

# N-View Methods

Diana Mateus

Computer Aided Medical Procedures  
Technische Universität München

3D Computer Vision II

Inspired on Slides from Adrien Bartoli

# Outline

---

## ① Structure from Motion (SFM) with Uncalibrated Cameras

### Non Euclidean SFM Batch Algorithms

- Affine Factorization

- Orthographic Factorization

- Perspective Factorization

### Sequential Algorithms

### Hierarchical Algorithms

## ② Applications

# Structure From Motion (SFM) with Uncalibrated Cameras

---

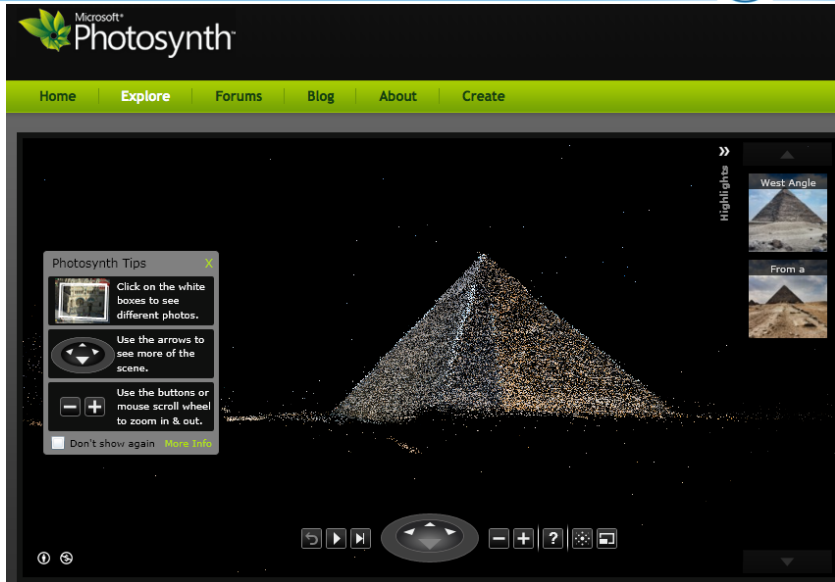
**Requirements** A set of pictures taken from one camera in motion or multiple cameras.

**Goal** Compute from the pictures:

- a (Euclidean) 3D model of the scene structure.
- the camera parameters (intrinsic and extrinsic).

**Assumptions**




- The scene is rigid.
- Some of the camera intrinsic parameters are known or constant.



Microsoft®  
**Photosynth**



Home | Explore | Forums | Blog | About | Create

**Photosynth Tips** X

-  Click on the white boxes to see different photos.
-  Use the arrows to see more of the scene.
-  Use the buttons or mouse scroll wheel to zoom in & out.

Don't show again [More Info](#)

**Highlights** >>

- West Angle 
- From a 

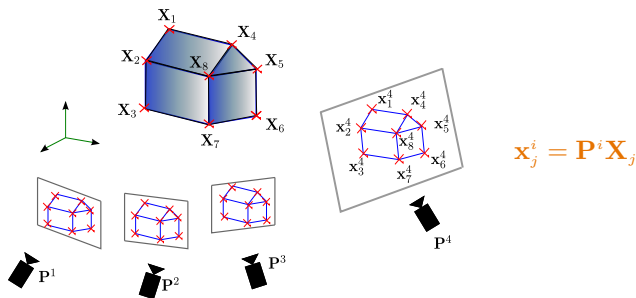
Navigation controls: Back, Forward, Stop, Directional Pad, Zoom In (+), Zoom Out (-), Help (?), Settings, Full Screen

## Why Using Uncalibrated Cameras?

---

- Calibrated approaches require (at least) one image of a calibration object (i.e. of which the Euclidean 3D model is known).
- Uncalibrated approaches:
  - Do not require a special material, neither user interaction.
  - Impose only mild assumptions on the imaging conditions (thus applicable to images from unknown sources, e.g. internet).
  - Deal with on-line calibration ( useful when camera intrinsic parameters vary, due to e.g. zooming).
  - Provide flexible and efficient methods for camera self-calibration.

## Preliminaries

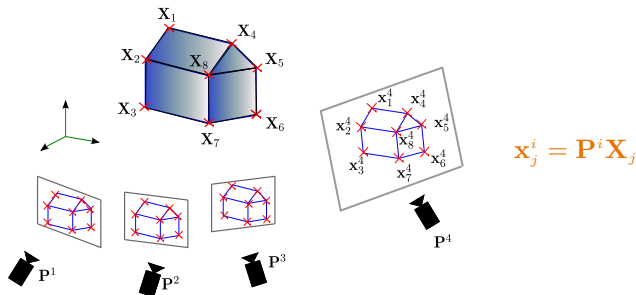


**Scene structure:** Set of 3D points  $\mathbf{X}_j$ , with  $j = \{1, \dots, n\}$ . Other features can be used (e.g. lines, curves, planes).

**Camera geometry:** Affine or perspective (pin-hole) projection, represented by matrices  $\mathbf{P}^i$ , with  $i = \{1, \dots, m\}$ .

**Images:** 2D points matched across the images with coordinates  $\mathbf{x}_j^i$ .

## Preliminaries



**Scene structure:** (**Unknown**) Set of 3D points  $\mathbf{X}_j$ , with  $j = \{1, \dots, n\}$ .  
Other features can be used (e.g. lines, curves, planes).

**Camera geometry:** (**Unknown**) Affine or perspective (pin-hole) projection, represented by matrices  $\mathbf{P}^i$ , with  $i = \{1, \dots, m\}$ .

**Images:** (**Given**) 2D points matched across the images with coordinates  $\mathbf{x}_j^i$ .

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :



## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).
  - $11m + 3n$  unknowns ? ( $3n$  due to 3D point coordinates).

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).
  - $11m + 3n$  unknowns ? ( $3n$  due to 3D point coordinates).
  - Projective reconstruction up to a projective transformation ( $4 \times 4$  matrix with 15 DoFs).

## Is it Possible?: A Counting Argument

---

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).
  - $11m + 3n$  unknowns ? ( $3n$  due to 3D point coordinates).
  - Projective reconstruction up to a projective transformation ( $4 \times 4$  matrix with 15 DoFs).
  - $11m + 3n - 15$  unknowns

## Is it Possible?: A Counting Argument

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).
  - $11m + 3n$  unknowns ? ( $3n$  due to 3D point coordinates).
  - Projective reconstruction up to a projective transformation ( $4 \times 4$  matrix with 15 DoFs).
  - $11m + 3n - 15$  unknowns
- The problem can be solved (in general)  $2nm \geq 11m + 3n - 15$   
e.g.  $n = 10 \quad m = 3$ ,

## Is it Possible?: A Counting Argument

- Look for cameras  $\mathbf{P}^i$  and 3D points  $\mathbf{X}_j$  that explain image points:

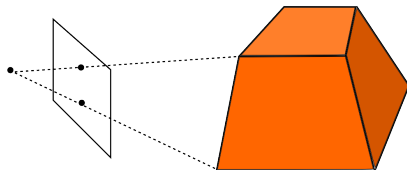
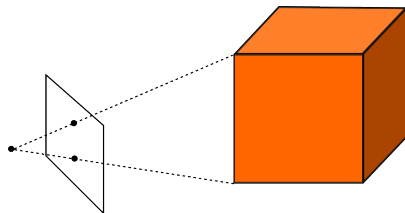
$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j \quad i = \{1, \dots, m\} \quad j = \{1, \dots, n\}$$

- For  $n$  points projected in  $m$  images :
  - Each point has 2 coordinates.
  - We can build  $2nm$  constraints.
  - We want to recover  $m$  projection matrices  $\mathbf{P}^i$  (Each  $3 \times 4$  with 11 d.o.f. under affine assumption).
  - $11m + 3n$  unknowns ? ( $3n$  due to 3D point coordinates).
  - Projective reconstruction up to a projective transformation ( $4 \times 4$  matrix with 15 DoFs).
  - $11m + 3n - 15$  unknowns
- The problem can be solved (in general)  $2nm \geq 11m + 3n - 15$   
e.g.  $n = 10$   $m = 3$ , then  $60 \geq 33 + 30 - 15 = 48$  .



# Projective Reconstruction

---



## SFM Method Overview

---

- 1 Find correspondences across the images.

## SFM Method Overview

---

- 1 Find correspondences across the images.
- 2 Initialization with a “good enough” sub-optimal estimate, e.g. using a Factorization-based algorithm:
  - Batch.
  - Sequential.
  - Hierarchical.

## SFM Method Overview

---

- ① Find correspondences across the images.
- ② Initialization with a “good enough” sub-optimal estimate, e.g. using a Factorization-based algorithm:
  - Batch.
  - Sequential.
  - Hierarchical.
- ③ Bundle Adjustment (Optional):
  - Non-linear minimization of the reprojection error (discrepancy between measured and predicted feature coordinates).

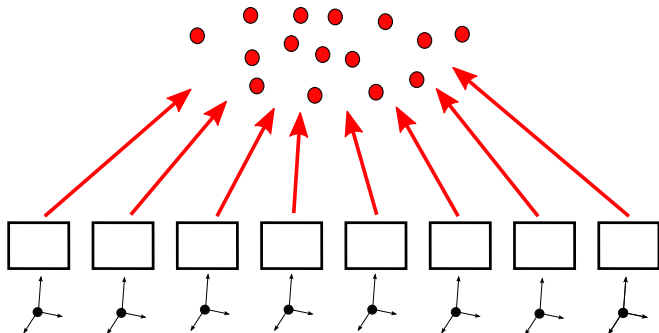
## SFM Method Overview

---

- 1 Find correspondences across the images.
- 2 Initialization with a “good enough” sub-optimal estimate, e.g. using a Factorization-based algorithm:
  - Batch.
  - Sequential.
  - Hierarchical.
- 3 Bundle Adjustment (Optional):
  - Non-linear minimization of the reprojection error (discrepancy between measured and predicted feature coordinates).
- 4 Self-Calibration (Optional).
  - Estimation of the intrinsic parameters of the camera using only the information available in the images taken by that camera.

## Batch Algorithms

---



## Batch Algorithms

---

- Uniform and simultaneous use of all data.
- Two type of approaches:

### Factorization

- Simple and fast.
- All points need to be seen in all images.
- measurements (points) need to be valid.
- Orthographic [Tomasi, 1992].
- Affine [Kahl and Heyden, 1998].
- Projective [Sturm, 1996], iterative [Triggs, 1996; Heyden, 1997].

**Closure constraints** Handles unseen and erroneous points, but suffers from algebraic approximation, e.g. [Kahl and Heyden, 1998].

# Outline

---

## 1 Structure from Motion (SFM) with Uncalibrated Cameras

### Non Euclidean SFM Batch Algorithms

- Affine Factorization

- Orthographic Factorization

- Perspective Factorization

### Sequential Algorithms

### Hierarchical Algorithms

## 2 Applications



## Affine Factorization

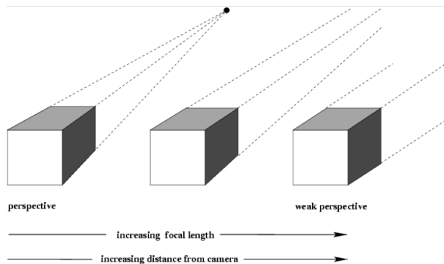
Assumptions:

- All points are seen in all views.
- There are no blunders.
- Affine camera model (uncalibrated) 8 d.o.f.:

$$\underbrace{\begin{pmatrix} x_{j;1}^i \\ x_{j;2}^i \end{pmatrix}}_{\mathbf{x}_j^i} = \underbrace{\begin{pmatrix} P_{11}^i & P_{12}^i & P_{13}^i \\ P_{21}^i & P_{22}^i & P_{23}^i \end{pmatrix}}_{\mathbf{P}^i} \underbrace{\begin{pmatrix} X_{j;1} \\ X_{j;2} \\ X_{j;3} \end{pmatrix}}_{\mathbf{X}_j} + \underbrace{\begin{pmatrix} t_1^i \\ t_2^i \end{pmatrix}}_{\mathbf{t}^i}$$

- $\mathbf{x}_j^i$  and  $\mathbf{X}_j$  in inhomogeneous coordinates.

## Affine Factorization: Camera model



## Affine Factorization: Goal

Find a reconstruction of the scene that minimizes the geometric error in the image coordinate measurements. That is, estimate:

- Cameras  $\{\mathbf{P}^i, \mathbf{t}^i\}$
- 3D points  $\{\mathbf{X}_j\}$

such that the distance between the estimated image points  $\hat{\mathbf{x}}_j^i = \mathbf{P}^i \mathbf{X}_j + \mathbf{t}^i$  and measured image points  $\mathbf{x}_j^i$  is minimized:

$$\min_{\mathbf{P}^i, \mathbf{t}^i, \mathbf{X}_j} \sum_{i=1}^m \sum_{j=1}^n \|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\|^2$$

$$\min_{\mathbf{P}^i, \mathbf{t}^i, \mathbf{X}_j} \sum_{i=1}^m \sum_{j=1}^n \|\mathbf{x}_j^i - (\mathbf{P}^i \mathbf{X}_j + \mathbf{t}^i)\|^2$$

## Affine Factorization: Algorithm Overview

---

- 1 Compute the translations  $\mathbf{t}^i$ .
- 2 Center the data.
- 3 Construct the  $2m \times n$  measurement matrix  $\mathbf{M}$  (a single matrix equation).
- 4 Factorize  $\mathbf{M}$ , i.e. find the closest rank 3 matrix which is closest to  $\mathbf{M}$  in Frobenious norm (SVD of  $\mathbf{M}$  truncated to rank 3).

# 1. Compute the Translations

---

- Calculate the average of the points seen e.g. in image 1:

$$\mathbf{x}_1^1 = \mathbf{P}^1 \mathbf{X}_1 + \mathbf{t}^1$$

$$\vdots$$

$$\mathbf{x}_n^1 = \mathbf{P}^1 \mathbf{X}_n + \mathbf{t}^1$$

$$\frac{1}{n} \sum_{j=1}^n \mathbf{x}_j^1 = \frac{1}{n} \mathbf{P}^1 \sum_{j=1}^n \mathbf{X}_j + \mathbf{t}^1$$

# 1. Compute the Translations

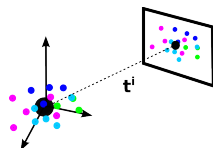
- Calculate the average of the points seen e.g. in image 1:

$$\mathbf{x}_1^1 = \mathbf{P}^1 \mathbf{X}_1 + \mathbf{t}^1$$

$$\vdots$$

$$\mathbf{x}_n^1 = \mathbf{P}^1 \mathbf{X}_n + \mathbf{t}^1$$

$$\frac{1}{n} \sum_{j=1}^n \mathbf{x}_j^1 = \frac{1}{n} \mathbf{P}^1 \sum_{j=1}^n \mathbf{X}_j + \mathbf{t}^1$$



- Affine cameras map the centroid of a set of 3D points to the centroid of their projection. Set the (arbitrary) origin of the 3D coordinate frame to the centroid of the unknown 3D points

$$\left( \frac{1}{n} \mathbf{P}^a \sum_{j=1}^n \mathbf{X}_j \right):$$

$$\mathbf{t}^i = \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j^i$$

## 2. Center Image Points

- Transform the image measurements by subtracting the centroid of the points in each image.

$$\mathbf{x}_j^i = \mathbf{P}^i \mathbf{X}_j + \mathbf{t}^i \quad \Rightarrow \quad \begin{aligned} \mathbf{m}_j^i &\triangleq \mathbf{x}_j^i - \mathbf{t}^i \\ &= (\mathbf{P}^i \mathbf{X}_j + \mathbf{t}^i) - \mathbf{t}^i \\ &= \mathbf{P}^i \mathbf{X}_j \end{aligned}$$

- The resultant centered projection model is bilinear.
- Removing the translation, simplifies the problem:

$$\min_{\mathbf{P}^i, \mathbf{X}_j} \sum_{i=1}^m \sum_{j=1}^n \|\mathbf{m}_j^i - (\mathbf{P}_j^i \mathbf{X}_j)\|^2$$

- All points must be seen in all cameras. (why?)

### 3. A Single Matrix Equation

- Gather the centered projections for image 1 in a single matrix equation:

$$\underbrace{\begin{pmatrix} \mathbf{m}_1^1 & \mathbf{m}_2^1 & \dots & \mathbf{m}_n^1 \end{pmatrix}}_{2 \times n} = \underbrace{\mathbf{P}^1}_{2 \times 3} \underbrace{\begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_n \end{pmatrix}}_{3 \times n}$$

- Further, gather the equations for all images

$$\underbrace{\begin{pmatrix} \mathbf{m}_1^1 & \mathbf{m}_2^1 & \dots & \mathbf{m}_n^1 \\ \mathbf{m}_1^2 & \mathbf{m}_2^2 & \dots & \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{m}_1^m & \mathbf{m}_2^m & \dots & \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_n \end{pmatrix}}_{\mathbf{S}}$$



### 3. A Single Matrix Equation

$$\underbrace{\begin{pmatrix} \mathbf{m}_1^1 & \mathbf{m}_2^1 & \dots & \mathbf{m}_n^1 \\ \mathbf{m}_1^2 & \mathbf{m}_2^2 & \dots & \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{m}_1^m & \mathbf{m}_2^m & \dots & \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\left( \mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_n \right)}_{\mathbf{S}}$$

- The  $(2m \times n)$  matrix  $\mathbf{M}$  is the measurement matrix.
- The  $(2m \times 3)$  matrix  $\mathbf{P}$  is the (affine) projection matrix.
- The  $(3 \times n)$  matrix  $\mathbf{S}$  is the structure matrix.

### 3. A Single Matrix Equation

$$\underbrace{\begin{pmatrix} \mathbf{m}_1^1 & \mathbf{m}_2^1 & \dots & \mathbf{m}_n^1 \\ \mathbf{m}_1^2 & \mathbf{m}_2^2 & \dots & \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{m}_1^m & \mathbf{m}_2^m & \dots & \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\left( \mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_n \right)}_{\mathbf{S}}$$

- The  $(2m \times n)$  matrix  $\mathbf{M}$  is the measurement matrix.
- The  $(2m \times 3)$  matrix  $\mathbf{P}$  is the (affine) projection matrix.
- The  $(3 \times n)$  matrix  $\mathbf{S}$  is the structure matrix.
- In the absence of noise, the measurement matrix has rank 3 at most.

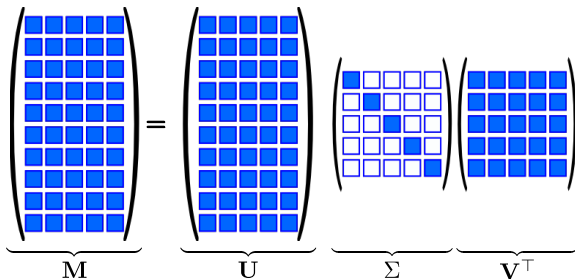
### 3. A Single Matrix Equation

---

- In the presence of noise  $\mathbf{M} = \mathbf{P}\mathbf{S}$  is not exactly verified.
- We look for the matrix  $\hat{\mathbf{M}}$  that is the closest possible to  $\mathbf{M}$  in Frobenius norm sense.

$$\begin{aligned}\|\mathbf{M} - \hat{\mathbf{M}}\|_F^2 &= \sum_{ij} (M_{ij} - \hat{M}_{ij})^2 \\ &= \sum_{ij} \|(\mathbf{m}_j^i - \hat{\mathbf{m}}_j^i)\|^2 \\ &= \sum_{ij} \|(\mathbf{m}_j^i - \hat{\mathbf{P}}^i \hat{\mathbf{X}}_j)\|^2\end{aligned}$$

## 4. Factorize M


$$\underbrace{\begin{pmatrix} \text{10x10 grid of blue squares} \end{pmatrix}}_M = \underbrace{\begin{pmatrix} \text{10x10 grid of blue squares} \end{pmatrix}}_U \underbrace{\begin{pmatrix} \text{10x10 grid with blue squares on diagonal} \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} \text{10x10 grid of blue squares} \end{pmatrix}}_{V^T}$$

- The measurement matrix is factorized with SVD as  $M = U\Sigma V^T$ .

## 4. Factorize M

$$\begin{array}{c}
 M = \underbrace{\begin{pmatrix} \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \\ \text{yellow} & \text{blue} \end{pmatrix}}_{\mathbf{U}} \underbrace{\begin{pmatrix} \text{green} & \text{white} & \text{white} & \text{white} \\ \text{white} & \text{green} & \text{white} & \text{white} \\ \text{white} & \text{white} & \text{white} & \text{blue} \\ \text{white} & \text{white} & \text{white} & \text{white} \end{pmatrix}}_{\Sigma} \underbrace{\begin{pmatrix} \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} & \text{red} \end{pmatrix}}_{\mathbf{V}^T} \approx \underbrace{\begin{pmatrix} \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \\ \text{yellow} & \text{yellow} & \text{yellow} \end{pmatrix}}_{\mathbf{U}'} \underbrace{\begin{pmatrix} \text{green} & \text{white} & \text{white} \\ \text{white} & \text{green} & \text{white} \\ \text{white} & \text{white} & \text{green} \end{pmatrix}}_{\Sigma'} \underbrace{\begin{pmatrix} \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \\ \text{red} & \text{red} & \text{red} \end{pmatrix}}_{\mathbf{V}'^T} \\
 \\
 \underbrace{\hspace{10em}}_{\hat{\mathbf{P}}} \underbrace{\hspace{10em}}_{\hat{\mathbf{S}}} \\
 \underbrace{\hspace{10em}}_{\hat{\mathbf{P}}} \underbrace{\hspace{10em}}_{\hat{\mathbf{S}}}
 \end{array}$$

- The estimate of the joint **projection matrix**  $\hat{\mathbf{P}}$  is the first 3 columns of  $\mathbf{U}$ .
- The estimate of the **structure matrix**  $\hat{\mathbf{S}}$  is the first 3 rows of  $\mathbf{V}^T$ .

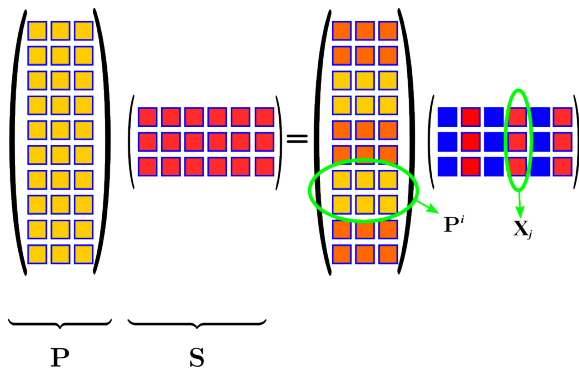
## 4. Factorize $M$

$$\underbrace{\begin{pmatrix} \text{12x12 grid of blue squares} \end{pmatrix}}_{\hat{M}} = \underbrace{\begin{pmatrix} \text{12x12 grid of yellow squares} \end{pmatrix}}_{\hat{P}} \underbrace{\begin{pmatrix} \text{3x12 grid of red squares} \end{pmatrix}}_{\hat{S}} = \underbrace{\begin{pmatrix} \text{12x12 grid of yellow squares} \end{pmatrix}}_{\hat{P}} \underbrace{\begin{pmatrix} \text{3x6 grid of green squares} \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \text{6x6 grid of green squares} \end{pmatrix}}_{A^{-1}} \underbrace{\begin{pmatrix} \text{3x12 grid of red squares} \end{pmatrix}}_{\hat{S}}$$

$$\underbrace{\hat{P} \quad A \quad A^{-1}}_{\hat{P}'} \quad \underbrace{\hat{S}}_{\hat{S}'}$$

- Closest rank-3 approximation  $\hat{M}$  yields Maximum Likelihood Estimate (assumption of isotropic mean-zero Gaussian noise).
- Affine ambiguity.

## Identify the estimated parameters



## Summary of the Algorithm

---

**Input:**  $n$  points in  $m$  images  $\mathbf{x}_j^i$ .

**Outputs:** 3D affine reconstruction defined by 3D points  $\mathbf{X}_j^i$  and cameras  $(\mathbf{P}^i, \mathbf{t}^i)$ .

- 1 Compute the **translation** vectors  $\mathbf{t}^i = \frac{1}{n} \sum_{j=1}^n \mathbf{x}_j^i$ .
- 2 **Center** the image points  $\mathbf{m}_j^i = \mathbf{x}_j^i - \mathbf{t}^i$ .
- 3 Form and factorize the centered **measurement matrix** using SVD:  
 $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ .
- 4 Extract the joint **projection matrix**  $\hat{\mathbf{P}}$  as the first 3 columns of  $\mathbf{U}$ .
- 5 Extract the joint **structure matrix**  $\hat{\mathbf{S}}$  as the first 3 rows of  $\mathbf{V}^\top$ .
- 6 Extract the individual projection matrices  $\mathbf{P}^i$  from  $\hat{\mathbf{P}}$  and the individual 3D points  $\mathbf{X}_j$  from  $\hat{\mathbf{S}}$ .



# Outline

---

## ① Structure from Motion (SFM) with Uncalibrated Cameras

### Non Euclidean SFM Batch Algorithms

- Affine Factorization

- Orthographic Factorization

- Perspective Factorization

- Sequential Algorithms

- Hierarchical Algorithms

## ② Applications

## Orthographic Factorization

---

[Tomasi Kanade, 1992] method first estimates the affine structure and then **upgrades it to Euclidean**. It operates on a batch of at least 3 frames with known correspondences (i.e., with tracked features).

- Factorize  $\mathbf{M}$  through singular value decomposition:

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

- The **affine** reconstruction can be computed as:

$$\tilde{\mathbf{P}} = \mathbf{U}, \quad \tilde{\mathbf{S}} = \mathbf{V}^T$$

- Upgrade to a metric reconstruction, by finding an invertible matrix  $\mathbf{A}$ , such that:

$$\hat{\mathbf{P}} = \tilde{\mathbf{P}}\mathbf{A}^{-1}, \quad \hat{\mathbf{S}} = \mathbf{A}\tilde{\mathbf{S}}$$

## Orthographic Factorization

---

To find  $\mathbf{A}$  additionally consider metric constraints under an orthographic camera model:

$$\mathcal{P}_{3 \times 4}^i = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1/f & f \end{bmatrix}_{3 \times 4} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}_{4 \times 4}$$

- This model appears as the limit of the general perspective projection as the focal length  $f$  becomes large with respect to the distance of the camera to the object.

## Orthographic Factorization

---

To find  $\mathbf{A}$  additionally consider metric constraints under an orthographic camera model:

$$\mathcal{P}_{3 \times 4}^i = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{3 \times 4} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}_{4 \times 4}$$

- This model appears as the limit of the general perspective projection as the focal length  $f$  becomes large with respect to the distance of the camera to the object.

## Orthographic Factorization

To find  $\mathbf{A}$  additionally consider metric constraints under an orthographic camera model:

$$\begin{aligned}
 \mathcal{P}_{3 \times 4}^i &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{3 \times 4} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}_{4 \times 4} \\
 &= \begin{bmatrix} \mathbf{r}_1^{i\top} & t_1^i \\ \mathbf{r}_2^{i\top} & t_2^i \\ \mathbf{0}^\top & 1 \end{bmatrix}_{3 \times 4}
 \end{aligned}$$

- This model appears as the limit of the general perspective projection as the focal length  $f$  becomes large with respect to the distance of the camera to the object.
- 5 d.o.f.s (3 due to the rotation and 2 due to the translation)

## Orthographic Factorization

---

$$\hat{\mathbf{P}}^i = \begin{bmatrix} \hat{\mathbf{p}}_1^{i\top} \\ \hat{\mathbf{p}}_2^{i\top} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^{i\top} & 0 \\ \mathbf{r}_2^{i\top} & 0 \end{bmatrix}$$

Remove the translation as in the affine case. Then, the orthographic projection matrix is characterized by two orthonormal vectors  $\mathbf{r}_1^i$  and  $\mathbf{r}_2^i$ :

$$\begin{aligned} \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_1^i &= 1 \\ \hat{\mathbf{p}}_2^{i\top} \hat{\mathbf{p}}_2^i &= 1 \quad \rightarrow \\ \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_2^i &= 0 \end{aligned}$$

## Orthographic Factorization

$$\hat{\mathbf{P}}^i = \begin{bmatrix} \hat{\mathbf{p}}_1^{i\top} \\ \hat{\mathbf{p}}_2^{i\top} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^{i\top} & 0 \\ \mathbf{r}_2^{i\top} & 0 \end{bmatrix}$$

Remove the translation as in the affine case. Then, the orthographic projection matrix is characterized by two orthonormal vectors  $\mathbf{r}_1^i$  and  $\mathbf{r}_2^i$ :

$$\begin{aligned} \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_1^i &= 1 \\ \hat{\mathbf{p}}_2^{i\top} \hat{\mathbf{p}}_2^i &= 1 \\ \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_2^i &= 0 \end{aligned} \rightarrow$$

$$\begin{aligned} \tilde{\mathbf{p}}_1^{i\top} \mathbf{A}^{-1} \mathbf{A}^{-\top} \tilde{\mathbf{p}}_1^i &= 1 \\ \tilde{\mathbf{p}}_2^{i\top} \mathbf{A}^{-1} \mathbf{A}^{-\top} \tilde{\mathbf{p}}_2^i &= 1 \\ \tilde{\mathbf{p}}_1^{i\top} \mathbf{A}^{-1} \mathbf{A}^{-\top} \tilde{\mathbf{p}}_2^i &= 0 \end{aligned}$$

Remember that  
 $\hat{\mathbf{P}} = \tilde{\mathbf{P}} \mathbf{A}^{-1}$

## Orthographic Factorization

$$\hat{\mathbf{P}}^i = \begin{bmatrix} \hat{\mathbf{p}}_1^{i\top} \\ \hat{\mathbf{p}}_2^{i\top} \end{bmatrix} = \begin{bmatrix} \mathbf{r}_1^{i\top} & 0 \\ \mathbf{r}_2^{i\top} & 0 \end{bmatrix}$$

Remove the translation as in the affine case. Then, the orthographic projection matrix is characterized by two orthonormal vectors  $\mathbf{r}_1^i$  and  $\mathbf{r}_2^i$ :

$$\begin{array}{l} \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_1^i = 1 \\ \hat{\mathbf{p}}_2^{i\top} \hat{\mathbf{p}}_2^i = 1 \\ \hat{\mathbf{p}}_1^{i\top} \hat{\mathbf{p}}_2^i = 0 \end{array} \rightarrow \begin{array}{l} \tilde{\mathbf{p}}_1^{i\top} \quad \mathbf{C} \quad \tilde{\mathbf{p}}_1^i = 1 \\ \tilde{\mathbf{p}}_2^{i\top} \quad \mathbf{C} \quad \tilde{\mathbf{p}}_2^i = 1 \\ \tilde{\mathbf{p}}_1^{i\top} \quad \mathbf{C} \quad \tilde{\mathbf{p}}_2^i = 0 \end{array}$$



## Orthographic Factorization

---

$$\begin{array}{l} \tilde{\mathbf{p}}_1^{i\top} \mathbf{C} \tilde{\mathbf{p}}_1^i = 1 \\ \tilde{\mathbf{p}}_2^{i\top} \mathbf{C} \tilde{\mathbf{p}}_2^i = 1 \\ \tilde{\mathbf{p}}_1^{i\top} \mathbf{C} \tilde{\mathbf{p}}_2^i = 0 \end{array}$$

- The orthographic constraints lead to **3** linear equations on symmetric matrix **C** (6DoF)
- **A** can be obtained from **C** through Cholesky factorization and inversion.
  - The Cholesky factorization decomposes of a symmetric, positive-definite matrix into the product of a lower triangular matrix and its conjugate transpose  $\mathbf{C} = \mathbf{L}\mathbf{L}^\top$
  - $\mathbf{A} = \mathbf{L}^{-1}$

# Outline

---

## 1 Structure from Motion (SFM) with Uncalibrated Cameras

### Non Euclidean SFM Batch Algorithms

- Affine Factorization

- Orthographic Factorization

- Perspective Factorization

- Sequential Algorithms

- Hierarchical Algorithms

## 2 Applications

## Perspective Factorization

---

According to perspective camera model (after centering the points):

$$\lambda_j^i \mathbf{m}_j^i = \mathbf{P}^i \mathbf{X}_j = \begin{bmatrix} \mathbf{p}_1^{i\top} \\ \mathbf{p}_2^{i\top} \\ \mathbf{p}_3^{i\top} \end{bmatrix}_{3 \times 4} \mathbf{X}_j$$

- Each  $\mathbf{P}^i$  is  $3 \times 4$
- $\mathbf{X}_j$  is a  $(4 \times 1)$  homogenous vector.
- $\mathbf{m}_j^i = \begin{bmatrix} x_j^i \\ y_j^i \\ 1 \end{bmatrix}$ , with  $x_j^i, y_j^i$  the centered image coordinates.
- $\lambda_j^i$  is a scale factor for each point: the “projective depth”.

## Perspective Factorization

$$\lambda_j^i \mathbf{m}_j^i = \mathbf{P}^i \mathbf{X}_j = \begin{bmatrix} \mathbf{P}_1^{iT} \\ \mathbf{P}_2^{iT} \\ \mathbf{P}_3^{iT} \end{bmatrix} \mathbf{X}_j$$

- The single matrix equation becomes:

$$\underbrace{\begin{pmatrix} \lambda_1^1 \mathbf{m}_1^1 & \lambda_2^1 \mathbf{m}_2^1 & \dots & \lambda_n^1 \mathbf{m}_n^1 \\ \lambda_1^2 \mathbf{m}_1^2 & \lambda_2^2 \mathbf{m}_2^2 & \dots & \lambda_n^2 \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{m}_1^m & \lambda_2^m \mathbf{m}_2^m & \dots & \lambda_n^m \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_n \end{pmatrix}}_{\mathbf{S}}$$

## Perspective Factorization

$$\underbrace{\begin{pmatrix} \lambda_1^1 \mathbf{m}_1^1 & \lambda_2^1 \mathbf{m}_2^1 & \dots & \lambda_n^1 \mathbf{m}_n^1 \\ \lambda_1^2 \mathbf{m}_1^2 & \lambda_2^2 \mathbf{m}_2^2 & \dots & \lambda_n^2 \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{m}_1^m & \lambda_2^m \mathbf{m}_2^m & \dots & \lambda_n^m \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\left( \mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_n \right)}_{\mathbf{S}}$$

Can we still use factorization?

## Perspective Factorization

$$\underbrace{\begin{pmatrix} \lambda_1^1 \mathbf{m}_1^1 & \lambda_2^1 \mathbf{m}_2^1 & \dots & \lambda_n^1 \mathbf{m}_n^1 \\ \lambda_1^2 \mathbf{m}_1^2 & \lambda_2^2 \mathbf{m}_2^2 & \dots & \lambda_n^2 \mathbf{m}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m \mathbf{m}_1^m & \lambda_2^m \mathbf{m}_2^m & \dots & \lambda_n^m \mathbf{m}_n^m \end{pmatrix}}_{\mathbf{M}} = \underbrace{\begin{pmatrix} \mathbf{P}^1 \\ \mathbf{P}^2 \\ \vdots \\ \mathbf{P}^m \end{pmatrix}}_{\mathbf{P}} \underbrace{\left( \mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_n \right)}_{\mathbf{S}}$$

- Collect measurements in one image  $\mathbf{M}^i = [\mathbf{m}_1^i \quad \mathbf{m}_2^i \quad \dots \quad \mathbf{m}_n^i]$
- Collect projective depths per camera  $\Lambda^i = \text{diag}(\lambda_1^i, \lambda_2^i, \dots, \lambda_n^i)$ .

$$\begin{pmatrix} (\mathbf{M}^1) & (\Lambda^1) \\ (\mathbf{M}^2) & (\Lambda^2) \\ \vdots & \vdots \\ (\mathbf{M}^m) & (\Lambda^m) \end{pmatrix} = \underbrace{\begin{pmatrix} (\mathbf{P}^1) \\ (\mathbf{P}^2) \\ \vdots \\ (\mathbf{P}^m) \end{pmatrix}}_{\mathbf{P}} \underbrace{\left( \mathbf{X}_1 \quad \mathbf{X}_2 \quad \dots \quad \mathbf{X}_n \right)}_{\mathbf{S}}$$

## Perspective Factorization

$$\begin{array}{rcl}
 \mathbf{M}^1 \Lambda^1 & = & \mathbf{P}^1 \mathbf{S} \\
 \mathbf{M}^2 \Lambda^2 & = & \mathbf{P}^2 \mathbf{S} \\
 \vdots & & \vdots \\
 \mathbf{M}^m \Lambda^m & = & \mathbf{P}^m \mathbf{S}
 \end{array}$$

- $\mathbf{M}^i$  are known, but  $\Lambda^i, \mathbf{P}^i$  and  $\mathbf{S}$  are unknown.
- $\mathbf{P}\mathbf{S}$  has rank 4: product of a  $3m \times 4$  and a  $4 \times n$  matrices .
- Assume that  $\Lambda^i$  are known, then  $\mathbf{P}\mathbf{S}$  can be computed.
  - Use the singular value decomposition:  $\mathbf{P}\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^\top$
  - In the noise-free case  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, 0, \dots, 0)$
  - A reconstruction is obtained by setting:
    - $\mathbf{P}$  = the first 4 columns of  $\mathbf{U}\Sigma$ .
    - $\mathbf{S}$  = the first 4 rows of  $\mathbf{V}^\top$ .

## Perspective Factorization

---

$$\begin{pmatrix} \mathbf{M}^1 \Lambda^1 \\ \mathbf{M}^2 \Lambda^2 \\ \vdots \\ \mathbf{M}^m \Lambda^m \end{pmatrix} = \mathbf{P} \mathbf{S}$$

- $\mathbf{M}^i$  are known, but  $\Lambda^i$ ,  $\mathbf{P}^i$  and  $\mathbf{S}$  are unknown.
- $\mathbf{P} \mathbf{S}$  has rank 4: product of a  $3m \times 4$  and a  $4 \times n$  matrices .
- Assume that  $\Lambda^i$  are known, then  $\mathbf{P} \mathbf{S}$  can be computed.
  - Use the singular value decomposition:  $\mathbf{P} \mathbf{S} = \mathbf{U} \Sigma \mathbf{V}^\top$
  - In the noise-free case  $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3, \sigma_4, 0, \dots, 0)$
  - A reconstruction is obtained by setting:
    - $\mathbf{P}$  = the first 4 columns of  $\mathbf{U} \Sigma$ .
    - $\mathbf{S}$  = the first 4 rows of  $\mathbf{V}^\top$ .



## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - 1 Normalize image data using isotropic scaling.

## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - ① Normalize image data using isotropic scaling.
  - ② Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).

## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - ① Normalize image data using isotropic scaling.
  - ② Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).
  - ③ Normalize the depths  $(\alpha^i \beta_j \lambda_j^i) \mathbf{m}_j^i = (\alpha^i \mathbf{P}^i)(\beta_j \mathbf{X}_j)$ .

The closer  $\lambda_j^i = 1$ , the error being minimized represents better the geometric distance. Heuristic:

    - multiply each row of  $\mathbf{P}^i$  by an  $\alpha^i$  so that it has unit norm.
    - Do the same with  $\beta_j$  and the columns of  $\mathbf{X}_j$

## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - 1 Normalize image data using isotropic scaling.
  - 2 Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).
  - 3 Normalize the depths  $(\alpha^i \beta_j \lambda_j^i) \mathbf{m}_j^i = (\alpha^i \mathbf{P}^i) (\beta_j \mathbf{X}_j)$ .
  - 4 Factorize  $\mathbf{M}$  and obtain an estimate of  $\mathbf{P}$  and  $\mathbf{S}$ .

## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - 1 Normalize image data using isotropic scaling.
  - 2 Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).
  - 3 Normalize the depths  $(\alpha^i \beta_j \lambda_j^i) \mathbf{m}_j^i = (\alpha^i \mathbf{P}^i) (\beta_j \mathbf{X}_j)$ .
  - 4 Factorize  $\mathbf{M}$  and obtain an estimate of  $\mathbf{P}$  and  $\mathbf{S}$ .
  - 5 If  $\sigma_5$  is sufficiently small then STOP.

## Iterative Perspective Factorization Algorithm

---

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - 1 Normalize image data using isotropic scaling.
  - 2 Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).
  - 3 Normalize the depths  $(\alpha^i \beta_j \lambda_j^i) \mathbf{m}_j^i = (\alpha^i \mathbf{P}^i)(\beta_j \mathbf{X}_j)$ .
  - 4 Factorize  $\mathbf{M}$  and obtain an estimate of  $\mathbf{P}$  and  $\mathbf{S}$ .
  - 5 If  $\sigma_5$  is sufficiently small then STOP.
  - 6 Use  $\mathbf{m}$ ,  $\mathbf{P}$  and  $\mathbf{S}$  to estimate  $\Lambda^i$  from the camera equations (linearly)  $\mathbf{m}^i \Lambda^i = \mathbf{P}^i \mathbf{S}$

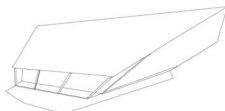
## Iterative Perspective Factorization Algorithm

- When  $\Lambda^i$  are unknown the following algorithm can be used:
  - ① Normalize image data using isotropic scaling.
  - ② Initialize  $\lambda_j^i = 1$  (e.g. set  $\lambda_j^i = 1$ , affine approximation).
  - ③ Normalize the depths  $(\alpha^i \beta_j \lambda_j^i) \mathbf{m}_j^i = (\alpha^i \mathbf{P}^i)(\beta_j \mathbf{X}_j)$ .
  - ④ Factorize  $\mathbf{M}$  and obtain an estimate of  $\mathbf{P}$  and  $\mathbf{S}$ .
  - ⑤ If  $\sigma_5$  is sufficiently small then STOP.
  - ⑥ Use  $\mathbf{m}$ ,  $\mathbf{P}$  and  $\mathbf{S}$  to estimate  $\Lambda^i$  from the camera equations (linearly)  $\mathbf{m}^i \Lambda^i = \mathbf{P}^i \mathbf{S}$
  - ⑦ GOTO 3.
- In general the algorithm minimizes a proximity measure  $\text{Prox}(\Lambda, \mathbf{P}, \mathbf{S}) = \sigma_5$ .
- Convergence is not guaranteed.
- Note that structure and motion are recovered up to an arbitrary projective transformation.

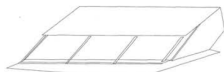
## Euclidean Update



Projective



Projective



Affine



Affine



Euclidean



Euclidean



Euclidean



Euclidean

- The affine and perspective reconstructions may be upgraded to a metric reconstruction by:
  - Supplying metric information on the scene.
  - Using auto-calibration methods.
  - A combination of the two.
  - Hartley Zisserman - Chap 10.



# Outline

---

## 1 Structure from Motion (SFM) with Uncalibrated Cameras

Non Euclidean SFM Batch Algorithms

Affine Factorization

Orthographic Factorization

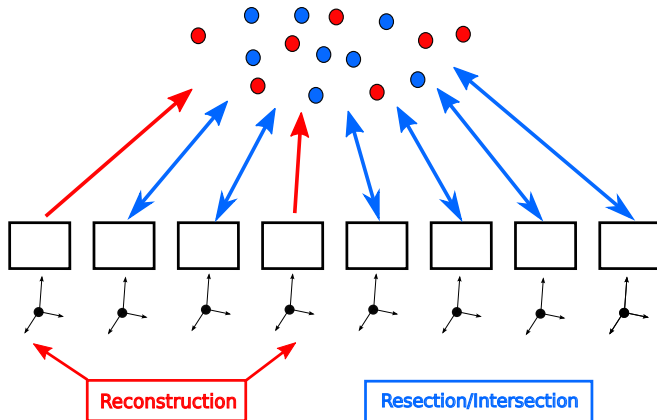
Perspective Factorization

**Sequential Algorithms**

Hierarchical Algorithms

## 2 Applications

## Sequential Algorithms



## Sequential Algorithms

---

- Register each view in turn.
- Pick a “small” number of views and compute a partial 3D model using e.g. factorization.
- Repeat until the 3D model is complete.
  - Pick an unregistered view and apply resection, i.e. compute the camera matrix. The view must share sufficiently many points with the partial 3D model.
  - Apply triangulation (intersection) to reconstruct new 3D features appearing in the latest registered view.

## Sequential Algorithms

---

- Pros**
- Robustness to blunders is easily incorporated
  - On-line processing
- Cons**
- The error in the initial partial 3D model conditions the final complete 3D model, i.e. it may make the 3D model drift away from the optimal one.
- Note:** If computational cost is not an important criteria, Bundle Adjustment is launched in the inner loop to prevent drifting.

# Outline

---

## 1 Structure from Motion (SFM) with Uncalibrated Cameras

Non Euclidean SFM Batch Algorithms

Affine Factorization

Orthographic Factorization

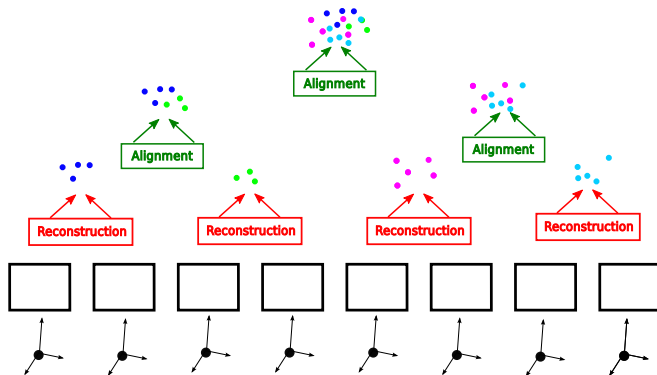
Perspective Factorization

Sequential Algorithms

Hierarchical Algorithms

## 2 Applications

# Hierarchical Algorithms



## Hierarchical Algorithms

---

- **Split** the image sequence into subsequences [Fitzgibbon et al., 1998 ; Nistr 2000].
- **Reconstruct** a partial 3D model from each subsequence using e.g. factorization.
- Hierarchically **merge** the partial 3D models (by computing a 3D homography, e.g. [Csurka et al., 1999]).
- This approach distributes the error over the entire sequence and can be easily made **robust**.

## Extensions

---

- Extension to **paraperspective** cameras [Poelman and Kanade, 1993].
- Other **features** (lines and conics) [Kahl and Heyden, 1998 ; Quan, 1997].
- Extension to **projective** cameras [Sturm and Triggs, 1996]. The projective depths of each image point are recovered before factorization and the algorithm does not longer minimize the reprojection error.
- Missing data [Jacobs, 1997 ; Martinec and Pajdla, 2002].
- Multibody [Costeira and Kanade, 1995].
- Deformable [Bregler et al., 2000].



## Recall: SFM Method Overview

---

- 1 Initialization: Compute a “good enough” sub-optimal estimate.
  - Batch.
  - Sequential.
  - Hierarchical algorithms.
- 2 **Bundle Adjustment** (Optional):
  - Non-linear minimization of the reprojection error (discrepancy between measured and predicted feature coordinates).
- 3 Self-Calibration.

## Problem Statement

- 3D Points  $\mathbf{X}_j$ , with
  - with  $j = \{1, \dots, n\}$
  - $n$  points.
- viewed by a set of cameras  $\mathbf{P}^i$ , with
  - $i = \{1, \dots, m\}$
  - $m$  views.

**Problem:** Given the projections  $\mathbf{x}_j^i$  of the  $n$  points on the  $m$  cameras, find:

- The set of camera matrices  $\hat{\mathbf{P}}^i$
- and the set of points  $\hat{\mathbf{X}}_j$

such that:

$$\hat{\mathbf{P}}^i \hat{\mathbf{X}}_j = \mathbf{x}_j^i$$

## Noisy Measurements

---

- If the image measurements are noisy, then the projection equation is not satisfied exactly.

$$\hat{\mathbf{P}}^i \hat{\mathbf{X}}_j \neq \mathbf{x}_j^i$$

- Use Maximum Likelihood estimation assuming noise is Gaussian.

## Maximum Likelihood

- Estimate projection matrices  $\hat{\mathbf{P}}^i$  and 3D points  $\hat{\mathbf{X}}_j$  which project **exactly** to corrected image points  $\hat{\mathbf{x}}_j^i$
- while minimizing the distance between the estimated  $\hat{\mathbf{x}}_j^i$  and the measured  $\mathbf{x}_j^i$  points for every view.

$$\min_{\hat{\mathbf{P}}^i, \hat{\mathbf{X}}_j} \sum_{ij} d(\hat{\mathbf{P}}^i \hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2$$

- where  $d(\mathbf{x}, \mathbf{y})$  is the geometric image distance between homogeneous points  $\mathbf{x}$  and  $\mathbf{y}$ .

## Bundle Adjustment

---

$$\min_{\hat{\mathbf{P}}^i, \hat{\mathbf{X}}_j} \sum_{ij} d(\hat{\mathbf{P}}^i \hat{\mathbf{X}}_j, \mathbf{x}_j^i)^2$$

$$\min_{\hat{\mathbf{P}}^i, \hat{\mathbf{X}}_j} \dots$$

Adjust the bundle of rays  
between each camera center and  
the set of 3D points.

$$\min_{\hat{\mathbf{P}}^i, \hat{\mathbf{X}}_j} \dots$$

Adjust the bundle of rays  
between each 3D point and the  
set of camera centers.

## Non-linear Least Squares

---

Assume a problem can be modeled with a parameterized function  $f$  (parameters  $\mathbf{p}$ ) such that:

$$\hat{\mathbf{x}} = f(\mathbf{p})$$

LSQ optimization seeks the parameters  $\mathbf{p}$  that minimize the difference between the measure  $\mathbf{x}$  and estimation  $\hat{\mathbf{x}}$ :

$$\arg \min_{\mathbf{p}} \|\mathbf{x} - f(\mathbf{p})\|$$

- Iterative optimization algorithms:
  - Newton.
  - Levenberg-Marquardt
  - Sparse Levenberg-Marquardt

## Number of parameters to optimize

---

- Each camera matrix  $\mathbf{P}^i$  has 11 DoFs.
- There are  $m$  views.
- Each point  $\mathbf{X}_j$  has 3 DoFs.
- There are  $n$  points.

There are  $3n + 11m$  parameters to optimize !!!!

(This is a lower bound, it can be more due to over-parameterization)

- e.g. when using Levenberg Marquardt, matrices of  $(3n + 11m) \times (3n + 11m)$  have to be factored or inverted, which is computationally expensive.

## Practical Implementation

---

- Initialization
  - Use factorization for initialization.
  - Use available information:  
e.g. coplanarity leads to a closed form solution.
- Tackle the size of the problem by:
  - Reducing the number of views  $m$  or the number of points  $n$ .
    - Do not include all the points or all the views. These can be then computed by resection or triangulation.
    - Partition the data, e.g. hierarchical approach.
  - Interleave: iterate between minimizing the projection error by:
    - varying the entries of the cameras  $\hat{\mathbf{P}}^i$ .
    - varying the entries of the points  $\hat{\mathbf{X}}^j$
  - Use sparse methods.



## Bundle Adjustment

---

Used as a final step of any reconstruction.

### Pros

- Provides ML estimate.
- Tolerant to missing data.
- Assigns covariances to each measure.
- Can be extended to include priors and constraints on camera parameters and point positions.

### Cons:

- It requires a good initialization.
- It can become an extremely large minimization problem.

## Recall

---

A  $3 \times 4$  projection matrix  $\mathbf{P}$  can be decomposed into

$$\mathbf{P} \propto \mathbf{K}[\mathbf{R} \quad \mathbf{t}]$$

$(\mathbf{R}; \mathbf{t})$  is the pose or extrinsic parameters

$\mathbf{K}$  is the calibration or intrinsic parameters

$$\mathbf{K} = \begin{pmatrix} \alpha f & sf & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{pmatrix}$$

$f$  is the focal length

$(u_0, v_0)$  is the principal point

$\alpha$  is the aspect ratio

$s$  is the skew

## Problem Statement

- The projective 3D model is defined up to a projective transformation

$$\mathbf{P}^i \mathbf{X}_j \propto \mathbf{P}^i \mathbf{H} \mathbf{H}^{-1} \mathbf{X}_j = \mathbf{P}^{i0} \mathbf{X}_j^0$$

where  $\mathbf{H}$  is a 3D ( $4 \times 4$ ) homography, with  $\det(\mathbf{H}) \neq 0$  and 15 parameters.

- The goal is to obtain a Euclidean 3D model defined up to a similarity (7 parameters).
- The number of unknowns is thus  $15 - 7 = 8$
- We are looking for a projective transformation  $\mathbf{Z}$  such that:
  - $\mathbf{P}^i \mathbf{Z}$  are projection matrices
  - $\mathbf{Z}^{-1} \mathbf{X}_j$  are 3D points

in a Euclidean coordinate frame.

## Internal Camera Calibration

---

The internal camera calibration can be computed from prior knowledge (Chapter 19):

- on the scene, e.g. right angles, equal lengths.
- on the camera motion, e.g. pure translation, stereo rig.
- on the intrinsic parameters themselves:
  - Assume they are all unknown and constant.
  - Varying focal length calibration [Triggs, 1997 ; Pollefeys et al., 1998].

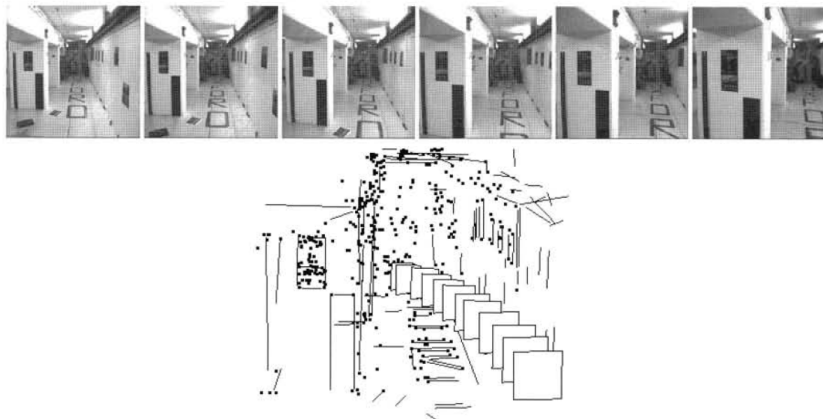
The estimated internal camera parameters allow us to compute a metric reconstruction from initially uncalibrated images

# Outline

---

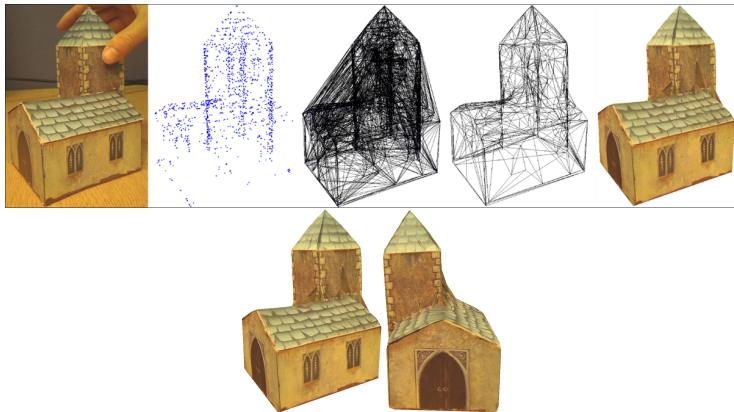
- 1 Structure from Motion (SFM) with Uncalibrated Cameras
  - Non Euclidean SFM Batch Algorithms
    - Affine Factorization
    - Orthographic Factorization
    - Perspective Factorization
  - Sequential Algorithms
  - Hierarchical Algorithms
- 2 Applications

## Example Applications



Automatic reconstruction of piecewise planar models from multiple views  
[Baillard, Zisserman-CVPR1999]

## Example Applications



Creating 3D models with a simple webcam [Pan, Reitmayr, Drummond - BMVC 2009]

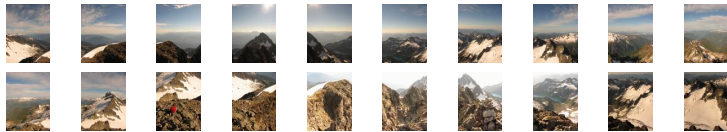
## Comercial Software

---

- 2d3 ([www.2d3.com](http://www.2d3.com))
  - Boujou (camera tracking)
  - SteadyMove (video stabilizer)
  - ...
- Realviz (bought by Autodesk)
  - MatchMover (camera tracking)
  - Stitcher (image mosaicing)
  - ImageModeler (computer-aided 3D models from images)
  - ...
- Google Street view
- Microsoft Fototurism



# Autostitch



<http://people.cs.ubc.ca/~mbrown/autostitch/autostitch.html>

## Street View



[http://www.talkmunich.com/services/munich\\_map.php](http://www.talkmunich.com/services/munich_map.php)

# Bundler

## **Bundler: Structure from Motion for Unordered Image Collections**



# Photo Tourism



## Photo Tourism

Exploring photo collections in 3D

**Microsoft**



(a)



(b)



(c)

# Photosynth



<http://photosynth.net/>

# Outline

---

## 1 Structure from Motion (SFM) with Uncalibrated Cameras

- Non Euclidean SFM Batch Algorithms

  - Affine Factorization

  - Orthographic Factorization

  - Perspective Factorization

- Sequential Algorithms

- Hierarchical Algorithms

## 2 Applications