Where am I?
Using Vanishing Points

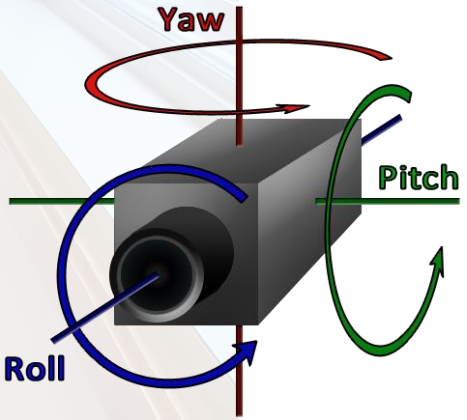
Vanishing point

Vanishing line for horizon

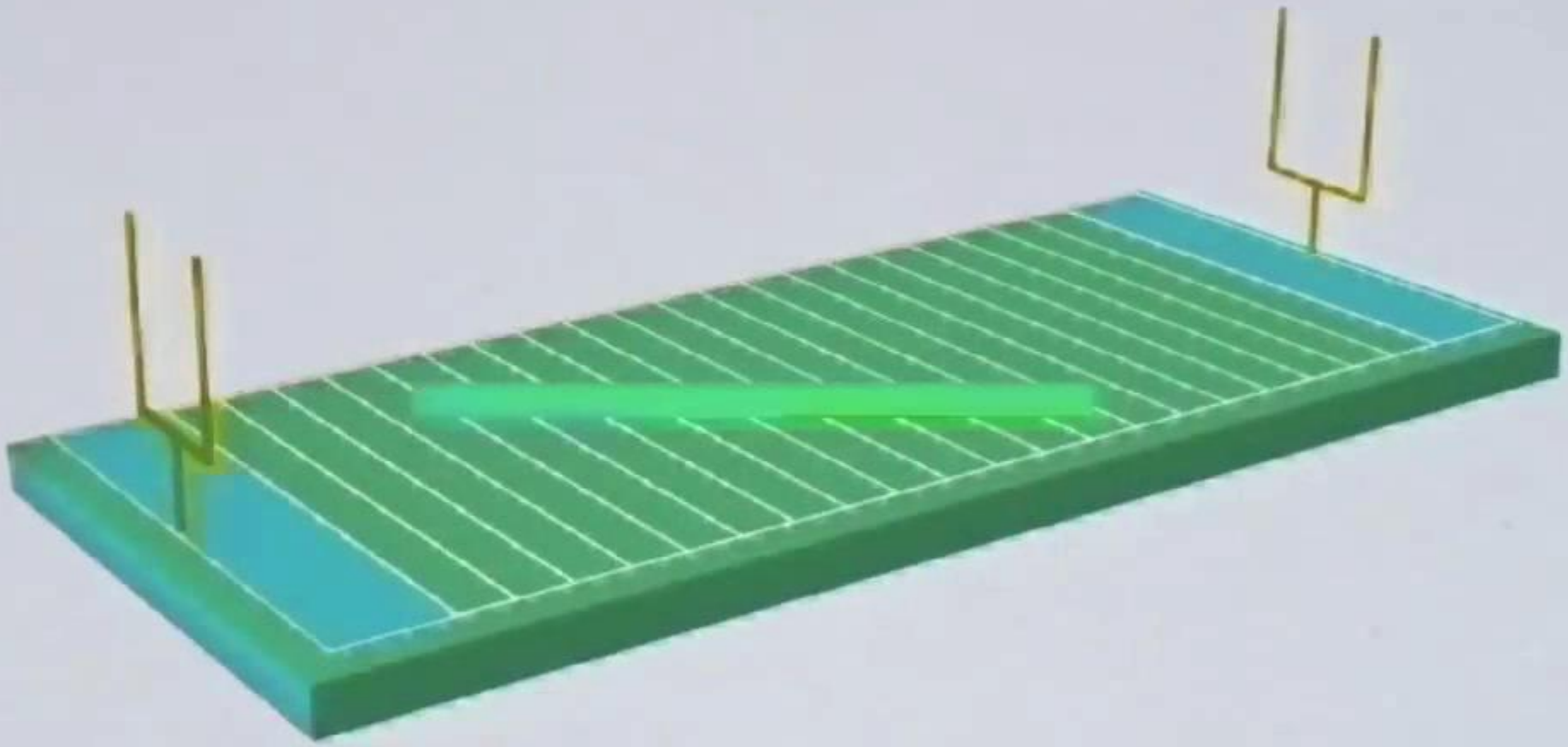
Vanishing point

What can vanishing line tell us about me?

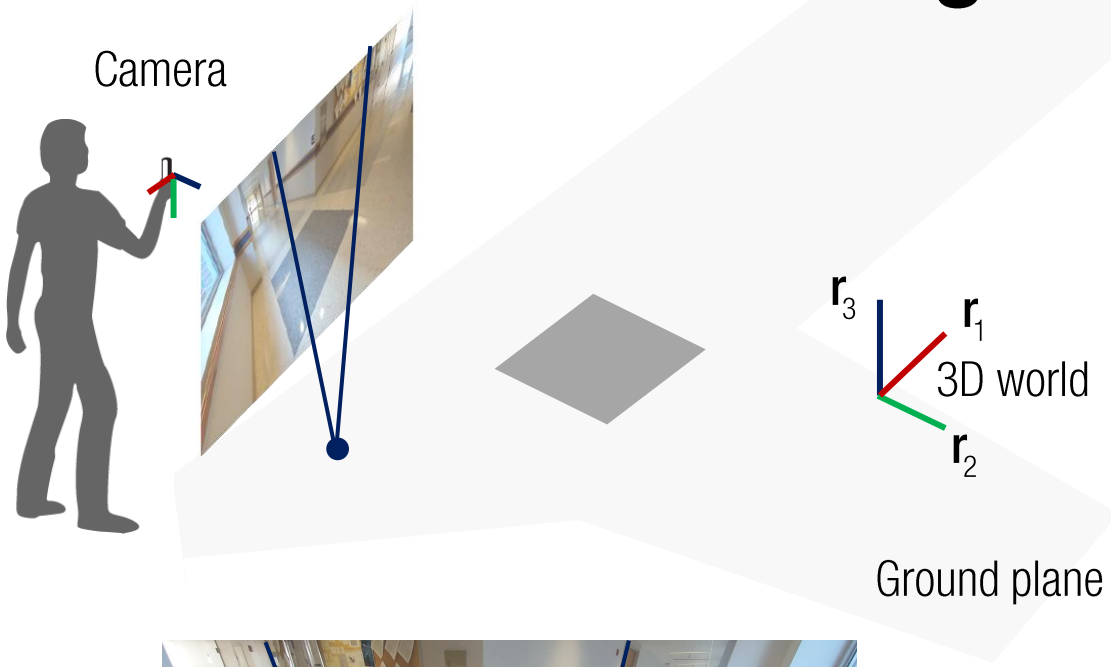
- Horizon
- Camera pitch angle (looking down)
- Camera roll angle (tilted toward right)

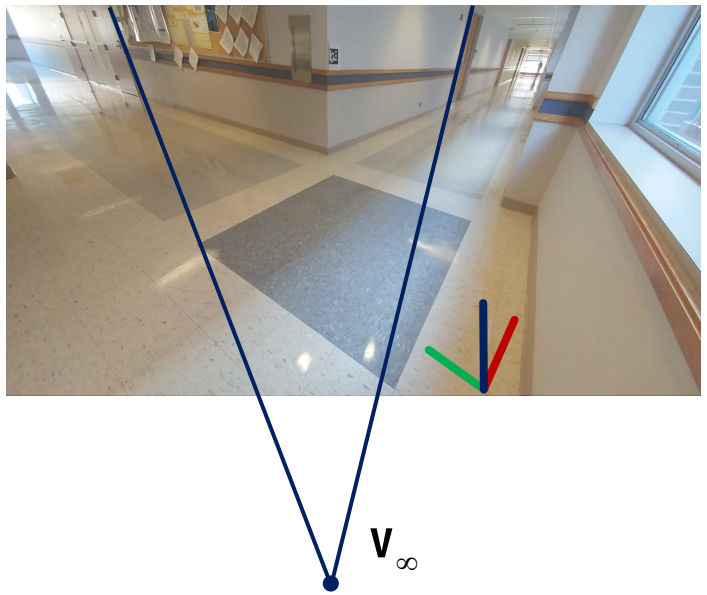


Keller Hall

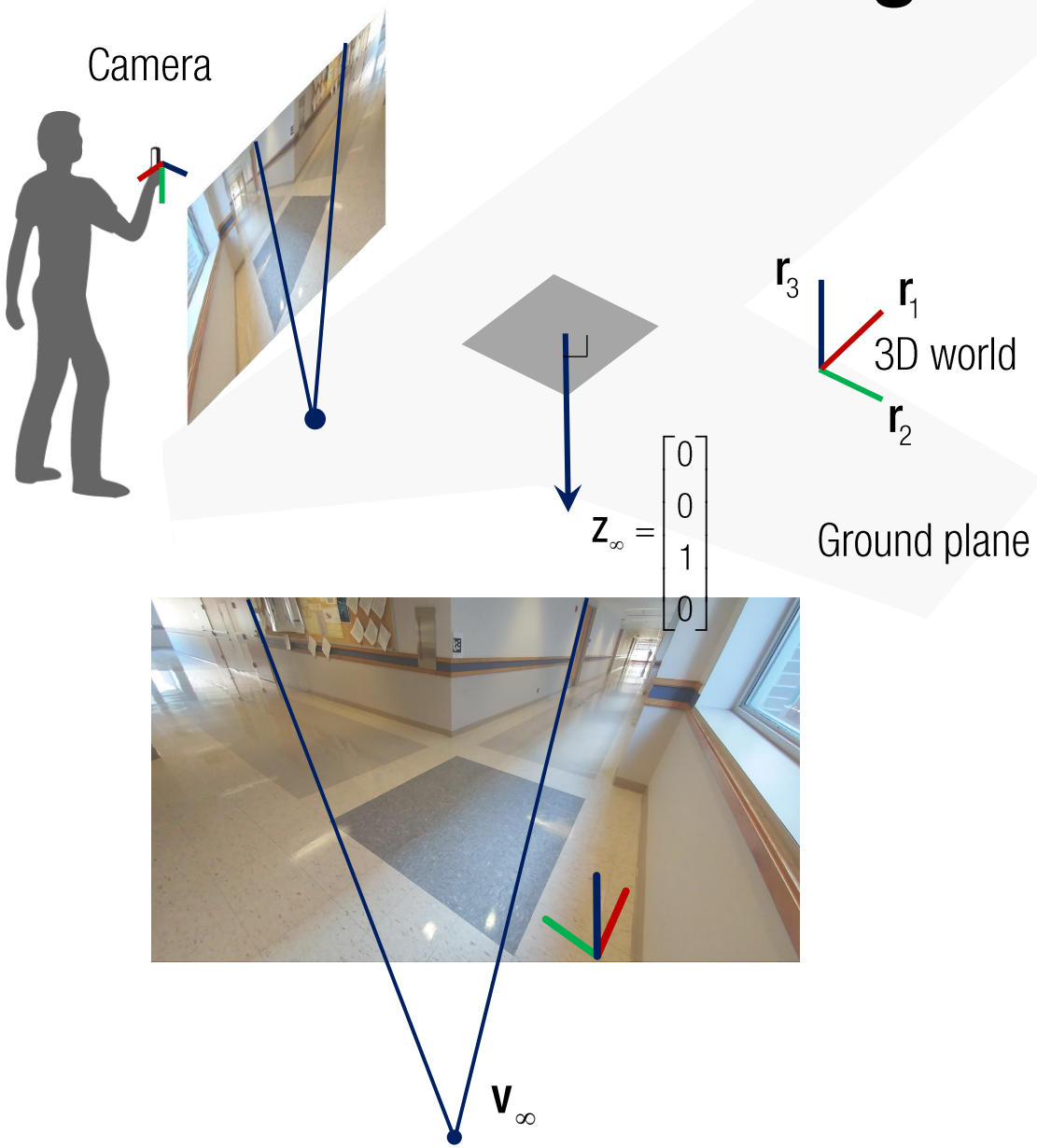


What can a Vanishing Point tell us about?

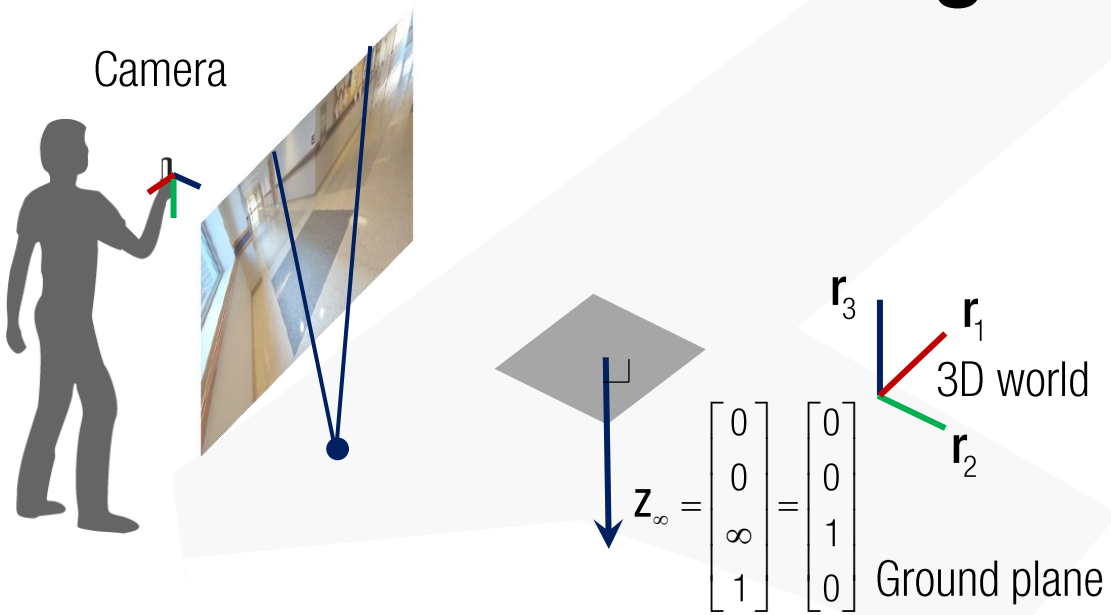




What can a Vanishing Point tell us about?



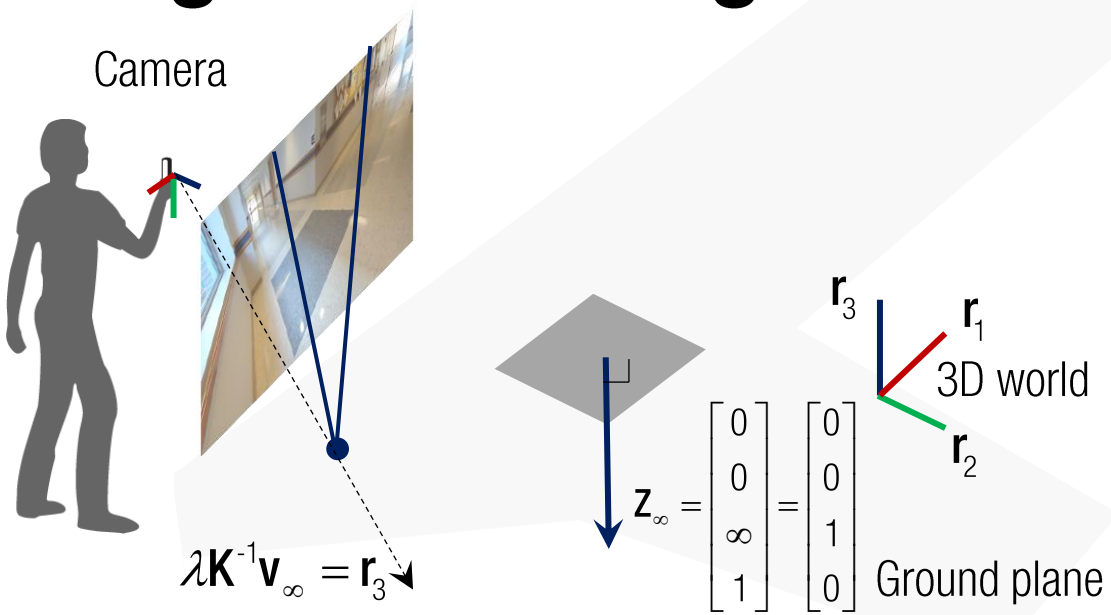
What can a Vanishing Point tell us about?



$$\lambda \mathbf{v}_\infty = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$\mathbf{v}_\infty$

Single Vanishing Point

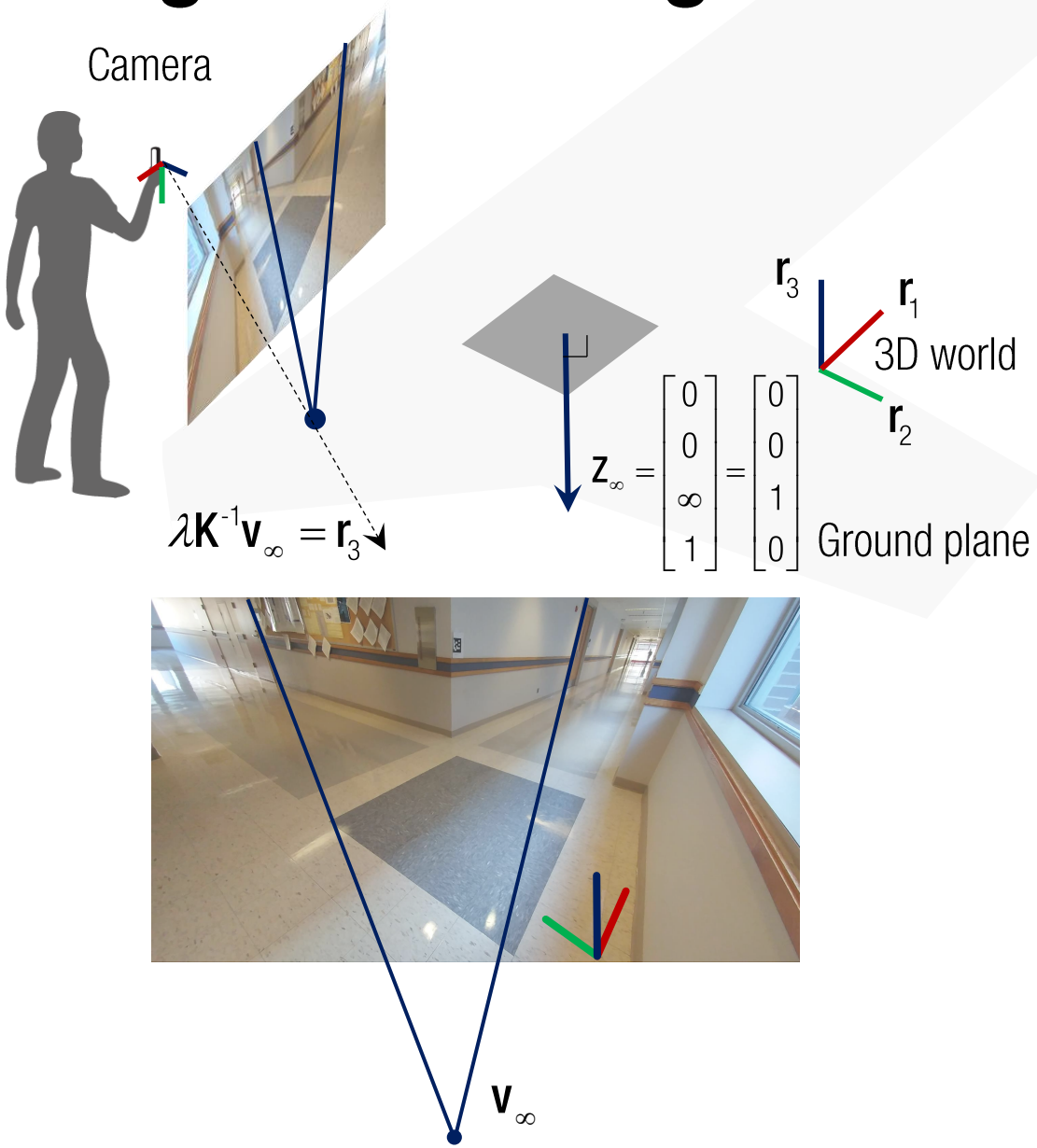


v_∞

$$\lambda \mathbf{v}_\infty = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\lambda \mathbf{v}_\infty = \mathbf{K}\mathbf{r}_3$$

Single Vanishing Point



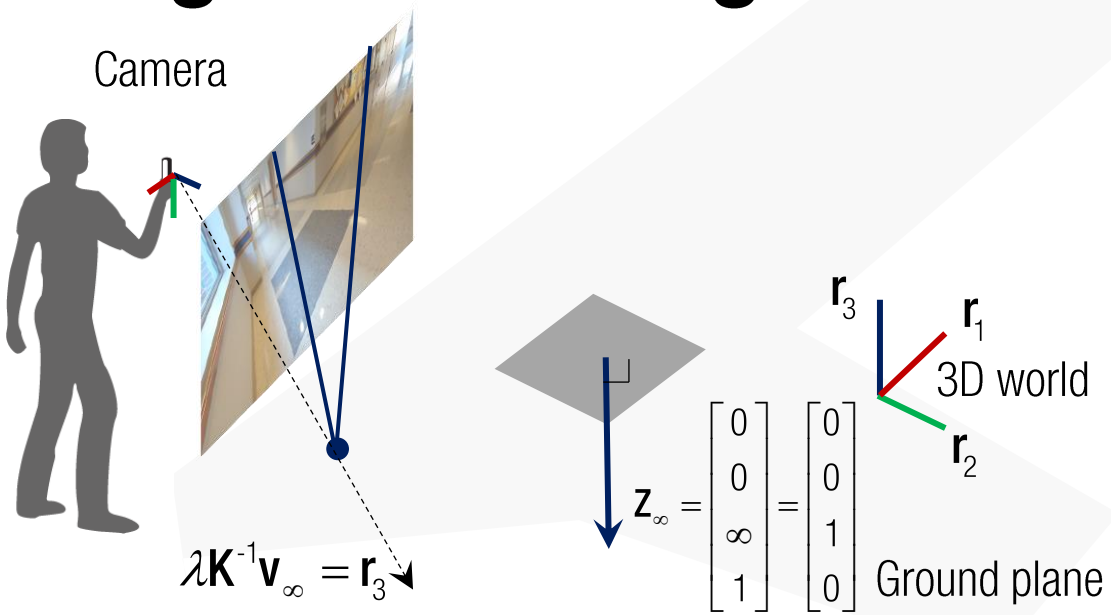
$$\lambda \mathbf{v}_\infty = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C\end{bmatrix}\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

$$\lambda \mathbf{v}_\infty = \mathbf{K}\mathbf{r}_3$$

$$\longrightarrow \mathbf{r}_3 = \frac{\mathbf{K}^{-1}\mathbf{v}_\infty}{\left\|\mathbf{K}^{-1}\mathbf{v}_\infty\right\|}$$ because $\mathbf{r}_3$ is a unit vector.

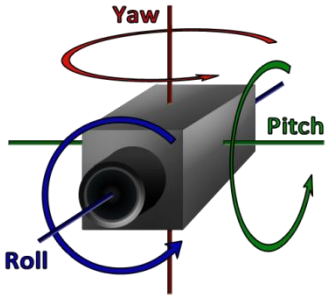
Z vanishing point tells us about the surface normal of the ground plane

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.

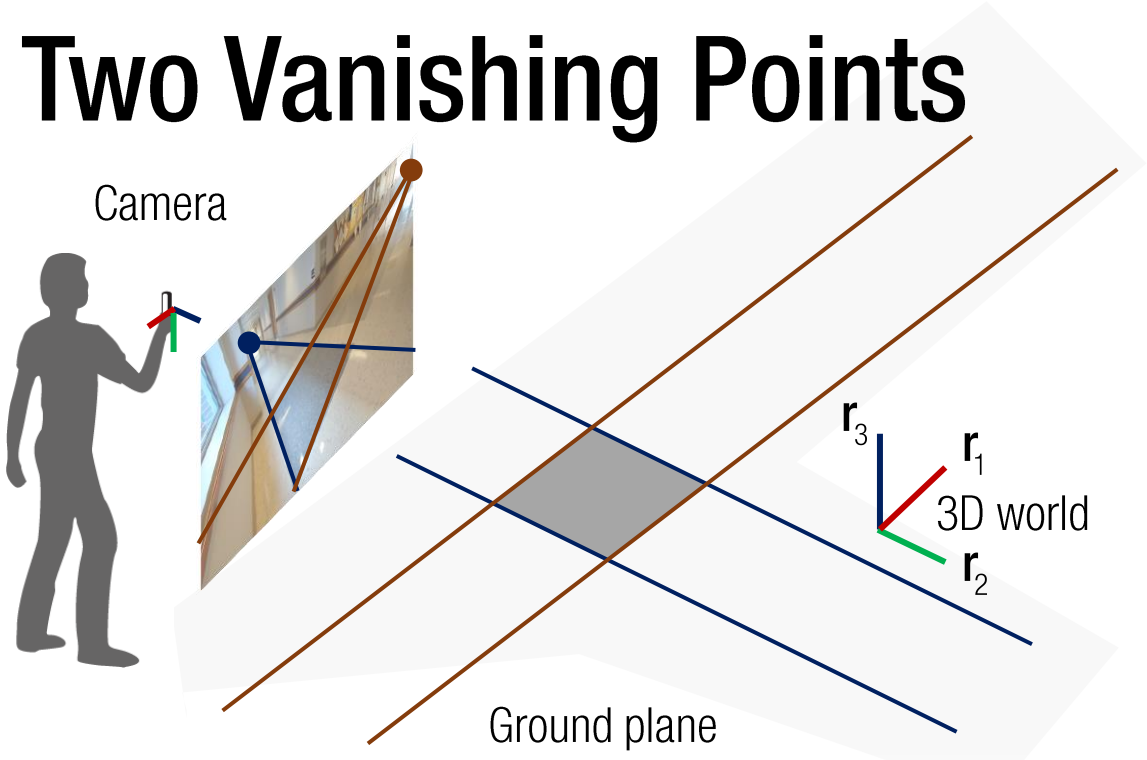
$$\begin{bmatrix} r_1 & r_2 & r_3 \end{bmatrix} = \left(R_{yaw}R_{pitch}R_{roll}\right)^T = \begin{bmatrix} \bullet & \bullet & -\sin\beta \\ \bullet & \bullet & \cos\beta\sin\gamma \\ \bullet & \bullet & \cos\beta\cos\gamma \end{bmatrix}$$

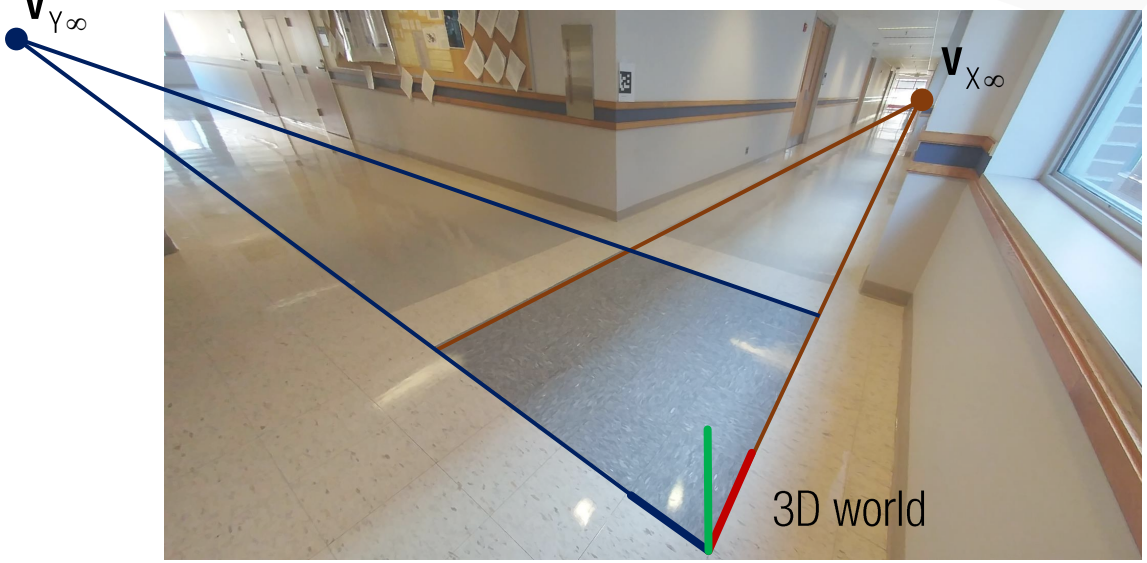
$\longrightarrow$

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

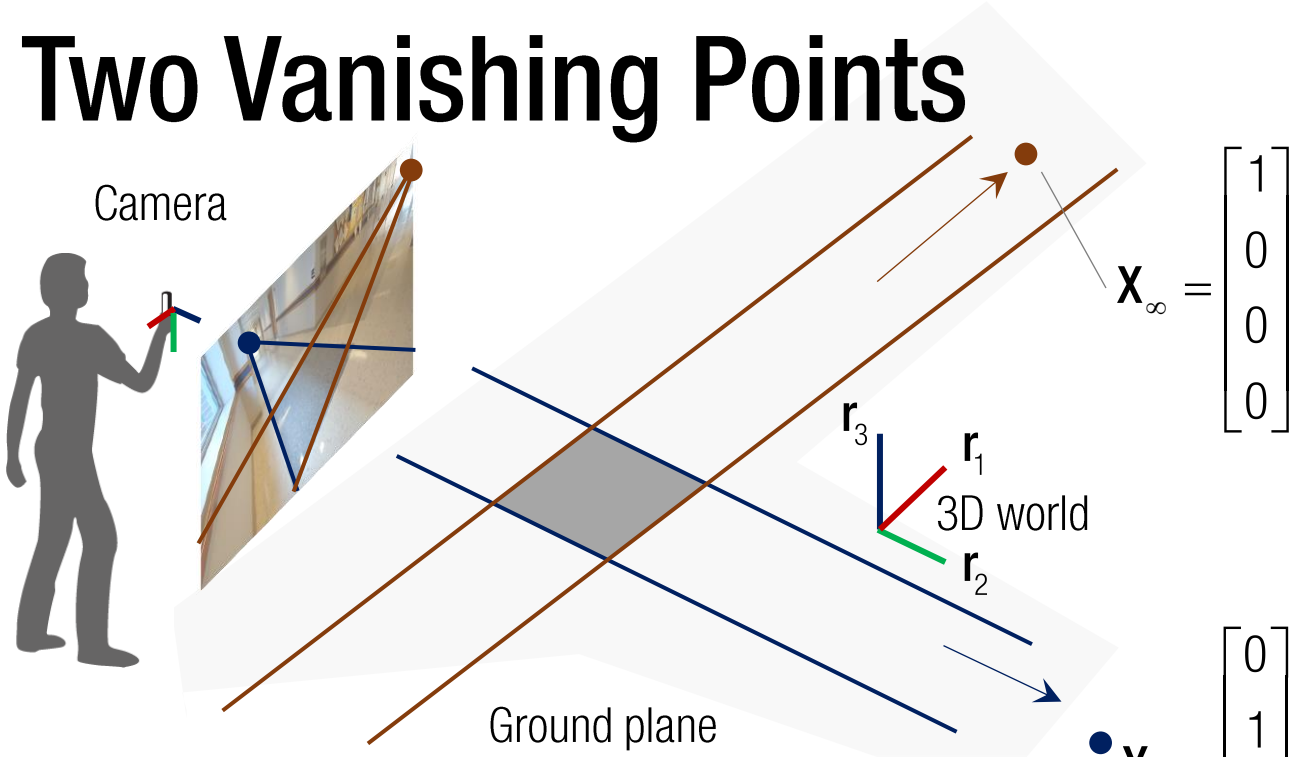
Roll: $\gamma = \tan^{-1}\frac{r_{32}}{r_{33}}$

Two Vanishing Points



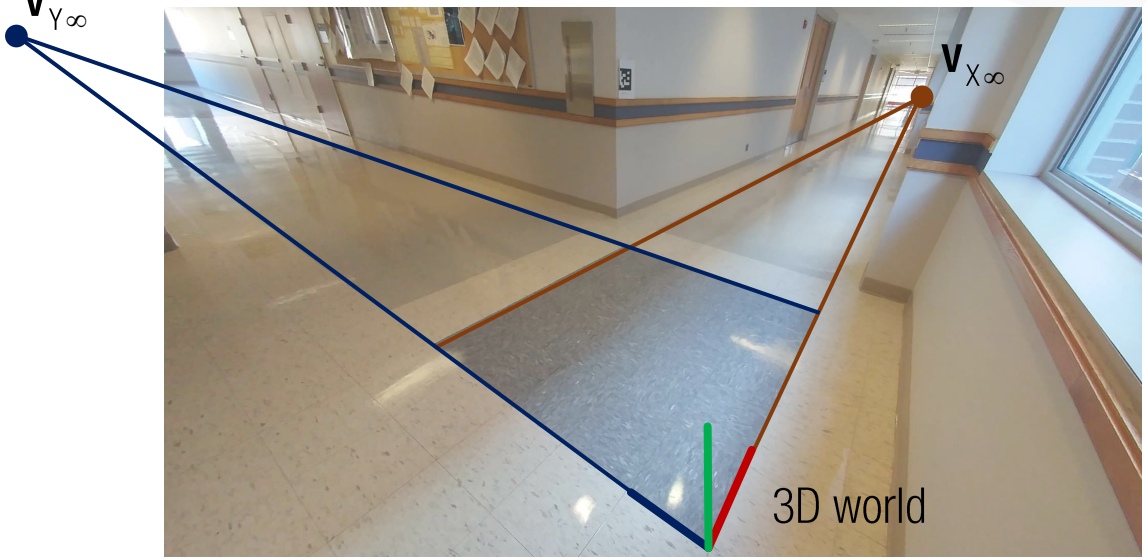


Two Vanishing Points

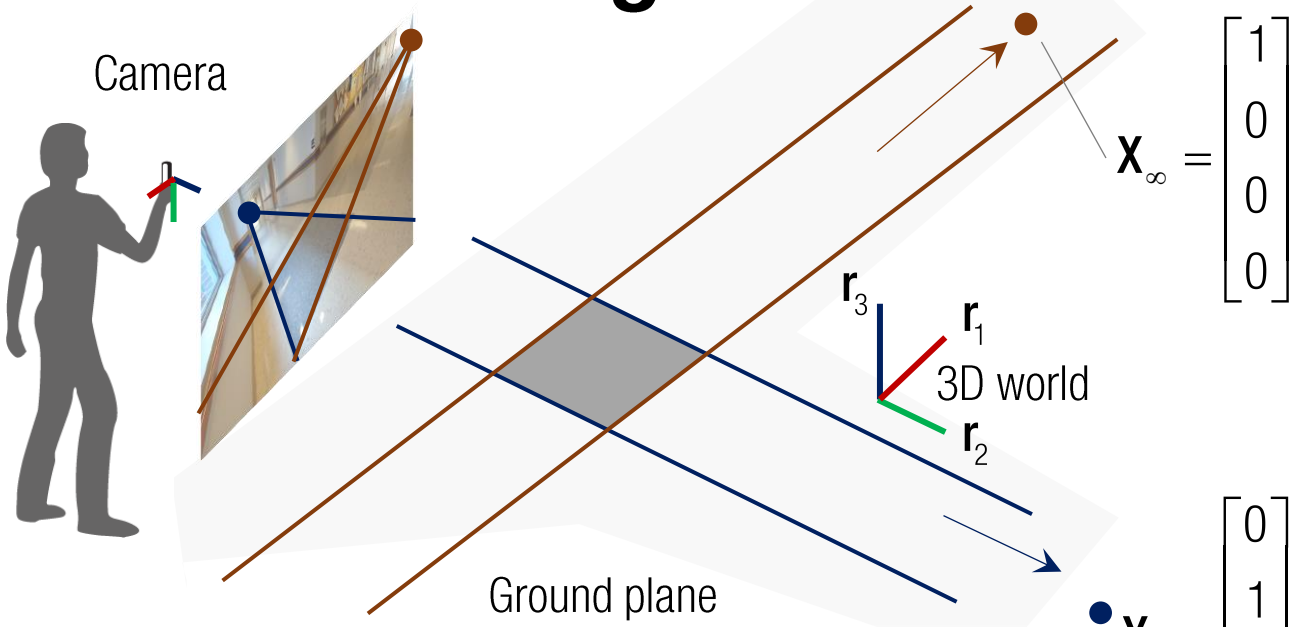


$$\mathbf{X}_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{Y}_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



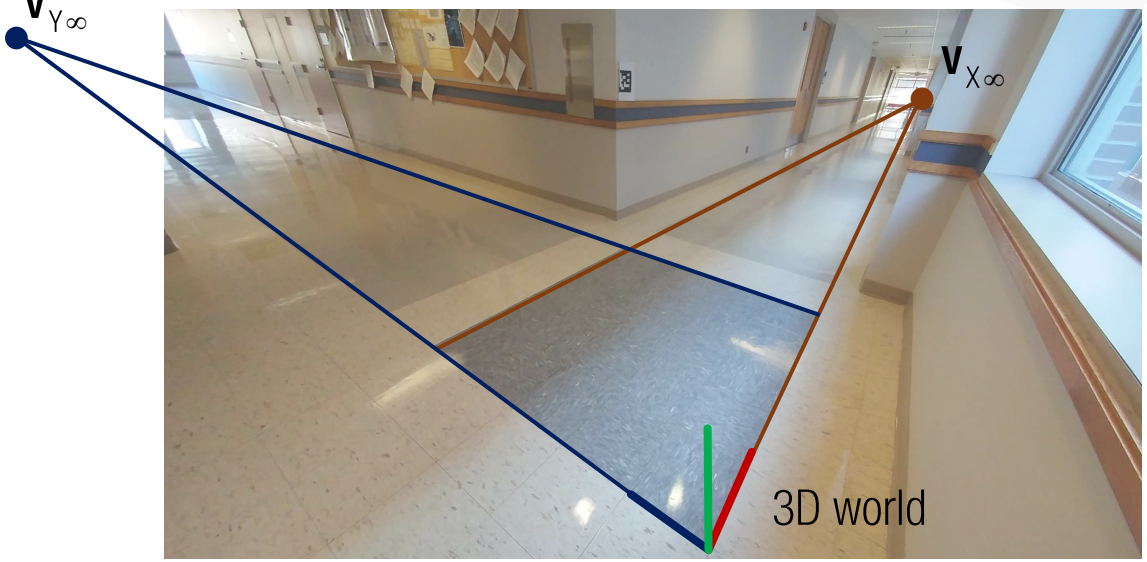
Two Vanishing Points



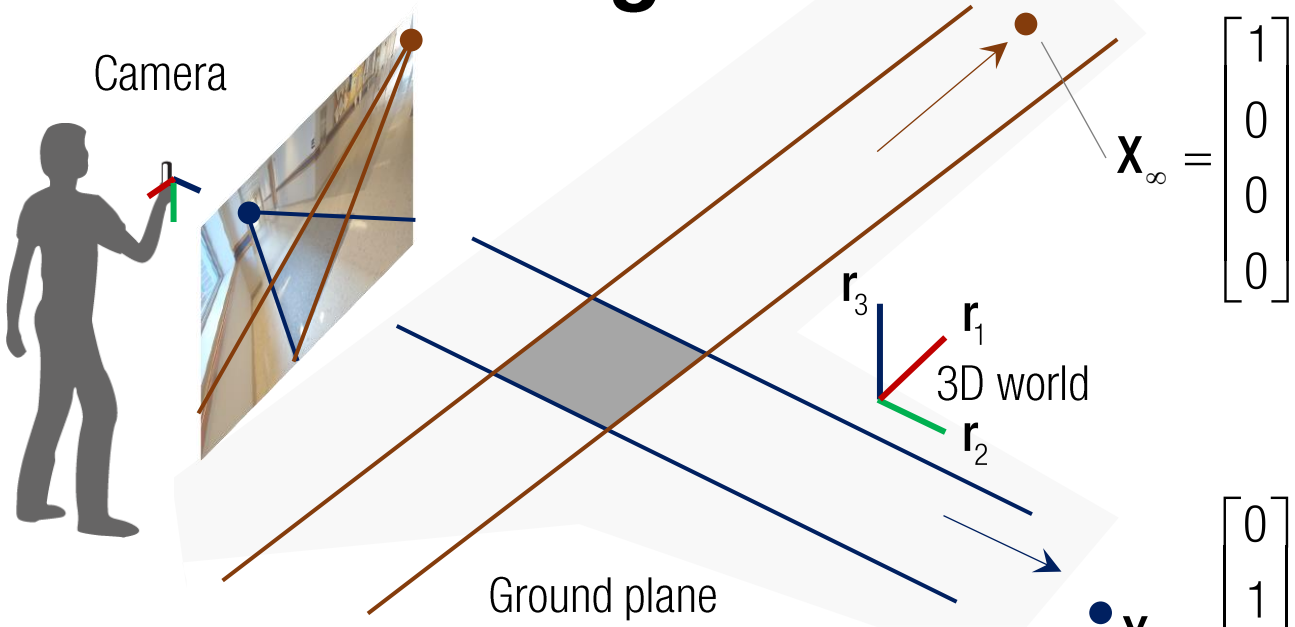
$$\mathbf{X}_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{Y}_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\lambda \mathbf{v}_{X\infty} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\mathbf{X}_\infty$$
$$= \mathbf{K}\mathbf{r}_1$$

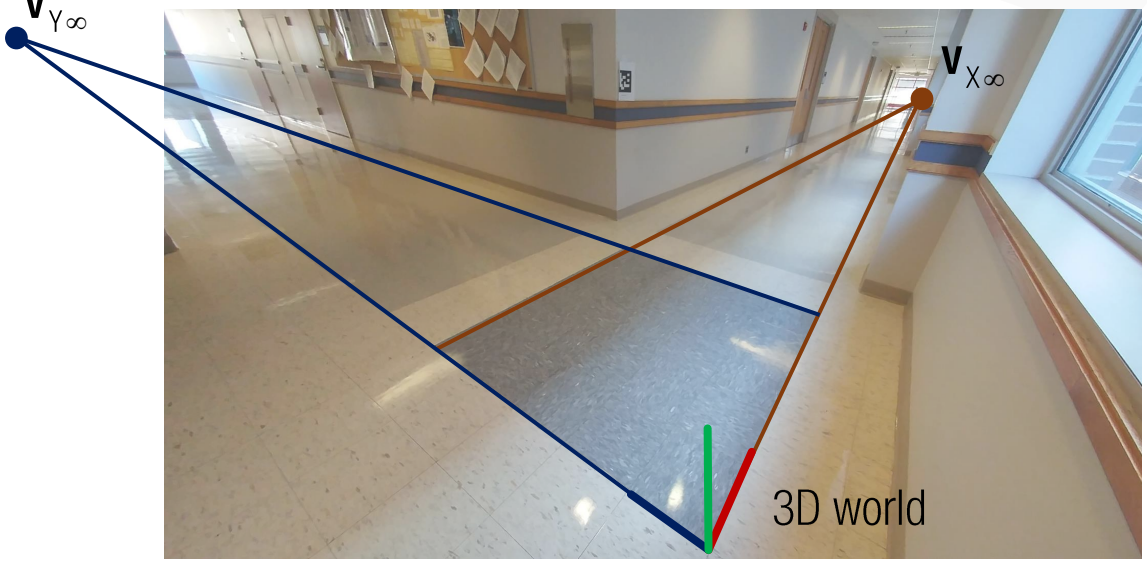


Two Vanishing Points



$$\mathbf{X}_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

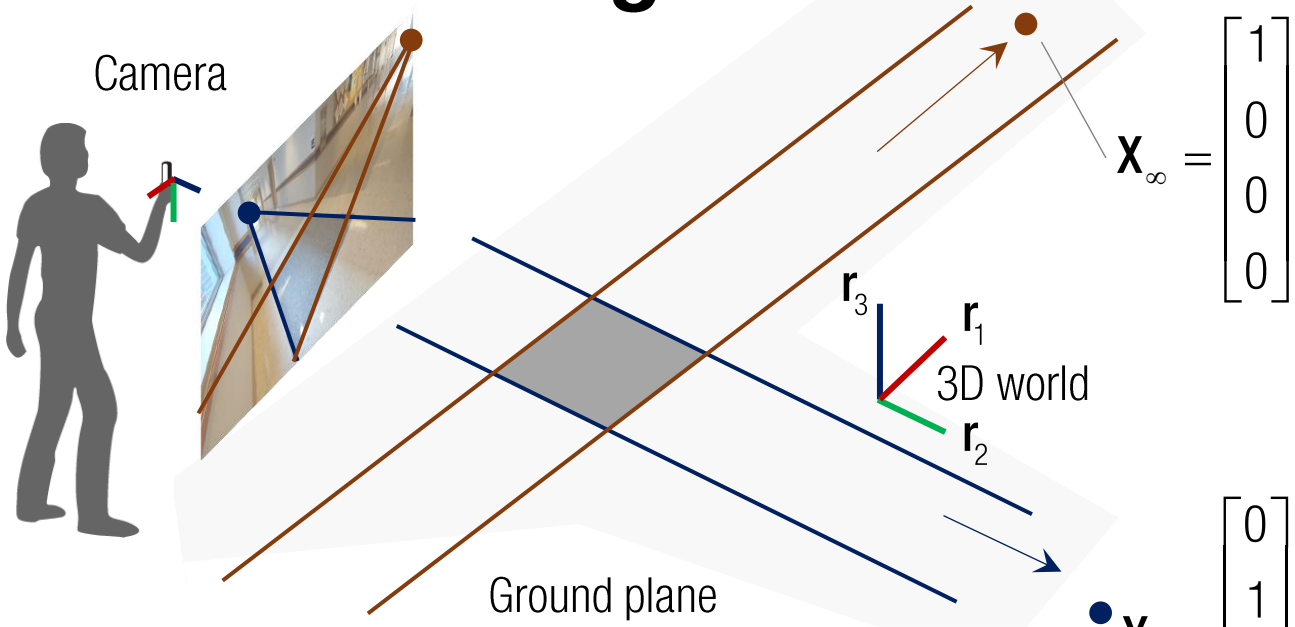
$$\mathbf{Y}_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$



$$\lambda \mathbf{v}_{X\infty} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\mathbf{X}_\infty$$
$$= \mathbf{K}\mathbf{r}_1$$

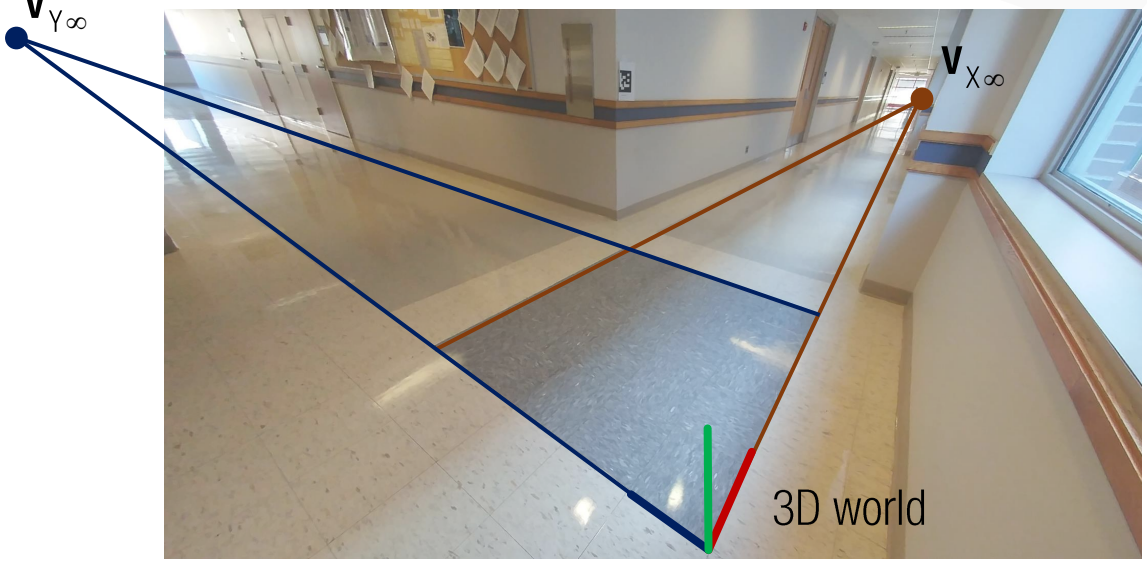
$$\lambda \mathbf{v}_{Y\infty} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\mathbf{Y}_\infty$$
$$= \mathbf{K}\mathbf{r}_2$$

Two Vanishing Points



$$\mathbf{X}_\infty = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathbf{Y}_\infty = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

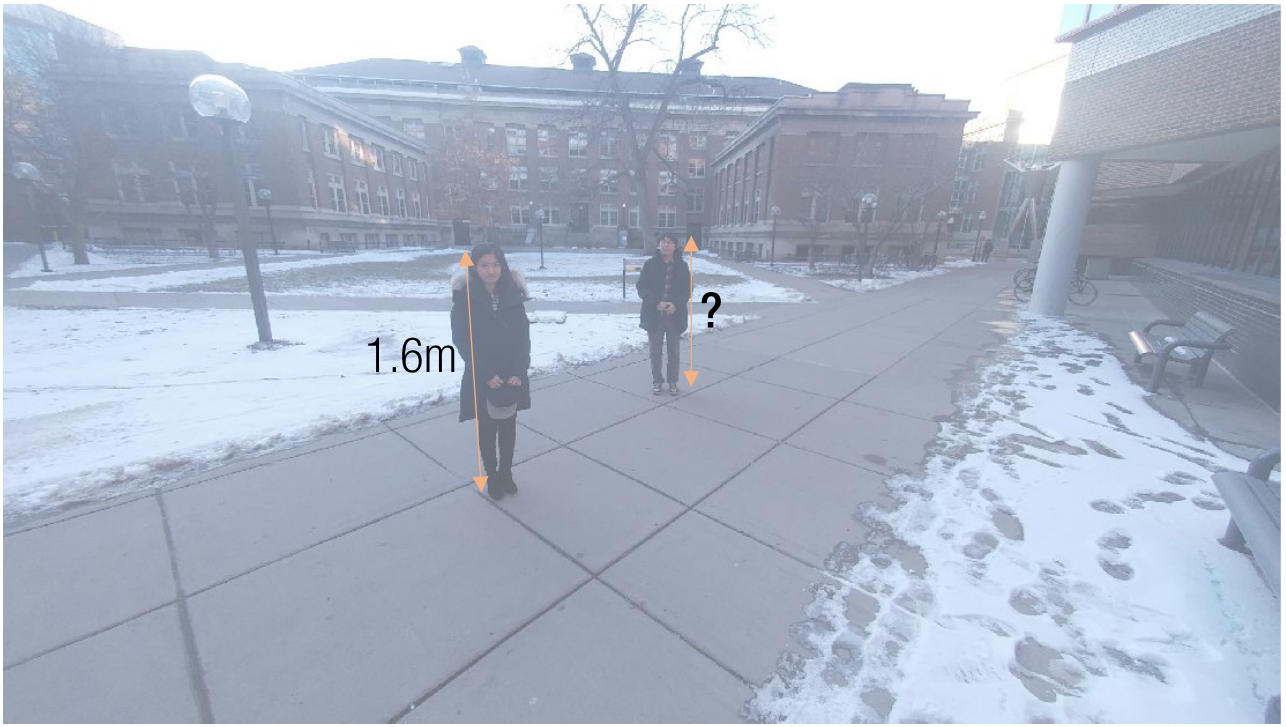


$$\lambda \mathbf{v}_{X\infty} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\mathbf{X}_\infty = \mathbf{K}\mathbf{r}_1$$

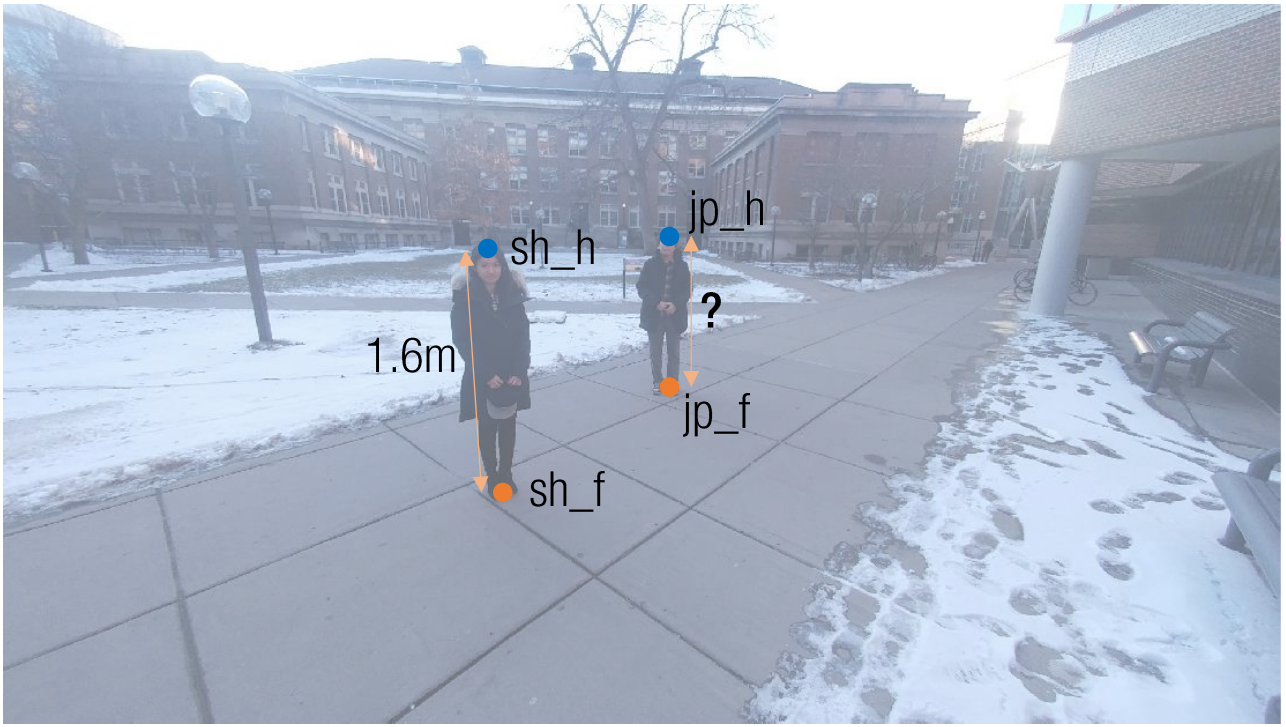
$$\lambda \mathbf{v}_{Y\infty} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}_W^C \end{bmatrix}\mathbf{Y}_\infty = \mathbf{K}\mathbf{r}_2$$

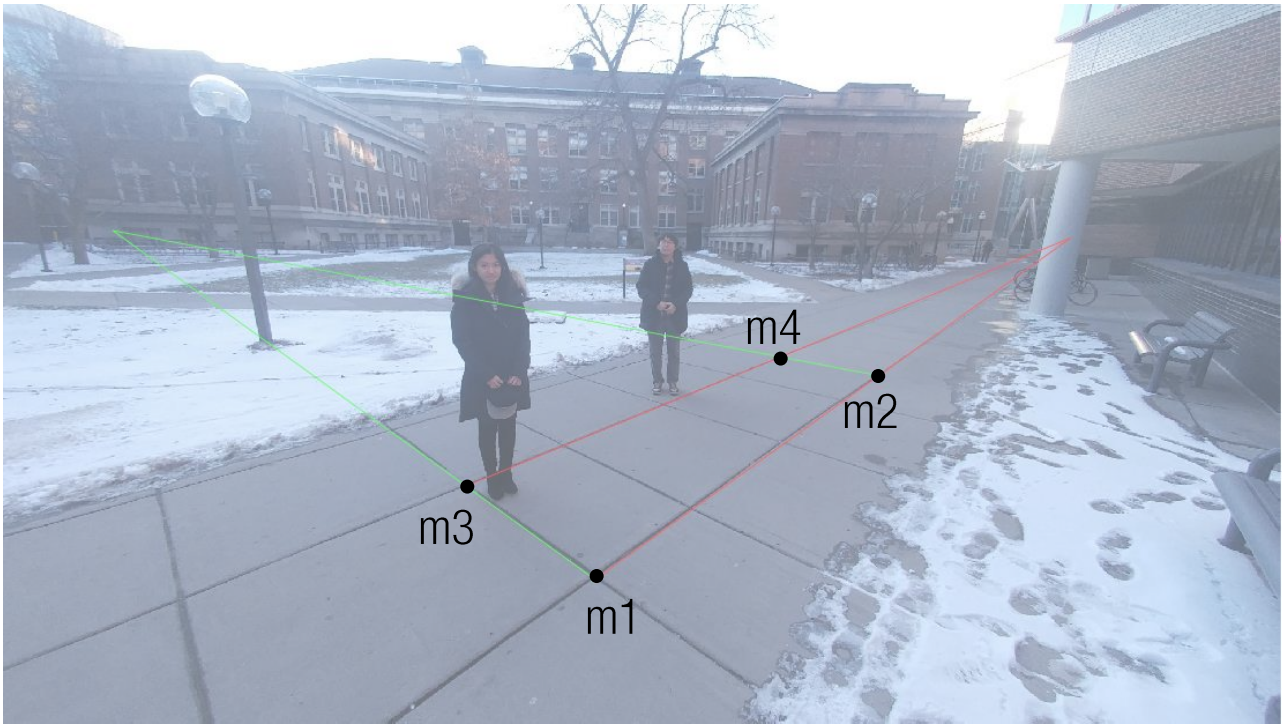
$$\rightarrow \quad \mathbf{r}_1 = \frac{\mathbf{K}^{-1}\mathbf{v}_{X\infty}}{\left\|\mathbf{K}^{-1}\mathbf{v}_{X\infty}\right\|}, \quad \mathbf{r}_2 = \frac{\mathbf{K}^{-1}\mathbf{v}_{Y\infty}}{\left\|\mathbf{K}^{-1}\mathbf{v}_{Y\infty}\right\|}$$

$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$: Orthogonality constraint

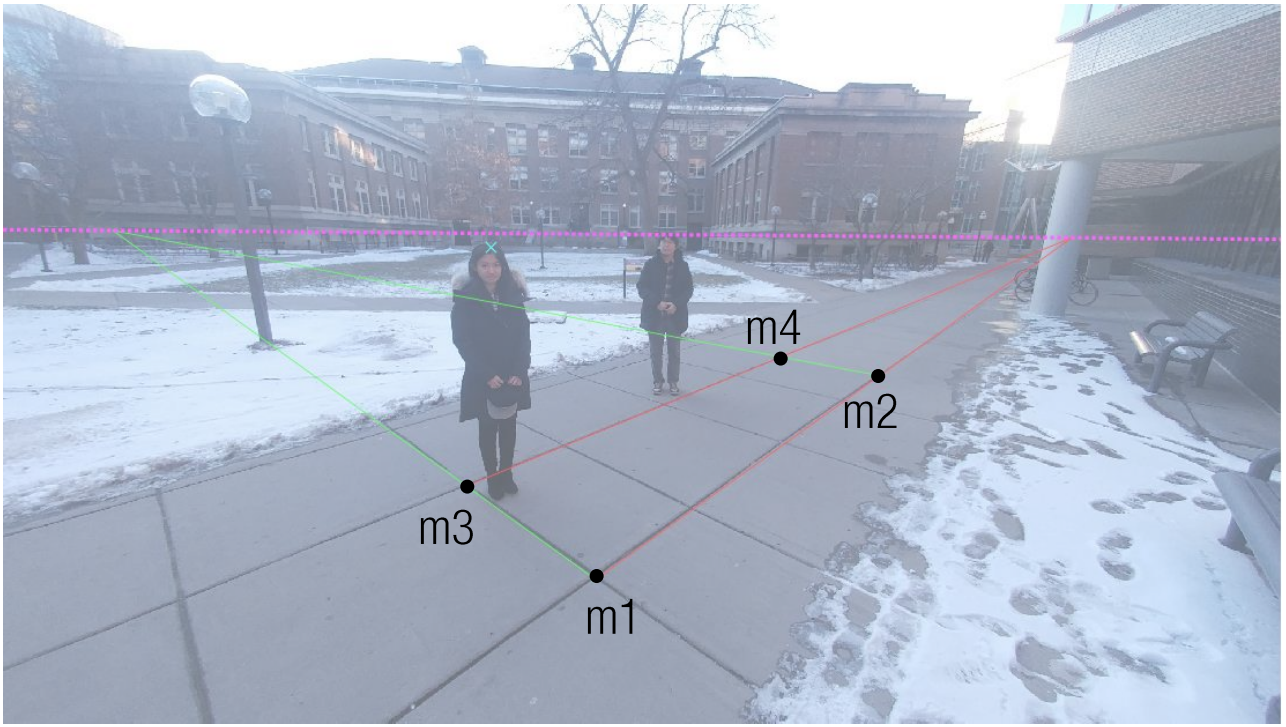

1.6m
?

sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];



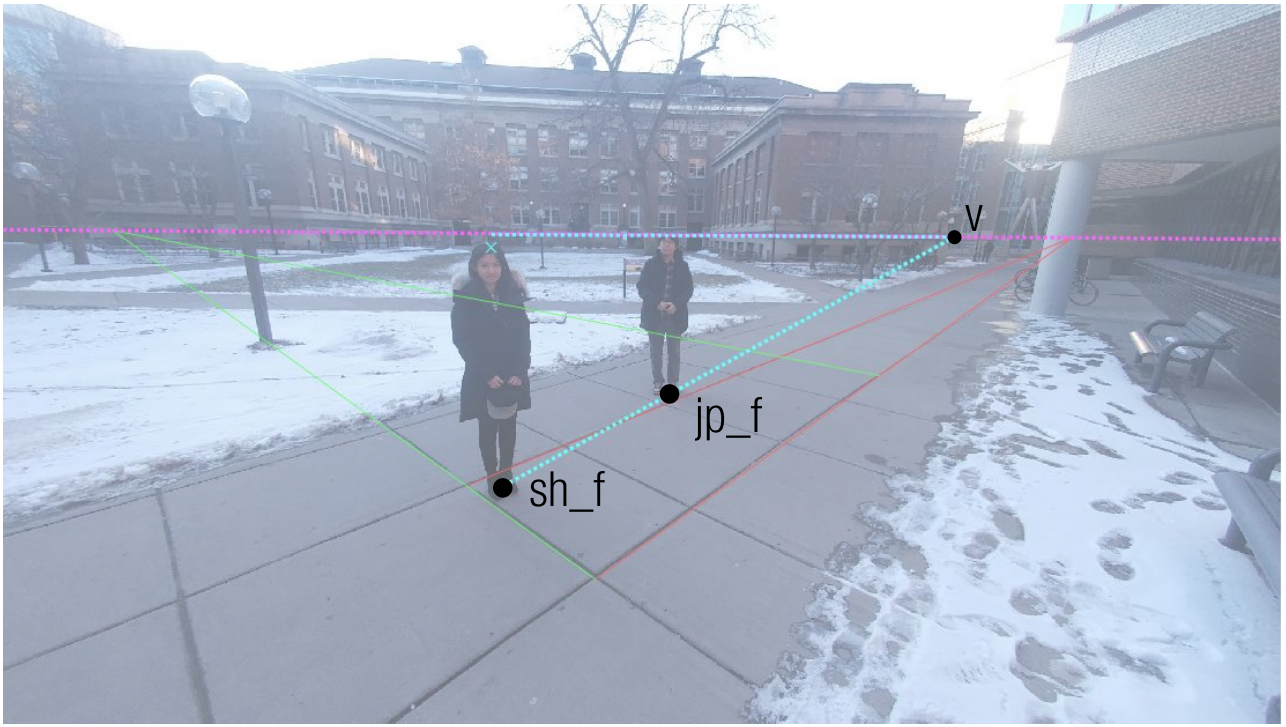



```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);
```




```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

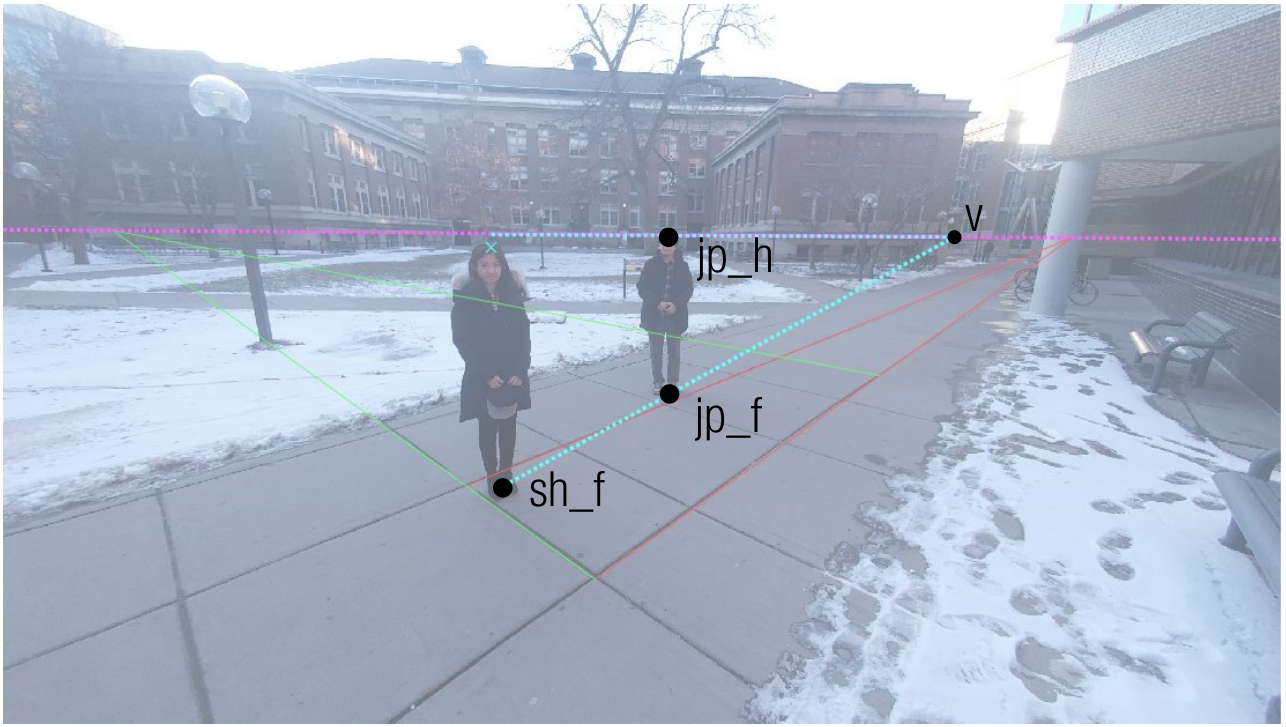
← Vanishing line




```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

← Vanishing line

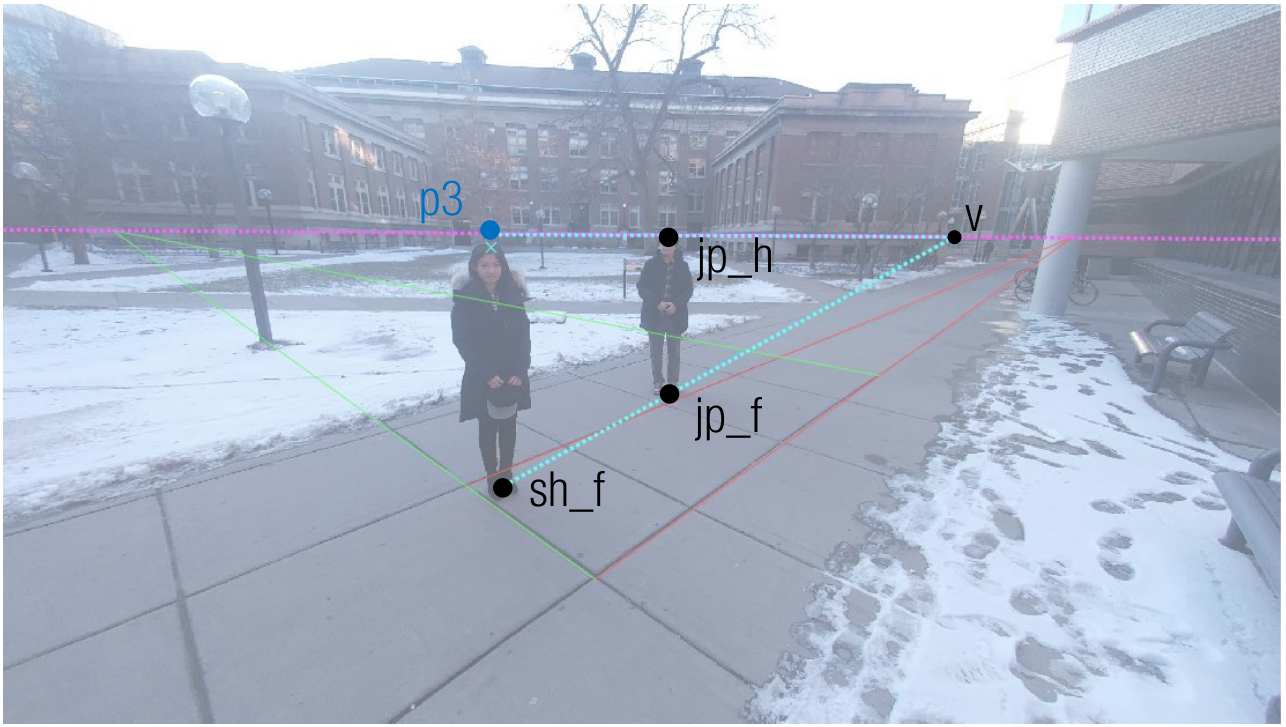
```
line_sh_jp_f = GetLineFromTwoPoints(sh_f, jp_f);
v = GetPointFromTwoLines(line_sh_jp_f, l);
```




```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

← Vanishing line

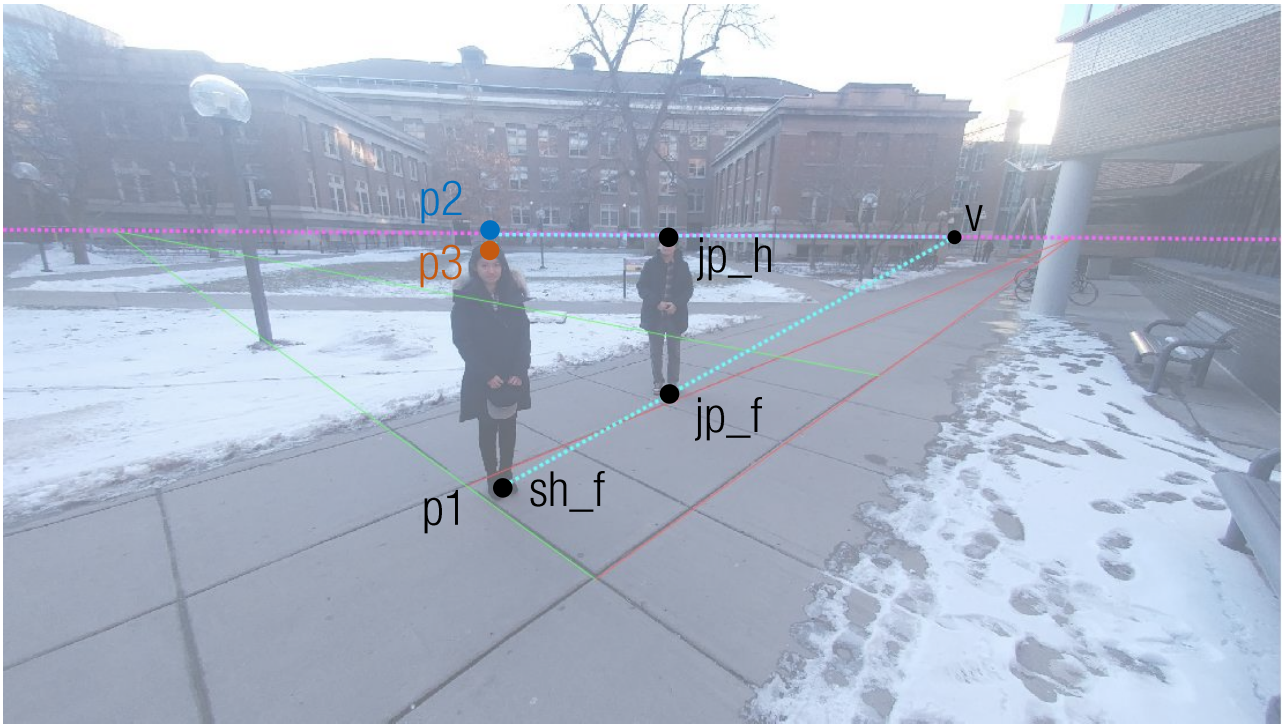
```
line_sh_jp_f = GetLineFromTwoPoints(sh_f, jp_f);
v = GetPointFromTwoLines(line_sh_jp_f, l);

line_jp_h_v = GetLineFromTwoPoints(jp_head, v);
```




```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

← Vanishing line

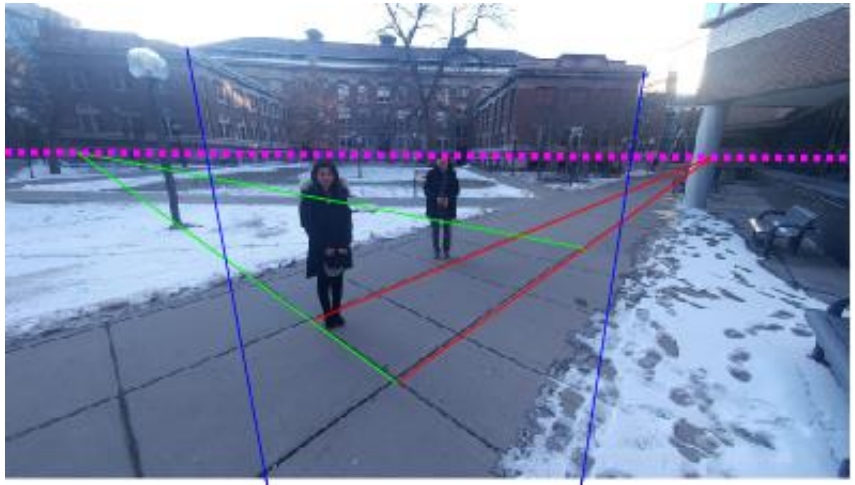
```
line_sh_jp_f = GetLineFromTwoPoints(sh_f, jp_f);
v = GetPointFromTwoLines(line_sh_jp_f, l);

line_jp_h_v = GetLineFromTwoPoints(jp_head, v);
line_sh = GetLineFromTwoPoints(sh_h, sh_f);
p2  = GetPointFromTwoLines(line_jp_head_v, line_sh);
```




```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

← Vanishing line

```
line_sh_jp_f = GetLineFromTwoPoints(sh_f, jp_f);
v = GetPointFromTwoLines(line_sh_jp_f, l);

line_jp_h_v = GetLineFromTwoPoints(jp_head, v);
line_sh = GetLineFromTwoPoints(sh_h, sh_f);
p2  = GetPointFromTwoLines(line_jp_head_v, line_sh);
p3 = sh_h;
p1 = sh_f;
```

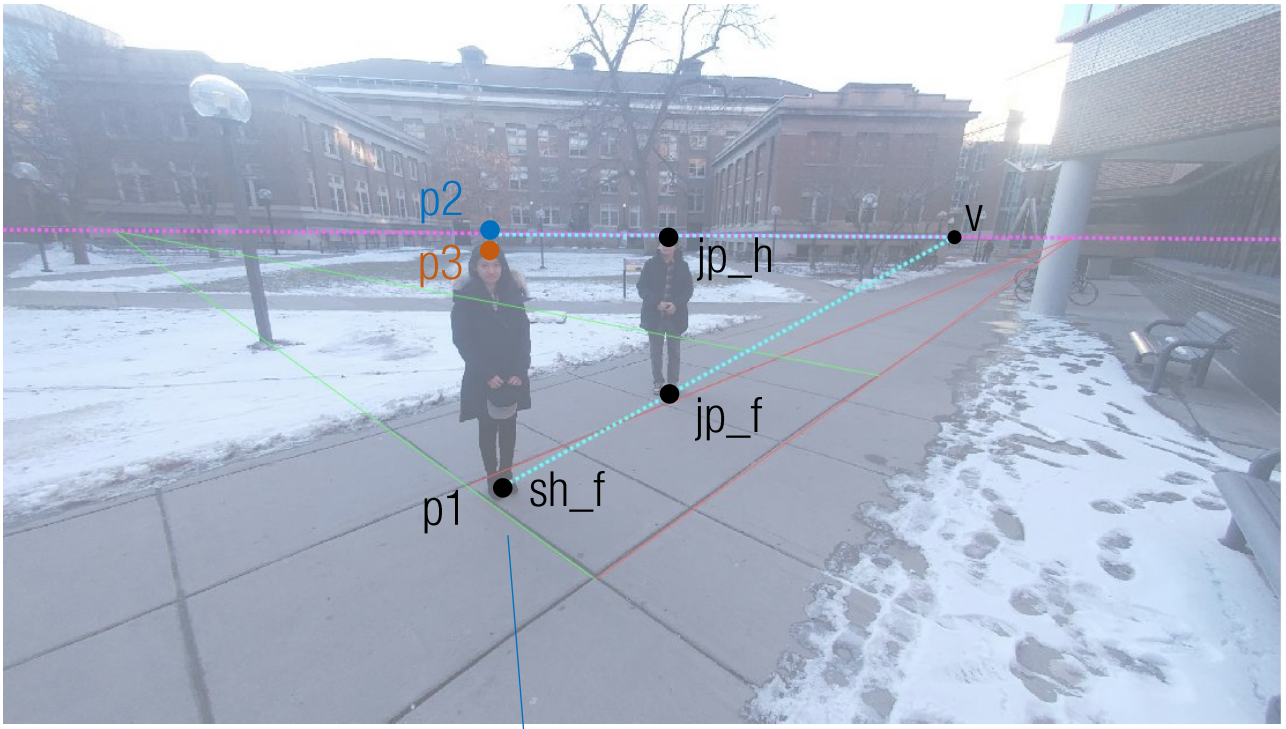


p4

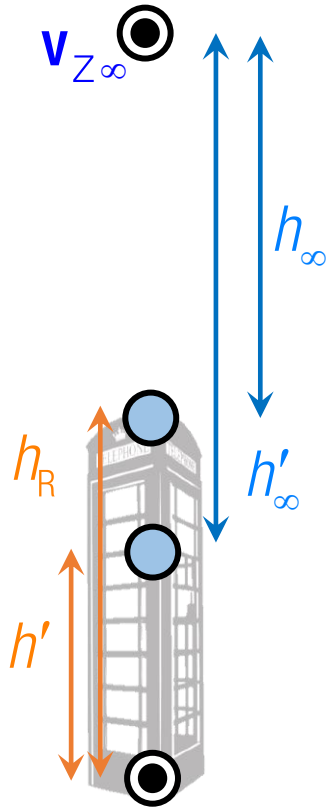
```
sh_f = [1504;1447;1];
sh_h = [1468;730;1];
jp_f = [1997;1175;1];
jp_h = [1997;695;1];

l11 = GetLineFromTwoPoints(m1,m2);
l12 = GetLineFromTwoPoints(m3,m4);
l21 = GetLineFromTwoPoints(m1,m3);
l22 = GetLineFromTwoPoints(m2,m4);

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

line_sh_jp_f = GetLineFromTwoPoints(sh_f, jp_f);
v = GetPointFromTwoLines(line_sh_jp_f, l);

line_jp_h_v = GetLineFromTwoPoints(jp_head, v);
line_sh = GetLineFromTwoPoints(sh_h, sh_f);
p3  = GetPointFromTwoLines(line_jp_head_v, line_sh);
p2 = sh_h;
p1 = sh_f;

l31 = GetLineFromTwoPoints(m5,m6);
l32 = GetLineFromTwoPoints(m7,m8)
v3 = GetPointFromTwoLines(l31,l32);
p4 = v3;
```

← Vanishing line

ComputeHeightFromCrossRatio.m

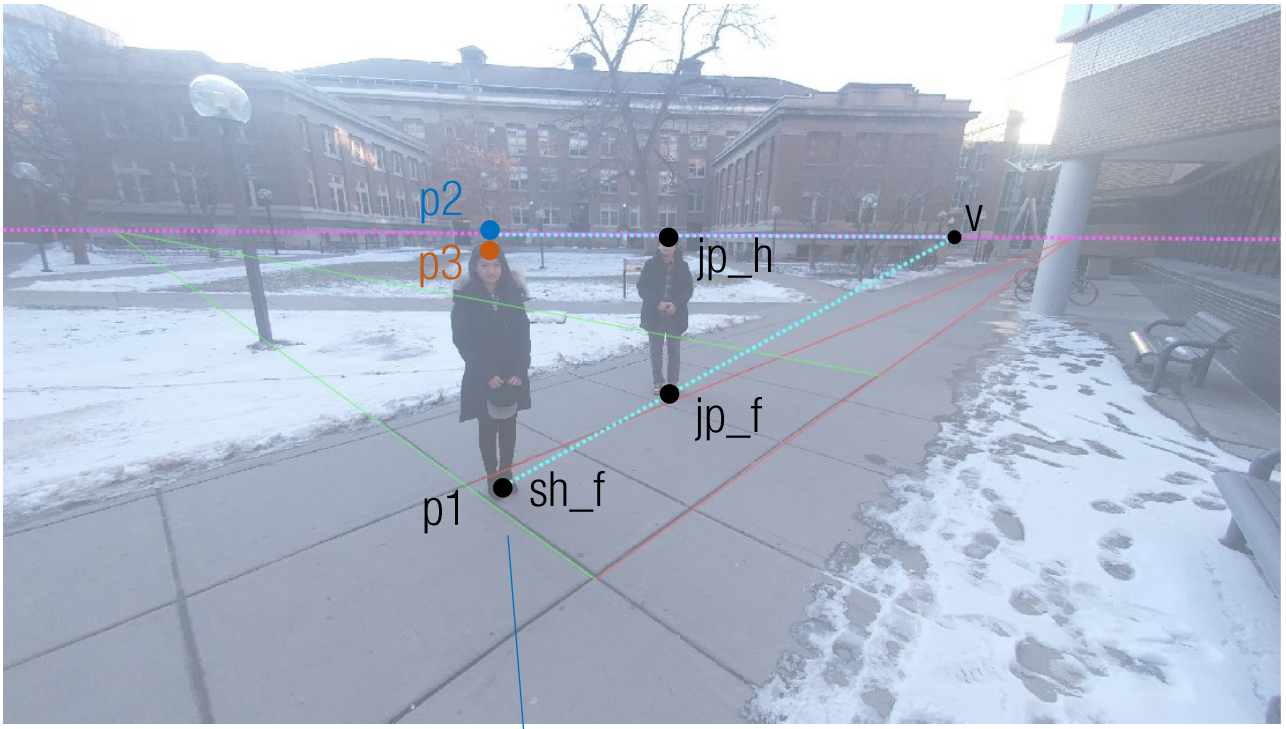



```
h_prime = norm(p1-p2);
h_R = norm(p1-p3);
h_prime_inf = norm(p4-p2);
h_inf = norm(p4-p3);
```



$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

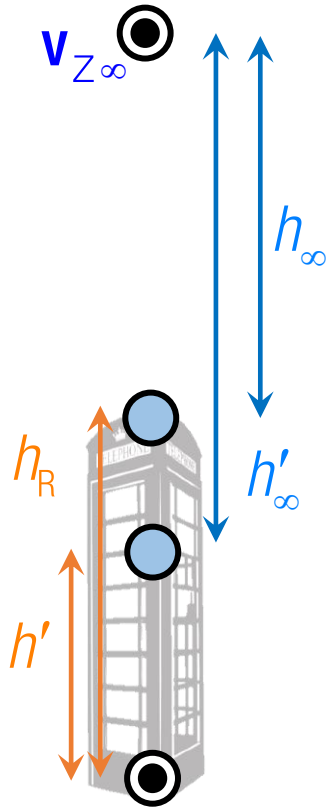
ComputeHeightFromCrossRatio.m




```
h_prime = norm(p1-p2);
h_R = norm(p1-p3);
h_prime_inf = norm(p4-p2);
h_inf = norm(p4-p3);

H = H_R * h_prime * h_prime_inf / h_R / h_inf
```

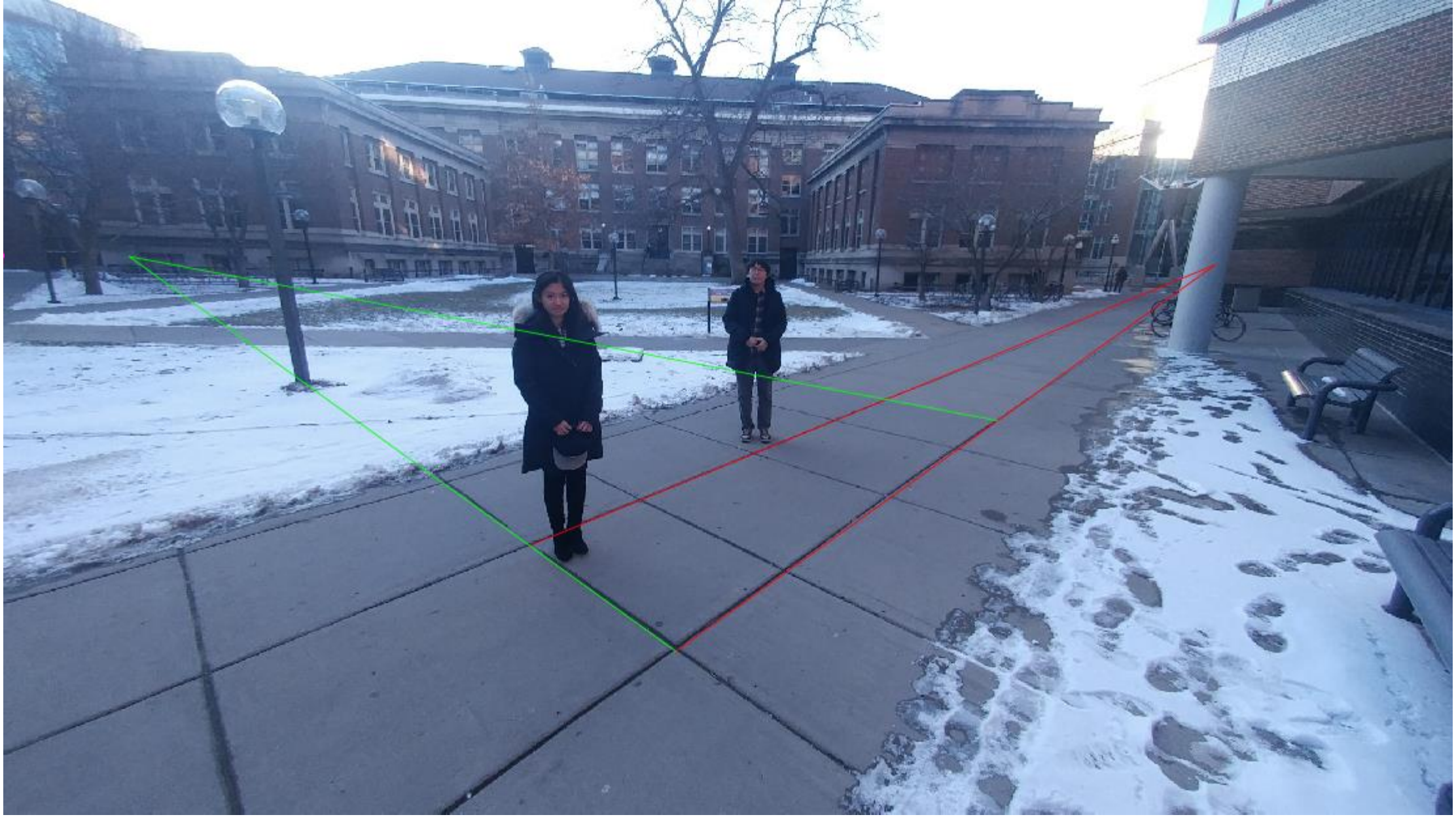
H =
1.6779

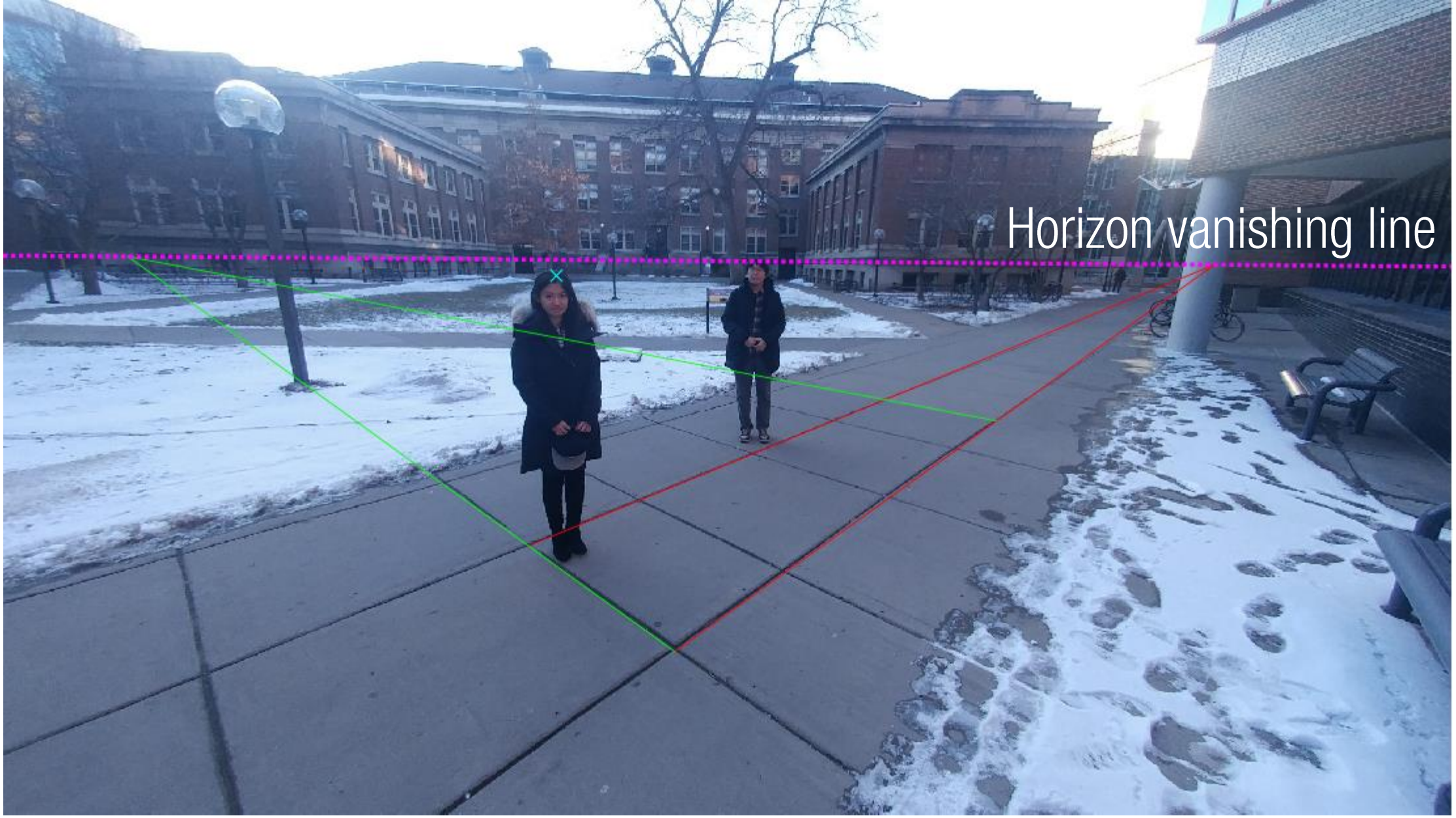
Ground truth: 1.7m

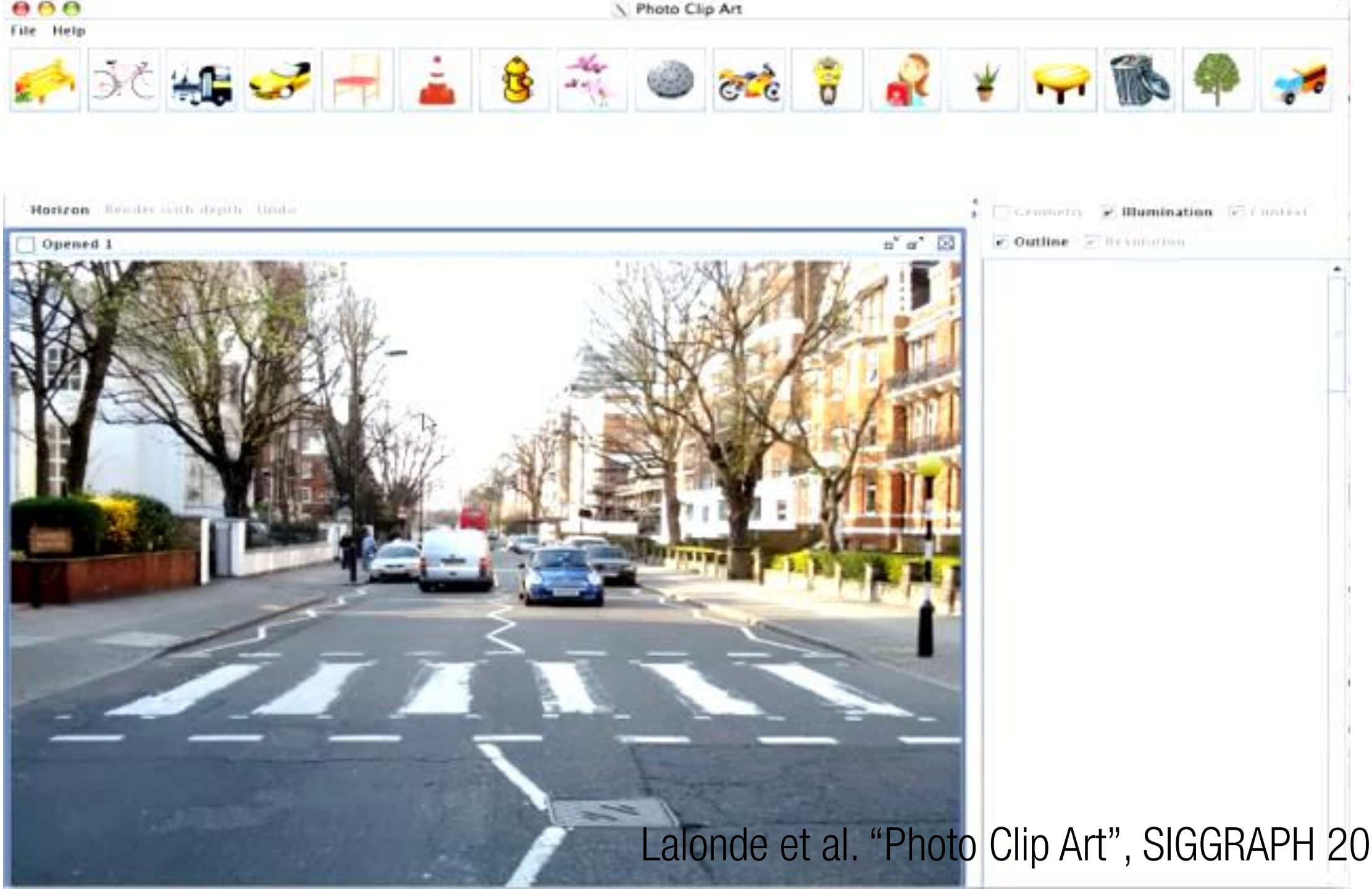


$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

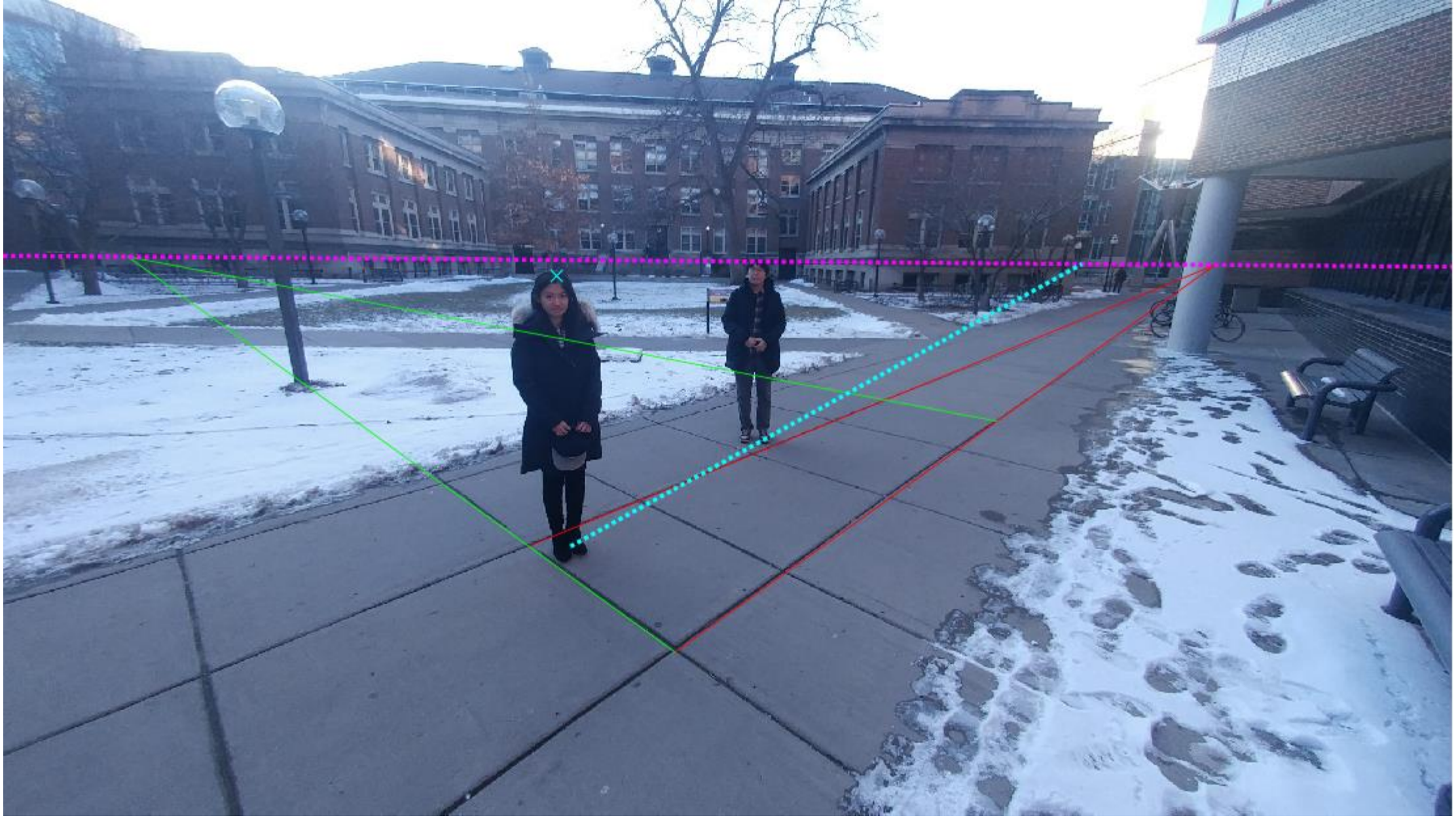
ComputeHeightFromCrossRatio.m

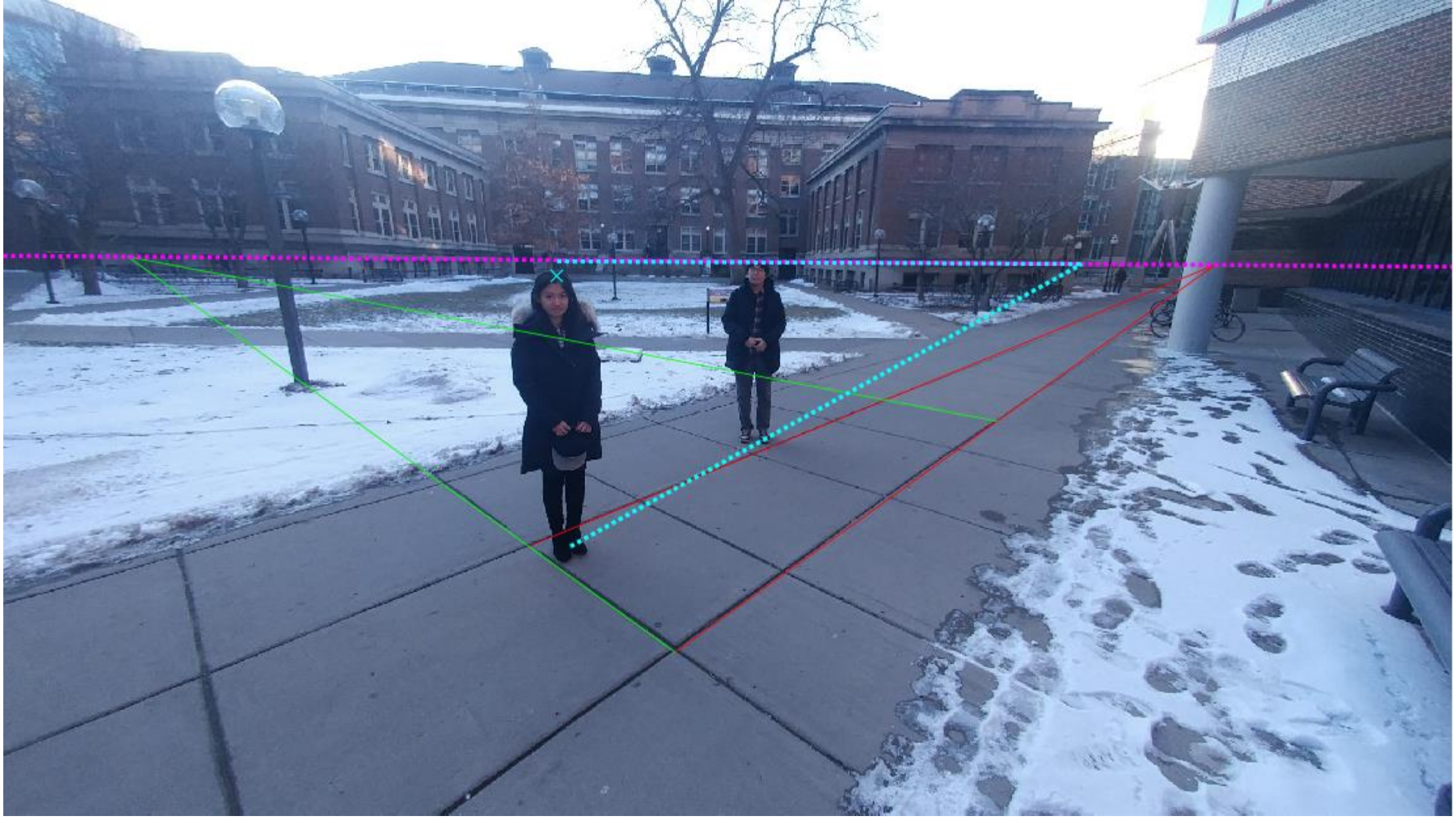


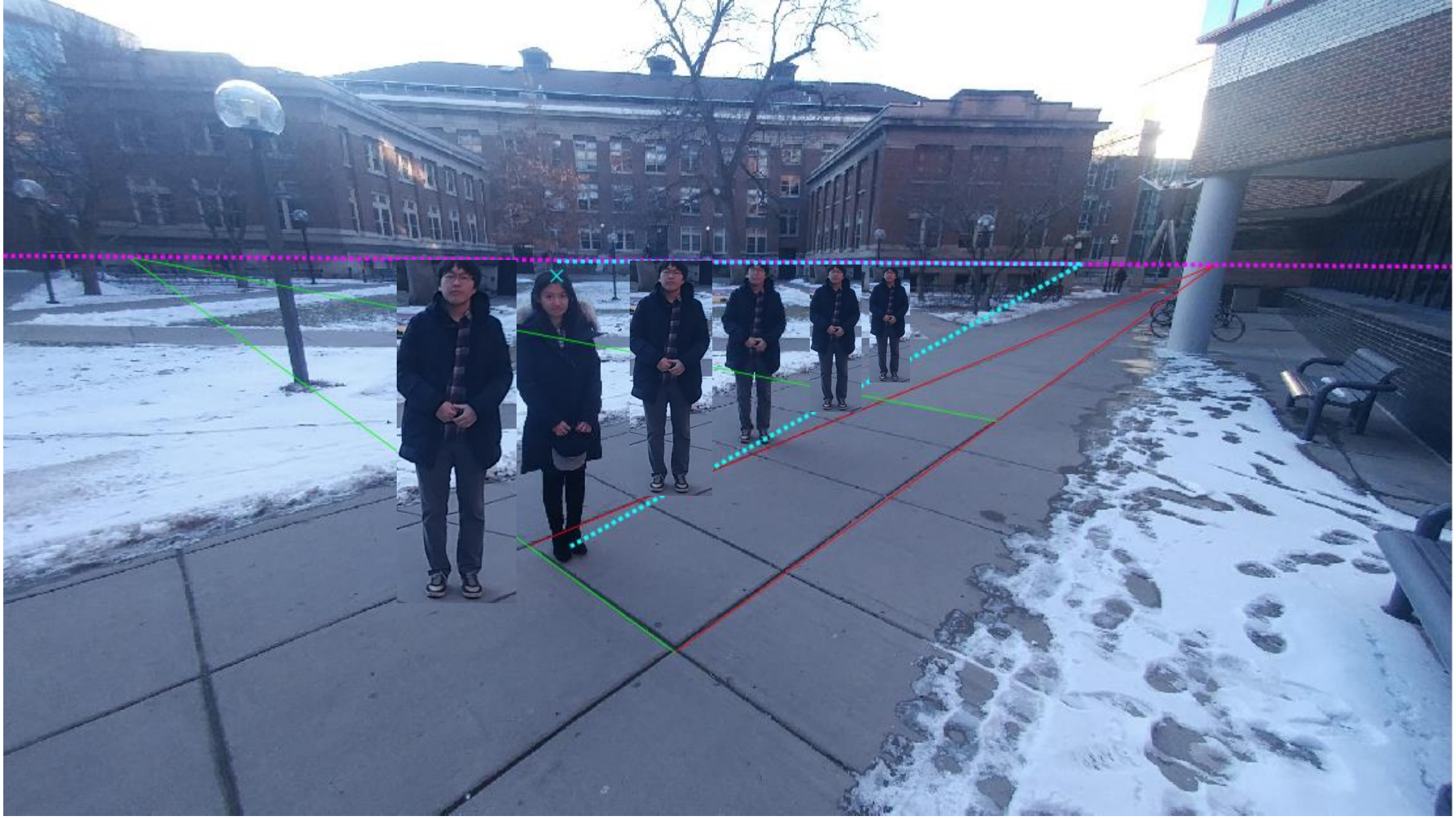

Horizon vanishing line

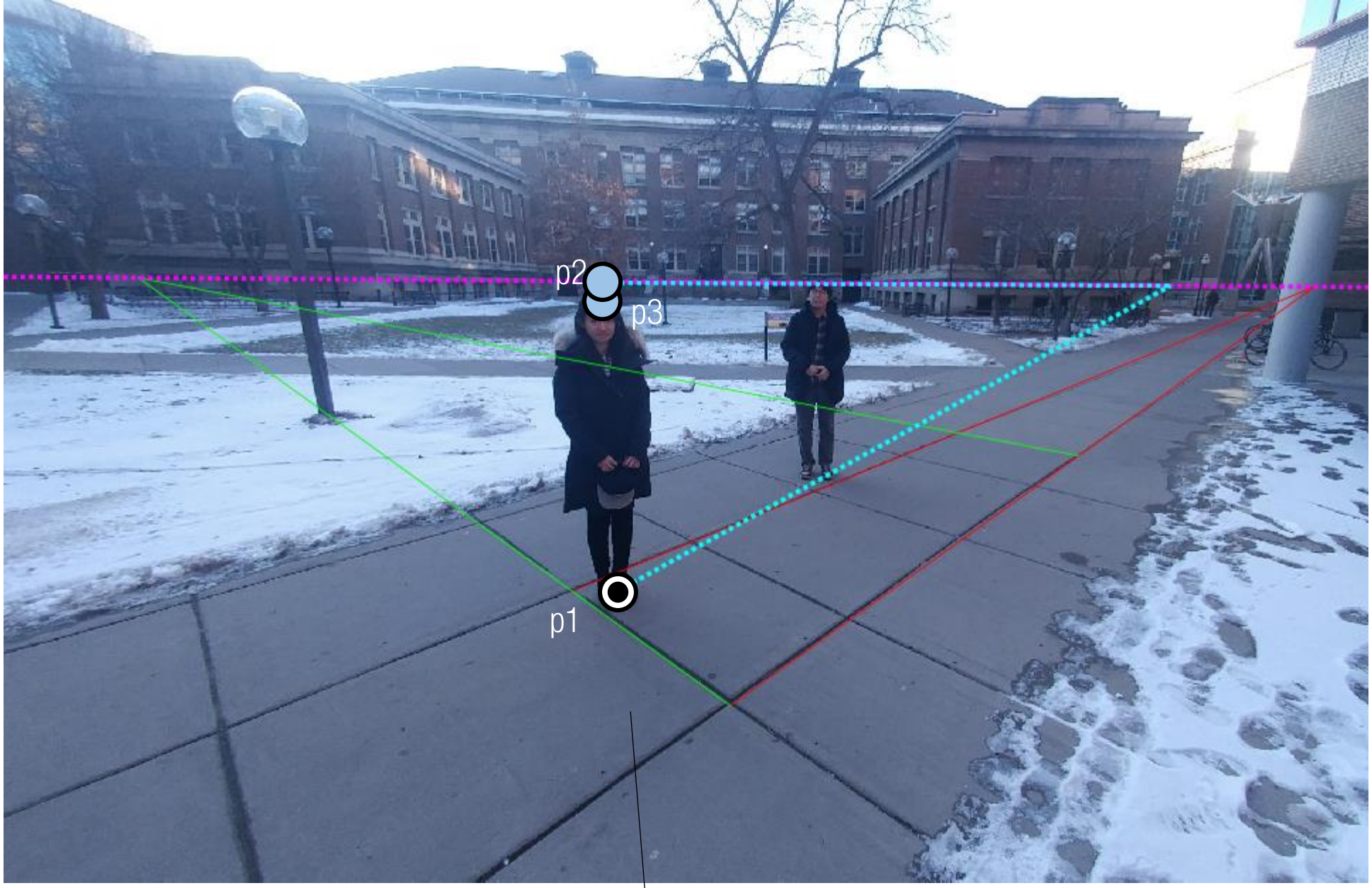


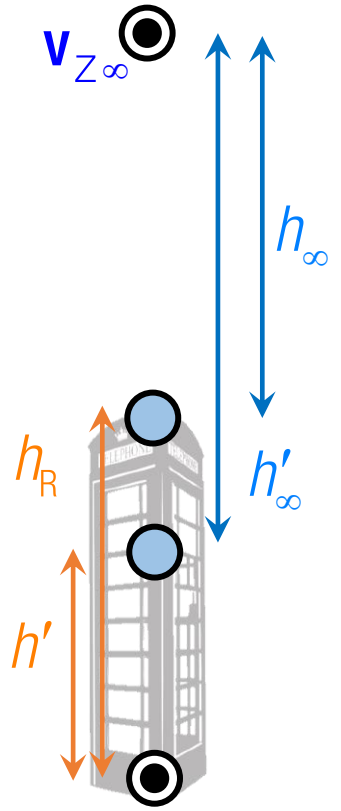
Lalonde et al. “Photo Clip Art”, SIGGRAPH 2007





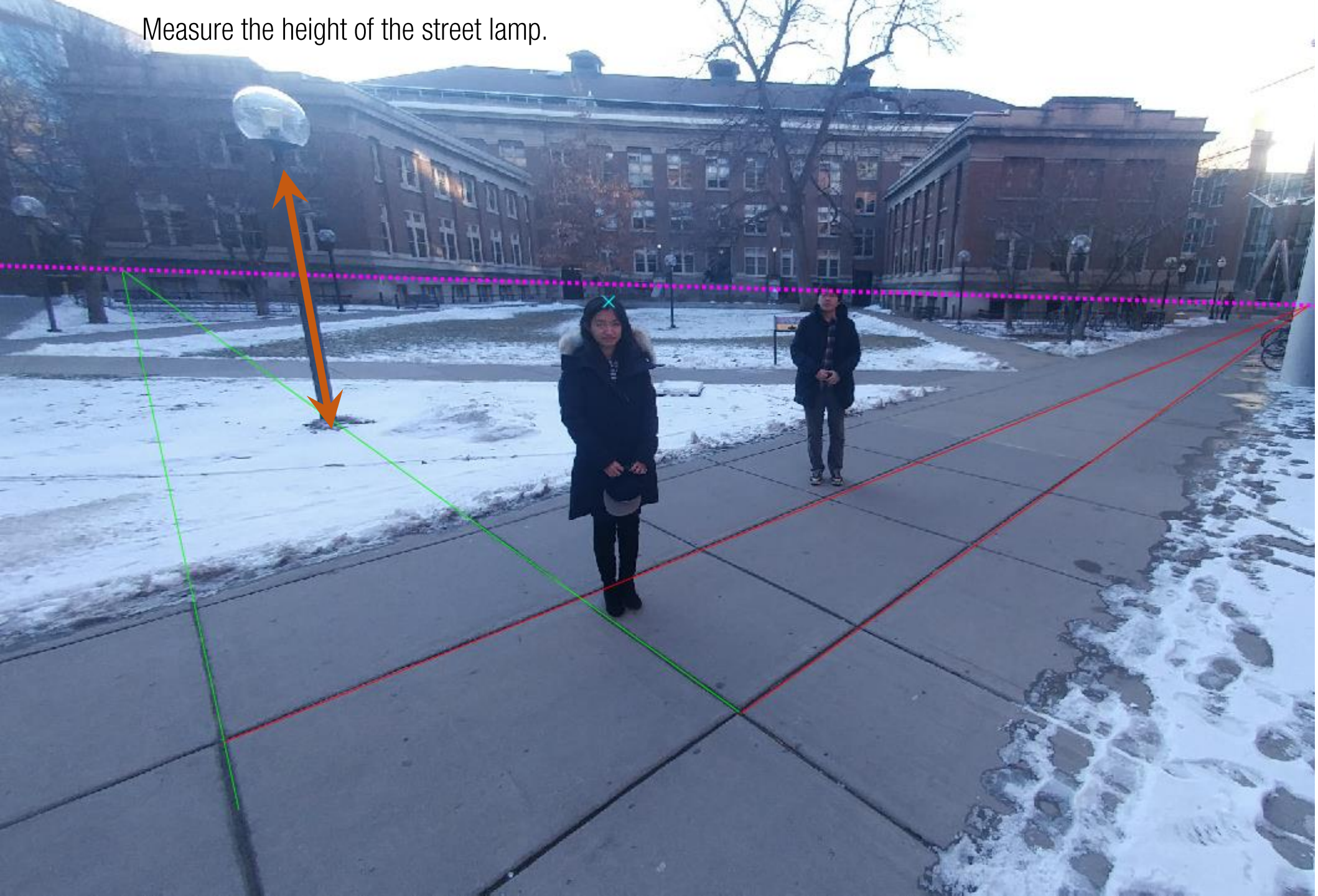


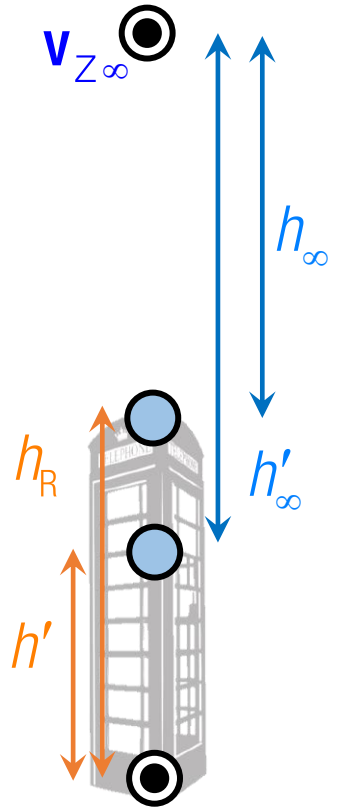




$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

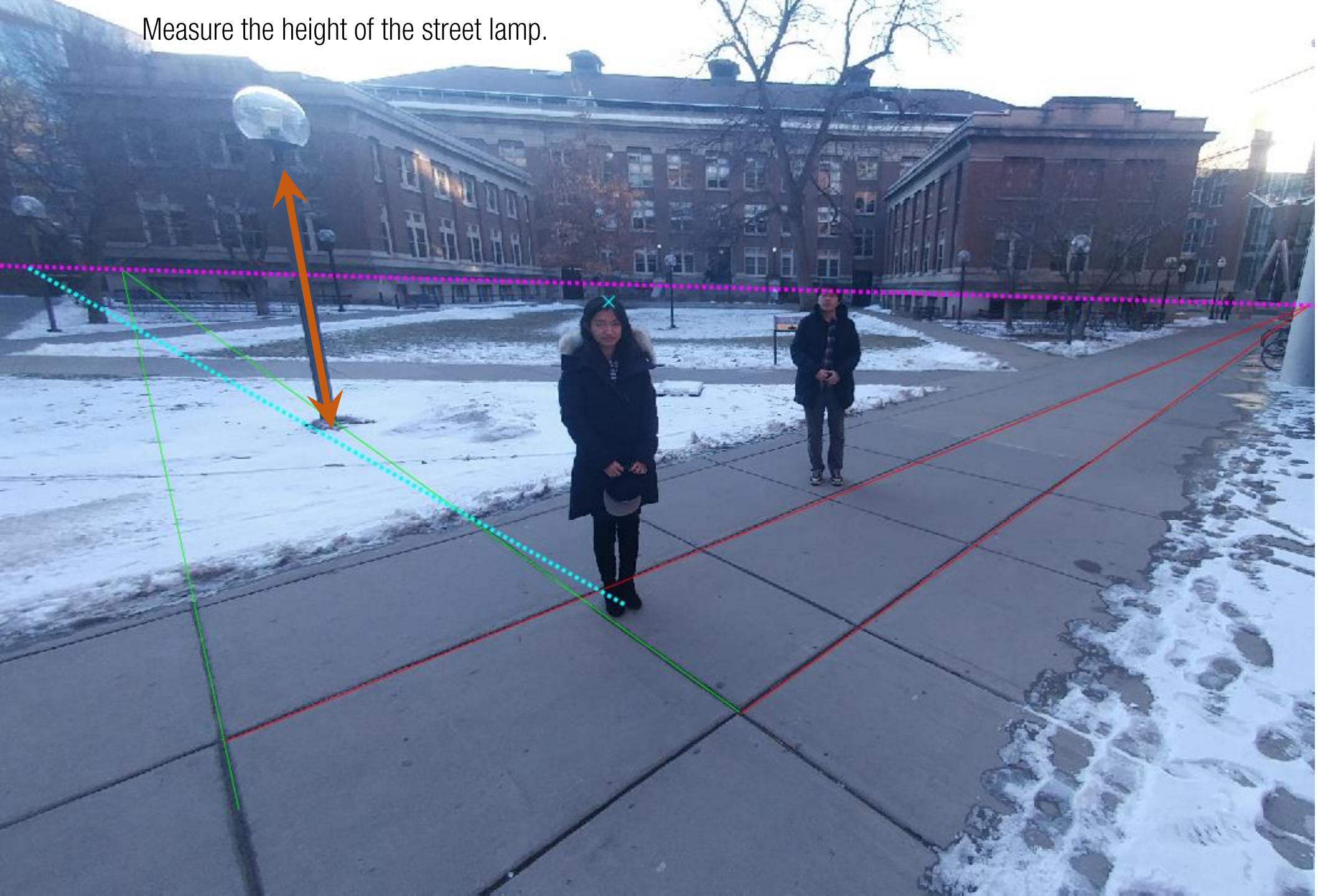
Measure the height of the street lamp.

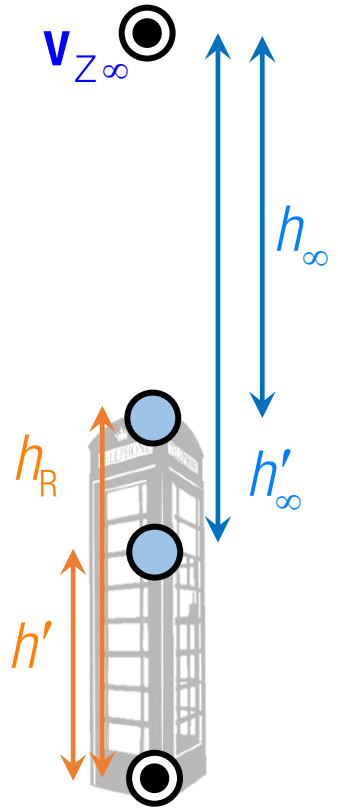




$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

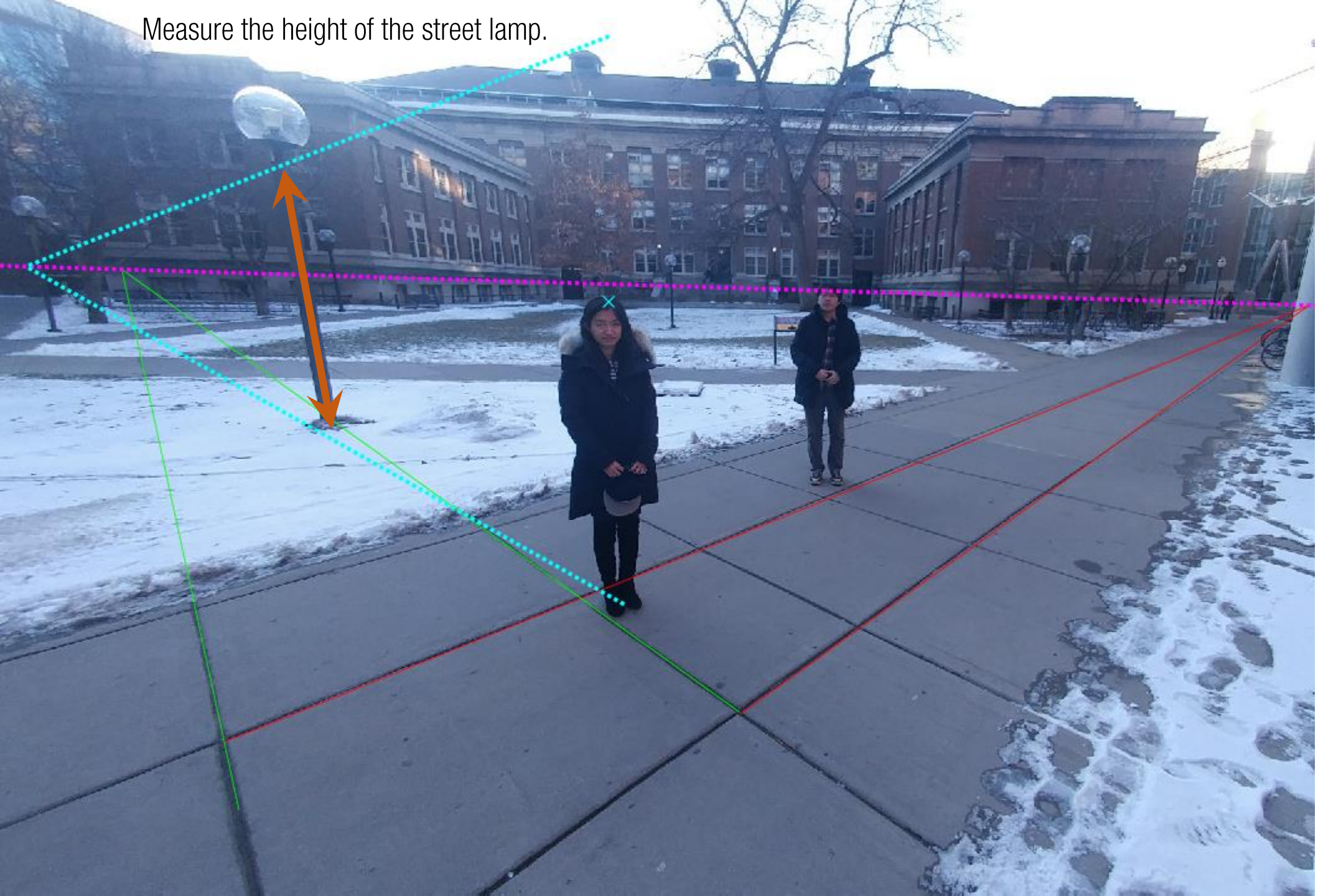
Measure the height of the street lamp.

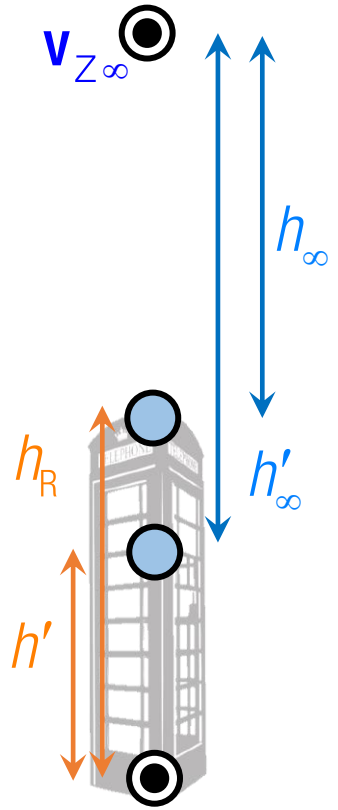




$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

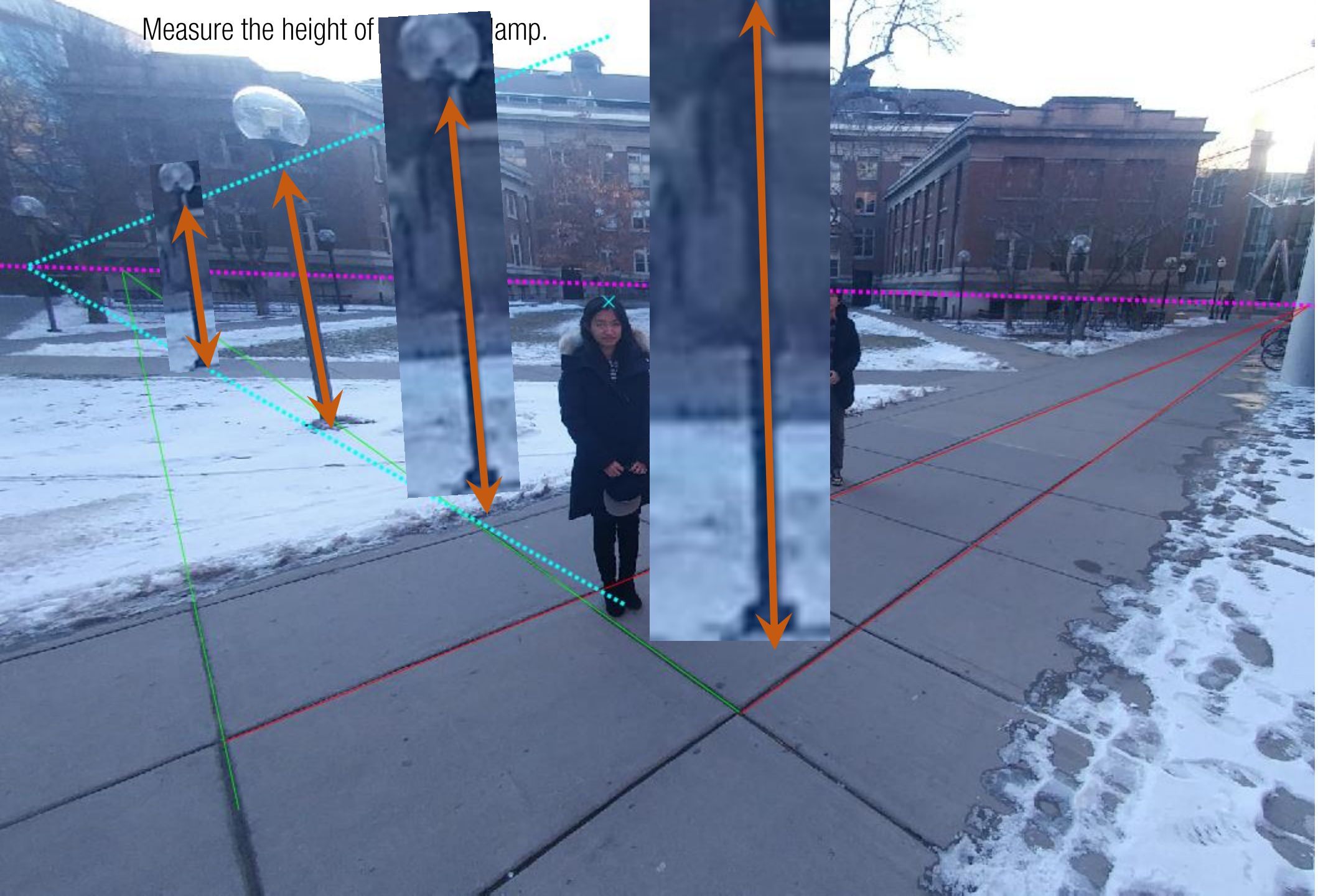
Measure the height of the street lamp.

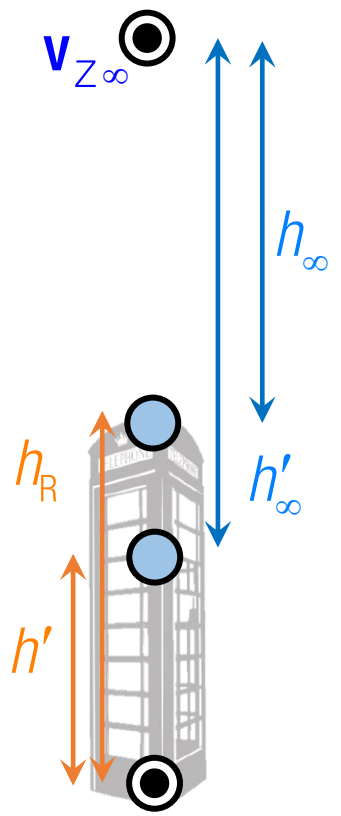




$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

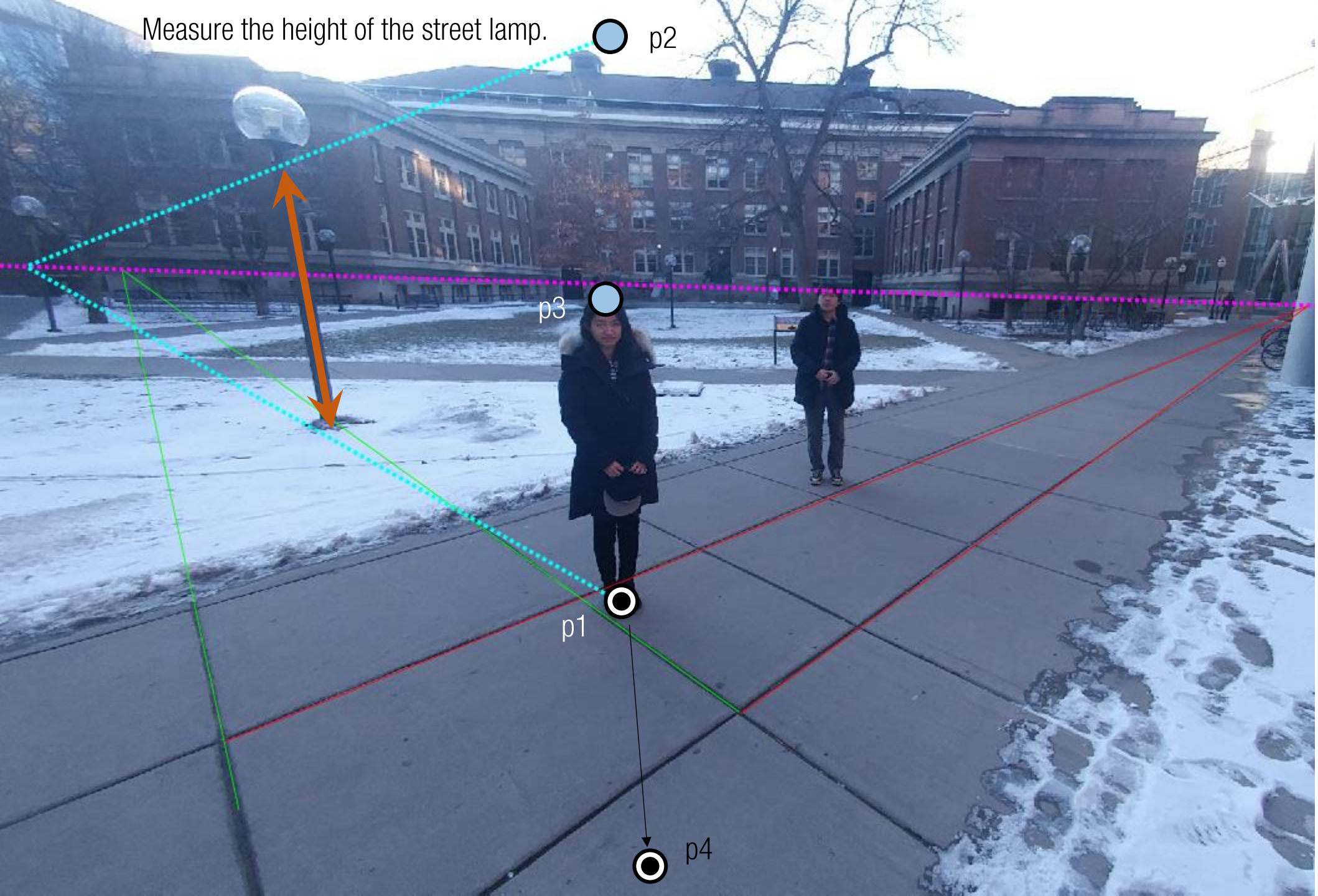
Measure the height of amp.

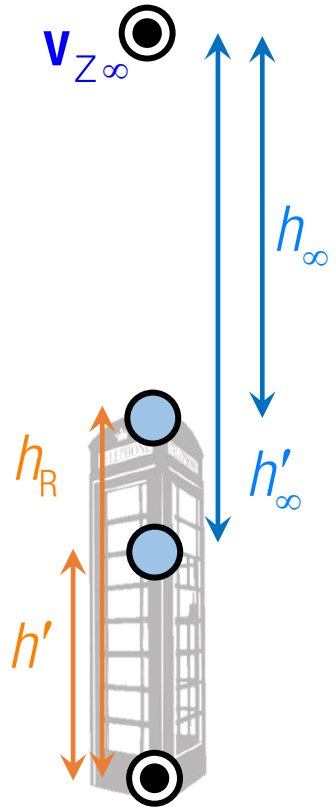




$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

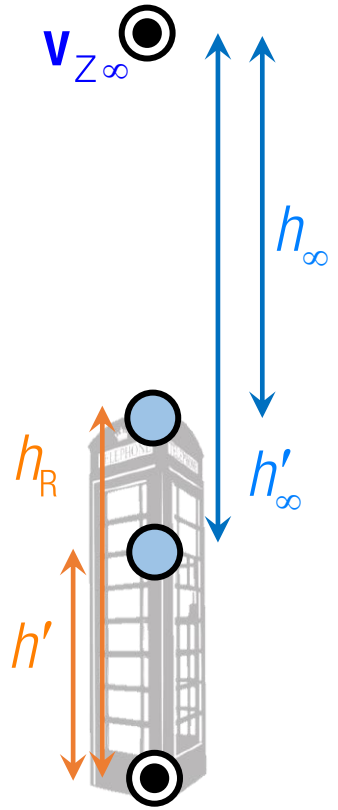
Measure the height of the street lamp.





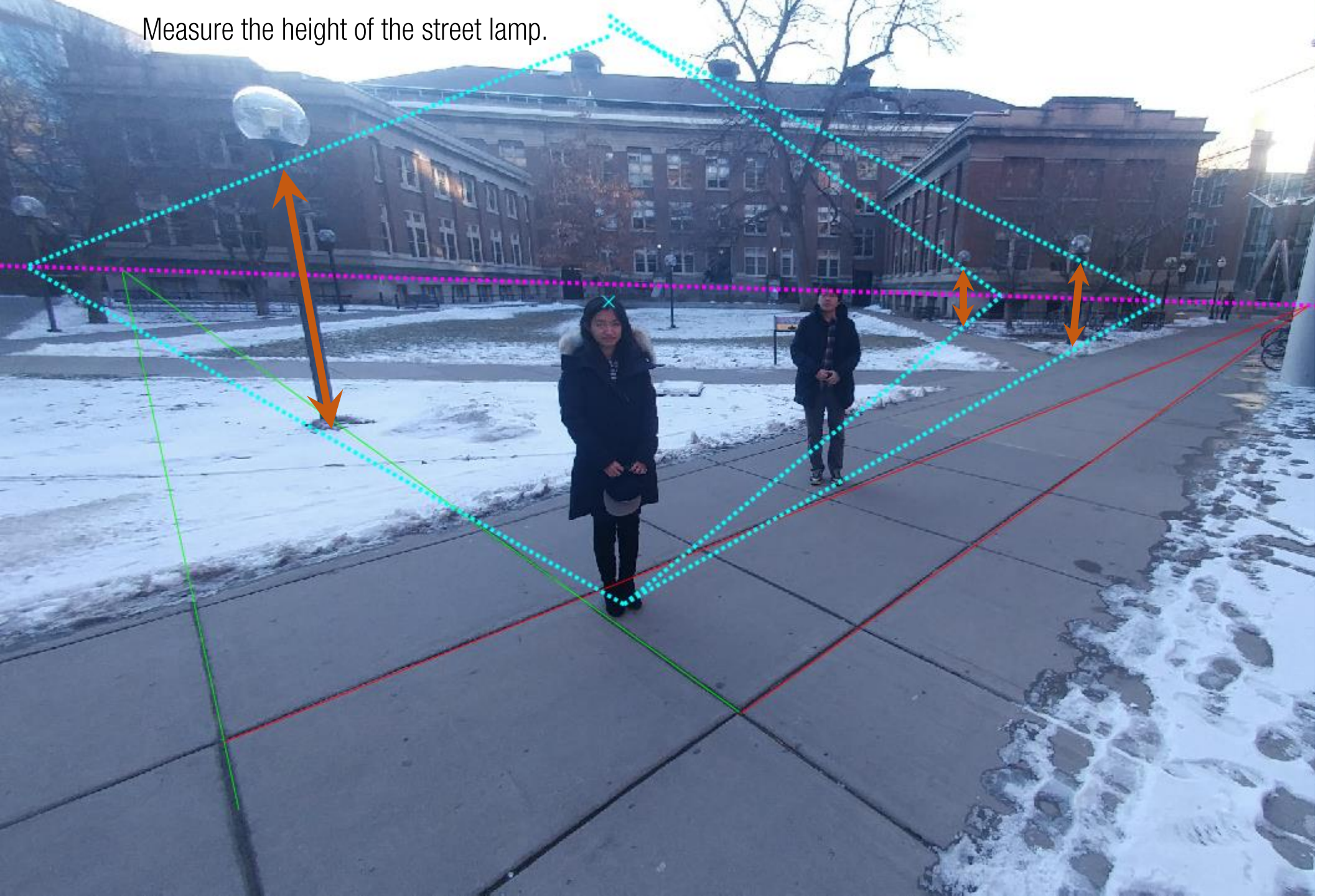
$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

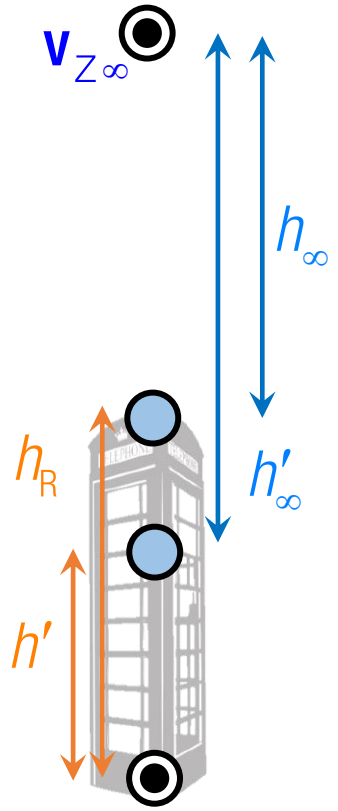
Measure the height of the street lamp.



$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

Measure the height of the street lamp.



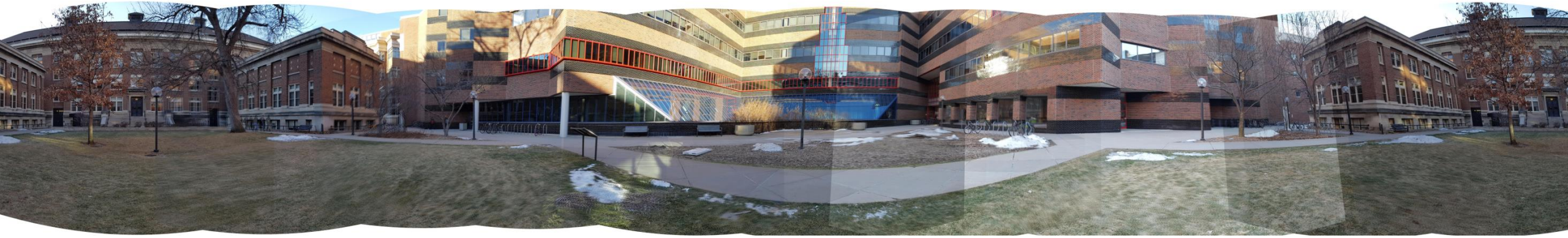


$$H = H_R \frac{h' h_\infty}{h_R h'_\infty}$$

Measure the height of the street lamp.

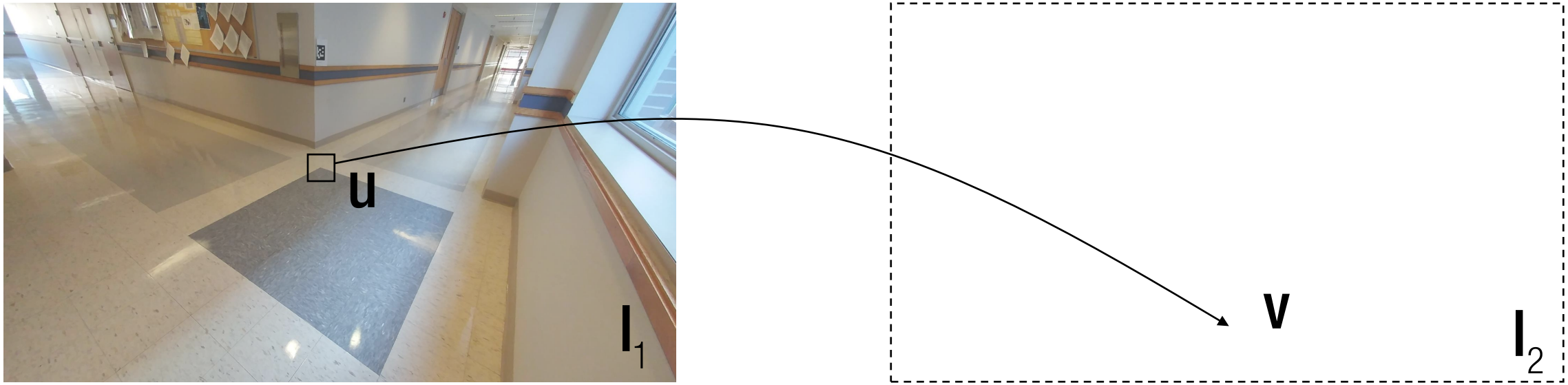
H = 2.8108m

Image Transform



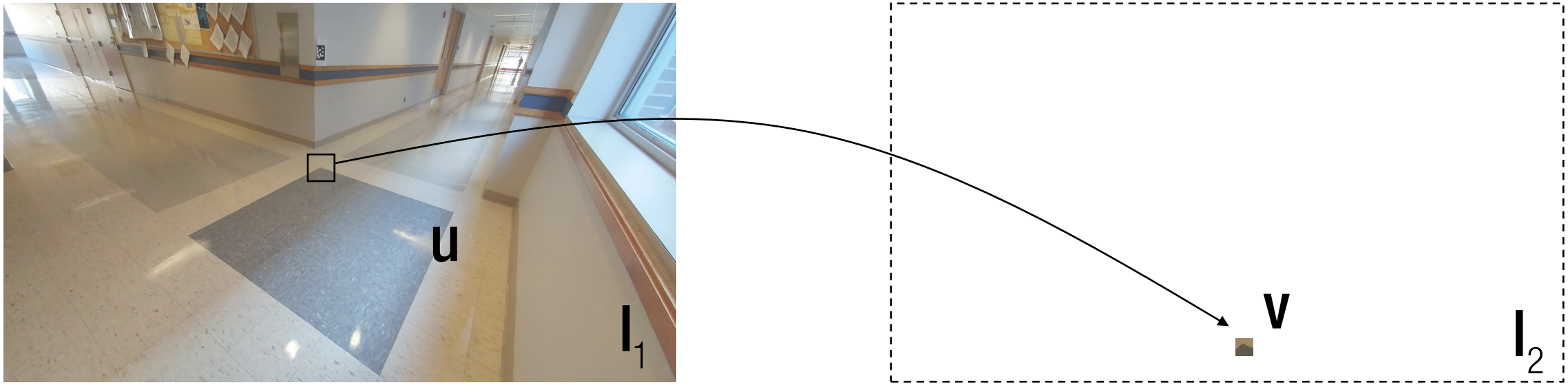
Panorama Image (Keller+Lind Hall)

Image Warping (Coordinate Transform)



$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$

Image Warping (Coordinate Transform)



$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$: Pixel transport

Image Warping (Coordinate Transform)

$$I_2(v) = I_1(u)$$: Pixel transport

Cf) Image Filtering (Pixel Transform)

$$\mathbf{I}_2 = g(\mathbf{I}_1)$$: Pixel transform

Uniform Scaling

$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$

Uniform Scaling

$$\mathbf{I}_2(\mathbf{v}) = \mathbf{I}_1(\mathbf{u})$$

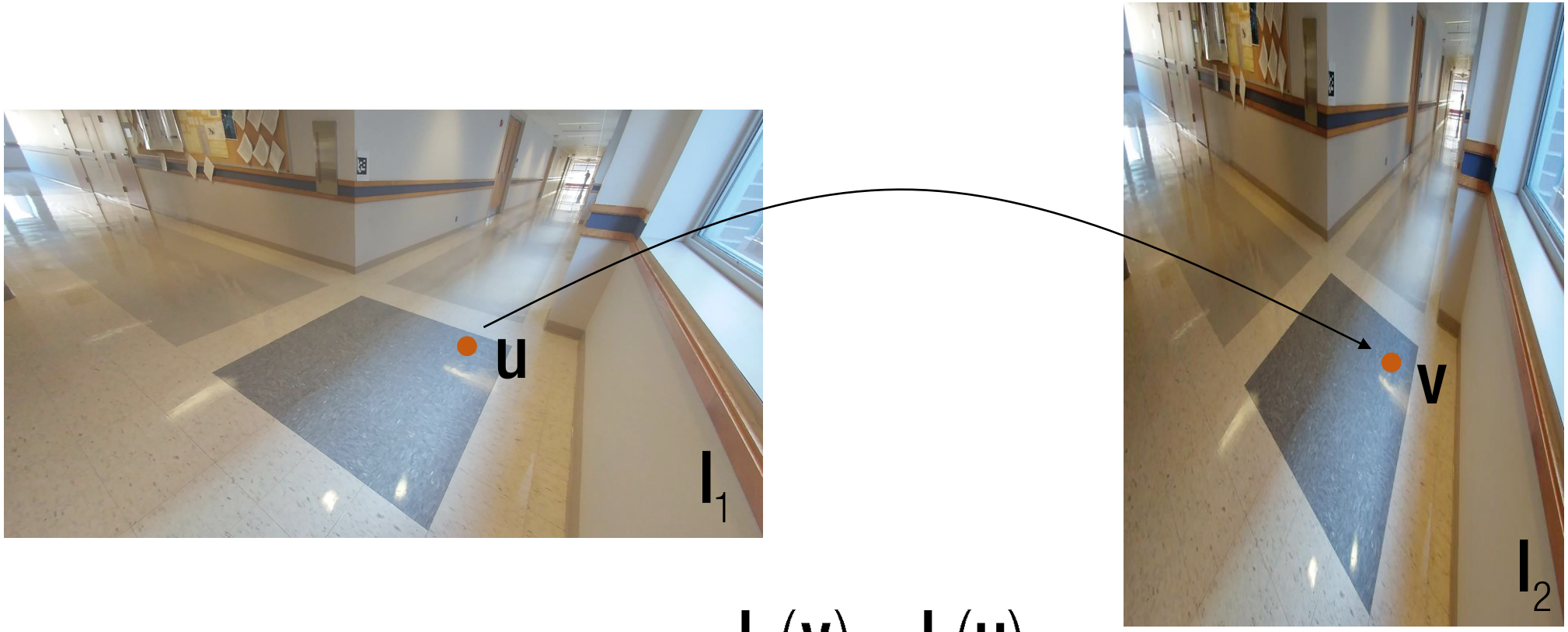
$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \quad ? \quad \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Uniform Scaling

$$\mathbf{I}_2(\mathbf{v}) = \mathbf{I}_1(\mathbf{u})$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & \\ & s_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \qquad s_x = s_y$$

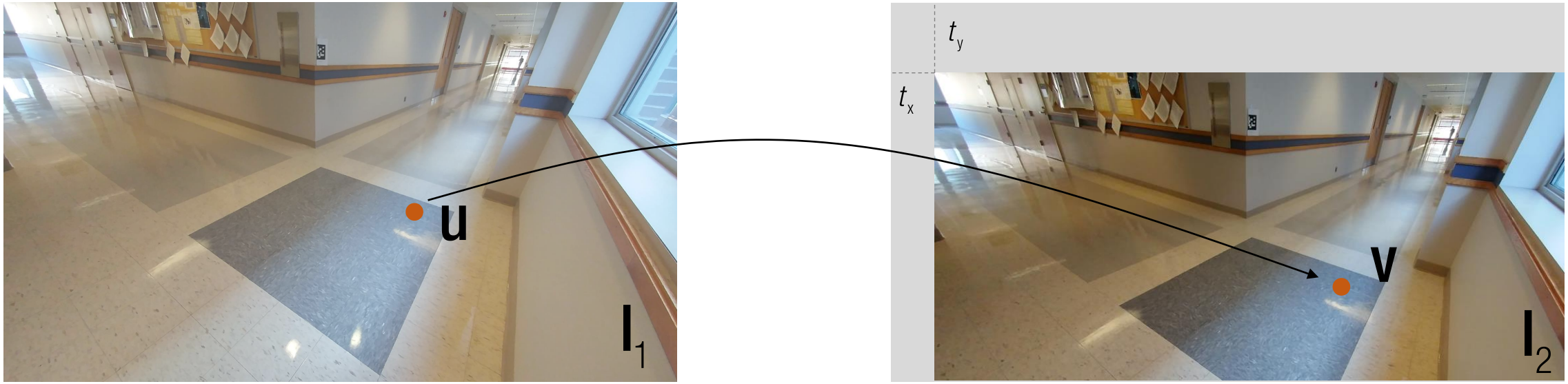
Aspect Ratio Change



$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & & \\ & s_y & \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \qquad s_x \neq s_y$$

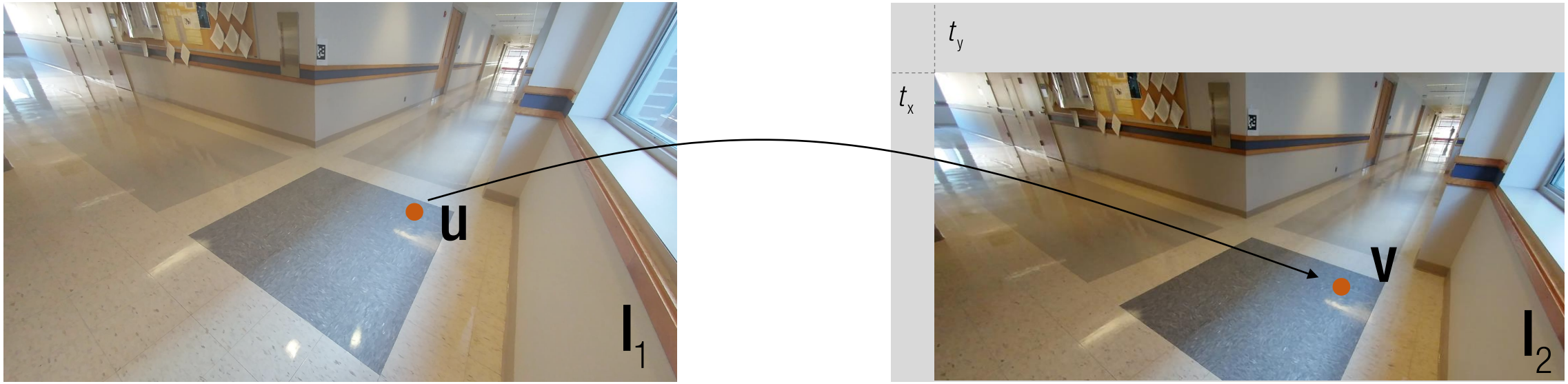
Translation



$$\mathbf{I}_2(\mathbf{v}) = \mathbf{I}_1(\mathbf{u})$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \; ? \; \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Translation



$$\mathbf{I}_2(\mathbf{v}) = \mathbf{I}_1(\mathbf{u})$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & & t_x \\ & 1 & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Rotation

$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$

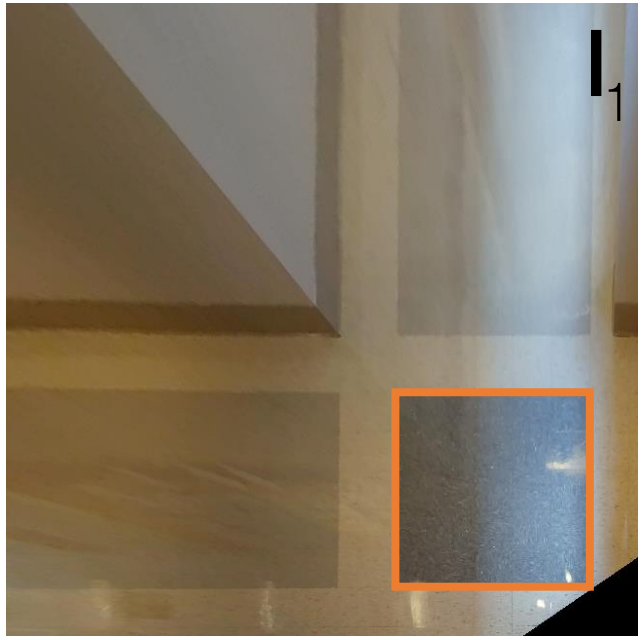
$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \quad ? \quad \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

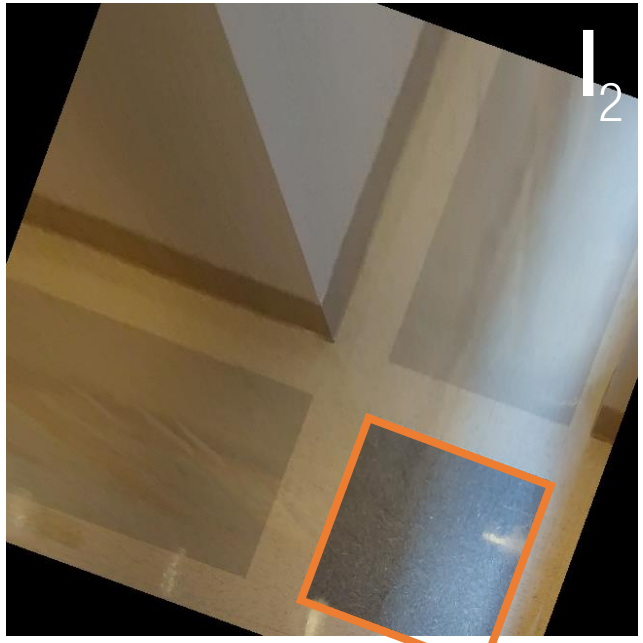
Rotation

$$I_2(\mathbf{v}) = I_1(\mathbf{u})$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & \\ \sin\theta & \cos\theta & \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean Transform SE(3)

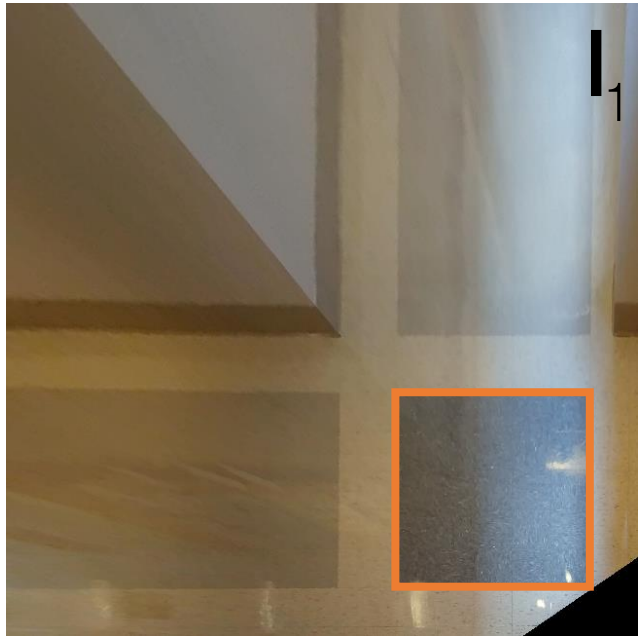


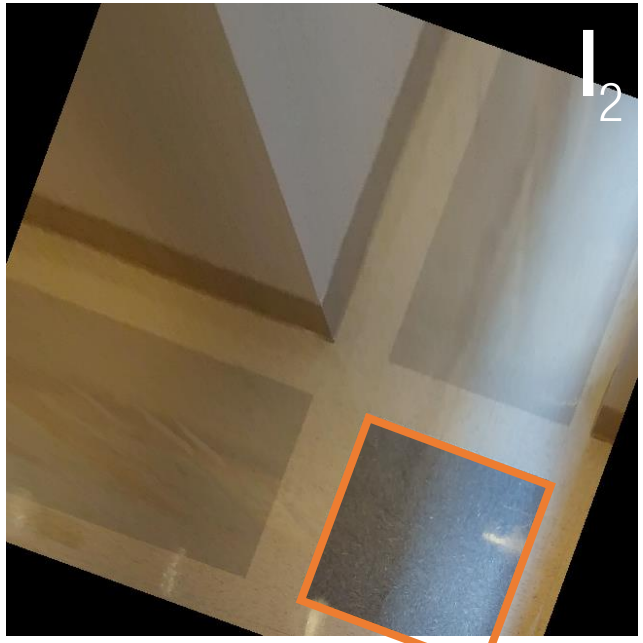


Rotation around the center of image

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \quad ? \quad \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean Transform SE(3)

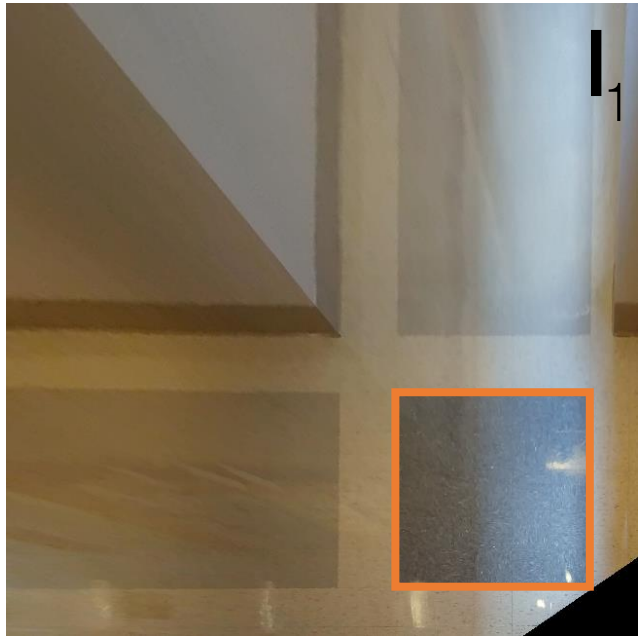


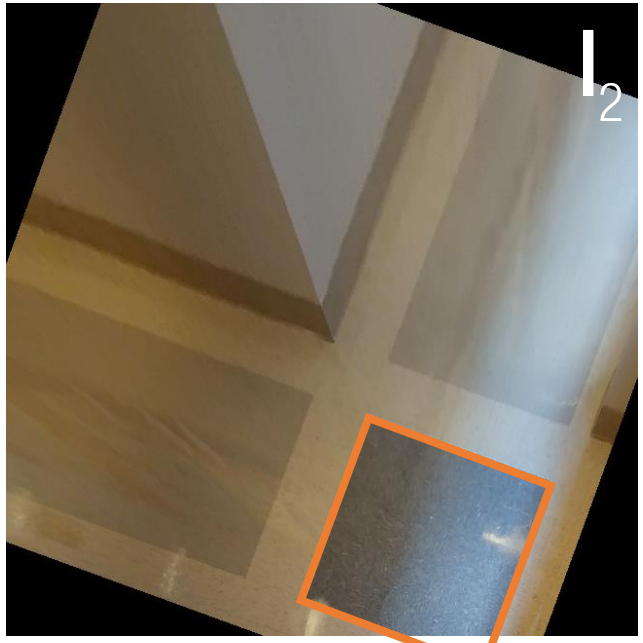


Rotation around the center of image

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean Transform SE(3)





Rotation around the center of image

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

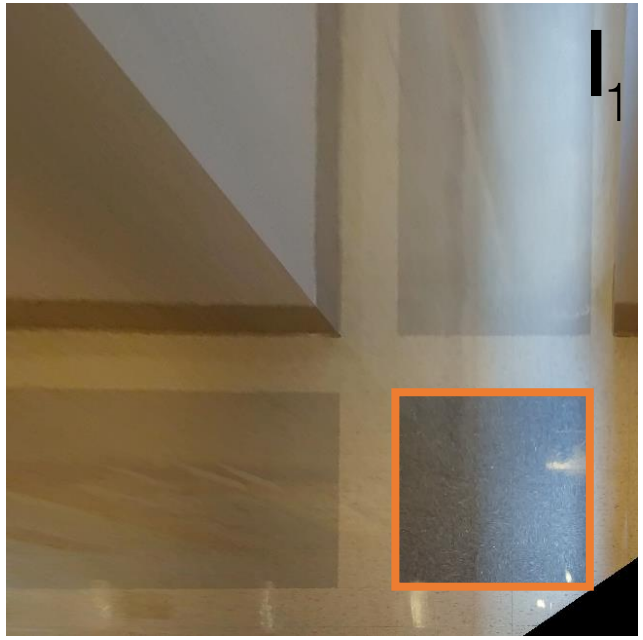
Invariant properties

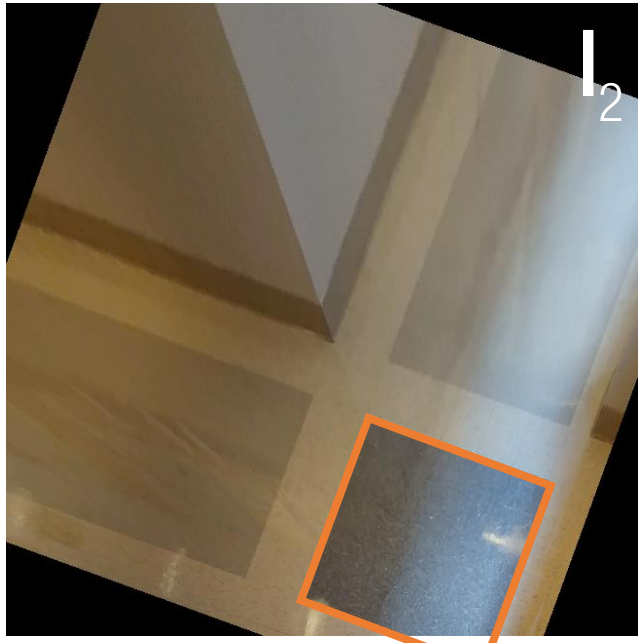
- Length
- Angle
- Area

Degree of freedom

3 (2 translation+1 rotation)

Euclidean Transform SE(3)





Rotation around the center of image

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} \longrightarrow \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix}$$

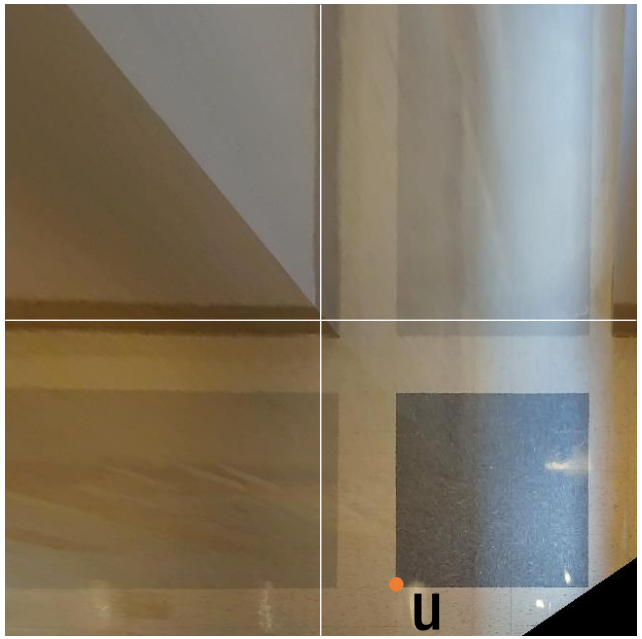
Invariant properties

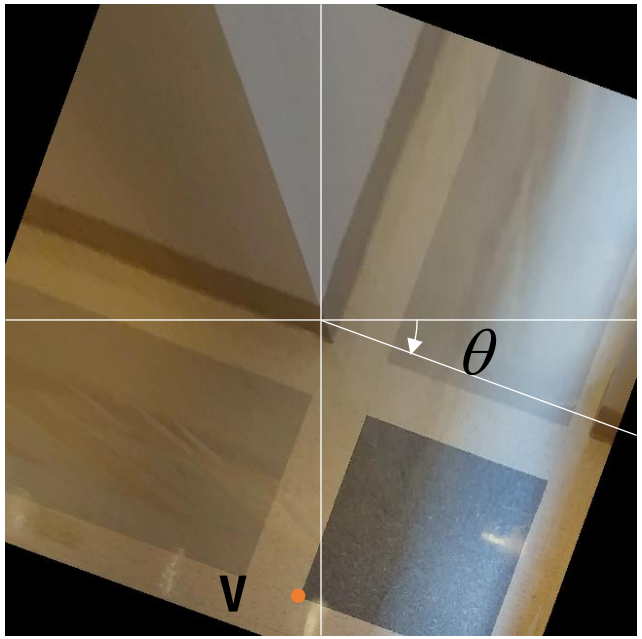
- Length
- Angle
- Area

Degree of freedom

3 (2 translation+1 rotation)

Euclidean Transform SE(3)



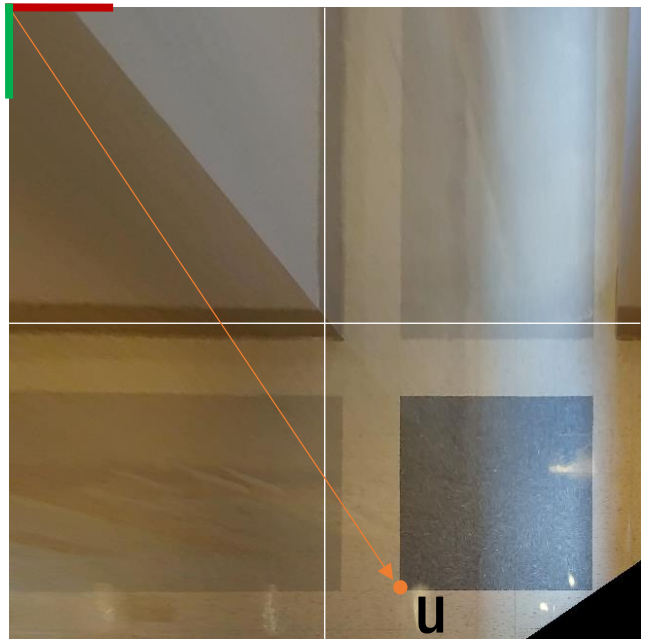


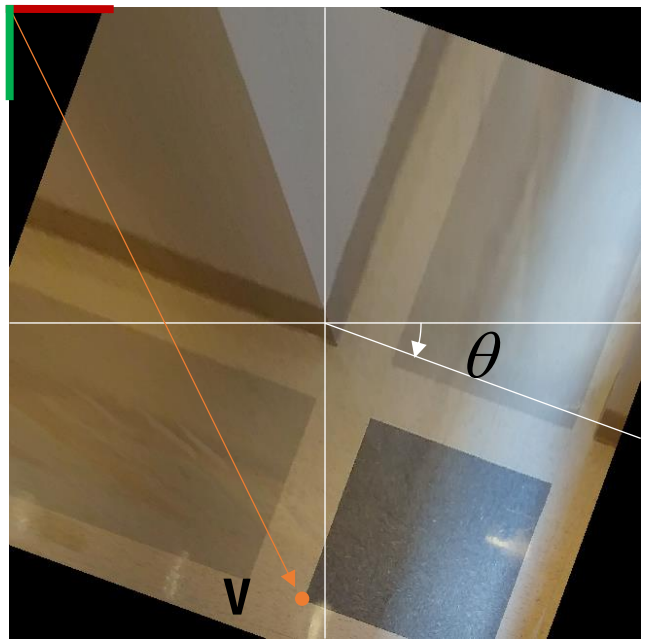
Rotate about the image center

$$\begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \longrightarrow \mathbf{v} = \mathbf{R}\mathbf{u} + \mathbf{t}$$

$\mathbf{t}$?

Euclidean Transform SE(3)



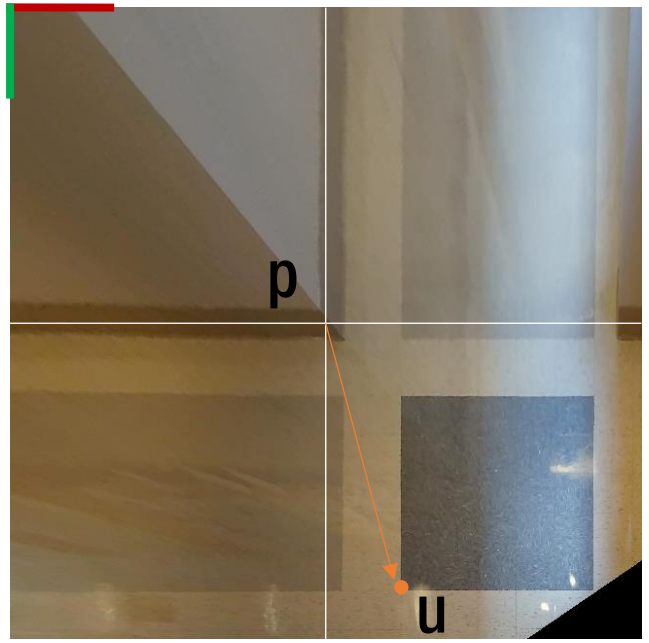


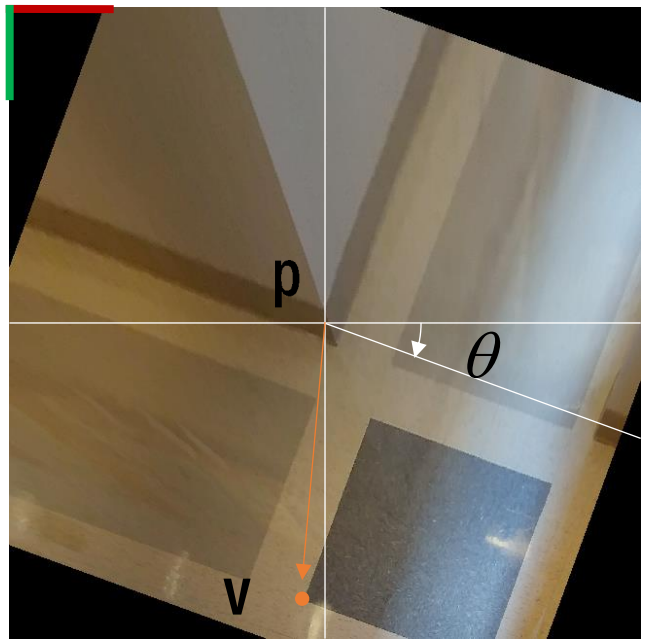
Rotate about the image center

$$\begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \longrightarrow \mathbf{v} = \mathbf{R}\mathbf{u} + \mathbf{t}$$

t ?

Euclidean Transform SE(3)

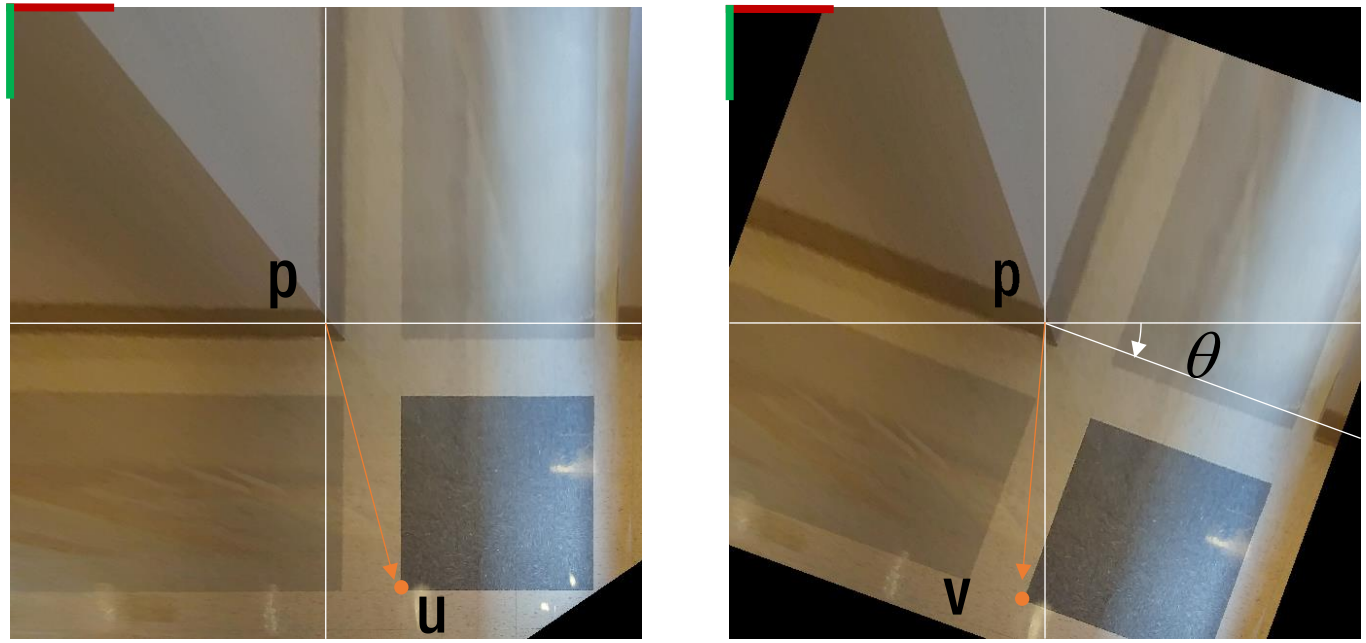




Rotate about the image center

$$\begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \longrightarrow \mathbf{v} = \mathbf{R}\mathbf{u} + \mathbf{t}$$

Euclidean Transform SE(3)



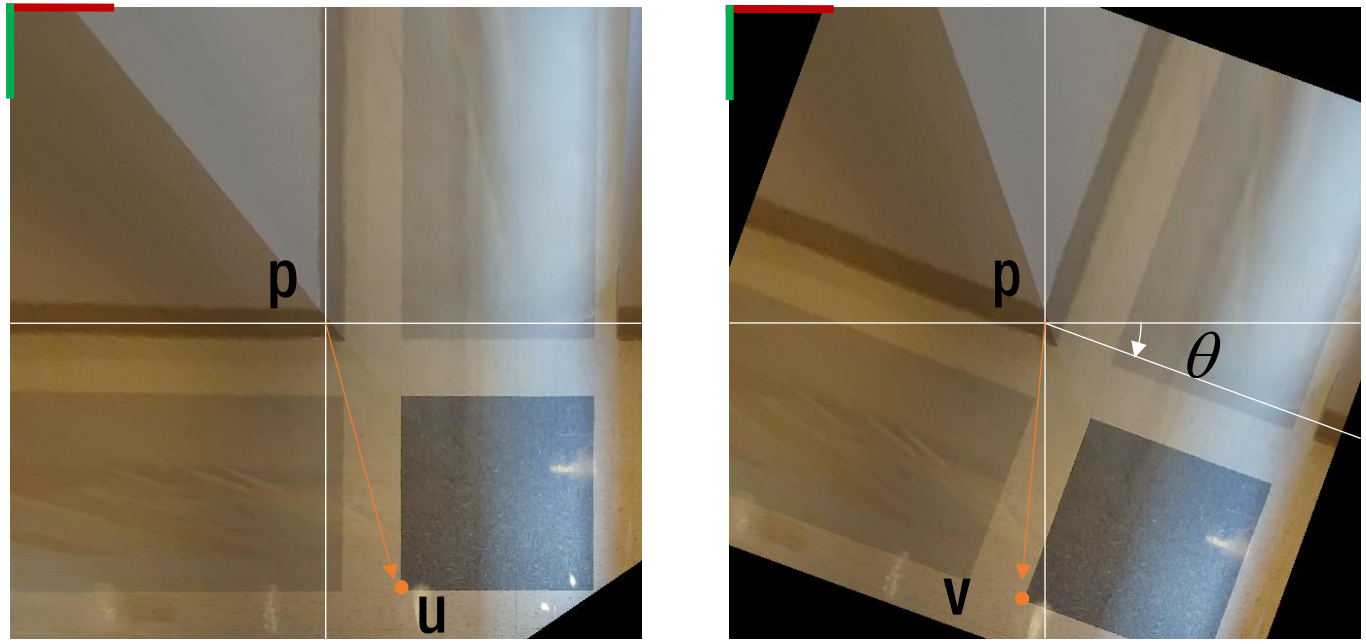
Rotate about the image center

$$\begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ 1 \end{bmatrix} \longrightarrow \mathbf{v} = \mathbf{R}\mathbf{u} + \mathbf{t}$$

$$\bar{\mathbf{u}} = \mathbf{u} - \mathbf{p} \qquad \bar{\mathbf{v}} = \mathbf{v} - \mathbf{p}$$

$$\longrightarrow \quad \bar{\mathbf{v}} = \mathbf{R}\bar{\mathbf{u}}$$

Euclidean Transform SE(3)



Rotate about the image center

$$\begin{bmatrix} v \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ 1 \end{bmatrix} \longrightarrow v = Ru + t$$

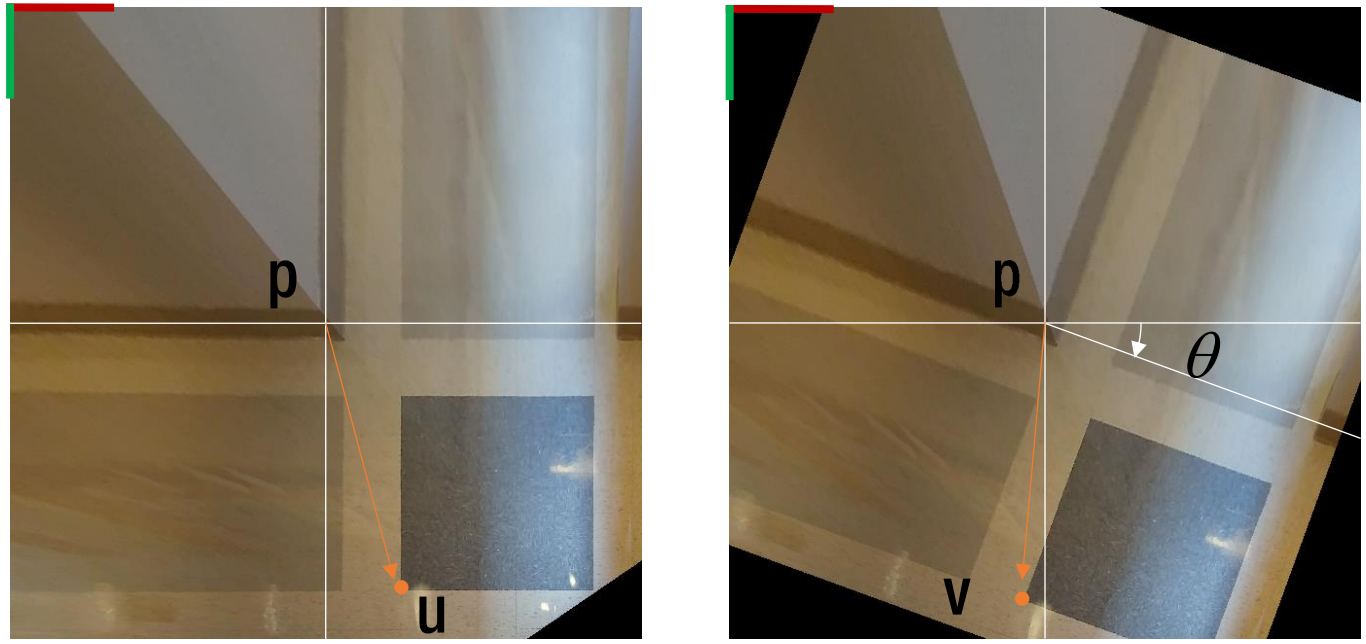
$$\bar{u} = u - p \qquad \bar{v} = v - p$$

$$\longrightarrow \bar{v} = R\bar{u}$$

$$\longrightarrow v - p = R(u - p)$$

$$\longrightarrow v = Ru - Rp + p$$

Euclidean Transform SE(3)



Rotate about the image center

$$\begin{bmatrix} v \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ 1 \end{bmatrix} \longrightarrow v = Ru + t$$

$$\bar{u} = u - p \qquad \bar{v} = v - p$$

$$\longrightarrow \bar{v} = R\bar{u}$$

$$\longrightarrow v - p = R(u - p)$$

$$\longrightarrow v = Ru - Rp + p$$

$$\longrightarrow t = -Rp + p$$

Euclidean Transform SE(3)

```
im = imread('rect.png');

theta = 20/180*pi;

R = [cos(theta) -sin(theta);
     sin(theta) cos(theta)];
p = [size(im,2)/2; size(im,1)/2];
```

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Rotation around the center of image

Euclidean Transform SE(3)

RectificationViaEuclidean.m

```
im = imread('rect.png');

theta = 20/180*pi;

R = [cos(theta) -sin(theta);
     sin(theta) cos(theta)];
p = [size(im,2)/2; size(im,1)/2];

T = [R -R*t+t;0 0 1];

im_warped = ImageWarpingEuclidean(im, T);
```

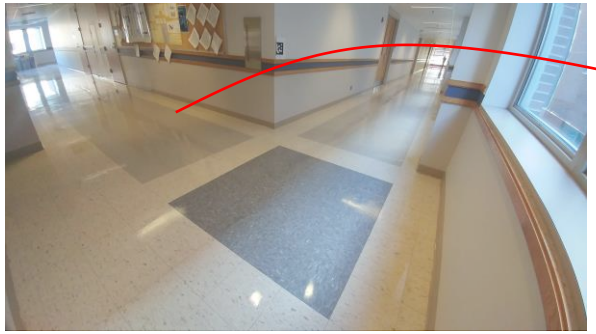
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & -\mathbf{Rp}+\mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Rotation around the center of image

Recall: MATLAB Efficient Image Transform

Distorted image

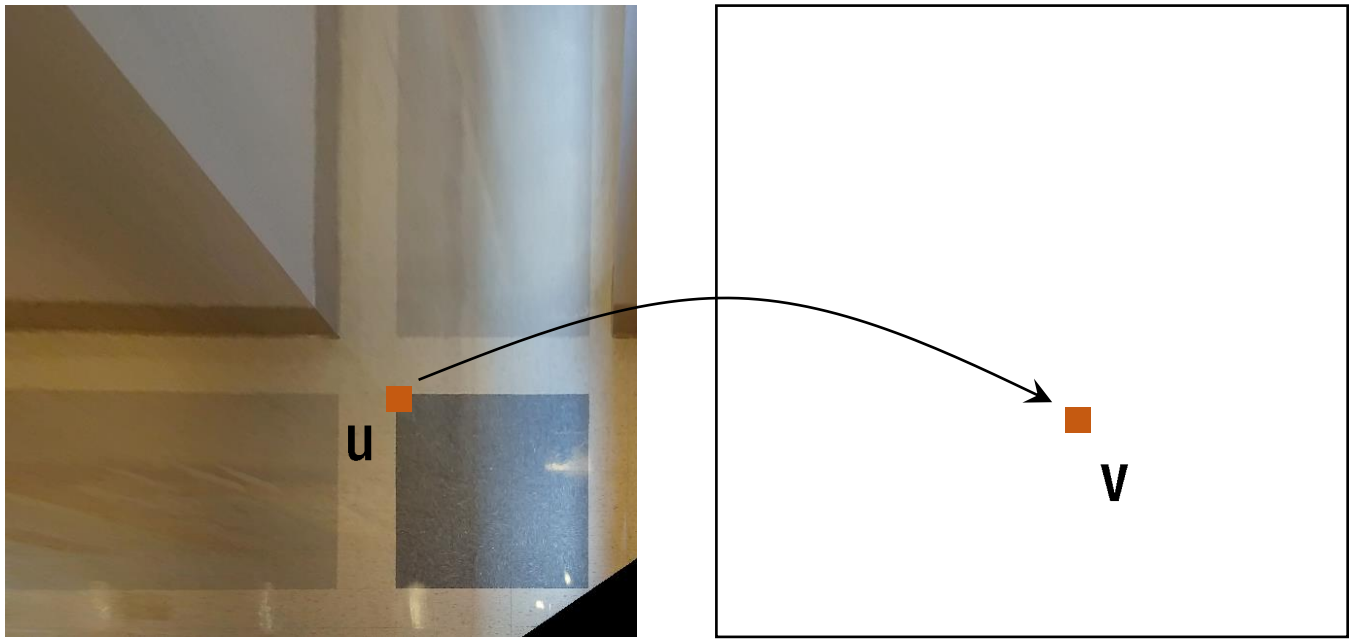


Undistorted image


```
[[X, Y] = meshgrid(1:(size(im,2)), 1:(size(im,1)));
h = size(X, 1); w = size(X,2);

X_n = (X-px)/f;
Y_n = (Y-py)/f;

r_u = sqrt(X_n.^2+Y_n.^2);

L = 1 + k * r_u.^2;

X_dist_n = X_n.* L;
Y_dist_n = Y_n.* L;

imUndistortion(:,:,1) = reshape(interp2(im(:,:,1), X_dist(:), Y_dist(:)), [h, w]);
imUndistortion(:,:,2) = reshape(interp2(im(:,:,2), X_dist(:), Y_dist(:)), [h, w]);
imUndistortion(:,:,3) = reshape(interp2(im(:,:,3), X_dist(:), Y_dist(:)), [h, w]);
```

UndistortImageRadial.m

Euclidean Transform SE(3)

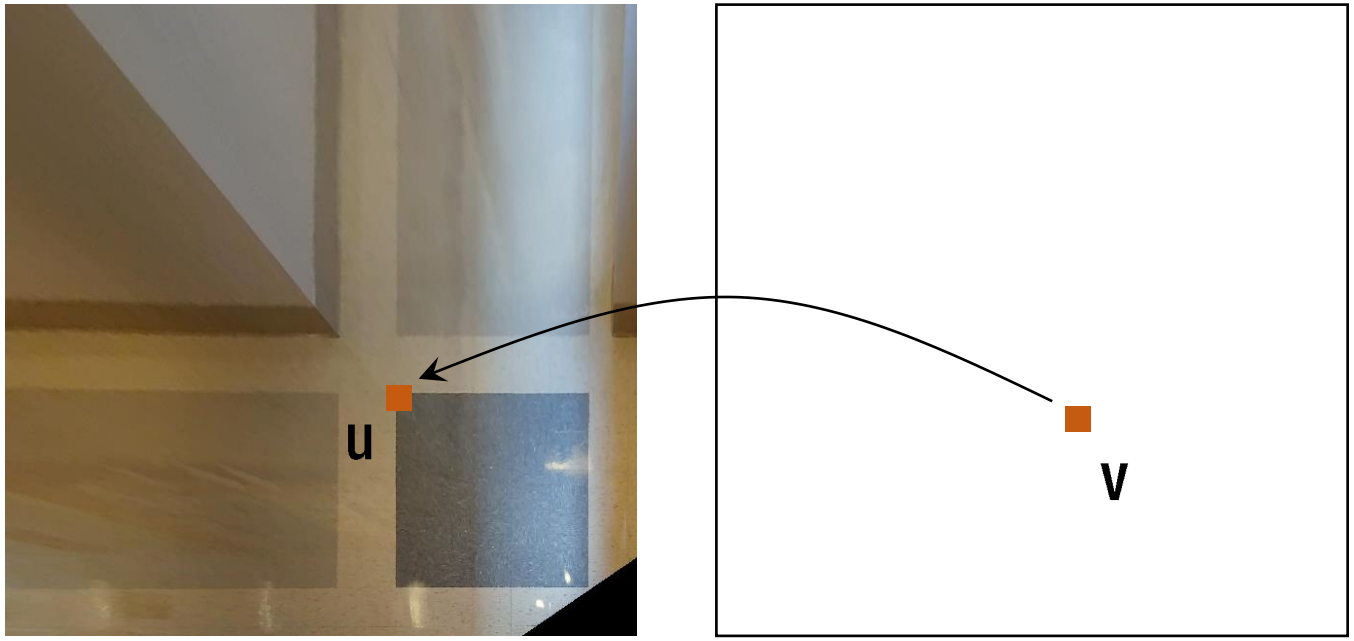


$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

ImageWarpingEuclidean.m

```
function im_warped = ImageWarpingEuclidean(im, H)

im = double(im);
```

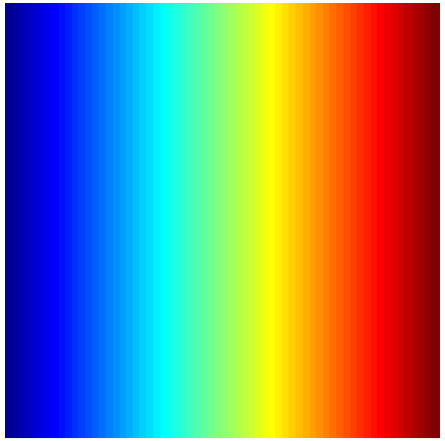
Euclidean Transform SE(3)



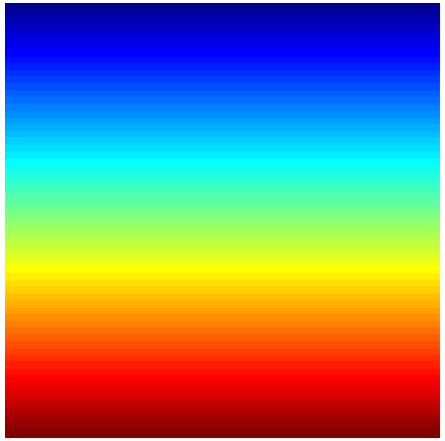
$$\mathbf{H}^{-1}\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

ImageWarpingEuclidean.m

```
function im_warped = ImageWarpingEuclidean(im, H)

im = double(im);

H = inv(H);

[v_x, v_y] = meshgrid(1:(size(im,2)), 1:(size(im,1)));
h = size(v_x, 1); w = size(v_x,2);
```

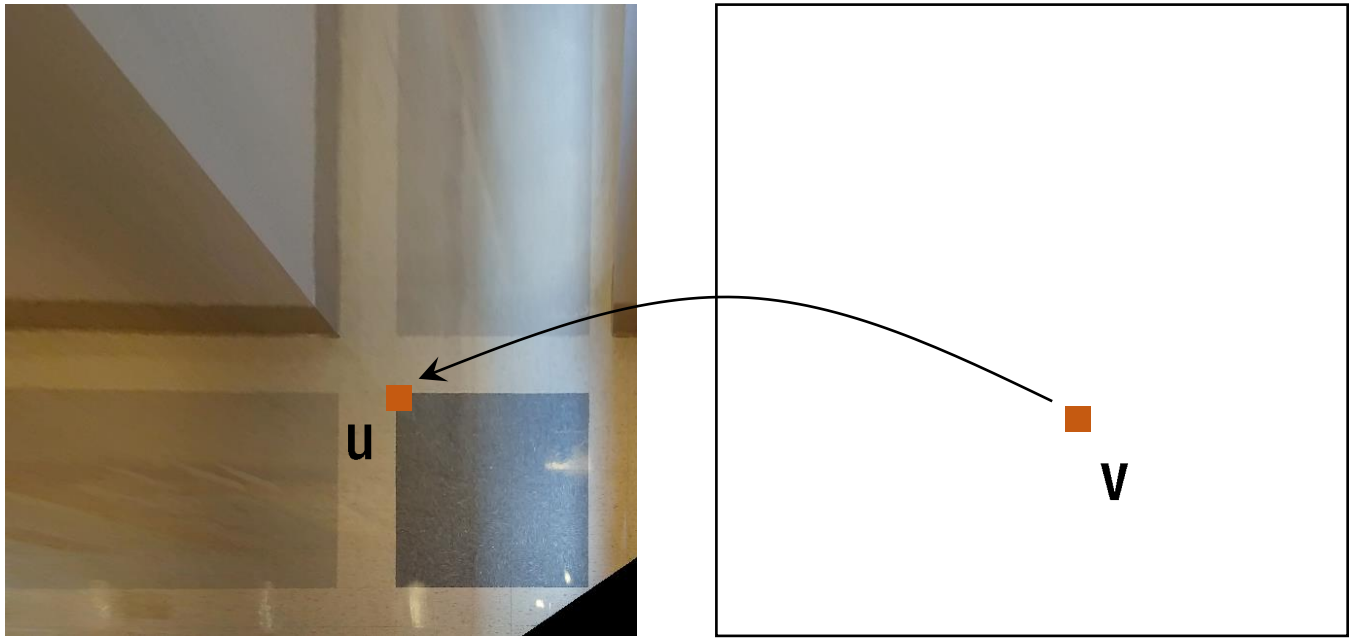


v_x



v_y

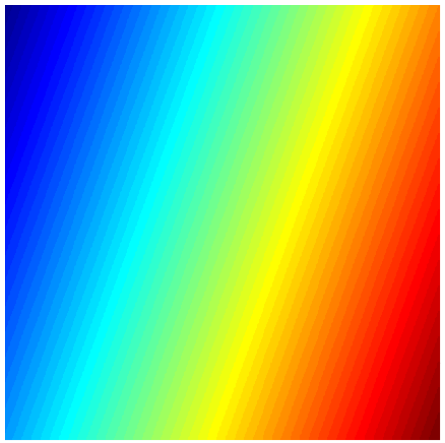
Euclidean Transform SE(3)



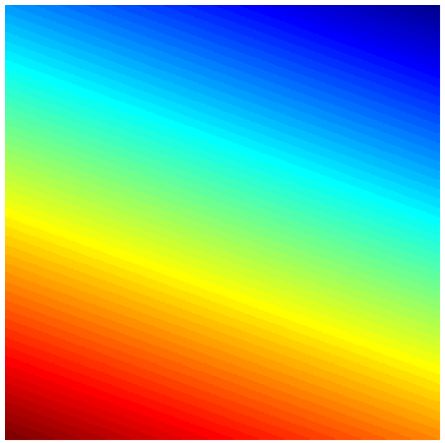
$$\mathbf{H}^{-1}\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

ImageWarpingEuclidean.m

```
function im_warped = ImageWarpingEuclidean(im, H)

im = double(im);

H = inv(H);

[v_x, v_y] = meshgrid(1:(size(im,2)), 1:(size(im,1)));
h = size(v_x, 1); w = size(v_x,2);

u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);
```

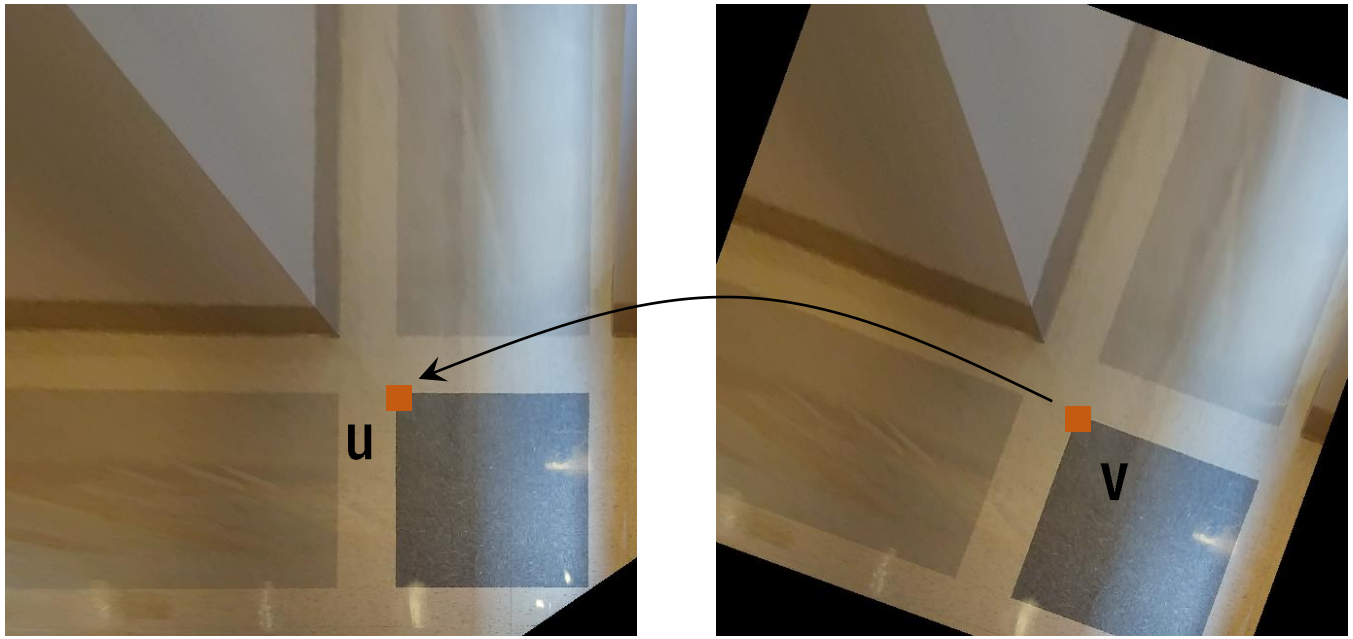


u_x



u_y

Euclidean Transform SE(3)

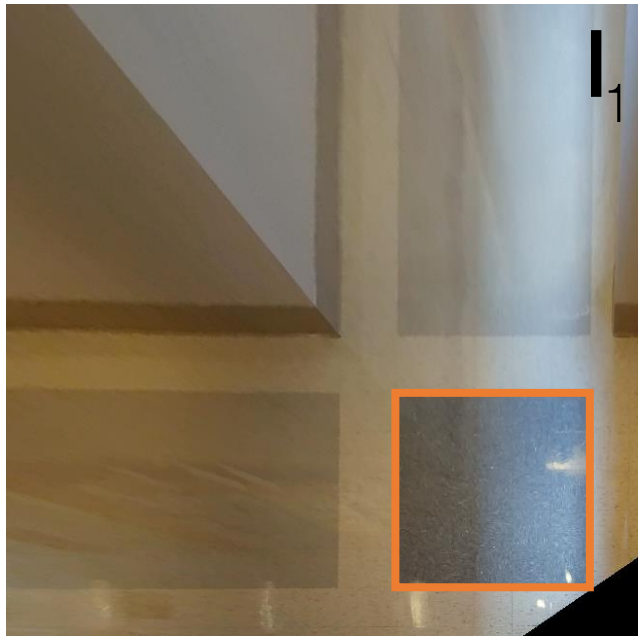


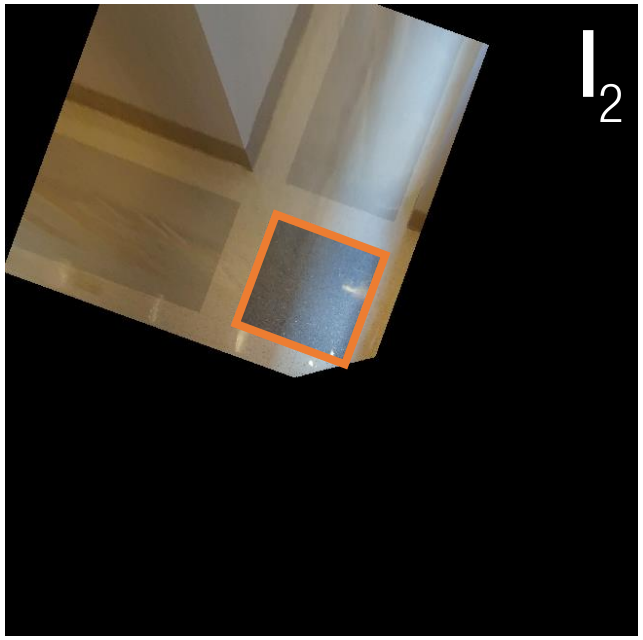
$$\mathbf{H}^{-1}\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

ImageWarpingEuclidean.m

```
function im_warped = ImageWarpingEuclidean(im, H)

im = double(im);

H = inv(H);

[v_x, v_y] = meshgrid(1:(size(im,2)), 1:(size(im,1)));
h = size(v_x, 1); w = size(v_x,2);

u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);

im_warped(:,:,1) = reshape(interp2(im(:,:,1), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,2) = reshape(interp2(im(:,:,2), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,3) = reshape(interp2(im(:,:,3), u_x(:), u_y(:)), [h, w]);

im_warped = uint8(im_warped);
```

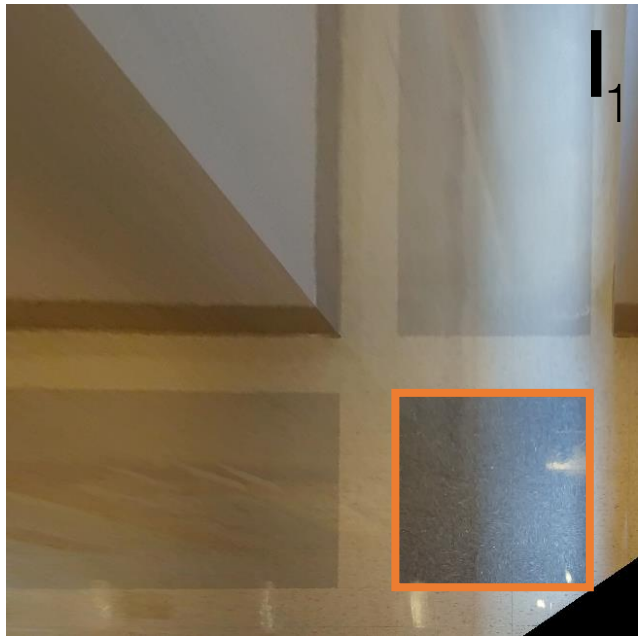
Similarity Transform

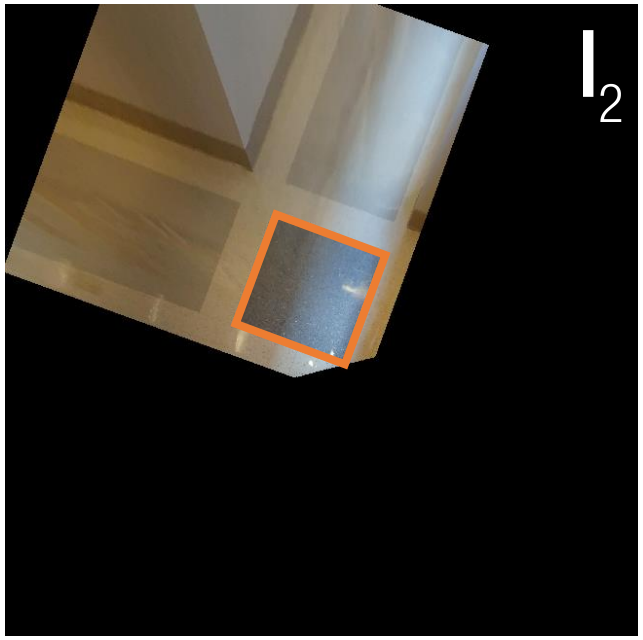




$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \quad ? \quad \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

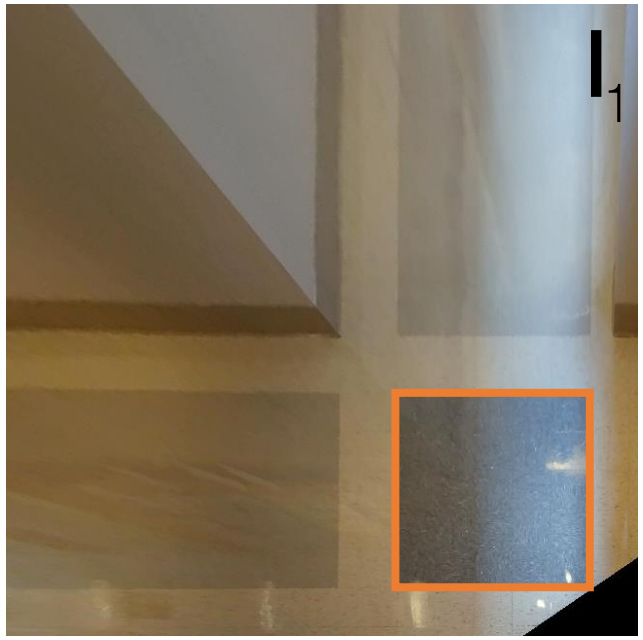
Similarity Transform

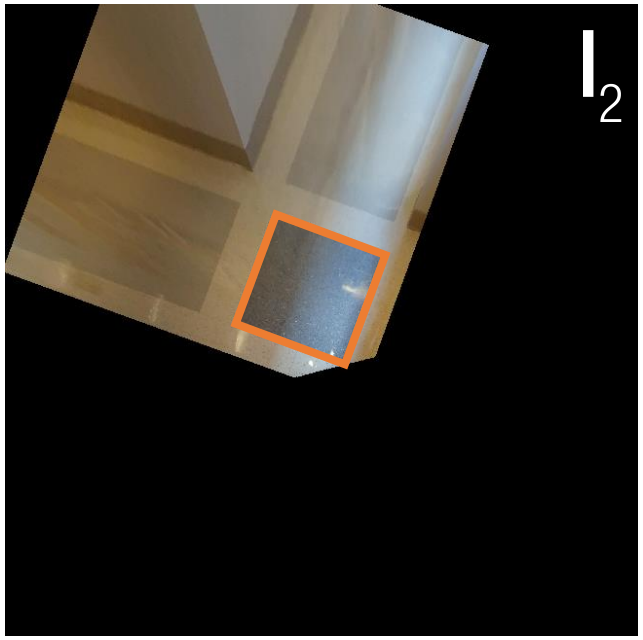




$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & & \\ & \alpha & \\ & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Similarity Transform





Invariant properties

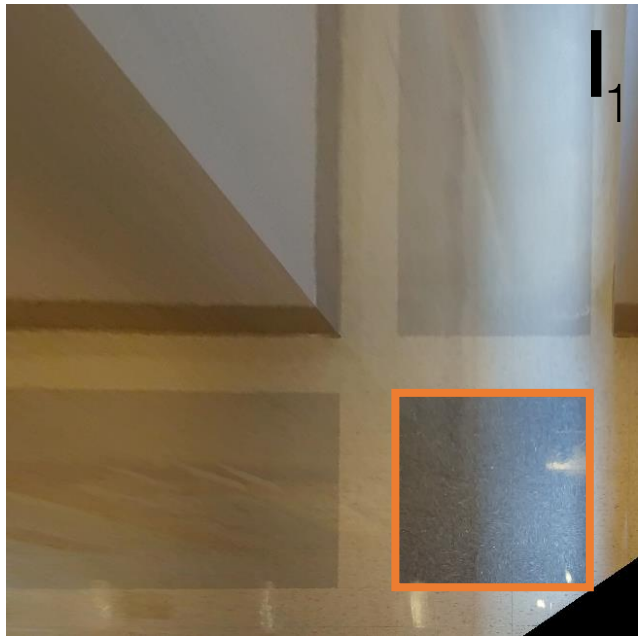
- Length ratio
- Angle

Degree of freedom

4 (2 translation+1 rotation+1 scale)

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & & \\ & \alpha & \\ & & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

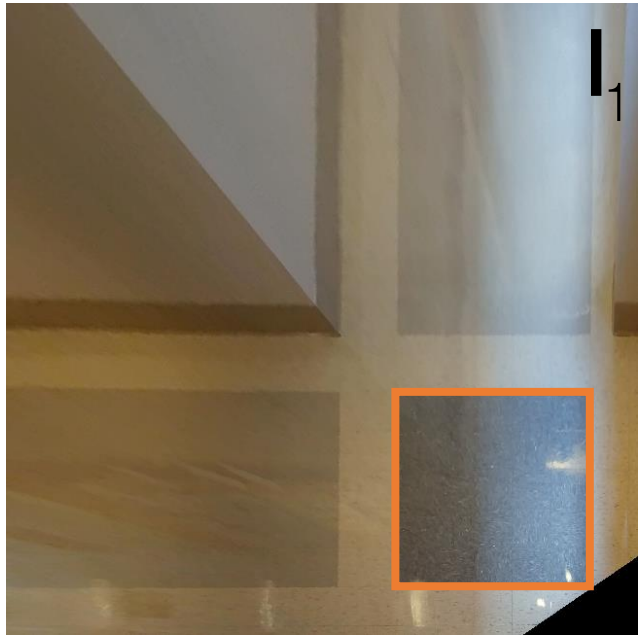
Affine Transform



$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean transform

Affine Transform

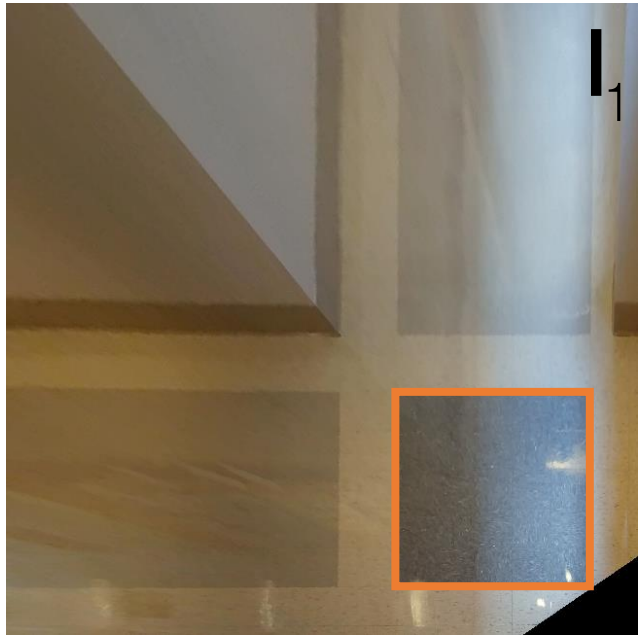


$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean transform

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Affine Transform



Invariant properties

- Parallelism
- Ratio of area
- Ratio of length

Degree of freedom

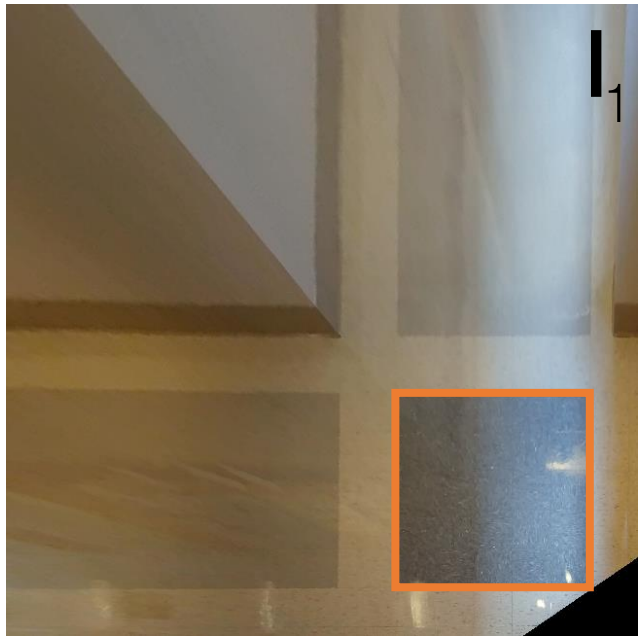
6

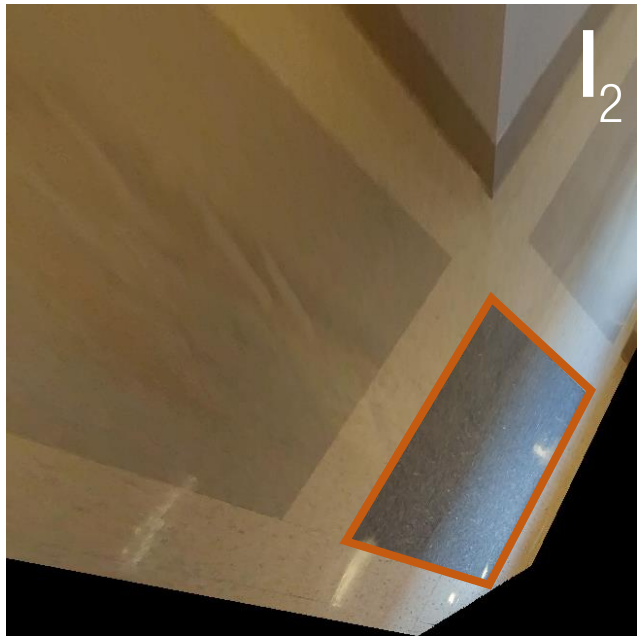
$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Euclidean transform

$$\begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

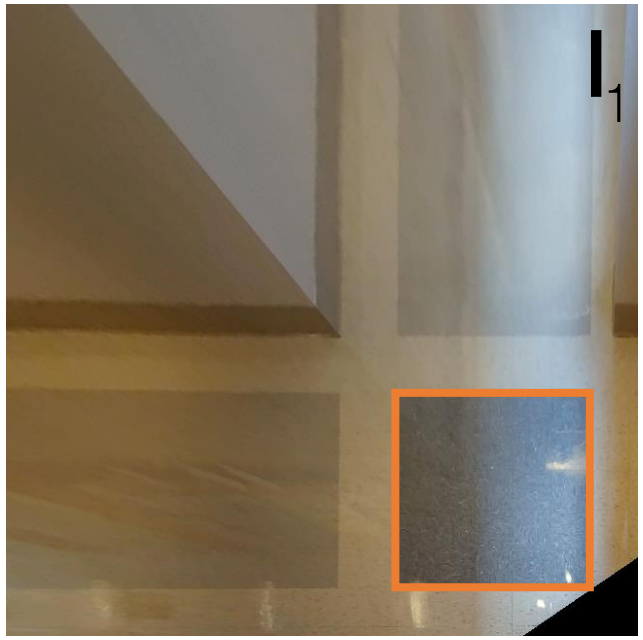
Perspective Transform (Homography)

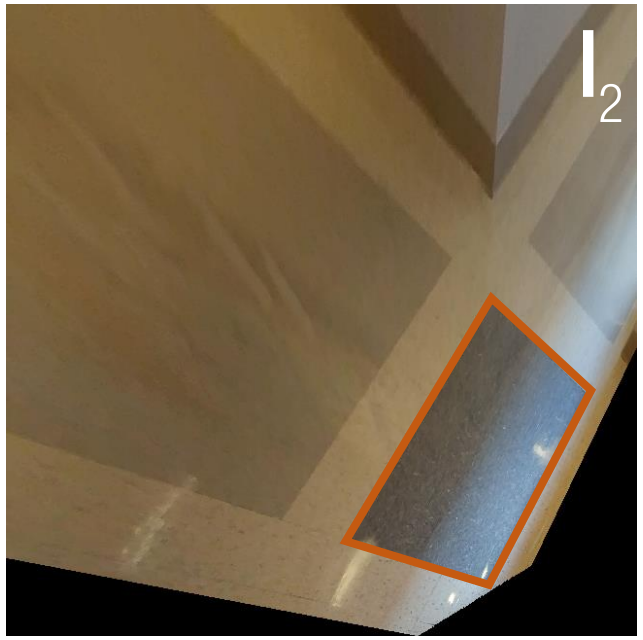




$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Perspective Transform (Homography)

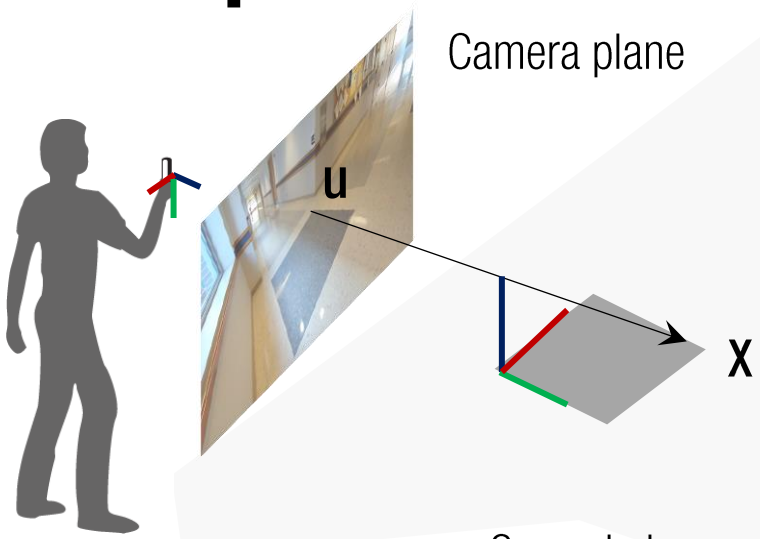




$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

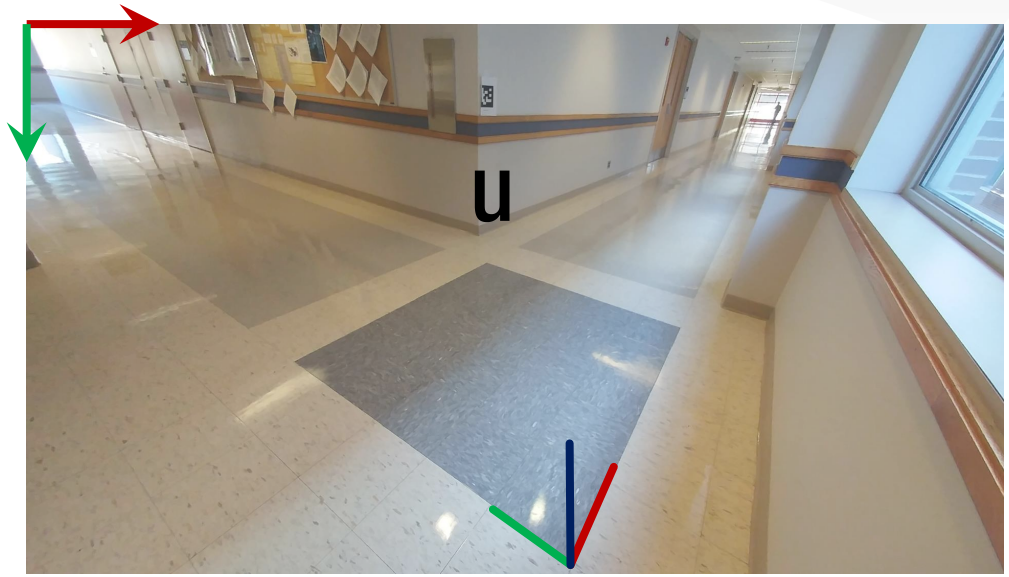
: General form of plane to plane linear mapping

Perspective Transform (Homography)

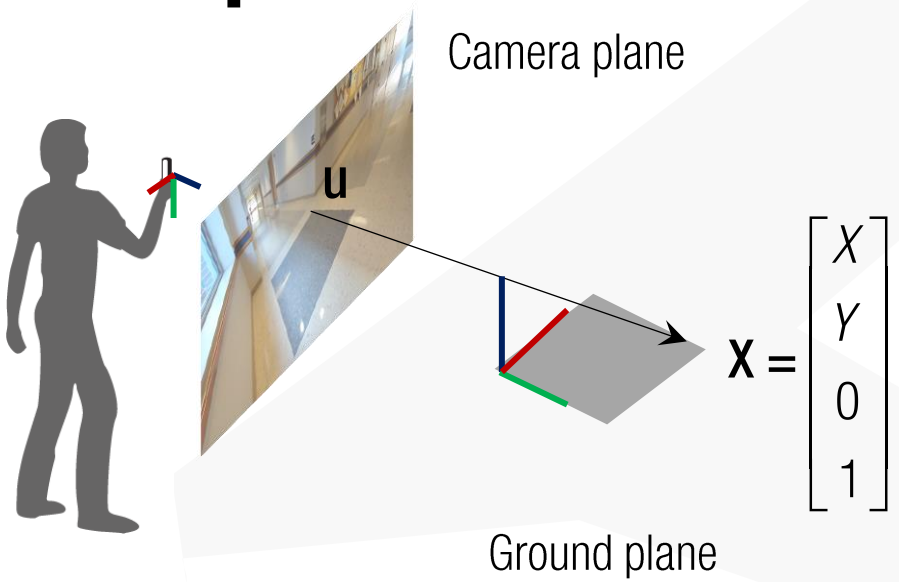


$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

Camera plane Ground plane

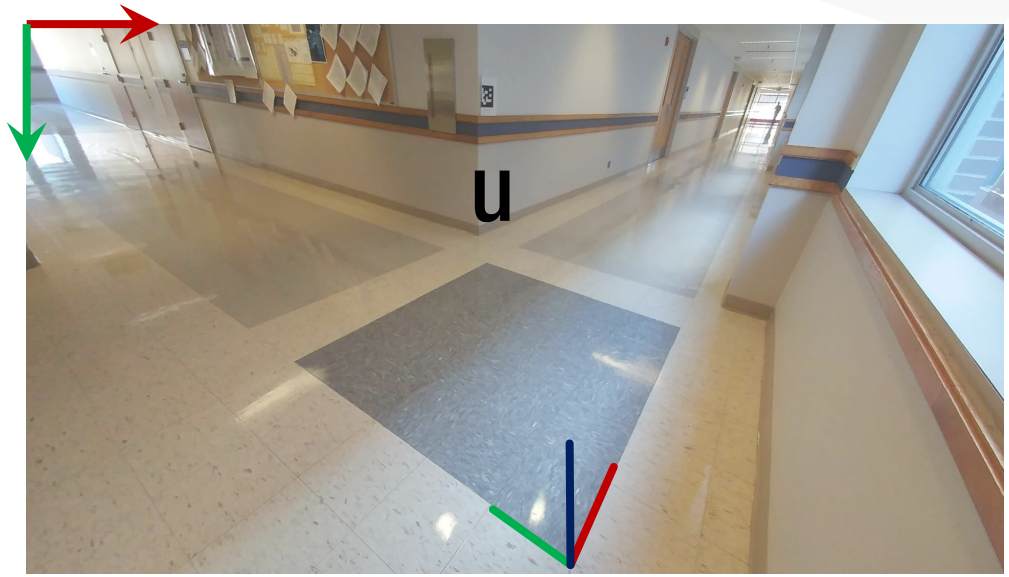


Perspective Transform (Homography)

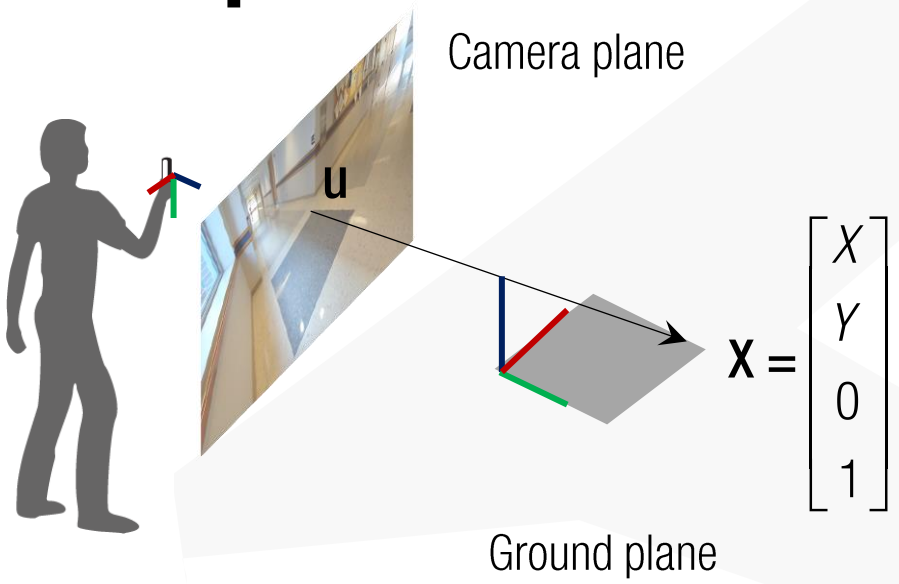


$$\lambda \mathbf{u} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

Camera plane Ground plane



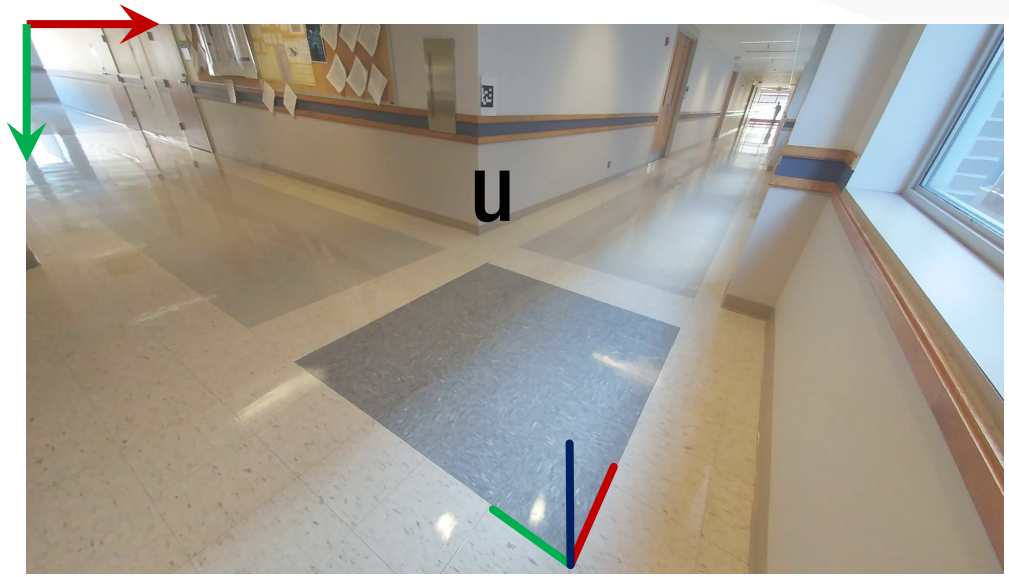
Perspective Transform (Homography)



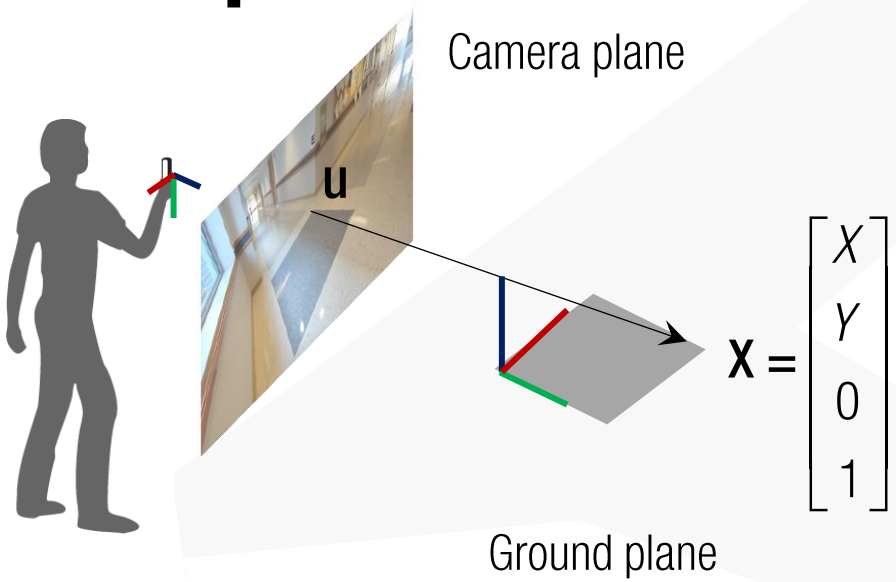
$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

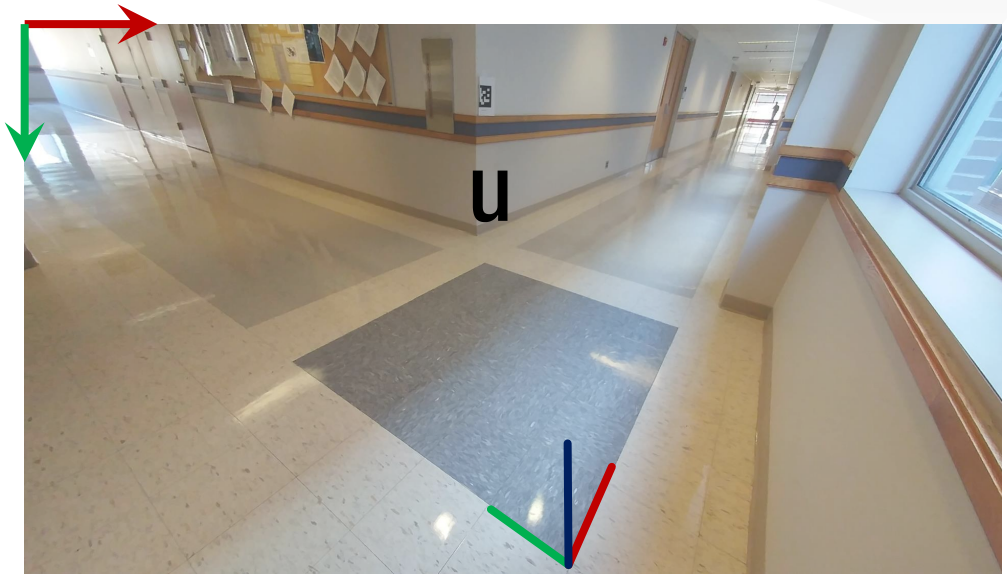
Camera plane Ground plane

$$\longrightarrow \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix}$$



Perspective Transform (Homography)





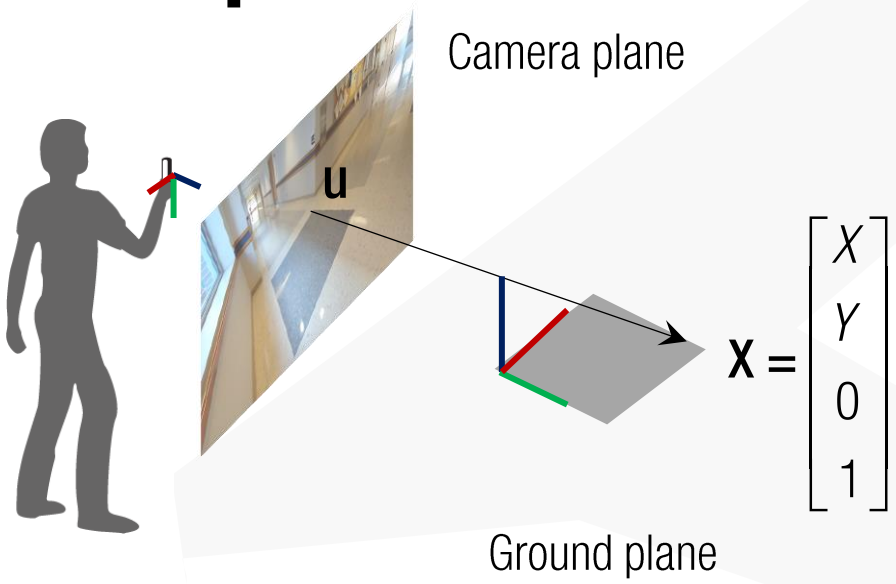
$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

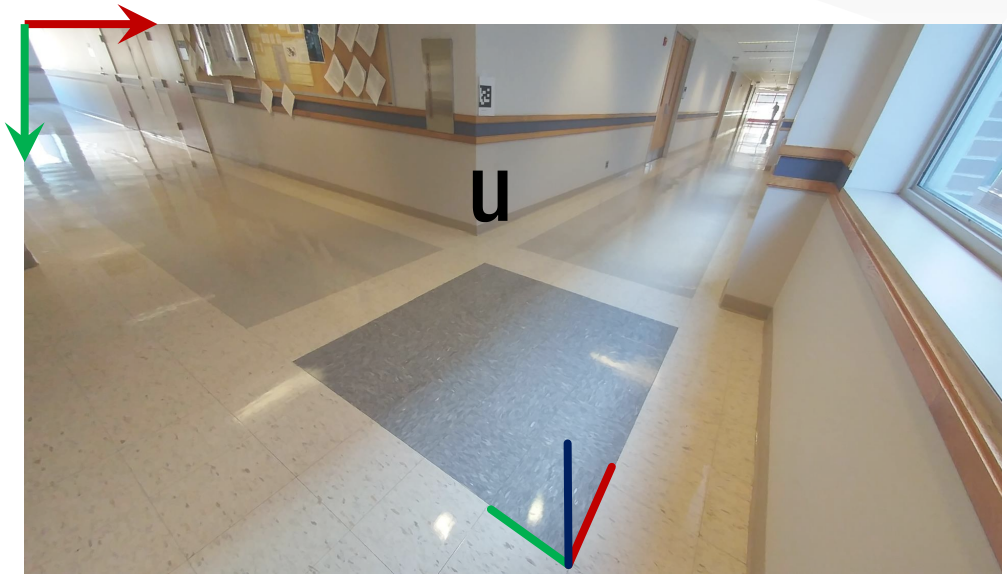
Camera plane — Ground plane

$$\longrightarrow \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix}$$

$$\longrightarrow \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

Perspective Transform (Homography)





$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

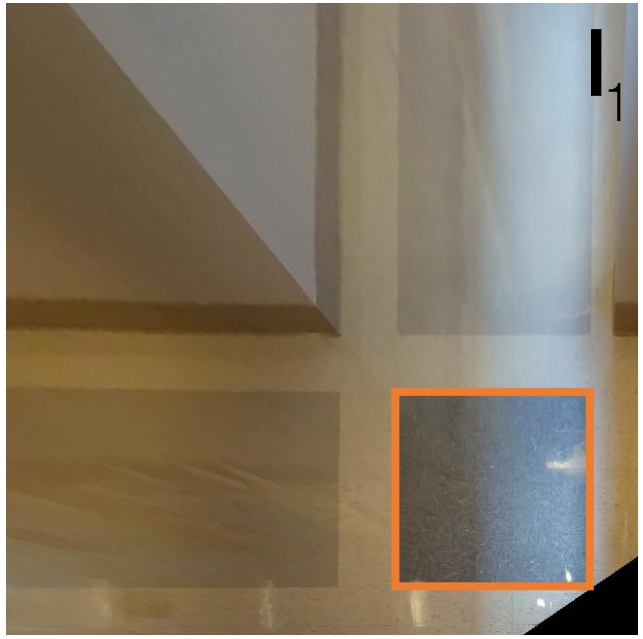
Camera plane — Ground plane

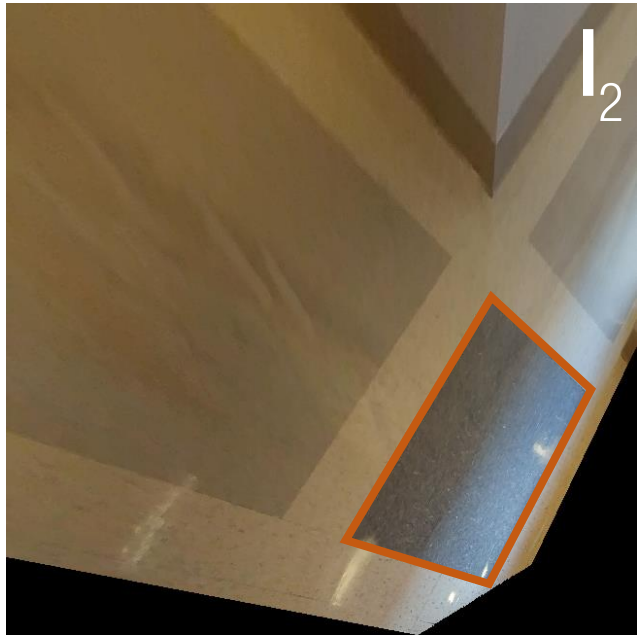
$$\longrightarrow \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\0\\1\end{bmatrix}$$

$$\longrightarrow \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

$$\longrightarrow \lambda \begin{bmatrix}u_x\\u_y\\1\end{bmatrix} = \mathbf{H}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

Perspective Transform (Homography)

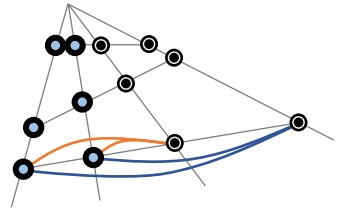




$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

Invariant properties

- Cross ratio



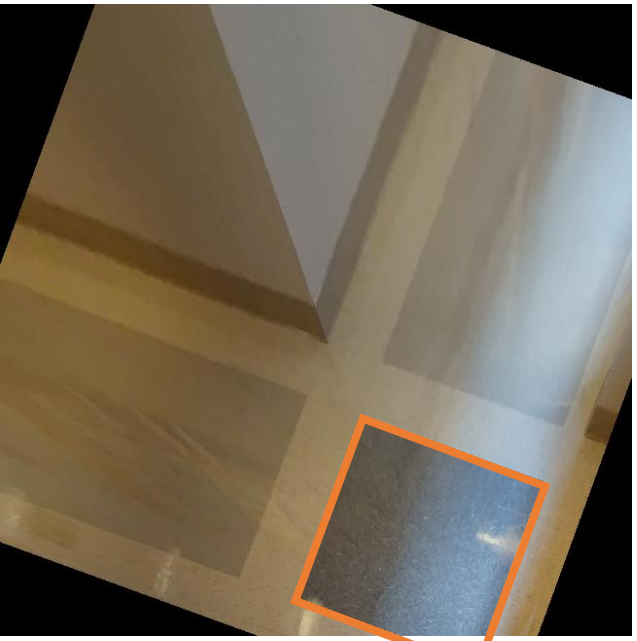
- Concurrency

- Colinearity

Degree of freedom

8 (9 variables – 1 scale)

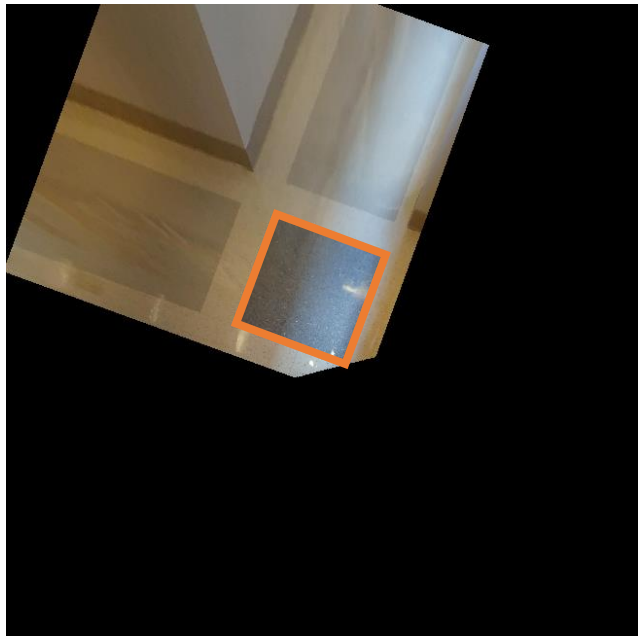
Hierarchy of Transformations



Euclidean (3 dof)

- Length
- Angle
- Area

$$\begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ & & 1 \end{bmatrix}$$



Similarity (4 dof)

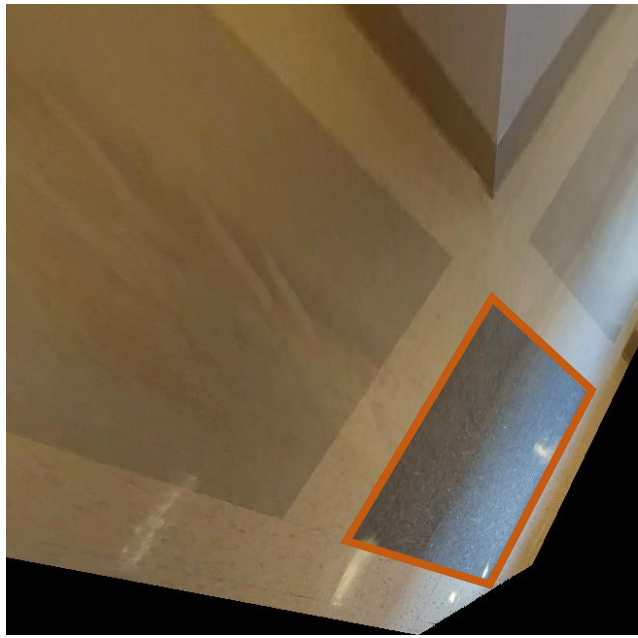
- Length ratio
- Angle

$$\begin{bmatrix} \alpha\cos\theta & -\alpha\sin\theta & t_x \\ \alpha\sin\theta & \alpha\cos\theta & t_y \\ & & 1 \end{bmatrix}$$

Affine (6 dof)

- Parallelism
- Ratio of area
- Ratio of length

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix}$$

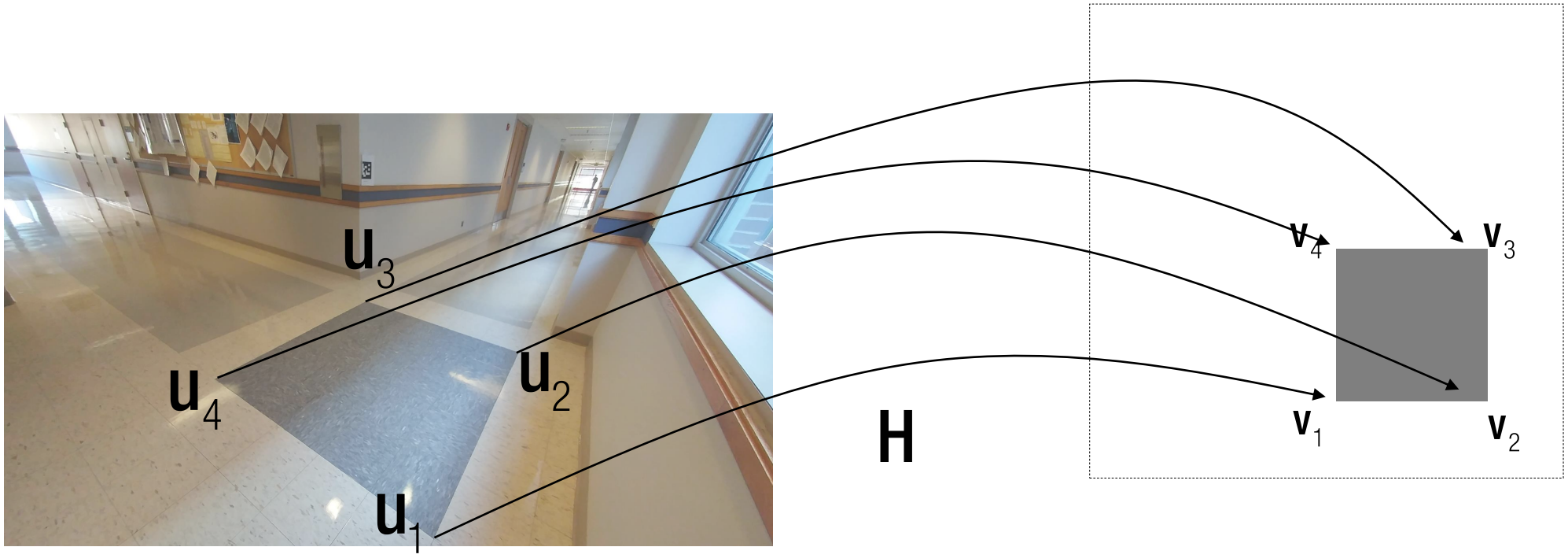


Projective (8 dof)

- Cross ratio
- Concurrency
- Colinearity

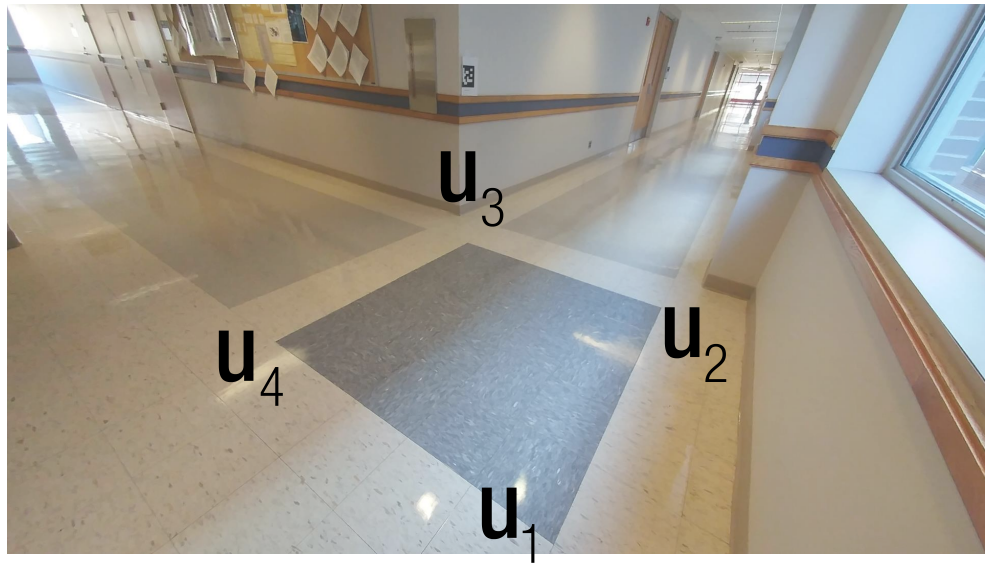
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$$

Fun with Homography



The image can be rectified as if it is seen from top view.

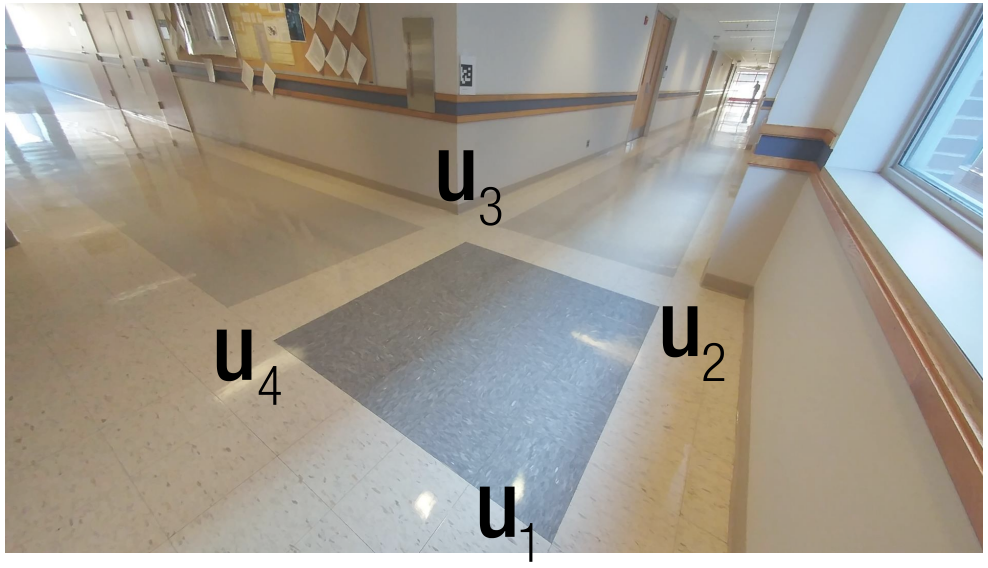
Fun with Homography



RectificationViaHomography.m

```
u = [u1'; u2'; u3'; u4'];
v = [v1'; v2'; v3'; v4'];

% Need at least non-colinear four points
H = ComputeHomography(v, u);

im_warped = ImageWarping(im, H);
```

Fun with Homography



RectificationViaHomography.m

```
u = [u1'; u2'; u3'; u4'];
v = [v1'; v2'; v3'; v4'];

% Need at least non-colinear four points
H = ComputeHomography(v, u);

im_warped = ImageWarping(im, H);
```

ImageWarping.m

```
u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);
u_z = H(3,1)*v_x + H(3,2)*v_y + H(3,3);

u_x = u_x./u_z;
u_y = u_y./u_z;
```

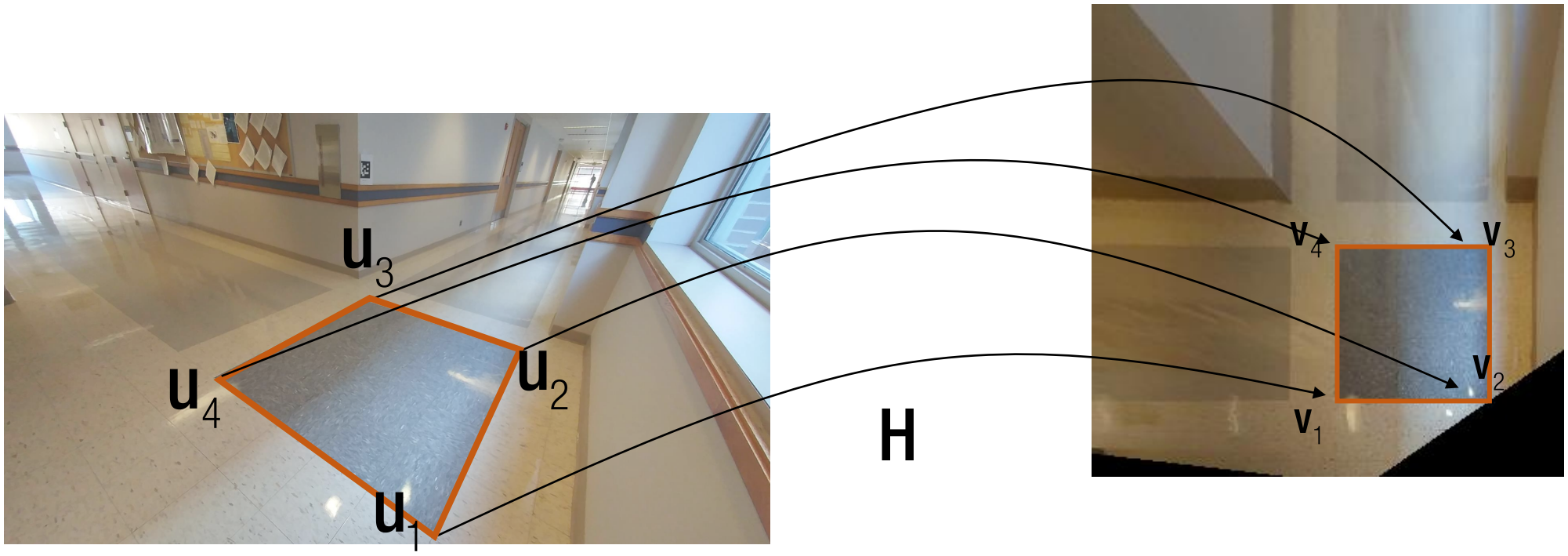
$$\lambda \begin{bmatrix} v_x \\ v_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} u_x \\ u_y \\ 1 \end{bmatrix}$$

```
im_warped(:,:,1) = reshape(interp2(im(:,:,1), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,2) = reshape(interp2(im(:,:,2), u_x(:), u_y(:)), [h, w]);
im_warped(:,:,3) = reshape(interp2(im(:,:,3), u_x(:), u_y(:)), [h, w]);

im_warped = uint8(im_warped);
```

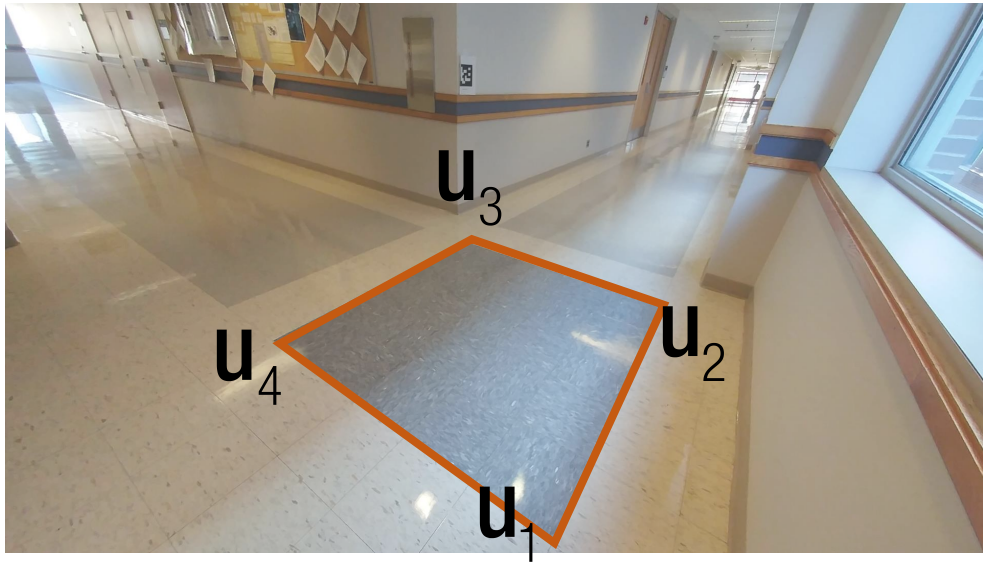
Cf) ImageWarpingEuclidean.m

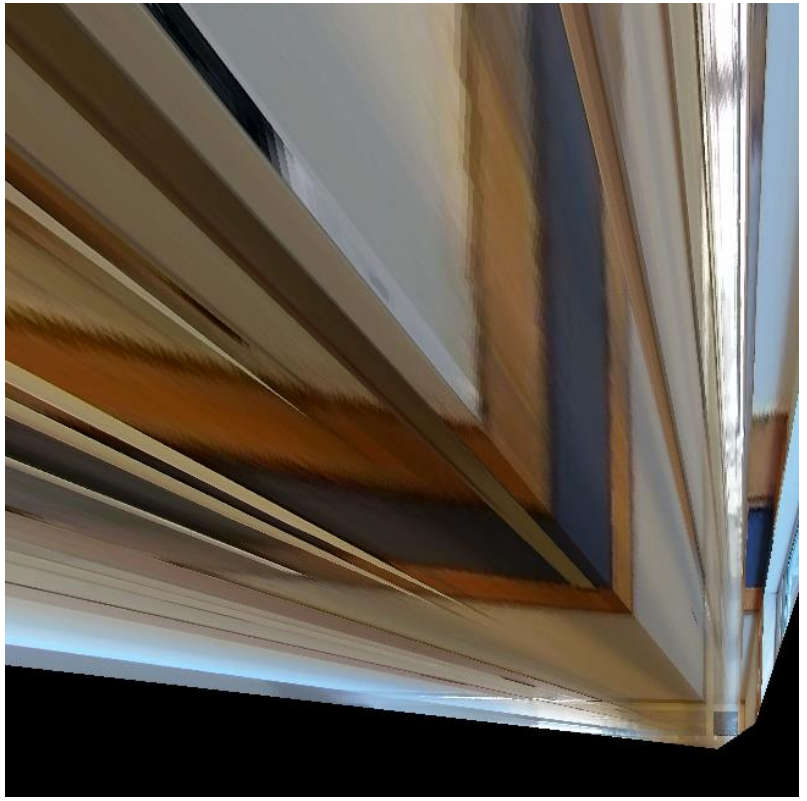
```
u_x = H(1,1)*v_x + H(1,2)*v_y + H(1,3);
u_y = H(2,1)*v_x + H(2,2)*v_y + H(2,3);
```

Fun with Homography



Fun with Homography





Fun with Homography

Fun with Homography

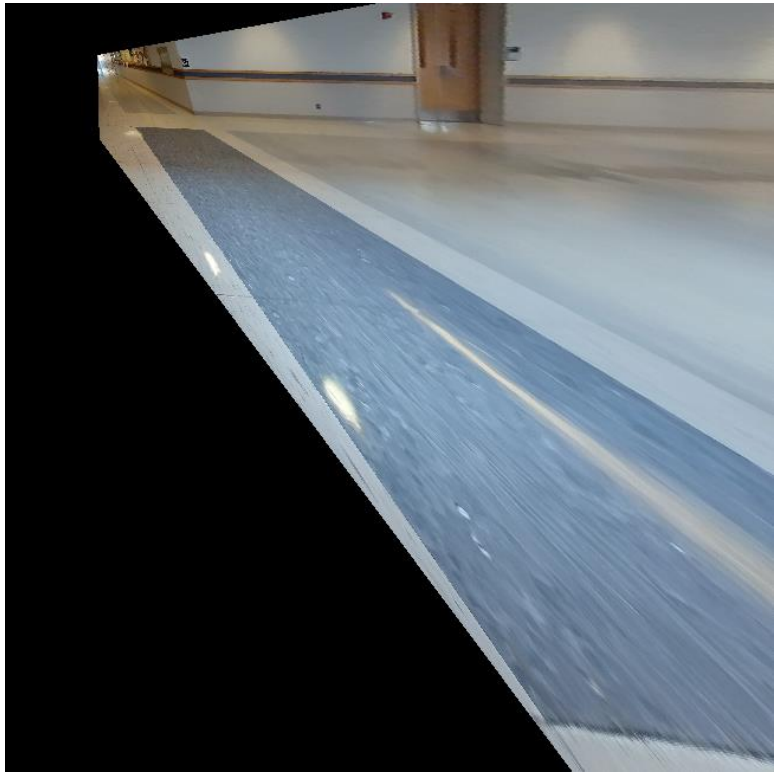


Image Inpainting

Image Inpainting

Virtual Advertisement

Image Transform via Plane

Keller entrance left

Keller entrance right

Image Transform via 3D Plane

$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

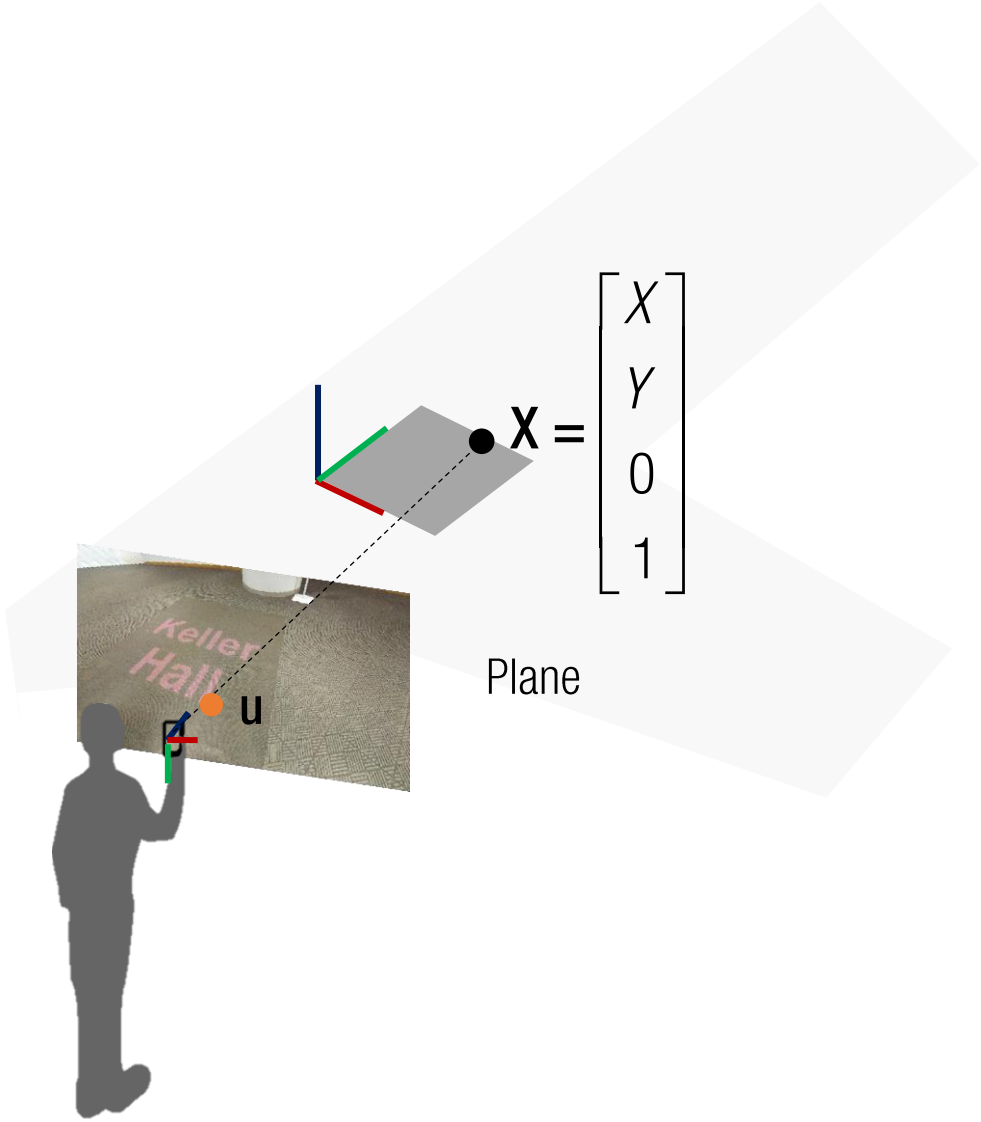


Image Transform via 3D Plane

$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\longrightarrow \quad \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

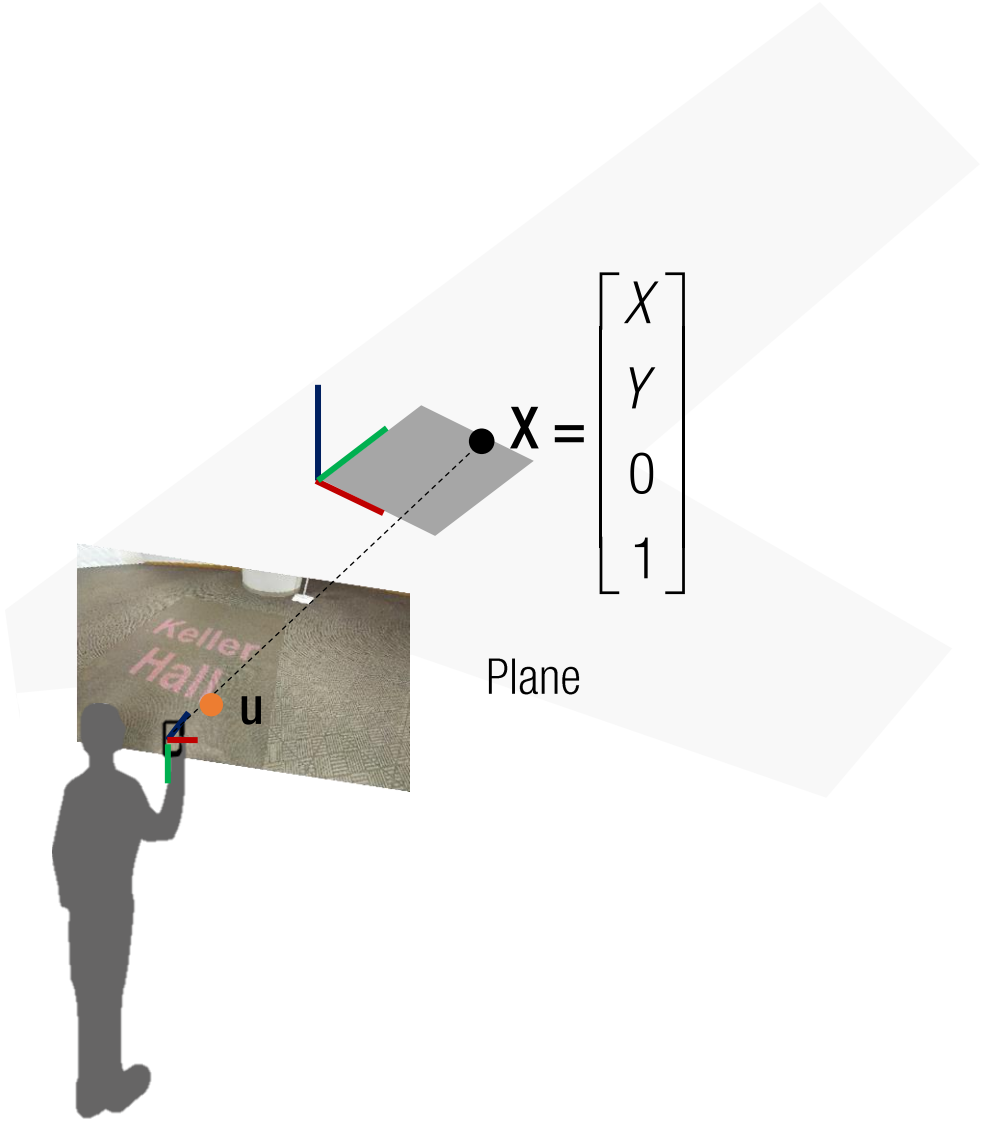
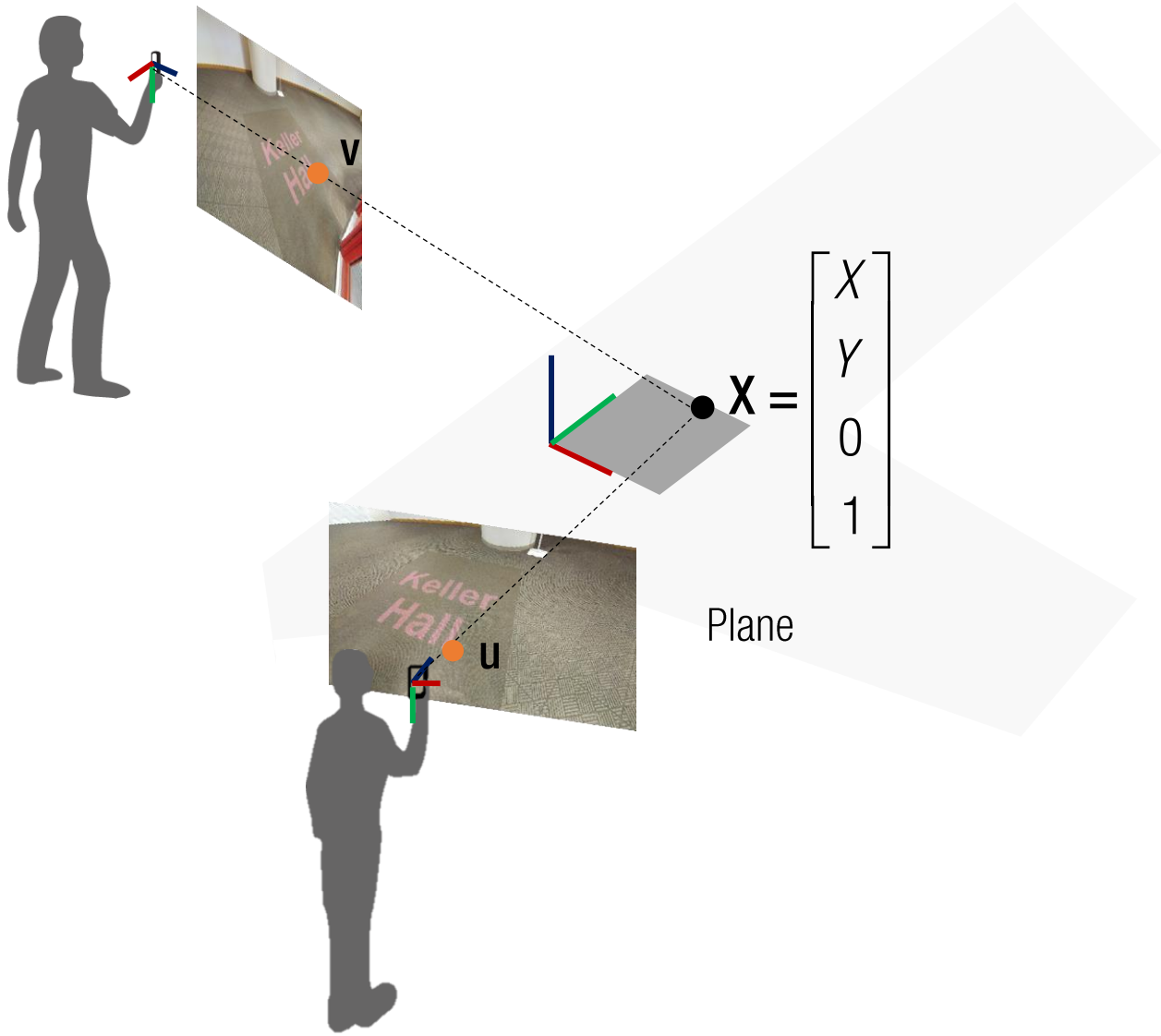


Image Transform via 3D Plane

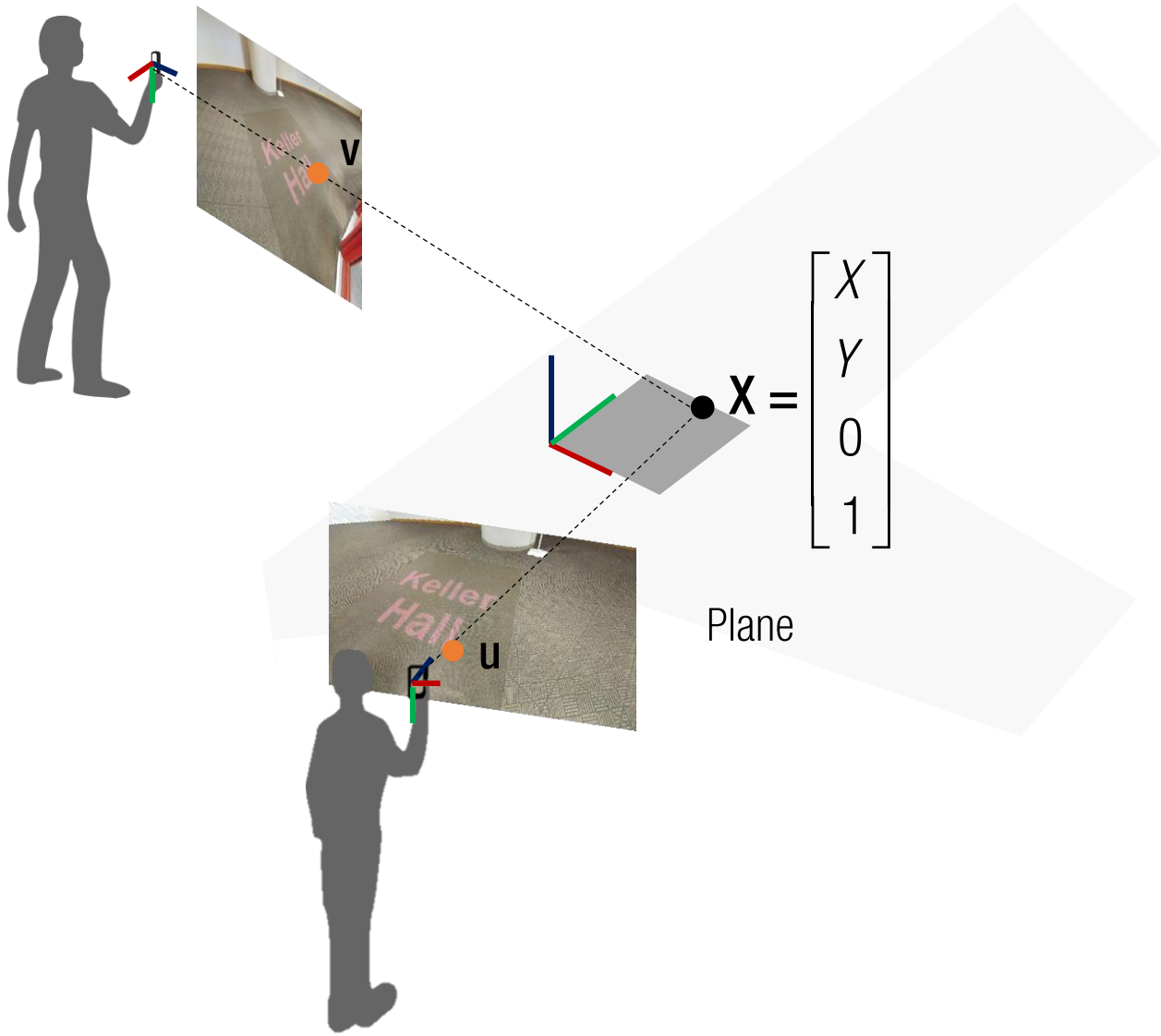


$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\longrightarrow \quad \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

$$\mu \mathbf{v} = \mathbf{K}'\begin{bmatrix}\mathbf{r}_1' & \mathbf{r}_2' & \mathbf{t}'\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

Image Transform via 3D Plane



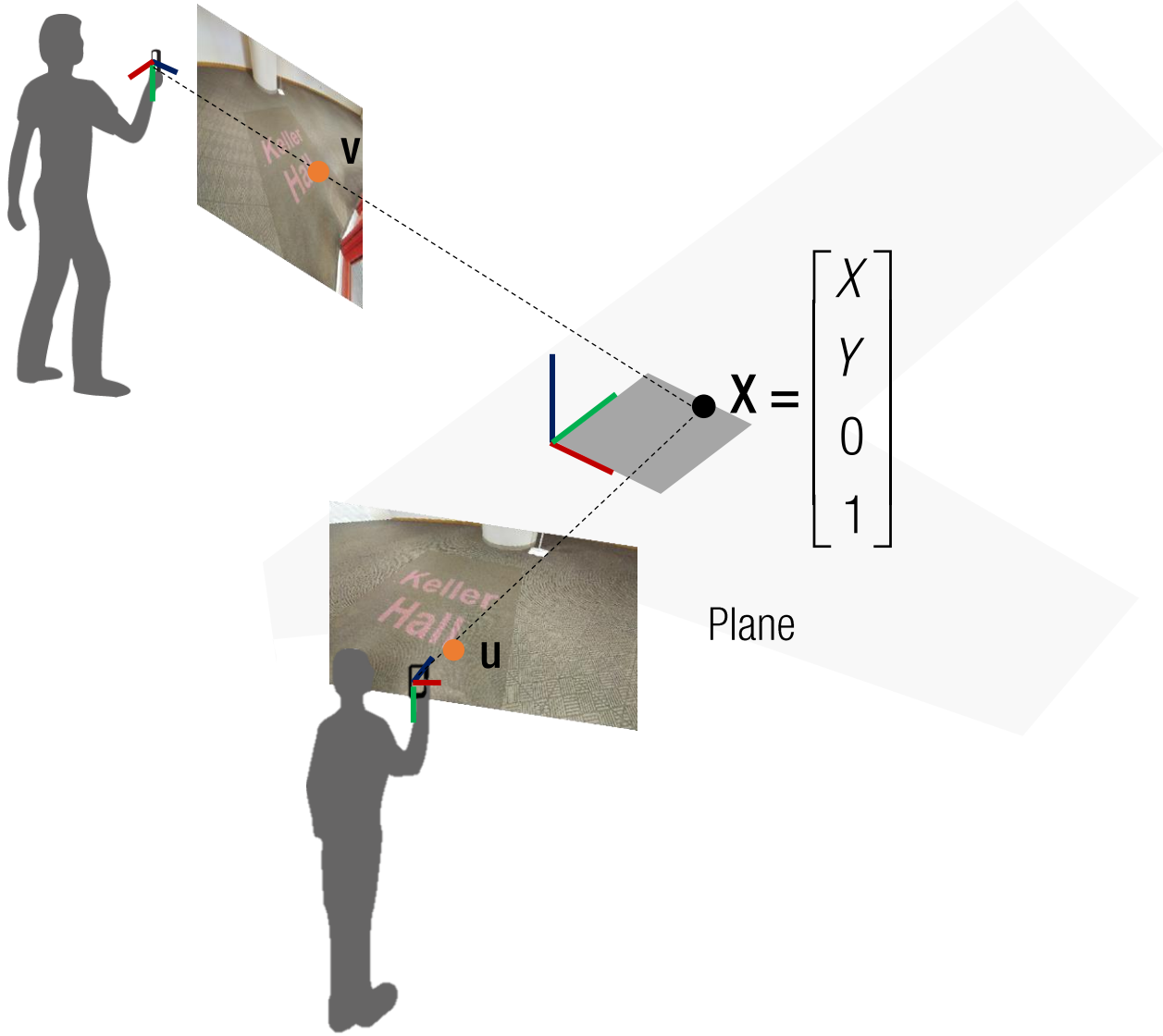
$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\longrightarrow \quad \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

$$\mu \mathbf{v} = \mathbf{K}'\begin{bmatrix}\mathbf{r}_1' & \mathbf{r}_2' & \mathbf{t}'\end{bmatrix}\begin{bmatrix}X\\Y\\1\end{bmatrix}$$

How are two image coordinates (**u**,**v**) related?

Image Transform via 3D Plane



$$\lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

$$\longrightarrow \quad \lambda \mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

$$\mu \mathbf{v} = \mathbf{K}'\begin{bmatrix}\mathbf{r}_1' & \mathbf{r}_2' & \mathbf{t}'\end{bmatrix}\begin{bmatrix}X \\ Y \\ 1\end{bmatrix}$$

How are two image coordinates (**u**,**v**) related?

$$\longrightarrow \quad \lambda\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}^{-1}\mathbf{K}^{-1}\mathbf{u} = \begin{bmatrix}X \\ Y \\ 1\end{bmatrix} = \mu\begin{bmatrix}\mathbf{r}_1' & \mathbf{r}_2' & \mathbf{t}'\end{bmatrix}^{-1}\mathbf{K}'^{-1}\mathbf{v}$$

$$\alpha \mathbf{v} = \mathbf{K}'\begin{bmatrix}\mathbf{r}_1' & \mathbf{r}_2' & \mathbf{t}'\end{bmatrix}\begin{bmatrix}\mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t}\end{bmatrix}^{-1}\mathbf{K}^{-1}\mathbf{u}$$

$$\alpha \mathbf{v} = \mathbf{H}\mathbf{u}$$

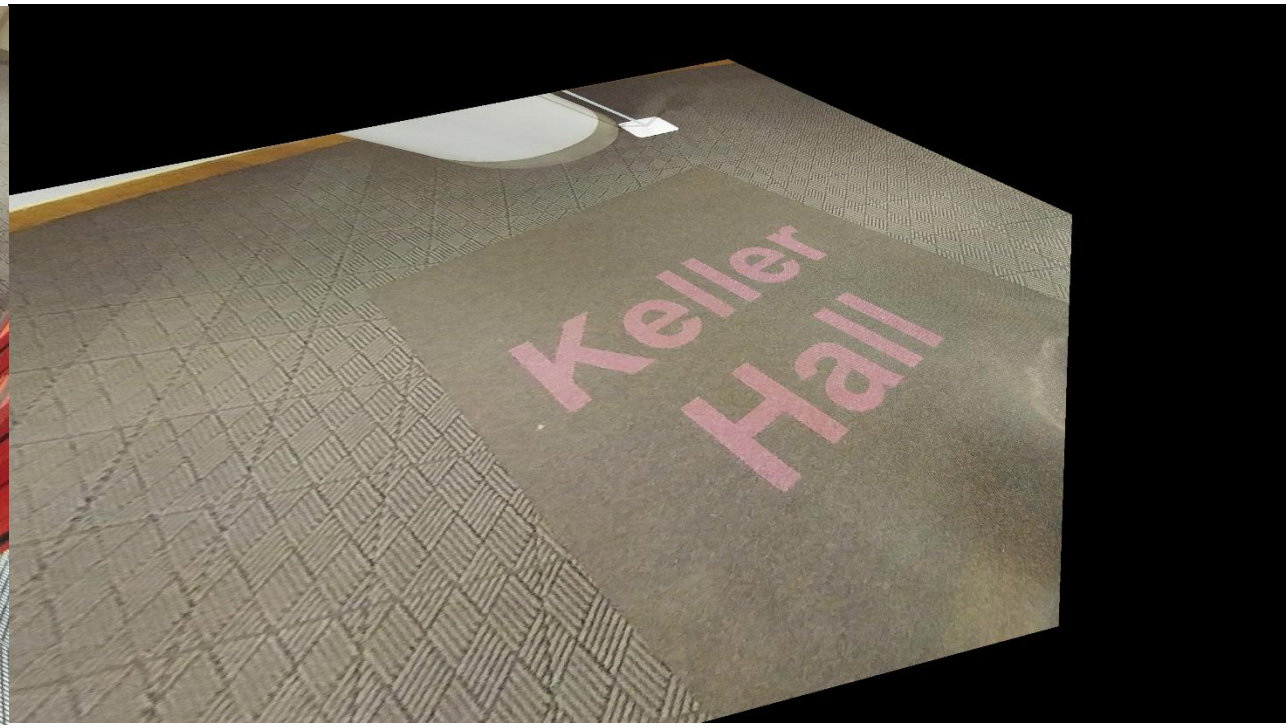
Image Transform via 3D Plane

Keller entrance left

Keller entrance right

Image Transform via 3D Plane

Keller entrance left



Right image to left

Image Transform via 3D Plane

Image Transform via 3D Plane

Left image to right

Right image to left

Image Transform via 3D Plane

360 Panorama

https://www.youtube.com/watch?v=H6SsB3JYqQg

Image Transform by Pure 3D Rotation

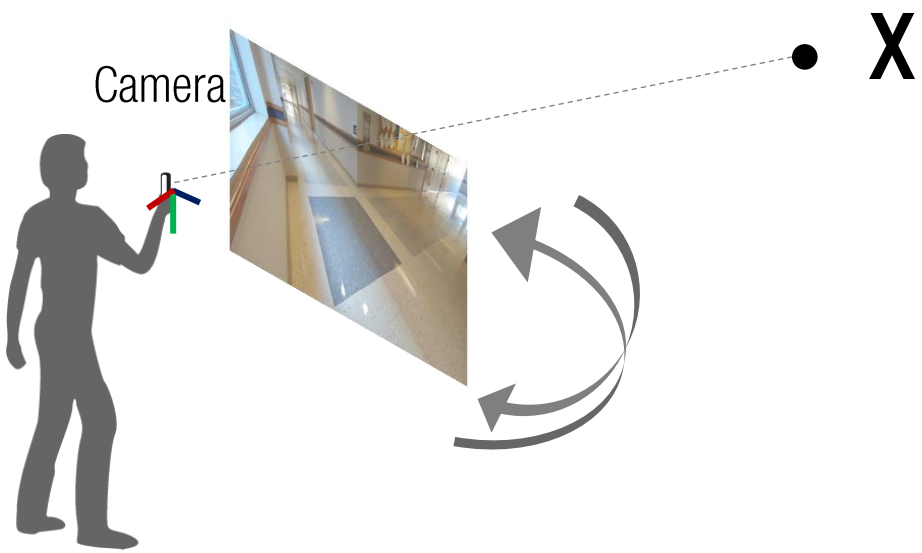
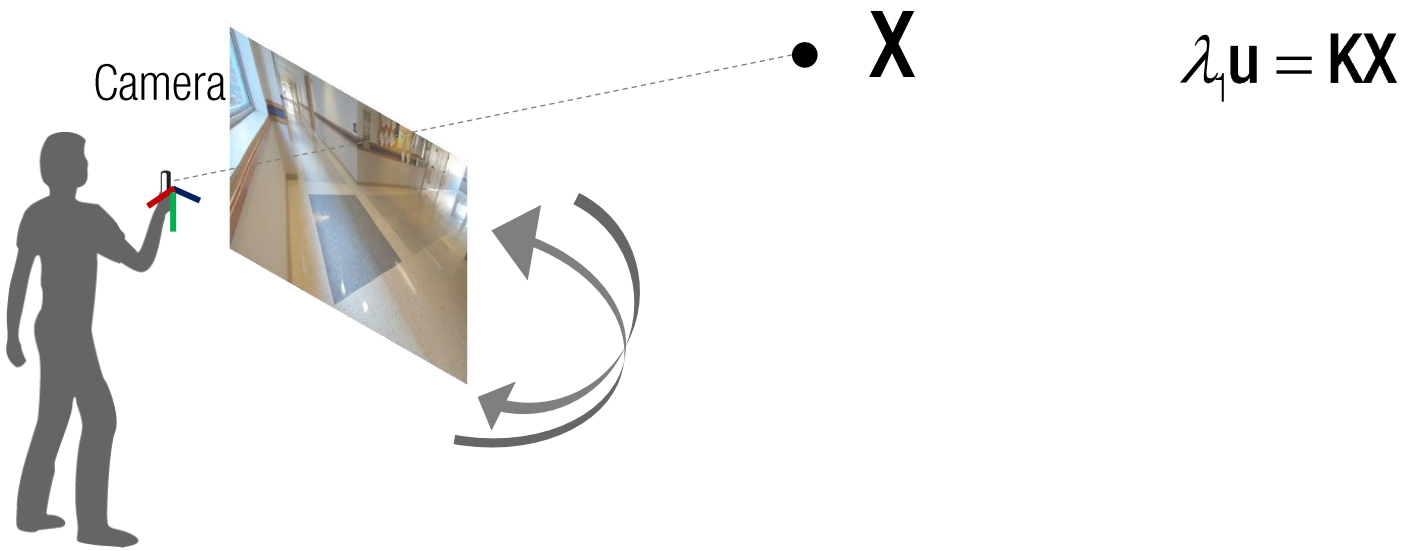
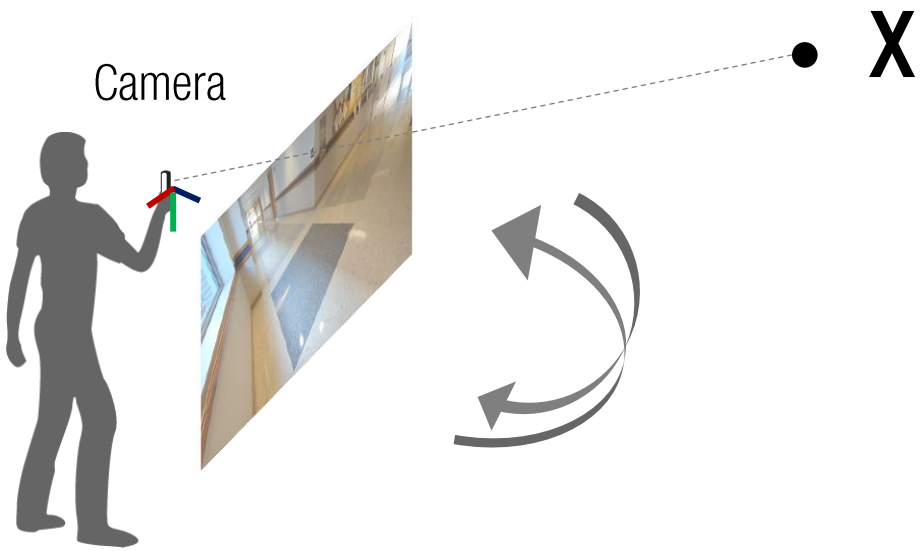


Image Transform by Pure 3D Rotation



$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

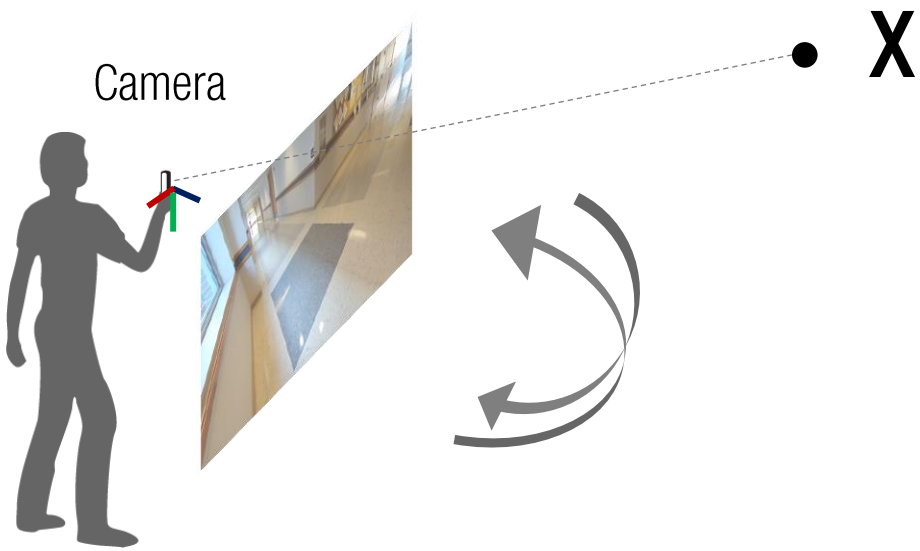
Image Transform by Pure 3D Rotation



$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

$$\lambda_2 \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{X}$$

Image Transform by Pure 3D Rotation

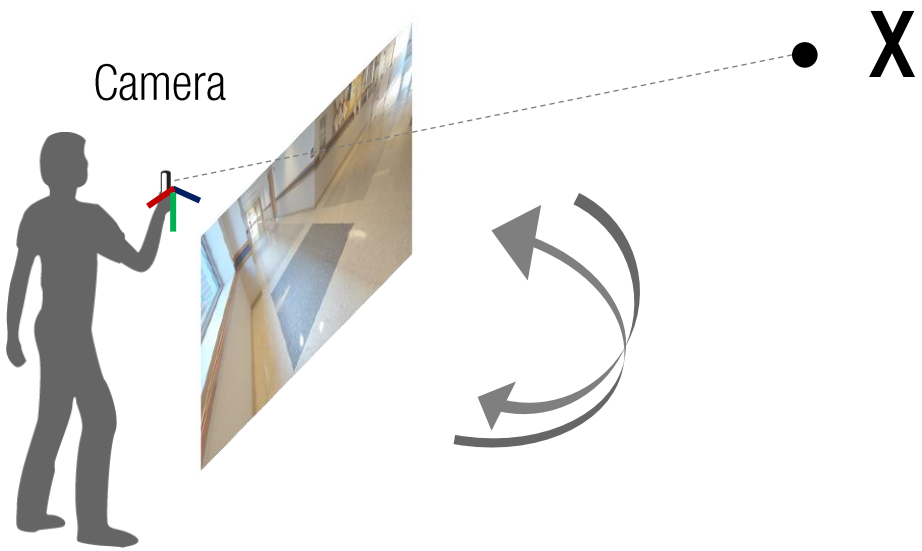


$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

$$\lambda_2 \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{X}$$

$$\longrightarrow \mathbf{X} = \lambda_1 \mathbf{K}^{-1}\mathbf{u} = \lambda_2 \mathbf{R}^{\mathsf{T}}\mathbf{K}^{-1}\mathbf{v}$$

Image Transform by Pure 3D Rotation



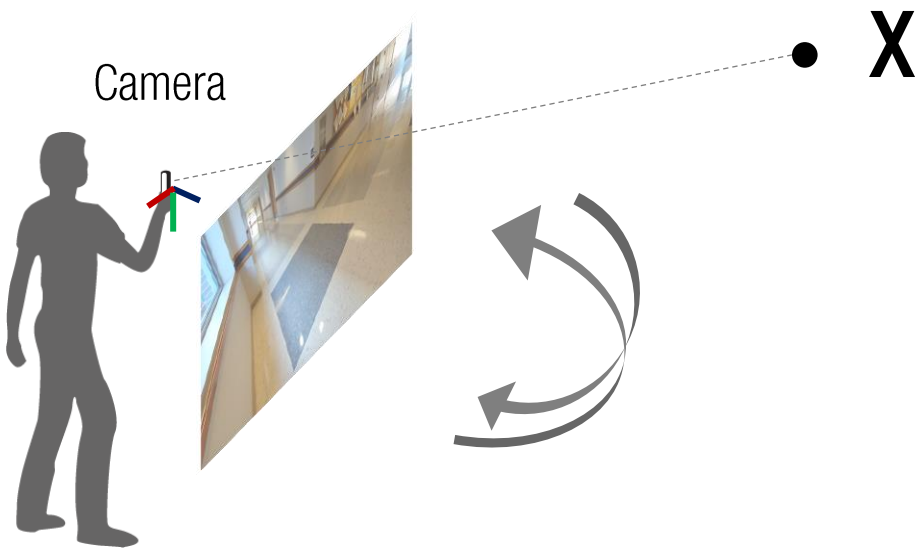
$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

$$\lambda_2 \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{X}$$

$$\longrightarrow \mathbf{X} = \lambda_1 \mathbf{K}^{-1}\mathbf{u} = \lambda_2 \mathbf{R}^{\mathsf{T}}\mathbf{K}^{-1}\mathbf{v}$$

$$\longrightarrow \lambda \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{u}$$

Image Transform by Pure 3D Rotation



$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

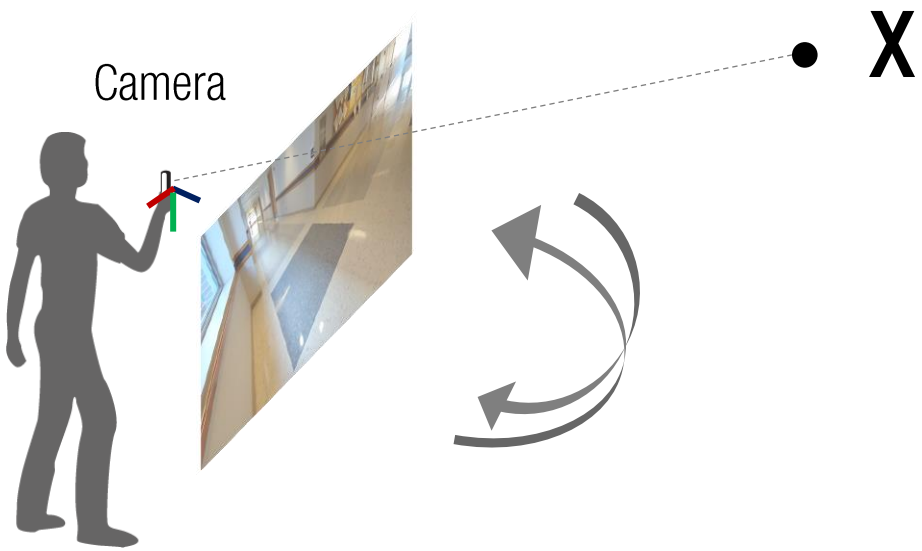
$$\lambda_2 \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{X}$$

$$\longrightarrow \mathbf{X} = \lambda_1 \mathbf{K}^{-1}\mathbf{u} = \lambda_2 \mathbf{R}^{\mathsf{T}}\mathbf{K}^{-1}\mathbf{v}$$

$$\longrightarrow \lambda \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{u}$$

$$\longrightarrow \mathbf{H} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}$$

Image Transform by Pure 3D Rotation



$$\lambda_1 \mathbf{u} = \mathbf{K}\mathbf{X}$$

$$\lambda_2 \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{X}$$

$$\longrightarrow \mathbf{X} = \lambda_1 \mathbf{K}^{-1}\mathbf{u} = \lambda_2 \mathbf{R}^{\mathsf{T}}\mathbf{K}^{-1}\mathbf{v}$$

$$\longrightarrow \lambda \mathbf{v} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}\mathbf{u}$$

$$\longrightarrow \mathbf{H} = \mathbf{K}\mathbf{R}\mathbf{K}^{-1}$$

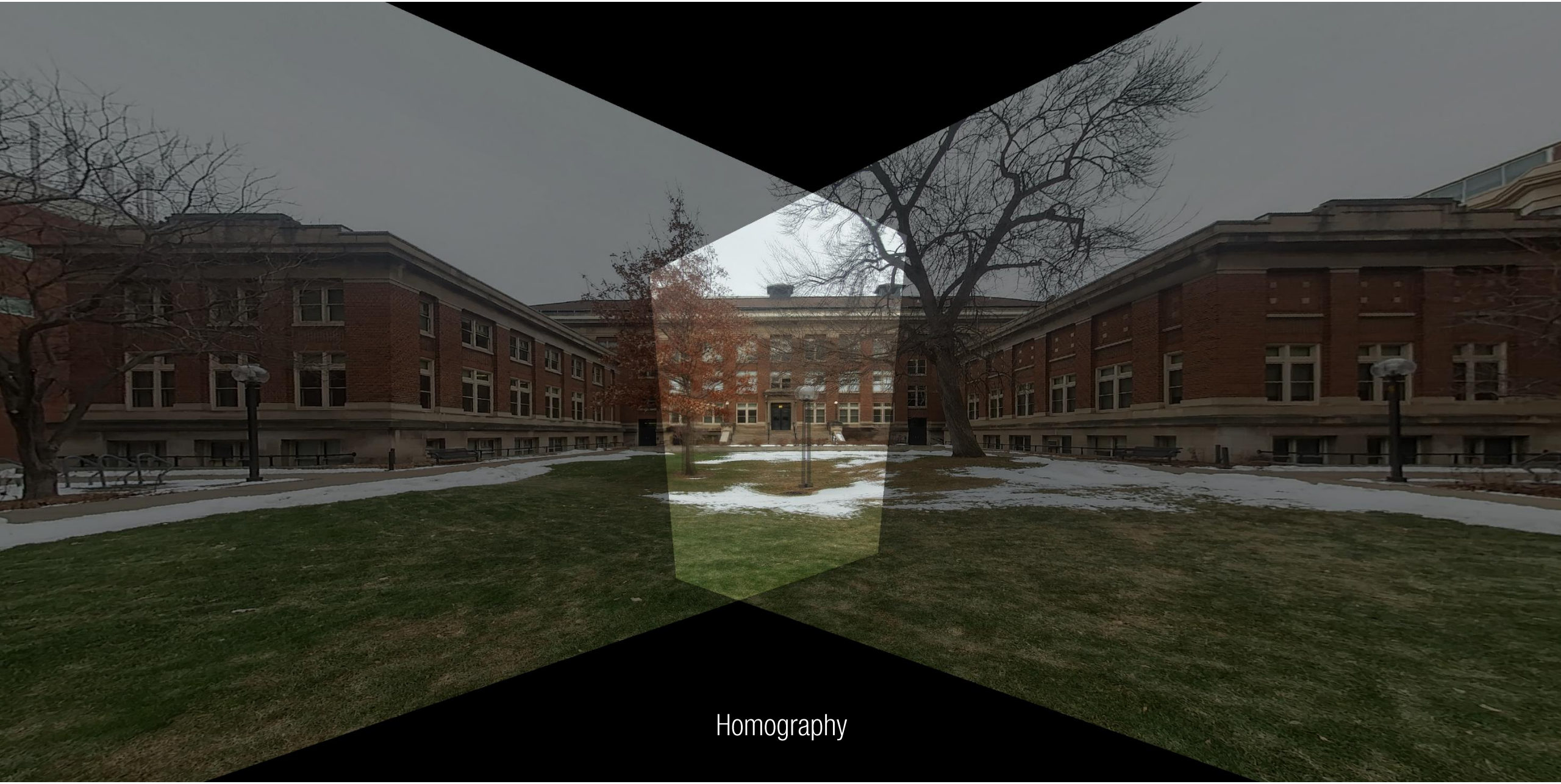
$$\longrightarrow \mathbf{R} = \mathbf{K}^{-1}\mathbf{H}\mathbf{K}$$

$$\lambda \mathbf{u} = \mathbf{H}\mathbf{v}$$

Lind Hall Left

Lind Hall Right

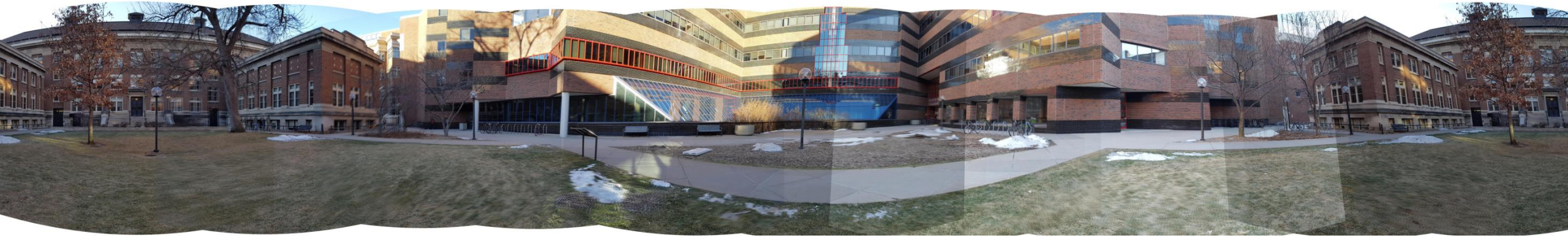
Euclidean Transform (Translation)


Homography

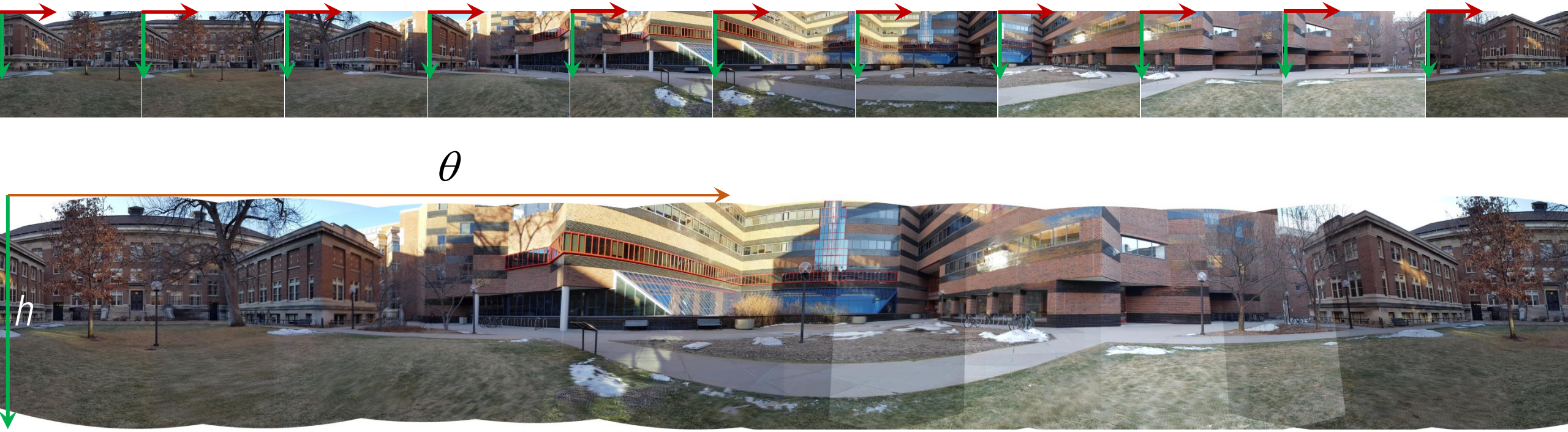
Homography

HW #2: Image Panorama (Cylindrical Projection)

HW #2: Image Panorama (Cylindrical Projection)

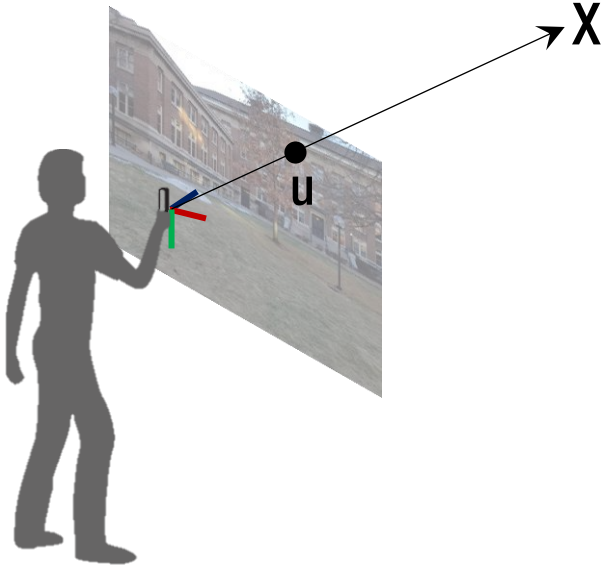


HW #2: Image Panorama (Cylindrical Projection)

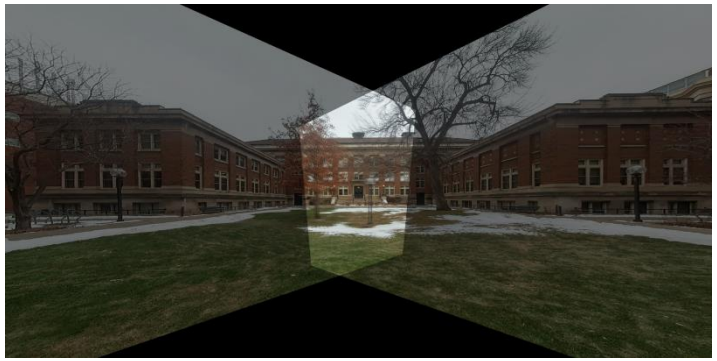


HW #2: Image Panorama (Cylindrical Projection)

R_1 R_2 R_3 R_4 R_5 R_6 R_7 R_8 R_9 R_{10} R_{11}

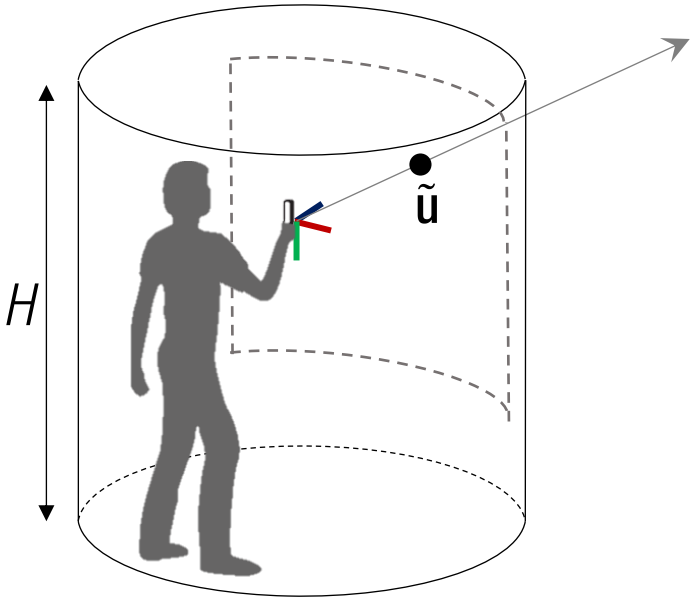


$$X = \lambda K^{-1} R^T u$$



$$R = K^{-1} H K$$

HW #2: Image Panorama (Cylindrical Projection)

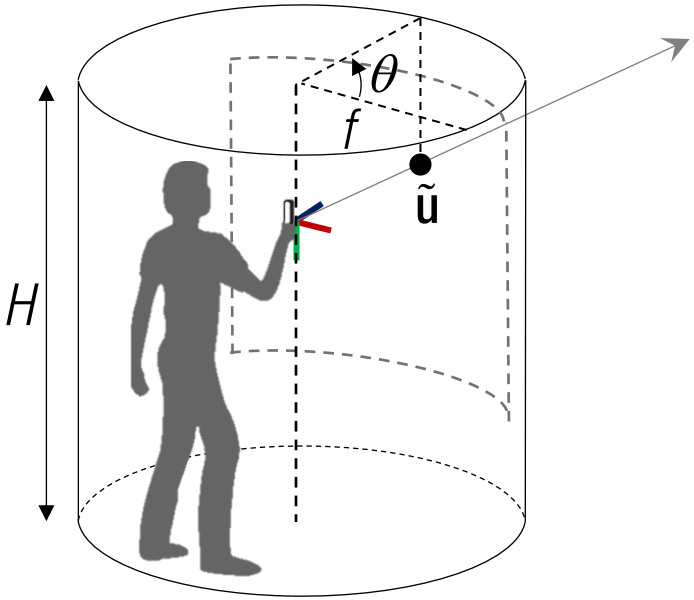


$$X = \lambda K^{-1} R^{T} u$$

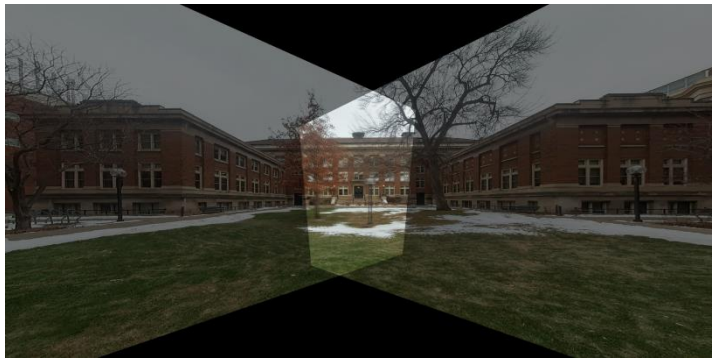
$$R = K^{-1} H K$$

$$\tilde{u} =$$

HW #2: Image Panorama (Cylindrical Projection)



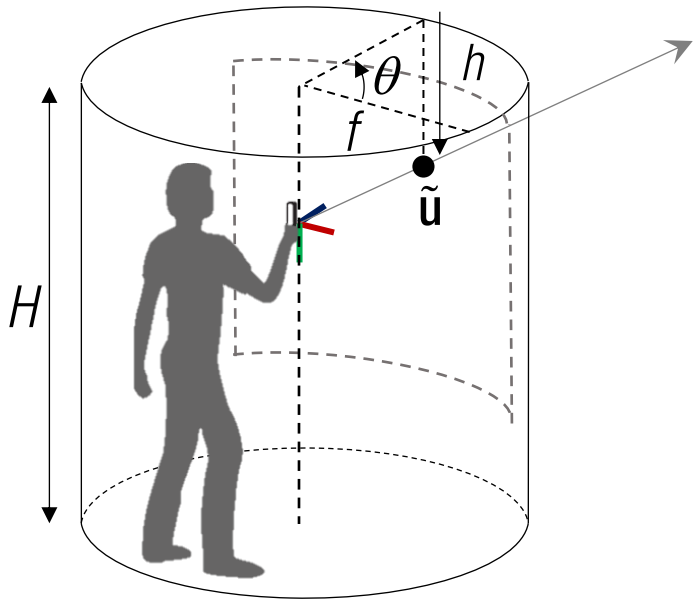
$$X = \lambda K^{-1} R^{T} u$$



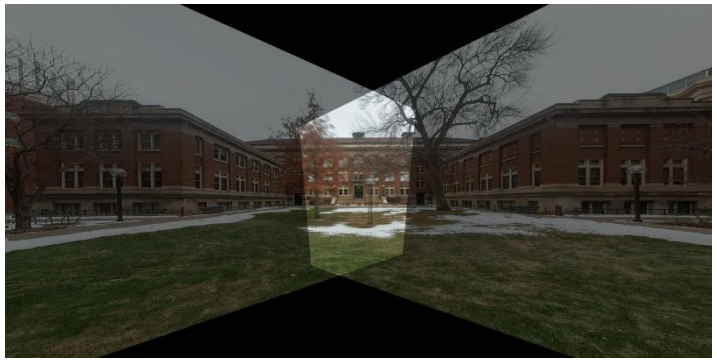
$$R = K^{-1} H K$$

$$\tilde{u} = \begin{bmatrix} f\cos\theta \\ \\ f\sin\theta \end{bmatrix}$$

HW #2: Image Panorama (Cylindrical Projection)



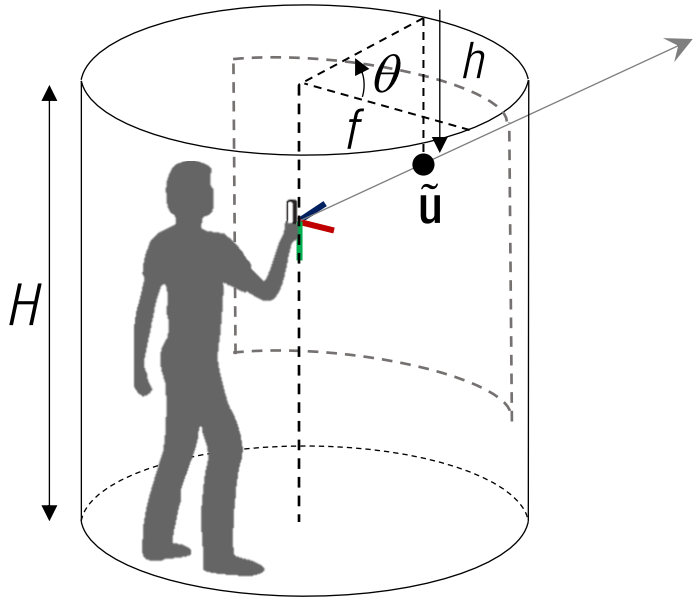
$$X = \lambda K^{-1} R^{T} u$$



$$R = K^{-1} H K$$

$$\tilde{u} = \begin{bmatrix} f\cos\theta \\ h - \frac{H}{2} \\ f\sin\theta \end{bmatrix}$$

HW #2: Image Panorama (Cylindrical Projection)



$$X = \lambda K^{-1} R^{T} u$$

$$R = K^{-1} H K$$

$$\tilde{u} = \begin{bmatrix} f\cos\theta \\ h - \frac{H}{2} \\ f\sin\theta \end{bmatrix} = \lambda K^{-1} R^{T} u$$

HW #2: Image Panorama (Cylindrical Projection)