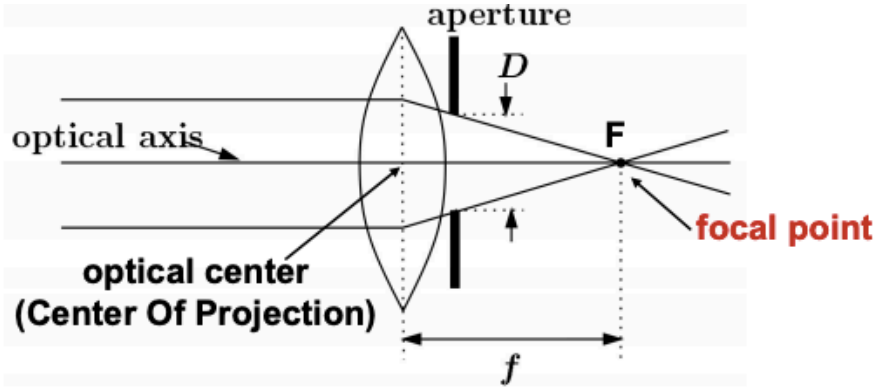
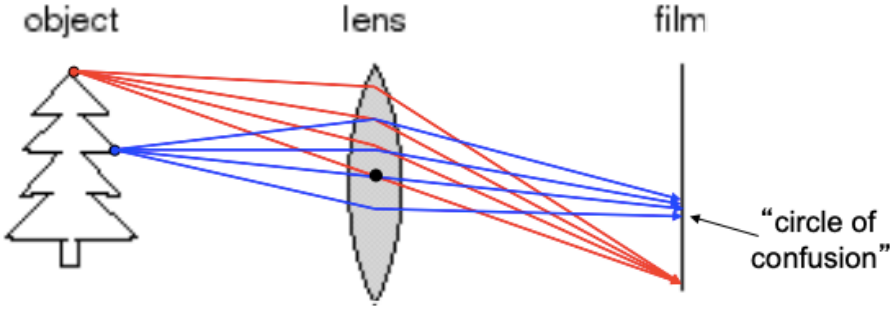
Lenses

from https://courses.cs.washington.edu/courses/cse576/17sp/notes/CamerasStereo17.pdf

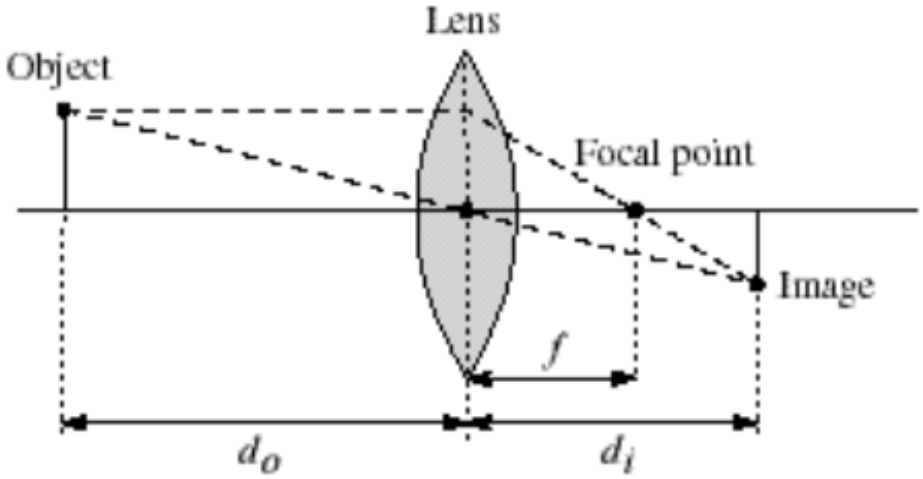




A lens focuses parallel rays onto a single focal point

- focal point at a distance *f* beyond the plane of the lens
  - *f* is a function of the shape and index of refraction of the lens
- Aperture of diameter D restricts the range of rays
  - aperture may be on either side of the lens
- Lenses are typically spherical (easier to produce)
- Real cameras use many lenses together (to correct for aberrations)

Thin lenses



Thin lens equation: $\frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f}$

- Any object point satisfying this equation is in focus

Statistics of matching correspondence: Fundamental Matrix

from “IVPV Handbook of Optics”, Vol II
edited by Michael Bass
http://photonics.intec.ugent.be/education/IVPV/res_handbook/v2ch17.pdf

from “Multiple View Geometry in Computer Vision”
Second Edition by Hartley & Zisserman
https://www.robots.ox.ac.uk/~vgg/hzbook/

(Note, consult authors and books for any use.)

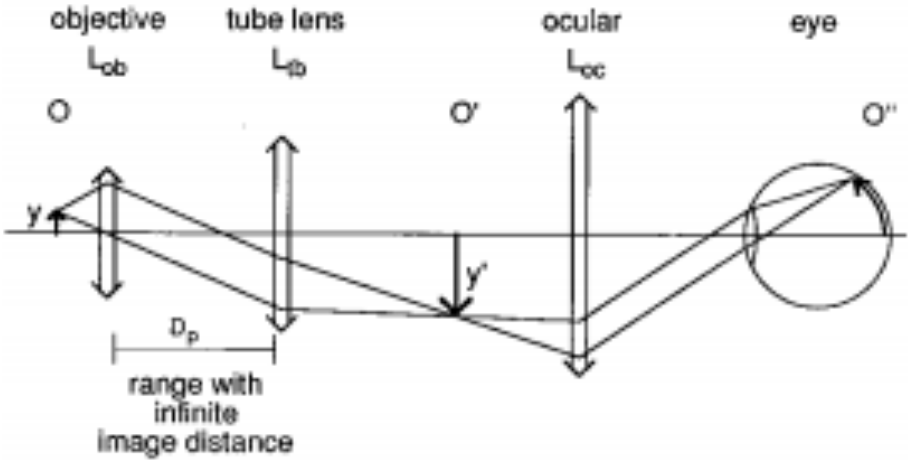


FIGURE 4 Ray path in microscope with infinity-corrected objective and tube lens.

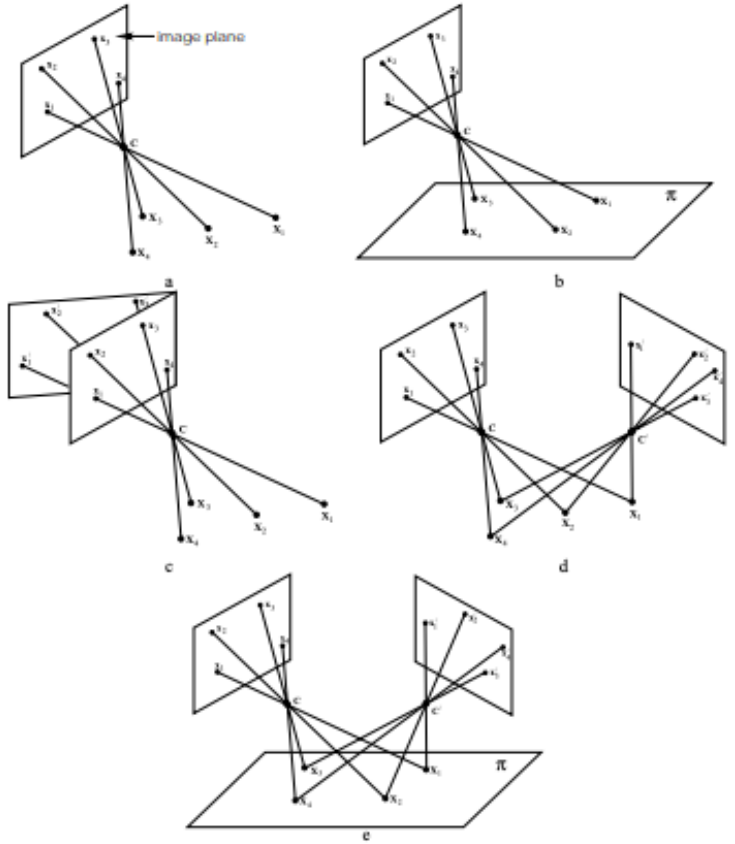


Fig. 1.1. **The camera centre is the essence.** *(a) Image formation: the image points* $\mathbf{x}_i$ *are the intersection of a plane with rays from the space points* $\mathbf{X}_i$ *through the camera centre* $\mathbf{C}$. *(b) If the space points are coplanar then there is a projective transformation between the world and image planes,* $\mathbf{x}_i = \mathrm{H}_{3\times3}\mathbf{X}_i$. *(c) All images with the same camera centre are related by a projective transformation,* $\mathbf{x}'_i = \mathrm{H}'_{3\times3}\mathbf{x}_i$. *Compare (b) and (c) – in both cases planes are mapped to one another by rays through a centre. In (b) the mapping is between a scene and image plane, in (c) between two image planes. (d) If the camera centre moves, then the images are in general not related by a projective transformation, unless (e) all the space points are coplanar.*

2D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

2D Translation. $x' = x + t$

$$x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x}$$

$$\bar{x}' = \begin{bmatrix} I & t \\ 0^T & 1 \end{bmatrix} \bar{x}$$

whereI is the (2×2) identity matrix

2D Affine: $x' = A\bar{x}$

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix} \bar{x}.$$

preserves parallel property of lines.

2D Rotation + translation (a.k.a. rigid body motion, a.k.a. 2D euclidian): $x' = Rx+t$

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x} \qquad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

2D Scaled rotation (a.k.a. similarity transform): $x' = sRx + t$

$$x' = \begin{bmatrix} sR & t \end{bmatrix} \bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x},$$

preserves angles between lines.

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{3\times3}$	8	straight lines	

Table 2.1 Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 2×3 matrices are extended with a third $[0^T\ 1]$ row to form a full 3×3 matrix for homogeneous coordinate transformations.

2D Projective (a.k.a. perspective transformation, homography): $\tilde{x}' = \hat{H}\tilde{x}$

Note that $\tilde{H}$ is homogeneous, i.e., it is only defined up to a scale, Two H matrices that differ only by scale are equivalent.

The resulting homogeneous coordinate $\tilde{x}'$ must be normalized in order to obtain an inhomogeneous result:

$$x' = \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + h_{22}} \quad \text{and} \quad y' = \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + h_{22}}.$$

preserves straightness property of lines.

Planar surface flow: Bilinear interpolant:

$x' = a0 + a1x + a2y + a6x^2 + a7xy$

$y' = a3 + a4x + a5y + a7x^2 + a6xy$

planar surface with a small 3D motion

Bilinear interpolant:

$x' = a0 + a1x + a2y + a6xy$

$y' = a3 + a4x + a5y + a7xy$

useful for interpolation of sparse grids using splines , but does not preserve straight lines

2D transformations

http://www.cs.cmu.edu/~16385/s17/Slides/10.0_2D_Transforms.pdf

Basic 2D transformations as 3x3 matrices

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

Translate

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

Scale

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\Theta & -\sin\Theta & 0 \\ \sin\Theta & \cos\Theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Rotate

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

Shear

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

3D Translation. $x' = x + t$

$$x' = \begin{bmatrix} I & t \end{bmatrix} \bar{x}$$

whereI is the (3×3) identity matrix

3D Affine: $x' = Ax^{-}$

$$x' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{bmatrix} \bar{x}$$

preserves parallel property of lines and planes.

3D Rotation + translation (a.k.a. rigid body motion, a.k.a. 3D euclidian): $x' = Rx+t$

$$x' = \begin{bmatrix} R & t \end{bmatrix} \bar{x}$$

where R is a 3 × 3 orthonormal rotation matrix with $RR^T = I$ and $|R| = 1$.

can describe a rigid motion using
$x' = R(x-c) = Rx - Rc$

where c is the center of rotation (often the camera center).

3D Scaled rotation (a.k.a. similarity transform): $x' = sRx + t$

$$x' = \begin{bmatrix} sR & t \end{bmatrix} \bar{x} = \begin{bmatrix} a & -b & t_x \\ b & a & t_y \end{bmatrix} \bar{x},$$

preserves angles between lines and planes.

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} I \mid t \end{bmatrix}_{3\times4}$	3	orientation	
rigid (Euclidean)	$\begin{bmatrix} R \mid t \end{bmatrix}_{3\times4}$	6	lengths	
similarity	$\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times4}$	7	angles	
affine	$\begin{bmatrix} A \end{bmatrix}_{3\times4}$	12	parallelism	
projective	$\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times4}$	15	straight lines	

Table 2.2 Hierarchy of 3D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 3 × 4 matrices are extended with a fourth $[0^T\ 1]$ row to form a full 4 × 4 matrix for homogeneous coordinate transformations. The mnemonic icons are drawn in 2D but are meant to suggest transformations occurring in a full 3D cube.

3D Projective (a.k.a. perspective transformation, homography): $\tilde{x}' = \hat{H}\tilde{x}$

Note that $\tilde{H}$ is homogeneous, i.e., it is only defined up to a scale, Two H matrices that differ only by scale are equivalent.

The resulting homogeneous coordinate $\tilde{x}'$ must be normalized in order to obtain an inhomogeneous result:

preserves straightness property of lines.

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

Transformation	Matrix	# DoF	Preserves	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	2	orientation	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	3	lengths	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times3}$	4	angles	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times3}$	6	parallelism	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times3}$	8	straight lines	

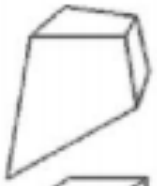
Table 2.1 Hierarchy of 2D coordinate transformations. Each transformation also preserves the properties listed in the rows below it, i.e., similarity preserves not only angles but also parallelism and straight lines. The 2×3 matrices are extended with a third $[\mathbf{0}^T\ 1]$ row to form a full 3×3 matrix for homogeneous coordinate transformations.

Transform	Matrix	Parameters $\boldsymbol{p}$	Jacobian $\boldsymbol{J}$
translation	$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$	(t_x, t_y)	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
Euclidean	$\begin{bmatrix} c_\theta & -s_\theta & t_x \\ s_\theta & c_\theta & t_y \end{bmatrix}$	(t_x, t_y, θ)	$\begin{bmatrix} 1 & 0 & -s_\theta x - c_\theta y \\ 0 & 1 & c_\theta x - s_\theta y \end{bmatrix}$
similarity	$\begin{bmatrix} 1+a & -b & t_x \\ b & 1+a & t_y \end{bmatrix}$	(t_x, t_y, a, b)	$\begin{bmatrix} 1 & 0 & x & -y \\ 0 & 1 & y & x \end{bmatrix}$
affine	$\begin{bmatrix} 1+a_{00} & a_{01} & t_x \\ a_{10} & 1+a_{11} & t_y \end{bmatrix}$	$(t_x, t_y, a_{00}, a_{01}, a_{10}, a_{11})$	$\begin{bmatrix} 1 & 0 & x & y & 0 & 0 \\ 0 & 1 & 0 & 0 & x & y \end{bmatrix}$
projective	$\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$	$(h_{00}, h_{01}, \ldots, h_{21})$	(see Section 6.1.3)

Table 6.1 Jacobians of the 2D coordinate transformations $\boldsymbol{x}' = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{p})$ shown in Table 2.1, where we have re-parameterized the motions so that they are identity for $\boldsymbol{p} = 0$.

3D transformations

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

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 parallelism, 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

Normalize by last coordinate to recover 3D points

$$\begin{bmatrix} \lambda x \\ \lambda y \\ \lambda z \\ \lambda \end{bmatrix}$$

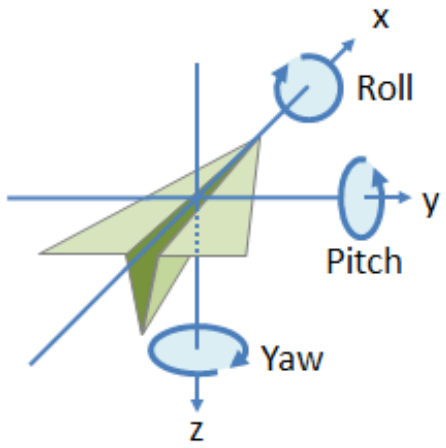
3D transformations: Gimbal Lock

from http://danceswithcode.net/engineeringnotes/rotations_in_3d/rotations_in_3d_part1.html

$$R_x(u) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(u) & -\sin(u) \\ 0 & \sin(u) & \cos(u) \end{bmatrix} \quad \text{(eq 8a)}$$

$$R_y(v) = \begin{bmatrix} \cos(v) & 0 & \sin(v) \\ 0 & 1 & 0 \\ -\sin(v) & 0 & \cos(v) \end{bmatrix} \quad \text{(eq 8b)}$$

$$R_z(w) = \begin{bmatrix} \cos(w) & -\sin(w) & 0 \\ \sin(w) & \cos(w) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{(eq 8c)}$$



Axis of Rotation	Euler Angle Name	Euler Angle Symbol
x	Roll	u
y	Pitch	v
z	Yaw	w

$$R_z(w)R_y(v)R_x(u) = \begin{bmatrix} c(v)\,c(w) & s(u)\,s(v)c(w) - c(u)\,s(w) & s(u)\,s(w) + c(u)\,s(v)\,c(w) \\ c(v)\,s(w) & c(u)\,c(w) + s(u)\,s(v)\,s(w) & c(u)\,s(v)\,s(w) - s(u)\,c(w) \\ -s(v) & s(u)\,c(v) & c(u)\,c(v) \end{bmatrix} \quad \text{(eq 9)}$$

consider what happens in the special case where the pitch angle v = +90° or –90°. Under both of these conditions, cos(v) = 0, and from equation 9 we can see that r11, r21, r32 and r33 must all equal zero. Since the atan2 function is not defined at (0,0), equations 10a and 10c are not valid when the pitch angle v = +/– 90°.

This is the dreaded "gimbal lock." It occurs because, at a pitch angle of + and – 90°, the yaw and roll axes of rotation are aligned with each other in the world coordinate system, and therefore produce the same effect. This means there is no unique solution: any orientation can be described using an infinite number of yaw and roll angle combinations. To handle the gimbal lock condition, we must detect when the pitch angle is +/- 90°. This can be accomplished directly (by finding the pitch angle v using equation 10b), or by testing r31.

If pitch angle v = –90°, then r31 will equal 1, and;

u + w = atan2(– r12,– r13) (eq 11a)

If pitch angle v = +90°, then r31 will equal –1, and;

u – w = atan2(r12,r13) (eq 11b)

In practice, we would set one of the angles to zero and solve for the other. For example, we could set the yaw angle (w) to zero and solve equations 11a and 11b for the roll angle (u).

Note also that although Euler angles are susceptible to gimbal lock, rotation matrices are not. For every possible rotation there is one and only one rotation matrix. Also, for rotation matrices, the mapping is continuous. That is, small changes in rotation will always equate to small changes in the rotation matrix.

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

3D Rotation represented by a rotation axis and angle (a.k.a. Rodriguez’s formula).

rotation axis and angle as a vector $\omega = \theta \hat{n}$

let $[\hat{n}]$ be the matrix form of the cross product operator with the vector $\hat{n} = (\hat{n}_x , \hat{n}_y , \hat{n}_z)$,

$$[\hat{\boldsymbol{n}}]_\times = \begin{bmatrix} 0 & -\hat{n}_z & \hat{n}_y \\ \hat{n}_z & 0 & -\hat{n}_x \\ -\hat{n}_y & \hat{n}_x & 0 \end{bmatrix}$$

and

$$\boldsymbol{R}(\hat{\boldsymbol{n}}, \theta) = \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times + (1 - \cos\theta)[\hat{\boldsymbol{n}}]_\times^2$$

good for small rotations

For infinitesimal or instantaneous) rotations and θ expressed in radians, Rodriguez’s formula simplifies to:

$$\boldsymbol{R}(\boldsymbol{\omega}) \approx \boldsymbol{I} + \sin\theta[\hat{\boldsymbol{n}}]_\times \approx \boldsymbol{I} + [\theta\hat{\boldsymbol{n}}]_\times = \begin{bmatrix} 1 & -\omega_z & \omega_y \\ \omega_z & 1 & -\omega_x \\ -\omega_y & \omega_x & 1 \end{bmatrix}$$

from”Bundle Adjustment — A Modern Synthesis” by Triggs et al.:
Rotations should be parametrized using either quaternions subject to $\|q\|^2 = 1$, or local perturbations $R\delta R$ or $\delta R\, R$ of an existing rotation R, where δR can be any well-behaved 3 parameter small rotation approximation, e.g. $\delta R = (I + [\delta r]_\times)$, the Rodriguez formula, local Euler angles, etc.
State updates: Just as state vectors x represent points in some nonlinear space, state updates $x \rightarrow x + \delta x$ represent displacements in this nonlinear space that often can not be represented exactly by vector addition.

3D transformations

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

3D Rotation represented Unit quaternions:

(Shoemake 1985)

a unit length 4-vector whose components can be written as

$q = (q_x, q_y, q_z, q_w)$ or $q = (x, y, z, w)$

Unit quaternions live on the unit sphere $\|q\| = 1$ and *antipodal* (opposite sign) quaternions, q and –q, represent the same rotation

“origin” $q_o = (0, 0, 0, 1)$.

$$q = (v, w) = (\sin\frac{\theta}{2}\hat{n}, \cos\frac{\theta}{2}).$$

to express Rodriguez’s formula:

$$\begin{aligned} R(\hat{n}, \theta) &= I + \sin\theta[\hat{n}]_\times + (1 - \cos\theta)[\hat{n}]_\times^2 \\ &= I + 2w[v]_\times + 2[v]_\times^2. \end{aligned}$$

$$R(q) = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - zw) & 2(xz + yw) \\ 2(xy + zw) & 1 - 2(x^2 + z^2) & 2(yz - xw) \\ 2(xz - yw) & 2(yz + xw) & 1 - 2(x^2 + y^2) \end{bmatrix}. \quad (2.41)$$

The diagonal terms can be made more symmetrical by replacing $1 - 2(y^2 + z^2)$ with $(x^2 + w^2 - y^2 - z^2)$, etc.

Given two quaternions $q_0 = (v_0, w_0)$ and $q_1 = (v_1, w_1)$,
the *quaternion multiply* operator is defined as:

$$q_2 = q_0 q_1 = (v_0 \times v_1 + w_0 v_1 + w_1 v_0,\ w_0 w_1 - v_0 \cdot v_1).$$

quaternion division:

$$q_2 = q_0 / q_1 = q_0 q_1^{-1} = (v_0 \times v_1 + w_0 v_1 - w_1 v_0,\ -w_0 w_1 - v_0 \cdot v_1).$$

good for tracking of a smoothly moving camera, since there are no discontinuities in the representation. It is also easier to interpolate between rotations and to chain rigid transformations (Murray, Li, and Sastry 1994; Bregler and Malik 1998).

can use quaternions, and update estimates using an incremental rotation, as described in Section 6.2.2

Degrees of Freedom (DOF)

Similarity Transformation in 2D space:
4 DOF: 2 for translation, 1 for rotation, 1 for scale

Affine Transformation in 2D space:
6 DOF: 2 for translation, 1 for rotation, 1 for scale, 1 for aspect ration, 1 for shear

Cartesian 3D space:
6 DOF (three for position and three for orientation)

Rigid Body in D-dimensional space:
translational: d
rotational: d*(d-1)/2

Projective Transformation (homogenous coordinates):
8 DOF: 2 for translation, 1 for rotation, 1 for scale, 1 distortion of x for depth, 1 distortion of y for depth

Rotation in 3D space:

$$R_{z,\theta} = \begin{vmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad R_{x,\theta} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{vmatrix} \quad R_{y,\theta} = \begin{vmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{vmatrix}$$

D. Nister. An efficient solution to the five-point relative pose problem. In Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'03), volume 2, pages 195–202, June 2003

	1 View Known Structure	2 Views Unknown Structure	$n > 2$ Views Unknown Structure
Known intrinsics	Unique	Two-fold or unique	Unique
Unknown fixed focal length	Unique in general	1 d.o.f.	Unique in general
Unknown variable intrinsics	3 d.o.f.	2 d.o.f.	3n-4 d.o.f.

Table 1: The degrees of ambiguity in the face of planar degeneracy for pose estimation and structure and motion estimation. The motion is assumed to be general and the structure is assumed to be dense in the plane. See the text for further explanation.

A few matrix operations used for transformations

because translation is not a linear transformation (see Strang Chap 7), one has to keep it as a separate transformation matrix or vector when performing operations on a sequence of matrices such as inverse and transpose operations.

translation matrix: inverse changes the signs of the translation elements, but not the diagonal.
rotation matrix: inverse is the transpose of rotation matrix.
scaling matrix: inverse is performed on each element, that is, the reciprocal.

(A*B*C)^-1 = (C^-1) * (B^-1) * (A^-1)

NOTE: the normalization suggested by Hartley is explored further in
Chojnacki et al. 2003, "Revisiting Hartley's Normalized Eight-Point Algorithm"
Some excerpts:
xc is the centroid of x coordinates.
yc is the centroid of y coordinates.
s is the root mean square distance of (x-xc,y-yc) to origin divided by sqrt(2)

u1_normalized = T1 * u1; u2_normalized = T2 * u2
denormalized FM = transpose(T2) * FM_normalized * T1.
denormalized u1 = T1^-1 * u1_normalized; denormalized u2 = T2^-1 * u2_normalized
FM_normalized = inverse(transpose(T2)) * FM * inverse(T1)
u2^T * FM * u1 = u2_normalized^T * FM_normalized * u1_normalized = residual

$$T = \begin{vmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & -xc \\ 0 & 1 & -yc \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1/s & 0 & -xc/s \\ 0 & 1/s & -yc/s \\ 0 & 0 & 1 \end{vmatrix}$$

$$T^{-1} = \begin{vmatrix} 1 & 0 & xc \\ 0 & 1 & yc \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} s & 0 & xc \\ 0 & s & yc \\ 0 & 0 & 1 \end{vmatrix}$$

$$T^{-1} * T^{-T} = \begin{vmatrix} 1 & 0 & xc \\ 0 & 1 & yc \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} s^2 & 0 & 0 \\ 0 & s^2 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xc & yc & 1 \end{vmatrix}$$

$$= \begin{vmatrix} s^2 + xc^2 & xc*yc & xc \\ yc*xc & s^2 + yc^2 & yc \\ xc & yc & 1 \end{vmatrix}$$

from that can see $T^{-T} = \begin{vmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xc & yc & 1 \end{vmatrix} = \begin{vmatrix} s & 0 & 0 \\ 0 & s & 0 \\ xc & yc & 1 \end{vmatrix}$

$$T^{T} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -xc & -yc & 1 \end{vmatrix} \begin{vmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \\ -xc/s & -yc/s & 1 \end{vmatrix}$$

$$(T^{T})^{-1} = \left(\begin{vmatrix} 1/s & 0 & 0 \\ 0 & 1/s & 0 \\ 0 & 0 & 1 \end{vmatrix} \right)^{-1} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ xc & yc & 1 \end{vmatrix} = \begin{vmatrix} s & 0 & 0 \\ 0 & s & 0 \\ xc & yc & 1 \end{vmatrix}$$

can see that $(T^{-1})^{T} = (T^{T})^{-1}$

One can write the extrinsic parameter transformation as separate matrices to make inverse operations easier (and can do the same for intrinsic parameter transformation):

$$\begin{pmatrix} x^c \\ y^c \\ z^c \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}} \underbrace{\begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{R}} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

from Wetzstein, “EE 267 Virtual Reality [Stanford] Course Notes: 6-DOF Pose Tracking with the VRduino“

then use

(A*B*C)^-1 = (C^-1) * (B^-1) * (A^-1)

and

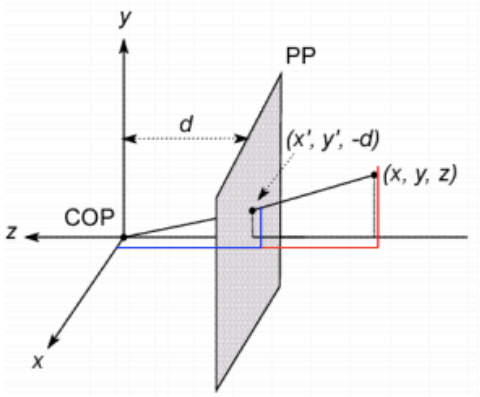
translation matrix: inverse changes the signs of the translation elements, but not the diagonal.

rotation matrix: inverse is the transpose of rotation matrix.

Common Camera Projections

from https://courses.cs.washington.edu/courses/cse576/17sp/notes/CamerasStereo17.pdf

Modeling projection



$y/z = y'/-d$

$y' = -d(y/z)$

The coordinate system

- We will use the pin-hole model as an approximation
- Put the optical center (**C**enter **O**f **P**rojection) at the origin
- Put the image plane (**P**rojection **P**lane) *in front of* the COP
- The camera looks down the *negative* z axis
  - we need this if we want right-handed-coordinates

Projection equations

- Compute intersection with PP of ray from (x,y,z) to COP
- Derived using similar triangles

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

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

$$(x', y') \rightarrow (-d\frac{x}{z}, \ -d\frac{y}{z})$$

12

perspective projection:

Projection is a matrix multiply using homogeneous coordinates:

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

divide by third coordinate

projection matrix | 3D point | 2D point

when z=infinity, we have

"parallel projection" a.k.a. Orthographic

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

Perspective Projection

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

points are projected unto an image plane by dividing them by their z-components, sometimes scaled between near and far clipping planes.

for inhomogeneous and homogeneous coordinates, respectively:

$$\bar{\boldsymbol{x}} = \mathcal{P}_z(\boldsymbol{p}) = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad \tilde{\boldsymbol{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\boldsymbol{p}} \quad \tilde{\boldsymbol{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -z_{\text{far}}/z_{\text{range}} & z_{\text{near}} z_{\text{far}}/z_{\text{range}} \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\boldsymbol{p}},$$

Note, cannot recover the distance to the 3D point after projection.
In comp graphics, projection is usually in 2-steps:
1) project 3D coordinates into *normalized device coordinates* in the range (x, y, z) ∈ [−1, −1] × [−1, 1] × [0, 1]
2) rescale these coordinates to integer pixel coordinates using a *viewport* transformation

disparity, the inverse depth, is defined using $z_{near} = 1$, $z_{far} \rightarrow \infty$, and switching the sign of the third row.

distance to a surface point can be measured using range sensors and stereo matching algorithms.

using that distance or the disparity, the 3D location can be estimated using inverse of a 4X4 matrix.

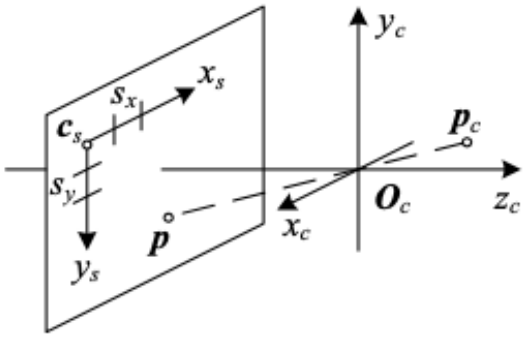
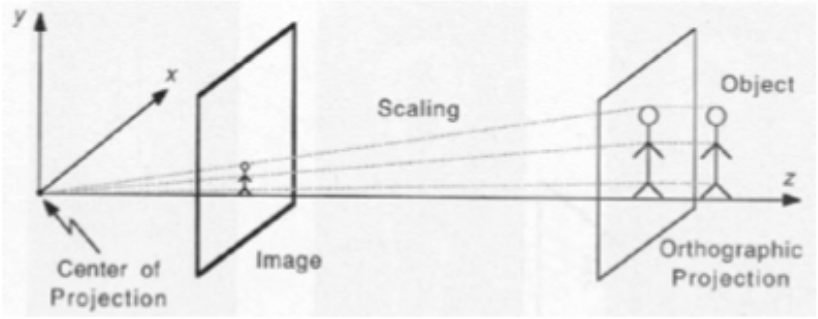


Fig 2.8 Projection of a 3D camera-centered point p_c onto the sensor planes at location p. O_c is the camera center (nodal point), c_s is the 3D origin of the sensor plane coordinate system, and s_x and s_y are the pixel spacings.

Common Camera Projections

http://www.cs.cmu.edu/~16385/s17/Slides/11.2_Camera_Models.pdf

Weak Perspective Camera for distant objects

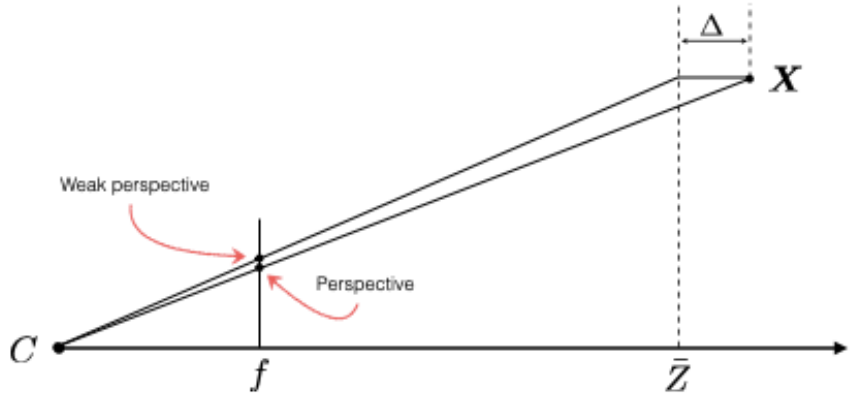


$$\mathbf{P} = \mathbf{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \bar{Z} \end{bmatrix}$$

Affine camera

(assuming that axes are aligned)

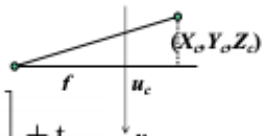
Perspective vs Weak Perspective Projection



Common Camera Projections

https://courses.cs.washington.edu/courses/csep576/11sp/pdf/

3D → 2D Perspective Projection



$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} \mathrm{R} \end{bmatrix}_{3\times 3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathrm{t}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} f & 0 & u_c \\ 0 & f & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

Richard Szeliski Image Stitching 29

Homogeneous coordinates

Is this a linear transformation?

- no—division by z is nonlinear

Trick: add one more coordinate:

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

homogeneous image coordinates — homogeneous scene coordinates

Converting *from* homogeneous coordinates

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

Perspective Projection

Projection is a matrix multiply using homogeneous coordinates:

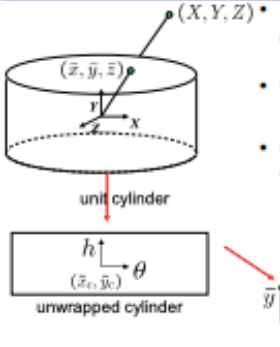
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/d & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ -z/d \end{bmatrix} \Rightarrow (-d\frac{x}{z}, -d\frac{y}{z})$$

divide by third coordinate

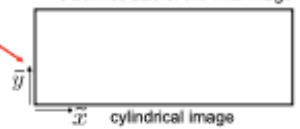
This is known as **perspective projection**

- The matrix is the **projection matrix**

Cylindrical projection



- Map 3D point (X,Y,Z) onto cylinder
  $(\hat{x}, \hat{y}, \hat{z}) = \frac{1}{\sqrt{X^2+Z^2}}(X, Y, Z)$
- Convert to cylindrical coordinates
  $(sin\theta, h, cos\theta) = (\bar{x}, \bar{y}, \bar{z})$
- Convert to cylindrical image coordinates
  $(\bar{x}, \bar{y}) = (s\theta, sh) + (\bar{x}_c, \bar{y}_c)$
  – s defines size of the final image



Richard Szeliski Image Stitching 46

Global alignment

- Register *all* pairwise overlapping images
- Use a 3D rotation model (one R per image)
- Use direct alignment (patch centers) or feature based
- *Infer* overlaps based on previous matches (incremental)
- Optionally *discover* which images overlap other images using feature selection (RANSAC)

Richard Szeliski Image Stitching 75

Perspective Projection

from https://courses.cs.washington.edu/courses/cse576/17sp/notes/CamerasStereo17.pdf

using the negative z direction (constant w/ right-hand coordinate system:

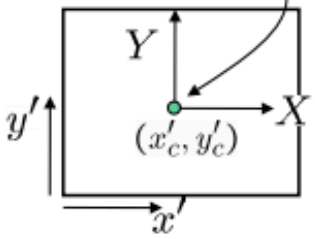
Camera parameters

A camera is described by several parameters

- Translation T of the optical center from the origin of world coords
- Rotation R of the image plane
- focal length f, principal point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"

Projection equation

$$\mathbf{x} = \begin{bmatrix} wx \\ wy \\ w \end{bmatrix} = \begin{bmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \Pi \mathbf{X}$$



- The projection matrix models the cumulative effect of all parameters
- Useful to decompose into a series of operations

identity matrix

$$\Pi = \begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R}_{3x3} & \mathbf{0}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{I}_{3x3} & \mathbf{T}_{3x1} \\ \mathbf{0}_{1x3} & 1 \end{bmatrix} \leftarrow [\text{tx, ty, tz}]^T$$

intrinsics projection rotation translation

- The definitions of these parameters are **not** completely standardized
  - especially intrinsics—varies from one book to another

The Camera Matrix

from “Refinement in 3D Reconstruction using Cheirality Constraints” by Sengupta and Das” (https://pdfs.semanticscholar.org/b603/af27e5e72352ab75782e4b07f5f2aa669a57.pdf)

The Camera matrix contains the information of the camera; both the internal parameters, namely the zoom, focus, skew and external parameters like rotation and the translation with respect to the world coordinate frame. The camera matrix is given as P.

$$P = K[R|\mathbf{t}]$$

where, K contains the camera internal parameters, R and t are rotation and translation parameters.. The internal parameters are arranged in an upper-triangular matrix,

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

where α_x and α_y represent the focal length in terms of pixel dimension in the x and y direction respectively, s is the camera skew and $(x0, y0)^T$ are the coordinates of the principal point.

Cheirality constraints arise from the fact that any point that lies in an image must lie in front of the camera producing that image. The depth of a point X = [X, Y, Z, T], with respect to the camera P = [MIm], is given as

$$depth(\mathbf{X}; P) = \frac{sign(detM)w}{\mathrm{T}||\mathbf{m}^3||}$$

sign(depth(X; P)) is known as the cheirality of point X with respect to camera P.

Perspective Projection, camera matrices

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any use of this. it’s presented here for educational purposes only.)

points are projected unto an image plane by dividing them by their z-components, sometimes scaled between near and far clipping planes.

fo

$$\boldsymbol{p} = \begin{bmatrix} \boldsymbol{R}_s \mid \boldsymbol{c}_s \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \boldsymbol{M}_s \bar{\boldsymbol{x}}_s. \qquad (2.53)$$

sensor homography $\boldsymbol{M}_s$,

where M_s is sensor homography.
parameterized by 8 unknowns:
the 3 parameters describing the rotation R_S,
the 3 parameters describing the translation c_S,
and the two scale factors (s_x , s_y).
often a 3X3 matrix is used
(x_s, y_s) are the pixel coordinates from the image sensor
(s_x, s_y) are the pixel spacings (e.g. in units of microns)
O_c is the camera projection center
c_s is the translation between cameras
Rs is the 3D rotation

The relationship between the 3D pixel center $\boldsymbol{p}$ and the 3D camera-centered point $\boldsymbol{p}_c$ is given by an unknown scaling s, $\boldsymbol{p} = s\boldsymbol{p}_c$. We can therefore write the complete projection between $\boldsymbol{p}_c$ and a homogeneous version of the pixel address $\tilde{\boldsymbol{x}}_s$ as

$$\tilde{\boldsymbol{x}}_s = \alpha \boldsymbol{M}_s^{-1} \boldsymbol{p}_c = \boldsymbol{K} \boldsymbol{p}_c. \qquad (2.54)$$

The 3×3 matrix $\boldsymbol{K}$ is called the *calibration matrix* and describes the camera *intrinsics* (as opposed to the camera’s orientation in space, which are called the *extrinsics*).

K is the camera intrinsic matrix (a.k.a. the calibration matrix), which is used to map 3D camera-centered points p_c to 2D pixel coordinates $\tilde{x}$. It has 7 degrees of freedom, but 8 in practice.

The camera’s orientation in space is called extrinsics (e.g. (R, t))

NOTE that multi-view geometry in practice treats K as an upper-left triangular matrix with 5 degrees of freedom.

Pinhole camera geometry (cont.)

intrinsic parameters:
f is the focal distance in mm
o_x, o_y is the principal point in pixel coordinates
s_x is the width of the pixel in mm
s_y is the height of the pixel in mm
ø is the angle between the axes, usually 90 degrees
The aspect ratio s_x/s_y is usually 1
extrinsic parameters:
Rotation and Translation of the camera in world coord system.

P can be rewritten as P = **K***[I | 0]***G** = **K***[**R** | **t**]

A scale factor **λ** applied is P = **λ*****K***[**R** | **t**]
This is a 3x4 full rank matrix and can be factorized by QR factorization.

P can be rewritten as P = [$\mathbf{P_{3x3}}$ | $\mathbf{p_4}$] where $\mathbf{P_{3x3}}$ is the first 3 rows of P and $\mathbf{p_4}$ is the 4th column.

If **λ**=1, the matrix is normalized and the distance of **M** from the focal plane of the camera is *depth*.

For the image plane at infinity, the image of points doesn't depend on camera position.
The angle between 2 rays is cos θ = $\mathbf{m1^T * \omega * m2 / (}$sqrt$\mathbf{(m1^T * \omega * m1) * (}$sqrt$\mathbf{(m2^T * \omega * m2)}$

(Notes on pinhole camera followed by a few notes on multi-view geometry are from
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html)

Algorithms for Linear Least Squares Problems

Algorithms for Linear Least Squares Problems

January 1991

DOI: 10.1007/978-3-642-76717-3_3

https://www.researchgate.net/publication/268856866_Algorithms_for_Linear_Least_Squares_Problems

orthographic and para-perspective projections

Shape and Motion from Image Streams: a Factorization Method
Full Report on the Orthographic Case

Carlo Tomasi Takeo Kanade

March 1992
Cornell TR 92-1270 and Carnegie Mellon CMU-CS-92-104

A Paraperspective Factorization Method for Shape and Motion Recovery

Conrad J. Poelman and Takeo Kanade

School of Computer Science, Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213-3890
(Conrad.Poelman@cs.cmu.edu, tk@cs.cmu.edu)

In a shape-from-motion problem, we are given a sequence of F images taken from a camera that is moving relative to an object. We locate P prominent feature points in the first image, and track these points from each image to the next, recording the coordinates (u_{fp}, v_{fp}) of each point p in each image f. Each feature point p that we track corresponds to a single world point, located at position $\mathbf{s}_p$ in some fixed world coordinate system. Each image f was taken at some camera orientation, which we describe by the orthonormal unit vectors $\mathbf{i}_f$, $\mathbf{j}_f$, and $\mathbf{k}_f$, where $\mathbf{i}_f$ and $\mathbf{j}_f$ correspond to the x and y axes of the camera's image plane, and $\mathbf{k}_f$ points along the camera's line of sight. We describe the position of the camera in each frame f by the vector $\mathbf{t}_f$ indicating the camera's focal point. This formulation is illustrated in Fig. 1.

The result of the feature tracker is a set of P feature point coordinates (u_{fp}, v_{fp}) for each of the F frames of the image sequence. From this information, our goal is to estimate the shape of the object as $\hat{\mathbf{s}}_p$ for each object point, and the motion of the camera as $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, $\hat{\mathbf{k}}_f$ and $\hat{\mathbf{t}}_f$ for each frame in the sequence.

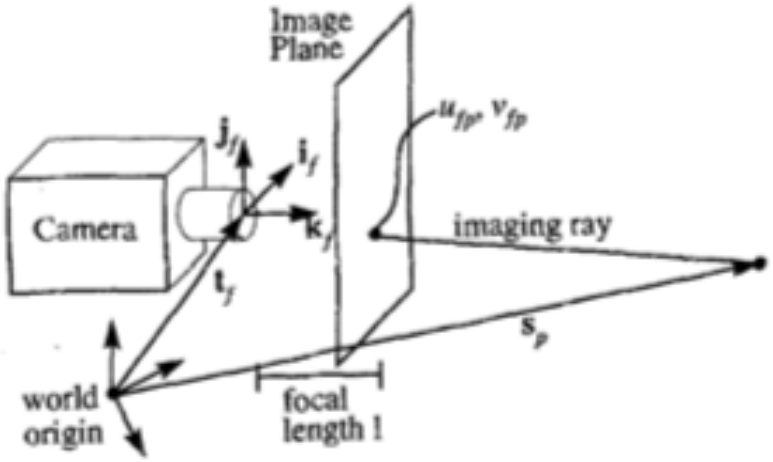


Fig. 1. Coordinate system

orthographic and para-perspective projections

orthographic

$$\mathbf{W} = \begin{bmatrix} U \\ V \end{bmatrix}$$

is called the *measurement matrix*. The rows of the matrices **U** and **V** are then registered by subtracting from each entry the mean of the entries in the same row:

$$\tilde{u}_{fp} = u_{fp} - a_f$$
$$\tilde{v}_{fp} = v_{fp} - b_f \qquad (1)$$

where

$$a_f = \frac{1}{P}\sum_{p=1}^{P} u_{fp}$$

$$b_f = \frac{1}{P}\sum_{p=1}^{P} v_{fp}$$

This produces two new $F \times P$ matrixes $\tilde{\mathbf{U}} = [\tilde{u}_{fp}]$ and $\tilde{\mathbf{V}} = [\tilde{v}_{fp}]$. The matrix

$$\tilde{W} = \begin{bmatrix} \tilde{U} \\ \tilde{V} \end{bmatrix} \qquad (2)$$

is called the *registered measurement matrix*. This is the input to our factorization method.

Using the SVD t $\tilde{W} = RS$

```
r = MatrixUtil.multiply(svd(wc).u3,
      sqrt(svd(wc).s3));
s = MatrixUtil.multiply(sqrt(svd(wc).s3),
      svd(wc).vT3);
```

paraperspective

We can combine (3), for all points p from 1 to P, and all frames f from 1 to F, into the single matrix equation

$$\begin{bmatrix} u_{11} & \cdots & u_{1P} \\ \cdots & \cdots & \cdots \\ u_{F1} & \cdots & u_{FP} \\ v_{11} & \cdots & v_{1P} \\ \cdots & \cdots & \cdots \\ v_{F1} & \cdots & v_{FP} \end{bmatrix} = \begin{bmatrix} \mathbf{m}_1 \\ \cdots \\ \mathbf{m}_F \\ \mathbf{n}_1 \\ \cdots \\ \mathbf{n}_F \end{bmatrix} \begin{bmatrix} \mathbf{s}_1 & \cdots & \mathbf{s}_P \end{bmatrix} + \begin{bmatrix} x_1 \\ \cdots \\ x_F \\ y_1 \\ \cdots \\ y_F \end{bmatrix} \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}, \qquad (7)$$

or in short

$$W = MS + T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}. \qquad (8)$$
$$W^* = W - T\begin{bmatrix} 1 & \cdots & 1 \end{bmatrix} = MS.$$

Using the SVD t $W^* = \hat{M}\hat{S}$,

```
m = MatrixUtil.multiply(svd(wc).u3, sqrt(svd(wc).s3));
s = MatrixUtil.multiply(sqrt(svd(wc).s3),svd(wc).vT3);
```

orthographic and para-perspective projections

orthographic

Impose the metric constraints,

$$\hat{i}_f^T QQ^T \hat{i}_f = 1, \quad \hat{j}_f^T QQ^T \hat{j}_f = 1, \quad \hat{i}_f^T QQ^T \hat{j}_f = 0$$

where Q is a 3×3 matrix, and $\hat{i}_f^T$, $\hat{j}_f^T$ are elements of $\hat{R}$.

where the $3F$×6 matrix G, the 6×1 vector I, and the $3F$×1 vector c are defined by

$$G = \begin{bmatrix} g^T(i_1,i_1) \\ \vdots \\ g^T(i_F,i_F) \\ g^T(j_1,j_1) \\ \vdots \\ g^T(j_F,j_F) \\ g^T(i_1,j_1) \\ \vdots \\ g^T(i_F,j_F) \\ g^T(i_1,j_1) \\ \vdots \\ g^T(i_F,j_F) \end{bmatrix}, I = \begin{bmatrix} I_1 \\ \vdots \\ I_6 \end{bmatrix}, c = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \quad g(a,b) = \begin{bmatrix} a_1b_1 \\ a_1b_2+a_2b_1 \\ a_1b_3+a_3b_1 \\ a_2b_2 \\ a_2b_3+a_3b_2 \\ a_3b_3 \end{bmatrix}.$$

define $L = QQ^T$ and solve the linear system of equations for L and use Cholesky decomposition to get Q.

$$GI = c, \quad I = G^* c. \quad L = \begin{bmatrix} I_1 I_2 I_3 \\ I_2 I_4 I_5 \\ I_3 I_5 I_6 \end{bmatrix},$$

$$L = U\Sigma U^T \quad Q = U\Sigma_+^{1/2} \quad R_0 = [i_1 j_1 k_1]$$

$$R = RR_0, \quad S = R_0^T S,$$

paraperspective

The metric constraints are as followings:

$$|m_f|^2 = \hat{m}_f^T QQ^T \hat{m}_f, \quad |n_f|^2 = \hat{n}_f^T QQ^T \hat{n}_f, \quad m_f \cdot n_f = \hat{m}_f^T QQ^T \hat{n}_f$$

where Q is a 3×3 matrix, and $\hat{m}_f, \hat{n}_f$ are elements of the estimated motion matrix $\hat{M}$.

use m for i; use n for j

$$G = \begin{bmatrix} g^T(i_1,i_1) \\ \vdots \\ g^T(i_F,i_F) \\ g^T(j_1,j_1) \\ \vdots \\ g^T(j_F,j_F) \\ g^T(i_1,j_1) \\ \vdots \\ g^T(i_F,j_F) \end{bmatrix}, I = \begin{bmatrix} I_1 \\ \vdots \\ I_6 \end{bmatrix}, c = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

populate c with these constraints:

$$\frac{|m_f|^2}{1+x_f^2} - \frac{|n_f|^2}{1+y_f^2} = 0$$

$$m_f \cdot n_f - x_f y_f \frac{1}{2}\left(\frac{|m_f|^2}{1+x_f^2} + \frac{|n_f|^2}{1+y_f^2}\right) = 0$$

$$|m_1| = 1.$$

orthographic and para-perspective projections

paraperspective motion recovery

$$A = (L\Lambda^{1/2})$$

Once the matrix A has been determined, we compute the shape matrix $S = A^{-1}S$ and the motion matrix $M = \hat{M}A$. For each frame f, we now need to recover the camera orientation vectors $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$, as well as the depth to the object z_f, from the vectors $\mathbf{m}_f$ and $\mathbf{n}_f$, which are the rows of M. From (6) we see that

$$\hat{\mathbf{i}}_f = z_f\mathbf{m}_f + x_f\hat{\mathbf{k}}_f \qquad \hat{\mathbf{j}}_f = z_f\mathbf{n}_f + y_f\hat{\mathbf{k}}_f. \tag{19}$$

Since the $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$ produced must be orthonormal, they can be written as functions of only three rotational variables. We can then view the problem as, for each frame f, solving an overconstrained system of 6 equations (the expansion of (19) to each of its vector components) in 4 variables (the three rotational variables and z_f). These small

depth z_f from (15) $\quad \dfrac{|\mathbf{m}_f|^2}{1+x_f^2} = \dfrac{|\mathbf{n}_f|^2}{1+y_f^2} \qquad \left(= \dfrac{1}{z_f^2} \right)$. (15)

Once we know x_f, y_f, z_f, $\hat{\mathbf{i}}_f$, $\hat{\mathbf{j}}_f$, and $\hat{\mathbf{k}}_f$, we can calculate $\mathbf{t}_f$ using (4) and (5).

$$z_f = -\mathbf{t}_f \cdot \mathbf{k}_f \tag{4}$$

$$x_f = -\frac{\mathbf{t}_f \cdot \mathbf{i}_f}{z_f} \qquad y_f = -\frac{\mathbf{t}_f \cdot \mathbf{j}_f}{z_f} \tag{5}$$

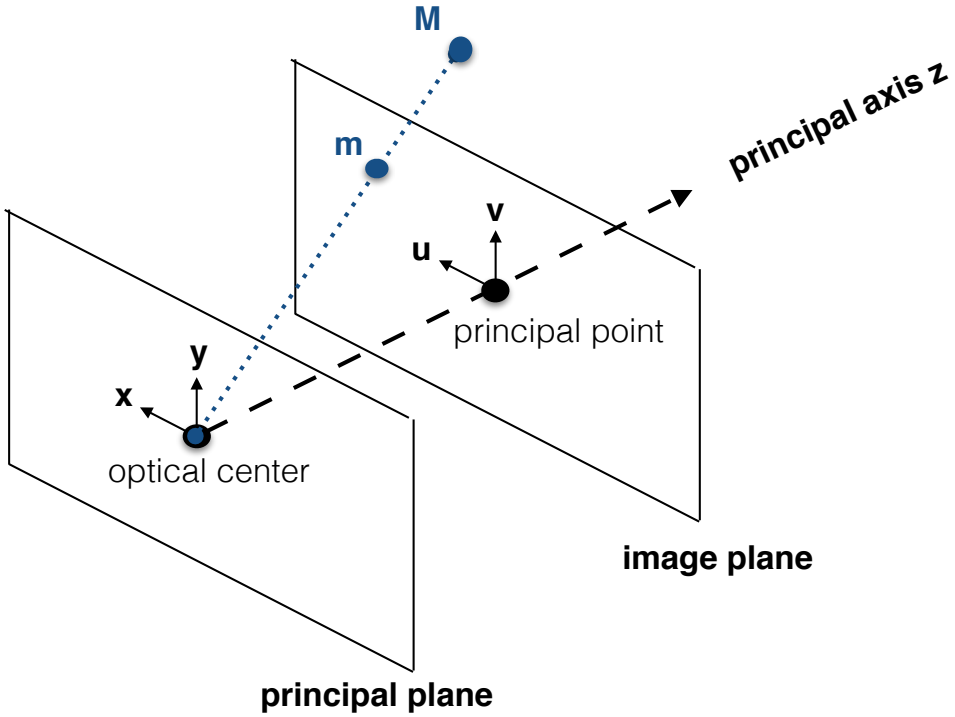
Fundamental Matrix: 7 point algorithm

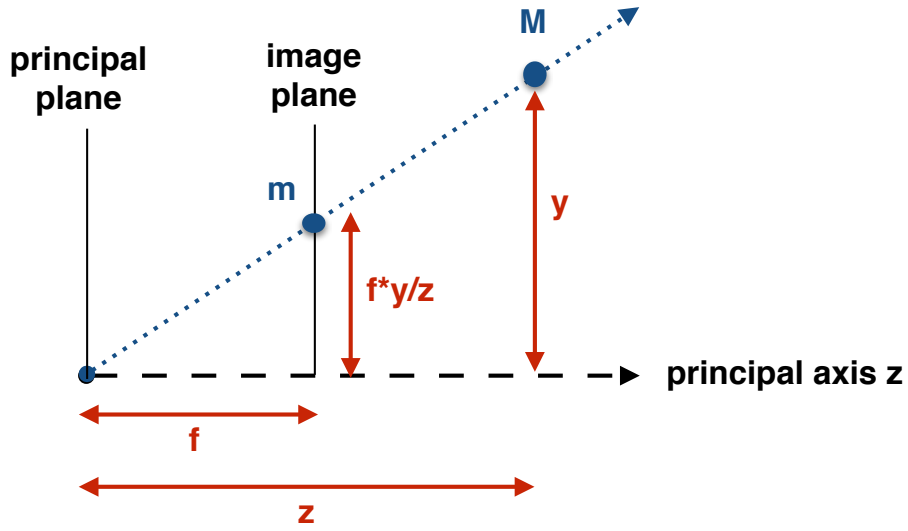
(see src/…/EpipolarTransformer.java)
(see https://www.robots.ox.ac.uk/~vgg/hzbook/code/vgg_multiview/vgg_F_from_7pts_2img.m from "Multiple View Geometry in Computer Vision
Second Edition, by Hartley and Zisserman)

7-Point algorithm (for use with uncalibrated cameras):

```
(1)Transform the coordinates
(2)build matrix A with the normalized x,y points
(3) compute linear least square solution to the least eigenvector of f.
     solve A = U * D * V^T   for A*f = [..x...]*f = 0
     A has rank 7.  f has rank 2.  calculate [U,D,V] from svd(A)
(4) [U,D,V] = svd(F,0);
     The homogeneous system AX = 0 : X is the null space of matrix A.
     The system is nullable because rank 7 < number of columns, 9.
     The nullable system must have a solution other than trivial where|A| = 0.
     There should be 9-7=2 linearly independent vectors u1, u2, ... , un-r that span the null space of A.
     The right null space of A reduced by SVD is then 2D and the last
        2 columns of V can be extracted and reshaped to [3x3] as F1 and F2.
     A linear convex combination of F1 and F2 form the estimate of F.
     F = α*F1 + (1 - α)*F2  where α is between 0 and 1
     The eigenvalues of F are possible only if the determinant of F is 0.
           det A = 0 ==> det(α*F1 + (1 - α)*F2) = 0
           because det(F1 + F2) != det(F1) + det(F2), have to step through the
             determinant of the sums, and group the terms by a^3, a^2, a^1, and a^0
             and then solve for the cubic roots as the values of 'a'.
     The determinant of F is a polynomial function, the characteristic
        polynomial whose degree is the order of the matrix, which is 3 in this
        case. Therefore, the answer(s) to determinant(F) = 0 requires the cubic
        roots of the equation.
     After the cubic root(s) are solved, they are back substituted into :
        Fi = a(i) * FF{1} + (1-a(i)) * FF{2};
        to get the solutions Fi which may be one or 3 solutions
(5) denormalize the fundamental matrix
     The related part of the normalization equation: inv(T_2) * F * inv(T_1)
     so denormalizing is:
         F = (T_1)^T * F * T_2
(6) validate the 1 to 3 solutions:
(7) estimate the error in the fundamental matrix by calculating epipolar
     lines for points in image 1 and find their nearest points in image 2
     and measure the perpendicular distance from the epipolar line for
     those nearest points.
```

Pinhole camera geometry





camera projection matrix P is defined in z***m** = P***M**
where M = $(x,y,z,1)^T$ and m = $(f*x/z, f*y/z, 1)^T$

The right side of the projection eqn can be modified by rotation matrix **R** and translation vector **t** to put the coordinates in the (external) world coordinate system. The matrix is | **R t** |
The six parameters are called *external parameters*. | 0 1 |
rows of **R** and the *optical center* describe the *camera reference frame* in world coordinates.

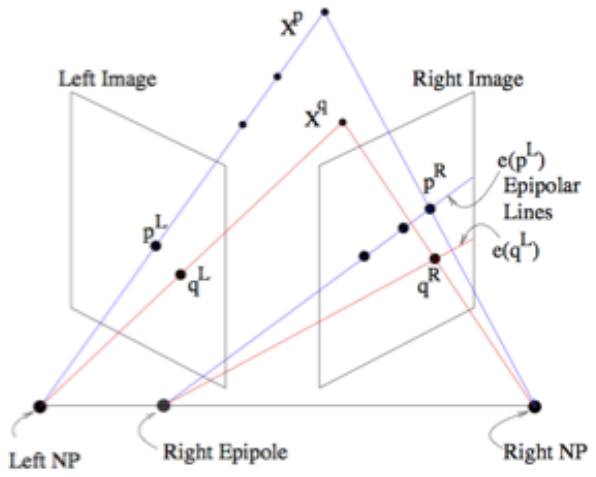
The left side of the projection eqn can be modified to change coordinates in the image frame.

$$K = \begin{vmatrix} f/s_x & f/s_x*\cot\theta & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{vmatrix}$$

K is the *camera calibration matrix,*
result is pixel coords in image plane.

Projection, stereo and panoramic images

X^p and X^q are the physical location of objects.



http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf
(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

$(X_L)^T * F * X_R = 0$ for any pair or points in the images.

Note: the "Essential matrix" is a matrix used if the camera details are known. The "bifocal tensor", a.k.a. "fundamental matrix" does not need camera details.

NP are the nadir points of the cameras.

The line connecting the left and right NP is the baseline.

The projection of the left nadir, NP is seen as a the left epipole w.r.t. right image. The epipoles may or may not be within the border of the images.

The projection of X^p to left nadir, NP is seen as an epipole line in the right image. If more than one epipole line is present in the right image, they converge at the right epipole (which might not be within the boundaries of the image).

Once the points in the left image, X_L are matched with points in the right image, X_R, if there are at least 7 points, one can determine the "bifocal tensor",
a.k.a. "fundamental matrix, relating the points in the 2 images using a 3x3 matrix of rank 2.

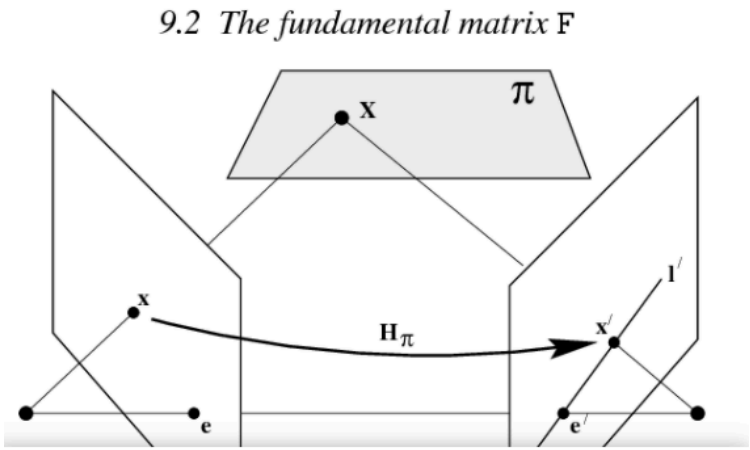


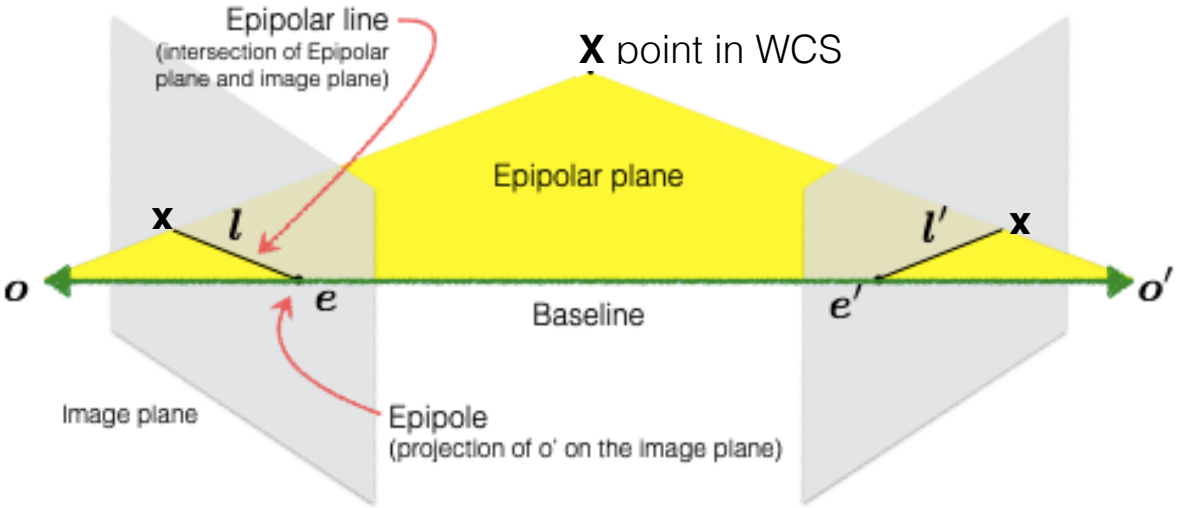
Fig. 9.5. A point x in one image is transferred via the plane π to a matching point x' in the second image. The epipolar line through x' is obtained by joining x' to the

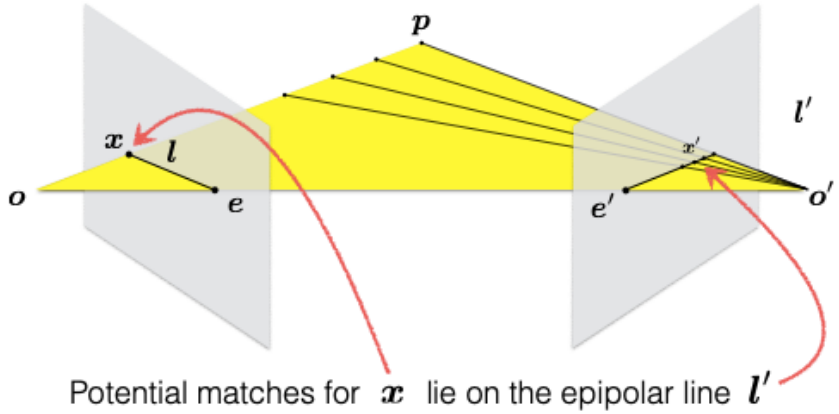
Multiple View Geometry in Computer Vision by Wrobel, 2001

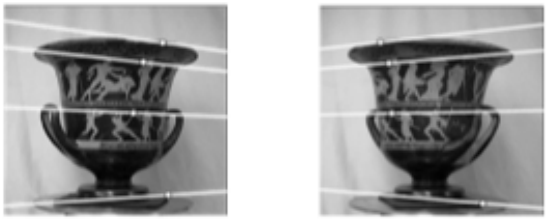
Epipolar Geometry

http://www.cs.cmu.edu/~16385/s17/Slides/12.1_Epipolar_Geometry.pdf

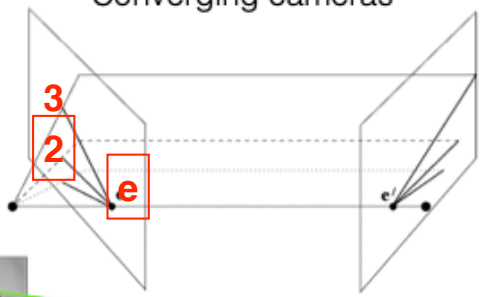
http://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf

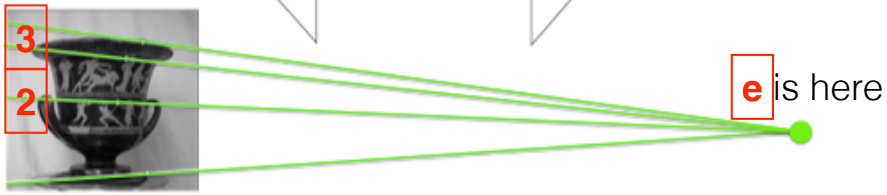




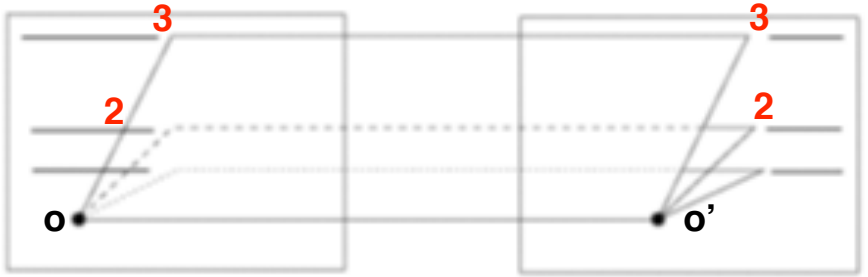


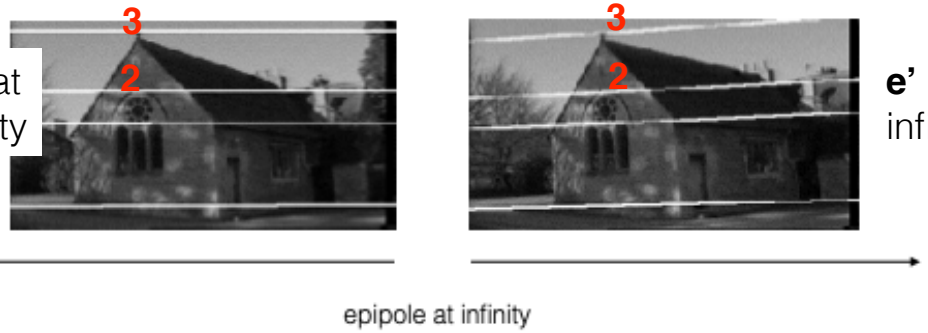
Converging cameras





Parallel cameras





Epipolar Geometry

https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html

The epipolar geometry describes the geometric relationship between two perspective views of the same 3D scene.

3 cases:

Intrinsic and extrinsic parameters known:

can describe the epipolar geometry in terms of projection matrices.

intrinsic parameters only known:

need to normalize the coordinates.

can describe the epipolar geometry in terms of the essential matrix

neither intrinsic nor extrinsic known:

can describe the epipolar geometry in terms of the fundamental matrix

Epipolar Geometry: intrinsic and extrinsic params known

https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html

Intrinsic and extrinsic parameters known:

can describe the epipolar geometry in terms of projection matrices.

uses the optical ray to derive the epipolar line $\lambda^*\mathbf{m} = P^*\mathbf{M}$ where the optical ray is the locus of points projected onto **m**. it's the parametric line passing thru camera projection center and a special point at infinity that projects onto **m**.

$$\mathbf{M} = \begin{pmatrix} -P_{3\times3}^{-1}\mathbf{p}_4 \\ 1 \end{pmatrix} + \zeta \begin{pmatrix} P_{3\times3}^{-1}\mathbf{m} \\ 0 \end{pmatrix}, \quad \zeta \in \mathbb{R}.$$

where ζ is λ

$$\zeta_r \mathbf{m}_r = P_r \mathbf{M} = \underbrace{P_r \begin{pmatrix} -P_{3\times3,\ell}^{-1}\mathbf{p}_{4,\ell} \\ 1 \end{pmatrix}}_{\mathbf{e}_r} + \zeta_\ell P_r \begin{pmatrix} P_{3\times3,\ell}^{-1}\mathbf{m}_\ell \\ 0 \end{pmatrix}$$

$\mathbf{m}_\ell$' is the projection onto the right image plane of the point at infinity of the optical ray of $\mathbf{m}_\ell$

$$\mathbf{m}_\ell' = P_{3\times3,r} P_{3\times3,l}^{-1} \mathbf{m}_\ell$$

then the right epipolar line is the equation of a line through the right epipole $\mathbf{e}_r$ and $\mathbf{m}_\ell$'

$$\zeta_r \mathbf{m}_r = \mathbf{e}_r + \zeta_\ell P_{3\times3,r} P_{3\times3,l}^{-1} \mathbf{m}_\ell$$

The equation for the left epipolar line is obtained in a similar way.

Epipolar Geometry: intrinsic params known

https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html

intrinsic parameters only known:

need to normalize the coordinates.

can describe the epipolar geometry in terms of the essential matrix

***normalize the coordinates*: m = K^{-1}m (**can use *Camera.pixelToCameraCoordinates())*

$P_\ell = [I|0] \quad \text{and} \quad P_r = [R|\mathbf{t}].$ **with** $\zeta_r \mathbf{m}_r = \mathbf{e}_r + \zeta_\ell P_{3\times3,r} P_{3\times3,l}^{-1} \mathbf{m}_\ell$

==> $\zeta_r \mathbf{m}_r = \mathbf{t} + \zeta_\ell R \mathbf{m}_\ell;$

In homogeneous coordinates the homogeneous line through two points is expressed as their cross product:

$\mathbf{m}_r^T (\mathbf{t} \times R\mathbf{m}_\ell) = 0,$ **==>** $\mathbf{m}_r^T [\mathbf{t}]_\times R\mathbf{m}_\ell = 0.$

Essential Matrix $E \triangleq [\mathbf{t}]_\times R,$

$\mathbf{m}_r^T E \mathbf{m}_\ell = 0.$

The ***essential matrix holds information on the extrinsic parameters*** and is a singular matrix.
It has 5 DOF (3 rotation and 2 translation)

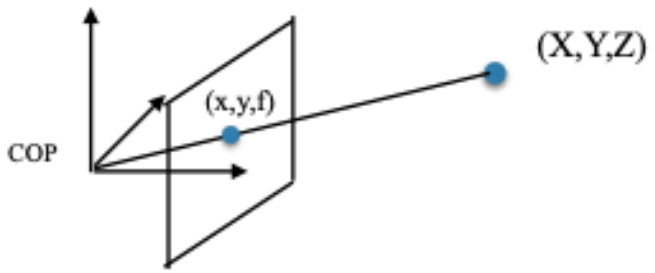
Intrinsics Calibrated 2-View Geometry

http://16720.courses.cs.cmu.edu/lec/two-view_lec14.pdf

Epipolar geometry (Calibrated)

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + \boldsymbol{T}$$

Projecting from camera coordinate system to image coordinates



$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\lambda \mathbf{x} = K\mathbf{X}$$

$$K = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{X}_1 = \lambda_1 \boldsymbol{x}_1, \quad \boldsymbol{X}_2 = \lambda_2 \boldsymbol{x}_2$$

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + \boldsymbol{T}$$

1. Take (left) cross product of both sides with T
2. Take (left) dot product of both sides with $\mathbf{x}_2$

$$\lambda_2 \hat{T}\boldsymbol{x}_2 = \hat{T}R\lambda_1 \boldsymbol{x}_1 + \underbrace{\hat{T}T}_{=\mathbf{0}}$$

$$\lambda_2 \underbrace{\boldsymbol{x}_2^\top \hat{T}\boldsymbol{x}_2}_{=0} = \boldsymbol{x}_2^\top \hat{T}R\lambda_1 \boldsymbol{x}_1$$

E is *essential* matrix

$$\boldsymbol{x}_2^\top \hat{T}R\boldsymbol{x}_1 = 0 \rightarrow \mathbf{x}_2 \cdot (\mathbf{T} \times R\mathbf{x}_1) = 0 \rightarrow \boxed{\boldsymbol{x}_2^\top E\boldsymbol{x}_1 = 0}$$

Epipolar Geometry

https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html

neither intrinsic nor extrinsic known:

can describe the epipolar geometry in terms of the fundamental matrix.

$P_\ell = K_\ell[I|0]$ and $P_r = K_r[R|\mathbf{t}]$. **with** $\zeta_r \mathbf{m}_r = \mathbf{e}_r + \zeta_\ell P_{3\times3,r} P_{3\times3,l}^{-1} \mathbf{m}_\ell$

$$\zeta_r \mathbf{m}_r = \mathbf{e}_r + \zeta_\ell K_r R K_\ell^{-1} \mathbf{m}_\ell \quad \text{with} \quad \mathbf{e}_r = K_r \mathbf{t},$$

==>

which states that point $\mathbf{m}_r$ lies on the line through $\mathbf{e}_r$ and $K_r R K_\ell^{-1} \mathbf{m}_\ell$. As in the case of the essential matrix, this can be written in homogeneous coordinates as:

$$\mathbf{m}_r^T [\mathbf{e}_r]_\times K_r R K_\ell^{-1} \mathbf{m}_\ell = 0.$$

The **fundamental matrix** $F = [\mathbf{e}_r]_\times K_r R K_\ell^{-1}$ ==> $\mathbf{m}_r^T F \mathbf{m}_\ell = 0.$

F is 3x3 and has 7 DOF

as on other slides: $\mathbf{l}_r = F\mathbf{m}_\ell.$ and $\mathbf{l}_\ell = F^T \mathbf{m}_r.$

right null vector: and left null vector

$F\mathbf{e}_\ell = 0$ $\mathbf{e}_r^T F = 0$

$F = K_r^{-T} E K_\ell^{-1}.$ and $\zeta_r F^T \mathbf{m}_r = \zeta_\ell (e_\ell \times \mathbf{m}_\ell)$

rank of F is 8 if number of points = 8, rank is 9 if number of points > 8

In the 8-point algorithm, a singularity constraint ensures that the epipolar lines meet in a common epipole

Uncalibrated 2-View Geometry

http://16720.courses.cs.cmu.edu/lec/two-view_lec14.pdf

Uncalibrated case

$$\lambda_1 \mathbf{x_1} = K_1 \mathbf{X}_1$$
$$\lambda_2 \mathbf{x_2} = K_2 \mathbf{X}_2$$

$$\boxed{\boldsymbol{X}_2 = R\boldsymbol{X}_1 + \boldsymbol{T}}$$

$$\boldsymbol{x}_2^\top \widehat{T} R \boldsymbol{x}_1 = 0$$

$$E = \hat{T}R$$

$$F = {K_2}^{-T} E K_1^{-1}$$

F is *fundamental* matrix

$$\mathbf{x_2}^T {K_2}^{-T} \hat{T} R K_1^{-1} \mathbf{x_1} = 0$$
$$\mathbf{x_2}^T F \mathbf{x_1} = 0$$
$$\mathbf{m}_r^T F \mathbf{m}_\ell = 0 \iff \mathrm{vec}(\mathbf{m}_r^T F \mathbf{m}_\ell) = 0$$
$$\iff (\mathbf{m}_r^T \otimes \mathbf{m}_\ell^T)\,\mathrm{vec}\, F = 0$$

this can also be written as:

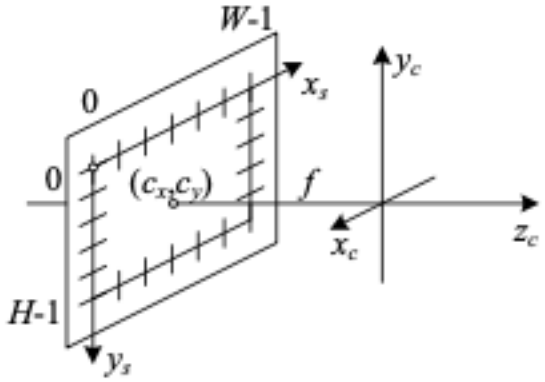
$$F = [\mathbf{e}_2]_\times K_2 \; [\mathbf{t}]_\times^{-1} E \quad K_1^{-1}$$

$$K_2^{-T}[\mathbf{t}]_\times$$

Perspective Projection, camera matrices

from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any of this. use it’s presented here for educational purposes only.)

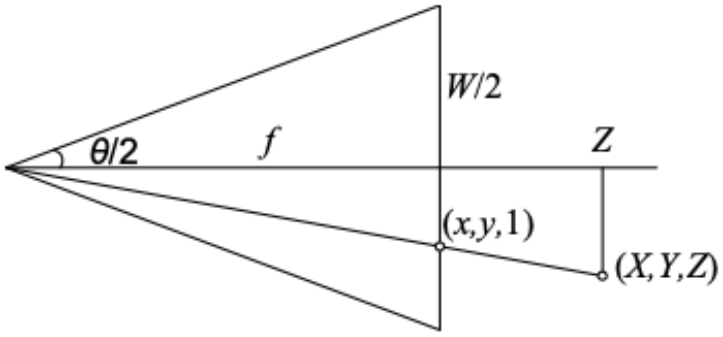
Fig 2.9 Simplified camera intrinsics showing the focal length f and the optical center (c_x,c_y). The image width and height are W and H.



$$\boldsymbol{K} = \begin{bmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

(c_x,c_y) is the *optical center* in pixel coordinates

if set the origin at roughly the center of the image, e.g., (cx,cy) = (W/2,H/2), where W and H are the image height and width, the camera model has a single unknown, i.e., the focal length f. θ is the FOV. $f = (W/2)(\tan(\theta/2))^{-1}$



camera matrix

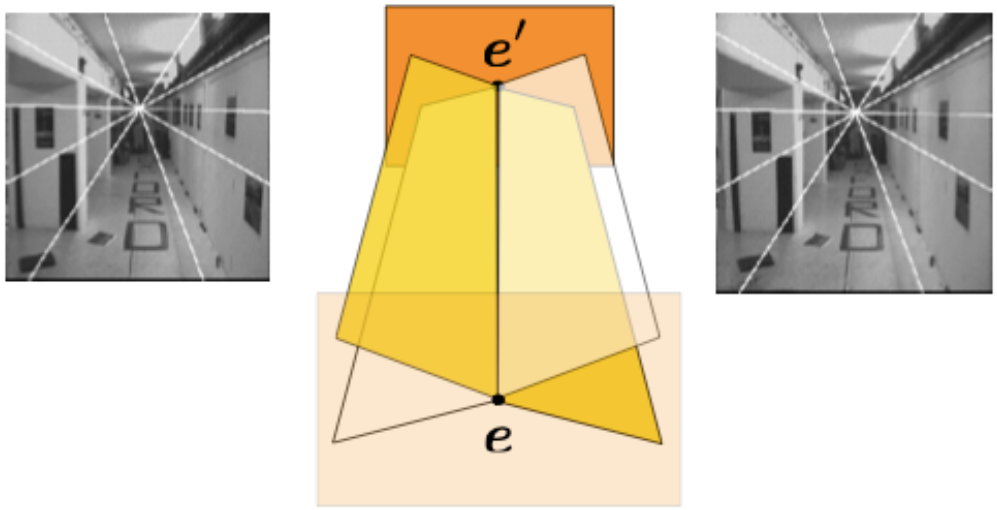
$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}|\boldsymbol{t}]$$

Epipolar Projection and images

http://www.cs.cmu.edu/~16385/s17/Slides/12.1_Epipolar_Geometry.pdf

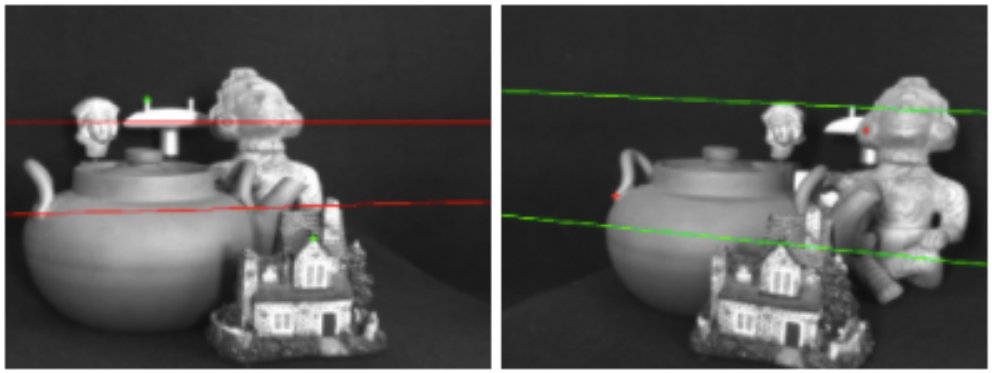
http://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf

Forward moving camera



Epipole has same coordinates in both images.
Points move along lines radiating from "Focus of expansion"

Rotating camera:



https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html

Revisiting Homographies

http://16720.courses.cs.cmu.edu/lec/transformations.pdf

Projection of planar points

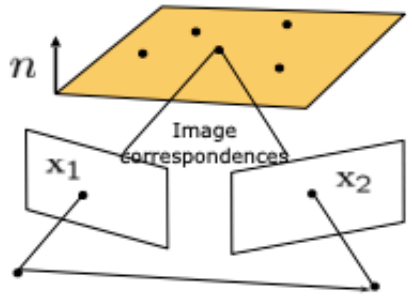
$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} fr_{11} & fr_{12} & ft_x \\ fr_{21} & fr_{22} & ft_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

holds for any instrinc matrix K

Two-views of a plane



$$\lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = H_1 \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\lambda_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H_2 \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = H_2 H_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

[LHS and RHS are related by a scale factor]

[Aside: H usually invertible]

Revisiting Homographies

http://16720.courses.cs.cmu.edu/lec/transformations.pdf

Given (x_1,y_1) and H, how do we compute (x_2,y_2)?

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$x_2 = \frac{\lambda x_2}{\lambda} = \frac{ax_1 + by_1 + c}{gx_1 + hy_1 + i}$$

$$x_2(gx_1 + hy_1 + i) = ax_1 + by_1 + c$$

reformat to:

$$AH(:) = \begin{bmatrix} 0 \\ 0 \\ \vdots \end{bmatrix}$$

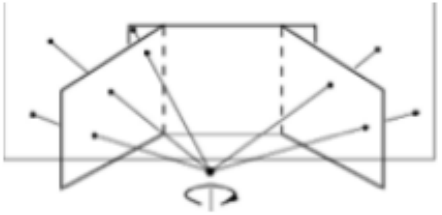
Homogenous linear system

H is determined only up to scale factor (8 DOFs)
Need 4 points minimum.
To handle more points, use RANSAC

$$\min_{||H(:)||^2=1} ||AH(:)||^2$$

Minimum right singular vector of A (eigenvector of ATA)

Special case of 2 views: rotations about camera center



$$\begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix} = R \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \end{bmatrix}$$

K_2

$$\lambda_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \end{bmatrix}$$

$$\cdots$$

$$\lambda \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 R K_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

No translation.
The homography is solved similarly though.

Camera, Image, and World Coordinate Transformations

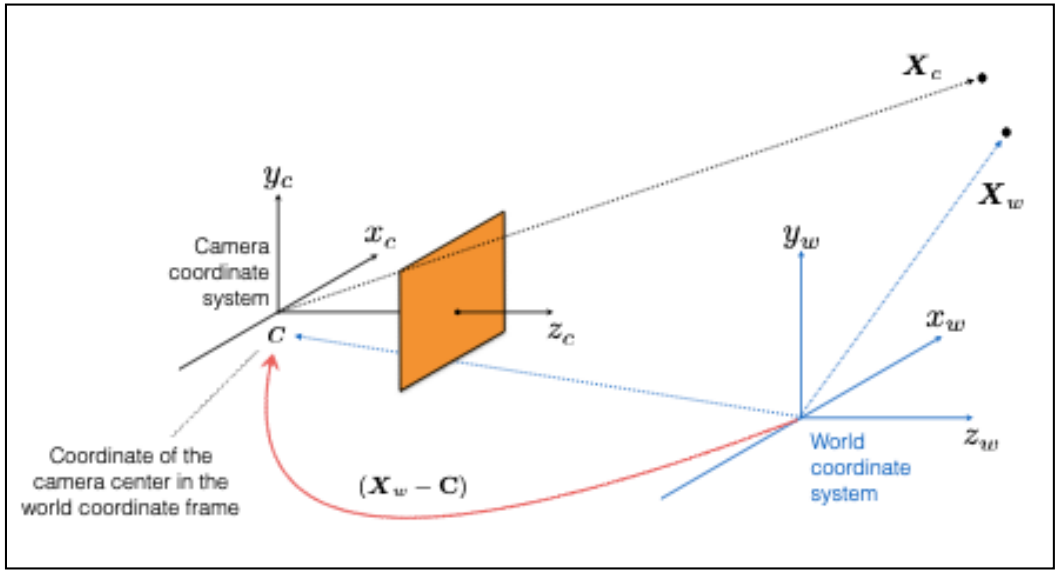
http://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf

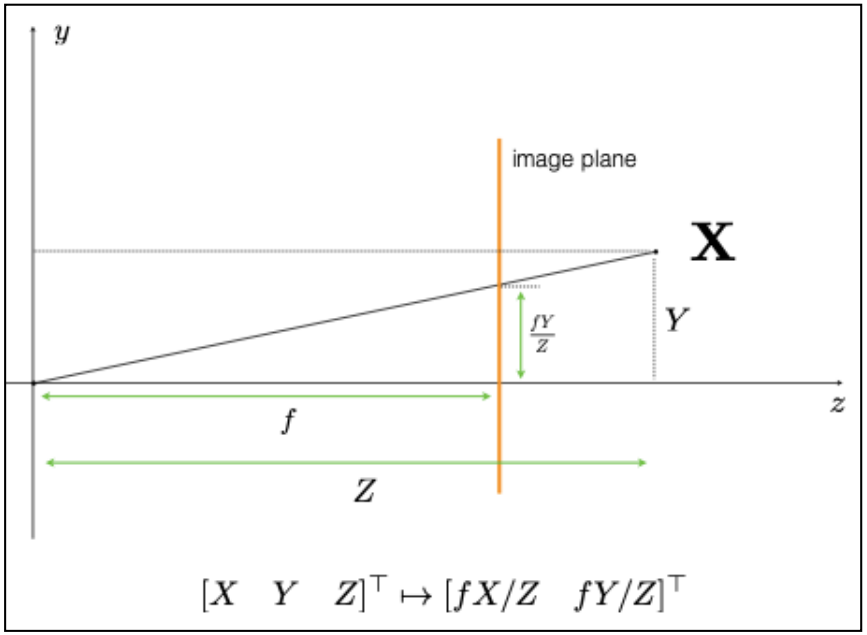
A camera is a mapping between the 3D world and a 2D image

$$\boldsymbol{x} = \mathbf{PX}$$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image 3 x 1 | Camera matrix 3 x 4 | homogeneous world point 4 x 1





Camera, Image, and World Coordinate Transformations

http://www.cs.cmu.edu/~16385/s17/Slides/11.1_Camera_matrix.pdf

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

homogeneous image 3 x 1 | Camera matrix 3 x 4 | homogeneous world point 4 x 1

$$\mathbf{P} = \mathbf{KR}[\mathbf{I}|-\mathbf{C}]$$

3x3 intrinsics | 3x3 3D rotation | 3x3 identity | 3x1 3D translation

Another way to write the mapping

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

where

$$\mathbf{t} = -\mathbf{RC}$$

(rotate first then translate)

$$\mathbf{P} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{ccc|c} r_1 & r_2 & r_3 & t_1 \\ r_4 & r_5 & r_6 & t_2 \\ r_7 & r_8 & r_9 & t_3 \end{array}\right]$$

intrinsic parameters | extrinsic parameters

$$\mathbf{R} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}$$

3D rotation | 3D translation

Points and Epipolar geometry:Essential Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf

Essential Matrix vs Homography

What's the difference between the essential matrix and a homography?

They are both 3 x 3 matrices but …

$$\boldsymbol{l}' = \mathbf{E}\boldsymbol{x}$$

Essential matrix maps a **point** to a **line**

$$\boldsymbol{x}' = \mathbf{H}\boldsymbol{x}$$

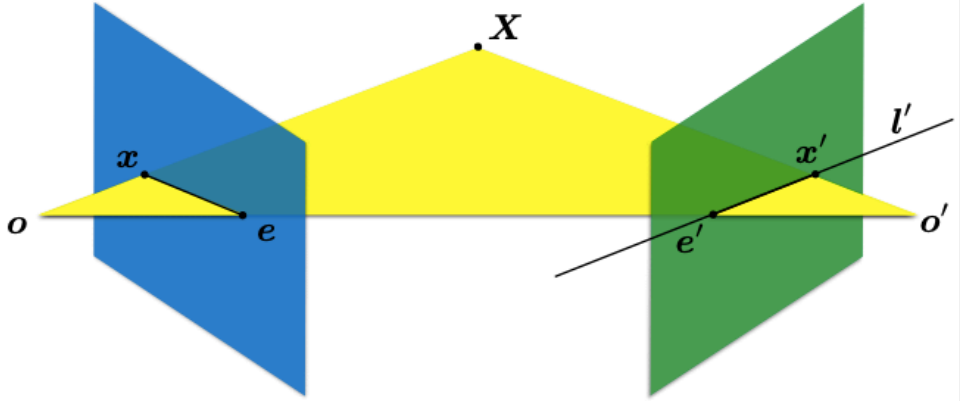
Homography maps a **point** to a **point**

Points and Epipolar geometry:Essential Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf

Given a point in one image,
multiplying by the **essential matrix** will tell us
the **epipolar line** in the second view.

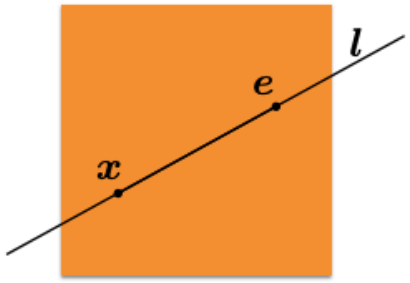
$$\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$$



NOTE: o to o' is the baseline.

Epipolar Line

$ax + by + c = 0$ in vector form $\boldsymbol{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$

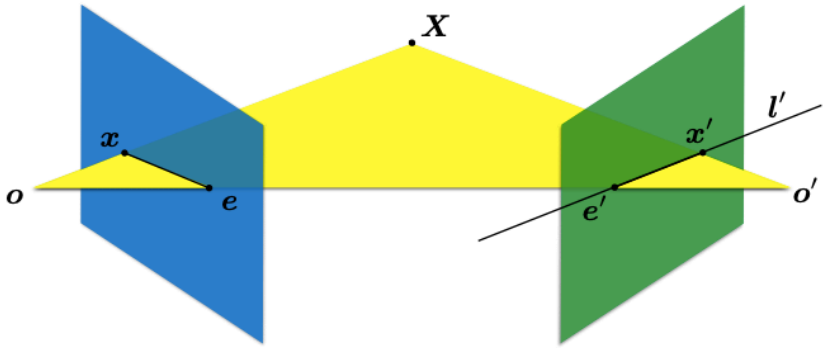


If the point $\boldsymbol{x}$ is on the epipolar line $\boldsymbol{l}$ then

$$\boldsymbol{x}^\top \boldsymbol{l} = 0$$

So if $\boldsymbol{x}^\top \boldsymbol{l} = 0$ and $\mathbf{E}\boldsymbol{x} = \boldsymbol{l}'$ then

$$\boldsymbol{x}'^\top \mathbf{E}\boldsymbol{x} = 0$$



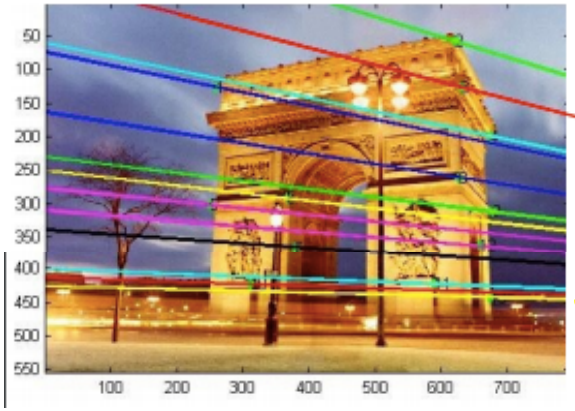
https://www.robots.ox.ac.uk/~vgg/hzbook/hzbook1/
HZepipolar.pdf
e' is the **left null-space of F**.
Similarly Fe = 0, i.e.
e is the **right null-space of F**.

coords of epipole e' w.r.t. left image coords is where the right camera is w.r.t. left image coords=
svd(F^T*F).u[2]/svd(F^T*F).u[2][2]

Points and Epipolar geometry:Essential Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.4_8Point_Algorithm.pdf

coords of right camera center in left image coords is epipole e =
svd(F^T * F).u[2]/svd(F^T * F).u[2][2]. also found as svd(F).v[2]/svd(F).v[2][2]



epipole e

```
>> [u,d] = eigs(F' * F)
```

eigenvectors

```
u =
    -0.0013    0.2586   -0.9660
     0.0029   -0.9660   -0.2586
     1.0000    0.0032   -0.0005
```

eigenvalue

```
d = 1.0e8*
   -1.0000         0         0
         0   -0.0000         0
         0         0   -0.0000
```

Eigenvector associated with smallest eigenvalue

```
>> uu = u(:,3)
(  -0.9660    -0.2586    -0.0005)
```

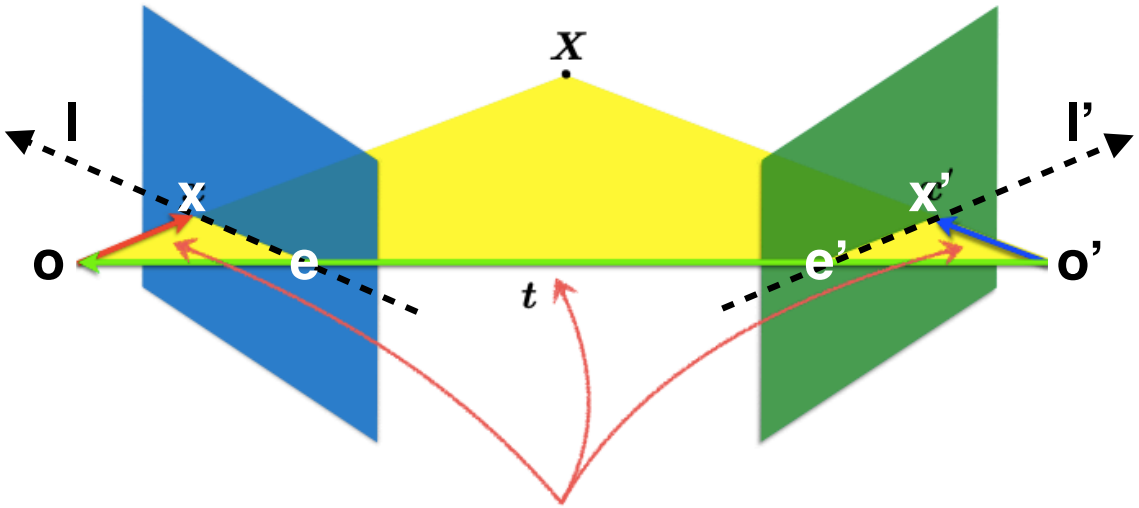
Epipole projected to image coordinates

```
>> uu / uu(3)
(1861.02     498.21     1.0)
```

NOTE:
SVD(A).U == SVD(A^T).V == SVD(AA^T).U == SVD(AA^T).V
SVD(A).V == SVD(A^T).U == SVD(A^TA).V == **SVD(A^TA).U**
A is mXn. SVD(A).V is nXn. SVD(A).U is mXm

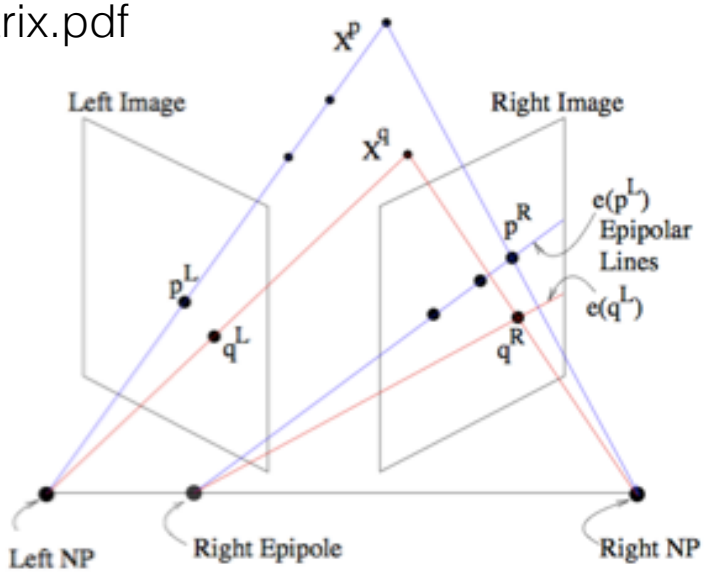
Points and Epipolar geometry:Essential Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.2_Essential_Matrix.pdf



these three vectors are coplanar $\boldsymbol{x}, \boldsymbol{t}, \boldsymbol{x}'$ so

$\boldsymbol{x}^\top(\boldsymbol{t} \times \boldsymbol{x}) = 0$ also $(\boldsymbol{x} - \boldsymbol{t})^\top(\boldsymbol{t} \times \boldsymbol{x}) = 0$



http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf

rigid motion	coplanarity
$\boldsymbol{x}' = \mathbf{R}(\boldsymbol{x} - \boldsymbol{t})$	$(\boldsymbol{x} - \boldsymbol{t})^\top(\boldsymbol{t} \times \boldsymbol{x}) = 0$

$$(\boldsymbol{x}'^\top \mathbf{R})(\boldsymbol{t} \times \boldsymbol{x}) = 0$$

$$(\boldsymbol{x}'^\top \mathbf{R})([\mathbf{t}_\times]\boldsymbol{x}) = 0$$

$$\boldsymbol{x}'^\top(\mathbf{R}[\mathbf{t}_\times])\boldsymbol{x} = 0$$

$$\boxed{\boldsymbol{x}'^\top \mathbf{E}\boldsymbol{x} = 0}$$

Essential Matrix
[Longuet-Higgins 1981]

Longuet-Higgins equation	$\boldsymbol{x}'^\top \mathbf{E}\boldsymbol{x} = 0$	
Epipolar lines	$\boldsymbol{x}^\top \boldsymbol{l} = 0$	$\boldsymbol{x}'^\top \boldsymbol{l}' = 0$
	$\boldsymbol{l}' = \mathbf{E}\boldsymbol{x}$	$\boldsymbol{l} = \mathbf{E}^T \boldsymbol{x}'$
Epipoles	$\boldsymbol{e}'^\top \mathbf{E} = \mathbf{0}$	$\mathbf{E}\boldsymbol{e} = \mathbf{0}$

(points are in normalized camera coordinates)

e is a vector in the right null space of E (singular value=0)

e' is a vector in the left null space of E (singular value=0)

Points and Epipolar geometry:Fundamental Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.3_Fundamental_Matrix.pdf

$$\hat{x}'^{\top} \mathbf{E} \hat{x} = 0$$

The Essential matrix operates on image points expressed in **normalized coordinates**
(points have been aligned (normalized) to camera coordinates)

$$\hat{x}' = \mathbf{K}^{-1} x' \qquad \hat{x} = \mathbf{K}^{-1} x$$

camera point — image point

Writing out the epipolar constraint in terms of image coordinates

$$x'^{\top} \mathbf{K}'^{-\top} \mathbf{E} \mathbf{K}^{-1} x = 0$$

$$x'^{\top} (\mathbf{K}'^{-\top} [\mathbf{t}_{\times}] \mathbf{R} \mathbf{K}^{-1}) x = 0$$

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

properties of the F matrix

Longuet-Higgins equation $x'^{\top} \mathbf{F} x = 0$

Epipolar lines $x^{\top} l = 0$ $\quad x'^{\top} l' = 0$

$l' = \mathbf{F} x$ $\quad l = \mathbf{F}^{T} x'$

Epipoles $e'^{\top} \mathbf{F} = 0$ $\quad \mathbf{F} e = 0$

(points are in image coordinates)

Points and Epipolar geometry:Fundamental Matrix

http://www.cs.cmu.edu/~16385/s17/Slides/12.4_8Point_Algorithm.pdf

$$\begin{bmatrix} x'_m & y'_m & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_m \\ y_m \\ 1 \end{bmatrix} = 0$$

Set up a homogeneous linear system with 9 unknowns

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_Mx'_M & x_My'_M & x_M & y_Mx'_M & y_My'_M & y_M & x'_M & y'_M & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

Faugeras 1992

In the case where at least eight point correspondences have been obtained between two images of an uncalibrated stereo rig, if we arbitrarily choose five of those correspondences and consider that they are the images of five points in general positions (i.e not four of them are coplanar), then it is possible to reconstruct the other three points and any other point arising from a correspondence between the two images in the projective coordinate system defined by the five points. This reconstruction is uniquely defined up to an unknown projective transformation of the environment.

In the case where at least eight point correspondences have been obtained between two images of an uncalibrated stereo rig, if we arbitrarily choose four of these correspondences and consider that they are the images of four points in general positions (i.e not coplanar), then it is possible to reconstruct the other four points and any other point arising from a correspondence between the two images in the affine coordinate system defined by the four points. This reconstruction is uniquely defined up to an unknown affine transformation of the environment.

Torr & Murray, 1997, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix"

Consider the movement of a set of point image projections from an object which undergoes a rotation and non-zero translation between views. After the motion, the set of homogeneous image points $\{\underline{\mathbf{x}}_i\}, i = 1, \ldots n$, as viewed in the first image is transformed to the set $\{\underline{\mathbf{x}}_i'\}$ in the second image, positions related by

$$\underline{\mathbf{x}}_i'^{\top} \mathbf{F} \underline{\mathbf{x}}_i = 0 \qquad (1)$$

where $\underline{\mathbf{x}} = (x, y, \zeta)^{\top}$ is a homogeneous image coordinate and

$$\mathbf{F} = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix}$$

is the fundamental Matrix (Faugeras 1992). Although 3×3, the matrix has only seven degrees of freedom because only the ratio of parameters is significant and because $\det \mathbf{F} = 0$. Throughout, underlining a symbol $\underline{x}$ indicates the perfect or noise-free quantity, distinguishing it from $x = \underline{x} + \Delta x$, the value corrupted by noise (assumed Gaussian).

Statistics of matching correspondence: Fundamental Matrix

from “Robust Parameter Estimation in Computer Vision”, 1999 Stewart
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.3914&rep=rep1&type=pdf
from “Computer Vision: Algorithms and Applications” by Szeliski
(Note, please consult author and book for any of this. use it’s presented here for educational purposes only.)

see equations (1) and (2) of Zhang, Z. 1996, RR-2927, INRIA, ffinria-00073771
”Determining the Epipolar Geometry and its Uncertainty: A Review".
And:
https://courses.cs.washington.edu/courses/csep576/11sp/pdf/SfM.pdf

Essential matrix

Co-planarity constraint:

$$x' \approx R x + t$$
$$[t]_\times x' \approx [t]_\times R x$$
$$x'^T [t]_\times x' \approx x'^T [t]_\times R x$$
$$x'^T E x = 0 \text{ with } E = [t]_\times R$$

- Solve for E using least squares (SVD)
- t is the least singular vector of E
- R obtained from the other two sing. vectors

CSE 576, Spring 2008 Structure from Motion 44

Fundamental matrix

Camera calibrations are unknown

$x' F x = 0$ with $F = [e]_\times H = K'[t]_\times R K^{-1}$

- Solve for F using least squares (SVD)
  - re-scale (x_i, x_i') so that $|x_i| \approx 1/2$ [Hartley]
- e (epipole) is *still* the least singular vector of F
- H obtained from the other two s.v.s
- “plane + parallax” (projective) reconstruction
- use self-calibration to determine K [Pollefeys]

CSE 576, Spring 2008 Structure from Motion 45

F has 7 degrees of freedom so can be estimated from at least 7 correspondence pairs.
For any point, $\tilde{x}'$, in image 2, $F\tilde{x}'$ defines a line in image 1 through which the point corresponding to $\tilde{x}'$ must pass. Similarly, for any point, $\tilde{x}$, in image 1, $F^T \tilde{x}$ defines a line in image 2 through which the point corresponding to $\tilde{x}$ must pass. All such lines, known as epipolar lines, pass through the epipoles, which are the null space of F^T and F, respectively.

Points and Epipolar geometry:Fundamental Matrix

from Strang's "Introduction to Linear Algebra",
Subspaces of matrix A which is mxn:

- row space:
  - is denoted C(A^T) and is a subspace of R^n.
  - all A^T*y.
  - dimension is nxr.
- column space:
  - is denoted C(A) and is a subspace of R^m.
  - all A*x. (the system A*x=b is solvable iff b is in column space).
  - dimension is mxr.
- Null space:
  - denoted as N(A) and is a subspace of R^n. a.k.a. the kernel of A.
  - Ax=0.
  - is orthogonal complement to ROW SPACE of A.
  - dimension is nx(n-r).
- left null space:
  - denoted as N(A^T) and is a subspace of R^m. a.k.a. as the kernel of A^T.
  - all A^T*y = 0. all column vectors x such that x^T*A = 0^T.
  - is the orthogonal complement to COLUMN SPACE of A.
  - dimension is mx(m-r).

Ax = λx, where λ is an eigenvalue of A.
λ tells whether the special vector x is stretched or shrunk or reversed or unchanged when it is multiplied by A.
nullspace entries: for eigenvalue lambda = 0 , P*x=0*x
Singular matrices have λ = 0 (and their determinants are 0).

A*f = 0 (in terms of the subspaces of matrix A, we determine the null space by the A's row space transposed times x = 0 to find what's orthogonal to the rows)

f orthogonal to A and eigenvalue=0 is smallest eigenvector of A which can be found with
SVD(A).V or SVD(A^TA).V or SVD(A).U or SVD(AA^T).U then dimension reduction to rank=2 for F where A = U * S * V^T

NOTE:
SVD(A).U == SVD(A^T).V == SVD(AA^T).U == SVD(AA^T).V
SVD(A).V == SVD(A^T).U == SVD(A^TA).V == SVD(A^TA).U
A is mXn. SVD(A).V is nXn. SVD(A).U is mXm

Points and Epipolar geometry:Fundamental Matrix and ***smallest eigenvalue***

from Wirtz and Guhr 2014,
"Distribution of the Smallest Eigenvalue in Complex and Real Correlated Wishart Ensembles"
https://arxiv.org/pdf/1310.2467.pdf

In linear discriminant analysis it ***[smallest eigenvalue]*** gives the leading contribution for the threshold estimate [21]. It is most sensitive to noise in the data [18]. In linear principal component analysis, ***the smallest eigenvalue determines the plane of closest fit [18]***. It is also crucial for the identification of single statistical outliers [17]. In numerical studies involving large random matrices, the condition number is used, which depends on the smallest eigenvalue [22, 23]. In wireless communication the Multi–Input–Multi– Output (MIMO) channel matrix of an antenna system is modeled by a random matrix [24]. The smallest eigenvalue of C yields an estimate for the error of a received signal [25, 26, 27]. In finance, the optimal portfolio is associated with the eigenvector to the smallest eigenvalue of the covariance matrix, which is directly related to the correlation matrix [28]. This incomplete list of examples shows the influence of the smallest eigenvalue in applications. ***Further information on the role of the smallest eigenvalue is given in Appendix A***.

[17] Barnett V, Lewis T. Outliers in Statistical Data. first edition ed. John Wiley & Sons; 1980.
[18] Gnanadesikan R. Methods for Statistical Data Analysis of Multivariate Oberservations. Second edition ed. John Wiley & Sons; 1997.

Points and Epipolar geometry:Fundamental Matrix: evaluating best fit

Torr & Murray, 1997, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix"

https://www.robots.ox.ac.uk/ActiveVision/Publications/torr_murray_ijcv1997/torr_murray_ijcv1997.pdf

2.2. *Orthogonal Least Squares Regression: Method OR*

Ignoring for the moment the problem of enforcing the rank 2 constraint, given $n \geq 8$ correspondences this system appears an archetype for solution by linear least squares regression. Linear here refers to linearity in the parameters f_i — equation (1) written out is just

$$f_1 \underline{x}'_i \underline{x}_i + f_2 \underline{x}'_i \underline{y}_i + f_3 \underline{x}'_i \zeta + f_4 \underline{y}'_i \underline{x}_i + f_5 \underline{y}'_i \underline{y}_i + f_6 \underline{y}'_i \zeta + f_7 \underline{x}_i \zeta + f_8 \underline{y}_i \zeta + f_9 \zeta^2 = 0 \ .$$

Because errors exist in all the measured coordinates x, y, x', y', orthogonal least squares (Pearson 1901) rather than ordinary least squares should be used, minimizing the sum of the squares of the distances shown in part (a) rather than (b) of Figure 2.

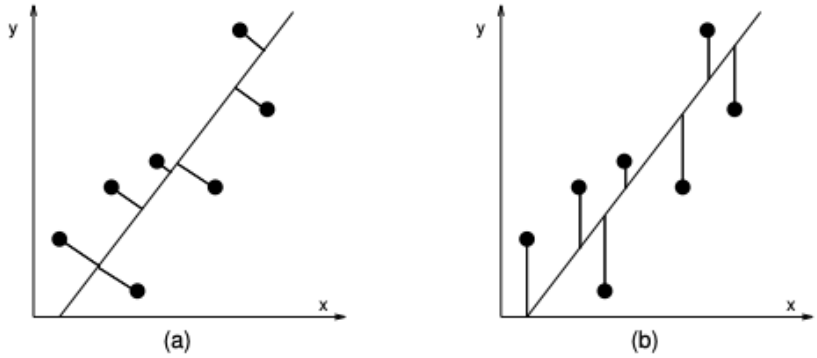


Fig. 2. The (a) orthogonal and (b) ordinary least squares distances.

Consider fitting a hyperplane $\mathbf{f} = (f_1, f_2, \ldots, f_p)$ through a set of n points in $\mathcal{R}^p$ with coordinates $\mathbf{z}_i = (z_{i_1}, z_{i_2}, \ldots, z_{i_p})$, taking the centroid of the data as origin. (Centring is a standard statistical technique that involves shifting the coordinate system of the data points so that the centroid lies at the origin. The best fitting hyperplane passes through the centroid of the data (Pearson 1901).) Assuming that the noise is Gaussian and that the elements of $\mathbf{z}$ have equal variance[1], the hyperplane $\mathbf{f}$ with maximum likelihood is estimated by minimizing the perpendicular sum of Euclidean distances from the points to the plane (Pearson 1901; Kendall & Stuart 1983)

$$\min_{\mathbf{f}} \sum_{i=1}^{n} (\mathbf{f}^\top \mathbf{z}_i)^2$$

. . .

Method S1.

The orthogonal least squares method (OR) will in fact produce a sub-optimal estimate of $\mathbf{F}$ because the residuals in the minimization

$$r_i = f_1 x'_i x_i + f_2 x'_i y_i + f_3 x'_i \zeta + f_4 y'_i x_i + f_5 y'_i y_i + f_6 y'_i \zeta + f_7 x_i \zeta + f_8 y_i \zeta + f_9 \zeta^2 \qquad (2)$$

are not Gaussianly distributed. The cause of this is

Points and Epipolar geometry:Fundamental Matrix: evaluating best fit

Torr & Murray, 1997, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix"

https://www.robots.ox.ac.uk/ActiveVision/Publications/torr_murray_ijcv1997/torr_murray_ijcv1997.pdf

Method S2. Luong and Faugeras (1993)

the epipolar line $\mathbf{F}\underline{\mathbf{x}}$ in image two. Noisy measurements will however not lie on their associated epipolar lines exactly. The perpendicular distance of a point $\mathbf{x}$ to the predicted epipolar line $\mathbf{x}'^{\top}\mathbf{F}$ in the first image is

$$e_1 = \frac{xr_x + yr_y + r_\zeta}{(r_x^2 + r_y^2)^{1/2}} = \frac{r}{(r_x^2 + r_y^2)^{1/2}}.$$

and the distance of a point $\mathbf{x}'$ to the epipolar line $\mathbf{F}\mathbf{x}$ in the second image is

$$e_2 = \frac{r}{(r_{x'}^2 + r_{y'}^2)^{1/2}}.$$

. . .

its epipolar line is used. This ensures that each image receives equal consideration. The distance $e = (e_1^2 + e_2^2)^{1/2}$ is referred to as the epipolar distance. It is assumed that the errors in the measured location of each point are Gaussian with variance σ^2. If the co-

w_E, so that $e = rw_E$. Effectively then the epipolar weighting is

$$w_E = \left(\frac{1}{r_x^2 + r_y^2} + \frac{1}{r_{x'}^2 + r_{y'}^2}\right)^{1/2},$$

which is not dissimilar to the Sampson weighting

$$w_S = \left(\frac{1}{r_x^2 + r_y^2 + r_{x'}^2 + r_{y'}^2}\right)^{1/2}.$$

In summary, three least squares methods with different error terms have been discussed:

1. Method OR uses the sum of squares of *algebraic distances*:
$$R^2 = \sum r_i^2, \text{ where } r_i = \mathbf{x}_i'^{\top}\mathbf{F}\mathbf{x}_i.$$
2. Method S1 uses the sum of squares of the Sampson distances:
$$D^2 = \sum (w_{Si} r_i)^2 = \sum (r_i / \nabla r_i)^2 = \sum d_i^2.$$
3. Method S2 uses the sum of squares of the epipolar distances:
$$E^2 = \sum (w_{Ei} r_i)^2 = \sum (e_{1i}^2 + e_{2i}^2) = \sum e_i^2.$$

In terms of probability distributions, the first corresponds to something intractable, the second is a first order approximation to a χ^2 distribution, and the third is a χ^2 distribution.

Statistics of matching correspondence

from “Robust Parameter Estimation in Computer Vision”, 1999 Stewart
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.3914&rep=rep1&type=pdf

Robust Estimation

breakdown point: the minimum fraction of outlying data that can cause an estimate to diverge arbitrarily far from the true estimate. (e.g. for least squares, 1 outlier can harm the estimate, so the breakdown point is 0). a breakdown point of 0.5 is traditionally used.

influence function*:* the change in an estimate caused by insertion of outlying data as a function of the distance of the data from the (uncorrupted) estimate. (e.g. for least squares, the influence function is simply proportional to the distance of the point from the estimate.) influence function should approach 0 with increasing distance.

statistical efficiency: ratio of minimum possible variance to the actual variance of an estimate. the minimum possible variance is determined by a distribution such as a gaussian. the upper bound of efficiency is then 1.

Linear Regression or estimation of univariate or multivariate location and scatter are used.

Let **X** = {**x_i**} be a set of data points (vectors)

Let **a** be a k-dimensional parameter vector to be estimated.

r_i_a = r(**x_i, a**) is the objective function is defined in terms of an error distance or residual function, a true geometric distance is ideal. euclidean, mahalanobis of the covariance matrix, etc.

(Mahalanobis distance is a measure of the distance between a point and a distribution in terms of standard deviations.)

M-estimators and least-median of squares

Statistics of matching correspondence

from “Robust Parameter Estimation in Computer Vision”, 1999 Stewart
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.3914&rep=rep1&type=pdf

M-estimators: generalizations of maximum likelihood and least squares.

$\boldsymbol{\sigma}_i^2$ = variance (scale) assoc. w/ scalar r_i_a

ρ(u) = a robust loss function

â is the M-estimator of **a**.

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{\mathbf{x}_i \in X} \rho(r_{i,\mathbf{a}}/\sigma_i)$$

ψ(u) = ***ρ’(u)*** is essentially proportional to the influence function.

w(u) · u = ψ(u) is a weight function.

iteratively re-weighted least-squares (IRLS) alternates between calculating weights $w_i = w(r\_i\_a/\boldsymbol{\sigma}_i)$ using the current estimate of **a** then solving following eqn. to estimate **a** with fixed weights**.**

$$\sum_{\mathbf{x}_i \in X} \psi(r_{i,\mathbf{a}}/\sigma_i) \frac{dr_{i,\mathbf{a}}}{d\mathbf{a}} \frac{1}{\sigma_i} = 0.$$

Beaton and Tukey [4]	$\rho(u) = \begin{cases} \frac{a^2}{6}\left[1-\left(1-\left(\frac{u}{a}\right)^2\right)^3\right] & \lvert u\rvert \le a \\ \frac{a^2}{6} & \lvert u\rvert > a \end{cases}$	$\psi(u) = \begin{cases} u\left[1-\left(\frac{u}{a}\right)^2\right]^2 & \lvert u\rvert \le a \\ 0 & \lvert u\rvert > a \end{cases}$
Cauchy [32]	$\rho(u) = \frac{b^2}{2}\log\left[1+\left(\frac{u}{b}\right)^2\right]$	$\psi(u) = \frac{u}{1+(u/b)^2}$
Huber [34, Chapter 7]	$\rho(u) = \begin{cases} \frac{1}{2}u^2 & \lvert u\rvert \le c \\ \frac{1}{2}c(2\lvert u\rvert - c) & c < \lvert u\rvert \end{cases}$	$\psi(u) = \begin{cases} 1 & \lvert u\rvert \le c \\ c\frac{u}{\lvert u\rvert} & \lvert u\rvert > c \end{cases}$

Table 1: Three different robust loss functions, $\rho(u)$, and associated ψ functions. The “tuning parameters” a, b and c are often tuned to obtain 95% efficiency [32].

Statistics of matching correspondence

from “Robust Parameter Estimation in Computer Vision”, 1999 Stewart
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.3914&rep=rep1&type=pdf

Least-Median of Squares: has a breakdown point of 0.5.
â is the LMS estimate of **a**. $\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \underset{\mathbf{x}_i \in X}{\operatorname{median}} r_{i,\mathbf{a}}^2.$

because the median is not differentiable, search techniques are needed.
e.g. for regression lines, can computationally solve for median exactly and efficiently.

a random sampling technique: randomly select a number of *k*-point subsets of the *N* data points. A parameter vector, $\mathbf{a_s}$, is fit to the points in each subset, *s*. Each $\mathbf{a_s}$ is tested as a hypothesized fit by calculating the squared residual distance $(r_{i\mathbf{a_s}})^2$ of each of the *N-k* points in X-s and finding the median. The $\mathbf{a_s}$ corresponding to the smallest median over S subsets is chosen as the estimate, **â**. Overall, this computation requires O(S N) time using linear-time median finding techniques.

Determining the number of subsets **S** is important. S needs to be large enough to have high probability of containing al “good” points, that is, no outliers.
p is the minimum fraction of good points.
p^k is the probability that a set contains all good points.

$$P_g = 1 - (1 - p^k)^S.$$

choose P_g and solve for S:

P_g	k	p	S
0.99	3	0.5	35
0.99	6	0.6	97
0.99	6	0.5	293

Statistics of matching correspondence

from “Robust Parameter Estimation in Computer Vision”, 1999 Stewart
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.7.3914&rep=rep1&type=pdf

Least-Median of Squares (cont.):

â can be used to refine the fit.

s* is the subset used to form **â**.

ϕ^{-1} is the inverse of the cumulative normal distribution.

$$\hat{\sigma} = 1/\Phi^{-1}(0.75)\left(1 + \frac{5}{N-k}\right)\sqrt{\underset{\mathbf{x}_i \in X - s^*}{\text{median}}\, r^2_{i,\hat{\mathbf{a}}}},$$

if all N points are sampled from a normal distribution with variance σ^2 , then $\hat{\sigma} \to \sigma$ as $N \to \infty$.
can choose data points such that $(r_{ias})^2 < (\theta\, \hat{\sigma})^2$ where θ is constant, typically about 2.5 then re-estimate **â**.

Random Sample Consensus (RANSAC):

a minimal subset random sampling search technique, the objective function to be maximized is the number of data points (inliers) having absolute residuals smaller than a predefined value. Equivalently, this may be viewed as minimizing the number of outliers, which may then be viewed as a binary robust loss function that is 0 for small (absolute) residuals, 1 for large absolute residuals, and has a discontinuous transition at θ.

Torr & Murray, 1997, "The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix"

https://www.robots.ox.ac.uk/ActiveVision/Publications/torr_murray_ijcv1997/torr_murray_ijcv1997.pdf

Using RANSAC w/ 8-point or 7-point methods:

RANSAC uses iterations over random subsets of size 7 or 8 of the total point set (which is usually over-determined, having more than 8 points), solves for the fundamental matrix (F), evaluates the fit using epipolar projections of all points, and keeps the best F.

The number of iterations is pre-calculated for the probability that all points are not outliers. Then upon each improved F in the iterations, the number of iterations is readjusted downwards.

Table 3. The number m of subsamples required to ensure $\Upsilon \geq 0.95$ for given p and ϵ, where Υ is the probability that all the data points selected in one subsample are non-outliers.

Features	Fraction of Contaminated Data, ϵ						
p	5%	10%	20%	25%	30%	40%	50%
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95
6	3	4	10	16	24	63	191
7	3	5	13	21	35	106	382
8	3	6	17	29	51	177	766

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

2 Degeneracy and the Fundamental matrix

Suppose that a camera takes two noise free images of a 3D object, and undergoes a rotation and non-zero translation between the two images. The set of image points $\{\underline{\mathbf{x}}_i\}, i = 1, \ldots n$, in the first image (we adopt the convention of signifying noise free data by underlining it) is transformed to the set $\{\underline{\mathbf{x}}_i'\}$, in the second, where $\underline{\mathbf{x}}_i$ and $\underline{\mathbf{x}}_i'$ have homogeneous coordinates, $\underline{\mathbf{x}}_i = (\underline{x}_i, \underline{y}_i, 1)^\top$, and $\underline{\mathbf{x}}_i' = (\underline{x}_i', \underline{y}_i', 1)^\top$. The two sets of image points are related by

$$\underline{\mathbf{x}}_i'^\top \mathbf{F} \underline{\mathbf{x}}_i = 0 \qquad 1 \leq i \leq n \tag{1}$$

where $\mathbf{F}$ is a 3×3 matrix of rank 2. It is known as the fundamental matrix [11, 18].

Recall that data are degenerate if the true data (the noise free correspondences) admit multiple solutions for the fundamental matrix. It can be seen that an understanding is required of cases that lead to the true data being degenerate. The next subsection analyses all such cases.

2.1 Geometric Degeneracy

A perfect (i.e. noise free and no outliers) set of correspondences: $\{\underline{\mathbf{x}}_i\} \leftrightarrow \{\underline{\mathbf{x}}_i'\}, i = 1, \ldots n$, is geometrically degenerate with respect to $\mathbf{F}$ if it fails to define a unique epipolar transform, or equivalently if there exist linearly independent rank two matrices, $\mathbf{F}_j$, $j = 1, 2$, such that

$$\underline{\mathbf{x}}_i'^\top \mathbf{F}_1 \underline{\mathbf{x}}_i = \underline{\mathbf{x}}_i'^\top \mathbf{F}_2 \underline{\mathbf{x}}_i = 0 \qquad (1 \leq i \leq n) \ . \tag{2}$$

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

for n correspondences >= 8 in general position: $\dim(N) = 1$: There is no degeneracy:

for n correspondences == 7 or for both camera optical centers and and any number of 3D points lie on a quadric surface referred to as the critical

- $\dim(N) = 2$ imposing the constraint that FM rank=2, one must save the cubic equation in alpha, and that has 1 or 3 solutions. if it has 1 solution, it is not degenerate.
- $\dim(N) = 3$ this structure can arise from an observation of a special type of structure (a plane) or from he camera undergoing a special type of motion (a rotation about its optic center)
- $\dim(N) = 4$ points are not in general position, but lie on a line

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

1. **Varieties V of dimension 3.**

   (a) *Fundamental matrix*, degree 2, DOF 7, correspondences 7, for 1 or 3 solutions, 8 for unique solution.

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0 \quad \text{where} \quad \mathbf{F} = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \tag{10}$$

   (b) *Affine fundamental matrix*, degree 1, DOF 4, correspondences 4.

$$\mathbf{x}'^{\top}\mathbf{F}_A\mathbf{x} = 0 \quad \text{where} \quad \mathbf{F}_A = \begin{bmatrix} 0 & 0 & a_1 \\ 0 & 0 & a_2 \\ a_3 & a_4 & a_5 \end{bmatrix}. \tag{11}$$

   (c) *Translation fundamental matrix*, degree 2, DOF 2, correspondences 2.

$$\mathbf{x}'^{\top}\mathbf{F}_T\mathbf{x} = 0 \quad \text{where} \quad \mathbf{F}_T = \begin{bmatrix} 0 & g_3 & -g_2 \\ -g_3 & 0 & g_1 \\ g_2 & -g_1 & 0 \end{bmatrix}. \tag{12}$$

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

2. **Varieties V of dimension 2.**

(a) *Quadratic image transformation*, $\mathbf{Q}$, degree 4, DOF 14, correspondences 7;
$\mathbf{x}' = \mathbf{F}_1\mathbf{x} \times \mathbf{F}_2\mathbf{x}$, where $\mathbf{F}_1$, $\mathbf{F}_2$ are 3×3 fundamental matrices.

(b) *Projective image transformation*, degree 4, DOF 8, correspondences 4.

$$\mathbf{x}' = \mathbf{H}\mathbf{x} \quad \text{where} \quad \mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix}. \tag{13}$$

(c) *Affine image transformation*, degree 1, DOF 6, correspondences 3.

$$\mathbf{x}' = \mathbf{K}\mathbf{x} \quad \text{where} \quad \mathbf{K} = \begin{bmatrix} k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 \\ 0 & 0 & 1 \end{bmatrix}. \tag{14}$$

(d) *Image translation*, degree 1, DOF 1, correspondence 1.

$$\mathbf{x}' = \mathbf{L}\mathbf{x} \quad \text{where} \quad \mathbf{L} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{15}$$

(e) *No Motion*, degree 1, DOF 0, correspondences 0.

Table 1: *Models that are fitted to sets of correspondences. The models (top to bottom) are given in order of decreasing generality. For example the affine fundamental matrix and the translational fundamental matrix are both special cases of the fundamental matrix.*

Points and Epipolar geometry:Fundamental Matrix: ambiguous geometries

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

7.3 Using RANSAC to Detect Degeneracy—PLUNDER-DL

number of dependent significance levels. Rather what is required is some sort of score function for each of the fitted models; and this score function must perforce be composed of some combination of the available information: the number of inliers and outliers, and the degree and dimensionality of the model. The choice of scoring function must be dictated by the application of the algorithm. In order to accomplish this within this paper we have adapted the ideas of Minimum Message Length (MML) [14] and Minimum Description Length (MDL) [38]. The general idea of these methods is to minimize the binary digits (*bits*) needed to encode a signal in order to send the signal over a communication channel and decode it on the other side of the channel. The idea that

The new PLUNDER-DL approach differs from MDL in that it does not assume knowledge of the residuals or their probability distributions, as it encodes only the deterministic and not the stochastic part of the signal (which is assumed to be small). All that is needed is the set of inliers and the set of outliers to the model. In the results section we will argue that in the vast majority of cases further sophistry is unnecessary and adds an extra computational burden for little gain.

Consider a point in $\mathcal{R}^4$, in order to encode this point 4 coordinates are needed. Let the image size be $R \times R$ window, e.g. 512×512, and the coordinates are measured up to a resolution of ϵ, e.g. the Harris corner detector [15, 17] can achieve sub-pixel accuracy of 0.1 pixels in the location of a point. If a point is located within $\mathcal{R}^4$ then the minimum encoding length for each coordinate is $\log a = \log \frac{R}{\epsilon}$. Let the model parameters storage be $\log b$. As a fundamental matrix has 7 DOF then it may be encoded by seven parameters. As the variety defined by $\mathbf{F}$ has dimension three, each inlier may be encoded by 3 coordinates. Outliers to the fundamental matrix may not be so economically described being points in $\mathcal{R}^4$, and hence require 4 coordinates to describe them. Thus if the fit to the fundamental matrix provides n_i^F inliers and n_o^F outliers the PLUNDER-DL code length is

$$\mathbf{PL}(\mathbf{F}) = (3n_i^F + 4n_o^F)\log a + 7\log b \quad . \tag{30}$$

If for a homography there are n_i^H inliers and n_o^H outliers then the corresponding PLUNDER-DL score is (recalling that the dimension of $\mathbf{H}$ is 2 and there are 8 degrees of freedom)

$$\mathbf{PL}(\mathbf{H}) = (2n_i^H + 4n_o^F)\log a + 8\log b \quad . \tag{31}$$

1. Estimate the set of inliers for each model using RANSAC, as described in Section 6.
2. Compute PLUNDER-DL scores for each model using (33).
3. Accept the model with the lowest PLUNDER-DL score.

Table 5: *A brief summary of PLUNDER-DL.*

excepting, that a threshold is used for calculating inliers

In the absence of *a priori* information we choose to store the parameters with the same amount of storage as the data, i.e. $\log b = \log a$.

$$\mathbf{PL} = (n_i \text{ dimension of model } + 4n_o) + \text{ DOF } , \tag{33}$$

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

4.1 Determining Goodness of Fit

Within this section the traditional χ^2 method of determining the goodness of fit is outlined, and its deficiencies revealed. Assuming that the image coordinates are perturbed by Gaussian noise with mean zero and known variance σ^2 (thus there are no outliers), then the distance of a perturbed point correspondence from the algebraic variety defined by the fundamental matrix also follows a Gaussian distribution. Hence the sum of squares of these distances follows a χ^2 distribution. The acceptability of a given $\mathbf{F}$ may be judged in terms of its errors. The true fundamental matrix is unknown and an estimate must perforce be used in its place. If the estimate is sufficiently close to the true fundamental matrix then the sum of squares of distances to the estimate d_V, divided by their standard deviation, has a χ^2 distribution with $d_f = n - 7$ degrees of freedom, where n is the number of image correspondences, and seven is the number of degrees of freedom of the fundamental matrix. The suitability of a given $\mathbf{F}$ may be tested under the assumption that a fit is good if it lies within the *critical region* of all solutions, defined as those with sum of squares of distances below $T_F = \chi^2_{d_f}(\alpha)$, where α is the required degree of confidence, i.e. the critical region in the space of $\mathbf{F}$ is given by

$$\{\mathbf{F}|D_V(\mathbf{F}) \leq T_F\} \tag{21}$$

typically $\alpha = 95\%$. We have given two other types of fundamental matrix that might occur in practice, the error $D_V(\mathbf{F}_T)$ has $d_f = n - 2$ degrees of freedom, $D_V(\mathbf{F}_A)$ has $d_f = n - 4$ degrees of freedom.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

Model	corr	Figure 3	Figure 5	Figure 6	Figure 7	Figure 8	Figure 9
Number of matches n		36	142	406	190	80	332
Fundamental	7	31/120	134/471	305/1326	131/645	75/252	241/1094
Translation	2	27/119	88/482	305/1322	69/693	14/308	241/1089
Affine Fundamental	4	28/120	128/444	304/1324	64/700	46/278	240/1092
Quadratic	18	31/100	131/324	270/1102	127/524	38/262	240/866
Projectivity	8	27/98	127/318	268/1096	71/626	36/256	239/858
Affinity	6	27/96	128/315	239/1152	53/600	32/262	239/856

Table 6: *For each model the number of inliers/PLUNDER-DL score (n_i/**PL**) is given. The model selected is underlined. corr refers to the minimum number of correspondences needed to instantiate a model.*

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

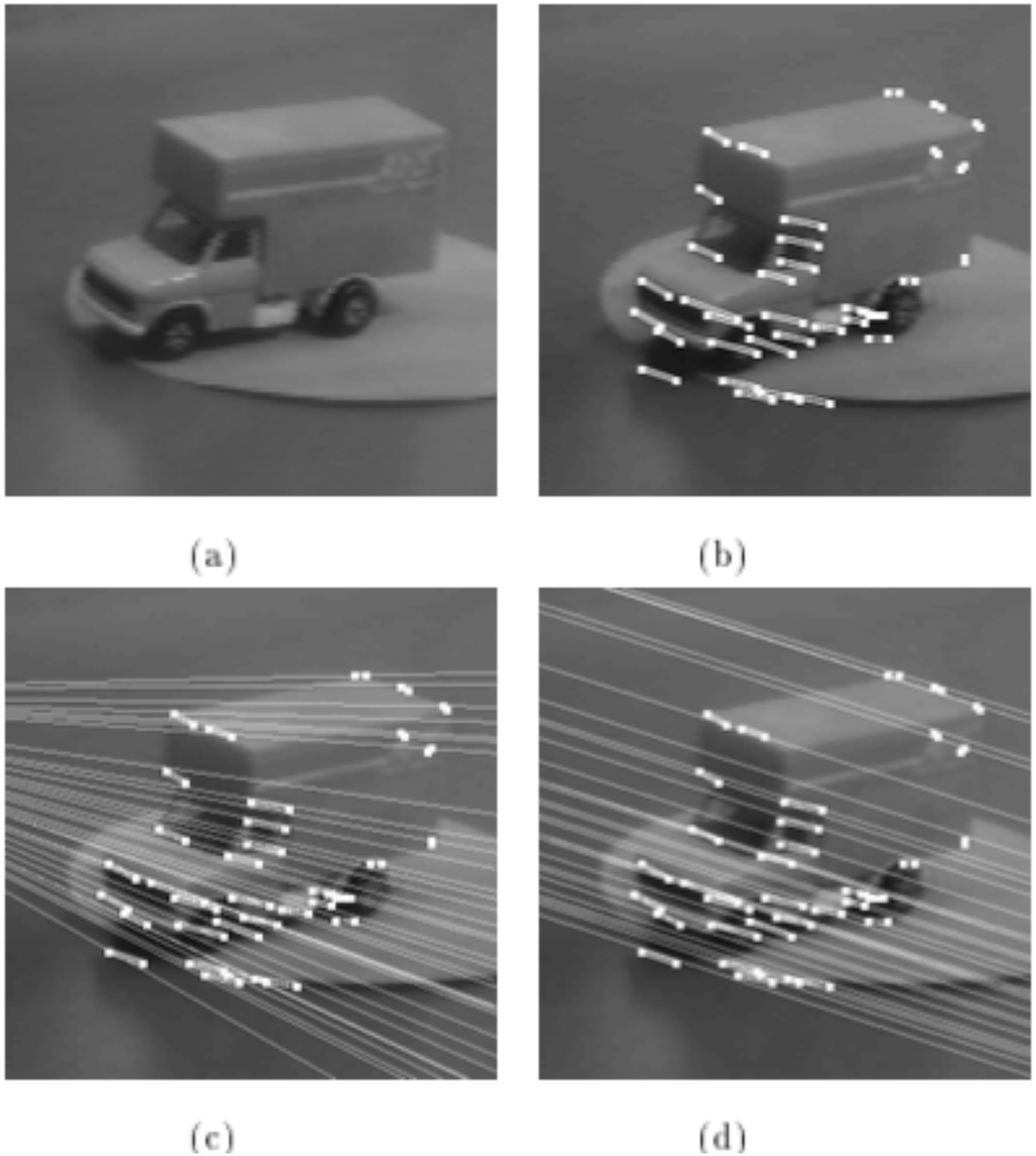


Figure 3: Rotating toy truck sequence (a) first image (b) second image with correspondences from (a) to (b) superimposed, (c) (d) Two epipolar geometries that fit the correspondences shown in Figure 3 (b) equally well. It can be seen that the data are degenerate. (d) is an affine camera epipolar geometry with parallel epipolar lines.

The model selected is an affinity, established by 3 point correspondences; 27 points are consistent with the affinity and 31 with a full fundamental matrix which has 7 - 3 = 4 more degrees of freedom. The data are thus clearly degenerate with respect to F, with most of the points lying on the plane of the truck.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

(a)

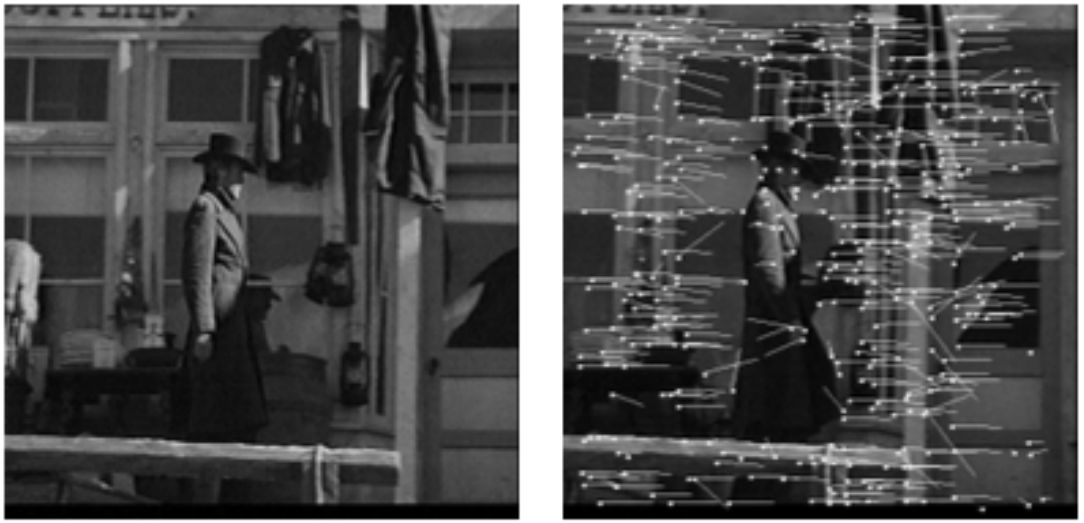
(b)

Figure 5: (a) First image and (b) second image with matches. The camera motion is composed of a large rotation of the camera about the optic axis and translation along the line of sight.

In Figure 5 the camera motion combines a translation perpendicular to the image plane with a cyclotorsion. The best fitting degenerate mo del is an affinity with 128 points, as expected.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz



(a) (b)

(c) (d)

Figure 6: Two images from the gun fighter sequence. The camera

is tracking to the right and the gun fighter raises his gun. (a) first image, (b) second image and matches, (c) inliers, and (d) outliers.

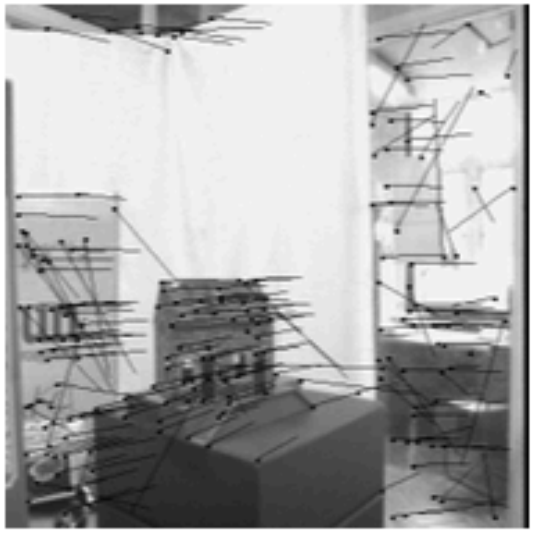
Figure 6 (a)(b) shows a mobile gun fighter viewed by a tracking camera. The gunfighter is moving to the right as the camera moves in parallel to keep him in the center of the image. Points in independent motion on the gun fighter are identified if they are inconsistent with the motion model chosen for the background motion. The data support a projectivity as most of the correspondences are on the saloon wall.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

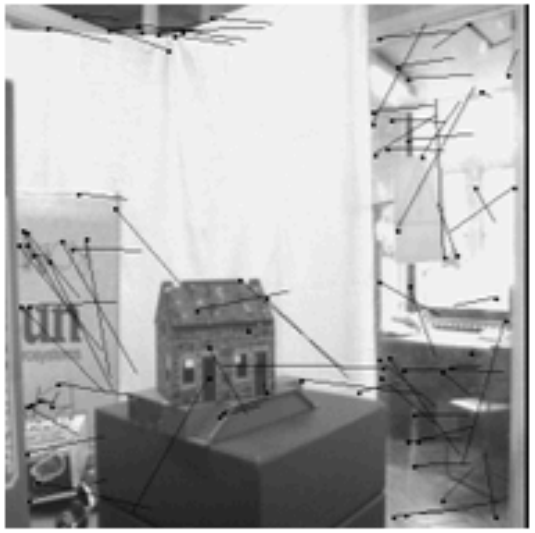
(a)

(b)



(c)

(d)



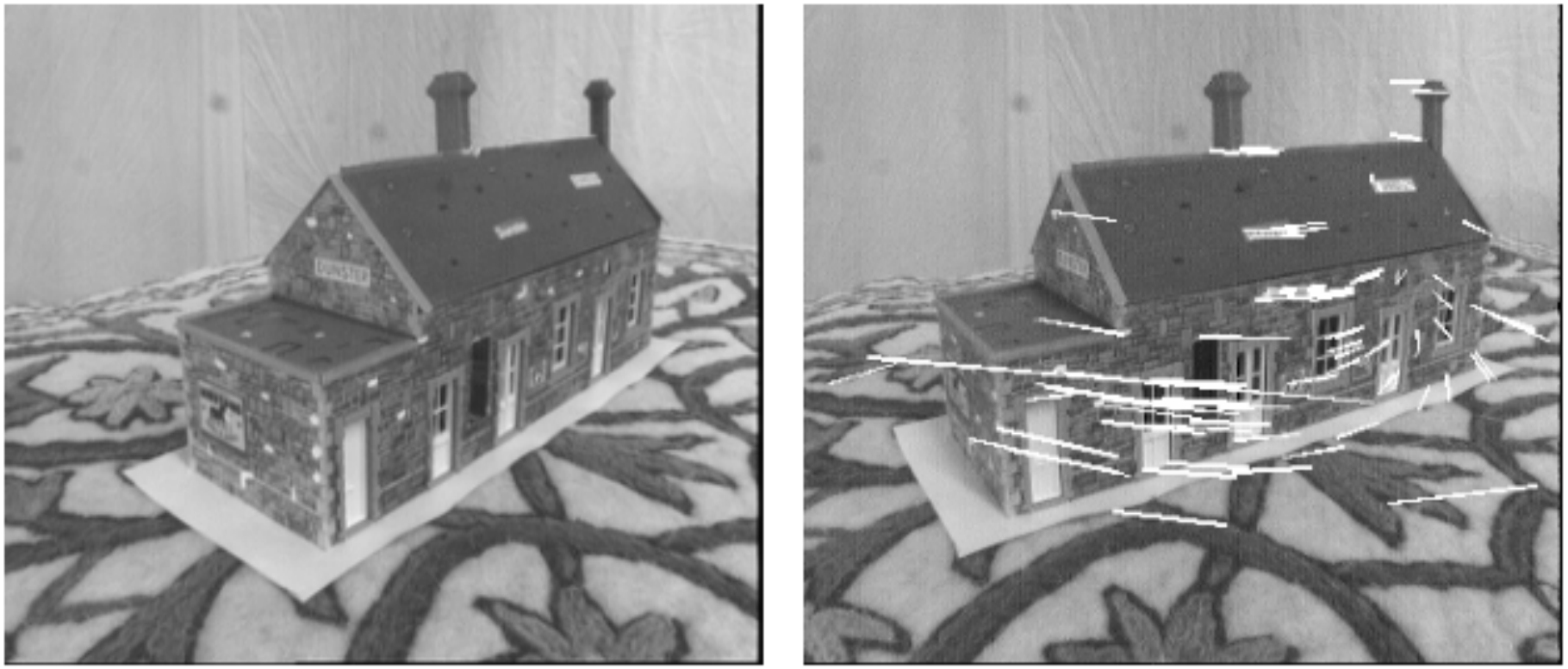
(e)

Figure 7: Indoor sequence, camera translating and rotating while fixated on the model house: (a) first image, (b) second image, (c) matches, (d) inliers, and (e) outliers. The data are found to be degenerate with respect to the fundamental matrix.

Figure 7 shows a scene in which a camera rotates and translates whilst fixating on a model house. The important thing about this image pair is that the structure recovery for this image pair proved highly unstable. The reason for this instability is not immediately apparent until the good fit of the quadratic transformation is witnessed, indicating that the structure is close to a critical surface.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz



(a) (b)

Figure 8: Indoor sequence, camera translating and rotation to fixate on the house. (a) left image, (b) right image and inliers.

The scene is correctly flagged as a general motion, as the translation and rotational components of the camera motion were both significant, and full perspective structure can be recovered.

Points and Epipolar geometry:Fundamental Matrix: ambiguous solutions

Torr, Zisserman, & Maybank 1996, "Robust Detection of Degenerate Configurations whilst Estimating the Fundamental Matrix https://www.robots.ox.ac.uk/~phst/Papers/CVIU97/m.ps.gz

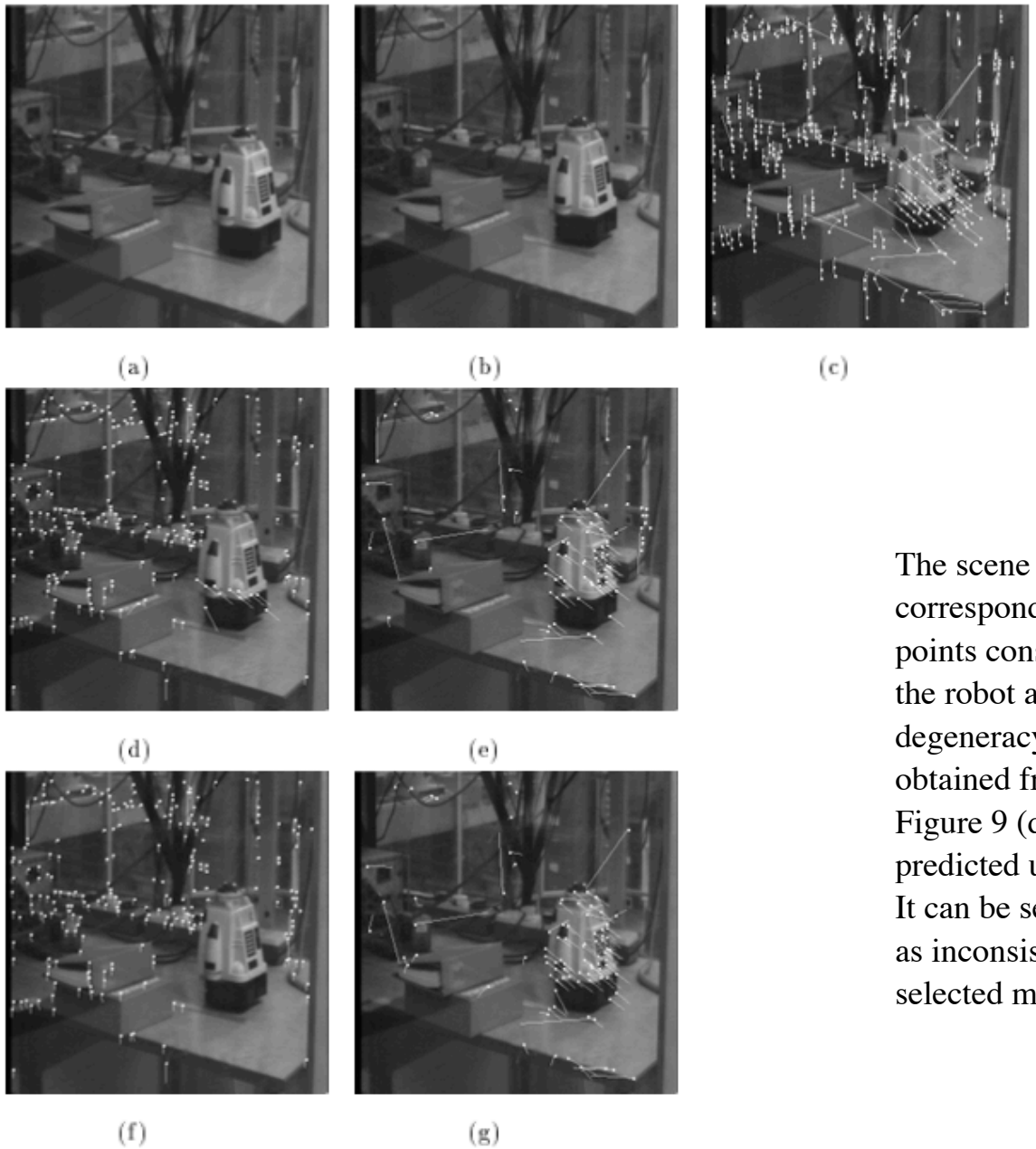


Figure 9: (a) first image, (b) second image, (c) matches of the sequence with an independently moving robot viewed from a camera, translating vertically down (parallel to the image plane). The robot moves along the table. Fundamental matrix inliers (d) and outliers (e) for F. It can be seen that the independently moving robot is not fully segmented. Plunder.dl indicated an affinity fit, with inliers (f) and outliers (g). The independently moving robot is fully segmented.

The scene contains an independently moving robot. The correspondences are shown in (c). The aim is to cluster all the points consistent with the background whilst excluding points on the robot and outliers. To demonstrate the advantage of the new degeneracy detection algorithm we compare our results with those obtained from a more conventional robust estimator, RANSAC. Figure 9 (d) shows the inliers and Figure 9 (e) shows the outliers predicted using a full fundamental matrix estimated by RANSAC. It can be seen in (d) that not all the points on the robot are flagged as inconsistent with the background. However, the PLUNDER.DL selected model (an affinity) does eliminate these matches.

Points and Epipolar geometry:Fundamental Matrix: cheirality

Wei Xu and J. Mulligan,
"Robust relative pose estimation with integrated cheirality constraint," ***2008 19th International Conference on Pattern Recognition*****, Tampa, FL, USA, 2008, pp. 1-4, doi: 10.1109/ICPR.2008.4761509.**

https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.2538&rep=rep1&type=pdf

1. Compute feature correspondences between two images taken from different viewpoints.
2. Repeat for n hypothesis-and-test trials. In each trial do the following:
    1. (a) Select a random sample of the minimum number of correspondences; use a hypothesis generator to generate a hypothesis of the epipolar geometry (F or E);
    2. (b) *Check the cheirality of the sample points*. If points disagree on cheirality discard hypothesis, goto (a). Otherwise record the cheirality of the sample/hypothesis CH.
    3. (c) Reconstruct *all the correspondences* and for each reconstructed point compute the reprojection error: e2 i = i(d(p1, pˆ1)2 + d(p2, pˆ2)2) where d is the Euclidean distance in the image between the original correspondence points (p1, p2) and the projections of the reconstructed point (ˆp1, pˆ2).
    4. (d) Count the number of points with distance ei less than some threshold T. These points are regarded as inliers consistent with the epipolar geometry hypothesis.
    5. (e) *For each inlier, we check its cheirality Ci. If Ci = CH, it's an inlier consistent with the hypothesis in both epipolar geometry and cheirality*. Record the number of cheirality-verified inliers.
3. Select the largest refined inlier set consistent with its generating hypothesis in both epipolar geometry and cheirality.
4. Use all the refined inliers to re-estimate the epipolar geometry (F or E).
5. Derive E from F (if necessary), and re-compute the relative motion [R, t] from the re-estimated E.

popular epipolar geometry framework. For generality considerations, we use Nister's approach [7] to evaluate the cheirality of the correspondences. Given an estimated E, the cheirality C_i of a correspondence has four possible configurations: $[R_a|t_u]$, $[R_a|-t_u]$, $[R_b|t_u]$ and $[R_b|-t_u]$ where $R_a = UDV^T$ and $R_b = UD^TV^T$; $[U, \Sigma, V] = \text{svd}(E)$ with $det(U) > 0$ and $det(V) > 0$; D is a predefined matrix and t_u is the third column of U.

[7] D. Nister. An efficient solution to the five-point relative pose problem. In Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'03), volume 2, pages 195–202, June 2003

Points and Epipolar geometry:Fundamental Matrix: cheirality

D. Nister. An efficient solution to the five-point relative pose problem. In Proc. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR'03), volume 2, pages 195–202, June 2003

The uncalibrated methods fail when faced with coplanar scene points, since there is then a continuum of possible solutions.

In the calibrated setting, coplanar scene points only cause at most a two-fold ambiguity [15, 17]. With a third view, the ambiguity is in general resolved.

chirality constraint: scene points should be in front of the cameras

Image points are represented by homogeneous 3-vectors q and q' in the first and second view, respectively. World points are represented by homogeneous 4-vectors Q. A per-

Any combination of R and t according to the above prescription satisfies the epipolar constraint (4). To resolve the inherent ambiguities, it is assumed that the first camera matrix is $[I \mid 0]$ and that t is of unit length. There are then the following four possible solutions for the second camera matrix: $P_A \equiv [R_a \mid t_u]$, $P_B \equiv [R_a \mid -t_u]$, $P_C \equiv [R_b \mid t_u]$ $P_D \equiv [R_b \mid -t_u]$. One of the four choices corresponds to the true configuration. Another one corresponds to the twisted pair which is obtained by rotating one of the views 180 degrees around the baseline. The remaining two correspond to reflections of the true configuration and the twisted pair. For example, P_A gives one configuration. P_C corresponds to its twisted pair, which is obtained by applying the transformation

$$H_t \equiv \begin{bmatrix} I & & & 0 \\ -2v_{13} & -2v_{23} & -2v_{33} & -1 \end{bmatrix}. \qquad (22)$$

P_B and P_D correspond to the reflections obtained by applying $H_r \equiv diag(1, 1, 1, -1)$. In order to determine which choice corresponds to the true configuration, the cheirality constraint [1] is imposed. One point is sufficient to resolve the ambiguity. The point is triangulated using the view pair $([I \mid 0], P_A)$ to yield the space point Q and cheirality is tested. If $c_1 \equiv Q_3 Q_4 < 0$, the point is behind the first camera. If $c_2 \equiv (P_A Q)_3 Q_4 < 0$, the point is behind the second camera. If $c_1 > 0$ and $c_2 > 0$, P_A and Q correspond to the true configuration. If $c_1 < 0$ and $c_2 < 0$, the reflection H_r is applied and we get P_B. If on the other hand $c_1 c_2 < 0$, the twist H_t is applied and we get P_C and the point $H_t Q$. In this case, if $Q_3 (H_t Q)_4 > 0$ we are done. Otherwise, the reflection H_r is applied and we get P_D.

Points and Epipolar geometry:Fundamental Matrix: cheirality

chirality constraint: scene points should be in front of the cameras

R. Hartley. Cheirality invariants. In DARPA'93, pages 745–753, 1993.
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.4493&rep=rep1&type=pdf

from "Multiple View Geometry in Computer Vision"
Second Edition by Hartley & Zisserman
https://www.robots.ox.ac.uk/~vgg/hzbook/code/
(Note, consult author and book for any of this. use it's presented here for educational purposes only.)

see vgg_multiview/vgg_F_from_7pts_2img.m:50:function OK

Camera self-calibration

Camera Calibration: a USU Implementation*

Lili Ma, YangQuan Chen, and Kevin L. Moore
Center for Self-Organizing and Intelligent Systems
Department of Electrical and Computer Engineering
Utah State University
Email: lilima@cc.usu.edu

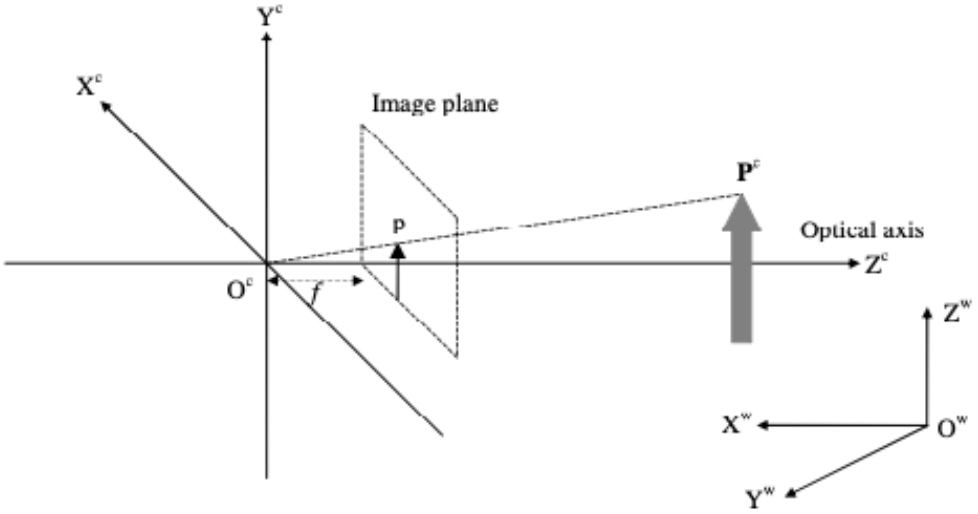
October 29, 2018

Table 1: List of Variables

Variable	Explanation
n	Number of feature points in each image
N	Number of images taken for calibration
f	Focal length
θ	Skewness angle between two image axes
$P^w = [X^w, Y^w, Z^w]^T$	3-D point in world reference frame
$P^c = [X^c, Y^c, Z^c]^T$	3-D point in camera reference frame
$p = [x^c,\ y^c]^T$	2-D point in camera frame with $z = f$
$M_i = [X^w, Y^w, 1]^T$	3-D point in world reference frame with $Z^w = 0$
m_i	The corresponding projected point in image plane of M_i
$\alpha, \beta, \gamma, u_0, v_0$	5 intrinsic parameters
$(s_x,\ s_y)$	Effective sizes of the pixel in the horizontal and vertical directions
$(f_x,\ f_y)$	$f_x = f/s_x,\ f_y = f/s_y$
$\mathbf{k} = (k_1,\ k_2)$	Distortion coefficients
$(u_d,\ v_d)$	Real observed distorted image points
$(u,\ v)$	Ideal projected undistorted image points
$(x,\ y)$	$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 1/f \begin{bmatrix} x^c \\ y^c \\ f \end{bmatrix} = A^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$
$(x',\ y')$	$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = A^{-1} \begin{bmatrix} u_d \\ v_d \\ 1 \end{bmatrix}$
λ	Scaling factor
J	Objective function
$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	Camera intrinsic matrix
$\mathbf{B} = \mathbf{A^{-T}A^{-1}}$	Absolute conic
$\mathbf{R}$	3×3 rotation matrix
$\mathbf{t}$	3×1 translation vector
$[\mathbf{R}\ \mathbf{t}]$	Camera extrinsic matrix
$\mathbf{H}$	Homography matrix
$\mathbf{L}$	Matrix used to estimate homography matrix $\mathbf{H}$
$\mathbf{V}$	Matrix stacking constraints to estimate intrinsic parameters

$$\therefore A^{-1} = \begin{bmatrix} \frac{1}{\alpha} & -\frac{\gamma}{\alpha\beta} & -\frac{u_0}{\alpha} + \frac{v_0\gamma}{\alpha\beta} \\ 0 & \frac{1}{\beta} & -\frac{v_0}{\beta} \\ 0 & 0 & 1 \end{bmatrix}$$

Camera self-calibration (cont)



the projection from 3-D world reference frame to image plane (2-D) causes direct depth information to be lost so that each point on the image plane corresponds to a ray in the 3-D space [7]

Figure 1: The perspective camera model

in Figure 1, the image of P^c is the point at which the straight line through O_c and P^c intersects the image plane. The basic perspective projection [8] in the camera frame is

$$\begin{bmatrix} x^c \\ y^c \end{bmatrix} = \frac{f}{Z^c} \begin{bmatrix} X^c \\ Y^c \end{bmatrix}, \tag{1}$$

Camera self-calibration (cont)

4.1 Extrinsic Parameters

The extrinsic parameters are defined as any set of geometric parameters that uniquely define the transformation between the world reference frame and the camera frame. A typical choice for describing the transformation is to use a 3×1 vector $\mathbf{t}$ and a 3×3 orthogonal rotation matrix $\mathbf{R}$ such that $P^c = \mathbf{R}P^w + \mathbf{t}$. According to Euler's rotation theorem, an arbitrary rotation can be described by only three parameters. As a result, the rotation matrix $\mathbf{R}$ has 3 degree-of-freedom and the extrinsic parameters totally have 6 degree of freedom. Given a rotation matrix $\mathbf{R}$ in Equation (2), one method to get the 3 parameters that uniquely describe this matrix is to extract ZYZ Euler angles [9], denoted by (a, b, c), such that

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2}$$

$$\mathbf{R} = \mathbf{R}_z(a)\ \mathbf{R}_y(b)\ \mathbf{R}_z(c), \tag{3}$$

where

$$\mathbf{R}_z(c) = \begin{bmatrix} \cos(c) & -\sin(c) & 0 \\ \sin(c) & \cos(c) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_y(b) = \begin{bmatrix} \cos(b) & 0 & \sin(b) \\ 0 & 1 & 0 \\ -\sin(b) & 0 & \cos(b) \end{bmatrix}. \tag{4}$$

When $sin(b) \neq 0$, the solutions for (a, b, c) are

$$\begin{aligned} b &= \arctan 2\ (\sqrt{r_{31}^2 + r_{32}^2}, r_{33}), \\ a &= \arctan 2\ (r_{23}/\sin(b), r_{13}/\sin(b)), \\ c &= \arctan 2\ (r_{32}/\sin(b), -r_{31}/\sin(b)). \end{aligned} \tag{5}$$

Camera self-calibration (cont)

The intrinsic parameters are as follows:

- The focal length: f
- The parameters defining the transformation between the camera frame and the image plane
  Neglecting any geometric distortion and with the assumption that the CCD array is made of rectangular grid of photosensitive elements, we have:

$$\begin{aligned} x^c &= -(u - u_0)\ s_x \\ y^c &= -(v - v_0)\ s_y \end{aligned} \tag{6}$$

  with (u_0, v_0) the coordinates in pixel of the image center and s_x, s_y the effective sizes of the pixel in the horizontal and vertical direction respectively. Let $f_x = f/s_x, f_y = f/s_y$, the current set of intrinsic parameters are u_0, v_0, f_x, and f_y.

- The parameter describing the skewness of the two image axes: $\gamma = f_y \tan\theta$
  The skewness of two image axes is illustrated in Figure 4.

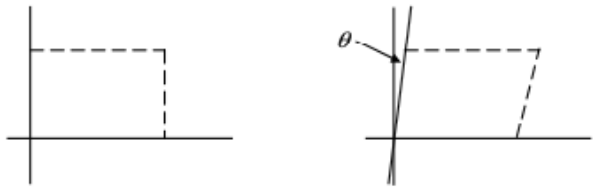


Figure 4: Skewness of two image axes

Camera self-calibration (cont)

- The parameters characterizing the radial distortion: k_1 and k_2
  The radial distortion is governed by the Equation [10]

$$F(r) = r\ f(r) = r\ (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots). \tag{7}$$

Two coefficients for distortion are usually enough. The relationship between the distorted and the undistorted image points can be approximated using

$$\begin{aligned} u_d &= u + (u - u_0)\ [k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \\ v_d &= v + (v - v_0)\ [k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \end{aligned} \tag{8}$$

where (u_d, v_d) are the real observed distorted image points and (u, v) the ideal projected undistorted image points. So, till now, the set of intrinsic parameters are $u_0, v_0, f_x, f_y, \gamma, k_1$, and k_2.

Camera self-calibration (cont)

4.3 Projection Matrix

With homogeneous transform and the camera parameters, we can have a 3×4 matrix $\mathbf{M}$, called the *projection matrix*, that directly links a point in the 3-D world reference frame to its projection in the image plane. That is:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{M} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \begin{bmatrix} X^w \\ Y^w \\ Z^w \\ 1 \end{bmatrix}. \tag{9}$$

λ is *distance of* X *from camera focal plane, but is only equal to the third coordinate of* X *when* M *is normalized, else it is equal to the square root of the sum of squares of the third row of M.*
It' is referred to as a scale factor.

where λ is an arbitrary scaling factor and the matrix $\mathbf{A}$ fully depends on the intrinsic parameters. The calibration method used in this work is to first estimate the projection matrix and then use the absolute conic to estimate the intrinsic parameters [1, 2]. From Equation (1) and (6), we have

$$u = -\frac{f}{s_x}\frac{X^c}{Z^c} + u_0. \tag{10}$$

From Equation (9), we have

$$u = \alpha \frac{X^c}{Z^c} + u_0 \tag{11}$$

with scaling factor $\lambda = Z^c$. From the above two equations, we get $\alpha = -f/s_x = -f_x$. In a same manner, $\beta = -f_y$.

Camera self-calibration (cont)

Table 4: Camera Calibration Procedures

Linear Parameter Estimation

Estimate Homographies (Section 6.1)
Let N be the number of images that we want to observe
Loop for i from 1 to N
    Assume the calibration object is at $Z^w = 0$
    Establishes the i^{th} homography between the calibration object and its image
    Change the orientation of either calibration object or camera
End Loop

Estimate Intrinsic Parameters (Section 6.3)
For each homography we have 2 constraints concerning the 5 intrinsic parameters
Now we have $2N$ constraints and we can solve the 5 intrinsic parameters using `SVD`

Estimate Extrinsic Parameters (Section 6.4)
Using the estimated intrinsic parameters and homographies, we can estimate the extrinsic parameters

Estimate Distortion Coefficients (Section 6.5)
Using the estimated intrinsic and extrinsic parameters, we can get the ideal projected image points
Along with the real observed image points, we can estimate the two distortion coefficients (k_1, k_2)

Nonlinear Optimization
(Section 6.6)
Take all parameters estimated above as an initial guess
Use some nonlinear optimization routine, we can get the final estimated values

Camera self-calibration (cont)

(1) estimate homographies (the projection matrices)

assume the calibration object is $Z^w = 0$ in the world reference frame.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} X^w \\ Y^w \\ 1 \end{bmatrix}, \qquad \mathbf{H} = \mathbf{A}\ [\mathbf{r_1} \quad \mathbf{r_2} \quad \mathbf{t}].$$

Let M_i and m_i be the model and its image point respectively. assume m_i is corrupted by Gaussian noise with mean 0 and covariance matrix Λ_m_i. Then the maximum likelihood estimation of H is obtained by minimizing

$$\sum_i (m_i - \hat{m}_i)^T \mathbf{\Lambda}_{m_i} (m_i - \hat{m}_i),$$

where $\bar{h}_i$ is the i^{th} row of H.

assume $\mathbf{\Lambda}_{m_i} = \sigma^2\ \mathbf{I}$ for all i.

Let $\mathbf{x} = [\bar{\mathbf{h}}_1^T, \bar{\mathbf{h}}_2^T, \bar{\mathbf{h}}_3^T]$, then

When we are given n points, we have n above equations and we can write them in matrix form as $\mathbf{Lx} = \mathbf{0}$ where

$$\mathbf{L} = \begin{bmatrix} X_1^w & Y_1^w & 1 & 0 & 0 & 0 & -u_1 X_1^w & -u_1 Y_1^w & -u_1 \\ 0 & 0 & 0 & X_1^w & Y_1^w & 1 & -v_1 X_1^w & -v_1 Y_1^w & -v_1 \\ X_2^w & Y_2^w & 1 & 0 & 0 & 0 & -u_2 X_2^w & -u_2 Y_2^w & -u_2 \\ 0 & 0 & 0 & X_2^w & Y_2^w & 1 & -v_2 X_2^w & -v_2 Y_2^w & -v_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ X_n^w & Y_n^w & 1 & 0 & 0 & 0 & -u_n X_n^w & -u_n Y_n^w & -u_n \\ 0 & 0 & 0 & X_n^w & Y_n^w & 1 & -v_n X_n^w & -v_n Y_n^w & -v_n \end{bmatrix}.$$

The matrix $\mathbf{L}$ is a $2n \times 9$ matrix and the solution is well known to be the right singular vector of $\mathbf{L}$ associated with the smallest singular value.

Camera self-calibration (cont)

(2) Estimate Intrinsic Parameters

Having solved for x = $[\bar{\mathbf{h}}_1^T, \bar{\mathbf{h}}_2^T, \bar{\mathbf{h}}_3^T]$, we have $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3] = \lambda\ \mathbf{A}\ [\mathbf{r}_1, \mathbf{r}_2, \mathbf{t}]$,

Constraints on the Intrinsic Parameters

λ an arbitrary scalar

$\mathbf{r}_1, \mathbf{r}_2$ are orthogonal

$$\begin{aligned} \mathbf{r}_1{}^T \mathbf{r}_2 &= 0 \\ \mathbf{r}_1{}^T \mathbf{r}_1 &= \mathbf{r}_2{}^T \mathbf{r}_2. \end{aligned}$$

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

$$\begin{aligned} &\mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2 = 0 \\ &\mathbf{h}_1^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2. \end{aligned}$$

typo: should be
$\mathbf{H} = \mathbf{A}\ [\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}]$.

Camera self-calibration (cont)

Estimate the Intrinsic Parameters

$$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$ Camera intrinsic matrix

the absolute conic is the plane at infinity x^2 + y^2 + z^2 = 0 = t, so is unaffected by camera extrinsic parameters,

WCS point M is (^M^T, 0) => m = A{R|t] (^M^T 0) = AR^M^T

^M^T*^M=0 => m^T(A^-T * A^-1)*m = 0

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

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix}.$$

Define $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$.

Let the i^{th} column vector of $\mathbf{H}$ be $\mathbf{h}_i = [h_{i1}, h_{i2}, h_{i3}]^T$

$$\mathbf{V}_{ij} = [h_{i1}h_{j1},\ h_{i1}h_{j2} + h_{i2}h_{j1},\ h_{i2}h_{j2},\ h_{i3}h_{j1} + h_{i1}h_{j3},\ h_{i3}h_{j2} + h_{i2}h_{j3},\ h_{i3}h_{j3}]^T, \qquad (24)$$

$$\begin{bmatrix} \mathbf{V}_{12}^T \\ (\mathbf{V}_{11} - \mathbf{V}_{12})^T \end{bmatrix} \mathbf{b} = \mathbf{0}.$$

If N images of the calibration object are taken, by stacking N such equations, we have $\mathbf{Vb} = \mathbf{0}$ where $\mathbf{V}$ is a $2N \times 6$ matrix.

When $N \geq 3$, we will have a unique solution defined up to a scaling factor. The solution is well known to be the right singular vector of $\mathbf{V}$ associated with the smallest singular value. The matrix $\mathbf{B}$

Camera self-calibration (cont)

Having solved for $\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T$

The matrix $\mathbf{B}$ is estimated up to a scaling factor $\mathbf{B} = \lambda\, \mathbf{A}^{-T}\mathbf{A}^{-1}$. After estimation of $\mathbf{B}$, the intrinsic parameters can be extracted from $\mathbf{B}$ by

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2), \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11}, \\ \alpha &= \sqrt{\lambda/B_{11}}, \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)}, \\ \gamma &= -B_{12}\alpha^2\beta/\lambda, \\ u_0 &= \gamma v_0/\beta - B_{13}\alpha^2/\lambda. \end{aligned} \tag{26}$$

Camera self-calibration (cont)

(3) Estimate Extrinsic Parameters

$\mathbf{A} = \begin{bmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix}$	Camera intrinsic matrix

Once $\mathbf{A}$ is known, the extrinsic parameters can be estimated as:

$$\begin{aligned} \mathbf{r}_1 &= \lambda\, \mathbf{A}^{-1}\mathbf{h}_1 \\ \mathbf{r}_2 &= \lambda\, \mathbf{A}^{-1}\mathbf{h}_2 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2 \\ \mathbf{t} &= \lambda\, \mathbf{A}^{-1}\mathbf{h}_3 \end{aligned} \tag{27}$$

where $\lambda = 1/||\mathbf{A}^{-1}\mathbf{h}_1||_2 = 1/||\mathbf{A}^{-1}\mathbf{h}_2||_2$. Of course, due to the noise in the data, the computed matrix $\mathbf{R} = [\mathbf{r}_1\ \mathbf{r}_2\ \mathbf{r}_3]$ does not satisfy the properties of a rotation matrix $\mathbf{R}\mathbf{R}^T = \mathbf{I}$. One way to estimate the best rotation matrix from a general 3×3 matrix $\mathbf{R}$ is: by the Matlab function `svd` with $[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \texttt{svd}\,(\mathbf{R})$. The best rotation matrix will be $\mathbf{U}\mathbf{V}^T$.

Camera self-calibration (cont)

(4) Estimate Radial Distortion Coefficients

Assume the center of distortion is the same as the principal point, Equation (8) describes the relationship between the ideal projected undistorted image points (u, v) and the real observed distorted image points (u_d, v_d). Given n points in N images, we can stack all equations together to obtain totally $2Nn$ equations in matrix form as $\mathbf{Dk} = \mathbf{d}$, where $\mathbf{k} = [k_1, k_2]^T$. The linear least-square solutions for $\mathbf{k}$ is

$$\mathbf{k} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{d}. \tag{28}$$

```
eqn(8):
 u + (u-u_0)*[k_1*(x^2 + y^2) + k_2*(x^2+y^2)^2] = u_d
 v + (v-v_0)*[k_1*(x^2 + y^2) + k_2*(x^2+y^2)^2] = v_d

 [k_1*(x^2 + y^2) + k_2*(x^2+y^2)^2]*(u-u_0) + u = u_d
 [k_1*(x^2 + y^2) + k_2*(x^2+y^2)^2]*(v-v_0) + v = v_d

 [ [(x^2 + y^2)   (x^2+y^2)^2]*(u-u_0)   u ] * [k1] = u_d
 [ [(x^2 + y^2)   (x^2+y^2)^2]*(v-v_0)   v ]   [k2] = v_d
                                               [1 ]

  ==> D = [ [(x^2 + y^2)   (x^2+y^2)^2]*(u-u_0)   u ]
          [ [(x^2 + y^2)   (x^2+y^2)^2]*(v-v_0)   v ]
          [ next point on 2 rows                    ]
          [...                                      ]

  ==> d =[u_d]
         [v_d]
         [u_d for next point]
```

Camera self-calibration (cont)

(5) Nonlinear Optimization

Assume that the image points are corrupted independently by identically distributed noise, the maximum likelihood estimation can be obtained by minimizing the following objective function

$$J = \sum_{i=1}^{N}\sum_{j=1}^{n} ||m_{ij} - \hat{m}(\mathbf{A}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, M_j)||^2, \tag{29}$$

where $\hat{m}(\mathbf{A}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, M_j)$ is the projection of point M_j in the i^{th}image using the estimated parameters. This is a nonlinear optimization problem that can be solved by Matlab optimization function `fminunc`.

In our implementation, one observation is that without an initial estimation of distortion coefficients (k_1, k_2), simply setting them to be 0, gives the same optimization results as the case with a good initial guess. Clearly, a "good" initial guess of the distortion coefficients is not practically required.

$M_i = [X^w, Y^w, 1]^T$	3-D point in world reference frame with $Z^w = 0$
m_i	The corresponding projected point in image plane of M_i

11.3.3 *Gradient descent nonlinear refinement ("bundle adjustment")*

The quality of the estimates obtained by Algorithm 11.7 is measured by the reprojection error[19]

$$e_r = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} \left\| \boldsymbol{x}_i^j - \pi(\Pi_i \boldsymbol{X}^j) \right\|^2. \tag{11.18}$$

[19] We use π to denote the standard planar projection $\pi : [X, Y, Z]^T \mapsto [X/Z, Y/Z, 1]^T$, introduced in Chapter 3.

Camera self-calibration (cont)

(5) Nonlinear Optimization

see Levenberg-Marquardt_projection_notes.pdf

and notes in Task 6 of story_tasks_007.txt

$$P = \lambda \mathbf{A}^*[\,\mathbf{R} \mid \mathbf{t}\,]$$

Camera self-calibration (cont)

Constrained Nonlinear Optimization

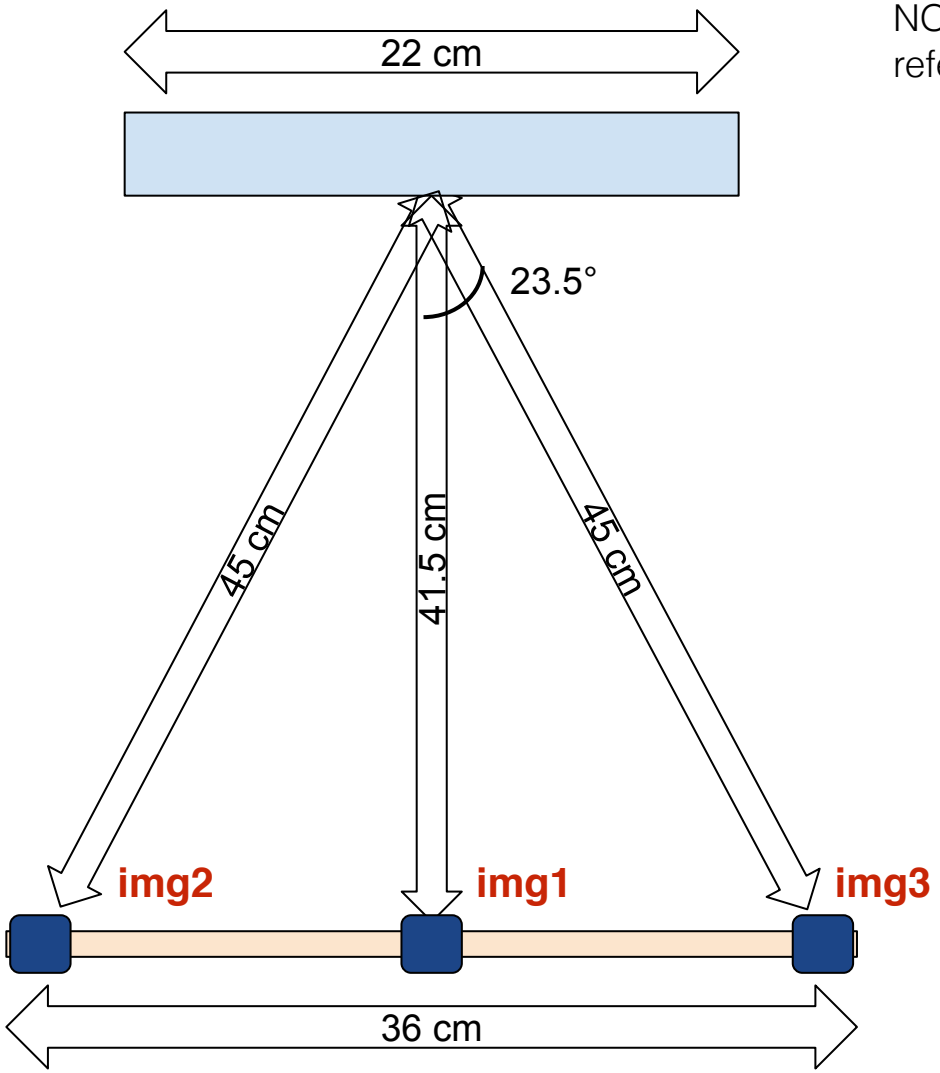
Another idea that might speed up the nonlinear optimization is to add some constrains, such as:

- u_0 is close to half of image's length
- v_0 is close to half of image's width
- angle between two image axes is close to 90 degree
- $\alpha,\ \beta$ are greater than 0

Camera self-calibration (cont)

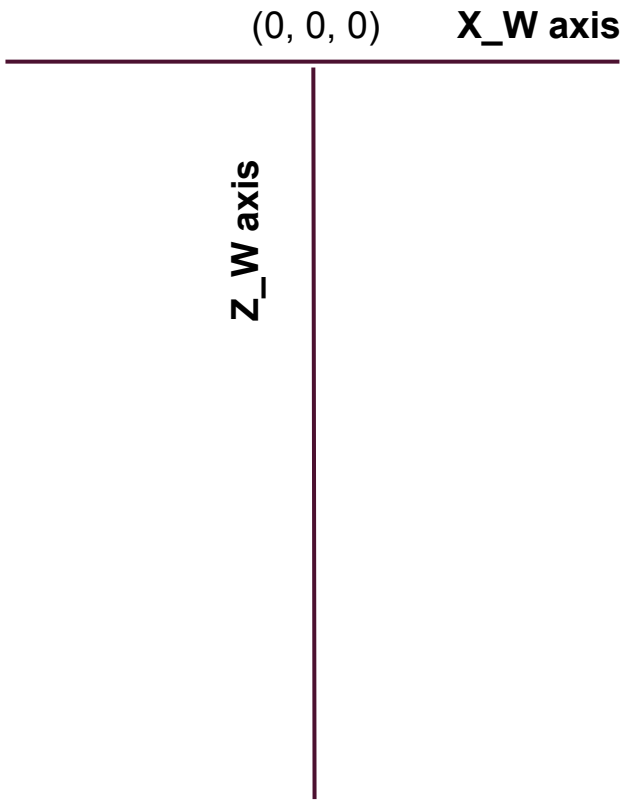
an example in this project's test code of 3 camera poses for images of the same book

Top View.
Looking down at book (light blue) and 3 camera positions (dark blue)

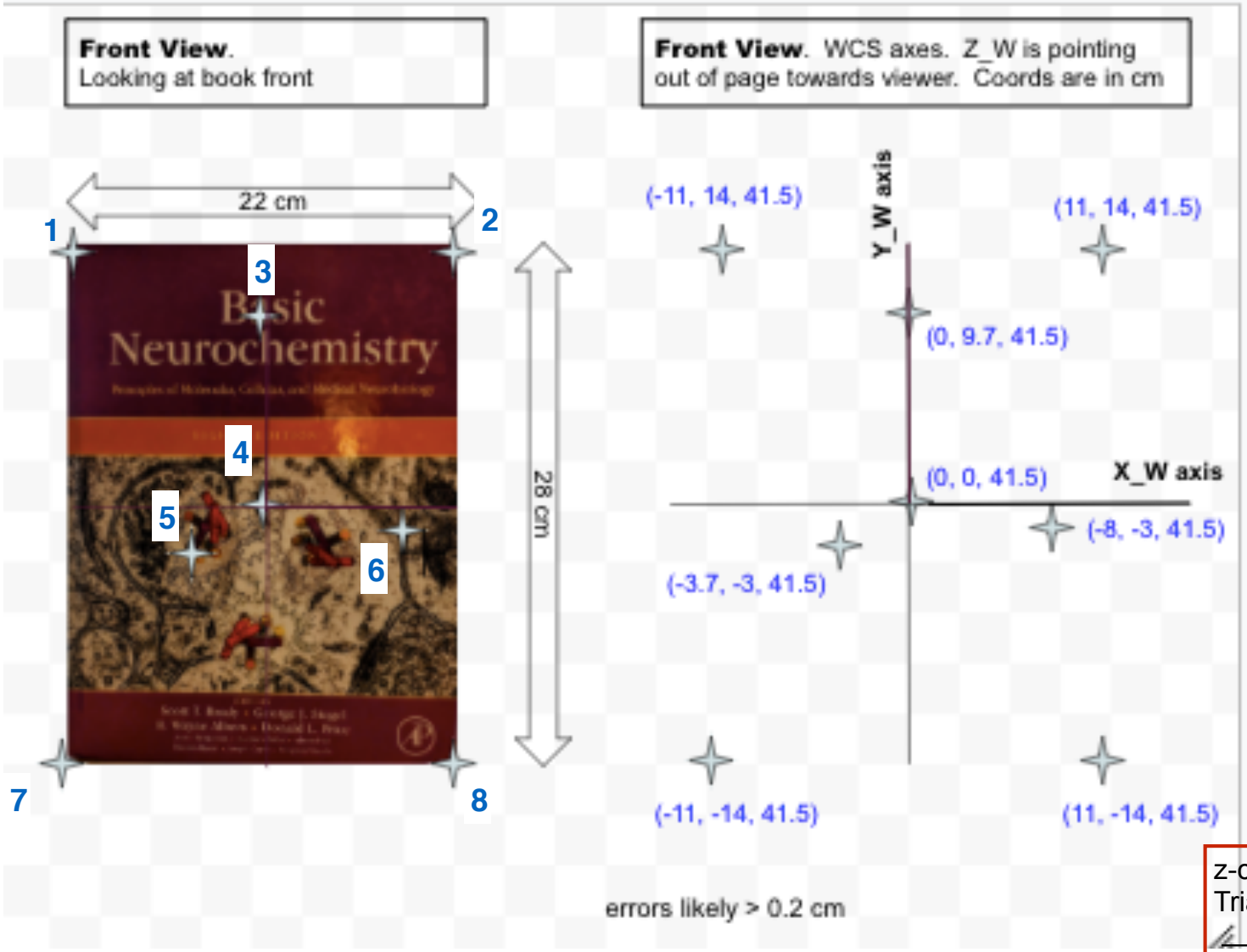


Top View. WCS axes. Y_W is pointing out of page towards the viewer.

NOTE: the WCS reference frame is not the Earth's reference frame instead it's a table's reference frame.



Camera self-calibration (cont)



z-coord found using Triangulation.calculateWCSPoint()

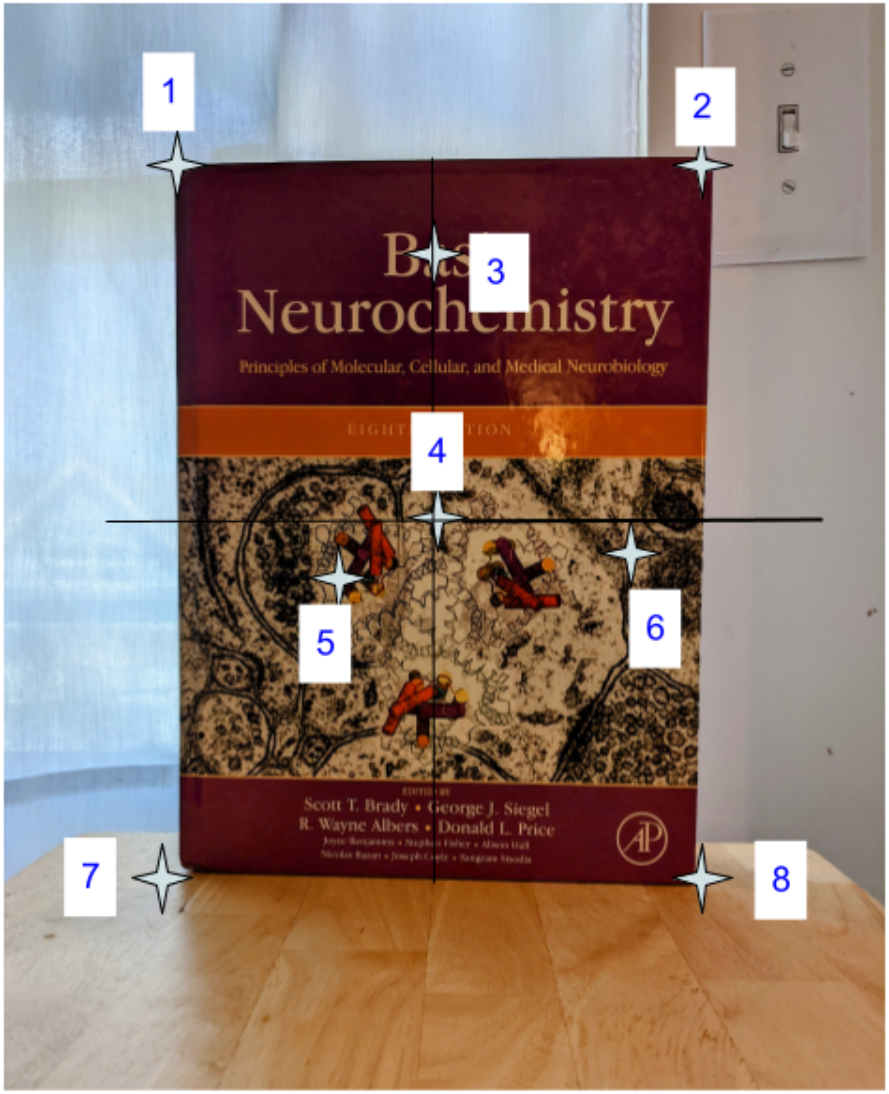
	x-coord	y-coord	z-coord
xw[0]=	-1.692e+01,	-3.061e+01,	2.969e+01
xw[1]=	1.704e+01,	-2.268e+01,	2.810e+01
xw[2]=	-7.236e-01,	-2.789e+01,	4.122e+01
xw[3]=	-7.018e-01,	-7.526e+00,	4.309e+01
xw[4]=	-9.202e+00,	-1.750e+00,	4.099e+01
xw[5]=	1.462e+01,	-3.271e+00,	3.703e+01
xw[6]=	-1.826e+01,	2.066e+01,	3.163e+01
xw[7]=	1.782e+01,	1.601e+01,	3.332e+01

Camera self-calibration (cont)

Front View. image coordinate frame.

(0, 0, 0) **u axis**

v axis



CameraCalibration.estimateCamera() shows I need to use many more than 8 points for a good solution.

images are 3024 × 4032

errors likely > 8 pixels

	img2	img1	img3
#1	678, 718	608, 530	744, 806
#2	2210, 886	2462, 512	2286, 526
#3	1504, 1102	1484, 858	1410, 992
#4	1516, 1814	1486, 1678	1432, 1760
#5	1228, 2014	1154, 1882	1164, 1940
#6	2042, 1936	2142, 1810	2018, 1866
#7	698, 2944	614, 2802	788, 2728
#8	2210, 2782	2366, 2848	2276, 2930

multi-view geometry (cont.)

In general:
— dot product of a point and a line is zero if the point lies on the line
— if the *intrinsic parameters* are known, the relationship between corresponding points is given by the *essential matrix*: $\mathbf{m_r^T * E * m_l^T}$
— when no camera details are available, one can use the *fundamental matrix*.
It's a 3x3 rank 2 homogeneous matrix. It has 7 degrees of freedom.
For any point $\mathbf{m_l}$ in the left image, the corresponding epipolar line $\mathbf{l_r}$ in the right image can be expressed as $\mathbf{l_r = F * m_l}$ and vice versa $\mathbf{l_r = F^T * m_r}$
— the *essential and fundamental matrices* are related through $\mathbf{F} = \mathbf{K_r^{-T}} * \mathbf{E} * \mathbf{K_l^{-T}}$ where Kr and Kl are the left and right camera matrices

Regarding 3D reconstruction:
— when both *intrinsic* and *extrinsic* camera parameters are known, the reconstruction is solved unambiguously by triangulation.
— when only *intrinsic* parameters are known, extrinsic parameters can be estimated and the reconstruction can be solved up to an unknown scale factor, that is **R** can be estimated, but **t** can be only up to a scale factor. the epipolar geometry is the essential matrix and the solvable projection is euclidean (rigid+ uniform scale)
— when neither *intrinsic* nor *extrinsic* parameters are known, i.e., the only information available are pixel correspondences, the reconstruction can be solved up to an unknown, global projective transformation of the world.

In Trifocal geometry, a trifocal tensor is used.

Statistics of matching correspondence: Fundamental Matrix

from “Multiple View Geometry in Computer Vision”
Second Edition by Hartley & Zisserman
https://www.robots.ox.ac.uk/~vgg/hzbook/
(Note, consult author and book for any of this. use it’s presented here for educational purposes only.)

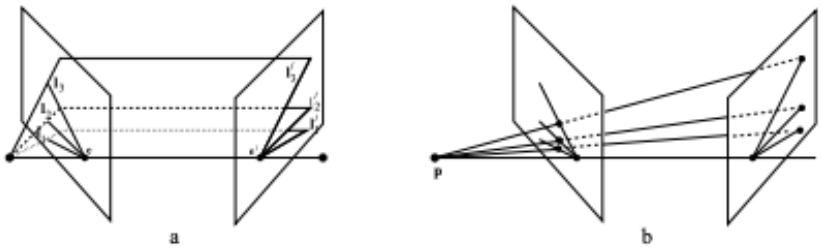


Fig. 9.6. **Epipolar line homography.** *(a) There is a pencil of epipolar lines in each image centred on the epipole. The correspondence between epipolar lines, $\mathbf{l}_i \leftrightarrow \mathbf{l}'_i$, is defined by the pencil of planes with axis the baseline. (b) The corresponding lines are related by a perspectivity with centre any point* $\mathbf{p}$ *on the baseline. It follows that the correspondence between epipolar lines in the pencils is a 1D homography.*

- $\mathtt{F}$ is a rank 2 homogeneous matrix with 7 degrees of freedom.
- **Point correspondence**: If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then $\mathbf{x}'^{\mathsf{T}}\mathtt{F}\mathbf{x} = 0$.
- **Epipolar lines**:
  - $\mathbf{l}' = \mathtt{F}\mathbf{x}$ is the epipolar line corresponding to $\mathbf{x}$.
  - $\mathbf{l} = \mathtt{F}^{\mathsf{T}}\mathbf{x}'$ is the epipolar line corresponding to $\mathbf{x}'$.
- **Epipoles**:
  - $\mathtt{F}\mathbf{e} = \mathbf{0}$.
  - $\mathtt{F}^{\mathsf{T}}\mathbf{e}' = \mathbf{0}$.
- **Computation from camera matrices** $\mathtt{P}, \mathtt{P}'$:
  - General cameras, $\mathtt{F} = [\mathbf{e}']_\times \mathtt{P}'\mathtt{P}^+$, where $\mathtt{P}^+$ is the pseudo-inverse of $\mathtt{P}$, and $\mathbf{e}' = \mathtt{P}'\mathbf{C}$, with $\mathtt{P}\mathbf{C} = \mathbf{0}$.
  - Canonical cameras, $\mathtt{P} = [\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = [\mathtt{M} \mid \mathbf{m}]$, $\mathtt{F} = [\mathbf{e}']_\times \mathtt{M} = \mathtt{M}^{-\mathsf{T}}[\mathbf{e}]_\times$, where $\mathbf{e}' = \mathbf{m}$ and $\mathbf{e} = \mathtt{M}^{-1}\mathbf{m}$.
  - Cameras not at infinity $\mathtt{P} = \mathtt{K}[\mathtt{I} \mid \mathbf{0}]$, $\mathtt{P}' = \mathtt{K}'[\mathtt{R} \mid \mathbf{t}]$, $\mathtt{F} = \mathtt{K}'^{-\mathsf{T}}[\mathbf{t}]_\times \mathtt{R}\mathtt{K}^{-1} = [\mathtt{K}'\mathbf{t}]_\times \mathtt{K}'\mathtt{R}\mathtt{K}^{-1} = \mathtt{K}'^{-\mathsf{T}}\mathtt{R}\mathtt{K}^{\mathsf{T}}[\mathtt{K}\mathtt{R}^{\mathsf{T}}\mathbf{t}]_\times$.

Table 9.1. *Summary of fundamental matrix properties.*

Bundle Adjustment (see slides further on)

from “Bundle Adjustment — A Modern Synthesis”
by Triggs, McLauchlan, Hartley, and Fitzgibbon
Vision Algorithms’99, LNCS 1883, pp. 298–372, 2000
http://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Triggs00.pdf
(Note, consult author for any use of figures and text - it’s presented here for educational purposes only.)

(not implemented in this project currently)

Bundle adjustment: refining a visual reconstruction to produce jointly optimal 3D structure and viewing parameter (camera pose and/or calibration) estimates, by minimizing a cost function that quantifies the model fitting error, and jointly that the solution is simultaneously optimal with respect to both structure and camera variations… a large sparse geometric parameter estimation problem, the parameters being the combined 3D feature coordinates, camera poses and calibrations.

cost function: often use non-quadratic M-estimator-like distributional models to handle outliers more integrally, and many include additional penalties related to overfitting, model selection and system performance (priors, MDL).
modelled as a sum of opaque contributions from the independent information sources (individual observations, prior distributions, overfitting penalties ...). The functional forms of these contributions and their dependence on fixed quantities such as observations will usually be left implicit

main conclusion will be that robust statistically-based error metrics based on total (inlier + outlier) log likelihoods should be used, to correctly allow for the presence of outliers… A typical ML cost function would be the *summed negative log likelihoods* of the prediction errors of all the observed image features. For Gaussian error distributions, this reduces to the *sum of squared covariance-weighted prediction errors* (§3.2). A MAP estimator (is Bayesian) would typically add cost terms giving certain structure or camera calibration parameters a bias towards their expected values.

first realize that geometric fitting is really a special case of parametric probability density estimation.
Maximizing the likelihood corresponds to fitting this predicted observation density to the observed data. The geometry and camera model only enter indirectly, via their influence on the predicted distributions.
ML estimation is naturally robust to outliers.

Pose Estimates

http://www.cs.cmu.edu/~16385/s17/Slides/11.3_Pose_Estimation.pdf

1 camera

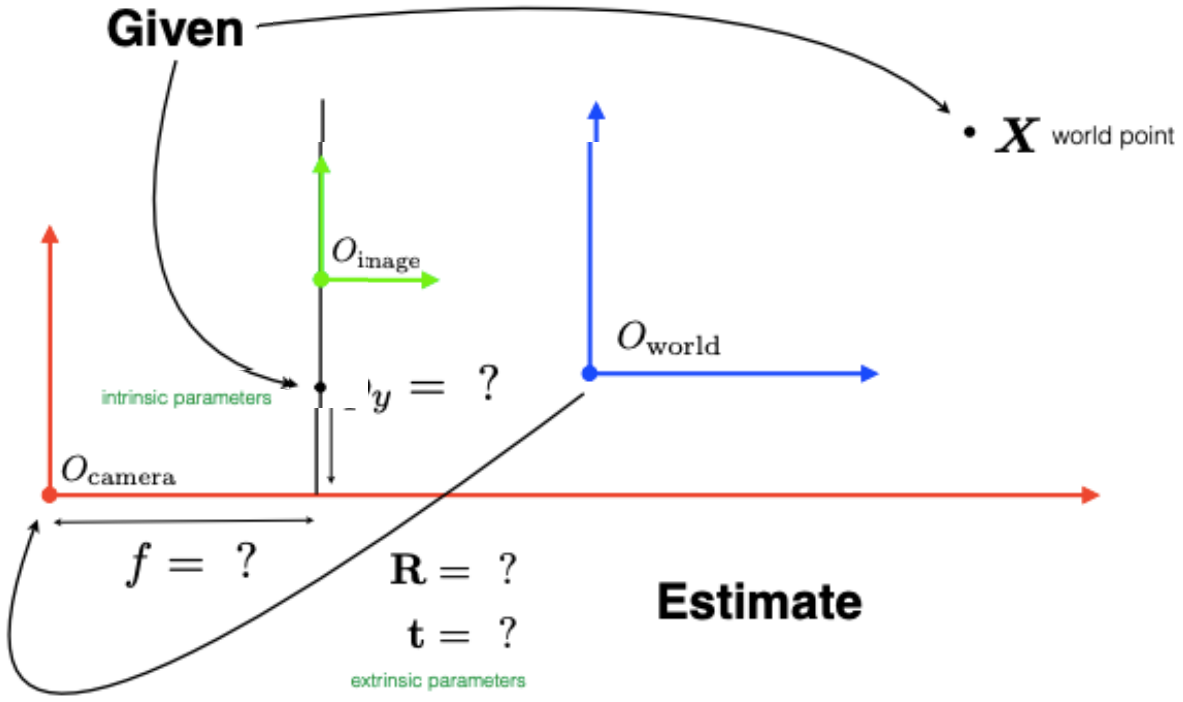
Given a set of matched points $\{\mathbf{X}_i, \boldsymbol{x}_i\}$ and camera model $\boldsymbol{f}(\mathbf{X}; \boldsymbol{p})$

$\mathbf{X}_i$: point in 3D space; $\boldsymbol{x}_i$: point in the image; $\boldsymbol{f}$: projection model; $\boldsymbol{p}$: parameters

Find the (pose) estimate of $\mathbf{P}$

Camera matrix

$$\boldsymbol{x} = \boldsymbol{f}(\mathbf{X}; \boldsymbol{p}) = \mathbf{PX}$$



From Szeliski 2010:
Pose estimate is a.k.a. *perspective-3-point-problem* (P3P), with extensions to larger numbers of points collectively known as PnP (Haralick, Lee, Ottenberg *et al.* 1994; Quan and Lan 1999; Moreno-Noguer, Lepetit, and Fua 2007)

Pose Estimates

http://www.cs.cmu.edu/~16385/s17/Slides/11.3_Pose_Estimation.pdf

Mapping between 3D point and image points

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} — & \boldsymbol{p}_1^\top & — \\ — & \boldsymbol{p}_2^\top & — \\ — & \boldsymbol{p}_3^\top & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

What are the unknowns?

Inhomogeneous coordinates

$$x' = \frac{\boldsymbol{p}_1^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}} \qquad y' = \frac{\boldsymbol{p}_2^\top \boldsymbol{X}}{\boldsymbol{p}_3^\top \boldsymbol{X}} \rightarrow \begin{aligned} \boldsymbol{p}_2^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} y' = 0 \\ \boldsymbol{p}_1^\top \boldsymbol{X} - \boldsymbol{p}_3^\top \boldsymbol{X} x' = 0 \end{aligned} \rightarrow \begin{bmatrix} \boldsymbol{X}_1^\top & \mathbf{0} & -x'\boldsymbol{X}_1^\top \\ \mathbf{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \mathbf{0} & -x'\boldsymbol{X}_N^\top \\ \mathbf{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix} = \mathbf{0}$$

Solve for

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} \|\mathbf{A}\boldsymbol{x}\|^2 \text{ subject to } \|\boldsymbol{x}\|^2 = 1$$

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{X}_1^\top & \mathbf{0} & -x'\boldsymbol{X}_1^\top \\ \mathbf{0} & \boldsymbol{X}_1^\top & -y'\boldsymbol{X}_1^\top \\ \vdots & \vdots & \vdots \\ \boldsymbol{X}_N^\top & \mathbf{0} & -x'\boldsymbol{X}_N^\top \\ \mathbf{0} & \boldsymbol{X}_N^\top & -y'\boldsymbol{X}_N^\top \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}_1 \\ \boldsymbol{p}_2 \\ \boldsymbol{p}_3 \end{bmatrix}$$

considering noise, rank is 12, solving in right null space. need N>=6.

Equivalently, solution ***x*** is the Eigenvector corresponding to smallest Eigenvalue of $\mathbf{A}^\top\mathbf{A}$

Solution **x** is the column of **V** corresponding to smallest singular value of

SVD

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$$

A is m x n = 2*N x 3*4
U is 2*N x 2*N
V is 3*4 x 3*4

A^TA is 3*4 x 3*4
U is 3*4 x 3*4
V is 3*4 x 3*4

Pose Estimates

http://www.cs.cmu.edu/~16385/s17/Slides/11.3_Pose_Estimation.pdf

$$\begin{bmatrix} — & \boldsymbol{p}_1^\top & — \\ — & \boldsymbol{p}_2^\top & — \\ — & \boldsymbol{p}_3^\top & — \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix}$$

now have p_1, p_2, p_3

How do you get the intrinsic and extrinsic parameters from the projection matrix?

Decomposition of the Camera Matrix

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

$$\begin{aligned} \mathbf{P} &= \mathbf{K}[\mathbf{R}|\mathbf{t}] \\ &= \mathbf{K}[\mathbf{R}|-\mathbf{R}\mathbf{c}] \\ &= [\mathbf{M}|-\mathbf{M}\mathbf{c}] \end{aligned}$$

Find the camera center **C**

$$\mathbf{P}\mathbf{c} = \mathbf{0}$$

SVD of P!

***c** is the Eigenvector corresponding to smallest Eigenvalue*

Find intrinsic **K** and rotation **R**

$$\mathbf{M} = \mathbf{K}\mathbf{R}$$

right upper triangle — orthogonal

use RQ decomposition

Simple AR program

1. Compute point correspondences (2D and AR tag)
2. Estimate the pose of the camera **P**
3. Project 3D content to image plane using **P**

Triangulation

https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html#XBeaZisMur97

given 2 camera data: projection matrices and pairs of correspondence in the images, determine the reconstructed 3D scene point. (see next few slides for DLT solution details)

the image point $\mathbf{m}_\ell' = P_{3\times3,r} P_{3\times3,l}^{-1} \mathbf{m}_\ell$ which represents the projection onto the right image plane of the point at infinity of the optical ray of $\mathbf{m}_\ell$

Analytically, the reconstructed 3D point **M** can be found by solving for parameter ζ_ℓ or ζ_r in Eq. (24). l

$$\mathbf{e}_r = \zeta_r m_r - \zeta_\ell m'_\ell$$

The depth ζ_r and ζ_ℓ are unknown. Both encode the position of **M** in space, as ζ_r is the depth of **M** wrt the right camera and ζ_ℓ is the depth of **M** wrt the left camera.

The three points $\mathbf{m}_r$,$\mathbf{e}_r$ and $\mathbf{m}_\ell'$ are known and are collinear, so we can solve for ζ_ℓ using the following closed form expressions [41]:

$$\zeta_r = \frac{(\mathbf{e}_r \times \mathbf{m}'_\ell) \cdot (\mathbf{m}_r \times \mathbf{m}'_\ell)}{||\mathbf{m}_r \times \mathbf{m}'_\ell||^2}$$

The reconstructed point **M** can then be calculated by inserting the value ζ into Equation (14).

$$\mathbf{M} = \begin{pmatrix} -P_{3\times3}^{-1}\mathbf{p}_4 \\ 1 \end{pmatrix} + \zeta \begin{pmatrix} P_{3\times3}^{-1}\mathbf{m} \\ 0 \end{pmatrix}, \quad \zeta \in \mathbb{R}. \tag{14}$$

Please note that, provided that P is normalized, the parameter ζ in the equation of the optical ray correspond to the depth of the point **M**.

In reality, camera parameters and image locations are known only approximately. The back-projected rays therefore do not actually intersect in space. It can be shown, however, that Formula (43) solve Eq. (42) in a least squares sense [26].

Triangulation

http://www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf

2 cameras

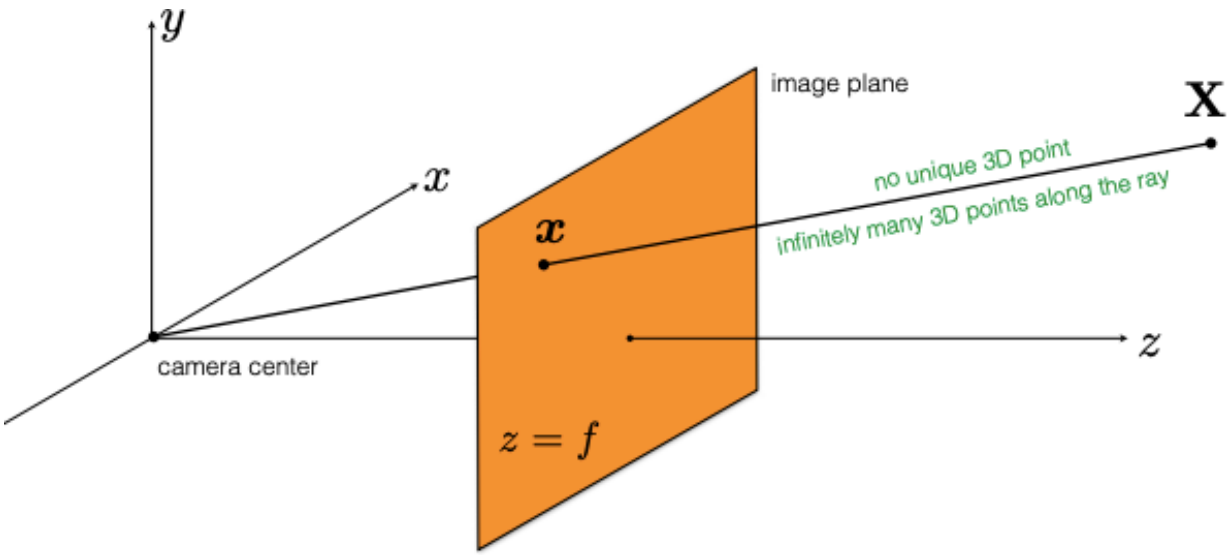
Given a set of (noisy) matched points $\{\boldsymbol{x}_i, \boldsymbol{x}_i'\}$ and camera matrices $\mathbf{P}, \mathbf{P}'$

Estimate the 3D point $\mathbf{X}$

$$\underset{\text{known}}{\boldsymbol{x}} = \boldsymbol{f}(\mathbf{X}; \boldsymbol{p}) = \underset{\text{known}}{\mathbf{P}}\mathbf{X}$$

With just 1 camera's image point, infinitely many 3D points along the ray

With 1 pair correspondence from 2 cameras, and if the measurements were **perfect**, could determine X location.



Triangulation

http://www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf

$$\mathbf{x} = \mathbf{P}\boldsymbol{X} \quad \text{(homogeneous coordinate)} \qquad \mathbf{x} = \alpha\mathbf{P}\boldsymbol{X} \quad \text{(inhomogeneous coordinate)}$$

Same ray direction but differs by a scale factor

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \alpha \begin{bmatrix} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

known (left matrix) known (right matrix)

cross product

$$\mathbf{x} \times \mathbf{P}\boldsymbol{X} = \mathbf{0}$$

Cross product of two vectors of same direction is zero
(this equality removes the scale factor)

$$= \alpha \begin{bmatrix} — & \boldsymbol{p}_1^\top & — \\ — & \boldsymbol{p}_2^\top & — \\ — & \boldsymbol{p}_3^\top & — \end{bmatrix} \begin{bmatrix} | \\ \boldsymbol{X} \\ | \end{bmatrix}$$

$$= \alpha \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix}$$

cross product

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \times \begin{bmatrix} \boldsymbol{p}_1^\top \boldsymbol{X} \\ \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_3^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} y\boldsymbol{p}_3^\top \boldsymbol{X} - \boldsymbol{p}_2^\top \boldsymbol{X} \\ \boldsymbol{p}_1^\top \boldsymbol{X} - x\boldsymbol{p}_3^\top \boldsymbol{X} \\ x\boldsymbol{p}_2^\top \boldsymbol{X} - y\boldsymbol{p}_1^\top \boldsymbol{X} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

echelon reduced:

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

formatting the point projection product as A:

$$\mathbf{A}_i \boldsymbol{X} = \mathbf{0}$$

Triangulation

http://www.cs.cmu.edu/~16385/s17/Slides/11.4_Triangulation.pdf

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\mathbf{A}_i \boldsymbol{X} = \mathbf{0}$$

Concatenate the 2D points from both images

1X4

$$\begin{bmatrix} y\boldsymbol{p}_3^\top - \boldsymbol{p}_2^\top \\ \boldsymbol{p}_1^\top - x\boldsymbol{p}_3^\top \\ y'\boldsymbol{p}_3'^\top - \boldsymbol{p}_2'^\top \\ \boldsymbol{p}_1'^\top - x'\boldsymbol{p}_3'^\top \end{bmatrix} \boldsymbol{X} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

add the remaining n-1 pair point correspondences to A

$$\mathbf{A}\boldsymbol{X} = \mathbf{0}$$

To find the fit perpendicular to A, hence X, can use the eigenvector corresponding to smallest eigenvalue of **A^T*A**

NOTE:
SVD(A).U == SVD(A^T).V == SVD(AA^T).U == SVD(AA^T).V
SVD(A).V == SVD(A^T).U == SVD(A^TA).V == SVD(A^TA).U
A is mXn. SVD(A).V is nXn. SVD(A).U is mXm

A is (4N)x4
A^TA is 4x4
U is 4x4
V is 4x4

Camera Calibration

http://16720.courses.cs.cmu.edu/lec/transformations.pdf

Given: real world scene points [$\mathbf{X_1}$, $\mathbf{X_2}$, $\mathbf{X_3}$, …]
and their corresponding projected points in the image plane
[$\mathbf{x_1}$, $\mathbf{x_2}$, $\mathbf{x_3}$,…] (which are the x,y points as pairs [{u_1, v_1), {u_2, v_2), {u_3, v_3),…])

Estimate: the camera matrix **M** that minimizes the distance between the actual points in the image, $\mathbf{x_i}$, and their predicted projections $\mathbf{MX_i}$

Problems:

- The projection is (in general) non-linear
- **M** is defined up to an arbitrary scale factor

$\mathbf{x_i = MX_i}$

non-linear relations between coordinates:

$$u_i = \frac{m_1^T \mathbf{X_i}}{m_3^T \mathbf{X_i}} \qquad v_i = \frac{m_2^T \mathbf{X_i}}{m_3^T \mathbf{X_i}}$$

make them linear:

$$m_1^T \mathbf{X_i} - (m_3^T \mathbf{X_i}) u_i = 0$$
$$m_2^T \mathbf{X_i} - (m_3^T \mathbf{X_i}) v_i = 0$$

Camera Calibration

http://16720.courses.cs.cmu.edu/lec/transformations.pdf
http://www.cs.cmu.edu/~16385/s17/Slides/11.3_Pose_Estimation.pdf

Write them in matrix form:

$$\begin{bmatrix} \mathbf{X_i}^T & 0 & -u_i\mathbf{X_i}^T \\ 0 & \mathbf{X_i}^T & -v_i\mathbf{X_i}^T \end{bmatrix} m = 0 \qquad m = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \end{bmatrix}$$

Put all the relations for all the points into a single matrix:

$$\begin{bmatrix} \mathbf{X_1}^T & 0 & -u_1\mathbf{X_1}^T \\ 0 & \mathbf{X_1}^T & -v_1\mathbf{X_1}^T \\ \boxed{?} & \boxed{?} & \boxed{?} \\ \mathbf{X_N}^T & 0 & -u_N\mathbf{X_N}^T \\ 0 & \mathbf{X_N}^T & -v_N\mathbf{X_N}^T \end{bmatrix} m = 0$$

(vector of 0's)

In noise-free case: $\mathbf{Lm} = 0$

In noisy case: $\min_{||m||^2=1} ||Lm||^2$ Min right singular vector of L
(or eigenvector of LTL)

or: minimize reproduction Error:

$$\left(u_i - \frac{m_1^T \mathbf{X_i}}{m_3^T \mathbf{X_i}}\right)^2 + \left(v_i - \frac{m_2^T \mathbf{X_i}}{m_3^T \mathbf{X_i}}\right)^2$$

Initialize nonlinear optimization with "algebraic" solution

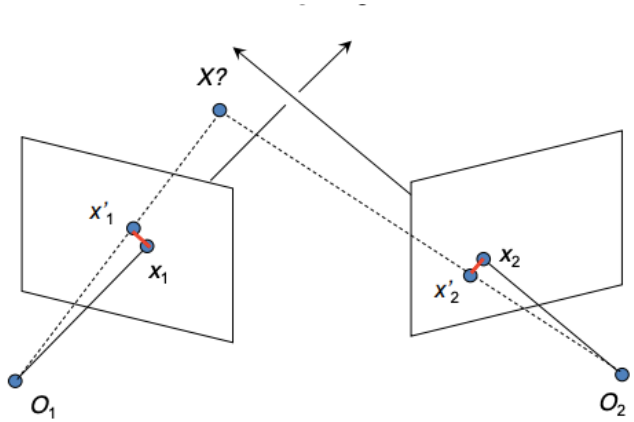
multiple Camera Calibration: bundle adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Bundle adjustment

change back to notation $\mathbf{x}_i$ for point (u_i, v_i) and use Proj():

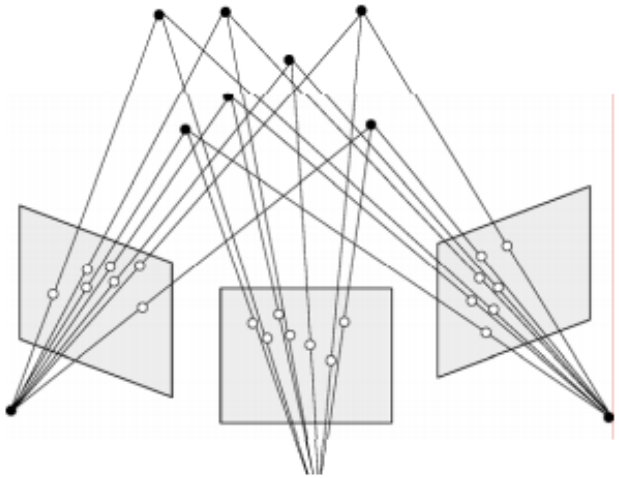
$$\left(u_i - \frac{m_1^T \mathbf{X}_i}{m_3^T \mathbf{X}_i}\right)^2 + \left(v_i - \frac{m_2^T \mathbf{X}_i}{m_3^T \mathbf{X}_i}\right)^2$$



$$\min_{\mathbf{X}} f(\mathbf{X}) = ||\mathbf{x}_1 - Proj(\mathbf{X}, M_1)||^2 + ||\mathbf{x}_2 - Proj(\mathbf{X}, M_2)||^2$$

For this equation, its easier to parameterize camera position in absolute coordinates instead of relative ones

Generalize triangulation to multiple points from multiple cameras



$$\min_{\mathbf{X}_1, \mathbf{X}_2, \ldots} \sum_{i=1}^{m} \sum_{j=1}^{n} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

multiple Camera Calibration: bundle adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

basic structure from motion (SFM) pipeline:

1. Find candidate correspondences (interest points + descriptor matches)
2. Select subset that are consistent with epipolar constraints on pairs of images (RANSAC + fundmental matrix)
3. Solve for 3D points and camera that minimize reprojection error

(Lots of variants; e.g., iteratively build map of 3D points and cameras as new images arrive)

$$\min_{\mathbf{X}_1,\mathbf{X}_2,\ldots,M_1,M_2,\ldots} \sum_{i=1}^{m}\sum_{j=1}^{n} ||\mathbf{x}_{ij} - Proj(\mathbf{X}_j, M_i)||^2$$

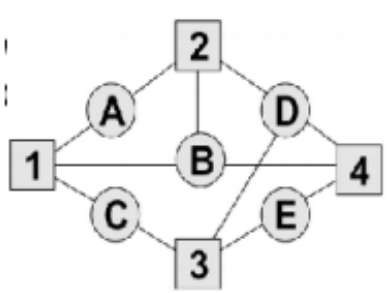
multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

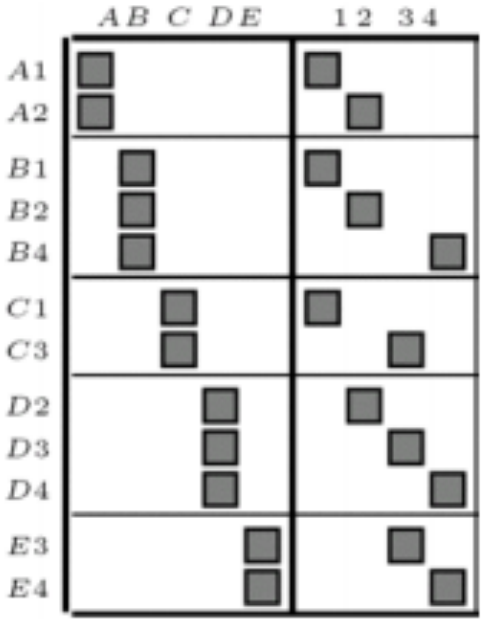
Bundle adjustment: nonlinear least-squares

Encode visibility graphs of what points are seen in what images, i.e.

- Features: A,B,C,D,E
- Images: 1,2,3



Implies jacobian of error function is sparse



Bundle Adjustment — A Modern Synthesis

Bill Triggs[1], Philip McLauchlan[2], Richard Hartley[3] and Andrew Fitzgibbon[4]

[1] INRIA Rhône-Alpes, 655 avenue de l'Europe, 38330 Montbonnot, France.
Bill.Triggs@inrialpes.fr ◇ *http://www.inrialpes.fr/movi/people/Triggs*

[2] School of Electrical Engineering, Information Technology & Mathematics
University of Surrey, Guildford, GU2 5XH, U.K.
P.McLauchlan@ee.surrey.ac.uk ◇ *http://www.ee.surrey.ac.uk/Personal/P.McLauchlan*

[3] General Electric CRD, Schenectady, NY, 12301
hartley@crd.ge.com

[4] Dept of Engineering Science, University of Oxford, 19 Parks Road, OX1 3PJ, U.K.
awf@robots.ox.ac.uk ◇ *http://www.robots.ox.ac.uk/ awf*

multiple Camera Calibration: bundle adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

SFM 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 2D homogenous vector remains exactly the same. It is impossible to recover the absolute scale of the scene.

$$\mathbf{x} \equiv M\mathbf{X}$$

$$M\mathbf{X} = (\frac{1}{k}M)(k\mathbf{X})$$

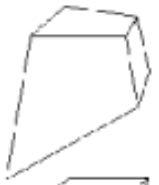
More generally: if we transform the scene using a transformation Q and apply the inverse transformation to the camera matrices, nothing changes

$$M\mathbf{X} = (MQ^{-1})(Q\mathbf{X})$$

Using notation P instead of M to emphasize Projection, can see:

- 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

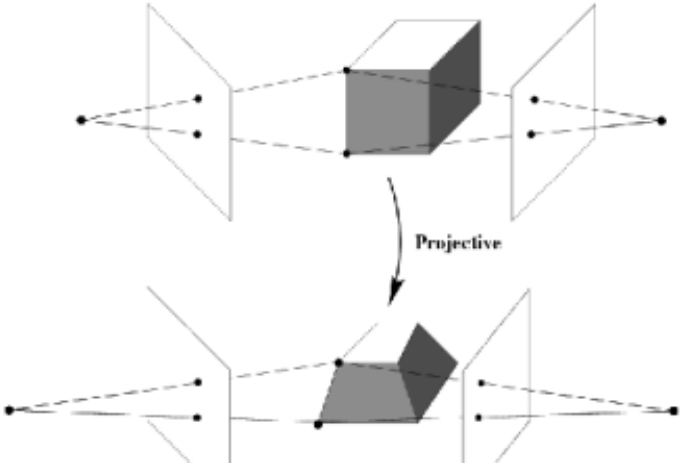
$$\mathbf{x} = \mathbf{PX} = \left(\mathbf{PQ}^{-1}\right)\left(\mathbf{QX}\right)$$

Projective	$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix}$		Preserves intersection
Affine	$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$		Preserves parallellism,
Similarity	$\begin{bmatrix} sR & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles, ratios
Euclidean	$\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix}$		Preserves angles.

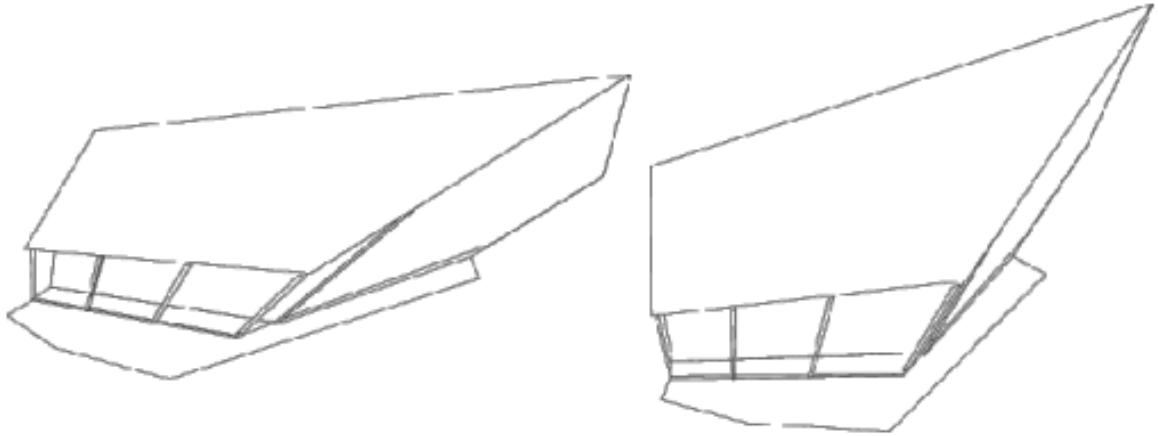
multiple Camera Calibration: bundle adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

SFM Projective ambiguity: $x \equiv M\mathbf{X} = (MQ^{-1})(Q\mathbf{X})$



$$\mathbf{Q}_p = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$$

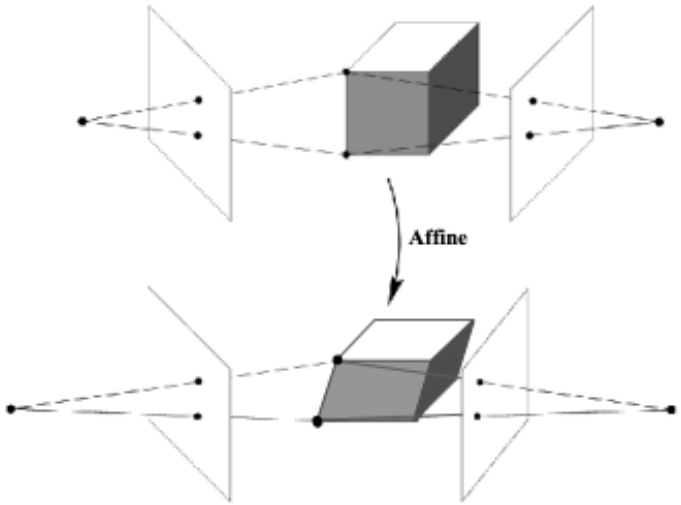


(straight line are preserved)

multiple Camera Calibration: bundle adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

SFM Affine ambiguity: $x \equiv M\mathbf{X} = (MQ^{-1})(Q\mathbf{X})$



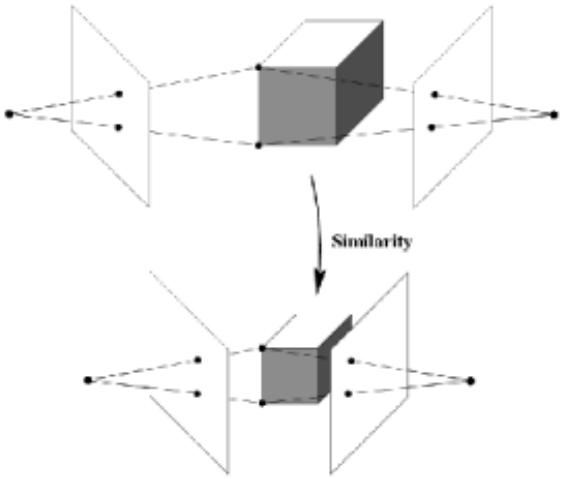
$$\mathbf{Q_A} = \begin{bmatrix} \mathrm{A} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$$

(parallel lines are preserved)

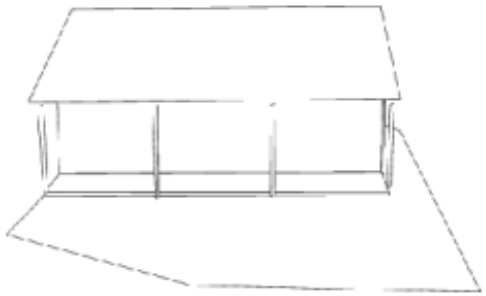
multiple Camera Calibration: bundle adjustment

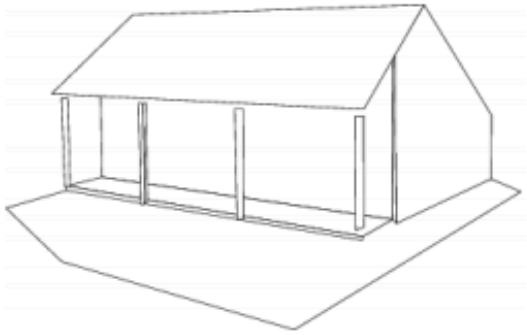
http://16720.courses.cs.cmu.edu/lec/sfm.pdf

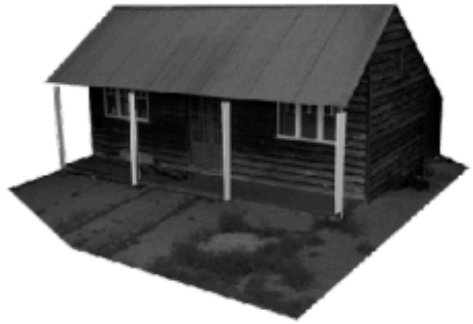
SFM Similarity ambiguity: $x \equiv M\mathbf{X} = (MQ^{-1})(Q\mathbf{X})$



$$\mathbf{Q_s} = \begin{bmatrix} s\mathrm{R} & \mathrm{t} \\ 0^{\mathsf{T}} & 1 \end{bmatrix}$$





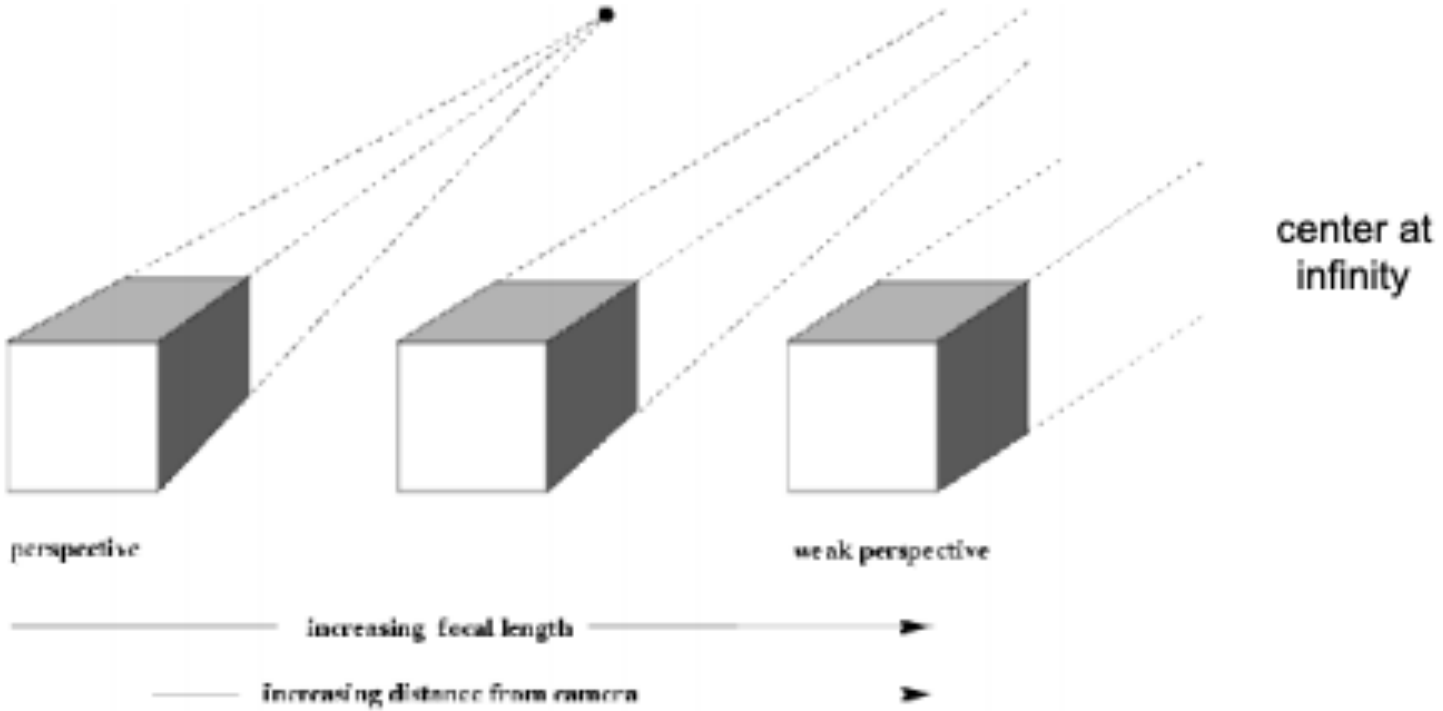


angles+lengths preserved (but can't recover world coordinate system)

multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Bundle adjustment: <u>Affine cameras</u> (the math is easier)



multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Bundle adjustment: Affine cameras (the math is easier)

Another common notation for a projection matrix:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

Sometimes notationally convenient because
2D homogenous coordinates allow us to write
2D transformations as matrix multiplication

Model as 3D affine transformation + orthographic projection + 2D affine transformation

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

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ & & & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Affine camera defined by 8 parameters

multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Bundle adjustment: <u>Affine cameras</u> (the math is easier)

$$\begin{bmatrix} x \\ y \end{bmatrix} = \left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ \hline & & & 1 \end{array}\right] Q_a Q_a^{-1} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad Q_a = \left[\begin{array}{ccc|c} \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \hline & & & 1 \end{array}\right]$$

$$= \left[\begin{array}{ccc} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{A}\mathbf{X} + \mathbf{b}$$

2D points = linear transformation of 3D points + 2D translation

Q_a must be 3D affine transformation (12 parameters) inorder to maintain structure of affine projection matrix

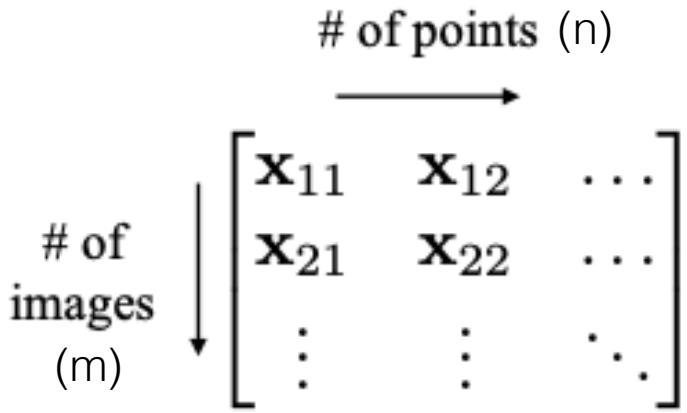
multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Affine SFM: Algorithm Summary (Dr. Deva Ramanan's favorite algorithm)

Given: m images of n fixed 3D points, use the m*n correspondences x_{ij} to estimate m projection matrices A_i and translation vectors b_i , and n points X_j

- Given: *m* images and *n* features $\mathbf{x}_{ij}$
- For each image *i*, center the feature coordinates
- Construct a $2m \times n$ measurement matrix **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 **D**:
  - Compute SVD: $\mathbf{D} = \mathbf{U}\,\mathbf{W}\,\mathbf{V}^T$
  - Create $\mathbf{U}_3$ by taking the first 3 columns of **U**
  - Create $\mathbf{V}_3$ by taking the first 3 columns of **V**
  - Create $\mathbf{W}_3$ by taking the upper left 3×3 block of **W**
- Create the motion and shape matrices:
  $M = U_3, \quad S = \Sigma_3 V_3^T$
- Eliminate affine ambiguity



multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Affine SFM: Algorithm Summary: (given m images and n features…)

- The reconstruction is defined up to an arbitrary 3D *affine* transformation **Q** (12 degrees of freedom):

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

- We have $2mn$ knowns and $8m + 3n$ unknowns (minus 12 dof for affine ambiguity)
- Thus, we must have $2mn >= 8m + 3n - 12$
- For m=2 views, we need n=4 point correspondences

# of points (n) →

# of images (m) ↓

$$\begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \dots \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

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

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

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

subtracts the centroid of image points

multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Affine SFM: Algorithm Summary: (given m images and n features…)

Now each normalized point x_{ij} is related to the 3D point $\mathbf{X_i}$ by

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

Given a set of 2D correspondences:
center them in each image
Affine image projection now becomes linear:
(represent A_i by a 2x3 matrix and $x\hat{}_{ij}$ is 2 vector)

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

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

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

The measurement matrix $\mathbf{D} = \mathbf{MS}$ must have rank 3!

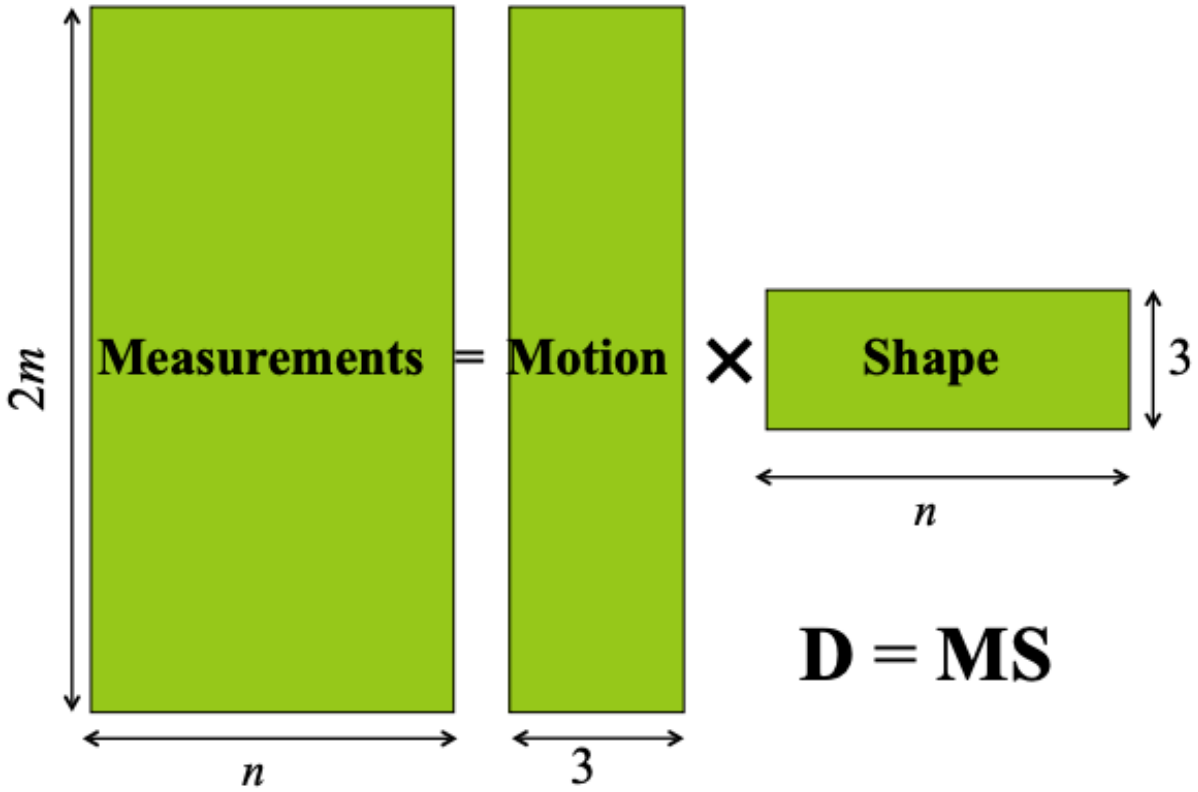
multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Affine SFM: Algorithm Summary: (given m images and n features…)

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

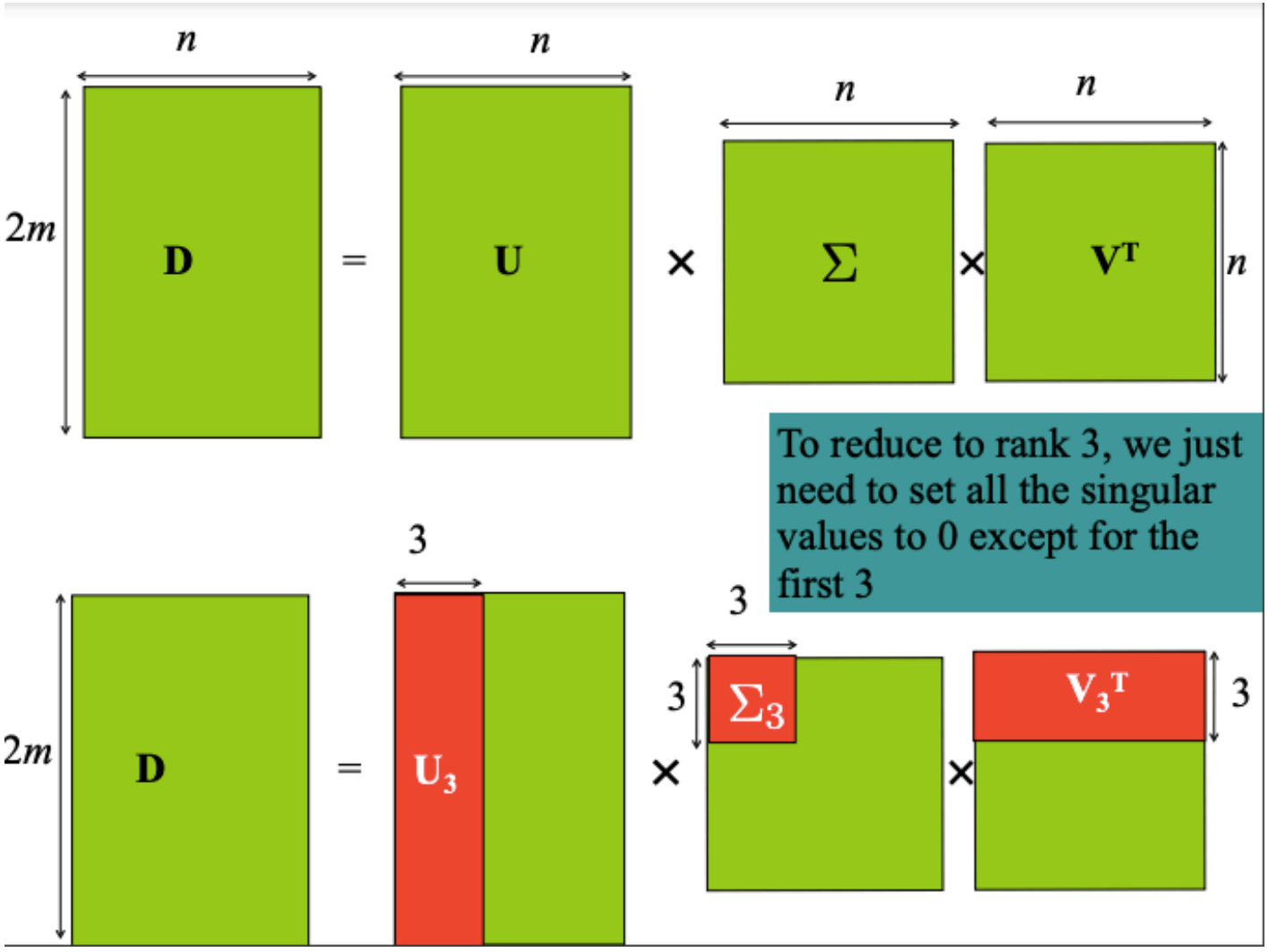
Fundamental Decomposition



multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

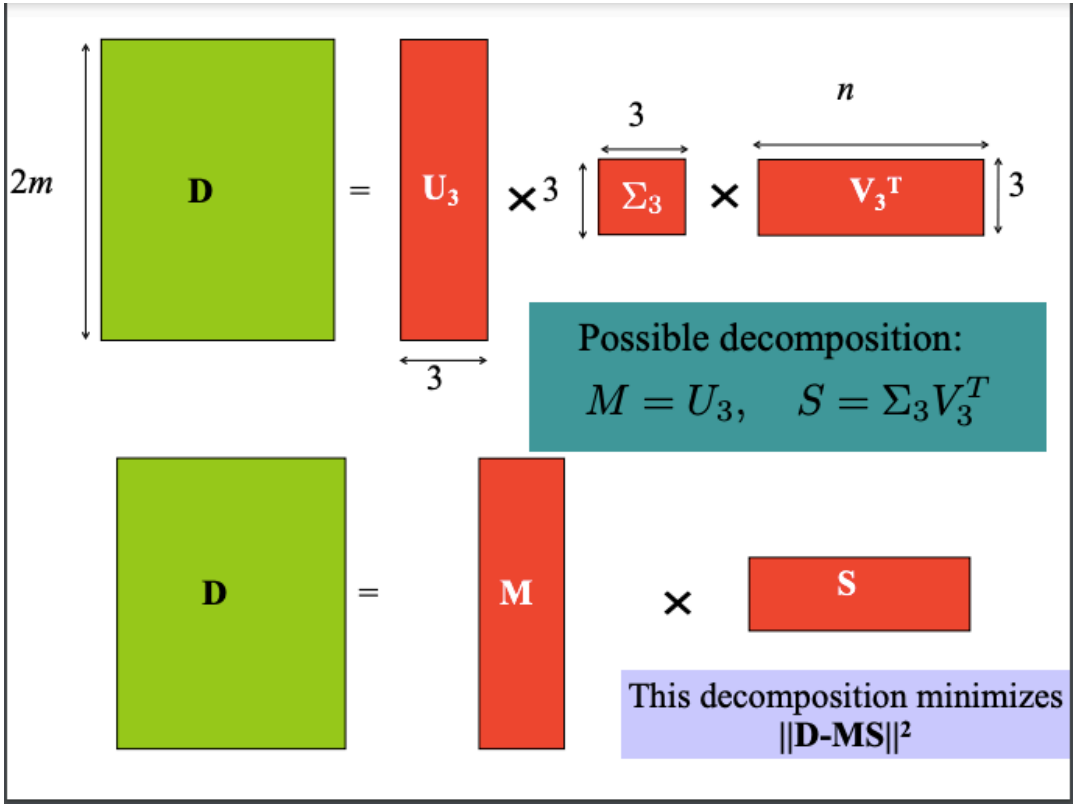
Affine SFM: <u>Algorithm Summary</u> (Dr. Deva Ramanan's favorite algorithm)



multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Affine SFM: <u>Algorithm Summary</u> (Dr. Deva Ramanan's favorite algorithm)



- The decomposition is not unique. We get the same **D** by using any 3×3 matrix **C** and applying the transformations **M** → **MC**, **S** → **C**$^{-1}$**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)

multiple Camera Calibration: Bundle Adjustment

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

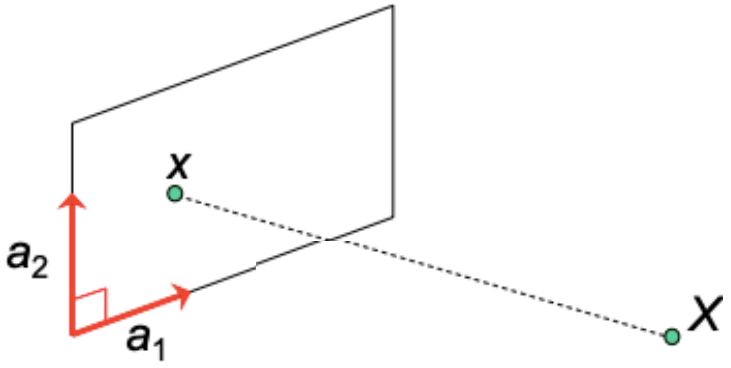
Affine SFM: Algorithm Summary (Dr. Deva Ramanan's favorite algorithm)

Eliminating the affine ambiguity

enforce image axes to be ortthonogal and length 1

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

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



$$M = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \end{bmatrix} \quad \text{from} \quad \mathbf{D} = \underbrace{\begin{bmatrix} \hat{\mathbf{x}}_{11} & \hat{\mathbf{x}}_{12} & \cdots & \hat{\mathbf{x}}_{1n} \\ \hat{\mathbf{x}}_{21} & \hat{\mathbf{x}}_{22} & \cdots & \hat{\mathbf{x}}_{2n} \\ & & \ddots & \\ \hat{\mathbf{x}}_{m1} & \hat{\mathbf{x}}_{m2} & \cdots & \hat{\mathbf{x}}_{mn} \end{bmatrix}}_{\text{points } (n)} \quad \begin{matrix} \text{cameras} \\ (2m) \end{matrix} = \underbrace{\begin{bmatrix} \mathbf{A}_1 \\ \mathbf{A}_2 \\ \vdots \\ \mathbf{A}_m \end{bmatrix}}_{\begin{matrix} \text{cameras} \\ (2m \times 3) \end{matrix}} \underbrace{\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \cdots & \mathbf{X}_n \end{bmatrix}}_{\text{points } (3 \times n)}$$

We want C such that $A_i\, CC^T A^T_i = Id$

let $L = C\, C^T$, then optimization is $\min_L ||A_i L A_i^T - Id||^2$

This translates into $3m$ equations in $\mathbf{L} = \mathbf{CC}^T$:

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

- Solve for **L**
- Recover **C** from **L** by Cholesky decomposition: $\mathbf{L} = \mathbf{CC}^T$ (in practice, easy to do)
- Update **M** and **S**: $\mathbf{M} = \mathbf{MC}$, $\mathbf{S} = \mathbf{C}^{-1}\mathbf{S}$

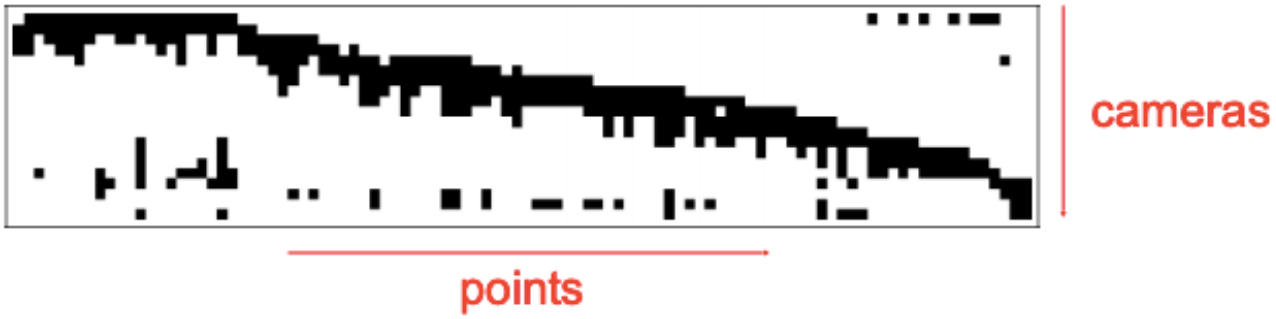
Source: M. Hebert

multiple Camera Calibration: Bundle Adjustment Extensions

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

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:



Sequential structure from motion

- Intuition: exploit low-rank redundancy of matrix
- 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

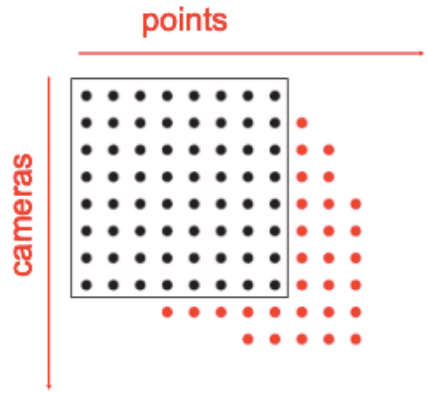
multiple Camera Calibration: Bundle Adjustment Extensions

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Sequential structure from motion

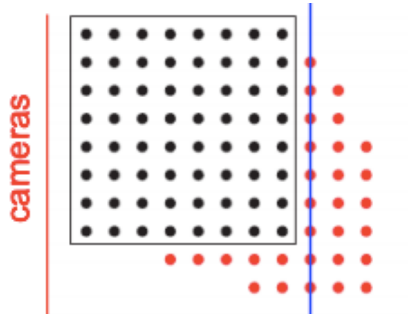
F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce.
Segmenting, Modeling, and Matching Video Clips Containing Multiple Moving Objects.
PAMI 2007.

(1) Perform factorization on a dense sub-block



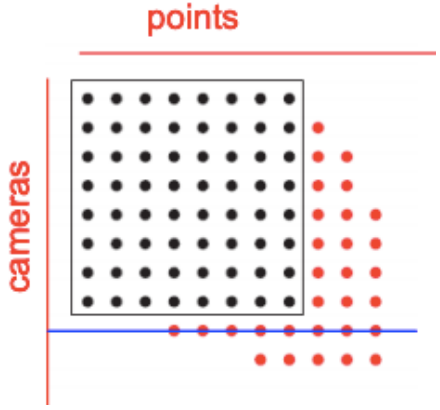
(2) Solve for a new 3D point visible by at least two known cameras (linear least squares)

$$\min_S ||D - MS||^2$$



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

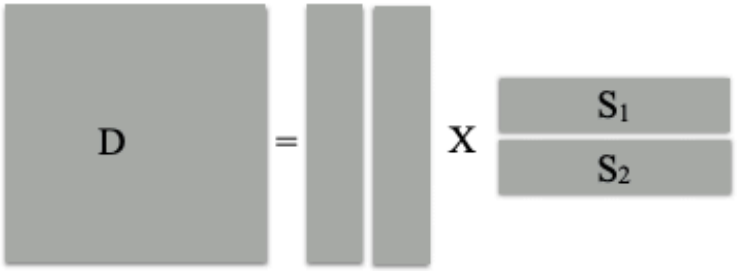
$$\min_M ||D - MS||^2$$



multiple Camera Calibration: Bundle Adjustment Extensions

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Nonrigid structure motion



Assume 3D shapes are a linear combination of K basis shapes

$$S = \sum_{k=1}^{K} \alpha_k S_k$$

For each image, we need to solve for camera matrix *and* scaling coefficients

Simply relax rank constraint on measurement matrix from 3 to 3K!

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Projective structure from motion

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

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

- Problem: estimate *m* projection matrices $\mathbf{M}_i$ 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**:

$$\mathbf{X} \rightarrow \mathbf{QX}, \mathbf{M} \rightarrow \mathbf{MQ}^{-1}$$

- We can solve for structure and motion when

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

- For two cameras, at least 7 points are needed

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

Projective SFM: Two-camera case

- Compute fundamental matrix **F** between the two views (from 7 or more correspondences)
- First camera matrix: M = **[I|0]**
- Second camera matrix: M' = $[\mathbf{A}_{3x3}|\mathbf{b}_{3x1}]$
- Then one can compute **A,b** from F as follows:
  - **b** is the right null vector, or epipole ($\mathbf{F}^T\mathbf{b} = 0$)

$A = -\hat{\mathbf{b}}F$ (where hat notation refers to skew symmetric matrix)

For proof, see F & P Sec 8.3

(Forsyth & Ponce, "Computer Vision. A Modern Approach")

For more than 2 images, things get more implicated

http://16720.courses.cs.cmu.edu/lec/sfm.pdf

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 $\mathbf{M}_i = \mathbf{K}\,[\mathbf{R}_i \mid \mathbf{t}_i]$
- Can use constraints on the form of the calibration matrix: orthogonal image axis
- Can use vanishing points

Reconstruction (2 view Structure From Motion, SFM)

http://www.cs.cmu.edu/~16385/s17/Slides/12.5_Reconstruction.pdf
Szeliski 2010, Chap 7
Ma, Soatto, Kosecká, and Sastry 2012, "An Invitation to 3-D Vision", pg 121

Given a set of (noisy) matched points $\{\boldsymbol{x}_i, \boldsymbol{x}'_i\}$

Estimate the camera matrices $\mathbf{P}, \mathbf{P}'$ ← 'motion'

Estimate the 3D point $\mathbf{X}$ ← 'structure'

---

(1) compute the fundamental matrix
(2) compute the camera matrices P and P' from the fundamental matrix. (see Hartely and Zisserman 2004 "Multiple View Geometry in Computer Vision ", and code vgg_P_from_F.m their website http://www.robots.ox.ac.uk/~vgg/hzbook/code/ which has several authors to acknowledge.

P = [I | 0] and **P' = [[e']_x F | e']** <—- Hartley et al. use P' = [-1*[e]_x*F | e]

If have intrinsic camera matrices K and K':

Essential matrix: $\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K}$

SVD: $\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ → FOUR solutions: $\mathbf{E} = [\mathbf{R}|\mathbf{T}]$

Let $\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$\mathbf{R}_{1,2} = \pm\mathbf{U}\mathbf{W}\mathbf{V}^{\top}$ $\mathbf{R}_{3,4} = \pm\mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top}$ ~~two possible rotations~~

$\mathbf{T}_1 = U_3$ $\mathbf{T}_2 = -U_3$ two possible translations

*Two of 4 possible rotations (keep the 2 w/ det(R)=1)
*Of the 4 combinations, keep the one with the most postive Z value in triangulation results. +Z is in front of camera.

Reconstruction (2 view Structure From Motion, SFM)

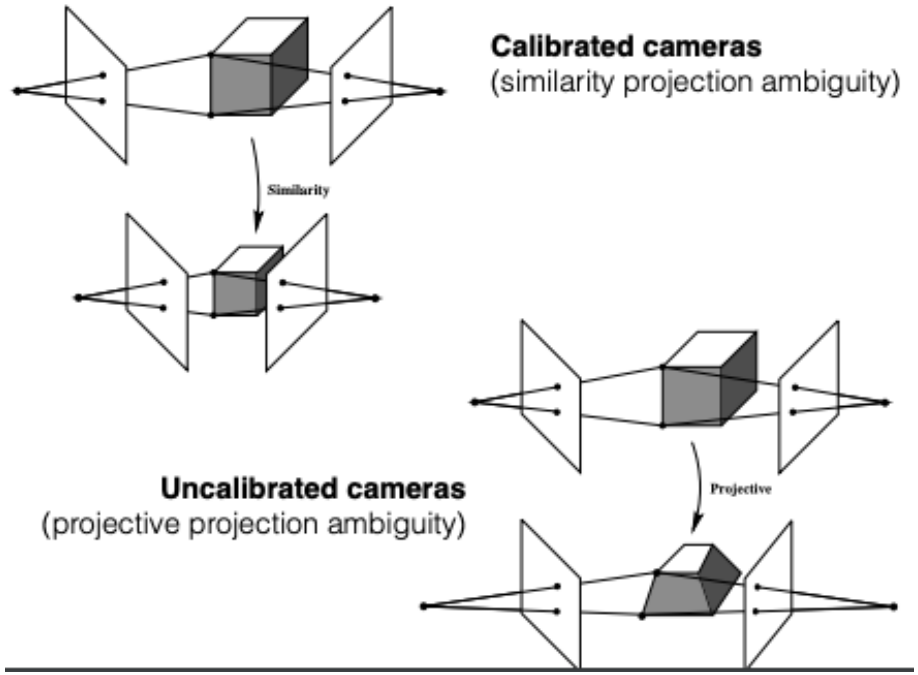
http://www.cs.cmu.edu/~16385/s17/Slides/12.5_Reconstruction.pdf

From points correspondences to camera displacement

1. Normalize the image points **x,x'** using **K,K'**
2. Use the 8-point algorithm to find an approximation of **E** (SVD!)
3. Project **E** to essential space (SVD!!)
   (set smallest SV to zero)
4. Recover possible solutions for **R** and **T** (SVD!!!)
5. Use point correspondence to find the correct **R,T** pair (don't use SVD…)

Procedure for Reconstruction

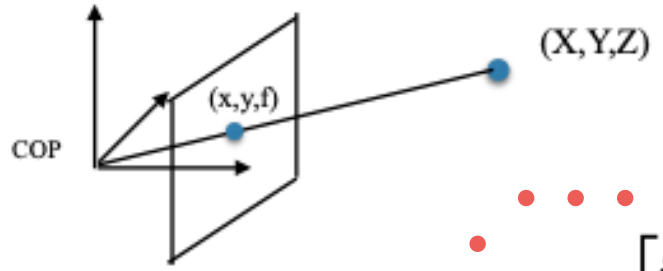
1. Compute the Fundamental Matrix **F** from points correspondences
   8-point algorithm
2. Compute the camera matrices **P** from the Fundamental matrix
   $\mathbf{P} = [\,\mathbf{I} \mid \mathbf{0}\,]$ and $\mathbf{P'} = [\,[\mathbf{e'}_x]\mathbf{F} \mid \mathbf{e'}\,]$
3. For each point correspondence, compute the point **X** in 3D space (triangularization)
   DLT with $\mathbf{x} = \mathbf{P}\,\mathbf{X}$ and $\mathbf{x'} = \mathbf{P'}\,\mathbf{X}$



Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Projecting from camera coordinate system to image coordinates

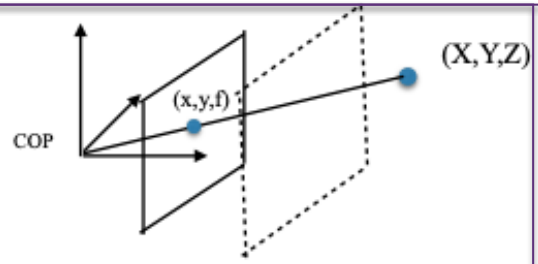


$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$\lambda \mathbf{x} = K\mathbf{X}$$

$$K = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Projecting from camera coordinate system to *normalized* image coordinates



If K is known, work with warped image

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

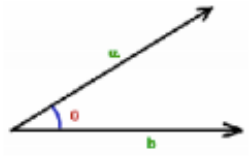
$$\lambda \mathbf{x}' = \mathbf{X}$$

To simplify notation, we'll use **x** instead of **x'**

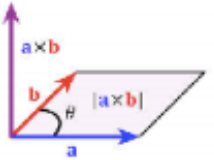
Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Dot product: $\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}||\ ||\mathbf{b}||cos\theta$



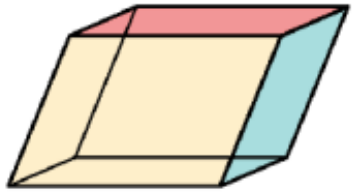
Cross product: $\mathbf{a} \times \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \sin\theta\ \mathbf{n}$



Cross product matrix: $\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \equiv \hat{\mathbf{a}}\mathbf{b}$

Important property (skew symmetric): $\hat{\mathbf{a}}^T = -\hat{\mathbf{a}}$

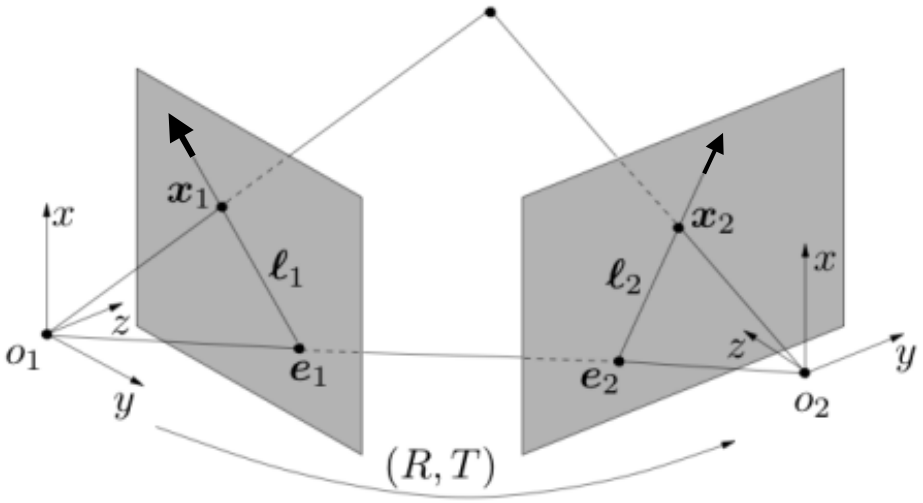
$\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})$ = volume of parallelpiped
= 0 for coplanar vectors



Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Calibrated 2-view geometry



$$\boxed{\boldsymbol{X}_2 = R\boldsymbol{X}_1 + \boldsymbol{T}}$$

$$\boldsymbol{X}_1 = \lambda_1 \boldsymbol{x}_1, \quad \boldsymbol{X}_2 = \lambda_2 \boldsymbol{x}_2$$

$$\boxed{\boldsymbol{X}_2 = R\boldsymbol{X}_1 + \boldsymbol{T}}$$

$$\boldsymbol{X}_1 = \lambda_1 \boldsymbol{x}_1, \quad \boldsymbol{X}_2 = \lambda_2 \boldsymbol{x}_2$$

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + \boldsymbol{T}$$

Take (left) cross product of both sides with T

$$\lambda_2 \widehat{T}\boldsymbol{x}_2 = \widehat{T}R\lambda_1 \boldsymbol{x}_1 + \underbrace{\widehat{T}T}_{=\mathbf{0}}$$

Take (left) dot product of both sides with $\boldsymbol{x}_2$

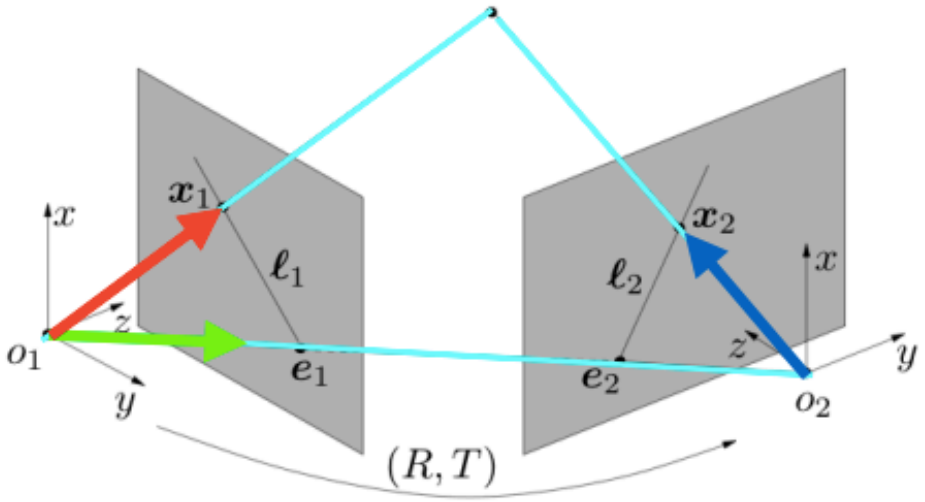
$$\lambda_2 \underbrace{\boldsymbol{x}_2^\top \widehat{T}\boldsymbol{x}_2}_{=0} = \boldsymbol{x}_2^\top \widehat{T}R\lambda_1 \boldsymbol{x}_1$$

$$\boldsymbol{x}_2^\top \widehat{T}R\boldsymbol{x}_1 = 0$$

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Geometric derivation



Simply the coplanar constraint applied to 3 vectors from camera 2's coordinate system

$$\mathbf{x}_2 \cdot (\mathbf{T} \times R\mathbf{x}_1) = 0$$

translation vector = epipole in right image (in homogenous coordinates)

$$\boldsymbol{x}_2^\top \widehat{T} R \boldsymbol{x}_1 = 0$$ ← **T** behaves like a cross-product (skew symmetric matrix)

$$\boxed{\boldsymbol{x}_2^\top E \boldsymbol{x}_1 = 0}$$ E is *essential* matrix

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

In uncalibrated case, we need to account for camera intrinsics:

$$\lambda \mathbf{x} = K\mathbf{X}$$

$$E = \hat{T}R$$

$$F = {K_2}^{-T} E K_1^{-1}$$

given E, can recover R and T, but need to distinguish the valid solution among 4 geometries

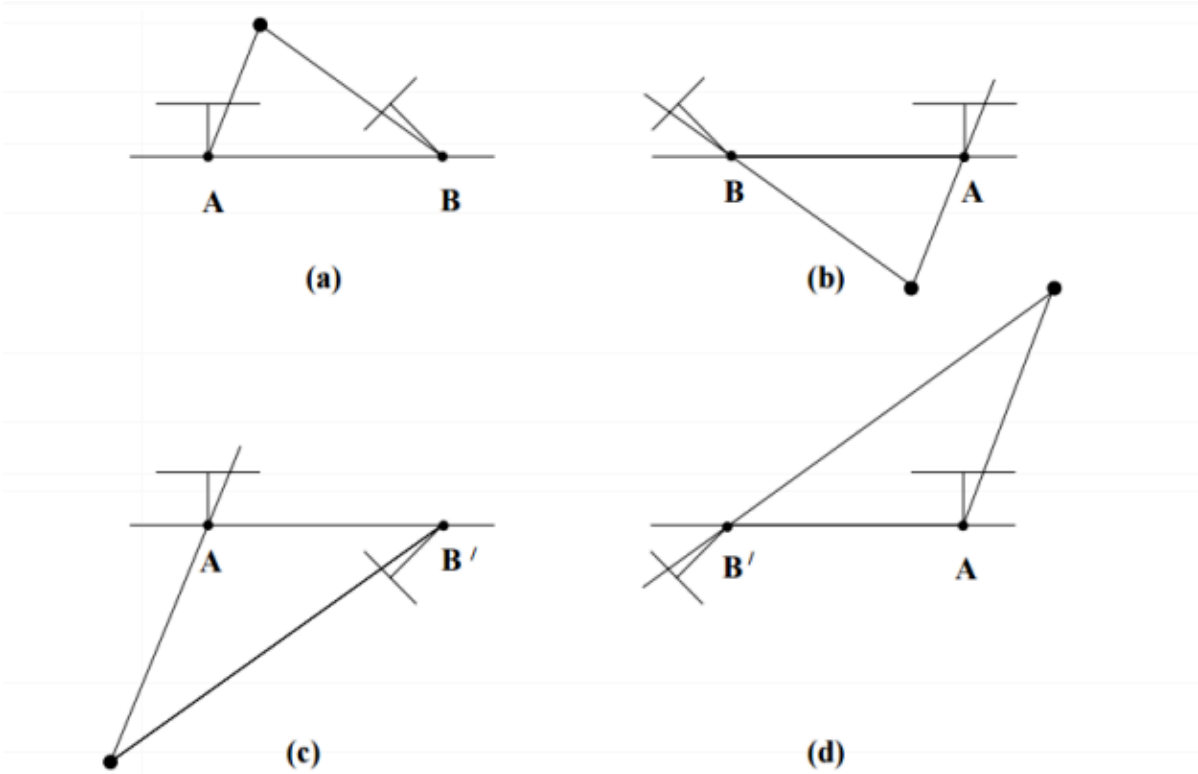


Fig. 8.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

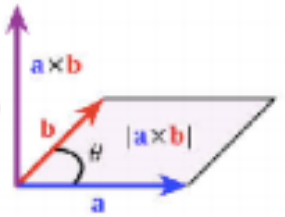
Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Background: SVDs of skew symmetric matrices

Any skew-symmetric matrice ($A = -A^T$) can be thought of as a cross-product

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{bmatrix} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \equiv \hat{\mathbf{a}}\mathbf{b}$$



SVD of a skew-symmetric matrix:

$$\hat{\mathbf{a}} = \begin{bmatrix} -\mathbf{e}_2 & \mathbf{e}_1 & \mathbf{e}_3 \end{bmatrix} \begin{bmatrix} ||a|| & 0 & 0 \\ 0 & ||a|| & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix} \quad \text{where } \mathbf{e}_3 = \mathbf{a} \,/\, ||\mathbf{a}||$$

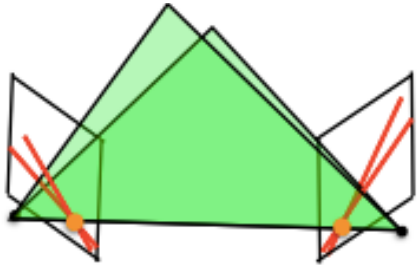
One singular value is 0 and the other two = $||\mathbf{a}||$

$$\hat{\mathbf{a}} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{bmatrix} \begin{bmatrix} ||a|| & 0 & 0 \\ 0 & ||a|| & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{e}_1^T \\ \mathbf{e}_2^T \\ \mathbf{e}_3^T \end{bmatrix}$$

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Recovering T,R from E



1. Universal scale ambiguity

Doubling T results in same epipolar lines

Let's fix $||\mathbf{T}|| = 1$

[notation switch] → unit-vector: $\hat{\mathbf{t}}$

skew-symmetric matrix: $\hat{\mathbf{t}}_\times$

Numerous methods for recovering **t**,R from E exist:
SVD, Louget-Higgen's alg, etc.

SVD-based approach for noise-free E (Szeliski Chap 7.2)

$$\boldsymbol{x}_2^\top \widehat{T} R \boldsymbol{x}_1 = 0$$

or, using skew-symmetric notation:

$\mathbf{x_2}^T\ [t_x]\ \mathbf{R}\ \mathbf{x_1} = 0$ and $\mathbf{E} = [t_x]\ \mathbf{R}$

take cross-product: $[t_x]^T\ \mathbf{E} = [t_x]^T\ [t_x]\ \mathbf{R}$

$[t_x]^T\ \mathbf{E} = 0$

translation vector = epipole in right image

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf
and from Szeliski 2010 textbook draft
"Computer Vision: Algorithms and Applications"

To estimate this direction $\hat{\boldsymbol{t}}$, observe that under ideal noise-free conditions, the essential matrix $\boldsymbol{E}$ is singular, i.e., $\hat{\boldsymbol{t}}^T \boldsymbol{E} = 0$. This singularity shows up as a singular value of 0 when an SVD of $\boldsymbol{E}$ is performed,

$$\boldsymbol{E} = [\hat{\boldsymbol{t}}]_\times \boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \hat{\boldsymbol{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^T \\ \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \end{bmatrix} \tag{7.18}$$

When $\boldsymbol{E}$ is computed from noisy measurements, the singular vector associated with the smallest singular value gives us $\hat{\boldsymbol{t}}$. (The other two singular values should be similar but are not, in general, equal to 1 because $\boldsymbol{E}$ is only computed up to an unknown scale.)

Set translation direction = smallest left singular vector of E

But we can't distinguish E from -E, so we only know direction up to a sign

Aside: $\mathbf{v}_2$ = epipole in left image

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf
and from Szeliski 2010 textbook draft
"Computer Vision: Algorithms and Applications"

Unfortunately, we only know both $\boldsymbol{E}$ and $\hat{\boldsymbol{t}}$ up to a sign. Furthermore, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are not guaranteed to be rotations (you can flip both their signs and still get a valid SVD). For this reason, we have to generate all four possible rotation matrices

$$\boldsymbol{R} = \pm \boldsymbol{U} \boldsymbol{R}_{\pm 90^\circ}^T \boldsymbol{V}^T \tag{7.25}$$

and keep the two whose determinant $|\boldsymbol{R}| = 1$. To disambiguate between the remaining pair of potential rotations, which form a *twisted pair* (Hartley and Zisserman 2004, p. 240), we need to pair them with both possible signs of the translation direction $\pm\hat{\boldsymbol{t}}$ and select the combination for which the largest number of points is seen in front of both cameras.[4]

where $\boldsymbol{R}_{90^\circ} = \begin{bmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{bmatrix}$

The property that points must lie in front of the camera, i.e., at a positive distance along the viewing rays emanating from the camera, is known as *chirality* (Hartley 1998). In addition to determining the signs of the rotation and translation, as described above, the chirality (sign of the distances) of the points in a reconstruction can be used inside a RANSAC procedure (along with the reprojection errors) to distinguish between likely and unlikely configurations.[5] Chirality can also be used to transform projective reconstructions (Sections 7.2.1 and 7.2.2) into *quasi-affine* reconstructions (Hartley 1998).

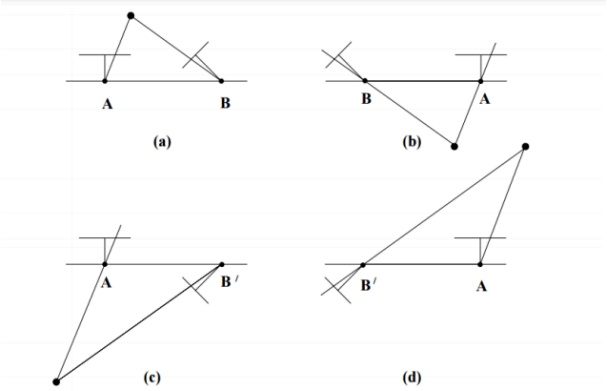
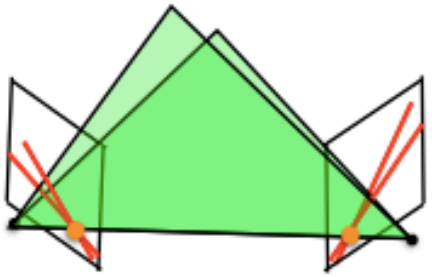


Fig. 8.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Essential and Fundamental Matrices



$$E = \hat{T}R$$

$$\mathbf{x}_2^T E \mathbf{x}_1 = 0$$

$$E = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \mathbf{e}_2 \end{bmatrix} \begin{bmatrix} \sigma & & \\ & \sigma & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{e}_1^T \end{bmatrix}$$

"Proof": properties of skew-symmetric matrices

$$F = K_2^{-T} E K_1^{-1}$$

$$\mathbf{x}_2^T F \mathbf{x}_1 = 0$$

$$F = \begin{bmatrix} \mathbf{u}_0 & \mathbf{u}_1 & \mathbf{e}_2 \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_0^T \\ \mathbf{v}_1^T \\ \mathbf{e}_1^T \end{bmatrix}$$

Proof: scale ambiguity

where e1, e2 are epipoles in right and left images

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Formal characterizations

Ma et al, An Invitation to 3D Vision

Theorem 5.1 (Characterization of the essential matrix). *A non-zero matrix $E \in \mathbb{R}^{3\times 3}$ is an essential matrix* if and only if *E has a singular value decomposition (SVD): $E = U\Sigma V^T$ with*

$$\Sigma = diag\{\sigma, \sigma, 0\}$$

for some $\sigma \in \mathbb{R}_+$ and $U, V \in SO(3)$.

SO(3) is special the orthogonal group of proper n-dimensional rotation matrices where n=3.
an improper rotation matrix requires mirrors
http://scipp.ucsc.edu/~haber/ph251/rotreflect_17.pdf

Remark 6.1. *Characterization of the fundamental matrix. A non-zero matrix $F \in \mathbb{R}^{3\times 3}$ is a fundamental matrix if F has a singular value decomposition (SVD): $E = U\Sigma V^T$ with*

$$\Sigma = diag\{\sigma_1, \sigma_2, 0\}$$

for some $\sigma_1, \sigma_2 \in \mathbb{R}_+$.

Reconstruction (2 view Structure From Motion, SFM)

a few more details on estimating R and T:
http://16720.courses.cs.cmu.edu/lec/two-view_lec15.pdf

Rectification

http://www.cs.cmu.edu/~16385/s17/Slides/13.1_Stereo_Rectification.pdf

Stereo Rectification Algorithm

1. Estimate **E** using the 8 point algorithm (SVD)
2. Estimate the epipole **e** (SVD of **E**)
3. Build $\mathbf{R}_{rect}$ from **e** ← see next slide
4. Decompose **E** into **R** and **T** ← see prev slides
5. Set $\mathbf{R}_1=\mathbf{R}_{rect}$ and $\mathbf{R}_2 = \mathbf{R}\mathbf{R}_{rect}$
6. Rotate each left camera point (warp image)
   $[x' \ y' \ z'] = \mathbf{R}_1 [x \ y \ z]$
7. Rectified points as $\mathbf{p} = f/z'[x' \ y' \ z']$
8. Repeat 6 and 7 for right camera points using $\mathbf{R}_2$

6. Rotate each left camera point $\mathbf{x'} \sim \mathbf{Hx}$ where $\mathbf{H} = \mathbf{K}\mathbf{R}_1$

*You may need to alter the focal length (inside **K**) to keep points within the original image size

Rectification

http://www.cs.cmu.edu/~16385/s17/Slides/13.1_Stereo_Rectification.pdf

Building Rrect from epipole e (i.e. the left epipole)

Let $R_{\text{rect}} = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$ Given: epipole **e** (using SVD on E) (translation from **E)**

$$\boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||}$$ epipole coincides with translation vector

$$\boldsymbol{r}_2 = \frac{1}{\sqrt{T_x^2 + T_y^2}} \begin{bmatrix} -T_y & T_x & 0 \end{bmatrix}$$ cross product of e and the direction vector of the optical axis

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2$$ orthogonal vector

If $\boldsymbol{r}_1 = \boldsymbol{e}_1 = \frac{T}{||T||}$ and $\boldsymbol{r}_2$ $\boldsymbol{r}_3$ orthogonal

then $R_{\text{rect}} \boldsymbol{e}_1 = \begin{bmatrix} \boldsymbol{r}_1^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_2^\top \boldsymbol{e}_1 \\ \boldsymbol{r}_3^\top \boldsymbol{e}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

Where is this point located on the image plane?

At x-infinity

Note that the rotations are explained as 2 steps by Monasse, Morel, and Tang 2011 (though their algorithm is only for epipoles are outside the image domain).

1. Compute homographies H_1 and H′_1 by rotating both cameras respectively so that the **left epipole (e_x, e_y, 1) is transformed to (e_x, e_y, 0) and the right epipole (e′_x, e′_y ,1) to (e′_x, e′_y ,0)**.

2. Rotate both cameras so that **(e_x, e_y, 1) is transformed to (1,0,0) and (e′_x, e′_y ,1) to (1,0,0)**. The corresponding homographies are denoted by H_2 and H′_2 .

3. Rotate one camera or both cameras together to compensate the residual relative rotation between both cameras around the baseline. The corresponding homographies are denoted by H_3 and H′_3 .

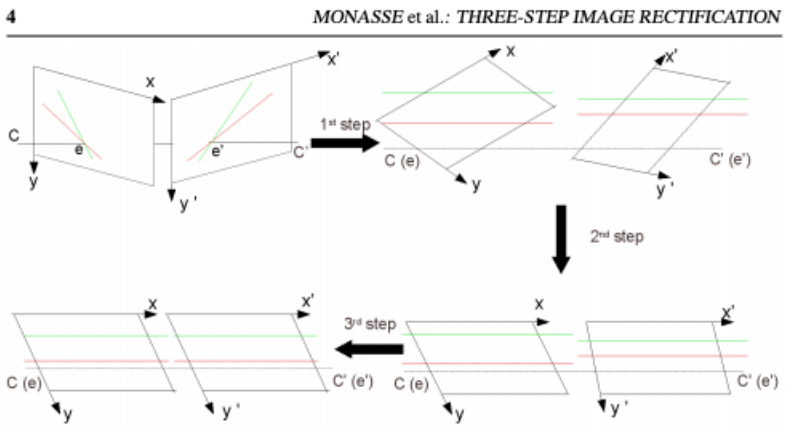


Figure 1: Three-step rectification. First step: the image planes become parallel to CC'. Second step: the images rotate in their own plane to have their epipolar lines also parallel to CC'. Third step: a rotation of one of the image planes around CC' aligns corresponding epipolar lines in both images. Note how the pairs of epipolar lines become aligned.

Bundle Adjustment

from “Bundle Adjustment — A Modern Synthesis”
by Triggs, McLauchlan, Hartley, and Fitzgibbon
Vision Algorithms’99, LNCS 1883, pp. 298–372, 2000
http://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Triggs00.pdf
(Note, consult author for any use of figures and text - it’s presented here for educational purposes only.)

Pattern Matrices and Factorization:
The most common ***top-down*** method is called nested dissection or recursive partitioning: recursively split the factorization problem into smaller sub-problems, solve these independently, and then glue the solutions together along their common boundaries. Splitting involves choosing a separating set of variables, whose deletion will separate the remaining variables into two or more independent subsets. This corresponds to finding a (vertex) graph cut of the elimination graph, i.e. a set of vertices whose deletion will split it into two or more disconnected components.
Given such a partitioning, the variables are reordered into ***connected components***, with the separating set ones last. This produces a pattern matrix such as ‘block tridiagonal matrix’, ’arrowhead matrix’, or ‘bundle hessian’, etc.
**Finding a globally minimal partition sequence is NP complete but several effective heuristics exist.

Many ***bottom-up*** variable ordering heuristics exist.

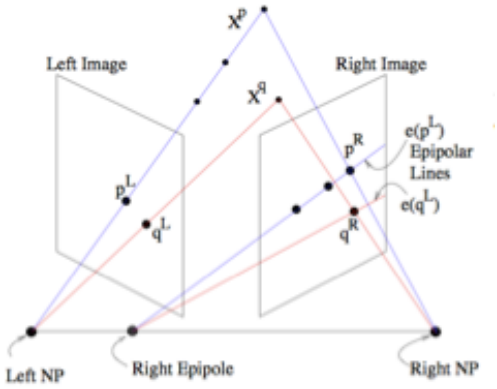
Deciding the reconstructed coordinate system is in the realm of **Gauge Freedom (**a gauge is a reference frame**)**:
constraining the coordinate values of certain aspects of the reconstructed structure — features, cameras or combinations of these — whatever the rest of the structure might be.

Bundle Adjustment

from “Bundle Adjustment in the Large”, 2010
by Agarwal, Snavely, Seitz, and Szeliski
https://homes.cs.washington.edu/~sagarwal/bal.pdf
(Note, consult author for any use of figures and text - it’s presented here for educational purposes only.)

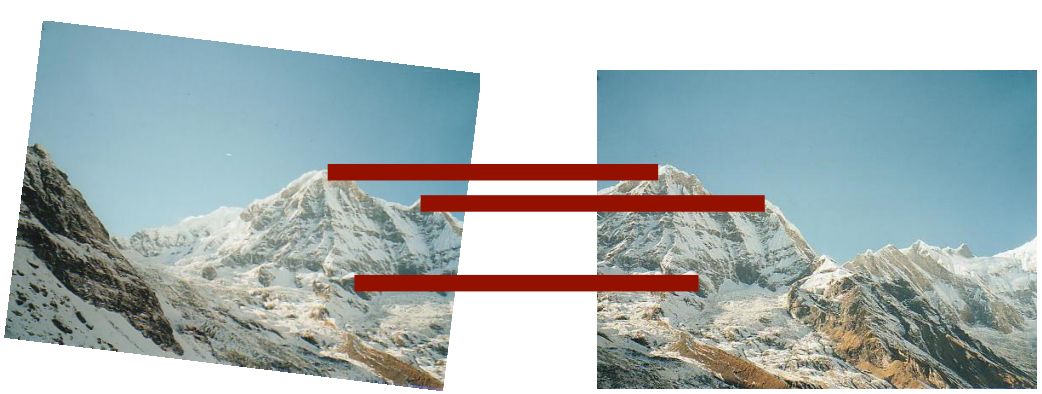
treated as re-weighted non-linear least squares problem .

Projection, stereo and panoramic images



http://www.cs.toronto.edu/~jepson/csc420/notes/epiPolarGeom.pdf
(note, the image is posted on an educational site and copied here without following up on permissions. Any further use of the image should follow up on the origins and permissions.)

panoramic images here are from Brown & Lowe 2003

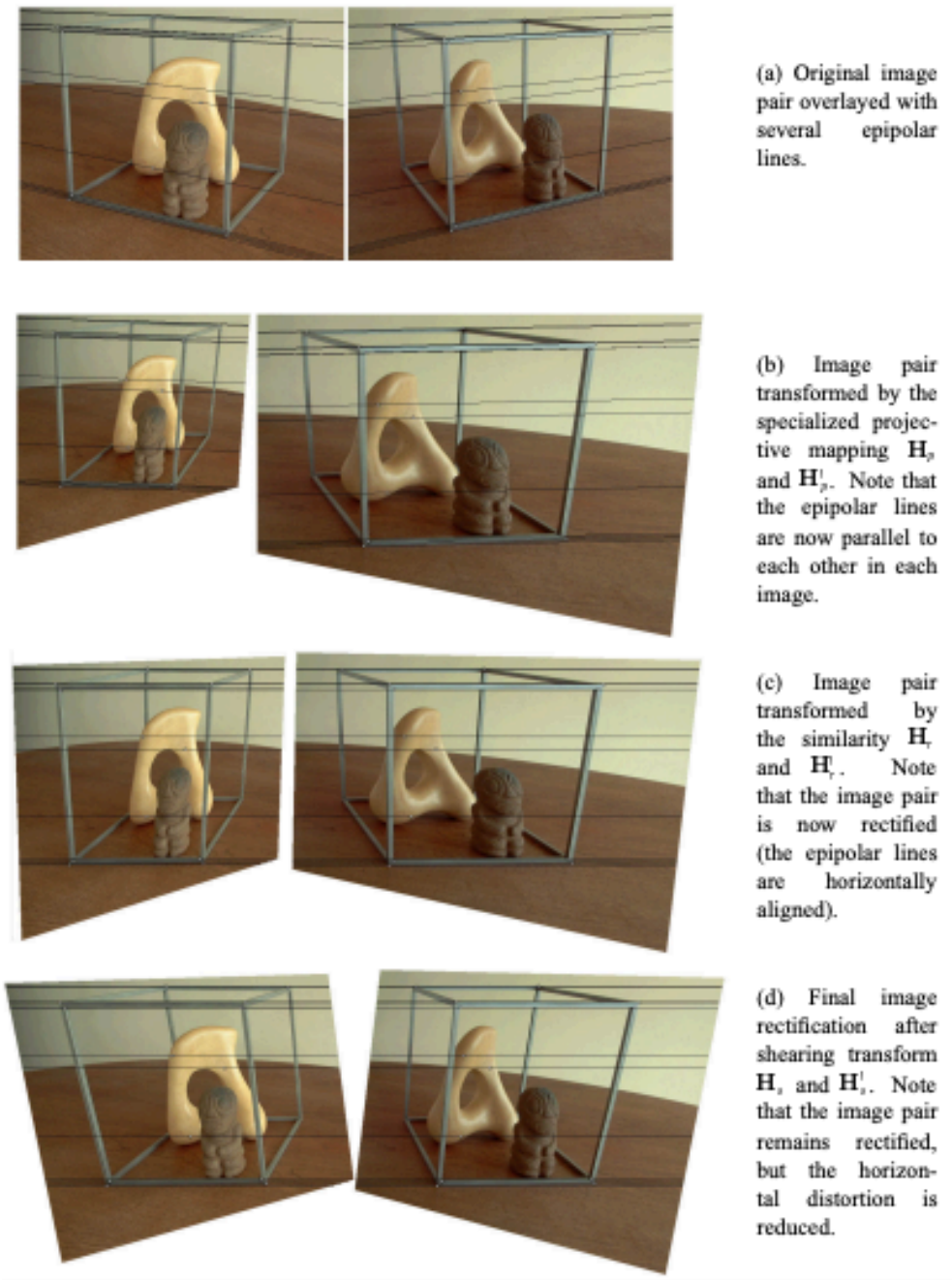


rectification and registration

from “Computing Rectifying Homographies for **Stereo Vision**”
by Loop & Zhang, 1999
https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr99-21.pdf

the transforms Hp and H0p were found that map the epipoles e and e0 to points at 1. In this section we define a pair of similarity transforms Hr and H0r that rotate these points at inf. into alignment with the direction $i = [1\ 0\ 0]^T$ as required for rectification. Additionally, a translation in the v-direction on one of the images is found to exactly align the scan-lines in both images.

… reduce the distortion with a shearing transform.



(a) Original image pair overlayed with several epipolar lines.

(b) Image pair transformed by the specialized projective mapping $\mathbf{H}_p$ and $\mathbf{H}'_p$. Note that the epipolar lines are now parallel to each other in each image.

(c) Image pair transformed by the similarity $\mathbf{H}_r$ and $\mathbf{H}'_r$. Note that the image pair is now rectified (the epipolar lines are horizontally aligned).

(d) Final image rectification after shearing transform $\mathbf{H}_s$ and $\mathbf{H}'_s$. Note that the image pair remains rectified, but the horizontal distortion is reduced.

Figure 3: An example showing various stages of the proposed rectification algorithm.

Projection, stereo and panoramic images

A review on bundle adjustment and 3D rectification and tracking in Section 1 and 2 of this paper:
(not implemented in this project)

from “Flight Dynamics-based Recovery of a UAV Trajectory using Ground Cameras”
by Rozantsev et al. 2017
http://openaccess.thecvf.com/content_cvpr_2017/papers/Rozantsev_Flight_Dynamics-Based_Recovery_CVPR_2017_paper.pdf

We propose a new method to estimate the 6-dof trajectory of a flying object such as a quadrotor UAV within a 3D airspace monitored using multiple fixed ground cameras…Our method requires neither perfect single-view tracking nor appearance matching across views. For robustness, we allow the tracker to generate multiple detections per frame in each video. The true detections and the data association across videos is estimated using robust multi-view triangulation and subsequently refined during our bundle adjustment procedure.

…

existing multi-camera tracking methods are designed to track people, vehicles to address indoor and outdoor surveillance tasks, where the targets are often on the ground. In contrast, small drones must be tracked within a 3D volume that is orders of magnitude larger

…

Most existing multi-camera systems also rely on accurate camera calibration that requires someone to collect calibration data by walking around in the scene.

…

3D from structured light

active sensing method called phase shifting method (PSM)
(not implemented in this project)

Projection, stereo and panoramic images

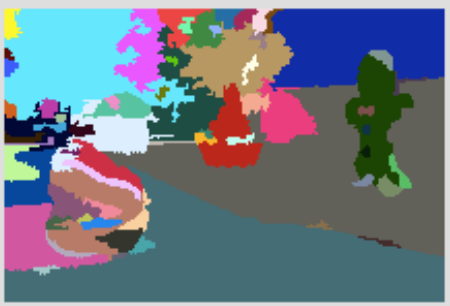
single point perspective and two and three point perspectives sometimes can have "foreshortened" object dimensions if the axes of the object are not parallel to the camera plane.

For example, when creating correspondence for the gingerbread man in the image below using partial shape matching and euclidean transformation for evaluation, half of the object is unmatched within a tolerance.

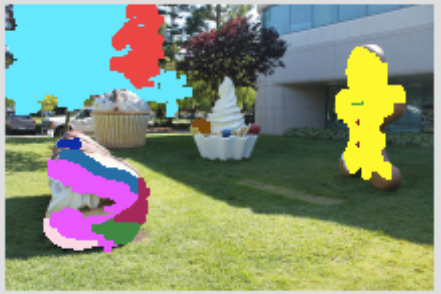
So, a rough calculation of the angle of inclination is needed to find the other matching points consistent with a single-point perspective projection.

The "foreshortening" along the x-axis is corrected using cosine of the angle, that is the true dimension is the measured divided by cosine of angle.
The foreshortening along the y axis can be approximated by the ratio of the far sides of two triangles, one embedded in the other.

image for the search for gingerbread man

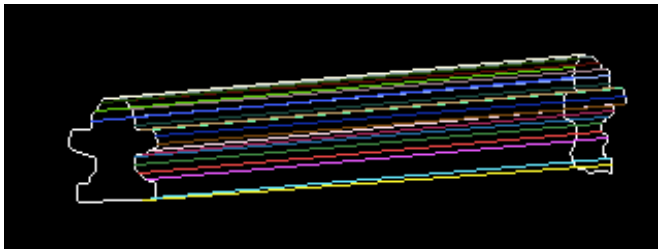


segmentation

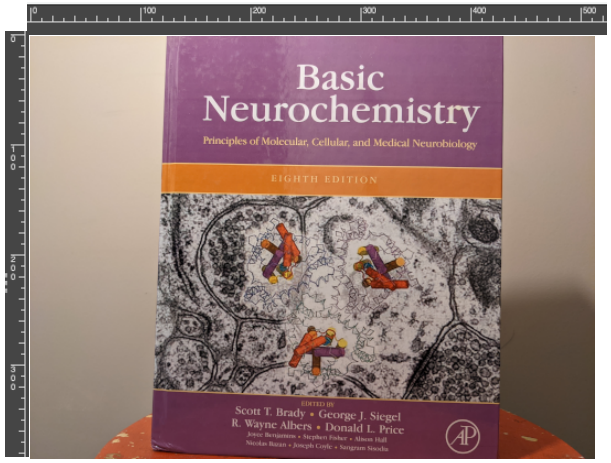


segmentation filtered to colors similar to the template object gingerbread man (not shown)

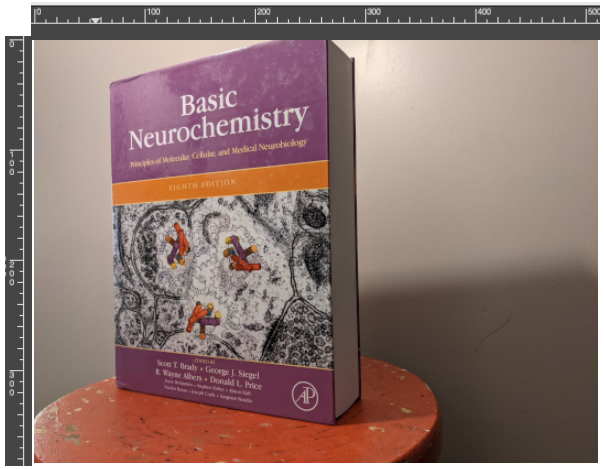
euclidean transformed matches …need a perspective projection search for the other half



SVD and epipolar details on test images

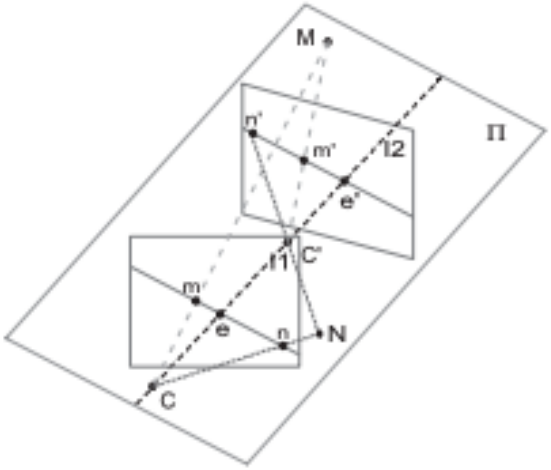


Coords:
262, 356
316, 342
260, 305
284, 279
234, 217
177, 76
216, 63
220, 158



Coords:
156, 308
191, 301
153, 270
167, 249
135, 202
97, 97
119, 83
125, 156

somewhat similar to Fig 2 from Oram "Rectification for Any Epipolar Geometry", but Fig 2 image 1 is image 2 here and image 2 here is further to right and further in z.




```
intrinsic camera matrix (for 512x384 image)
K = | 322     0  256 |
    |   0   322  192 |
    |   0     0    1 |
```

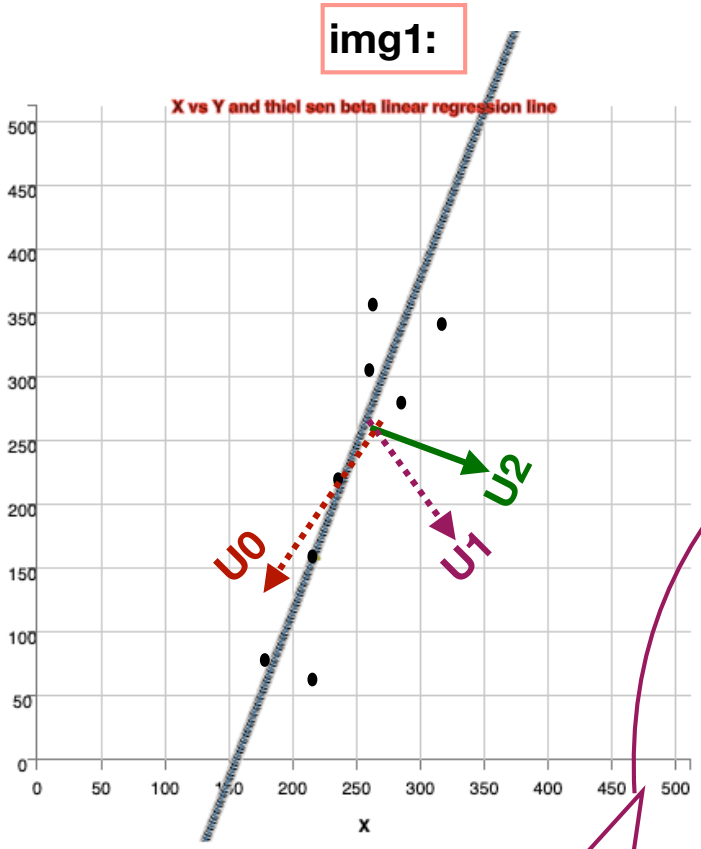
```
estimated rotation about y-axis is
about 35 degrees
```

```
book is 22 cm wide and 28 cm long.
using the pixel dimensions and book
size in pixels the scale from image
to world is roughly 0.22m/(310 pix
* 1.4e-6) ~ 507 in units of meters.
```

```
pixel width = 1.4e-3mm
image dimensions originally 4032x3024,
    then divided by 7.875 to 512x384, 300 dpi
FOV = 77 degrees = 1.344 radians

focal length_full = (4032/2.) / tan(1.344/2.)
             ~ 1604 pixels = 2.245 mm

focal length_512x384 ~ 322 pixels.
not yet including other scaling such as
magnification
```

SVD and epipolar details on test images

img1:



u0 = [-0.7077/0.0027, -0.7065/.0027, 1]
= [-262.1, -261.7, 1]
==> u0 theta = 45 degrees
u1 = [0.7065/0.0049, -0.7077/0.0049, 1]
= [144.2, -144.4, 1]
==> u1 theta = -45 degrees

Coords:

```
X =
262.000  , 316.000  , 260.000  284.000  234.000  177.000  216.000  220.000
356.000  , 342.000  , 305.000  279.000  217.000  76.000   63.000   158.000
1.000    , 1.000    , 1.000      1.000  1.000     1.000   1.000    1.000

mean=246.1250, 224.5000, 1.0000
stDev=43.5511, 115.3640, 0.0000
```

thiel sen y-Intercept, slope = (-407.64, 2.62)

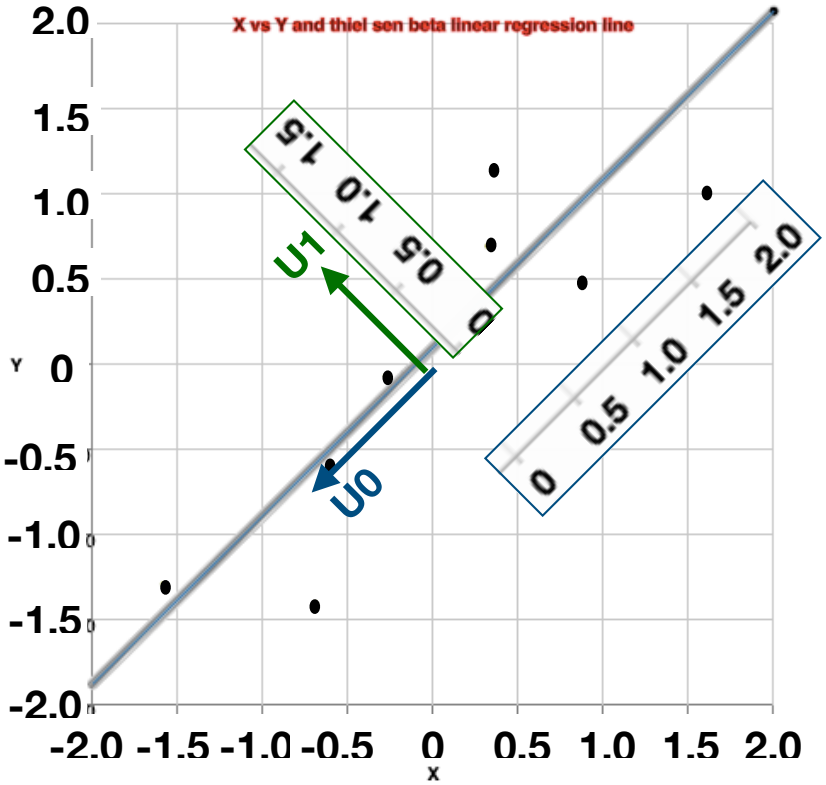
```
SVD(X):
U=
-0.7077, 0.7065, -0.0054
-0.7065, -0.7077, 0.0015
-0.0027, 0.0049, 1.0000

S=[984.6910338036112, 157.01103159470784, 0.3223419627696029]

V^T=
-0.4437, -0.4725, -0.4057, -0.4043, -0.3239, -0.1817, -0.2004, -0.2715
-0.4256, -0.1195, -0.2047, 0.0205, 0.0749, 0.4539, 0.6880, 0.2778
0.3909, -0.5772, 0.1854, -0.3374, 0.2078, 0.5001, -0.2126, 0.1655
-0.3909, -0.4000, 0.3803, 0.6804, -0.0498, 0.0988, -0.2504, -0.0684
-0.3261, 0.0470, -0.0843, -0.1033, 0.8999, -0.1365, -0.1738, -0.1229
-0.2312, 0.4917, -0.0691, -0.0872, -0.1348, 0.6723, -0.4115, -0.2302
-0.2634, 0.0864, 0.7687, -0.4571, -0.0758, -0.0770, 0.2452, -0.2271
-0.2955, 0.1337, 0.1083, -0.1792, -0.1008, -0.1568, -0.3448, 0.8351
```

u2 theta = atan(0.0015/-0.0054)
vector x and y lengths scaled by 100
= (100*cos(theta), 100*sin(theta))
= (96.35, -26.76)
drawn relative to (256, 256) in plot

NOTE: naively, one can see here and on next page that using geometric median in an ***alternative*** **normalization** would require adjustments in use of SVD analysis.

SVD and epipolar details on test images

img1 after standard normalization:



Normalized Coords:

```
A = X =
0.365  , 1.604  0.319  0.870  -0.278  -1.587  -0.692  -0.600
1.140  , 1.019  0.698  0.472  -0.065  -1.287  -1.400  -0.576
mean=0,0;  stDev=1,1
```

thiel sen y-Intercept, slope = (10.2, 0.99)

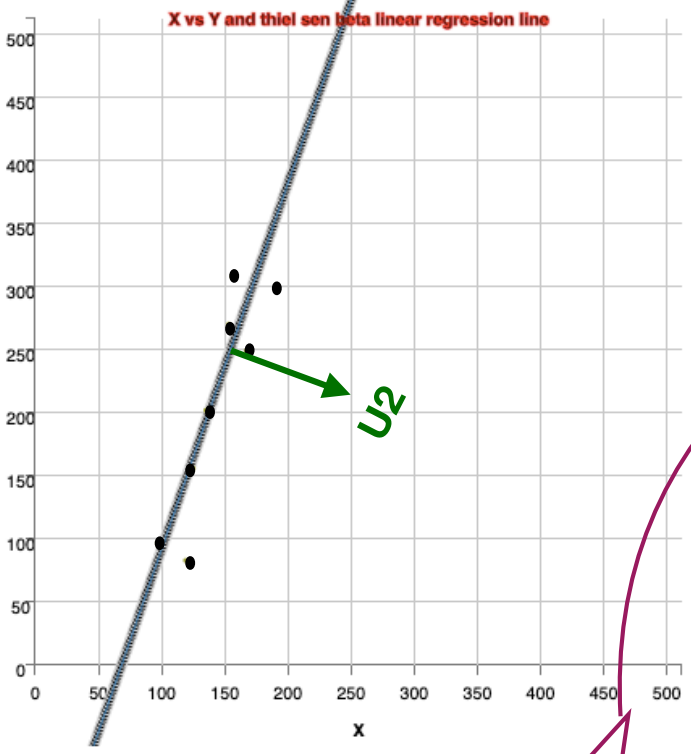
```
SVD(Normalized X w/ dimension reduction):
U=
-0.7071, -0.7071
-0.7071,  0.7071

S=[3.613605578424382, 0.9704920007811456]

V^T=
-0.2944, -0.5133, -0.1989, -0.2626, 0.0672, 0.5625, 0.4093, 0.2302
0.5649, -0.4269, 0.2763, -0.2894, 0.1555, 0.2186, -0.5160, 0.0171
-0.3131, 0.0857, 0.9273, -0.0291, -0.0009, 0.1034, 0.1443, 0.0497
-0.0304, -0.4073, 0.0581, 0.8962, 0.0490, 0.1115, -0.1051, 0.0257
-0.0399, 0.1690, -0.0341, 0.0347, 0.9812, -0.0254, 0.0637, -0.0016
0.3001, 0.5531, -0.0299, 0.1822, -0.0742, 0.7458, 0.0437, -0.0795
0.6135, -0.1346, 0.1373, 0.0652, -0.0022, -0.2108, 0.7268, -0.0991
0.1652, 0.1688, 0.0048, 0.0673, -0.0248, -0.1068, -0.0148, 0.9630
```

for i <= rank, $A * v_i = \sigma * u_i$
(for i > rank, A*v_i = 0 and A^T*u_i = 0)

```
U0 * S0 = [-2.555  , -2.555]
U1 * S1 = [-0.686  , 0.686]

A * V =
-2.555  , -0.686  -0.000  -0.000  -0.000  0.000  0.000  0.000
-2.555  , 0.686  0.000  -0.000  -0.000  0.000  0.000  0.000

A * v0 = [-2.555  , -2.555]
a^T * u0 = [-1.064, -1.855, -0.719, -0.949, 0.243, 2.033, 1.479, 0.832]
a^T * u1 = [0.548, -0.414, 0.268, -0.281, 0.151, 0.212, -0.501, 0.017]
```

SVD and epipolar details on test images

img2:



u0 = [-0.5408/-0.0036, -0.8412/-0.0036, 1]
= [150.2, 233.7, 1]
==> u0 theta = 57 degrees
u1 = [0.8411/0.0102, -0.5408/0.0102, 1]
= [82.5, -53, 1]
==> u1 theta = -32.7 degrees

Coords:

```
X =
156.000  , 191.000  , 153.000  167.000  135.000  97.000  119.000  125.000
308.000  , 301.000  , 270.000  249.000  202.000  97.000  83.000  156.000
1.000  , 1.000  , 1.000  1.000  1.000  1.000  1.000  1.000

mean=142.8750, 208.25, 1.0000
stDev=29.8302, 88.5272, 0.0000
```

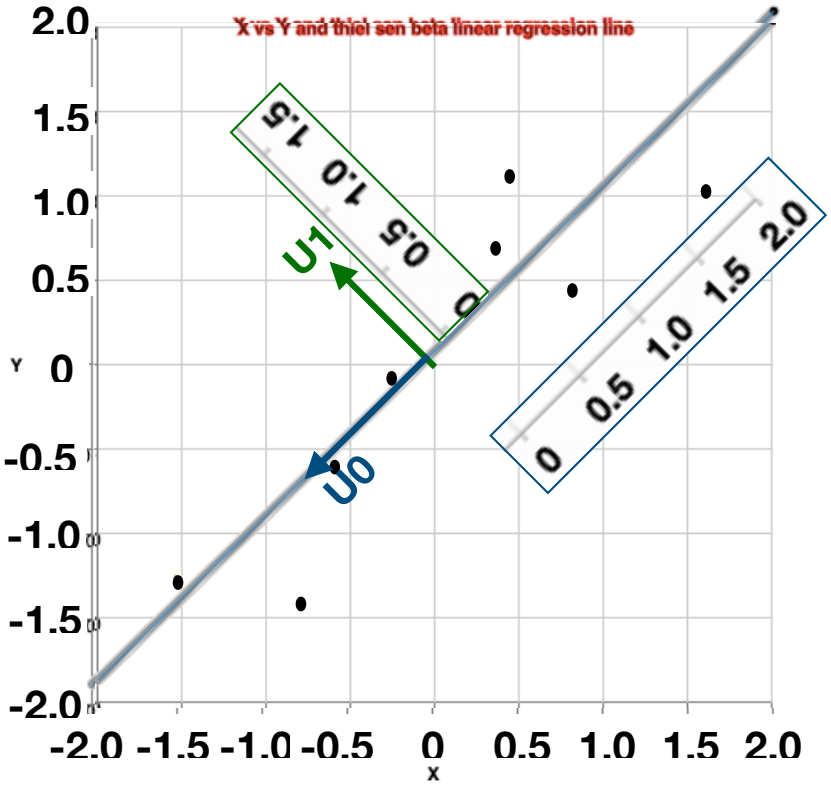
thiel sen y-Intercept, slope = (-197.4, 2.9)

```
SVD(X):
U=
-0.5408, 0.8411, -0.0105
-0.8412, -0.5408, 0.0025
-0.0036,  0.0102, 0.9999
S=[751.94, 77.04, 0.41]

V^T=
-0.4567, -0.4741, -0.4121, -0.3987, -0.3231, -0.1783, -0.1784, -0.2644
-0.4586, -0.0273, -0.2246, 0.0756, 0.0562, 0.3783, 0.7167, 0.2699
0.3257, -0.6127, 0.1700, -0.3167, 0.2146, 0.5447, -0.1038, 0.1891
-0.0066, -0.4255, -0.2453, 0.8435, 0.0139, 0.0590, -0.2061, -0.0329
-0.3923, -0.0698, 0.1100, -0.0394, 0.8647, -0.2144, -0.1157, -0.1431
-0.4748, 0.0728, 0.6221, 0.1278, -0.2574, 0.4116, -0.2046, -0.2973
0.1439, -0.4350, 0.4627, 0.0737, -0.1031, -0.4470, 0.5484, -0.2436
-0.2738, -0.1367, 0.2784, 0.0151, -0.1469, -0.3313, -0.2124, 0.8076
```

u2 theta = atan(0.0025/-0.0105)
vector x and y lengths scaled by 100
= (100*cos(theta), 100*sin(theta))
= (97.3, -23.2)
drawn relative to (256, 256) in plot

SVD and epipolar details on test images

img2 after standard normalization:



Normalized Coords:

```
A = X =
0.440  , 1.613  0.339  0.809  -0.264  -1.538  -0.800  -0.599
1.127  , 1.048  0.698  0.460  -0.071  -1.257  -1.415  -0.590
mean=0,0;  stDev=1,1
```

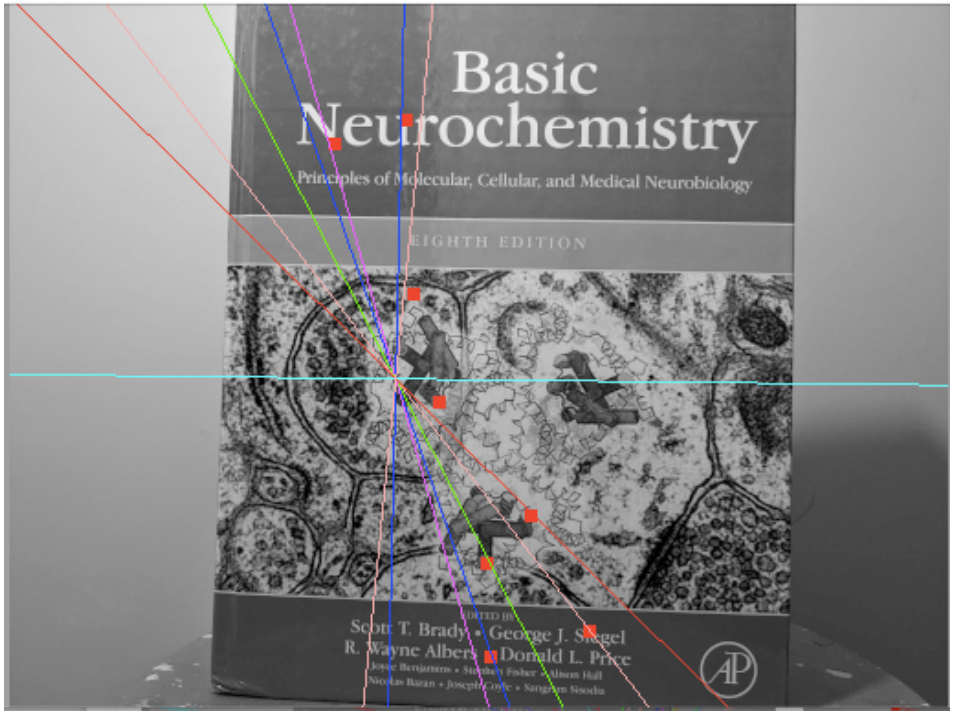
thiel sen y-Intercept, slope = (9.06, 0.98)

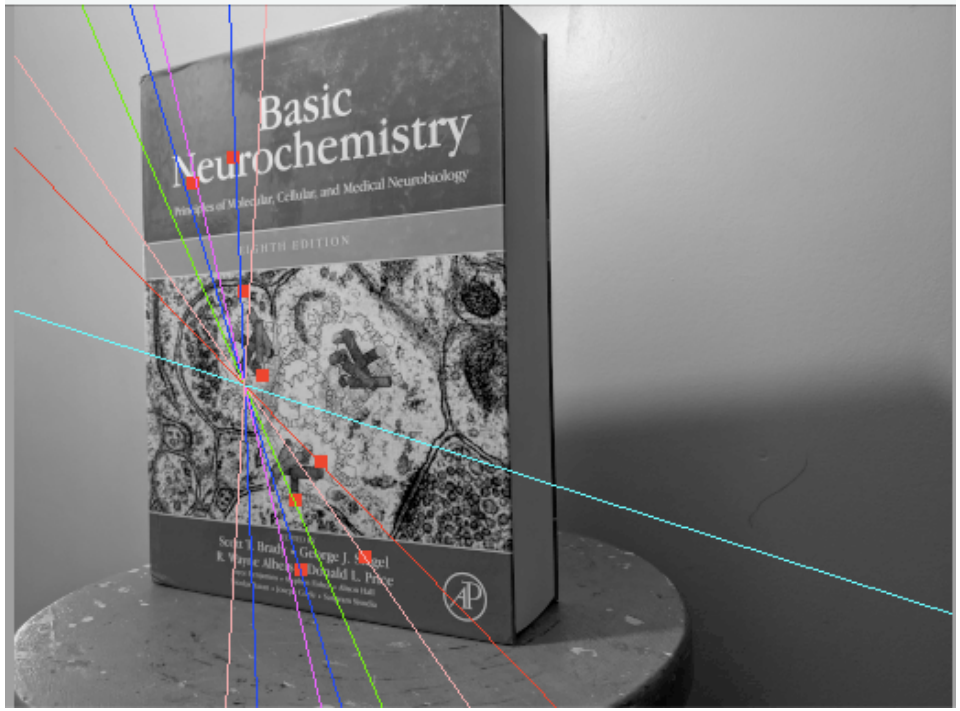
```
SVD(Normalized X w/ dimension reduction):
U=
-0.7071, -0.7071
-0.7071,  0.7071

S=[3.6376280065322755, 0.8761635042000003]

V^T=
-0.3046, -0.5173, -0.2016, -0.2467, 0.0650, 0.5432, 0.4306, 0.2312
0.5543, -0.4565, 0.2890, -0.2812, 0.1561, 0.2269, -0.4959, 0.0073
-0.3223, 0.0926, 0.9230, -0.0247, -0.0024, 0.0974, 0.1511, 0.0512
-0.0266, -0.3895, 0.0568, 0.9071, 0.0454, 0.1042, -0.0902, 0.0215
-0.0405, 0.1676, -0.0345, 0.0326, 0.9818, -0.0255, 0.0593, -0.0004
0.2859, 0.5470, -0.0318, 0.1659, -0.0702, 0.7613, 0.0287, -0.0728
0.6181, -0.1015, 0.1355, 0.0648, -0.0041, -0.2052, 0.7298, -0.1016
0.1734, 0.1621, 0.0077, 0.0618, -0.0230, -0.1037, -0.0265, 0.9632

U0 * S0 = [-2.572, -2.572]
U1 * S1 = [-0.620, 0.620]

A * V =
-2.572  , -0.620  0.000  -0.000  0.000  -0.000  -0.000  -0.000
-2.572  , 0.620  -0.000  -0.000  -0.000  0.000  -0.000  0.000

A * v0 = [-2.572, -2.572]
a^T * u0 = [-1.108, -1.882, -0.733, -0.897, 0.237, 1.976, 1.566, 0.841]
a^T * u1 = [0.486, -0.400, 0.253, -0.246, 0.137, 0.199, -0.434, 0.006]
```

SVD and epipolar details on test images

over-determined —> least squares; smallest eigenvector in SVD(A).V^T for orthogonal to matrix A reformatted to 2 dimensions.

using the 8 points, non-generally located, and the >= 8 point algorithm





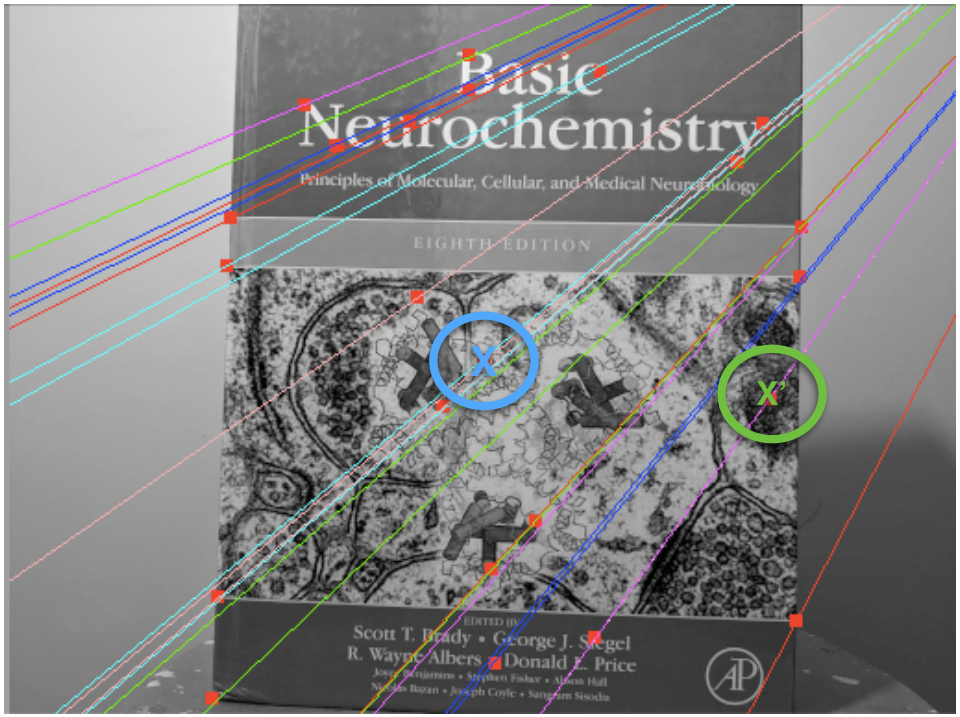

```
RANSAC fit=nMatchedPoints=8 nMaxMatchable=8
tolerance=0.20078232591749098
meanDistFromModel=6.824e-02 stDevFromMean=5.229e-02

de-normalized FM=
   2.839e-05, -8.525e-05, 1.144e-02
   5.055e-05, 5.464e-06, -1.174e-02
   -1.403e-02, 9.546e-03, 1.000e+00
```

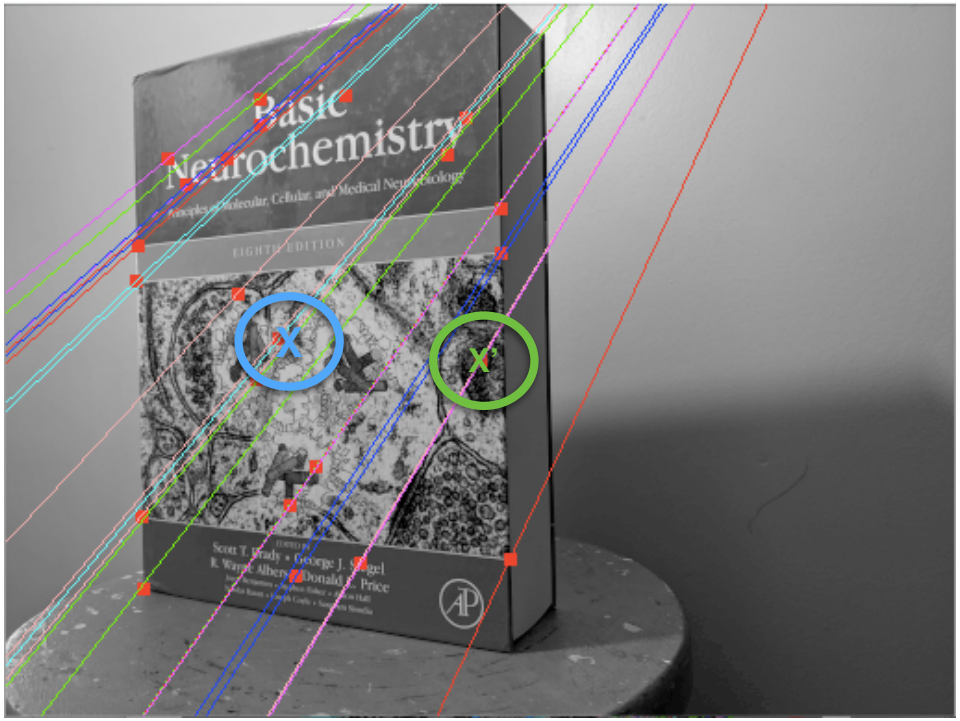
numbers not updated yet

SVD and epipolar details on test images

over-determined —> least squares; smallest eigenvector in SVD(A).V^T for orthogonal to matrix A reformatted to 2 dimensions.

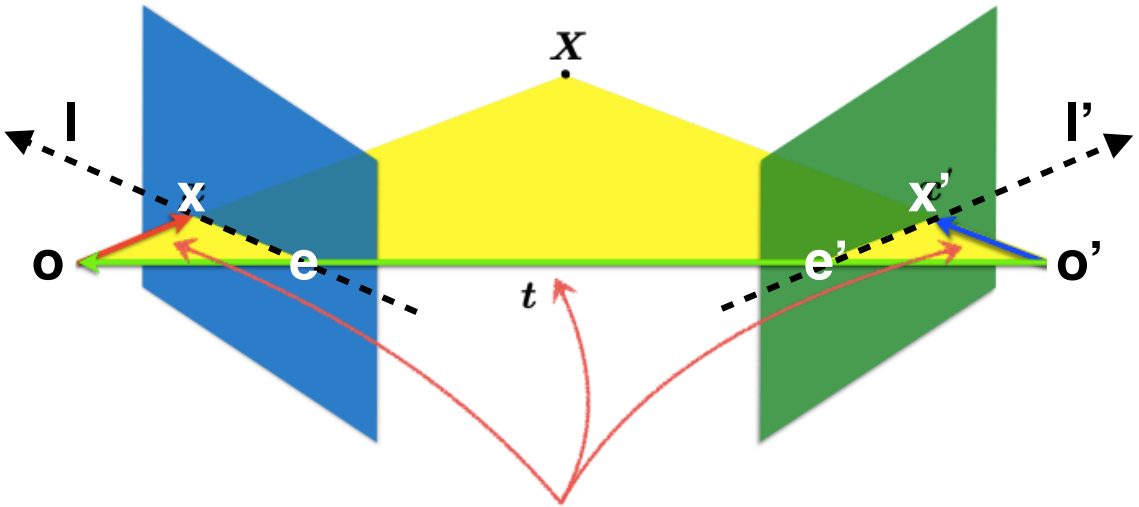
using 24 points more generally distributed and the >= 8 point algorithm





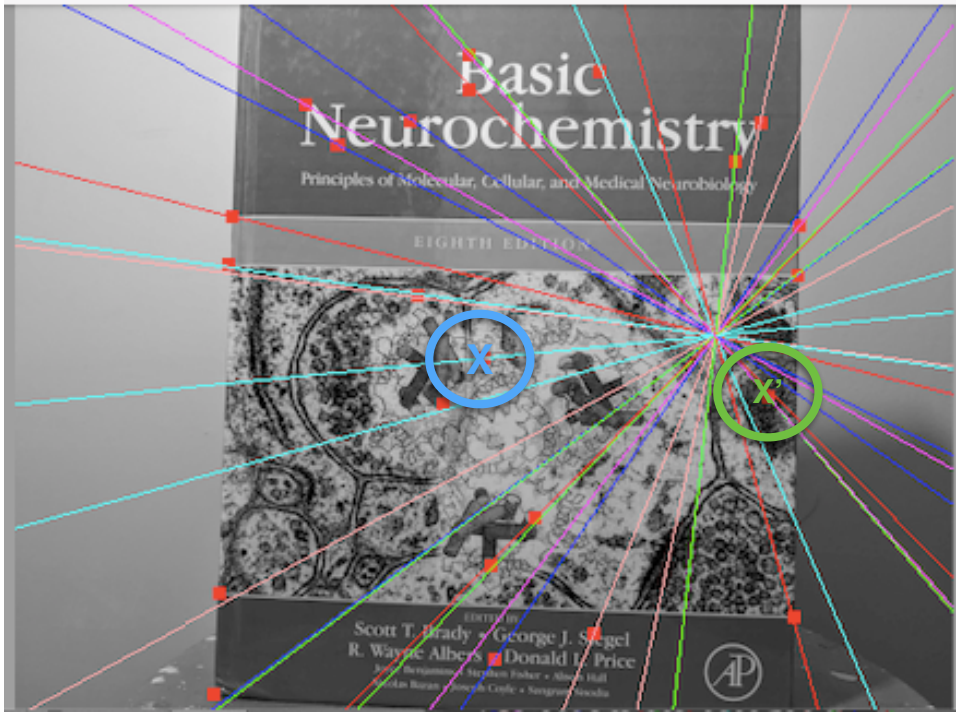

```
RANSAC fit=nMatchedPoints=24 nMaxMatchable=24
tolerance=1.1203557027113982
meanDistFromModel=3.817e-01 stDevFromMean=2.918e-01

de-normalized FM=
  9.677e-06, -1.799e-05, -9.523e-03
  2.067e-05, 7.137e-08, -1.410e-02
  8.891e-04, 9.916e-03, 1.000e+00

P2=
  4.263e-04, 4.754e-03, 4.794e-01, -8.776e-01
  7.802e-04, 8.702e-03, 8.776e-01, 4.795e-01
 -2.278e-05, 8.563e-06, 1.694e-02, -1.596e-03
```

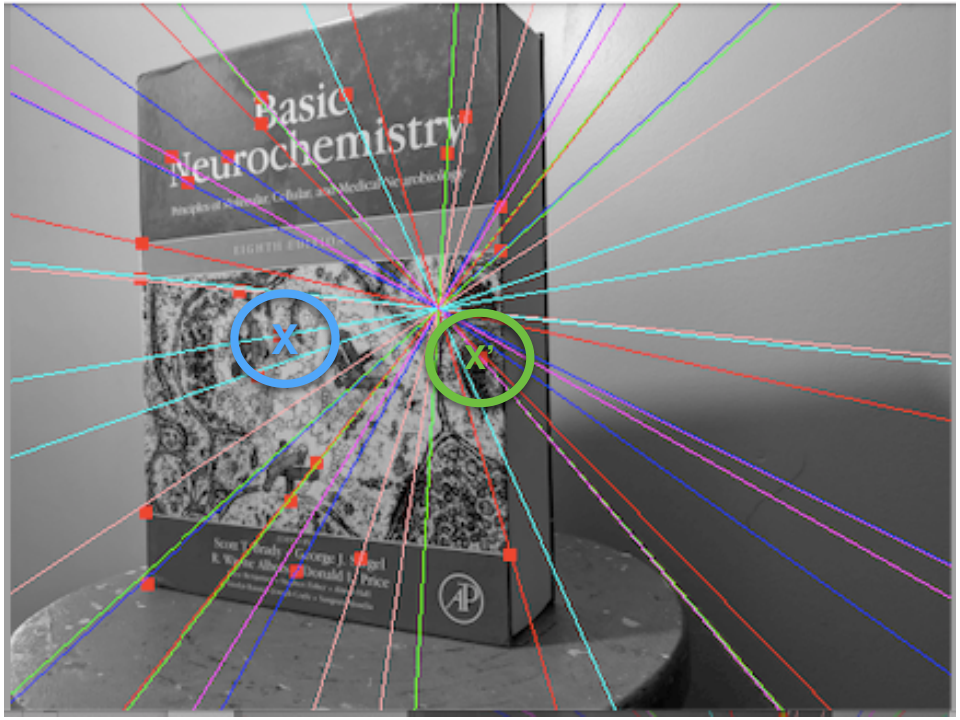
numbers not updated yet



SVD and epipolar details on test images

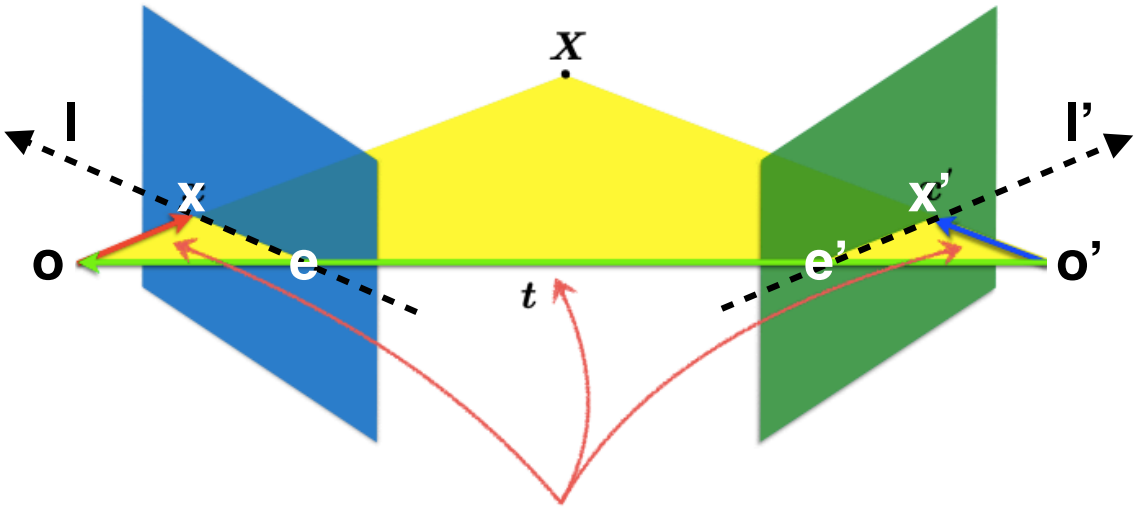
using 24 points more generally distributed & RANSAC w/ 7-pt algorithm and nIterations for 95% probability of good random subsample of size 7 (nIter dynamically decr by re-estimating w/ smaller outlier %).






```
RANSAC fit=nMatchedPoints=24 nMaxMatchable=24
tolerance=0.8033804327526433
meanDistFromModel=2.210e-01 stDevFromMean=2.092e-01

de-normalized FM=
  2.573e-06, -5.055e-05, 8.106e-03
  4.471e-05, 2.159e-06, -1.748e-02
 -7.995e-03, 1.143e-02, 1.000e+00

P2=
 -4.626e-03, 6.614e-03, 5.786e-01, 8.157e-01
  6.522e-03, -9.325e-03, -8.156e-01, 5.785e-01
  3.498e-05, 3.100e-05, -1.894e-02, 3.498e-03
```

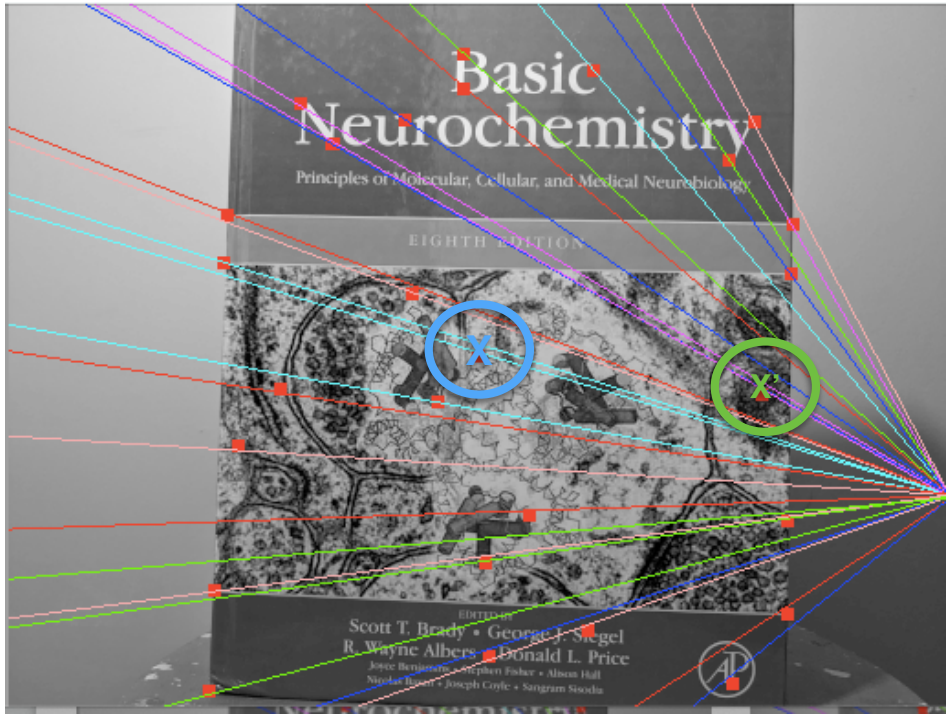
numbers not updated yet

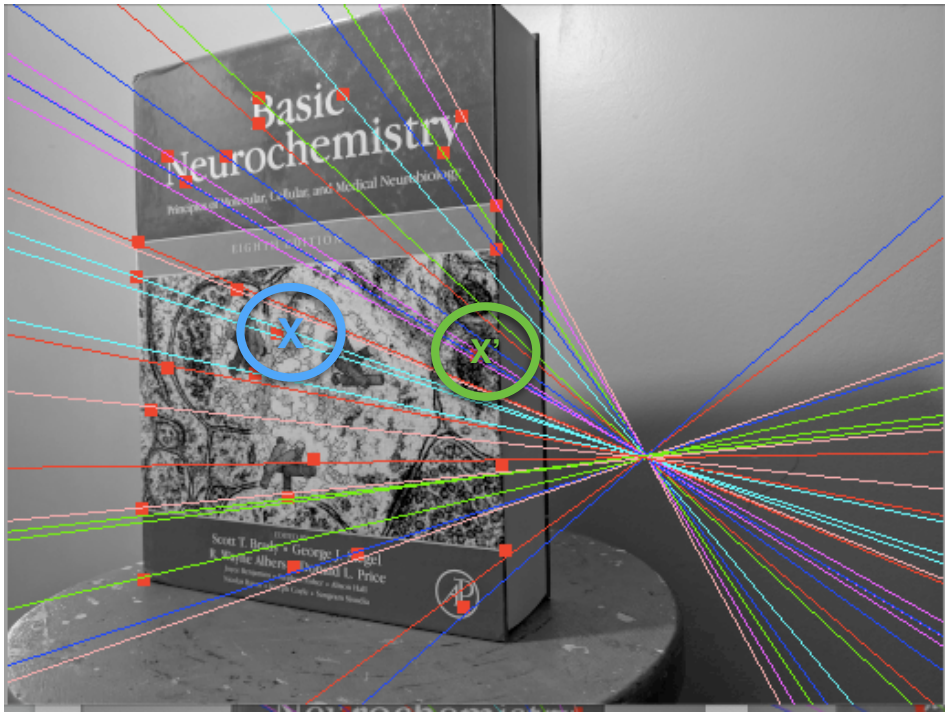


SVD and epipolar details on test images

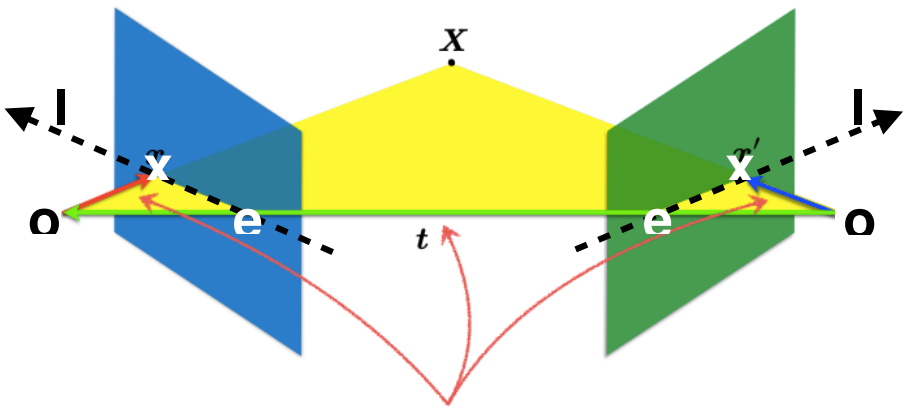
7-pt alg: rank 7 < number of A columns, 9, —> solving for matrix right null space of A orthogonal to the row space of A for 2 independent vectors orthogonal to A —> det(2 linear convex combinations)=0 —> 3rd order characteristic root finding.

using 28 points more generally distributed & RANSAC w/ 7-pt algorithm and nIterations for 95% probability of good random subsample of size 7 (nIter fixed at 382).

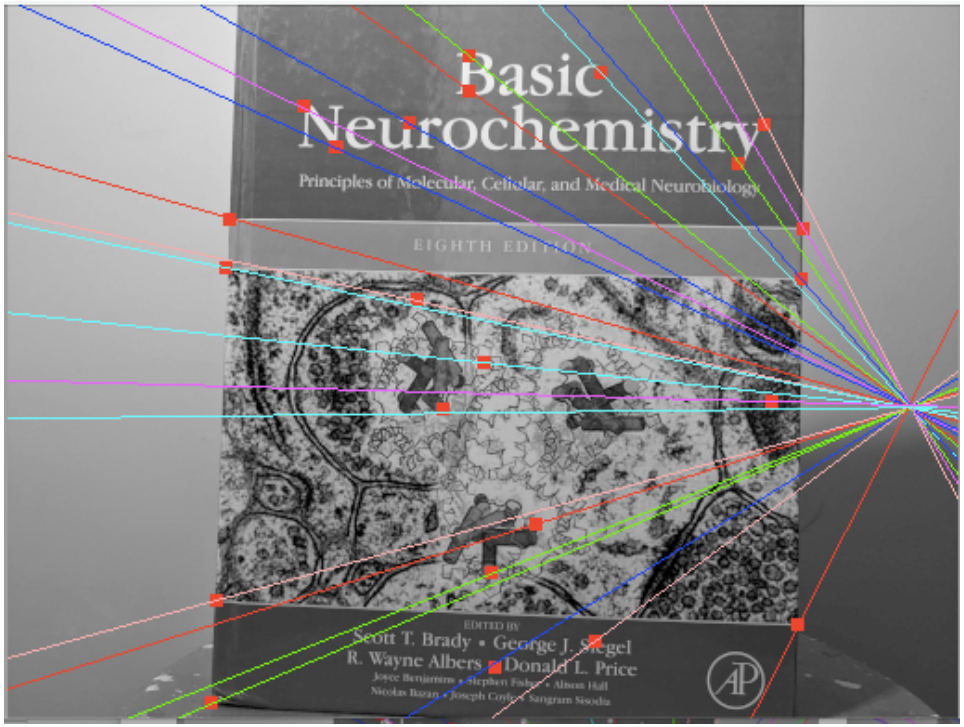


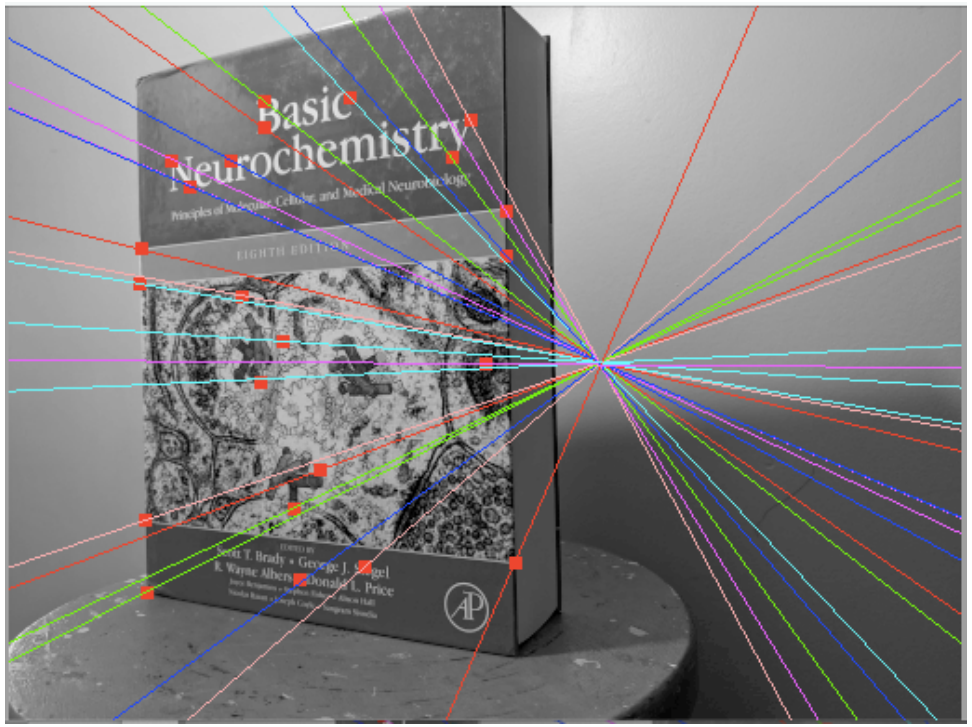



```
RANSAC fit=nMatchedPoints=24 nMaxMatchable=24
tolerance=0.276846736782158 meanDistFromMode
l=5.875e-02 stDevFromMean=7.210e-02

fm=
 -2.980e-02, -8.519e-01, 6.611e-01
  8.368e-01, 2.665e-02, -1.845e+00
-6.622e-01, 1.929e+00, 7.745e-02

de-normalized FM=
    -1.2479e-06, -3.5676e-05, 1.0175e-02
     3.5041e-05, 1.1161e-06, -1.8315e-02
    -8.2864e-03, 1.2175e-02, 1.0000e+00
```



SVD and epipolar details on test images





The fundamental matrix terms solved for by the orthogonal component of best fit to A:

```
            A[i][0] = x1 * x2; <- FM[0][0]
            A[i][1] = x1 * y2; <- FM[1][0]
            A[i][2] = x1;      <- FM[2][0]
            A[i][3] = y1 * x2; <- FM[0][1]
            A[i][4] = y1 * y2; <- FM[1][1]
            A[i][5] = y1;      <- FM[2][1]
            A[i][6] = x2;      <- FM[0][2]
            A[i][7] = y2;      <- FM[1][2]
            A[i][8] = 1;
```

where FM is orthogonal to the linear fit to the column vectors of A.

solution to fundamental matrix using normalized coords:

```
    -2.980e-02, -8.519e-01   6.611e-01
     8.368e-01, 2.665e-02,  -1.845e+00
    -6.622e-01, 1.929e+00,   7.745e-02
```

using the normalization scale and offset to transform the x1, y1, x2, y2 back to image pixel coordinates:

de-normalized fundamental matrix:

```
  -1.2479e-06, -3.5676e-05, 1.0175e-02
  3.5041e-05, 1.1161e-06, -1.8315e-02
  -8.2864e-03, 1.2175e-02, 1.0000e+00
```

epipoles are derived from the de-normalized fundamental matrix as orthogonal vectors to the linear fit to the column vectors in the de-normalized fundamental matrix. (note, the column vector contents are seen in the panel to left of this)

SVD and epipolar details on test images

http://www.cs.cmu.edu/~16385/s17/Slides/12.5_Reconstruction.pdf

Decomposing **F** into **R** and **T**

If we have calibrated cameras we have $\mathbf{K}$ and $\mathbf{K}'$

Essential matrix: $\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K}$

SVD: $\mathbf{E} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ Let $\mathbf{W} - \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

We get FOUR solutions:

$$\mathbf{E} = [\mathbf{R}|\mathbf{T}]$$

$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^{\top}$ $\mathbf{R}_2 = \mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top}$ $\mathbf{T}_1 = U_3$ $\mathbf{T}_2 = -U_3$

two possible rotations — two possible translations

We get FOUR solutions:

$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^{\top}$ $\mathbf{T}_1 = U_3$

$\mathbf{R}_1 = \mathbf{U}\mathbf{W}\mathbf{V}^{\top}$ $\mathbf{T}_2 = -U_3$

$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top}$ $\mathbf{T}_2 = -U_3$

$\mathbf{R}_2 = \mathbf{U}\mathbf{W}^{\top}\mathbf{V}^{\top}$ $\mathbf{T}_1 = U_3$

Which one do we choose?

Compute determinant of R, valid solution must be equal to 1
(note: det(R) = -1 means rotation and reflection)

Compute 3D point using triangulation, valid solution has positive Z value
(Note: negative Z means point is behind the camera)

```
    [junit] E=
    [junit] 1.005e-01, -1.307e+00, -9.243e-01
    [junit] 1.265e+00, 1.612e-01, 1.344e+00
    [junit] 1.134e+00, -1.252e+00, 1.139e-01
    [junit]
    [junit] R1=
    [junit] -2.303e-01, 5.509e-01, -8.022e-01
    [junit] 6.603e-01, -5.170e-01, -5.447e-01
    [junit] -7.148e-01, -6.551e-01, -2.447e-01
    [junit]
    [junit] R2=
    [junit] 9.974e-01, 7.176e-02, 5.316e-03
    [junit] -7.125e-02, 9.952e-01, -6.728e-02
    [junit] -1.012e-02, 6.673e-02, 9.977e-01
    [junit]
    [junit] t1=
    [junit] -6.346e-01, -4.874e-01, 5.997e-01
    [junit] t2=
    [junit] 6.346e-01, 4.874e-01, -5.997e-01
    [junit] det(R1)=1.000e+00
    [junit] det(R2)=1.000e+00
```

estimated rotation about y axis from P2 = −21 degrees

```
    [junit] P2=
    [junit] -1.311e+00, 5.133e-01, -8.613e-01, -6.346e-01
    [junit] 7.801e-01, -1.578e+00, -4.821e-01, -4.874e-01
    [junit] -7.538e-01, -7.391e-01, -1.303e+00, 5.997e-01
    [junit]
    [junit] SVD(p2).U==
    [junit]  -7.721e-01,  3.202e-02,  -6.346e-01
    [junit]  4.321e-01,  7.588e-01,  -4.874e-01
    [junit]  -4.660e-01,  6.505e-01,  5.997e-01

[junit] choosing solution: R2, T1
    [junit] e1=
    [junit] -0.963, -0.781, 1.000
    [junit] e2=
    [junit] -1.058, -0.813, 1.000
    [junit] e1_normalized=
    [junit] -0.963, -0.781, 1.000
    [junit] e2_normalized=
    [junit] -1.058, -0.813, 1.000
    [junit] rRect=
    [junit] -0.369, -0.283, 0.348
    [junit] 0.609, -0.793, 0.000
    [junit] 0.276, 0.212, 0.465
    [junit]
    [junit] rRect*e1=
    [junit] 0.968, 0.000, -0.000
```

SVD and epipolar details on test images

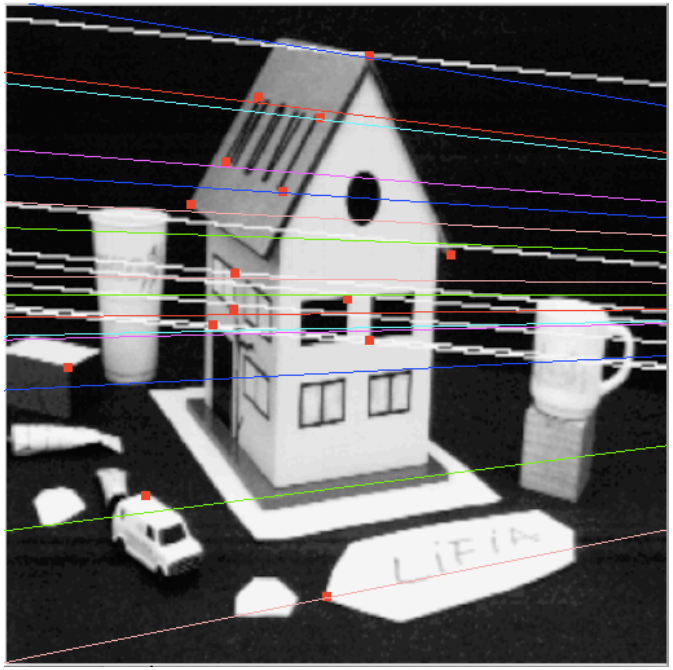
In Defense of the Eight-Point Algorithm Richard I. Hartley

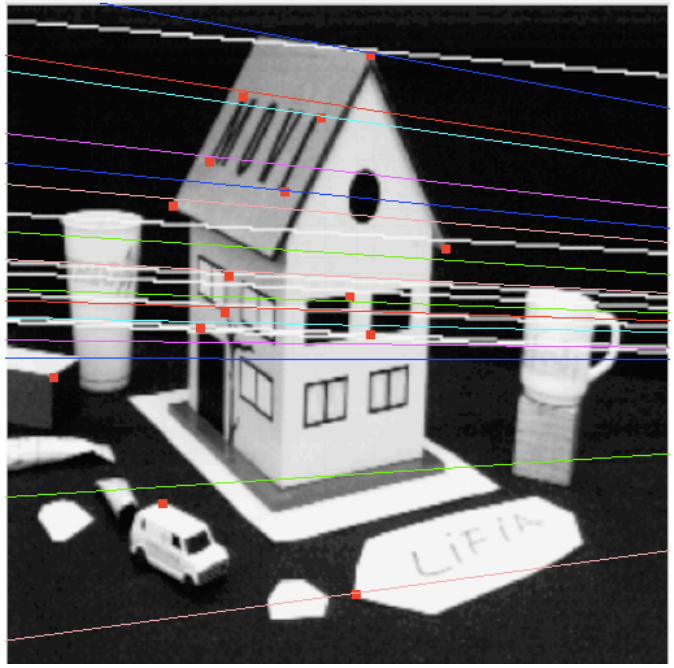
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 6, JUNE 1997

To verify this conclusion, below is the fundamental matrix for the pair of house images in Fig 1.[3]

$$F = \begin{pmatrix} -9.796e-08 & 1.473e-06 & -6.660e-04 \\ -6.346e-07 & 1.049e-08 & 7.536e-03 \\ 9.107e-04 & -7.739e-03 & -2.364e-02 \end{pmatrix} \quad (7)$$

```
normalized by F[2][2]:
4.144e-06, -6.231e-05, 2.817e-02
 2.684e-05, -4.437e-07, -3.188e-01
-3.852e-02, 3.274e-01, 1.000e+00
```





From this project's code: 16 points and their colored epipolar lines give similar sol'n to Hartley '97.

```
de-normalized FM=
3.455e-06, 7.863e-05, -2.257e-02
-8.895e-05, 6.490e-06, 1.246e-01
1.920e-02, -1.255e-01, 1.000e+00
```

Section 6 in Hartley 1997 on what to roughly expect in the FM by a typical coordinate magnitude (image size) for many cameras:
For a typical coordinate will be of the order of (100, 100, 1), the corresponding fundamental matrix FM will be obtained from the original one (see paper) by multiplying the first two rows, and the first two columns by 10^-2. Entries in the the top left 2 ¥ 2 block will be multiplied by 10^-4.

FM = | 1e-4	1e-4	1e-2 |
| 1e-4	1e-4	1e-2 |
| 1e-2	1e-2	1 |

.one can see that in the terms that are solved for as orthogonal to the assignments in the matrix A.
A[i][0], A[i][1] , A[i][3], A[i][4] are terms x1*x2, x1*y2, y1*x2, y1*y2, respectively

SVD and epipolar details on test images

In Defense of the Eight-Point Algorithm Richard I. Hartley
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 6, JUNE 1997

The normalized stats for the16 correspondences:

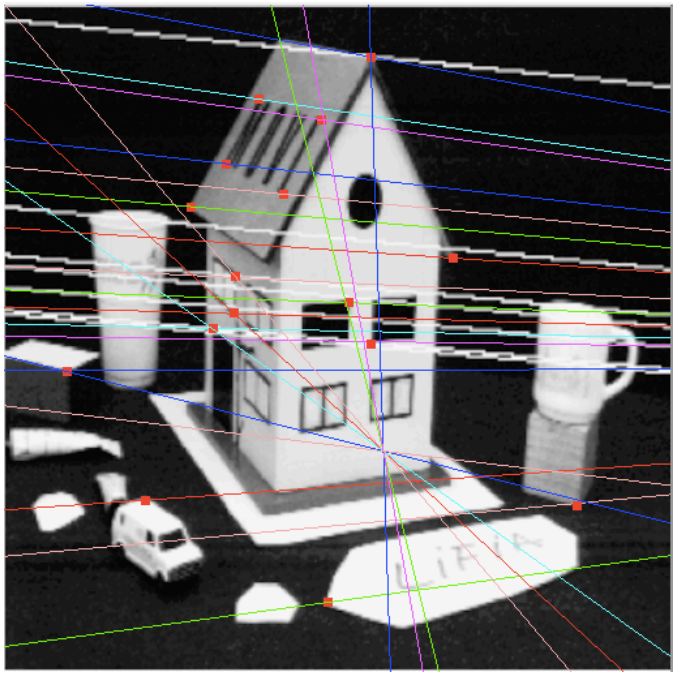
```
fm before de-normalization FM=
-2.430e-03, -5.531e-02, 3.050e-02
6.257e-02, -4.565e-03, -7.036e-01
-3.560e-03, 7.049e-01, -2.492e-04

dist sampson^2=
   1.667e-04,  7.893e-04,  5.188e-04,  7.936e-04,  1.074e-03,  2.336e-03,
   9.733e-03,  2.382e-02,  5.401e-05,  7.407e-03,  9.218e-04,  8.365e-05,
   7.631e-04,  5.049e-05,  1.338e-03,  3.144e-03
dist perp to epipolar^2=
   6.674e-04, 3.157e-03, 2.075e-03, 3.174e-03, 4.298e-03, 9.342e-03,
   3.894e-02, 9.530e-02, 2.163e-04, 2.966e-02, 3.689e-03, 3.347e-04,
   3.053e-03, 2.019e-04, 5.353e-03, 1.258e-02
Total dist_sampson = 2.302e-01
Total dist_perp = 4.605e-01
```

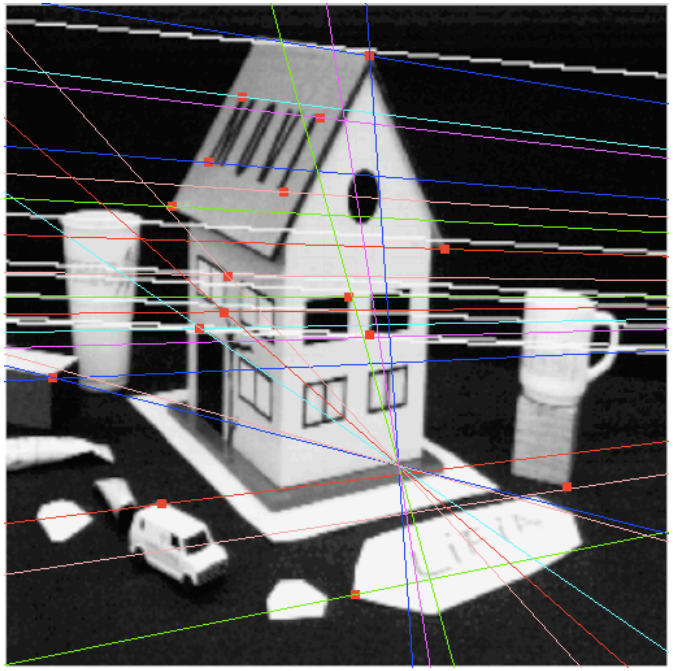
The de-normalized stats for the16 correspondences:

```
de-normalized FM=
4.135e-06, -1.065e-04, 2.302e-02
9.411e-05, 7.768e-06, -1.500e-01
-2.698e-02, 1.493e-01, 1.000e+00

samspon errors=nMatchedPoints=16 nMaxMatchable=16 tolerance=2.0
  meanDistFromModel = 3.848e-01 stDevFromMean=4.398e-01
  tol=2.0

epipolar fit=nMatchedPoints=14 nMaxMatchable=16 tolerance=2.0
  meanDistFromModel = 4.824e-01 stDevFromMean=4.257e-01
  tol=2.0
```

SVD and epipolar details on test images

In Defense of the Eight-Point Algorithm Richard I. Hartley
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 6, JUNE 1997





From this project's code: 8 out of the 16 points and their colored epipolar lines give similar sol'n to Hartley '97.

```
de-normalized FM=
-1.867e-06, -1.522e-04, 5.262e-02
1.570e-04, 5.769e-06, -4.772e-02
-5.523e-02, 4.414e-02, 1.000e+00
```

SVD and epipolar details on test images

In Defense of the Eight-Point Algorithm Richard I. Hartley
IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 6, JUNE 1997

The normalized stats for the 8 correspondences:

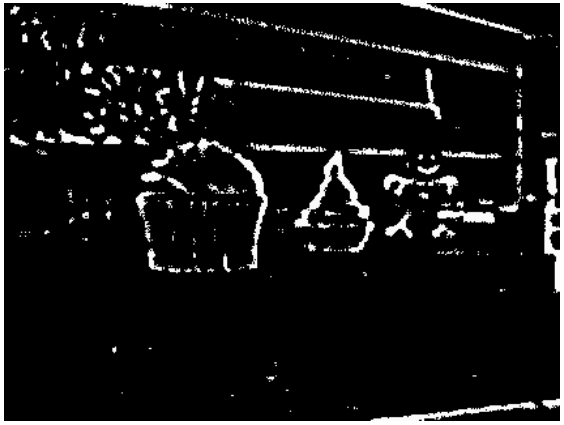
```
fm before de-normalization FM=
5.306e-03, -4.461e-01, 5.010e-01
4.327e-01, -1.640e-02, -3.059e-01
-4.370e-01, 2.779e-01, 1.306e-02

dist sampson^2=
   3.691e-03,  5.770e-04,  8.350e-03,  1.686e-03,  3.993e-03,  1.151e-02,
   1.313e-02,  2.375e-05
dist perp to epipolar^2=
   1.485e-02, 2.325e-03, 3.411e-02, 6.791e-03, 1.602e-02, 4.712e-02,
   5.255e-02, 9.731e-05
Total dist_sampson = 2.073e-01
Total dist_perp = 4.170e-01
```

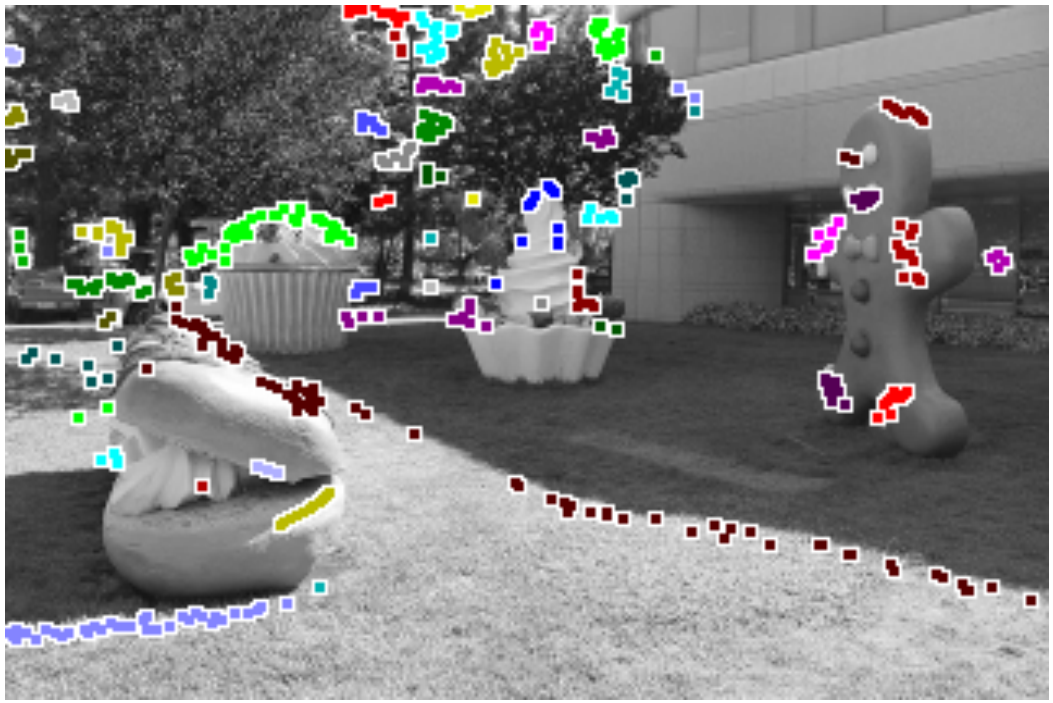
The de-normalized stats for the 8 correspondences:

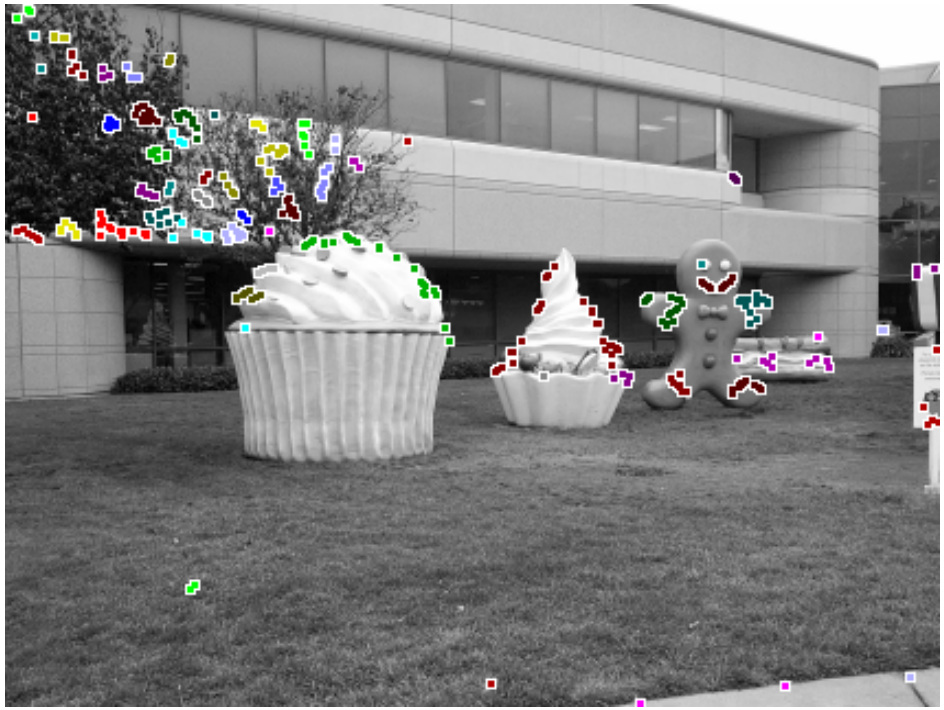
```
de-normalized FM=
4.135e-06, -1.065e-04, 2.302e-02
9.411e-05, 7.768e-06, -1.500e-01
-2.698e-02, 1.493e-01, 1.000e+00

samspon errors=nMatchedPoints=6 nMaxMatchable=8 tolerance=2.0
  meanDistFromModel = 4.841e-01 stDevFromMean=5.931e-01
  tol=2.0
epipolar fit=nMatchedPoints=5 nMaxMatchable=8 tolerance=2.0
  meanDistFromModel = 5.015e-01 stDevFromMean=3.390e-01
  tol=2.0
```

The remaining notes and snapshots on corners, stereo matching and matching under conditions of different lighting, pose and location are in file **doc/colorSegmentation3.pdf**

Note: tried the use of a segmentation filter based upon the atrous wavelet to limit the points to be matched. (later methods of MSER + HOGs proved more successful).







Some Notation

http://16720.courses.cs.cmu.edu/lec/image_formation_lec9.pdf

[Using Matlab's rows x columns]

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= K_{3\times3} \begin{bmatrix} R_{3\times3} & T_{3\times1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= M_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Claims (without proof):

1. A 3x4 matrix 'M' can be a camera matrix iff det(M) is not zero
2. M is determined only up to a scale factor

$$M_{3\times4} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} A_{3\times3} & b_{3\times1} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= A_{3\times3} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + b_{3\times1}$$

$$M = \begin{bmatrix} m_1^T \\ m_2^T \\ m_3^T \end{bmatrix}, \quad A = \begin{bmatrix} a_1^T \\ a_2^T \\ a_3^T \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Some Notation

http://16720.courses.cs.cmu.edu/lec/image_formation_lec9.pdf

Applying the projection matrix

$$x = \frac{1}{\lambda}([X \quad Y \quad Z]\, a_1 + b_1)$$

$$y = \frac{1}{\lambda}([X \quad Y \quad Z]\, a_2 + b_2)$$

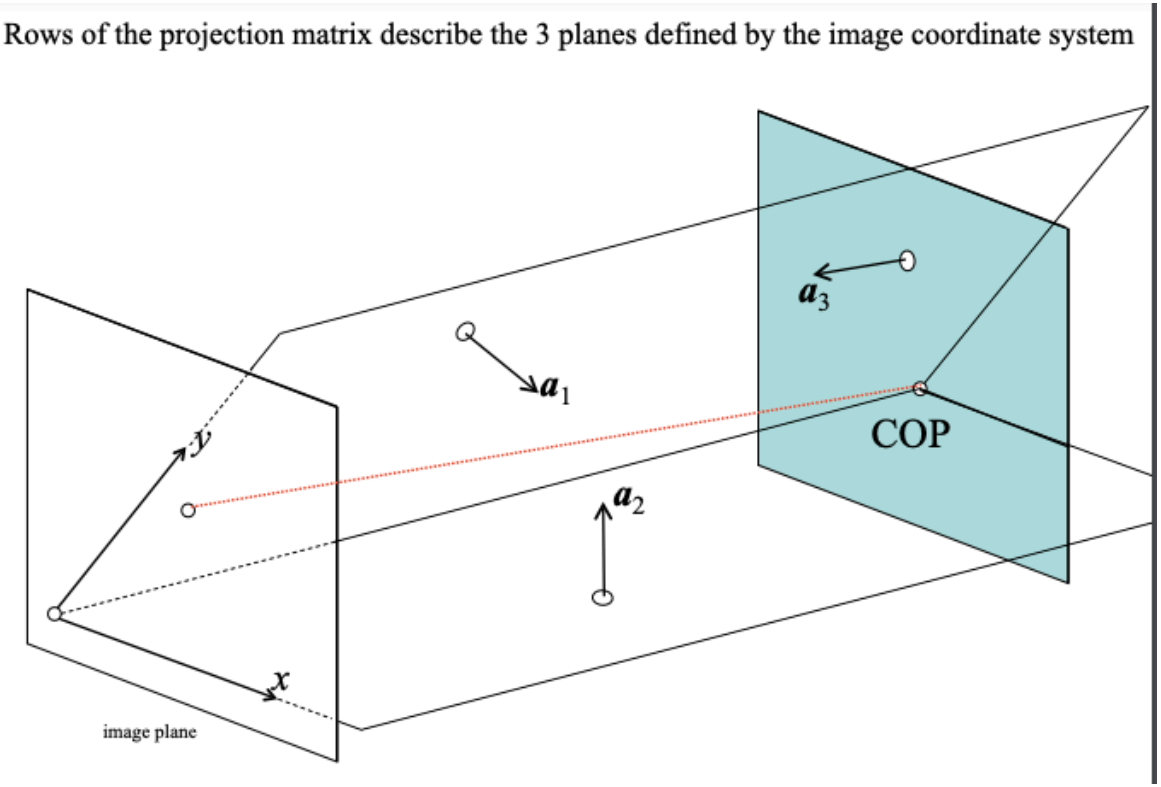
$$\lambda = [X \quad Y \quad Z]\, a_3 + b3$$

Set of 3D points that project to x = 0: $[X \quad Y \quad Z]\, a_1 + b_1 = 0$

Set of 3D points that project to y = 0: $[X \quad Y \quad Z]\, a_2 + b_2 = 0$

Set of 3D points that project to x = inf or y = inf: $[X \quad Y \quad Z]\, a_3 + b_3 = 0$

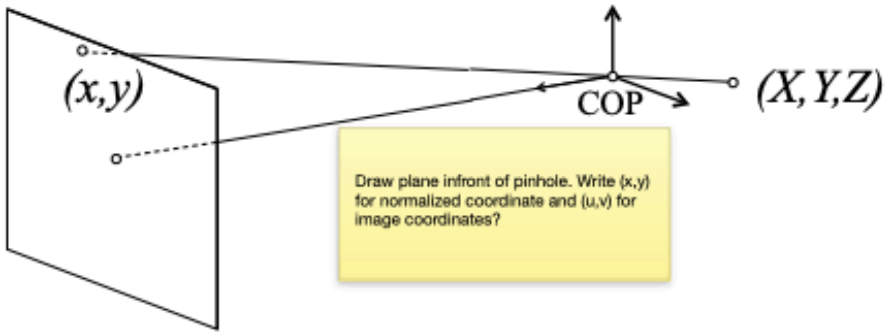
Rows of the projection matrix describe the 3 planes defined by the image coordinate system



Some Notation

http://16720.courses.cs.cmu.edu/lec/image_formation_lec9.pdf

Other geometric properties



What's set of (X,Y,Z) points that project to same (x,y)?

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \lambda w + b \quad \text{where} \quad w = A^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, b = -A^{-1}b$$

What's the position of COP / pinhole?

$$A \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + b = 0 \quad \Rightarrow \quad \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = -A^{-1}b$$

Some Notation

http://16720.courses.cs.cmu.edu/lec/image_formation_lec9.pdf

Affine cameras

Captures 3D affine transformation + orthographic projection + 2D affine transformation

perspective

weak perspective

$$m_3^T = [0 \quad 0 \quad 0 \quad 1]$$

$$m_3^T = [0 \quad 0 \quad 0 \quad 1] \Rightarrow \begin{aligned} x &= [X \quad Y \quad Z]\, a_1 + b_1 \\ y &= [X \quad Y \quad Z]\, a_2 + b_1 \end{aligned}$$

Image coordinates (x,y) are an *affine* function of world coordinates (X,Y,Z)

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

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ & & & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{A}\mathbf{X} + \mathbf{b}$$

- Projection defined by 8 parameters
- Parallel lines project to parallel lines
- 2D points = linear projection of 3D points (+ 2D translation)

- Example: Weak-perspective projection model
- Projection defined by 8 parameters
- Parallel lines project to parallel lines
- The transformation can be written as a direct linear transformation plus an offset