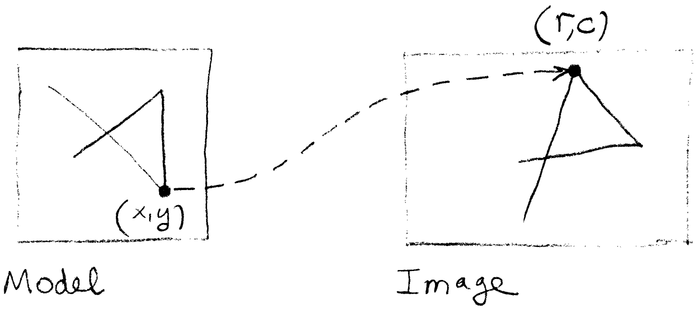
Ch. 11 Matching in 2-D



mapping from one 2-D coordinate space to another is called a <u>2-D transformation</u>

<u>Image registration</u> process by which points of two images from similar viewpoints of basically the same scene are geometrically transformed so that corresponding feature points of the two images have the same coordinates after transformation

11.2 Representation of points

If $\underline{P}_j = [x, y] = \begin{bmatrix} x \\ y \end{bmatrix}$ is some feature point and $\underline{C}$ is some reference frame, then we denote the <u>coordinates</u> of the point relative to the coordinate system as ${}^{C}\underline{P}_j$

The <u>homogeneous coordinates</u> of a 2-D point $\underline{P} = \begin{bmatrix} x \\ y \end{bmatrix}$ are

$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix}$ where s is a scale factor, usually 1.

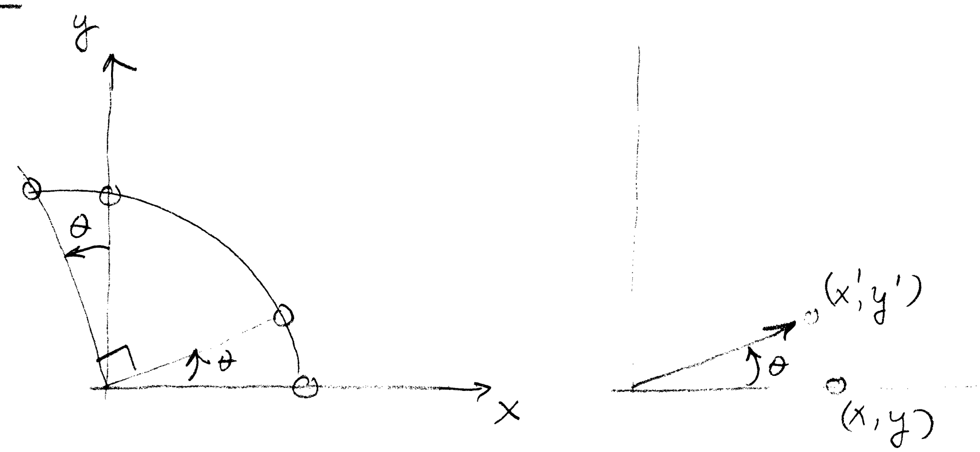
11.3 Affine Mapping Functions

represent spatial transformations by multiplication of a matrix and a homogeneous point

Scaling

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cx \\ cy \end{bmatrix} = \underbrace{\begin{bmatrix} c & 0 \\ 0 & c \end{bmatrix}}_{\text{scaling matrix}} \begin{bmatrix} x \\ y \end{bmatrix}$$

rotation



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = R_\theta \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ 1 \end{bmatrix}$$

Can't do a translation with a 2x2, We have to use a 3x3.

Rotation, Scaling, Translation

$$\begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s & 0 & 0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

$\xrightarrow{\hspace{6em}}$ performed in order

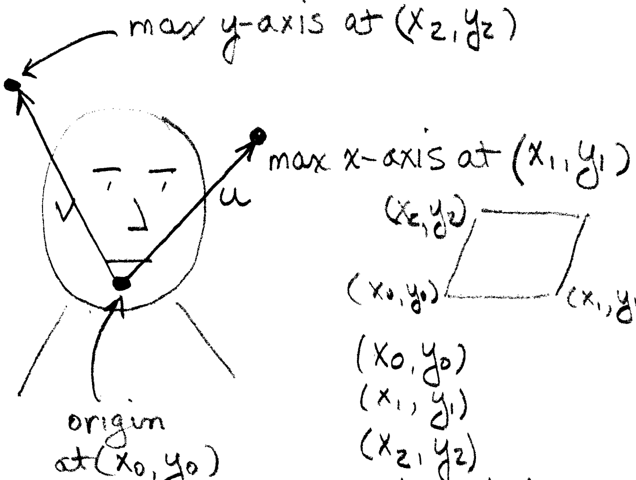
↑ world coordinates

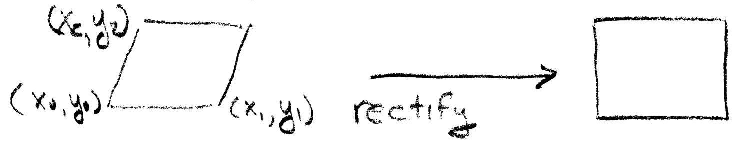
↑ image coordinates

In this transformation there are only 4 unknowns (θ, s, x_0, y_0) as compared to a polynomial warp which had 18.

Example affine warp

Use affine transformation to extract data



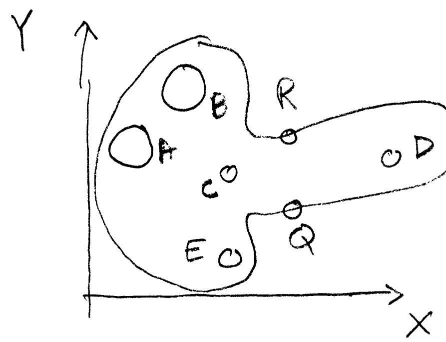


(x_0, y_0)
(x_1, y_1)
(x_2, y_2)
are in original (x, y)
coordinate system.

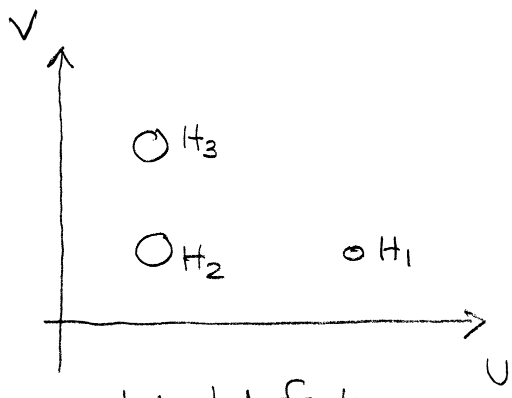
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_1 - x_0}{n} & \frac{x_2 - x_0}{m} & x_0 \\ \frac{y_1 - y_0}{n} & \frac{y_2 - y_0}{m} & y_0 \\ 0 & 0 & 1 \end{bmatrix} \underbrace{\begin{bmatrix} r \\ c \\ 1 \end{bmatrix}}_{\text{row-column in original image}}$$

the output is $m \times n$ pixels
the input is simply distance $x_0 - x_1$, etc.

You can use affine transformations for recognition as well as location.



Model object



detected features
is this the model?

MODEL POINT LOCATIONS

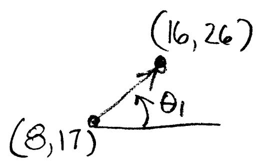
Point	Coordinates	to A	to B	to C	to D	to E
A	(8,17)	0	12	15	37	21
B	(16,26)	12	0	12	30	26
C	(23,16)	15	12	0	22	15
D	(45,20)	37	30	22	0	30
E	(22,1)	21	26	15	30	0

IMAGE POINT	LOCATIONS	to H_1	to H_2	to H_3
H_1	(31,9)	0	21	26
H_2	(10,12)	21	0	12
H_3	(10,24)	26	12	0

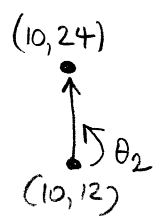
Note: coordinates and distances are for centers of holes.

Assume calibrated camera, scale factor corrected to 1

① Consider angle A → B $\quad \theta_1 = \tan^{-1}\left(\frac{26-17}{16-8}\right) = \tan^{-1}\left(\frac{9}{8}\right) = 0.844$



② Let's test $H_2 \to H_3$ in image



obviously $\theta_2 = \tan^{-1}\left(\frac{24-12}{10-10}\right) = \frac{\pi}{2}$

if $H_2 \to H_3$ matches $A \to B$ the rotation is $\theta_2 - \theta_1 = 1.571 - .844 = .727$

If this is correct we solve the affine transformation for u_0, v_0

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 10 \\ 12 \\ 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & u_0 \\ \sin\theta & \cos\theta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 17 \\ 1 \end{bmatrix}$$

coordinates of A

coordinates of H_2 (10,12)

This rotates A by 0.727 radians and translates by (u_0, v_0)

Solving this equation gives $u_0 = 15.3$, $v_0 = -5.95$

To test this we try the other points and see if they match

This process is called recognition by alignment.

Shear and reflection

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ e_u & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ e_u u + v \\ 1 \end{bmatrix}$$ v-axis shear

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & e_v & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u + e_v v \\ v \\ 1 \end{bmatrix}$$ u-axis shear

In a shear the displacement is proportional to their distance for the u or v axis.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ -v \\ 1 \end{bmatrix}$$ reflection about u-axis

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} -u \\ v \\ 1 \end{bmatrix}$$ reflection about v-axis

11.4 Best affine transformation

general 2D affine transform

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

six unknowns requires 3 matching pairs of points $\begin{bmatrix} x_j \\ y_j \end{bmatrix}, \begin{bmatrix} u_j \\ v_j \end{bmatrix}$

You can solve this just like we did for polynomial warps for more than 3 points.

$$u_j = a_{11} x_j + a_{12} y_j + a_{13}$$
$$v_j = a_{21} x_j + a_{22} y_j + a_{23}$$

Rearranging since the a's are the unknowns.

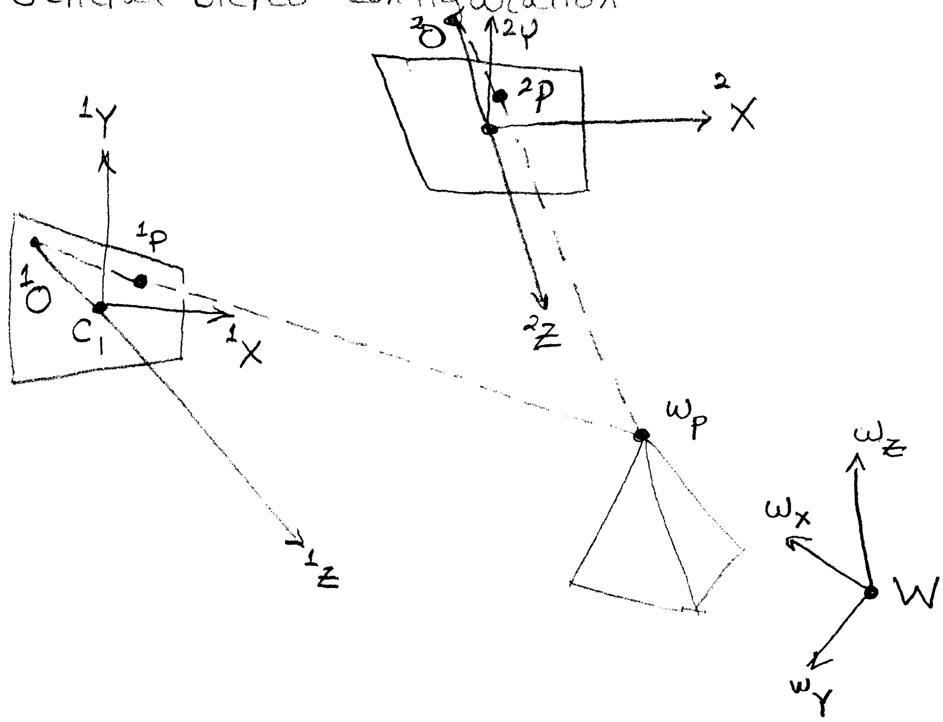
$$\begin{bmatrix} x_j & y_j & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_j & y_j & 1 \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{21} \\ a_{22} \\ a_{23} \end{bmatrix} = \begin{bmatrix} u_j \\ v_j \end{bmatrix}$$

We can simply use M points pairs to get 2M equations and solve by the pseudoinverse.

Equation (11.17) of Shapiro and Stockman gives the matrix for a least squared error solution.

Ch. 13 3-D Sensing and Object Pose Computation

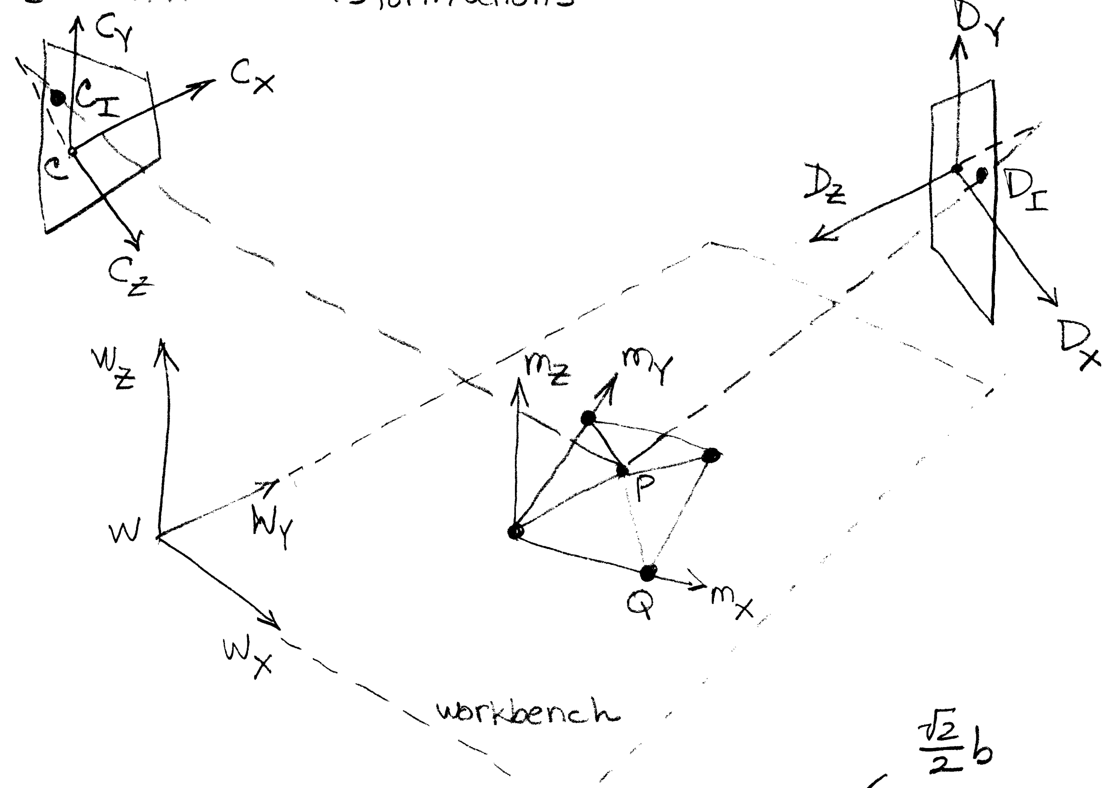
13.1 General Stereo Configuration

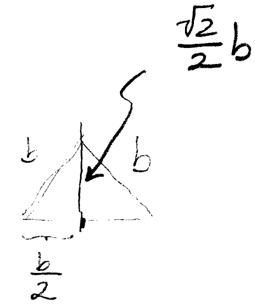


basic ideas

1. coordinates of 3-D points are computed by intersecting the two imaging rays from the corresponding image points
2. one of the cameras can be replaced by a projector — laser beam, grid, or something similar
3. if you know something about the object such as its height you can eliminate one of the cameras.

13.2 3-D Affine transformations





in it's own frame m the vertex of the pyramid

$$^{M}\underline{P} = \begin{bmatrix} {}^{m}p_x \\ {}^{m}p_Y \\ {}^{m}p_z \end{bmatrix} = \begin{bmatrix} b/2 \\ b/2 \\ \frac{\sqrt{2}}{2}b \end{bmatrix}$$

representation of pyramid apex relative to workbench

$$^{W}\underline{P} = \begin{bmatrix} {}^{W}p_x \\ {}^{W}p_Y \\ {}^{W}p_z \end{bmatrix} = \underbrace{TR} \begin{bmatrix} b/2 \\ b/2 \\ \frac{\sqrt{2}}{2}b \end{bmatrix}$$

$$^{W}\underline{P} = {}^{W}_{M}\underline{\underline{T}}\;{}^{M}\underline{P}$$

where ${}^{W}_{M}\underline{\underline{T}}$ is the transformation from model to world coordinates (in 3D)

This notation is due to Craig, Intro to Robotics

13.2.2. Translation

$$\underline{{}^2P} = \underline{\underline{T}}(x_0, y_0, z_0)\, \underline{{}^1P}$$

$$\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} = \begin{bmatrix} {}^1P_x + x_0 \\ {}^1P_y + y_0 \\ {}^1P_z + z_0 \\ 1 \end{bmatrix}$$

13.2.3 Scaling

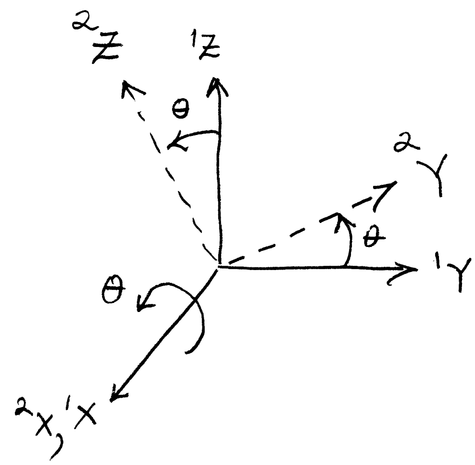
$$\underline{{}^2P} = \underline{\underline{S}}(s_x, s_y, s_z)\, \underline{{}^1P}$$

$$\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x\, {}^1P_x \\ s_y\, {}^1P_y \\ s_z\, {}^1P_z \\ 1 \end{bmatrix}$$

13.2.4 Rotation

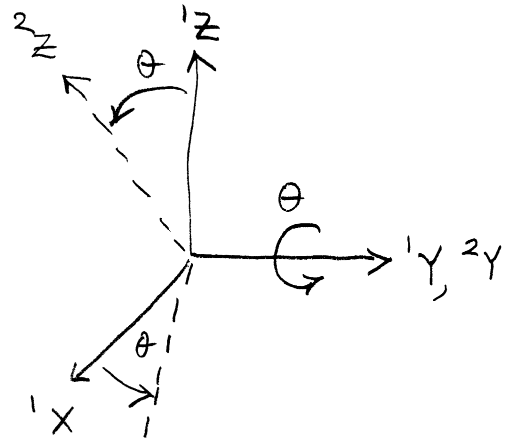
$$ {}^2\underline{P} = \underline{\underline{R}}({}^1X, \theta)\, {}^1\underline{P} $$

$$ \begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} $$



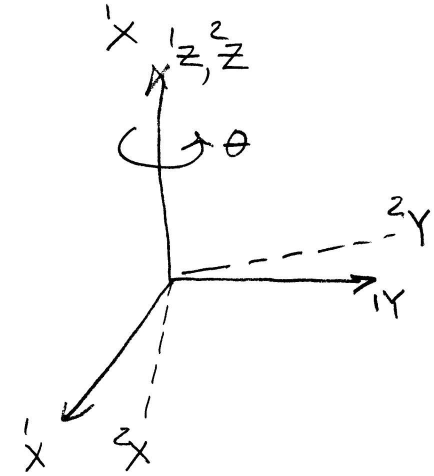
$$ {}^2\underline{P} = \underline{\underline{R}}({}^1Y, \theta)\, {}^1\underline{P} $$

$$ \begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} $$



$$ {}^2\underline{P} = \underline{\underline{R}}({}^1Z, \theta)\, {}^1\underline{P} $$

$$ \begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} $$

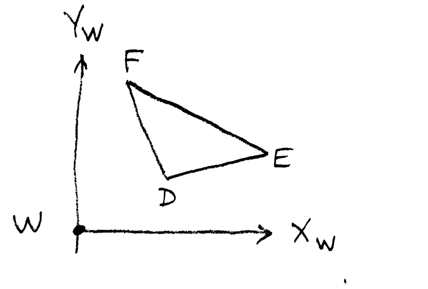


Arbitrary rotation + translation $\quad {}^2\underline{P} = \underline{\underline{R}}(A, \theta)\, {}^1P$

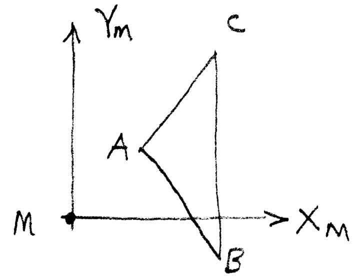
$$ \begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} $$

13.2.6 Alignment via Transformation Calculus

provides the basis for aligning any rigid model via correspondences between three model points and three sensed points

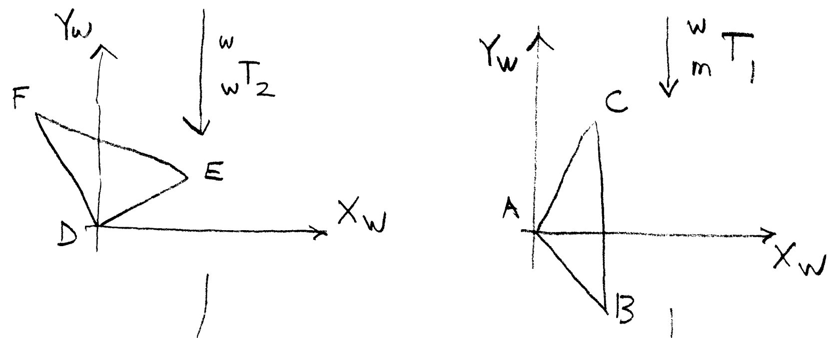


sensed world coordinates

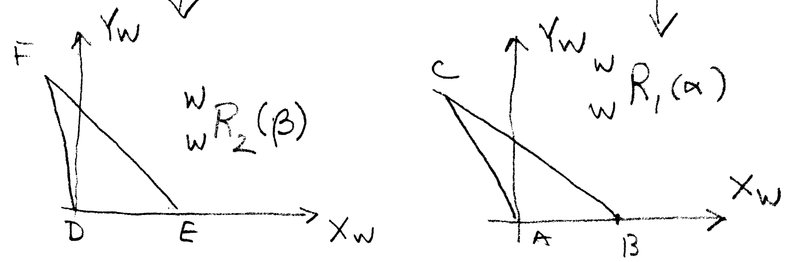


model points

① Input data points and sensed points.

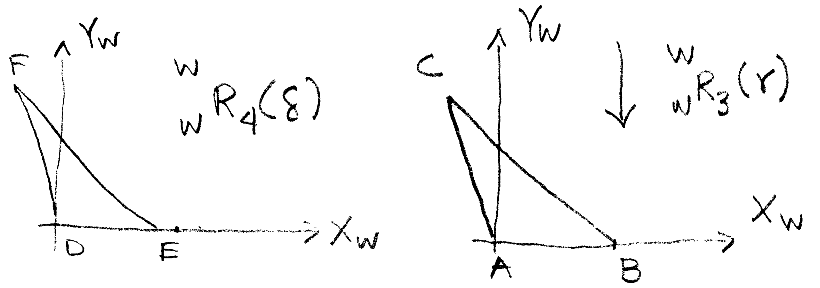


② shift sensed object and model to origin in world coordinates



③ rotate sides to X axis in world coordinates. This aligns origin and one side.

Note both are z rotations.



④ Now rotate about X axis to put triangles in XY plane.

Then

$$^{W}_{W}R_4 \; ^{W}_{W}R_2 \; ^{W}_{W}T_2 \; ^{W}P_i = \; ^{W}_{W}R_3 \; ^{W}_{W}R_1 \; ^{W}_{M}T_1 \; ^{M}P_i$$

Or, to map the model to the world

$$^{W}P_i = T_2^{-1} R_2^{-1} R_4^{-1} R_3 R_1 T_1 \; ^{M}P_i$$

13.3 The camera model used for 3-D measurements

$$ {}^{I}P = {}^{I}_{W}C\ {}^{W}P $$

measuredworld coordinates

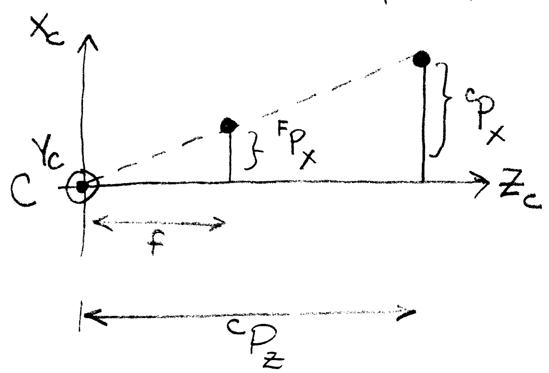
the camera matrix from world to image coordinates

measured (imaged) image coordinates

$$\begin{bmatrix} s\,{}^{I}P_r \\ s\,{}^{I}P_c \\ s \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} {}^{W}P_x \\ {}^{W}P_y \\ {}^{W}P_z \\ 1 \end{bmatrix}$$

usually this scale factor is set to 1 as well.

To prove this consider perspective



world & camera coordinates are identical

by similar triangles $\quad \dfrac{{}^{F}P_x}{f} = \dfrac{{}^{C}P_x}{{}^{C}P_z} \quad$ and (not shown) $\quad \dfrac{{}^{F}P_Y}{f} = \dfrac{{}^{C}P_Y}{{}^{C}P_Z}$

$$ {}^{F}P_x = \frac{f}{{}^{C}P_z}\,{}^{C}P_x \qquad {}^{F}P_Y = \frac{f}{{}^{C}P_Z}\,{}^{C}P_Y $$

A camera matrix for perspective

$$\begin{bmatrix} s\,{}^{F}P_x \\ s\,{}^{F}P_Y \\ s\,{}^{F}P_z \\ s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} {}^{C}P_x \\ {}^{C}P_Y \\ {}^{C}P_z \\ 1 \end{bmatrix}$$

Camera matrix for rotation translation $\quad {}^{C}P = T(t_x, t_y, t_z)\,R(\alpha, \beta, \gamma)\,{}^{W}P$

The general idea is to model the camera by the geometric transformations followed by a perspective transform.

Three rotations and three translations will look like.

$$\begin{bmatrix} {}^{C}p_x \\ {}^{C}p_y \\ \cancel{{}^{C}p_z} \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ \cancel{r_{31}} & \cancel{r_{32}} & \cancel{r_{33}} & \cancel{t_z} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^{W}p_x \\ {}^{W}p_y \\ {}^{W}p_z \\ 1 \end{bmatrix}$$

drop this row because there is only a constant value for z.

$$\begin{bmatrix} s\,{}^{F}p_x \\ s\,{}^{F}p_y \\ s \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & 1 \end{bmatrix} \begin{bmatrix} {}^{W}p_x \\ {}^{W}p_y \\ {}^{W}p_z \\ 1 \end{bmatrix}$$

does everything but the scaling.

But scaling is a simple matrix and this includes inversion.

$${}^{I}p = \begin{bmatrix} sr \\ sc \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -\frac{1}{d_y} & 0 \\ \frac{1}{d_x} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{S} \begin{bmatrix} s\,{}^{F}p_x \\ s\,{}^{F}p_y \\ s \end{bmatrix}$$

Final camera matrix

$${}^{I}p = \underbrace{{}^{I}_{F}S}_{\text{scaling}}\ \underbrace{{}^{F}_{C}\Pi}_{\text{perspective}}\ \underbrace{{}^{C}_{W}TR}_{\text{translation rotation}}\ {}^{W}p$$

This is an 11-parameter camera model based on affine transforms and perspective

13.4 Best Affine calibration matrix

Use a calibration jig with well known points!

For each calibration point j

$$\begin{bmatrix} su_j \\ sv_j \\ s \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_j \\ z_j \\ 1 \end{bmatrix}$$

or $su_j = c_{11} x_j + c_{12} y_j + c_{13} z_j + c_{14}$

$$s v_j = c_{21} x_j + c_{22} y_j + c_{23} z_j + c_{24}$$

$$s = c_{31} x_j + c_{32} y_j + c_{33} z_j + 1$$

$$(c_{31} x_j + c_{32} y_j + c_{33} z_j + 1)\, u_j = c_{11} x_j + c_{12} y_j + c_{13} z_j + c_{14}$$

$$(c_{31} x_j + c_{32} y_j + c_{33} z_j + 1)\, v_j = c_{21} x_j + c_{22} y_j + c_{23} z_j + c_{24}$$

Rearranging

$$x_j c_{11} + y_j c_{12} + z_j c_{13} + c_{14} - x_j u_j c_{31} - y_j u_j c_{32} - z_j u_j c_{33} = u_j$$

$$x_j c_{21} + y_j c_{22} + z_j c_{23} + c_{24} - x_j v_j c_{31} - y_j v_j c_{32} - z_j v_j c_{33} = v_j$$