- Large-Scale 3D Reconstruction

Given set of images $\{I_1, \cdots, I_N\}$, find
- motion of cameras w.r.t. F_W $\{P_1, \cdots, P_N\}$
- 3D scene points $\{X_1, \cdots, X_N\}$

- 3-step pipeline

1) Data association : check two images are related using image correspondences and robust two-view geometry

Keypoint / descriptor matching

RANSAC, pose estimation (E/F)

2) Structure-from-Motion (SfM) : initial 3D reconstruction by triangulation, then refine w/ bundle adjustment

PnP, EPnP

Gauss-Newton, Levenberg-Marquadt

3) Multi-view Stereo (MVS) : get dense 3D model using plane sweeping algorithm

Unstructured raw images $\xrightarrow{\text{Data association}}$ Scene graph $\xrightarrow{\text{SfM}}$ Sparse Model $\xrightarrow{\text{MVS}}$ Dense Model (obtain depth of every pixel)

- Data association

Connect images w/ overlapping views → Connected components represent a single 3D scene

↓ steps

1) Extract image keypoints / descriptors e.g. ORB, SIFT, SURF

2) Establish keypoint correspondences

3) Take geometric verification (find outliers) ∵ keypoints matching is purely based on appearance

↳ Given $x_i \leftrightarrow x_i'$ correspondences, take RANSAC-based two-view geometry (E/F w/o outliers)

if inlier counts > threshold, image pairs added to scene graph

- Problem : these steps are computationally expensive!

for N images and K keypoints / image, querying 1 image $= O(NK^2)$

↓ Solution : use bag-of-words image retrieval

- Place Recognition / Image Retrieval

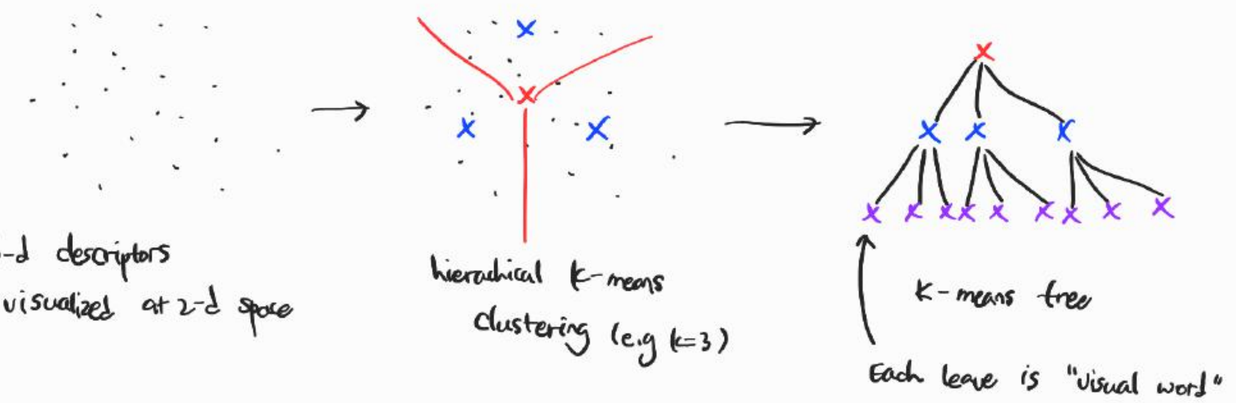
efficient tree-based search s.t. query image features → find associated image from database

- steps

1) Extract image keypoints and descriptors from training data

↳ can use crawled images for training

2) perform K-means hierarchical clustering on descriptors



3) Build inverted file index

∴ Feature points of training data no longer needed!

For each image, extract image features and (tree-) search for closest leaf nodes

Build frequency count for each image

e.g. × × × × × × × × ×

1 2 0 0 0 1 3 1 5 Image #1 counts

4) Given query image, retrieve visually similar image

e.g. × × × × × × × × ×

1 2 0 0 0 1 3 1 5 Image #1 counts

↕ ↕ ↕ Match = 3

1 1 1 2 0 1 0 0 0 Query image counts

* frequency counts can be normalized i.e high frequency node ↓ weight

∵ non-discriminative node

- Time complexity

For K image features per image, N # of images, b/L width/depth of K-means tree,

[Naive data association : $O(NK^2)$
[Image retrieval : $O(KbL)$ → Much lower!

∴ Data association [Take image retrieval to eliminate infeasible edges in scene graph
[Take geometric verification for remaining edges

- SfM Paradigms

1) Incremental 2) Global 3) Hierarchical

- Incremental SfM

- Initialization

1) choose two non-panoramic views ($\|t\| \neq 0$) w/ highest inlier keypoint matching → Seed pair

∵ difficult to derive H, E, F from pure rotation ↳ ∵ ↑ accuracy for least-square problem

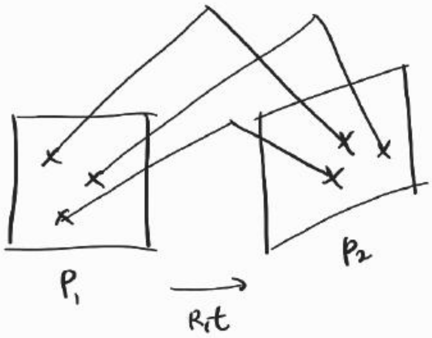
2) Take 4-point algorithm (planar scene, H) or 8-point algorithm (non-planar scene, E/F)

→ Decompose to R, t or P, P'

3) Set scale of translation to 1 ($\|t\| = 1$ or $\|p'\| = 1$)

4) Triangulate inlier 2D-2D correspondences to 3D points

5) Apply bundle adjustment to refine camera matrices & 3D points



Initialization of first two views (P_1, P_2)

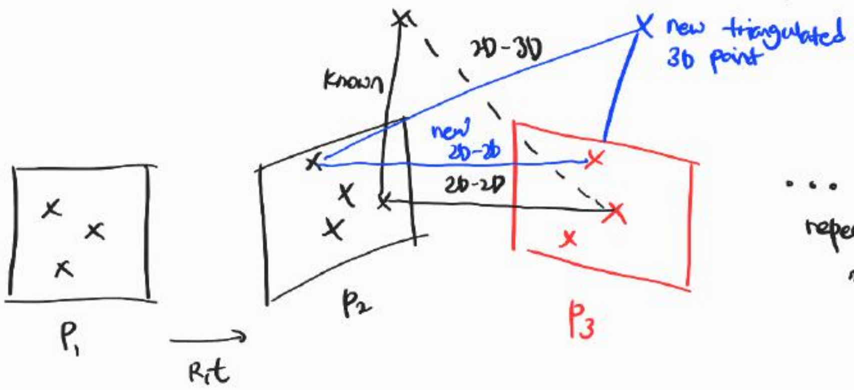
- Subsequent views

1) Find 2D-2D correspondence of existing-new views

2) Since 3D point of existing view is known, find 2D (of new view) - 3D correspondences

3) Solve PnP to find P_3

4) Take 2D-2D correspondences (that x has matching at P_1) to triangulate new 3D points



... repeat w/ many other views

- Refinement stage

$$\min_{P, X} \|x - \pi(P, X)\|$$

Multi-view non-linear optimization using bundle-adjustment

⇒ Minimize reprojection error

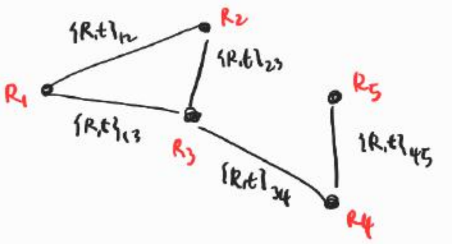
- Global SfM

1) Compute relative poses $\{R, t\}_{ij}$ of all edges in scene graph (by decomposing calculated E, F)

2) Estimate global rotations

$$\min_R \|R_{ij} - R_i R_j\|$$

$$R \in SO(3) \underset{[\omega]_\times = \log(R)}{\overset{R = \exp([\omega]_\times)}{\rightleftarrows}} \omega \in \mathbb{R}^3 \quad ([\omega]_\times \in so(3))$$



Scene graph

$\{R, t\}_{ij}$: known relative

R_i, t_i : unknown global

Then, $R_{ij} = R_i R_j^T \rightarrow \omega_{ij} = \omega_j - \omega_i = \begin{array}{c} \begin{bmatrix} 0 & 0 & \cdots & -I & \cdots & I & \cdots & 0 \end{bmatrix} \\ \qquad\qquad\quad ith \quad\; jth \end{array} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix} = A\omega_{global}$

$\therefore$ Least-squares problem : $A\omega_{global} = \omega_{rel}$ where stacked $\omega_{rel} \in \mathbb{R}^{3\times(\#\text{ of relative motion})}$

3) Estimate global translations

$$\min_t \left\| t_{ij} - \frac{t_i - t_j}{\|t_i - t_j\|} \right\| \longrightarrow Ax = b \text{ problem}$$

Relative t_{ij} normalized s.t. $\|t_{ij}\| = 1$ $\because$ up-to-scale

4) Triangulate to 3D using R_i, t_i & refine using bundle adjustment

- Hierarchical SfM

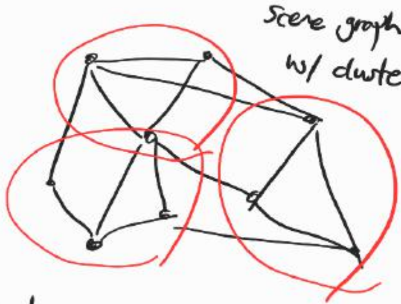
1) Hierarchical clustering of scene graph

2) Reconstruct clusters independently (Incremental / Global SfM)

3) Merge clusters using similarity transformations

↳ absolute orientation (R,t) + find scale using point correspondences

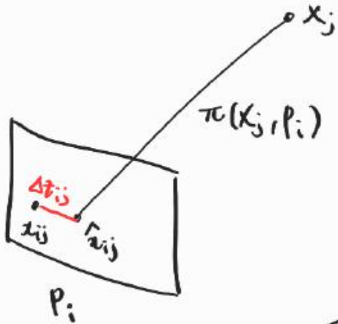
Scene graph w/ clusters



- Bundle Adjustment

↳ Refine step to jointly optimize 3D points X_j and camera poses P_i

$X_j \rightarrow$ 3D point was triangulated during initialization of SfM



<u>Cost function</u>

$$\underset{P,X}{\operatorname{argmin}} \sum_i \sum_j \| \underbrace{x_{ij} - \pi(P_i, X_j)}_{\Delta z_{ij}} \|^2_{\omega_{ij}}$$

where ω_{ij} : Measurement error covariance

$\pi(P_i, X_j) = \dfrac{P_i X_j}{\hat{x}_3}$ to make homogenous form $\begin{pmatrix} \hat{x}_1 / \hat{x}_3 \\ \hat{x}_2 / \hat{x}_3 \\ 1 \end{pmatrix}$

another form

$$\underset{P}{\operatorname{argmin}} \sum_i \sum_j \Delta z_{ij}^T W_{ij} \Delta z_{ij}$$

where P is $(12N \times 3M)$ vector w/ $12N$ $P_1 \cdots P_N$ parameters and $3M$ $X_1 \cdots X_j$ 3D points

- Iterative Estimation Methods

For non-linear function $f : X = f(P)$ where $X \in \mathbb{R}^N$ is measurement vector
$P \in \mathbb{R}^M$ is Euclidean parameter vector

↓ objective

Find $\hat{P}$ most nearly satisfies X closest to true value $\bar{X}$

e.g. Bundle Adjustment $\left[\begin{array}{l}\text{Measured value } X : \text{observed image coordinate } x_{ij} \\ \text{Parameter } \hat{p} : P, X\end{array}\right.$

Redefine objective : minimize error ε

Find $\hat{p}$ s.t. $X = f(\hat{p}) - \varepsilon$ or $\underset{\hat{p}}{\operatorname{argmin}}\, g(\hat{p}) = \frac{1}{2}\|\varepsilon\|^2 = \frac{1}{2}\varepsilon^T\varepsilon$

- Iterative estimation methods

1) Newton's Method

initial estimated value $p_0 \rightarrow$ error $g(p_0)$

Taylor series expansion : $g(p_0+\Delta) = g(p_0) + g_p\Delta + \Delta^T g_{pp}\Delta/2 + \cdots$

where $g_p = \left.\frac{\partial g}{\partial p}\right|_{p=p_0}$, $g_{pp} = \left.\frac{\partial^2 g}{\partial p^2}\right|_{p=p_0}$

* Jacobian and Hessian Matrices

For $f(x) : R^n \rightarrow R^m$ $\left[\begin{array}{l}\text{Jacobian } J : \text{1st order partial derivatives} \\ \text{Hessian } H : \text{2nd order ————}\end{array}\right.$

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in R^{m\times n} \quad \text{or} \quad J_{ij} = \frac{\partial f_i}{\partial x_j}$$

$$H = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \in R^{n\times n} \quad \text{or} \quad H_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j}$$

Then, for $p_1 = p_0 + \Delta$:

$g(p_1) = g(p_0) + \underbrace{g_p}_{J}\Delta + \Delta^T \underbrace{g_{pp}}_{H}\Delta/2$ $\because$ Taylor series expansion

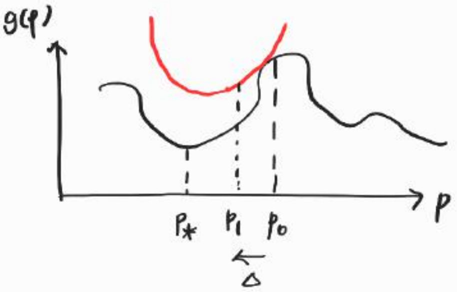
By differentiating w.r.t. Δ :

$0 = g_p + g_{pp}\Delta \rightarrow g_{pp}\Delta = -g_p$ (inhomogeneous linear equation $Ax=b$ form)

Solve for Δ by g_{pp}^{+} or least-square approach

$\therefore$ Iterate $p_{i+1} = p_i + \Delta$ until convergence

- Remarks



Assumption : approx. Quadratic cost function near minimum

$\therefore$ If p_0 far from p_*, assumption fails $\rightarrow$ slow / fail convergence

$\ominus$ $\uparrow$ Computational cost calculating H

2) Gauss - Newton method

$g(p) = \frac{1}{2}\|\varepsilon(p)\|^2 = \varepsilon(p)^T\varepsilon(p)/2$

$\frac{dg}{dp} = g_p = \varepsilon_p^T\varepsilon$ where $\frac{d\varepsilon}{dp} = \varepsilon_p = \frac{d(x - f(p))}{dp} = \frac{df(p)}{dp} = f_p = J$

$\therefore g_p = J^T\varepsilon$

$\frac{d^2g}{dp^2} = g_{pp} = \varepsilon_p^T\varepsilon_p + \varepsilon_{pp}^T\varepsilon \simeq \varepsilon_p^T\varepsilon_p = J^TJ$

Hessian matrix that is computationally expensive
since second - order ε_{pp} is negligable, ignore it!

$\Rightarrow g_{pp}\Delta = -g_p$ becomes $J^TJ\Delta = -J^T\varepsilon$ "normal equation"

If weighted cost function i.e. $g(p) = \frac{1}{2}\|\varepsilon(p)\|_\Sigma^2 = \varepsilon(p)^T\Sigma\,\varepsilon(p)/2$:

$\underline{J^T\Sigma^{-1}J}\Delta = -J^T\Sigma^{-1}\varepsilon \rightarrow Ax = b$ form

symmetric, positive definite matrix

3) Gradient Descent

Recall $g_p = \varepsilon_p^T\varepsilon \rightarrow -g_p = -\varepsilon_p^T\varepsilon$ is most rapid decrease of cost function

$\therefore$ Iteratively decrease to negative gradient direction

↳ $\lambda\Delta = -g_p$ where λ controls step size

* Newton's method w/ $H \simeq \lambda I$ = Gradient Descent

$\ominus$ slow convergence $\oplus$ $\downarrow$ computational cost

4) Levenberg - Marquardt method $\simeq$ 2) + 3)

Recall normal equations $J^TJ\Delta = -J^T\varepsilon$

$\rightarrow$ Augmented normal equations : $(J^TJ + \lambda I)\Delta = -J^T\varepsilon$

- Updating λ

1. Initialization : $\lambda = 10^{-3} \cdot \mathrm{avg}(\mathrm{trace}(J^TJ))$
2. $(J^TJ + \lambda I)\Delta = -J^T\varepsilon \rightarrow$ calculate $\Delta \rightarrow$ update p $(p' \leftarrow p + \Delta)$

Compare $\varepsilon(p)$ and $\varepsilon(p')$

$$\left[\begin{array}{l} \text{if } \varepsilon(p) > \varepsilon(p') : \text{error} \downarrow, \text{ accept } p', \ \lambda' \leftarrow \lambda/10 \\ \text{else} : \text{error} \uparrow, \text{ reject } p', \ \lambda' \leftarrow 10\lambda \end{array}\right.$$

- Justification of LM method

  (i) Small λ

  $(J^TJ + \cancel{\lambda I})\Delta = -J^T\varepsilon \longrightarrow$ Normal equation $J^TJ\Delta = -J^T\varepsilon$

  $\therefore$ Small λ = close to min = Gauss-Newton method fits well

  (ii) Large λ

  $(\cancel{J^TJ} + \lambda I)\Delta = -J^T\varepsilon \longrightarrow \lambda\Delta = -J^T\varepsilon = -g_p$

  $\therefore$ Large λ = far from min = Gradient Descent works well

- Sparse Levenberg-Marquardt algorithm

  Problem of L-M method : intractable for large # of parameters to optimize e.g. BA

  $\because$ Solving normal equation $\underline{J^TJ}\Delta = -J^T\varepsilon$ takes $O(N^3)$

  sparse matrix

  $\rightarrow$ find better method to calculate inverse!

  $\downarrow$ Solution: sparse L-M

  Partition $p \in \mathbb{R}^M$ into a, b s.t. $p = (a^T, b^T)^T$ $\rightarrow$ e.g. BA : camera/3D point parameter partition

  Then, $J = [\partial\hat{x}/\partial p] = [A|B]$ where $A = [\partial\hat{x}/\partial a]$, $B = [\partial\hat{x}/\partial b]$

  $\downarrow$ Normal equation $J^TJ\delta = -J^T\varepsilon$ into block form

$$\begin{bmatrix} U^* & W \\ W^T & V^* \end{bmatrix}\begin{pmatrix} \delta a \\ \delta b \end{pmatrix} = \begin{pmatrix} \varepsilon_A \\ \varepsilon_B \end{pmatrix}$$

  $\downarrow$ Multiplying $\begin{bmatrix} I & -WV^{*-1} \\ 0 & I \end{bmatrix}$ on both sides

$$\begin{bmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{bmatrix}\begin{pmatrix} \delta a \\ \delta b \end{pmatrix} = \begin{pmatrix} \varepsilon_A - WV^{*-1}\varepsilon_B \\ \varepsilon_B \end{pmatrix}$$

By making it to 0, first equation only depends to δa

$\therefore (U^* - WV^{*-1}W^T)\delta a = \varepsilon_A - WV^{*-1}\varepsilon_B \rightarrow$ Solve δa

$*$ $(U^* - WV^{*-1}W^T)$ is smaller & sparser than J^TJ from original normal equation

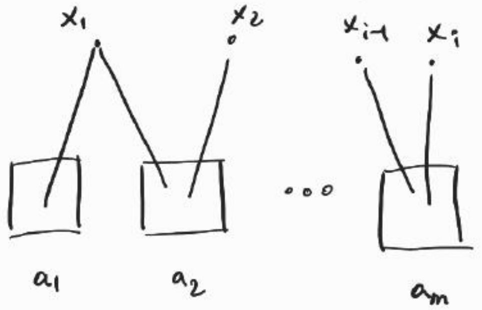
$W^T\delta a + V^*\delta b = \varepsilon_B \rightarrow$ solve δ_b

↓ update $p = ((a+\delta a)^T, (b+\delta b)^T)^T$

Again like L-M method
$\left[\begin{array}{l} \text{if } \varepsilon(p) > \varepsilon(p') : \text{ error}\downarrow, \text{ accept } p', \ \lambda' \leftarrow \lambda/10 \\ \text{else} : \text{ error}\uparrow, \text{ reject } p', \ \lambda' \leftarrow 10\lambda \end{array}\right.$

- Example : Bundle Adjustment

↳ Sparse L-M is advantageous due to lack of parameter interaction



$X = \{X_1, X_2, \cdots, X_i\}$ where $X_i = (x_{i1}^T, x_{i2}^T, \cdots, x_{im}^T)^T$

x_{ij} : image of X_i to j-th camera

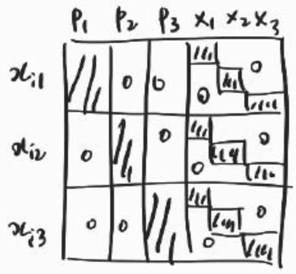
$a = (a_1^T, a_2^T, \cdots, a_m^T)^T$ where $a_j \in R^{12\times1}$ projective matrix

↓ sparsity

$\frac{\partial \hat{x}_{ij}}{\partial a_k} = 0$ unless $j = k$ $\because$ image projection $\hat{x}_{ij}$ is only dependent to j-th camera

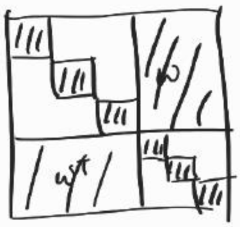
$\frac{\partial \hat{x}_{ij}}{\partial X_k} = 0$ unless $i = k$ $\because$ image projection $\hat{x}_{ij}$ is only dependent to i-th 3D point

$\therefore$ J and $H (= J^T J)$ are both sparse

$J =$ 

eg 3 cameras, 3 3D points

All non-diagonal entries of sub-matrix is zero

$H = J^T J =$ 

Hessian matrix is also the adjacency matrix of a graph

↓

Camera parameters

$$\underbrace{\begin{bmatrix} U^* - WV^{*-1}W^T & 0 \\ W^T & V^* \end{bmatrix}}_{\text{Schur's complement}} \begin{pmatrix} \delta a \\ \delta b \end{pmatrix} = \begin{pmatrix} \varepsilon_A - WV^{*-1}\varepsilon_B \\ \varepsilon_B \end{pmatrix}$$

3D point params

Since normally $|\delta a| < |\delta b|$, finding inverse of $U^* - WV^{*-1}W^T$ is computationally small

* Schur's complement : Block diagonal, sparse, Symmetric positive Matrix

$(U^* - WV^{*-1}W^T)\delta a = \varepsilon_A - WV^{*-1}\varepsilon_B \rightarrow$ Solve $x (= \delta a)$ of $Ax = b$

- sparse matrix factorization
  - (i) $A = LU$: $L(Ux) = b \rightarrow Ly = b,\ y = Ux$
  - (ii) $A = QR$: $Q(Rx) = b \rightarrow y = Q^T b,\ y = Rx$
  - (iii) $A = LL^T$ (cholesky) : $L(L^T x) = b$
- Iterative methods
  - (i) Conjugate gradient
  - (ii) Gauss - Seidel

• Problem of Fill-in

After matrix factorization, factorized matrix becomes dense

e.g. sparse A $\xrightarrow[A = LL^T]{}$ Dense L

$\downarrow$ solution : Reorder sparse matrix s.t. single non-zero element per row & column

$Ax = b \xrightarrow[\text{Permutation matrix}]{P} (P^T A P)(P^T x) = P^T b$

But this step is NP-complete!

$\downarrow$ Approximate solutions

(i) Minimum degree
(ii) Column approximate minimum degree permutation
(iii) Reverse Cuthill - Mckee
(iv) Nested Dissection