DTU

Daniel Olesen, DTU Space

30540 – Photogrammetry (5)

Todays Lecture

- Recap on previous lectures (Camera Model, RO/AO and Triangulation)
- Bundle Adjustment
- Aero-triangulation
  - Direct and indirect geo-refererencing
- Orthoimage (backward projection)

Pinhole Camera Model

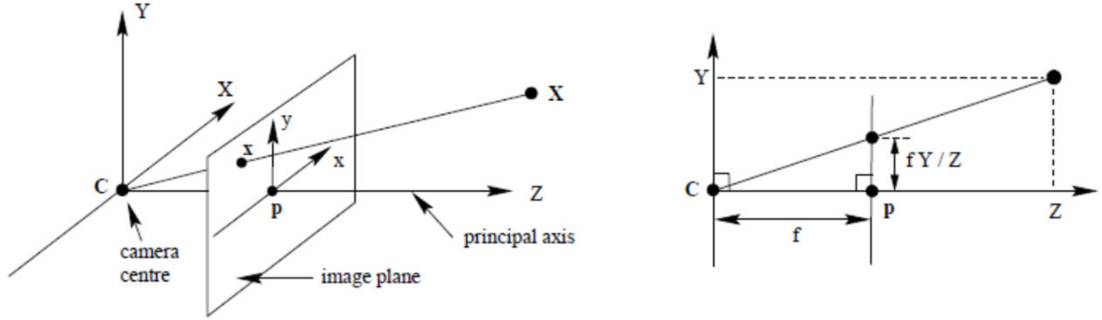


Fig. 6.1. **Pinhole camera geometry.** C *is the camera centre and* p *the principal point. The camera centre is here placed at the coordinate origin. Note the image plane is placed in front of the camera centre.*

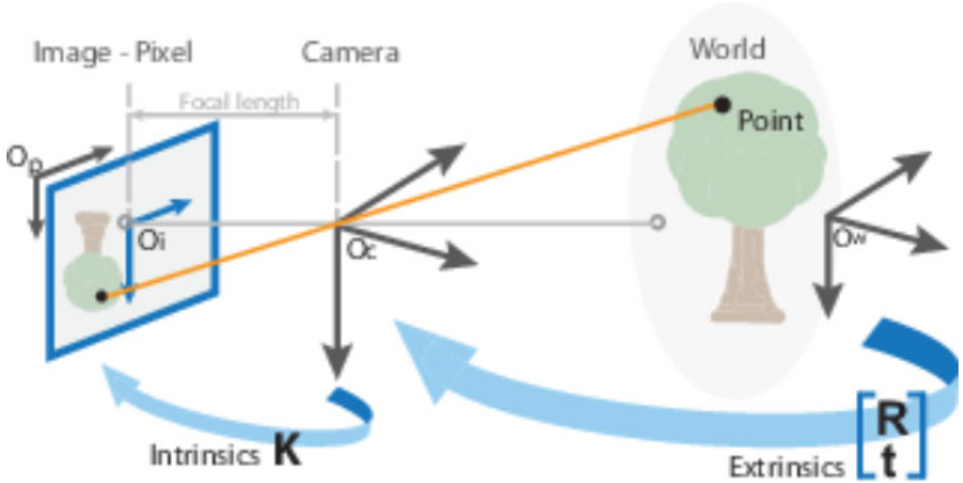
$$\boxed{(X, Y, Z)^{\mathsf{T}} \mapsto (fX/Z, fY/Z)^{\mathsf{T}}}$$ → Cartesian coordinates

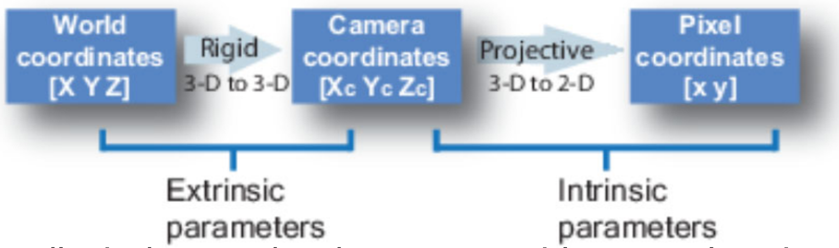
$$\boxed{\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}$$ → Homogeneous coordinates

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

Source: Hartley & Zisserman: Multiple View Geometry

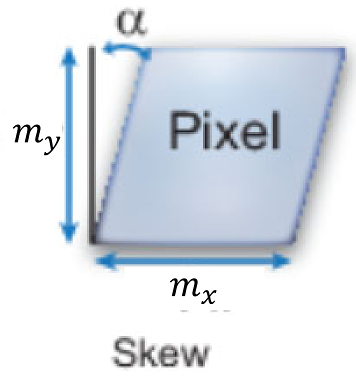
Extrinsic and intrinsic parameters





Source: https://se.mathworks.com/help/vision/ug/camera-calibration.html

intrinsic (calibration) parameters



Mathworks Inc

$$K = \begin{bmatrix} f \cdot m_x & s & p_x \\ 0 & f \cdot m_y & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

f	Focal-length in pixels
m_x, m_y	Scaling-factor in x- and y-directions
p_x, p_y	Principal-point offset in pixels
s	Skew-parameter $s = f \cdot m_y \cdot \tan(\alpha)$

Complete camera-model

$\mathbf{x} = \mathbf{P}\mathbf{X}$

Intrinisic (or calibration parameters)

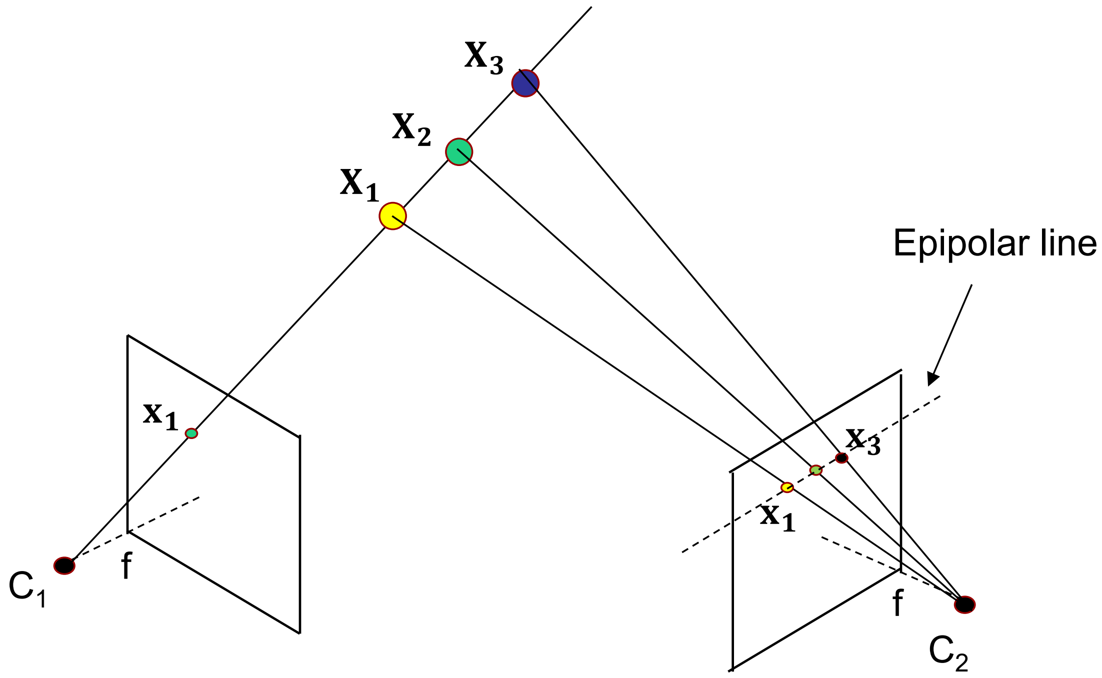
$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix}_{imag} = K[\mathbf{R} \quad \mathbf{t}] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{world}$$

Extrinsics (rotation + translation)

There is a total of 11 parameters in this model (5 intrinsics + 6 extrinsics

Pixel coordinates often expressed as (u,v) = $\{\frac{\lambda x}{\lambda}, \frac{\lambda y}{\lambda}\}$

3D reconstruction using 2 cameras (stereo)



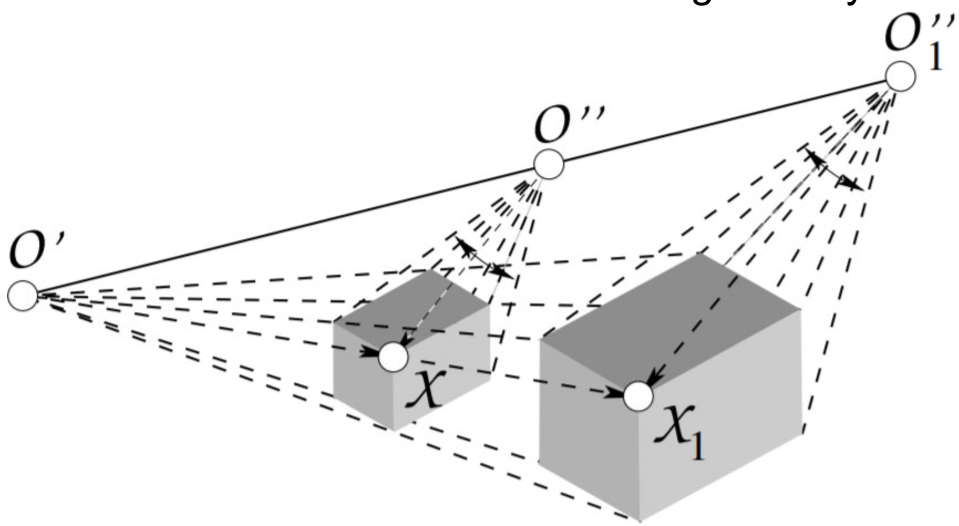
In case we have two cameras with known position and orientation (pose), we can from the second view distinguish between X1, X2 and X3

X1, X2 and X3 all lies on what is known as a epipolar line in the imageplane in the second view. We will later on establish a geometrical constraint between this line and the point in the left view.

Using two cameras to view the same scene is called stereo vision

Relative Orientation

- Suppose we only have two views of a scene (with calibrated cameras and unknown poses) and are given a minimum of 8 point correspondences what are we then able to determine about the two views and the geometry of the object?



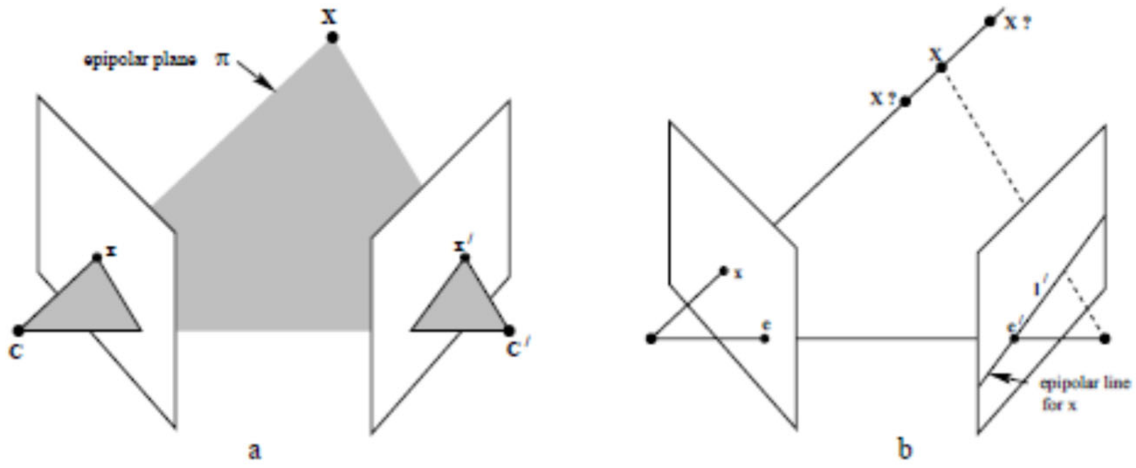
Essential Matrix (Relative Orientation)

- The essential matrix ”encodes” the relative translation and orientation between 2 camera poses – but only up to a scale ambiguity
  - The matrix has a 5 degrees of freedoms corresponding to 2 translations and 3 rotations

- Is usually solved by point-correspondences in both views
  - Longuet-Higgins (1981): 8-point algorithm
  - Nistér (2003): 5-point algorithm

- Applies only to the calibrated case where intrinsics are known!
  - An similar case for uncalibrated cameras is known as the **Fundamental** Matrix.

Epipolar lines



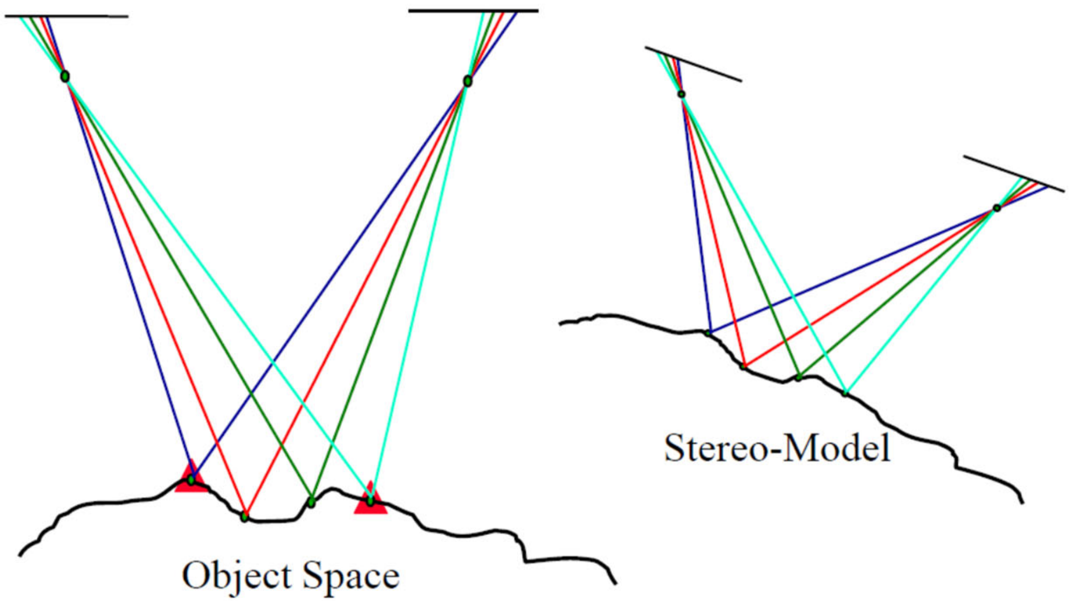
If the Essential Matrix is known, it can restrict search of a point correspondence to one line in the second image.

Epipolar lines:

$$l' = E^T x$$

$$l = Ex'$$

Absolute Orientation



The RO model uses the camera-coordinate system of the first camera as reference

In order to relate the model to the physical world, the exact location of this must be determined in object-space (world) coordinates and the scale factor must be resolved!

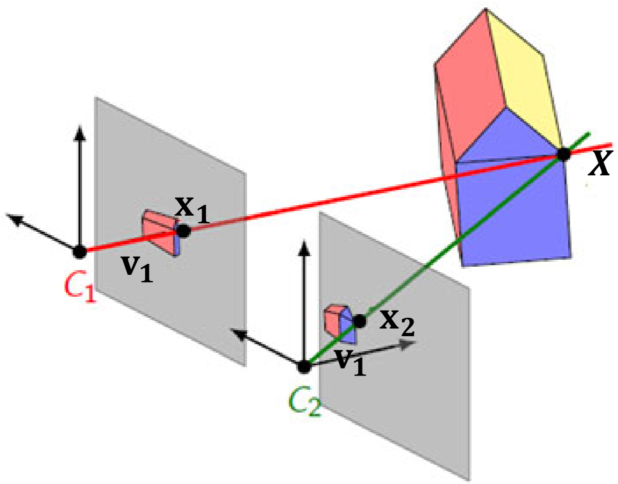
Exterior Orientation of Stereo-pair

- Fully described by 12 parameters
  - $X_l, Y_l, Z_l, \omega_l, \phi_l, \kappa_l$
  - $X_r, Y_r, Z_r, \omega_r, \phi_r, \kappa_r$

- Can be decomposed into
  - Relative Orientation (5 params)
    - 2 translations + 3 rotations
  - Absolute Orientation (7 params)
    - 3 rotations + 3 translations + 1 scale-factor

Triangulation (midpoint algorithm)

- If the intrinsics and extrinsics of both cameras are known, an object point can be found from the intersection point of the rays from each camera!



$$\mathbf{X} = \mathbf{C_1} + (\mathbf{x_1} - \mathbf{C_1})\alpha_1 = \mathbf{C_1} + \mathbf{v_1}\alpha_1$$

$$\mathbf{X} = \mathbf{C_2} + (\mathbf{x_2} - \mathbf{C_2})\alpha_2 = \mathbf{C_2} + \mathbf{v_2}\alpha_2$$

$$\downarrow$$

$$\begin{bmatrix} \mathbf{I_3} & -\mathbf{v_1} & \mathbf{0} \\ \mathbf{I_3} & \mathbf{0} & -\mathbf{v_2} \end{bmatrix} \begin{bmatrix} \mathbf{X} \\ \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} \mathbf{C_1} \\ \mathbf{C_2} \end{bmatrix} \longrightarrow \begin{bmatrix} \mathbf{X} \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \text{ is solvable from L.S.}$$

Note that all quanties above are expressed as inhomogeneous coordinates in the world-coordinate system

Triangulation (midpoint algorithm)

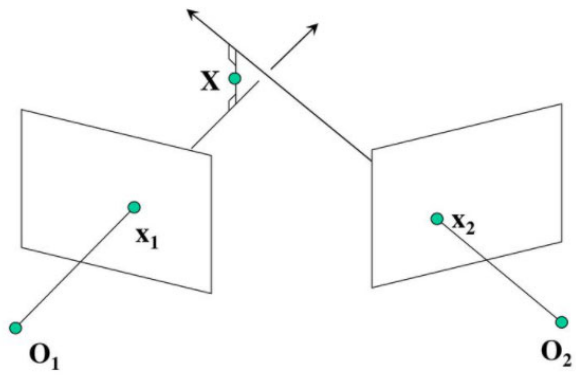
Due to a limited resolution in each camera (finite number of pixels), the back-projected rays will generally not intersect.

The midpoint-algorithm will however return the midpoint from the shortest possible line between the two rays…

This is equivalent to minimize:

$$\epsilon = d(\mathbf{l_1}, \mathbf{X})^2 + d(\mathbf{l_2}, \mathbf{X})^2$$

Which is the 3D error between the two rays.



Linear Triangulation (minimizing the algebraic error)

- The mid-point method is considered to be poor for practical problems

- If the 3D point is physically much closer to one of the cameras, finding the midpoint is a poor strategy for the actual location of the point

- Another disadvantage is that this method can not naturally be extended to triangulation from multiple views.

- A better algorithm, which aims to minimize the algebraic error is presented in the following. The starting point of this algorithm is the projective equation from the pinhole camera model,

$$\mathbf{x}_i = \mathbf{P}_i\mathbf{X}$$

We can parametrize $\mathbf{P}$ in terms of the row vectors, $\mathbf{P}_\mathrm{i} = \begin{bmatrix} \mathbf{p}_i^1 \\ \mathbf{p}_\mathrm{i}^2 \\ \mathbf{p}_\mathrm{i}^3 \end{bmatrix}$

Linear Triangulation

As we are dealing with homogenous coordinates we have that

$$\mathbf{x}_\mathrm{i} = \lambda_i \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$$\mathbf{x}_\mathrm{i} = \mathbf{P}_\mathbf{i}\mathrm{X} \Rightarrow \begin{bmatrix} \lambda_i x_i \\ \lambda_i y_i \\ \lambda_i \end{bmatrix} = \begin{bmatrix} \mathbf{p}_i^1 \\ \mathbf{p}_i^2 \\ \mathbf{p}_i^3 \end{bmatrix} \mathbf{X}$$

For each row, we can form the equations:

$\lambda_i x_i = \mathbf{p}_i^1 \mathbf{X}$ (1)

$\lambda_i y_i = \mathbf{p}_i^2 \mathbf{X}$ (2)

$\lambda_i = \mathbf{p}_i^3 \mathbf{X}$ (3)

If we divide (1) and (2) with (3) we can eliminate λ_i, i.e.

$$x_i = \frac{\mathbf{p}_i^1 \mathbf{X}}{\mathbf{p}_i^3 \mathbf{X}} \qquad y_i = \frac{\mathbf{p}_i^2 \mathbf{X}}{\mathbf{p}_i^3 \mathbf{X}}$$

$$(\mathbf{p}_i^3 x_i - \mathbf{p}_i^1)\mathbf{X} = 0$$

$$(\mathbf{p}_i^3 y_i - \mathbf{p}_i^2)\mathbf{X} = 0$$

$$\begin{bmatrix} \mathbf{p}_1^3 x_1 - \mathbf{p}_1^1 \\ \mathbf{P}_1^3 y_1 - \mathbf{P}_1^2 \\ \mathbf{P}_2^3 x_2 - \mathbf{P}_2^1 \\ \mathbf{P}_2^3 y_2 - \mathbf{P}_2^2 \end{bmatrix} \mathbf{X} = 0 \implies \mathbf{AX} = 0$$

X is the right null-space of A (can be solved by SVD)

MATLAB:

```
[U,S,V]=svd(A); X=V(:,end);
```

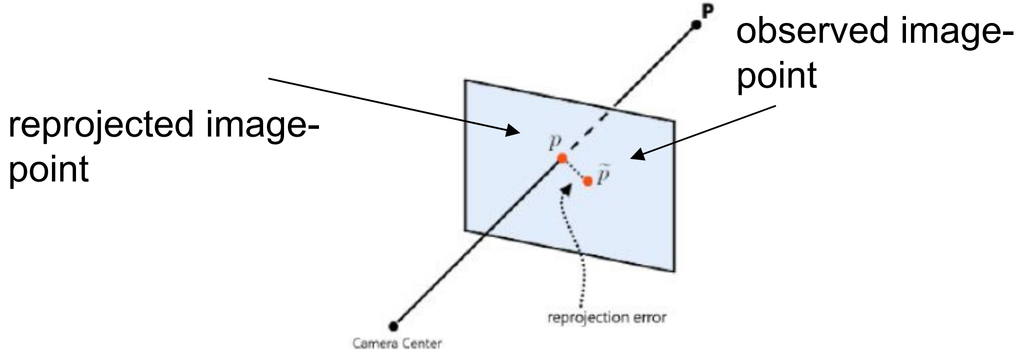
Non-linear Triangulation

- An even better triangulation method would be to minimize a meaningful geometric quantity such as the reprojection error,

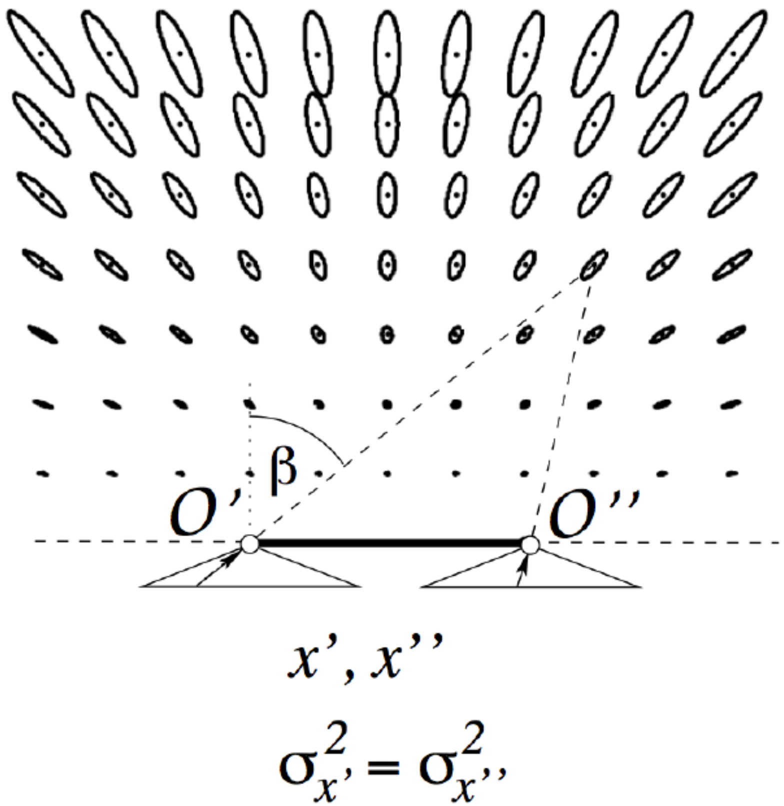
$$\epsilon = d(\bar{\mathbf{x}}_1, \mathbf{x}_1)^2 + d(\bar{\mathbf{x}}_2, \mathbf{x}_2)^2$$

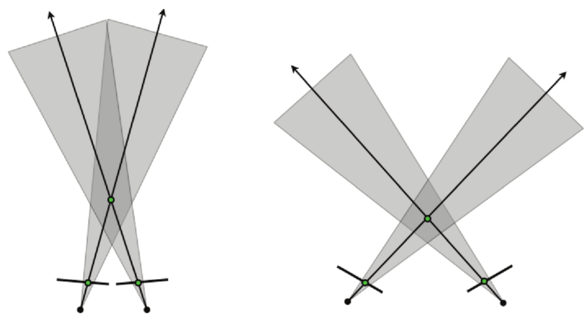
  Where $\bar{\mathbf{x}}_1$, $\bar{\mathbf{x}}_2$ denotes the actual image-coordinates and $\mathbf{x}_1$, $\mathbf{x}_2$ the reprojections, i.e. $\mathbf{x}_1 = \mathbf{P}_1\mathbf{X}$ and $\mathbf{x}_2 = \mathbf{P}_2\mathbf{X}$

- This cannot be calculated in closed-form, but can be done iteratively by e.g. employing Gauss-Newton. The starting point from the algorithm would usually be a linear estimate of the triangulated point.



Triangulation Uncertainty





(a) Small baseline (b) Large baseline

Exercise (10 min)

- Re-visit assignment 3 part 2 and experiment with various baselines between the two cameras
  - Try with baselines (d=0.25, d=0.5, d=1, d=2, d=5)
  - Analyze the triangulation accuracy for various baselines, by using ***norm(pest-ptrue)*** to quantify the overall accuracy
  - What happens with the number of triangulated points as you vary the baseline distance?
- Discuss your findings in small groups (2-3 persons)

Bundle Adjustment

- We have so far only considered 2-view (or stereo) reconstruction, in practise during photogrammetric campaigns the same 3D points are often seens in multiple views
- From a statistical perspective, it would be benificial to have more obserservations of the same 3D point from multiple images as this often can reduce measurement uncertainty
- A Bundle adjustment algorithm is fundamentally an optimization problem, which aims to minimize the reprojection error from 3D points to the observed image points. This includes refinement of the camera matrix, P as well as the location of the point it self.

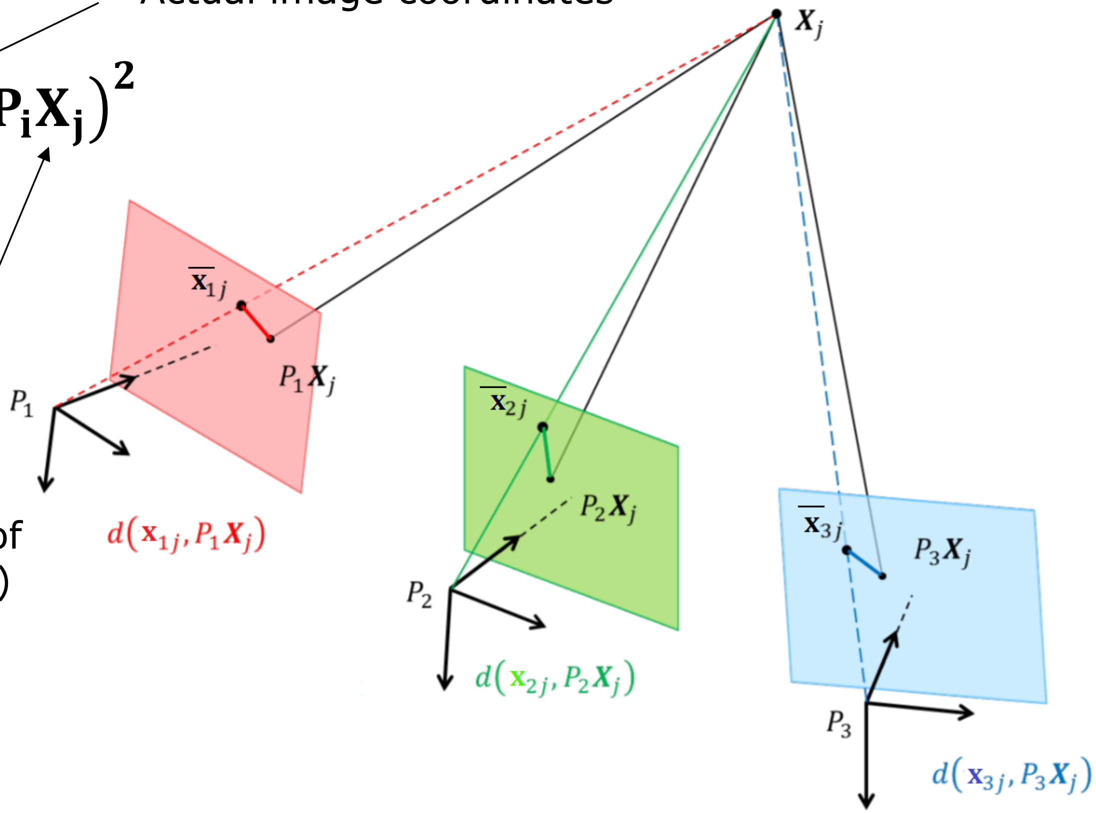
Bundle Adjustment

Actual image coordinates

$$\min_{\mathbf{P_i},\, \mathbf{X_j}} \sum_{i=1}^{m} \sum_{j=1}^{n} d\left(\overline{\mathbf{x_{ij}}}, \mathbf{P_i}\mathbf{X_j}\right)^2$$

Camera projection matrix

Estimate of 3D (world) point



Bundle Adjustment

- The optimization problem is often solved by a Non-linear least squares solver, such as the Levenberg-Marquadt algorithm.

- Assuming a calibrated camera and undistorted images, we can assume the following parameterization

Euler angles (Alternative representation of R)

$$d\left(\overline{\mathbf{x_{ij}}}, \mathbf{x_{ij}}\right) = \overline{\mathbf{x_{ij}}} - \mathbf{x_{ij}}\left(\mathbf{P_i}, \mathbf{X_j}\right) = \boldsymbol{x_{ij}} - \mathbf{x_{ij}}\left(\omega_i, \phi_i, \kappa_i, t_{x,i}, t_{y,i}, t_{z,i}, X_j, Y_j, Z_j\right) = \overline{\mathbf{x_{ij}}} - \mathbf{x_{ij}}(\beta_{ij})$$

We further assume, that (Taylor approximation)

$$\mathbf{x_{ij}}\left(\beta_{ij} + \delta_{ij}\right) \approx \mathbf{x_{ij}}\left(\beta_{ij}\right) + J_{ij}\delta_{ij}$$

- The Jacobian associated with each image point is thus

$$\boldsymbol{J_{ij}} = \left[\frac{\partial \mathbf{x_{ij}}}{\partial \mathrm{\omega_i}} \quad \frac{\partial \mathbf{x_{ij}}}{\partial \phi_i} \quad \frac{\partial \mathbf{x_{ij}}}{\partial \kappa_i} \quad \frac{\partial \mathbf{x_{ij}}}{\partial \mathrm{t_{x,i}}} \quad \frac{\partial \mathbf{x_{ij}}}{\partial \mathrm{t_{y,i}}} \quad \frac{\partial \mathbf{x_{ij}}}{\partial \mathrm{t_{z,i}}} \quad \frac{\partial \mathbf{x_{ij}}}{\partial X_j} \quad \frac{\partial \mathbf{x_{ij}}}{\partial Y_j} \quad \frac{\partial \mathbf{x_{ij}}}{\partial Z_j}\right]$$

Bundle Adjustment

$$S(\beta_{ij} + \delta_{ij}) \approx \sum_{i=1}^{m} \sum_{j=1}^{n} \left(\overline{\mathbf{x_{ij}}} - \mathbf{x_{ij}}(\beta_{ij}) + J_{ij}\delta_{ij}\right)^2$$

The cost-function can be represented in vector notation as:

$$S(\boldsymbol{\beta} + \boldsymbol{\delta}) \approx (\bar{\mathbf{x}} - \mathbf{x}(\boldsymbol{\beta}) + \boldsymbol{J}\boldsymbol{\delta})^2$$

How would we find a minimum for the above equation??

$$\boxed{\frac{\nabla S(\beta + \delta)}{\nabla \delta} = 0 \Rightarrow (\boldsymbol{J}^T\boldsymbol{J})\boldsymbol{\delta} = \boldsymbol{J}^T(\bar{\mathbf{x}} - \mathbf{x}(\boldsymbol{\beta}))}$$

The above procedure must be iterated, as the linearization is only valid close to the linearization point, β

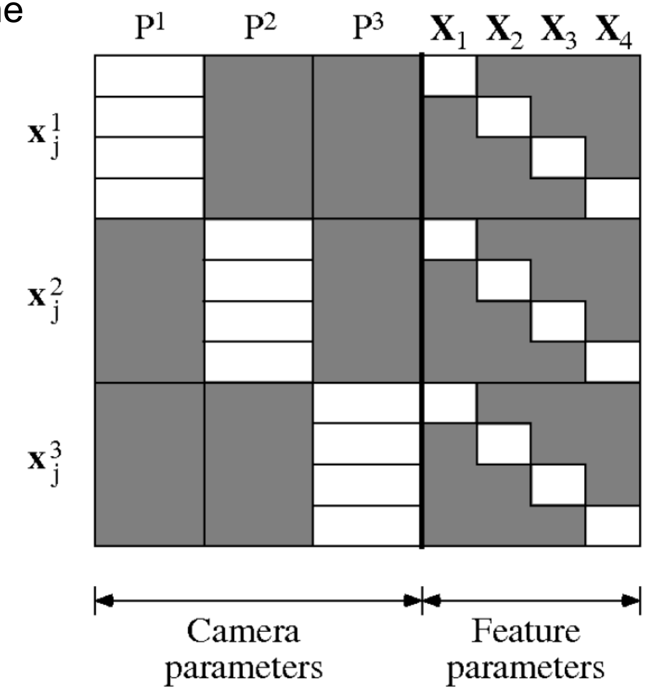
The result should be recognized as a linearized version of the normal equation (LS)

See also: https://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

Bundle Adjustment

As an example with 3 views and 4 world points, the Jacobian would be like:

Rows represents the reprojected points



Number of parameters to optimize = 3*n + 6*m, where n is the number of World Points and M is the number of camera views (for a single calibrated camera)

Why do we have 3*n + 6*m parameters to optimize? Any suggestions?

Bundle Adjustment

- Bundle adjustment is statistical optimal
  – Maximum-Likelihood estimator under Gaussian noise

- Exploit all observations and considers uncertainties and correllation

- Requires an initial estimate

- Needs fewer control points compared to P3P (which needs at least 4 per image)

Aerial Triangulation

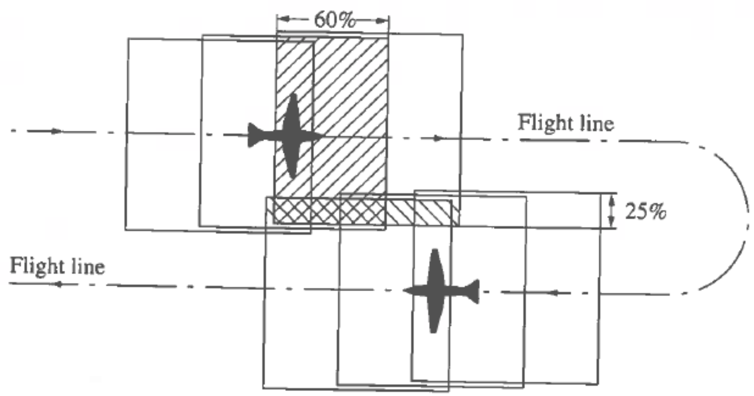


Figure 1-6 Photographic overlap.

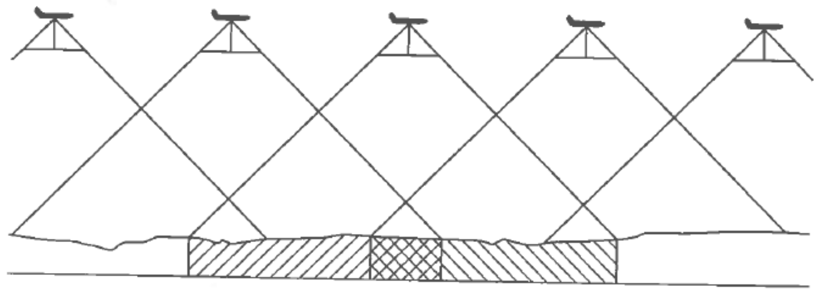
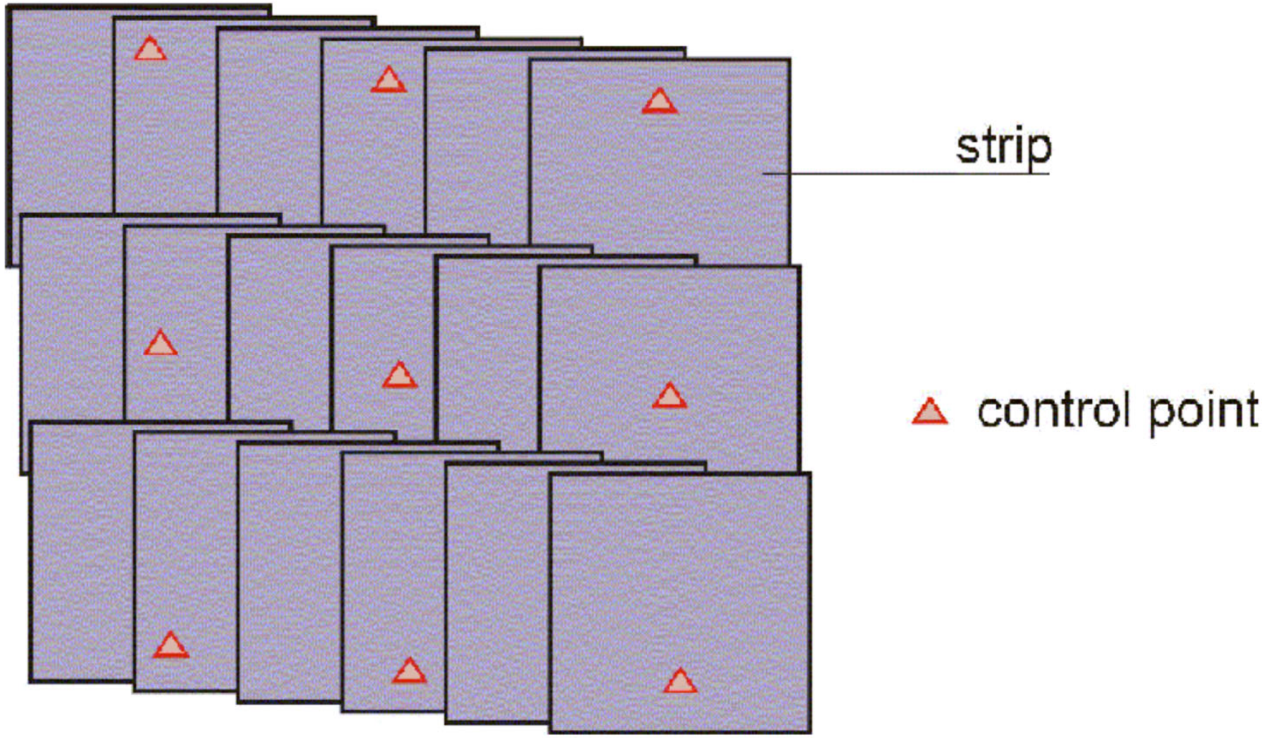


Figure 1-7 Overlap along flight line.

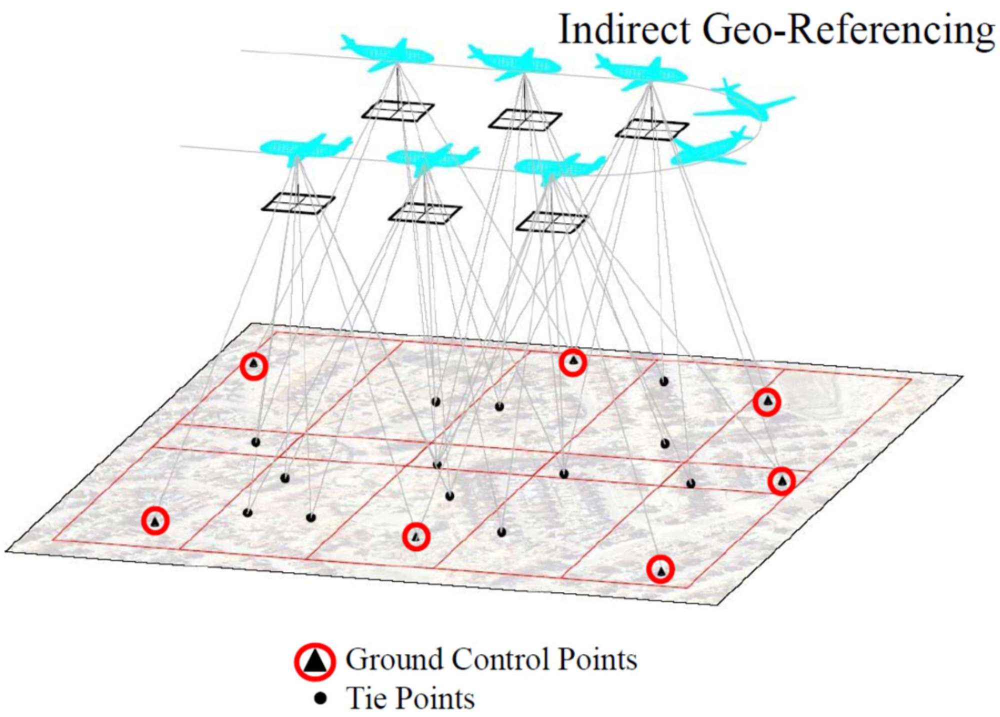
AT is the task of estimating the 3D location of points using aerial images

Bundle-adjustment is the default framework for this process

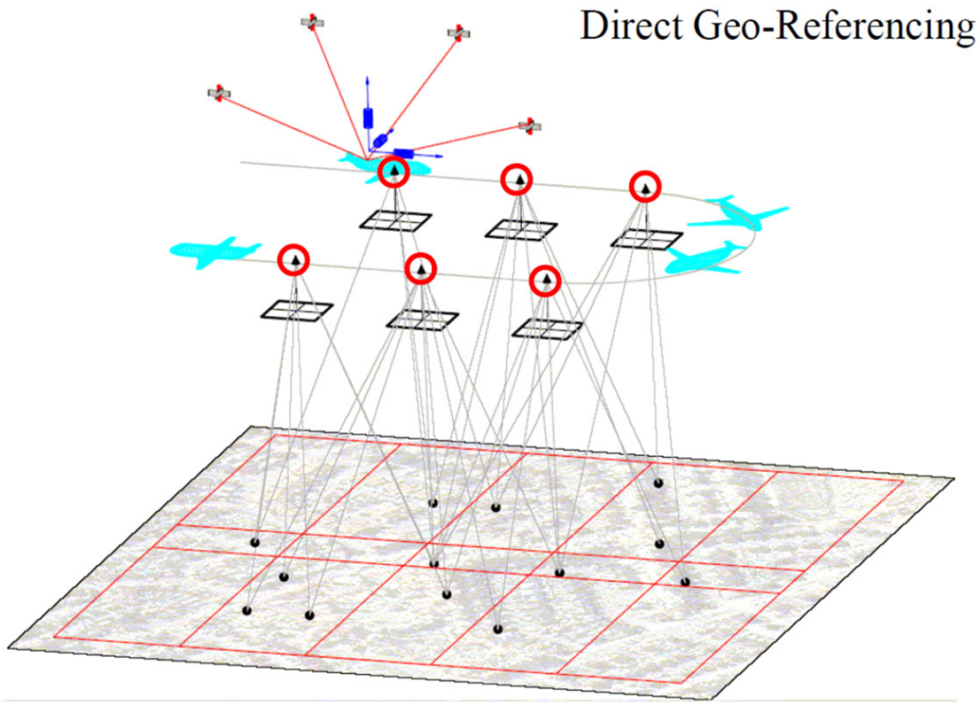
Aerial Triangulation



Aerial Triangulation



Aerial Triangulation



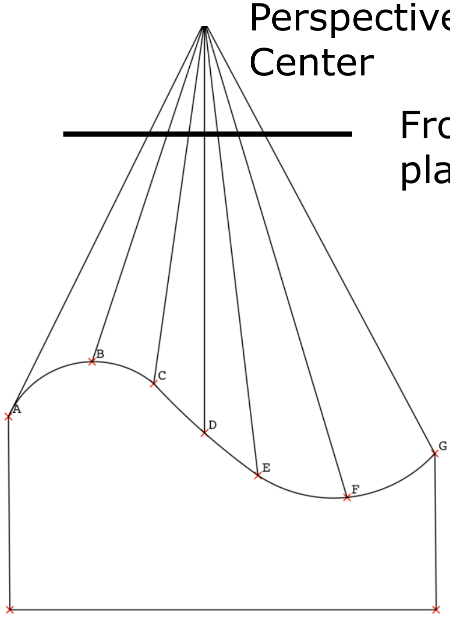
Perspective vs Orthographic Projection (Image versus Map)

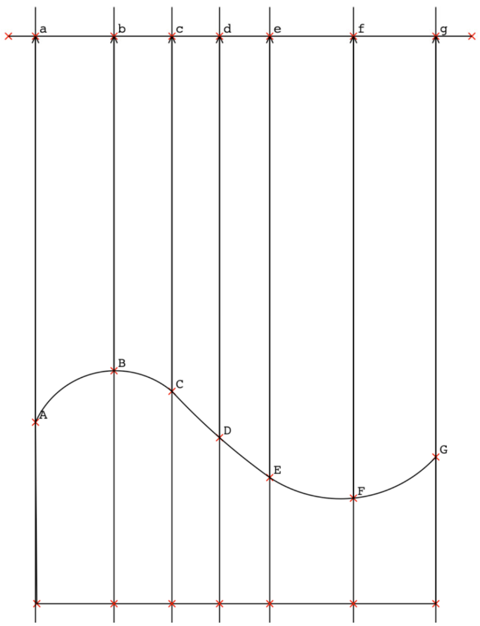
Images
- Perspective Projection
- Non-uniform scale

Maps / Orthographic Projections
- Orthographic (parallel projections)
- Uniform scale

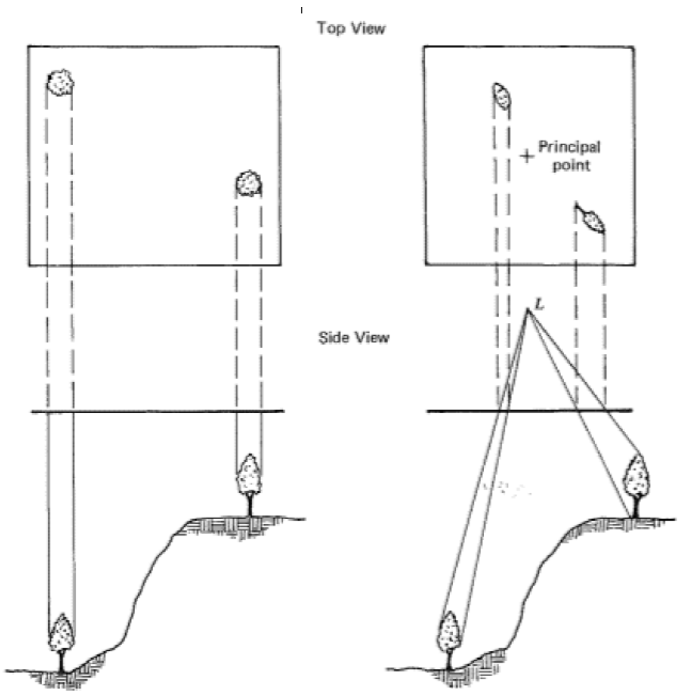
Perspective Center

Frontal image-plane





Aerial Photos: Relief Displacement



(a) Map (orthographic projection)
Constant scale
No relief displacement

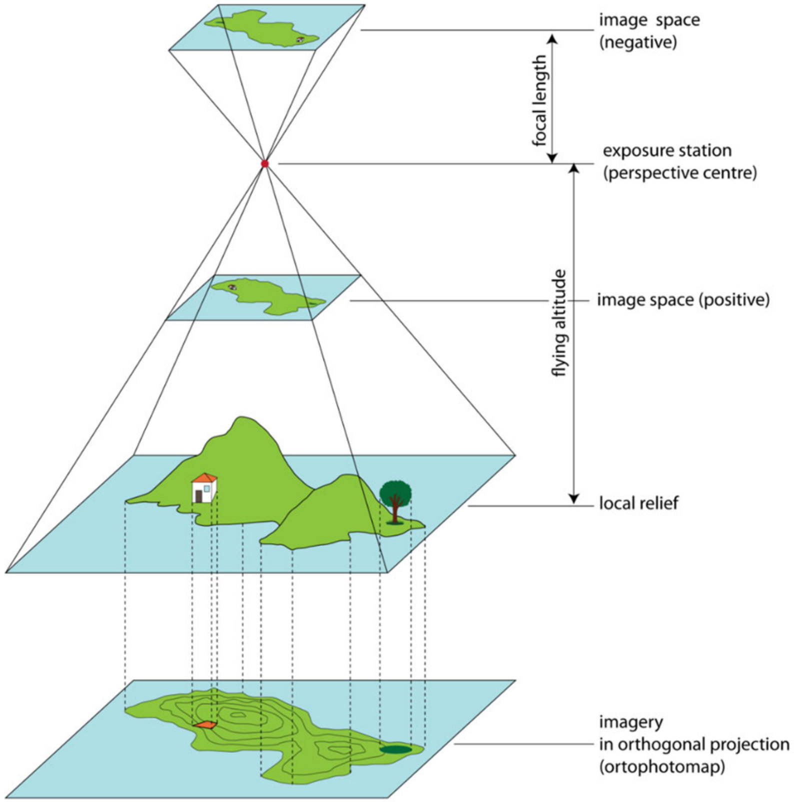
(b) Photo (perspective projection)
Varied scale
Relief displacement

For aerial images we encounter Relief displacement, which makes objects of certain heights appear tilted in the photographs.

It depends on the height of the object and is more pronounced near edges of the photograph

For a true orthophoto this effect is also mitigated

Photogrammetric Products: Orthophotos



An Orthophoto is a rectified aerial image, which has a uniform horizontal scale and hence can be be used as a ”map”

An orthographic projection requires that 3D information of objects, buildings and terrain is known in order to ensure a uniform horizontal scale.

https://ncsu-geoforall-lab.github.io/uav-lidar-analytics-course/lectures/HM_Photogrammetry_and_SfM.html

Orthoimage (backward projection)

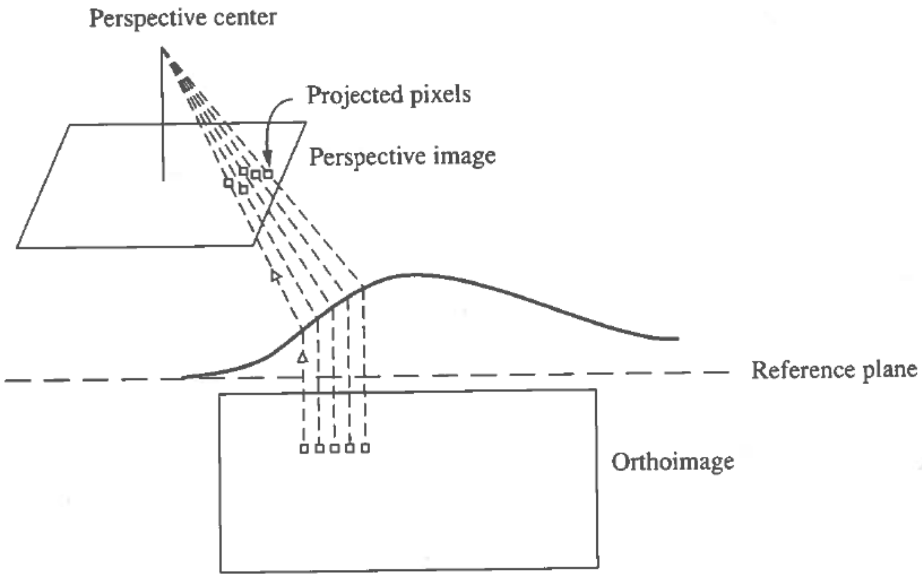


Figure 8-11 Orthoimage production using backward projection. Each pixel in the orthoimage is projected onto the terrain model and the resulting object space point is projected back into the perspective image to determine the corresponding gray value. Interpolation is done in the perspective image.