Where am I?
Using Vanishing Points

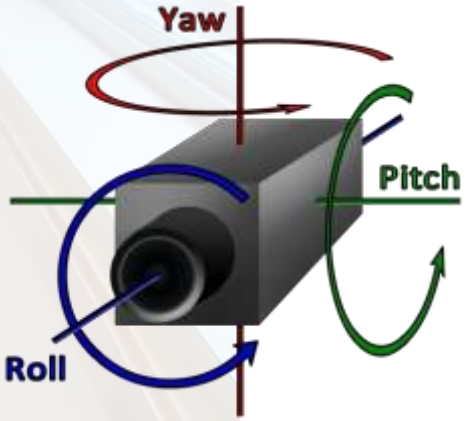
Vanishing point

Vanishing line for horizon

Vanishing point

What can vanishing line tell us about me?

- Horizon
- Camera pitch angle (looking down)
- Camera roll angle (tilted toward right)

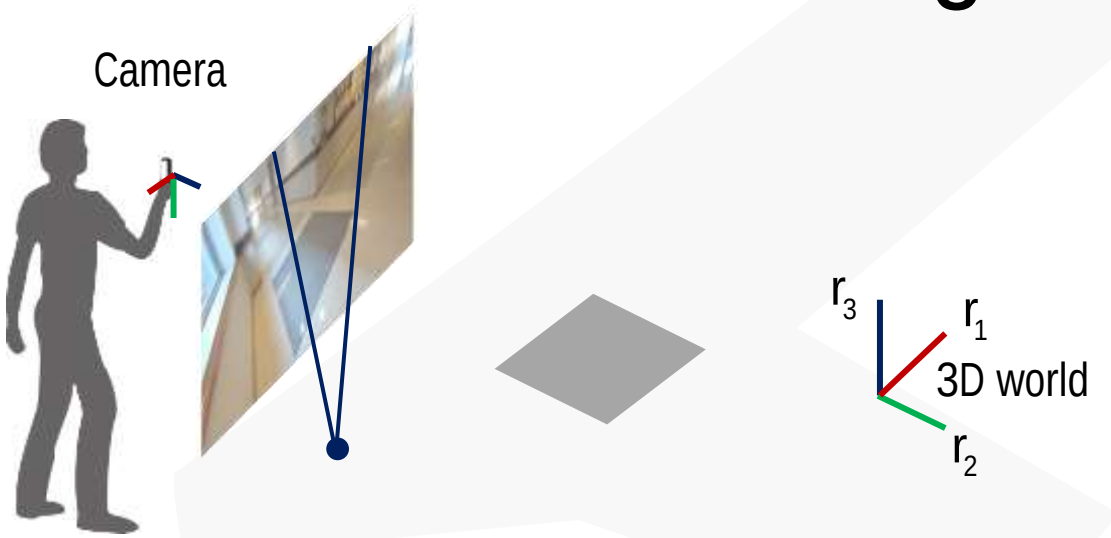


Where am I w.r.t. Ground Plane?

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

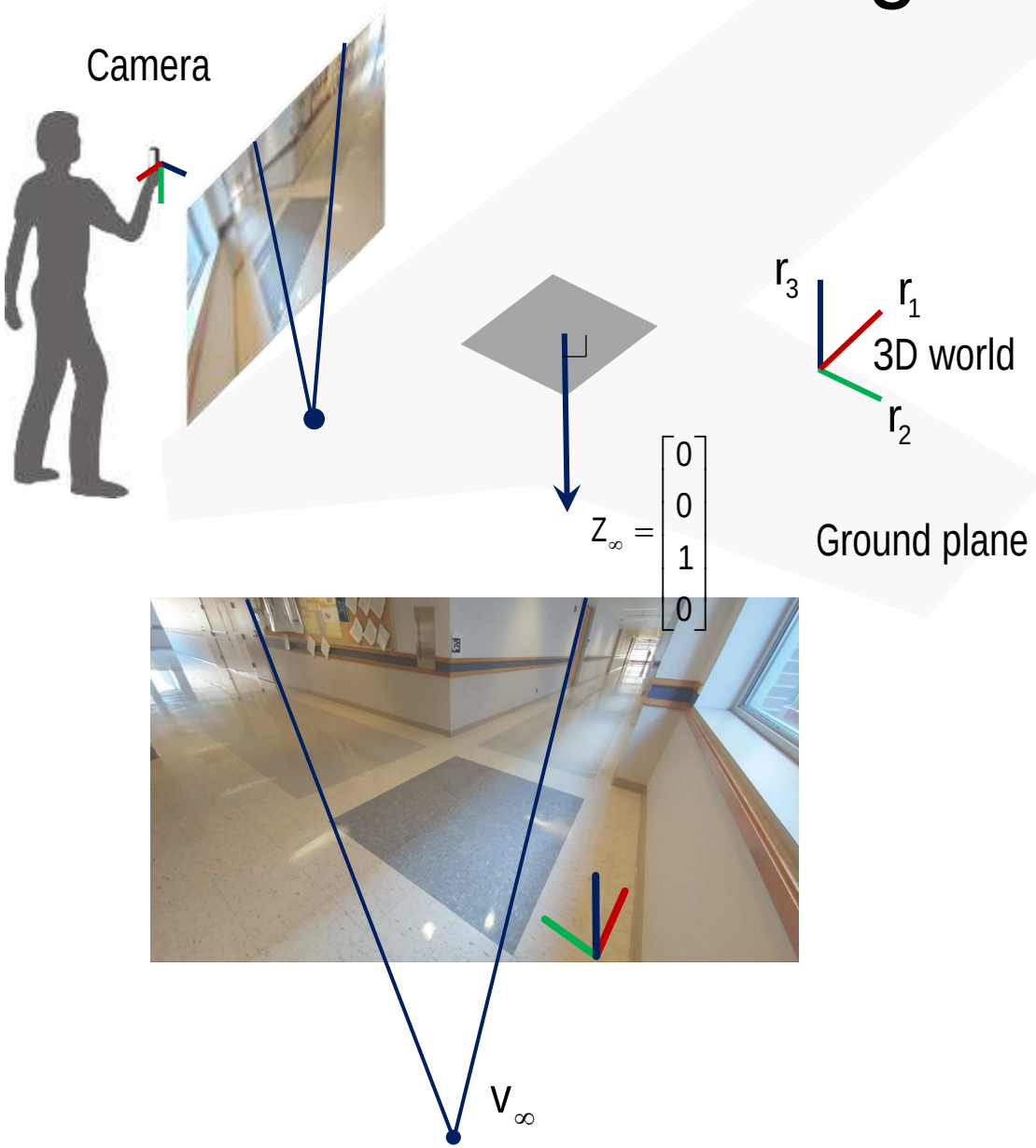
How to compute?

What can a Vanishing Point tell us about?

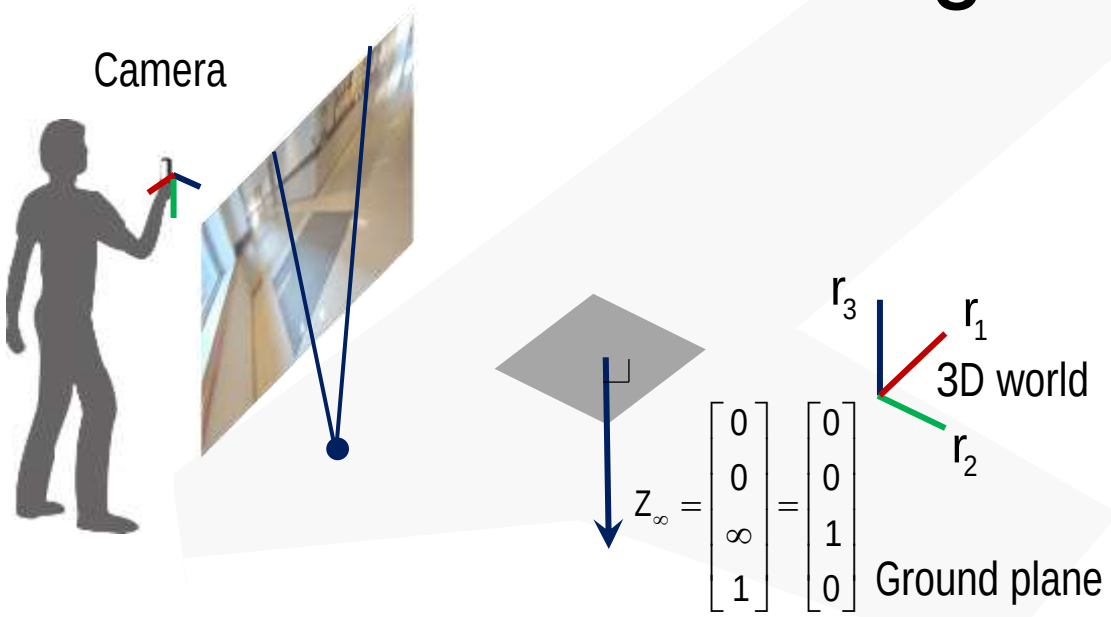


v_∞

What can a Vanishing Point tell us about?

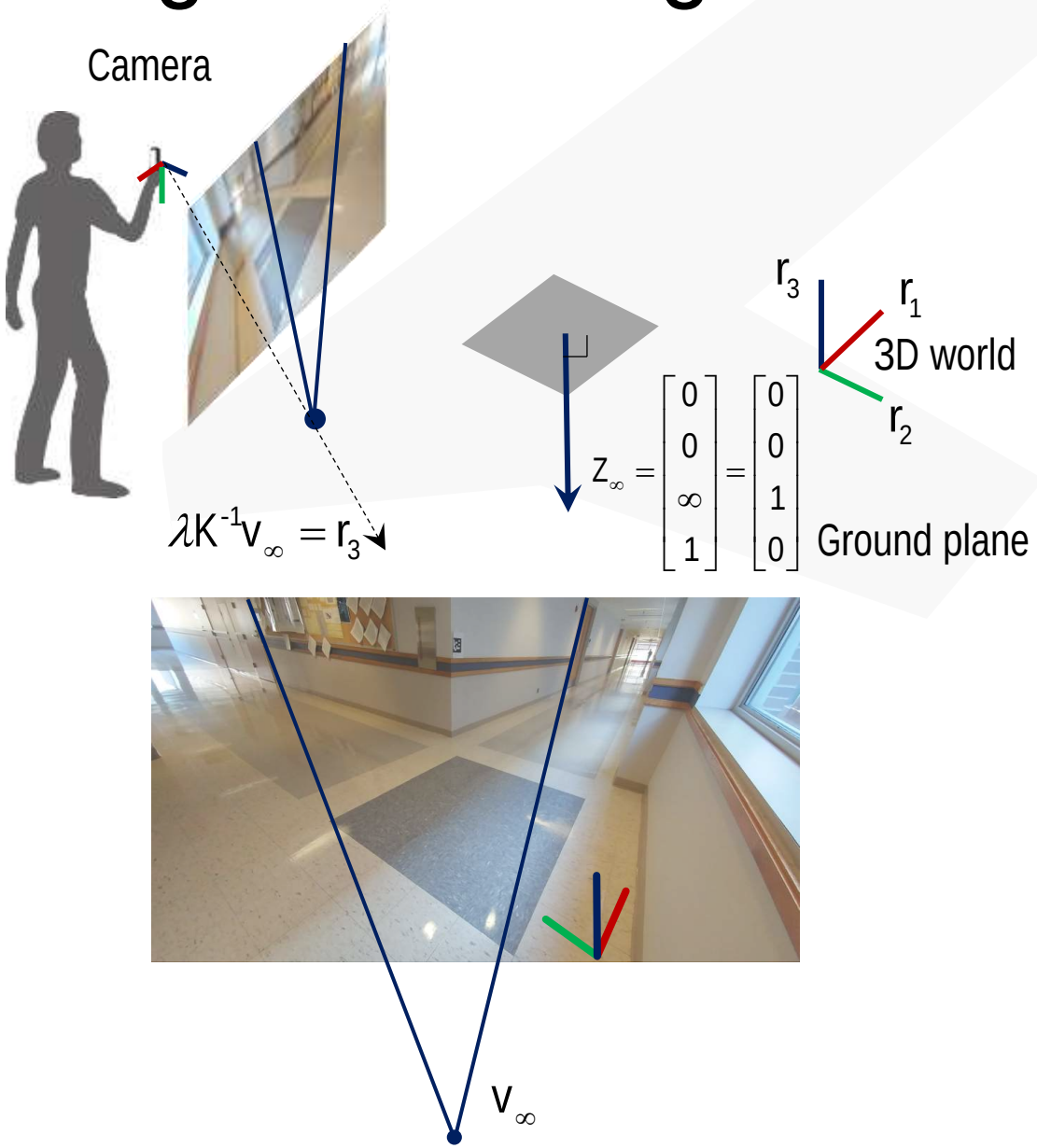


What can a Vanishing Point tell us about?



$$\lambda v_\infty = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

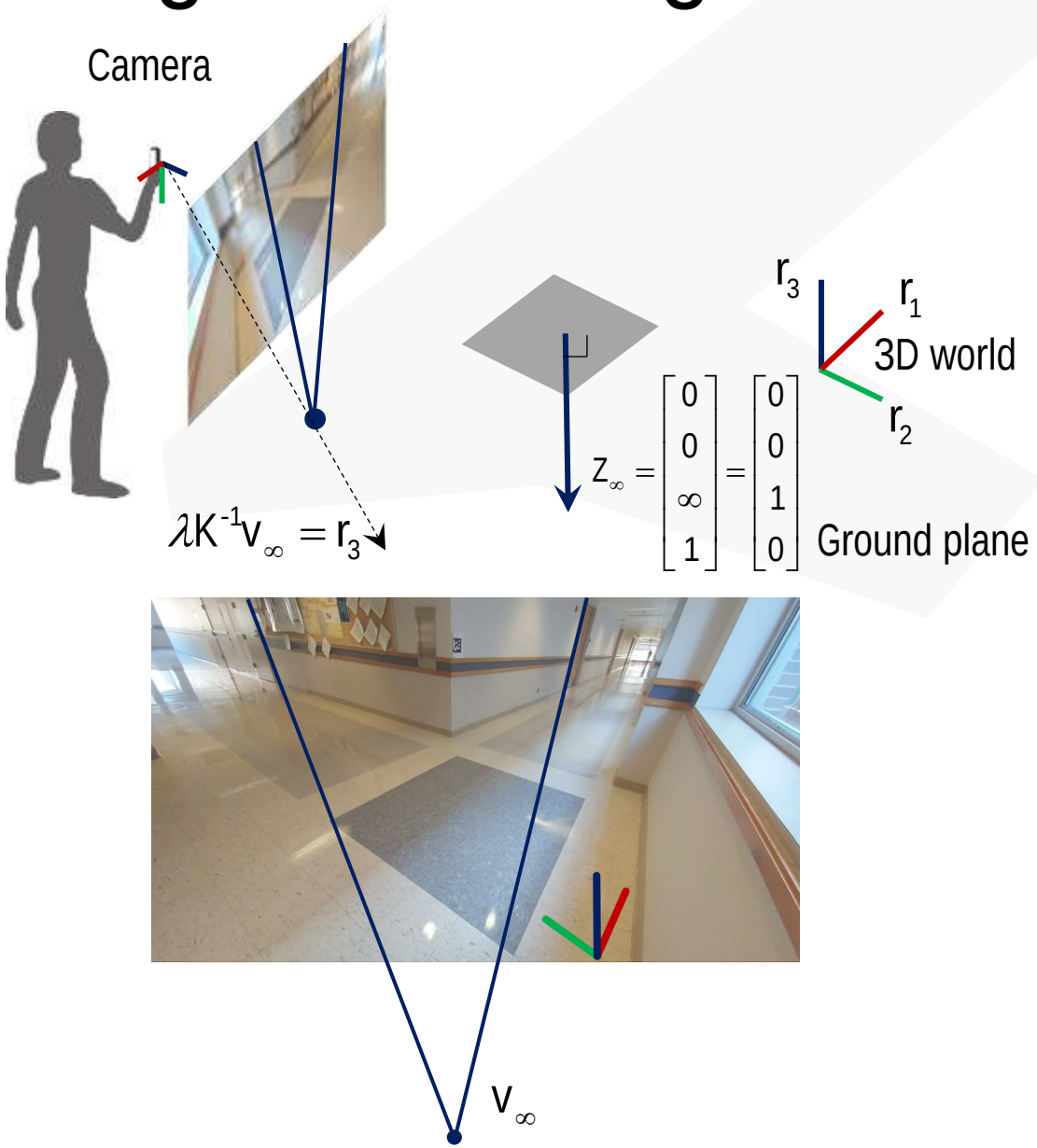
Single Vanishing Point



$$\lambda v_\infty = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\lambda v_\infty = K r_3$$

Single Vanishing Point



$$\lambda v_\infty = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

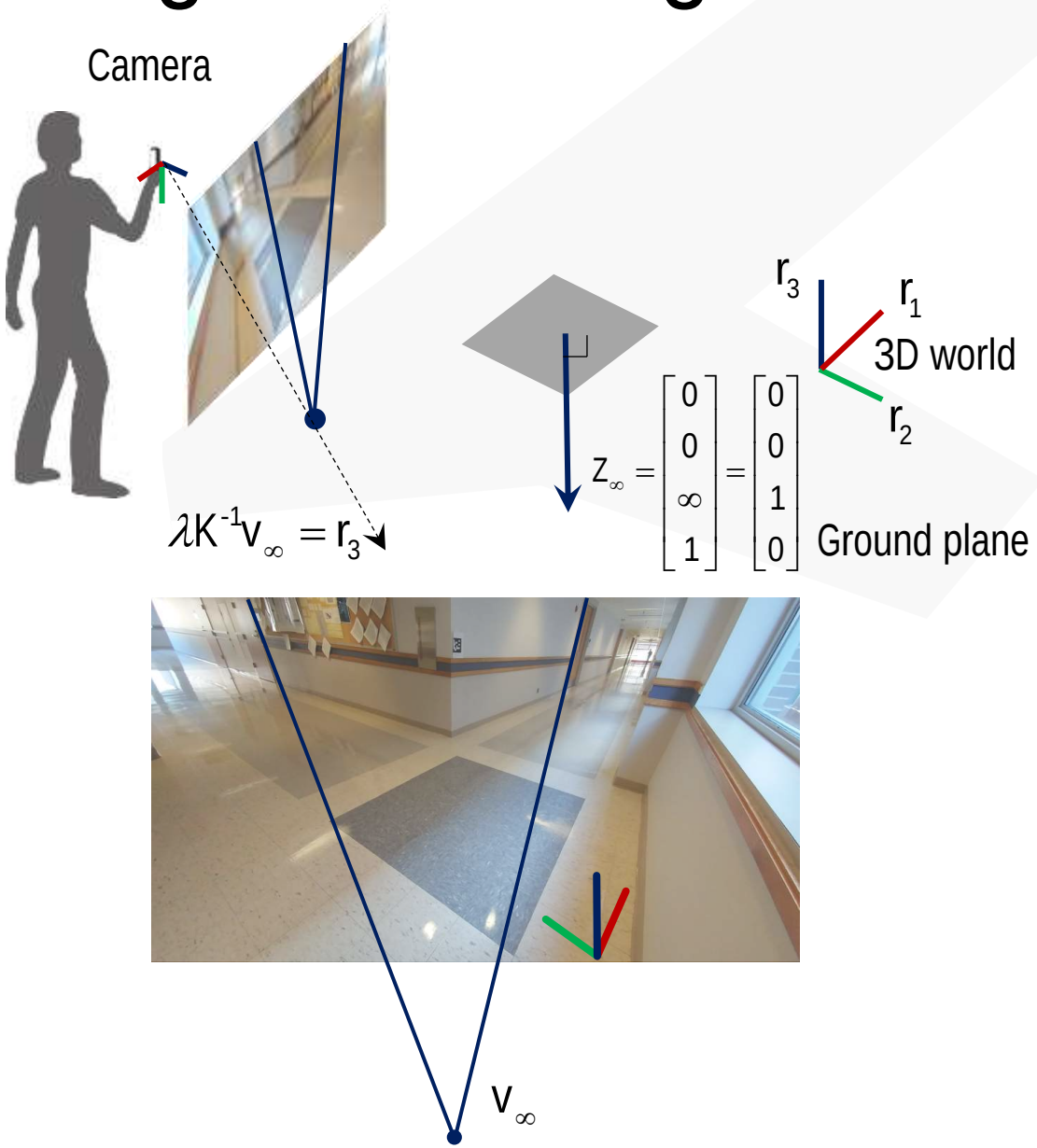
$$\lambda v_\infty = K r_3$$

$$\longrightarrow \quad r_3 = \frac{K^{-1} v_\infty}{\left\| K^{-1} v_\infty \right\|}$$

because r_3 is a unit vector.

Z vanishing point tells us about the surface normal of the ground plane

Single Vanishing Point



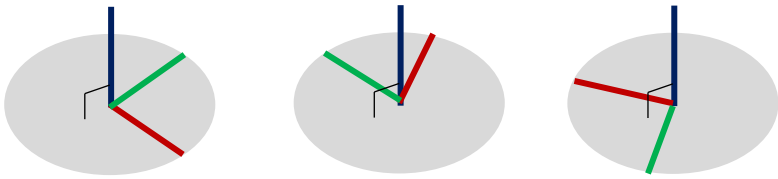
$$\lambda v_\infty = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\lambda v_\infty = K r_3$$

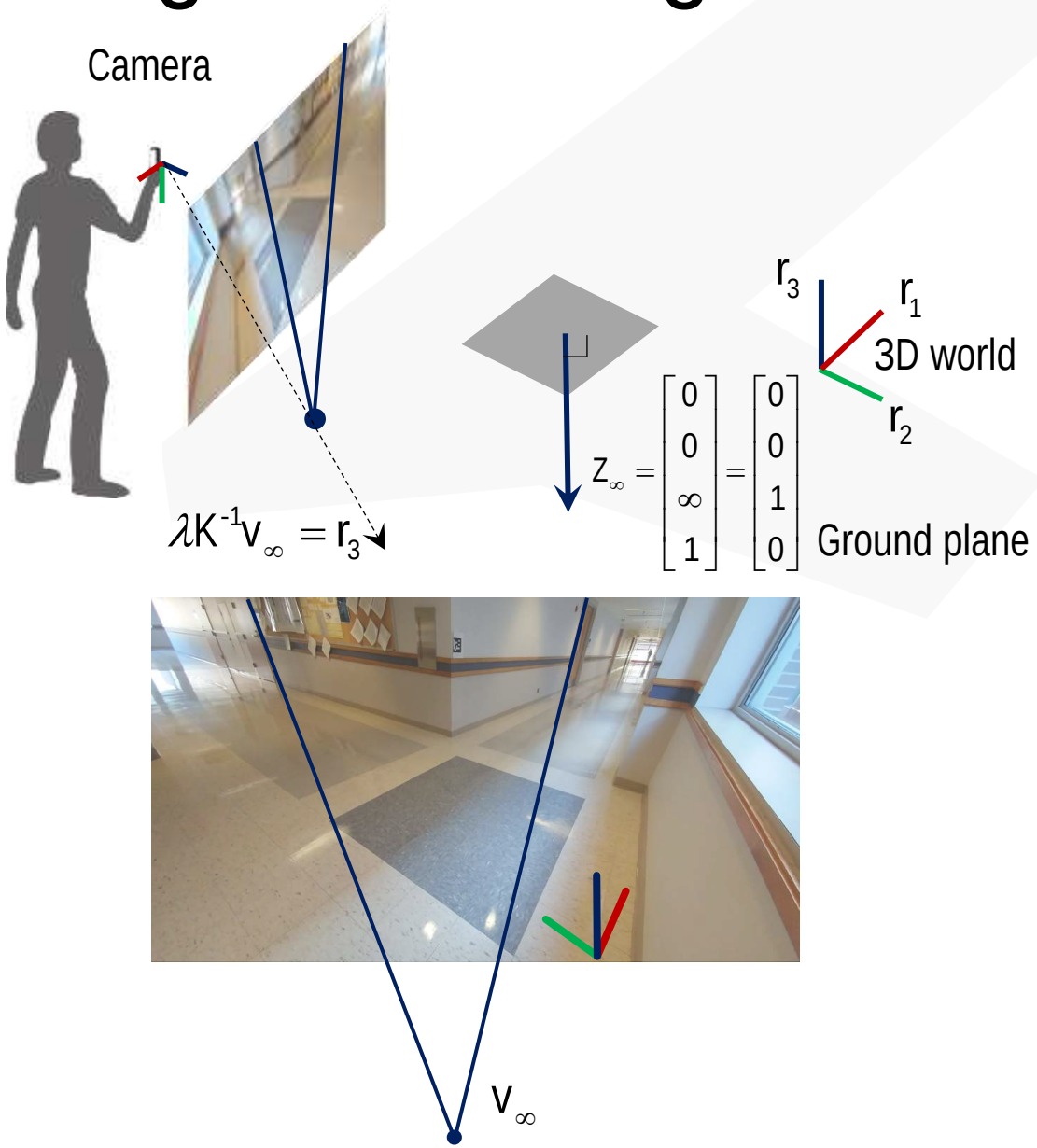
$$\longrightarrow \quad r_3 = \frac{K^{-1} v_\infty}{\left\| K^{-1} v_\infty \right\|}$$ because r_3 is a unit vector.

Z vanishing point tells us about the surface normal of the ground plane

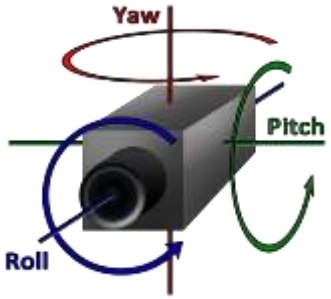
Rotation ambiguity



Single Vanishing Point

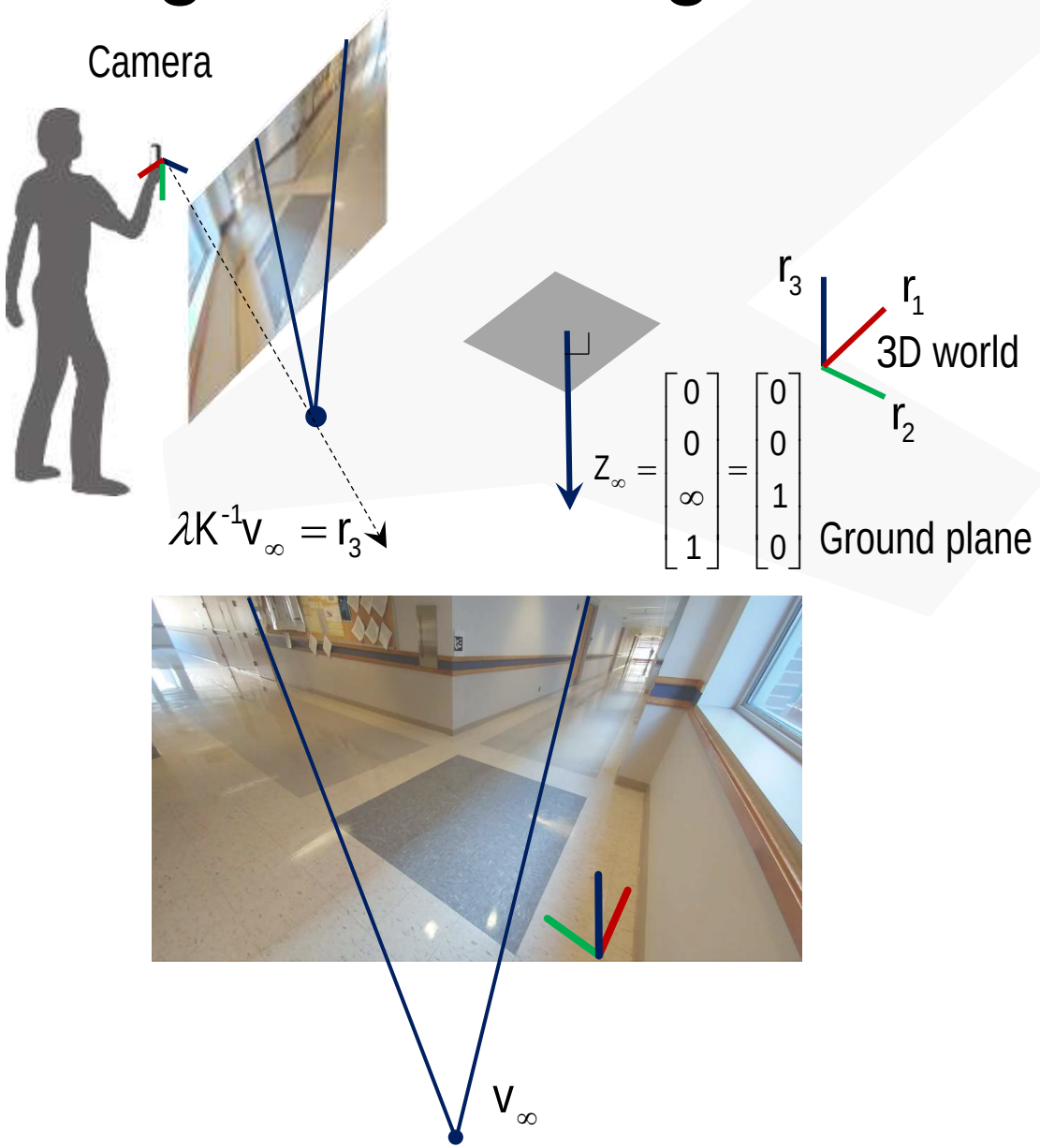


$$r_3 = \frac{K^{-1}v_\infty}{\left\| K^{-1}v_\infty \right\|}$$

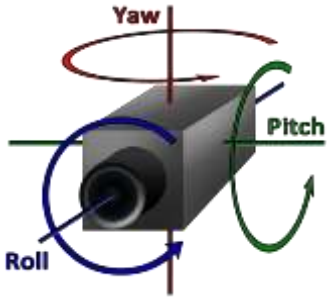


Roll and pitch angle can be computed by the ground plane.

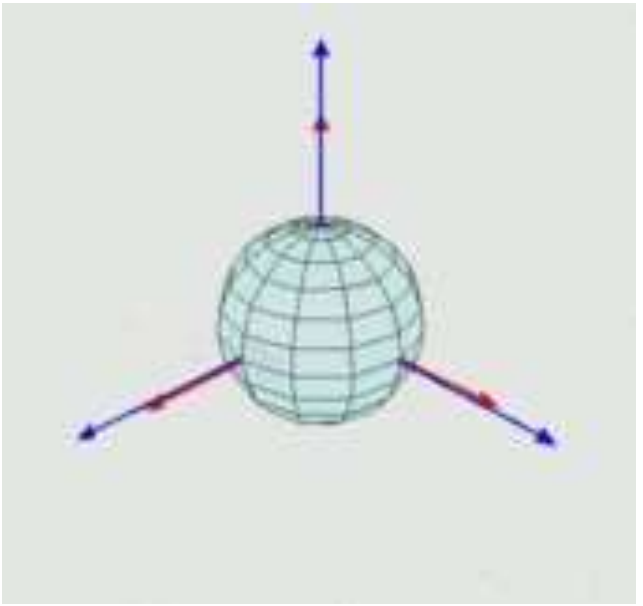
Single Vanishing Point



$$r_3 = \frac{K^{-1}v_\infty}{\|K^{-1}v_\infty\|}$$

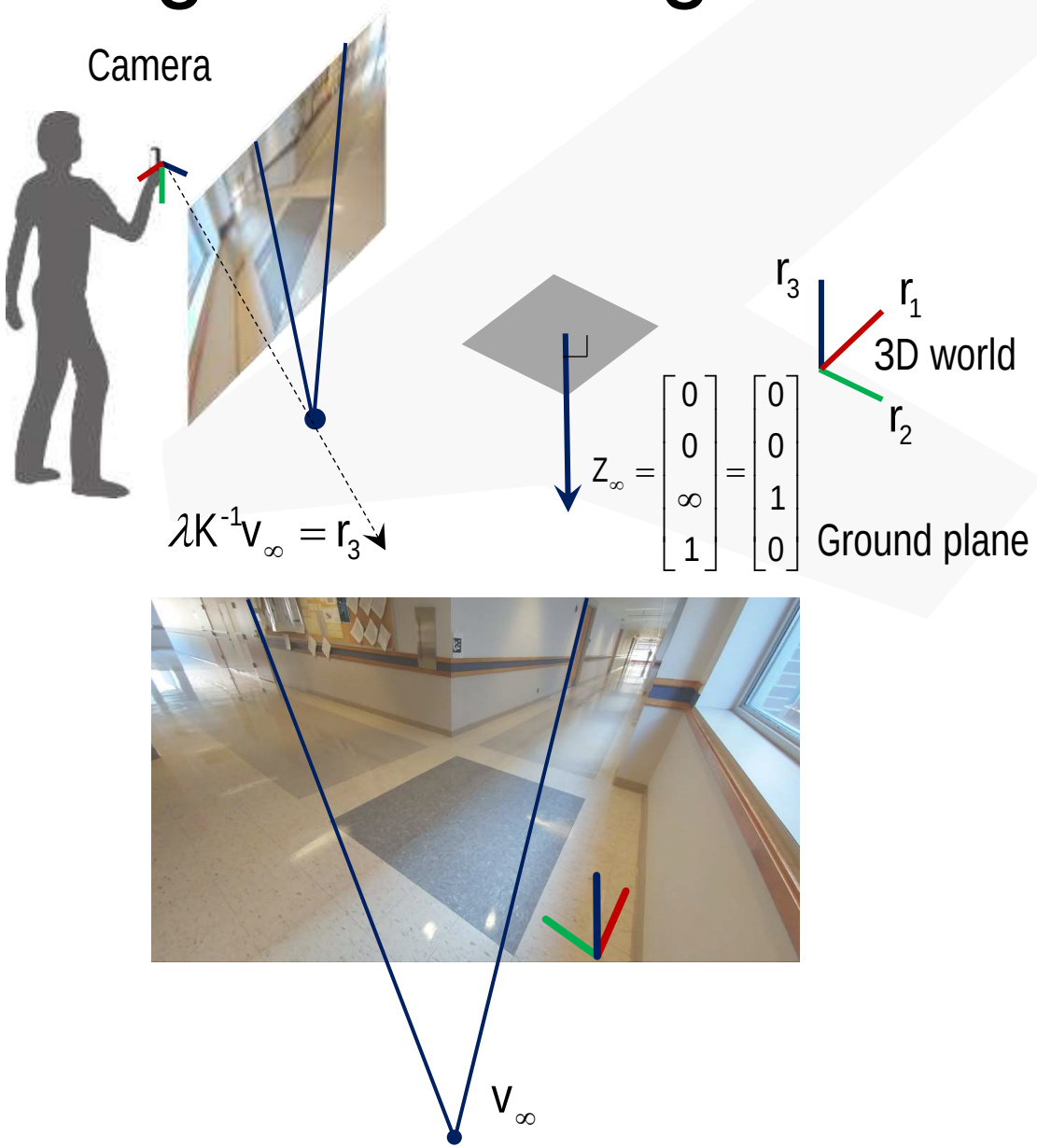


Roll and pitch angle can be computed by the ground plane.

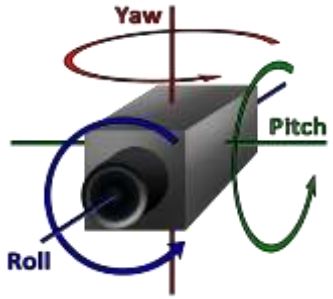


3D spatial rotation: a sequence of Euler angle rotation (roll/pitch/yaw).

Single Vanishing Point



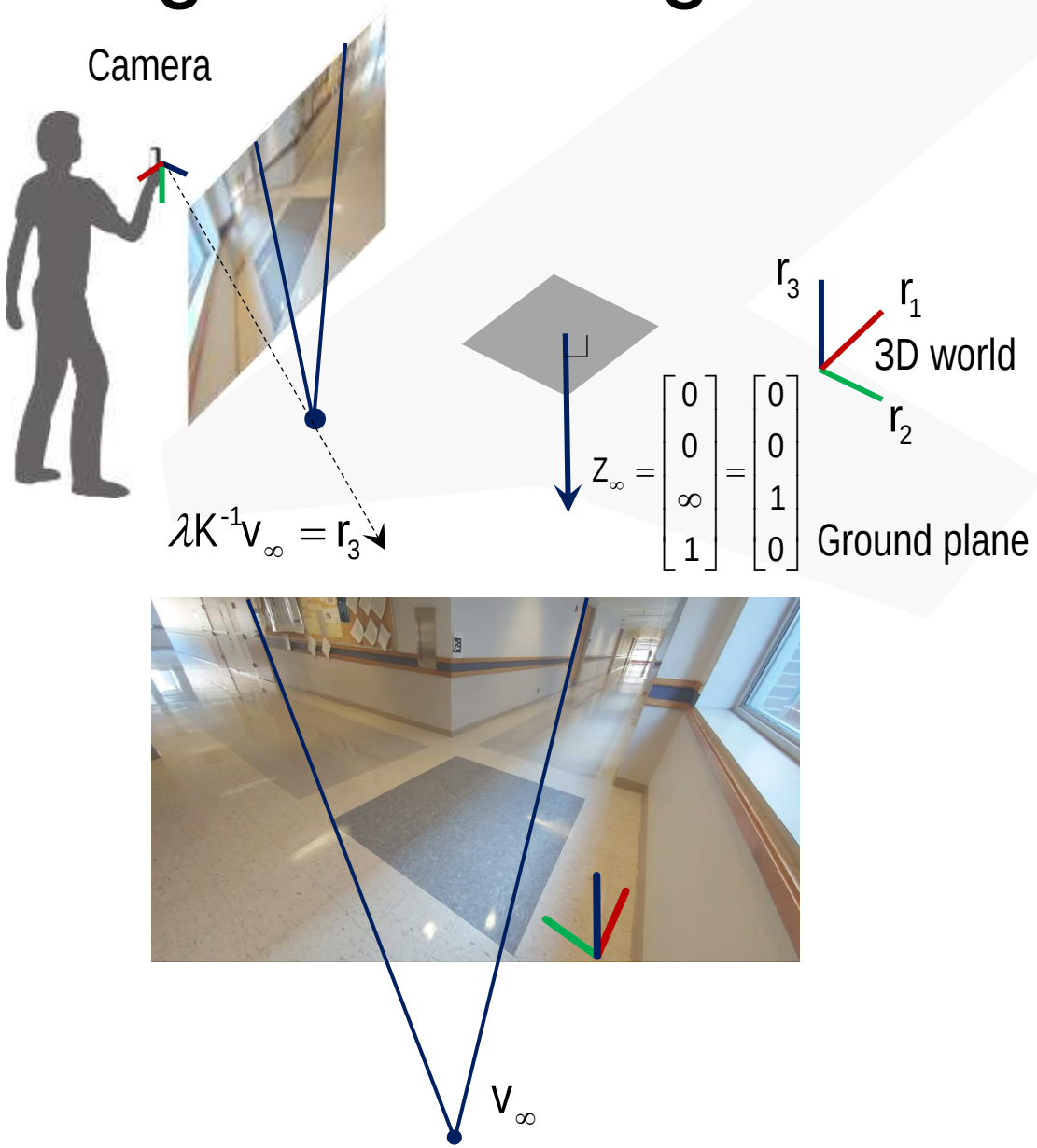
$$r_3 = \frac{K^{-1}v_\infty}{\left\|K^{-1}v_\infty\right\|}$$



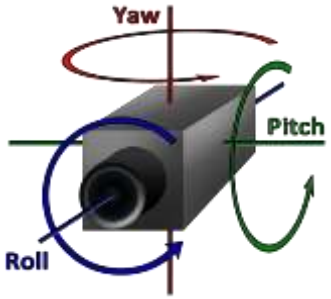
Roll and pitch angle can be computed by the ground plane.

Yaw: rotation about z axis: $R_{yaw} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Single Vanishing Point



$$r_3 = \frac{K^{-1}v_\infty}{\left\|K^{-1}v_\infty\right\|}$$

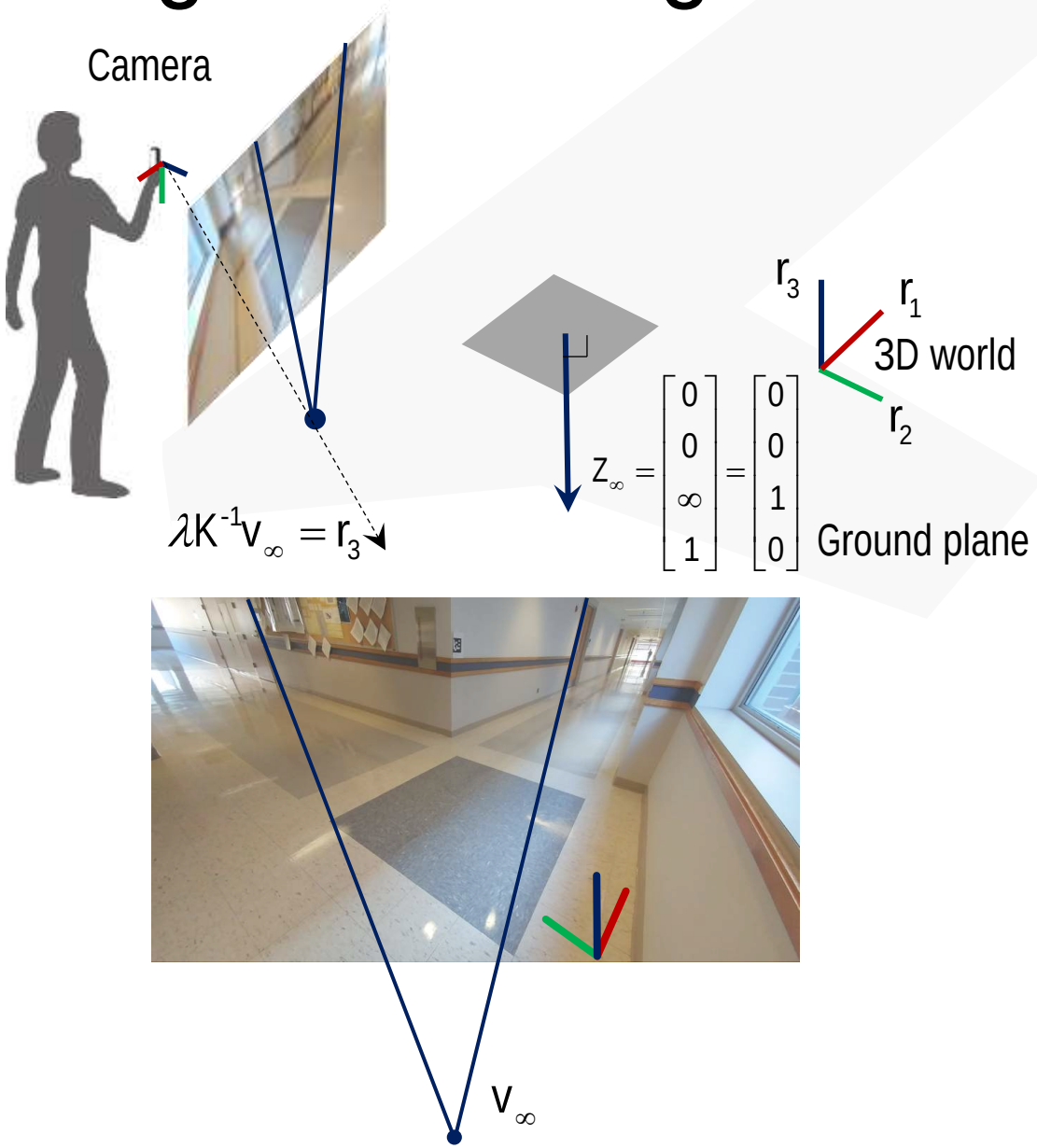


Roll and pitch angle can be computed by the ground plane.

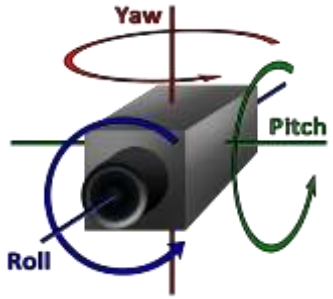
Yaw: rotation about z axis: $R_{yaw} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Pitch: rotation about y axis: $R_{pitch} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$

Single Vanishing Point



$$r_3 = \frac{K^{-1}v_\infty}{\left\|K^{-1}v_\infty\right\|}$$



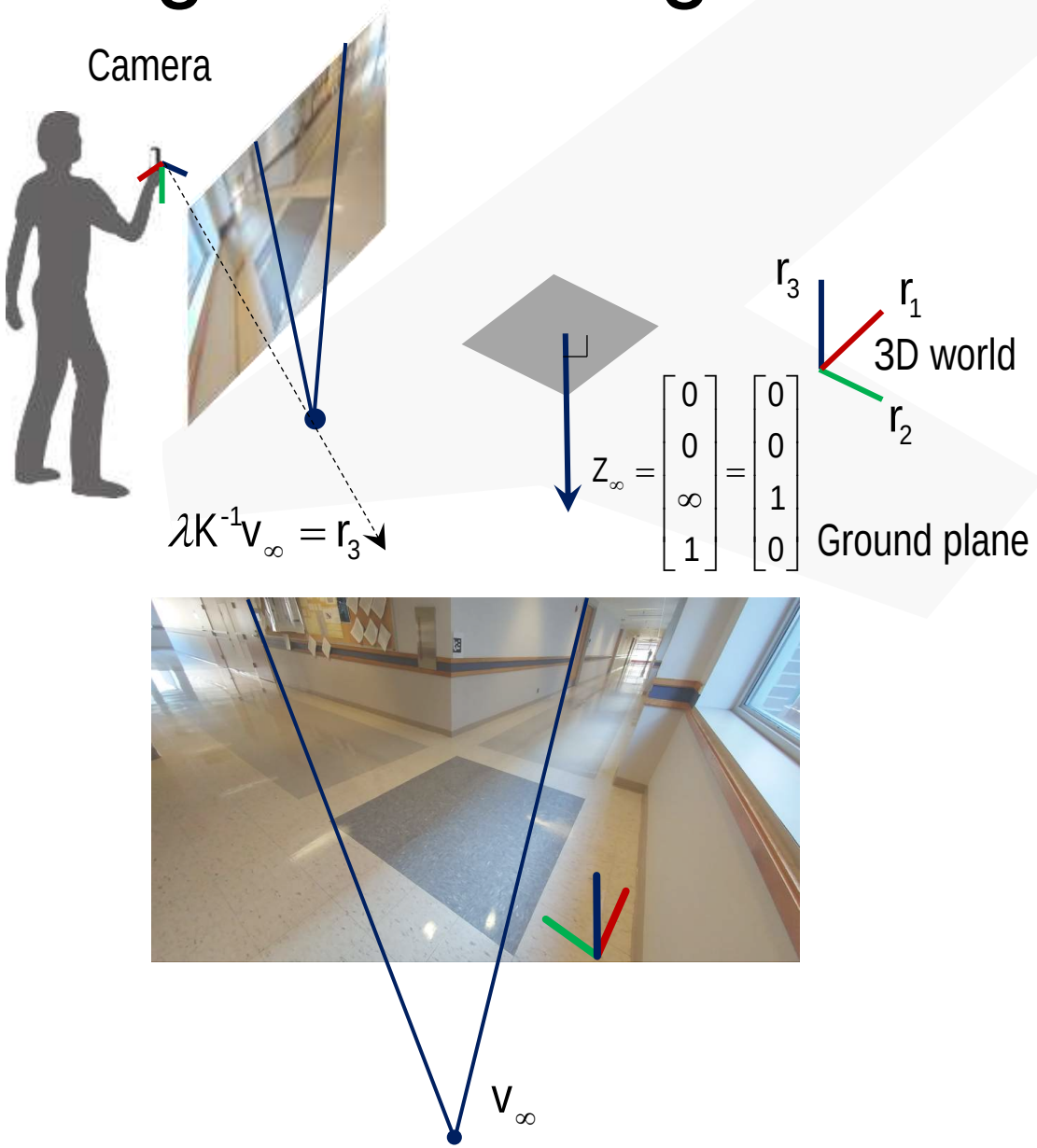
Roll and pitch angle can be computed by the ground plane.

Yaw: rotation about z axis: $R_{yaw} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

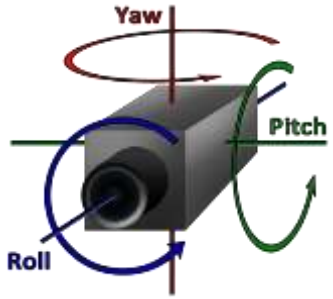
Pitch: rotation about y axis: $R_{pitch} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$

Roll: rotation about x axis: $R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$

Single Vanishing Point



$$r_3 = \frac{K^{-1}v_\infty}{\left\|K^{-1}v_\infty\right\|}$$



Roll and pitch angle can be computed by the ground plane.

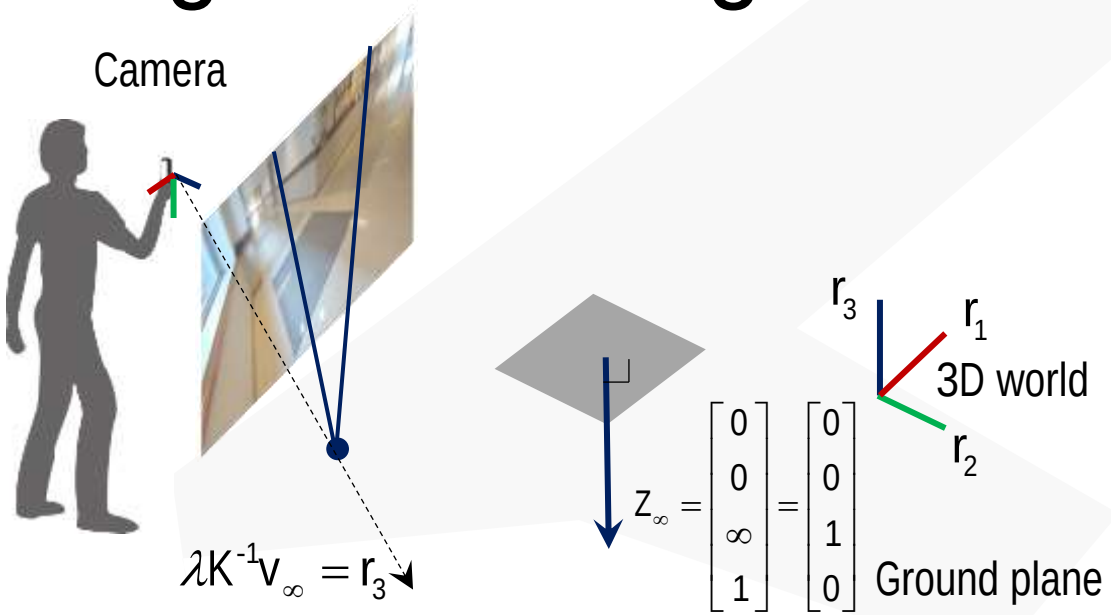
Yaw: rotation about z axis: $R_{yaw} = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Pitch: rotation about y axis: $R_{pitch} = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix}$

Roll: rotation about x axis: $R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix}$

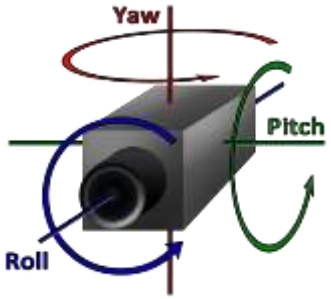
$$\rightarrow \quad \begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} = \left(R_{yaw}R_{pitch}R_{roll}\right)^T = \begin{bmatrix} \bullet & \bullet & -\sin\beta \\ \bullet & \bullet & \cos\beta\sin\gamma \\ \bullet & \bullet & \cos\beta\cos\gamma \end{bmatrix}$$

Single Vanishing Point



v_∞

$$r_3 = \frac{K^{-1} v_\infty}{\left\| K^{-1} v_\infty \right\|}$$



Roll and pitch angle can be computed by the ground plane.

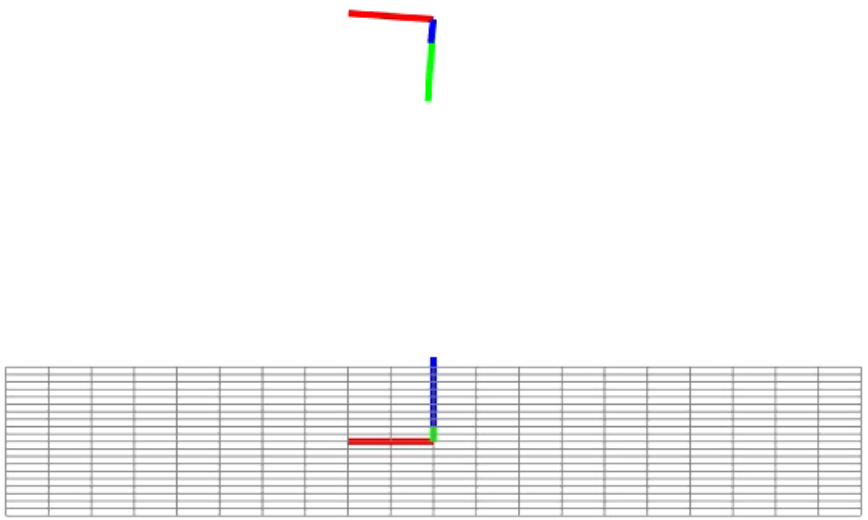
$$\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} = \left(R_{yaw} R_{pitch} R_{roll} \right)^T = \begin{bmatrix} \bullet & \bullet & -\sin\beta \\ \bullet & \bullet & \cos\beta \sin\gamma \\ \bullet & \bullet & \cos\beta \cos\gamma \end{bmatrix}$$

$\longrightarrow$

Pitch: $\beta = \tan^{-1}\left(-\frac{r_{31}}{\sqrt{r_{32}^2 + r_{33}^2}} \right)$

Roll: $\gamma = \tan^{-1} \frac{r_{32}}{r_{33}}$

Single Vanishing Point (Exercise)

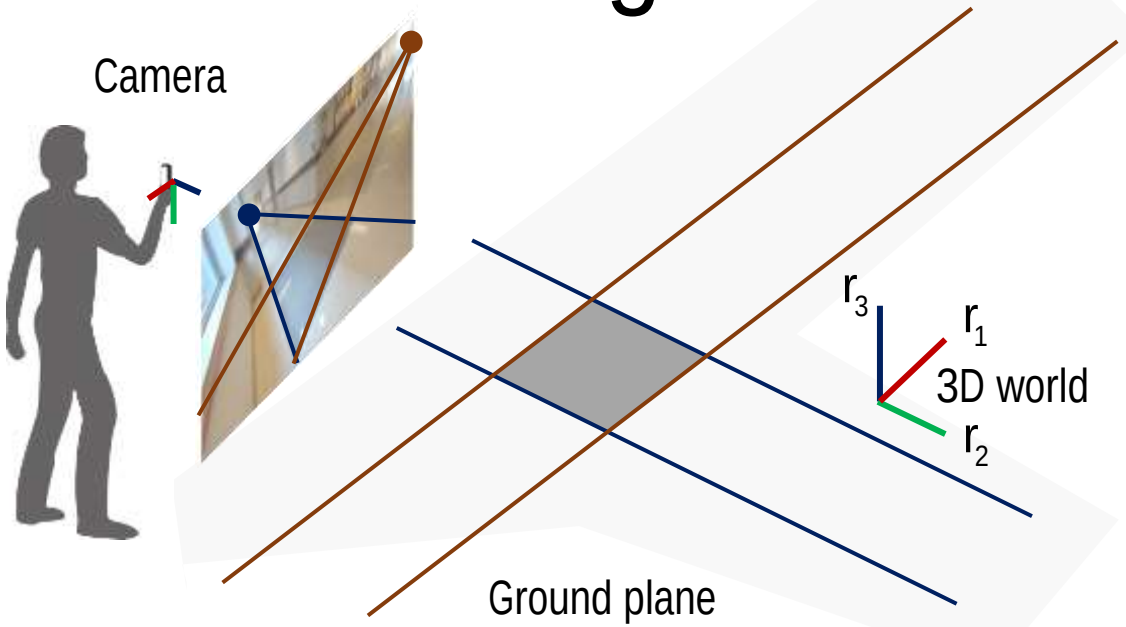


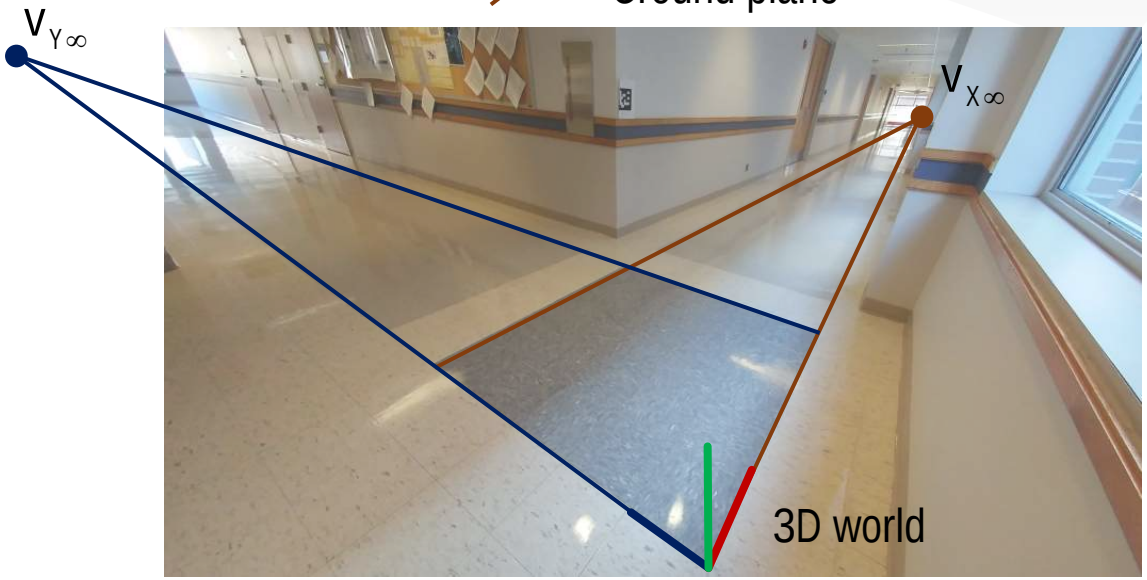
ComputeCameraUsingVanishingPoint.m

```
f = 1224;
K = [f 0 size(im,2)/2;
     0 f size(im,1)/2;
     0 0 1];

m1 = [2563;25;1];
m2 = [2439;545;1];
m3 = [571;25;1];
m4 = [723;498;1];

l1 = GetLineFromTwoPoints(m1,m2);
l2 = GetLineFromTwoPoints(m3,m4);

v1 = GetPointFromTwoLines(l1,l2);
v1 = v1/v1(3);

r3 = inv(K)*v1/norm(inv(K)*v1)
pitch = atan(-r3(1)/norm(r3(2:3)))
roll = atan(r3(2)/r3(3))
```

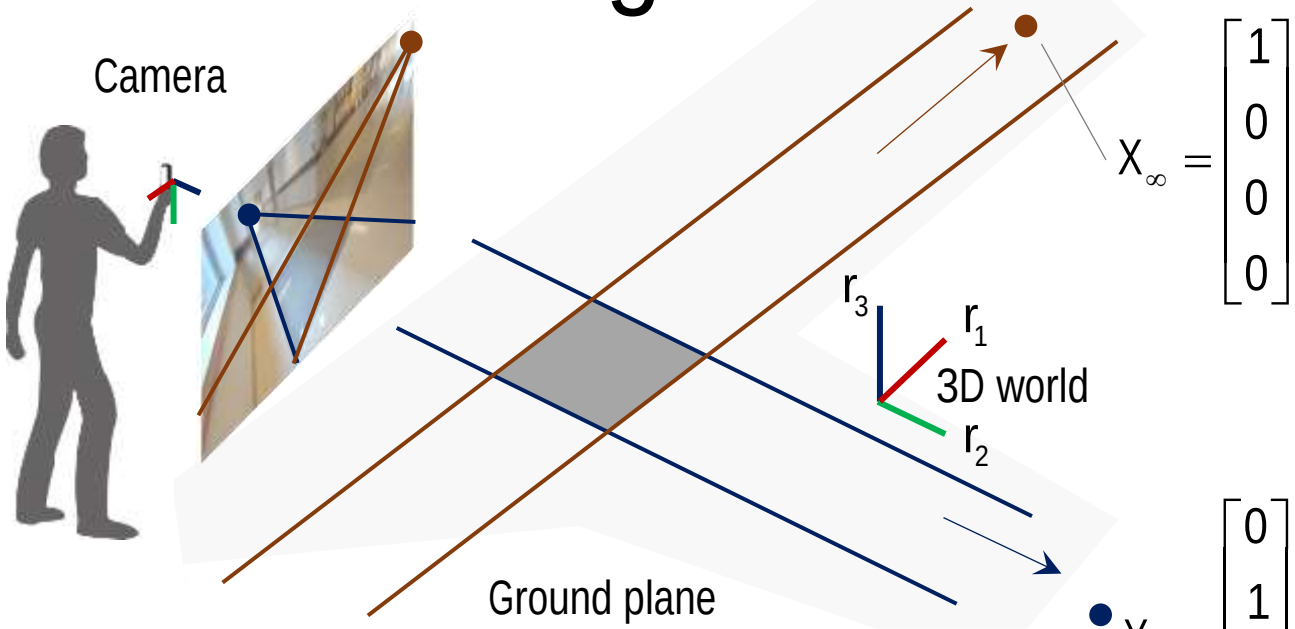
```
r3 =

  -0.0736
   0.8959
   0.4381


pitch =

   0.0736


roll =

   1.1160
```

Two Vanishing Points



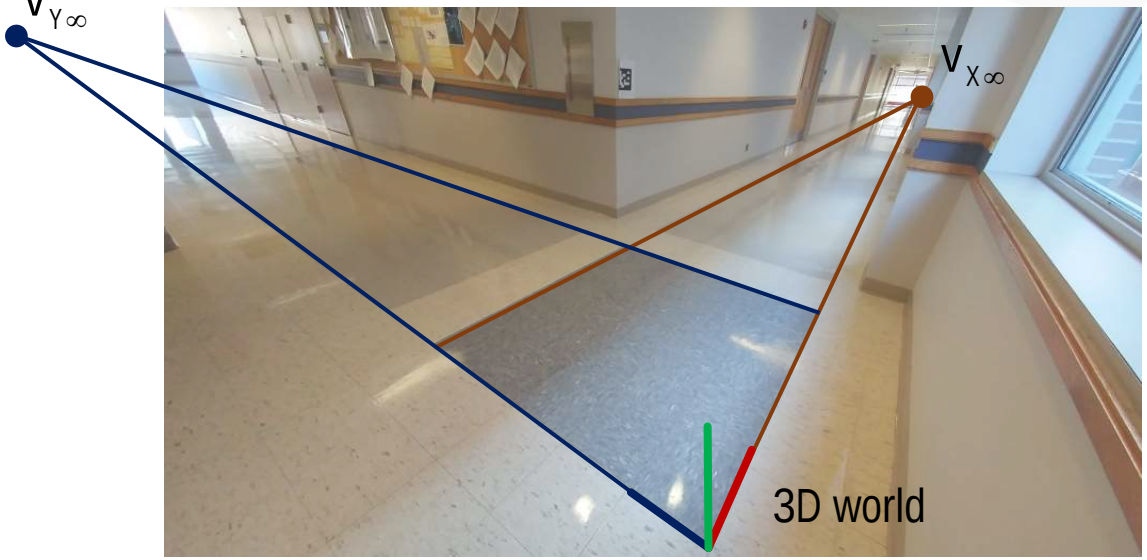


Two Vanishing Points

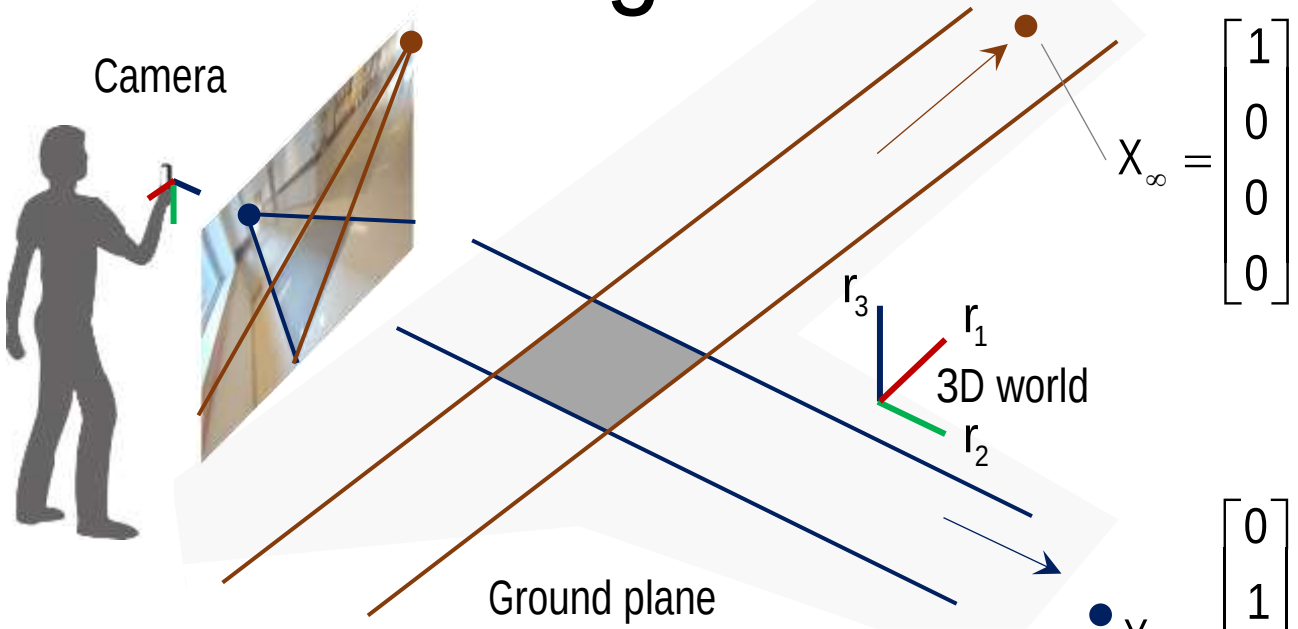


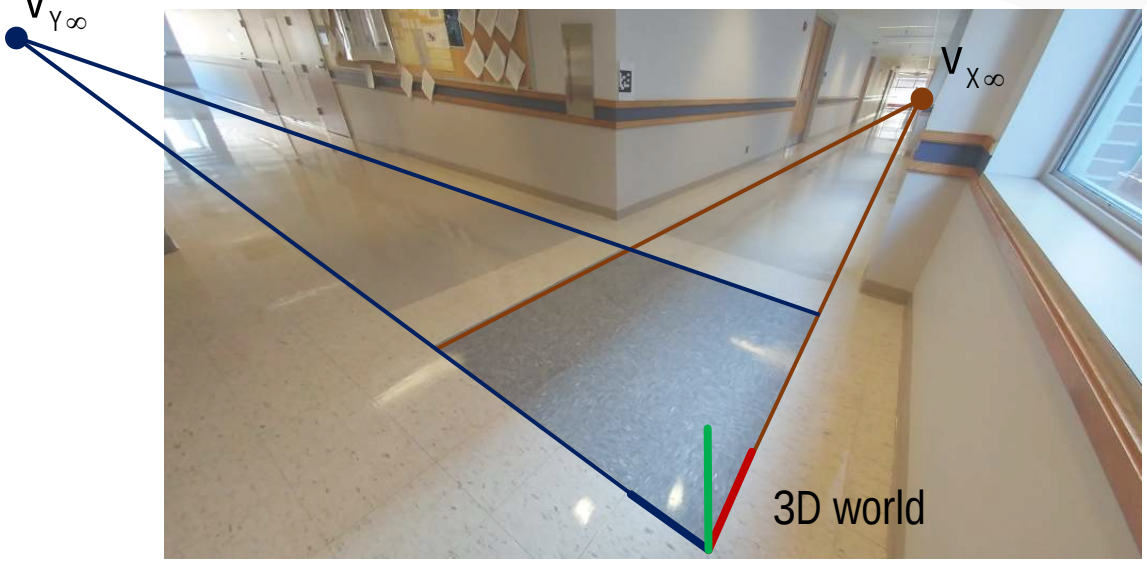
$$X_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$Y_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



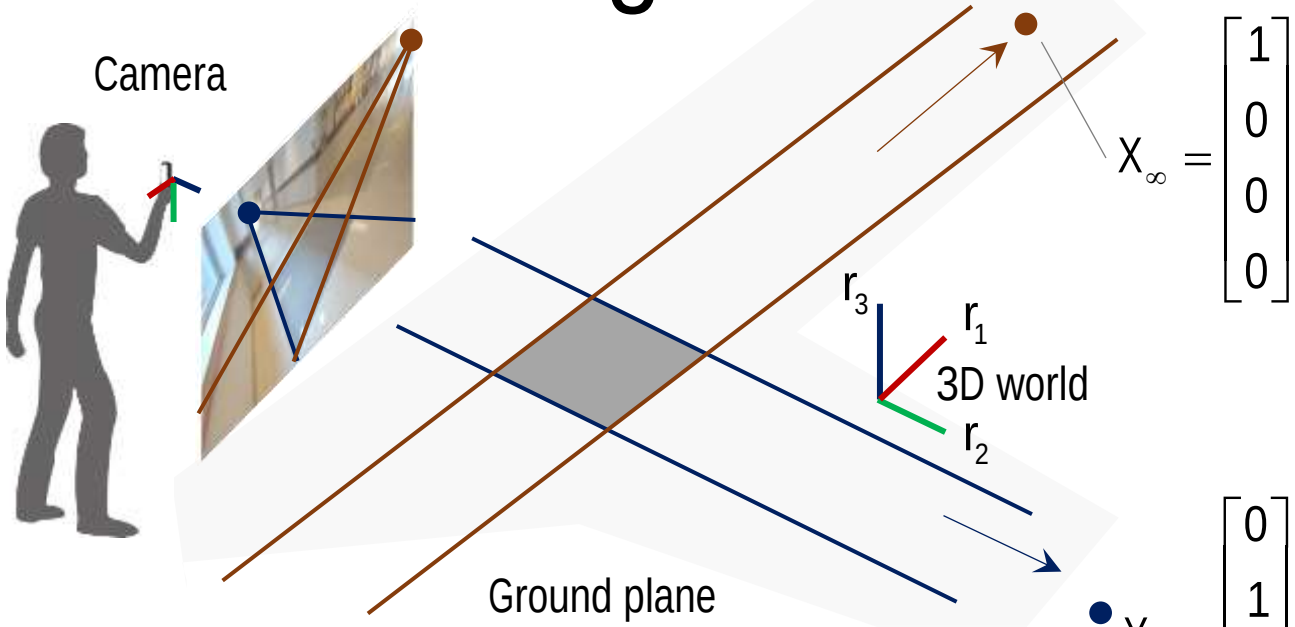
Two Vanishing Points

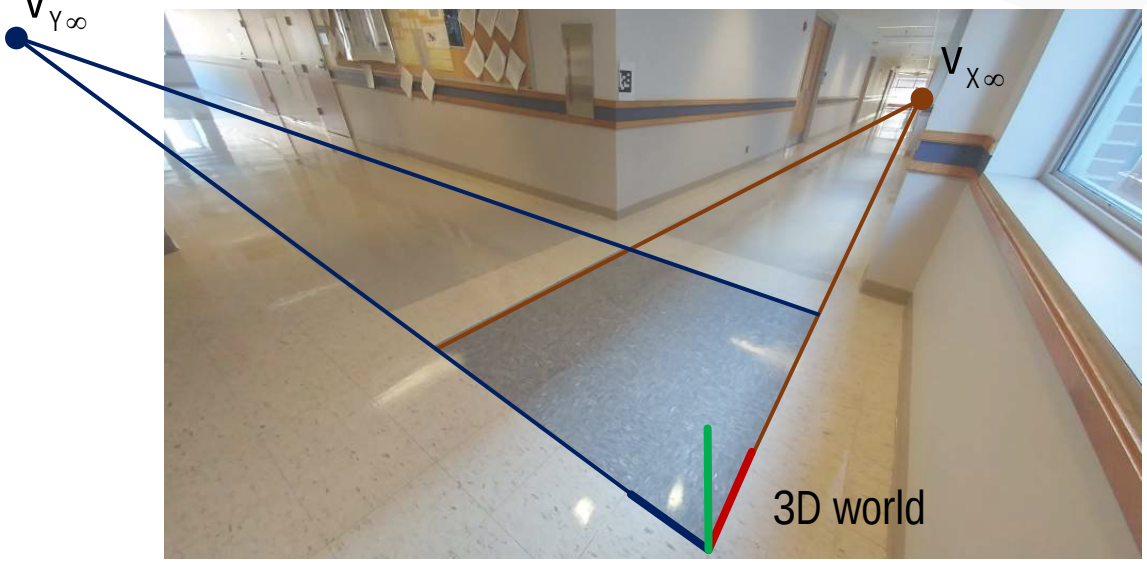




$$\lambda v_{X\infty} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} X_\infty$$
$$= K r_1$$

Two Vanishing Points

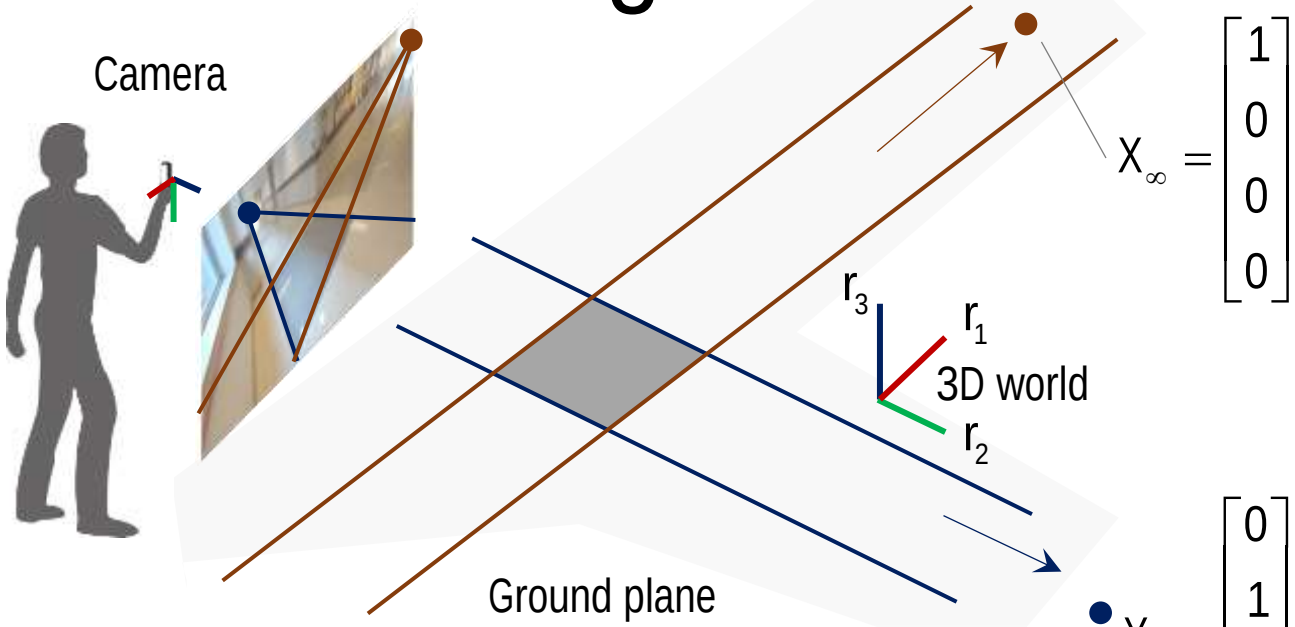




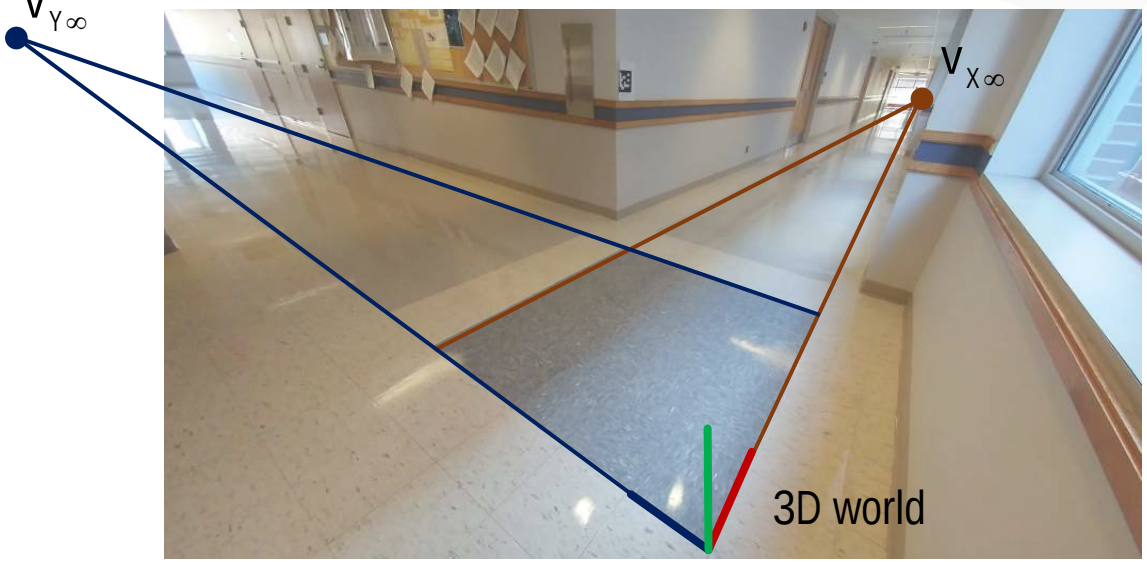
$$\lambda v_{X\infty} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} X_\infty$$
$$= K r_1$$

$$\lambda v_{Y\infty} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} Y_\infty$$
$$= K r_2$$

Two Vanishing Points



$$X_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad Y_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



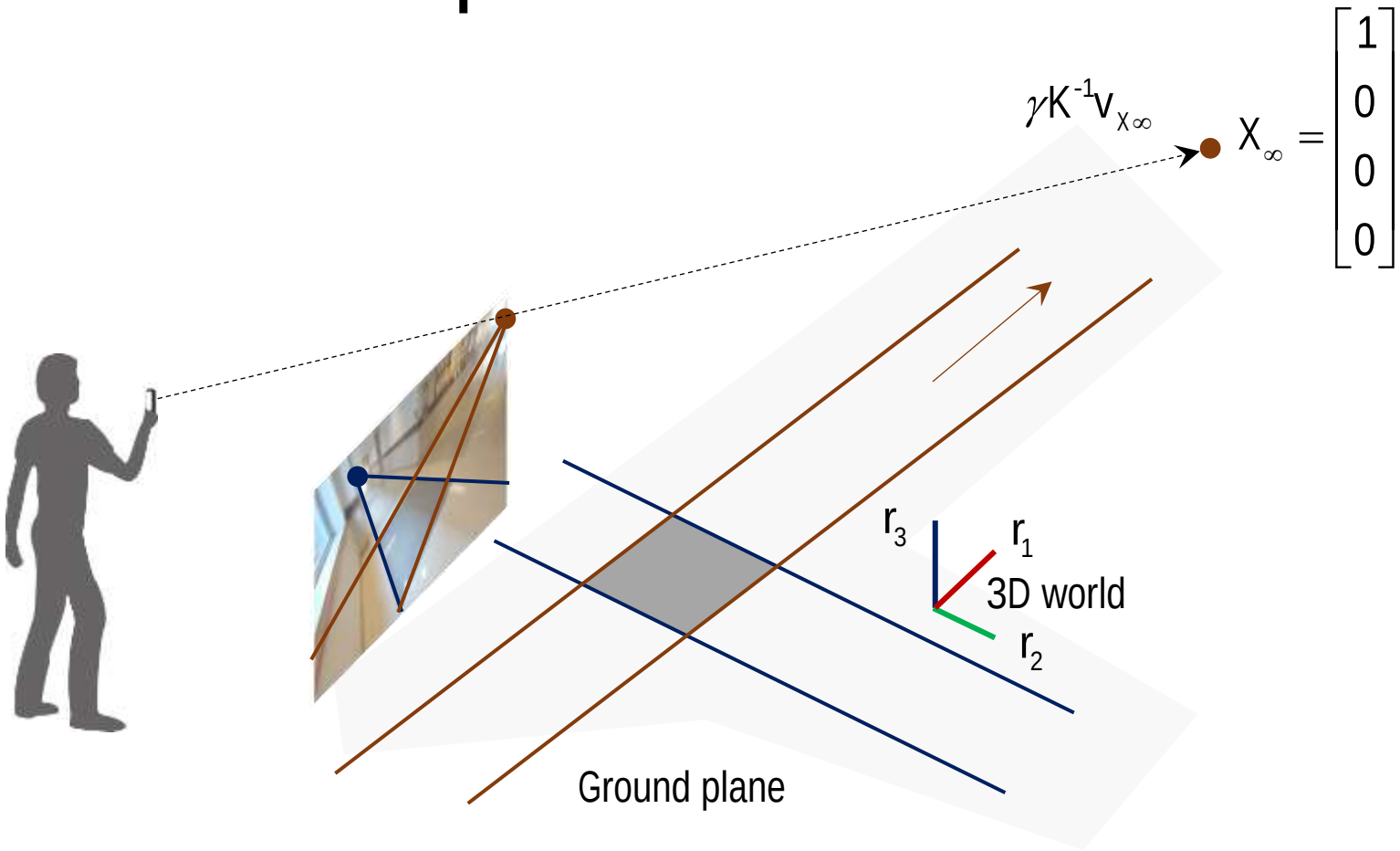
$$\lambda v_{X\infty} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} X_\infty = K r_1$$

$$\lambda v_{Y\infty} = K \begin{bmatrix} r_1 & r_2 & r_3 & t_W^C \end{bmatrix} Y_\infty = K r_2$$

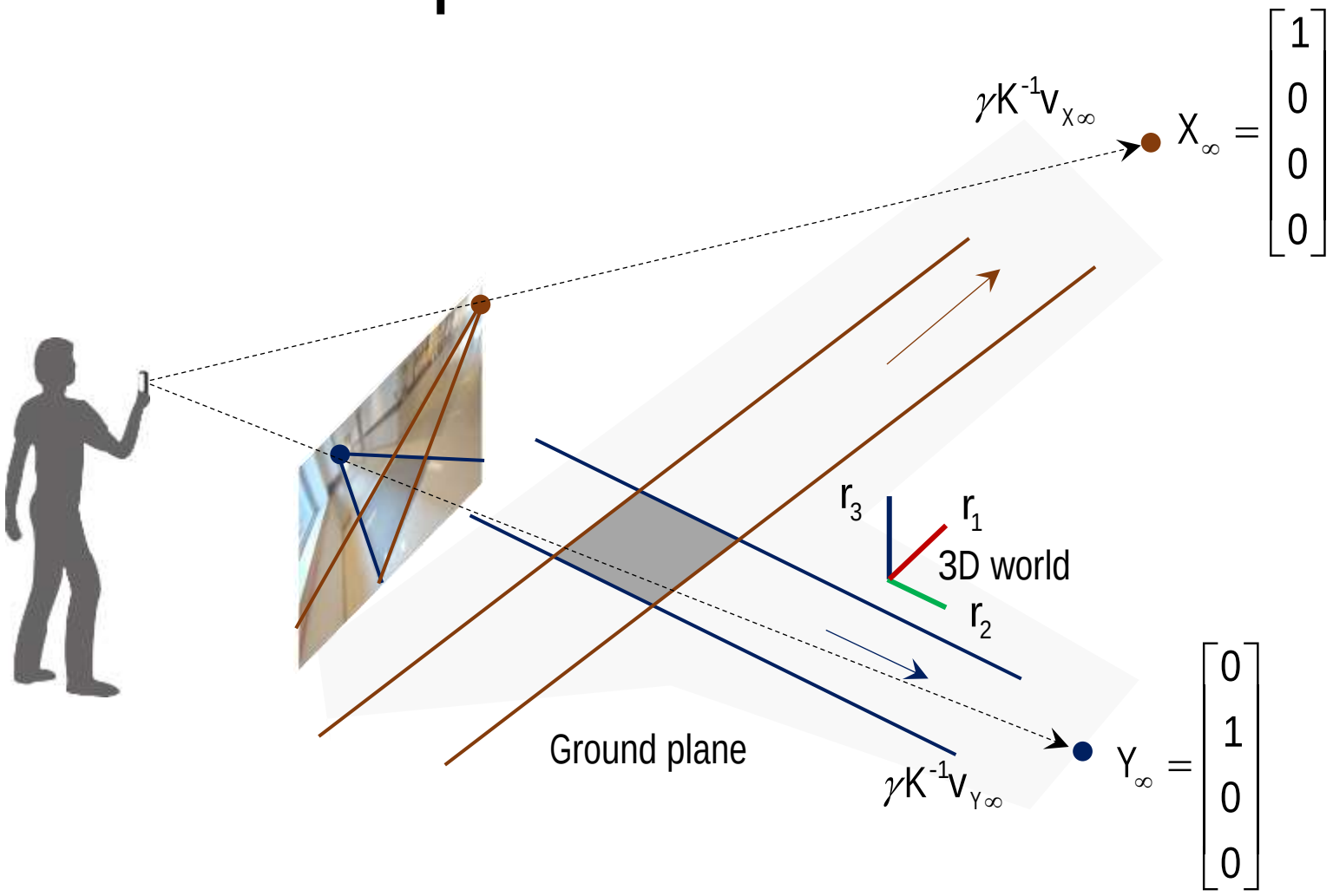
$$\rightarrow \quad r_1 = \frac{K^{-1} v_{X\infty}}{\left\| K^{-1} v_{X\infty} \right\|}, \quad r_2 = \frac{K^{-1} v_{Y\infty}}{\left\| K^{-1} v_{Y\infty} \right\|}$$

$r_3 = r_1 \times r_2$: Orthogonality constraint

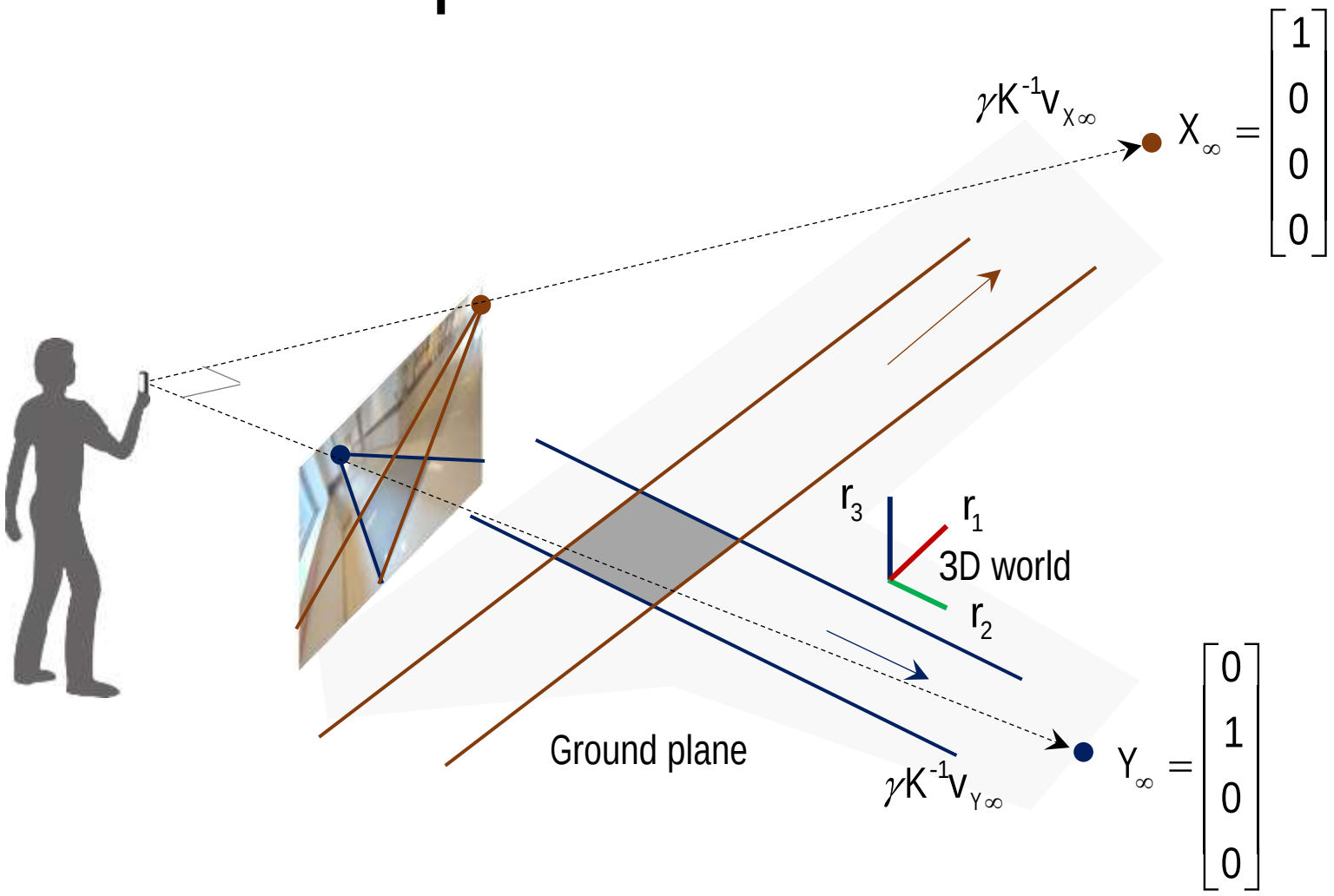
Geometric Interpretation



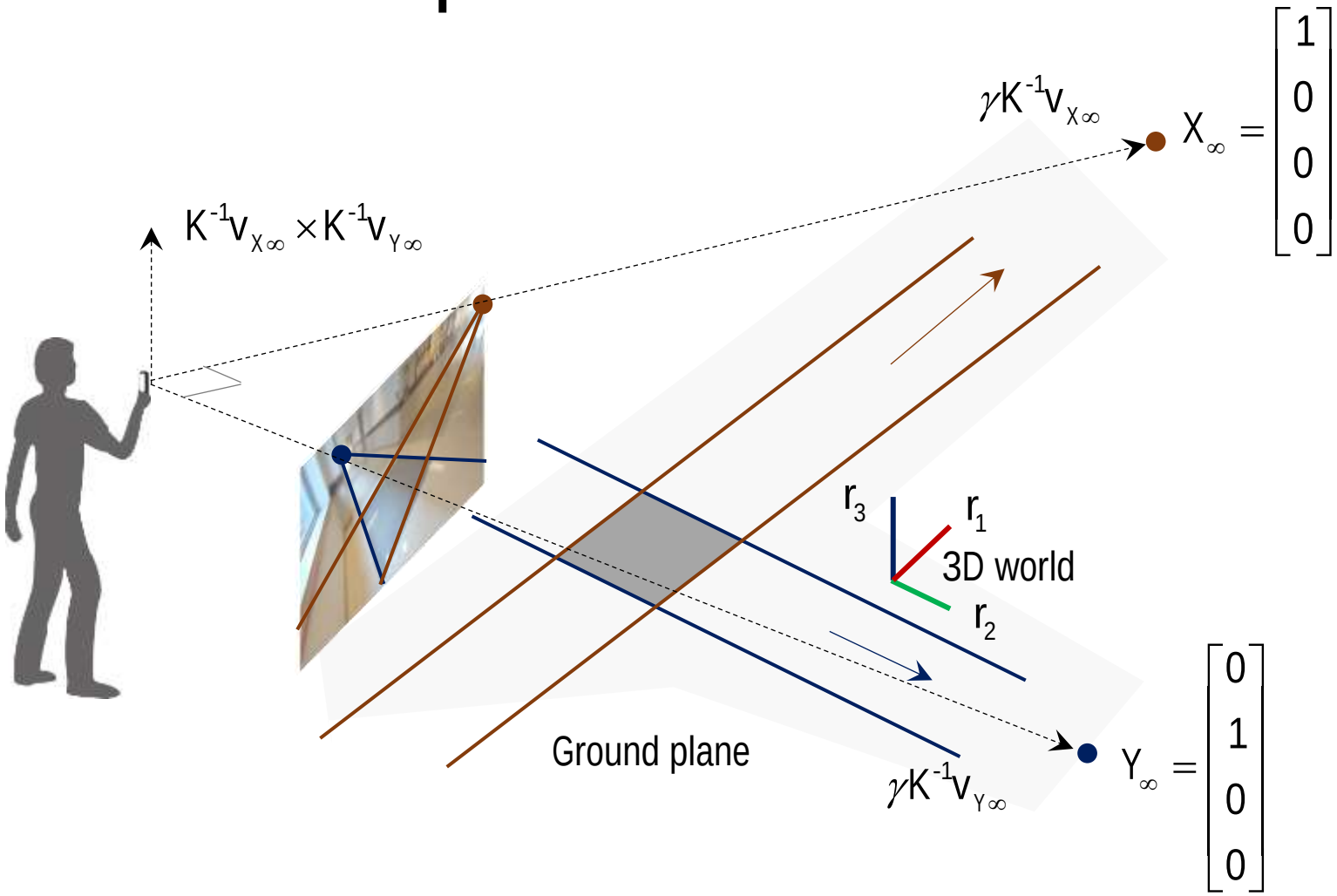
Geometric Interpretation



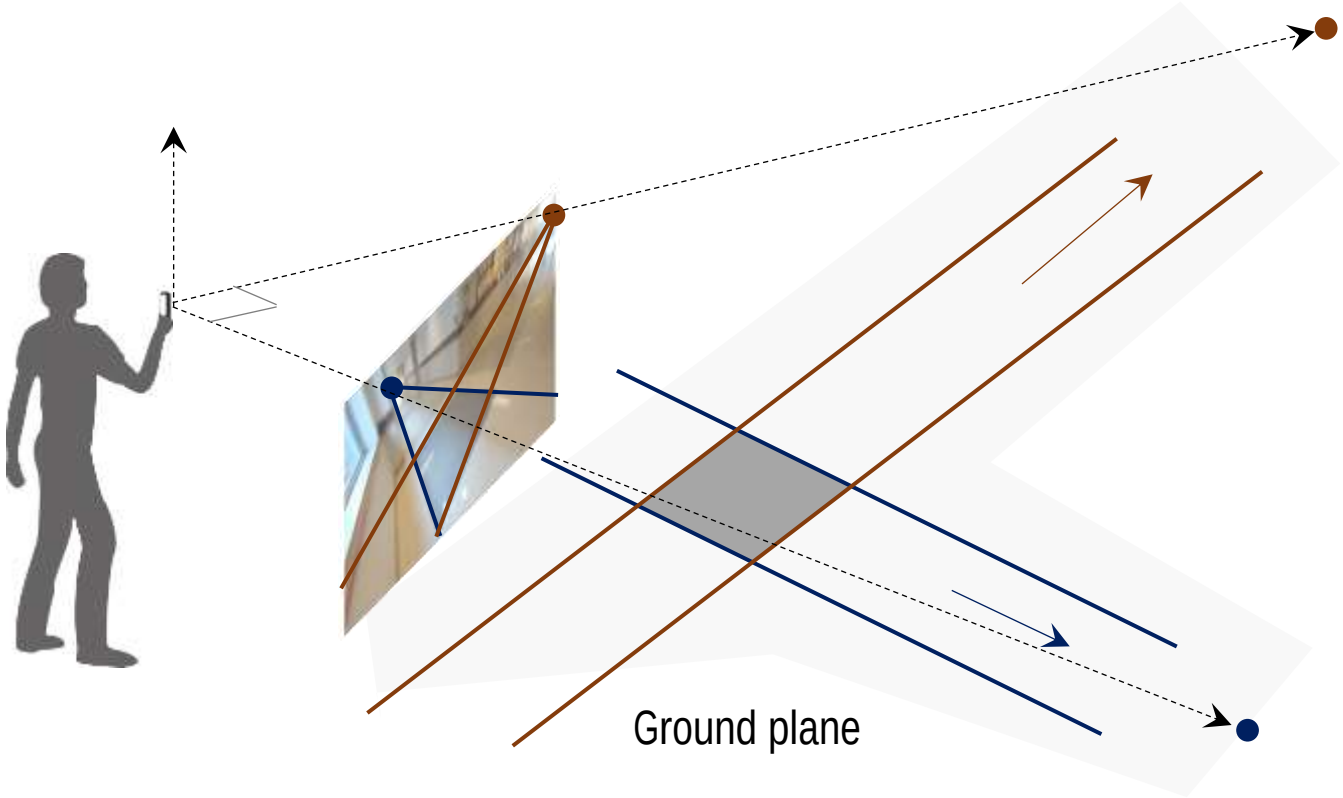
Geometric Interpretation



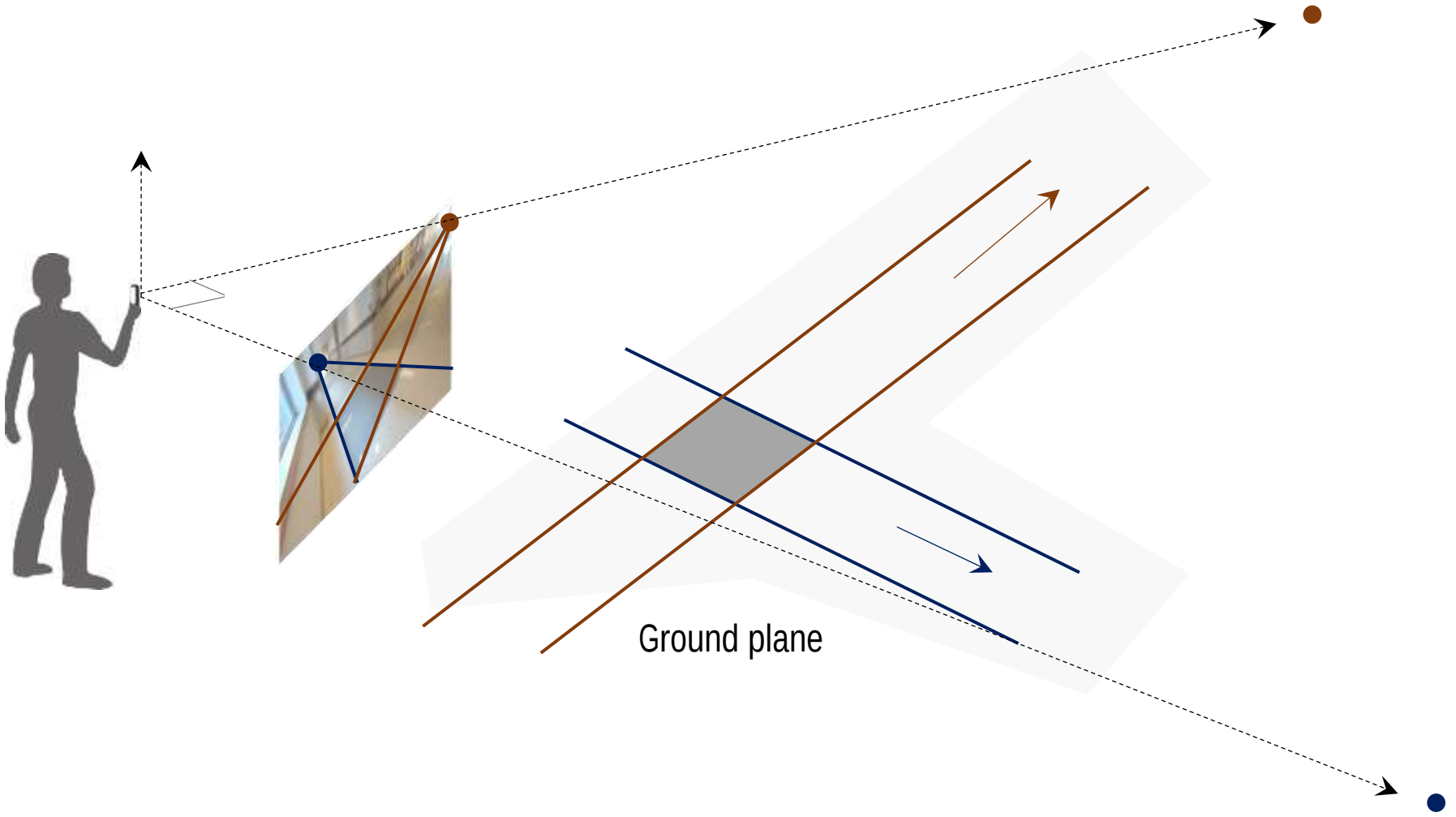
Geometric Interpretation



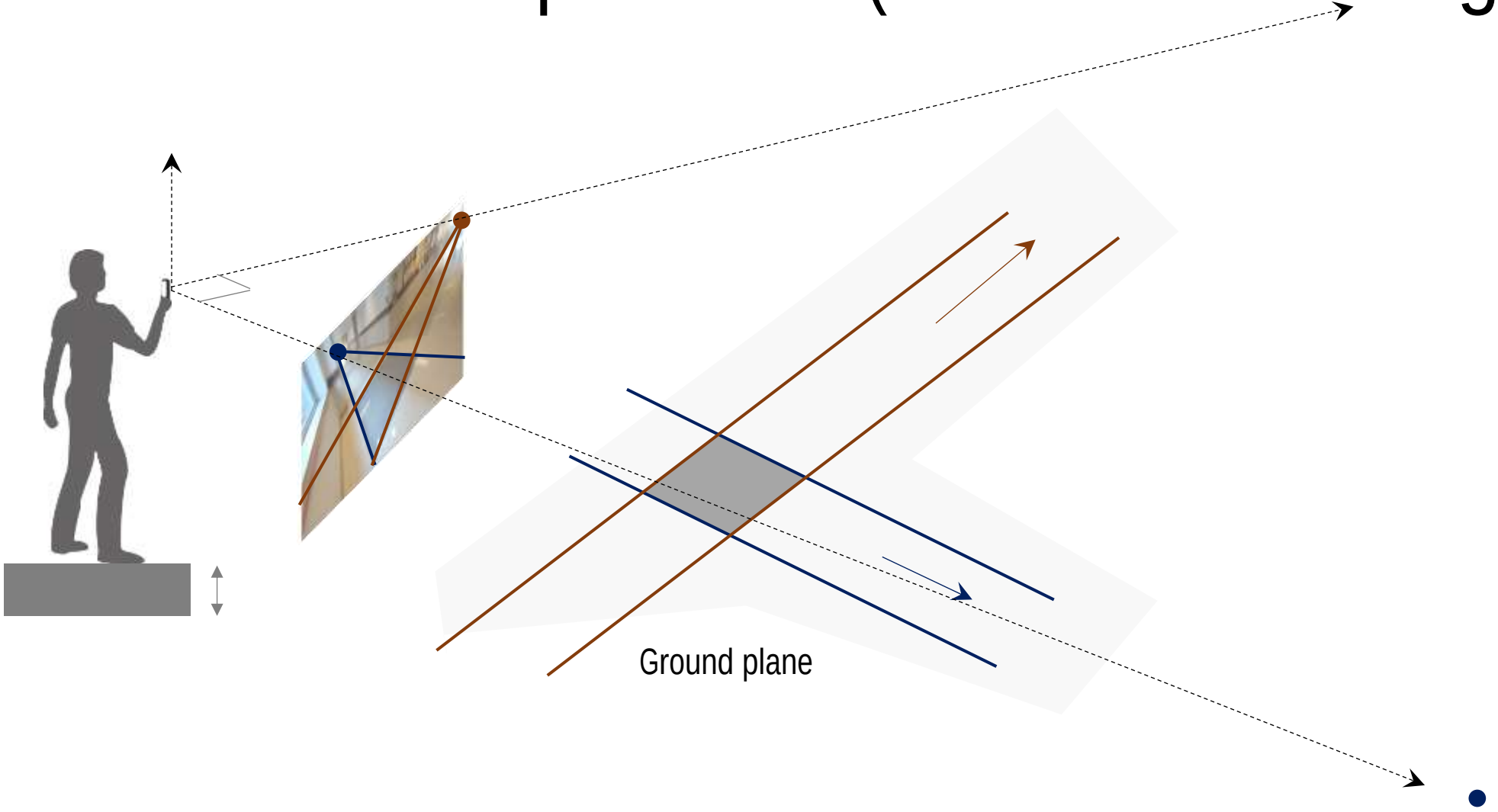
Geometric Interpretation (Translation Ambiguity)



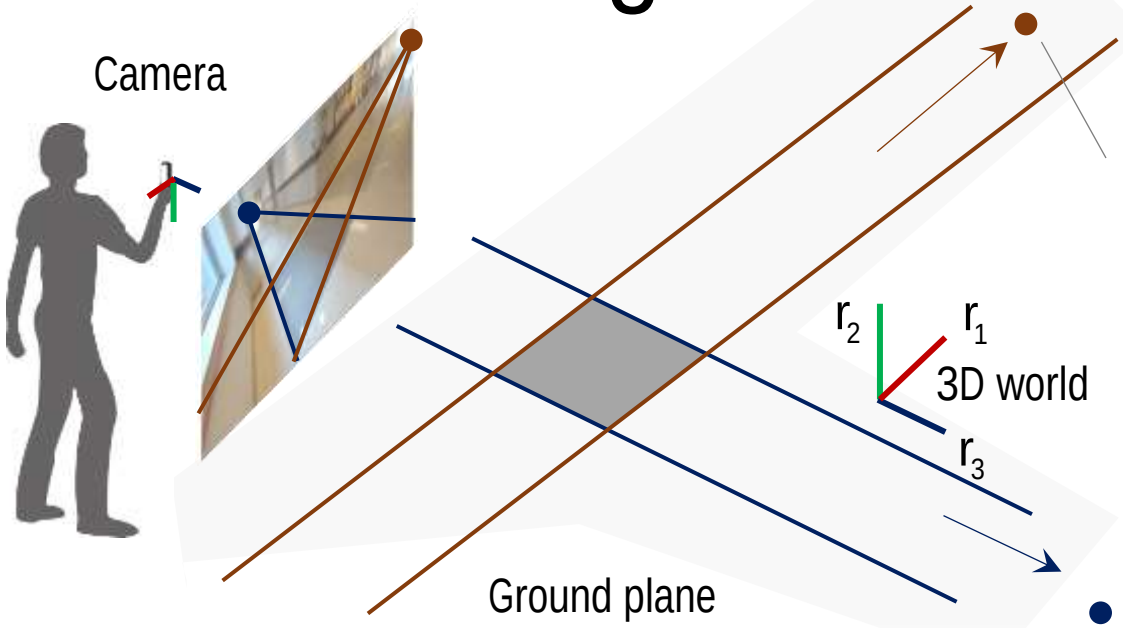
Geometric Interpretation (Translation Ambiguity)

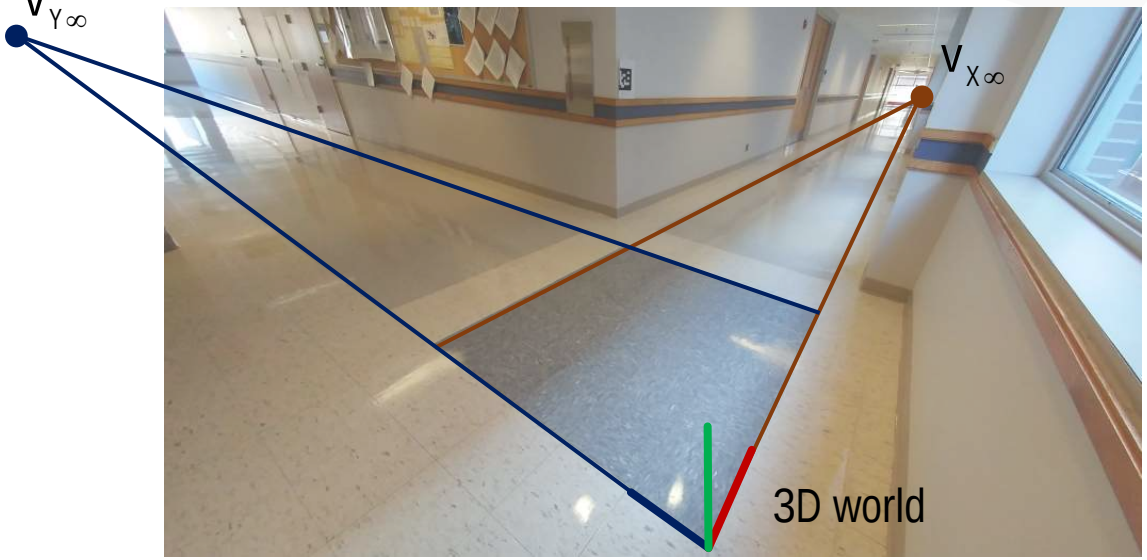


Geometric Interpretation (Translation Ambiguity)



Two Vanishing Points





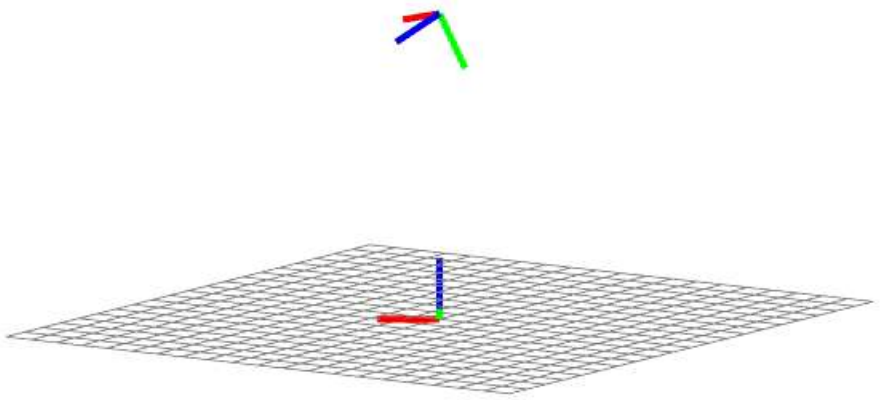
ComputeCameraUsingTwoVanishingPoints.m

```
f = 4000;
K = [f 0 size(im,2)/2;
     0 f size(im,1)/2;
     0 0 1];

l11 = GetLineFromTwoPoints(m11,m12);
l12 = GetLineFromTwoPoints(m13,m14);

l21 = GetLineFromTwoPoints(m21,m22);
l22 = GetLineFromTwoPoints(m23,m24);

v1 = GetPointFromTwoLines(l11,l12);
v2 = GetPointFromTwoLines(l21,l22);

r1 = inv(K)*v1/norm(inv(K)*v1);
r2 = inv(K)*v2/norm(inv(K)*v2);

r3 = Vec2Skew(r1)*r2;
```

Not orthogonal matrix!

```
R =
    0.2448   -0.5178    0.0424
   -0.1737   -0.1960   -0.6978
    0.9539    0.8327   -0.1379

det(R) =
    0.5077

R'*R =
    0.3299    0.0294   -0.2036
    0.0294    0.5555   -0.2327
   -0.2036   -0.2327    1.6224
```

Two Vanishing Points



ComputeCameraUsingTwoVanishingPoints.m

```
f = 1224;
K = [f 0 size(im,2)/2;
     0 f size(im,1)/2;
     0 0 1];

l11 = GetLineFromTwoPoints(m11,m12);
l12 = GetLineFromTwoPoints(m13,m14);

l21 = GetLineFromTwoPoints(m21,m22);
l22 = GetLineFromTwoPoints(m23,m24);

v1 = GetPointFromTwoLines(l11,l12);
v2 = GetPointFromTwoLines(l21,l22);

r1 = inv(K)*v1/norm(inv(K)*v1);
r2 = inv(K)*v2/norm(inv(K)*v2);

r3 = Vec2Skew(r1)*r2;
```

Change focal length

Orthogonal matrix!

```
R =
    0.5846   -0.8496    0.0508
   -0.4149   -0.3216   -0.8367
    0.6972    0.4180   -0.5405

det(R) =
    0.9948

R'*R =
    1.0662   -0.0118    0.0250
   -0.0118    0.9757    0.0285
    0.0250    0.0285    0.9530
```