Structure from Motion | definition

- **Goal:** given many image point correspondences, compute simultaneously the 3D structure and the relative pose.
Structure from Motion | definition

- **Problem formulation:** Given many points *correspondence* between two images, \( \{(u^i_1, v^i_1), (u^i_2, v^i_2)\} \), simultaneously compute the 3D location \( P_i \), the camera relative-motion parameters \( (R, t) \), and camera intrinsic \( K_{1,2} \) that satisfy


\[
\begin{align*}
\lambda_1 \begin{bmatrix} u^i_1 \\ v^i_1 \\ 1 \end{bmatrix} &= K_1[0|0]. \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix} \\
\lambda_2 \begin{bmatrix} u^i_2 \\ v^i_2 \\ 1 \end{bmatrix} &= K_2[R|T]. \begin{bmatrix} X^i_w \\ Y^i_w \\ Z^i_w \\ 1 \end{bmatrix}
\end{align*}
\]
Structure from Motion | definition

- We study the case in which the camera is «calibrated» ($K$ is known)
- Thus, we want to find $R$, $T$, $P_i$ that satisfy

\[
\lambda_1 \begin{bmatrix} \bar{u}_1^i \\ \bar{v}_1^i \\ 1 \end{bmatrix} = [I|0] \cdot \begin{bmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{bmatrix}
\]

\[
\lambda_2 \begin{bmatrix} \bar{u}_2^i \\ \bar{v}_2^i \\ 1 \end{bmatrix} = [R|T] \cdot \begin{bmatrix} X_w^i \\ Y_w^i \\ Z_w^i \\ 1 \end{bmatrix}
\]
How many knowns and unknowns?

- **4n knowns:**
  - $n$ correspondences; each one $(u_{1}^{i}, v_{1}^{i})$ and $(u_{2}^{i}, v_{2}^{i})$, $i = 1 \ldots n$

- **5 + 3n unknowns**
  - 5 for the motion up to a scale (rotation $\mapsto 3$, translation $\mapsto 2$)
  - $3n = \text{number of coordinates of the } n \text{ 3D points}$

Does a solution exist?

- Yes, if and only if the number of independent equations $\geq$ number of unknowns
  \[\Rightarrow 4n \geq 5 + 3n \Rightarrow n \geq 5\]
Cross Product (or Vector Product) | review

\[ \vec{a} \times \vec{b} = \vec{c}, \quad \|\vec{c}\| = \|\vec{a}\|\|\vec{b}\| \sin(\theta) \cdot \hat{n} \]

- Vector cross product takes two vectors and returns a third vector that is perpendicular to both inputs
  \[ \vec{a} \cdot \vec{c} = 0 \]
  \[ \vec{b} \cdot \vec{c} = 0 \]

- The cross product of two parallel vectors = 0
- The vector cross product also can be expressed as the product of a skew-symmetric matrix and a vector

\[
\mathbf{a} \times \mathbf{b} = \begin{bmatrix}
0 & -a_z & a_y \\
 a_z & 0 & -a_x \\
-a_y & a_x & 0 \\
\end{bmatrix}
\begin{bmatrix}
b_x \\
b_y \\
b_z \\
\end{bmatrix} = [\mathbf{a}_x] \mathbf{b}
\]
Epipolar Geometry

\[ p_1, p_2, T \text{ are coplanar:} \]

\[ p_2^T \cdot n = 0 \quad \Rightarrow \quad p_2^T \cdot (T \times p_1') = 0 \quad \Rightarrow \quad p_2^T \cdot (T \times (Rp_1)) = 0 \]

\[ \Rightarrow \quad p_2^T [T]_x R p_1 = 0 \quad \Rightarrow \quad p_2^T E p_1 = 0 \quad \text{epipolar constraint} \]

\[ E = [T]_x R \quad \text{essential matrix} \]
Epipolar Geometry

- The Essential Matrix can be computed from 5 image correspondences [Kruppa, 1913]. The more points, the higher accuracy.

- The Essential Matrix can be decomposed into $R$ and $T$ by recalling that $E = [T]_x R$.
  
  Two distinct solutions for $R$ and $T$ are possible (i.e., 4-fold ambiguity)

\[
p_1 = \begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix} \quad p_2 = \begin{bmatrix} u_2 \\ v_2 \\ 1 \end{bmatrix} \quad \text{Normalized image coordinates}
\]

\[
p_2^T E p_1 = 0 \quad \text{Epipolar constraint}
\]

\[
E = [T]_x R \quad \text{Essential matrix}
\]
How to compute the Essential Matrix?

- The Essential Matrix can be computed from 5 image correspondences [Kruppa, 1913]. However, this solution is not simple. It took almost one century until an efficient solution was found! [Nister, CVPR’2004]

- The first popular solution uses 8 points and is called 8-point algorithm [Longuet Higgins, 1981]
The 8-point algorithm

\[ p_1 = (\bar{u}_1, \bar{v}_1, 1)^T, \quad p_2 = (\bar{u}_2, \bar{v}_2, 1) \quad p_2^T E p_1 = 0 \]

\[
\begin{bmatrix}
\bar{u}_2 & \bar{v}_2 & 1
\end{bmatrix}
\begin{bmatrix}
e_{11} & e_{12} & e_{13} \\
e_{21} & e_{22} & e_{23} \\
e_{31} & e_{32} & e_{33}
\end{bmatrix}
\begin{bmatrix}
\bar{u}_1 \\
\bar{v}_1 \\
1
\end{bmatrix} = 0
\]

Minimize:

\[
\sum_{i=1}^{N} (p_{i1}^T E p_{i1})^2
\]

under the constraint \( ||E||^2 = 1 \)

A linear least-square solution is given through Singular Value Decomposition by the eigenvector of \( Q \) corresponding to its smallest eigenvalue (which is the unit vector that minimizes \( |Q \cdot E|^2 \))
The 8-point algorithm

- function F = calibrated_eightpoint(p1, p2)

- p1 = p1'; % 3xN vector; each column = [u;v;1]
- p2 = p2'; % 3xN vector; each column = [u;v;1]

- Q = [p1(:,1).*p2(:,1) , ...
  p1(:,2).*p2(:,1) , ...
  p1(:,3).*p2(:,1) , ...
  p1(:,1).*p2(:,2) , ...
  p1(:,2).*p2(:,2) , ...
  p1(:,3).*p2(:,2) , ...
  p1(:,1).*p2(:,3) , ...
  p1(:,2).*p2(:,3) , ...
  p1(:,3).*p2(:,3) ] ;

- [U,S,V] = svd(Q);
- E = V(:,9);
- E = reshape(V(:,9),3,3)';