Transformations… continued

Lecture 6A

Summary: 3D Rotation Matrices

- Rotations about Principal Axes

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma & 0 \\ 0 & \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

About origin, in right handed coordinate system, counter clockwise when looking towards origin from positive axis

- Rotation matrix is orthonormal with Determinant of +1 and 3 dof
- Inverse of a rotation matrix is its transpose
- Concatenation of Rotations is also a rotation
- Small Angle Approximation
  - Sub cosθ =1 and sin θ = θ
  - Multiplication of two small angles = 0
- IMP: A rotation matrix transforms its own rows onto the principal axes
- Any 3D rotation matrix can be described as rotation about an axis **n** by an angle θ
- To rotate about given axis **n** by θ:
  - Rotate axes onto a principal axis
    - Two ways: by computing principal rotations or by composing appropriate matrix through cross products
  - Rotate about principal axes and then undo the earlier transformation
- To compute **n** and θ from a 3D rotation matrix
  - n is the eigenvector corresponding to the real eigenvalue of 1
  - θ can be computed by the other 2 eigenvalues, which are $\cos\theta \pm i\sin\theta$

Factorizing Transformations

- Opposite of Concatenation of Transformations
- Given a transformation matrix, decompose it into a sequence of simpler transformations
- Example:

$$\begin{bmatrix} a_1 & a_2 & b_1 \\ a_3 & a_4 & b_2 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & 0 \\ a_3 & a_4 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Question: How to factorize the multiplicative part?

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix}$$

- Is the factorization unique?

- Special Case: **A** is symmetric
  - Eigen values of a real symmetric matrix are real
  - Its eigenvectors can always be written as orthonormal matrix

$$\mathbf{A}\Phi=\Phi\Lambda$$

$$\mathbf{A}=\Phi\Lambda\Phi^T \qquad \text{(Implication?)}$$

- A non symmetric real matrix **M** can be decomposed as **M** = **U S V**^{T} (with **U** and **V** being orthonormal, **S** being a diagonal
- To compute U S and V,
  - Let $A = MM^T$
  - $A = (USV^T)(USV^T)^T$
  - $A = US^2U^T \qquad V^T = (US)^{-1}M$

Singular Value Decomposition

- Let **M** be a *m*-by-*n* matrix whose entries are real numbers. Then **M** may be decomposed as

  $$\mathbf{M} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^{T}$$

  where:
  - **U** is an *m*-by-*m* orthonormal matrix
  - **S** is an *m*-by-*n* matrix with non-negative numbers on the main diagonal and zeros elsewhere
  - **V** is an *n*-by-*n* orthonormal matrix
- Example

$$\begin{bmatrix}1&0&0&0&2\\0&0&3&0&0\\0&0&0&0&0\\0&4&0&0&0\end{bmatrix} = \begin{bmatrix}0&0&1&0\\0&1&0&0\\0&0&0&-1\\1&0&0&0\end{bmatrix} \cdot \begin{bmatrix}4&0&0&0&0\\0&3&0&0&0\\0&0&\sqrt{5}&0&0\\0&0&0&0&0\end{bmatrix} \cdot \begin{bmatrix}0&1&0&0&0\\0&0&1&0&0\\\sqrt{0.2}&0&0&0&\sqrt{0.8}\\0&0&0&1&0\\-\sqrt{0.8}&0&0&0&\sqrt{0.2}\end{bmatrix}$$

http://en.wikipedia.org/wiki/Singular_value_decomposition

Singular Value Decomposition

- Implication: We can take the multiplicative part of any transform and describe it as a sequence of a rotation, scaling and another rotation
- 2D Example: Decomposing an Affine Transformation

```
M =   0.95    0.49    0.46
      0.23    0.89    0.02
       0       0       1
```

```
>> U * S *V'

ans =

      0.95    0.49
      0.23    0.89
```

```
>> [U, S,V] = svd(M(1:2, 1:2))

U =
    -0.78156    -0.62384
    -0.62384     0.78156

S =
      1.2904          0
         0      0.56789

V =
    -0.68658    -0.72705
    -0.72705     0.68658
```

Interpretation in terms of angles?

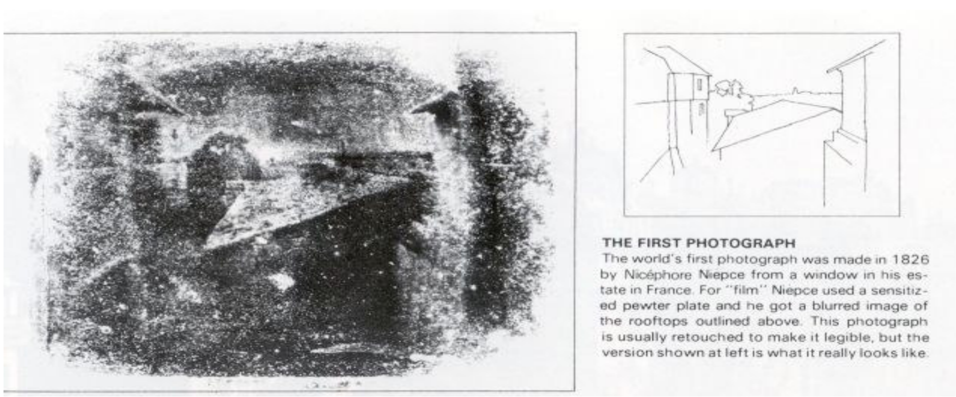
Singular Value Decomposition

- Implications: Even a simple shear can be written as a rotation$\rightarrow$scaling$\rightarrow$rotation
- Try visualizing it to understand how… [Exercise]

Summary of Transformations

Summary: 2D and 3D Transformations

- Image Registration
- 2D Transformations
  - Scaling
  - Shear
  - Rotation
  - Translation
- Inverse Transformations
- Rotation about an arbitrary point
- Concatenation of transformations
- Order of transformations
- Factorization of Transformations
- Displacement Models
  - Rigid / Euclidean
  - Similarity
  - Affine
  - Projective
  - Billinear, biquadratic etc
- Recovering the best affine transformation
  - Least Squared Error solution
  - Pseudo inverse
- Image Warping
- 3D Transformations
  - Rotations about Principal Axes
  - Rotations about Arbitrary Axes
- Properties of Rotation Matrices



Camera Model

Lecture 6B

Modeling a Camera

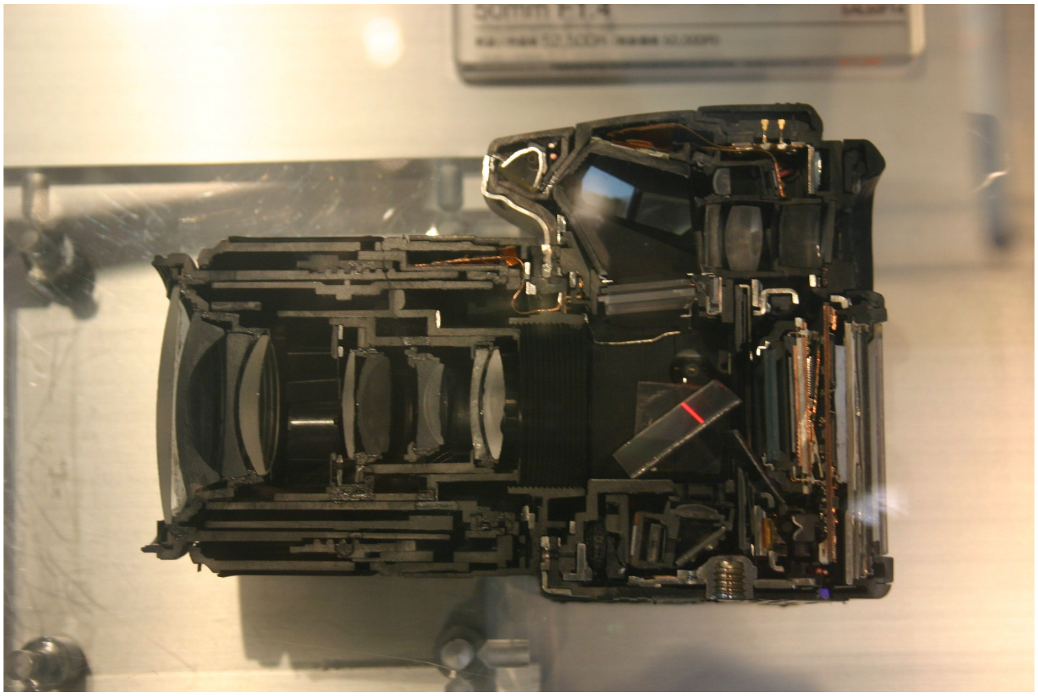
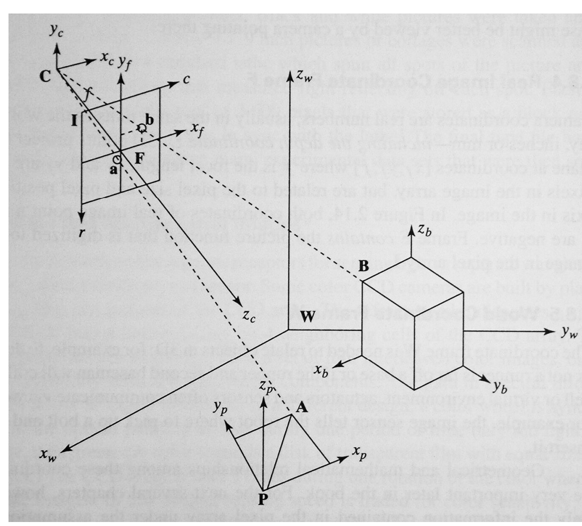


Image by Dr Yaser Sheikh, CMU

Frames of Reference

- World Coordinate Frame, W
- Object Coordinate Frame, O
  e.g. B, P
- Camera Coordinate Frame, C
- Real Image Coordinate Frame, F
- Pixel Coordinate Frame, I



Aperture vs Shutter speed

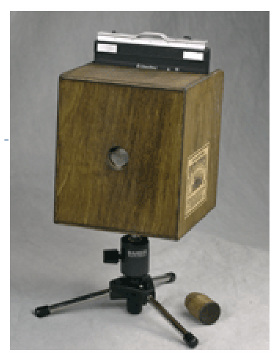
- If **shutter speed** is doubled, and **aperture area** is doubled, the same amount of light should enter the camera
- Therefore, to shoot an image, there are several valid combinations of aperture and shutter speed
- High shutter speed: for fast moving objects
- Large aperture: low depth of field

Focus

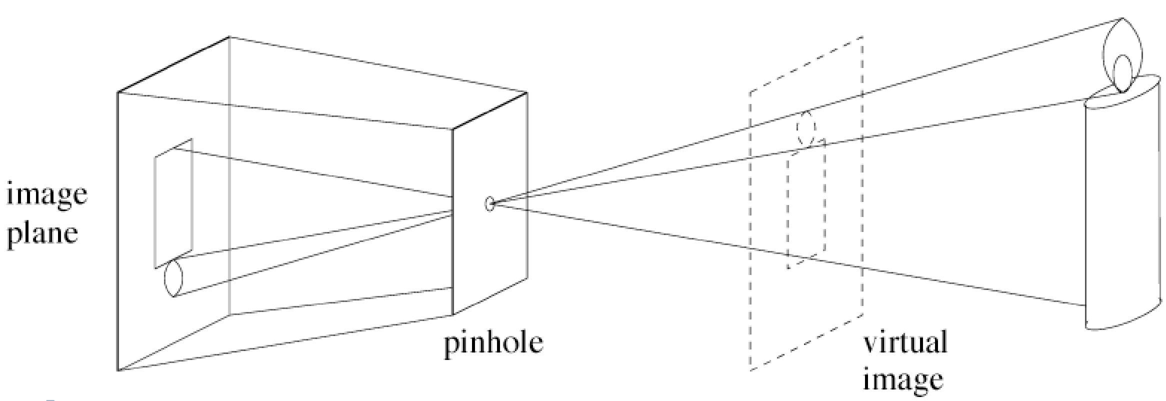
- In general, any single point on the film can have light coming from different directions
- Therefore a single point in the world may be mapped to several locations in the image
- This generates blur
- To remove blur, all rays coming from a single world point must converge to a single image point

Example of Shallow Depth of Field

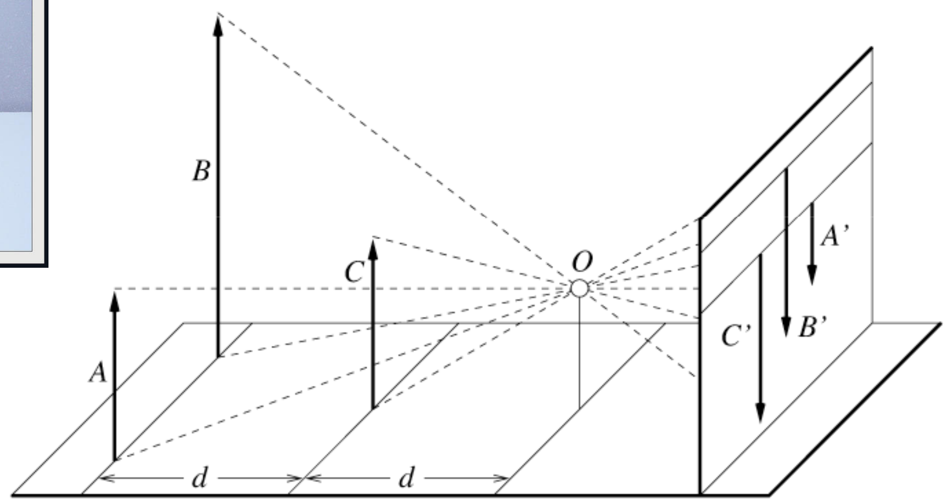
Pinhole Camera



- Lens is assumed to be single point
- Infinitesimally small aperture
- Has infinite depth of field i.e. everything is in focus



Distant objects are smaller



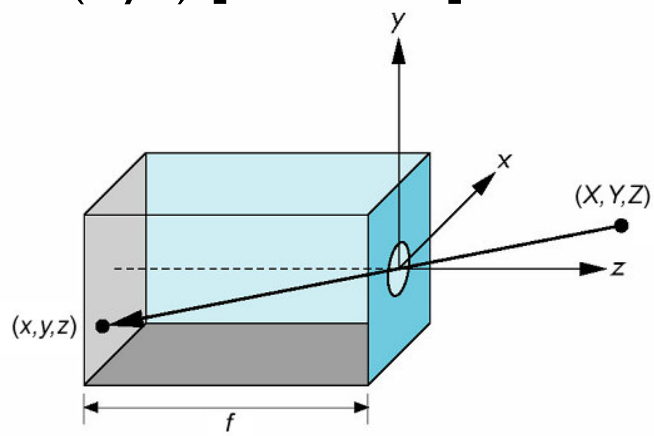
Slide Credit: Forsyth/Ponce http://www.cs.berkeley.edu/~daf/bookpages/slides.html and Khurram Shafique, Object Video

Pinhole Camera

- Advantage
  - Because of small aperture, everything is in focus (infinite depth of field)
  - Simple construction
- Disadvantage
  - Small aperture requires high exposure time, often too long for practical purposes

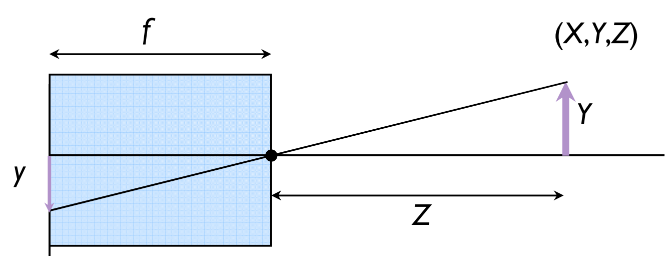
Image Formation – The Pin-Hole Camera

- Orient along z-axis
- World point (X,Y,Z) [camera frame]
- Image point at (x,y,z) [real frame]



Perspective Transform

Equation relating world coordinate and image coordinate?



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

$$y = -\frac{fY}{Z} \qquad x = -\frac{fX}{Z}$$

It is customary to use a negative sign to indicate that the image is always formed upside down

Perspective Transform

- We can write this as a matrix using the homogeneous coordinates

$$\begin{bmatrix} hx \\ hy \\ hz \\ h \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

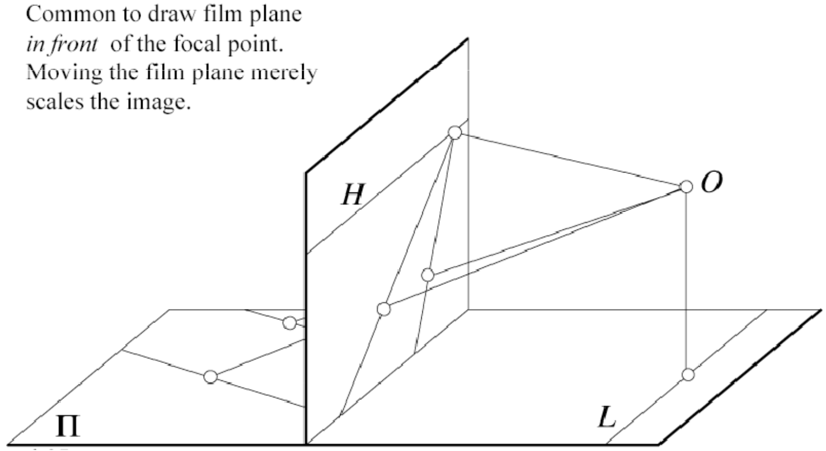
$$\begin{aligned} hx &= X \\ hy &= Y \\ h &= -\frac{Z}{f} \end{aligned}$$

$$x = -\frac{fX}{Z} \qquad y = -\frac{fY}{Z}$$

Perspective Transform: Some Properties

- Lines map to lines
- Polygons map to polygons
- Parallel lines meet

Common to draw film plane *in front* of the focal point. Moving the film plane merely scales the image.



Perspective Transform

- This relates the camera frame to the real image frame
- Example:
  - I take the image of a person (2m tall) standing 4m away from the camera, with a 35 mm camera using the geometry shown previously. How high will be the image?
  - Answer: $y = -(35)(2000)/4000 = -17.5$mm
  - i.e, the image will be formed inverted of length 17.5 mm
- How to convert to pixel frame (i.e. what will be the coordinates of the head of the person in the image?

Perspective Transform

- Suppose I know that the size of the film is 8cm x 6cm, and that the resolution of the camera is 640 x 480 pixels
- Implies, the center of the image is at 4cm x 3cm from the corner, and is at location (240, 320)
- Image will first be made right side up
- 17.5mm out of 60mm is 140 out of 480 pixels
- Hence the coordinates of the head will be (240-140 in x, same in y) = (100, 320)

Perspective Projection

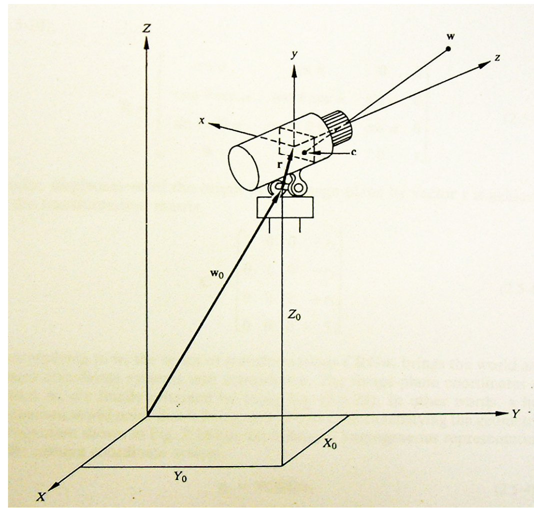
- This is for the case when the camera's optical axis is aligned with the world z-axis
  - Or: it relates camera frame to real image frame
- What if that is not the case?

Camera Model

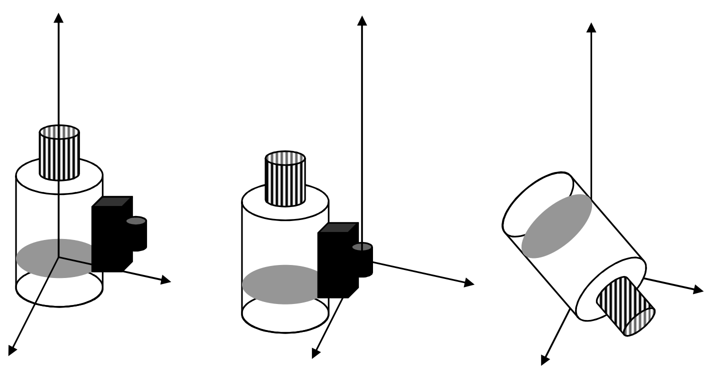
- If the camera is moved **T** from the origin, we should move the world point by $\mathbf{T}^{-1}$
- Then the perspective transform equation will be applicable
- Same holds for rotations

Camera Model

- Think that the camera was originally at the origin looking down Z axis
- Then it was translated by $(r_1, r_2, r_3)^T$, rotated by ϕ along X, θ along Z, then translated by $(x_0, y_0, z_0)^T$
- This is the scenario in the figure on right



Camera Model



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & r_1 \\ 0 & 1 & 0 & r_2 \\ 0 & 0 & 1 & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1}$$

$$\boxed{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{f} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$C_h = PCR^X_{-\phi}R^Z_{-\theta}GW_h$$

Camera Model

$$C_h = PCR^X_{-\phi}R^Z_{-\theta}GW_h$$

$$\text{where } P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix}, R^Z_{-\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$R^X_{-\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, G = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

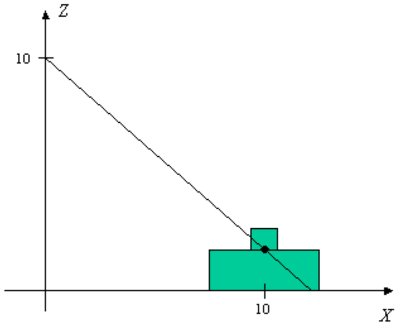
Camera Model

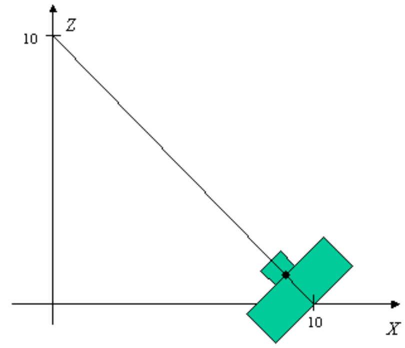
$$x = f\frac{(X - X_0)\cos\theta + (Y - Y_0)\sin\theta - r_1}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f},$$

$$y = f\frac{-(X - X_0)\sin\theta\cos\phi + (Y - Y_0)\cos\theta\cos\phi + (Z - Z_0)\sin\phi - r_2}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f}.$$

- This camera model is applicable in many situations
- For example, this is the typical surveillance camera scenario

Examples





$$C_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{0.01} & 1 \end{bmatrix} \mathbf{T}^{-1} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

$$C_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{0.01} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

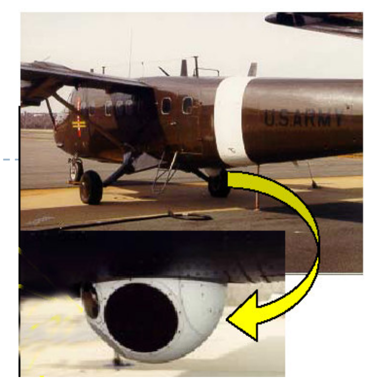
$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{0.01} & 1 \end{bmatrix} \begin{bmatrix} -10 \\ 0 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} -10 \\ 0 \\ 10 \\ -999 \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(-45^\circ) & 0 & \sin(-45^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(-45^\circ) & 0 & \cos\cos(-45^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C_h = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{0.01} & 1 \end{bmatrix} \mathbf{T}^{-1} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

0
0
-0.01
1

Aircraft Example



OTTER	system_id
TV	sensor_type
0001	serial_number
9.400008152666640300e+08	image_time
3.813193746469612200e+01	vehicle_latitude
-7.734523185193877700e+01	vehicle_longitude
9.949658409987658800e+02	vehicle_height
9.995171174441039900e-01	vehicle_pitch
1.701626418113209000e+00	vehicle_roll
1.207010551753029400e+02	vehicle_heading
1.658968732990974800e-02	camera_focal_length
-5.361314389557259100e+01	camera_elevation
-7.232969433546705000e+00	camera_scan_angle
480	number_image_lines
640	number_image_samples

cameraMat = perspective_transform * gimbal_rotation_y * gimbal_rotation_z * gimbal_translation * vehicle_rotation_x * vehicle_rotation_y * vehicle_rotation_z * vehicle_translation ;

$$\Pi_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\Delta T_x \\ 0 & 1 & 0 & -\Delta T_y \\ 0 & 0 & 1 & -\Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
c(1,1) = (cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg);
c(1,2) = -(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg);
c(1,3) = -cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll);
c(1,4) = -((cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg))*vx-(-(cos(c_scn)*cos(v_rll)-
         sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg))*vy-(-cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll))*vz;

c(2,1) = (-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg);
c(2,2) = -(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg);
c(2,3) = sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll);
c(2,4) = -((-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-
         sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-
         cos(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(3,1) = (cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*sin(v_hdg);
c(3,2) = -(cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*cos(v_hdg);
c(3,3) = -cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll);
c(3,4) = -
         ((cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-
         (cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(4,1) =
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_s
         cn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg);
c(4,2) = -
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_sc
         n)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg);
c(4,3) = -1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll);
c(4,4) = -
         ((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_s
         cn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_sc
         n)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-
         1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*vz+1;
```

Weak Perspective Projection

- Approximation to Perspective projection, approximately valid when distance of the camera is much greater than the depth variation of the object

$$x = -\frac{fX}{\bar{Z}} \qquad y = -\frac{fY}{\bar{Z}} \qquad \text{or} \qquad x = mX \qquad y = mY$$

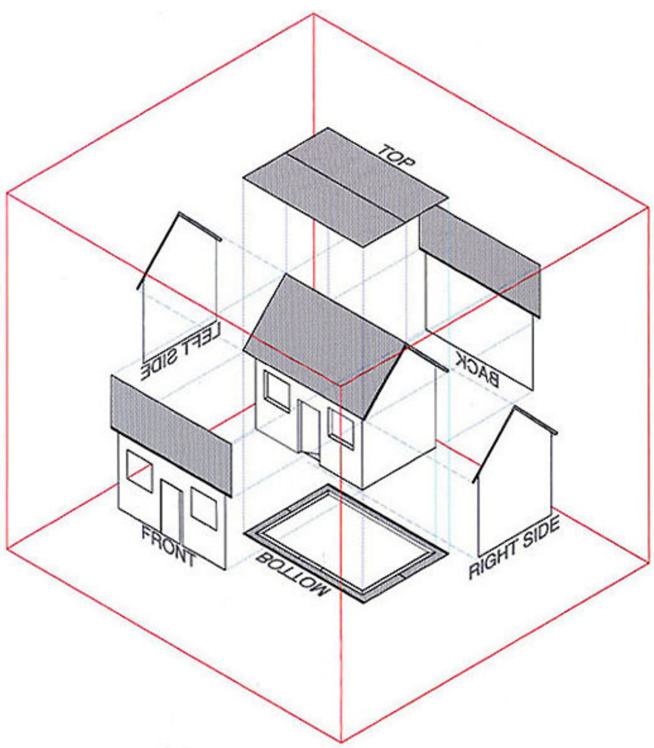
- Advantage: Computationally simpler [why?]
- Disadvantage: Not physically accurate

Orthographic Projection

- Scaling of weak perspective projection

$$x = X \quad y = Y$$

- Parallel lines remain parallel
- Useful for engineering drawings, scrolls, where the perspective shortening is not desired
- Computationally simpler



http://www2.arts.ubc.ca/TheatreDesign/crslib/drft_1/orthint.htm