UNIVERSITÄT BONN

Prof. Dr. Juergen Gall

Recapitulation 2

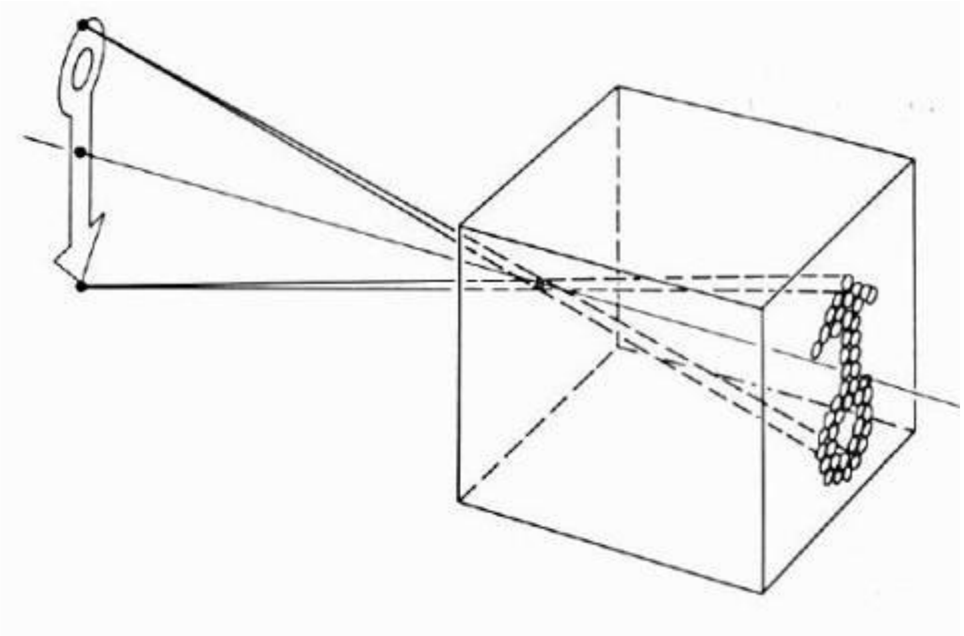
MA-INF 2201 - Computer Vision

WS24/25

Cameras

Lana Lazebnik

Pinhole camera model

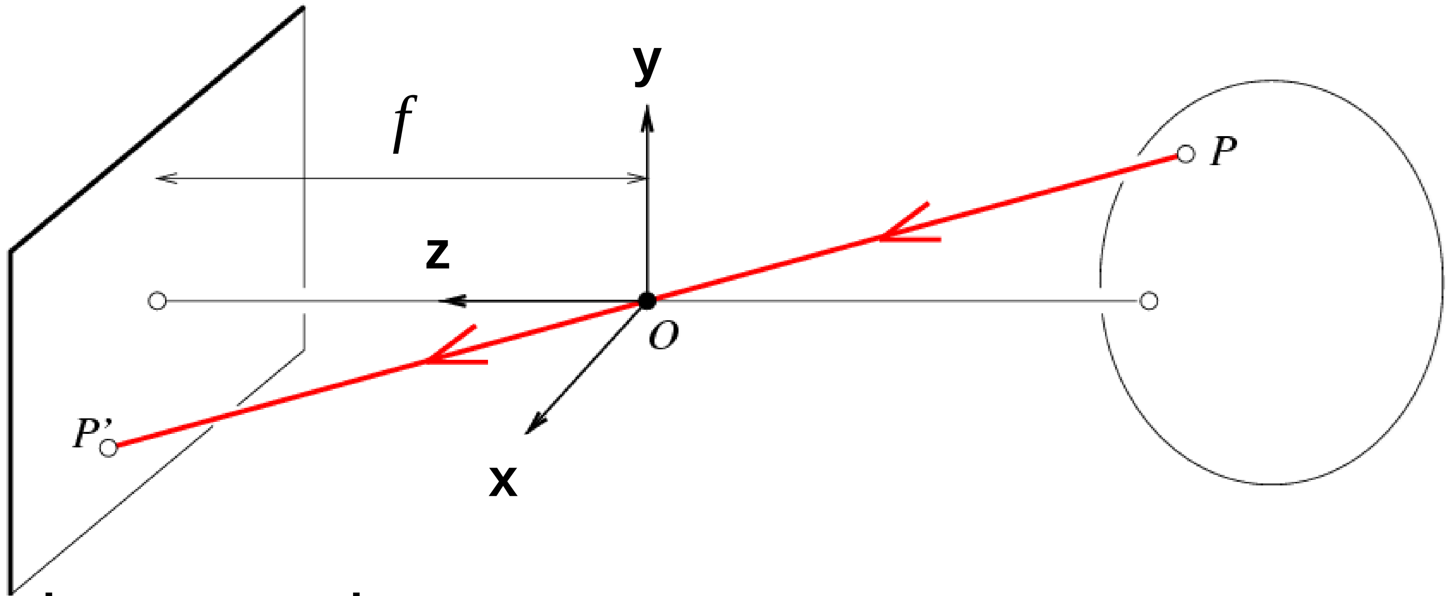


Pinhole model:

- Captures pencil of rays – all rays through a single point
- The point is called Center of Projection (focal point)
- The image is formed on the Image Plane

Steve Seitz

Modeling projection



- Projection equations
  - Compute intersection with image plane of ray from ***P*** = (x,y,z) to ***O***
  - Derived using similar triangles

$$(x, y, z) \rightarrow (f\,\frac{x}{z}, f\,\frac{y}{z}, f\,)$$

- We get the projection by throwing out the last coordinate:

$$(x, y, z) \rightarrow (f\,\frac{x}{z}, f\,\frac{y}{z})$$

J. Ponce, S. Seitz

Homogeneous coordinates

UNIVERSITÄT BONN

$$(x, y, z) \rightarrow (f\frac{x}{z}, f\frac{y}{z})$$

Is this a linear transformation?

- no—division by z is nonlinear

Trick: add one more coordinate:

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

homogeneous image coordinates

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

homogeneous scene coordinates

Converting *from* homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Steve Seitz

Perspective Projection Matrix

Projection is a matrix multiplication using homogeneous coordinates

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z/f \end{bmatrix} \Rightarrow \left(f\frac{x}{z}, f\frac{y}{z}\right)$$

divide by the third coordinate

In practice: lots of coordinate transformations…

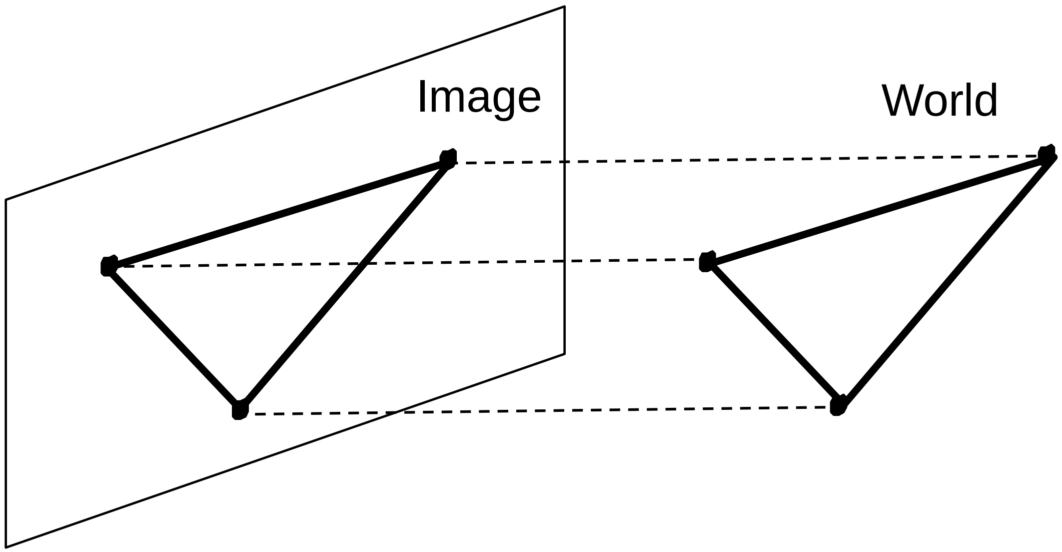
$$\begin{pmatrix} \text{2D} \\ \text{point} \\ \text{(3x1)} \end{pmatrix} = \begin{pmatrix} \text{Camera to} \\ \text{pixel coord.} \\ \text{trans. matrix} \\ \text{(3x3)} \end{pmatrix} \begin{pmatrix} \text{Perspective} \\ \text{projection matrix} \\ \text{(3x4)} \end{pmatrix} \begin{pmatrix} \text{World to} \\ \text{camera coord.} \\ \text{trans. matrix} \\ \text{(4x4)} \end{pmatrix} \begin{pmatrix} \text{3D} \\ \text{point} \\ \text{(4x1)} \end{pmatrix}$$

Lana Lazebnik

Orthographic Projection

Special case of perspective projection

- Distance from center of projection to image plane is infinite

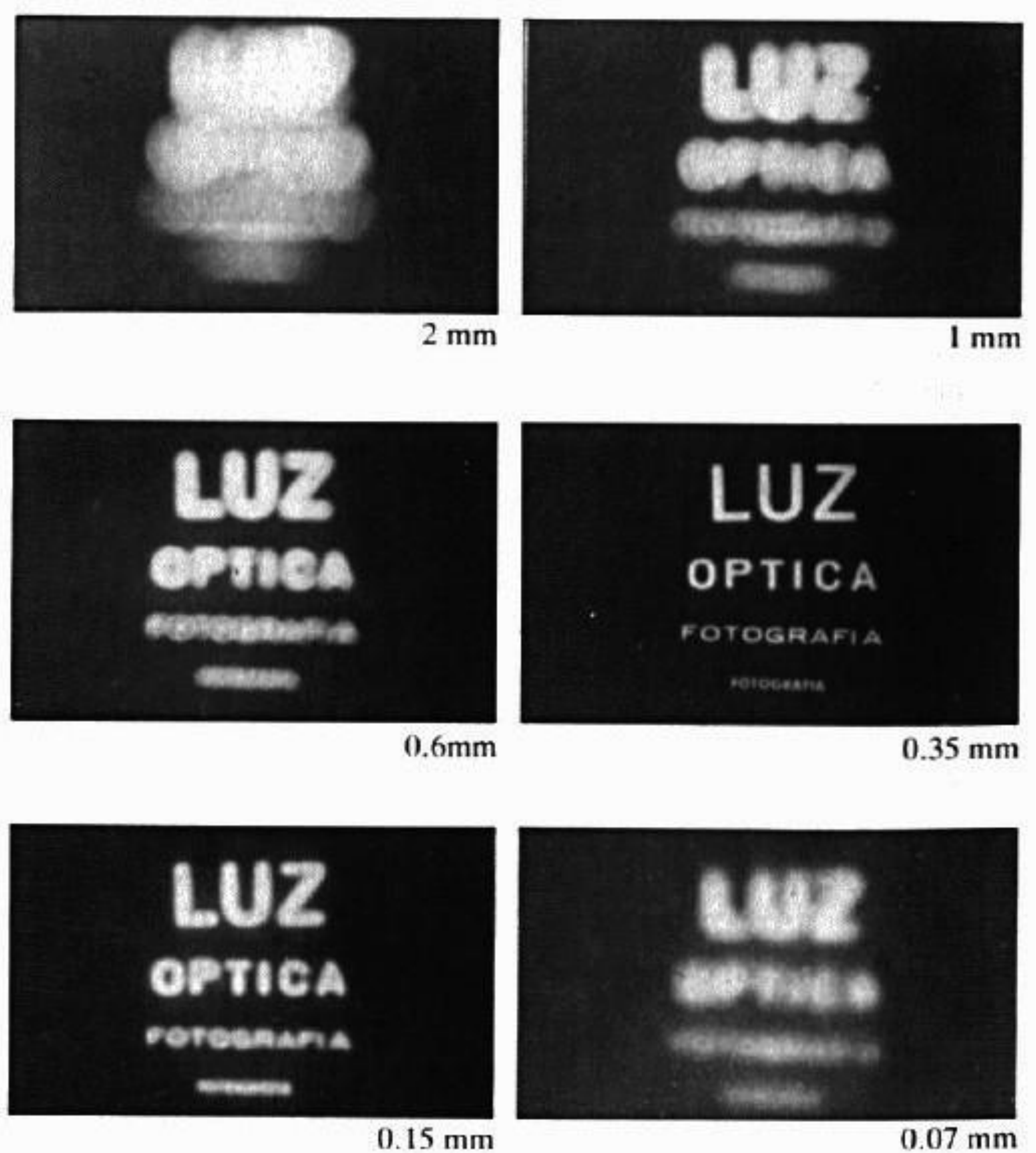


- Also called “parallel projection”
- What’s the projection matrix?

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

Steve Seitz

Shrinking the aperture



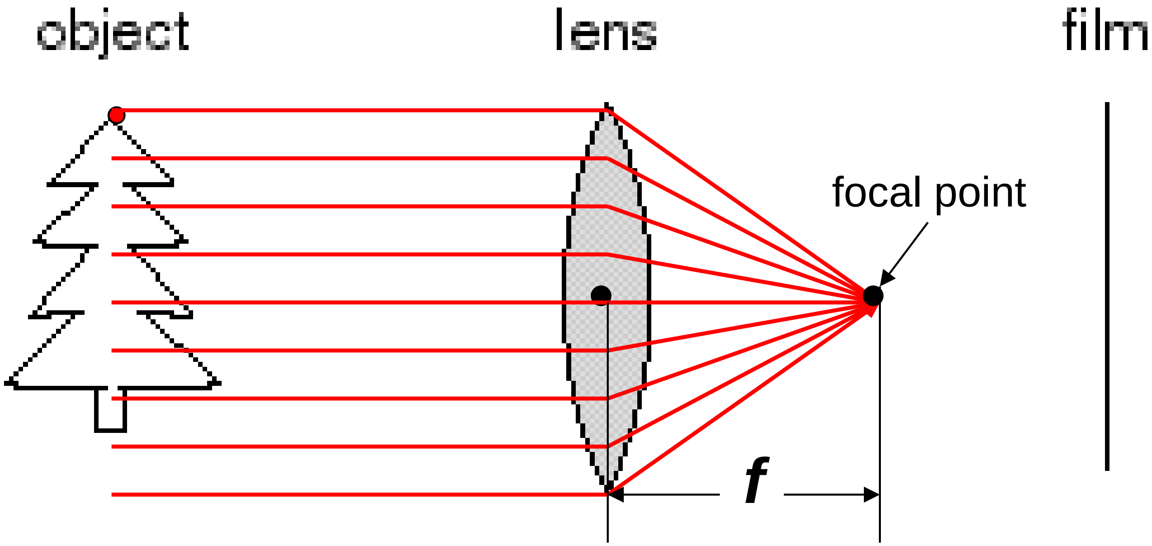
Lana Lazebnik

Adding a lens

A lens focuses light onto the film

– Thin lens model:

- Rays passing through the center are not deviated (pinhole projection model still holds)
- All parallel rays converge to one point on a plane located at the focal length f

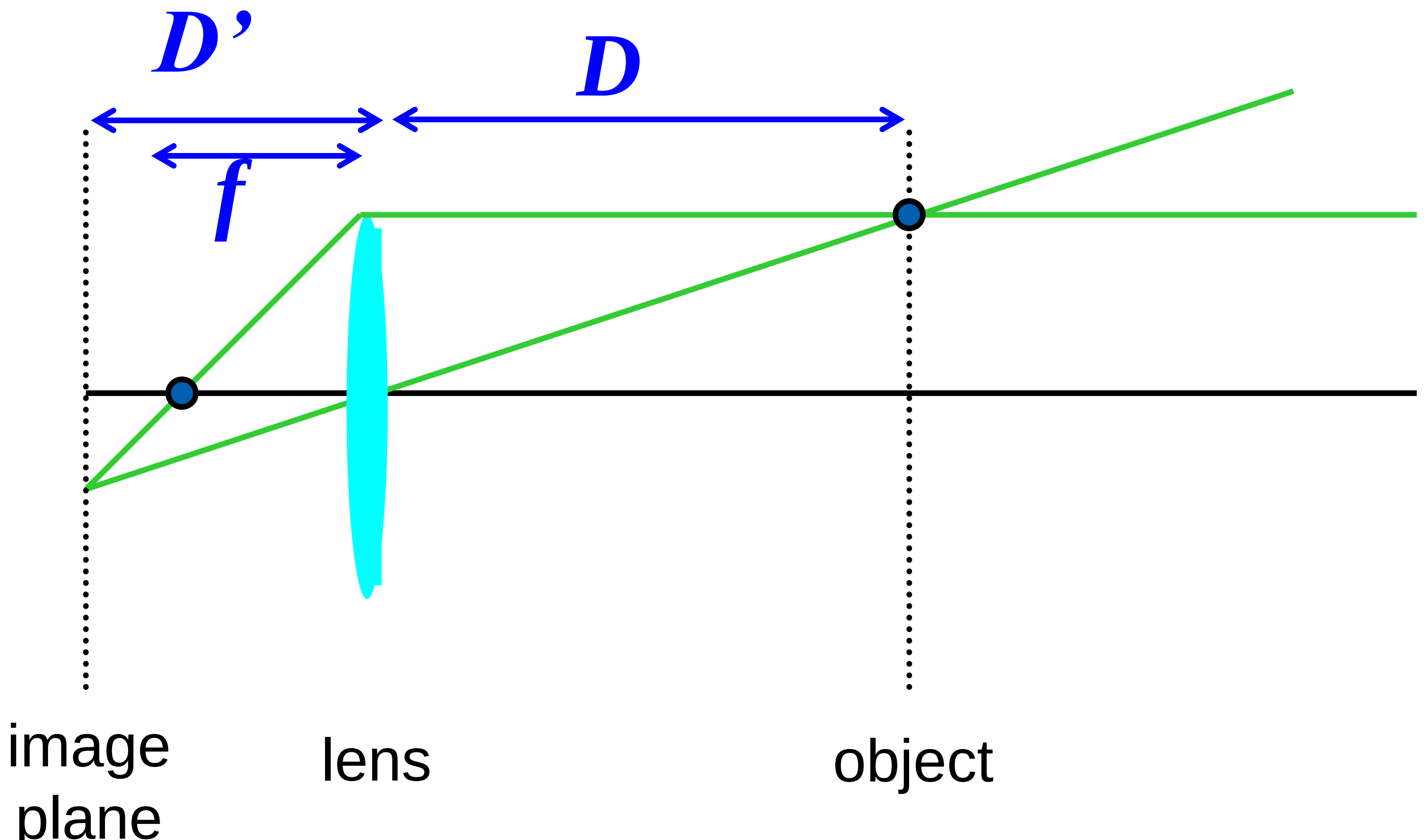


Steve Seitz

Thin lens formula

$$\boxed{\frac{1}{D'} + \frac{1}{D} = \frac{1}{f}}$$

Any point satisfying the thin lens equation is in focus.



Frédo Durand

Camera parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors

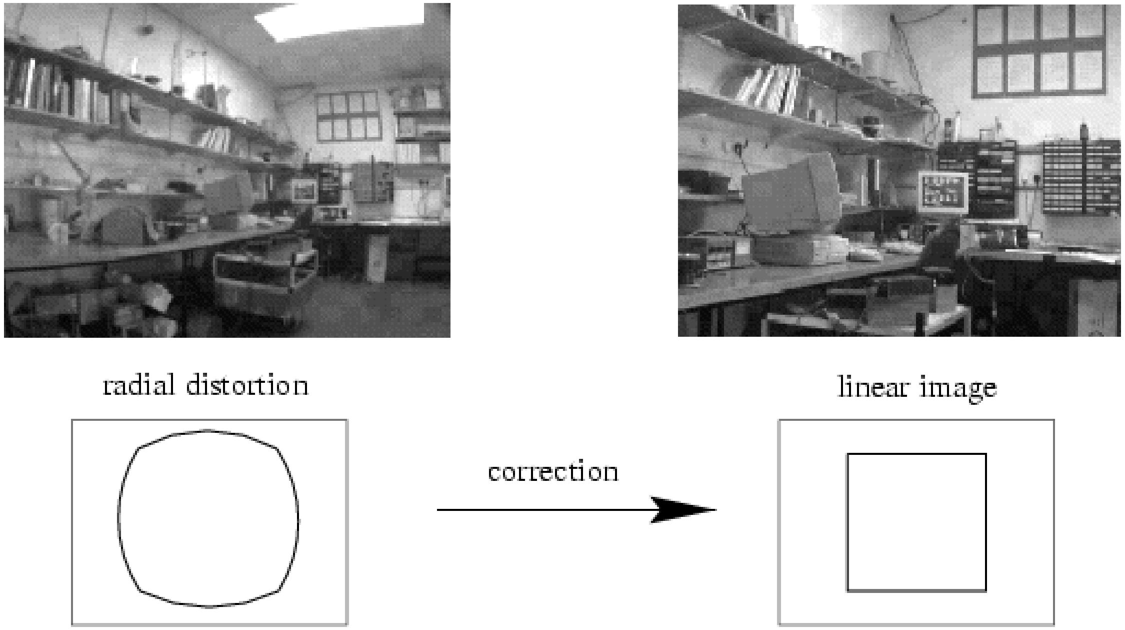
$$K = \begin{bmatrix} m_x & & \\ & m_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & & \beta_x \\ & \alpha_y & \beta_y \\ & & 1 \end{bmatrix}$$

Lazebnik

Camera parameters

- Intrinsic parameters
  - Principal point coordinates
  - Focal length
  - Pixel magnification factors
  - *Skew (non-rectangular pixels)*
  - *Radial distortion*

$$K = \begin{pmatrix} \alpha_x & \gamma & \beta_x \\ 0 & \alpha_y & \beta_y \\ 0 & 0 & 1 \end{pmatrix}$$



Lazebnik

Camera parameters

Intrinsic parameters

- Principal point coordinates
- Focal length
- Pixel magnification factors
- Skew (non-rectangular pixels)
- Radial distortion

Extrinsic parameters

- Rotation and translation relative to world coordinate system

Lazebnik

Camera calibration

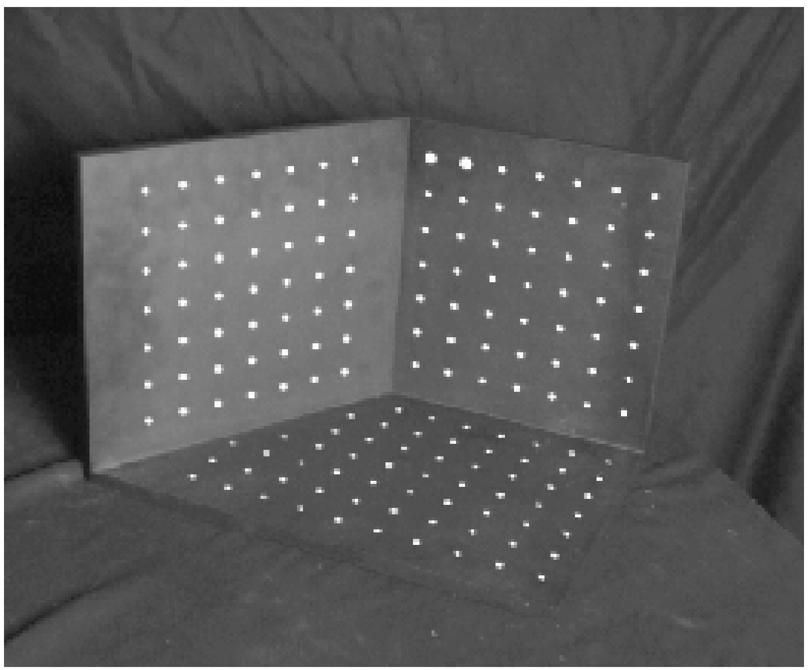
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

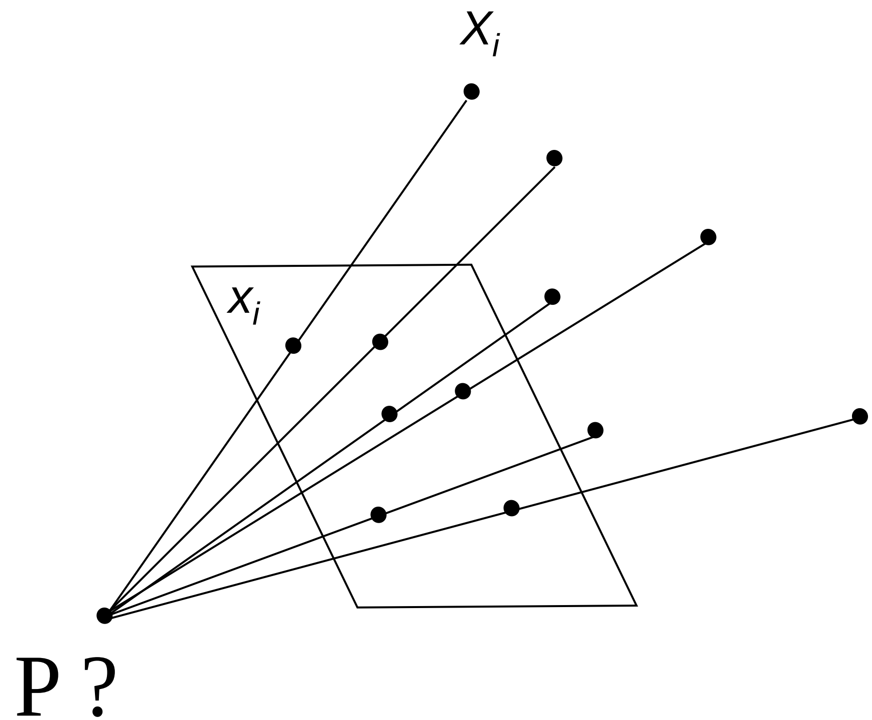
$$\begin{bmatrix}\lambda x \\ \lambda y \\ \lambda\end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}\begin{bmatrix}X \\ Y \\ Z \\ 1\end{bmatrix}$$

Source: D. Hoiem

Camera calibration

Given n points with known 3D coordinates X_i and known image projections x_i, estimate the camera parameters





Lazebnik

Camera calibration: Linear method

$$\lambda \mathrm{x}_i = \mathrm{P}\mathrm{X}_i \qquad \mathrm{x}_i \times \mathrm{P}\mathrm{X}_i = 0 \qquad \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathrm{P}_1^T \mathrm{X}_i \\ \mathrm{P}_2^T \mathrm{X}_i \\ \mathrm{P}_3^T \mathrm{X}_i \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & -\mathrm{X}_i^T & y_i \mathrm{X}_i^T \\ \mathrm{X}_i^T & 0 & -x_i \mathrm{X}_i^T \\ -y_i \mathrm{X}_i^T & x_i \mathrm{X}_i^T & 0 \end{bmatrix} \begin{pmatrix} \mathrm{P}_1 \\ \mathrm{P}_2 \\ \mathrm{P}_3 \end{pmatrix} = 0$$

Two linearly independent equations

Lazebnik

Camera calibration: Linear method

UNIVERSITÄT BONN

$$\begin{bmatrix} 0^T & X_1^T & -y_1 X_1^T \\ X_1^T & 0^T & -x_1 X_1^T \\ \cdots & \cdots & \cdots \\ 0^T & X_n^T & -y_n X_n^T \\ X_n^T & 0^T & -x_n X_n^T \end{bmatrix} \begin{pmatrix} P_1 \\ P_2 \\ P_3 \end{pmatrix} = 0 \qquad \mathrm{Ap} = 0$$

- P has 11 degrees of freedom (12 parameters, but scale is arbitrary)
- One 2D/3D correspondence gives us two linearly independent equations
- Homogeneous least squares
- 6 correspondences needed for a minimal solution

Lazebnik

Camera calibration: Linear method

Advantages: easy to formulate and solve

Disadvantages

- Doesn't directly tell you camera parameters
- Doesn't model radial distortion
- Can't impose constraints, such as known focal length and orthogonality

Non-linear methods are preferred

- Define error as difference between projected points and measured points
- Minimize error using Newton's method or other non-linear optimization

Source: D. Hoiem

Intrinsic Calibration with Planes

Use only one plane
- Print a pattern on a paper
- Attach the paper on a planar surface
- Show the plane freely a few times to the camera

Advantages
- Flexible
- Robust

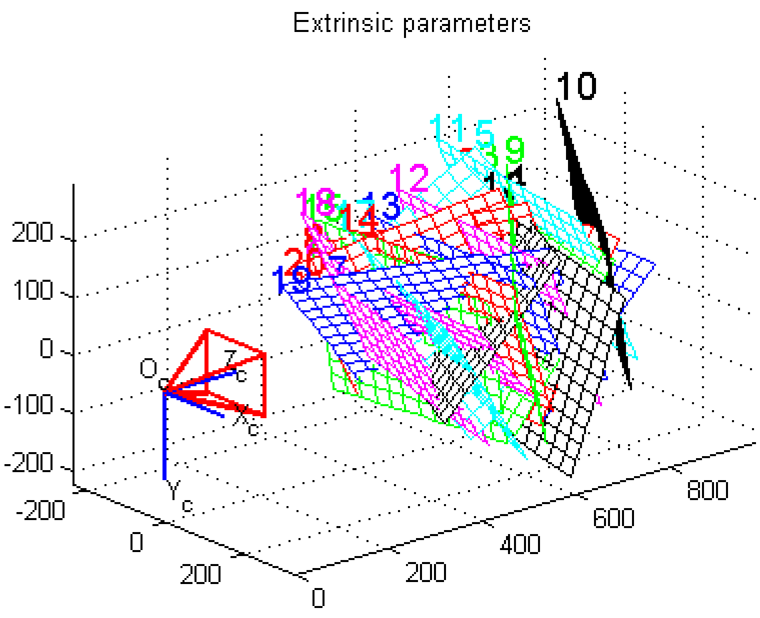
Implementation in OpenCV or Matlab toolbox:
http://www.vision.caltech.edu/bouguetj/calib_doc/

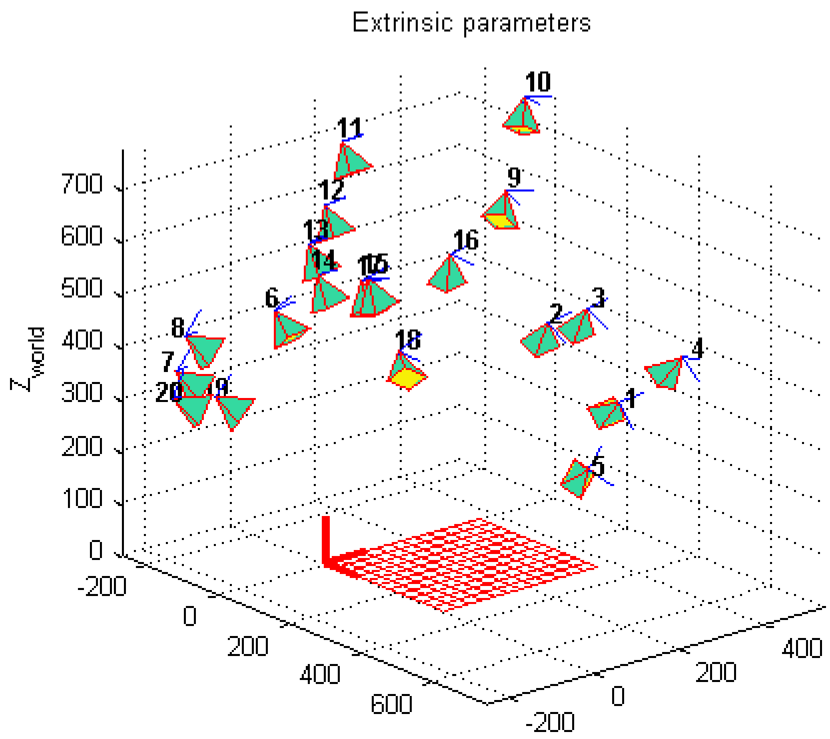
[Z. Zhang. Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. ICCV99]

Zhengyou Zhang

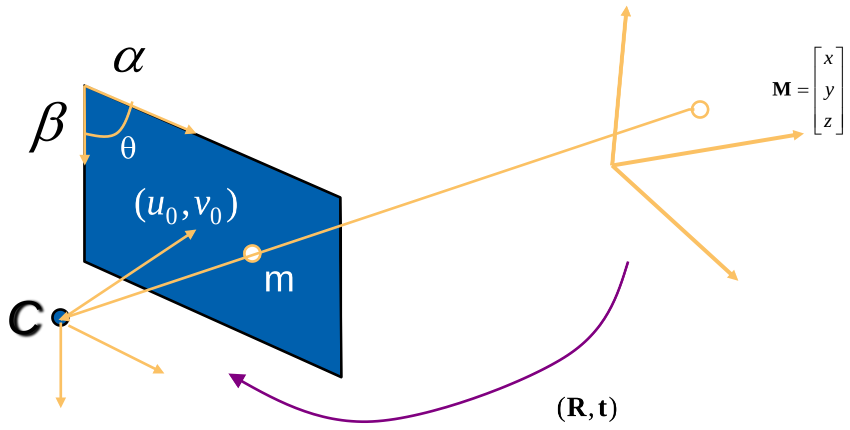
Calibration process

Extrinsic parameters





Camera Model

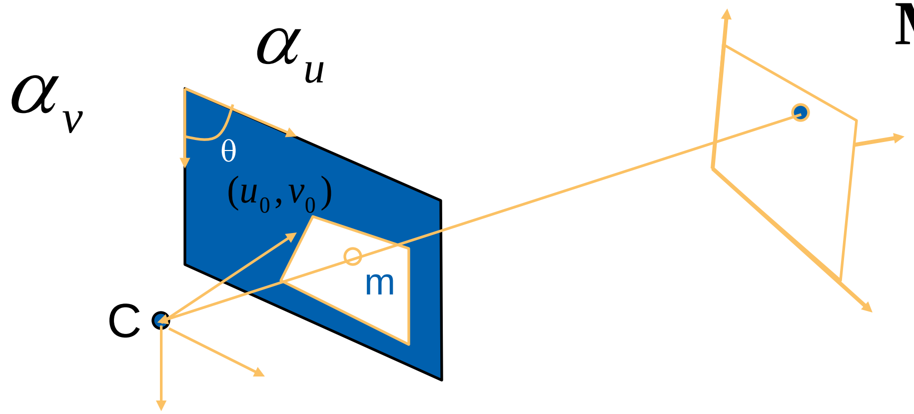


$$\underbrace{s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\tilde{\mathbf{m}}} = \underbrace{\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}}_{[\mathbf{R} \quad \mathbf{t}]} \underbrace{\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}}_{\tilde{\mathbf{M}}}$$

Zhengyou Zhang

Plane projection

For convenience, assume the plane at $z = 0$.



$$\mathbf{M} = \begin{bmatrix} x \\ y \end{bmatrix}$$

$$z = 0$$

The relation between image points and model points is then given by a homography **H**:

$$s\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{M}} \quad \text{with} \quad \mathbf{H} = \mathbf{A}[\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}] \qquad \mathbf{A}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

Zhengyou Zhang

Linear Equations

Let

$$\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix} \longleftarrow \text{symmetric}$$

Define $\mathbf{b} = [B_{11} \quad B_{12} \quad B_{22} \quad B_{13} \quad B_{23} \quad B_{33}]$ up to a scale factor

Rewrite

$$\mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2 = 0$$
$$\mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2$$

as linear equations:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} \qquad \begin{aligned} \mathbf{v}_{ij} = [&h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, \\ &h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T \end{aligned}$$

Zhengyou Zhang

Camera parameters

Intrinsic camera parameters

$$v_0 = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2)$$
$$\lambda = B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11}$$
$$\alpha = \sqrt{\lambda/B_{11}}$$
$$\beta = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)}$$
$$c = -B_{12}\alpha^2\beta/\lambda$$
$$u_0 = cv_0/\alpha - B_{13}\alpha^2/\lambda\ .$$

$$\underbrace{\begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{A}}$$

Rotation and translation

$$\mathbf{r}_1 = \lambda \mathbf{A}^{-1}\mathbf{h}_1,\ \mathbf{r}_2 = \lambda \mathbf{A}^{-1}\mathbf{h}_2,\ \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2,\ \mathbf{t} = \lambda \mathbf{A}^{-1}\mathbf{h}_3$$

$$\lambda = 1/\|\mathbf{A}^{-1}\mathbf{h}_1\| = 1/\|\mathbf{A}^{-1}\mathbf{h}_2\|$$

Distortion

Distortion model:

$$\breve{x} = x + x[k_1(x^2+y^2) + k_2(x^2+y^2)^2]$$
$$\breve{y} = y + y[k_1(x^2+y^2) + k_2(x^2+y^2)^2]$$

Centered at u_0 and v_0:

$$\breve{u} = u + (u-u_0)[k_1(x^2+y^2) + k_2(x^2+y^2)^2]$$
$$\breve{v} = v + (v-v_0)[k_1(x^2+y^2) + k_2(x^2+y^2)^2]$$

Solution:

$$\underbrace{\begin{bmatrix} (u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2 \end{bmatrix}}_{\text{D}} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \underbrace{\begin{bmatrix} \breve{u}-u \\ \breve{v}-v \end{bmatrix}}_{\text{d}}$$

$$\mathbf{k} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{d}$$

Non-linear optimization

In practice, closed-form solution is used for initialization of non-linear optimization problem

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\|\mathbf{m}_{ij}-\breve{\mathbf{m}}(\mathbf{A},k_1,k_2,\mathbf{R}_i,\mathbf{t}_i,\mathtt{M}_j)\|^2$$

Solved with Levenberg-Marquardt algorithm.

Without skew 2 at least images are needed, the more the better.

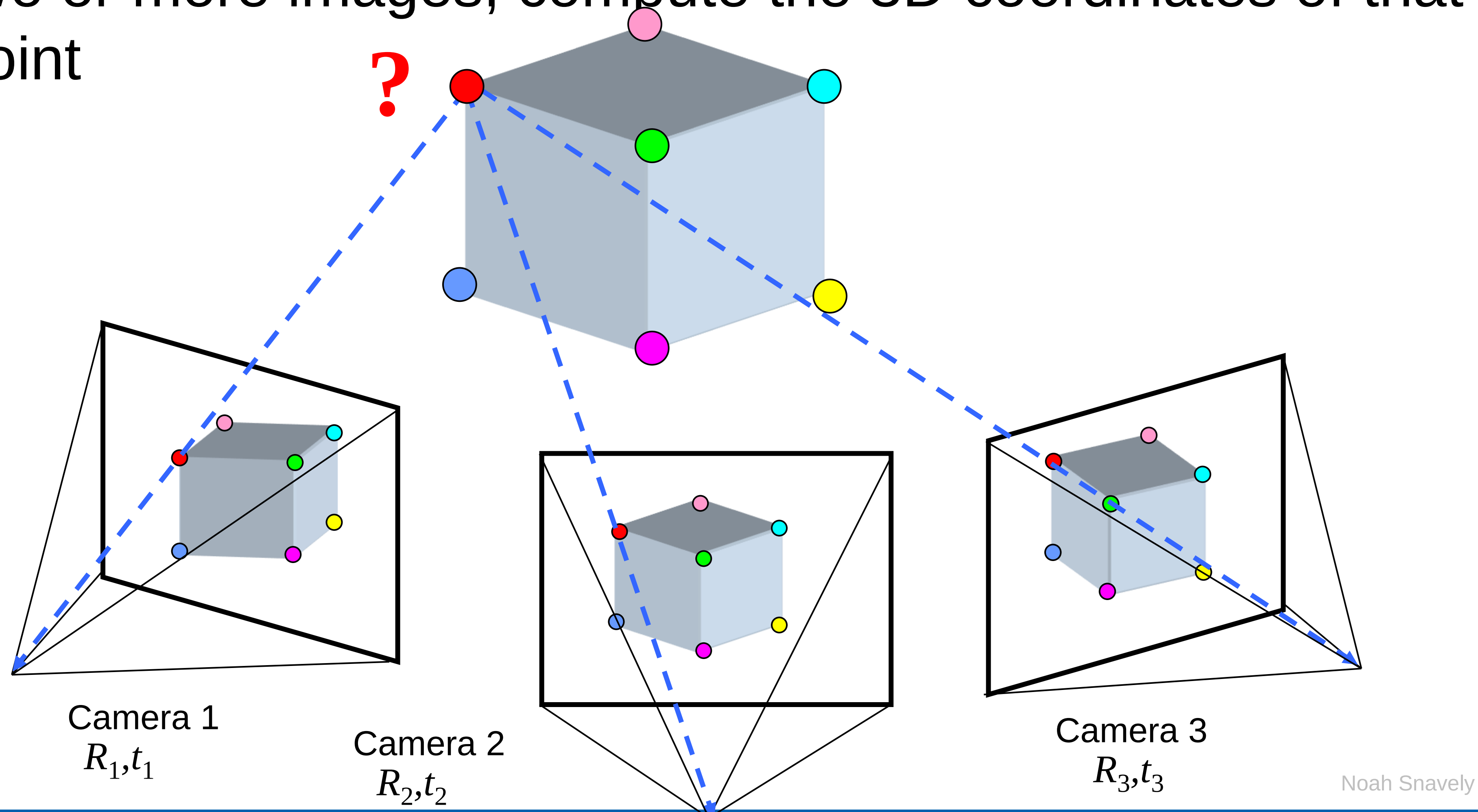
Solution

- Show the plane under *n* different orientations (n > 1)
- Estimate the n homography matrices
  (analytic solution followed by MLE)
- Solve analytically the 6 intermediate parameters *(defined up to a scale factor*)
- Extract the five intrinsic parameters
- Compute the extrinsic parameters
- Refine all parameters with MLE

Zhengyou Zhang

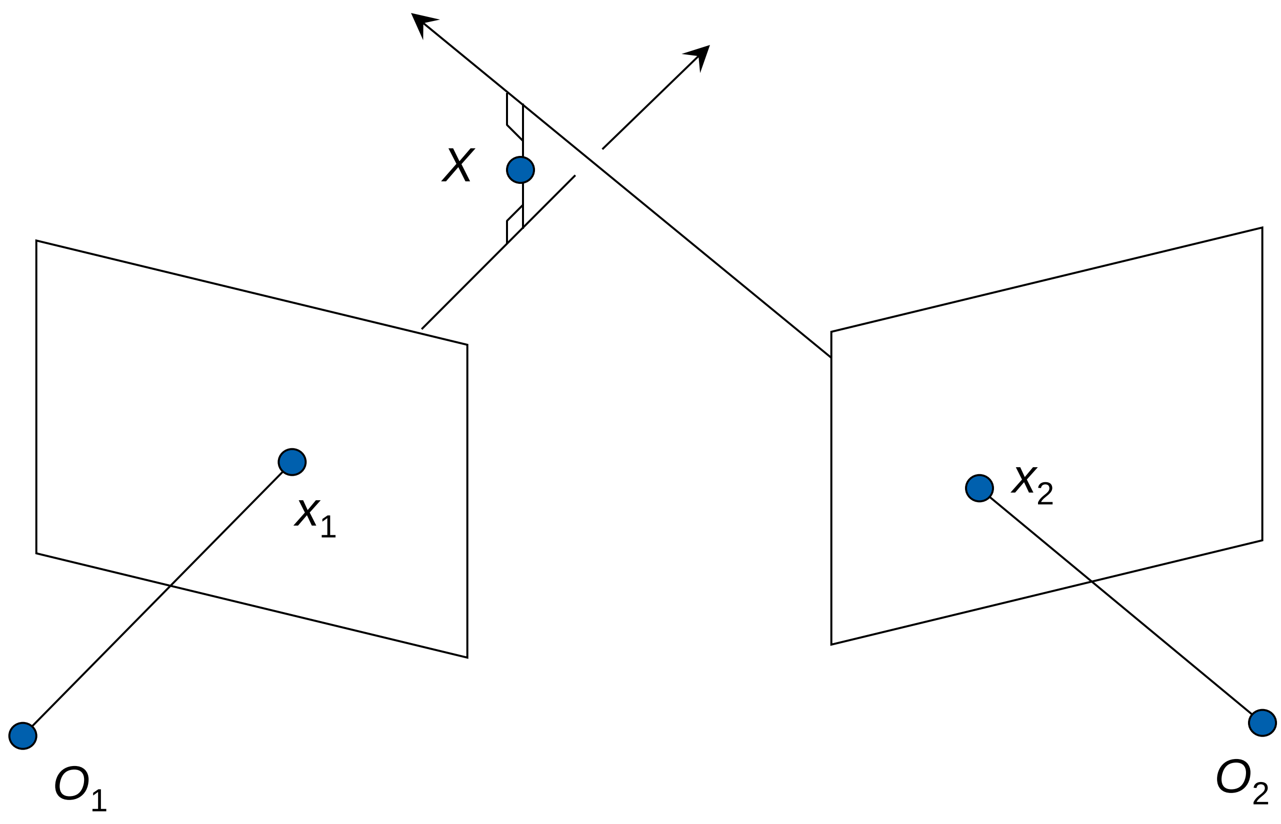
Multi-view geometry problems

Structure: Given projections of the same 3D point in two or more images, compute the 3D coordinates of that point



Triangulation: Geometric approach

Find shortest segment connecting the two viewing rays and let X be the midpoint of that segment



Lazebnik

Triangulation: Linear approach

$$\lambda_1 \mathrm{x}_1 = \mathrm{P}_1 \mathrm{X} \qquad \mathrm{x}_1 \times \mathrm{P}_1 \mathrm{X} = 0 \qquad [\mathrm{x}_{1\times}]\mathrm{P}_1 \mathrm{X} = 0$$

$$\lambda_2 \mathrm{x}_2 = \mathrm{P}_2 \mathrm{X} \qquad \mathrm{x}_2 \times \mathrm{P}_2 \mathrm{X} = 0 \qquad [\mathrm{x}_{2\times}]\mathrm{P}_2 \mathrm{X} = 0$$

Cross product as matrix multiplication:

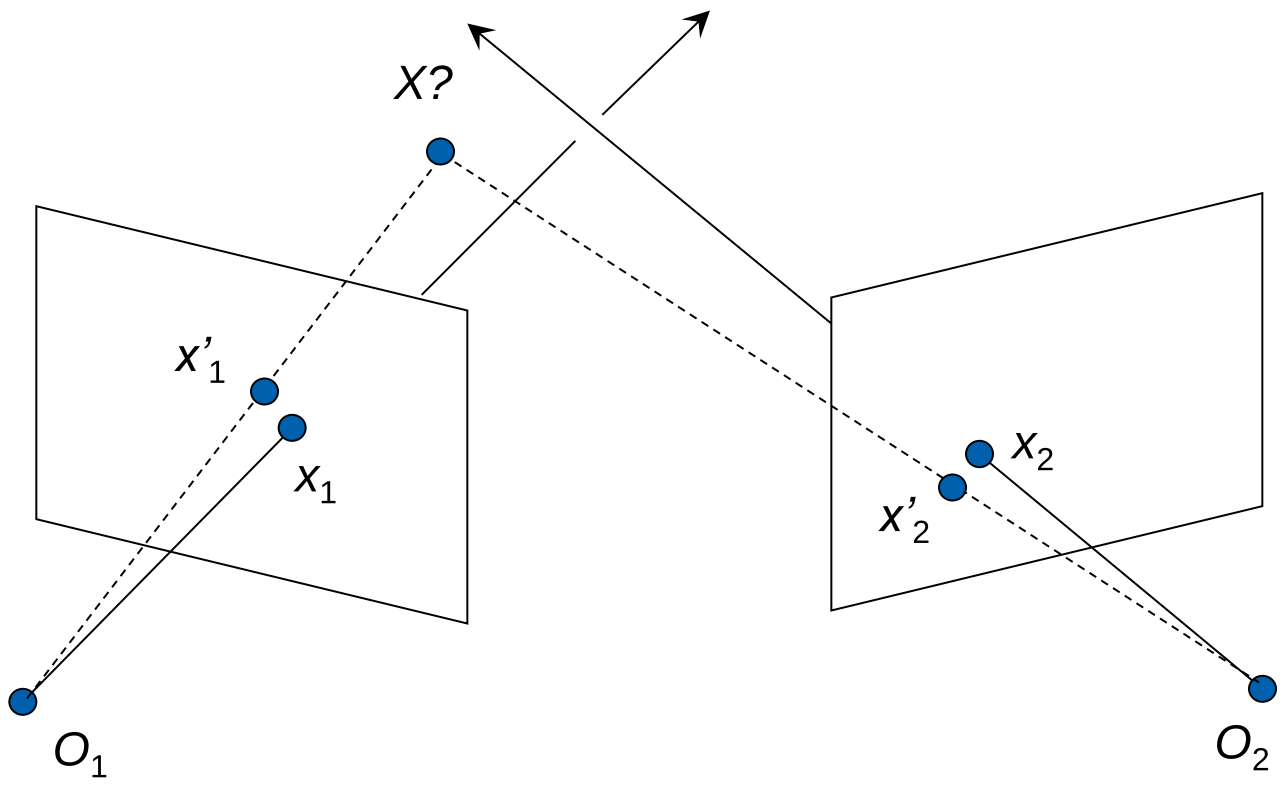
$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_\times]\mathbf{b}$$

Lazebnik

Triangulation: Nonlinear approach

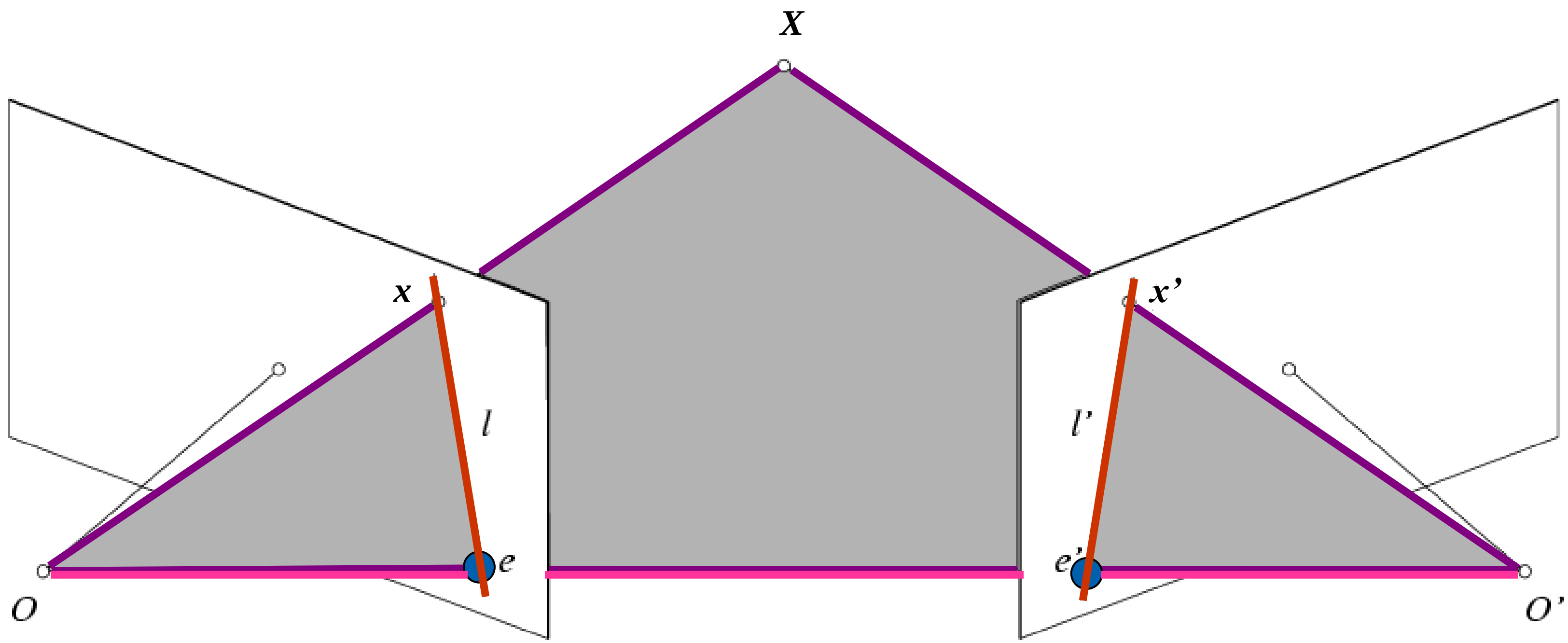
- Find X that minimizes

$$d^2(x_1, P_1 X) + d^2(x_2, P_2 X)$$



Lazebnik

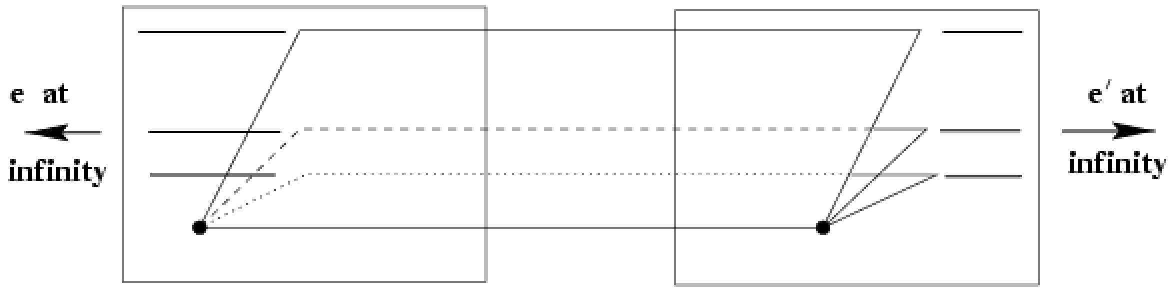
Epipolar geometry



- **Baseline** – line connecting the two camera centers
- **Epipolar Plane** – plane containing baseline (1D family)
- **Epipoles**
  = intersections of baseline with image planes
  = projections of the other camera center
- **Epipolar Lines** - intersections of epipolar plane with image planes (always come in corresponding pairs)

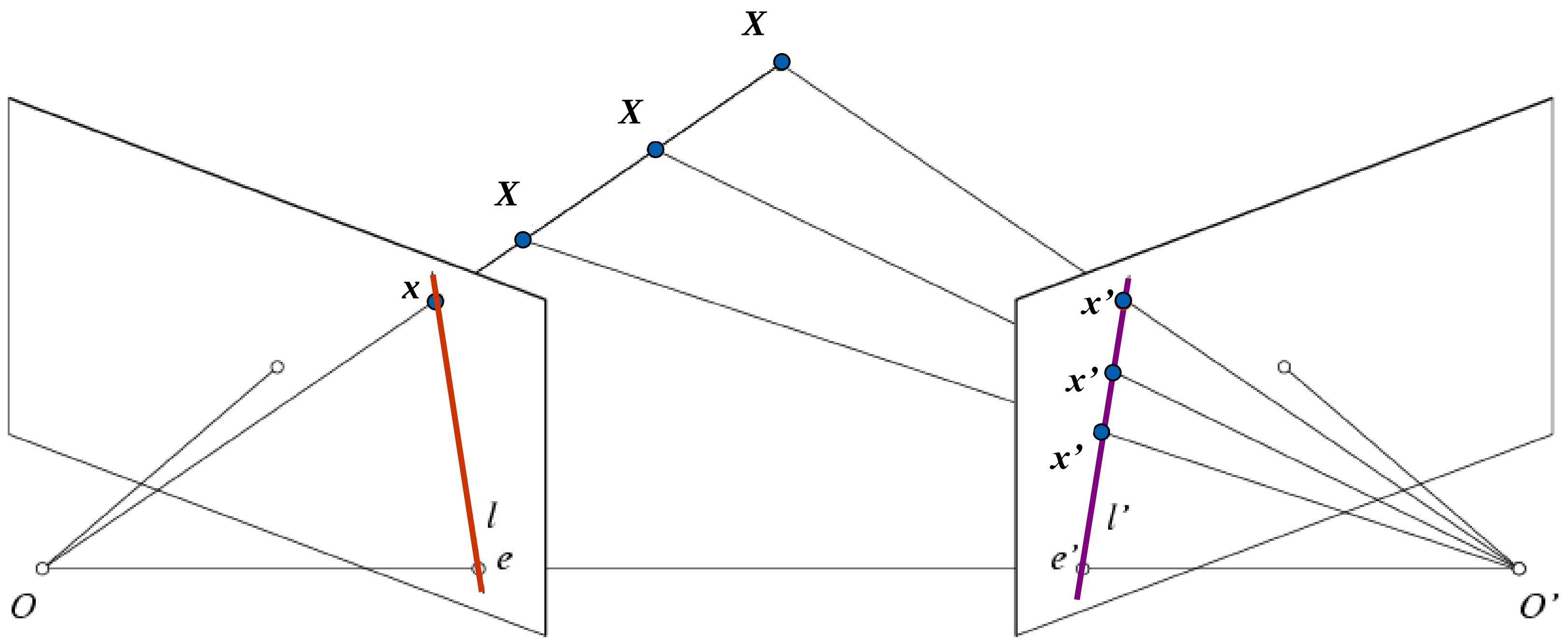
Lazebnik

Example: Motion parallel to image plane



Lazebnik

Epipolar constraint

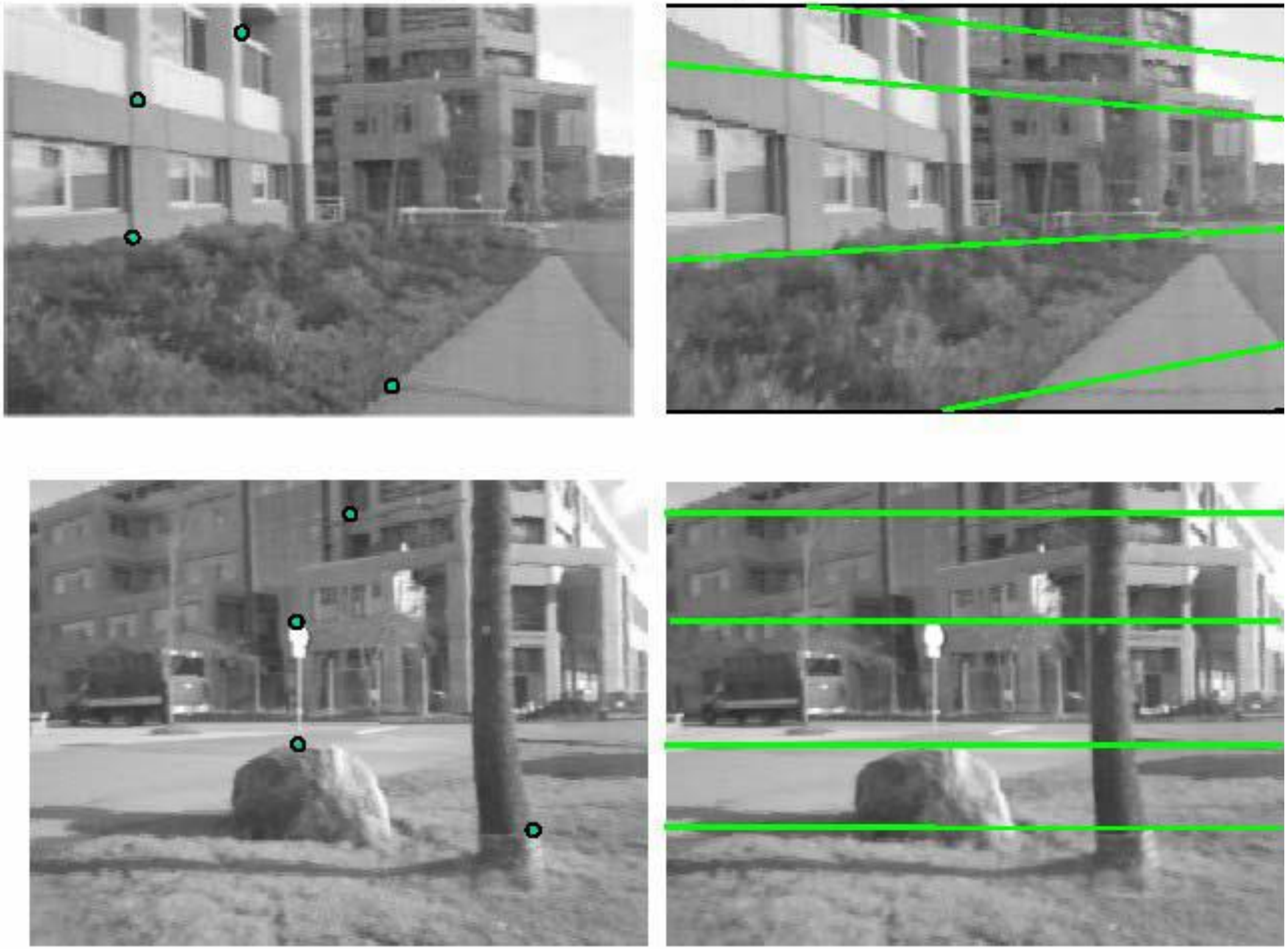


Potential matches for *x* have to lie on the corresponding epipolar line *l'*.

Potential matches for *x'* have to lie on the corresponding epipolar line *l*.

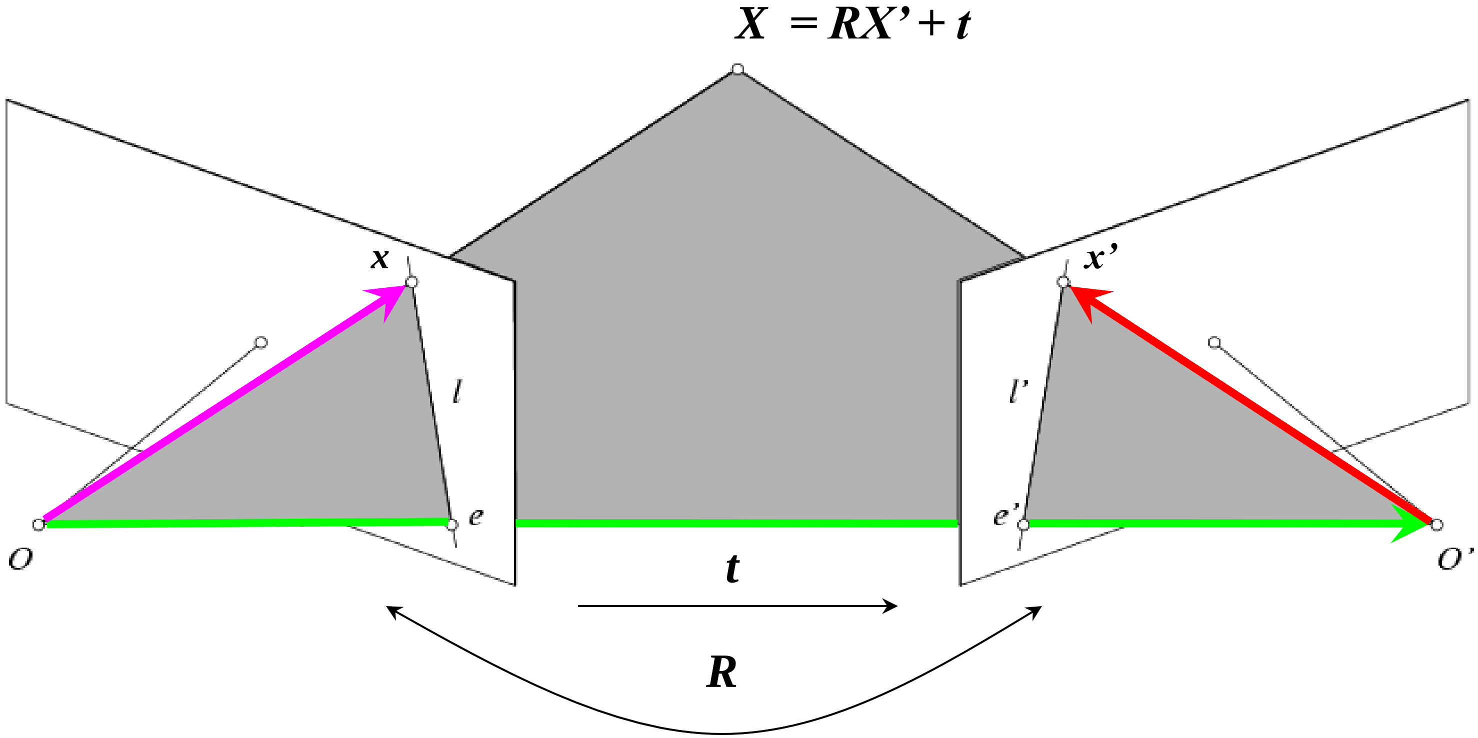
Lazebnik

Epipolar constraint example



Lazebnik

From geometry to algebra



$$\boxed{\mathbf{X}} = \mathbf{R}\boxed{\mathbf{X}'} + \boxed{\mathbf{T}}$$

$$\underbrace{\mathbf{T} \times \mathbf{X}}_{\text{Normal to the plane}} = \mathbf{T} \times \mathbf{R}\mathbf{X}' + \mathbf{T} \times \mathbf{T}$$

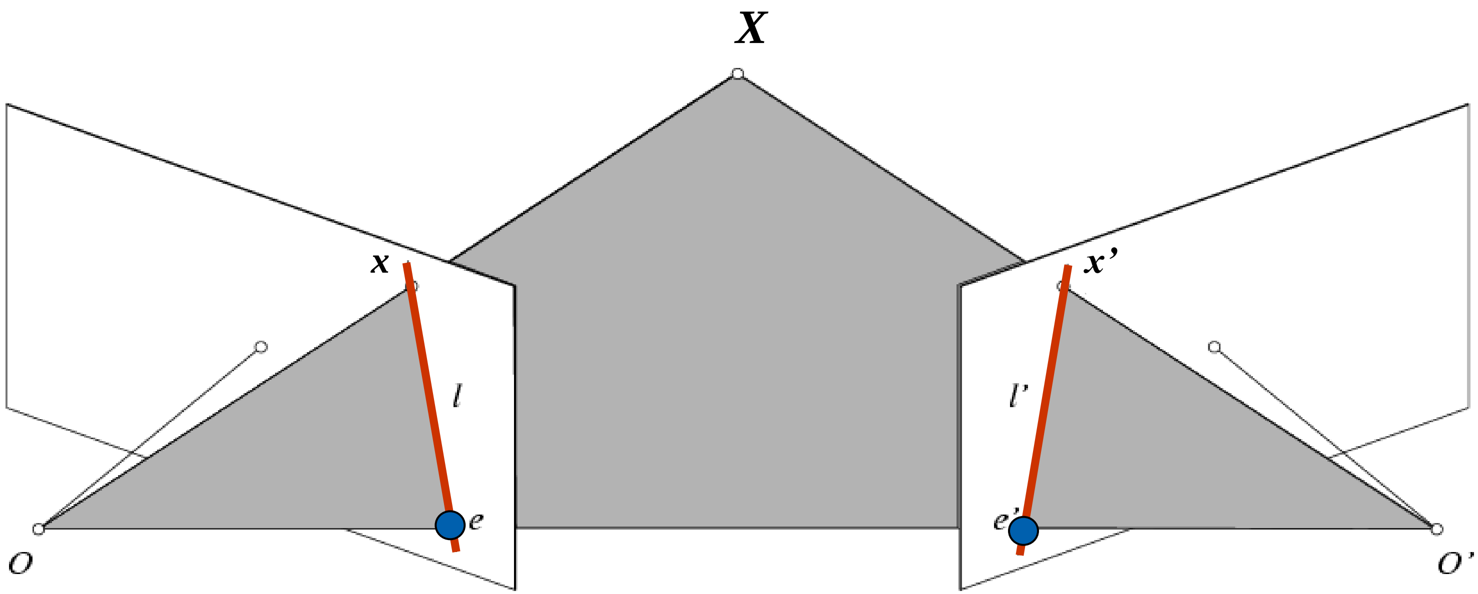
$$= \mathbf{T} \times \mathbf{R}\mathbf{X}'$$

$$\mathbf{X} \cdot (\mathbf{T} \times \mathbf{X}) = \mathbf{X} \cdot (\mathbf{T} \times \mathbf{R}\mathbf{X}')$$

$$= 0$$

Kristen Grauman

Epipolar constraint: Calibrated case

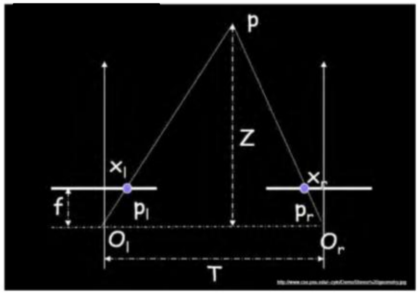


$$x \cdot [t \times (R x')] = 0 \quad \Rightarrow \quad x^T E x' = 0 \quad \text{with} \quad E = [t_\times] R$$

- E x' is the epipolar line associated with x' (l = E x')
- $E^T x$ is the epipolar line associated with x (l' = $E^T x$)
- E e' = 0 and $E^T e = 0$
- E is singular (rank two)
- E has five degrees of freedom

Lazebnik

Essential matrix example: parallel cameras



$$\mathbf{R} = \mathbf{I}$$

$$\mathbf{T} = [-d, 0, 0]^{\mathrm{T}}$$

$$\mathbf{E} = [\mathrm{T}_{\mathbf{x}}]\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & d \\ 0 & -d & 0 \end{pmatrix}$$

$$\mathbf{p} = [x, y, f]$$

$$\mathbf{p}' = [x', y', f]$$

$$\mathbf{p}'^{\mathrm{T}}\mathbf{E}\mathbf{p} = 0$$

$$\begin{bmatrix} x' & y' & f \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & d \\ 0 & -d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ f \end{bmatrix} = 0$$

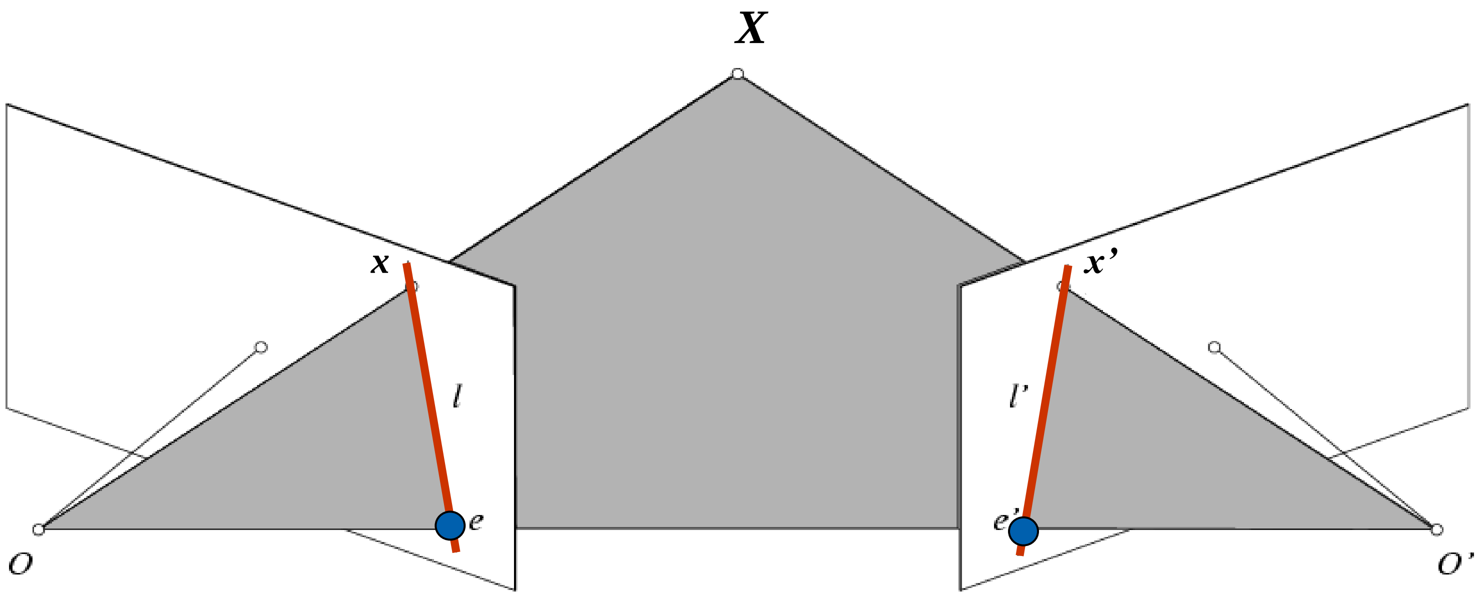
$$\Leftrightarrow \begin{bmatrix} x' & y' & f \end{bmatrix} \begin{bmatrix} 0 \\ df \\ -dy \end{bmatrix} = 0$$

$$\boxed{\Leftrightarrow y = y'}$$

For the parallel cameras, image of any point must lie on same horizontal line in each image plane.

Kristen Grauman

Epipolar constraint: Uncalibrated case



$$x^T F x' = 0 \quad \text{with} \quad F = K^{-T} E K'^{-1}$$

- $F x'$ is the epipolar line associated with x' ($l = F x'$)
- $F^T x$ is the epipolar line associated with x ($l' = F^T x$)
- $F e' = 0$ and $F^T e = 0$
- F is singular (rank two)
- F has seven degrees of freedom

Lazebnik

The eight-point algorithm

$\boldsymbol{x} = (u, v, 1)^T, \quad \boldsymbol{x}' = (u', v', 1)^T$

$$(u, v, 1)\begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = 0 \quad \Rightarrow \quad (uu', uv', u, vu', vv', v, u', v', 1)\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{pmatrix} = 0$$

$$\begin{pmatrix} u_1u_1' & u_1v_1' & u_1 & v_1u_1' & v_1v_1' & v_1 & u_1' & v_1' \\ u_2u_2' & u_2v_2' & u_2 & v_2u_2' & v_2v_2' & v_2 & u_2' & v_2' \\ u_3u_3' & u_3v_3' & u_3 & v_3u_3' & v_3v_3' & v_3 & u_3' & v_3' \\ u_4u_4' & u_4v_4' & u_4 & v_4u_4' & v_4v_4' & v_4 & u_4' & v_4' \\ u_5u_5' & u_5v_5' & u_5 & v_5u_5' & v_5v_5' & v_5 & u_5' & v_5' \\ u_6u_6' & u_6v_6' & u_6 & v_6u_6' & v_6v_6' & v_6 & u_6' & v_6' \\ u_7u_7' & u_7v_7' & u_7 & v_7u_7' & v_7v_7' & v_7 & u_7' & v_7' \\ u_8u_8' & u_8v_8' & u_8 & v_8u_8' & v_8v_8' & v_8 & u_8' & v_8' \end{pmatrix}\begin{pmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \end{pmatrix} = -\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \quad \Rightarrow$$

Minimize:

$$\sum_{i=1}^{N} (x_i^T F x_i')^2$$

under the constraint

$F_{33} = 1$

Lazebnik

The normalized eight-point algorithm

UNIVERSITÄT BONN

(Hartley, 1995)

- Center the image data at the origin, and scale it so the mean squared distance between the origin and the data points is 2 pixels
- Use the eight-point algorithm to compute F from the normalized points
- Enforce the rank-2 constraint (for example, take SVD of F and throw out the smallest singular value)
- Transform fundamental matrix back to original units: if T and T' are the normalizing transformations in the two images, than the fundamental matrix in original coordinates is $T^T F T'$

Lazebnik

From epipolar geometry to camera calibration

- Estimating the fundamental matrix is known as "weak calibration"
- If we know the calibration matrices of the two cameras, we can estimate the essential matrix: $E = K^T F K'$
- The essential matrix gives us the relative rotation and translation between the cameras, or their extrinsic parameters

Lazebnik

Estimating scene shape

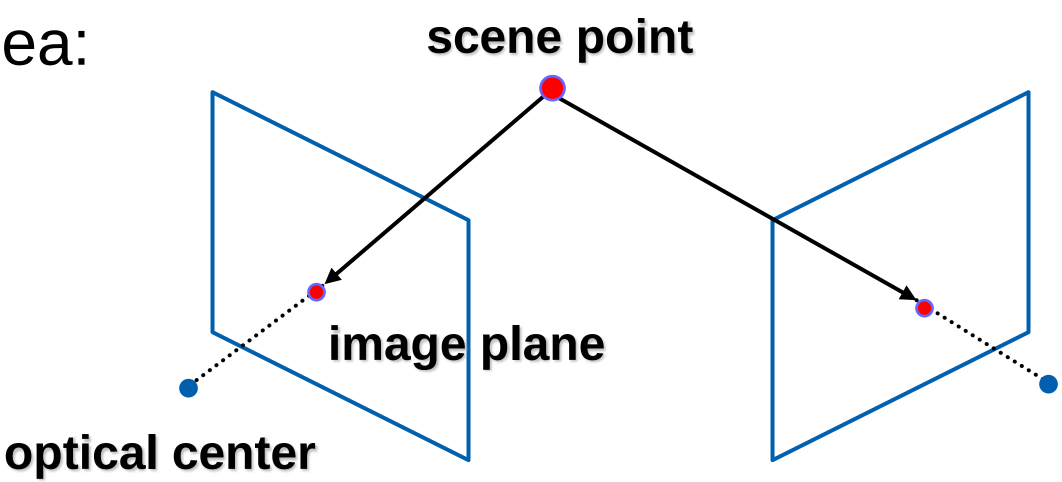
UNIVERSITÄT BONN

“Shape from X”: Shading, Texture, Focus, Motion…

Stereo:

- shape from “motion” between two views
- infer 3d shape of scene from two (multiple) images from different viewpoints

Main idea:



Kristen Grauman

Binocular stereo

Given a calibrated binocular stereo pair, fuse it to produce a depth image

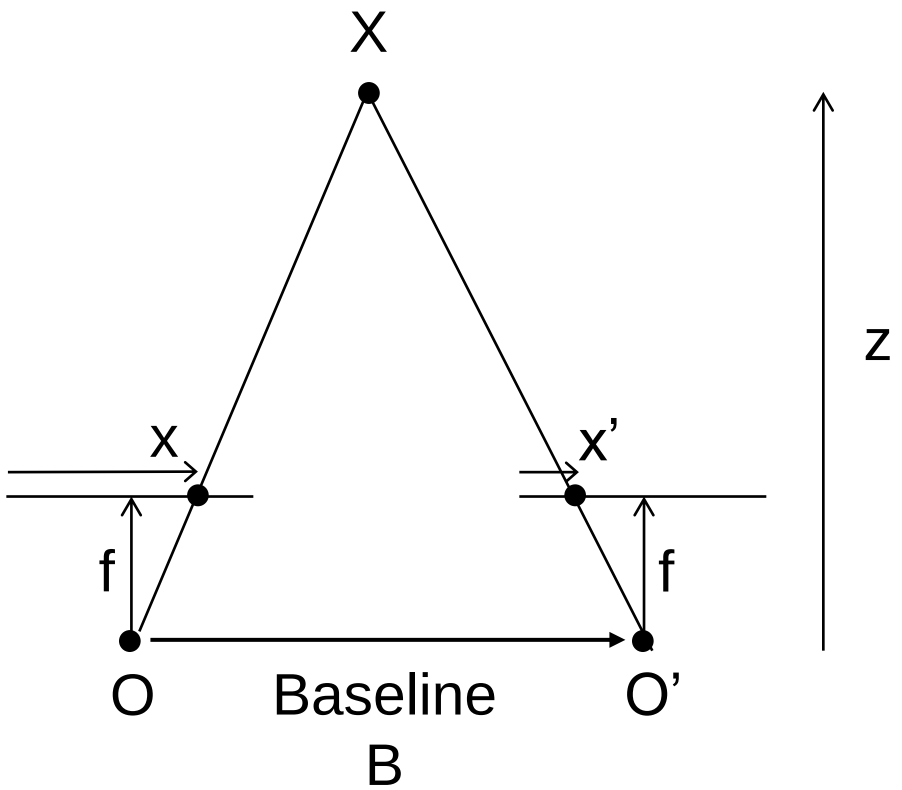
image 1

image 2

Dense depth map

Lazebnik

Depth from disparity



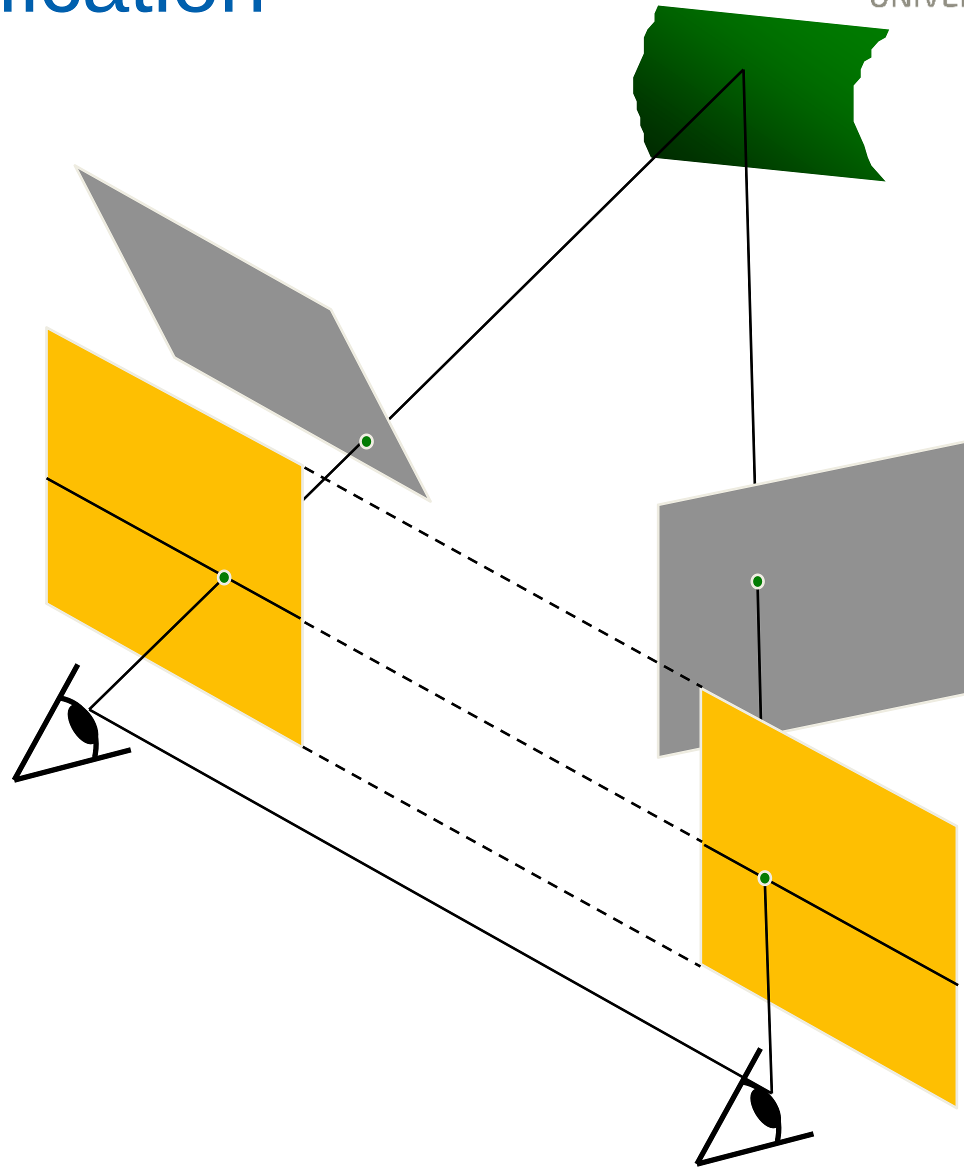
$$disparity = x - x' = \frac{B \cdot f}{z}$$

Disparity is inversely proportional to depth!

Lazebnik

Stereo image rectification

- Reproject image planes onto a common plane parallel to the line between optical centers
- Pixel motion is horizontal after this transformation
- Two homographies (3x3 transform), one for each input image reprojection
- C. Loop and Z. Zhang. Computing Rectifying Homographies for Stereo Vision. IEEE Conf. Computer Vision and Pattern Recognition, 1999.



Lazebnik

Rectification example

$\mathbf{H}_p$

Lazebnik

Rectification example

$\mathbf{H}_p$

$\mathbf{H}_r\mathbf{H}_p$

Lazebnik

Rectification example

$\mathbf{H}_r\mathbf{H}_p$

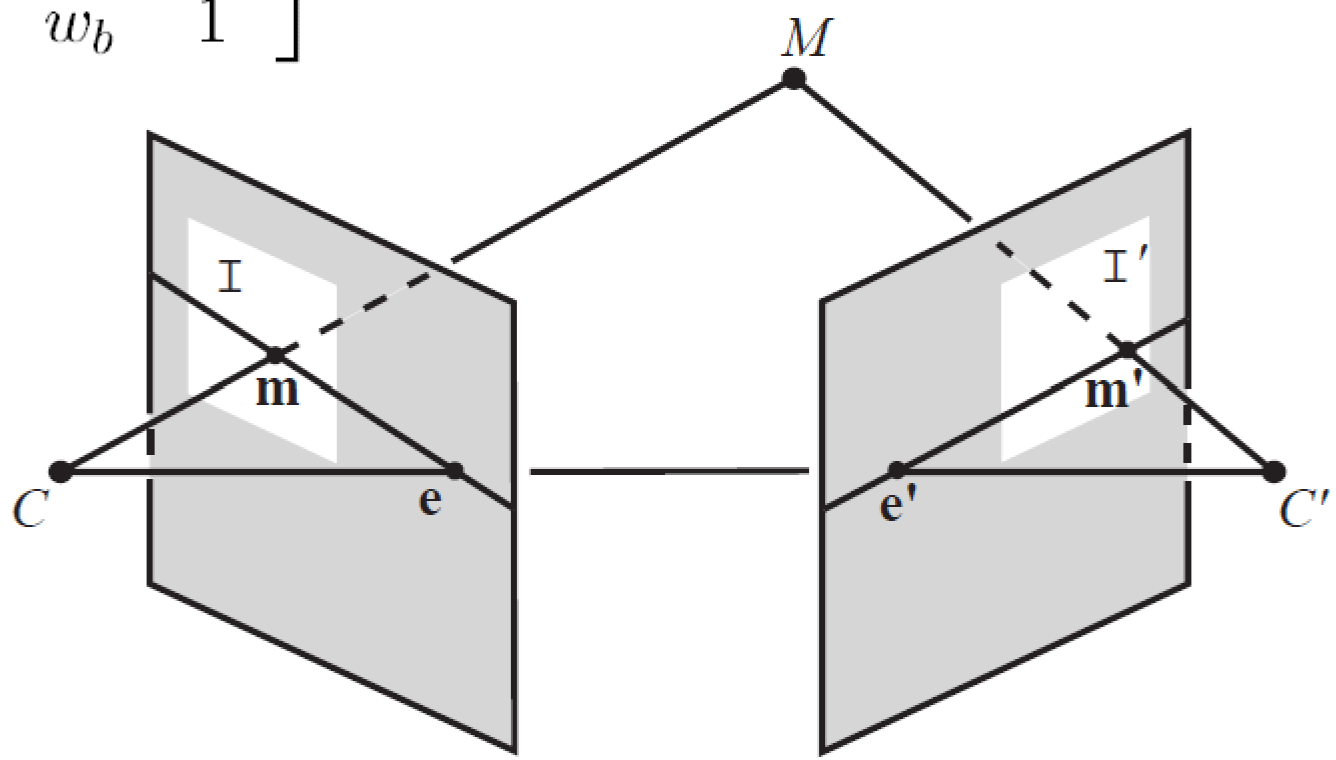
$\mathbf{H}_s\mathbf{H}_r\mathbf{H}_p$

Lazebnik

Rectification

Estimate two homographies **H** and **H’**

$$\mathbf{H} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix}$$

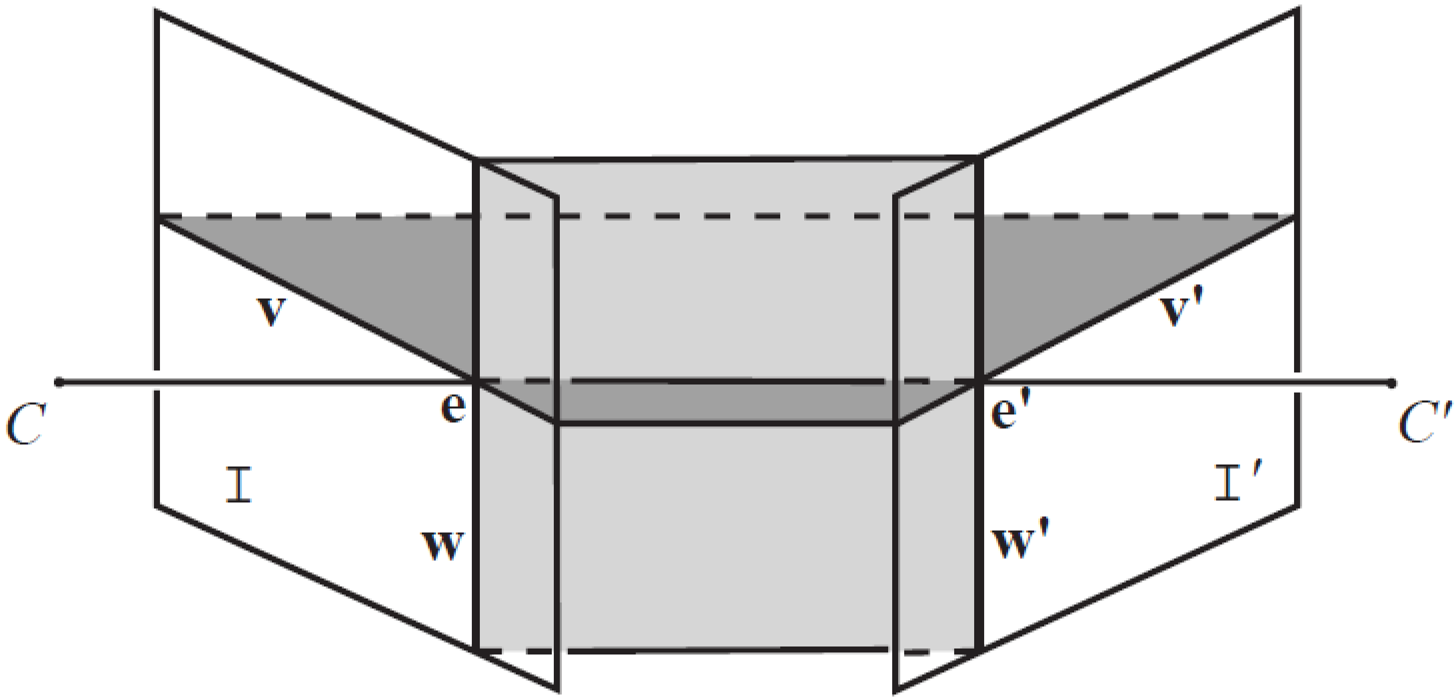


Rectification

- Map epipoles to infinity (1,0,0) (canonical form)
- Fundamental matrix after rectification:

$$\bar{\mathbf{F}} = [\,\mathbf{i}\,]_{\times} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

Rectification



$$\mathbf{H} = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \\ \mathbf{w}^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & w_c \end{bmatrix}$$

$$\mathbf{He} = \begin{bmatrix} \mathbf{u}^T\mathbf{e} & \mathbf{v}^T\mathbf{e} & \mathbf{w}^T\mathbf{e} \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T$$

Decompose homography

- Decompose **H** into affine transformation $\mathbf{H}_a$ and projective transformation $\mathbf{H}_p$
- Decompose affine trans. $\mathbf{H}_a$ into similarity trans. $\mathbf{H}_r$ and shearing trans. $\mathbf{H}_s$
- $\mathbf{H} = \mathbf{H}_a\mathbf{H}_p = \mathbf{H}_s\mathbf{H}_r\mathbf{H}_p$

$$\mathbf{H} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix}$$

$$\mathbf{H}_s = \begin{bmatrix} s_a & s_b & s_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{H}_r = \begin{bmatrix} v_b - v_c w_b & v_c w_a - v_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ w_a & w_b & 1 \end{bmatrix}$$

Estimate projective transformation

- Estimate $\mathbf{H}_p$:
- All pixels from images

$$\mathbf{H}_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ w_a & w_b & 1 \end{bmatrix}$$

$$\mathbf{P} = \begin{bmatrix} p_{1,u} - p_{c,u} & p_{2,u} - p_{c,u} & \cdots & p_{n,u} - p_{c,u} \\ p_{1,v} - p_{c,v} & p_{2,v} - p_{c,v} & \cdots & p_{n,v} - p_{c,v} \\ 0 & 0 & \cdots & 0 \end{bmatrix} \quad \mathbf{p}_c = \frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i$$

- Minimize:

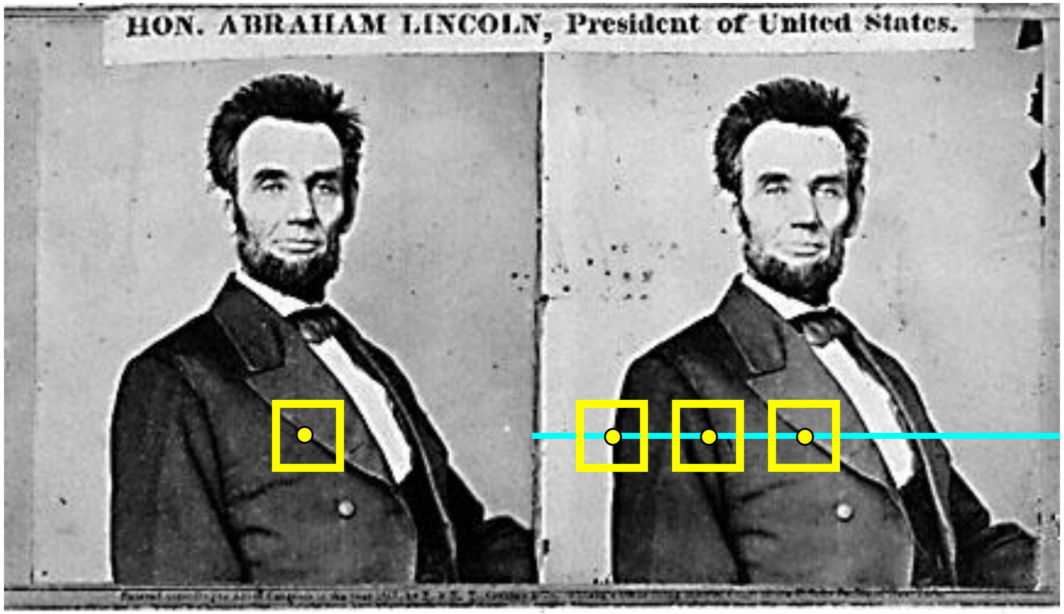
$$\frac{\mathbf{z}^T \overbrace{[\mathbf{e}]_\times^T \mathbf{P}\mathbf{P}^T [\mathbf{e}]_\times}^{\mathbf{A}} \mathbf{z}}{\mathbf{z}^T \underbrace{[\mathbf{e}]_\times^T \mathbf{p}_c \mathbf{p}_c^T [\mathbf{e}]_\times}_{\mathbf{B}} \mathbf{z}} + \frac{\mathbf{z}^T \overbrace{\mathbf{F}^T \mathbf{P}'\mathbf{P}'^T \mathbf{F}}^{\mathbf{A}'} \mathbf{z}}{\mathbf{z}^T \underbrace{\mathbf{F}^T \mathbf{p}'_c \mathbf{p}'^T_c \mathbf{F}}_{\mathbf{B}'} \mathbf{z}}$$

$$\mathbf{w} = [\mathbf{e}]_\times \mathbf{z}$$

$$\mathbf{w}' = \mathbf{F}\mathbf{z}$$

- Cholesky decomposition: $\mathbf{A} = \mathbf{D}^T\mathbf{D}$
- **y** is eigenvector with highest eigenvalue of $\mathbf{D}^{-T}\mathbf{B}\mathbf{D}^{-1}$
- **w and w'** is given by $\mathbf{w} = [\mathbf{e}]_\times \mathbf{z}$ $\quad \mathbf{w}' = \mathbf{F}\mathbf{z}$ $\quad \mathbf{z} = \mathbf{D}^{-1}\mathbf{y}$

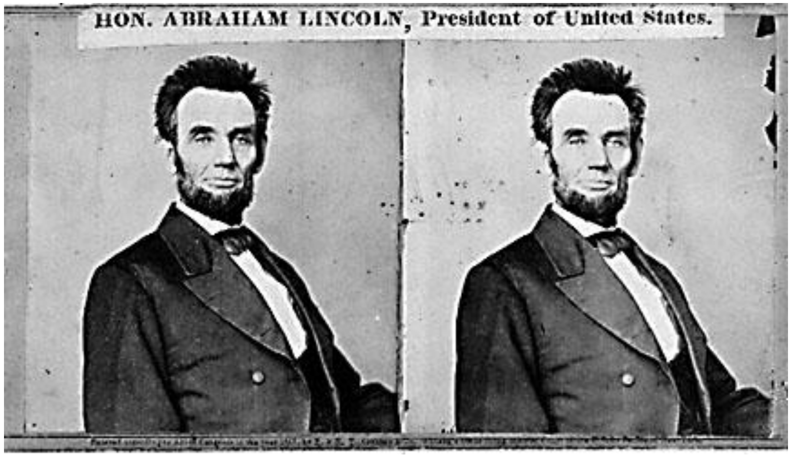
Basic stereo matching algorithm



- If necessary, rectify the two stereo images to transform epipolar lines into scanlines
- For each pixel x in the first image
  - Find corresponding epipolar scanline in the right image
  - Examine all pixels on the scanline and pick the best match x'
  - Compute disparity x-x' and set depth(x) = B*f/(x-x')

Lazebnik

Failures of correspondence search



Textureless surfaces

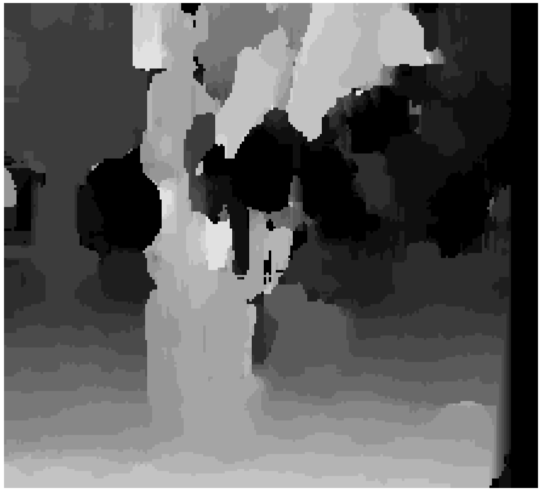
Occlusions, repetition

Non-Lambertian surfaces, specularities

Lazebnik

Effect of window size

W = 3



W = 20

– Smaller window
  - More detail
  - More noise
– Larger window
  - Smoother disparity maps
  - Less detail

Lazebnik

Non-local constraints

Uniqueness

- For any point in one image, there should be at most one matching point in the other image

Ordering

- Corresponding points should be in the same order in both views

Smoothness

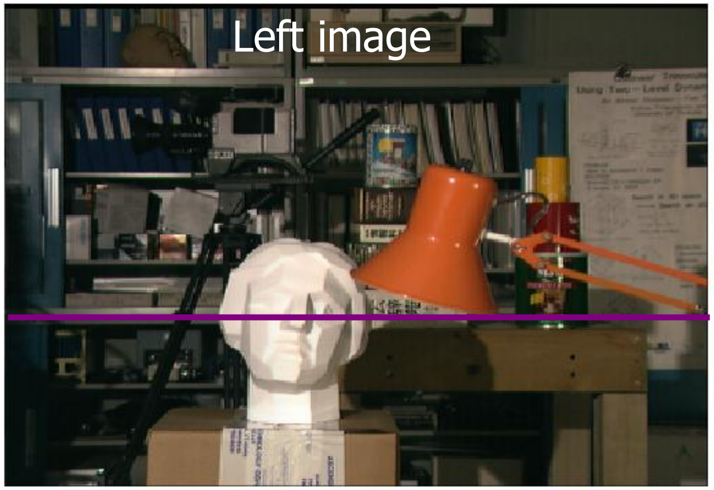
- We expect disparity values to change slowly (for the most part)

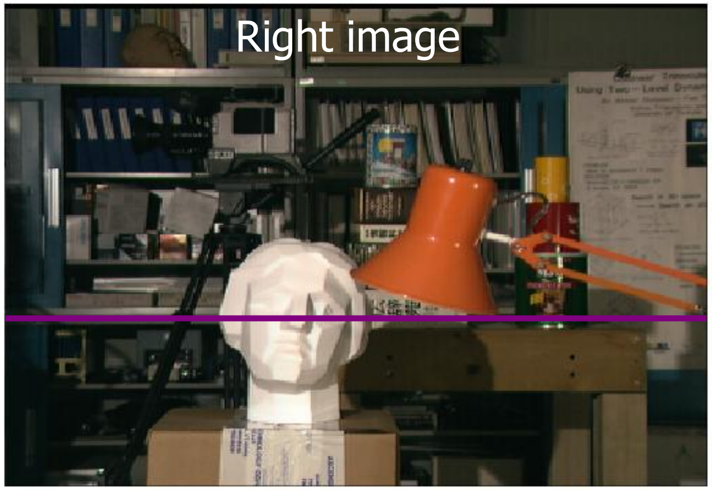
Lazebnik

Scanline stereo

Try to coherently match pixels on the entire scanline

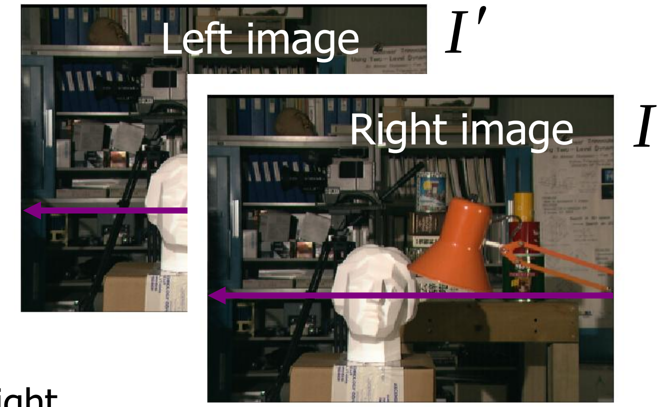
Different scanlines are still optimized independently

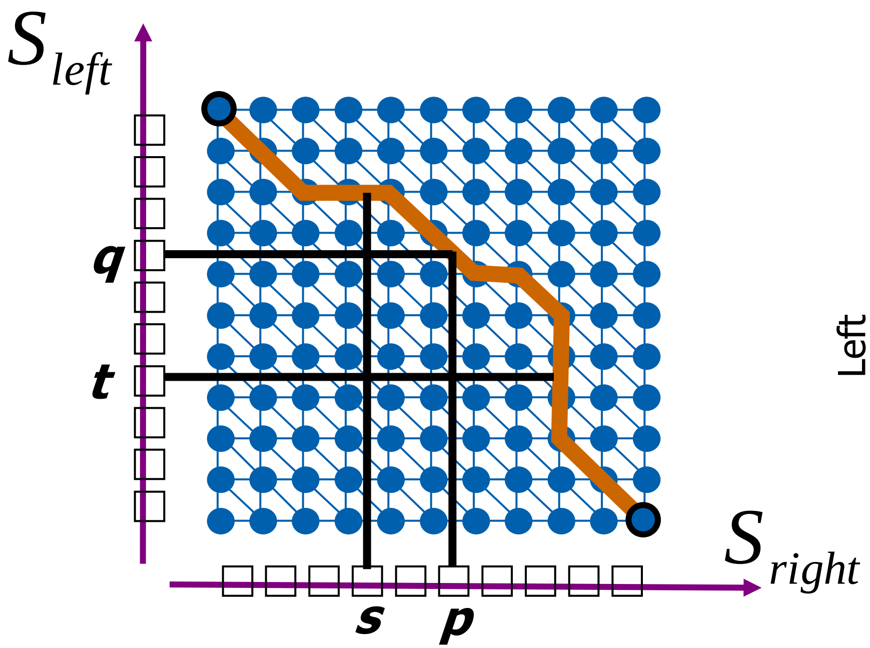


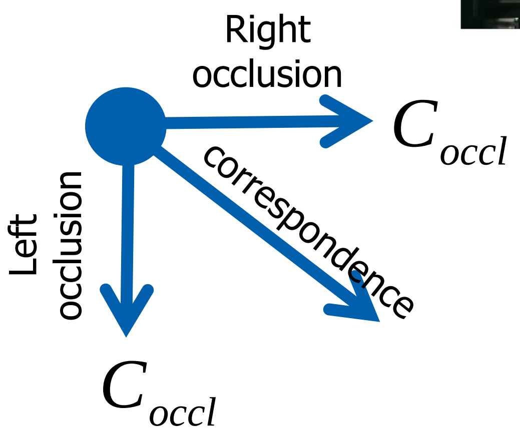


Lazebnik

“Shortest paths” for scan-line stereo







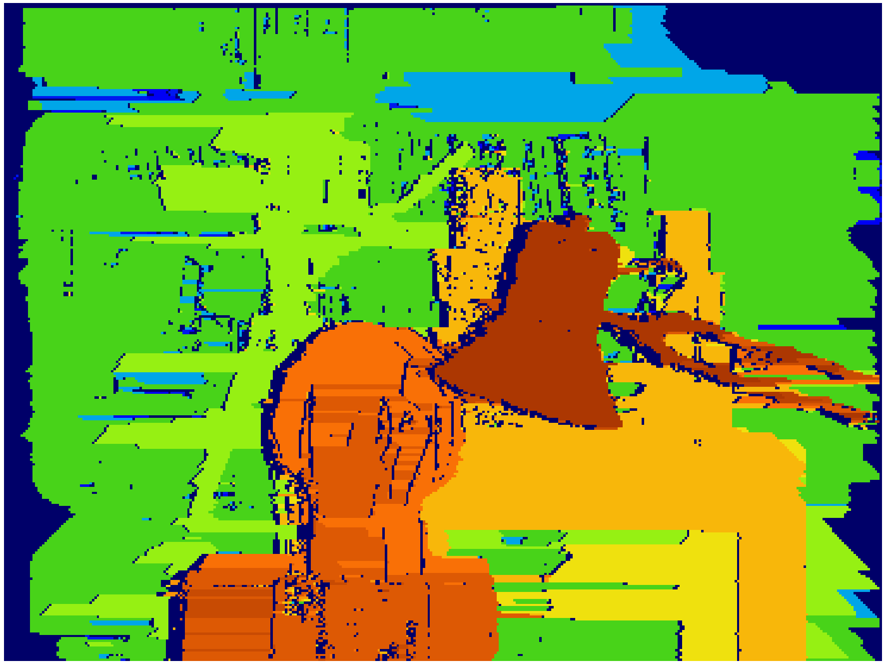
Can be implemented with dynamic programming

Ohta & Kanade ’85, Cox et al. ‘96

Y. Boykov

Coherent stereo on 2D grid

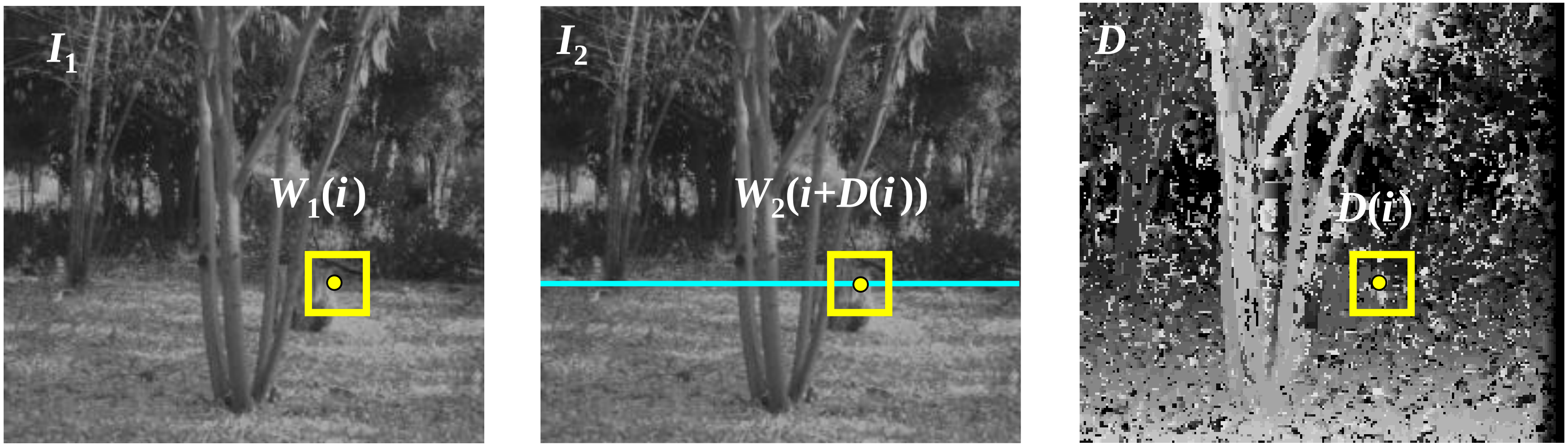
Scanline stereo generates streaking artifacts



Can’t use dynamic programming to find spatially coherent disparities/ correspondences on a 2D grid

Lazebnik

Stereo matching as energy minimization



$$E(D) = \underbrace{\sum_i \left(W_1(i) - W_2(i + D(i))\right)^2}_{\textit{data term}} + \lambda \underbrace{\sum_{\text{neighbors } i,j} \rho\left(D(i) - D(j)\right)}_{\textit{smoothness term}}$$

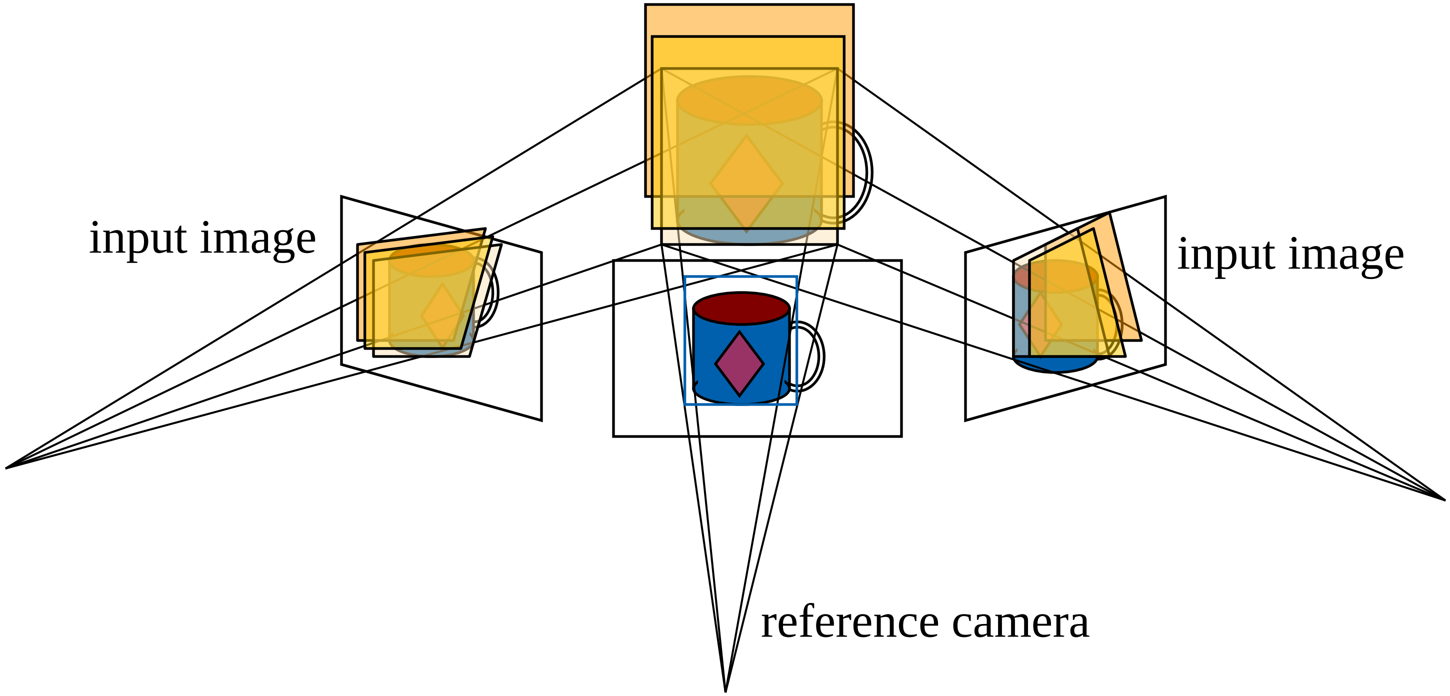
Energy functions of this form can be minimized using graph cuts

Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Lazebnik

Plane Sweep Stereo

- Choose a reference view
- Sweep family of planes at different depths with respect to the reference camera



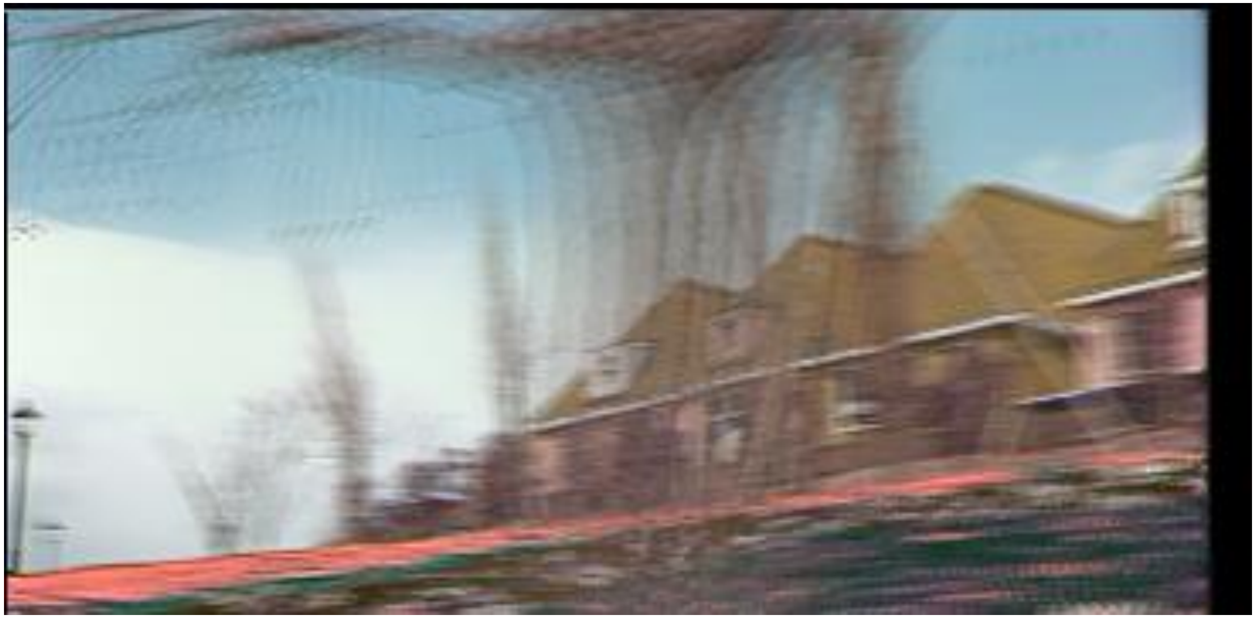
Each plane defines a homography warping each input image into the reference view

R. Collins. A space-sweep approach to true multi-image matching. CVPR 1996.

Lazebnik

Plane Sweep Stereo

- For each depth plane
  - For each pixel in the composite image stack, compute the variance

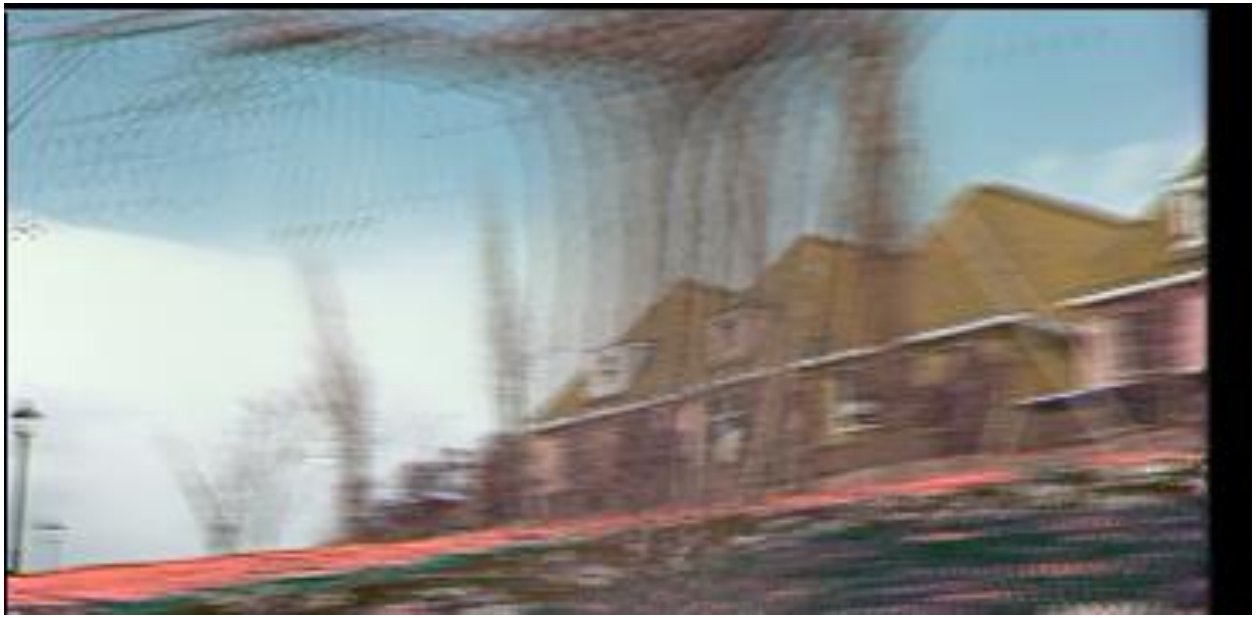


- For each pixel, select the depth that gives the lowest variance

Lazebnik

Plane Sweep Stereo

- For each depth plane
  - For each pixel in the composite image stack, compute the variance



- For each pixel, select the depth that gives the lowest variance

- Can be accelerated using graphics hardware

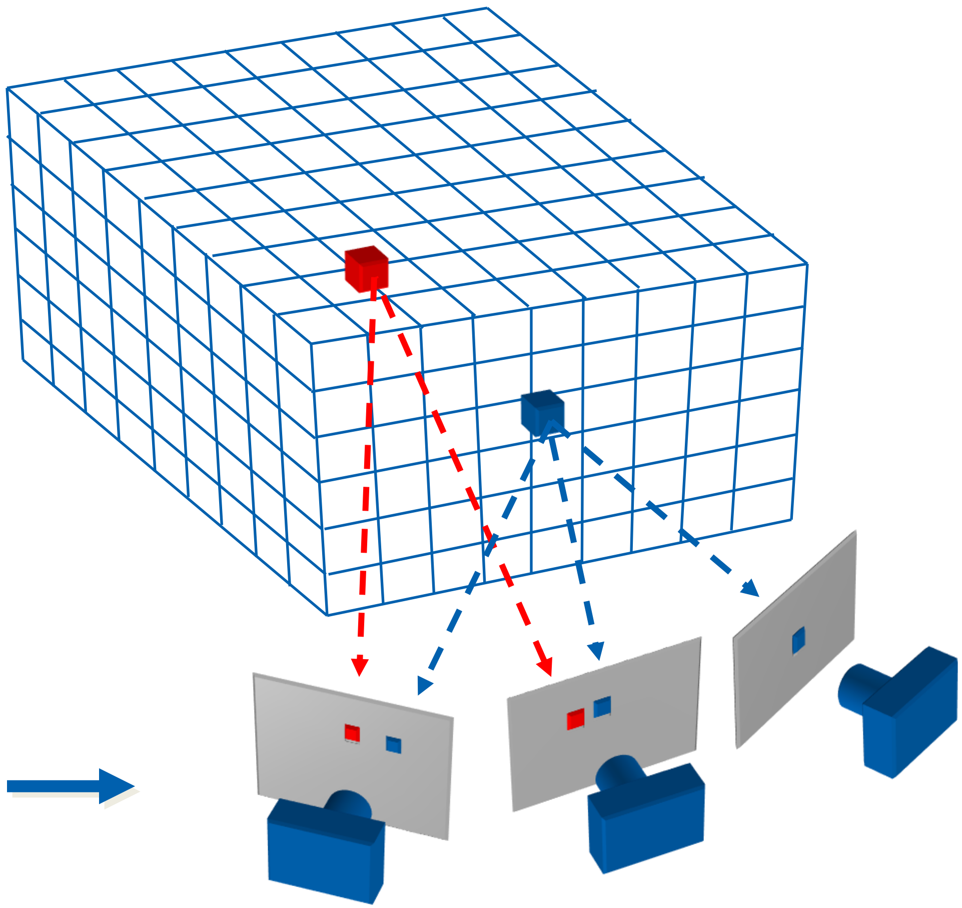
R. Yang and M. Pollefeys. *Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware*, CVPR 2003

Lazebnik

Volumetric Stereo / Voxel Coloring

Discretized Scene Volume

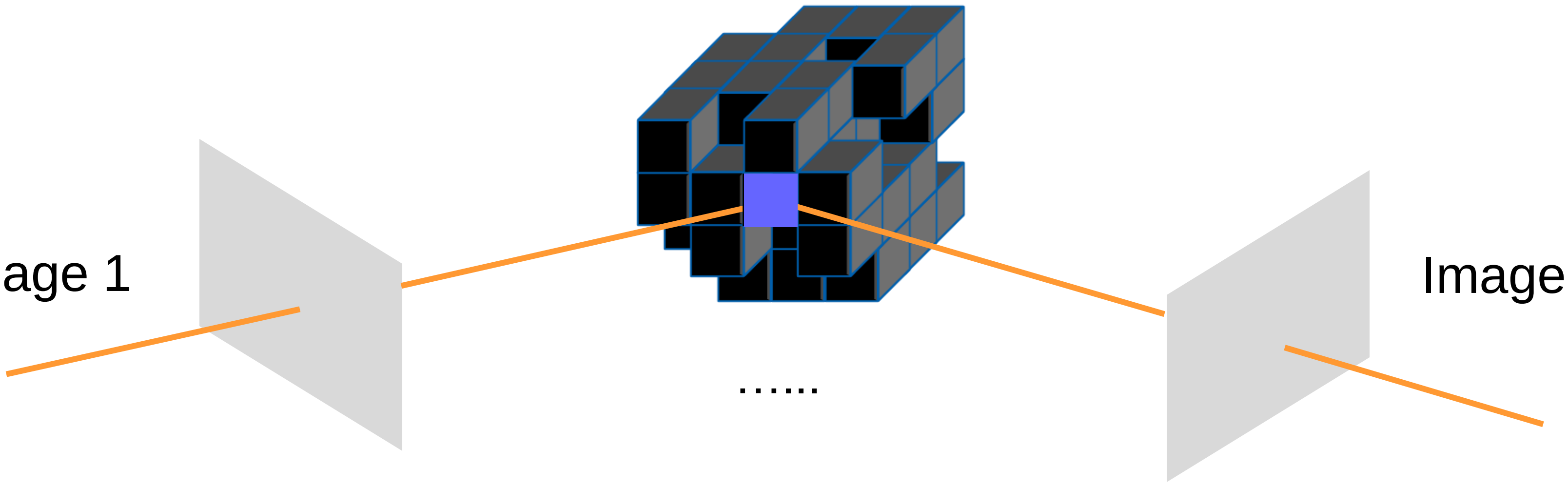
Input Images (Calibrated)



Lazebnik

Goal: Assign RGB values to voxels in V *photo-consistent* with images

Space Carving



- Space Carving Algorithm
  - Initialize to a volume V containing the true scene
  - Choose a voxel on the outside of the volume
  - Project to visible input images
  - Carve if not photo-consistent
  - Repeat until convergence

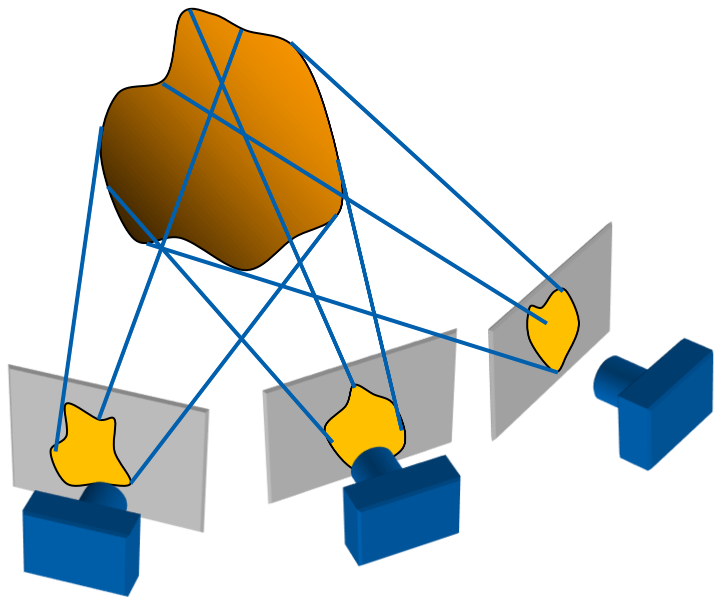
K. N. Kutulakos and S. M. Seitz, **A Theory of Shape by Space Carving**, *ICCV* 1999

Lazebnik

Reconstruction from Silhouettes

- The case of binary images: a voxel is photo-consistent if it lies inside the object's silhouette in all views

Binary Images

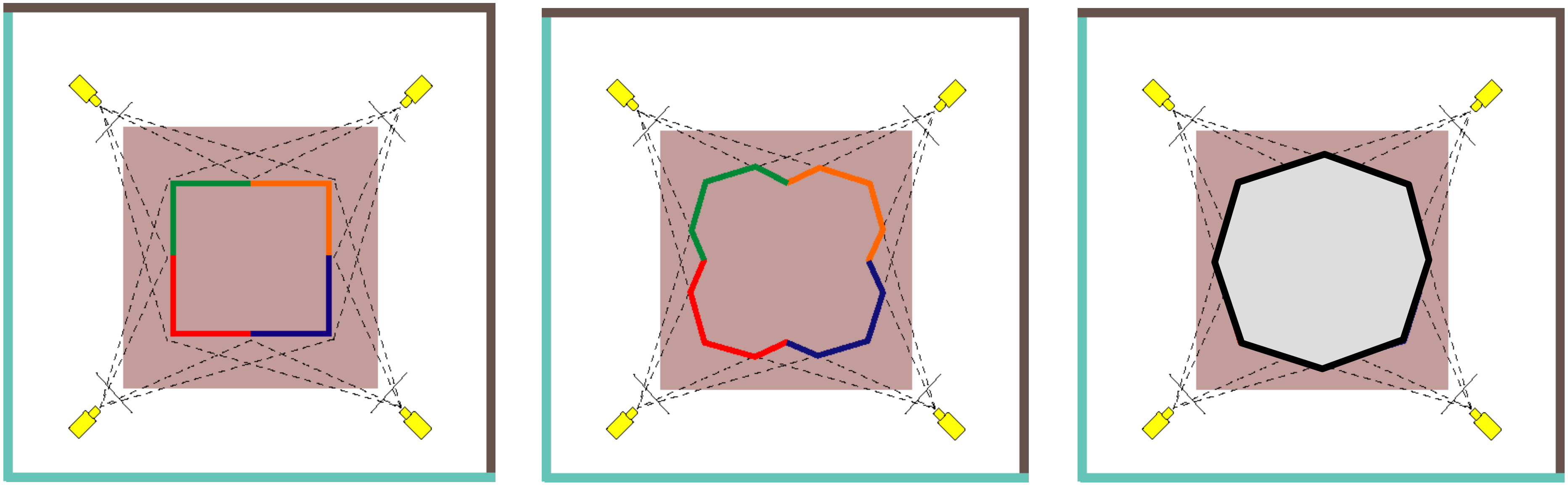


Finding the silhouette-consistent shape (*visual hull*):

- *Backproject* each silhouette
- Intersect backprojected volumes

Lazebnik

Photo-consistency vs. silhouette-consistency



True Scene Photo Hull Visual Hull

Lazebnik

Carved visual hulls

1. Compute visual hull
2. Use dynamic programming to find rims and constrain them to be fixed
3. Carve the visual hull to optimize photo-consistency

Yasutaka Furukawa and Jean Ponce, **Carved Visual Hulls for Image-Based Modeling**, ECCV 2006.

Lazebnik

Carved visual hulls: Pros and cons

- Pros
  - Visual hull gives a reasonable initial mesh that can be iteratively deformed
- Cons
  - Need silhouette extraction
  - Have to compute a lot of points that don't lie on the object
  - Finding rims is difficult
  - The carving step can get caught in local minima
- Possible solution: use sparse feature correspondences as initialization

Lazebnik

From feature matching to dense stereo

1. Extract features
2. Get a sparse set of initial matches
3. Iteratively expand matches to nearby locations
4. Use visibility constraints to filter out false matches
5. Perform surface reconstruction

Yasutaka Furukawa and Jean Ponce, **Accurate, Dense, and Robust Multi-View Stereopsis**, CVPR 2007.

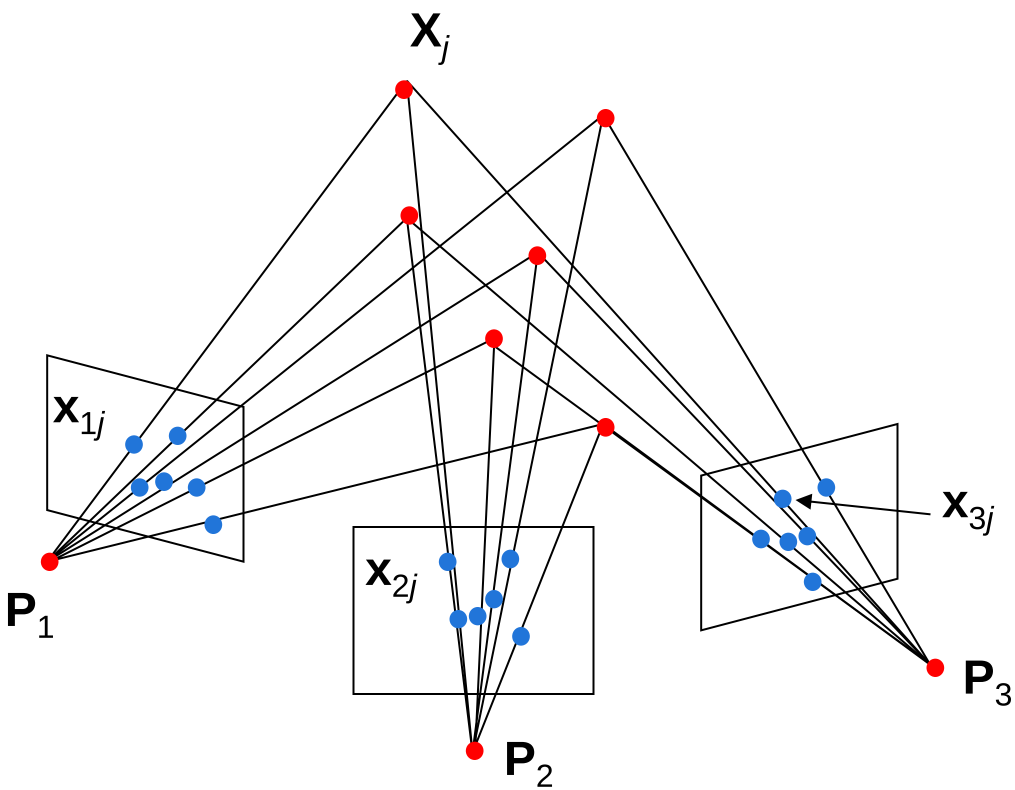
Lazebnik

Structure from motion

Given: m images of n fixed 3D points

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

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



Lazebnik

Structure from motion ambiguity

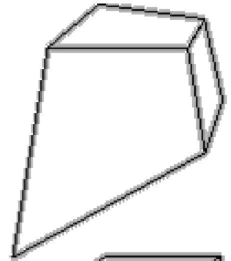
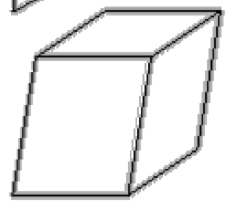
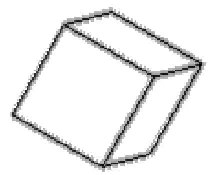
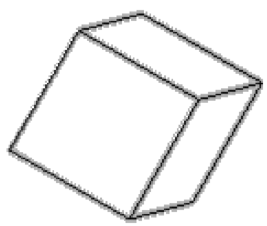
If we scale the entire scene by some factor *k* and, at the same time, scale the camera matrices by the factor of 1/*k*, the projections of the scene points in the image remain exactly the same

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

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

Lazebnik

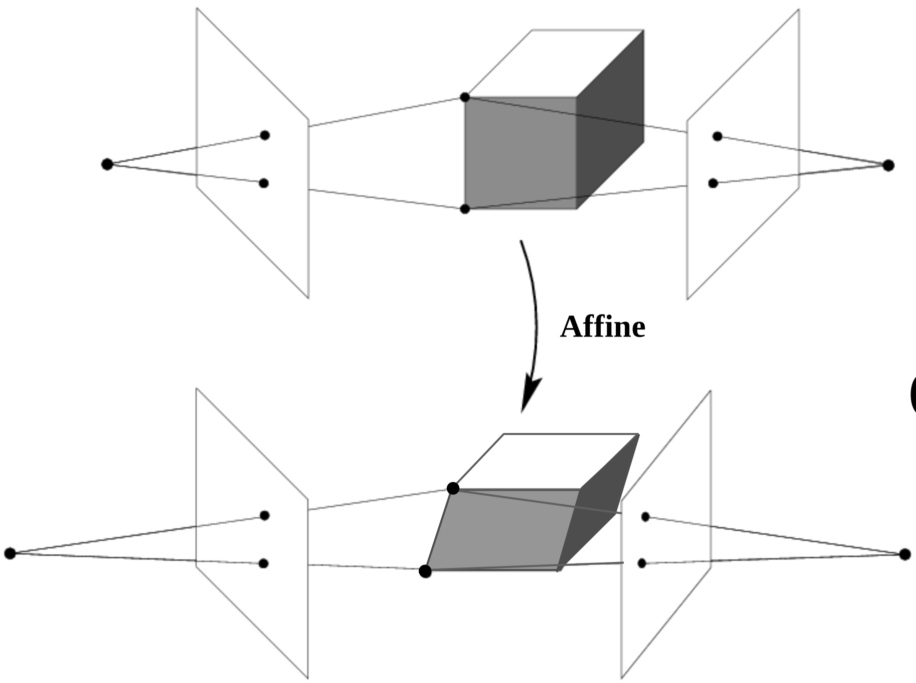
Types of ambiguity

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	$\begin{bmatrix} s\,R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios of length
Euclidean 6dof	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, lengths

- With no constraints on the camera calibration matrix or on the scene, we get a *projective* reconstruction
- Need additional information to *upgrade* the reconstruction to affine, similarity, or Euclidean

Lazebnik

Affine ambiguity

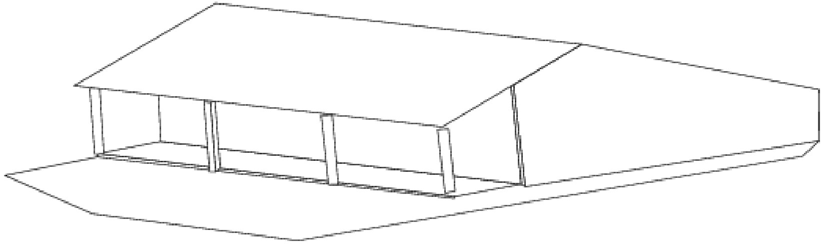


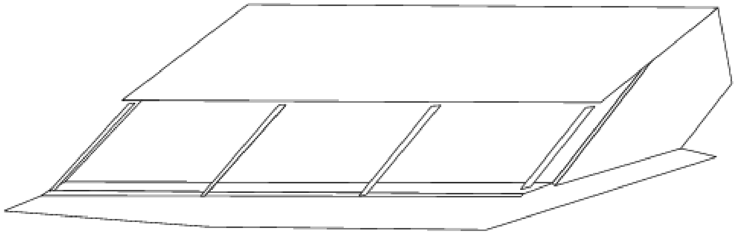
$$\mathbf{Q}_A = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathrm{T}} & 1 \end{bmatrix}$$

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

Lazebnik

Affine ambiguity





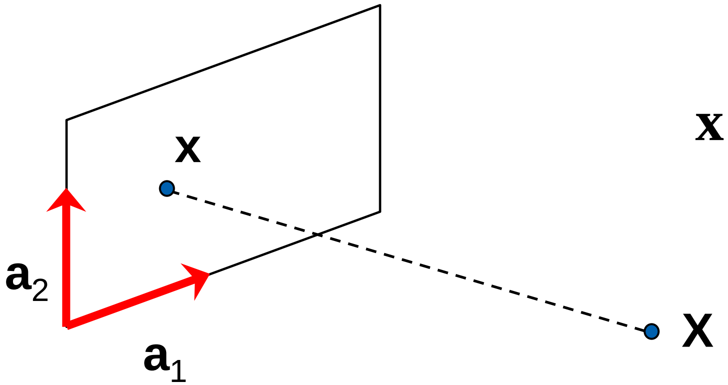
Lazebnik

Affine cameras

- A general affine camera combines the effects of an affine transformation of the 3D space, orthographic projection, and an affine transformation of the image:

$$\mathbf{P} = [3\times3\,\text{affine}]\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}[4\times4\,\text{affine}] = \begin{bmatrix}a_{11}&a_{12}&a_{13}&b_1\\a_{21}&a_{22}&a_{23}&b_2\\0&0&0&1\end{bmatrix} = \begin{bmatrix}\mathbf{A}&\mathbf{b}\\\mathbf{0}&\mathbf{1}\end{bmatrix}$$

- Affine projection is a linear mapping + 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 world origin

Lazebnik

Affine structure from motion

UNIVERSITÄT BONN

- Given: *m* images of *n* fixed 3D points:

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

- Problem: use the *mn* correspondences $\mathbf{x}_{ij}$ to estimate *m* projection matrices $\mathbf{A}_i$ and translation vectors $\mathbf{b}_i$, and *n* points $\mathbf{X}_j$
- The reconstruction is defined up to an arbitrary *affine* transformation **Q** (12 degrees of freedom):

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

- We have 2*mn* knowns and 8*m* + 3*n* unknowns (minus 12 dof for affine ambiguity)
- Thus, we must have 2*mn* >= 8*m* + 3*n* – 12
- For two views, we need four point correspondences

Lazebnik

Affine structure from motion

UNIVERSITÄT BONN

- Centering: subtract the centroid of the image points

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

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

- For simplicity, assume that the origin of the world coordinate system is at the centroid of the 3D points
- After centering, each normalized point $\mathbf{x}_{ij}$ is related to the 3D point $\mathbf{X}_i$ by

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

Lazebnik

Affine structure from motion

- Let's create a $2m \times n$ data (measurement) matrix:

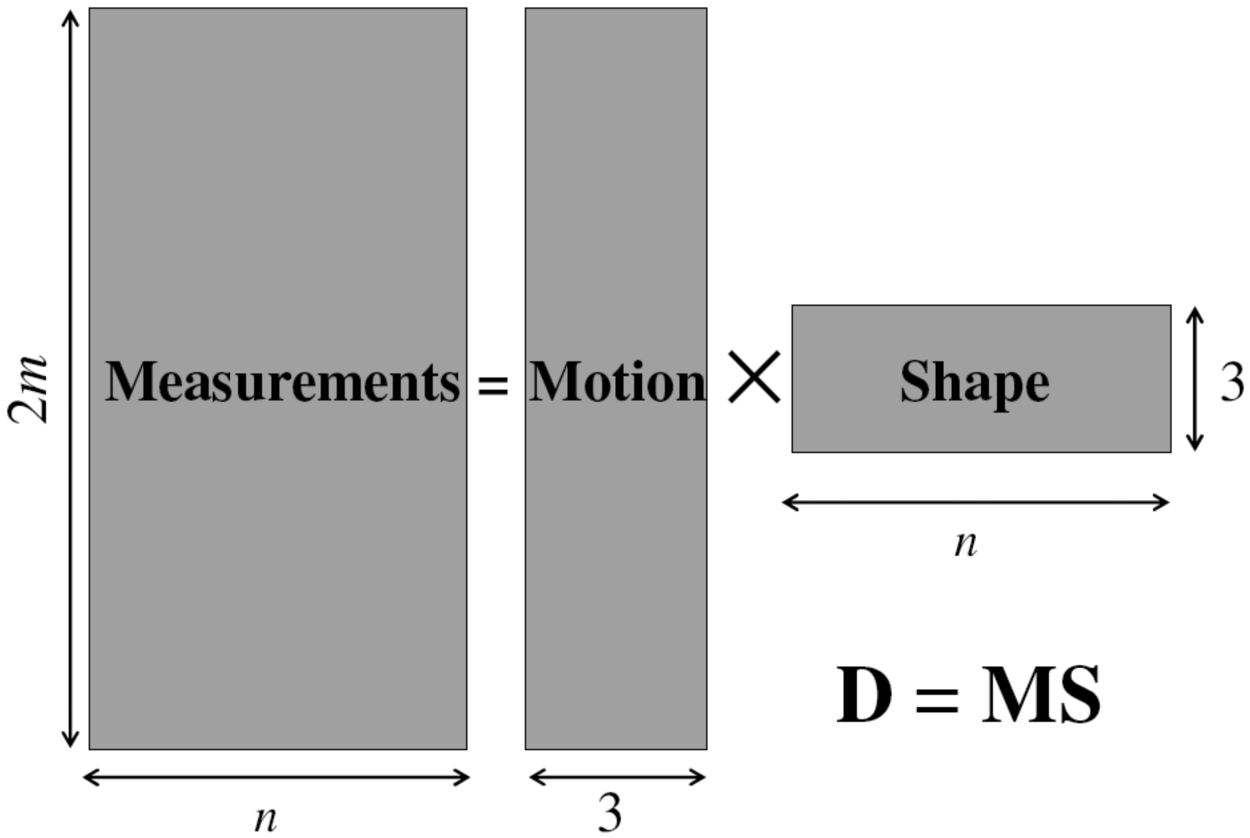
$$\mathbf{D} = \begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}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

The measurement matrix $\mathbf{D} = \mathbf{MS}$ must have 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.

Lazebnik

Factorizing the measurement matrix

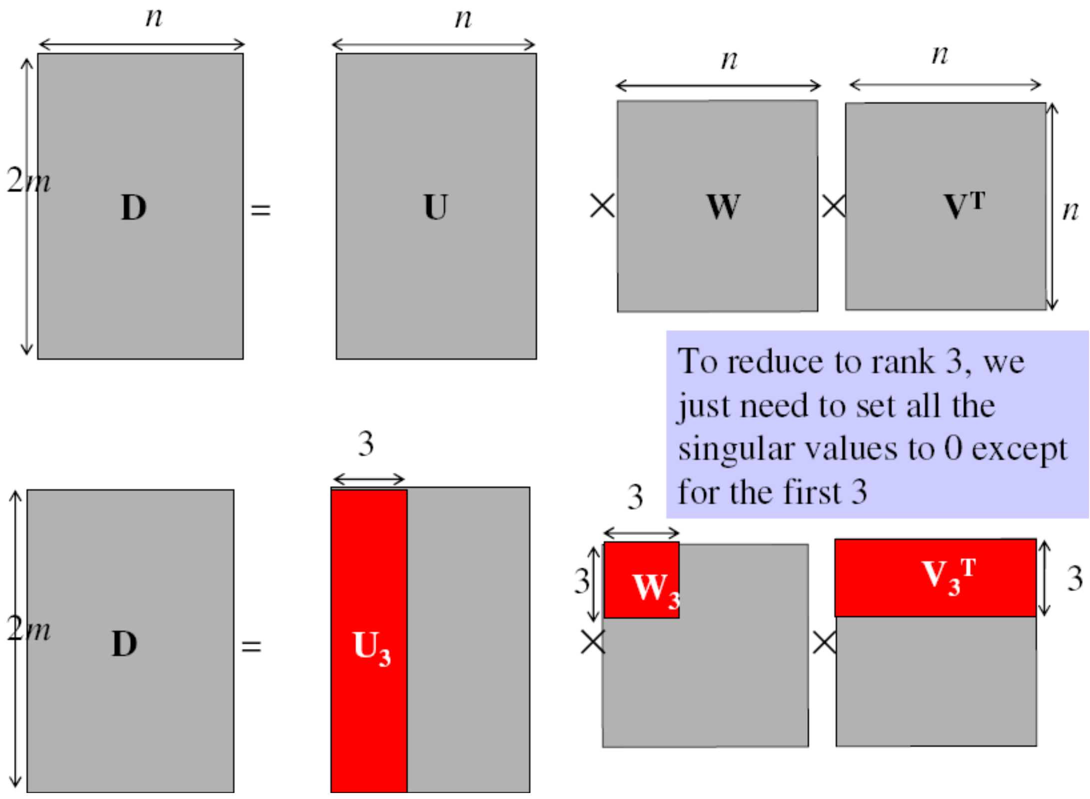


$$\mathbf{D} = \mathbf{M}\mathbf{S}$$

Source: M. Hebert

Factorizing the measurement matrix

- Singular value decomposition of D:

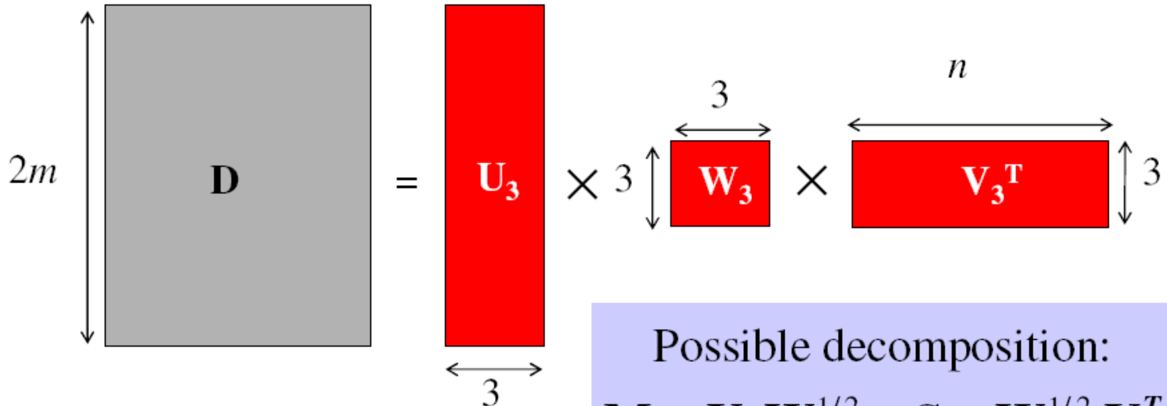


To reduce to rank 3, we just need to set all the singular values to 0 except for the first 3

Source: M. Hebert

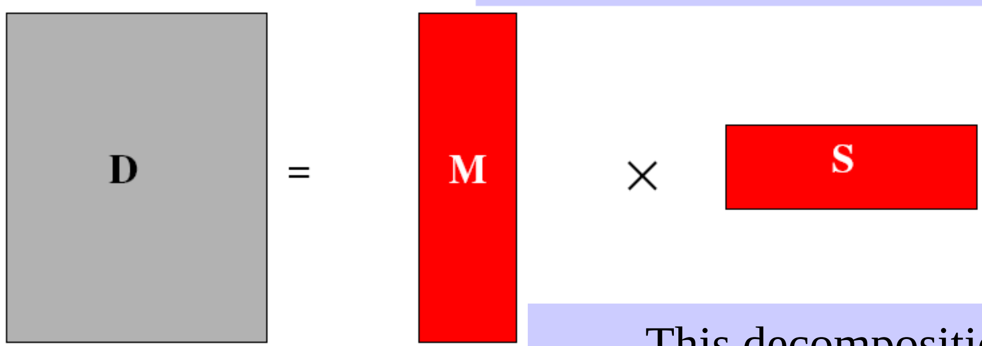
Factorizing the measurement matrix

- Obtaining a factorization from SVD:



Possible decomposition:

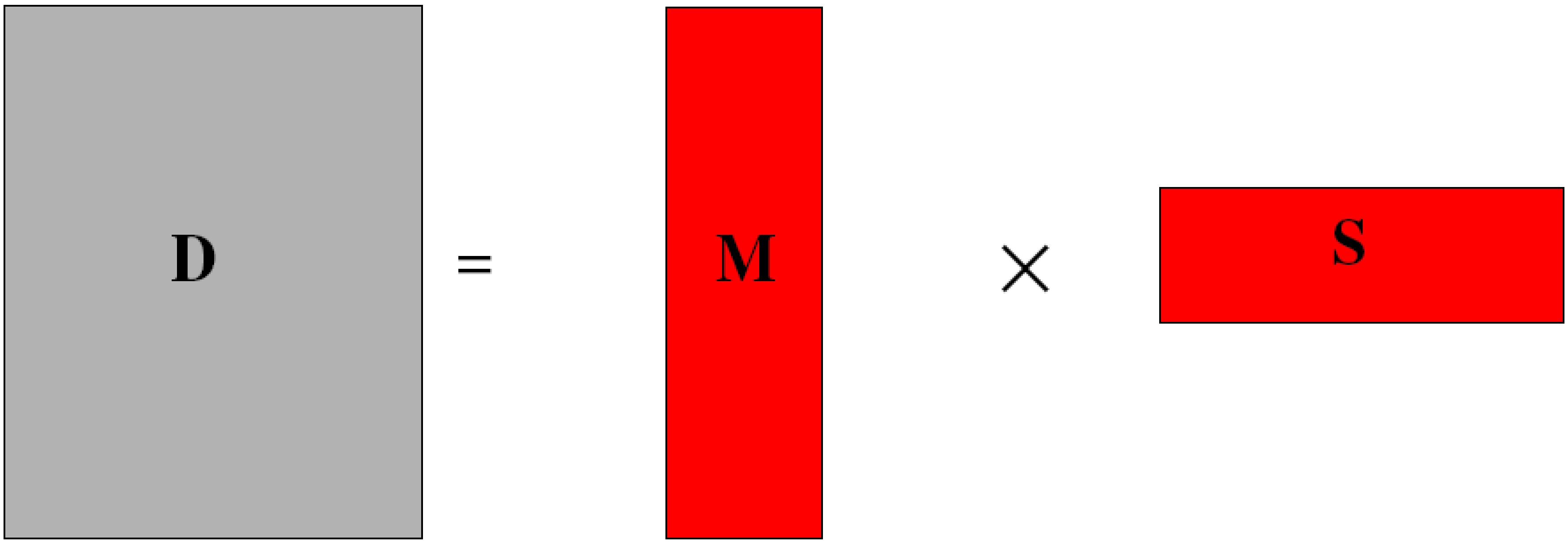
$$\mathbf{M} = \mathbf{U}_3 \mathbf{W}_3^{1/2} \qquad \mathbf{S} = \mathbf{W}_3^{1/2}\, \mathbf{V}_3^{T}$$



This decomposition minimizes
$|\mathbf{D}\text{-}\mathbf{MS}|^2$

Source: M. Hebert

Affine ambiguity

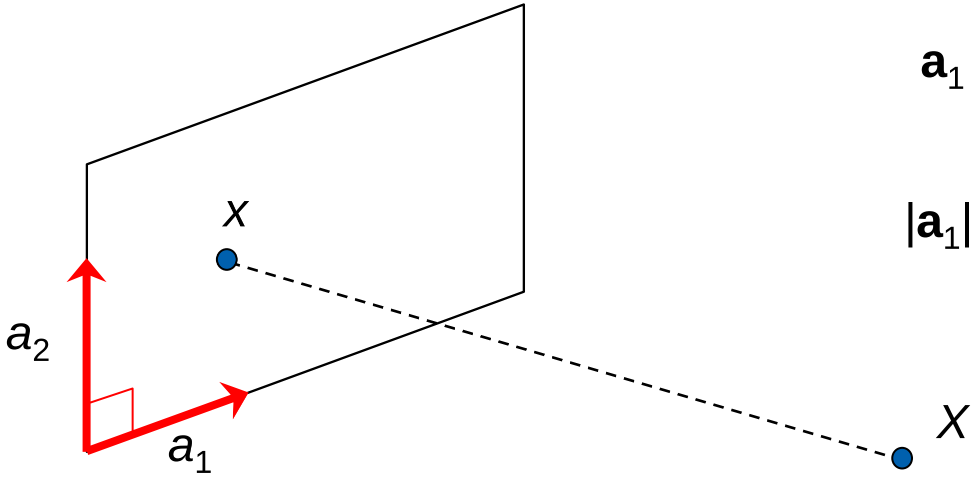


- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC**, **S** → $\mathbf{C}^{-1}\mathbf{S}$
- That is because we have only an affine transformation and we have not enforced any Euclidean constraints (like forcing the image axes to be perpendicular, for example)

Source: M. Hebert

Eliminating the affine ambiguity

- Orthographic: image axes are perpendicular and of unit length



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

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

Source: M. Hebert

Solve for orthographic constraints

UNIVERSITÄT BONN

Three equations for each image i

$$\begin{aligned} \mathbf{m}_{i1}^T \mathbf{C}\mathbf{C}^T \mathbf{m}_{i1} &= 1 \\ \mathbf{m}_{i2}^T \mathbf{C}\mathbf{C}^T \mathbf{m}_{i2} &= 1 \\ \mathbf{m}_{i1}^T \mathbf{C}\mathbf{C}^T \mathbf{m}_{i2} &= 0 \end{aligned} \quad \text{where} \quad \mathbf{M}_i = \begin{bmatrix} \mathbf{m}_{i1}^T \\ \mathbf{m}_{i2}^T \end{bmatrix}$$

- Two options:
  - Solve for **C** (Newton's method, quadratic)
  - Solve linearly $\mathbf{L} = \mathbf{C}\mathbf{C}^T$
  - Recover **C** from **L** by SVD or Cholesky decomposition: $\mathbf{L} = \mathbf{C}\mathbf{C}^T$
- Update **M** and **S**: $\mathbf{M}' = \mathbf{M}\mathbf{C}$, $\mathbf{S}' = \mathbf{C}^{-1}\mathbf{S}$

Algorithm summary

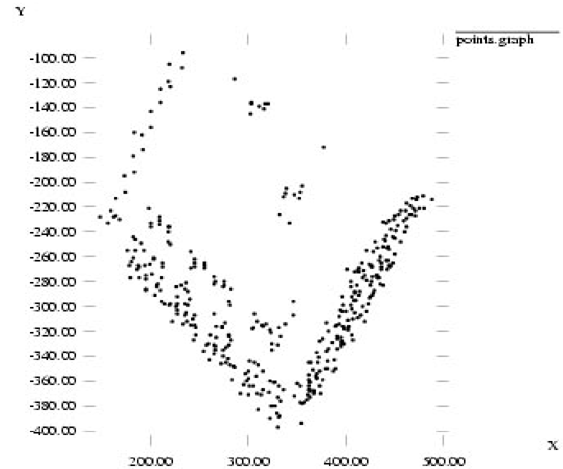
- Given: m images and n features $\mathbf{x}_{ij}$
- For each image i, center the feature coordinates
- Construct a $2m \times n$ measurement matrix $\mathbf{D}$:
  - Column j contains the projection of point j in all views
  - Row i contains one coordinate of the projections of all the n points in image i
- Factorize $\mathbf{D}$:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\ \mathbf{W}\ \mathbf{V}^{\mathrm{T}}$
  - 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:
  - $\mathbf{M} = \mathbf{U}_3\mathbf{W}_3^{1/2}$ and $\mathbf{S} = \mathbf{W}_3^{1/2}\ \mathbf{V}_3^{\mathrm{T}}$ (**or** $\mathbf{M} = \mathbf{U}_3$ and $\mathbf{S} = \mathbf{W}_3\mathbf{V}_3^{\mathrm{T}}$)
- Eliminate affine ambiguity

Source: M. Hebert

Reconstruction results

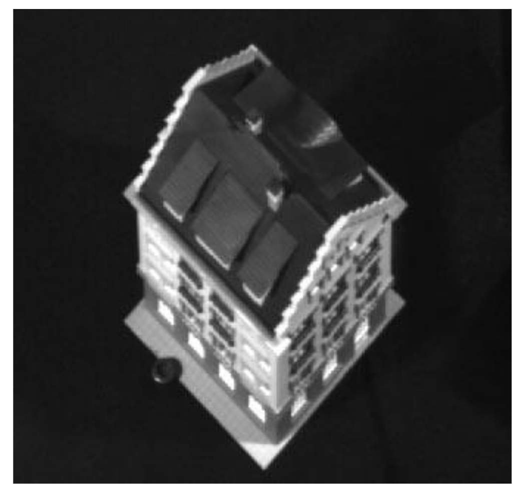
1

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.

Lazebnik

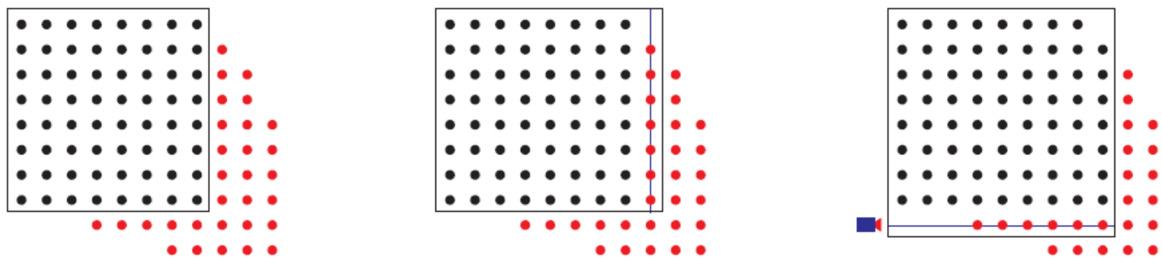
Dealing with missing data

- So far, we have assumed that all points are visible in all views
- In reality, the measurement matrix typically looks something like this:

Lazebnik

Dealing with missing data

- Possible solution: decompose matrix into dense sub-blocks, factorize each sub-block, and fuse the results
  – Finding dense maximal sub-blocks of the matrix is NP-complete (equivalent to finding maximal cliques in a graph)
- Incremental bilinear refinement



(1) Perform factorization on a dense sub-block

(2) Solve for a new 3D point visible by at least two known cameras (linear least squares)

(3) Solve for a new camera that sees at least three known 3D points (linear least squares)

F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects. PAMI 2007.

Lazebnik

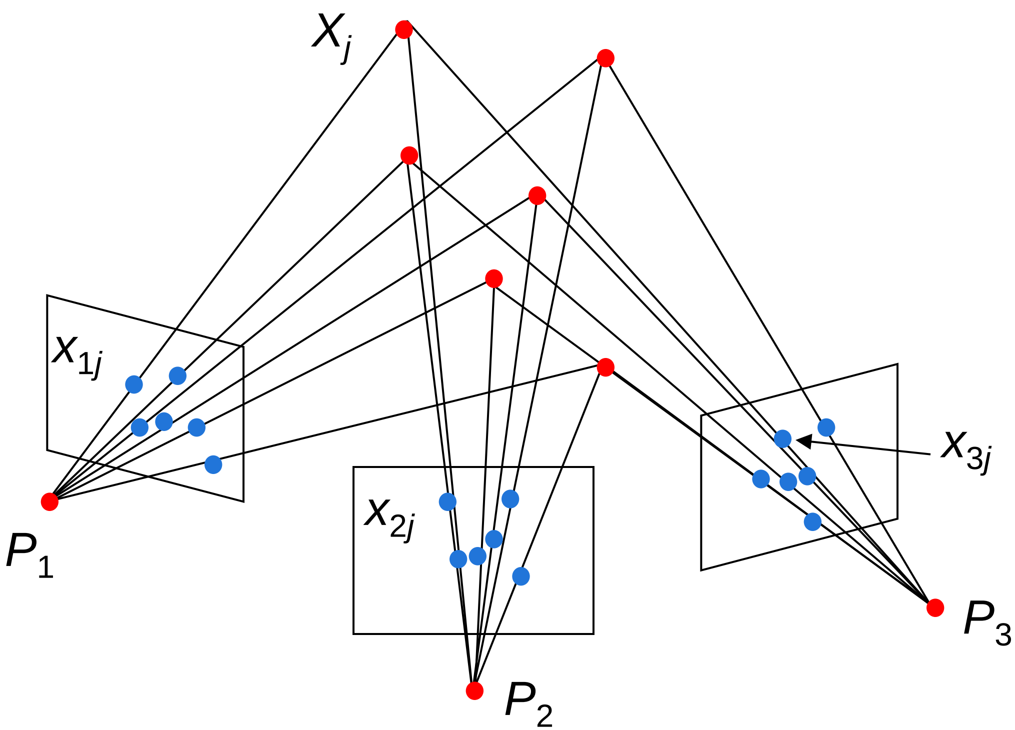
Projective structure from motion

UNIVERSITÄT BONN

- Given: *m* images of *n* fixed 3D points

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

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



Lazebnik

Projective structure from motion

UNIVERSITÄT BONN

- Given: *m* images of *n* fixed 3D points

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

- Problem: estimate *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

$$2mn >= 11m + 3n - 15$$

- For two cameras, at least 7 points are needed

Lazebnik

Projective SFM: Two-camera case

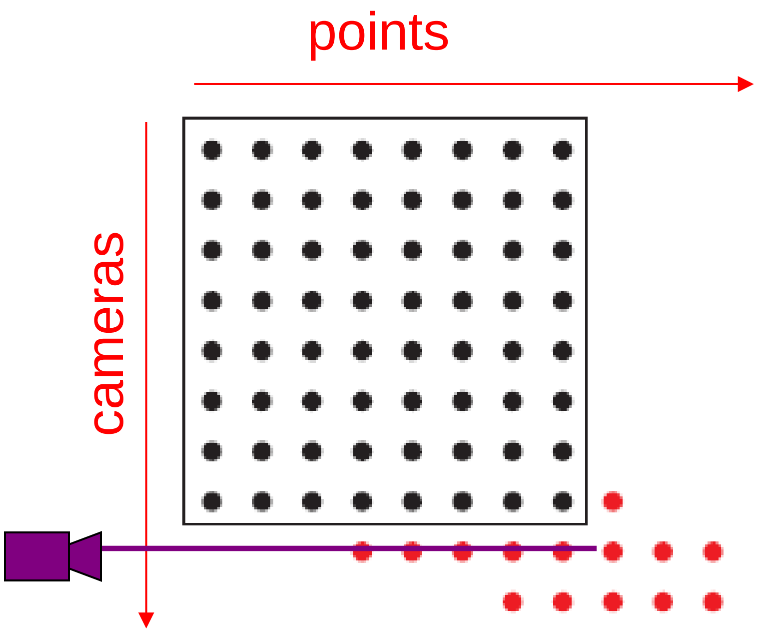
- Compute fundamental matrix $\mathbf{F}$ between the two views
- First camera matrix: $[\mathbf{I}|\mathbf{0}]$
- Second camera matrix: $[\mathbf{A}|\mathbf{b}]$
- Then $\mathbf{b}$ is the epipole ($\mathbf{F}^T\mathbf{b} = 0$), $\mathbf{A} = -[\mathbf{b}_\times]\mathbf{F}$

Lazebnik

Sequential structure from motion

UNIVERSITÄT BONN

- 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*

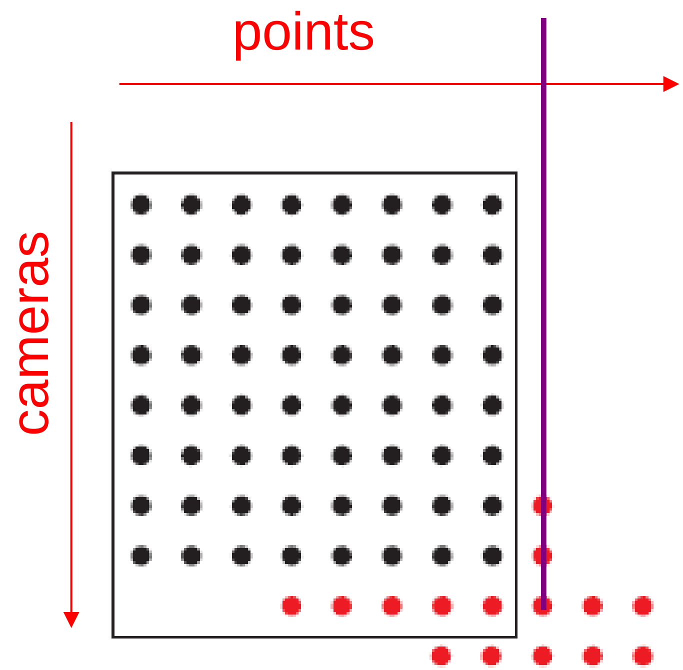


Lazebnik

Sequential structure from motion

UNIVERSITÄT BONN

- 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*

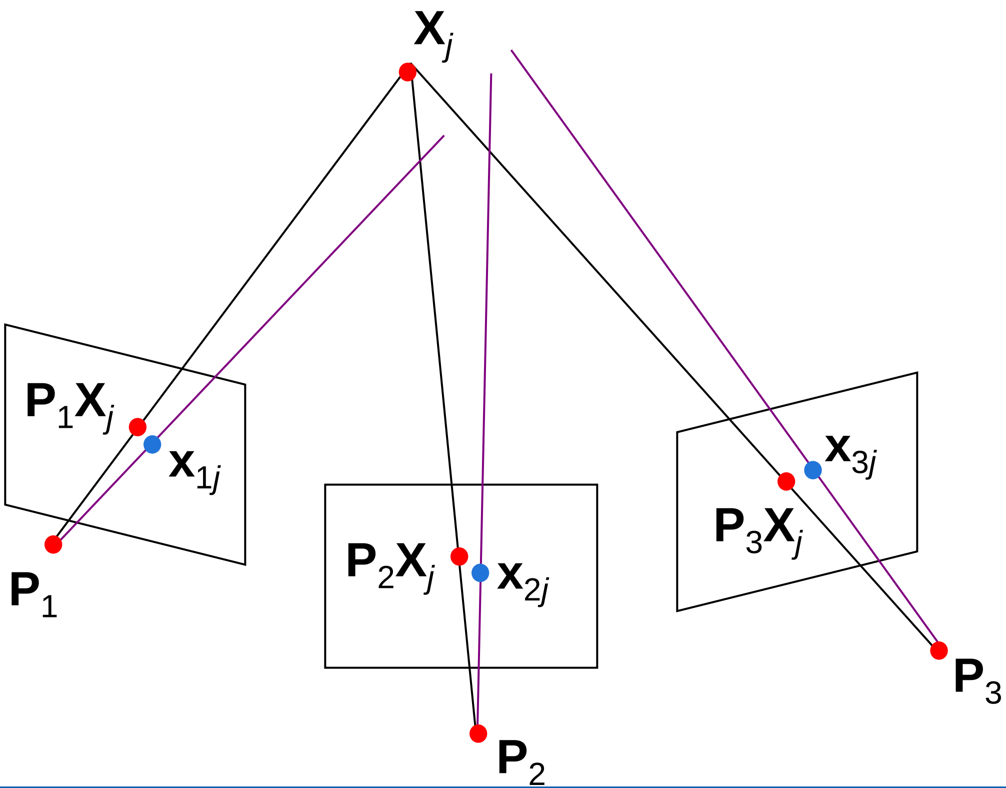


Lazebnik

Bundle adjustment

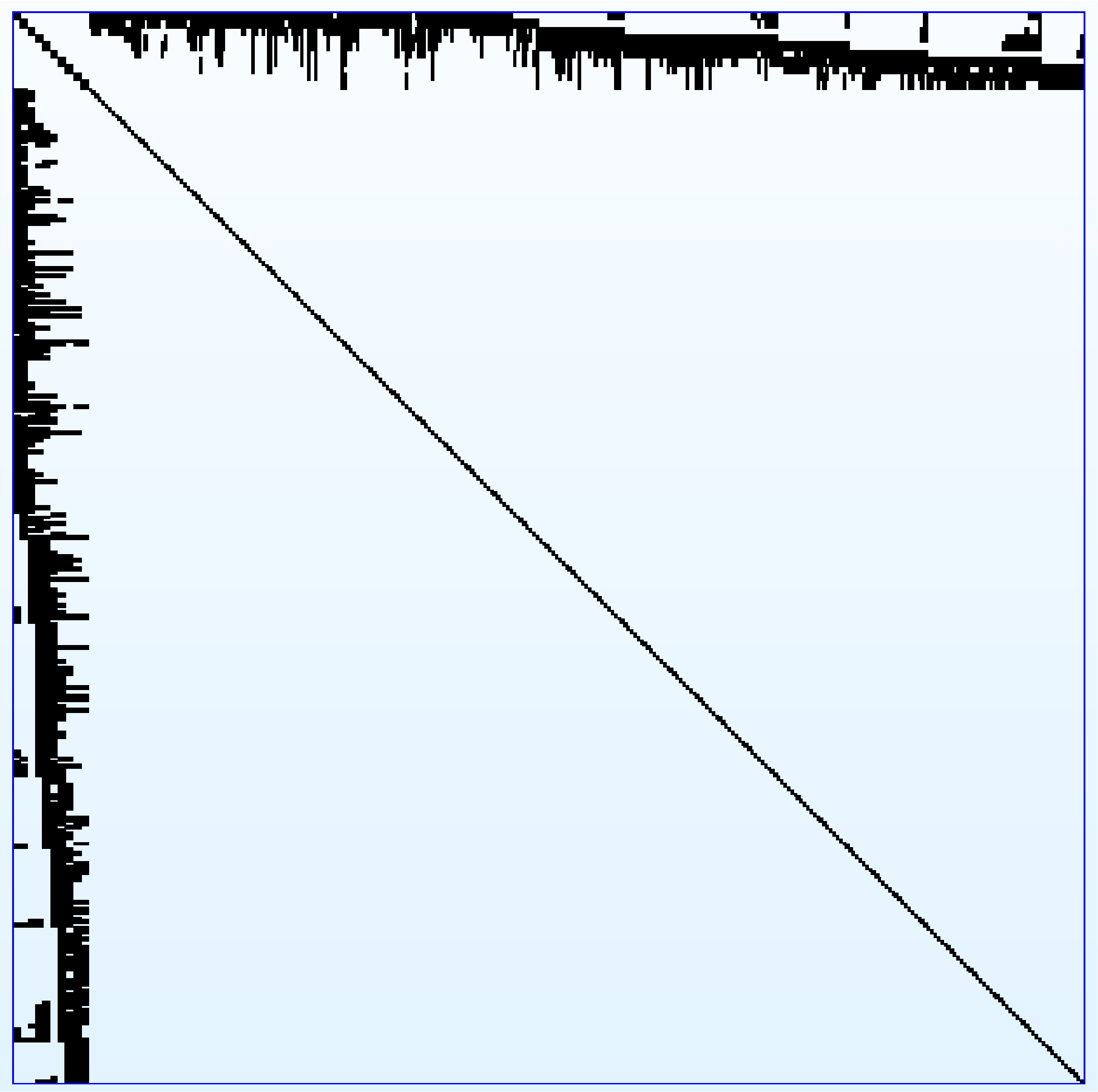
- Non-linear method for refining structure and motion
- Minimizing reprojection error

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



Lazebnik

Hessian (real problem)



Black: non-zero

Lourakis

Self-calibration

- Self-calibration (auto-calibration) is the process of determining intrinsic camera parameters directly from uncalibrated images
- For example, when the images are acquired by a single moving camera, we can use the constraint that the intrinsic parameter matrix remains fixed for all the images
  - Compute initial projective reconstruction and find 3D projective transformation matrix **Q** such that all camera matrices are in the form $\mathbf{P}_i = \mathbf{K}\,[\mathbf{R}_i \mid \mathbf{t}_i]$
- Can use constraints on the form of the calibration matrix: zero skew

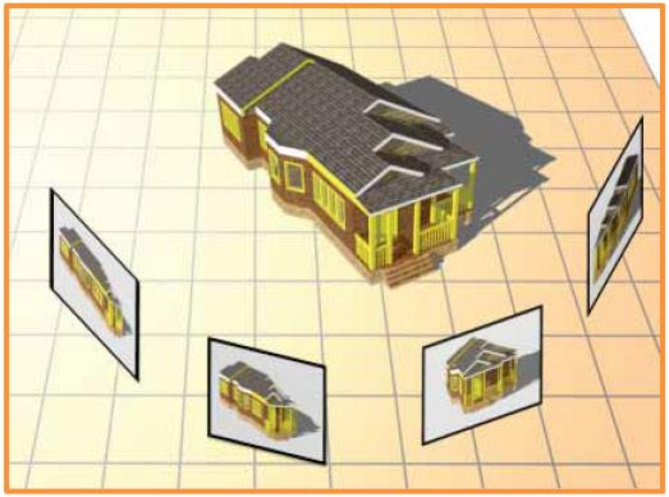
Lazebnik

Factorization Method for Rigid SFM

Kontsevich *et al.* 1987, Tomasi and Kanade 1992

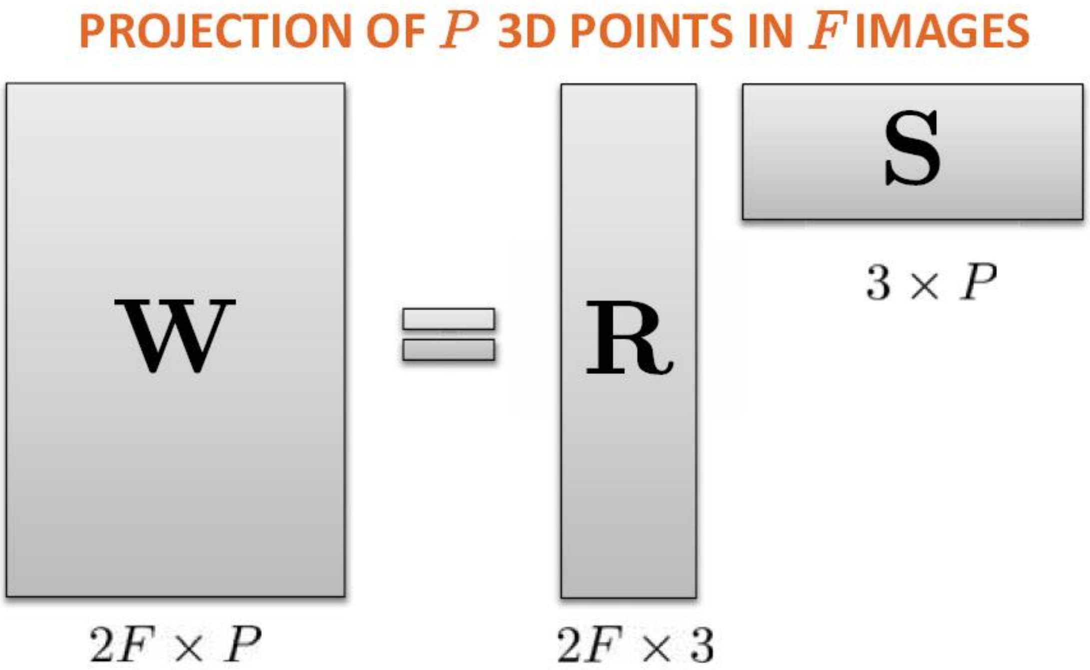
ASSUMPTIONS

- Orthographic Camera
- At least 3 images
- Rigid Scene
- Camera Motion
- Corresponding points available



FACTORIZATION METHOD FOR RIGID SFM

Kontsevich *et al.* 1987, Tomasi and Kanade 1992



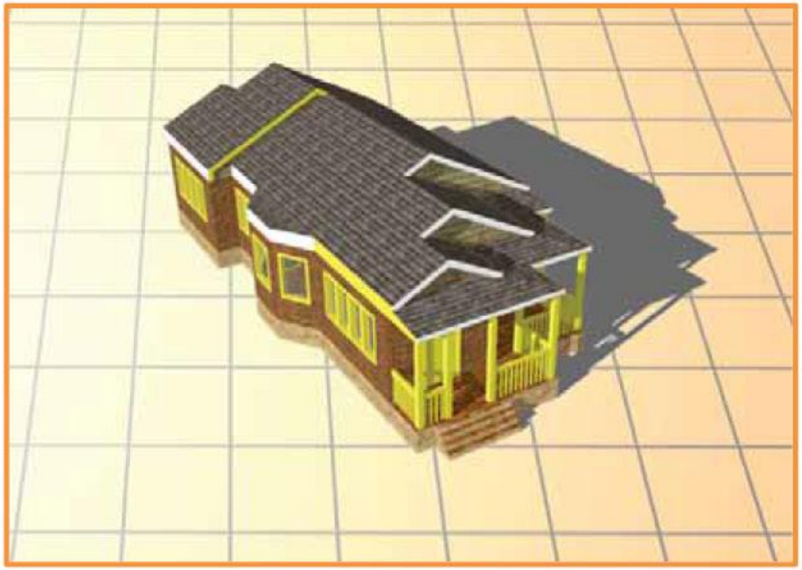
$$\mathbf{W}_{\text{measurement}} = \mathbf{R}_{\text{motion}} \times \mathbf{S}_{\text{shape}}$$

NonRigid Structure

3D Structure That Deforms Over Time

RIGID STRUCTURE

$$\mathbf{S}_{3\times P} = \begin{bmatrix} X_1 & X_2 & \dots & X_P \\ Y_1 & Y_2 & \dots & Y_P \\ Z_1 & Z_2 & \dots & Z_P \end{bmatrix}$$

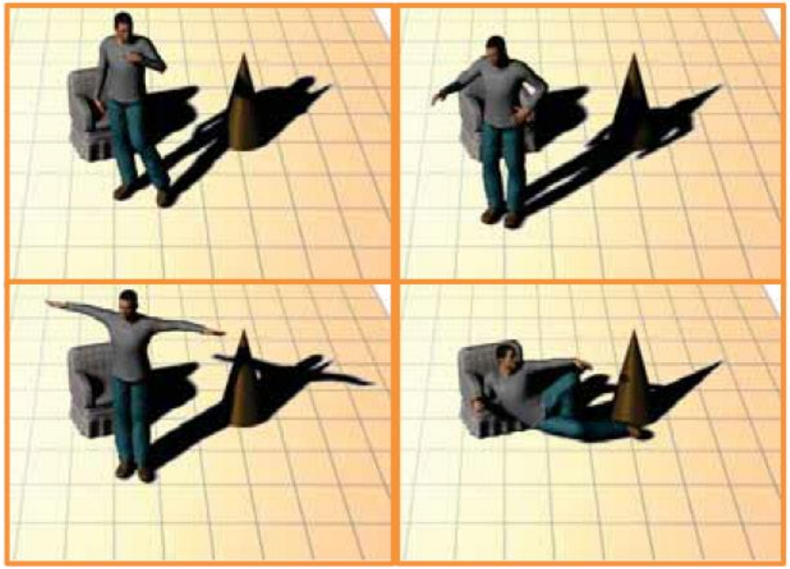


NonRigid Structure

3D Structure That Deforms Over Time

RIGID STRUCTURE

$$\mathbf{S}_{3\times P} = \begin{bmatrix} X_1 & X_2 & \dots & X_P \\ Y_1 & Y_2 & \dots & Y_P \\ Z_1 & Z_2 & \dots & Z_P \end{bmatrix}$$



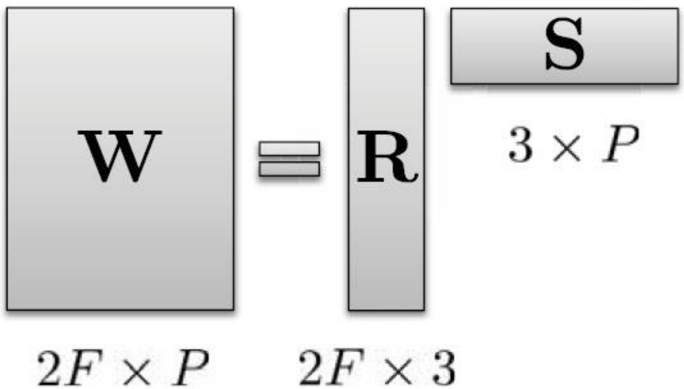
NONRIGID STRUCTURE

$$\mathbf{S}_{3F\times P} = \begin{bmatrix} \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1P} \\ Y_{11} & Y_{12} & \dots & Y_{1P} \\ Z_{11} & Z_{12} & \dots & Z_{1P} \end{bmatrix}_{3\times P} \\ \\ \begin{bmatrix} X_{21} & X_{22} & \dots & X_{2P} \\ Y_{21} & Y_{22} & \dots & Y_{2P} \\ Z_{21} & Z_{22} & \dots & Z_{2P} \end{bmatrix}_{3\times P} \\ \vdots \\ \begin{bmatrix} X_{F1} & X_{F2} & \dots & X_{FP} \\ Y_{F1} & Y_{F2} & \dots & Y_{FP} \\ Z_{F1} & Z_{F2} & \dots & Z_{FP} \end{bmatrix}_{3\times P} \end{bmatrix}$$

NonRigid Structure From Motion

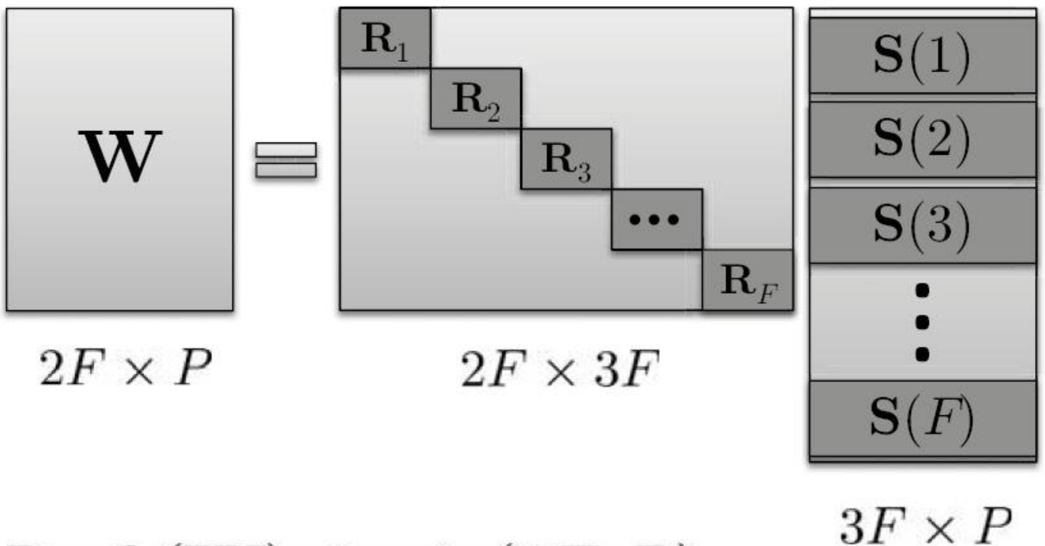
Comparison with Rigid Structure from Motion

RIGID SFM



$$\mathrm{Rank}(\mathbf{W}) \leq 3$$

NONRIGID SFM



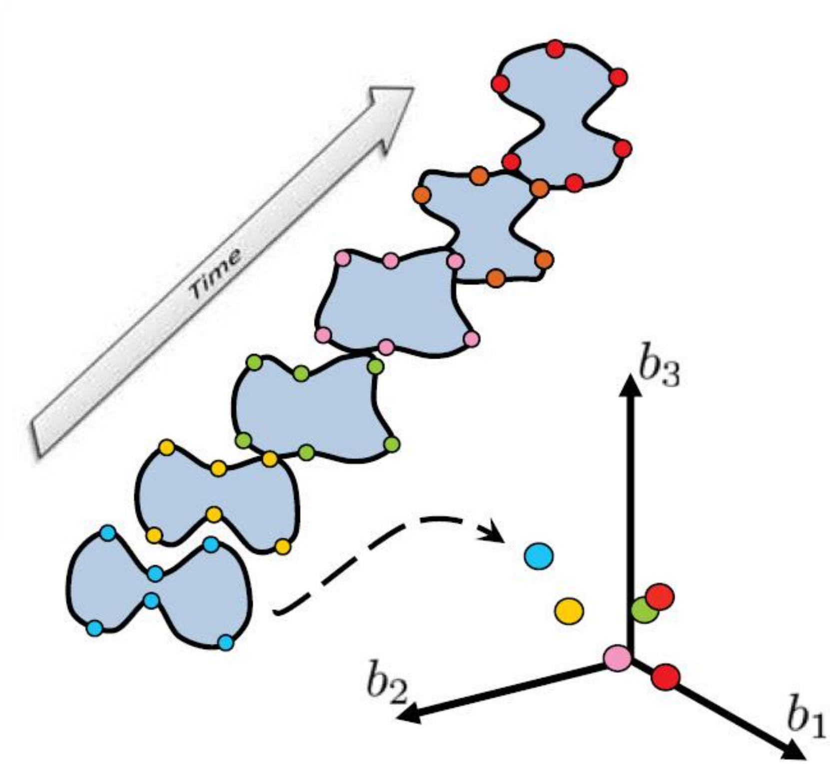
$$\mathrm{Rank}(\mathbf{W}) \leq \min(2F, P)$$

Nonrigid Structure From Motion

Two Major Approaches

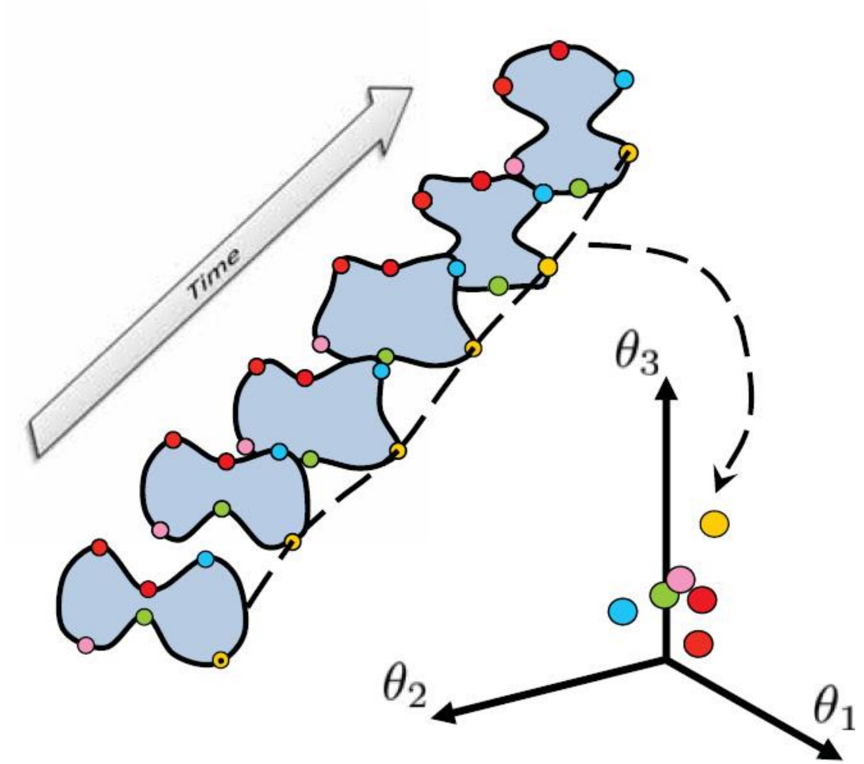
Shape Basis

3D points at each time instant lie in a low dimensional subspace



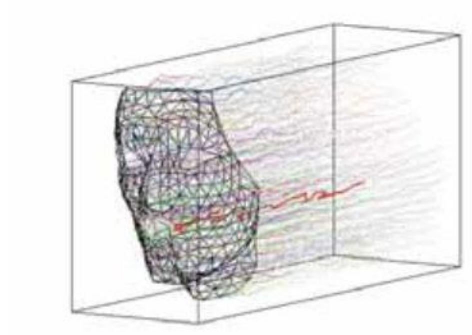
Trajectory Basis

Trajectory of each point over time lies in a low dimensional subspace

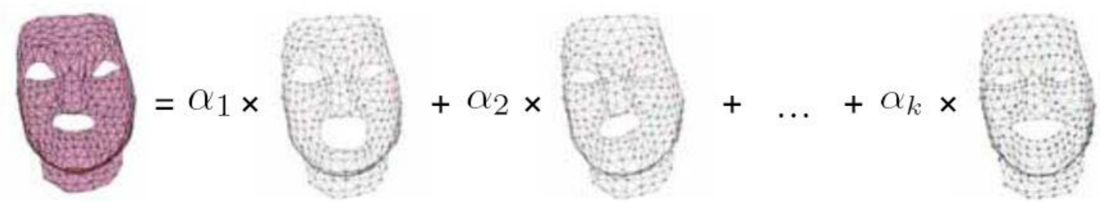


LINEAR SHAPE MODEL

[T. Cootes et al. 91, Bregler et al. 97]



$$\begin{bmatrix} \mathbf{X}_{11} & \cdots & \mathbf{X}_{1P} \\ \mathbf{X}_{21} & & \mathbf{X}_{2P} \\ \vdots & & \vdots \\ \mathbf{X}_{F1} & \cdots & \mathbf{X}_{FP} \end{bmatrix}$$



LINEAR SHAPE MODEL

$$\begin{bmatrix} \mathbf{X}_{11} & \cdots & \mathbf{X}_{1P} \\ \mathbf{X}_{21} & & \mathbf{X}_{2P} \\ \vdots & & \vdots \\ \mathbf{X}_{F1} & \cdots & \mathbf{X}_{FP} \end{bmatrix} = \begin{bmatrix} \omega_{11} & \cdots & \omega_{1k} \\ \omega_{21} & & \omega_{2k} \\ \vdots & & \vdots \\ \omega_{F1} & \cdots & \omega_{Fk} \end{bmatrix} \begin{bmatrix} -\mathbf{b_1}- \\ -\mathbf{b_2}- \\ \vdots \\ -\mathbf{b_k}- \end{bmatrix}$$

BREGLER *et al.* 2000

Nested SVD

$$
\begin{bmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1P} \\ \mathbf{x}_{21} & & \mathbf{x}_{2P} \\ \vdots & & \vdots \\ \mathbf{x}_{F1} & \cdots & \mathbf{x}_{FP} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 & & & \\ & \mathbf{R}_2 & & \\ & & \ddots & \\ & & & \mathbf{R}_F \end{bmatrix} \begin{bmatrix} \omega_{11} & \cdots & \omega_{1k} \\ \omega_{21} & & \omega_{2k} \\ \vdots & & \vdots \\ \omega_{F1} & \cdots & \omega_{Fk} \end{bmatrix} \begin{bmatrix} -\mathbf{b_1}- \\ -\mathbf{b_2}- \\ \vdots \\ -\mathbf{b_k}- \end{bmatrix}
$$

$$
= \underbrace{\begin{bmatrix} \omega_{11}\mathbf{R}_1 & \cdots & \omega_{1k}\mathbf{R}_1 \\ \omega_{21}\mathbf{R}_2 & & \omega_{2k}\mathbf{R}_2 \\ \vdots & & \vdots \\ \omega_{F1}\mathbf{R}_F & \cdots & \omega_{Fk}\mathbf{R}_F \end{bmatrix}}_{2F \times 3k} \underbrace{\begin{bmatrix} -\mathbf{b}_1- \\ -\mathbf{b}_2- \\ \vdots \\ -\mathbf{b}_k- \end{bmatrix}}_{3k \times P}
$$

BREGLER *et al.* 2000

Outer SVD

$$\underbrace{\begin{bmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1P} \\ \mathbf{x}_{21} & & \mathbf{x}_{2P} \\ \vdots & & \vdots \\ \mathbf{x}_{F1} & \cdots & \mathbf{x}_{FP} \end{bmatrix}}_{\mathbf{W}} = \underbrace{\begin{bmatrix} \omega_{11}\mathbf{R}_1 & \cdots & \omega_{1k}\mathbf{R}_1 \\ \omega_{21}\mathbf{R}_2 & & \omega_{2k}\mathbf{R}_2 \\ \vdots & & \vdots \\ \omega_{F1}\mathbf{R}_F & \cdots & \omega_{Fk}\mathbf{R}_F \end{bmatrix}}_{\mathbf{H}:\ 2F \times 3k} \underbrace{\begin{bmatrix} -\mathbf{b}_1- \\ -\mathbf{b}_2- \\ \vdots \\ -\mathbf{b}_k- \end{bmatrix}}_{\mathbf{B}:\ 3k \times P}$$

SVD

$$\mathbf{W} = \mathbf{U}\mathbf{D}\mathbf{V}^T$$

$$\mathbf{W} = (\mathbf{U}\mathbf{D}^{\frac{1}{2}})(\mathbf{D}^{\frac{1}{2}}\mathbf{V}^T)$$

$$\mathbf{W} = \hat{\mathbf{H}}\hat{\mathbf{B}}$$

BREGLER *et al.* 2000

Inner SVD

$$\mathbf{W} = \hat{\mathbf{H}}\hat{\mathbf{B}}$$

$$\mathbf{H} = \begin{bmatrix} \omega_{11}\mathbf{R}_1 & \cdots & \omega_{1k}\mathbf{R}_1 \\ \omega_{21}\mathbf{R}_2 & & \omega_{2k}\mathbf{R}_2 \\ \vdots & & \vdots \\ \omega_{F1}\mathbf{R}_F & \cdots & \omega_{Fk}\mathbf{R}_1 \end{bmatrix}$$

$$\mathbf{h}_1 = \begin{bmatrix} \omega_{11}r_1^1 & \omega_{11}r_1^2 & \omega_{11}r_1^3 & \cdots & \omega_{1k}r_1^1 & \omega_{1k}r_1^2 & \omega_{1k}r_1^3 \\ \omega_{11}r_1^4 & \omega_{11}r_1^5 & \omega_{11}r_1^6 & \cdots & \omega_{1k}r_1^4 & \omega_{1k}r_1^5 & \omega_{1k}r_1^6 \end{bmatrix}$$

$$\mathbf{h}_1' = \underbrace{\begin{bmatrix} \omega_{11}r_1^1 & \omega_{11}r_1^2 & \omega_{11}r_1^3 & \omega_{11}r_1^4 & \omega_{11}r_1^5 & \omega_{11}r_1^6 \\ \omega_{12}r_1^1 & \omega_{12}r_1^2 & \omega_{12}r_1^3 & \omega_{12}r_1^4 & \omega_{12}r_1^5 & \omega_{12}r_1^6 \\ \vdots & & & & & \vdots \\ \omega_{1k}r_1^1 & \omega_{1k}r_1^2 & \omega_{1k}r_1^3 & \omega_{1k}r_1^4 & \omega_{1k}r_1^5 & \omega_{1k}r_1^6 \end{bmatrix}}_{\text{rank 1}} = \begin{bmatrix} \omega_{11} \\ \omega_{12} \\ \vdots \\ \omega_{1k} \end{bmatrix} \begin{bmatrix} r_1^1 & r_1^2 & r_1^3 & r_1^4 & r_1^5 & r_1^6 \end{bmatrix}$$

SVD $\mathbf{h}_1' = \mathbf{u}\mathbf{d}\mathbf{v}^T = \hat{\omega}\hat{\mathbf{r}}$

METRIC RECTIFICATION USING ORTHONORMALITY CONSTRAINTS

BREGLER *et al.* 2000
OVERVIEW

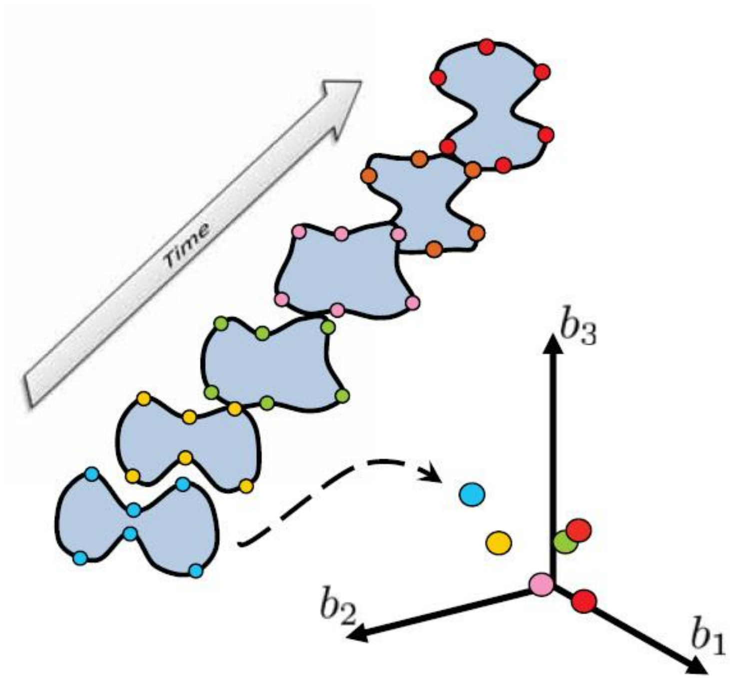
- OUTER SVD: PERFORM SVD ON **W** TO GET ESTIMATES OF:
  - **H**: CAMERA PROJECTIONS AND COEFFICIENTS
    - INNER SVD: PERFORM SVD ON **H** TO GET ESTIMATES OF:
      - OMEGA: COEFFICIENTS
      - **R**: CAMERA PROJECTIONS
    - METRIC RECTIFY USING ORTHONORMALITY CONSTRAINTS
  - **B**: THE SHAPE BASIS

Nonrigid Structure From Motion

Two Major Approaches

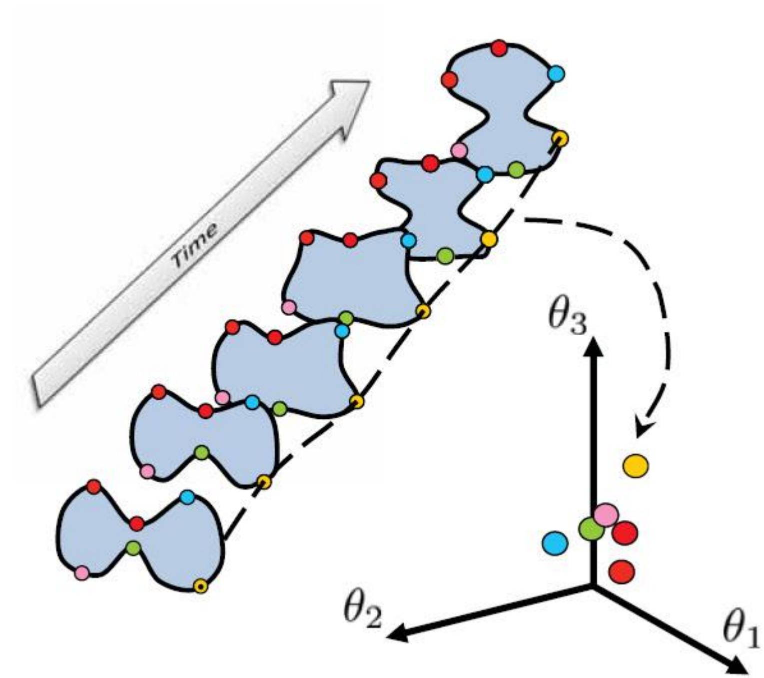
Shape Basis

3D points at each time instant lie in a low dimensional subspace



Trajectory Basis

Trajectory of each point over time lies in a low dimensional subspace



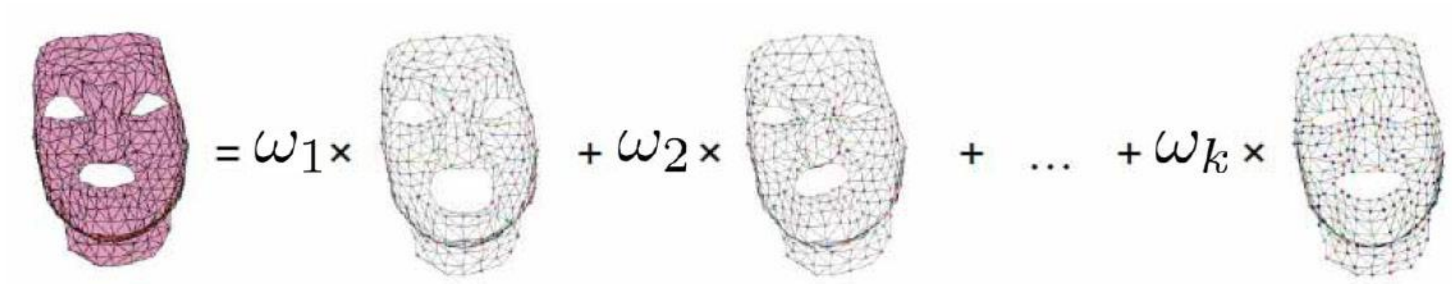
Dynamic Structure

Shape Representation

$$\mathbf{S}_{3F\times P} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & \cdots & \mathbf{X}_{1P} \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \cdots & \mathbf{X}_{2P} \\ \vdots & \vdots & & \vdots \\ \mathbf{X}_{F1} & \mathbf{X}_{F2} & \cdots & \mathbf{X}_{FP} \end{bmatrix}$$

Shape

LINEAR SHAPE MODEL

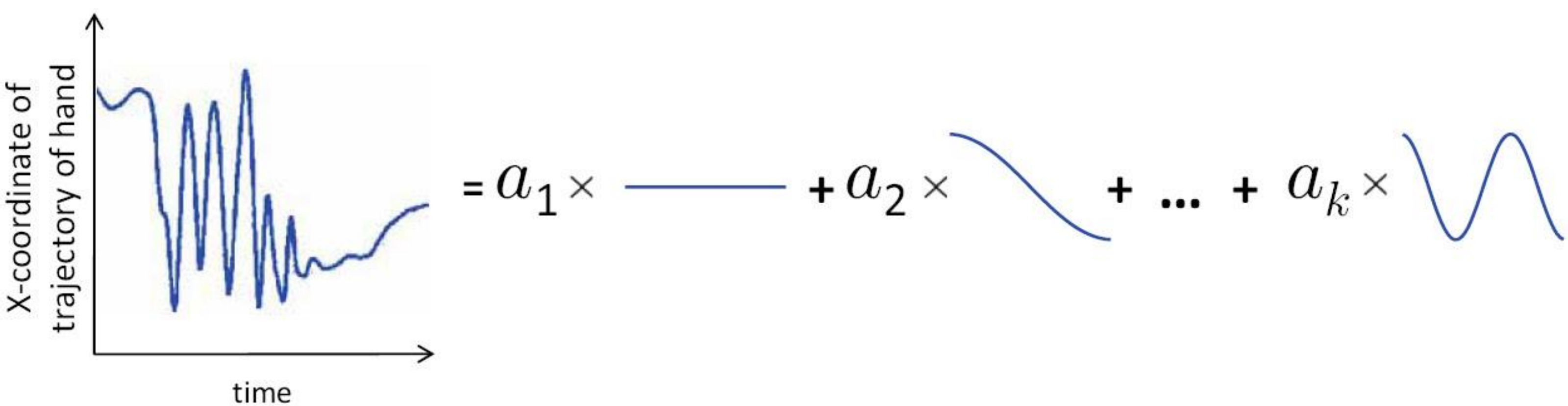


Dynamic Structure

Trajectory Representation

$$\mathbf{S}_{3F\times P} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & \cdots & \mathbf{X}_{1P} \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \cdots & \mathbf{X}_{2P} \\ \vdots & \vdots & & \vdots \\ \mathbf{X}_{F1} & \mathbf{X}_{F2} & \cdots & \mathbf{X}_{FP} \end{bmatrix} \downarrow \text{Trajectory}$$

LINEAR TRAJECTORY MODEL

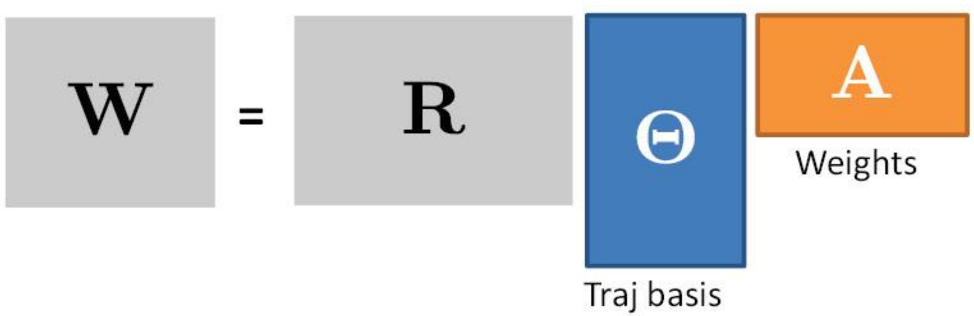


Duality

Weights and Bases

SHAPE FACTORIZATION

TRAJECTORY FACTORIZATION



Shape weights are trajectory basis and trajectory weights are shape basis

UNIVERSITÄT BONN