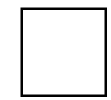
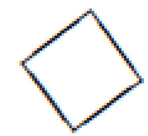
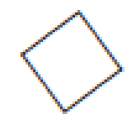
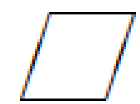
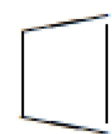
Ollscoil na Gaillimhe
University of Galway

Dr Waqar Shahid Qureshi
Waqarshahid.qureshi@universityofgalway.ie

# Revision from Tranformation

# Hierarchy of 3D Transformations

| Transformation | Matrix | # DoF | Preserves | Icon |
|---|---|---|---|---|
| translation | $\begin{bmatrix} I \mid t \end{bmatrix}_{3\times 4}$ | 3 | orientation |  |
| rigid (Euclidean) | $\begin{bmatrix} R \mid t \end{bmatrix}_{3\times 4}$ | 6 | lengths |  |
| similarity | $\begin{bmatrix} sR \mid t \end{bmatrix}_{3\times 4}$ | 7 | angles |  |
| affine | $\begin{bmatrix} A \end{bmatrix}_{3\times 4}$ | 12 | parallelism |  |
| projective | $\begin{bmatrix} \tilde{H} \end{bmatrix}_{4\times 4}$ | 15 | straight lines |  |

# 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
- IMP: A rotation matrix transforms its own rows onto the principal axes

- Any 3D rotation matrix can be described as rotation about an axis $\mathbf{n}$ by an angle $\theta$
- To rotate about given axis $\mathbf{n}$ by $\theta$:
  - Rotate axes onto a principal axis
    - by composing appropriate matrix through cross products
  - Rotate about principal axes and then undo the earlier transformation
  - OR use Rodriguez formula
- To compute $\mathbf{n}$ and $\theta$ from a 3D rotation matrix
  - $\mathbf{n}$ is the eigenvector corresponding to the real eigenvalue of 1
  - $\theta$ can be computed by the other 2 eigenvalues, which are $\cos\theta \pm i\sin\theta$
  - To disambiguate angle values, check for consistency with Rodriguez formula

# Camera Model

Part-1

# Pinhole camera

First described by Ibn Al-Haytham

ابو على، الحسن بن الحسن بن الهيثم

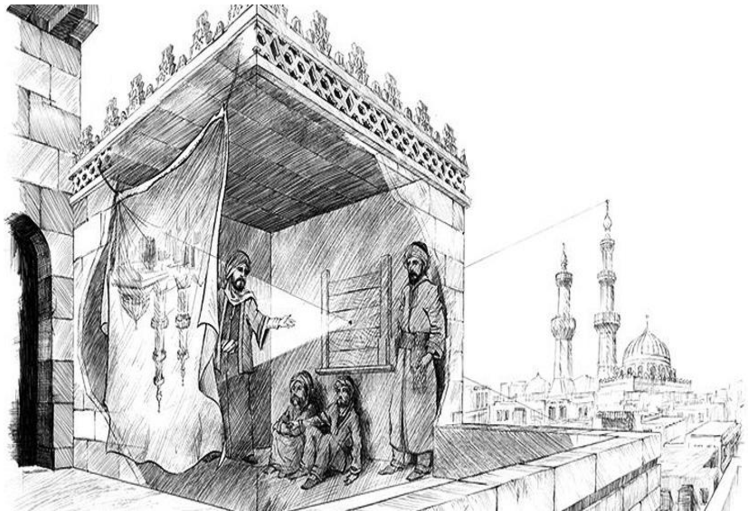
in his 7-volume work

كتاب المناظر

He termed it

بيت المظلم Al-beit Al-muzlim

which was later translated into Latin as “camera obscura”

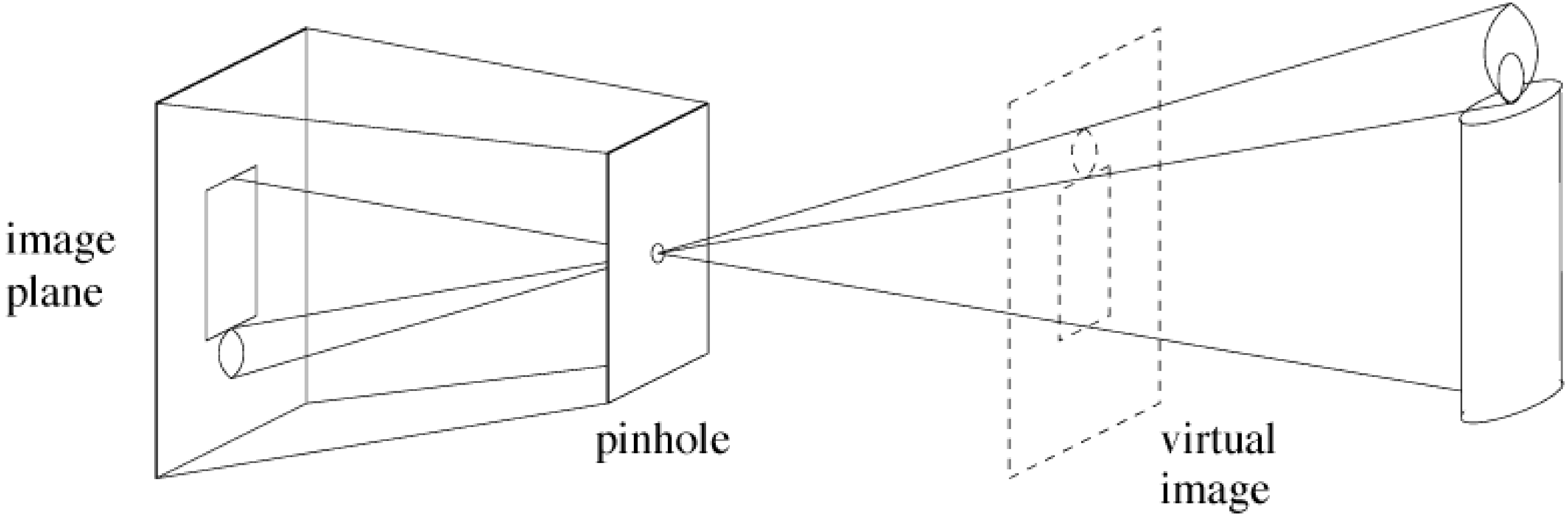



OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

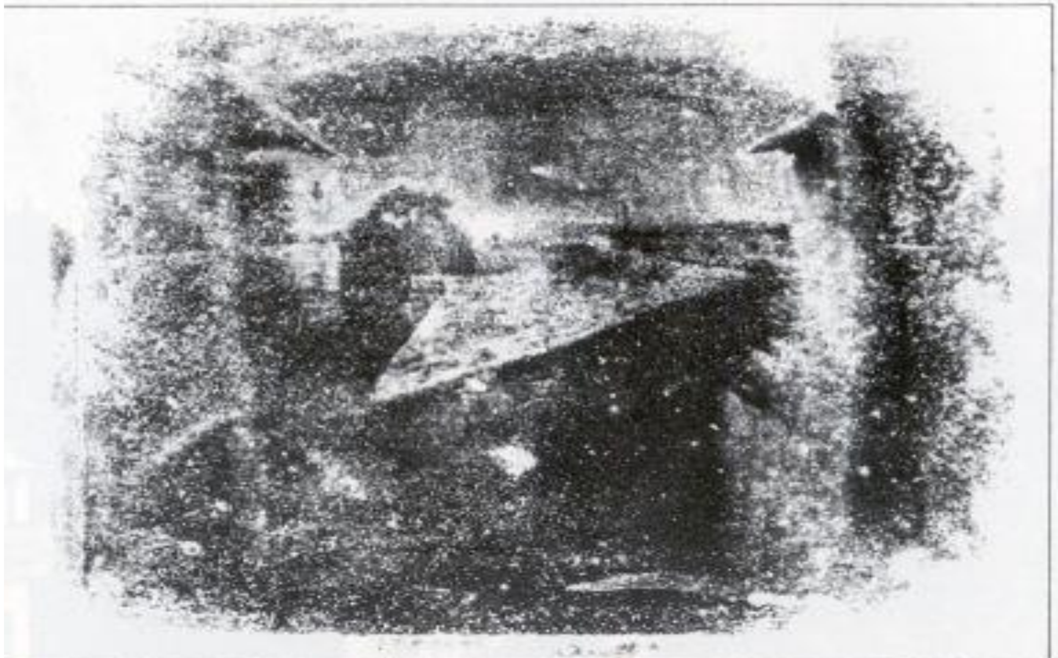
http://www.ibnalhaytham.com/discover/who-was-ibn-al-haytham/

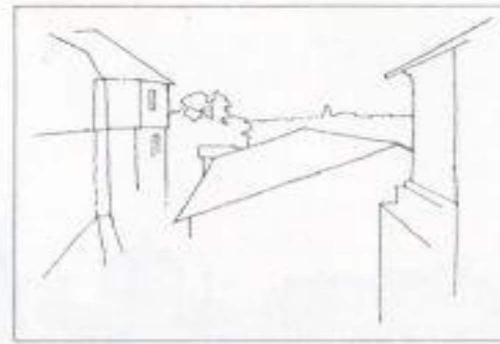
# Pinhole Camera

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



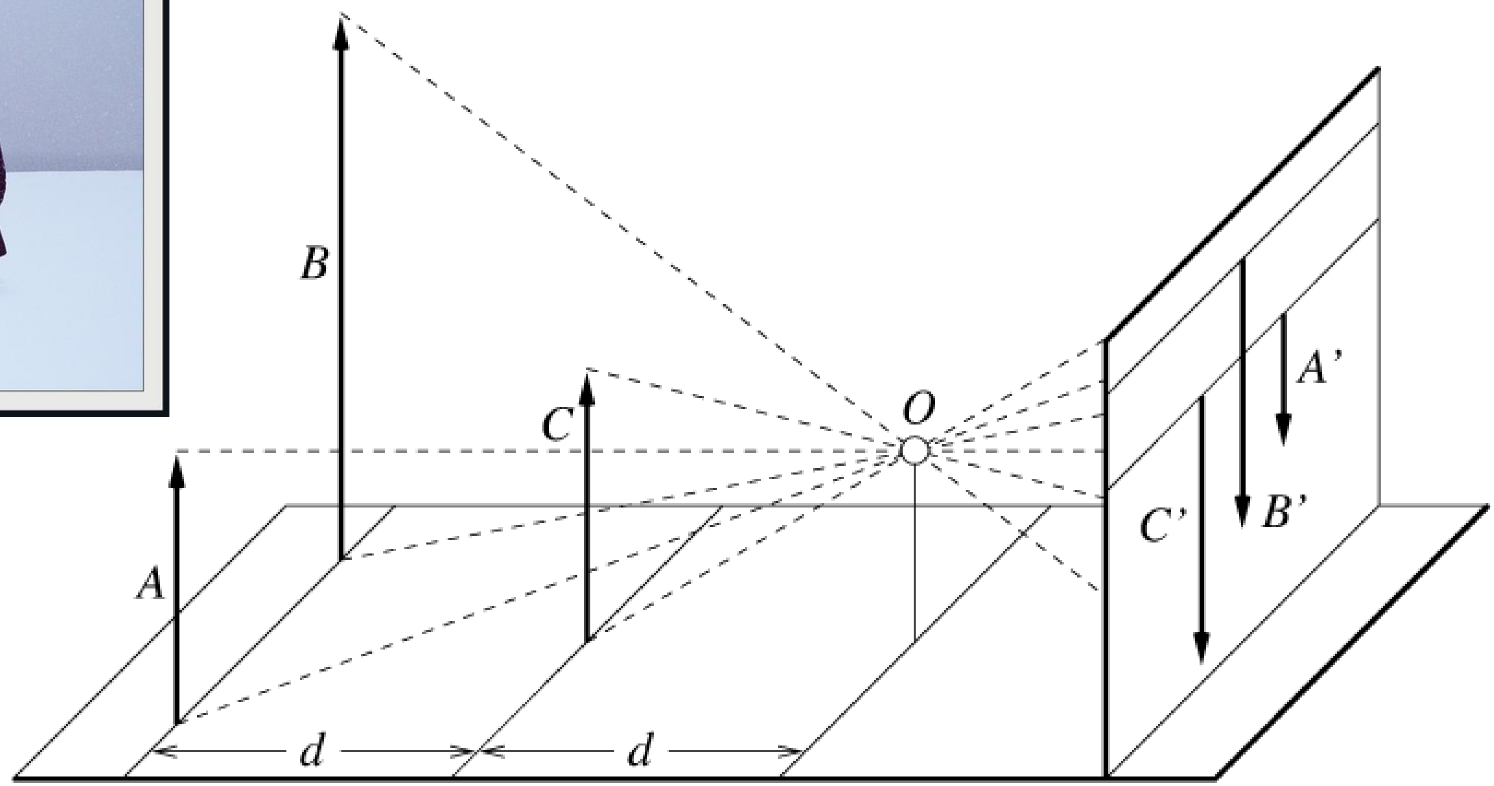
# The first photograph on record





**THE FIRST PHOTOGRAPH**
The world's first photograph was made in 1826 by Nicéphore Niepce from a window in his estate in France. For "film" Niepce used a sensitized pewter plate and he got a blurred image of the rooftops outlined above. This photograph is usually retouched to make it legible, but the version shown at left is what it really looks like.

# Pinhole Camera Properties: Distant objects are smaller

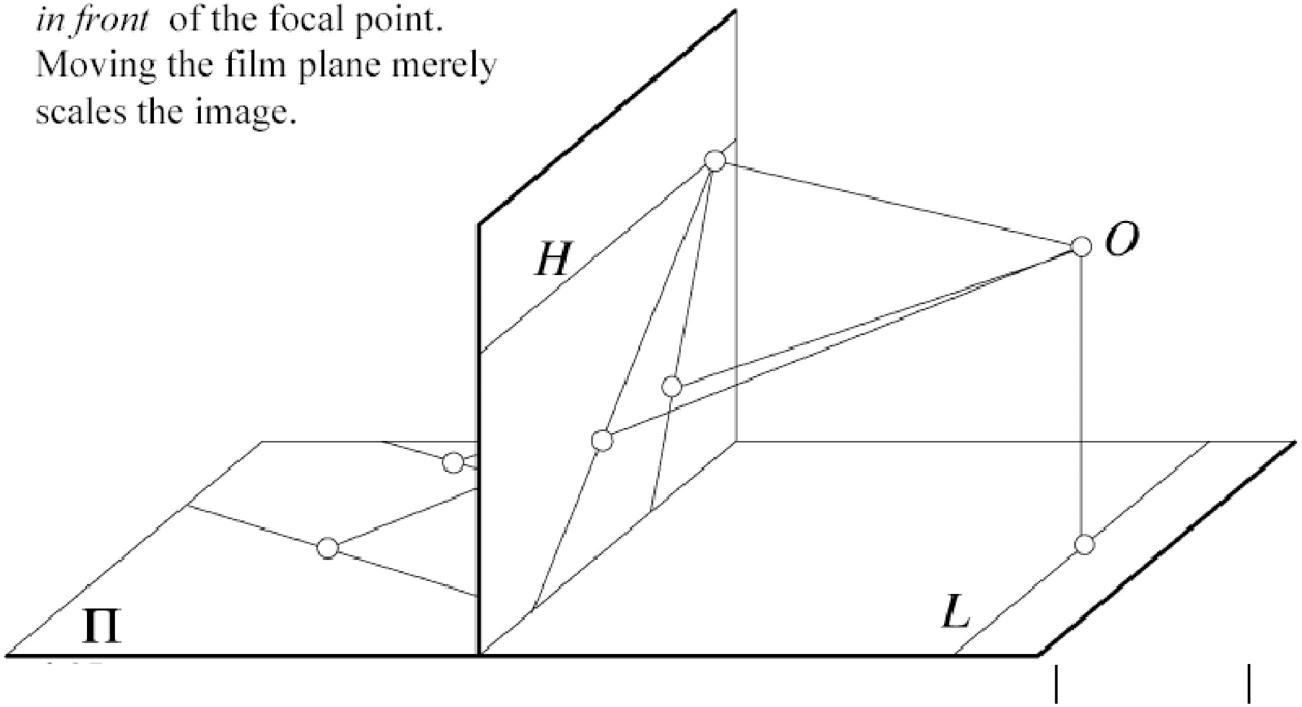



Slide Credit: Forsyth/Ponce http://www.cs.berkeley.edu/~daf/bookpages/slides.html
and Khurram Shafique, Object Video

# Pinhole Camera Properties

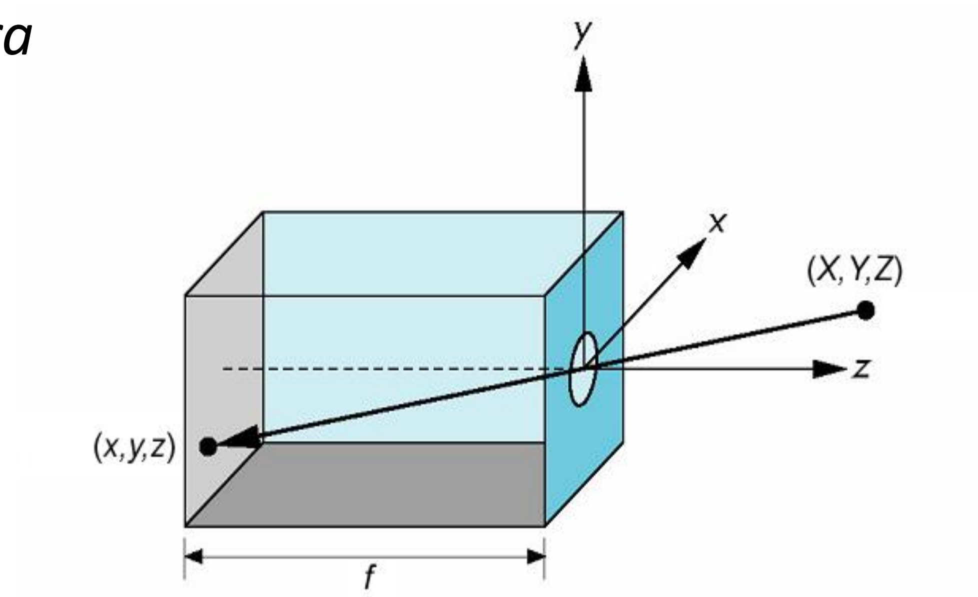
- Lines map to lines
- Polygons map to polygons
- Parallel lines meet

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



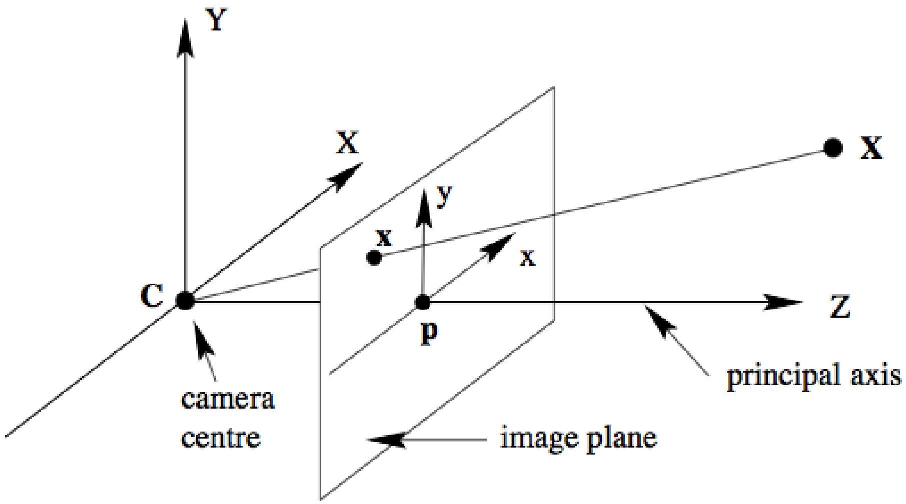
# Pinhole Camera

- Pinhole is considered the *center of projection*, *camera center* or *optical center*
- Let center of projection be at the origin of Euclidean space
- Plane $Z\ \mathbb{R}^3 f$ is the *imaging plane*, or *focal plane*
- Line from camera center, perpendicular to imaging plane is the *principal axis* or *principal ray*



# Pinhole Camera in Canonical Configuration

- Camera center $\mathbf{C}$ is at Euclidean origin
- Principal axis aligned with $Z$-axis
- *Principal point* $\mathbf{p}$ is the point where principal axis cuts the imaging plane
- Imaging plane is often taken by convention to be in front of the camera



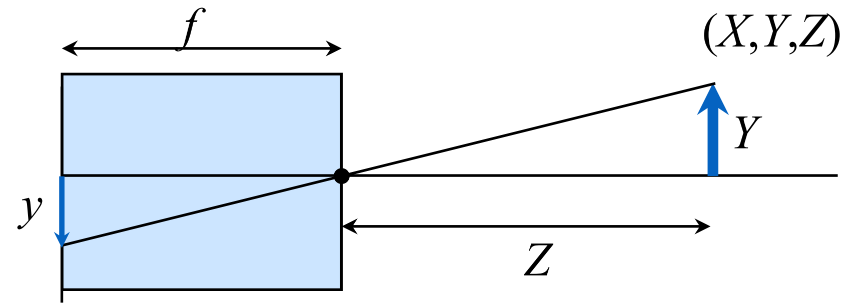
# Pinhole Camera in Canonical Configuration

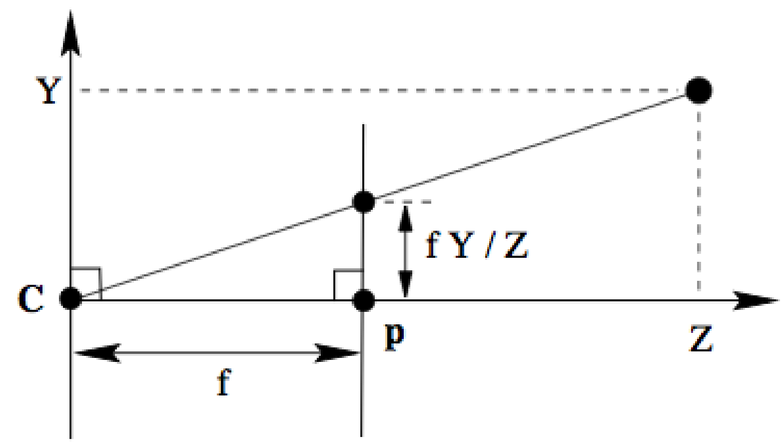
- Camera maps point $(X, Y, Z)^{\top}$ to $(x,y)^{\top}$
- By similar triangles

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

$$y = \frac{fY}{Z}$$

- Similarly $x = \dfrac{fX}{Z}$




- Thus, the camera maps $(X, Y, Z)^{\top} \mapsto \left(\frac{fX}{Z}, \frac{fY}{Z}\right)^{\top}$
- This mapping is from $\mathbb{R}^3 \mapsto \mathbb{R}^2$

# Central Projection

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

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

- Verify that since $hx = X, \quad hy = Y, \quad h = \frac{Z}{f}$

- Hence

$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

# Central Projection

- Camera in canonical view (centered at origin with optical axis aligned with world $Z$ axis, image axes aligned with $X$ and $Y$)

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

- Since any scaling of a homogeneous equation is valid, it is often written as

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

# Central Projection

- The camera can be more compactly written as

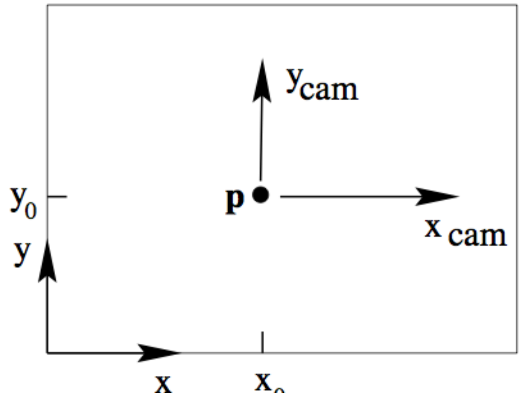
$$\mathbf{x} = \mathtt{P}\mathbf{X}$$

- where $\mathtt{P}$ is a 3x4 matrix that maps from $\mathbb{P}^3 \mapsto \mathbb{P}2$
- $\mathtt{P}$ may also be written as:

$$\mathtt{P} = \mathrm{diag}(f, f, 1)\,[\mathtt{I} \mid \mathbf{0}]$$

# Principal Point Offset

- The expression $\mathtt{P} = \mathrm{diag}(f, f, 1)\,[\mathtt{I} \mid \mathbf{0}]$ assumes that image origin is at the principal point.
- This may not be the case in general. For example:




- If the image coordinates of the principal point are $(p_x, p_y)$T, then the camera mapping will be

$$(X, Y, Z)^\top \mapsto \left(\frac{fX}{Z} + p_x, \frac{fY}{Z} + p_y\right)^\top$$

# Central Projection with Principal Point Offset

- In matrix form, this mapping becomes

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Sometimes, for convenience, it is also written as

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Or $$\mathbf{x} = \mathtt{K}\left[\mathtt{I} \mid \mathbf{0}\right]\mathbf{X}$$
- $\mathtt{K}$ is called the camera calibration matrix, which is a 3x3 matrix of internal camera parameters

# CCD Camera

- We have assumed same units for world & image coordinates
- In a CCD camera, image coordinates are measured in pixels
- Some CCD cameras also have non-square pixels
- We can convert to pixel units as

$$\mathtt{K} = \begin{bmatrix} m_x f & 0 & x_0 \\ 0 & m_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

  where $m_x$ and $m_y$ are scale factors of pixels per unit length, needed to convert to pixel dimension
  - $m_x$ = #of pixels in $x$ direction / size of CCD array in x direction
  - $m_y$ = #of pixels in $y$ direction / size of CCD array in y direction
- $(x_0, y_0)$ is principal point offset in pixel dimensions

$$x_0 = m_x p_x, \quad y_0 = m_y p_y$$

## Example

If you had a camera where:

- The CCD array is 2000 pixels wide and 1500 pixels tall.
- The physical size of the CCD sensor is 20 mm by 15 mm.
- The focal length $f = 50$ mm.

You would calculate:

- $m_x = \frac{2000 \text{ pixels}}{20 \text{ mm}} = 100 \text{ pixels/mm}$
- $m_y = \frac{1500 \text{ pixels}}{15 \text{ mm}} = 100 \text{ pixels/mm}$

Thus, your matrix $K$ would have the scale factors $m_x f$ and $m_y f$, and the principal point offset $(x_0, y_0)$.

# Pinhole camera in general view

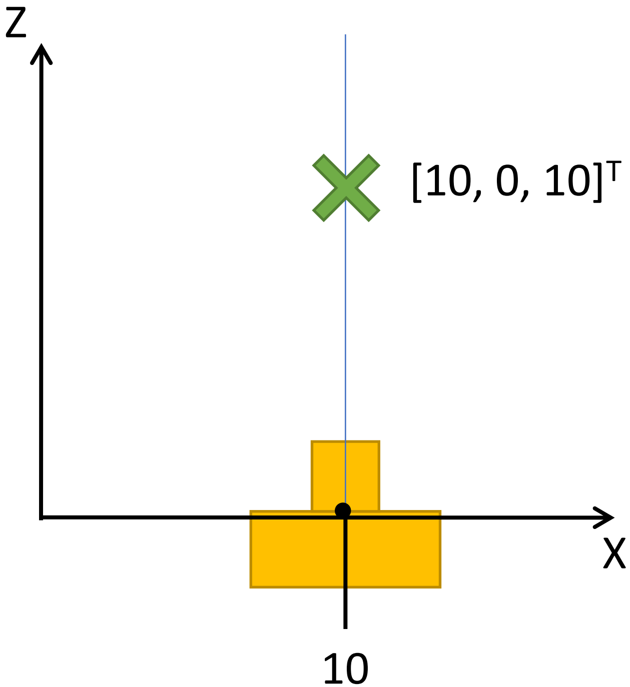
- This is for the case when the camera's optical axis is aligned with the world z-axis
- What if that is not the case?

# Pinhole camera in general view

- If the camera center is at coordinates $\mathbf{C}$ in the world, i.e. the camera is moved $\mathbf{C}$ from the origin, we should move the world point by $\mathbf{C}^{-1}$
- Then the perspective transform equation will be applicable
- Same holds for rotations

# Example

- Translation by 10 units to the right

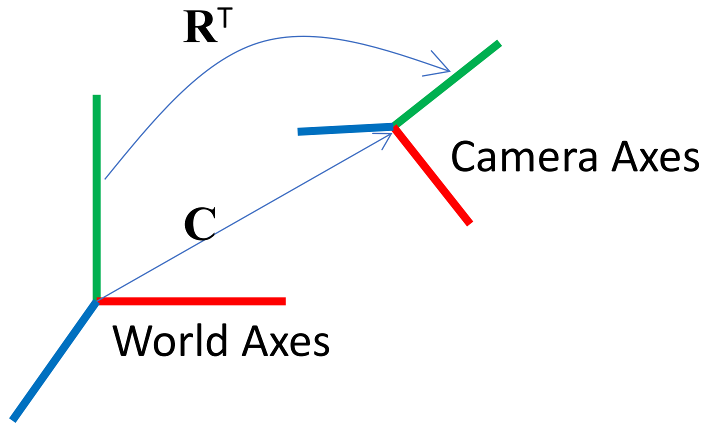



$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 0 \\ 10 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 10 \end{bmatrix}$$

# Pinhole camera in general view

- In general, the camera center is at a rotation of $\mathrm{R}^{\mathsf{T}}$, followed by a translation of $\mathrm{C}$ from the world origin
- Then




$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left( \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -C_x \\ 0 & 1 & 0 & -C_y \\ 0 & 0 & 1 & -C_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \right)$$

# Pinhole camera in general view

Canonical View
$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

General View
$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathrm{K}\left[\mathrm{R} \mid \mathrm{T}\right]\mathbf{X}$$

$$\mathbf{x} = \mathrm{KR}\left[\mathrm{I}_{3\times 3} \mid -\tilde{\mathbf{C}}\right]\mathbf{X}$$

**x** – image point
**X** – world point
**K** – 3x3 matrix of internal camera parameters
[**R** | **T**] – 3x4 matrix of external camera parameters
  **R** – rotation needed to align camera to world axes
  **T** – Translation needed to bring camera to world origin
  **T** = **-RC** where **C** is the vector of camera center

# Camera Model Example

- 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 $\phi$ along X, $\theta$ along Z, then translated by $(x_0, y_0, z_0)^T$
- This is the scenario in the figure on right

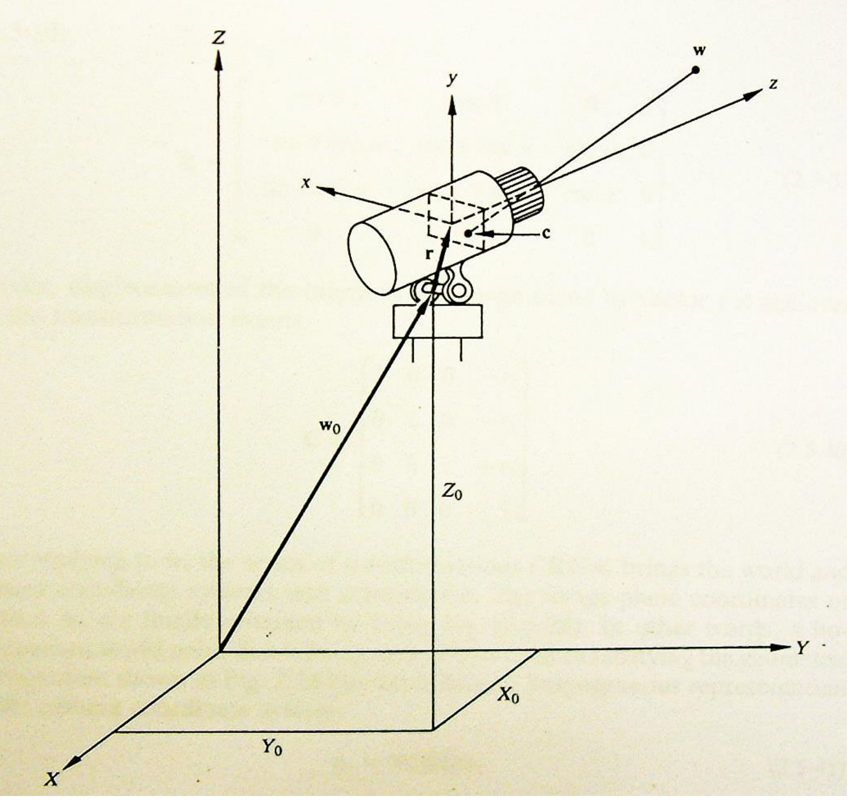
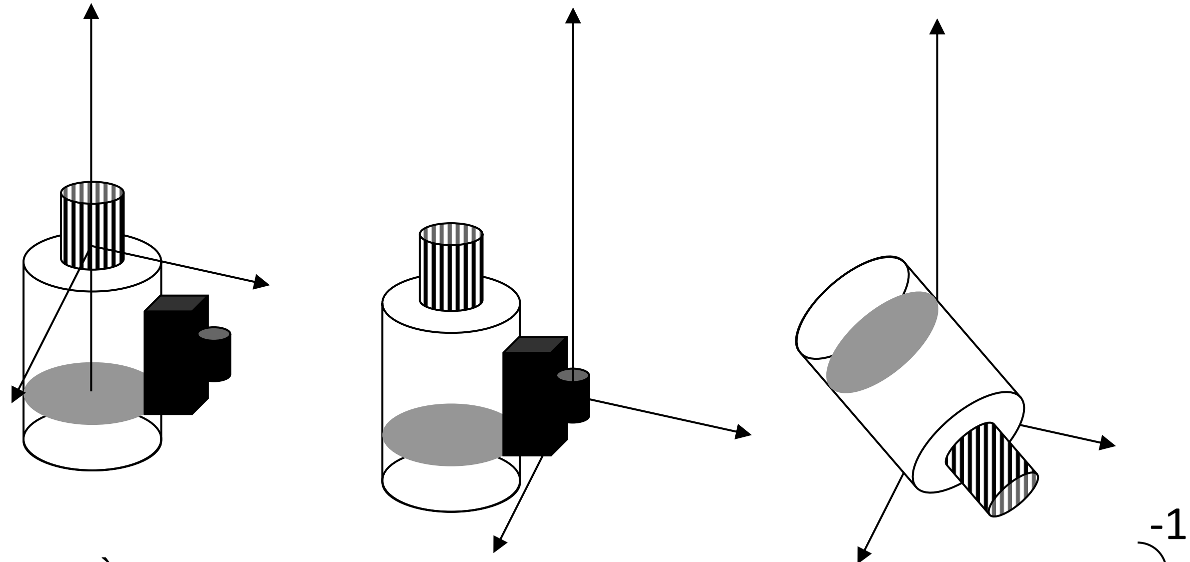



Figure Reference: Gonzales and Woods, "Digital Image Processing"

# Camera Model Example



$$\begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \left( \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & r_1 \\ 0 & 1 & 0 & r_2 \\ 0 & 0 & 1 & r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \right)^{-1}$$

$$\begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

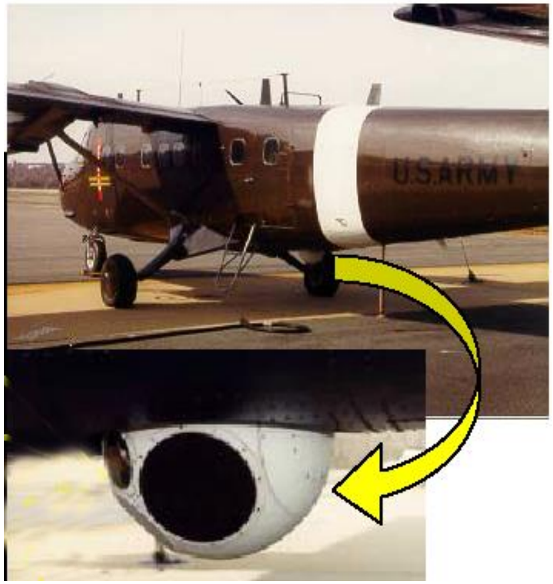
# Camera Model Example

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

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

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

# Aircraft Example




| | |
|---|---|
| OTTER | system_id |
| TV | sensor_type |
| 0001 | serial_number |
| 9.4000081526664030 0e+08 | image_time |
| 3.8131937464696122 00e+01 | vehicle_latitude |
| -7.7345231851938777 00e+01 | vehicle_longitude |
| 9.9496584099876588 00e+02 | vehicle_height |
| 9.9951711744410399 00e-01 | vehicle_pitch |
| 1.7016264181132090 00e+00 | vehicle_roll |
| 1.2070105517530294 00e+02 | vehicle_heading |
| 1.6589687329909748 00e-02 | camera_focal_length |
| -5.3613143895572591 00e+01 | camera_elevation |
| -7.2329694335467050 00e+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 ;

$$\mathbf{P} = \begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\Delta T_x \\ 0 & 1 & 0 & -\Delta T_y \\ 0 & 0 & 1 & -\Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
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;
```

# Pinhole camera in general view

Canonical View

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Surveillance Camera Example (Small gimbal translation ignored)

$$\begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Aircraft Example

$$\begin{bmatrix} m_x f & 0 & x_0 & 0 \\ 0 & m_y f & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -\Delta T_x \\ 0 & 1 & 0 & -\Delta T_y \\ 0 & 0 & 1 & -\Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Perspective Transform for Canonical View

Rotation needed to align camera with world axes

Translation by Inverse of Camera Center

# Summary: Perspective Camera Model

- The perspective camera model can be written as

$$\mathbf{x} = \mathrm{K} \left[ \mathrm{R} \mid -\mathrm{R}\tilde{\mathbf{C}} \right] \mathbf{X}$$

Image point $\in \mathbb{P}^2$

3x3 matrix of internal camera parameters (intrinsic parameters)

3x4 matrix of external camera parameters (extrinsic parameters)

World point $\in \mathbb{P}^3$

$$\mathrm{K} = \begin{bmatrix} m_x f & 0 & x_0 \\ 0 & m_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_0 = m_x p_x, \quad y_0 = m_y p_y$$

# Special Case 1:
# Perspective Camera Looking at a Plane

- Consider the case of camera looking at a plane
- This scenario occurs frequently in imaging applications

# Special Case 1:
# Perspective Camera Looking at a Plane

- Without loss of generality, we can assume that the plane has equation $Z = 0$

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} m_x f & 0 & x_0 \\ 0 & m_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_1 & r_2 & r_3 & T_x \\ r_4 & r_5 & r_6 & T_y \\ r_7 & r_8 & r_9 & T_z \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

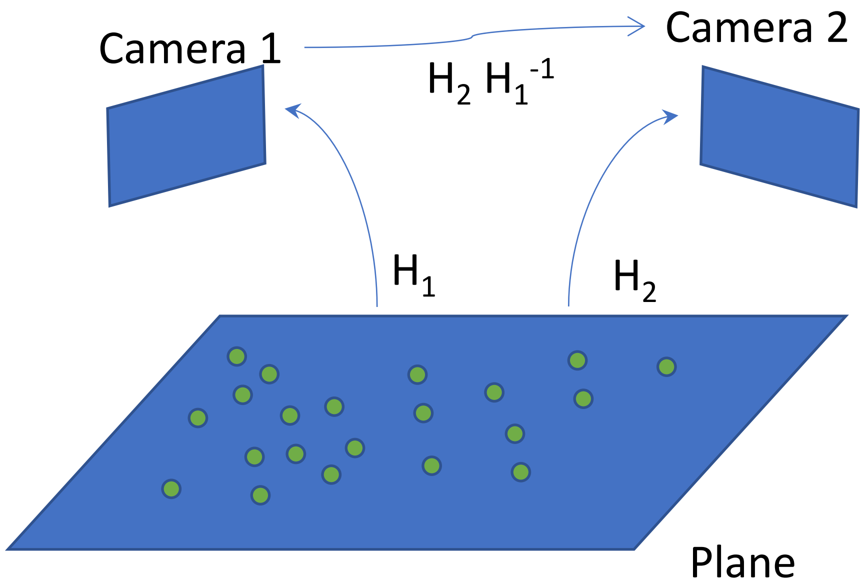
- The third column of $[\mathtt{R} \mid \mathbf{T}]$ does not matter in this case and can be dropped. So we can rewrite the system as:

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

- Conclusion: The 3D points lying on a plane are related to the image points by a homography.

# Special Case 1:
# Perspective Camera Looking at a Plane

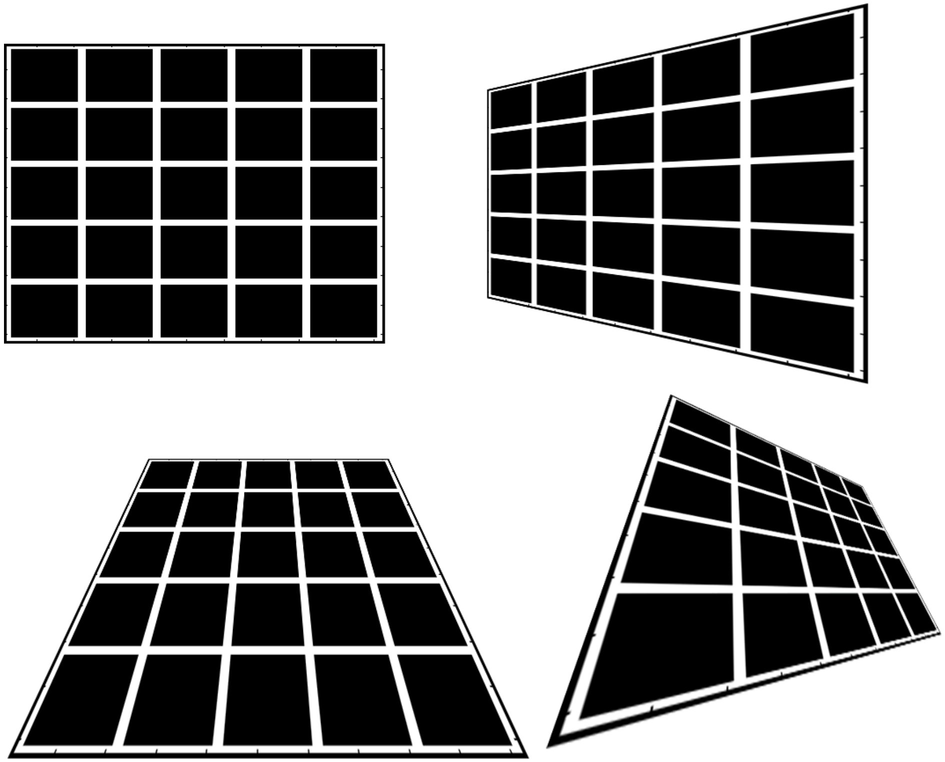
- Now consider relationship between *two* images of a plane




- Two images of a plane taken by a perspective camera are related by a homography

# Special Case 1:
# Perspective Camera Looking at a Plane

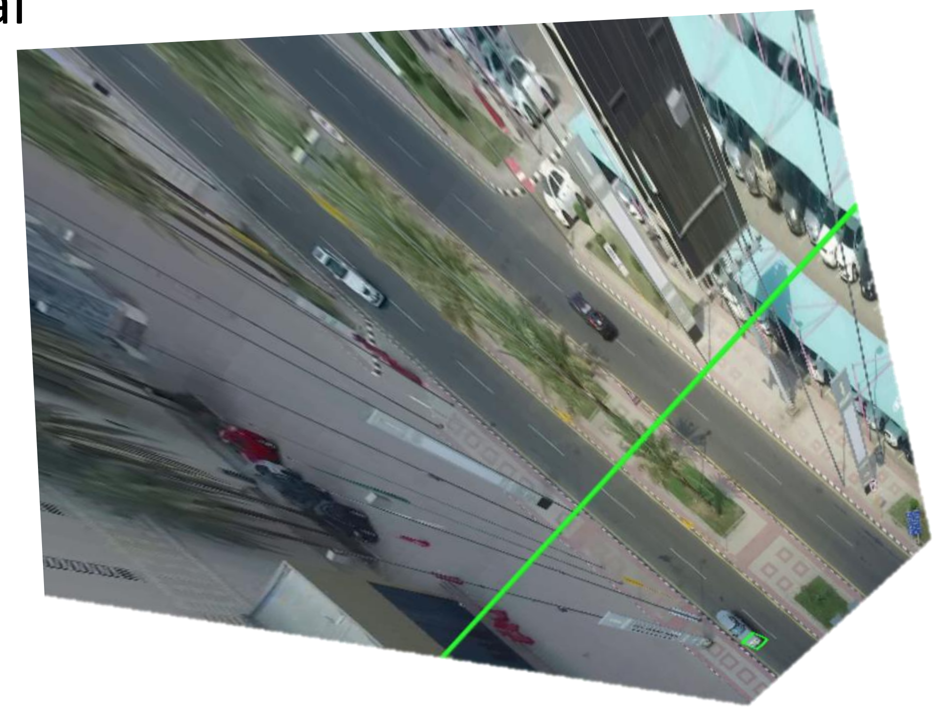
- It is no surprise that these projective transformations look like images of a plane taken from different angles



# Application: Rectification

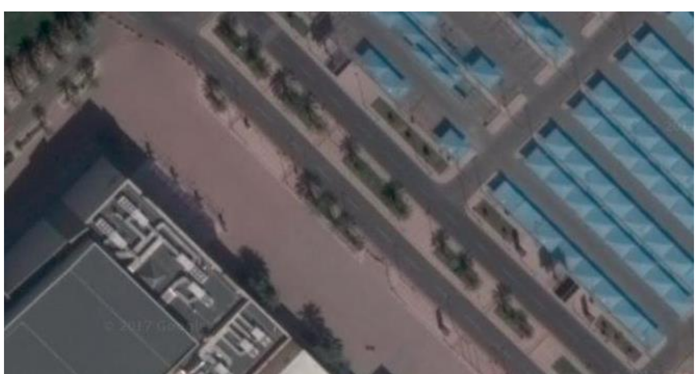
- A perspective image of a plane can be transformed into one in which the plane is fronto-parallel, i.e. the optical axis coincides with the plane normal


H



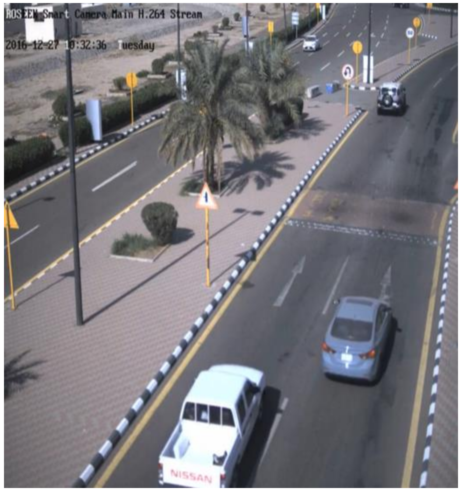
Rectified image

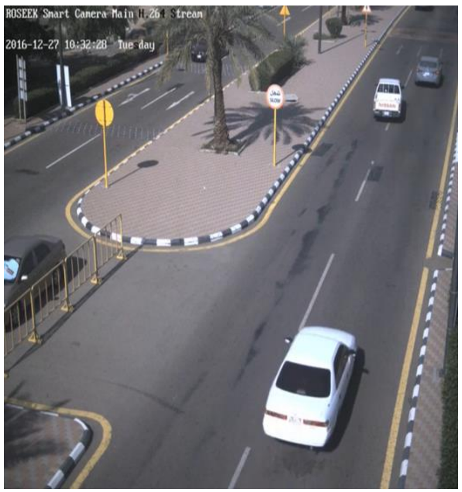
H

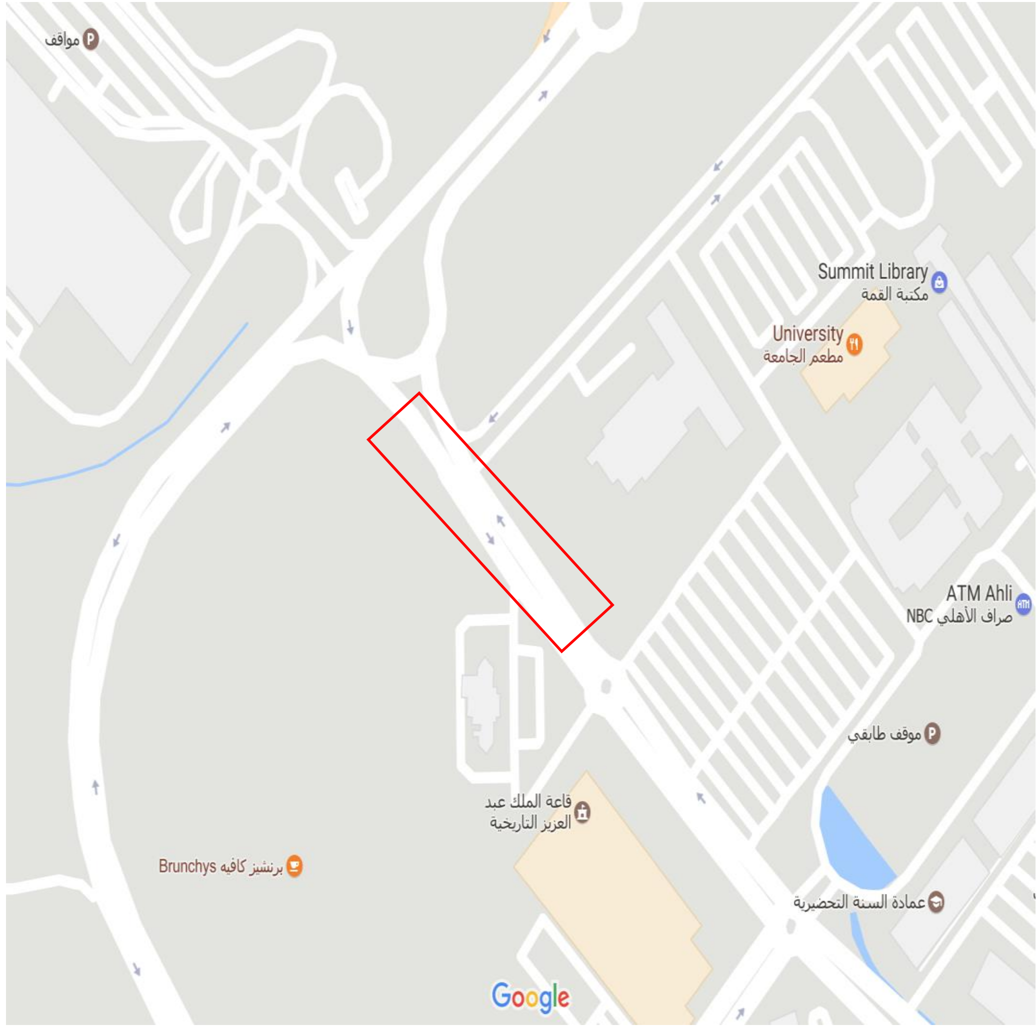


Satellite image

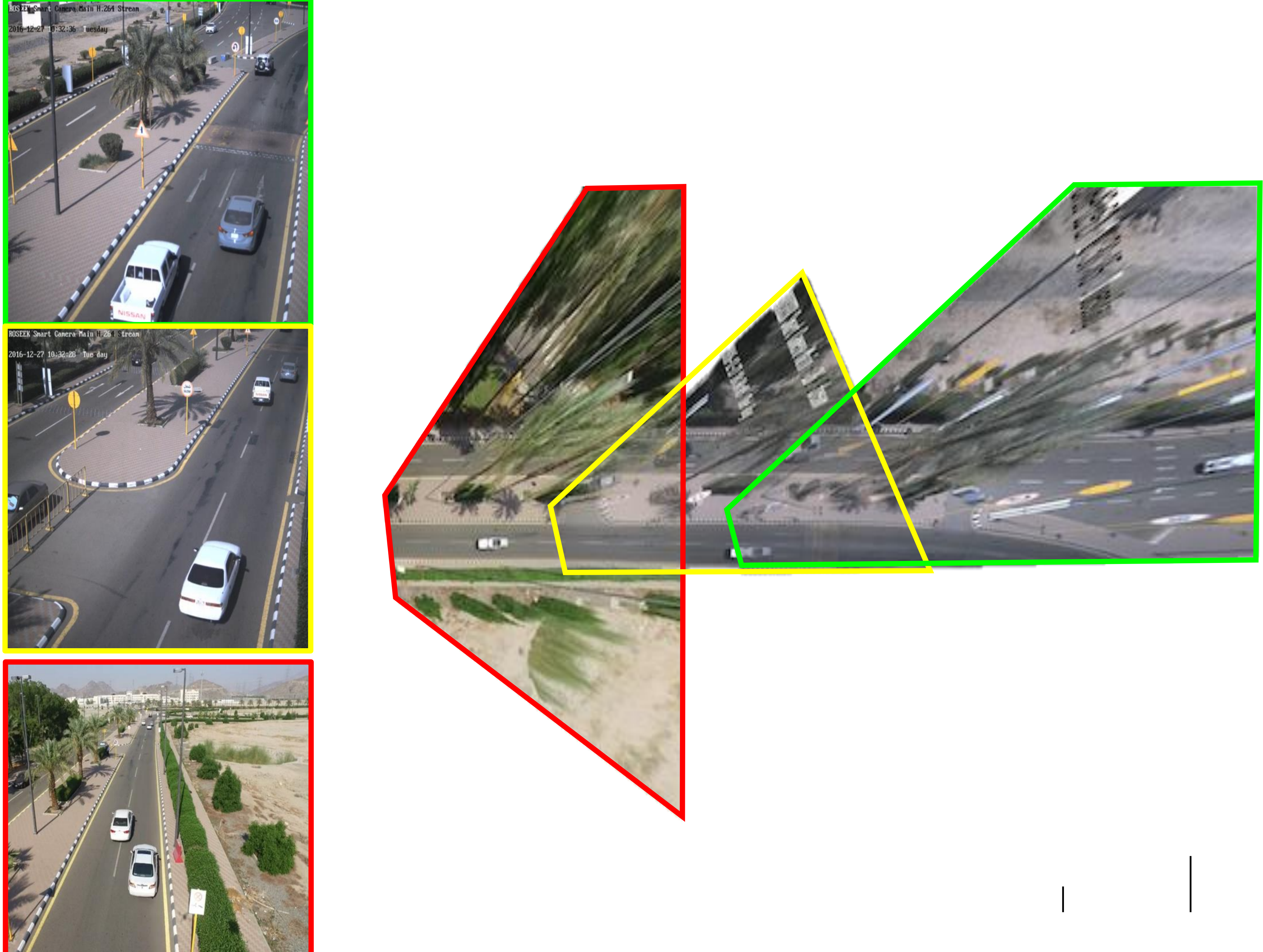
# Application: Rectification



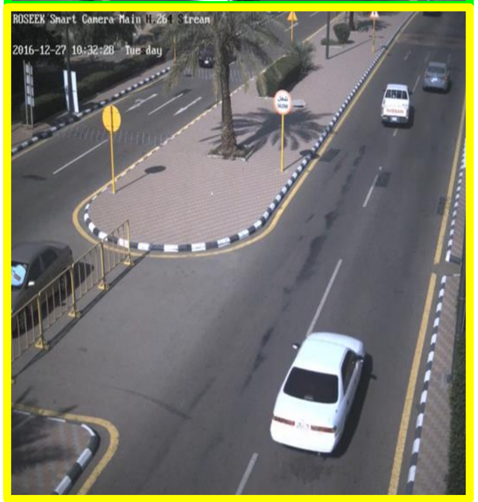





Road Segment, covered by three cameras



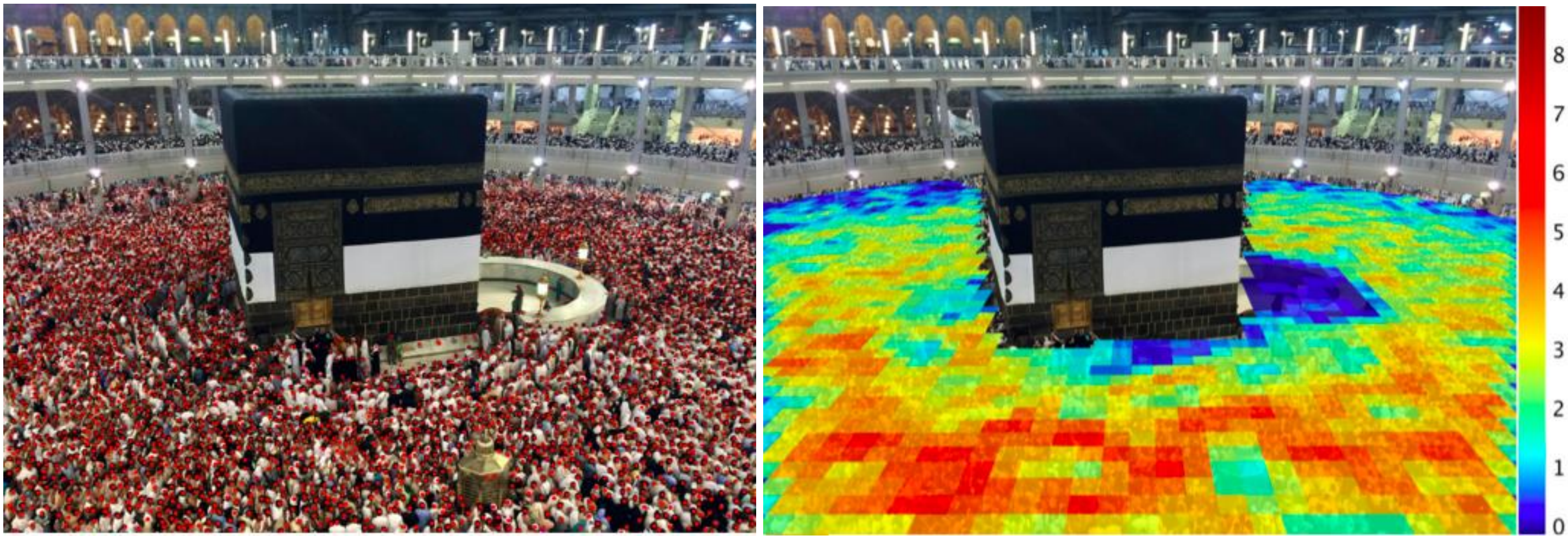
2016-12-27 10:32:36
2016-12-27 10:32:28
NISSAN



Merged view of the road

# Another Example of Rectification

- Measuring crowd density as persons per square meter



# Special Case 2:
# Rotation about Camera Center (Pure Rotation)

- Consider the case of camera that does not translates but only rotates about its optical center
- This scenario also occurs frequently in imaging applications

# Special Case 2:
# Rotation about Camera Center (Pure Rotation)

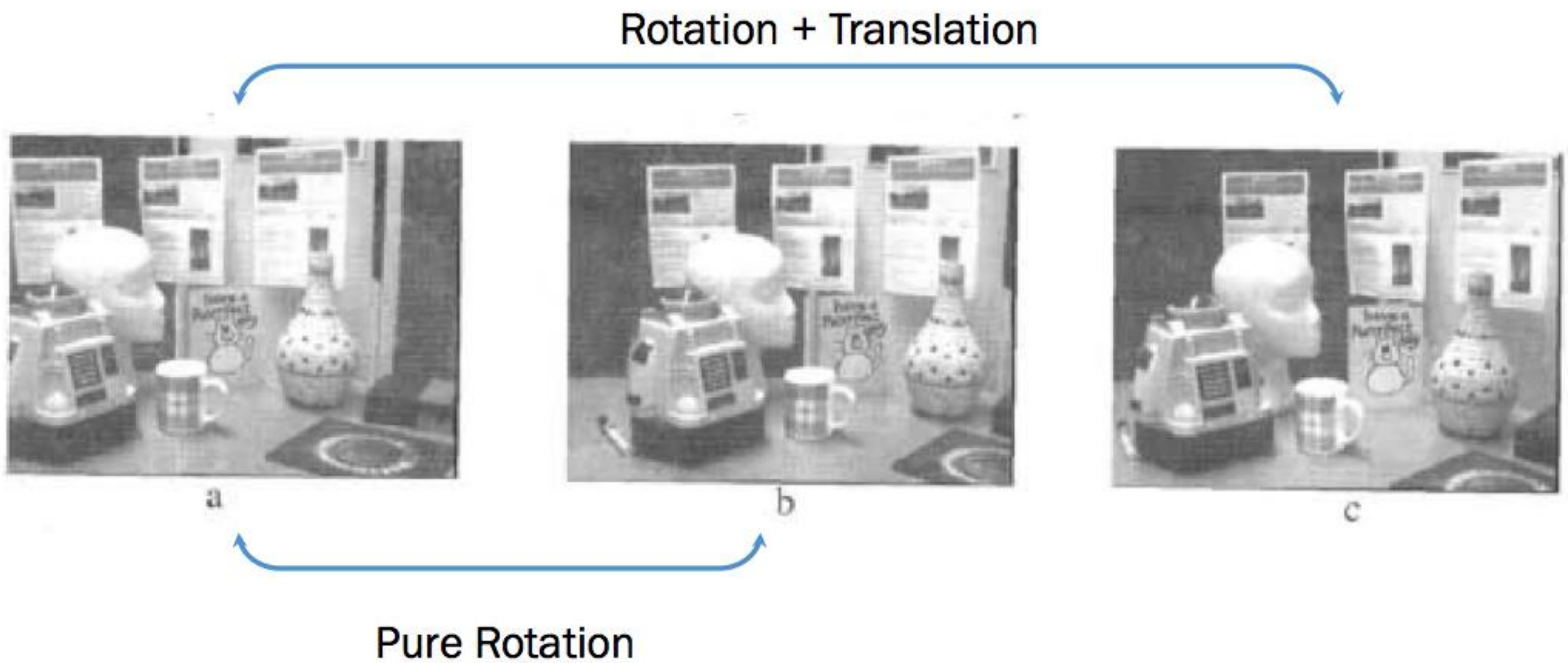
- $\mathbf{x}$ and $\mathbf{x}'$ are images of a point $\mathbf{X}$ before and after rotation of $\mathtt{R}$ the camera respectively.
- Then

$$\mathbf{x} = \mathtt{K}\,[\mathtt{I} \mid \mathbf{0}]\,\mathbf{X}$$

$$\begin{aligned}\mathbf{x}' &= \mathtt{K}\,[\mathtt{R} \mid \mathbf{0}]\,\mathbf{X} \\ &= \mathtt{KR}\,[\mathtt{I} \mid \mathbf{0}]\,\mathbf{X} \\ &= \mathtt{KRK}^{-1}\mathtt{K}\,[\mathtt{I} \mid \mathbf{0}]\,\mathbf{X} \\ &= \mathtt{KRK}^{-1}\mathbf{x}\end{aligned}$$

- Hence $\mathbf{x}' = \mathtt{KRK}^{-1}\mathbf{x}$ $\mathbf{x}' = \mathtt{H}\mathbf{x}$
- This type of homography is called a *conjugate rotation* homography

# Special Case 2:
# Rotation about Camera Center (Pure Rotation)




- The relative motion of objects in two images which are at different distances in the world is termed parallax
- A homography will not generate any parallax.
- Hence pure rotation of camera does not generate parallax

# Camera Calibration

# Recall: Perspective Camera Model

- The perspective camera model can be written as

$$\mathbf{x} = \mathtt{K}\left[\mathtt{R} \mid -\mathtt{R}\tilde{\mathbf{C}}\right]\mathbf{X}$$

Image point $\in \mathbb{P}^2$

3x3 matrix of internal camera parameters (intrinsic parameters)

3x4 matrix of external camera parameters (extrinsic parameters)

World point $\in \mathbb{P}^3$

$$\mathtt{K} = \begin{bmatrix} m_x f & 0 & x_0 \\ 0 & m_y f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$x_0 = m_x p_x, \quad y_0 = m_y p_y$$

# Camera Calibration

- In general, the camera model looks like:

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathrm{P}\mathbf{X}$$

- P is a 3x4 matrix of rank 3
- Calibration is the process of finding the parameters $[p_{11} \ldots p_{34}]$
- If x and X are known, then we can solve for the unknown parameters in P

# Camera Calibration

- Camera model

$$\begin{bmatrix} hx \\ hy \\ h \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- In inhomogeneous from

$$x = \frac{hx}{h} = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \qquad y = \frac{hy}{h} = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

- Multiplying both sides by denominator and rearranging

$$p_{11}X + p_{12}Y + p_{13}Z + p_{14} - p_{31}Xx - p_{32}Yx - p_{33}Zx - p_{34}x = 0$$

$$p_{21}X + p_{22}Y + p_{23}Z + p_{24} - p_{31}Xy - p_{32}Yy - p_{33}Zy - p_{34}y = 0$$

# Camera Calibration

$$p_{11}X + p_{12}Y + p_{13}Z + p_{14} - p_{31}Xx - p_{32}Yx - p_{33}Zx - p_{34}x = 0$$
$$p_{21}X + p_{22}Y + p_{23}Z + p_{24} - p_{31}Xy - p_{32}Yy - p_{33}Zy - p_{34}y = 0$$

- These equations have 12 unknowns
- Each correspondence between a world point and an image point yields two equations
- If 6 correspondences are known, we can solve for the unknowns

# Camera Calibration

$$p_{11}X + p_{12}Y + p_{13}Z + p_{14} - p_{31}Xx - p_{32}Yx - p_{33}Zx - p_{34}x = 0$$
$$p_{21}X + p_{22}Y + p_{23}Z + p_{24} - p_{31}Xy - p_{32}Yy - p_{33}Zy - p_{34}y = 0$$

- Separating out the known and the unknown terms

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

# Camera Calibration

- Given n correspondences...

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

$$\mathtt{A}\mathbf{p} = \mathbf{0}$$

# Camera Calibration

- This system Ap = 0 is a homogeneous system.
- A is rank deficient: rank(A) = 11 (at most)
- Solution?
- The null vector of A represents the p which is the solutions to the system Ap = 0
- How to find null space?
  1. null(A) in MATLAB, or
  2. Take SVD of A, as svd(C) = $USV^T$. The column of V corresponding to the singular value of zero represents the solution

     (in practice, you will have to take the smallest singular value)

# Camera Calibration: Summary

- Given a set of world points (in 3D coordinates) and their corresponding image points, we solve for the 3x4 camera matrix that relates them.
- This transforms into a problem of the form Ap = 0, which can be solved by finding the null vector of **A**.

- A more robust solution is through Direct Linear Transform, DLT (not covered in this class)

# Camera Calibration: Solving for Extrinsic and Intrinsic Parameters

- After finding **p**, we end up with a 3x4 camera matrix relating world points to image points

$$\mathbf{x} = \mathrm{P}\mathbf{X}$$
$$\mathrm{P} = \mathrm{KR}\left[\mathrm{I} \middle| -\tilde{\mathbf{C}}\right]$$

- How can I find camera rotation, translation and intrinsic parameters?
- Note that P has 12 terms and 11 degrees of freedom.

$$\mathrm{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

# Camera Calibration: Solving for Extrinsic and Intrinsic Parameters

- Solving for Camera Center C:
- Consider P times C

$$\mathbf{PC} = \mathrm{KR}\left[\mathrm{I}| - \tilde{\mathbf{C}}\right]\mathbf{C} = ?$$

$$\mathbf{PC} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

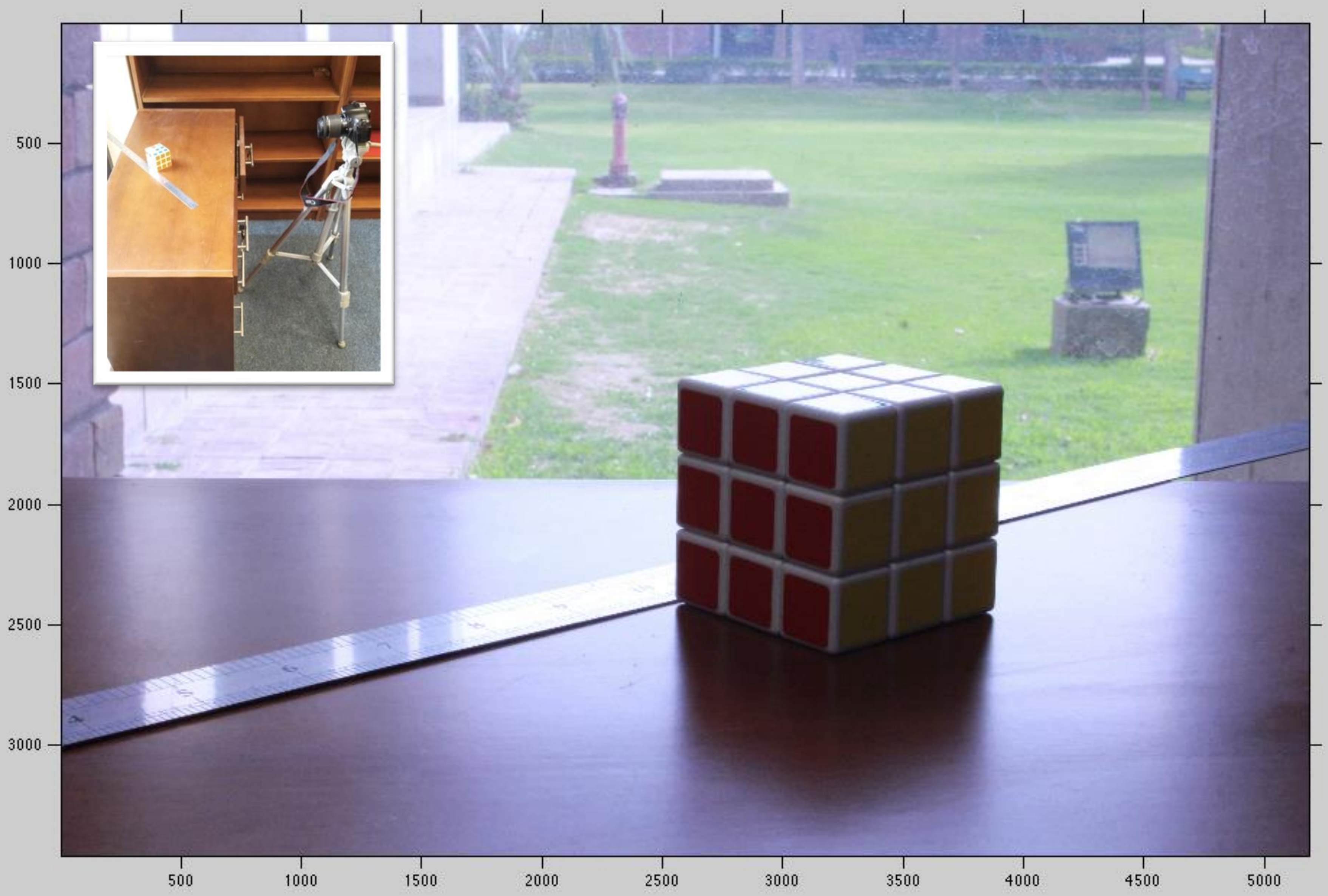
- Hence, camera center C is a null vector of P
- Note that PC = $(0, 0, 0)^T$ is undefined in image plane, which is exactly what we should expect, since image of camera center is undefined.

# Camera Calibration: Solving for Extrinsic and Intrinsic Parameters

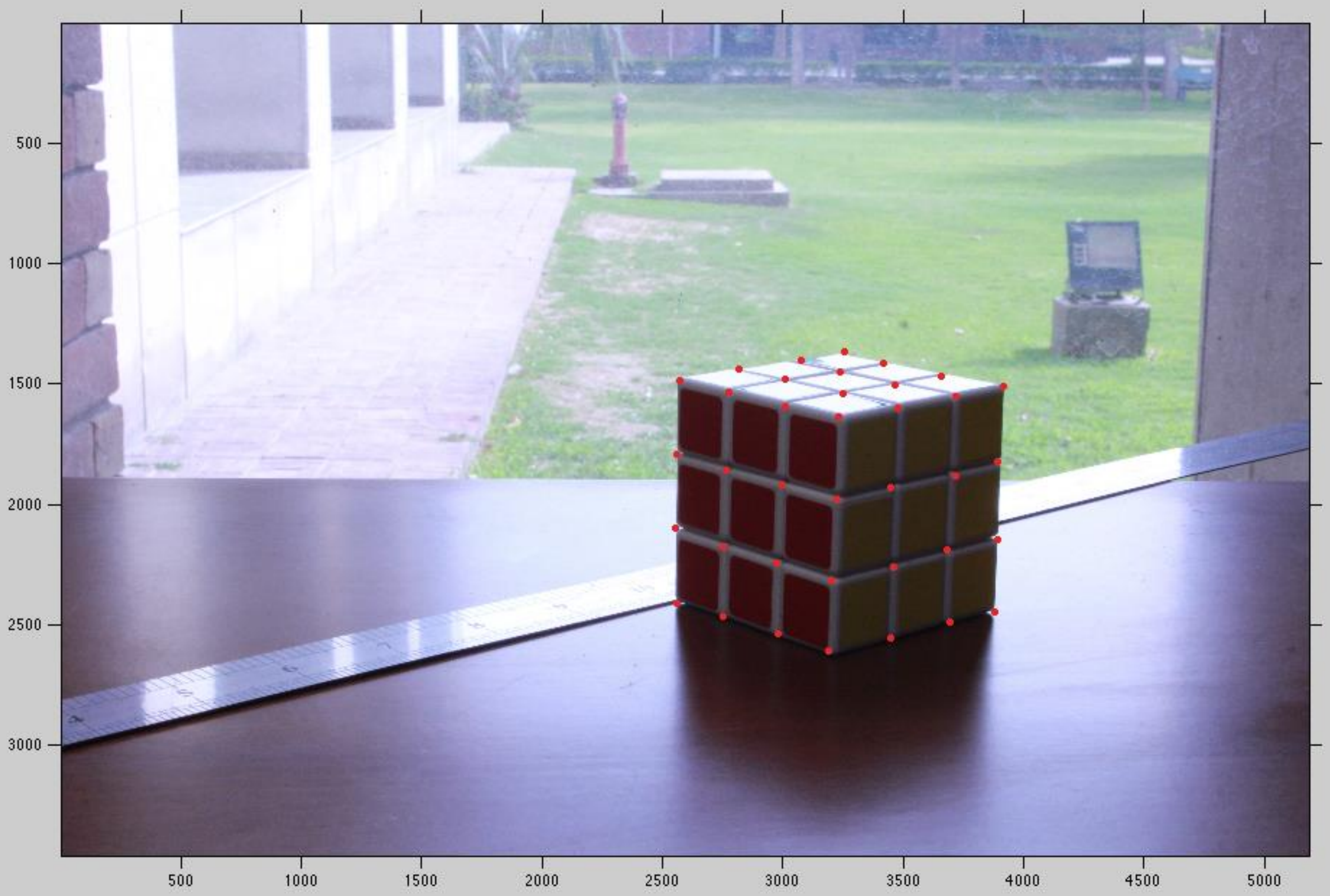
- Solving for **K** and **R**
- Note that **P** = [**KR | -KRC**]
- Hence first 3x3 block of P i.e. $M_{3x3}$ = KR
- K is an upper triangular matrix, R is an orthonormal matrix
- Solved through RQ decomposition
  RQ decomposition decomposes a matrix into an upper triangular matrix times an orthonormal matrix

# Camera Calibration Example

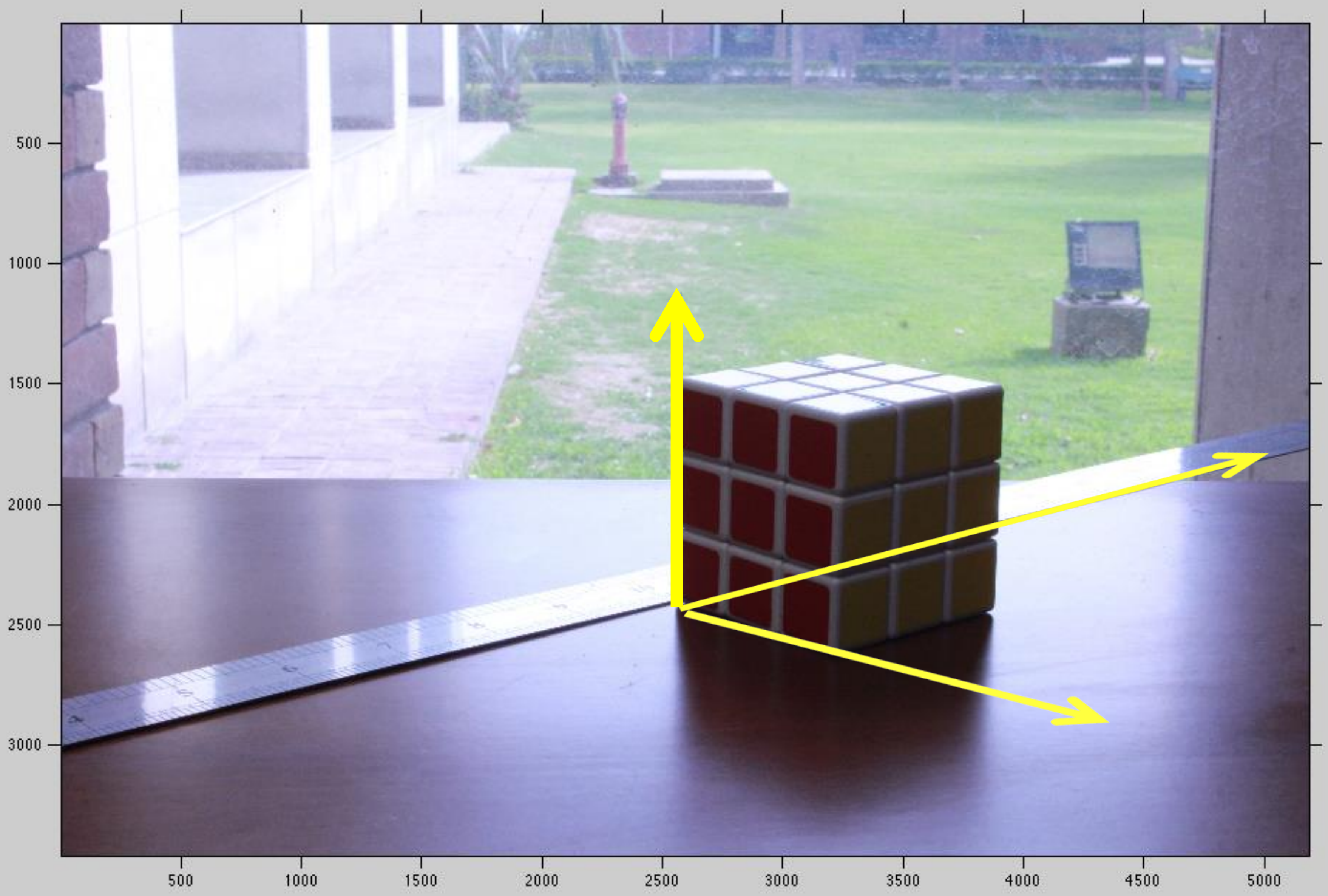
# Take Image of a Calibration Target



# Select Image Points



# Choose world coordinate system



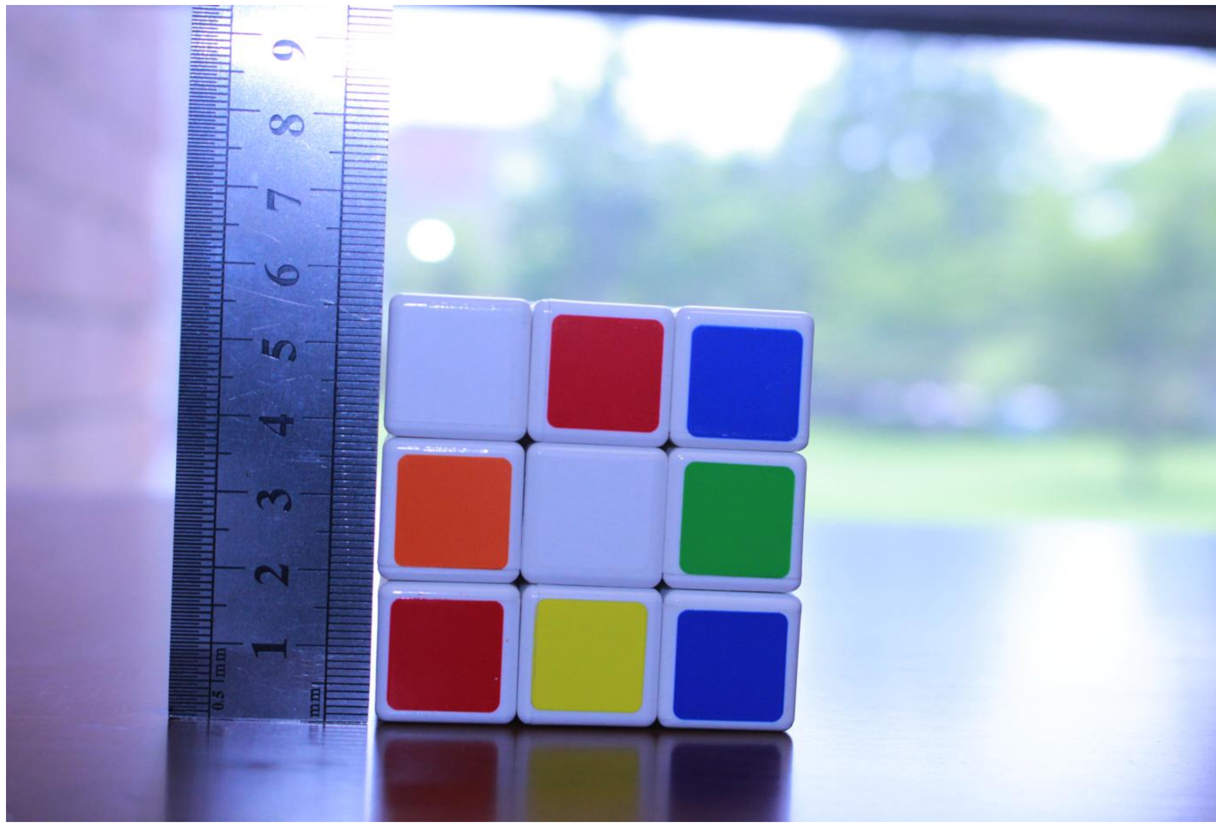
# Specify World Points

Vertices on the cube (in same order as image points)

| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 0 | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Cube height = 57mm
One box side = 19mm

Therefore, scale each coordinate by 19 mm

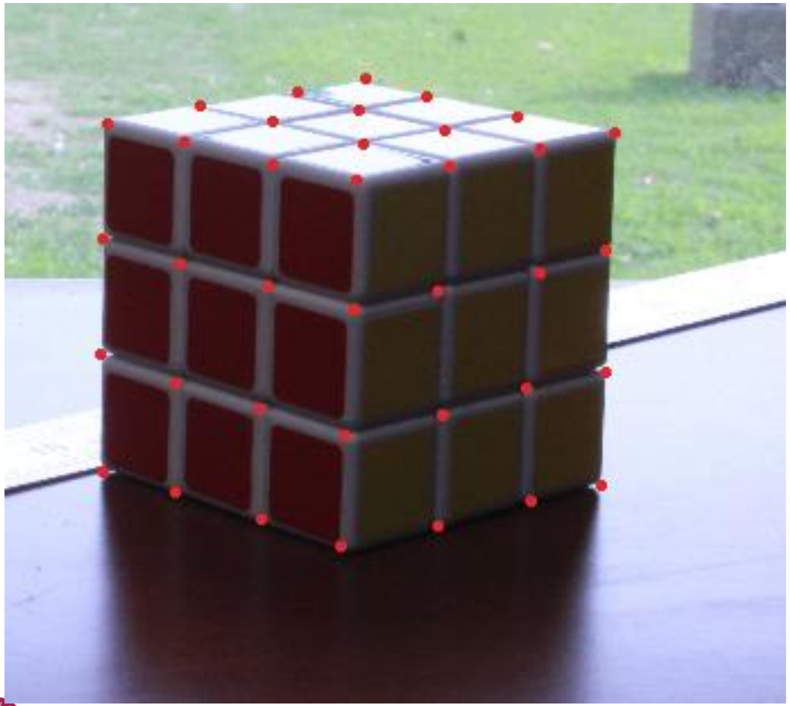


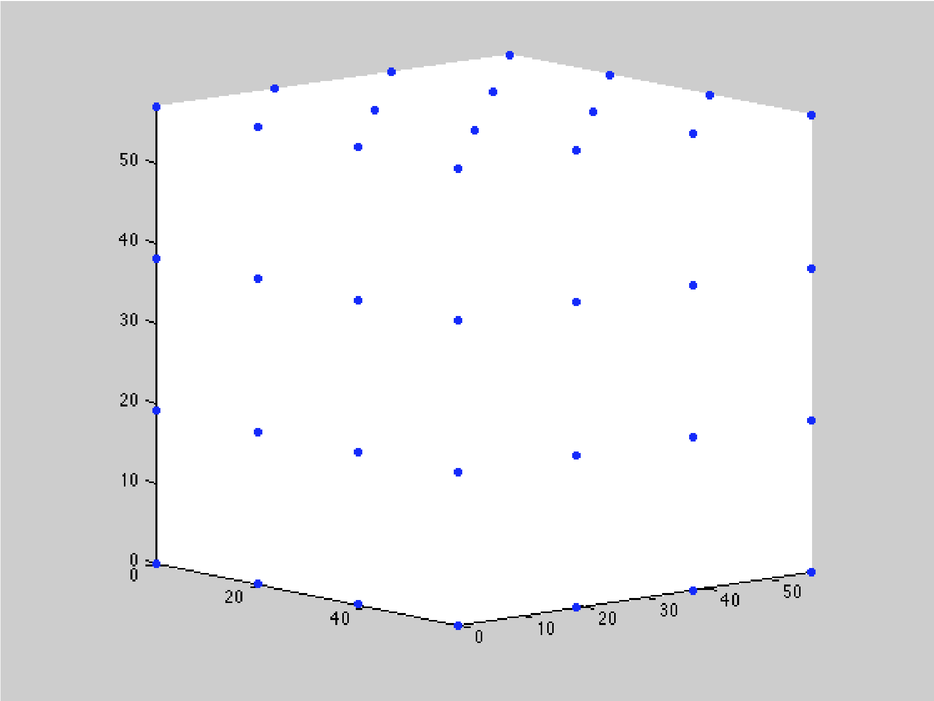
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 19 | 38 | 57 | 0 | 19 | 38 | 57 | 0 | 19 | 38 | 57 | 0 | 19 | 38 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 38 | 38 | 38 | 19 | 19 | 19 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 19 | 38 | 57 | 19 | 38 | 57 | 19 | 38 | 57 | 19 | 38 | 57 | 19 | 38 | 57 | 19 | 38 | 57 | 19 | 38 | 57 |
| 0 | 0 | 0 | 0 | 19 | 19 | 19 | 19 | 38 | 38 | 38 | 38 | 57 | 57 | 57 | 57 | 0 | 0 | 0 | 19 | 19 | 19 | 38 | 38 | 38 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 | 57 |

# Specify World Points

Vertices on the cube (in same order as image points)

```
0 19 38 57  0 19 38 57  0 19 38 57  0 19 38 57 57 57 57 57 57 57 57 57 57 57 57 57 38 38 38 19 19 19  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 19 38 57 19 38 57 19 38 57 19 38 57 19 38 57 19 38 57 19 38 57
0  0  0  0 19 19 19 19 38 38 38 38 57 57 57 57  0  0  0 19 19 19 38 38 38 57 57 57 57 57 57 57 57 57 57 57 57
```





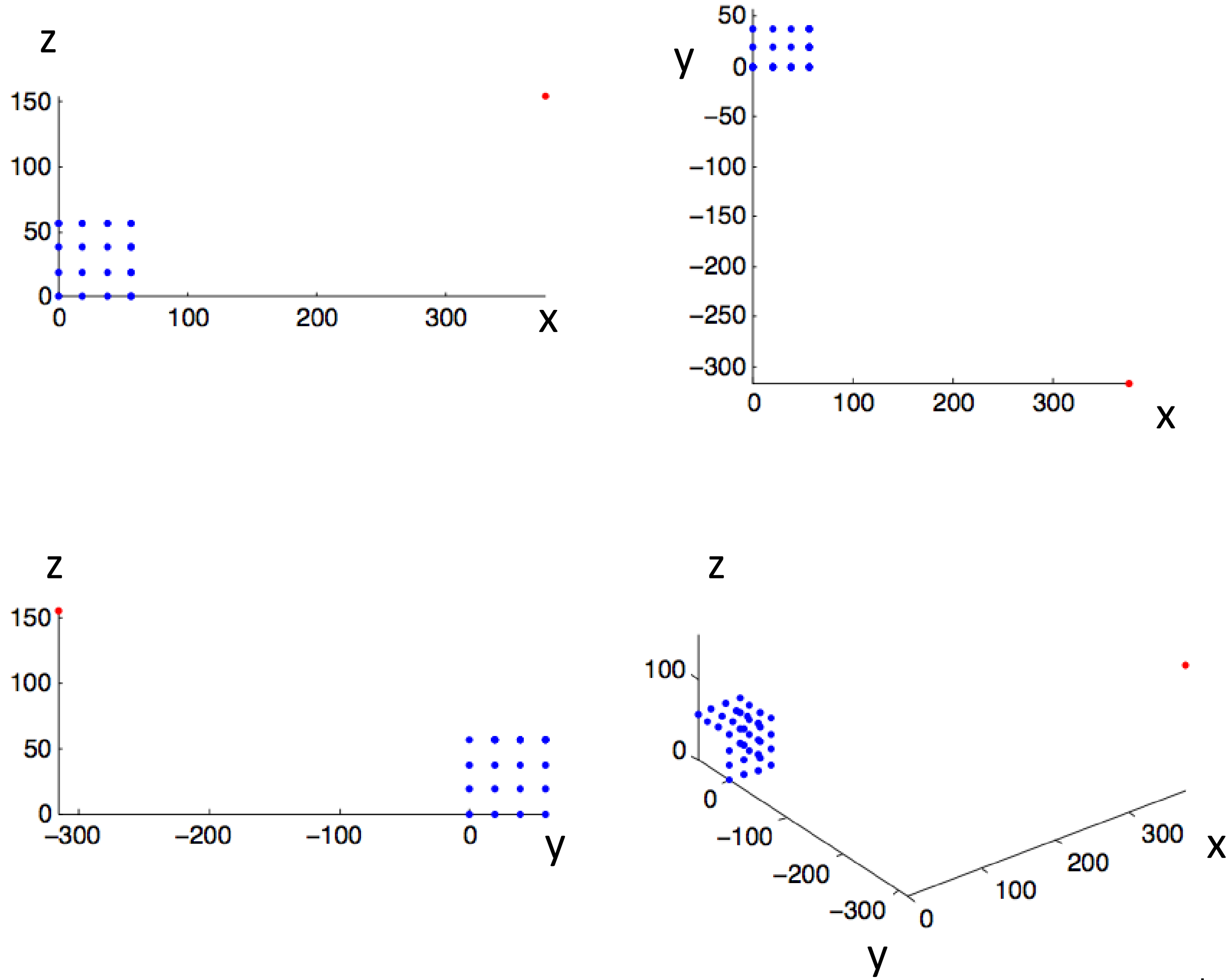
# Find the solution

- Set up matrix A and find its right null vector through SVD
  - warning: null(A) is unlikely to work! (WHY?)
- Reshape into a 3x4 matrix

```
P =
  -0.00010835   4.3034e-05    0.0047453      -0.68373
   -0.0019211   -0.0044849   0.00023615       -0.7297
   4.1144e-07  -3.5796e-07   1.1421e-07   -0.00028537
```

# Find camera center from **P**

$$C = \begin{bmatrix} 375.89 \\ -315.53 \\ 155.53 \\ 1 \end{bmatrix}$$



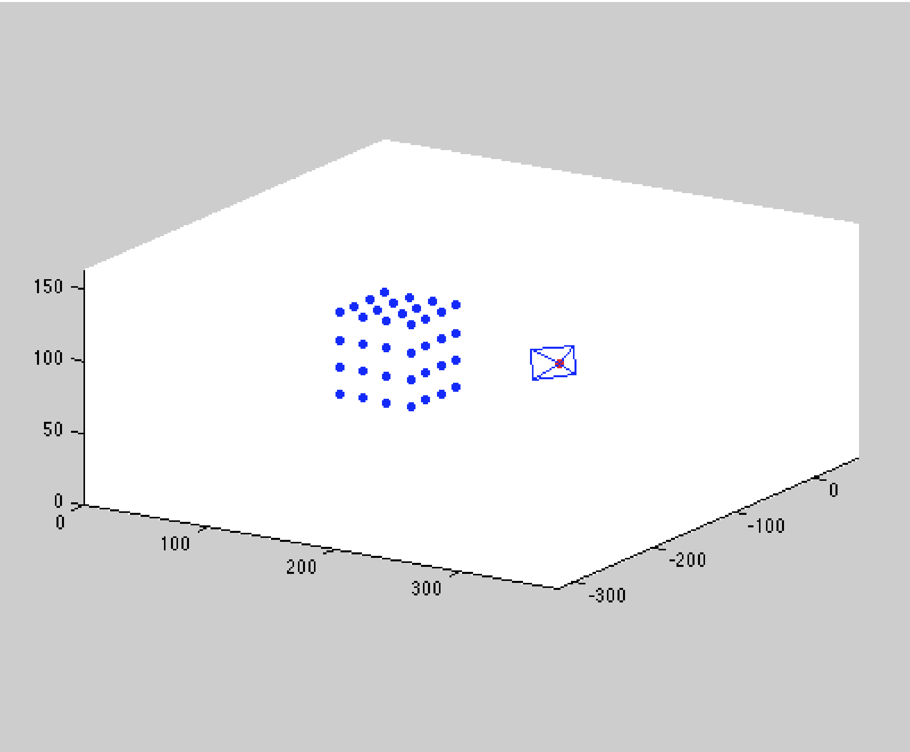
# Compute **K** and **R** by RQ Factorization

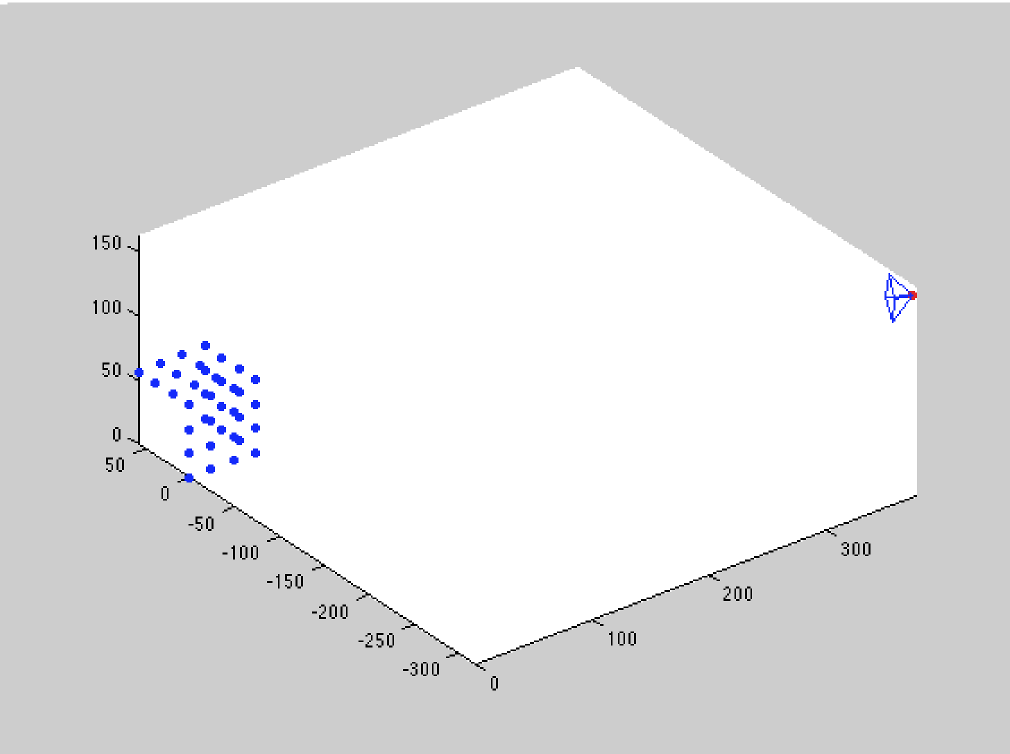
K =

| | | |
|---|---|---|
| -8376.2 | 66.191 | 1552.4 |
| 0 | -8336.6 | 2712 |
| 0 | | 1 |

R =

| | | |
|---|---|---|
| -0.16524 | 0.12231 | 0.97864 |
| -0.65379 | -0.75651 | -0.01584 |
| -0.73842 | 0.64244 | -0.20497 |



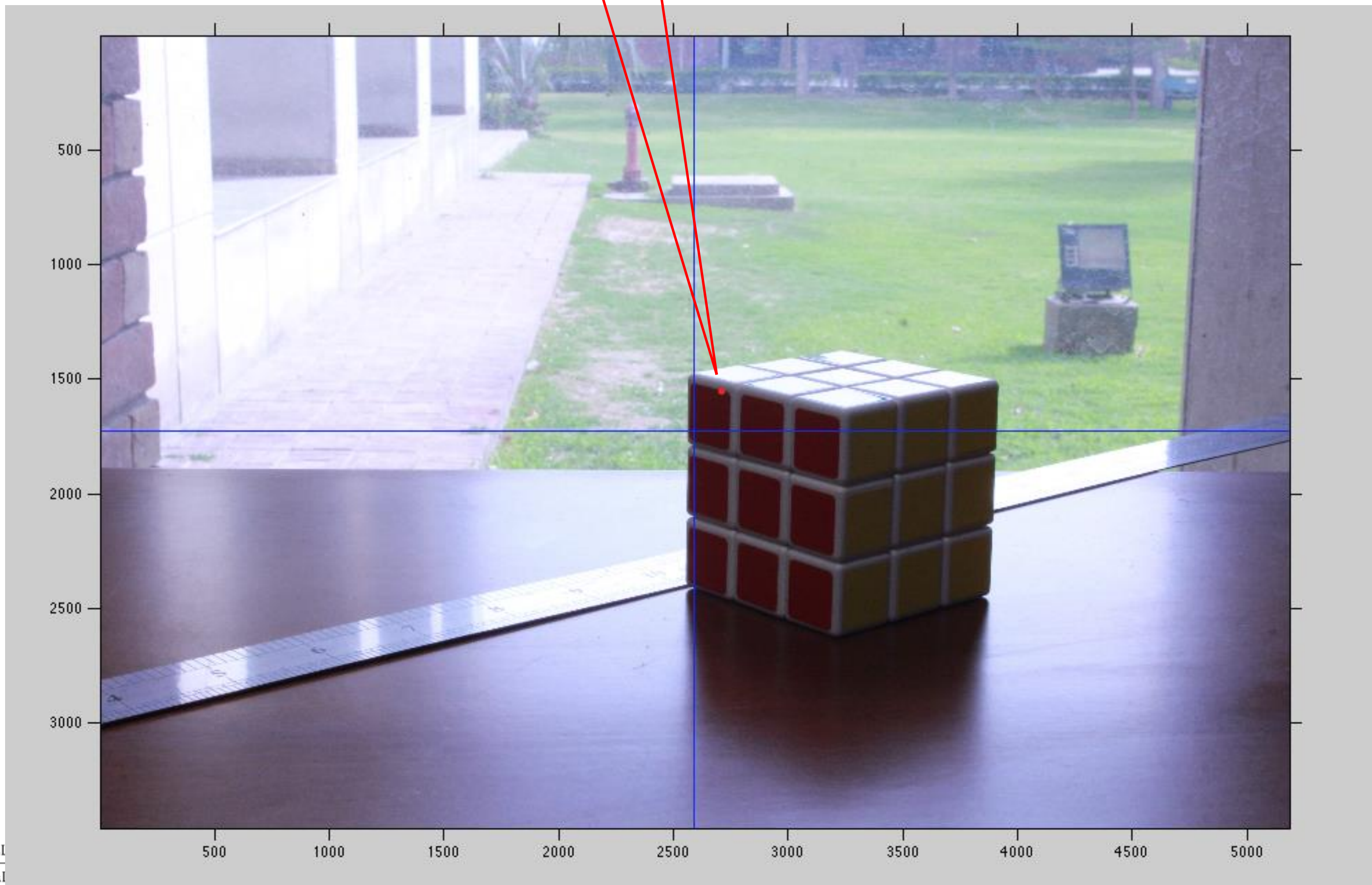
# Principal Point

K =

| | | |
|---|---|---|
| -8376.2 | 66.191 | 1552.4 |
| 0 | -8336.6 | 2712 |
| 0 | | 1 |

R =

| | | |
|---|---|---|
| -0.16524 | 0.12231 | 0.97864 |
| -0.65379 | -0.75651 | -0.01584 |
| -0.73842 | 0.64244 | -0.20497 |



# Focal Length

K =

| | | |
|---|---|---|
| -8376.2 | 66.191 | 1552.4 |
| 0 | -8336.6 | 2712 |
| 0 | | 1 |

R =

| | | |
|---|---|---|
| -0.16524 | 0.12231 | 0.97864 |
| -0.65379 | -0.75651 | -0.01584 |
| -0.73842 | 0.64244 | -0.20497 |

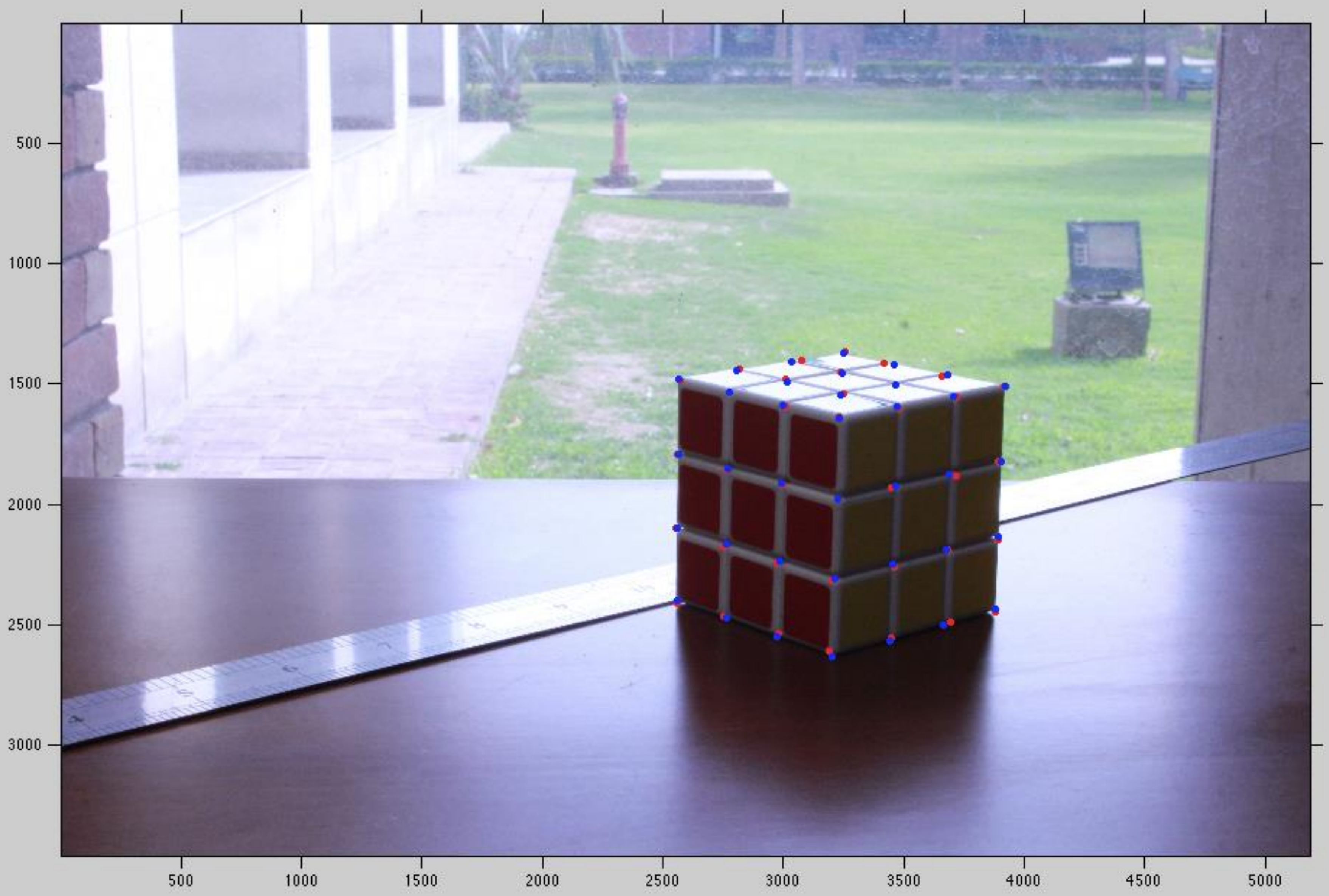
$m_x f = 8376.2$   $m_y f = 8336.6$

- Image was taken by Canon 600D
- Canon 600D uses APS-C format CMOS sensor*
- Size of APS-C sensor is 23.6mm x 15.7mm
- Image size is 5184 x 3456

- Hence
- $m_x$ = 3456/15.7 = 220.13 pix/mm
- $m_y$ = 5184/23.6 = 219.66 pix/mm
- f = [38.051, 37.952] mm
- Verification from EXIF data: 37mm
- Angle between x & y axis = 90.455°

* https://www.ephotozine.com/article/complete-guide-to-image-sensor-pixel-size-29652

# Back-project World Points into Image



# Camera Anatomy: Column Vectors of **P**

- Let the columns of P be $p_i$ , i = 1,…,4
- Then $p_1$, $p_2$, $p_3$ are the points at infinity of the world X, Y and Z axes respectively.
- Consider the point at infinity along X-axis: $D = (1, 0, 0, 0)^T$
- This will be imaged at

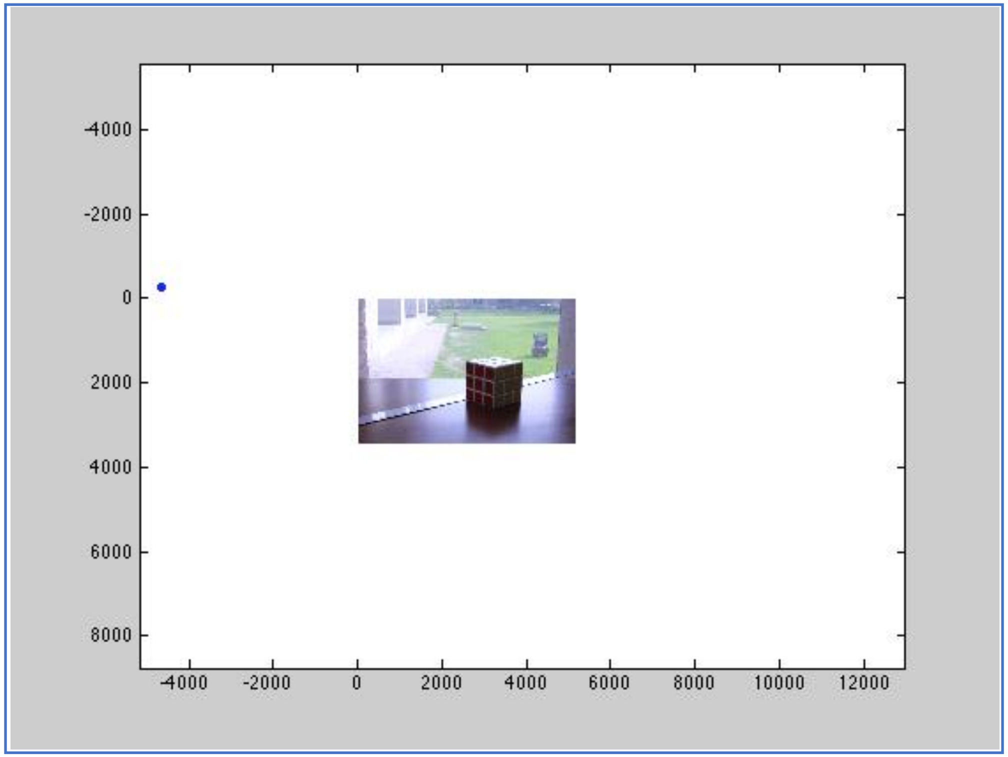
$$\mathrm{P}\mathbf{D} = \mathbf{p}_1$$

- Hence,
  - The first column of P is the image of the point at infinity along X-axis
  - The second column of P is the image of the point at infinity along Y-axis
  - The third column of P is the image of the point at infinity along Z-axis
- Similarly, $p_4$ the fourth column of P is …
- the image of world origin $(0,0,0,1)^T$

# Camera Anatomy: Column Vectors of **P**

```
P =

 -0.00010835    4.3034e-05     0.0047453      -0.68373
  -0.0019211    -0.0044849    0.00023615       -0.7297
  4.1144e-07   -3.5796e-07    1.1421e-07   -0.00028537
```

```
-263.35
-4669.2
      1
```

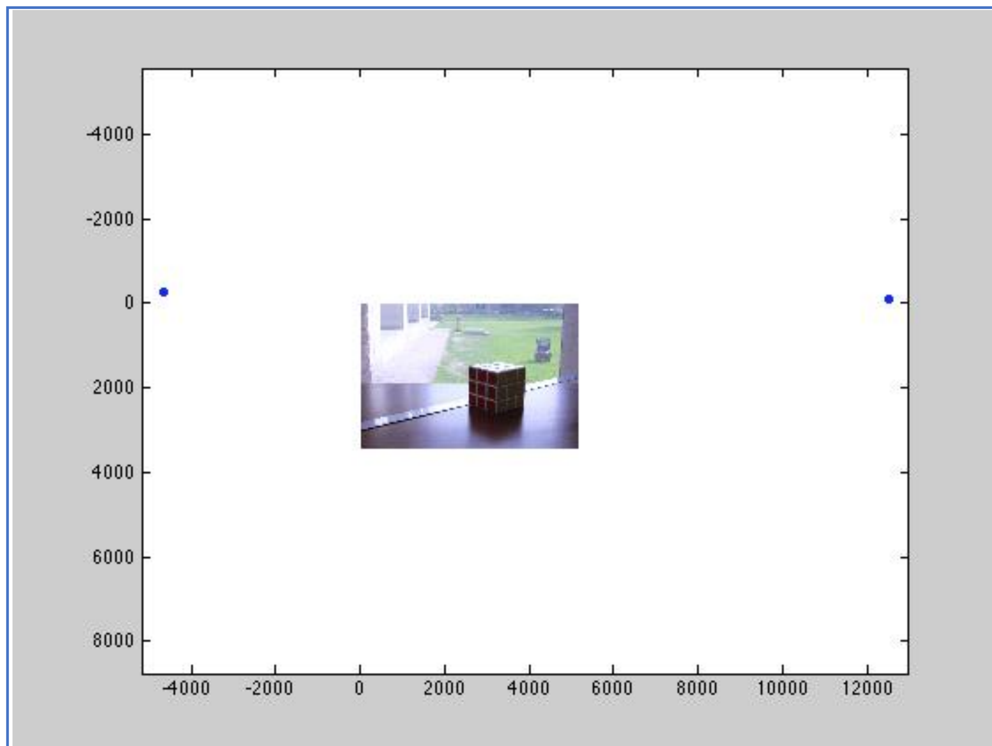


# Camera Anatomy:
# Column Vectors of **P**

P =

| | | | |
|---|---|---|---|
| -0.00010835 | 4.3034e-05 | 0.0047453 | -0.68373 |
| -0.0019211 | -0.0044849 | 0.00023615 | -0.7297 |
| 4.1144e-07 | -3.5796e-07 | 1.1421e-07 | -0.00028537 |

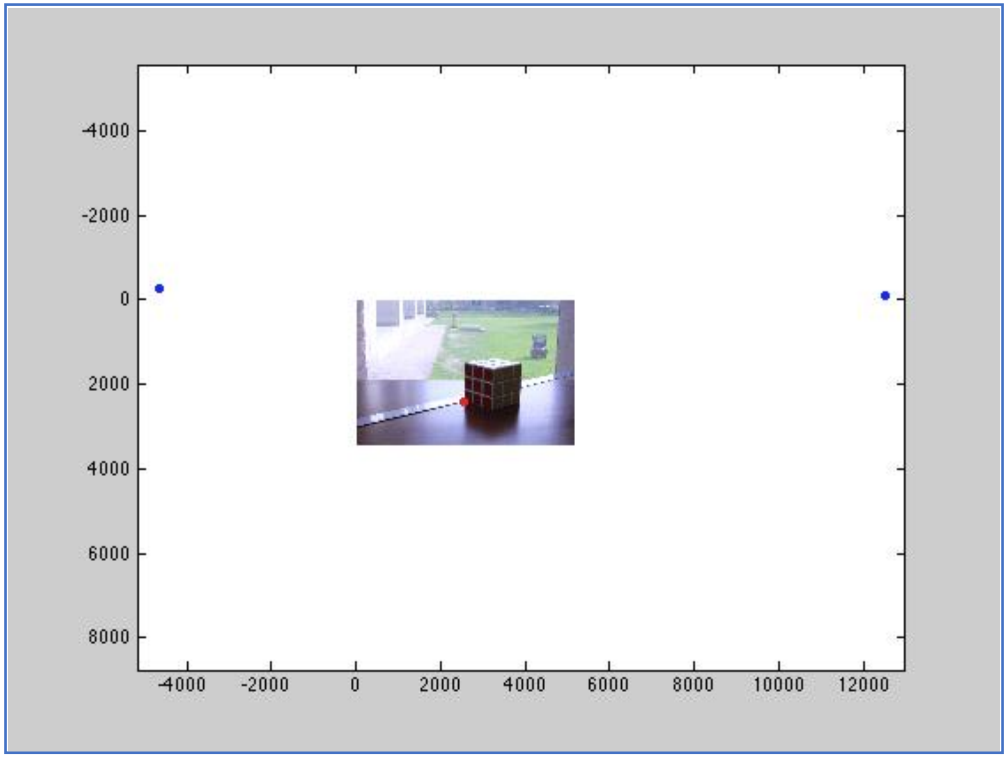
| | |
|---|---|
| -263.35 | -120.22 |
| -4669.2 | 12529 |
| 1 | 1 |



# Camera Anatomy: Column Vectors of **P**

```
P =

  -0.00010835    4.3034e-05     0.0047453      -0.68373
   -0.0019211    -0.0044849    0.00023615       -0.7297
   4.1144e-07   -3.5796e-07    1.1421e-07   -0.00028537
```

```
2396
2557
   1
```

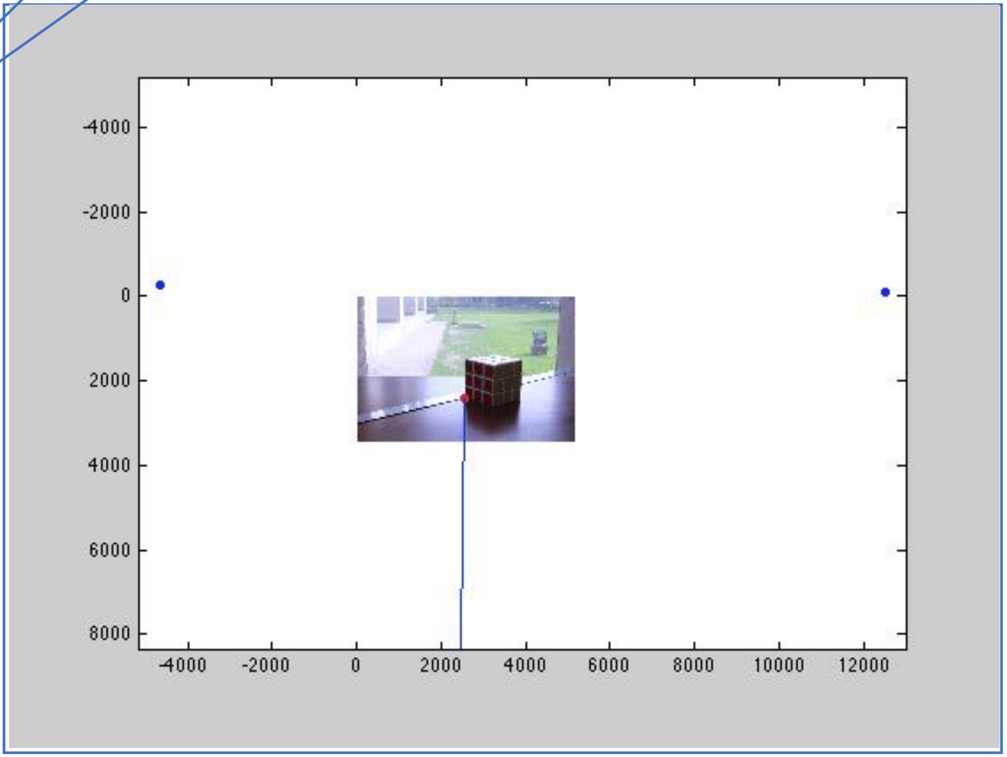


# Camera Anatomy: Column Vectors of **P**

P =

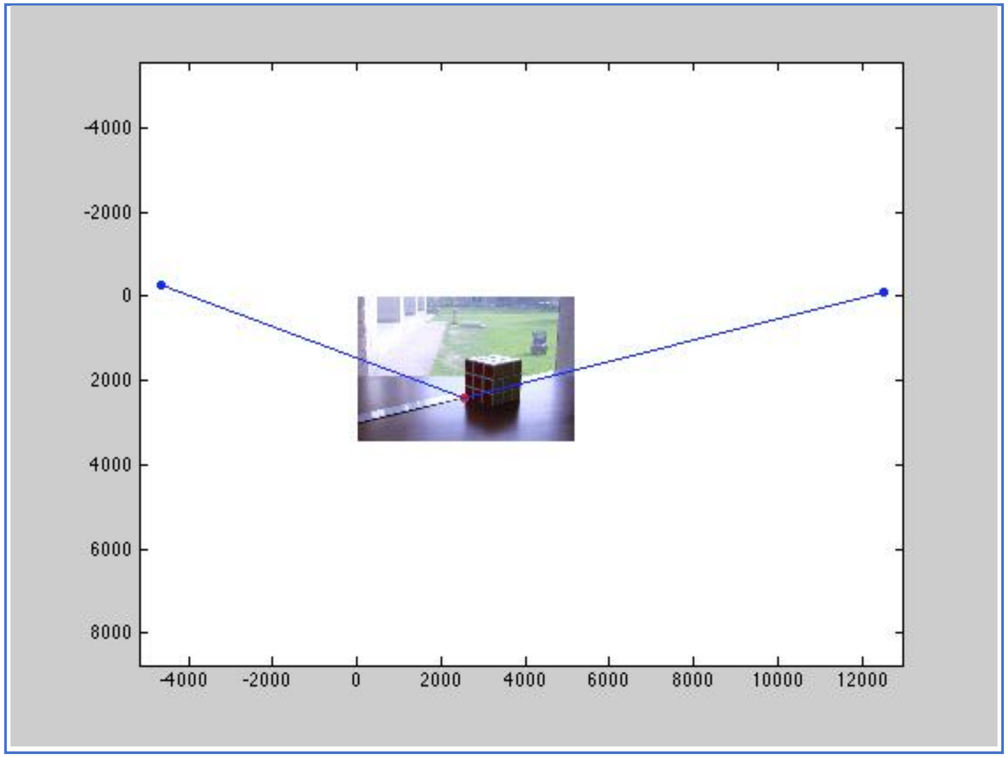
| | | | |
|---|---|---|---|
| -0.00010835 | 4.3034e-05 | 0.0047453 | -0.68373 |
| -0.0019211 | -0.0044849 | 0.00023615 | -0.7297 |
| 4.1144e-07 | -3.5796e-07 | 1.1421e-07 | -0.00028537 |

41550
2067.7
1



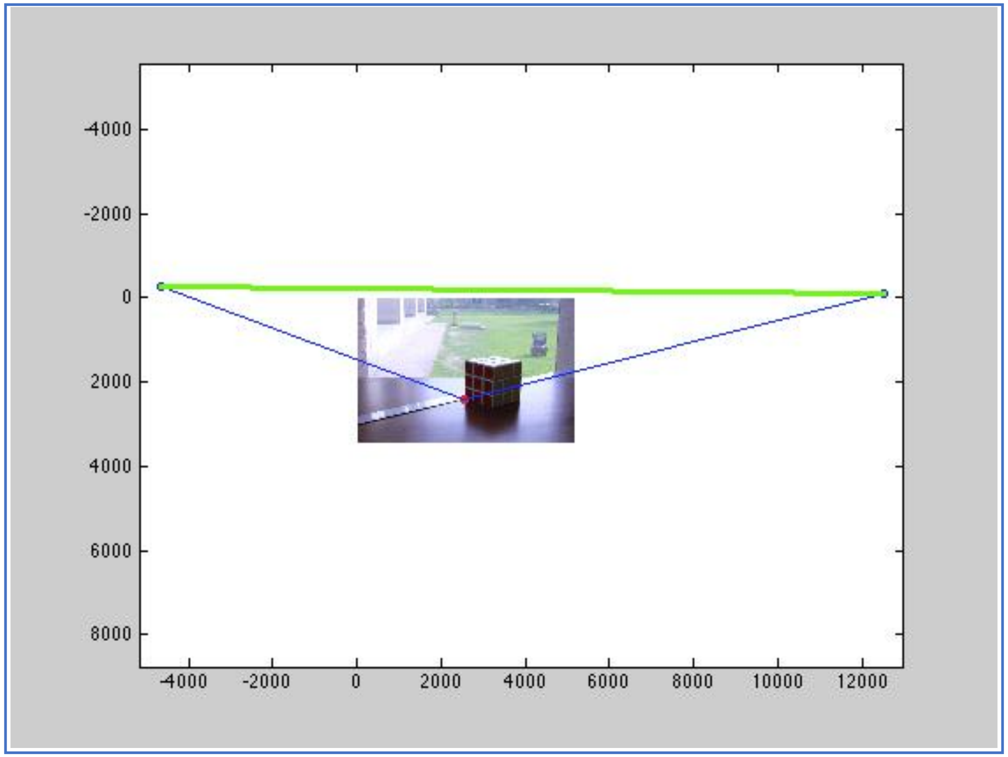
# Camera Anatomy: Column Vectors of **P**

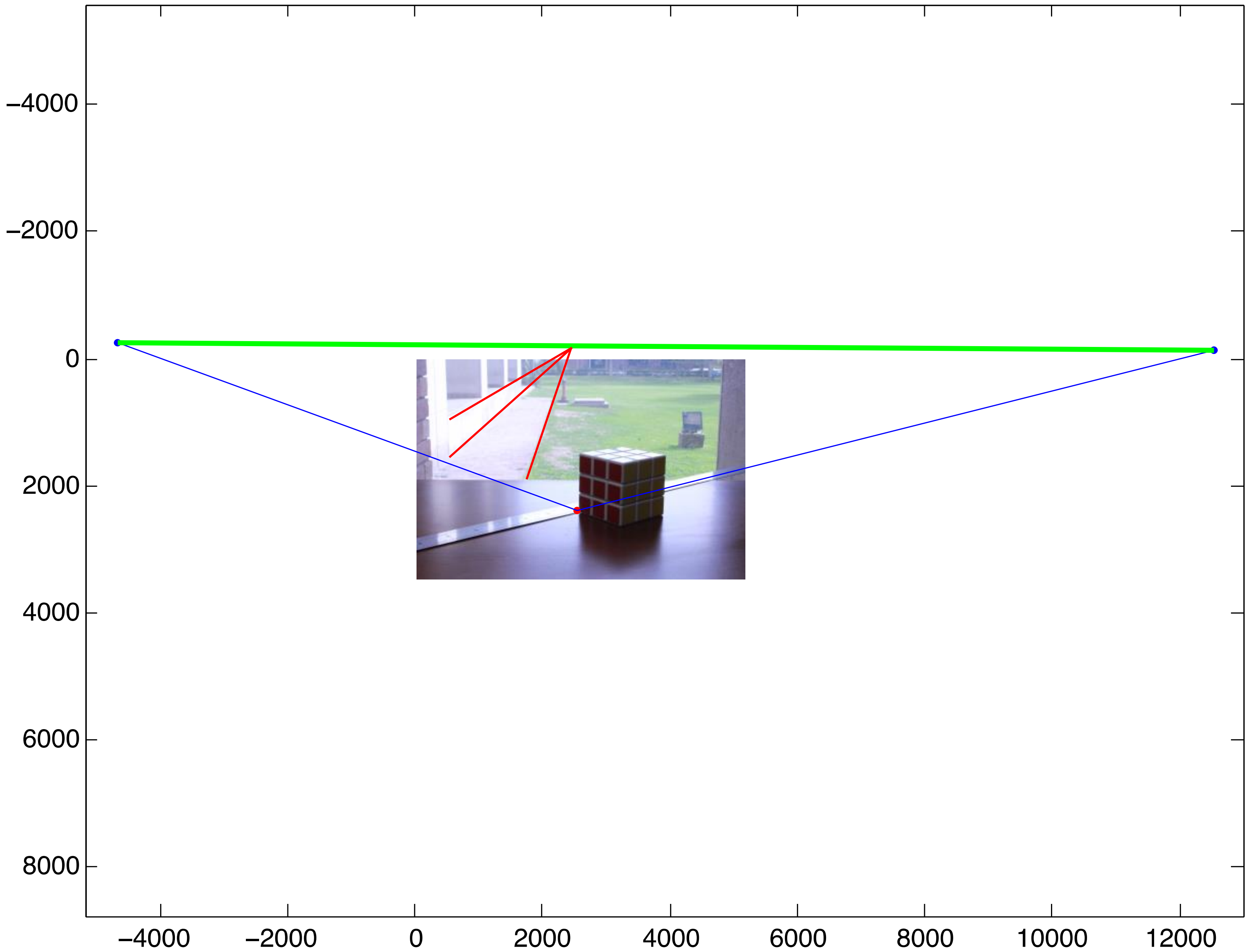
```
P =

  -0.00010835   4.3034e-05    0.0047453      -0.68373
   -0.0019211   -0.0044849   0.00023615       -0.7297
   4.1144e-07  -3.5796e-07   1.1421e-07  -0.00028537
```



# Camera Anatomy: Column Vectors of **P**

```
P =

  -0.00010835    4.3034e-05     0.0047453       -0.68373
   -0.0019211    -0.0044849    0.00023615        -0.7297
   4.1144e-07   -3.5796e-07    1.1421e-07   -0.00028537
```





−4000
−2000
0
2000
4000
6000
8000
−4000
−2000
0
2000
4000
6000
8000
10000
12000

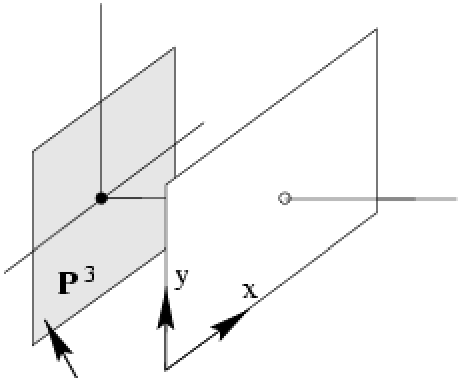
# Camera Anatomy: Row Vectors of P

- Row vectors are 4-vectors, which may be interpreted as planes.

$$\mathtt{P} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} = \begin{bmatrix} \mathbf{P}^{1\top} \\ \mathbf{P}^{2\top} \\ \mathbf{P}^{3\top} \end{bmatrix}$$

# Camera Anatomy: Row Vectors of **P**

- **Principal plane**: Plane through camera center, parallel to image plane, consisting of set of points X imaged on, line at infinity of image,
- i.e. $PX = (x,y,0)^T$
- Thus, a point lies on principal plane iff $P^{3T}X = 0$
- Thus, $P^3$ is the vector representing the principal plane
- Also, C lies on $P^3$ (verify)

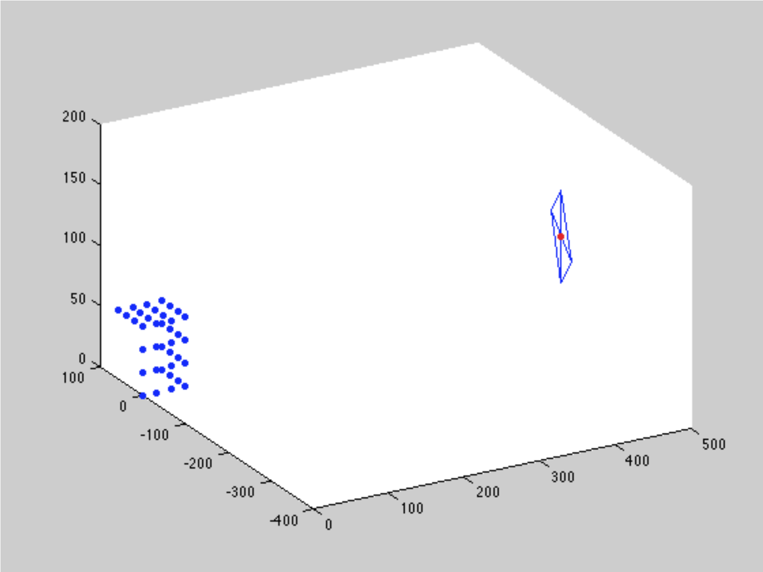


# Camera Anatomy: Row Vectors of **P**

```
P =

  -0.00010835   4.3034e-05    0.0047453     -0.68373
   -0.0019211   -0.0044849   0.00023615      -0.7297
   4.1144e-07  -3.5796e-07   1.1421e-07  -0.00028537
```

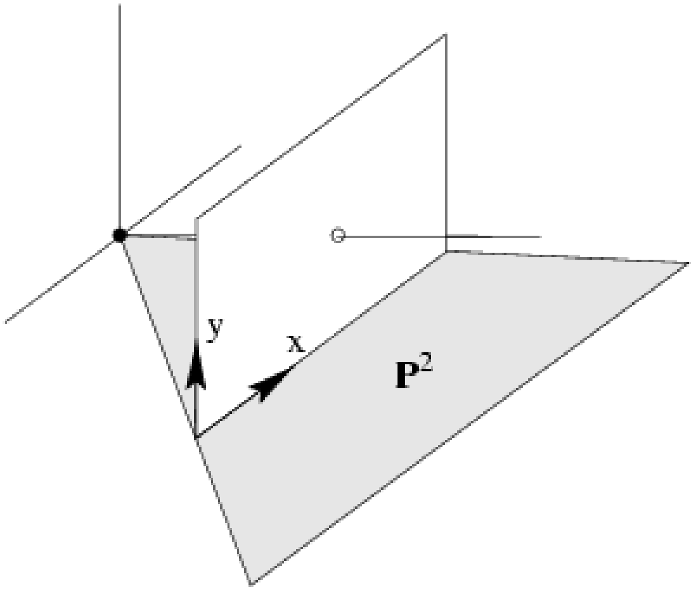
$P^3 = [4.1144e\text{-}07 \;\; -3.5796e\text{-}07 \;\; 1.1421e\text{-}07 \;\; -0.00028537]^T$

Verify $P^{3T}C = 1.3212e\text{-}14$



# Camera Anatomy: Row Vectors of **P**

- **Axis planes**: Consider the set of points X in $P^1$
- They must satisfy $P^{1T}X = 0$
- Hence, they will be imaged at $PX = (0, y, w)^T$, i.e. they are points on the y-axis of the image
- Also, C lies on $P^1$ (verify)
- Hence, $P^1$ is defined by the join of C and line x = 0 in the image
- Similarly, $P^2$ is defined by the join of C and line y = 0



# Camera Anatomy: Row Vectors of **P**

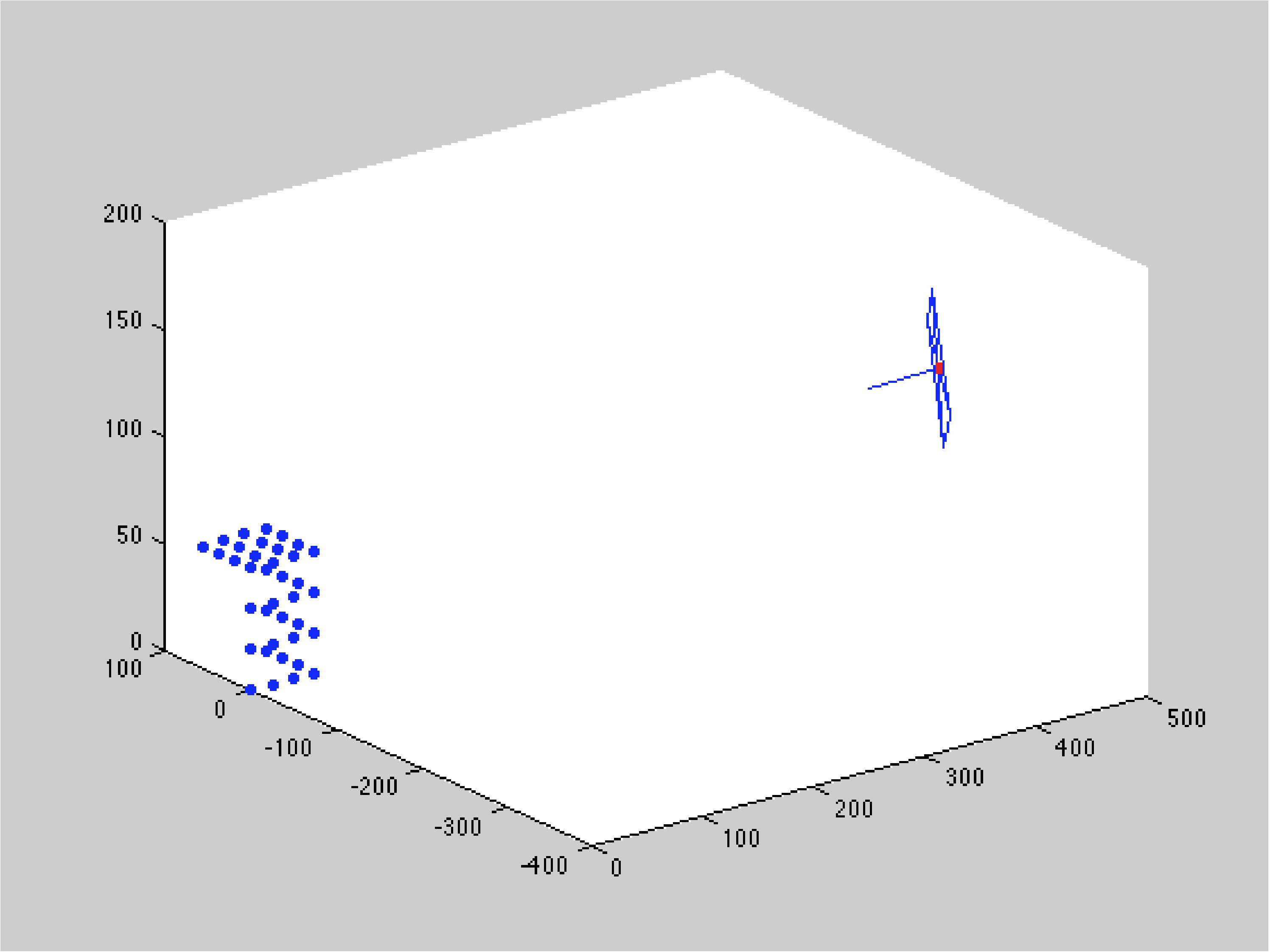
- Axis planes are dependent on the choice of image axes, but principal plane is not.
- Intersection of planes $P^1$ and $P^2$ is the line joining the camera center and the image origin, i.e. the back projection of image origin
  - Not the optical axis in general…
- Camera center lies on all three planes $P^1$, $P^2$, $P^3$

# Camera Anatomy: Principal Point and Principal Axis

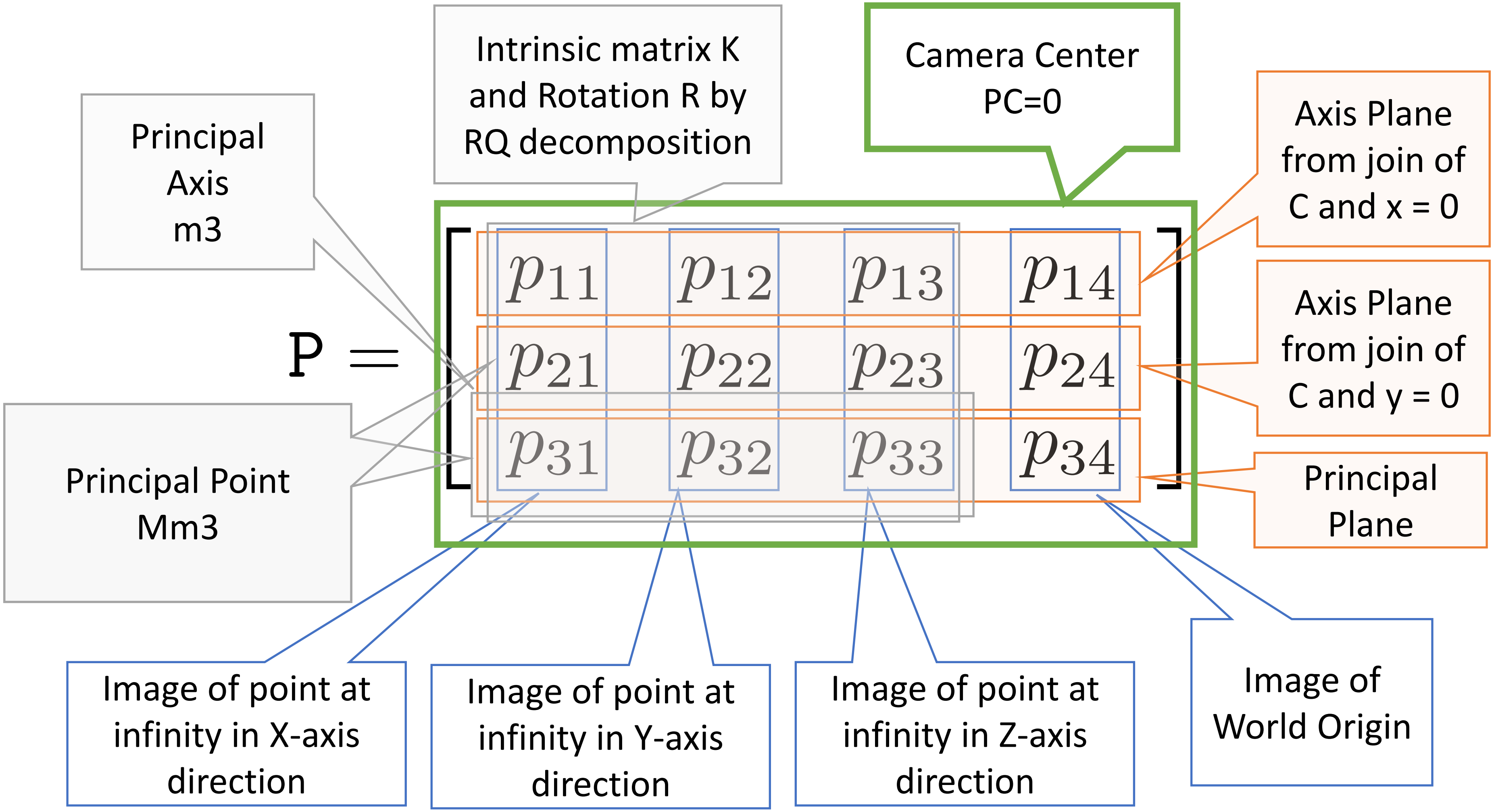
- The principal axis is the ray through the camera center perpendicular to the principal plane $P^3$, which will intersect the imaging plane at the principal point.
- In general, the normal of a plane $\pi=(\pi_1, \pi_2, \pi_3, \pi_4)^T$ is given by $(\pi_1, \pi_2, \pi_3)^T$
- Thus principal axis is given by $(p_{31}, p_{32}, p_{33})^T$
- Consider the point at infinity in the direction of principal axis, i.e. $(p_{31}, p_{32}, p_{33}, 0)^T = \hat{\mathbf{P}}^3$
- We can project this point back to the image to get the coordinates of the principal point as $\mathbf{x}_0 = \mathtt{P}\hat{\mathbf{P}}^3$
- This involves only the first 3x3 block of P. Hence

$$\mathbf{x}_0 = \mathtt{M}\mathbf{m}^3$$

- where $m^{3T}$ is the last row of M, and M the first 3x3 block of P


200
150
100
50
0
100
0
-100
-200
-300
-400
0
100
200
300
400
500

# Summary: Camera Anatomy



# Action of Camera on Points, Lines and Planes

# Action of Camera on Points

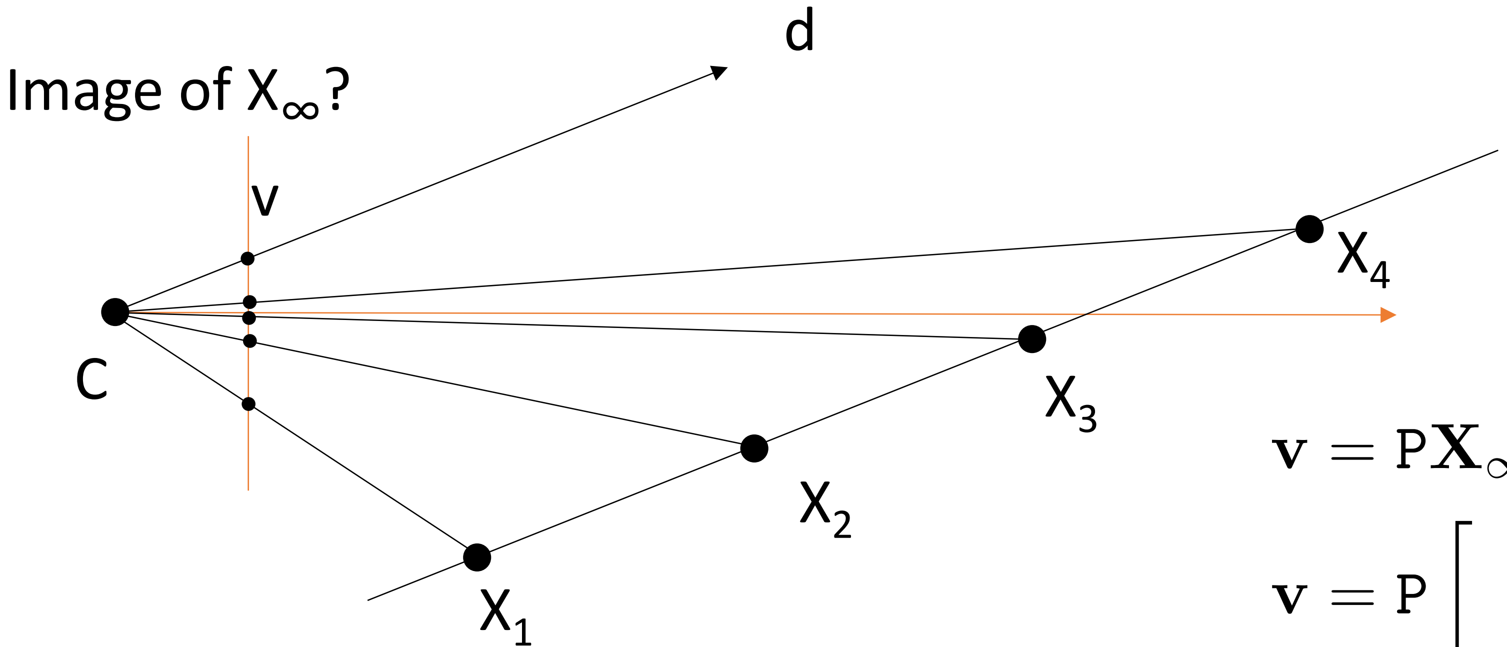
- Action of camera on points is familiar to us

$$\mathbf{x} = \mathrm{K}\left[\mathrm{R} \mid -\mathrm{R}\tilde{\mathbf{C}}\right]\mathbf{X}$$

- In canonical view

$$\mathbf{x} = \mathrm{K}\left[\mathrm{I} \mid \mathbf{0}\right]\mathbf{X}$$

# Vanishing Points




$$\mathbf{v} = \mathtt{P}\mathbf{X}_{\infty}$$

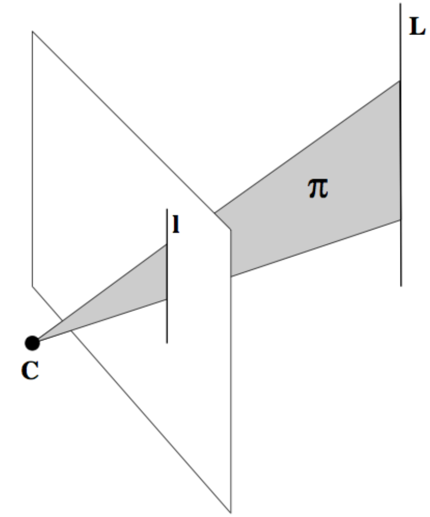
$$\mathbf{v} = \mathtt{P}\begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$

$$\mathbf{v} = \mathtt{K}\left[\mathtt{I} \mid \mathbf{0}\right]\begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$

$$\mathbf{v} = \mathtt{K}\mathbf{d}$$

# Action of Camera on Lines

- A line in 3-space will project to a line in the image.
- Geometric Proof: The join of the line L in 3-space and the camera center C forms a plane, and this plane will intersect the imaging plane in a line.
- Algebraic Proof:




Consider two points $\mathbf{A}$ and $\mathbf{B}$ in 3-space. The line formed by their join can be parameterized as $\mathbf{X}(\mu) = \mathbf{A} + \mu\mathbf{B}$. $\mathbf{X}(\mu)$ will project in the image to

$$\begin{aligned} \mathbf{x}(\mu) &= \mathtt{P}(\mathbf{A} + \mu\mathbf{B}) = \mathtt{P}\mathbf{A} + \mu\mathtt{P}\mathbf{B} \\ &= \mathbf{a} + \mu\mathbf{b} \end{aligned}$$

which is the line joining image points $\mathbf{a}$ and $\mathbf{b}$.

# Back Projection of Points to Rays

- Given a point x in the image of a camera with matrix **P**, we want to determine set of points in the world that map to this point
- i.e. the ray in the world that this point back-projects to.
- Note that camera center C always lies along the ray.
- Claim: $P^{+}x$ also lies along the ray, where $P^{+} = P^{T}(PP^{T})^{-1}$
- Proof:

The image of $P^{+}x$ will be at

$$PP^{+}x$$

$$= PP^{T}(PP^{T})^{-1}x$$

$$= x$$

Therefore $P^{+}x$ must be along the ray

# Back Projection of Points to Rays

- Since we know two points along the ray, $\mathbf{C}$ and $P^{+}x$
- Therefore, points along the ray can be written in parametric form as

$$\mathbf{X}(\lambda) = \mathrm{P}^{+}\mathbf{x} + \lambda\mathbf{C}$$

# Back Projection of Points to Rays

- In canonical view, relationship is even more simpler
- Consider the back-projected ray which is given by a direction d
- All points on this ray can be written as the join of camera center and the ray vector

$$\tilde{\mathbf{X}}(\lambda) = \mathbf{0} + \lambda\mathbf{d} = \lambda\mathbf{d}$$

- The image of such a point in canonical view will be at

$$\mathbf{x} = \mathtt{K}\left[\mathtt{I} \mid \mathbf{0}\right]\left[\begin{array}{c} \lambda\mathbf{d} \\ 1 \end{array}\right]$$

$$\mathbf{x} = \mathtt{K}\mathbf{d}$$

- So, given x, the ray direction in canonical view can be computed simply as $\mathbf{d} = \mathtt{K}^{-1}\mathbf{x}$

- Camera calibration matrix K relates the image point to a ray direction

# Back Projection of Lines

- Lines in image will back project to planes in the world.
- Result: The set of points in the world mapping to a line l via the camera matrix P is the plane $P^T l$
- Proof

  A point x lies on l iff $x^T l = 0$.

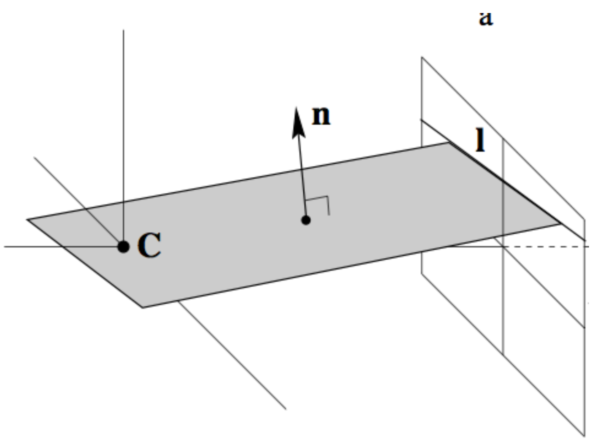
  A world point X maps to a point PX, which will lie on l iff

  $$X^T P^T l = 0$$

  Thus if $P^T l$ is taken to represent a plane, then X lies on this plane iff X maps to a point on the line l

  Hence, $P^T l$ is the plane which is the back projection of line l.

# Relationship between Image Line and Plane Normal




- Result: An image line l defines a plane through the camera center with normal direction $n = K^T l$ in the camera canonical coordinate system
- Proof:

  Points x on line l back-project to directions $d = K^{-1}x$

  Since these direction vectors lie on the plane, they are orthogonal to the plane normal n.

  Thus, $d^T n = 0$

  $x^T K^{-T} n = 0$

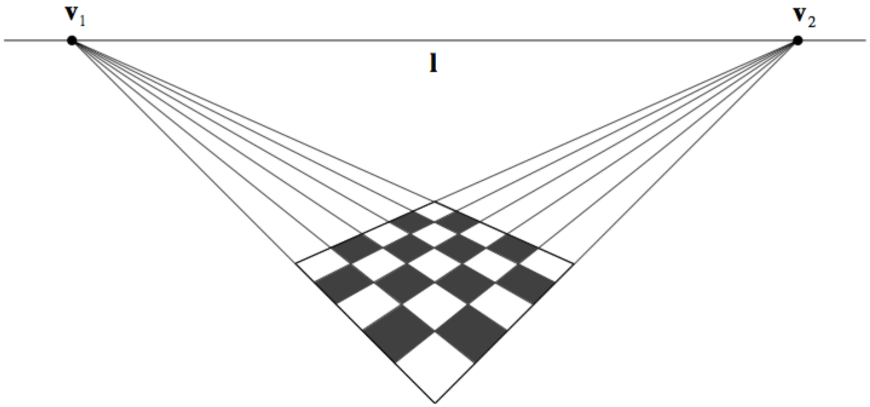
  Since points on l satisfy $x^T l = 0$, it follows that

  $l = K^{-T} n$

  Hence $n = K^T l$

# Vanishing Line

- The vanishing line of a plane is the image of the line at infinity of the plane
- The vanishing line will depend only on the *orientation* of the plane, and not on its absolute position
- Parallel planes have the same vanishing line



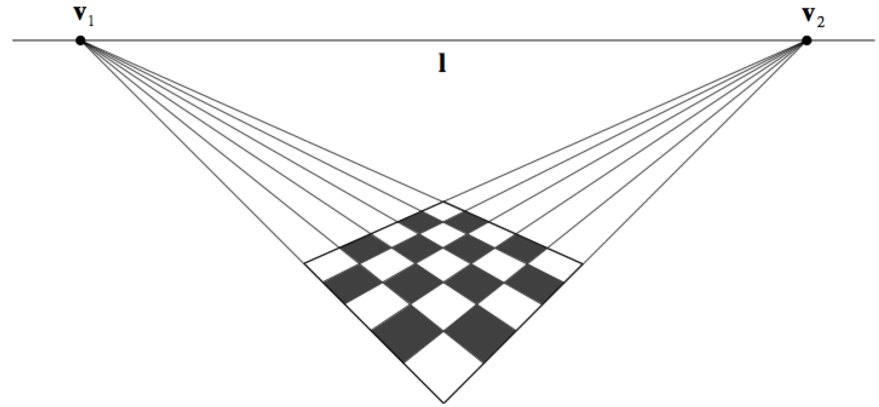
# Vanishing Line

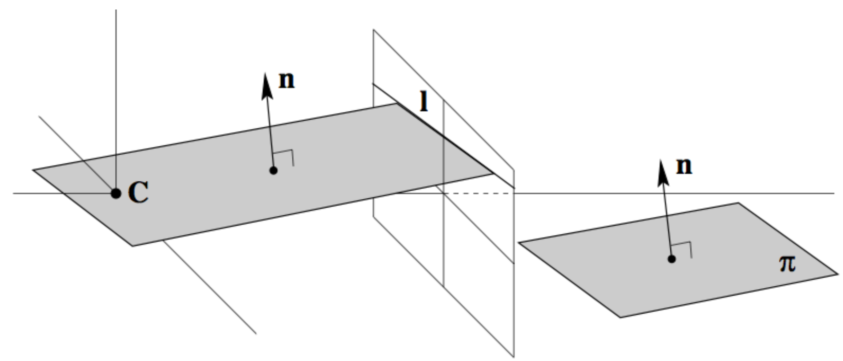
- If camera calibration matrix K is known, a scene plane's vanishing line can be used to find the plane's orientation relative to the camera.
- In the canonical coordinate system, the orientation of the plane having the vanishing line l in the image is given by

$$n = K^T l$$

- The vanishing line as a function of plane normal is given by

$$l = K^{-T} n$$

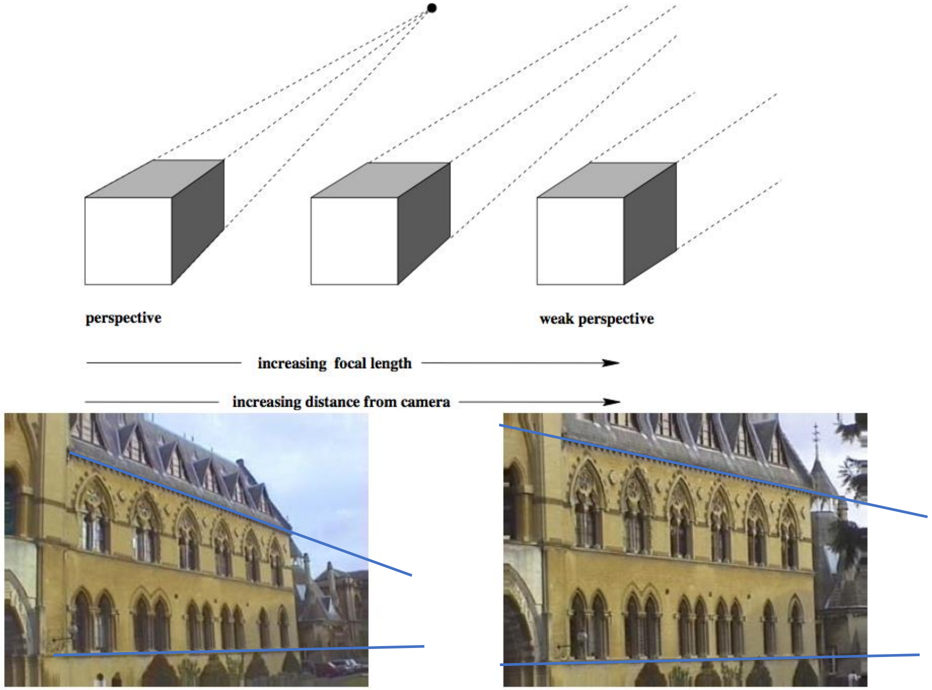


# Orthographic Cameras

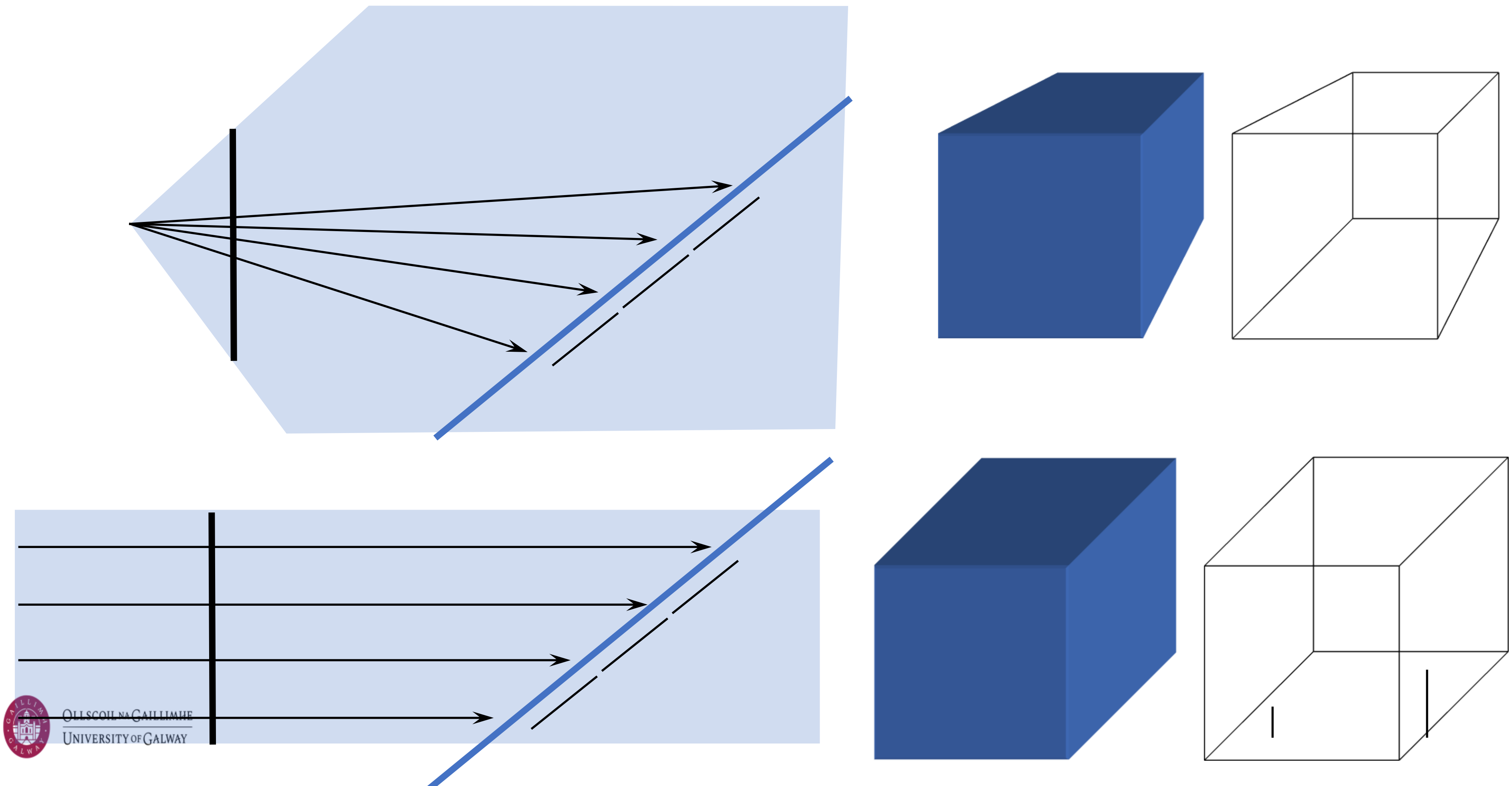
# Cameras at Infinity

- Consider the image sequence in which the camera moves away from an object but zooms in to keep the size of the object the same.



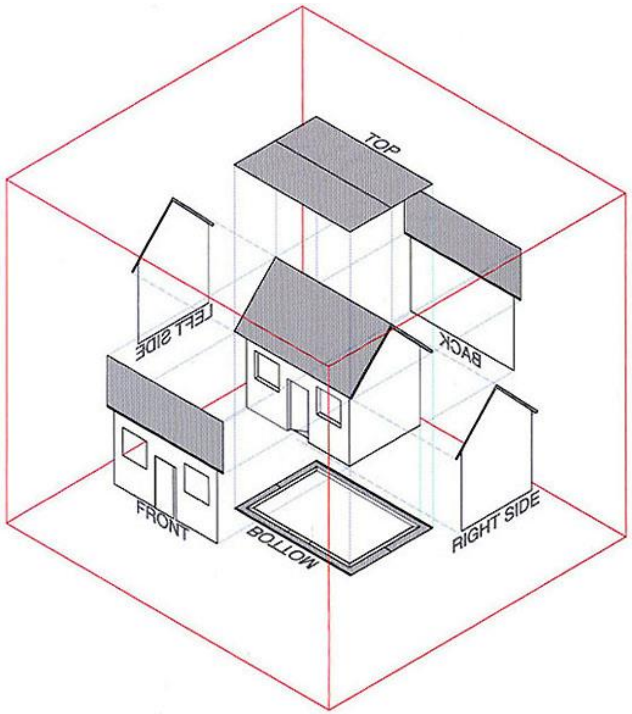
# Cameras at Infinity

- If the camera center moves back from the scene to infinity, then all rays entering the camera will be parallel



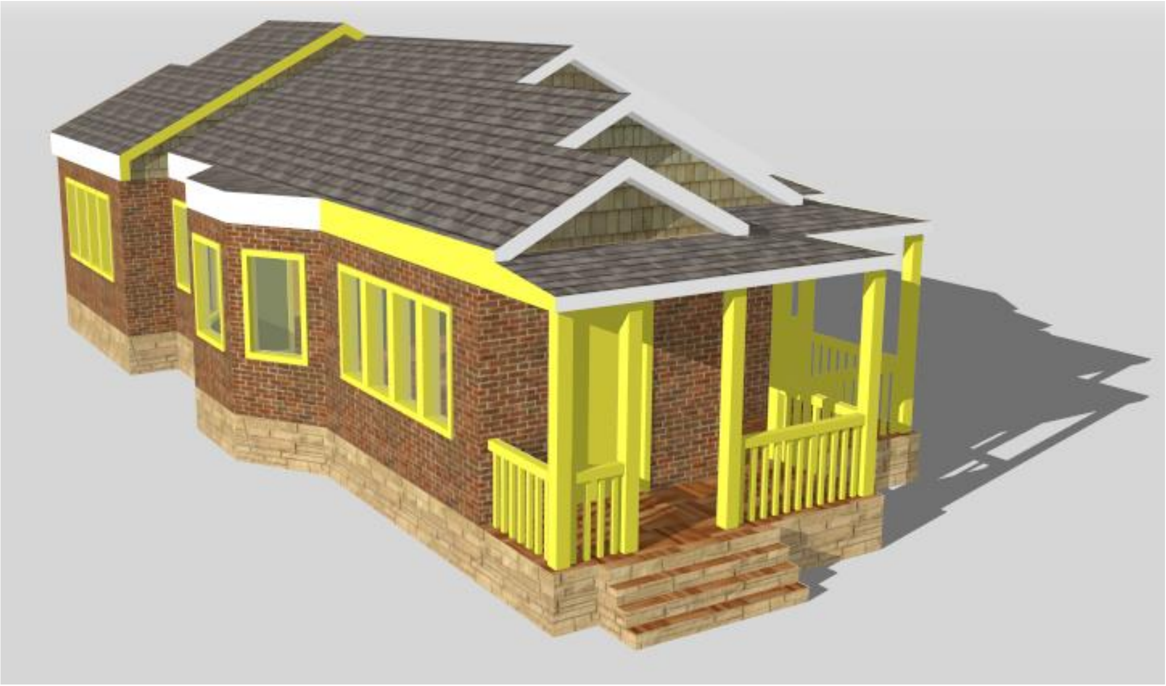
# Another Type of Camera: Orthographic Camera

- Parallel Lines remain parallel and do not converge (also termed parallel projection)

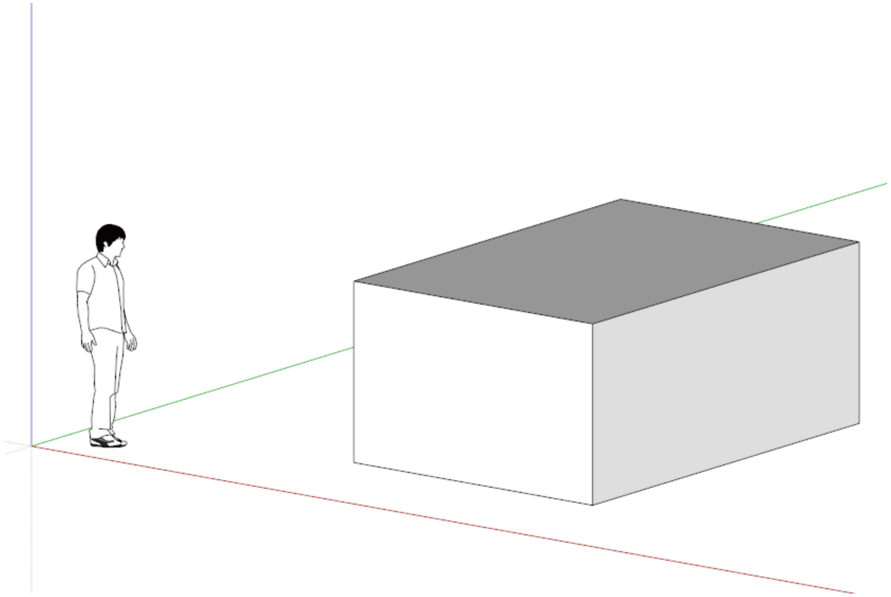



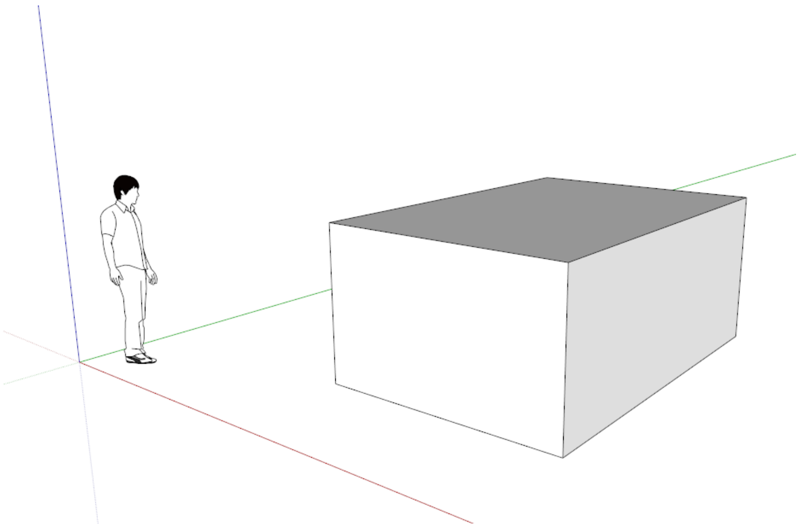
# Type of Projection?

# Type of Projection?

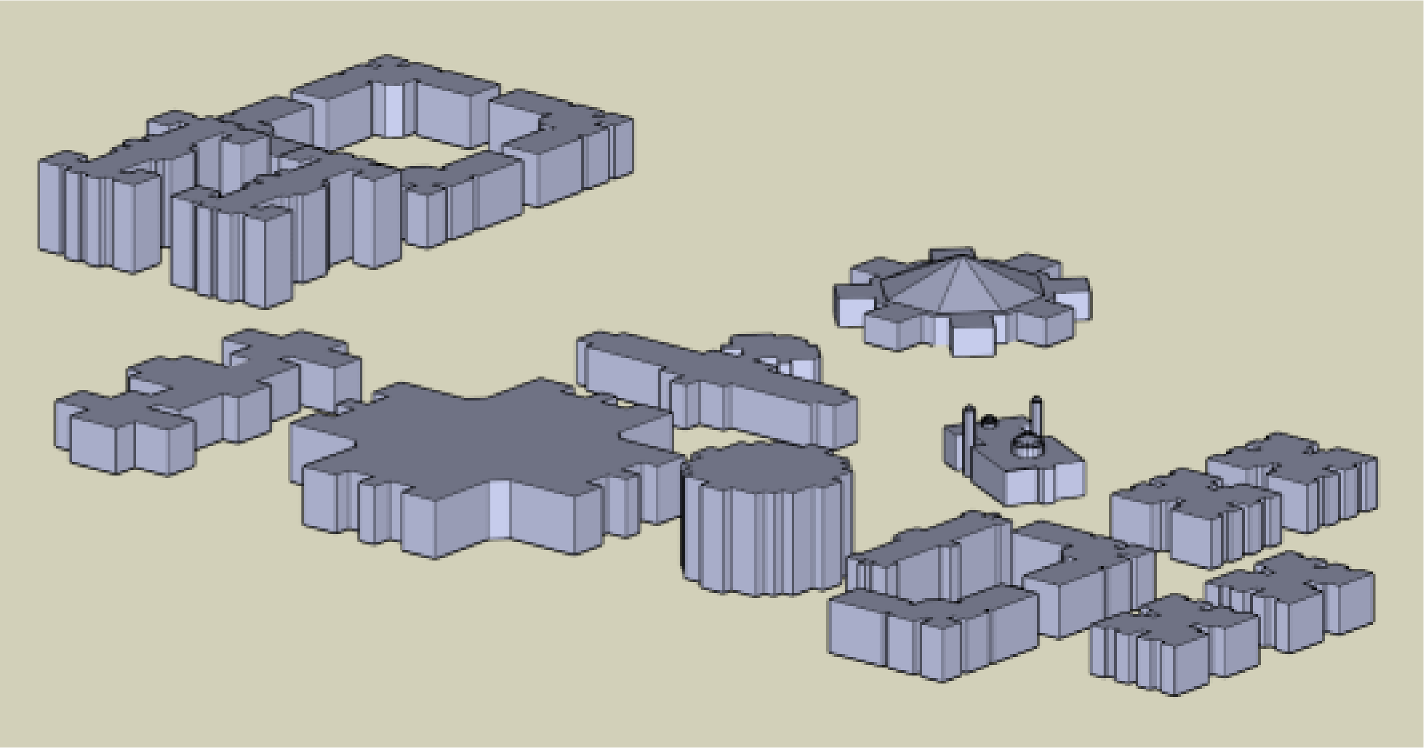


# Type of Projection?

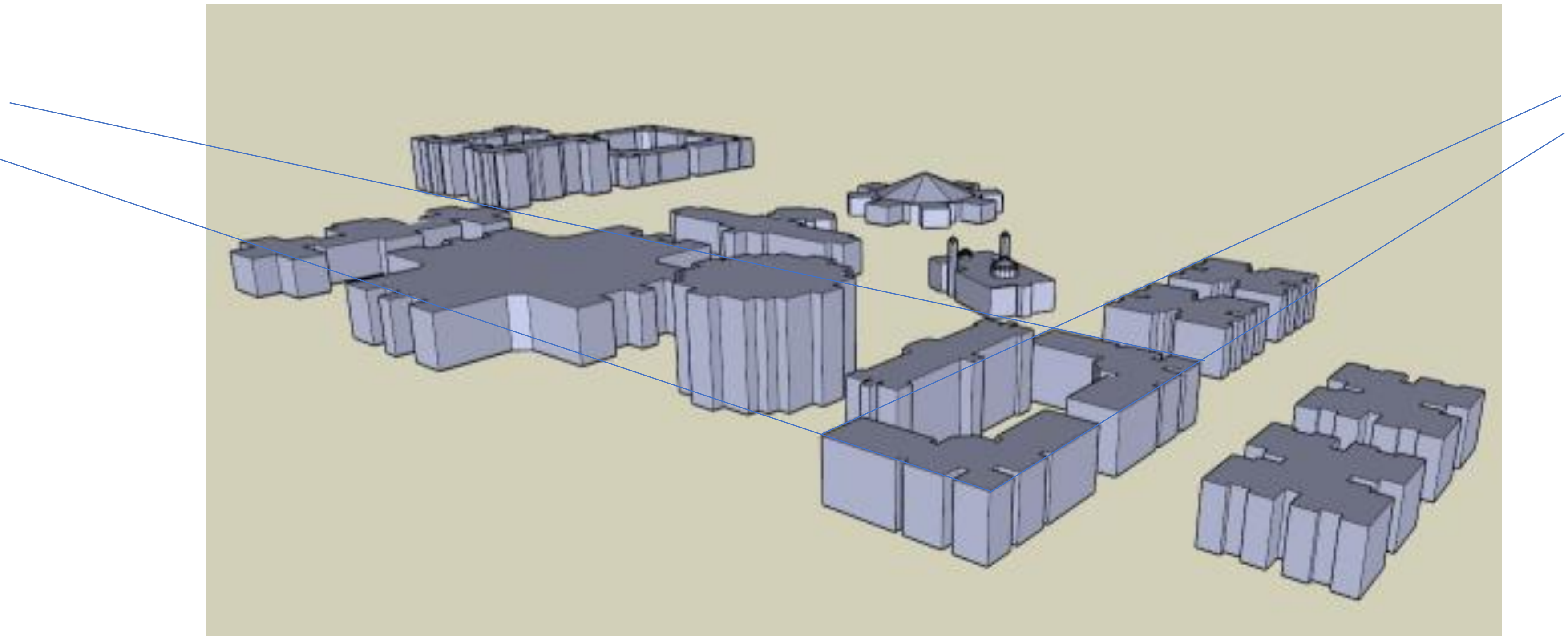




# Type of Projection?

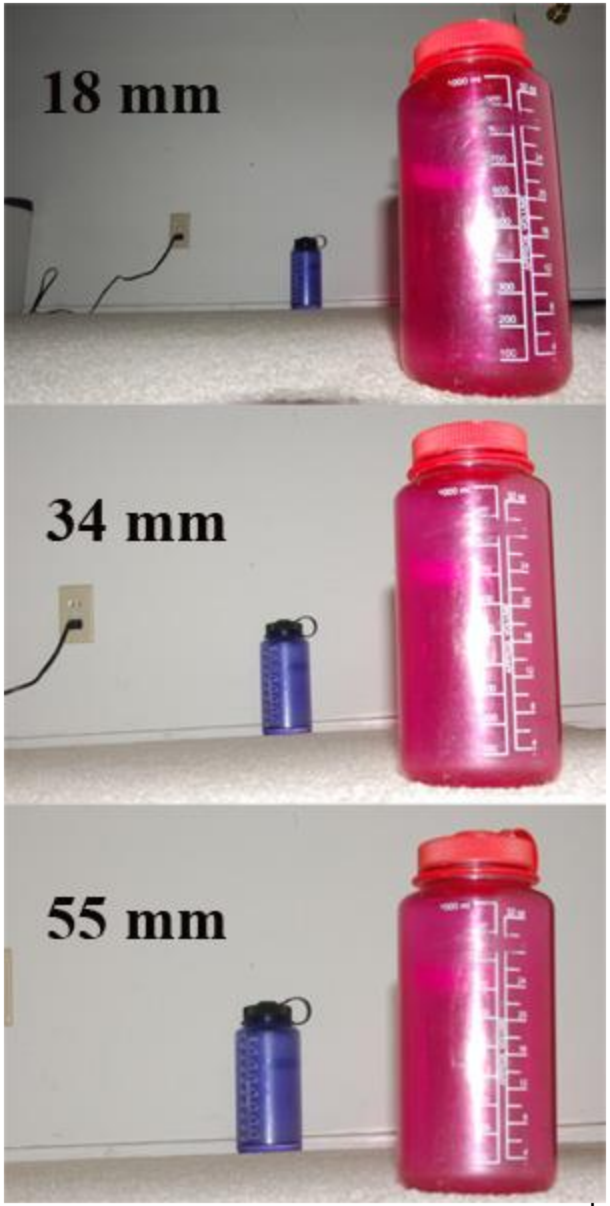


# Type of Projection?



# Perspective Distortion

- Perspective distortion: the effect that further away objects appear smaller in size
- As focal length increases (more zoom), perspective distortion becomes less
- Orthographic camera can be considered as being very far away so there is no variation in Z, and having very long focal length. Hence it has no perspective distortion.
- Equal lengths in the world will appear of equal size in the image



# Orthographic Projection

- In canonical view:
- The relationship between image coordinates and scene coordinates is:

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

- In matrix form

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

M (first 3x3 block of P) is now a singular matrix

Last row is $[0, 0, 0, 1]^T$, so this is also termed as an affine camera

Camera center is given by null(P) = $[0\ 0\ 1\ 0]^T$ is a point at infinity

# Orthographic Projection

- In general view:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- This can also be written as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_X \\ r_{21} & r_{22} & r_{23} & t_Y \\ r_{31} & r_{32} & r_{33} & t_Z \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Or

$$\tilde{\mathbf{x}} = \mathtt{K}_o \left[\mathtt{R} \mid \mathtt{T}\right] \mathbf{X}$$

- Note that 3rd row of R|T does not matter

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \begin{bmatrix} t_X \\ t_Y \end{bmatrix}$$

# Relationship between Orthographic and Perspective Projection

- Consider a pinhole camera which is very far away and zoomed into the scene.
- Since the depth variation of the scene is small compared to the distance of the camera, it may be approximated by a constant value.
- Hence

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

- Since $\frac{f}{\bar{Z}}$ is now a constant, we can write

$$x = mX \qquad y = mY$$

- which is scaled orthographic projection

# Some properties of Orthographic Projection

- Parallel lines remain parallel
- There is no perspective distortion. Equal lengths in the world appear as equal lengths in the image.
- Images of a plane taken by an orthographic camera are related by an affine transformation [Proof?]

# Practical Cases of Orthographic Camera

- A camera which is actually very far away compared to the depth variation in the scene can be approximated by an orthographic camera
- Orthographic projection is often used in computer games
- Scanner is provides an orthographic projection of a document image
- Also used in some algorithms as an approximation for mathematical simplicity (e.g. Tomasi / Kanade Structure for Motion Algorithm)

# Satellite Cameras

- Satellite images are typically taken by a line scan pushbroom camera, which is orthographic along the direction of motion and perspective in the line-scan direction