Computer Vision (CS763)

Teaching cameras to "see"

Structure from Motion

Arjun Jain

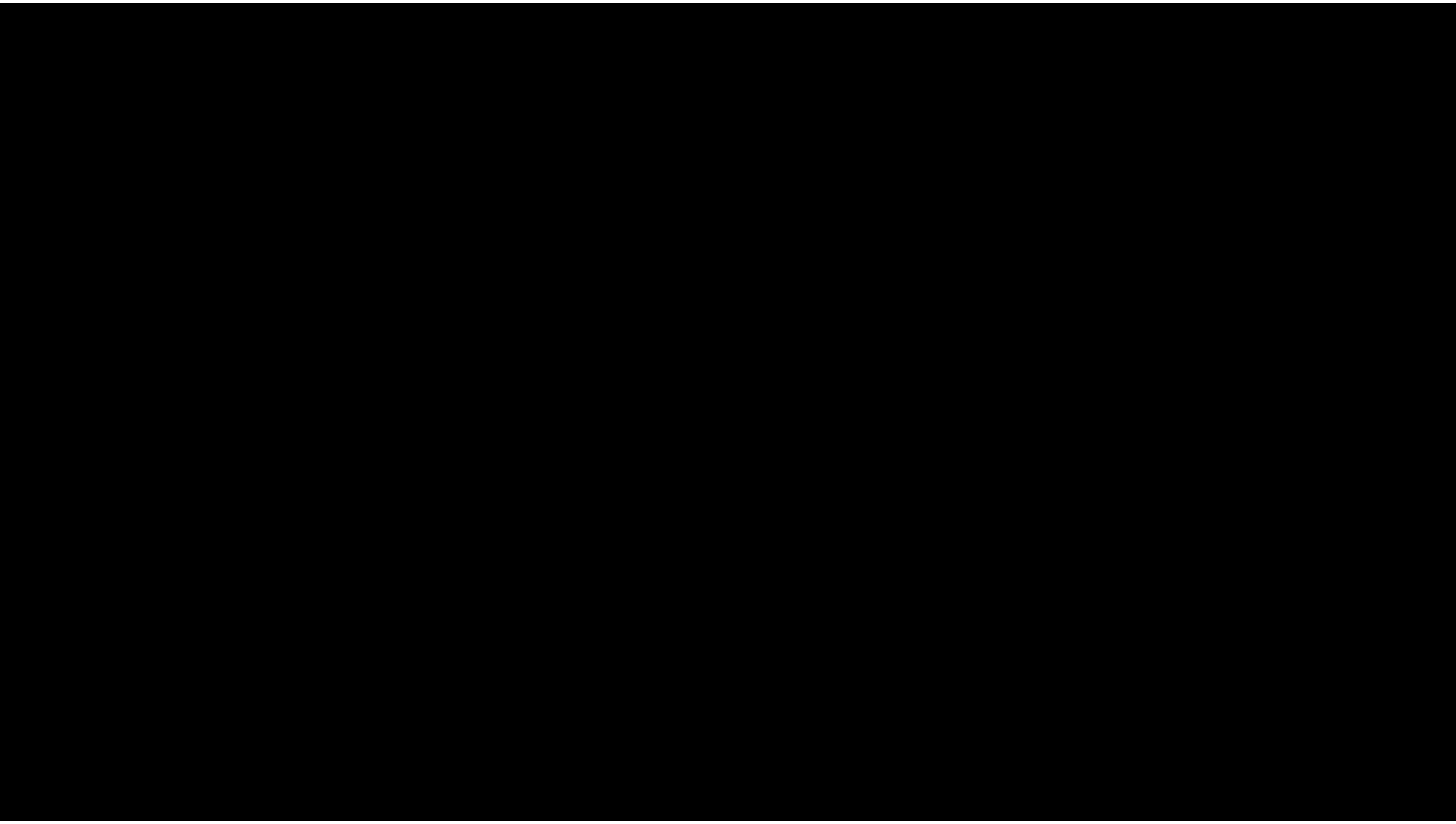
Most slides courtesy Ajit Rajwade

https://www.cse.iitb.ac.in/~ajitvr/CS763_Spring2017/ImageAlignment.pdf

Structure from

- Stereo
- Motion
- Shading
- Texture
- Contours
- Silhouettes
- …

Human Perception of SfM



https://backyardbrains.com/experiments/eye

Problem Definition

- **Structure from motion**: Infer the object’s 3D structure or shape (i.e. the X,Y,Z coordinates of several points on the object’s surface) given point correspondences when the object is in relative motion w.r.t. a camera

In this Lecture:

- We are going to study an interesting algorithm called based on factorization.
- It was developed by Tomasi and Kanade and was published in the early 90s.
- The algorithm not the most advanced but is simple and elegant.

Tomasi and Kanade, "Shape and motion from image streams under orthography: a factorization method", International Journal of Computer Vision, 1992.
http://link.springer.com/article/10.1007%2FBF00129684#page-1

Algorithm Input and Assumptions

- The camera model is orthographic.
- **Given**: A sequence of some F>=3 images of a non-planar object.
- The camera may be actually moving and the object could be still, or vice-versa, or both could be in motion.
- For simplicity but without loss of generality, we will assume the former.

Algorithm Input and Assumptions

- Let the object consist of $n \geq 3$ non-coplanar points – **P1**, **P2**, …, **Pn** measured in some world coordinate system.
- We will assume that
  1. these n 3D points are visible in each of the F frames, and
  2. the corresponding n image points are tracked and marked out in each of the F frames.

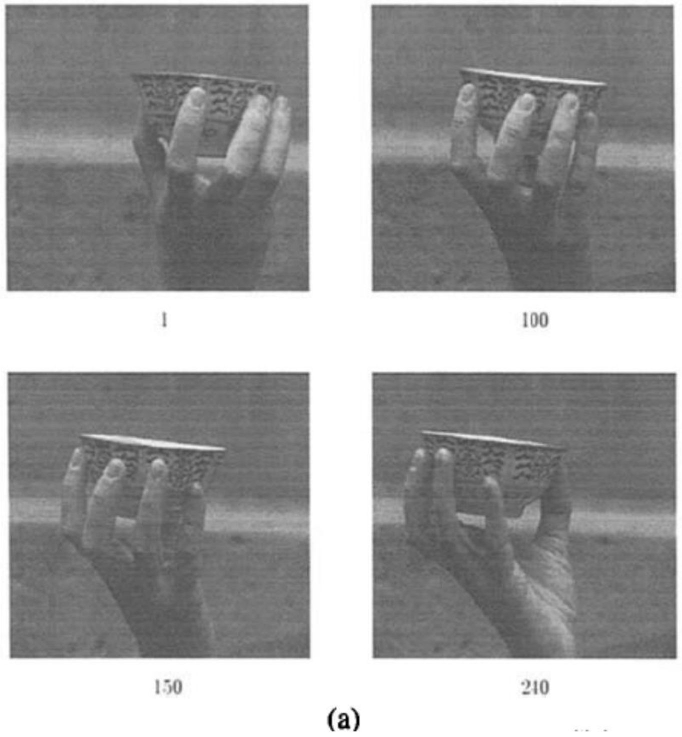
Algorithm Input and Assumptions (Repeating)

- We are assuming an orthographic camera.
- We are assuming that the whole video sequence is obtained a priori with points tracked.

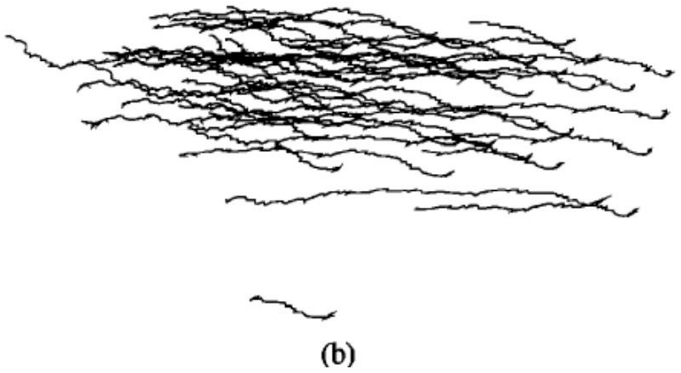
Algorithm Input and Assumptions

- We also assume there is out-of-plane rotation of the camera and not pure translation

Input



Images



KLT Tracks

Algorithm

- Let $\mathbf{p}_{\mathrm{ij}} = (x_{\mathrm{ij}}, y_{\mathrm{ij}})$ be the projection of the j-th 3D point (j = 1 to n) in the i-th frame (i = 1 to F)
- Assemble a matrix **W** (size 2F x n) as follows:

$$\mathbf{W} = \begin{pmatrix} x_{11} & x_{12} & . & . & x_{1n} \\ x_{21} & x_{22} & . & . & x_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ x_{F1} & x_{F2} & . & . & x_{Fn} \\ y_{11} & y_{12} & . & . & y_{1n} \\ y_{21} & y_{22} & . & . & y_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ y_{F1} & y_{F2} & . & . & y_{Fn} \end{pmatrix}$$

Algorithm

- Now consider the following matrix $\tilde{\mathbf{W}}$ (same size as earlier – 2F x n):

$$\tilde{\mathbf{W}} = \begin{pmatrix} \tilde{x}_{11} & \tilde{x}_{12} & . & . & \tilde{x}_{1n} \\ \tilde{x}_{21} & \tilde{x}_{22} & . & . & \tilde{x}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{x}_{F1} & \tilde{x}_{F2} & . & . & \tilde{x}_{Fn} \\ \tilde{y}_{11} & \tilde{y}_{12} & . & . & \tilde{y}_{1n} \\ \tilde{y}_{21} & \tilde{y}_{22} & . & . & \tilde{y}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{y}_{F1} & \tilde{y}_{F2} & . & . & \tilde{y}_{Fn} \end{pmatrix}$$

$$\tilde{x}_{ij} = x_{ij} - \bar{x}_i, \bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij}$$

$$\tilde{y}_{ij} = y_{ij} - \bar{y}_i, \bar{y}_i = \frac{1}{n}\sum_{j=1}^{n} y_{ij}$$

i.e., for each frame, compute the centroid of the 2D points. Deduct the centroid from the points in every frame to create the new matrix on the left. Why do we do this? We will see soon.

Algorithm

- Now consider the following matrix $\tilde{\mathbf{W}}$ (same size as earlier – 2F x n):

$$\tilde{\mathbf{W}} = \begin{pmatrix} \tilde{x}_{11} & \tilde{x}_{12} & . & . & \tilde{x}_{1n} \\ \tilde{x}_{21} & \tilde{x}_{22} & . & . & \tilde{x}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{x}_{F1} & \tilde{x}_{F2} & . & . & \tilde{x}_{Fn} \\ \tilde{y}_{11} & \tilde{y}_{12} & . & . & \tilde{y}_{1n} \\ \tilde{y}_{21} & \tilde{y}_{22} & . & . & \tilde{y}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{y}_{F1} & \tilde{y}_{F2} & . & . & \tilde{y}_{Fn} \end{pmatrix}$$

$$\tilde{x}_{ij} = x_{ij} - \bar{x}_i, \bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \longleftarrow (1)$$

$$\tilde{y}_{ij} = y_{ij} - \bar{y}_i, \bar{y}_i = \frac{1}{n}\sum_{j=1}^{n} y_{ij} \longleftarrow (2)$$

Note that this matrix actually has rank at the most 3 under ideal conditions (no noise in point coordinates).

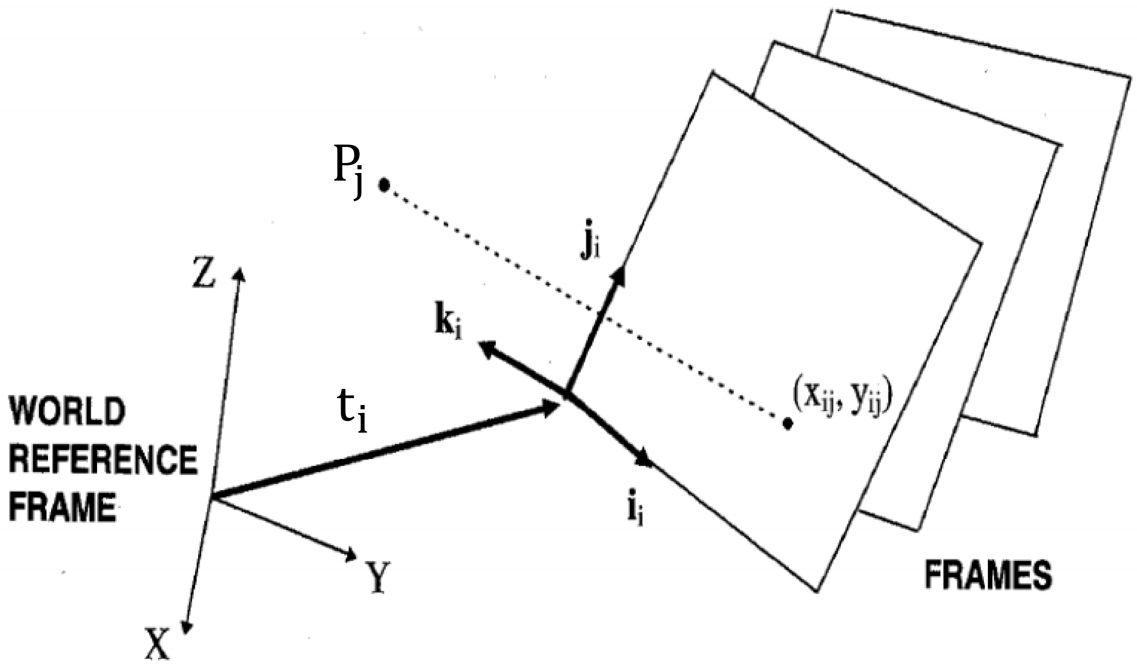
Orthographic Projection

- We assumed the 3D object is stationary and the camera to moving (performing rotation and translation).

- Each time the camera moves, its extrinsic parameters change, i.e. the rotation transformation between the camera axes and the world coordinate axes changes, and also the translation vector between the origin of the camera coordinate system and the origin of the world coordinate system changes.

Orthographic Projection

- In the i-th frame, let the translation vector be given as t_i. Let the axes of the camera as measured in the world coordinate system be given as i_i, j_i and $k_i = i_i \times j_i$



$$x_{ij} =$$
$$y_{ij} =$$

$\uparrow$

(3)

Algorithm

- Without loss of generality, we will assume that the origin of the world coordinate system is at the centroid of the 3D object.

- That is:

$$\boxed{\frac{1}{n}\sum_{j=1}^{n} P_j = 0} \longleftarrow (4)$$

Algorithm

- Now consider equations $(1), (2), (3)$ and (4)

$$\tilde{x}_{ij} = x_{ij} - \bar{x}_i, \bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_{ij} \qquad \tilde{y}_{ij} = y_{ij} - \bar{y}_i, \bar{y}_i = \frac{1}{n}\sum_{j=1}^{n} y_{ij} \qquad \frac{1}{n}\sum_{j=1}^{n} P_j = 0$$

$$x_{ij} = {i_i}^T\,(P_j - t_i)$$
$$y_{ij} = {j_i}^T\left(P_j - t_i\right)$$

- And combine them:

$$\tilde{x}_{ij} = {i_i}^T\,(P_j - t_i) - \frac{1}{n}{i_i}^T\sum_{m=1}^{n}(P_m - t_i)$$
$$\tilde{y}_{ij} = {j_i}^T\,(P_j - t_i) - \frac{1}{n}{j_i}^T\sum_{m=1}^{n}(P_m - t_i)$$

$$\Rightarrow$$

$$\tilde{x}_{ij} = {i_i}^T P_j$$
$$\tilde{y}_{ij} = {j_i}^T P_j$$

Algorithm, Rank Theorem

- Reminder:

$$\tilde{\mathbf{W}} = \begin{pmatrix} \tilde{x}_{11} & \tilde{x}_{12} & . & . & \tilde{x}_{1n} \\ \tilde{x}_{21} & \tilde{x}_{22} & . & . & \tilde{x}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{x}_{F1} & \tilde{x}_{F2} & . & . & \tilde{x}_{Fn} \\ \tilde{y}_{11} & \tilde{y}_{12} & . & . & \tilde{y}_{1n} \\ \tilde{y}_{21} & \tilde{y}_{22} & . & . & \tilde{y}_{2n} \\ . & . & . & . & . \\ . & . & . & . & . \\ \tilde{y}_{F1} & \tilde{y}_{F2} & . & . & \tilde{y}_{Fn} \end{pmatrix}$$

$$\tilde{x}_{ij} = {i_i}^T P_j$$

$$\tilde{y}_{ij} = {j_i}^T P_j$$

$\tilde{\mathrm{W}}$ can be simply re-written as

$$\tilde{\mathrm{W}} = \begin{bmatrix} {i_1}^T \\ {i_2}^T \\ \dots \\ {i_F}^T \\ {j_1}^T \\ {j_2}^T \\ \dots \\ {j_F}^T \end{bmatrix} [P_1 \quad P_2 \quad \dots \quad P_n]$$

R S

R has size 2F x 3 and has rank 3 as F ≥ 3 and camera motion has rotation
S has size 3 x n and will have rank 3 if the points in S are non-coplanar.

So in the absence of noise $\widetilde{W}$ has rank 3 exactly.

Algorithm: Find R and S

- In real life, there is noise in measurements. So:

- Given the matrix $\tilde{\mathbf{W}}$, we compute its SVD as follows:

Reduced form of the SVD as $\tilde{\mathbf{W}}$ has rank only 3

$$\tilde{\mathbf{W}}_{2F\times n} = \mathbf{U}_{2F\times 2F}\mathbf{D}_{2F\times n}(\mathbf{V}_{n\times n})^T = \underbrace{\mathbf{U}_{2F\times 3}\mathbf{D}^{1/2}_{3\times 3}}_{\mathbf{R}}\underbrace{\mathbf{D}^{1/2}_{3\times 3}(\mathbf{V}_{n\times 3})^T}_{\mathbf{S}}$$

- For i = 1 to F, the i^{th} and $(F+i)^{th}$ rows of **R** give you the vectors $\mathbf{i}_i$ and $\mathbf{j}_i$ respectively. Since $\mathbf{k}_i = \mathbf{i}_i \times \mathbf{j}_i$, we have axes of the camera coordinate system in the i-th frame. Comparing the camera coordinate systems across consecutive frames tells you how much the camera rotated from one frame to another.

Motion

- The columns of **S** give the 3D point coordinates.

Structure

Algorithm: Translation?

- How to find translation? Homework

But Wait, Problem!

- The obtained R and S may not be unique because for any invertible 3 x 3 matrix Q, we have:
  $\widetilde{\boldsymbol{W}} = \mathbf{RS} = \mathbf{R}(\mathbf{QQ}^{-1})\mathbf{S} = (\mathbf{RQ})(\mathbf{Q}^{-1}\mathbf{S})$

- How do we fix this?

Problem Solution:

- Problem: R and S not unique because:
  $\widetilde{\boldsymbol{W}} = \mathbf{RS} = \mathbf{R}(\mathbf{QQ}^{-1})\mathbf{S} = (\mathbf{RQ})(\mathbf{Q}^{-1}\mathbf{S})$

- Observe that the rows of a rotation matrix (here $\mathbf{RQ}$) must have unit magnitude. Also, Any corresponding rows must be perpendicular to each other. So we solve for Q by observing that:

$$\boldsymbol{i_i}^T \mathbf{QQ}^{\mathrm{T}} \boldsymbol{i_i} = \mathbf{1}$$
$$\boldsymbol{j_i}^T \mathbf{QQ}^{\mathrm{T}} \boldsymbol{j_i} = \mathbf{1}$$
$$\boldsymbol{i_i}^T \mathbf{QQ}^{\mathrm{T}} \boldsymbol{j_i} = \mathbf{0}$$

These 3 equations are true for all i from 1 to F (i.e. for each frame). These equations are called the **metric properties** or metric constraints on R. Recall that $\boldsymbol{i_i}$ and $\boldsymbol{j_i}$ are obtained from the R matrix that you get from the SVD of $\widetilde{\boldsymbol{W}}$.

Problem Solution:

- Now we can solve for $\mathbf{Q}$ which satisfy the equations below using Newton's method (details later):

$$\begin{aligned} \boldsymbol{i_i}^T\mathbf{Q}\mathbf{Q}^\mathrm{T}\boldsymbol{i_i} - \mathbf{1} &= \mathbf{0} \\ \boldsymbol{j_i}^T\mathbf{Q}\mathbf{Q}^\mathrm{T}\boldsymbol{j_i} - \mathbf{1} &= \mathbf{0} \\ \boldsymbol{i_i}^T\mathbf{Q}\mathbf{Q}^\mathrm{T}\boldsymbol{j_i} &= \mathbf{0} \end{aligned}$$

These 3 equations are true for all i from 1 to F (i.e. for each frame). Recall that $\boldsymbol{i_i}$ and $\boldsymbol{j_i}$ are obtained from the R matrix that you get from the SVD of $\widetilde{\boldsymbol{W}}$.

- The R and S matrices will be as follows:

$$\mathbf{R} \leftarrow \mathbf{RQ}$$
$$\mathbf{S} \leftarrow \mathbf{Q}^{-1}\mathbf{S}$$

- But these solutions are also unique only up to some unknown orthonormal transformation $\mathbf{R_0}$ i.e.

$$\widetilde{\mathbf{W}} = \mathbf{RS} = \mathbf{R}\mathbf{R_0}\mathbf{R_0}^\mathrm{T}\mathbf{S}$$

Problem Solution:

- Note that this $\mathbf{R_0}$ cannot be uniquely obtained by exploiting the metric properties unlike the case of $\mathbf{Q}$ (why?)

- All this means is that the if you assumed all the camera positions were rotated by some fixed $\boldsymbol{R_0}$ in every frame, the object coordinates would rotate by a fixed $\boldsymbol{R_0}^{-1}$in every frame.

- Note that this ambiguity stems from the fact that though we fixed the origin, we never fixed the orientation of the world coordinate system!

Problem Solution:

- This can be resolved by assuming that in the first frame, the world coordinate system is aligned with the camera coordinate system. So, to do this alignment, we simply pick $\boldsymbol{i_1}, \boldsymbol{j_1}$ and infer $\boldsymbol{k_1}$ $(\boldsymbol{i_1} \times \boldsymbol{j_1})$ to construct a matrix $\mathbf{A}$ and say $\mathbf{AR_0} = \mathrm{I}$.

$$\widetilde{\mathrm{W}} = \begin{bmatrix} {i_1}^T \\ {i_2}^T \\ \dots \\ {i_F}^T \\ {j_1}^T \\ {j_2}^T \\ \dots \\ {j_F}^T \end{bmatrix} [P_1 \quad P_2 \quad \dots \quad P_n]$$

- Find $\boldsymbol{R_0} = \boldsymbol{A}^T \boldsymbol{I}$

- Once we get $\mathbf{R_0}$, we can multiply it with all of $\mathbf{R}$ and ${\mathbf{R_0}}^{-1}$ with $\boldsymbol{S}$.

How to Estimate Q?

- Reconsider the following equations (totally 3F in number):

$$\begin{aligned}\boldsymbol{i_i}^{\boldsymbol{T}}\mathbf{QQ}^{\mathrm{T}}\boldsymbol{i_i} - \mathbf{1} &= \mathbf{0}\\ \boldsymbol{j_i}^{\boldsymbol{T}}\mathbf{QQ}^{\mathrm{T}}\boldsymbol{j_i} - \mathbf{1} &= \mathbf{0}\\ \boldsymbol{i_i}^{\boldsymbol{T}}\mathbf{QQ}^{\mathrm{T}}\boldsymbol{j_i} &= \mathbf{0}\end{aligned}$$

- This is a system of non-linear equations, the variables being the 9 entries of $\mathbf{Q}$ which we rearrange to yield vector $\mathbf{q}$. We will label each equation as $f_k(\mathbf{q}) = 0$ (k = 1 to 3F).

- We construct a $3F \times 1$ equation vector $f(\mathbf{q}) = \mathbf{0}$ by clubbing all the $f_k(\boldsymbol{q})$.

How to Estimate Q?

- No closed form solution unlike linear case ☹, so, we will employ Newton-Raphson method of root-finding.

- Start with an initial guess for $\mathbf{q}$ (one initialization can $\mathbf{q_c}$= vectorized identity matrix)

- If $\mathbf{q_c}$ is indeed the true solution, $f(\mathbf{q_c}) = \mathbf{0}$ and we stop (however, this won't happen in the first when c=0, right?)

- Instead we want to find a vector $\mathbf{\Delta}$ (9x1) such that $f(\mathbf{q_c} + \mathbf{\Delta}) = \mathbf{0}$ for all k (1 to 3F)

Taylor Series: Jacobian

Suppose $f : \mathbf{R}^n \to \mathbf{R}^m$ is differentiable. Its *derivative* or *Jacobian* at a point $x \in \mathbf{R}^n$ is denoted $Df(x) \in \mathbf{R}^{m\times n}$, defined as

$$(Df(x))_{ij} = \left.\frac{\partial f_i}{\partial x_j}\right|_x, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n.$$

The first order Taylor expansion of f at (or near) x is given by

$$\hat{f}(y) = f(x) + Df(x)(y - x).$$

When $y - x$ is small, $f(y) - \hat{f}(y)$ is very small. This is called the *linearization* of f at (or near) x.

How to Estimate Q?

- We can find the first order linear approximation as:

$$f(\mathbf{q_c}+\mathbf{\Delta}) = f(\mathbf{q_c}) + \left(\frac{\partial f}{\partial \mathbf{q}}\right)_{\mathbf{q}=\mathbf{q_c}} \mathbf{\Delta}$$

Note that this is 3Fx9 Note that $\mathbf{\Delta}$ is 9x1

- But we want $f(\mathbf{q_c}+\mathbf{\Delta}) = \mathbf{0}$. Hence, *we* have:

$$\left(\frac{\partial f}{\partial \mathbf{q}}\right)_{\mathbf{q}=\mathbf{q_c}} \mathbf{\Delta} = -f(\mathbf{q_c})$$

How to Estimate Q?

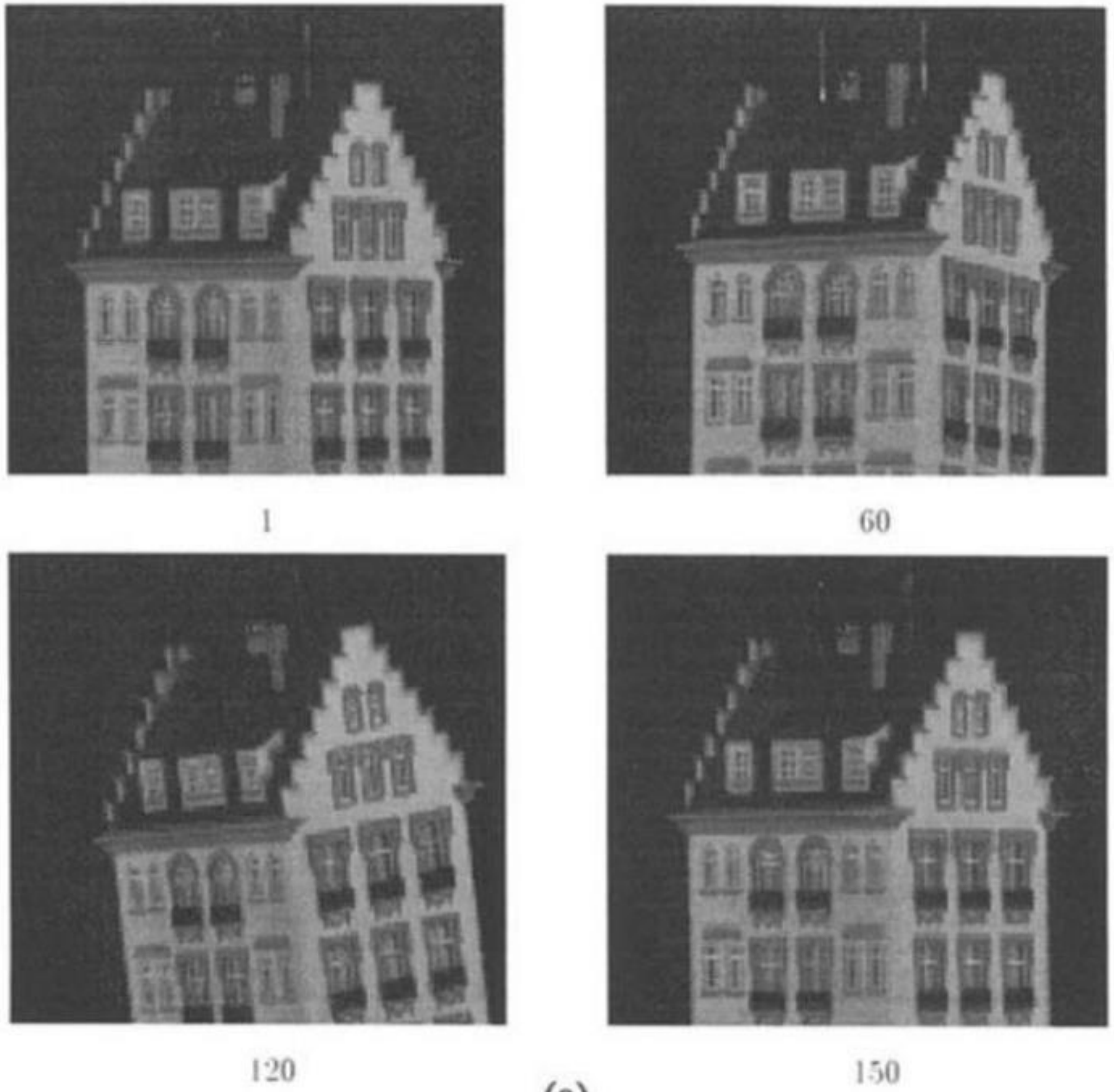
- One can solve for $\mathbf{\Delta}$ by pseudo-inverse.

- But this solution will not exactly satisfy all the equations as we performed a linear approximation which was not fully accurate, and also because a least squares solution for $\mathbf{\Delta}$ is not guaranteed to yield $f(\mathbf{q_c} + \mathbf{\Delta}) = \mathbf{0}$.

How to Estimate Q?

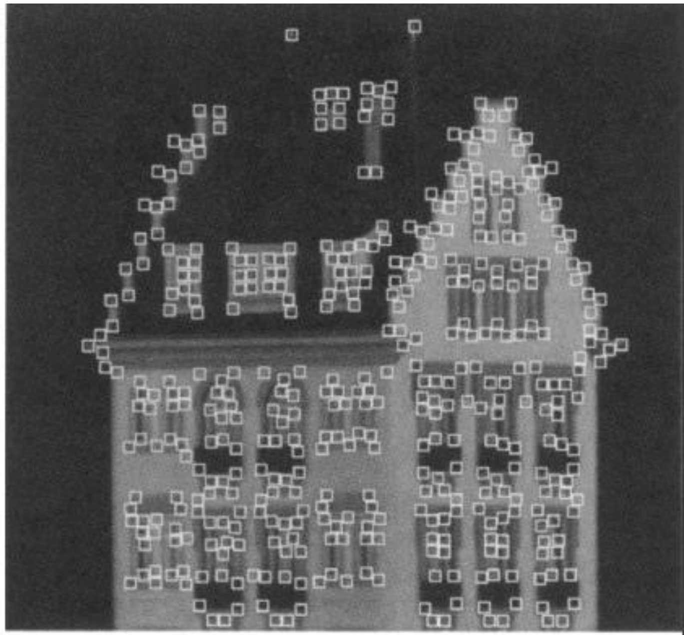
- Hence, we update our solution from $\mathbf{q_c}$ to
$$\mathbf{q_{c+1}} = \mathbf{q_c} + \mathbf{\Delta}$$
  - We repeat the previous steps with c=0,1,2,… and so on until we reach a time when $f(\mathbf{q_c} + \mathbf{\Delta}) \approx \mathbf{0}$.

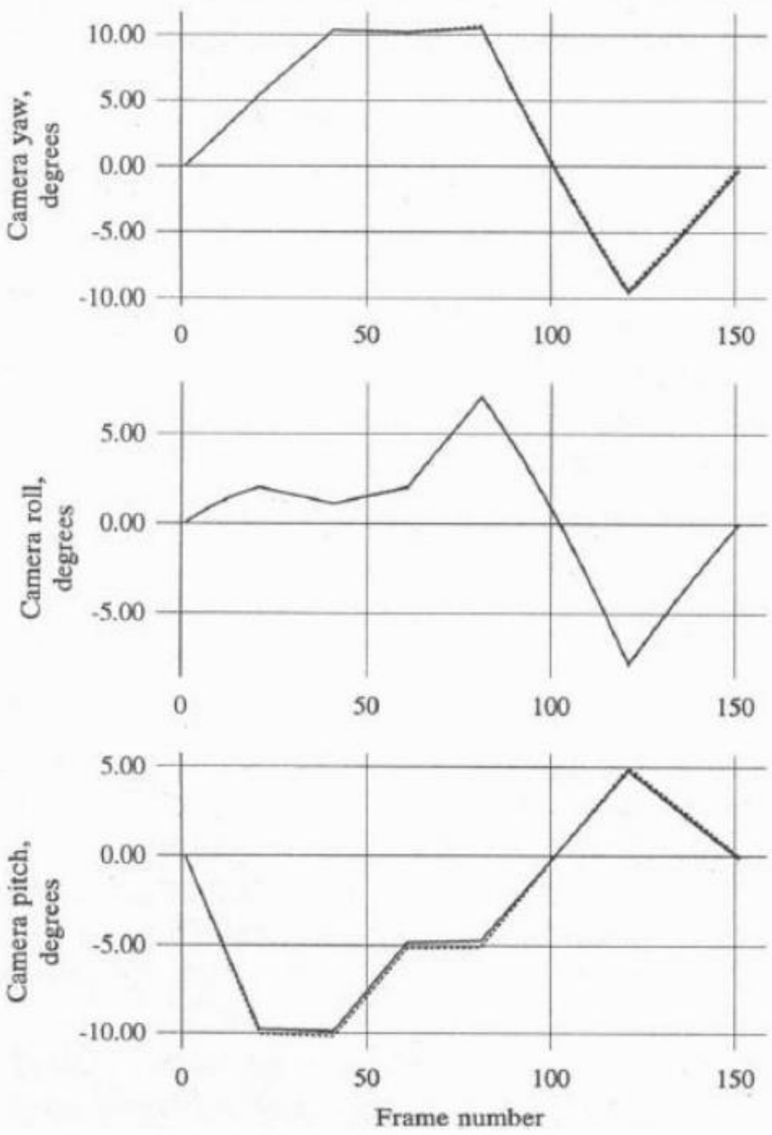
- This overall method is called Newton-Raphson method of root-finding.

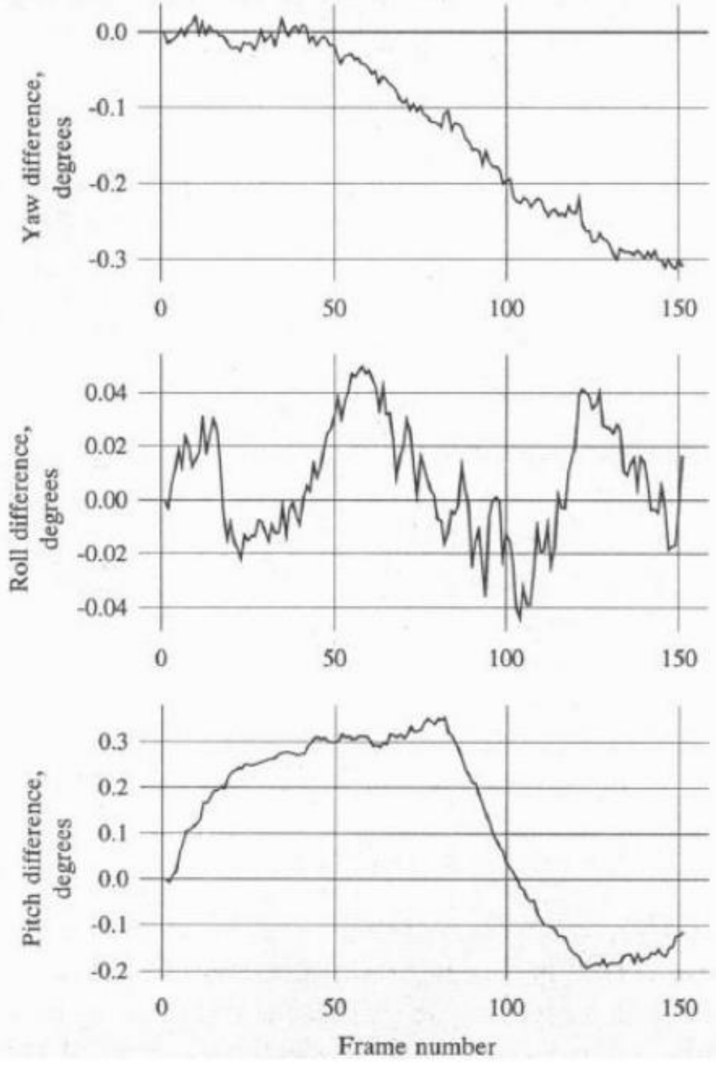


(a)

Fig. 2a. The "Hotel" stream: four of the 150 frames.




Camera yaw, degrees
10.00
5.00
0.00
-5.00
-10.00
0
50
100
150
Camera roll, degrees
5.00
0.00
-5.00
0
50
100
150
Camera pitch, degrees
5.00
0.00
-5.00
-10.00
0
50
100
150
Frame number


Yaw difference, degrees
0.0
-0.1
-0.2
-0.3
0
50
100
150
Roll difference, degrees
0.04
0.02
0.00
-0.02
-0.04
0
50
100
150
Pitch difference, degrees
0.3
0.2
0.1
0.0
-0.1
-0.2
0
50
100
150
Frame number

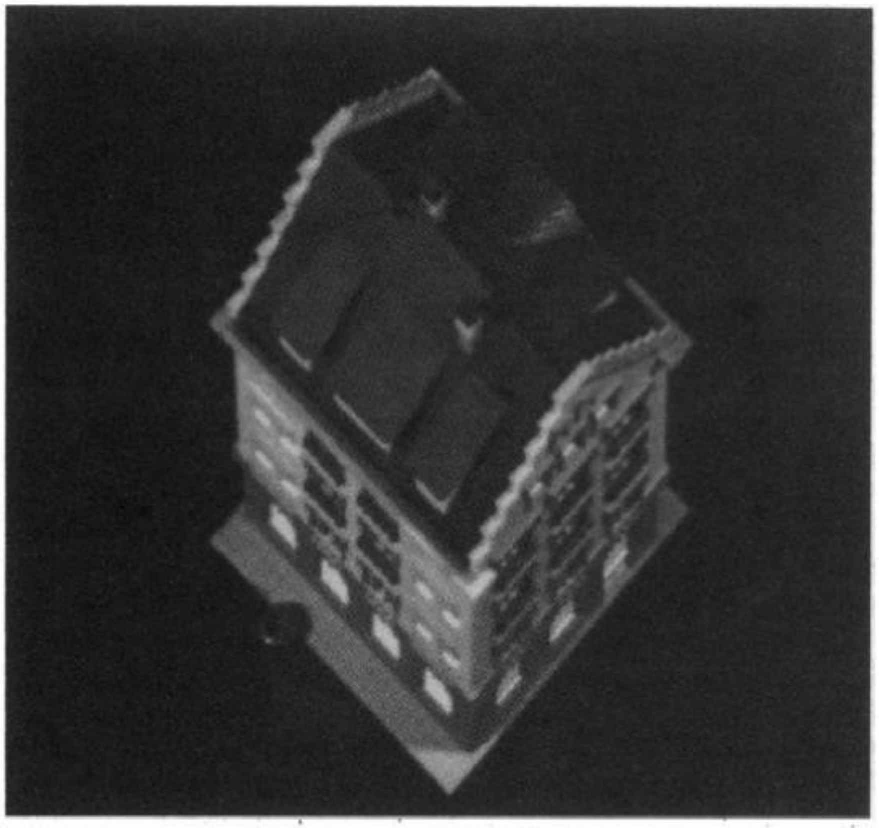
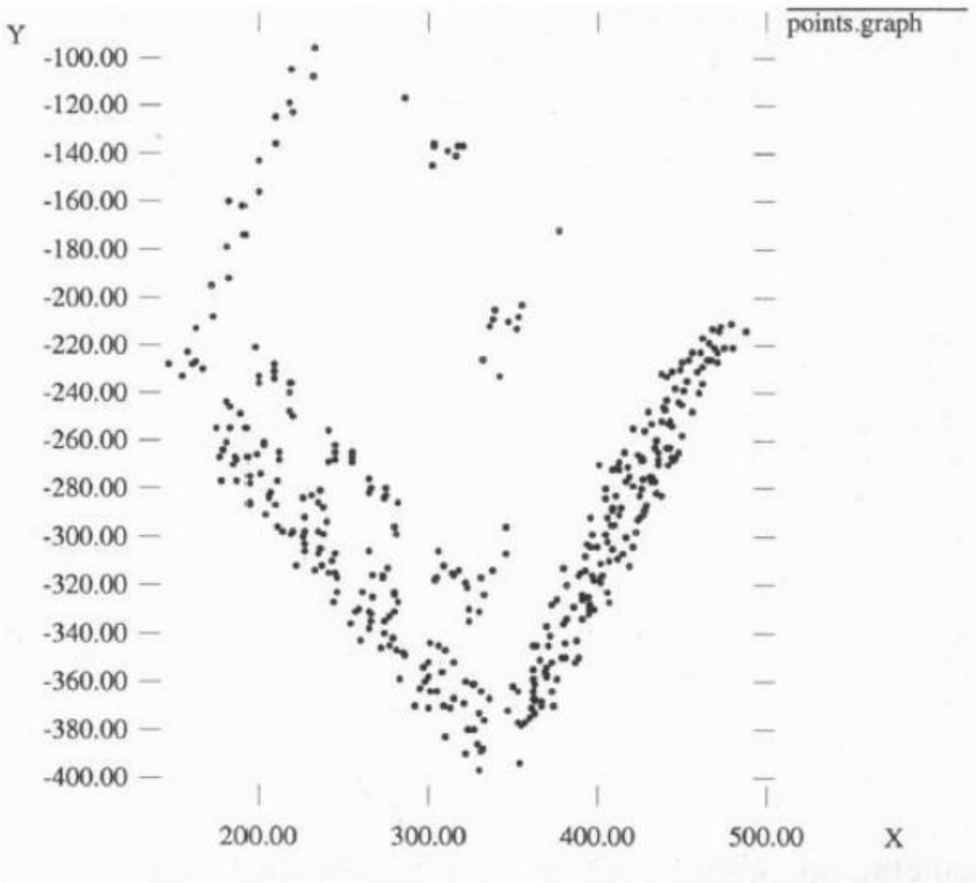


FIG. 4. (*Upper*) View of the computed shape from approximately above the building. (*Lower*) Real picture from above the building.



Slide Information

- The slide template has been created by Cyrill Stachniss (cyrill.stachniss@igg.uni-bonn.de) as part of the photogrammetry and robotics courses.
- A lot of material from Ajit Rajwade's CS763 course
- **I tried to acknowledge all people from whom I used images or videos. In case I made a mistake or missed someone, please let me know.**

Arjun Jain, ajain@cse.iitb.ac.in