Lecture 7 - Structure from motion II

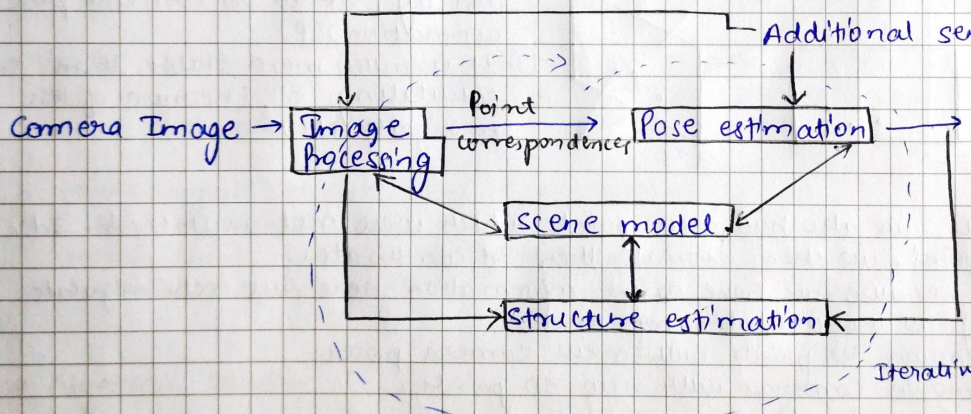
→ Offline sfm → ① No realtime requirements, all images are available at once.
② Eg: as basis for dense 3D model reconstruction.

Realtime sfm → ① Realtime requirements, images are available one by one, output required at each time step.
② Eg: for mobile augmented reality in unknown environments.

→ Comparison.

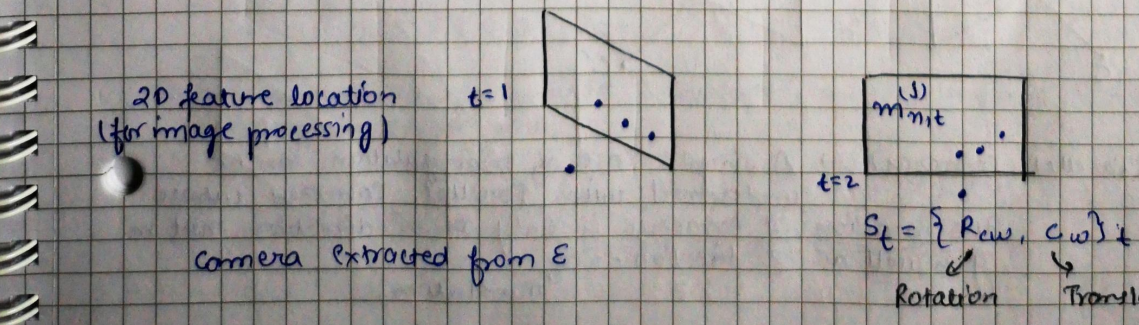
Offline sfm	Realtime sfm
① uses many features and computationally intense image processing methods	① Reduced no. of features and lightweight image processing methods.
② Forward and backward feature matching	② sequential / Recursive processing scheme.
③ Global bundle Adjustment	③ Local bundle adjustment (keyframe/window based).

→ Structure and motion - Basic Pipeline.

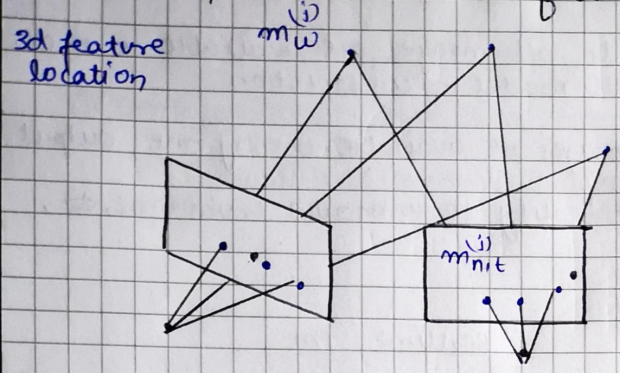


→ Realtime sfm → Alternating estimation of Camera Poses and 3D feature locations (triangulation) and from a (continuous) image sequence.
→ we assume that the camera are calibrated. (we know K, intrinsic camera parameters) and we know how to extract the camera position and orientation from "E". (which defines the geometry b/w 2 images).

① How do we initialize the first camera Poses?

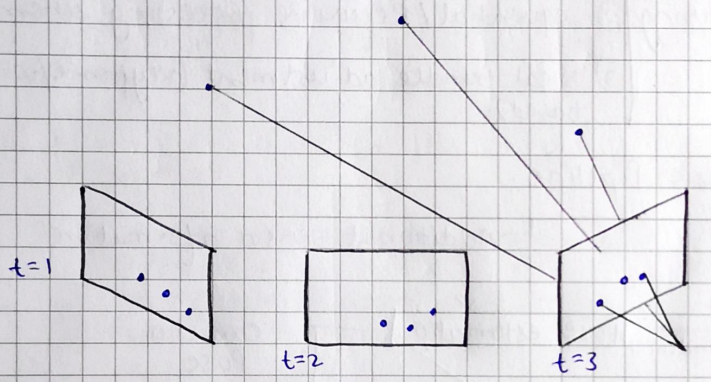


(2) Now, we have the two first camera poses, how do we go on?



Triangulate the 3D points.
(we have the structure of the scene)

(3) when we have a new frame i.e. estimate the next camera pose.

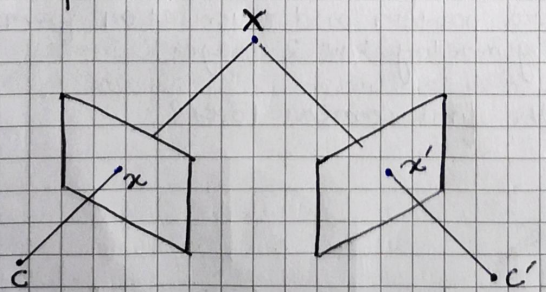


(1) The first step is to propagate the info. from the 2nd frame to the 3rd/new frame (by tracking of 2D points)
(3) Once we have these points in the new frame, we will do the camera pose estimation directly (3d to 2D camera pose estimation) (P)
(3) It is much more stable than calculating "E". (because of less ambiguity, etc)

(4) In addition to this, we also had new points which were not available in 3d. For the ~~new~~ new point, we then estimate their 3d coordinate.
(5) In the next step, because we have more information here, we retriangulate our points using all the available information.
(6) we refine the known 3D points with new camera poses.
(7) we refine the known cameras with new 3D points.

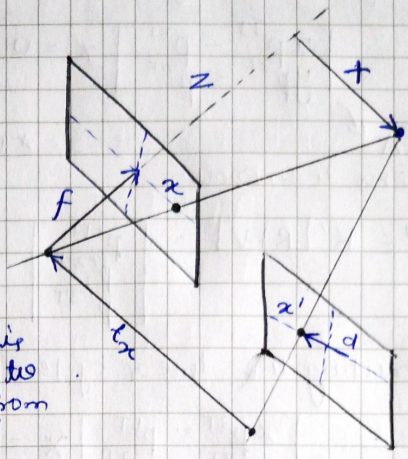
⇒ <u>Triangulation</u>

objective → Given atleast 2 known cameras (C and C') and 2 corresponding feature points (x and x') (i.e. the 2 camera views), estimate the 3D point X



⇒ Triangulation (Parallel cameras) → A simple case of triangulation can be understood with parallel cameras where the motion of cameras is only in 'x' direction not in y as well as z direction. → translation.

$K = K' = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $R = I$ $t = \begin{pmatrix} t_x \\ 0 \\ 0 \end{pmatrix}$



Then $y = y'$ (epipolar lines) and depth Z can be computed from disparity:

$$\frac{Z}{f} = \frac{t_x}{d} \quad (d = x' - x)$$

$$\boxed{Z = \frac{f\, t_x}{d}}$$

we have a depth which is inversely proportional to "d" (shift of the point from image to the other)

disparity of image point

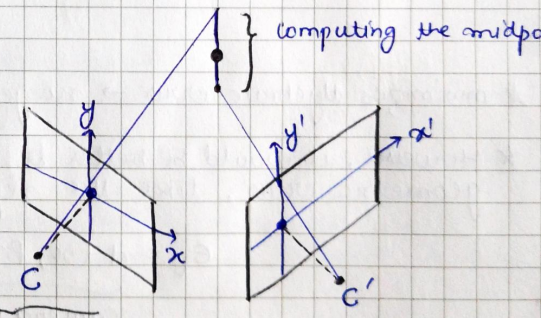
* derivation via equal triangles $\frac{x}{f} = \frac{X}{Z}$ $\frac{x'}{f} = \frac{X + t_x}{Z}$

$$\frac{x'}{f} = \frac{x}{f} + \frac{t_x}{Z}$$

* feature displacement (disparity) is inversely proportional to depth.
as $d \to 0$, $Z \to \infty$

⇒ Vector solution → Let's say instead of stereo cameras, we have two separate cameras. Due to noise, the two rays (which are projected from the image point) might not meet.

→ we compute the mid point of the shortest line between the two rays.
→ This can be further solved algebraically by least squares method.



⇒ Algebraic solution.

① Equation for 1 camera view

image point $x \;\alpha\; PX$ (up to scale) $\iff \lambda x = PX$ (scale factor) $\iff x \times \begin{bmatrix} p^{1T} \\ p^{2T} \\ p^{3T} \end{bmatrix} X = 0$

x — projected point
P — matrix (3×4 camera matrix)
X — 3d point (3DOF)
we know x, P. we have to calculate the 3d point X.
p^{iT} — i^{th} row of P

$\lambda x = PX$
$\lambda x \times x = x \times PX$
$0 = x \times PX$
doing cross prod on both the side

② multiple camera views

$$x \times PX = 0 \quad \text{and} \quad x' \times P'X = 0$$

③ one camera view can provide 3 equations, only 2 linearly independent
i.e. dropping the 3rd row

$$P = K[R|t] = \begin{bmatrix} p^{1T} \\ p^{2T} \\ p^{3T} \end{bmatrix}$$

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

④ Construction of a linear system.

$$\begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \end{bmatrix} X = 0$$

2×4 matrix A
i.e. 2 equations 4 unknowns

* when we increase the camera views i.e.

$$P' = K'[R|t] = \begin{bmatrix} p'^{1T} \\ p'^{2T} \\ p'^{3T} \end{bmatrix} \qquad \begin{aligned} x(p'^{3T}X) - (p'^{1T}X) &= 0 \\ y(p'^{3T}X) - (p'^{2T}X) &= 0 \end{aligned}$$

again constructing the linear system.

4 equations, 4 unknowns.

$$\begin{cases} \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \end{bmatrix} X = 0 & A_1 \\ \vdots \\ \begin{bmatrix} xp'^{3T} - p'^{1T} \\ yp'^{3T} - p'^{2T} \end{bmatrix} X = 0 & A_2 \\ \vdots \end{cases}$$

⑤ Stacking equations for all camera views:

$$\begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{bmatrix} X = 0$$

(i) Again, the linear system ($AX=0$) can be solved by SVD.

(ii) Recalling →

① minimize $\|AX\|_2^2$, subject to $\|X\|_2^2 = 1$

② then normalize the 3D point X by dividing the 4th entry of X.

* minimizes algebraic error → no geometrical interpretation.

* However, it would be better to find the 3D point that minimize a meaningful geometric error, like the reprojection error.

$$\epsilon_i = \underbrace{d(x, PX_i)^2}_{\text{camera 1}} + \underbrace{d(x', P'X_i)^2}_{\text{camera 2}}$$

should be 0 or as small as possible.

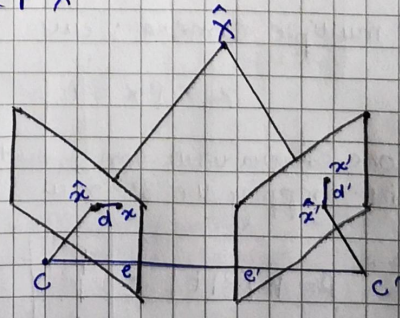
⇒ minimizing the Geometric Error

objective → Estimate the 3D point $\hat{X}$, which exactly satisfies the supplied camera geometry P and P', so that it projects as

$$\hat{x} \propto P\hat{X} \qquad \hat{x}' \propto P'\hat{X}$$

① If the measurement noise in the image points is Gaussian with mean equal to zero, minimizing the reprojection error gives the maximum likelihood estimation of X.

② where $\hat{x}$ and $\hat{x}'$ are closest to actual image measurements.



$$\min d(x, \hat{x})^2 + d(x', \hat{x}')^2 \quad \text{subject to} \quad \hat{x} \propto P\hat{X} \quad \text{and} \quad \hat{x}' \propto P'\hat{X}$$

③ This assumes perfect camera poses meaning P is error free.
④ Non-linear problem → can be solved by Levenberg-Marquard method.

⇒ Triangulation - Properties.

① The smaller the angle (small baseline, big distance), the bigger the reconstruction uncertainty.

⇒ Bundle Adjustment.

① The whole idea of BA is to optimize all the above computations that we just studied

② Valid poses estimation $[R_1|t_1]$, $[R_2|t_2]$, $[R_3|t_3]$ .... and 3D points X^1, X^2, X^3 .... must let the re-projection close to the observations, i.e. to minimize the reprojection error.

$$\underset{X^j, R_i, t_i}{\operatorname{argmin}} \sum_i \sum_j \| x_i^j - K[R_i|t_i] X^j \|_2^2 \longrightarrow \text{cost function.}$$

③ Global Bundle Adjustment (BA):
→ Refine the visual reconstruction to produce jointly optimal camera poses and 3D points.

$$\underset{\text{estimate.}}{\underset{\text{minimize over parameter vector containing all camera poses and 3D points}}{\operatorname{argmin}}} \sum_i \sum_j r_i^j \longrightarrow \text{Residual / Reprojection error.}$$

over all frames and 3D points

④ For each camera, we have 6 parameters, & for every 3D point, we have 3 parameters.
→ $6R + 3L$ parameters must be estimated. (for optimization)
→ matrices are sparse (having lot of zeros), and therefore can be highly optimized).

⑤ minimizing the cost function:

$$\underset{X^j, R_i, t_i}{\operatorname{argmin}} \sum_i \sum_j \| x_i^j - K[R_i|t_i] X^j \|_2^2 \quad \} \text{ non-linear optimization.}$$

⇒ Drift (A big problem)

① Both offline SFM and SLAM will drift (diverge) very quickly.
② measurement has errors.
③ Uncertainties in camera poses propagate to the triangulated 3D points and vice-versa.

⇒ Because of the above factors we have to perform drift correction.

⇒ Init - (Initializing with 3 images) (instead of two)

① The length of the translation vector is set to 1 ($\because \|t\| = 1$) for 2 views.

② this defines the scaling (or "arbitrary" scaling of the world):
→ for the 3rd view, the scaling must be computed and coherent with the defined scale.

③ R_{12}, R_{13}, R_{23} and t_{12}, t_{13}, t_{23} are the rotation and translation vector among the three views 1, 2 and 3.

$$P_1 = K_1 (I, 0)$$

$$P_2 = K_2 (R_{12}, t_{12})$$

$$P_3 = K_3 (R_{13}, \lambda_{13} t_{13})$$ → scaling factor b/w view 1 and 3.

$$\lambda_{13} = \frac{(t_{23} \times t_{13})^T (t_{23} \times R_{23} t_{12})}{\| t_{23} \times t_{13} \|^2}$$

⇒ Drift reduction - methods

① Offline Sfm: If we have an offline system, the reprojection error is minimized through global bundle adjustment, where the estimated 3D points and camera poses will be refined globally.

② Online Sfm (Realtime)

① Filter based → estimated 3D points and camera poses could be refined recursively in Extended Kalman filter (EKF)

② Keyframe based → for improving realtime efficiency, local bundle adjustment is used only on keyframes (selected good frames), combined with global bundle adjustment, when it is necessary.

③ Loop closure → loop closure is based on the concept of recognizing the place where the camera visited before, close the loop and eliminate drift.

③ Feature level → ① Reduce drifts in feature tracking / matching
② Extend feature tracks
③ Recapture lost features. (when a person enters a scene, some features gets lost,).

④ Geometric level → (Careful, precise, robust) 3D point initialization.
→ How?
→ Triangulate over a whole set of camera views ($>>2$)
→ Enforce a minimal angle θ between view rays.
→ Use RANSAC to eliminate outliers.
→ Use only well reconstructed points for further estimation.

Summary → Incorporate uncertainties, e.g. simple stochastic model and WLS (weighted least square) estimation: All entities modelled as Gaussian Random Variables. (since, the noise is gaussian)

⇒ Initial triangulation using RANSAC:
↳ Trial and error.

→ Algorithm
① if $\theta \geq \theta_0$: store the new camera view
② if size of history $> n$
(a) Until no more valid results or more options found
(a.1) Triangulate the pairs of camera views
(a.2) Validate the results.

③ If a valid result is found:
(a) Compute a least squares (LS) estimate from all inliers
(b) Compute a weighted least squares (WLS) estimate from all inliers.

⇒ 3D point refinement.

→ Incorporate new camera view, each time the feature is observed in an image.
→ methods:
→ Repeated triangulation
→ Recursive filtering (E.g. using extended Kalman filter)

⇒ Loop closure → Detection and recognition of already visited and mapped area after the camera exploring a long time. It is done by understanding the scene)

⇒ Scene Understanding.

① The content of an image can be inferred from the frequency of words.

→ visual words = independent features
→ Represent the images based on a histogram of word occurences (bag
→ Each detected feature is assigned to the closest entry in the codebook.