



Intro to

Computer Vision

with Prof. Kosta Derpanis

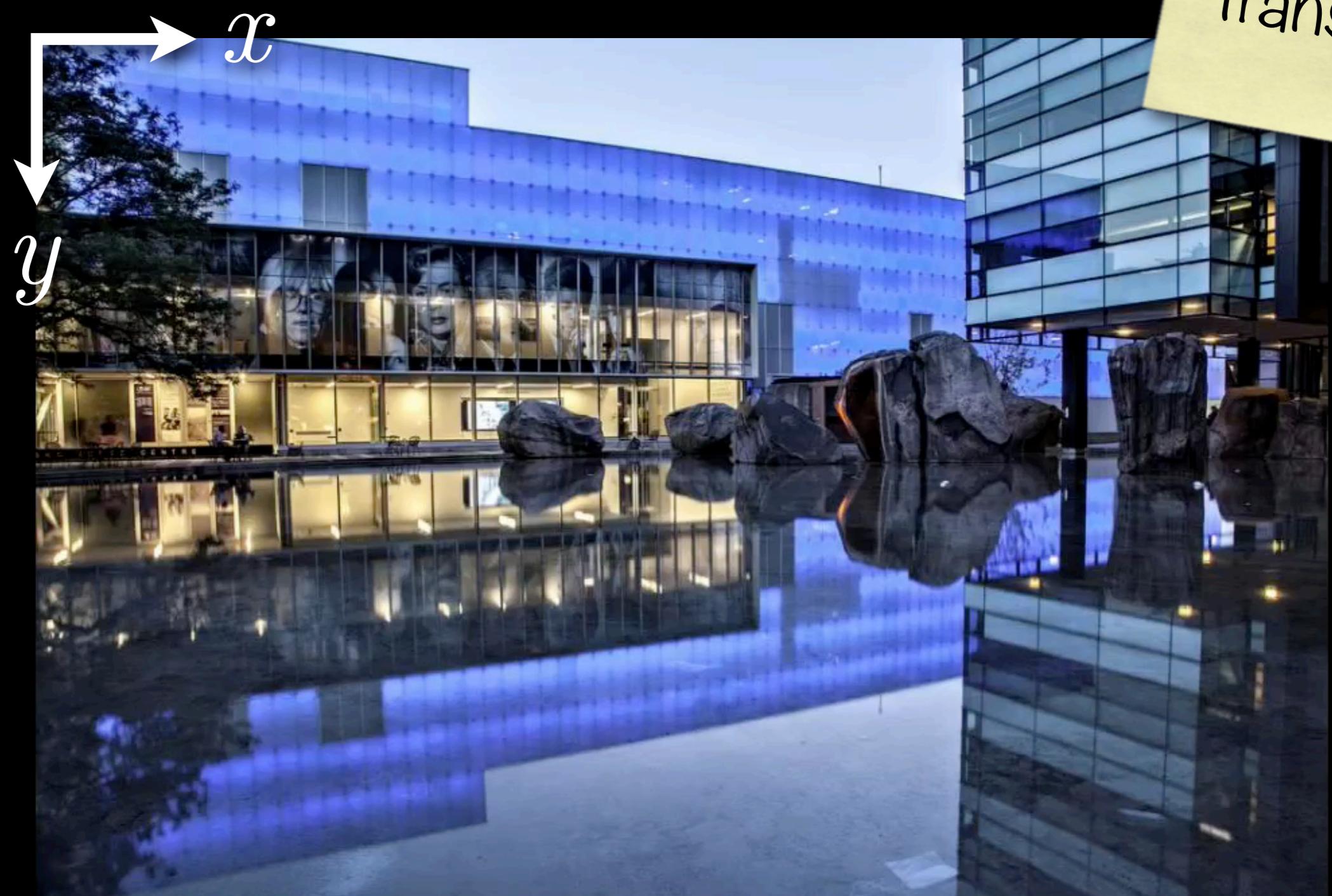
Image Transformations

translation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} \mathbf{I}_{2 \times 2} & \mathbf{t}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & 1 \end{pmatrix} \mathbf{x}$$

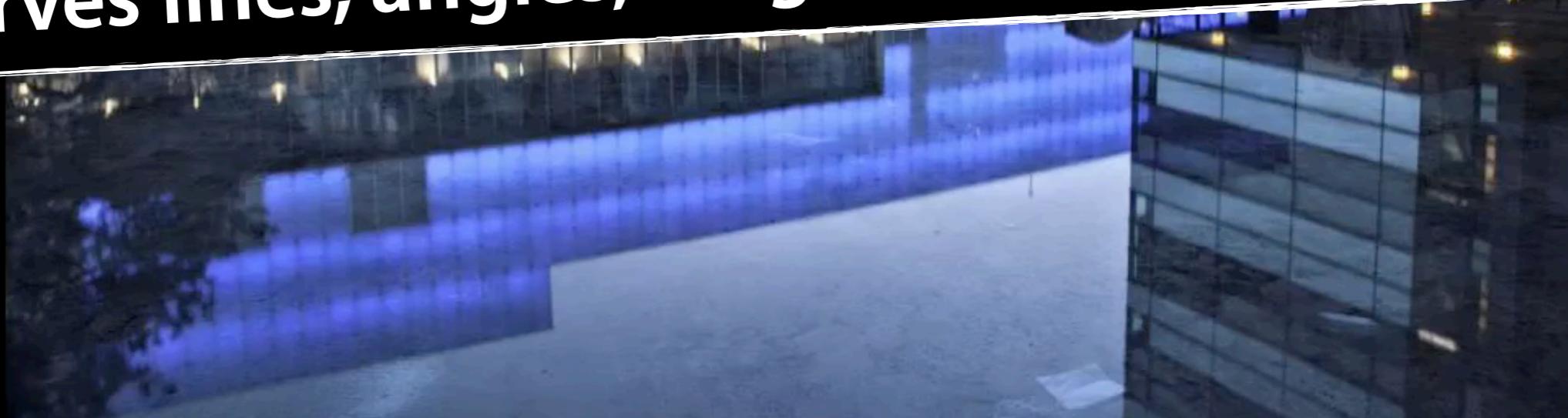
two degrees of freedom



translation



preserves lines, angles, lengths, areas and orientations



Euclidean

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{R}_{2 \times 2} & \mathbf{t}_{2 \times 1} \\ \mathbf{0}_{1 \times 2} & 1 \end{pmatrix} \mathbf{x}$$

three degrees of freedom



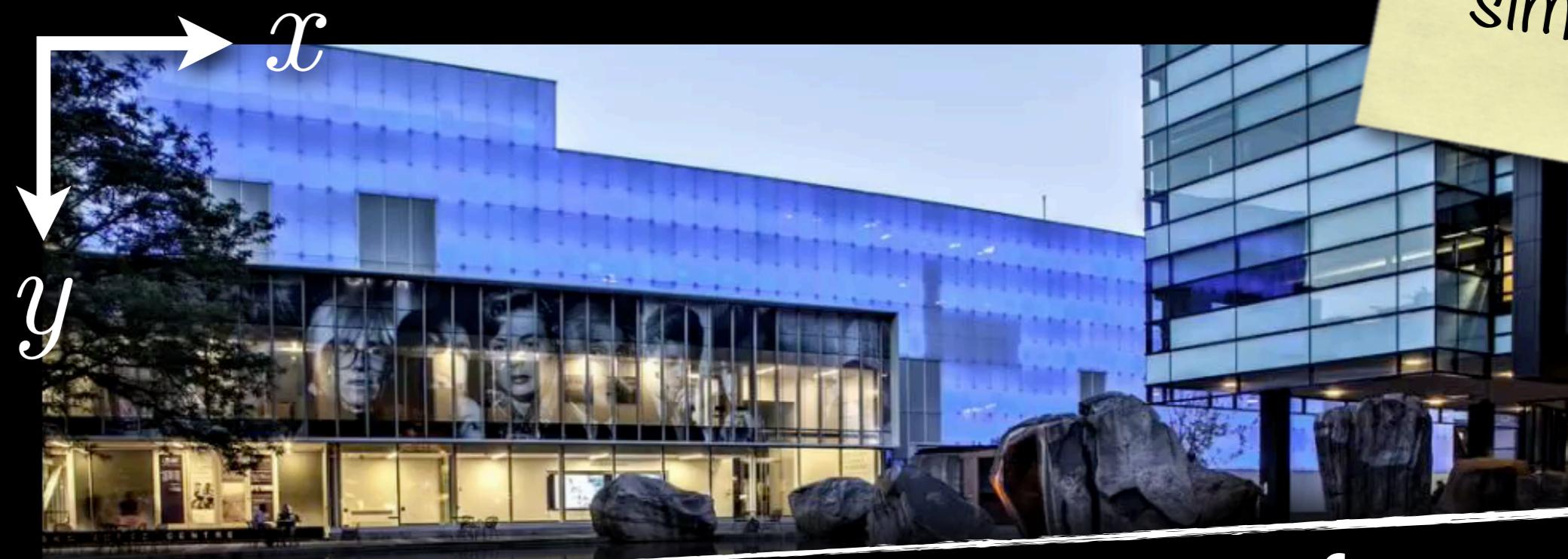
preserves lines, angles, lengths and areas



similarity

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \cos \theta & -\alpha \sin \theta & t_x \\ \alpha \sin \theta & \alpha \cos \theta & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
$$= \begin{pmatrix} \alpha R_{2 \times 2} & t_{2 \times 1} \\ 0_{1 \times 2} & 1 \end{pmatrix} \mathbf{x}$$

four degrees of freedom



preserves lines, angles and ratios of areas



affine

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

six degrees of freedom



preserves lines, parallel lines and ratios of areas



homography

$$w \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

divide by last component to complete transformation

homography

$$w \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

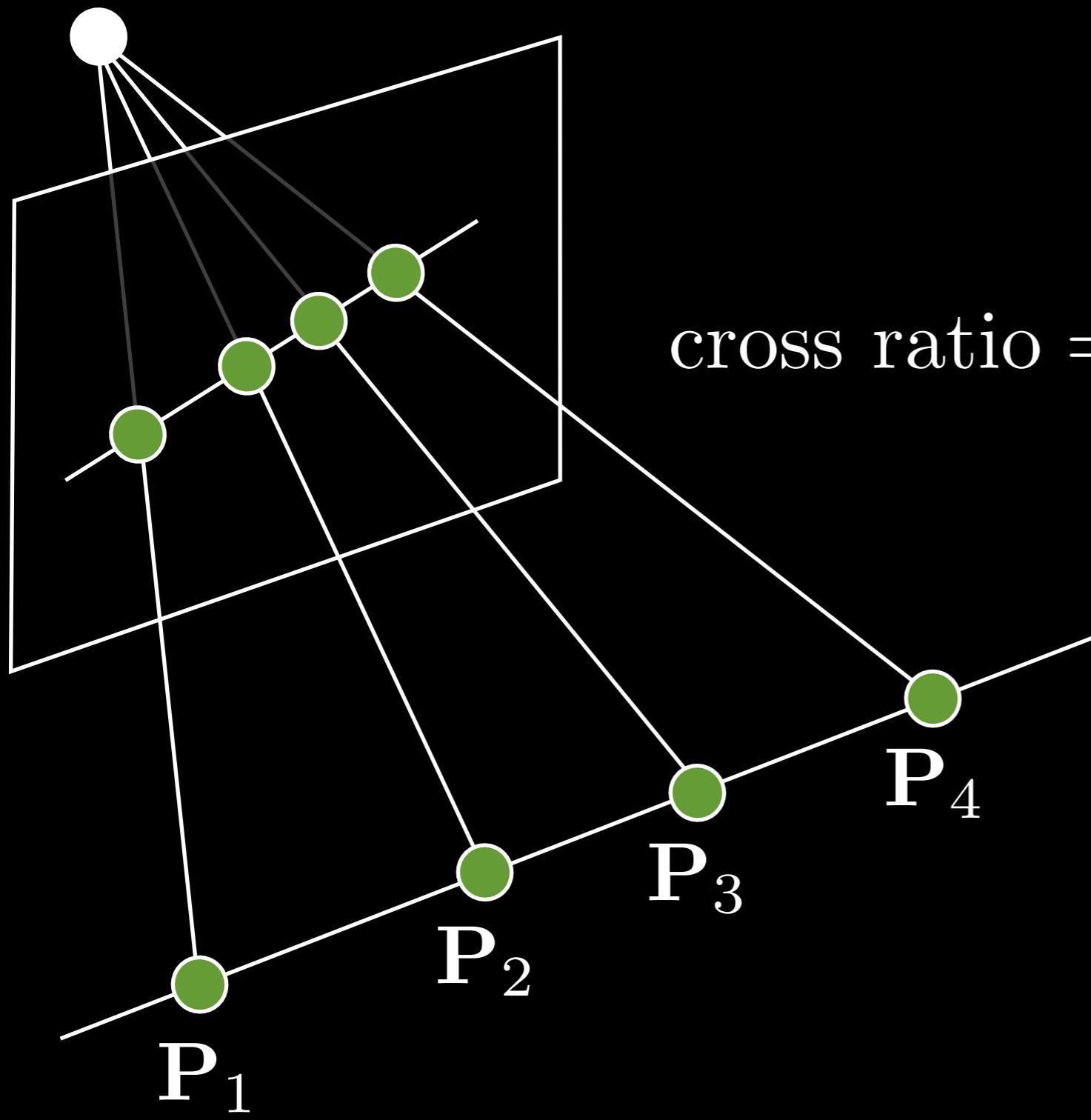
eight degrees of freedom



preserves lines and cross ratio

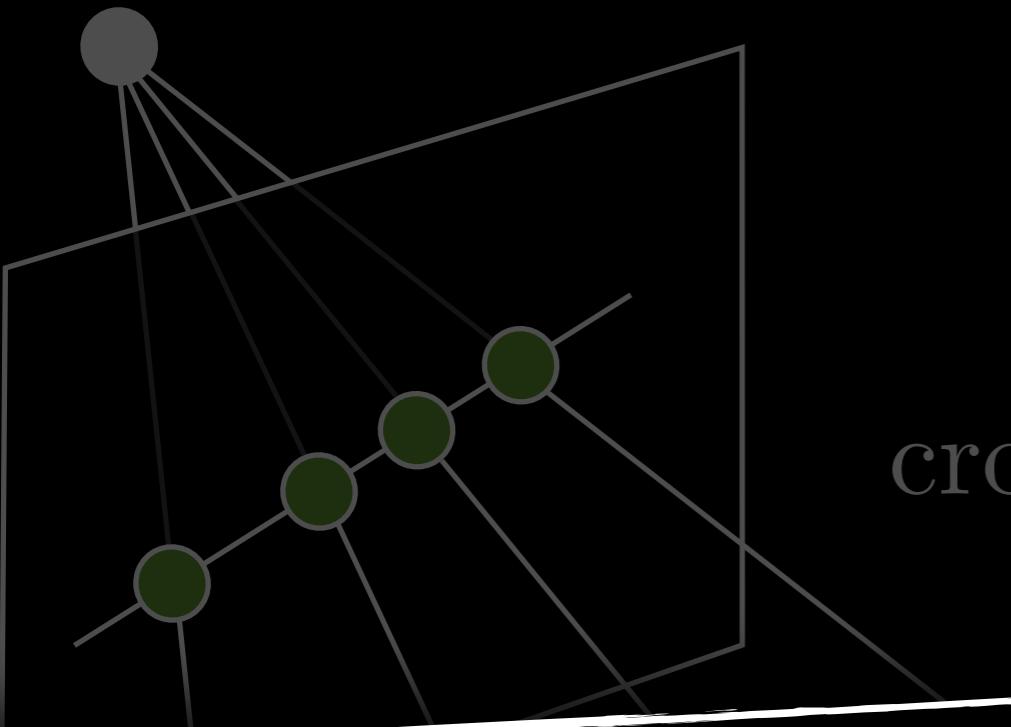


cross ratio



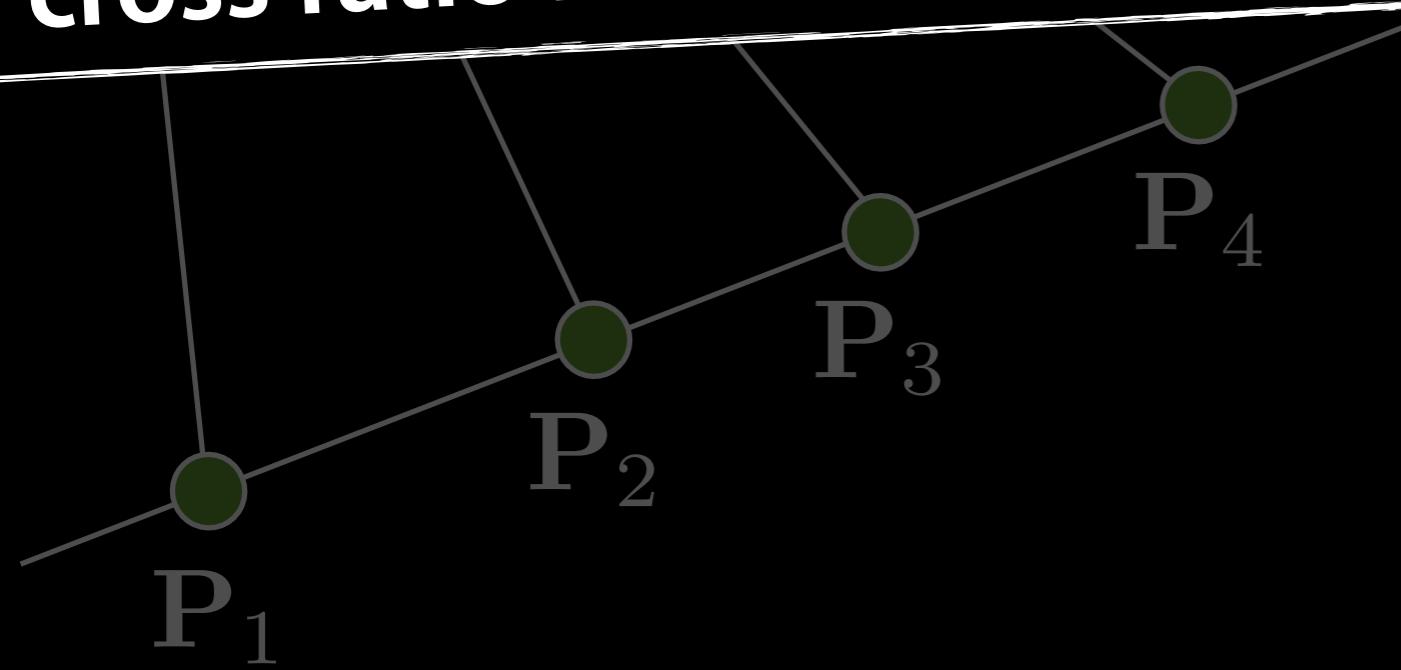
$$\text{cross ratio} = \frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|}$$

cross ratio

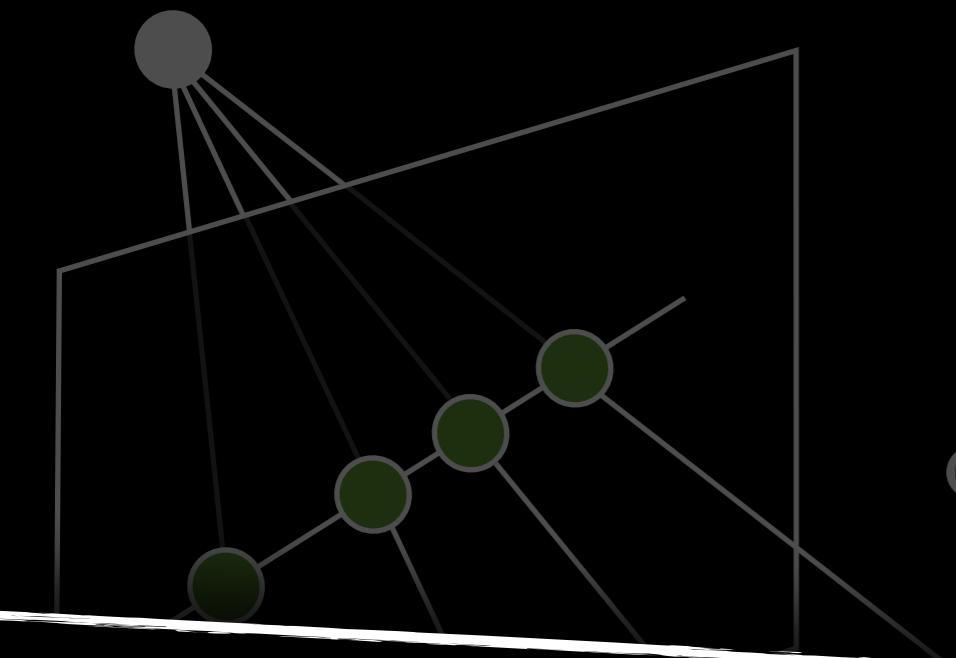


$$\text{cross ratio} = \frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|}$$

cross ratio is invariant under perspective projection

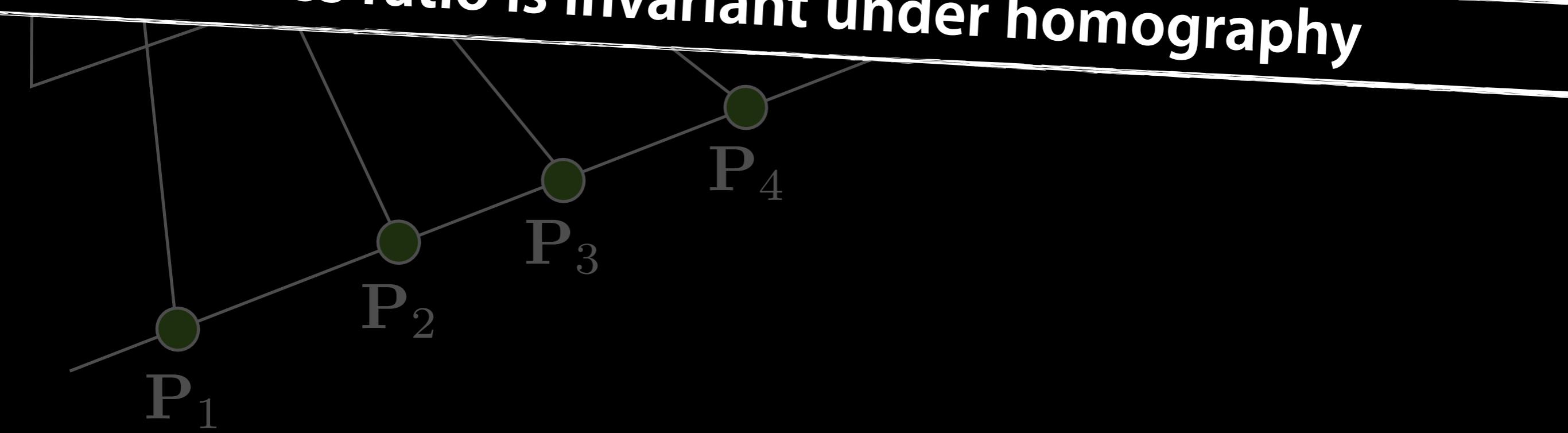


cross ratio

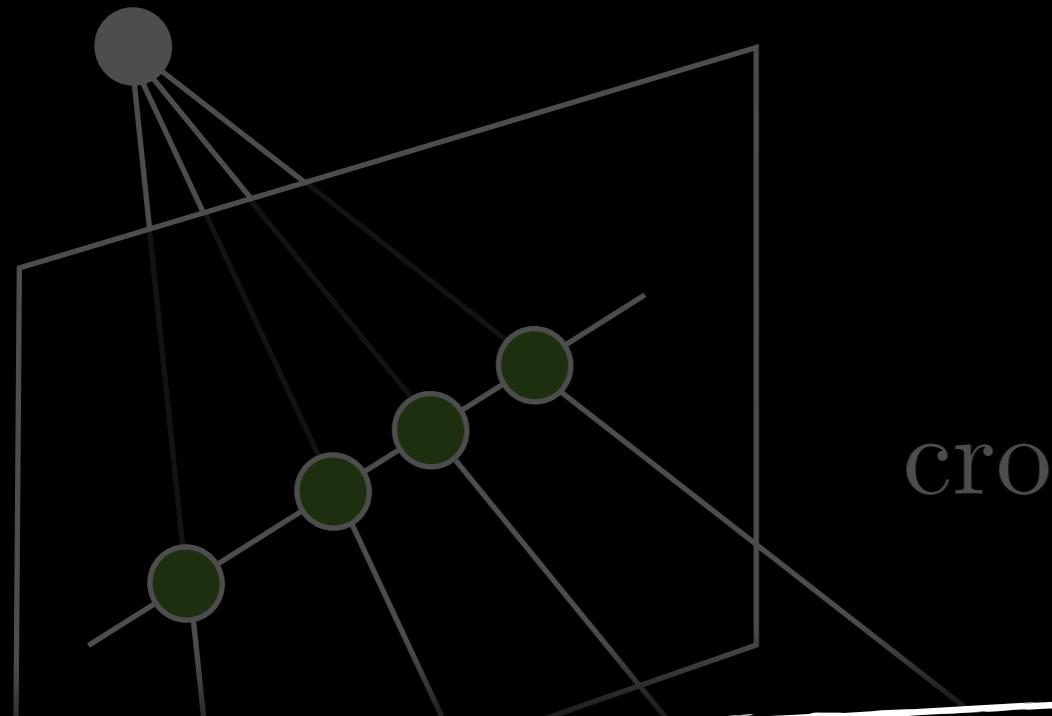


$$\text{cross ratio} = \frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|}$$

cross ratio is invariant under homography

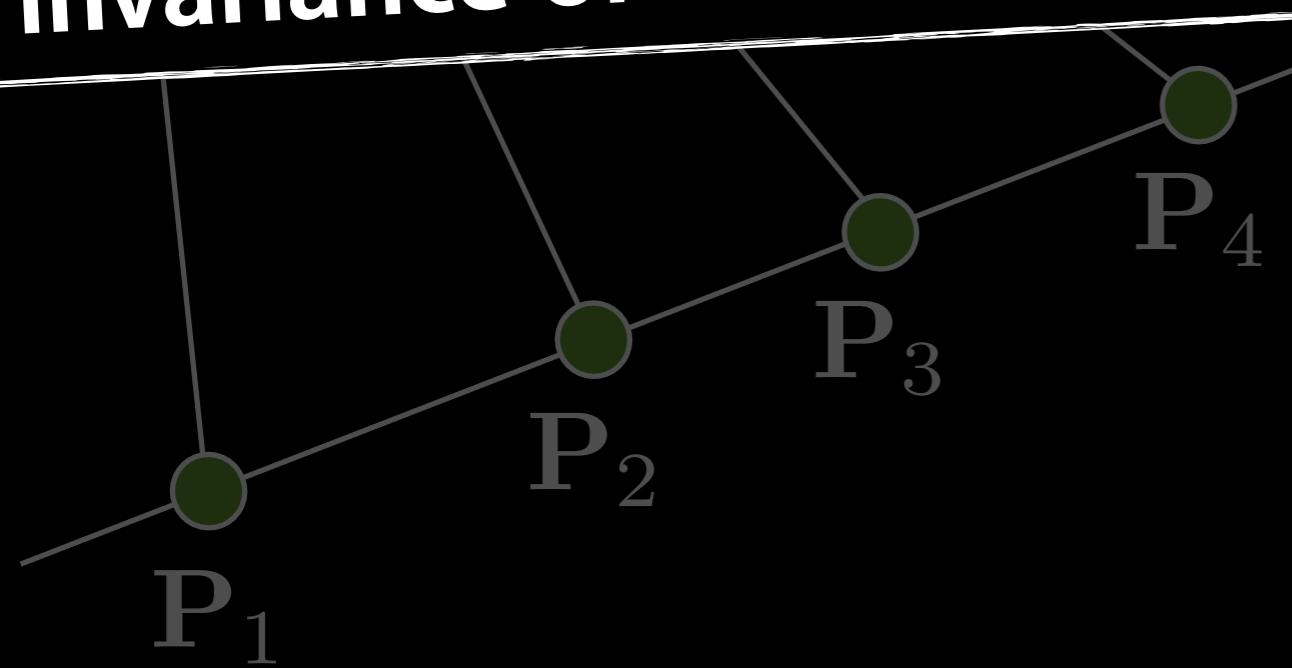


cross ratio



$$\text{cross ratio} = \frac{\|P_3 - P_1\| \|P_4 - P_2\|}{\|P_3 - P_2\| \|P_4 - P_1\|}$$

invariance of cross ratio remains under permutation

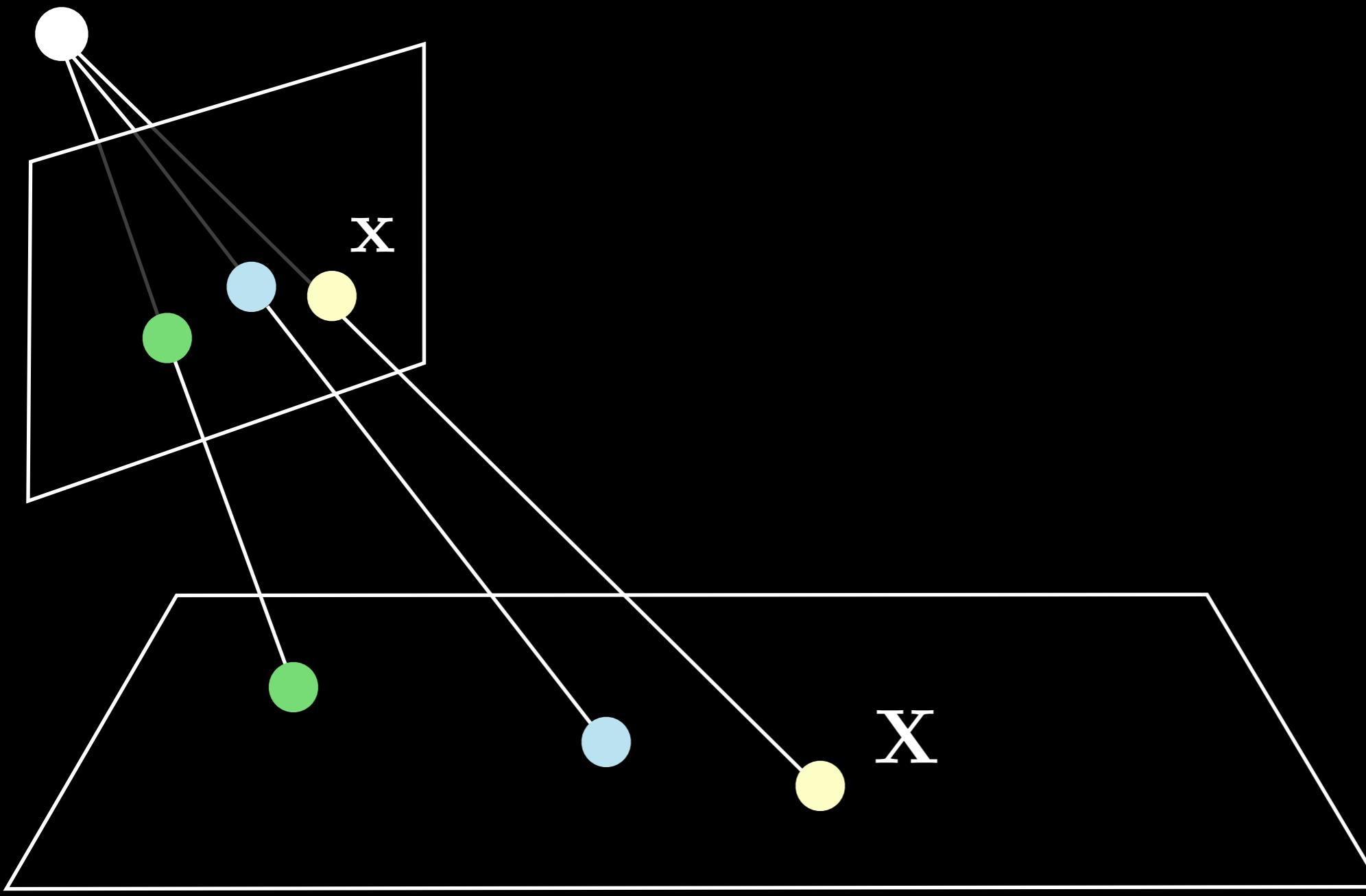


transformations are nested



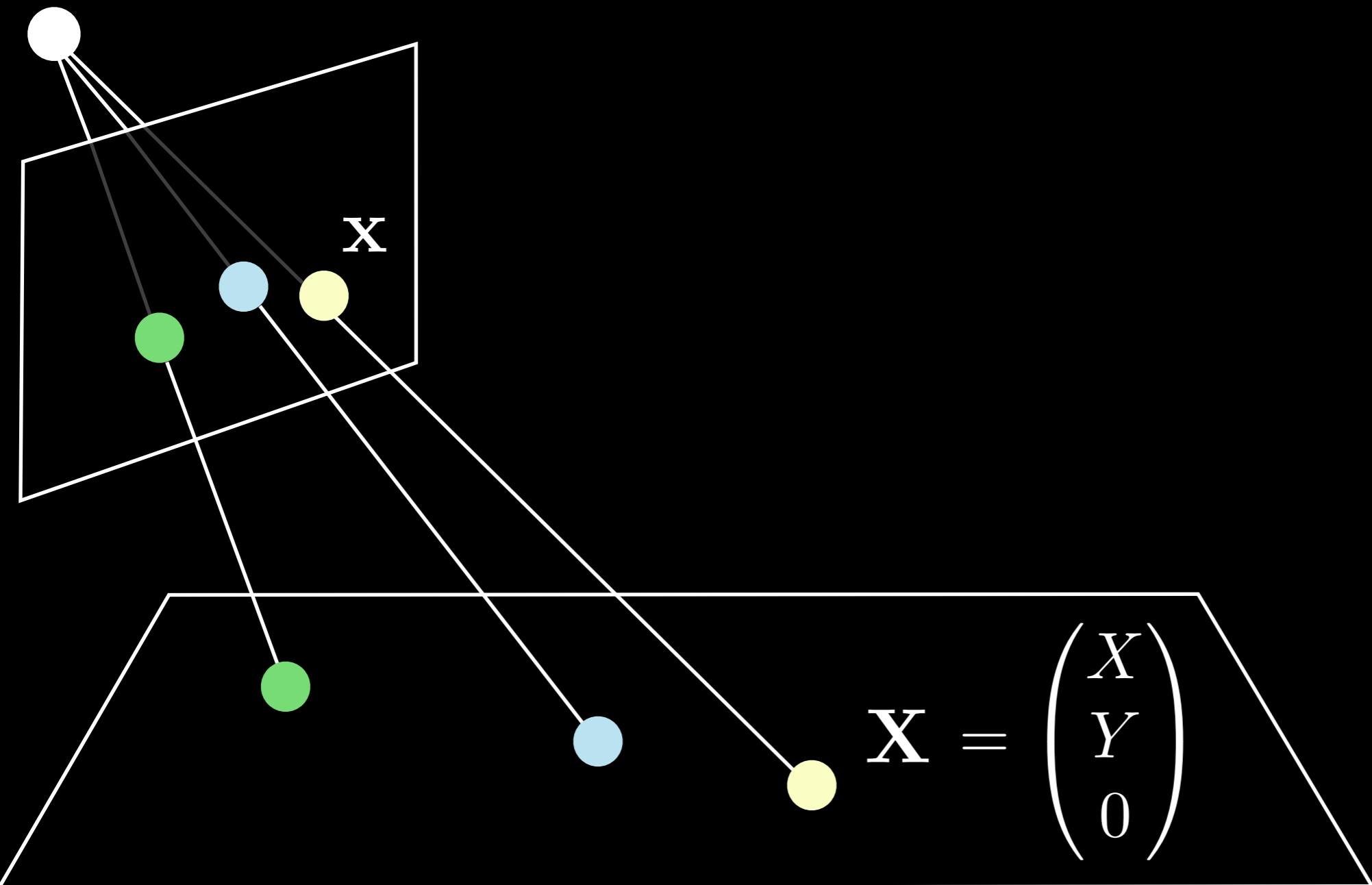
image

$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{M}_{\text{int}} \quad \mathbf{M}_{\text{ext}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$



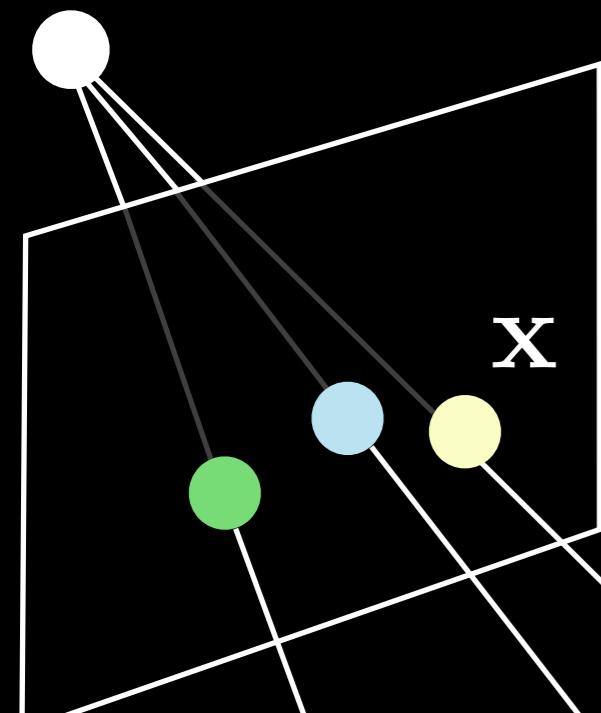
image

$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{M}_{\text{int}} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$



image

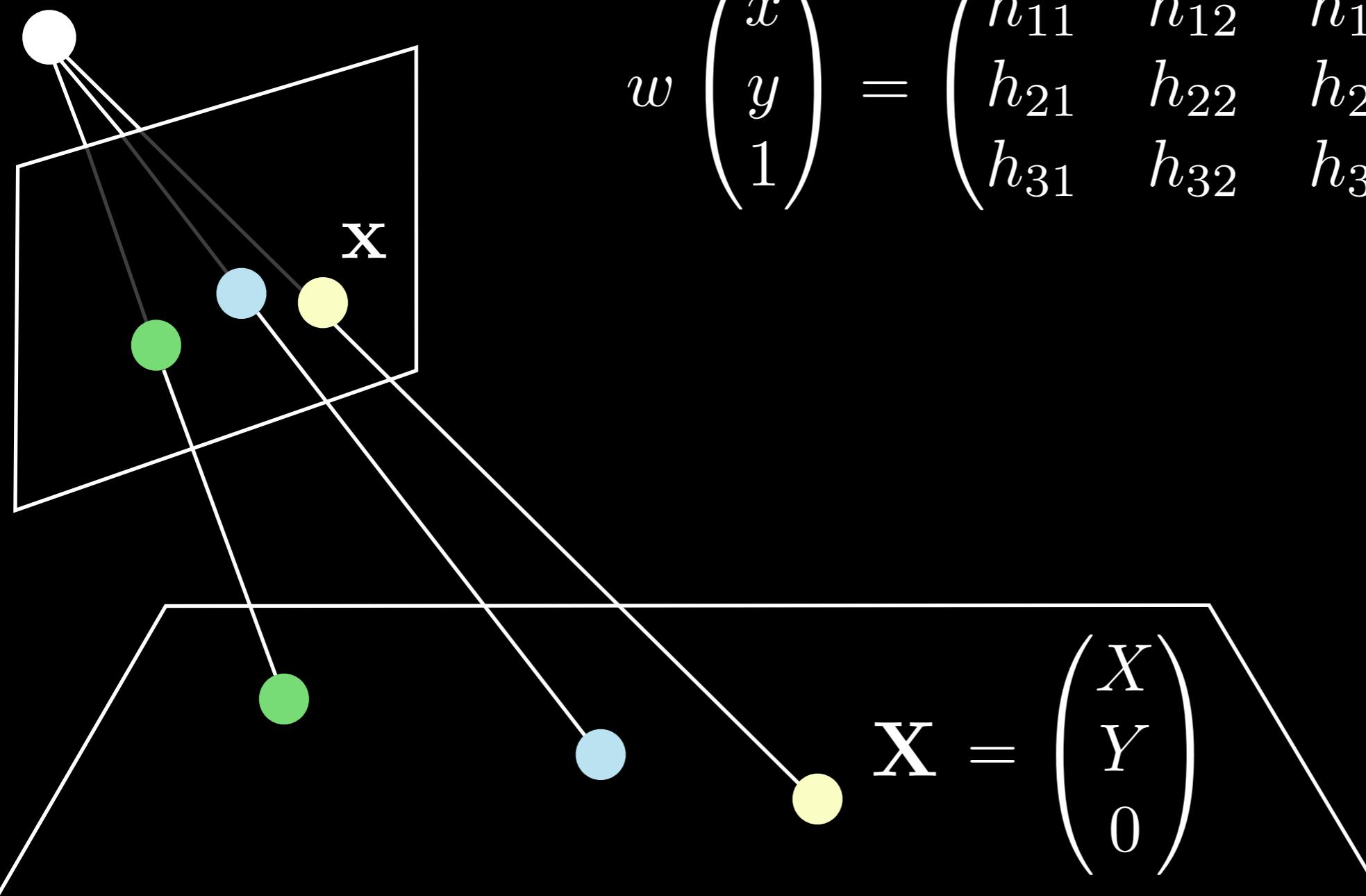
$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{M}_{\text{int}} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix}$$



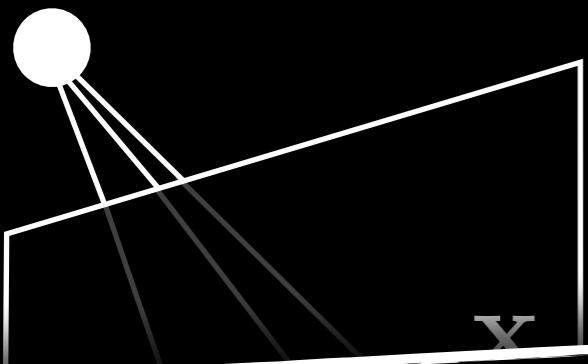
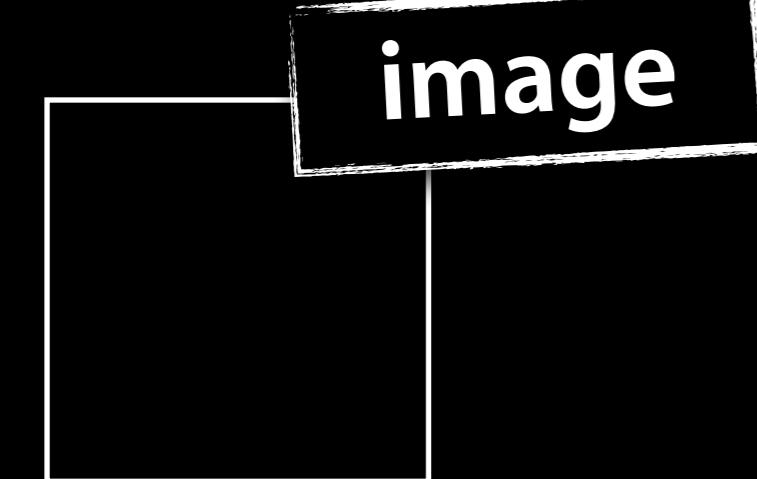
$$= \mathbf{M}_{\text{int}} \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

image

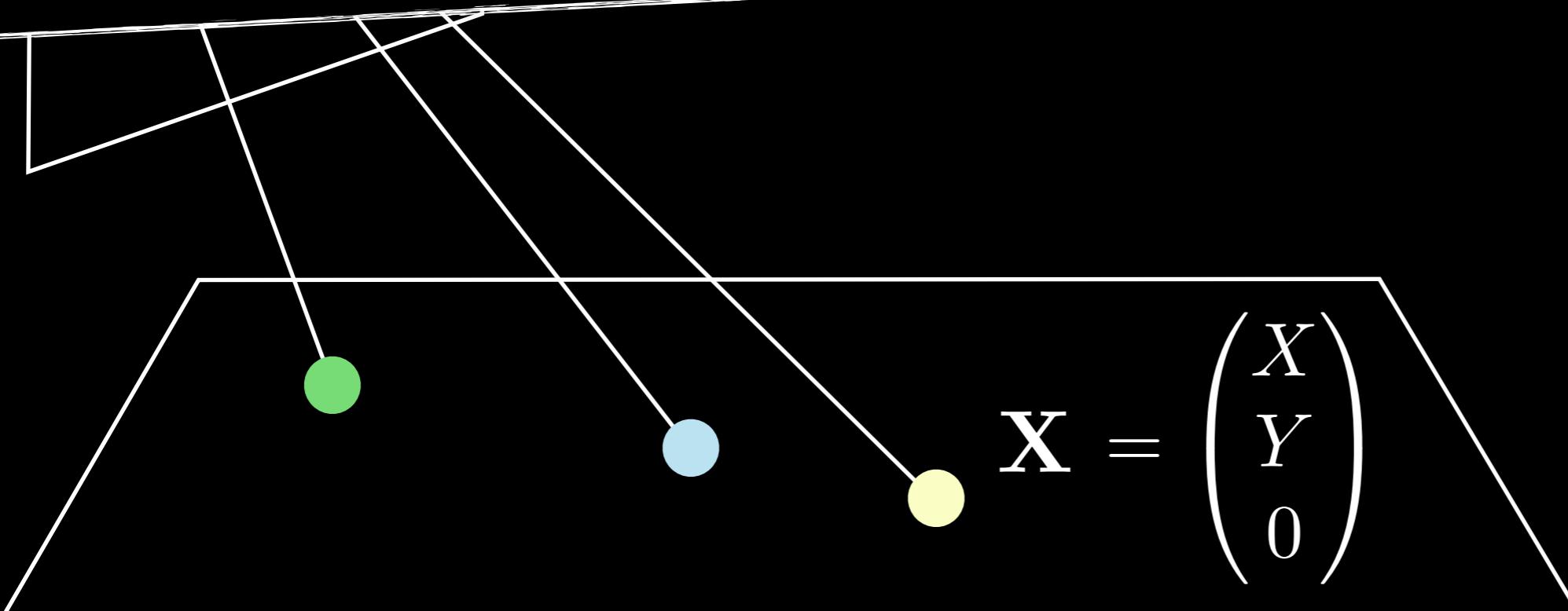


image



$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

3D-to-2D projection reduces to 2D-to-2D mapping



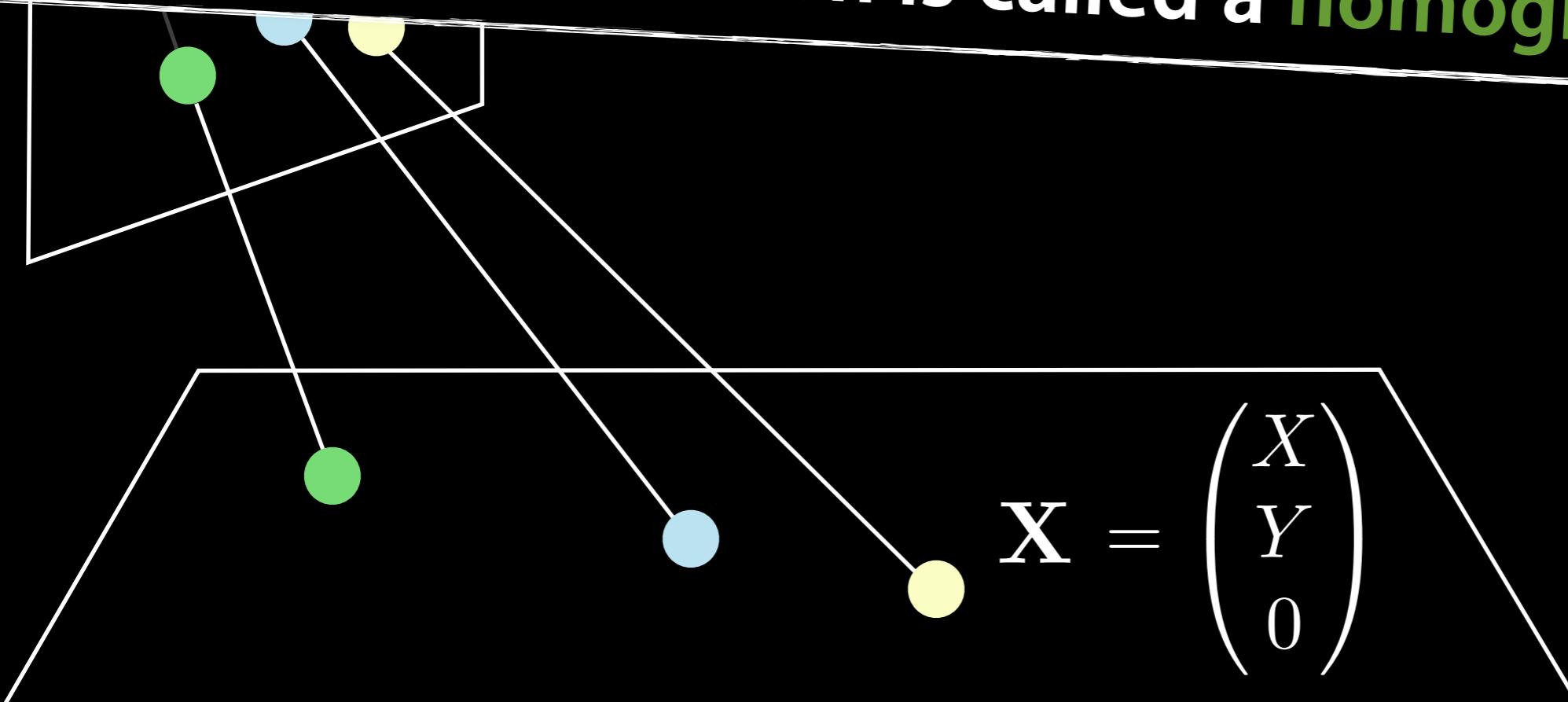
$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

image



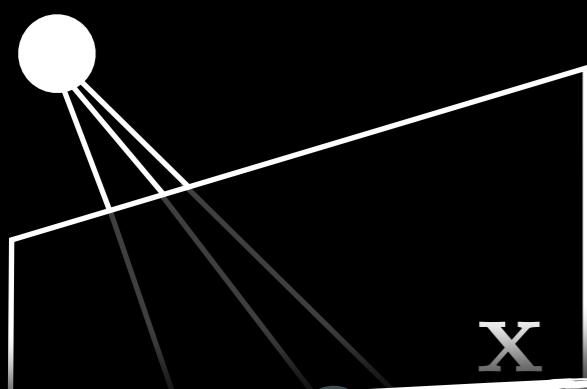
$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

transformation is called a **homography**



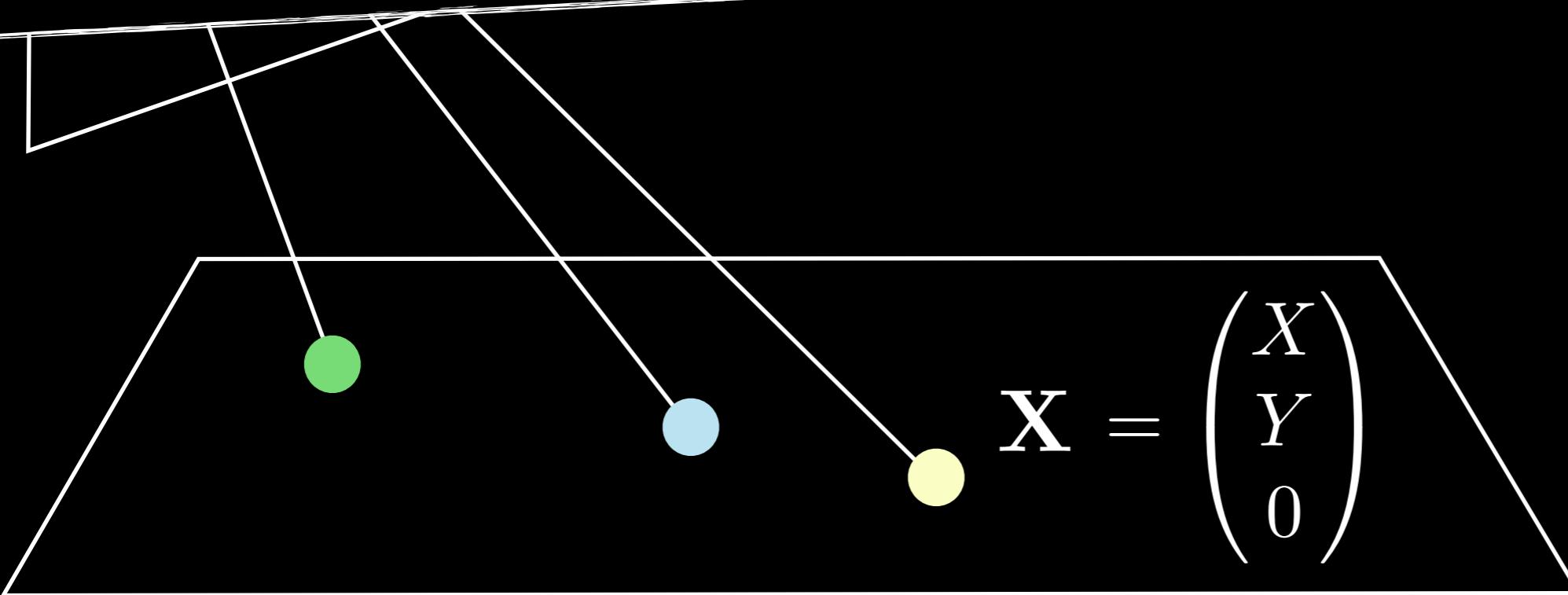
$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

image



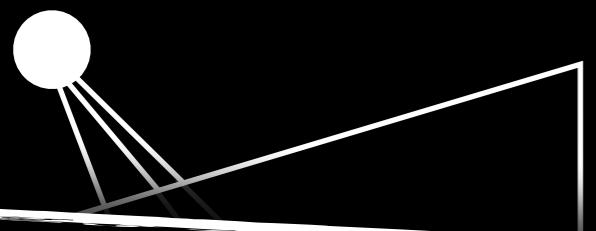
$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

homography is an invertible transformation



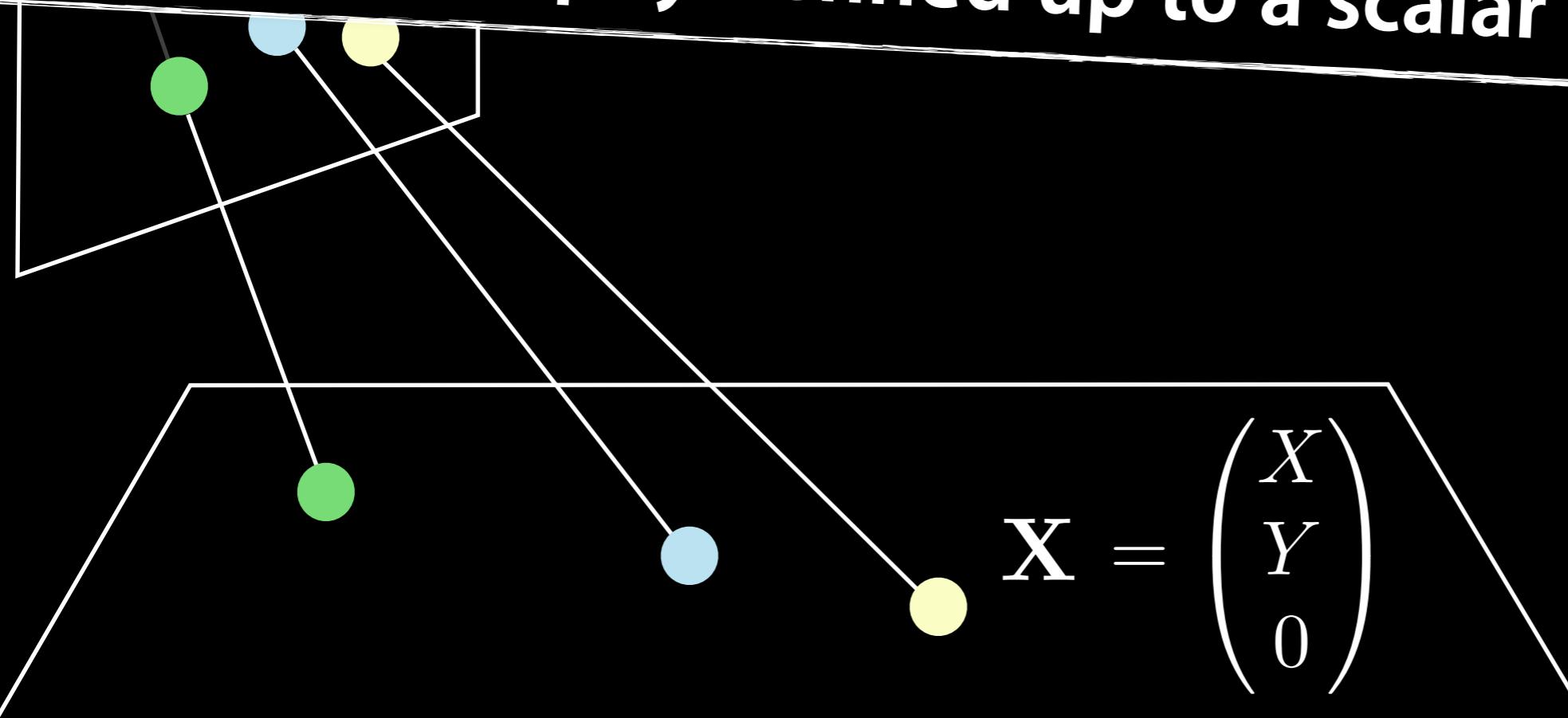
$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

image



$$w \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

homography defined up to a scalar multiple



$$\mathbf{X} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}$$

16

ND & 10

2ND HALF TOTAL YARDS

120

53

AU

ESPN



FLORIDA ST

13

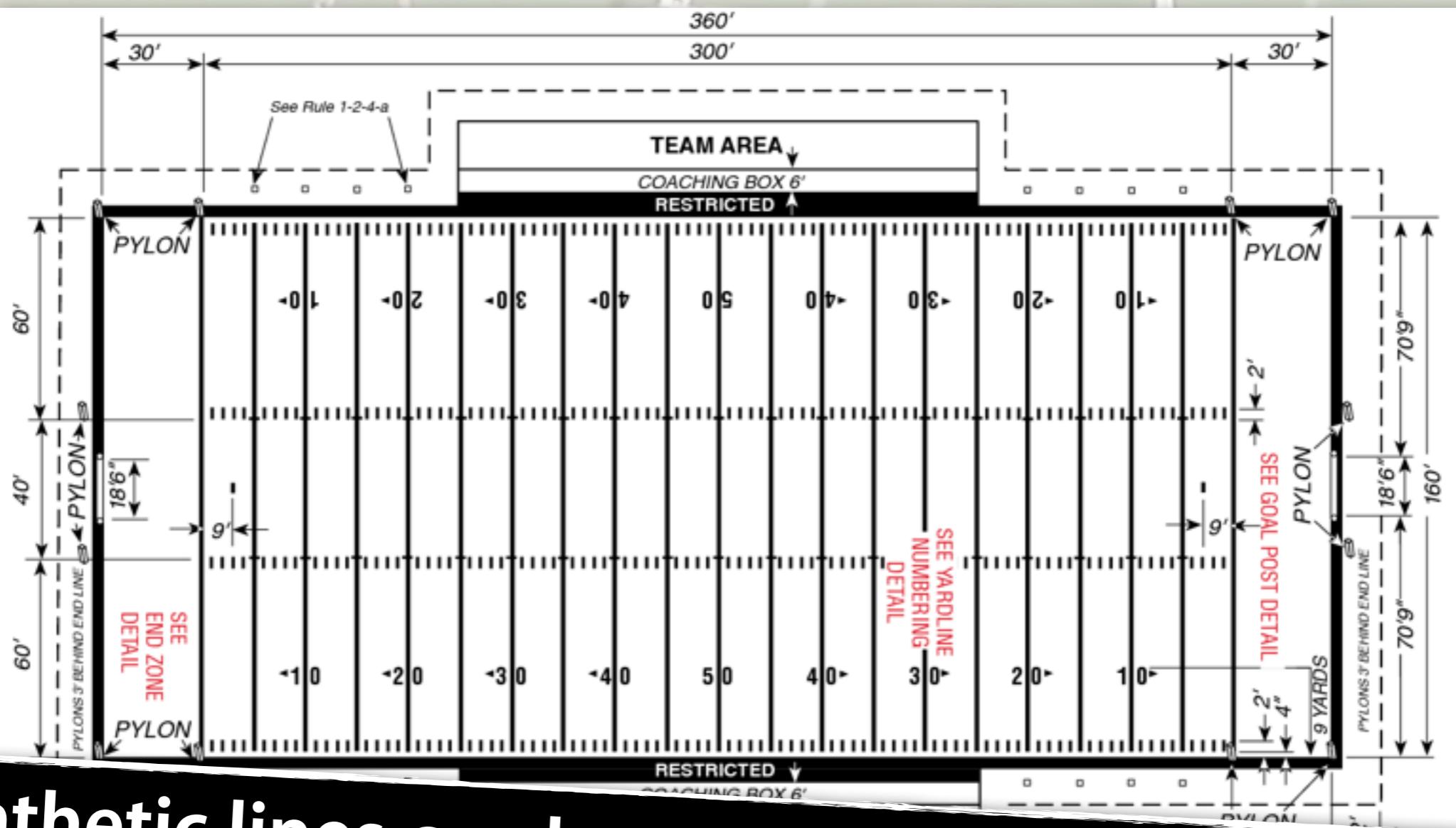
AUBURN

21

4th | 11:35

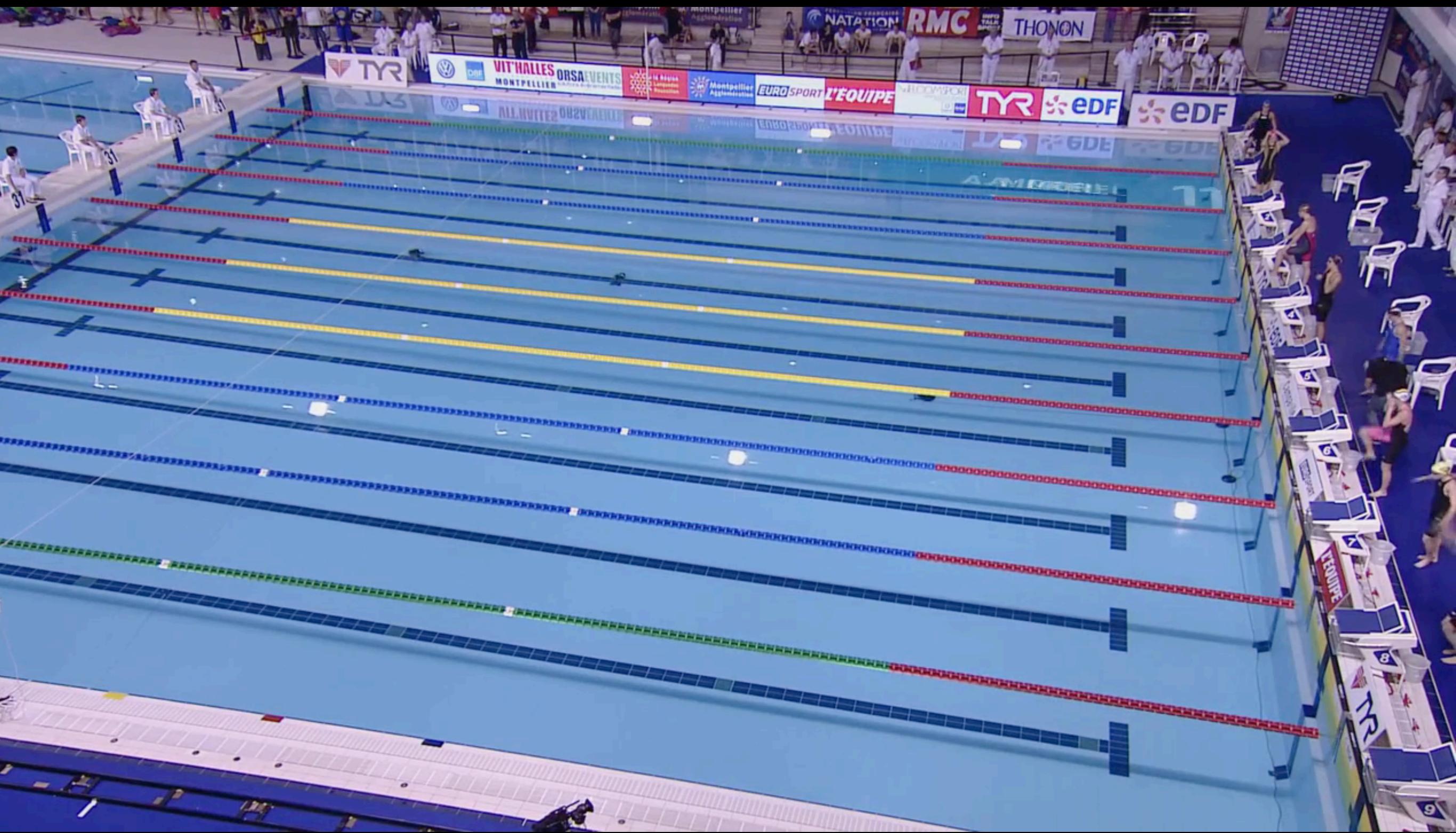
16

2nd & 10

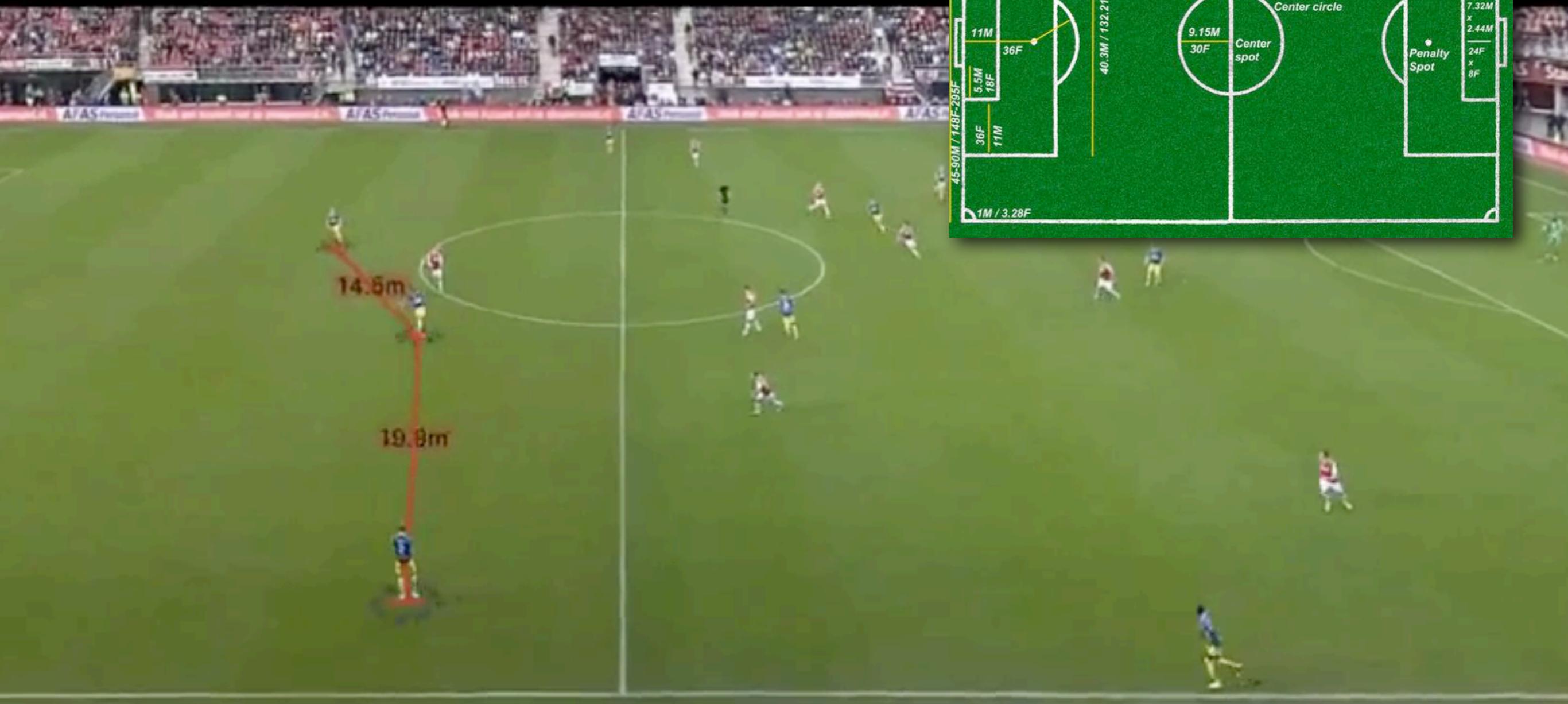
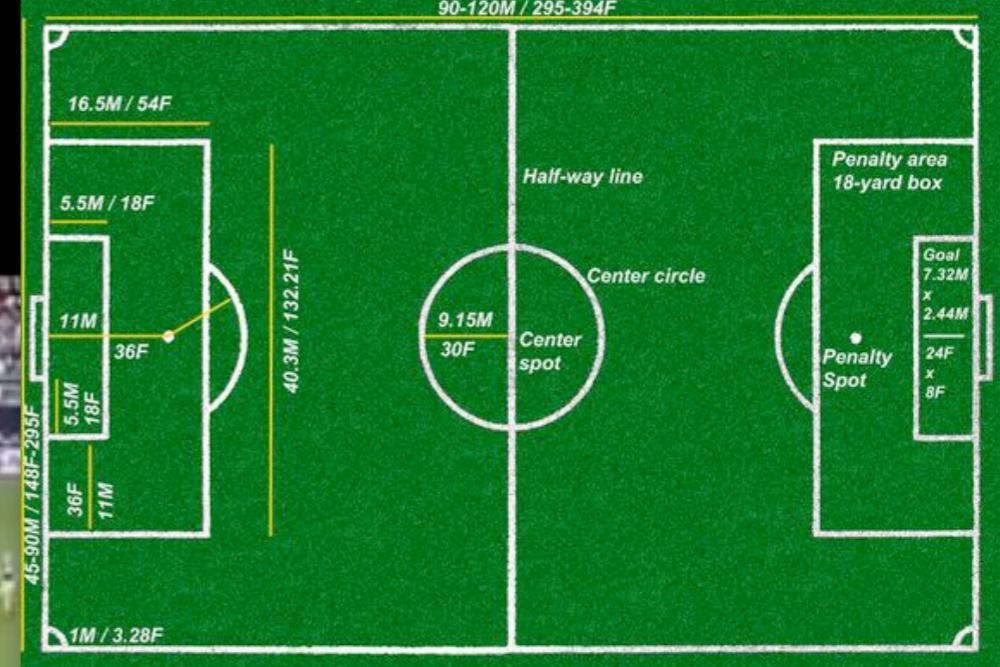


synthetic lines can be mapped by a homography

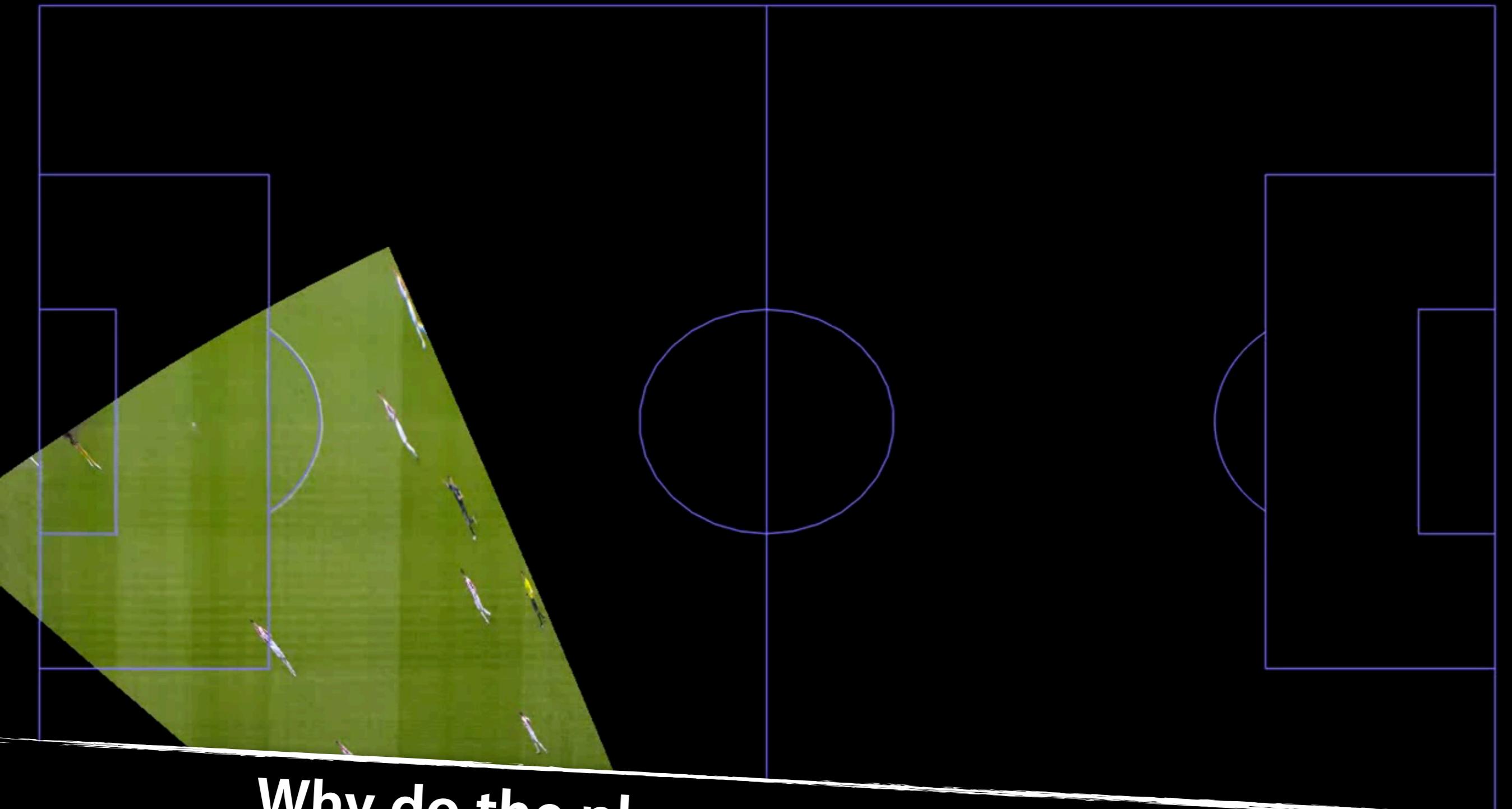




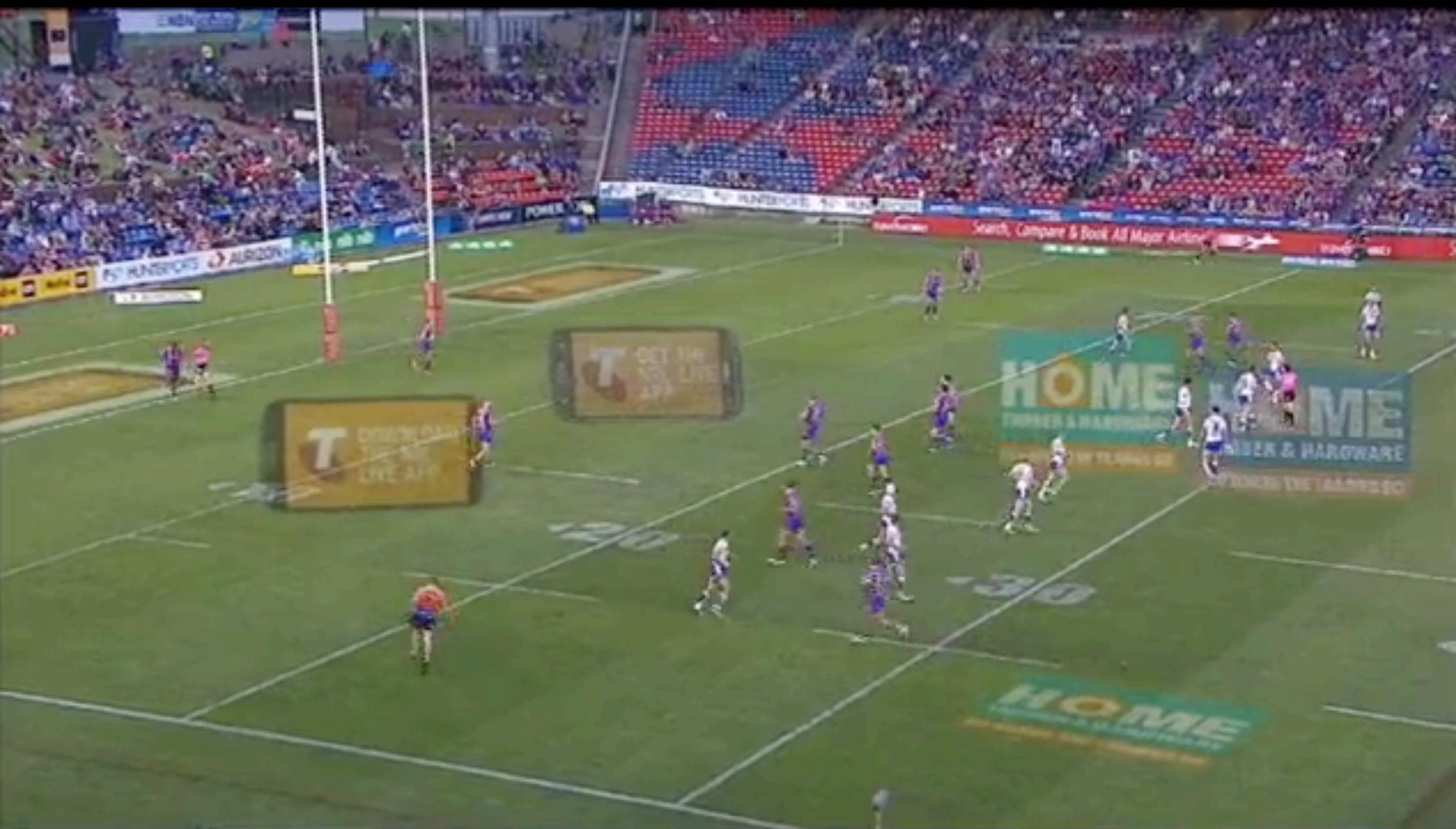
90-120M / 295-394F







Why do the players appear stretched?



Search, Compare & Book All Major Airlines

HOME
CENTER & HARDWARE

HOME CENTER & HARDWARE

HOME
CENTER & HARDWARE

HOME CENTER & HARDWARE

HOME
CENTER & HARDWARE



CS

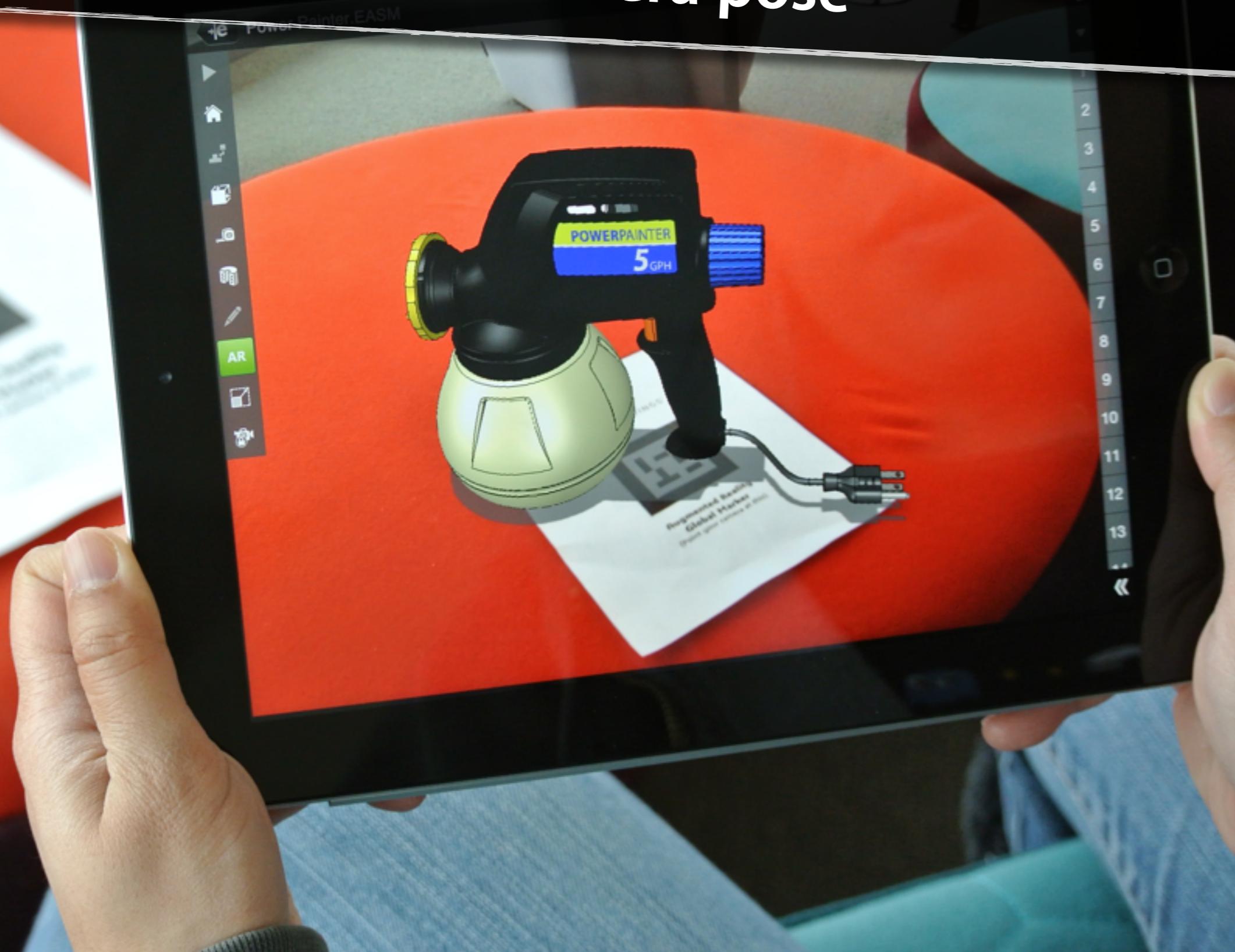


cS



PIERSON
UNIVERSITY

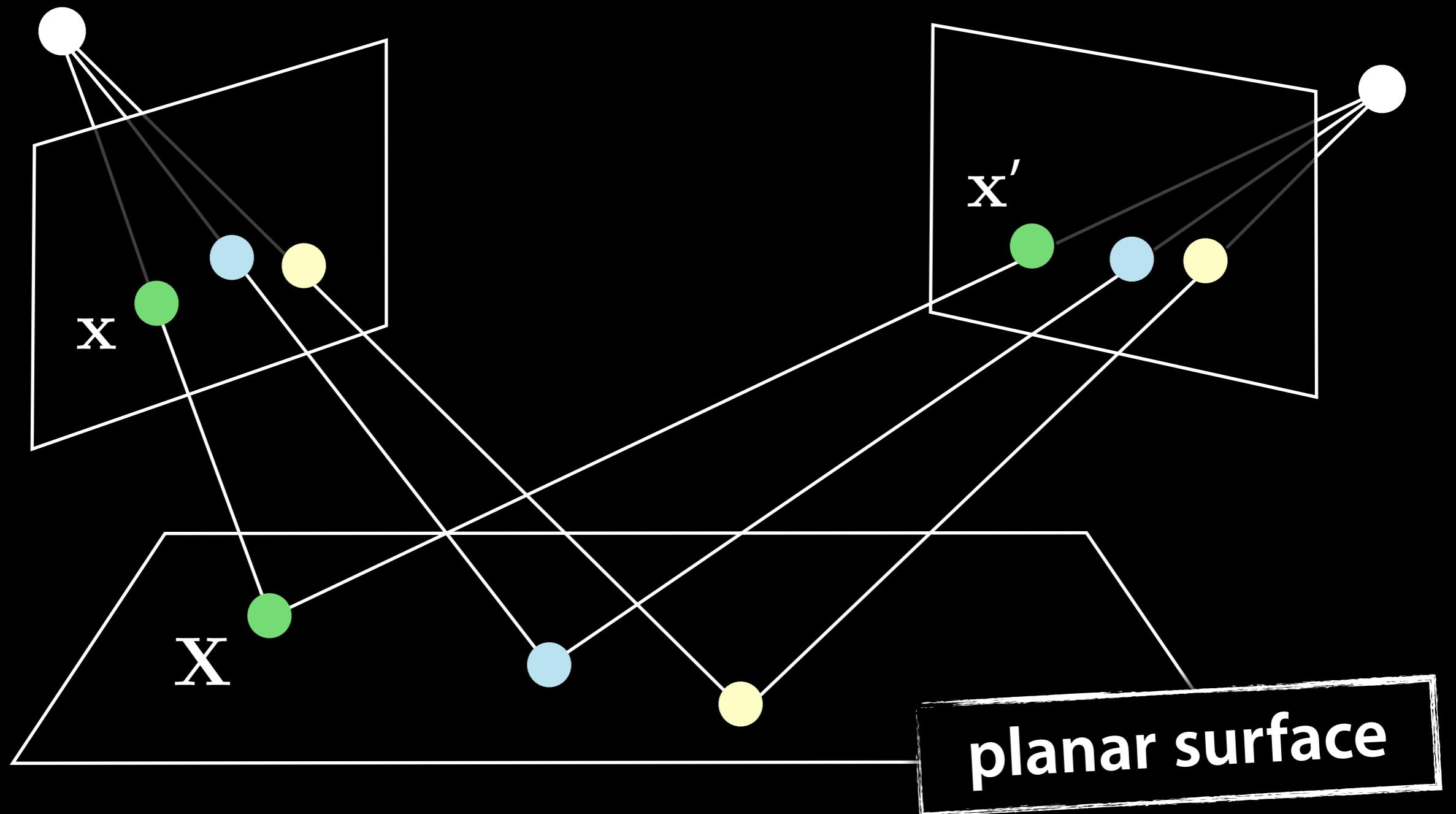
Given a planar homography and the camera intrinsics,
can recover the camera pose



Behind the Scenes

the
creators project

Box by Bot and Dolly
Creators Project



planar surface

image 1

image 2

$$w \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = H_1 H_2^{-1} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

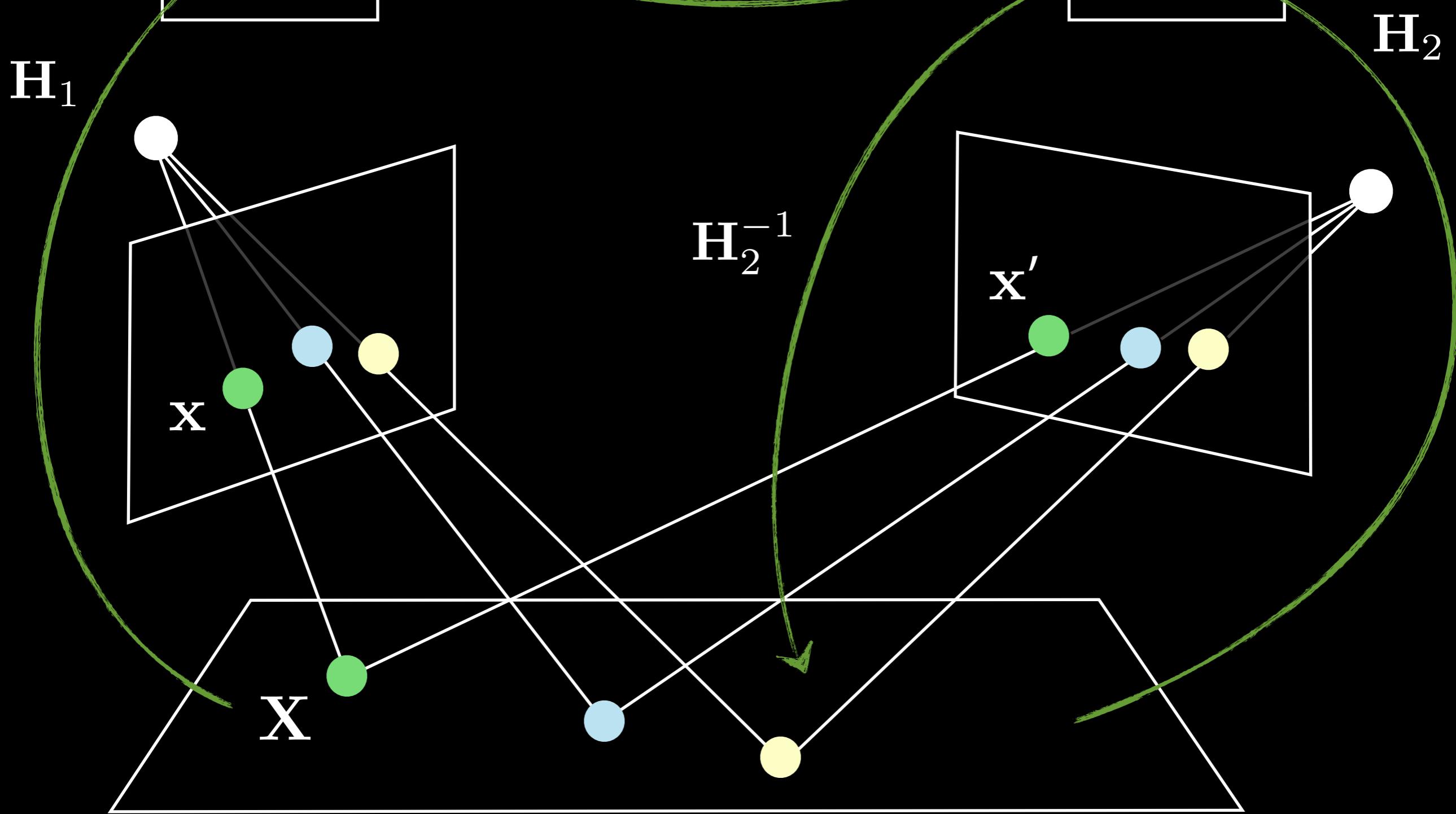
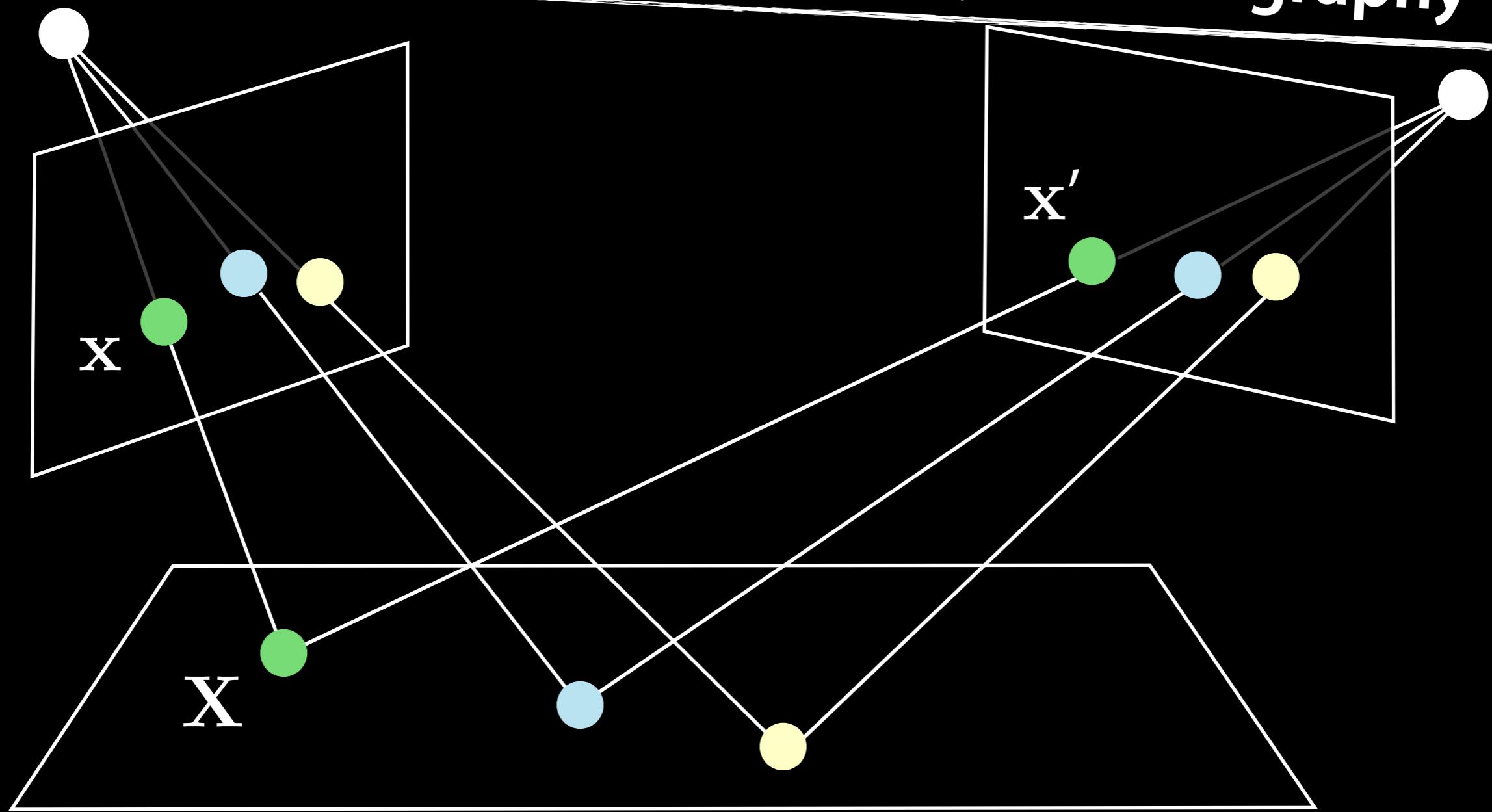


image 1

image 2

$$w \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{H} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

images of a plane are related by a homography





3D Lego Terracotta Army

Leon Keer, Ruben Poncia, Remko van Schaik and Peter Westerink



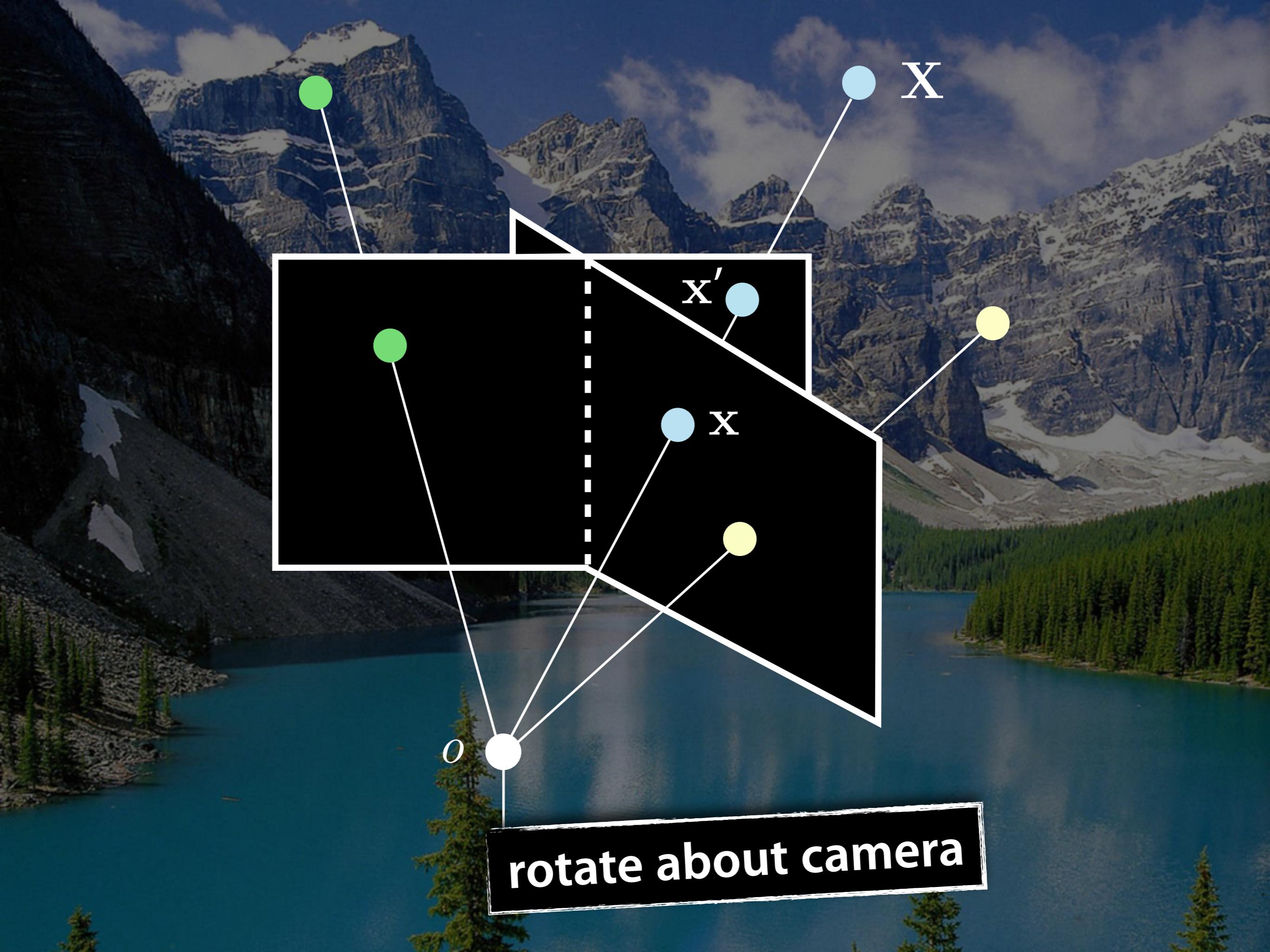


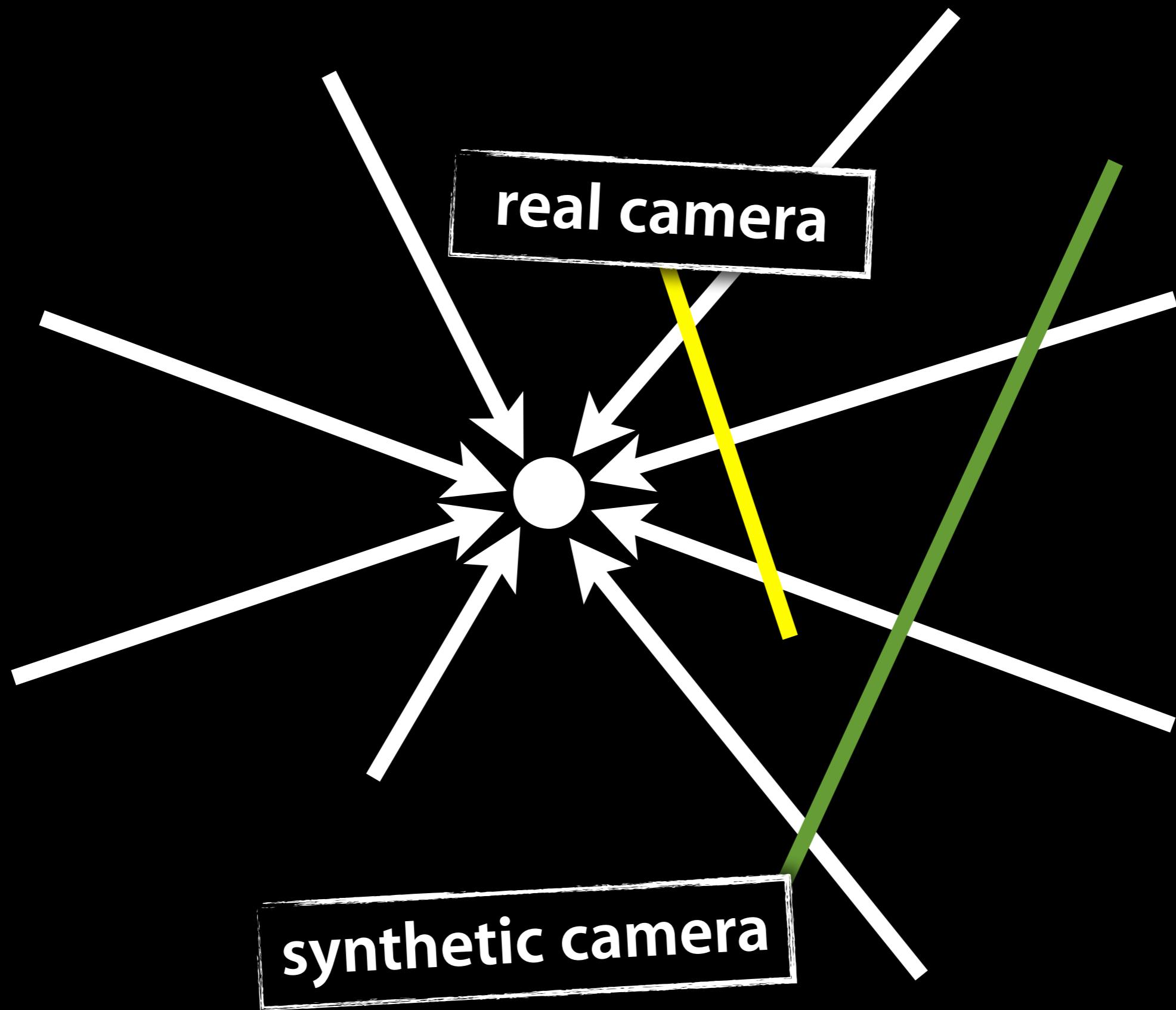




Perspective Warp

ADOBE CREATIVE CLOUD





rotating
camera

$$w_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{M}_{\text{int}} \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$= \mathbf{M}_{\text{int}} \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$= \mathbf{H}_1 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

rotating
camera

$$w_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = \mathbf{H}_1 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$w_2 \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix} = \mathbf{H}_2 \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$w_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = w_2 \mathbf{H}_1 \mathbf{H}_2^{-1} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

rotating
camera

$$w_1 \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = w_2 \mathbf{H}_1 \mathbf{H}_2^{-1} \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

images from a rotating camera are related by a homography

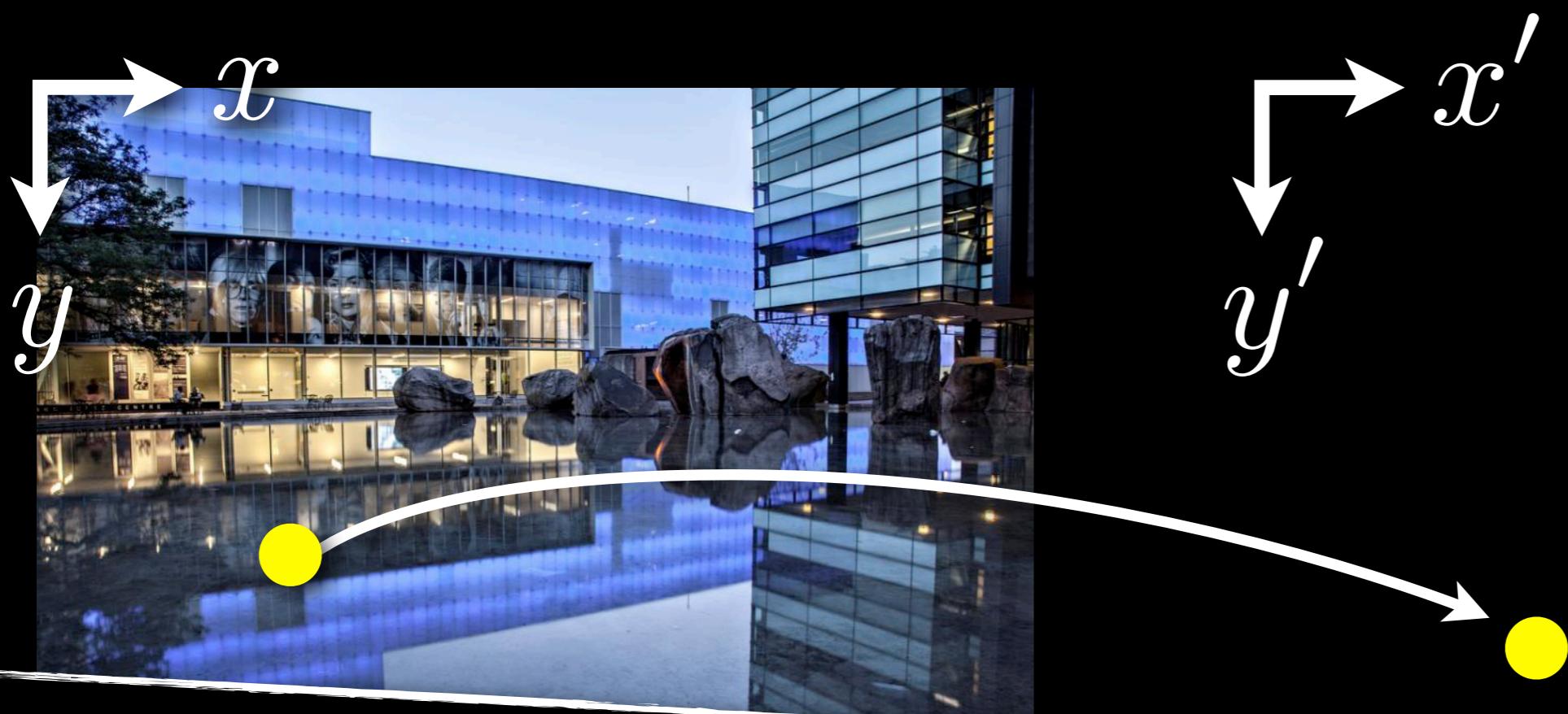
FORWARD v INVERSE

WARP

practical issues when warping images

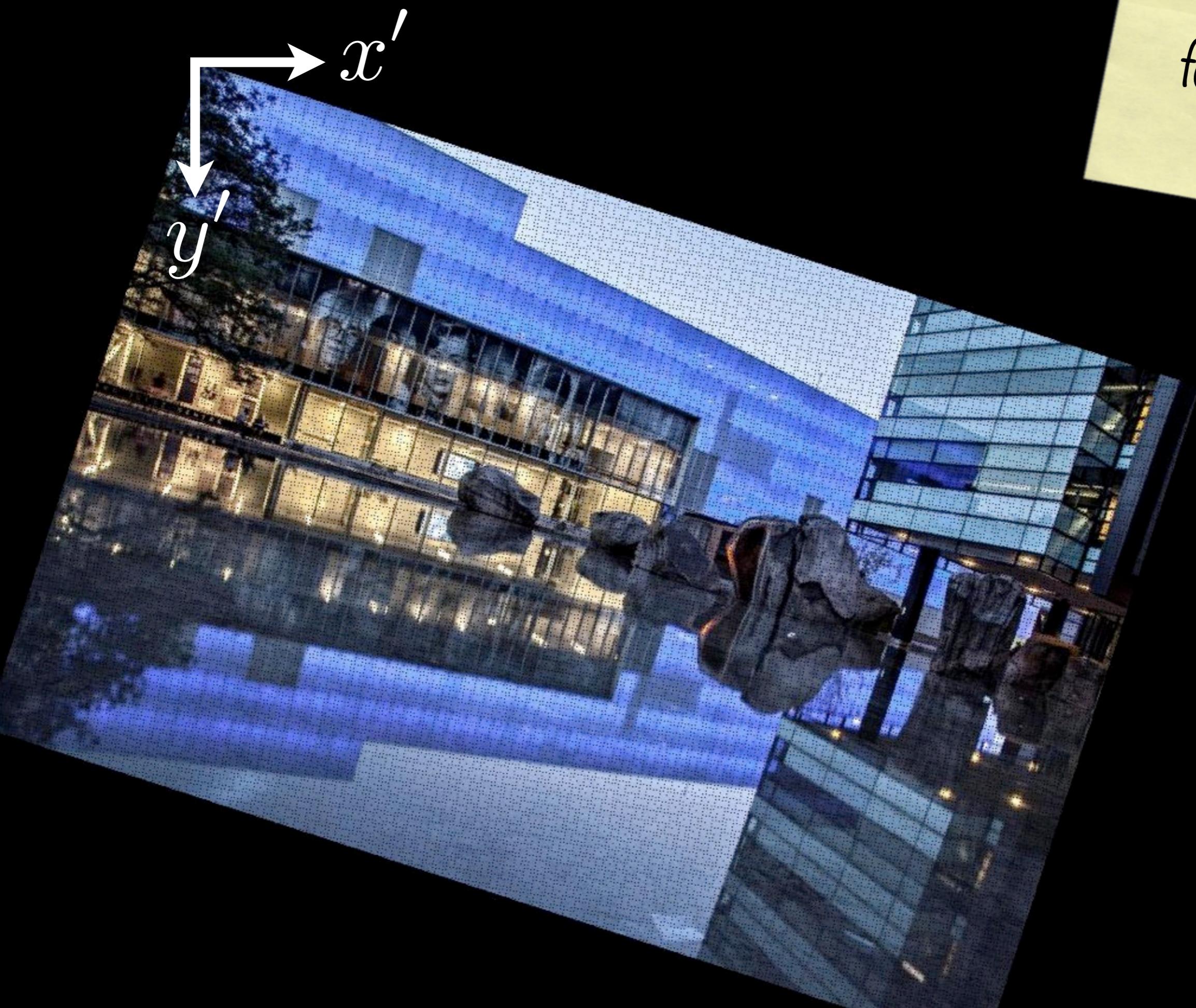
forward
warp

$$\mathbf{x}' = \mathbf{T}\mathbf{x}$$

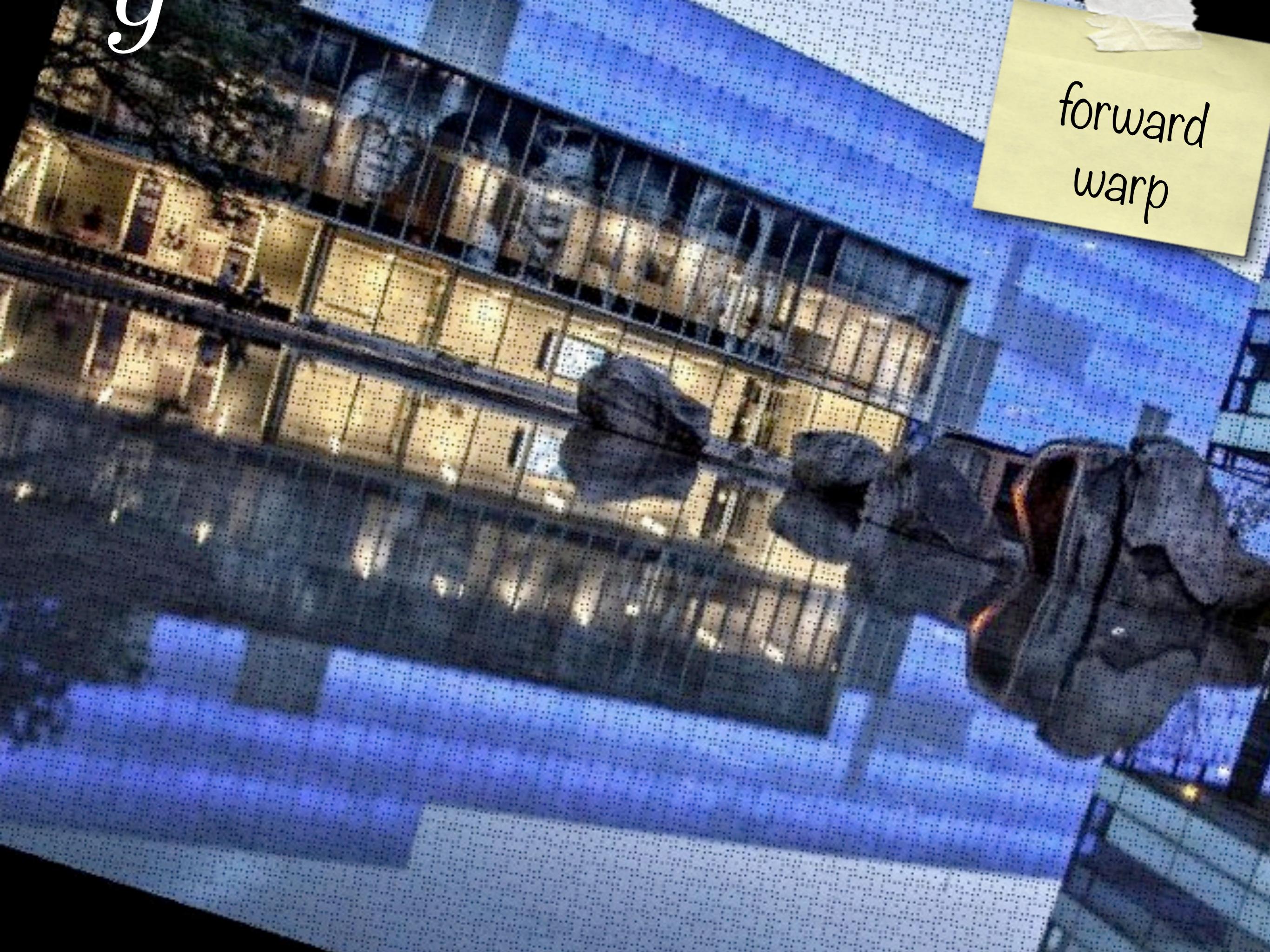


copy pixel value from input to mapped output

forward
warp

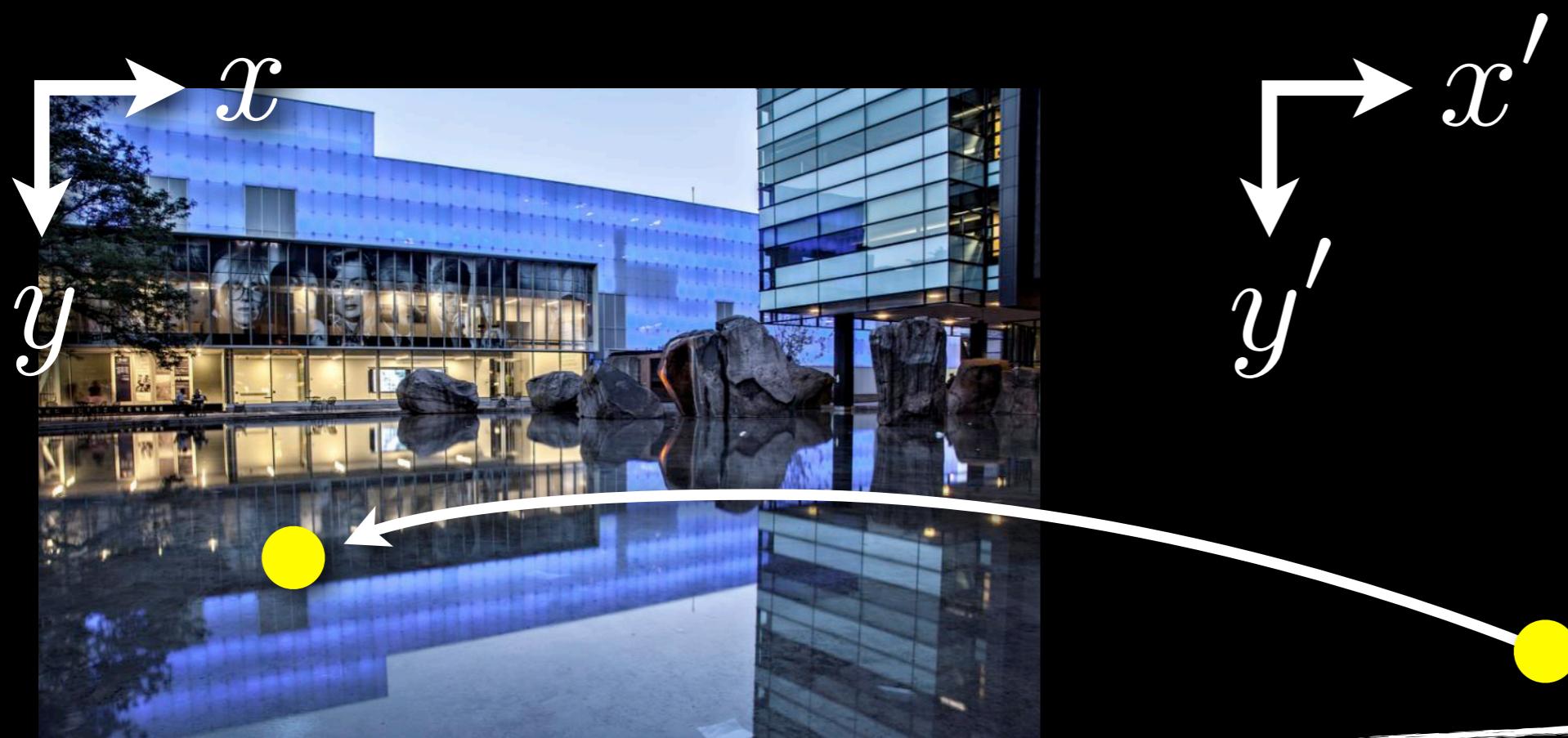


forward
warp



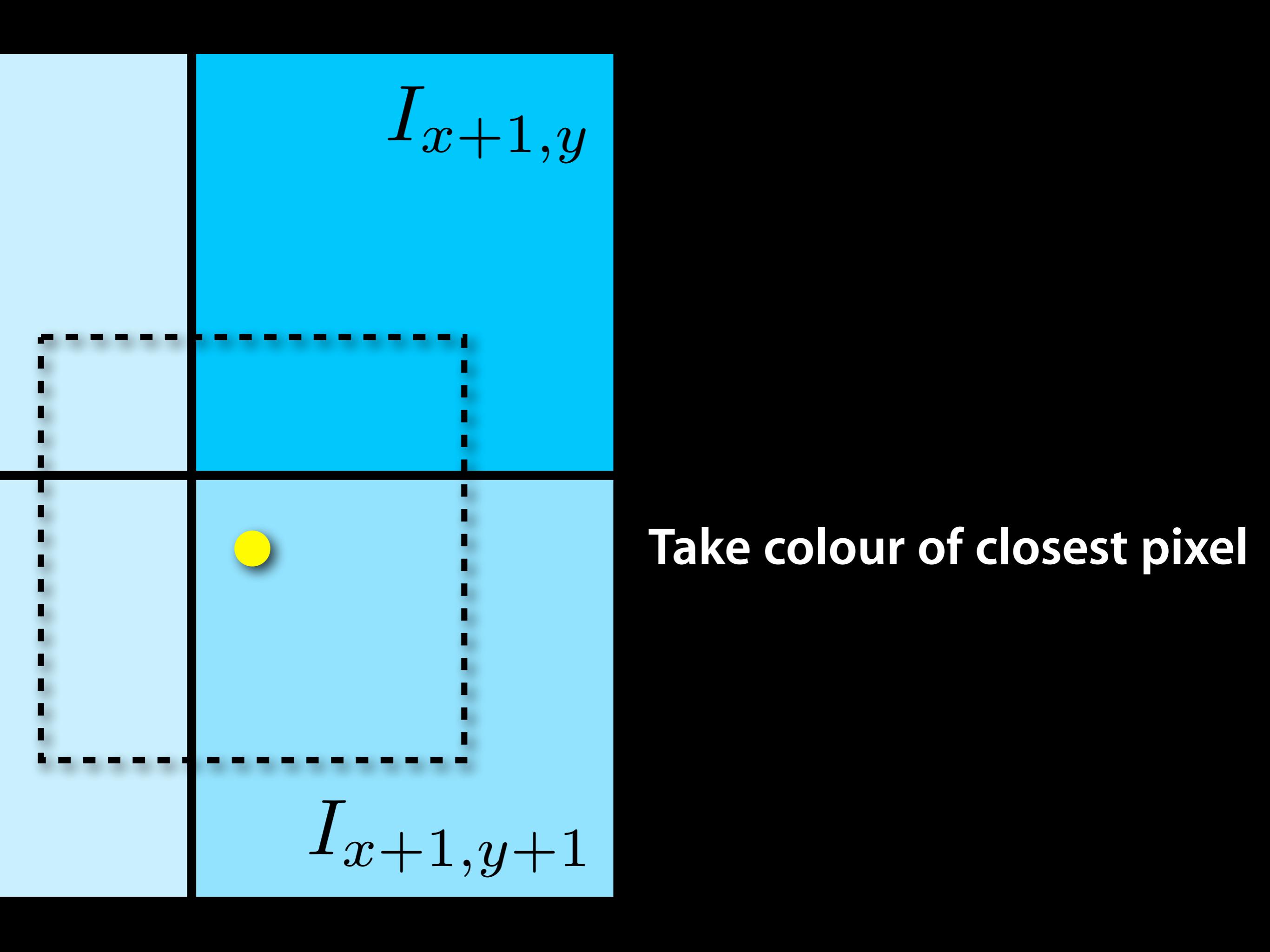
inverse
warp

$$\mathbf{x} = \mathbf{T}^{-1} \mathbf{x}'$$



for each output pixel, map to input and “copy” value

$I_{x,y}$ $I_{x+1,y}$ $I_{x,y+1}$ $I_{x+1,y+1}$ 

 $I_{x+1,y}$  $I_{x+1,y+1}$

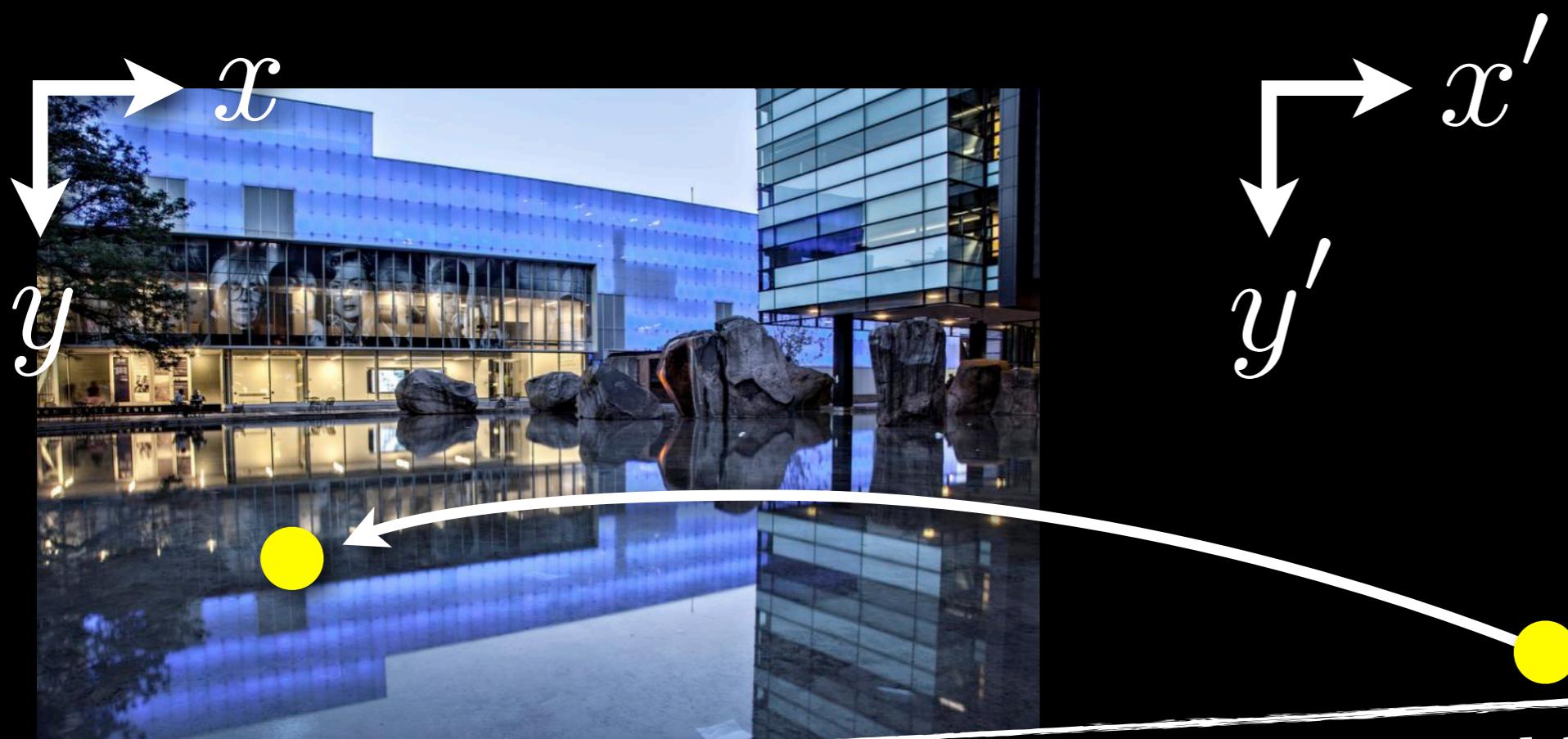
Take colour of closest pixel

$I_{x+1,y}$  $I_{x+1,y+1}$

**Take weighted average of
colours of closest pixels**

inverse
warp

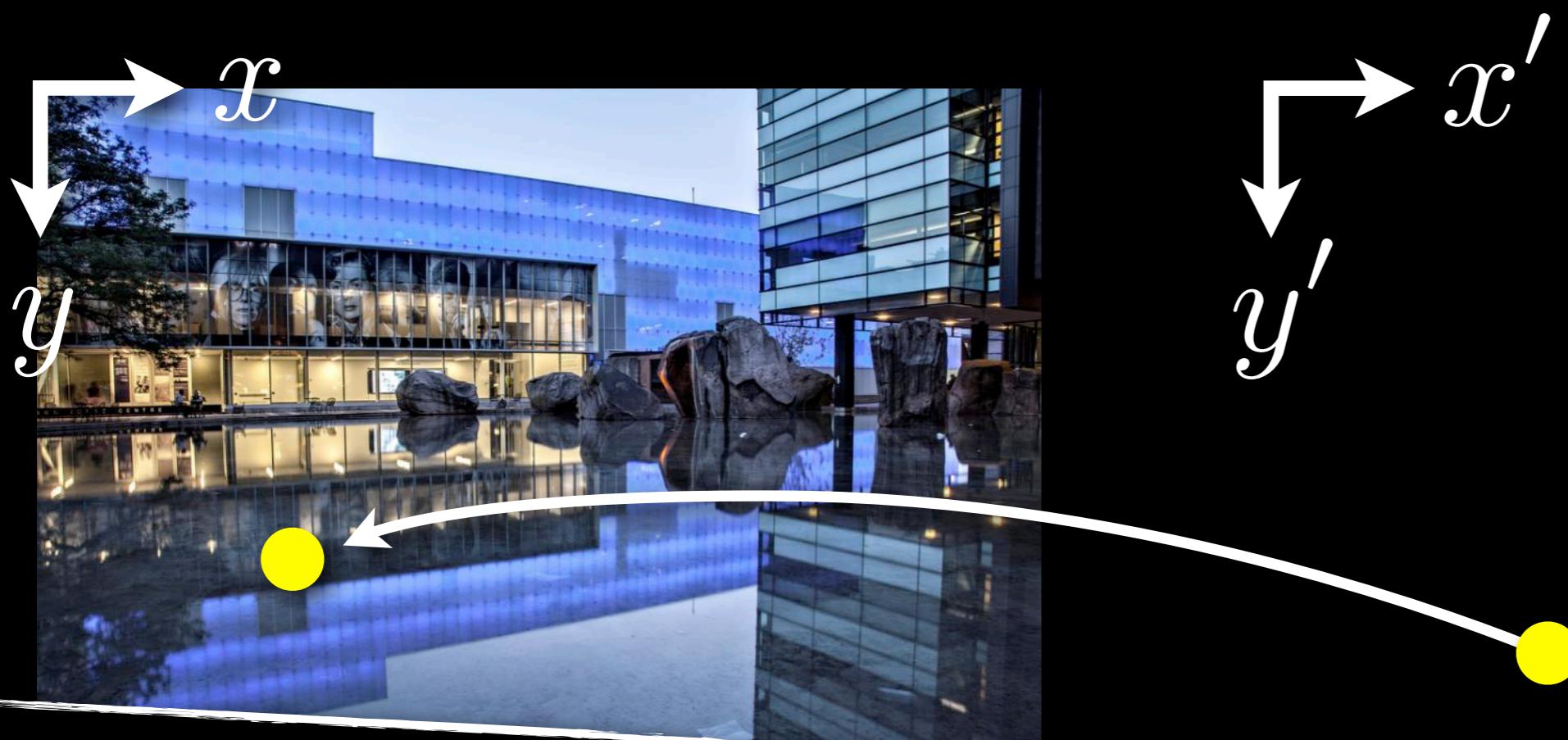
$$\mathbf{x} = \mathbf{T}^{-1} \mathbf{x}'$$



for each output pixel, map to input and “copy” value

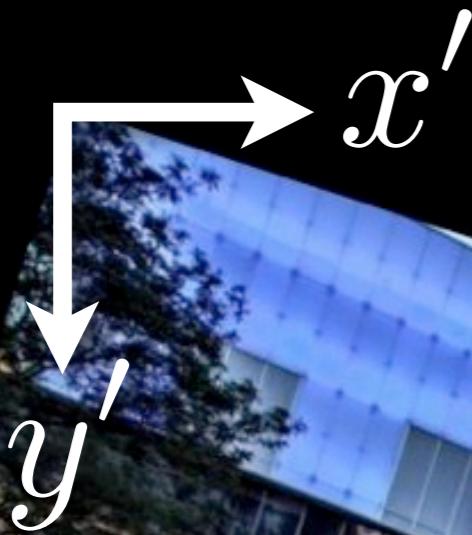
inverse
warp

$$\mathbf{x} = \mathbf{T}^{-1} \mathbf{x}'$$

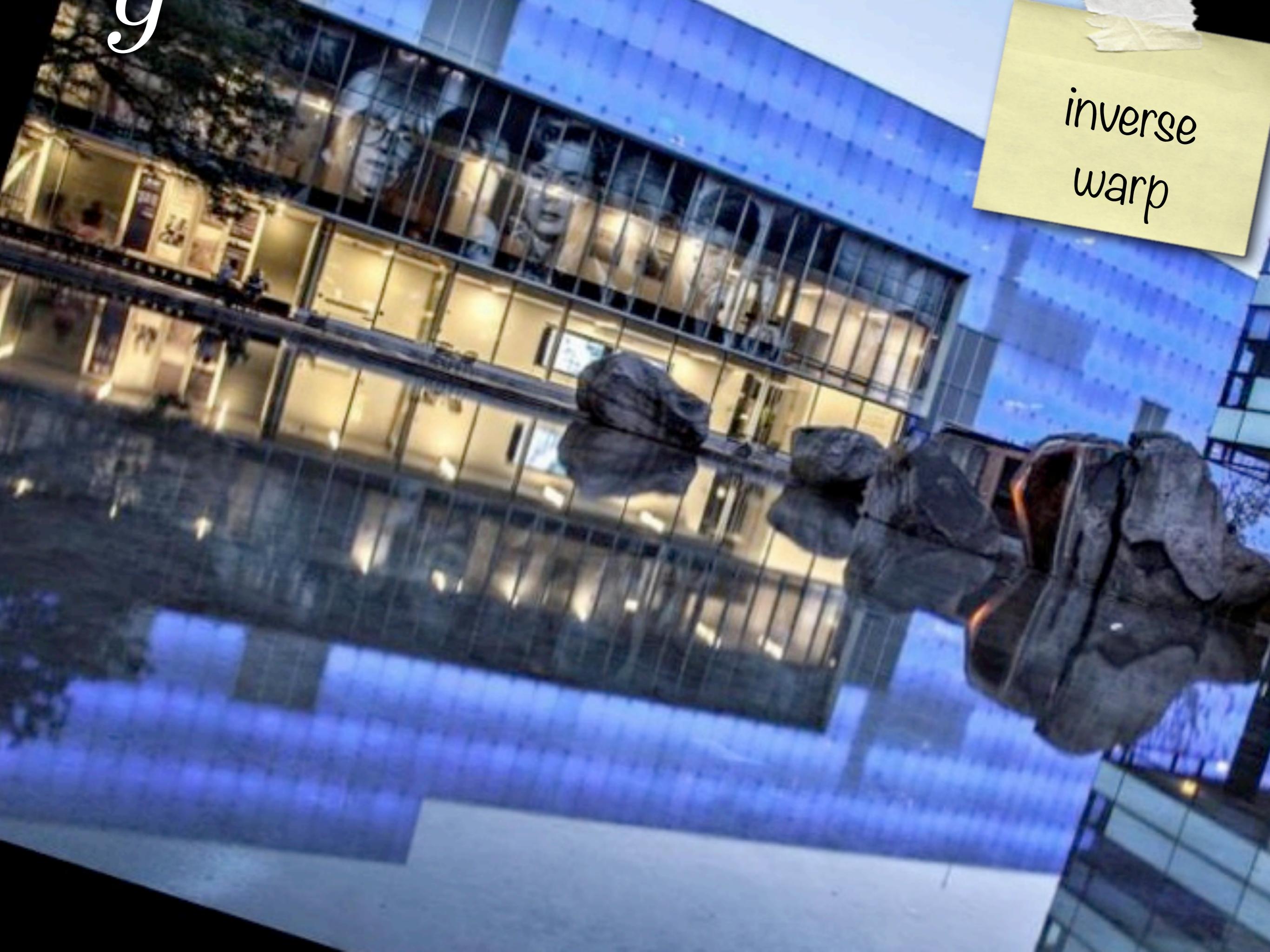


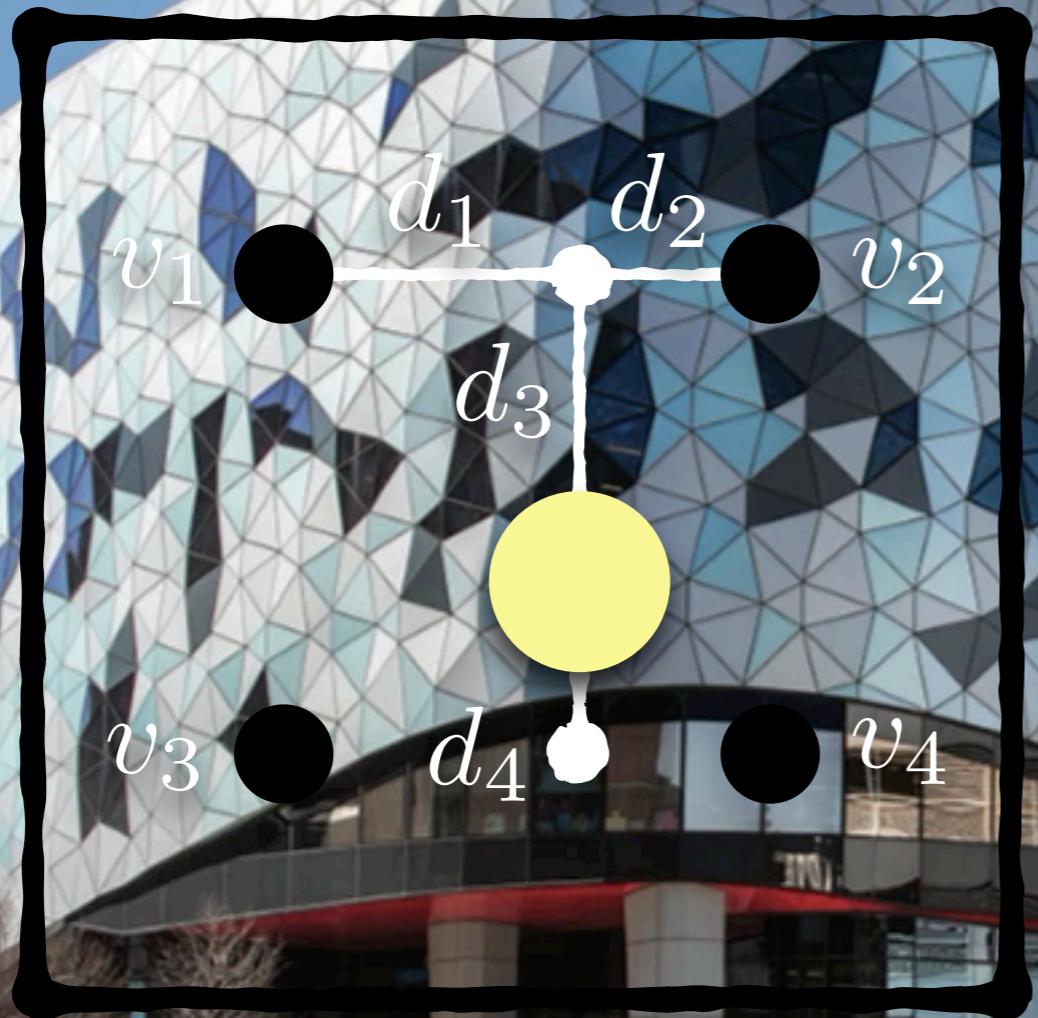
assumes transformation is invertible

inverse
warp



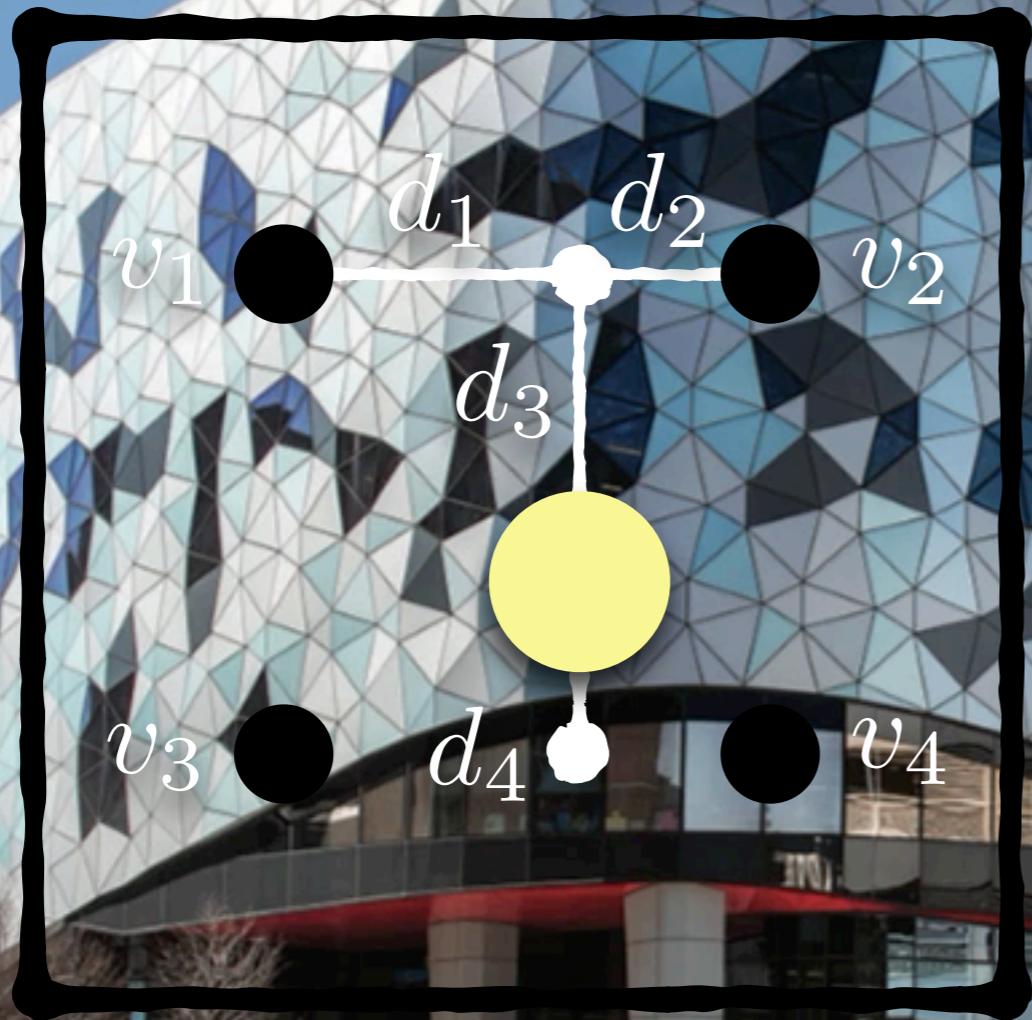
inverse
warp





$$q_1 = v_1 d_2 + v_2 d_1$$

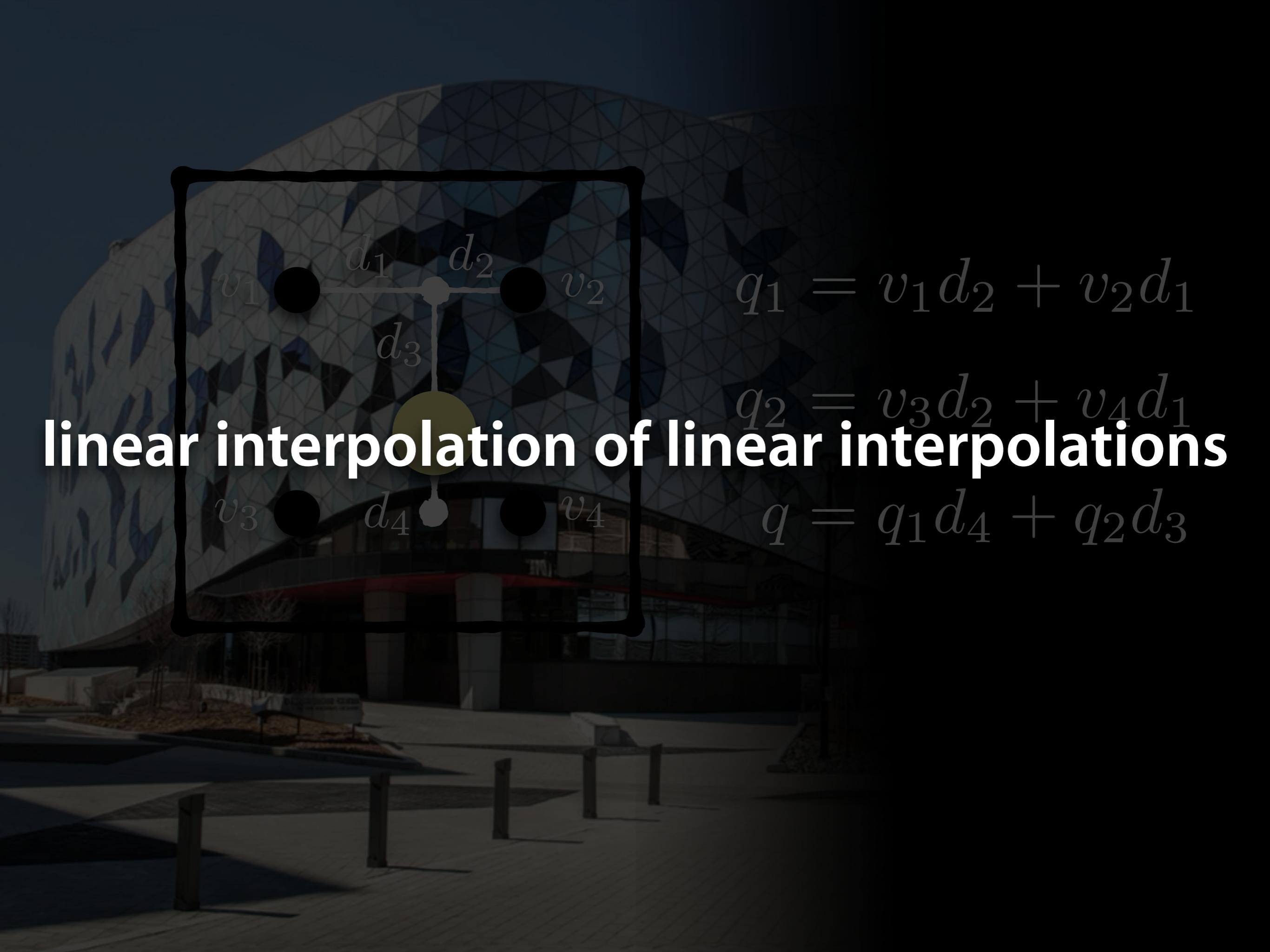
$$q_2 = v_3 d_2 + v_4 d_1$$



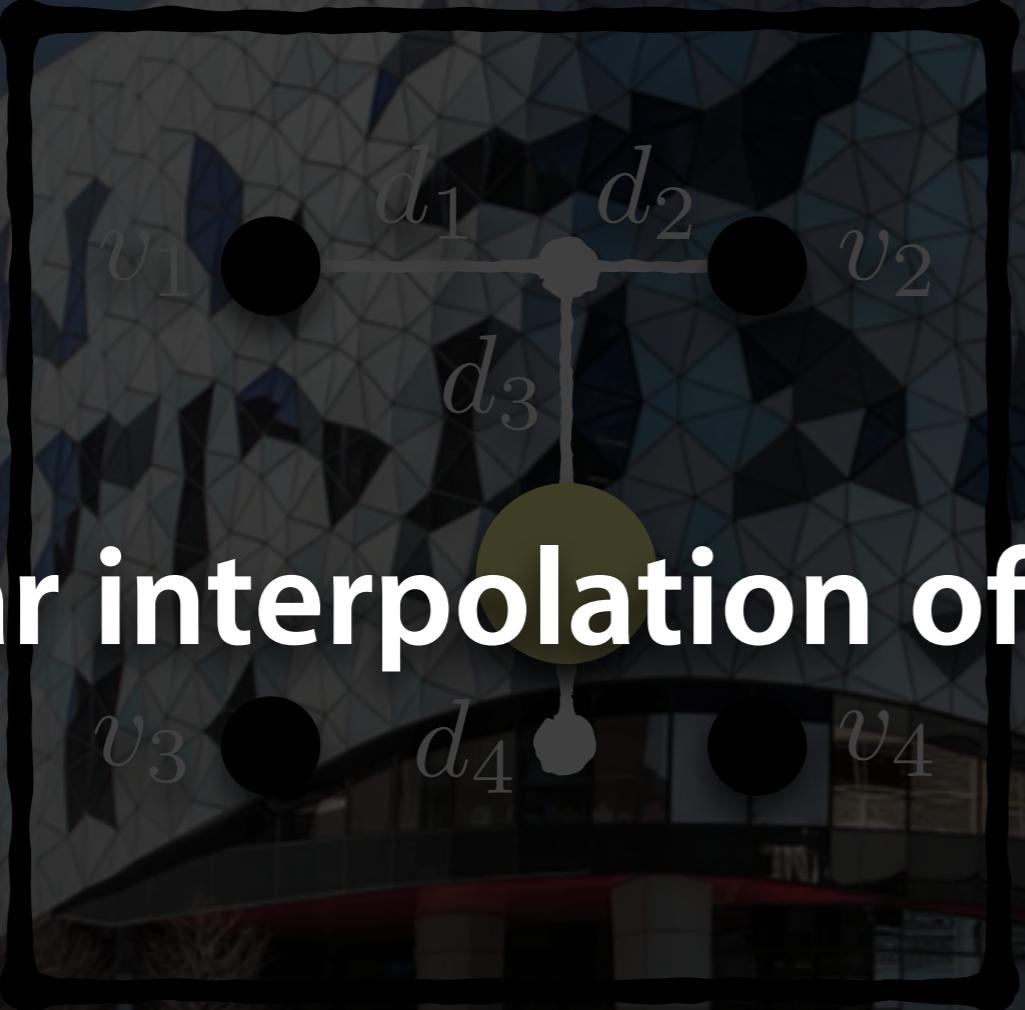
$$q_1 = v_1 d_2 + v_2 d_1$$

$$q_2 = v_3 d_2 + v_4 d_1$$

$$q = q_1 d_4 + q_2 d_3$$



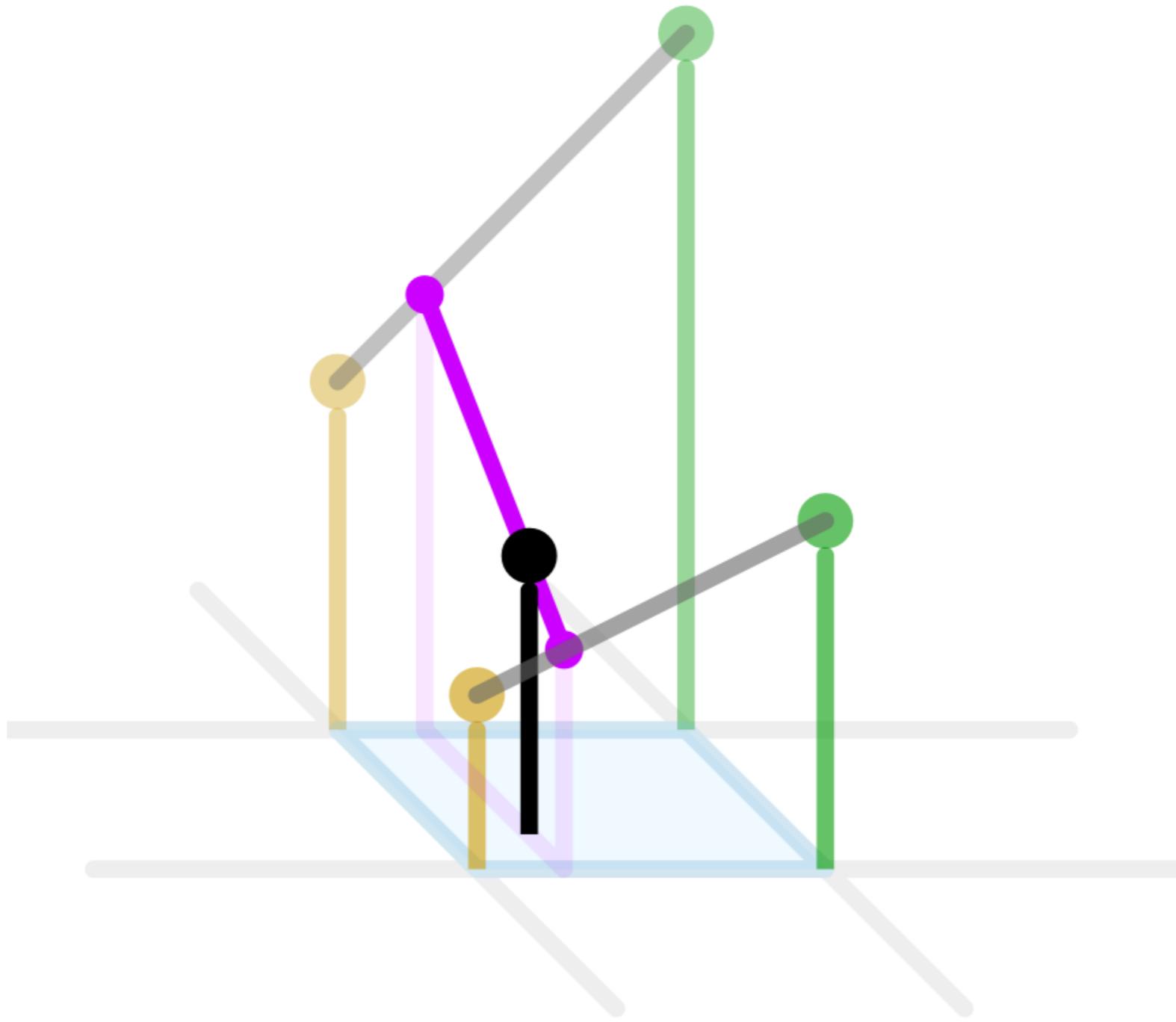
linear interpolation of linear interpolations



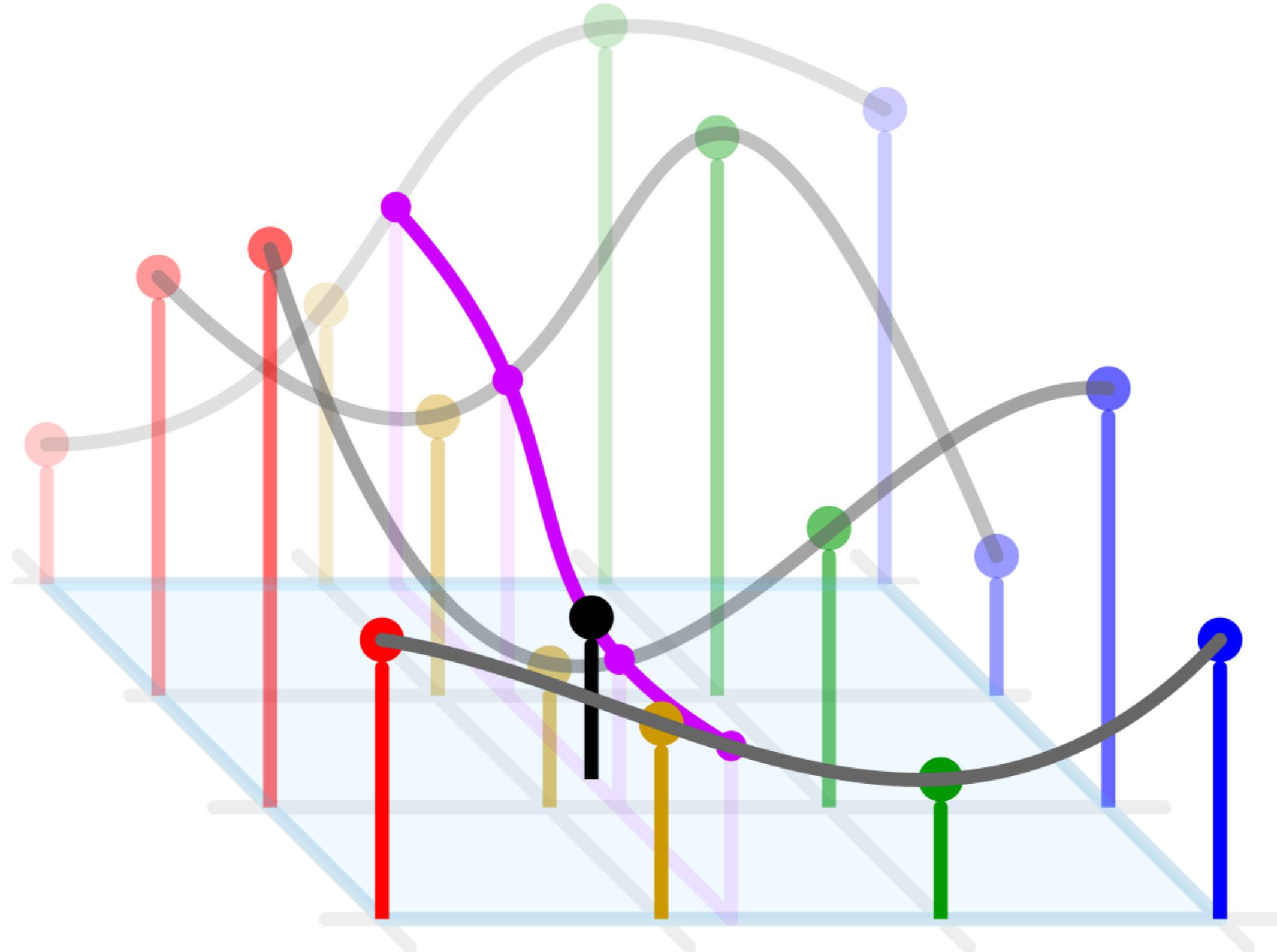
$$q_1 = v_1 d_2 + v_2 d_1$$

$$q_2 = v_3 d_2 + v_4 d_1$$

$$q = q_1 d_4 + q_2 d_3$$



bilinear interpolation



bicubic interpolation

```
img = imread('ghost-building-original.jpg');

imshow(img)
[X Y] = ginput(4);

X2 = [X(1:2); X(1:2)];
Y2 = [Y(1); Y(1); Y(3) + 50; Y(3) + 50];

tform = maketform('projective', [X Y], [X2 Y2]);
img_out = imtransform(img, tform, 'bicubic');

imshow(img_out)
```



```
imshow( img )
[ X Y ] = ginput( 4 );
```

```
img = imread('ghost-building-original.jpg');

imshow(img)
[X Y] = ginput(4);

X2 = [X(1:2); X(1:2)];
Y2 = [Y(1); Y(1); Y(3) + 50; Y(3) + 50];
```

set the position of selected points in the output image

```
img_out = imtransform(img, CTRM,
```

```
imshow(img_out)
```

```
img = imread('ghost-building-original.jpg');

imshow(img)
[X Y] = ginput(4);

X2 = [X(1:2); X(1:2)];
Y2 = [Y(1); Y(1); Y(3) + 50; Y(3) + 50];

tform = maketform('projective', [X Y], [X2 Y2]);
img_out = imtransform(img, tform, 'bilinear');
```

create homography based on point correspondences

```
img = imread('ghost-building-original.jpg');

imshow(img)
[X Y] = ginput(4);

X2 = [X(1:2); X(1:2)];
Y2 = [Y(1); Y(1); Y(3) + 50; Y(3) + 50];

tform = maketform('projective', [X Y], [X2 Y2]);
img_out = imtransform(img, tform, 'bicubic');
```

warp input image based on transformation

```
img = imread('ghost-building-original.jpg');

imshow(img)
[X Y] = ginput(4);

X2 = [X(1:2); X(1:2)];
Y2 = [Y(1); Y(1); Y(3) + 50; Y(3) + 50];

tform = maketform('projective', [X Y], [X2 Y2]);
img_out = imtransform(img, tform, 'bicubic');

imshow(img_out)
```



planar
rectification