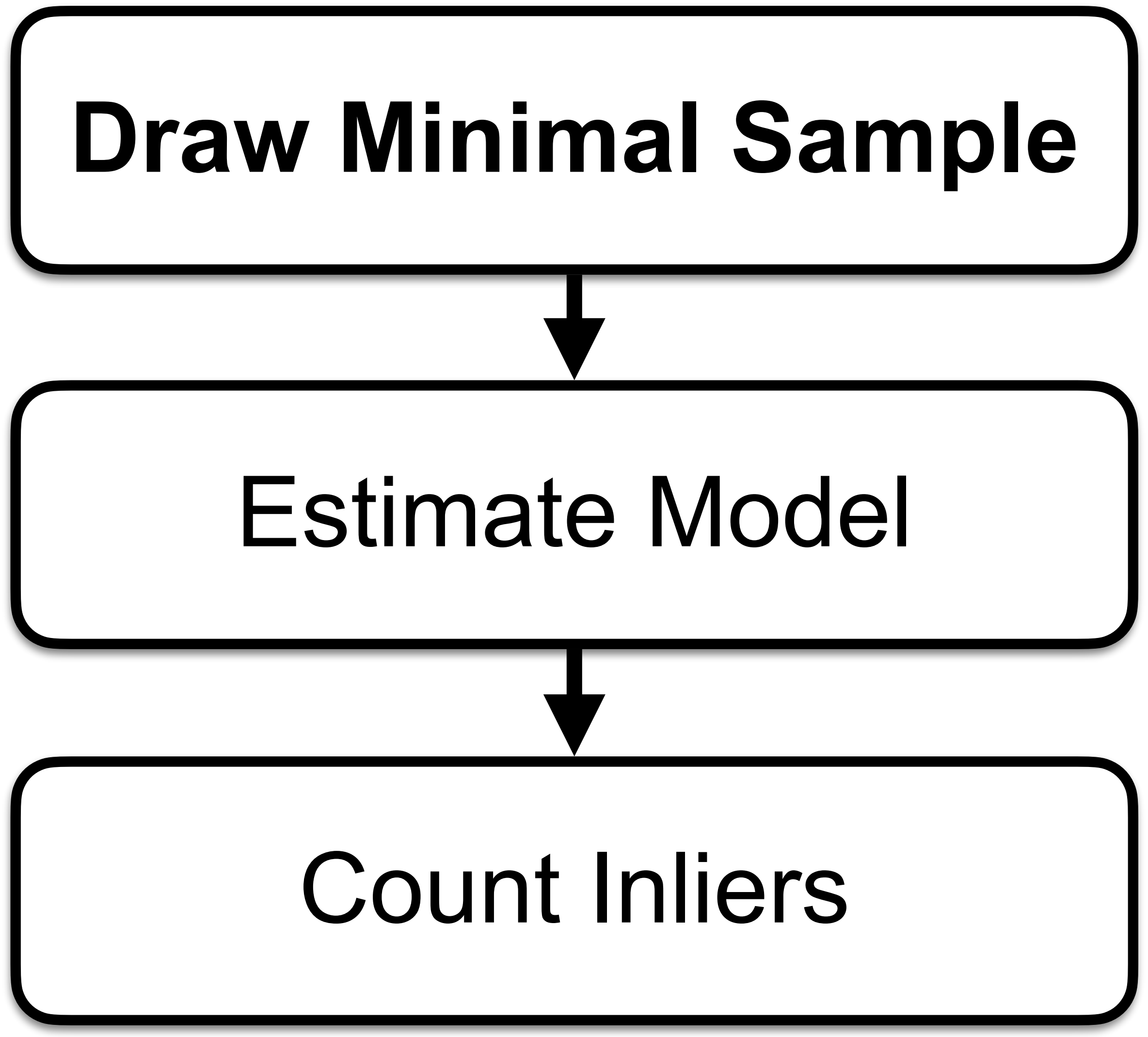

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



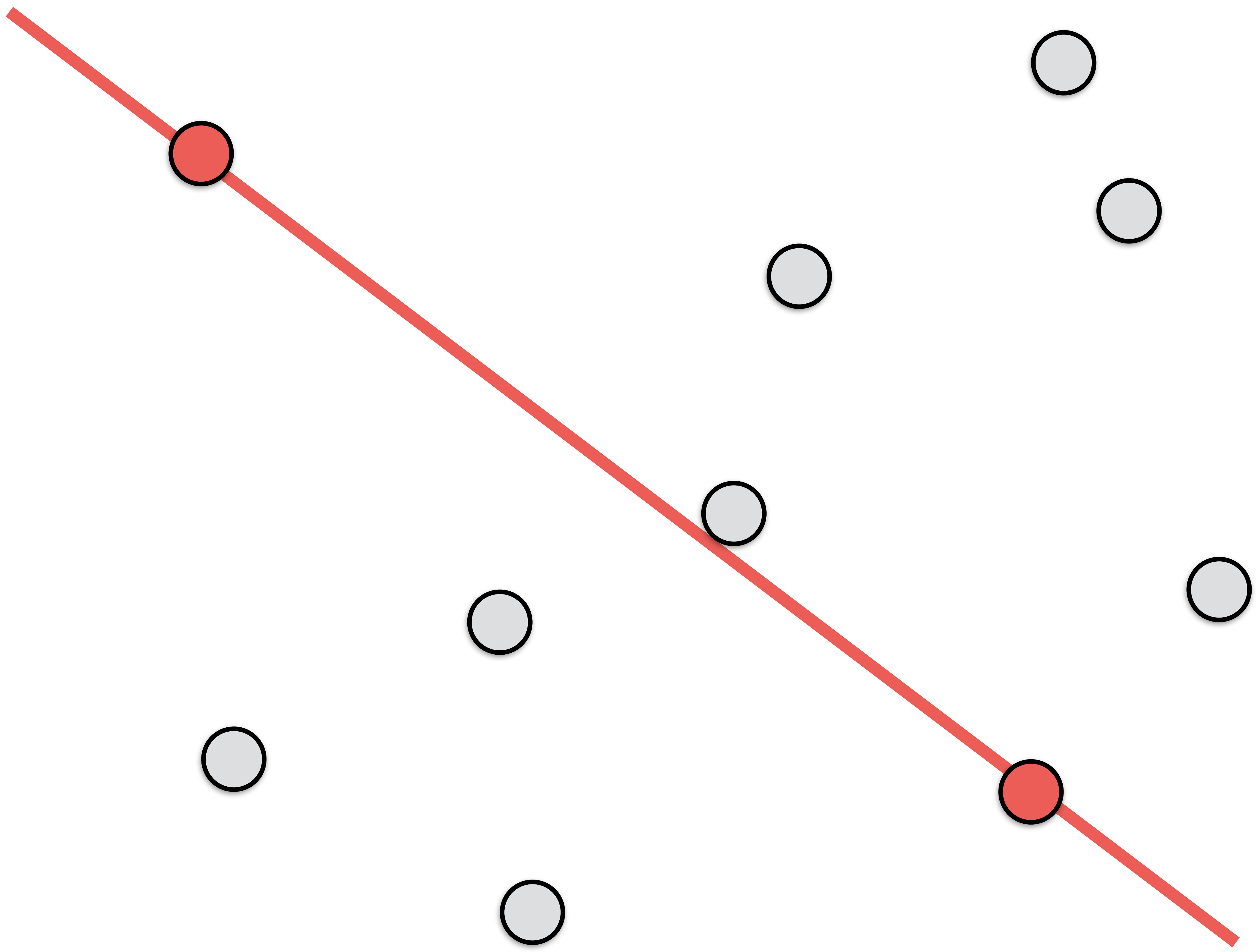
Repeat:



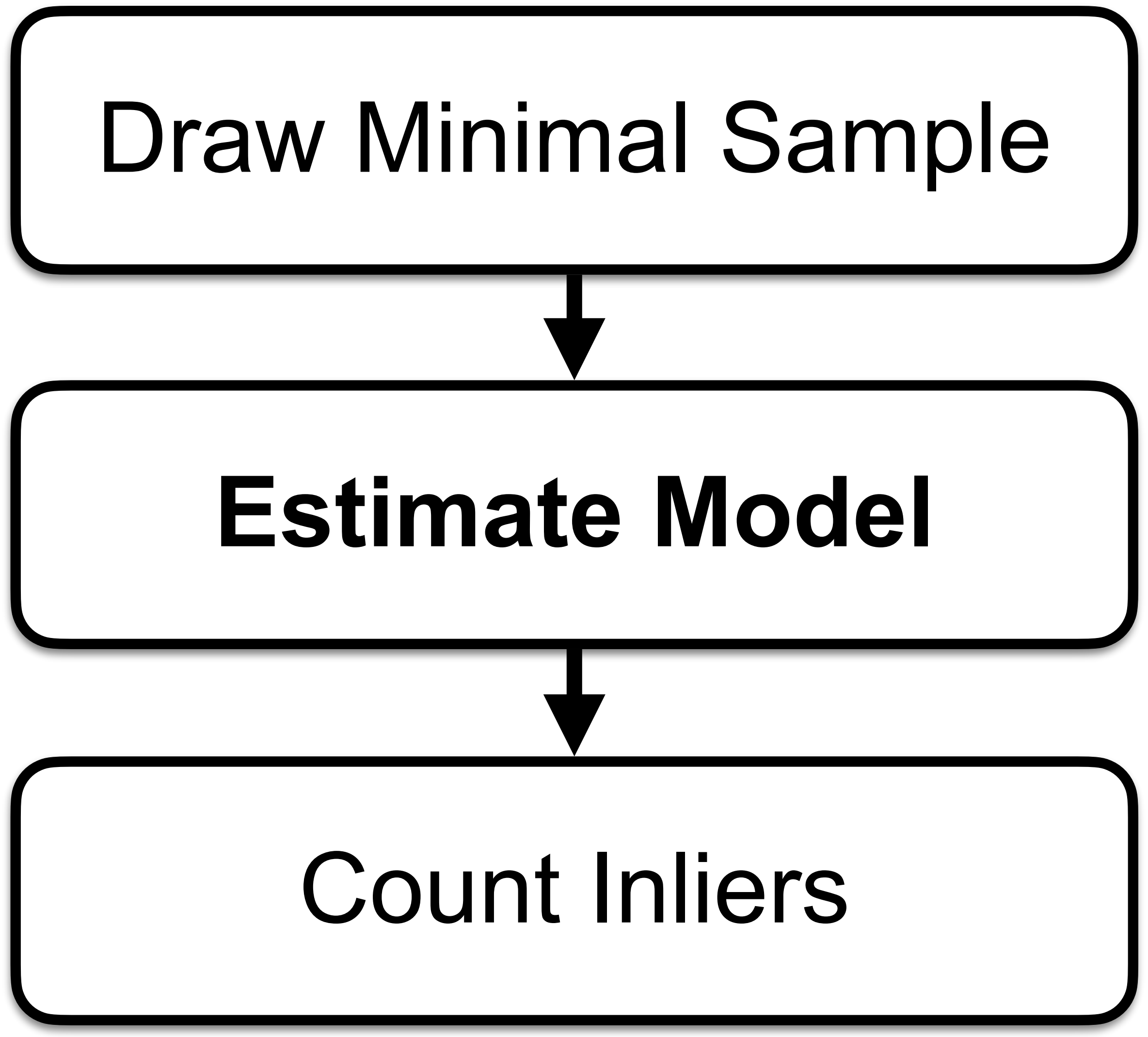

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



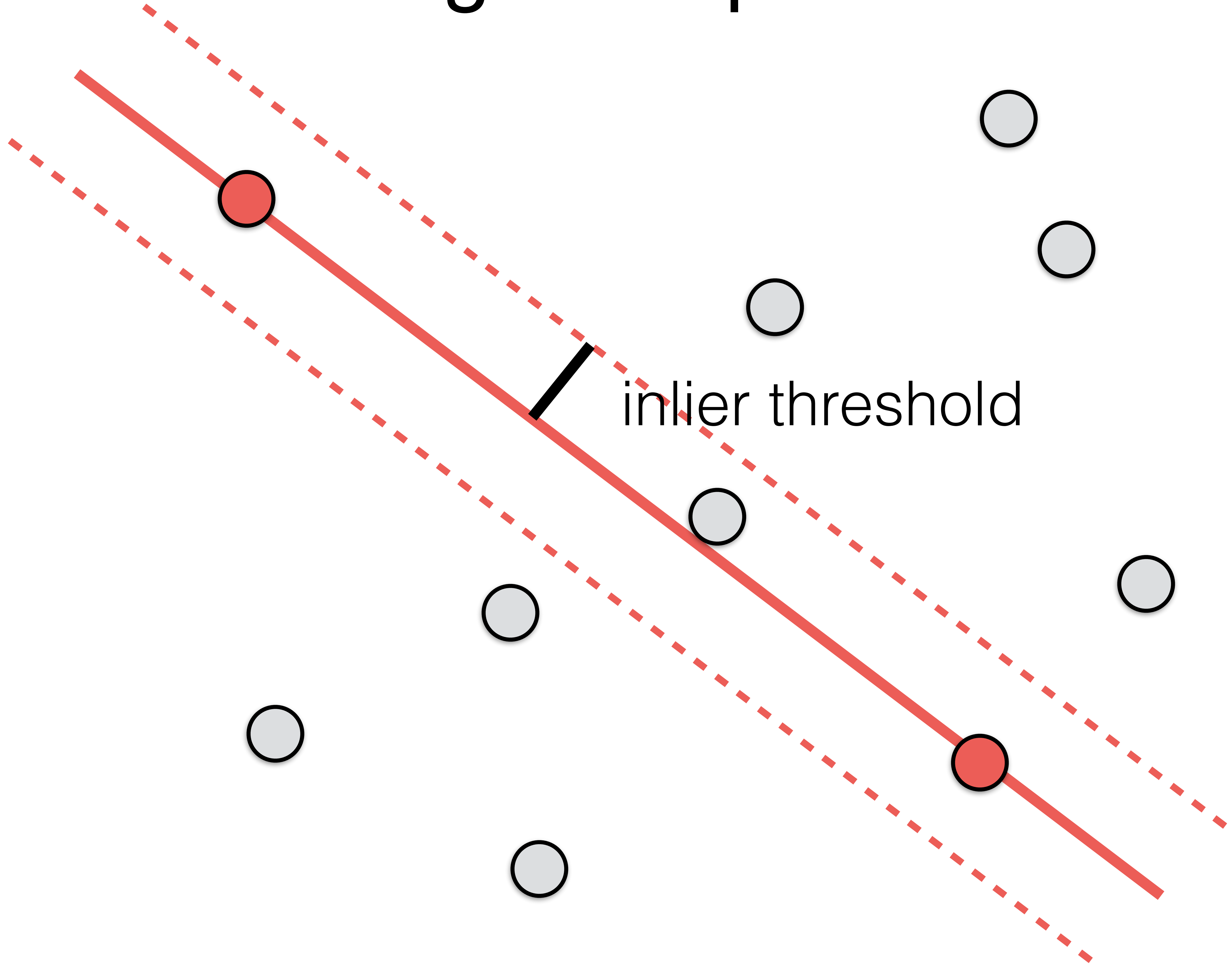
Repeat:




[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

2D line fitting example




Repeat:

Draw Minimal Sample

Estimate Model

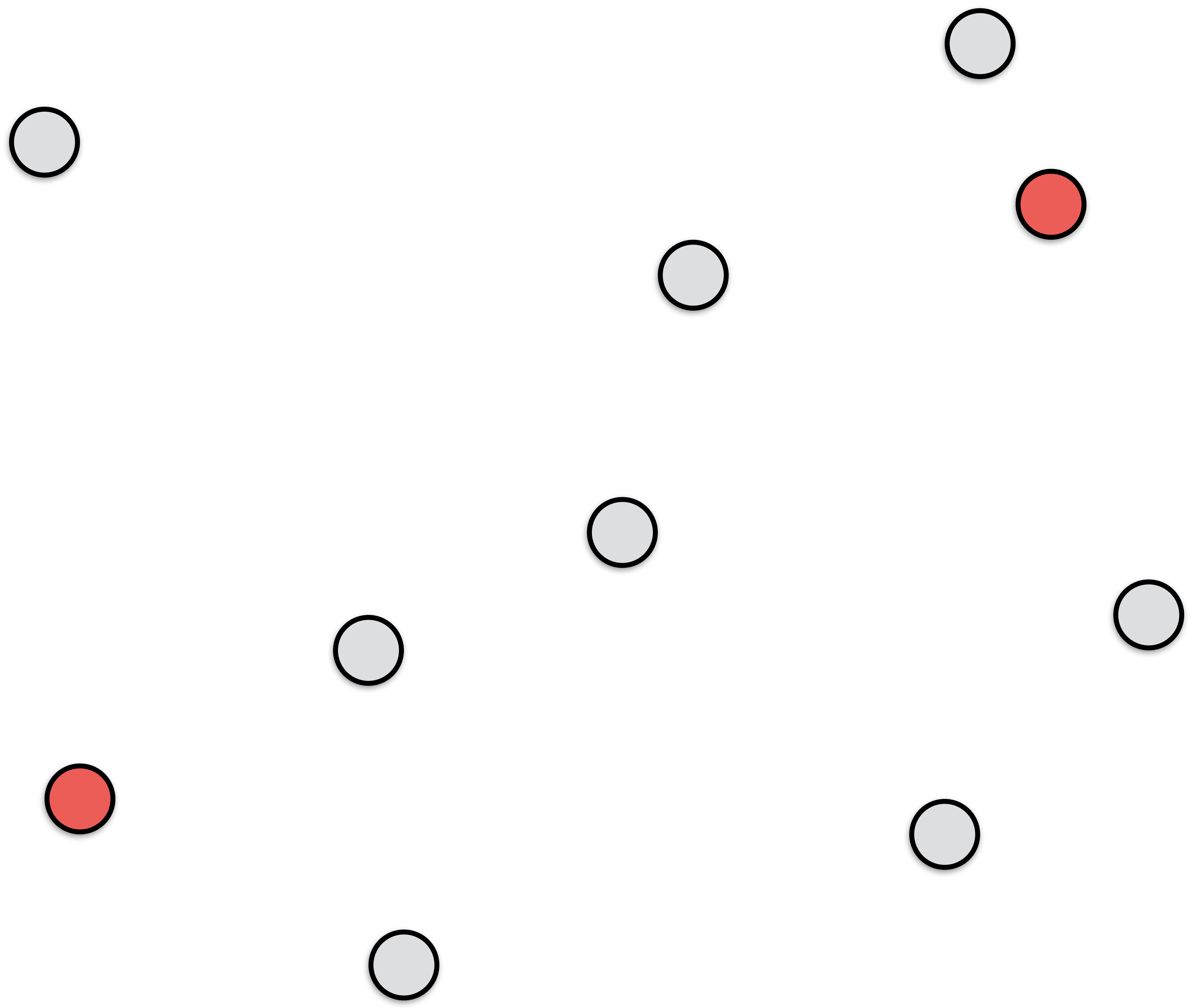
**Count Inliers**

best number of inliers: 3

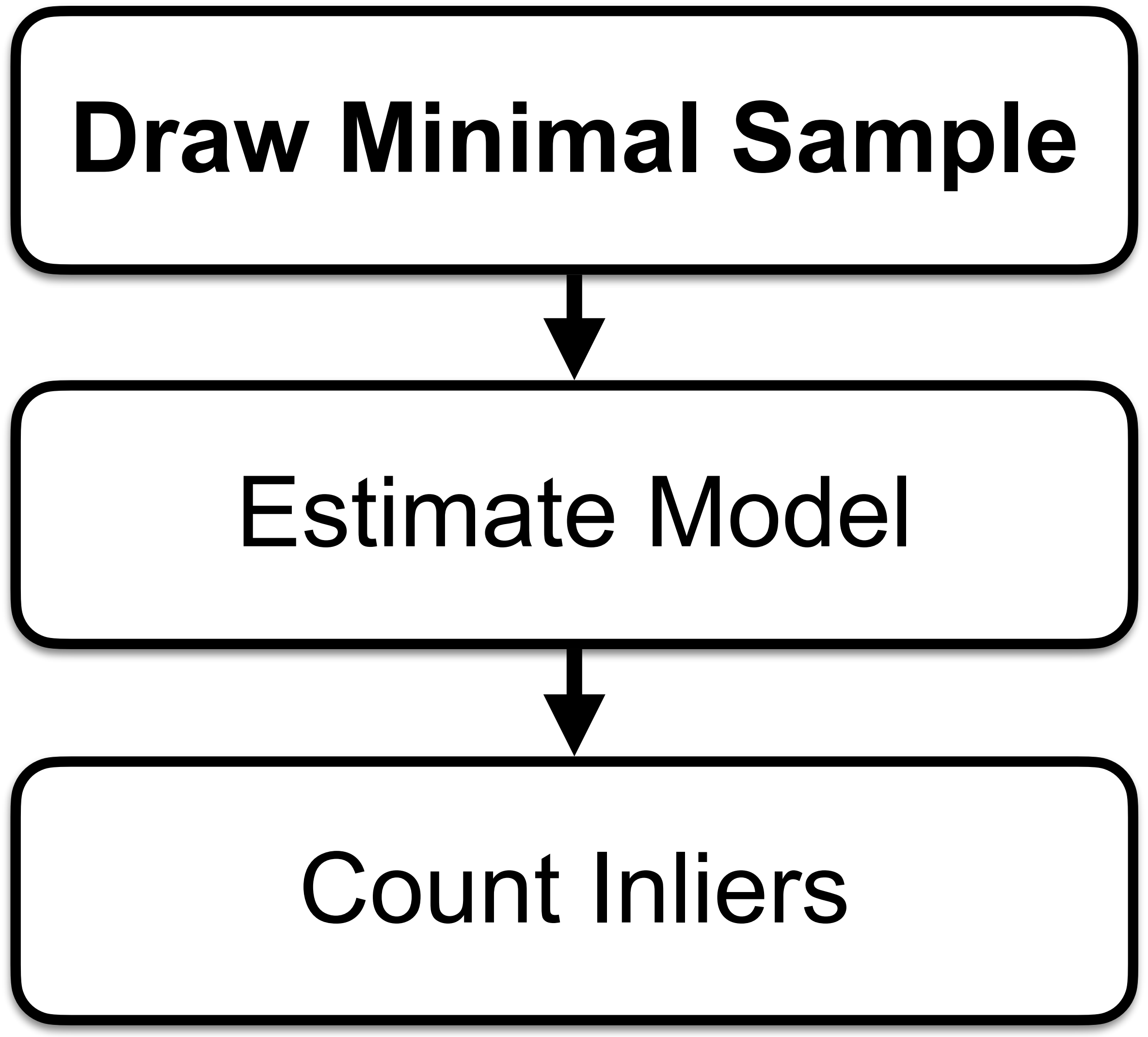
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

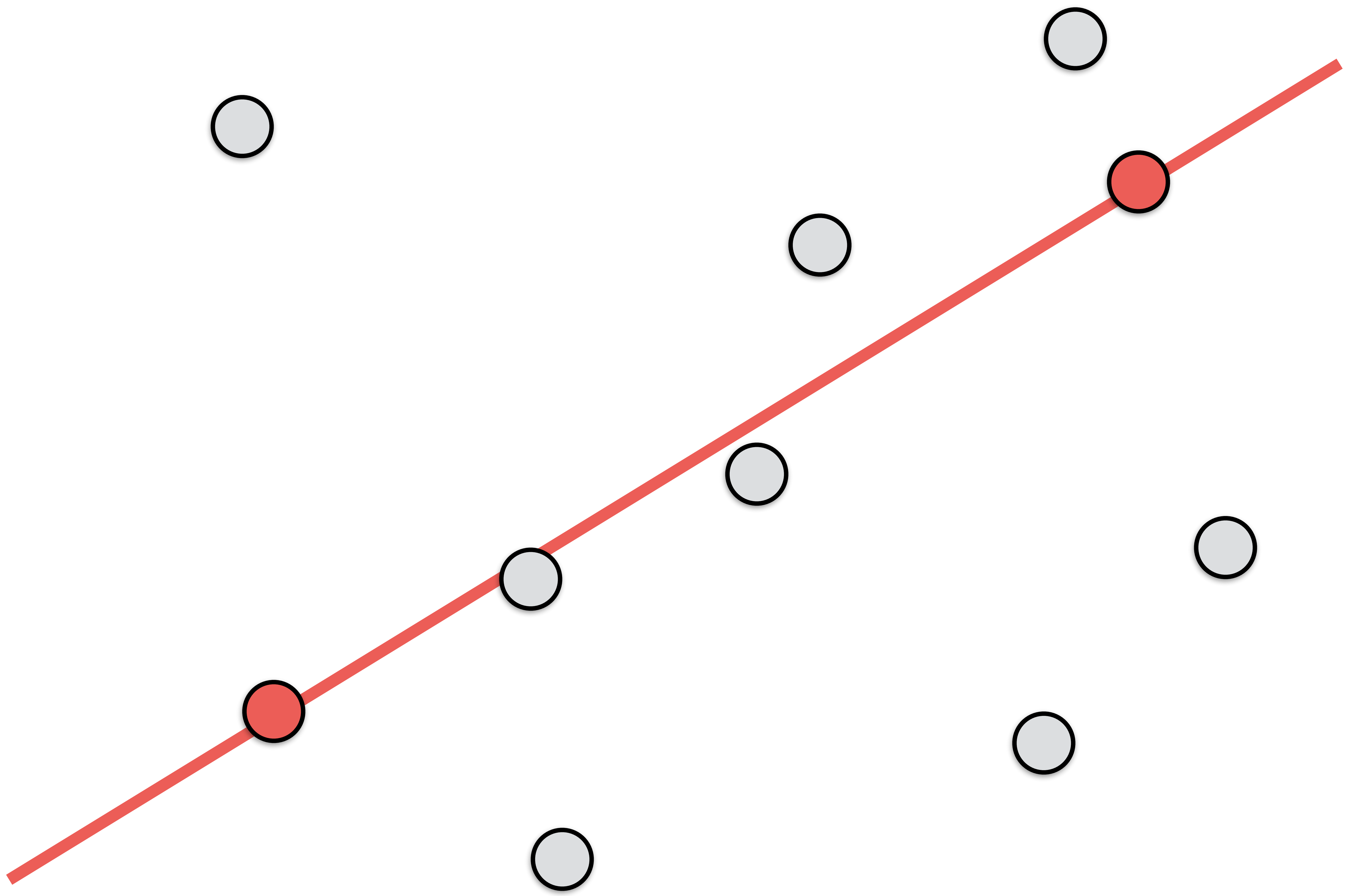



best number of inliers: 3

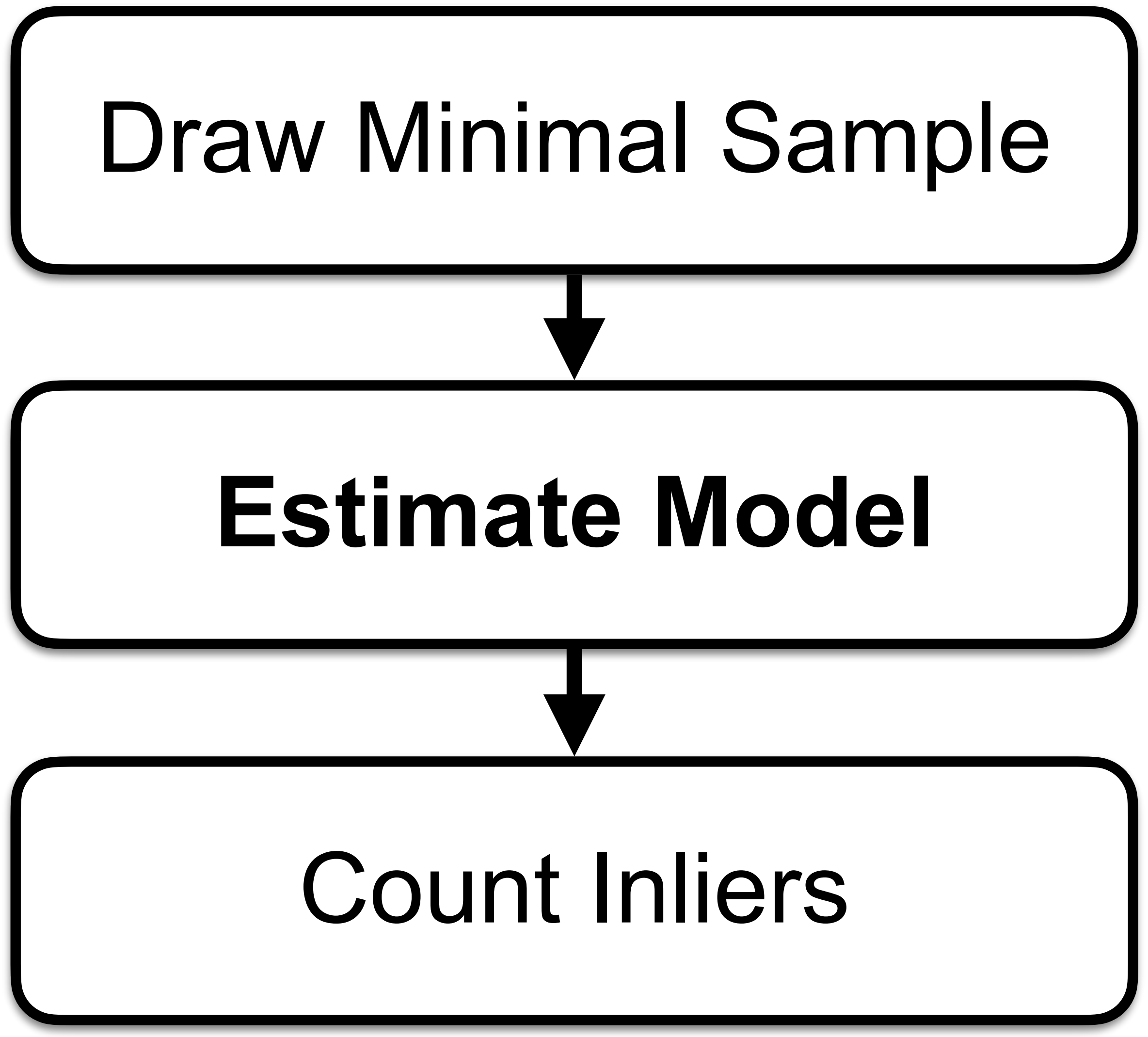
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

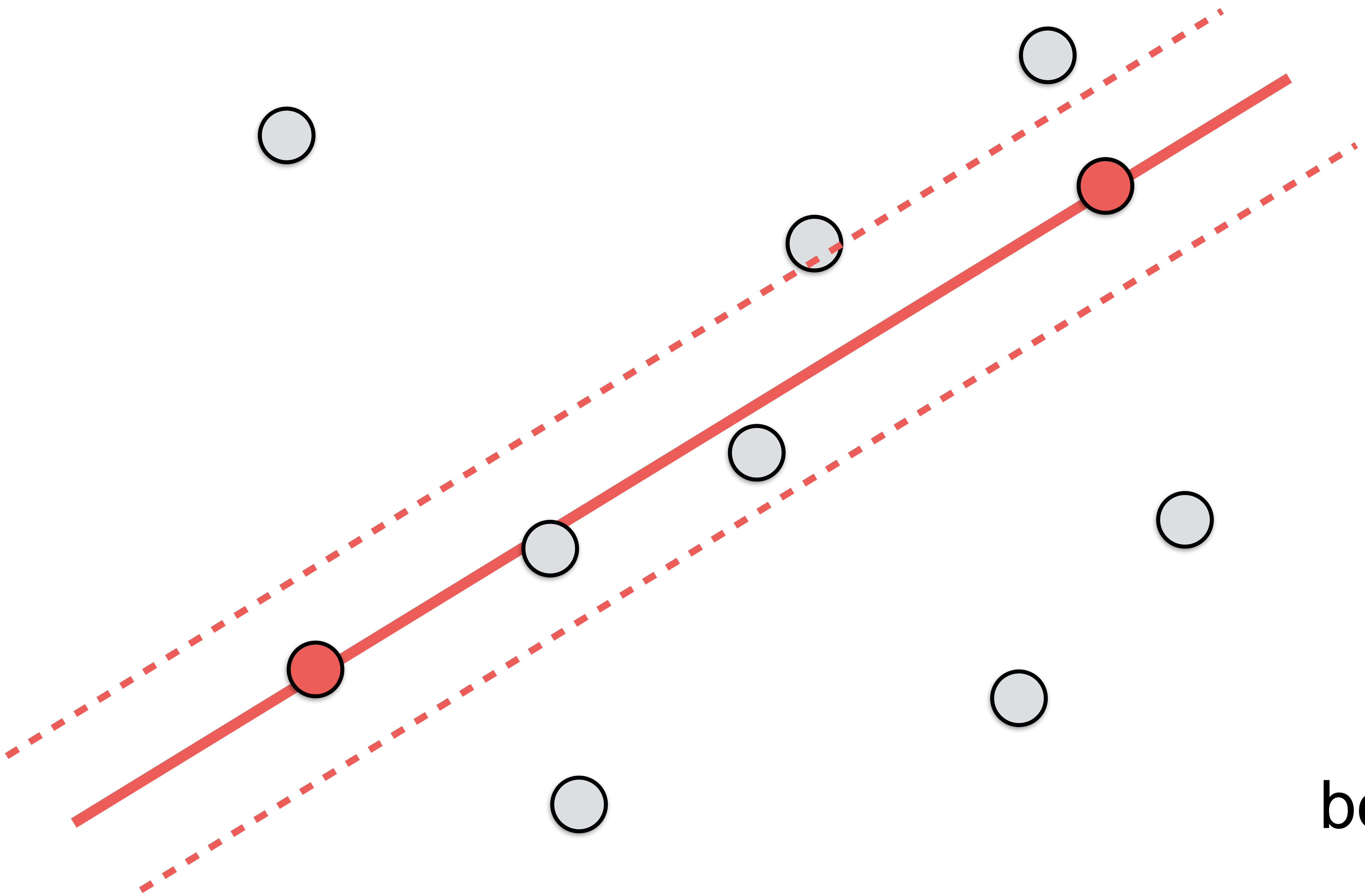



best number of inliers: 3

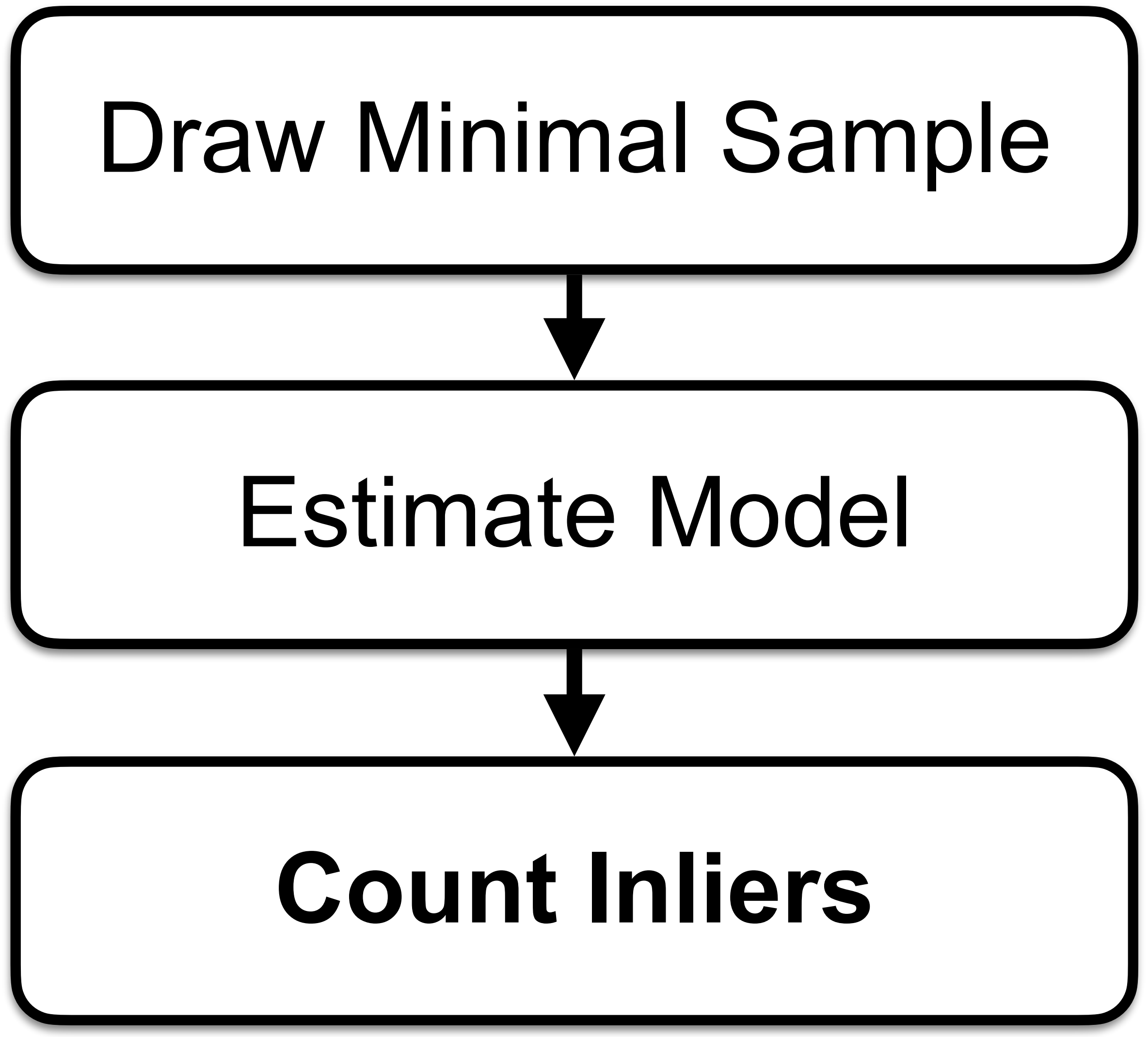
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

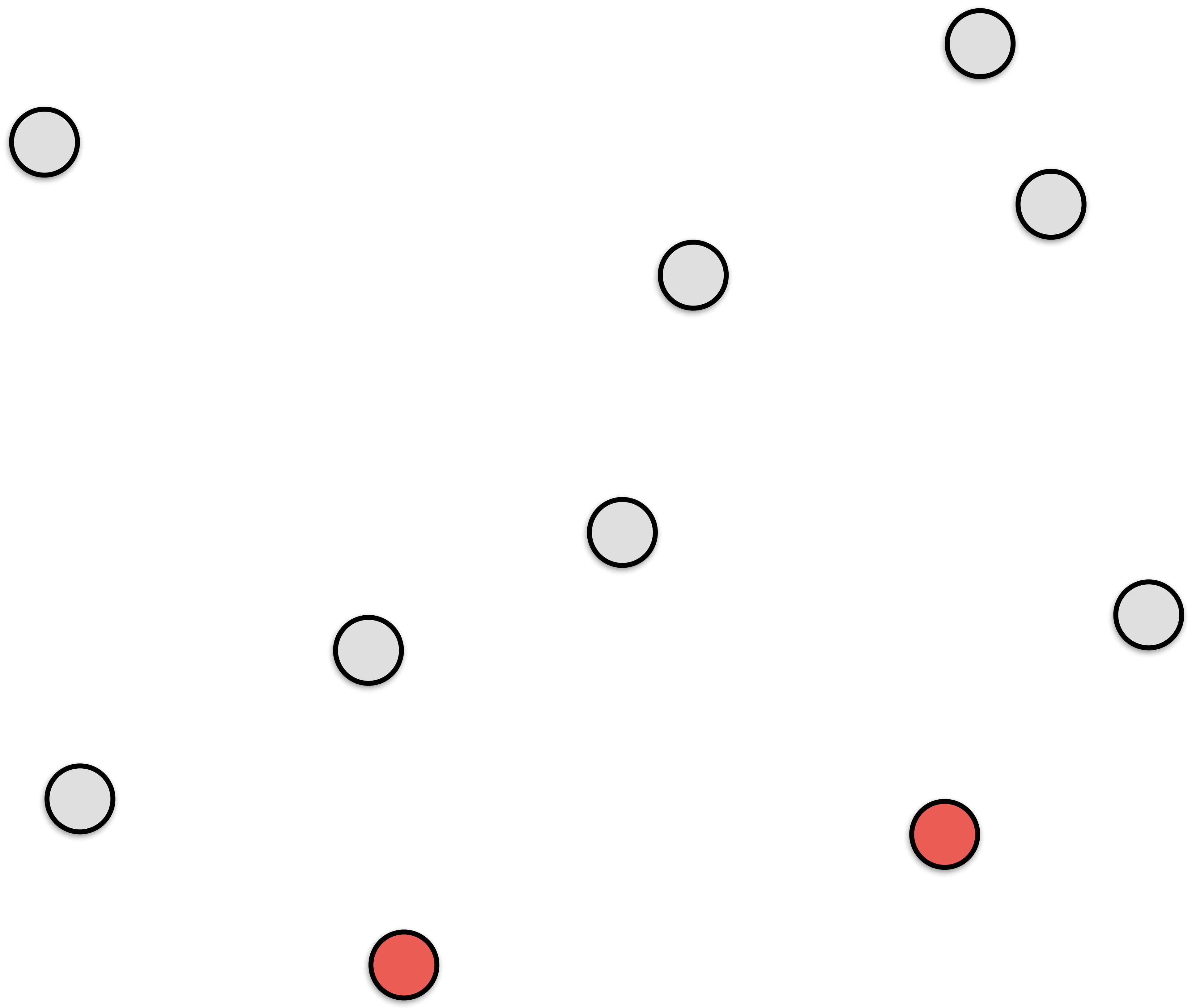



best number of inliers: 4

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

**Draw Minimal Sample**

↓

Estimate Model

↓

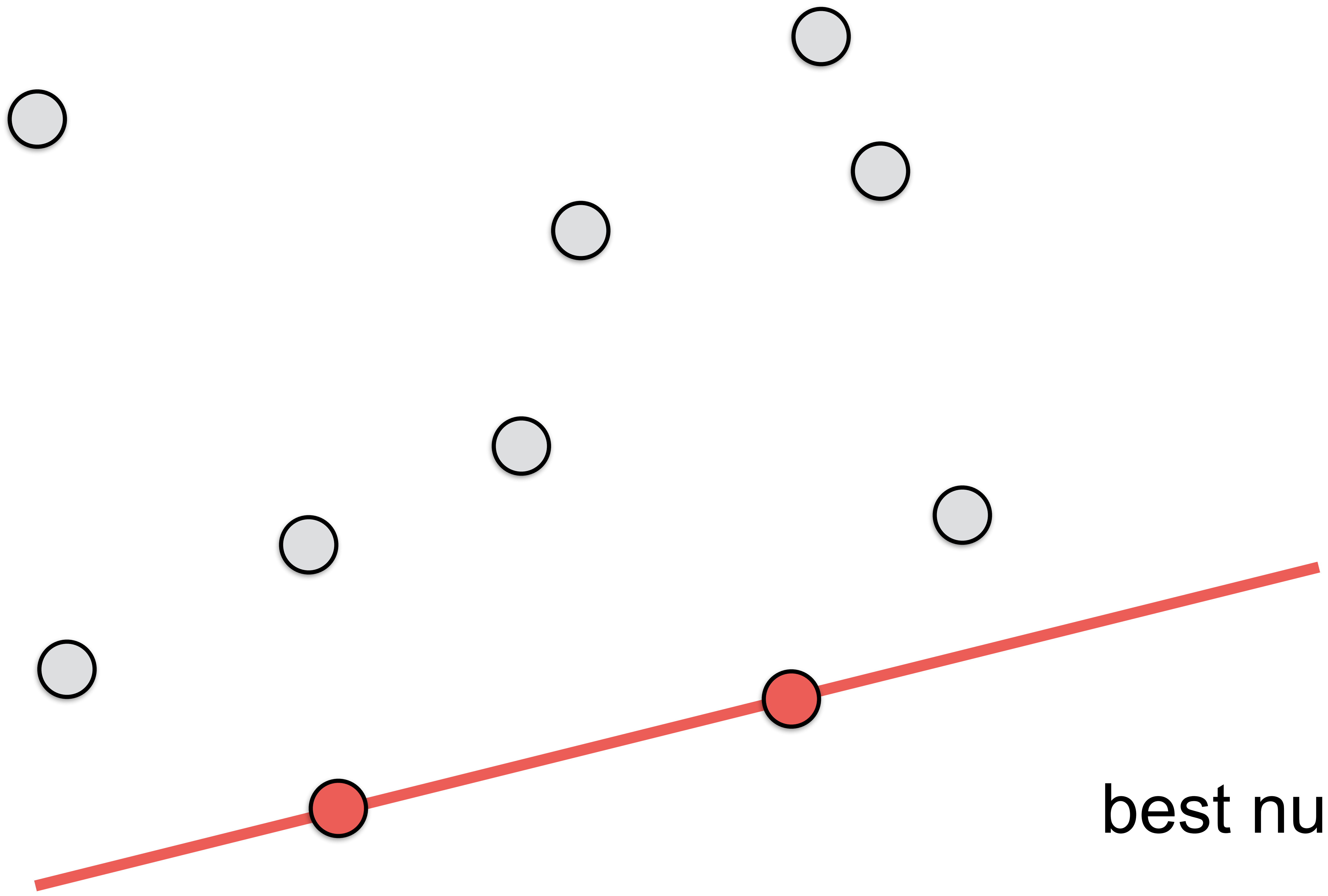
Count Inliers

best number of inliers: 4

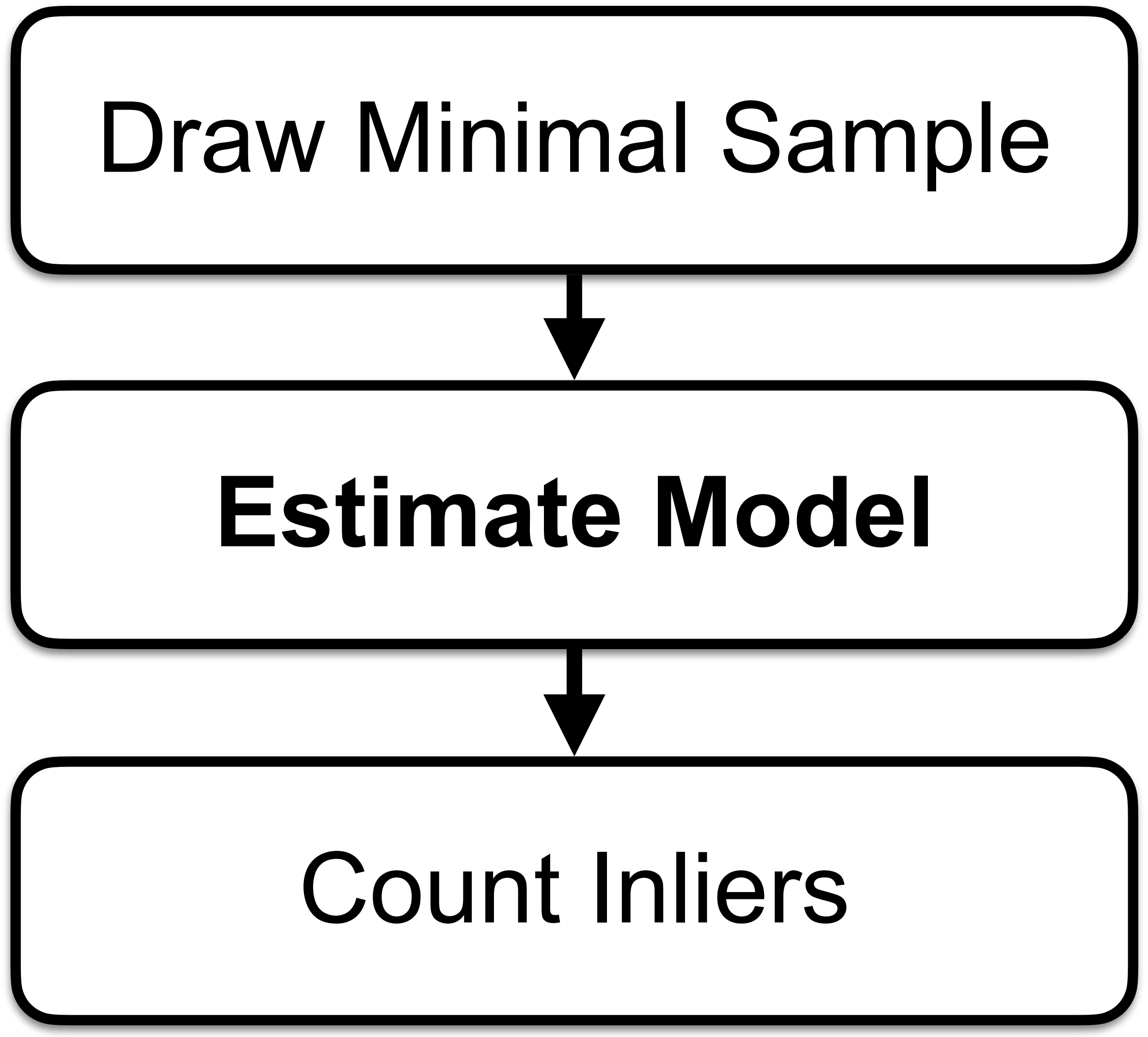
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

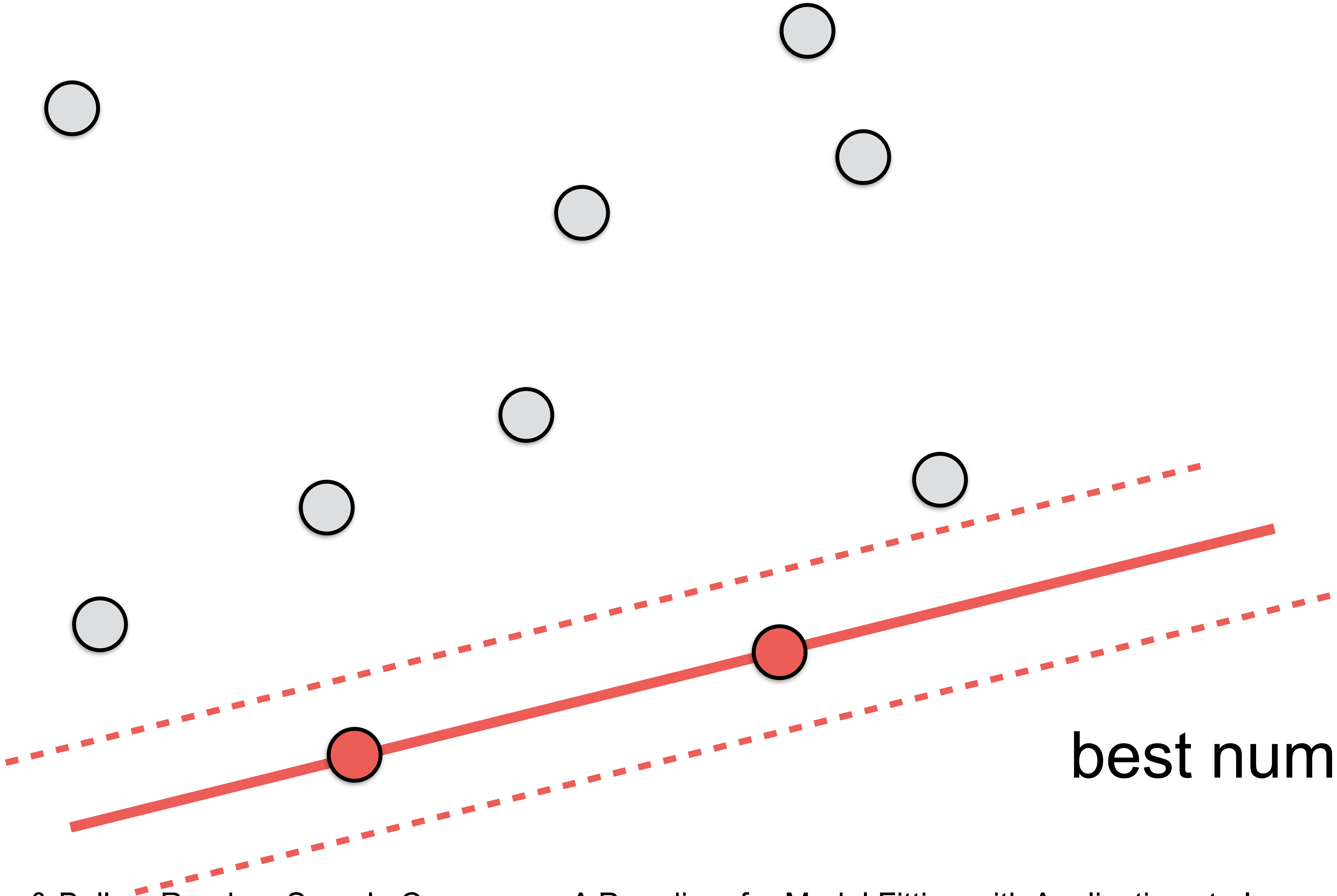



best number of inliers: 4

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

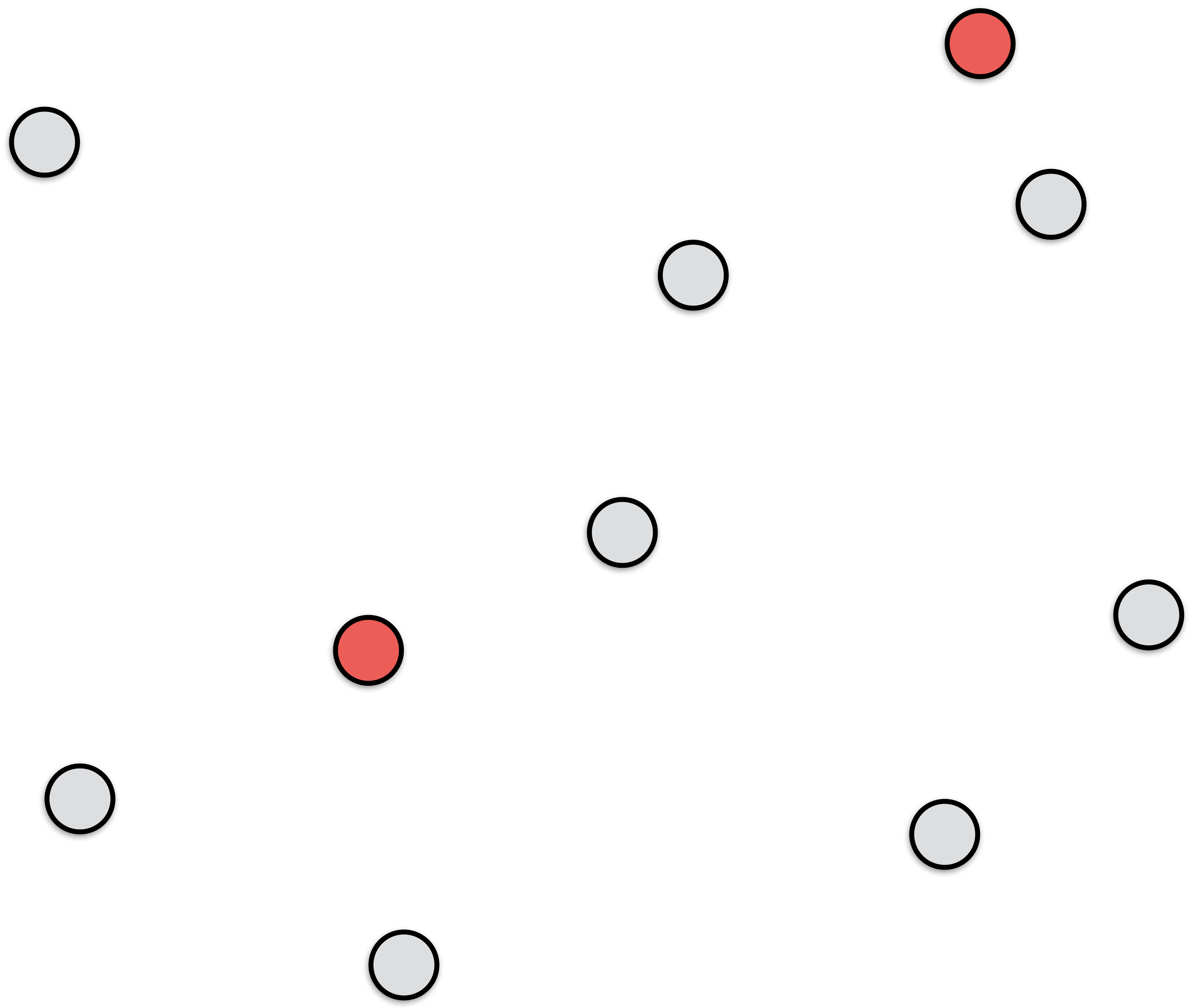
Draw Minimal Sample
↓
Estimate Model
↓
**Count Inliers**

best number of inliers: 4

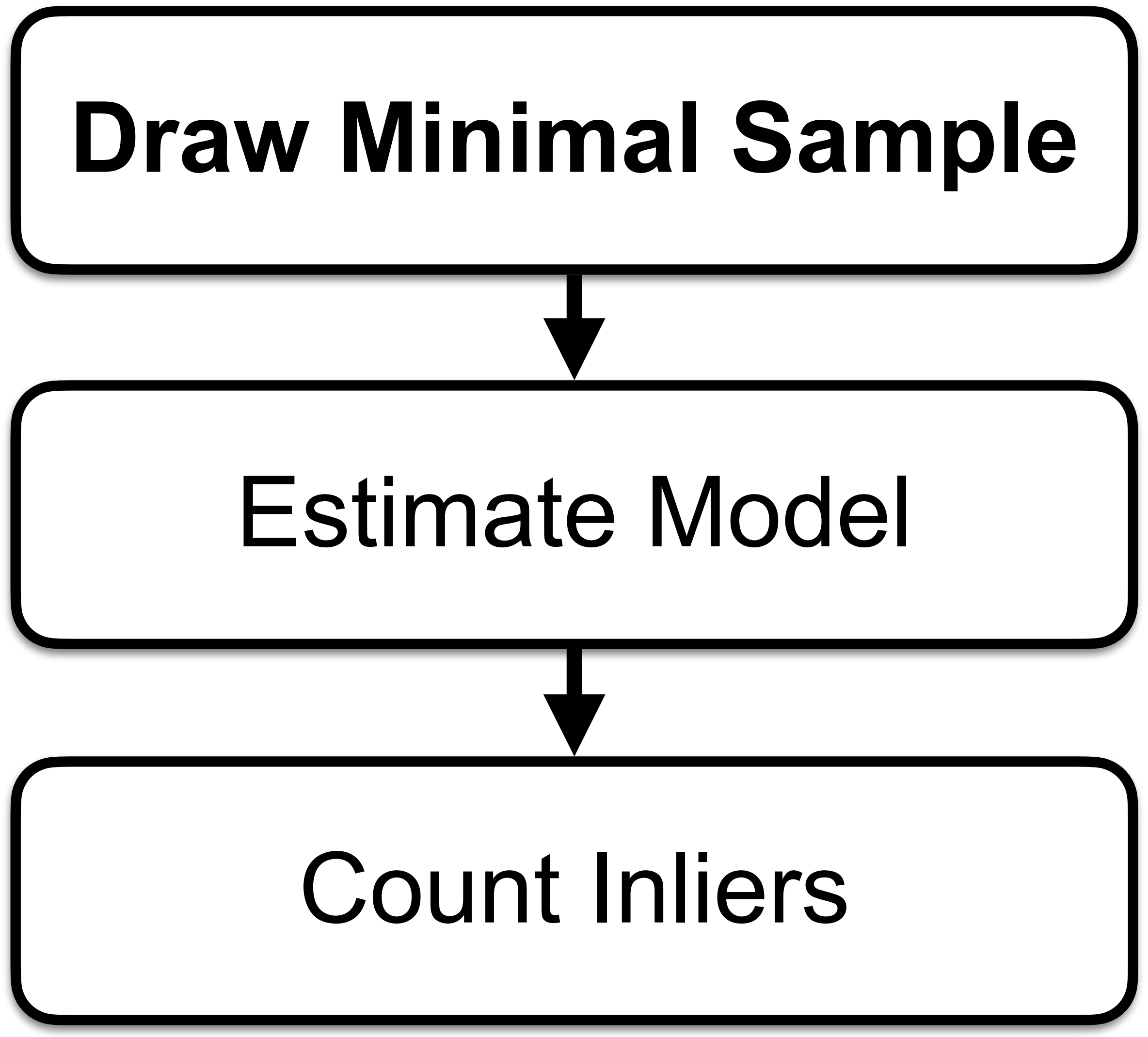
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

## 2D line fitting example



Repeat:

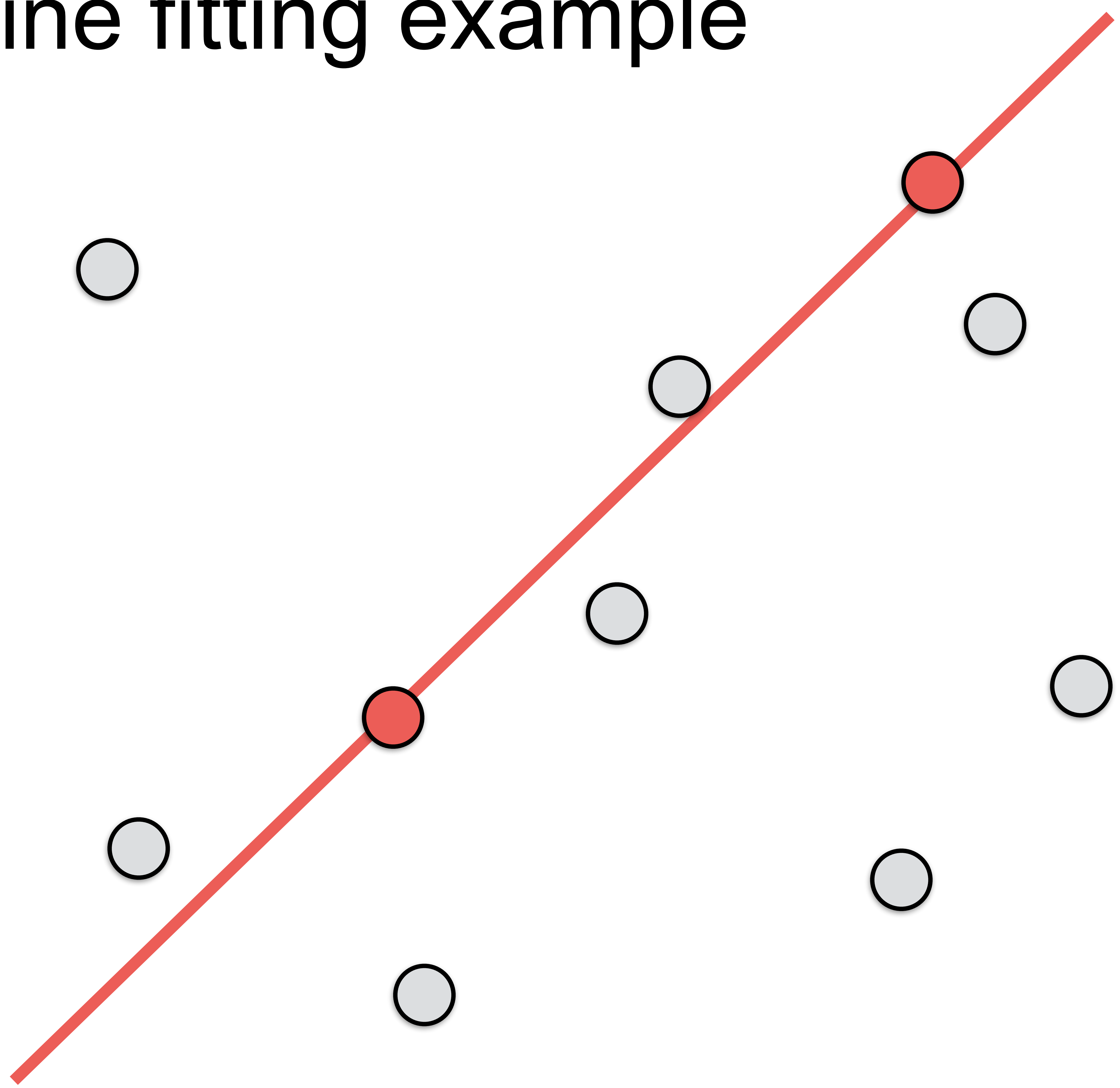



best number of inliers: 4

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

2D line fitting example



Repeat:

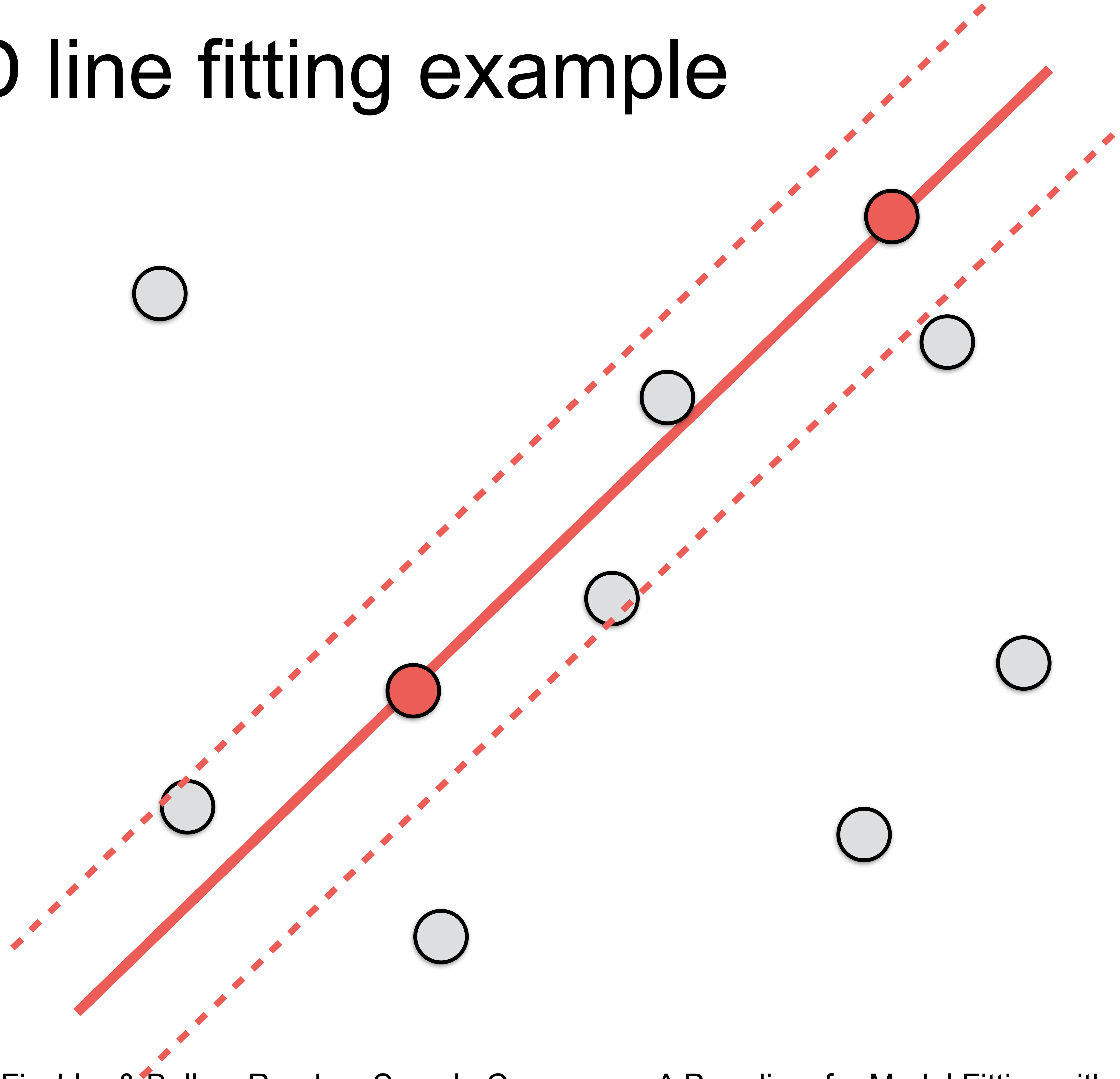
Draw Minimal Sample
↓
**Estimate Model**
↓
Count Inliers

best number of inliers: 4

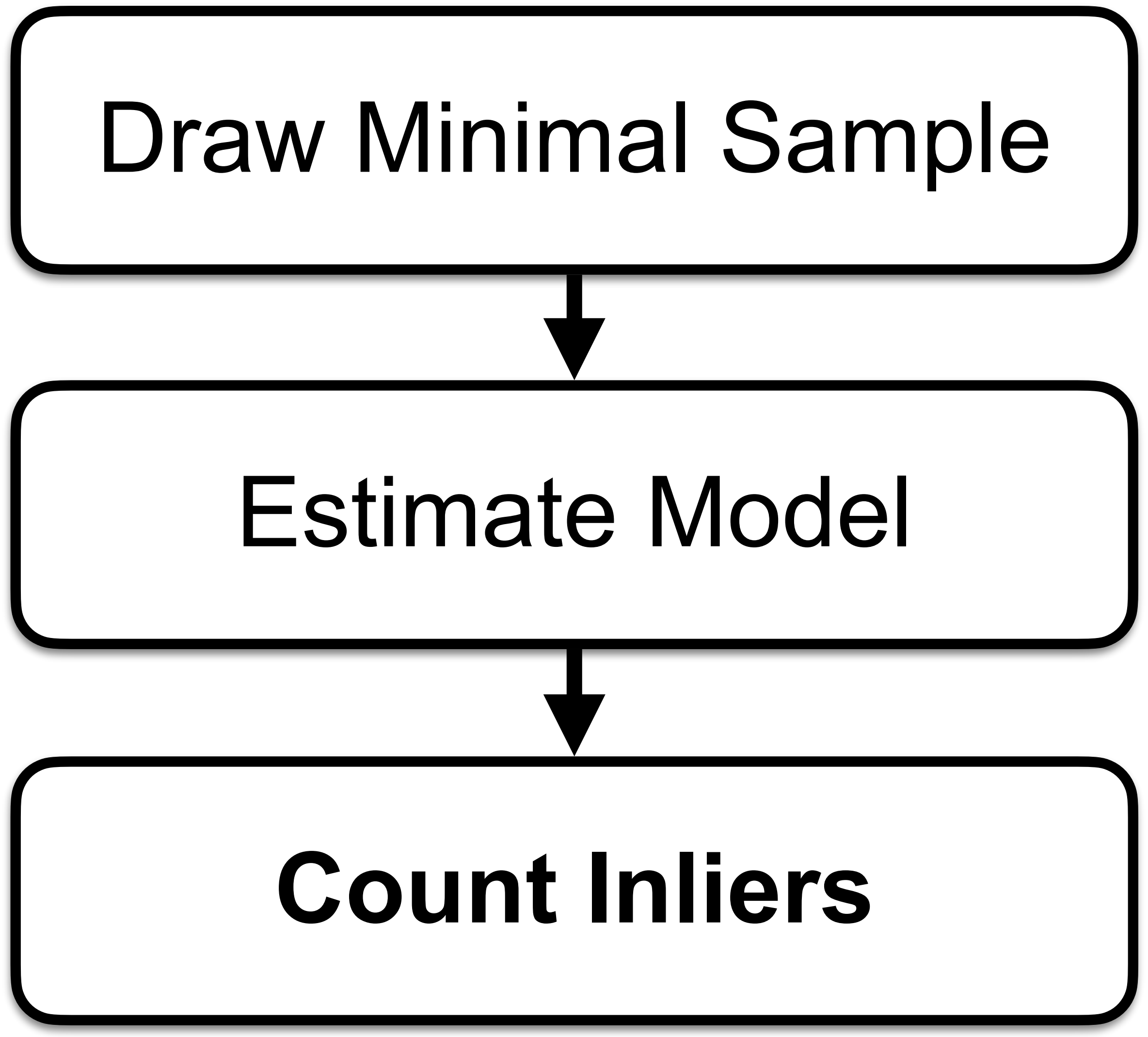
[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC

2D line fitting example



Repeat:




best number of inliers: 5

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Let's assume we know the **inlier ratio** ε (fraction of inliers)

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Let's assume we know the **inlier ratio** $\varepsilon$ (fraction of inliers)
- Probability of picking an inlier randomly: $\varepsilon$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Let’s assume we know the **inlier ratio** $\varepsilon$ (fraction of inliers)
- Probability of picking an inlier randomly: $\varepsilon$
- Probability of picking n inlier randomly: $\varepsilon^n$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Let's assume we know the **inlier ratio** $\varepsilon$ (fraction of inliers)
- Probability of picking an inlier randomly: $\varepsilon$
- Probability of picking n inlier randomly: $\varepsilon^n$
- Probability of non-all inlier sample (≥ 1 outlier): $(1-\varepsilon^n)$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Let's assume we know the **inlier ratio** $\varepsilon$ (fraction of inliers)
- Probability of picking an inlier randomly: $\varepsilon$
- Probability of picking n inlier randomly: $\varepsilon^n$
- Probability of non-all inlier sample (≥ 1 outlier): $(1-\varepsilon^n)$
- Probability of not picking all-inlier sample in k iterations: $(1-\varepsilon^n)^k$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k<\eta$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k < \eta$
- In practice: Compute maximum number iterations $k_{max}$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k < \eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k < \eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

$\Leftrightarrow k_{max} \ln(1-\varepsilon^n) = \ln(\eta)$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k < \eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

$\Leftrightarrow k_{max} \ln(1-\varepsilon^n) = \ln(\eta)$

$\Leftrightarrow k_{max} = \ln(\eta) / \ln(1-\varepsilon^n)$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k<\eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

$\Leftrightarrow k_{max} \ln(1-\varepsilon^n) = \ln(\eta)$

$\Leftrightarrow k_{max} = \ln(\eta) / \ln(1-\varepsilon^n)$

  - Note: $k_{max}(\varepsilon) > k_{max}(\varepsilon')$ if $\varepsilon < \varepsilon'$

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k<\eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

$\Leftrightarrow k_{max} \ln(1-\varepsilon^n) = \ln(\eta)$

$\Leftrightarrow k_{max} = \ln(\eta) / \ln(1-\varepsilon^n)$

  - Note: $k_{max}(\varepsilon) > k_{max}(\varepsilon')$ if $\varepsilon < \varepsilon'$
- How do we know inlier ratio $\varepsilon$?

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# RANSAC - Termination Criterion

- Terminate if $(1-\varepsilon^n)^k < \eta$
- In practice: Compute maximum number iterations $k_{max}$
  - Find $k_{max}$ such that $(1-\varepsilon^n)^{kmax} = \eta$

$\Leftrightarrow k_{max} \ln(1-\varepsilon^n) = \ln(\eta)$

$\Leftrightarrow k_{max} = \ln(\eta) / \ln(1-\varepsilon^n)$

  - Note: $k_{max}(\varepsilon) > k_{max}(\varepsilon')$ if $\varepsilon < \varepsilon'$
- How do we know inlier ratio $\varepsilon$?
- In practice more than $k_{max}(\varepsilon)$ steps necessary as not every all-inlier sample leads to best model (due to, e.g., noise, degeneracies, etc.)

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support (**#inliers** / **robust cost func.**) of model

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support (**#inliers** / **robust cost func.**) of model

[Chum, Matas, Optimal Randomized RANSAC. PAMI 2008]

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support (**#inliers** / **robust cost func.**) of model

If new best model

**Perform Local Optimization (LO)**

[Lebeda, Matas, Chum, Fixing the Locally Optimized RANSAC. BMVC 2012] [code]

[Chum, Matas, Optimal Randomized RANSAC. PAMI 2008]

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support (**#inliers** / **robust cost func.**) of model

If new best model

**Perform Local Optimization (LO)**

update best model, η

[Chum, Matas, Optimal Randomized RANSAC. PAMI 2008]

[Lebeda, Matas, Chum, Fixing the Locally Optimized RANSAC. BMVC 2012] [code]

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

# Model Fitting With RANdom SAmple Consensus (RANSAC)

**While** probability of missing correct model >η

Estimate model from *n* random data points

Estimate support (**#inliers** / **robust cost func.**) of model

If new best model

**Perform Local Optimization (LO)**

[Lebeda, Matas, Chum, Fixing the Locally Optimized RANSAC. BMVC 2012] [code]

update best model, η

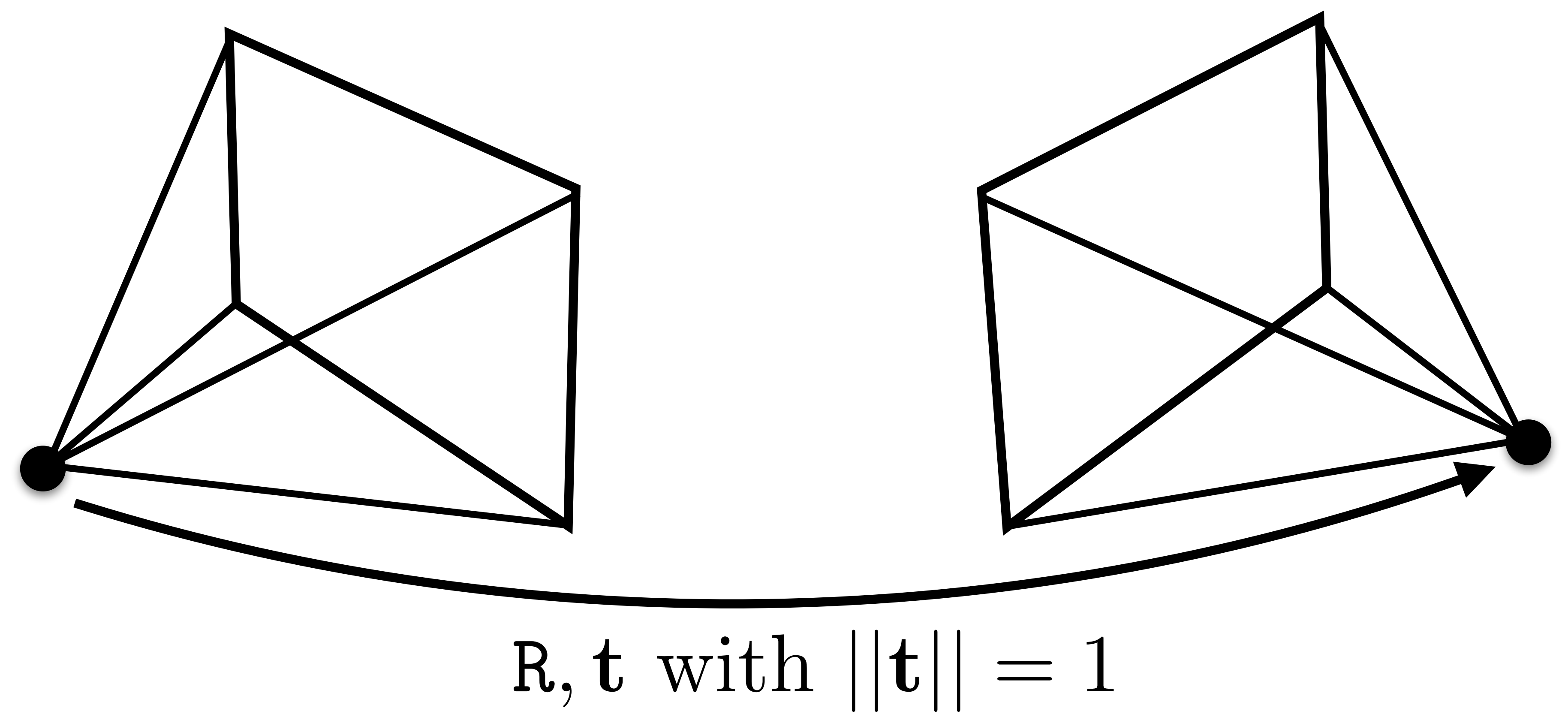
**Return**: Model with most inliers / lowest cost

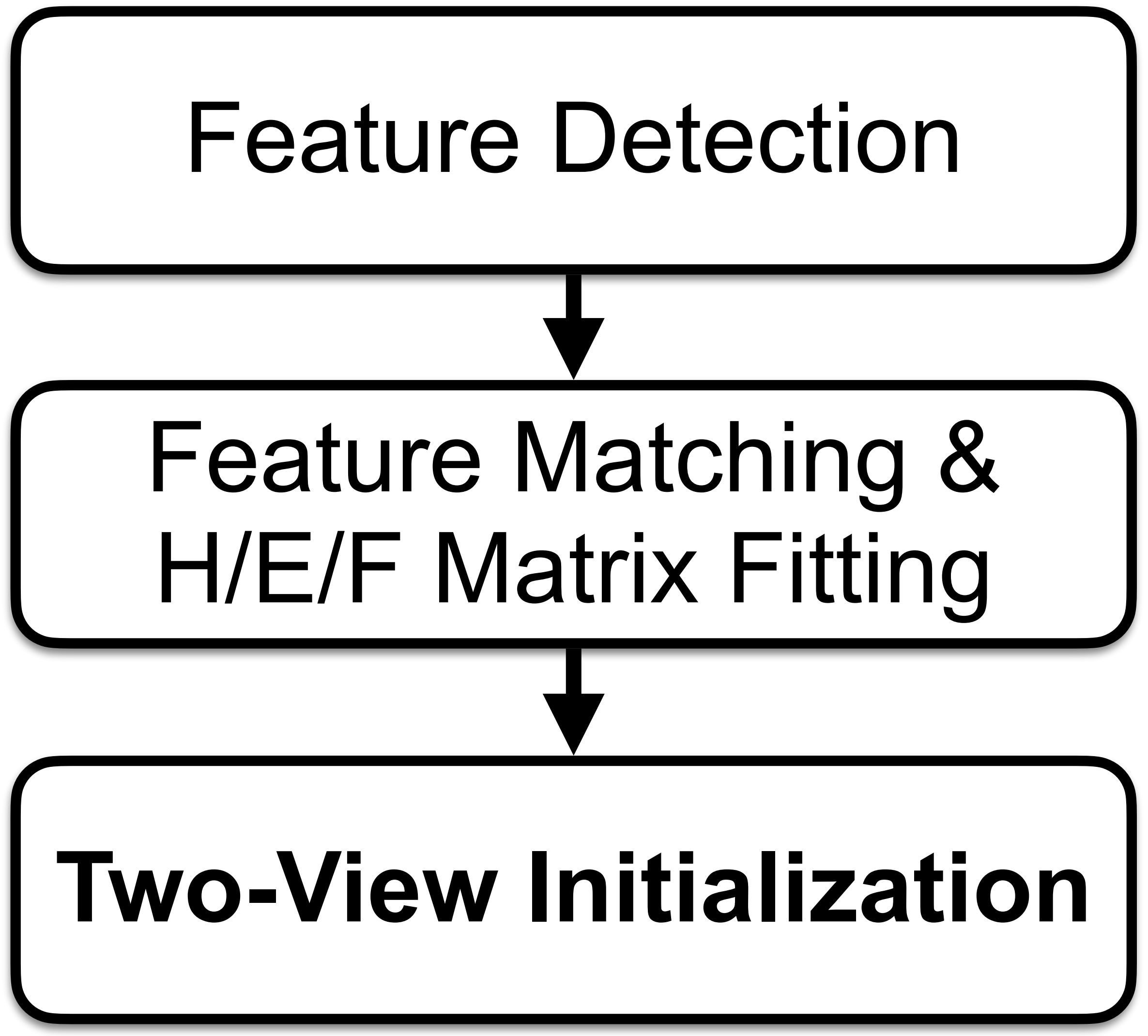
[Chum, Matas, Optimal Randomized RANSAC. PAMI 2008]

- See also USAC [Raguram et al., PAMI'13] [code] (good overview, nice implementation)
- Never use standard RANSAC!

[Fischler & Bolles, Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. CACM 1981]

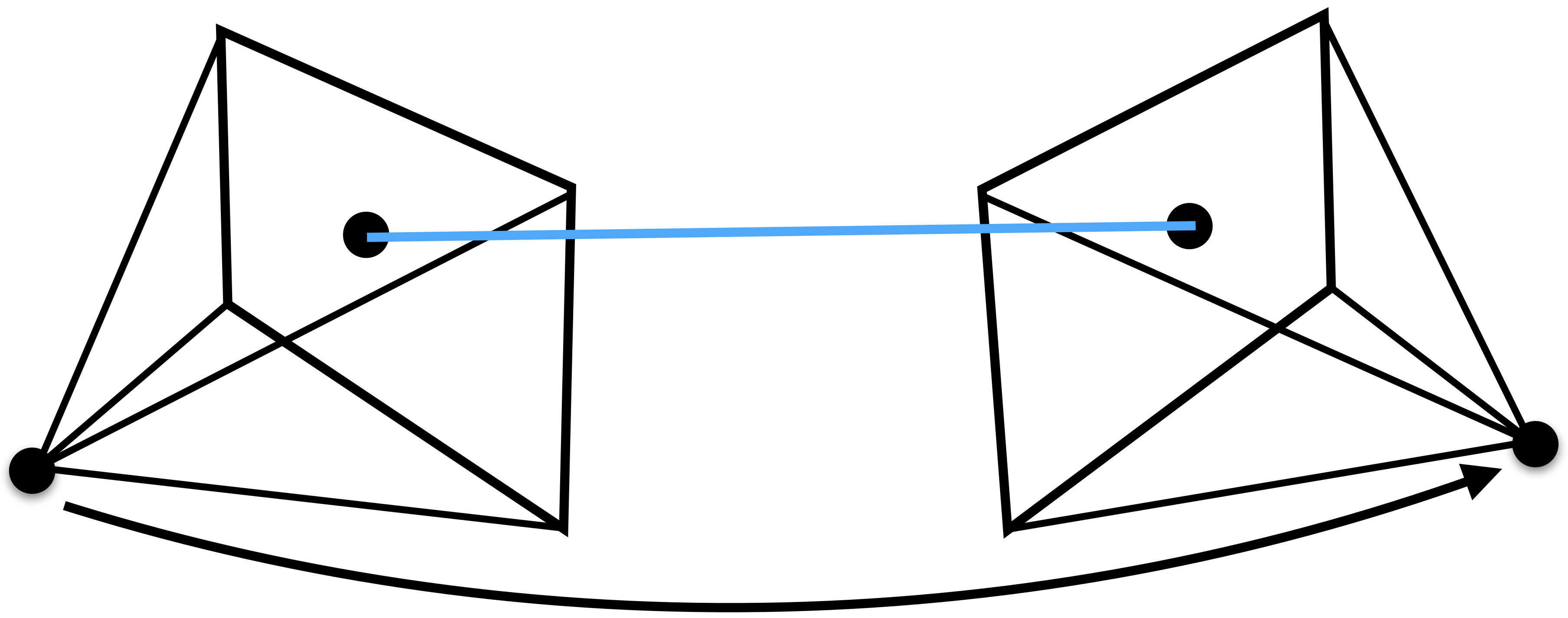
# Sequential / Incremental SfM



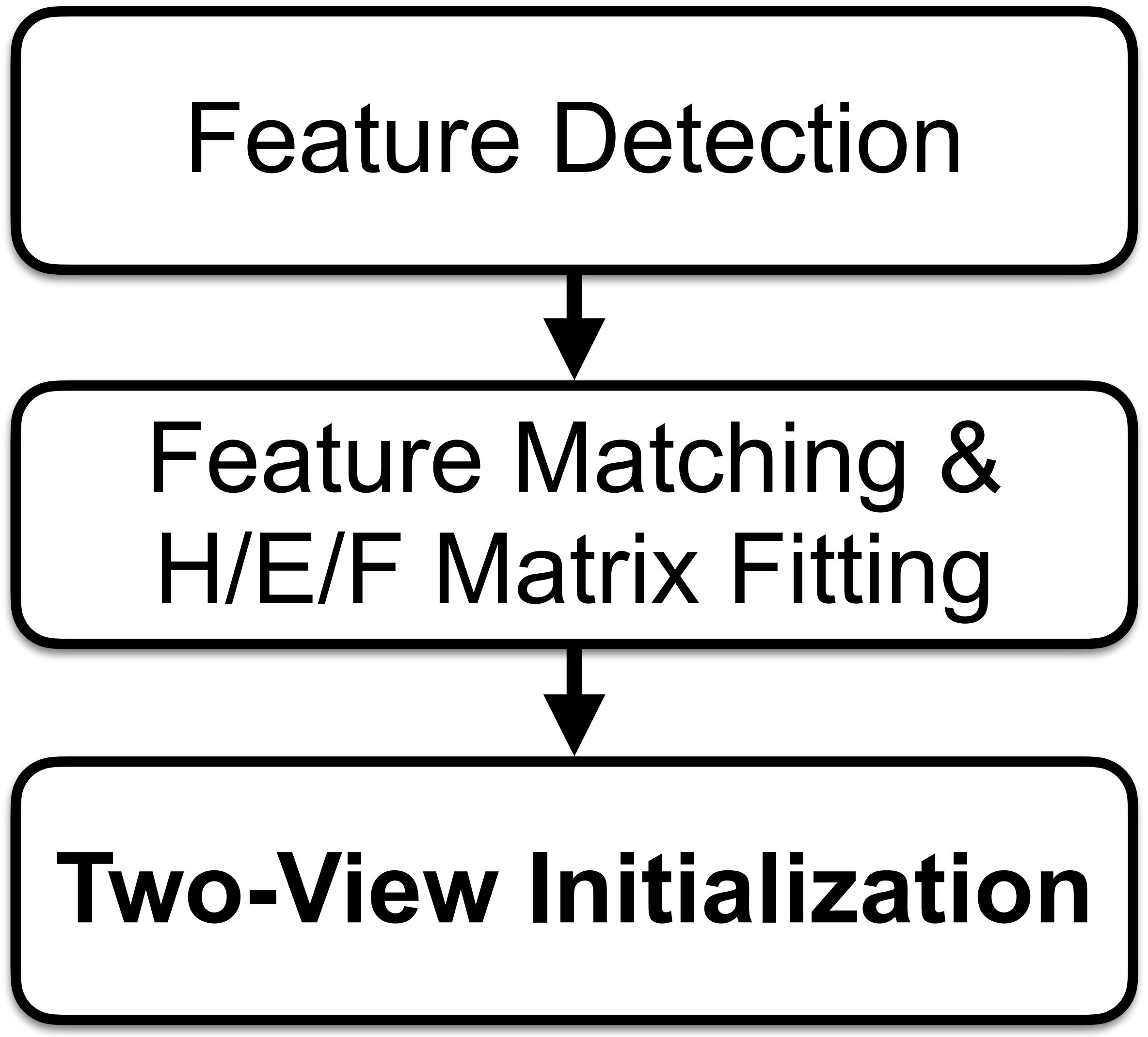




- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

# Sequential / Incremental SfM

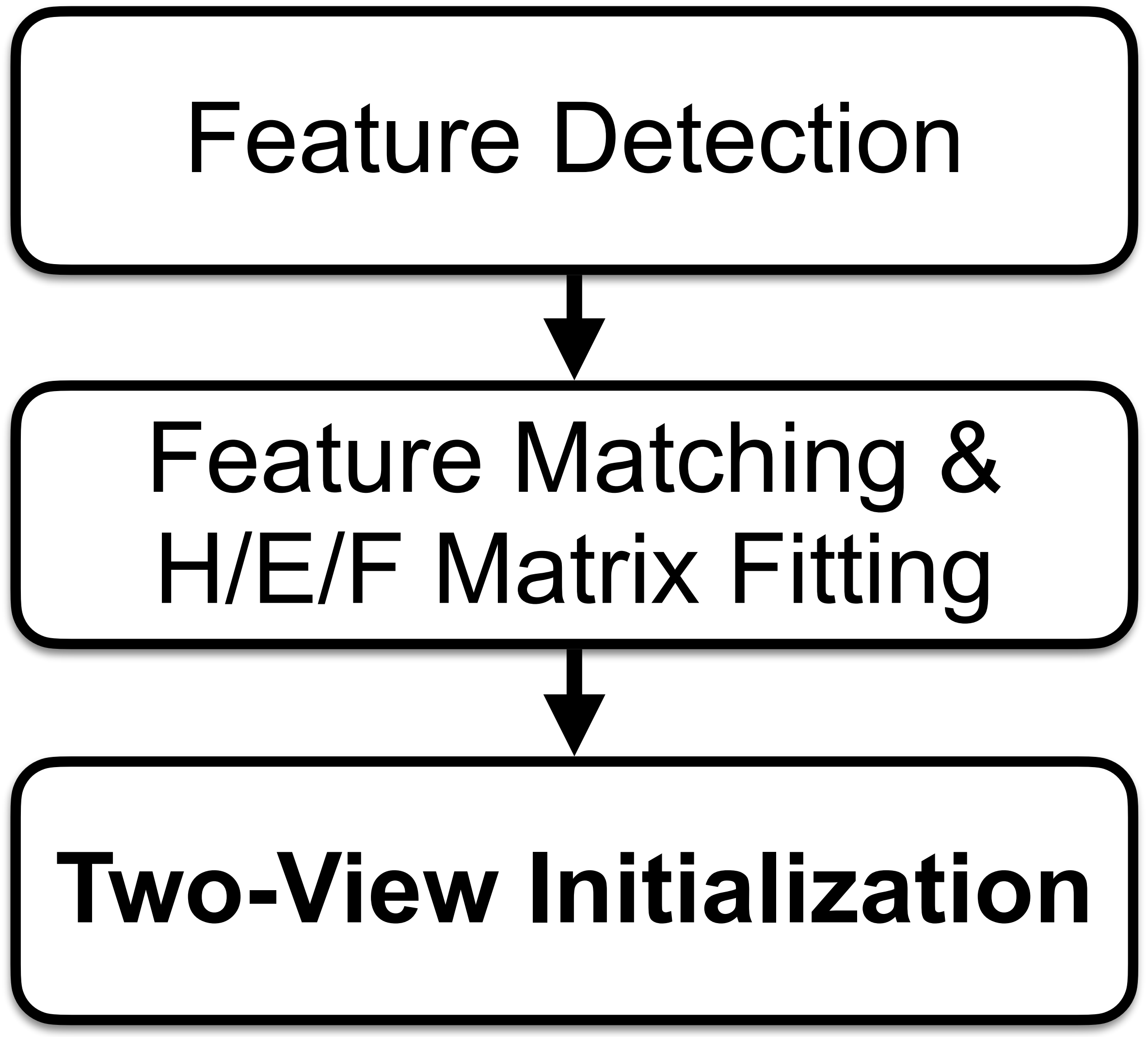


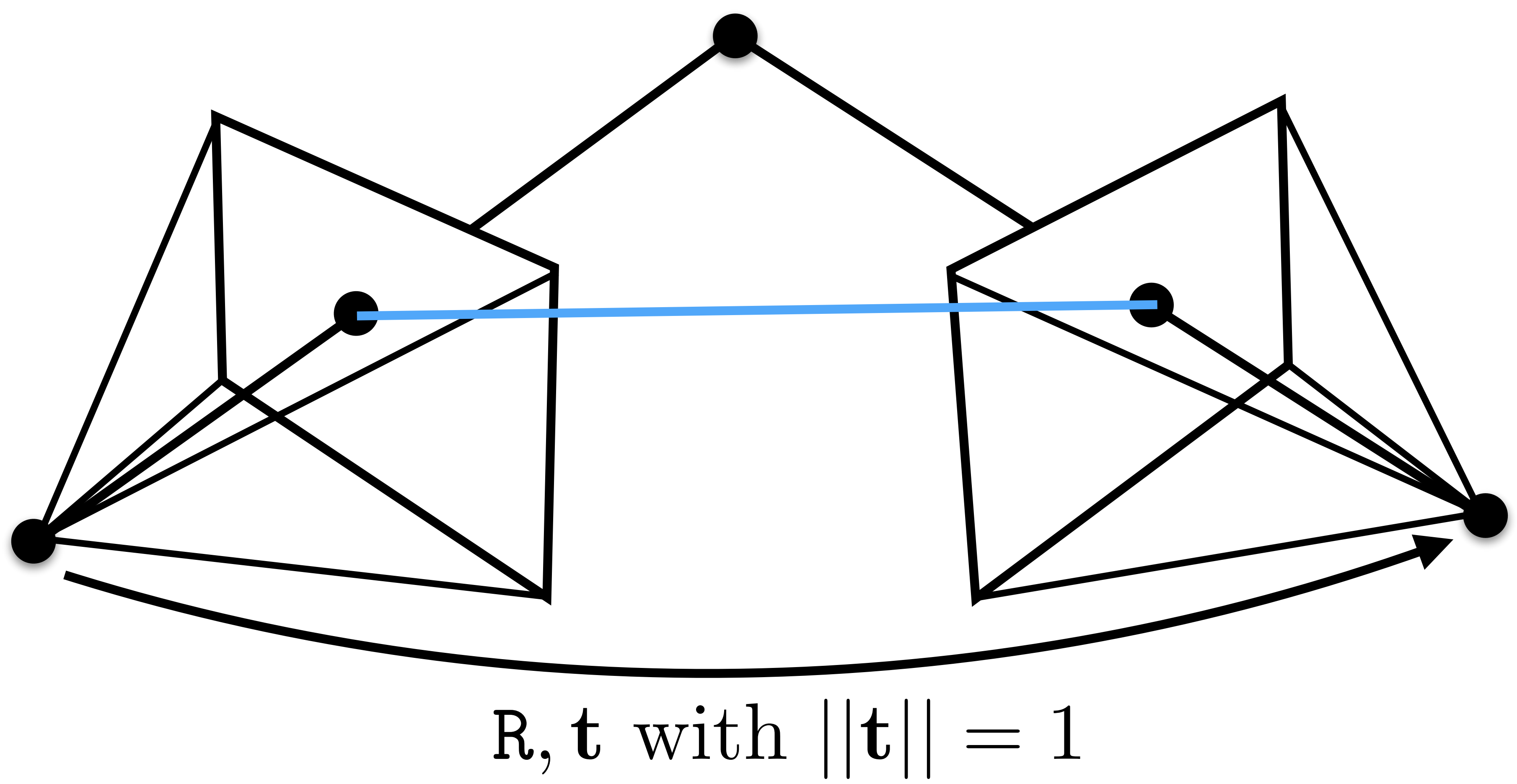
$$\mathtt{R}, \mathbf{t} \text{ with } ||\mathbf{t}|| = 1$$




- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

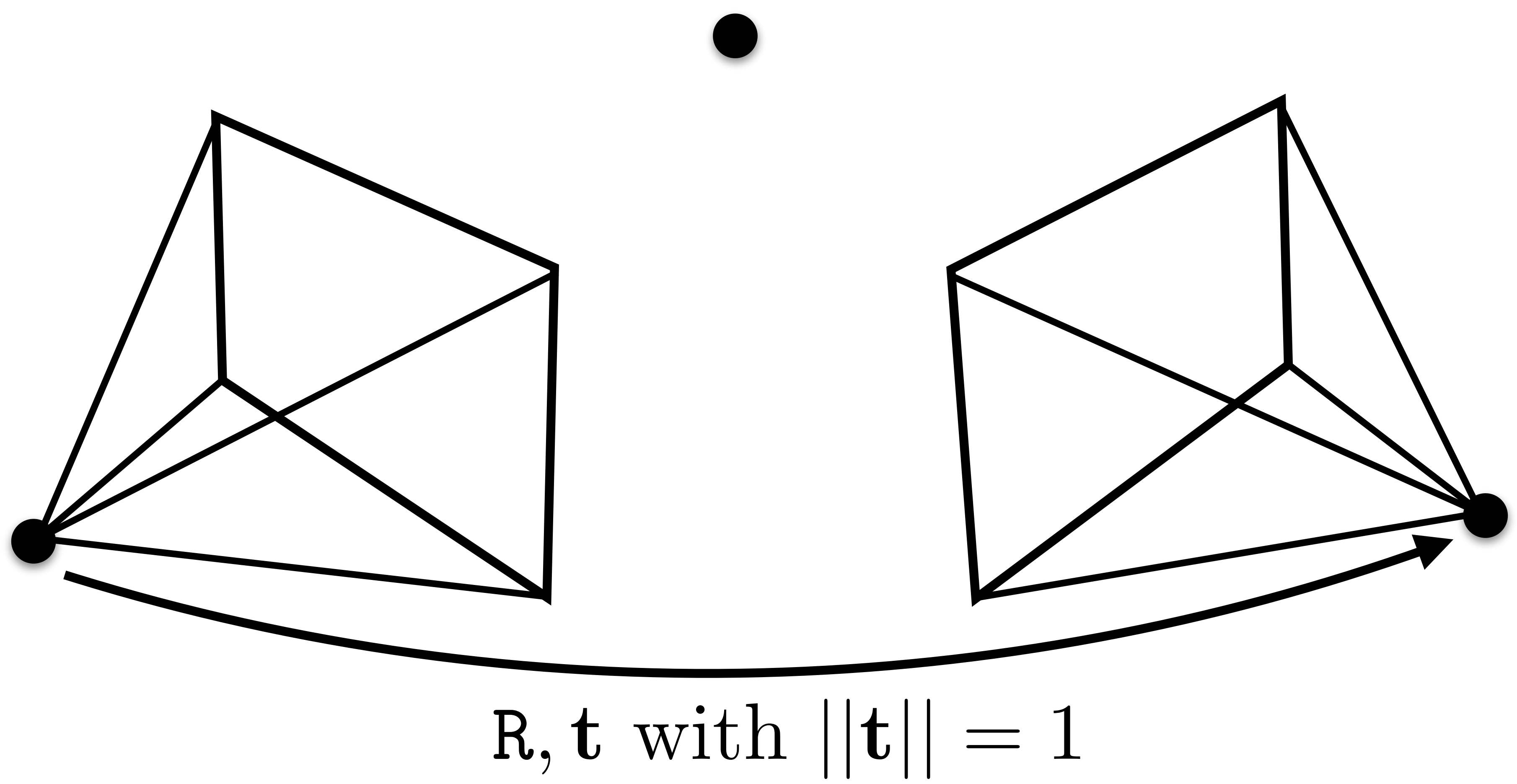
# Sequential / Incremental SfM

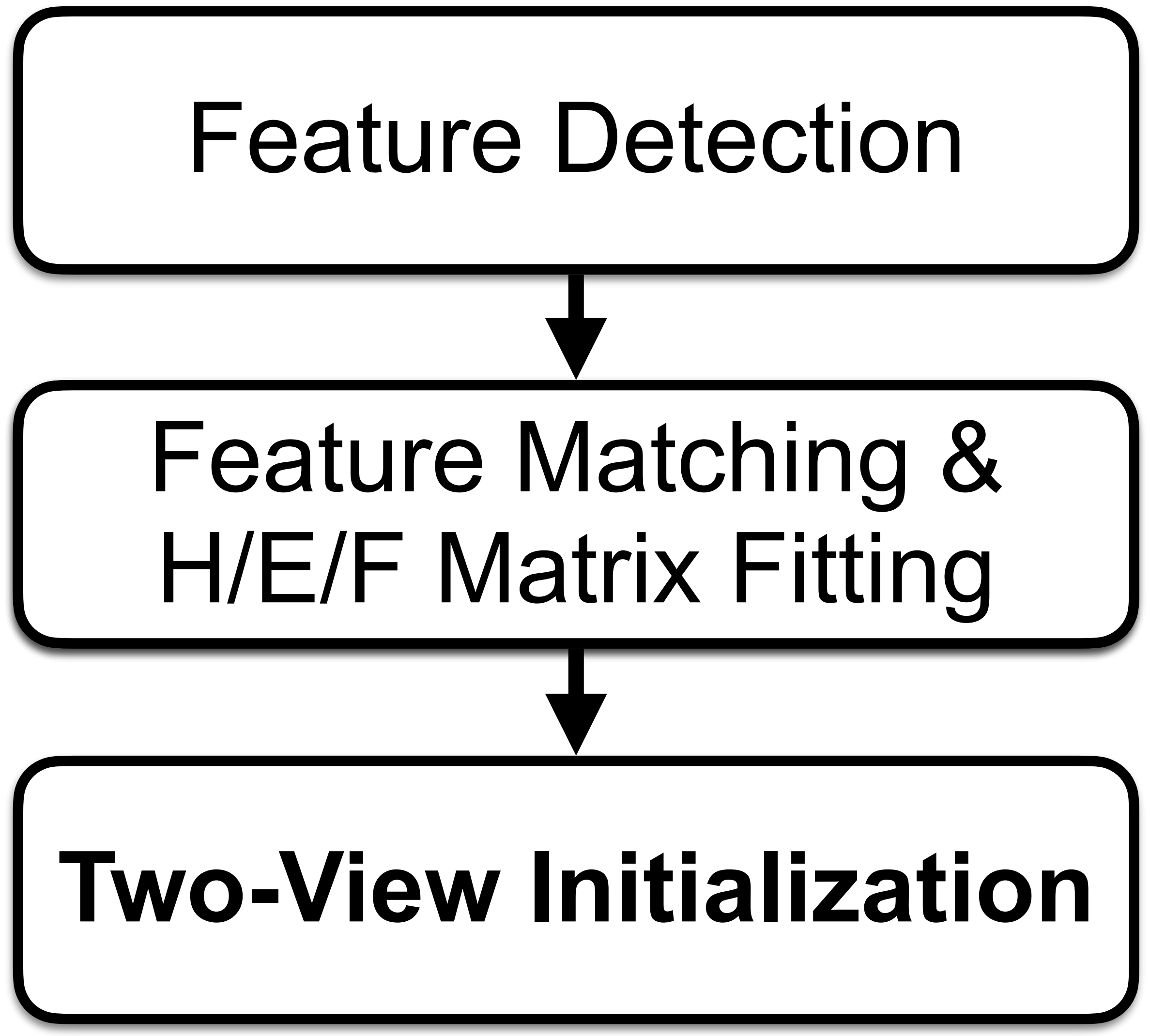



- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

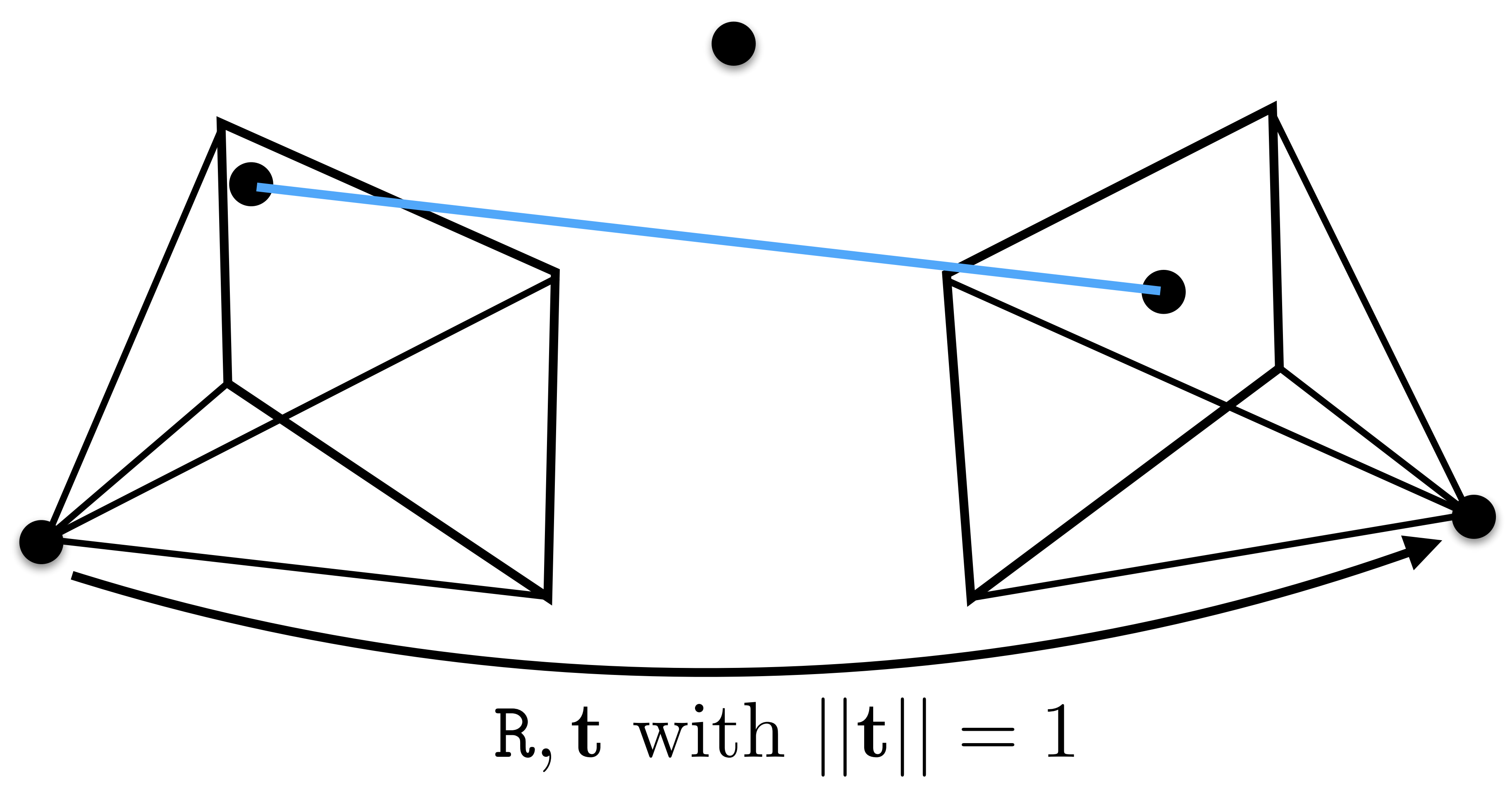
# Sequential / Incremental SfM




- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

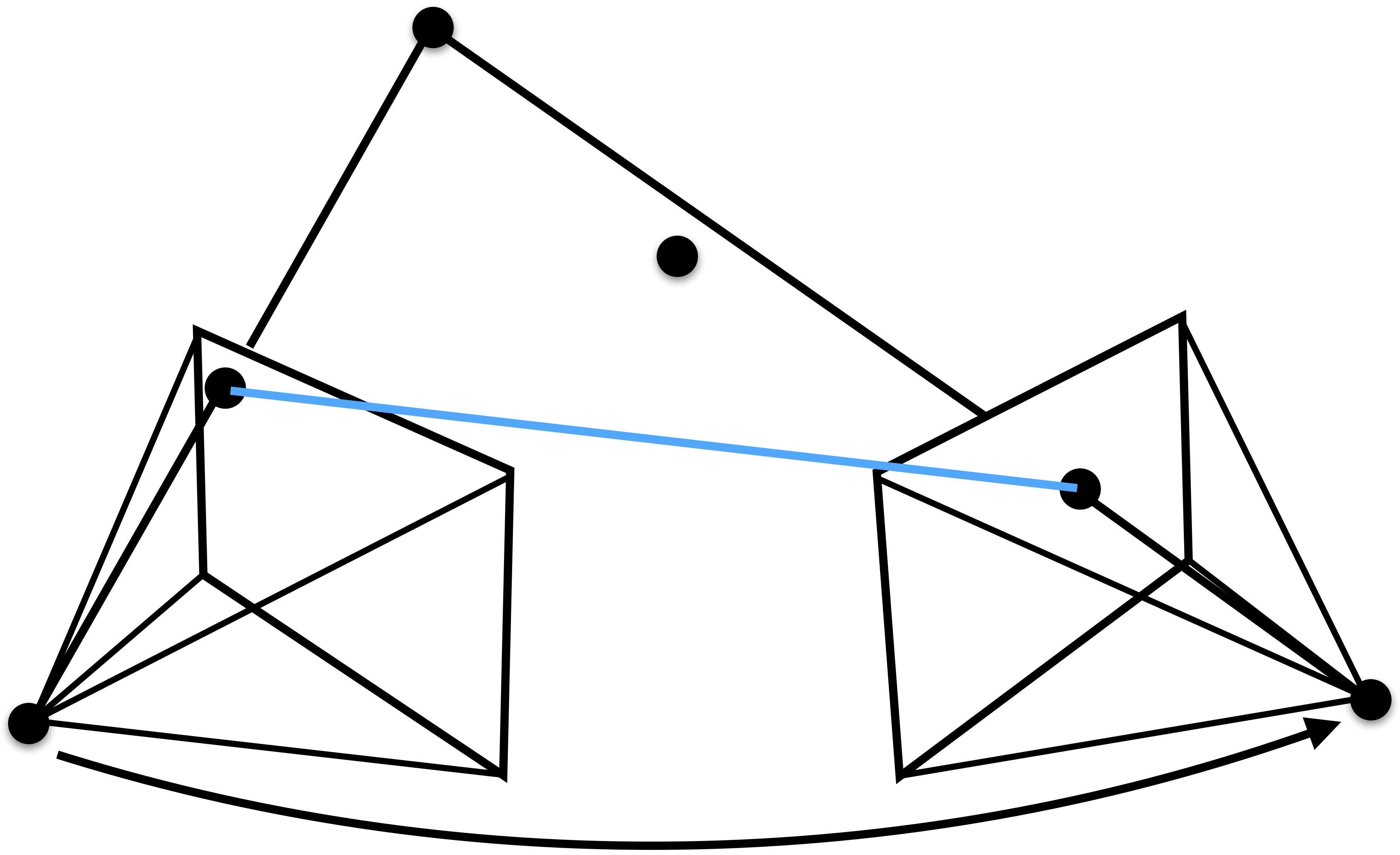
# Sequential / Incremental SfM



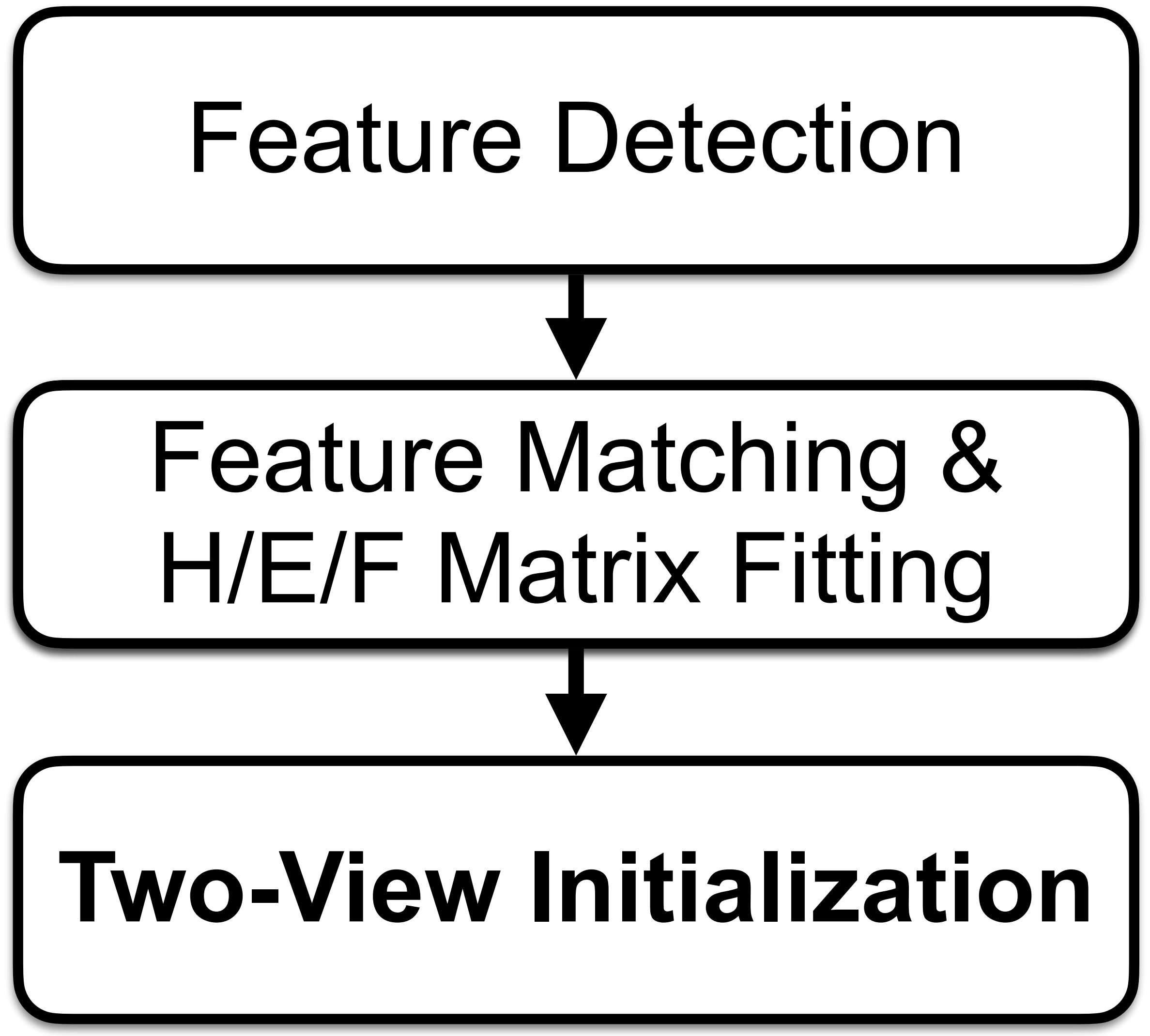

Feature Detection
↓
Feature Matching & H/E/F Matrix Fitting
↓
**Two-View Initialization**

- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

# Sequential / Incremental SfM

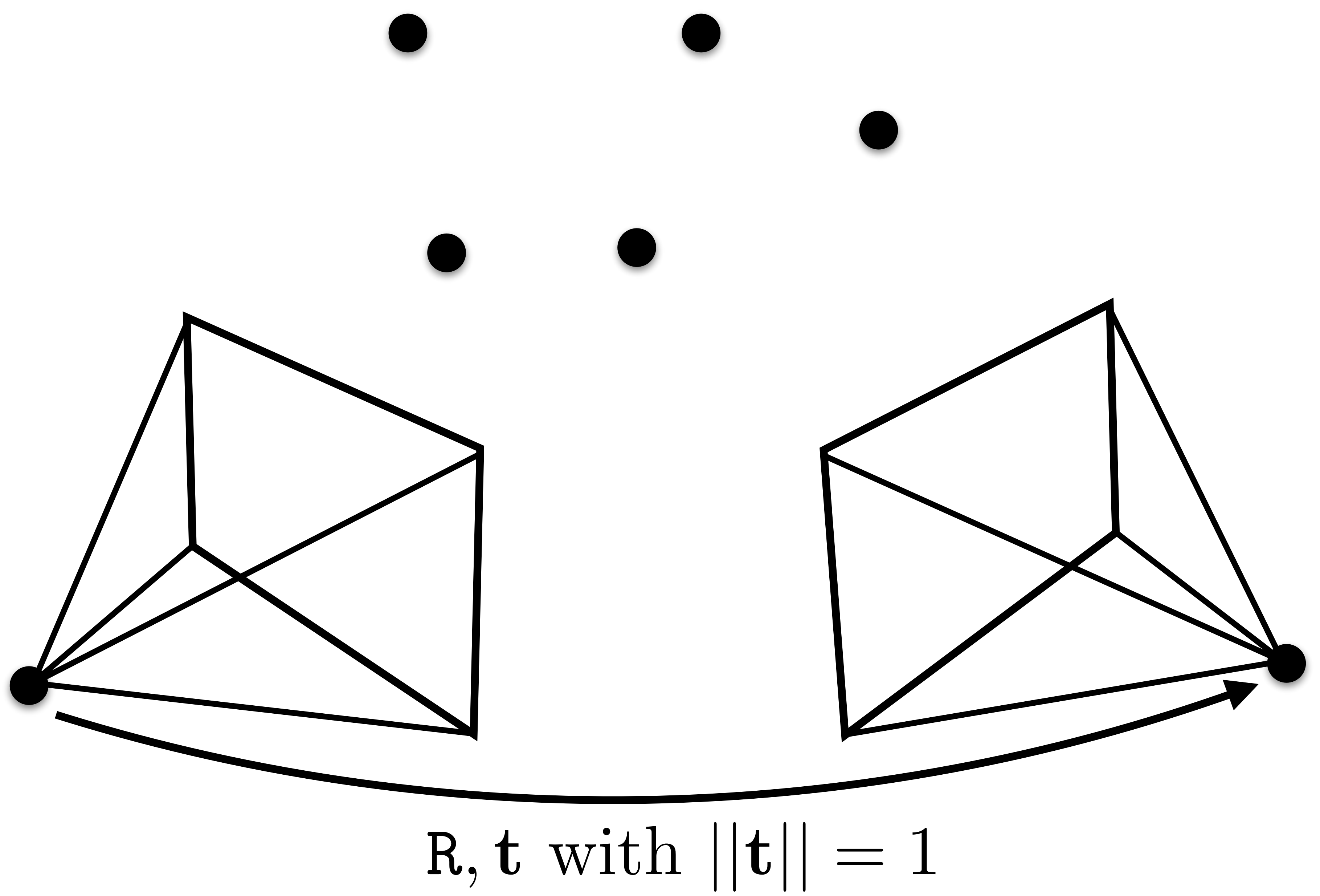


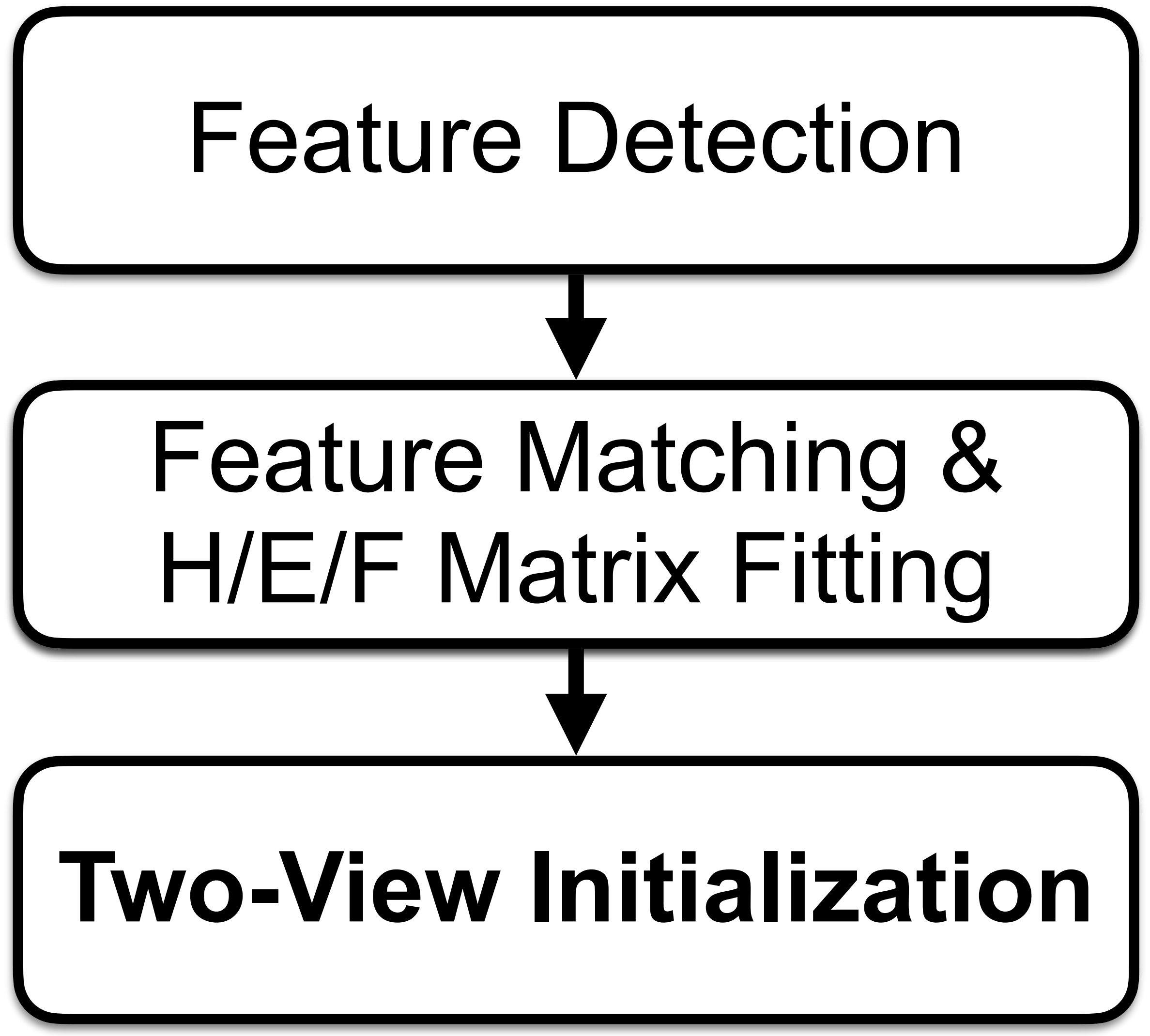
$\mathtt{R}, \mathbf{t}$ with $||\mathbf{t}|| = 1$




- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

# Sequential / Incremental SfM

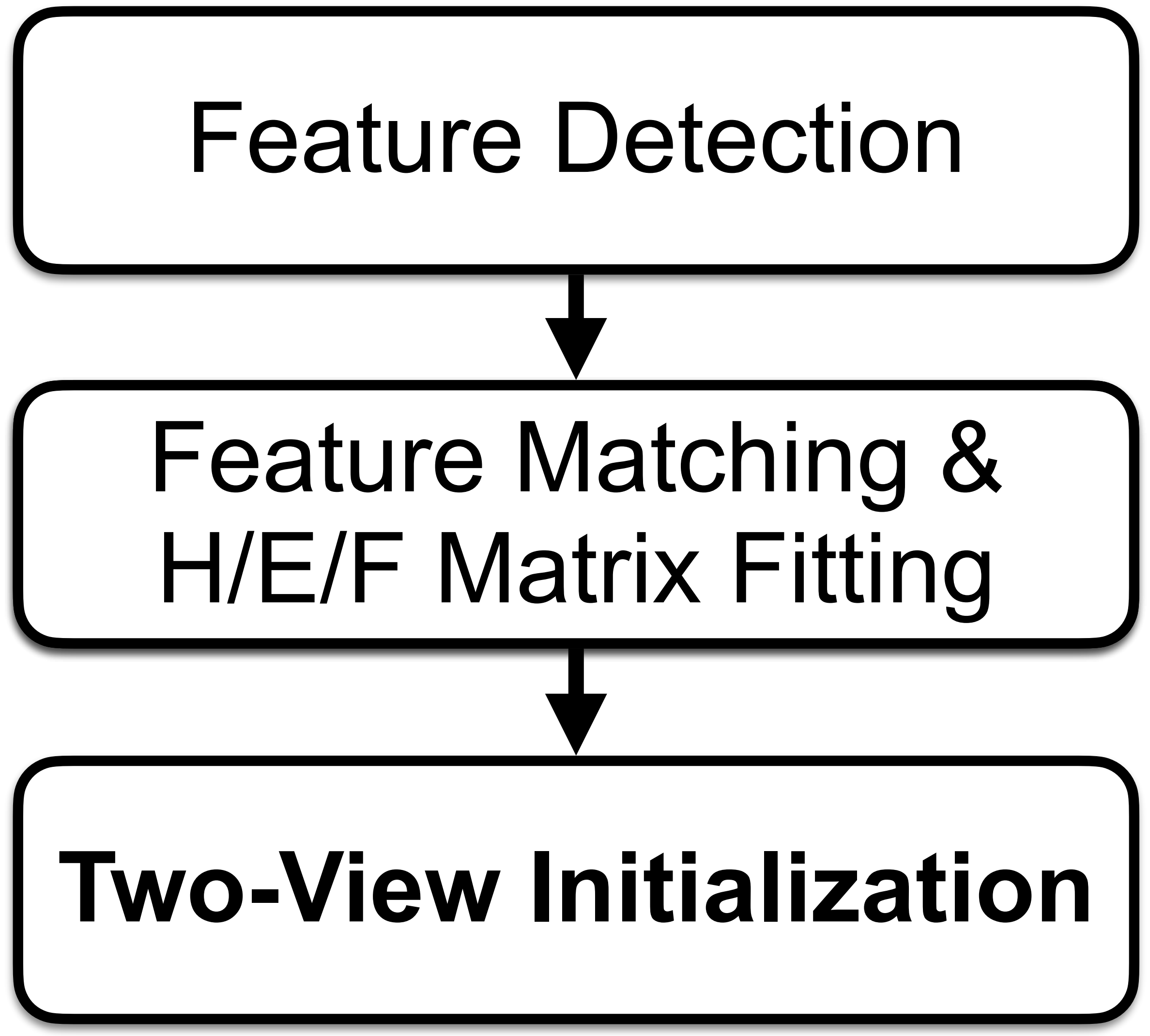



- Extract relative rotation and translation from H/E/F matrix
- Use 2D-2D matches to **triangulate** 3D structure

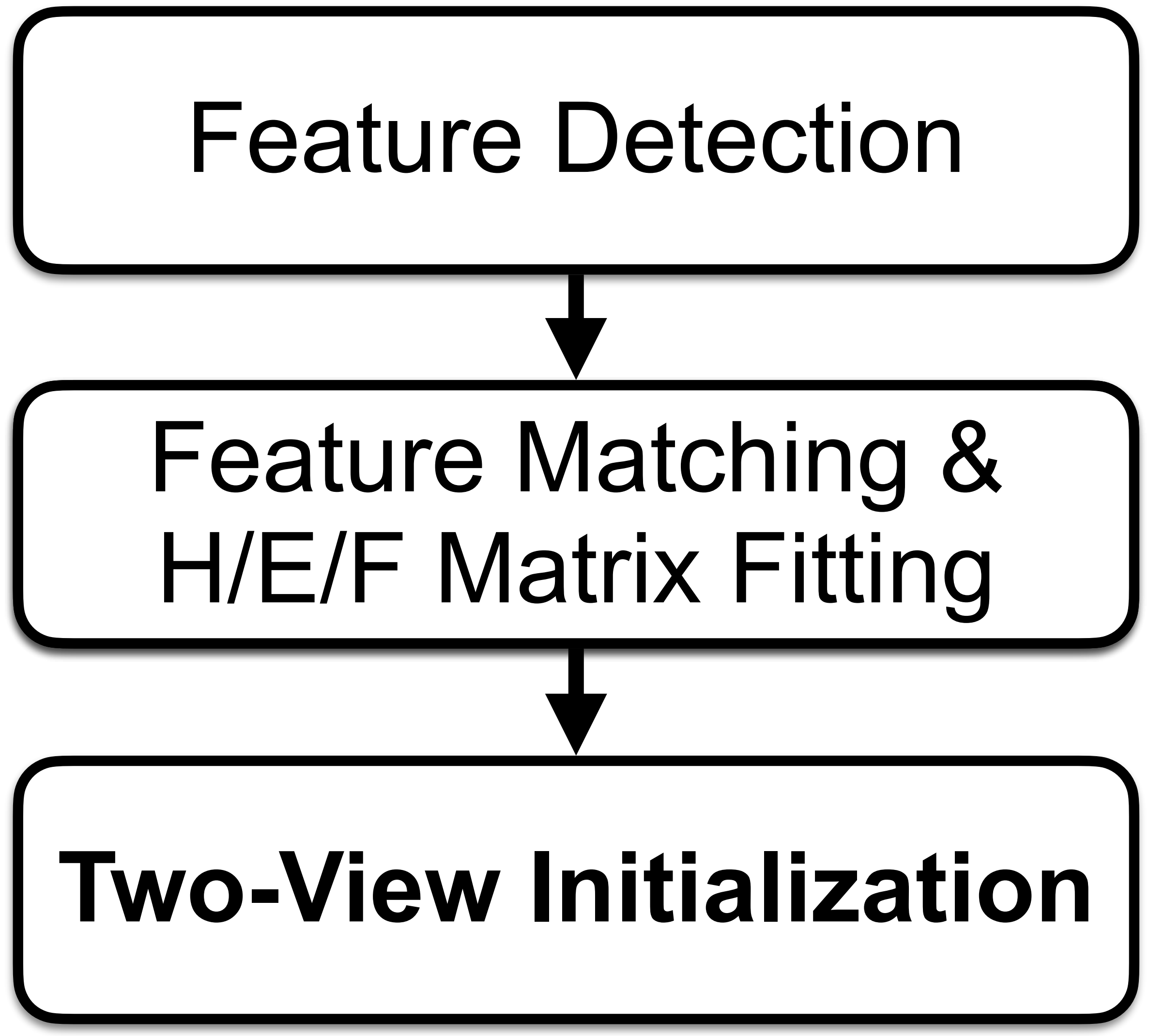
# Sequential / Incremental SfM

- **How to select a good initial pair?**



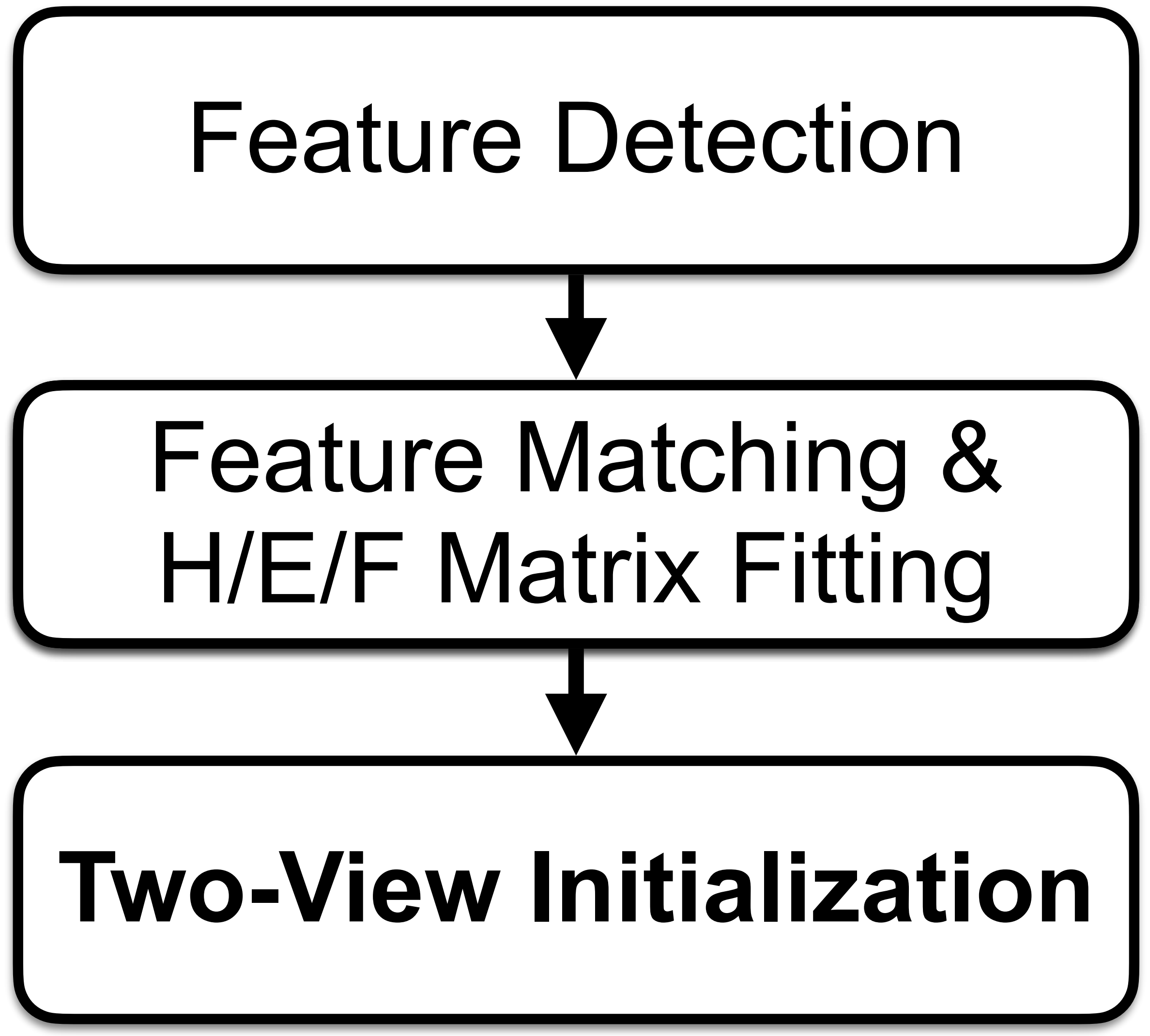
# Sequential / Incremental SfM

- **How to select a good initial pair?**
- Criteria:



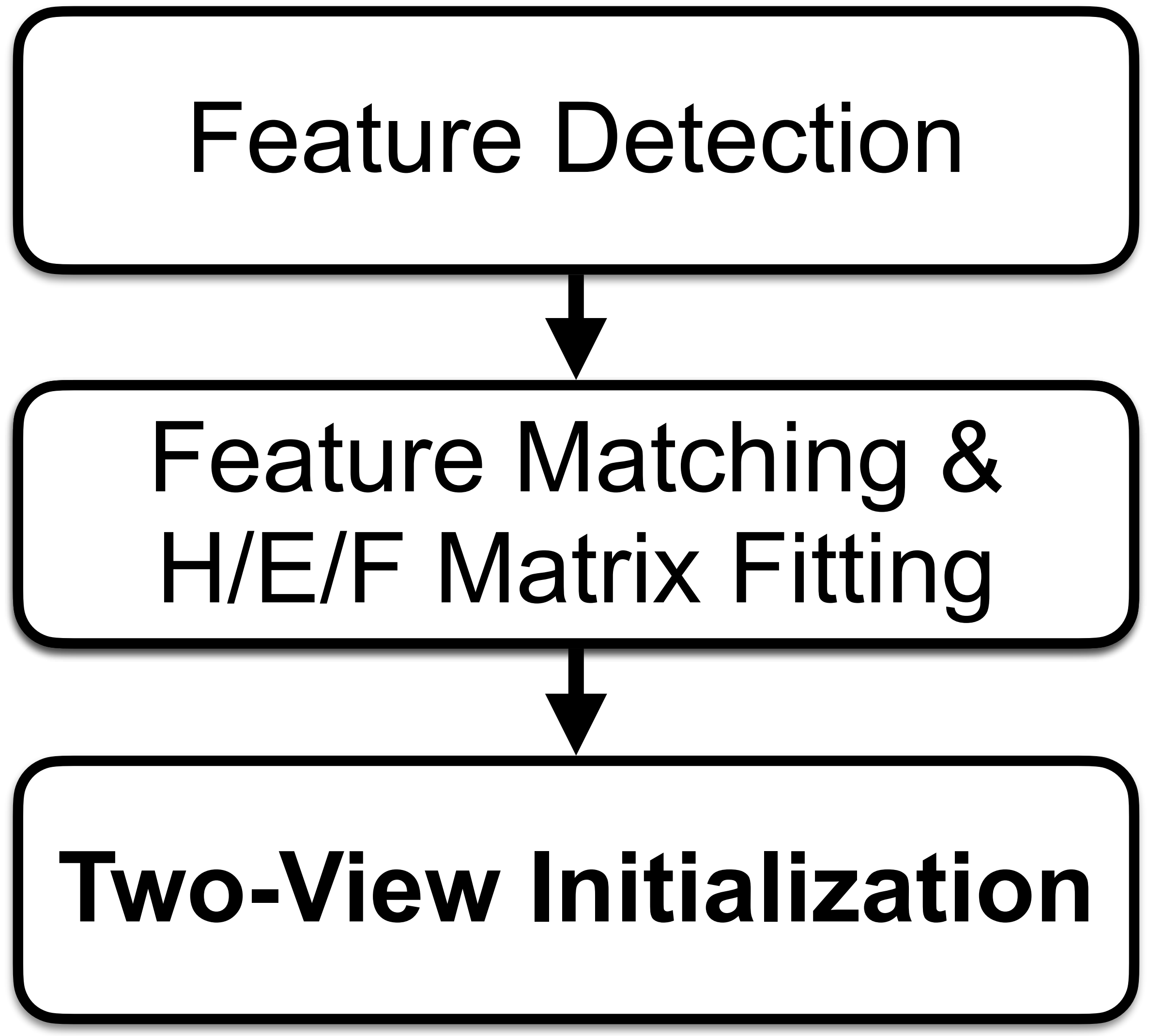
# Sequential / Incremental SfM

- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches



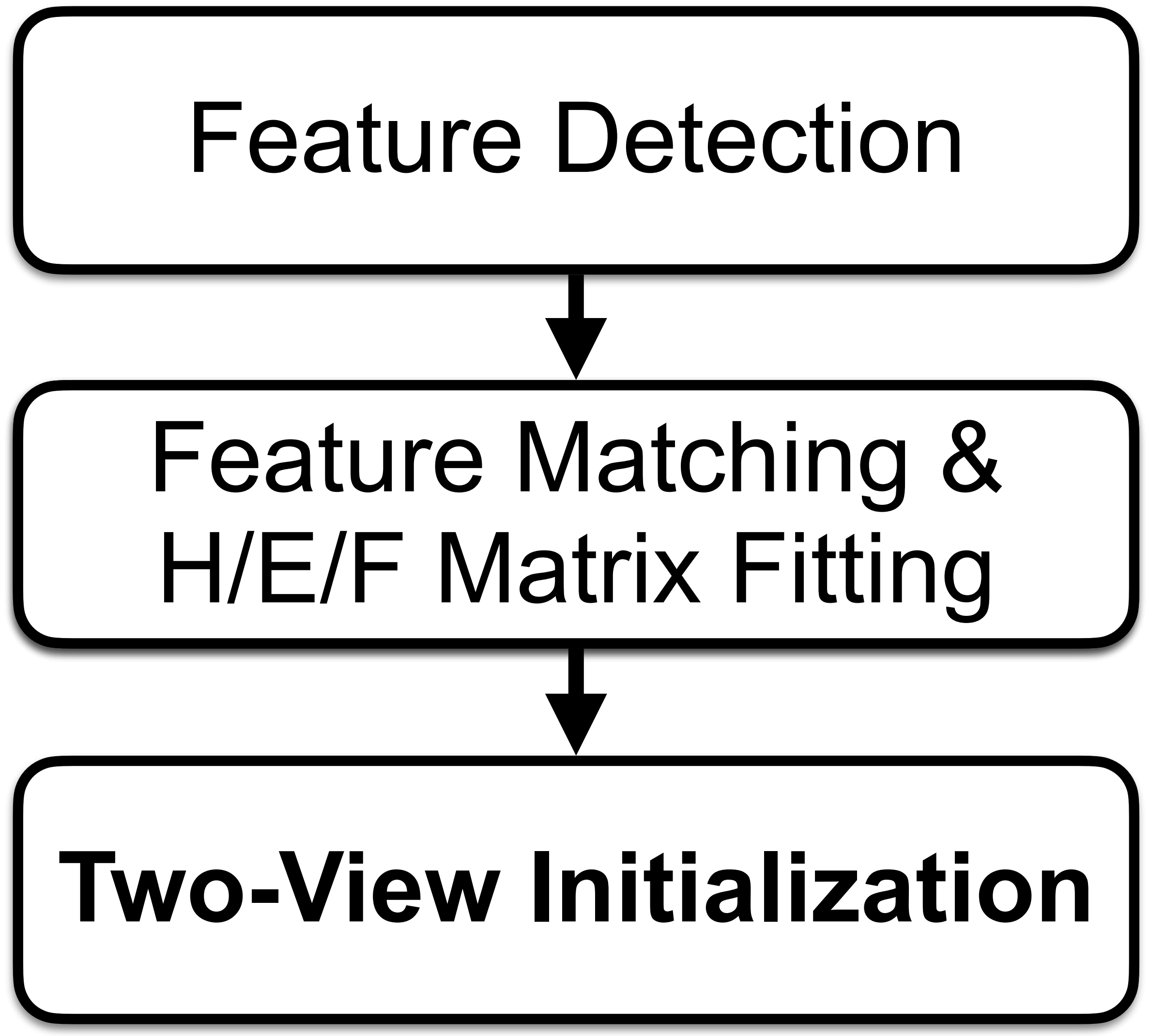
# Sequential / Incremental SfM

- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches
  - Non-planar scene (planar scenes are degeneracy for F-matrix fitting)

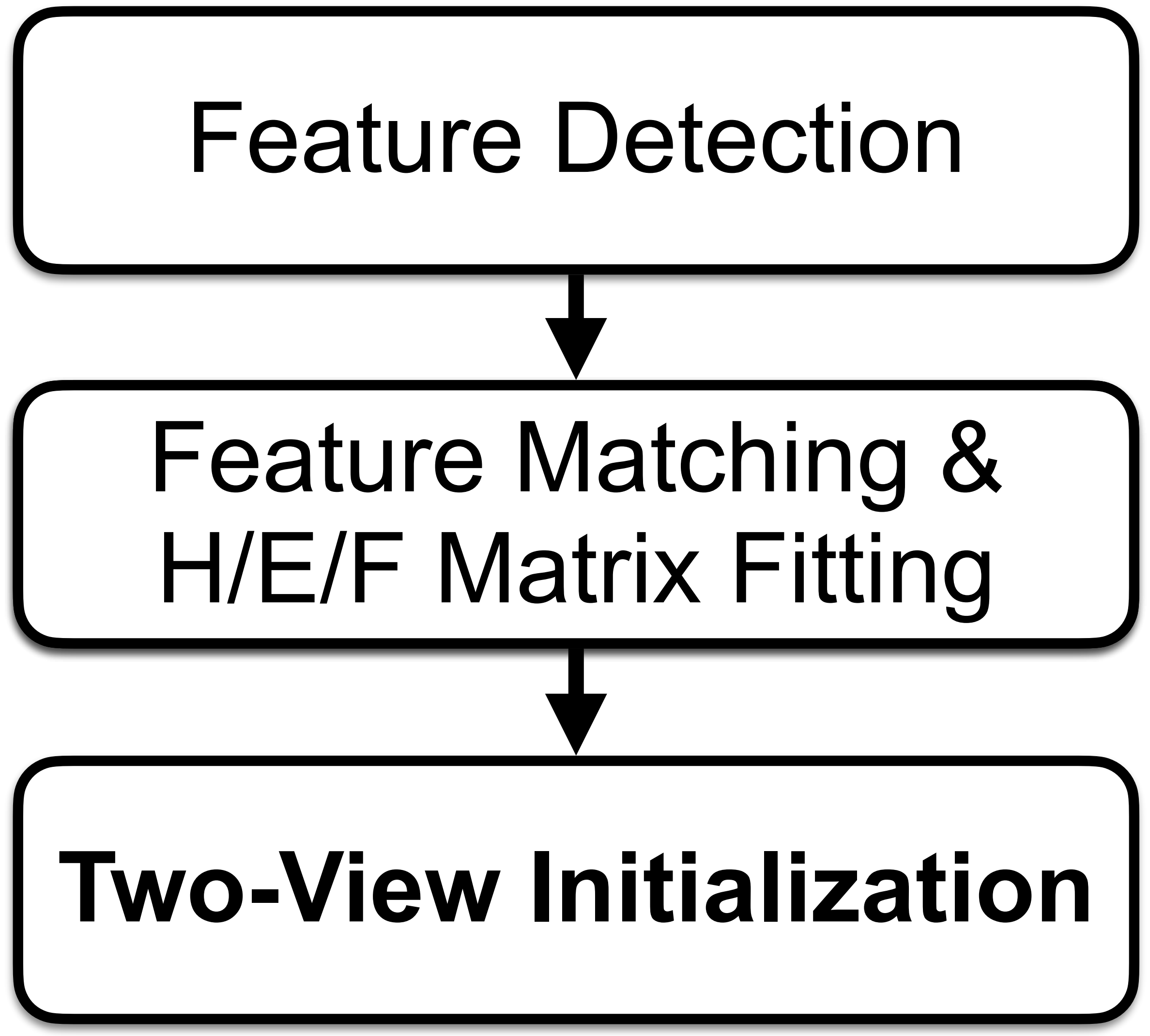


# Sequential / Incremental SfM

- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches
  - Non-planar scene (planar scenes are degeneracy for F-matrix fitting)
    - Compute both H and E/F matrix

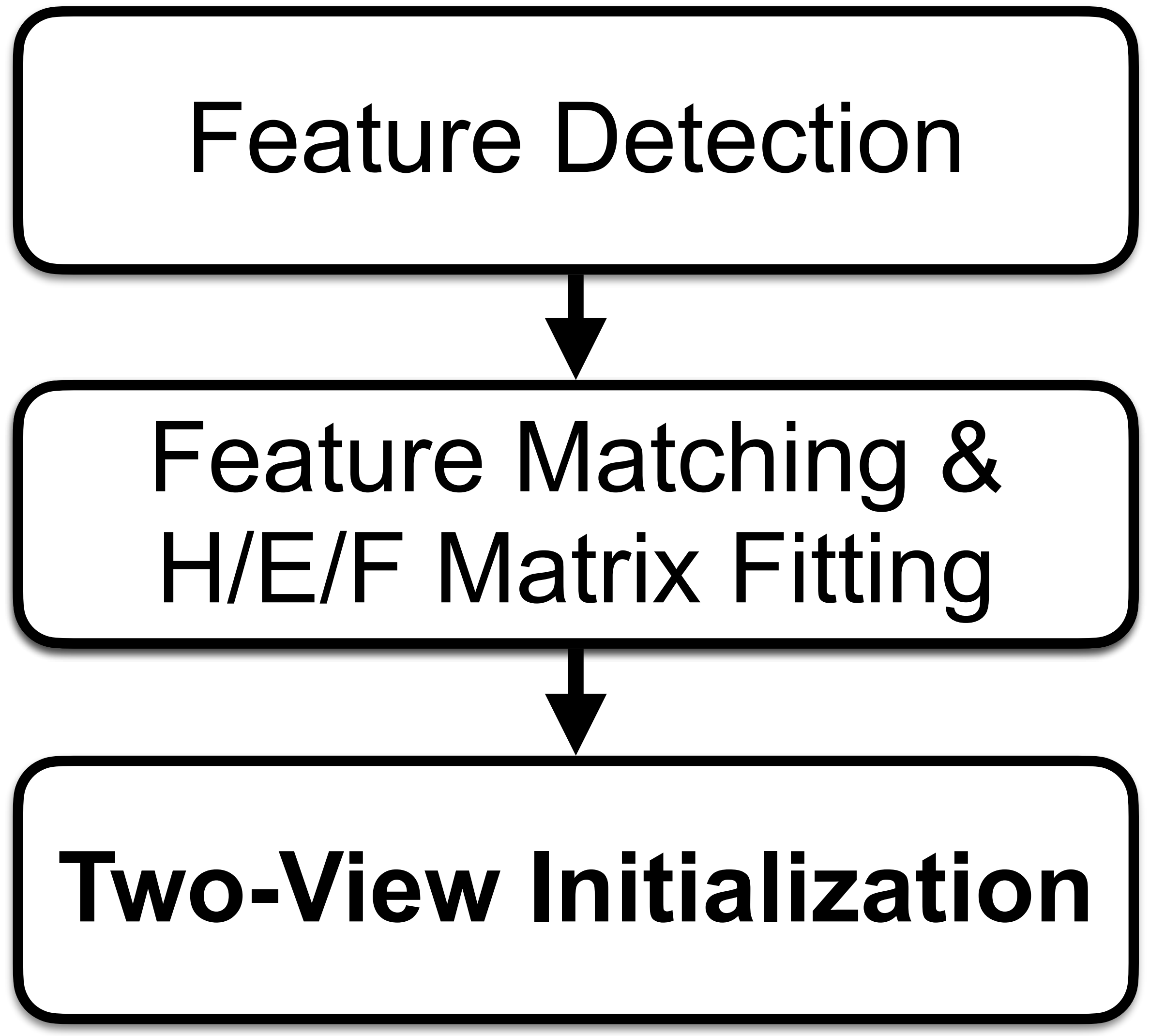


# Sequential / Incremental SfM




- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches
  - Non-planar scene (planar scenes are degeneracy for F-matrix fitting)
    - Compute both H and E/F matrix
    - Select pair with large ratio #inliers(E/F) / #inliers(H)

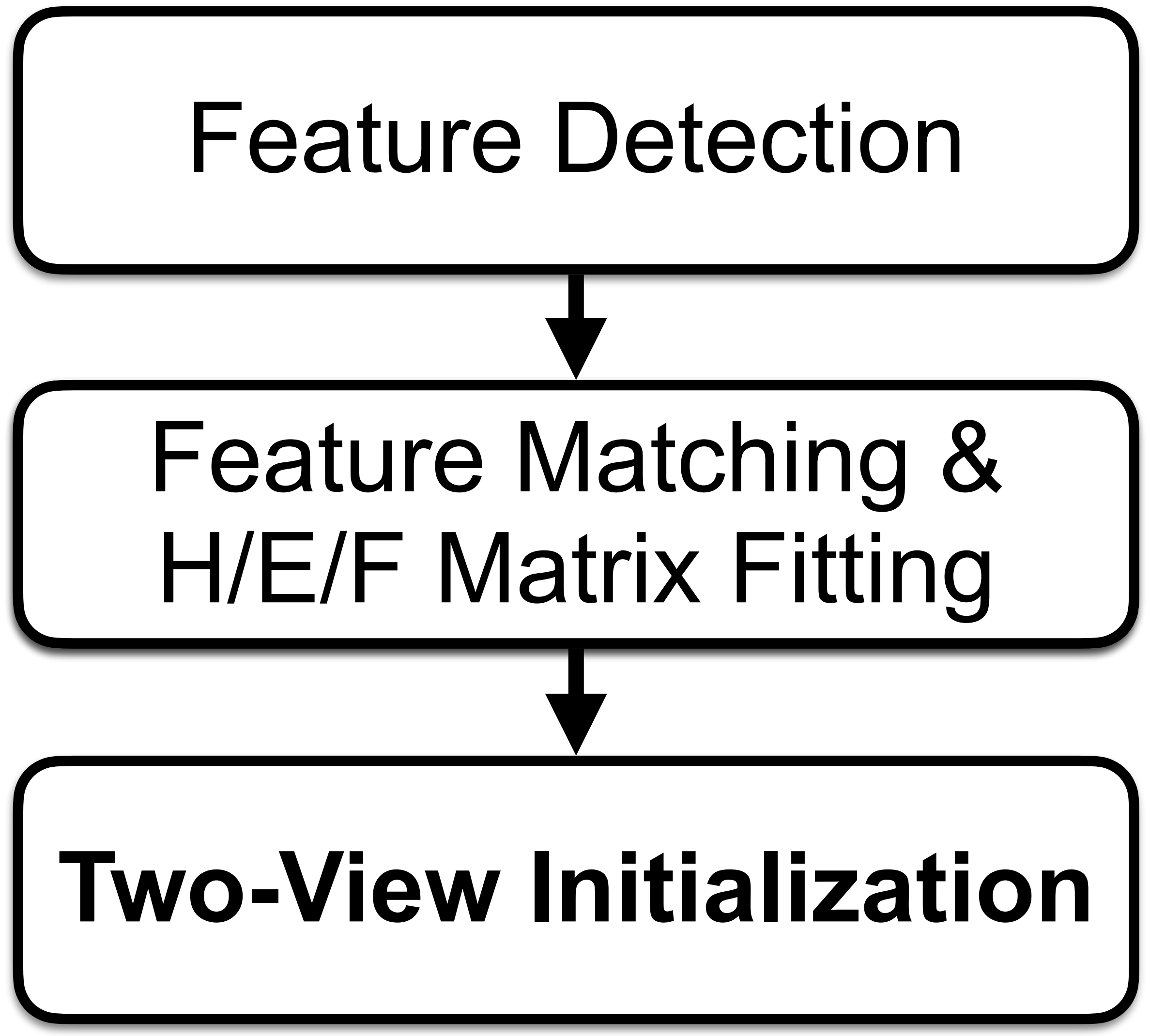
# Sequential / Incremental SfM



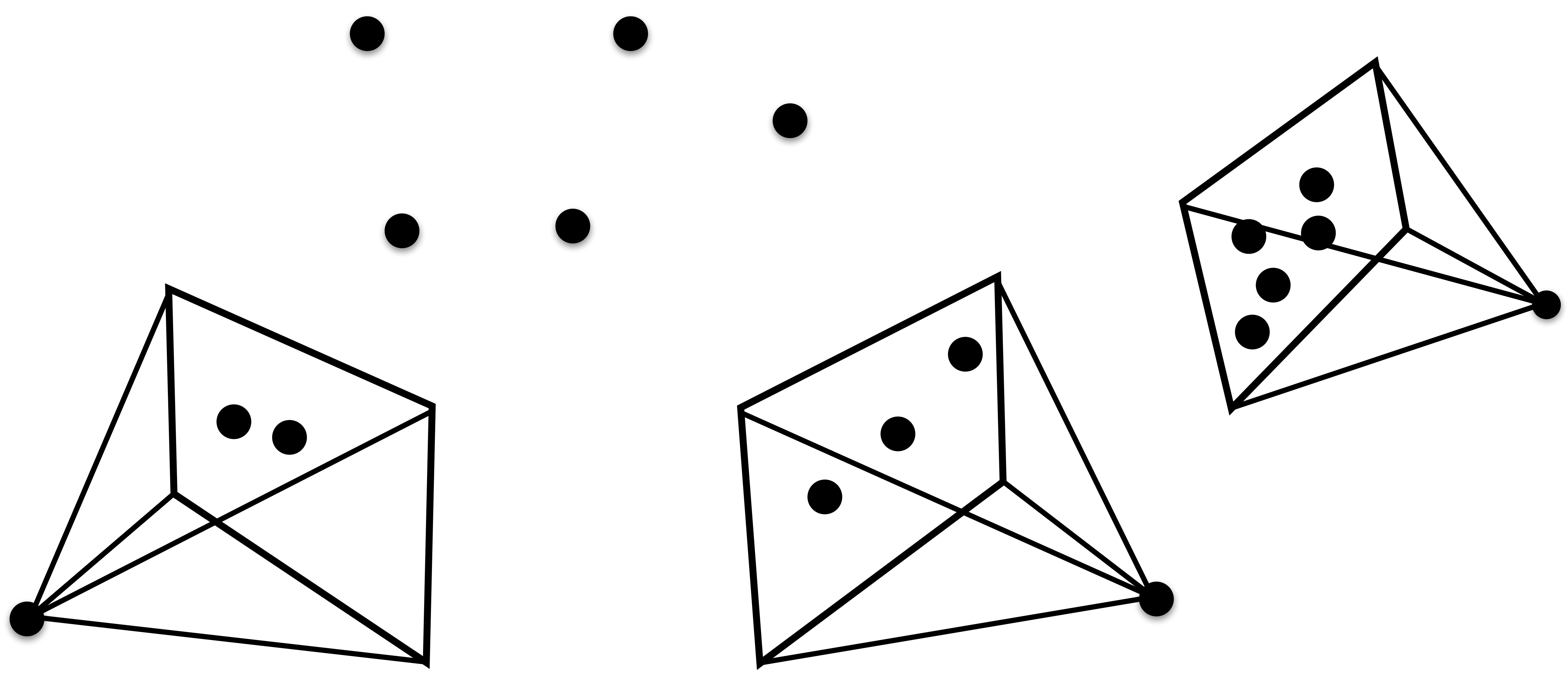

- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches
  - Non-planar scene (planar scenes are degeneracy for F-matrix fitting)
    - Compute both H and E/F matrix
    - Select pair with large ratio #inliers(E/F) / #inliers(H)
  - No pure forward motion (triangulation inaccurate / impossible)

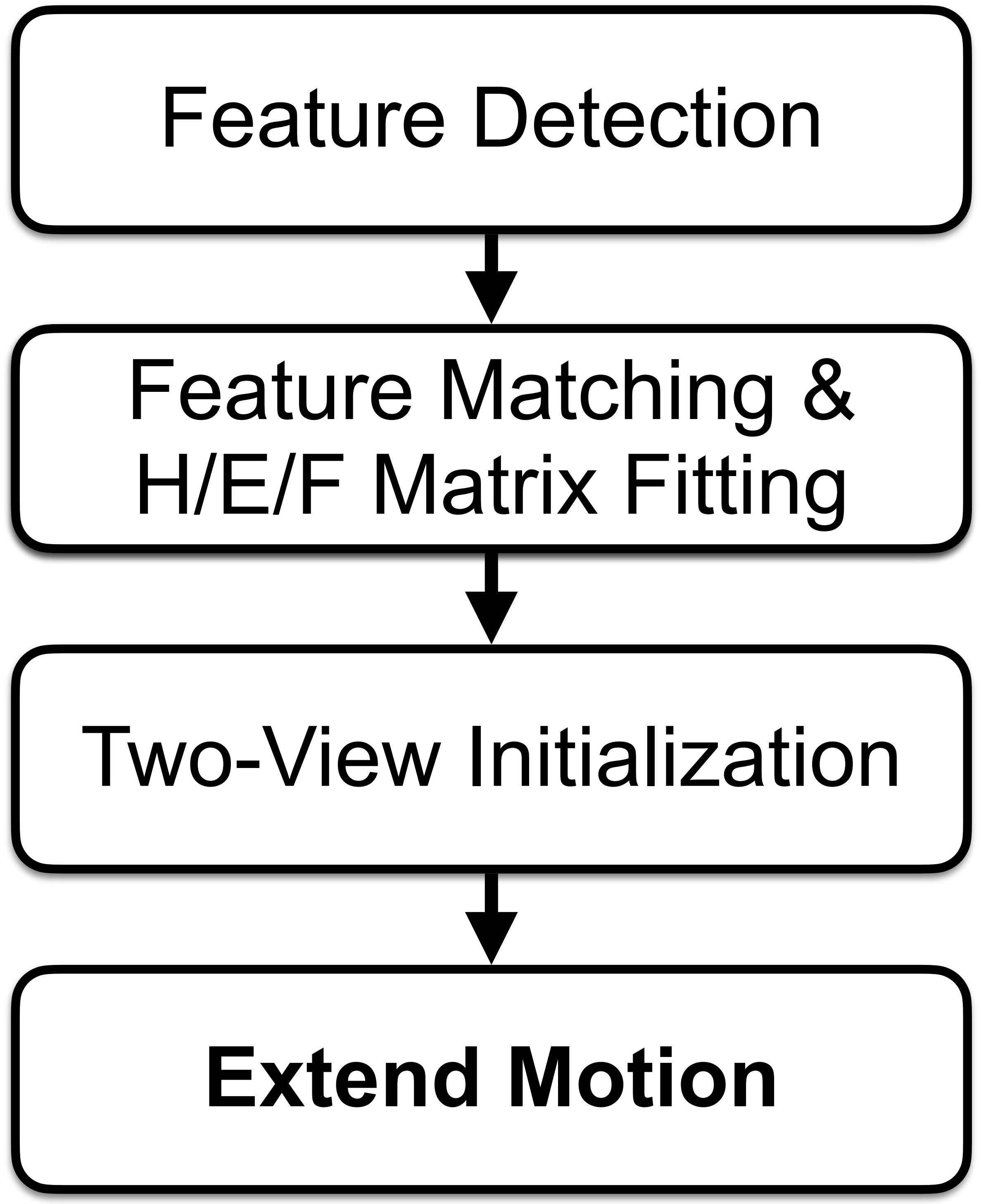
# Sequential / Incremental SfM

- **How to select a good initial pair?**
- Criteria:
  - Accurate relative pose ≈ many inlier matches
  - Non-planar scene (planar scenes are degeneracy for F-matrix fitting)
    - Compute both H and E/F matrix
    - Select pair with large ratio #inliers(E/F) / #inliers(H)
  - No pure forward motion (triangulation inaccurate / impossible)
- In practice, try out multiple initial pairs



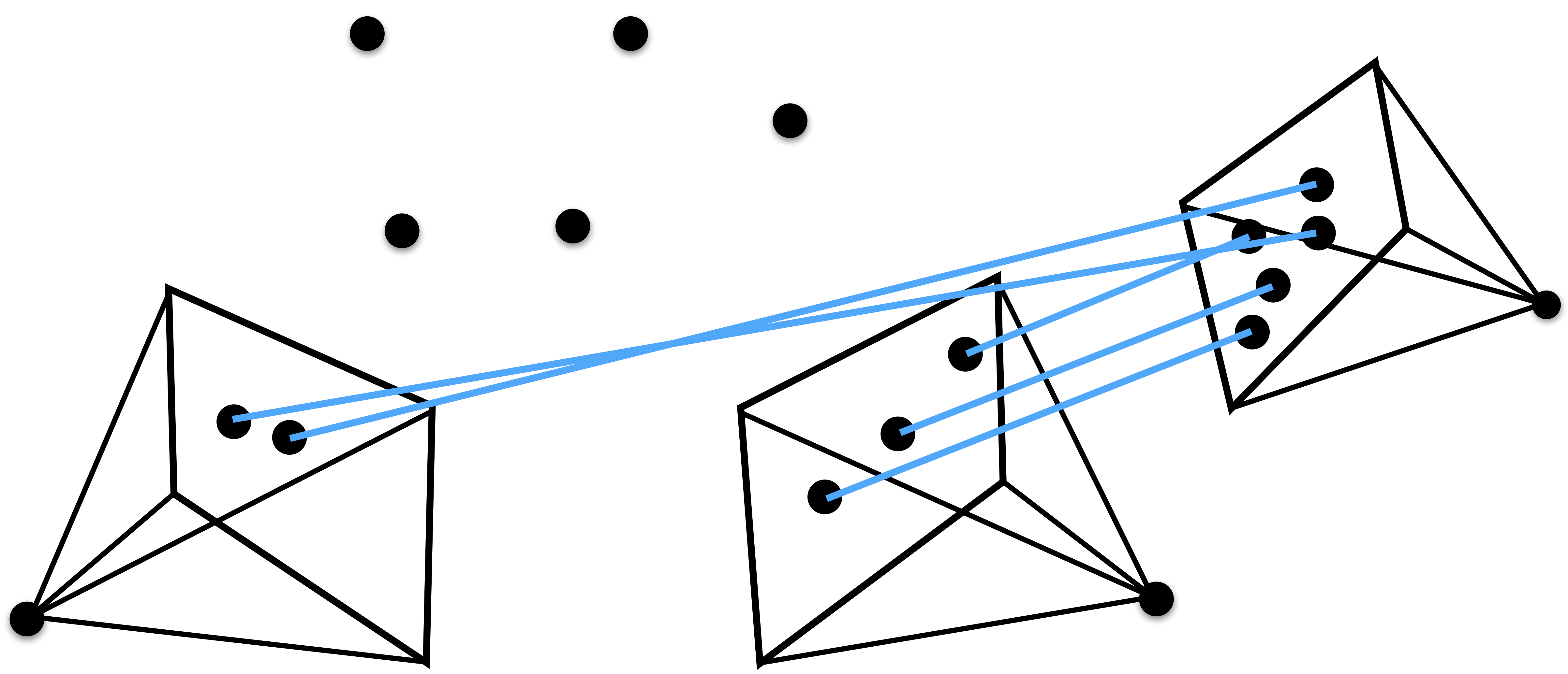
# Sequential / Incremental SfM

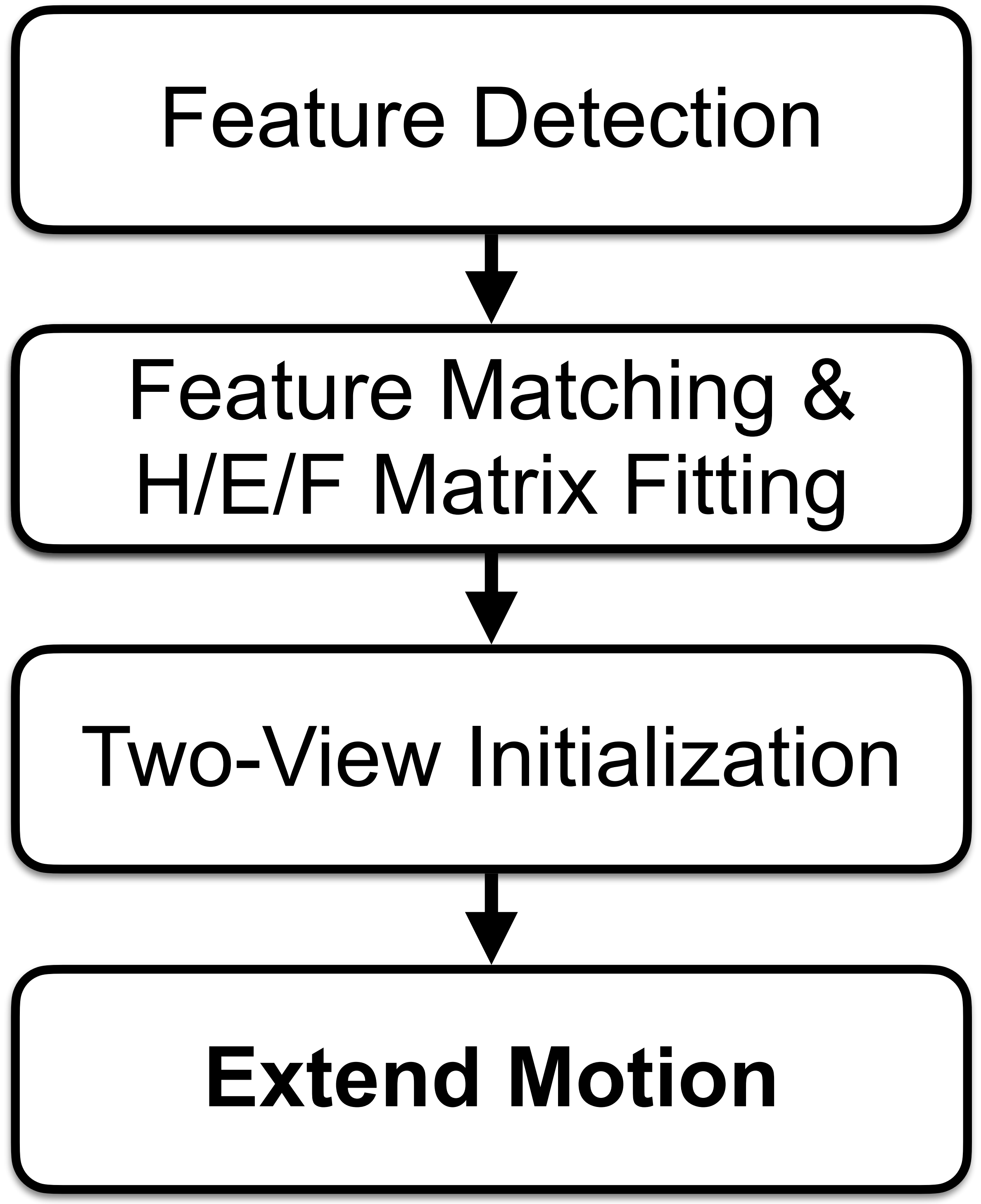





- Pick image(s) with large number of matches to existing cameras
- Obtain 2D-3D matches from 2D-2D matches
- Estimate absolute pose of new image

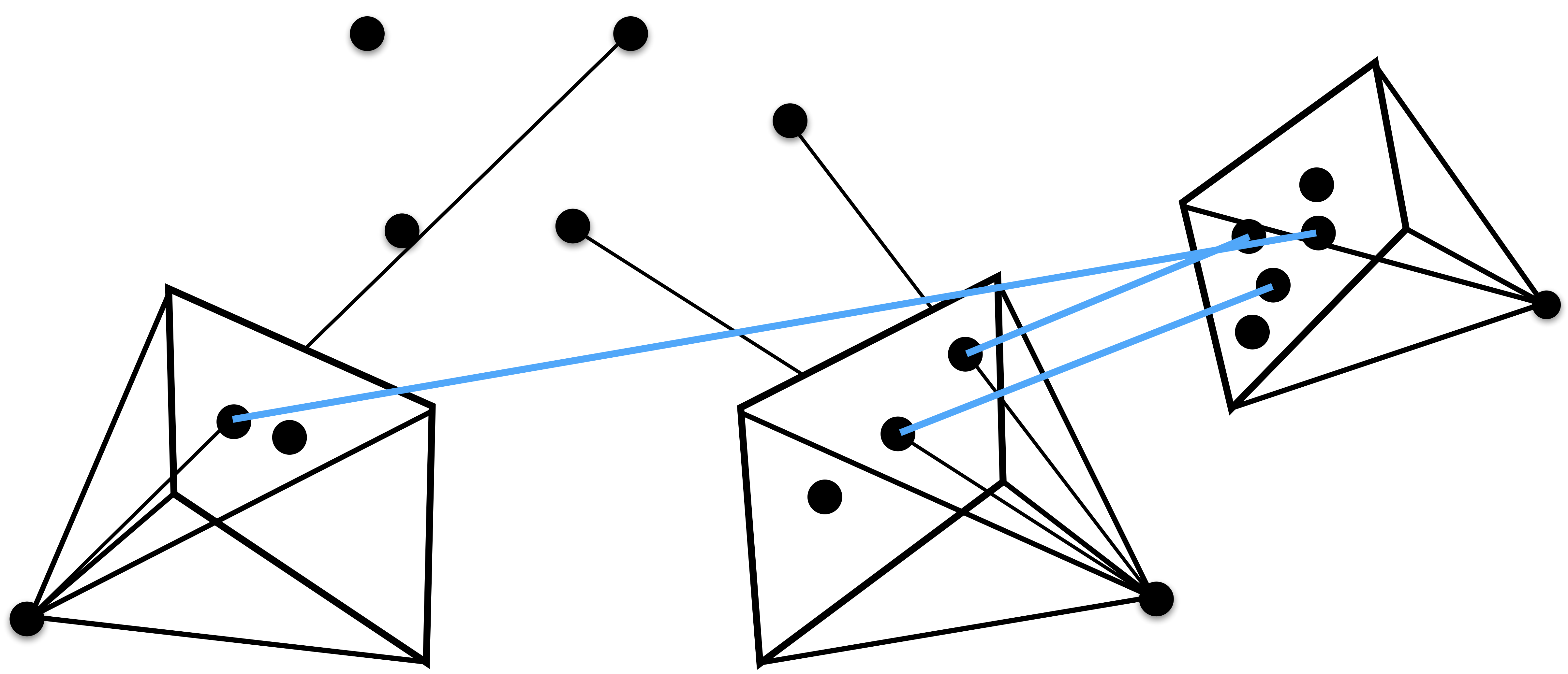
# Sequential / Incremental SfM

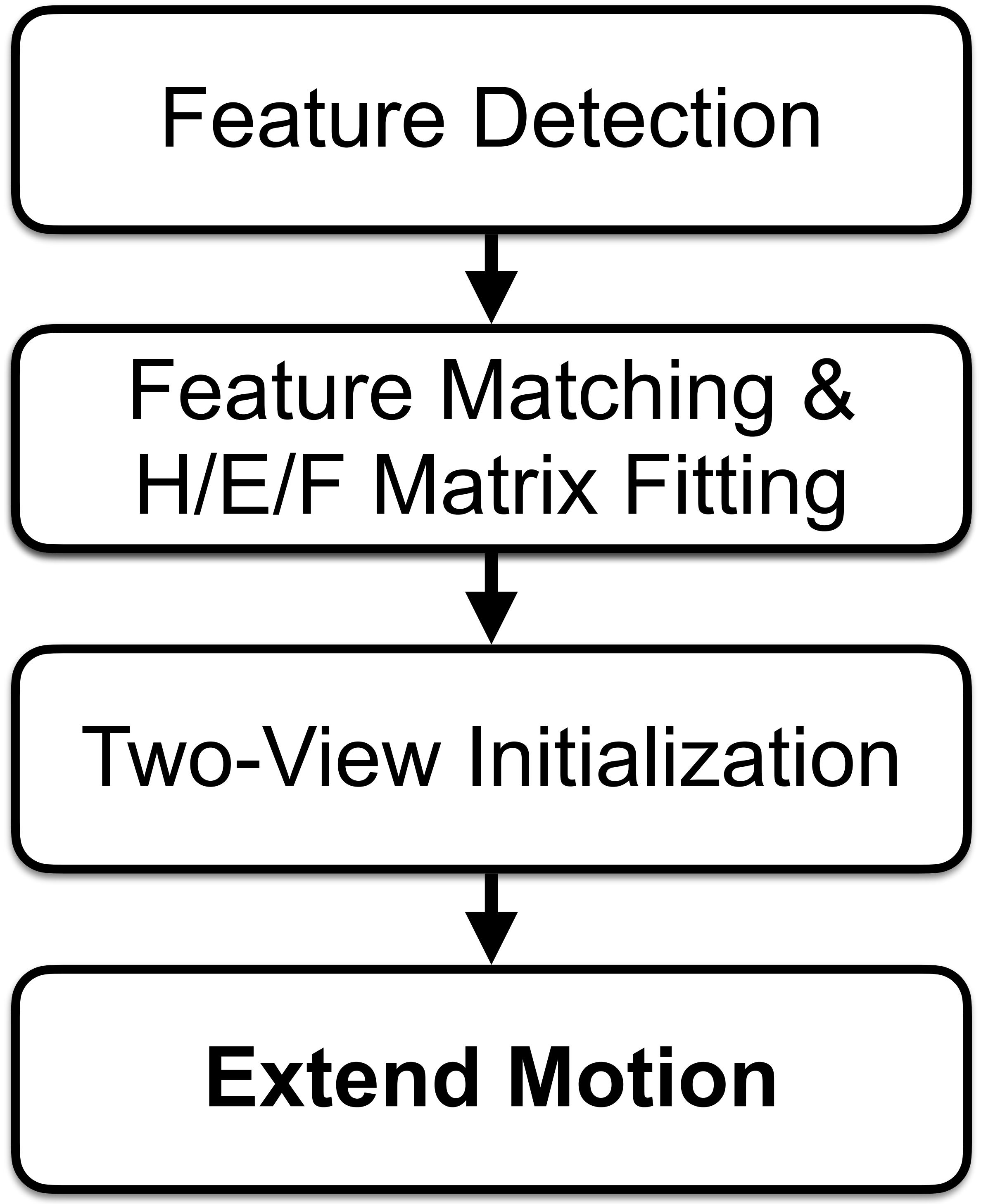





- Pick image(s) with large number of matches to existing cameras
- Obtain 2D-3D matches from 2D-2D matches
- Estimate absolute pose of new image

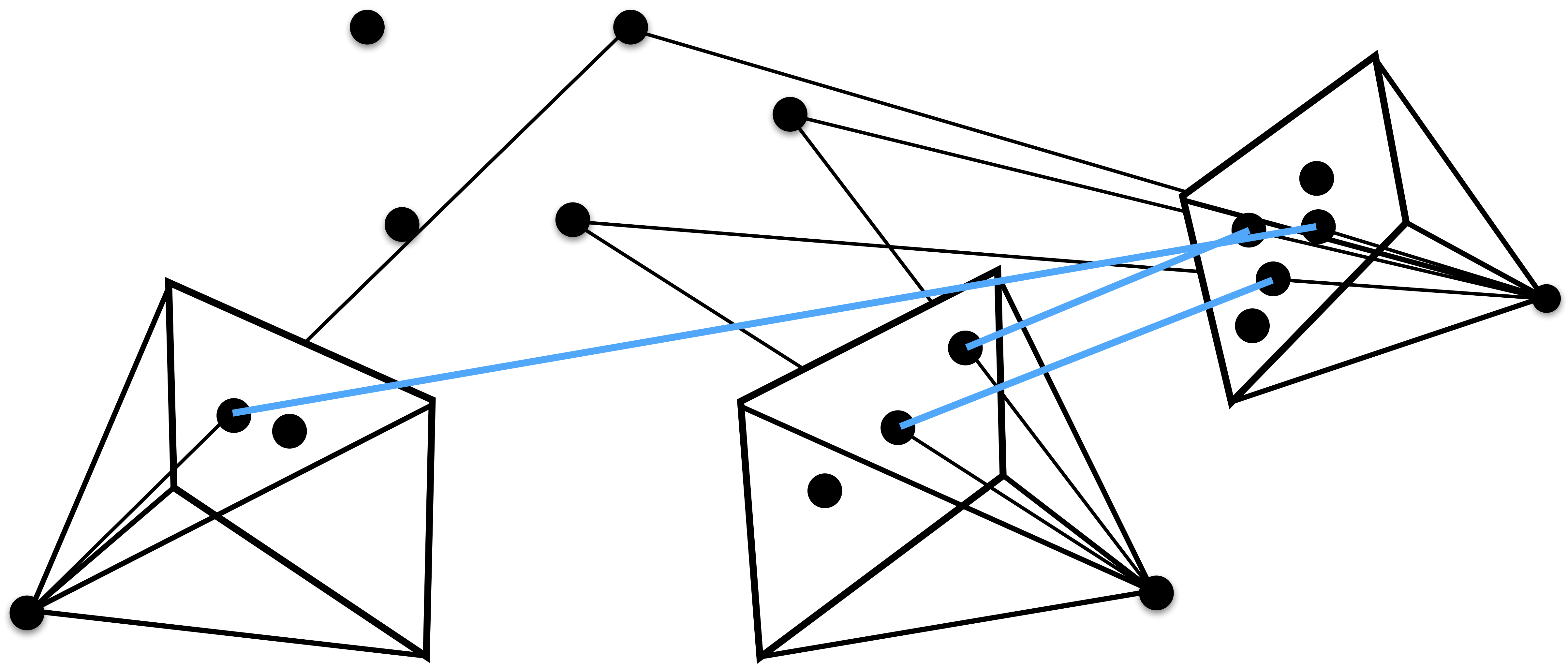
# Sequential / Incremental SfM

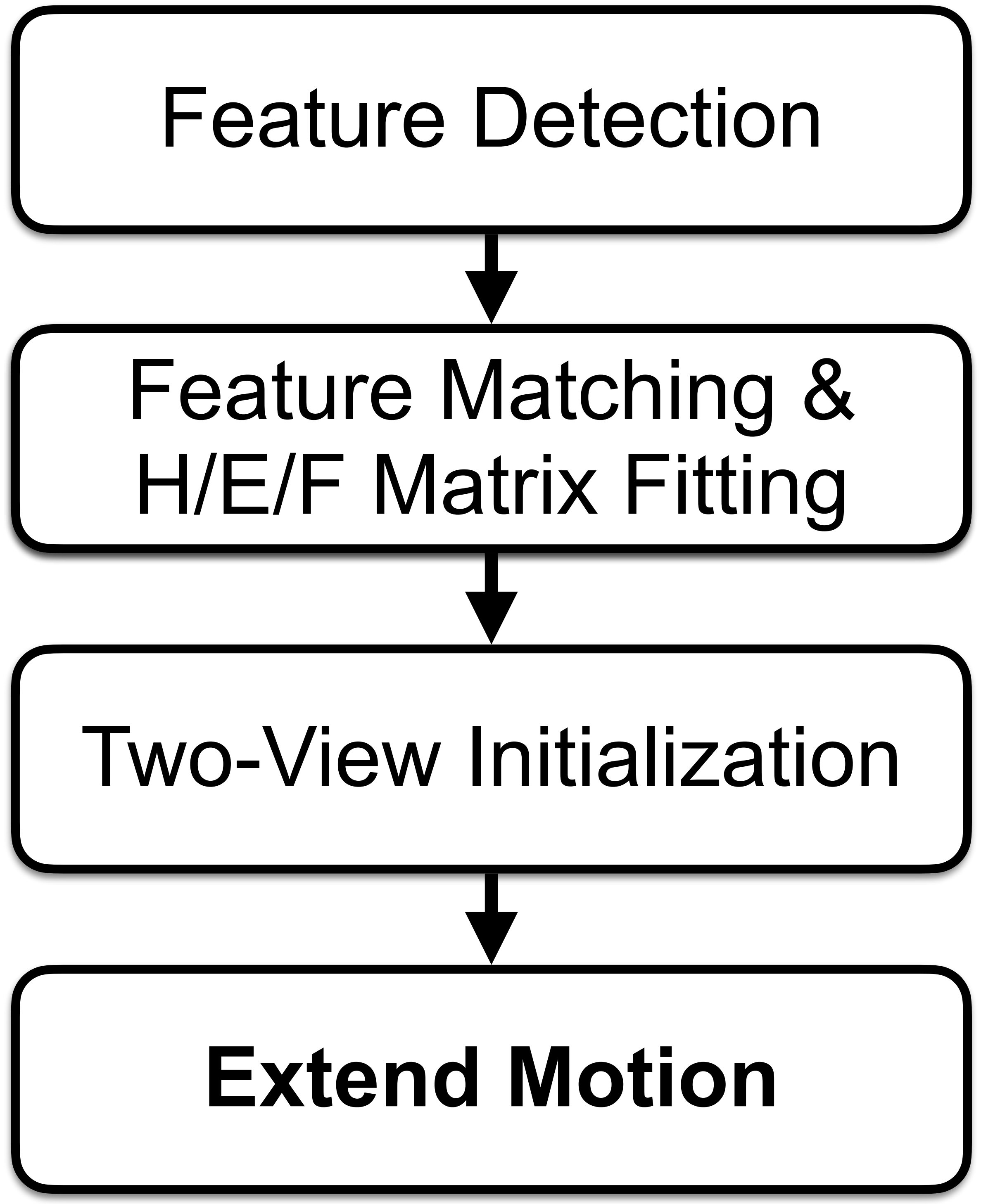





- Pick image(s) with large number of matches to existing cameras
- Obtain 2D-3D matches from 2D-2D matches
- Estimate absolute pose of new image

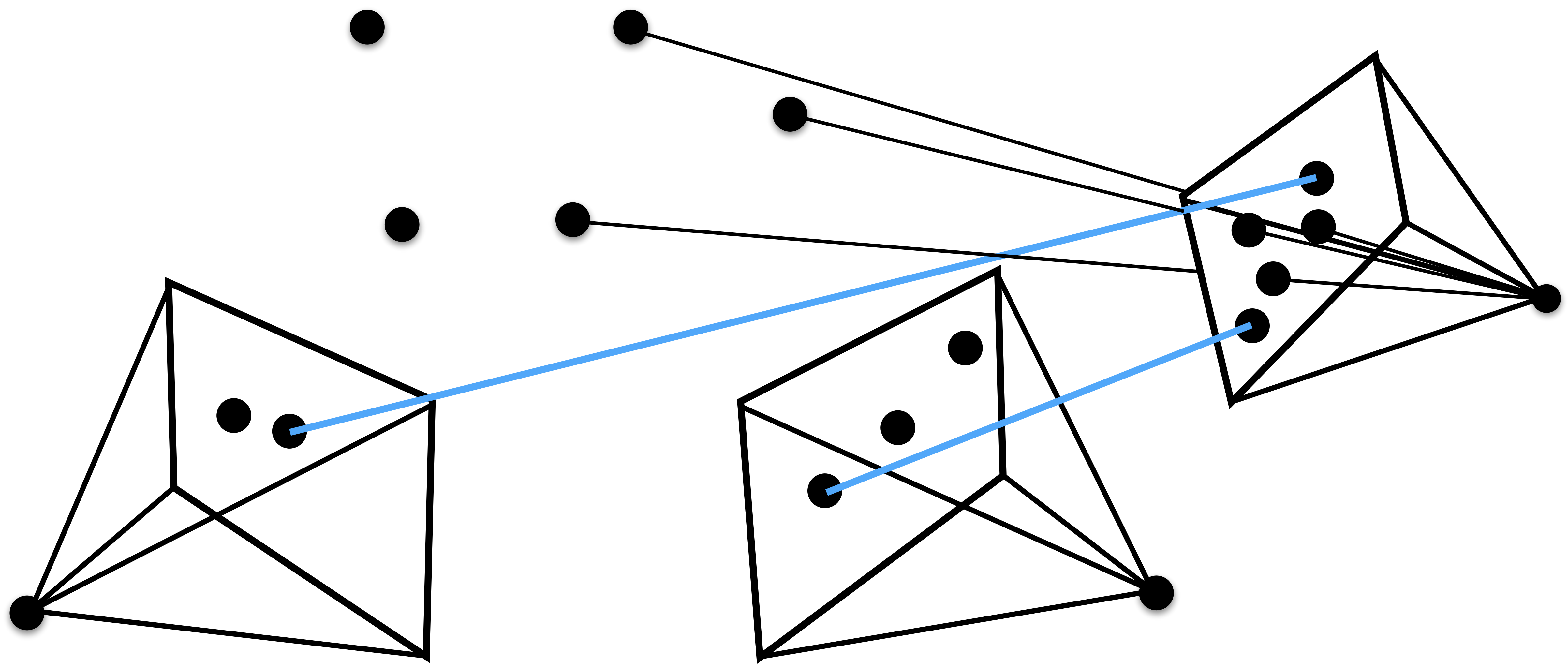
# Sequential / Incremental SfM






- Pick image(s) with large number of matches to existing cameras
- Obtain 2D-3D matches from 2D-2D matches
- Estimate absolute pose of new image

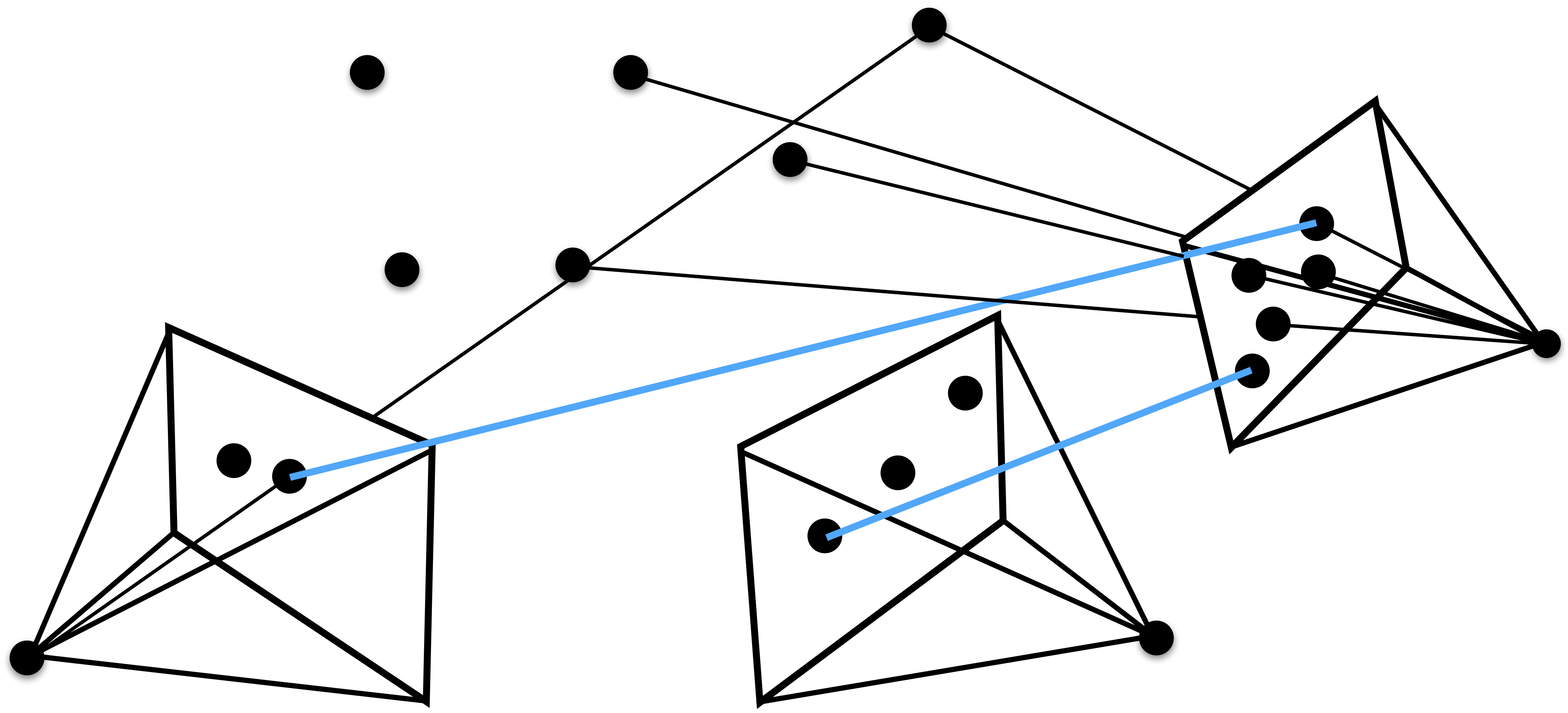
# Sequential / Incremental SfM

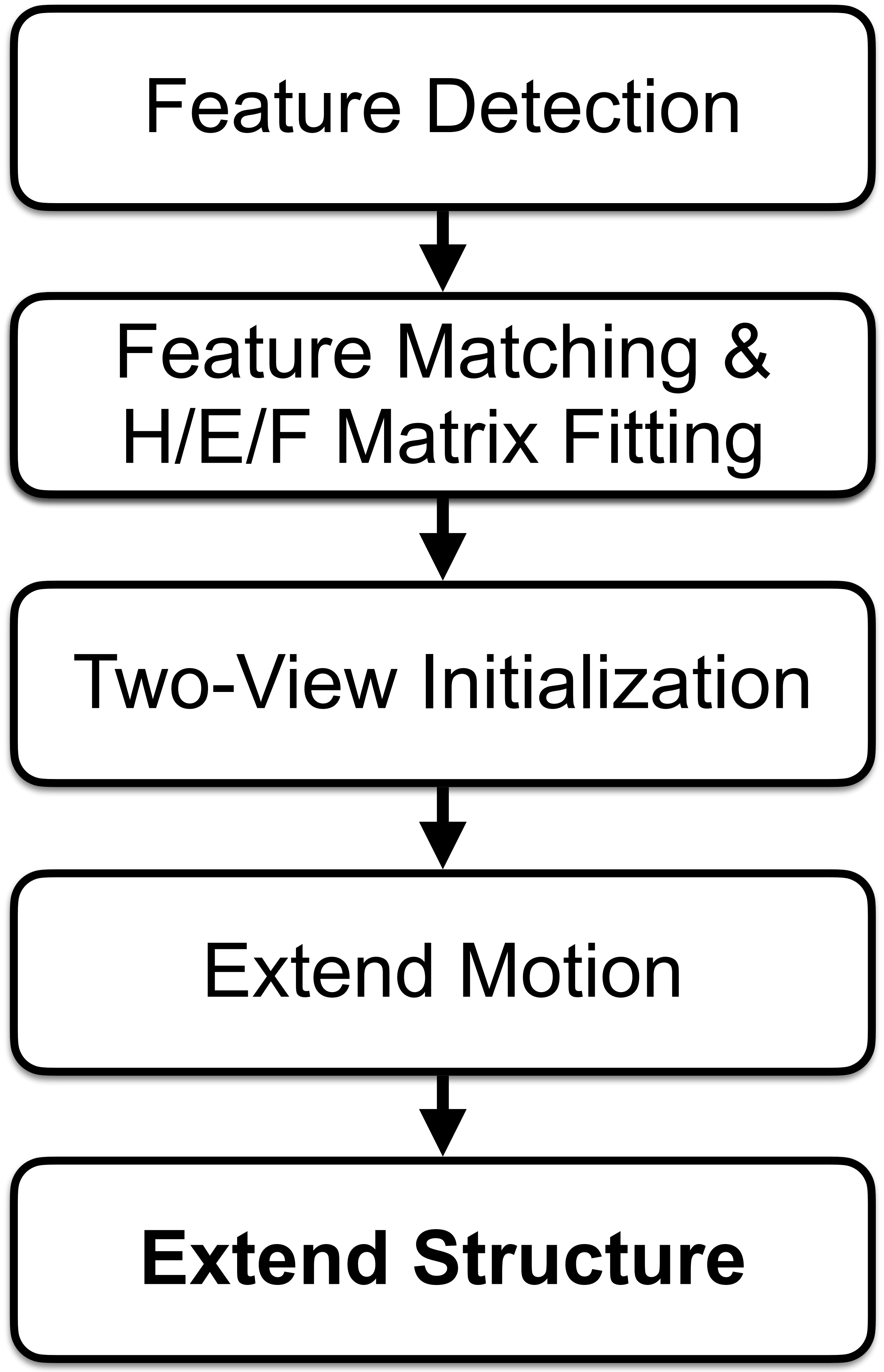


Feature Detection
↓
Feature Matching & H/E/F Matrix Fitting
↓
Two-View Initialization
↓
Extend Motion
↓
**Extend Structure**

- Associate existing 3D points with new features
- Triangulate new 3D points for features without associated 3D points

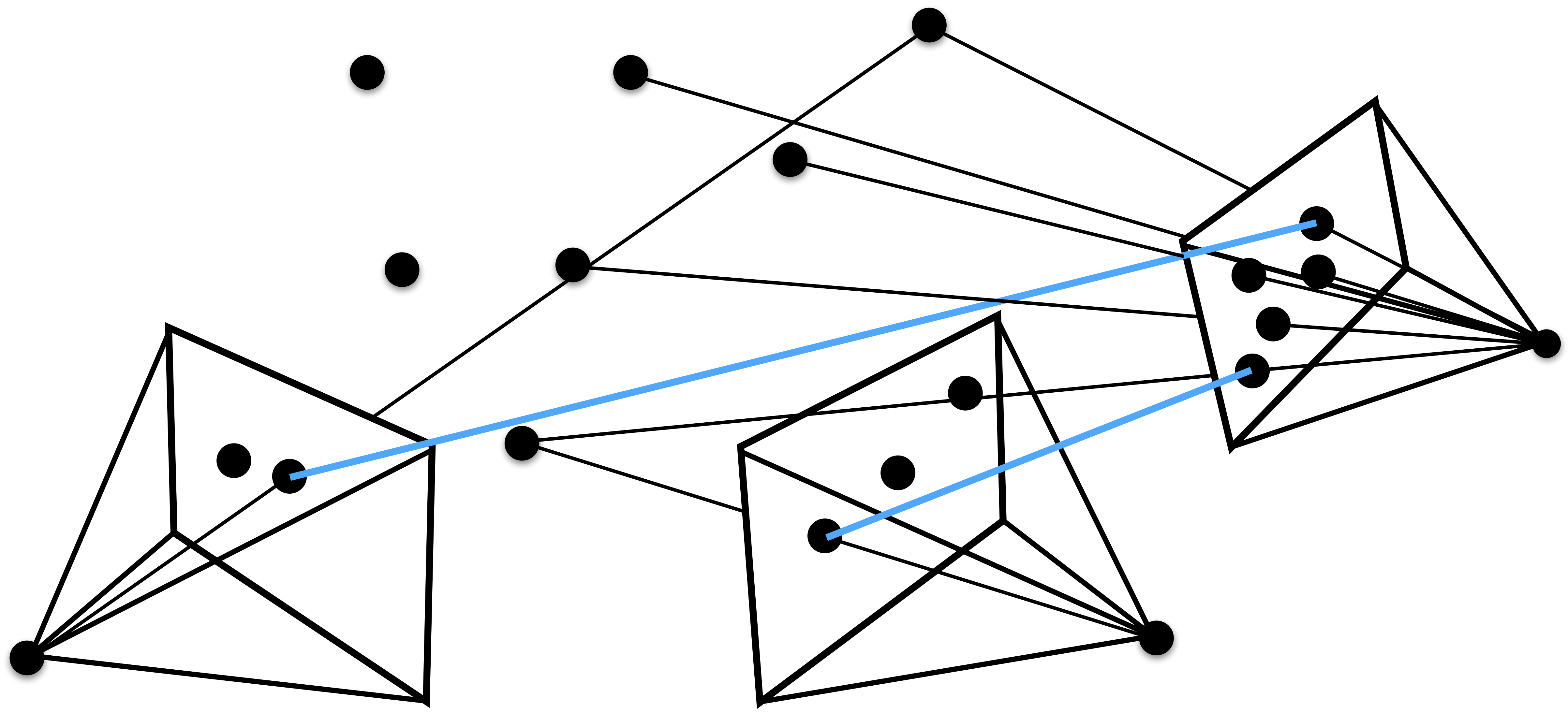
# Sequential / Incremental SfM

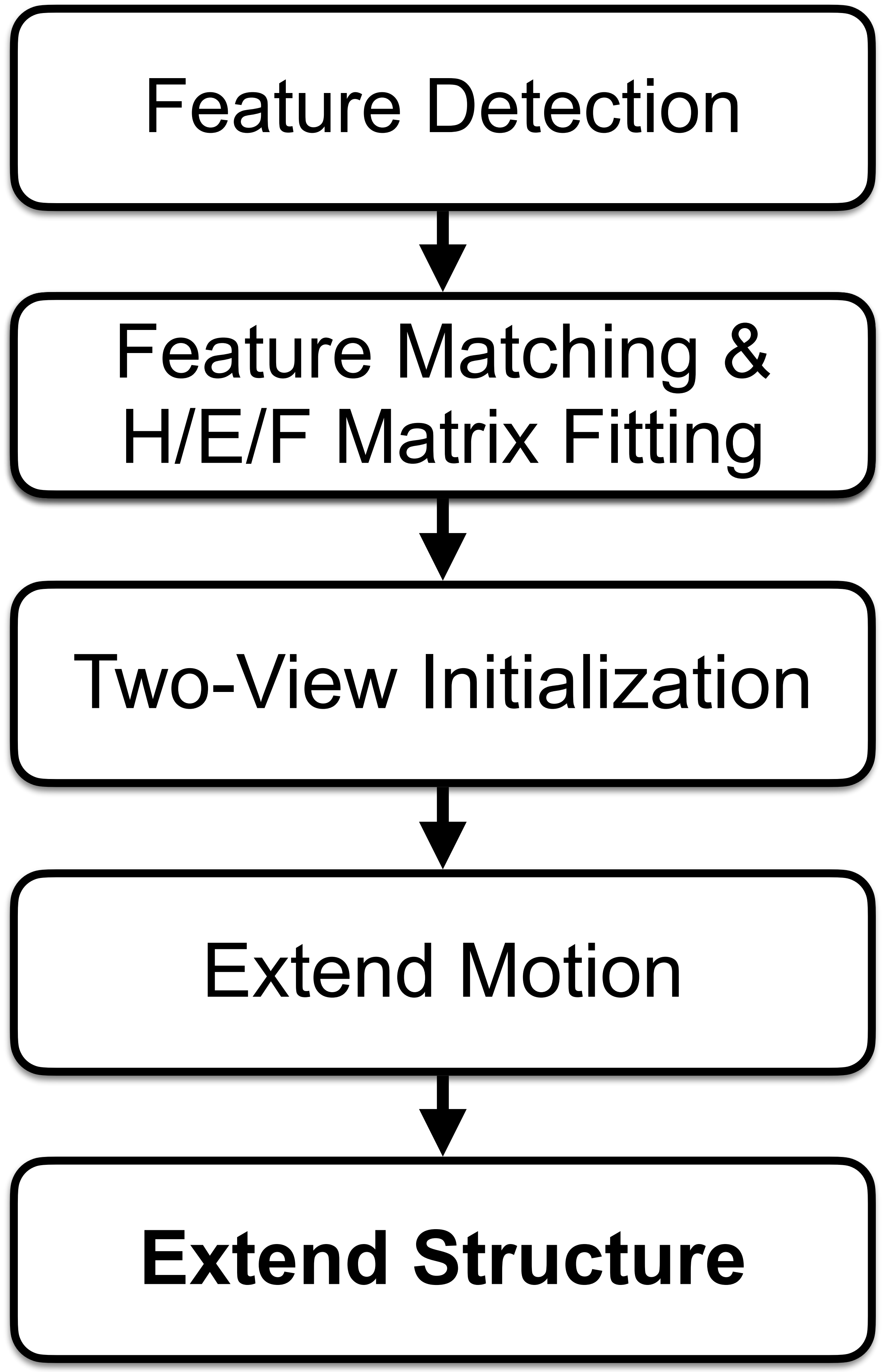





- Associate existing 3D points with new features
- Triangulate new 3D points for features without associated 3D points

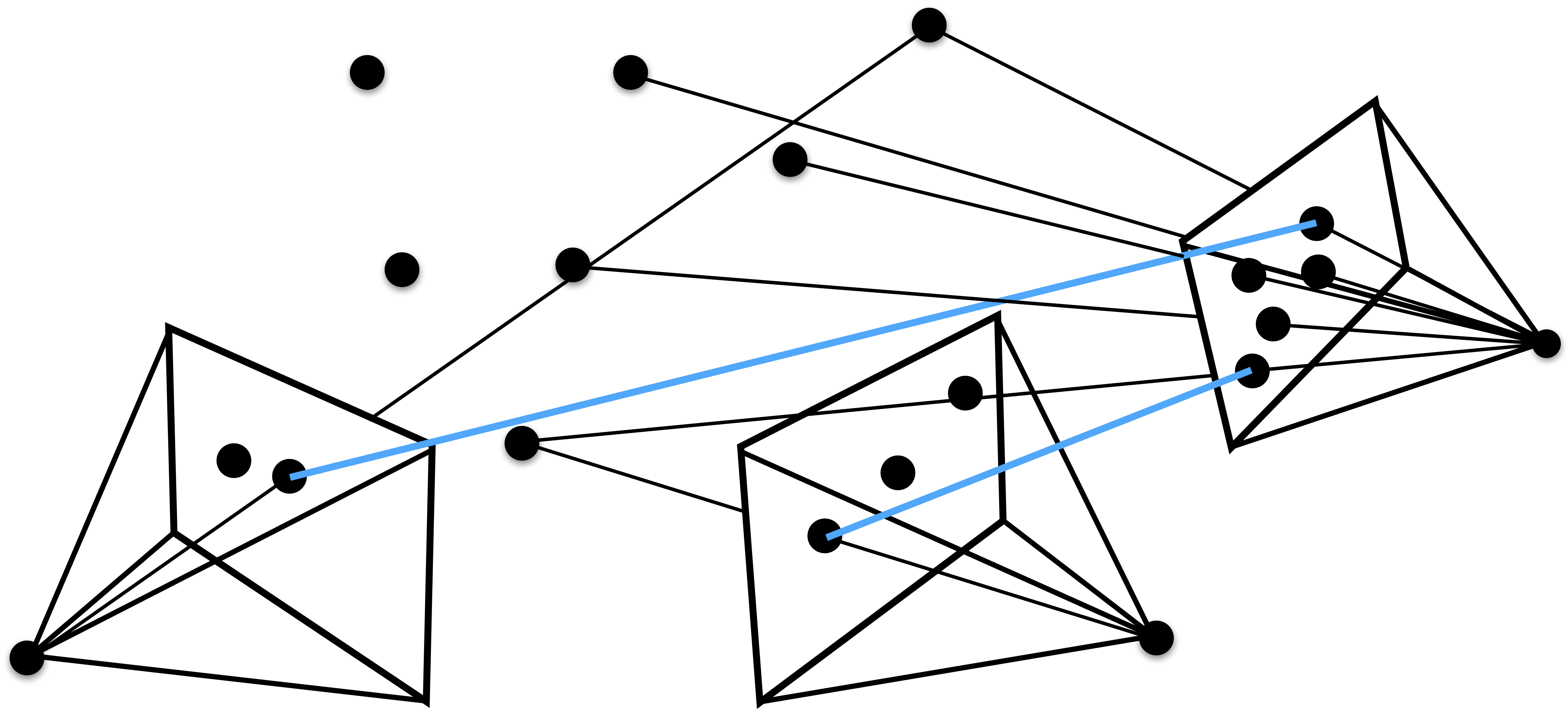
# Sequential / Incremental SfM

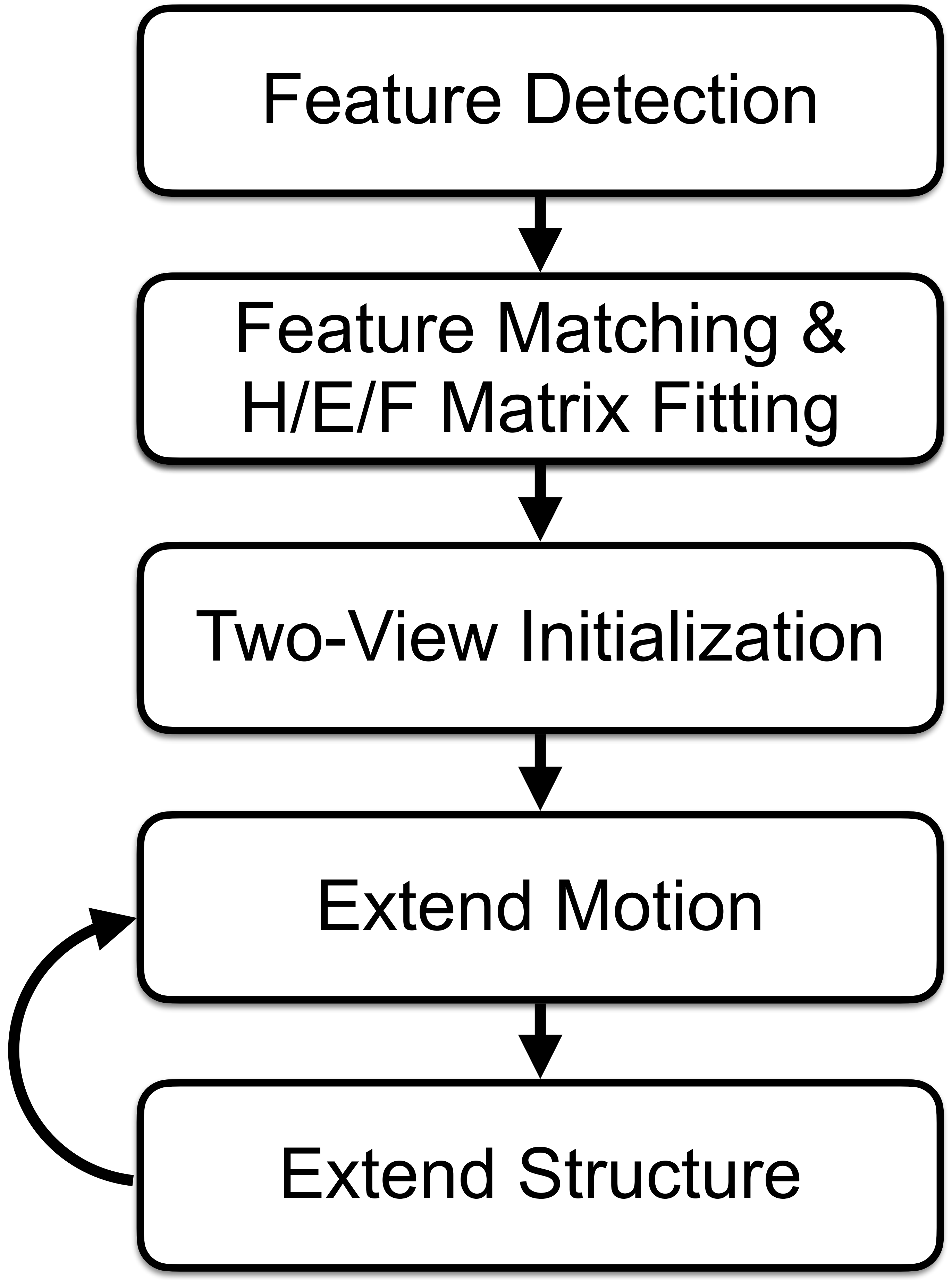





- Associate existing 3D points with new features
- Triangulate new 3D points for features without associated 3D points

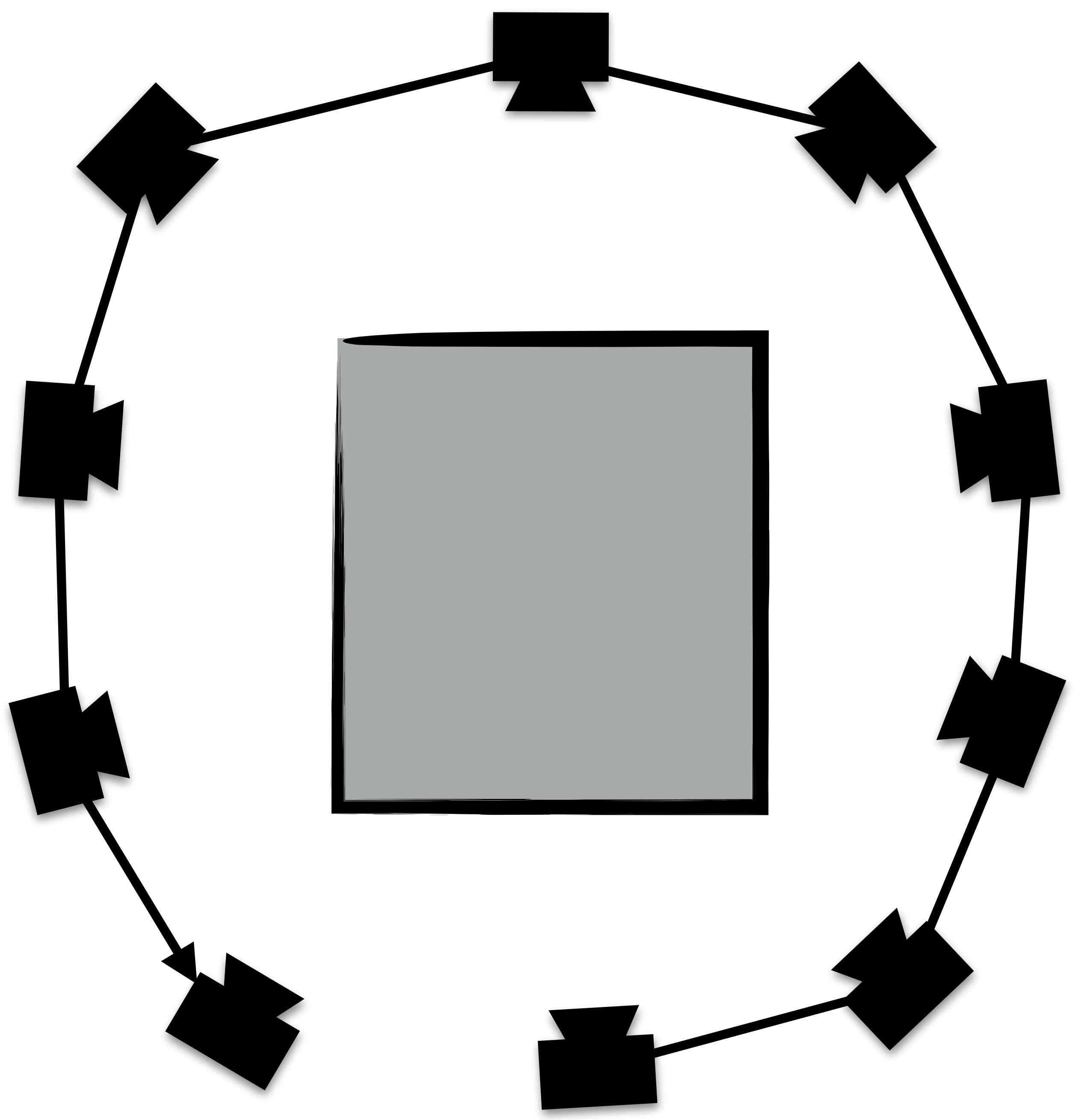
# Sequential / Incremental SfM



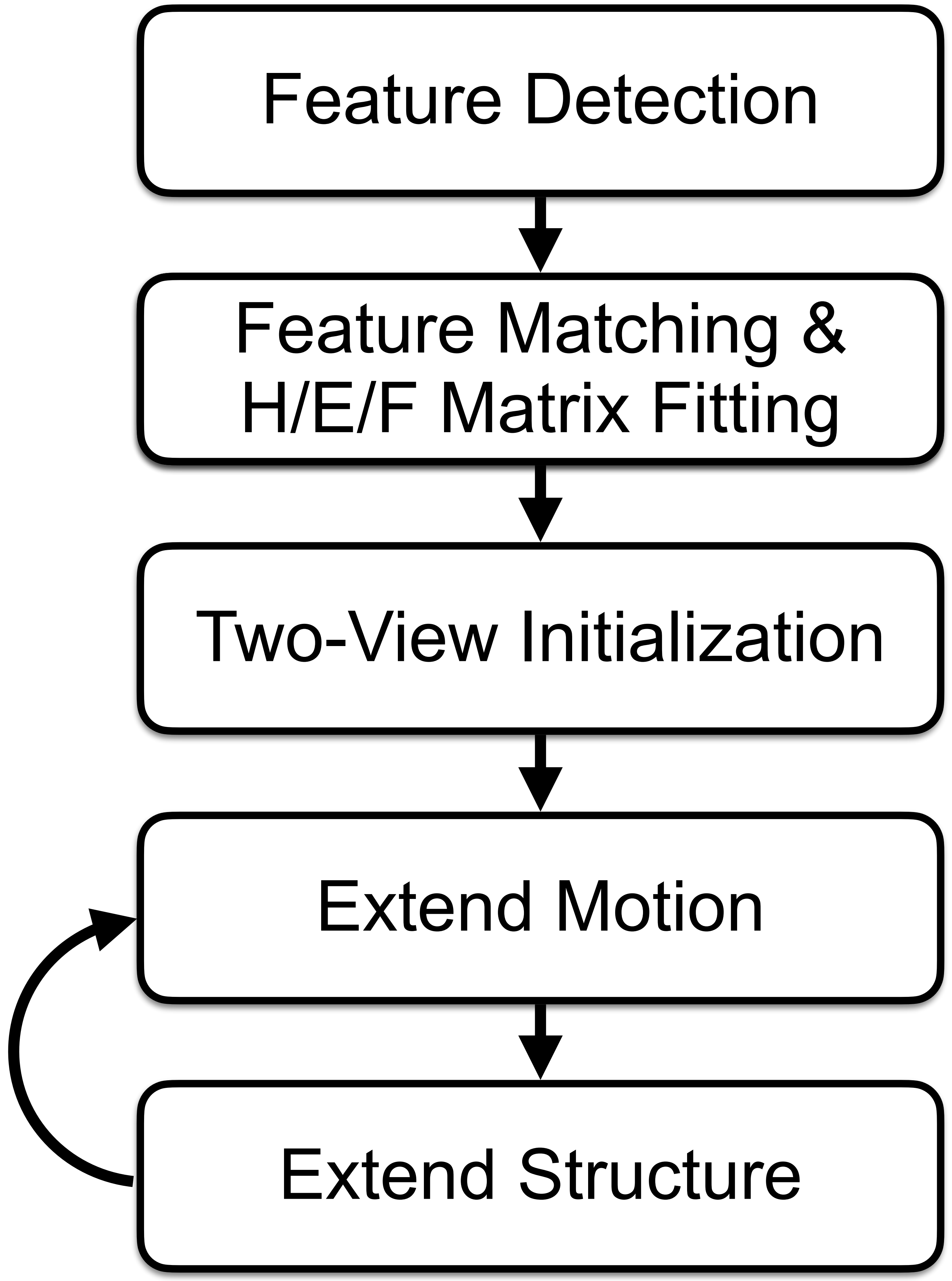



- Associate existing 3D points with new features
- Triangulate new 3D points for features without associated 3D points

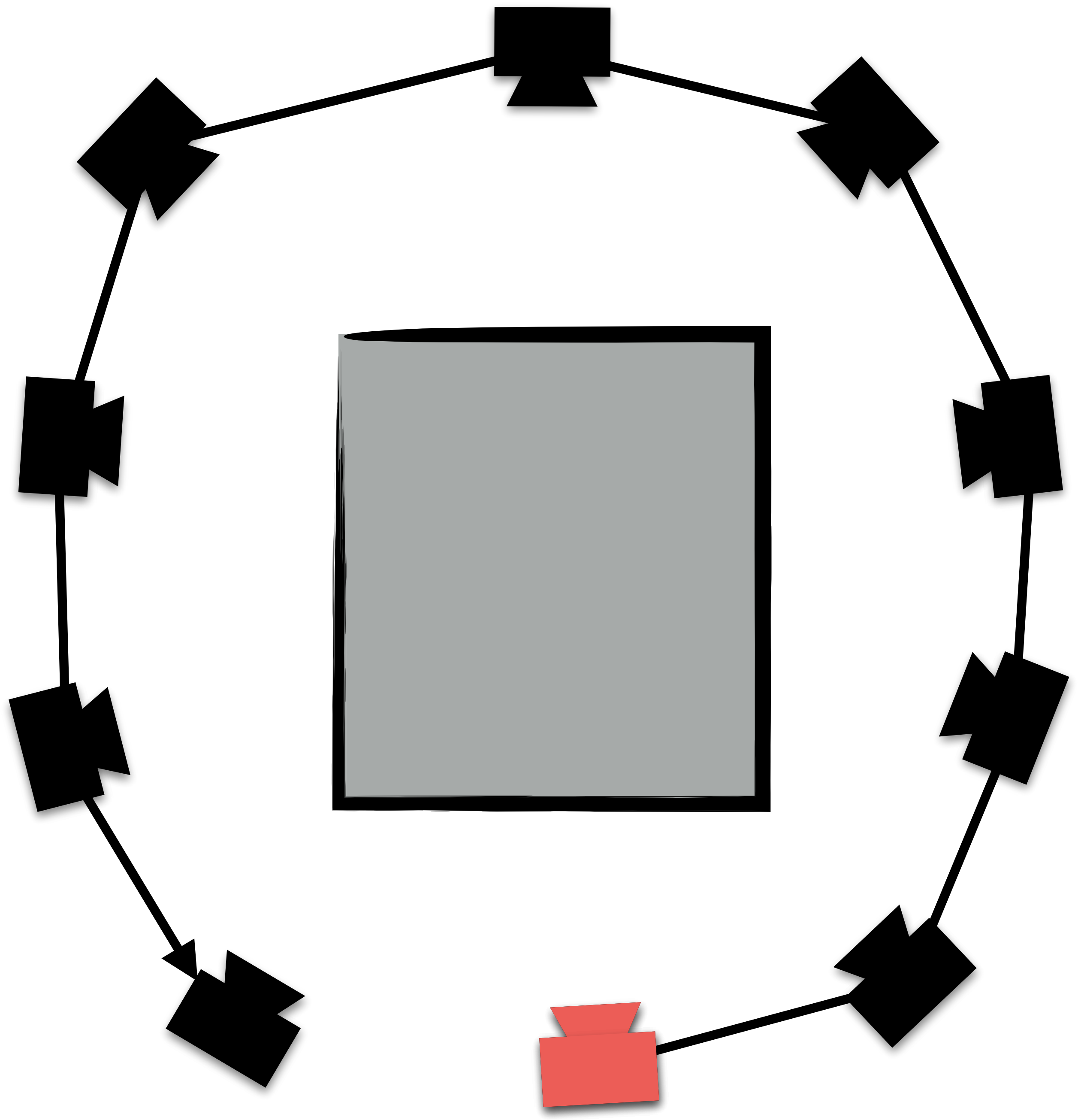
# Sequential / Incremental SfM



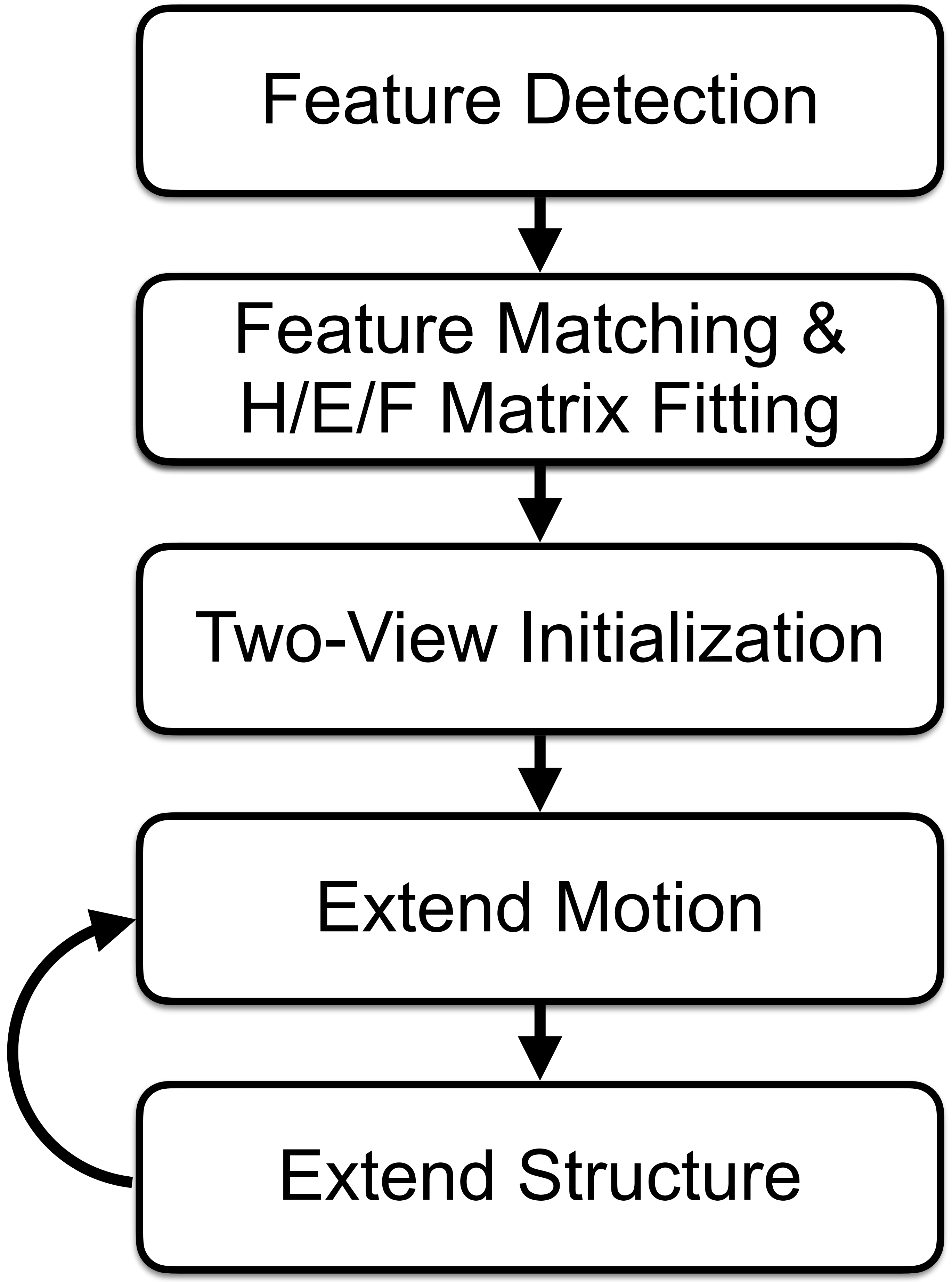
True trajectory



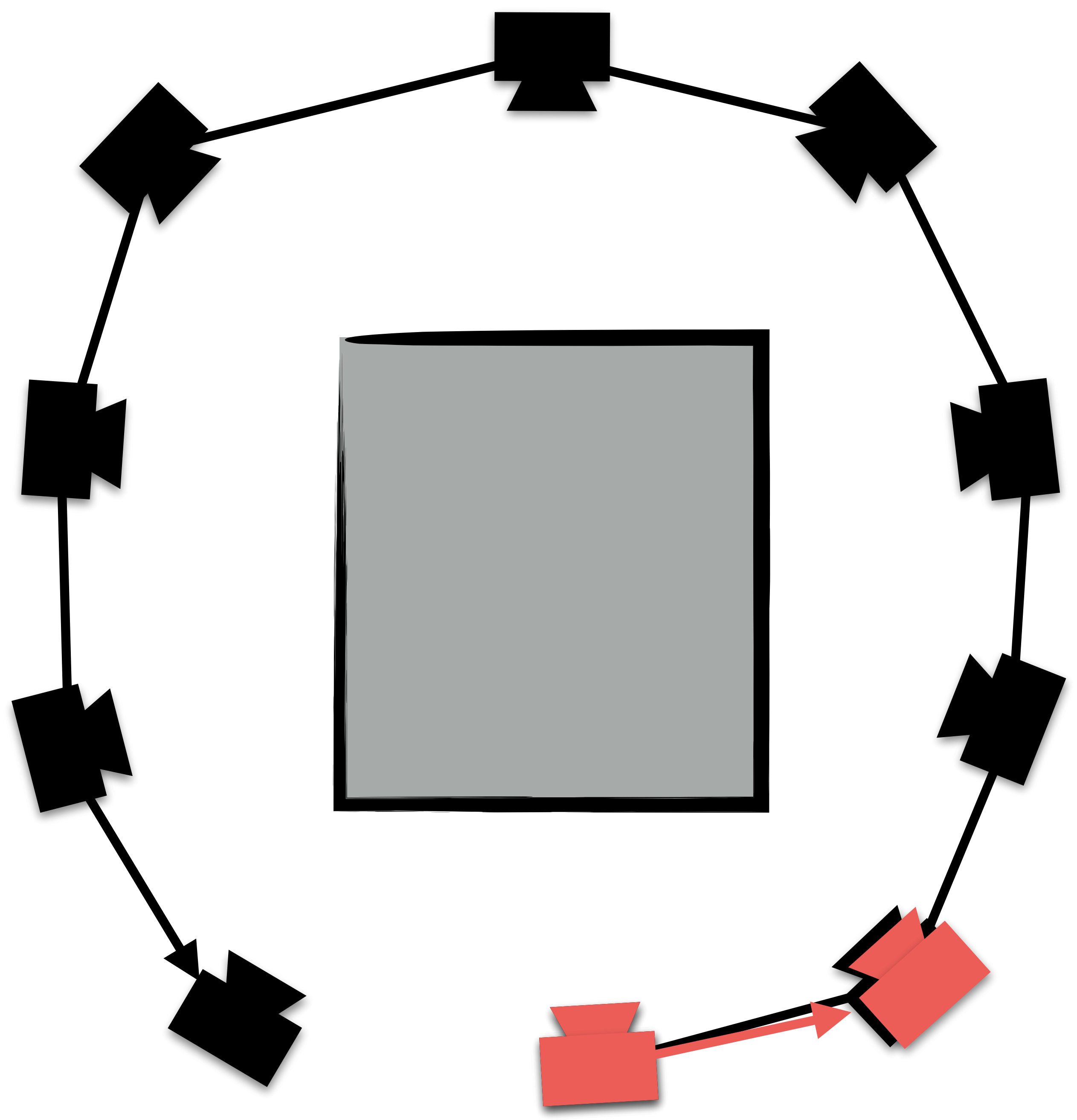
# Sequential / Incremental SfM



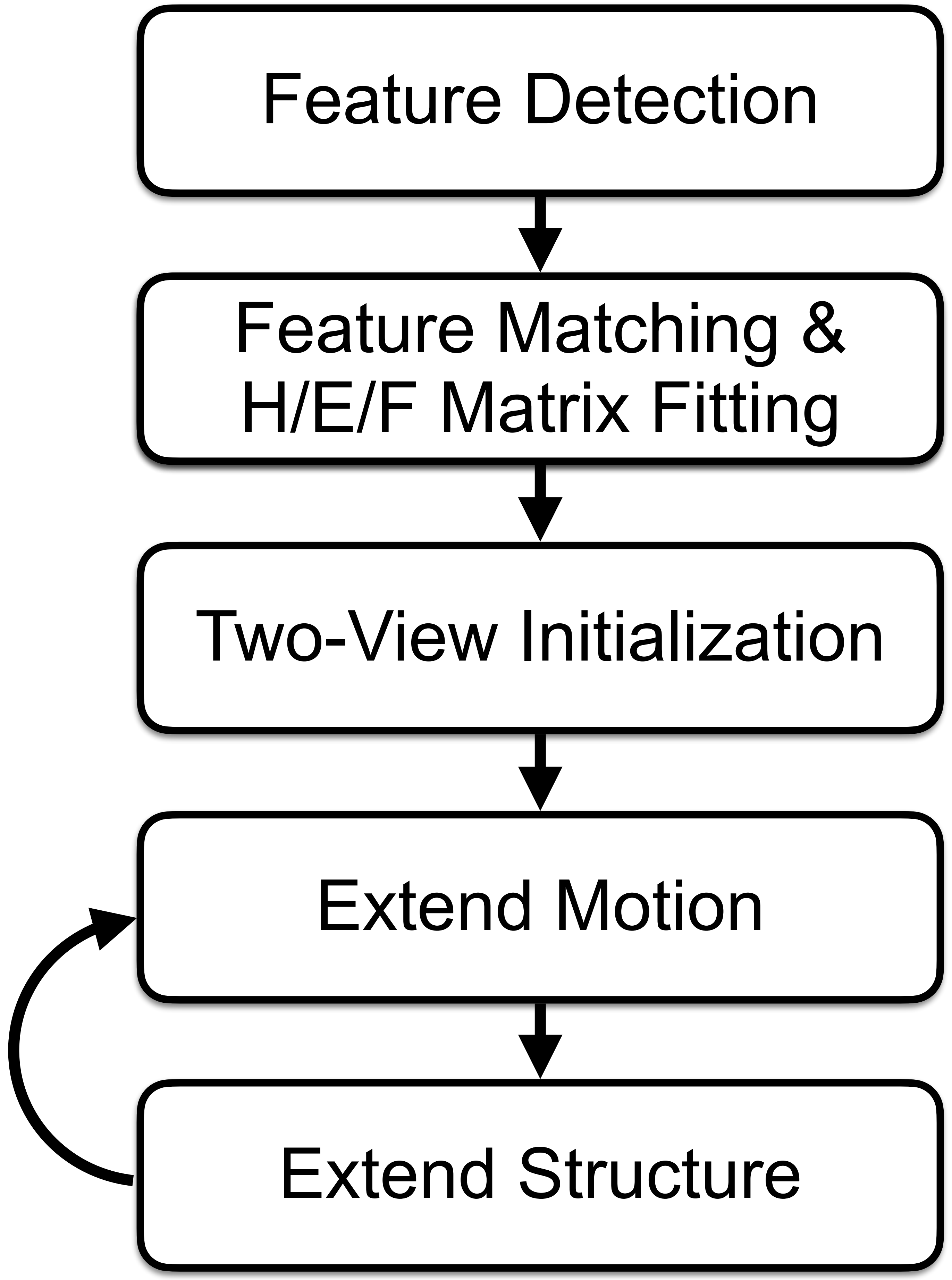
True trajectory
Estimated trajectory



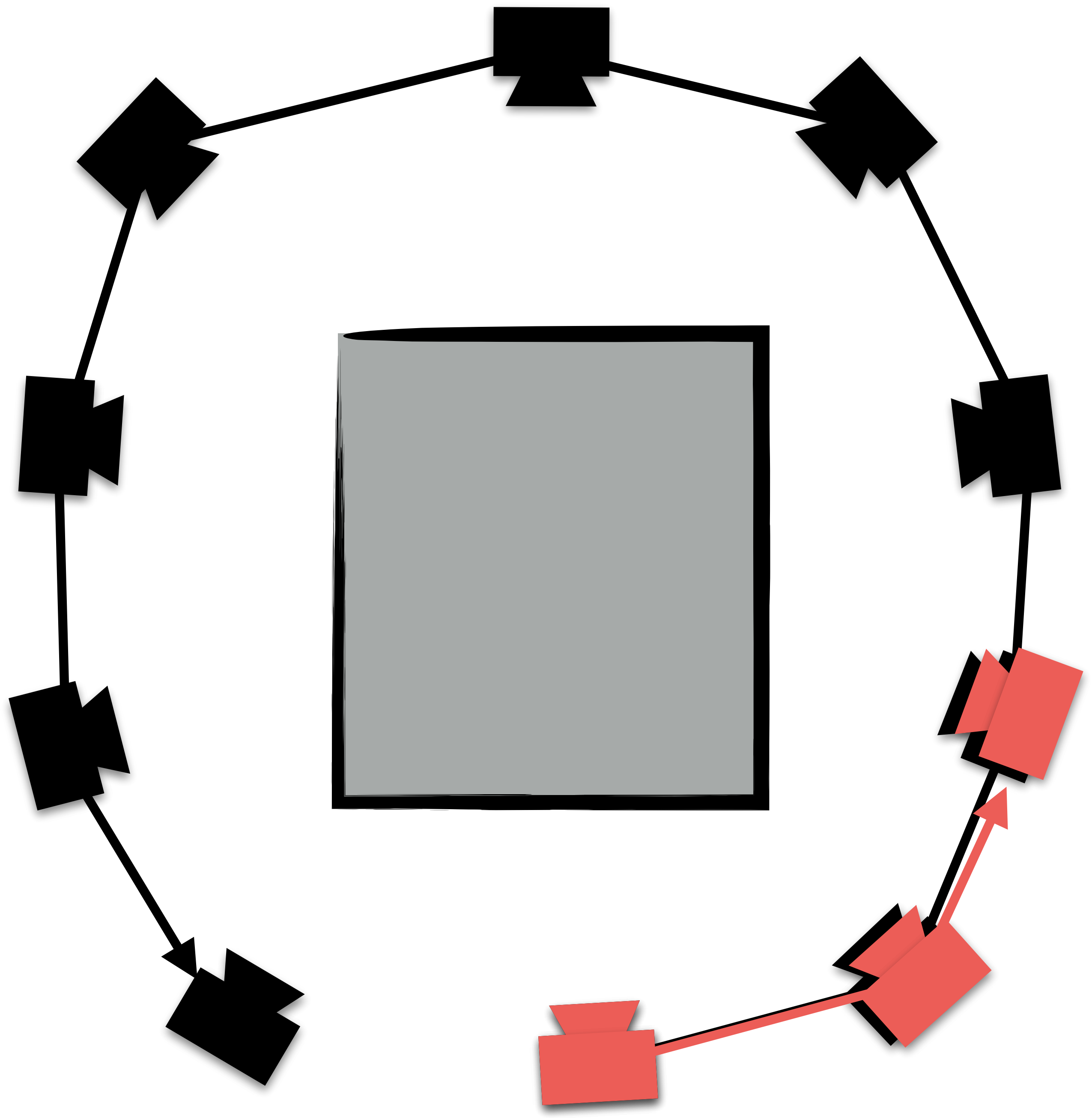
# Sequential / Incremental SfM



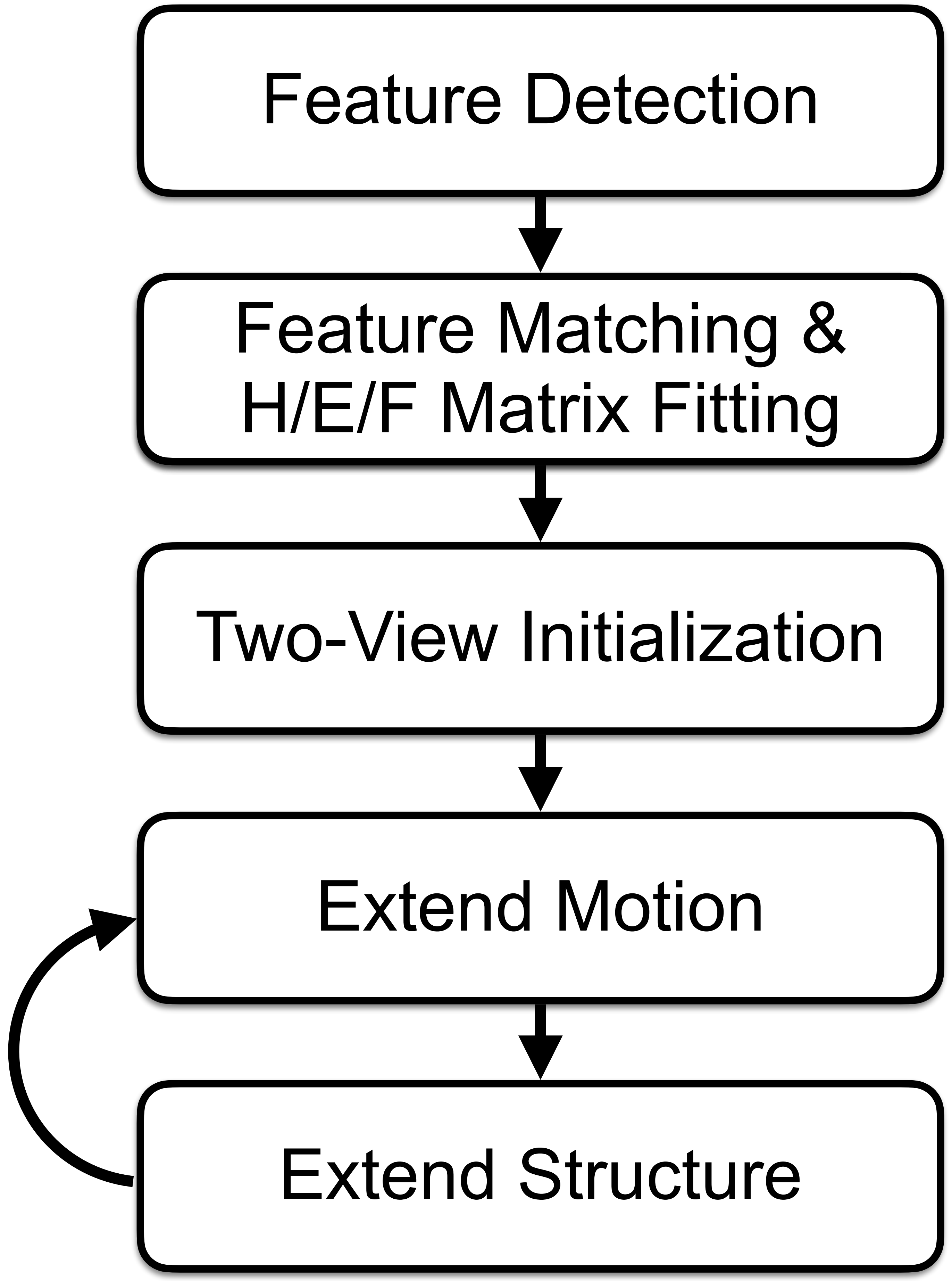
True trajectory
Estimated trajectory



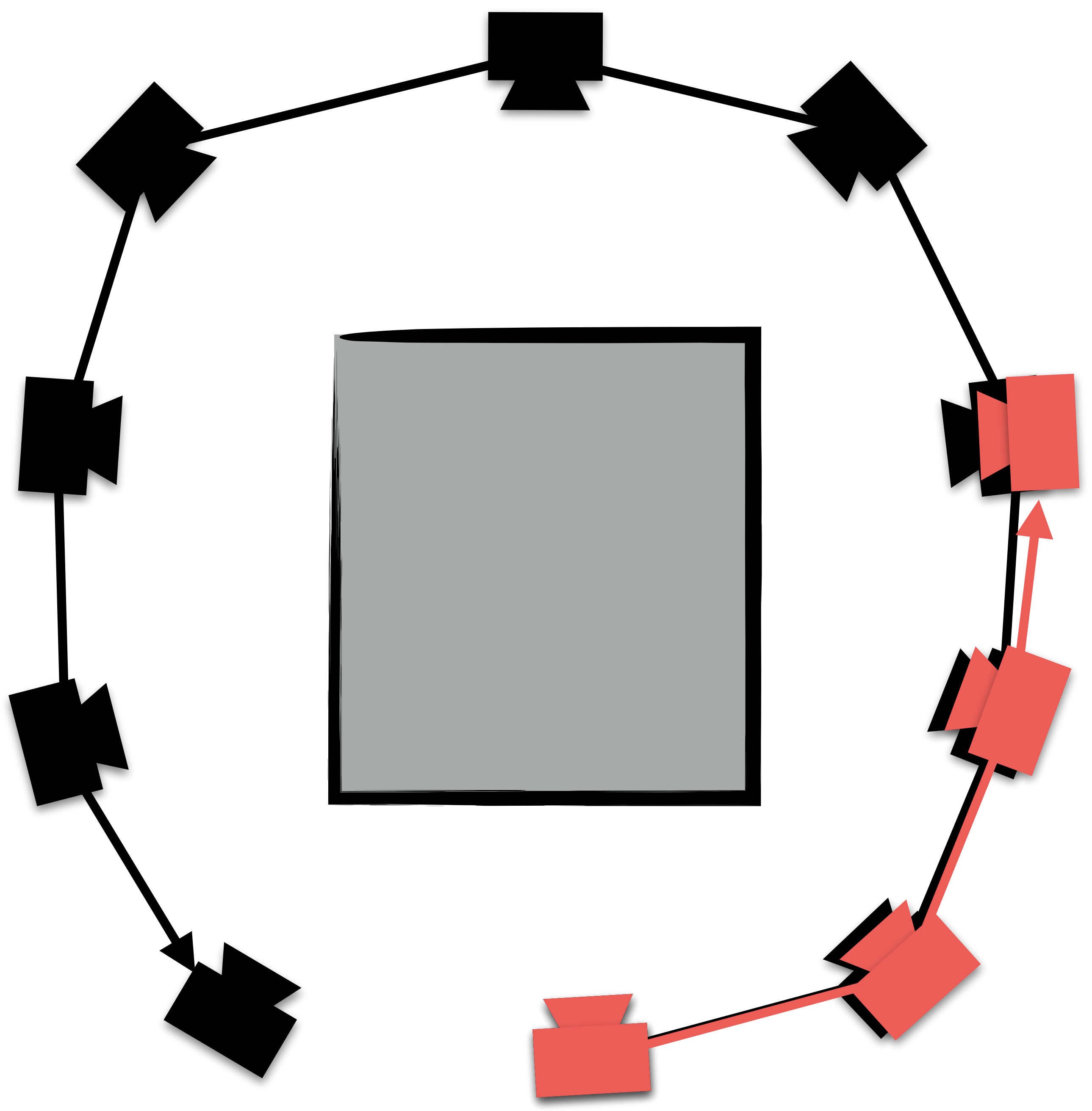
# Sequential / Incremental SfM



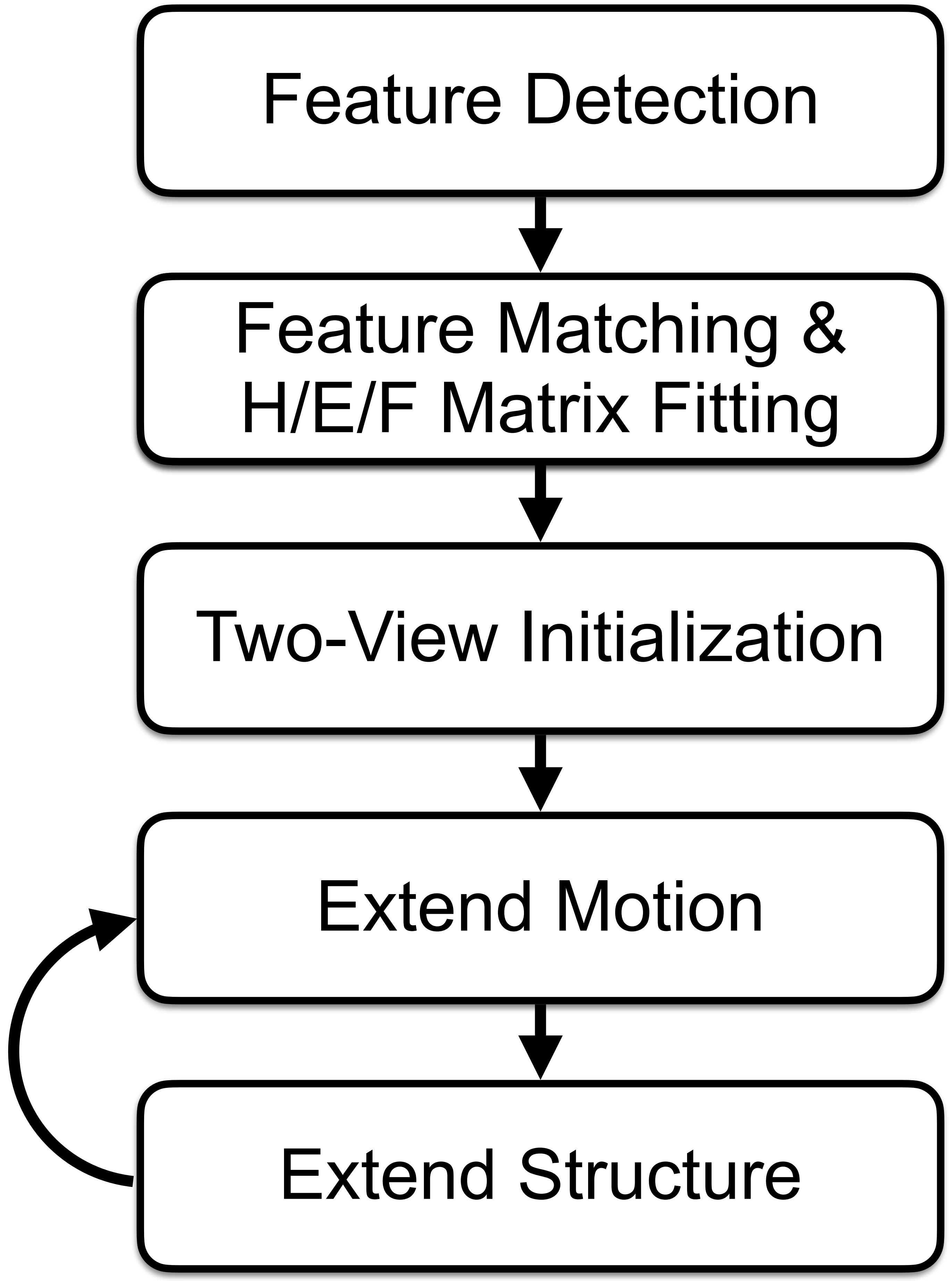
True trajectory
Estimated trajectory



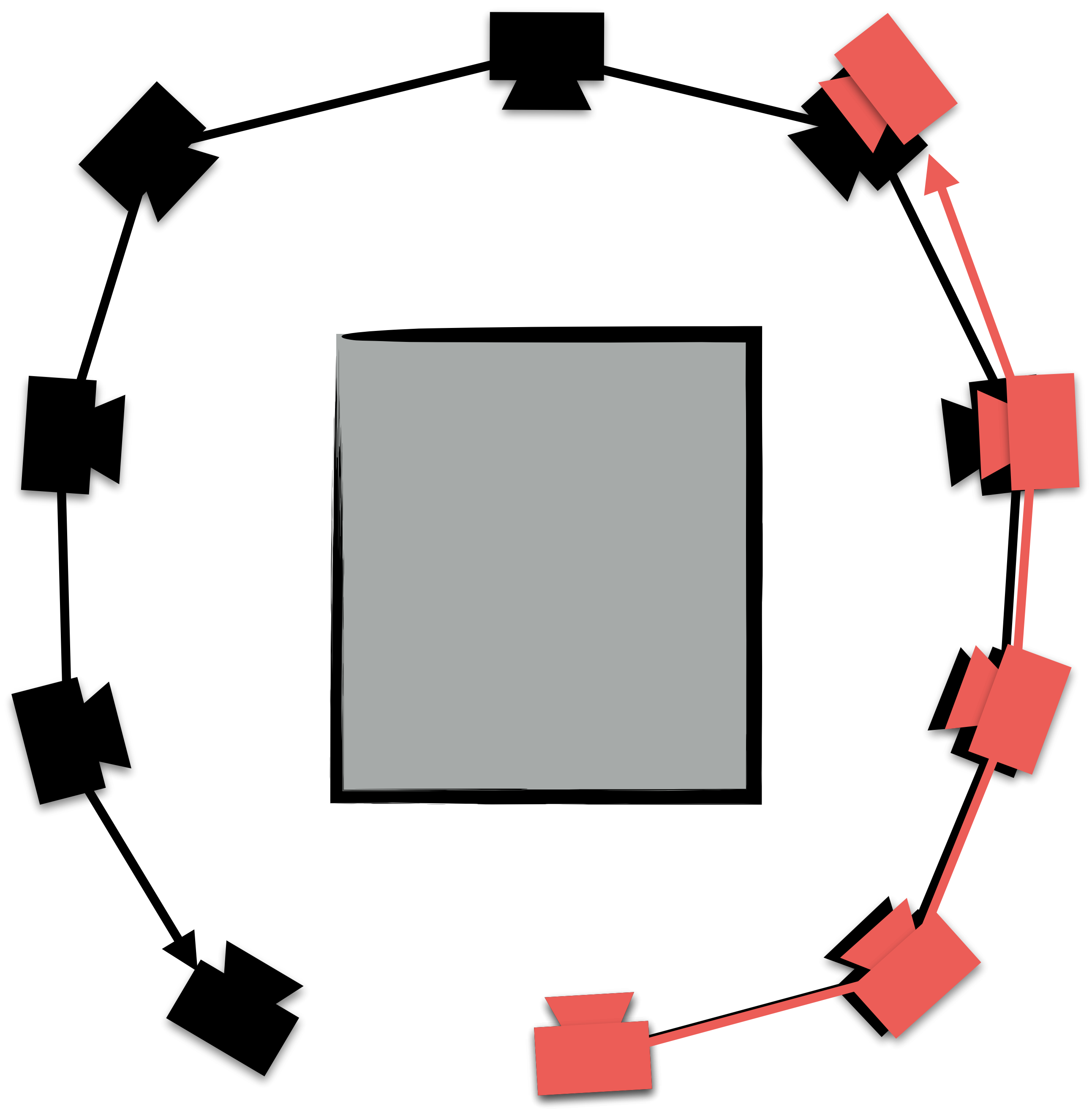
# Sequential / Incremental SfM



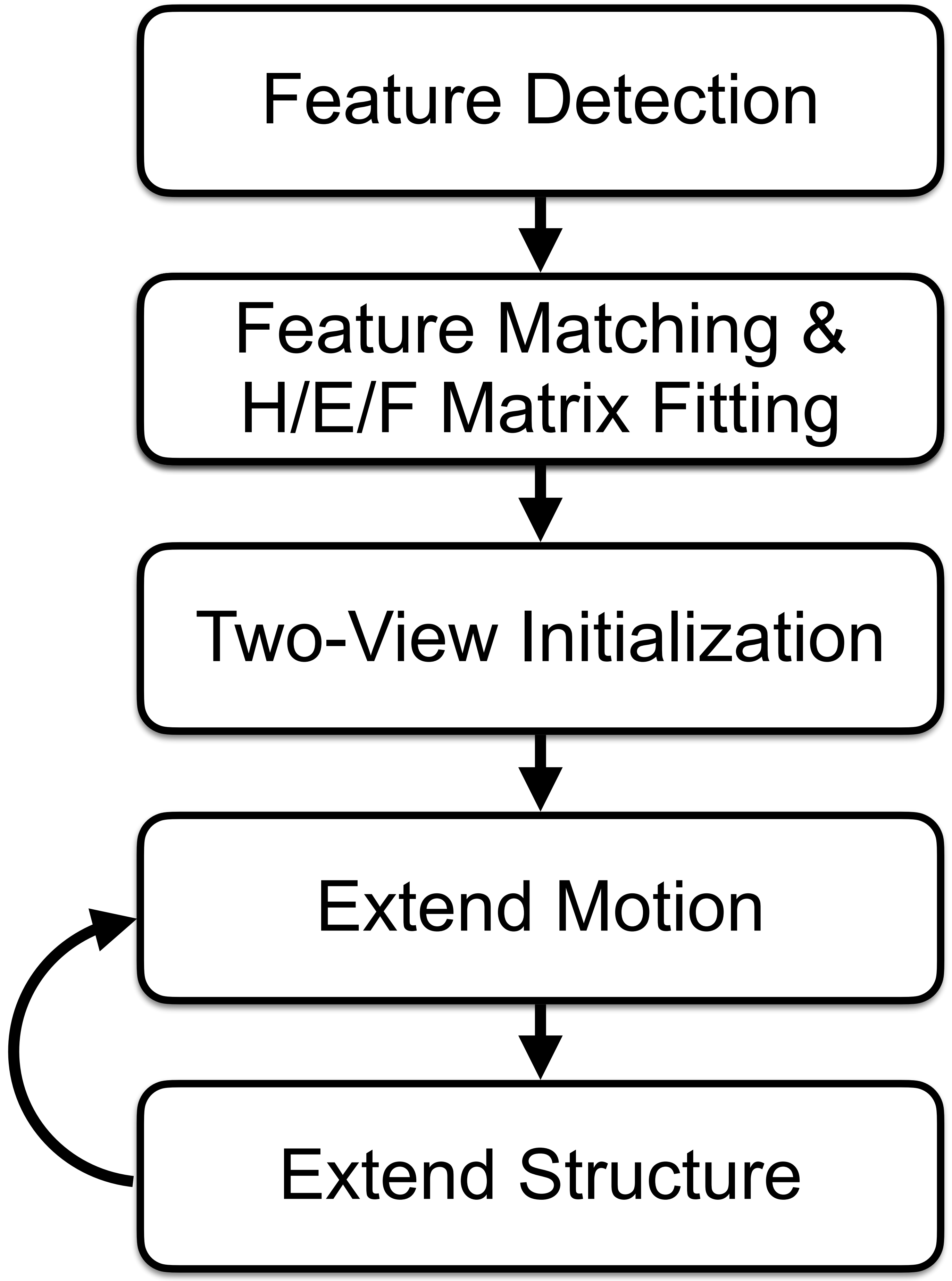
True trajectory
Estimated trajectory



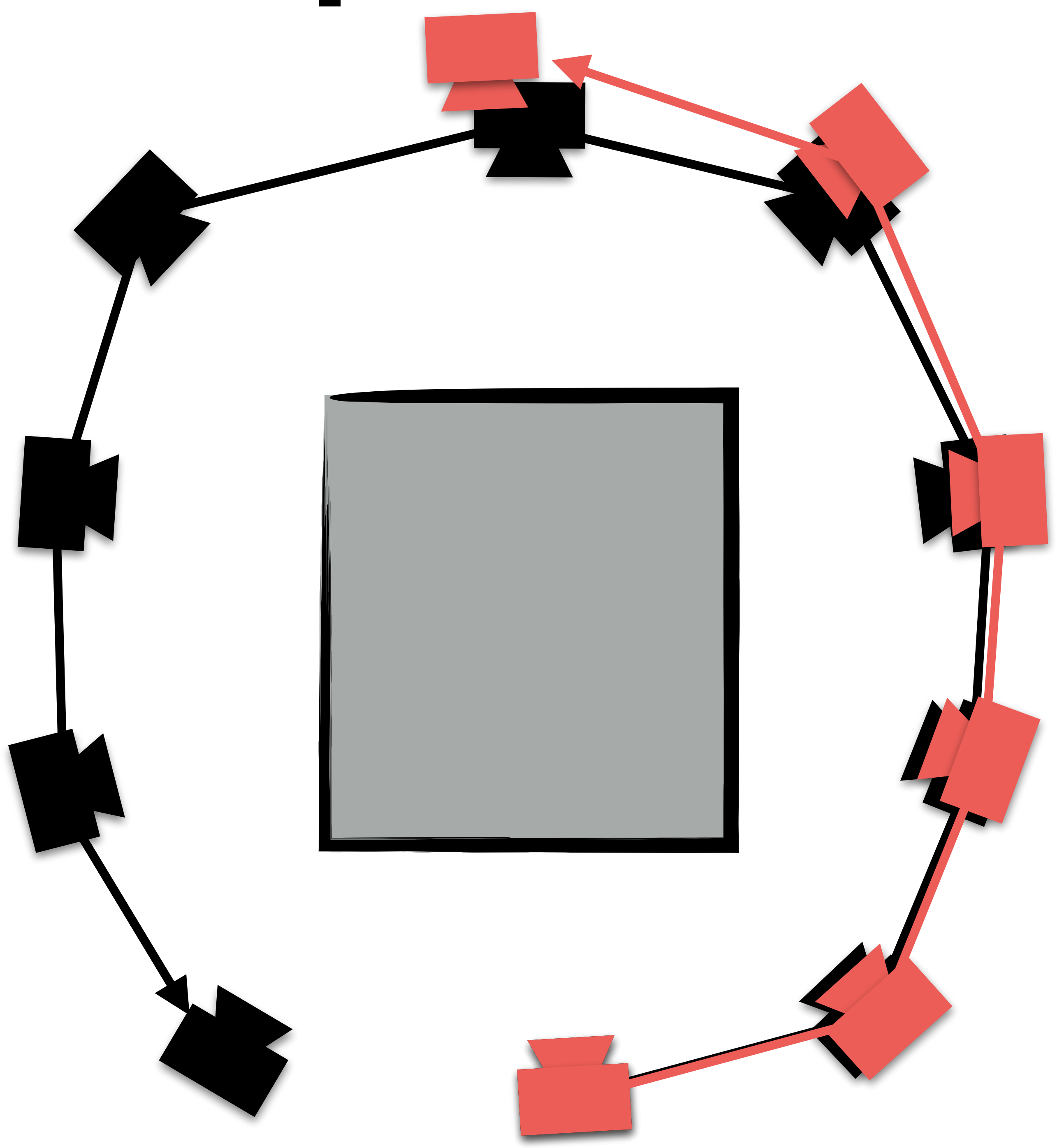
# Sequential / Incremental SfM



True trajectory
Estimated trajectory

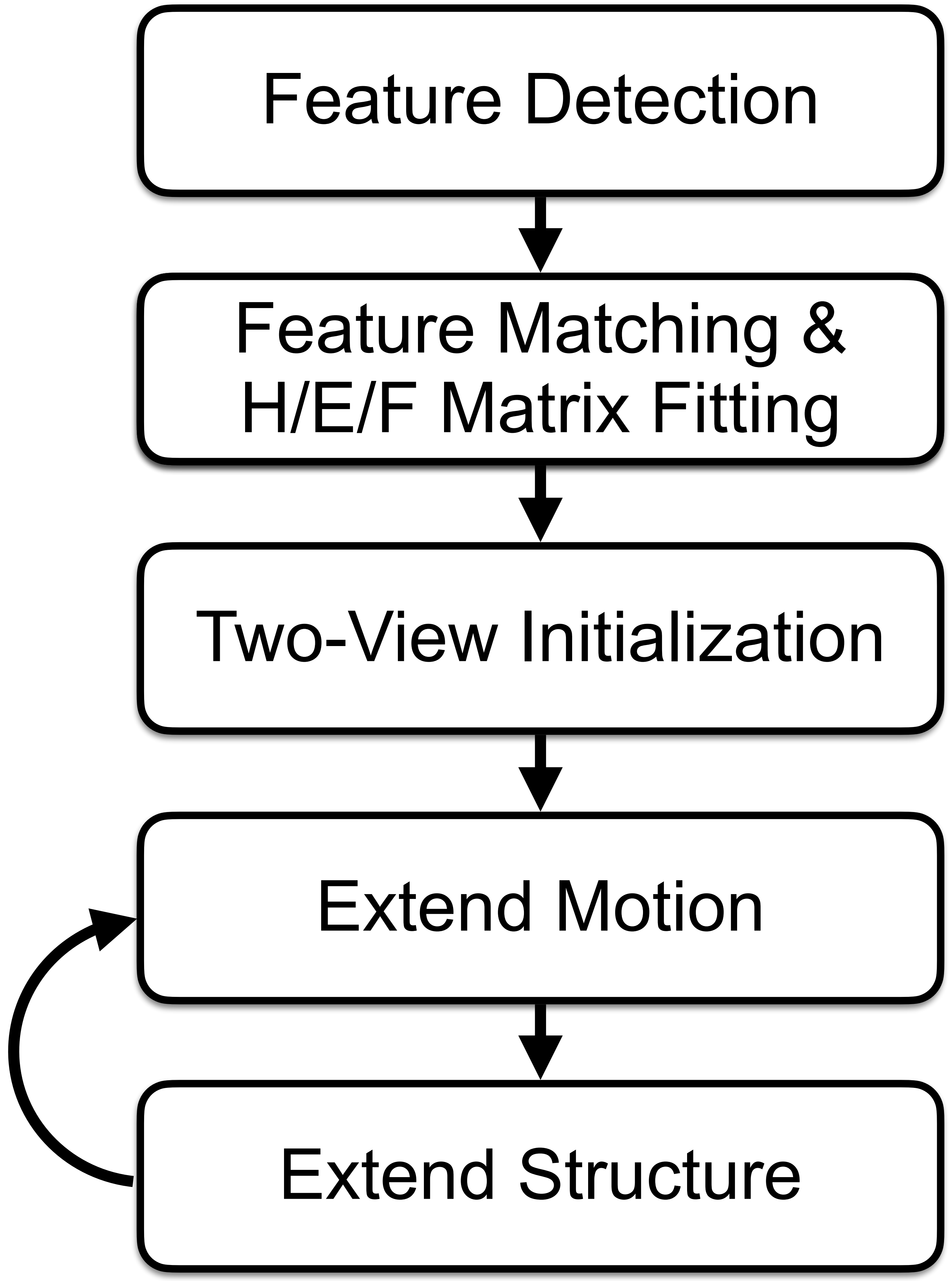


# Sequential / Incremental SfM

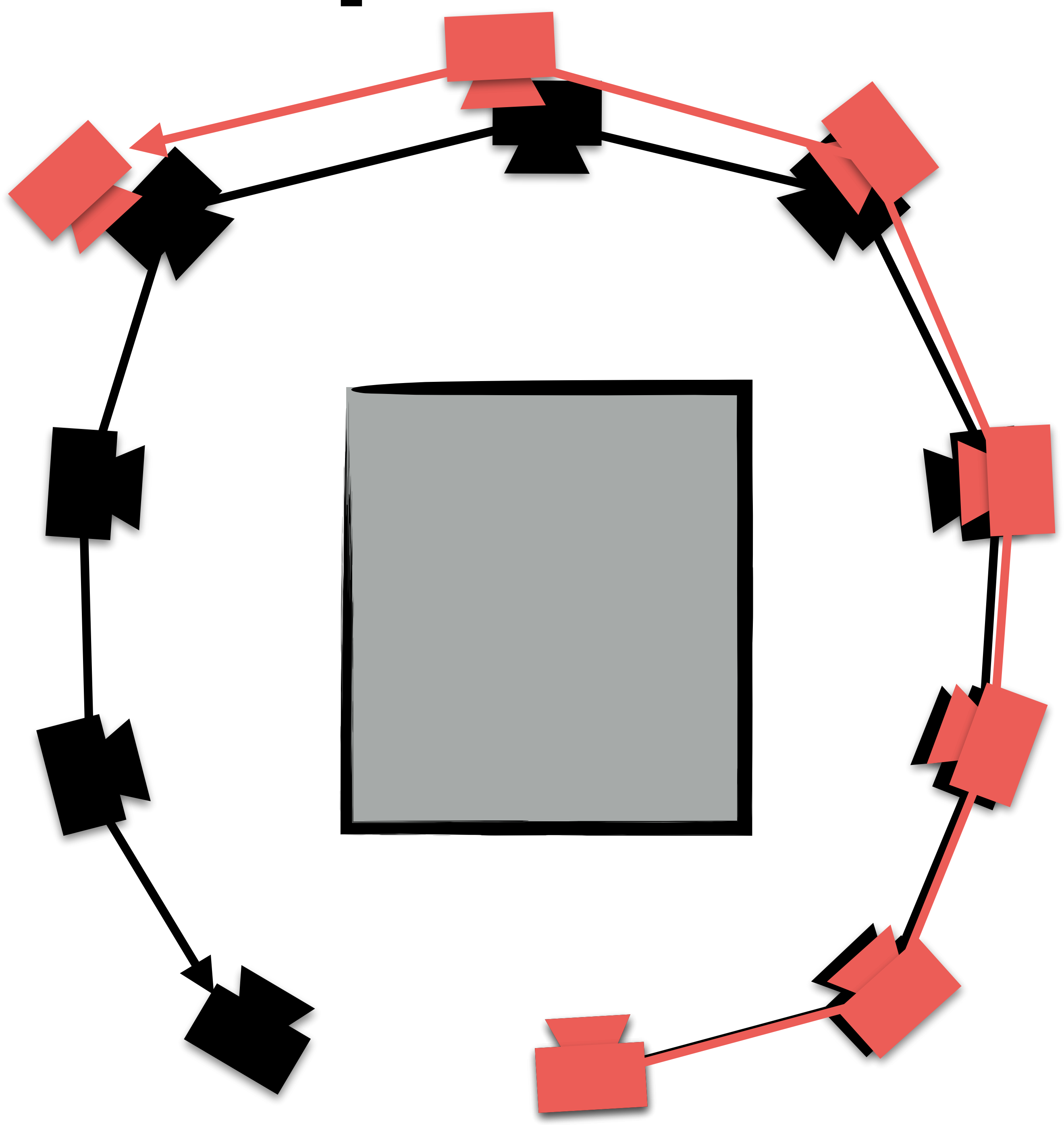


True trajectory

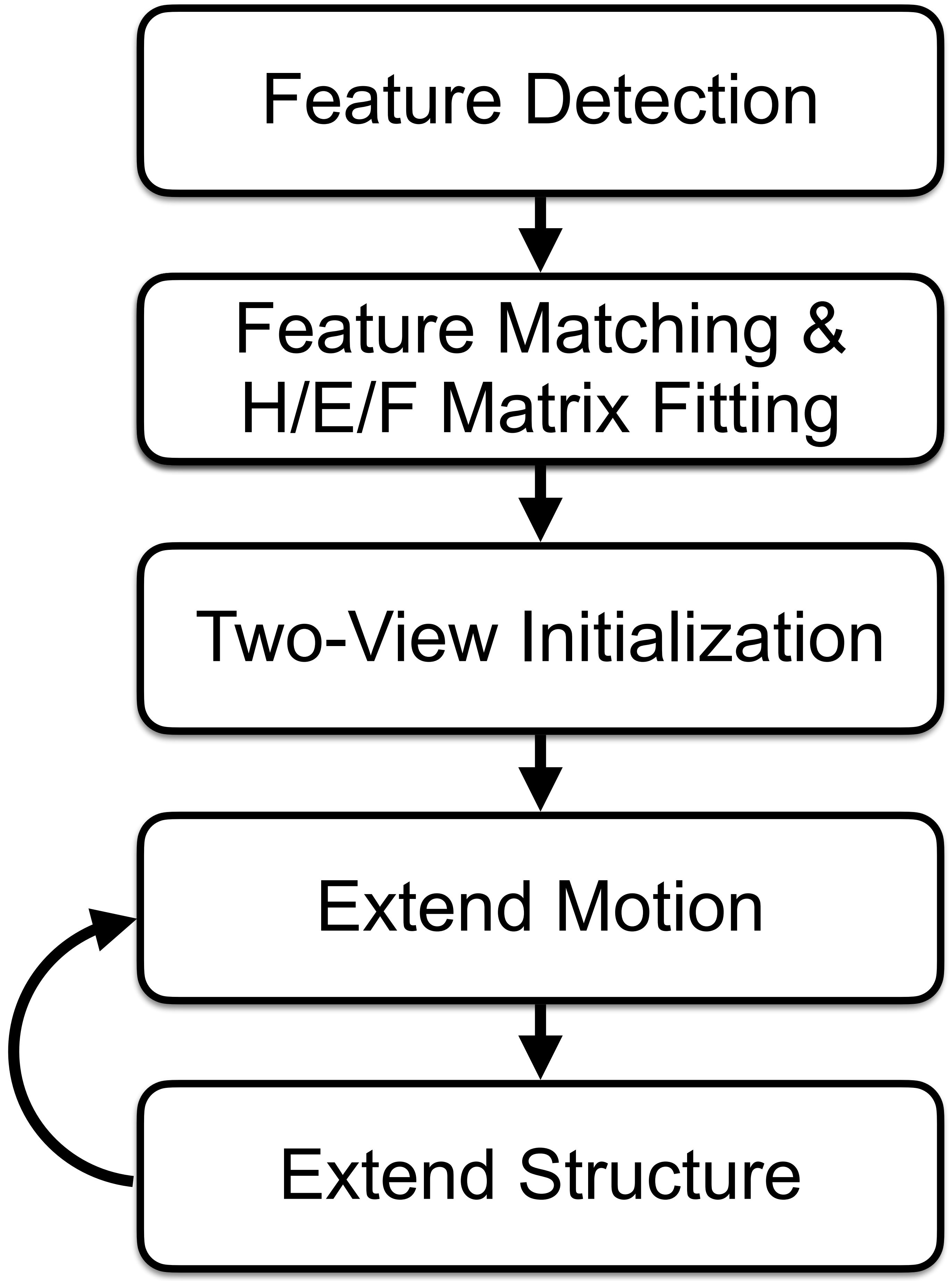
Estimated trajectory



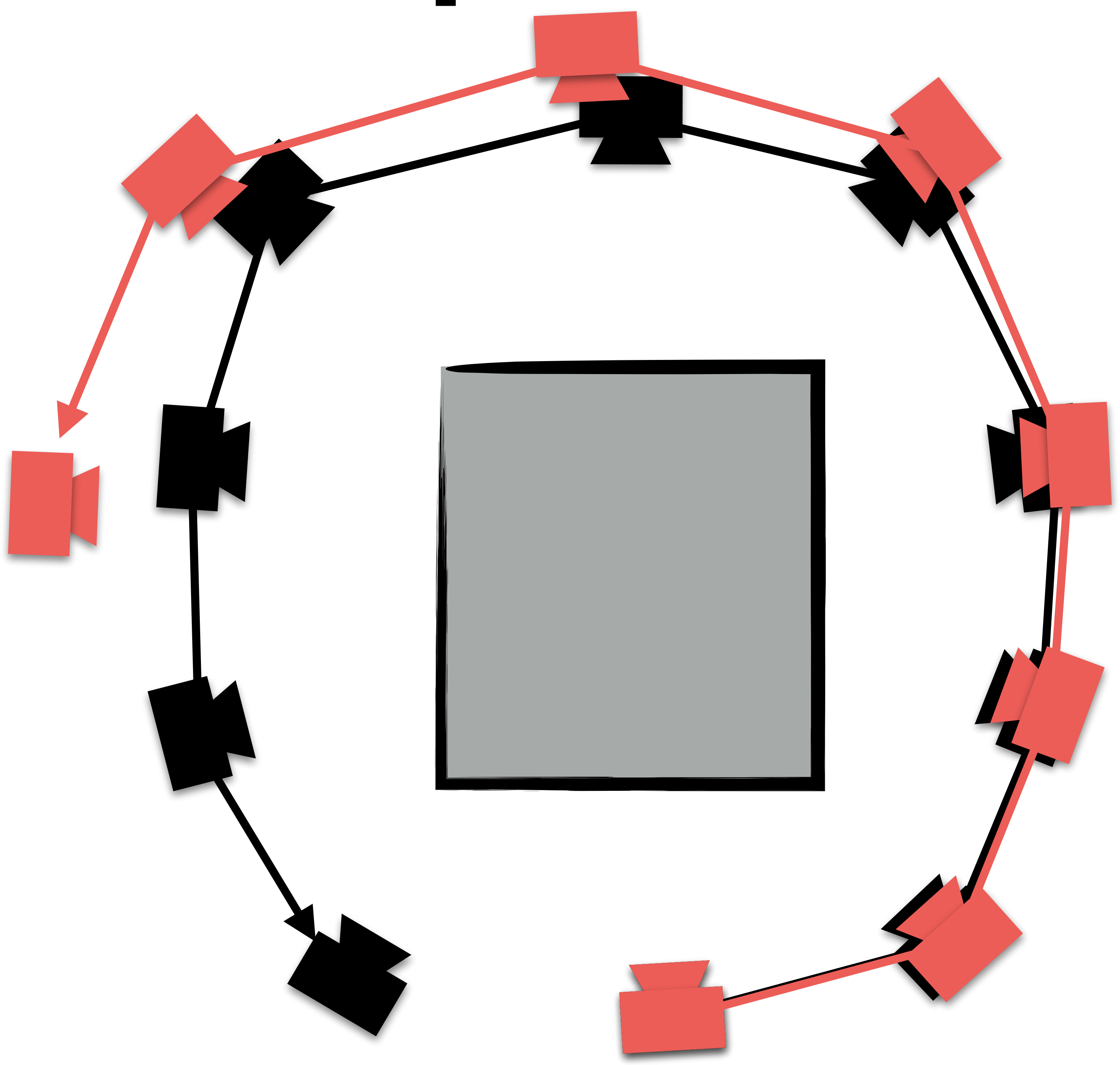
# Sequential / Incremental SfM



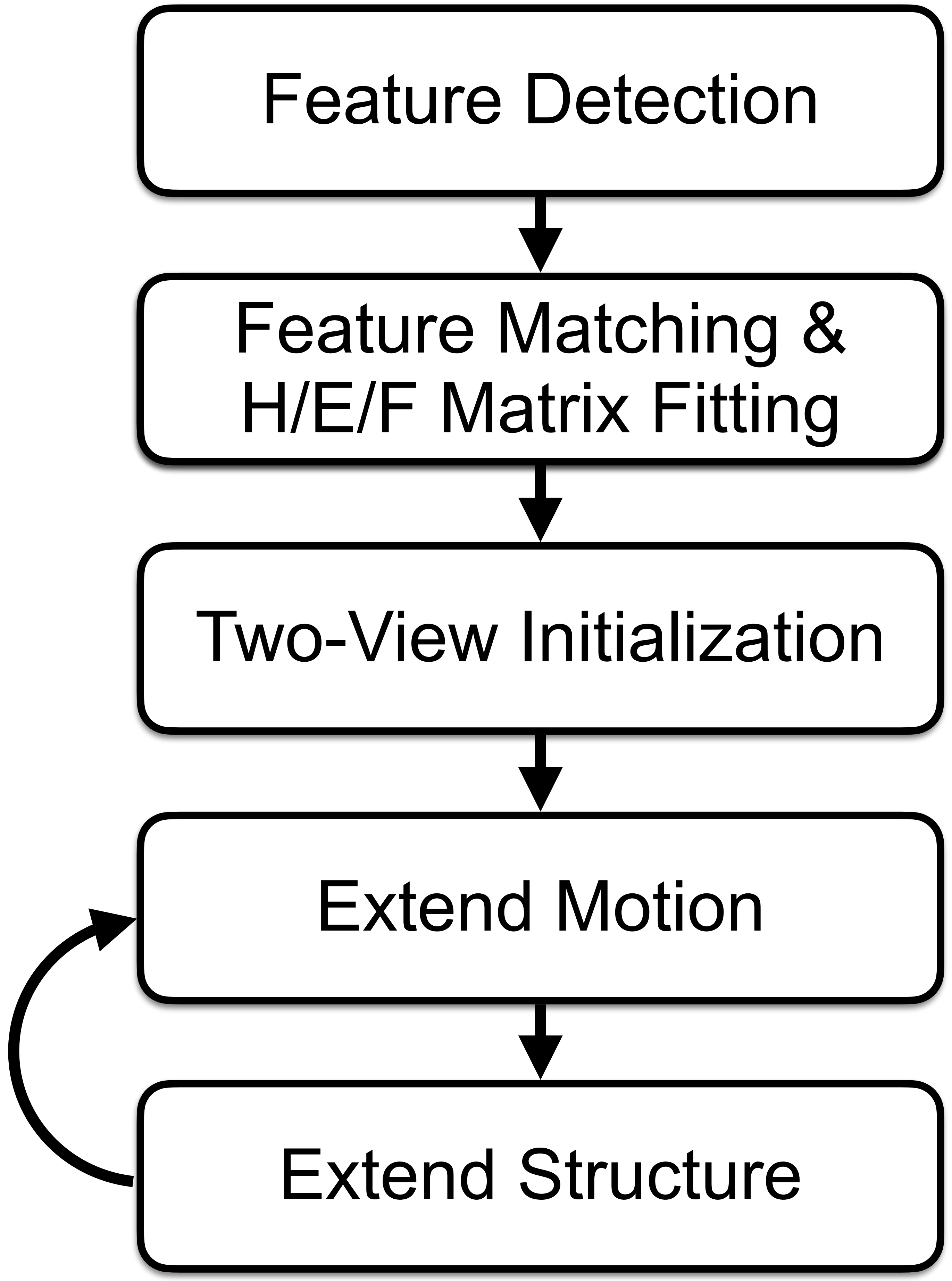
True trajectory
Estimated trajectory



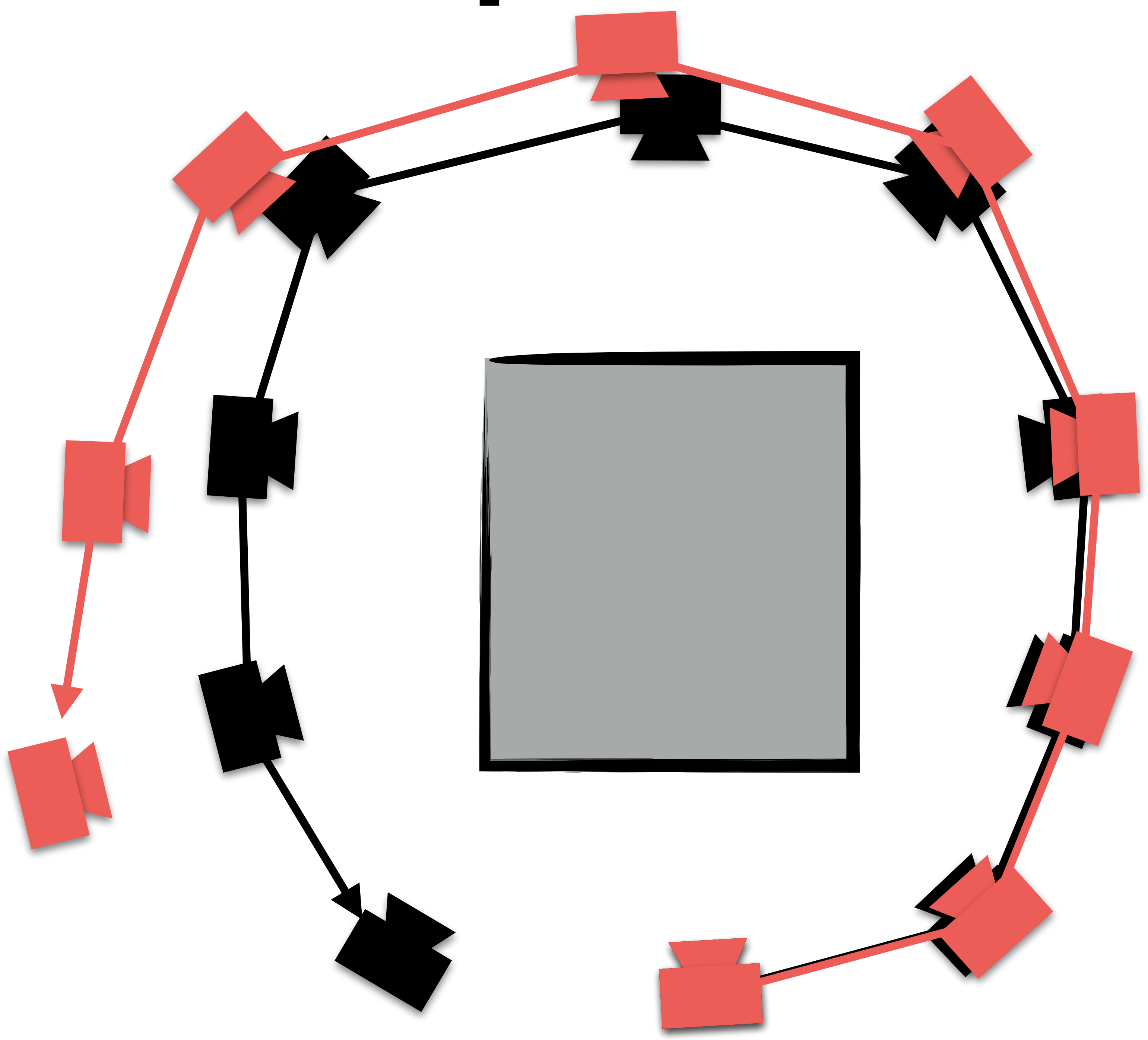
# Sequential / Incremental SfM



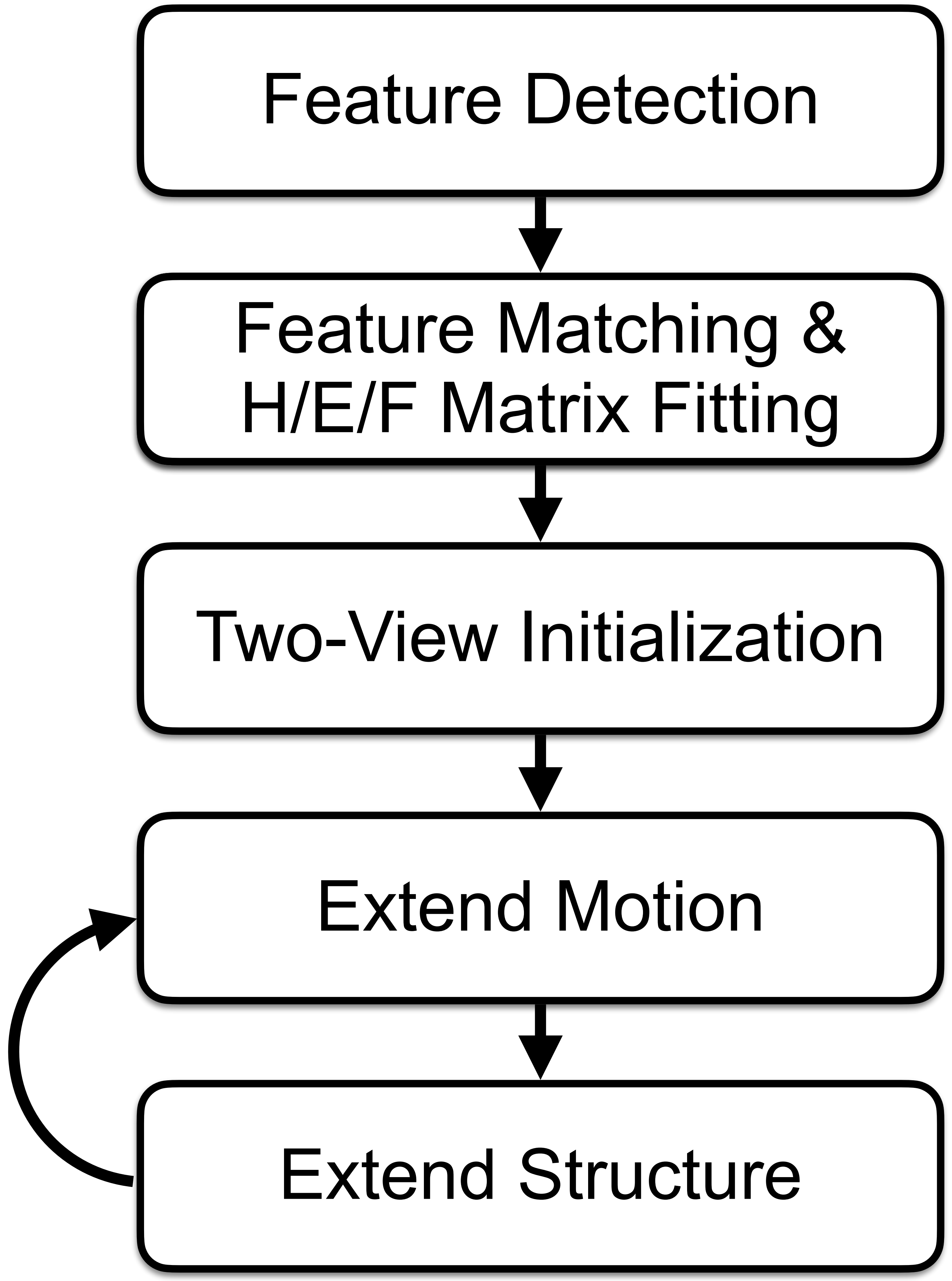
True trajectory
Estimated trajectory



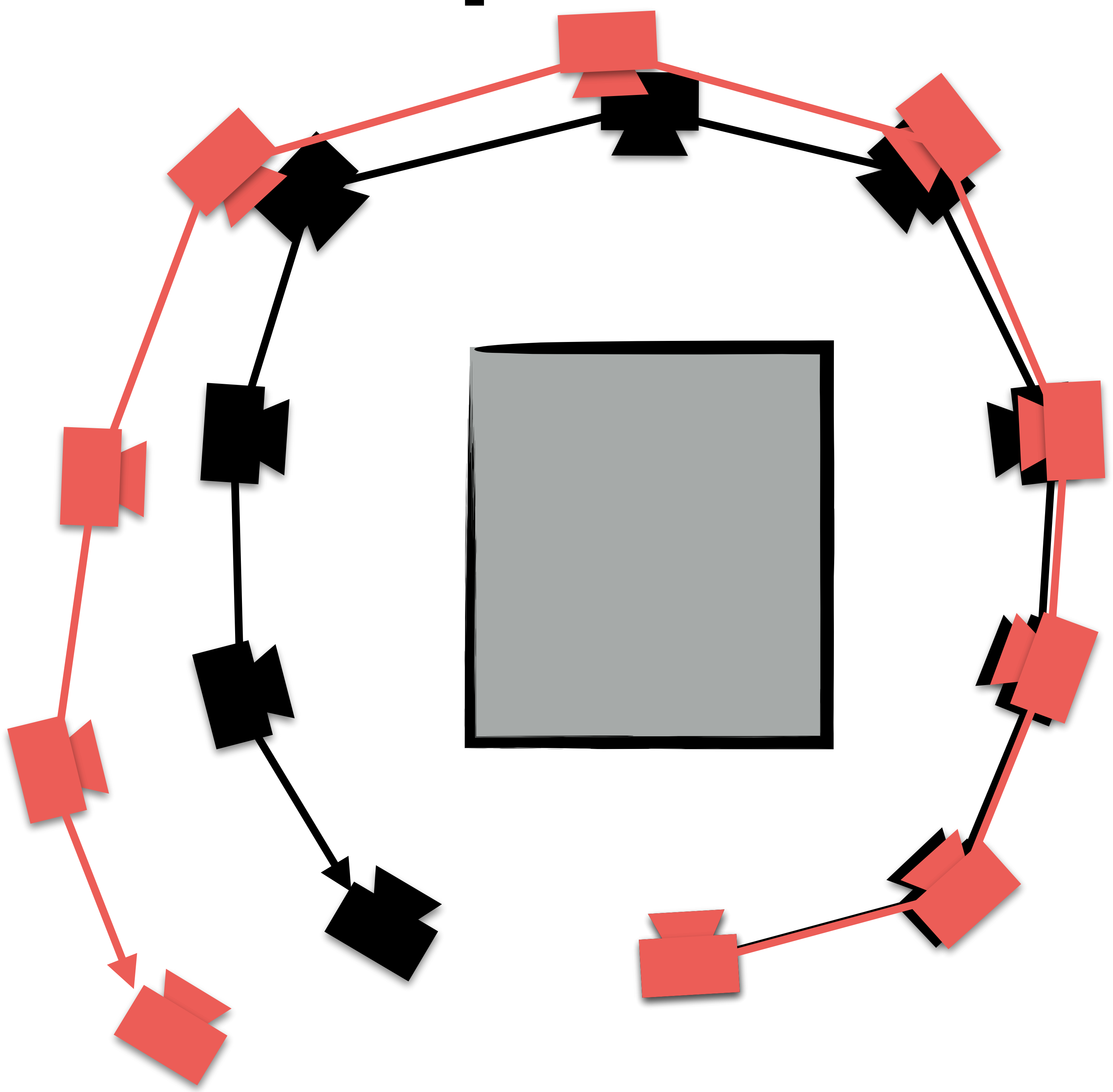
# Sequential / Incremental SfM



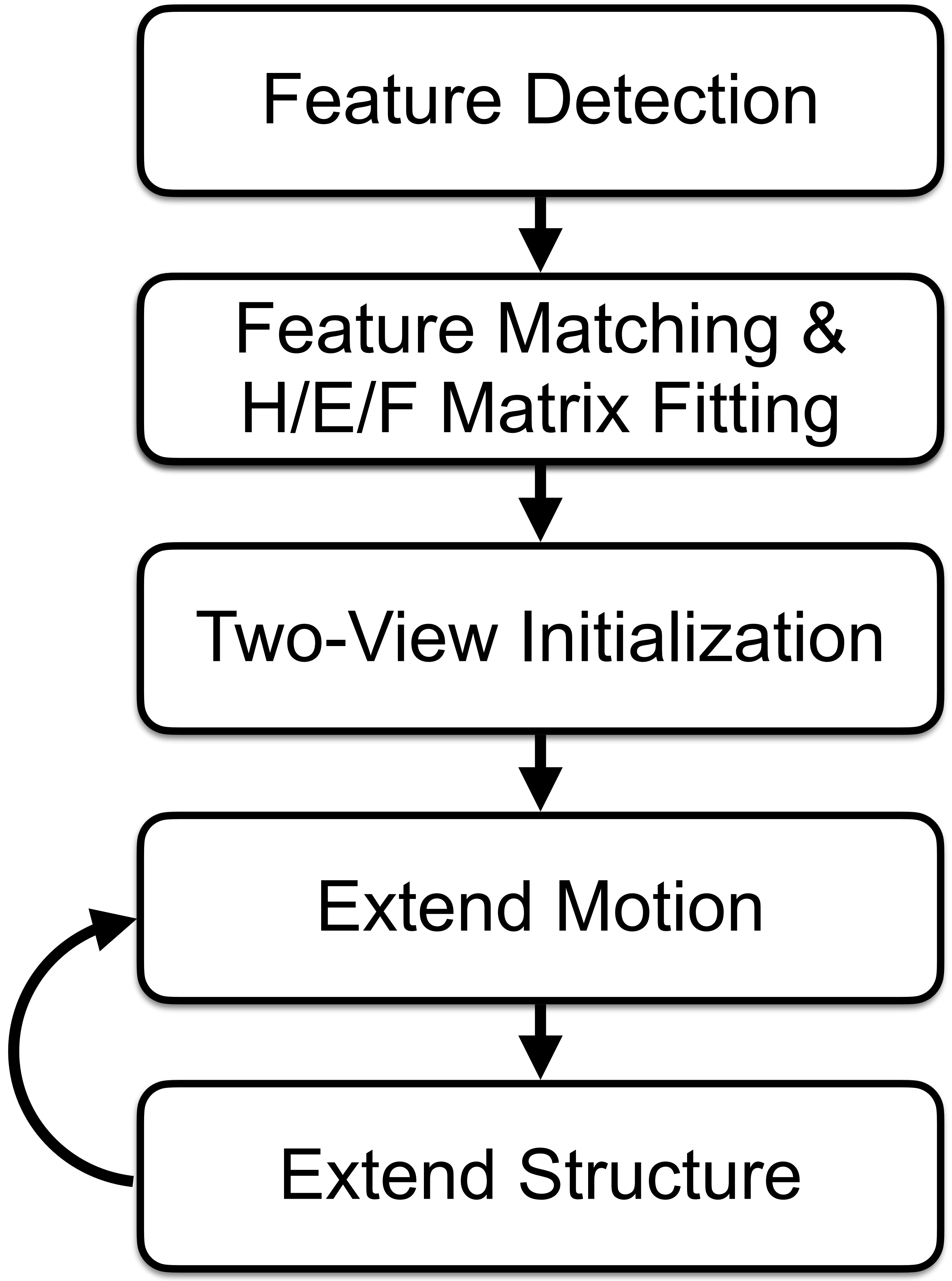
True trajectory
Estimated trajectory



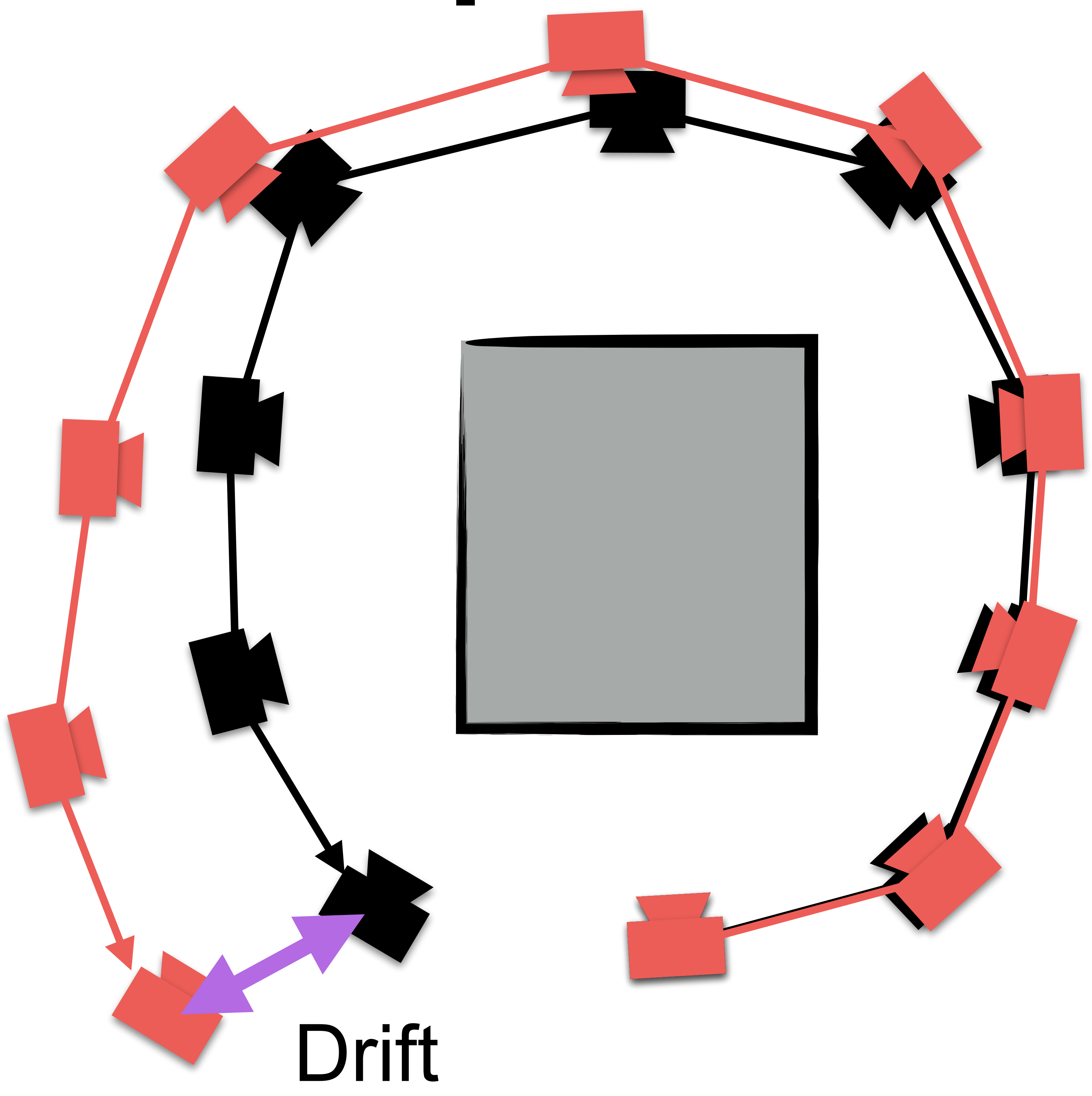
# Sequential / Incremental SfM



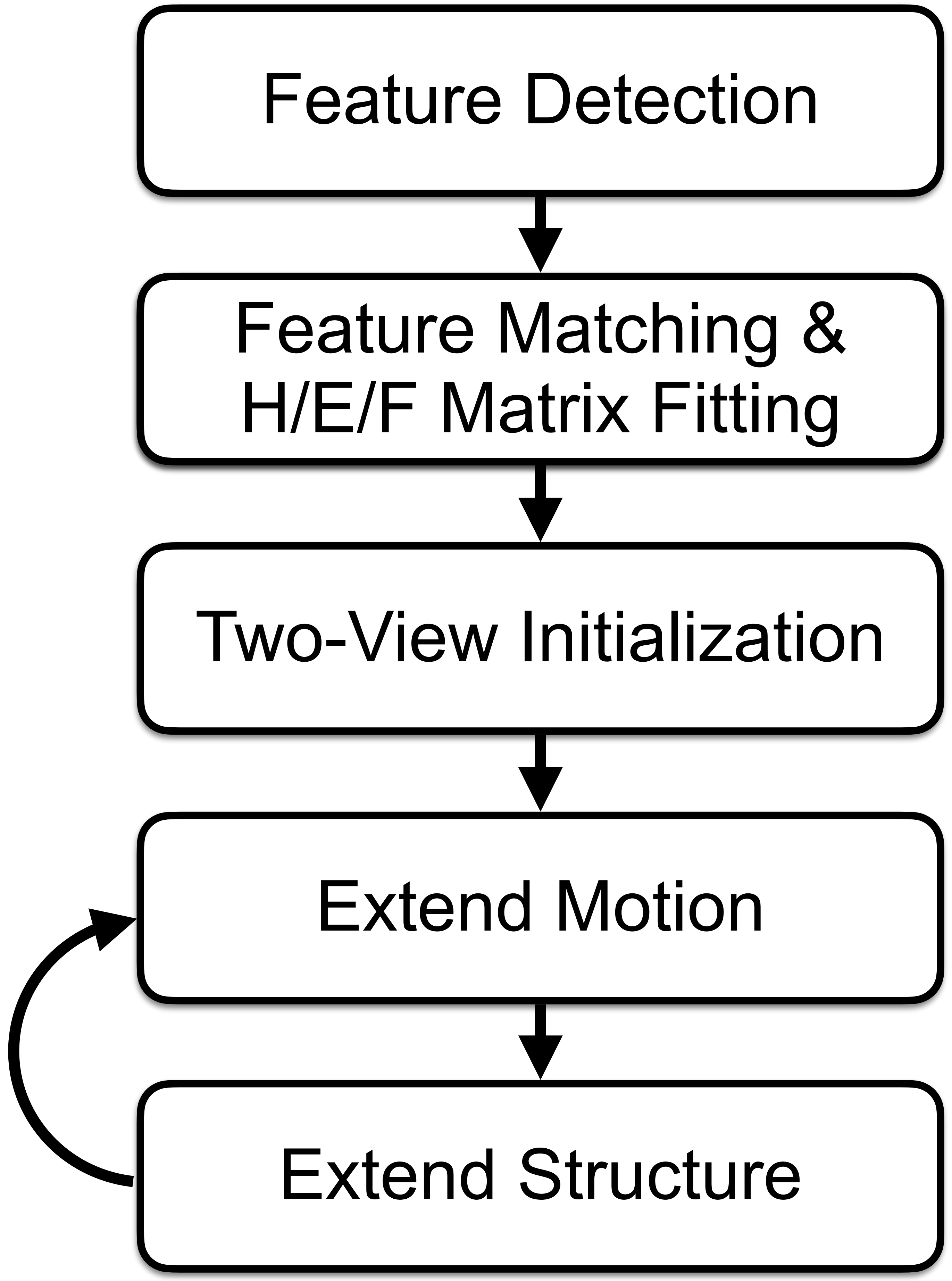
True trajectory
Estimated trajectory



# Sequential / Incremental SfM

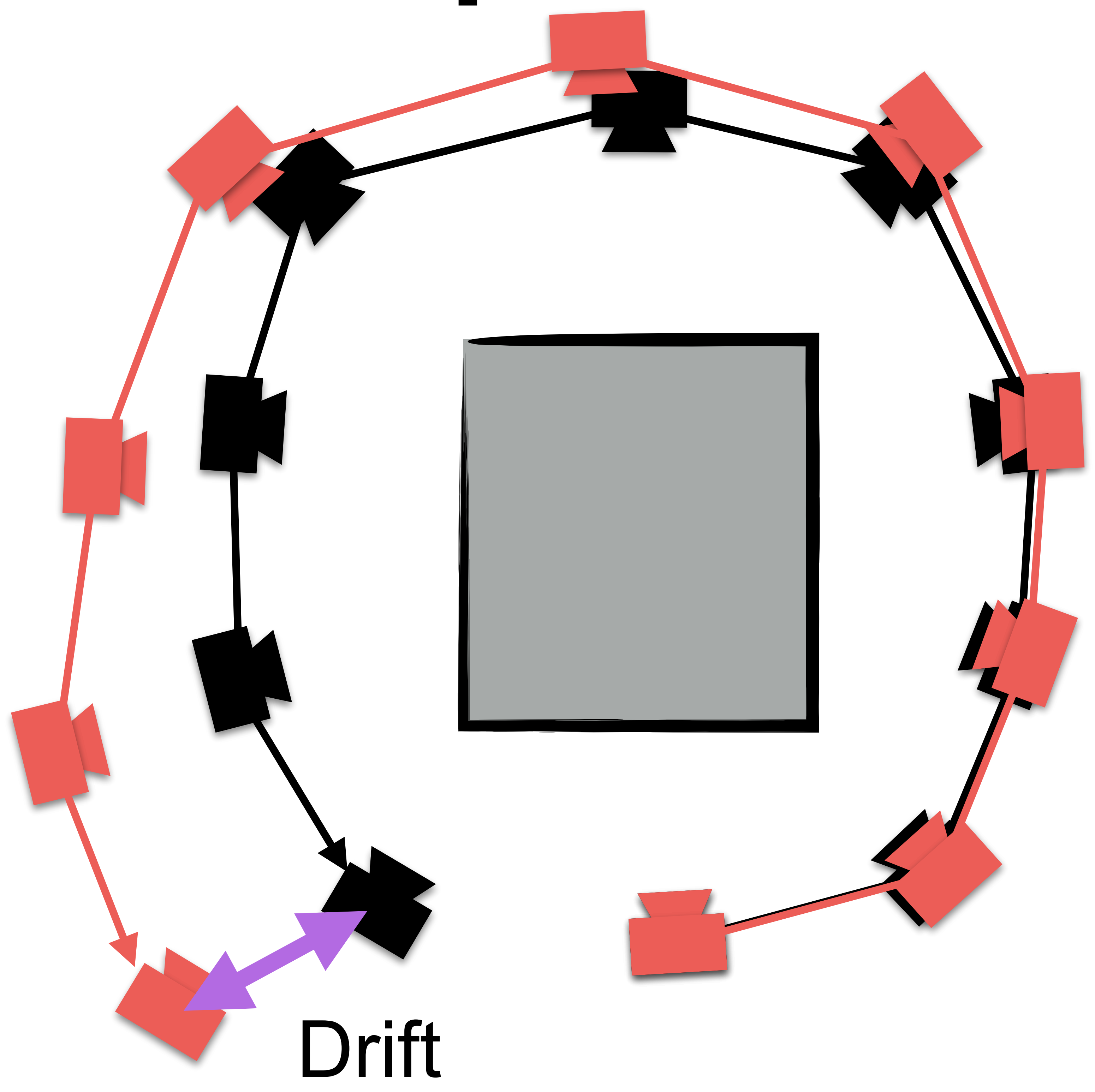



True trajectory
Estimated trajectory

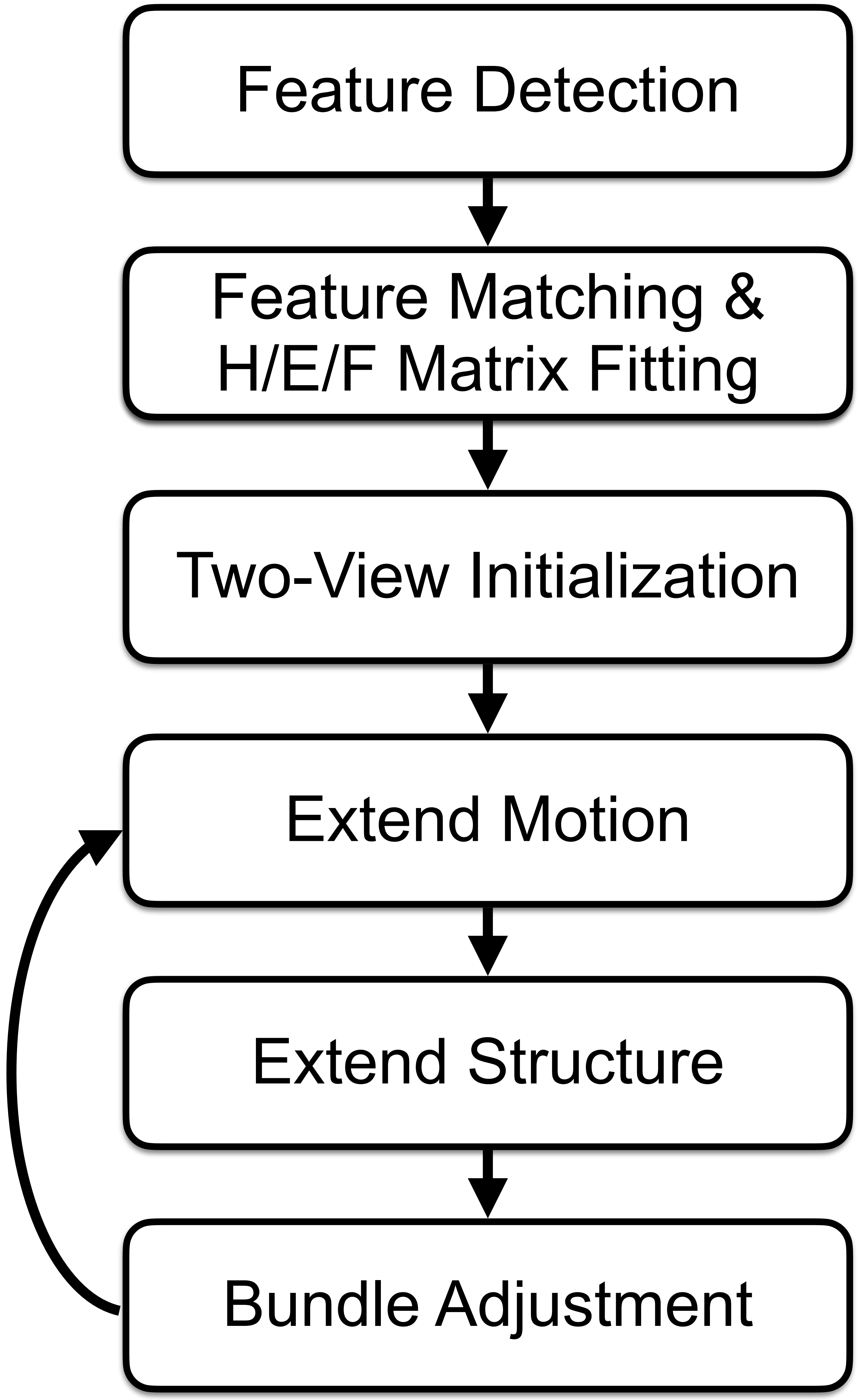



- Errors accumulate, leading to drift over time
- Adjust motion and structure frequently

# Sequential / Incremental SfM

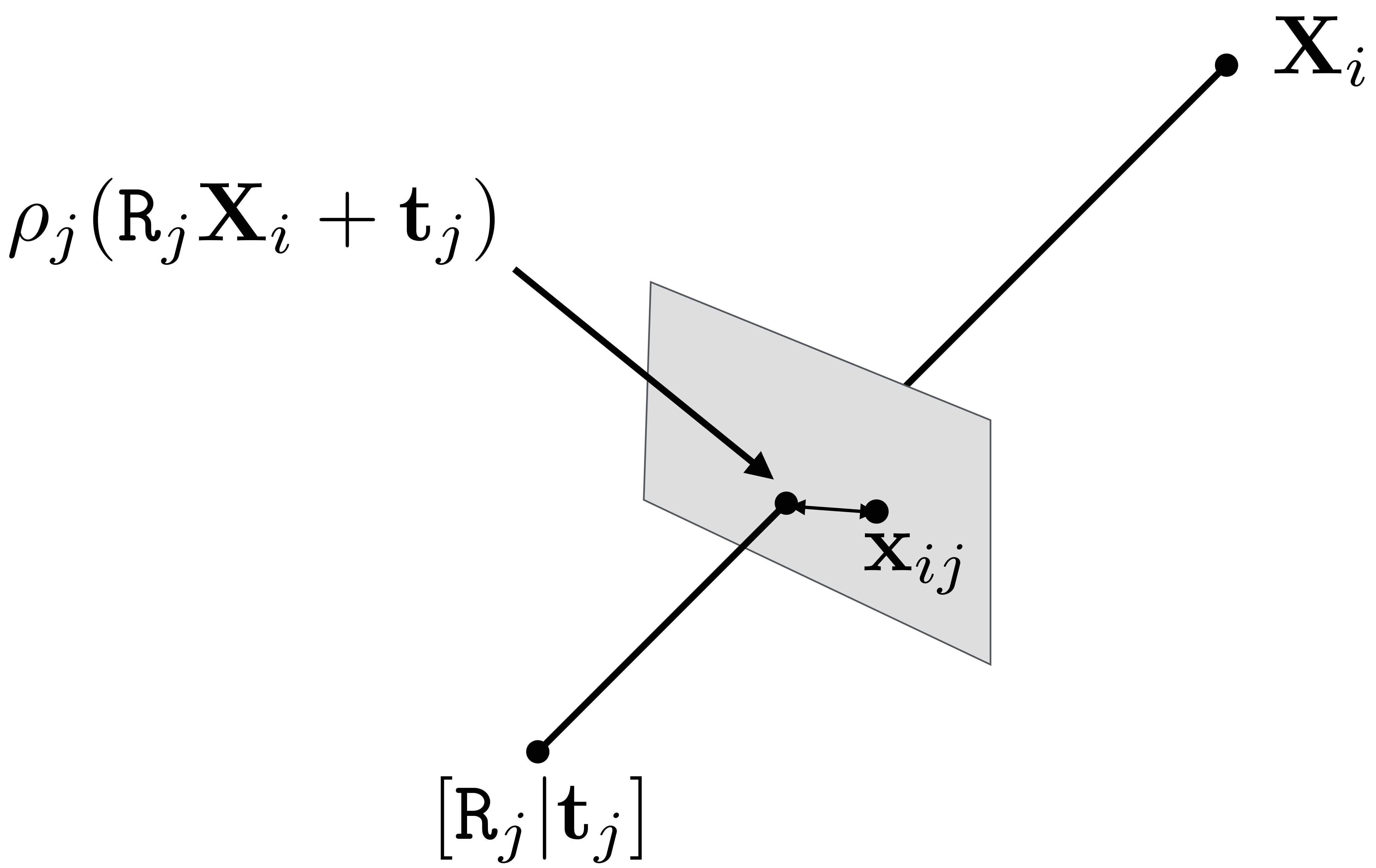


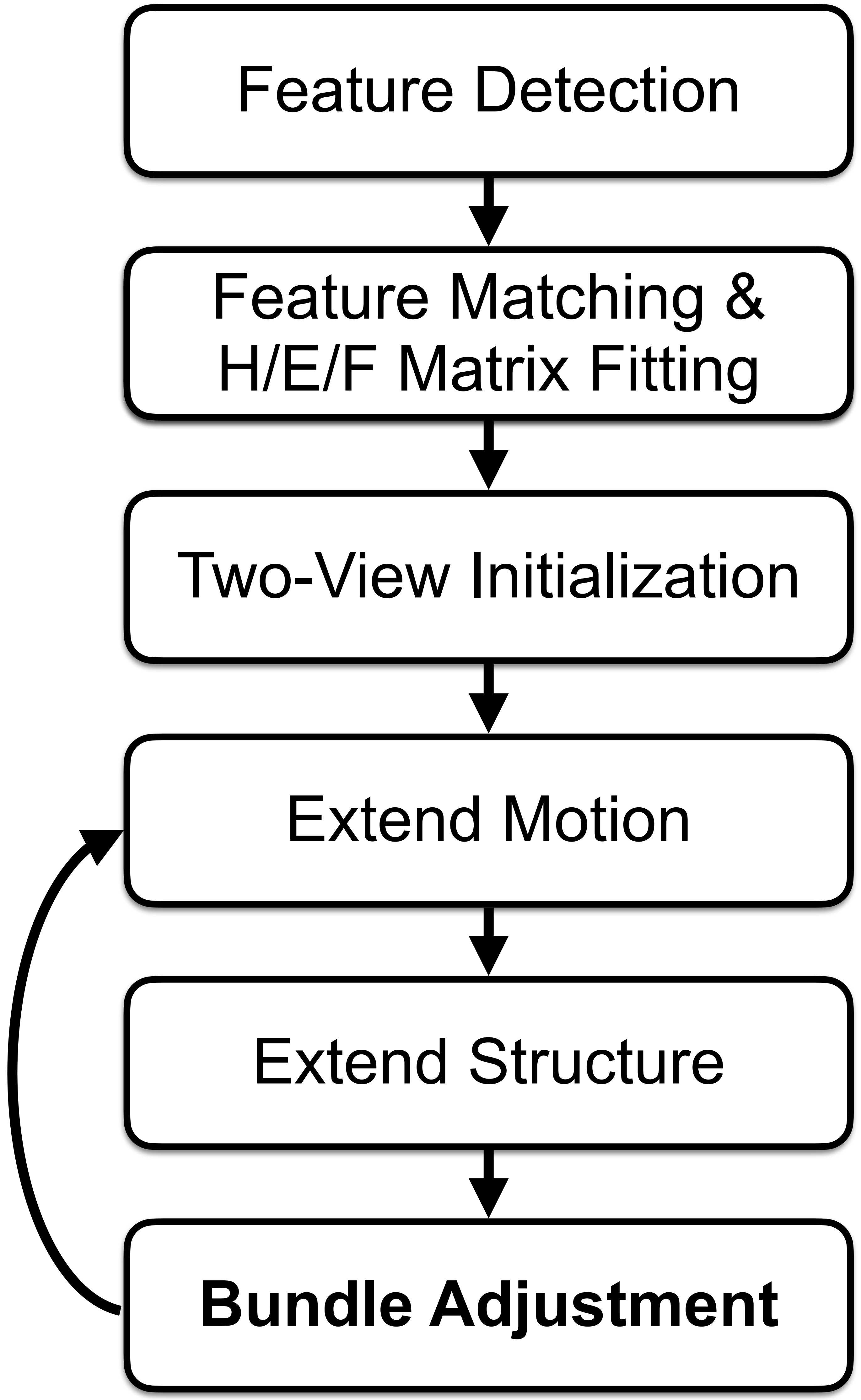

True trajectory
Estimated trajectory



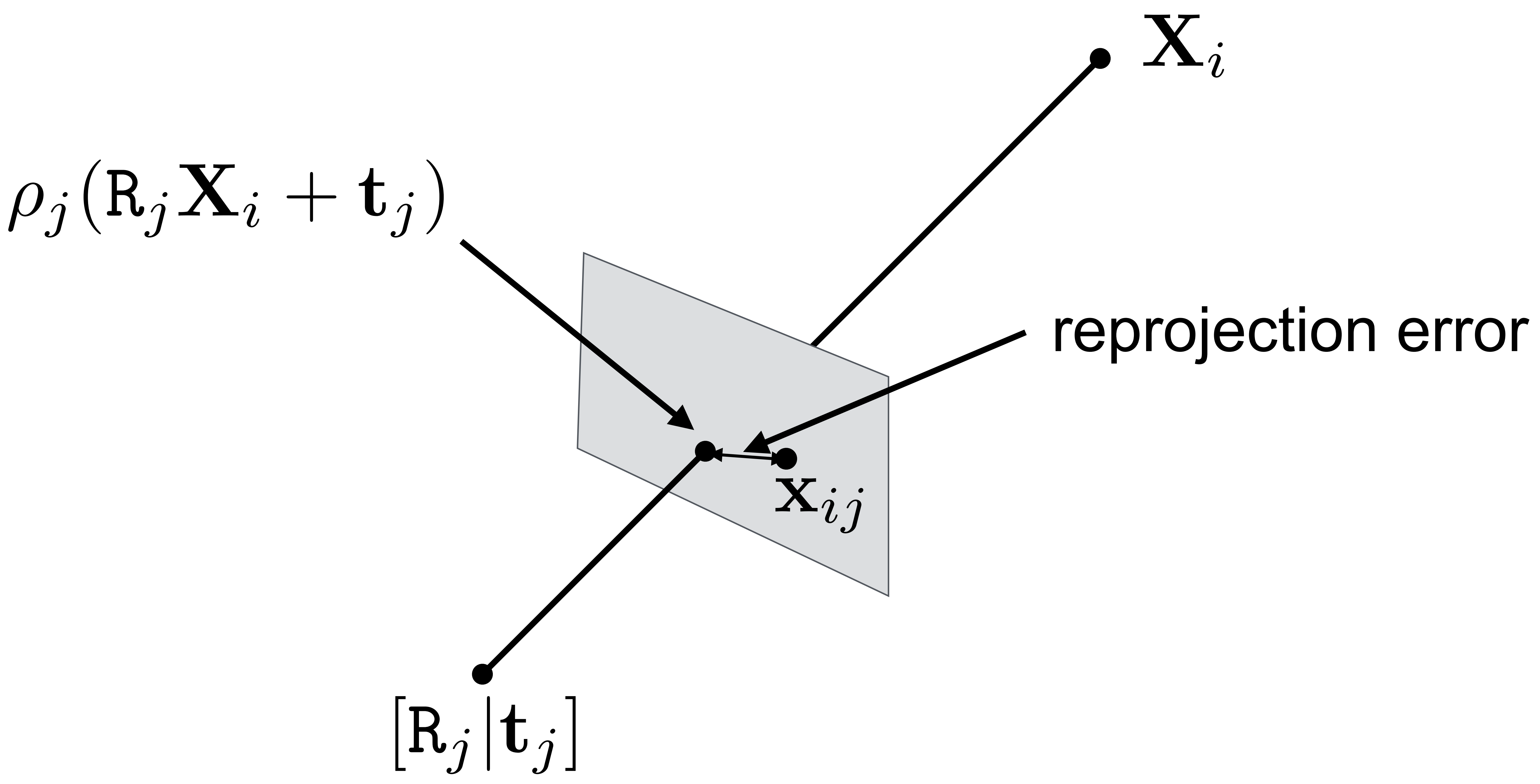

- Errors accumulate, leading to drift over time
- Adjust motion and structure frequently

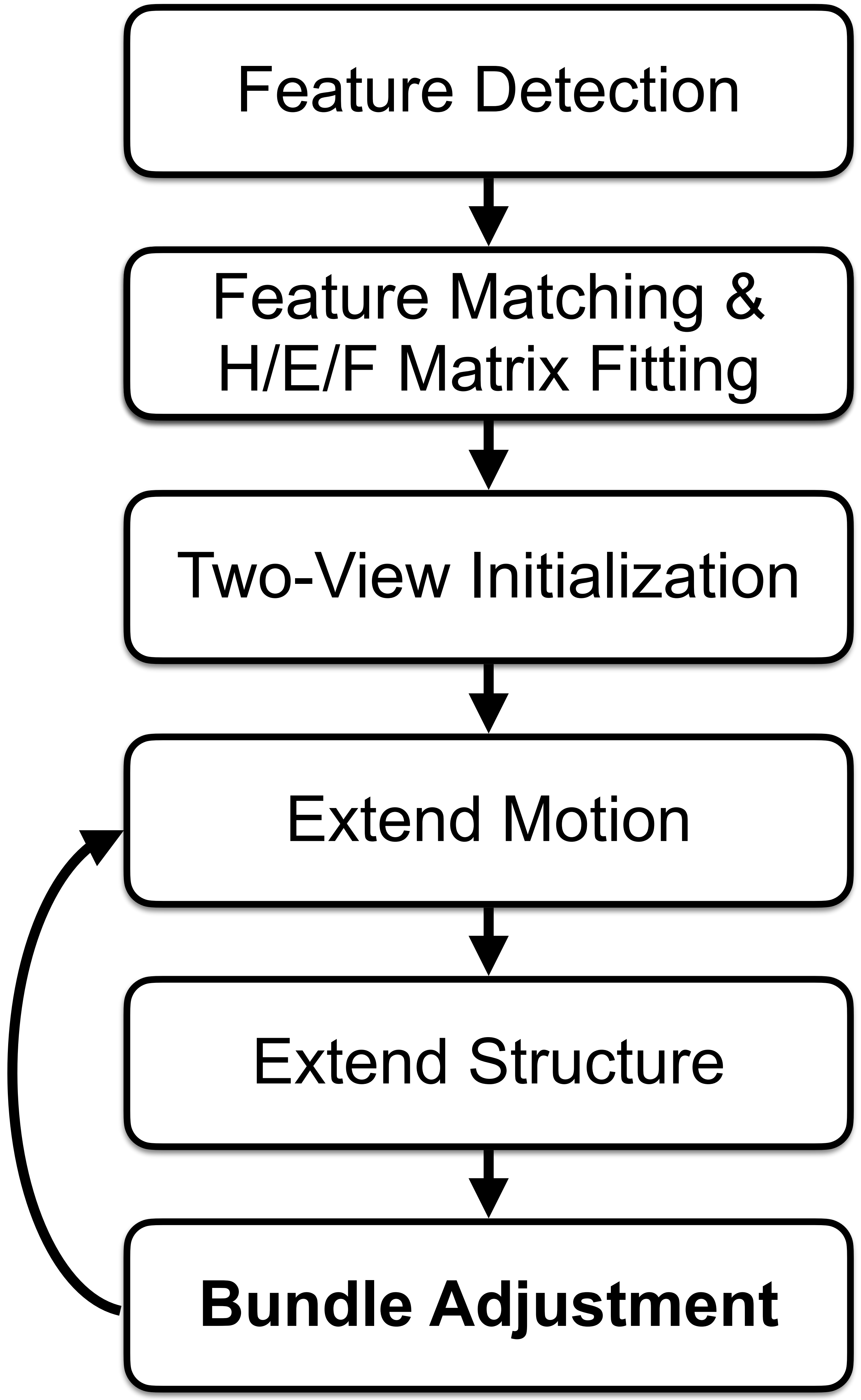
# Bundle Adjustment



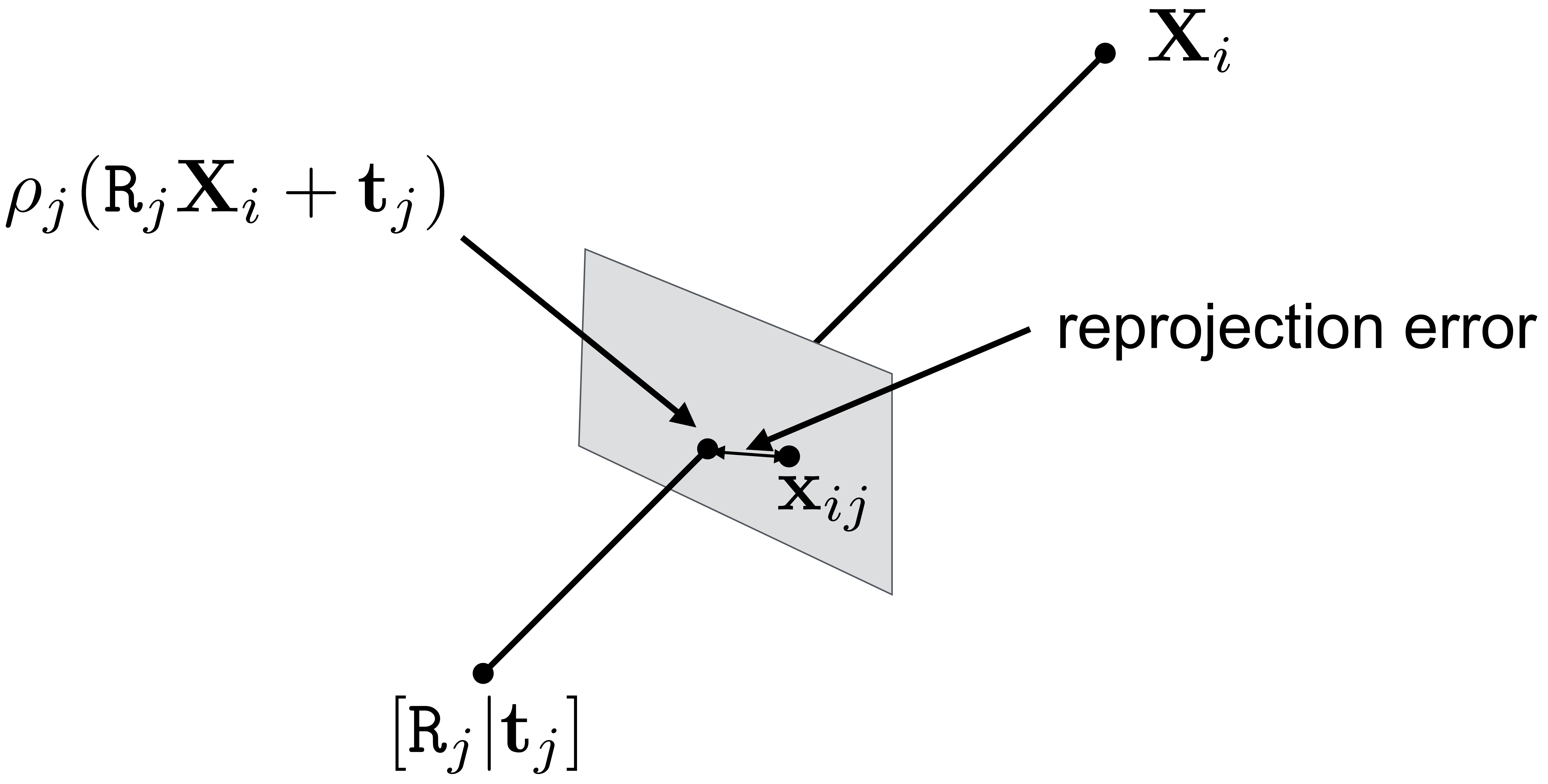



# Bundle Adjustment

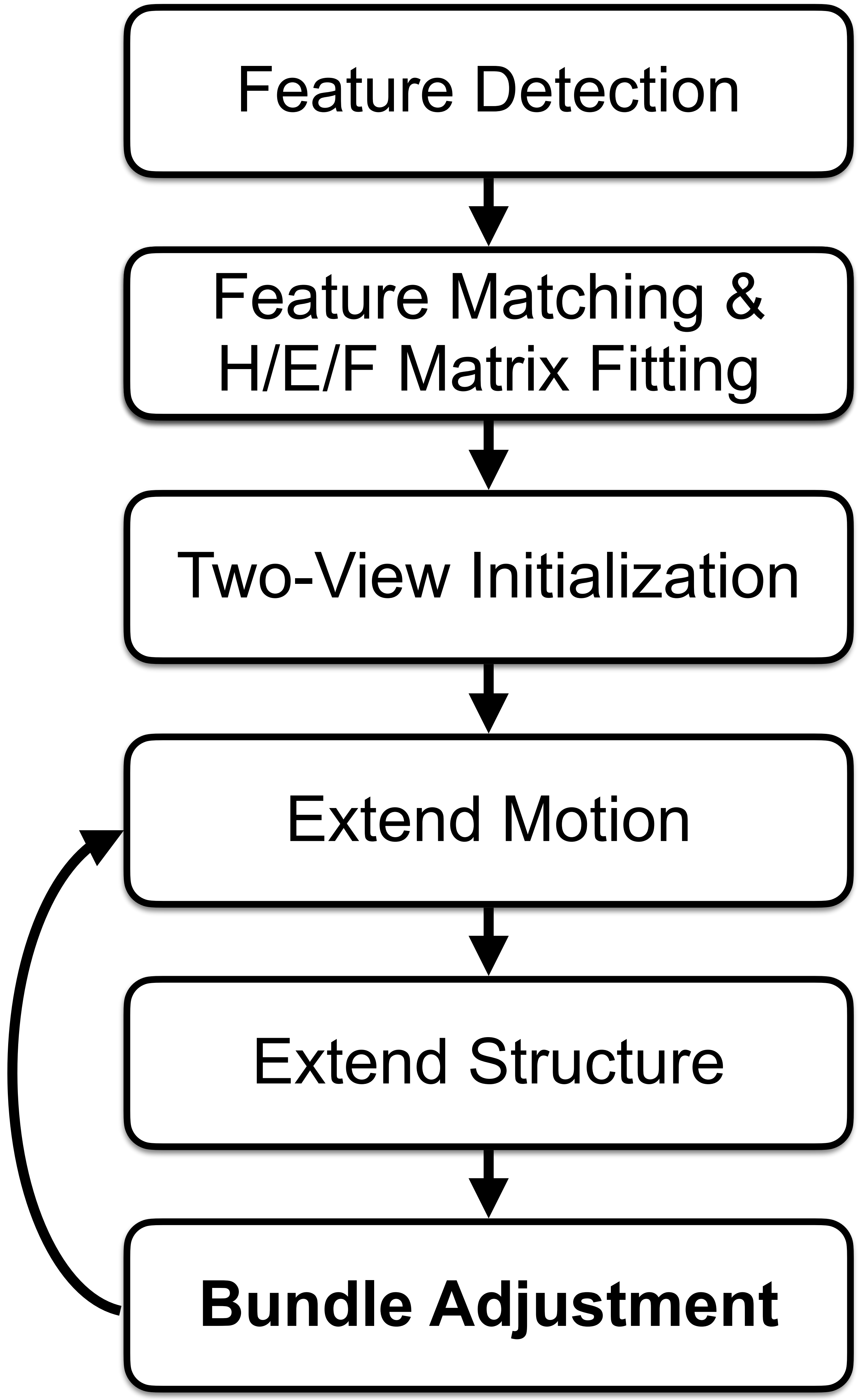


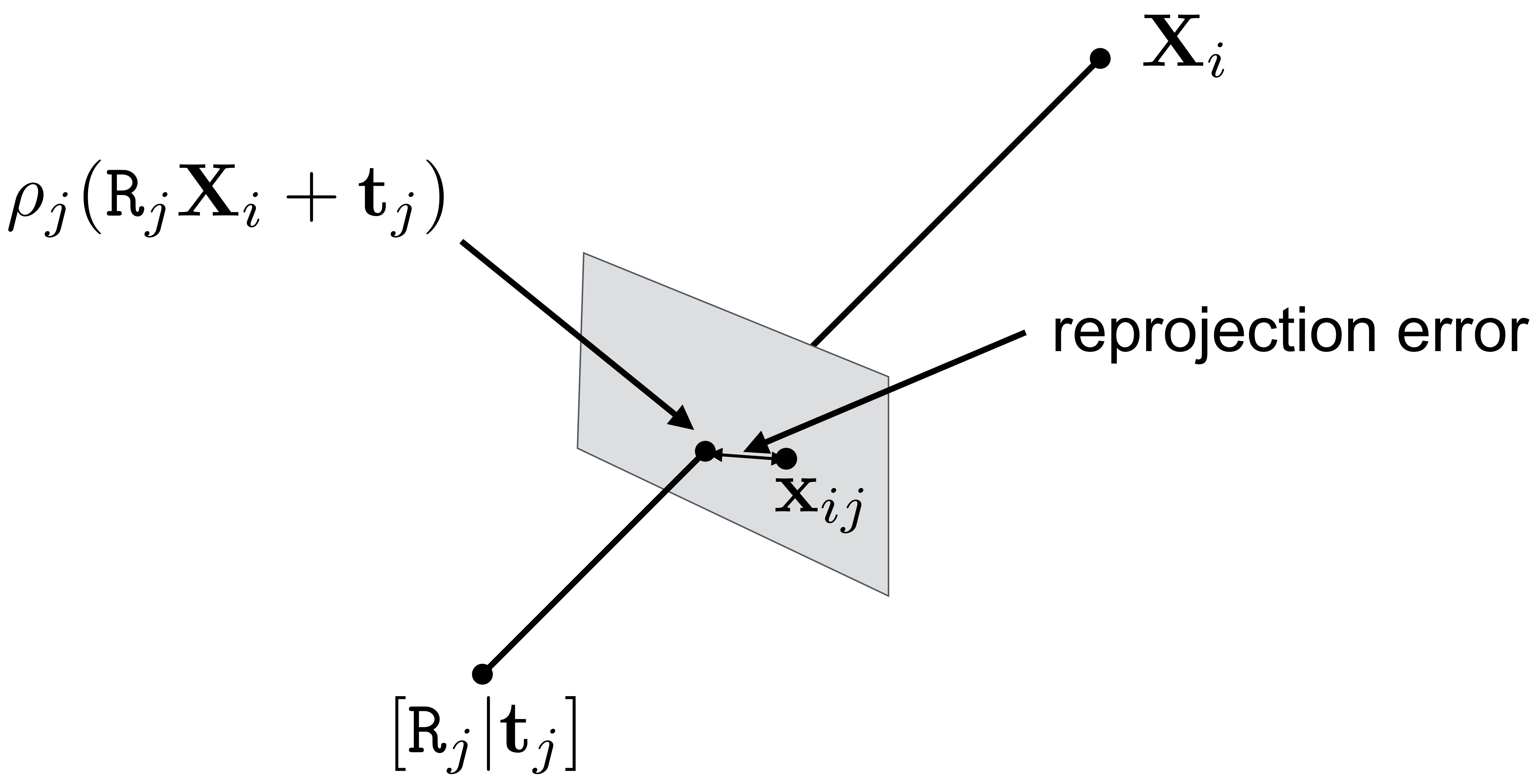
# Bundle Adjustment




$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j \mathbf{X}_i + \mathbf{t}_j)||^2$$

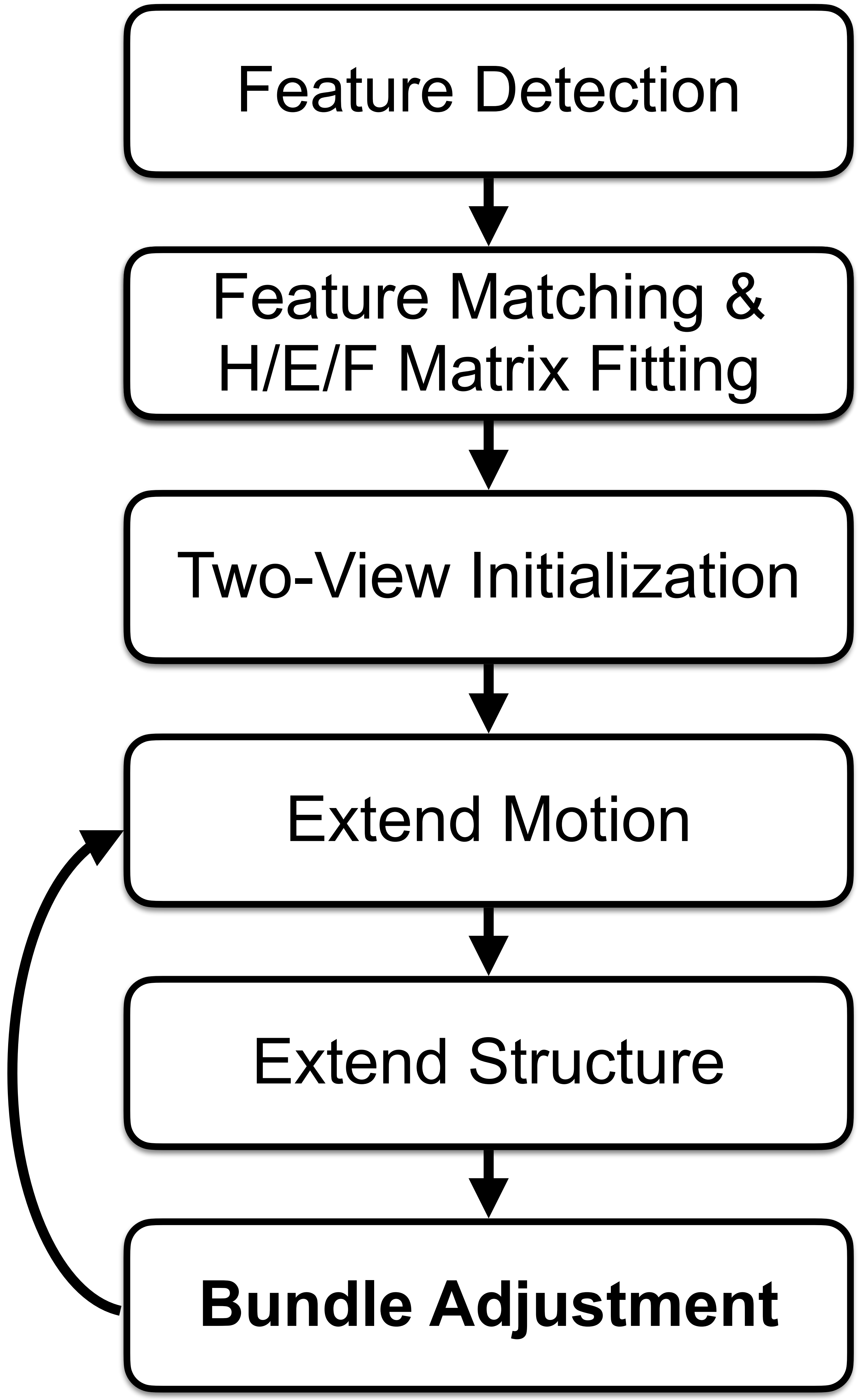


# Bundle Adjustment

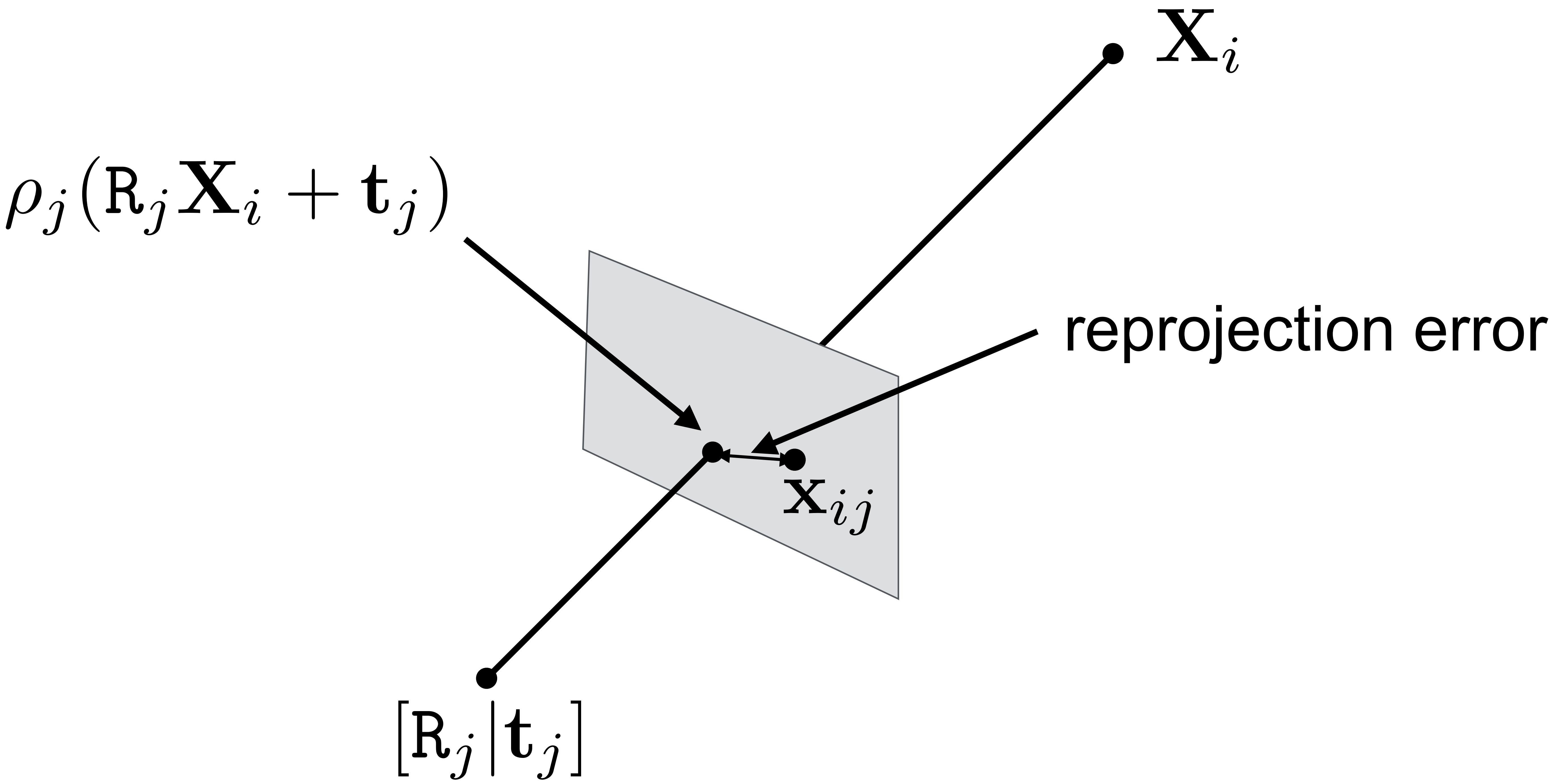



$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j \mathbf{X}_i + \mathbf{t}_j)||^2$$

point i visible in image j?



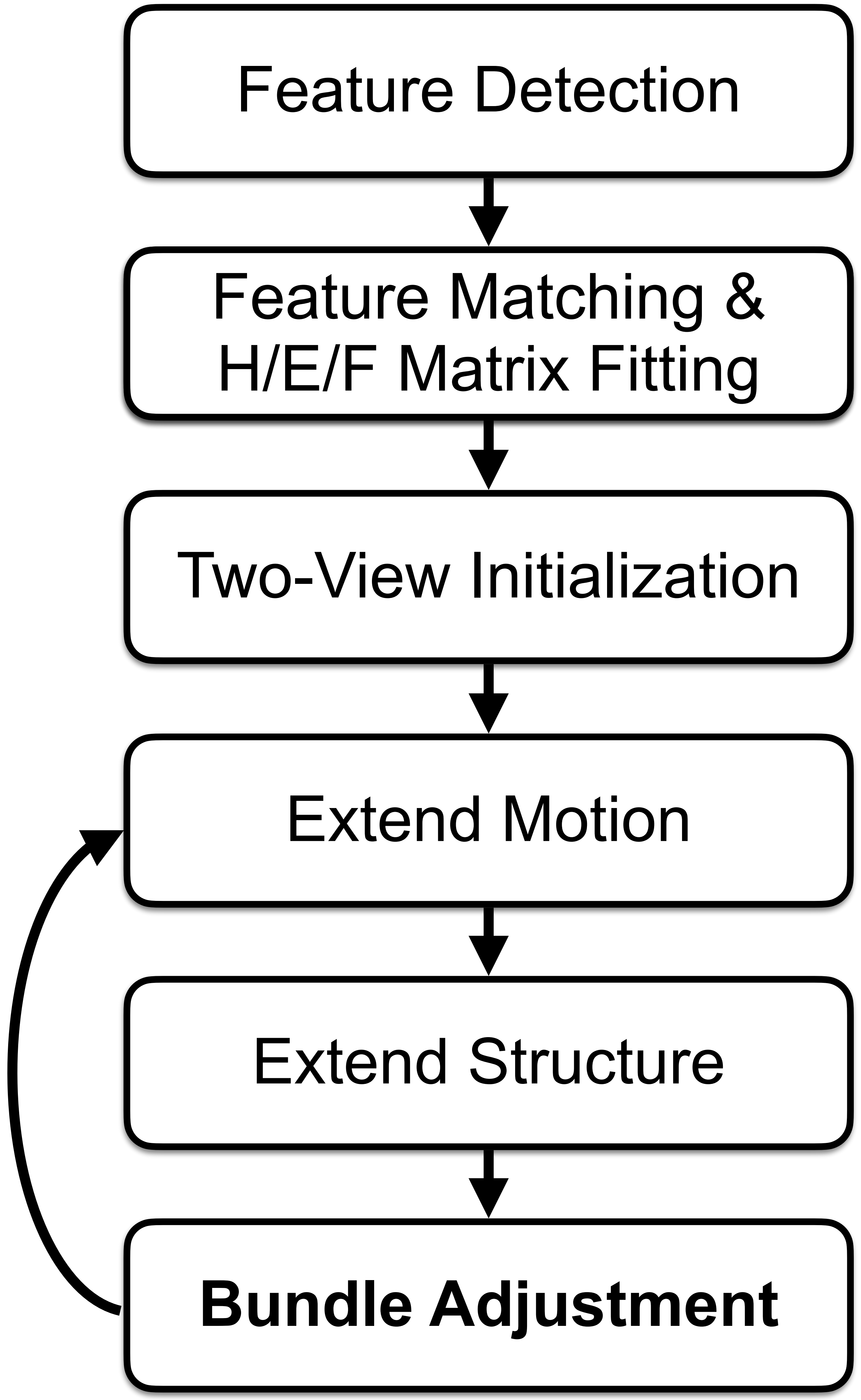
# Bundle Adjustment



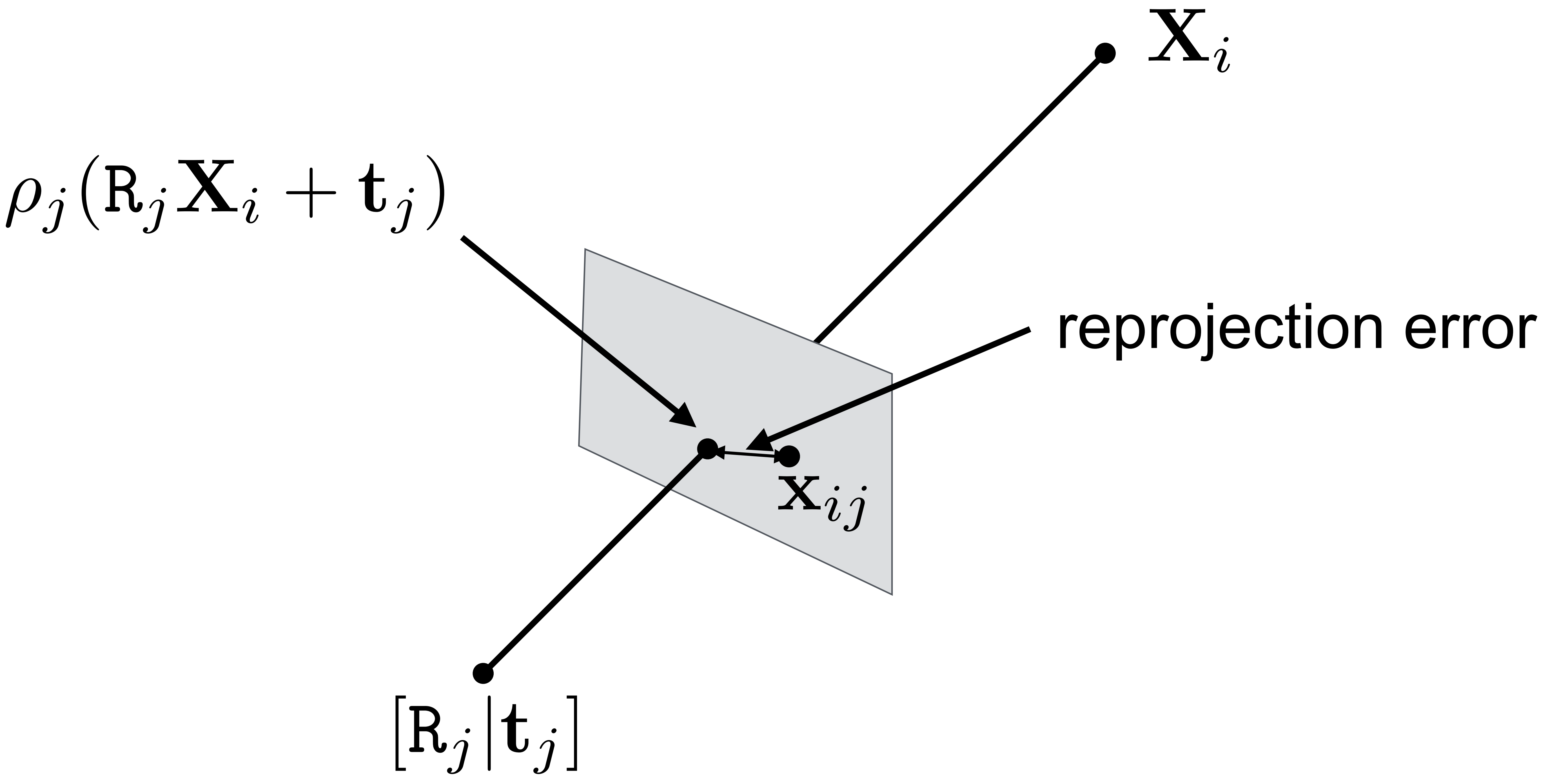

$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j \mathbf{X}_i + \mathbf{t}_j)||^2$$

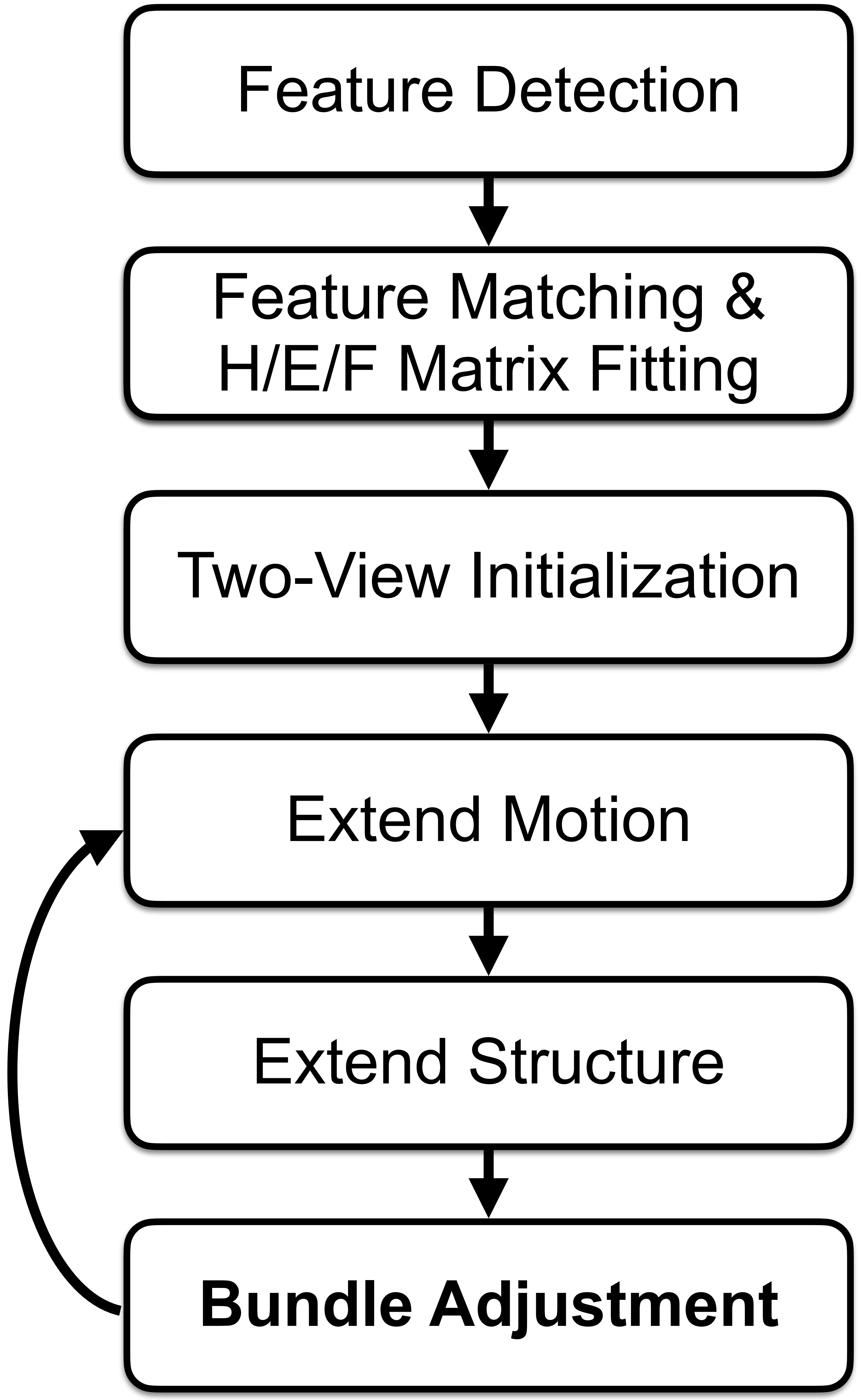
Can also refine camera intrinsics

point i visible in image j?



# Bundle Adjustment




$$\underset{\text{camera poses, 3D points}}{\operatorname{argmin}} \sum_i \sum_j \Delta_{ij} ||\mathbf{x}_{ij} - \rho_j(\mathtt{R}_j\mathbf{X}_i + \mathbf{t}_j)||^2$$

Can also refine camera intrinsics (camera poses, 3D points)

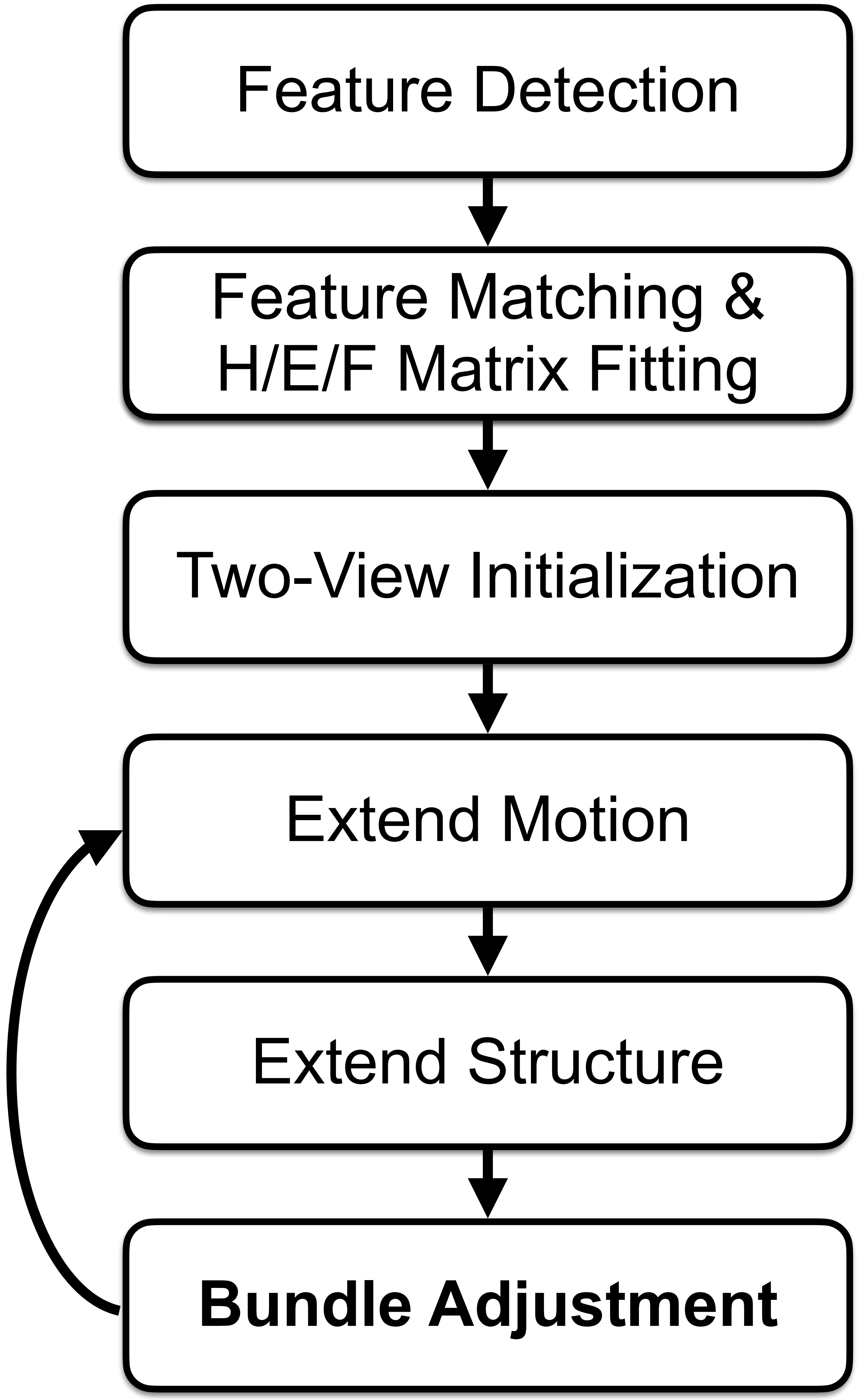
point i visible in image j? ($\Delta_{ij}$)

Cost function is highly non-linear → refine from initialization

# Bundle Adjustment

## Gradient descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ,$$

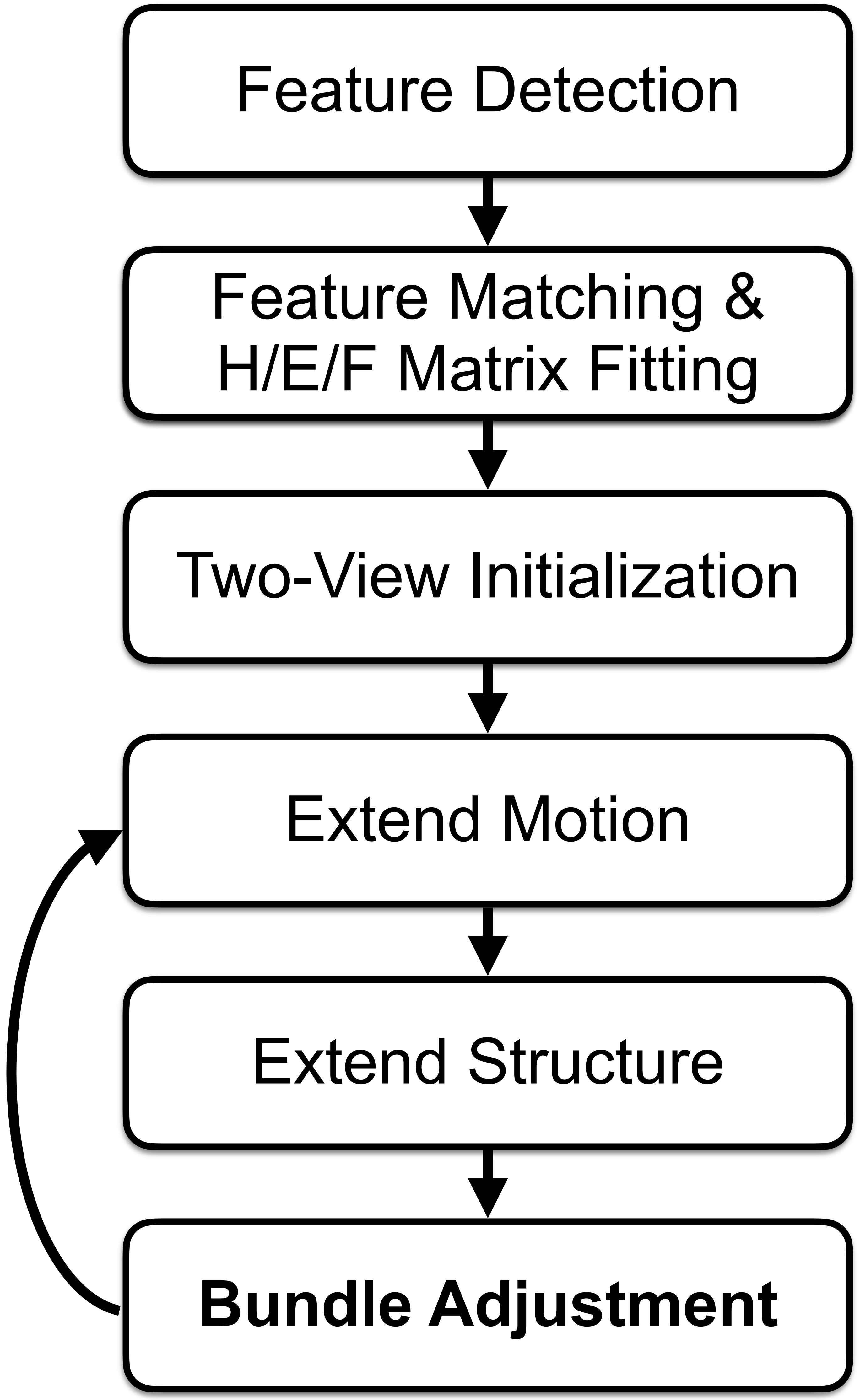



slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}, \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$



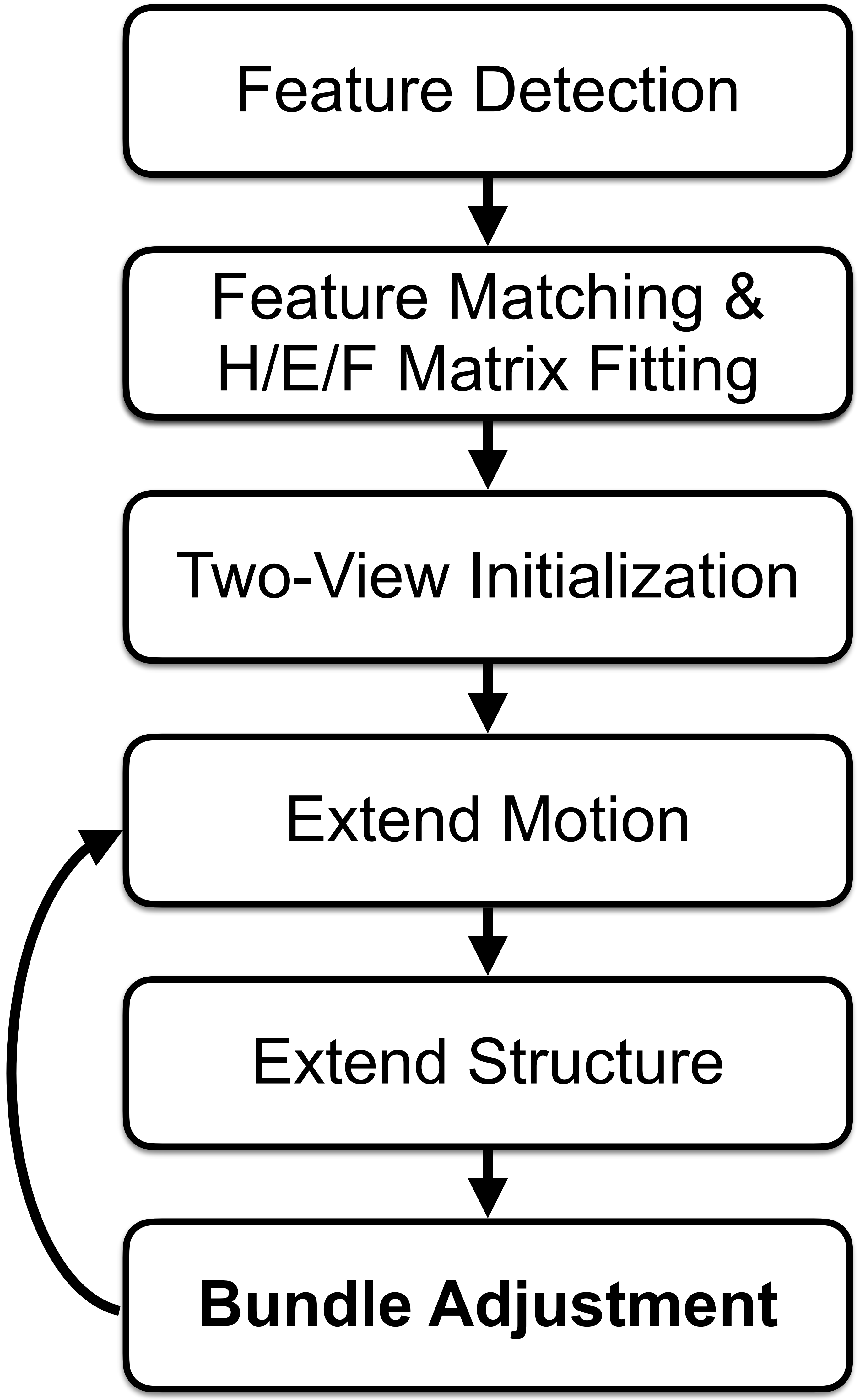

slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \;,\; \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ,\; \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$




slide credit: Gim Hee Lee

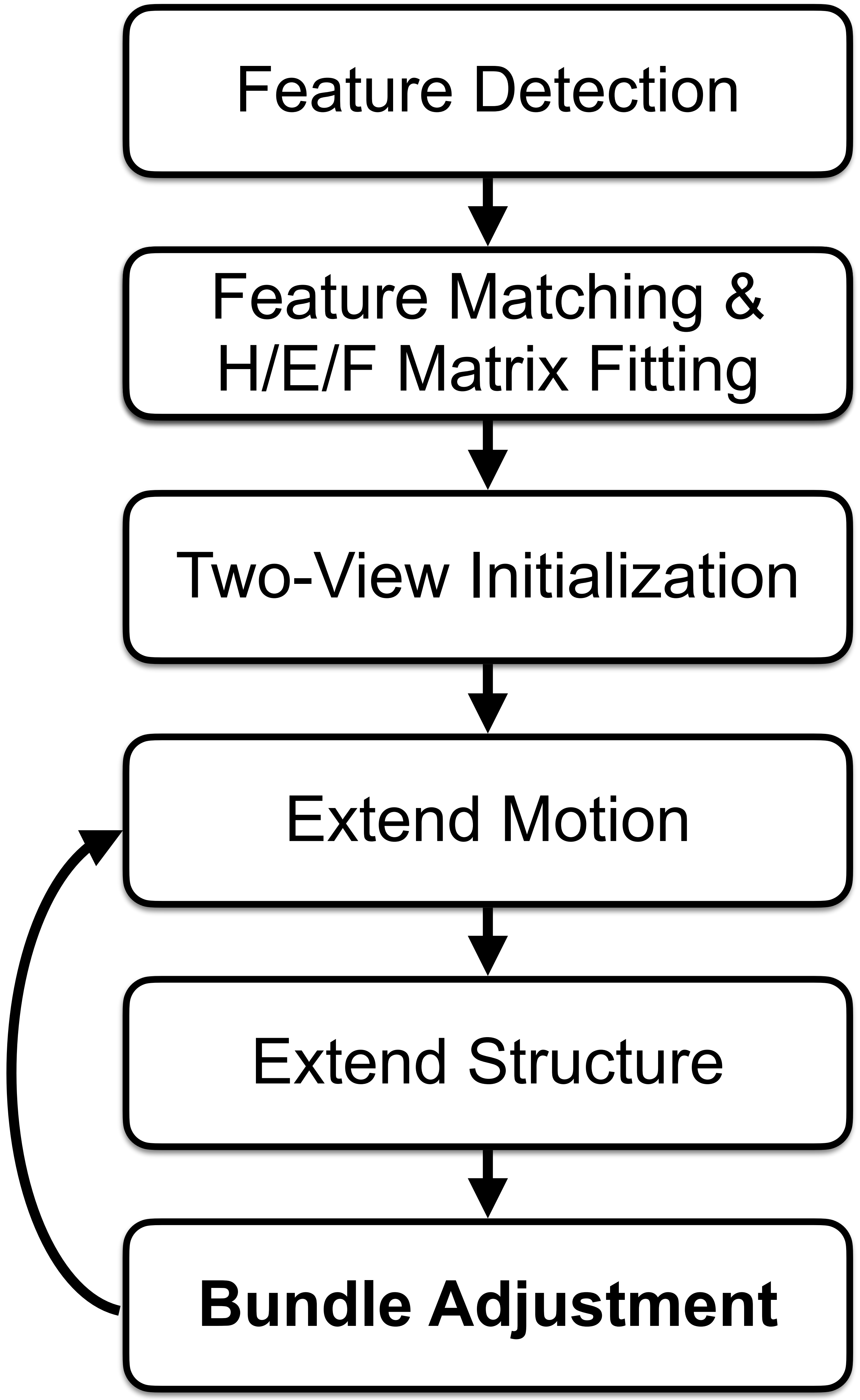
# Bundle Adjustment

**Gradient descent**

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$




slide credit: Gim Hee Lee

# Bundle Adjustment

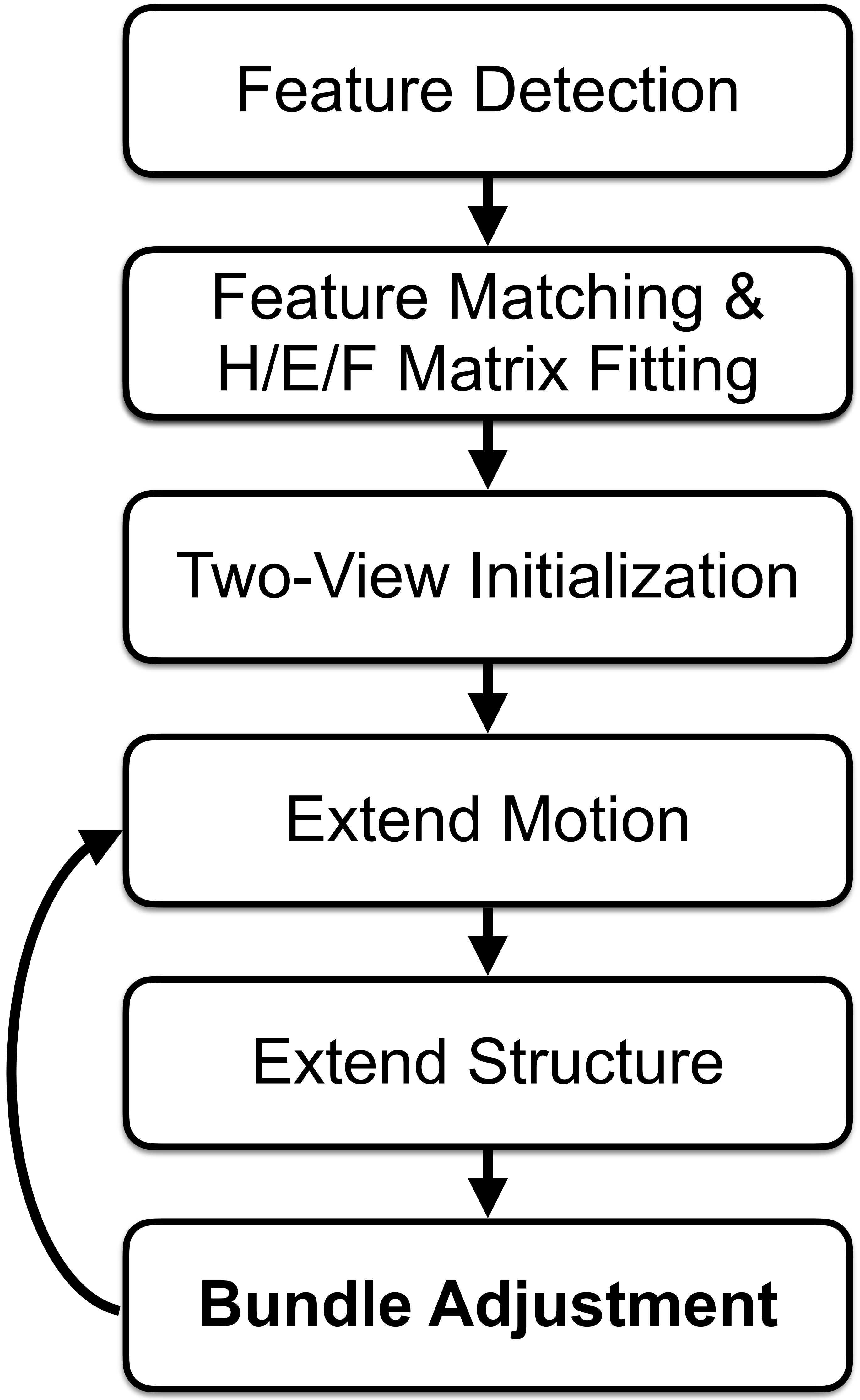
## Gradient descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \;,\; \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} ,\; \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ : Jacobian




slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

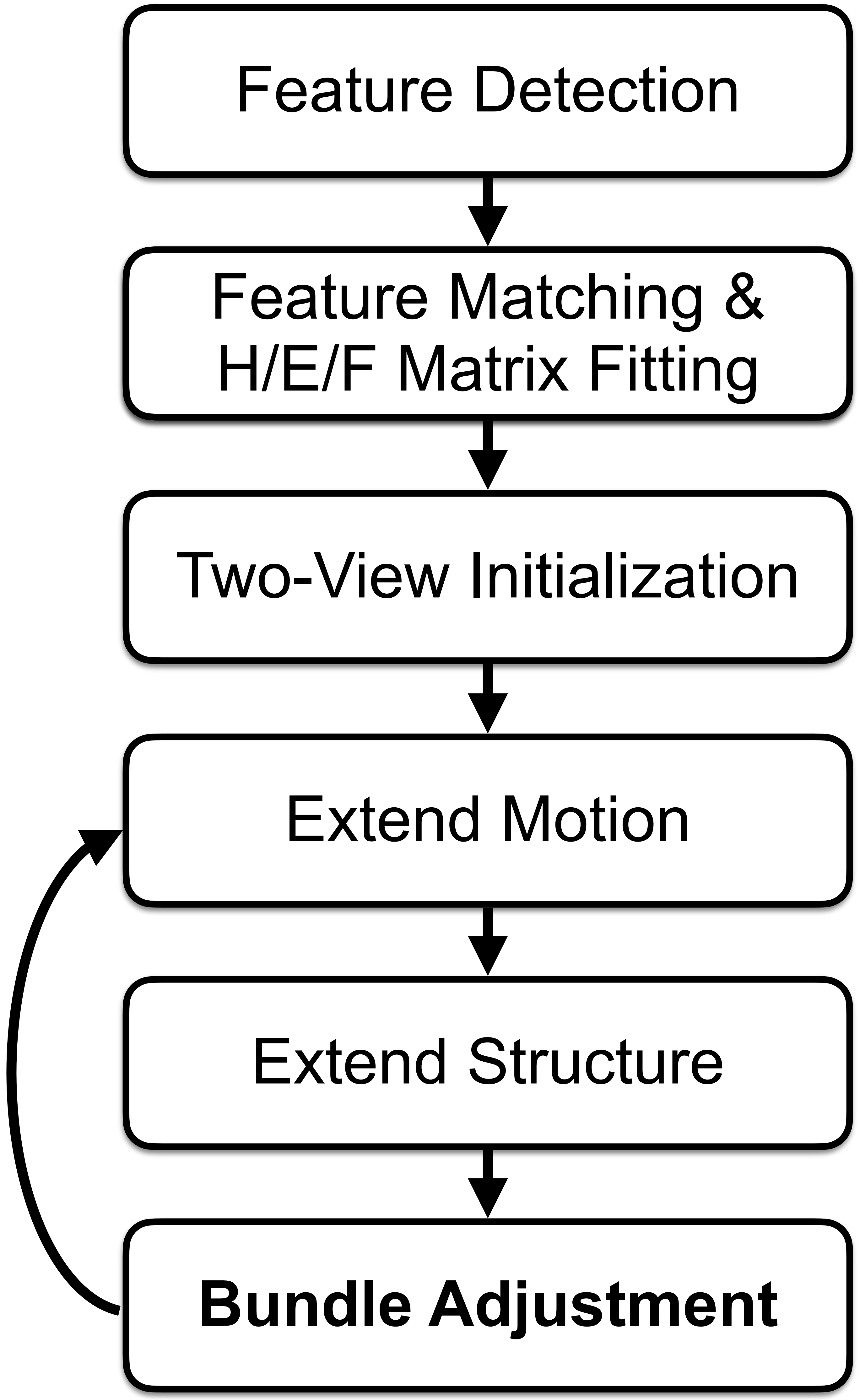
$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \;, \; \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}, \; \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ : Jacobian




slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

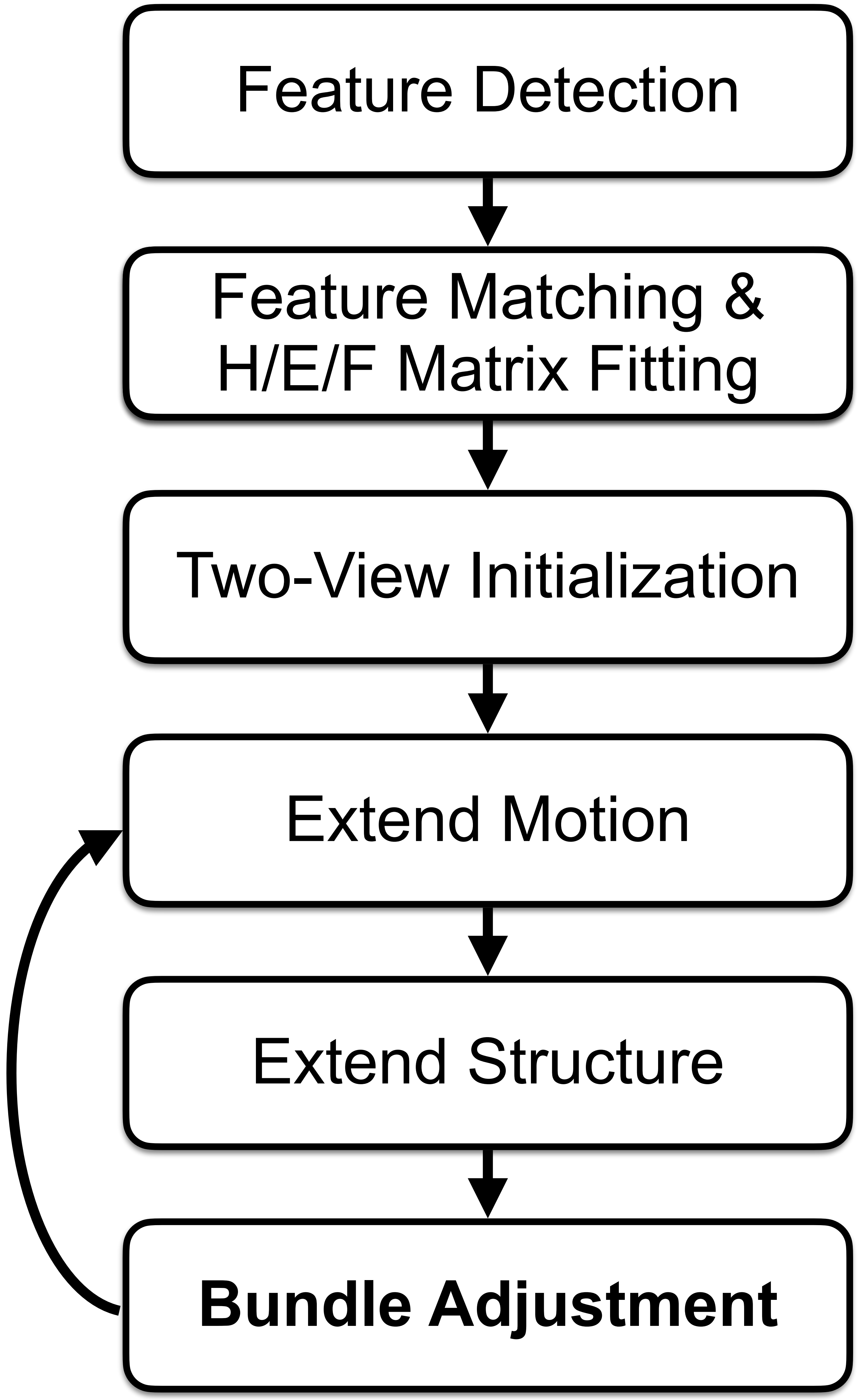
$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$

Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ : Jacobian $\quad \eta$ : Step size




slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

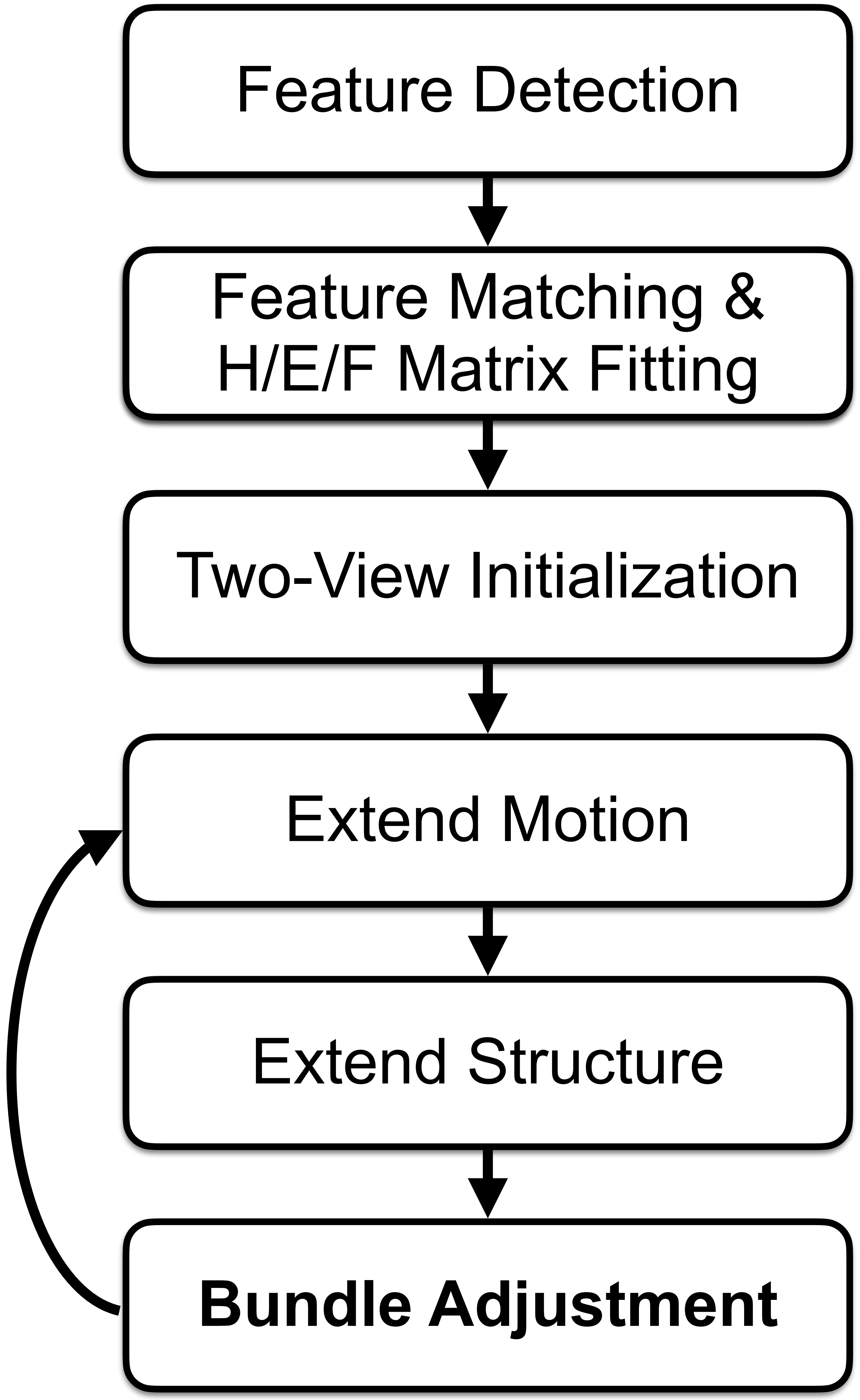
$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix}, \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Iterate until convergence or for fixed number of iterations

- Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$
- Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ : Jacobian    $\eta$ : Step size




slide credit: Gim Hee Lee

# Bundle Adjustment

## Gradient descent

$$\min_{\mathbf{X}} f(\mathbf{X}) = \min_{\mathbf{X}} \sum_i \Delta_i^T \Delta_i \ , \ \Delta_i = \mathbf{x}_i - \begin{pmatrix} \frac{\mathbf{P}_1^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \\ \frac{\mathbf{P}_2^i \mathbf{X}}{\mathbf{P}_3^i \mathbf{X}} \end{pmatrix} , \ \Delta = \begin{pmatrix} \Delta_1 \\ \vdots \\ \Delta_n \end{pmatrix}$$

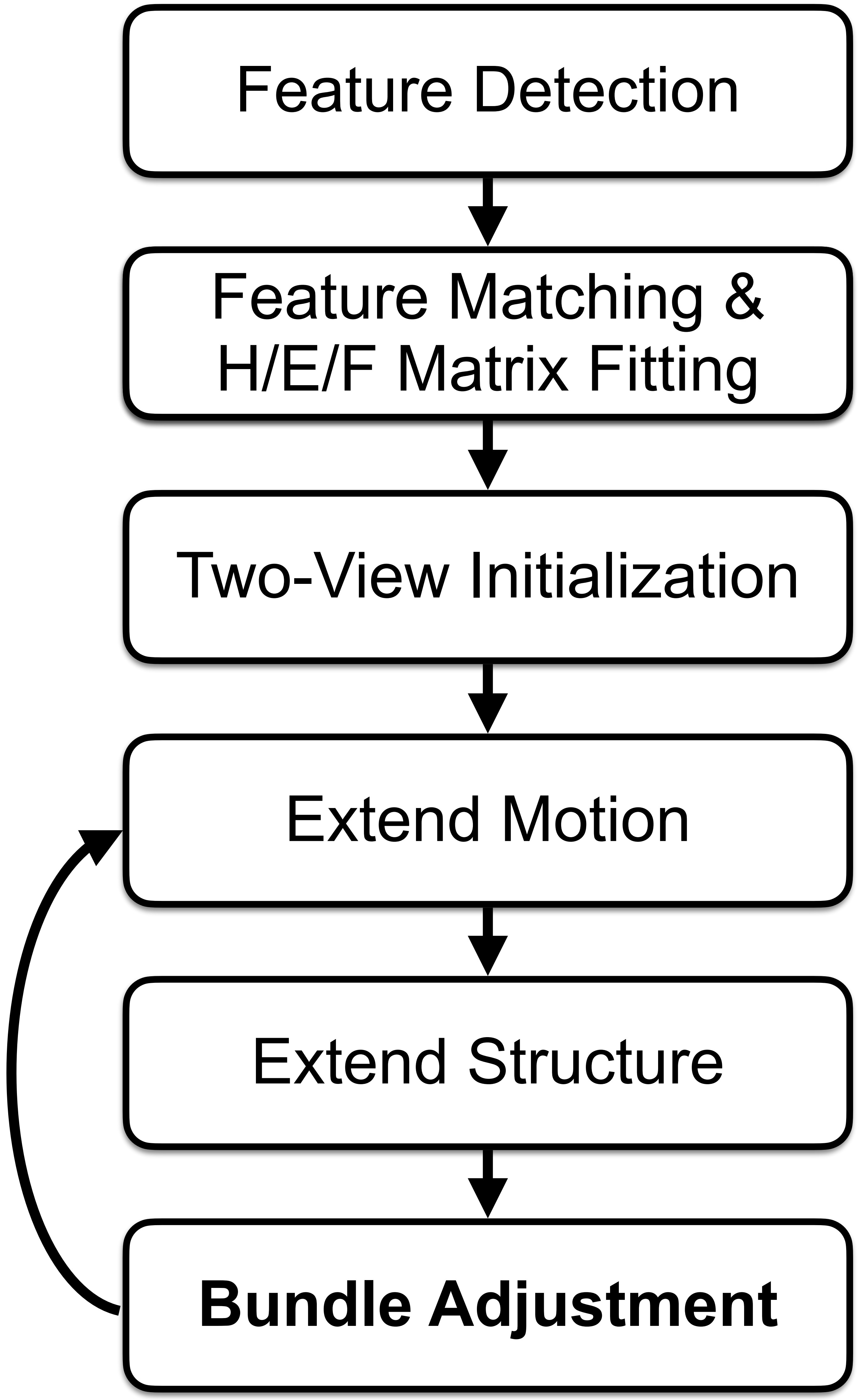
Initialization: $\mathbf{X}_k = \mathbf{X}_0$

Iterate until convergence or for fixed number of iterations

- Compute gradient: $\nabla f(\mathbf{X}_k) = \mathrm{J}^T \Delta$
- Update: $\mathbf{X}_{k+1} = \mathbf{X}_k - \eta \nabla f(\mathbf{X}_k)$

$\mathrm{J} = \frac{\partial f(\mathbf{X})}{\partial \mathbf{X}}$ : Jacobian $\quad \eta$ : Step size

Slow convergence near minimum point!



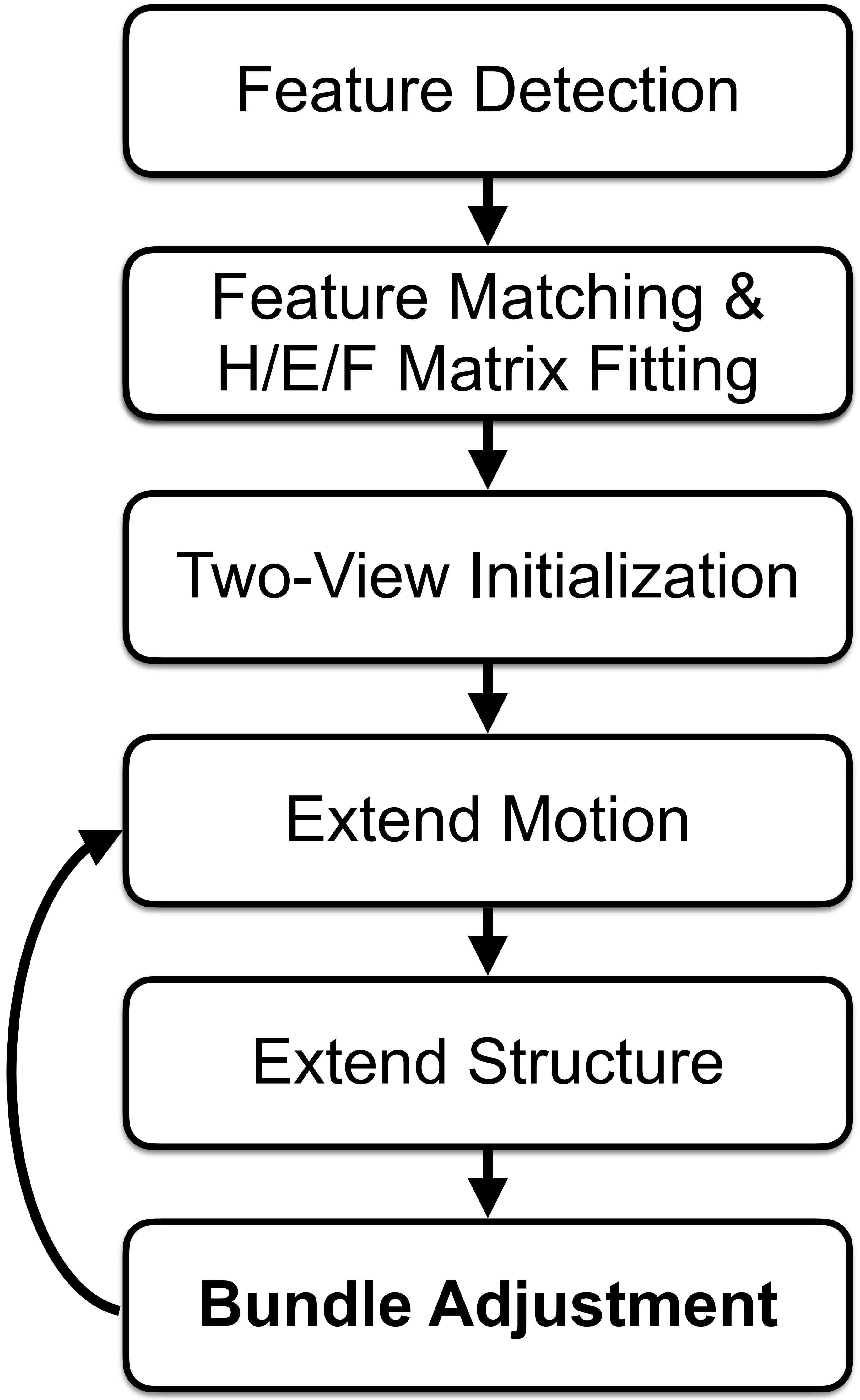

slide credit: Gim Hee Lee

# Bundle Adjustment

**Newton's method**

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathtt{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$




slide credit: Gim Hee Lee

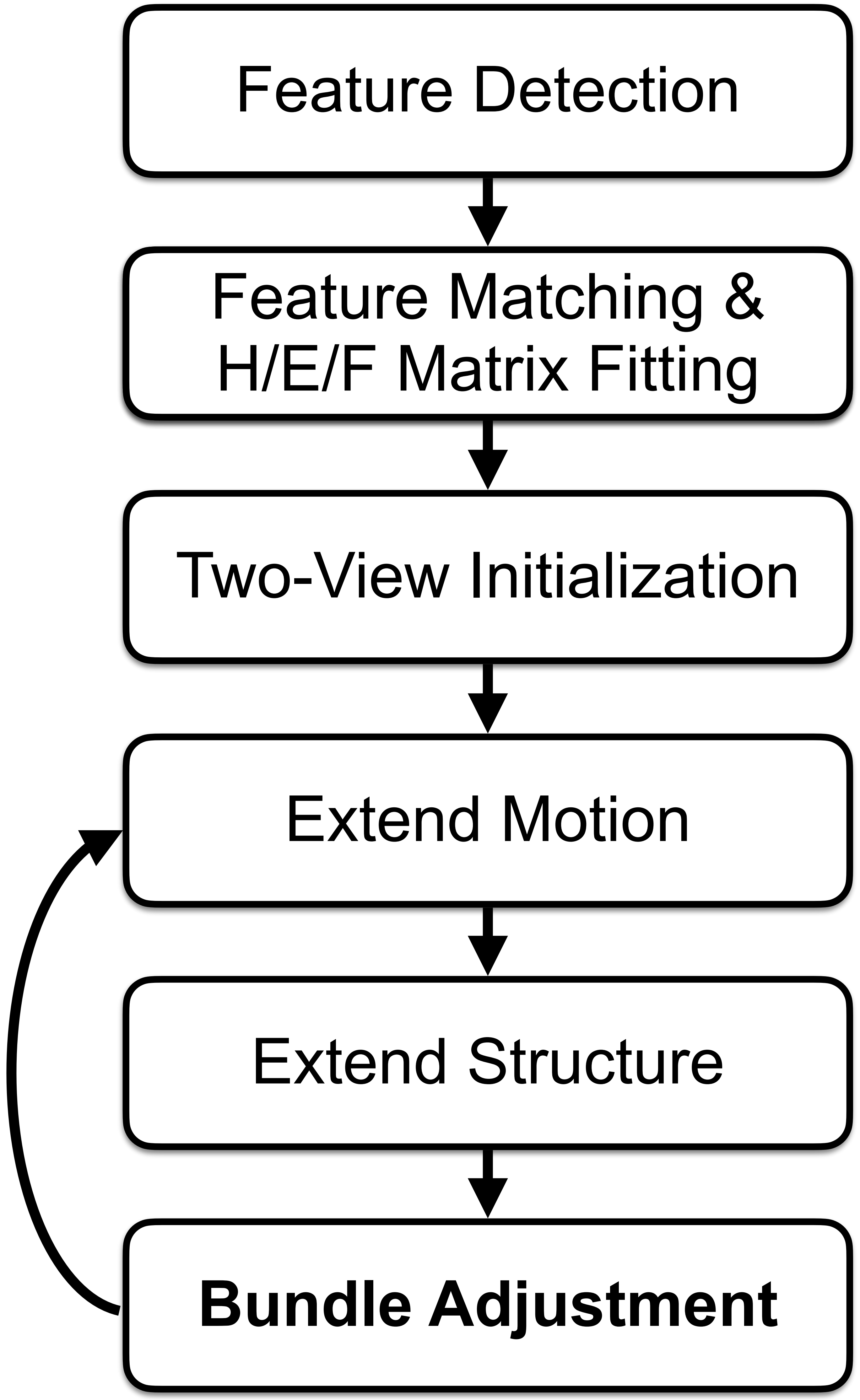
# Bundle Adjustment

## Newton's method

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathtt{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$

Hessian matrix: $$\mathtt{H} = \frac{\partial^2 f(\mathbf{X}+\delta)}{\partial^2 \delta}\Big|_{\mathbf{X}=\mathbf{X}_k}$$




slide credit: Gim Hee Lee

# Bundle Adjustment

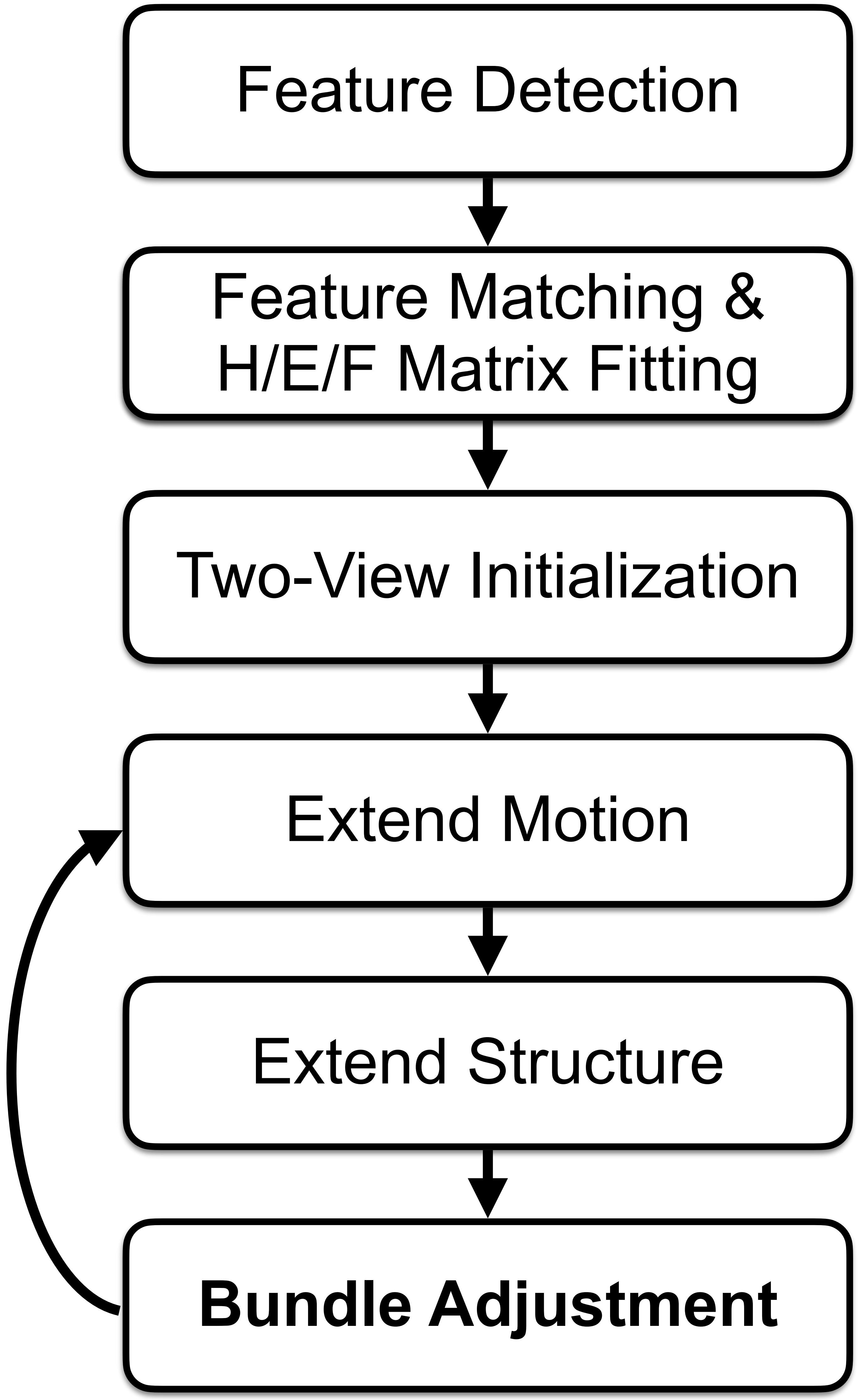
**Newton's method**

2nd order approximation (quadratic Taylor expansion):

$$f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k} = f(\mathbf{X}) + \nabla f(\mathbf{X})^T \delta + \frac{1}{2}\delta^T \mathtt{H} \delta \Big|_{\mathbf{X}=\mathbf{X}_k}$$

Hessian matrix: $$\mathtt{H} = \frac{\partial^2 f(\mathbf{X}+\delta)}{\partial^2 \delta}\Big|_{\mathbf{X}=\mathbf{X}_k}$$

Find $\delta$ that minimizes $f(\mathbf{X}+\delta)|_{\mathbf{X}=\mathbf{X}_k}$!




slide credit: Gim Hee Lee

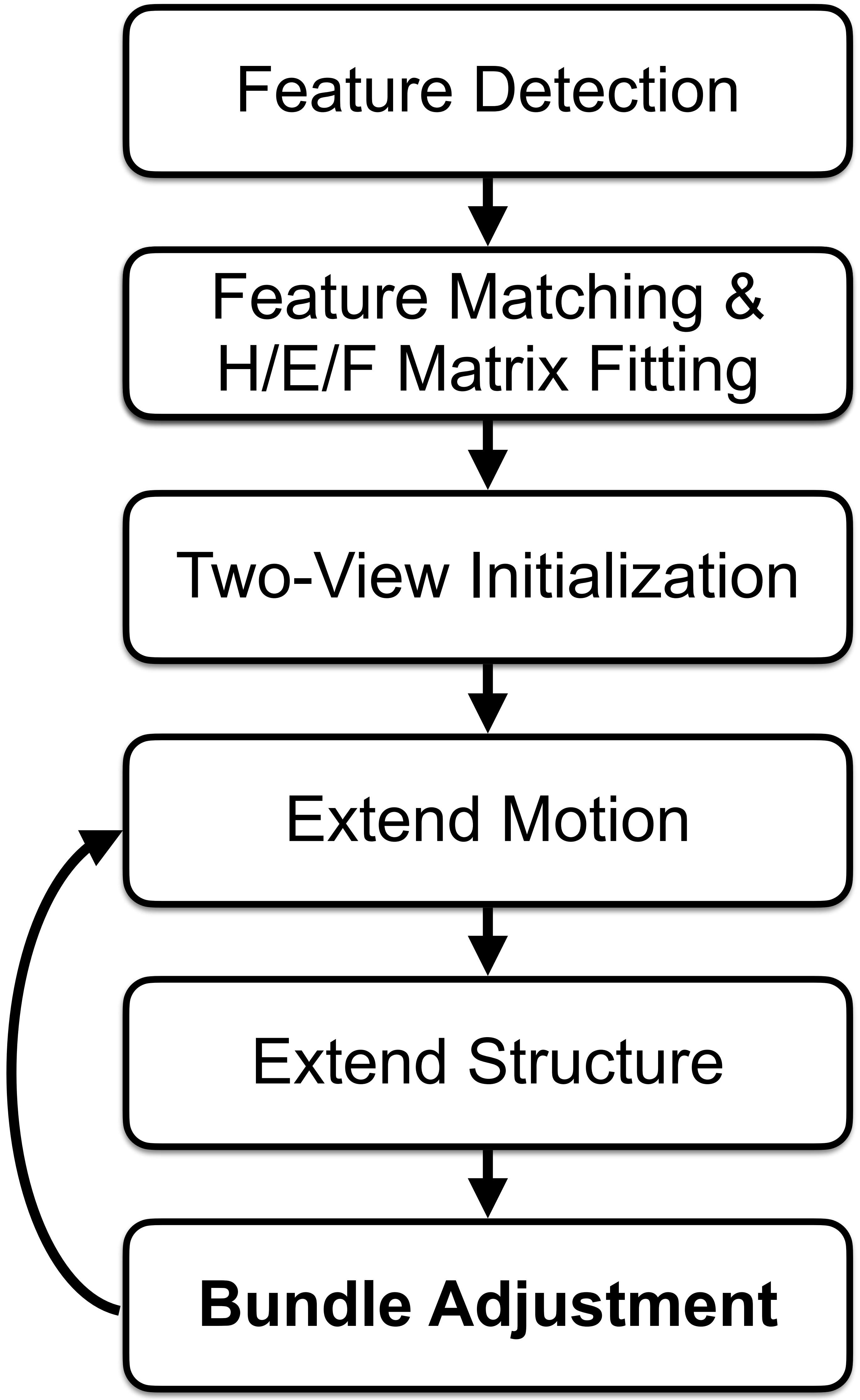
# Bundle Adjustment

**Newton's method**

Differentiate and set to 0 gives:

$$\delta = -\mathtt{H}^{-1}\nabla f(\mathbf{X}_k)$$

Update: $$\mathbf{X}_{k+1} = \mathbf{X}_k + \delta$$




slide credit: Gim Hee Lee

# Bundle Adjustment

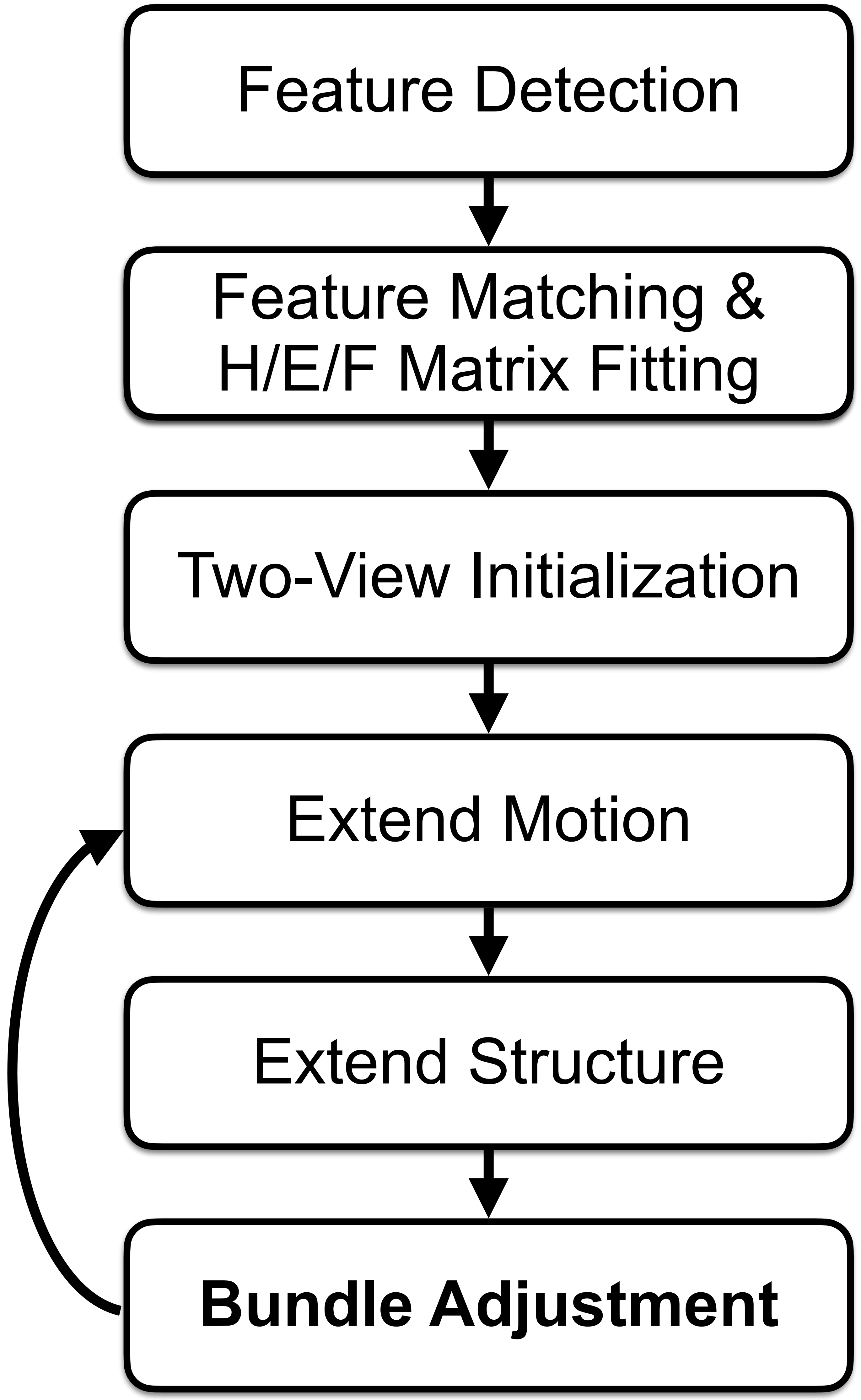
**Newton's method**

Differentiate and set to 0 gives:

$$\delta = -\mathtt{H}^{-1} \nabla f(\mathbf{X}_k)$$

Update: $$\mathbf{X}_{k+1} = \mathbf{X}_k + \delta$$

Computation of H is not trivial (2nd order derivatives) and optimization might get stuck at saddle point!



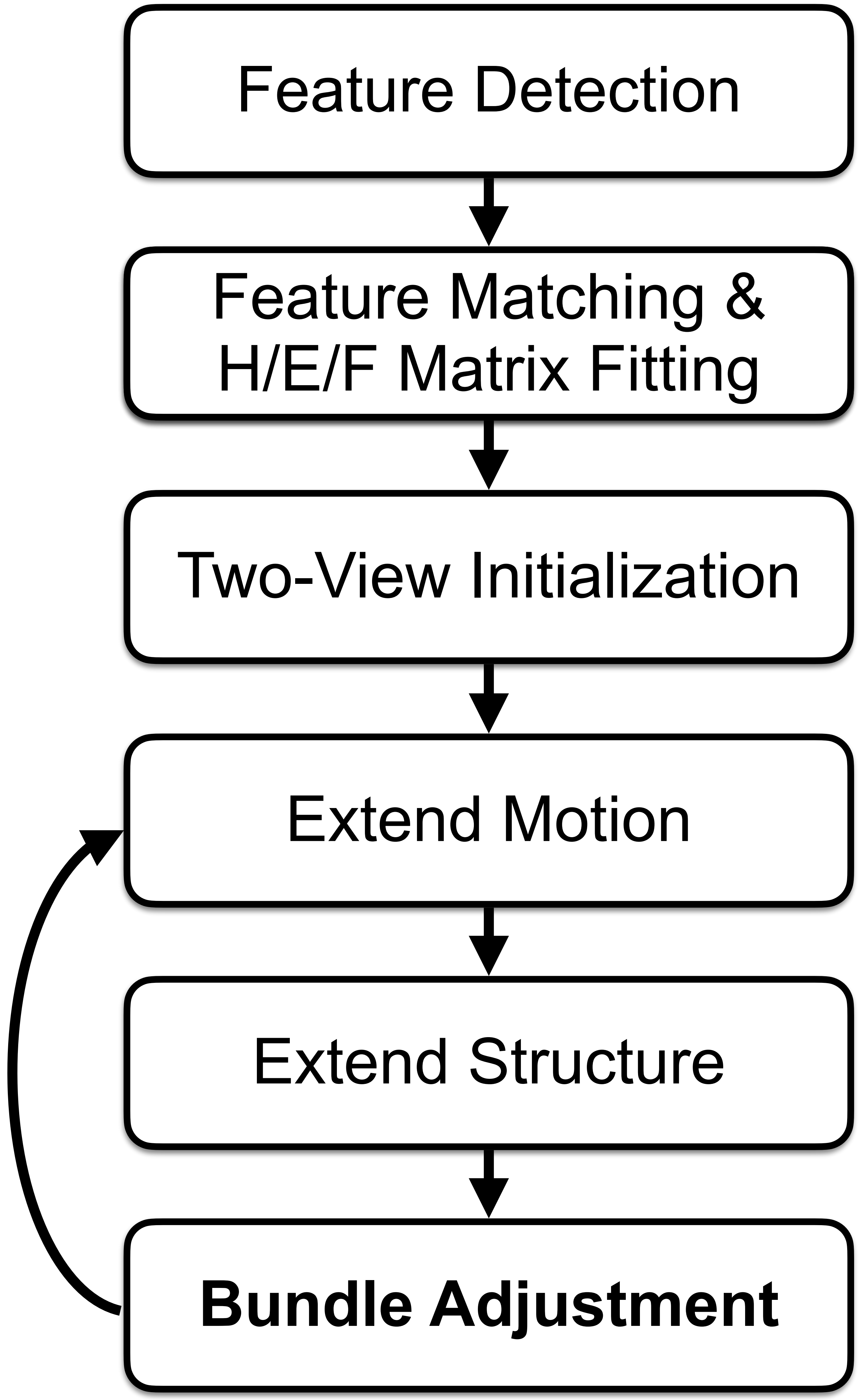

slide credit: Gim Hee Lee

# Bundle Adjustment

## Gauss-Newton

Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$




slide credit: Gim Hee Lee

# Bundle Adjustment

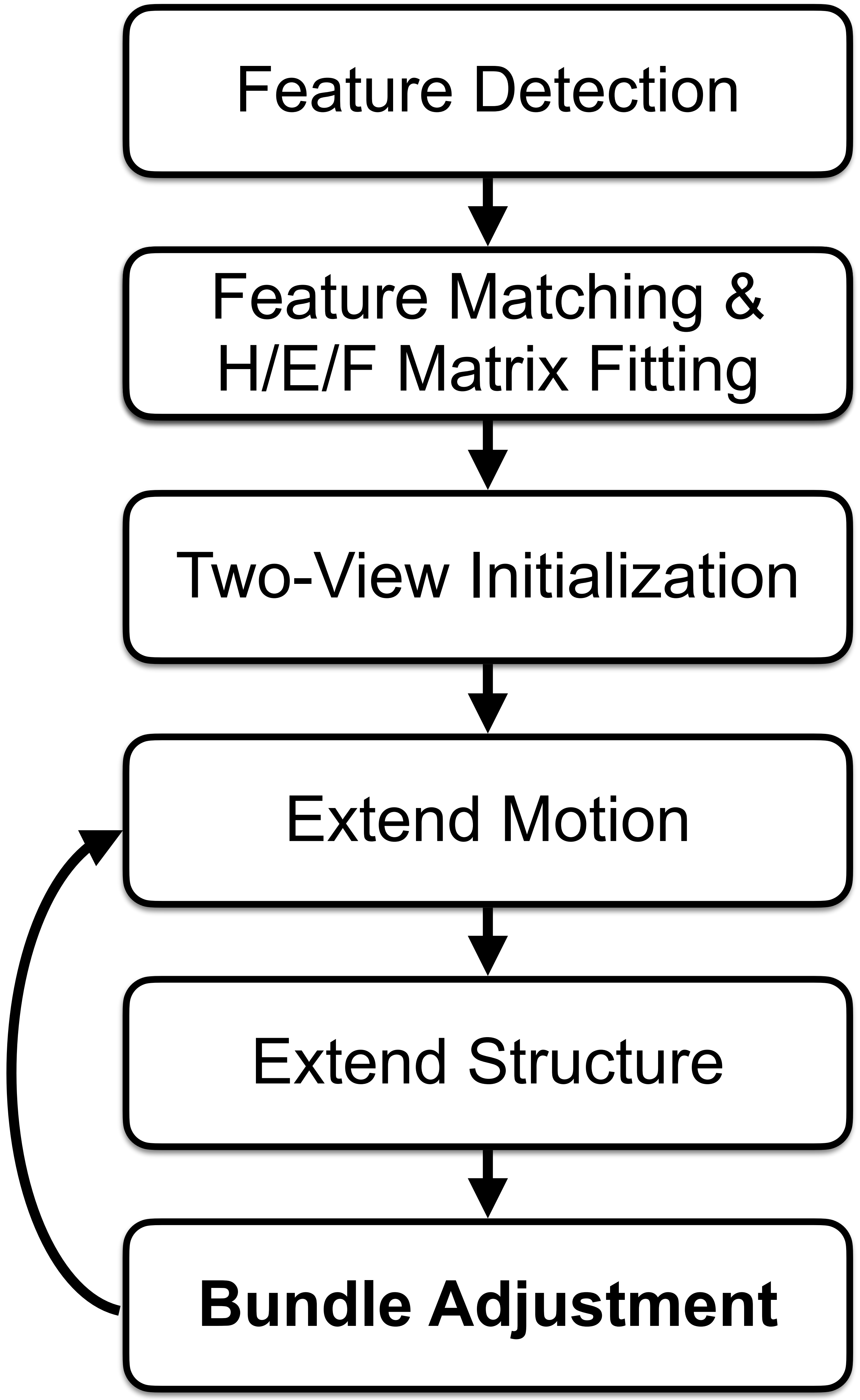
## Gauss-Newton

Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$

Solve normal equation:

$$\mathtt{J}^T \mathtt{J}\delta = -\mathtt{J}^t \Delta$$




slide credit: Gim Hee Lee

# Bundle Adjustment

**Gauss-Newton**

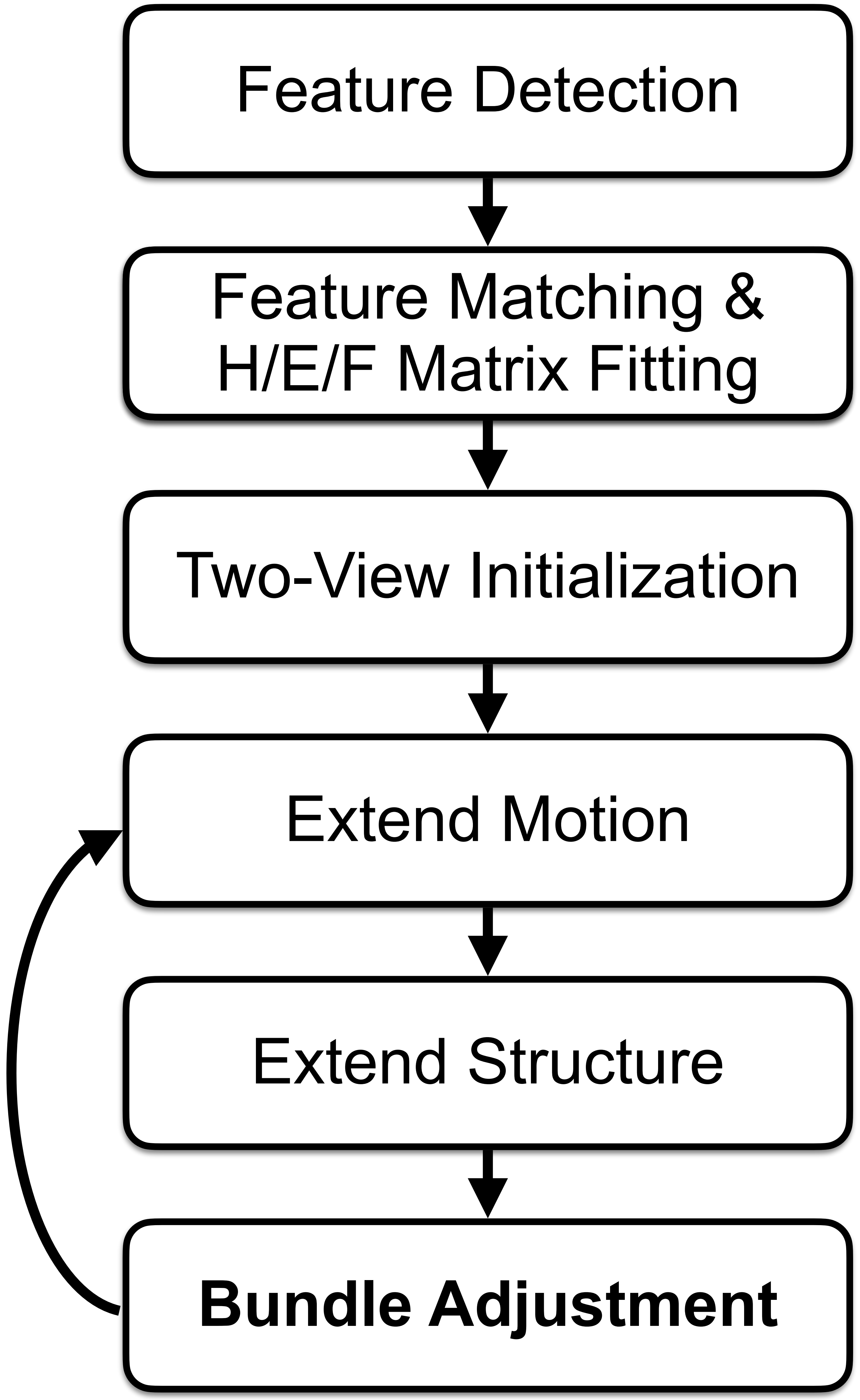
Approximate Hessian matrix by dropping 2nd order terms:

$$\mathtt{H} \approx \mathtt{J}^T \mathtt{J}$$

Solve normal equation:

$$\mathtt{J}^T \mathtt{J} \delta = -\mathtt{J}^t \Delta$$

Might get stuck and slow convergence at saddle point!



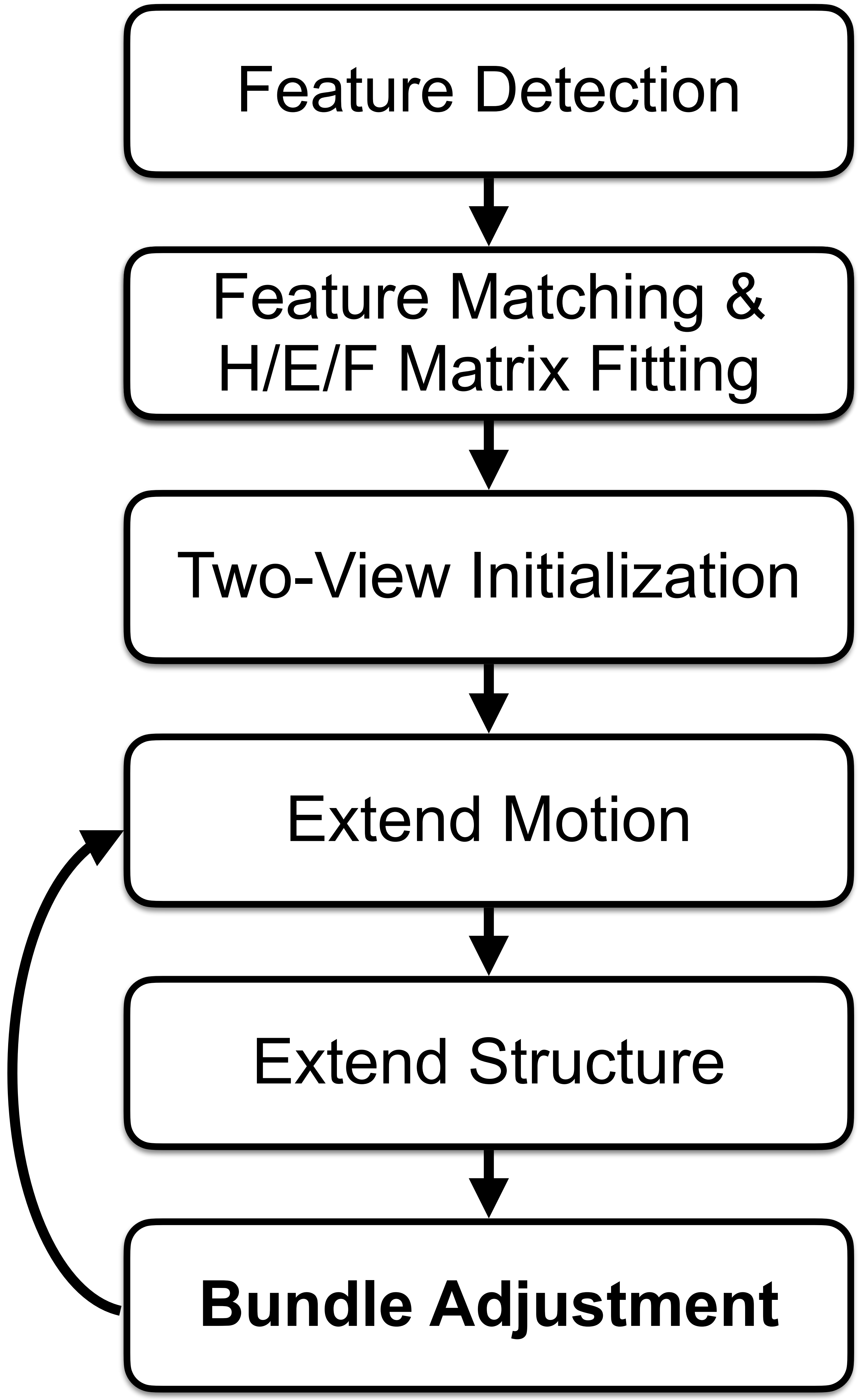

slide credit: Gim Hee Lee

# Bundle Adjustment

## Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$




slide credit: Gim Hee Lee

# Bundle Adjustment

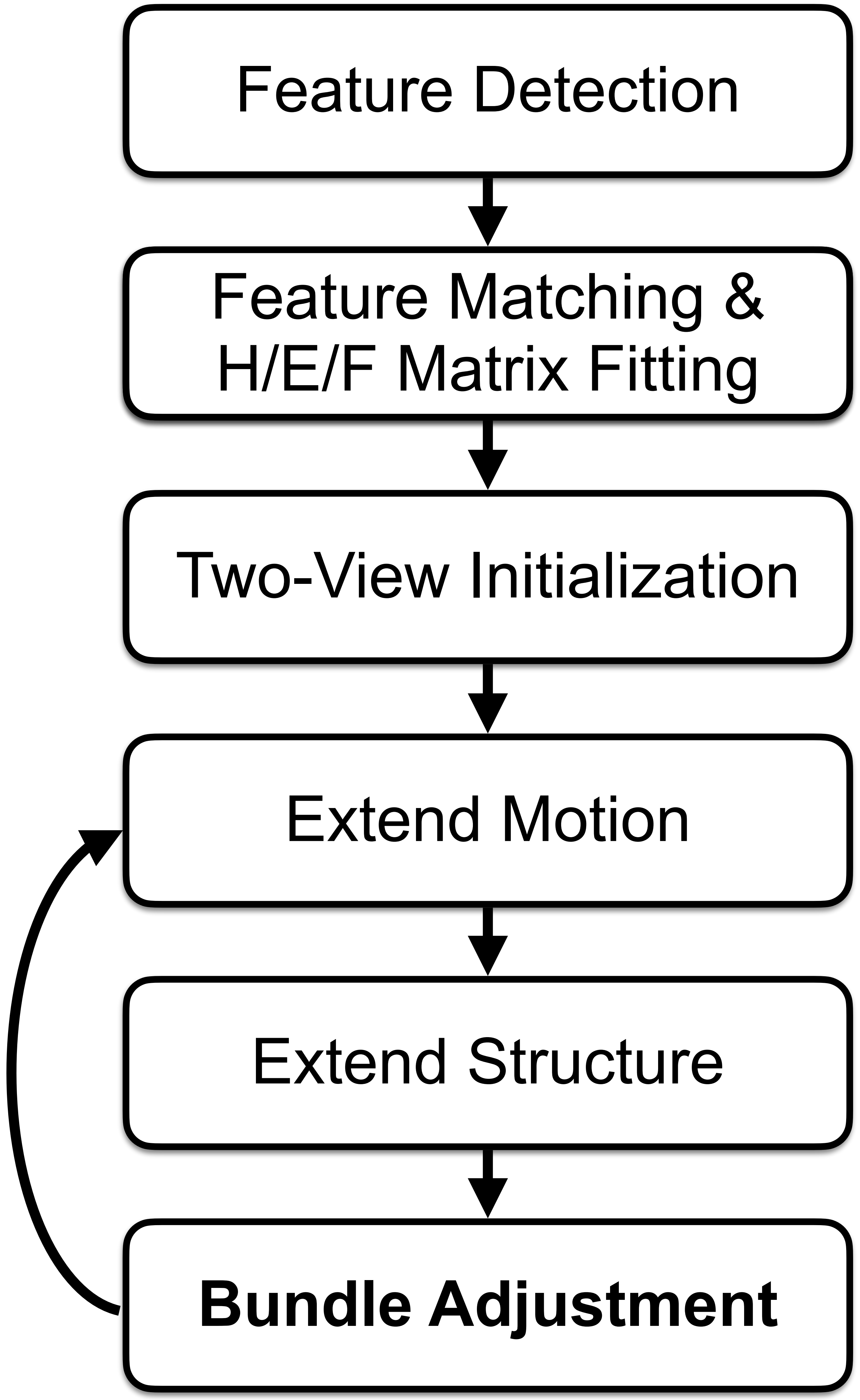
## Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

$\lambda \rightarrow 0$: Gauss-Newton (when convergence is rapid)

$\lambda \rightarrow \infty$: Gradient descent (when convergence is slow)




slide credit: Gim Hee Lee

# Bundle Adjustment

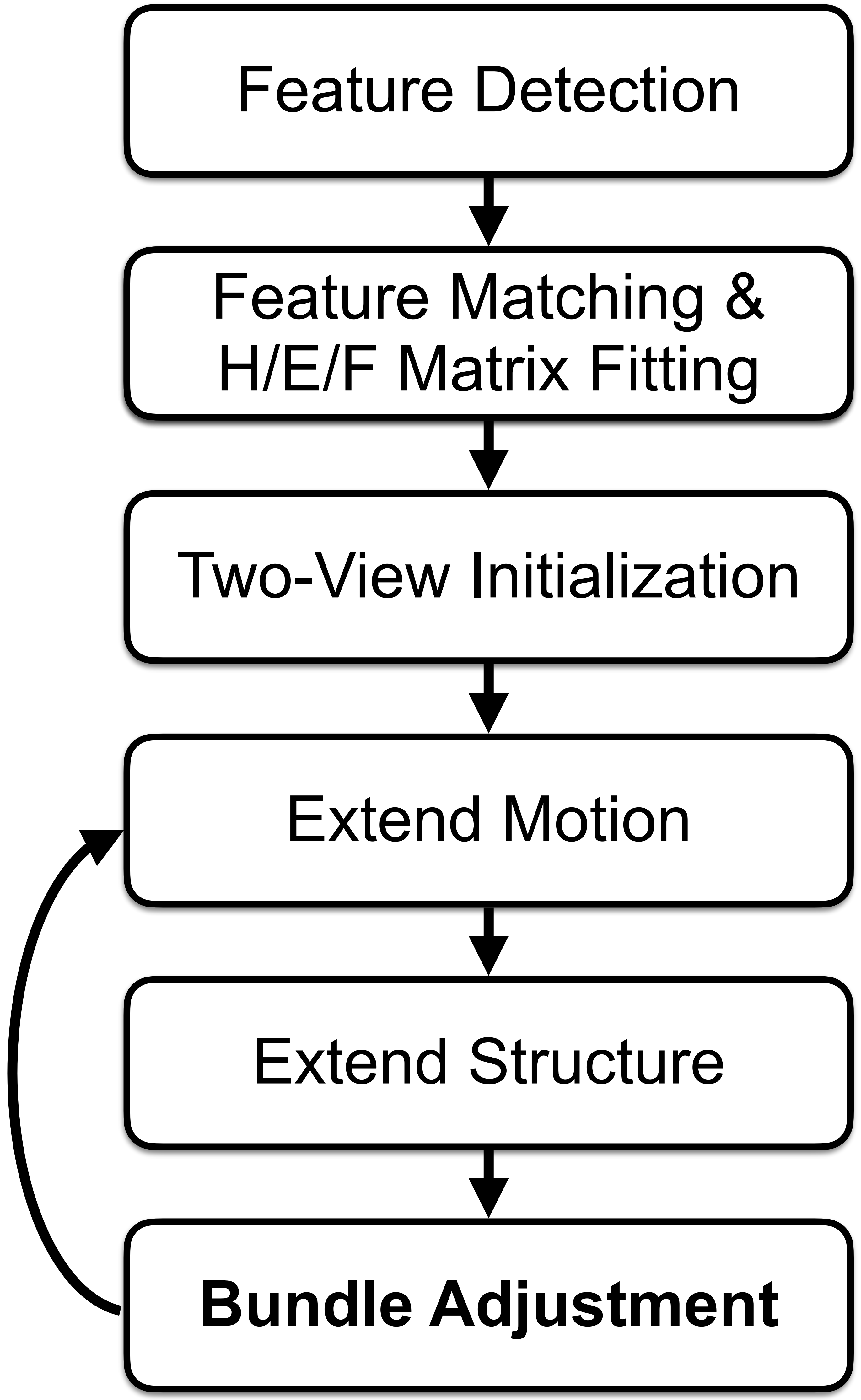
**Levenberg-Marquardt**

Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda\mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

$\lambda \to 0$: Gauss-Newton (when convergence is rapid)
$\lambda \to \infty$: Gradient descent (when convergence is slow)

Adapt λ during optimization:




slide credit: Gim Hee Lee

# Bundle Adjustment

**Levenberg-Marquardt**

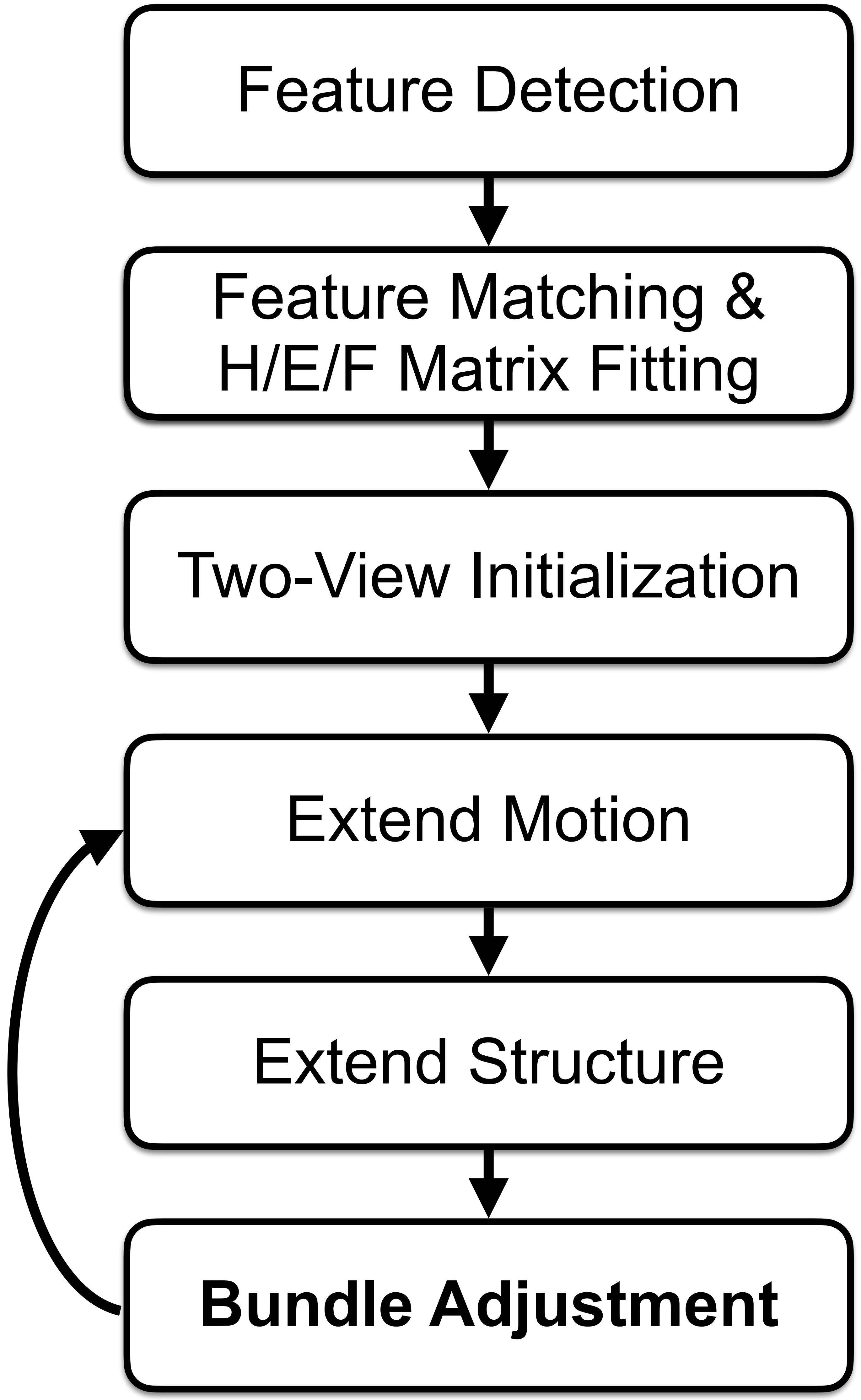
Regularized Gauss-Newton with damping factor

$$\left(\mathtt{J}^T\mathtt{J} + \lambda \mathtt{I}\right)\delta = -\mathtt{J}^t\Delta$$

$\lambda \rightarrow 0$: Gauss-Newton (when convergence is rapid)
$\lambda \rightarrow \infty$: Gradient descent (when convergence is slow)

Adapt λ during optimization:

- Decrease λ when function value decreases




slide credit: Gim Hee Lee

# Bundle Adjustment

## Levenberg-Marquardt

Regularized Gauss-Newton with damping factor

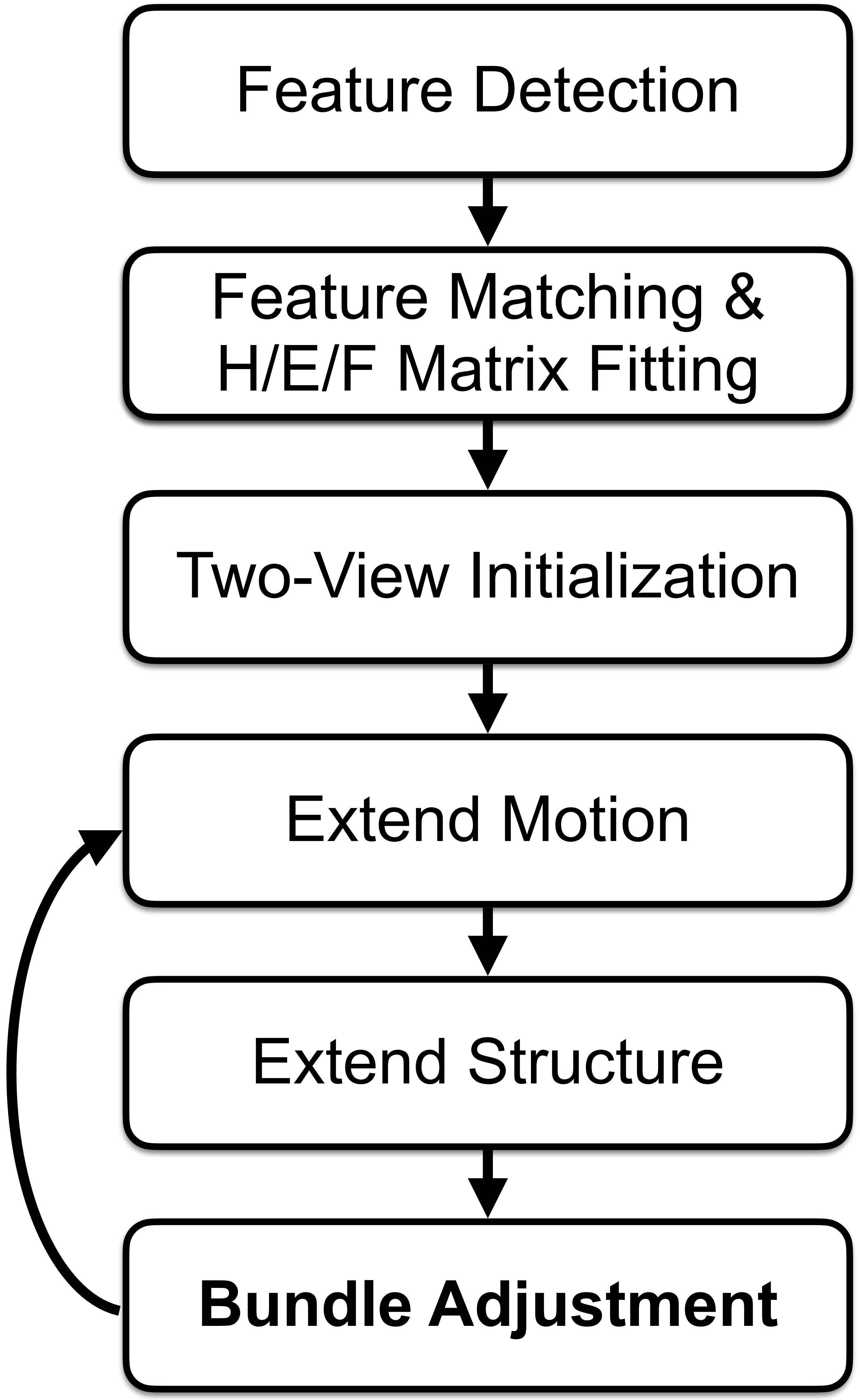
$$\left(J^T J + \lambda I\right) \delta = -J^t \Delta$$

$\lambda \to 0$: Gauss-Newton (when convergence is rapid)

$\lambda \to \infty$: Gradient descent (when convergence is slow)

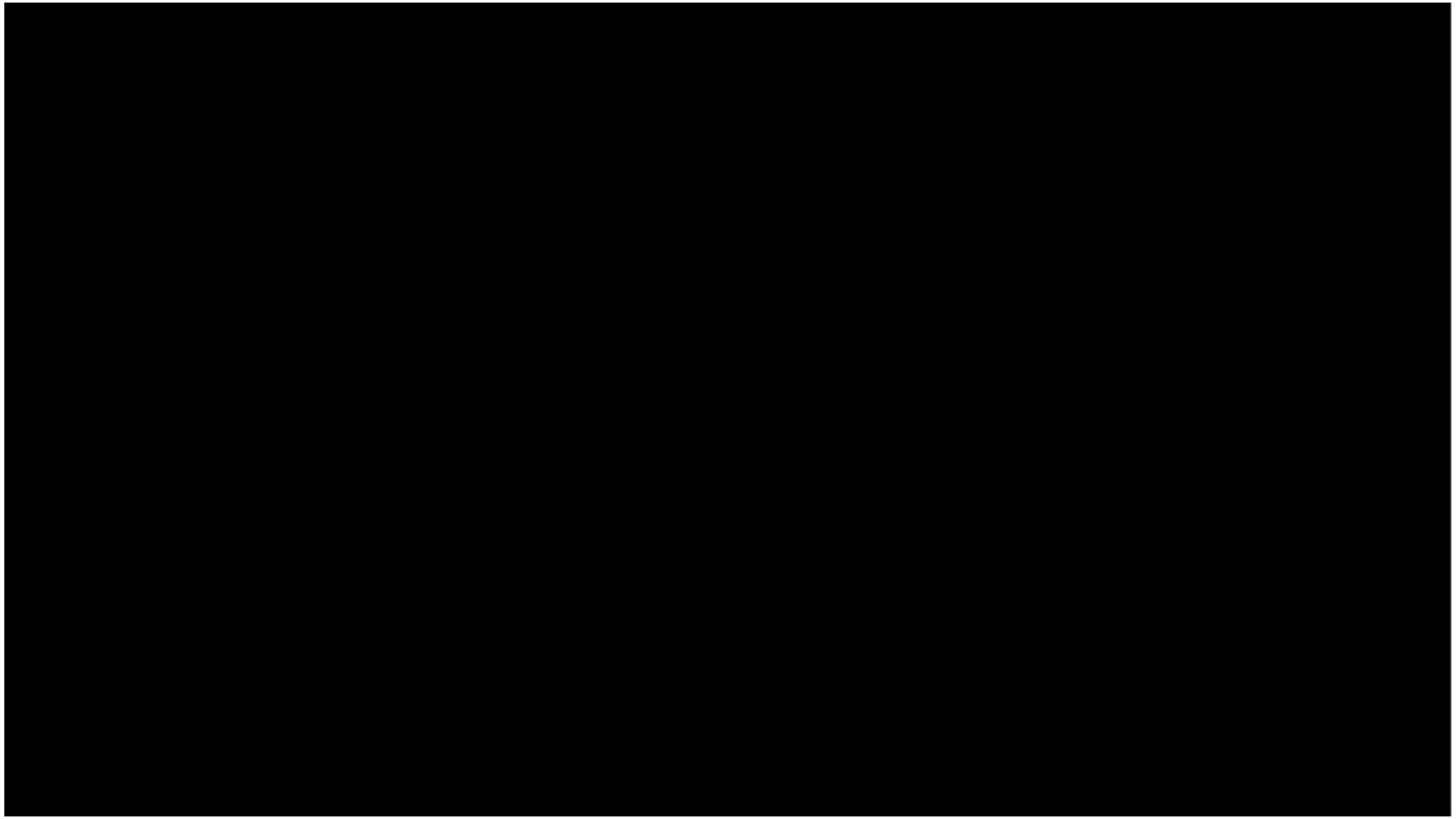
Adapt λ during optimization:

- Decrease λ when function value decreases
- Increase λ otherwise

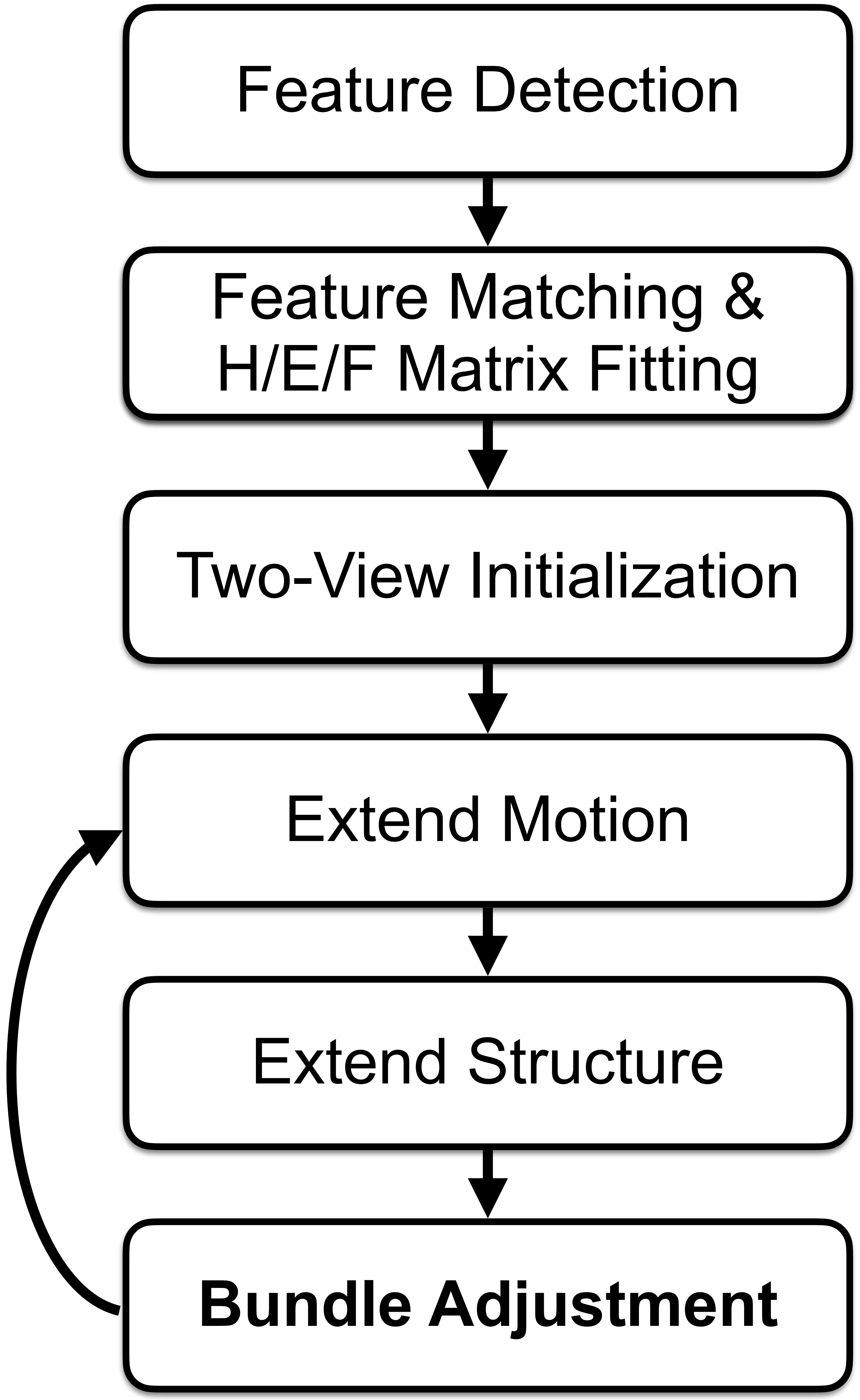



slide credit: Gim Hee Lee

# Bundle Adjustment



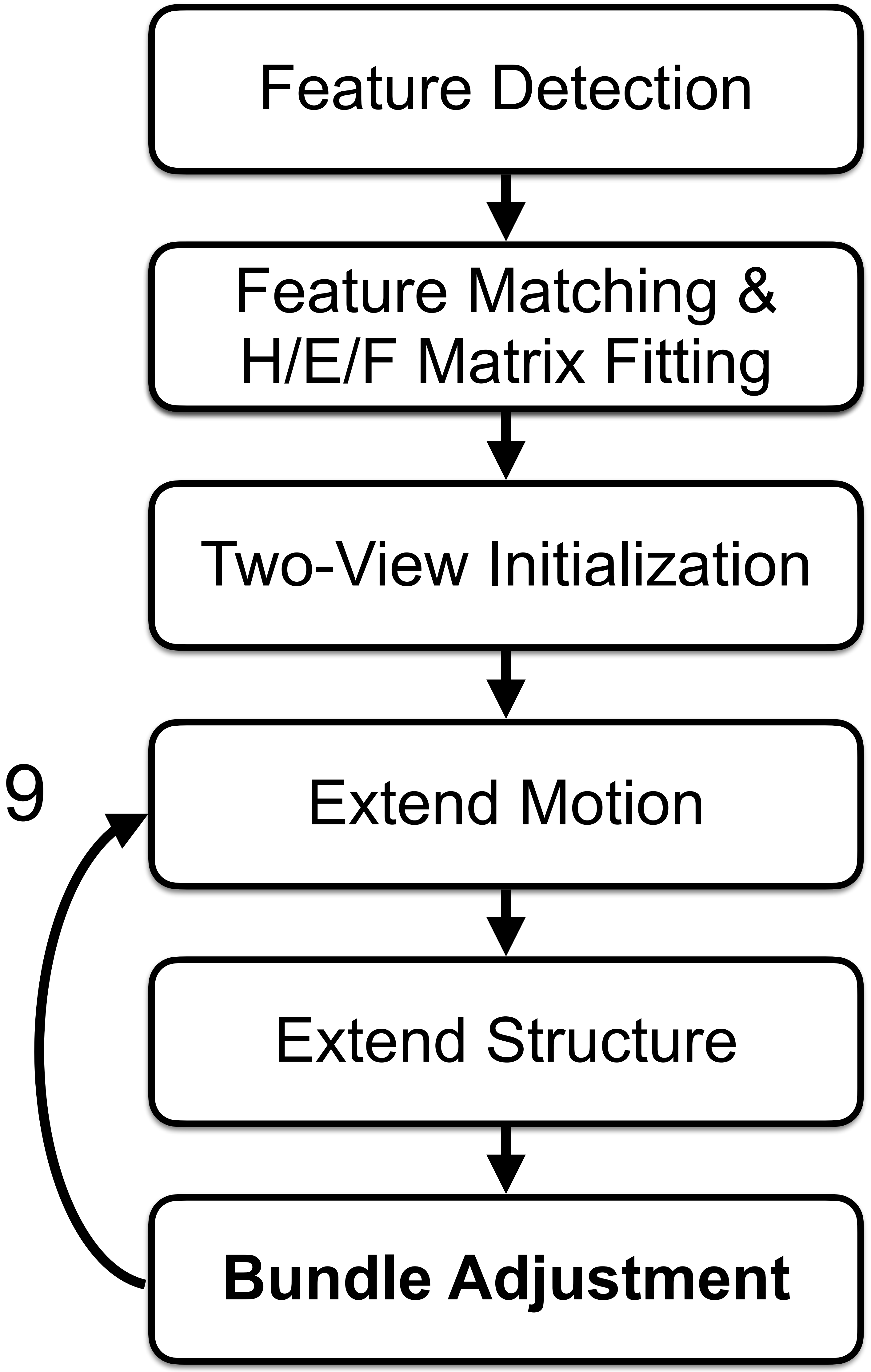
Reconstruction of the old inner city of Aachen, Germany, using the Bundler SfM software




slide credit: Gim Hee Lee

# Bundle Adjustment

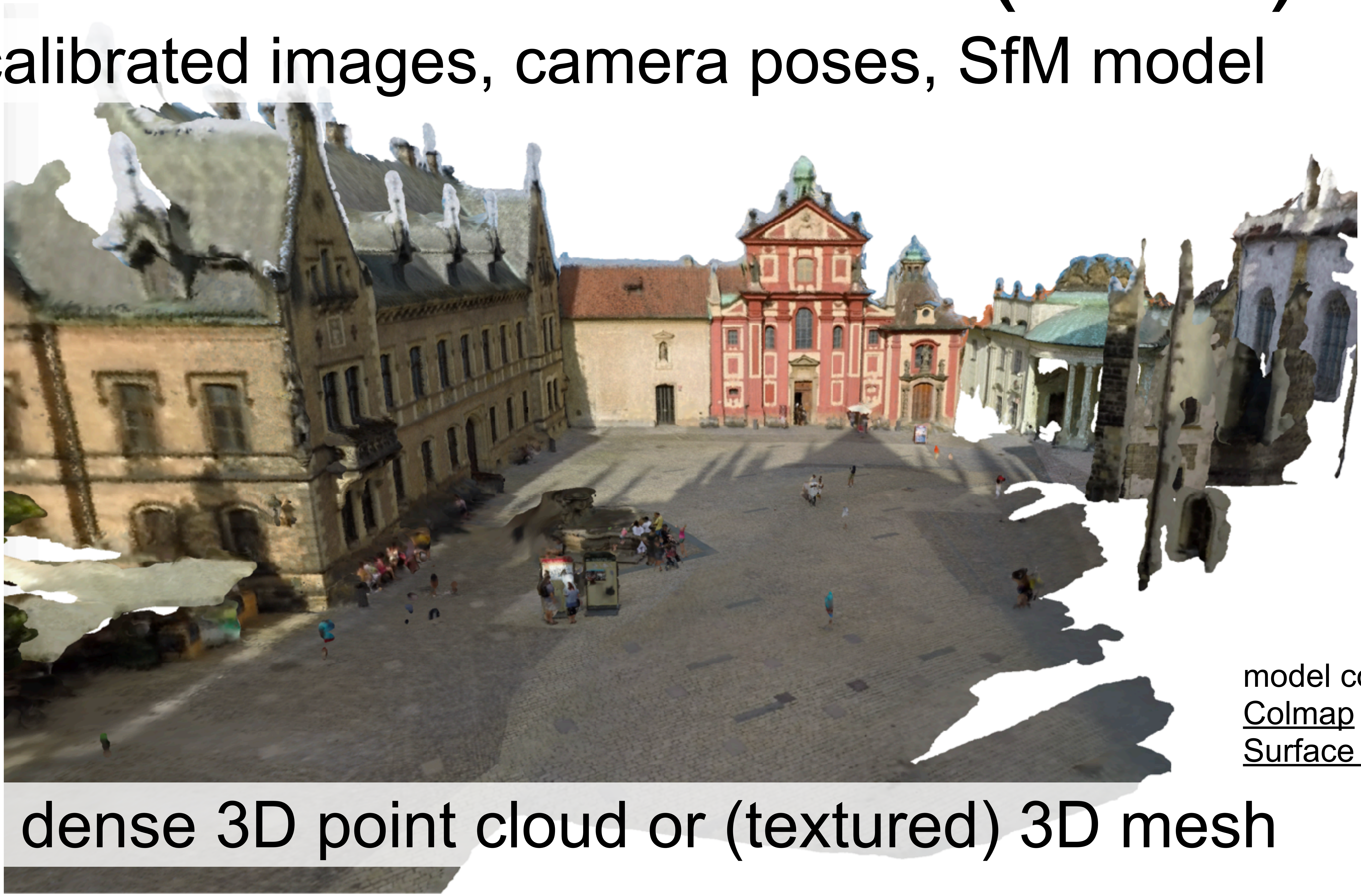
- Not covered here:
  - Sparse structure of the bundle adjustment problem
  - Efficient strategies (e.g., Schur Complement Trick)
  - …
- Recommended reading:
  - Triggs et al., Bundle Adjustment - A Modern Synthesis, 1999




slide credit: Gim Hee Lee

# Multi-View Stereo (MVS)

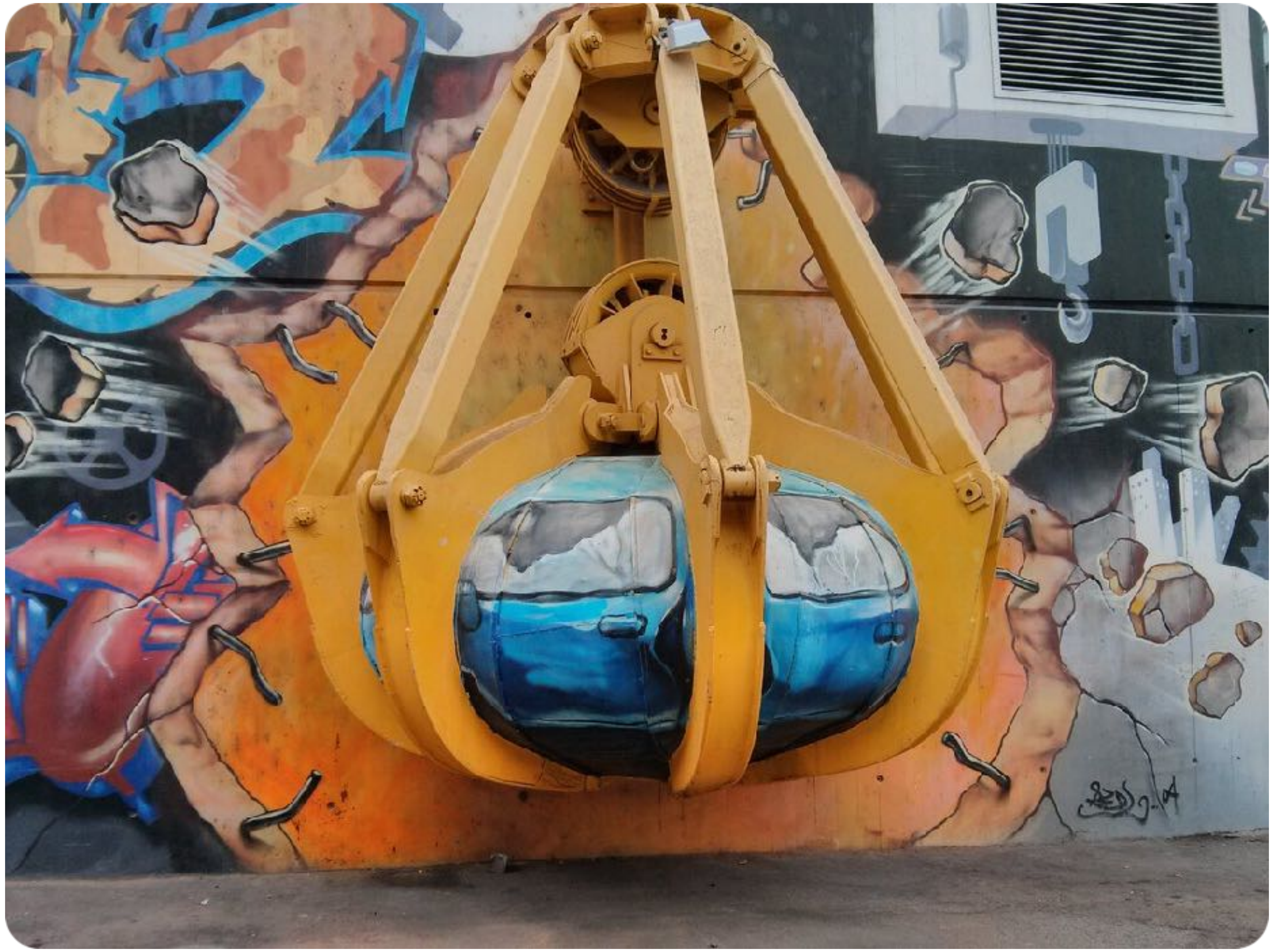
Input: calibrated images, camera poses, SfM model



model computed using Colmap and Poisson Surface Reconstruction

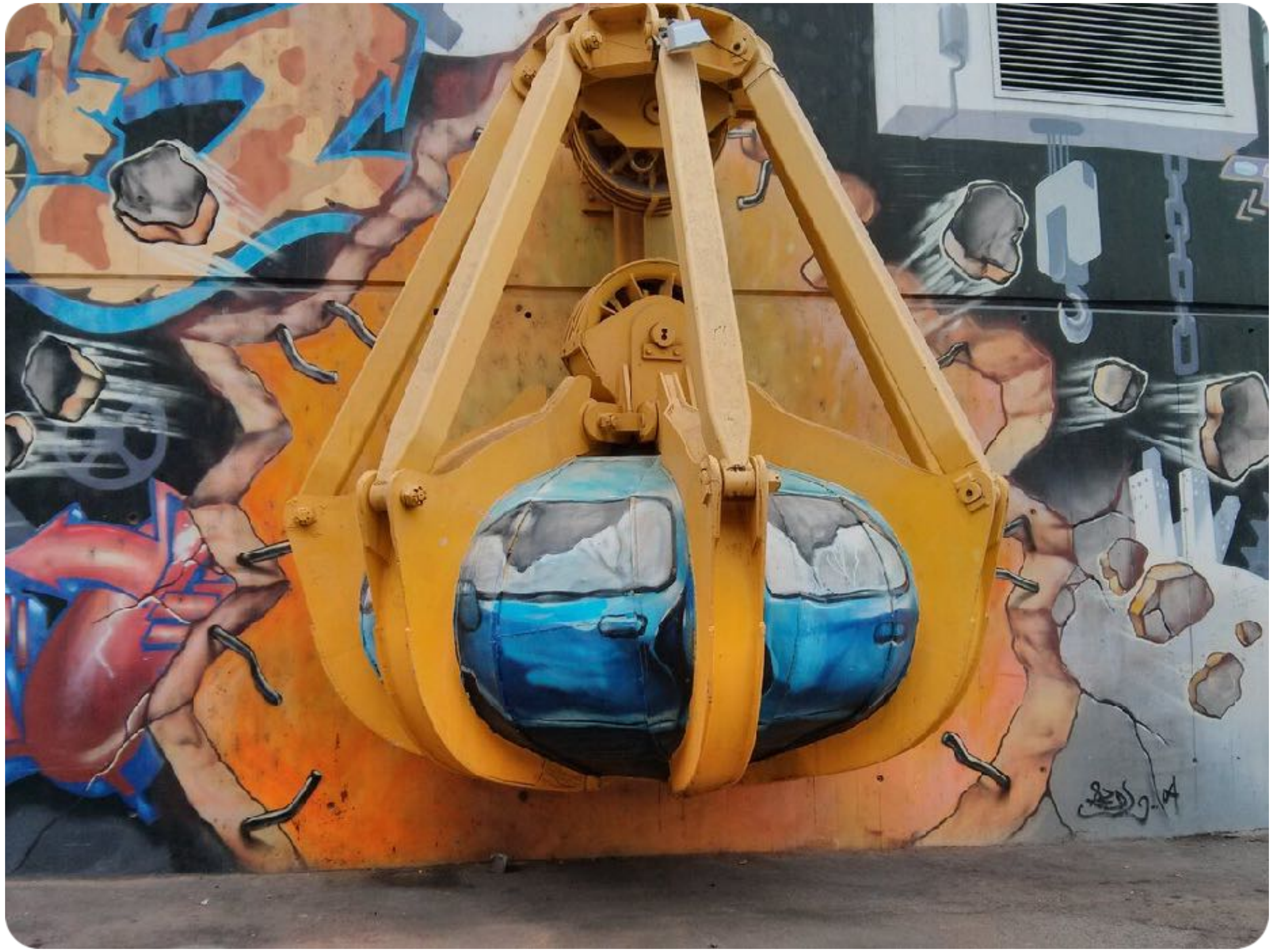
Output: dense 3D point cloud or (textured) 3D mesh

# Multi-View Stereo In a Nutshell



- Use known epipolar relation to find dense matches between images
- Create dense point cloud

# Multi-View Stereo In a Nutshell



- Use known epipolar relation to find dense matches between images
- Create dense point cloud

# Multi-View Stereo In a Nutshell

- Use known epipolar relation to find dense matches between images
- Create dense point cloud

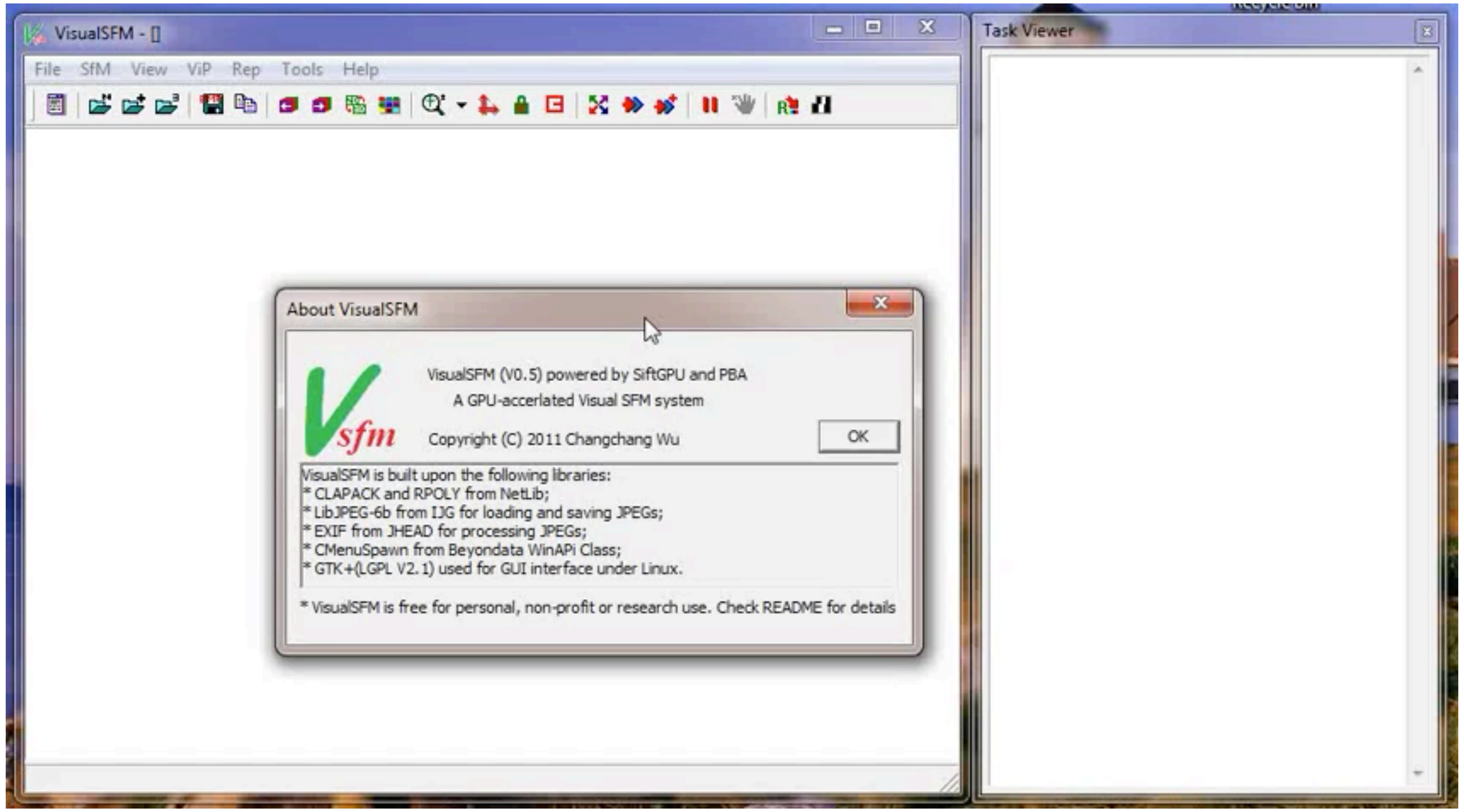
# Multi-View Stereo In a Nutshell

- Use known epipolar relation to find dense matches between images
- Create dense point cloud

# 3D Reconstruction Packages

- Bundler (https://github.com/snavely/bundler_sfm)
- Linux (Windows also supported), open source
- SfM pipeline, MVS pipelines can read file format
- Showed nice results on internet photo collections
- Not state-of-the-art anymore

# 3D Reconstruction Packages



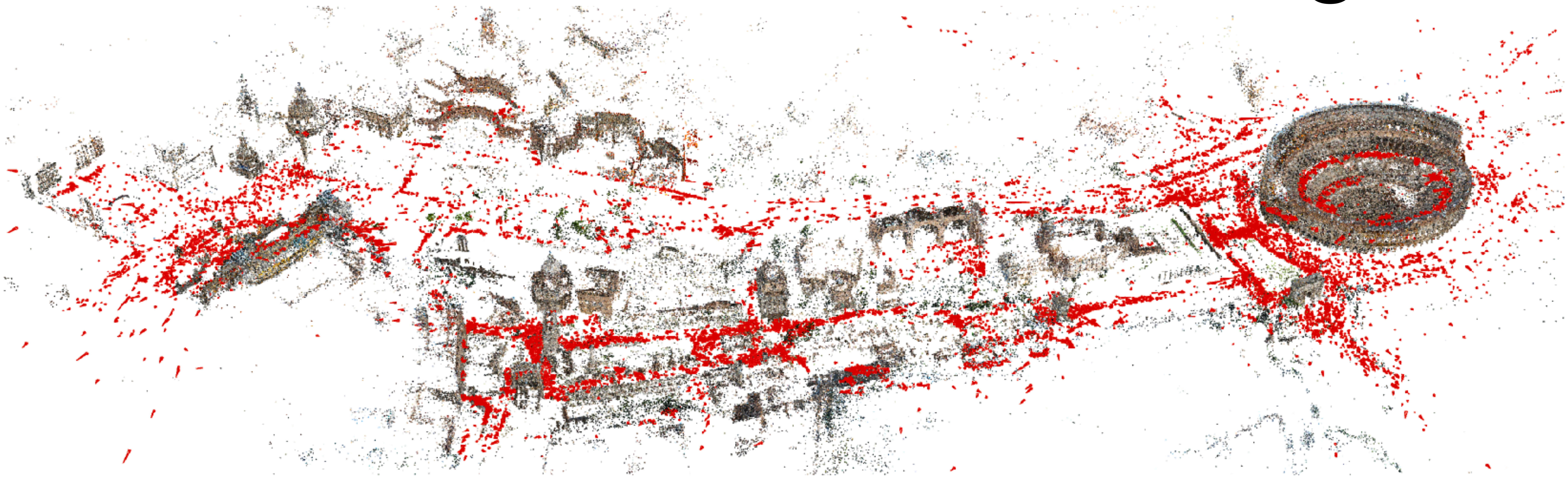

https://www.youtube.com/watch?v=5ceiOd8Yx3g

- VisualSFM (http://ccwu.me/vsfm/)
- Linux, Mac OS X, Windows, closed source
- SfM pipeline, interface to external MVS software
- Graphical User Interface
- Very efficient due to use of GPU

# 3D Reconstruction Packages

- OpenMVG (https://github.com/openMVG/openMVG)
- Linux, Mac OS X, Windows, open source
- SfM pipeline, MVS pipelines can read file format
- Very modular, functionality not in other packages (full multi-camera support)
- Not very efficient, no GUI

# 3D Reconstruction Packages



- COLMAP (https://colmap.github.io/index.html)
- Linux, Mac OS X, Windows, open source
- SfM and MVS (NVidia GPU required for MVS)
- Efficient pipeline, GUI
- High code quality, very great tool for research!

# 3D Reconstruction Packages


- COLMAP (https://colmap.github.io/index.html)
- Linux, Mac OS X, Windows, open source
- SfM and MVS (NVidia GPU required for MVS)
- Efficient pipeline, GUI
- High code quality, very great tool for research!

# 3D Reconstruction Packages

- AliceVision Meshroom (https://alicevision.org/)
- Linux, Windows, open source
- SfM and MVS (NVidia GPU required for both)
- Includes work by Tomas Pajdla and his PhD students
- Have not tried it yet, on my Todo list

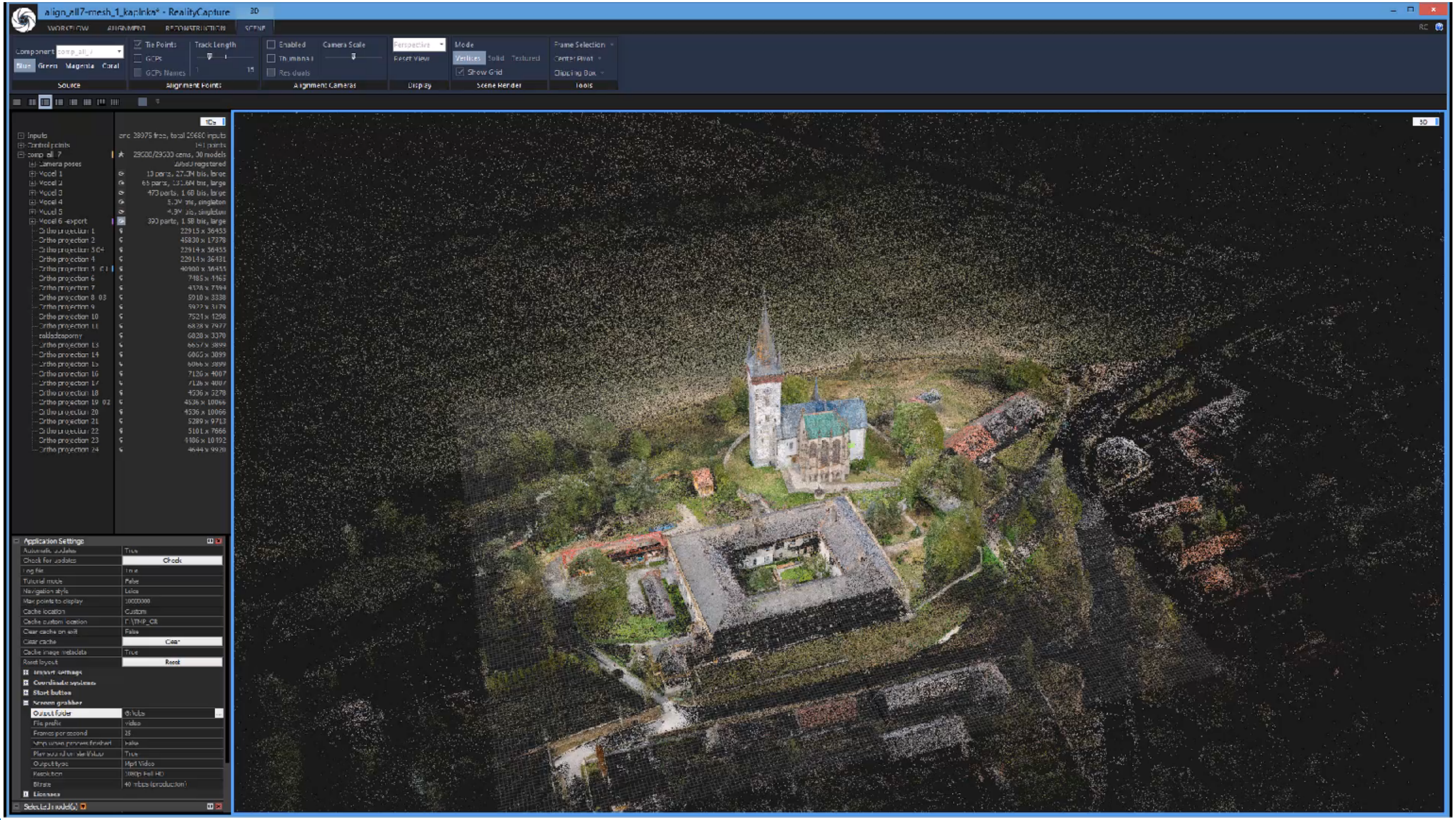
# 3D Reconstruction Packages

- RealityCapture (https://www.capturingreality.com/)
- Commercial software, Windows only
- Start-up (CapturingReality) out of Slovakia, former PhD students at CVUT, recently acquired by Epic Games
- Both SfM and MVS (MVS requires NVidia GPU)
- Highly efficient, SfM takes a few minutes even for large scenes
- Very high quality (probably best software out there)

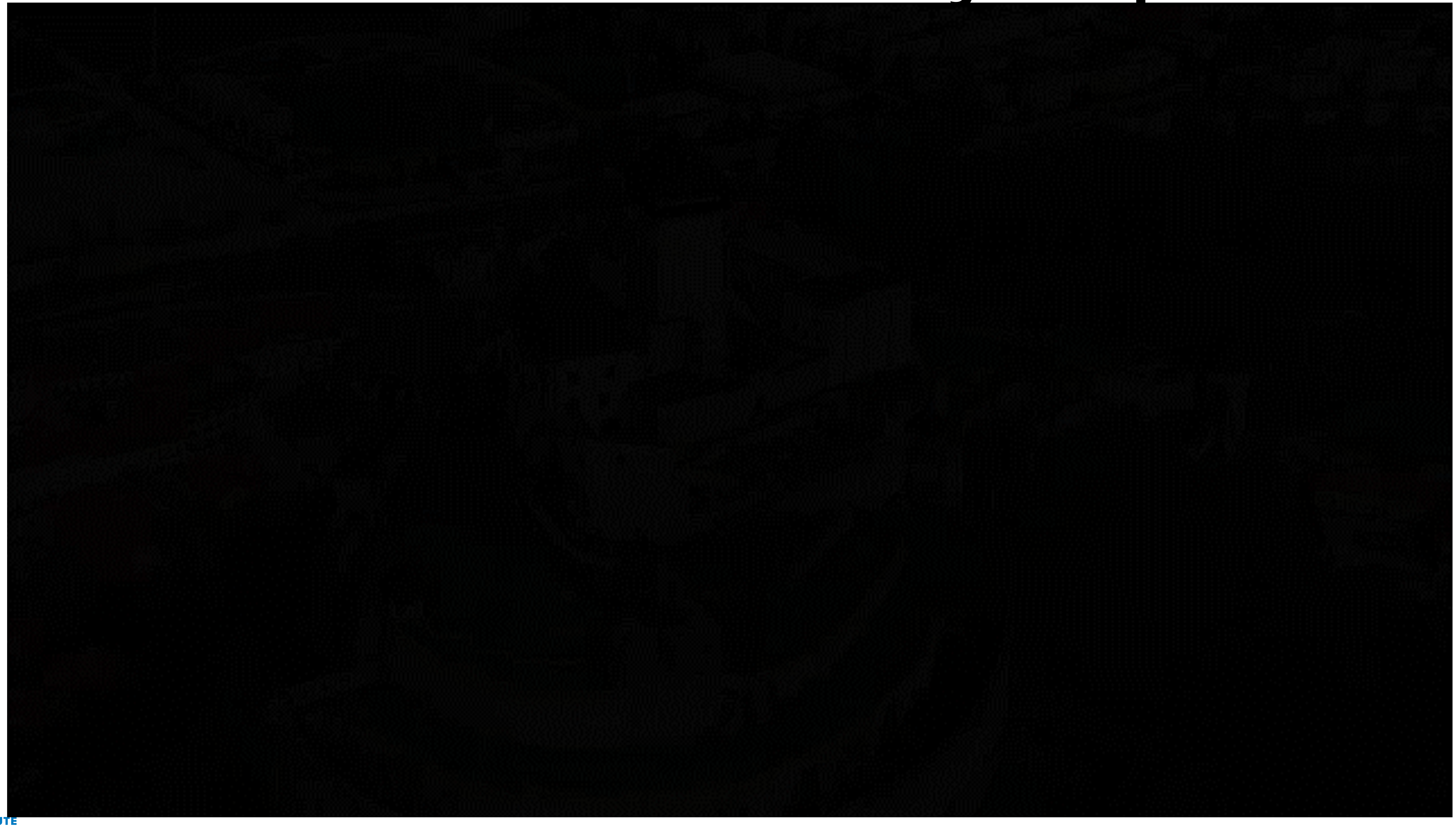
# 3D Reconstruction Packages

- RealityCapture (https://w
- Commercial software, V
- Start-up (CapturingRea students at CVUT, recently acquired by Epic
- Both SfM and MVS (MVS )
- Highly efficient, SfM tak a few minutes even for large scenes
- Very high quality (probably best software out there)

# Results with RealityCapture



# Results with RealityCapture



# 3D Reconstruction Packages

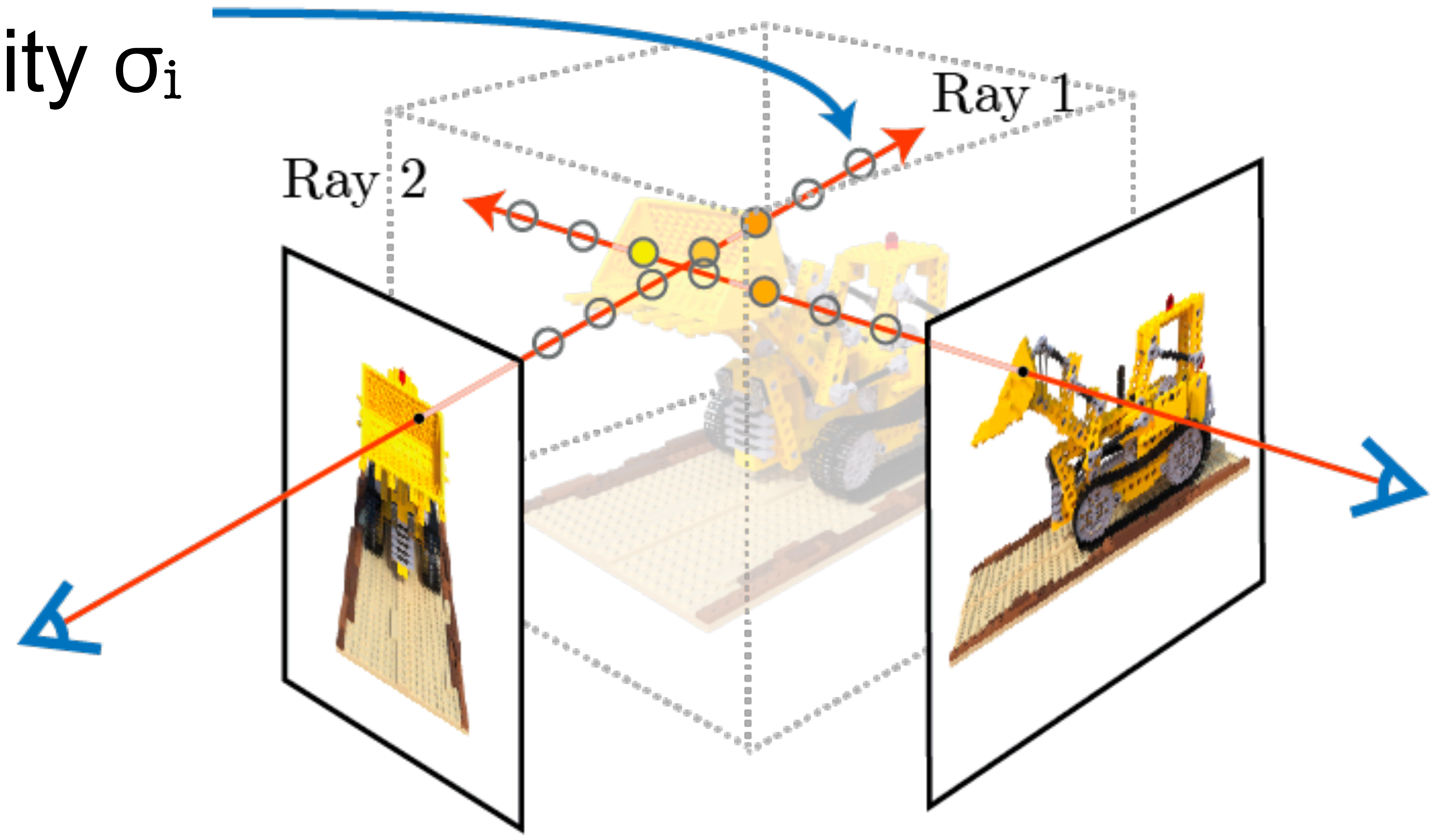
- Many more commercial packages available
  - Agisoft Metashape (https://www.agisoft.com/)
  - Pix4D (https://www.pix4d.com/)
  - …

# Bonus: Neural Radiance Fields (NeRFs)

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

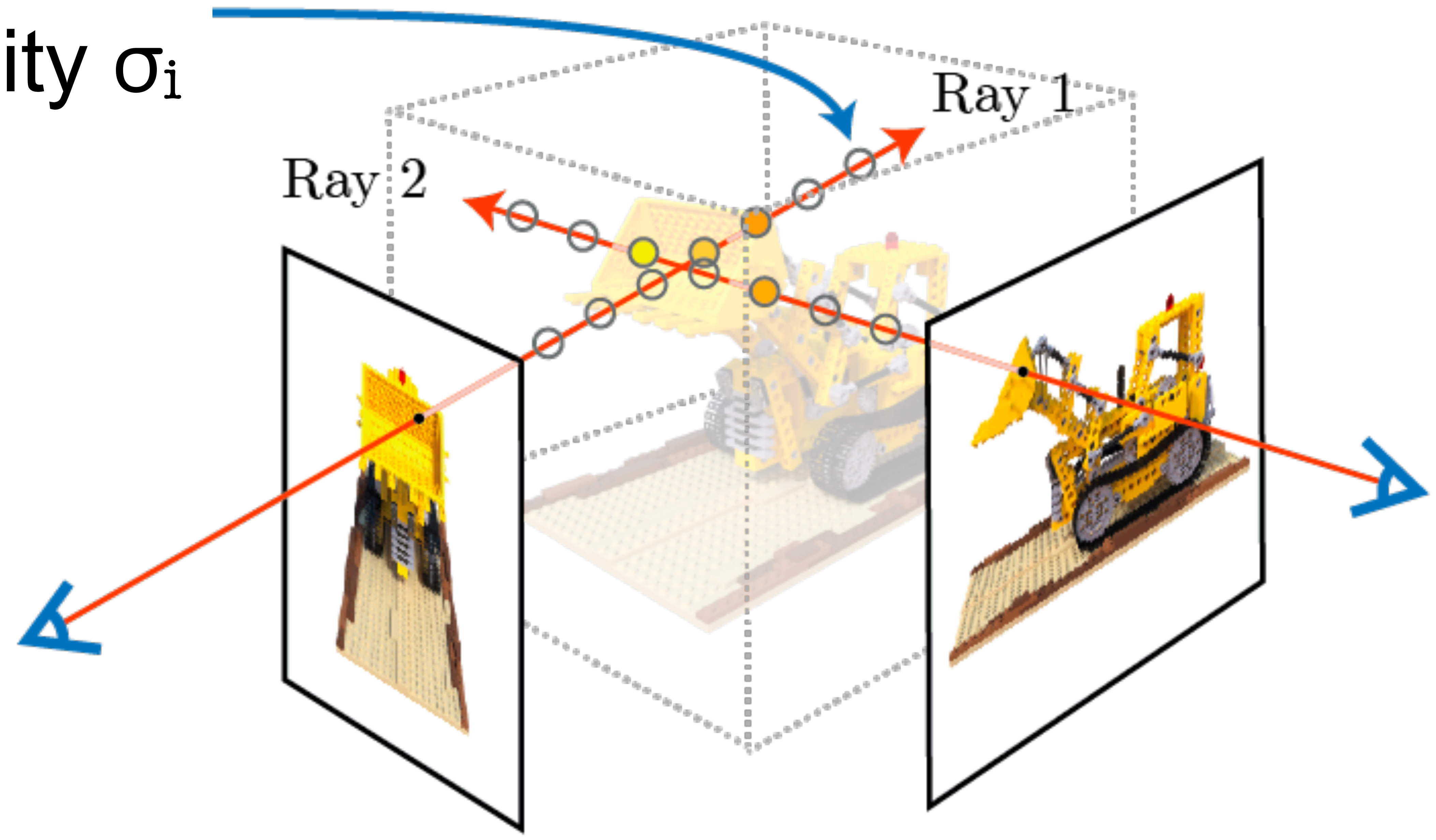
color $\mathbf{c}_i$,
volume density $\sigma_i$




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$

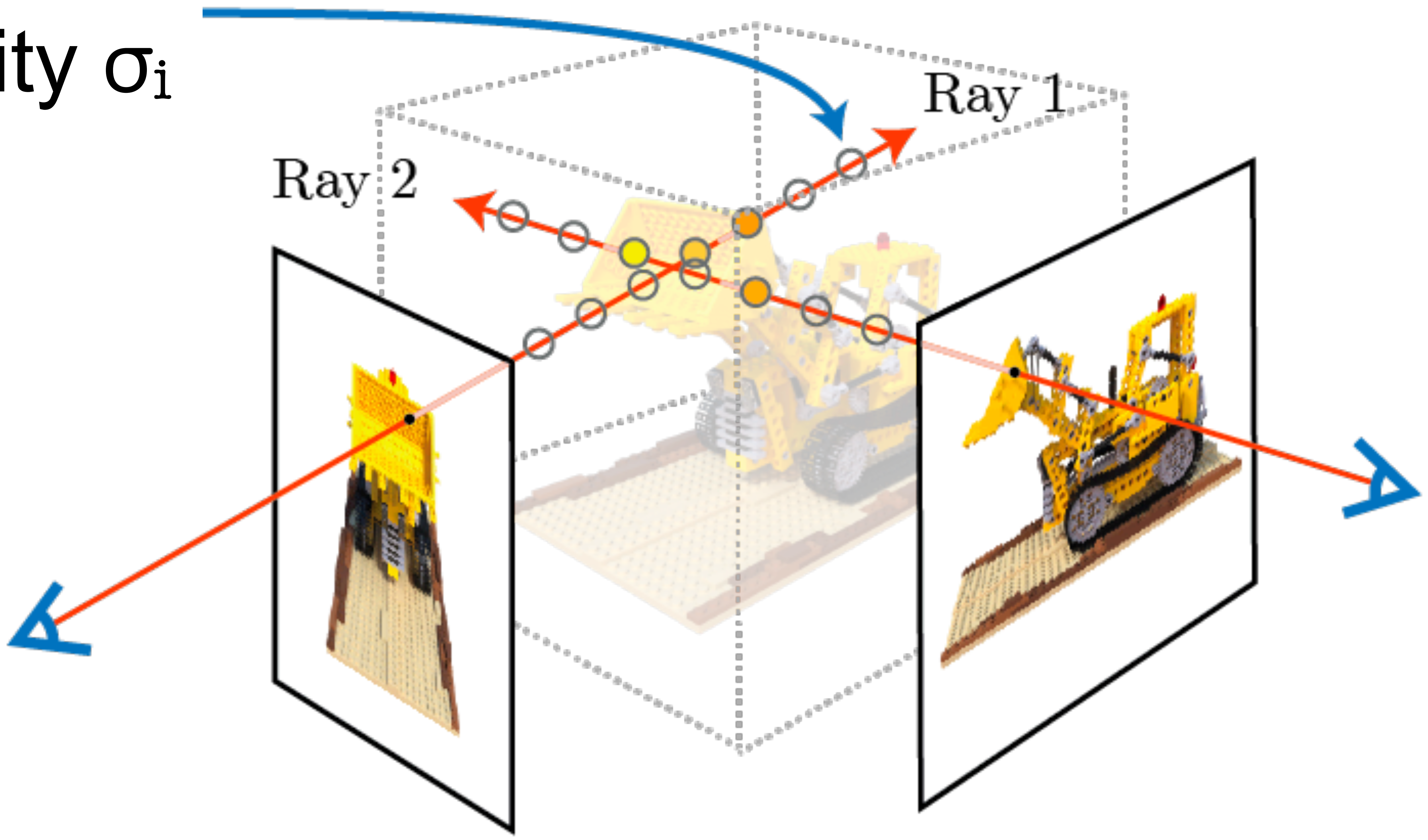



$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$

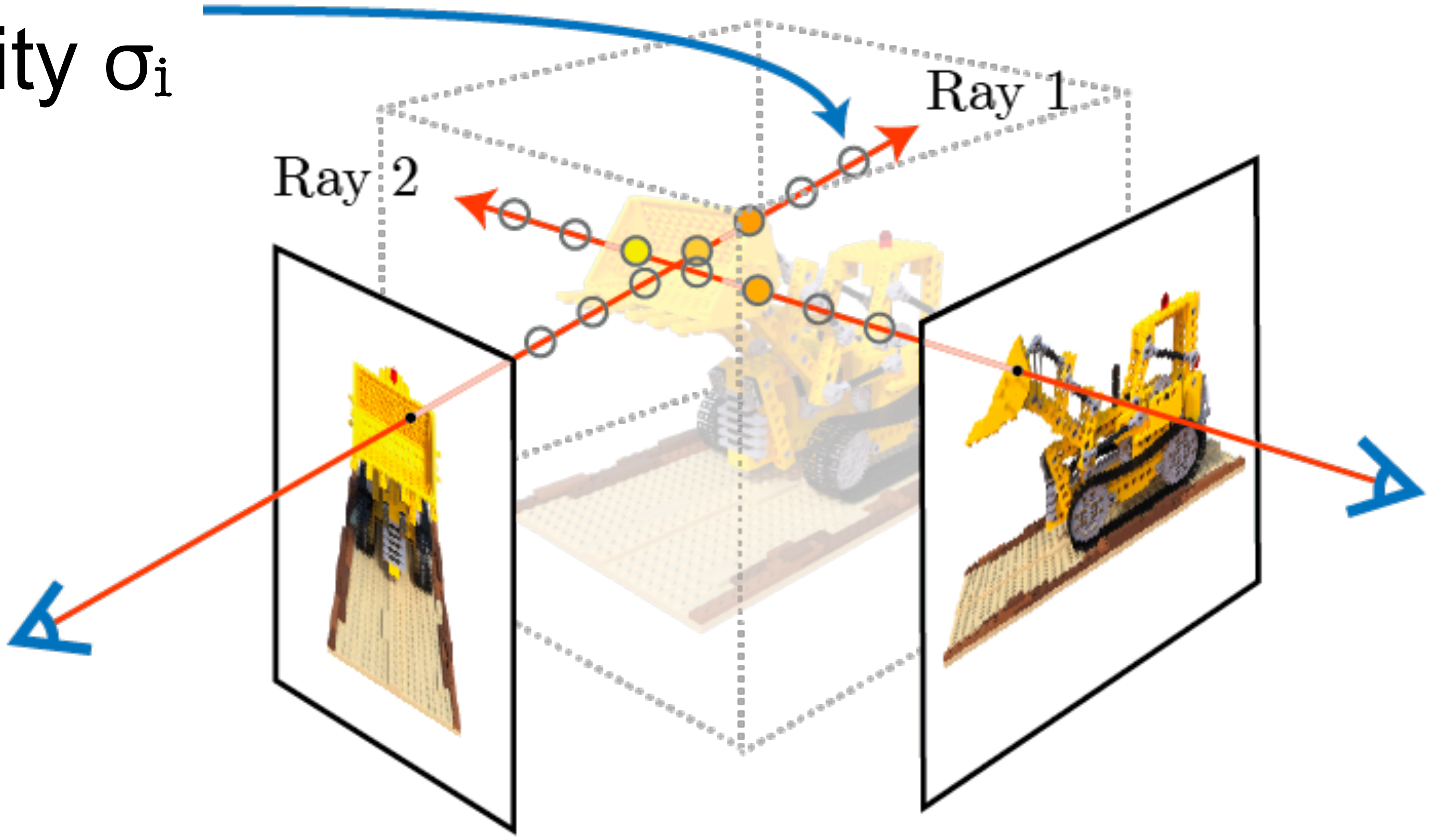



$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

final color

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering




$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

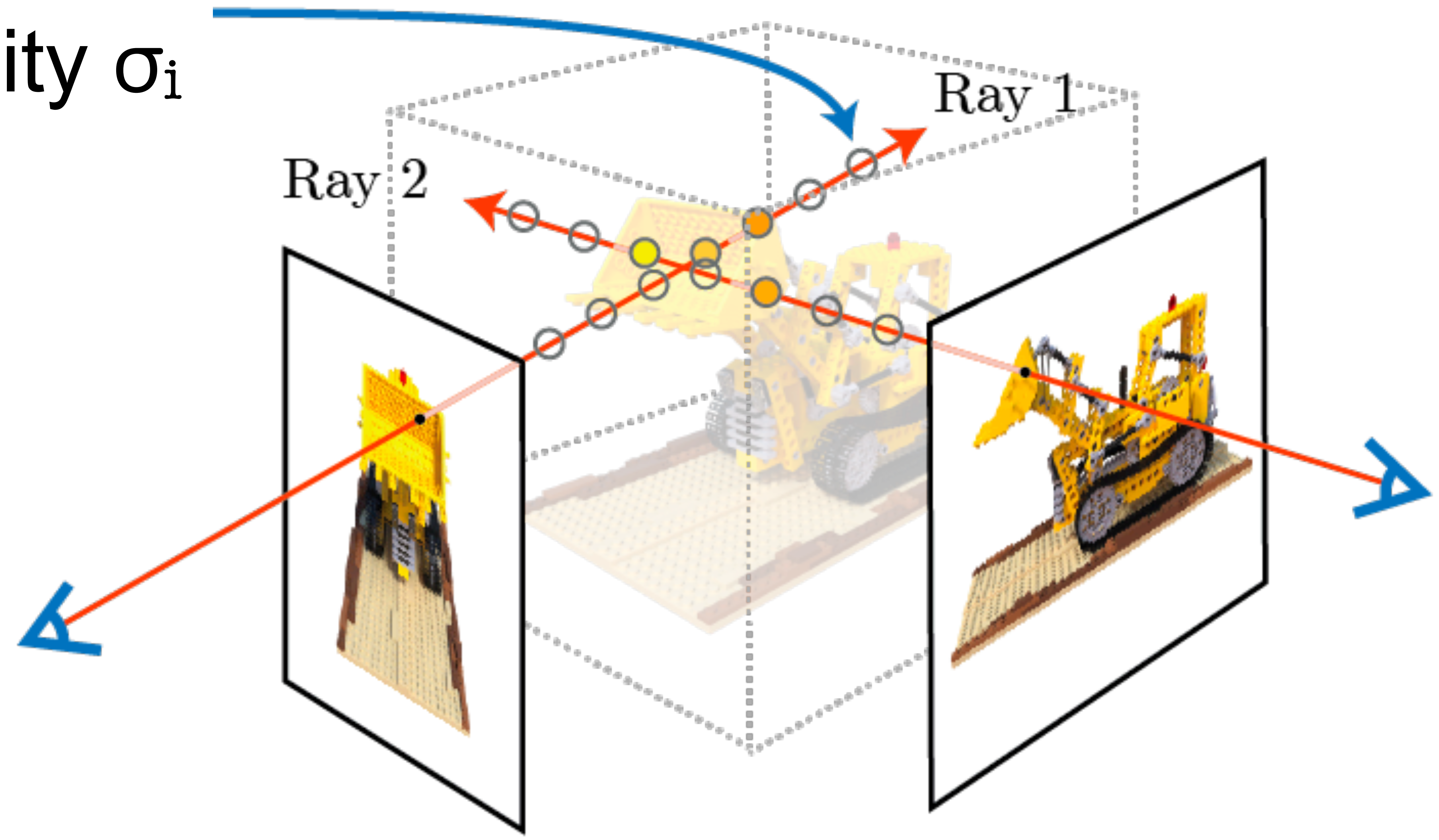
final color

color at i-th sample

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$



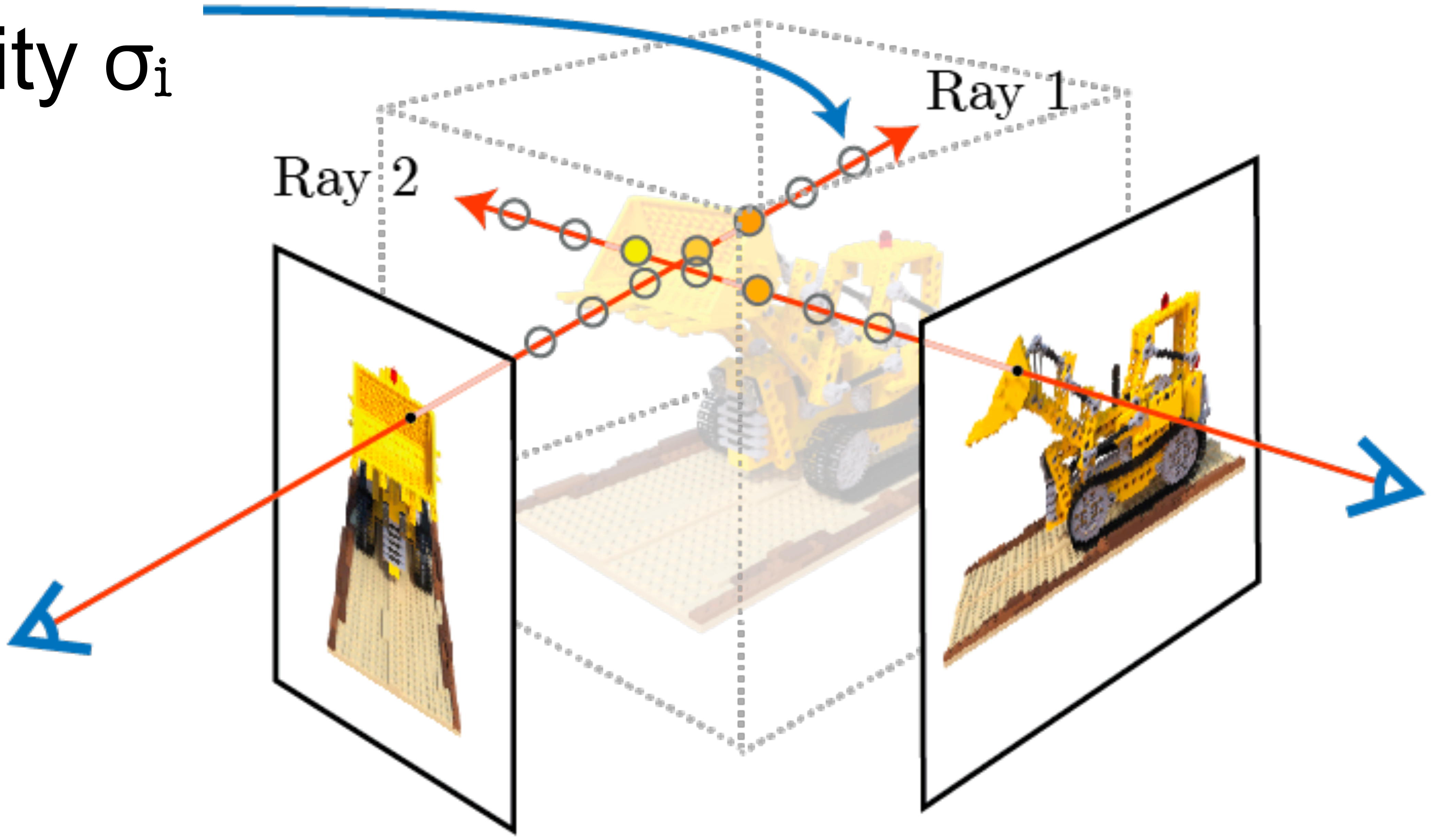

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

final color — visibility — color at i-th sample

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$




$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} \underbrace{T_i(1 - \exp(-\sigma_i \delta_i))}_{\text{visibility}} \mathbf{c}_i$$

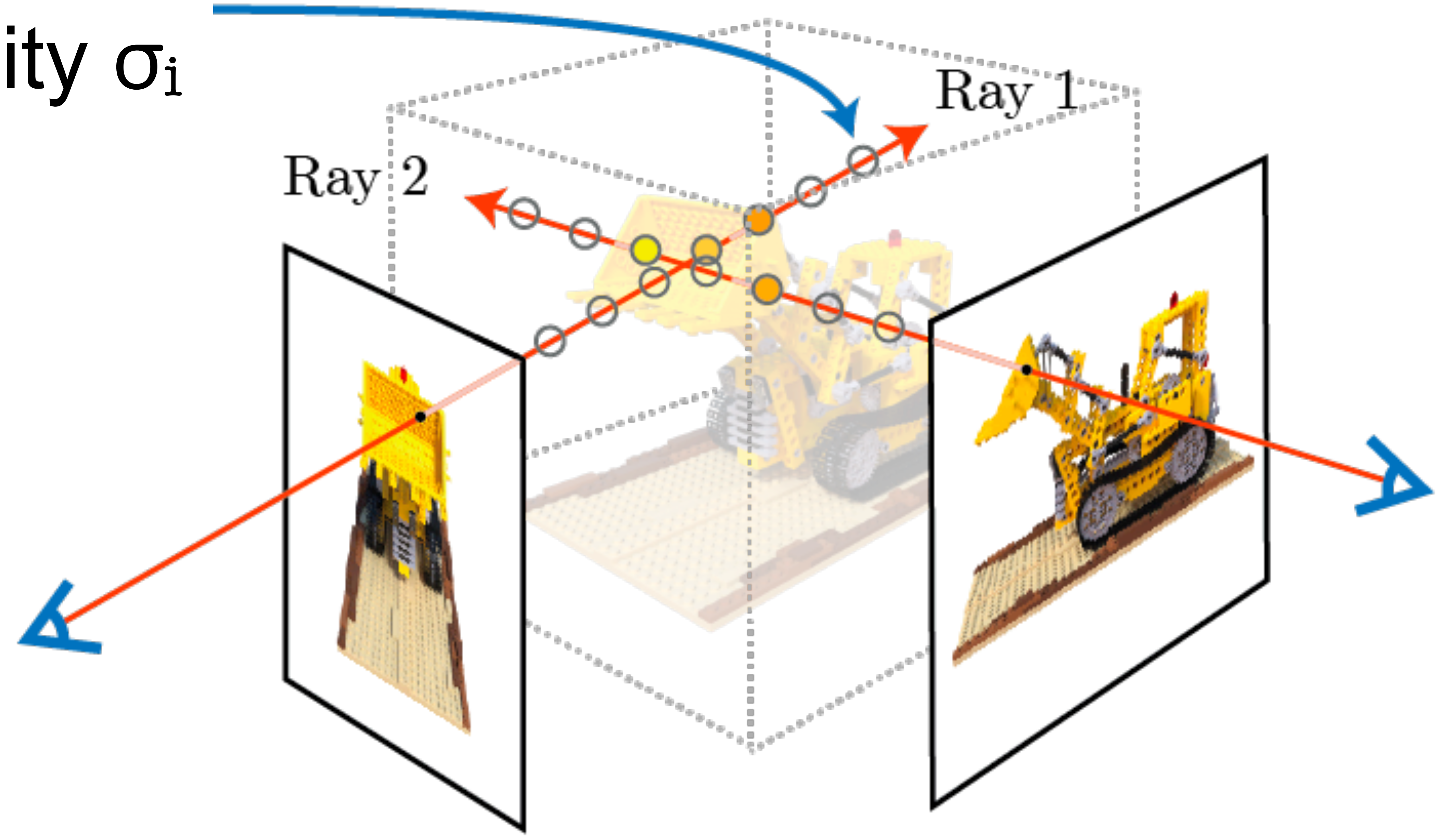
final color — visibility — color at i-th sample

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$




$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

final color — visibility — color at i-th sample

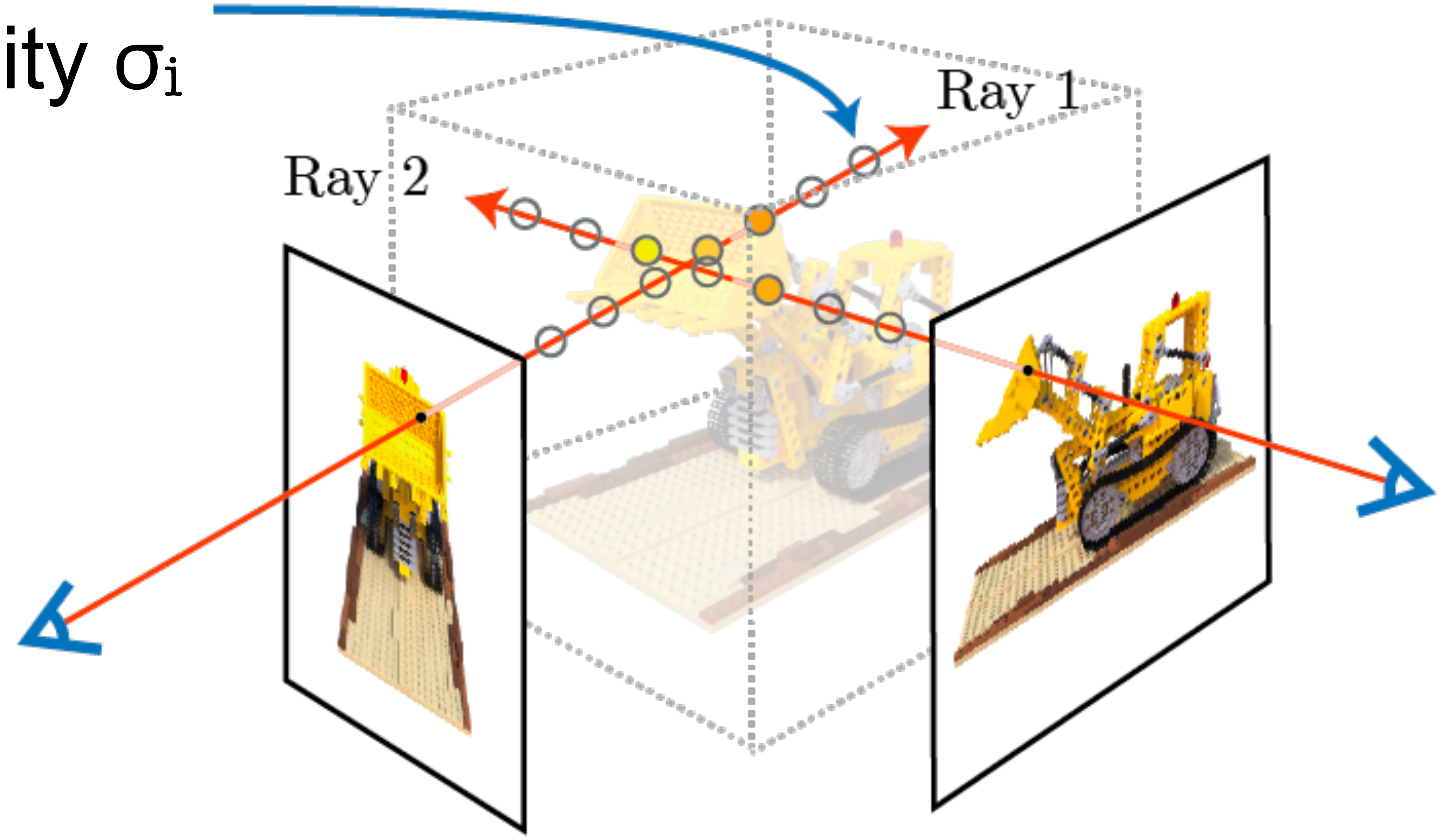
distance to previous sample

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$




distance to previous sample

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i (1 - \exp(-\sigma_i \delta_i)) \mathbf{c}_i$$

final color — visibility — color at i-th sample

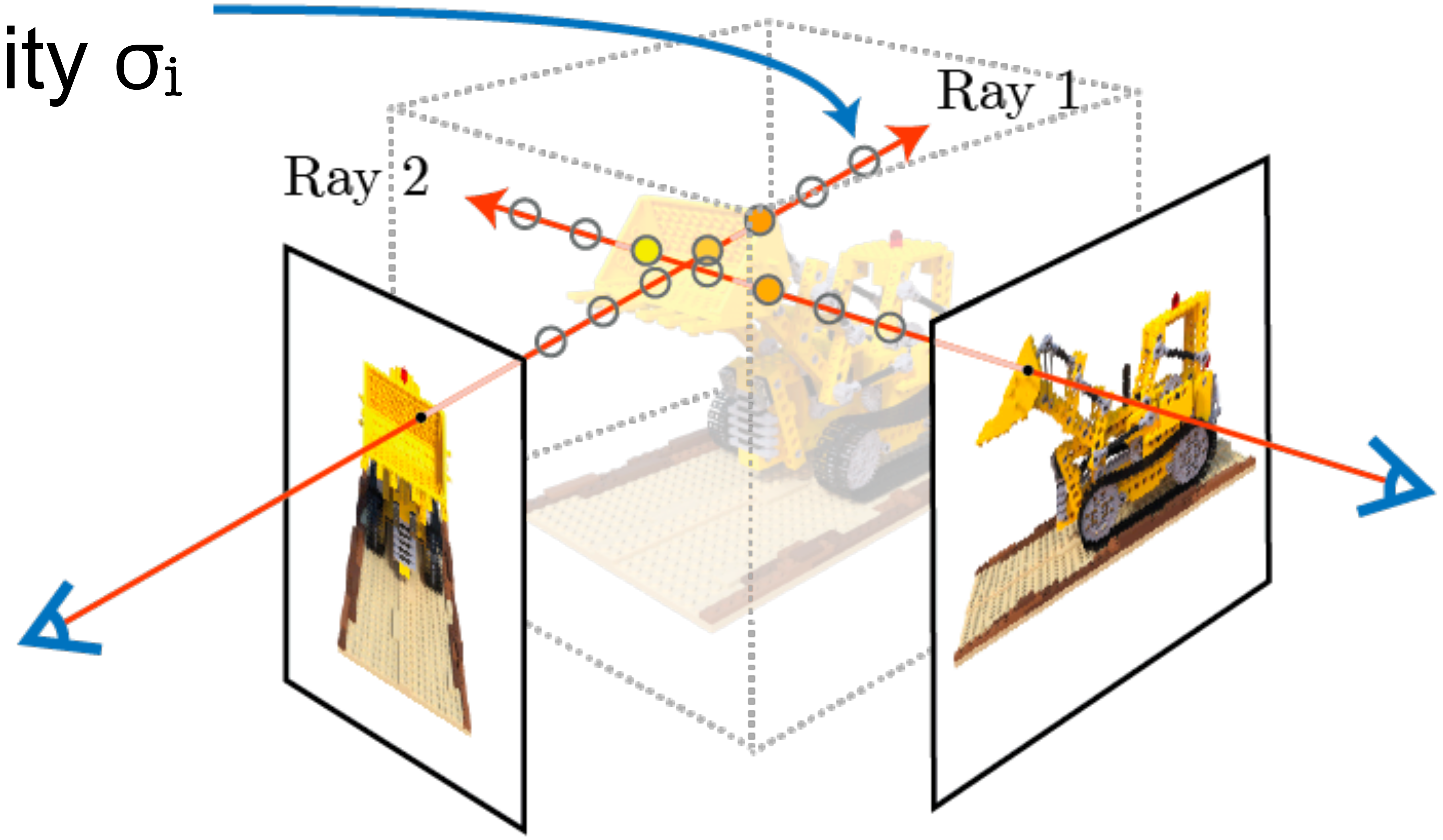
$$T_i = \exp\left( - \sum_{j=1}^{i-1} \sigma_j \delta_j \right)$$

small if high density before this sample

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Rendering

color $\mathbf{c}_i$,
volume density $\sigma_i$




Fully differentiable!

distance to previous sample

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i$$

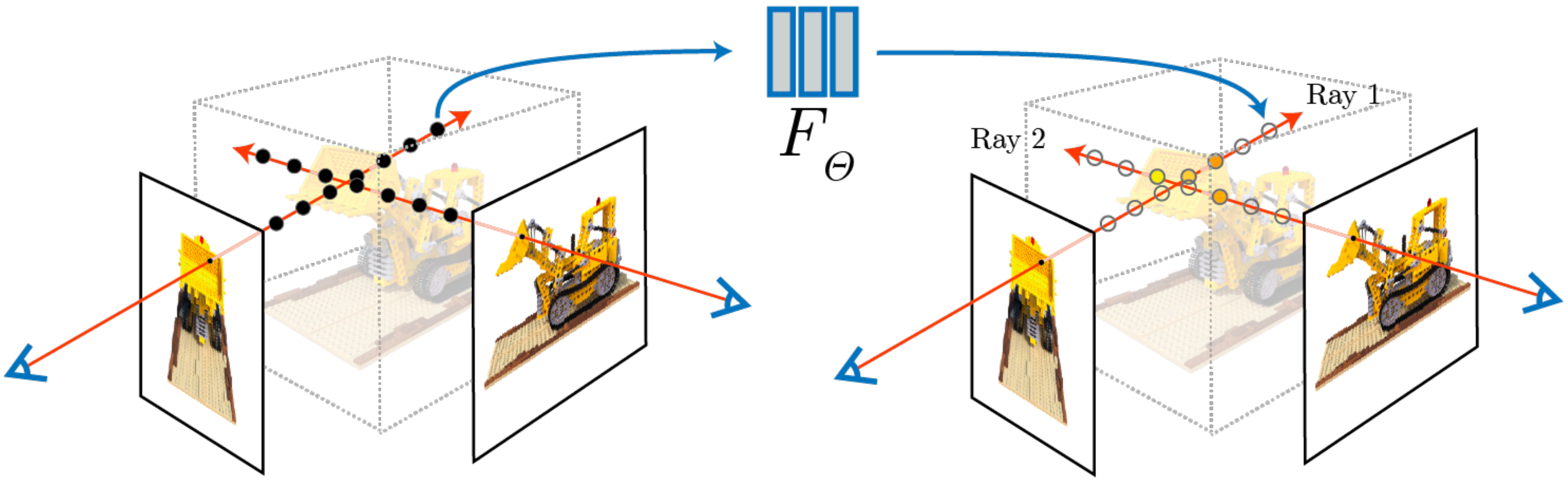
final color — visibility — color at i-th sample

$$T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

small if high density before this sample

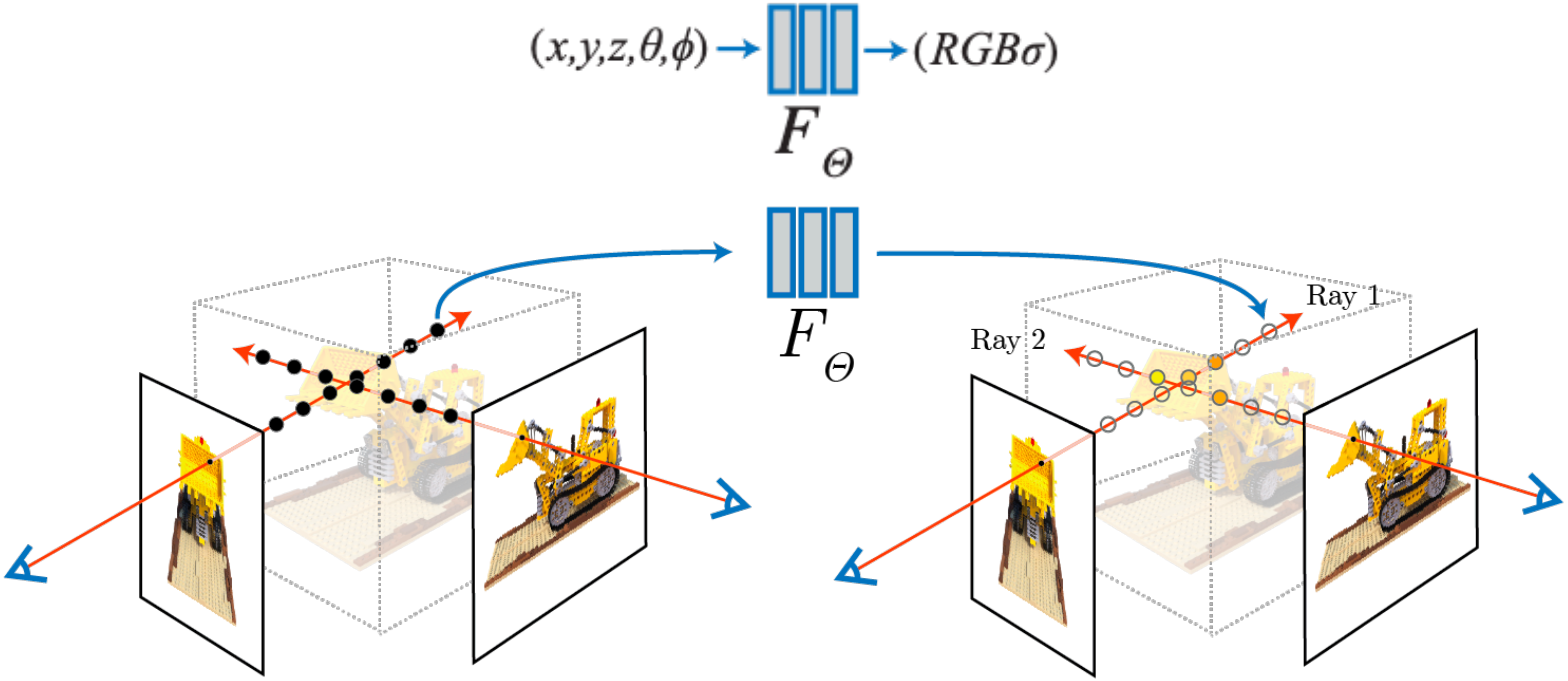
[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Neural Radiance Fields (NeRFs)




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

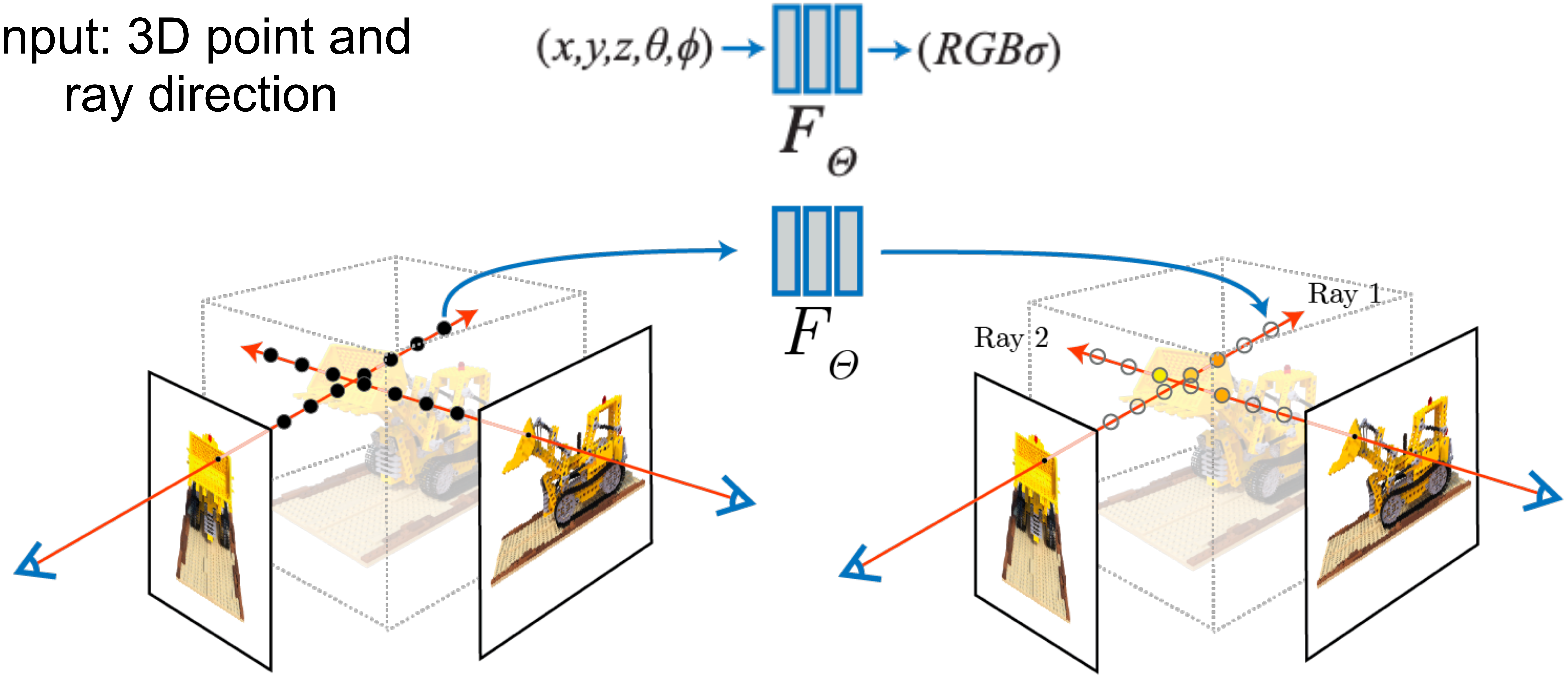
# Neural Radiance Fields (NeRFs)




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Neural Radiance Fields (NeRFs)

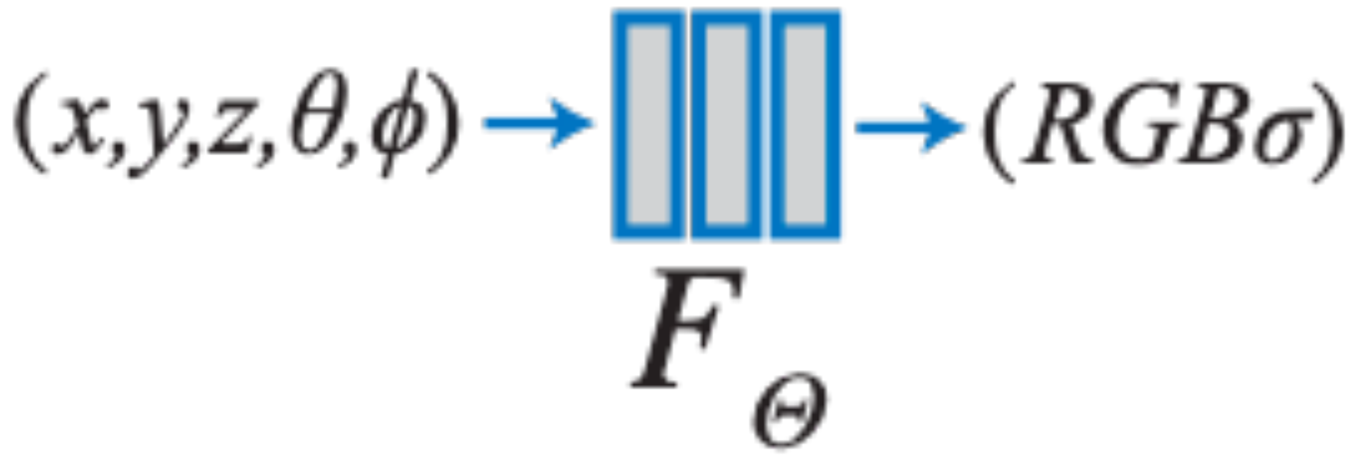
input: 3D point and ray direction



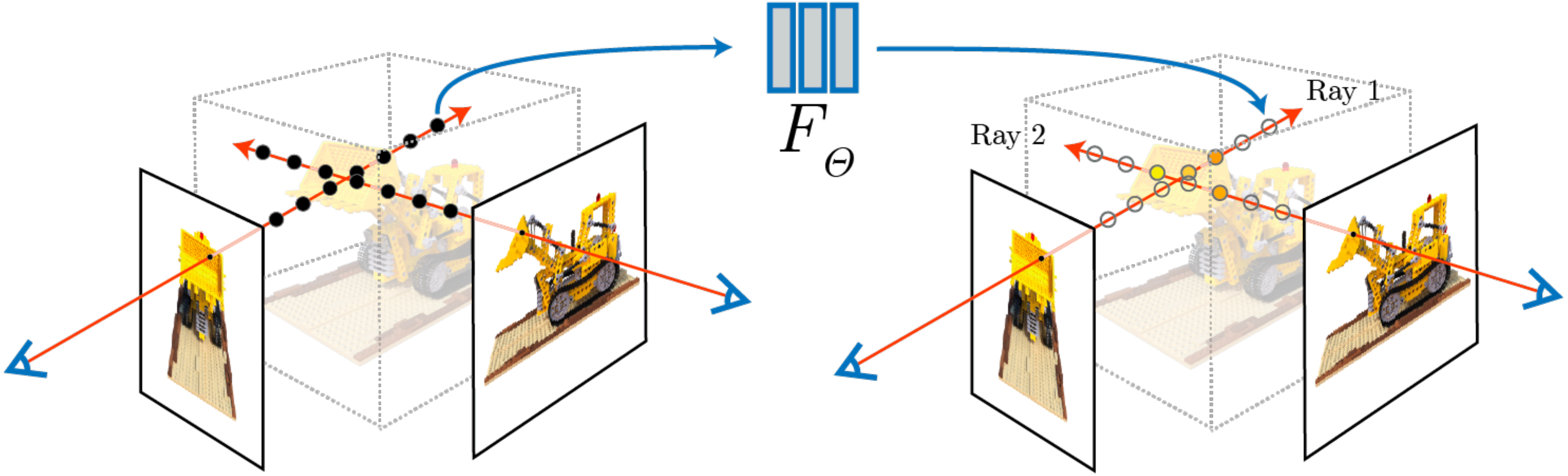

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Neural Radiance Fields (NeRFs)

input: 3D point and ray direction



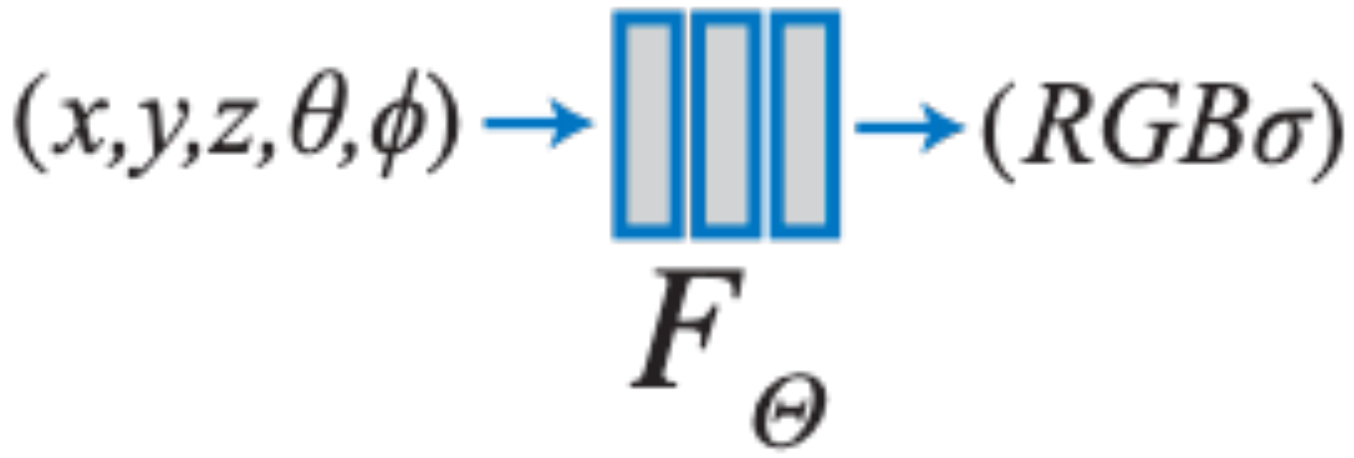

output: color and density



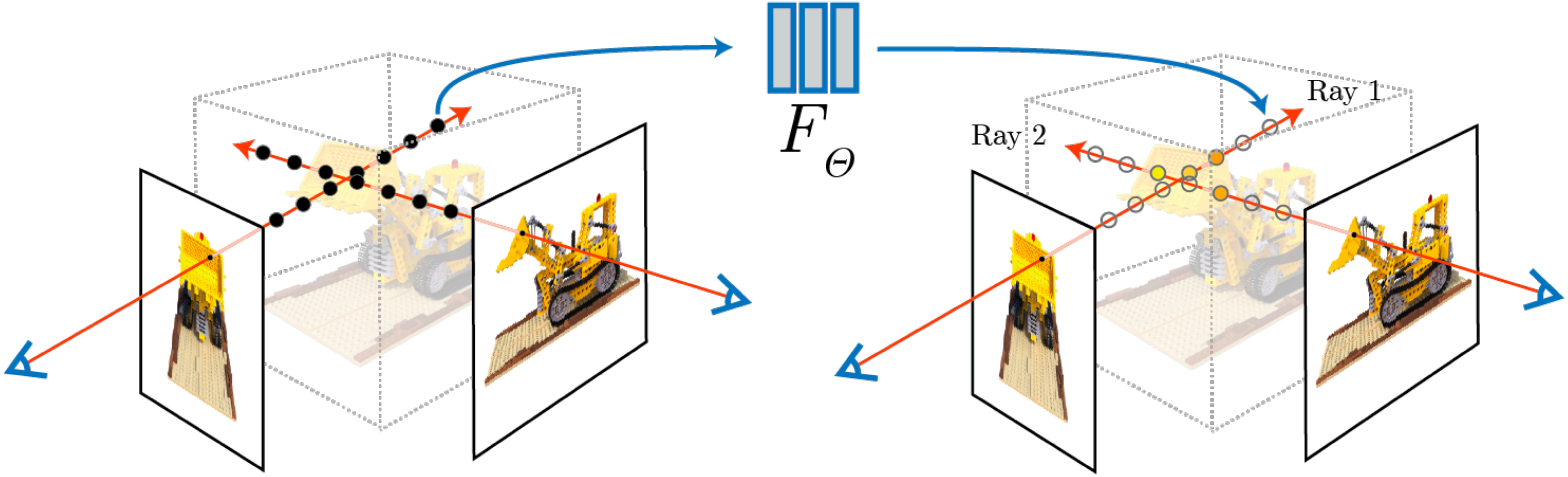

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Neural Radiance Fields (NeRFs)

input: 3D point and ray direction



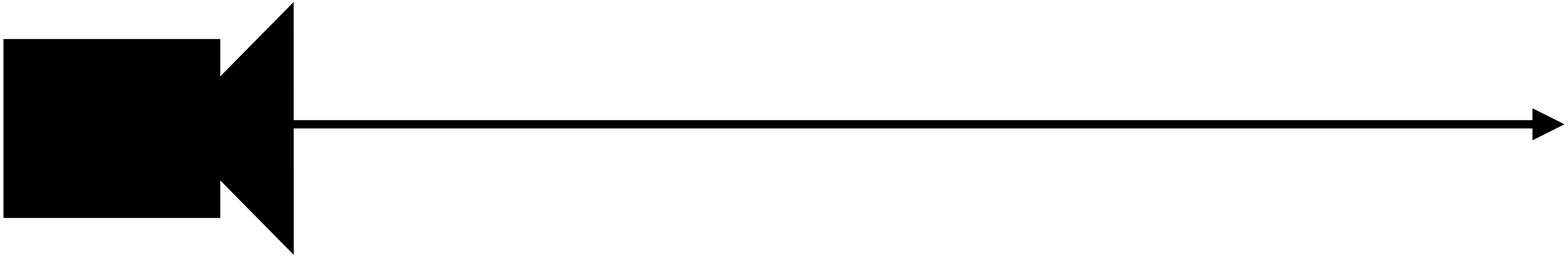

output: color and density




**Continuous** scene representation (vs. discrete voxel volumes)

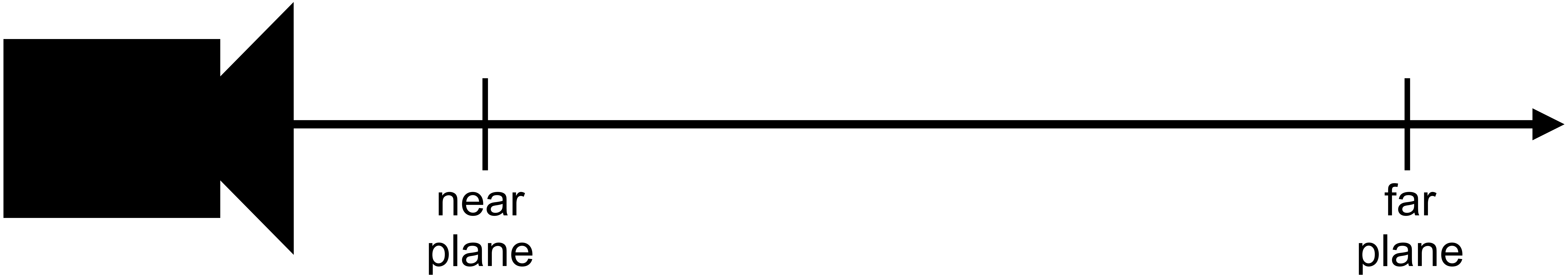
[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Sampling



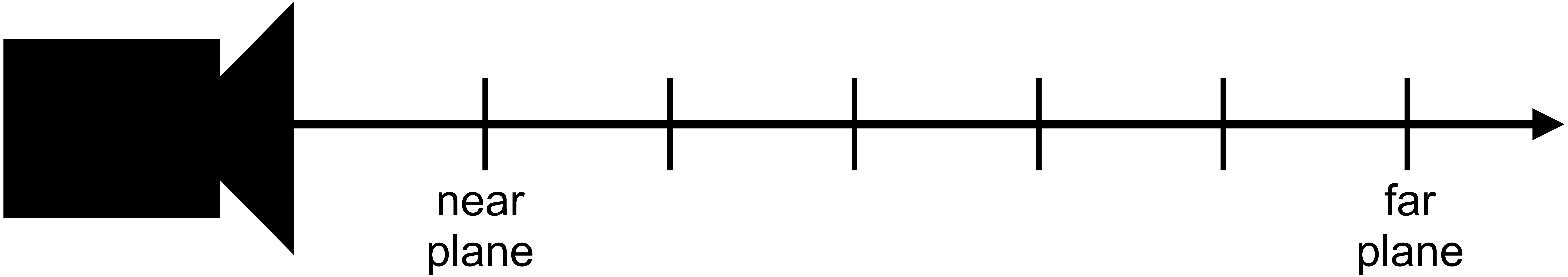
[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Sampling




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Sampling

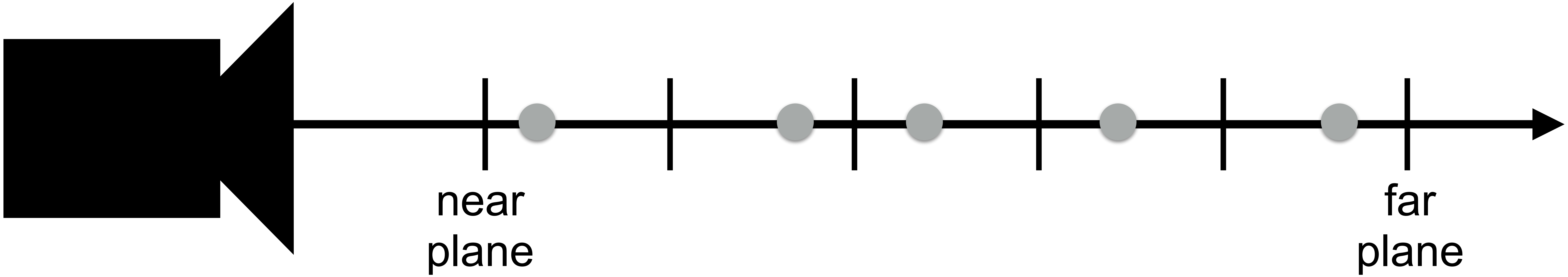



subdivide into equally sized intervals

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Volume Sampling

uniform sampling inside intervals → continuous sampling of the volume

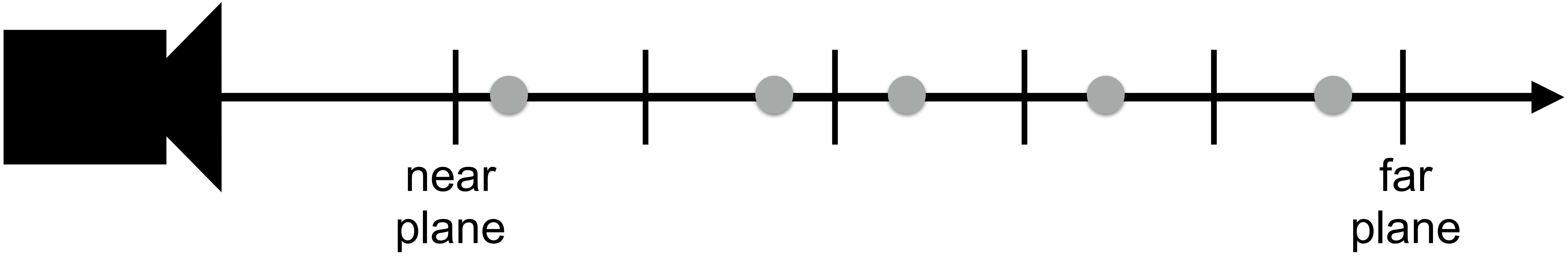



subdivide into equally sized intervals

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Hierarchical Volume Sampling

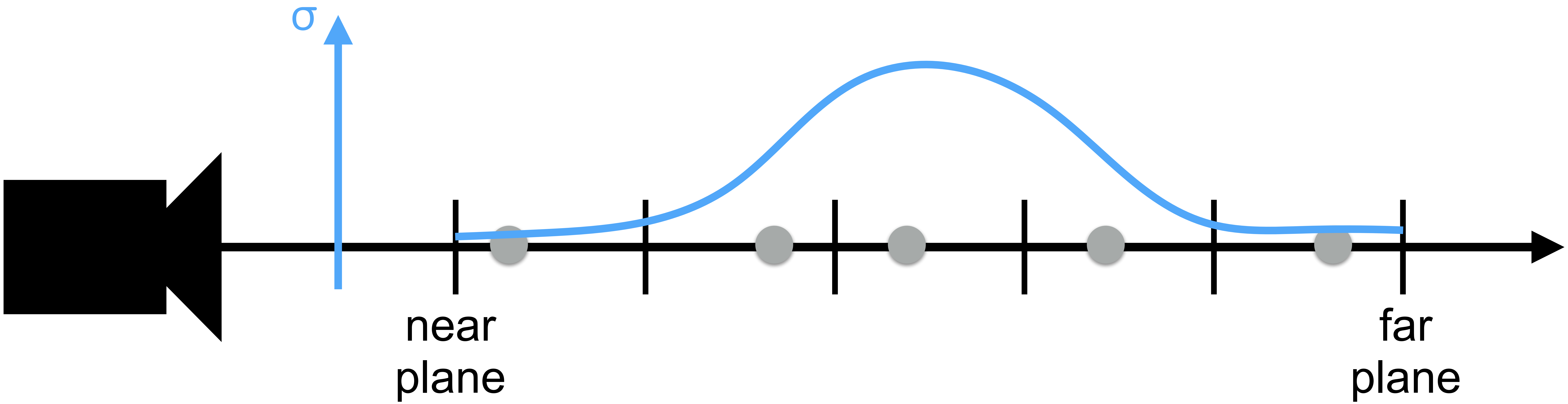
Coarse sampling (coarse network): Uniform sampling in equally-spaced intervals




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Hierarchical Volume Sampling

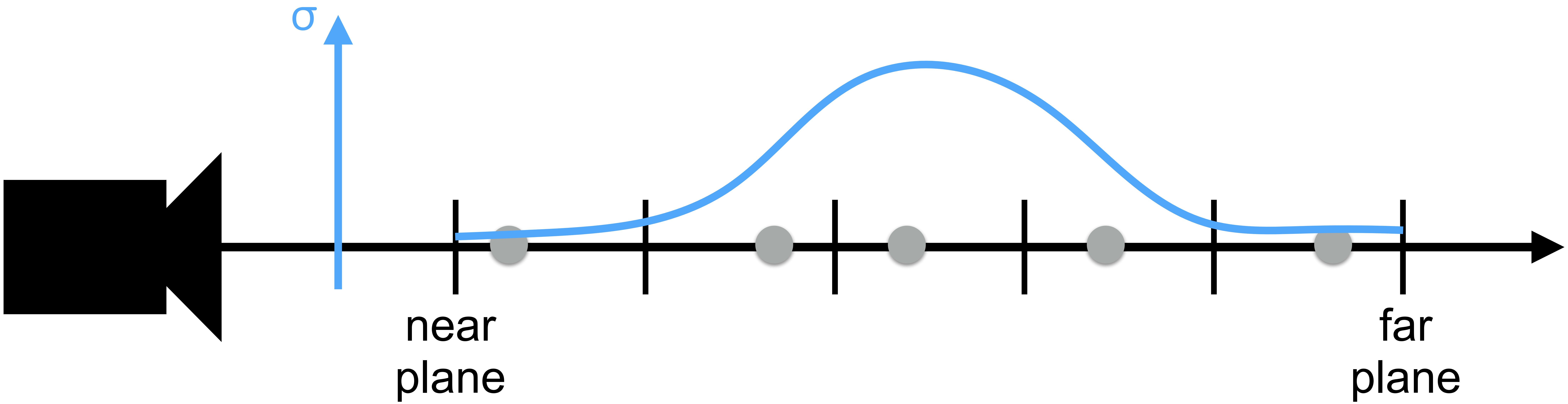
Coarse sampling (coarse network): Uniform sampling in equally-spaced intervals




[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Hierarchical Volume Sampling

Coarse sampling (coarse network): Uniform sampling in equally-spaced intervals

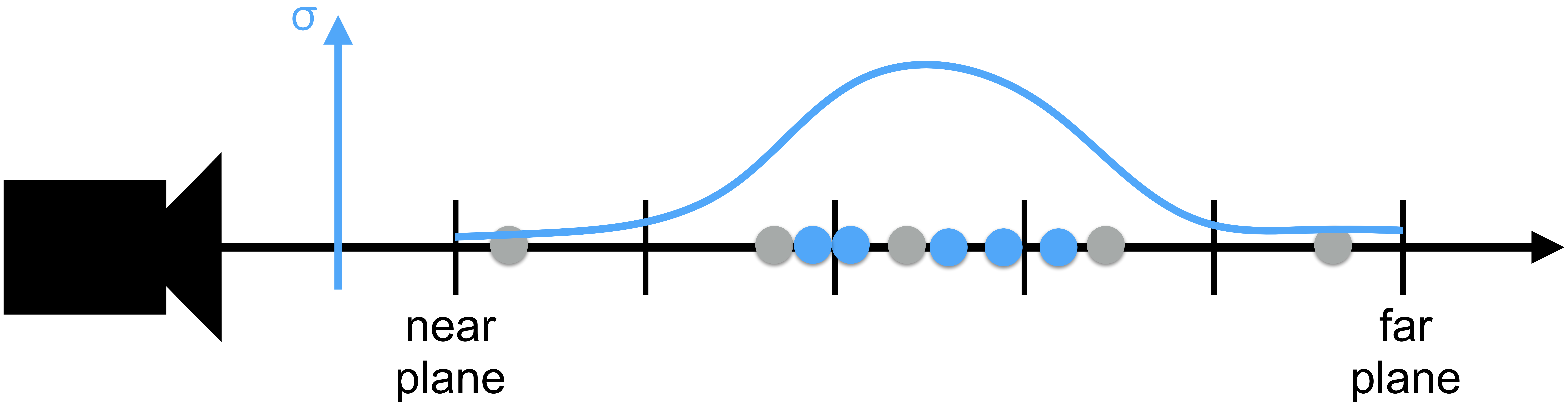



“Fine” sampling (“fine” network): Sample according to observed densities

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Hierarchical Volume Sampling

Coarse sampling (coarse network): Uniform sampling in equally-spaced intervals

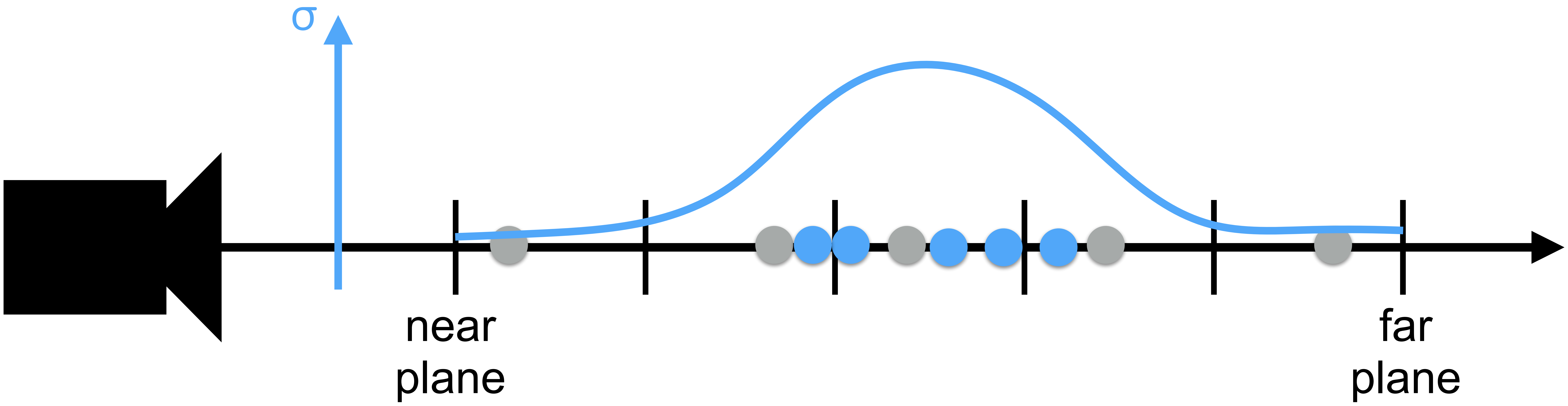



“Fine” sampling (“fine” network): Sample according to observed densities

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Hierarchical Volume Sampling

Coarse sampling (coarse network): Uniform sampling in equally-spaced intervals




“Fine” sampling (“fine” network): Sample according to observed densities

All samples are used during volume rendering

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Training

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

[Müller, Evans, Schied, Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, SIGGRAPH 2022]

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Training

ground truth color

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

[Müller, Evans, Schied, Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, SIGGRAPH 2022]

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Training

ground truth color

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

color predicted by “fine” network

[Müller, Evans, Schied, Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, SIGGRAPH 2022]

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Training

ground truth color

$$\mathcal{L} = \sum_{\mathbf{r} \in \mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

color predicted by coarse network

color predicted by "fine" network

[Müller, Evans, Schied, Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, SIGGRAPH 2022]

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Training

ground truth color

$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[ \left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

color predicted by coarse network

color predicted by “fine” network

Trained individually per scene, can now be done in a matter of minutes

[Müller, Evans, Schied, Keller, Instant Neural Graphics Primitives with a Multiresolution Hash Encoding, SIGGRAPH 2022]

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]

# Neural Radiance Fields (NeRFs)

Synthetic Scenes

[Mildenhall, Srinivasan, Tancik, et al., NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis, ECCV 2020]