

# **Lecture 8.2**

## **Structure from Motion**

Thomas Opsahl

# More-than-two-view geometry

## Correspondences (matching)

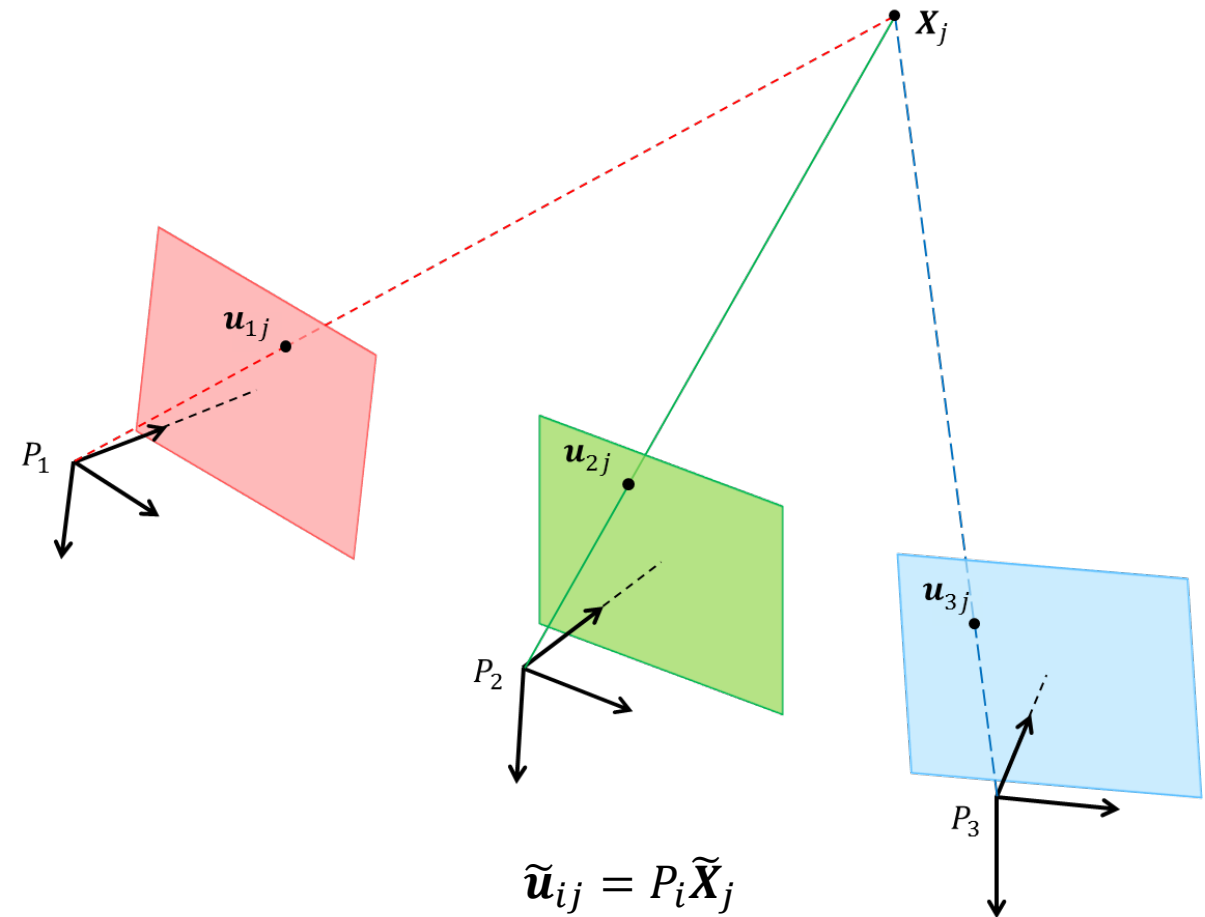
- More views enables us to reveal and remove more mismatches than we can do in the two-view case
- More views also enables us to predict correspondences that can be tested with or without the use of descriptors

## Scene geometry (structure)

- Effect of more views on determining the 3D structure of the scene?

## Camera geometry (motion)

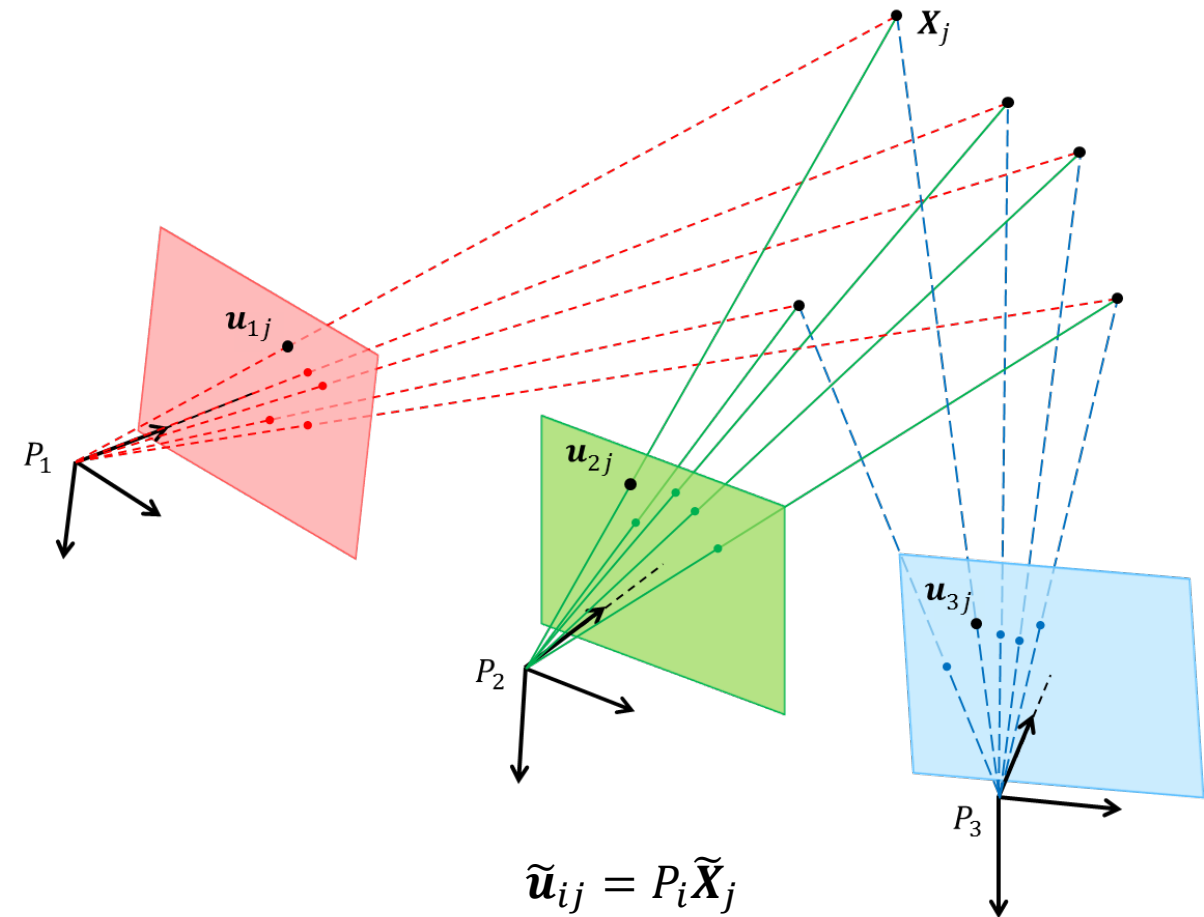
- Effect of more views on determining camera poses?



# Structure from Motion

## Problem

Given  $m$  images of  $n$  fixed 3D points, estimate the  $m$  projection matrices  $P_j$  and the  $n$  points  $X_j$  from the  $m \cdot n$  correspondences  $u_{ij} \leftrightarrow u_{kj}$

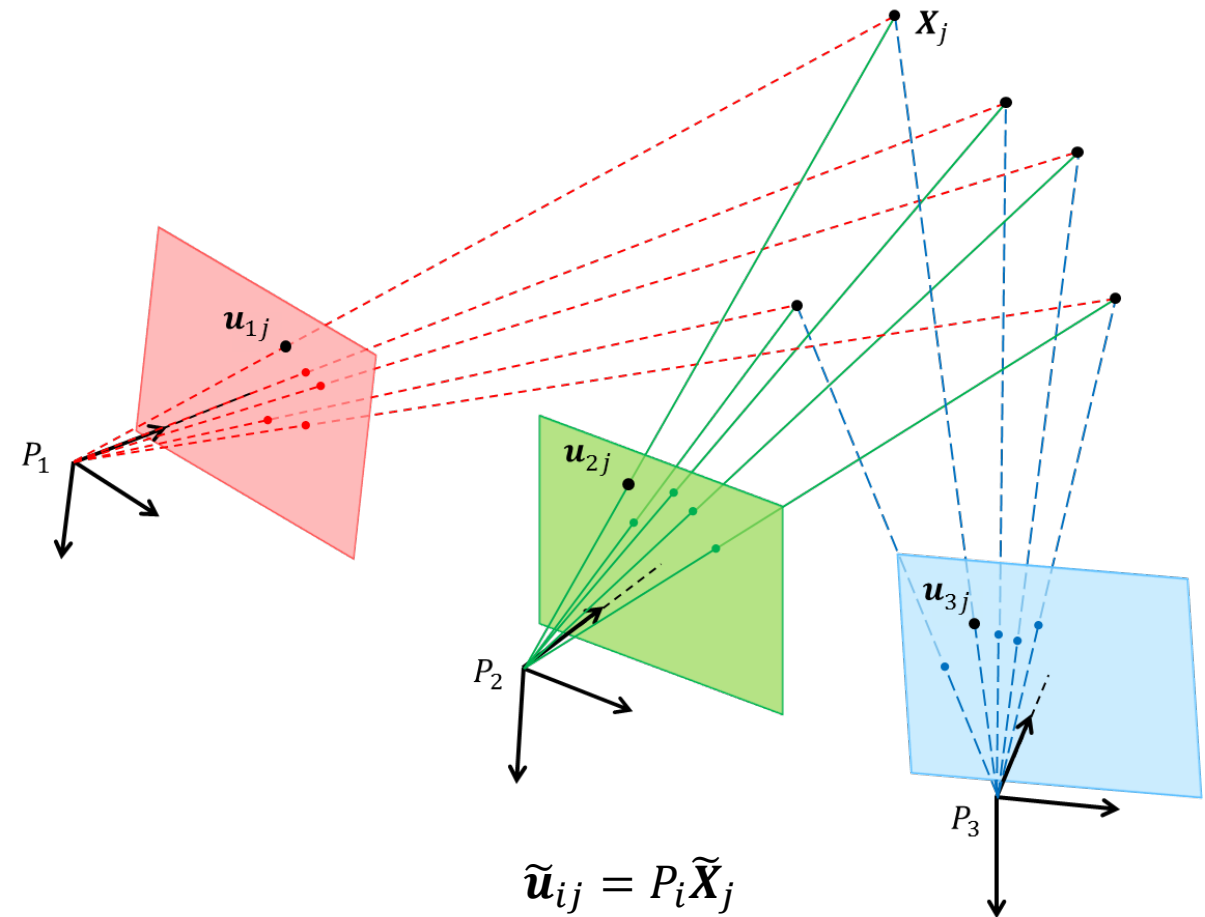


# Structure from Motion

## Problem

Given  $m$  images of  $n$  fixed 3D points, estimate the  $m$  projection matrices  $P_j$  and the  $n$  points  $X_j$  from the  $m \cdot n$  correspondences  $u_{ij} \leftrightarrow u_{kj}$

- We can solve for structure and motion when
$$2mn \geq 11m + 3n - 15$$
- In the general/uncalibrated case, cameras and points can only be recovered up to a projective ambiguity ( $\tilde{u}_{ij} = P_i Q^{-1} Q \tilde{X}_j$ )
- In the calibrated case, they can be recovered up to a similarity (scale)
  - Known as Euclidean/metric reconstruction

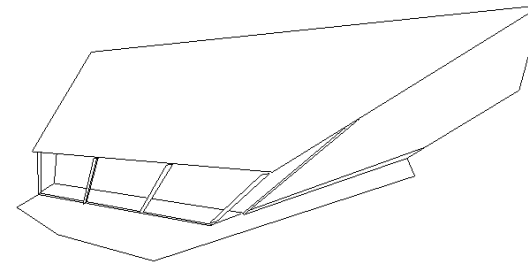
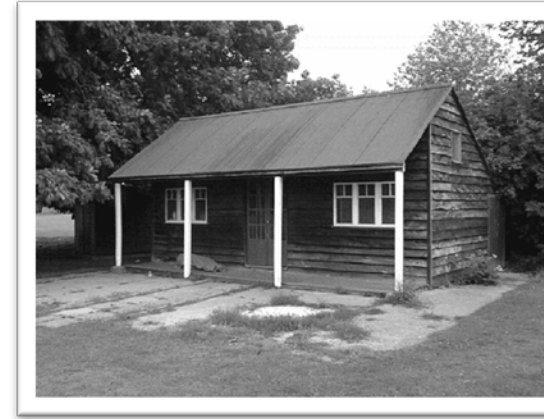


# Structure from Motion

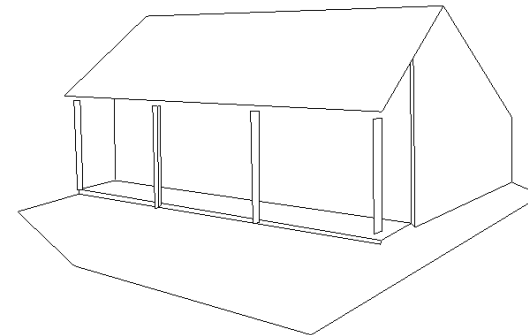
## Problem

Given  $m$  images of  $n$  fixed 3D points, estimate the  $m$  projection matrices  $P_j$  and the  $n$  points  $X_j$  from the  $m \cdot n$  correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{u}_{kj}$

- We can solve for structure and motion when
$$2mn \geq 11m + 3n - 15$$
- In the general/uncalibrated case, cameras and points can only be recovered up to a projective ambiguity ( $\tilde{\mathbf{u}}_{ij} = P_i Q^{-1} Q \tilde{X}_j$ )
- In the calibrated case, they can be recovered up to a similarity (scale)
  - Known as Euclidean/metric reconstruction



Projective reconstruction



Metric reconstruction

Images courtesy of Hartley & Zisserman <http://www.robots.ox.ac.uk/~vgg/hzbook/>

# Structure from Motion

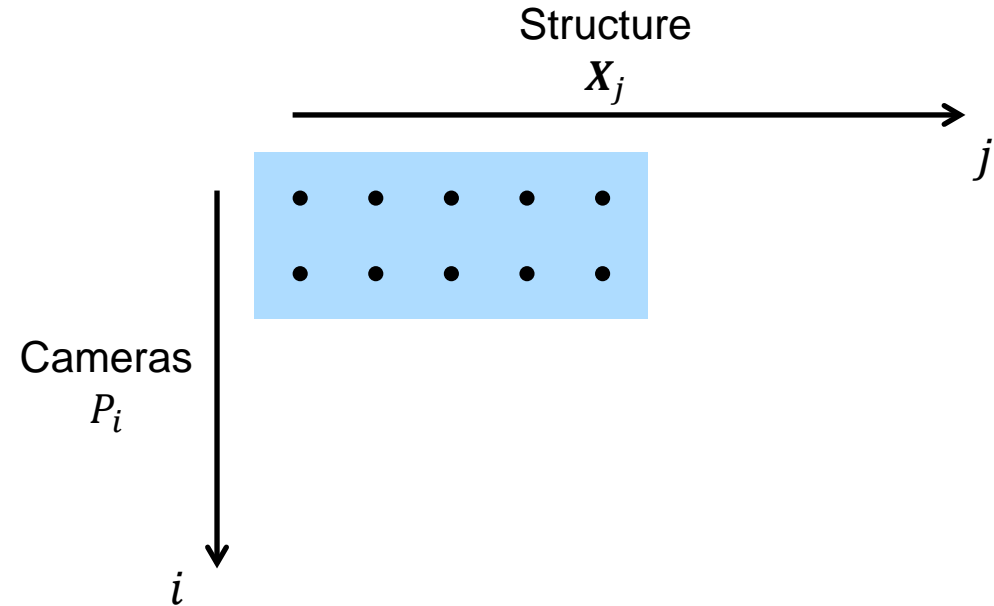
## Problem

Given  $m$  images of  $n$  fixed 3D points, estimate the  $m$  projection matrices  $P_j$  and the  $n$  points  $X_j$  from the  $m \cdot n$  correspondences  $u_{ij} \leftrightarrow u_{kj}$

- We can solve for structure and motion when
$$2mn \geq 11m + 3n - 15$$
- In the general/uncalibrated case, cameras and points can only be recovered up to a projective ambiguity ( $\tilde{u}_{ij} = P_i Q^{-1} Q \tilde{X}_j$ )
- In the calibrated case, they can be recovered up to a similarity (scale)
  - Known as Euclidean/metric reconstruction
- This problem has been studied extensively and several different approaches have been suggested
- We will take a look at a couple of these
  - Sequential structure from motion
  - Bundle adjustment

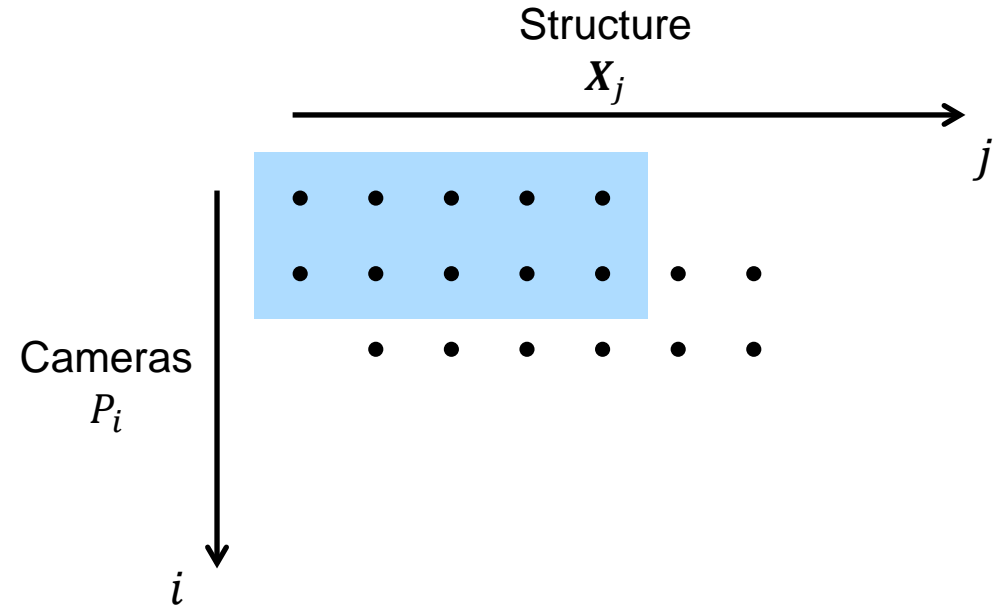
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation



# Sequential SfM

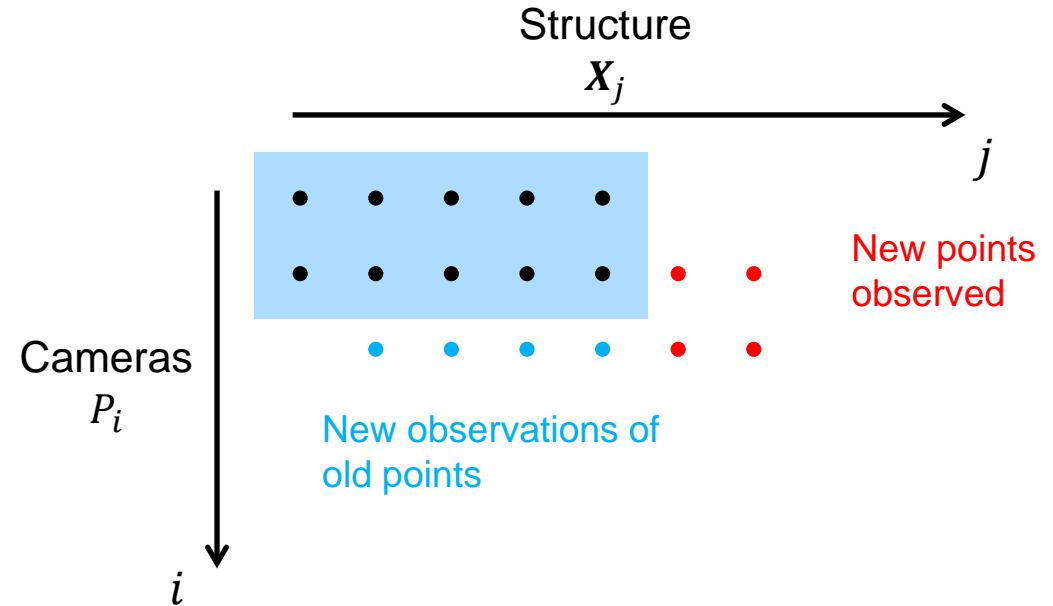
- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation





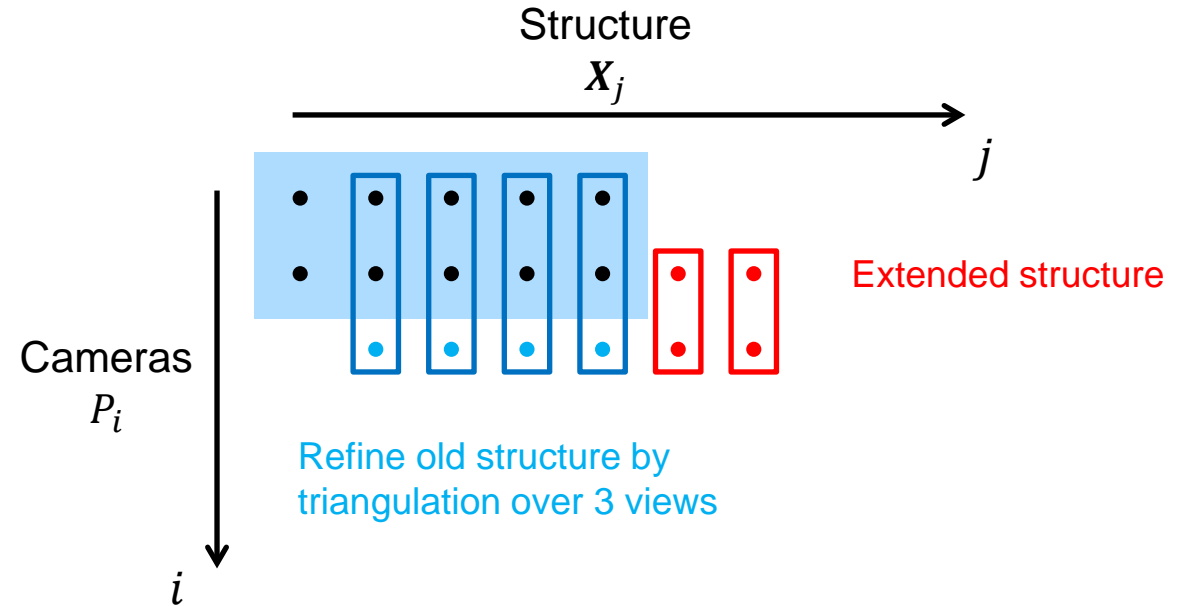
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation



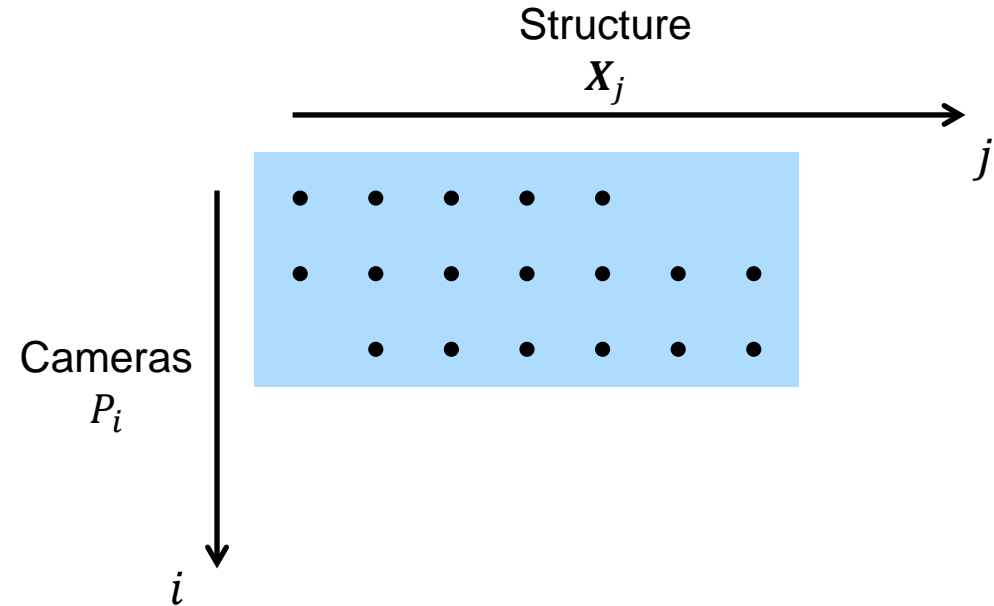
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1\mathbf{t}_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation



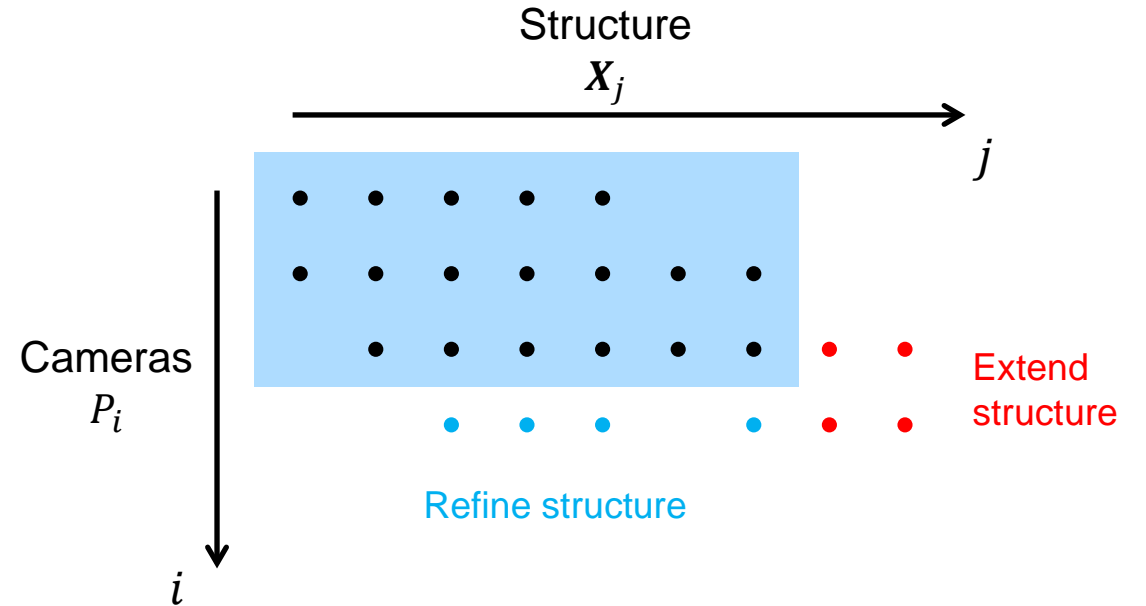
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation



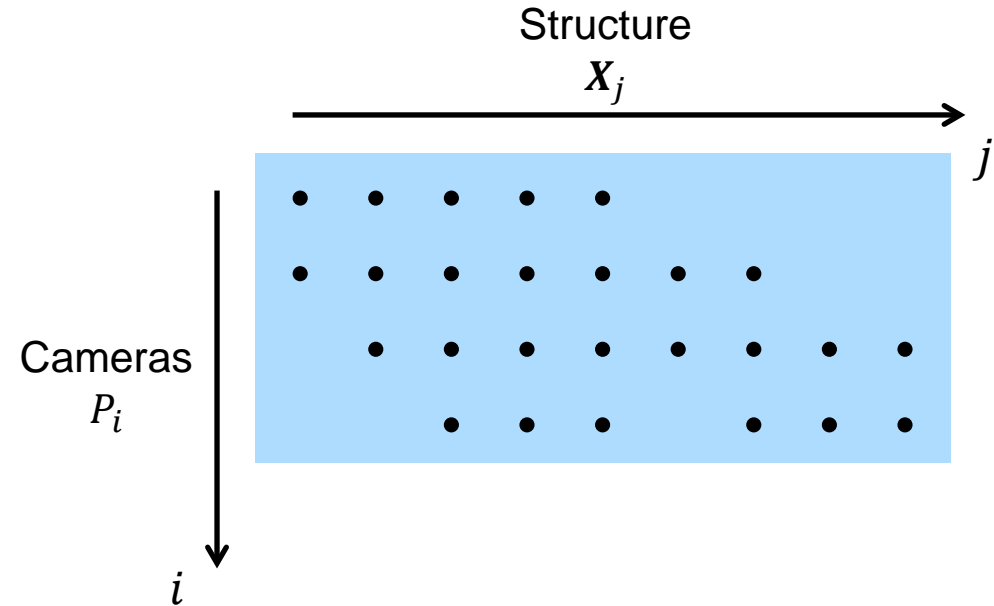
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation



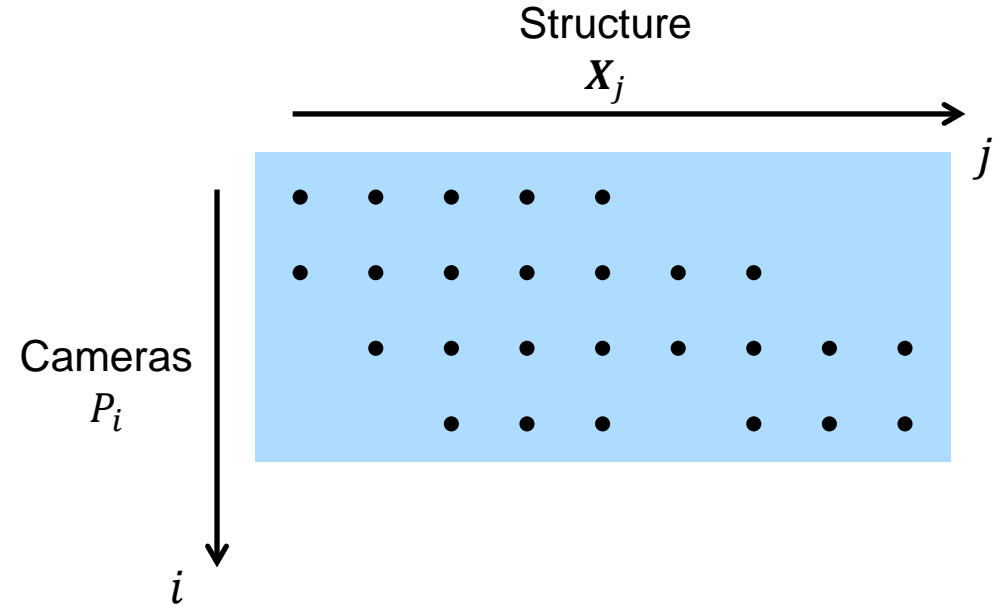
# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation



# Sequential SfM

- Initialize motion from two images
  - $F \rightarrow (P_1, P_2)$
  - $E \rightarrow (P_1, P_2) = (K_1[I \quad \mathbf{0}], K_2[^1R_2 \quad ^1t_2])$
- Initialize the 3D structure by triangulation
- For each additional view
  - Determine the projection matrix  $P_i$ , e.g. from 2D-3D correspondences  $\mathbf{u}_{ij} \leftrightarrow \mathbf{X}_j$
  - Refine and extend the 3D structure by triangulation
- The resulting structure and motion can be refined in a process known as bundle adjustment

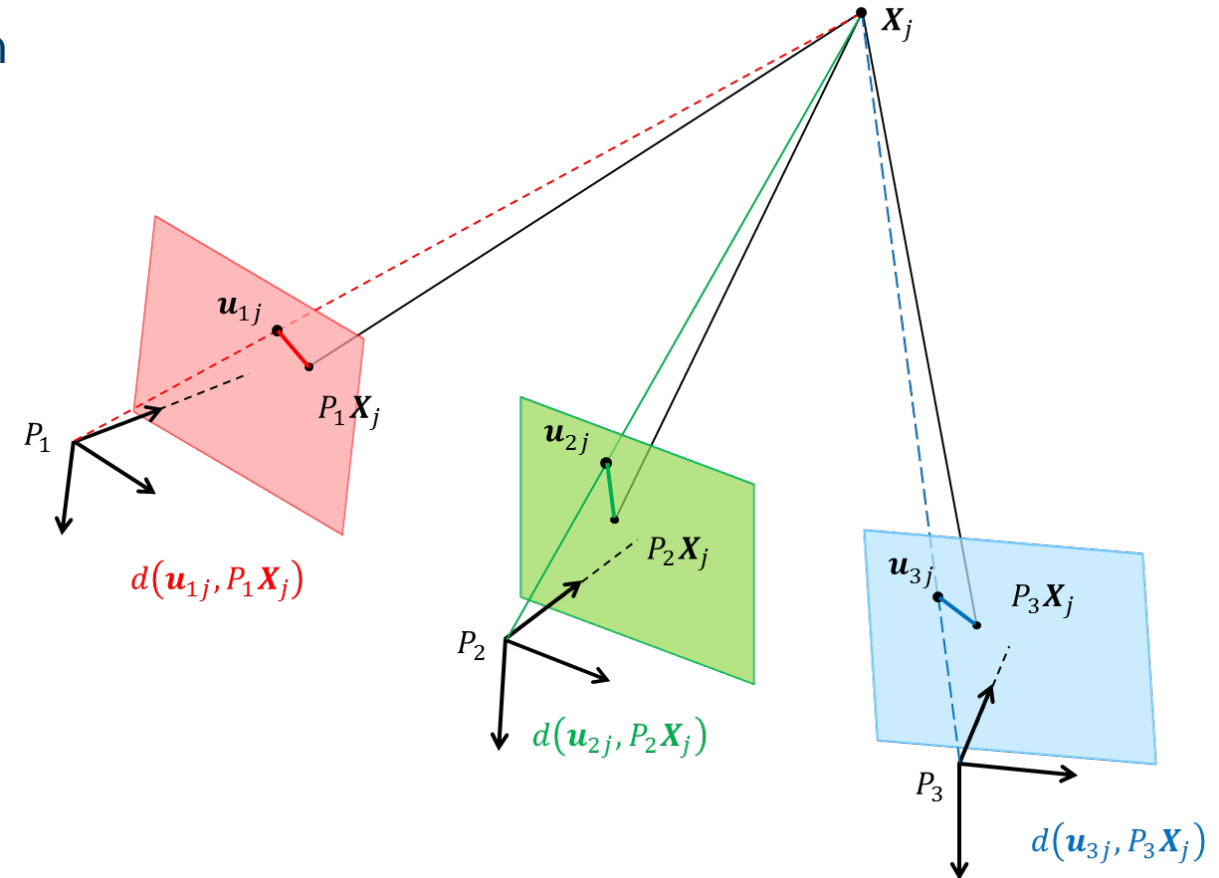


# Bundle adjustment

- Non-linear method that refines structure and motion by minimizing the sum of squared reprojection errors

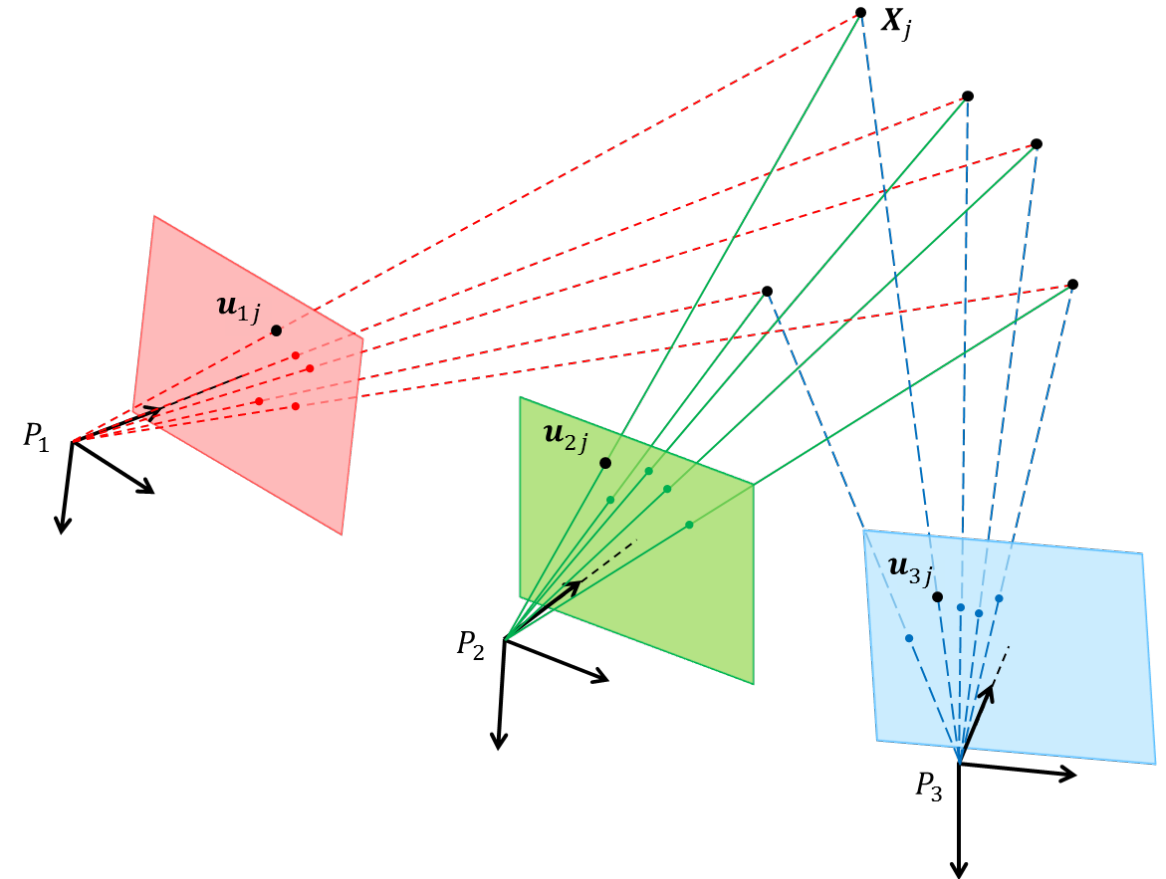
$$\epsilon = \sum_{i=1}^m \sum_{j=1}^n d(\tilde{\mathbf{u}}_{ij}, P_i \tilde{\mathbf{X}}_j)^2$$

- Camera calibration can be solved as part of bundle adjustment by including intrinsic parameters and skew parameters in the cost function
- Need initial estimates for all parameters!
  - 3 per 3D point
  - ~12 per camera depending on parameterization
  - Some intrinsic parameters, like the focal length, can be initialized from image EXIF data



# Bundle adjustment

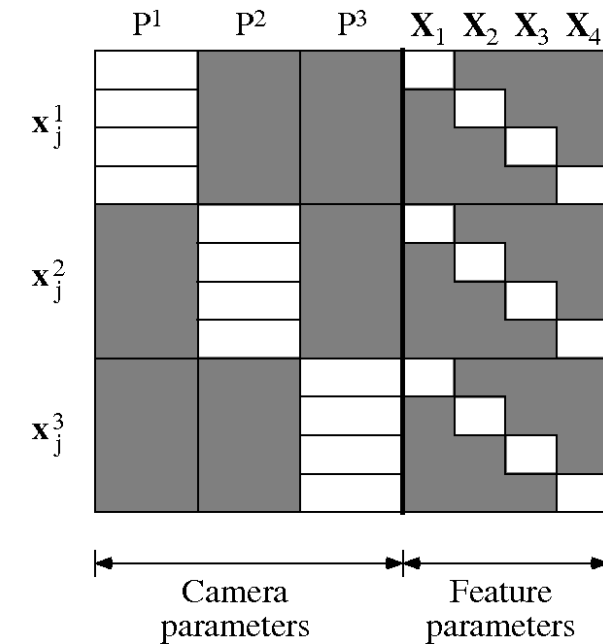
- There are several strategies that deal with the potentially extreme number of parameters
- Reduce the number of parameters by not including all the views and/or all the points
  - Perform bundle adjustment only on a subset and compute missing views/points based on the result
  - Divide views/points into several subsets which are bundle adjusted independently and merge the results
- Interleaved bundle adjustment
  - Alternate minimizing the reprojection error by varying only the cameras or only the points
  - This is viable since each point is estimated independently given fixed cameras, and similarly each camera is estimated independently from fixed points





# Bundle adjustment

- Sparse bundle adjustment
  - For each iteration, iterative minimization methods need to determine a vector  $\Delta$  of changes to be made in the parameter vector
  - In Levenberg-Marquardt each such step is determined from the equation
$$(J^T J + \lambda I) \Delta = -J^T \epsilon$$
where  $J$  is the Jacobian matrix of the cost function and  $\epsilon$  is the vector of errors
  - For the bundle adjustment problem the Jacobian matrix has a sparse structure that can be exploited in computations



The sparse structure of the Jacobian matrix for a bundle adjustment problem with 3 cameras and 4 3D points

# Bundle adjustment

- Sparse bundle adjustment
  - For each iteration, iterative minimization methods need to determine a vector  $\Delta$  of changes to be made in the parameter vector
  - In Levenberg-Marquardt each such step is determined from the equation
$$(J^T J + \lambda I) \Delta = -J^T \epsilon$$
where  $J$  is the Jacobian matrix of the cost function and  $\epsilon$  is the vector of errors
  - For the bundle adjustment problem the Jacobian matrix has a sparse structure that can be exploited in computations
- Combined with parallel processing the before mentioned strategies has made it possible to solve extremely large SfM problems

# Bundle adjustment

- Sparse bundle adjustment
  - For each iteration, iterative minimization methods need to determine a vector  $\Delta$  of changes to be made in the parameter vector
  - In Levenberg-Marquardt each such step is determined from the equation
$$(J^T J + \lambda I) \Delta = -J^T \epsilon$$
where  $J$  is the Jacobian matrix of the cost function and  $\epsilon$  is the vector of errors
  - For the bundle adjustment problem the Jacobian matrix has a sparse structure that can be exploited in computations
- Combined with parallel processing the before mentioned strategies has made it possible to solve extremely large SfM problems
- S. Agarwal et al, *Building Rome in a Day*, 2011
  - Cluster of 62-computers
  - 150 000 unorganized images from Rome
  - ~37 000 image registered
  - Total processing time ~21 hours
  - SfM time ~7 hours
- J. Heinly et al, *Reconstructing the World in Six Days*, 2015
  - 1 dual processor PC with 5 GPU's (CUDA)
  - ~96 000 000 unordered images spanning the globe
  - ~1.5 000 000 images registered
  - Total processing time ~5 days
  - SfM time ~17 hours

# Bundle adjustment

- SBA – Sparse Bundle Adjustment
  - A generic sparse bundle adjustment C/C++ package based on the Levenberg-Marquardt algorithm
  - Code (C and Matlab mex) available at <http://www.ics.forth.gr/~lourakis/sba/>
  - CVSBA is an OpenCV wrapper for SBA [www.uco.es/investiga/grupos/ava/node/39/](http://www.uco.es/investiga/grupos/ava/node/39/)
- Ceres
  - By Google (used in production since 2010)
  - A C++ library for modeling and solving large, complicated optimization problems like SfM
  - Homepage: [www.ceres-solver.org](http://www.ceres-solver.org)
  - Code available on GitHub <https://github.com/ceres-solver/ceres-solver>
- GTSAM – Georgia Tech Smoothing and Mapping
  - A C++ library based on factor graphs that is well suited for SfM ++
  - Code (C++ library and Matlab toolbox) available at <https://borg.cc.gatech.edu/borg/download>
- g<sup>2</sup>o – General Graph Optimization
  - Open source C++ framework for optimizing graph-based nonlinear error functions
  - Homepage: <https://openslam.org/g2o.html>
  - Code available on GitHub <https://github.com/RainerKuemmerle/g2o>

# Bundle adjustment

- Bundler
  - A structure from motion system for unordered image collections written in C and C++
  - SfM based on a modified version SBA (default) or Ceres
  - Homepage: <http://www.cs.cornell.edu/~snaveley/bundler/>
  - Code available on GitHub [https://github.com/snaveley/bundler\\_sfm](https://github.com/snaveley/bundler_sfm)
- VisualSfM
  - A GUI application for 3D reconstruction using structure from motion
  - Output works with other tools that performs dense 3D reconstruction
  - Homepage: <http://ccwu.me/vsfm/>
- RealityCapture
  - A state-of-the-art photogrammetry software that automatically extracts accurate 3D models from images, laser-scans and other input
  - Homepage: <https://www.capturingreality.com/>

# Example

## Holmenkollen 2-view SfM

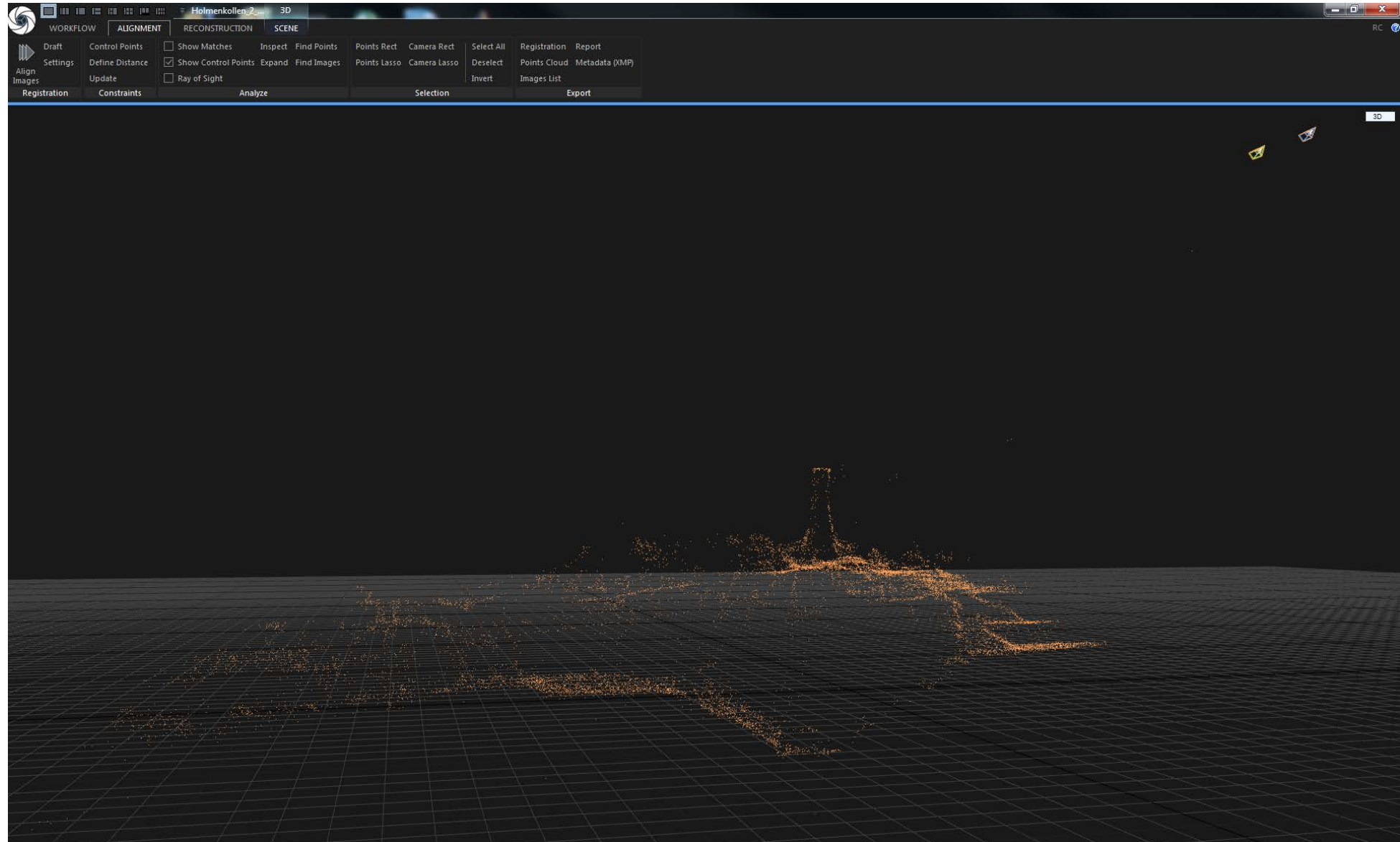
More than 10 000 points are registered in addition to the 2 cameras

The screenshot displays the Agisoft Metashape software interface for a 2-view SfM project. The top menu bar includes options like Workflow, Alignment, Reconstruction, and Scene. The left sidebar shows the project hierarchy, including Images, Control points, and Component 0. The bottom-left panel shows the 'Selected input' settings, including file name, model, width, height, features, and various calibration options. The main 3D view area shows a point cloud of the Holmenkollen ski jump structure, with two camera views (2D) overlaid. The point cloud is composed of numerous orange points, indicating a dense registration of features from the two input images.



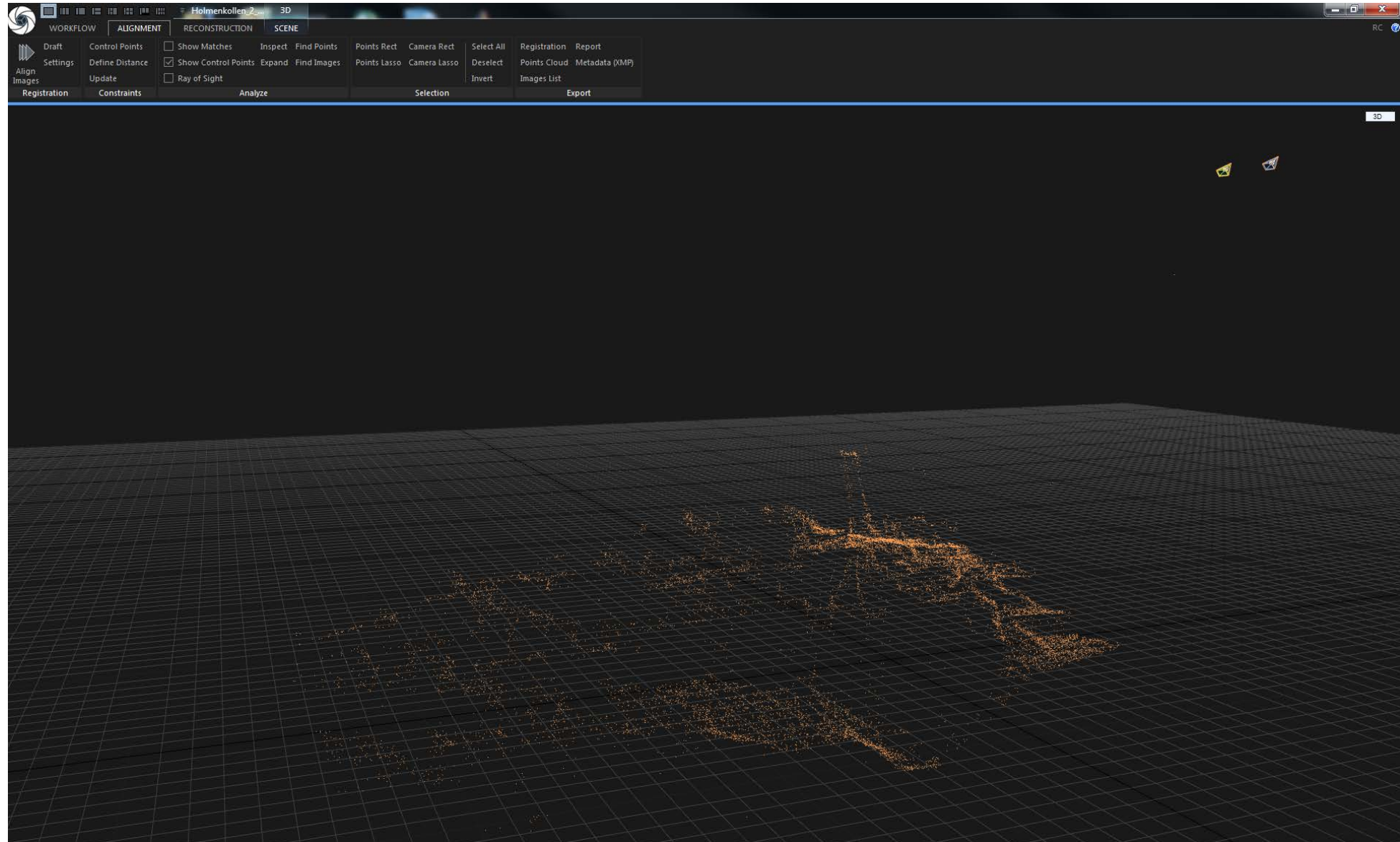
# Example

## Holmenkollen 2-view SfM



# Example

## Holmenkollen 2-view SfM





# Example

## Holmenkollen 3-view SfM

More than 16 000 points are registered in addition to the 3 cameras

**Workflow** | **Alignment** | **Reconstruction** | **Scene**

**Tools:** Draft, Settings, Define Distance, Control Points, Show Matches, Find Points, Points Rect, Camera Rect, Select All, Registration, Report, Align Images, Update, Show Control Points, Expand, Find Images, Points Lasso, Camera Lasso, Deselect, Points Cloud, Metadata (XMP), Images List, Export

**Registration** | **Constraints** | **Analyze** | **Selection** | **Export**

**Images:**

- \_MG\_8045.JPG | f: 48.7mm #pts: 40000
- \_MG\_8046.JPG | f: 48.7mm #pts: 40000
- \_MG\_8047.JPG | f: 48.7mm #pts: 40000

**Control points:** empty

**Component 0:** 3/3 cams, 0 models

**Selected input(s):**

File name	< Different values >
Model	Canon Canon EOS 5D Mark II
Width	5616
Height	3744
Features	40000
Features source	Use all image features
Enable alignment	True
Enable meshing	True
Enable texturing and coloring	True
Weight in texturing	1.000000
Downscale for depth-map	1
Registered	True
Enable in component	True
Lock pose for continue	False
<b>Prior pose</b>	
Locked pose group	
Absolute pose	Unknown
<b>Prior calibration</b>	
Calibration group	-1
Prior	Approximate
Focal length (35mm)	48.678719
Principal point x	0.000000
Principal point y	0.000000
Skew	0.000000
Aspect ratio	1.000000
<b>Prior lens distortion</b>	
Lens group	-1
Prior	Approximate
Model	No lens distortion
<b>Registration calibration</b>	
Focal length (35mm)	< Different values >

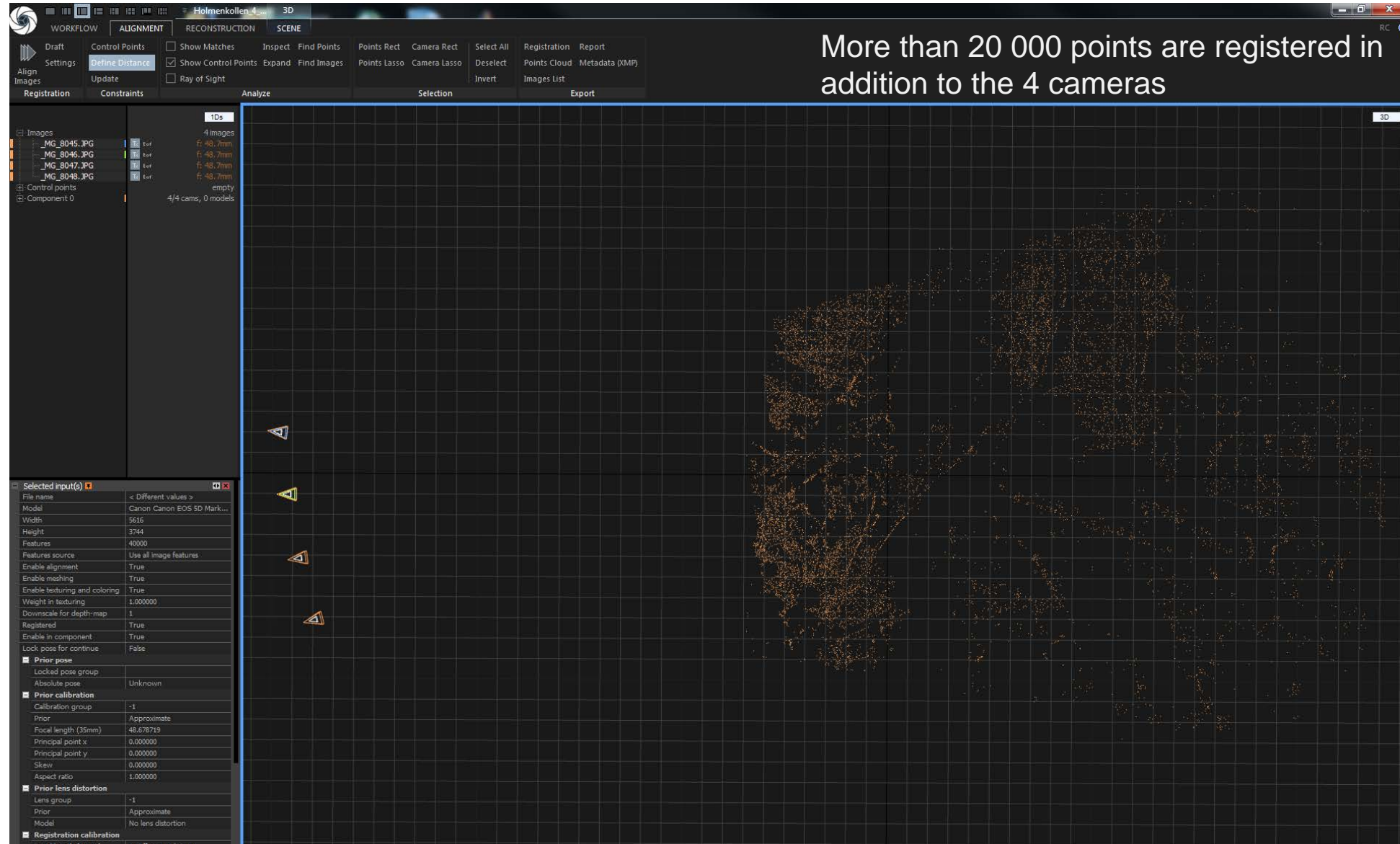
**2D Views:**

- Top-left: Aerial view of the ski jump structure.
- Top-right: Aerial view of the ski jump structure.
- Bottom-left: Aerial view of the ski jump structure.

**3D View:** Point cloud of the ski jump structure.

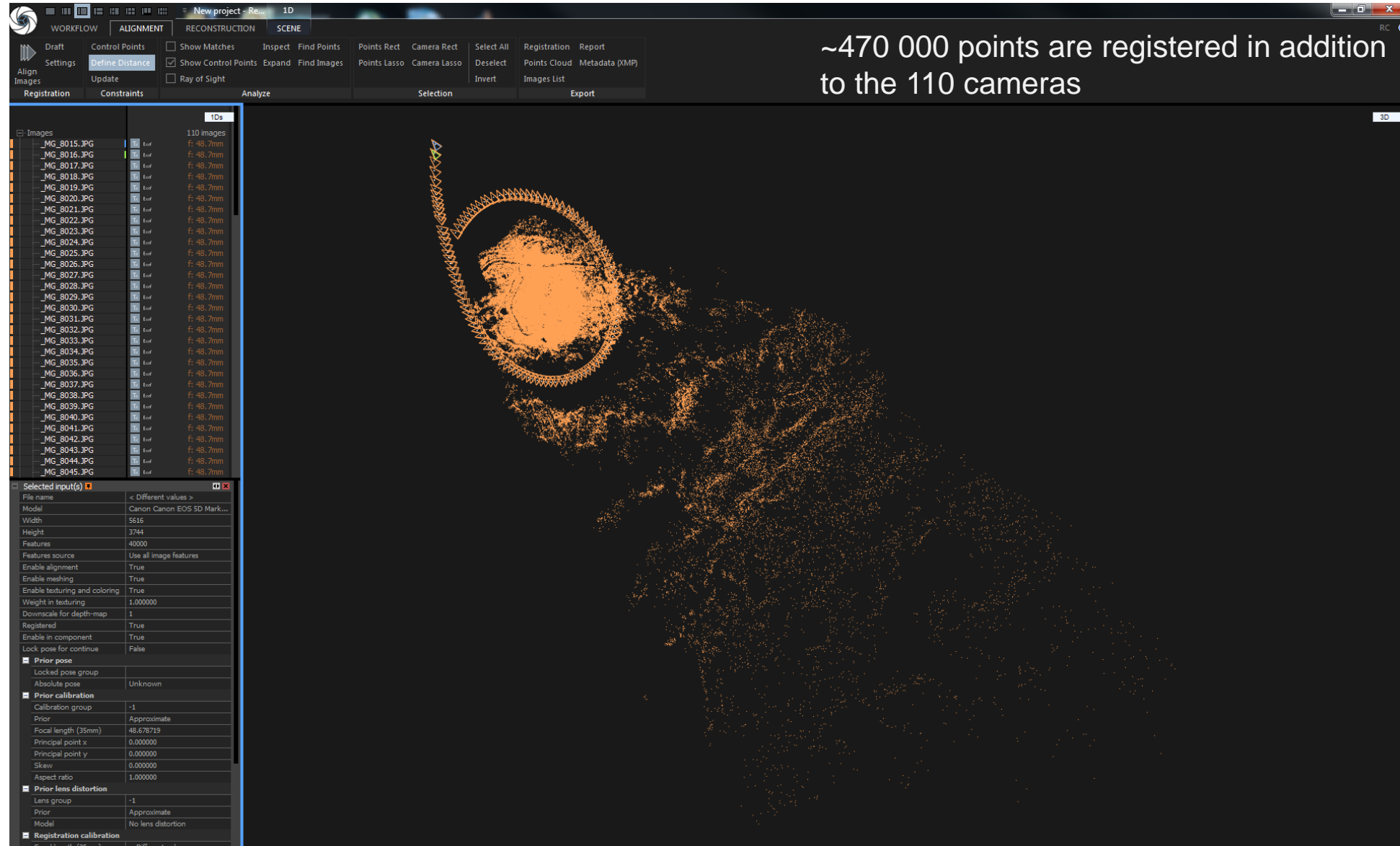
# Example

## Holmenkollen 4-view SfM



# Example

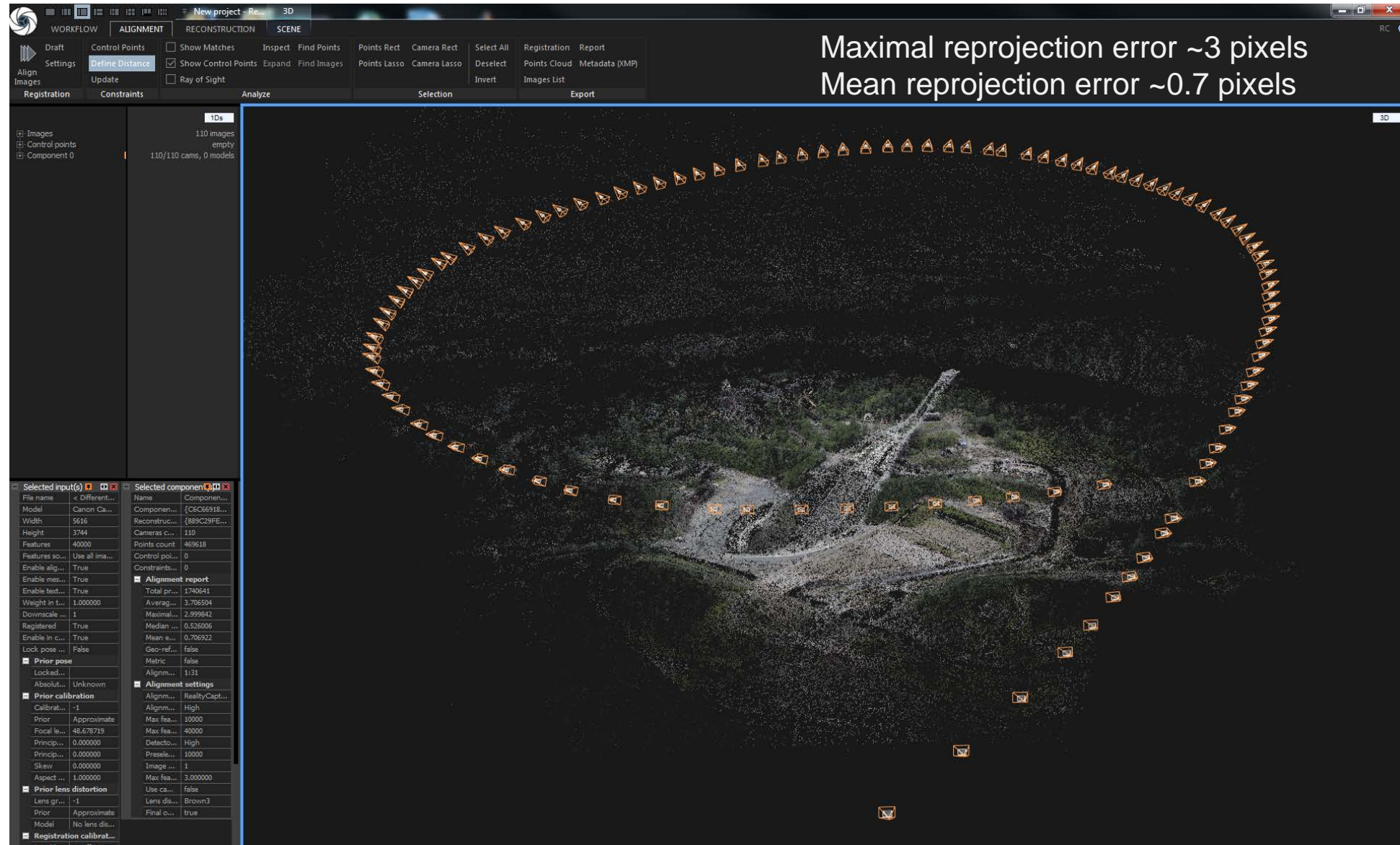
## Holmenkollen 110-view SfM





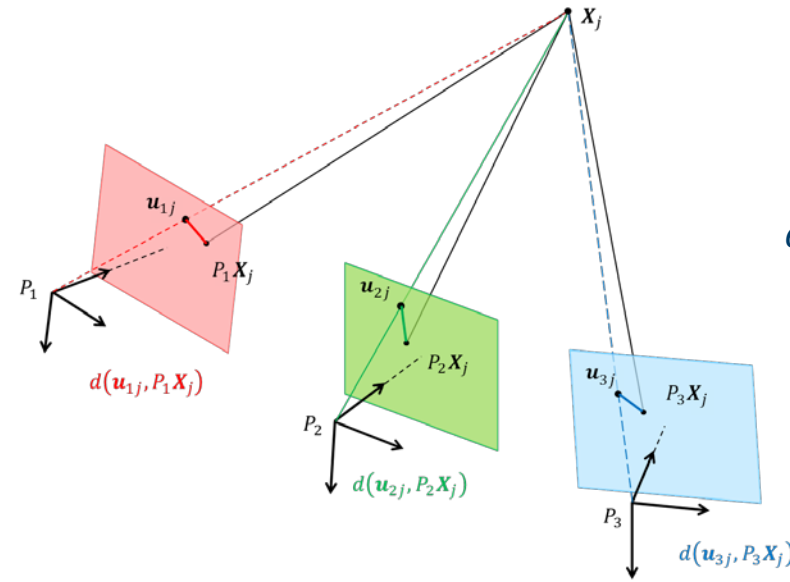
# Example

## Holmenkollen 110-view SfM



# Summary

- Structure from motion
  - Sequential SfM
  - Bundle adjustment
- Additional reading:
  - Szeliski: 7.3-7.5
- Optional reading:
  - Snavely N. Seitz S. M., Szeliski R., *Modeling the World from Internet Photo Collections*, 2007
  - S. Agarwal et al, *Building Rome in a Day*, 2011
  - J. Heinly et al, *Reconstructing the World in Six Days*, 2015



$$\epsilon = \sum_{i=1}^m \sum_{j=1}^n d(\tilde{u}_{ij}, P_i \tilde{X}_j)^2$$

