

Folien zur Vorlesung am 17.06.2025
3D Computer Vision



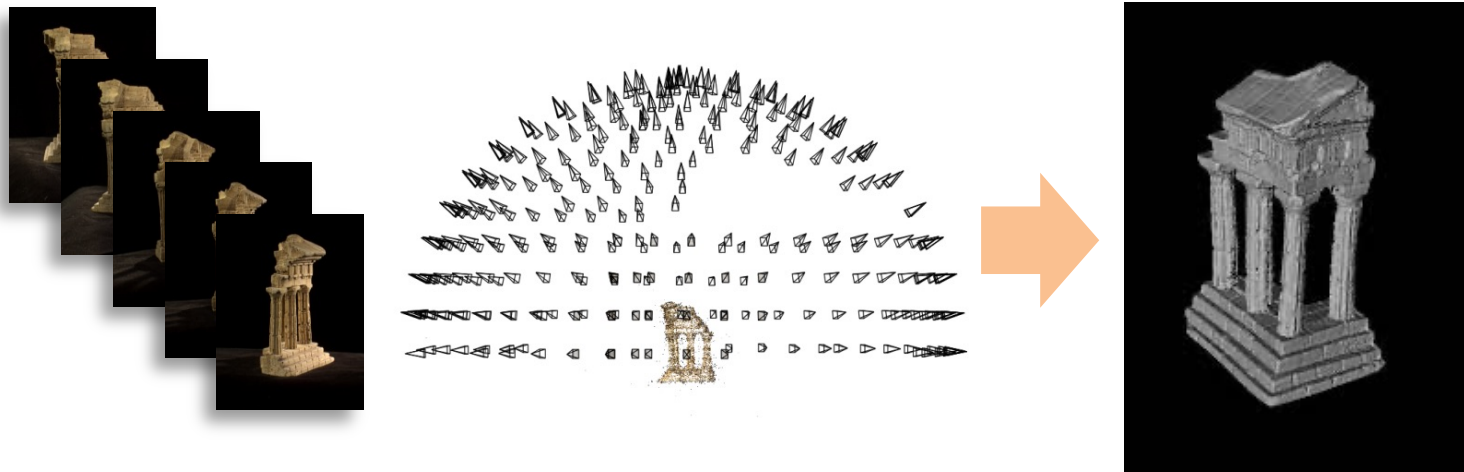
STRUCTURE FROM MOTION

References

- Book:
 - Computer Vision: Algorithms and Applications, Richard Szeliski → In the 2nd Edition Chapter 11, available at <http://szeliski.org/book>
- Slides:
 - [Introduction to Computer Vision, CS5670, Spring 2022, Cornell Tech](#) → [Noah Snavely](#) and others: [Yung-Yu Chuang](#), [Fredo Durand](#), [Alexei Efros](#), [William Freeman](#), [James Hays](#), [Svetlana Lazebnik](#), [Andrej Karpathy](#), [Fei-Fei Li](#), [Srinivasa Narasimhan](#), [Silvio Savarese](#), [Steve Seitz](#), [Richard Szeliski](#), and [Li Zhang](#).
 - Most slides are copied from there.
- Videos:
 - Shree Nayar: <https://fpcv.cs.columbia.edu/> → Today: a summary of the Reconstruction II → Structure from Motion series

Reminder: multi-view stereo

Problem formulation: given several images of the same object or scene, compute a representation of its 3D shape



Structure from motion

- Multi-view stereo assumes that cameras are calibrated
 - Extrinsics and intrinsics are known for all views
- How do we compute calibration if we don't know it? In general, this is called *structure from motion*

Large-scale structure from motion



Dubrovnik, Croatia. 4,619 images (out of an initial 57,845).
Total reconstruction time: 23 hours
Number of cores: 352

[From „Building Rome in a day“](#)

Two views



- Solve for Fundamental matrix / Essential matrix
- Factorize into intrinsics, rotation, and translation

What about more than two views?

- The geometry of three views is described by a $3 \times 3 \times 3$ tensor called the *trifocal tensor*
- The geometry of four views is described by a $3 \times 3 \times 3 \times 3$ tensor called the *quadrifocal tensor*
- After this it starts to get complicated...
 - Instead, we explicitly solve for camera poses and scene geometry

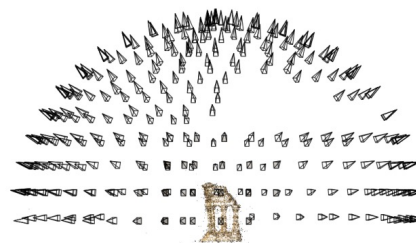
Structure from motion

- Given many images, how can we
 - a) figure out where they were all taken from?
 - b) build a 3D model of the scene?

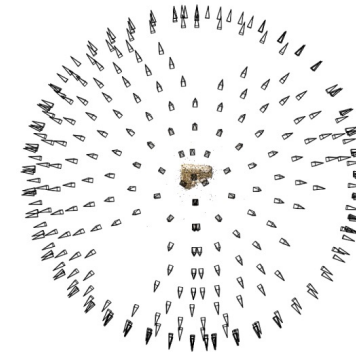


This is the **structure from motion (SfM)** problem

Structure from motion



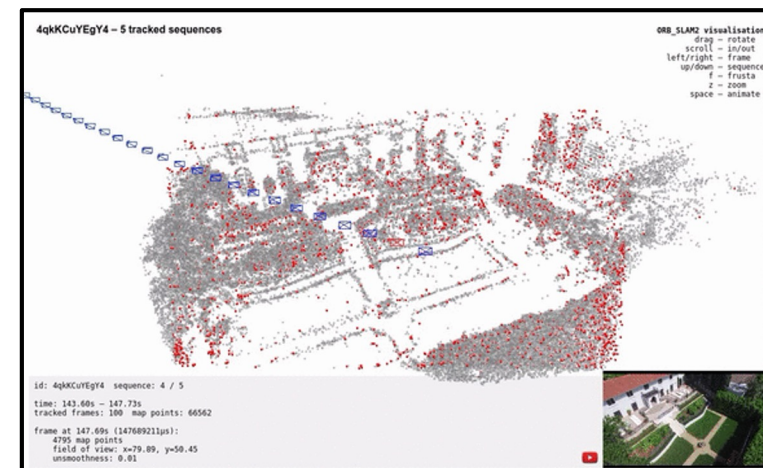
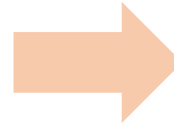
Reconstruction (side)



(top)

- Input: images with 2D points in correspondence
 $p_{ij} = (u_{ij}, v_{ij})$
- Output
 - structure: 3D location \mathbf{x}_i for each point p_i
 - motion: camera parameters $\mathbf{R}_j, \mathbf{t}_j$ & possibly \mathbf{K}_j
- Objective function: minimize *reprojection error*

Also doable from video

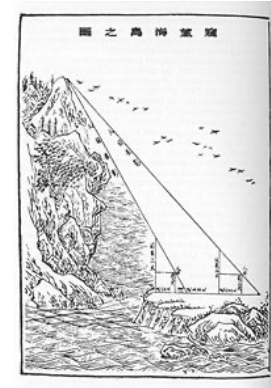


What we've seen so far...

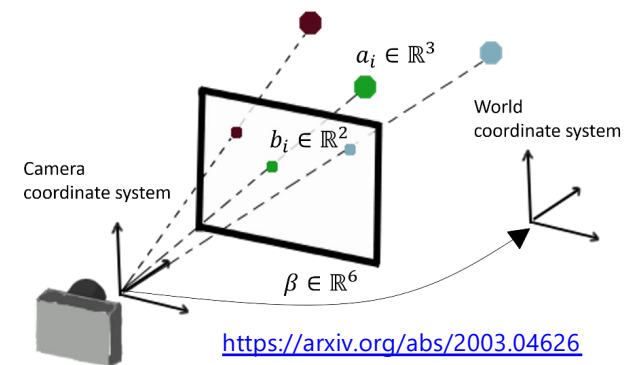
- 2D transformations between images
 - Translations, affine transformations, homographies...
- Fundamental matrices
 - Represent relationships between 2D images in the form of corresponding 2D lines
- **What's new:** Explicitly representing 3D geometry of cameras *and points*

Triangulation & camera calibration

- Suppose we have known camera parameters, each of which observes a point
 - How can we compute the 3D location of that point?
 - This is called *triangulation* (known since ancient times)
- On the other hand: Suppose we have known 3D points
 - And have matches between these points and an image
 - How can we compute the camera parameters?
 - This is called *camera calibration* (or *camera resectioning*)



[Liu Hui](#) (c. 263), How to measure the height of a sea island.



<https://arxiv.org/abs/2003.04626>

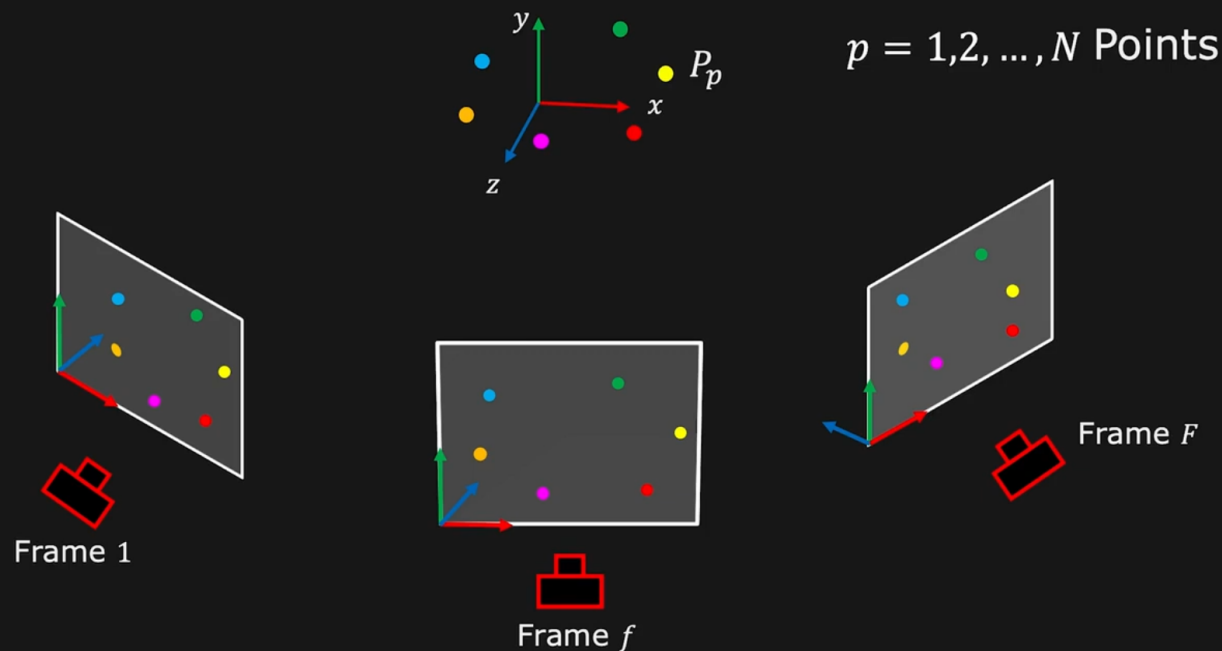
Structure from motion

- What if we don't know 3D points or camera parameters?
- SfM solves both of these problems *at once*
- A kind of chicken-and-egg problem
 - (but solvable)

Next: Classic approach using an assumption that simplifies the problem.

Afterwards: Extended approaches...

Orthographic Structure from Motion



Given sets of corresponding image points (2D): $(u_{f,p}, v_{f,p})$
Find scene points (3D) P_p , assuming **orthographic camera**.

© 2020 Shree K. Nayar

[Tomasi 1992]

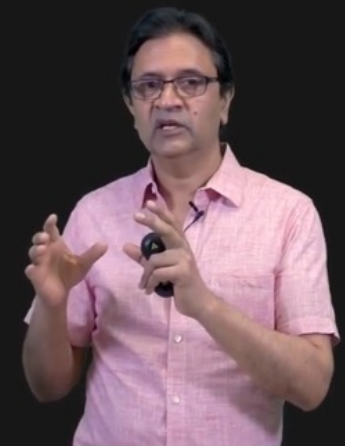
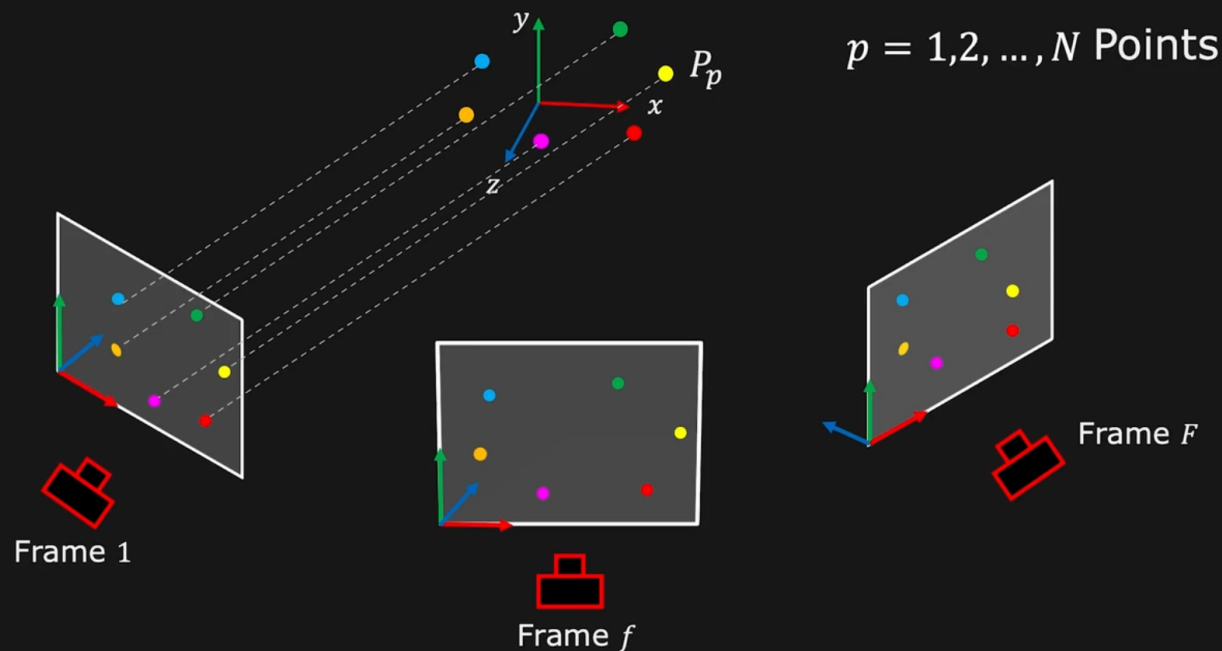


Image from [Shree Nayar FPCV videos](#)

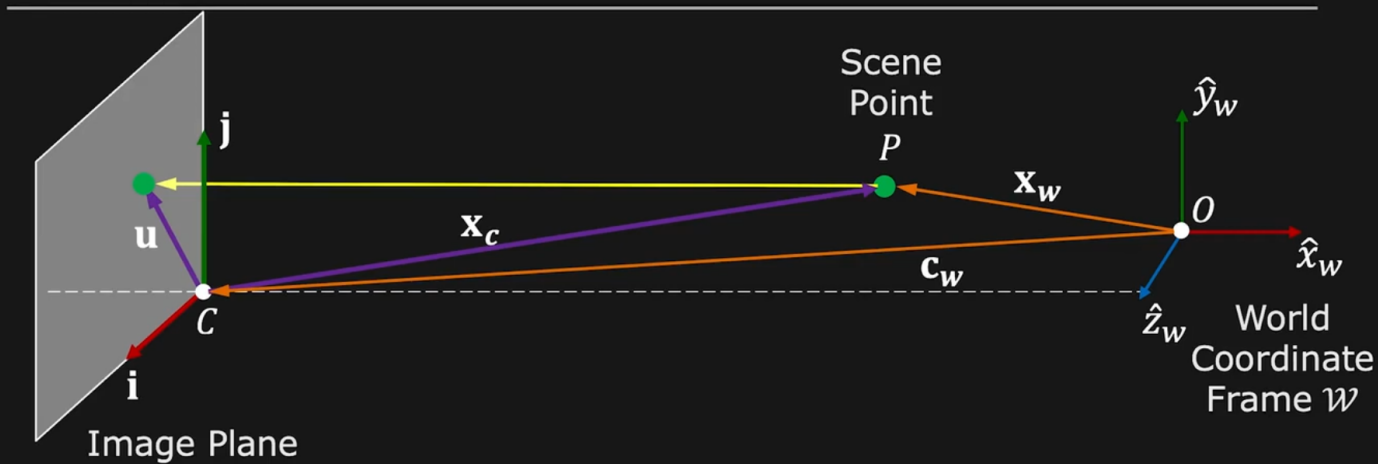
Orthographic Structure from Motion



Given sets of corresponding image points (2D): $(u_{f,p}, v_{f,p})$

Find scene points (3D) P_p , assuming **orthographic camera**.

From 3D to 2D: Orthographic Projection

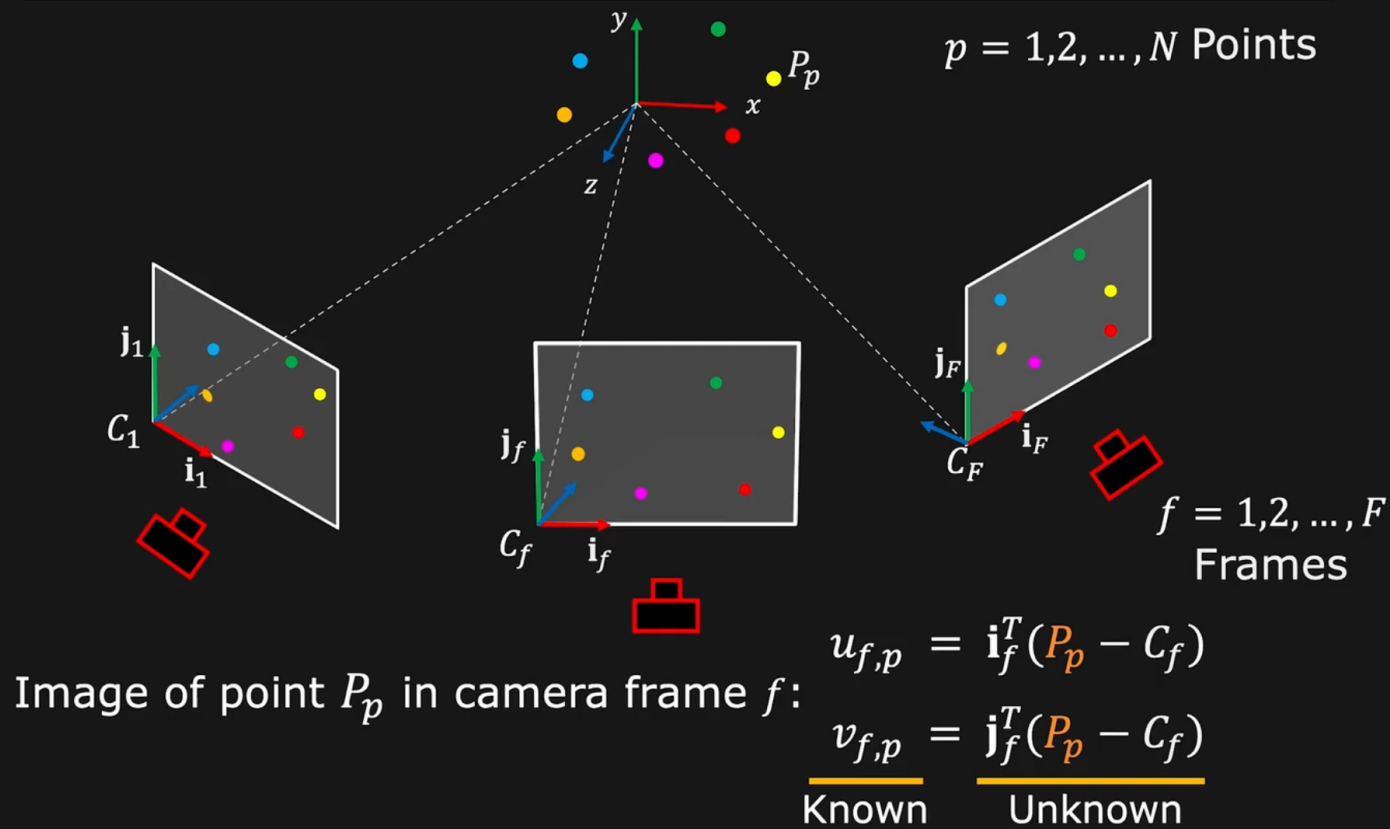


$$u = \mathbf{i}^T \mathbf{x}_c = \mathbf{i}^T (\mathbf{x}_w - \mathbf{c}_w) = \mathbf{i}^T (P - C)$$

$$v = \mathbf{j}^T \mathbf{x}_c = \mathbf{j}^T (\mathbf{x}_w - \mathbf{c}_w) = \mathbf{j}^T (P - C)$$

$$\begin{aligned} u &= \mathbf{i}^T (P - C) \\ v &= \mathbf{j}^T (P - C) \end{aligned}$$

Orthographic SFM

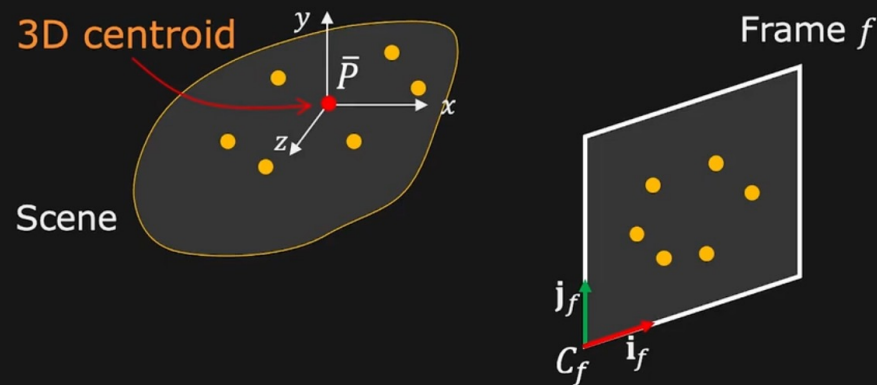


We can remove \mathbf{C}_f from equations to simplify SFM problem.

© 2020 Shree K. Nayar

Image from [Shree Nayar FPCV videos](#)

Centering Trick

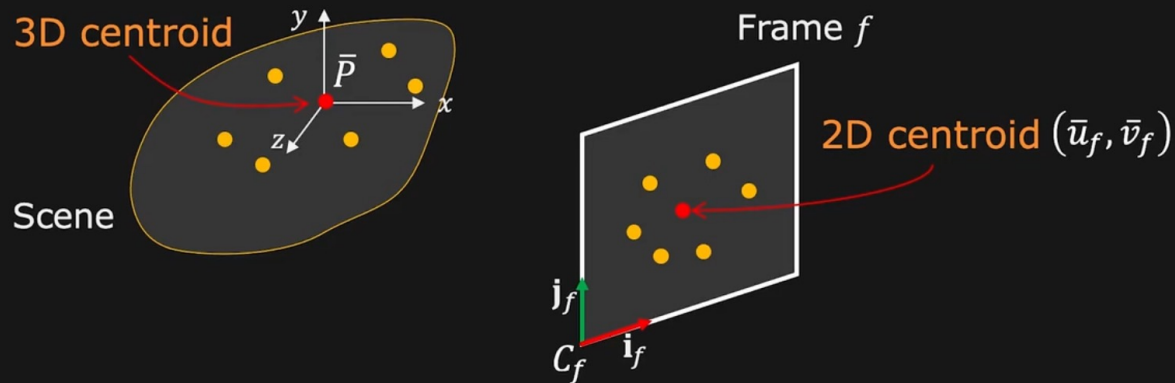


Assume origin of world at centroid of scene points:

$$\frac{1}{N} \sum_{p=1}^N P_p = \bar{P} = \mathbf{0}_w$$

We will compute scene points w.r.t their centroid!

Centering Trick



Centroid (\bar{u}_f, \bar{v}_f) of the image points in frame f :

$$\bar{u}_f = \frac{1}{N} \sum_{p=1}^N u_{f,p} = \frac{1}{N} \sum_{p=1}^N \mathbf{i}_f^T (P_p - C_f)$$

$$\bar{u}_f = \cancel{\frac{1}{N} \mathbf{i}_f^T \sum_{p=1}^N P_p} - \frac{1}{N} \sum_{p=1}^N \mathbf{i}_f^T C_f$$

$$\boxed{\bar{u}_f = -\mathbf{i}_f^T C_f}$$

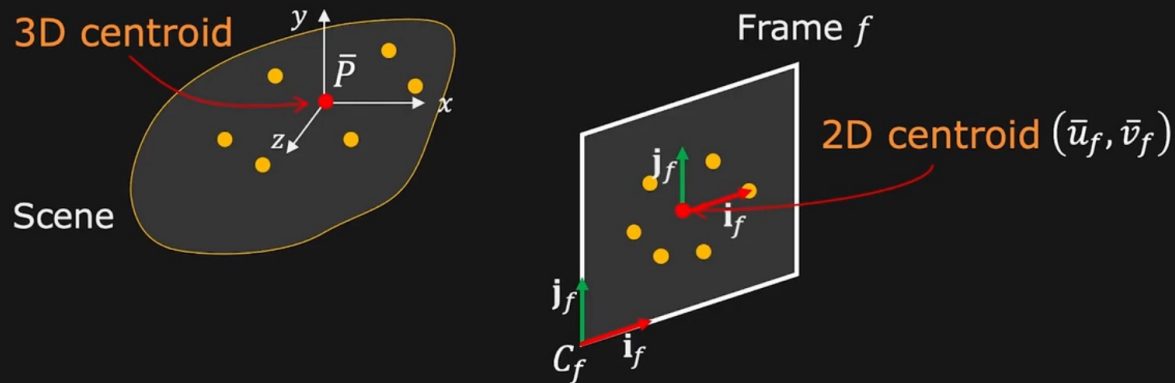
$$\bar{v}_f = \frac{1}{N} \sum_{p=1}^N v_{f,p} = \frac{1}{N} \sum_{p=1}^N \mathbf{j}_f^T (P_p - C_f)$$

$$\bar{v}_f = \cancel{\frac{1}{N} \mathbf{j}_f^T \sum_{p=1}^N P_p} - \frac{1}{N} \sum_{p=1}^N \mathbf{j}_f^T C_f$$

$$\boxed{\bar{v}_f = -\mathbf{j}_f^T C_f}$$

Image from [Shree Nayar FPCV videos](#)

Centering Trick



Shift camera origin to the centroid (\bar{u}_f, \bar{v}_f) .

Image points w.r.t. (\bar{u}_f, \bar{v}_f) :

$$\tilde{u}_{f,p} = u_{f,p} - \bar{u}_f$$

$$= \mathbf{i}_f^T (P_p - C_f) - \mathbf{i}_f^T C_f$$

$$\boxed{\tilde{u}_{f,p} = \mathbf{i}_f^T P_p}$$

$$\tilde{v}_{f,p} = v_{f,p} - \bar{v}_f$$

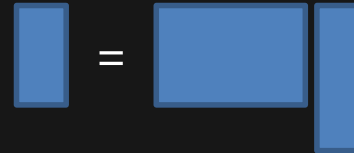
$$= \mathbf{j}_f^T (P_p - C_f) - \mathbf{j}_f^T C_f$$

$$\boxed{\tilde{v}_{f,p} = \mathbf{j}_f^T P_p}$$

Camera locations C_f now removed from equations.

Observation Matrix W

$$\begin{aligned}\tilde{u}_{f,p} &= \mathbf{i}_f^T P_p \\ \tilde{v}_{f,p} &= \mathbf{j}_f^T P_p\end{aligned} \quad \Rightarrow \quad \begin{bmatrix} \tilde{u}_{f,p} \\ \tilde{v}_{f,p} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_f^T \\ \mathbf{j}_f^T \end{bmatrix} P_p$$


$$\begin{bmatrix} \tilde{u}_{f,p} \\ \tilde{v}_{f,p} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_f^T \\ \mathbf{j}_f^T \end{bmatrix} P_p$$

Observation Matrix W

$$\begin{aligned}\tilde{u}_{f,p} &= \mathbf{i}_f^T P_p \\ \tilde{v}_{f,p} &= \mathbf{j}_f^T P_p\end{aligned} \quad \Rightarrow \quad \begin{bmatrix} \tilde{u}_{f,p} \\ \tilde{v}_{f,p} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_f^T \\ \mathbf{j}_f^T \end{bmatrix} P_p$$

$$\begin{array}{c} \text{Point 1} \quad \text{Point 2} \quad \dots \quad \text{Point N} \\ \text{Image 1} \quad \tilde{u}_{1,1} \quad \tilde{u}_{1,2} \quad \dots \quad \tilde{u}_{1,N} \\ \text{Image 2} \quad \tilde{u}_{2,1} \quad \tilde{u}_{2,2} \quad \dots \quad \tilde{u}_{2,N} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \text{Image F} \quad \tilde{u}_{F,1} \quad \tilde{u}_{F,2} \quad \dots \quad \tilde{u}_{F,N} \\ \hline \text{Image 1} \quad \tilde{v}_{1,1} \quad \tilde{v}_{1,2} \quad \dots \quad \tilde{v}_{1,N} \\ \text{Image 2} \quad \tilde{u}_{2,1} \quad \tilde{u}_{2,2} \quad \dots \quad \tilde{v}_{2,N} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \text{Image F} \quad \tilde{v}_{F,1} \quad \tilde{v}_{F,2} \quad \dots \quad \tilde{v}_{F,N} \end{array} = \begin{array}{c} \begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \\ \begin{array}{ccc} \text{Point 1} & \text{Point 2} & \text{Point N} \\ [P_1 & P_2 & \dots & P_N] \end{array} \end{array}$$

$W_{2F \times N}$ $M_{2F \times 3}$
 Centroid-Subtracted Feature Points (Known) Camera Motion (Unknown)

$S_{3 \times N}$
 Scene Structure (Unknown)

Image from [Shree Nayar FPCV videos](#)

Observation Matrix W

$$\begin{array}{c} \text{Image 1} \\ \text{Image 2} \\ \vdots \\ \text{Image F} \\ \text{Image 1} \\ \text{Image 2} \\ \vdots \\ \text{Image F} \end{array} \begin{bmatrix} \tilde{u}_{1,1} & \tilde{u}_{1,2} & \dots & \tilde{u}_{1,N} \\ \tilde{u}_{2,1} & \tilde{u}_{2,2} & \dots & \tilde{u}_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{u}_{F,1} & \tilde{u}_{F,2} & \dots & \tilde{u}_{F,N} \\ \tilde{v}_{1,1} & \tilde{v}_{1,2} & \dots & \tilde{v}_{1,N} \\ \tilde{u}_{2,1} & \tilde{u}_{2,2} & \dots & \tilde{v}_{2,N} \\ \vdots & \vdots & \vdots & \vdots \\ \tilde{v}_{F,1} & \tilde{v}_{F,2} & \dots & \tilde{v}_{F,N} \end{bmatrix} = \begin{bmatrix} \mathbf{i}_1^T \\ \mathbf{i}_2^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \mathbf{j}_2^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \begin{bmatrix} P_1 & P_2 & \dots & P_N \end{bmatrix}$$

Can we find M and S from W ?

Important Properties of Matrix Rank

- $\text{Rank}(A^T) = \text{Rank}(A)$
- $\text{Rank}(A_{m \times n} B_{n \times p}) = \min(\text{Rank}(A_{m \times n}), \text{Rank}(B_{n \times p}))$
 $\leq \min(m, n, p)$
- $\text{Rank}(AA^T) = \text{Rank}(A^T A) = \text{Rank}(A^T) = \text{Rank}(A)$
- $A_{m \times m}$ is invertible iff $\text{Rank}(A_{m \times m}) = m$

Rank of Observation Matrix

$$\begin{array}{ccccc} W & = & M & \times & S \\ 2F \times N & & 2F \times 3 & & 3 \times N \end{array}$$

We know:

$$\text{Rank}(MS) \leq \text{Rank}(M) \quad \text{Rank}(MS) \leq \text{Rank}(S)$$

$$\Rightarrow \text{Rank}(MS) \leq \min(3, 2F) \quad \text{Rank}(MS) \leq \min(3, N)$$

$$\Rightarrow \text{Rank}(W) = \text{Rank}(MS) \leq \min(3, N, 2F)$$

Rank Theorem: $\text{Rank}(W) \leq 3$

Singular Value Decomposition (SVD)

For any matrix A there exists a factorization:

$$A_{M \times N} = U_{M \times M} \cdot \Sigma_{M \times N} \cdot V^T_{N \times N}$$

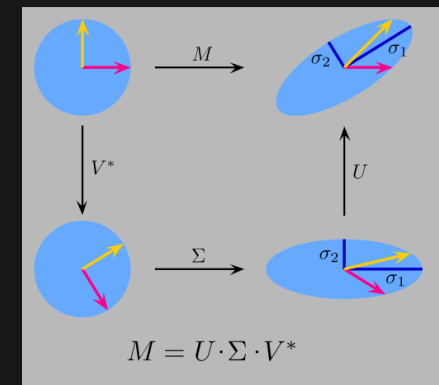
where U and V^T are **orthonormal** and Σ is **diagonal**.

MATLAB: `[U,S,V] = svd(A)`

$$\Sigma_{M \times N} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & \sigma_4 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & \sigma_N \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{bmatrix}$$

$\sigma_1, \dots, \sigma_N$: **Singular Values**

If $\text{Rank}(A) = r$ then A has r non-zero singular values.



Georg-Johann, CC BY-SA 3.0
<<https://creativecommons.org/licenses/by-sa/3.0/>>, via Wikimedia Commons

Enforcing Rank Constraint

Using SVD:

$$W = U \Sigma V^T$$

$$= \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ U & & & & & \\ & & & & & \\ & & & & & \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{bmatrix} \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ V^T & & & & & \end{bmatrix}$$

$2F \times 2F$
 $2F \times N$
 $N \times N$

Since $Rank(W) \leq 3$, $Rank(\Sigma) \leq 3$.

All except first 3 diagonal elements of Σ must be 0.

Enforcing Rank Constraint

Using SVD: $W = U \Sigma V^T$

$$= \begin{bmatrix} \begin{matrix} U_1 \\ U_2 \end{matrix} \end{bmatrix} \begin{bmatrix} \begin{matrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{matrix} \end{bmatrix} \begin{bmatrix} \begin{matrix} V_1^T \\ V_2^T \end{matrix} \end{bmatrix}$$

$\begin{matrix} 3 & 2F-3 \\ 2F \times 2F \end{matrix}$
 $\begin{matrix} 2F \times N \end{matrix}$
 $\begin{matrix} 3 & N-3 \\ N \times N \end{matrix}$

Since $\text{Rank}(W) \leq 3$, $\text{Rank}(\Sigma) \leq 3$.

Submatrices U_2 and V_2^T do not contribute to W .

Enforcing Rank Constraint

Using SVD: $W = U \Sigma V^T$

$$= \begin{bmatrix} \begin{matrix} U_1 \\ U_2 \end{matrix} \end{bmatrix} \begin{bmatrix} \begin{matrix} \sigma_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots \end{matrix} \end{bmatrix} \begin{bmatrix} \begin{matrix} V_1^T \\ V_2^T \end{matrix} \end{bmatrix}$$

$\begin{matrix} 3 & 2F-3 \\ 2F \times 2F \end{matrix}$
 $\begin{matrix} 2F \times N \end{matrix}$
 $\begin{matrix} 3 & N-3 \\ N \times N \end{matrix}$

$$\boxed{W = U_1 \Sigma_1 V_1^T}$$

$(2F \times 3) (3 \times 3) (3 \times P)$

Factorization (Finding M, S)

$$W = \underbrace{U_1 (\Sigma_1)^{1/2}}_{(2F \times 3)} \underbrace{(\Sigma_1)^{1/2} V_1^T}_{(3 \times N)}$$

$= M? \quad = S?$

Factorization (Finding M, S)

$$W = \underbrace{U_1 (\Sigma_1)^{1/2}}_{(2F \times 3)} \underbrace{(\Sigma_1)^{1/2} V_1^T}_{(3 \times N)}$$

$$= M? \quad = S?$$

Not so fast. Decomposition not unique!

For any 3x3 non-singular matrix Q :

$$W = \underbrace{U_1 (\Sigma_1)^{1/2} Q}_{(2F \times 3)} \underbrace{Q^{-1} (\Sigma_1)^{1/2} V_1^T}_{(3 \times N)} \text{ is also valid.}$$

$$= M \quad = S \dots \text{for some } Q.$$

How to find the matrix Q ?

Orthonormality of M

The Motion Matrix M :

$$M = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} = \underbrace{U_1(\Sigma_1)^{1/2}}_{\text{Computed}} Q = \begin{bmatrix} \hat{\mathbf{i}}_1^T \\ \vdots \\ \hat{\mathbf{i}}_F^T \\ \hat{\mathbf{j}}_1^T \\ \vdots \\ \hat{\mathbf{j}}_F^T \end{bmatrix} Q = \begin{bmatrix} \hat{\mathbf{i}}_1^T Q \\ \vdots \\ \hat{\mathbf{i}}_F^T Q \\ \hat{\mathbf{j}}_1^T Q \\ \vdots \\ \hat{\mathbf{j}}_F^T Q \end{bmatrix}$$

Computed

Orthonormality Constraints:

$$\mathbf{i}_f \cdot \mathbf{i}_f = \mathbf{i}_f^T \mathbf{i}_f = 1$$

$$\mathbf{j}_f \cdot \mathbf{j}_f = \mathbf{j}_f^T \mathbf{j}_f = 1$$

$$\mathbf{i}_f \cdot \mathbf{j}_f = \mathbf{i}_f^T \mathbf{j}_f = 0$$



$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f = 1$$

$$\hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f = 1$$

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f = 0$$

Orthonormality of M

- We have computed $(\hat{\mathbf{i}}_f^T, \hat{\mathbf{j}}_f^T)$ for $f = 1, \dots, F$.

$$\left. \begin{aligned} \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f &= 1 \\ \hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 1 \\ \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f &= 0 \end{aligned} \right\} Q \text{ is unknown.}$$

- Q is 3×3 matrix, 9 variables, $3F$ quadratic equations.
- Q can be solved with 3 or more images ($F \geq 3$) using Newton's method.

Final Solution:

$$M = U_1 (\Sigma_1)^{1/2} Q$$

Camera Motion

$$S = Q^{-1} (\Sigma_1)^{1/2} V_1^T$$

Scene Structure

Summary: Orthographic SFM

1. Detect and track feature points.
2. Create the centroid subtracted matrix W of corresponding feature points.
3. Compute SVD of W and enforce rank constraint.

$$W = U \Sigma V^T = U_1 \Sigma_1 V_1^T$$

$(2F \times 3) \quad (3 \times 3) \quad (3 \times P)$

4. Set $M = U_1 (\Sigma_1)^{1/2} Q$ and $S = Q^{-1} (\Sigma_1)^{1/2} V_1^T$.
5. Find Q by enforcing the orthonormality constraint.

Results from 1992

- Input

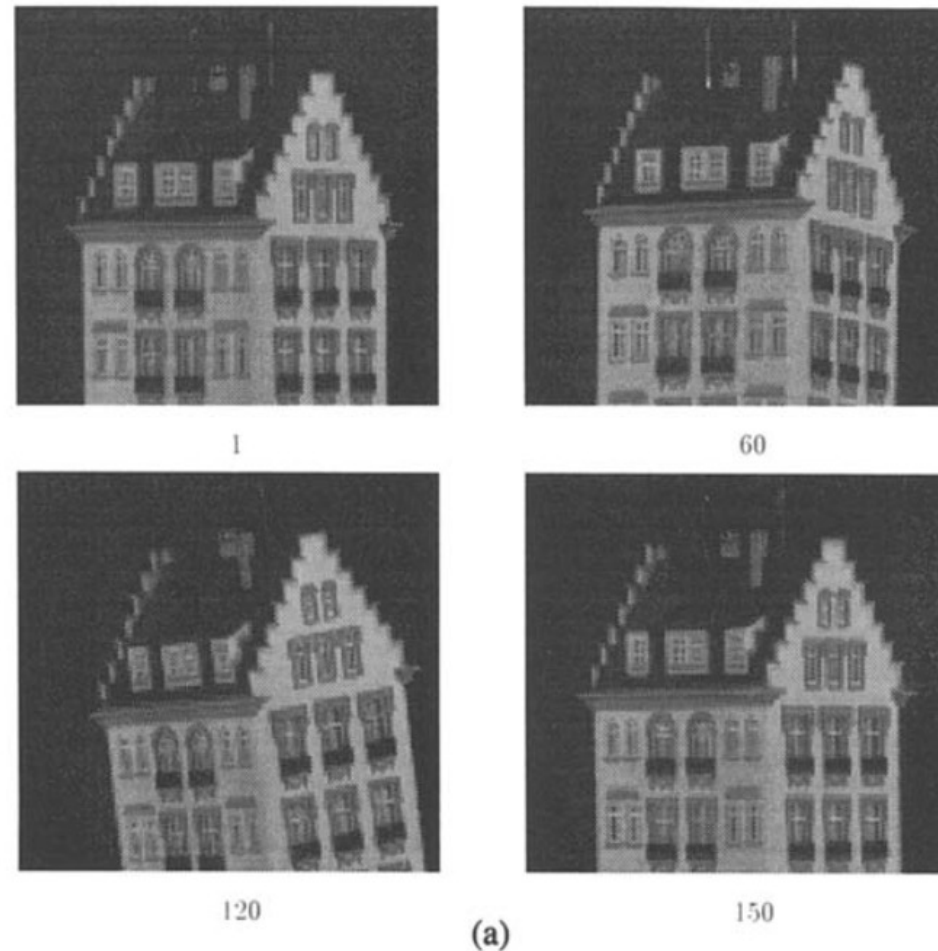


Fig. 2a. The “Hotel” stream: four of the 150 frames.

Image from [\[Tomas1992\]](#)

Results from 1992

- Resulting 3D reconstruction and new viewpoint rendering

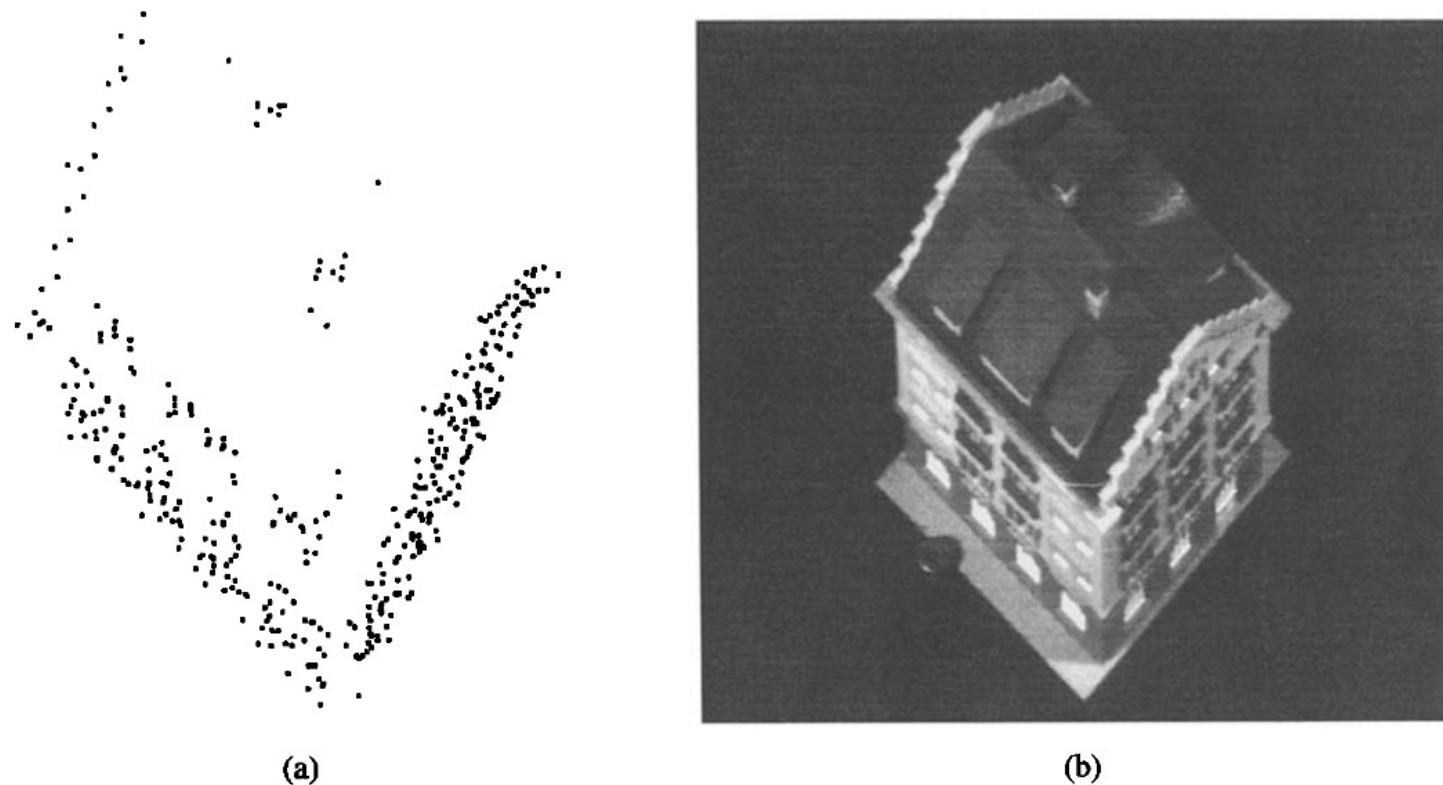


Fig. 4. Qualitative shape results for the “Hotel” stream: top view of the (a) computed and (b) actual shape.

Image from [\[Tomas1992\]](#)

More modern results: Photo Tourism

Photo Tourism

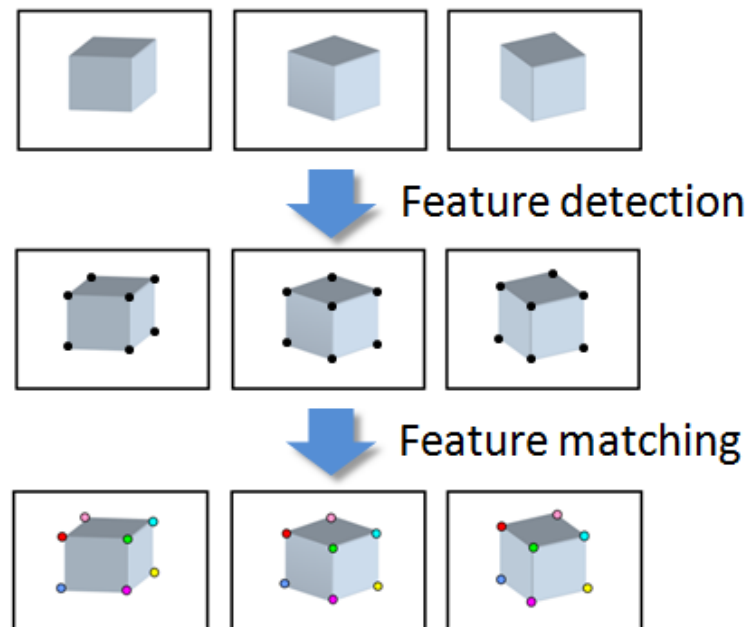
Exploring photo collections in 3D

Noah Snavely Steven M. Seitz Richard Szeliski
University of Washington *Microsoft Research*

SIGGRAPH 2006

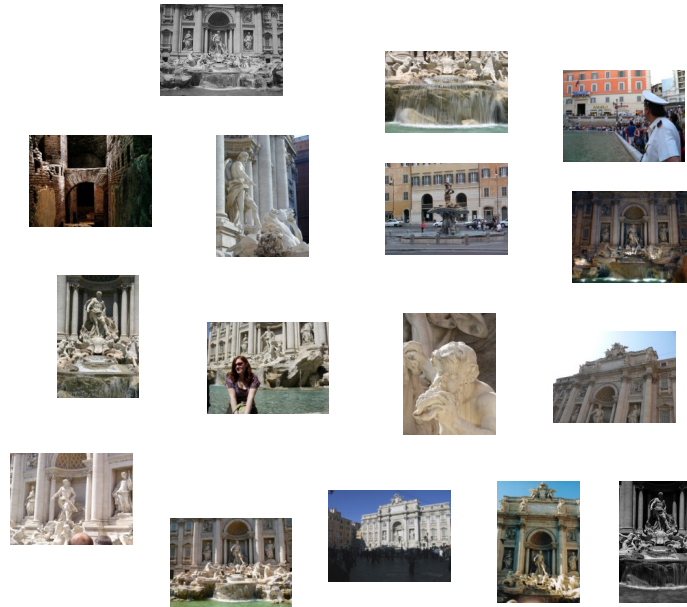
First step: how to get correspondence?

- Feature detection and matching



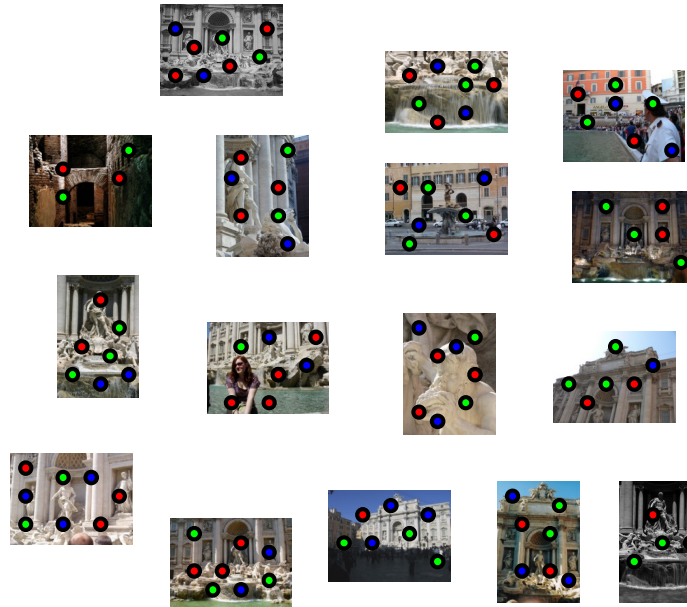
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



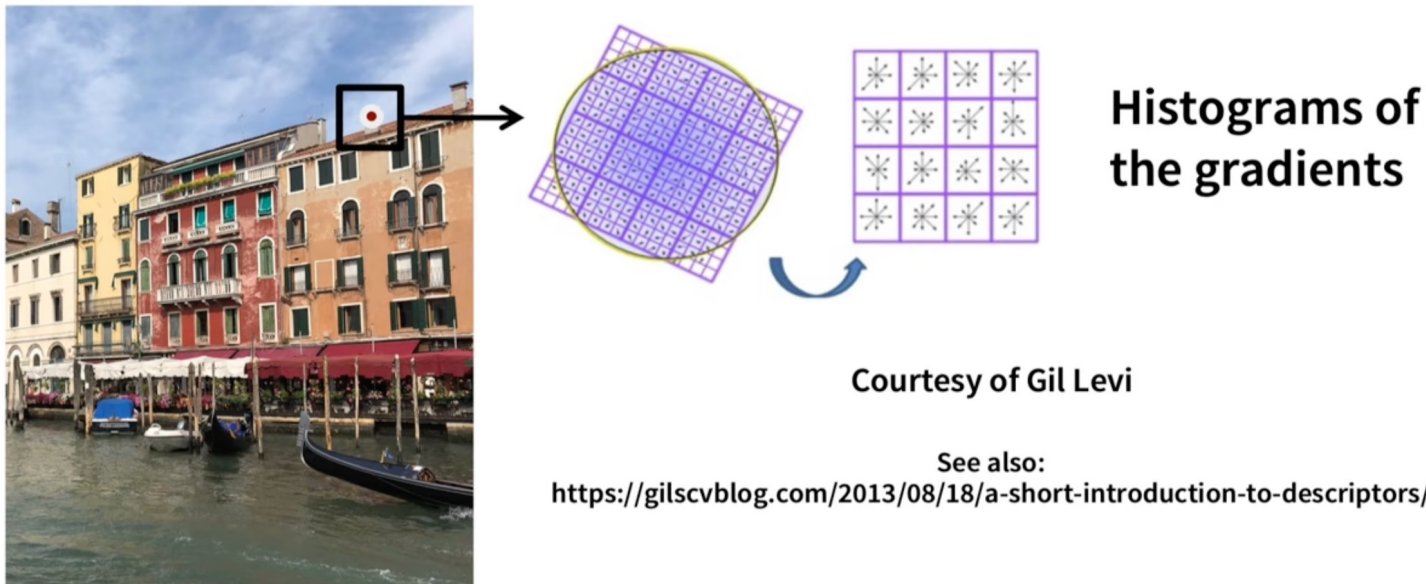
Feature detection

Detect features using [SIFT](#) [Lowe, IJCV 2004]



Video: Feature Detection

<https://www.youtube.com/watch?v=4AvTMVD9ig0>



Courtesy of Gil Levi

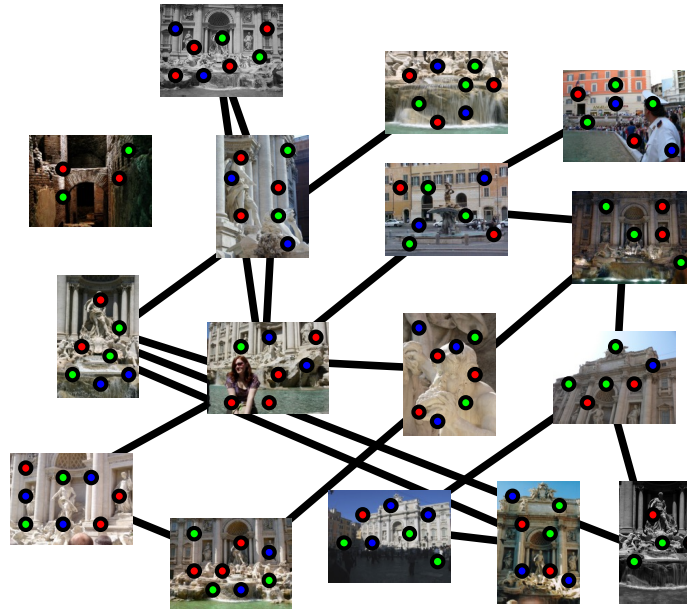
See also:

<https://gilscvblog.com/2013/08/18/a-short-introduction-to-descriptors/>

SIFT - 5 Minutes with Cyrill

Feature matching

Match features between each pair of images

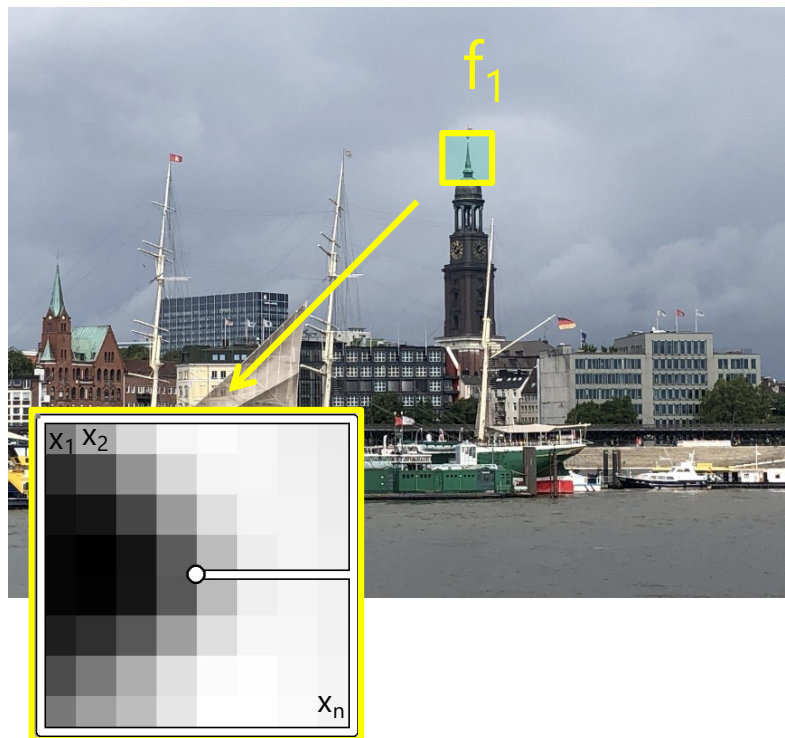


Feature distance

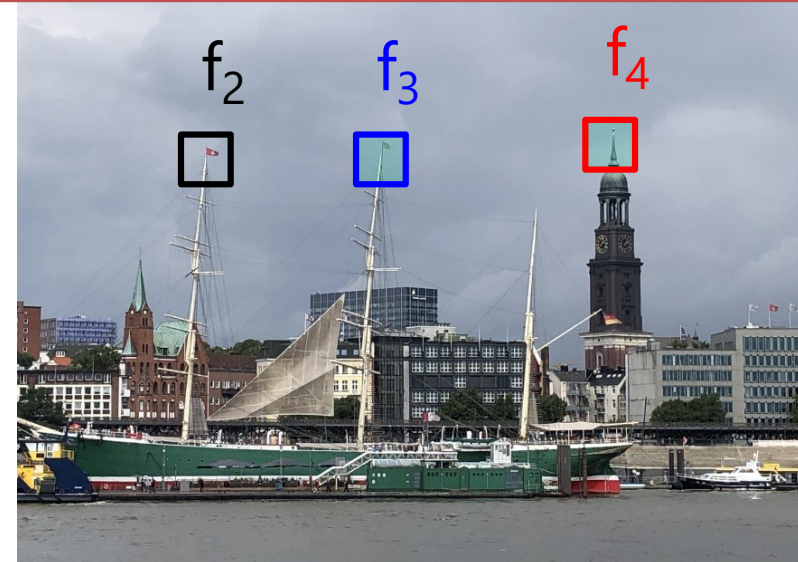
How to measure the difference between f_1 and f_2 , f_3 or f_4 ?

- A simple approach: L_2 -distance, $\| \vec{x}_{f_1} - \vec{x}_{f_i} \|$ (aka SSD)*

*Attention: It is about the distance of the feature vectors (so for SIFT a 128 dimensional space)



$$\vec{x}_{f_1} = (x_1, x_2, \dots, x_n)$$



$$\vec{x}_{f_2} = (x'_1, x'_2, \dots, x'_n)$$

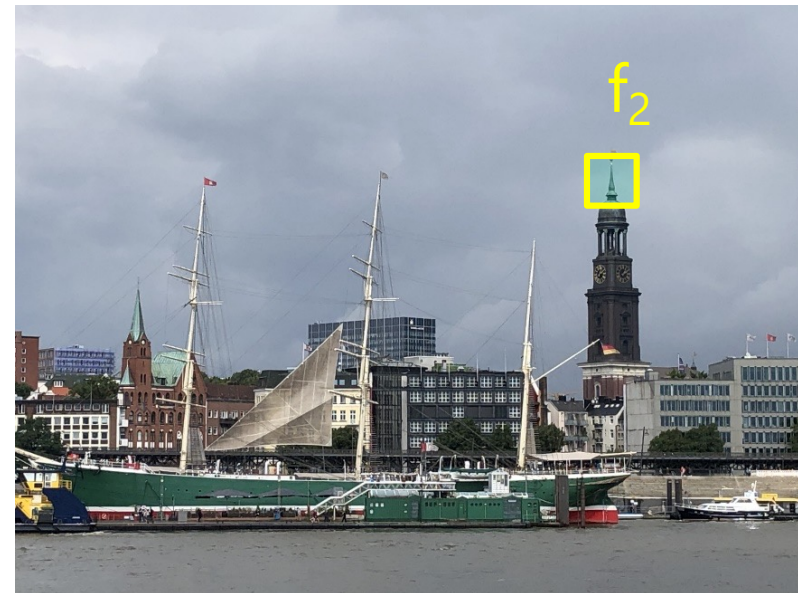
$$\vec{x}_{f_3} = (x''_1, x''_2, \dots, x''_n)$$

$$\vec{x}_{f_4} = (x'''_1, x'''_2, \dots, x'''_n)$$

Feature distance

How to measure the difference between f_1 and f_2 ?

- A simple approach: L_2 -distance, $\| \vec{x}_{f_1} - \vec{x}_{f_i} \|$ (aka SSD)

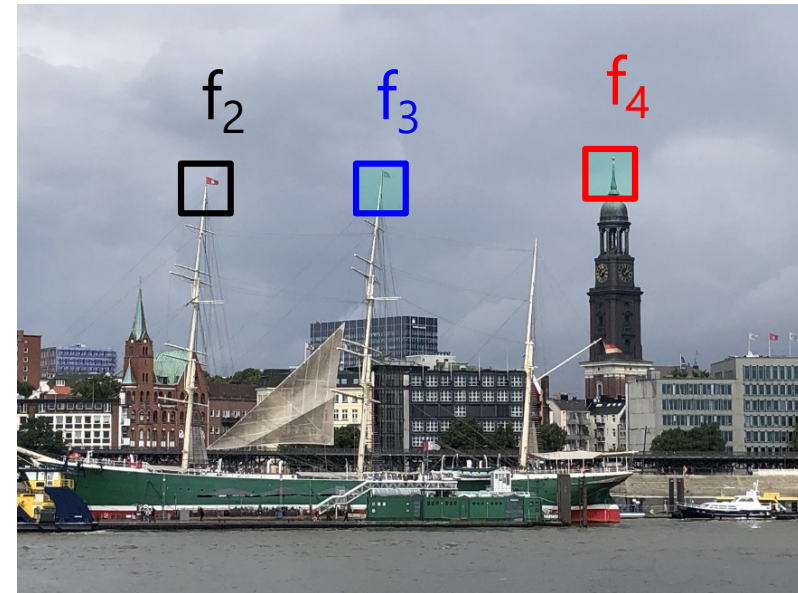


$$L_2\text{-distance: } \| \vec{x}_{f_1} - \vec{x}_{f_2} \| = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \dots + (x_n - x'_n)^2}$$

Feature distance

How to measure the difference between f_1 and f_2 , f_3 or f_4 ?

- A simple approach: L_2 -distance, $\|\vec{x}_{f_1} - \vec{x}_{f_i}\|$ (aka SSD)

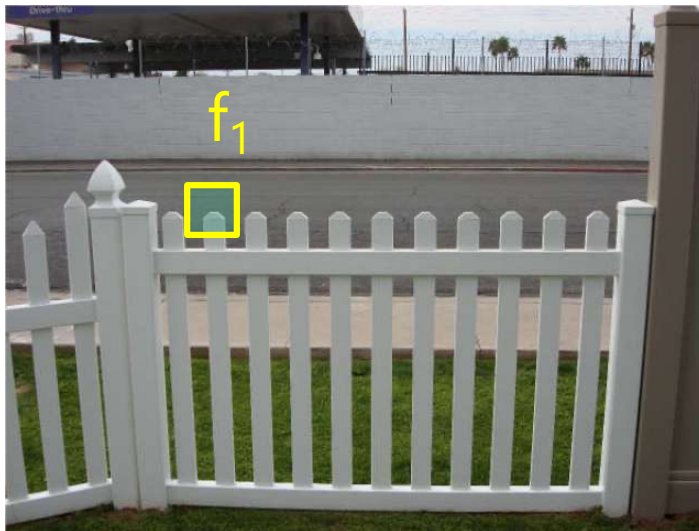


Which L_2 -distance is the smallest?

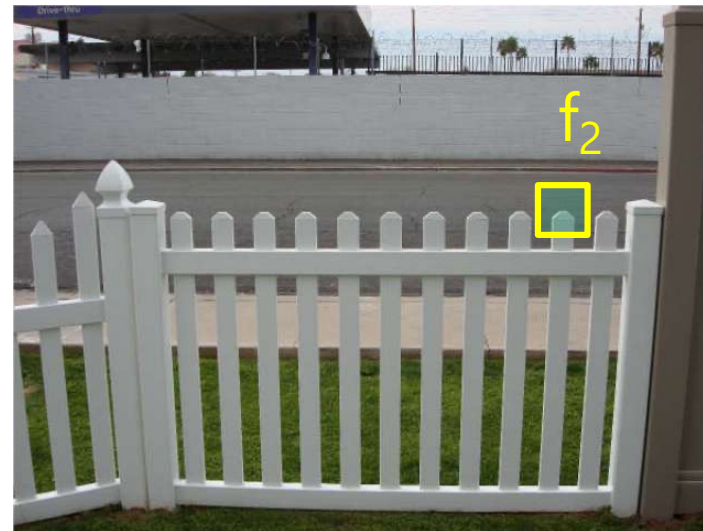
$$\begin{aligned} \times L_2(f_1, f_2) &= \|\vec{x}_{f_1} - \vec{x}_{f_2}\| \\ \times L_2(f_1, f_3) &= \|\vec{x}_{f_1} - \vec{x}_{f_3}\| \\ L_2(f_1, f_4) &= \|\vec{x}_{f_1} - \vec{x}_{f_4}\| \quad \checkmark \end{aligned}$$

Feature distance

Attention: this can result in small distances for ambiguous (false) matches



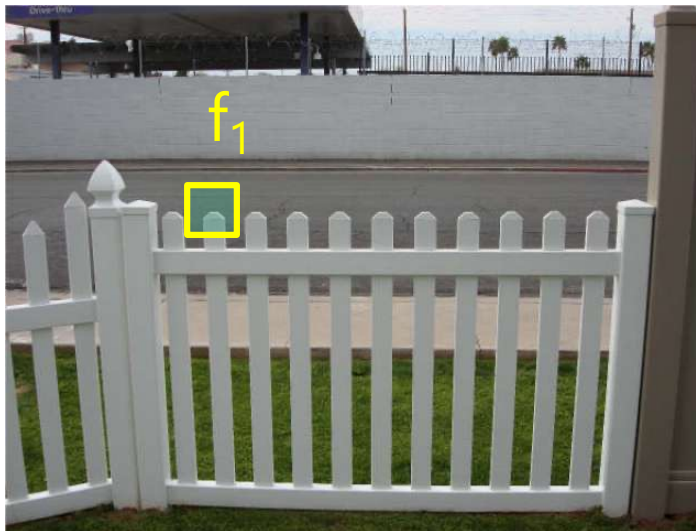
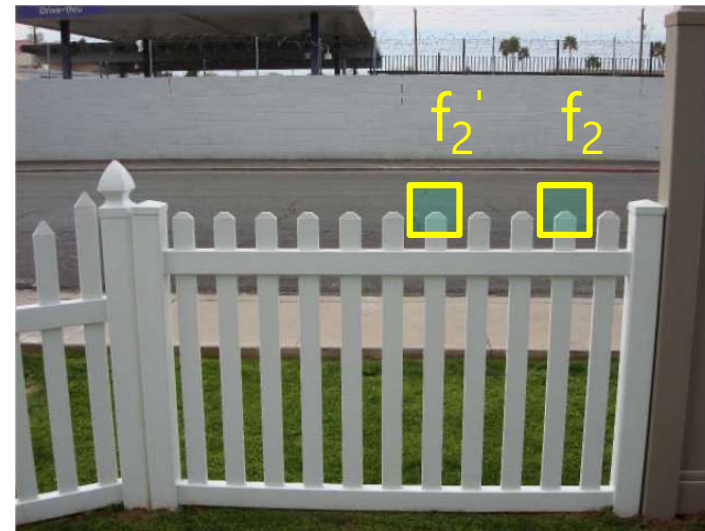
I_1



I_2

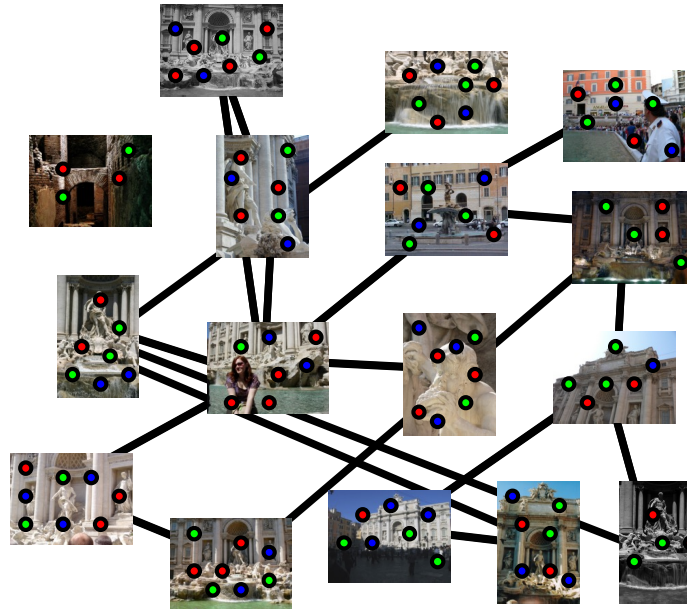
Feature distance

- Better approach: ratio distance = $\| f_1 - f_2 \| / \| f_1 - f_2' \|$
 - f_2 is the best SSD match to f_1 in I_2
 - f_2' is the 2nd best SSD match to f_1 in I_2
 - gives large values for ambiguous matches

 I_1  I_2

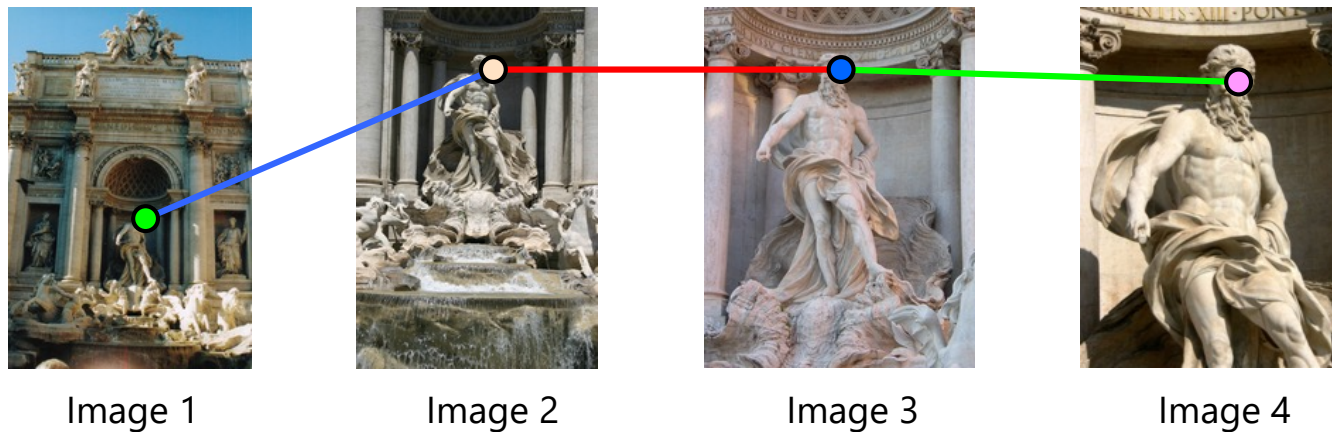
Feature matching

Refine matching using [RANSAC](#) to estimate fundamental matrix between each pair

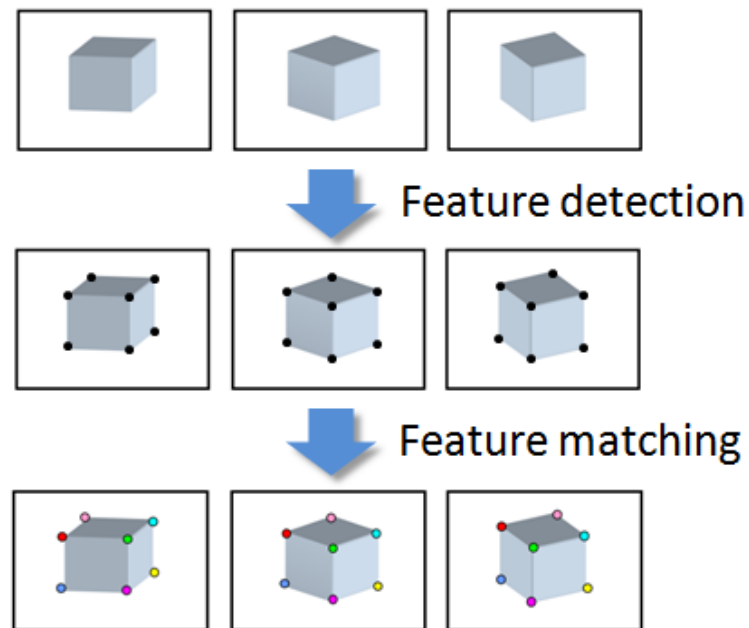


Correspondence estimation

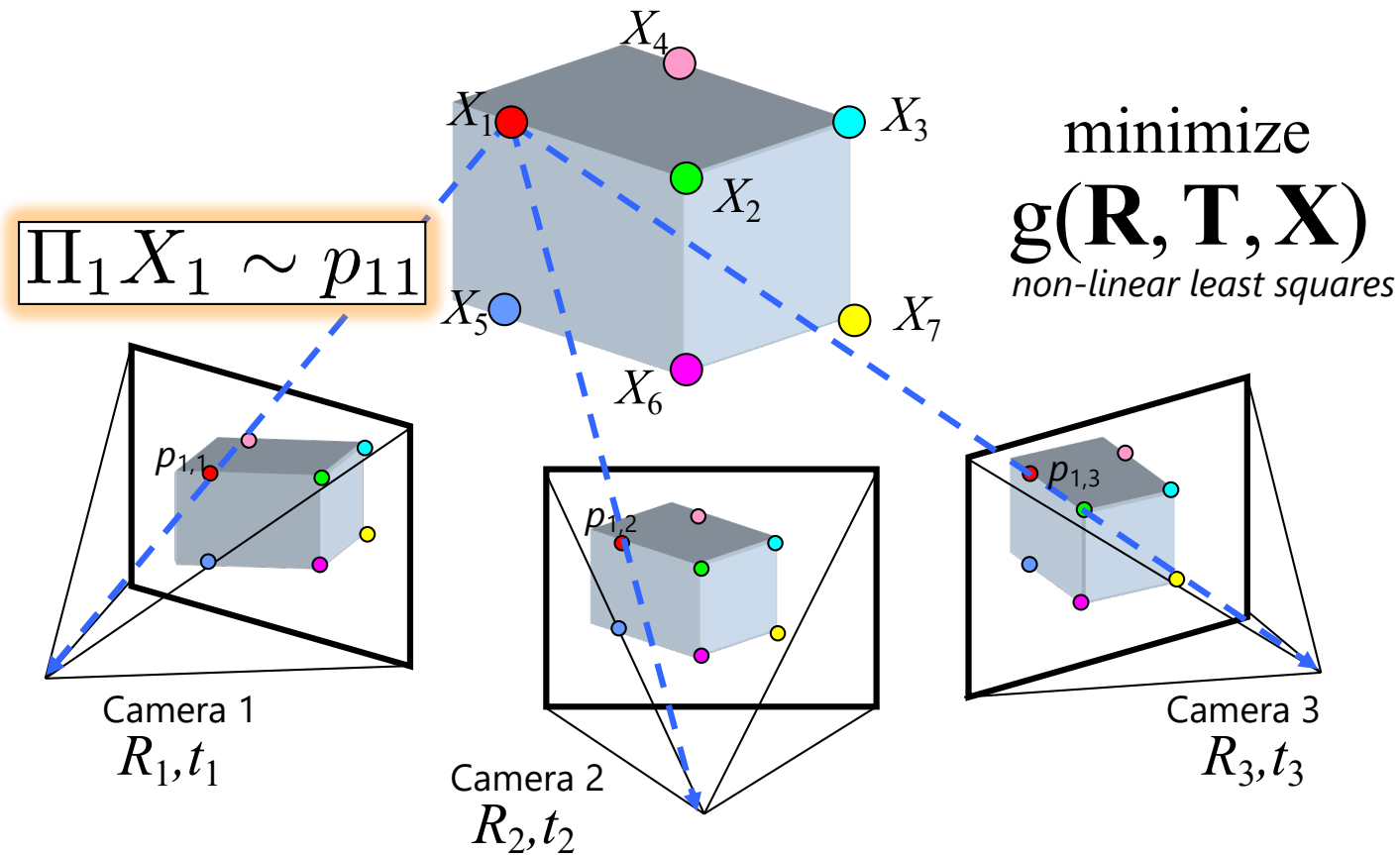
- Link up pairwise matches to form connected components of matches across several images



Input to Structure from Motion



Structure from motion



Problem size

- What are the variables?
 - Cameras and points
- How many variables per camera?
 - 6 (if calibrated), more if uncalibrated
- How many variables per point?
 - 3
- Trevi Fountain collection
 - 466 input photos
 - + > 100,000 3D points
 - = very large optimization problem

Structure from motion

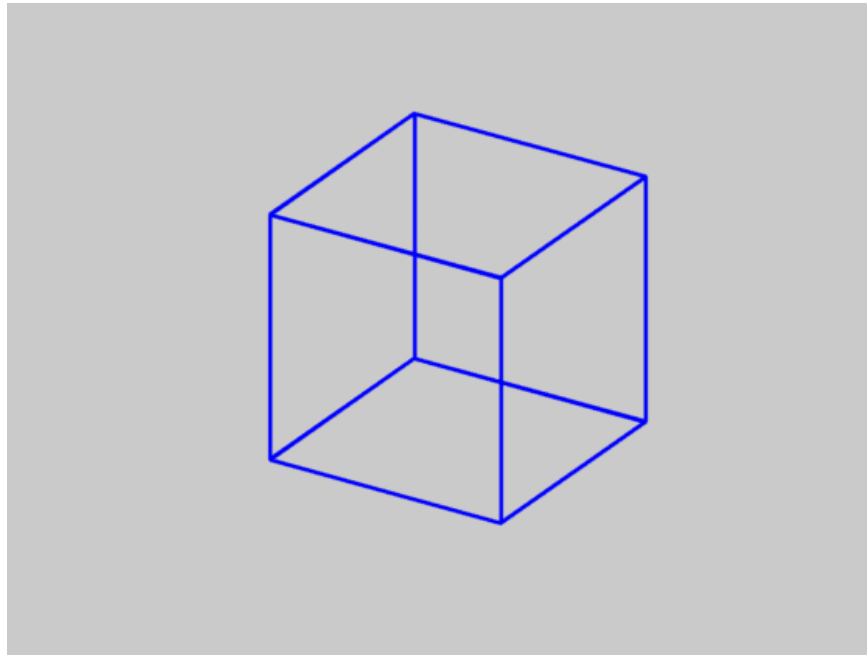
- Minimize sum of squared reprojection errors:

$$g(\mathbf{X}, \mathbf{R}, \mathbf{T}) = \sum_{i=1}^m \sum_{j=1}^n \underbrace{w_{ij}}_{\substack{\downarrow \\ \text{indicator variable:} \\ \text{is point } i \text{ visible in image } j?}} \cdot \left\| \underbrace{\mathbf{P}(\mathbf{x}_i, \mathbf{R}_j, \mathbf{t}_j)}_{\substack{\text{predicted} \\ \text{image location}}} - \underbrace{\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix}}_{\substack{\text{observed} \\ \text{image location}}} \right\|^2$$

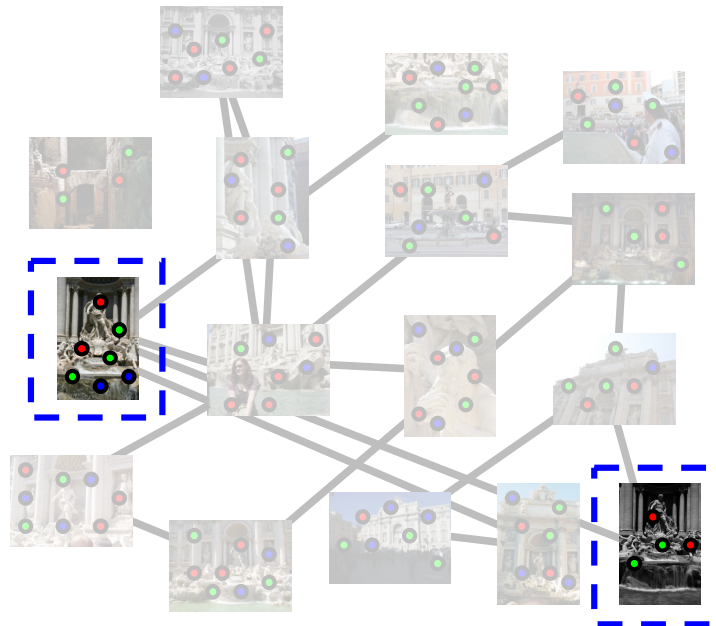
- Minimizing this function is called *bundle adjustment*
 - Optimized using non-linear least squares, e.g. Levenberg-Marquardt algorithm

Is SfM always uniquely solvable?

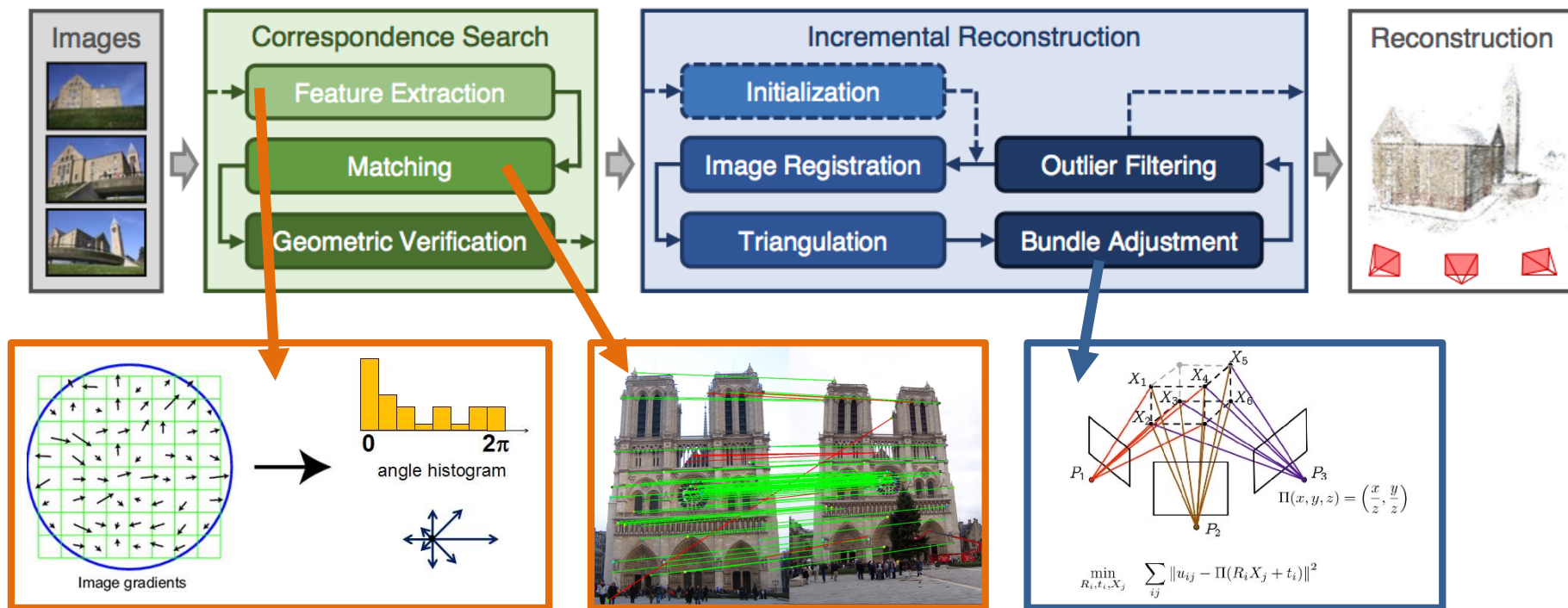
- No...



Incremental structure from motion



Complete SfM pipeline



[images from Hays2018, Szeliski2020 and [Schönberger2016](#)]

Incremental structure from motion



Video from Noah Snavely: [3D Old City of Dubrovnik](#)

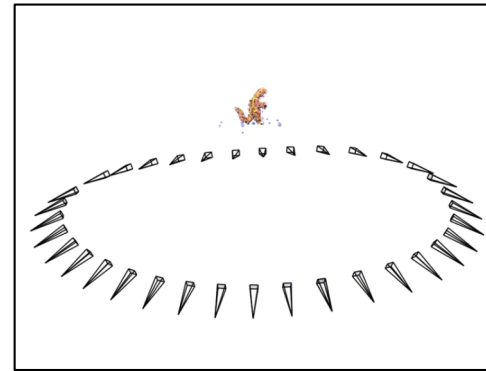
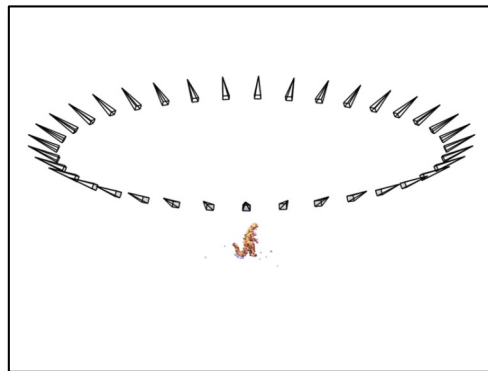
BigSFM: Reconstructing the World from Internet Photos



Images from <https://www.cs.cornell.edu/projects/bigsfm/> check the site for more results.

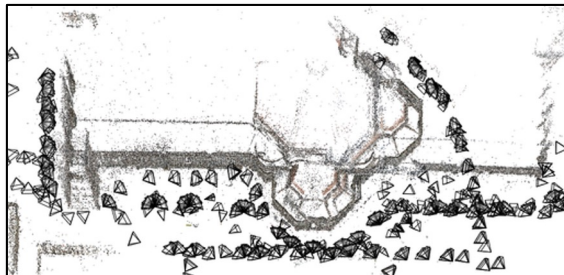
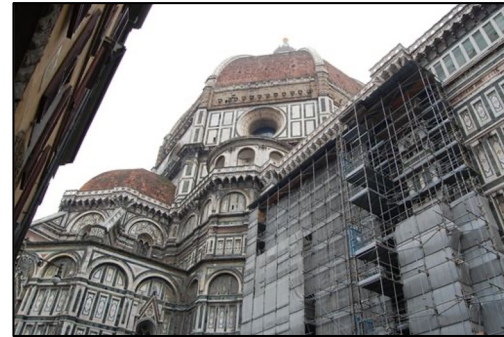
SfM – Failure cases

- Necker reversal



Structure from Motion – Failure cases

- Repetitive structures: Symmetries in man-made scenes

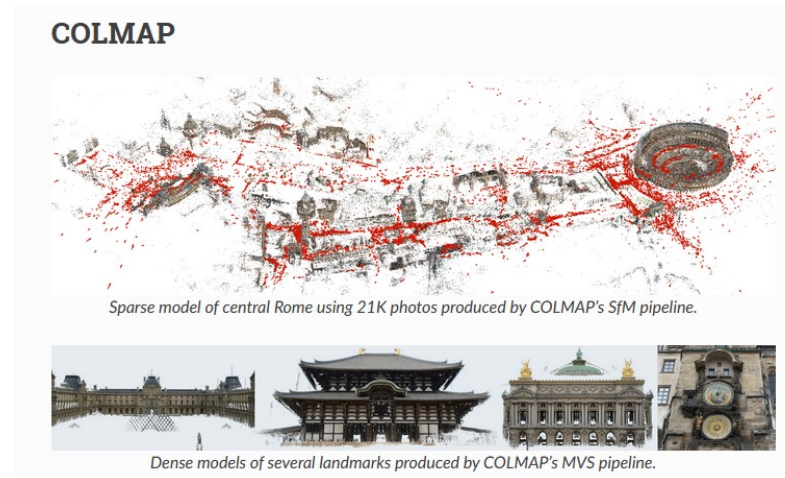


SfM applications

- 3D modeling
- Surveying
- Robot navigation and mapmaking
- Virtual and augmented reality
- Visual effects ("Match moving")
 - https://www.youtube.com/watch?v=RdYWp70P_kY

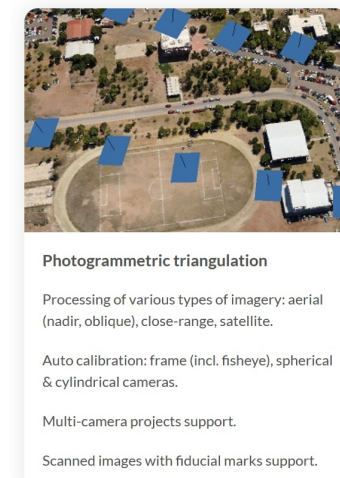
SfM implementations

- The scientific standard (foss):
<https://colmap.github.io/>



- Agisoft Metashape:
<https://www.agisoft.com/features/professional-edition/>
– Licenses available at HFU

- see more at <https://tu-dresden.de/Photogrammetrie>



Applications: Virtual Reality & Augmented Reality



Oculus

<https://www.youtube.com/watch?v=KOG7yTz1iTA>

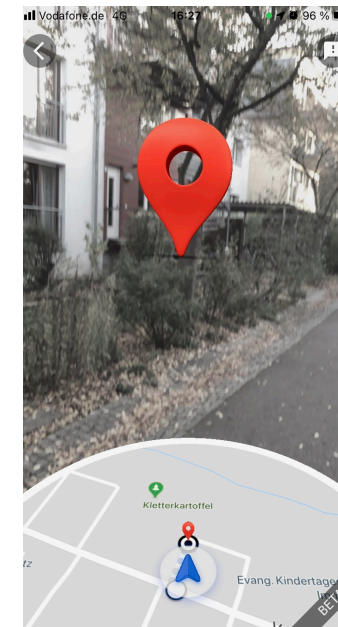
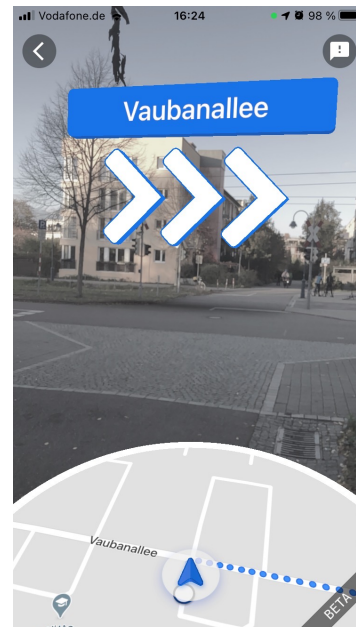


Hololens

<https://www.youtube.com/watch?v=FMtvrTGnPO4>

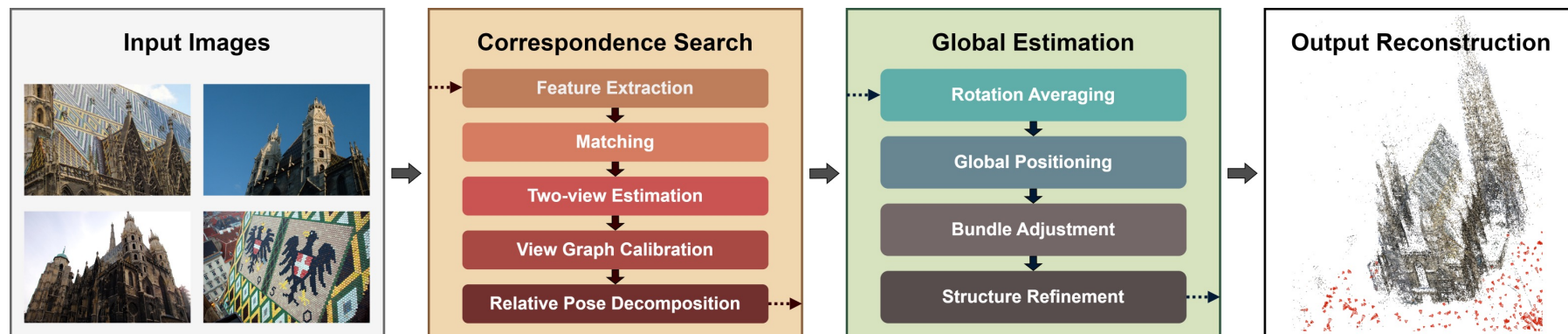
Application: AR walking directions

- Feature based localization → [ARCore 2022](#)



New trends

- GLOMAP
– Improvement of COLMAP

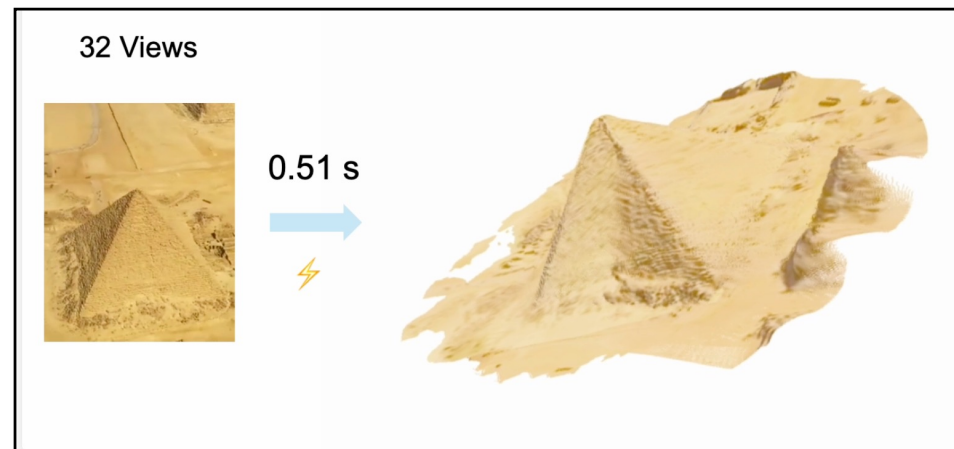


New trends

[Dust3r](#) and [Mast3r](#) (Naver Labs)

- the idea: do not start from scratch for each scene
- using large visual foundation models (similar to DepthAnything)

[VGGT](#) (Oxford + Meta AI)



- More to be explored in the next workshop.