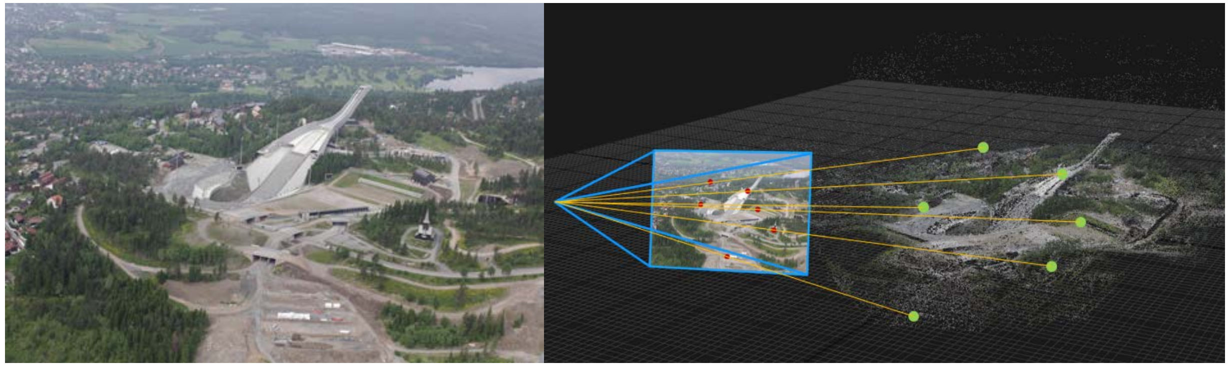
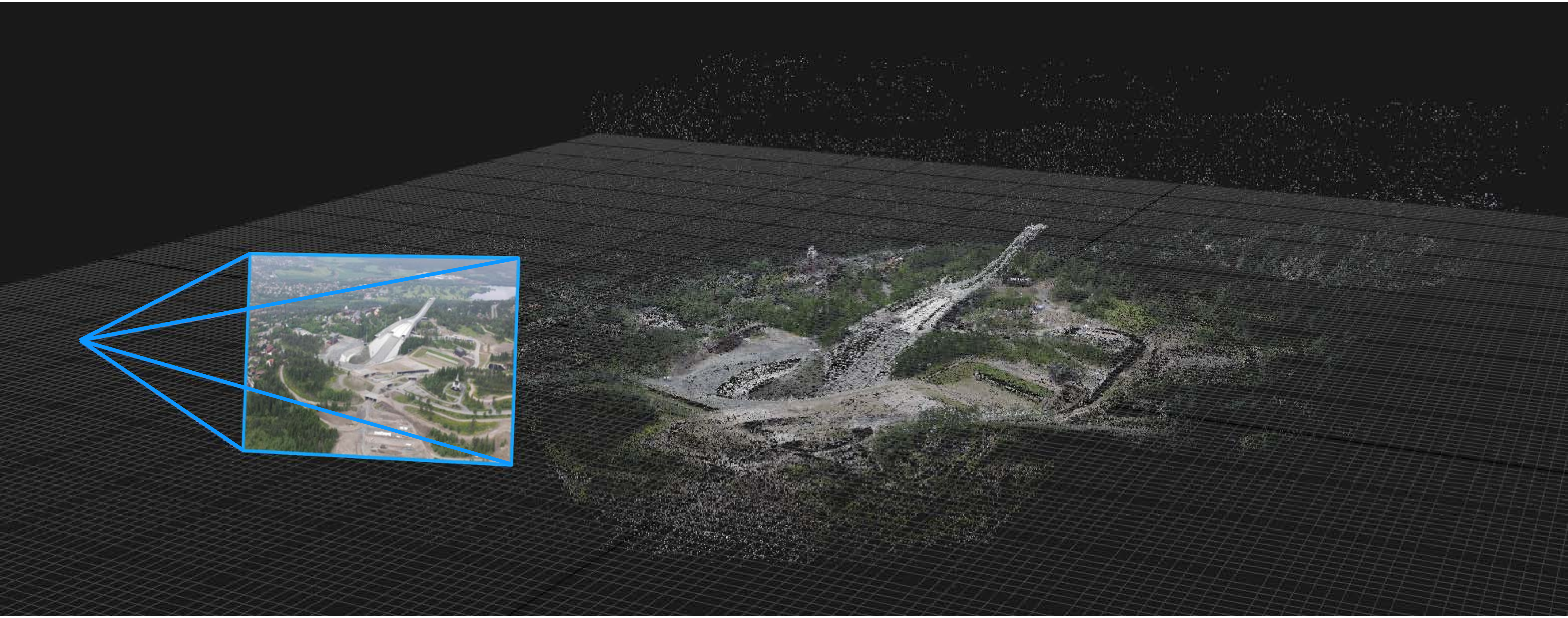
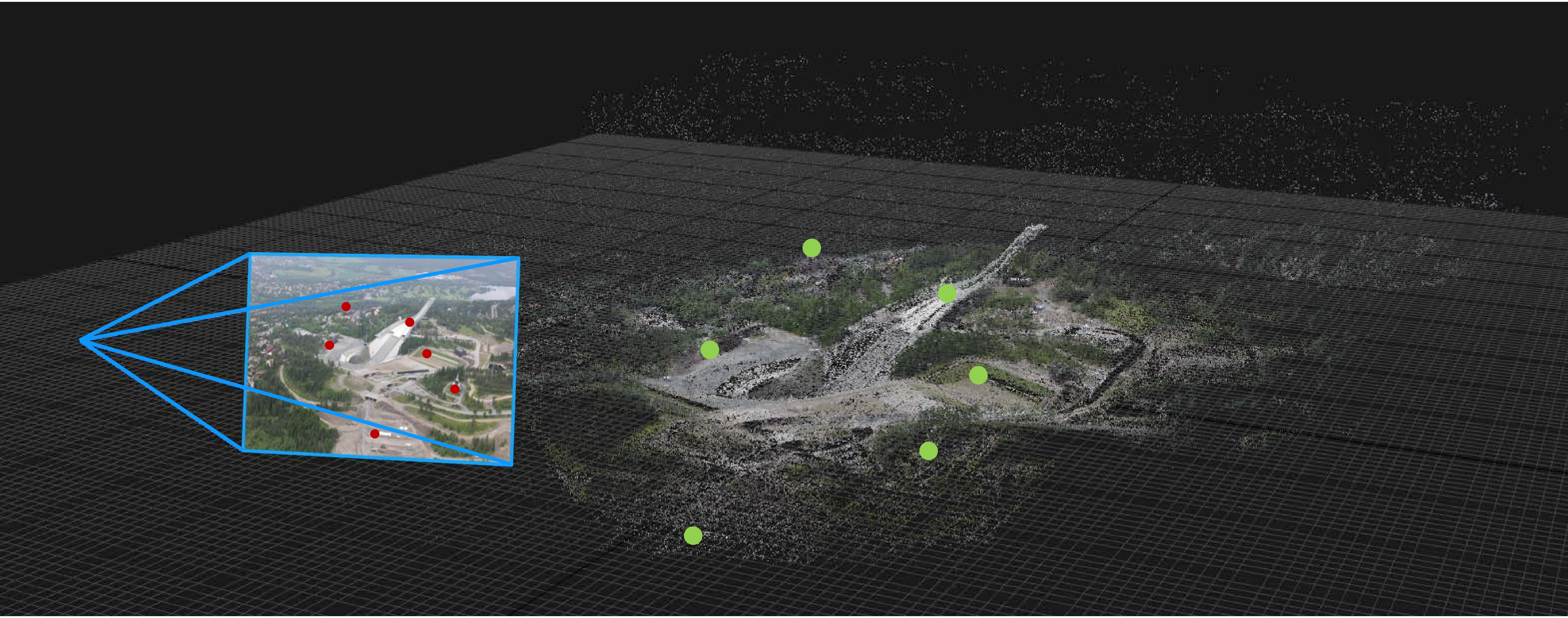
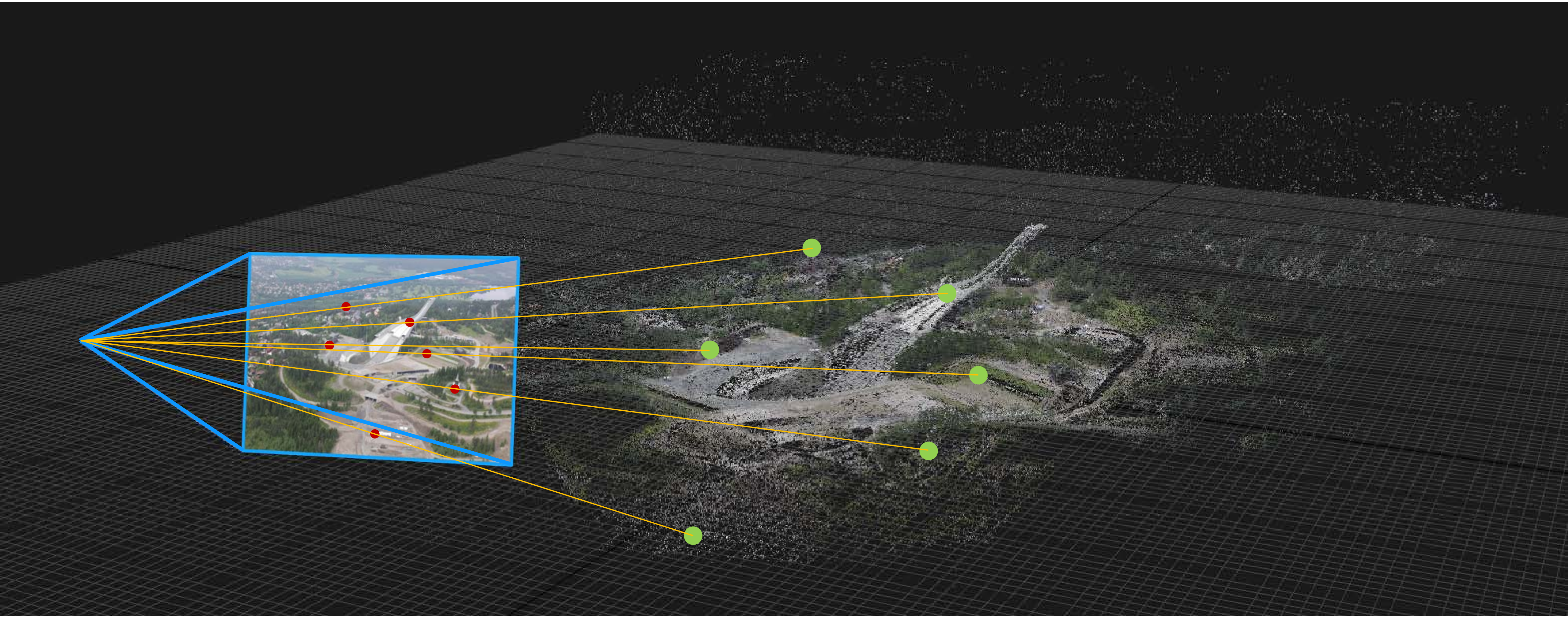


Lecture 5.2
Pose from known 3D points

Trym Vegard Haavardsholm







World geometry from correspondences

	Structure (scene geometry)	Motion (camera geometry)	Measurements
Pose estimation	Known	Estimate	3D to 2D correspondences
Triangulation, Stereo	Estimate	Known	2D to 2D correspondences
Reconstruction, Structure from Motion	Estimate	Estimate	2D to 2D correspondences

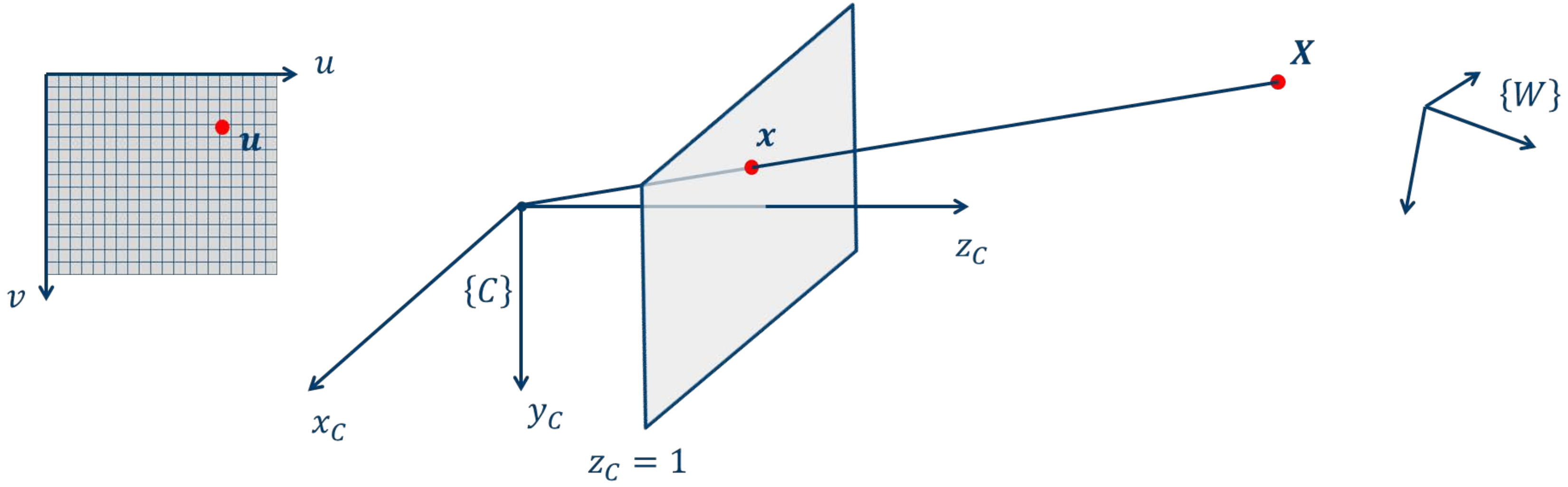
Pose estimation from known 3D points

Given at least 6 3D to 2D correspondences

$$\left\{ \tilde{\boldsymbol{X}}_i, \tilde{\boldsymbol{u}}_i \right\}$$

we can estimate a camera matrix $\hat{P}$ that fits with the camera projection model

$$\tilde{\boldsymbol{u}}_i = P\tilde{\boldsymbol{X}}_i$$



where P is given by

$$P = K[R \mid \boldsymbol{t}]$$

Pose from *P*

- Decomposition of the Camera Matrix

$$
\begin{aligned}
P &= K[R \mid \mathbf{t}] \\
&= K[R \mid -R\mathbf{C}] \\
&= [M \mid -M\mathbf{C}]
\end{aligned}
$$

Pose from *P*

- Decomposition of the Camera Matrix

$$
\begin{aligned}
P &= K[R \mid \mathbf{t}] \\
&= K[R \mid -R\mathbf{C}] \\
&= [M \mid -M\mathbf{C}]
\end{aligned}
$$

Find the camera center ***C***

Find intrinsic *K* and rotation *R*

Pose from *P*

- Decomposition of the Camera Matrix

$$P = K[R \mid \mathbf{t}]$$

$$= K[R \mid -R\mathbf{C}]$$

$$= [M \mid -M\mathbf{C}]$$

Find the camera center ***C***

$$P\mathbf{C} = 0$$

SVD of *P*!
C is the right singular vector corresponding to the smallest singular value

Find intrinsic *K* and rotation *R*

Pose from *P*

- Decomposition of the Camera Matrix

$$P = K[R \mid \mathbf{t}]$$
$$= K[R \mid -R\mathbf{C}]$$
$$= [M \mid -M\mathbf{C}]$$

Find the camera center ***C***

$$P\mathbf{C} = 0$$

SVD of *P*!
C is the right singular vector
corresponding to the smallest singular value

Find intrinsic *K* and rotation *R*

$$M = KR$$

RQ decomposition

RQ decomposition

- RQ-factorization is a decomposition of a matrix M into the product $M = RQ$ where R is an upper triangular matrix and Q is an orthogonal matrix

- The RQ in ''RQ-decomposition'' corresponds to KR in the expression $M = KR$,
  Upper triangular matrix R ←→ Camera calibration matrix K
  Orthogonal matrix Q ←→ Rotation matrix R

- In addition we know:
  - *K* has positive diagonal: $(KD)(D^{-1}R)$ where
  - det(R) = 1: Enforce det(M) > 0

RQ decomposition

- When only QR decomposition is available:

$$M = \left(Q\,R\right)^{-1} = R^{-1}\,Q^{-1}$$

- In matlab:

```
function [R,Q] = rq(M)
[Q,R] = qr(rot90(M,3));
R = rot90(R,2)';
Q = rot90(Q);
```

Pose estimation from known 3D points

1. Extract local features
2. Establish 3D to 2D correspondences $\{\tilde{\boldsymbol{X}}_i, \tilde{\boldsymbol{u}}_i\}$
3. Estimate P from $\{\tilde{\boldsymbol{X}}_i, \tilde{\boldsymbol{u}}_i\}$
4. Let M be the first 3×3 matrix of P
5. Normalize P by multiplying all elements with sign(det(M))
6. Compute the camera center ***C*** by SVD
7. QR-decomposition of M gives us $M = \hat{K}\hat{R}$
   where $\hat{K}$ is an upper triangular like K and $\hat{R}$ is an orthogonal matrix like R
8. Define $D = diag(sign(K_{11}), sign(K_{22}), sign(K_{33}))$
9. Then $K = \hat{K}D$ and $R = D\hat{R}$ and $\boldsymbol{t} = -R\boldsymbol{C}$

Matlab example

```matlab
% Estimate P by basic DLT.
est.P = estimateCameraMatrix_DLT(u,X);
est.P = est.P * sign(det(est.P(1:3, 1:3)));

% Estimate C.
[U,S,V] = svd(est.P);
est.C = V(1:3,end) / V(end, end);

% Estimating K and R by RQ decomposition.
[est.K,est.R] = rq(est.P(1:3,1:3));

% Enforce positive diagonal of K by changing signs in both est.K and est.R.
D = diag(sign(diag(est.K)));
est.K = est.K*D;
est.R = D*est.R;
est.K = est.K/est.K(end,end);

%Determine t from the estimated C.
est.t = -est.R*est.C;
```

Real world example

- 6 correspondences from map

Real world example

- 6 correspondences from map

- Position error: ~100m

- Estimated focal length: 52 mm
- True focal length: 50 mm

What if we know parts of *P*?

- Iterative methods
  - Initialize with P from DLT, clamp known parameters
  - Minimize geometric error

$$\sum_i d(\boldsymbol{u}_i, P\boldsymbol{X}_i)^2$$

What if we know parts of *P*?

- Iterative methods
  - Initialize with P from DLT, clamp known parameters
  - Minimize geometric error

$$\sum_i d(\boldsymbol{u}_i, P\boldsymbol{X}_i)^2$$

  - Better: Soft constraints, for example:

$$\sum_i d(\boldsymbol{u}_i, P\boldsymbol{X}_i)^2 + ws^2 + w(f_x - f_y)^2$$

What if we know parts of *P*?

- Iterative methods
  - Initialize with P from DLT, clamp known parameters
  - Minimize geometric error

$$\sum_i d(\boldsymbol{u}_i, P\boldsymbol{X}_i)^2$$

  - Better: Soft constraints, for example:

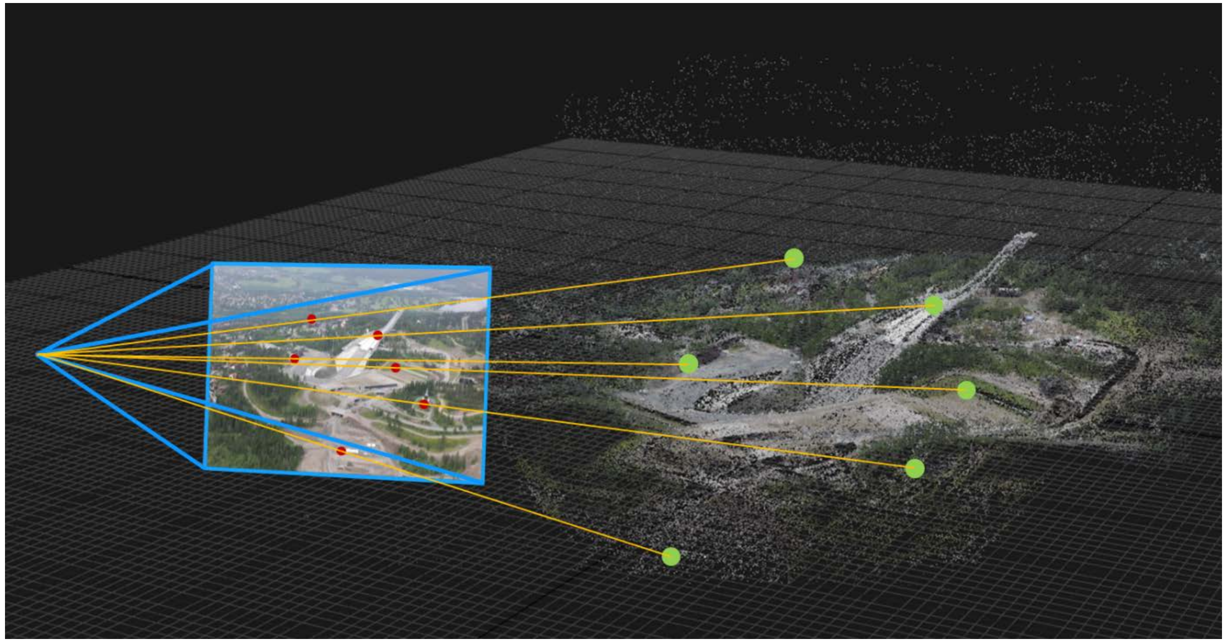
$$\sum_i d(\boldsymbol{u}_i, P\boldsymbol{X}_i)^2 + ws^2 + w(f_x - f_y)^2$$

- Use Perspective-*n*-Point (P*n*P)

n-Point Pose Problem (P*n*P)

- Several different methods available
  - Typically fast non-iterative methods
  - Minimal in number of points
  - Accuracy comparable to iterative methods

- Examples
  - P3P
    - Estimate pose, known K
  - P4Pf
    - Estimate pose and focal length
  - P6P
    - DLT!
  - R6P
    - Estimate pose with rolling shutter



Summary

- Pose from known 3D points
  - Decomposition of *P* matrix
  - P*n*P

- Further reading:
  - Torstein Sattler,
    CVPR 2015 Tutorial on Large-Scale Visual Place Recognition and Image-Based Localization