AARHUS
UNIVERSITET

Computer Vision & Machine Learning

Alexandros Iosifidis
@
Department of Electrical and Computer Engineering
Aarhus University

This week

More on epipolar geometry

Recovering structure:

- Triangulation: estimating 3D positions from matched points in multiple images
- Structure from motion: estimating 3D positions from matched points from multiple images of a moving camera
- Structure from light: use shades to extract shape

Image alignment and warping

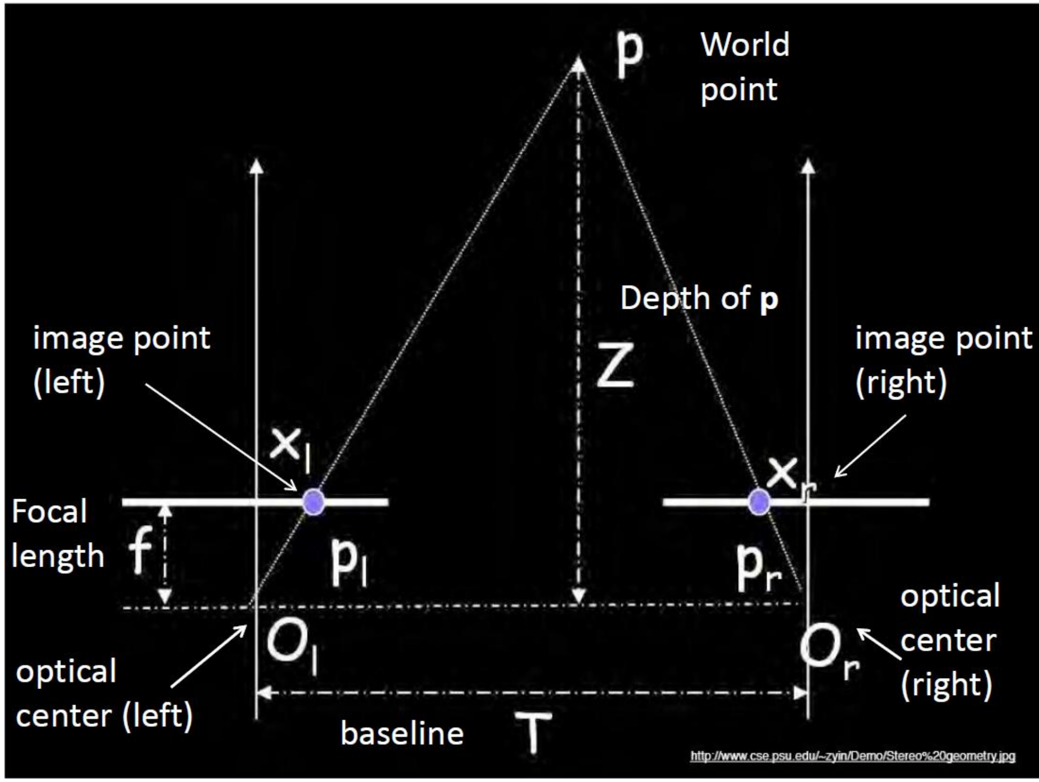
Slides based on:

- Richard Szeliski: CV: Algorithms & Applications
- Epipolar Geometry & the Fundamental Matrix

Stereo vision

Stereo cameras:

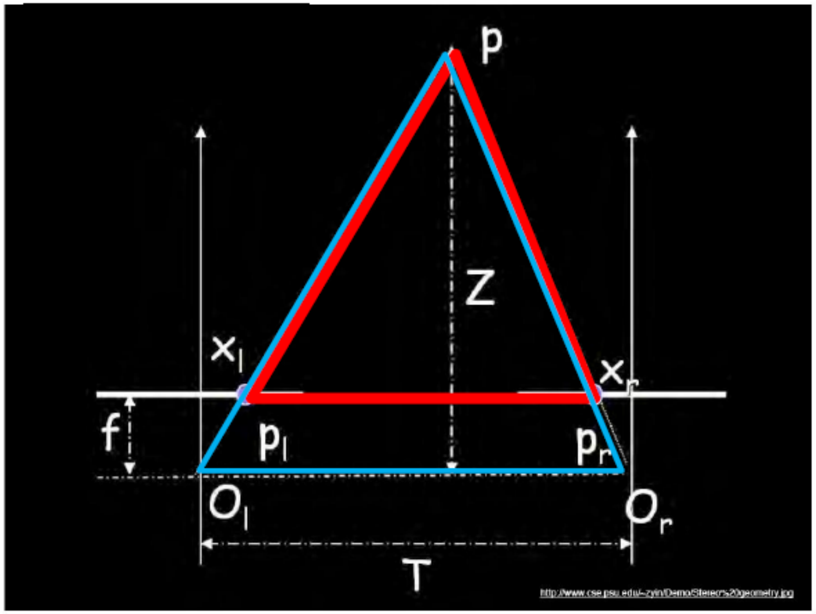
- Formed by two cameras, placed one next to the other (parallel optical axes)
- Information regarding (relative) depth of objects in the scene based on triangulation



Stereo vision

Stereo cameras:

- Formed by two cameras, placed one next to the other (parallel optical axes)
- Information regarding (relative) depth of objects in the scene based on triangulation



Similar triangles (p_l, P, p_r) and (O_l, P, O_r):

$$\frac{T + x_l - x_r}{Z - f} = \frac{T}{Z}$$

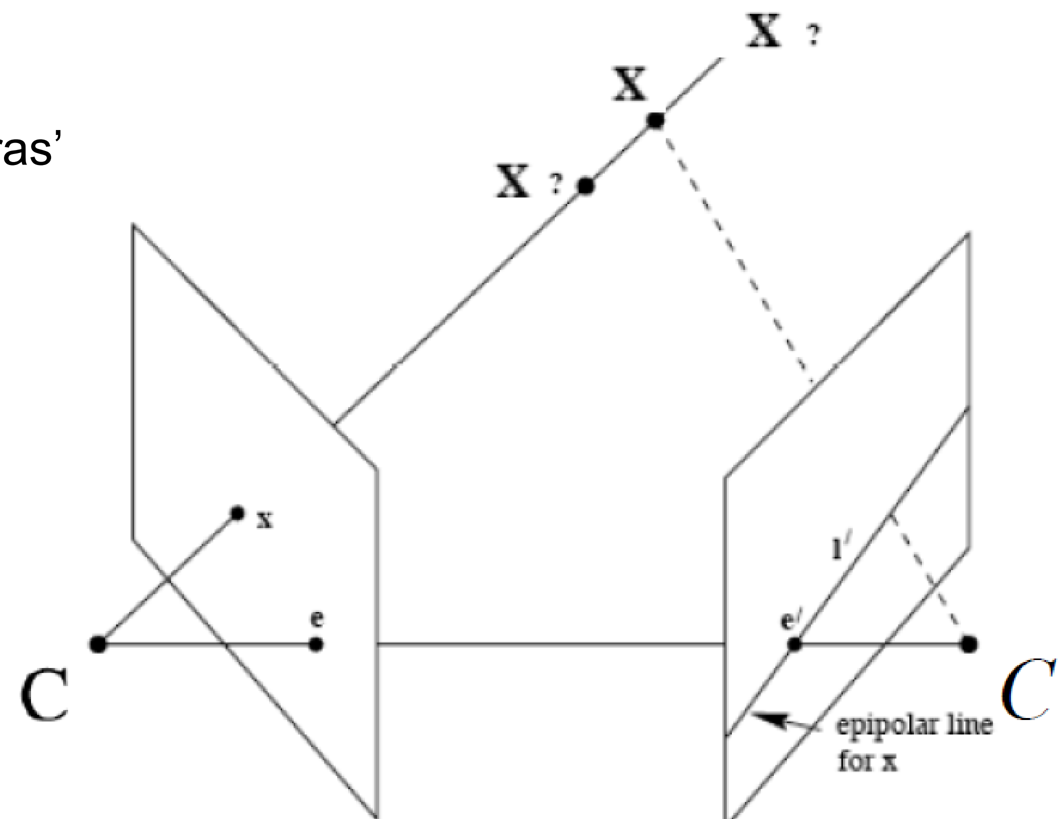
$$Z = f\frac{T}{x_r - x_l}$$

disparity

Epipoles

The point x in left image corresponds to a line (called epipolar line) l’ in the image on the right

Epipolar line passes through the epipole **e** (the intersection of cameras’ baseline with the image plane)



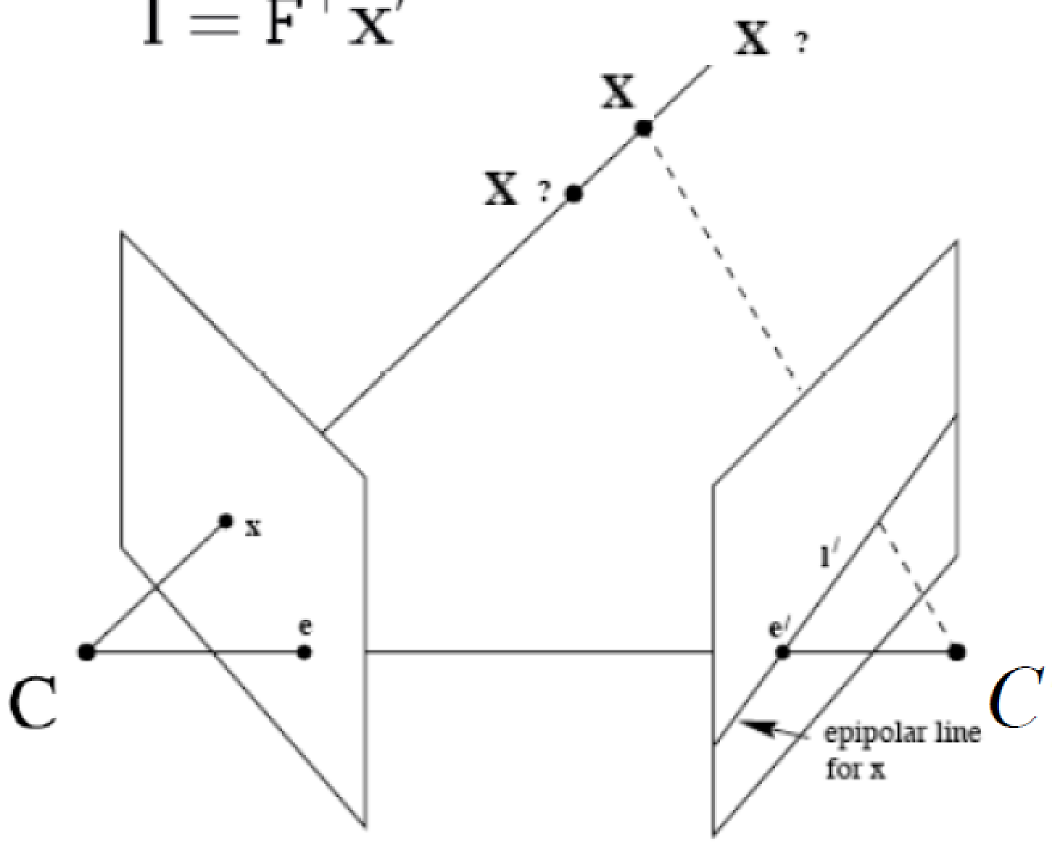
Fundamental Matrix

The Fundamental matrix performs a mapping from a point in one image to a line in the other

$$\mathbf{l}' = \mathbf{F}\mathbf{x} \qquad \mathbf{l} = \mathbf{F}^\top \mathbf{x}'$$

If **x** and **x'** correspond to the same 3D point **X**:

$$\mathbf{x}'^\top \mathbf{F}\mathbf{x} = 0$$



Fundamental Matrix

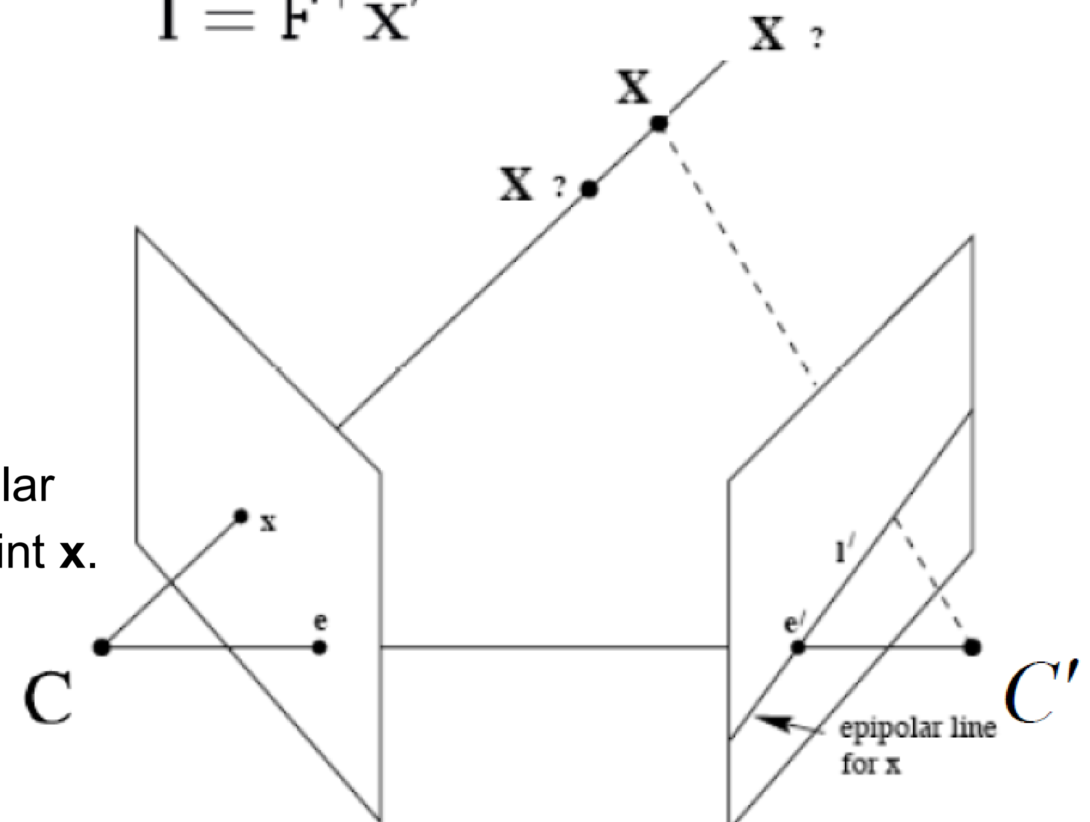
The Fundamental matrix performs a mapping from a point in one image to a line in the other

$$\mathbf{l}' = \mathbf{F}\mathbf{x} \qquad \mathbf{l} = \mathbf{F}^\top \mathbf{x}'$$

If **x** and **x**' correspond to the same 3D point **X**:

$$\mathbf{x}'^\top \mathbf{F}\mathbf{x} = 0$$

This is because **x**' lies on the epipolar line **l**' = **Fx** corresponding to the point **x**. Thus, 0 = **x**'^{T}**l**' = x'^{T}Fx



Computing the Fundamental matrix

The matrix **F** can be computed using (multiple) point correspondences between two images:

- Each point correspondence creates one constraint on **F**

$$\overline{\mathbf{p}}_{right}^{\mathrm{T}}\mathbf{F}\overline{\mathbf{p}}_{left} = 0 \quad \Rightarrow \quad \begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0$$

- We collect n correspondences
- We solve for **F** using least squares (eigenvector trick)

$$\begin{bmatrix} u_1'u_1 & u_1'v_1 & u_1' & v_1'u_1 & v_1'v_1 & v_1' & u_1 & v_1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

From Fundamental to Essential matrix

If the cameras are calibrated (we know the calibration matrices **K** and **K**'):

$$F = K^{-T} E K'^{-1}$$
$$E = K^{T} F K'$$

From Fundamental to Essential matrix

If the cameras are calibrated (we know the calibration matrices **K** and **K**'):

$$F = K^{-T} E K'^{-1}$$
$$E = K^{T} F K'$$

Also we have $\mathbf{E} = \mathbf{T}_x \mathbf{R}$

where **R** is the rotation matrix and
t is the translation vector

From Fundamental to Essential matrix

If the cameras are calibrated (we know the calibration matrices **K** and **K'**):

$$F = K^{-T} E K'^{-1}$$
$$E = K^{T} F K'$$

Also we have $\mathbf{E} = \mathbf{T}_x \mathbf{R}$

where **R** is the rotation matrix and **t** is the translation vector

T is the skew-symmetric matrix of **t**

Cross products

Of particular interest are 3×3 skew-symmetric matrices. If $\mathbf{a} = (a_1, a_2, a_3)^{\mathsf{T}}$ is a 3-vector, then one defines a corresponding skew-symmetric matrix as follows:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}.$$

Note that any skew-symmetric 3×3 matrix may be written in the form $[\mathbf{a}]_\times$ for a suitable vector $\mathbf{a}$. Matrix $[\mathbf{a}]_\times$ is singular, and $\mathbf{a}$ is its null-vector (right or left). Hence, a 3×3 skew-symmetric matrix is defined up to scale by its null-vector.

The cross product (or vector product) of two 3-vectors $\mathbf{a} \times \mathbf{b}$ (sometimes written $\mathbf{a} \wedge \mathbf{b}$) is the vector $(a_2 b_3 - a_3 b_2, a_3 b_1 - a_1 b_3, a_1 b_2 - a_2 b_1)^{\mathsf{T}}$. The cross product is related to skew-symmetric matrices according to

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = \left(\mathbf{a}^{\mathsf{T}} [\mathbf{b}]_\times\right)^{\mathsf{T}}.$$

From Fundamental to Essential matrix

If the cameras are calibrated (we know the calibration matrices **K** and **K**'):

$$F = K^{-T} E K'^{-1}$$
$$E = K^{T} F K'$$

Also we have $\mathbf{E} = \mathbf{T}_x \mathbf{R}$

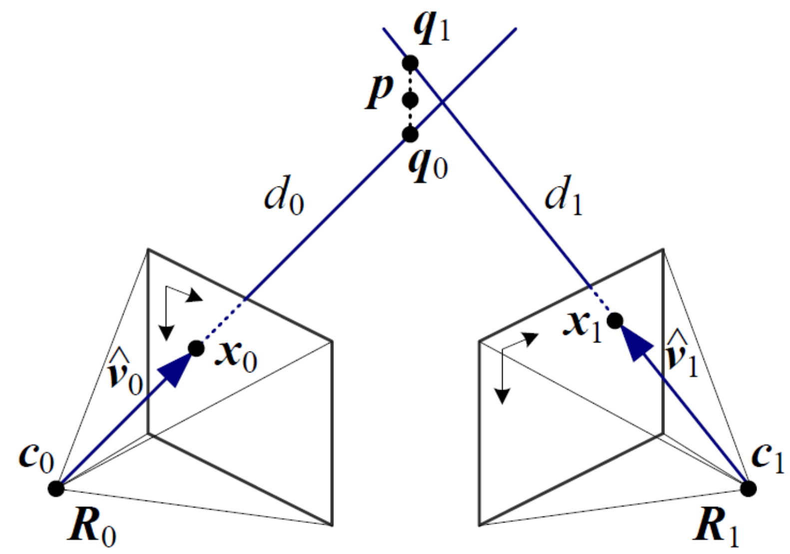
If we decompose **E** using SVD decomposition $E = U \Sigma V^T$

then $R = UWV^T$ or $R = UW^T V^T$, where $W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

and t$= u_3$ or t$= -u_3$

Triangulation

Generally, the two rays C → **x** and C' → **x**' will not exactly intersect
- We solve using SVD (finding a least squares solution to a system of equations)



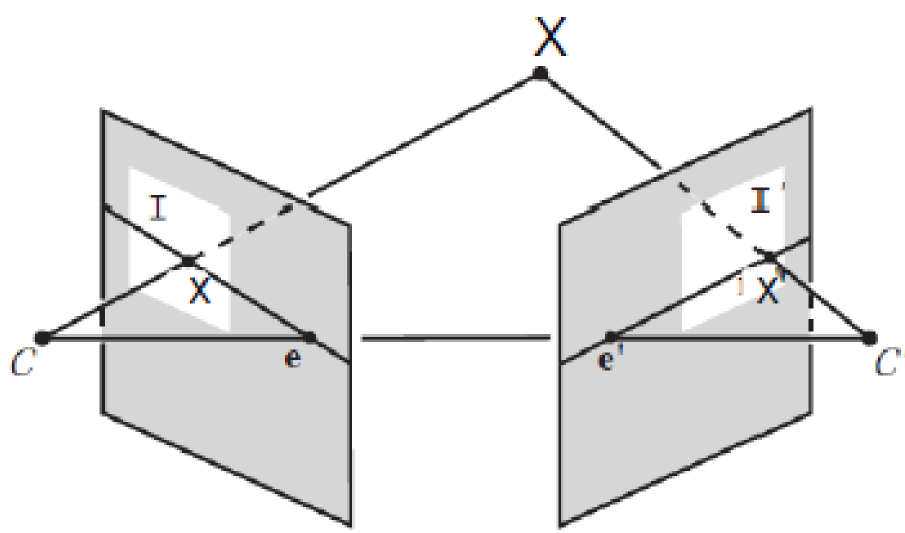
Triangulation

Generally, the two rays C → **x** and C' → **x**' will not exactly intersect

- We solve using SVD (finding a least squares solution to a system of equations)

$$\mathbf{x} \times (\mathbf{P}\mathbf{X}) = 0 \qquad \mathbf{x}' \times (\mathbf{P}'\mathbf{X}) = 0$$

$$\mathbf{A}\mathbf{X} = \mathbf{0} \qquad \mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$



Triangulation

Given **P**, **P'**, **x**, **x'**

1. Precondition points and projection matrices
2. Create matrix **A**
3. [U,S,V] = svd(**A**)
4. **X** = V(:,end)

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \mathbf{x}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \mathbf{p}_3^T \end{bmatrix} \qquad \mathbf{P}' = \begin{bmatrix} \mathbf{p}_1'^T \\ \mathbf{p}_2'^T \\ \mathbf{p}_3'^T \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} u\mathbf{p}_3^T - \mathbf{p}_1^T \\ v\mathbf{p}_3^T - \mathbf{p}_2^T \\ u'\mathbf{p}_3'^T - \mathbf{p}_1'^T \\ v'\mathbf{p}_3'^T - \mathbf{p}_2'^T \end{bmatrix}$$

Pros and Cons

- Works for any number of corresponding images
- Not projectively invariant

Multi-view geometry

Questions:

- Scene geometry (structure): Given 2D point matches in two or more images, where are the corresponding points in 3D?

- Correspondence (stereo matching): Given a point in just one image, how does it constrain the position of the corresponding point in another image?

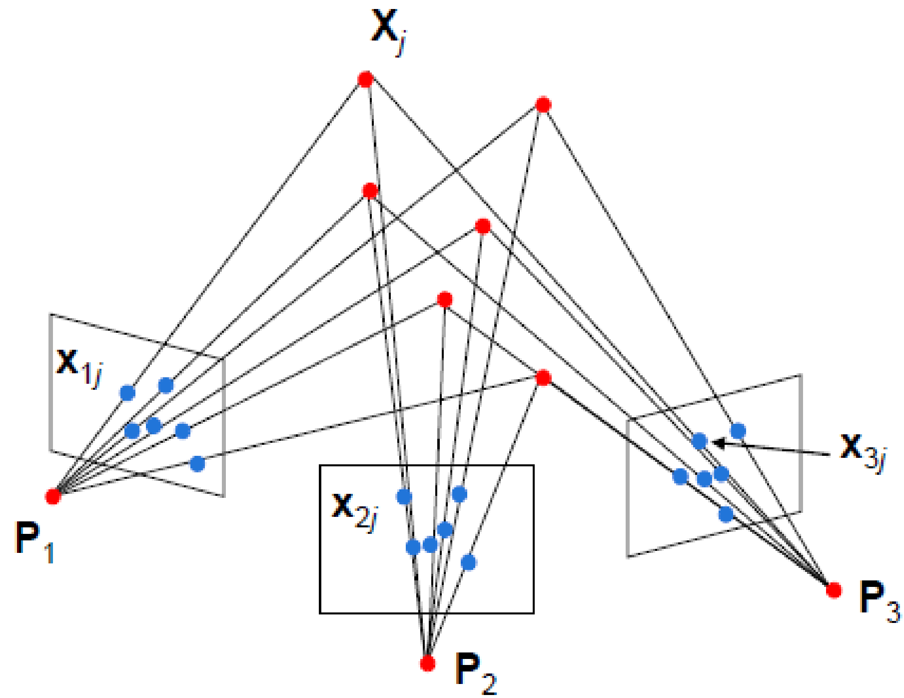
- Camera geometry (motion): Given a set of corresponding points in two or more images, what are the camera matrices for these views?

Structure from motion

Given m images of n fixed 3D points

$$\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

estimate m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$



Structure from motion

Ambiguity:

- If we scale the entire scene by some factor k and, at the same time, scale the camera matrices by the factor of 1/k, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\mathbf{X})$$

- It is impossible to recover the absolute scale of the scene.

Structure from motion

Ambiguity:

- If we scale the entire scene by some factor k and, at the same time, scale the camera matrices by the factor of 1/k, the projections of the scene points in the image remain exactly the same:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\mathbf{X})$$

It is impossible to recover the absolute scale of the scene

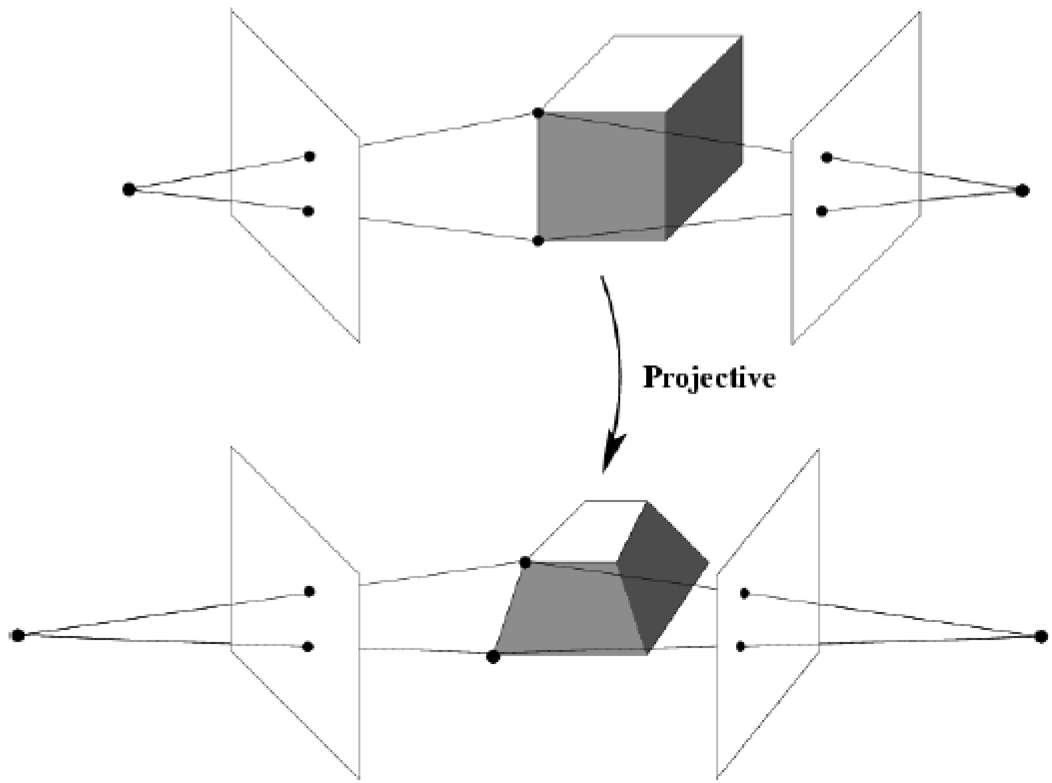
- More generally, if we transform the scene using a transformation Q and apply the inverse transformation to the camera matrices, then the images do not change

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

Structure from motion

Projective ambiguity:

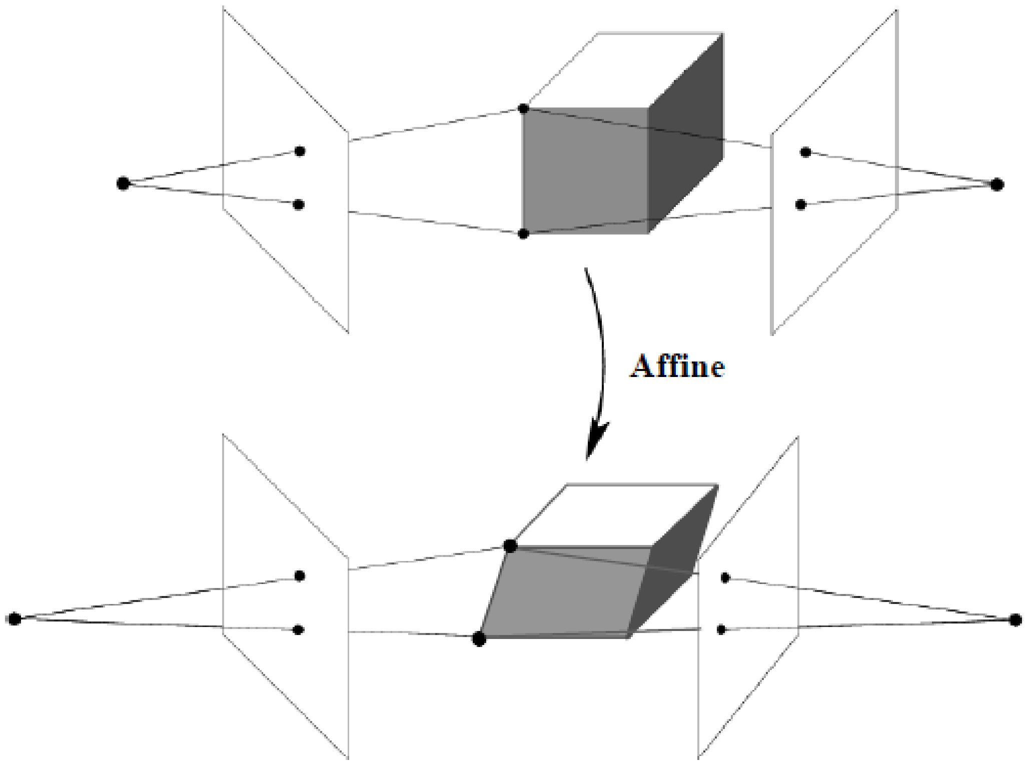
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$



Structure from motion

Affine ambiguity:

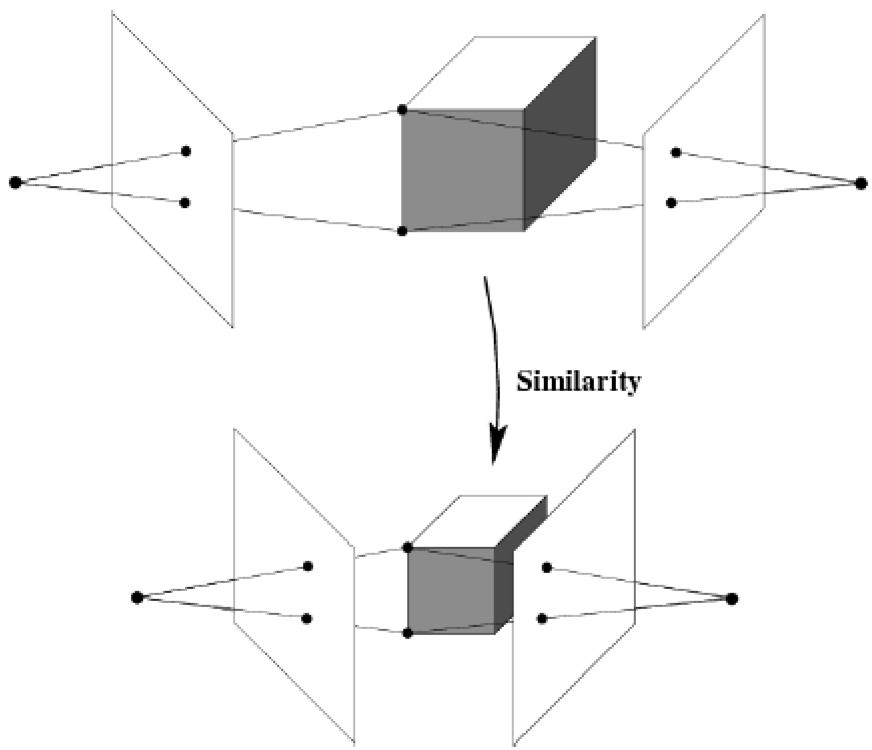
$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$



Structure from motion

Similarity ambiguity:

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)\left(\mathbf{Q}\mathbf{X}\right)$$

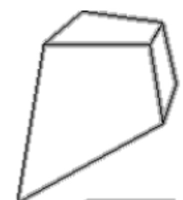
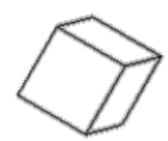
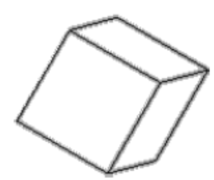


3D transformations

With no constraints on the camera calibration matrix or on the scene, we get the projective reconstruction

We need additional information in order to obtain an affine or Euclidean similarity

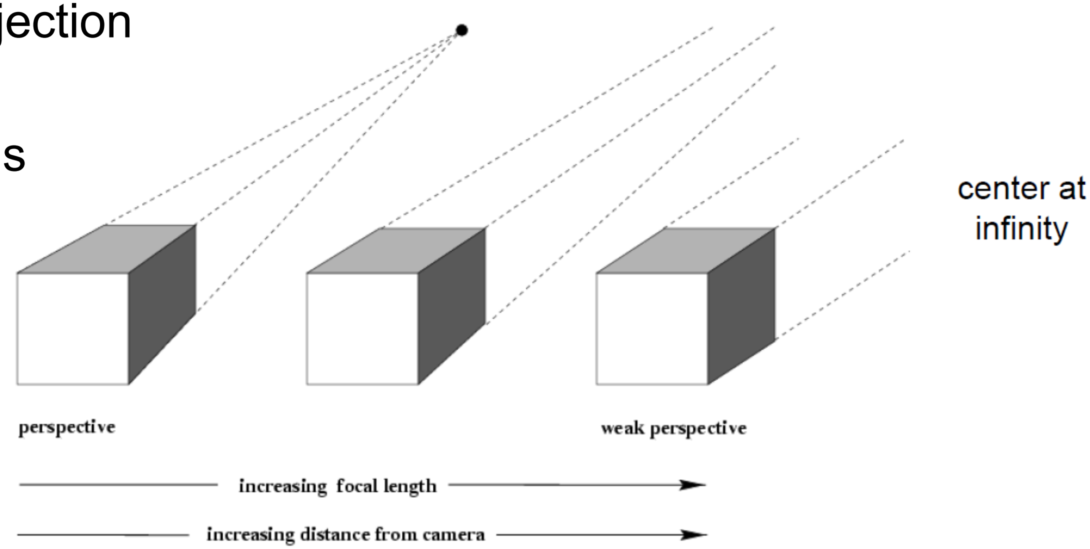
Degrees of freedom

Projective 15dof	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection and tangency
Affine 12dof	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism, volume ratios
Similarity 7dof	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

Structure from motion

We need to choose the type of projection (cameras) we will use:

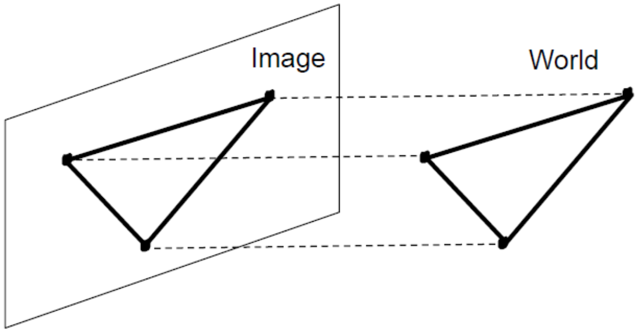
- Affine cameras for simpler models



Structure from motion

<u>Orthographic projection:</u>

- Special case of perspective projection
- Distance from center of projection to image plane is infinite



- Projection matrix has the form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

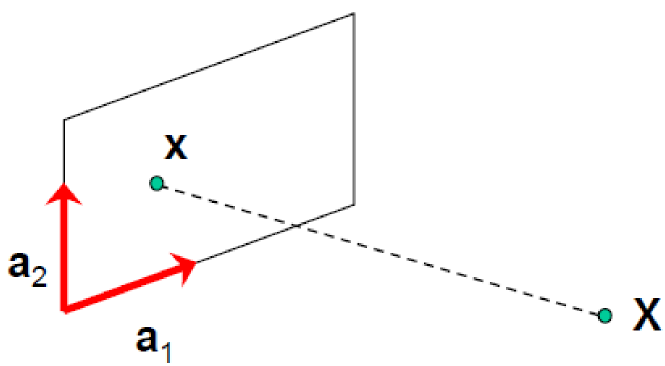
Structure from motion

Affine cameras:

- Combine the effects of an affine transformation of the 3D space, orthographic projection and an affine transformation of the image:

$$\mathbf{P} = [3\times 3\,\text{affine}]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}[4\times 4\,\text{affine}] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

- Affine projection is a linear mapping plus a translation in inhomogeneous coordinates



$$\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \mathbf{AX} + \mathbf{b}$$

Projection of the world origin

Structure from motion

Affine structure from motion:

- Given m images of n 3D points
- Use the mn correspondences $\mathbf{x}_{ij}$ to estimate the m projection matrices $\mathbf{A}_i$, the translation vectors $\mathbf{b}_i$ and n points $\mathbf{X}_j$

$$\mathbf{x}_{ij} = \mathbf{A}_i \mathbf{X}_j + \mathbf{b}_i, \quad i = 1, \dots, m, \; j = 1, \dots, n$$

- The reconstruction is defined up to an arbitrary affine transformation $\mathbf{Q}$

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \rightarrow \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{Q}^{-1}, \qquad \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix} \rightarrow \mathbf{Q} \begin{pmatrix} \mathbf{X} \\ 1 \end{pmatrix}$$

- Number of unknowns is 8m+3n – 12 (degrees of freedom is equal to 12)
- Number of knowns is 2mn
- To solve such a linear system we need 2mn ≥ 8m+3n-12
- For two views, we need n = 4

Structure from motion

<u>Affine structure from motion:</u>

In order to reduce the number of parameters (translation vector **b**)

- We place the origin of the world coordinate system at the centroid of the 3D points
- (Centering) subtract the centroid of the image points

$$\hat{\mathbf{x}}_{ij} = \mathbf{x}_{ij} - \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_{ik} = \mathbf{A}_i\mathbf{X}_j + \mathbf{b}_i - \frac{1}{n}\sum_{k=1}^{n}(\mathbf{A}_i\mathbf{X}_k + \mathbf{b}_i)$$

$$= \mathbf{A}_i\left(\mathbf{X}_j - \frac{1}{n}\sum_{k=1}^{n}\mathbf{X}_k\right) = \mathbf{A}_i\hat{\mathbf{X}}_j$$

- After centering, each (centered) point $\mathbf{x}_{ij}$ is related to the 3D point $\mathbf{X}_j$ by

$$\hat{\mathbf{x}}_{ij} = \mathbf{A}_i\mathbf{X}_j$$

Structure from motion

<u>Affine structure from motion:</u>

- Create a matrix **D** of 2mn data points (measurements)

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}$$

cameras ($2m$)

points (n)

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992

Structure from motion

Affine structure from motion:

- Create a matrix **D** of 2mn data points (measurements)

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}}_{\substack{\text{cameras} \\ (2m \times 3)}} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (3 \times n)}$$

- The matrix **D** = **MS** must be of rank 3

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992

Structure from motion

<u>Affine structure from motion:</u>

- Create a matrix **D** of 2mn data points (measurements)

$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}}_{\text{cameras } (2m \times 3)} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (3 \times n)}$$

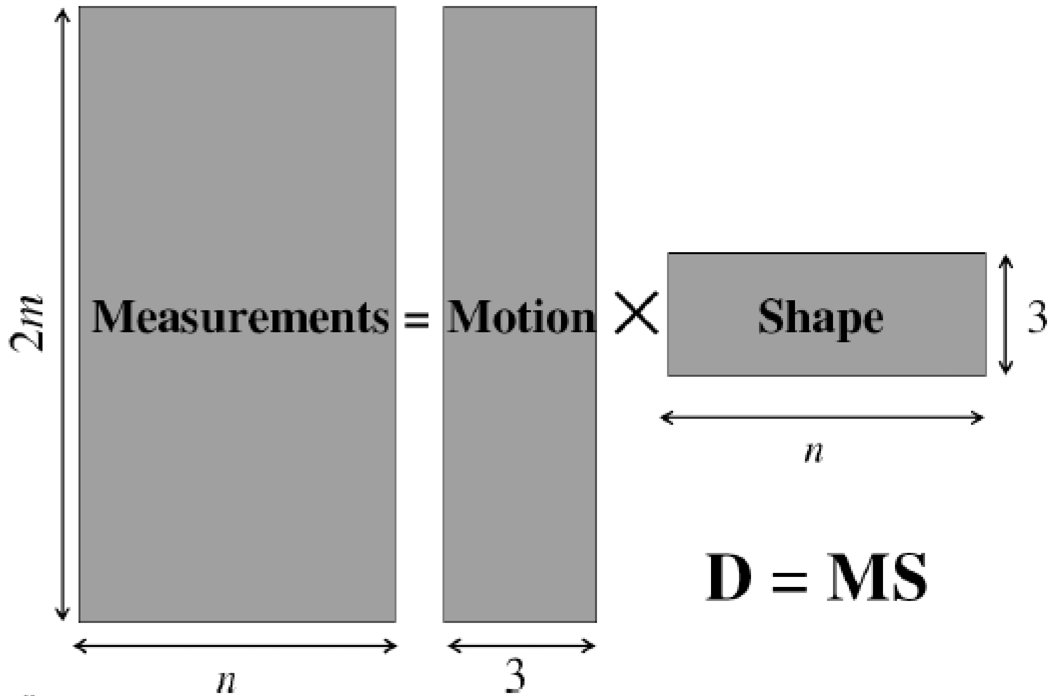
- The matrix **D** = **MS** must be of rank 3
- We use the above structure in order to obtain matrices **M** and **S**

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992

Structure from motion

Affine structure from motion:

- Factorization of the matrix **D**

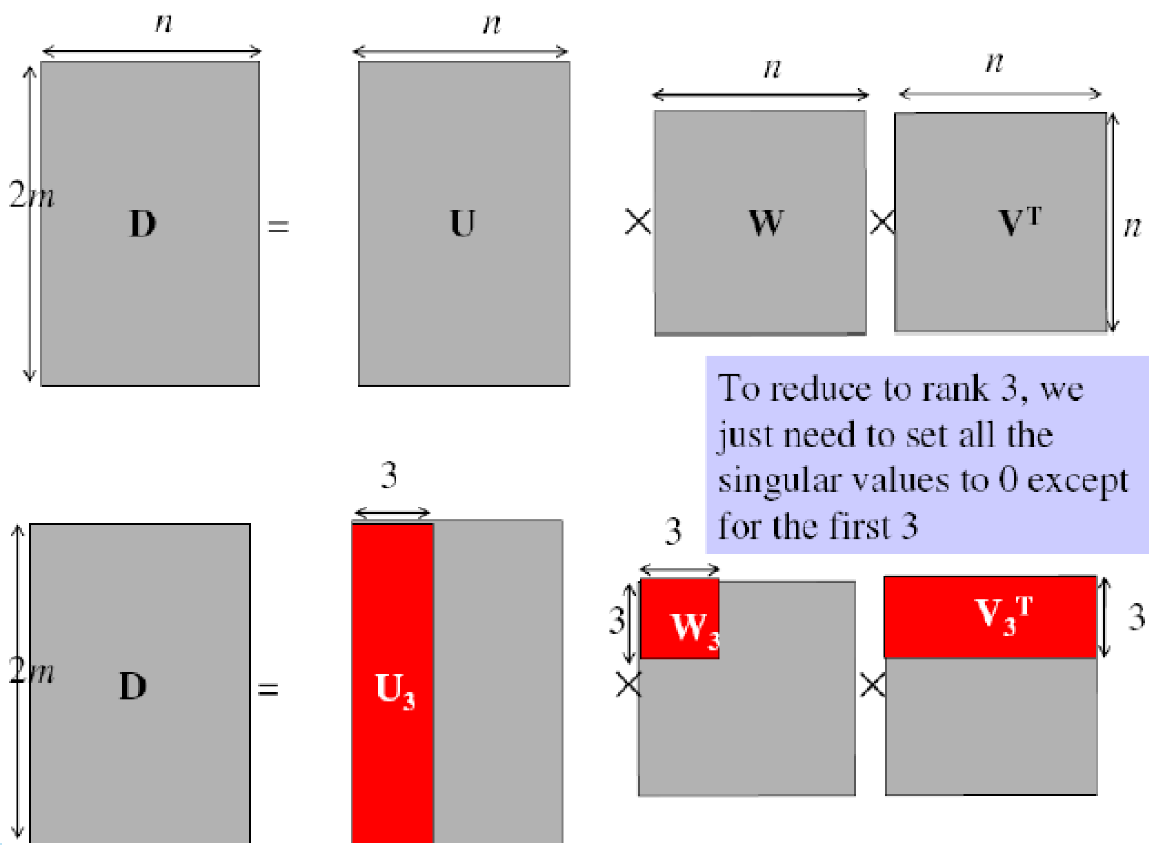


Source: M. Hebert

Structure from motion

Affine structure from motion:

- Singular Value Decomposition of **D**

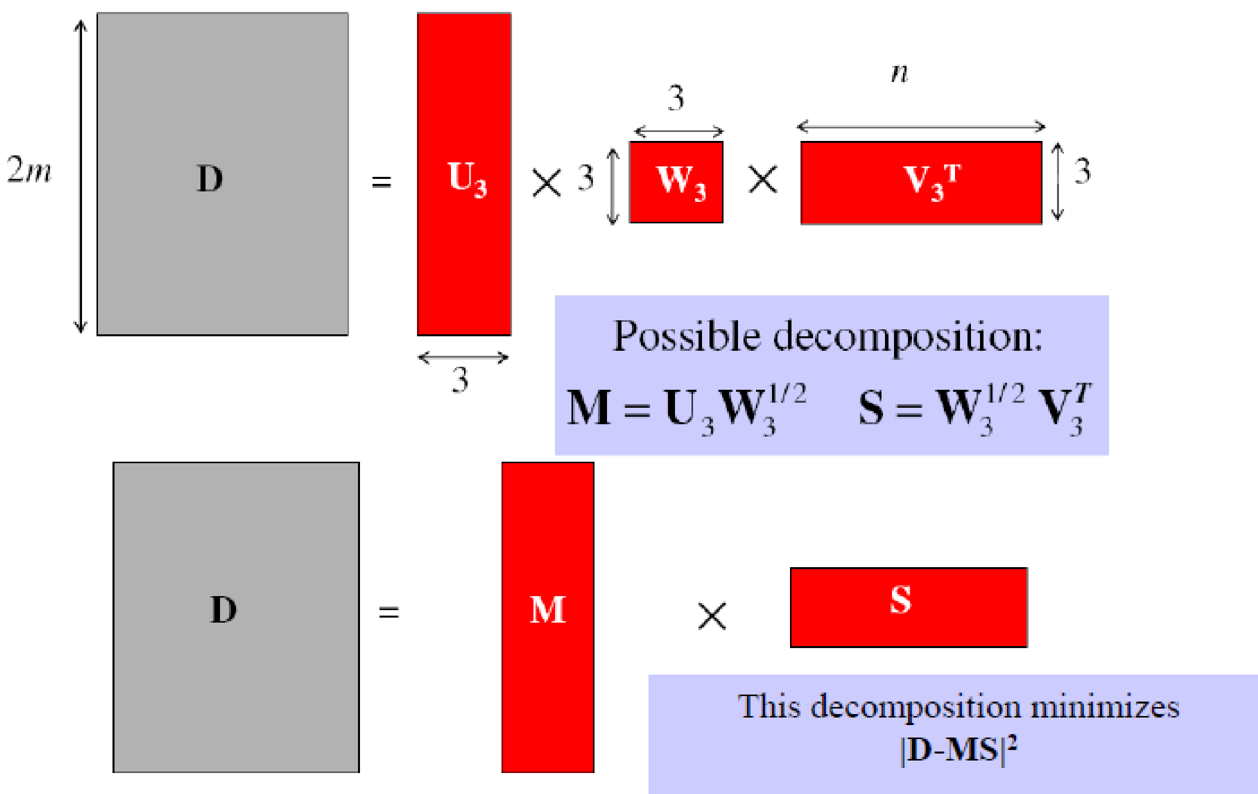


Source: M. Hebert

Structure from motion

Affine structure from motion:

- Singular Value Decomposition of **D**



Source: M. Hebert

Structure from motion

Affine ambiguity:

- The decomposition is not unique

$$\mathbf{D} = \mathbf{M}\,\mathbf{S} \quad \text{and} \quad \mathbf{D} = (\mathbf{MC})(\mathbf{C}^{-1}\mathbf{S})$$

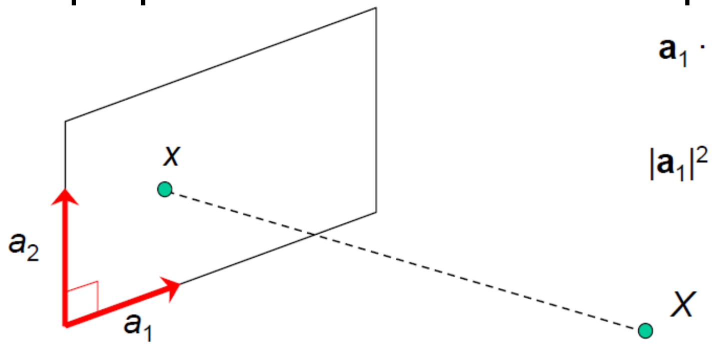
- This is why the above decomposition corresponds to an affine transformation

Source: M. Hebert

Structure from motion

<u>Eliminating the affine ambiguity:</u>

- Transform each projection matrix $\mathbf{A}_i$ to another matrix $\mathbf{A}_i\mathbf{C}$ to get orthographic projection: image axes are perpendicular and scale is equal to 1



$\mathbf{a}_1 \cdot \mathbf{a}_2 = 0$

$|\mathbf{a}_1|^2 = |\mathbf{a}_2|^2 = 1$

- This can be described by 3m equations in $\mathbf{L} = \mathbf{C}\mathbf{C}^T$

$$\mathbf{A}_i \mathbf{L} \mathbf{A}_i^T = \mathbf{Id}, \qquad i = 1, \ldots, m$$

  - Solve for $\mathbf{L}$
  - Recover $\mathbf{C}$ from $\mathbf{L}$ by applying Cholesky decomposition $\mathbf{L} = \mathbf{C}\mathbf{C}^T$
  - Update $\mathbf{M}$ and $\mathbf{S}$ by $\mathbf{M} = \mathbf{M}\,\mathbf{C}$ and $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

Source: M. Hebert

Structure from motion

Algorithm:

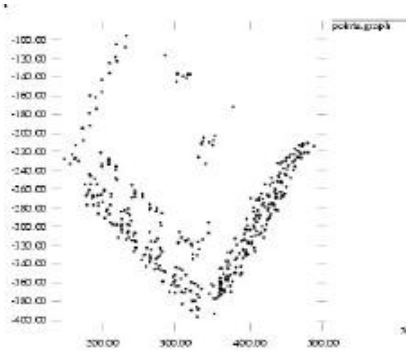
- Given: m images and n features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - Column j contains the projection of point j in all views
  - Row i contains one coordinate of the projections of all the n points in image i
- Factorize $\mathbf{D}$:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\ \mathbf{W}\ \mathbf{V}^T$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of $\mathbf{U}$
  - Create $\mathbf{V}_3$ by taking the first 3 columns of $\mathbf{V}$
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of $\mathbf{W}$
- Create the motion and shape matrices:
  - $\mathbf{M} = \mathbf{U}_3\mathbf{W}_3^{½}$ and $\mathbf{S} = \mathbf{W}_3^{½}\ \mathbf{V}_3^T$ (or $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^T$)
- Eliminate affine ambiguity

Structure from motion

Examples:

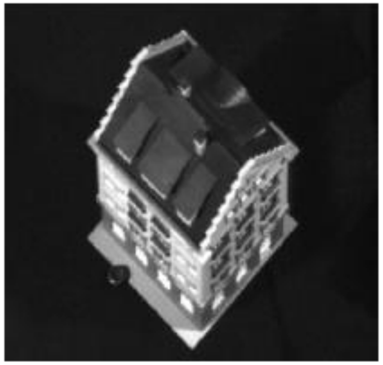
1

60



120

150

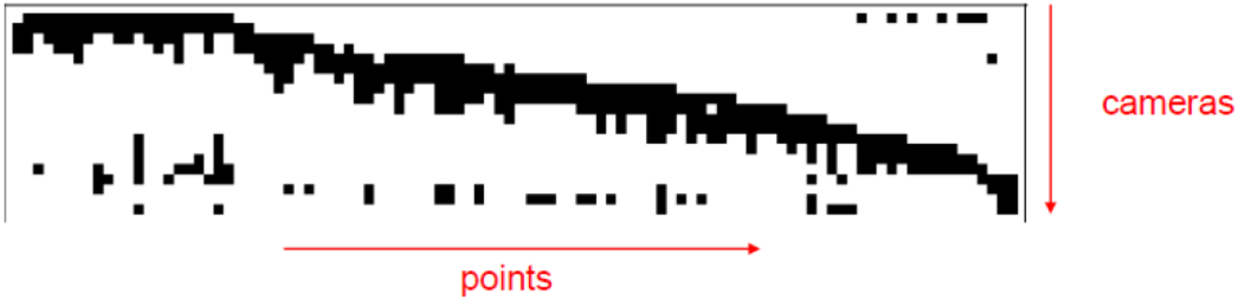


C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992

Structure from motion

<u>Missing data:</u>

- In reality, the measurement matrix has some missing values (some points are not visible from all views)



- One solution is to apply an iterative process
  - Solve using a dense submatrix of visible points
  - Iteratively add new cameras

C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. IJCV, 9(2):137-154, November 1992

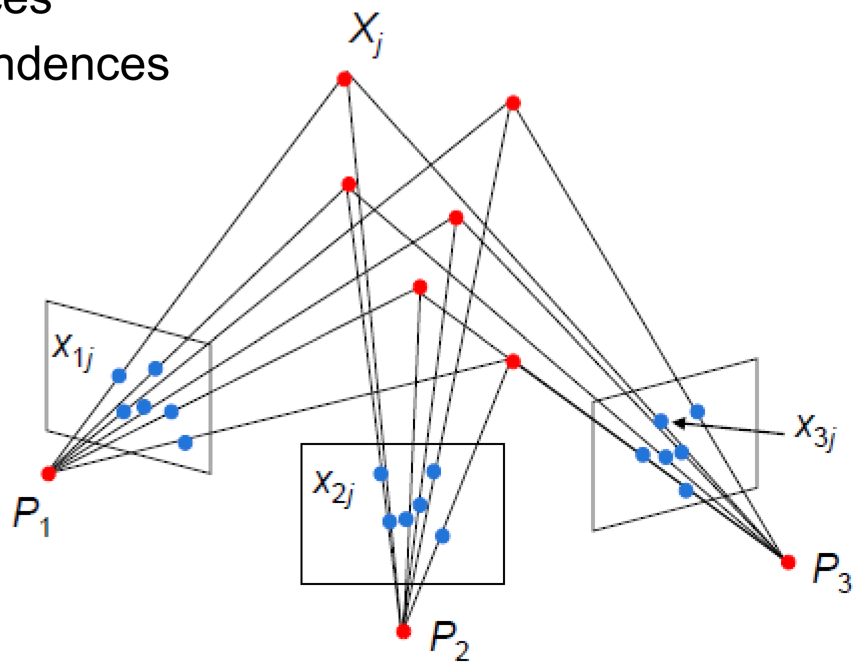
Structure from motion

Projective structure from motion:

- Given m images of n 3D points

$$z_{ij}\,\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

we want to estimate the m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$



Structure from motion

Projective structure from motion:

- Given m images of n 3D points

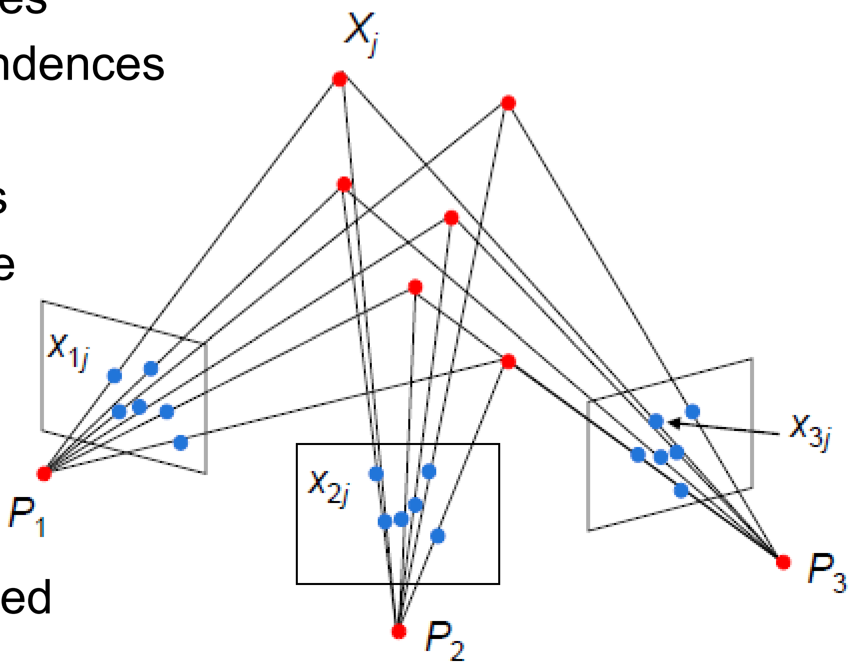
$$z_{ij}\,\mathbf{x}_{ij} = \mathbf{P}_i \mathbf{X}_j, \quad i = 1, \dots, m, \quad j = 1, \dots, n$$

we want to estimate the m projection matrices $\mathbf{P}_i$ and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$

- With no calibration info, cameras and points can only be recovered up to a 4x4 projective transformation **Q**:

$$\mathbf{X} \rightarrow \mathbf{QX}, \ \mathbf{P} \rightarrow \mathbf{PQ}^{-1}$$

- We can solve for structure and motion when: $2mn \geq 11m + 3n - 15$
- For two cameras, at least 7 points are needed



Structure from motion

Projective structure from motion with two cameras:

- Compute the fundamental matrix **F** between the two views
- First camera matrix : $[\mathbf{I}|\mathbf{0}]$
- Second camera matrix: $[\mathbf{A}|\mathbf{b}]$

- Then:

$$z\mathbf{x} = [\mathbf{I} \mid \mathbf{0}]\mathbf{X}, \quad z'\mathbf{x}' = [\mathbf{A} \mid \mathbf{b}]\mathbf{X}$$

$$z'\mathbf{x}' = \mathbf{A}[\mathbf{I} \mid \mathbf{0}]\mathbf{X} + \mathbf{b} = z\mathbf{A}\mathbf{x} + \mathbf{b}$$

$$z'\mathbf{x}' \times \mathbf{b} = z\mathbf{A}\mathbf{x} \times \mathbf{b}$$

$$(z'\mathbf{x}' \times \mathbf{b}) \cdot \mathbf{x}' = (z\mathbf{A}\mathbf{x} \times \mathbf{b}) \cdot \mathbf{x}'$$

$$\mathbf{x}'^{\mathrm{T}}[\mathbf{b}_{\times}]\mathbf{A}\mathbf{x} = 0$$

$$\mathbf{F} = [\mathbf{b}_{\times}]\mathbf{A}$$

$\mathbf{b}$: epipole ($\mathbf{F}^{\mathrm{T}}\mathbf{b} = 0$), $\quad \mathbf{A} = -[\mathbf{b}_{\times}]\mathbf{F}$

Structure from motion

Projective structure from motion with two cameras:

- The matrix **D** has the form

$$\mathbf{D} = \begin{bmatrix} z_{11}\mathbf{x}_{11} & z_{12}\mathbf{x}_{12} & \cdots & z_{1n}\mathbf{x}_{1n} \\ z_{21}\mathbf{x}_{21} & z_{22}\mathbf{x}_{22} & \cdots & z_{2n}\mathbf{x}_{2n} \\ & & \ddots & \\ z_{m1}\mathbf{x}_{m1} & z_{m2}\mathbf{x}_{m2} & \cdots & z_{mn}\mathbf{x}_{mn} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \vdots \\ \mathbf{P}_m \end{bmatrix}}_{\text{cameras } (3m \times 4)} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (4 \times n)}$$

- **D** = **MS** has rank equal to 4

- Iterative approach:
  - Keeping the depths z constant, we factorize **D** to find **M** and **S**
  - Keeping **D** constant, we solve for z

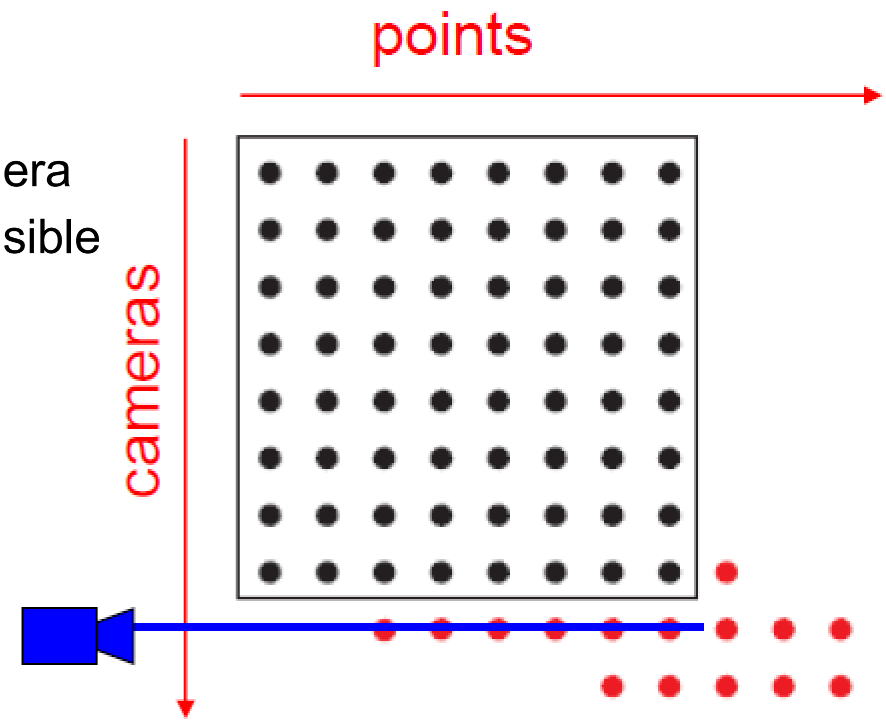
Structure from motion

Process:

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration



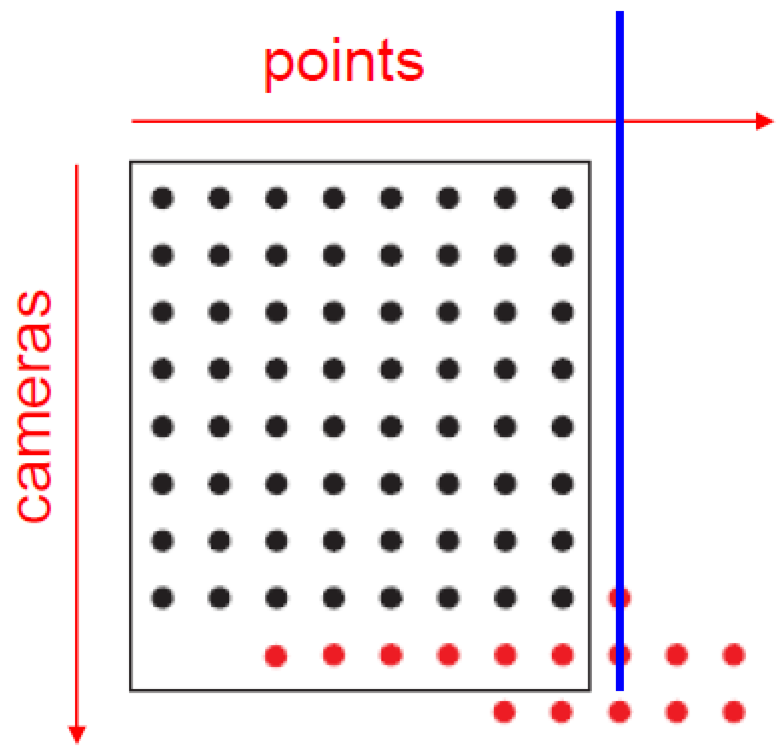
Structure from motion

Process:

- Initialize motion from two images using fundamental matrix

- Initialize structure by triangulation

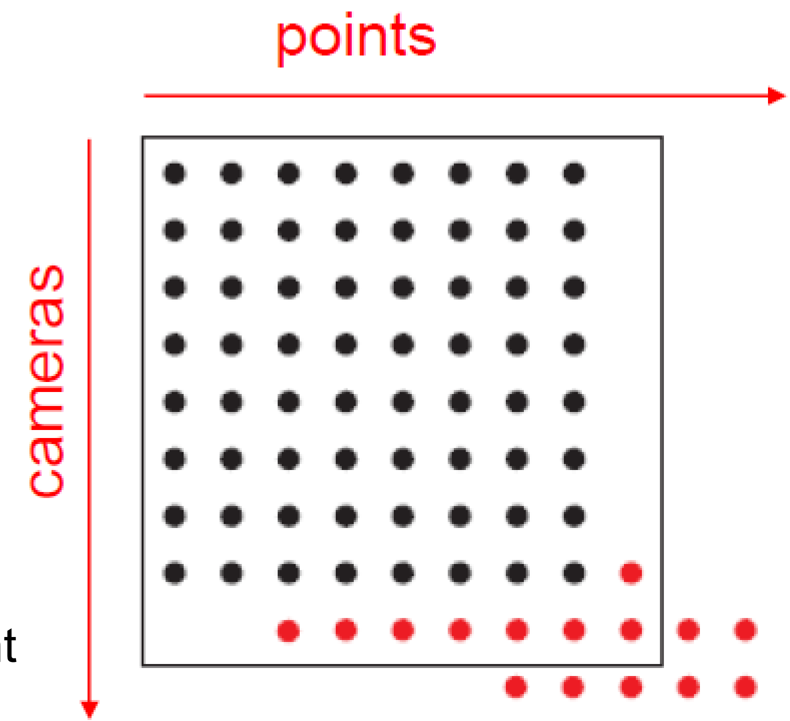
- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera-triangulation



Structure from motion

<u>Process:</u>

- Initialize motion from two images using fundamental matrix
- Initialize structure by triangulation
- For each additional view:
  - Determine projection matrix of new camera using all the known 3D points that are visible in its image – calibration
  - Refine and extend structure: compute new 3D points, re-optimize existing points that are also seen by this camera-triangulation
- Refine structure and motion: bundle adjustment

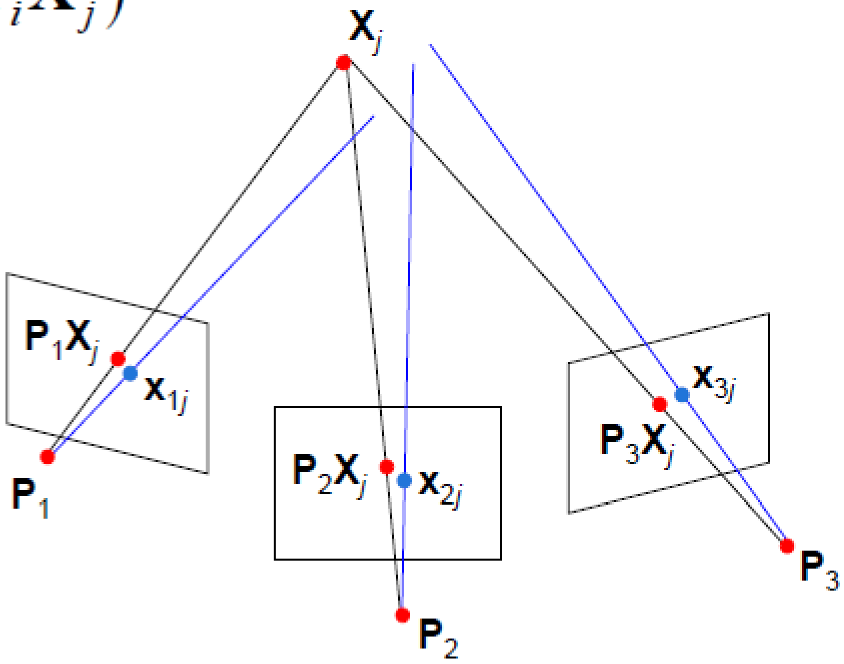


Structure from motion

Bundle adjustment:

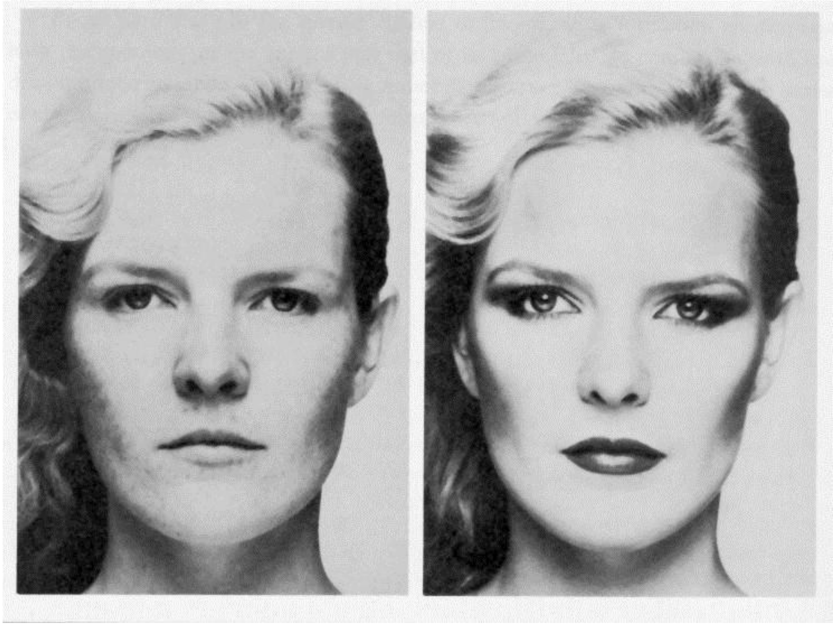
- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i\mathbf{X}_j\right)^2$$



Photometric Stereo

A technique used for estimating the surface normals of objects by observing that object under different lighting conditions.



R. Woodham, Photometric Method for Determining Surface Orientation from Multiple Images. Optical Engineering 19(1)139-144 (1980).

Diffuse reflection

Diffuse reflection governed by Lambert's law
- Viewed brightness does not depend on viewing direction
- Brightness does depend on direction of illumination
- This is the model most often used in computer vision

Lambert's Law: $I_e = k_d \mathbf{N} \mathbf{L} I_i$

L, N, V are unit vectors

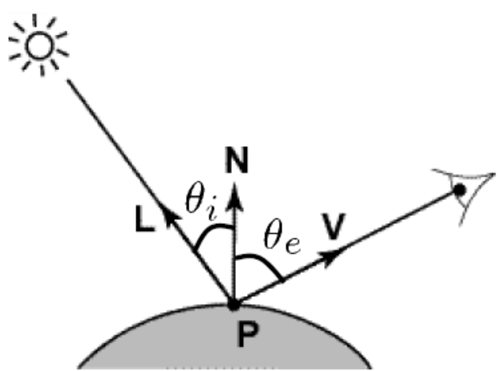
I_e is the outgoing radiance

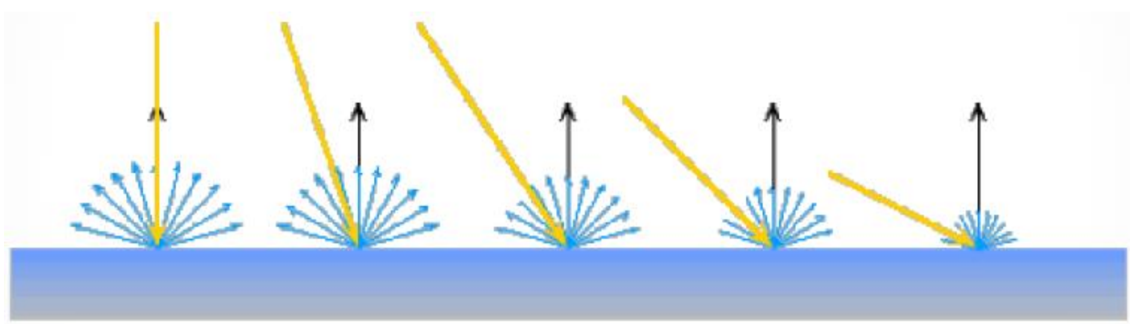
I_i is the incoming radiance

k_d is called albedo

BRDF for **Lambertian surface**

$$\rho(\theta_i, \varphi_i, \theta_e, \varphi_e) = k_d \cos(\theta_i)$$



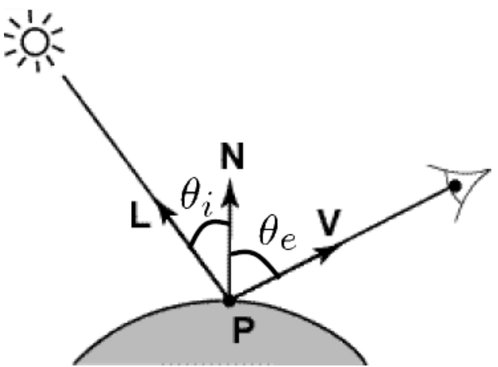


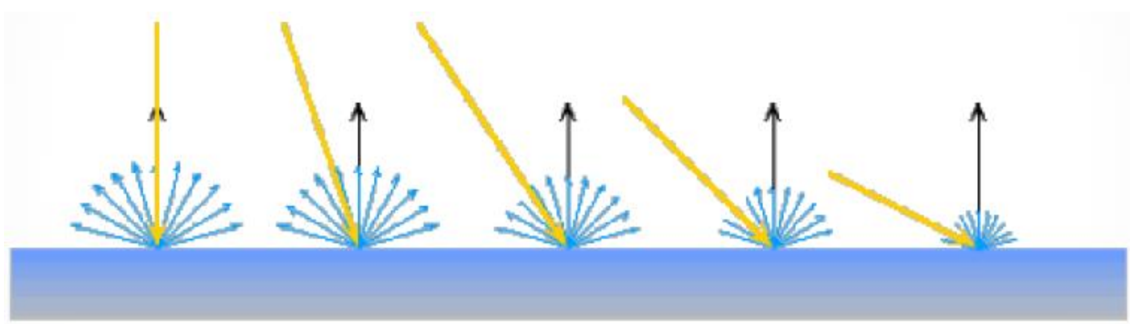
Diffuse reflection

Diffuse reflection governed by Lambert's law
- Viewed brightness does not depend on viewing direction
- Brightness does depend on direction of illumination
- This is the model most often used in computer vision

Lambert's Law: $I_e = k_d \mathbf{N}\,\mathbf{L}\,I_i$
Simplifying assumptions:
- $I = I_e$: camera response function f is the identity function
- $I_i = 1$: light source intensity is 1

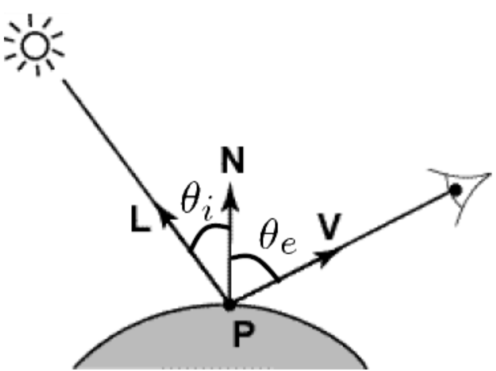
$$I = k_d \mathbf{N} \cdot \mathbf{L}$$





Shape from shading

We can directly measure the angle between the normal and the light source:
- Not quite enough information to compute surface shape
- But can be if you add some additional information, for example
  - assume a few of the normals are known (e.g., along silhouette)
  - constraints on neighboring normals—"integrability"
  - smoothness
- Hard to get it to work well in practice
  - plus, how many real objects have constant albedo?

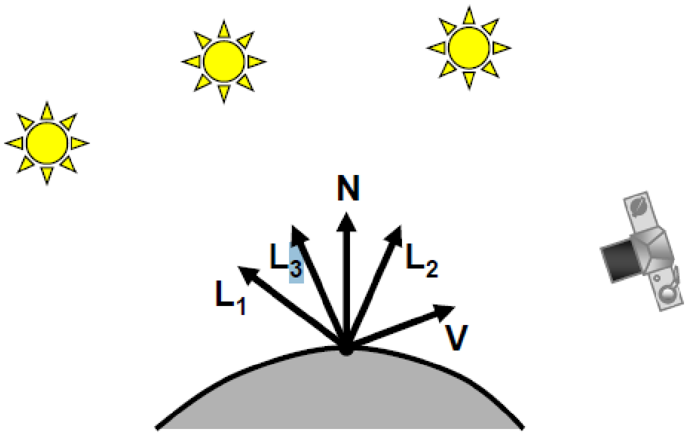


Suppose $k_d = 1$

$$\begin{aligned} I &= k_d \mathbf{N} \cdot \mathbf{L} \\ &= \mathbf{N} \cdot \mathbf{L} \\ &= \cos\ \theta_i \end{aligned}$$

Photometric stereo

Capture a surface in different lighting conditions

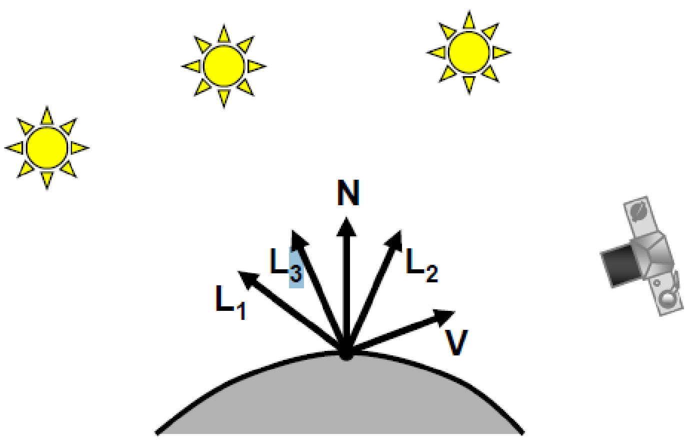


$$
\begin{aligned}
I_1 &= k_d \mathbf{N} \cdot \mathbf{L_1} \\
I_2 &= k_d \mathbf{N} \cdot \mathbf{L_2} \\
I_3 &= k_d \mathbf{N} \cdot \mathbf{L_3}
\end{aligned}
$$

$$
\begin{bmatrix} I_1 & I_2 & I_3 \end{bmatrix} = k_d \mathbf{N}^T \begin{bmatrix} \mathbf{L_1} & \mathbf{L_2} & \mathbf{L_3} \end{bmatrix}
$$

Photometric stereo

Capture a surface in different lighting conditions



$$
\begin{aligned}
I_1 &= k_d \mathbf{N} \cdot \mathbf{L_1} \\
I_2 &= k_d \mathbf{N} \cdot \mathbf{L_2} \\
I_3 &= k_d \mathbf{N} \cdot \mathbf{L_3}
\end{aligned}
$$

$$
\underbrace{\begin{bmatrix} I_1 & I_2 & I_3 \end{bmatrix}}_{\substack{\mathbf{I} \\ 1 \times 3}} = \underbrace{k_d \mathbf{N}^T}_{\substack{\mathbf{G} \\ 1 \times 3}} \underbrace{\begin{bmatrix} \mathbf{L_1} & \mathbf{L_2} & \mathbf{L_3} \end{bmatrix}}_{\substack{\mathcal{L} \\ 3 \times 3}}
$$

- Solve the equations

$$
\mathbf{G} = \mathbf{I}\mathbf{L}^{-1}
$$

$$
k_d = ||\mathbf{G}||
$$

$$
\mathbf{N} = \frac{1}{k_d}\mathbf{G}
$$

Photometric stereo

We can get better results by using more images/lights:

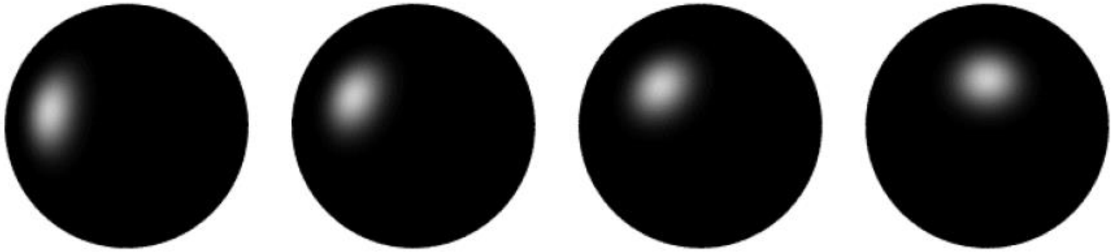
$$\begin{bmatrix} I_1 & \dots & I_n \end{bmatrix} = k_d \mathbf{N}^T \begin{bmatrix} \mathbf{L_1} & \dots & \mathbf{L_n} \end{bmatrix}$$

- Then use a least squares solution, where we solve for **N** and k_d:

$$\begin{aligned} \mathbf{I} &= \mathbf{G}\mathbf{L} \\ \mathbf{I}\mathbf{L}^{\mathbf{T}} &= \mathbf{G}\mathbf{L}\mathbf{L}^{\mathbf{T}} \\ \mathbf{G} &= (\mathbf{I}\mathbf{L}^{\mathbf{T}})(\mathbf{L}\mathbf{L}^{\mathbf{T}})^{-1} \end{aligned}$$

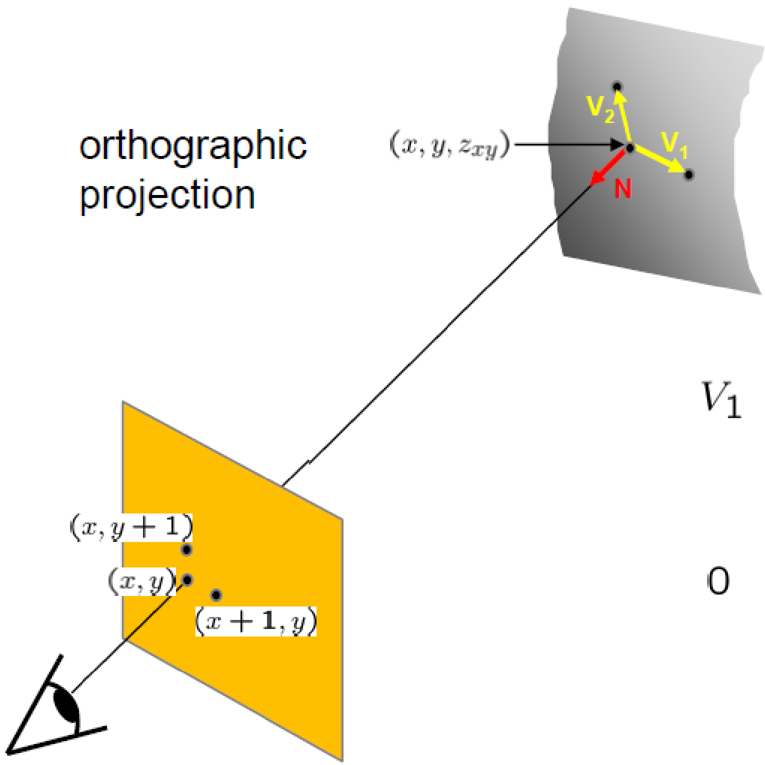
Specular reflection

Moving light source



To detect where a light source is → place a chrome sphere in the scene

Depth from normals



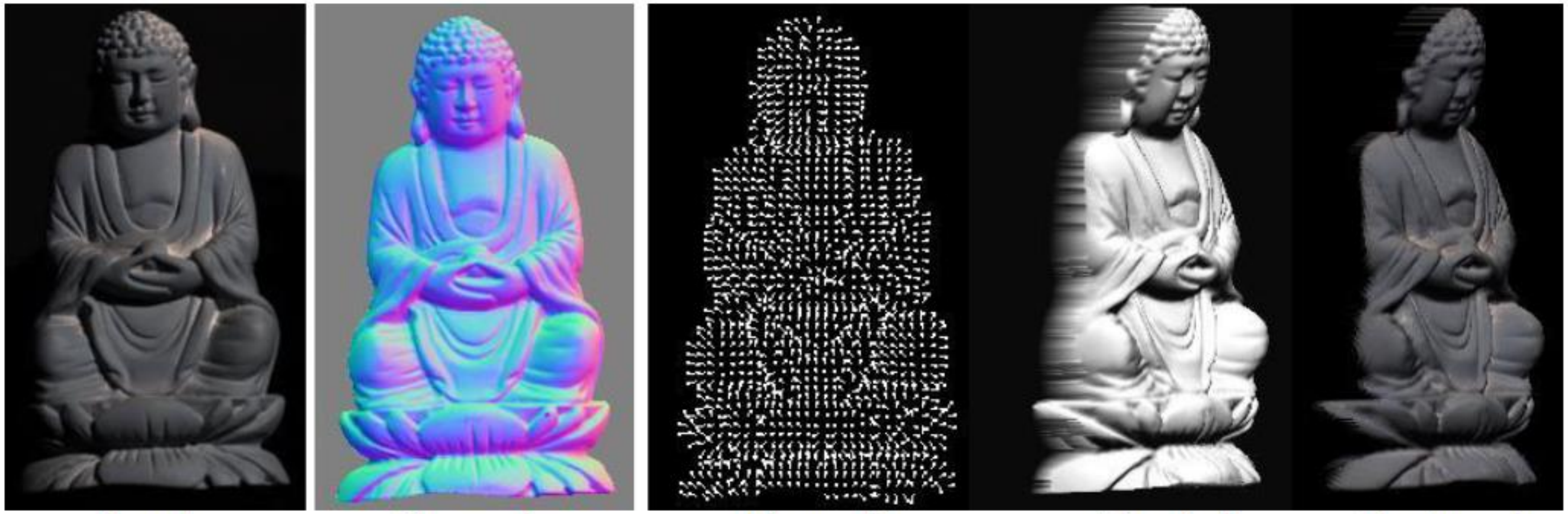
$$V_1 = (x+1, y, z_{x+1,y}) - (x, y, z_{xy})$$
$$= (1, 0, z_{x+1,y} - z_{xy})$$

$$0 = N \cdot V_1$$
$$= (n_x, n_y, n_z) \cdot (1, 0, z_{x+1,y} - z_{xy})$$
$$= n_x + n_z(z_{x+1,y} - z_{xy})$$

Get a similar equation for $\mathbf{V}_2$

- Each normal gives us two linear constraints on z
- compute z values by solving a matrix equation

Some results



Input
(1 of 12)

Normals

Normals

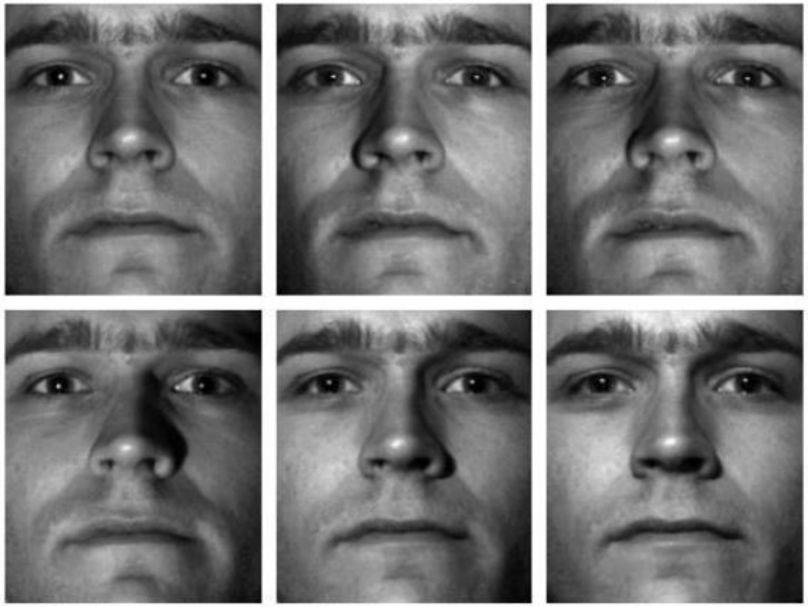
Shaded
rendering

Textured
rendering

You may read the details here:

http://pages.cs.wisc.edu/~csverma/CS766_09/Stereo/stereo.html

Some results



from Athos Georghiades

http://cvc.yale.edu/people/Athos.html

Limitations

Big problems
- doesn’t work for shiny objects, semi-translucent objects
- shadows, inter-reflections

Smaller problems
- camera and lights have to be distant
- calibration requirements
  - measure light source directions, intensities
  - camera response function (newer work addresses some of these issues)

Some papers for further reading:

- Zickler, Belhumeur, and Kriegman, "Helmholtz Stereopsis: Exploiting Reciprocity for Surface Reconstruction." IJCV, Vol. 49 No. 2/3, pp 215-227.
- Hertzmann & Seitz, “Example-Based Photometric Stereo: Shape Reconstruction with General, Varying BRDFs.” IEEE Trans. PAMI 2005

Image alignment

Applications:

- image blending
- Panoramas

(a) (b) (c)

Image alignment

Applications:

- image blending
- Panoramas
- Object recognition

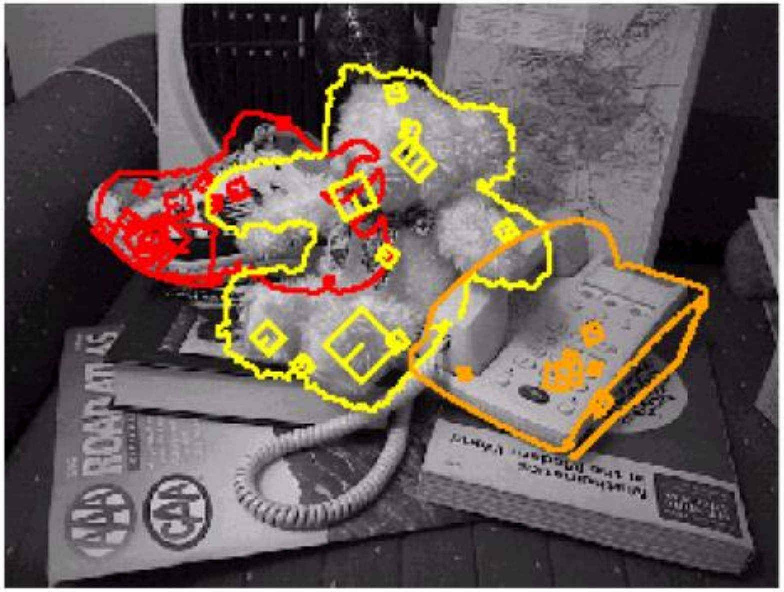
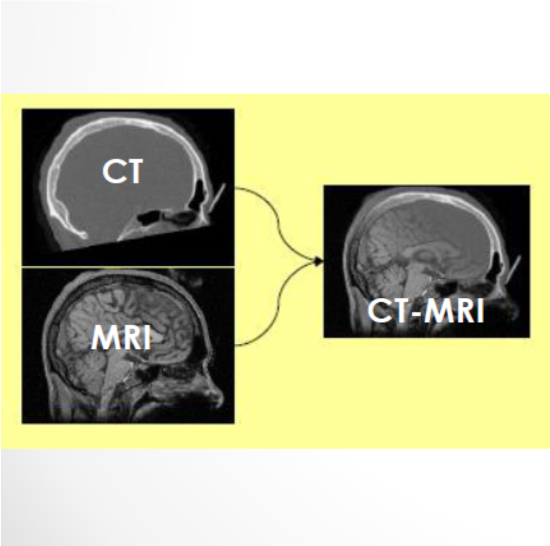
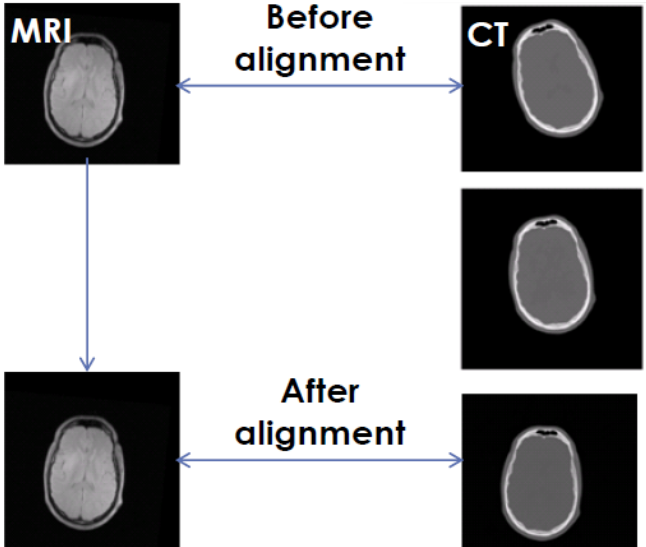


Image alignment

Applications:
- image blending
- Panoramas
- Object recognition
- Medical image registration





Motion models

Image transformations:
- Hierarchy
- Transformation model

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$$

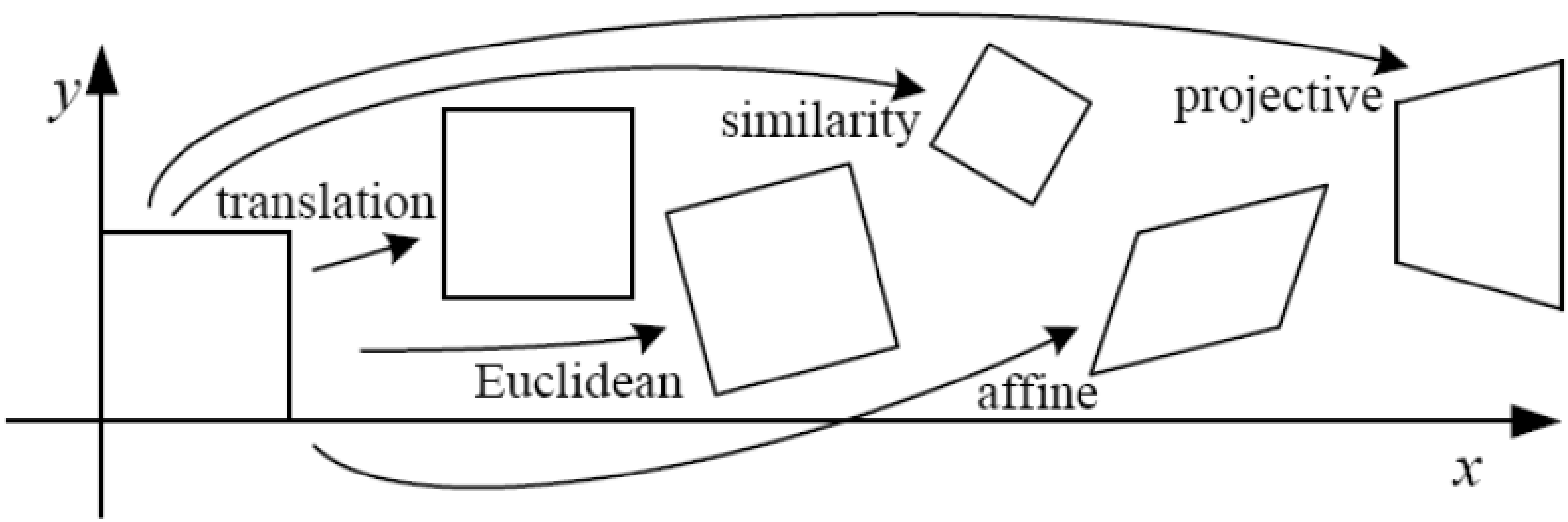
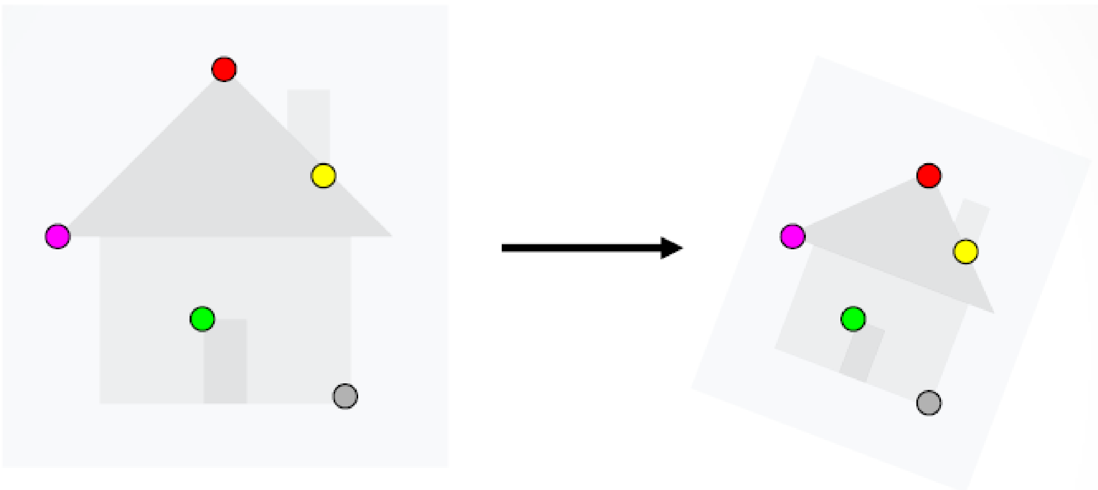


Image alignment

Two broad approaches:

- Direct (pixel-based) alignment
  - Search for alignment where most pixels agree
- Feature-based alignment
  - Search for alignment where extracted features agree
  - Can be verified using pixel-based alignment



Affine transformation

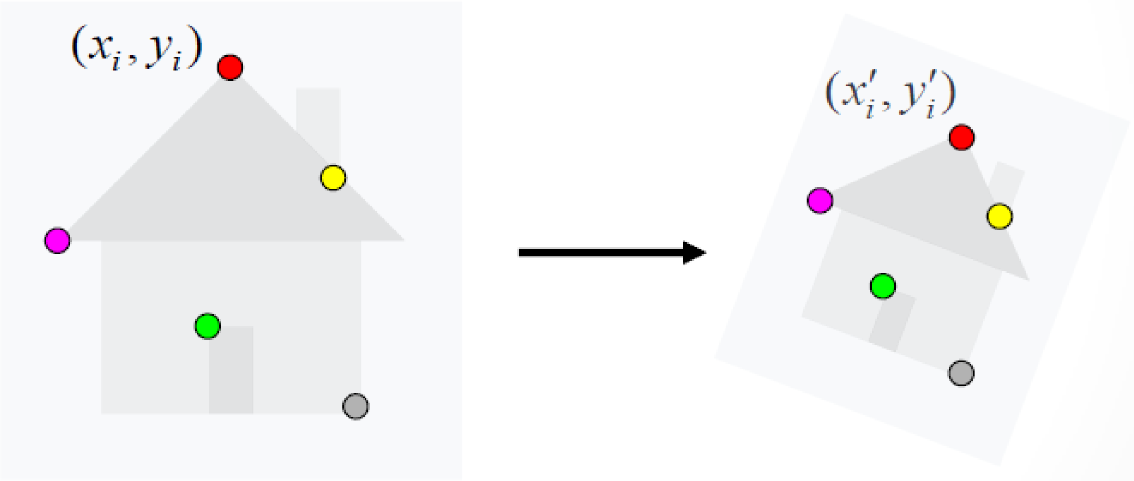
Affine model approximates:
 - perspective projection of planar objects

Affine transformation

Affine model approximates:

- perspective projection of planar objects
- Assuming we know the correspondences, we can use the affine transformation model

$$\begin{bmatrix} x_i' \\ y_i' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}$$



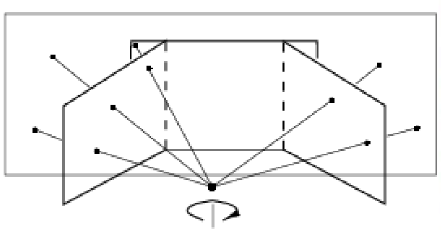
Affine transformation

Affine model approximates:

- perspective projection of planar objects
- Assuming we know the correspondences, we can use the affine transformation model
- Given a set of 3-pairs of points, parameters of the model can be estimated:

$$\begin{bmatrix} & & \cdots & & & \\ x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & x_i & y_i & 0 & 1 \\ & & \cdots & & & \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} \cdots \\ x_i' \\ y_i' \\ \cdots \end{bmatrix}$$

Panoramas



Panoramas

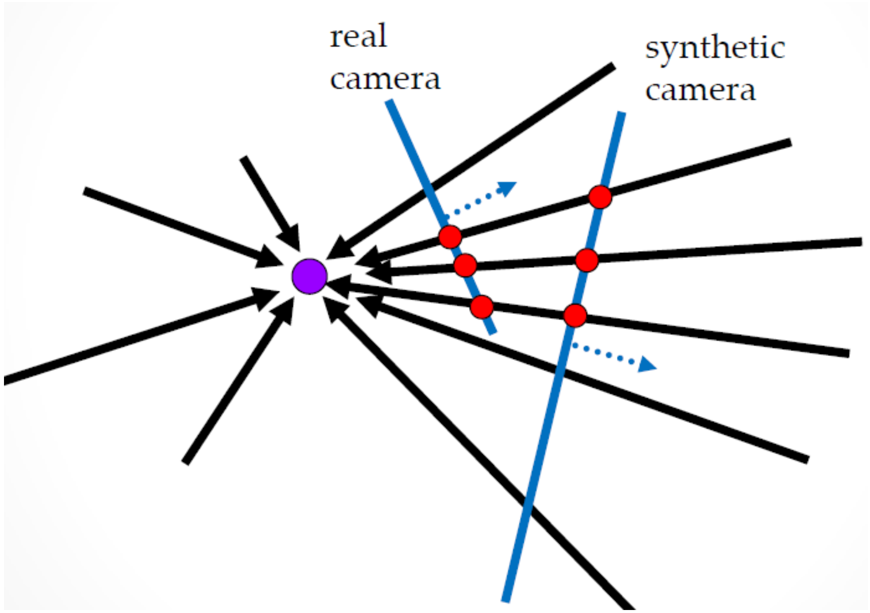
Process:

- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- Repeat for more images

Panoramas

Process:
- Take a sequence of images from the same position
  - Rotate the camera about its optical center
- Compute transformation between second image and first
- Transform the second image to overlap with the first
- Blend the two together to create a mosaic
- Repeat for more images

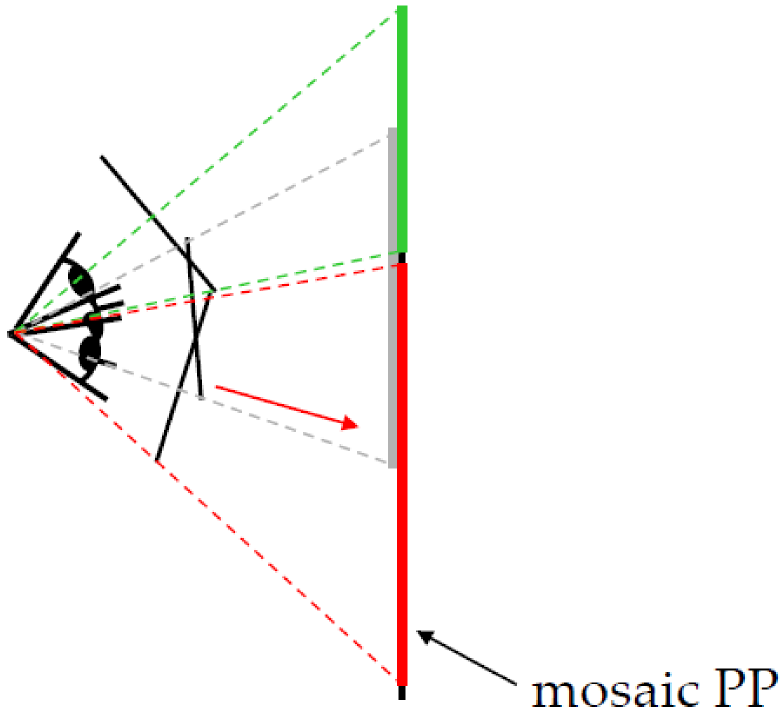
The above process can generate synthetic Camera view, as long as it has <u>the same Center of projection</u>



Panoramas

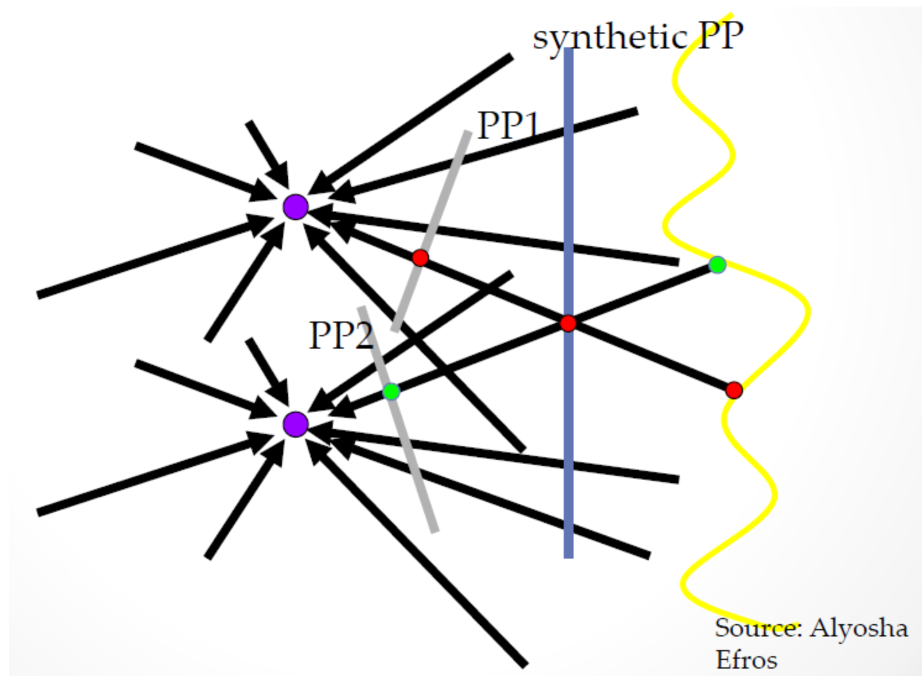
The mosaic has a natural interpretation in 3D

- The images are re-projected onto a common plane
- The mosaic is formed on this plane
- Mosaic is a synthetic wide-angle camera



Panoramas

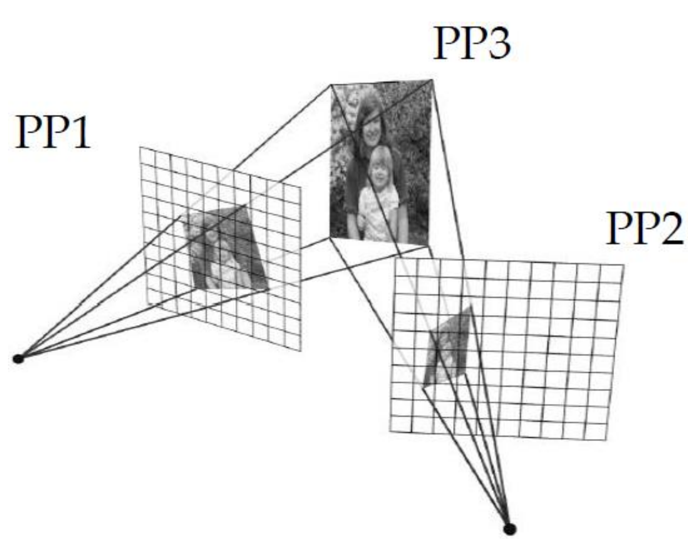
What happens when we change the camera center?



Panoramas

What happens when we change the camera center?

Planar (or far away) scene assumption:
- example: aerial photographs



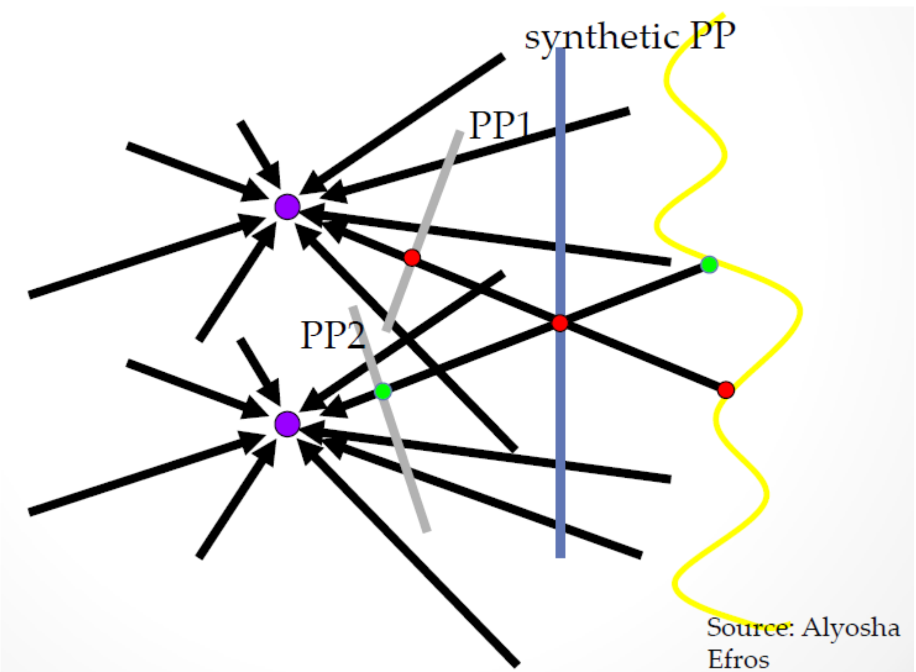


Image warping

Parametric (global) warps

Image warping vs filtering

Image filtering → change in the range of image values

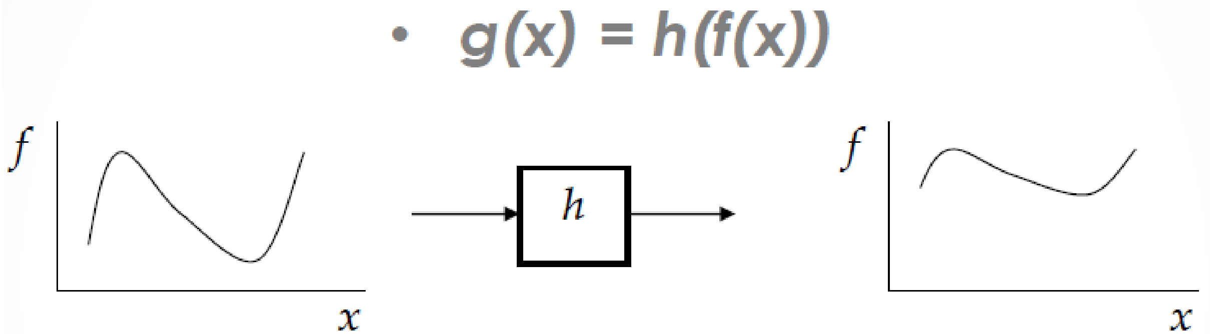


Image warping: change in the domain (structure) of the image

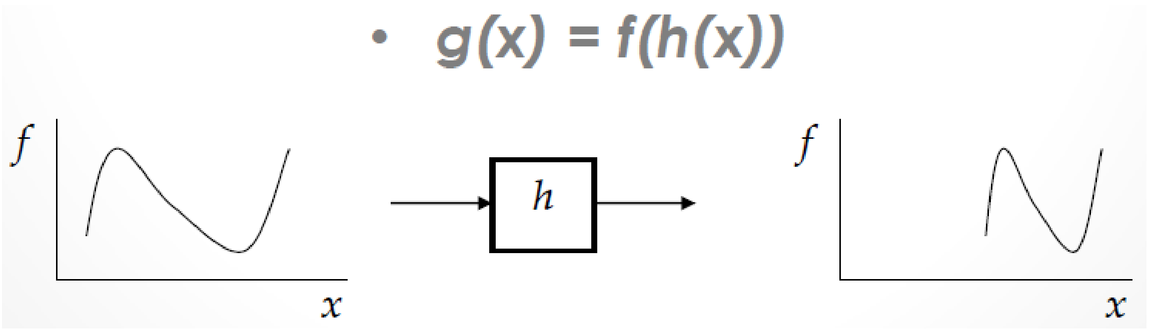


Image warping vs filtering

Image filtering → change in the range of image values

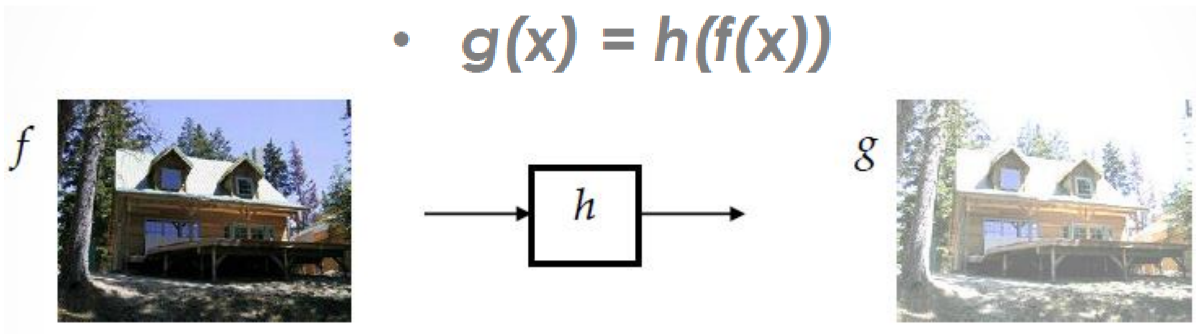


Image warping: change in the domain (structure) of the image

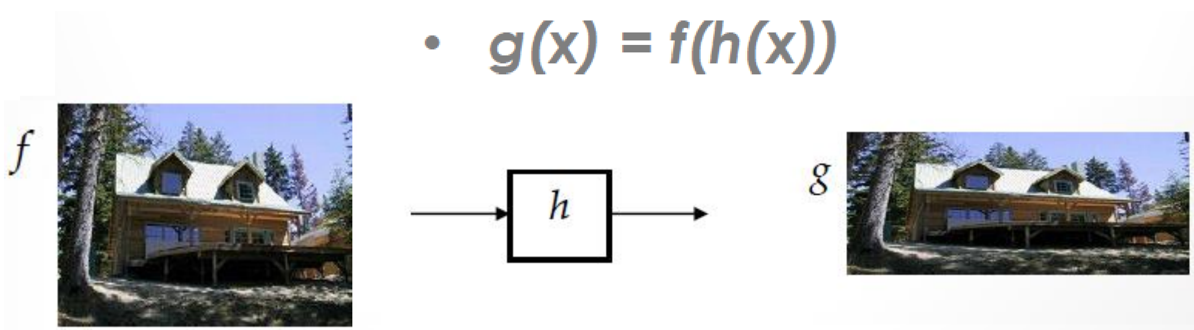


Image warping

Given

- a coordinate transformation **x**' = h(**x**) and
- a source image f(**x**)

How can we calculate the transformed image g(**x**') = f(h(**x**))?

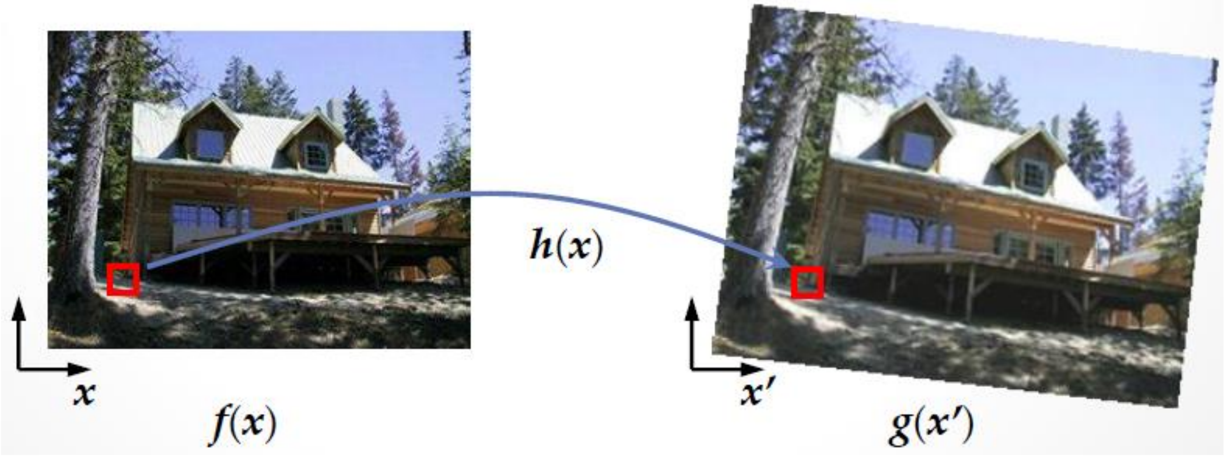


Image warping

Given
- a coordinate transformation **x**' = h(**x**) and
- a source image f(**x**)

How can we calculate the transformed image g(**x**') = f(h(**x**))?

Process:
- Get each pixel g(**x**') from its corresponding location **x**' = h(**x**) in f(**x**)

This process leads to many real-valued pixel coordinates!

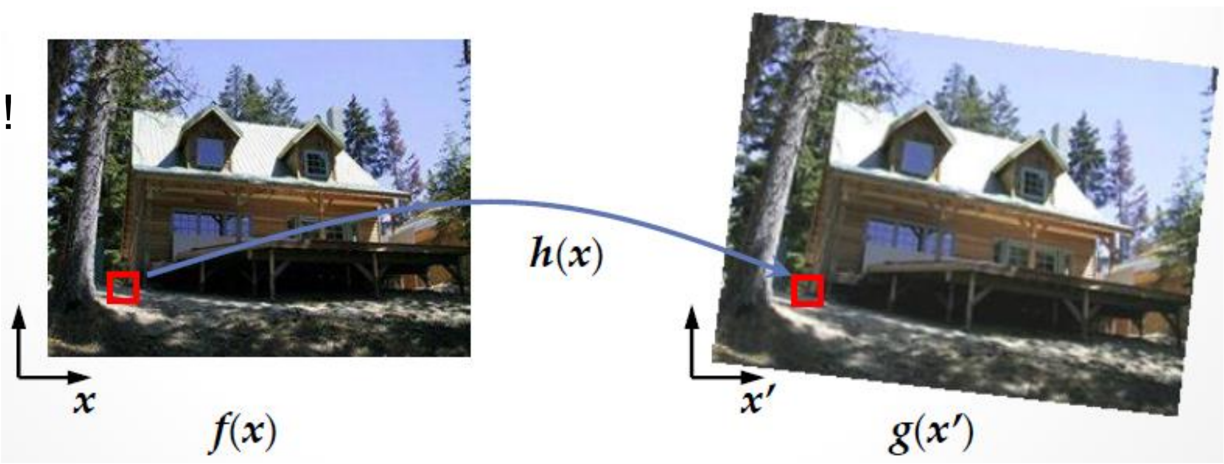


Image warping

Given

- a coordinate transformation **x**’ = h(**x**) and
- a source image f(**x**)

How can we calculate the transformed image g(**x**’) = f(h(**x**))?

Process:

- Get each pixel g(**x**’) from its corresponding location **x**’ = h(**x**) in f(**x**)

This process leads to many real-valued pixel coordinates!

We can re-sample color values from interpolated source image

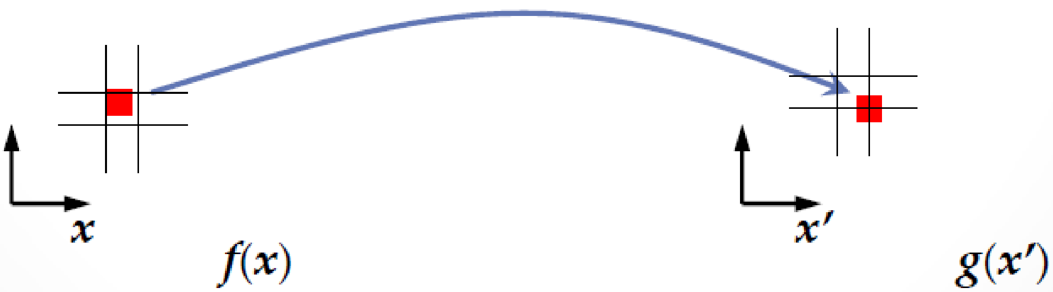


Image warping

Inverse warping:

- Given each pixel g(**x**',**y**') from its corresponding location (x,y) = T^{-1}(x',y') in the first image
- If pixel comes from 'between' two pixels in the input image (i.e. the calculated (x,y) are real values), then we apply interpolation in the input image colors

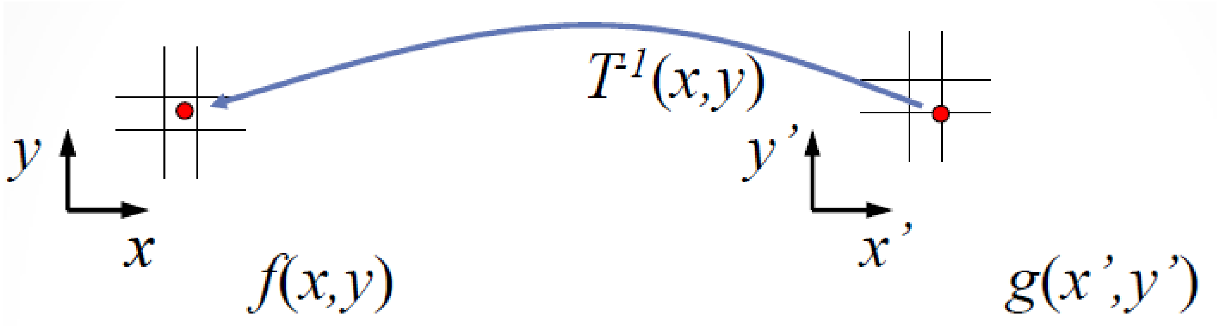


Image warping

Inverse warping:

- Given each pixel g(**x**',**y**') from its corresponding location (x,y) = T^{-1}(x',y') in the first image
- If pixel comes from 'between' two pixels in the input image (i.e. the calculated (x,y) are real values), then we apply interpolation in the input image colors

Interpolation types:

- nearest neighbor
- bilinear
- bicubic
- sinc / FIR

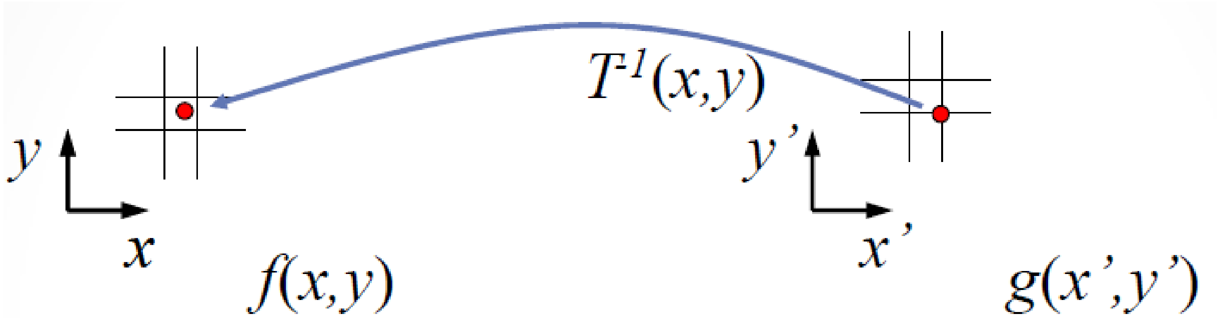


Image warping

Parametric (global) warps

Non-parametric (local) warps

Apply a local warp in image locations