Spatial transforms

We have already done geometric transforms of form

$$g(x,y) = f(x',y') = f\,[a(x,y),\ b(x,y)].$$

Consider transforms of form

$$\left.\begin{aligned} a(x,y) &= x + x_0 \\ b(x,y) &= y + y_0 \end{aligned}\right\}\ \text{a translation of } (x,y) \text{ to } (x+x_0, y+y_0)$$

Homogeneous coordinates

These transformations can be represented in matrix form:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \quad \text{for any point } (x,y)$$

$$\text{multiplying out}\quad \left.\begin{aligned} a(x,y) &= x + x_0 \\ b(x,y) &= y + y_0 \end{aligned}\right\}\ \text{this is a translation}$$

However, we cando other types of transforms

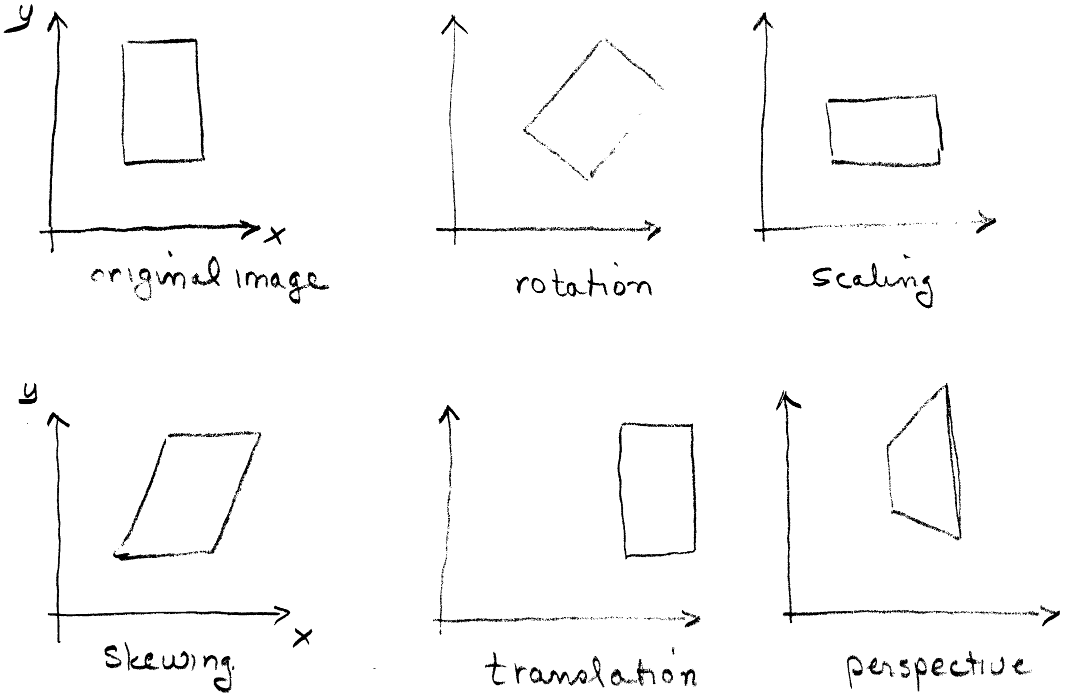
$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{c} & 0 & 0 \\ 0 & \frac{1}{d} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\text{gives}\quad \left.\begin{aligned} a(x,y) &= \frac{x}{c} \\ b(x,y) &= \frac{y}{d} \end{aligned}\right\}\ \text{this is a scaling}$$

general geometric transforms

Ballard & Brown, Ch. 2

p. 17-22 optics
p. 467 homogeneous coordinates
p. 477 geometric transformations



original image

rotation

scaling

skewing

translation

perspective

Ballard & Brown:

use linear transformations to relate object and image coordinates

homogeneous coordinates – we may not want to pivot or rotate about the origin

$$(x, y) \longrightarrow (a, b, c) \quad \text{where} \quad x = \frac{a}{c}, \; y = \frac{b}{c}$$

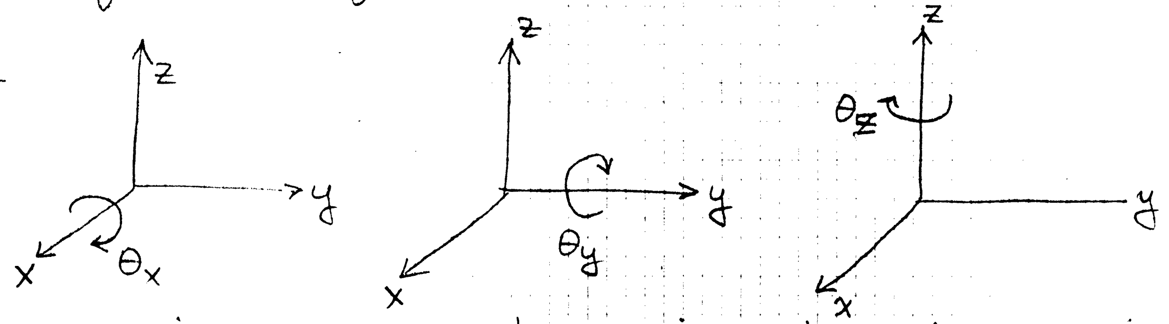
if $c = 1$ these are normalized

in general, homogeneous coordinates are not unique

linear transformations of homogeneous coordinates

Examples:

③ rotation



In our book! use clockwise direction as seen looking into origin along coordinate axis

$$\underset{R_{x,\theta}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & +\sin\theta_x & 0 \\ 0 & -\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{R_{y,\theta}}{\begin{bmatrix} \cos\theta_y & 0 & -\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ +\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}} \quad \underset{R_{z,\theta}}{\begin{bmatrix} \cos\theta_z & +\sin\theta_z & 0 & 0 \\ -\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

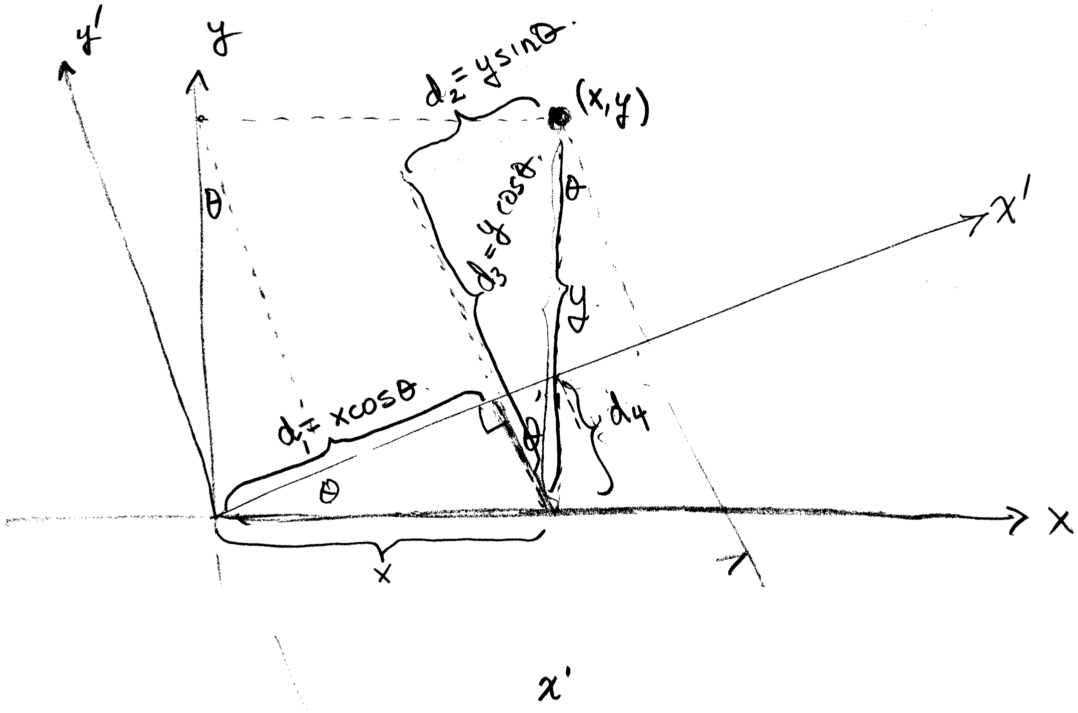
scaling: multiplying coordinate by a constant

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

skewing: representing one coordinate as a linear combination of others.

$$\begin{bmatrix} 1 & k & n & 0 \\ d & 1 & p & 0 \\ e & m & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \text{rotation} = \text{scale} + \text{skew}.$$

④ derivation of rotation



x' calculation:

$\cos\theta = \frac{d_1}{x}$ $\quad\sin\theta = \frac{opp}{hyp}$.

$\therefore d_1 = x\cos\theta$ $\quad\sin\theta = \frac{d_2}{y}$ $\therefore d_2 = y\sin\theta$.

$\therefore x' = x\cos\theta + y\sin\theta$.

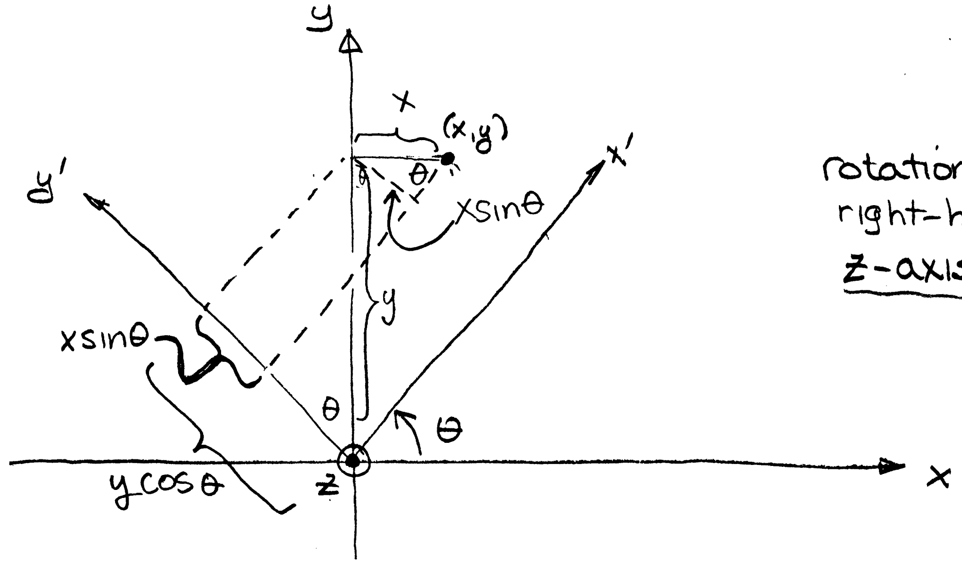
y' calculation

$\cos\theta = \frac{d_3}{y}$ $\therefore d_3 = y\cos\theta$.

$\sin\theta = \frac{d_4}{x}$ $\therefore d_4 = x\sin\theta$.

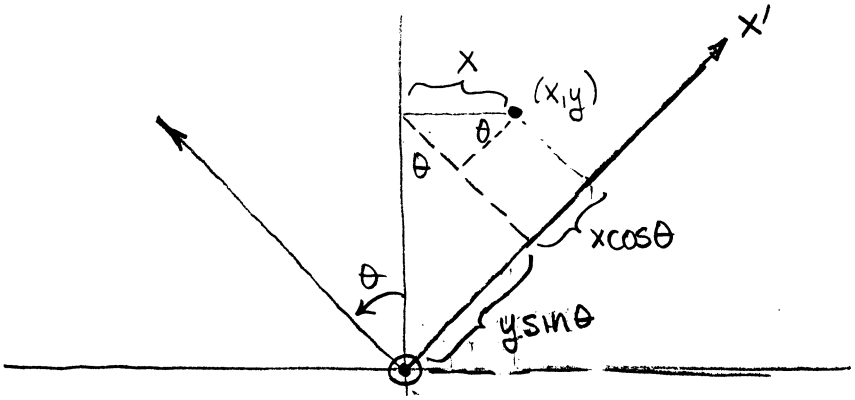
$y' = y\cos\theta - x\sin\theta$.

Derivation of rotational transformations



rotation +θ (given by right-hand rule) about <u>z-axis.</u>

$$y' = y\cos\theta - x\sin\theta$$

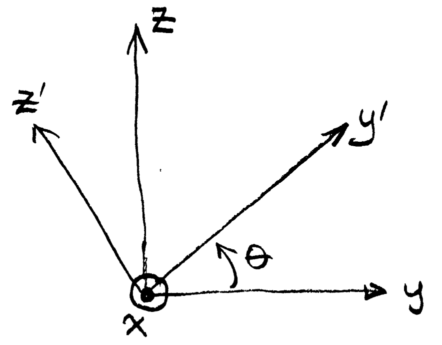


$$x' = y\sin\theta + x\cos\theta$$

$$\begin{bmatrix} \cos\theta & +\sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

as given.

the same derivations hold true for similar rotations

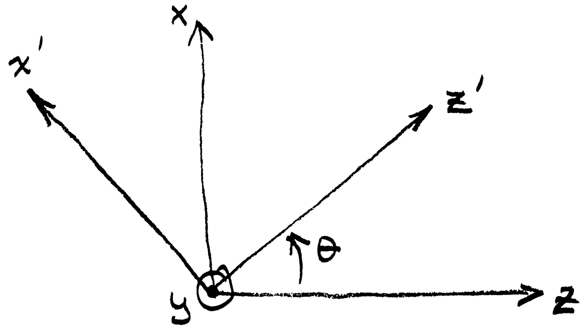


rotation +θ given by right hand rule.

$$y' = z\sin\theta + y\cos\theta$$

$$z' = z\cos\theta - y\sin\theta$$

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



$$z' = x\sin\theta + z\cos\theta$$

$$x' = x\cos\theta - z\sin\theta$$

$$R_{y,\theta} \quad \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ +\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

homogeneous coordinates make it easy to do compound transformations.

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{T_2} \underbrace{\begin{bmatrix} \cos\theta & +\sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{R_{z,\theta}} \underbrace{\begin{bmatrix} 1 & 0 & +x_0 \\ 0 & 1 & +y_0 \\ 0 & 0 & 1 \end{bmatrix}}_{T_1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

T_2	$R_{z,\theta}$	T_1
translate origin back to (x_0, y_0)	rotate angle θ about z-axis <u>at (x_0, y_0)</u>	translate origin to (x_0, y_0)

$$\xleftarrow[\text{sequence of operations}]{\hspace{8em}}$$

$$\begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = T_1 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

$$\begin{bmatrix} a' \\ b' \\ 1 \end{bmatrix} = R_{z,\theta} \begin{bmatrix} a \\ b \\ 1 \end{bmatrix} = R_{z,\theta} T_1 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a'' \\ b'' \\ 1 \end{bmatrix} = T_2 \begin{bmatrix} a' \\ b' \\ 1 \end{bmatrix} = T_2 R_{z,\theta} T_1 \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

homogeneous coordinates make it easy to do compound transformations

$$\begin{bmatrix} a(x,y) \\ b(x,y) \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_0 \\ 0 & 1 & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ +\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

translate origin to (x_0,y_0) rotate θ about (x_0,y_0) translate origin back to original origin

$$\xrightarrow{\hspace{6em}}$$

sequence of operations

Separable implementation (sort of like a separable Fourier transform).

$$a(x,y) = x\cos\theta - y\sin\theta \qquad (13)$$

$$b(x,y) = x\sin\theta + y\cos\theta \qquad (14)$$

from (13) solve for $x = \dfrac{a(x,y) + y\sin\theta}{\cos\theta}$ (17)

substitute into (14)

$$b = +\left(\frac{a + y\sin\theta}{\cos\theta}\right)\sin\theta + y\cos\theta$$

$$= -\frac{a\sin\theta}{\cos\theta} + \frac{y\sin^2\theta}{\cos\theta} + \frac{y\cos^2\theta}{\cos\theta}$$

$$b(x,y) = \frac{+a(x,y)\sin\theta + y\sin^2\theta + y\cos^2\theta}{\cos\theta} \qquad (18)$$

$$= +\frac{a\sin\theta + y}{\cos\theta}$$

(1) do first transform (compresses features in x).

Use (13) $a(x,y) = \dfrac{x\cos\theta - y\sin\theta}{\quad}$ } transform only x which is linear

$b(x,y) = y.$

[———— intermediate image ————]

(2) then do $a(x,y) = x$ (expand features in y)

$$b(x,y) = \frac{-a(x,y)\sin\theta + y}{\cos\theta}$$

might be more efficient

translation

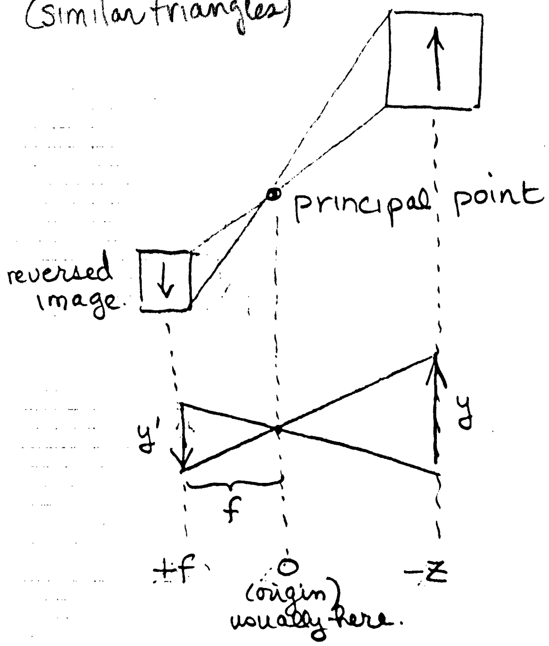
different than robotics (multiply left to right)

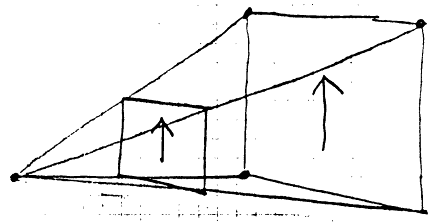
translation matrix :
$$\begin{bmatrix} 1 & 0 & 0 & t \\ 0 & 1 & 0 & u \\ 0 & 0 & 1 & v \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

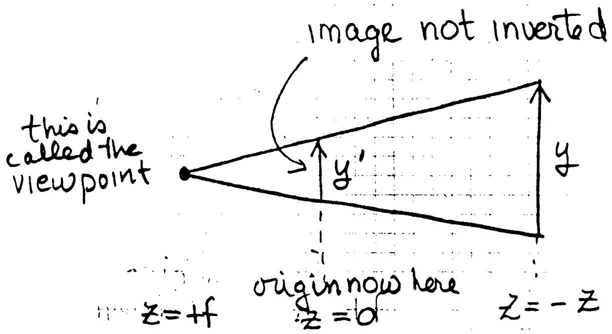
T

$$\begin{bmatrix} 1 & 0 & 0 & t \\ 0 & 1 & 0 & u \\ 0 & 0 & 1 & v \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x+t \\ y+u \\ z+v \\ 1 \end{bmatrix} \text{translation}$$

perspective
(similar triangles)







Perspective transformations in x, y

$$\frac{y'}{f} = \frac{y}{f-z} \qquad \frac{x'}{f} = \frac{x}{f-z}$$

mathematically, we can have a z-perspective transformation, but in practice it cannot occur.

Note as $f \to \infty$ perspective transform → orthographic transform (projection)

$f \to 0$ lots of distortion (not useful)

Ballard & Brown:

use linear transformations to relate object and image coordinate

homogeneous coordinates — we may not want to pivot or rotate about the origin

$(x, y) \longrightarrow (a, b, c)$ where $x = \frac{a}{c}$, $y = \frac{b}{c}$

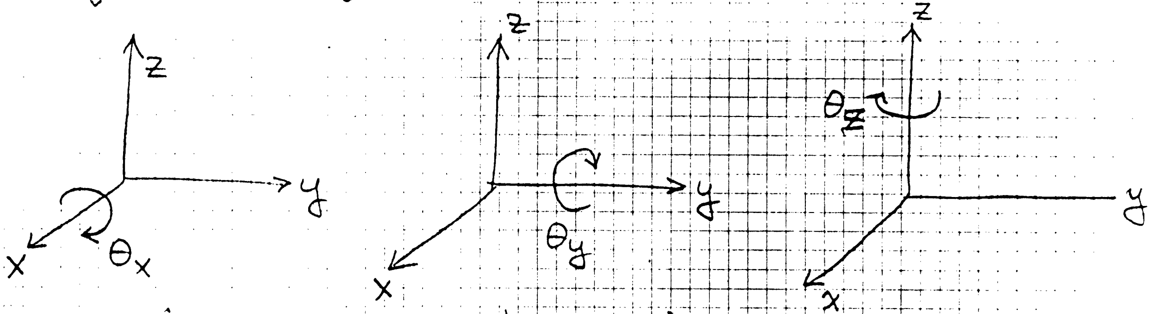
if $c = 1$ these are normalized

in general, homogeneous coordinates are not unique

linear transformations of homogeneous coordinates

Examples:

rotation



use clockwise direction as seen looking into origin along coordinate axis

$R_{x,\theta}$ $R_{y,\theta}$ $R_{z,\theta}$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & +\sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta_y & 0 & +\sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ +\sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

scaling: multiplying coordinate by a constant

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

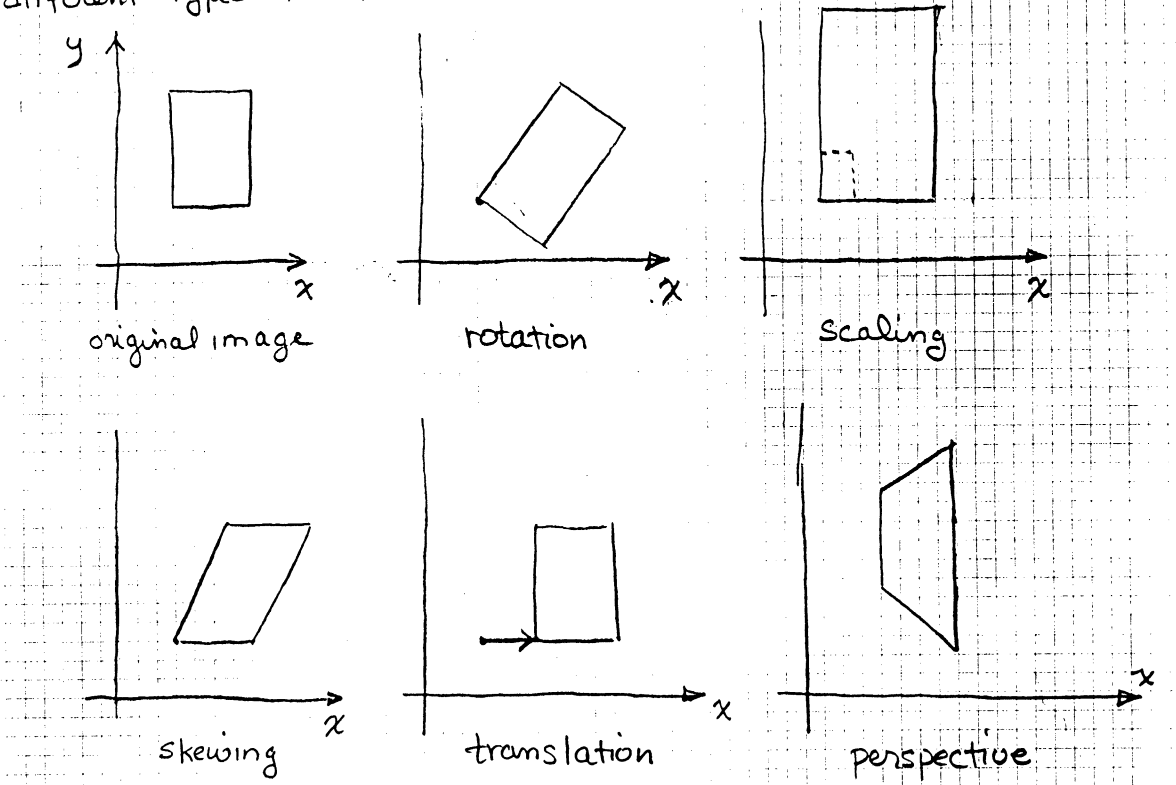
skewing: representing one coordinate as a linear combination of others.

$$\begin{bmatrix} 1 & k & n & 0 \\ d & 1 & p & 0 \\ e & m & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

rotation = scale + skew

References : Ballard & Brown, Ch. 2, p. 17-22 (optics)
p. 467 homogeneous coordinates
p. 477 geometric transformations

different types of effects:



original image

rotation

scaling

skewing

translation

perspective

A perspective transformation is <u>exactly</u> an imaging transformation.

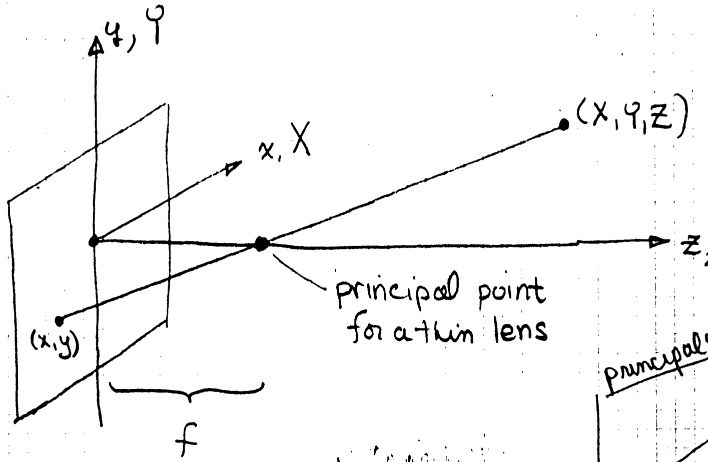
Transform which shows perspective in z-only.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix}$$

Actual coordinate transformation

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ 1-\frac{z}{f} \end{bmatrix}$$

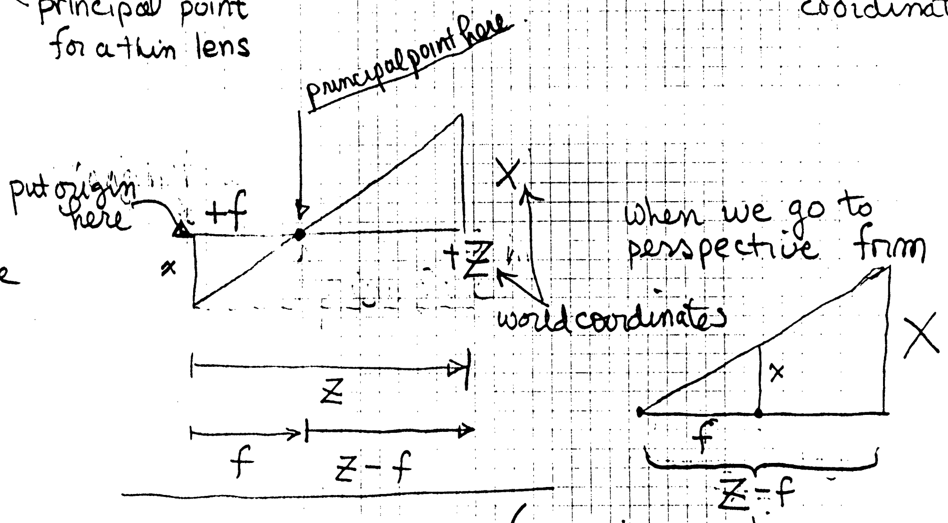
all elements scaled b $\frac{f-z}{f}$. In general, we don't see a z-perspecti transform with a came although we regard it as occurring.



(x, y, z) camera coordinates

(X, Y, Z) world coordinates

Examining figure



$$\therefore \frac{X}{Z-f} = \frac{-x}{f}$$ (x is negative because its inverted)

$$\therefore \frac{x}{f} = -\frac{X}{Z-f} = \frac{X}{f-Z}$$

similarily

$$\frac{y}{f} = -\frac{Y}{Z-f} = \frac{Y}{f-Z}$$

The focal plane coordinates are then

$$x = \frac{f}{f-Z} X$$

$$y = \frac{f}{f-Z} Y$$

This is exactly the perspective transformation, i.e.

$$x' = \frac{x}{1-\frac{z}{f}} = \frac{f}{f-z} x$$

Note: mapping a 3-D scene onto the image plane is a many-to-one transformation.

we actually have the parametric equation of The line given the image plane coordinates (x_o, y_o)

$$x_o = \frac{f}{f-Z} X$$

$$X = \frac{f x_o - x_o Z}{f} = x_o - \frac{x_o}{f} Z$$

$$Y = y_o - \frac{y_o}{f} Z$$

Note that given any (x_o, y_o) There are infinitely many points on the line given by $(X(Z), Y(Z), Z)$. This means that you CANNOT recover 3-D point information by means of the inverse perspective transform unless you know at least one of the world coordinates of the point.

general form of transformation matrix

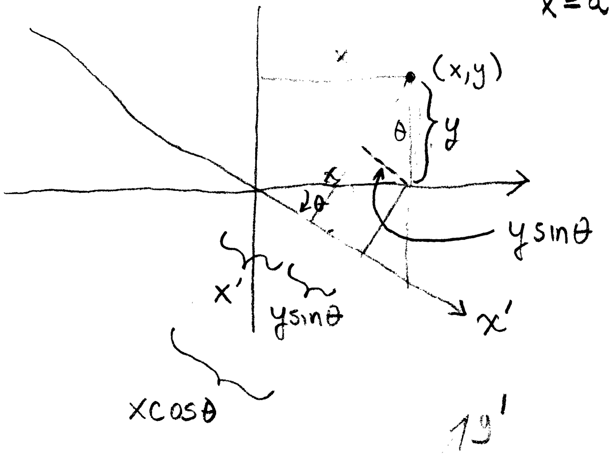
$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \left[\begin{array}{cc|c} \text{scale x} & \text{rot} & \overset{\text{translation}}{x_t} \\ \text{rot} & \text{scale y} & y_t \\ \hline \multicolumn{2}{c}{\text{~~not used~~}} & \text{zoom} \end{array}\right] \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

expansion to 3-D

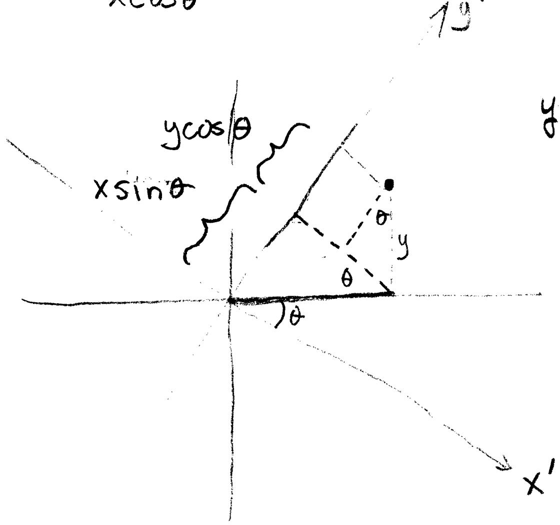
$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \left[\begin{array}{ccc|c} x_{scale} & \text{rot.} & & x_t \\ & y_{scale} & & y_t \\ \text{rot} & & z_{scale} & z_t \\ \hline \multicolumn{3}{c}{\text{perspective}} & \text{zoom} \end{array}\right]$$

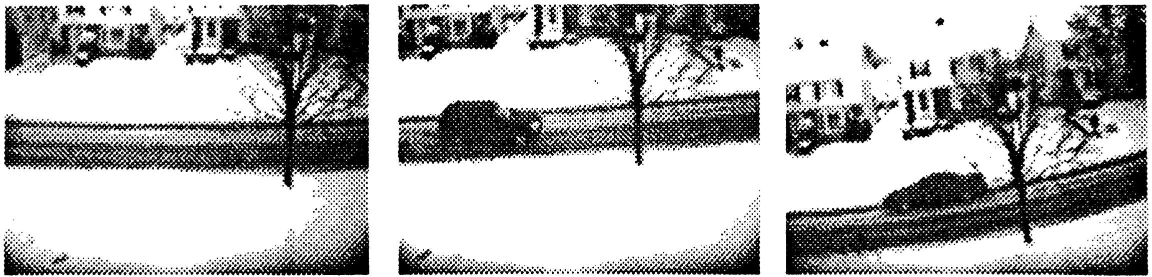
simple expansion of plane rotation

$x' = a(x,y) = x\cos\theta - y\sin\theta.$

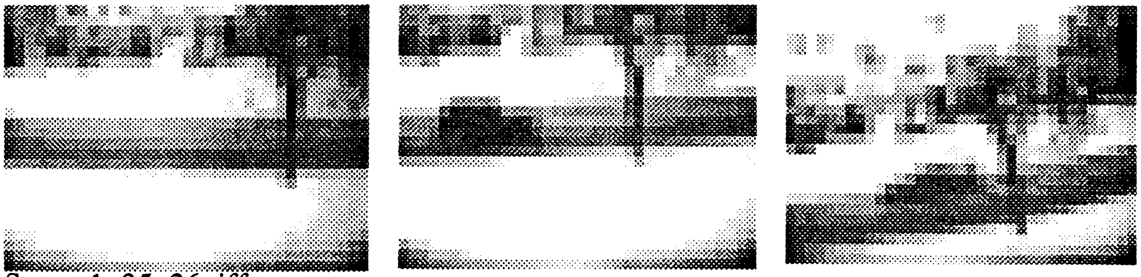


$y' = b(x,y) = x\sin\theta + y\cos\theta$

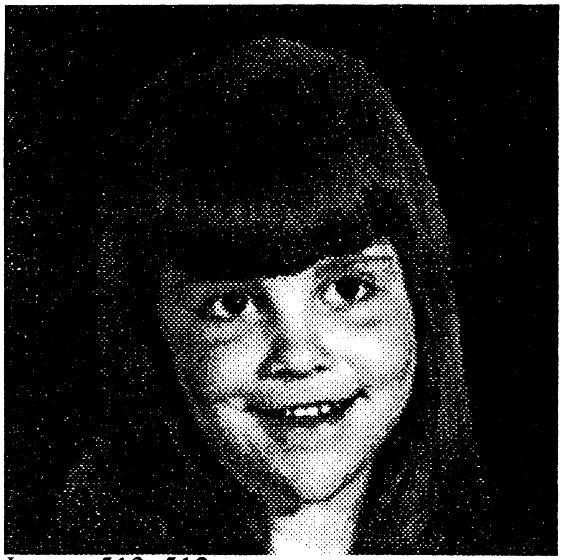


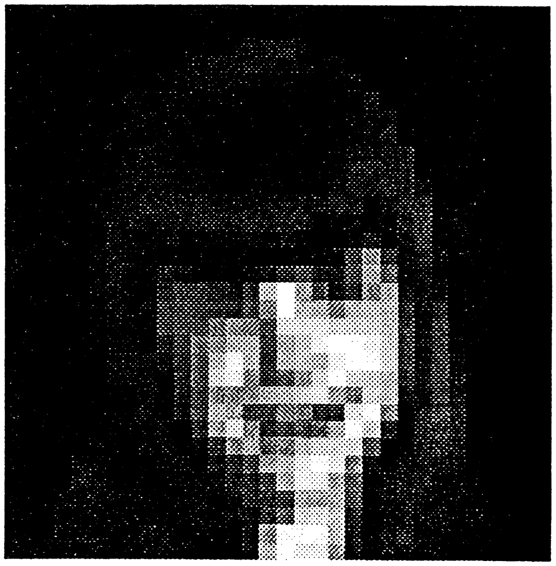


street-1.tiff

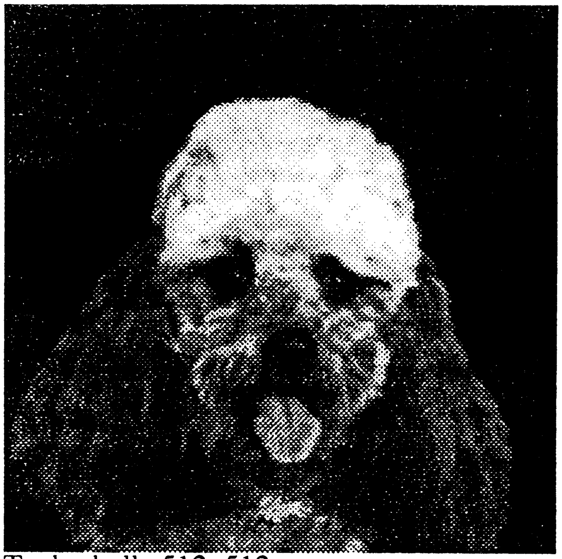


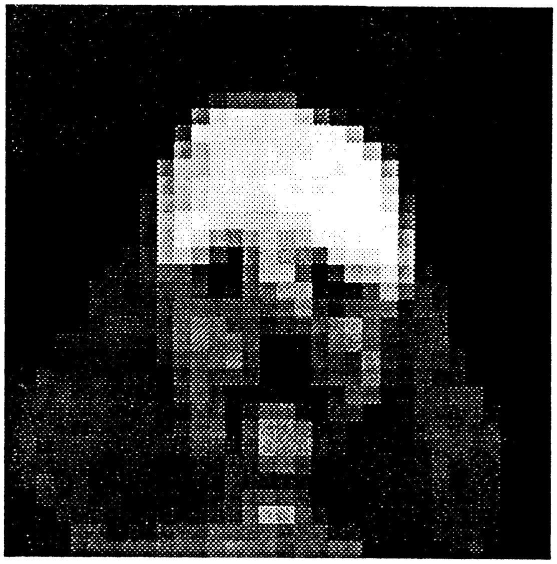
Street-1_35x26.tiff





Jennry 512x512





Tenkerbelle 512x512

8.3.2 General transformations

show geometric transform of Ranger spacecraft camera.

8.3.3. Specification by control points

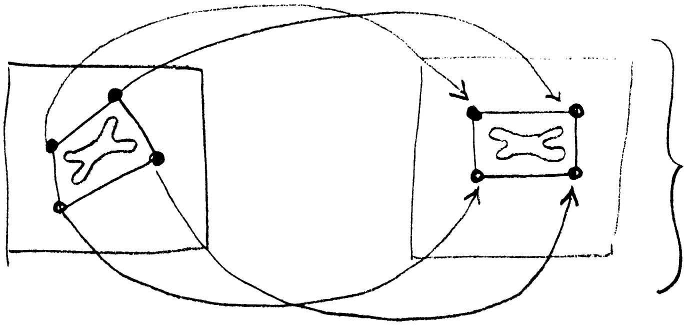
specify the spatial transform as a series of displacement values for selected <u>control points</u> in the image. Displacement of non-control points determined by interpolation

often use polynomials up to 5th degree.

sometimes need more complex transformations

break picture into polygons and use

piecewise bilinear mapping functions



control grid interpolation

8.3.4. Polynomial warping

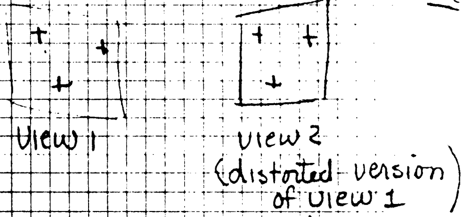
# of control points = # of terms in polynomial — linear equations

# of control points > # of terms in polynomial

use fitting techniques

8.3.5 Control grid interpolation

Geometric correction and registration



perspective correction of imagery :

Given two images of the same scene taken by sensors with different or time-varying orientations, correct one of the images to the viewpoint of the other.

(lots of industrial & tactical applications using geometrically distributed sensors; satellites with varying viewing angles, etc)

polynomial warp model

- mathematical model of the distortion
- a set of corresponding image points of the form $(\underline{x}, \underline{w})$

in image #1 $\underline{x}$ is the vector location of the undistorted image plane points

in image #2 $\underline{w}$ is the vector location of the distorted (or imaged) poin

Now, warp $\underline{w}$ into $\underline{x}$

points in this set are often called control points

useful for multiple sensor images (IR and visible)

image 1	$f_1(\underline{x})$	x_1, x_2 coordinates
image 2	$f_2(\underline{w})$	w_1, w_2 coordinates

$$\left.\begin{aligned} x_1 &= g_1(w_1, w_2) \\ x_2 &= g_2(w_1, w_2) \end{aligned}\right\}$$

i.e. what is the transform from $\underline{w} \rightarrow \underline{x}$ so that we can warp Image 2

↑ approximate by N^{th} order 2-D polynomials

$$x_1 = \sum_{i=0}^{N} \sum_{j=0}^{N} k_{ij}^{(1)} w_1^i w_2^j$$

$$x_2 = \sum_{i=0}^{N} \sum_{j=0}^{N} k_{ij}^{(2)} w_1^i w_2^j$$

Given a set of m corresponding control points in each coordinate system

$$(X_{1i}, X_{2i}, W_{1i}, W_{2i}) \qquad i = 1, 2, 3, \dots, m$$

For $N=2$ (2nd order warp), 2nd order in each variable

eqn (1)

example for $i = k$

image #1 image #2

$$X_{1k} = k^{(1)}_{00} + k^{(1)}_{10} w_{1k} + k^{(1)}_{01} w_{2k} + k^{(1)}_{11} w_{1k} w_{2k} + k^{(1)}_{20} (w_{1k})^2$$

$$+ k^{(1)}_{02} (w_{2k})^2 + k^{(1)}_{21} (w_{1k})^2 w_{2k} + k^{(1)}_{12} w_{1k} (w_{2k})^2$$

w's are known

$$+ k^{(1)}_{22} (w_{1k})^2 (w_{2k})^2$$

up to second order in each variable

X_{2k} looks identical

$$= k^{(2)}_{00} + k^{(2)}_{10} w_{1k} + k^{(2)}_{01} w_{2k} + k^{(2)}_{11} w_{1k} w_{2k} + \dots$$

for $N=2$ 18 coefficients (9 per equation) need to be estimated

$\Rightarrow$ at least 9 control points are needed.

in general for N (order of warp) need $2(N+1)^2$ points

Notes:

① The estimation equations are linear in $k^{(\ )}_{ij}$ (the variables, the w's are known)

② The estimation process is separable since $k^{(1)}_{ij}$ and $k^{(2)}_{ij}$ only appear in one of the equations, i.e. $k^{(1)}_{ij}$ and $k^{(2)}_{ij}$ can be found separately.

③ The $k^{(\ )}_{ij}$ coefficients are the respective powers of w_1 and w_2 which are identical and only need to be computed once.

not a rotation or a translation

Bilinear interpolation

$$G(x,y) = F(x',y') = F(\underline{ax+by}+cxy+d,\ \underline{ex+fy}+gxy+h)$$

transform defined by a through h

if a quadrilateral maps to a quadrilateral specifying vertices gives 2 sets of 4 linear equations in <u>four unknowns</u>.

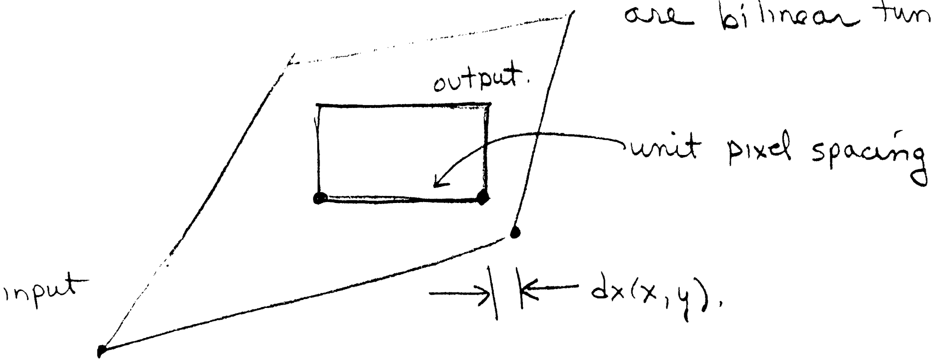
$x_1 \to x_1'$
$y_1 \quad y_1'$
$x_2 \to x_2'$
$y_2 \quad y_2'$
$x_3 \to x_3'$
$y_3 \quad y_3'$
$x_4 \to x_4'$
$y_4 \quad y_4'$

1 set for x
1 set for y. } each set has 4 equations

Better way to define it

$$G(x,y) = F\left[x + dx(x,y),\ y + dy(x,y)\right].$$

pixel displacements that are bilinear functions of x and y.



$dx(x,y)$ & $dy(x,y)$ bilinear in x & y.

$\Rightarrow$ linear in x <u>along</u> each horizontal line in output

for each output line $dx(x+1,y) = dx(x,y) + \Delta x$

where Δx varies for each line

8-4 Applications of Geometric operations

8.4.1. Geometric calibration

remove camera induced geometric distortion from images

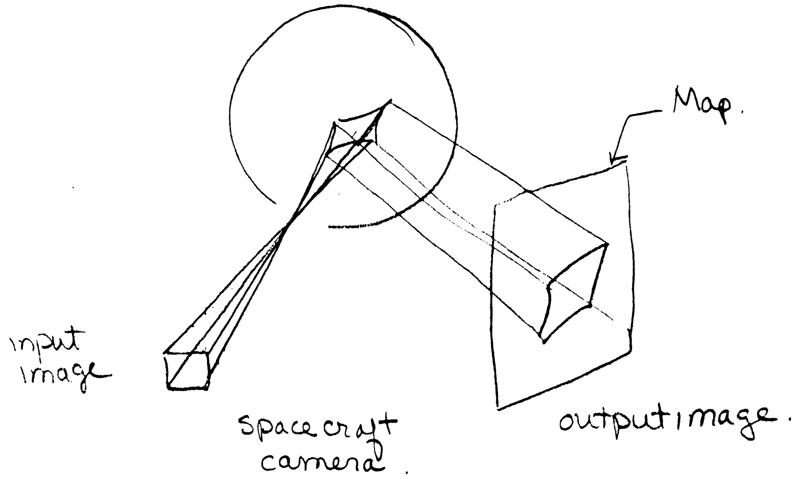
8.4.2. Image Rectification — transform into rectangular pixel coordinates

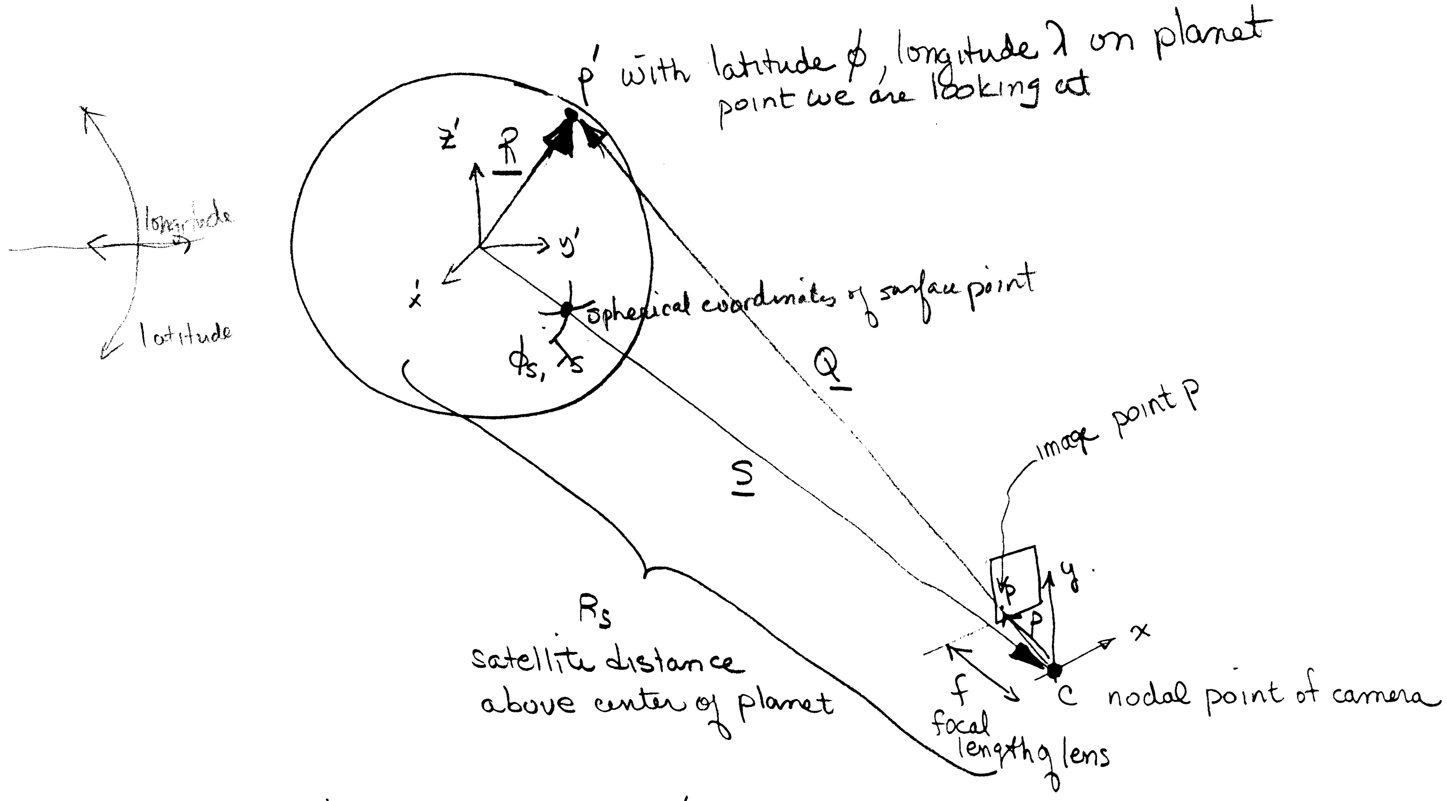
fisheye lens — 5th order polynomial warp. implemented in polar coordinates

8.4.3. Image Registration — match two images

8.4.4. Image Format Conversion

8.4.5 map Projection





satellite above latitude ϕ_s, longitude λ_s

$$\underline{P} = \begin{bmatrix} x_p \\ y_p \\ f \end{bmatrix}$$ camera pixel position coordinates

$$\underline{P} \ \& \ \underline{Q} \text{ are collinear} \Rightarrow \underline{P} = \underbrace{\frac{f}{Q_z}}_{\text{scale factor}} \underline{Q}$$

in vector notation $\vec{\underline{Q}} = \vec{\underline{R}} - \vec{\underline{S}}$

$\underline{R}$ position of object on planet in planet coordinates

$\underline{S}$ position of camera in planet coordinates

$$\underbrace{\begin{bmatrix} Q_x \\ Q_y \\ Q_z \end{bmatrix}}_{\text{satellite coordinates}} = \overset{\text{3x3 transform matrix}}{[m]} \underbrace{\begin{bmatrix} R\cos\phi\cos\lambda - R_s\cos\phi_s\cos\lambda_s \\ R\cos\phi\sin\lambda - R_s\cos\phi_s\sin\lambda_s \\ R\sin\phi - R_s\sin\phi_s \end{bmatrix}}_{\text{planet centered}} \xRightarrow{\text{implies}} \begin{matrix} x_p = \dfrac{f}{Q_z} Q_x \\ y_p = \dfrac{f}{Q_z} Q_y \end{matrix}$$

in camera coordinate frame

$[m]$: planet centered coordinates to spacecraft coordinates

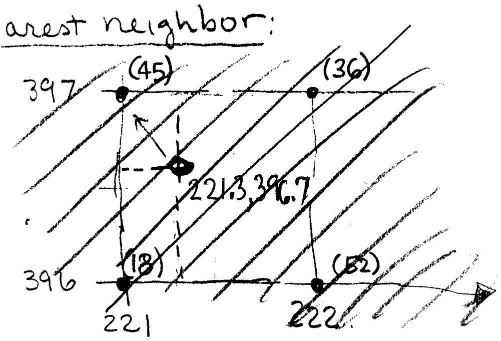
x_p: camera coordinates

$\frac{f}{Q_z} Q_x$: space craft centered

gives planet coordinat

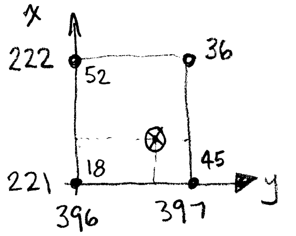
1. Let $F(221, 396) = 18$, $F(221, 397) = 45$, $F(222, 396) = 52$ and $F(222, 397) = 36$. What is $F(221.3, 396.7)$, obtained by nearest neighbor interpolation? By bilinear interpolation? Write the bilinear equation (Eq. 2), showing the values of the coefficients. Draw a graph similar to Figure 8-3.

Nearest neighbor:



Since $F(221, 397) = 45$ and is closest neighbor, then $F(221.3, 396.7) = 45$

−4 points



Bilinear interpolation

Using formula from text,

$$f(x,y) = ax + by + cxy + d$$

$$= [F(222,396) - F(221,396)]x + [F(221,397) - F(221,396)]y$$

$$+ [F(222,397) + F(221,396) - F(221,397) - F(222,396)]xy$$

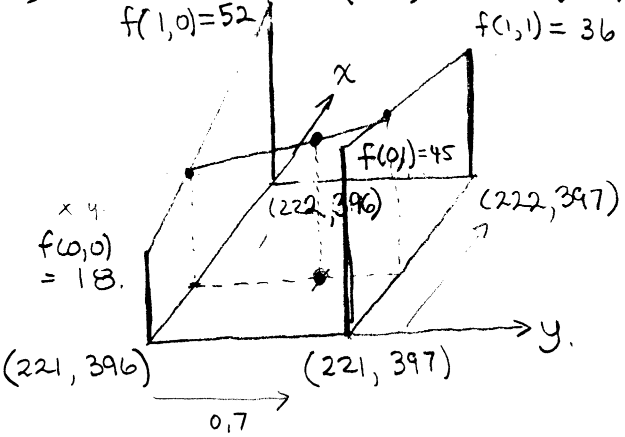
$$+ F(221,396)$$

$$= [52-18]x + [45-18]y + [36+18-45-52]xy + 18$$

$$f(x,y) = 34x + 27y - 43xy + 18$$

$$f(0.3, 0.7) = 34(0.3) + 27(0.7) - 43(.3)(.7) + 18 = 38.07$$

−5 points



−1 no drawing

by inspection $f(x,0) = 18 + x(52-18) = 18 + 34x$

$f(x,1) = 45 + x(36-45) = 45 - 9x$

$$f(\overset{0.3}{\cancel{x}}, y) = f(0.3,0) + y[f(0.3,1) - f(0.3,0)]$$

$$f(0.3,0.7) = f(0.3,0) + 0.7[f(0.3,1) - f(0.3,0)]$$

$$= 18 + 34(0.3) + 0.7[(45 - 2.7) - (18 + 10.2)] = 18 + 10.2 + 0.7[14.1] \approx 38.07$$

$34 \times .3 = 10.2$

$45 - 18 = 27.0$, $27.0 - 10.2 = 16.8$, $16.8 - 2.7 = 14.1$

5(2) We can find the translation and scaling.

If there is no rotation, we can find the scale from the length of the diagonal.

for first image $\quad \text{diag}_{\text{first}} = \sqrt{(504-84)^2 + (433-103)^2} = \sqrt{420^2 + 330^2} = \sqrt{285300}$

$= 534.13$

for new image. $\quad \text{diag}_{\text{second}} = \sqrt{(377-107)^2 + (439-94)^2} = \sqrt{270^2 + 345^2} = \sqrt{191,925}$

$= 438.09$

$$\therefore \text{ scaling} = \frac{438.09}{534.13} = 0.82.$$

(or 1.22 depending upon which way you do ratio)

Now, how to translate. Use $(x,y) = (103, 84)$ and $(x', y') = (107, 94)$.

$$\begin{bmatrix} \underset{107}{x'} \\ \underset{94}{y'} \\ s' \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & .82 \end{bmatrix} \begin{bmatrix} \underset{103}{x} \\ \underset{84}{y} \\ 1 \end{bmatrix}$$

overall scaling

$$\Rightarrow x_t = \frac{x'}{.82} - x = \frac{107}{.82} - 103 = 27.49$$

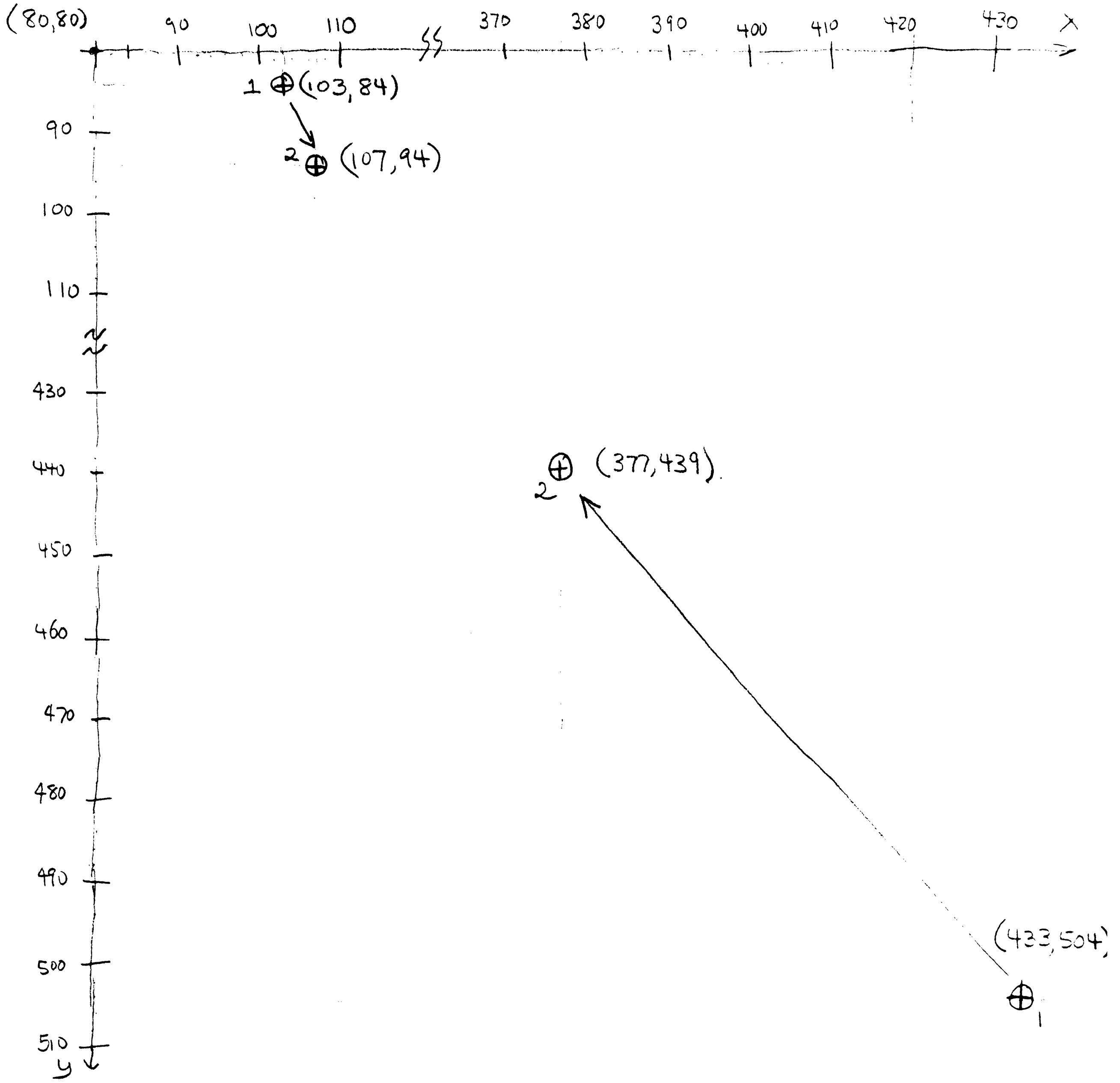
$$y_t = \frac{y'}{.82} - y = \frac{94}{.82} - 84 = 30.63$$

Now, let's do the problem rigorously

Grading: 3.5 correct rotation
3.5 points scale
3 points translation
−1.5 if didn't use
homogeneous
coordinates or
simple translation

5. Suppose you have two digitized images of a section of a city taken from the top of a tall building 25 years apart and you wish to display changes by projecting an overlay of the two images. You find a corner of a building that is located at 103, 84 in the first image and at 107, 94 in the second image, and a window that is located at 433, 504 in the first image and at 377, 439 in the second image. Has there been any (a) translation? (b) rotation? (c) change in scale? How much?

Write the geometric transformation required to register the second image with the first. Assume that there has been no geometric distortion beyond translation, rotation, and change in scale.



There are many ways to solve problem.

Rotation is toughest. Look at diagonal of each image.

$$m_{first} = \frac{\Delta y}{\Delta x} = \frac{504 - 84}{433 - 103} = \frac{420}{330} = 1.27 \qquad m_{second} = \frac{439 - 94}{377 - 107} = \frac{345}{270} = 1.28$$

Since the slopes of the diagonals are approx. equal $\Rightarrow$ no rotation.

since we have only 4 points we must restrict the number of variables. Furthermore, we are restricted to rotation, scale and translation.

write
$$\begin{bmatrix} wx'' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & x_t \\ -\sin\theta & \cos\theta & y_t \\ 0 & 0 & s \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

where θ = tangle of rotation

(xt, yt) = translation

s = scale

$$(x_0, y_0)_1 = (103, 84) \longrightarrow (x', y')_1 = (107, 94)$$
$$(x_0, y_0)_2 = (433, 504) \longrightarrow (x', y')_2 = (377, 439)$$

Mult plyout Write equations (1) and (2) for point pair ①

$$w107 = 103\cos\theta + 84\sin\theta + xt \qquad (1)$$
$$w94 = -103\sin\theta + 84\cos\theta + yt \qquad (2)$$
$$w = s$$

for pointpair ②

$$w377 = 433\cos\theta + 504\sin\theta + xt \qquad (3)$$
$$w439 = -433\sin\theta + 504\cos\theta + yt \qquad (4)$$
$$w = s.$$

Combining

(1) −(3) $$w(107-377) = (103-433)\cos\theta + (84-504)\sin\theta \qquad (5)$$

(2) −(4) $$w(94-439) = (-103+433)\sin\theta + (84-504)\cos\theta \qquad (6)$$

Simplifying

$$-270w = -330\cos\theta - 420\sin\theta$$
$$-345w = +330\sin\theta - 420\cos\theta$$

Now eliminate $\underline{\underline{w}}$ from these equations

$$\frac{-330\cos\theta - 420\sin\theta}{-270} = \frac{330\sin\theta - 420\cos\theta}{-345}$$

this gives θ

$$1.22\cos\theta + 1.55\sin\theta = -0.956\sin\theta + 1.217\cos\theta$$

$$\left(\frac{330}{270} - \frac{420}{345}\right)\frac{\cancel{\cos\theta}}{\cancel{\cos\theta}} = \left(-\frac{330}{345} - \frac{420}{270}\right)\frac{\sin\theta}{\cos\theta}$$

$$\tan\theta = \frac{\frac{330}{270} - \frac{420}{345}}{-\frac{330}{345} - \frac{420}{270}} = -1.92 \times 10^{-3}$$

$\therefore\ \theta \approx -0.1°$, a negligible quantity to first order in (1), (2), (3), (4) using the figure for θ

$$107w = 103 - .162 + x_t$$
$$94w = .198 + 84 + y_t$$
$$377w = 433 - .968 + x_t$$
$$439w = .831 + 504 + y_t$$

making some approximations

$$107w = 103 + x_t$$
$$94w = 84.2 + y_t$$
$$377w = 432 + x_t$$
$$439w = 504.8 + x_t$$

$$107w - 377w = 103 - 432$$
$$w = \frac{103 - 432^{old}}{107 - 377_{new.}} = 1.22$$

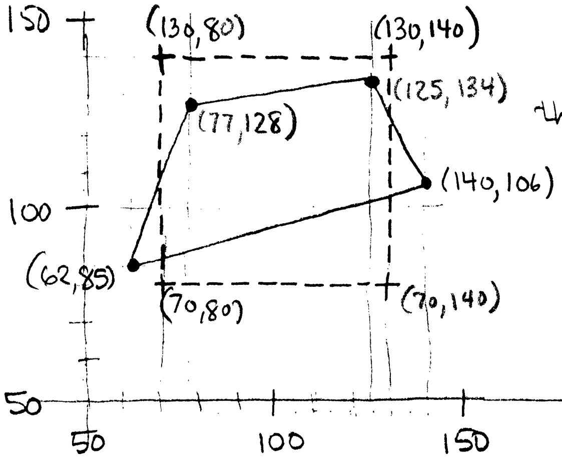
$$x_t = 107(1.22) - 103 = 27.5$$
$$y_t = 94(1.22) - 84.2 = 30.5$$

The formal transformation from (x,y) to (x',y') is then

$$\begin{bmatrix} 1.22x' \\ 1.22y' \\ 1.22 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 27.5 \\ 0 & 1 & 30.5 \\ 0 & 0 & 1.22 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}.$$

which is NOT too different.

7. Suppose you have a digitized photograph of the ground taken at an angle from behind the window of an airplane. You want to rectify the image so that it appears as if you are looking straight down. A square cotton field has corners at pixel coordinates (62, 85), (77, 128), (125, 134), and (140, 106). Derive the geometric transformation that will rectify the image. Plot the cotton field in the image before and after rectification.



this is the original image. made up

(62,85)	$\longrightarrow$	(70, 80)
(77,128)	$\longrightarrow$	(70, 140)
(125,134)	$\longrightarrow$	(130, 140)
(140,106).	$\longrightarrow$	(130, 80)

To rectify you MUST pick the new points. There are many. One possible set is shown above. We have 4 points which map to 4 new points.

For the bilinear transform $g(x',y') = g[f(x,y), g(x,y)] = p(x',y')$.

Let $f(x,y) = ax + by + cxy + d$

Then
$$f(62,85) = 70$$
$$f(77,128) = 70$$
$$f(125,134) = 130$$
$$f(140,106) = 130$$

Setting up the equations

$$a62 + b85 + c62*85 + d = 70$$
$$a77 + b128 + c77*128 + d = 70$$
$$a125 + b134 + c125*134 + d = 130$$
$$a140 + b106 + c140*106 + d = 130$$

in matrix form
$$\begin{bmatrix} 62 & 85 & 5270 & 1 \\ 77 & 128 & 9856 & 1 \\ 125 & 134 & 16750 & 1 \\ 140 & 106 & 14840 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} 70 \\ 70 \\ 130 \\ 130 \end{bmatrix}$$

I wrote a MATLAB program to solve for a, b, c, d.

The resulting values were

$a = -0.6121$
$b = -1.2889$
$c = 0.0141$
$d = 143.2663$

I then set up the equations for y and solved for a second bilinear transformation using my MATLAB program.

The second set of equations was very similar

$$\begin{bmatrix} 62 & 85 & 5270 & 1 \\ 77 & 128 & 9856 & 1 \\ 125 & 134 & 16750 & 1 \\ 140 & 106 & 14840 & 1 \end{bmatrix} \begin{bmatrix} a' \\ b' \\ c' \\ d' \end{bmatrix} = \begin{bmatrix} 80 \\ 140 \\ 140 \\ 80 \end{bmatrix}.$$

These coefficients were determined to be

$a' = -1.1554$
$b' = 1.0366$
$c' = .0071$
$d' = 25.8815$

```
% new program which will compute bilinear transformation given point pairs
clc,clg;
clear

%do the x coordinate
P_1=[62,85];
P_2=[77,128];
P_3=[125,134];
P_4=[140,106];
A=[P_1(1) P_1(2) P_1(1)*P_1(2) 1; ...
   P_2(1) P_2(2) P_2(1)*P_2(2) 1; ...
   P_3(1) P_3(2) P_3(1)*P_3(2) 1; ...
   P_4(1) P_4(2) P_4(1)*P_4(2) 1];
B_1=[70; 70; 130; 130];
COEFF_1=inv(A)*B_1
X_PRIME=COEFF_1(1)*P_1(1)+COEFF_1(2)*P_1(2)+COEFF_1(3)*P_1(1)*P_1(2)+COEFF_1(4)
X_PRIME=COEFF_1(1)*P_2(1)+COEFF_1(2)*P_2(2)+COEFF_1(3)*P_2(1)*P_2(2)+COEFF_1(4)
X_PRIME=COEFF_1(1)*P_3(1)+COEFF_1(2)*P_3(2)+COEFF_1(3)*P_3(1)*P_3(2)+COEFF_1(4)
X_PRIME=COEFF_1(1)*P_4(1)+COEFF_1(2)*P_4(2)+COEFF_1(3)*P_4(1)*P_4(2)+COEFF_1(4)

%do the y coordinate
A=[P_1(1) P_1(2) P_1(1)*P_1(2) 1; ...
   P_2(1) P_2(2) P_2(1)*P_2(2) 1; ...
   P_3(1) P_3(2) P_3(1)*P_3(2) 1; ...
   P_4(1) P_4(2) P_4(1)*P_4(2) 1];
B_2=[80; 140; 140; 80];
COEFF_1=inv(A)*B_2
Y_PRIME=COEFF_1(1)*P_1(1)+COEFF_1(2)*P_1(2)+COEFF_1(3)*P_1(1)*P_1(2)+COEFF_1(4)
Y_PRIME=COEFF_1(1)*P_2(1)+COEFF_1(2)*P_2(2)+COEFF_1(3)*P_2(1)*P_2(2)+COEFF_1(4)
Y_PRIME=COEFF_1(1)*P_3(1)+COEFF_1(2)*P_3(2)+COEFF_1(3)*P_3(1)*P_3(2)+COEFF_1(4)
Y_PRIME=COEFF_1(1)*P_4(1)+COEFF_1(2)*P_4(2)+COEFF_1(3)*P_4(1)*P_4(2)+COEFF_1(4)
```

```
Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe



COEFF_1 =

   -0.6121
   -1.2889
    0.0141
  143.2663


X_PRIME =

   70.0000


X_PRIME =

   70.0000


X_PRIME =

  130.0000


X_PRIME =

  130.0000


COEFF_1 =

   -1.1554
    1.0366
    0.0071
   25.8815
```

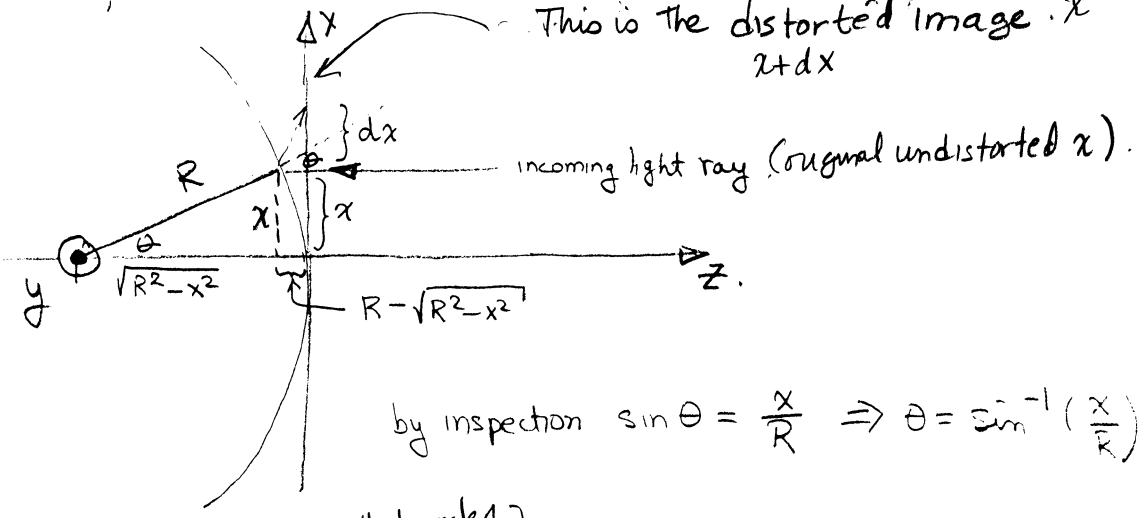
```
Y_PRIME =

    80.0000


Y_PRIME =

   140.0000


Y_PRIME =

   140.0000


Y_PRIME =

    80.0000

»
```

/3

9. The police have a photograph taken during the commission of a crime. Unfortunately, The robbery took place behind the tourist with the camera. You notice in the photo a large chrome-plated sphere that appears to be acting like a mirror. You use a film digitizer with 25-μ pixel spacing, and find That the image of the ball measures 360 pixels in diameter. The actual ball is 3 feet in, and it was 27 feet from the camera position. Develop an equation for the geometric transformation that will rectify The image of the robbers. You may assume that the radius of the ball is negligible compared to its distance from the camera and from the crime.

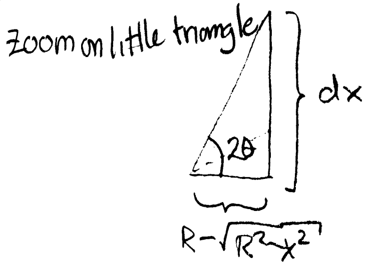
The whole issue is how to map the light rays from the object plane to the image plane.

In one dimension, this can be written as



Δx
This is the distorted image. x'
$x+dx$
$\}dx$
incoming light ray (original undistorted x).
R
x
$\}x$
θ
$\sqrt{R^2-x^2}$
y
z.
$R-\sqrt{R^2-x^2}$

by inspection $\sin\theta = \frac{x}{R} \Rightarrow \theta = \sin^{-1}\left(\frac{x}{R}\right)$



$$\tan 2\theta = \frac{dx}{R-\sqrt{R^2-x^2}}$$

∴ procedure for $x = -5, \dots +5$

compute $\theta_x = \sin^{-1}\left(\frac{x}{R}\right)$

$dx = \left(R-\sqrt{R^2-x^2}\right)\tan 2\theta_x$

$x' = x + dx$

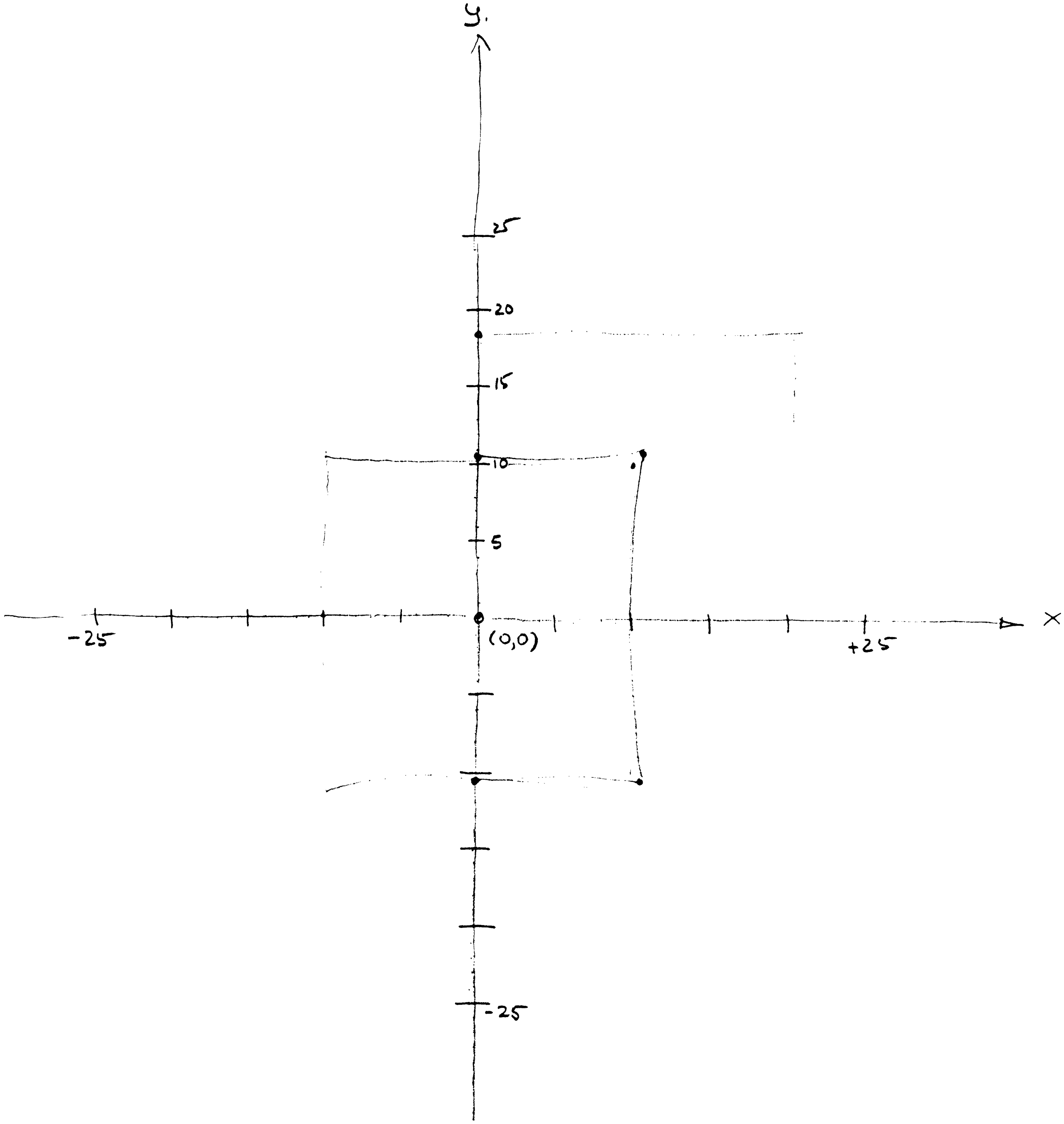
repeat for $y = -5, \dots, 5$

$\theta_y = \sin^{-1}\left(\frac{y}{R}\right)$

$dy = \left(R-\sqrt{R^2-y^2}\right)\tan 2\theta_y$

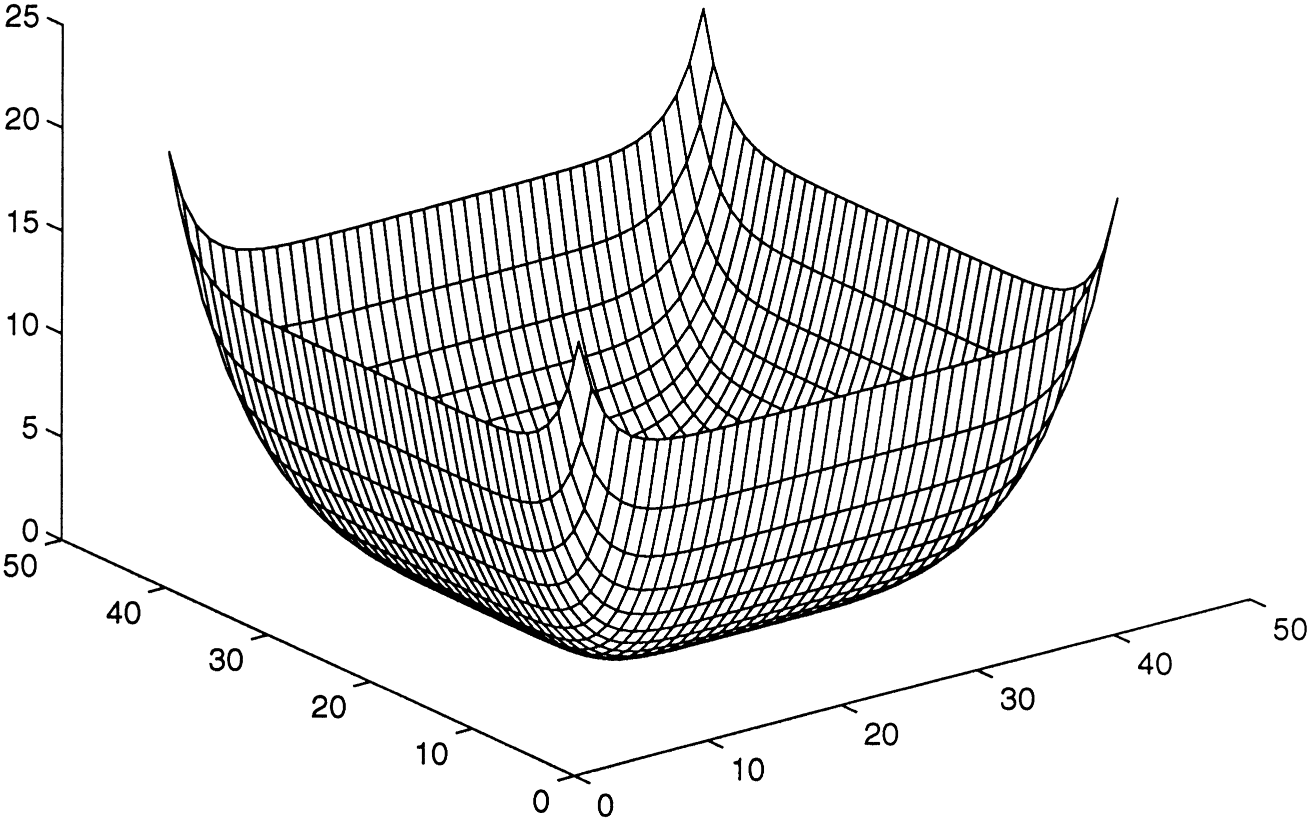
$y' = y + dy.$

Now create a table and use warp function!

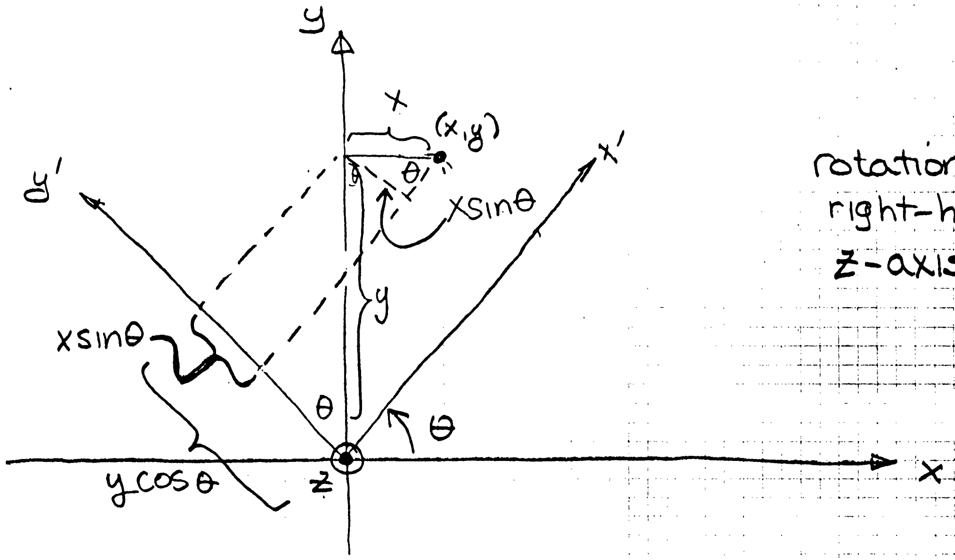

y.
25
20
15
10
5
x
-25
(0,0)
+25
-25

```
% geometrix spherical mirror transformation
clc,clg;
clear

%radius of sphere, inches
R=36

%do the x coordinate
x=-20:1:20;
%do the y coordinate
y=-20:1:20;

thetax=asin(x/R);
dx=(R-sqrt(R*R-x.*x)).*tan(thetax*2);
xprime=x+dx;
thetay=asin(y/R);
dy=(R-sqrt(R*R-y.*y)).*tan(thetay*2);
yprime=y+dy;

for m=1:41
   for n=1:41
   z(m,n) = sqrt(dx(m)*dx(m)+dy(n)*dy(n));
   end
end
mesh(z)
```

```
Commands to get started: intro, demo, help help
Commands for more information: help, whatsnew, info, subscribe


R =

   36


xprime =

  Columns 1 through 7

  -34.6449  -29.9763  -26.3538  -23.4116  -20.9374  -18.7993  -16.9110
     20        19        18        17        16        15        14
  Columns 8 through 14

  -15.2132  -13.6638  -12.2319  -10.8941   -9.6325   -8.4328   -7.2836
     13        12        11        10         9         8         7
  Columns 15 through 21

   -6.1752   -5.0998   -4.0505   -3.0211   -2.0062   -1.0008         0
     6         5         4         2         -         1
  Columns 22 through 28

    1.0008    2.0062    3.0211    4.0505    5.0998    6.1752    7.2836

  Columns 29 through 35

    8.4328    9.6325   10.8941   12.2319   13.6638   15.2132   16.9110

  Columns 36 through 41

   18.7993   20.9374   23.4116   26.3538   29.9763   34.6449


yprime =

  Columns 1 through 7
  -34.6449  -29.9763  -26.3538  -23.4116  -20.9374  -18.7993  -16.9110

  Columns 8 through 14

  -15.2132  -13.6638  -12.2319  -10.8941   -9.6325   -8.4328   -7.2836

  Columns 15 through 21

   -6.1752   -5.0998   -4.0505   -3.0211   -2.0062   -1.0008         0

  Columns 22 through 28

    1.0008    2.0062    3.0211    4.0505    5.0998    6.1752    7.2836

  Columns 29 through 35

    8.4328    9.6325   10.8941   12.2319   13.6638   15.2132   16.9110

  Columns 36 through 41

   18.7993   20.9374   23.4116   26.3538   29.9763   34.6449

»
```

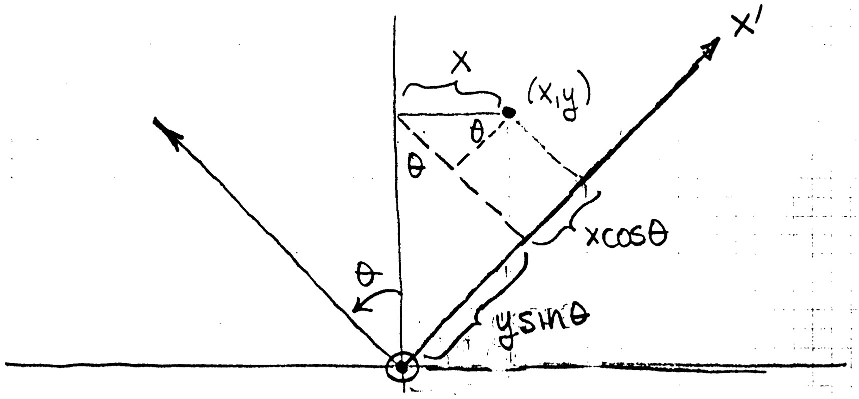

25
20
15
10
5
0
50
40
30
20
10
0
0
10
20
30
40
50

Derivation of rotational transformations



rotation +θ (given by right-hand rule) about z-axis.

$$y' = y\cos\theta - x\sin\theta$$



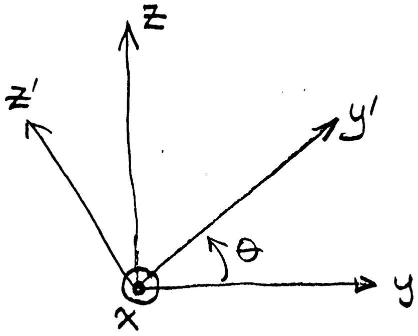
$$x' = y\sin\theta + x\cos\theta$$

This is pre-multiplication

$$[x \;\; y \;\; z \;\; 1]\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

as given.

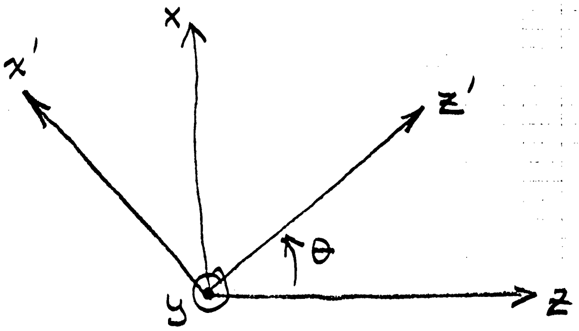
the same derivations hold true for similar rotations



rotation +θ given by right hand rule.

$$y' = z\sin\theta + y\cos\theta$$
$$z' = z\cos\theta - y\sin\theta$$

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



$$z' = x\sin\theta + z\cos\theta$$
$$x' = x\cos\theta - z\sin\theta$$

$$\begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

2

Use pseudo inverse as a system of, for example, 9 points

1

Notes on curve fitting in MATLAB:

(1) Write as a matrix equation using the coefficients as variables

i.e. if equations are.

$$a_{111}x_0 + a_{112}y_0 + a_{113}z_0 + a_{114} - a_{131}x_{i1}x_0 - a_{132}x_{i1}y_0 - a_{133}x_{i1}z_0 - x_{i1} = 0$$

$$a_{121}x_0 + a_{122}y_0 + a_{123}z_0 + a_{124} - a_{131}y_{i}x_0 - a_{132}y_{i1}y_0 - a_{133}y_{i1}z_0 - y_{i1} = 0.$$

Where (x_0, y_0, z_0) and (x_i, y_i) are the known calibration points, i.e. data.

Then rewrite as

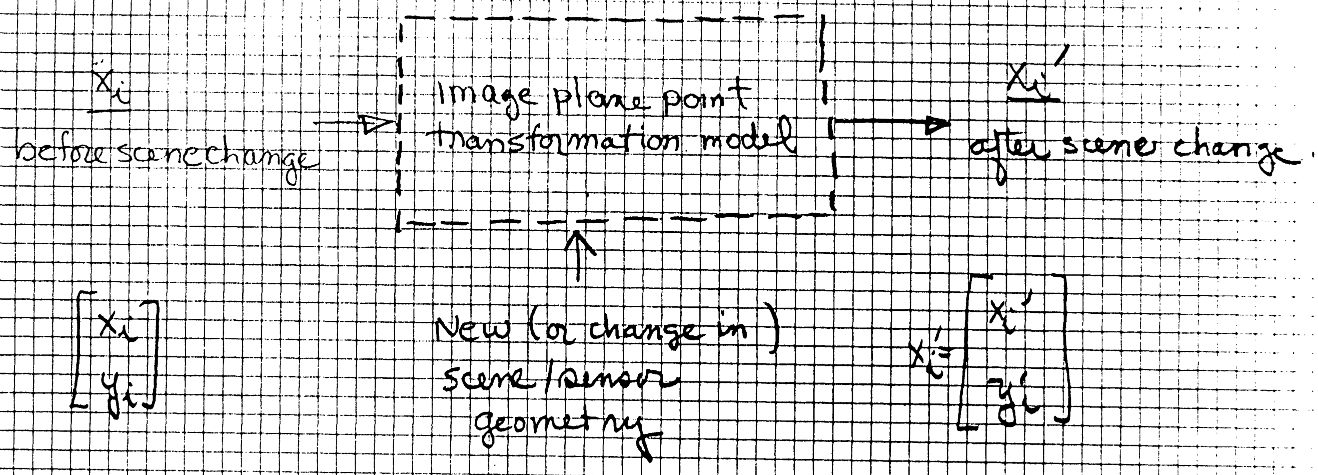
$$\begin{bmatrix} x_0 & y_0 & z_0 & 1 & 0 & 0 & 0 & 0 & -x_{i1}x_0 & -x_{i1}y_0 & -x_{i1}z_0 \\ 0 & 0 & 0 & 0 & x_0 & y_0 & z_0 & 1 & -y_{i1}x_0 & -y_{i1}y_0 & -y_{i1}z_0 \end{bmatrix} \begin{bmatrix} a_{111} \\ a_{112} \\ a_{113} \\ a_{114} \\ a_{121} \\ a_{122} \\ a_{123} \\ a_{124} \\ a_{131} \\ a_{132} \\ a_{133} \end{bmatrix} = \begin{bmatrix} x_{i1} \\ y_{i1} \end{bmatrix}.$$

These are 2 equations in 11 unknowns. We will need at least 6 pairs of known points to solve this equation.

Problem is that system is over determined.

$$\underline{\underline{U}}a = X$$

image plane – image plane (geometric) transformations



$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \qquad \underline{x}_i' = \begin{bmatrix} x_i' \\ y_i' \end{bmatrix}$$

$$\underset{\substack{\uparrow \\ \text{geometrically} \\ \text{perturbed image}}}{f'(\underline{x}_i)} = f(\underline{x}_i') = f\left[\underbrace{g(\underline{x}_i, \underline{a})}_{\substack{\text{planar} \\ \text{transform}}}\right]$$

models changes in intensity only through mapping of cordinate intensities

Application #1:

modeling image plane – image plane transformations due to a change in viewing conditions

Application #2:

modeling the movement or "motion" of 3-D object points relating their positions in the image plane (using the p-p transform) before and after the motion.

model procedure

1. compute using the p-p transform The mapping of object points in the image plane.

2. Compute the same, for the corresponding object point locations That results from the change.

3. Relate the results from these two computations.

p-p transform

$$[w x_i \;\; w y_i \;\; w] = [\, x_0 \;\; y_0 \;\; z_0 \;\; 1 \,]\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -\frac{1}{f}\\ 0 & 0 & 1\end{bmatrix}$$

ignoring z

consider Application #1, a change in focal length f,

Example: autofocus inspection camera which "autonomously" adjusts field of view to fit size of part.

$$[w' x_i' \;\; w' y_i' \;\; w'] = [x_0 \;\; y_0 \;\; z_0 \;\; 1]\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & -\frac{1}{f'}\\ 0 & 0 & 1\end{bmatrix}$$

since z_0 does not change.

$$w = -\frac{z_0}{f} + 1 \qquad w' = -\frac{z_0}{f'} + 1$$

$$wf - f = -z_0 \qquad w'f' - f' = -z_0$$

$$\therefore \quad f(w-1) \approx f'(w'-1)$$

for large magnifications and reasonably small focal length changes $|w| >> 1$ giving

$$fw \approx f'w'$$

relationship of image points before and after transformation is

$$\begin{bmatrix} w'x_i' \\ w'y_i' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{f}{f'} \end{bmatrix} \begin{bmatrix} wx_i \\ wy_i \\ w \end{bmatrix}$$

Ok, but how does this relate to a transformation of object coordinates

Suppose $\underline{\hat{X}_0} = [x_0 \; y_0 \; z_0]$ moves along $\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$ to a new position

$$\hat{X}_0' = [x_0 + d_x \quad y_0 + d_y \quad z_0 + d_z]$$

Then $\quad \underline{\hat{X}_0'} = \underline{\underline{T}}\, \underline{\hat{X}_0}$

$$\text{or} \quad [x_0 + dx \quad y_0 + dy \quad z_0 + dz \quad 1] = [x_0 \quad y_0 \quad z_0 \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & d_z & 1 \end{bmatrix}$$

If your model is $\quad \underline{\hat{X}_i} = P\, \underline{\hat{X}_0}$

and $\quad \underline{\hat{X}_i'} = PT\, \underline{\hat{X}_0}$

yields a non-linear relationship between image points

super homogeneous coordinates

$$[w'x_i' \quad w'y_i' \quad w' \quad 1] = [wx_i \quad wy_i \quad w \quad 1] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ d_x & d_y & \frac{d_z}{f} & 1 \end{bmatrix}$$

makes the problem linear.

Notes:

① The only Thing that requires superhomogeneous coordinates is w' and That is accounted for using The term

$$-\frac{d_z}{f}$$

② If $d_x = d_y = 0$ (i.e. only motion along the optical axis) you get exactly the case of a scale change.

③ The result implicitly involves the object - image plane distance through the magnification ratio in w.

Potential application but too long to do: camera pan & tilt

1. order of pan & tilt is important mechanically and mathematicall
2. good example of transformation matrices
3. reasonable engineering approximations
4. framework for example in camera control.
   (Dzialo & Schalkoff, 1986)

Control Considerations in tracking moving objects using time-varying perspective-projective imagery, IEEE Trans. Indust. Electronics

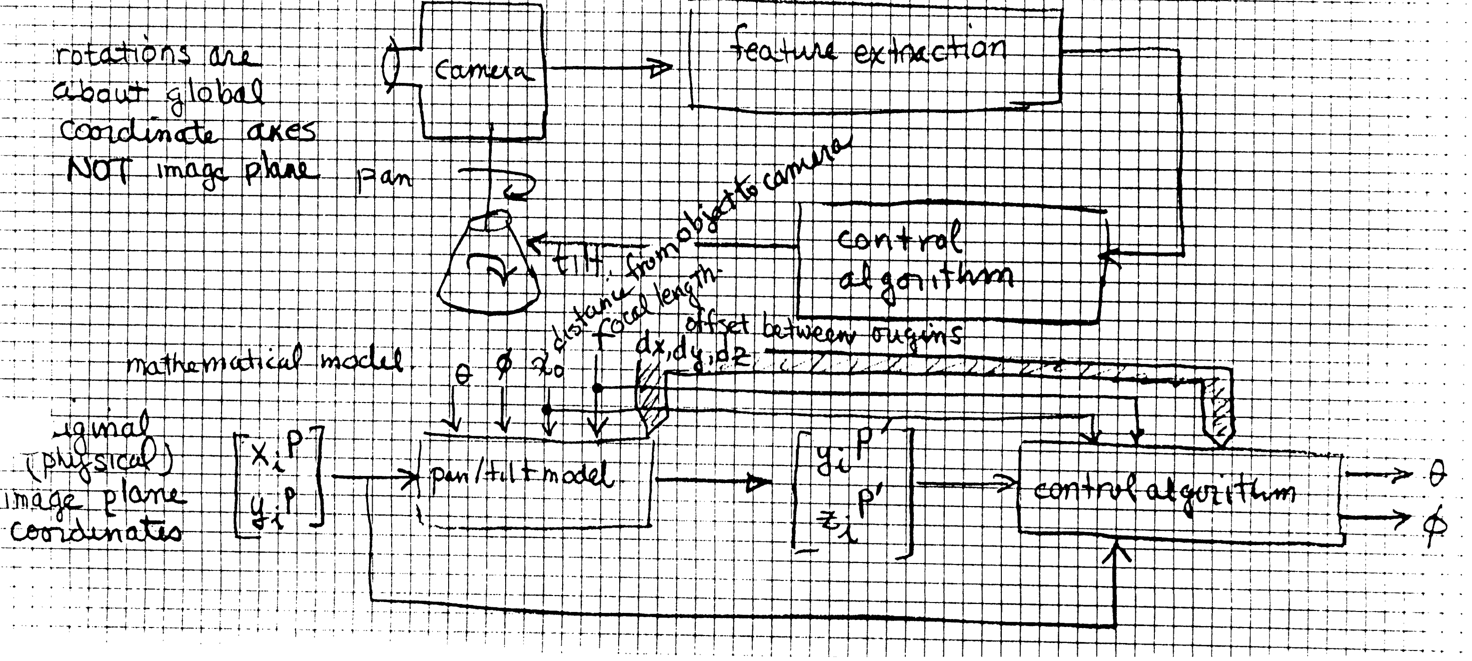

rotations are about global coordinate axes NOT image plane
Camera
feature extraction
pan
tilt
control algorithm
distance from object to camera
focal length
offset between origins
mathematical model
θ
ϕ
z_0
dx, dy, dz
riginal (physical) image plane coordinates
$\begin{bmatrix} x_i^p \\ y_i^p \end{bmatrix}$
pan/tilt model
$\begin{bmatrix} y_i^{p'} \\ z_i^{p'} \end{bmatrix}$
control algorithm
θ
ϕ

Image plane analysis

① compute the original (unperturbed) image plane point locations as a function of object point locations. The perspective-projection transform.

② Translate the image plane coordinate system to the global coordinate system origin (about which the pan-tilt transformation occur). Note that both coordinate systems here are related by a simple translation; in practice other transformations may also be required.

③ Perform the pan-tilt transformations in the global coordinate system.

④ Transform back to the image plane coordinate system. (This is the inverse of ②).

⑤ Compute the new image plane locations as a function of the object point locations.

⑥ Relate the image plane coordinates in ⑤ to those in ①. This step gives an image-plane to image-plane transformation of the form:

$$[x_i' \quad y_i' \quad w_i' \quad 1] = [x_i \quad y_i \quad w_i \quad 1] \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & a_{43} & 1 \end{bmatrix}$$

a_{ij} is a function of sensor geometry and object/sensor motion.

Not going to derive the transformation, but.

$$\begin{bmatrix} x_i' & y_i' & w_i' & 1 \end{bmatrix} = \begin{bmatrix} x_i & y_i & w_i & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\theta\sin\phi & \frac{\cos\theta\sin\phi}{f} \\ 0 & \cos\theta & -\frac{\sin\theta}{f} \\ -f\sin\phi & f\sin\theta\cos\phi & \cos\theta\cos\phi \\ A & E & F \end{bmatrix}$$

where $A = (f+d_x)\sin\phi + d_y\cos\phi - d_y$

$$E = -(f+d_x)\cos\phi\sin\theta + d_y\sin\phi\sin\theta + d_z\cos\theta - d_z$$

$$F = \frac{-(f+d_x)\cos\phi\cos\theta + d_y\sin\phi\cos\theta - d_z\sin\theta + d_x + f}{f}$$

The physical coordinates of new image points are then given by:

$$x_i^{p'} = \frac{(x_i\cos\phi - w_i f\sin\phi + A)}{G}$$

$$y_i^{p'} = \frac{x_i\sin\theta\sin\phi + y_i\cos\theta + f w_i\sin\theta\cos\phi + E}{G}$$

where $G = \dfrac{x_i\cos\theta\sin\phi - y_i\sin\theta + f w_i\cos\theta\cos\phi + fF}{f}$

Expressing x_i and y_i in physical (image plane) coordinates

$$\left.\begin{aligned} x_i^{p'} &= \frac{x_i^p\cos\phi - f\sin\phi + \frac{A}{w_i}}{H} \\ y_i^{p'} &= \frac{x_i^p\sin\theta\sin\phi + y_i^p\cos\theta + f\sin\theta\cos\phi + \frac{E}{w_i}}{H} \\ \text{where } H &= \frac{x_i^p\cos\theta\sin\phi - y_i^p\sin\theta + f\cos\theta\cos\phi + \frac{fF}{w_i}}{f} \end{aligned}\right\}$$

this is lin
in super
homogeneou
coordinate
non-linear
homogeneou
cordinate:

For a system with a large magnification ratio $\frac{x_0}{f}$.

Suppose $x_0 = 300$ feet (100 meters)

$f = 100$ mm

$$\frac{x_0}{f} = 1000$$

$$w_i = -\frac{x_0}{f} + 1 \quad \text{so} \quad w_i \approx -\frac{x_0}{f}$$

and $\frac{1}{w_i}$ can be neglected in these equations.

Good, because hard to measure offset vector $[d_x \ d_y \ d_z]$ since it's in the camera case.

with this assumption

$$x_i^{p'} = \frac{x_i^p \cos\phi - f\sin\phi + A'}{H'}$$

$$y_i^{p'} = \frac{x_i^p \sin\theta \sin\phi + y_i^p \cos\theta + f\sin\theta\cos\phi + E'}{H'}$$

where
$$A' = -\left[(f + dx)\sin\phi + d_y(\cos\phi - 1)\right]\frac{f}{x_0}$$

$$E' = -\left[-(f+d_x)\cos\phi\sin\theta + d_y\sin\phi\sin\theta + d_z(\cos\theta - 1)\right]\frac{f}{x_0}$$

$$H' = \frac{x_i^p\cos\theta\sin\phi - y_i^p\sin\theta + f\cos\theta\cos\phi + fF'}{f}$$

$$F' = -\frac{\left[-(f+d_x)\cos\theta\cos\phi + d_y\sin\phi\cos\theta - d_z\sin\theta + d_x + f\right]}{x_0}$$

For small-angle approximations

$$x_i^{p'} = x_i^p - f\tan\phi$$

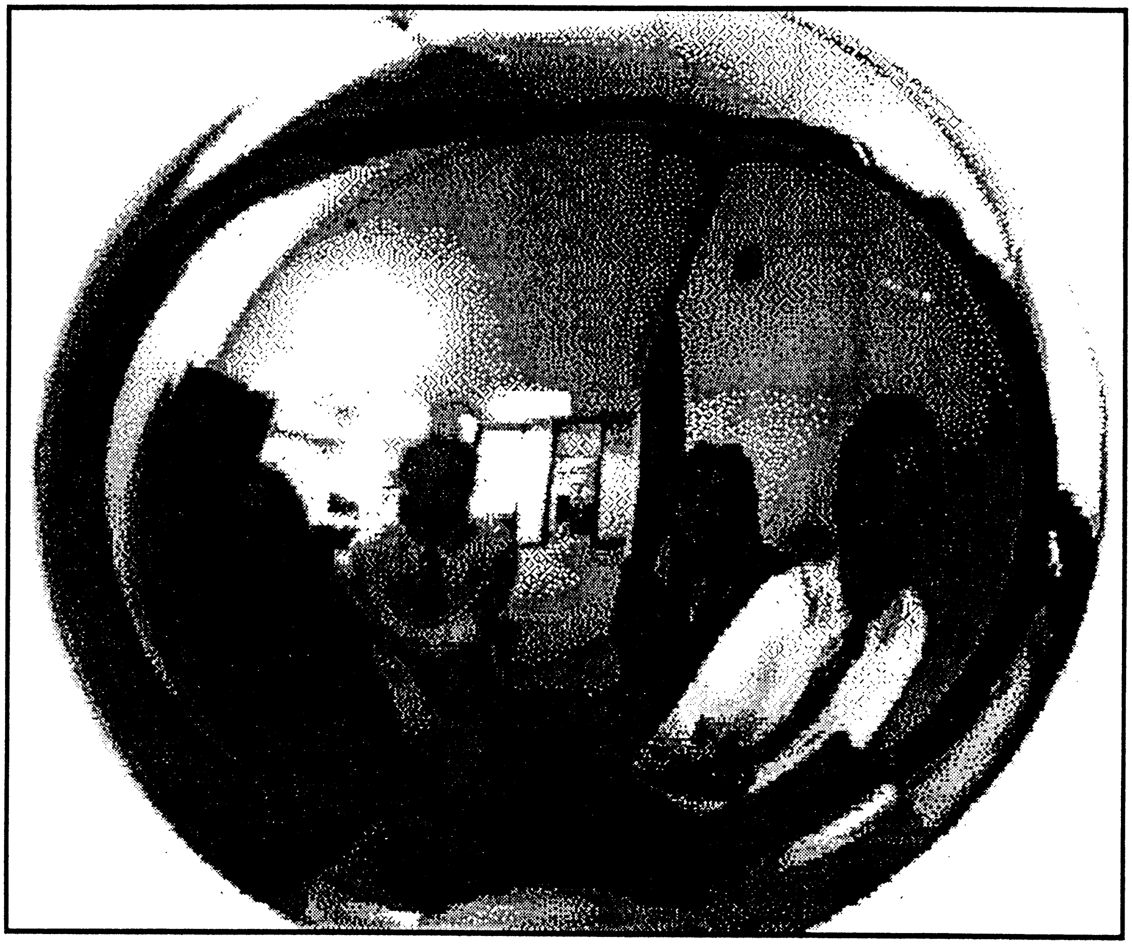
$$y_i^{p'} = x_i^p \tan\theta\tan\phi + y_i^p + f\tan\theta$$

The inverse equations for object centering are then

$$\phi = \mathrm{Tan}^{-1}\left(-\frac{x_i^{p'} - x_i^p}{f}\right)$$

$$\theta = \mathrm{Tan}^{-1}\left(\frac{y_i^{p'} - y_i^p}{x_i^p\tan\phi + f}\right)$$

Bank Robbery Project



The above image was taken by a security camera at a surburban Houston, Texas branch bank during a daring daylight robbery. It shows the reflections of two of the suspects and two hostages in a decorative silver ball that is 30 cm in horizontal diameter. The camera was 30 feet away.

The Chief Investigator, Texas Ranger Captain Billy Clyde Pucket (pucket@texas.rangers.org), believes he knows who the suspects are, and he is trying to get an arrest warrant to bring then in for questioning and for a lineup in front of witnesses before they spend all the money they stole. State Superior Court Judge Roy Whaptner (hanging.roy@frontier.justice.tx), however, refuses to issue the warrants because he cannot identify the suspects by comparing their mug shots (see below) to the picture, due to the geometric distortion produced by the spherical mirror.

Your job is to "unroll" the image so that the four faces appear relatively undistorted, and Capt. Pucket can convince Judge Whaptner to issue a bench warrant for the arrests. First try modeling the mirror as a sphere and using a mathematically-defined warp. If that result is still distorted, try using fiducial markings in the image to square it away. Click on the images to download GIF files. Send your results to ftp.texas.rangers.org/robbery/bank/bad-guys.

Dallas PD 0179254
Perker, Bonnie
(bonnie@oak-cliff.strip-joint.org)

Texas Dept. Public Safety
1707552 - Clyde Barrell
(clyde@petty.larceny.net)

Last modified 7 November, 1995.
This page is maintained by Ken Castleman.
E-mail castlman@phoenix.net with your comments.