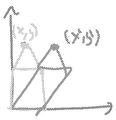
Shear transformation - the bottom is fixed but the top can move/distract/displace
shear happens in sensor of camera

Transformations in homogenous coordinates



$$\begin{cases} y' = y \\ x' = x + Sh_x\, y \end{cases}$$

if the object is lengthier (greater y), then the shear alters the x in proportion to y
y doesnt change.

2D Shear along x relative to the origin:

$$SH_x(s_x) = \begin{bmatrix} 1 & s_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} SH & 0 \\ 0 & 1 \end{bmatrix}$$

Transformations in homogenous coordinates

Transformation properties:

1. Combine by matrix multiplication. $\rightarrow p' = R\,T\,p$ first second
2. Preserve the homogeneous coordinate.
3. Inverted by matrix inversion.

order of multiplication matters based on what happened first.
RTp means the point p was first translated and then rotated.

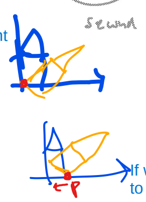
Multiply Tp first and then R (Tp)

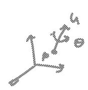
Generally, rotation is done w.r.t origin point

Examples:

- rotation about arbitrary point:

$$R_{p,u}(\theta) = T(p)\,R_u(\theta)\,T(-p)$$





If we want to rotate by an arbitrary point p, then we move the point p to origin, meaning translate by -p and then rotate and then translate back to the place where the point was

- scale about arbitrary point:

$$S_p(s_x, s_y, s_z) = T(p)\,S(s_x, s_y, s_z)\,T(-p)$$

Transformations in homogeneous coordinates

$$(TR)^{-1} = R^{-1}\,T^{-1}$$

this is the formula for inverse. we can also interpret it like, if we want to inverse(cancel) the transformation we did,
we cancel what we did last first and then the second thing last, means in reverse we have to cancel.
So, if we rotated and translated initially, we write it as TR. WHen we cancel, we cancel translation first and then rotation ...so it R-1 T-1

$$T^{-1}(t_x, t_y, t_z) = T(-t_x, -t_y, -t_z)$$

$$S^{-1}(s_x, s_y, s_z) = S(1/s_x, 1/s_y, 1/s_z)$$

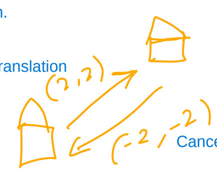
$$R_u^{-1}(\theta) = R_u(-\theta) = R_u^T(\theta)$$

For, rotation inverse and transpose are same, because it is an orthogonal matrix

$$\hookrightarrow R^T R = I$$

Canceling means undoing the action.

Translation



Cancellation of translation

Transformations between coordinate systems

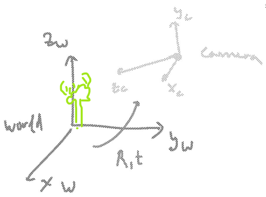
Let (x_w, y_w, z_w) be a world coordinate system. Let (x_c, y_c, z_c) be a camera coordinate system. Assume that the camera coordinate system is translated by t and rotated by R with respect to the world coordinate system.

A point $p^{(w)}$ in world coordinates is given by $p^{(c)}$ in camera coordinates. $p^{(c)}$ is related to $p^{(w)}$ by:

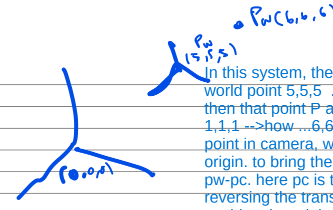
$$p^{(c)} = M_{c \leftarrow w} p^{(w)} \tag{26}$$

The world and camera coordinate system are different. if we want to change from world to camera coorodinates, then we use this formula

$$\begin{aligned} M_{c \leftarrow w} &= (T(t)R)^{-1} && (27) \\ &= R^{-1}T^{-1}(t) && (28) \\ &= R^T T(-t) && (29) \end{aligned}$$



$M_{c \leftarrow w}$: align camera with world (cancel translation then cancel rotation)



In this system, the camera's origin 0,0,0 is actually world point 5,5,5 . Consider a point P 6,6,6 according to world, then that point P according to camera will be 1,1,1 -->how ...6,6,6-5,5,5 --->so to convert the point world to the point in camera, we have to act like the camera origin is the world origin. to bring the world origin to camera origin, we have to do pw-pc. here pc is the camera origin according to world. so, we are reversing the translation of the camera so that we could move the world to the origin of the camera

How did the camera coordinate get there???--->it was at the origin of th world , it was then rotated and trnslatied w.r.t the world coordinate system. Now, we wanted to align the world origin/points with whom as base the camera was rotated to the camera's current origin. So, it actually means to be reversing what we did before.

If we want to reverse it, then cancel translation first and then rotation to get back the world origin and camera origin... same as the point where the previous world coordinate was xc,yc,zc---->So, it looks like as though we are moving the camera back to the world origin 0,0,0 at the end because whatever(world) was used as base to move the camera from the origin of the world to some point of the world, Now we are actually moveing th eworld itself to get back to the origin of the camera.

Transformations between coordinate systems

The rotation R is given by:

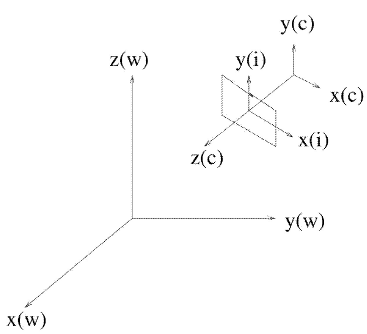
taking the unit vectors of the camera coordinate s/y gives the Rotation matrix that was rotation initially done by camera from the origin of the world coordinate system to the camera coordinate system

$$R = \begin{bmatrix} \hat{x}_c & \hat{y}_c & \hat{z}_c \end{bmatrix}$$

The inverse transformation is given by:

$$\begin{aligned} M_{w \leftarrow c} &= (M_{c \leftarrow w})^{-1} \\ &= (R^T T(-t))^{-1} \\ &= T(t)R \end{aligned}$$

General camera model



General camera model

P is 3dH and p is 2dH

Perspective projection in camera coordinates:

We have $\underbrace{p^{(c)}}_{2dH} = M P^{(c)}_{3dH}$

$$p^{(c)} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P^{(c)} = K[I|0]P^{(c)} \quad (34)$$

The perspective projection we want:

We want $\underbrace{p^{(i)}}_{2dh} = Q \underbrace{P^{(w)}}_{3dh}$ (35)

Almost image coordinate, but 3x1-->2DH, incorporate some camera intrinsic parameters M to attain p(i)

$$\begin{aligned} \underbrace{p^{(i)}}_{2dh} &= M_{i\leftarrow c}p^{(c)} && (36)\\ &= M_{i\leftarrow c}K[I|0]P^{(c)} && (37)\\ &= M_{i\leftarrow c}K[I|0]M_{c\leftarrow w}\underbrace{P^{(w)}}_{3d} && (38) \end{aligned}$$

Assuming that the camera is translated by t and rotated by R with respect to the world ($R = [\ \hat{x}_c \quad \hat{y}_c \quad \hat{z}_c\ ]$):

perspective projection -a technique used to project three-dimensional points onto a two-dimensional plane, mimicking how the human eye perceives depth and distance.

p(c) = (X',Y',Z') represent the transformed position of a 3D point in the camera's perspective, where and are used for mapping to the 2D image plane, and Z provides depth information essential for understanding the spatial relationship of objects in the scene.

when we divide by Z', we get the 2d point, However we should take into account, the values of ku,kv,uo,vo-->and that is why we have the extra matrix Mi<-c

General camera model

As detailed in before slides

$$\begin{aligned} M_{c\leftarrow w} &= R^T T(-t) \\ &= \begin{bmatrix} R & 0 \\ 0 & 1 \end{bmatrix}^T \begin{bmatrix} I & -t \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R^T & -R^T t \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} R^* & T^* \\ 0 & 1 \end{bmatrix} \end{aligned}$$

The rotation/translation of the world with respect to the camera are extrinsic camera parameters:

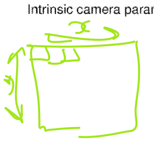
Camera caliberation- generally returns R* and T*

$$\begin{aligned} R^* &= R^T \\ T^* &= -R^T t \end{aligned}$$

While R and T are intrinsic camera parameters where the rotation/translation of the camera happens w.r.t to origin of the world

General camera model

Intrinsic camera parameters:



notation	meaning	units
k_u	scale in x relating pixels to mm	pixels/mm
k_v	scale in y relating pixels to mm	pixels/mm
u_0	translation of the principal point in x	pixels
v_0	translation of the principal point in y	pixels
f	focal length	mm
α_u, α_v	focal length	pixels

Ku= number of pixels/mm in x
Kv = number of pixels/mm in y
if the pixels are square, then Ku = kv

The transformation $M_{i\leftarrow c}$ is composed of scale and translation:

$$M_{i\leftarrow c} = \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

u0 and v0 corresponds to the translation of the center of the image w.r.t to the camera coordinate system

u0 and v0 are the same for an ideal camera.
For a 1000x1000 image, u0=v0=500.
However, if there is a defect in the camera, it would be different.

While going from camera to image coordinates, there is no rotation. but there is translation, because the origin of the camera corrdinate system and image center point vary. There is also scaling transformation as camera corrdinate is in mm and image is in pixel.

General camera model

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Combining the transformation matrices:

$$\begin{aligned} p^{(i)} &= M_{i \leftarrow c} K [I|0] M_{c \leftarrow w} P^{(w)} \\ &= K^*[I|0] \begin{bmatrix} R^* & t^* \\ 0 & 1 \end{bmatrix} P^{(w)} \\ &= K^*[R^*|t^*] P^{(w)} \end{aligned}$$

This is the perspective projection that also includes the transformations in the coordinate s/y

The matrix K^* contains the intrinsic camera parameters:

$$\begin{aligned} K^* &= M_{i \leftarrow c} K \\ &= \begin{bmatrix} k_u & 0 & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Look at example sum in word document

The parameters α_u, α_v specify scale in pixels.

General camera model

Shear is the smallest distortion that happens from the sensors. It is usually acceptable but for important tasks that need high precision and care, it needs to be corrected.

To correct this, we add the additional terms in the K* term

When allowing for shear:

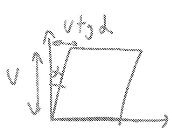
$$\begin{aligned} K^* &= \begin{bmatrix} \alpha_u & \alpha_u \tan\alpha & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

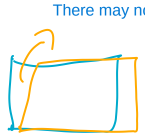
$s \approx 0$

s is almost 0, it is very small value

The parameter s is skew (in pixels).

There may not be this big a diff but there may be 0.1pixel diff - but accuracy matters in some sensitive tasks





Radial lens distortion

Refer diagram from notes doc

$$p^{(i)} = \begin{bmatrix} 1/\lambda & & \\ & 1/\lambda & \\ & & 1 \end{bmatrix} K^*[R^*|T^*] P^{(w)}$$

wide angle lens usually creates distortion
Eg - security cameras, classroom cameras

More distortion seen away from the center of the image

linear distortion coefficient

$$\lambda = 1 + k_1 d + k_2 d^2$$

quadratic distortion coefficient

The lines are supposed to be parallel as per real world but distorted due to wide angle lens.

d = distance from center

the lambda value depends on the distance (d) between the distorted point and the center of the image.

larger shrink away from the center

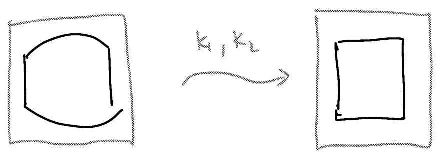
k1 and k2 depends on the camera calibration

Here , we again have division and hence it is non linear. So we cannot use it directly.
Instead, we transform the distorted image to the original (undistorted world type)image using warp, then we use that image for further computation

there is a warp function to do this

Radial lens distortion

Warp the image to correct distortion
(warp using estimated distortion parameters)



Weak perspective camera

Perspective

$$\underset{3\times 4}{M} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix}$$

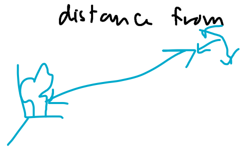
weak perspective

$$M_\infty = \underset{3\times 4}{\begin{bmatrix} \text{———} \\ \text{———} \\ 0 \quad 0 \quad 0 \quad 1 \end{bmatrix}}$$

Weak perspective camera

placing the object far from the camera, makes it a weak perspective camera

weak perspective is correct when depth variation in the scene is small compared with distance from camera



depth variation

$$e = |M_\infty P - MP| = \frac{\Delta}{d_0}(MP - P_0)$$

distance from center

distance from camera

Affine is not real but used for simplification

Affine camera

used to convert 3dh(4x1) to 2dH (3x1)

$$M_{affine} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

computational model