Photogrammetry and Drone Image Processing

Course offered by Prof. Mirko Reguzzoni

Outline

Least Squares

- Observations
- Deterministic Model
- Stochastic Model
- Target Function
- Solution
- Solution Accuracy
- Examples
  - * How to do computations
    - + “design” problem
  - * Transformation estimation (non-linear problem)
    - + linearization
    - + over-parametrization
    - + constraints
  - * data snooping (outlier detection)
    - + tests

Analytical Photogrammetry

- concept
- 3 step of photogrammetry
- transformation
  - * 2D <-> 2D
  - * 3D <-> 3D (rotation)
  - * 2D <-> 3D
    - + orientation parameter definition (internal+external)
    - + collinearity equations (proof)
      - orientation form and restitution form
    - + DLT (Direct Linear Transformation) (2D <-> 3D)
      - Homography (2D <-> 2D)

+Z=0

- design survey
  - * normal case (to design the survey)
    - + restitution formulas
    - + error propagation
  - * empirical formulas (simple)
  - * simulation (LS)
- orientation (estimate the orientation parameters by LS based on double points)
  - * external orientation (E.O.)
    - + space resection
    - + relative / absolute orientation
    - + BBA (N photos, more general case)
  - * internal orientation (I.O.) = calibration
    - + Brown model

Digital Photogrammetry

- introduction
  - * basics of images
  - * resolution
    - + geometrical resolution
    - + radiametric resolution
    - + spectral resolution
    - + temporal resolution

Properties

  - * format & compression
- image enhancement (general)
  - * low pass filters // smooth
  - * high pass filters // sharp
    - + edge detector
- image algorithms for photogrammetry
  - * image pyramids
  - * resampling
  - * interesting point operators
  - * image matching algorithm
    - + FBM (Feature based matching)
    - + ABM (Area based matching)
    - + relational matching

Product Photogrammetry

- 3D models
  - * vectorial models
    - + Rubik cube
  - * raster models
    - + DTM
- 2D models
  - * photoplane
  - * orthophoto
    - + google map

Labs

- close-range photogrammetry
- drone survey

2021.09.16 Introduction and Least Squares

Introduction

Recording: 00:00:00 ~ 02:00:00

Least Squares

Observation equations: $\underline{Y}_0 = \underline{y} + \underline{v}$ (n equations) — Truth; Noise/observation errors

Estimation procedure → Estimate: $\hat{\underline{y}} = \underline{y} + \underline{\hat{\varepsilon}}$ — Truth; Estimation error

$$\begin{cases} Y_{01} = y_1 + v_1 \\ Y_{02} = y_2 + v_2 \\ \vdots \\ Y_{0n} = y_n + v_n \end{cases} \qquad \underline{Y}_0 = \begin{bmatrix} Y_{01} \\ Y_{02} \\ \vdots \\ Y_{0n} \end{bmatrix}$$

Model the truth (y)

Deterministic model: $\underline{y} = A\underline{x} + \underline{a}$ (m parameters, $m \le n$) $= f(\underline{x})$

$$\begin{cases} y_1 = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m + a_1 \\ y_2 = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m + a_2 \\ \vdots \\ y_n = a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m + a_m \end{cases}$$

This model has to be LINEAR such that it can be expressed by matrix notation.

what is the difference between deterministic model and stochastic model?

Design matrix A ($n \times m$) $= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}$ Known term $\underline{a}$ ($n \times 1$) $= \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$ Parameter $\underline{x}$ ($m \times 1$) $= \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$

Model the noise (v)

Stochastic model: $E[\underline{v}] = 0 \quad C_{vv} = \sigma_0^2 Q \quad (Q \text{ is given})$

Use 0 because we do not have system errors. Accuracy of the instrument, variance of the residuals/noise (a prior)

The covariance matrix shows how much the data is correlated and it has to be positive definite: $\underline{\lambda}^T C_{vv} \underline{\lambda} > 0 \;, \forall \underline{\lambda} \Rightarrow \det(C_{vv}) > 0$

Typically, $Q = I$ (simplest one => all the data has the same variance and they are uncorrelated one to another)

$C_{y_0 y_0} = C_{yy} + C_{vv} = C_{vv}$

v is a random variable, which can be described by probability distribution function. For easy description, use mean and covariance (That is, assume that v is normal distribution with zero mean.)

Can we model the noise deterministic?

If we have deterministic model for noise, it can be included into the deterministic model. For example, the model we get can be the truth model+calibration parameter.

In the LS Adjustment, we assume that v is only random. This means that if we have deterministic errors and we don't model it into the model, it can be a mistake.

- $n < m$ => under determine problem (infinite solutions)
- $n = m$ => iso determine problem (one exact solution)
- $n > m$ (typical) => over determine problem => no exact solution, but we can define a TARGET FUNCTION to choose the solution

Target function: $\min_{\underline{x}} \Phi(\underline{x}) = (\underline{Y}_0 - \underline{y})^T Q^{-1} (\underline{Y}_0 - \underline{y}) = (\underline{Y}_0 - A\underline{x} - \underline{a})^T Q^{-1} (\underline{Y}_0 - A\underline{x} - \underline{a})$ # quadratic form = number

From the deterministic model; From the stochastic model

To weight more the data which have lower variance $w_i = \frac{1}{\sigma_i^2}$

$\frac{d\Phi(\underline{x})}{d\underline{x}} = 0$

Principle of LS: $\min = (\underline{Y}_0 - \hat{\underline{y}})^T Q^{-1} (\underline{Y}_0 - \hat{\underline{y}}) = (\underline{Y}_0 - A\hat{\underline{x}} - \underline{a})^T Q^{-1} (\underline{Y}_0 - A\hat{\underline{x}} - \underline{a})$

Solution:

$$\hat{\underline{x}} = (\underbrace{A^T Q^{-1} A}_{\text{Normal matrix}})^{-1} A^T Q^{-1} (\underline{Y}_0 - \underline{a}) = \underbrace{N^{-1} A^T Q^{-1}}_{P} (\underline{Y}_0 - \underline{a})$$

estimated parameters

Accuracy: Estimated observations $\hat{\underline{y}} = A\hat{\underline{x}} + \underline{a} = \underline{y} + \underline{\hat{\varepsilon}}$ → Residuals $\hat{\underline{v}} = \underline{Y}_0 - \hat{\underline{y}}$ → Variance of the estimated residuals (a posteriori) $\hat{\sigma}_0^2 = \frac{\hat{\underline{v}}^T Q^{-1} \hat{\underline{v}}}{n-m}$

Increase if you have more bad data of a wrong model

Why "n-m"? Because m data is used to estimate the parameters x

$$\hat{C}_{\hat{x}\hat{x}} = \hat{\sigma}_0^2 N^{-1} \qquad \hat{C}_{\hat{y}\hat{y}} = A C_{\hat{x}\hat{x}} A^T = \hat{\sigma}_0^2 A N^{-1} A^T$$

$P^T C_{y_0 y_0} P$

Inferred by covariance propagation

(1D) $Y = aX + b$	(ND) $\underline{Y} = A\underline{X} + \underline{b}$
σ_x^2 given → $\sigma_Y^2 = a^2 \sigma_x^2$	$C_{\underline{X}\underline{X}}$ given → $C_{\underline{Y}\underline{Y}} = A\, C_{\underline{X}\underline{X}} A^T$
μ_x given → $\mu_Y = a\mu_x + b$	$\mu_{\underline{X}}$ given → $\mu_{\underline{Y}} = A\mu_{\underline{X}} + \underline{b}$

$C_{vv} = \sigma_0^2 Q$

$Y_0 = y + v \rightarrow C_{y_0 y_0} = C_{vv} = \sigma_0^2 Q$

$$\hat{\sigma}_0^2 = \frac{(Y_0 - \hat{y})^T Q^{-1} (Y_0 - \hat{y})}{n-m}$$

$$\hat{\underline{x}} = (A^T Q^{-1} A)^{-1} A^T Q^{-1} (y_0 - a) = N^{-1} A^T Q^{-1} (y_0 - a) = P(y_0 - a)$$

$$C_{\hat{x}\hat{x}} = P\, C_{y_0 y_0} P^T = (N^{-1} A^T Q^{-1})\, \sigma_0^2 Q\, (N^{-1} A^T Q^{-1})^T = \sigma_0^2 N^{-1} A^T (Q^{-1})^T A (N^{-1})^T$$

$\hat{y} = A\hat{x} + a$

$\rightarrow C_{\hat{y}\hat{y}} = A C_{\hat{x}\hat{x}} A^T = A \sigma_0^2 N^{-1} A^T$

2021.09.17 Least Squares

How to do computations

// recording: 00:55:00
// see exercise below

Normal matrix N = $A^T Q^{-1} A$

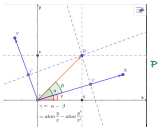
Do you prefer large det N or small det N?

$\hat{C}_{\hat{x}\hat{x}} = \hat{\sigma}_0^2 N^{-1}$ => the smaller the det N, the larger the noise of the deterministic parameters
So the good design is a design in which the det N is as large as possible. (Consider comparing det N obtained in Exercise 1 with the t=0.1,0.2,0.3,0.5 which have a smaller det N and worst result)

Combined with Q in design matrix N
=> the quality of the estimation depends partly on the quality of the instrument (Q), also depends on the capability of designing well the survey ("design" problem).

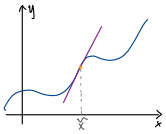
Transformation estimation (non-linear transformation)

Case: 2D -> 2D (rotation)



P = double point

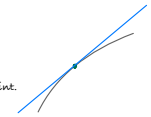
Linearization



Also called tangent line approximation.

Basic idea: given an approximated point in the curve, we can use the tangent line of the curve in that point to approximate other values along the curve around the approximated point

What does it mean linearizing?
You need a point around which you want to linearize the formula (any point.) => approximated points
And then linearization is around the approximated point which is the tangent of the curve in the approximated point.



Non-linear model:

$$y = f(x) \cong A\delta x + b \quad (m \text{ parameters}, m \le n) \quad \#\ n \text{ observations}$$

$$\begin{cases} y_1 = f_1(x_1,x_2,\cdots,x_m) \cong f_1(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) + \left.\frac{\partial f_1}{\partial x_1}\right|_{\tilde{x}}(x_1-\tilde{x}_1) + \left.\frac{\partial f_1}{\partial x_2}\right|_{\tilde{x}}(x_2-\tilde{x}_2) + \cdots + \left.\frac{\partial f_1}{\partial x_m}\right|_{\tilde{x}}(x_m-\tilde{x}_m) \\ y_2 = f_2(x_1,x_2,\cdots,x_m) \cong f_2(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) + \left.\frac{\partial f_2}{\partial x_1}\right|_{\tilde{x}}(x_1-\tilde{x}_1) + \left.\frac{\partial f_2}{\partial x_2}\right|_{\tilde{x}}(x_2-\tilde{x}_2) + \cdots + \left.\frac{\partial f_2}{\partial x_m}\right|_{\tilde{x}}(x_m-\tilde{x}_m) \\ \vdots \\ y_n = f_n(x_1,x_2,\cdots,x_m) \cong f_n(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) + \left.\frac{\partial f_n}{\partial x_1}\right|_{\tilde{x}}(x_1-\tilde{x}_1) + \left.\frac{\partial f_n}{\partial x_2}\right|_{\tilde{x}}(x_2-\tilde{x}_2) + \cdots + \left.\frac{\partial f_n}{\partial x_m}\right|_{\tilde{x}}(x_m-\tilde{x}_m) \end{cases}$$

Jacobian Matrix

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_m} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_m} \\ \vdots & & & \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_m} \end{bmatrix}_{\tilde{x}} \quad A$$

$$b = \begin{bmatrix} f_1(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) \\ f_2(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) \\ \vdots \\ f_n(\tilde{x}_1,\tilde{x}_2,\cdots,\tilde{x}_m) \end{bmatrix} \quad l$$

$$\delta x = \begin{bmatrix} x_1-\tilde{x}_1 \\ x_2-\tilde{x}_2 \\ \vdots \\ x_m-\tilde{x}_m \end{bmatrix} \quad x$$

$$\delta\hat{x} = (J^T Q^{-1} J)^{-1} J^T Q^{-1}(Y_0 - b) = N^{-1} A^T Q^{-1}(Y_0 - b)$$

$$\hat{x} = \tilde{x} + \delta\hat{x} \rightarrow \tilde{x} \text{ and then iterate until convergence.}$$

If your approximated points are good => converge
If your approximated points are not good => not converge or converge to a solution which is not the optimal one

$C_{\hat{x}\hat{x}} = \hat{\sigma}_0^2 N^{-1}$

// see the example in exercise 2

over-parametrization

Step:
1. Replace physical parameters with artificial algebric parameters to change the non-linear problem to the linear one
2. Solve the new linear problem => estimated algebric parameters
3. Get the original physical parameters based on the estimated algebric parameters

// see exercise 3.

Typically way of preceding a non-linear system:

Given a non linear system,
1. Over-parametrization
2. Estimate algebric parameters
3. Compared them to physical parameters, which is done with approximation => approximated values
4. Used the result generated in step 3 to Linearization this non-linear system

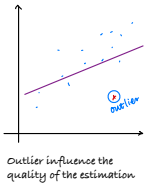
constraints

The constraints can be considered as pseudo observations (see exercise 4).

Note that in this case, the constraints are deterministic which means the noise for these pseudo observations are 0. However, if we use 0 directly, det Q will be 0 which violate the requirement for Q (should be > 0). Therefore, we can use a very small value (for example, 10^-6) instead of 0 to avoid this problem.

Data snooping (outlier detection)

Compare $\bar{\sigma}_0^2$ and $\hat{\sigma}_0^2$
- $\bar{\sigma}_0^2 = \hat{\sigma}_0^2$ You model is correct
- $\bar{\sigma}_0^2 < \hat{\sigma}_0^2$ You model is somehow misleading



Outlier influence the quality of the estimation

Given a:

Test on $\hat{\sigma}_0^2$: (Quality) $H_0 : \sigma_0^2 = \bar{\sigma}_0^2$

$$\frac{\hat{\sigma}_0^2}{\bar{\sigma}_0^2}(n-m) \sim \chi^2_{n-m}$$

$\chi^2_{emp} = \frac{\hat{\sigma}_0^2}{\bar{\sigma}_0^2}(n-m)$ ↕ compare

Test on $\hat{x}_i$: (Estimate) $H_0 : x_i = \bar{x}_i$

$$\frac{\hat{x}_i - \bar{x}_i}{\hat{\sigma}_{\hat{x}_i}} \sim t_{n-m}$$

$\hat{\sigma}_{\hat{x}_i}$ ↓ temp ↓ compare

standardization

Can be used to check if
- your model is correct
- there is any outlier in your dataset

If the residuals is getting larger and larger ($\hat{\sigma}_0^2$ increase), it maybe because
- using of a wrong model
- there exists outliers

Is it true that the smaller the estimated variance, the better the fitting of the data?

If you use a model which n=m, all you are doing is simply observing all the noise into your solution. If the residuals are all zero, you are not able to evaluate the accuracy of your estimate.

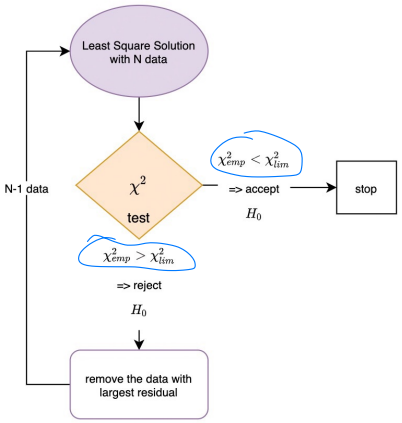
So having a smaller sigma_square is significant only when you have a large dataset (REDUNDANCY).

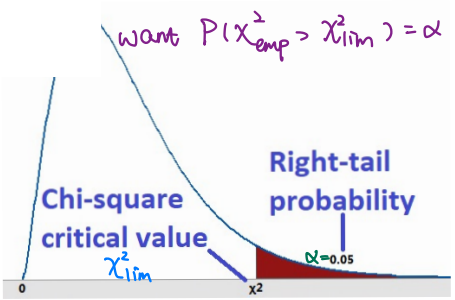
Outlier rejection: if the test on $\hat{\sigma}_0^2$ fails, (1) look for the maximum residual, (2) delete the corresponding observation and (3) repeat the procedure till the test is passed.

Instead of the estimated residuals, it would be better to look for the maximum normalized residual:

$$\hat{\underline{u}}_i = \frac{\hat{\underline{v}}_i}{\sigma_0 \sqrt{Q_{\hat{v}\hat{v}_{ii}} - \left(AN^{-1}A^T\right)_{ii}}}$$

$$C_{\hat{v}\hat{v}} = \sigma_0^2 \left(Q - AN^{-1}A^T\right)$$





For example, with double points, outliers means wrong clicking. So that this process can be used to identify it.

Exercise 1: linear regression

Consider the following observations in time and interpolate them with a linear model.
Assume that the observations have the same accuracy and that they are uncorrelated.

t	$Y_0(t)$
1	5.09
2	6.91
3	9.36
5	12.71

$n=4, m=2$

$Y = Ax + Q$ (deterministic + stochastic)

$$Y_{0_i} = at_i + b + v_i \qquad i=1,2,3,4$$

$$\underline{Y}_0 = \begin{bmatrix} 5.09 \\ 6.91 \\ 9.36 \\ 12.71 \end{bmatrix} \quad \text{Design matrix } A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 5 & 1 \end{bmatrix} \quad n \times m \; 4\times 2 \quad \text{Known term } \underline{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \underline{x} = \begin{bmatrix} a \\ b \end{bmatrix} \quad Q=I \quad C_{VV} = \sigma_0^2 Q$$

Normal Matrix $N = \begin{bmatrix} 39 & 11 \\ 11 & 4 \end{bmatrix}$ $\quad N = A^T Q^{-1} A$

$\det N = 35 = 39\times 4 - 11\times 11$

$$N^{-1} = \frac{1}{35}\begin{bmatrix} 4 & -11 \\ -11 & 39 \end{bmatrix}$$

$$\hat{\underline{x}} = \begin{bmatrix} 1.9254 \\ 3.2226 \end{bmatrix} = N^{-1}A^T y_0 \qquad \hat{\underline{y}} = \begin{bmatrix} 5.1480 \\ 7.0734 \\ 8.9989 \\ 12.8497 \end{bmatrix} \qquad \hat{\underline{v}} = \begin{bmatrix} -0.0580 \\ -0.1634 \\ 0.3611 \\ -0.1397 \end{bmatrix} \qquad \hat{\sigma}_0^2 = 0.09 \qquad \sigma_0^2 = 0.3$$

$$C_{\hat{x}\hat{x}} = \begin{bmatrix} 0.0103 & -0.0283 \\ -0.0283 & 0.1003 \end{bmatrix} = \hat{\sigma}_0^2 N^{-1} = \frac{\hat{\sigma}_0^2}{\det N}\begin{bmatrix} 4 & -11 \\ -11 & 39 \end{bmatrix}$$

$$C_{\hat{y}\hat{y}} = \begin{bmatrix} 0.0540 & & & \\ & 0.0283 & & \\ & & 0.0231 & \\ & & & 0.0746 \end{bmatrix} \quad A C_{\hat{x}\hat{x}} A^T$$

(covariances are not computed)

The center points are estimated better than the border points

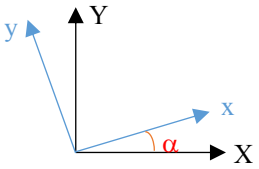
Exercise 2: non-linear model, solved by linearization

// recording: 01:37:00

Estimate the rotation angle between two 2D reference systems by using the three given double points.
Assume that the observations have the same accuracy and that they are uncorrelated. $\rightarrow Q = I$

	x	y	X	Y
P1	0.0993	0.8871	-0.3413	0.8814
P2	0.4710	0.2295	0.2372	0.3895
P3	-0.5932	0.1885	-0.5769	-0.1266

$n = 3\times 2$
$m = 1$



parameter, noise

$$\begin{cases} X_i = x_i \cos\alpha - y_i \sin\alpha + v_i \\ Y_i = x_i \sin\alpha + y_i \cos\alpha + \eta_i \end{cases} \quad i = 1,2,3$$

Model $f(x, y, \alpha)$

$$\begin{cases} \tilde{X}_i = x_i \cos\tilde{\alpha} - y_i \sin\tilde{\alpha} \\ \tilde{Y}_i = x_i \sin\tilde{\alpha} + y_i \cos\tilde{\alpha} \end{cases} \quad i = 1,2,3$$

$$\begin{cases} X_i \cong \tilde{X}_i + (-x_i \sin\tilde{\alpha} - y_i \cos\tilde{\alpha})(\alpha - \tilde{\alpha}) \\ Y_i \cong \tilde{Y}_i + (x_i \cos\tilde{\alpha} - y_i \sin\tilde{\alpha})(\alpha - \tilde{\alpha}) \end{cases} \quad i = 1,2,3$$

$$\tilde{\alpha} = \arctan\left(\frac{Y_1}{X_1}\right) - \arctan\left(\frac{y_1}{x_1}\right) = 27.5537°$$

$$\underline{Y}_0 = \begin{bmatrix} -0.3413 \\ 0.8814 \\ 0.2372 \\ 0.3895 \\ -0.5769 \\ -0.1266 \end{bmatrix} \quad J = \begin{bmatrix} -0.8325 \\ -0.3223 \\ -0.4214 \\ 0.3114 \\ 0.1073 \\ -0.6131 \end{bmatrix} \quad \underline{b} = \begin{bmatrix} -0.3223 \\ 0.8325 \\ 0.3114 \\ 0.4214 \\ -0.6131 \\ -0.1073 \end{bmatrix} \quad \delta\underline{x} = \delta\alpha \quad Q = I$$

$$N = 1.4588 \qquad J^T(\underline{Y}_0 - \underline{b}) = 0.0371 \qquad \delta\hat{\underline{x}} = 1.4556° \qquad \hat{\underline{x}} = 29.0093°$$

$$\hat{\underline{y}} = \begin{bmatrix} -0.3434 \\ 0.8243 \\ 0.3007 \\ 0.4293 \\ -0.6104 \\ -0.1228 \end{bmatrix} \quad \hat{\underline{v}} = \begin{bmatrix} 0.0022 \\ 0.0572 \\ -0.0635 \\ -0.0398 \\ 0.0334 \\ -0.0037 \end{bmatrix} \quad \hat{\sigma}_0^2 = 0.0020 \quad C_{\hat{x}\hat{x}} = \sigma_{\hat{x}}^2 = 0.0014[rad^2] \quad \sigma_{\hat{x}} = 2.1239°$$

NOTE: the estimated value $\hat{\underline{x}}$ should be used as a new approximate value $\tilde{\underline{x}}$ and the procedure should be iterated until convergence, namely until the absolute difference between two consecutive estimates of $\hat{\underline{x}}$ is below a fixed threshold.

Exercise 3: non-linear model, solved by over-parametrization

The problem is the same of exercise 2.

$$\begin{cases} X_i = x_i \cos\alpha - y_i \sin\alpha \\ Y_i = x_i \sin\alpha + y_i \cos\alpha \end{cases} \quad i = 1,2,3$$

$$\begin{cases} a = \cos\alpha \\ b = \sin\alpha \end{cases} \rightarrow \begin{cases} X_i = x_i a - y_i b \\ Y_i = x_i b + y_i a \end{cases}$$

$$\underline{Y}_0 = \begin{bmatrix} -0.3413 \\ 0.8814 \\ 0.2372 \\ 0.3895 \\ -0.5769 \\ -0.1266 \end{bmatrix} \quad A = \begin{bmatrix} 0.0993 & -0.8871 \\ 0.8871 & 0.0993 \\ 0.4710 & -0.2295 \\ 0.2295 & 0.4710 \\ -0.5932 & -0.1885 \\ 0.1885 & -0.5932 \end{bmatrix} \quad \underline{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \underline{x} = \begin{bmatrix} a \\ b \end{bmatrix} \quad Q = I$$

$$N = \begin{bmatrix} 1.4588 & 0 \\ 0 & 1.4588 \end{bmatrix} \quad A^T Y_0 = \begin{bmatrix} 1.2675 \\ 0.7031 \end{bmatrix}$$

$$\underline{\hat{x}} = \begin{bmatrix} 0.8689 \\ 0.4820 \end{bmatrix} \qquad \underline{\hat{y}} = \begin{bmatrix} -0.3413 \\ 0.8187 \\ 0.2986 \\ 0.4264 \\ -0.6063 \\ -0.1221 \end{bmatrix} \qquad \underline{\hat{v}} = \begin{bmatrix} 0 \\ 0.0627 \\ -0.0614 \\ -0.0370 \\ 0.0293 \\ -0.0045 \end{bmatrix} \qquad \hat{\sigma}_0^2 = 0.0025 \qquad C_{\hat{x}\hat{x}} = \begin{bmatrix} 0.0017 & 0 \\ 0 & 0.0017 \end{bmatrix}$$

$= \begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}$

$$\alpha = \arctan\left(\frac{b}{a}\right) \cong \arctan\left(\frac{\hat{b}}{\hat{a}}\right) - \frac{\hat{b}}{\hat{a}^2+\hat{b}^2}(a-\hat{a}) + \frac{\hat{a}}{\hat{a}^2+\hat{b}^2}(b-\hat{b}) = const - \frac{\hat{b}\,a}{\hat{a}^2+\hat{b}^2} + \frac{\hat{a}\,b}{\hat{a}^2+\hat{b}^2}$$

$$\hat{\alpha} = \arctan\left(\frac{\hat{b}}{\hat{a}}\right) = 29.0188°$$

$$J = \begin{bmatrix} -\dfrac{\hat{b}}{\hat{a}^2+\hat{b}^2} & \dfrac{\hat{a}}{\hat{a}^2+\hat{b}^2} \end{bmatrix} \qquad \sigma_{\hat{\alpha}}^2 = JC_{\hat{x}\hat{x}}J^T \qquad \sigma_{\hat{\alpha}} = 2.3820°$$

Exercise 4: non-linear model, solved by over-parametrization and constraints

The problem is the same of exercise 2.

$$\begin{cases} X_i = x_i \cos\alpha - y_i \sin\alpha \\ Y_i = x_i \sin\alpha + y_i \cos\alpha \end{cases} \qquad i = 1,2,3$$

$n = 6+2 = 8$
$m = 4$

$$\begin{cases} X_i = x_i a_{11} + y_i a_{12} \\ Y_i = x_i a_{21} + y_i a_{22} \end{cases} \quad \text{with} \quad \underbrace{\begin{cases} a_{11} = a_{22} \\ a_{12} = -a_{21} \end{cases}}_{\text{constraints}} \rightarrow \begin{cases} X_i = x_i a_{11} + y_i a_{12} \\ Y_i = x_i a_{21} + y_i a_{22} \\ 0 = a_{11} - a_{22} \\ 0 = a_{12} - a_{21} \end{cases}$$

$$\underline{Y}_0 = \begin{bmatrix} -0.3413 \\ 0.8814 \\ 0.2372 \\ 0.3895 \\ -0.5769 \\ -0.1266 \\ 0 \\ 0 \end{bmatrix} \qquad n\times 1 \quad 8\times 1$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{21} & a_{22} \\ 0.0993 & 0.8871 & 0 & 0 \\ 0 & 0 & 0.0993 & 0.8871 \\ 0.4710 & 0.2295 & 0 & 0 \\ 0 & 0 & 0.4710 & 0.2295 \\ -0.5932 & 0.1885 & 0 & 0 \\ 0 & 0 & -0.5932 & 0.1885 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \qquad n\times m \quad 8\times 4$$

$$\underline{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad n\times 1 \quad 8\times 1 \qquad \underline{x} = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{21} \\ a_{22} \end{bmatrix} \qquad m\times 1 \quad 4\times 1$$

Note that the design matrix now becomes easier

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10^{-6} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-6} \end{bmatrix} \qquad n\times n \quad 8\times 8$$

To avoid $\det Q = 0$

$$N = 10^6 \begin{bmatrix} 1.000000583565 & 0.000000084412 & 0 & -1 \\ 0.000000084412 & 1.000000875196 & 1 & 0 \\ 0 & 1 & 1.000000583565 & 0.000000084412 \\ -1 & 0 & 0.000000084412 & 1.000000875196 \end{bmatrix} \qquad A^T Y_0 = \begin{bmatrix} 0.4200 \\ -0.3571 \\ 0.3461 \\ 0.8475 \end{bmatrix}$$

$$\underline{\hat{x}} = \begin{bmatrix} 0.8689 \\ -0.4820 \\ 0.4820 \\ 0.8689 \end{bmatrix} \qquad \hat{\sigma}_0^2 = 0.0025 \qquad C_{\hat{x}\hat{x}} = \begin{bmatrix} 0.0017 & 0 & 0 & 0.0017 \\ 0 & 0.0017 & -0.0017 & 0 \\ 0 & -0.0017 & 0.0017 & 0 \\ 0.0017 & 0 & 0 & 0.0017 \end{bmatrix}$$

$(A^T Q^{-1} A)^{-1} A^T Q^{-1} (y_0 - e_0)$

$$\hat{\alpha} = 29.0188° \qquad \sigma_{\hat{\alpha}} = 2.3820° \qquad \text{(see Exercise 3)}$$

Exercise 5: non-linear model, solved by over-parametrization with outliers

The problem is the same of exercise 2 with an additional double point.

	x	y	X	Y
P1	0.0993	0.8871	-0.3413	0.8814
P2	0.4710	0.2295	0.2372	0.3895
P3	-0.5932	0.1885	-0.5769	-0.1266
P4	0.1689	0.6629	-0.6235	0.2814

$$\begin{cases} X_i = x_i \cos\alpha - y_i \sin\alpha \\ Y_i = x_i \sin\alpha + y_i \cos\alpha \end{cases} \qquad i = 1,2,3,4$$

$$\begin{cases} a = \cos\alpha \\ b = \sin\alpha \end{cases} \rightarrow \begin{cases} X_i = x_i a - y_i b \\ Y_i = x_i b + y_i a \end{cases}$$

$$\underline{Y}_0 = \begin{bmatrix} -0.3413 \\ 0.8814 \\ 0.2372 \\ 0.3895 \\ -0.5769 \\ -0.1266 \\ -0.6235 \\ 0.2814 \end{bmatrix} \qquad A = \begin{bmatrix} 0.0993 & -0.8871 \\ 0.8871 & 0.0993 \\ 0.4710 & -0.2295 \\ 0.2295 & 0.4710 \\ -0.5932 & -0.1885 \\ 0.1885 & -0.5932 \\ 0.1689 & -0.6629 \\ 0.6629 & 0.1689 \end{bmatrix} \qquad \underline{a} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \underline{x} = \begin{bmatrix} a \\ b \end{bmatrix} \qquad Q{=}I$$

$$N = \begin{bmatrix} 1.9267 & 0 \\ 0 & 1.9267 \end{bmatrix} \qquad A^T Y_0 = \begin{bmatrix} 1.3488 \\ 1.1639 \end{bmatrix}$$

$$\underline{\hat{x}} = \begin{bmatrix} 0.7000 \\ 0.6041 \end{bmatrix} \qquad \underline{\hat{y}} = \begin{bmatrix} -0.4664 \\ 0.6810 \\ 0.1911 \\ 0.4452 \\ -0.5291 \\ -0.2264 \\ -0.2822 \\ 0.5661 \end{bmatrix} \qquad \underline{\hat{v}} = \begin{bmatrix} 0.1251 \\ 0.2004 \\ 0.0461 \\ -0.0557 \\ -0.0478 \\ 0.0998 \\ -0.3412 \\ -0.2846 \end{bmatrix} \qquad \hat{\sigma}_0^2 = 0.0451$$

$H_0 : \sigma_0^2 = \bar{\sigma}_0^2 = (0.05)^2$

$\chi^2_{emp} = \frac{\hat{\sigma}_0^2}{\bar{\sigma}_0^2}(n-m) = \frac{0.0451}{0.0025} \cdot 6 = 108.3074 \qquad \chi^2_{n-m=6;\alpha=1\%} = 16.8119 \qquad H_0$ is rejected

$$C_{\hat{v}\hat{v}} = \begin{bmatrix} 0.5864 & & & & & & & \\ & 0.5864 & & & & & & \\ & & 0.8575 & & & & & \\ & & & 0.8575 & & & & \\ & & & & 0.7989 & & & \\ & & & & & 0.7989 & & \\ & & & & & & 0.7571 & \\ & & & & & & & 0.7571 \end{bmatrix}$$

$$\underline{\hat{u}} = \begin{bmatrix} 0.1634 \\ 0.2617 \\ 0.0498 \\ -0.0602 \\ -0.0535 \\ 0.1117 \\ -0.3922 \\ -0.3271 \end{bmatrix}$$ → P4 is the double point with the highest normalized residuals → outlier

The point P4 is disregarded and the least-squares adjustment is redone considering P1, P2, P3 only (see Exercise 3)

$$\underline{\hat{x}} = \begin{bmatrix} 0.8689 \\ 0.4820 \end{bmatrix} \qquad \underline{\hat{y}} = \begin{bmatrix} -0.3413 \\ 0.8187 \\ 0.2986 \\ 0.4264 \\ -0.6063 \\ -0.1221 \end{bmatrix} \qquad \underline{\hat{v}} = \begin{bmatrix} 0 \\ 0.0627 \\ -0.0614 \\ -0.0370 \\ 0.0293 \\ -0.0045 \end{bmatrix} \qquad \hat{\sigma}_0^2 = 0.0025$$

$H_0 : \sigma_0^2 = \bar{\sigma}_0^2 = (0.05)^2$

$\chi^2_{emp} = \frac{\hat{\sigma}_0^2}{\bar{\sigma}_0^2}(n-m) = \frac{0.0025}{0.0025} \cdot 4 = 4 \qquad \chi^2_{n-m=4;\alpha=1\%} = 13.2767 \qquad H_0$ is accepted

$\hat{\alpha} = 29.0188° \qquad \sigma_{\hat{\alpha}} = 2.3820°$ (see Exercise 3)

2021.09.23 Outlier detection; Photogrammetric survey steps; 2D Transformations

// recording begins 00:43:30

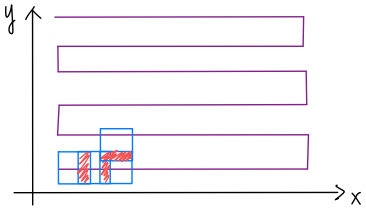
Photogrammetry is to use the camera to take photos and reconstruct what you see using these photos (target).

Double points:
- used to link image and image / image and object

Photogrammetry survey steps

Step 0: camera callibration — to known the distortions the camera produces

Step 1: Acquisition of images: can be assimilated to a central perspective onto a plane (3D -> 2D)
- Nadiral and convergent images



- Nadiral: a nadir image is a satellite image or aerial photo of the Earth taken vertically.

Nonlinear problem

Step 2. Orientation: Mathematical reconstruction of the position and attitude of the images at the time of the acquisition

$(\zeta, \eta) = f(x, y, z, \theta)$

In the photo — in the reality

- position of camera
- attitude of camera
- geometry of camera

Collinearity Equation

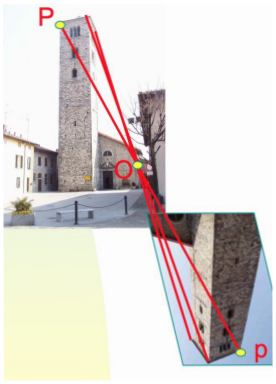
Linear problem

Step 3. Restitution (2D -> 3D): computation of point coordinates of the object.
- impossible from a single image

$(x, y, z) = f^{-1}(\zeta, \eta, \theta)$

Principle of photogrammetric restitution:

If one has at least two images that take the same portion of the object (OVERLAP), and if one reconstructs their position at the time of the acquisition (ORIENTATION), one can compute the object coordinates by INTERSECTION
- With three images one has more <<control>> (more double points, more data, the number of the parameters remain more or less the same => larger redundancy => improve the estimate)



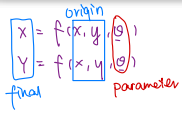
- P = a point on the object
- O = the perspective center
- Collinearity (共线性): P, O and the projection onto the image are on the same straight line

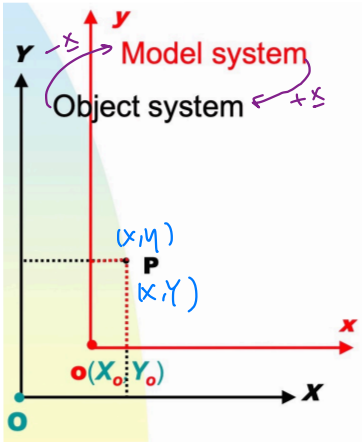
Image Distortion: Unfortunately, the perspective is not exact due to the presence of optical distortions and other effects that introduce additional deformation
- non-perfectly planar support
- atmospheric refraction

Adopted conventions
- Object System: it is related to the taken object and it is the one in which the final coordinates have to be expressed
- Image/model System: it is related to the acquisition sensor (camera, laser scanner, ...)
- Transformation parameters
  * acquisition: Object -> Image/Model
  * restitution: Image/Model -> Object
  * computation:
    1. A sufficient number of DOUBLE POINTS are needed to be known
    2. The parameters are computed as UNKNOWNS of the resulting system of equations, with a redundant number of double points, a Least Squares adjustment is applied.
  * use: after knowing the parameters, it is possible to transform point coordinates from one system to another (Img -> object)

Recording: 01:26:31

2D Transformation





Translation: model system P(x, y) -> object system P(X, Y)

$$\begin{cases} X = X_0 + x \\ Y = Y_0 + y \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \begin{bmatrix} x \\ y \end{bmatrix}$$

matrix notation $\underline{X} = \underline{X}_0 + \underline{x}$

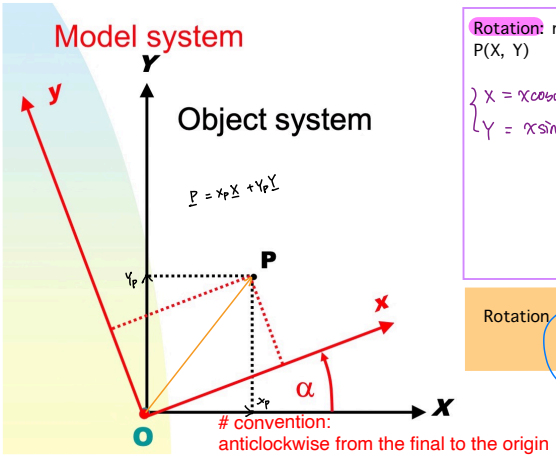
Inverse translation: object system P(X, Y) -> model system P(x, y)

$$\begin{cases} x = X - X_0 \\ y = Y - Y_0 \end{cases} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix} - \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

matrix notation $\underline{x} = \underline{X} - \underline{X}_0$

m = #parameters = 2 => (X0, Y0)

2 equations for one point -> at least one double point is needed (n >= 1)



convention:
anticlockwise from the final to the origin

Rotation: model system P(x, y) -> object system P(X, Y)

$$\begin{cases} X = x\cos\alpha - y\sin\alpha \\ Y = x\sin\alpha + y\cos\alpha \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$\underline{X} = R\,\underline{x}$

Inverse rotation: object system P(X, Y) -> model system p(x, y) // Can be seen as rotation with angle -α

$$\begin{cases} x = X\cos\alpha + Y\sin\alpha \\ y = -X\sin\alpha + Y\cos\alpha \end{cases} \Rightarrow \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}$$

$\underline{x} = R^{-1}\,\underline{X} = R^{T}\,\underline{X}$

m = #parameters = 1 => (rotation angle α)
At least one double point is needed (n >= 1)

Rotation Matrix R is orthonormal: $R = [\underline{r}_1, \underline{r}_2]$

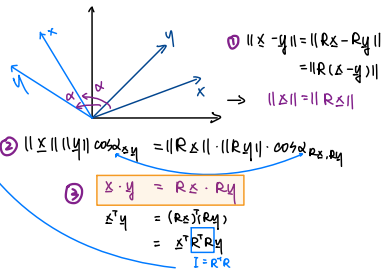
$\sum_i r_{i1}^2 = \sum_i r_{i2}^2 = 1$, $\sum_i r_{i1} r_{i2} = 0$

$\|\underline{r}_1\| = \|\underline{r}_2\| = 1$, $\underline{r}_1 \cdot \underline{r}_2 = 0$

unit vector which orthogonal to each other

$\|\underline{r}_1\|\,\|\underline{r}_2\|\cos\theta = \cos\theta = 0 \Rightarrow \theta = 90°$

$\Leftrightarrow R^{-1} = R^{T}$



Exam question: why it is orthonormal?

Degree of freedom of R
= 4 (elements) - 3 (conditions)
= 1 (independent parameters)

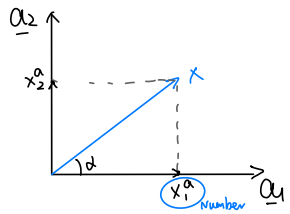
Exam question: why R^T = R^-1

General Rotation from system b to system a

$\underline{a} = [\underline{a}_1, \underline{a}_2, \cdots, \underline{a}_n]$ $\quad \|\underline{a}_i\| = 1, \forall i\ ;\ \underline{a}_i \cdot \underline{a}_j = 0, \forall i,j$

$\underline{b} = [\underline{b}_1, \underline{b}_2, \cdots, \underline{b}_n]$ $\quad \|\underline{b}_i\| = 1, \forall i\ ;\ \underline{b}_i \cdot \underline{b}_j = 0, \forall i,j$

The rotation between a and b.



$\underline{x} = x_1^a\,\underline{a}_1 + x_2^a\,\underline{a}_2$

$x_1^a = \underline{x}\cdot\underline{a}_1 = \|\underline{x}\|\|\underline{a}_1\|\cos\alpha = \|\underline{x}\|\cos\alpha$

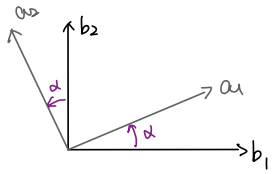
$\underline{x} = (\underline{x}\cdot\underline{a}_1)\underline{a}_1 + (\underline{x}\cdot\underline{a}_2)\underline{a}_2 = \sum_i (\underline{x}\cdot\underline{a}_i)\,\underline{a}_i$ // system a $= \sum_i (\underline{x}\cdot\underline{b}_i)\,\underline{b}_i$ // system b

$\Rightarrow x_i^b = \underline{x}\cdot\underline{b}_i = [\sum_j (\underline{x}\cdot\underline{a}_j)\,\underline{a}_j]\cdot\underline{b}_i$ (x_j^a over $\underline{x}\cdot\underline{a}_j$)

$= \sum_j (x_j^a\,\underline{a}_j)\cdot\underline{b}_i = \sum_j (\underline{a}_j\cdot\underline{b}_i)\,x_j^a$ (r_{ij} over $\underline{a}_j\cdot\underline{b}_i$)

(a) number:

$\begin{cases} x_1^b = r_{11}x_1^a + r_{12}x_2^a \\ x_2^b = r_{21}x_1^a + r_{22}x_2^a \end{cases}$ $\quad R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}$ $\quad \underline{x}^b = R\,\underline{x}^a$



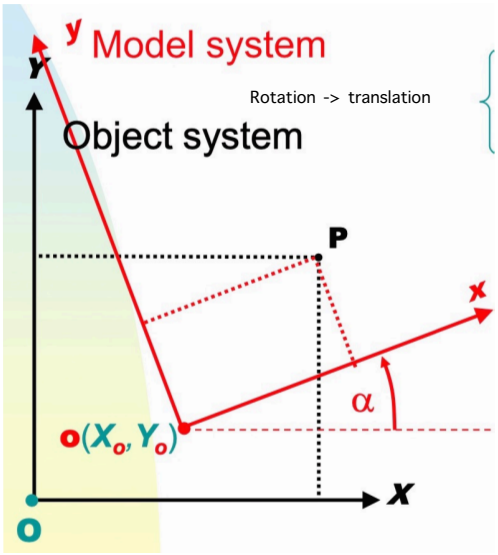
$r_{11} = \underline{a}_1\cdot\underline{b}_1 = \|\underline{a}_1\|\|\underline{b}_1\|\cos\alpha = \cos\alpha$

$r_{12} = \underline{a}_2\cdot\underline{b}_1 = \cos(90° + \alpha) = -\sin\alpha$

$r_{21} = \underline{a}_1\cdot\underline{b}_2 = \cos(90° - \alpha) = \sin\alpha$

$r_{22} = \underline{a}_2\cdot\underline{b}_2 = \cos\alpha$

$\Rightarrow R = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix}$



Roto-translation: model system P(x, y) -> object system P(X, Y)

$$\begin{cases} X = X_0 + x\cos\alpha - y\sin\alpha \\ Y = Y_0 + x\sin\alpha + y\cos\alpha \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

\# rotation -> traslation

Non-linear w.r.t. the parameter
Linear w.r.t the coordinates

$$\underline{X} = \underline{X}_0 + R\,\underline{x}$$

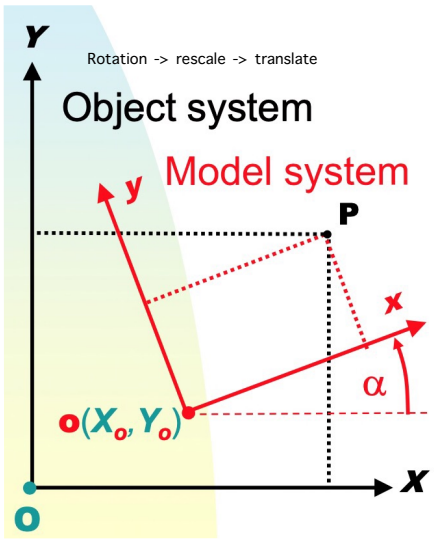
- translation -> rotation: $\underline{X} = R(\underline{X}_0 + \underline{x}) = R\underline{X}_0 + R\underline{x}$

Inverse transformation: object system P(X, Y) -> model system P(x, y)

$$\underline{x} = R^T(\underline{X} - \underline{X}_0)$$

m = #parameters = 3 => (X0, Y0, α) (linear / non-linear)
n = #double points >= 2

Believe
YOURSELF
co



Conformal: model system P(x, y) -> object system P(X, Y)

$$\begin{cases} X = X_0 + m\,x\cos\alpha - m\,y\sin\alpha \\ Y = Y_0 + m\,x\cos\alpha + m\,y\sin\alpha \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix} + m \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\underline{X} = \underline{X}_0 + m\,R\,\underline{x}$$

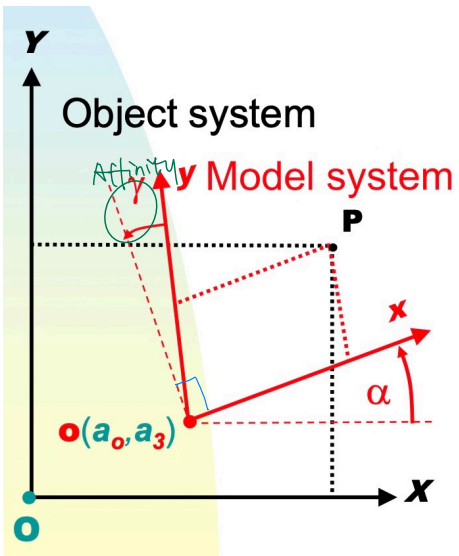
Inverse transformation: object system P(X, Y) -> model system P(x, y)

$$\underline{x} = \frac{1}{m} R^T(\underline{X} - \underline{X}_0)$$

m = #parameters = 4 => (α, m, X0, Y0)
n = #double points >= 2

Non-linear ω.r.t. the parameter
Linear ω.r.t the coordinates

TAKE IT EASY



仿射

Affine: mode system P(x, y) -> object system P(X, Y)

$$\begin{cases} X = a_0 + a_1 x + a_2 y \\ Y = a_3 + a_4 x + a_5 y \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} a_0 \\ a_3 \end{bmatrix} + \begin{bmatrix} a_1 & a_2 \\ a_4 & a_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Translation

$$\underline{X} = \underline{X}_0 + A\,\underline{x}$$

Linear ω.r.t parameters as well as coordinates

Inverse transformation: object system P(X, Y) -> model system P(x, y)

$$\underline{x} = A^{-1}(\underline{X} - \underline{X}_0)$$

m = #parameters = 6
- 2 translations $a_0 = X_0$, $a_3 = Y_0$
- 2 scales $m_x = \sqrt{(a_1^2 + a_4^2)}$, $m_y = \sqrt{a_2^2 + a_5^2}$
- 1 rotation and 1 affinity
  $\alpha = -\operatorname{atan}\frac{a_4}{a_1}$ $\gamma = \operatorname{atan}\frac{a_2}{a_5} - \alpha$

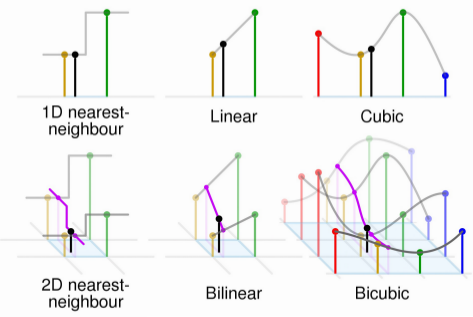
n = #double points >= 3

Bilinear (non-linear transformation) // Cut on x/y => line

$$\begin{cases} X = a_0 + a_1 x + a_2 y + a_3 xy = (b_0 + b_1 x)(b_2 + b_3 y) \\ Y = a_4 + a_5 x + a_6 y + a_7 xy = (b_4 + b_5 x)(b_6 + b_7 y) \end{cases}$$

m = #parameters = 8
n = #double points >= 4

// Non-Linear w.r.t coordinates. Linear w.r.t. parameters

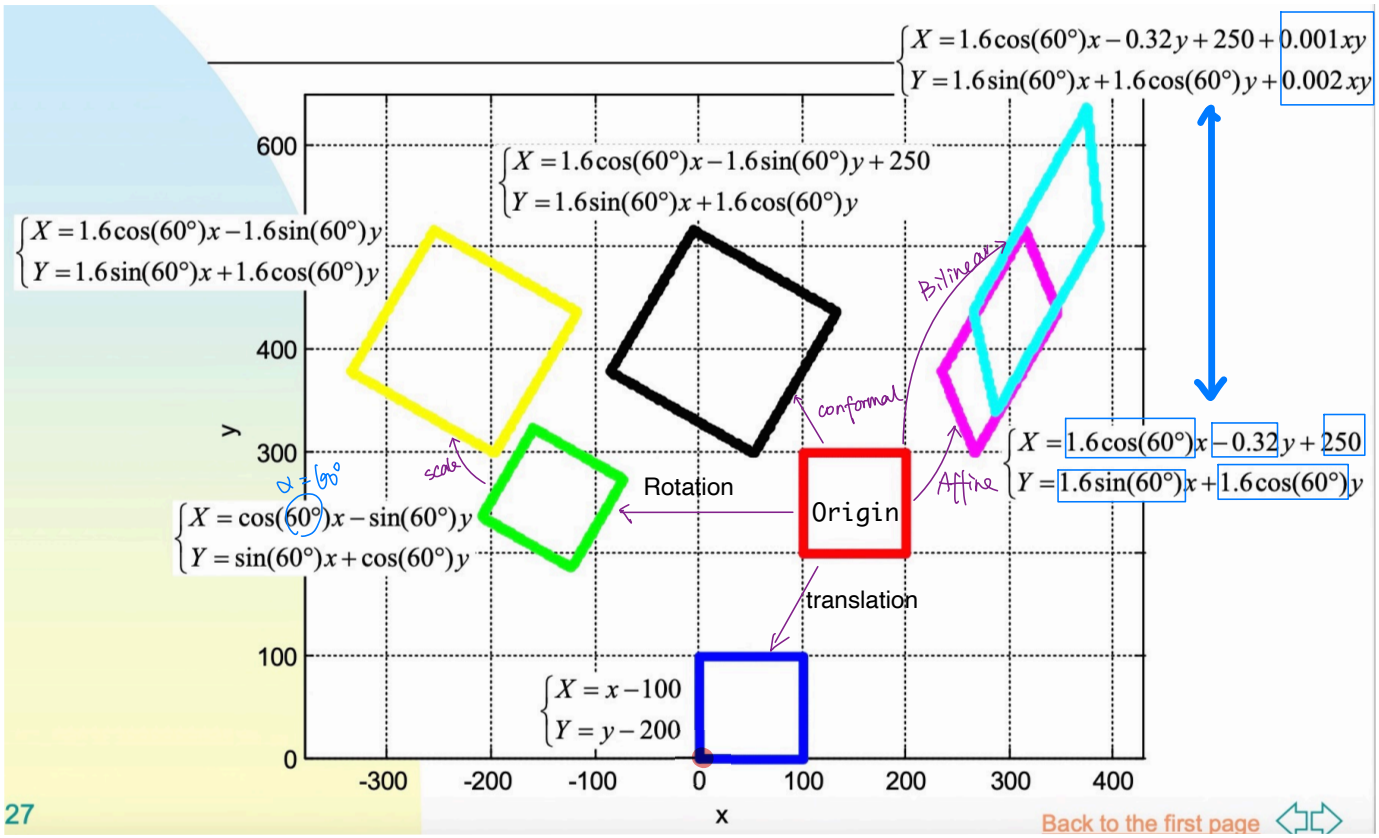


Bicubic (non-linear transformation) // Cut on x/y => cubic

$$\begin{cases} X = (b_0 + b_1 x + b_2 x^2 + b_3 x^3)(b_4 + b_5 y + b_6 y^2 + b_7 y^3) \\ Y = (b_8 + b_9 x + b_{10} x^2 + b_{11} x^3)(b_{12} + b_{13} y + b_{14} y^2 + b_{15} y^3) \end{cases}$$

m = #parameters = 16
n = #double points >= 8

// Non-Linear w.r.t coordinates. Linear w.r.t. parameters

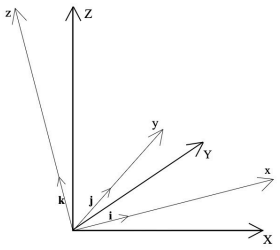


Back to the first page

2021.09.24 3D transformations; Cardan angles; Orientation parameters

// Recording begin from
00:32:30

3D Transformations



$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_o \\ Y_o \\ Z_o \end{bmatrix} + m\,R \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \Leftarrow \quad \begin{cases} X = X_o + m(r_{11}x + r_{12}y + r_{13}z) \\ Y = Y_o + m(r_{21}x + r_{22}y + r_{23}z) \\ Z = Z_o + m(r_{31}x + r_{32}y + r_{33}z) \end{cases}$$

Object coordinates = 3 translations + Isotropic scale factor × Spatial rotation matrix [3x3] × Model coordinates

Linear transformation w.r.t. the coordinates

- m = #parameters = 3 (translations) + 1 (scale) + 3 (degree of freedom of R) = 7
- n = #double points >= 7/3 = 3

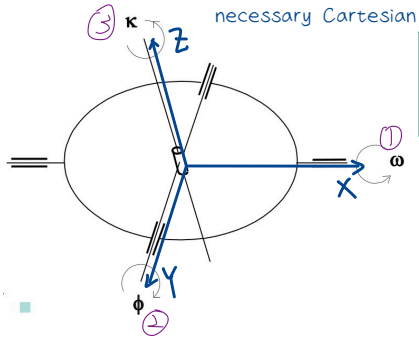
$$R = \begin{bmatrix} \cos\kappa X & \cos\varphi X & \cos\omega X \\ \cos\kappa Y & \cos\varphi Y & \cos\omega Y \\ \cos\kappa Z & \cos\varphi Z & \cos\omega Z \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Properties of R:
- orthonormal matrix $R^{-1} = R^T$
  - * normality: $r_i^T r_i = 1, \forall i \Leftrightarrow \|r_i\| = 1, \forall i$
  - * orthogonality: $r_i^T r_j = 0, \forall i \neq j$
- $3 + 3 = 6$ conditions
- degree of freedom of R = 9 (elements) - 6 (conditions) = 3 (independent parameters)

How to parametrization of the 3D rotation matrix:
- direction cosines of the coordinate axes
- components of the unit vectors of the coordinate axes
- Cardan angles (typically used in photogrammetry)
- other types of angles (Euler, ...)
- algebraic parametrizations (Rodriguez, ...)

Cardan angles

These axes are not necessary Cartesian axes.



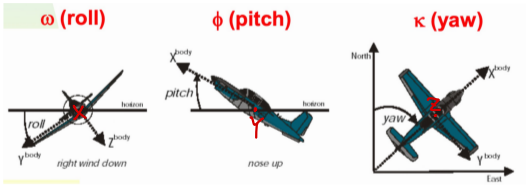
3 sequential rotations around 3 axes:
- primary rotation around x (ω)
- secondary rotation around y (φ)
- tertiary rotation around z (κ)

Around which the angle ω, φ and κ are applied?

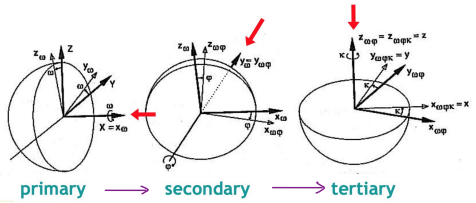
Draw typically Cartesian axes, then draw ω around X (Y->Y', Z-> Z'), φ around Y' (Z -> Z'') and then κ around Z''.

Note that when you change the order of rotation, you get different result.

Cardan rotations correspond to the 3 possible rotations of an aircraft and therefore of a photo acquired from a camera which is rigidly fixed to it.



Convention: rotations are considered as positive if they are ANTICLOCKWISE when looking through the axis towards the origin (arrow -> origin).

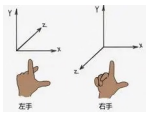


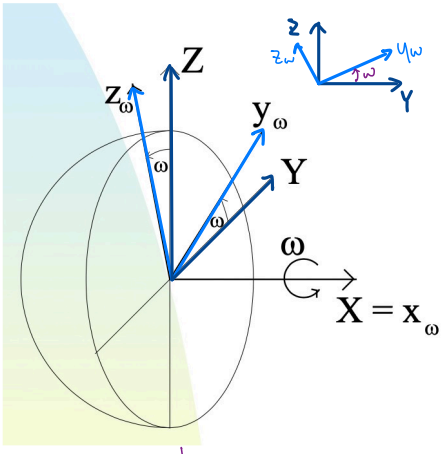
$$R = R(\omega, \varphi, \kappa) = \text{combination of } \{R_\omega^{2D}, R_\varphi^{2D}, R_\kappa^{2D}\} = R_\kappa R_\varphi R_\omega$$

$$\rightarrow \underline{X}' = R_\omega \underline{X} \rightarrow \underline{X}'' = R_\varphi \underline{X}' = R_\varphi (R_\omega \underline{X}) \rightarrow \underline{X}''' = R_\kappa \underline{X}'' = R_\kappa (R_\varphi (R_\omega \underline{X}))$$

Again, convention: the angle is positive in the transformation from final system (X) to origin system (x)

右手坐标系：

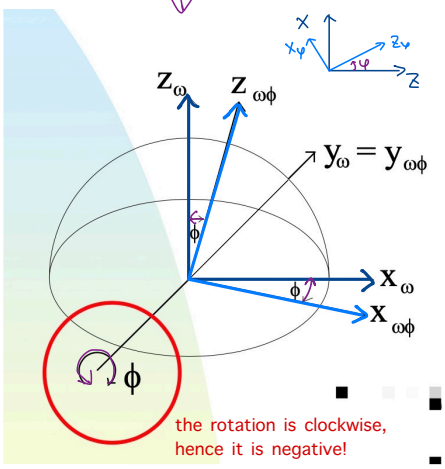




Primary Rotation R_ω

$$\begin{cases} X = x \\ Y = y\cos\omega - z\sin\omega \\ Z = y\sin\omega + z\cos\omega \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

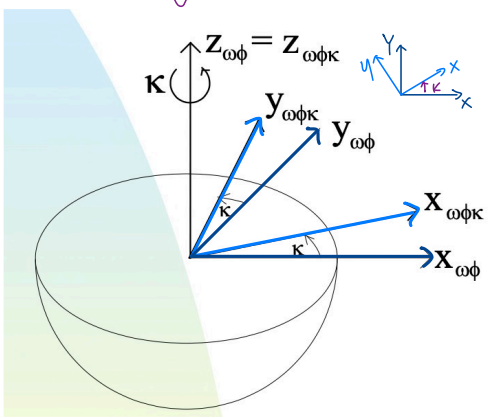
$$\underline{X} = R_\omega \, \underline{x}$$



Secondary Rotation R_φ

$$\begin{aligned} Z &= z\cos\varphi - x\sin\varphi \\ X &= z\sin\varphi + x\cos\varphi \\ Y &= y \end{aligned} \Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\underline{X} = R_\varphi \, \underline{x}$$



Tertiary Rotation R_κ

$$\begin{cases} X = x\cos\kappa - y\sin\kappa \\ Y = x\sin\kappa + y\cos\kappa \\ Z = z \end{cases} \Rightarrow \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

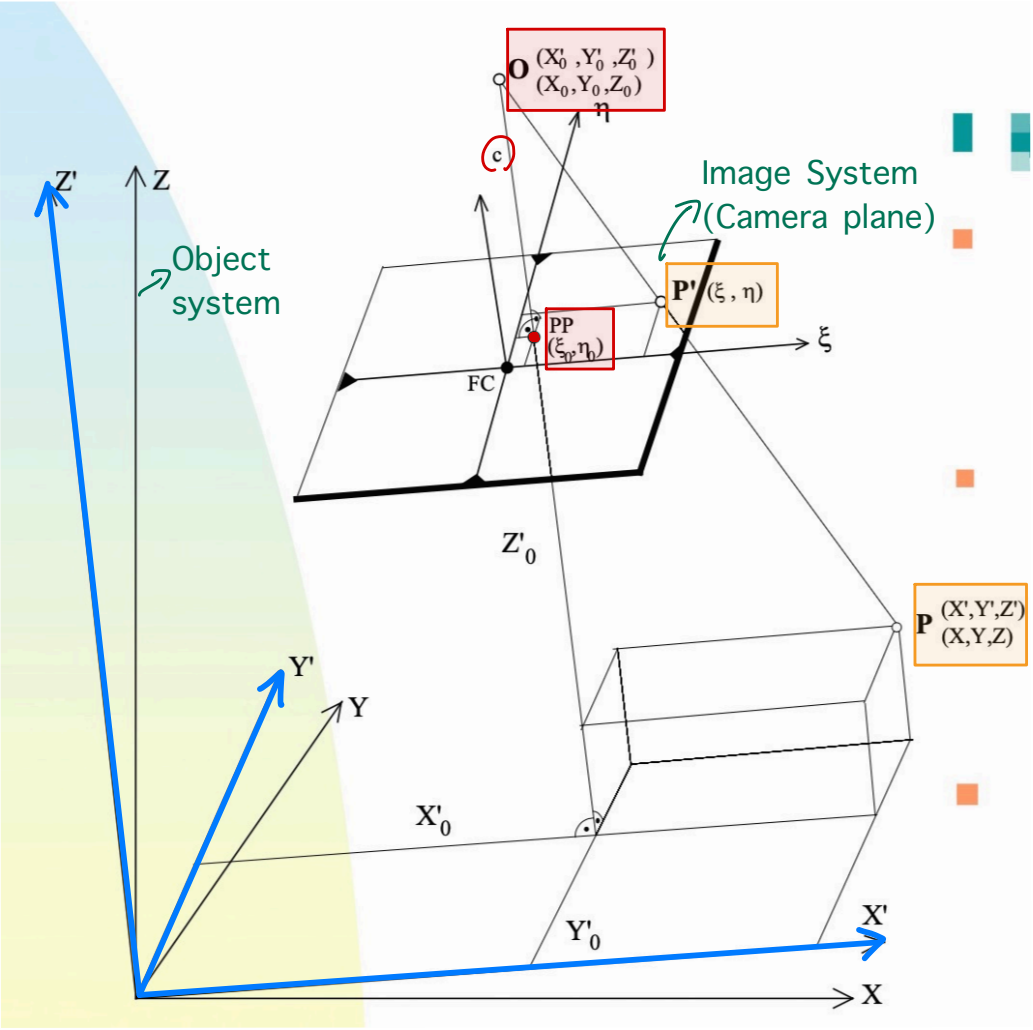
$$\underline{X} = R_\kappa \, \underline{x}$$

The final rotation matrix (formed from the 3 single rotation matri

$$R = R_\kappa R_\varphi R_\omega = \begin{bmatrix} \cos\kappa & -\sin\kappa & 0 \\ \sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & \sin\varphi \\ 0 & 1 & 0 \\ -\sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & -\sin\omega \\ 0 & \sin\omega & \cos\omega \end{bmatrix}$$

Analytical formulation of the central projection

// recording: 01:39:00



Parameters of collinearity equation $(\xi, \eta) = \underline{f}(X, Y, Z, \underline{\vartheta})$ // 2 equations

- 3 parameters of internal orientation (model of the camera)
  - * principal distance (distance between the projection center O and the image plane): c
  - * principal point (intersection between the center projection to the image plane): PP (ξ_0, η_0) in the image system
    - + Note that do not mix the principal point and the origin center of the image system
    - + In order to determine the coordinate of the principle point, you need to know also the size of the pixel

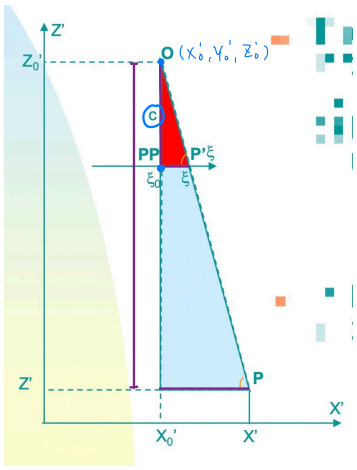
- 6 parameters of external orientation (where is the camera) (position + attitude: orientation of the camera)
  - * 3 Cardan angles (between the object system and image system): ω, φ, κ (rotation from image RS to object RS)
  - * 3 coordinates of the projection (taken) center O (X0, Y0, Z0) in object RS

m = #double points (ground control points) >= 9/2 = 5

Collinearity Equations

$(\xi, \eta) = f(X, Y, Z, \underline{\theta})$

$\underline{\theta} = c, \xi_0, \eta_0$ + X_0, Y_0, Z_0 + ω, φ, κ (position + attitude)



Consider an auxiliary RS (X', Y', Z') that is parallel to the image system (ξ, η, ζ)

The conditions of collinearity in the plane X'Z' are written by exploiting the similarity between the red and cyan triangles:

$$\frac{\xi - \xi_0}{c} = \frac{X' - X_0'}{Z_0' - Z'}$$

In the same way, the conditions in the plane Y'Z' can be written:

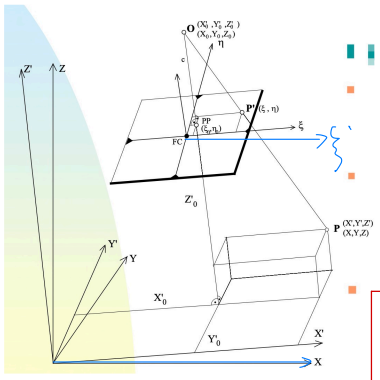
$$\frac{\eta - \eta_0}{c} = \frac{Y' - Y_0'}{Z_0' - Z'}$$

$$\Rightarrow \begin{cases} \xi = \xi_0 + c\,\dfrac{X' - X_0'}{Z_0' - Z'} \\ \eta = \eta_0 + c\,\dfrac{Y' - Y_0'}{Z_0' - Z'} \end{cases}$$

collinearity equations in the case when the two systems are aligned:

$(\xi, \eta) = f(X', Y', Z', \underline{\theta})$ (Image)

EQ①



The RS(X', Y', Z') is connected to the object RS (X, Y, Z) by means of the rotation matrix R:
// convention: rotation from the image (ξ, η, ζ) to the object (X, Y, Z)

$$\begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix} = R \begin{bmatrix} X' - X_0' \\ Y' - Y_0' \\ Z' - Z_0' \end{bmatrix} \quad R(\omega, \varphi, \kappa)$$

$$\Leftarrow \begin{cases} \underline{X} = R\,\underline{X}' \\ \underline{X}_0 = R\,\underline{X}_0' \end{cases} \quad \underline{X} - \underline{X}_0 = R\,(\underline{X}' - \underline{X}_0')$$

// include the central projection

$$\underline{X} - \underline{X}_0 = R\,(\underline{X}' - \underline{X}_0') \longrightarrow \underline{X}' - \underline{X}_0' = R^T(\underline{X} - \underline{X}_0)$$

EQ②

Note that here, R is the rotation matrix from the image RS to the object RS

EQ① + EQ②

$$\xi = \xi_0 - c\,\frac{r_{11}(X - X_0) + r_{21}(Y - Y_0) + r_{31}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$

$$\eta = \eta_0 - c\,\frac{r_{12}(X - X_0) + r_{22}(Y - Y_0) + r_{32}(Z - Z_0)}{r_{13}(X - X_0) + r_{23}(Y - Y_0) + r_{33}(Z - Z_0)}$$

Collinearity equations

$\Rightarrow (\xi, \eta) = f(X, Y, Z, \underline{\theta})$

// Orientation form

Input: double points (at least 5 points because of 9 parameters)
Method: Least Square Adjustment for Nonlinear function (linearization or over-parametrization) $\Rightarrow \hat{\underline{\theta}}$

Inverting the equations (restitution form) =>

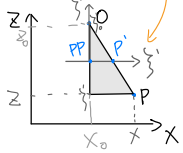
$$X = X_0 + (Z - Z_0)\,\frac{r_{11}(\xi - \xi_0) + r_{12}(\eta - \eta_0) - r_{13}c}{r_{31}(\xi - \xi_0) + r_{32}(\eta - \eta_0) - r_{33}c}$$

$$Y = Y_0 + (Z - Z_0)\,\frac{r_{21}(\xi - \xi_0) + r_{22}(\eta - \eta_0) - r_{23}c}{r_{31}(\xi - \xi_0) + r_{32}(\eta - \eta_0) - r_{33}c}$$

$(X, Y, Z) = f^{-1}(\xi, \eta, \hat{\underline{\theta}})$

// Note that in the same convention as orientation, but with the inverse procedure.

① Define a auxiliary image system (ξ', η', ζ') which parallel to the object system



② $\frac{\xi' - \xi_0'}{\zeta_0' - \zeta'} = \frac{X - X_0}{Z_0 - Z}$, $\frac{\eta' - \eta_0'}{\zeta_0' - \zeta'} = \frac{Y - Y_0}{Z_0 - Z}$

③ $\underline{\xi}' = R\,\underline{\xi}$

Aux.Object ← Image

Step for orientation form: $(\xi, \eta) = f(X, Y, Z, \underline{\theta})$
1. Define an Object auxiliary system
2. Triangle similarity leading to collinearity equations w.r.t. auxiliary object (image v.s. auxiliary)
3. Rotation between auxiliary object and object
4. Substitution into 2)

Step for restitution form: $(X, Y, Z) = f^{-1}(\xi, \eta, \underline{\theta})$
1. Define an image auxiliary system
2. Triangle similarity leading to collinearity equations w.r.t. auxiliary image (object v.s. auxiliary)
3. Rotation between auxiliary image and image
4. substitution into 2)

Why $(\zeta - \zeta_0 = -c)$?

The coordinate of projection center O in Image System is (ξ_0, η_0, ζ_0), and ζ stays on the plane. (PP)

Since the convention used is the object is above

Use of the collinearity equations:
- ORIENTATION
  - * computation of the 6 parameters of the external orientation:
    - + #unknowns = 6 per image
    - + for each measured point in both the object and image coordinates (**ground control point - GCP**), 2 equations can be written
    - + with #GCP >= 3 points, it can be written a No. of equations >= 3*2=6, and compute the 6 unknown parameters
  - * note that the internal orientation parameters are usually known, except for the case of the so-called AUTOCALIBRATION
- RESTITUTION
  - * #unknowns = 3 (X,Y,Z)
  - * given the 6 external orientation parameters and measuring the image coordinates (ξ, η) of the same point (homologous) at least on 2 images, it is possible to write at least 4 collinearity equations and compute its object coordinates.

$\hat{\underline{\theta}}$

The coordinates of projection center 0:

$O(X_0, Y_0, Z_0)$ Object System

$O(X_0', Y_0', Z_0')$ Auxiliary object System

$O(\xi_0', \eta_0', \zeta_0')$ Auxiliary Image System

$O(\xi_0, \eta_0, \zeta_0)$ Image System

$(\xi, \eta) = f(X, Y, Z, \underline{\theta})$ → physical → Collinearity Equation (# paras = 9)
→ algebric (linear) → DLT (# paras = 11)

DLT
(Direct Linear Transformation)

// over-parameterization

$Z=0$

$$\begin{cases} \xi = \dfrac{a_1 X + a_2 Y + a_3 Z + a_4}{b_1 X + b_2 Y + b_3 Z + 1} \\ \eta = \dfrac{a_5 X + a_6 Y + a_7 Z + a_8}{b_1 X + b_2 Y + b_3 Z + 1} \end{cases}$$

Why "1" here?

Otherwise, we have sets of coefficients which produce the same result. Since we want to estimate the coefficients, we want to estimate just one set of coefficients instead of infinite number of sets

$a_1 = \xi_0 r_{13} - c r_{11}$, $a_2 = \xi_0 r_{23} - c r_{21}$, $a_3 = \xi r_{33} - c r_{31}$, $a_4 = -\xi_0 r_{13} X_0 + c r_{11} X_0 - \xi_0 r_{23} Y_0 + c r_{21} Y_0 - \xi_0 r_{33} Z_0 + c r_{31} Z_0$

n = #parameters = 11
// the two additional degrees of freedom can be used to model the scale factors along ξ and η

m = #double points >= 11/2 = 6

$$\begin{cases} 0 = a_1 X + a_2 Y + a_3 Z + a_4 - \xi b_1 X - \xi b_2 Y - \xi b_3 Z - \xi \\ 0 = a_5 X + a_6 Y + a_7 Z + a_8 - \eta b_1 X - \eta b_2 Y - \eta b_3 Z - \eta \end{cases}$$

// linear w.r.t. the parameters

Given $P_1 (X_1, Y_1, Z_1)$, (ξ_1, η_1)
$P_2 (X_2, Y_2, Z_2)$, (ξ_2, η_2)

$$\Rightarrow A_{[11\times 11]} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 & b_1 & b_2 & b_3 & a_5 & a_6 & a_7 & a_8 \\ X_1 & Y_1 & Z_1 & 1 & -\xi X_1 & -\xi Y_1 & -\xi Z_1 & 0 & 0 & 0 & 0 \\ & & & & & & & & & & \\ & & & & & & & & & & \end{bmatrix}$$

$$\underline{a} = \begin{bmatrix} -\xi_1 \\ -\eta_1 \\ -\xi_2 \\ -\eta_2 \\ \vdots \end{bmatrix} , \quad \underline{Y}_0 = \underline{0}_{N\times 1}$$

LS → $\hat{a}_1, \hat{a}_2, \cdots, \hat{b}_1, \hat{b}_2 \cdots$ → Approximated physical $\hat{\underline{\theta}}$

Homography

// a particular case of DLT (with Z=0) and represents the central projection between two planes.

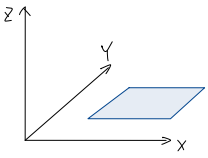
Homography is used to correct the perspective effect when the object is "flat" (for example, the facade of a building)

DLT + "Z=0" →

$$\begin{cases} \xi = \dfrac{a_1 X + a_2 Y + a_3}{b_1 X + b_2 Y + 1} \\ \eta = \dfrac{a_4 X + a_5 Y + a_6}{b_1 X + b_2 Y + 1} \end{cases}$$

n = #parameters = 8
m = #double points >= 8/2 = 4

// linear transformation from 2D to 2D



Photoplane

// assume: 2D object
// $\xi_0 = \eta_0 = Z = 0$

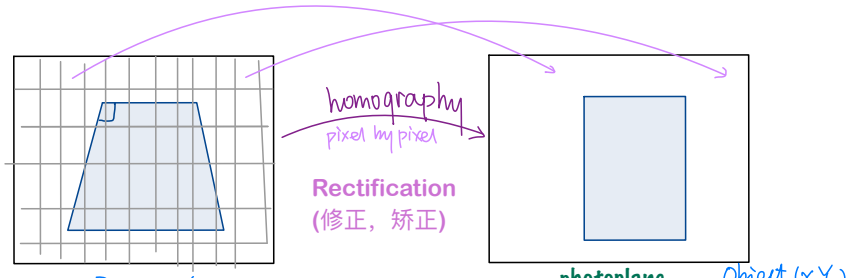


Image (ξ, η) photoplane Rectified Object (X, Y)

$$\begin{aligned} X &= X_0 - Z_0 \frac{r_{11}\xi + r_{12}\eta - r_{13}c}{r_{31}\xi + r_{32}\eta - r_{33}c} \\ Y &= Y_0 - Z_0 \frac{r_{21}\xi + r_{22}\eta - r_{23}c}{r_{31}\xi + r_{32}\eta - r_{33}c} \end{aligned}$$

Collinearity Equation
(restitution form)

$$\begin{aligned} X &= \frac{a_1\xi + a_2\eta + a_3}{b_1\xi + b_2\eta + 1} \\ Y &= \frac{a_4\xi + a_5\eta + a_6}{b_1\xi + b_2\eta + 1} \end{aligned}$$

Homography equation

n = #parameters = 8
m = #double points >= 8/2 = 4

The survey can be performed with a single image

Determination of the homography parameters starting from double points:

$$\left.\begin{aligned}(b_1\xi_i + b_2\eta_i + 1)\,X_i &= a_1\xi_i + a_2\eta_i + a_3\\ (b_1\xi_i + b_2\eta_i + 1)\,Y_i &= a_4\xi_i + a_5\eta_i + a_6\end{aligned}\right\} \Rightarrow \left\{\begin{aligned}b_1\xi_i X_i + b_2\eta_i X_i + X_i - a_1\xi_i - a_2\eta_i - a_3 &= 0\\ b_1\xi_i Y_i + b_2\eta_i Y_i + Y_i - a_4\xi_i - a_5\eta_i - a_6 &= 0\end{aligned}\right.$$

In matrix notation, it is possible to write a system as follows:

$$\begin{bmatrix} -\xi_1 & -\eta_1 & -1 & 0 & 0 & 0 & \xi_1\bar{X}_1 & \eta_1\bar{X}_1 \\ 0 & 0 & 0 & -\xi_1 & -\eta_1 & -1 & \xi_1\bar{Y}_1 & \eta_1\bar{Y}_1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -\xi_i & -\eta_i & -1 & 0 & 0 & 0 & \xi_i\bar{X}_i & \eta_i\bar{X}_i \\ 0 & 0 & 0 & -\xi_i & -\eta_i & -1 & \xi_i\bar{Y}_i & \eta_i\bar{Y}_i \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -\xi_n & -\eta_n & -1 & 0 & 0 & 0 & \xi_n\bar{X}_n & \eta_n\bar{X}_n \\ 0 & 0 & 0 & -\xi_n & -\eta_i & -1 & \xi_n\bar{Y}_n & \eta_n\bar{Y}_n \end{bmatrix} \begin{Bmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \end{Bmatrix} = \begin{Bmatrix} -\bar{X}_1 \\ -\bar{Y}_1 \\ \dots \\ \dots \\ -\bar{X}_i \\ -\bar{Y}_i \\ \dots \\ \dots \\ -\bar{X}_n \\ -\bar{Y}_n \end{Bmatrix}$$

$$\mathbf{A}\vec{x} = \vec{b}$$

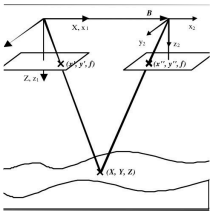
its least-squares solution is:

$$\vec{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\vec{b} \quad \Rightarrow \quad \hat{\underline{x}} = [\hat{a}_1, \hat{a}_2 \dots, \hat{b}_1, \hat{b}_2]^T$$

$$\left\{\begin{aligned} X_i &= \frac{\hat{a}_1\xi_i + \hat{a}_2\eta_i + \hat{a}_3}{\hat{b}_1\xi_i + \hat{b}_2\eta_i + 1}\\ Y_i &= \frac{\hat{a}_4\xi_i + \hat{a}_5\eta_i + a_6}{\hat{b}_1\xi_i + \hat{b}_2\eta_i + 1}\end{aligned}\right.$$

Step:
1. Acquisition: an image is acquired and GCPs are measured (by a total station)
2. Orientation
3. Restitution -> (X, Y, Z) for all image points

// by means of the estimated parameters, it is possible to remove the perspective deformation of the image



// with two images, the distance between the object and the image can be estimated

Normal case is a simplified case that allows for a stereoscopic version (3D) of the photos.
- GOAL: write analytically the restitution equations (in a simpler way).
- It is useful to compute the precision obtainable in the restitution of points as a function of the different parameters.(to design the survey)

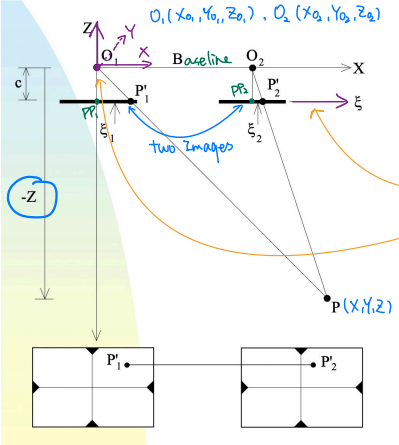
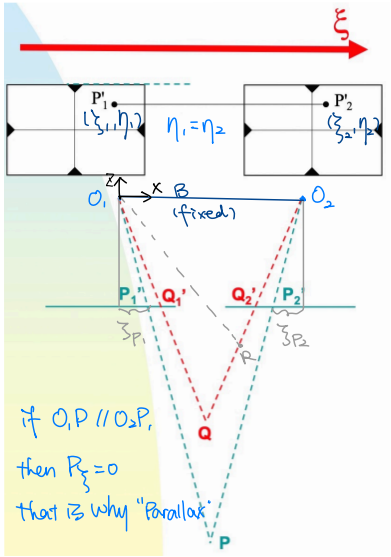


Image configuration:
- basic:
  * image planes // baseline and image planes are not rotated one and other. $\omega_1 = \varphi_1 = \kappa_1 = 0$, $\omega_2 = \varphi_2 = \kappa_2 = 0$ $\Rightarrow R = I$
  * the cameras are the same $(c_1 = c_2 = c)$
- additional
  * define the image system such that ξ parallel to the baseline
  * define the object system such that center in the central projection of the first image with X axis oriented aligned with the baseline and Z axis parallel to the optical axis (not mandatory, just a convenient choice) $X_{O_1} = Y_{O_1} = Z_{O_1} = 0$ $X_{O_2} = B$, $Y_{O_2} = Z_{O_2} = 0$
- assumption:
  * PP1 and PP2 corresponds to the center of the image planes (to simply the problem): $\xi_{O_1} = \xi_{O_2} = 0$, $\eta_{O_1} = \eta_{O_2} = 0$



If two images are acquired by a camera translated along a direction which is parallel to one of their sides (normal or nadiral images), the same point appears at two different positions (P1', P2').

The **parallax** along ξ is defined as $P_\xi = \xi_2 - \xi_1$

// $P_\eta = 0$

- relate to the distance: the closer is the point w.r.t the camera, the larger is the parallax
  * larger parallax -> smaller Z (note that Z is negative) -> close to the image plane (see below)

Stereoscopic vision:

If the two images of a couple of normal images are separately seen from each eye, it is possible to have a 3D perception. This is due to the fact that the points with larger parallax are seen as closer than the other ones (with a smaller parallax). This is the principle of the human 3D vision, where however other cognitive capabilities also play a role)

computation of the object coordinates

$(X, Y, Z) = f(\xi_1, \xi_2, \eta_1, \eta_2, B, c)$

// in a analytical way

$$\begin{cases} X = X_0 + (Z - Z_0)\dfrac{r_{11}(\xi - \xi_0) + r_{12}(\eta - \eta_0) - r_{13}c}{r_{31}(\xi - \xi_0) + r_{32}(\eta - \eta_0) - r_{33}c} \\ Y = Y_0 + (Z - Z_0)\dfrac{r_{21}(\xi - \xi_0) + r_{22}(\eta - \eta_0) - r_{23}c}{r_{31}(\xi - \xi_0) + r_{32}(\eta - \eta_0) - r_{33}c} \end{cases}$$

Collinearity Equation

With the image configuration

Image Sys. 1

$$X = 0 + (Z - 0)\frac{1\cdot(\xi_1 - 0) + 0\cdot(\eta_1 - 0) - 0\cdot c}{0(\xi_1 - 0) + 0\cdot(\eta_1 - 0) - 1\cdot c}$$

$$Y = 0 + (Z - 0)\frac{0(\xi_1 - 0) + 1\cdot(\eta_1 - 0) - 0\cdot c}{0(\xi_1 - 0) + 0\cdot(\eta_1 - 0) - 1\cdot c}$$

$\Rightarrow$

$$X = Z\cdot\frac{\xi_1}{-c}$$

$$Y = Z\cdot\frac{\eta_1}{-c}$$

$\Rightarrow$

Image System 2

$$X = B + (Z - 0)\frac{1\cdot(\xi_2 - 0) + 0\cdot(\eta_2 - 0) - 0\cdot c}{0(\xi_2 - 0) + 0\cdot(\eta_2 - 0) - 1\cdot c}$$

$$Y = 0 + (Z - 0)\frac{0\cdot(\xi_2 - 0) + 1\cdot(\eta_2 - 0) - 0\cdot c}{0\cdot(\xi_2 - 0) + 0\cdot(\eta_2 - 0) - 1\cdot c}$$

$\Rightarrow$

$$X = B + Z\cdot\frac{\xi_2}{-c}$$

$$Y = Z\cdot\frac{\eta_2}{-c}$$

$$Z = \frac{B\cdot c}{\xi_2 - \xi_1} = \frac{B\cdot c}{P_\xi}$$
$$X = -\frac{B\cdot c}{\xi_2 - \xi_1}\cdot\frac{\xi_1}{c}$$
$$Y = -\frac{B\cdot c}{\xi_2 - \xi_1}\cdot\frac{\eta_1}{c}$$

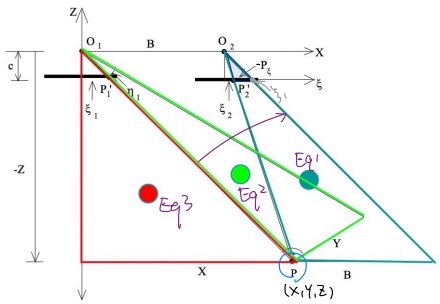
$(X,Y,Z) = f(\xi_1, \xi_2, \eta_1, \eta_2, B, c)$

Reduce the required parameters [3 (internal, same cameras) + 2 (image system) * 6 (external) = 15]
to 2 (B+c) => Simpler model!! So that it can be used to design the survey (the velocity -> B)!!

15 → 2

// in practice, determine the value of parameters based on the required accuracy

// in a geometrical way



$$\frac{B}{-P_\xi} = \frac{-Z}{c} \Rightarrow Z = \frac{Bc}{P_\xi}$$
$$\frac{X}{\xi_1} = \frac{-Z}{c} \Rightarrow X = -Z\cdot\frac{\xi_1}{c}$$
$$\frac{Y}{\eta_1} = \frac{-Z}{c} \Rightarrow Y = -Z\cdot\frac{\eta_1}{c}$$

computation of the precision

$y = ax + b \Rightarrow \sigma_y^2 = a^2\sigma_x^2$
$y = g(x) \Rightarrow \sigma_y^2 = \left(\frac{\partial g}{\partial x}\Big|_{\bar{x}}\right)^2\sigma_x^2$
$Z = X + Y \Rightarrow \sigma_Z^2 = \sigma_X^2 + \sigma_Y^2 + 2\sigma_{XY}$

By applying the variance propagation law to the expression of the point coordinates, their standard deviation can be computed.

// B and c are considered as errorless (fixed for the moment)

- Coordinate Z: starting from the expression of its mean value: $Z = \frac{Bc}{P_\xi} = g(P_\xi)$

  * by applying the covariance propagation law:

$$\sigma_Z^2 = \left(\frac{\partial g}{\partial P_\xi}\right)^2\sigma_{P_\xi}^2 = \left(\frac{Bc}{P_\xi^2}\right)^2\cdot\sigma_{P_\xi}^2 = \left(\frac{Z^2}{Bc}\right)^2\cdot 2\sigma_\xi^2$$

- Coordinate X: starting from the expression of its mean value: $X = -Z\cdot\frac{\xi_1}{c} = f(Z, \xi_1)$

  * by applying the covariance propagation law:

$$\sigma_X^2 = \left(\frac{\partial f}{\partial Z}\right)^2\sigma_Z^2 + \left(\frac{\partial f}{\partial \xi_1}\right)^2\sigma_{\xi_1}^2 = \left(\frac{\xi_1}{c}\right)^2\sigma_Z^2 + \left(\frac{Z}{c}\right)^2\sigma_{\xi_1}^2 = \left(\frac{\xi_1}{c}\cdot\frac{Z^2}{Bc}\right)^2\sigma_{P_\xi}^2 + m_b^2\sigma_{\xi_1}^2 = \left(\frac{\xi_1}{c}\cdot m_b\cdot\frac{Z}{B}\cdot\sigma_{P_\xi}\right)^2 + (m_b\cdot\sigma_{\xi_1})^2$$

m_b: scale of the image (bild in Germane)

Baseline ratio
// longer B is better

- Coordinate Y: similar to X. $Y = -Z\cdot\frac{\eta_1}{c} = f(Z, \eta_1)$

In summary:

$$\sigma_{e_X}: \sigma_X = \pm\sqrt{\left(\frac{\xi_1}{c}m_b\frac{Z}{B}\sigma_{P_\xi}\right)^2 + (m_b\cdot\sigma_{\xi_1})^2} = \pm m_b\sigma_\xi\cdot\sqrt{1 + 2\frac{\xi_1^2}{c^2}\cdot\frac{Z^2}{B^2}}$$
$$\sigma_{e_Y}^2: \sigma_Y = \pm\sqrt{\left(\frac{\eta_1}{c}m_b\frac{Z}{B}\sigma_{P_\xi}\right)^2 + (m_b\cdot\sigma_\eta)^2} = \pm m_b\sigma_\eta\cdot\sqrt{1 + 2\frac{\eta_1^2}{c^2}\cdot\frac{Z^2}{B^2}}$$
$$\sigma_{e_Z}^2: \sigma_Z = \pm\frac{Z^2}{c\cdot B}\sigma_{P_\xi} = \pm\sqrt{2}\frac{Z}{c}\frac{Z}{B}\sigma_\xi = \pm m_b\sigma_\xi\cdot\sqrt{2}\frac{Z}{B} \quad (m_b)$$

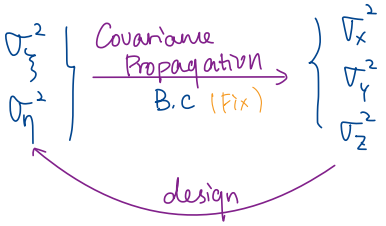
// $\sigma_Z^2 = \sigma_{\xi_1}^2 + \sigma_{\xi_2}^2 - 2\sigma_{\xi_1,\xi_2} \approx 2\sigma_\xi^2$ (Assume 0)

Having these propagation and Knowing the requirement of a survey,
We can calibrate the parameters without doing complicated computations

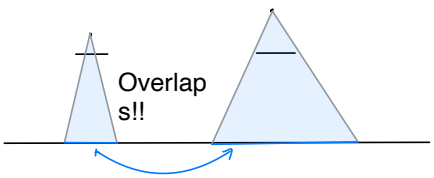
Considerations on the precision of the restitution points:
- for the same base ratio, the RMS of all coordinates are directly proportional to the photo scale
- for the same photo scale, the RMS of the Z coordinates are inversely proportional to the base ratio
- for the same base length B, the RMS of the Z coordinates increase as the square of Z (distance between camera and object)

If you want to have better performance, with fixed baseline ratio, consider $m_b = \frac{Z}{c}$)
- decrease Z => fly lower (the lower you fly, the closer between you and the target)
- increase c => increase the objective

Note that in photogrammetry, we need overlaps, so that we cannot choose B and c without considering overlaps. That is, when designing the survey, we need to consider B, c, Z and the overlaps. To have a better accuracy, decrease Z, increase c provided good overlaps!!!



Example: given Z, , c, we can calculate B. And then based of the fly velocity v, we can get the interval of taken photo. So that we can design the survey at very beginning to save the effort and MONEY!



🌻 Normal case is not the only way you have to design the survey. Normal case is useful when you have a drone or a aerial plane.

empirical formulas

// formulas come from experience or institution

Shape factor: 0.4 ~ 1.0 (depending on the complication of the surface)

Calibration error

$$\sigma_{x,y,z} = \sigma_{rest} = \frac{w_b \cdot \sqrt{2}}{\sqrt{k}} \sigma_0$$

restitution

Number of photos

<= make no distinguish between directions (X, Y, Z)

$$\hat{x} = \frac{x_1 + x_2 \cdots + x_N}{N}$$

$$\hat{\sigma}^2 = \frac{\sigma_{x_1}^2 + \sigma_{x_2}^2 + \cdots + \sigma_{x_N}^2}{N^2} = \frac{N\sigma_x^2}{N^2} = \frac{\sigma_x^2}{N} \Rightarrow \hat{\sigma} = \frac{\sigma_x}{\sqrt{N}}$$

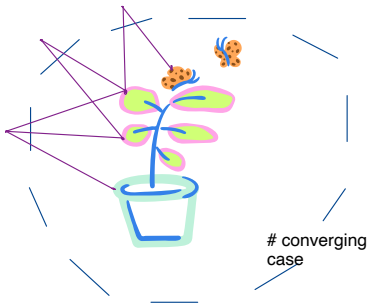
	σ_x
x_1	σ_{x_1}
x_2	σ_{x_2}
⋮	⋮
x_N	σ_{x_N}

TAKE IT EASY

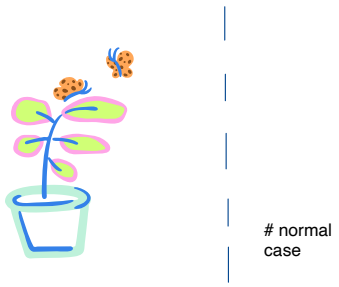
Simulation

// simulate the observations (double points)

// more complicated, but general



converging case



normal case

Two possibilities to design the survey.

Which one is better? How to estimate? By LS with simulated double points!!

$y_0 = y + v$ // Image Point

y ↑ collinearity equations

X = orientation parameters

Z = model = $f(x)$

$$\Rightarrow C_{xx} = \hat{\sigma}_0^2 (A^T A)^{-1}$$

Camera locations

Collination error σ_0

In summary,
move your cameras, click an button (to solve LS Adjustment, in fact, to write the design matrix), get the accuracy of the model.

// less computation!

Conclusion: in order to design the survey, you have basically three possibilities:
- if you are flying in a vertical way with a drone, you can use the normal case and covariance propagation
- if you are using the photogrammetry that close to the object and making a converging acquisition, you can use the empirical formulas
- in any case, you can make a LS Simulation and the key issue is evaluation of the covariance of the parameters which partly depends on the sigma square (calibration error) and partly depends on the design matrix (position of the photos)

2021.10.08 Orientation: space resection, relative/absolute orientation, bundle block adjustment

Collinearity equations

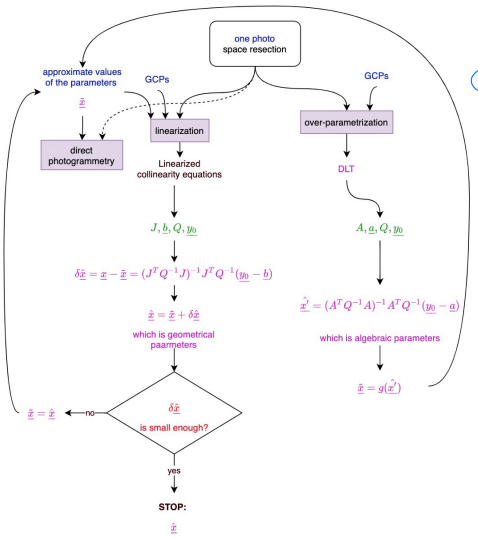
$$\begin{cases} \xi_i = \xi_0 - c\,\dfrac{r_{11}(X_i - X_0) + r_{21}(Y_i - Y_0) + r_{31}(Z_i - Z_0)}{r_{13}(X_i - X_0) + r_{23}(Y_i - Y_0) + r_{33}(Z_i - Z_0)} \\ \eta_i = \eta_0 - c\,\dfrac{r_{12}(X_i - X_0) + r_{22}(Y_i - Y_0) + r_{32}(Z_i - Z_0)}{r_{13}(X_i - X_0) + r_{23}(Y_i - Y_0) + r_{33}(Z_i - Z_0)} \end{cases}$$

Solved by LS Adjustment

Space resection

Orientation of a single photo.

// Problem: Rely on GCP and the activity to collect GCP is the one you want to minimize as much as possible.



Each photo is oriented directly using the GCP (>= 3) independently from the presence of other photos. For each GCP, write two equations:

$$\xi_i = f(\xi_0, c, X_0, Y_0, Z_0, \omega, \varphi, \kappa, X_i, Y_i, Z_i)$$
$$\eta_i = f(\eta_0, c, X_0, Y_0, Z_0, \omega, \varphi, \kappa, X_i, Y_i, Z_i)$$

Collinearity equations

#parameters = 6 (determined by double points [GCPs])

Solved by LS Adjustment:
- assume Q = I
- A = J (by linearization)

This method is employed in the ortho-photo production on areas with many GCP.

Direct photogrammetry: take the approximated value as the true value, which means that you don't need to make any adjustment. Instead, directly reconstruct the object without model.

In this case, no GCP is needed.

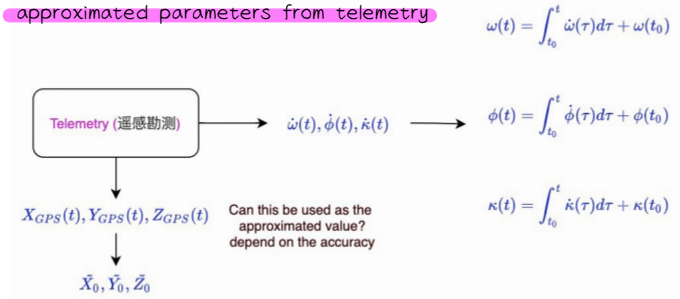
Cons of directly using linearization to solve LS problems:
- computational heavy
- Xi, Yi, Zi (where the photo is taken / how the photo orients) should be known prior which are not easy to provide
- if not start with good approximated points, you may not converge.

IN PRACTICE, given a photo:

1. Define the image reference system.
2. Collect the GPCs (Xi, Yi, Zi) [monographs] to define the object reference system
3. Look the corrected points in the image -> (ξi, ηi)
4. Orientation following the process in the above flow

Monograph: the description of a point you collect in the ground

approximated parameters from telemetry



DRIFT.
the mean of the noise increases as the time passes

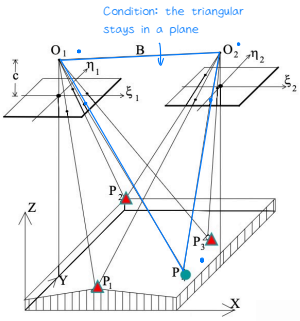
What happens when you want to do self-calibration (including the internal parameters)?
- c -> can be found in the EXIF file (Focal length 焦距).
- ξ0, η0 -> 0, 0. That is, expect the PP in the center of the image.

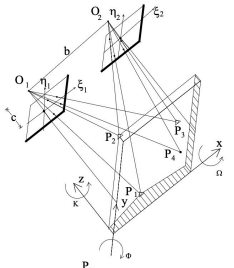
GCP (Ground Control Point): (X, Y, Z), expensive
Tie point: (ξ, η), cheap

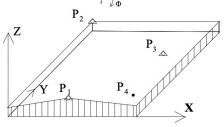
relative and absolute orientation

Simultaneous orientation of 2 photos in two subsequent steps

pros:
- minimizing the collection of GCPs: no matter how many photos you have, the minimum number of GCPs you need is 3

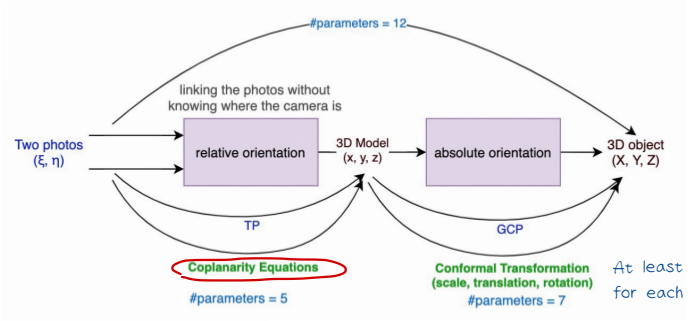






It exploits the fact that rays corresponding to homologous points (**tie points:** the one in the photo which connects two photos) intersect at the object point. (Condition)

It is not necessary to have 3 GCPs for every photo. It is an approximate solution based on linearized equations.



At least 3 GCPs are needed because for each GCP, write 3 equations

- R.O.: Large number of degree of freedom, which means infinite quantity of solutions for relative orientation.
- The Coplanarity Equations are non linear.
- The conformal equations are linear in the coordinates, but non-linear in the parameters.

Two steps:
1. Relative orientation (R.O.) of a stereoscopic couple of photos in an arbitrary RS (model system): link the images without worrying the results. (x, y, z) = f(ξ, η)
    - asymmetric method: the parameters of the camera 2 are chosen as unknown, keeping fixed the camera
1. Thus, the camera 2 is rotated and translated.
    - symmetric method
2. Absolute orientation (A.O.) in the RS defined by the GCPs (object system). (X, Y, Z) = f(x, y, z)

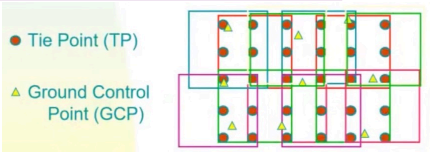
aerial triangulation

Simultaneous orientation of 2 or more photos in a single step

// the most precise solution

The aerial triangulation is a procedure for a simultaneous orientation of a block of photos.
- pros: releases the need of having, at least, 3 GCPs for each model (R./A.O.) or for each photo (space resection)

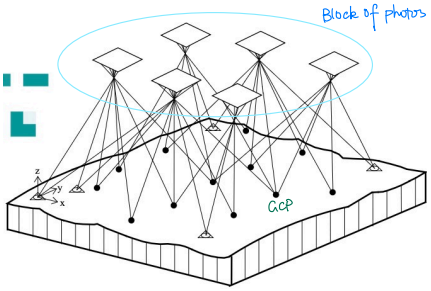
GCPs and tie points (and additional observations if available) are jointly used into a least-squares adjustment.
- tie points: link more images [(ξj, ηj)]
- GCP: may be seen on more than one image [(Xi, Yi, Zi), (ξi, ηi)]
  * GCPs contribute to the numerical stability of the solution
  * GCPs join parts of the block without common TPs



Bundle Block Adjustment (BBA)

// bundle = bundle of lines which should be converged in the right position.
// block = block of photos
// adjustment = find the right position using LS Adjustment

// that is, adjust a bundle of collinearity lines based on collinearity equations by LS Adjustment.



Input: n photos / TPs / GCPs / approximated values or over-parametrization / (option) GPS data, gyro data
- image coordinates of the GCPs and the tie points
- object coordinates of the GCPs
- possibly: E.O. Parameters determined from GPS/IMU

$X^{GPS} = X_0 + \delta X + v_X$
$Y^{GPS} = Y_0 + \delta Y + v_Y$
$Z^{GPS} = Z_0 + \delta Z + v_Z$

Procedure: for any TP and any GCP, write the corresponding collinearity equations.

$$\xi = \xi_0 - c\,\frac{r_{11}(X-X_0) + r_{12}(Y-Y_0) + r_{13}(Z-Z_0)}{r_{31}(X-X_0) + r_{32}(Y-Y_0) + r_{33}(Z-Z_0)}$$

$$\eta = \eta_0 - c\,\frac{r_{21}(X-X_0) + r_{22}(Y-Y_0) + r_{23}(Z-Z_0)}{r_{31}(X-X_0) + r_{32}(Y-Y_0) + r_{33}(Z-Z_0)}$$

Output (estimated parameters):
- external orientation of all images -> #parameters = 6n (n photos)
- in case the internal orientation of the camera -> self calibration
  * n photos + 1 camera -> #parameter = 6n + 1*3 = 6n+3
  * 2 collinearity equations for each GCP [(Xi, Yi, Zi), (ξi, ηi)] per photo (where the GCP is visible)
  * 2 collinearity equations for each TP [(ξj, ηj)] and for one photo, introducing 3 new unknowns (Xj, Yj, Zj) -> 2h equations for h-tuple TP + 3 additional unknowns
- restitution of the TP (Xj, Yj, Zj)

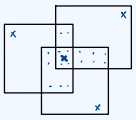
$$\xi_i = f(\xi_0, c, X_0, Y_0, Z_0, \omega, \phi, \kappa, X_i, Y_i, Z_i)$$
$$\eta_i = f(\eta_0, c, X_0, Y_0, Z_0, \omega, \phi, \kappa, X_i, Y_i, Z_i)$$

$$\xi_{i1} = f(\xi_0, c, X_{01}, Y_{01}, Z_{01}, \omega_1, \phi_1, \kappa_1, X_i, Y_i, Z_i)$$
$$\eta_{i1} = f(\eta_0, c, X_{01}, Y_{01}, Z_{01}, \omega_1, \phi_1, \kappa_1, X_i, Y_i, Z_i)$$
$$\xi_{i2} = f(\xi_0, c, X_{02}, Y_{02}, Z_{02}, \omega_2, \phi_2, \kappa_2, X_i, Y_i, Z_i)$$
$$\eta_{i2} = f(\eta_0, c, X_{02}, Y_{02}, Z_{02}, \omega_2, \phi_2, \kappa_2, X_i, Y_i, Z_i)$$

Note: at least 3 GCP are needed (same reason for A.O.).

Equations-unknowns balance:
For nTP, hTP, nGCP, hGCP, nPhoto,
Equations (2*hTP*nTP + 2*hGCP*nGCP) > Unknowns (6*nPhoto + 3 + 3*nTP)

예제 : 3 photos
- Equations -> 24 + 72 + 6 + 6 = 88
  * 4TP in 3 photos -> 4*(2*3) = 24
  * 13TP in 2 photos -> 13*(2*2) = 52
  * 1GCP in 3 photos -> 1*(2*3) = 6
  * 3GCP in 1 photos -> 3*(2*1) = 6
- unknowns -> 3*6(external parameters) + 3 (internal parameters) + 3*(4TP+13TP) = 72



PRINCIPLES
- Geometrical principle: the photos are oriented so that in the object space
  * the projective rays intersect as close as possible to the tie points
  * they pass as close as possible to the GCPs

- based on the (linearized collinearity equations and the solution is analytically determined by LS Adjustment with a proper redundancy

2021.10.14 Camera calibration; Introduction to digital photogrammetry

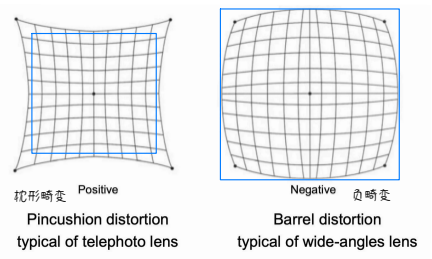
Camera calibration

In order to realize a high accuracy photogrammetric project, it is essential to calibrate the photogrammetric camera, that is, it is necessary to recover the internal orientation parameters.

The internal orientation of a camera is completely defined by knowing:
- the principal distance (c)
- the coordinates of the principal point (ξ, η) on the image plane
- the radial distortion and the tangential distortion that characterize the lens (K1, K2, K3, P1, P2) , b1 , b2
  * knowing the lens distortion allows to obtain distortion-free images and to use the frame in all its extension.

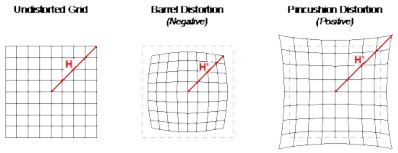
Different kinds of camera calibration:
- Laboratory calibration: an extremely precise calibration. Very expensive because it requires the use of expensive instrument, such as collimators or goniometers.
- calibration with a 3D test field: realized by acquiring a series of images of a 3D object which is characterized by a known geometry. (What we will do in the lab part)
- calibration with a 2D test field: realized by acquiring a series of images of a flat panel which is characterized by a known geometry.
- Self-calibration: with a classic Bundle Block Adjustment, a self-calibration is performed by simultaneously estimating the 3D coordinates of the object and the internal orientation parameters. (Analytical and Cheapest way)
- vanishing point method: only the coordinates of the principal point are estimated. (Changing c -> estimate of c and Z0 are correlated)

(geometical) distortion of the photos



Pincushion distortion
typical of telephoto lens

Barrel distortion
typical of wide-angles lens

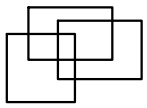


Brown's Model

The distortions are modellized by correcting the collinearity equations, using **Brown's model**, which is characterized by 10 parameters that represent the different components of the distortion:
- radial distortion (径向畸变): modelled by an odd polynomial function. It is the main component (coefficients: K1, K2 and K3) $\delta r = k_1 r + k_2 r^3 + k_3 r^5$
- tangential distortion (正切畸变): represents the not exact concentricity of the lens to the optical axis. It is an order of magnitude smaller Ethan the radial distortion (coefficients: P1 and P2)
- affine distortion (仿射畸变): models the fact that the pixel is not exactly a square (ξ/η ratio) (coefficient: b1)
- skew distortion (偏斜失真): models the fact that the pixel can be parallelogram instead of a square (coefficient: b2)

$r = \sqrt{\left(\frac{\xi-\xi_0}{d\xi}\right)^2 + \left(\frac{\eta-\eta_0}{d\eta}\right)^2}$

$\begin{cases} d\xi = \xi \cdot dr \\ d\eta = \eta \cdot dr \end{cases} \rightarrow r = \sqrt{d\xi^2 + d\eta^2} = dr \cdot r$



∀ photo : (X_0, Y_0, Z_0) $(\omega, \varphi, \kappa)$ → 6n. // n photos
∀ camera : (ξ_0, η_0, c) (Brown's model) → 10
→ # parameters = 6n + 10

Collinearity equations:

$$\begin{cases} \xi = \xi_0 - c\,\dfrac{r_{11}(X-X_0) + r_{21}(Y-Y_0) + r_{31}(Z-Z_0)}{r_{13}(X-X_0) + r_{23}(Y-Y_0) + r_{33}(Z-Z_0)} + d\xi \\ \eta = \eta_0 - c\,\dfrac{r_{12}(X-X_0) + r_{22}(Y-Y_0) + r_{32}(Z-Z_0)}{r_{13}(X-X_0) + r_{23}(Y-Y_0) + r_{33}(Z-Z_0)} + d\eta \end{cases}$$

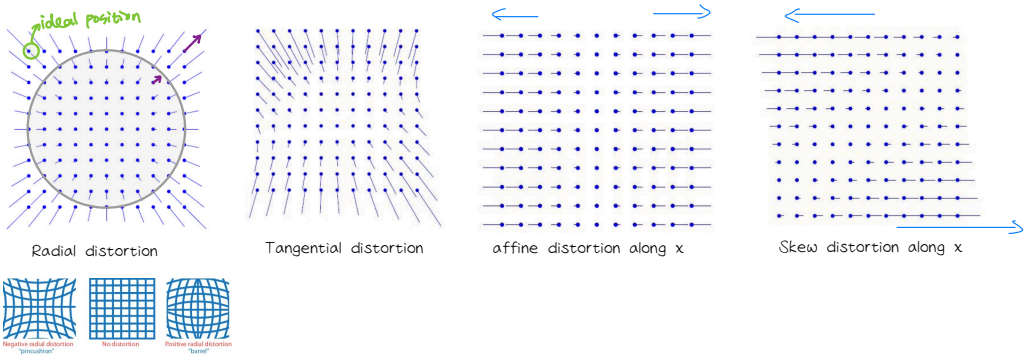
- X_0, Y_0, Z_0 represent the coordinates of the center of projection, expressed in the object reference system
- ξ_0, η_0, c represent the coordinates of the principal point and the principal distance in camera reference system
- ξ, η represent the image coordinates of a generic point P
- X,Y,Z represent the coordinates of a generic point P in the object reference system
- r_{ij} represents the elements of the rotation matrix, typically as a function of the Cardan angles φ, ω and κ of the transformation from camera to object reference system
- $d\xi$ and $d\eta$ represents the correction due to lens distortions

Brown's Model >

$$\begin{cases} d\xi = (\xi-\xi_0)\left[k_1 r^2 + k_2 r^4 + k_3 r^6\right] + P_1\left[r^2 + 2(\xi-\xi_0)^2\right] \\ \qquad + 2P_2(\xi-\xi_0)(\eta-\eta_0) + b_1(\eta-\eta_0)(\xi-\xi_0) + b_2(\eta-\eta_0) \\ d\eta = (\eta-\eta_0)\left[k_1 r^2 + k_2 r^4 + k_3 r^6\right] + P_2\left[r^2 + 2(\eta-\eta_0)^2\right] + 2P_1(\xi-\xi_0)(\eta-\eta_0) \end{cases}$$

$$\begin{cases} x = D_x - \dfrac{c \cdot N_x}{D} \\ y = D_y - \dfrac{c \cdot N_y}{D} \end{cases}$$

$$\begin{cases} D_x = x_0 + x(k_1 r^2 + k_2 r^4 + k_3 r^6) + P_1(r^2 + 2x^2) + 2P_2 \cdot x \cdot y + b_1 \cdot x + b_2 \cdot y \\ D_y = y_0 + y(k_1 r^2 + k_2 r^4 + k_3 r^6) + P_2(r^2 + 2y^2) + 2P_1 \cdot x \cdot y \end{cases}$$

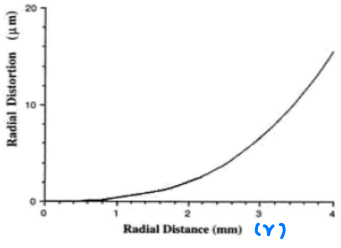


The most important kinds of distortion (radial and tangential) can be modeled with two curves:

- **the radial distortion curve**: modeled with odd polynomial functions. For practical applications, it can be stopped at power 7

$$\delta r = K_1 r^3 + K_2 r^5 + K_3 r^7$$

- the distortion curves are not symmetric, but the difference are smaller, so we can assume that the average value is a good approximation. In aerial photogrammetry, it can be modeled along the 4 semi-diagonals.



introduction to digital photogrammetry

Digital photogrammetry = photogrammetry based on the use of images in digital format.
- applications
  * close-range photogrammetry
  * aerial photogrammetry
- acquisition techniques:
  * direct by means of digital sensors
  * indirect by means of scanner (from analogue photos)
- instrument simplification: computers, SW, HD, devices/monitor for 3D vision
- automatic procedure for the orientation and for the reconstruction of surfaces (DSM)
- pros:
  * increased number of users
  * diffusion of photogrammetric techniques
  * cost reduction
  * if digital cameras are used, the photogrammetric process is completely based on computers
- cons
  * users who are no expert of photogrammetry can wrongly use the available tools
  * the automatic algorithm cannot replace the manual experience
  * final precision of the products is not always better than the one got by analogue photogrammetry (模拟摄影测量)

digital image

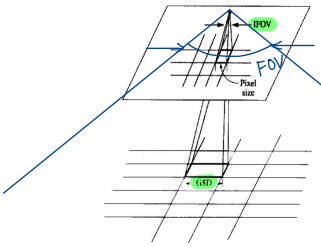
A **digital image** is composed by a regular matrix of pixels (picture elements), described in terms of both geometry and radiometry.

basic element of a digital image: **PIXEL (Picture x ELement)**

resolution

geometric resolution (几何分辨率)

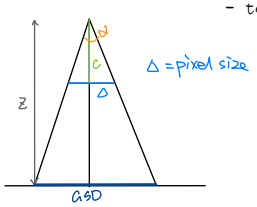
// capability of distinguishing two points on the target



- geometry of the pixel: theoretically, any geometrical shape, as long as regular; practically, a rectangular or square shape
- pixel ratio = "pixel width" / "pixel height"
- pixel size (px): size of a square pixel (pixel ratio = 1) (dimension of sensors)
- every pixel has a fixed position that allows to give a metric content to the digital image.

Instantaneous Field of View (IFOV): angle defined by the pixel size and the principal distance (α)

Ground Sample Distance (GSD): pixel projection (footprint) on the object (e.g. terrain)
- tells you the geometric resolution of the image



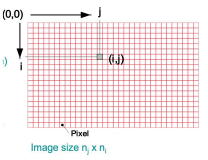
Δ = pixel size

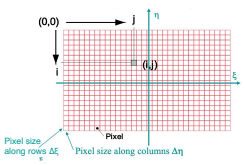
$$\frac{GSD}{Z} = \frac{\Delta}{c}$$

$$\Rightarrow GSD = \Delta \cdot \frac{Z}{c} = \Delta \cdot m_b$$

If you want smaller GSD:
- smaller sensor (Δ): but smaller sensor -> smaller collected energy -> more noisy observation -> not a good idea!
- smaller Z: closer to the object -> not so practice
- increase c: longer objective

"Pixel" reference system (i, j):
- they define the position of the barycenter of each pixel (row and column)
- the origin is usually fixed in the pixel at the upper left side corner (0, 0)





"Image" reference system (ξ, η):
- define the position of a point
- referred to a RS centered to the image

Radiometric resolution (辐射分辨率)

// capability of distinguishing two levels of colors

$DN_{min} = 0$

$DN_{max} = 2^N - 1$

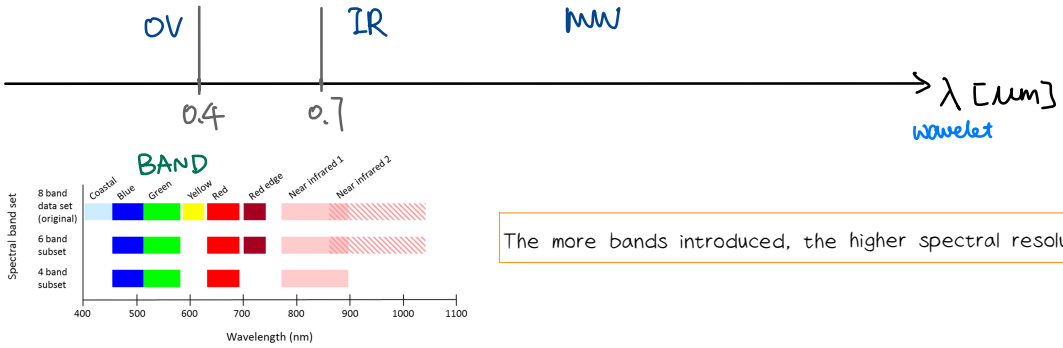
For each pixel, one/more numerical values are associated, describing the radiometry (channels) -> **DN (Digital Number)**

- each value is called **intensity** I(i,j)
- **radiometric depth of the image** = the number of bits (basic unit of a digital system) that is used for each channel
  - * 2 values -> 1 bit
  - * 4 values -> 2 bit
  - * 256 values -> 8 bit = 1 byte // most widespread format
  - * 2^N values -> N bit
- the more number of values, the more the radiometric resolution

spectral resolution (光谱分辨率)

// relate to the light/energy.

// is the number of **bands** you divide your signal.



The more bands introduced, the higher spectral resolution.

Colour images

- every color is given by the sum of 3 basic colors, each of them of different intensity (RED/GREEN/BLUE)
- two possible representation
  - * bitmap image
    - + the radiometry of each pixel is given by 3 integer numbers that express the intensity of the 3 basic colors (RGB)
    - + usually, this representation requires 3 bytes per pixel. (1 byte for 1 color)
  - * indexed image
    - + not all the possible combination of the 3 basic colors are present in a real image. Thus, the "most important" colors (usually 256) are selected
    - + each of the "most important" colors is described by
      - an index, used in the image matrix
      - a triad with the intensity of the 3 RGB components
    - + the set of the possible colors with their description is called "palette"

temporal resolution (时间分辨率)

distance of the time between two images of the same area, typically for monitor

different kinds of image

analogue and raster topology

- analogue image is an "almost continuous" representation, due to the finer grain w.r.t. the digital image
- the proximity relations between pixels follows different rules w.r.t. those in the continuous
- distinct points can become close pixels
- definition of new proximity rules

storing format

The matrix (or the matrices) that represent a digital image are stored in suitable "formats".

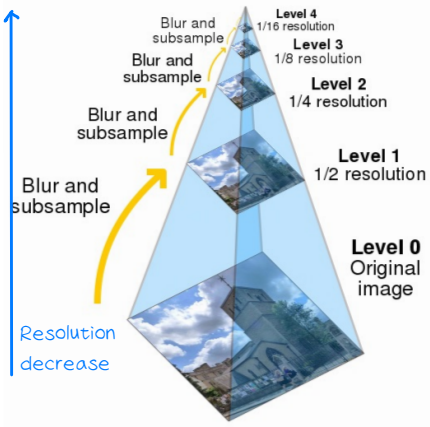
- A format for photogrammetric images has to (possible) guarantee:
  * to occupy the minimum space of memory
  * to keep the original radiometry
  * to allow a compression without loss of information (lossless)
  * to allow the management of multi-resolution images
  * to be of public domain
- widespread formats: TFF / BMP / JPEG / ECW (proprietary format used for the storing of big images - orthophoto)

compression

TYPES of compression

- lossless: without loss of information.
  * the decompressed image is identical to the original one w.r.t. radiometry and geometry
  * based on the use of codes that allow to save space in homogeneous areas from the radiometric point of view
  * example: LZW in the TIFF format
- lossy: with loss of information
  * the decompressed image is not identical to the original one:
    + radiometric differences especially in the high-contrast areas (corners, edges, ...)
    + geometrical differences with small displacements of objects.
  * based on the storing of its frequency transform rather than the original image
  * example: JPEG format (variable loss of information depending on the compression ratio)

image pyramids



(WIKI)
Pyramid, or pyramid representation, is a type of multi-scale signal representation developed by the computer vision, image processing and signal processing communities, in which a signal or an image is subject to repeated smoothing and subsampling. Pyramid representation is a predecessor to scale-space representation and multiresolution analysis.

There are two main types of pyramids: lowpass and bandpass.

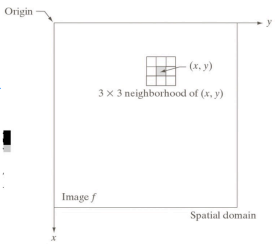
A lowpass pyramid is made by smoothing the image with an appropriate smoothing filter and then subsampling the smoothed image, usually by a factor of 2 along each coordinate direction. The resulting image is then subjected to the same procedure, and the cycle is repeated multiple times. Each cycle of this process results in a smaller image with increased smoothing, but with decreased spatial sampling density (that is, decreased image resolution). If illustrated graphically, the entire multi-scale representation will look like a pyramid, with the original image on the bottom and each cycle's resulting smaller image stacked one atop the other.

A bandpass pyramid is made by forming the difference between images at adjacent levels in the pyramid and performing image interpolation between adjacent levels of resolution, to enable computation of pixelwise differences.

image filtering

$y_0 = y + v$ (original noise)

$\hat{y} = F * y_0 = y_0 + e$ (estimated noise)



Spatial filters make use of a fixed sized neighborhood in an input image to calculate output intensities.

original(x,y) = smooth(x,y) + sharp(x,y)
-> smooth(x,y) = original(x,y)-sharp(x,y);
-> sharp(x,y) = original(x,y) - smooth(x,y)

NOTE: filtering an image means to apply a convolution to the image

low-pass

A low-pass filter is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency

Way to smooth images to remove noise or unwanted details.

- **linear filters**: use a weighted sum of pixels in the input image to calculate the output pixel.
  - * in most cases, the sum of weights is one, so the output brightness = input brightness
  - * use different convolution masks that work well for uniform noise

faster

- **nonlinear filters**: can not be calculated using just a weighted sum. Other operations (e.g., sort, log, sorting, selection) are involved in calculation.
  - * use selection and/or sorting to remove impulse noise

What is the difference between signal and noise?

The signal is correlated.
The noise is typically uncorrelated.

correlation and convolution

Formalize the phase "weighted sum of pixels" using correlation and convolution.

$g(x) = f(x) \circ w(x)$ $\quad g(x) = \int_{-\infty}^{\infty} f(x+u)w(u)du$ $\quad g(x) = \sum_{-\infty}^{\infty} f(x+u)\cdot w(u)$ $\quad g(x) = \sum_{-N/2}^{N/2} f(x+u)\cdot w(u)$

$g(x) = f(x) * w(x)$ $\quad g(x) = \int_{-\infty}^{\infty} f(x-u)w(u)du$ $\quad g(x) = \sum_{-\infty}^{\infty} f(x-u)\cdot w(u)$ $\quad g(x) = \sum_{-N/2}^{N/2} f(x-u)\cdot w(u)$

both formulas extend easily to two dimensions:

$g(x,y) = f(x,y) * w(x,y)$ $\quad g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-u, y-v)\, w(u,v)\, du\, dv$

$g(x,y) = \sum_{-\infty}^{\infty}\sum_{-\infty}^{\infty} f(x-u, y-v)\cdot w(u,v)$ $\quad g(x,y) = \sum_{-N/2}^{N/2}\sum_{-N/2}^{N/2} f(x-u, y-v)\cdot w(u,v)$

0.25 0.25 0.25 0.25	3 5 7 9 5 3 9 7 3 5 7 9 5 3 9 7	0.75 2 3 4 2.25 2 4 6 8 4 2 4 6 8 4 2 4 6 8 4 1.25 2 3 4 1.75
w(x,y)	f(x,y)	g(x,y)=f(x,y)*w(x,y)

Some important properties of convolution:

- f*g = g*f
- f*(g*h) = (f*g)*h
- f*(g+h) = f*g + f*h
- a(f*g) = (af)*g = f*(ag), where a=scalar
- f*δ = f, where δ = delta function
- d/dx(f*g) = (df/dx) * g = f * (dg/dx)

neighborhood averaging

// the easiest spatial filter to implement

Each (x,y) pixel in the image is replaced by the average of the pixels in an NxN neighborhood centered at (x,y)

- deal with image border:
  - * only process points within N/2 pixels of border -> black border on output image
  - * assume points outside the image are zero -> a gradual black border
  - * use wrap around to find points outside the image (mathematically best results)
  - * reduce neighborhood size near border of image (visually best results)
- result: this will smooth an image and remove noise and small details.

```
// Loop over output image
for (int y = N/2; y < Ydim-N/2; y++)
for (int x = N/2; x < Xdim-N/2; x++)
{
   // Loop over NxN window
   int sum = 0;
   for (int dy = y-N/2; dy <= y+N/2; dy++)
   for (int dx = x-N/2; dx <= x+N/2; dx++)
        sum += Input[dy][dx];
   Output[y][x] = sum / N*N;
}
```

weight: NxN

Cons:

- does not work well for isolated bright/dark pixels (impulse noise)

- Neighborhood sizes 3x3, 5x5, 9x9, 15x15, 35x35

$\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$ 3x3

NOT DETAILED MENTIONED IN THE CLASS

Binomial Filtering

Use Binomial coefficients as weights to give more emphasis to pixels near the center of NxN neighborhood

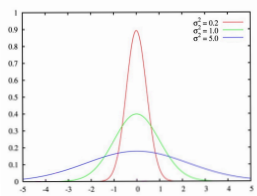
1	2	1		1	3	3	1		
2	4	2		3	9	9	3		
1	2	1		3	9	9	3		
				1	3	3	1		

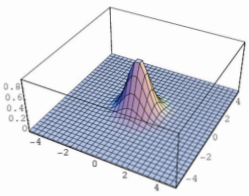
1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

```
// Loop over output image
int Weight[N][N] = { {1,2,1}, {2,4,2}, {1,2,1} };
for (int y = N/2; y < Ydim-N/2; y++)
for (int x = N/2; x < Xdim-N/2; x++)
{
   // Loop over NxN window
   int sum = 0;
   for (int dy = -N/2; dy <= N/2; dy++)
   for (int dx = -N/2; dx <= N/2; dx++)
      sum += Input[dy+y][dx+x] * Weight[dy+N/2][dx+N/2];
   Output[y][x] = sum / 16;
}
```

Gaussian Blurring

Use the Gaussian function to define the neighborhood weights.





```
// Initialize Gaussian weights
int Weight[N][N];
for (int y = -N/2; y <= N/2; y++)
for (int x = -N/2; x <= N/2; x++)
   Weight[y+N/2][x+N/2] =
      exp(-(x*x+y*y) / (2*sigma*sigma));
```

$$\exp\left[-\frac{-(x^2+y^2)}{2\sigma^2}\right]$$

Larger variance , larger filter

Outlier removal

Locate and remove noise pixels while leaving the rest of the image unchanged.

- ~~Algorithm:~~
  - for each pixel, calculate neighborhood average
  - compare smoothes pixel value to original pixel value
  - if difference greater than threshold, use smooth value // outlier
  - otherwise, use original value.

```
// Loop over output image
for (int y = N/2; y < Ydim-N/2; y++)
for (int x = N/2; x < Xdim-N/2; x++)
{
   // Loop over NxN window
   int sum = 0;
   for (int dy = y-N/2; dy <= y+N/2; dy++)
   for (int dx = x-N/2; dx <= x+N/2; dx++)
      sum += Input[dy][dx];
   if (abs(Input[y][x] – sum / N*N) > threshold)
      Output[y][x] = (sum – Input[y][x]) / (N*N – 1);
   else
      Output[y][x] = Input[y][x];
}
```

median filtering

The mean value of N values is not a robust statistic because an outlier can introduce bias. On the other hand, the median value of N values is a robust statistic

- pros: do not need to specifically identify outliers or decide how to replace them
- cons: median calculation is slower than mean calculation because of SORTING
- algorithm: for each pixel in the input image
  - copy the NxN neighborhood values into an array
  - sort the NxN values in ascending or descending order
  - use the midpoint value (the median) as the output pixel.
- nonlinear filter because sorting is based on pixel comparison and impossible to implement using only weighted averages.

k-Nearest Neighbors

Select the k-nearest neighbors in intensity

- ~~Algorithm:~~ for each pixel in the input image
  - copy the NxN neighborhood values into one array
  - store the NxN intensity differences in second array
  - sort both arrays based on intensity differences
  - average the k intensity values with smallest differences.
- non-linear filter because sorting is based on pixel comparison and impossible to implement using only weighted averages.

Original = Low-pass + high-pass ⇒ highpass = Original − lowpass

high-pass

// used more in photogrammetry

Techniques that sharpen images to enhance small details. Image sharpening can be thought of as the "dual" of image smoothing

Linear spatial filtering methods:
- spatial highpass filter
- unsharp masking
- derivative filters
- Laplacian filters

Nonlinear spatial filtering methods
- variance based enhancement
- Wallis operator
- gradient magnitude
- Laplacian Zero crossings

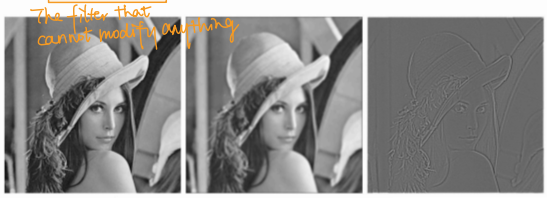
spatial highpass filter

Each (x,y) pixel in the image is replaced by the difference between the original pixel value and the average of pixels in an NxN neighborhood centered at (x,y), which can be implemented using an NxN convolution mask that combines averaging and substraction.

$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} - \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}\frac{1}{9} = \begin{bmatrix}-1&-1&-1\\-1&8&-1\\-1&-1&-1\end{bmatrix}\frac{1}{9}$$

Original Average Highpass

The filter that cannot modify anything



$$\begin{bmatrix}0&0&0\\0&1&0\\0&0&0\end{bmatrix} - \begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix}\frac{1}{16} = \begin{bmatrix}-1&-2&-1\\-2&12&-2\\-1&-2&-1\end{bmatrix}\frac{1}{16}$$

Original Binomial Highpass

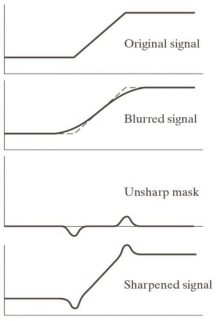
Note that, after filtering, also the noise is amplified

unsharp masking

Subtract a blurred image from original to get unsharp mask image. Then, add a multiple of unsharp mask to original to get sharpened image.

Unsharp = Original − Blurred

Sharp = Original + M · Unsharp

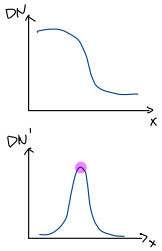



```
// Method to perform unsharp masking
void im_short::UnsharpMasking(
  int iterations, float weight)
{
  // Copy input image
  im_short in(*this);

  // Perform Binomial smoothing
  for (int count=0; count<iterations; count
    ++)
    in.Binomial();

  // Find intensity range of image
  int Min, Max;
  MinMax(Min, Max);

  // Process all pixels
  for (int index = 0; index<NumPixels;
      index++)
  {
    float result = Data1D[index] + weight *
      (Data1D[index] - in.Data1D[index]);
    if (result < Min)
      Data1D[index] = Min;
    else if (result > Max)
      Data1D[index] = Max;
    else
      Data1D[index] = (short)result;
  }
}
```

For more information, see the slides

derivative filters

Derivatives are used in a variety of image processing applications:
- enhancement
- edge detection



First derivative = gradient

$$\nabla f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (f_x, f_y)$$

$$|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2} \approx |f_x| + |f_y|$$

$$\tan\theta = f_y / f_x \rightarrow \theta = \tan^{-1}(f_y / f_x)$$

- derivatives of discrete functions are normally calculated using finite difference (by Taylor expansion)

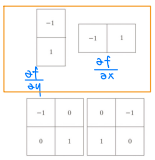
$$f(i+1,j) = f(i,j) + f'(i,j)(i+1-i) + f''(i,j)\frac{(i+1-i)^2}{2} + \cdots \quad O(x^2)$$

$$\rightarrow f'(i,j) = f(i+1,j) - f(i,j)$$

These derivates can be calculated using convolution using small masks

$$\frac{\partial f}{\partial x} = f(x+1,y) - f(x,y), \quad \frac{\partial f}{\partial y} = f(x,y+1) - f(x,y)$$

- 1x2 and 2x1 derivative masks



- 2x2 Roberts cross derivative masks

Square and odd. Perfect! (Logic Reason)

- 3x3 Prewitt derivative masks

- 3x3 Sobel derivative masks

consider $O(x^3)$

$$f(i+1,j) = f(i,j) + f'(i,j)\frac{(i+1-i)}{+1} + f''(i,j)\frac{(i+1-i)^2}{2} + O(x^3)$$

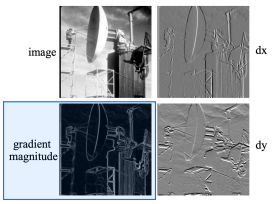
$$f(i-1,j) = f(i,j) + f'(i,j)\frac{(i-1-i)}{-1} + f''(i,j)\frac{(i-1-i)^2}{2} + O(x^3)$$

$$f(i+1,j) - f(i-1,j) = 2f'(i,j) + O(h^3) \Rightarrow f'(i,j) = \frac{1}{2}[f(i+1,j) - f(i-1,j)]$$

$$\left[-\frac{1}{2} \;\; 0 \;\; \frac{1}{2}\right]$$ odd, better!

To choose one pixel, not many.

The stronger the gradient, the stronger the corner



Second derivative = Laplacian

$$f''(x,y) = \frac{\partial f^2}{\partial x^2} + \frac{\partial f^2}{\partial y^2} = f_{xx} + f_{yy}$$

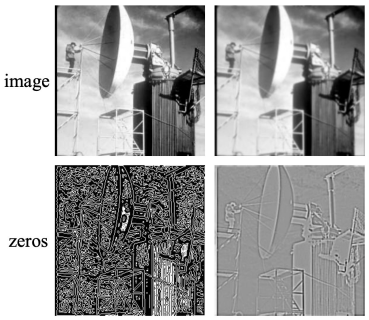
- the Laplacian is rotationally invariant like the gradient magnitude (梯度幅度)
- the Laplacian of a discrete function can be calculated using finite differences with corresponding convolution masks

$$f(i+1,j) + f(i-1,j) = 2f(i,j) + f''(i,j) \rightarrow f''(i,j) = f(i+1,j) - 2f(i,j) + f(i-1,j)$$

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$ 3x3 Laplacian mask

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} + \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

- the zeros of the Laplacian correspond to extreme of the gradient, so the Laplacian zero crossings are used for edge detection

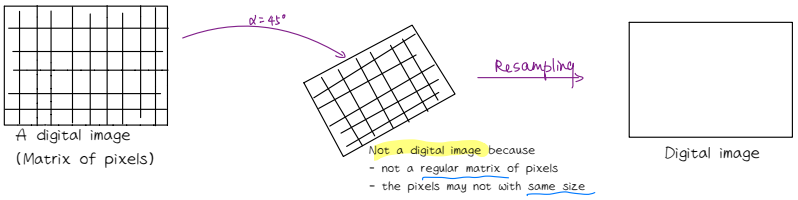


2021.11.04 Resampling; Interesting point operators; Feature-based matching

resampling

What kinds of problem "resampling" solved?

In order to get a digital image after transformation of a digital image.



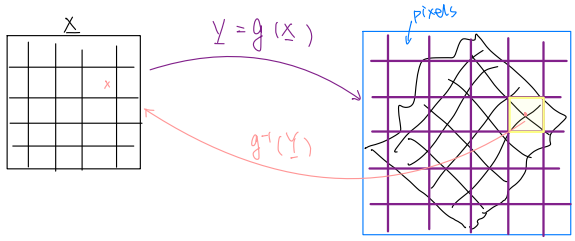
A digital image
(Matrix of pixels)

Not a digital image because
- not a regular matrix of pixels
- the pixels may not with same size

Digital image

Use case:
- photo plane

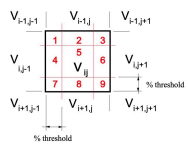
Resampling process



1. Define a bounding box containing the transformed image
2. Define pixels
3. Set the new pixels with specific colors
    1) find the barycenter of the new pixel
    2) project the center back to the original image through the inverse transformation.
    3) decide the color of the pixel depending on the color of the projected pixel

The simplest method is **Nearest Neighbor**
- HOW: the barycenter of each pixel of the resampled image (output image) is transformed back to the original image. And the resampled pixel value is the one of the pixel where the barycenter falls.
- pros: easy and fast
- cons: it could happen that more than one pixel of the resampled image fall in the same pixel of the original image (deformation -> loss the shape)

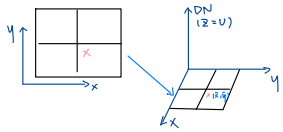
Interpolation method: taking also into account the surrounding pixels
- influence zones: the original pixel is subdivided into 9 zones (with different surfaces). Depending on the zone where the transformed pixels falls, the gray value is computed in a different



$$V_{R_2} = \frac{V_{i-1,j} + V_{ij}}{2} \qquad V_{R_1} = \frac{V_{i+1,j} + V_{ij} + V_{i,j-1} + V_{i+1,j-1}}{4}$$

$$V_{R_3} = \frac{V_{i,j-1} + V_{ij}}{2} \qquad V_{R_4} = \frac{V_{i+1,j} + V_{ij} + V_{i,j+1} + V_{i+1,j+1}}{4}$$

- interpolations
    * bilinear: the grid formed by the barycenters of the 4 closest pixels is used again, and the gray value of the resampled pixel is determined by means of a bilinear interpolation of the 4 gray values of the 4 closes pixels (grid nodes)



$$V = ax + by + cxy + d$$

4 data

$$\hat{a}, \hat{b}, \hat{c}, \hat{d}$$

$$\bar{V} = \hat{a}\bar{x} + \hat{b}\bar{y} + \hat{c}\bar{x}\bar{y} + \hat{d}$$

// #parameters = 4
// #data = 4 -> #equations = 4
// -> deterministic!

    + act like low-pass -> loss the detail
    * bicubic

// more see the slide

interesting point operators

Extract distinguishing points that are good candidates for the matching algorithms. (That is, find the double points automatically.)

Interesting point = possible candidates for double points

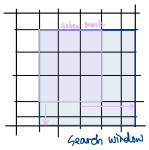
- Most common algorithm in Aerial photogrammetry: Förstner operator
- most common algorithm in Terrestrial photogrammetry: SIFT

all of the operators are based on the high-pass filter, computing the derivative of the pixels

Förstner operator

Based on the extraction of points characterized by a high gradient in the 4 directions (row, column and along the 2 diagonals)

1. a square window is considered (with side of 5-9 pixels) inside which the gradients are computed
   * for example, use Sober mask:
     + gi', gj' = derivative in the i, j direction respectively
     + g'_{i,k}, g'_{j, k} // k=position of the sober maskin the search window
     + compute $N = \begin{bmatrix} \Sigma g'^2_i & \Sigma g'_i g'_j \\ \Sigma g'_i g'_j & \Sigma g'^2_j \end{bmatrix}$
   * this operation is applied on the whole image

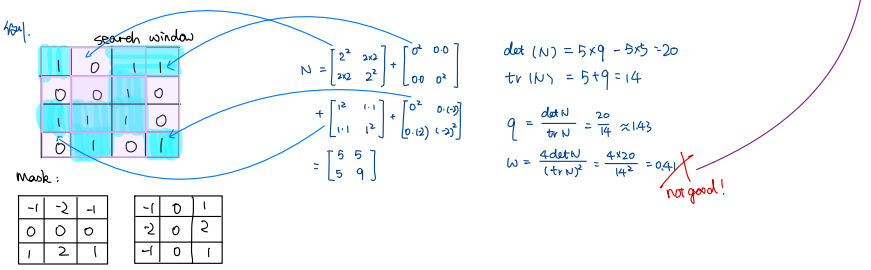


In this case, 9 derivatives can be computed (3 in the i direction, 3 in the j direction)

2. starting from the matrix N, the following two parameters, called **interest** (q) and **circularity** (w), are computed

$$q = q \cdot \frac{\det N}{tr(N^{-1})} = \frac{\det N}{tr\,N} = \frac{\lambda_1\lambda_2}{\lambda_1+\lambda_2} \qquad w = \frac{4\det N}{(tr\,N)^2} = 1 - \left(\frac{\lambda_1-\lambda_2}{\lambda_1+\lambda_2}\right)^2$$

   * the circularity must have a minimum value (0.9) to detect points with high contrast in all directions
3. selection of the extracted points which is performed by means of the interest:
   1) Points are sorted on the basis of the interest and the first ones of the sequence are chosen
   2) Once points are detected, their positions are defined as the barycenter of the gray scale window
      * this applies the definition of a minimum distance between the detected points to avoid adjacent points



SIFT operator

SIFT = Scale Invariant (Independent) Feature transform

SIFT is invariant for scale, rotation (around the optical axis) and results robust against variations of the camera attitude and changes of the scene lighting

- cons: the computational burden of SIFT is quite high, so that it is used on compressed images at a moderate resolution.

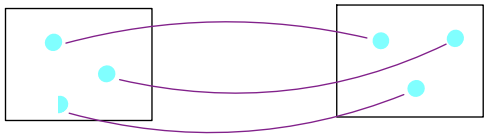
Working principles

SIFT is able to extract interest points from images (by a detector) and to identify the same point on various images (by a descriptor). For each point (x,y), an identifier vector is associated (generally a vector of 128 elements). The determination of homologous points is performed by comparing the descriptors among the various images.

How to detect tie points?
1. Find interesting points (with strong contrast w.r.t. the surrounding)
2. Couple the interesting points in different images

matching

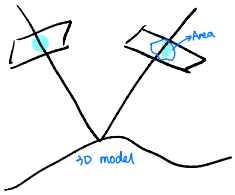
Feature based matching (FBM): you don't need any prior information on the orientation



● Feature

// matching features without the need of knowing where the features are

Area based matching (ABM): matching the area in which the feature is
// you need prior orientation (e.g., feature location) and prior 3D model



Typically, the process is: FBM (find approximated TPs) -> ABM (to improve accuracy of the TP)

General consideration for **Matching**
- philosophy of the coupling (strategy)
  * FBM (Feature based matching)
  * ABM (Area based matching)
- index for the coupling (evaluation)
  * linear correlation index
  * LSTM
  * ...

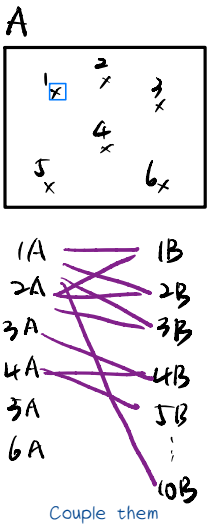
Feature based matching

1. Extraction of interesting points on both images
2. Make the couples (consider all the possible couples)
3. Compare the couples (Using the index to check if the couple is good or not): open a window for each interesting point (this window is called feature) to compare
   - the simplest window (feature) is 3x3

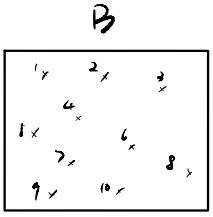
$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$\sum\sum (a_{ij} - b_{ij})$ ⟶ Not a good index, like: $\begin{bmatrix} 10 & 10 & 10 \\ 10 & 10 & 10 \\ 10 & 10 & 10 \end{bmatrix}$ $\begin{bmatrix} 20 & 20 & 20 \\ 10 & 10 & 10 \\ 0 & 0 & 0 \end{bmatrix}$

$\sum\sum (a_{ij} - b_{ij})^2$ ⟶ better, but still not good. Think about the scale case in which the geometry is the same but get the high value
   -> so, not a-b, but $b = \alpha^* a + \beta$
   -> for each couple, the **linear correlation coefficient ρ** between the windows around the considered points is computed
   // the higher the linear correlation index, the better the coupling



Couple them



× interesting points

List of interesting points in image A (left) and image B (right). Each interesting point has a coordinate(row, column). for example, 1A(16, 32) means the interesting point in pixel at row 16 and column 32

Why FMB based on linear correlation index?

Because you are looking for features that are differ by constant (offset) or scale factor

Linear Correlation index (to evaluate if the relationship between a and b is linear)

$$\mu_a = \frac{1}{N}\sum_{ij} a_{ij} \quad , \quad \sigma_a^2 = \frac{1}{N}\sum_{ij}(a_{ij}-\mu_a)^2$$

$$\mu_b = \frac{1}{N}\sum_{ij} b_{ij} \quad , \quad \sigma_b^2 = \frac{1}{N}\sum_{ij}(a_{ij}-\mu_b)^2$$

(green channel)

$$\sigma_{ab} = \frac{1}{N}\sum_{ij}(a_{ij}-\mu_a)(b_{ij}-\mu_b)$$

$$\Rightarrow \quad -1 \le \rho_{ab} = \frac{\sigma_{ab}}{\sqrt{\sigma_a \sigma_b}} \le 1$$

The closer to 1, the better the coupling

4. Selection on the homologous points: among all the couples for a specific point, the one with maximum ρ is selected

Question: what is the bayer filter implementing in the color image due to the camera limitation?

Question: if you have a color image composed by RGB, which is the channel you select to compute the correlation index?
Select the green because in the commercial camera due to the bayer filter, green is the most informative one, while the others are more interpolated

Cons:
- the precision ion the determination of homologous points is not high: cannot go down to a pixel-level accuracy
- the number of detected homologous point is high, most of them could be false.
- time consuming
  * how to speed up?
    + Look the couples only around the correlation line to reduce the number of couples to be computed

特点：
- do not necessarily require an approximate orientation to initialize the research
- make use of pyramids of images
  * because in lower level, we can reduce the number of homologous points and the number of possible combination
  * HOW: reduce the resolution to find the homologous point, which should be very strong. And then find the position of the corresponding points in the higher resolution without finding the additional homologous points.

FBM v.s. ABM

In the ABM, you are not looking for interesting points in image A in image B. You just look for the interesting points in the image A, and then look for the similar feature in the image B without having interesting points.

ABM can detect more complex tie point than FBM.

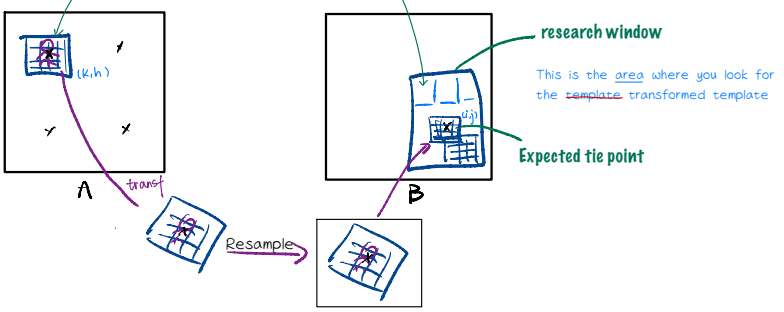
What happens if one feature rotated w.r.t. another one?

$$\begin{bmatrix} 1 & 1 & 1 \\ 10 & 10 & 10 \\ 20 & 20 & 20 \end{bmatrix} \quad \begin{bmatrix} 1 & 10 & 20 \\ 1 & 10 & 20 \\ 1 & 10 & 20 \end{bmatrix}$$

Area-based Matching

Input:
- approximate orientation
- approximate 3D model

template = a structure used as a reference = a window extracted from the images around the possible candidates to be homologous points



X_A = template in image A
$g(X_A)$ = transformation of the template
$X_{B(i,j)}$ = template in image B

$g(X_A, \underset{\text{Rotation angle}}{\theta})$ Affine $\longleftrightarrow$ vs $X_{B(i,j)}$ // required most similar, identical is impossible

Least Square Template Matching (LSTM)

LSTM principle: minimize the gray scale differences between template and slave according to the least-squares principle

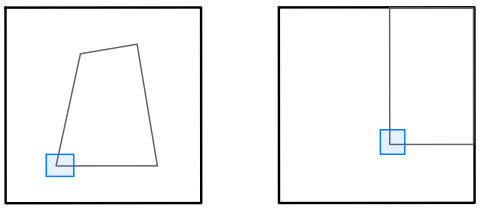
$$\min_{\theta,i,j} \sum_{k,h} \left[g(X_A, \theta)_{k,h} - X_{B(i,j)_{k,h}} \right]^2$$

Pixels inside the template

- the parameters to estimate describe the geometric and radiometric differences between the two images
  - geometric model

Affine $\begin{cases} x' = a_1 + a_2 x + a_3 y \\ y' = a_4 + a_5 x + a_6 y \end{cases}$

  - radiometric model $X'_A = g(X_A) = aX_A + b$



Pros:
- a sub-pixel technique that allows to reach precisions of the order of 1/20 pixel

Normalized cross-correlation

Principle:
1. Extract interesting point from image 1
2. For each of these points, compute the approximate position on the image 2
3. Open a window around the point on the image 1 (template)
4. Search the homologous point around the approximate position on the image 2 with the method of cross-correlation
5. The two detects points are considered as homologous if their correlation is higher than a threshold

Multi-photo geometrically constrained matching

The more photos, The better

Make use of more than two images.
It includes some geometric relations into the LS system to constrain the points:
- belong to the epipolar lines
- consistency of the point coordinates in the object system

pros: most robust method
cons: difficult to define the weights to assign to the observations.

relational matching

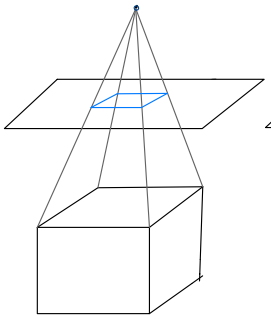
aim at working also with images of different types:
- strong geometric and radiometric differences
- images acquired with different sensors
- matching between images and digital/raster maps

Based on the extraction of features from both the images:
- detection of geometric or topologic relations among features of the same image
- coupling the objects on the basis of the relations

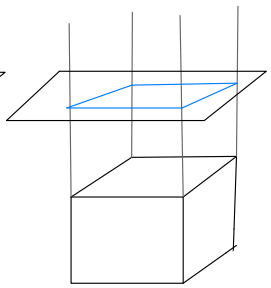
The algorithms provide approximate homologous points, starting from which FBM and ABM methods are then applied

Product Photogrammetry

2D model: Orthophoto



Image



Orthophoto
(georeferenced image: with CRS, no perspective distortion)

Image v.s. Orthophoto:
- an image is a central projection of the object
- an **Orthophoto** is the same image but without perspective distortions
  * the Orthophoto combines the characteristics of a traditional map (constant scale, possibility to measure angles, distances, ⋯) with those of an image (photographic realism)

Photoplane: assume that the object is 2D, therefore a point on the object is identified by a couple of values (X,Y) with constant Z (Z=0)

$$X = X_0 - Z_0 \frac{r_{11}\xi + r_{12}\eta - r_{13}c}{r_{31}\xi + r_{32}\eta - r_{33}c}$$

$$Y = Y_0 - Z_0 \frac{r_{21}\xi + r_{22}\eta - r_{23}c}{r_{31}\xi + r_{32}\eta - r_{33}c}$$

// $\xi_0 = \eta_0 = Z = 0$ // collinearity equations

Homography equation: transformation from the 2D object coordinates to the corresponding 2D image coordinates

$$X = \frac{a_1\xi + a_2\eta + a_3}{c_1\xi + c_2\eta + 1}$$

$$Y = \frac{b_1\xi + b_2\eta + b_3}{c_1\xi + c_2\eta + 1}$$

#parameters = 8
#equations for one point = 2 => #double points >= 8/2 = 4
Therefore, the survey can be performed on a single image

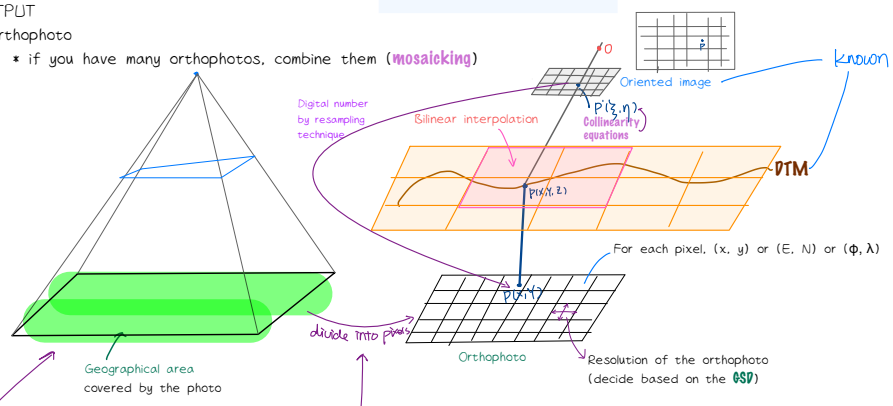
In the Orthophoto case:

INPUT
- 1 image
- orientation (internal / external) — camera — Oriented Image
- (3D) model of the object (**DTM**)

known for oriented photo:
X, Y, Z
C, ξ0, η0
X0, Y0, Z0
ω, ψ, κ

⇒ collinearity Equations → (ξ, η)

OUTPUT
- orthophoto
  * if you have many orthophotos, combine them (**mosaicking**)



Standard way of proceeding (indirect method)
1. calibration your camera
2. acquisition of photos (block)
3. bundle block adjust
   a. orientation
   b. restitution
4. Inside the geographic area, obtain a rectangular shape/image
5. Divide the geographical area in pixels => (X, Y) is known because the pixel is georeferenced
6. Project the pixel to the DTM model => (X, Y, Z) is known
   - if the resolution in DTM is different with that in orthophoto, then interpolate the value (for example, using bilinear interpolation)
7. Apply the collinearity equation to find the corresponding point on the original image => (ξ, η)
8. For each pixel, compute its DN based on its corresponding coordinate (ξ, η) in oriented image by resampling

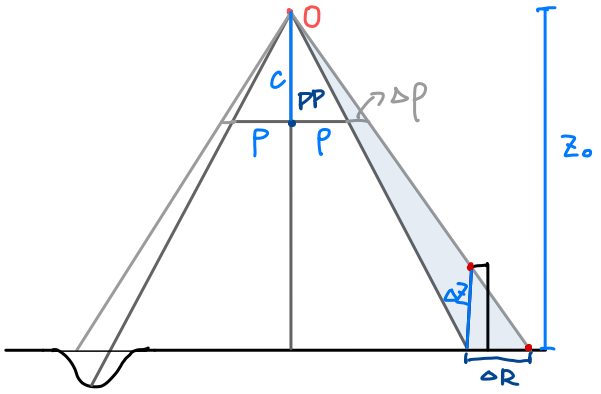
Anchor points method (only the DTM gird knots are transformed in a rigorous way):

compute an approximate transformation from (X,Y) of the terrain to (r, c) of the image for the points inside each grid stitch.

- In presence of break lines, a linear transformation is performed into the corresponding triangle
- otherwise, a bilinear transformation is performed

$X, Y \xrightarrow{lin\ /\ bilin} r, c$

Orthophoto error: object points that do not belong to the DTM are radially shifted on the Orthophoto



$$\frac{\Delta\rho}{c} = \frac{\Delta R}{Z_0} \Rightarrow \Delta\rho = \Delta R \frac{c}{Z_0}$$

Error of DTM on the Image

$$\frac{\Delta R}{\Delta Z} = \frac{\rho}{c} \Rightarrow \Delta R = \Delta Z \cdot \frac{\rho}{c}$$

Error of DTM

$$\Rightarrow \Delta\rho = \rho \cdot \frac{\Delta Z}{c\, m_b}$$

The problem of orthophoto:

- altitude error is transformed into planar error
- problems with facade and shadows:
  - * if the building not included into DTM -> the output wrongly project on the orthophoto the facade of the building
  - * if the building included into DTM -> roof instead of shadow

Use of **DSM** for the computation of true-orthophotos

- DSM in fact contains information on all objects that lie on the terrain (basically, it is the surface interpolation of the first LIDAR impulses)
- by means of the DSM, it is possible to force the building to stay in its correct position, but it is not possible to correctly reconstruct its perimeter

Lab: Close-range photogrammetry

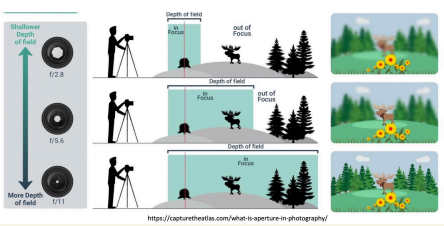
Steps for 3D manual modeling

1. camera calibration
    1) take photos to a calibration grid
    2) process them to estimate calibration parameters
2. 3D modeling
    1) take the photos to the object
    2) find tie points on the image (for referencing)
    3) collimate double points to reconstruct the model
    4) perform bundle block adjustment to orient the images
    5) draw lines and surfaces of the 3D object

预备知识：digital cameras

名词表

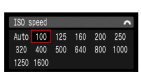
- Digital sensors
  - CCD (charge coupled device)
  - CMOS (complementary metal oxide semiconductor)
  - sensor dimensions
- colored digital images
  - bayer filter
  - demos acing
- full frame reflex
- UAVs (Unmanned Aerial Vehicles)
- taking photos
  - exposure: amount of light which reaches the camera sensor
    - + shutter speed (the "exposure triangle"): control the duration of the exposure <=> exposure time
    - + aperture: controls the area over which light can enter your camera
      - f-stop value
      - depth of field: the range of distance where objects appear in sharp focus
      - hyperfocal distance
    - + ISO speed (sensor sensitivity): controls the sensitivity of your camera's sensor to a given amount of light
      - the lowest possible ISO speed is recommended
- distortion
  - geometric distortion
  - radiometric distortion
- image formats
  - RAW
  - JPEG



https://capturetheatlas.com/what-is-aperture-in-photographs/


Low ISO Speed
(low image noise)


High ISO Speed
(high image noise)



Doubling the ISO speed, double the sensitivity of the sensor and consequently the exposure

Example:

ISO100, F/4.0, 1/60

corresponds to

ISO200, F/5.6, 1/60
or to
ISO200, F/4.0, 1/120

Goal of the lab:

- how to design and take photogrammetric surveys
- photogrammetric processing workflow by means of different software
- the results/outputs quality evaluation by means of statistical testing

2021.10.21 Close-range photogrammetry Lab: photo acquisition; calibration

photo acquisition

Require:
- a camera (use iPad?)
- a printed calibration grid
- a printed reference grid
- photomodeler software
- excel software (for statistical evaluation
- the object to measure (robik cube)

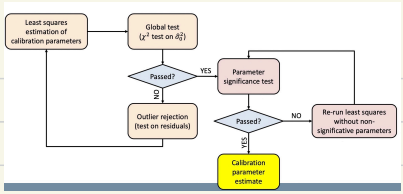
Note for the procedure:
- camera:
    * it is fundamental to perform the camera calibration in the same focus condition that will be used during the survey phase
    * it is required to disable the image self-rotation.
    * it is preferable disable the auto-focus and use e.g. hyperfocal distance.
        + Tap the object you want to focus, but hold your finger on the screen for a few seconds, when the square starts to pulse, release your finger. The square will be gone & AE/AF lock will appear at the bottom.
    * the camera has to be inclined by almost 45 degree
- calibration grid
    * all the dots must be correctly in focus to be recognized
    * the calibration pattern should occupy as much as possible area on the images
    * acquire 12 images from 4 sides of the calibration polygon
        + in order to uncorrelated the estimated parameters, it is fundamental to rotate the camera +-90 degree (for that disable the images self-rotation)
- take photos of the object
    * at least 8 photos: one for one side, one for one corner

calibration

Output:
- internal orientation
- external orientation
- distortion (using Brown model)

PROCESS (calibration project)
1. Add images
2. Run the calibration
2.1 point detection
2.2 calibration bundle block adjustment to estimate the internal orientation and distortion camera parameters
- Total Error: first indication of the quality of the calibration procedure
2.3 "Show Report"
- possible warnings
    * overlapping > 70%
- information about the camera
    * total error: first indication of the quality of the calibration procedure. It represents the value of the estimated standard deviation that should be close to 1, depending on the a-priori assumptions on the noise made by the software. If not:
        + there exists outliers
        +
    * camera calibration standard deviation: square root of the diagonal of the covariance matrix
2.4 save calibration report to text file (if cannot, take a screen shot)
3. Store the calibration

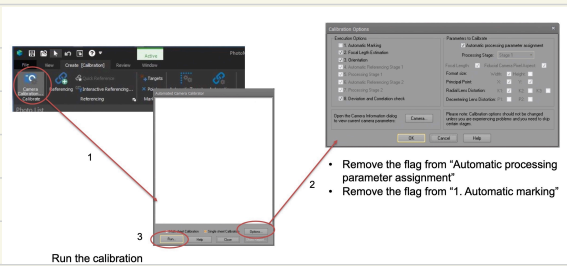
4. Evaluate the result: Global test



5. (if the global test fails) Remove points with high residuals

- in principle, a point at each time has to be removed, but to speed up the procedure, we can setup a threshold (e.g. 5 pixels) and remove all the points with residuals larger than the threshold
    * the threshold has to be chosen according to the expected quality of the camera (in general we expect residuals in the order of few pixels.
- 1001, 1002, 1003, 1004 are the coded targets, DO NOT DELETE THEM!!
- HOW?
    1. Point(Quality)
    2. Ordered by "Largest Residual (pixels)"
    3. Select the points and pressing canc/del

6. Re-run the calibration with custom parameters

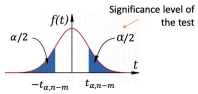


7. Repeat 4~6 until global test passed

8. Testing the significance of the parameters

Hypothesis: the parameter x_i is equal to 0

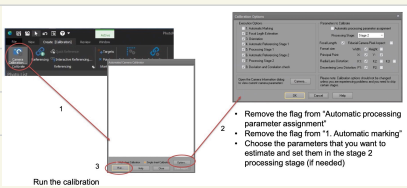
$$\frac{\hat{x}_i}{\hat{\sigma}_{x_i}} \sim t_{n-m}$$



Where n is the number of equations and m the number of estimated parameters.

If the test passes, the corresponding parameters can be neglected (the project has to be reprocessed without estimating it, if possible)

9. Re-run the calibration with custom parameters



2021.10.22 Close-range photogrammetry lab: photo orientation; model restitution

photo orientation

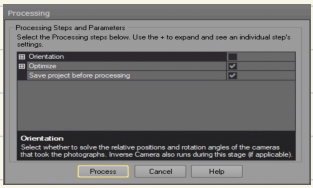
PROCESS (Manually Marked Project)

1. Load images
2. Points collimation
    - points on the grid: 3.7, 2.6, 4.2, 4.6, 5.4, 1.4, 2.2, 3.1
3. Image exterior orientation: process -> check orientation and "save project before processing"
4. Project scaling

model restitution

Process

1. Collimate the corners of the cube and run "process" keeping "Orientation" unchecked



- the total error is increased because now we have points on cube. This tells you that we have points on cube less accurate collimated

2. Draw lines
3. Surface definition
4. Texture projection
5. Export point(all) and line(all) for further analysis

- PROBLEM: software crashes when open line(all) table. Export this information manually by clicking the lines in the photos and exporting models with line informations for cross-validation

2021.11.03 Close-rage photogrammetry lab: model assessment

1. Comparison of two length

$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} = \left[(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2\right]^{\frac{1}{2}}$$

given $\sigma_{x_1}, \sigma_{x_2}, \sigma_{y_1}, \sigma_{y_2}, \sigma_{z_1}, \sigma_{z_2}$

By Covarian Propagation

$$\sigma_l^2 = \sum_i \left(\frac{\partial L}{\partial i}\right)^2 \sigma_i^2$$

$$\frac{\partial L}{\partial x_1} = \frac{1}{2} \cdot \frac{2 \cdot (x_1 - x_2) \cdot 1}{l} = \frac{(x_1 - x_2)}{l} \quad \cdots$$

Then. $\hat{l}_1 = l + v_1$

$\hat{l}_2 = l + v_2$

Assume $v_1 \sim N(0, \sigma_1^2)$, $v_2 \sim N(0, \sigma_2^2)$

$\longrightarrow \hat{l}_1 - \hat{l}_2 = l + v_1 - l - v_2 = v_1 - v_2 \sim N(0, \sigma_0^2)$

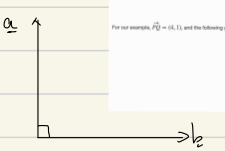
$\sigma_0^2 = \sigma_1^2 + \sigma_2^2 - 2\sigma_1\sigma_2$

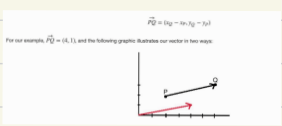
$$\longrightarrow \frac{\hat{l}_1 - \hat{l}_2}{\sigma_l} = \frac{\hat{l}_1 + \hat{l}_2}{\sqrt{\sigma_1^2 + \sigma_2^2}} \sim Z$$

[H_0: the two length are equal]

2. Verify if the angles are 90° angles

given $\underline{a} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$, $\underline{b} = \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$





β

Then $\underline{a}\,\underline{b} = \boxed{a_x b_x + a_y b_y + a_z b_z} = 0$

$a_x = a_{x_B} - a_{x_A}$

$$\sigma_\beta^2 = \sum_i \left(\frac{\partial \beta}{\partial i}\right)^2 \sigma_i^2$$

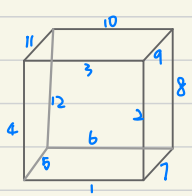
$$\frac{\partial \beta}{\partial a_x} = b_x \quad , \quad \frac{\partial \beta}{\partial a_y} = b_y \quad \frac{\partial \beta}{\partial a_z} = b_z$$

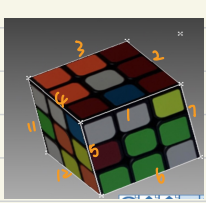
$$\frac{\partial \beta}{\partial b_x} = a_x \quad , \quad \frac{\partial \beta}{\partial b_y} = a_y \quad \frac{\partial \beta}{\partial b_z} = a_z$$

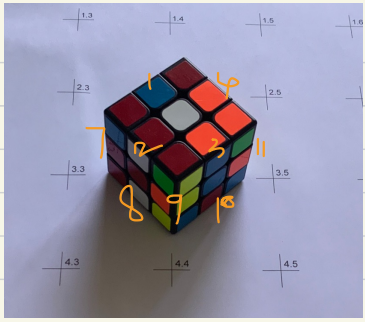
$$\Rightarrow \sigma_\beta^2 = b_x^2\sigma_{a_x}^2 + b_y^2\sigma_{a_y}^2 + b_z^2\sigma_{a_z}^2 + a_x^2\sigma_{b_x}^2 + a_y^2\sigma_{b_y}^2 + a_z^2\sigma_{b_z}^2$$

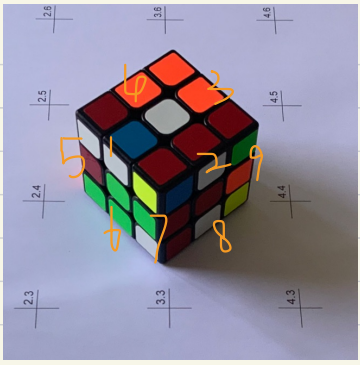
$H_0 : \beta = 0$

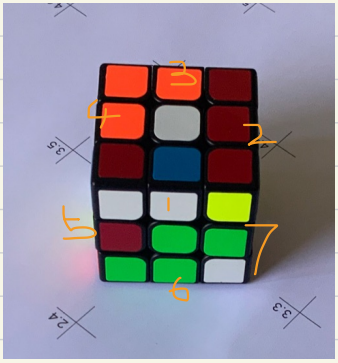
$$\frac{\hat{\beta} - 0}{\sigma_\beta} \sim Z$$

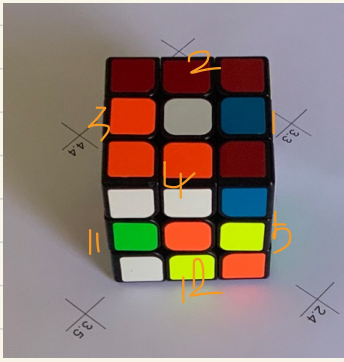


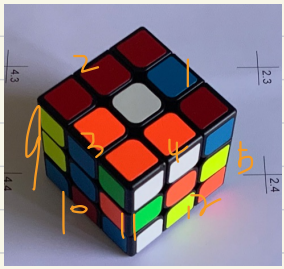


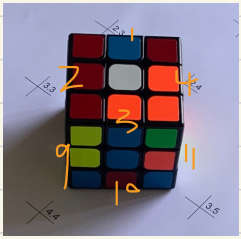


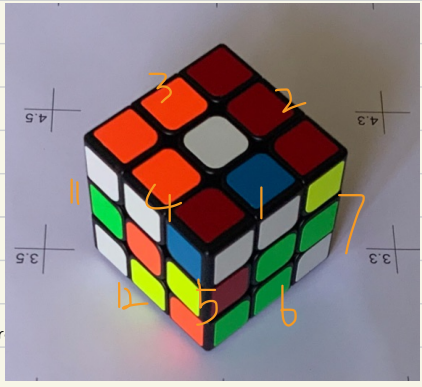






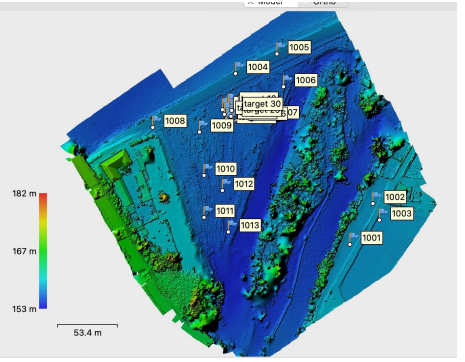






3. Compare the surface of each side: check if the area of six sides ar

Lab: Drone photogrammetry



2021.10.28 Design of the drone survey

Aim: survey of a portion of the Muzza Channel and of the Adda River, close to Cassano d'asda.
- reconstruction of the model to extract data for hydraulic modeling (evaluating the risk of flooding).
- final goal: extract a three-dimensional model of the area in order to extract DTM or vertical-cross section for 2D or 1D hydraulic analysis.

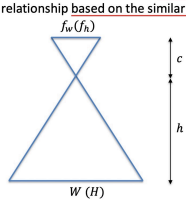
The required accuracy for the designing phase is about 5-8 cm in the vertical direction (according to modeling accuracy in hydraulic computations).

Instruments: drones, camera, GCP Markers, GPS Receivers

Flight Plan

FLIGHT PLAN: GSD

This relationship is equal to compute directly the GSD considering the following relationship based on the similar triangles



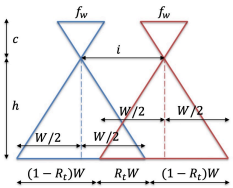
$$\frac{W}{h} = \frac{f_w}{c} \quad ① \quad W = f_w \cdot \frac{h}{c}, \quad H = f_h \cdot \frac{h}{c}$$

② $$GSD = \frac{W}{n_{px}} = \frac{h\, f_w}{c\, n_{px}} = \frac{h\, f_h}{c\, n_{py}}$$

where:
- h relative flight height [m]
- c principal distance [mm]
- $f_w(f_h)$ sensor size [mm] along width (length) direction
- $W(H)$ image footprint along width (length) [m]
- $n_{px}(n_{py})$ number of pixels along sensor width/length direction

几何标准差（Geometric Standard Deviation）

FLIGHT PLAN: BASELINE AND INTERAXIS



Equality condition:

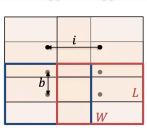
$$2\frac{W}{2} + i = 2(1-R_t)W + R_tW$$

$$i = 2W - 2R_tW + R_tW - W$$

③ $$i = (1-R_t)W$$

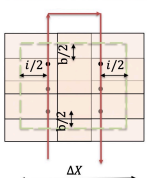
and analogously:

$$b = (1-R_\ell)H$$



where R_t and R_ℓ are the transversal and longitudinal overlapping, respectively

FLIGHT PLAN: BASELINE AND INTERAXIS



④ **NUMBER OF STRIPS:**
- along X direction: $n_x = \left\lceil \frac{\Delta X}{i} \right\rceil$
- along Y direction: $n_y = \left\lceil \frac{\Delta Y}{b} \right\rceil$

With the number of strips (integer) recompute the baseline and interaxis:

⑤ $$\bar{i} = \frac{\Delta X}{n_x} \qquad \bar{b} = \frac{\Delta Y}{n_y}$$

consequently, the true overlapping

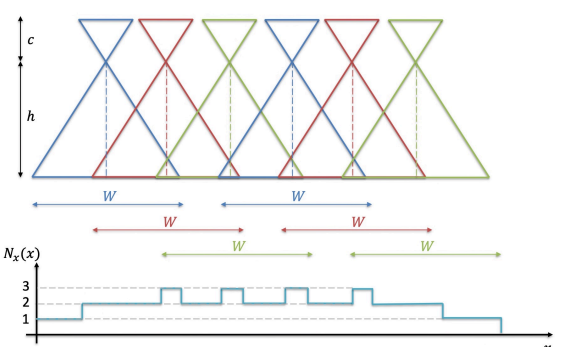
$$\bar{i} = (1-\bar{R}_t)W$$

$$\bar{b} = (1-\bar{R}_\ell)H$$

⑥ $$\bar{R}_t = 1 - \frac{\bar{i}}{W} \qquad \bar{R}_\ell = 1 - \frac{\bar{b}}{H}$$

⑦ $x_0 = (\bar{i}/2 : \bar{i} = \Delta X)'$
$y_0 = (\bar{b}/2 : \bar{b} = \Delta Y)'$

FLIGHT PLAN: IMAGE COVERAGE



FLIGHT PLAN: EXPECTED ACCURACY

Recalling the normal case accuracy:

$$\sigma_{z,t} = \frac{h^2}{c\,\bar{\imath}}\sigma_{P_\xi} \qquad \text{where: } \sigma_{P_\xi} = \sqrt{2}\sigma_\xi,$$

the computed true interaxis $\bar{\imath} = (1 - \bar{R}_t)W$ and the image footprint $W = \frac{f_w}{c}h$, we obtain:

$$\sigma_{z,t} = \frac{h^2}{c\,(1-\bar{R}_t)\frac{f_w}{c}h}\sigma_{P_\xi} = \frac{h\sqrt{2}}{(1-\bar{R}_t)f_w}\sigma_\xi$$

or, analogously, $\sigma_{z,\ell} = \frac{h\sqrt{2}}{(1-\bar{R}_\ell)f_h}\sigma_\eta$ on the other side

FLIGHT PLAN: EXPECTED ACCURACY

In general, we can assume that we are computing the Z coordinate as the mean of the Z_i (from along track couples) and Z_j (from cross track couples) estimates obtained from all the couples of images covering each x, y location:

$$Z(x,y) = \frac{\sum_{i=1}^{N_x(x)-1} Z_i + \sum_{j=1}^{N_y(y)-1} Z_j}{(N_x(x)-1) + (N_y(y)-1)}$$

// Average them

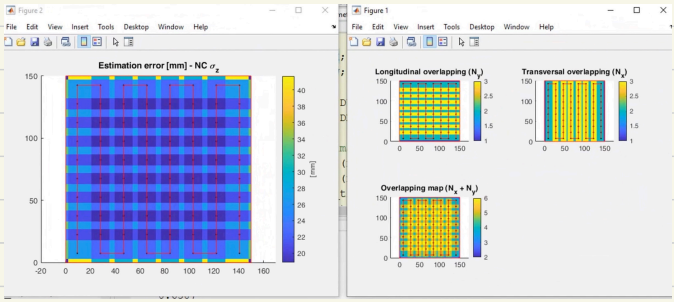
Therefore, by applying the covariance propagation law, we obtain:

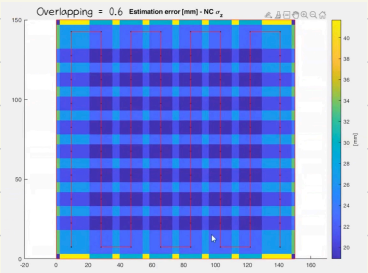
$$\sigma_Z(x,y) = \frac{\sqrt{(N_x(x)-1)\sigma_{Z,t}^2 + (N_y(y)-1)\sigma_{Z,\ell}^2}}{(N_x(x)-1) + (N_y(y)-1)}$$

summary

1. Find some geometrical parameters: baseline, height of flight, and so on
2. To calibrate them based on the accuracy you want to reach.
   - in normal case, you know how to propagate the calibration error from two photos to the accuracy of the vertical position
   - instead, you have two photos along longitudinal direction, and two photos between stripes (横向两张，纵向两张). so we have two different accuracy by a couple of images.
   - we have many couples of images, therefore, average all these couples to get the formula above.

matlab part: use Planning_empty.m to calculate the corresponding parameters





The error in borders is largest.

Could increase a little bit of overlapping.

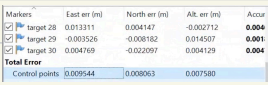
2021.11.11 + 11.12 Photogrammetric drone survey data processing

calibration

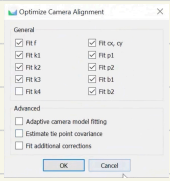
// see the beginning of recording in 2021.11.12

the process is same as the step 1~3, 5~10 in "Orientation", except that:

1. Using the images in calibration
2. Only GCP in calib_GCP_U00-32.txt is needed
3. The residual is in mm level.

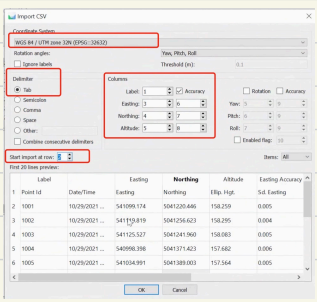


After the procedure, Tool -> Camera Calibration -> Adjusted, you can see the calibration result



Orientation

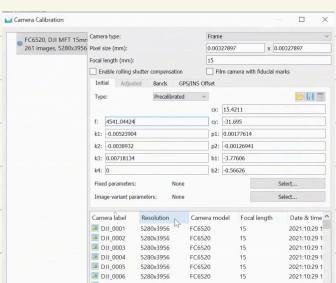
1. Import all photos in flight
2. uniform the reference system: convert reference -> projected coordinate system -> World Geodetic System 1984 ensemble -> WGS84/UTM zone 32N (from latitude/longitude/height to easting/northing)
3. import GCP coordinates: import reference -> choose GI_GCP_U00-32.txt



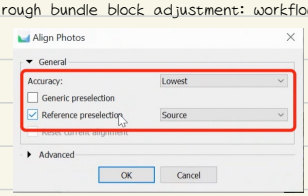
// after press OK, if it asks "Can't find match for '1001' entry. Create new marker?", choose "Yes to All"

// load also calib_GCP_U00-32.txt

4. Import camera certificate: Tools -> Camera Calibration
   1) open: x5s_gl.xml
   2) fix: Fixed parameters -> Select -> Check all
   3) right click "FC6250.....", choose "distortion plot"



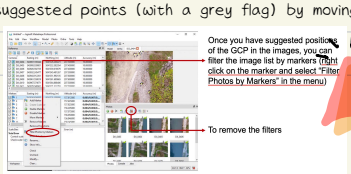
5. Start collimate points. We can place the maker manually, but it is the best way. Instead, since we have some priors (coordinates, ...), we can make use the functionalities offered by the software
   1) rough bundle block adjustment: workflow -> align photos...



// suggest to run in "medium" accuracy at home

- A green flag will appear and its position can be adjusted by moving it with the mouse
- If the points are suggested they are represented with a grey flag down

   2) collimate the suggested points (with a grey flag) by moving them to the right position (center of the marker)
   - SUGGESTION:

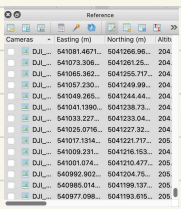


Tools/Markers/attach marker + refine marker

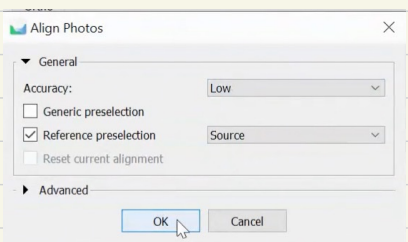
note: also align the "target points". Before doing it, do "align photos" with "reset ..." checked, to easy your jobs.

6. not all the points are used for Bundle Block Adjustment, use some of them as check points to validate the adjustment: divide all the points (GCP+target) into two groups so that we can compare the coordinates estimated by photogrammetry, and the coordinates estimated independently by GPS.
- How many checkpoints do we need? 20~30%
- How to choose the checkpoints?
    * for GCP: uniformly random choose. For example, 1003, 1006, 1012
    * for targets: since we have a lot of them in one area, we can use two of them as GCPs (for example, target2, target 17 and target22), and rest of them as checkpoints. Because basically, once you introduce a GCP, you are somehow constraining your model. Therefore, introducing many points in a single area means constraining your model to be very good in this area, which means that area would weight too much in the solution comparing with other areas.

7. uncheck all images , which means the information offered by GPS is not used in BBA. beacause the GPS information is low quality
- note that these information is useful in the beginning because it allows us to find the GCPs.



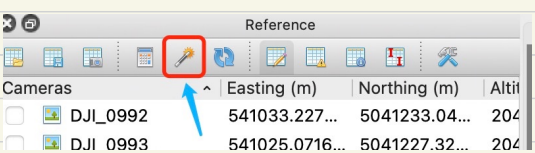
8. Do the Bundle Block Adjustment: workflow -> align photos



// remember to check "Reset current alignment"

Note that in the dataset of GI, image "DJI_0968" is corrupted. so we can disable or remove it.

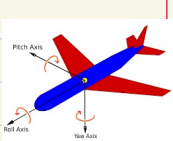
9. (BBA continue) optimize the solution based on GCP:

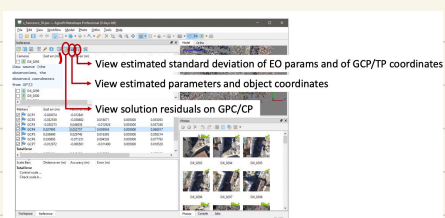


Note that because we use a fixed calibration certificate, "General" parameters cannot be choosen.

but we still can "estimate tie point covariance", which is useful to understand how the quality is distributed in space.

10. Evaluation (verify) the solution // recording: 2021.11.11 02:26:21



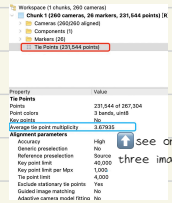


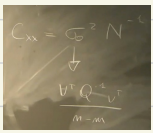
The estimated:
- are the angles reasonable according to our instrument? (02:28:36). "Roll set" should around 1 degree. If no, check the aligned point again, and do BBA again.

note: the instrument is very good, but we still have to perform photogrammetry process because there is an error in maintaining the camera along exactly vertical due to the fact that you have some engines that move the camera. Therefore, it is pseudo nadiral.

The error (residual):
1) the difference between the GPS and the estimated coordinates
- what is the accuracy of GPS? Almost 1 centimeter
- how much is the GSD? 1 centimeter
- if the accuracy for concentimetercheckpoint too much or too low w.r.t the expected accuracy and to our data?
    * if you got the few centimeter error, it is reasonable according to our input data. On one hand, we are not able to collimate better than 1 centimeter because it is the biggest size. On the other hand, the GPS accuracy is at the level of 1 centimeter.
    * so if you got 2 centimeters around this means that could be 1 centimeter error in the GPS plus 1 centimeter in the collimation. therefore, you can declare that your work is fine because you're interested in reprojection.
2) error in pixel: can be used to identify the outlier. So you need to check the point which have large pixel error.

The covariance:
- is the variance reasonable?

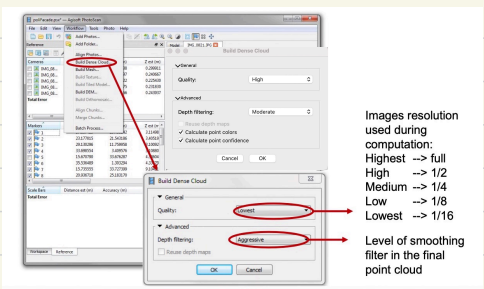




The standard deviations of GCP/TP in east/north/altitude should be small, because we have many tie points

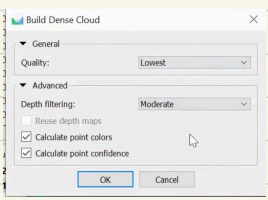
see one point in almost three images

11. Compute the point cloud



Images resolution used during computation:
Highest --> full
High --> 1/2
Medium --> 1/4
Low --> 1/8
Lowest --> 1/16

Level of smoothing filter in the final point cloud

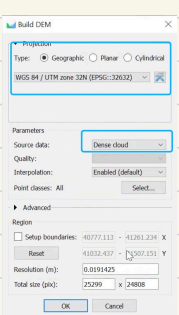


// the result cannot be used directly because it is not continuous

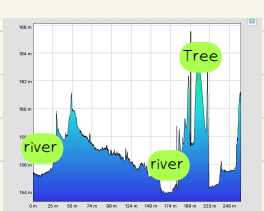
11. (SKIP) Build the TIN model: Workflow -> Build Mesh …

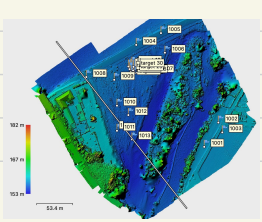
12. (SKIP) Project the images on the TIN model: workflow -> build Texture…

13. Build the DEM: workflow -> build DEM …
- how to choose the resolution?
    * Depends on GSD (larger than the size of GSD) because the maximum resolution you can get is GSD. But note that it is better to not set the resolution as the GSD because larger than GSD allows you to reduce the impact of outliers (smooth)
    * if GSD = 1 cm, choose 3~10 cm depends on the number of pixels you want.

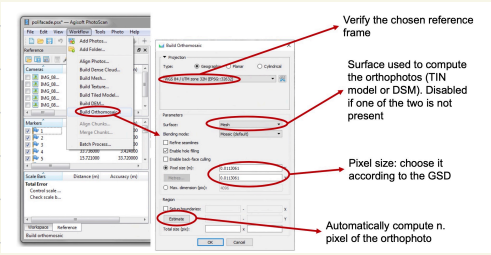


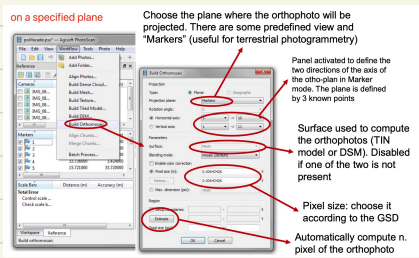
13.1 analyze the DEM (cross-section)
1) draw a line cross
2) right click the endpoint of the line and choose "Measure"

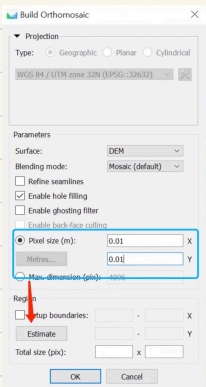




14 compute the Mosaicked Orthophoto







using the generated orthomosaic, you can

- compute the area and the volume of concrete: draw an area and measure
    * for volume:

15. Delivery

-