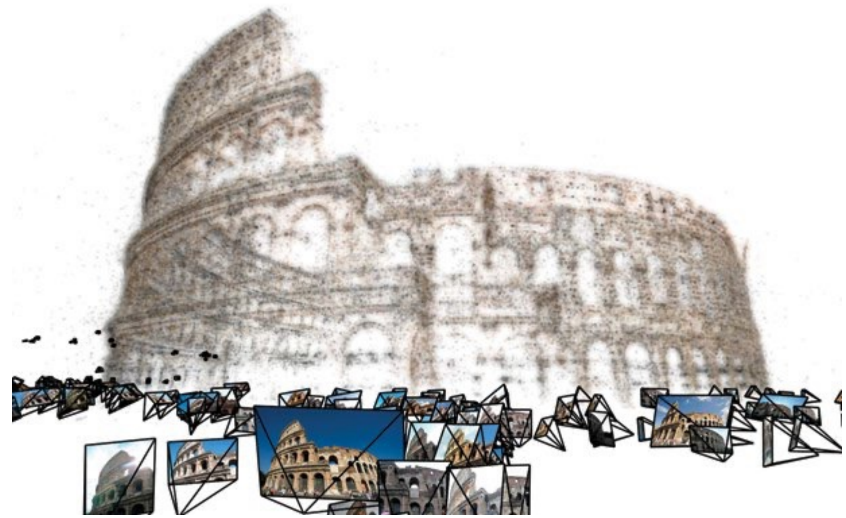
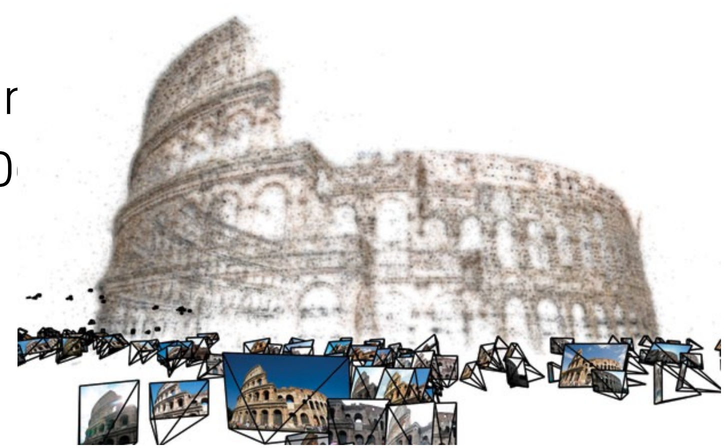


Structure from Motion

Programming Assignment 1

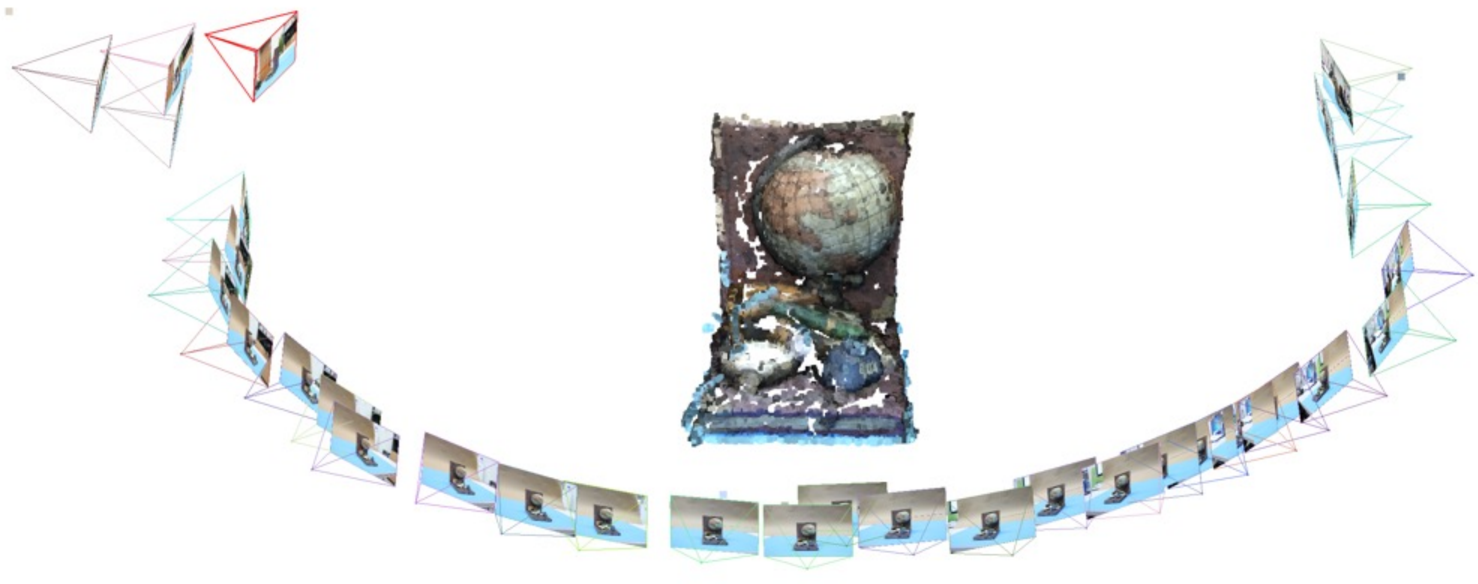
What is SfM?

- Structure from Motion (SfM)
  - The process of **estimating three-dimensional structures from two-dimensional image sequences** which may be coupled with local motion signals.
  - Input: Freely taken images with overlapped scenery
  - Output: camera pose and 3D structure of the scene
  - Reference
    - http://photosynth.net
    - N.Snavely et al., "Photo Tourism: Explorir
      photo collections in 3D", SIGGRAPH 200



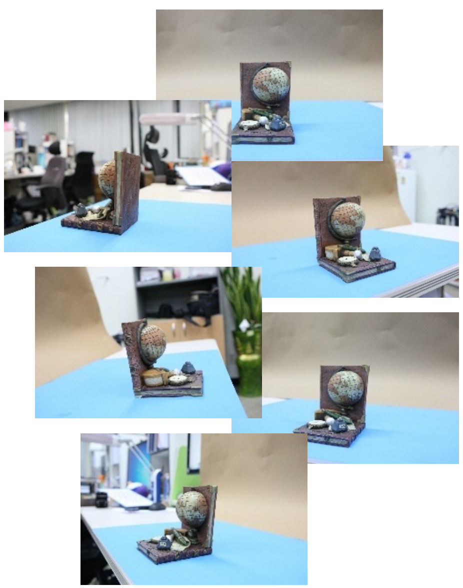
Output

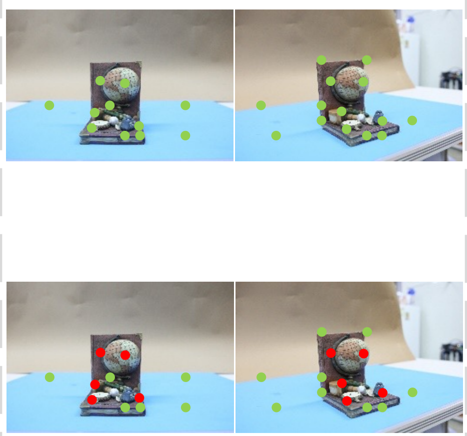
- Goal: Build a 3D & Estimate camera poses, given the set of images

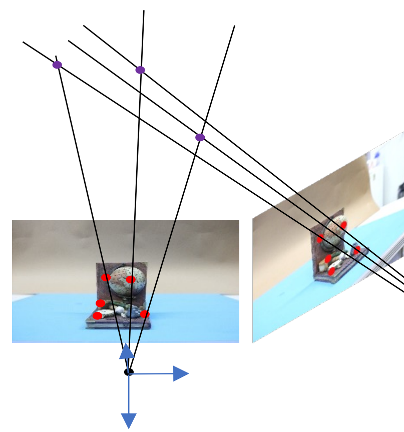


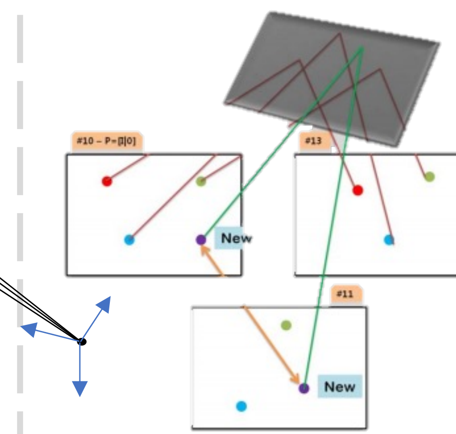
Overall Strategy

A set of images → Feature Extraction & Matching → (Two-view) Initialization Step → (> Three-view) Growing Step









[2] Hartley, Richard, and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
[3] Szeliski, Richard. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

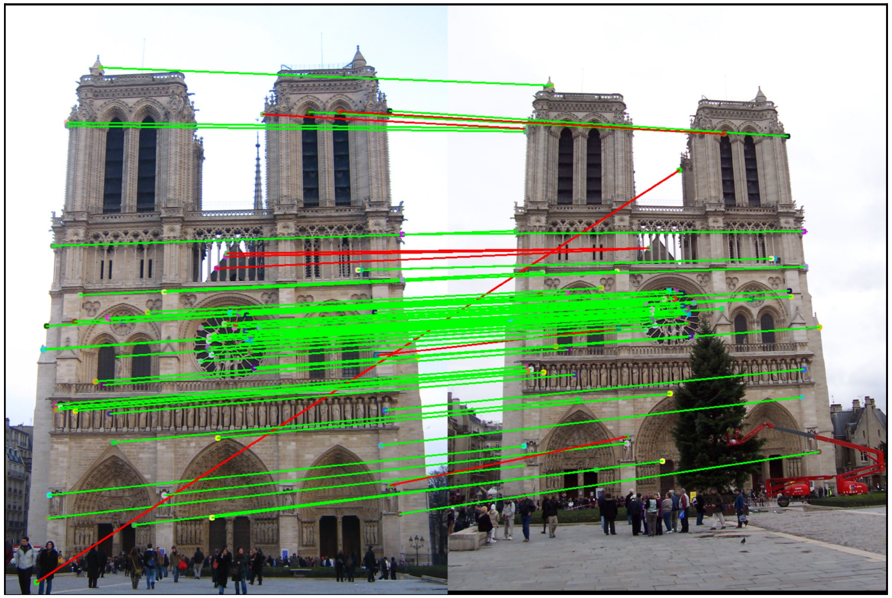
Overall Strategy

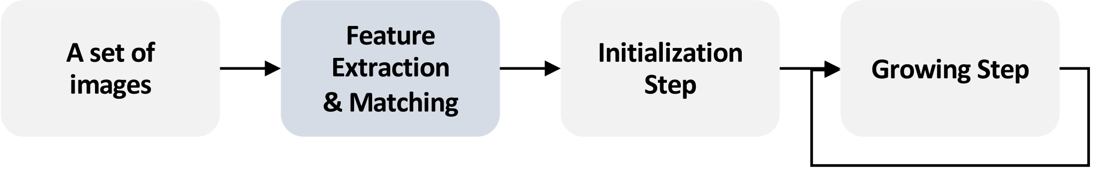
- Correspondence search, Relating images
  - 1. Extract SIFT from every image and find putative matches
  - 2. Outliers should be rejected by applying RANSAC
- Initialization Step
  - 3. Find the best image pair(simply, has the maximum matches or take the base-line into account)
  - 4. Estimate motion(R and t) and Reconstruct 3D points for the selected image pair. The camera coordinate of one camera is used for the world coordinate.
- Growing Step
  - 5. Search images which have enough points seeing the reconstructed 3D point
  - 6. Compute pose(R and t) for those images and reconstruct more 3D points seen from more than two images
  - ~~Bundle Optimization~~
- Repeat the Growing step until every camera is included.

Step I. Feature extraction & matching in PA1

[4] Lowe, David G. "Distinctive image features from scale-invariant keypoints."*International journal of computer vision* 60.2 (2004): 91-110.

- SIFT (Scale Invariant Feature Transform)
  - sift = cv2.xfeatures2d.SIFT_create()
  - keypoints, descriptor = sift.detectAndCompute(gray, None)





Step II. Essential matrix estimation

- Algorithm for Fundamental ($\mathbf{F}$) or Essential ($\mathbf{E}$) matrix
  - Un-calibrated (Unknown camera intrinsic $\mathbf{K}$)
    - Estimate $\mathbf{F}$ using 8-pts algorithm

      F = cv::findFundamentalMat (points1, points2,…)

    - Calculate $\mathbf{E}$ using intrinsic $\mathbf{K}$

      E = cv::sfm::essentialFromFundamental (F, K1, K2,…)

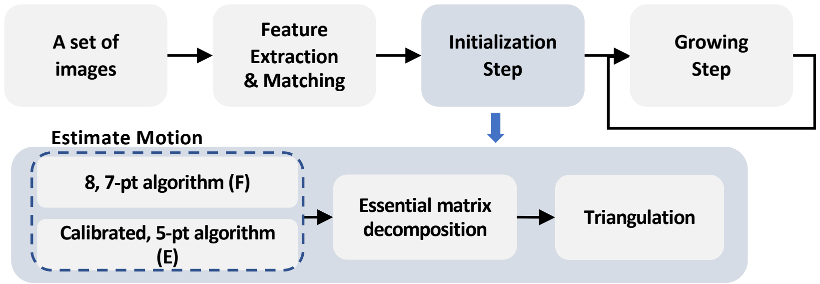
  - Calibrated (Known camera intrinsic $\mathbf{K}$)
    - Estimate $\mathbf{E}$ using 5-pts algorithm [7]

      E = cv.findEssentialMat(points1, points2, K, …)

$$\mathbf{E} = \mathbf{K}^T \mathbf{F} \mathbf{K}'$$

- $\mathbf{F}$ is fundamental matrix
- $\mathbf{E}$ is essential matrix
- $\mathbf{K}$ is intrinsic matrix
- $[\mathbf{R}|\mathbf{t}]$ is extrinsic matrix

+ with RANSAC !



[7] Nistér, David. "An efficient solution to the five-point relative pose problem." *TPAMI* 2004
[Opencv] https://docs.opencv.org/3.4/d7/d15/group__fundamental.html

Step III. Essential matrix decomposition

MVG 9.6 (p.257-p.260)
6.2.3 (p.162)

$$[t]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}$$

- Essential Matrix Decomposition to $[\mathbf{R}|\mathbf{t}]$
  - Essential matrix to camera matrix

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

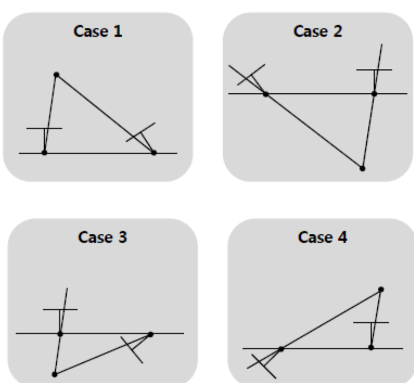
R1, R2, t = cv::decomposeEssentialMat(E,...)

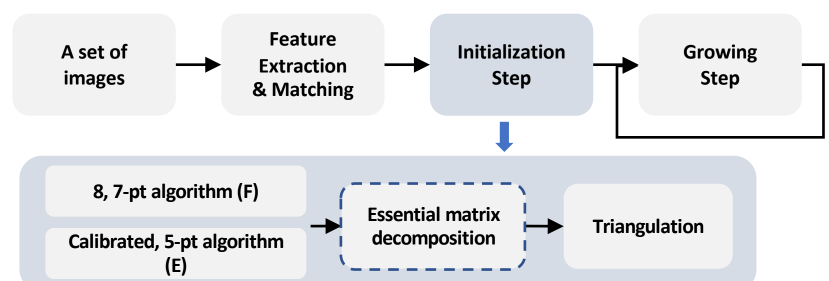
- Use SVD!

$$\mathbf{R} = \mathbf{UWV}^{\mathrm{T}} \text{ or } \mathbf{U}\boldsymbol{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}$$

$$\mathbf{t} = \boldsymbol{u}_3 \text{ or } -\boldsymbol{u}_3$$

- $SVD(E) = Udiag(1,1,0)V^T$
- $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
- $u_3 = U(0,0,1)^T$: The last column vector of U





Step IV. Triangulation

$\mathbf{X}$: 3D point
$\mathbf{x}$: Point on image coordinate
$\mathbf{K}$: Intrinsic matrix
$\mathbf{P}$ ($\mathbf{K}[\boldsymbol{R}|\boldsymbol{t}]$) : Extrinsic matrix

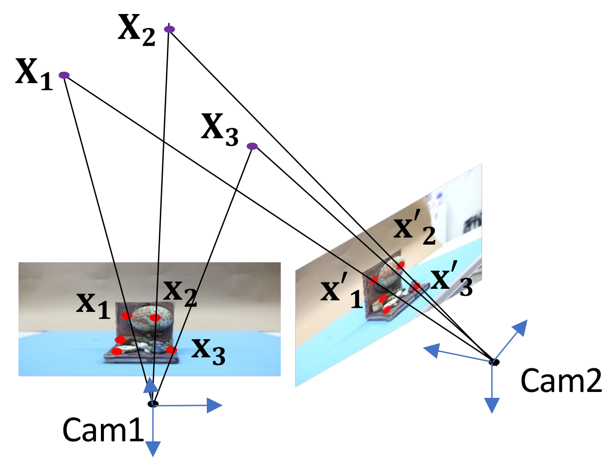
- Triangulation
  - Get 3D points from camera pose & correspondences

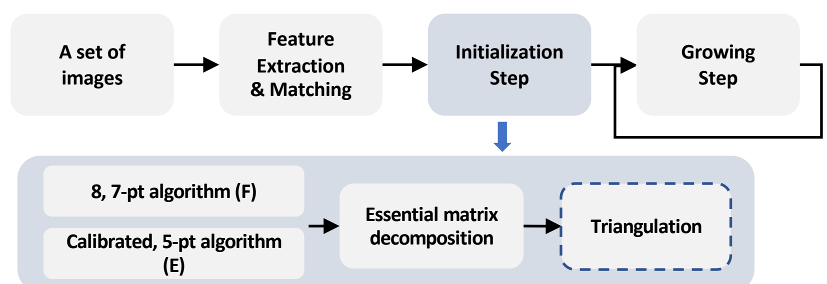
points3d = cv::sfm::triangulatePoints (points2d, projection_matrices,…)

$$\boldsymbol{x}_{ci} = P\boldsymbol{X}_i$$
$$[\boldsymbol{x}_{ci}]_\times P\boldsymbol{X}_i = 0$$

$$\begin{aligned} x(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{1T}\boldsymbol{X}) &= 0 \\ y(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{2T}\boldsymbol{X}) &= 0 \\ x(\boldsymbol{p}^{3T}\boldsymbol{X}) - y(\boldsymbol{p}^{1T}\boldsymbol{X}) &= 0 \end{aligned}$$

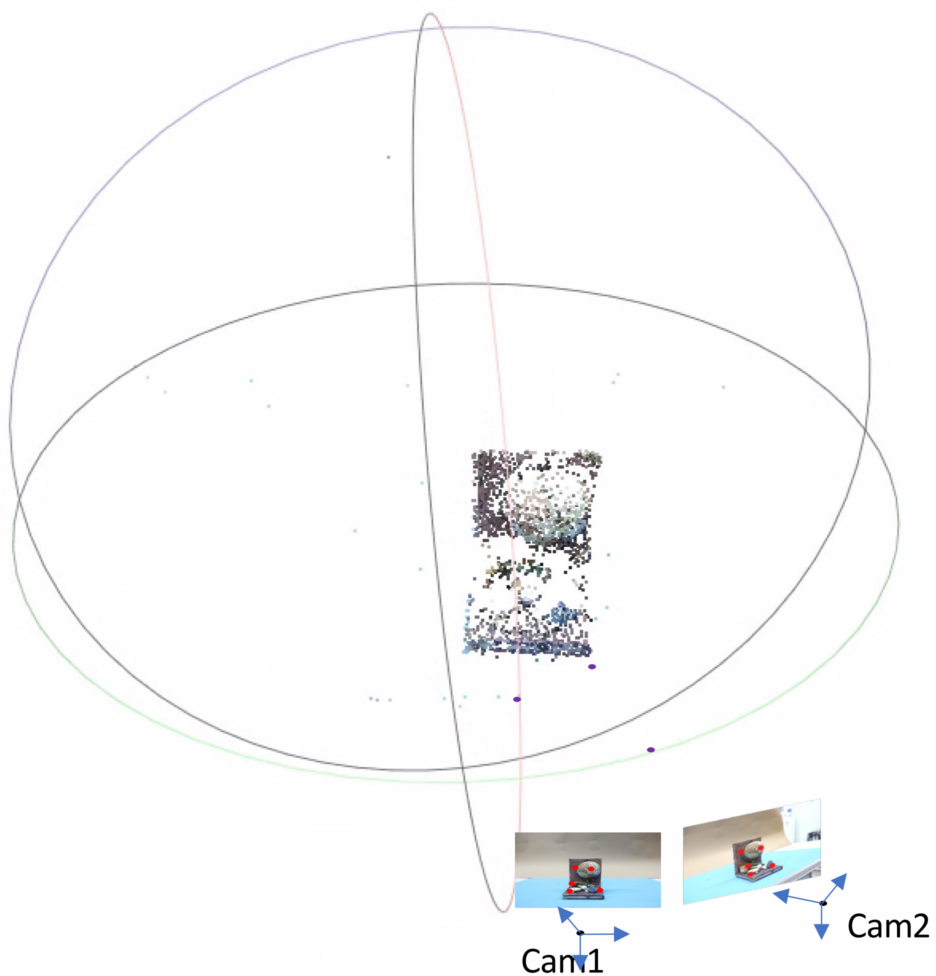
$$A\boldsymbol{X} = 0$$
$$A = \begin{bmatrix} x\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ y\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{2T} \end{bmatrix}$$





Output

- Goal: Build a 3D & Estimate camera poses, given the set of images



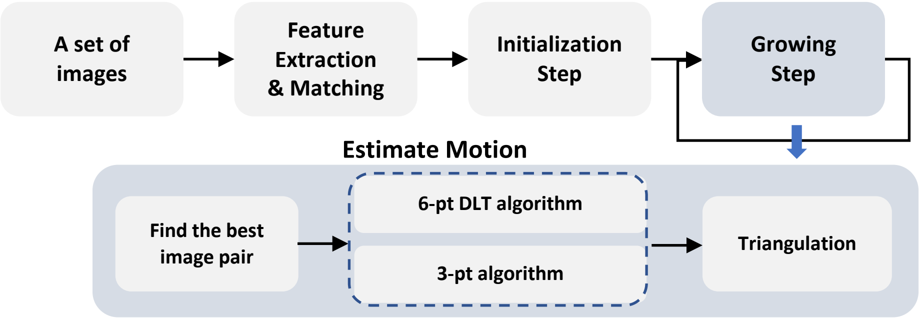
Step V. Growing step

- Estimate Camera matrix 'P' given a set of match points and 3D {x, X}
  - Perspective-n-Point (PnP) problem
    - The problem of estimating the pose of a calibrated camera given a set of n 3D points in the world and their corresponding 2D projections in the image.
    - When $n = 3$, PnP problem is in its minimal form of P3P and can be solved with three point correspondences.

R, t = cv.solvePnP(3D points, 2D point, K, distCoeffs, ...)

+ with RANSAC !

vs Epipolar geometry
(The geometric relation between two views)

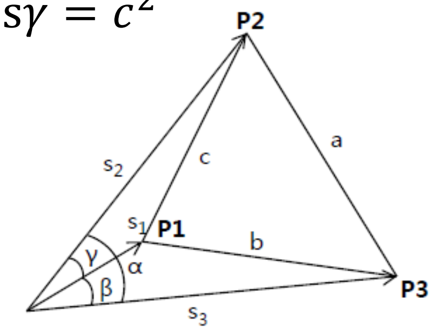


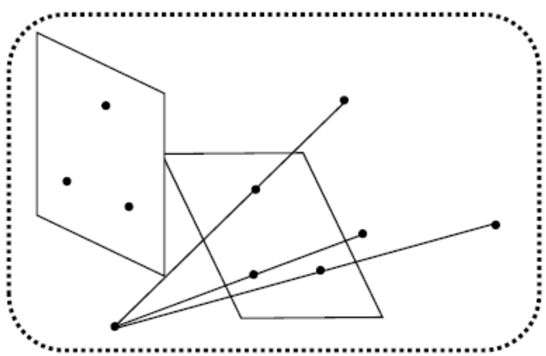
[8] Haralick, Bert M., et al. "Review and analysis of solutions of the three point perspective pose estimation problem." *IJCV* (1994)

Step V. Growing step

- 3-point algorithm

$$s_2{}^2 + s_3{}^2 - 2s_2s_3\cos\alpha = a^2$$
$$s_1{}^2 + s_3{}^2 - 2s_1s_3\cos\beta = b^2$$
$$s_1{}^2 + s_2{}^2 - 2s_1s_2\cos\gamma = c^2$$





j_i: Unit vector

$s_i = \|P_i\|$,

$P_i = s_i j_i, \qquad (i = 1,2,3)$

with the P_i, camera pose is linearly solved

Table I. The summary of characteristic of six solutions.

Authors	Features	Algebraic singularity
Grunert 1841	Direct solution, solve a fourth order polynomial	Yes
Finsterwalder 1903	Form a cubic polynomial and find the roots of two quadratics	Yes
Merritt 1949	Direct solution, solve a fourth order polynomial	Yes
Fischler and Bolles 1981	Another approach to form a fourth order polynomial	No
Linnainmaa et al. 1988	Generate an eighth order polynomial	No
Grafarend et al. 1989	Form a cubic polynomial and find intersection of two quadratics	Yes

Known : a, b, c, and unit vectors j_1, j_2, j_3

The **problem** is to determine the lengths s_1, s_2, s_3 from which the 3D vertex point positions P_1, P_2, and P_3 can be determined.

↓

Six solutions

Grunert(1841), Finsterwalder(1937), Merritt(1949), Fischler & Bolles(1981), Linnainmaa et al(1988), Grafarend et al(1989).

↓

Estimate Pose(P)

[8] Haralick, Bert M., et al. "Review and analysis of solutions of the three point perspective pose estimation problem." *IJCV* (1994)

Step V. Growing step

$\mathbf{X}$: 3D point
$\mathbf{x}$: Point on image coordinate
$\mathbf{K}$: Intrinsic matrix
$\mathbf{P}$ $(\mathbf{K}[\boldsymbol{R}|\boldsymbol{t}])$: Extrinsic matrix

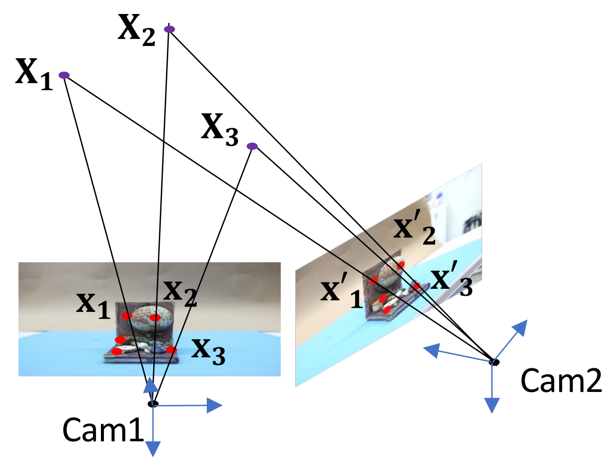
- Triangulation
  - Get 3D points from camera pose & correspondences

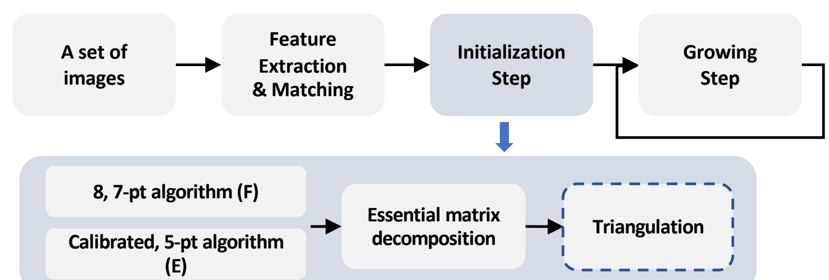
points3d = cv::sfm::triangulatePoints (points2d, projection_matrices,…)

$$\boldsymbol{x}_{ci} = P\boldsymbol{X}_i$$
$$[\boldsymbol{x}_{ci}]_\times P\boldsymbol{X}_i = 0$$

$$x(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{1T}\boldsymbol{X}) = 0$$
$$y(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{2T}\boldsymbol{X}) = 0$$
$$x(\boldsymbol{p}^{3T}\boldsymbol{X}) - y(\boldsymbol{p}^{1T}\boldsymbol{X}) = 0$$

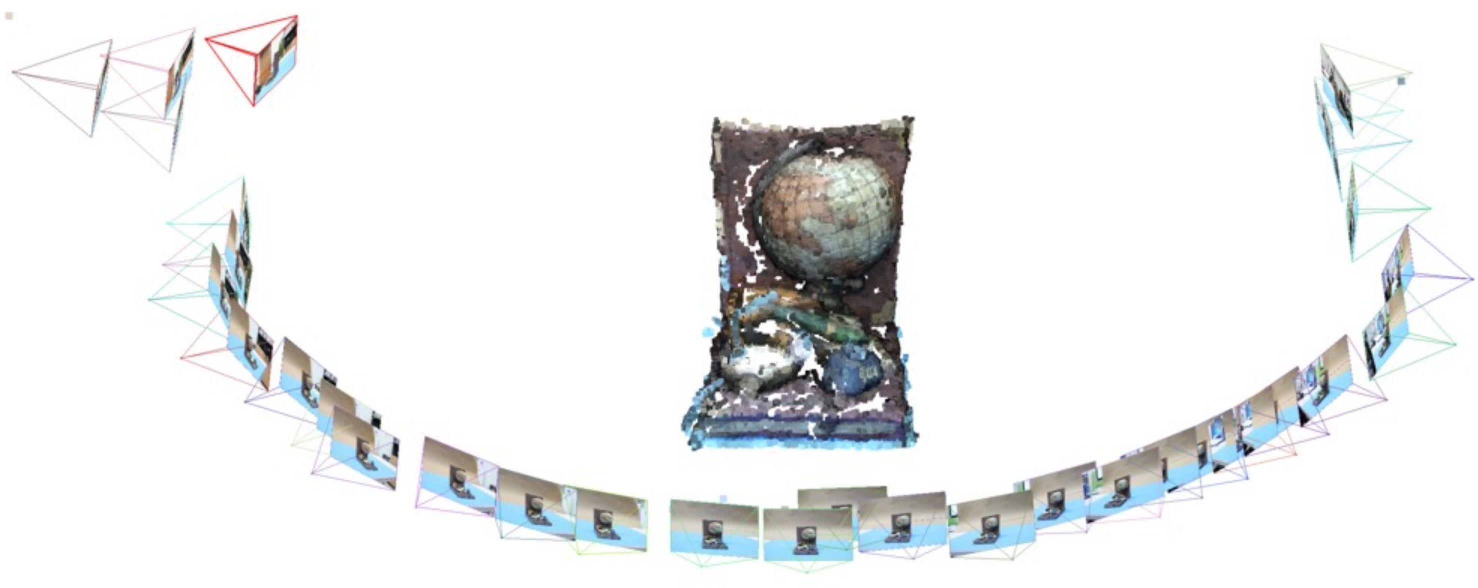
$$A\boldsymbol{X} = 0$$
$$A = \begin{bmatrix} x\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ y\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{2T} \end{bmatrix}$$





Output

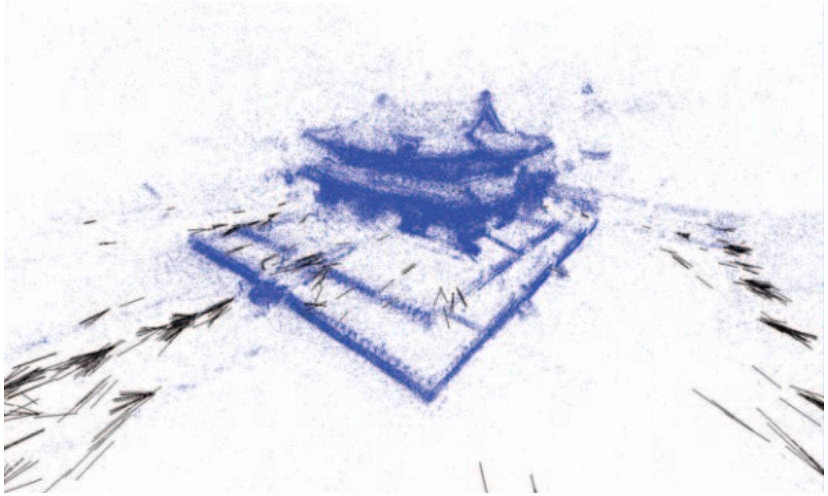
- Goal: Build a 3D & Estimate camera poses, given the set of images



~~Step VI. Optimization (you don’t need to implement this)~~

- Bundle adjustment
  - Refines a visual reconstruction to produce jointly optimal 3D structure and viewing parameters
  - ‘Bundle’ refers to the bundle of light rays leaving each 3D feature and converging on each camera center.

Before Bundle adjustment

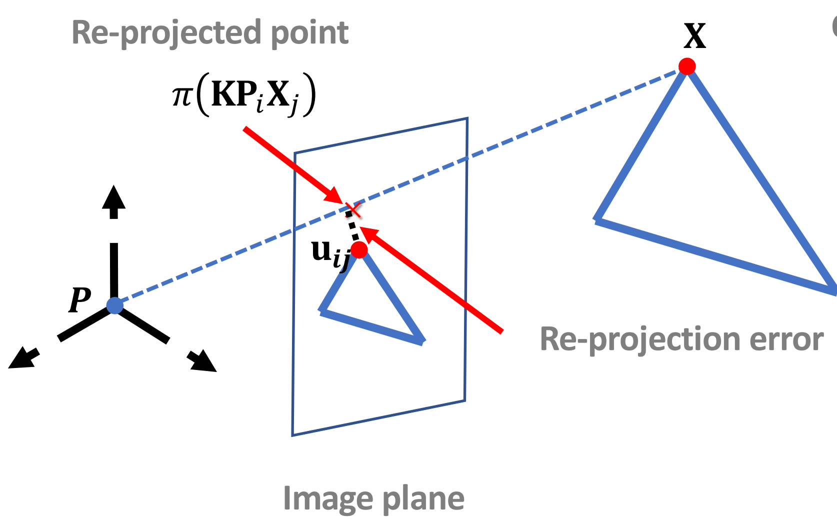


After Bundle adjustment

[6] Triggs, Bill, et al. "Bundle adjustment—a modern synthesis." *International workshop on vision algorithms*. Springer Berlin Heidelberg, 1999.
[7] Jeong, Yekeun, et al. "Pushing the envelope of modern methods for bundle adjustment." *IEEE transactions on pattern analysis and machine intelligence* (2012): 1605-1617.

~~Step VI. Optimization (you don't need to implement this)~~

- Bundle Adjustment's Mathematical Problem
  - Minimize re-projection error
  - Non-linear Least Square approach
  - Good approximate values are needed



Objective function

$$C(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} \parallel \mathbf{u}_{ij} - \langle \mathbf{KP}_i\mathbf{X}_j \rangle \parallel^2$$

n: The number of cameras
m: The number of features
$\pi(\cdot)$: The projection function ($\mathcal{R}^3 \rightarrow \mathcal{R}^2$)
w_{ij}: indicator variable
1 if visible, 0 other wise

http://ceres-solver.org/

Submission

- Submission should include…
  - Source code for Step I-V
  - Results (ply file) of 3D points & Camera position (Use meshlab tools)
  - Readme file explaining how to execute the program
  - Report includes
    - Results (3D points & Camera position screen capture)
    - Your understanding of each step (Step I to VI) in 2-3 sentences
    - Answer these questions
      - (Step II) What is RANSAC? and why should we use RANSAC?
      - (Step III) Why are four camera poses?
      - (Step III) Why should we apply $\mathbf{W}$ between $\mathbf{U}$ and $\mathbf{V}$ for $\mathbf{R}$
      - (Step V) What are the differences between epipolar geometry and PnP?
      - (Step VI) What is the difference between Gauss-Netwon method and Levenberg–Marquardt (LM) method (please describe in words, not equations)

Submission

- **Due : April 28, 11:59PM**
- **To : lms.dgist.ac.kr**

- Notice
  - [Delayed submission] The 25 score in 100 total will be degraded from the marked credit per day & only 3 days delayed submission allowed
    (e.g., 100 → 75 (1day) → 50 (2day) → 25 (3day) → 0)

Auxiliary References

- Multiple-View Geometry (HZ)
    - Basic Projective Geometry (ch. 2,3)
    - Camera Models and Calibration (ch. 6,8)
    - Epipolar Geometry and Implementation (ch. 9, 11)
    - Triangulation (ch. 12)

- Computer Vision: Algorithms and Applications (S)
    - Structure from Motion (ch. 7)

- Phillp Torr's Structure from Motion toolkit
    - Includes F-matrix estimation, RANSAC, Triangulation and etc.
    - http://cms.brookes.ac.uk/staff/PhilipTorr/

Thank you