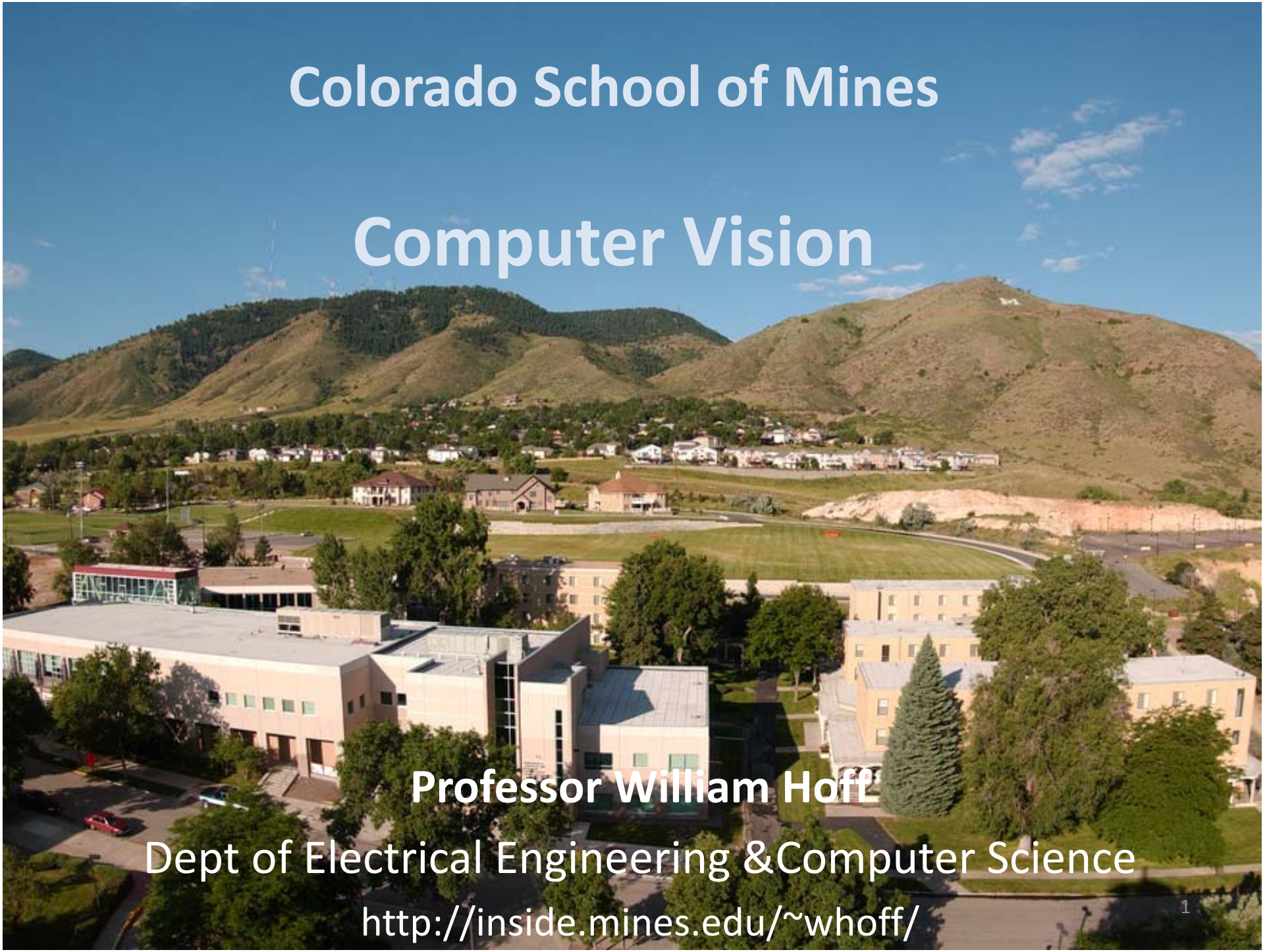

Colorado School of Mines
Computer Vision
Professor William Hoff
Dept of Electrical Engineering &Computer Science
http://inside.mines.edu/~whoff/

3D-2D Coordinate Transforms

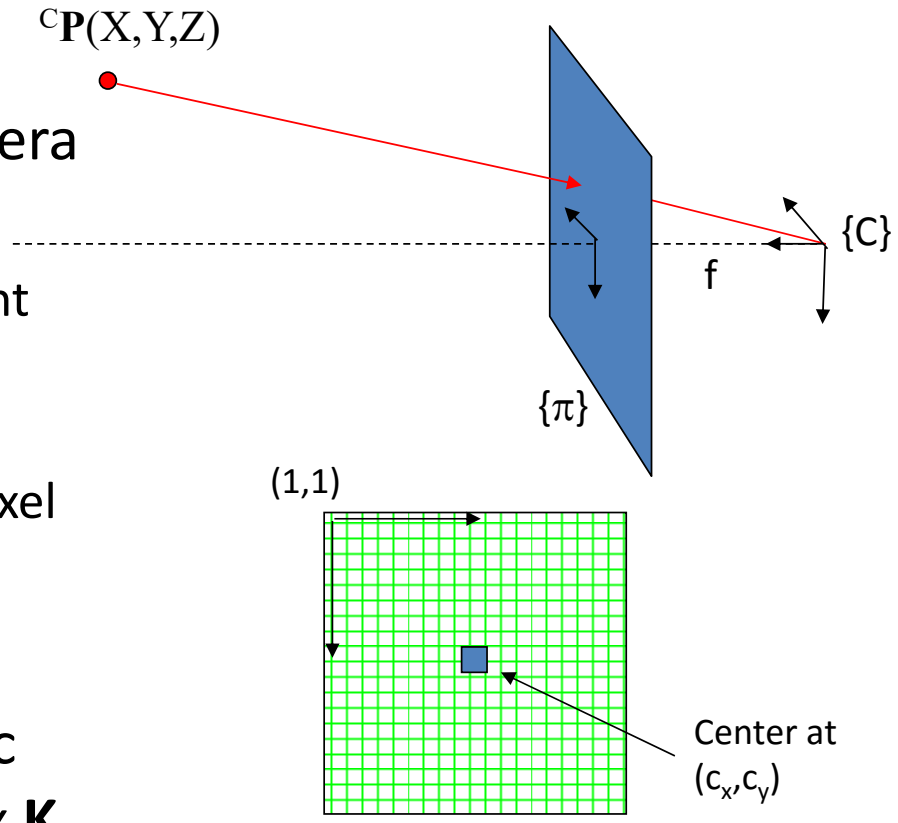
3D to 2D Projections

- We have already seen how to project 3D points onto a 2D image
  - We used the pinhole camera model
  - Also the geometry of similar triangles
- Now we will look at how to model this using matrix multiplication

- This will help us better understand and model:
  - Perspective projection
  - Other projection types, such as weak perspective projection
  - Special cases such as the projection of a planar surface

Intrinsic Camera Parameters

- Recall the intrinsic camera parameters, for a pinhole camera model
  - Focal length *f* and sensor element sizes *sx,sy*
    - Or, just focal lengths in pixels *fx,fy*
  - Optical center of the image at pixel location *cx, cy*



- We can capture all the intrinsic camera parameters in a matrix **K**

$$\mathbf{K} = \begin{pmatrix} f/s_x & 0 & c_x \\ 0 & f/s_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \quad \text{or} \quad \mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

3D to 2D Perspective Transformation

- We can project 3D points onto 2D with a matrix multiplication

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

- We treat the result as a 2D point in homogeneous coordinates. So we divide through by the last element.

$$\tilde{\mathbf{x}} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X/Z \\ Y/Z \\ 1 \end{pmatrix}$$

- Recall perspective projection is: $x = f\,X/Z + cx,\ y = f\,Y/Z + cy$
  - If f=1 and (cx,cy) = (0,0), then this is the image projection of the point
  - As we will see later, it is often useful to consider the projection of a point as if it were taken by a camera with f=1 and (cx,cy) = (0,0)
  - These are called "normalized image coordinates"

Complete Perspective Projection

- To project 3D points *represented in the coordinate system attached to the camera*, to the 2D image plane:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^{\mathrm{C}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 / x_3 \\ x_2 / x_3 \\ 1 \end{pmatrix} \qquad \mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

- To see this:

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} {}^{\mathrm{C}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} f_x X + c_x Z \\ f_y Y + c_y Z \\ Z \end{pmatrix} \sim \begin{pmatrix} f_x X / Z + c_x \\ f_y Y / Z + c_y \\ 1 \end{pmatrix}$$

Extrinsic Camera Matrix

- If 3D points are in world coordinates, we first need to transform them to camera coordinates

$$ {}^{C}\mathbf{P} = {}^{C}_{W}\mathbf{H}\,{}^{W}\mathbf{P} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \\ \mathbf{0} & 1 \end{pmatrix} {}^{W}\mathbf{P} $$

- We can write this as an extrinsic camera matrix, that does the rotation and translation, then a projection from 3D to 2D

$$ \mathbf{M}_{ext} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{pmatrix} $$

Complete Perspective Projection

- Projection of a 3D point ${}^{W}\mathbf{P}$ in the world to a point in the pixel image (x_{im}, y_{im})

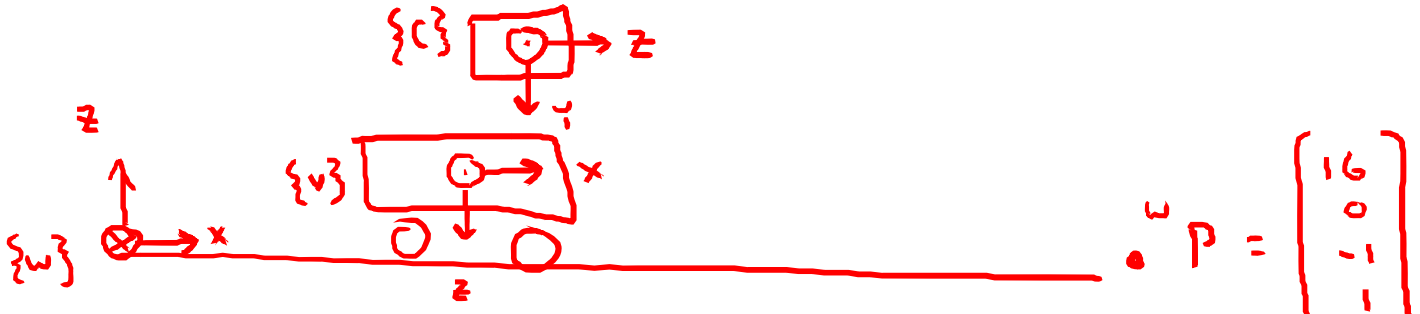
$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{K}\,\mathbf{M}_{ext} \; {}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \qquad x_{im} = x_1 / x_3, \;\; y_{im} = x_2 / x_3$$

  - where **K** is the intrinsic camera parameter matrix
  - and $\mathbf{M}_{ext}$ is the 3x4 matrix given by

$$\mathbf{M}_{ext} = \begin{pmatrix} {}^{C}_{W}\mathbf{R} & {}^{C}\mathbf{t}_{Worg} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{pmatrix}$$

Example

- A camera is mounted on a vehicle. It observes a point **P** in the world at (*X,Y,Z*) = (16,0,-1). What pixel would **P** project to?
- Assume *f*=512 pix, (*cx,cy*)=(256,256)



$${}^{w}P = \begin{bmatrix} 16 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

$${}^{w}_{v}H = \begin{bmatrix} 1 & 0 & 0 & 3 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad {}^{v}_{c}H = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

$$p = K M_{ext} {}^{w}P$$

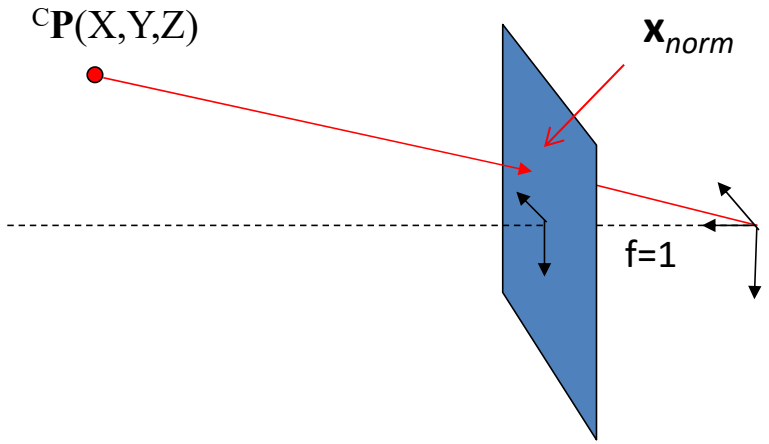
```
H_V_W = [ 1  0  0   5;
          0 -1  0   0;
          0  0 -1   1;
          0  0  0   1]

 H_S_V = [ 0  0  1   1;
           1  0  0   0;
           0  1  0  -2;
           0  0  0   1]

P_W = [ 16; 0; -1; 1];

K = [512  0   256;
      0  512  256;
      0   0    1 ];

R_C_V = [ 0  0  1;
          1  0  0;
          0  1  0];
R_V_C = R_C_V';

R_W_V = [1  0  0;
         0 -1  0;
         0  0 -1]';

R_W_C = R_V_C * R_W_V;

tCorg_V = [1; 0; -2; 1];

tCorg_W = H_V_W * tCorg_V;
tCorg_W = tCorg_W(1:3);

Mext = [ R_W_C   -R_W_C * tCorg_W ];

p = K * Mext * P_W;
p = p / p(3)
```

Back Projection

- If you have an image point, you can "back project" that point into the scene
- However, the resulting 3D point is not uniquely defined
  - It is actually a ray emanating from the camera center, out through the image point, to infinity
  - Any 3D point along that ray could have projected to the image point

- First, we convert the image point to "normalized" coordinates

$$\mathbf{x}_{norm} = \mathbf{K}^{-1}\,\mathbf{x}_{un-norm} = \begin{pmatrix} X/Z \\ Y/Z \\ 1 \end{pmatrix}$$



r = *s***x**$_{norm}$ *is a ray emanating outward to infinity*

Example

- Assume that the cameraman image was taken using a camera with focal length = 600 pixels, with *cx,cy* in middle of image.
  - Find the unit vector in the direction of the man's eye

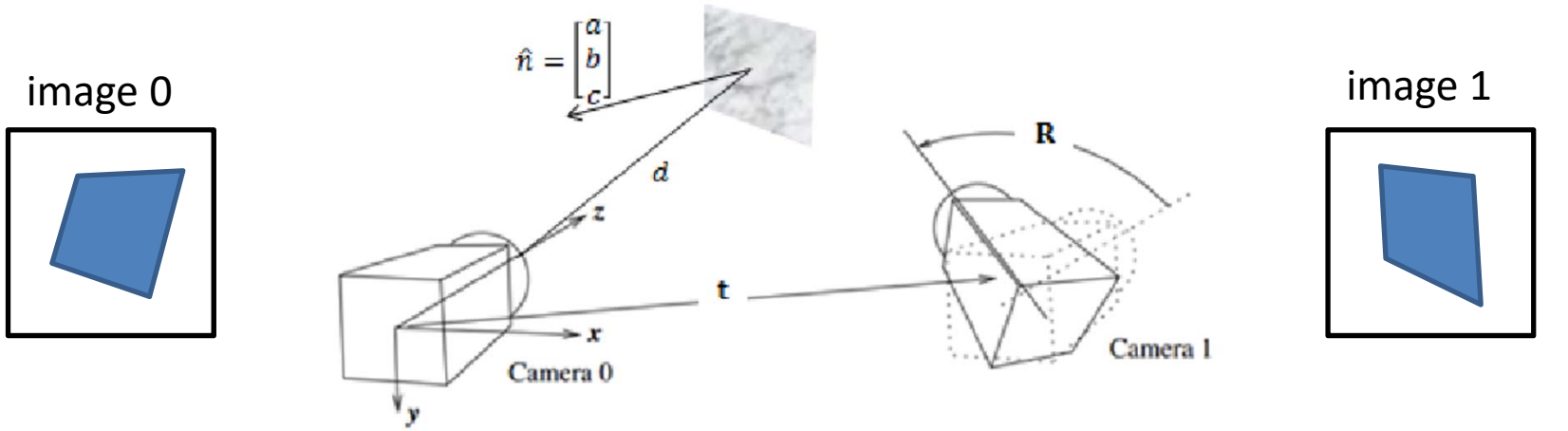
- Assume that the cameraman image was taken using a camera with focal length = 600 pixels, with cx,cy in middle of image.
  - Find the unit vector in the direction of the man's eye
- Solution:
  - The eye is at pixel (ximg=126, yimg=61)
  - Then pimg = K*pn, or pn = Kinv*pimg
  - Let u=unit vector to the eye = pn/norm(pn)

```
I = imread('cameraman.tif');
imshow(I,[]);

xeye = 126;
yeye = 61;
rectangle('Position', [xeye-5 yeye-5 10 10], 'EdgeColor', 'r');
K = [600  0  128;
       0  600 128;
       0   0   1];

pn = inv(K)*[ xeye; yeye; 1];

u = pn / norm(pn);
disp(u);
fprintf('u = %f %f %f\n', u(1), u(2), u(3));
```

u = -0.003313 -0.110976 0.993818

Special Case

- Consider a plane observed from two viewpoints
- It can be shown that the image points are transformed using a homography (i.e., an arbitrary 3x3 transformation)



$$\tilde{\mathbf{x}}_1 \sim \mathbf{H}\,\tilde{\mathbf{x}}_0$$

2D points in image 1 (homogeneous coords)

2D points in image 0 (homogeneous coords)

$$\begin{pmatrix} x_a \\ x_b \\ x_c \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_0 \\ y_0 \\ 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \begin{pmatrix} x_a / x_c \\ x_b / x_c \\ 1 \end{pmatrix}$$

Weak Perspective

- Sometimes it is better to use an approximation to perspective projection, called “weak” projection or scaled orthography
- This works if the average depth Z_{avg} to an object is much larger than the variation in depth within the object
  - Instead of $x = f\,X/Z$, $y = f\,Y/Z$
  - use $x = f\,X/Z_{avg}$, $y = f\,Y/Z_{avg}$

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & Z_{avg} \end{pmatrix} \overset{C}{\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}}, \qquad \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x_1 / x_3 \\ x_2 / x_3 \\ 1 \end{pmatrix}$$

- This makes the image coordinates (x,y) a linear function of the 3D coordinates (X,Y,Z)

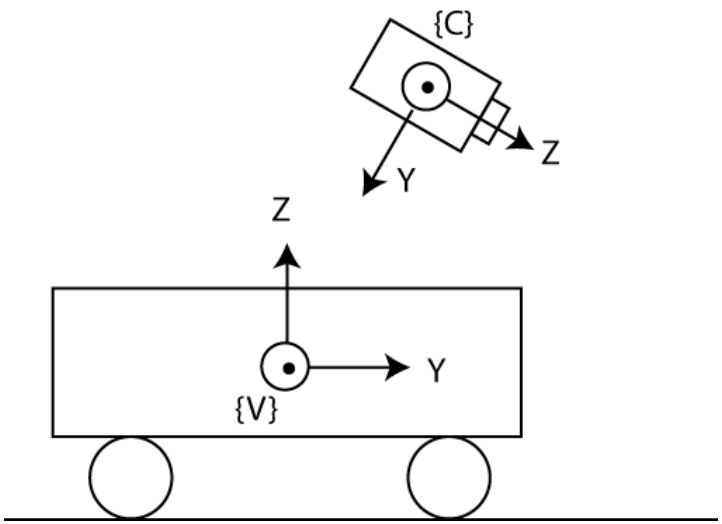
Special Case

- Small planar patch
  - Often we want to track a small patch on an object
  - We want to know how the image of that patch transforms as the object rotates
- Assume
  - Size of patch small compared to distance -> *weak perspective*
  - Rotation is small -> *small angle approximation*
  - Patch is planar
- It can be shown that the patch undergoes affine transformation

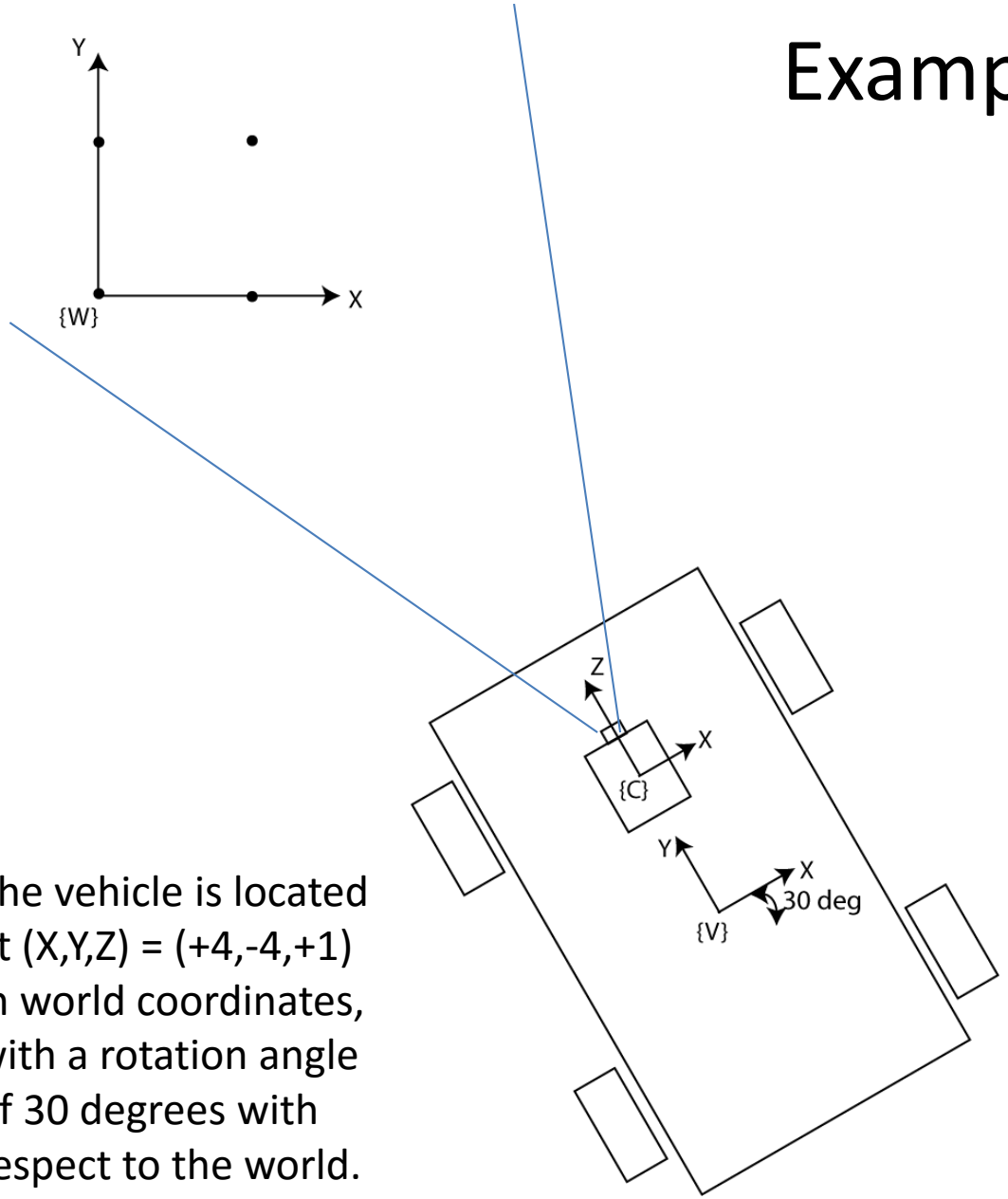
$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix}$$

Example

- A camera is mounted on a robot vehicle. The center of the camera is located at (X,Y,Z) = (0,+1 ,+2) in vehicle coordinates, and the camera is tilted downwards by 30 degrees



Example (continued)



- The vehicle is located at (X,Y,Z) = (+4,-4,+1) in world coordinates, with a rotation angle of 30 degrees with respect to the world.

- The camera observes four points on the ground, with world coordinates at (0,0,0), (1,0,0), (1,1,0), and (0,1,0).

- Generate a synthetic image of the points as seen from the camera, assuming the image is 200 pixels high by 300 pixels wide, with an effective focal length of 300 pixels

Solution:
We need to convert the four points from world coordinates to camera coordinates. If we knew the transformation from the vehicle to the world (${}^{W}_{V}\mathbf{H}$) and the camera to the vehicle (${}^{V}_{C}\mathbf{H}$), we can combine the transformations using ${}^{C}\mathbf{P} = {}^{C}_{V}\mathbf{H}\ {}^{V}_{W}\mathbf{H}\ {}^{W}\mathbf{P}$, where ${}^{V}_{W}\mathbf{H} = \left({}^{W}_{V}\mathbf{H}\right)^{-1}$, etc.

The transformation from vehicle to world: The vehicle is aligned with the world except for a rotation of +30 degrees about the Z axis. So

$$ {}^{W}_{V}\mathbf{R} = \mathbf{R}_Z(30) = \begin{pmatrix} 0.8660 & -0.5000 & 0 \\ 0.5000 & 0.8660 & 0 \\ 0 & 0 & 1 \end{pmatrix}. $$

Note that a good check for whether the rotation matrix is correct is to look at the columns of the rotation matrix. The columns are the unit vectors of the vehicle coordinate system, expressed in the world coordinate system. So for example, ${}^{W}\hat{\mathbf{x}}_V = \begin{pmatrix} 0.8660 \\ 0.5000 \\ 0 \end{pmatrix}$ is the first column of R, and it says that the x axis of the vehicle points mostly in the +X direction of the world, somewhat in the +Y direction of the world, and not at all in the Z direction. This is true by looking at the diagram of the scene.

The center of the vehicle is located at (X,Y,Z) = (+4,-4,+1) in world coordinates, so ${}^{W}\mathbf{t}_{Vorg} = (4,-4,1)^T$. So

$$ {}^{W}_{V}\mathbf{H} = \begin{pmatrix} \mathbf{R}_Z(30) & {}^{W}\mathbf{t}_{Vorg} \\ 0\ \ 0\ \ 0 & 1 \end{pmatrix} $$

The transformation from camera to vehicle: The camera is aligned with the vehicle except for a rotation about the X axis. We rotate -90 degrees in order to flip the Y axis so that it points down instead of forward as in the vehicle frame. Then we need to rotate an additional -30 degrees to account for the camera tilting down. So

$$ {}^{V}_{C}\mathbf{H} = \begin{pmatrix} \mathbf{R}_X(-120) & {}^{V}\mathbf{t}_{Corg} \\ 0\ \ 0\ \ 0 & 1 \end{pmatrix} $$

```
clear all
close all

% Want pose of the camera wrt world, H_c_w.

% Pose of vehicle to world is
tv_w = [4; -4; 1];
ax = 0;  ay = 0;  az = 30;
Rx = [1 0 0; 0 cosd(ax) -sind(ax); 0 sind(ax) cosd(ax) ];
Ry = [cosd(ay) 0 sind(ay); 0 1 0; -sind(ay) 0 cosd(ay)  ];
Rz = [cosd(az) -sind(az) 0; sind(az) cosd(az) 0; 0 0 1 ];
R_v_w = Rz*Ry*Rx;
H_v_w = [ R_v_w  tv_w;
          0 0 0 1 ];

% Pose of camera to vehicle is
tc_v = [0; 1; 2];
ax = -120;  ay = 0;  az = 0;
Rx = [1 0 0; 0 cosd(ax) -sind(ax); 0 sind(ax) cosd(ax) ];
Ry = [cosd(ay) 0 sind(ay); 0 1 0; -sind(ay) 0 cosd(ay)  ];
Rz = [cosd(az) -sind(az) 0; sind(az) cosd(az) 0; 0 0 1 ];
R_c_v = Rz*Ry*Rx;
H_c_v = [ R_c_v  tc_v;
          0 0 0 1 ];

% Want pose of world to camera.
% First form pose of camera to world:
H_c_w = H_v_w * H_c_v;

H_w_c = inv(H_c_w);

Mext = H_w_c(1:3, :);
```

```
% Intrinic camera matrix:
f = 300;
cx = 150;
cy = 100;
K = [ f   0   cx;
      0   f   cy;
      0   0   1 ];

% Define 3D points in the world.
% Each column is one point in homogeneous coords.
P_w = [
    0   1   1   0;
    0   0   1   1;
    0   0   0   0;
    1   1   1   1;
    ];

% Project points onto image
pimg = K*Mext*P_w;

pimg(1,:) = pimg(1,:) ./ pimg(3,:);
pimg(2,:) = pimg(2,:) ./ pimg(3,:);
pimg(3,:) = pimg(3,:) ./ pimg(3,:);

% Generate an image
I = zeros(200,300);
for i=1:size(pimg,2)
    r = round(pimg(2,i));
    c = round(pimg(1,i));
    I(r-1:r+1,c-1:c+1) = 1;
end
imshow(I,[]);
```