CSE 252D: Advanced Computer Vision

Manmohan Chandraker

Lecture 8: Advanced SFM (End of Module 1)

Virtual classrooms

- Virtual lectures on Zoom
  - Only host shares the screen
  - Keep video off and microphone muted
  - But please do speak up (remember to unmute!)
  - Slides uploaded on webpage just before class
- Virtual interactions on Zoom
  - Ask and answer plenty of questions
  - “Raise hand” feature on Zoom when you wish to speak
  - Post questions on chat window
  - Happy to try other suggestions!
- Lectures recorded and upload on Kaltura
  - Available under “My Media” on Canvas

Overall goals for the course

- Introduce the state-of-the-art in computer vision
- Study principles that make them possible
- Get understanding of tools that drive computer vision
- Enable one or all of several such outcomes
  - Pursue higher studies in computer vision
  - Join industry to do cutting-edge work in computer vision
  - Gain appreciation of modern computer vision technologies
- This is a great time to study computer vision!

Papers for Fri, Apr 23

- Building Rome in a Day
  - https://grail.cs.washington.edu/rome/rome_paper.pdf
- CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM
  - https://arxiv.org/abs/1804.00874
- Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry
  - https://arxiv.org/abs/1904.01892
- BA-Net: Dense Bundle Adjustment Network
  - https://arxiv.org/abs/1806.04807

Papers for Wed, Apr 28

- Deep Fundamental Matrix Estimation
  - https://openaccess.thecvf.com/content_ECCV_2018/html/Rene_Ranftl_Deep_Fundamental_Matrix_ECCV_2018_paper.html
- DSAC - Differentiable RANSAC for Camera Localization
  - https://arxiv.org/abs/1611.05705
- Unsupervised Monocular Depth Estimation with Left-Right Consistency
  - https://arxiv.org/abs/1609.03677
- LSD-SLAM: Large-Scale Direct Monocular SLAM
  - https://vision.in.tum.de/_media/spezial/bib/engel14eccv.pdf

Papers for Fri, Apr 30

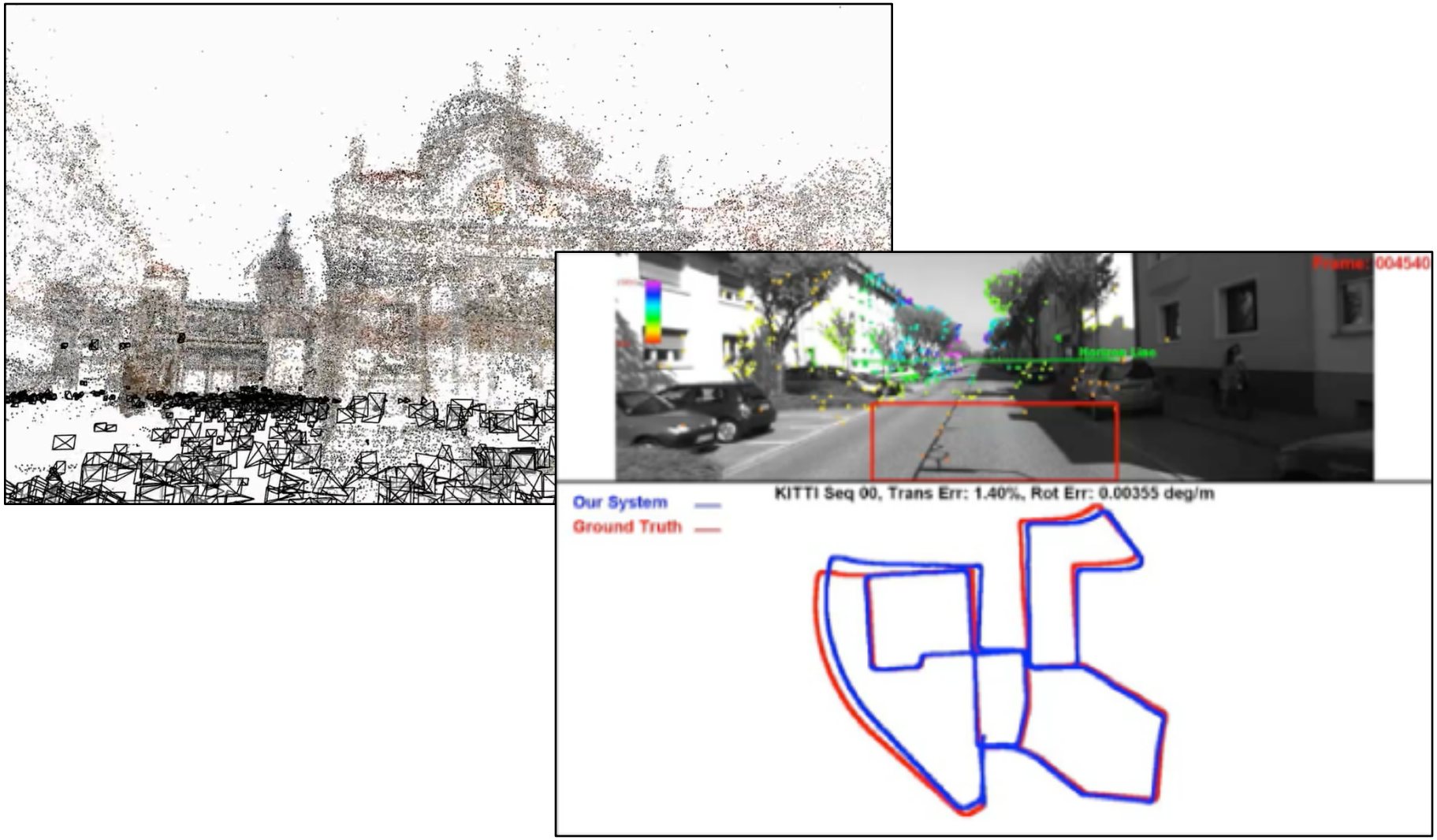
- A Discriminative Feature Learning Approach for Deep Face Recognition
  - https://ydwen.github.io/papers/WenECCV16.pdf
- SphereFace: Deep Hypersphere Embedding for Face Recognition
  - https://arxiv.org/abs/1704.08063
- ArcFace: Additive Angular Margin Loss for Deep Face Recognition
  - https://arxiv.org/abs/1801.07698
- CosFace: Large Margin Cosine Loss for Deep Face Recognition
  - https://arxiv.org/abs/1801.09414

Papers for Wed, May 05

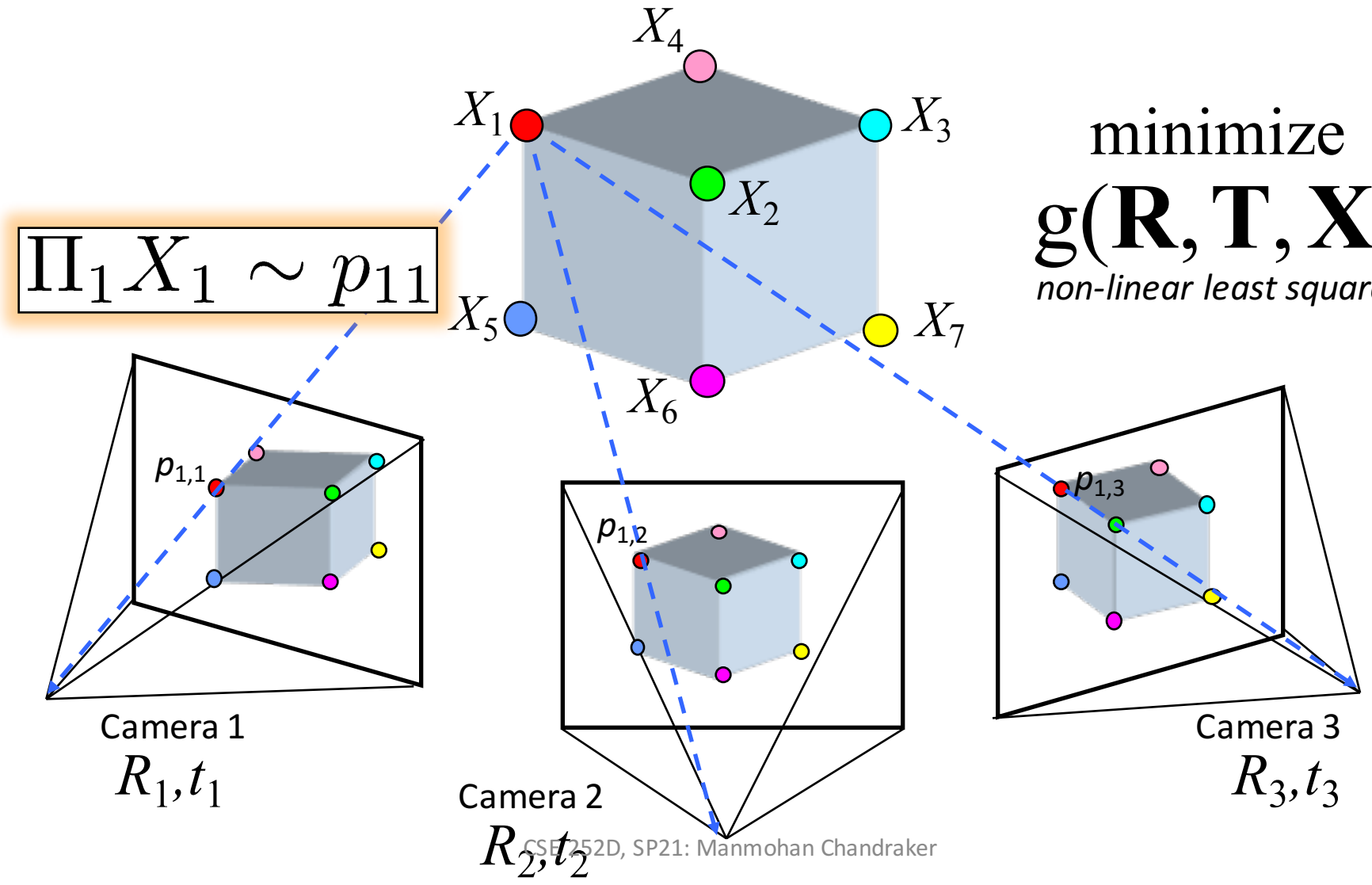
- Deep High-Resolution Representation Learning for Human Pose Estimation
  - https://arxiv.org/abs/1902.09212
- Simple Baselines for Human Pose Estimation and Tracking
  - https://arxiv.org/abs/1804.06208
- OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields
  - https://arxiv.org/abs/1812.08008
- End-to-end Recovery of Human Shape and Pose
  - https://arxiv.org/abs/1712.06584

Recap

Unordered or Ordered Images

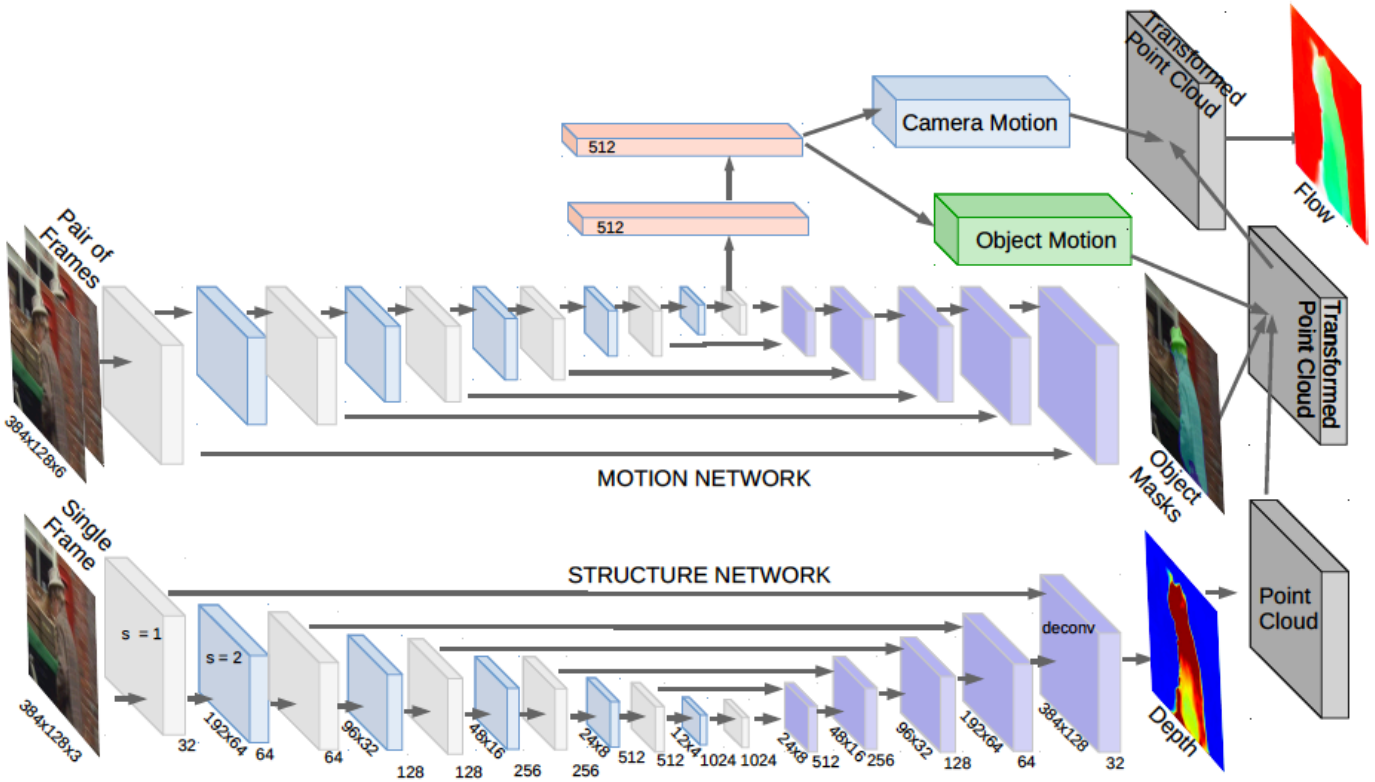


Structure from motion



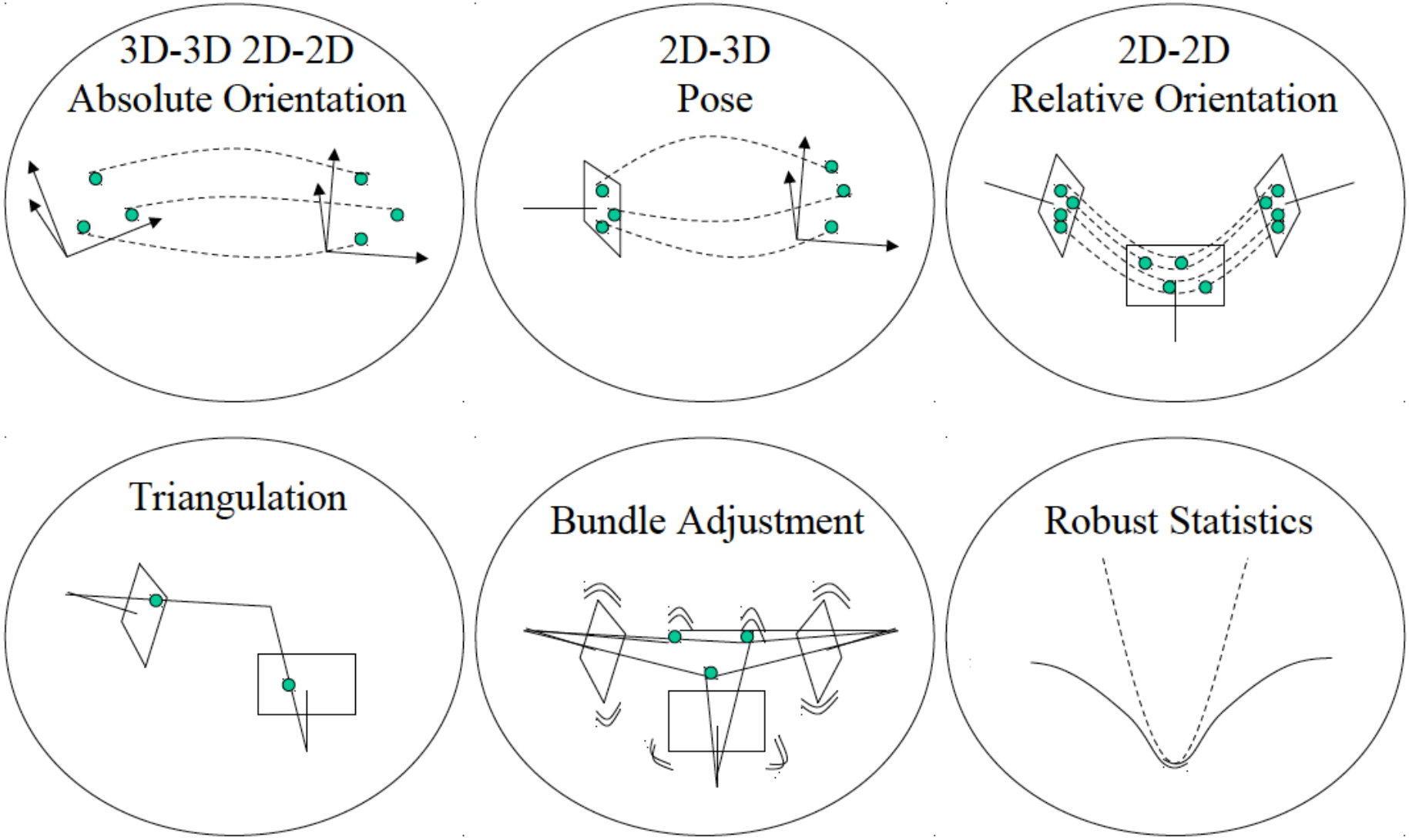
General Recipe to Learn Structure and Motion

- Estimate depths (convert to 3D points given calibration) in frame t
- Estimate motion from frame t to t+1 for background and objects
- Project estimated 3D points to frame t+1 using the estimated motions
- Use a consistency condition to declare matches as good

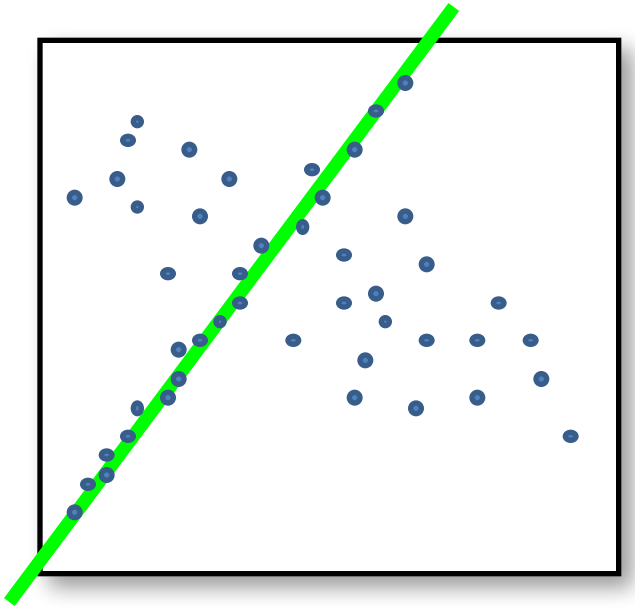


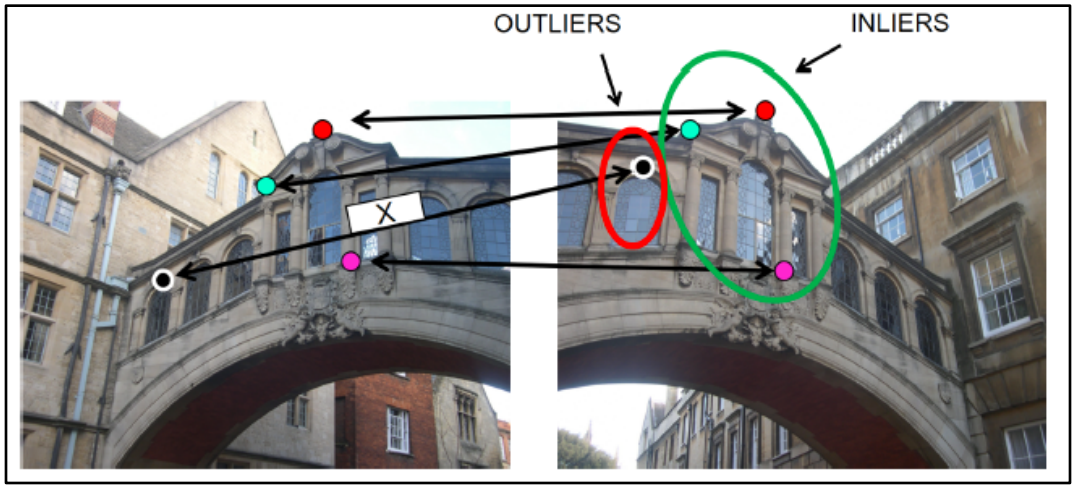
[SFM-Net, ICCV 2017]

Toolkit for Practical SFM



RANSAC

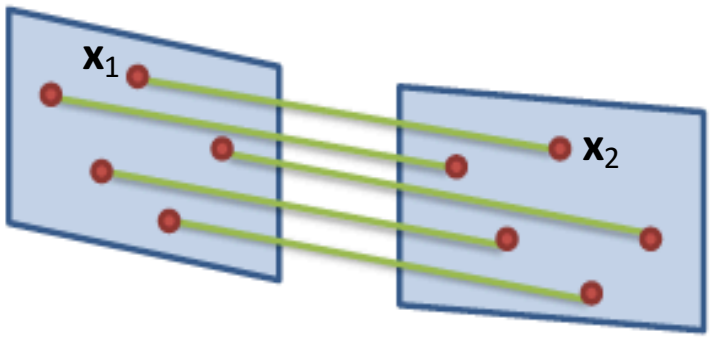




- General version:
  1. Randomly choose *s* samples
     (Typically *s* = minimum sample size to fit a model)
  2. Fit a model to those samples
  3. Count number of inliers consistent with the model
  4. Repeat *N* times
  5. Choose the model with the largest set of inliers

RANSAC to Estimate Fundamental Matrix

- For *N* times
  - Randomly pick 8 correspondences across two images
  - Compute $\mathbf{F}$ using these 8 correspondences
  - Among all correspondences, count number of inliers with $\mathbf{x}_1^T \mathbf{F} \mathbf{x}_2$ close to 0 (less than a threshold)
- Pick the $\mathbf{F}$ with the largest number of inliers



Some Recipes for SFM to Work

- Do everything you can to remove outliers

- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

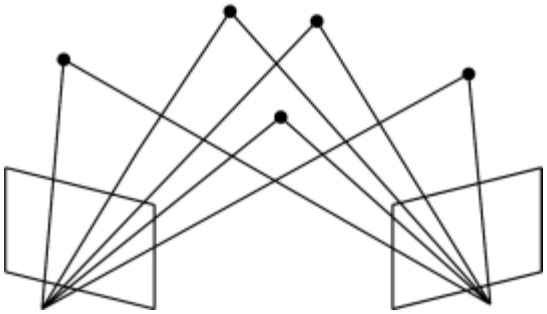
- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

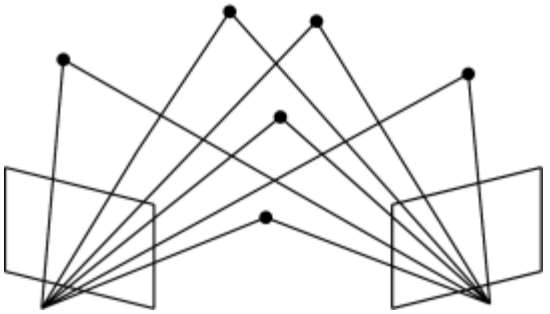
Minimal Problems in Computer Vision

- There is a large zoo of minimal problems that have known solutions

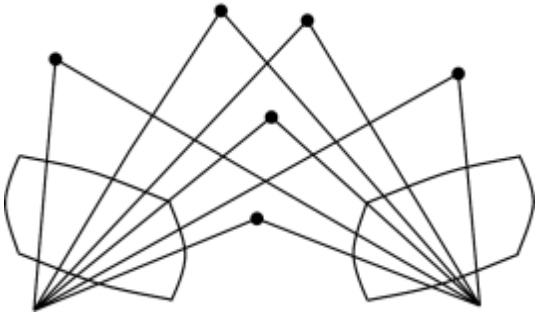
5-point relative pose with known K



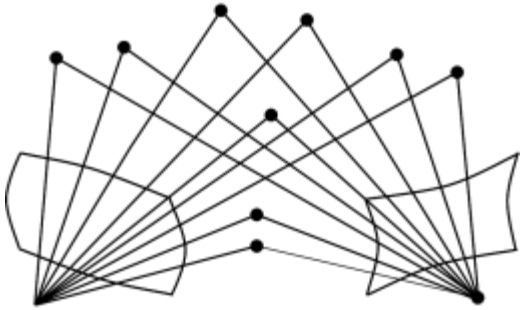
6-point relative pose with unknown f



6-point relative pose with unknown r

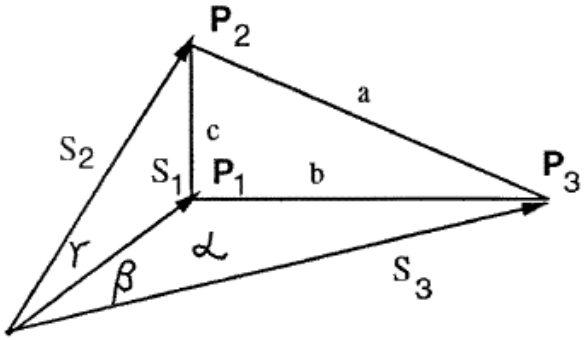


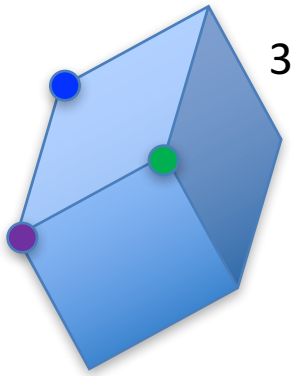
9-point relative pose with unknown and different f, r



Three-Point Absolute Pose Estimation

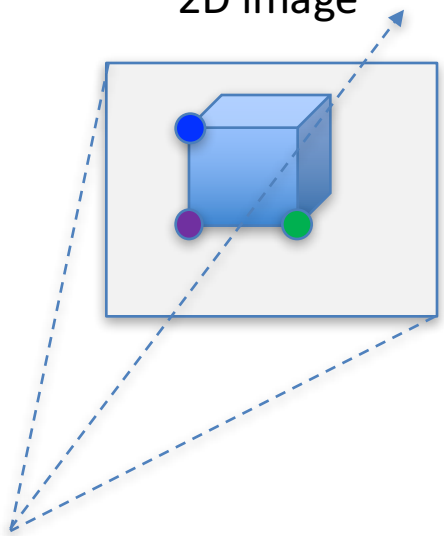
- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point (x, y)
- Given p_1, p_2, p_3 in world coordinates, find positions in camera coordinates





3D scene

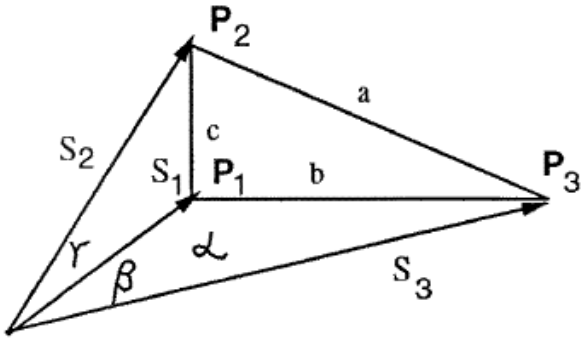
2D image



Determine how to move camera, such that corresponding 2D-3D points align

Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point (x, y)
- Given p_1, p_2, p_3 in world coordinates, find positions in camera coordinates



Knowns

$$a = \|p_2 - p_3\| \qquad u_i = f\frac{x_i}{z_i} \qquad \cos\alpha = j_2 \cdot j_3$$
$$b = \|p_1 - p_3\| \qquad v_i = f\frac{y_i}{z_i}. \qquad \cos\beta = j_1 \cdot j_3$$
$$c = \|p_1 - p_2\|. \qquad \cos\gamma = j_1 \cdot j_2$$

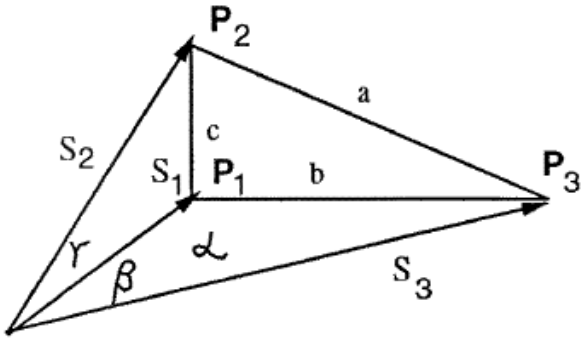
Distances between points | Image points | Angle between image rays j_i

Unknowns

Distance to 3D points from camera center, s_i

Three-Point Absolute Pose Estimation

- Find camera pose from given 3D-2D correspondences
- Minimal case is 3 points: 6 degrees of freedom, 2 constraints per point (x, y)
- Given p_1, p_2, p_3 in world coordinates, find positions in camera coordinates



Knowns

$$a = \|p_2 - p_3\| \qquad u_i = f\frac{x_i}{z_i} \qquad \cos\alpha = j_2 \cdot j_3$$
$$b = \|p_1 - p_3\| \qquad v_i = f\frac{y_i}{z_i}. \qquad \cos\beta = j_1 \cdot j_3$$
$$c = \|p_1 - p_2\|. \qquad \cos\gamma = j_1 \cdot j_2$$

Distances between points | Image points | Angle between image rays j_i

Constraints: Law of cosines

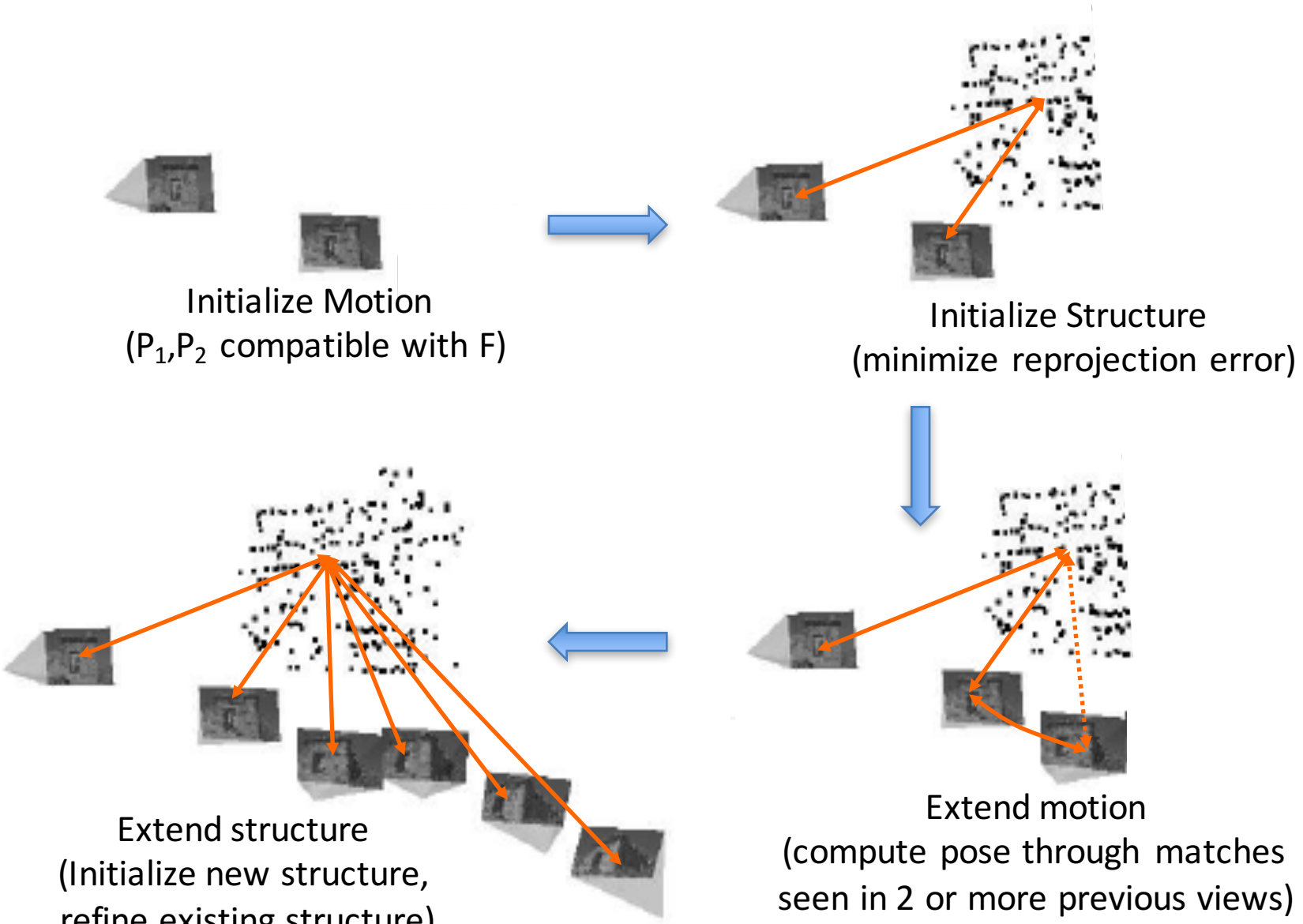
$$s_2^2 + s_3^2 - 2s_2 s_3 \cos\alpha = a^2$$
$$s_1^2 + s_3^2 - 2s_1 s_3 \cos\beta = b^2$$
$$s_1^2 + s_2^2 - 2s_1 s_2 \cos\gamma = c^2$$

Unknowns

Distance to 3D points from camera center, s_i

All techniques yield fourth-degree polynomial

Sequential Structure from Motion



Some Recipes for SFM to Work

- Do everything you can to remove outliers

- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

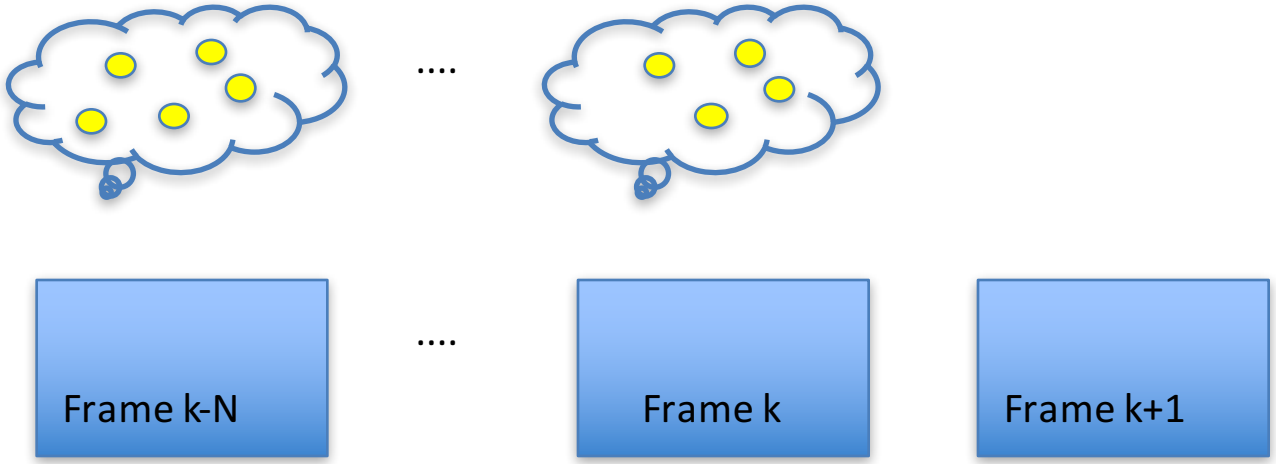
- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

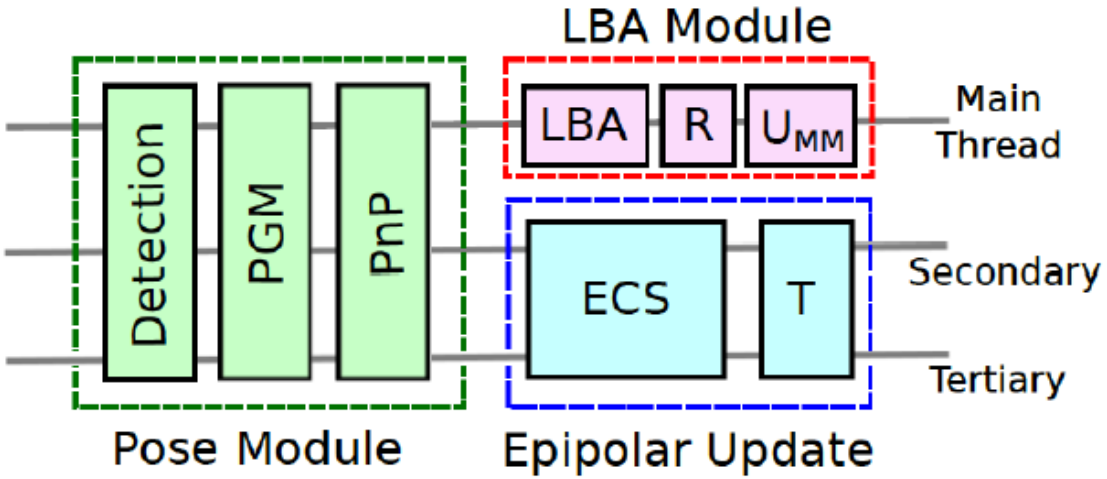
- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

Real-Time SFM Systems

Real-Time SFM: Steady-State

- Point cloud and camera poses available at frame k, find pose in frame k+1





[Song et al., Parallel Real-Time Monocular Visual Odometry]

Real-Time SFM: Two Options

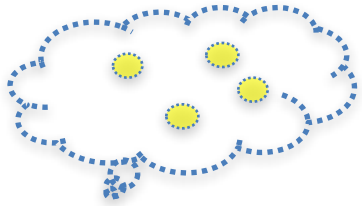
Option 1

Pose: [I | 0] Pose: [R_1 | t_1]

Find correspondences, estimate essential matrix, decompose into [R | t]

Real-Time SFM: Two Options

Option 1



Triangulate to get 3D points

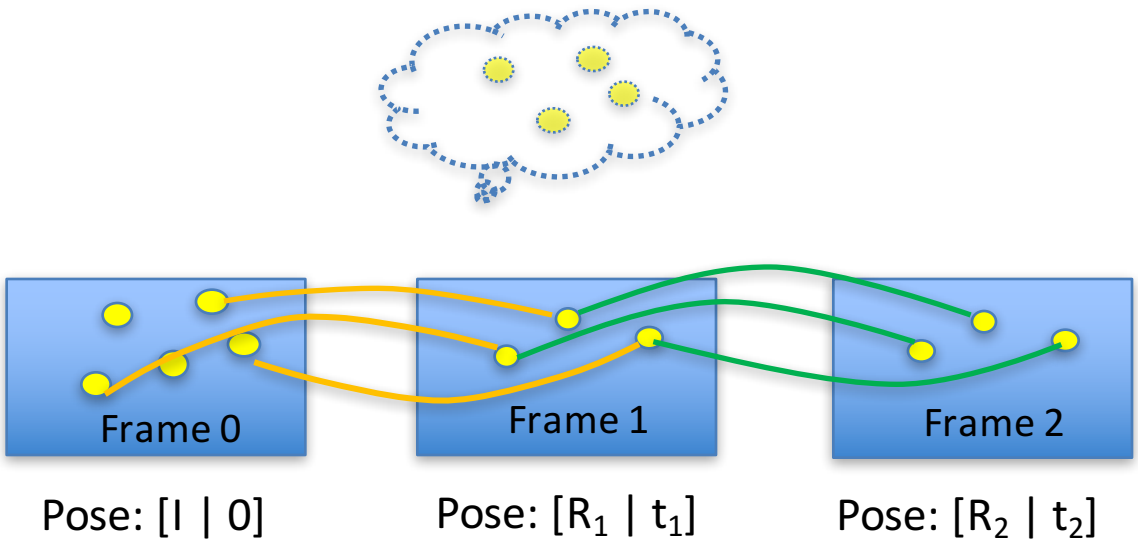
Pose: [I | 0]

Pose: [R_1 | t_1]

Find correspondences, estimate essential matrix, decompose into [R | t]

Real-Time SFM: Two Options

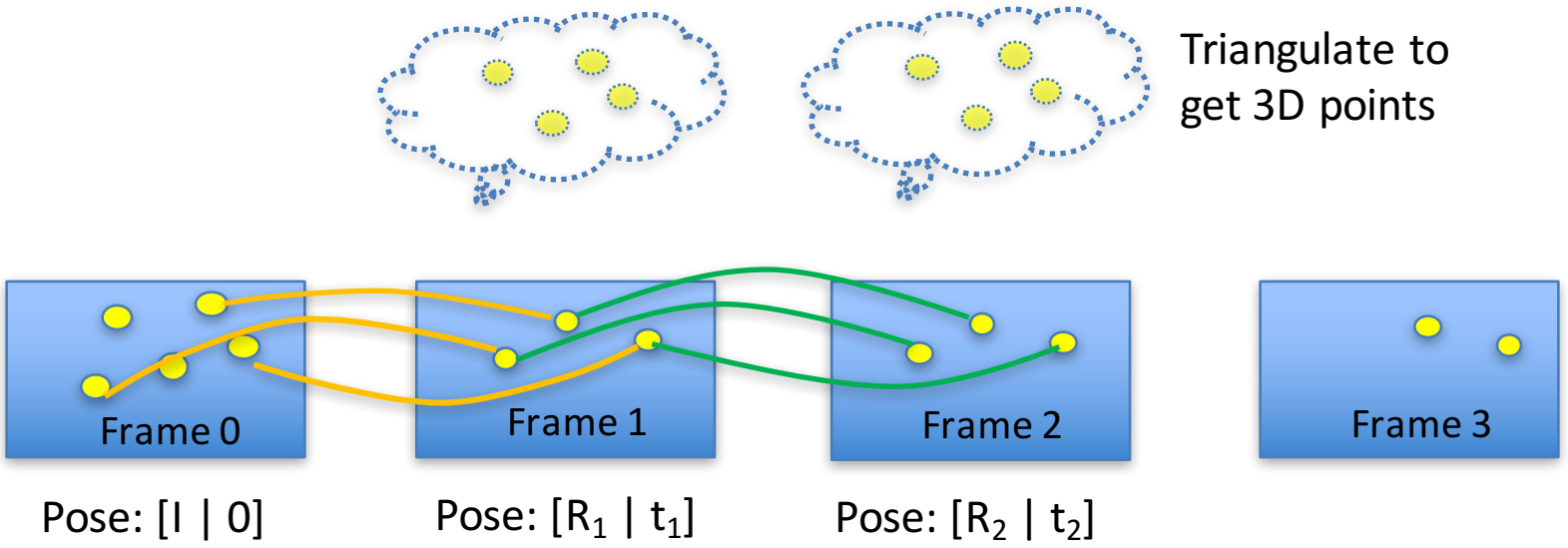
Option 1



Find correspondences, estimate essential matrix, decompose into [R | t]
Concatenate with pose of previous frame, to obtain pose [R_2 | t_2]

Real-Time SFM: Two Options

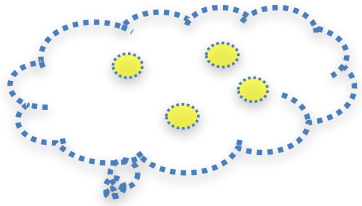
Option 1



Find correspondences, estimate essential matrix, decompose into [R | t]
Concatenate with pose [R_1 | t_1] of frame 1, to obtain pose [R_2 | t_2] of frame 2

Real-Time SFM: Two Options

Option 2



Triangulate to get 3D points

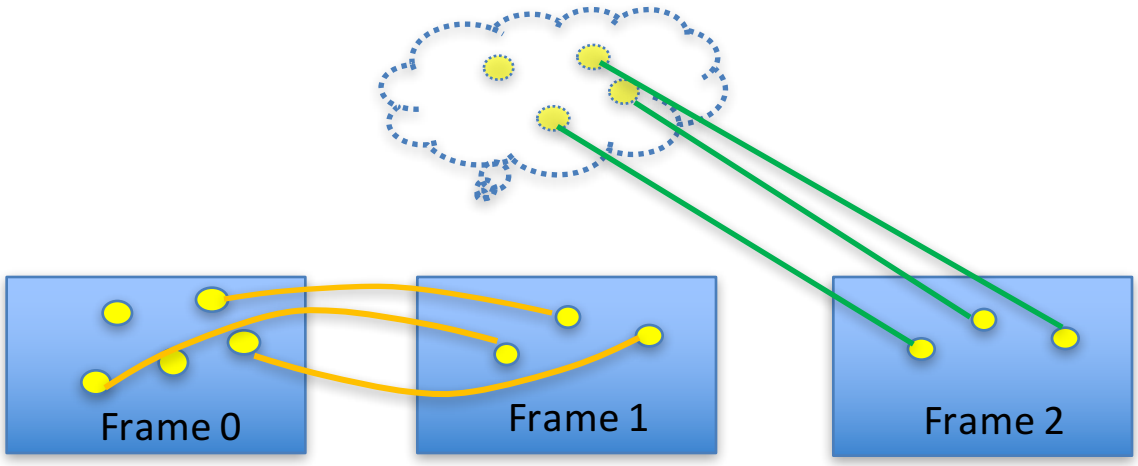
Pose: [I | 0]

Pose: [R_1 | t_1]

Initialize similar as earlier, to get pose of frame 1 and 3D points

Real-Time SFM: Two Options

Option 2

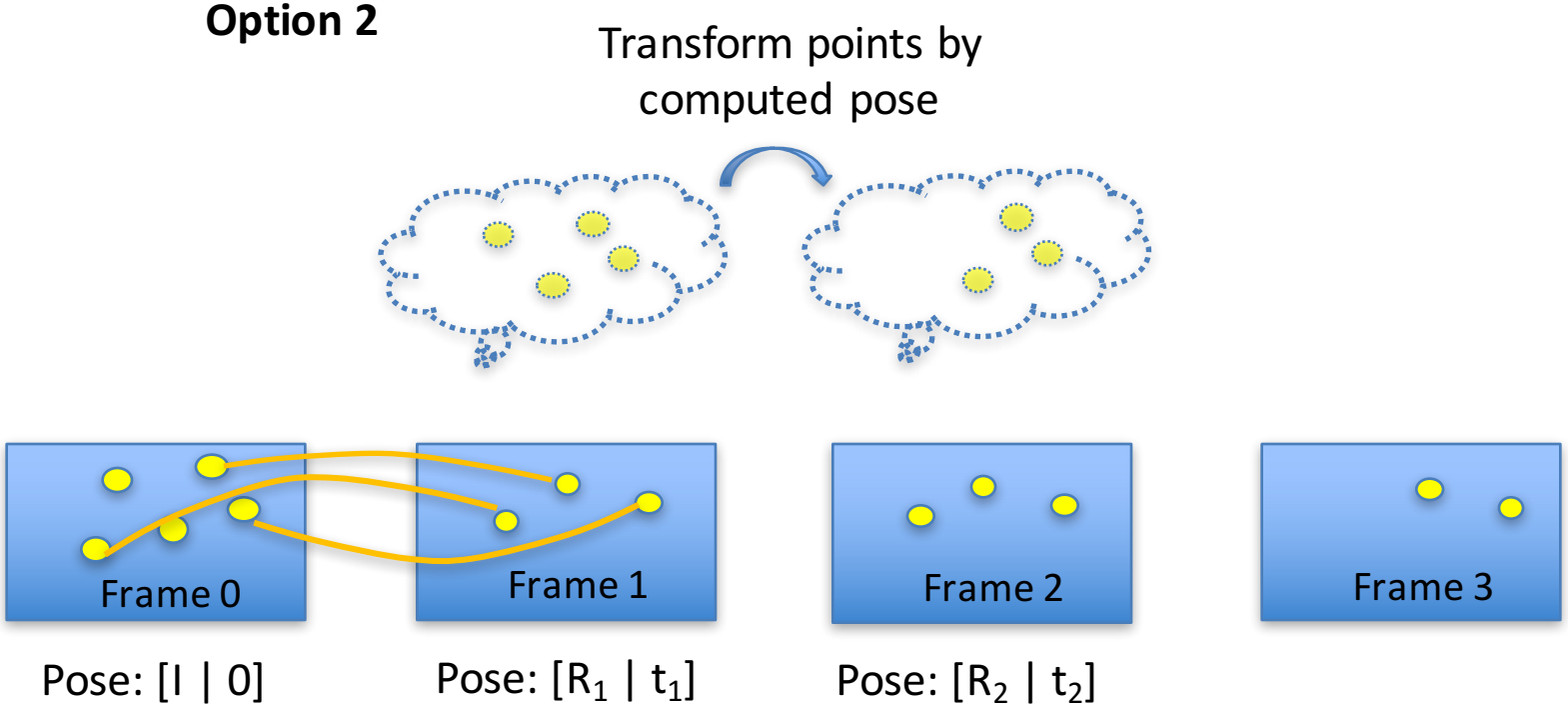


Pose: [I | 0] Pose: [R_1 | t_1] Pose: [R_2 | t_2]

Initialize similar as earlier, to get pose of frame 1 and 3D points
But for subsequent frames, use 3-point absolute pose to get pose [R_2 | t_2]

Real-Time SFM: Two Options

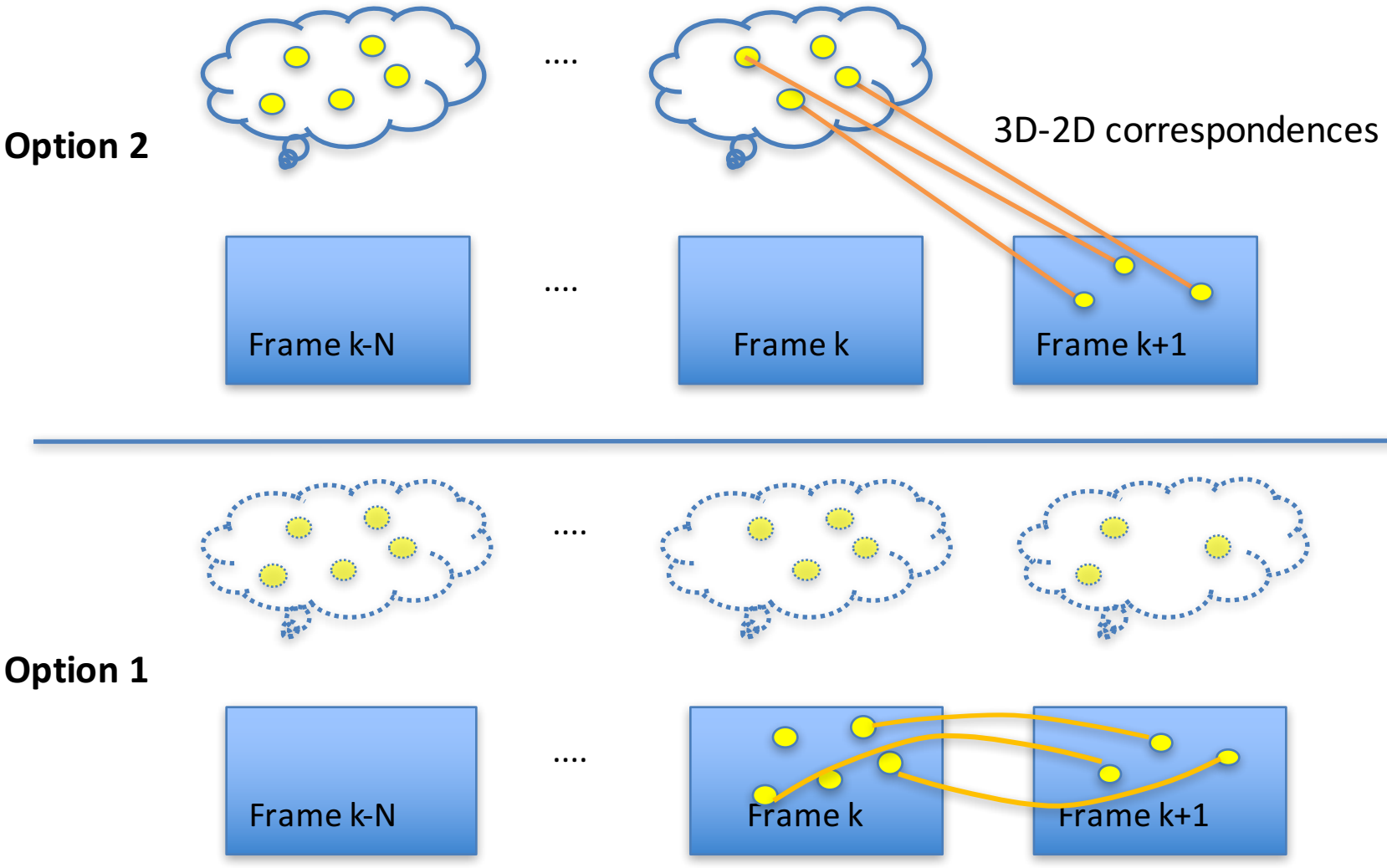
Option 2



Initialize similar as earlier, to get pose of frame 1 and 3D points
But for subsequent frames, use 3-point absolute pose to get pose [R_2 | t_2]

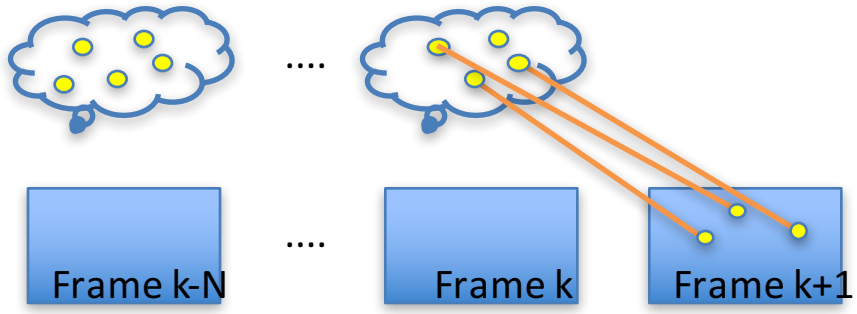
Real-Time SFM: Steady-State

- Usually absolute pose estimations rather than relative pose



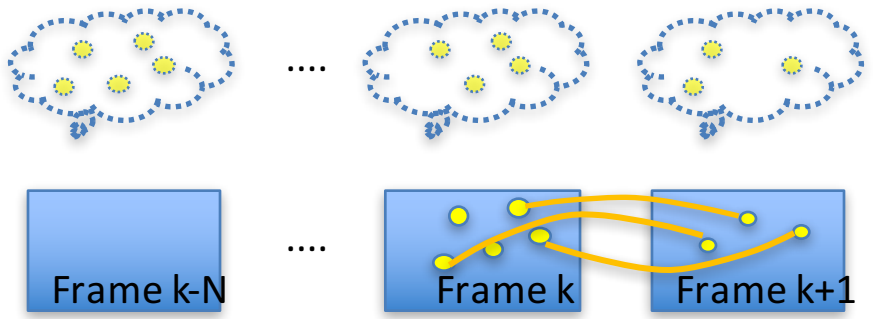
Real-Time SFM: Steady-State

- Long feature tracks can be constructed (same 3D point visible in many views)
- Repeated bundle adjustment and outlier removal
- Cheaper, less error to solve 3-point absolute pose



3D-2D correspondences

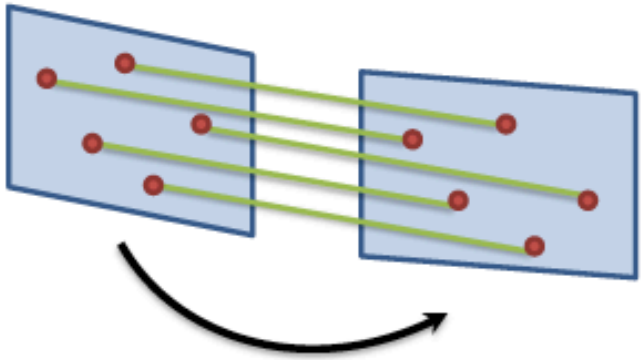
- Rely on per-frame matches, shorter baselines for reconstruction
- New points in every frame, less chances for outlier removal
- More expensive and higher error to solve 5-point relative pose



2D-2D correspondences

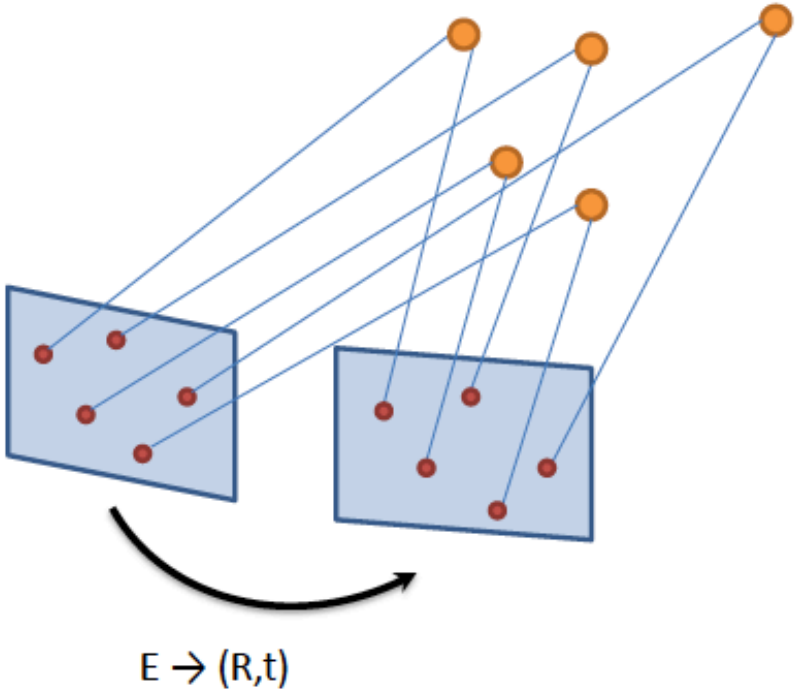
Step 1: Initialization

- Two views initialization:
  - 5-Point algorithm (Minimal Solver)
  - 8-Point linear algorithm
  - 7-Point algorithm



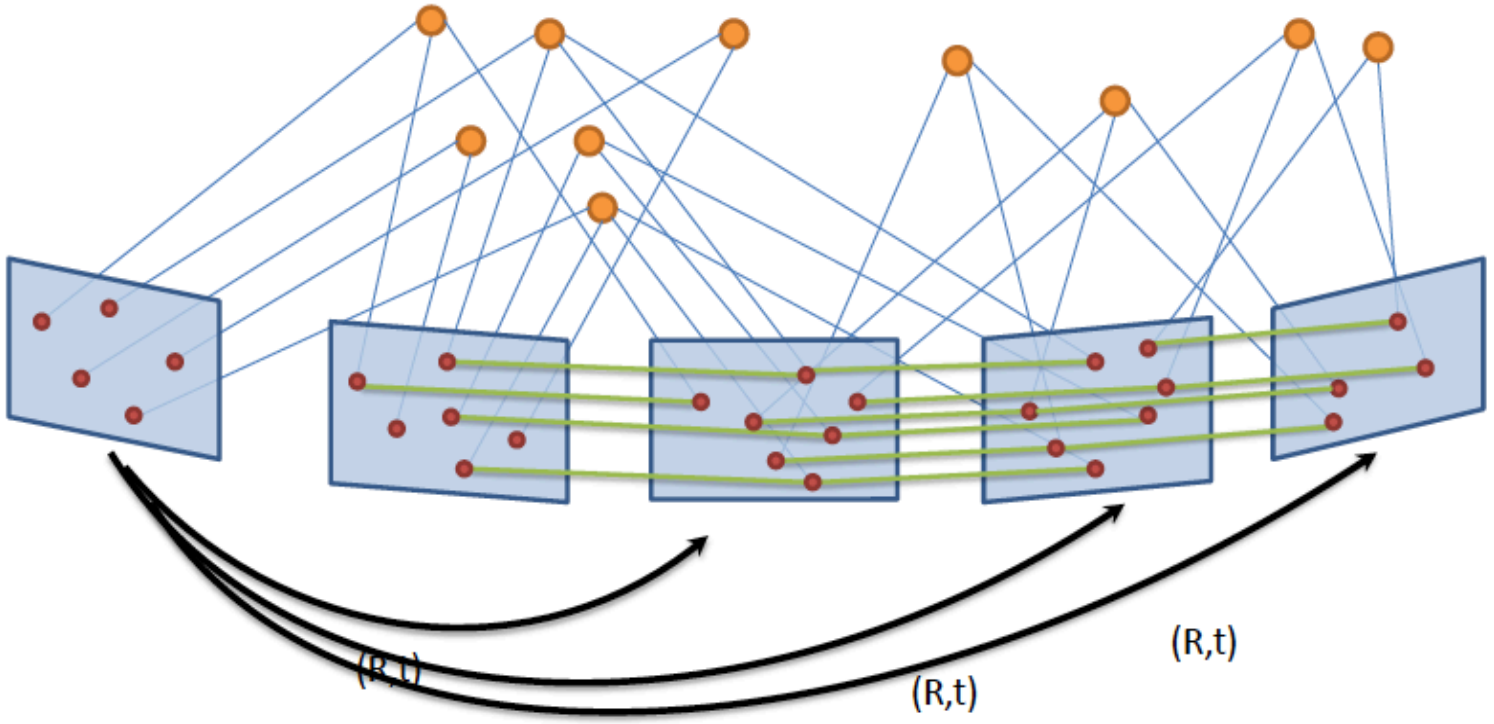
Step 2: Generate 3D Points

- Triangulation: 3D Points



Step 3: Estimate Pose in Next Views

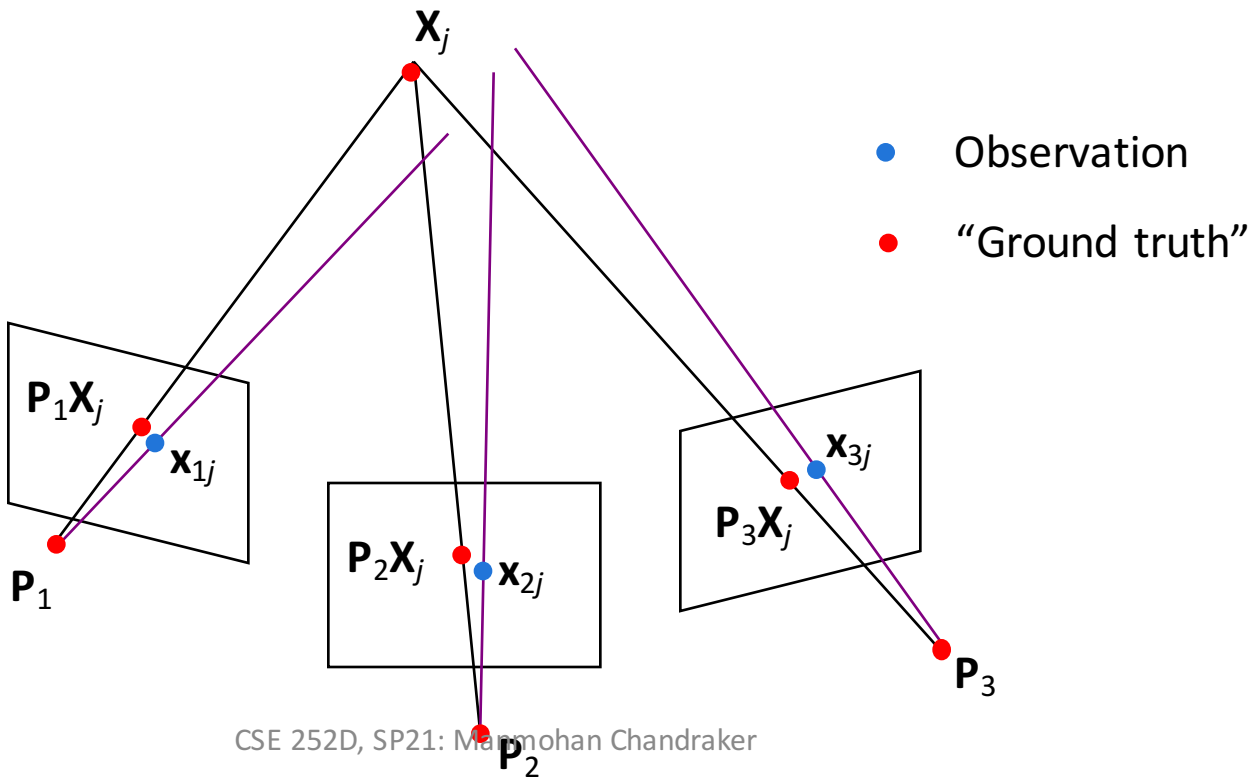
- Subsequent views: Perspective pose estimation



Bundle Adjustment

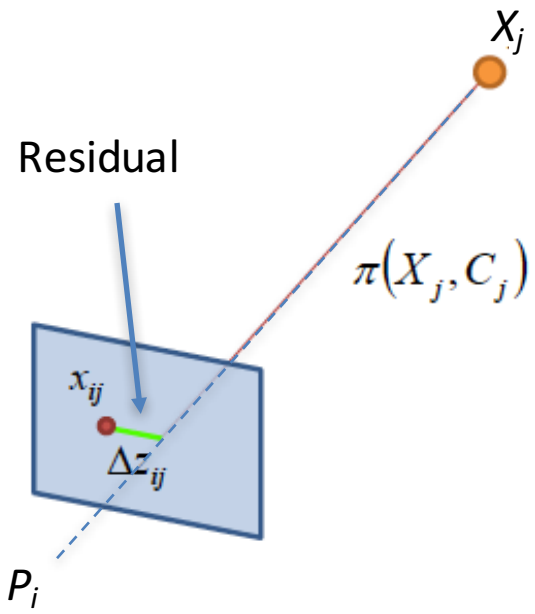
- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$Error\,(P,X) = \sum_{Cameras\ P_i} \sum_{Points\ X_j} |x_{ij} - P_i X_j|^2$$



Bundle Adjustment

- Refine a visual reconstruction to produce jointly optimal 3D structures X and camera poses P
- Minimize total reprojection errors Δz.



$$Error\,(P,X) = \sum_{Cameras\ P_i} \sum_{Points\ X_j} |x_{ij} - P_i X_j|^2$$

$$Error\,(P,X) = \sum_{Cameras\ P_i} \sum_{Points\ X_j} |\Delta z_{ij}|^2$$

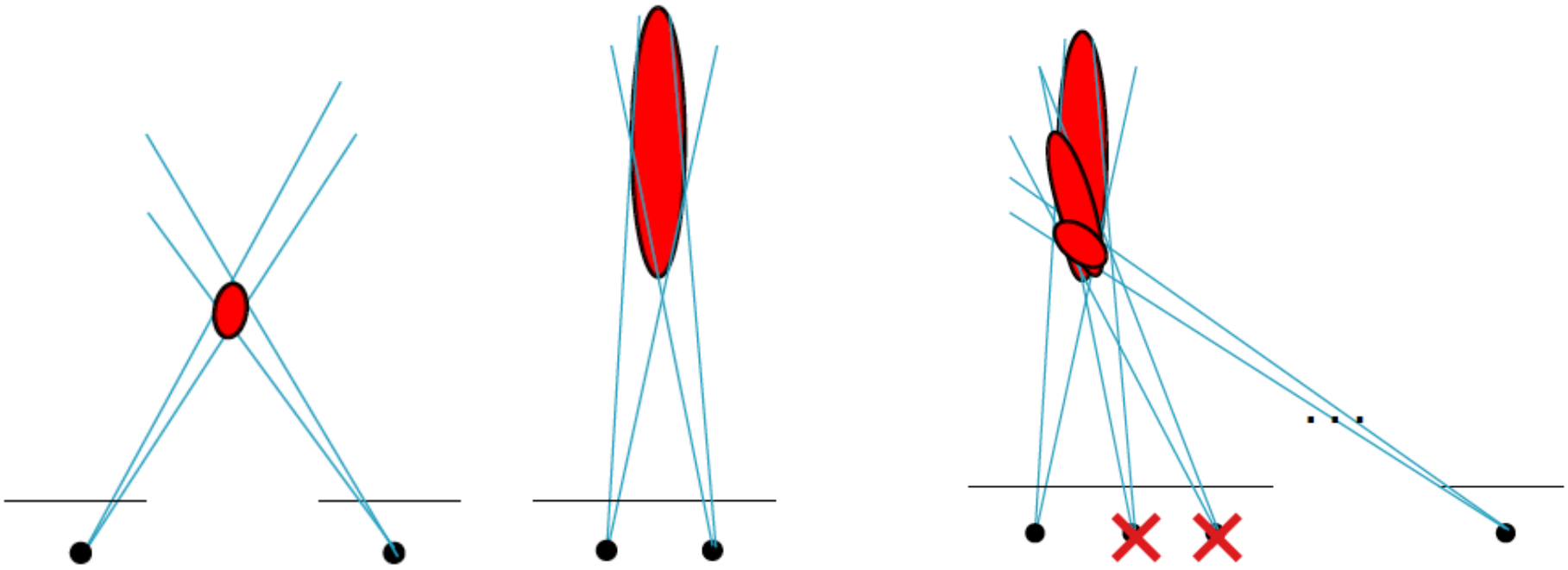
Minimize sum of squared residuals.

Bundle Adjustment: Nonlinear Least Squares

- Minimize the cost function:
  1. Gradient Descent
  2. Newton Method
  3. Gauss-Newton
  4. Levenberg-Marquardt

Keyframes

- Provide a large enough baseline for triangulation



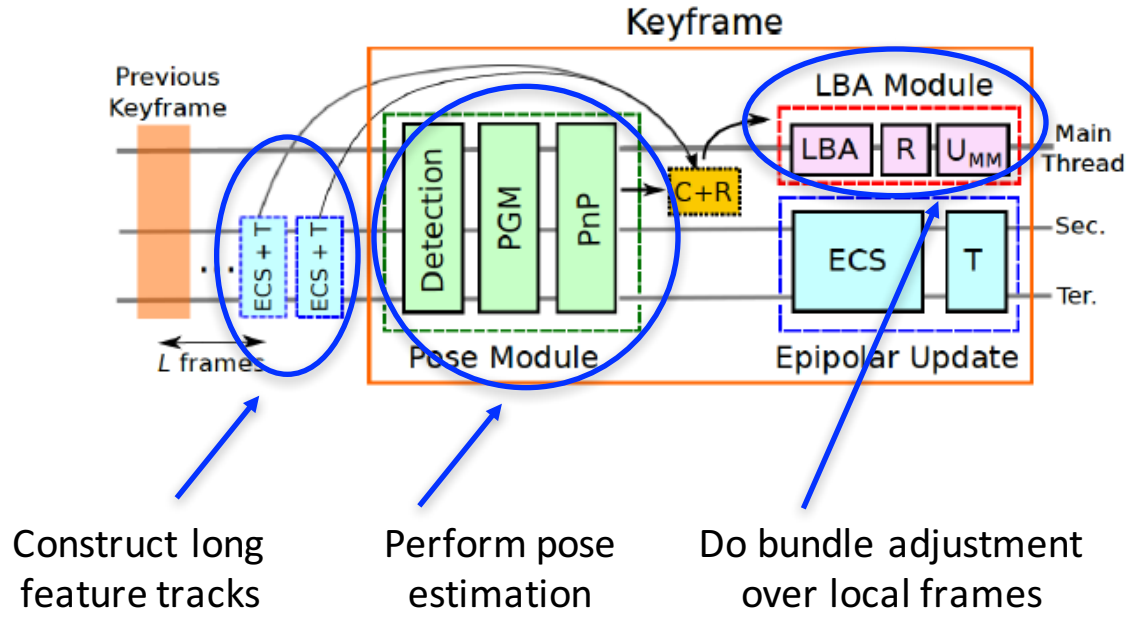
Wide baseline

Narrow baseline

Keyframes

- Provide a large enough baseline for triangulation

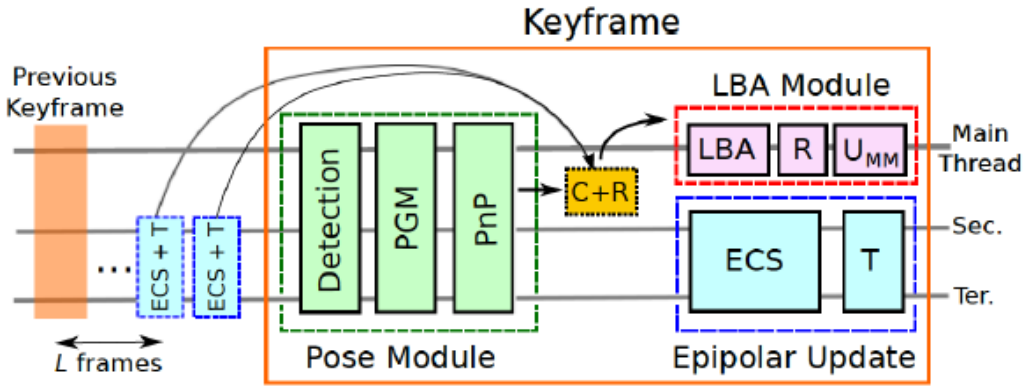
- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement



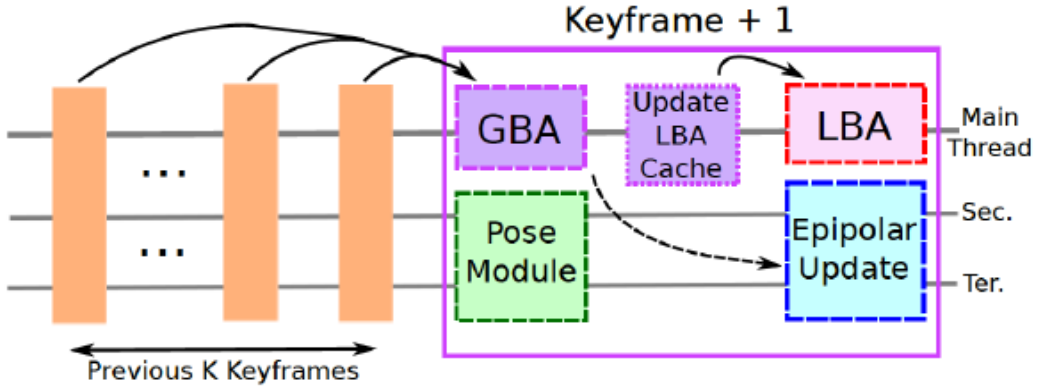
Keyframes

- Provide a large enough baseline for triangulation

- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement



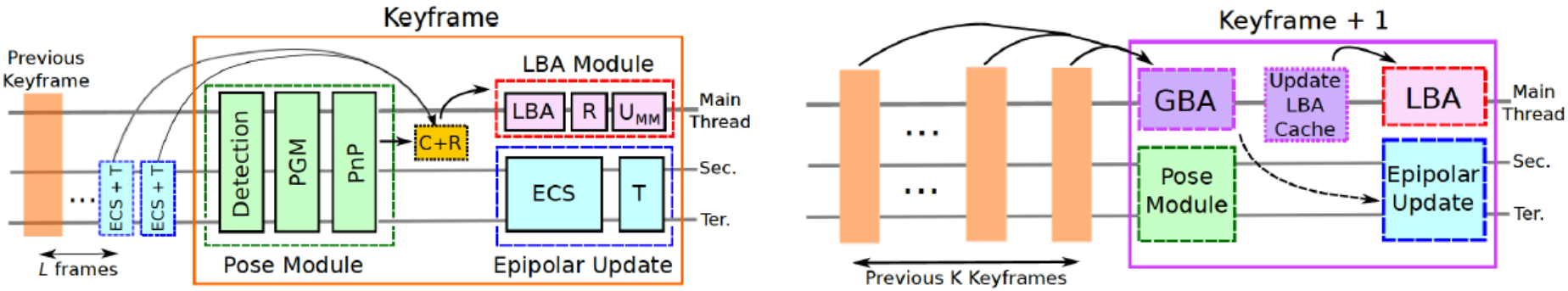
- Do a global bundle adjustment over keyframes and associated 3D points



Keyframes

- Provide a large enough baseline for triangulation
- Add new 3D points to point cloud at the keyframe
- Collect long tracks, triangulate and do nonlinear refinement
- Do a global bundle adjustment over keyframes and associated 3D points

- Criteria to add keyframes
  - Camera has moved far enough : $\frac{\textit{keyframe distance}}{\textit{average-depth}} > \textit{threshold}$
  - Keyframe not added for long enough
  - Number of remaining 3D points not large enough



[Song and Chandraker, 2015]

Some Recipes for SFM to Work

- Do everything you can to remove outliers

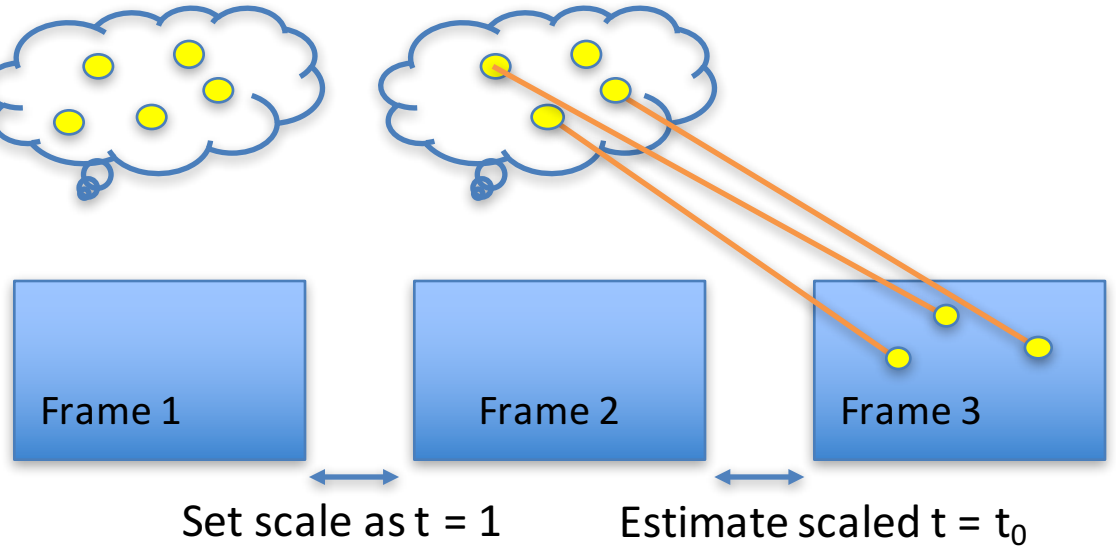
- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

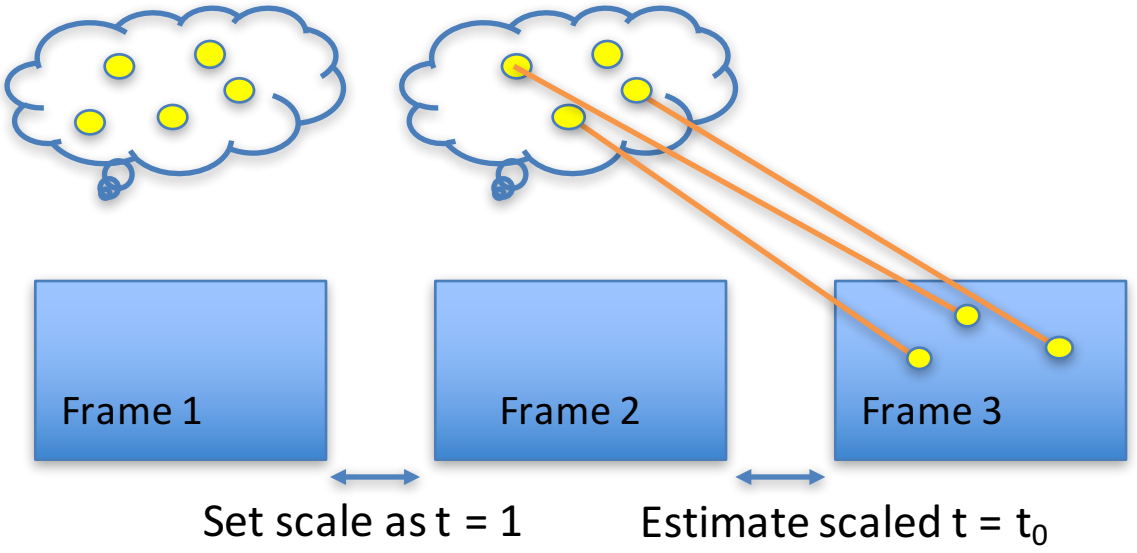
Scale Drift Correction

Choose scale arbitrarily, for example, translation from frame 1 to 2

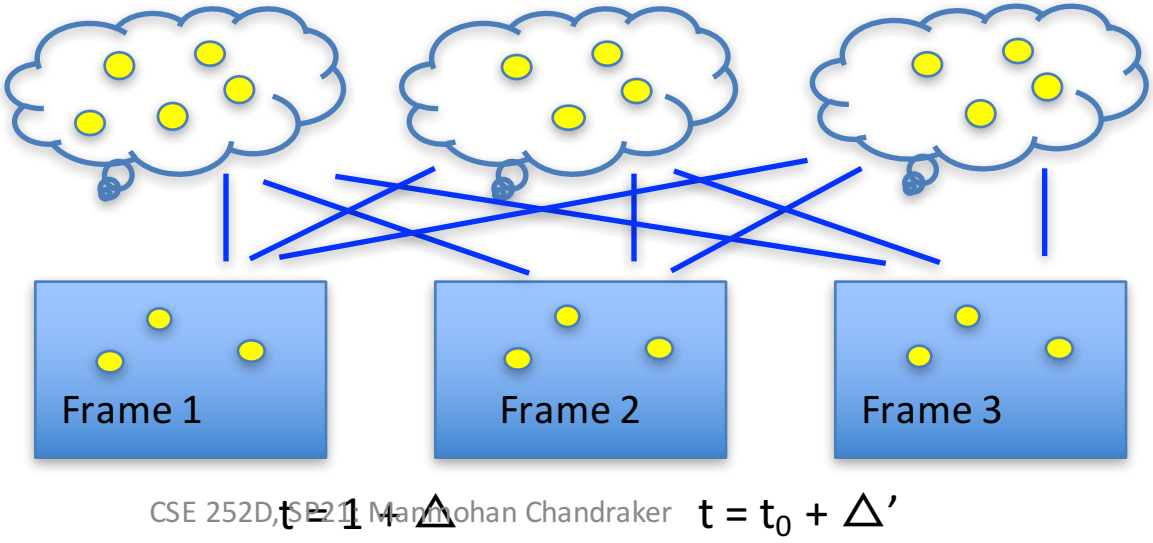


Scale Drift Correction

Choose scale arbitrarily, for example, translation from frame 1 to 2

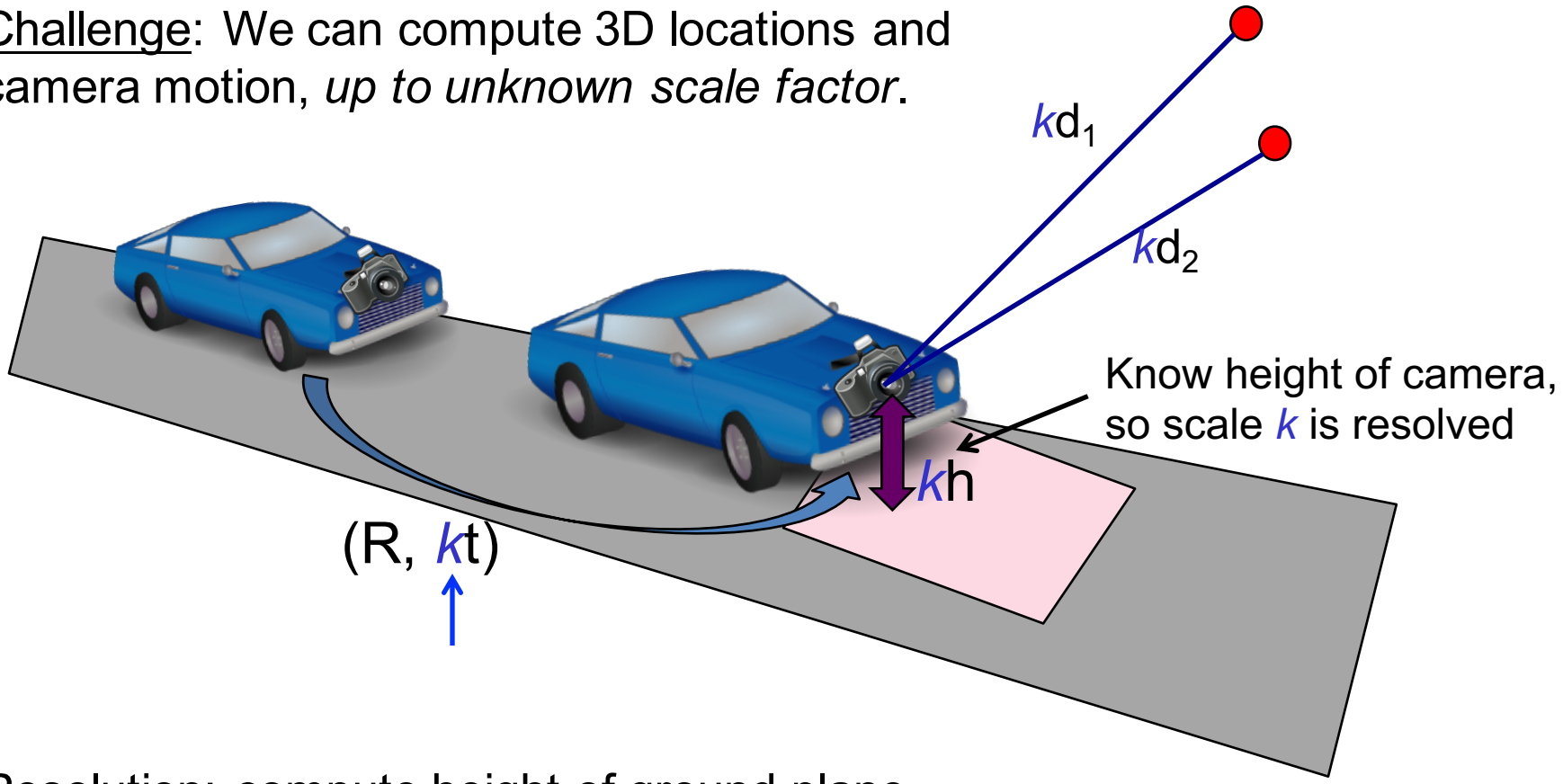


Every bundle adjustment causes 3D points and cameras to change



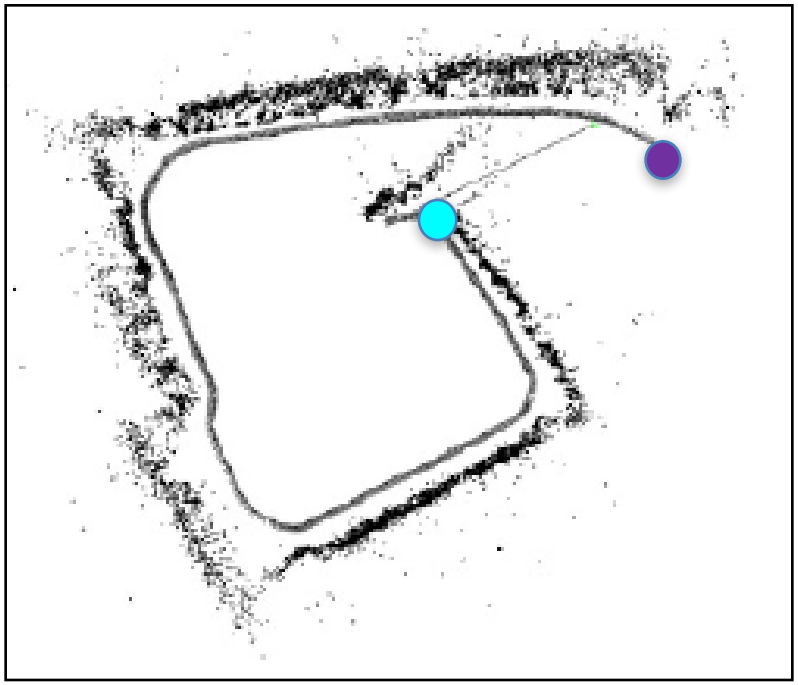
Scale Drift Correction

Challenge: We can compute 3D locations and camera motion, *up to unknown scale factor*.

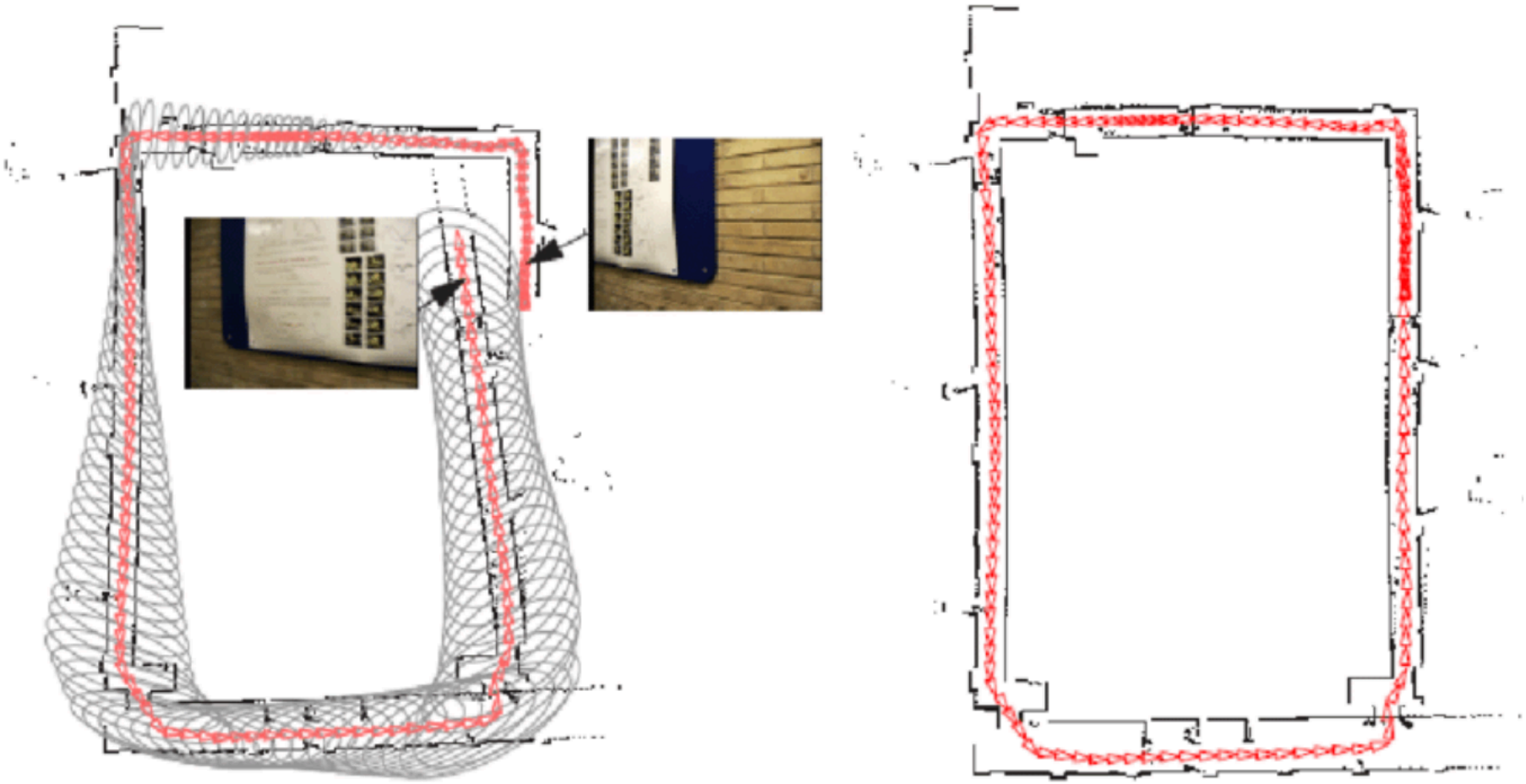


Resolution: compute height of ground plane.

Loop Closure



Loop Closure for Scale Drift Correction



Recognize whether same image is observed as in some previous frame.

Some Recipes for SFM to Work

- Do everything you can to remove outliers

- Solve minimal problems to estimate geometric entities
  - Keeps RANSAC tractable
  - Typically, expect to spend 0.01ms

- Strategically consider what variables to optimize
  - Keyframe-based designs are successful
  - Try to robustly build long feature tracks
  - Do bundle adjustment whenever possible

- Drift is inevitable, so have a plan to address it
  - Local scale correction and global pose correction when possible

Real-Time SFM Systems

- Use fundamental building blocks, with different system choices