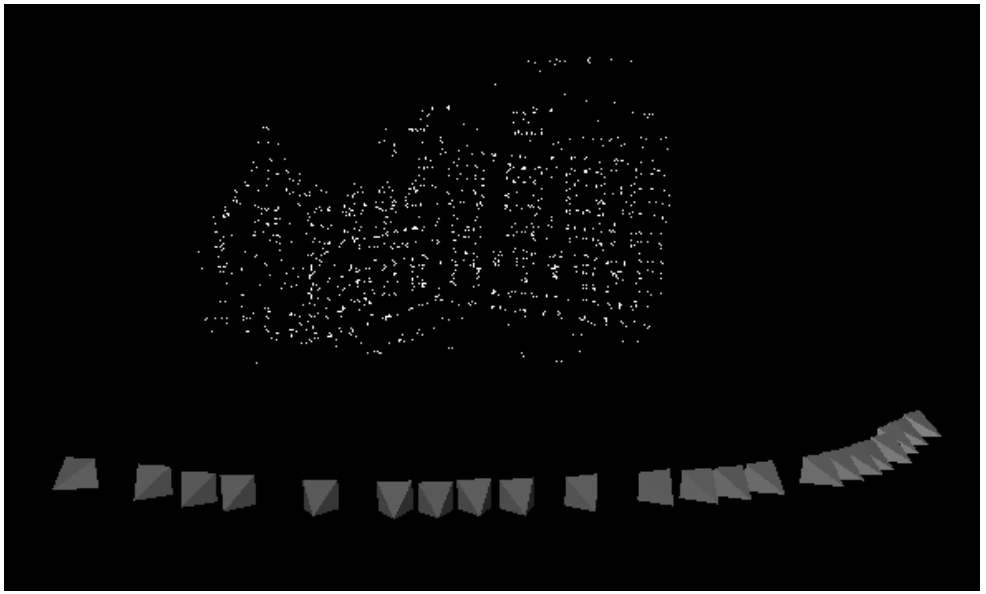
Structure from Motion



Read Chapter 7 in Szeliski's book

Schedule (tentative)

#	date	topic
1	Sep.19	Introduction and geometry
2	Sep.26	Camera models and calibration
3	Oct.3	Invariant features
4	Oct.10	Optical flow & Particle Filters
5	Oct.17	Multiple-view geometry
6	Oct.24	Model fitting (RANSAC, EM, …)
7	Oct.31	Image segmentation
8	Nov.7	Stereo Matching & MVS
9	Nov.14	**Structure-from-Motion & SLAM**
10	Nov.21	Specific object recognition
11	Nov.28	Shape from X
12	Dec.5	Object category recognition
13	Dec.12	Tracking
14	Dec.19	Research Overview & Lab tours

Today's class

- Structure from motion
  - factorization
  - sequential
  - bundle adjustment

Factorization

- Factorise observations in structure of the scene and motion/calibration of the camera

- Use **all points** in **all images** at the same time

✓ Affine factorisation

✓ Projective factorisation

Affine camera

The affine projection equations are

$$\begin{bmatrix} x_{ij} \\ y_{ij} \\ 1 \end{bmatrix} = \begin{bmatrix} P_i^x \\ P_i^y \\ 0001 \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x_{ij} \\ y_{ij} \end{bmatrix} = \begin{bmatrix} P_i^x \\ P_i^y \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} x_{ij} - P_i^{x4} \\ y_{ij} - P_i^{y4} \end{bmatrix} = \begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{bmatrix} = \begin{bmatrix} \overline{P}_i^{\,x} \\ \overline{P}_i^{\,y} \end{bmatrix} \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}$$

how to find the origin? or for that matter a 3D reference point?

affine projection preserves center of gravity

$$\tilde{x}_{ij} = x_{ij} - \sum_i x_{ij} \qquad \tilde{y}_{ij} = y_{ij} - \sum_i y_{ij}$$

Orthographic factorization

(Tomasi Kanade' 92)

The orthographic projection equations are

$$\overline{\mathrm{m}}_{ij} = \overline{\mathbf{P}}_i \, \overline{\mathrm{M}}_j, i = 1,\ldots, m, \; j = 1,\ldots, n$$

where

$$\overline{\mathrm{m}}_{ij} = \begin{bmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{bmatrix}, \overline{\mathbf{P}}_i = \begin{bmatrix} \overline{P}_i^{\,x} \\ \overline{P}_i^{\,y} \end{bmatrix}, \overline{\mathrm{M}}_j = \begin{bmatrix} X_j \\ Y_j \\ Z_j \end{bmatrix}$$

All equations can be collected for all i and j

$$\overline{\mathbf{m}} = \overline{\mathbf{P}}\overline{\mathbf{M}}$$

where

$$\overline{\mathbf{m}} = \begin{bmatrix} \overline{\mathrm{m}}_{11} & \overline{\mathrm{m}}_{12} & \cdots & \overline{\mathrm{m}}_{1n} \\ \overline{\mathrm{m}}_{21} & \overline{\mathrm{m}}_{22} & \cdots & \overline{\mathrm{m}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathrm{m}}_{m1} & \overline{\mathrm{m}}_{m2} & \cdots & \overline{\mathrm{m}}_{mn} \end{bmatrix}, \; \overline{\mathbf{P}} = \begin{bmatrix} \overline{\mathbf{P}}_1 \\ \overline{\mathbf{P}}_2 \\ \vdots \\ \overline{\mathbf{P}}_m \end{bmatrix}, \overline{\mathbf{M}} = \begin{bmatrix} \mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_n \end{bmatrix}$$

Note that $\mathbf{P}$ and $\mathbf{M}$ are resp. 2*m*x3 and 3x*n* matrices and therefore the rank of $\mathbf{m}$ is at most 3

Orthographic factorization

(Tomasi Kanade' 92)

Factorize $\mathbf{m}$ through singular value decomposition

$$\overline{\mathbf{m}} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}$$

An affine reconstruction is obtained as follows

$$\tilde{\mathbf{P}} = \mathbf{U}, \tilde{\mathbf{M}} = \Sigma\mathbf{V}^{\mathsf{T}}$$

Closest rank-3 approximation yields MLE!

$$\min \left\| \begin{bmatrix} \overline{\mathrm{m}}_{11} & \overline{\mathrm{m}}_{12} & \cdots & \overline{\mathrm{m}}_{1n} \\ \overline{\mathrm{m}}_{21} & \overline{\mathrm{m}}_{22} & \cdots & \overline{\mathrm{m}}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \overline{\mathrm{m}}_{m1} & \overline{\mathrm{m}}_{m2} & \cdots & \overline{\mathrm{m}}_{mn} \end{bmatrix} - \begin{bmatrix} \overline{\mathbf{P}}_1 \\ \overline{\mathbf{P}}_2 \\ \vdots \\ \overline{\mathbf{P}}_m \end{bmatrix} \begin{bmatrix} \mathrm{M}_1, \mathrm{M}_2, \ldots, \mathrm{M}_n \end{bmatrix} \right\|$$

$$\left(\sum_{ij} \|\mathrm{m}_{ij}\|^2 = \|\bar{\mathrm{m}} - \tilde{\mathrm{P}}\tilde{\mathrm{M}}\|_{\mathtt{frob}} \right)$$

Orthographic factorization

(Tomasi Kanade' 92)

Factorize $\mathbf{m}$ through singular value decomposition

$$\overline{\mathbf{m}} = \mathbf{U}\Sigma\mathbf{V}^{\mathsf{T}}$$

An affine reconstruction is obtained as follows

$$\widetilde{\mathbf{P}} = \mathbf{U}, \widetilde{\mathbf{M}} = \Sigma\mathbf{V}^{\mathsf{T}}$$

A metric reconstruction is obtained as follows

$$\overline{\mathbf{P}} = \widetilde{\mathbf{P}}\mathbf{A}^{-1}, \overline{\mathbf{M}} = \mathbf{A}\widetilde{\mathbf{M}}$$

Where A is computed from

$$\widetilde{\mathbf{P}}_i^{x\mathsf{T}} \mathbf{C} \widetilde{\mathbf{P}}_i^{x} = 1$$

$$\widetilde{\mathbf{P}}_i^{y\mathsf{T}} \mathbf{C} \widetilde{\mathbf{P}}_i^{y} = 1$$

$$\widetilde{\mathbf{P}}_i^{x\mathsf{T}} \mathbf{C} \widetilde{\mathbf{P}}_i^{y} = 0$$

3 linear equations per view on symmetric matrix $\mathbf{C}$ (6DOF)

$\mathbf{A}$ can be obtained from $\mathbf{C}$ through Cholesky factorisation and inversion

Examples

Tomasi Kanade’92,
Poelman & Kanade’94

Examples

Tomasi Kanade’92,
Poelman & Kanade’94

Examples

Tomasi Kanade’92,
Poelman & Kanade’94

Examples

Tomasi Kanade’92,
Poelman & Kanade’94

Perspective factorization

The camera equations

$$\lambda_{ij}\mathrm{m}_{ij} = \mathbf{P}_i \mathrm{M}_j, i = 1,\ldots, m,\ j = 1,\ldots, m$$

for a fixed image i can be written in matrix form as

$$\mathbf{m}_i \Lambda_i = \mathbf{P}_i \mathbf{M}$$

where

$$\mathbf{m}_i = \left[\mathrm{m}_{i1}, \mathrm{m}_{i2},\ldots,\mathrm{m}_{im}\right],\ \mathbf{M} = \left[\mathrm{M}_1, \mathrm{M}_2,\ldots,\mathrm{M}_m\right]$$
$$\Lambda_i = \mathrm{diag}\left(\lambda_{i1}, \lambda_{i2},\ldots,\lambda_{im}\right)$$

Perspective factorization

All equations can be collected for all i as

$$\mathbf{m} = \mathbf{PM}$$

where

$$\mathbf{m} = \begin{bmatrix} \mathbf{m}_1\Lambda_1 \\ \mathbf{m}_2\Lambda_2 \\ \cdots \\ \mathbf{m}_n\Lambda_n \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \cdots \\ \mathbf{P}_m \end{bmatrix}$$

In these formulas $\mathbf{m}$ are known, but Λ_i,$\mathbf{P}$ and $\mathbf{M}$ are unknown

Observe that $\mathbf{PM}$ is a product of a 3mx4 matrix and a 4xn matrix, i.e. it is a rank-4 matrix

Perspective factorization algorithm

Assume that Λ_i are known, then $\mathbf{PM}$ is known.

Use the singular value decomposition

$$\mathbf{PM}=U\Sigma V^T$$

In the noise-free case

$$\Sigma=\text{diag}(\sigma_1,\sigma_2,\sigma_3,\sigma_4,0,\ldots,0)$$

and a reconstruction can be obtained by setting:

$\mathbf{P}$=the first four columns of $U\Sigma$.

$\mathbf{M}$=the first four rows of V.

Iterative perspective factorization

When Λ_i are unknown the following algorithm can be used:

1. Set λ_{ij}=1 (affine approximation).

2. Factorize $\mathbf{PM}$ and obtain an estimate of $\mathbf{P}$ and $\mathbf{M}$. If σ_5 is sufficiently small then STOP.

3. Use $\mathbf{m}$, $\mathbf{P}$ and $\mathbf{M}$ to estimate Λ_i from the camera equations (linearly) $\mathbf{m}_i \Lambda_i = \mathbf{P}_i\mathbf{M}$

4. Goto 2.

In general the algorithm minimizes the *proximity measure* $P(\Lambda,\mathbf{P},\mathbf{M})=\sigma_5$

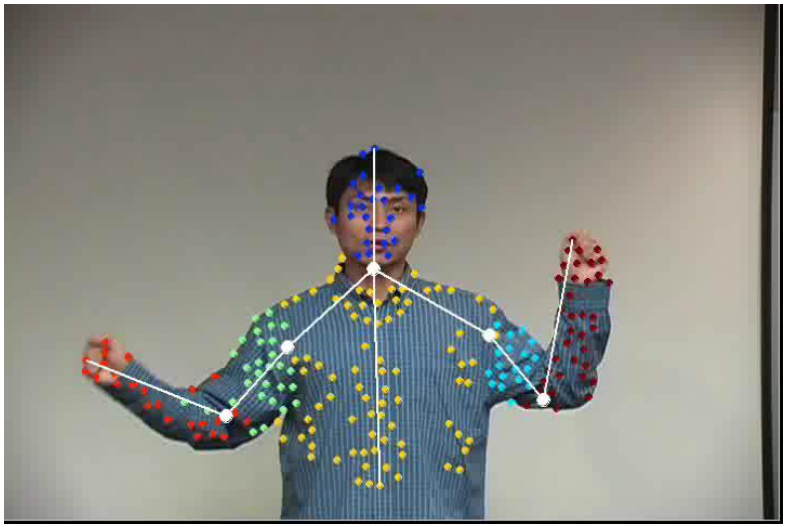
Note that structure and motion recovered up to an arbitrary projective transformation

Further Factorization work

Factorization with uncertainty

(Irani & Anandan, IJCV' 02)

Factorization for dynamic scenes

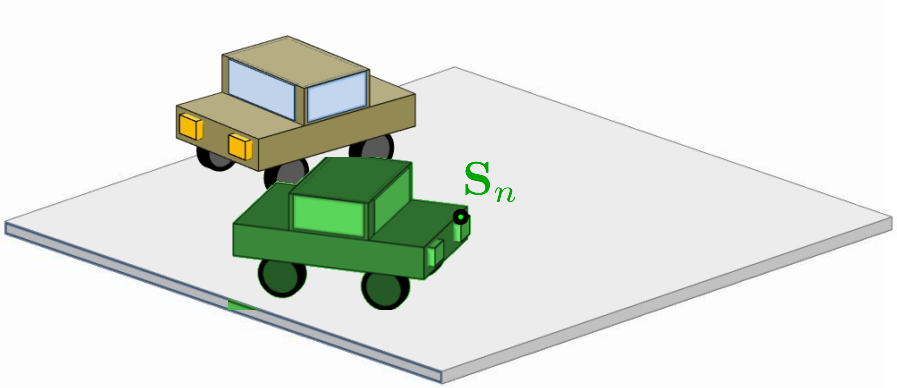


(Costeira and Kanade '94)

(Bregler et al. '00, Brand '01)

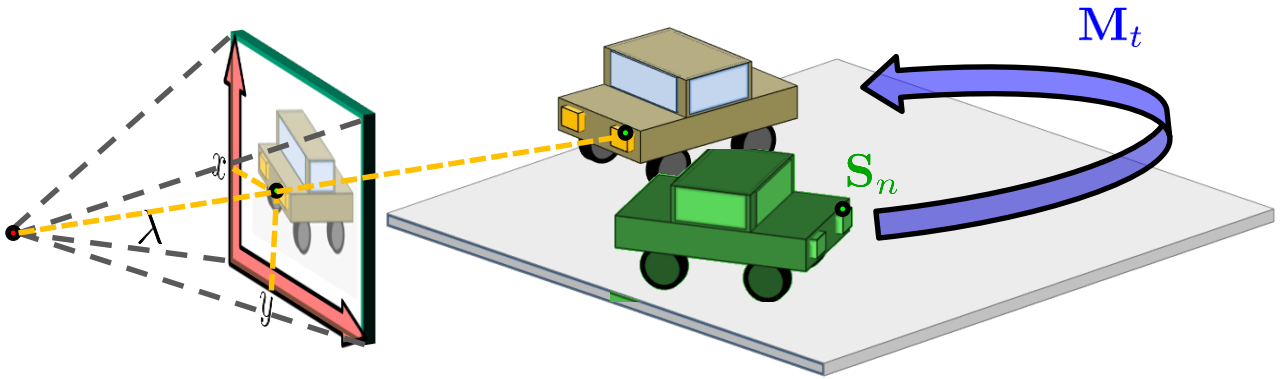
(Yan and Pollefeys, '05/' 06)

Structure from Motion (SfM)



- Projective camera

Structure from Motion (SfM)

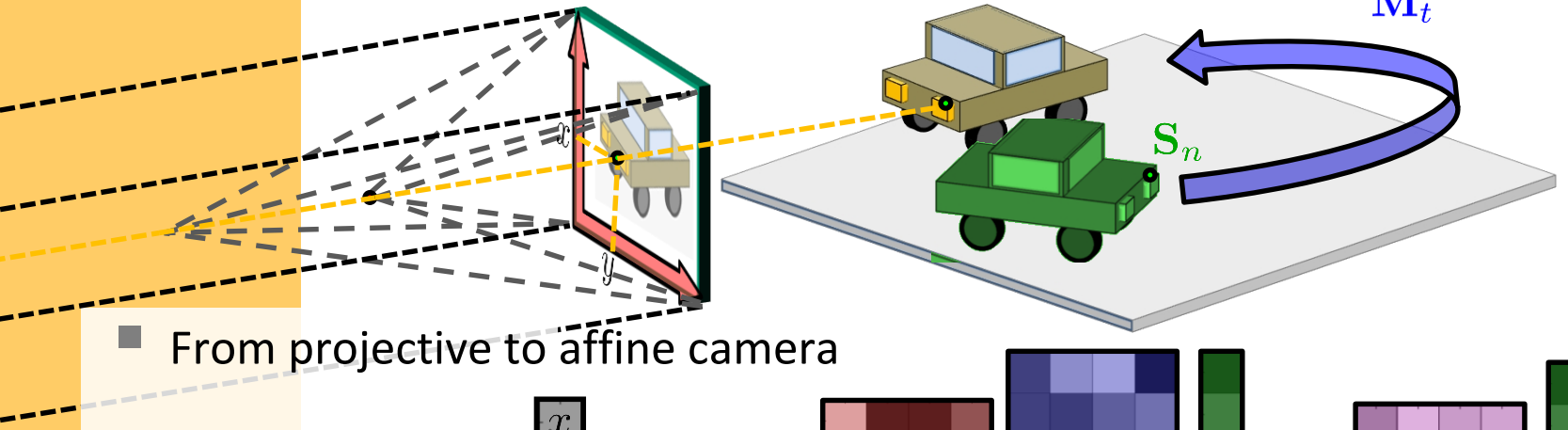


- Projective camera

$$\mathbf{x}_{t,k,n} = \mathbf{C}_k \; \mathbf{M}_t \; \mathbf{S}_n$$

$$\begin{bmatrix} x \\ y \\ \lambda \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ 1 \end{bmatrix}$$

Affine Structure from Motion (SfM)



- From projective to affine camera

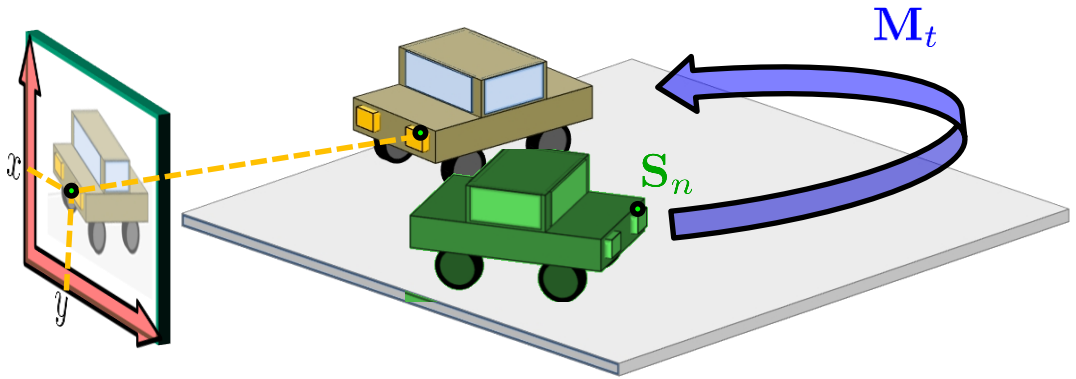
$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \end{bmatrix} \begin{bmatrix} \\ \\ \\ 1 \end{bmatrix}$$

- Projective camera $\mathbf{x}_{t,k,n} =$

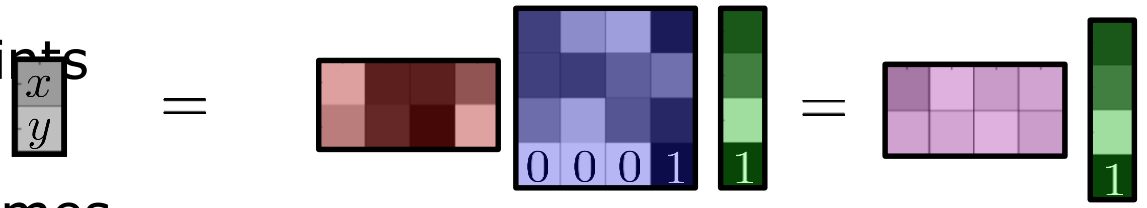
$$\begin{bmatrix} x \\ y \\ \lambda \end{bmatrix} = \mathbf{C}_k \quad \mathbf{M}_t \quad \mathbf{S}_n$$

$$= \begin{bmatrix} & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} & & & \\ & & & \\ & & & \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \\ \\ \\ 1 \end{bmatrix}$$

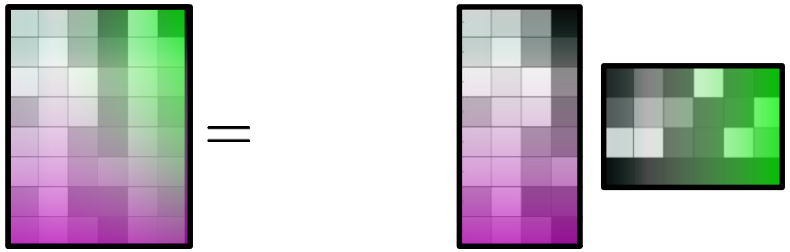
Rank-4 Factorization [Tomasi & Kanade]



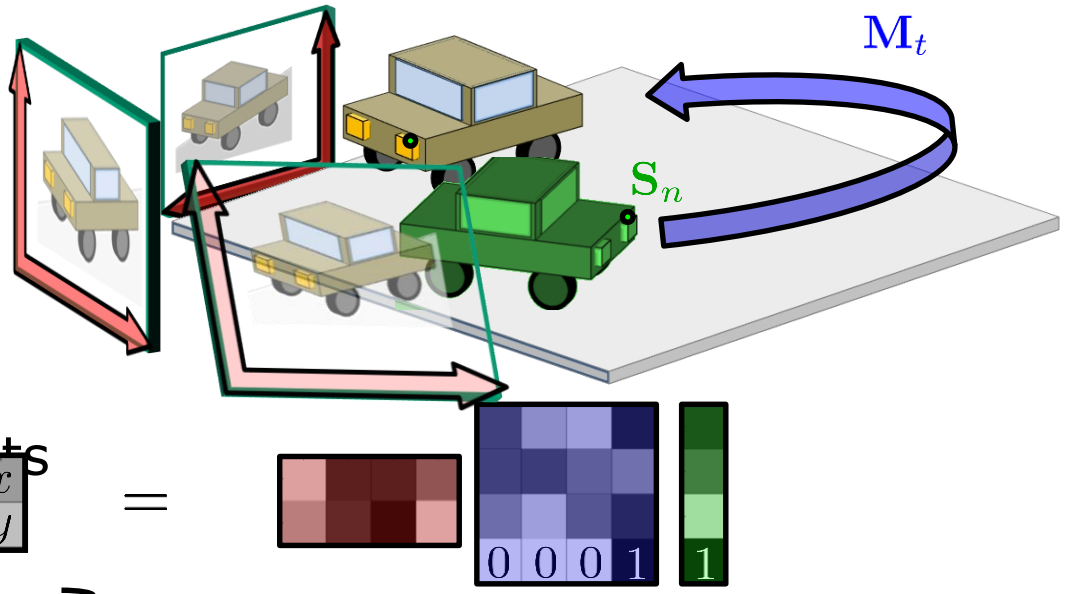
- Additional points



- Additional frames



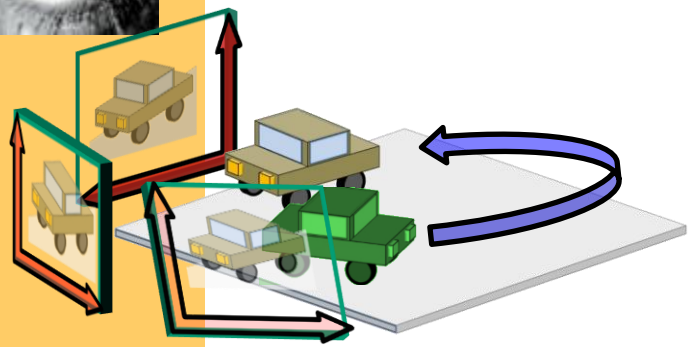
Motivation

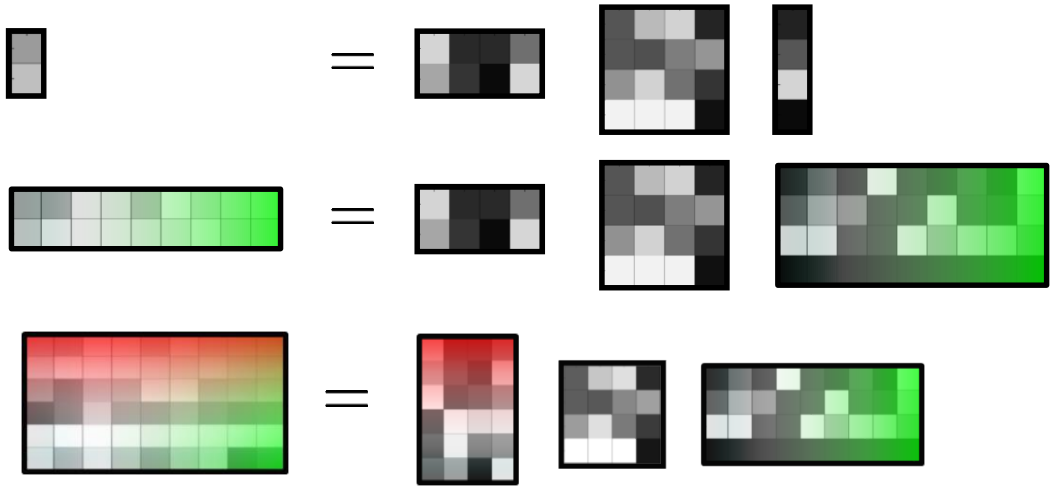


- Additional points
- Additional frames
- Additional cameras

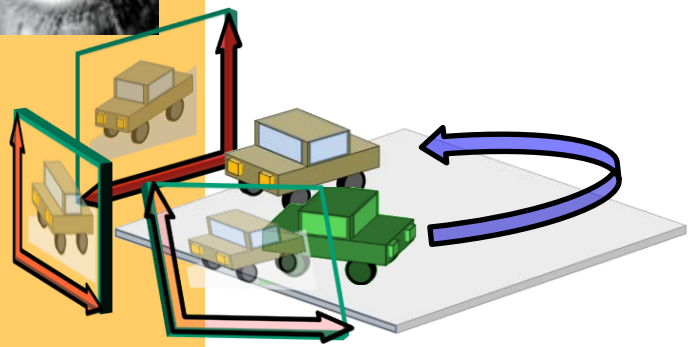
- Results in trilinear combination of structure, motion and camera poses
  - Models camera networks
  - Also applies to moving camera rig

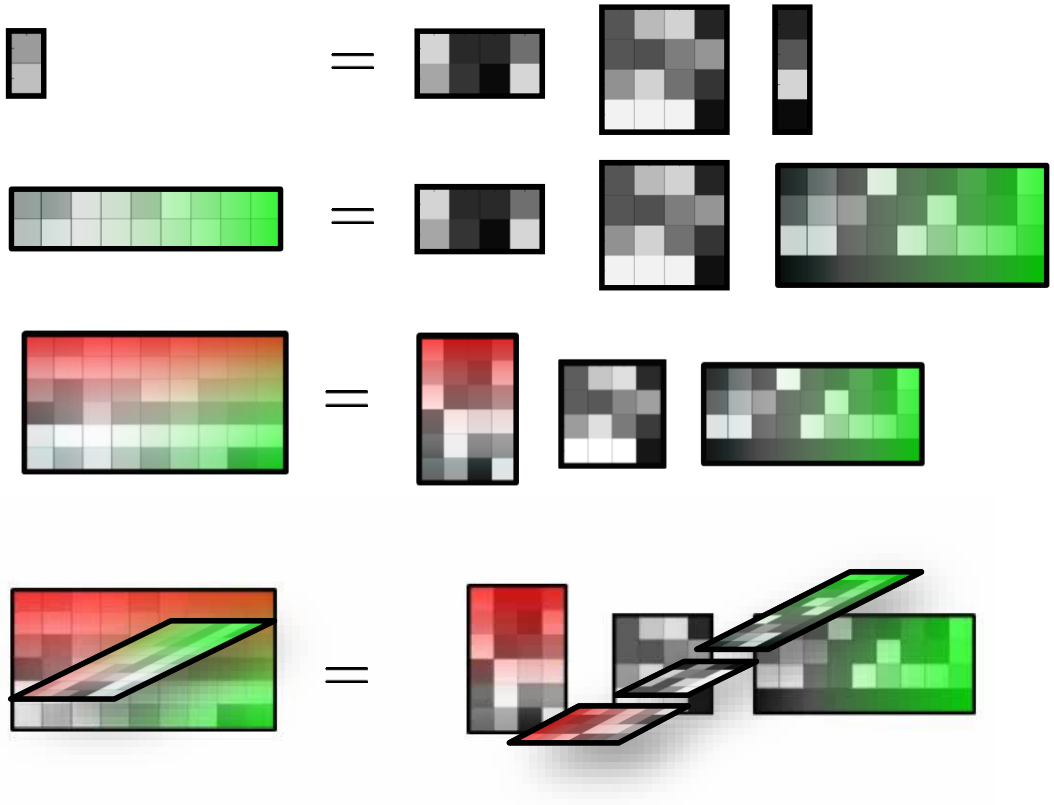
Affine Static Multi-Camera SfM



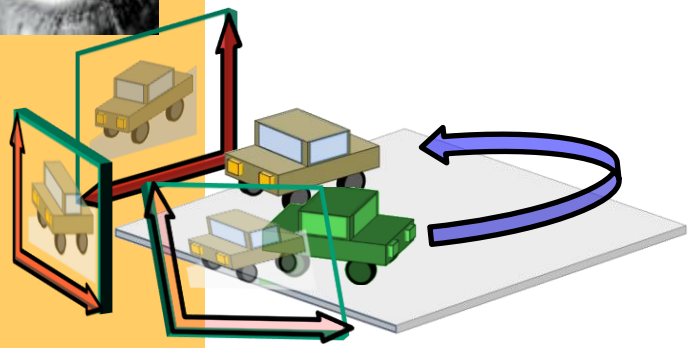


Affine Static Multi-Camera SfM

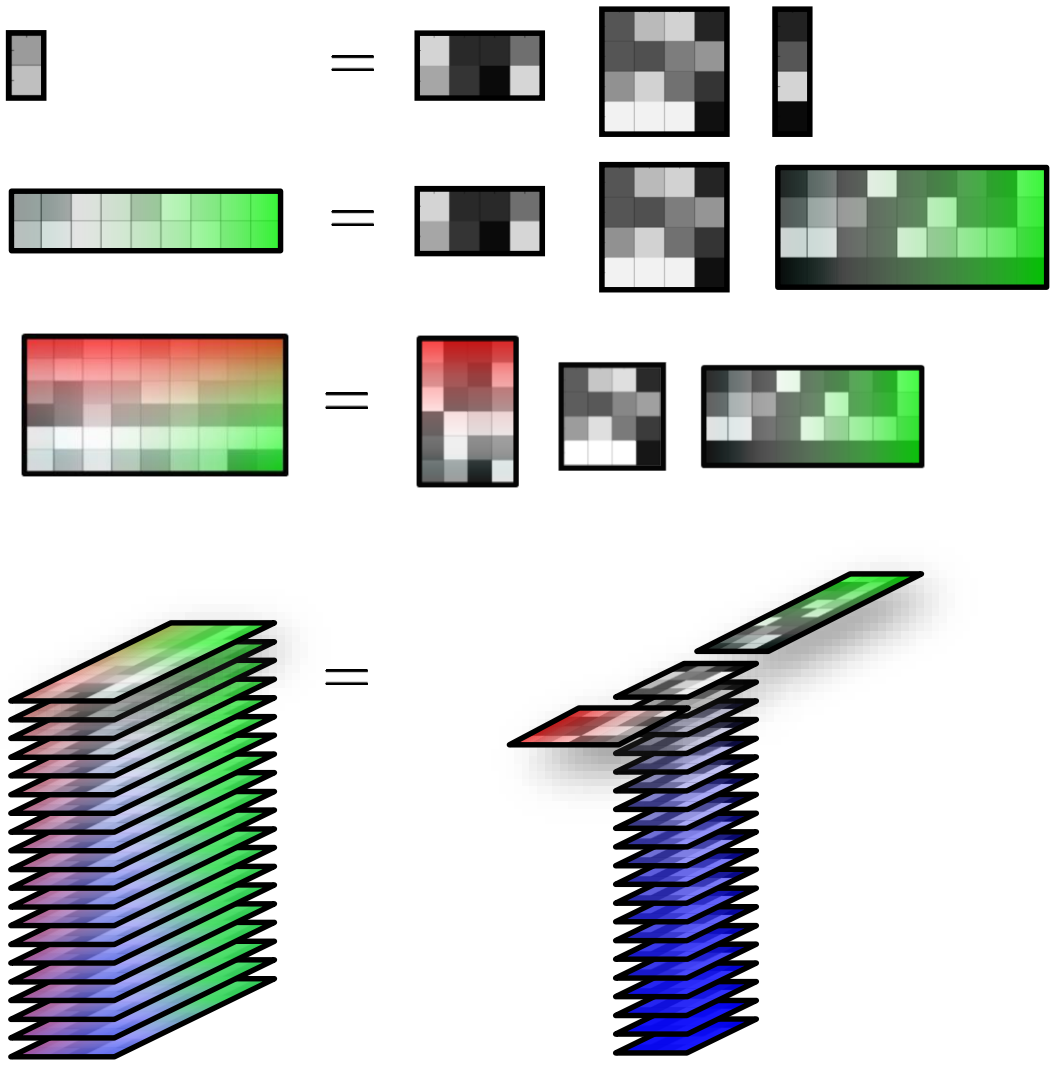




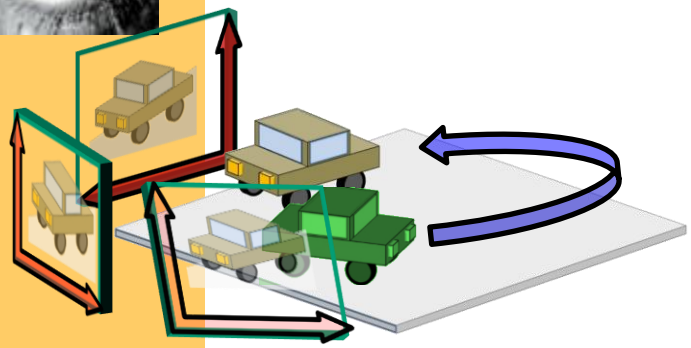
Affine Static Multi-Camera SfM



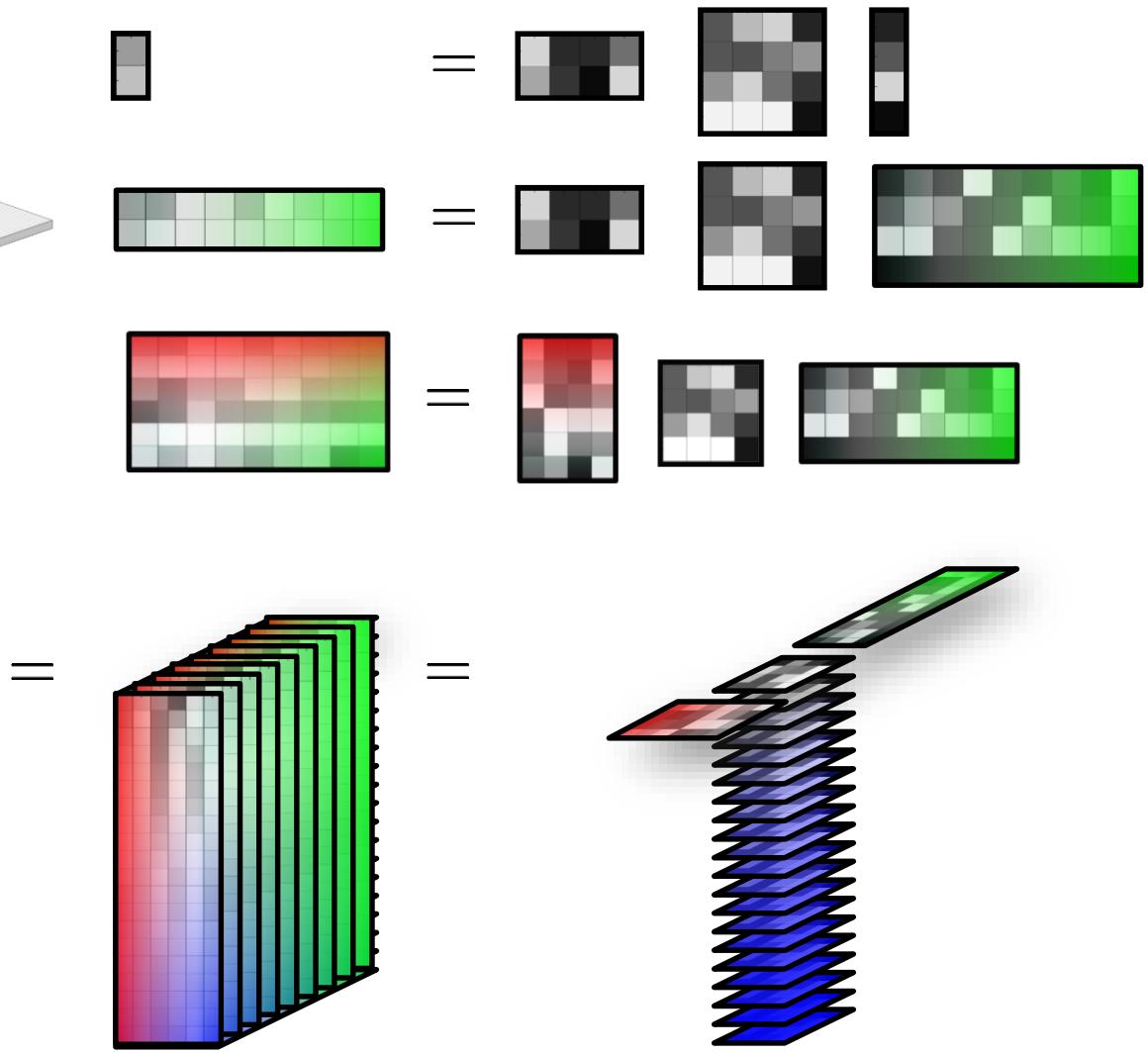
- Natural representation is a 3rd-order tensor



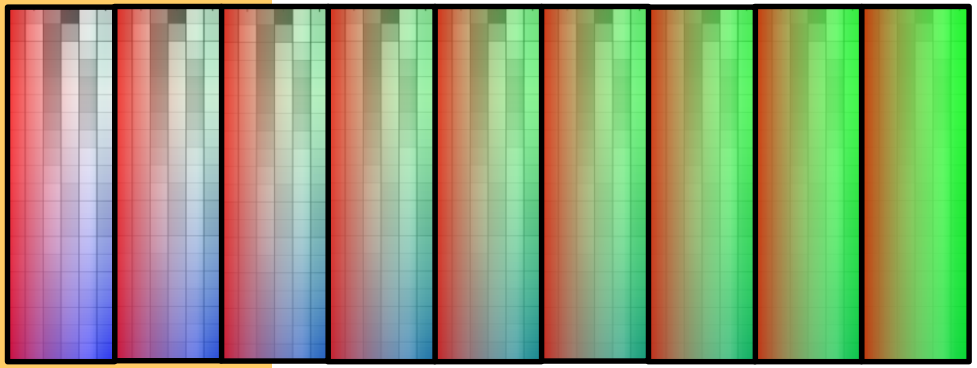
Affine Static Multi-Camera SfM



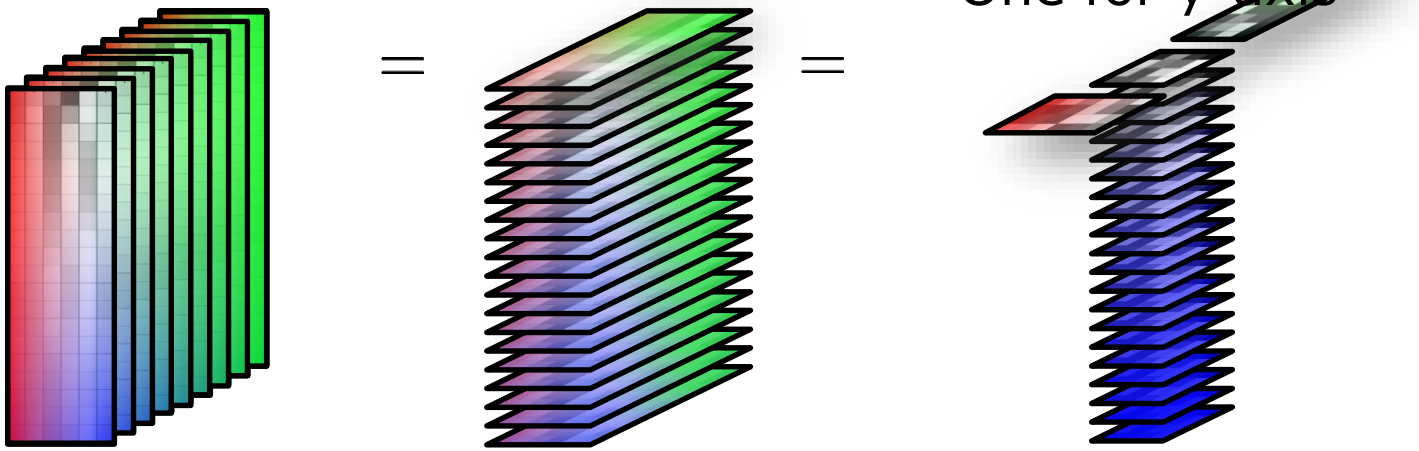
- Natural representation is a 3rd-order tensor



Flattening the Tensor: Tensor as a Matrix

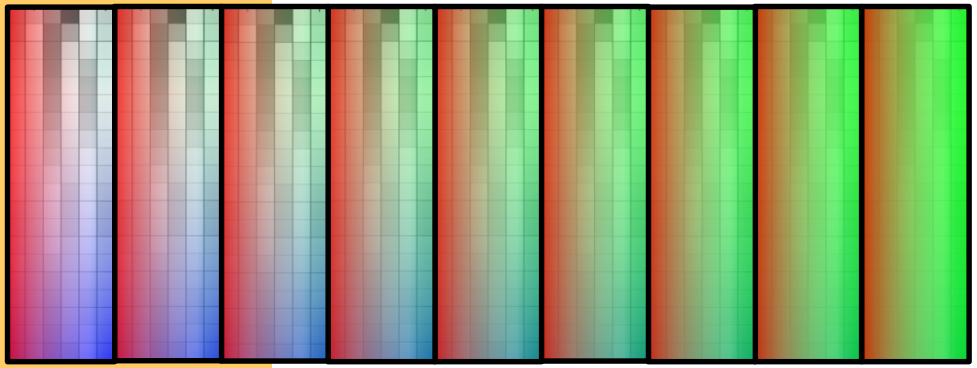


- Each column is a feature trajectory
- Each camera provides 2 trajectories per point
  - One for x-axis
  - One for y-axis



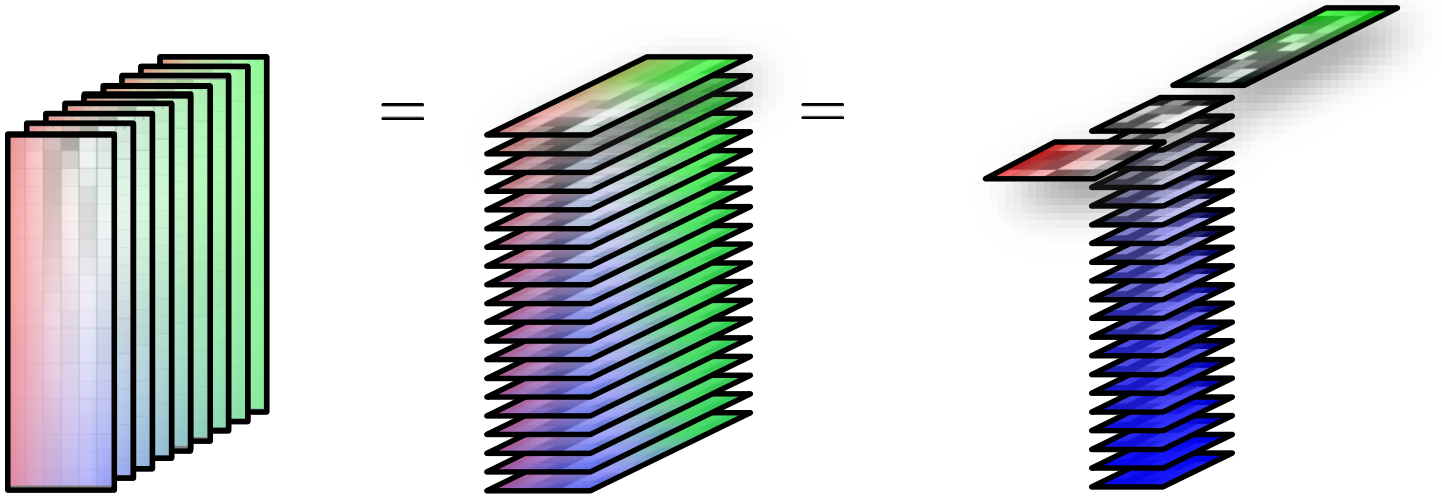
Main Theoretical Insight

(Angst and Pollefeys ICCV09/ECCV10)



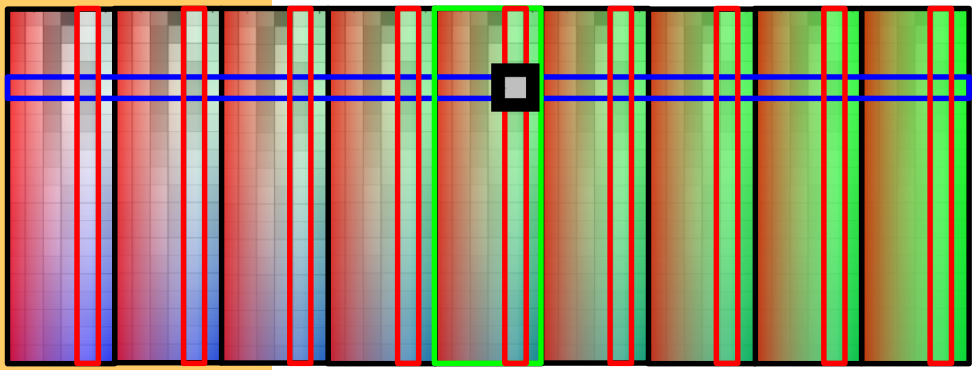
- For general motions
  - Data matrix has rank 13
- For planar motions
  - Data matrix has rank 5
- The trajectory
  - of any feature point
  - seen by any camera axis

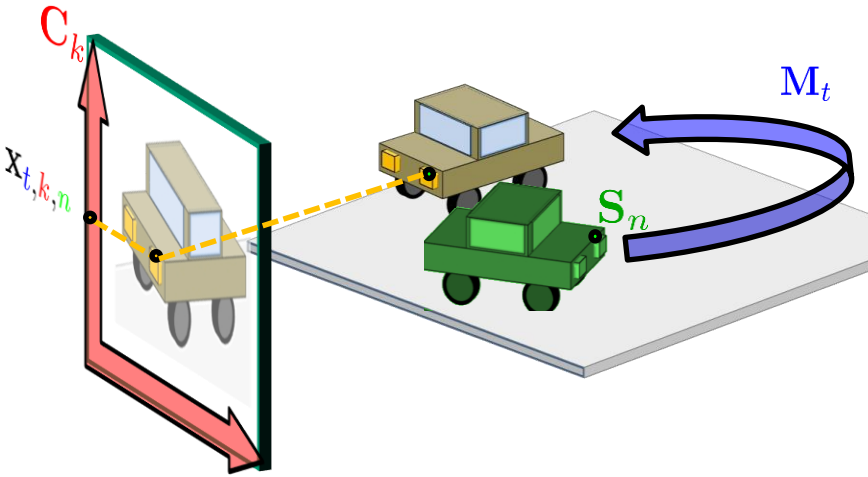
lies in a 13/5D subspace

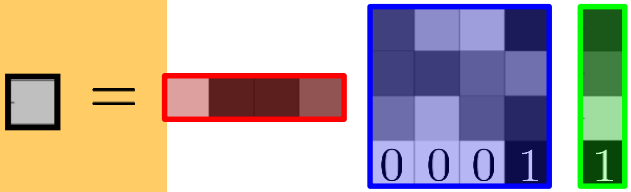


Low-Rank Matrix

$$\mathbf{W} = [\, \Downarrow_t \Rightarrow_k \Rightarrow_n \mathbf{x}_{t,k,n}]$$







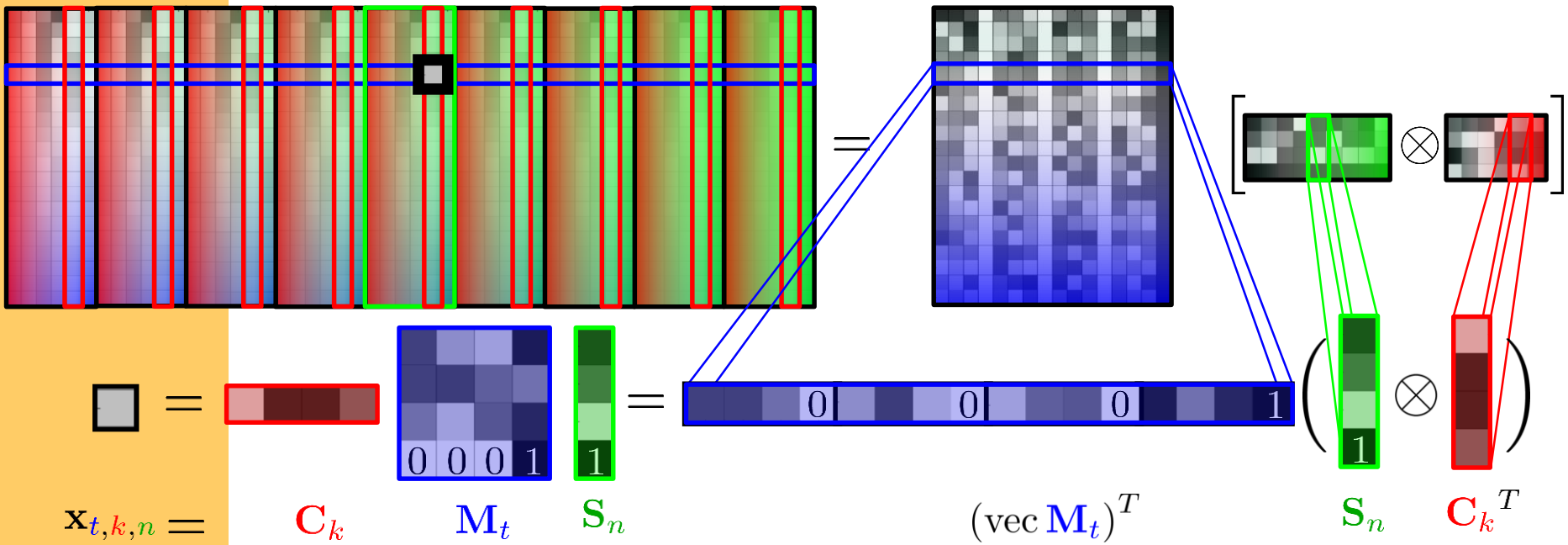
$\mathbf{x}_{t,k,n} =$ $\mathbf{C}_k$ $\mathbf{M}_t$ $\mathbf{S}_n$

- Entry of 3rd order tensor indexed by three indices
  - feature point $n \in \{1, \ldots, N\}$
  - frame $t \in \{1, \ldots, F\}$
  - camera axis $k \in \{1, \ldots, 2K\}$

Low-Rank Matrix: General Rigid Motion

$$\mathbf{W} = [\Downarrow_t \Rightarrow_k \Rightarrow_n \mathbf{x}_{t,k,n}]$$

$$= [\Downarrow_t \Rightarrow_{n,k} (\mathrm{vec}(\mathbf{M}_t))^T (\mathbf{S}_n \otimes \mathbf{C}_k^T)] = [\Downarrow_t (\mathrm{vec}\,\mathbf{M}_t)^T] \; [[\Rightarrow_n \mathbf{S}_n] \otimes [\Rightarrow_k \mathbf{C}_k^T]]$$



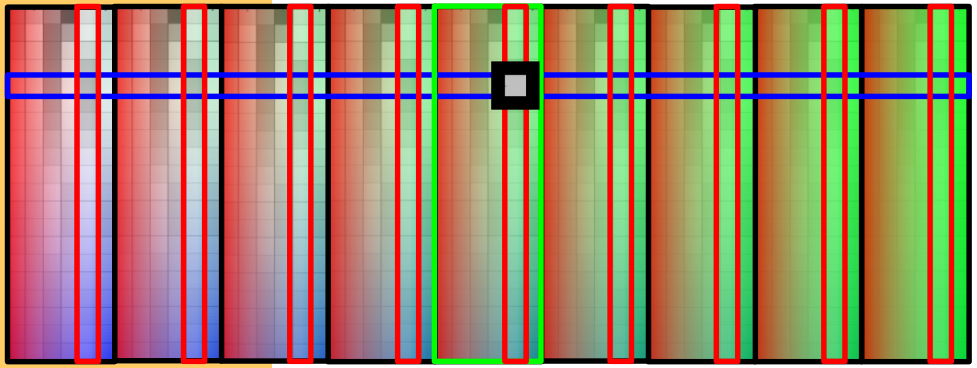
$$\mathbf{x}_{t,k,n} = \quad \mathbf{C}_k \quad \mathbf{M}_t \quad \mathbf{S}_n \quad (\mathrm{vec}\,\mathbf{M}_t)^T \quad \mathbf{S}_n \quad \mathbf{C}_k^T$$

- Use Kronecker-product property
- 13 non-zero columns
  - Rank-13 matrix factorization for general rigid motions

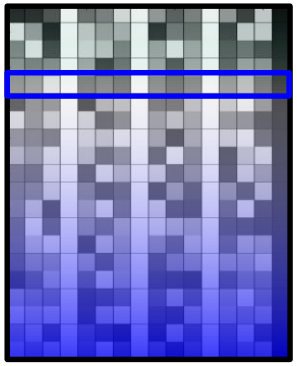
Low-Rank Matrix: Planar Rigid Motion

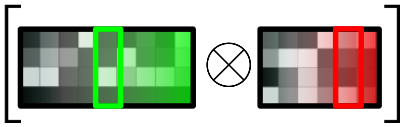
$$\mathbf{W} = [\Downarrow_t \Rightarrow_k \Rightarrow_n \mathbf{x}_{t,k,n}]$$

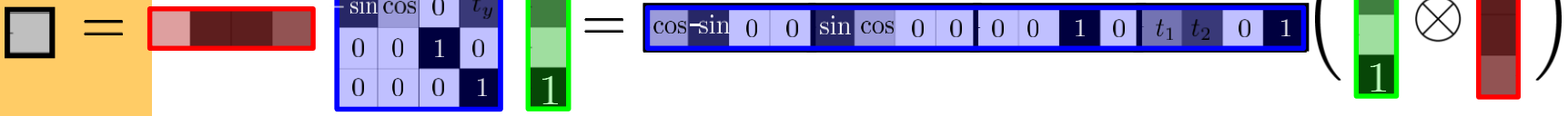
$$[\Downarrow_t \Rightarrow_{n,k} (\mathrm{vec}(\mathbf{M}_t))^T (\mathbf{S}_n \otimes \mathbf{C}_k^T)] = [\Downarrow_t (\mathrm{vec}\,\mathbf{M}_t)^T] \; [[\Rightarrow_n \mathbf{S}_n] \otimes [\Rightarrow_k \mathbf{C}_k^T]]$$



=

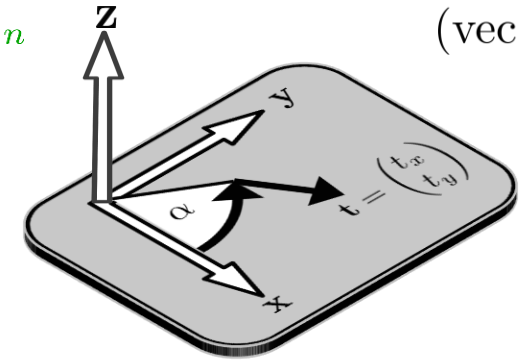






$$\mathbf{x}_{t,k,n} = \quad \mathbf{C}_k \quad \mathbf{M}_t \quad \mathbf{S}_n \quad \mathbf{z} \quad (\mathrm{vec}\,\mathbf{M}_t)^T \quad \mathbf{S}_n \quad \mathbf{C}_k{}^T$$

- Planar rigid motion
  - Rotation angle α
  - Translation $\mathbf{t} = (t_x, t_y)^T$



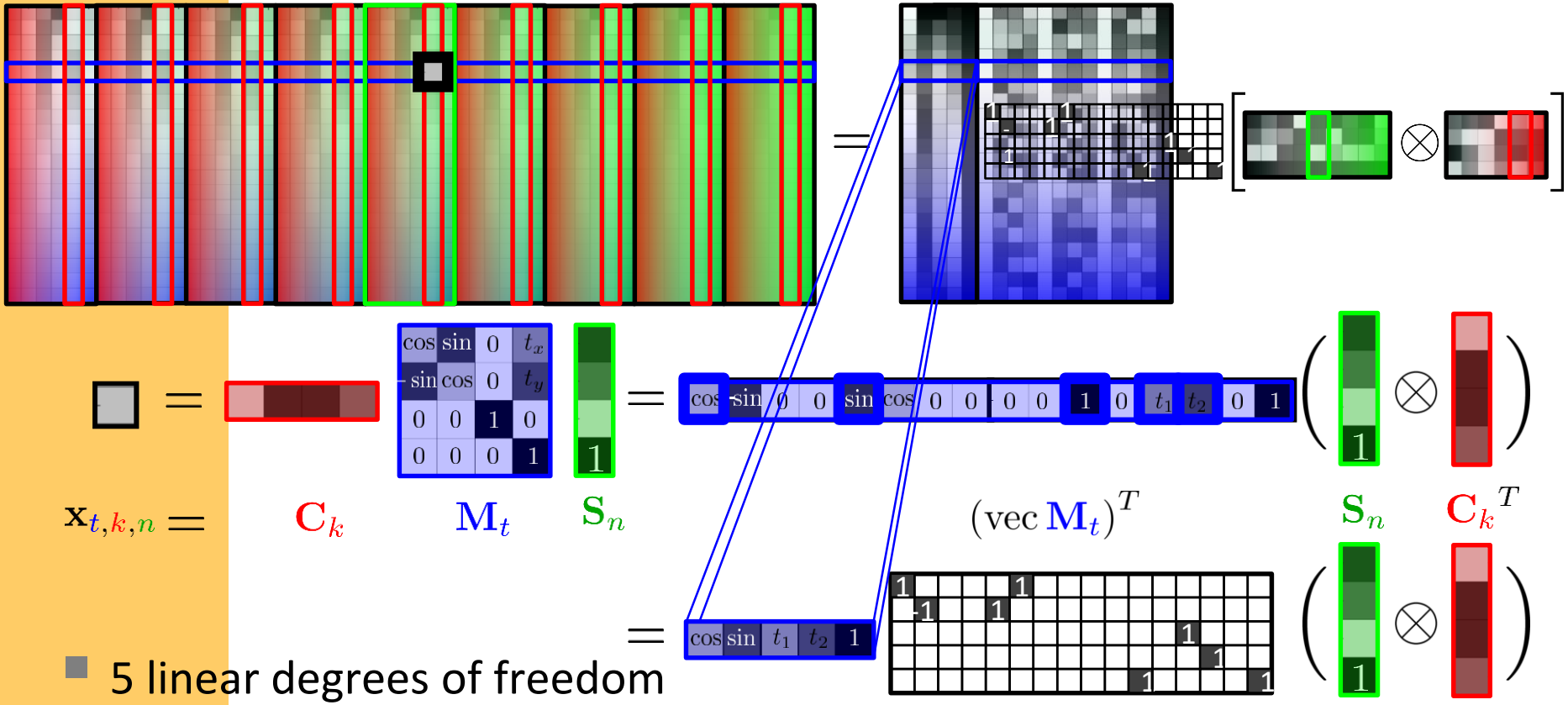
$\Leftrightarrow$

$$\begin{bmatrix} \cos\alpha & \sin\alpha & 0 & t_x \\ -\sin\alpha & \cos\alpha & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Low-Rank Matrix: Planar Rigid Motion

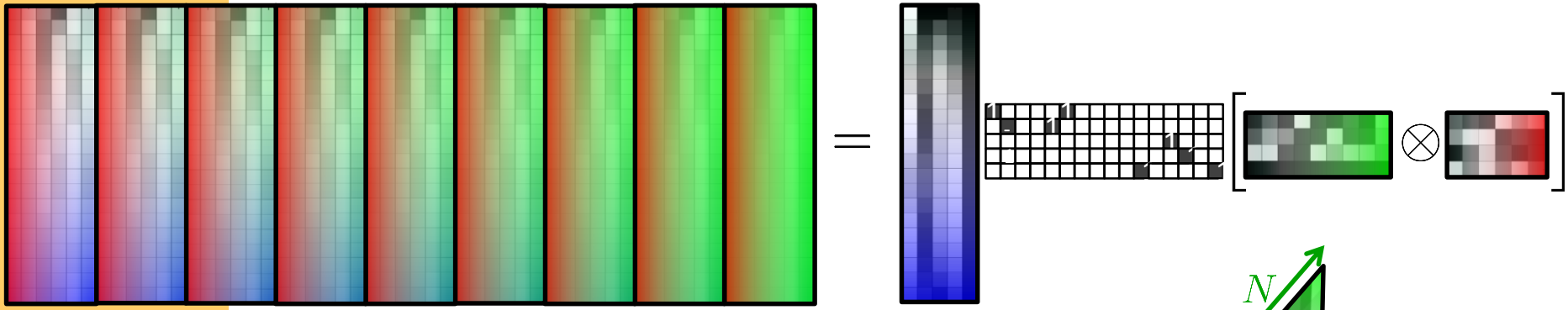
$$\mathbf{W} = [\Downarrow_t \Rightarrow_k \Rightarrow_n \mathbf{x}_{t,k,n}]$$

$$[\Downarrow_t \Rightarrow_{n,k} (\mathrm{vec}(\mathbf{M}_t))^T (\mathbf{S}_n \otimes \mathbf{C}_k^T)] = [\Downarrow_t (\mathrm{vec}\,\mathbf{M}_t)^T] \; [[\Rightarrow_n \mathbf{S}_n] \otimes [\Rightarrow_k \mathbf{C}_k^T]]$$

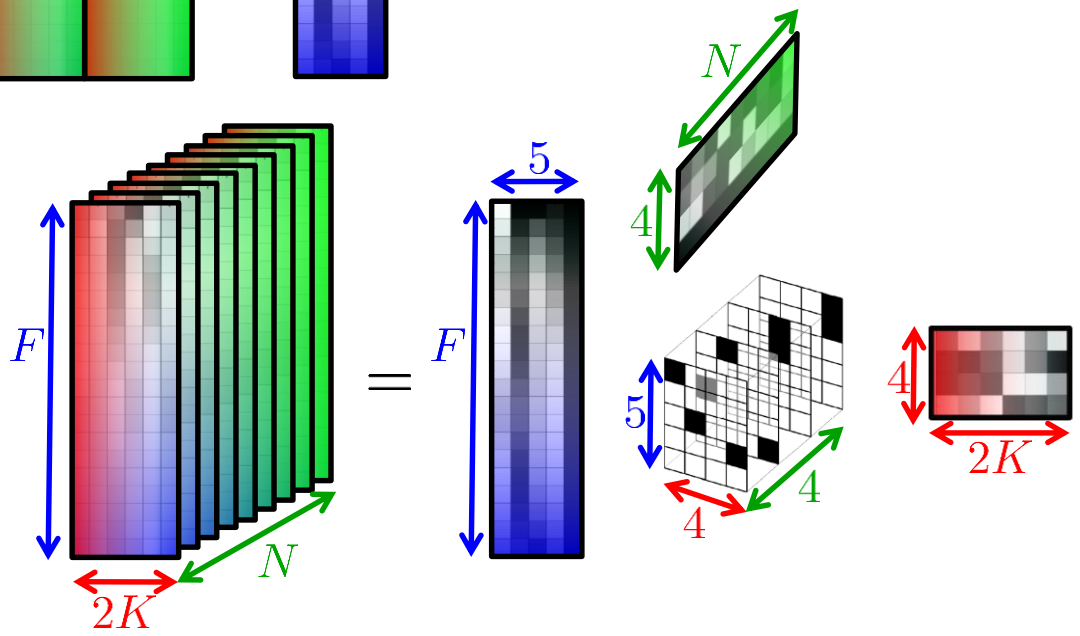


- 5 linear degrees of freedom
  - Rank-5 matrix factorization

Tucker Tensor Decomposition

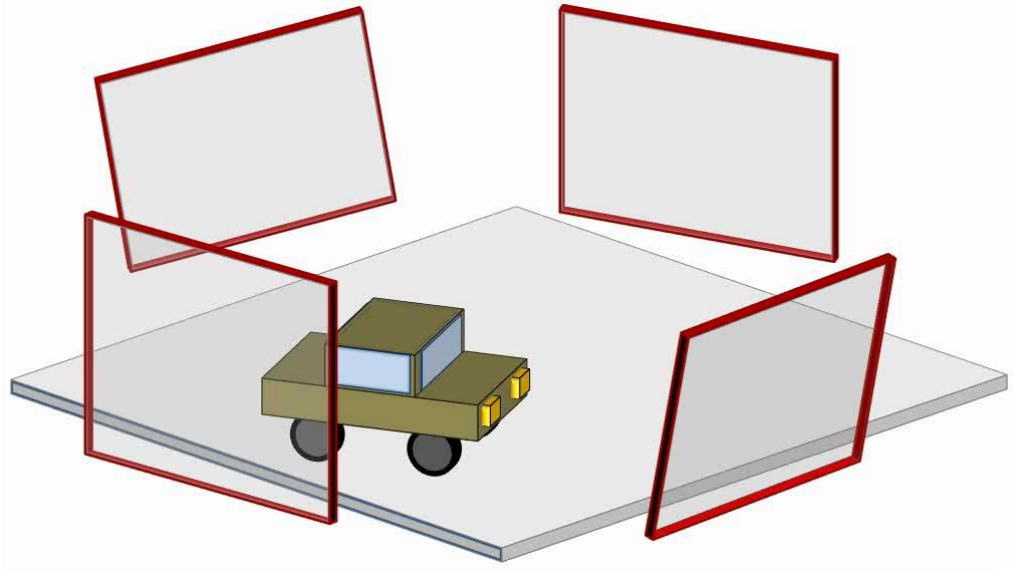


- Specific algebraic structure
- Tucker tensor decomposition
  - Core tensor (known)
  - 3 matrices
    - Motion
    - Structure
    - Cameras

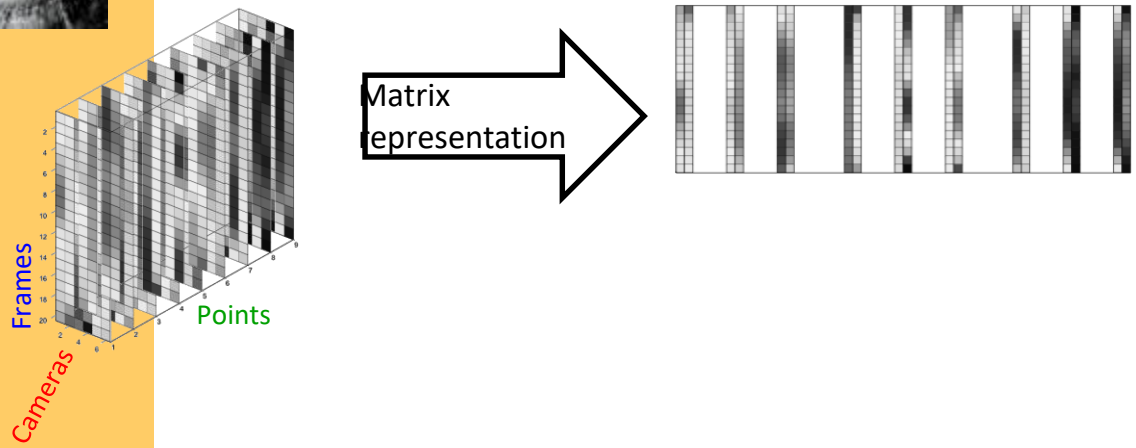


Problem Setup

- Multiple static and affine cameras observe planar motion of a rigid object
  - Translations parallel to plane
  - Rotation axes orthogonal to plane
- 2D feature point trajectories as input
  - Complete feature trajectories
  - No feature point correspondences between different cameras
- We strive for a closed-form solution



Algorithm

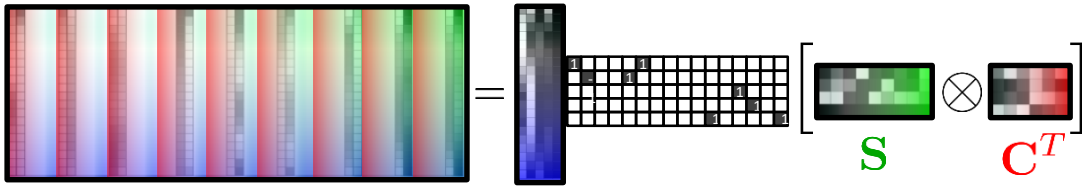


- Missing feature point correspondences
  - Lead to missing entries in data tensor
  - Columns either completely known or unknown

Algorithm (planar motion)

Rank-5 Factorization

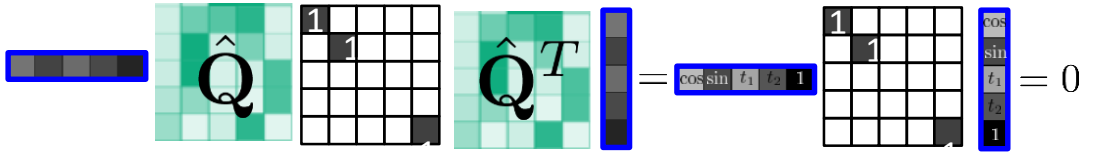
- Use only known columns
- Resulting matrix has still rank 5
- Fuses data from all cameras
- Even works for just 2 feature points



$$\mathbf{M} = \hat{\mathbf{M}}\hat{\mathbf{Q}}\tilde{\mathbf{Q}}^{-1}\mathbf{A} = \tilde{\mathbf{Q}}\hat{\mathbf{Q}}^{-1}\hat{\mathbf{A}}$$

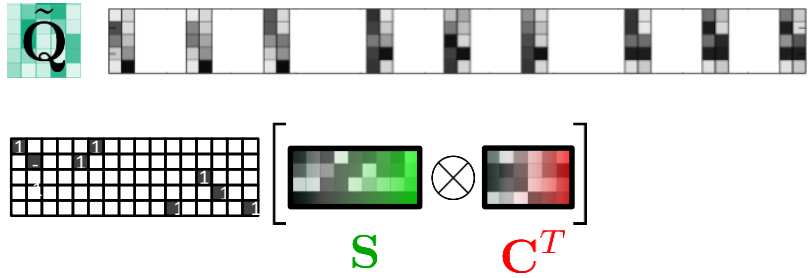
Sine/Cosine structure

- One constraint per frame
- Linear system in symmetric 5-by-5 matrix
- Non-linear rank constraint



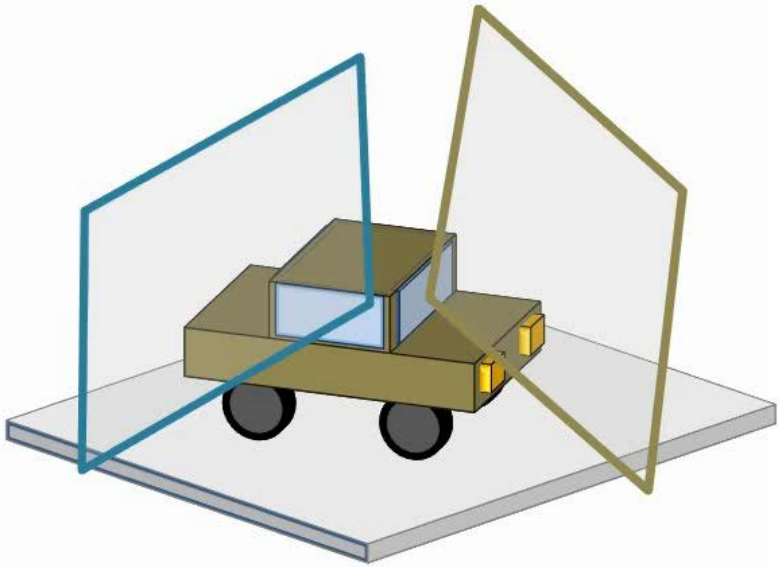
Kronecker structure

- Linear in 5-by-5 transformation matrix
- Bilinear in points and cameras
- Use Gauge-freedoms in plane of motion
  - Fix one camera: linear in its feature points
  - Register all cameras in common frame



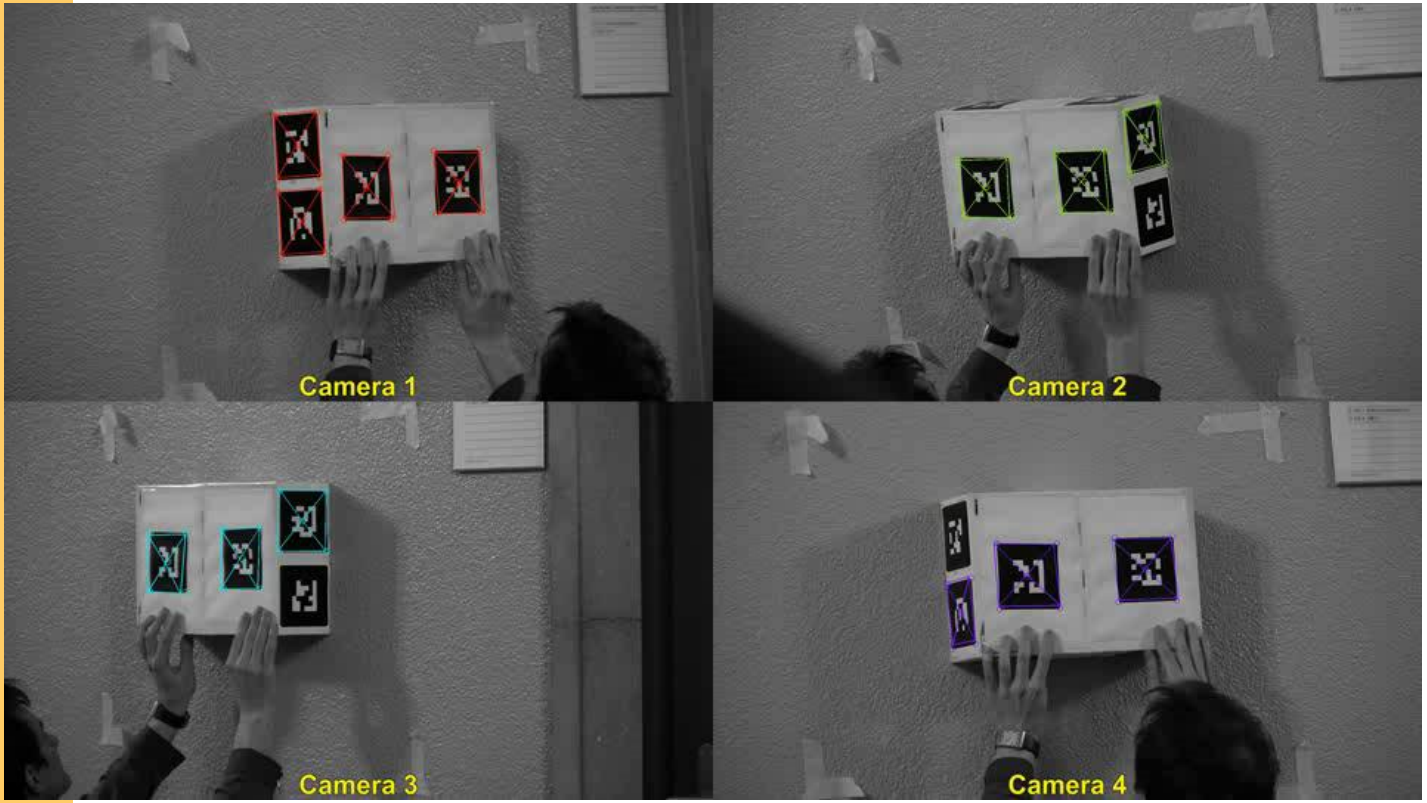
Ambiguities

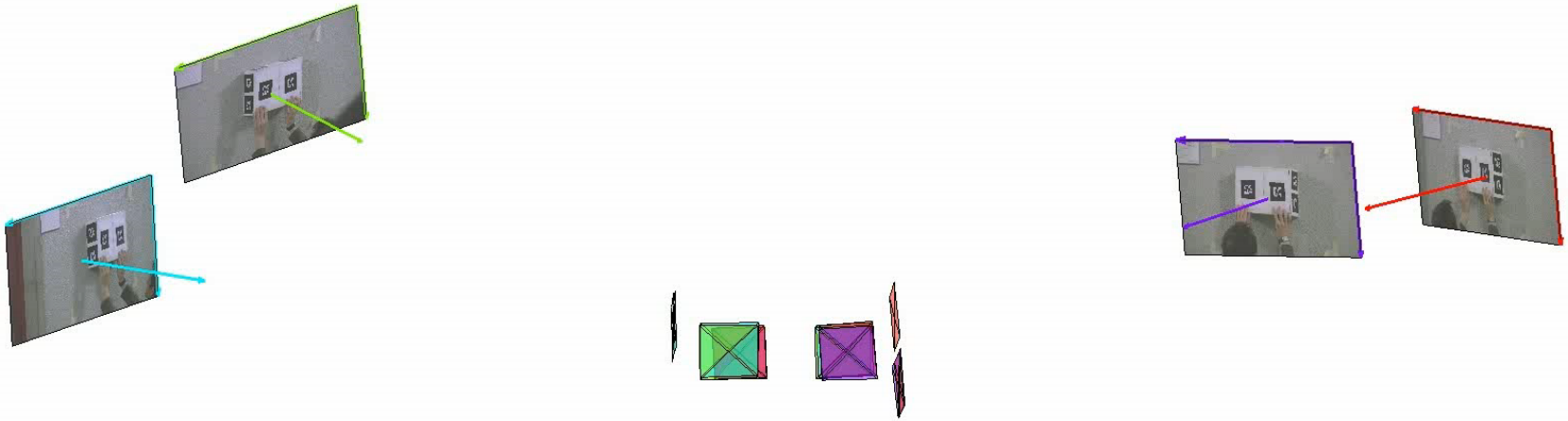
- Per camera scaling along rotation axis
  - Orthogonality & equality of norm constraints on x- & y-axis of same c
    - Assumption of perfectly square pixels
- Per camera translation along rotation axis
  - Not resolvable without feature point correspondences between different cameras
  - Heuristic: align centroids along axis of rotation



Results

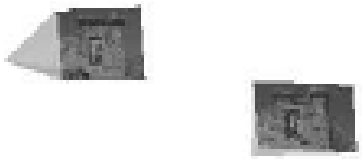
- Data set
  - 4 temporally synchronized cameras each tracking templates
  - No correspondences between different cameras established



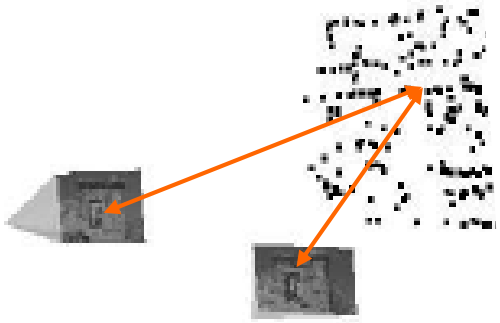


Completed frames indicated with black camera plane normals

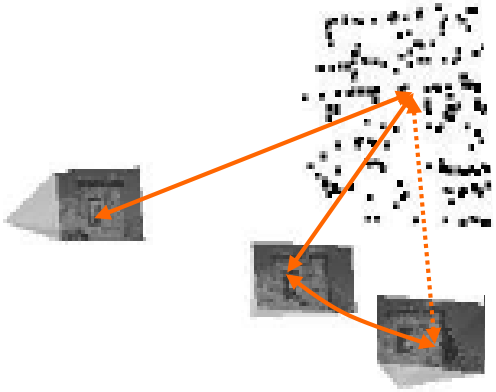
Sequential Structure and Motion Computation



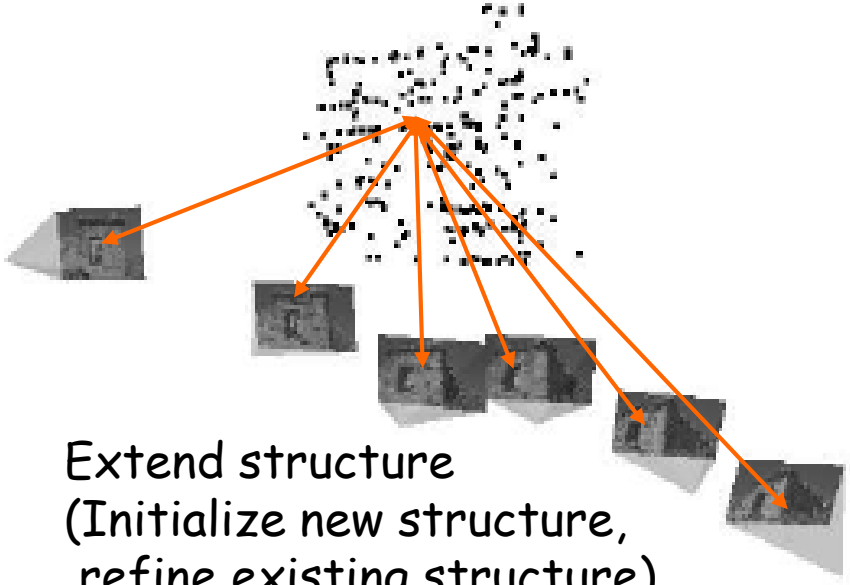
Initialize Motion
(P_1,P_2 compatibel with F)



Initialize Structure
(minimize reprojection error)



Extend motion
(compute pose through matches
seen in 2 or more previous views)



Extend structure
(Initialize new structure,
refine existing structure)

Refining structure and motion

- Minimize reprojection error

$$\min_{\hat{P}_k, \hat{M}_i} \sum_{k=1}^{m} \sum_{i=1}^{n} D\left(m_{ki}, \hat{P}_k \hat{M}_i\right)^2$$

  - Maximum Likelyhood Estimation
    (if error zero-mean Gaussian noise)
  - Huge problem but can be solved efficiently
    (Bundle adjustment)

Non-linear least-squares

$$\mathrm{X} = f(\mathrm{P}) \qquad \underset{\mathrm{P}}{\operatorname{argmin}} \left\| \mathrm{X} - f(\mathrm{P}) \right\|$$

- Newton iteration
- Levenberg-Marquardt
- Sparse Levenberg-Marquardt

Newton iteration

Taylor approximation

$$f(\mathrm{P}_0 + \Delta) \approx f(\mathrm{P}_0) + \mathrm{J}\Delta$$

Jacobian

$$\mathrm{J} = \frac{\partial \mathrm{X}}{\partial \mathrm{P}}$$

$$\left\| \mathrm{X} - f(\mathrm{P}_1) \right\|$$

$$\left\| \mathrm{X} - f(\mathrm{P}_1) \right\| \approx \left\| \mathrm{X} - f(\mathrm{P}_0) - \mathrm{J}\Delta \right\| = \left\| e_0 - \mathrm{J}\Delta \right\|$$

$$\Rightarrow \mathrm{J}^{\mathrm{T}}\mathrm{J}\Delta = \mathrm{J}^{\mathrm{T}} e_0 \Rightarrow \Delta = \left(\mathrm{J}^{\mathrm{T}}\mathrm{J}\right)^{-1}\mathrm{J}^{\mathrm{T}} e_0$$

$$\mathrm{P}_{i+1} = \mathrm{P}_i + \Delta$$

$$\Delta = \left(\mathrm{J}^{\mathrm{T}}\mathrm{J}\right)^{-1}\mathrm{J}^{\mathrm{T}} e_0$$

normal eq.

$$\Delta = \left(\mathrm{J}^{\mathrm{T}}\Sigma^{-1}\mathrm{J}\right)^{-1}\mathrm{J}^{\mathrm{T}}\Sigma^{-1} e_0$$

Levenberg-Marquardt

Normal equations

$$\mathrm{J}^{\mathrm{T}}\mathrm{J}\Delta = \mathrm{N}\Delta = \mathrm{J}^{\mathrm{T}}e_0$$

Augmented normal equations

$$\mathrm{N}'\Delta = \mathrm{J}^{\mathrm{T}}e_0 \qquad \mathrm{N}' = \mathrm{J}^{\mathrm{T}}\mathrm{J} + \lambda\,\mathrm{diag}(\mathrm{J}^{\mathrm{T}}\mathrm{J})$$

$$\lambda_0 = 10^{-3}$$

$$\text{success} : \lambda_{i+1} = \lambda_i / 10 \quad \text{accept}$$

$$\text{failure} : \lambda_i = 10\lambda_i \quad \text{solve again}$$

λ small ~ Newton (quadratic convergence)

λ large ~ descent (guaranteed decrease)

Levenberg-Marquardt

Requirements for minimization

- Function to compute f
- Start value P_0
- Optionally, function to compute J
  (but numerical ok, too)

Bundle Adjustment

$$\begin{aligned} \hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \end{aligned}$$

- What makes this non-linear minimization hard?
  - many more parameters: potentially slow
  - poorer conditioning (high correlation)
  - potentially lots of outliers
  - gauge (coordinate) freedom

6.2.2 Iterative algorithms

The most accurate (and flexible) way to estimate pose is to directly minimize the squared (or robust) reprojection error for the 2D points as a function of the unknown pose parameters in $(\boldsymbol{R}, \boldsymbol{t})$ and optionally $\boldsymbol{K}$ using non-linear least squares (Tsai 1987; Bogart 1991; Gleicher and Witkin 1992). We can write the projection equations as

$$\boldsymbol{x}_i = \boldsymbol{f}(\boldsymbol{p}_i; \boldsymbol{R}, \boldsymbol{t}, \boldsymbol{K}) \tag{6.41}$$

and iteratively minimize the robustified linearized reprojection errors

$$E_{\mathrm{NLP}} = \sum_i \rho \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{R}} \Delta \boldsymbol{R} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{t}} \Delta \boldsymbol{t} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{K}} \Delta \boldsymbol{K} - \boldsymbol{r}_i \right), \tag{6.42}$$

where $\boldsymbol{r}_i = \tilde{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_i$ is the current residual vector (2D error in predicted position), and the partial derivatives are with respect to the unknown pose parameters (rotation, translation, and optionally calibration). Note that if full 2D covariance estimates are available for the 2D feature locations, the above squared norm can be weighted by the inverse point covariance matrix, as in (6.11).

An easier to understand (and implement) version of the above non-linear regression problem can be constructed by re-writing the projection equations as a concatenation of simpler steps, each of which transforms a 4D homogeneous coordinate $\boldsymbol{p}_i$ by a simple transformation such as translation, rotation, or perspective division (Figure 6.5). The resulting projection

$$\begin{aligned} \boldsymbol{y}^{(1)} &= \boldsymbol{f}_{\mathrm{T}}(\boldsymbol{p}_i; \boldsymbol{c}_j) = \boldsymbol{p}_i - \boldsymbol{c}_j, && (6.43) \\ \boldsymbol{y}^{(2)} &= \boldsymbol{f}_{\mathrm{R}}(\boldsymbol{y}^{(1)}; \boldsymbol{q}_j) = \boldsymbol{R}(\boldsymbol{q}_j)\, \boldsymbol{y}^{(1)}, && (6.44) \\ \boldsymbol{y}^{(3)} &= \boldsymbol{f}_{\mathrm{P}}(\boldsymbol{y}^{(2)}) = \frac{\boldsymbol{y}^{(2)}}{z^{(2)}}, && (6.45) \\ \boldsymbol{x}_i &= \boldsymbol{f}_{\mathrm{C}}(\boldsymbol{y}^{(3)}; \boldsymbol{k}) = \boldsymbol{K}(\boldsymbol{k})\, \boldsymbol{y}^{(3)}. && (6.46) \end{aligned}$$

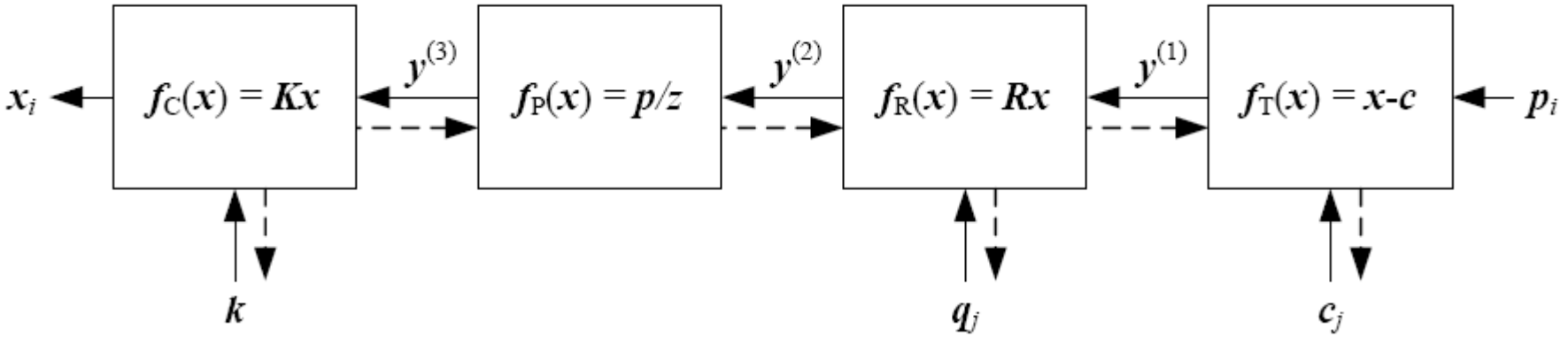
The advantage of this chained set of transformations is that each one has a simple partial derivative with respect to both its parameters and to its input. Thus, once the predicted value of $\tilde{\boldsymbol{x}}_i$ has been computed based on the 3D point location $\boldsymbol{p}_i$ and the current values of the pose parameters $(\boldsymbol{c}_j, \boldsymbol{q}_j, \boldsymbol{k})$, we can obtain all of the required partial derivatives using the chain rule

$$\frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{p}^{(k)}} = \frac{\partial \boldsymbol{r}_i}{\partial \boldsymbol{y}^{(k)}} \frac{\partial \boldsymbol{y}^{(k)}}{\partial \boldsymbol{p}^{(k)}}, \tag{6.47}$$

where $\boldsymbol{p}^{(k)}$ indicates in this case one of the parameter vectors that is being optimized. (This same "trick" is used in neural networks as part of the *backpropagation* algorithm (Bishop 2006).)

Chained transformations

- Think of the optimization as a set of transformations along with their derivatives



Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Linearize measurement equations

$$\hat{u}_i = f(\mathbf{m}, \mathbf{x}_i) + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m}$$

$$\hat{v}_i = g(\mathbf{m}, \mathbf{x}_i) + \frac{\partial g}{\partial \mathbf{m}} \Delta \mathbf{m}$$

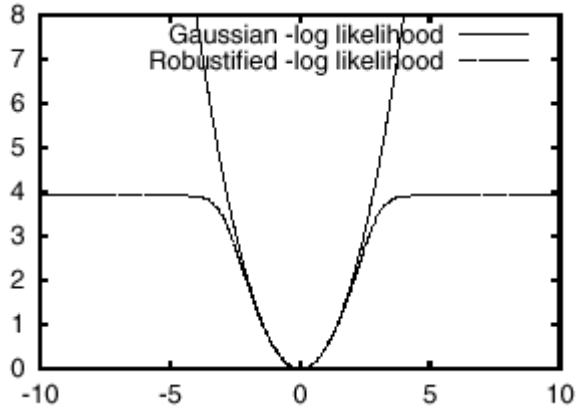
  - Substitute into log-likelihood equation: quadratic cost function in $\Delta\mathbf{m}$

$$\sum_i \sigma_i^{-2} (\hat{u}_i - u_i + \frac{\partial f}{\partial \mathbf{m}} \Delta \mathbf{m})^2 + \cdots$$

Robust error models

- Outlier rejection
  - use robust penalty applied to each set of joint measurements

$$\sum_i \sigma_i^{-2} \rho\left(\sqrt{(u_i - \hat{u}_i)^2 + (v_i - \hat{v}_i)^2}\right)$$



  - for extremely bad data, use random sampling [RANSAC, Fischler & Bolles, CACM'81]

Levenberg-Marquardt

- Iterative non-linear least squares [Press'92]
  - Solve for minimum $\frac{\partial C}{\partial \mathbf{m}} = 0$

$$\mathbf{A}\Delta\mathbf{m} = \mathbf{b}$$

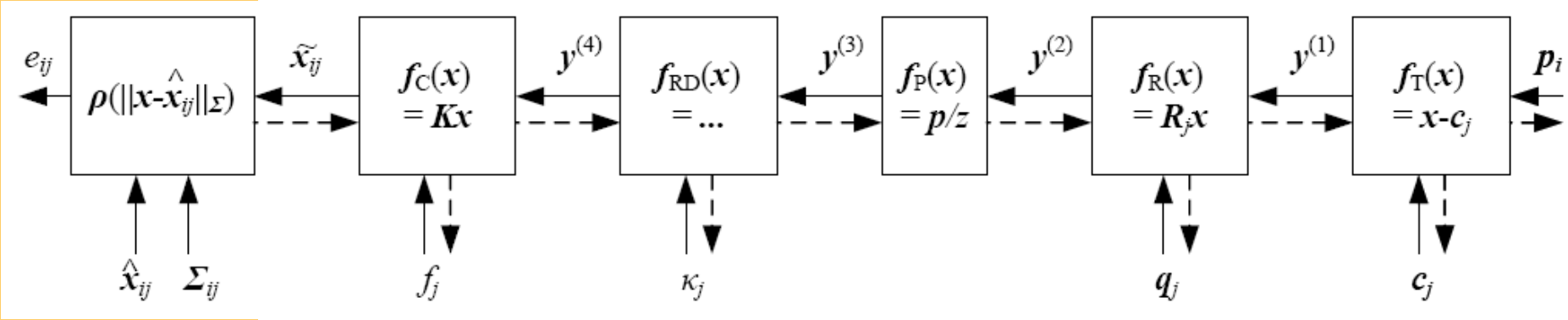
Hessian: $$\mathbf{A} = \left[\sum_i \sigma_i^{-2} \frac{\partial f}{\partial \mathbf{m}} \left(\frac{\partial f}{\partial \mathbf{m}}\right)^T + \cdots\right]$$

error: $$\mathbf{b} = \left[\sum_i \sigma_i^{-2} \frac{\partial f}{\partial \mathbf{m}} (u_i - \hat{u}_i) + \cdots\right]$$

Chained transformations

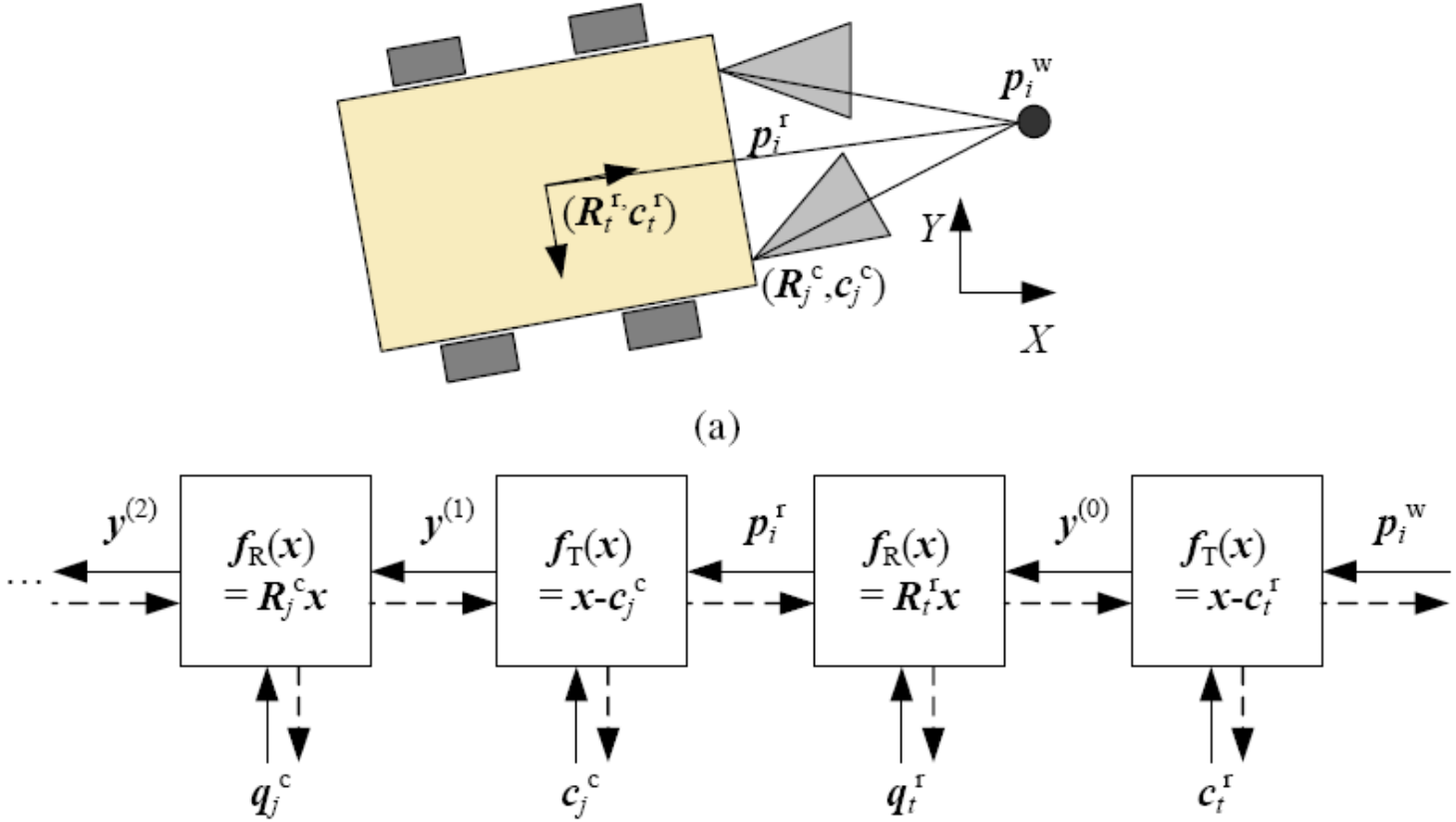
- Only some derivatives are non-zero:

  j^{th} camera, i^{th} point



Multi-camera rig

- Easy extension of generalized bundler:



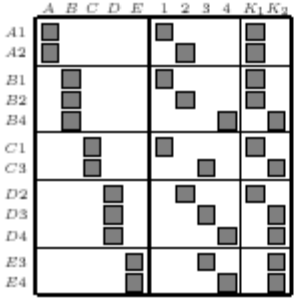
Lots of parameters: sparsity

$$\begin{aligned}\hat{u}_{ij} &= f(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i) \\ \hat{v}_{ij} &= g(\mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \mathbf{x}_i)\end{aligned}$$

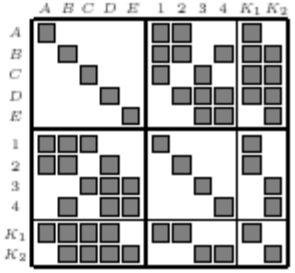
- Only a few entries in Jacobian are non-zero

$$\frac{\partial \hat{u}_{ij}}{\partial \mathbf{K}}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{R}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{t}_j}, \quad \frac{\partial \hat{u}_{ij}}{\partial \mathbf{x}_i},$$

J =



H =



Exploiting sparsity

- *Schur complement* to get *reduced camera*

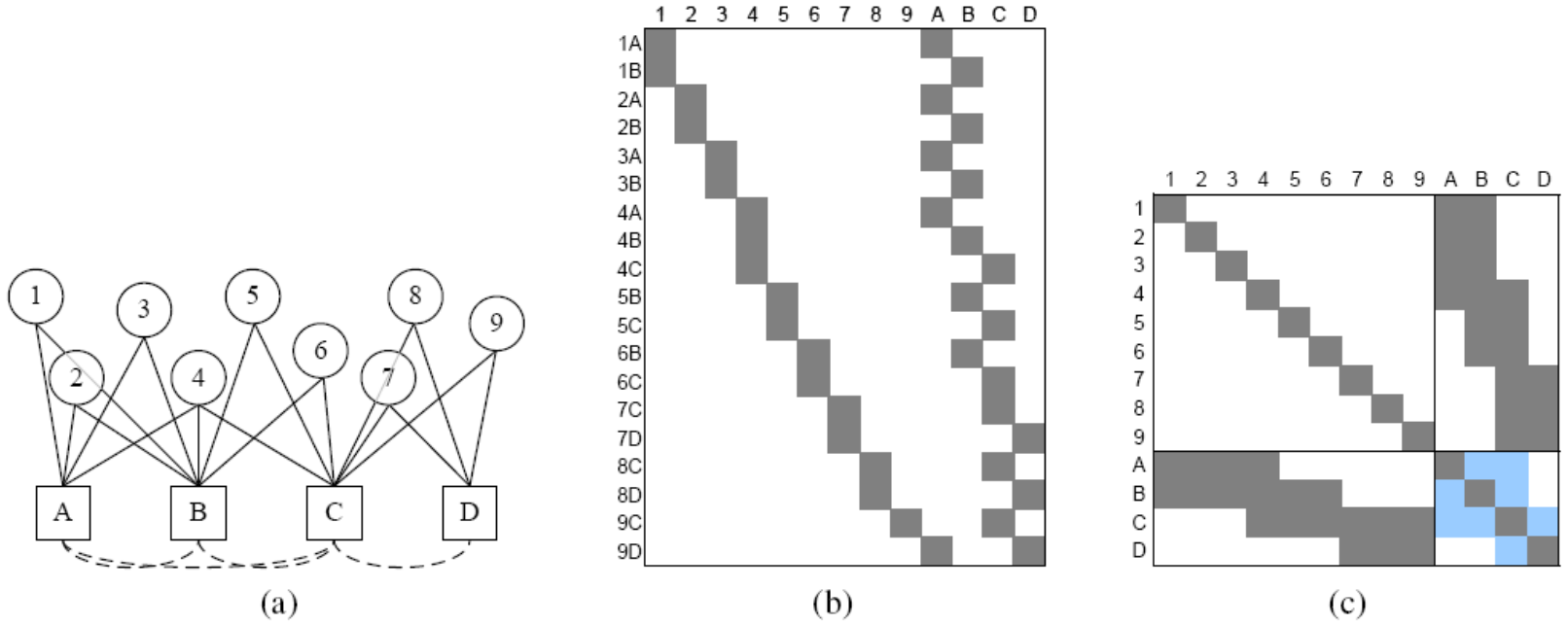


Figure 7.9 Bipartite graph for a toy structure from motion problem (a) and its associated Jacobian J (b) and Hessian A (c). Numbers (1–9) indicate 3D points while letters (A–D) indicate cameras. The dashed arcs and light blue squares indicate the fill-in that occurs when the structure (point) variables are eliminated.

Sparse Levenberg-Marquardt

- N^3complexity for solving $\Delta = \mathrm{N}'^{-1}\,\mathrm{J}^{\mathrm{T}} e_0$
  - prohibitive for large problems

    (100 views 10,000 points ~30,000 unknowns)

- Partition parameters
  - partition A
  - partition B (only dependent on A and itself)

Sparse bundle adjustment

residuals $D(\mathrm{m}_{ki}, \hat{\mathbf{P}}_k \hat{\mathrm{M}}_i)^2$

normal equations $\begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta(\mathbf{P}) \\ \Delta(\mathrm{M}) \end{bmatrix} = \begin{bmatrix} \epsilon(\mathbf{P}) \\ \epsilon(\mathrm{M}) \end{bmatrix}$

with

$$\mathbf{U}_k = \sum_i \left(\frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathbf{P}}_k}\right)^\top \frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathbf{P}}_k} \qquad \epsilon(\mathbf{P}_k) = \sum_i \left(\frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathbf{P}}_k}\right)^\top \epsilon_{ki}$$

$$\mathbf{V}_i = \sum_k \left(\frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathrm{M}}_i}\right)^\top \frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathrm{M}}_i} \qquad \epsilon(\mathrm{M}_i) = \sum_i \left(\frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathrm{M}}_i}\right)^\top \epsilon_{ki}$$

$$\mathbf{W}_{ki} = \left(\frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathbf{P}}_k}\right)^\top \frac{\partial \hat{\mathrm{m}}_{ki}}{\partial \hat{\mathrm{M}}_i}$$

note: tie points should be in partition A

Sparse bundle adjustment

normal equations:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta(\mathbf{P}) \\ \Delta(\mathbf{M}) \end{bmatrix} = \begin{bmatrix} \epsilon(\mathbf{P}) \\ \epsilon(\mathbf{M}) \end{bmatrix}$$

modified normal equations:

$$\begin{bmatrix} \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^\top & \mathbf{0} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta(\mathbf{P}) \\ \Delta(\mathbf{M}) \end{bmatrix} = \begin{bmatrix} \epsilon(\mathbf{P}) - \mathbf{W}\mathbf{V}^{-1}\epsilon(\mathbf{M}) \\ \epsilon(\mathbf{M}) \end{bmatrix}$$

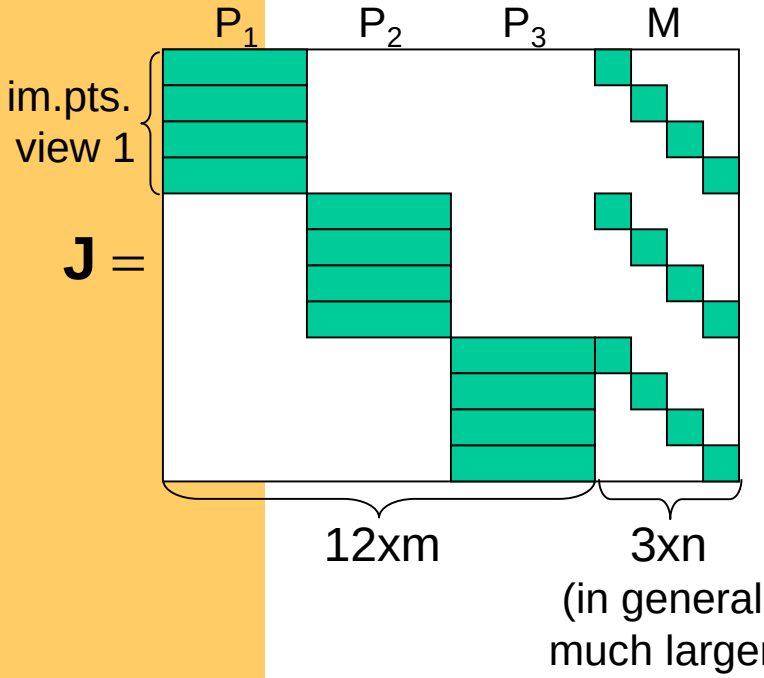
solve in two parts:

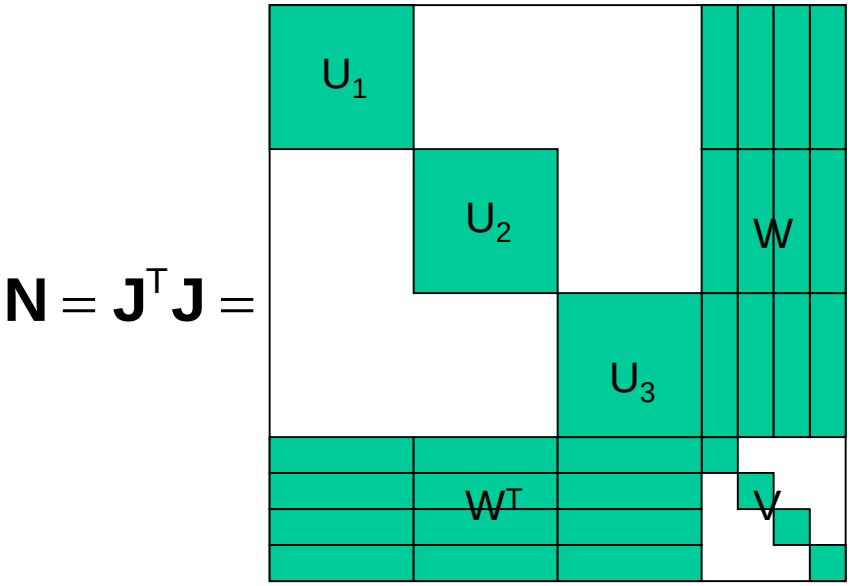
$$\left(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^\top\right) \Delta(\mathbf{P}) = \epsilon(\mathbf{P}) - \mathbf{W}\mathbf{V}^{-1}\epsilon(\mathbf{M})$$

$$\Delta(\mathbf{M}) = \mathbf{V}^{-1}\left(\epsilon(\mathbf{M}) - \mathbf{W}^\top \Delta(\mathbf{P})\right)$$

Sparse bundle adjustment

Jacobian of $\sum_{k=1}^{m}\sum_{i=1}^{n} D\left(\mathrm{m_{ki}}, \hat{\mathrm{P}}_k\left(\hat{\mathrm{M}}_i\right)\right)^2$ has sparse block structure





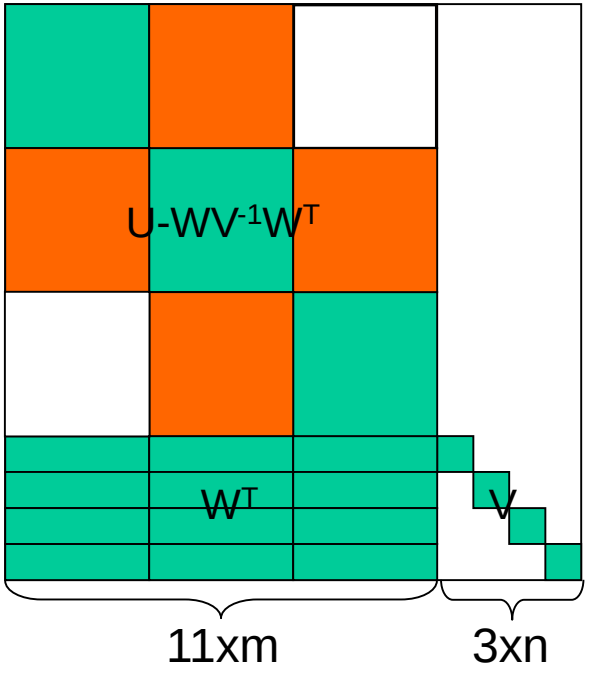
Needed for non-linear minimization

Sparse bundle adjustment

Eliminate dependence of camera/motion parameters on structure parameters

Note in general 3n >> 11m

$$\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \times \mathbf{N} =$$



Allows much more efficient computations

e.g. 100 views,10000 points,
solve ±1000x1000, not ±30000x30000

Often still band diagonal

use sparse linear algebra algorithms

Sparse bundle adjustment

normal equations:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{W}\mathbf{V}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U} & \mathbf{W} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta(\mathbf{P}) \\ \Delta(\mathbf{M}) \end{bmatrix} = \begin{bmatrix} \epsilon(\mathbf{P}) \\ \epsilon(\mathbf{M}) \end{bmatrix}$$

modified normal equations:

$$\begin{bmatrix} \mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^\top & \mathbf{0} \\ \mathbf{W}^\top & \mathbf{V} \end{bmatrix} \begin{bmatrix} \Delta(\mathbf{P}) \\ \Delta(\mathbf{M}) \end{bmatrix} = \begin{bmatrix} \epsilon(\mathbf{P}) - \mathbf{W}\mathbf{V}^{-1}\epsilon(\mathbf{M}) \\ \epsilon(\mathbf{M}) \end{bmatrix}$$

solve in two parts:

$$\left(\mathbf{U} - \mathbf{W}\mathbf{V}^{-1}\mathbf{W}^\top\right) \Delta(\mathbf{P}) = \epsilon(\mathbf{P}) - \mathbf{W}\mathbf{V}^{-1}\epsilon(\mathbf{M})$$

$$\Delta(\mathbf{M}) = \mathbf{V}^{-1}\left(\epsilon(\mathbf{M}) - \mathbf{W}^\top \Delta(\mathbf{P})\right)$$

Sparse bundle adjustment

- Covariance estimation

$$\Sigma_a = \left(U - WV^{-1}W\right)^+$$

$$\Sigma_b = Y^T \Sigma_a Y + V^{-1} \qquad Y = WV^{-1}$$

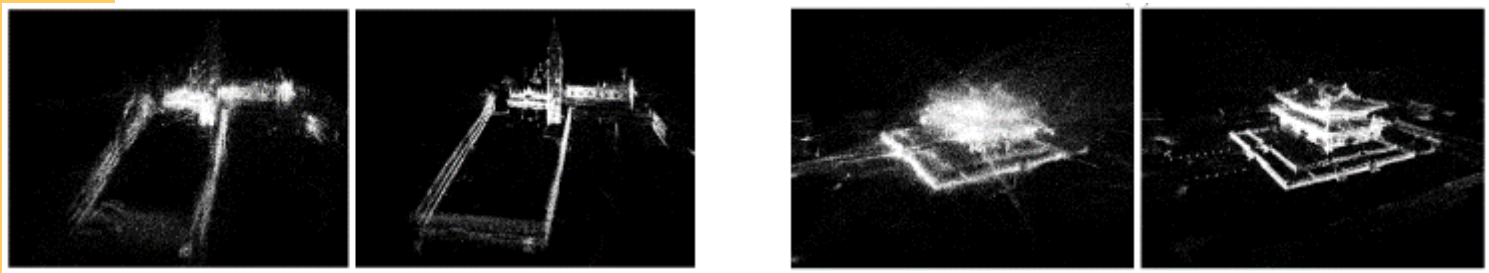
$$\Sigma_{ab} = -\Sigma_a Y$$

Direct vs. iterative solvers

- How do they work?
- When should you use them?

Large reduced camera matrices

- Use iterative preconditioned conjugate gradient
  - [Jeong, Nister, Steedly, Szeliski, Kweon, CVPR 2009]
  - [Agarwal, Snavely, Seitz, and Szeliski, ECCV 2009]
  - Block diagonal works well
  - Sometimes partial Cholesky does too
  - Direct solvers for small problems



Large-scale structure from motion

Richard Szeliski

Large-scale reconstruction

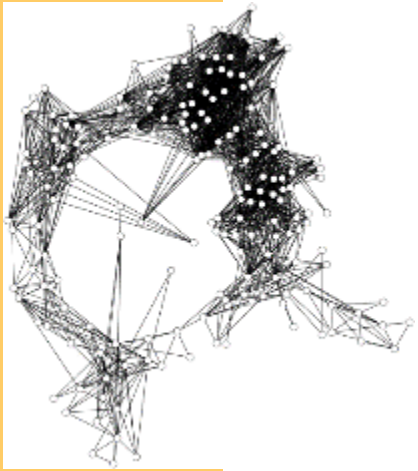
- Most of the models shown so far have had ~500 images

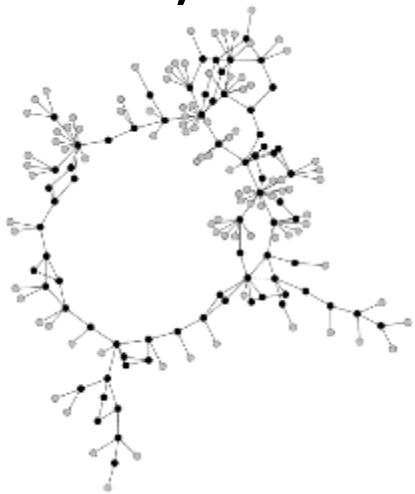
- How do we scale from 100s to 10,000s of images?

- Observation: Internet collections represent very non-uniform samplings of viewpoint

Skeletal graphs for efficient structure from motion

Noah Snavely, Steve Seitz,
Richard Szeliski
CVPR’2008





Skeletal set

- Goal: select a smaller set of images to reconstruct, without sacrificing *quality* of reconstruction

- Two problems:
  - How do we measure quality?
  - How do we find a subset of images with bounded quality loss?

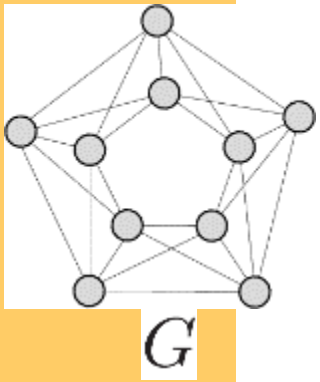
Skeletal set

Goal: given an image graph $G_{\mathcal{I}}$,

- select a small set *S* of ***important*** images to reconstruct, bounding the loss in ***quality*** of the reconstruction

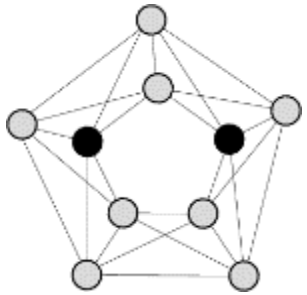
- Reconstruct the skeletal set *S*
- Estimate the remaining images with much faster pose estimation steps

Properties of the skeletal set



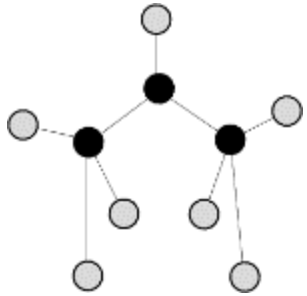
- Should touch all parts of G

  Dominating set



- Should form a single reconstruction

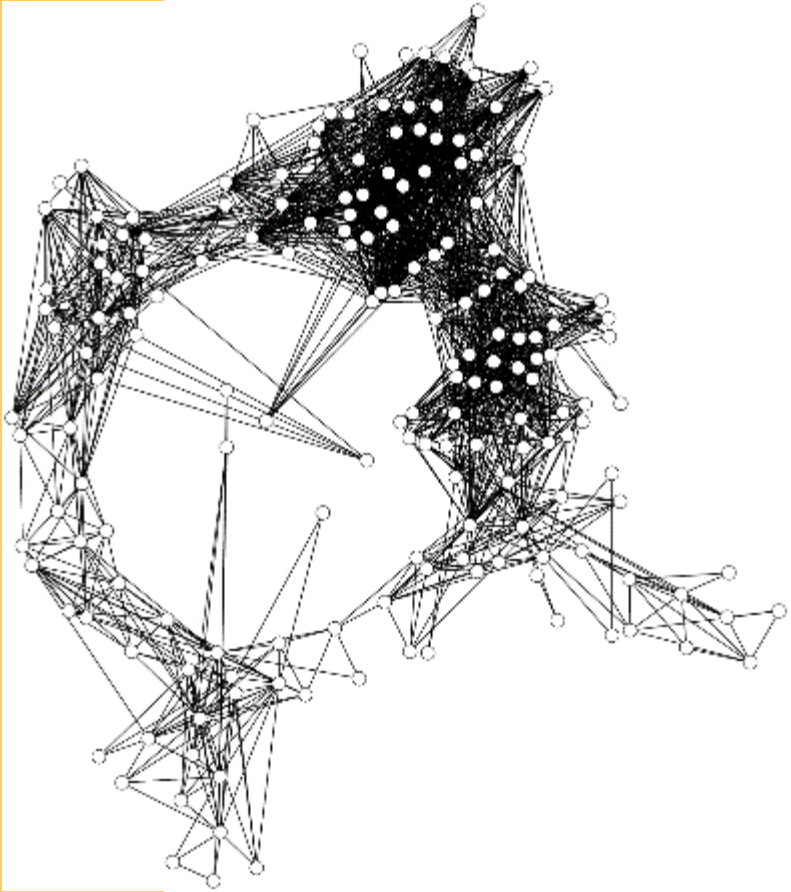
  Connected dominating set



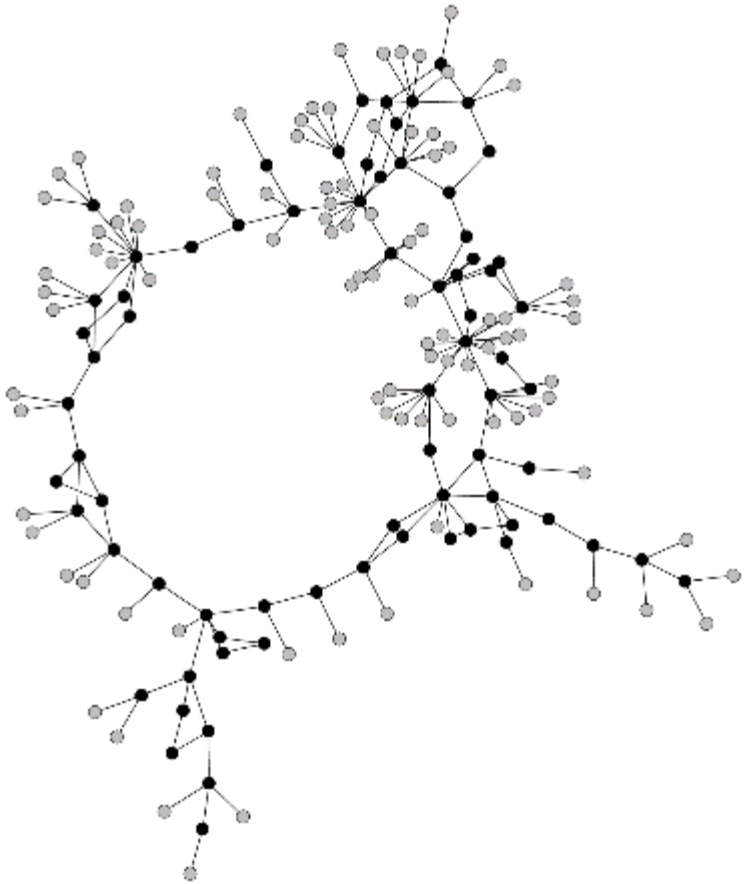
- Should result in an *accurate* reconstruction

?

Example



Full graph



Skeletal graph

Representing information in a graph

What kind of information?

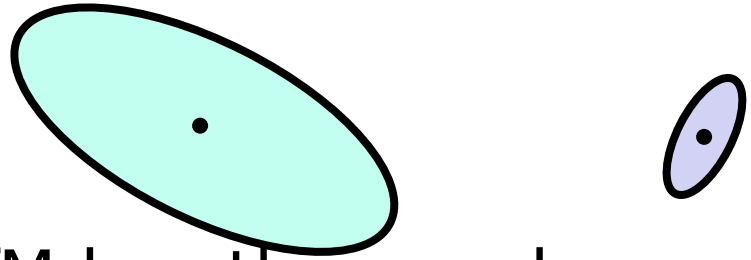
- No absolute information about camera positions
- Each edge provides information about the relative positions of two images

- … but not all edges are equally informative

We model information with the uncertainty (covariance) in pairwise camera positions

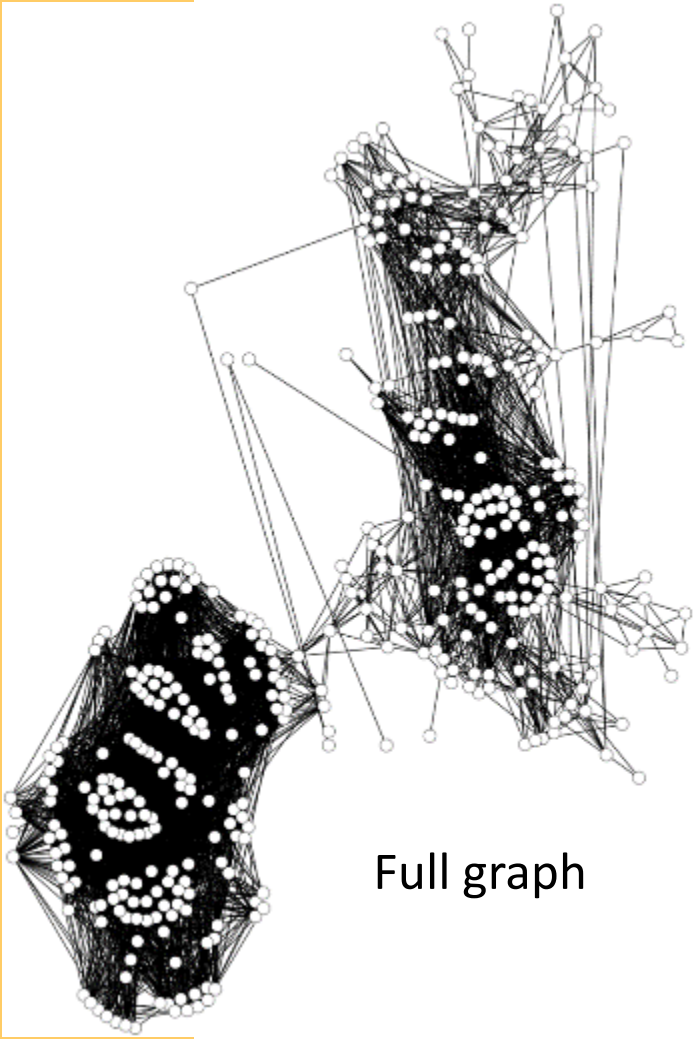
Estimating certainty

- Usually measured with covariance matrix

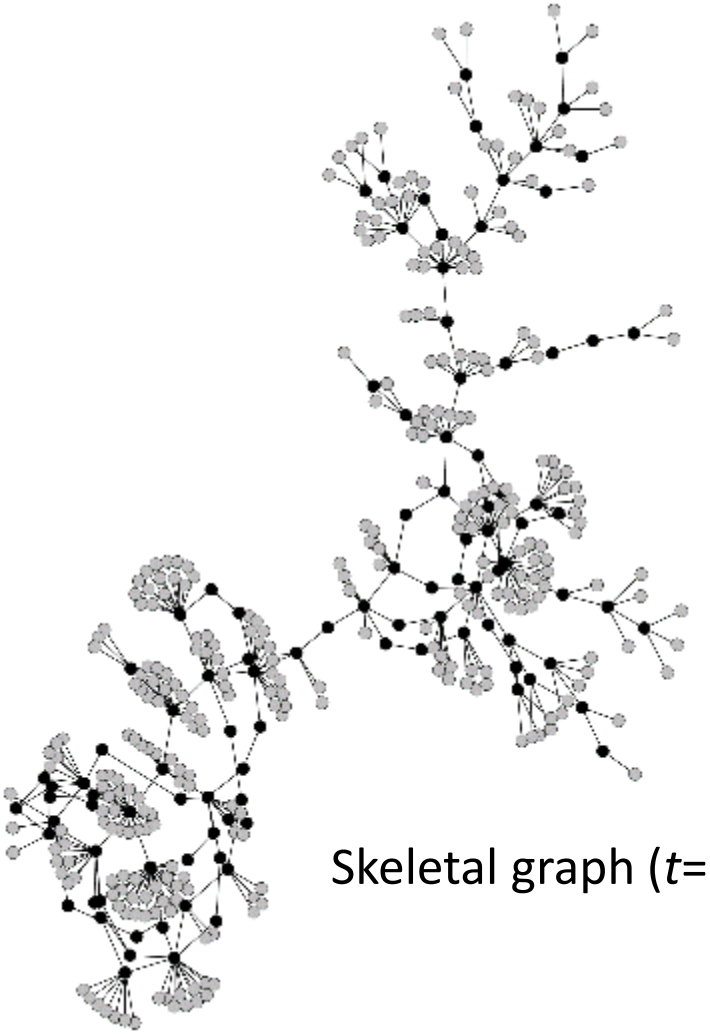


- SfM has thousands or millions of parameters
- We only measure covariance in camera positions (3x3 matrix for every camera)

Pantheon

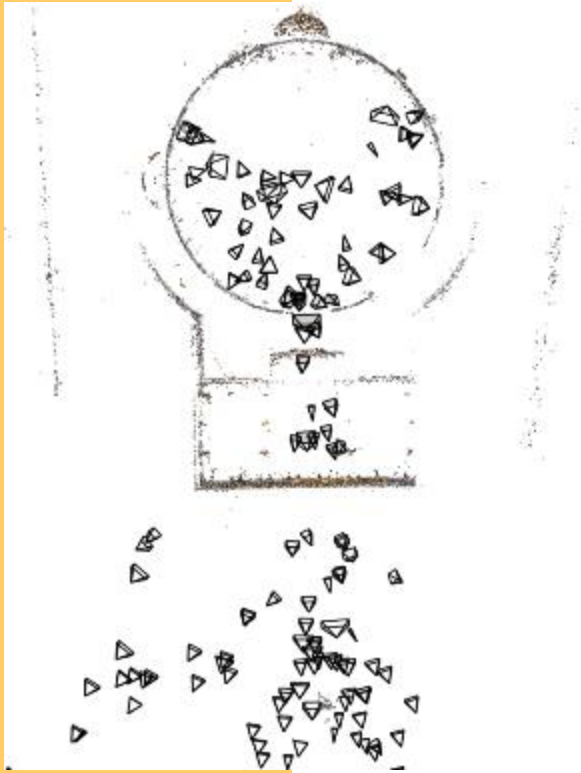


Full graph

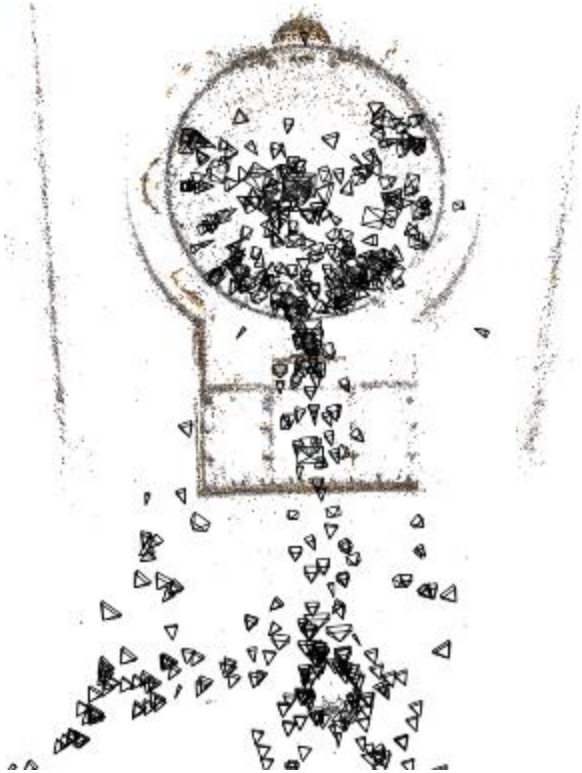


Skeletal graph (t=16)

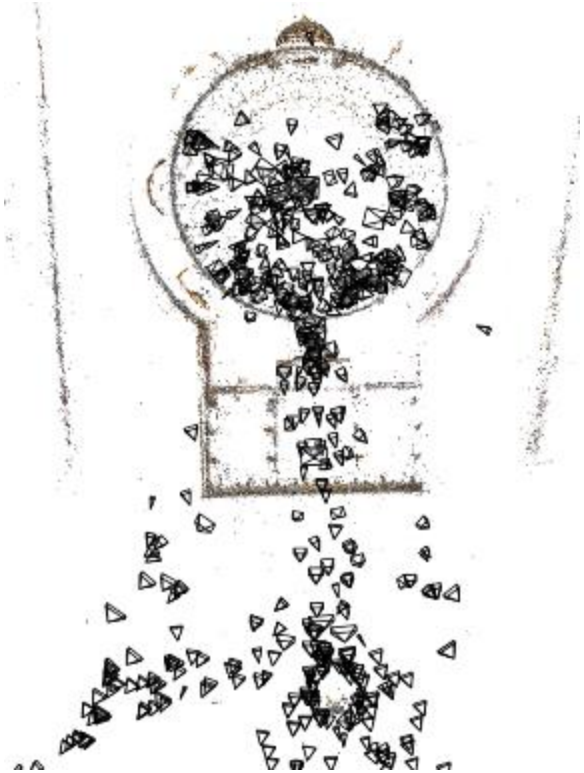
Pantheon



Skeletal reconstruction
101 images



After adding leaves
579 images

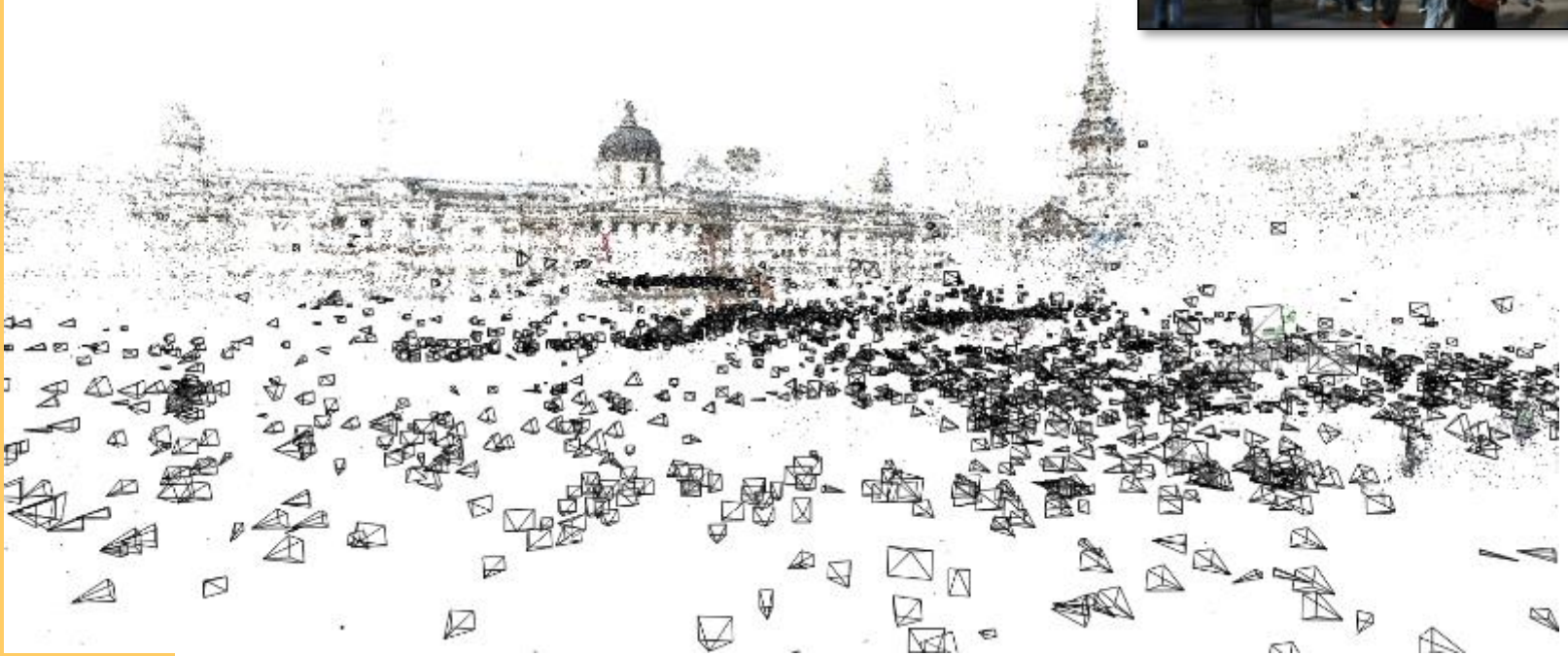


After final optimization
579 images

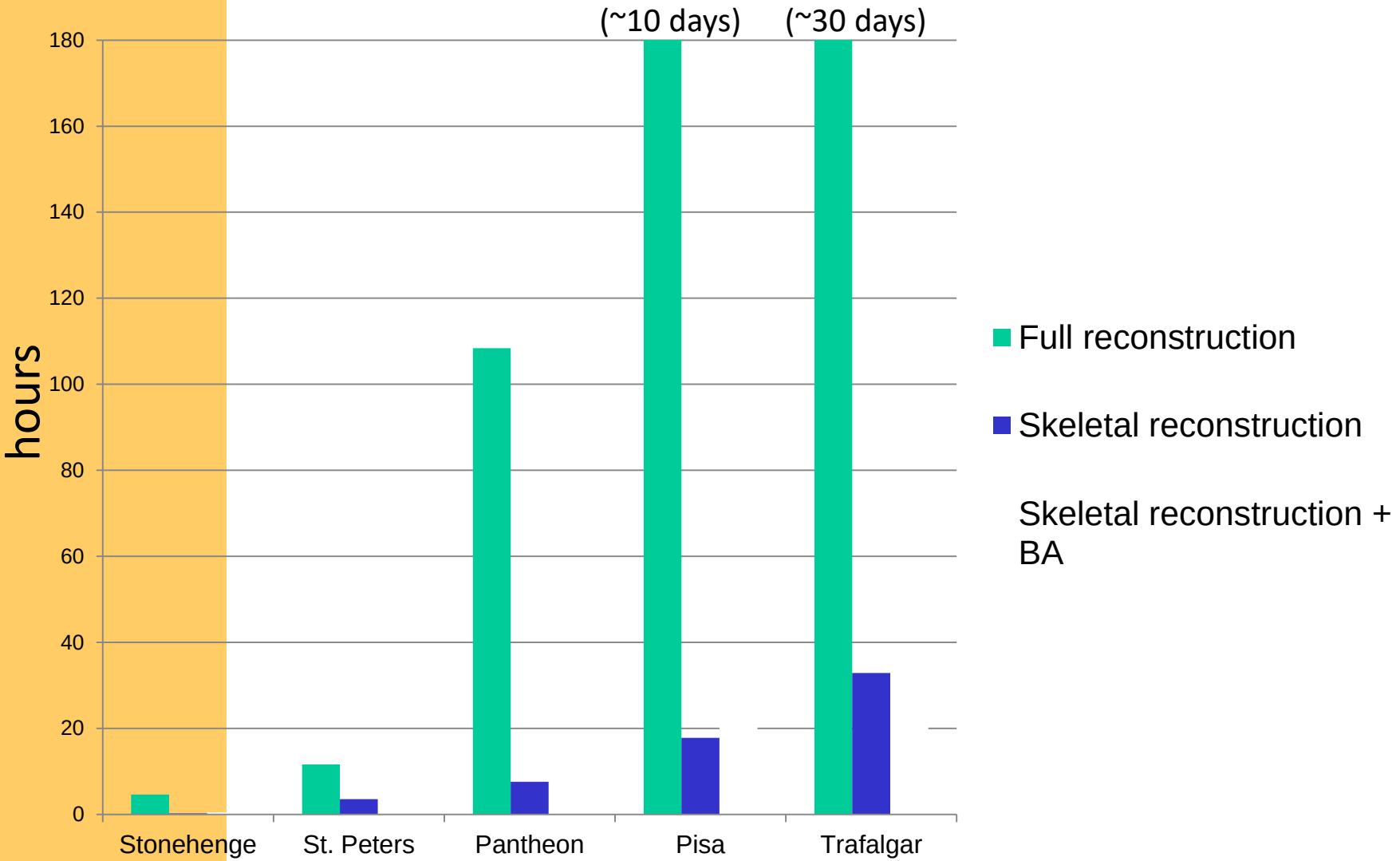
Trafalgar Square

2973 images registered (277 in skeletal set)

Trafalgar Square



Running time

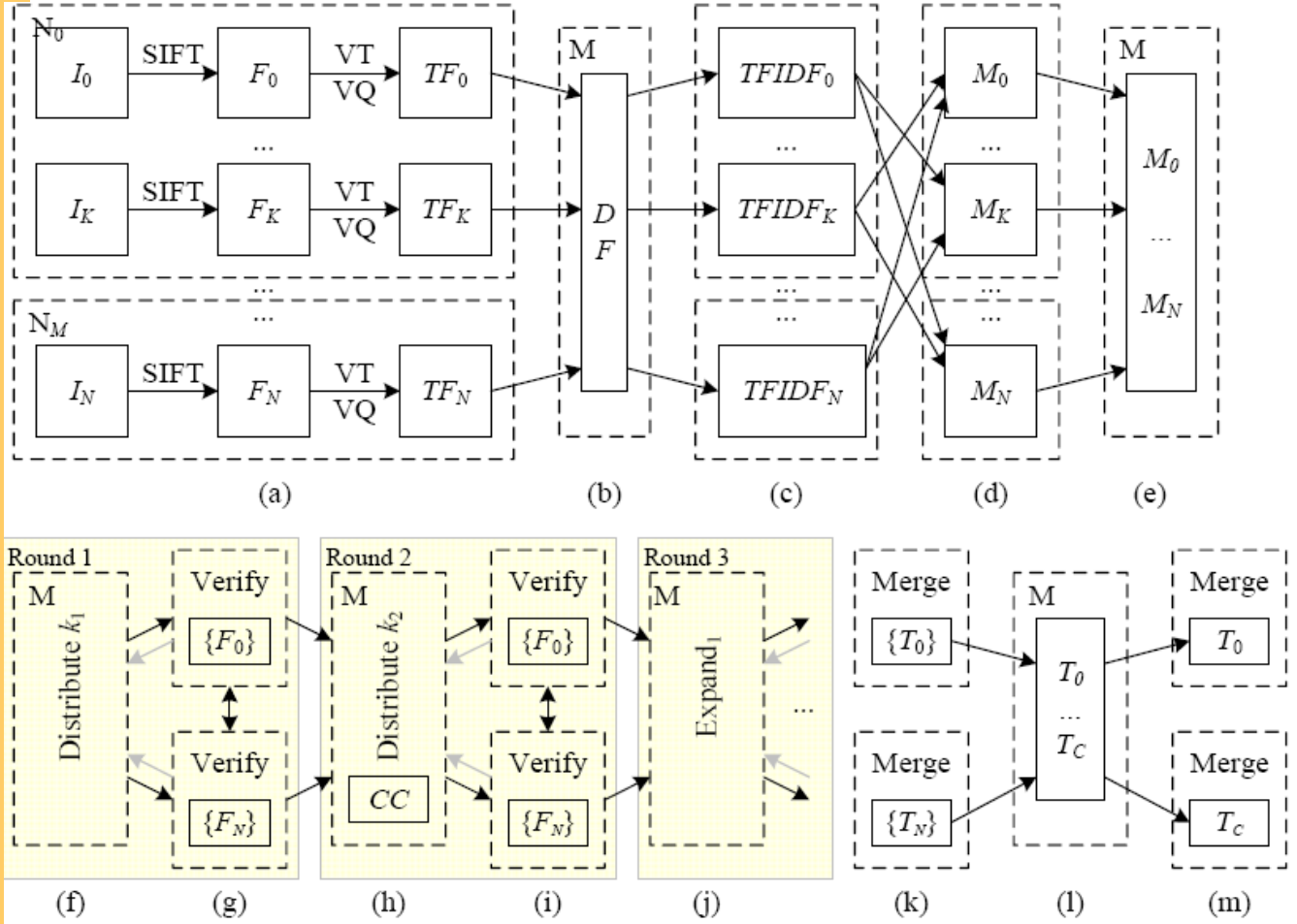


Building Rome in a Day

Sameer Agarwal, Noah Snavely,
Ian Simon, Steven M. Seitz,
Richard Szeliski

ICCV' 2009

Distributed matching pipeline



Query expansion

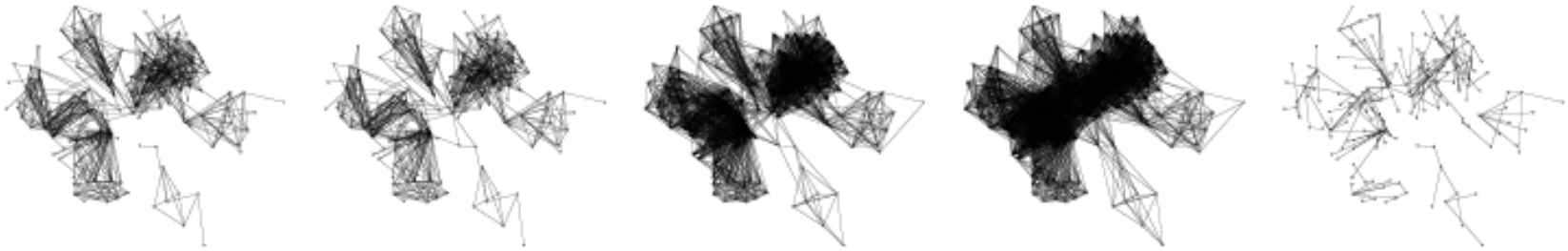


Figure 2. The evolution of the match graph as a function of the rounds of matching, and the skeletal set corresponding to it. Notice how the second round of matching merges the two components into one, and how rapidly the query expansion increases the density of the within component connections. The last column shows the skeletal set corresponding to the final match graph. The skeletal sets algorithm can break up connected components found during the match phase if it determines that a reliable reconstruction is not possible, which is what happens in this case.

Results: Dubrovnik

(a) Dubrovnik: Four different views and associated images from the largest connected component. Note that the component captures the entire old city, with both street-level and roof-top detail. The reconstruction consists of 4,585 images and 2,662,981 3D points with 11,839,682 observed features.

Results: Rome




Colosseum: 2,097 images, 819,242 points

Trevi Fountain: 1,935 images, 1,055,153 points



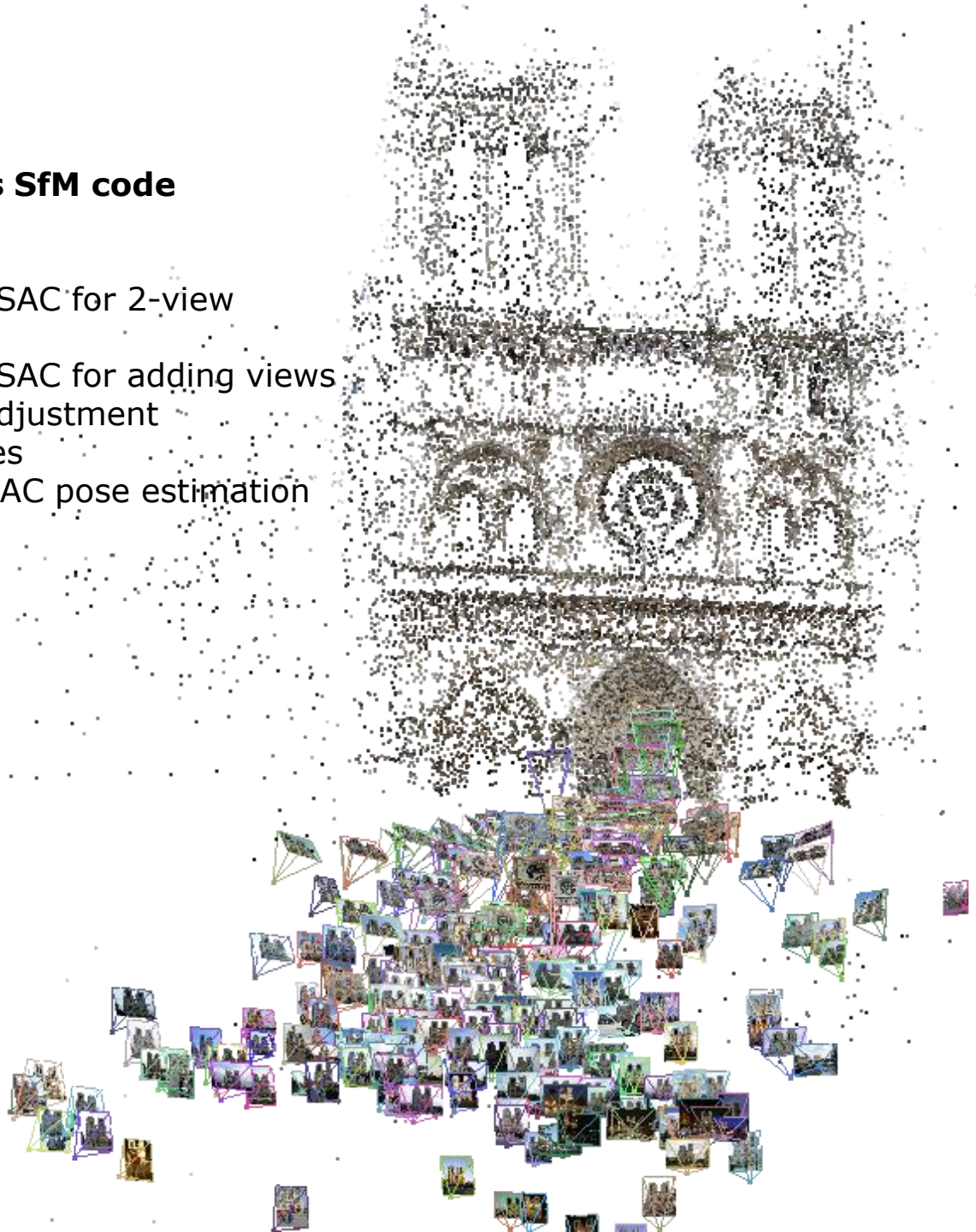

Pantheon: 1,032 images, 530,076 points

Hall of Maps: 275 images, 230,182 points

(b) Rome: Four of the largest connected components visualized at canonical viewpoints [14].

Changchang Wu's SfM code

for iconic graph
• uses 5-point+RANSAC for 2-view initialization
• uses 3-point+RANSAC for adding views
• performs bundle adjustment
For additional images
• use 3-point+RANSAC pose estimation



Rome on a cloudless day

- GIST & clustering (1h35) (Frahm et al. ECCV 2010)

SIFT & Geometric verification (11h36)

SfM & Bundle (8h35)

Dense Reconstruction (1h58)

Some numbers

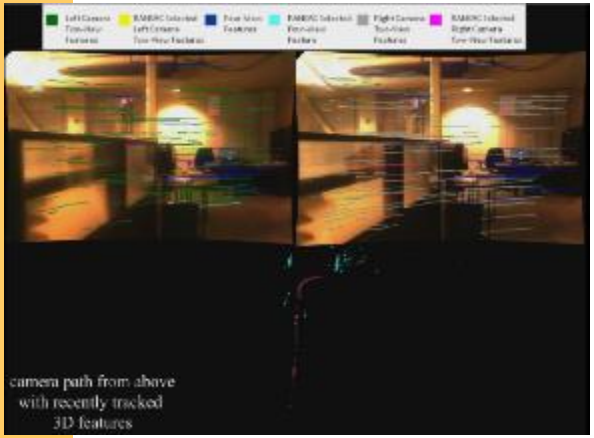
- 1PC
- 2.88M images
- 100k clusters
- 22k SfM with 307k images
- 63k 3D models
- Largest model 5700 images
- Total time 23h53

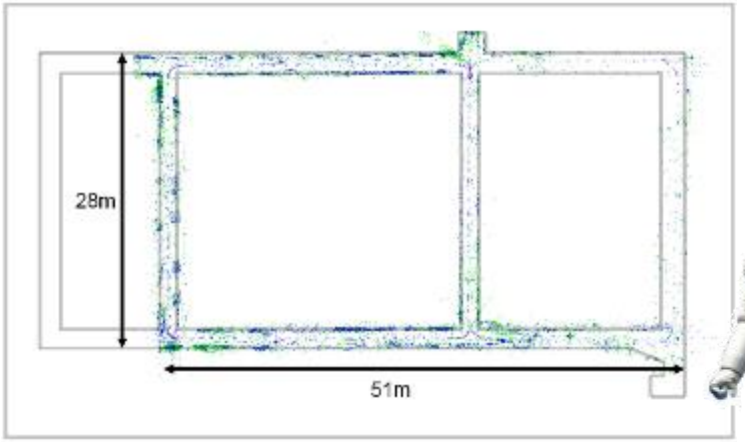
Related problems

- On-line structure from motion and SLaM (Simultaneous Localization and Mapping)
  - Kalman filter (linear)
  - Particle filters (non-linear)
  - Bundle-adjustment
  - Pose graph optimization

Visual SLAM

- Real-time Stereo Visual SLAM
  (Clipp et al., IROS2010)





Open challenges

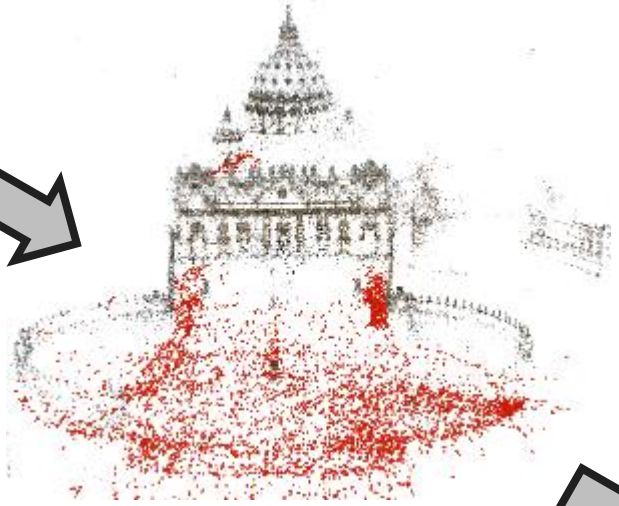
- Large scale structure from motion
  - Complete building
  - Complete city
- Life-long mapping
  - Merging maps
  - Detecting change
  - Avoid degradation over time

Open Source Pipeline

Unstructured Images

SfM

Sparse Model



MVS

Dense Model

COLMAP - 3D reconstruction pipeline:
https://github.com/colmap/colmap

Next week:
Specific object recognition