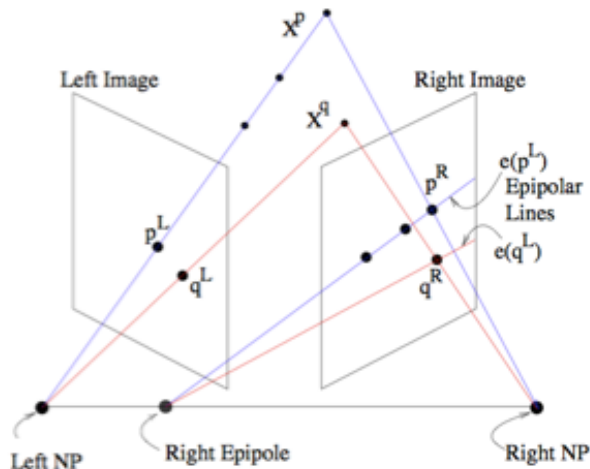


Projection, stereo and panoramic images

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



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.

<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.

9.2 The fundamental matrix F

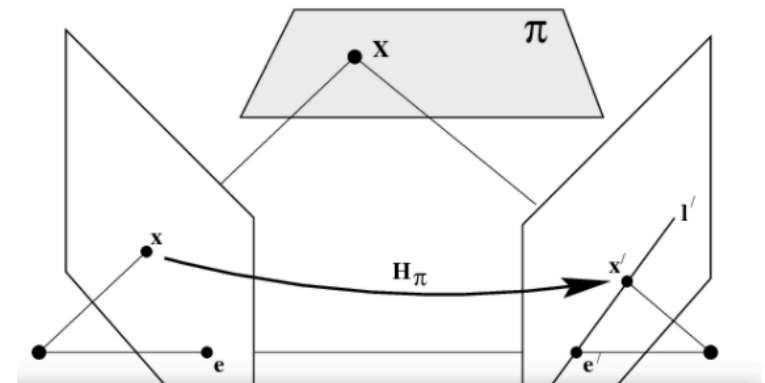


Fig. 9.5. A point x in one image is transferred via the plane π to a matching point x' in the second image. The epipolar line through x' is obtained by joining x' to the

Multiple View Geometry in Computer Vision by Wrobel, 2001

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. $\tilde{x}' = \tilde{x} + \tilde{t}$

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

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

where I is the (2x2) identity matrix

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

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

preserves parallel property of lines.

2D Rotation + translation (a.k.a. rigid body motion, a.k.a. 2D euclidian):
 $\tilde{x}' = R\tilde{x} + \tilde{t}$

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

2D Scaled rotation (a.k.a. similarity transform): $\tilde{x}' = sR\tilde{x} + \tilde{t}$

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

preserves angles between lines.

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

Note that \tilde{H} is homogeneous, i.e., it is only defined up to a scale, Two \tilde{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.






| Transformation | Matrix | # DoF | Preserves | Icon |
|-------------------|--|-------|----------------|---|
| translation | $\begin{bmatrix} I & t \end{bmatrix}_{2 \times 3}$ | 2 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} R & t \end{bmatrix}_{2 \times 3}$ | 3 | lengths |  |
| similarity | $\begin{bmatrix} sR & t \end{bmatrix}_{2 \times 3}$ | 4 | angles |  |
| affine | $\begin{bmatrix} A \end{bmatrix}_{2 \times 3}$ | 6 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{H} \end{bmatrix}_{3 \times 3}$ | 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.

Planar surface flow: Bilinear interpolant:

$$x' = a_0 + a_1x + a_2y + a_6x^2 + a_7xy$$

$$y' = a_3 + a_4x + a_5y + a_7x^2 + a_6xy$$

planar surface with a small 3D motion

Bilinear interpolant:

$$x' = a_0 + a_1x + a_2y + a_6xy$$

$$y' = a_3 + a_4x + a_5y + a_7xy$$

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

3D Translation. $\mathbf{x}' = \mathbf{x} + \mathbf{t}$

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

where \mathbf{I} is the (3×3) identity matrix

3D Affine: $\mathbf{x}' = \mathbf{A}\mathbf{x}$

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

preserves parallel property of lines and planes.

3D Rotation + translation (a.k.a. rigid body motion, a.k.a. 3D euclidian): $\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{t}$

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

where \mathbf{R} is a 3 × 3 orthonormal rotation matrix with $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ and $|\mathbf{R}| = 1$.

can describe a rigid motion using

$$\mathbf{x}' = \mathbf{R}(\mathbf{x} - \mathbf{c}) = \mathbf{R}\mathbf{x} - \mathbf{R}\mathbf{c}$$

where \mathbf{c} is the center of rotation (often the camera center).

3D Scaled rotation (a.k.a. similarity transform): $\mathbf{x}' = \mathbf{sR}\mathbf{x} + \mathbf{t}$

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

preserves angles between lines and planes.

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

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

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

preserves straightness property of lines.






| Transformation | Matrix | # DoF | Preserves | Icon |
|-------------------|---|-------|----------------|---|
| translation | $\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{3 \times 4}$ | 3 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{3 \times 4}$ | 6 | lengths |  |
| similarity | $\begin{bmatrix} \mathbf{sR} & \mathbf{t} \end{bmatrix}_{3 \times 4}$ | 7 | angles |  |
| affine | $\begin{bmatrix} \mathbf{A} \end{bmatrix}_{3 \times 4}$ | 12 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{4 \times 4}$ | 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 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 $[\hat{n}]_{\times}$ be the matrix form of the cross product operator with the vector $\hat{n} = (\hat{n}_x, \hat{n}_y, \hat{n}_z)$,

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

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

good for small rotations

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

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

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. $\delta R = (\mathbf{I} + [\delta r]_{\times})$, 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” $q_0 = (0, 0, 0, 1)$.

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

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

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.

$$P = K[R|t]$$

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 $(x_0, y_0)^T$ 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 $P = [M|m]$, is given as

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

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

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:

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

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 $(x, y, z) \in [-1, -1] \times [-1, 1] \times [0, 1]$
- 2) rescale these coordinates to integer pixel coordinates using a *viewport* transformation

disparity, the inverse depth, is defined using $z_{\text{near}} = 1, z_{\text{far}} \rightarrow \infty$, 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.

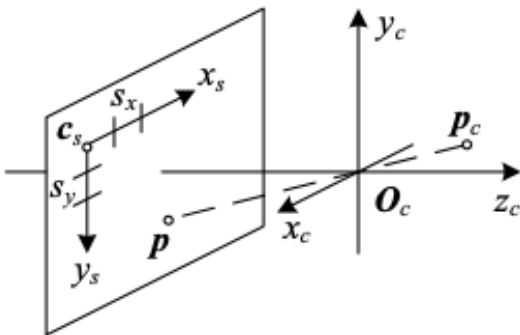


Fig 2.8 Projection of a 3D camera-centered point p_c onto the sensor planes at location p . O_c is the camera center (nodal point), c_s is the 3D origin of the sensor plane coordinate system, and s_x and s_y are the pixel spacings.

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.

for inhomogeneous and homogeneous coordinates, respectively:

$$\mathbf{p} = \left[\mathbf{R}_s \mid \mathbf{c}_s \right] \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix} = \mathbf{M}_s \tilde{\mathbf{x}}_s \quad \tilde{\mathbf{x}}_s = \alpha \mathbf{M}_s^{-1} \mathbf{p}_c = \mathbf{K} \mathbf{p}_c$$

where \mathbf{M}_s is sensor homography. parameterized by 8 unknowns: the 3 parameters describing the rotation \mathbf{R}_s , the 3 parameters describing the translation \mathbf{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)

\mathbf{O}_c is the camera projection center

\mathbf{c}_s is the original in \mathbf{O}_c

\mathbf{R}_s is the 3D rotation

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

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

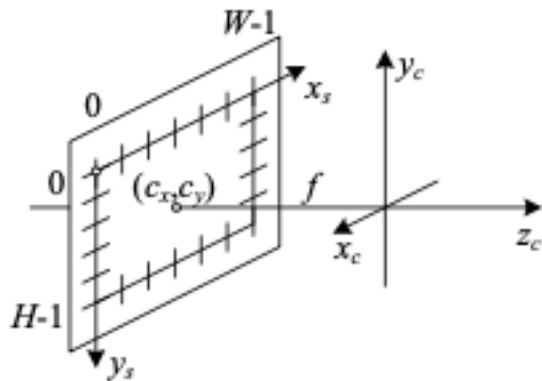
NOTE that multi-view geometry in practice treats \mathbf{K} as an upper-left triangular matrix with 5 degrees of freedom.

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.)

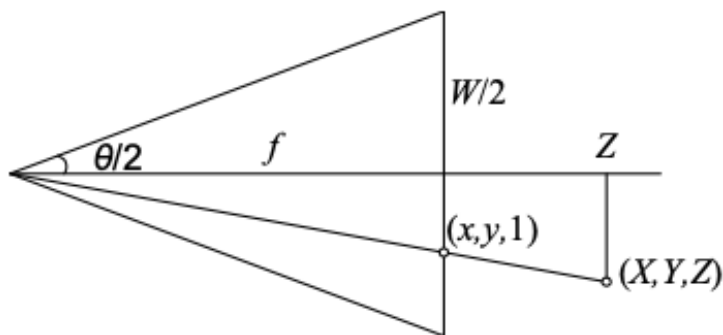
Fig 2.9 Simplified camera intrinsics showing the focal length f and the optical center (c_x, c_y) . The image width and height are W and H .



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

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

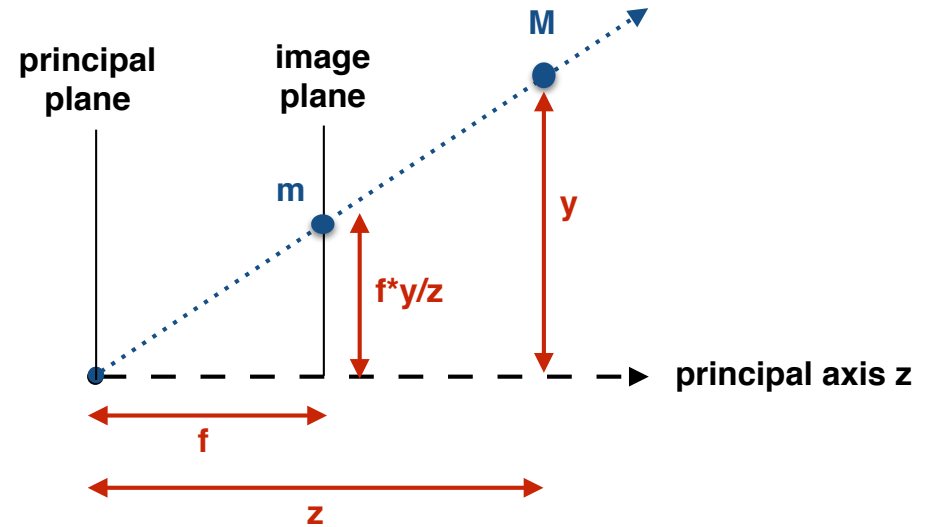
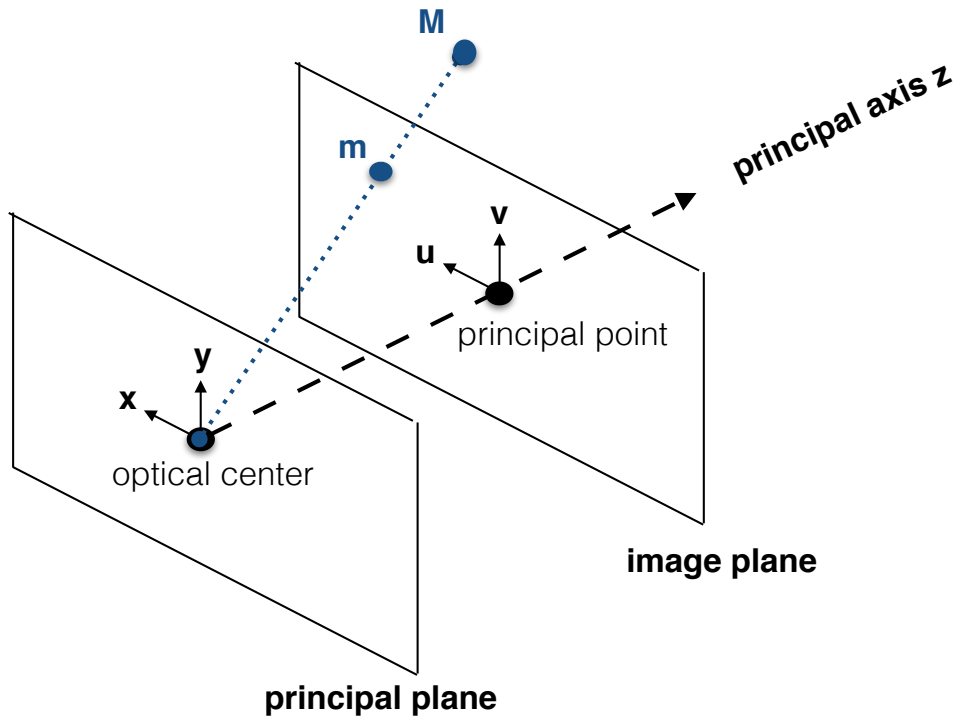
if set the origin at roughly the center of the image, e.g., $(c_x, c_y) = (W/2, H/2)$, where W and H are the image height and width, the camera model has a single unknown, i.e., the focal length f . θ is the FOV. $f = (W/2)(\tan(\theta/2))^{-1}$



camera matrix

$$P = K[R|t]$$

Pinhole camera geometry



camera projection matrix P is defined in $z^* \mathbf{m} = P^* \mathbf{M}$
 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 \mathbf{R} and translation vector \mathbf{t} to put the coordinates in the (external) world coordinate system. The matrix is $\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$
 The six parameters are called *external parameters*.
 rows of \mathbf{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.

$K = \begin{bmatrix} f/s_x & f/s_x \cot \theta & o_x \\ 0 & f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}$ K is the *camera calibration matrix*,
 result is pixel coords in image plane.

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

ϕ is the angle between the axes, usually 90 degrees

The aspect ratio s_x/s_y is usually 1

The extrinsic parameters depend upon Rotation and Translation of the camera.

P can be rewritten as $P = \mathbf{K}^* [\mathbf{I} \mid \mathbf{0}]^* \mathbf{G} = \mathbf{K}^* [\mathbf{R} \mid \mathbf{t}]$

A scale factor λ applied is $P = \lambda^* \mathbf{K}^* [\mathbf{R} \mid \mathbf{t}]$

This is a 3x4 full rank matrix and can be factorized by QR factorization.

P can be rewritten as $P = [\mathbf{P}_{3 \times 3} \mid \mathbf{p}_4]$ where $\mathbf{P}_{3 \times 3}$ is the first 3 rows of P and \mathbf{p}_4 is the 4th column.

If $\lambda=1$, the matrix is normalized and the distance of \mathbf{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 $\cos \theta = \mathbf{m1}^T * \omega * \mathbf{m2} / (\text{sqrt}(\mathbf{m1}^T * \omega * \mathbf{m1}) * (\text{sqrt}(\mathbf{m2}^T * \omega * \mathbf{m2}))$

extrinsic parameters are the position and orientation of the camera w.r.t. a coord frame.

(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)

projection and the principal and image planes

remembering that the center of projection is in the principal plane
one can define the position of m with respect to the 0,0 coordinates
in the image plane in which it is in.

u_c and v_c are the coordinates of the principal point.

m is at x, y in that image plane.

$$u = u_c + (x/\text{pixelWidth}) \quad \text{and} \quad v = v_c + (y/\text{pixelHeight})$$

Similarly, the coordinates in the world coordinate system can be transformed to the image plane coordinate system.

$$Z * u = Z * u_c + (X * f) / \text{pixelWidth}$$

$$Z * v = Z * v_c + (Y * f) / \text{pixelHeight}$$

using scaling factor:

$$\begin{bmatrix} \text{scale} * u \\ \text{scale} * v \\ \text{scale} \end{bmatrix} = \begin{bmatrix} (f/\text{pixelWidth}) & 0 & u_c & 0 \\ 0 & (f/\text{pixelHeight}) & v_c & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

camera calibration matrix:

$$C = \begin{bmatrix} r_1 * (f/\text{pixelWidth}) + u_c * r_3 & t_x * (f/\text{pixelWidth}) + u_c * t_z \\ r_2 * (f/\text{pixelHeight}) + v_c * r_3 & t_y * (f/\text{pixelHeight}) + v_c * t_z \\ r_3 & t_z \end{bmatrix}$$

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 | $\begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix}_{2 \times 3}$ | 2 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$ | 3 | lengths |  |
| similarity | $\begin{bmatrix} s\mathbf{R} & \mathbf{t} \end{bmatrix}_{2 \times 3}$ | 4 | angles |  |
| affine | $\begin{bmatrix} \mathbf{A} \end{bmatrix}_{2 \times 3}$ | 6 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{\mathbf{H}} \end{bmatrix}_{3 \times 3}$ | 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.

| Transform | Matrix | Parameters p | Jacobian 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 | $\begin{bmatrix} 1+h_{00} & h_{01} & h_{02} \\ h_{10} & 1+h_{11} & h_{12} \\ h_{20} & h_{21} & 1 \end{bmatrix}$ | $(h_{00}, h_{01}, \dots, h_{21})$ | (see Section 6.1.3) |

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

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 $\mathbf{X} = \{\mathbf{x}_i\}$ be a set of data points (vectors)

Let \mathbf{a} be a k-dimensional parameter vector to be estimated.

$r_i(\mathbf{a}) = r(\mathbf{x}_i, \mathbf{a})$ 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.

σ_i^2 = variance (scale) assoc. w/ scalar $r_{i,a}$

$\rho(u)$ = a robust loss function

$\hat{\mathbf{a}}$ is the M-estimator of \mathbf{a} .

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

$\psi(u) = \rho'(u)$ is essentially proportional to the influence function.

$w(u) \cdot u = \psi(u)$ is a weight function.

iteratively re-weighted least-squares (IRLS) alternates between calculating weights

$w_i = w(r_{i,a}/\sigma_i)$ using the current estimate of \mathbf{a} then solving following eqn. to estimate

\mathbf{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] | $\rho(u) = \begin{cases} \frac{a^2}{6} [1 - (1 - (\frac{u}{a})^2)^3] & u \leq a \\ \frac{a^2}{6} & u > a \end{cases}$ | $\psi(u) = \begin{cases} u [1 - (\frac{u}{a})^2]^2 & u \leq a \\ 0 & u > a \end{cases}$ |
| Cauchy [32] | $\rho(u) = \frac{b^2}{2} \log[1 + (\frac{u}{b})^2]$ | $\psi(u) = \frac{u}{1 + (u/b)^2}$ |
| Huber [34, Chapter 7] | $\rho(u) = \begin{cases} \frac{1}{2}u^2 & u \leq c \\ \frac{1}{2}c(2 u - c) & c < u \end{cases}$ | $\psi(u) = \begin{cases} u & u \leq c \\ c \frac{u}{ u } & u > c \end{cases}$ |

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.

$\hat{\mathbf{a}}$ is the LMS estimate of \mathbf{a} . $\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmin}} \operatorname{median}_{\mathbf{x}_i \in X} r_{i,\mathbf{a}}^2$.

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, $\hat{\mathbf{a}}$. 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 all “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 |

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.):

$\hat{\mathbf{a}}$ can be used to refine the fit.

s^* is the subset used to form $\hat{\mathbf{a}}$.

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

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

if all N points are sampled from a normal distribution with variance σ^2 , then $\sigma^{\wedge} \rightarrow \sigma$ as $N \rightarrow \text{infinity}$.
 can choose data points such that $(r_{i\mathbf{a}_s})^2 < (\theta \sigma^{\wedge})^2$ where θ is constant, typically about 2.5 then re-estimate $\hat{\mathbf{a}}$.

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 .

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.)

using corresponding points to establish the relationship between two different images of a scene taken with uncalibrated cameras.

The features are intensity corners or other distinctive locations detected by an interest operator.

Correspondence for each x_i is found by searching an area of image 2 established by the range of scene depths and camera motions determined a priori. Within this area, which is sometimes as small as 30 pixels on a side, but could be much larger depending on what restrictions are placed on camera motion, x_{0i} is the location of the image feature most similar to that of x_i , as decided by, for example, correlation of the surrounding image regions.

Let $\{\tilde{\mathbf{x}}_i\}$ be a set of features in image 1 in homogeneous coordinates

Let $\{\tilde{\mathbf{x}}'_i\}$ be a set of features in image 2 in homogeneous coordinates

Fundamental Matrix: $\tilde{\mathbf{x}}_i^T \mathbf{F} \tilde{\mathbf{x}}'_i = 0$ where \mathbf{F} is a 3x3, rank 2 matrix

\mathbf{F} has 7 degrees of freedom so can be estimated from at least 7 correspondence pairs.

For any point, $\tilde{\mathbf{x}}'$, in image 2, $\mathbf{F}\tilde{\mathbf{x}}'$ defines a line in image 1 through which the point corresponding to $\tilde{\mathbf{x}}'$ must pass. Similarly, for any point, $\tilde{\mathbf{x}}$, in image 1, $\mathbf{F}^T \tilde{\mathbf{x}}$ defines a line in image 2 through which the point corresponding to $\tilde{\mathbf{x}}$ must pass. All such lines, known as epipolar lines, pass through the epipoles, which are the null space of \mathbf{F}^T and \mathbf{F} , respectively.

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19,
NO. 6, JUNE 1997

"In Defense of the Eight-Point Algorithm" by Richard Hartley

(see src/.../EpipolarTransformer.java)

$$u_1 = (x_1, y_1, 1)^T$$

$$u_2 = (x_2, y_2, 1)^T$$

$$x_1 x_2 F_{11} + x_1 y_2 F_{21} + x_1 F_{31} + y_1 x_2 F_{12} + y_1 y_2 F_{22} + y_1 F_{32} + x_2 F_{13} + y_2 F_{23} + F_{33} = 0$$

$A * f = 0$ is the algebraic residual (a.k.a. fitting error, algebraic distance)

where $A = [x_1 x_2, x_1 y_2, x_1, y_1 x_2, y_1 y_2, y_1, x_2, y_2, 1]$

To avoid the trivial scale, $\|f\| = 1$ where f is the norm of f

And we need **least squares fits** because the set may be over determined and not have a zero solution: the vector f that minimizes $\|Af\|$ subject to the constraint $\|f\| = f^T f = 1$

the solution is the unit eigenvector corresponding to the smallest eigenvalue of $A^T A$.

Since $A^T A$ is semi-definite and symmetric, all of its eigenvectors are real and positive or zero. This eigenvector is what he calls the least eigenvector of $A^T A$ and it is found via the Jacobi algorithm or Singular Value Decomposition.

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19,
NO. 6, JUNE 1997

"In Defense of the Eight-Point Algorithm" by Richard Hartley

The solved for matrix will in general not have rank 2 and needs to, so further corrections are necessary:

matrix F is replaced by F' that minimizes the Frobenius Norm ($= \sqrt{\text{double summation of } i=1 \text{ to } i=n, j=1 \text{ to } j=m \text{ of } |a_{ij}|^2}$)

$\|F - F'\|$ subject to the condition $\det F' = 0$.

A convenient method of doing this is to use the Singular Value Decomposition (SVD).

let $F = U \cdot D \cdot V^T$ be the SVD of F , where D is diagonal matrix

$D = \text{diag}(r, s, t)$ satisfying $r \geq s \geq t$.

let $F' = U \cdot \text{diag}(r, s, 0) \cdot V^T$.

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19,
NO. 6, JUNE 1997

"In Defense of the Eight-Point Algorithm" by Richard Hartley
(see src/.../EpipolarTransformer.java)

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

- (1) Transforming the coordinates:
 Normalization for isotropic scaling.
 $utrans = T * u \Rightarrow u = utrans * inv(T)$
 $u_2^T * F * u_1 = 0$
 becomes $utrans_2^T * inv(T_2) * F * inv(T_1) * utrans_1 = 0$
 and $inv(T_2) * F * inv(T_1)$ is the fundamental matrix for
 $utrans_2 < -- > utrans_1$ which when found, will be subsequently
 denormalized.
 - a) points are translated so that their centroid is at the origin.
 - b) points are then scaled so that the average distance from the
 origin is $\sqrt{2}$
 - c) the transformation is applied to each of the 2 images separately.
- (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 8. **f has rank 2.**
 calculate [U,D,V] from $svd(A)$
- (4) **make the fundamental matrix have a rank of 2**
 by performing a svd and then reconstructing with the two largest
 singular values.
 $[U,D,V] = svd(F,0);$
 $F = U * diag([D(1,1) D(2,2) 0]) * V^T;$
- (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) 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.

(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.
 $\text{solve } A = U * D * V^T \text{ for } A*f = [..x...]*f = 0$
 A has rank 7. **f has rank 2.** calculate [U,D,V] from svd(A)
- (4) make the fundamental matrix have a rank of 2 by performing a svd and then reconstructing with the two largest singular values. $[U,D,V] = \text{svd}(F,0)$;
 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 u_1, u_2, \dots, u_{n-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 $[3 \times 3]$ as F1 and F2.
 A linear convex combination of F1 and F2 form the estimate of F.
 $F = \alpha * F1 + (1 - \alpha) * F2$ where α is between 0 and 1
 The eigenvalues of F are possible only if the determinant of F is 0.
 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 $\text{determinant}(F) = 0$ requires the cubic
 roots of the equation.
 After the cubic root(s) are solved, they are back substituted into :
 $F_i = a(i) * FF\{1\} + (1-a(i)) * FF\{2\}$;
 to get the solutions F_i which may be one or 3 solutions
- (5) denormalize the fundamental matrix
 The related part of the normalization equation: $\text{inv}(T_2) * F * \text{inv}(T_1)$
 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.

multi-view geometry

With point correspondences $\mathbf{m}_i \longleftrightarrow \mathbf{M}_i$, can solve for camera projection matrix P . Using the Kroenecker delta product and vec operator, coefficients are rewritten to their linearly independent forms of a matrix of size $2n \times 12$ called the coefficient matrix A

From a set of n point correspondences, we obtain a $2n \times 12$ coefficient matrix A , where n must be > 6 .

In general A will have rank 11 (provided that the points are not all coplanar) and the solution is the *1-dimensional right null-space of A* .

The linear system of equations for inexact data is solved using least squares.

The least-squares solution for $\text{vec } P^T$ is the singular vector corresponding to the smallest *singular value* decomposition of A and the direct linear transform.

The equation is usually written in form $A * h = 0$ where h is $\text{vec } P^T$ (its the one dimensional reshaping of the homograph matrix).

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*: $\mathbf{m}_r^T * \mathbf{E} * \mathbf{m}_l^T$
- 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 $\mathbf{l}_r = \mathbf{F} * \mathbf{m}_l$ and vice versa $\mathbf{l}_l = \mathbf{F}^T * \mathbf{m}_r$
- the *essential and fundamental matrices* are related through $\mathbf{F} = \mathbf{K}_r^{-T} * \mathbf{E} * \mathbf{K}_l^{-T}$ where \mathbf{K}_r and \mathbf{K}_l 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 \mathbf{R} can be estimated, but \mathbf{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 “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.)

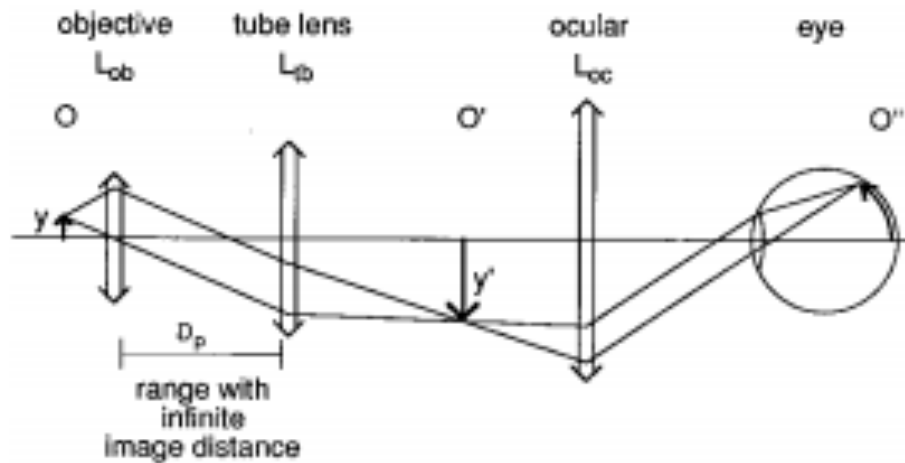


FIGURE 4 Ray path in microscope with infinity-corrected objective and tube lens.

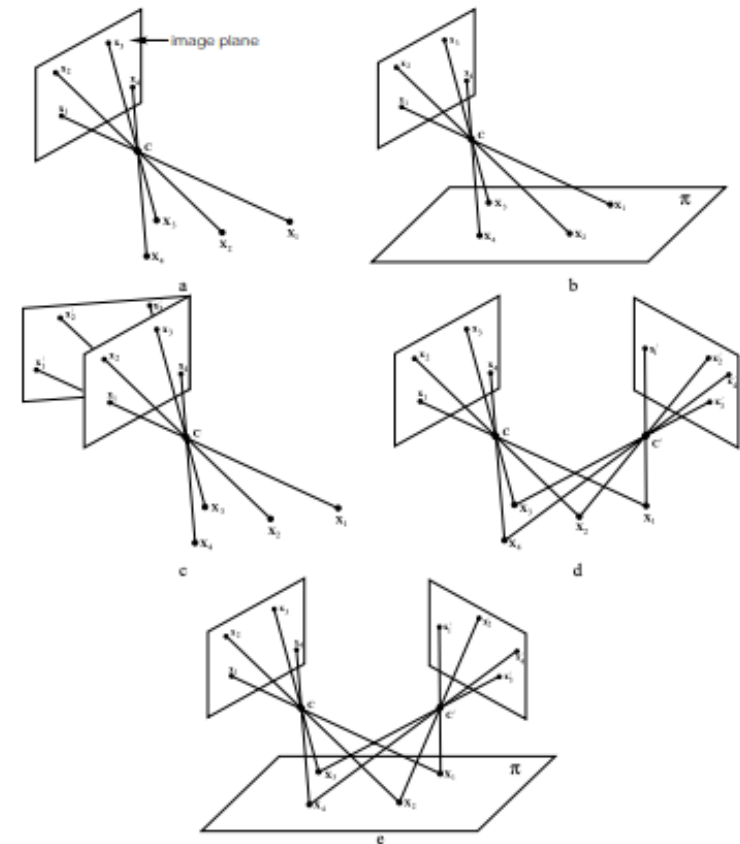


Fig. 1.1. **The camera centre is the essence.** (a) Image formation: the image points x_i are the intersection of a plane with rays from the space points X_i through the camera centre C . (b) If the space points are coplanar then there is a projective transformation between the world and image planes, $x_i = H_{3 \times 3} X_i$. (c) All images with the same camera centre are related by a projective transformation, $x_i = H_{3 \times 3} x_i$. 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.

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.)

9.3 Fundamental matrices arising from special motions

247

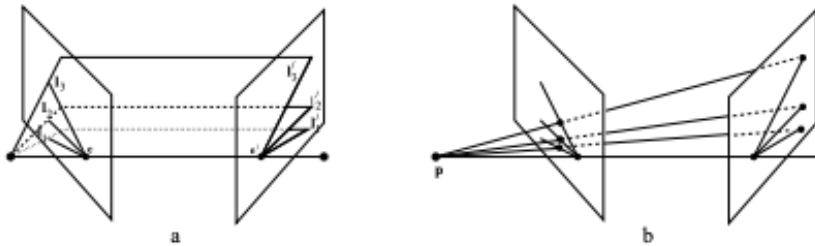


Fig. 9.6. **Epipolar line homography.** (a) There is a pencil of epipolar lines in each image centred on the epipole. The correspondence between epipolar lines, $l_i \leftrightarrow l'_i$, is defined by the pencil of planes with axis the baseline. (b) The corresponding lines are related by a perspectivity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

246

9 Epipolar Geometry and the Fundamental Matrix

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

Table 9.1. Summary of fundamental matrix properties.

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.)

(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.

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

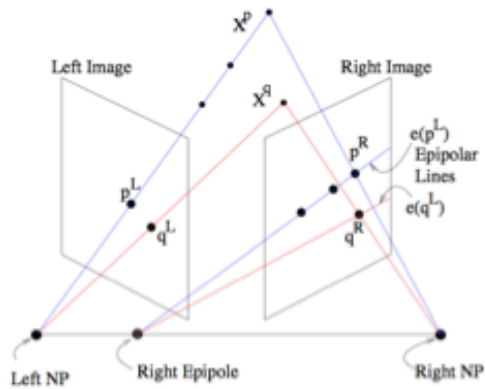
by Agarwal, Snavely, Seitz, and Szeliski

<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

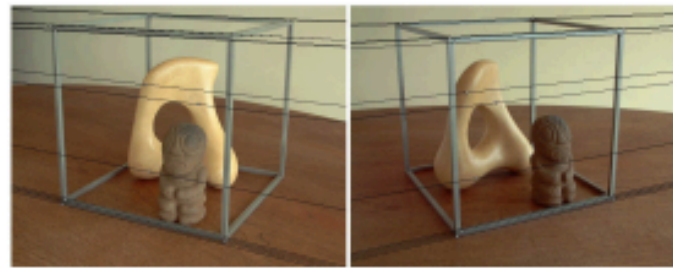
from “Computing Rectifying Homographies for **Stereo Vision**”

by Loop & Zhang, 1999

<https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr99-21.pdf>

the transforms H_p and H_{0p} were found that map the epipoles e and e_0 to points at 1. In this section we define a pair of similarity transforms H_r and H_{0r} 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 overlaid with several epipolar lines.



(b) Image pair transformed by the specialized projective mapping H_p and H_{0p} . Note that the epipolar lines are now parallel to each other in each image.



(c) Image pair transformed by the similarity H_r and H_{0r} . Note that the image pair is now rectified (the epipolar lines are horizontally aligned).



(d) Final image rectification after shearing transform H_s and H_{0s} . 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.

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.

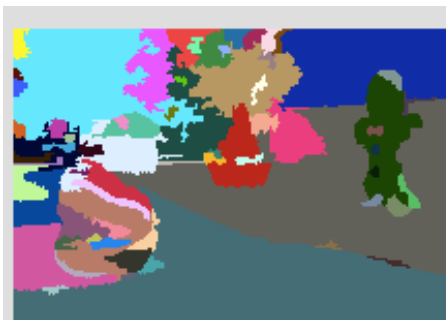
So, a rough calculation of the angle of inclination is needed to find the other matching points consistent with a single-point perspective projection.

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.

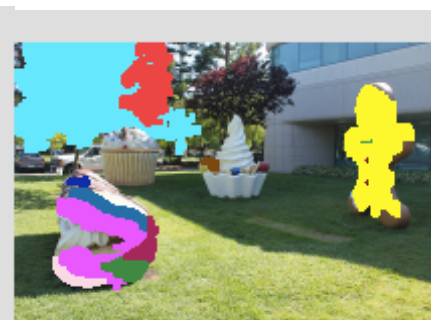
The foreshortening along the y axis can be approximated by the ratio of the far sides of two triangles, one embedded in the other.



**image for the search for
gingerbread man**

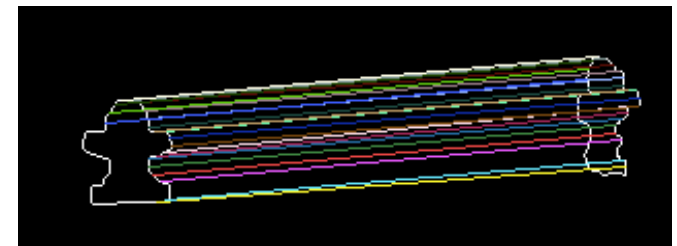


segmentation



**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**



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 here).

