The Ancient Secrets of Computer Vision

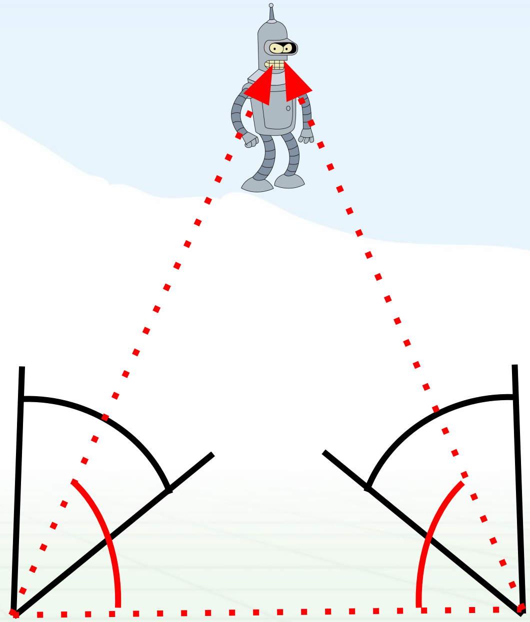
Topics:

3D, Depth Perception, & Stereo

How do we perceive depth?

How do we perceive depth?

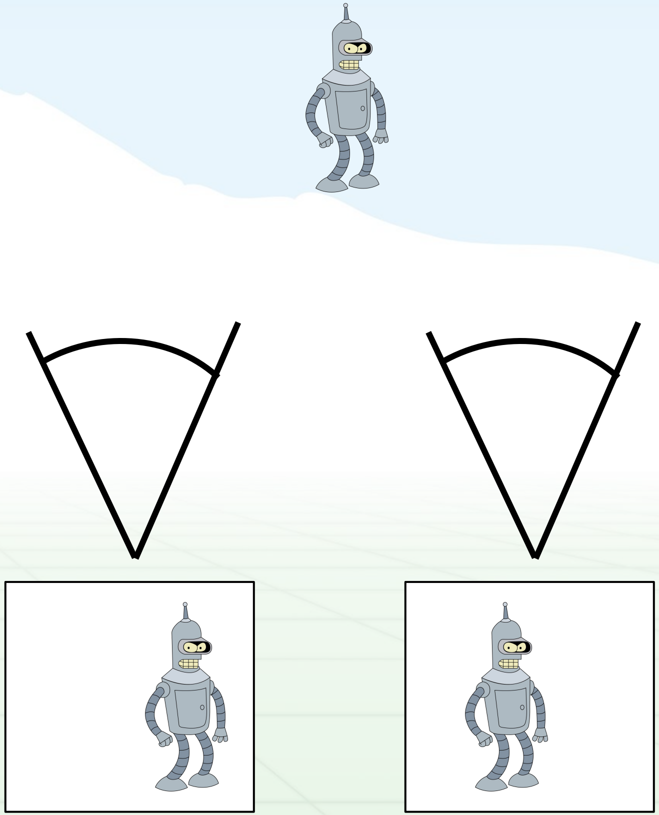
-Binocular Convergence



How do we perceive depth?

-Binocular Convergence

-Binocular Parallax

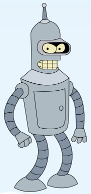


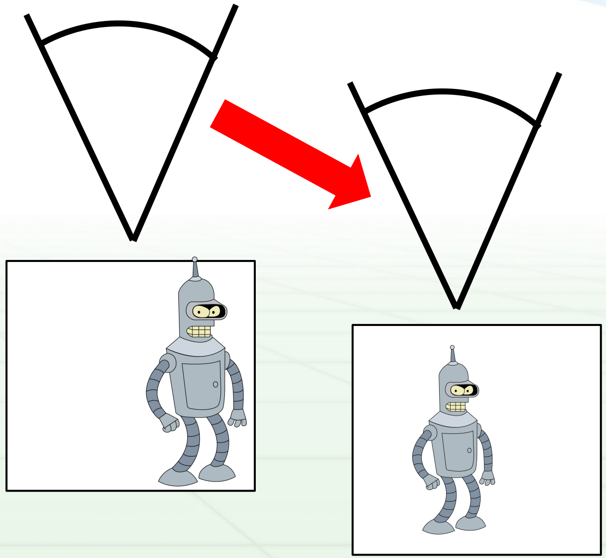
How do we perceive depth?

-Binocular Convergence

-Binocular Parallax

-Monocular Movement Parallax





How do we perceive depth?

-Binocular Convergence

-Binocular Parallax

-Monocular Movement Parallax

-Image Cues

How do we perceive depth?

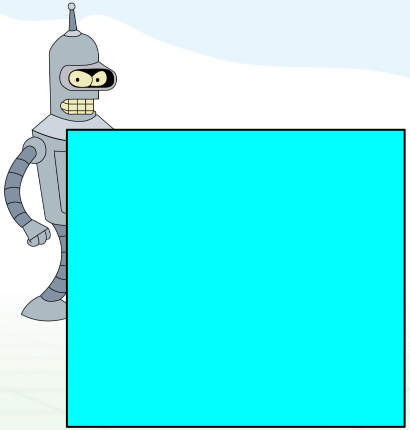
-Binocular Convergence

-Binocular Parallax

-Monocular Movement Parallax

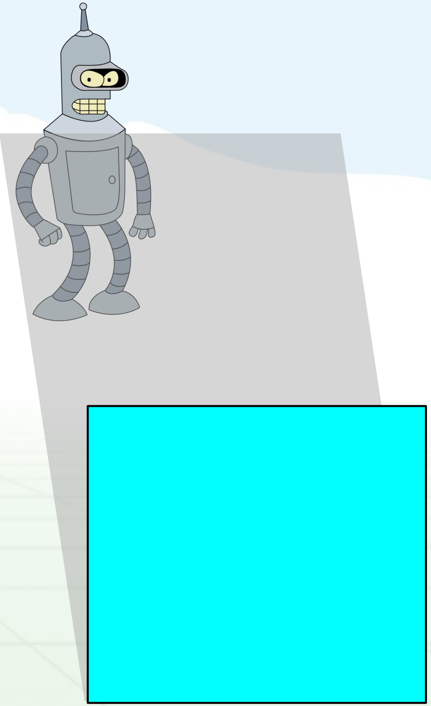
-Image Cues

   -Overlap

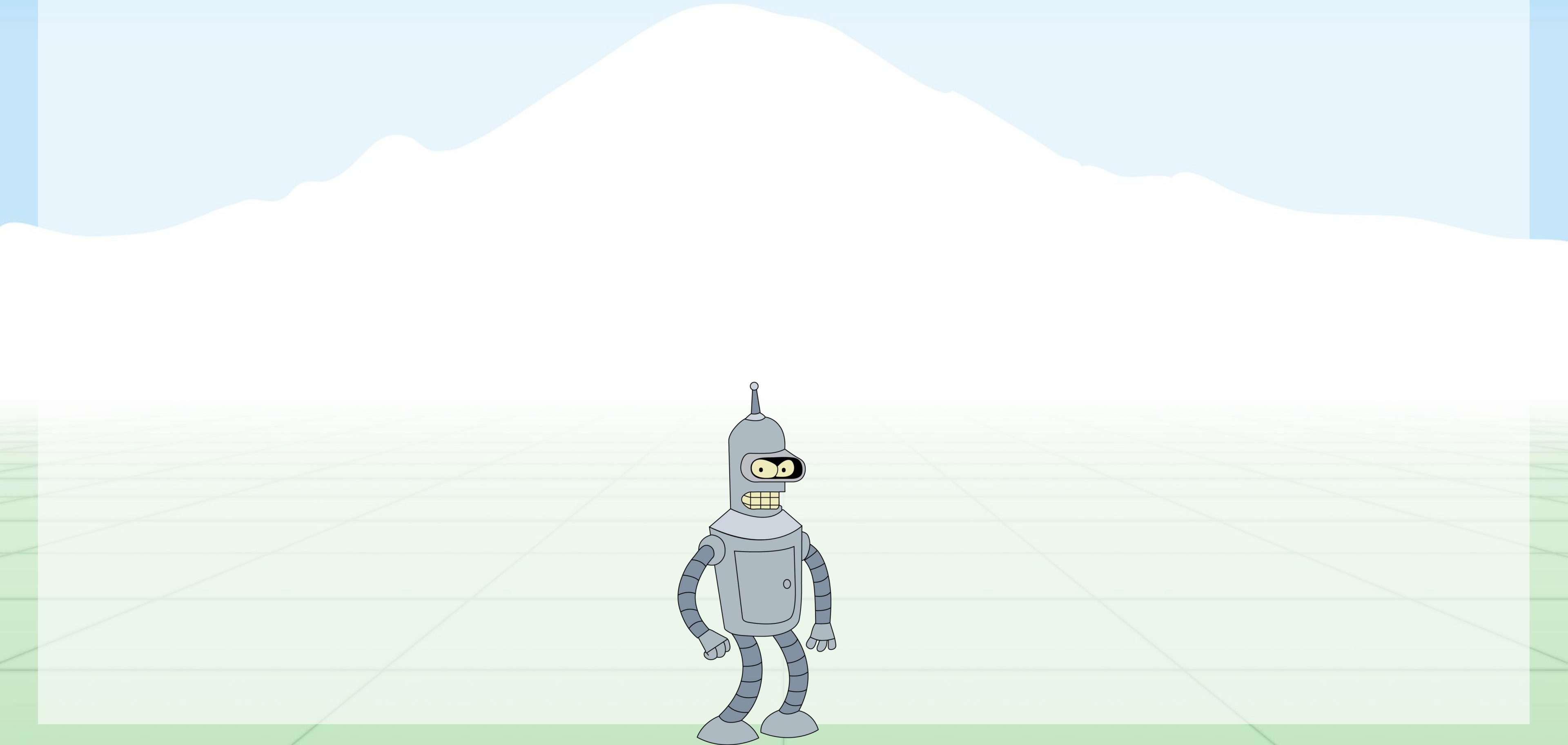


How do we perceive depth?

- Binocular Convergence
- Binocular Parallax
- Monocular Movement Parallax
- Image Cues
    - Overlap
    - Shadows

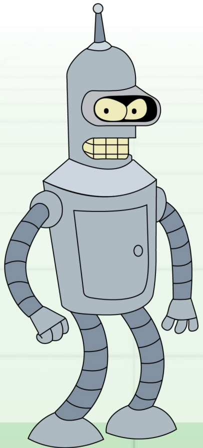


Interactive Demo



Interactive Demo

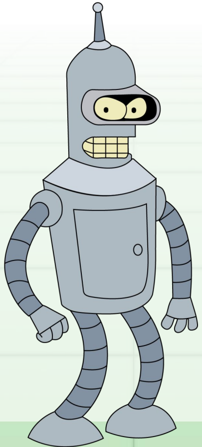
Close one eye



Interactive Demo

Close one eye

Cover Bender with your thumb

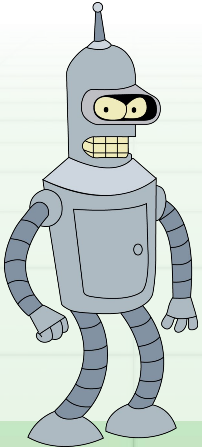


Interactive Demo

Close one eye

Cover Bender with your thumb

Open your other eye and close the first



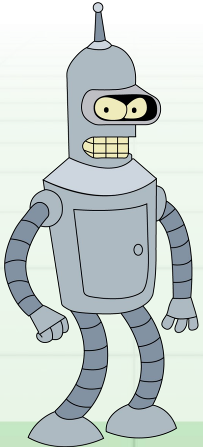
Interactive Demo

Close one eye

Cover Bender with your thumb

Open your other eye and close the first

Bender moved!



How do computers perceive depth?

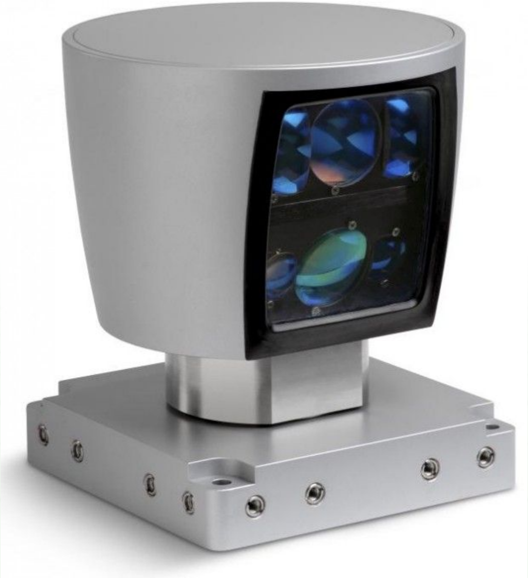
How do computers perceive depth?

-Stereo

How do computers perceive depth?

-Stereo

-Lidar



How do computers perceive depth?

- Stereo
- Lidar
- Structured light

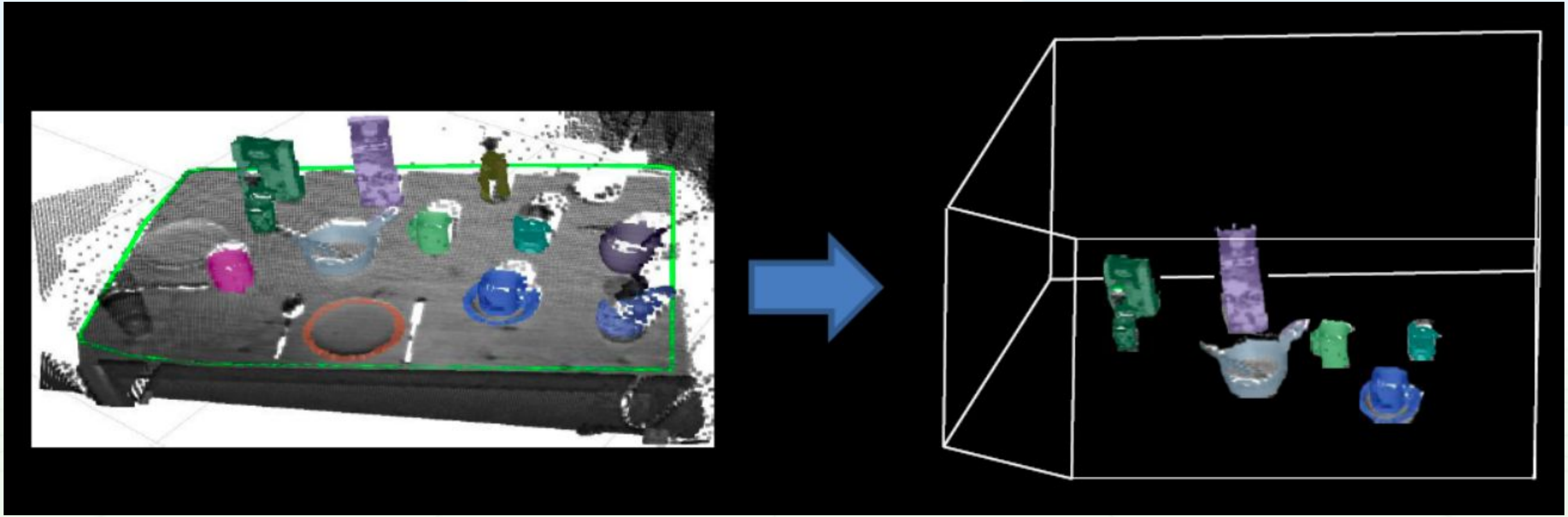
How do computers perceive depth?

- Stereo
- Lidar
- Structured light
- Time of flight

Why do we want depth?

Why do we want depth?

-Object segmentation

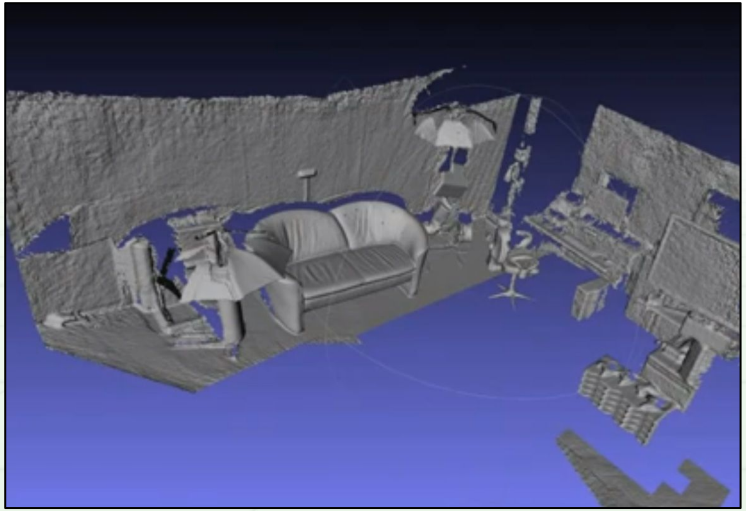


Why do we want depth?

-Object segmentation

-3D reconstruction

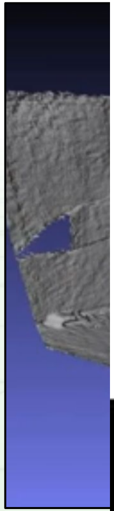
KinectFusion



Why c

-Objec

-3D re

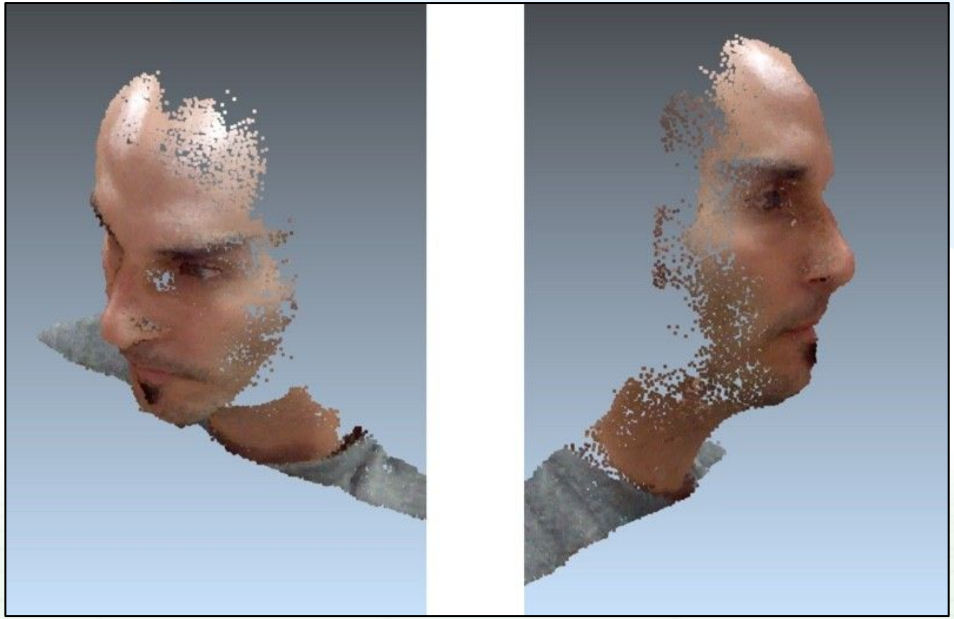


Why do we want depth?

- Object segmentation
- 3D reconstruction
- Navigation

Why do we want depth?

- Object segmentation
- 3D reconstruction
- Navigation
- Facial Recognition



Why do we want depth?

-Object segmentation

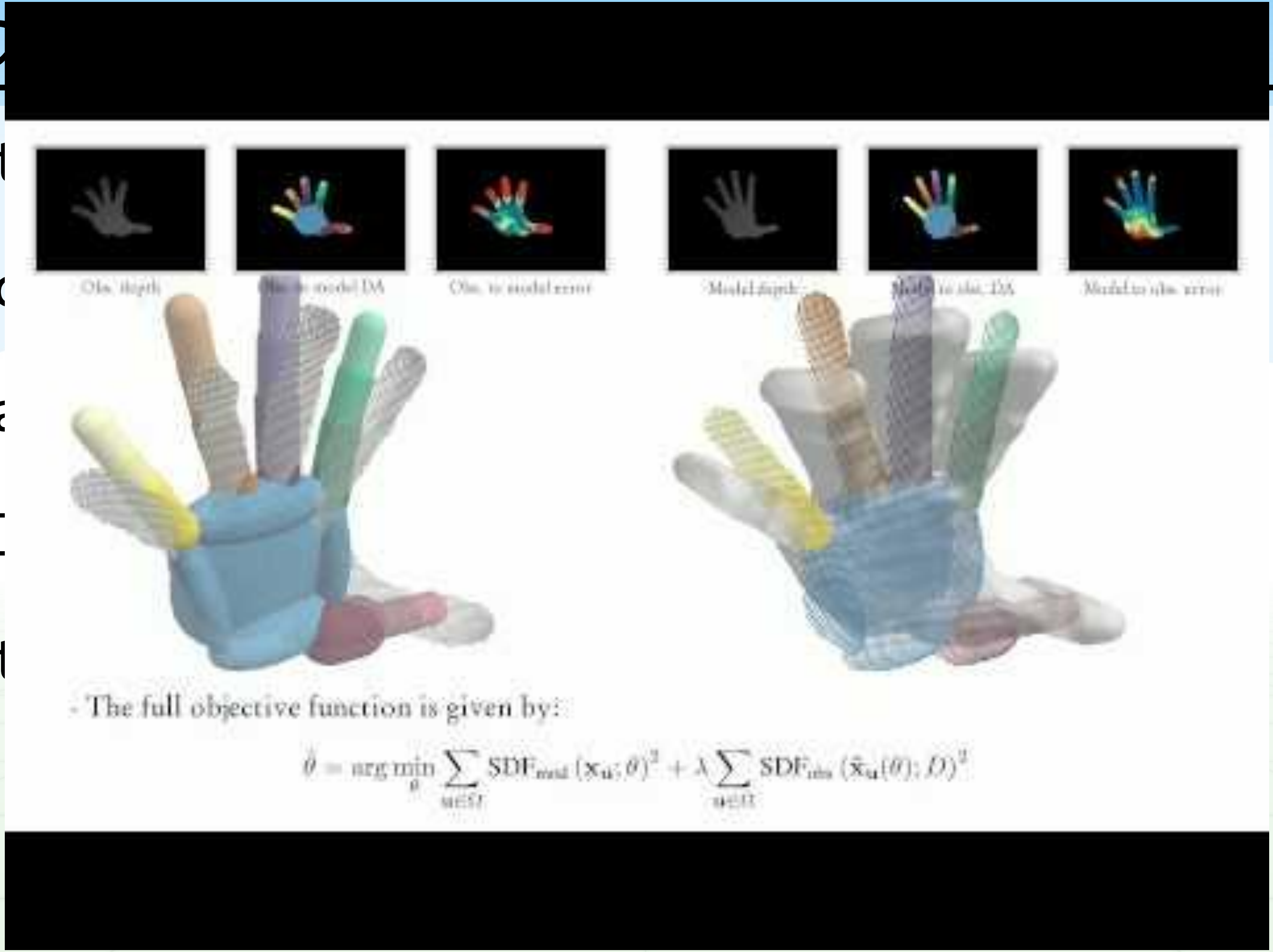
-3D reconstruction

-Navigation

-Facial Recognition

-Pose tracking

Why d



-Object

-3D rec

-Naviga

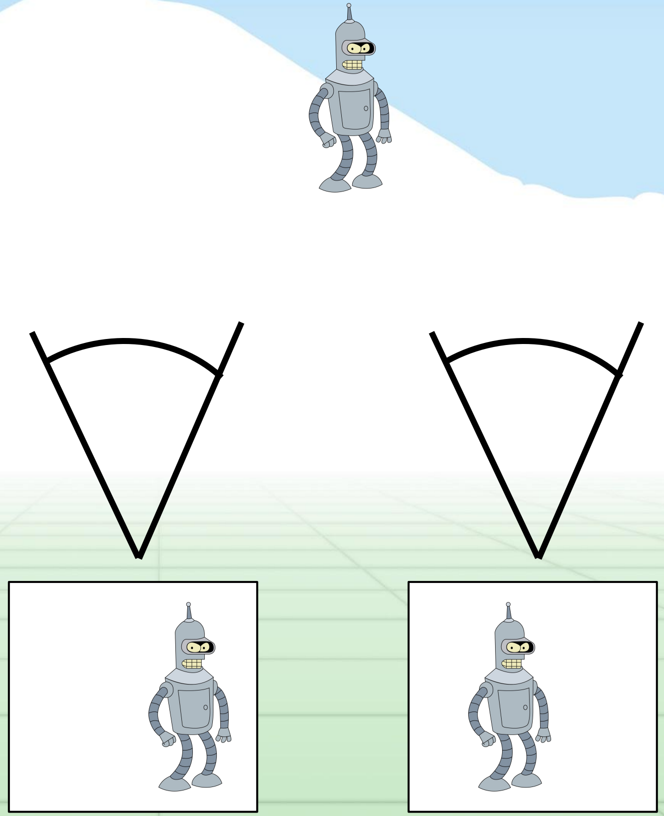
-Facial

-Pose t

Stereo

Stereo

aka Binocular Parallax



Stereo

Stereo

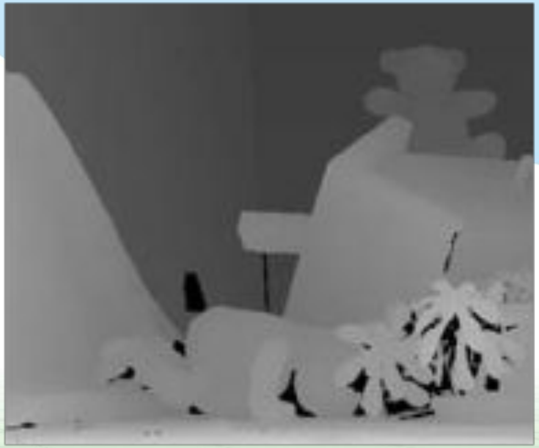
Displacement inversely proportional to distance from the camera.

Stereo

Stereo

How do we align different parts of two images?

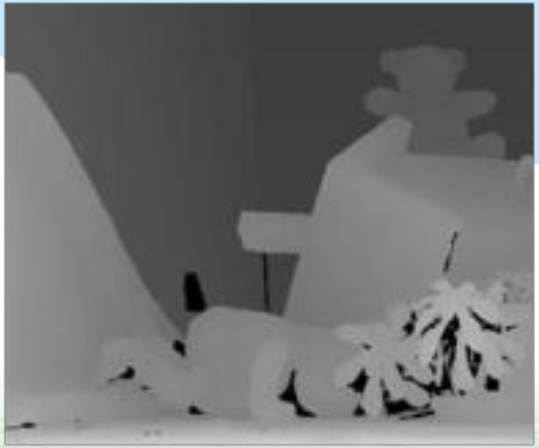




Stereo

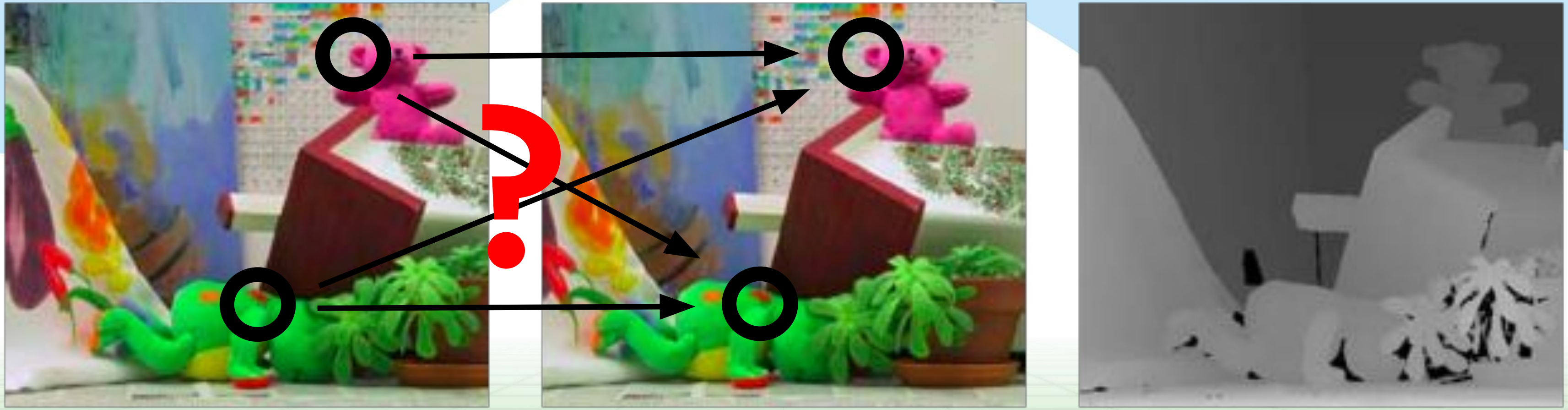
Feature matching!





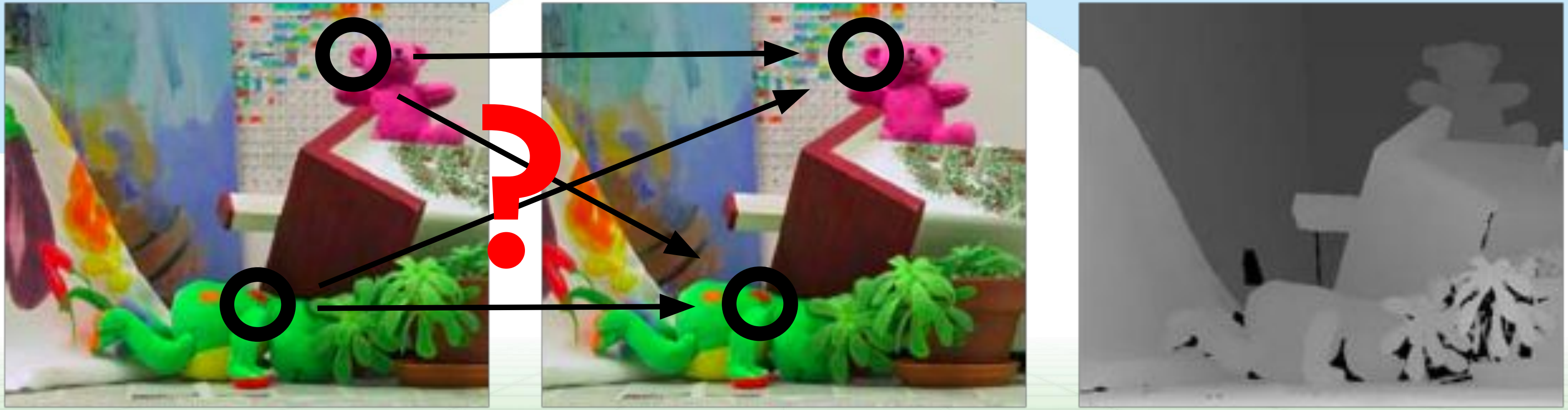
Stereo

How to do the matching? RANSAC? Bipartite Matching? ...



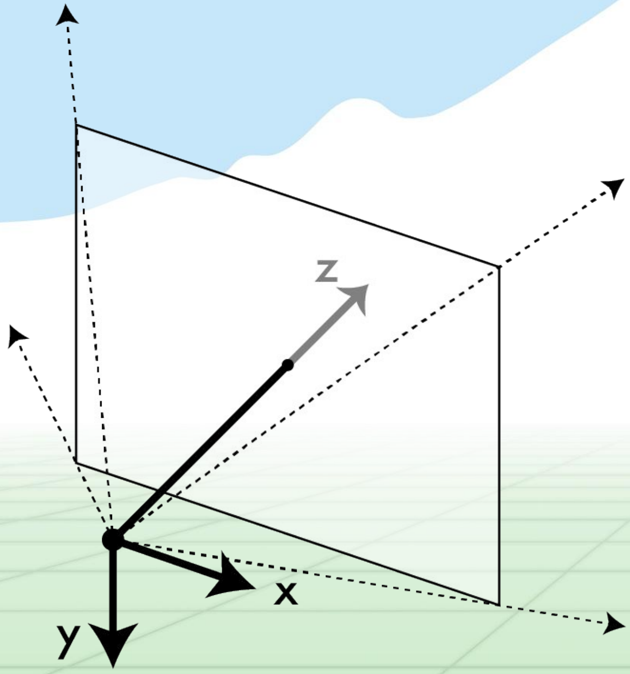
Stereo

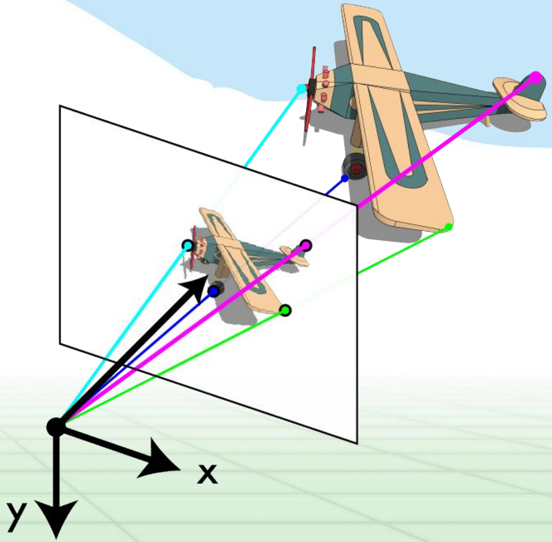
How to do the matching? RANSAC? Bipartite Matching? ...



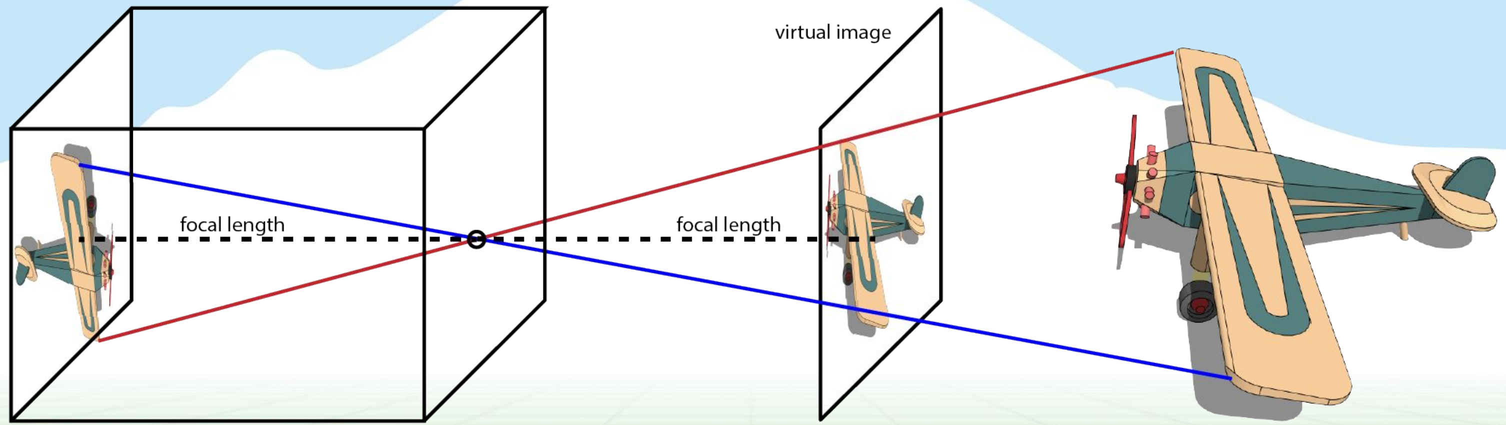
Very difficult, can we take advantage of the structure somehow?

Perspective Projection

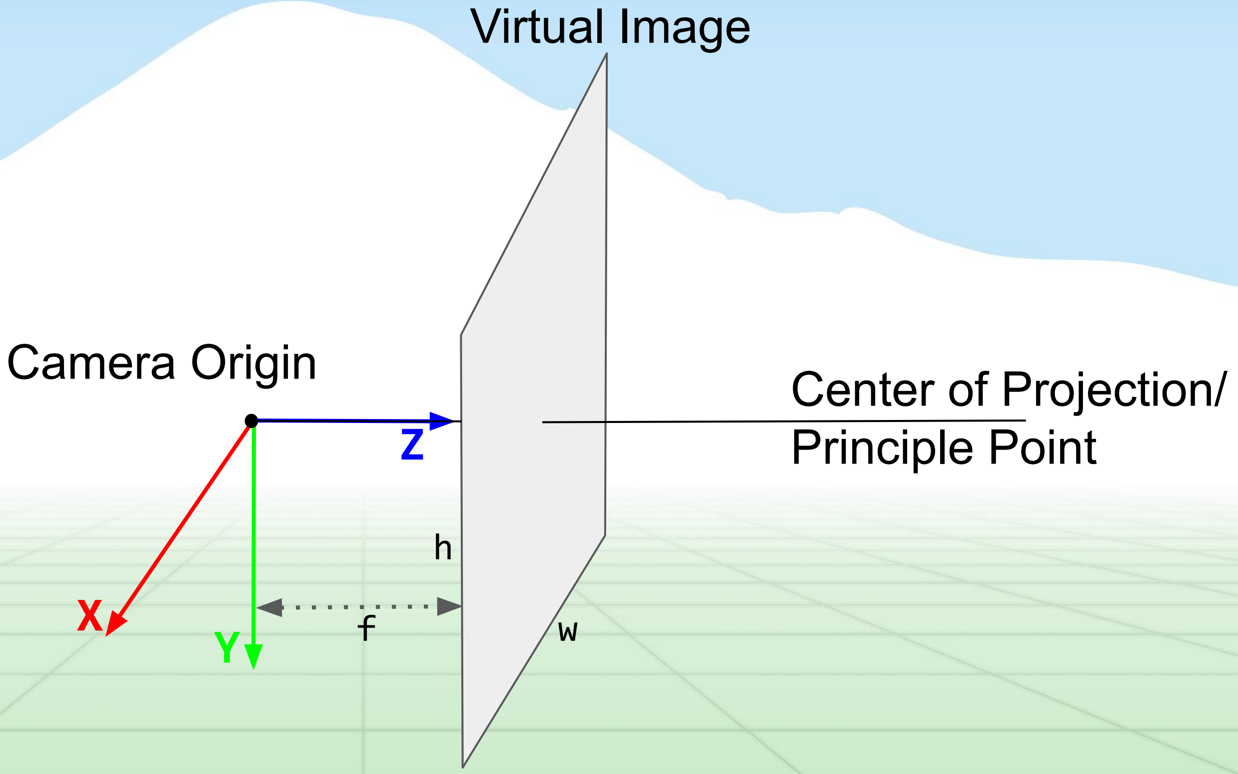




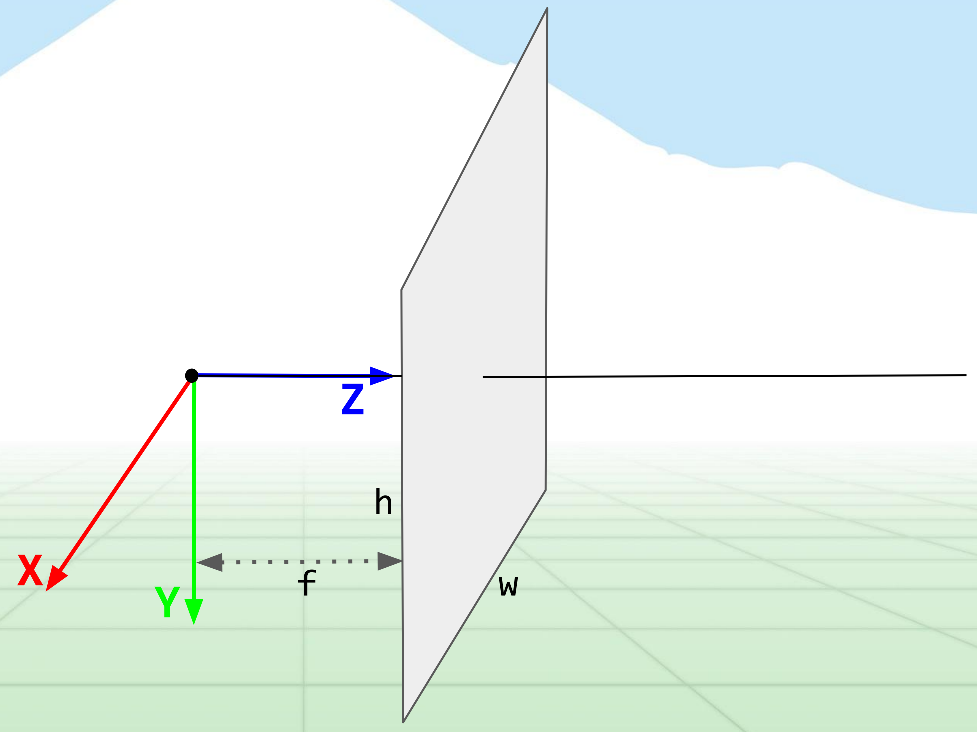
Perspective Projection



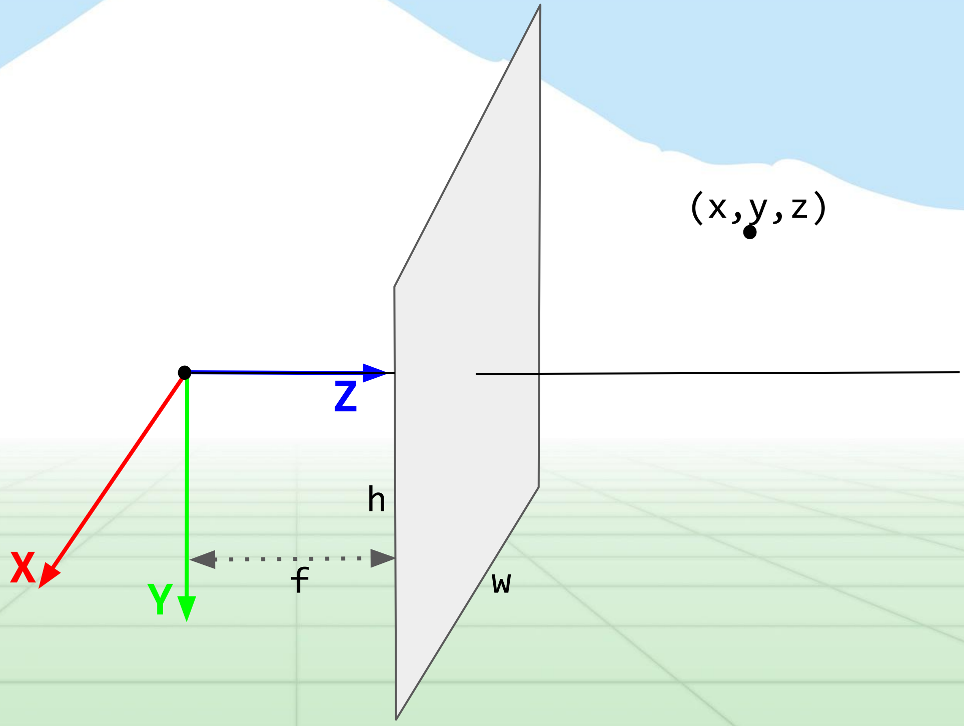
Perspective Projection



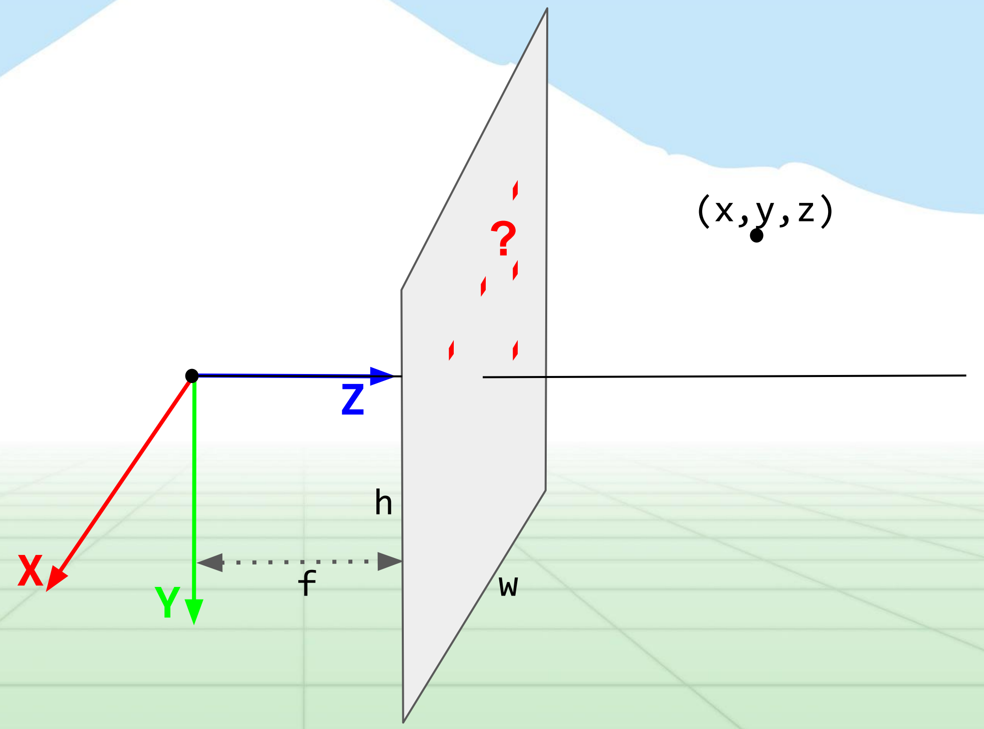
Perspective Projection



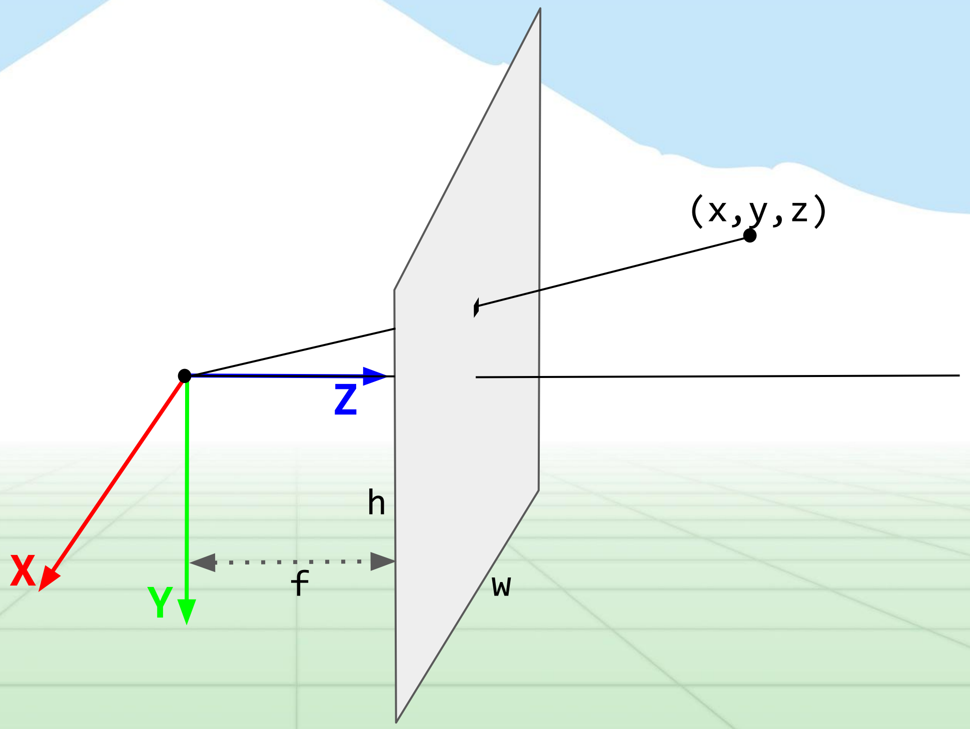
Perspective Projection



Perspective Projection



Perspective Projection

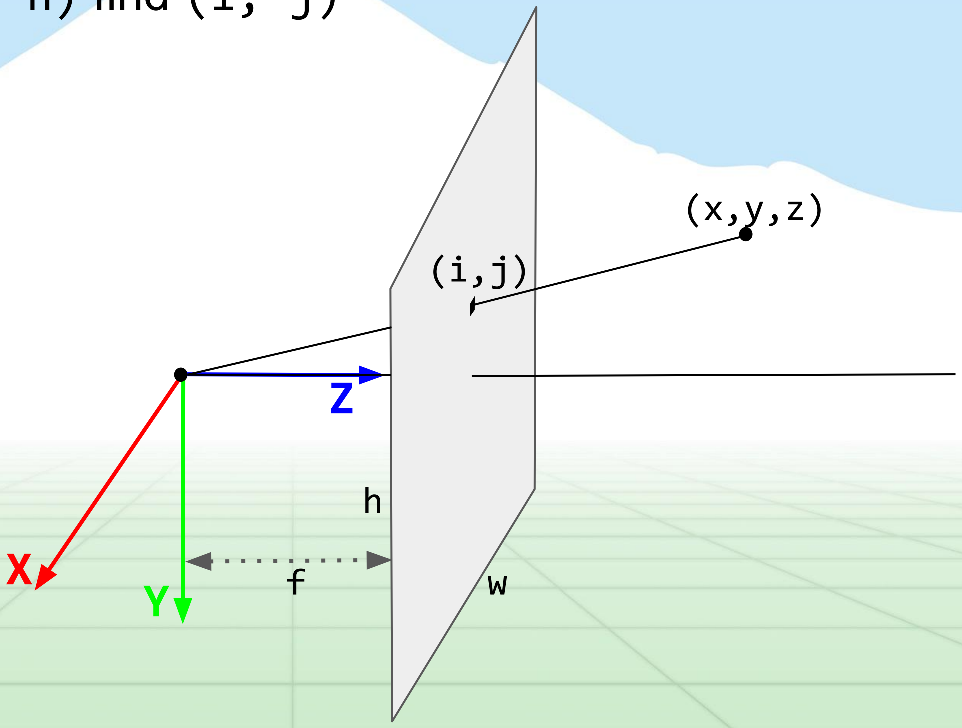


Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

i = ?

j = ?

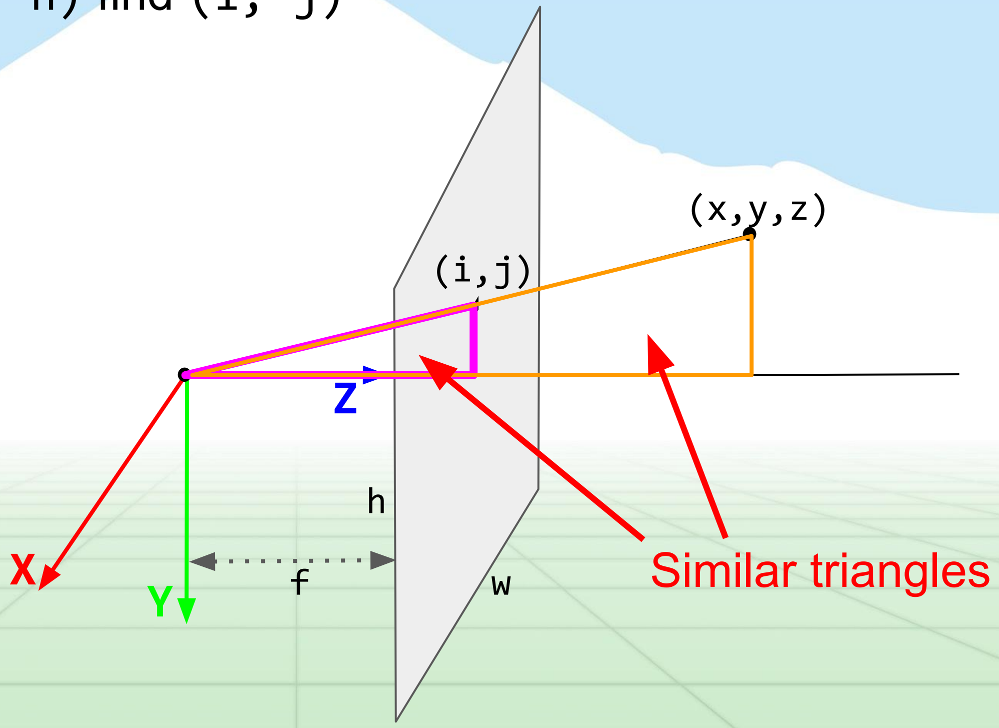


Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

$i = ?$

$j = ?$

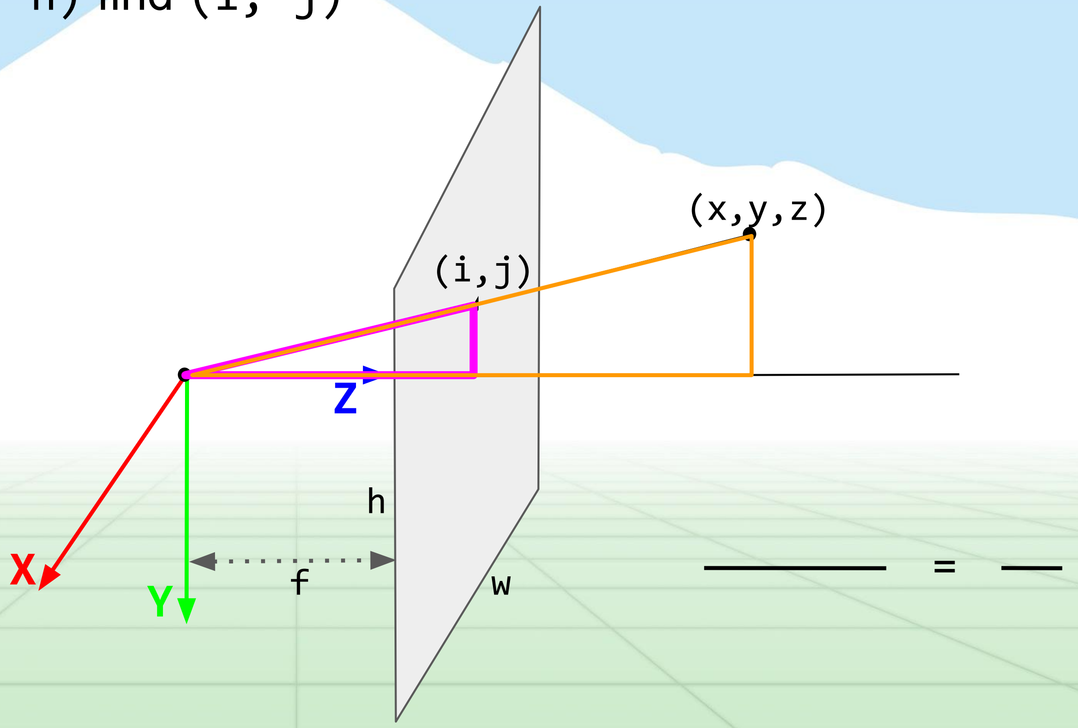


Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

$i = ?$

$j = ?$

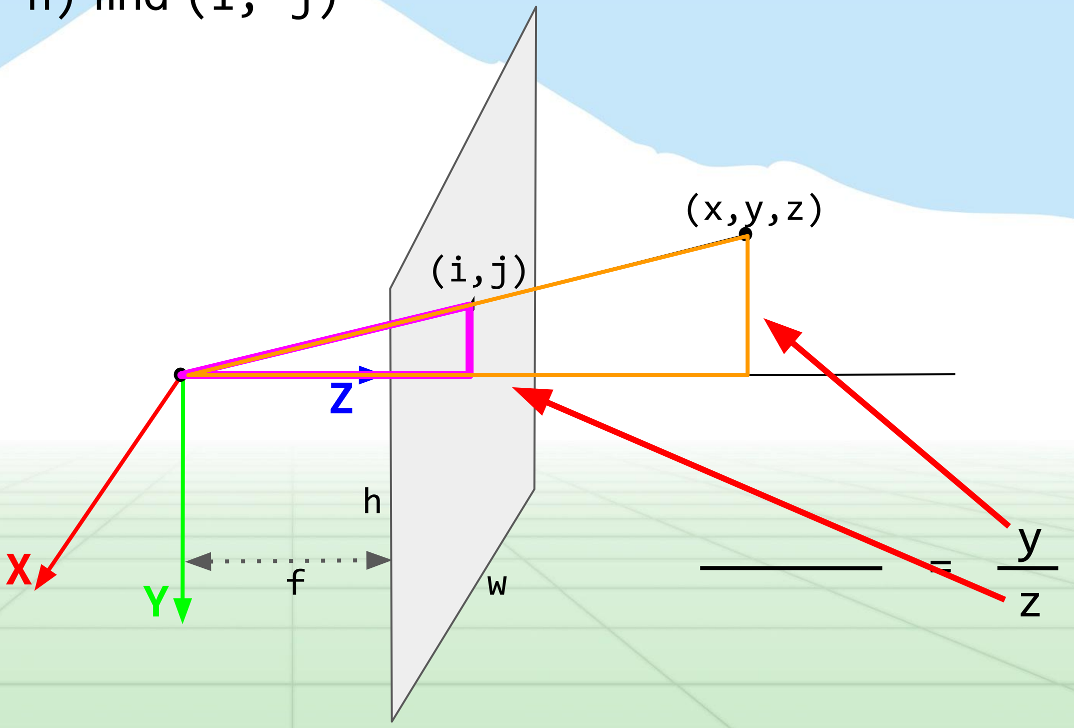


Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

i = ?

j = ?

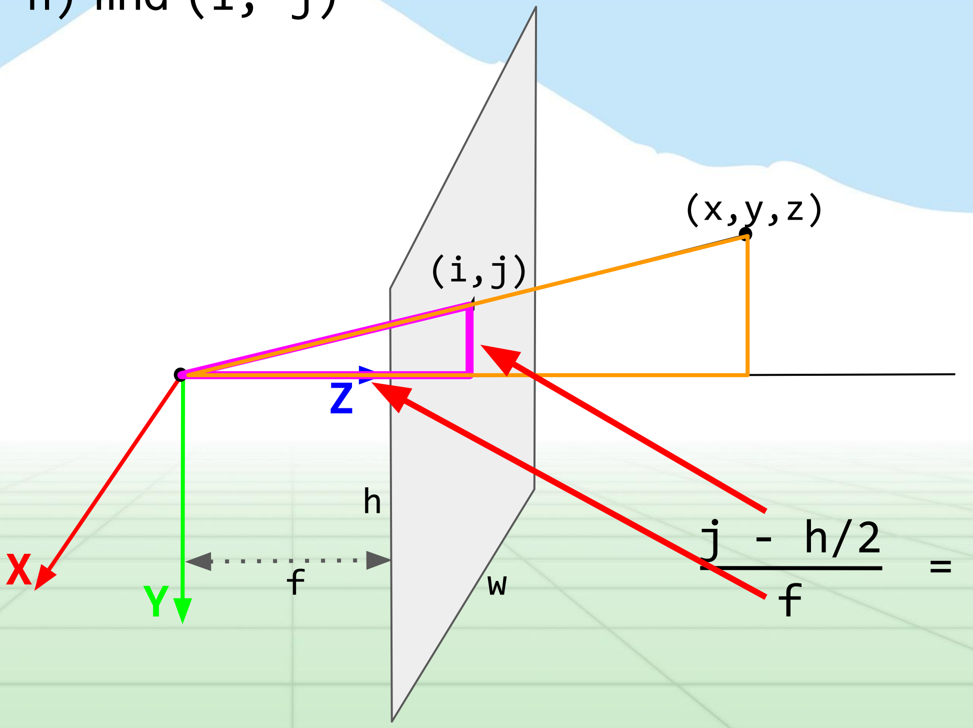


Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

$i = ?$

$j = ?$



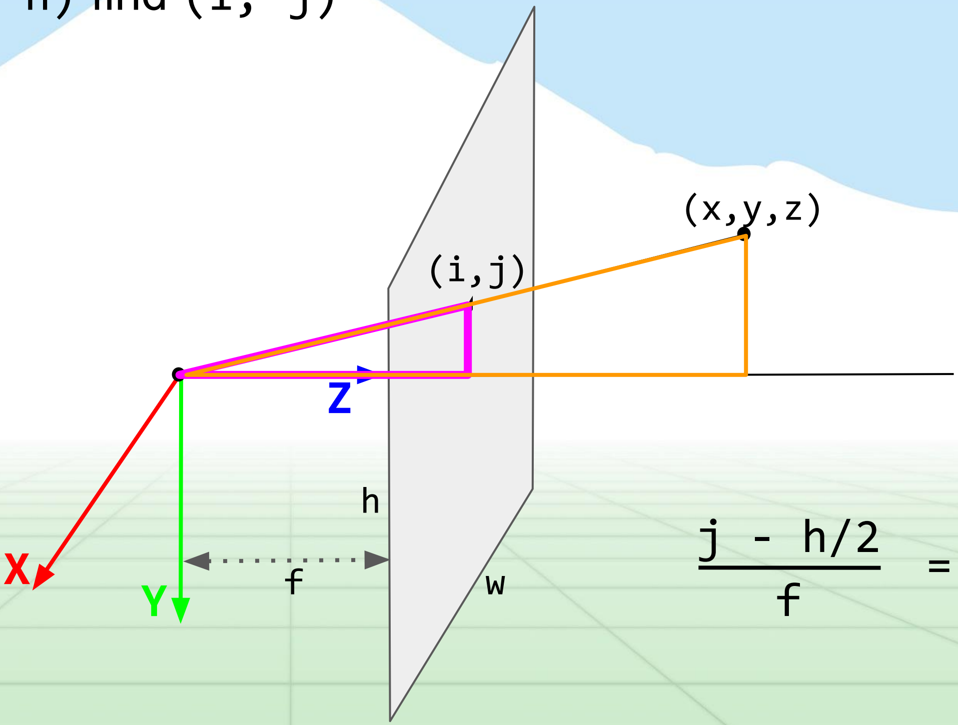
$$\frac{j - h/2}{f} = \frac{y}{z}$$

Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

i = ?

j = ?



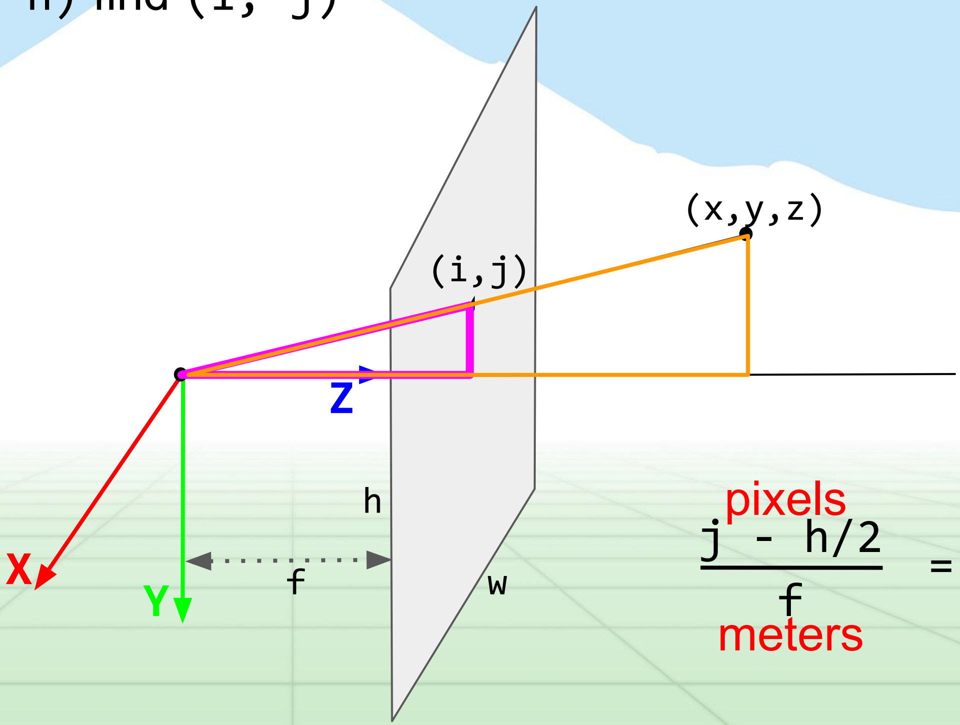
$$\frac{j - h/2}{f} = \frac{y}{z}$$

Perspective Projection

Given (x, y, z) and (f, w, h) find (i, j)

$i = \color{red}{?}$

$j = \color{red}{?}$



$$\frac{\overset{\color{red}{\text{pixels}}}{j - h/2}}{\underset{\color{red}{\text{meters}}}{f}} = \frac{y}{z}$$

Perspective Projection

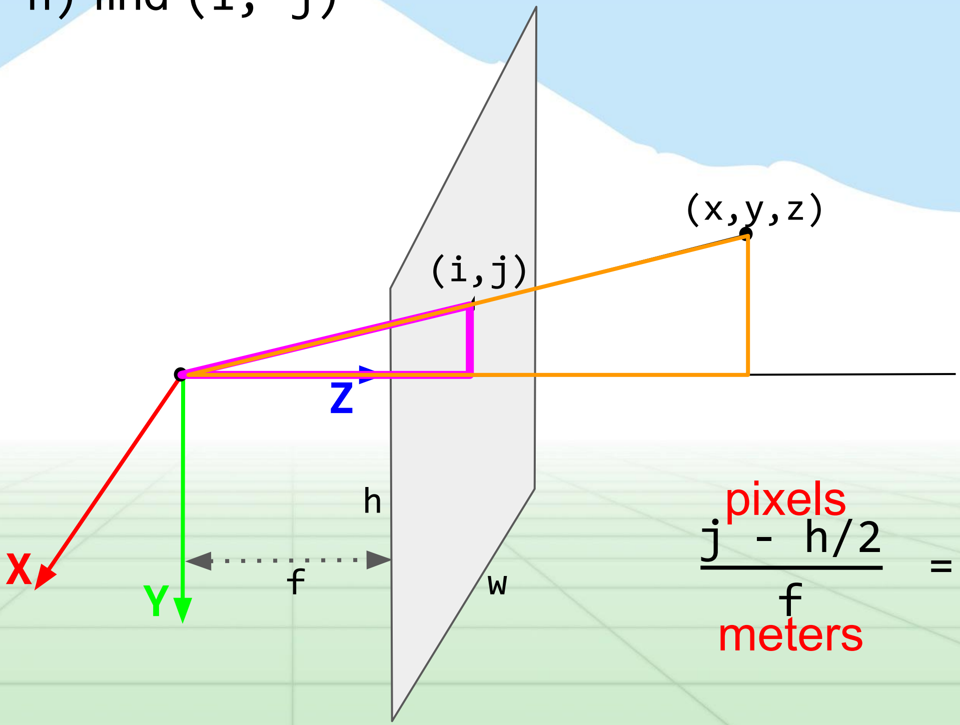
Given (x, y, z) and (f, w, h) find (i, j)

$i = ?$

$j = ?$

Pixel Pitch:

$$p = \frac{w\ (\text{pixels})}{w\ (\text{meters})}$$



$$\frac{\overset{\text{pixels}}{j - h/2}}{\underset{\text{meters}}{f}} = \frac{y}{z}$$

Perspective Projection

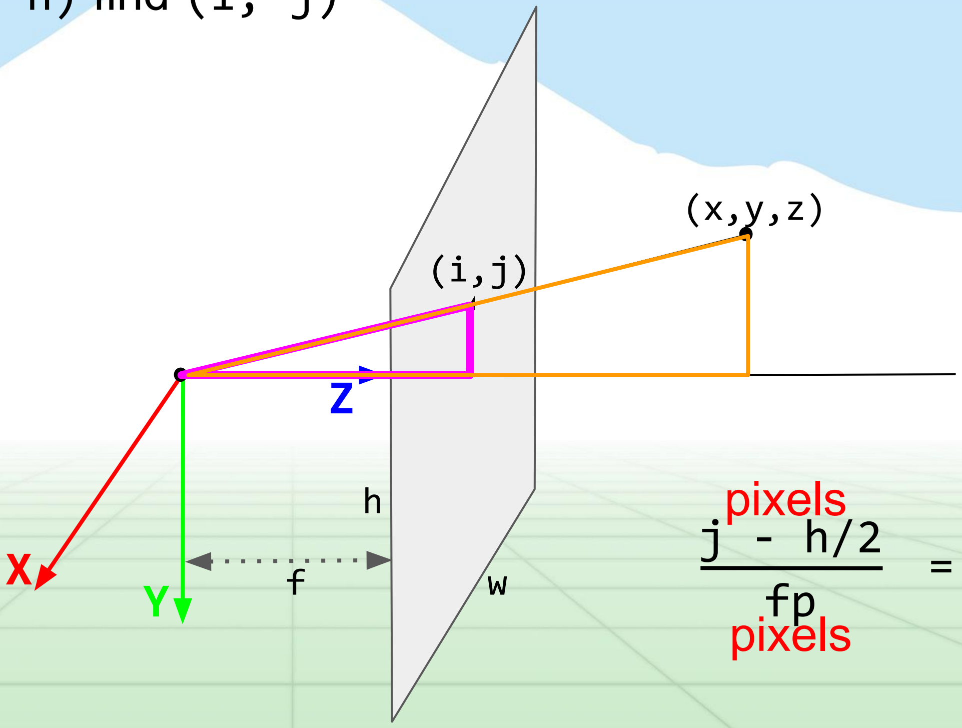
Given (x, y, z) and (f, w, h) find (i, j)

i = ?

j = ?

Pixel Pitch:

$$p = \frac{w\ (\text{pixels})}{w\ (\text{meters})}$$



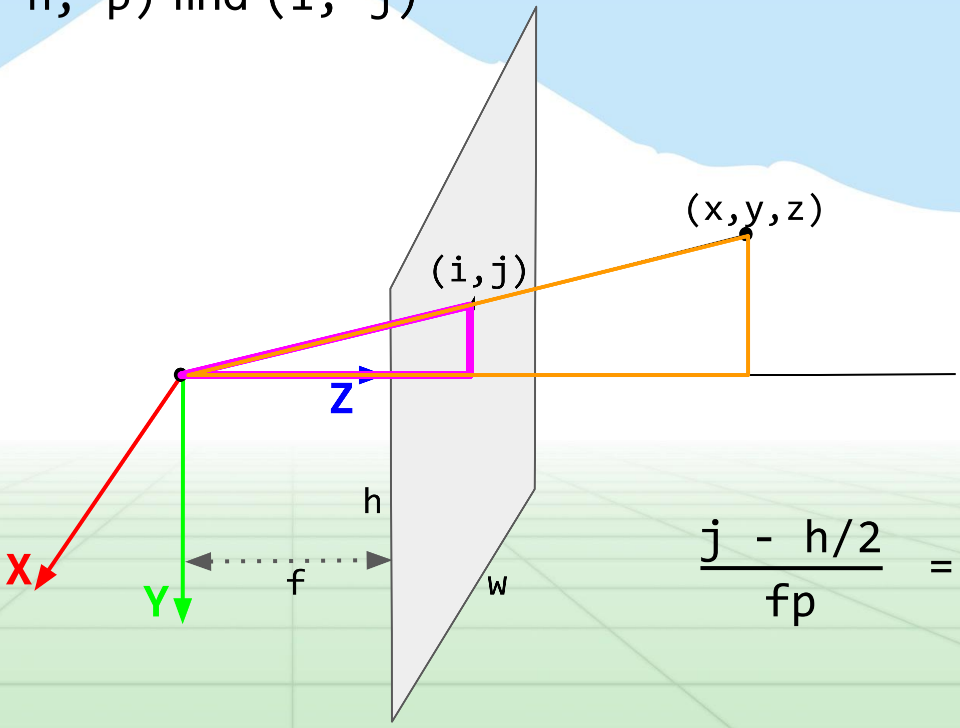
$$\frac{\overset{\text{pixels}}{j - h/2}}{\underset{\text{pixels}}{fp}} = \frac{y}{z}$$

Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = ?$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$



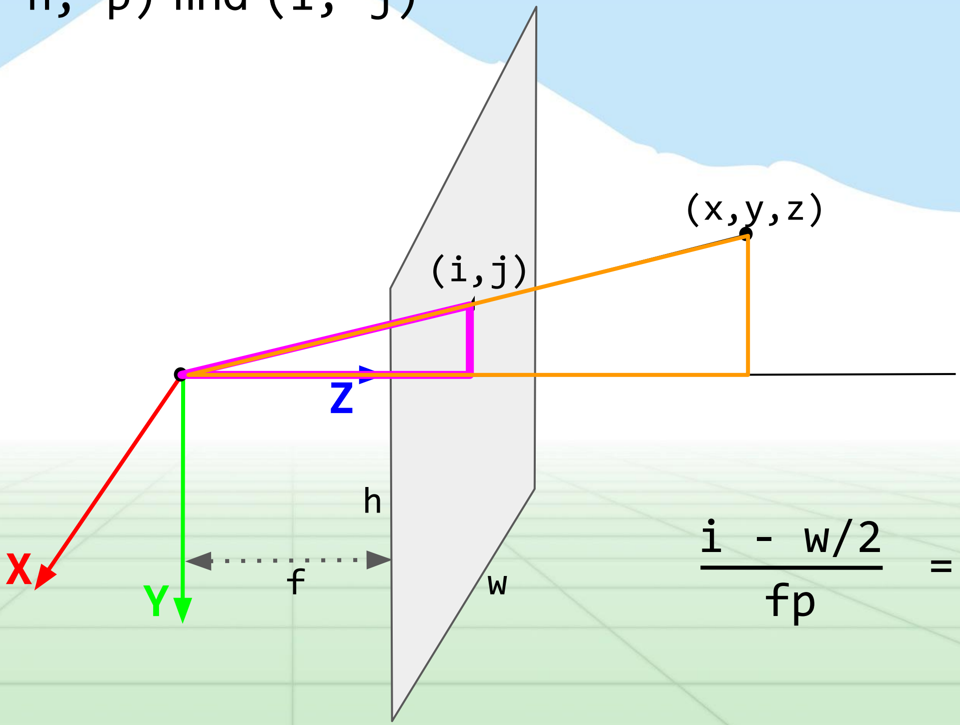
$$\frac{j - h/2}{fp} = \frac{y}{z}$$

Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$



$$\frac{i - w/2}{fp} = \frac{x}{z}$$

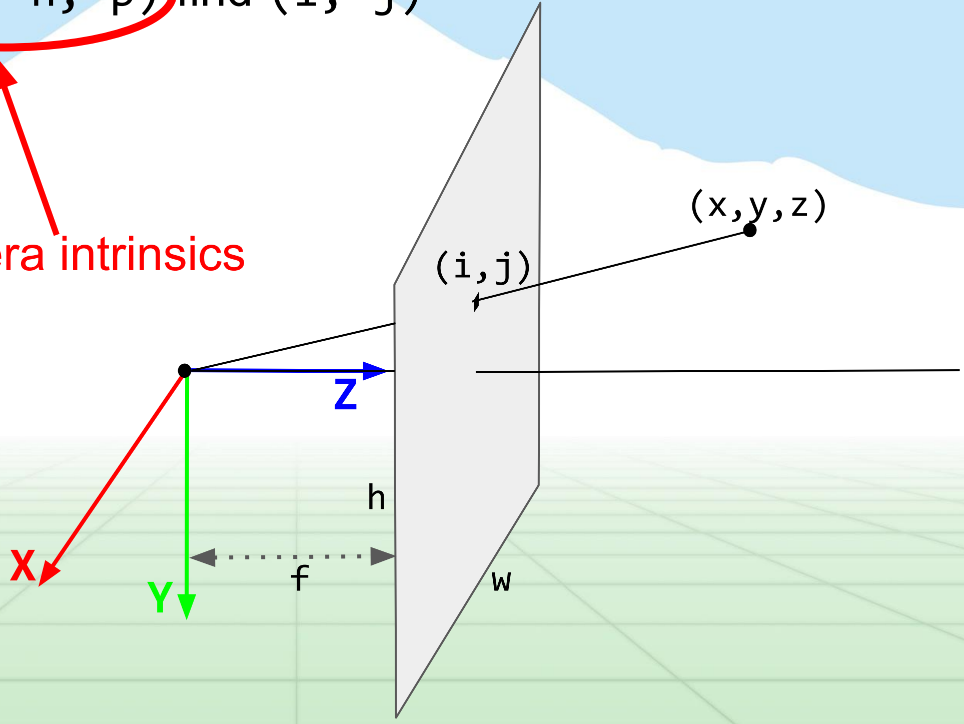
Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$

Camera intrinsics

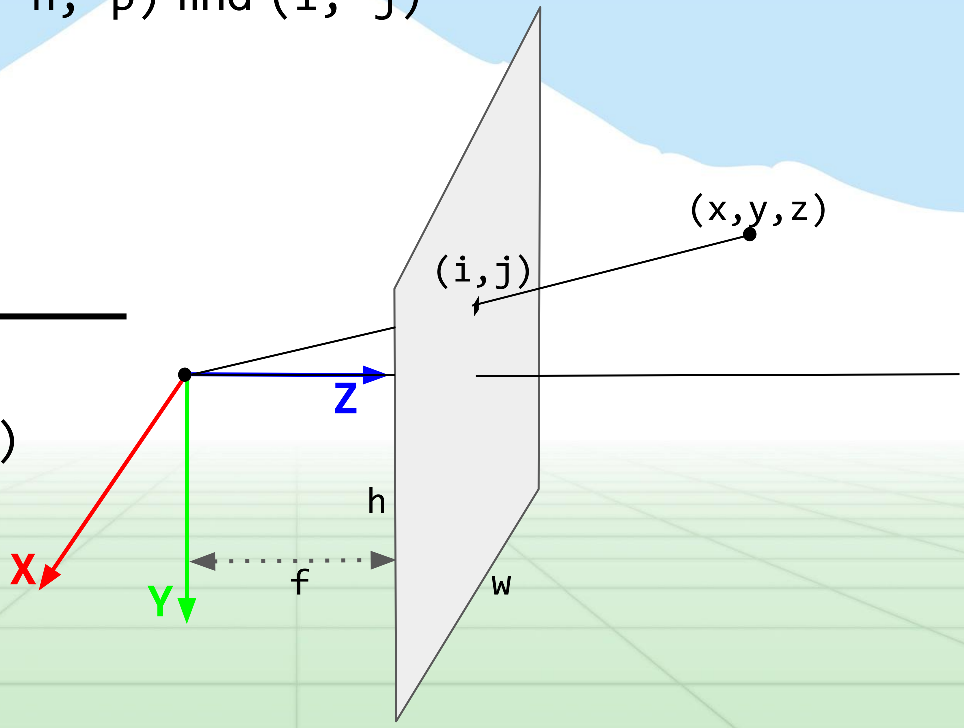


Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$



Given (i, j) and
(f, w, h, p) find (x, y, z)

x = ?

y = ?

z = ?

Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

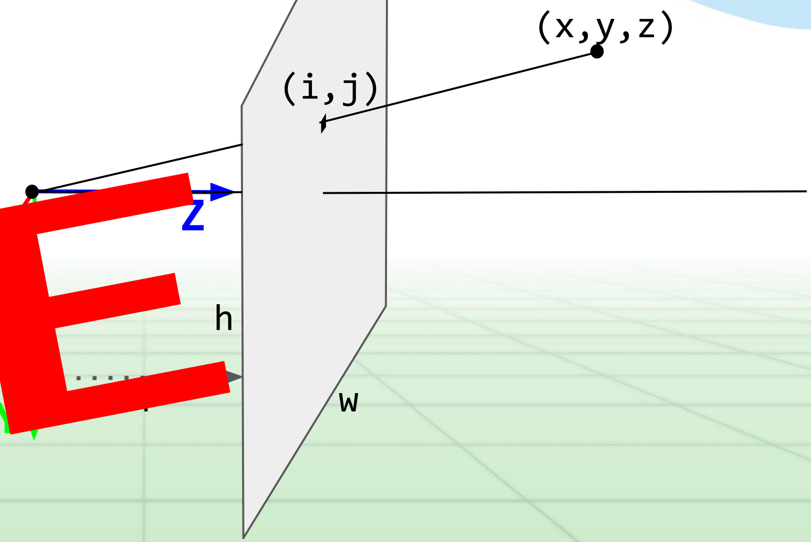
$$j = \frac{fpy}{z} + \frac{h}{2}$$

Given (i, j) and (f, w, h, p) find (x, y, z)

x = ?

y

z = ?



NOPE

Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$

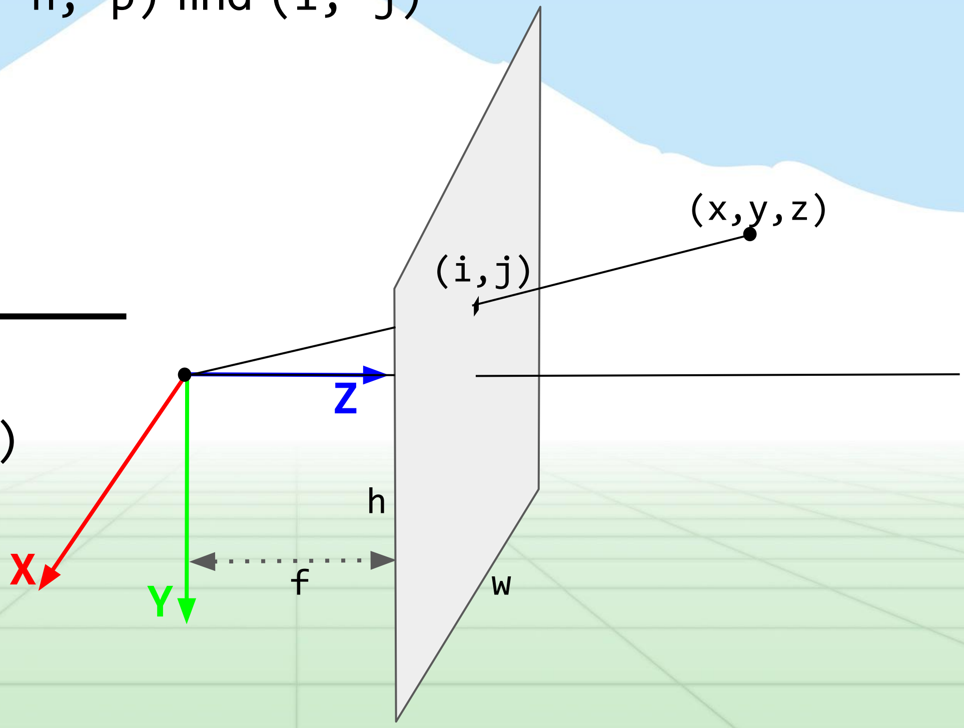
Depth

Given (i, j, d) and
(f, w, h, p) find (x, y, z)

x = ?

y = ?

z = ?



Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

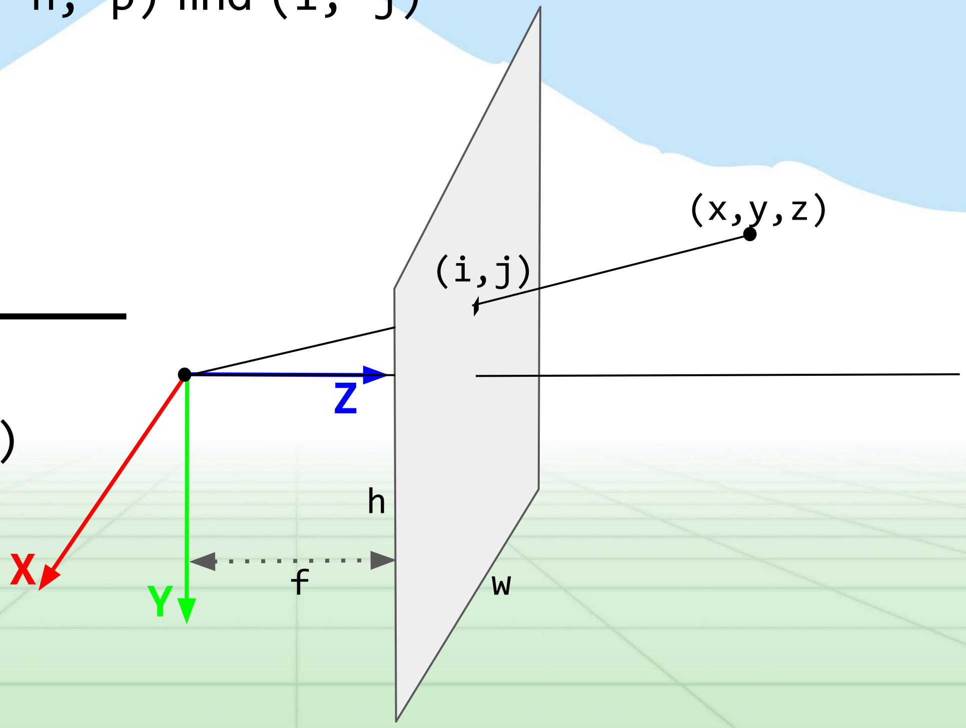
$$j = \frac{fpy}{z} + \frac{h}{2}$$

Given (i, j, d) and (f, w, h, p) find (x, y, z)

$$x = ?$$

$$y = ?$$

$$z = d$$



Perspective Projection

Given (x, y, z) and (f, w, h, p) find (i, j)

$$i = \frac{fpx}{z} + \frac{w}{2}$$

$$j = \frac{fpy}{z} + \frac{h}{2}$$

Given (i, j, d) and
(f, w, h, p) find (x, y, z)

$$x = \frac{(i - w/2)d}{fp}$$

$$y = \frac{(j - h/2)d}{fp}$$

$$z = d$$

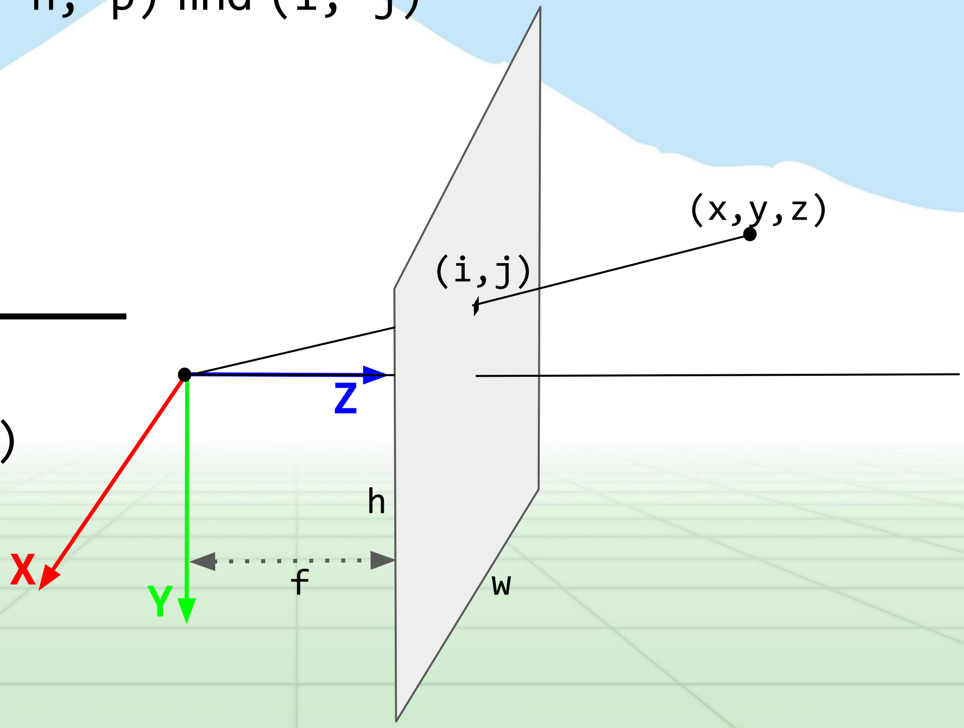


Image Coordinates

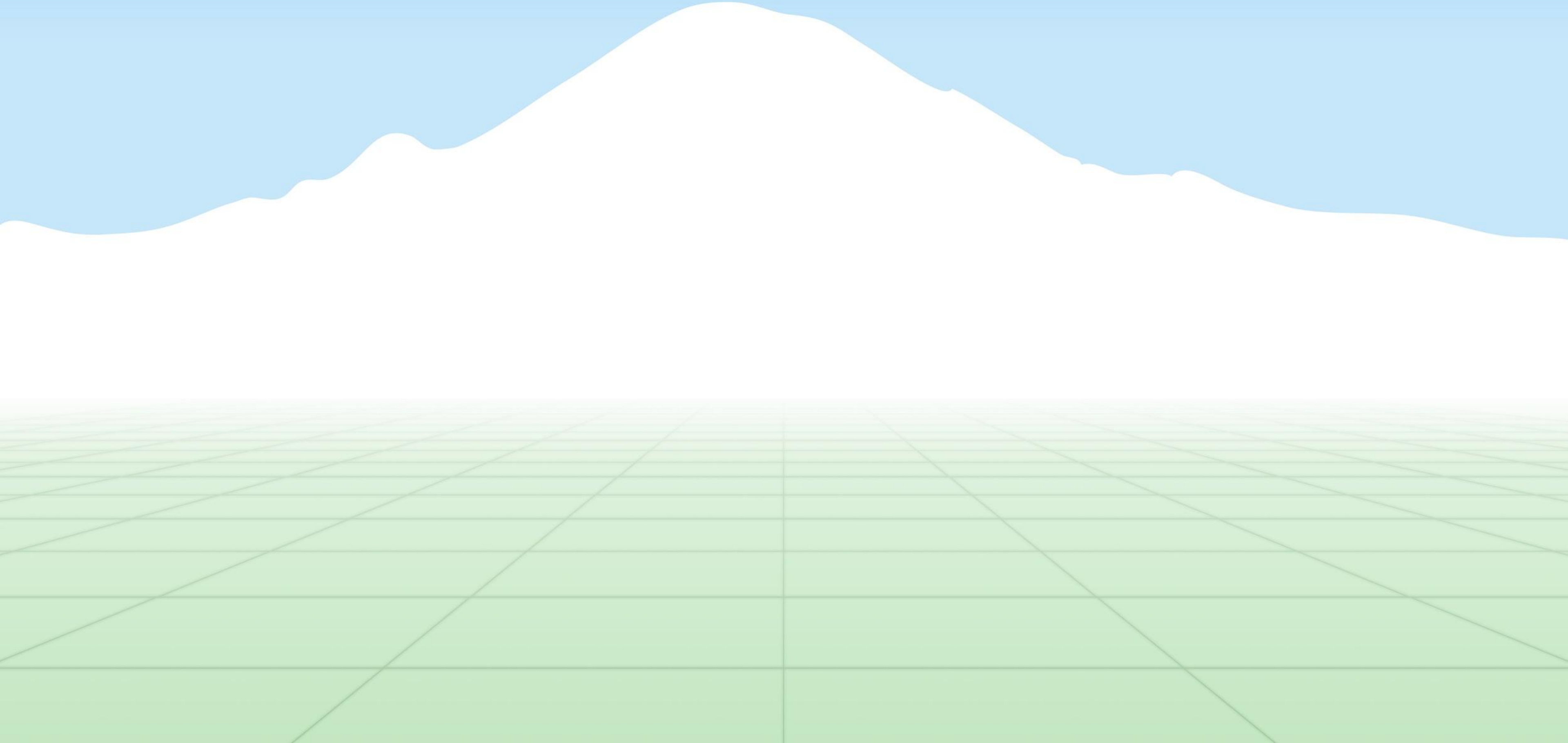


Image Coordinates

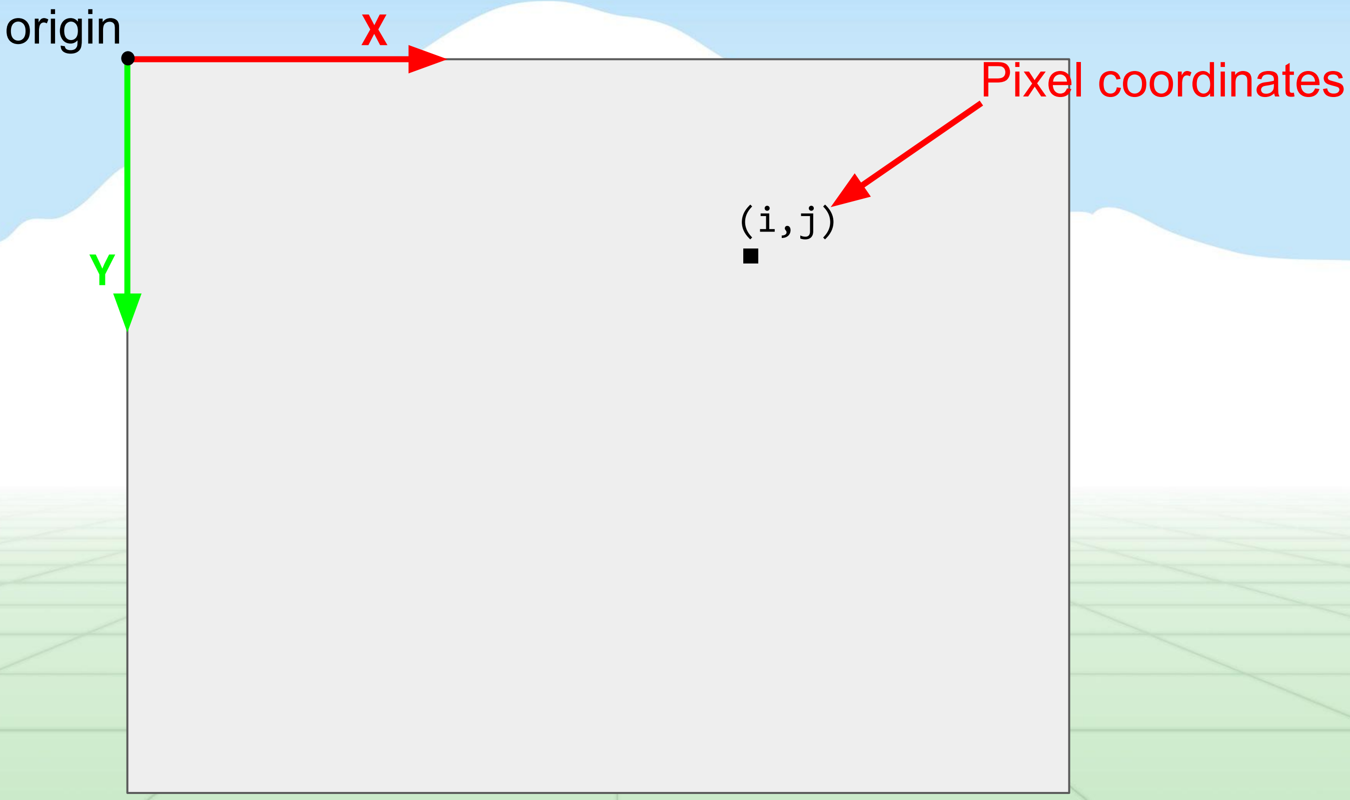


Image Coordinates

3D coordinates of the pixel on the *infinite* image plane.

x', y', z' in meters.

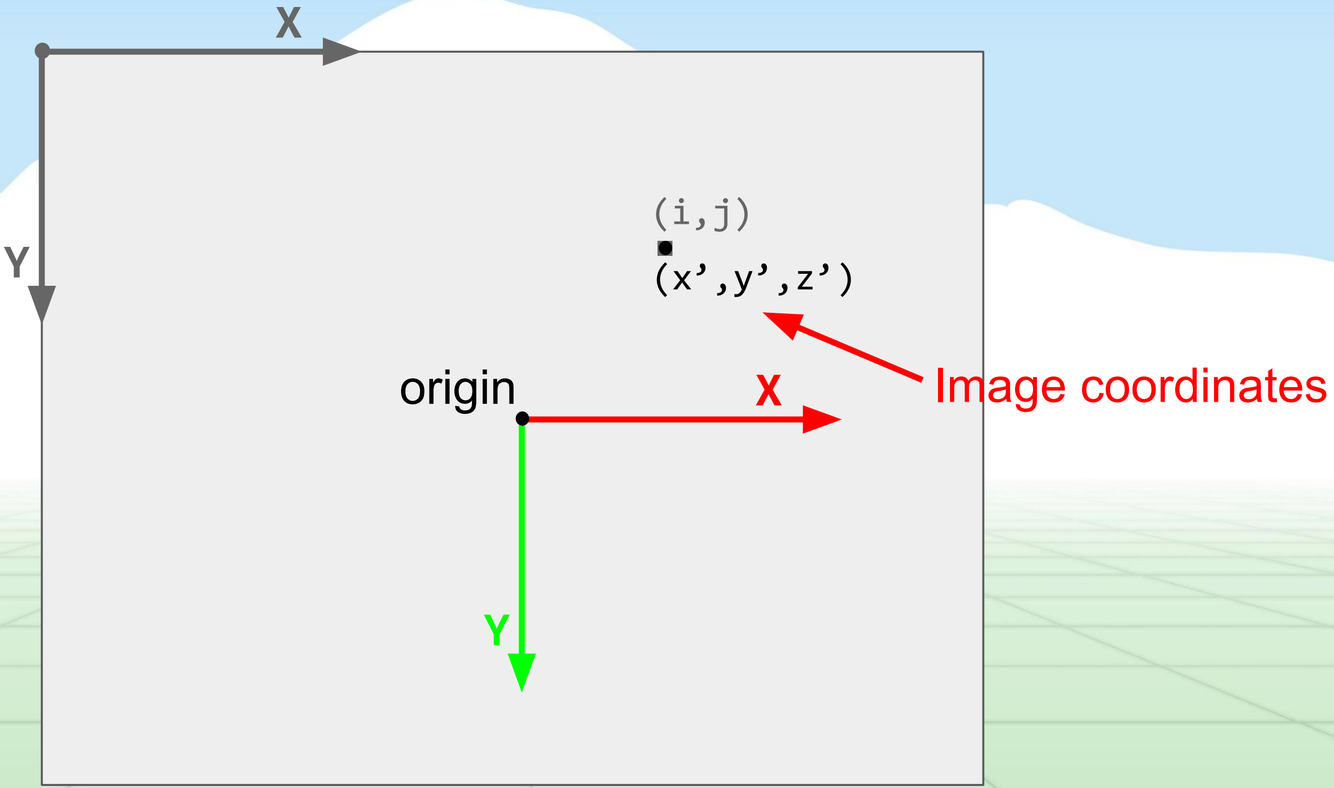


Image Coordinates

3D coordinates of the pixel on the *infinite* image plane.

x', y', z' in meters.

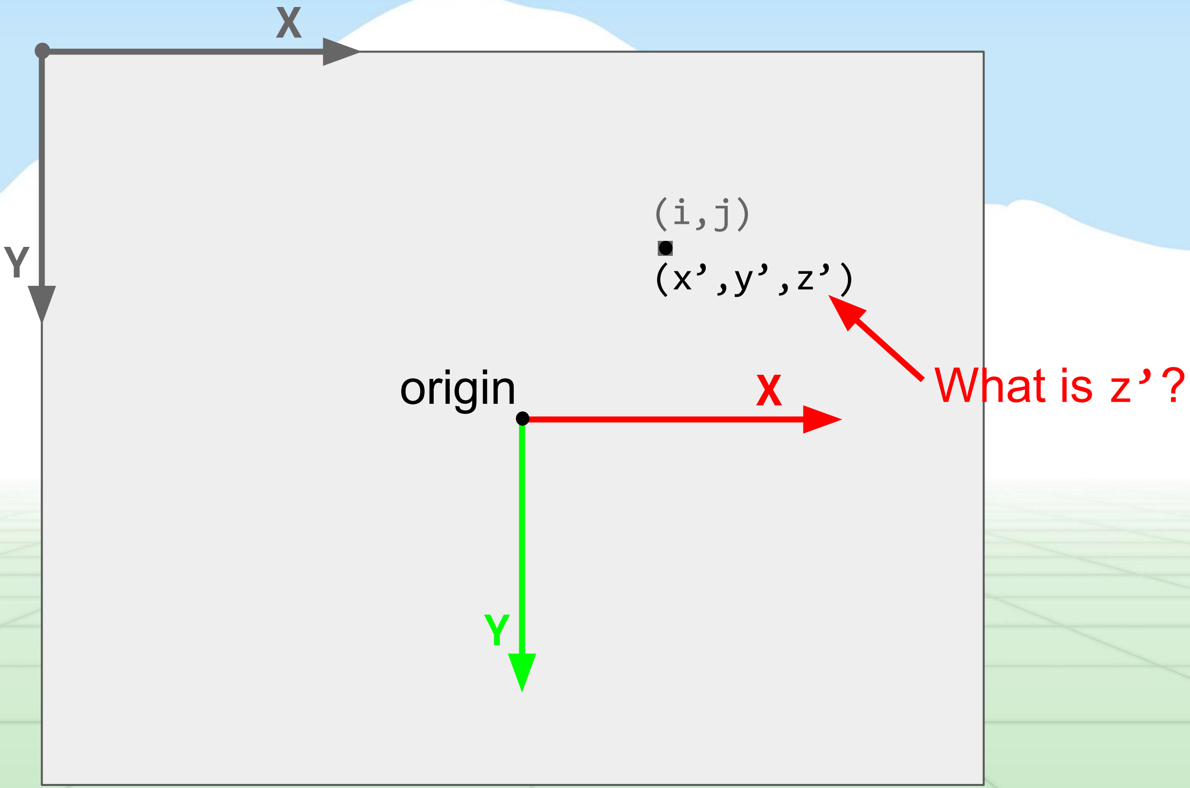


Image Coordinates

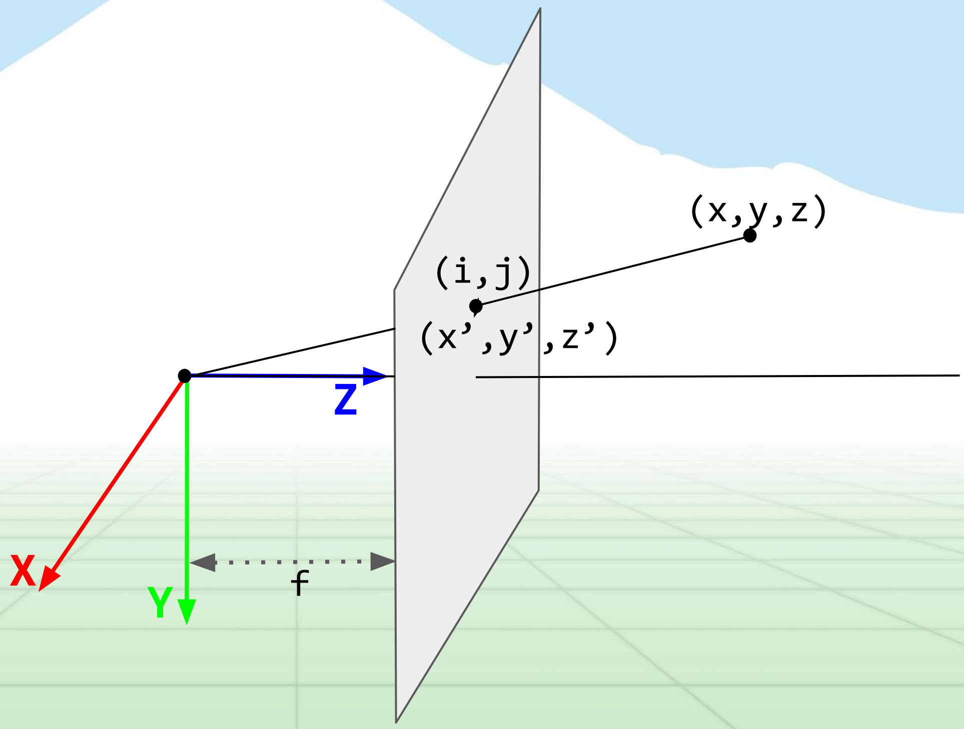


Image Coordinates

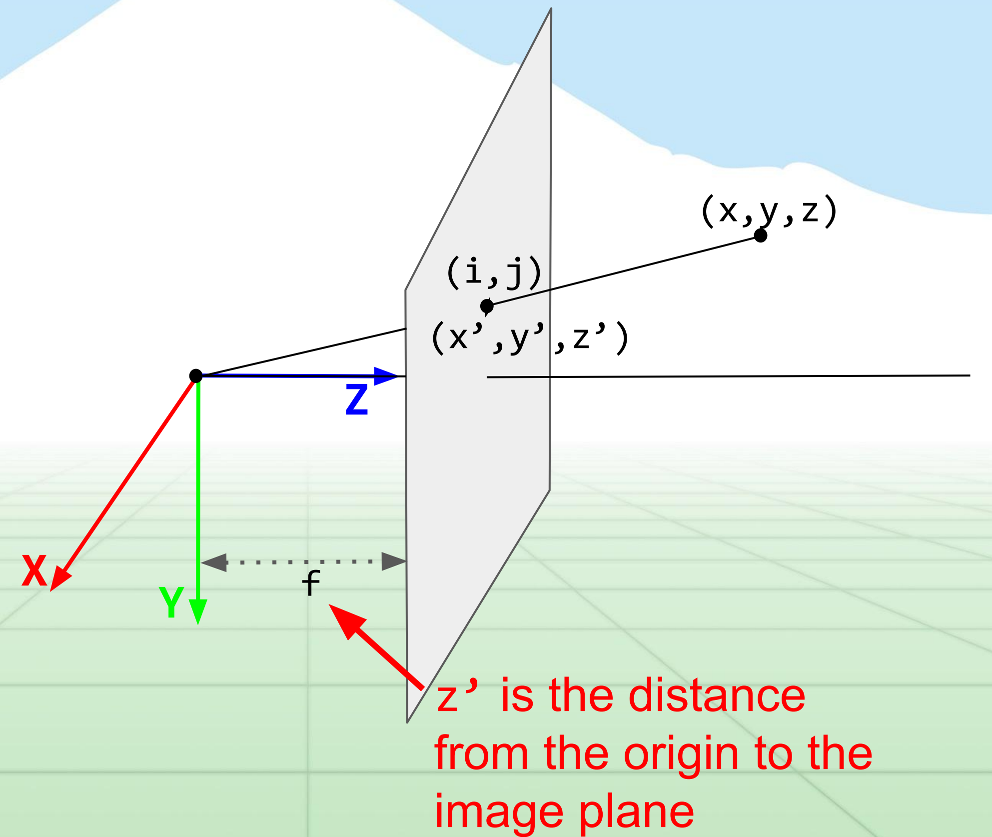


Image Coordinates

$$i = px' + \frac{w}{2}$$

$$j = py' + \frac{h}{2}$$

$$x' = \frac{fx}{z}$$

$$y' = \frac{fy}{z}$$

$$z' = f$$

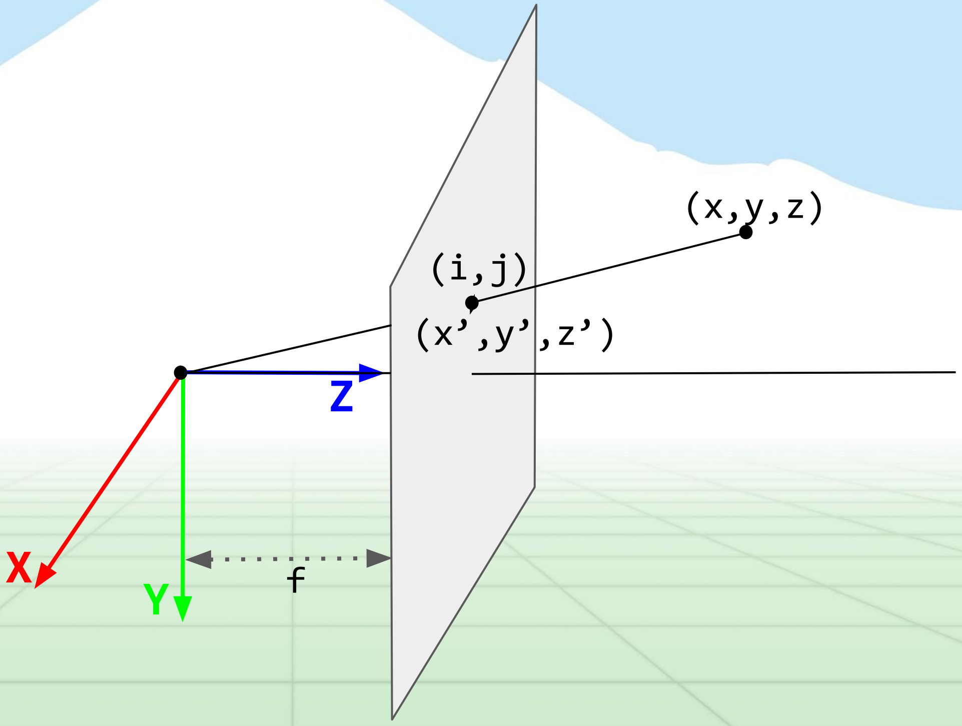


Image Coordinates

Re-scale (meters -> pixels)

$$i = px' + \frac{w}{2}$$

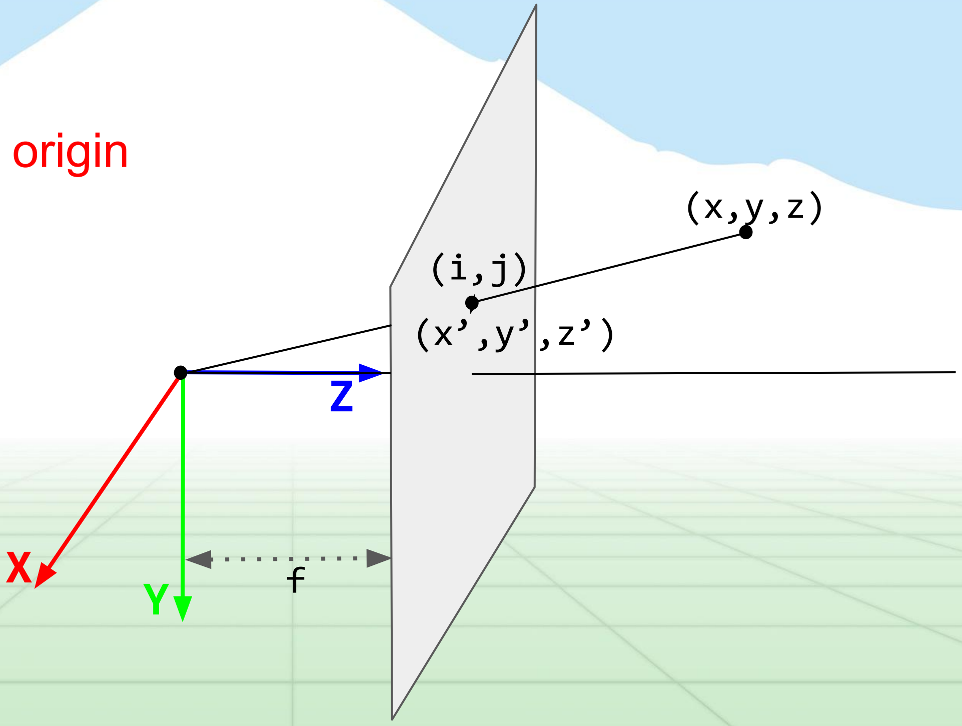
$$j = py' + \frac{h}{2}$$

Shift origin

$$x' = \frac{fx}{z}$$

$$y' = \frac{fy}{z}$$

$$z' = f$$



Normalized Image Coordinates

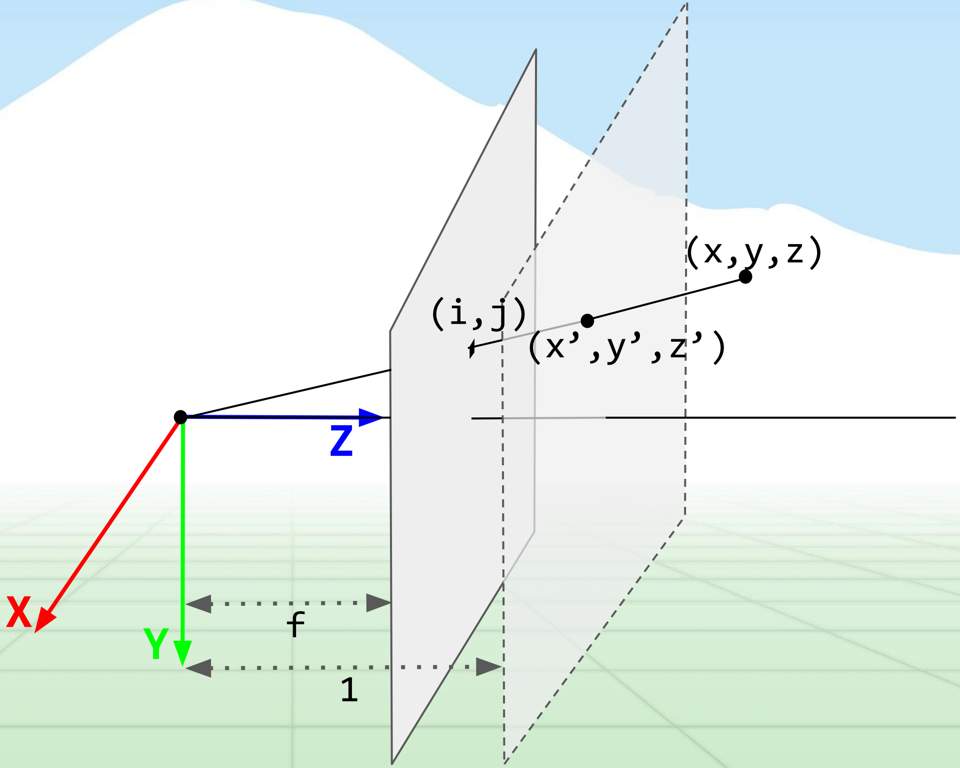
$$i = fpx' + \frac{w}{2}$$

$$j = fpy' + \frac{h}{2}$$

$$x' = \frac{x}{z}$$

$$y' = \frac{y}{z}$$

$$z' = 1$$



Normalized Image Coordinates

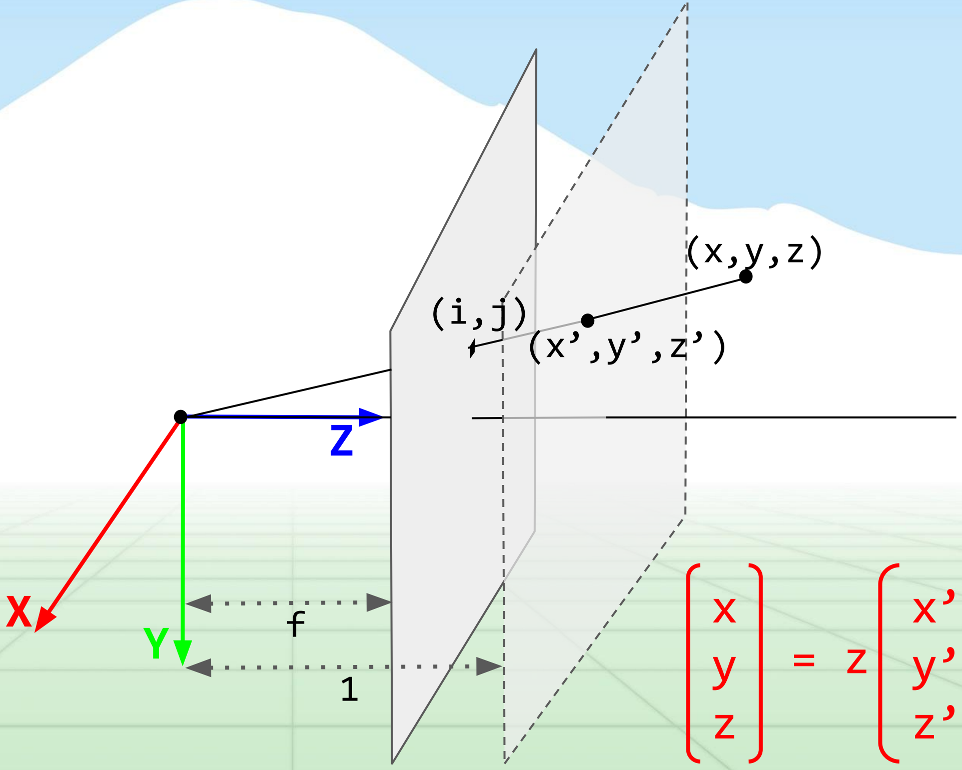
$$i = fpx' + \frac{w}{2}$$

$$j = fpy' + \frac{h}{2}$$

$$x' = \frac{x}{z}$$

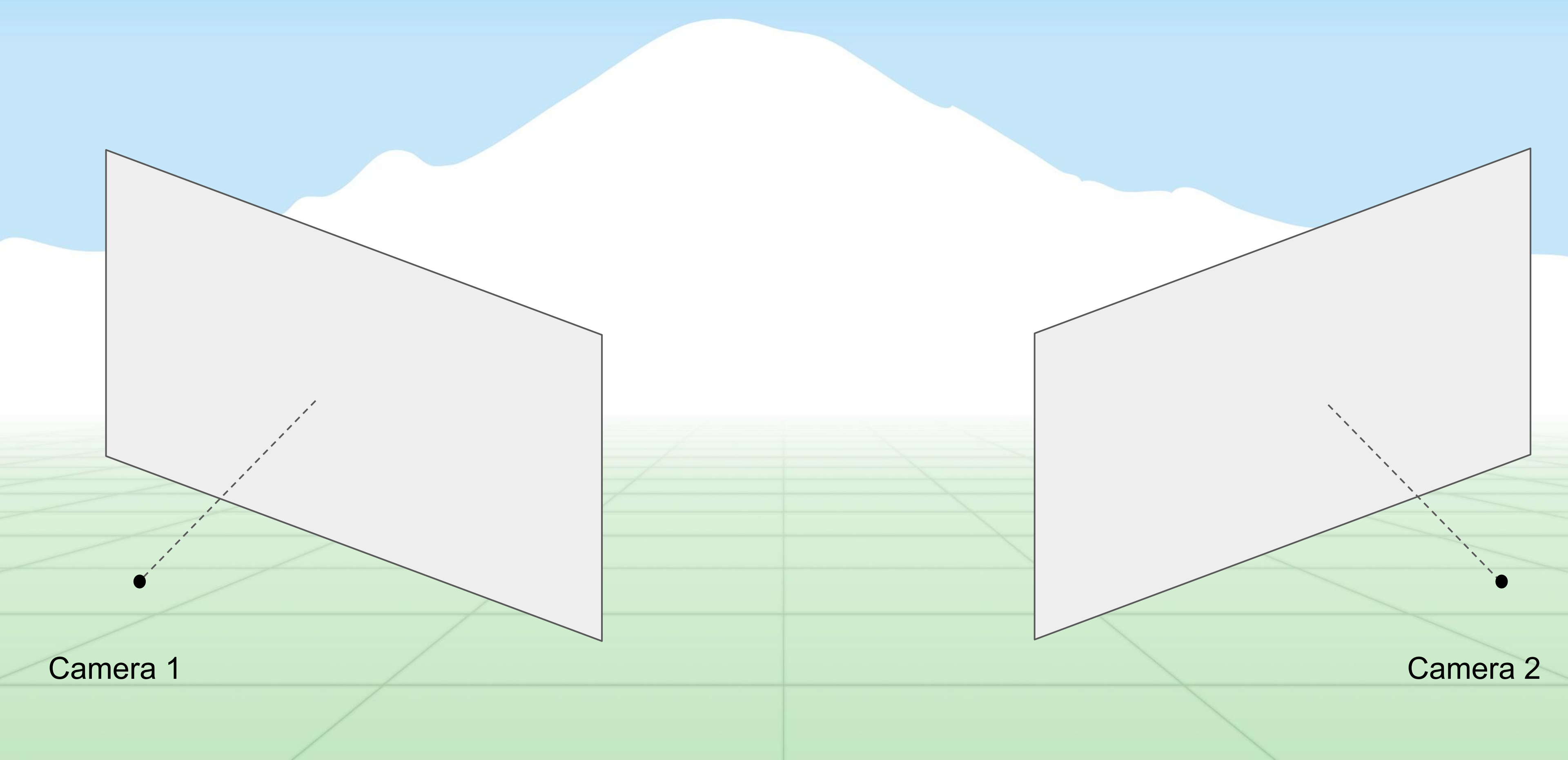
$$y' = \frac{y}{z}$$

$$z' = 1$$

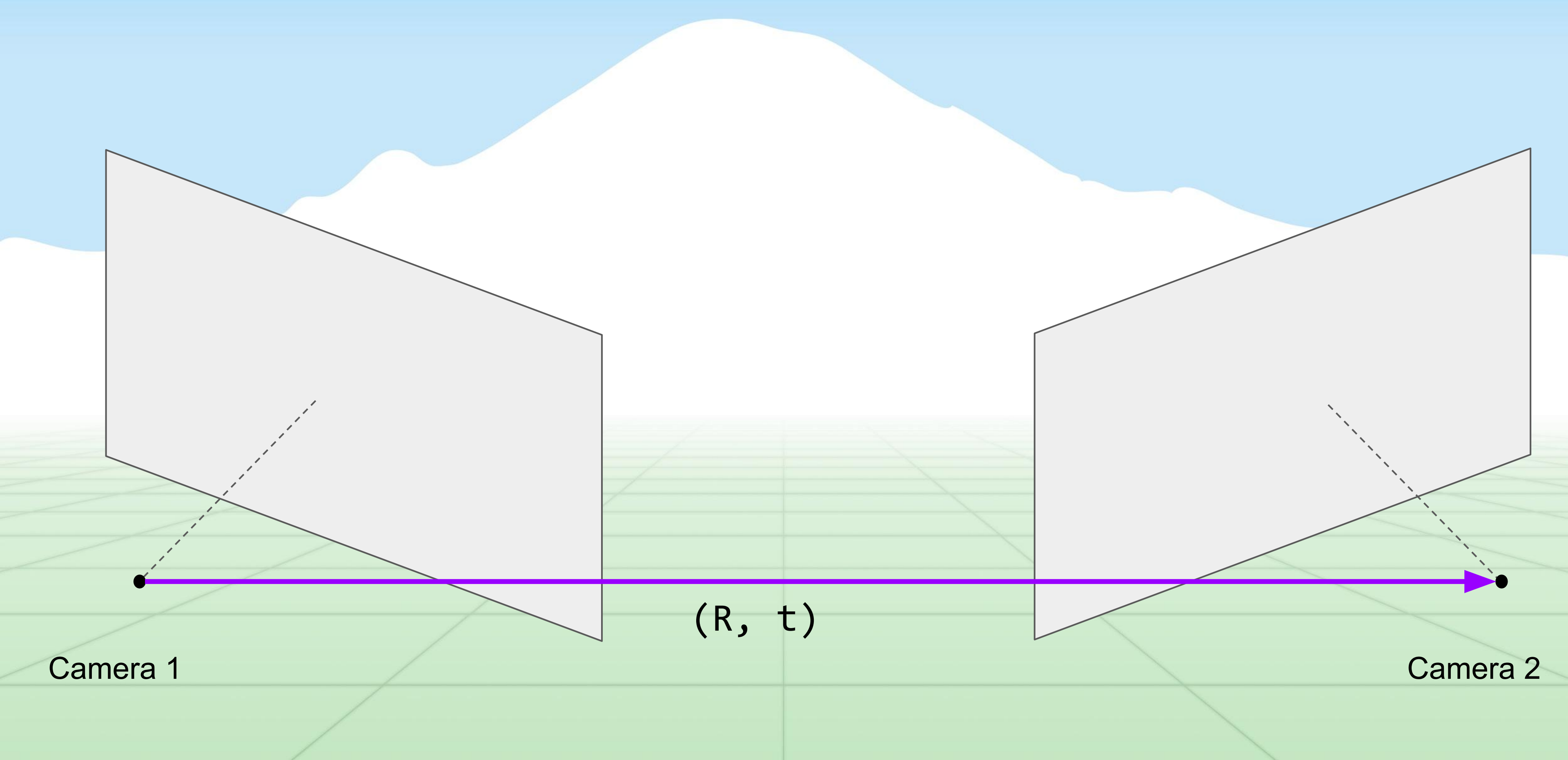


$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

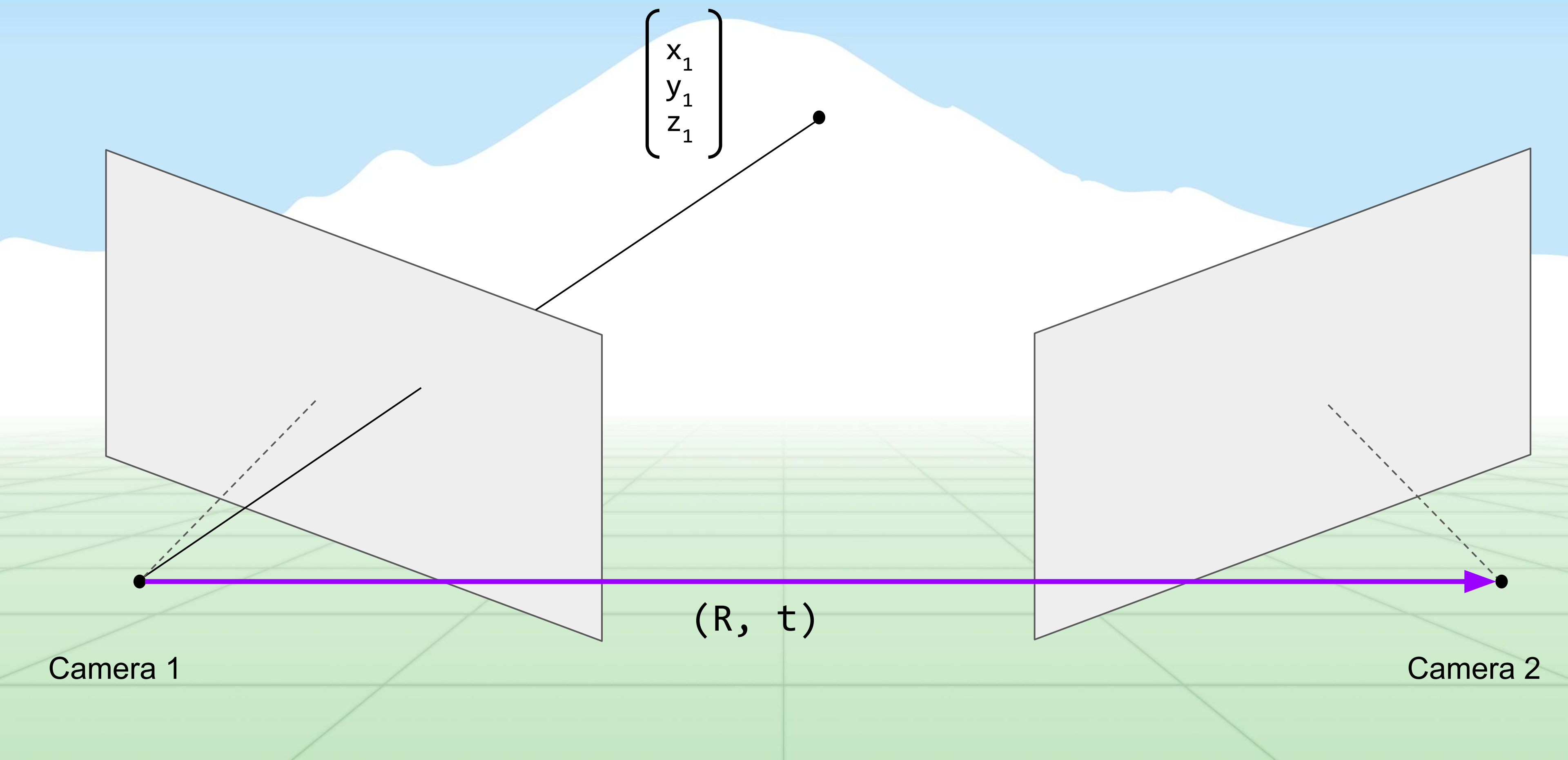
Stereo



Stereo

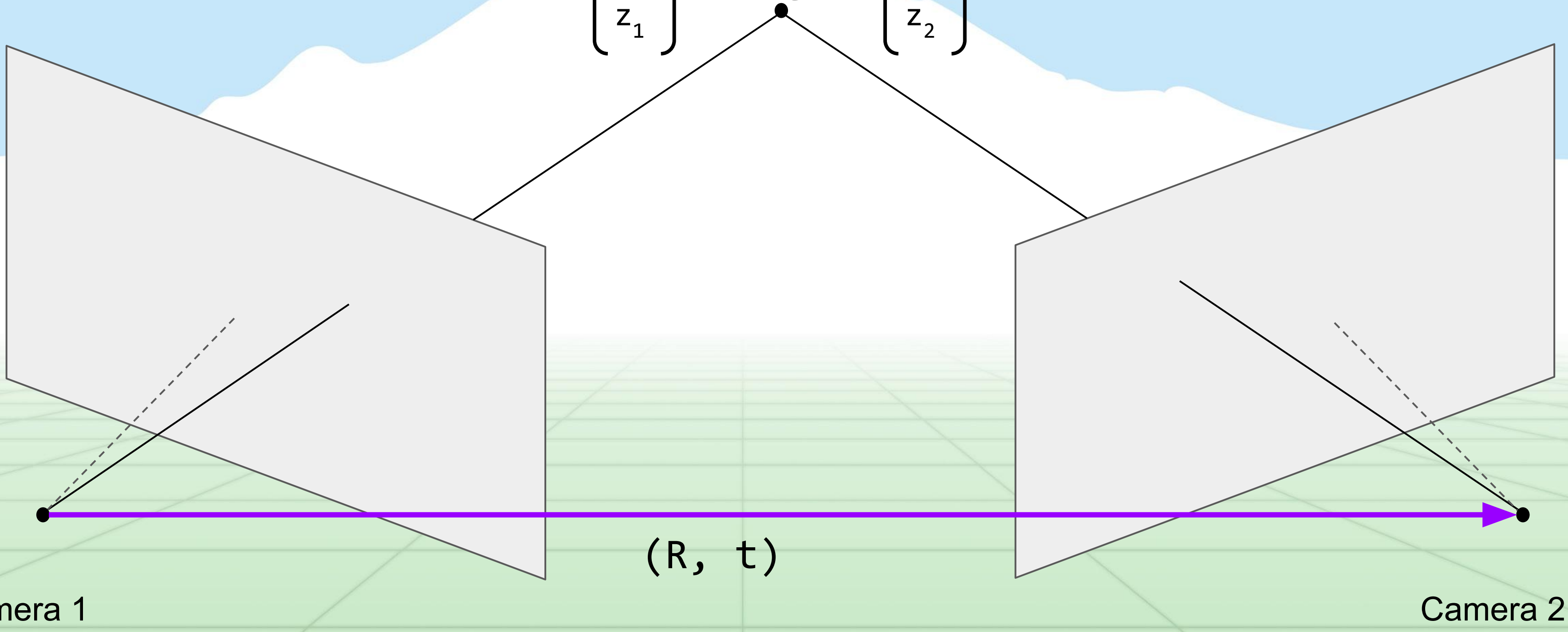


Stereo

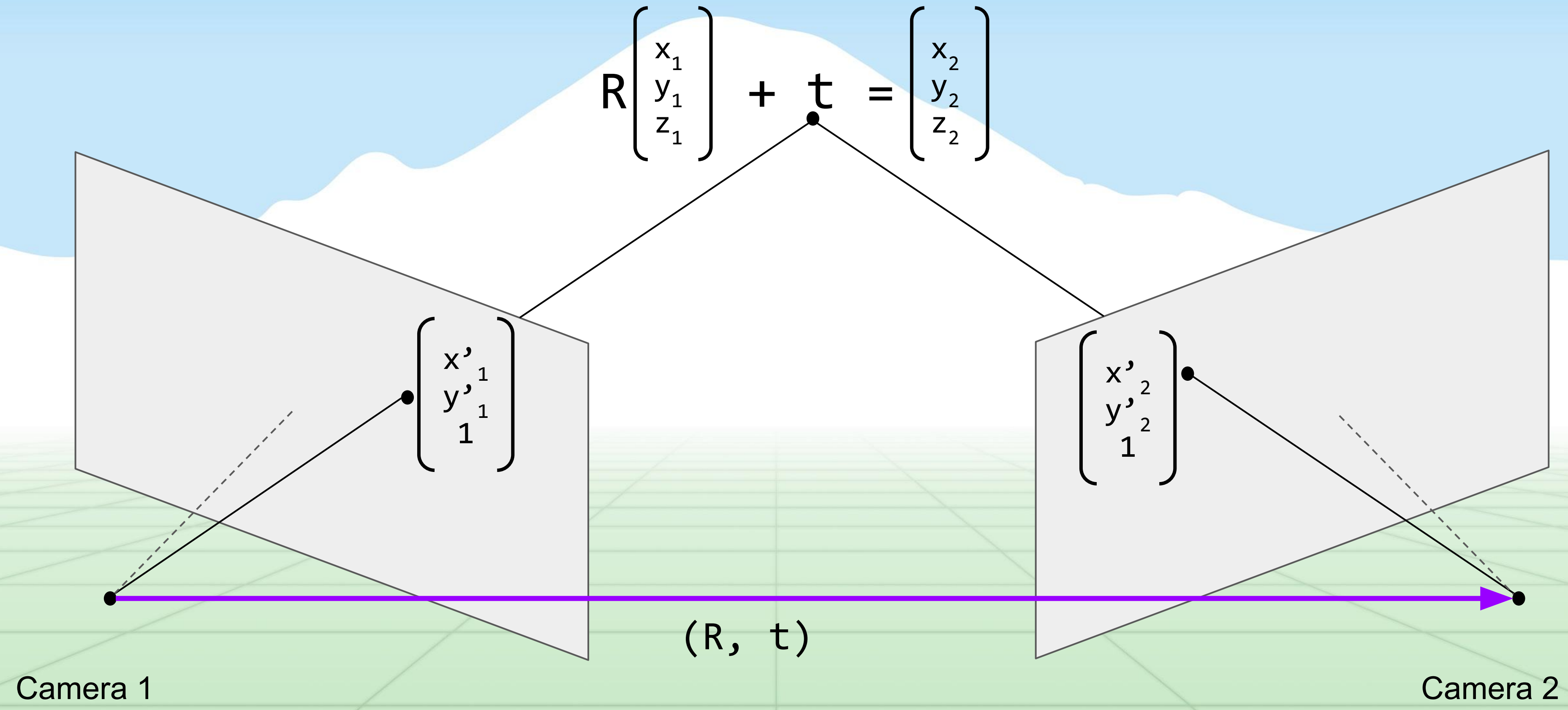


Stereo

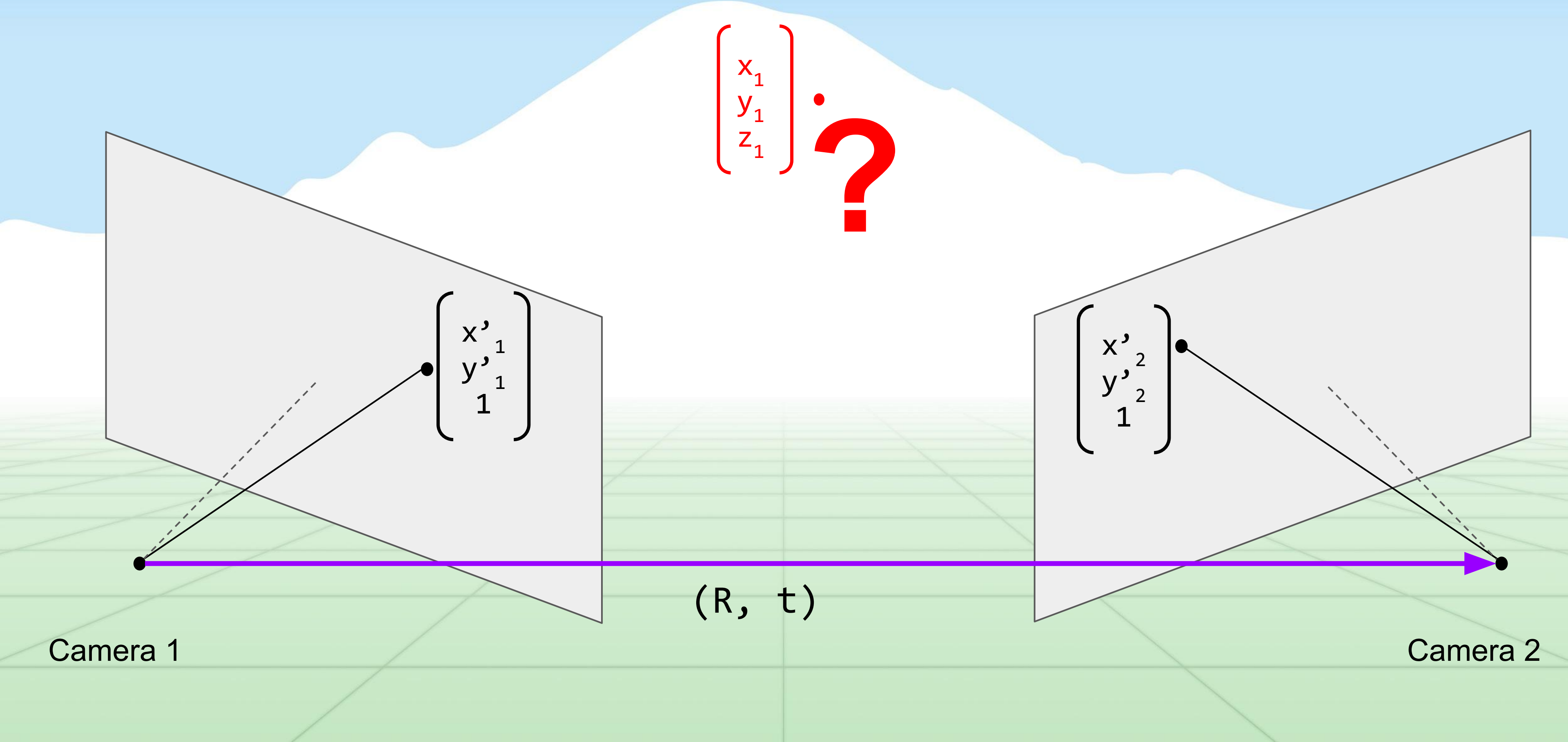
$$R\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + t = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$



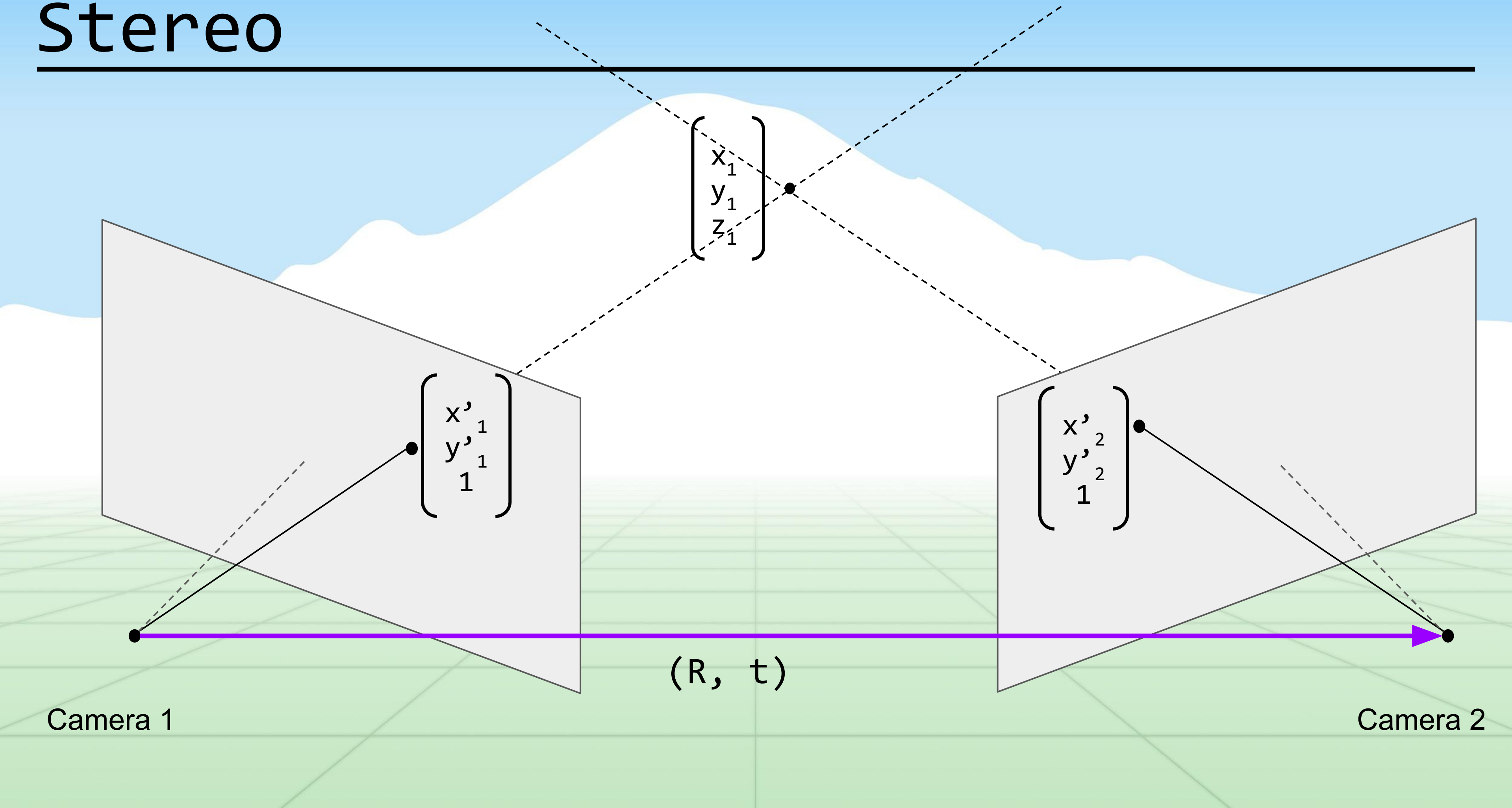
Stereo


$R\begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} + t = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$
$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$
$\begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$
(R, t)
Camera 1
Camera 2

Stereo



Stereo



Stereo

A line in 3D can be represented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} + s \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$$

Point on the line

Parametric variable

Slope of line

Camera

ra 2

Stereo

A line in 3D can be represented as:

Line 1 $$\begin{bmatrix}0\\0\\0\end{bmatrix} + s_1\begin{bmatrix}x'_1\\y'_1\\1\end{bmatrix}$$

Line 2 $$\begin{bmatrix}0\\0\\0\end{bmatrix} + s_2\begin{bmatrix}x'_2\\y'_2\\1\end{bmatrix} \xrightarrow{\text{Transform to camera 1 coordinate frame}} -t + R^{-1}s_2\begin{bmatrix}x'_2\\y'_2\\1\end{bmatrix}$$

Camera

ra 2

Stereo

A line in 3D can be represented as:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = z \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix}$$

Line 1

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + s_1 \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$$

Line 2

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + s_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix} \xrightarrow{\text{Transform to camera 1 coordinate frame}} -t + R^{-1} s_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$$

Camera 1

Camera 2

Stereo

A line in 3D can be represented as:

Line 1

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + z_1 \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix}$$

Line 2

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix} \xrightarrow{\text{Transform to camera 1 coordinate frame}} -t + R^{-1} z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix}$$

Camera

ra 2

Stereo

A line in 3D can be represented as:

Line 1

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + z_1 \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix}$$

Line 2

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix} \xrightarrow{\text{Transform to camera 1 coordinate frame}} -t + R^{-1} z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix}$$

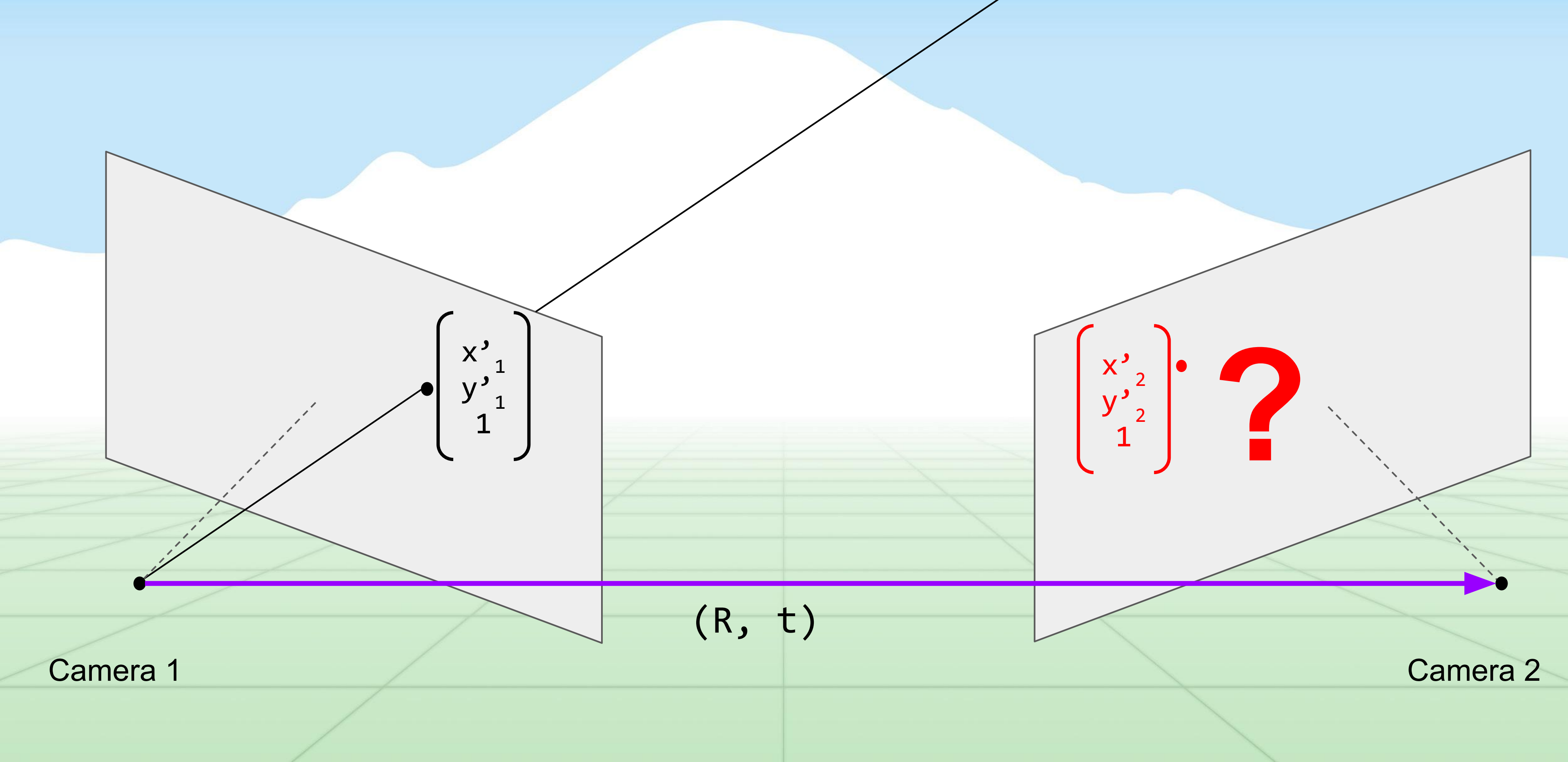
Solve system of linear equations

$$z_1 \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = -t + R^{-1} z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix}$$

Camera

ra 2

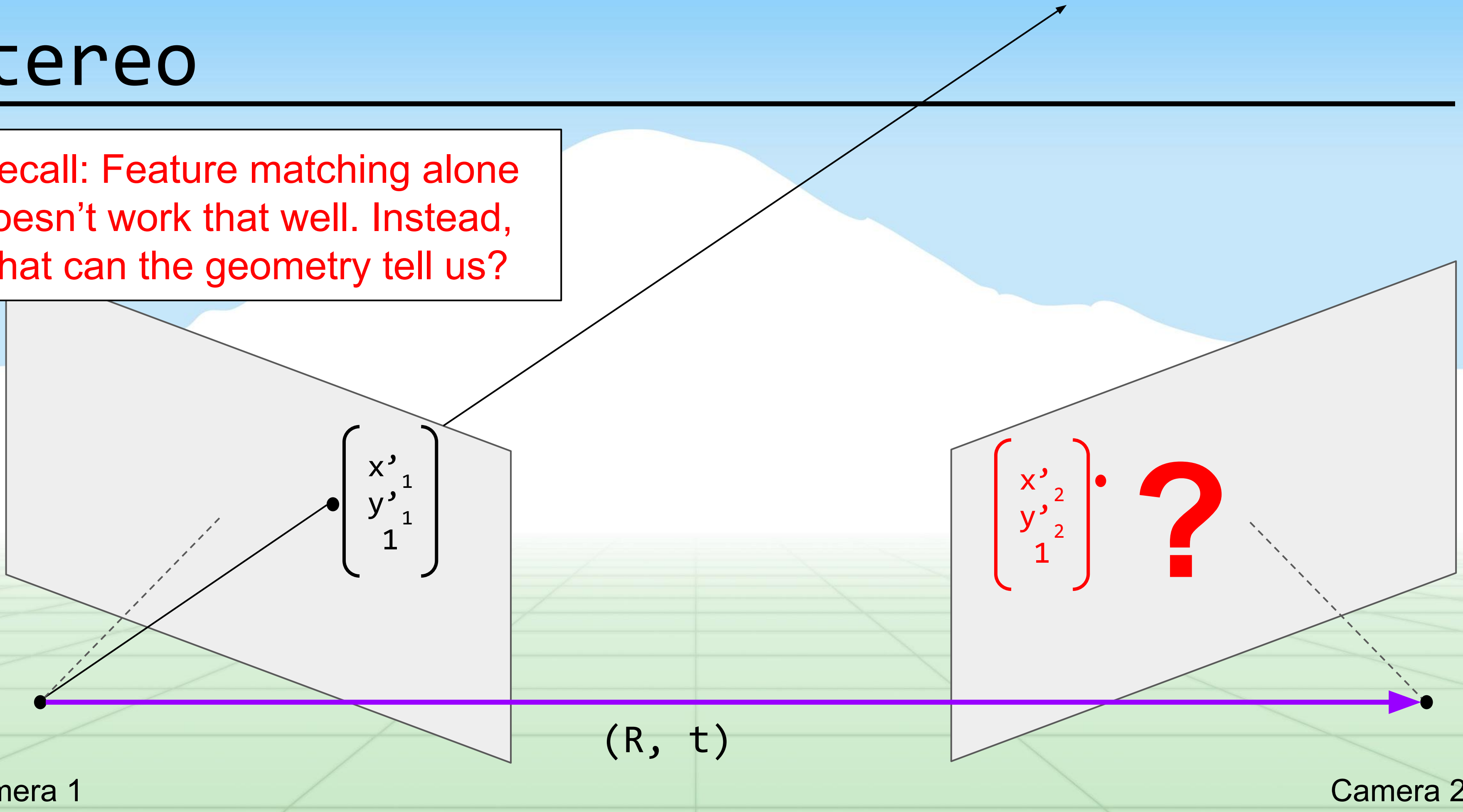
Stereo



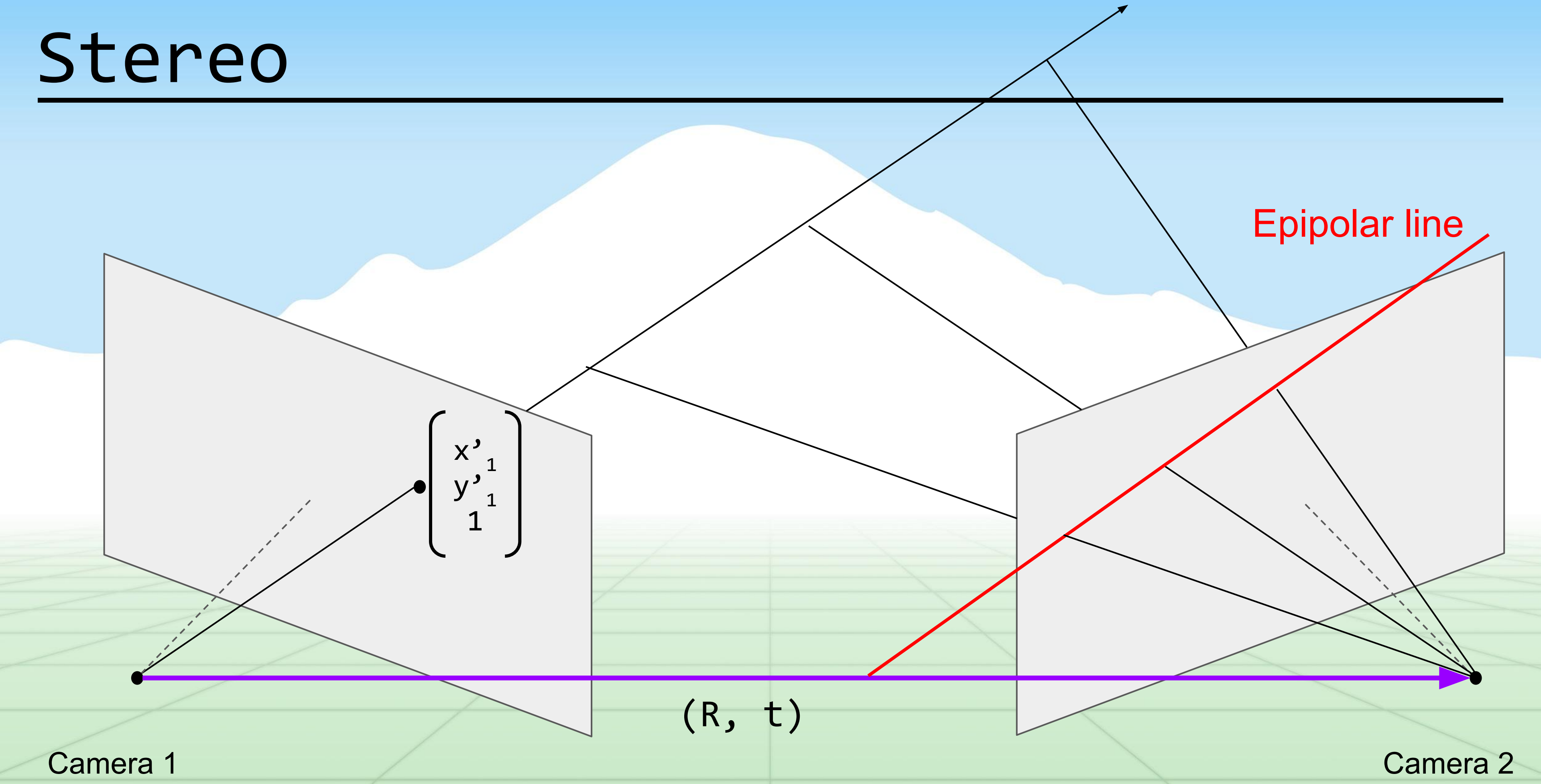
$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$
$\begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$?
(R, t)
Camera 1
Camera 2

Stereo

Recall: Feature matching alone doesn't work that well. Instead, what can the geometry tell us?

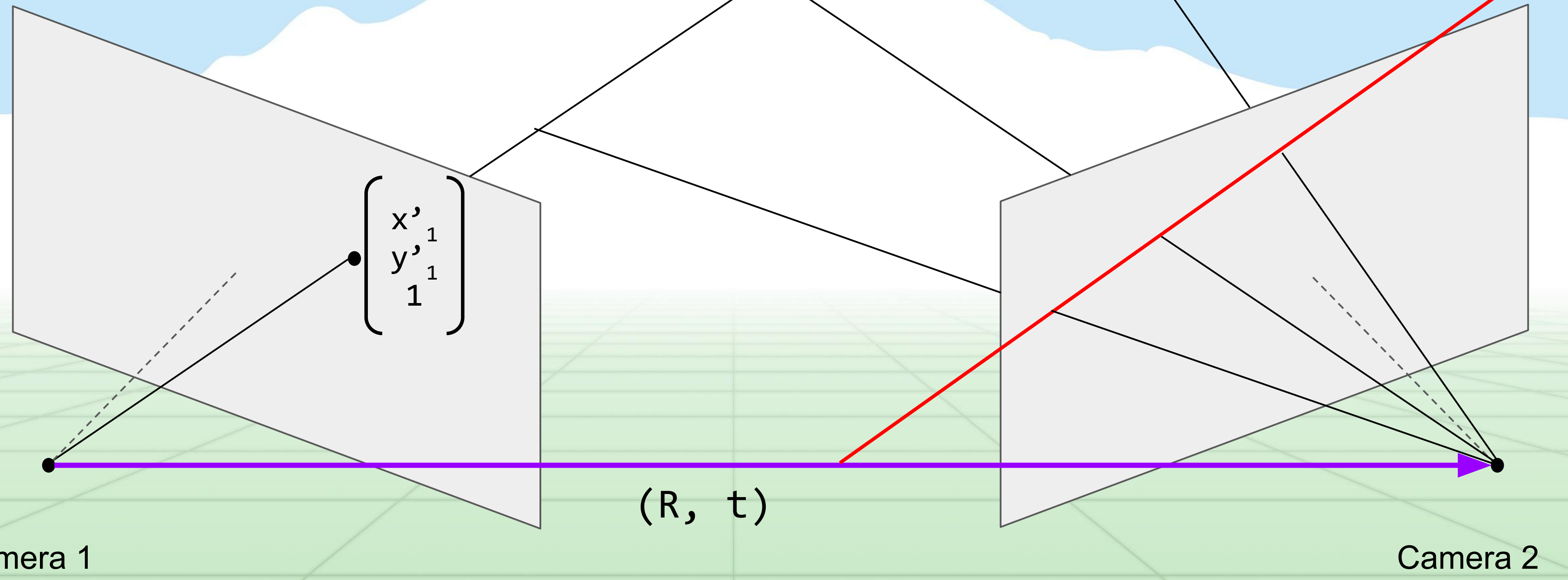


Stereo

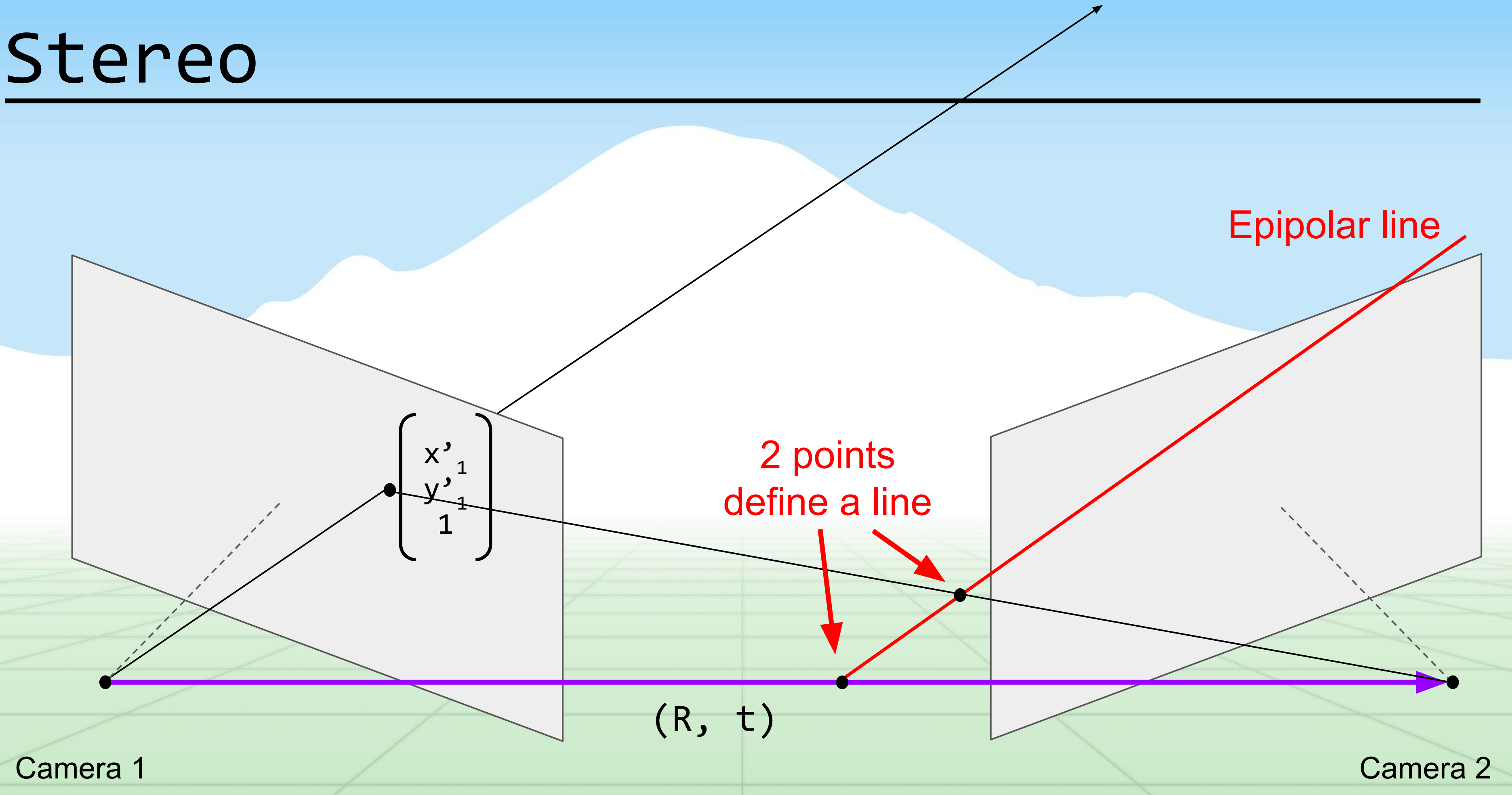


Stereo

Searching for the matching feature along this line is much easier



Stereo



Epipolar line
2 points
define a line
$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$
(R, t)
Camera 1
Camera 2

Stereo

$$\left[t + R\begin{pmatrix}x'_1\\ y'_1\\ 1\end{pmatrix}\right]\frac{1}{z''_1}$$

$$\left[t + R\begin{pmatrix}0\\ 0\\ 0\end{pmatrix}\right]\frac{1}{z'_1}$$

z coordinate
after transform

Camera

ra 2

Stereo

$$\left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1}$$

$$\left[t + R \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \right] \frac{1}{z`_1} \xrightarrow{\text{Simplify}} t \, \frac{1}{t_z}$$

Stereo

2D equation for a line:

$$y = mx + b$$

Camera

ra 2

Stereo

2D equation for a line:

$$y = mx + b$$

Doesn't handle vertical lines

Camera

ra 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

Camera

ra 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

$$\left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1}$$

Both our projected points have the format $\begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$
We can pretend they are points in 2D.

$$t \frac{1}{t_z}$$

Camera 1 ... Camera 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1} = 0$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot t \frac{1}{t_z} = 0$$

Camera

ra 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1} = 0$$

$\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ must be orthogonal to both vectors

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot t \, \frac{1}{t_z} = 0$$

Camera

ra 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$t \frac{1}{t_z} \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo

2D equation for a line:

$$\cancel{y = mx + b}$$

$$ax + by + c = 0$$

$$t\,\frac{1}{t_z} \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] \frac{1}{z''_1} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z}\, t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z} t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z}\, t \times \left[t + R\begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z}\, t \times R\begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z} t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z} t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$ax + by + c = 0$$

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z} t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z} t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$sax + sby + sc = 0$$

$$s \begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z} t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\cancel{\frac{1}{z''_1 t_z}} t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$sax + sby + sc = 0$$

$$s \begin{pmatrix} a \\ b \\ c \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z}\, t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z}\, t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z}\, t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\frac{1}{z''_1 t_z}\, t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\underbrace{\boxed{t \times R}}_{E} \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

E

essential matrix

(Longuet-Higgins, 1981)

Camera

ra 2

Stereo

$$\frac{1}{z''_1 t_z} t \times \left[t + R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} \right] = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

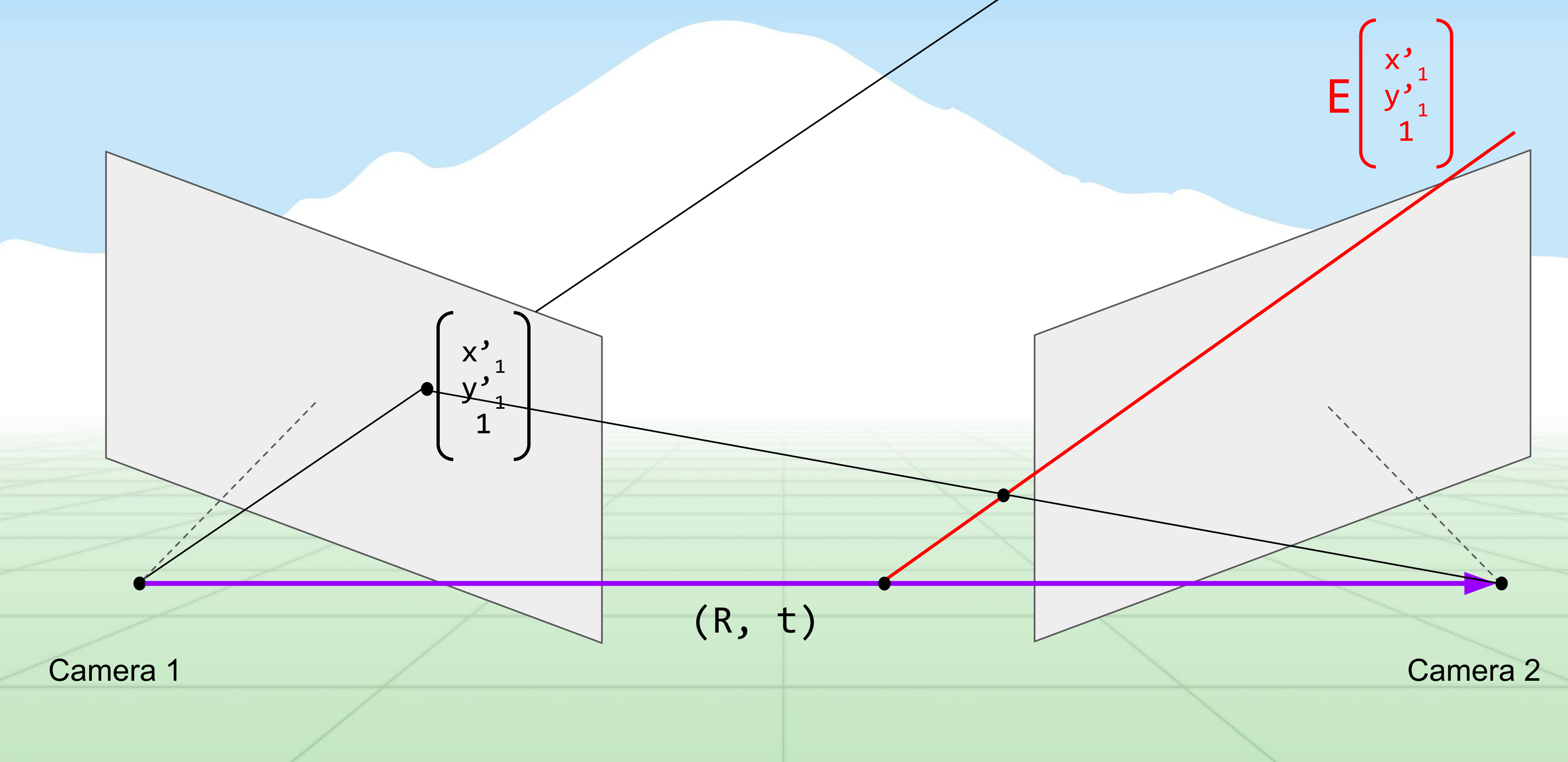
$$\frac{1}{z''_1 t_z} t \times R \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$E \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

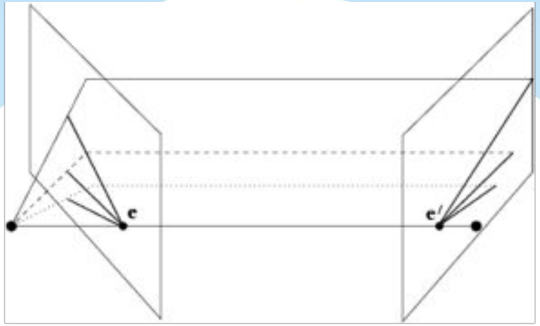
Camera

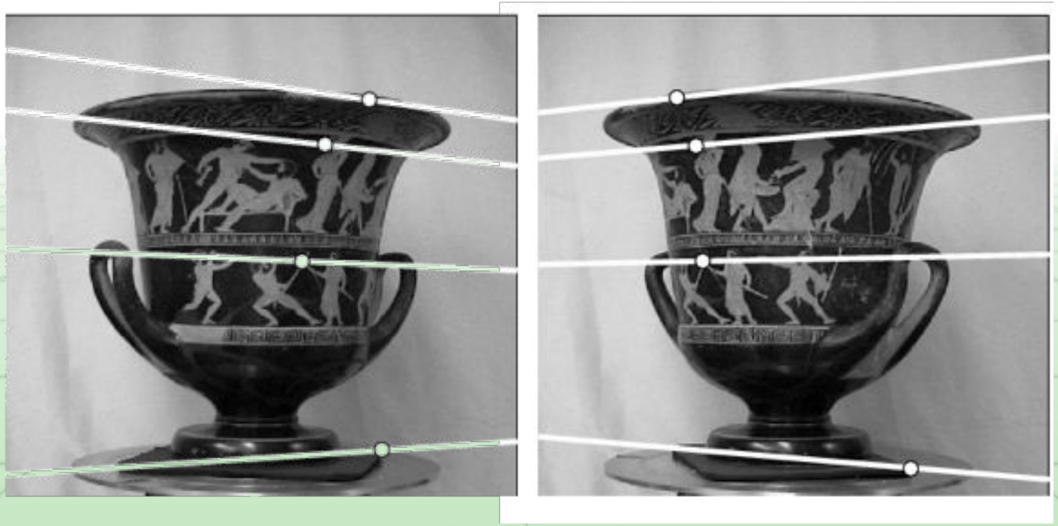
ra 2

Stereo



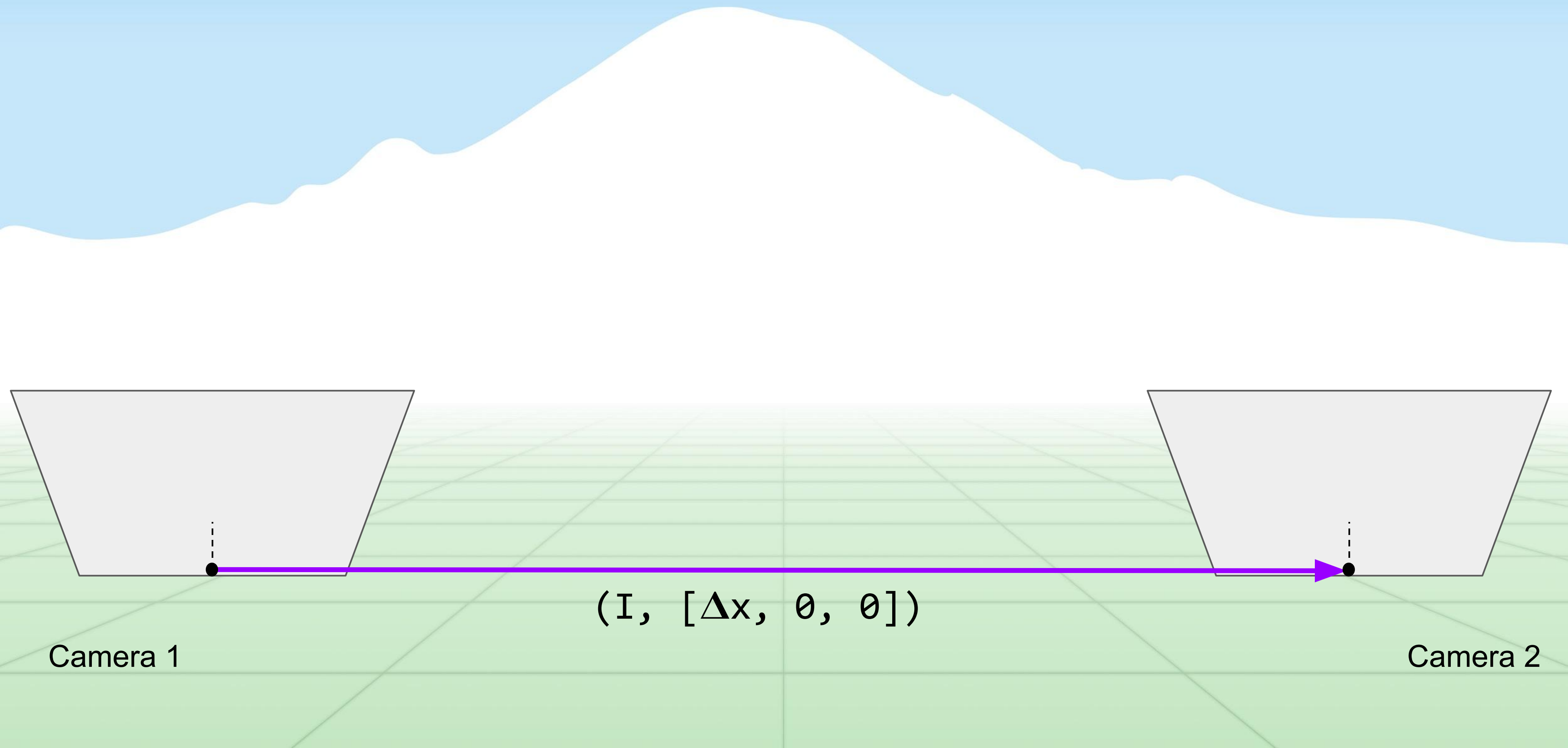
Stereo



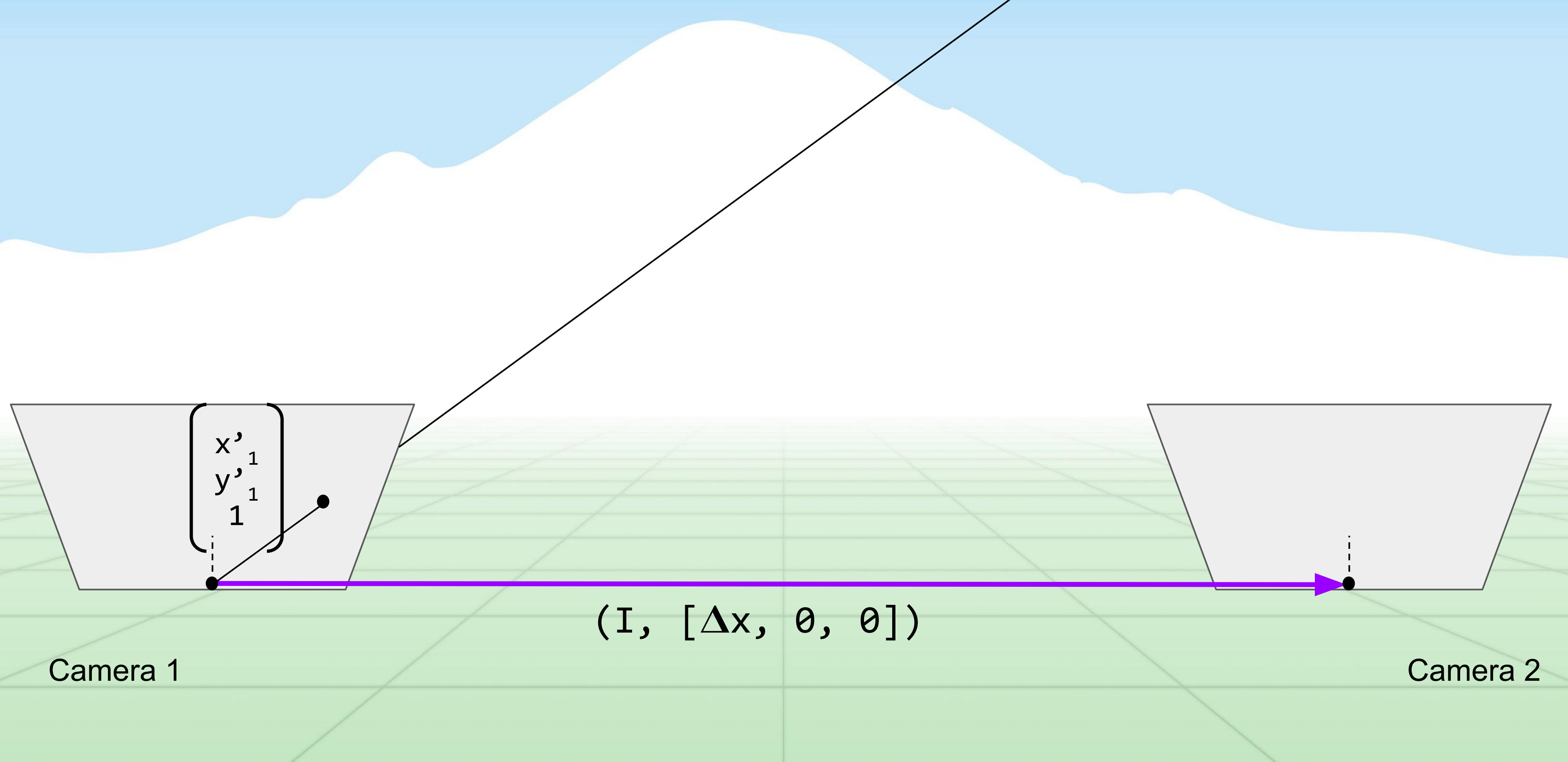


Stereo

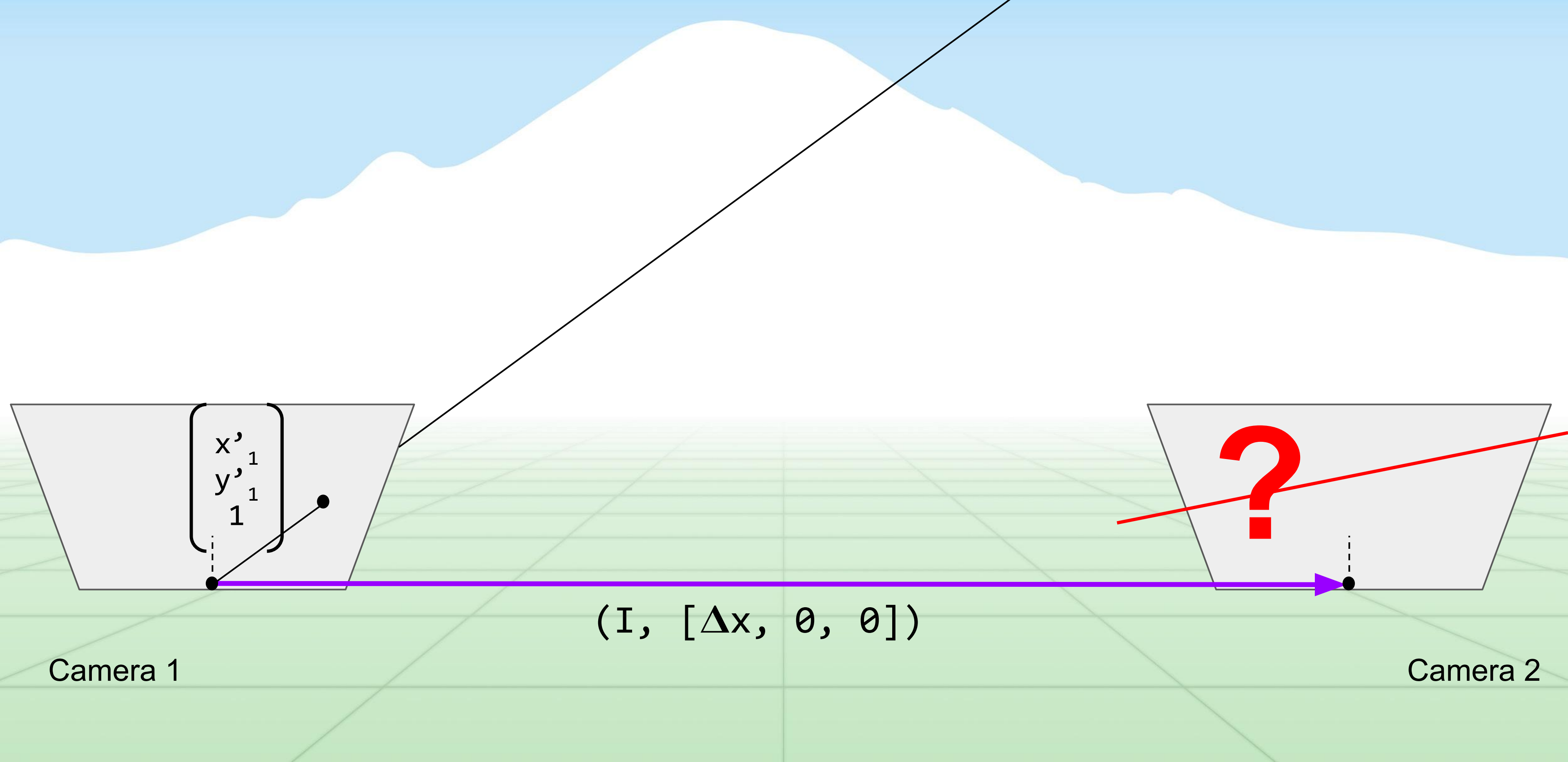
Stereo: Parallel Image Planes



Stereo: Parallel Image Planes



Stereo: Parallel Image Planes



Stereo: Parallel Image Planes

$$t \times R \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Camera

ra 2

Stereo: Parallel Image Planes

$$\begin{bmatrix} \Delta x \\ 0 \\ 0 \end{bmatrix} \times \mathrm{I} \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Camera

ra 2

Stereo: Parallel Image Planes

$$\begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} \times I \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Camera

ra 2

Stereo: Parallel Image Planes

$$\begin{bmatrix} \Delta x \\ 0 \\ 0 \end{bmatrix} \times I \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\begin{bmatrix} \Delta x \\ 0 \\ 0 \end{bmatrix} \times \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ -\Delta x \\ \Delta x y'_1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Camera

ra 2

Stereo: Parallel Image Planes

$$\begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} \times \mathrm{I} \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$\begin{pmatrix} \Delta x \\ 0 \\ 0 \end{pmatrix} \times \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

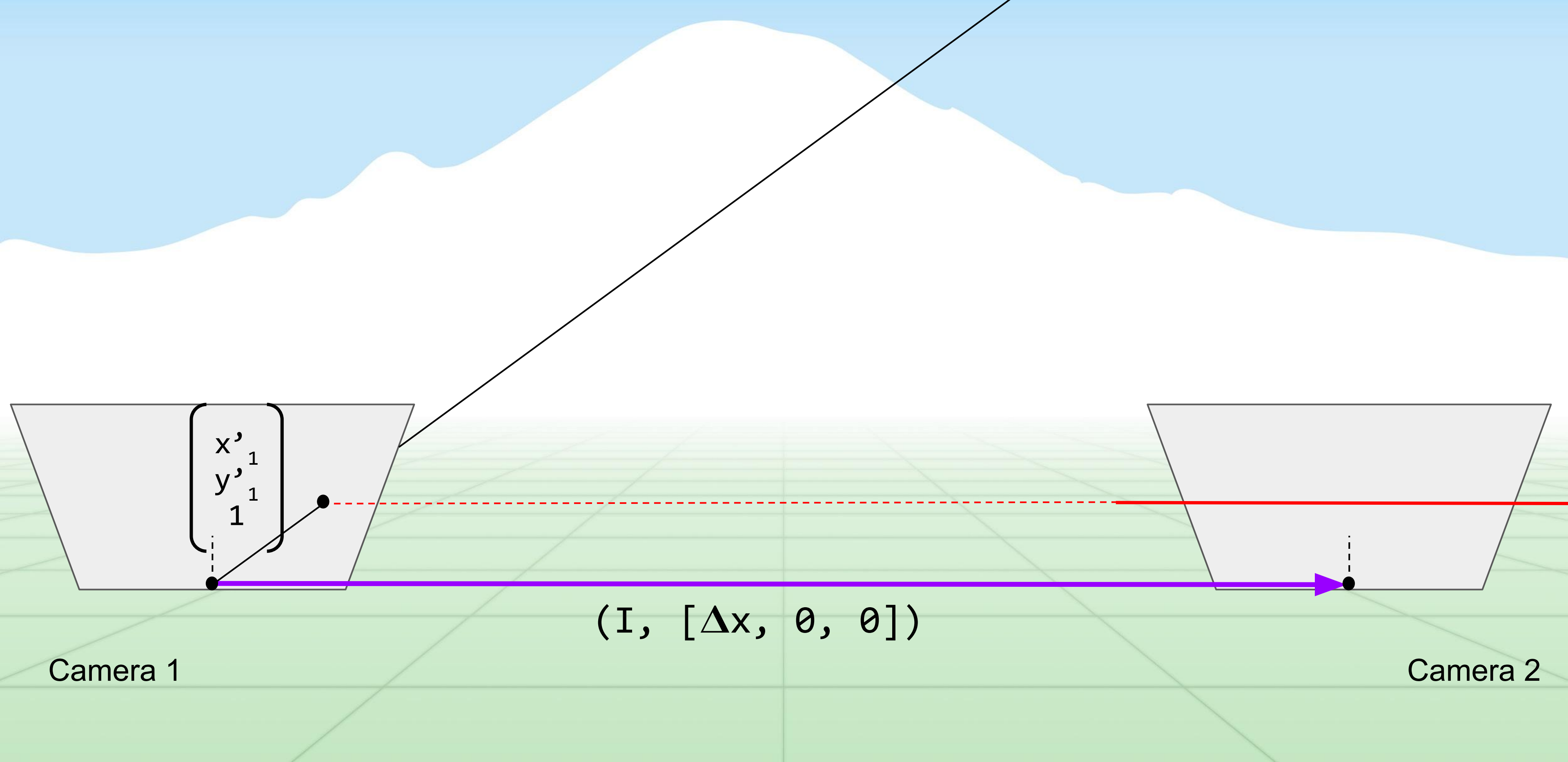
$$\begin{pmatrix} 0 \\ -\Delta x \\ \Delta x y'_1 \end{pmatrix} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$$(0)x + (-\Delta x)y + (\Delta x y'_1) = 0 \longrightarrow y = y'_1$$

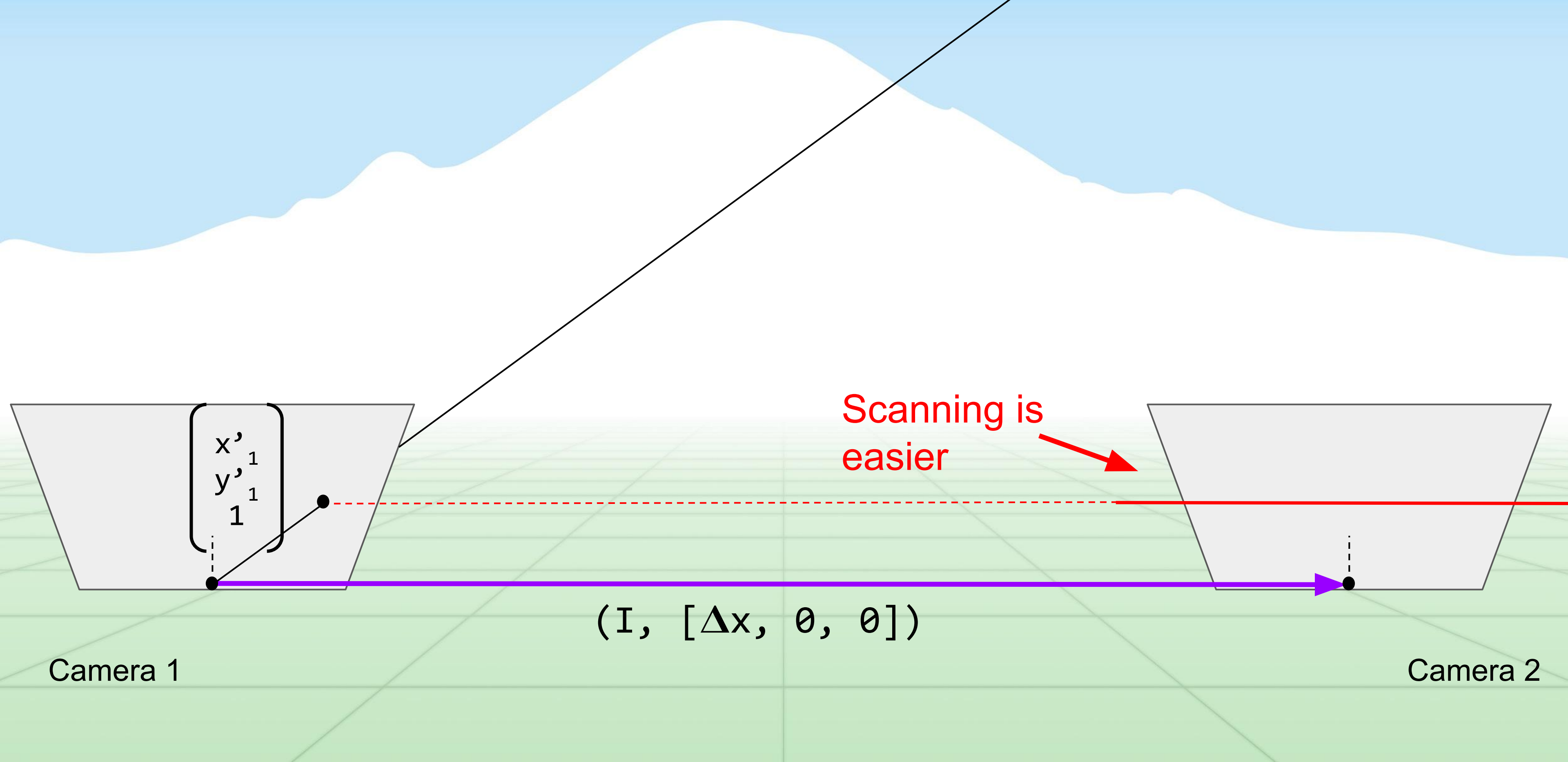
Camera

ra 2

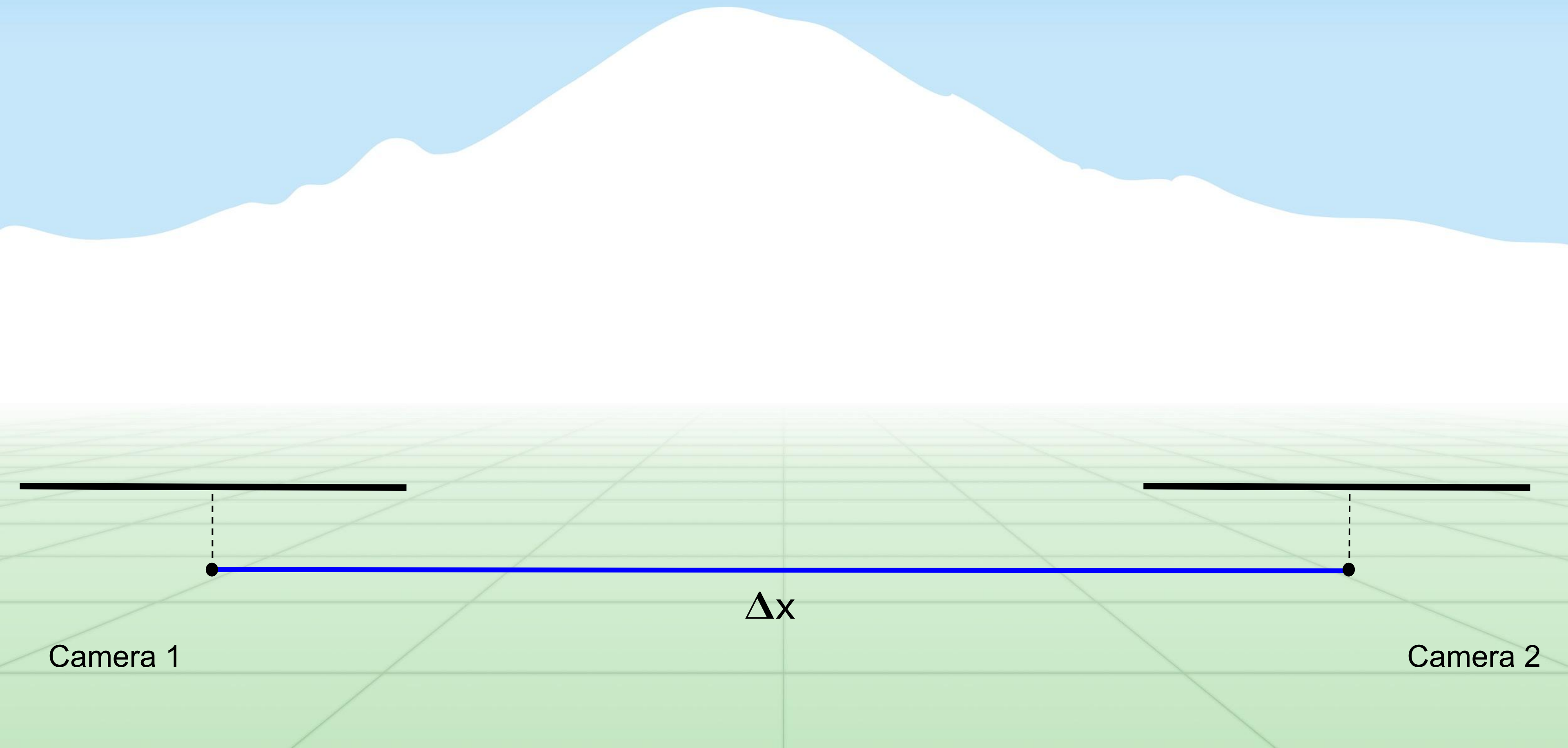
Stereo: Parallel Image Planes



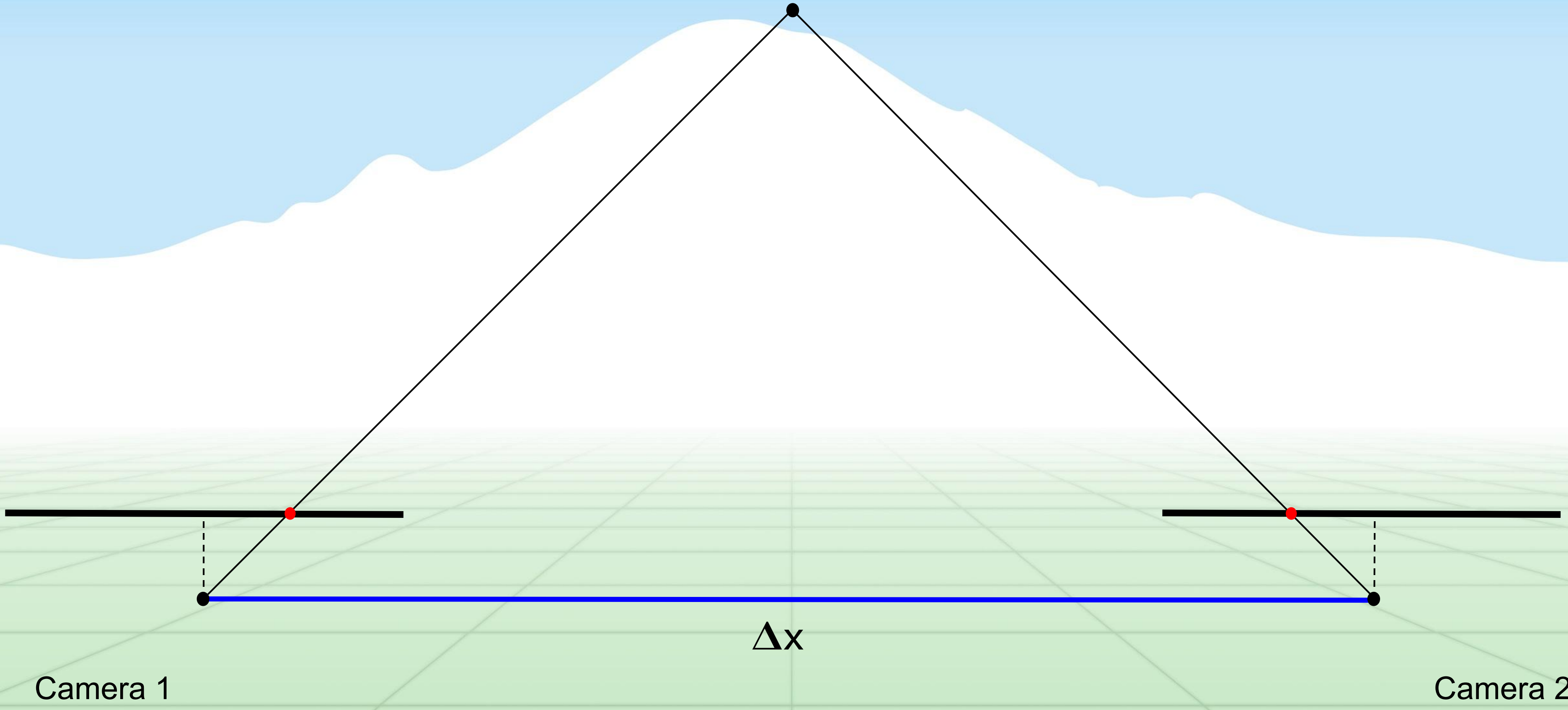
Stereo: Parallel Image Planes



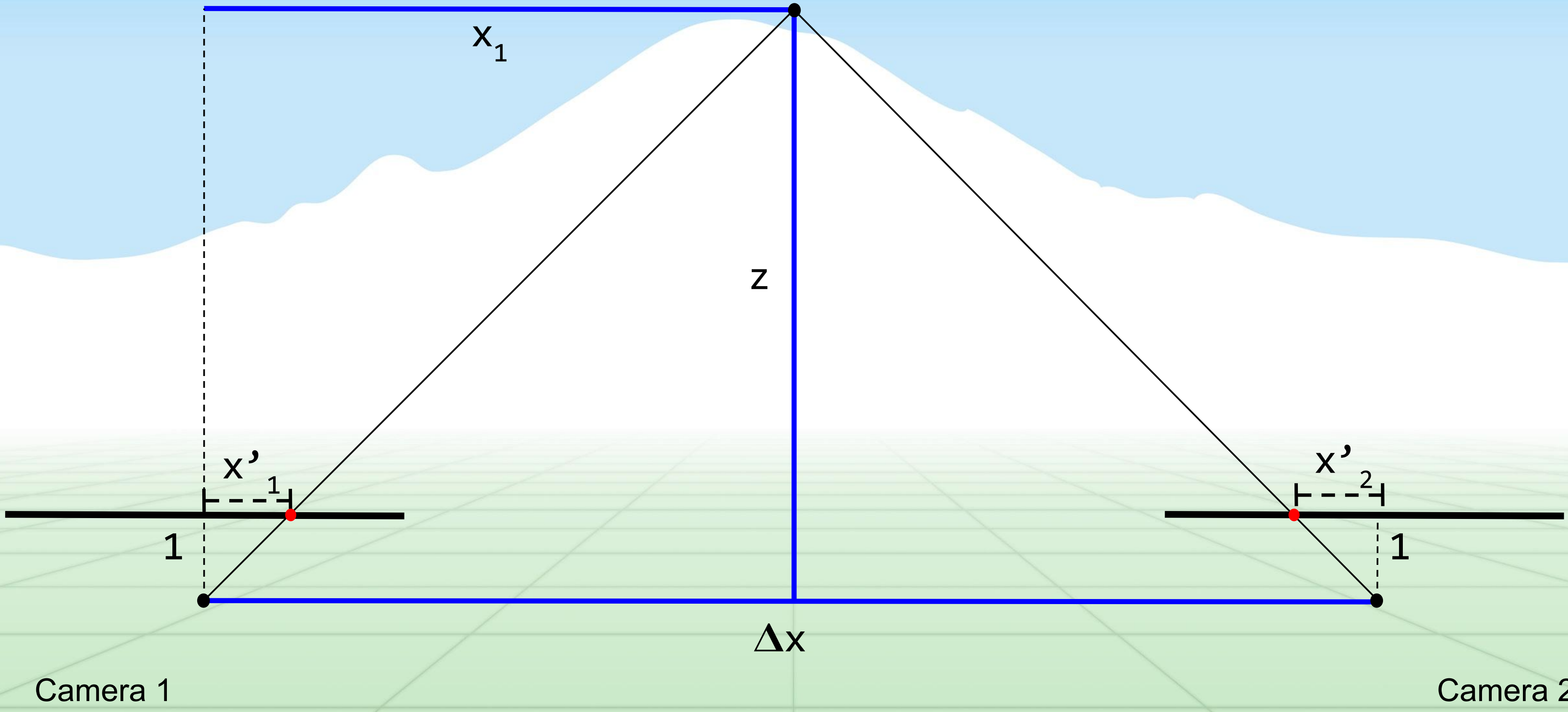
Stereo: Parallel Image Planes



Stereo: Parallel Image Planes

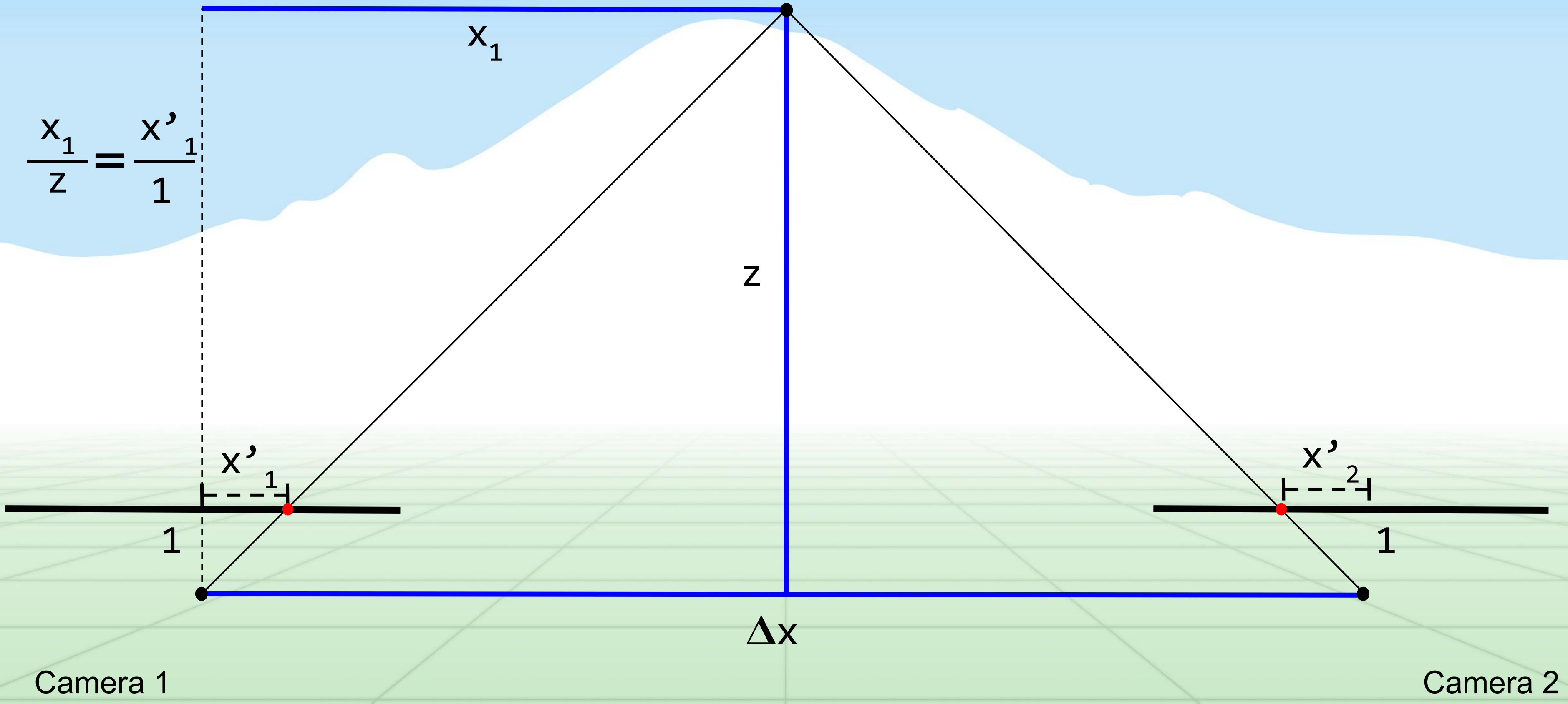


Stereo: Parallel Image Planes

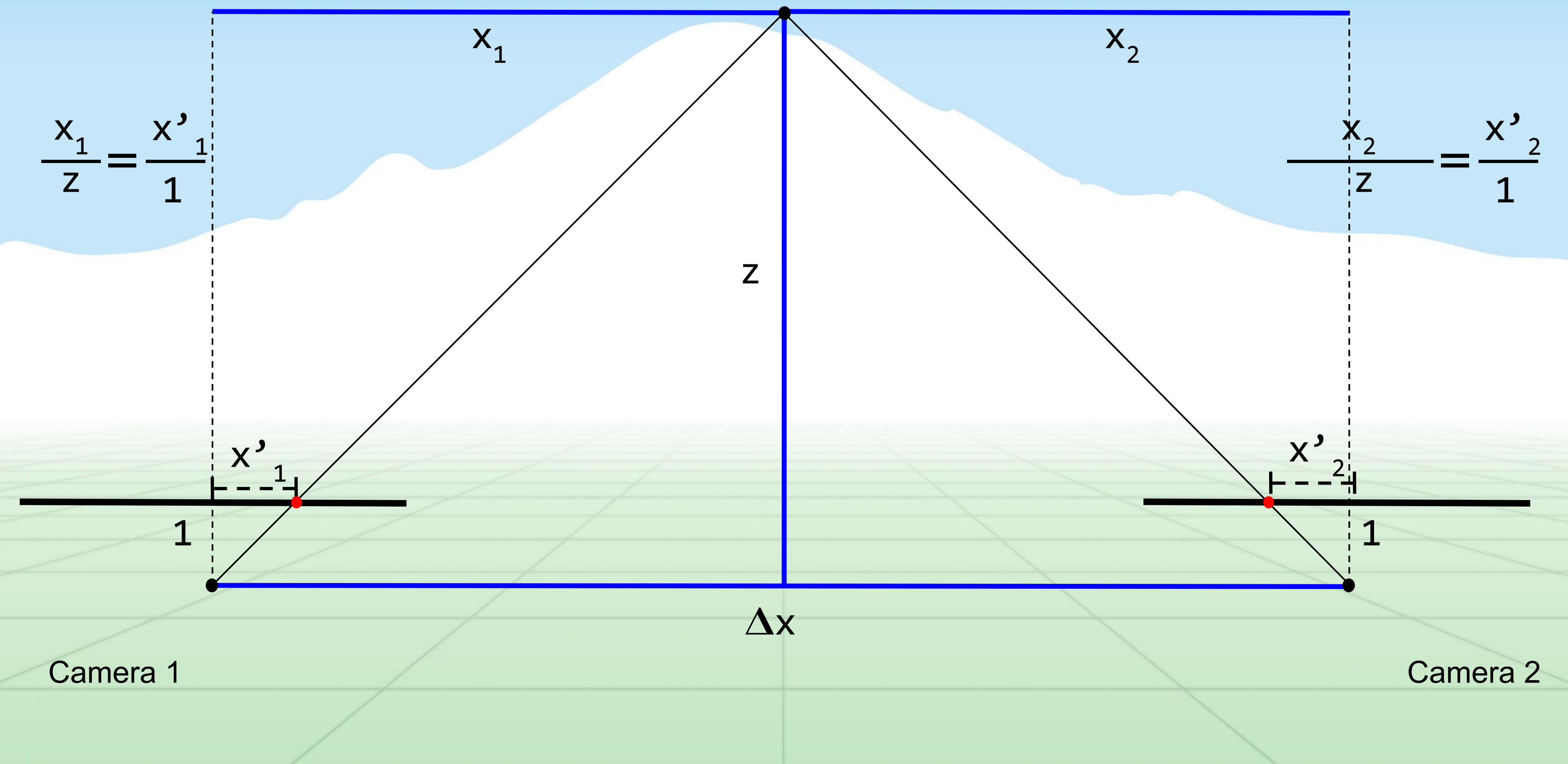


Stereo: Parallel Image Planes

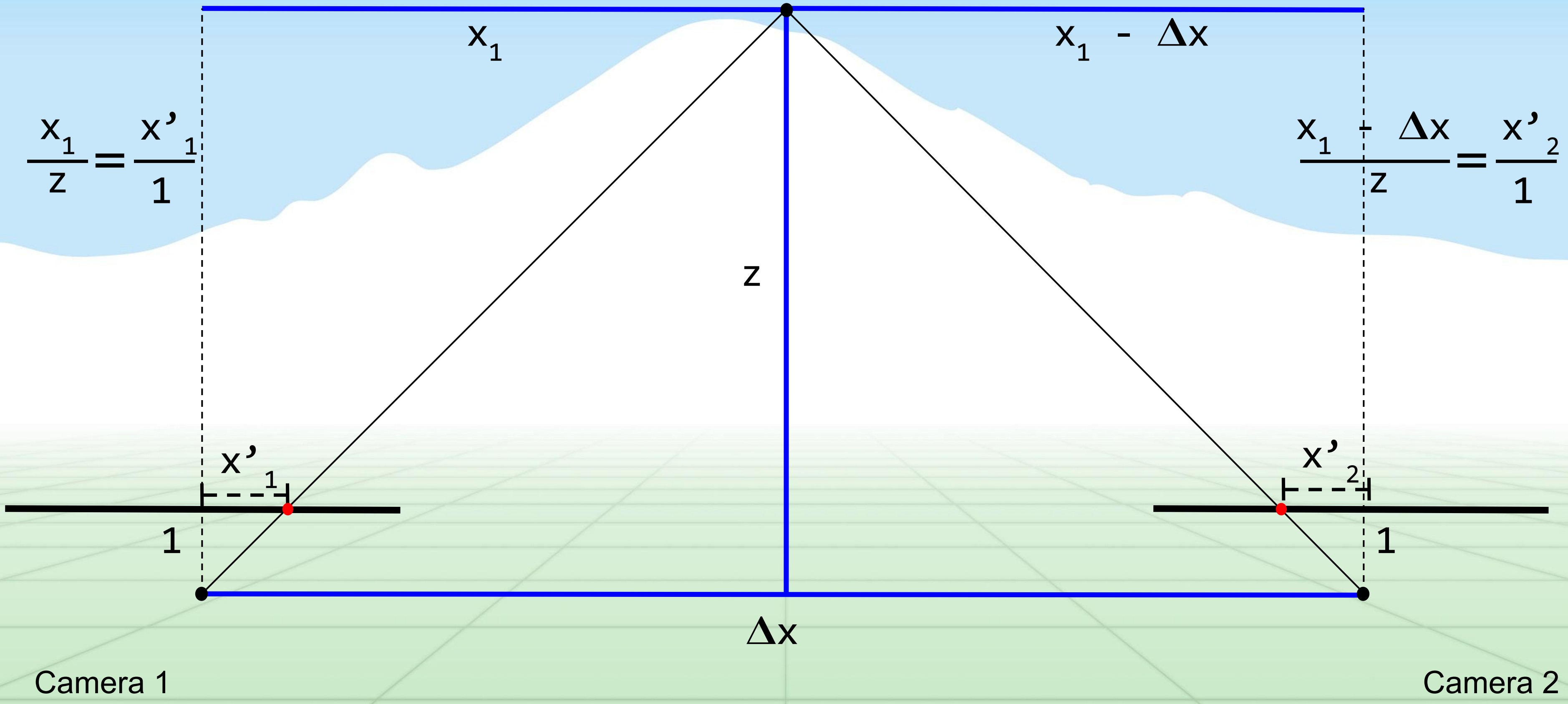
$$\frac{x_1}{z} = \frac{x'_1}{1}$$



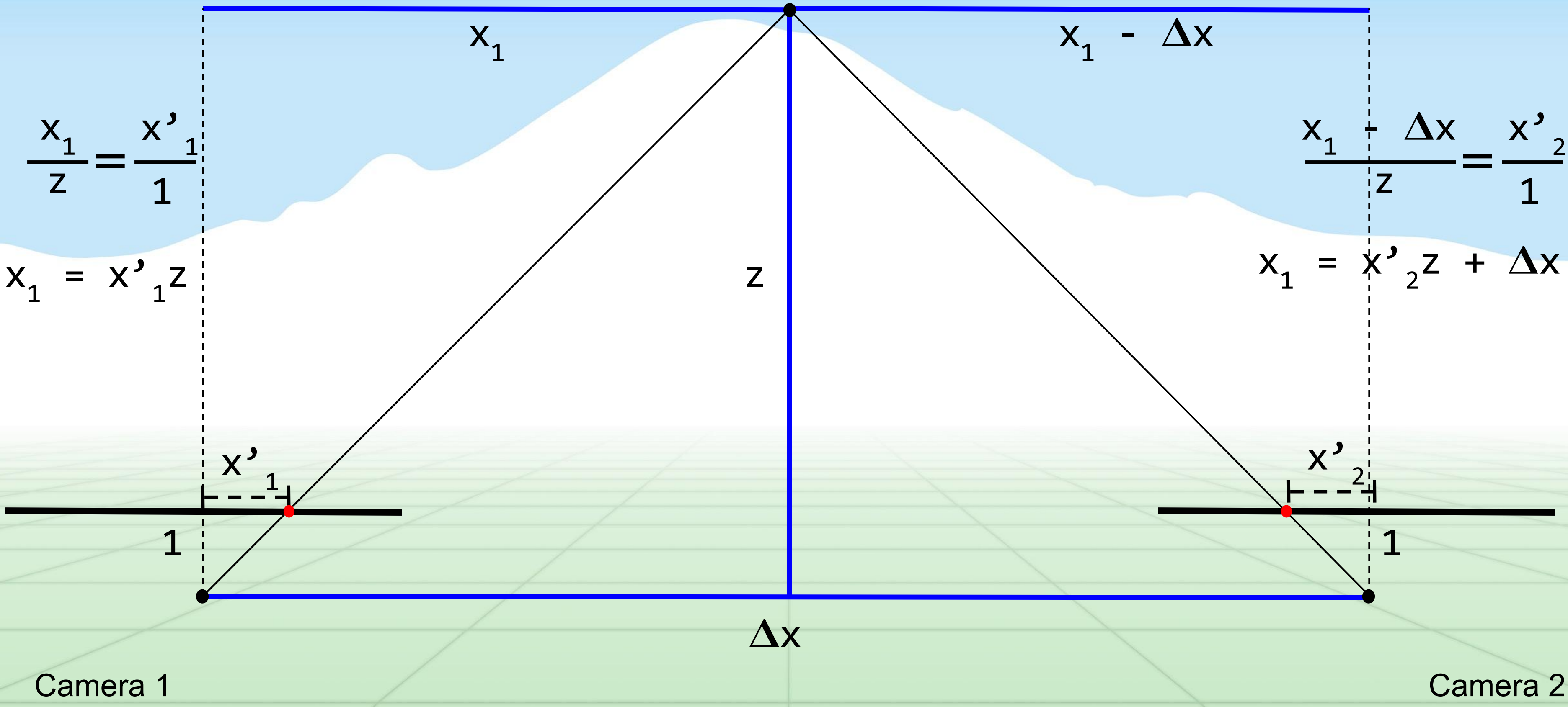
Stereo: Parallel Image Planes



Stereo: Parallel Image Planes



Stereo: Parallel Image Planes



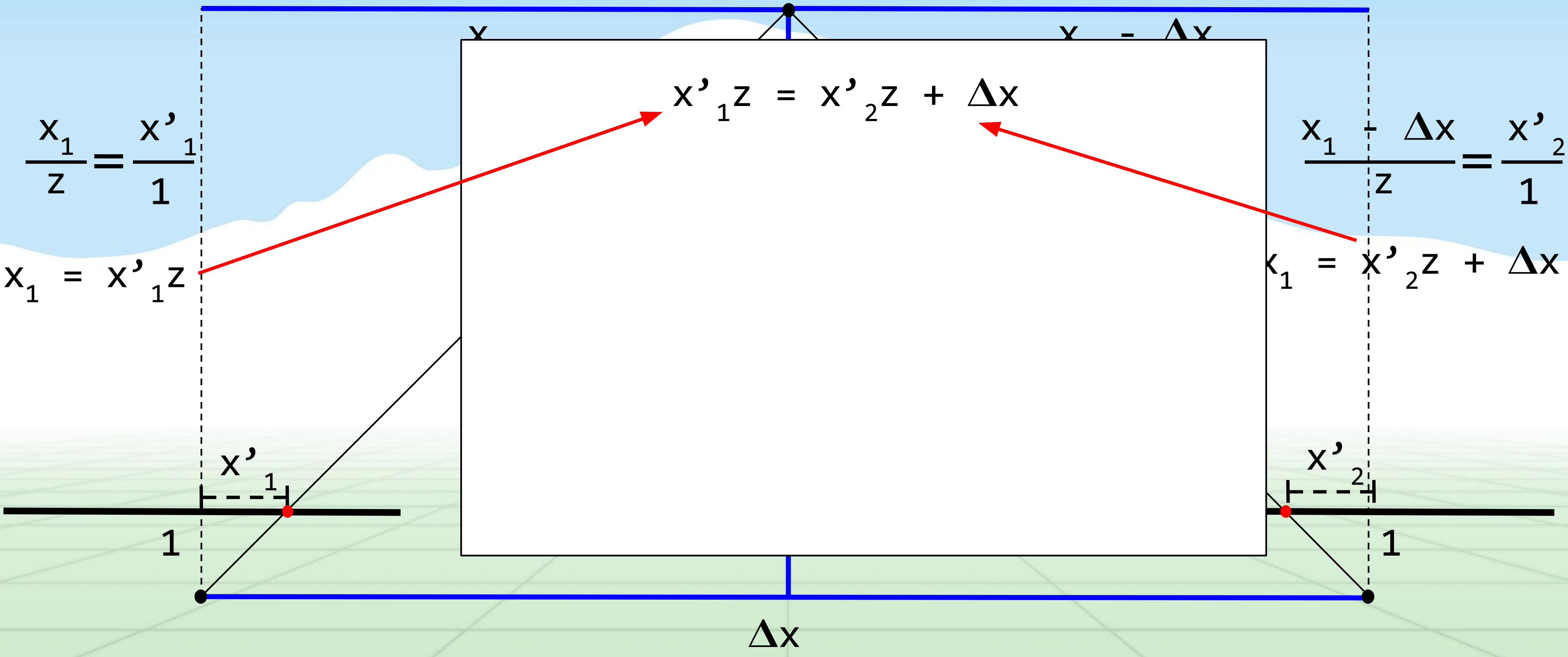
$$\frac{x_1}{z} = \frac{x'_1}{1}$$

$$x_1 = x'_1 z$$

$$\frac{x_1 - \Delta x}{z} = \frac{x'_2}{1}$$

$$x_1 = x'_2 z + \Delta x$$

Stereo: Parallel Image Planes



$$\frac{x_1}{z} = \frac{x'_1}{1}$$

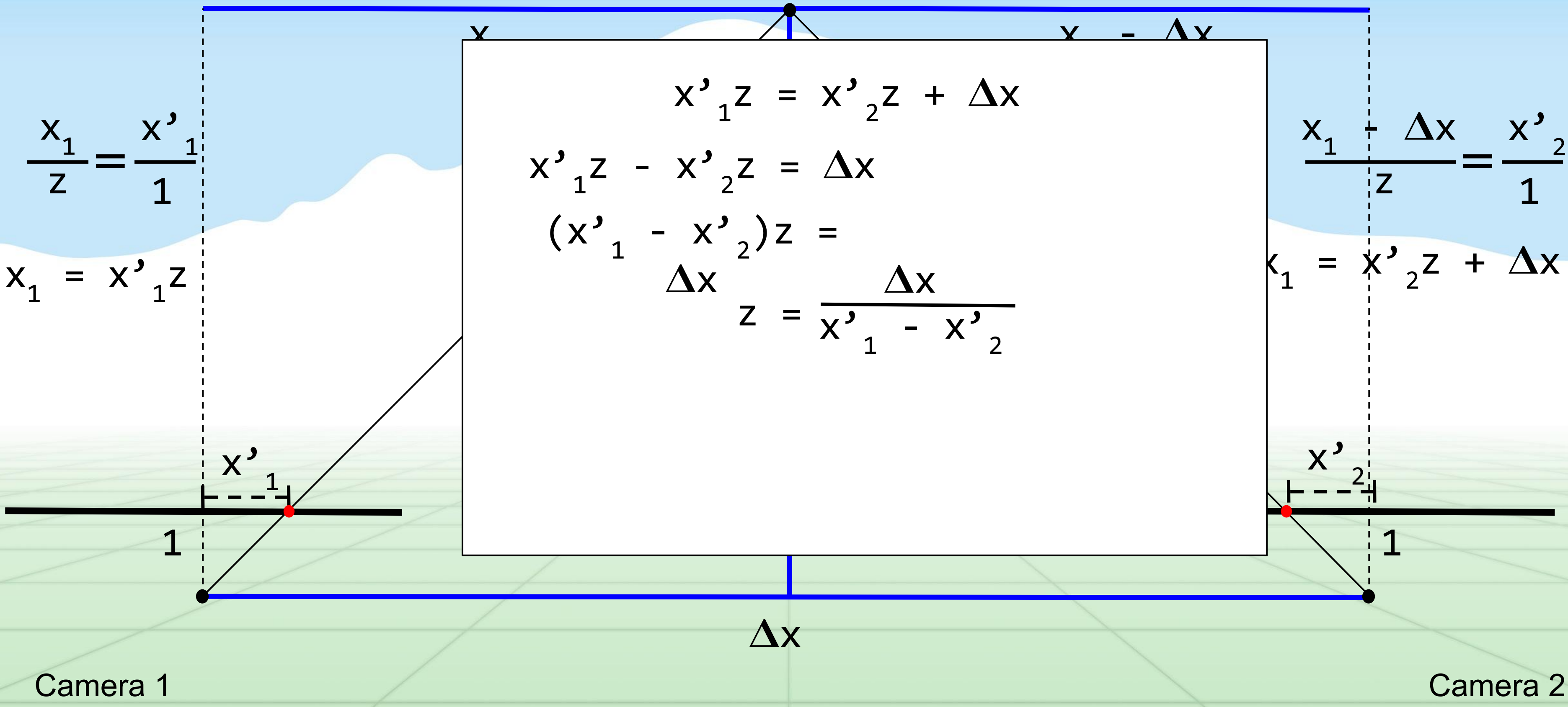
$$x_1 = x'_1 z$$

$$\frac{x_1 - \Delta x}{z} = \frac{x'_2}{1}$$

$$x_1 = x'_2 z + \Delta x$$

$$x'_1 z = x'_2 z + \Delta x$$

Stereo: Parallel Image Planes



$$\frac{x_1}{z} = \frac{x'_1}{1}$$

$$x_1 = x'_1 z$$

$$\frac{x_1 - \Delta x}{z} = \frac{x'_2}{1}$$

$$x_1 = x'_2 z + \Delta x$$

$$x'_1 z = x'_2 z + \Delta x$$

$$x'_1 z - x'_2 z = \Delta x$$

$$(x'_1 - x'_2) z = \Delta x$$

$$z = \frac{\Delta x}{x'_1 - x'_2}$$

Stereo: Parallel Image Planes

<u>Algorithm (assuming same intrinsics)</u>

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
```

Stereo: Parallel Image Planes

<u>Algorithm (assuming same intrinsics)</u>

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
    For every pixel i,j in I_1:
        f_ij = get_feature_descriptor(I_1, i, j)
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
    For every pixel i,j in I_1:
        f_ij = get_feature_descriptor(I_1, i, j)
        best_score <- INFINITY
        best_i <- NaN
        For i' in columns(I_2):
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
    For every pixel i,j in I_1:
        f_ij = get_feature_descriptor(I_1, i, j)
        best_score <- INFINITY
        best_i <- NaN
        For i’ in columns(I_2):
            f_i’j = get_feature_descriptor(I_2, i’, j)
            score = score_feature_match(f_ij, f_i’j)
            If score < best_score:
                best_score = score
                best_i = i’
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
    For every pixel i,j in I_1:
        f_ij = get_feature_descriptor(I_1, i, j)
        best_score <- INFINITY
        best_i <- NaN
        For i’ in columns(I_2):
            f_i’j = get_feature_descriptor(I_2, i’, j)
            score = score_feature_match(f_ij, f_i’j)
            If score < best_score:
                best_score = score
                best_i = i’
        depth[i, j] = fpΔx/(i - best_i)
    Return depth
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I₁, I₂, Δx, f, p):
    depth <- new_image_like(I₁)
    For every pixel i,j in I₁:
        f_ij = get_feature_descriptor(I₁, i, j)
        best_score <- INFINITY
        best_i <- NaN
        For i’ in columns(I₂):
            f_i’j = get_feature_descriptor(I₂, i’, j)
            score = score_feature_match(f_ij, f_i’j)
            If score < best_score:
                best_score = score
                best_i = i’
        depth[i, j] = fpΔx/(i - best_i)
    Return depth
```

Parallel image planes make things easy!

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth(I_1, I_2, Δx, f, p):
    depth <- new_image_like(I_1)
    For every pixel i,j in I_1:
        f_ij = get_feature_descriptor(I_1, i, j)
        best_score <- INFINITY
        best_i <- NaN
        For i’ in columns(I_2):
            f_i’j = get_feature_descriptor(I_2, i’, j)
            score = score_feature_match(f_ij, f_i’j)
            If score < best_score:
                best_score = score
                best_i = i’
        depth[i, j] = fpΔx/(i - best_i)
    Return depth
```

What are these?

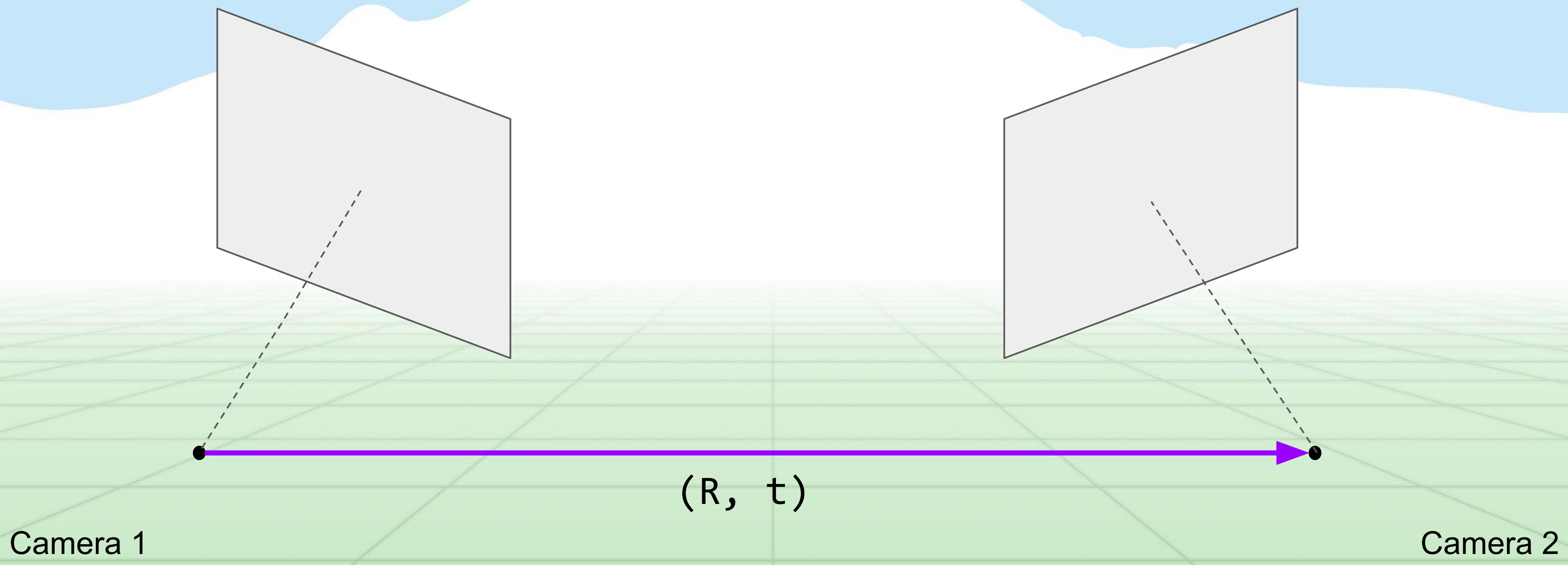
Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

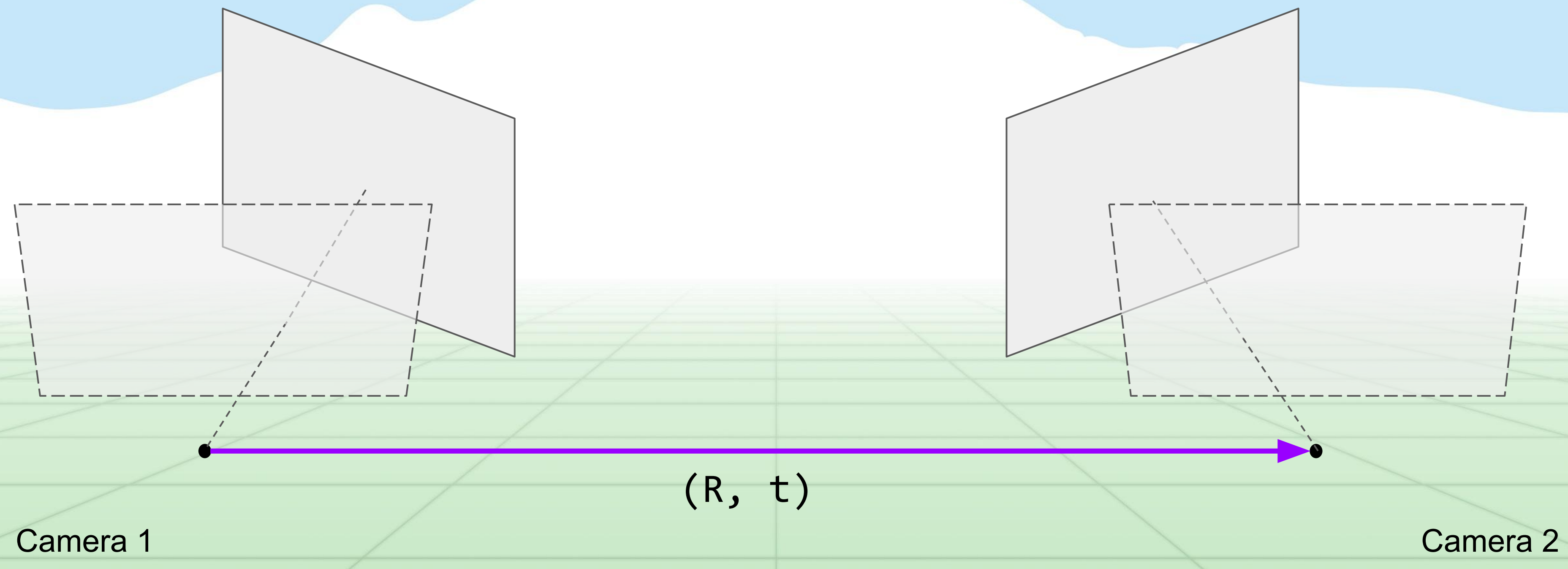

```
compu
    de
    Fo
                                          I1, i, j)

                                          tor(I2, i', j)
                                          tch(fij, fi'j)
            If score < best_score:
                best_score = score
                best_i = i'
        depth[i, j] = fpΔx/(i - best_i)
    Return depth
```

What are these?

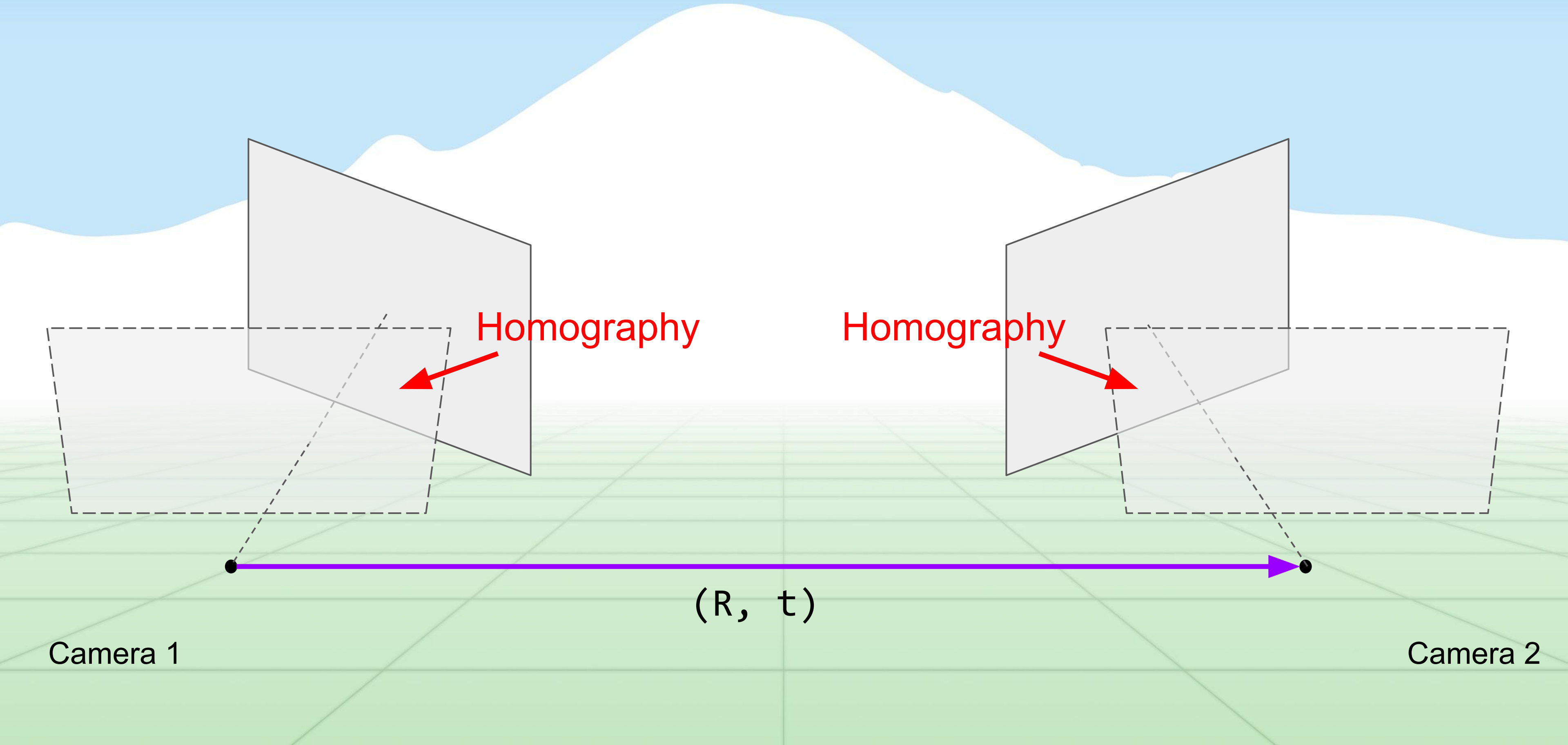
Stereo: Computing Homographies



Stereo: Computing Homographies



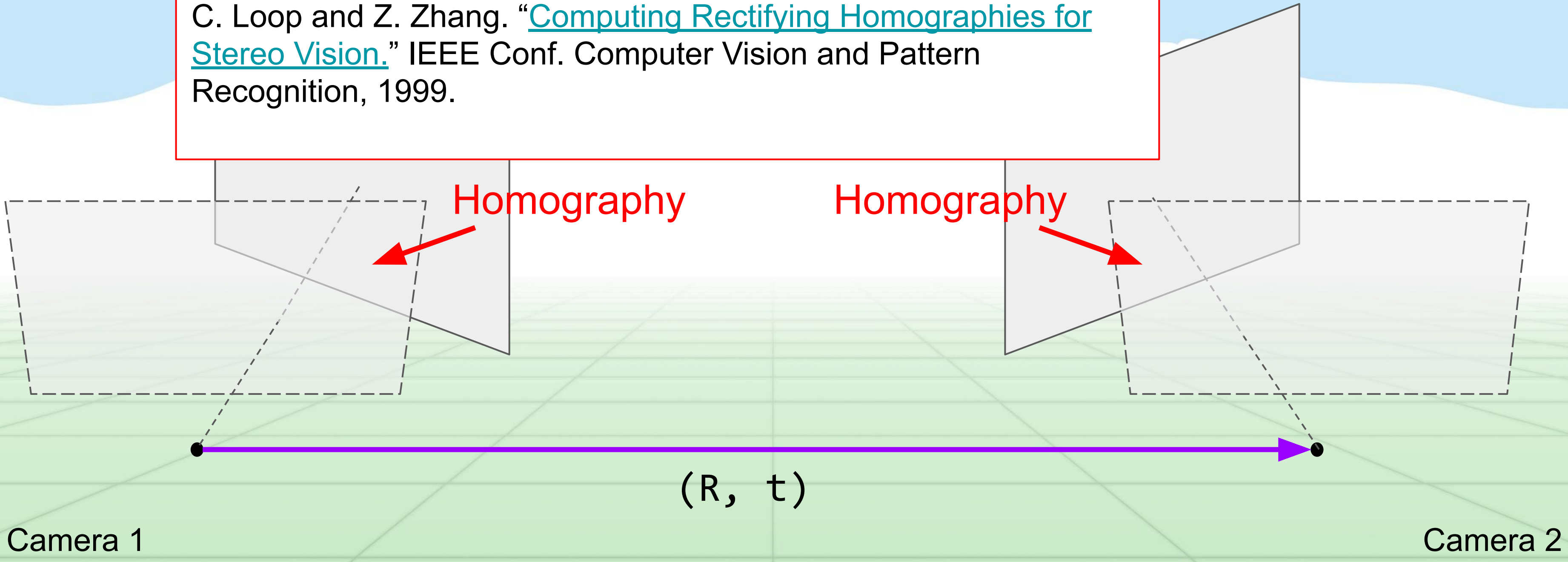
Stereo: Computing Homographies



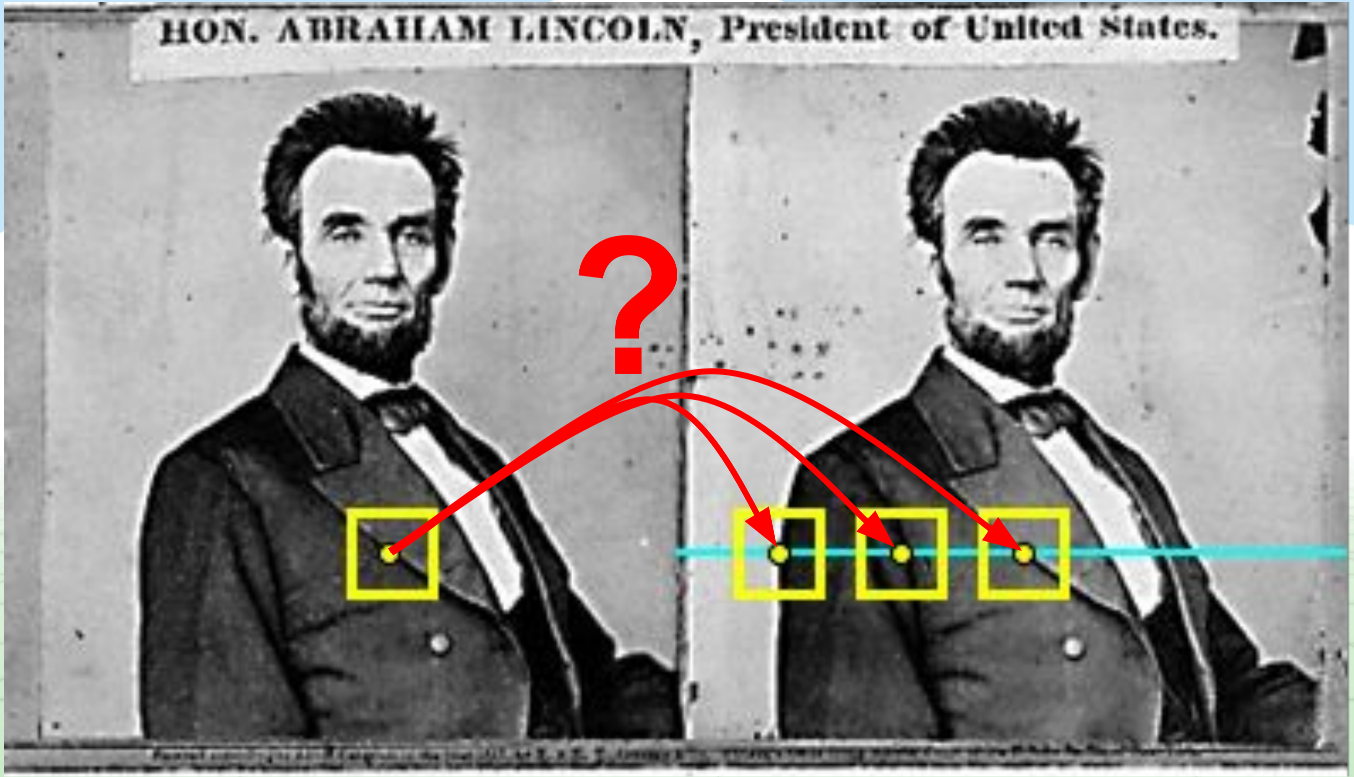
Stereo: Computing Homographies

Stereo image rectification

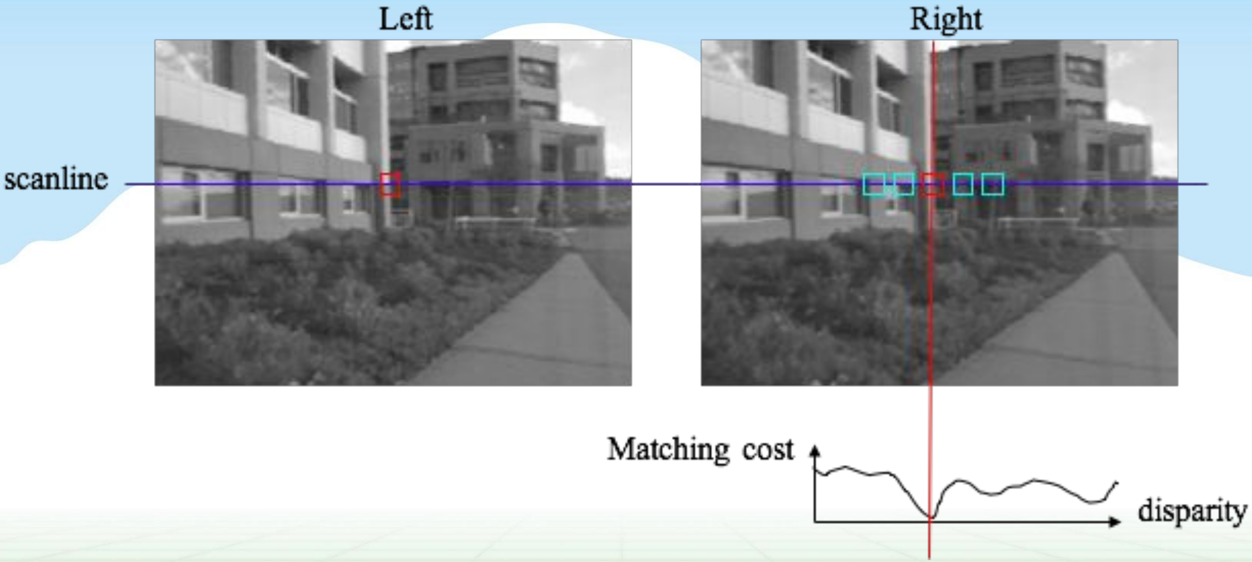
C. Loop and Z. Zhang. "Computing Rectifying Homographies for Stereo Vision." IEEE Conf. Computer Vision and Pattern Recognition, 1999.



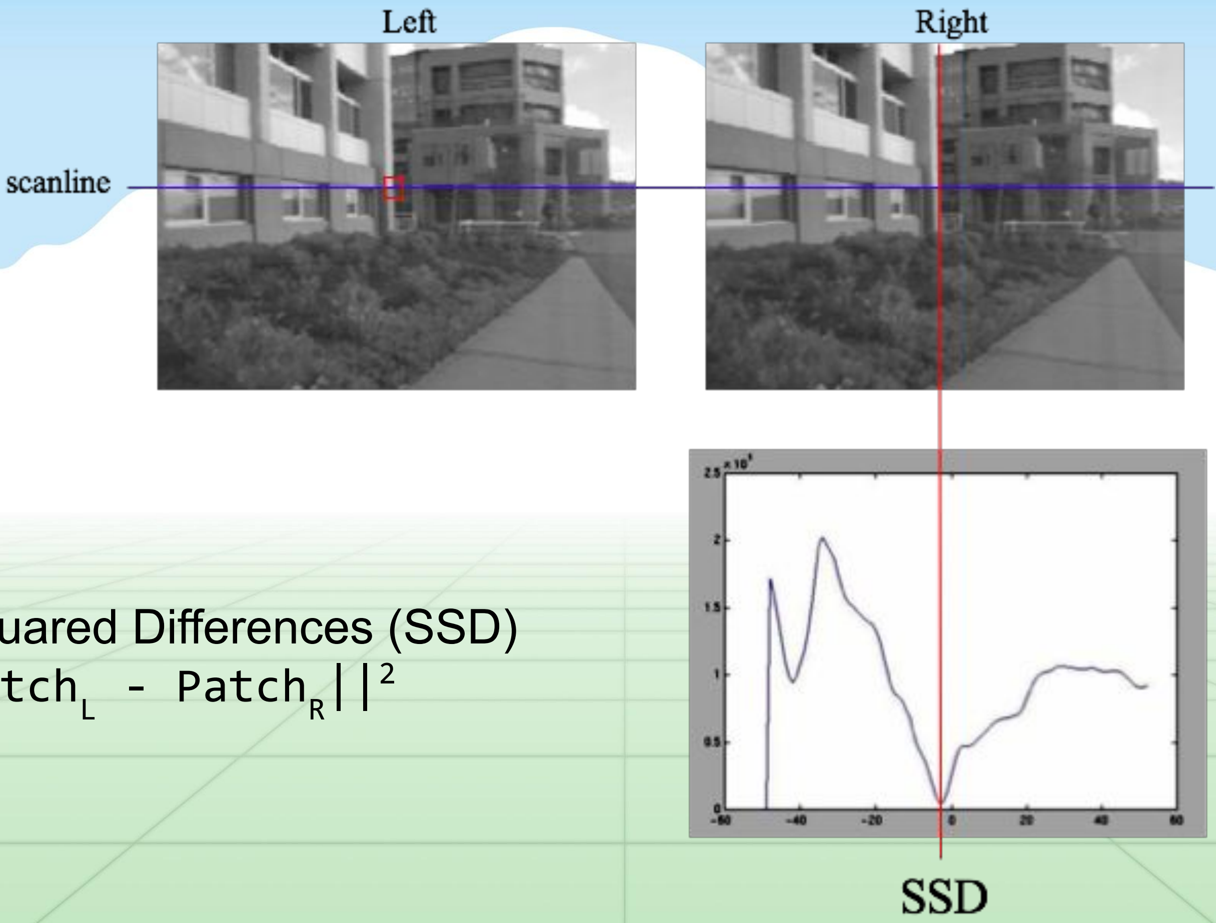
Feature Matching



Feature Matching



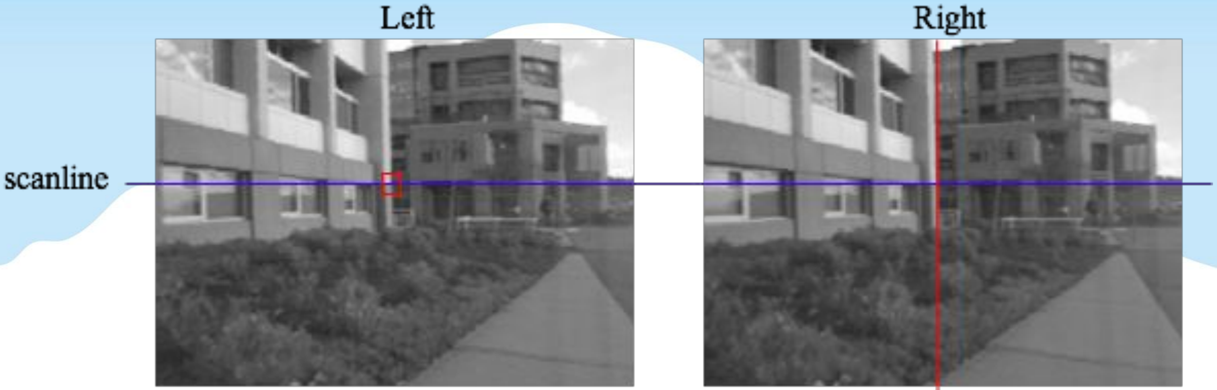
Feature Matching



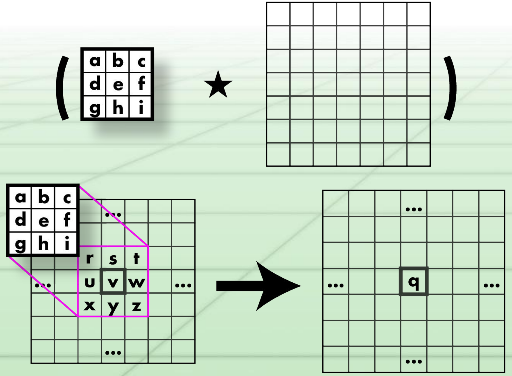
Sum of Squared Differences (SSD)

$$||Patch_L - Patch_R||^2$$

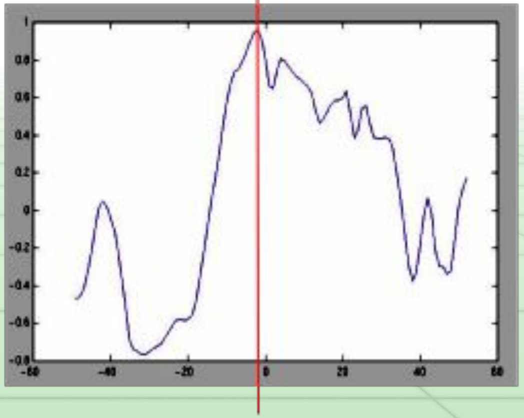
Feature Matching



Cross-Correlation



q = a×r + b×s + c×t + d×u + e×v + f×w + g×x + h×y + i×z

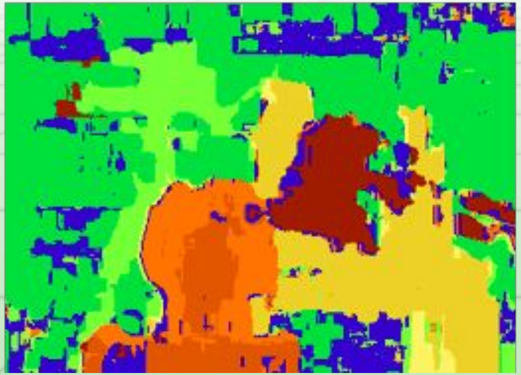


Norm. corr

Results with window search

Data

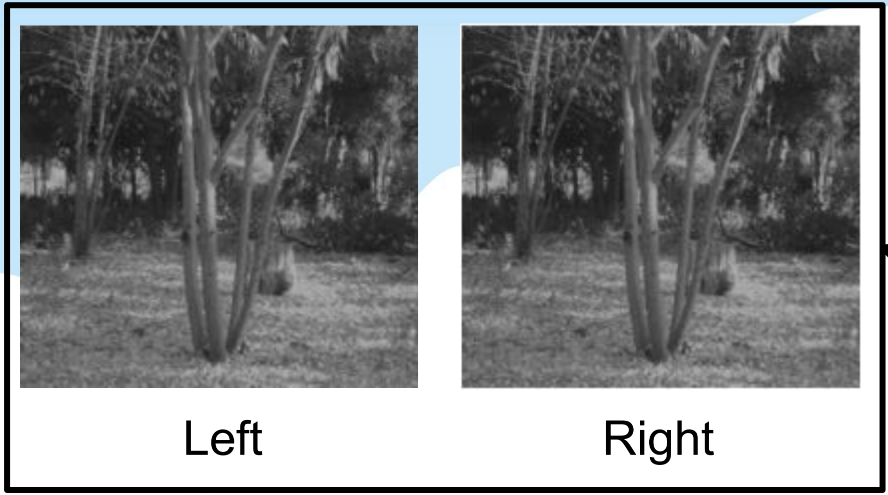
Window-based matching

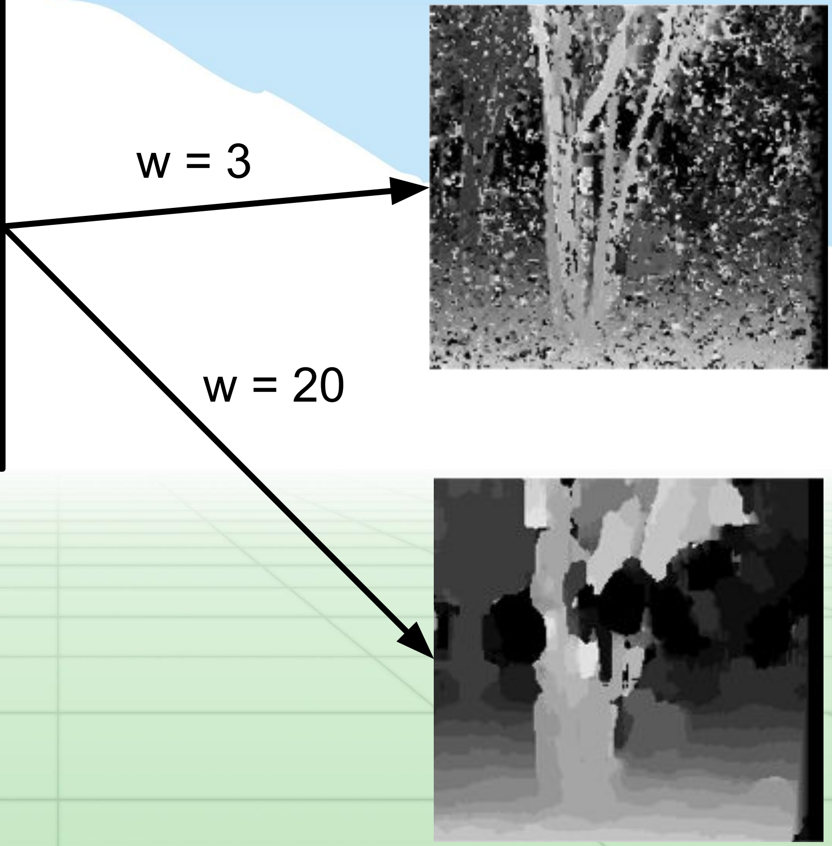


Ground truth

Results: Mercedes Magic Body Control

Effect of Window Size





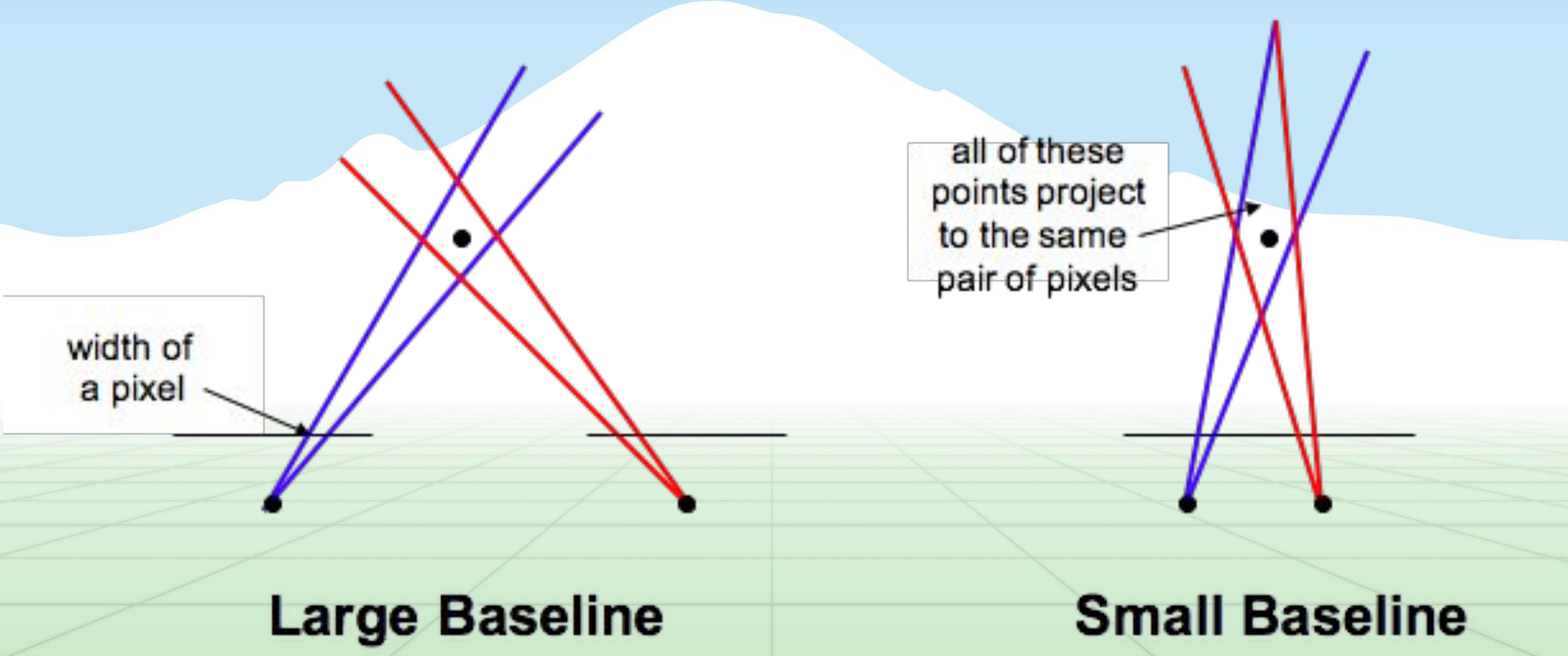
<u>Small window:</u>

- \+ Good precision, more detail
- \- Sensitive to noise

<u>Large window:</u>

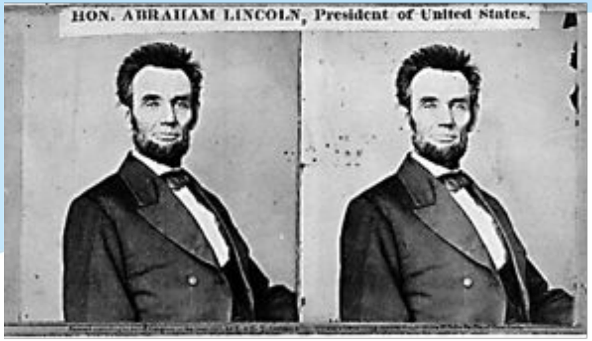
- \+ Robust to noise
- \- Reduced precision, less detail

Effect of baseline size



Failure of Correspondence Search

Failure of Correspondence Search



Textureless surfaces

Occlusions, repetition

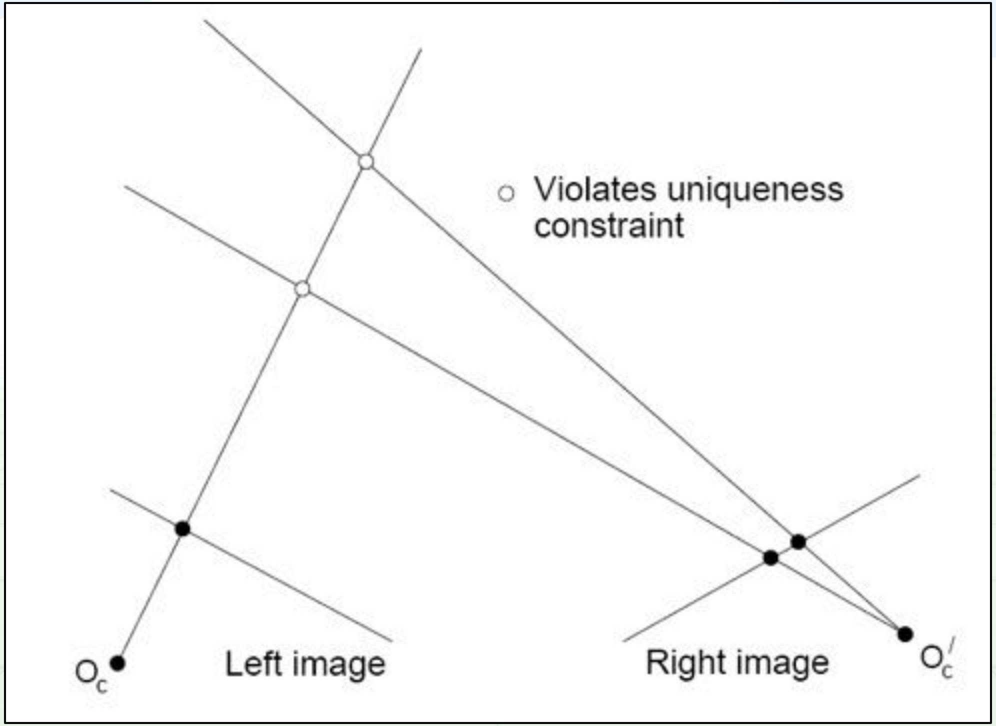
Non-Lambertian surfaces, specularities

How can we improve the matching?

Stereo Constraints

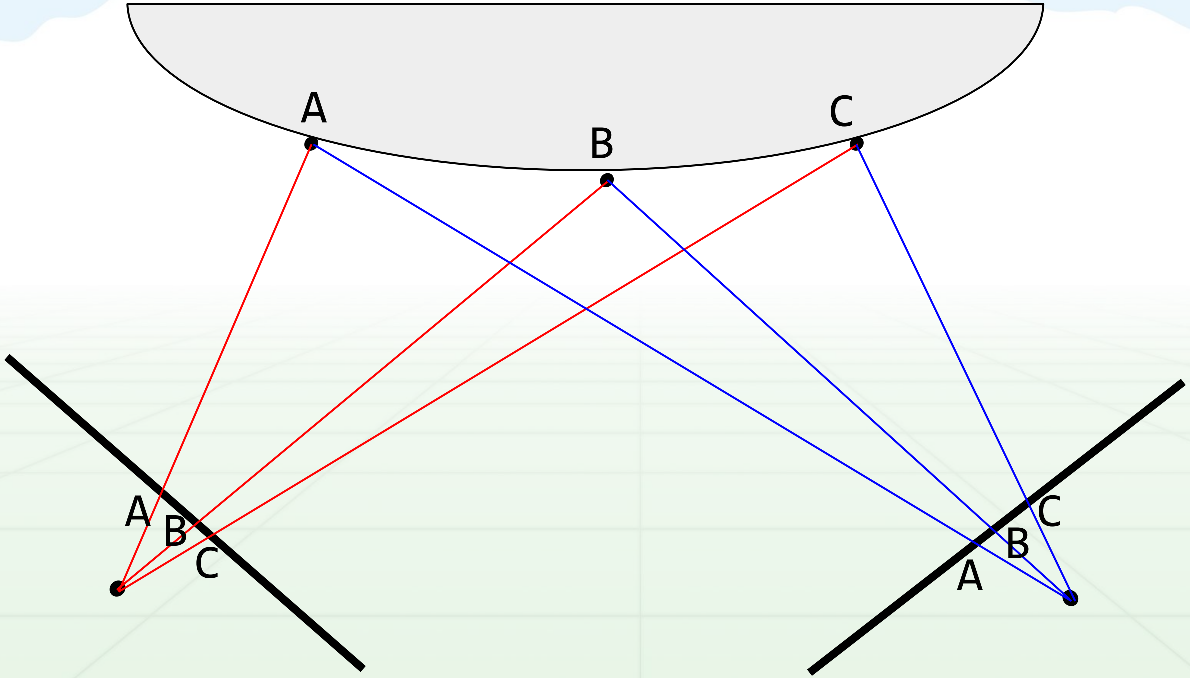
Stereo Constraints

- <u>Uniqueness:</u> For any point in one image, there should be at most one matching point in the other image



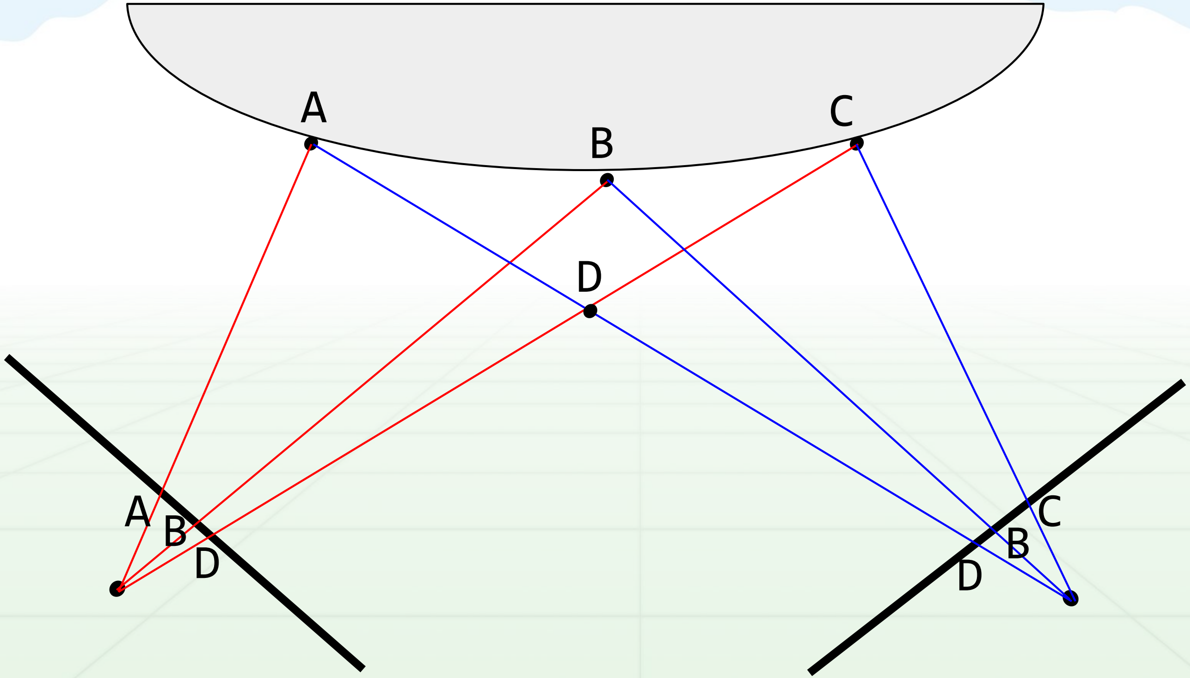
Stereo Constraints

- Uniqueness: For any point in one image, there should be at most one matching point in the other image
- Ordering: Corresponding points should be in the same order in both views



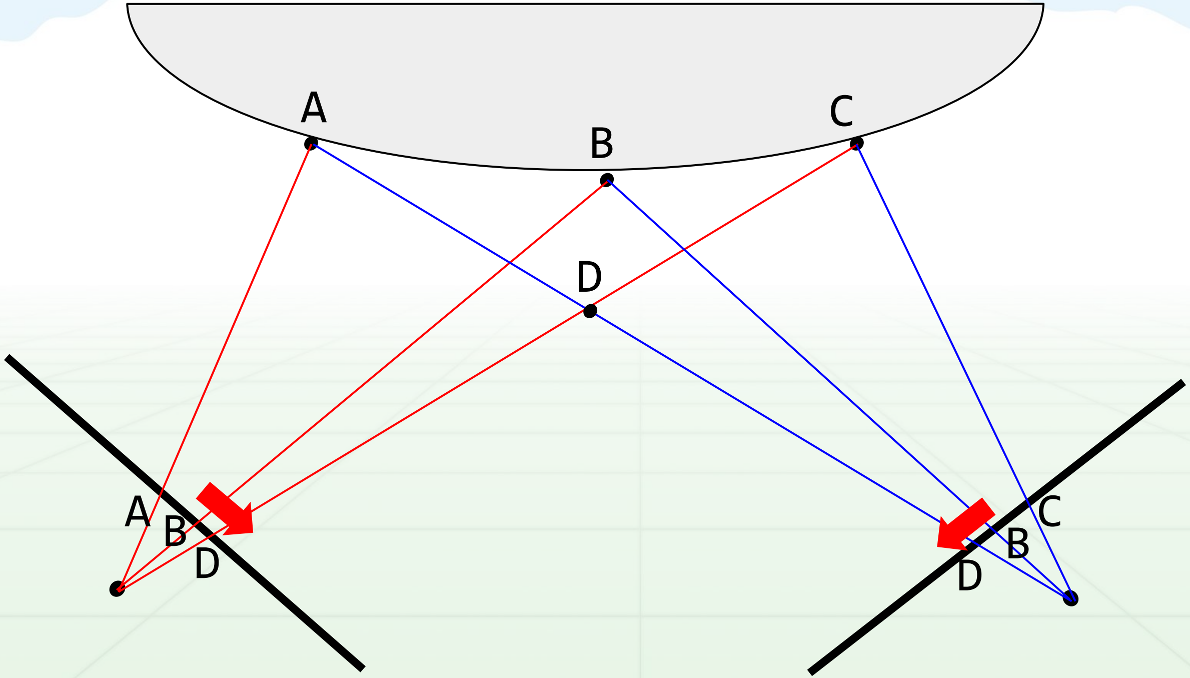
Stereo Constraints

- <u>Uniqueness:</u> For any point in one image, there should be at most one matching point in the other image
- <u>Ordering:</u> Corresponding points should be in the same order in both views



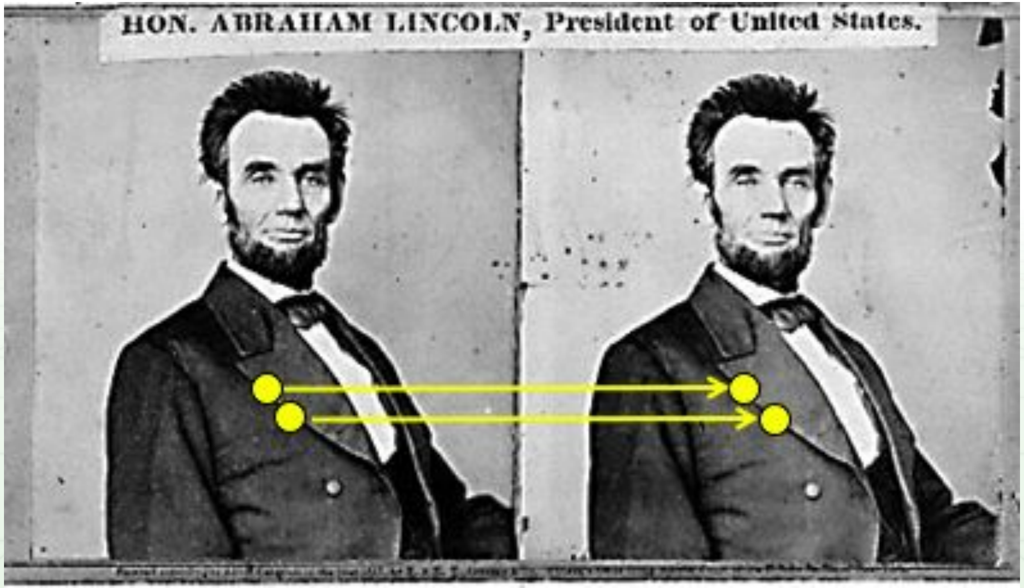
Stereo Constraints

- <u>Uniqueness:</u> For any point in one image, there should be at most one matching point in the other image
- <u>Ordering:</u> Corresponding points should be in the same order in both views



Stereo Constraints

- Uniqueness: For any point in one image, there should be at most one matching point in the other image
- Ordering: Corresponding points should be in the same order in both views
- Smoothness: We expect disparity values to change slowly (for the most part)



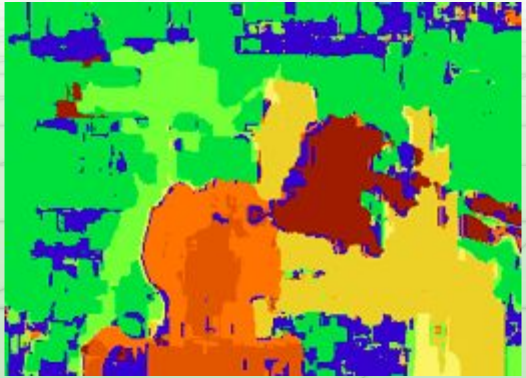
Results with graph cut

Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

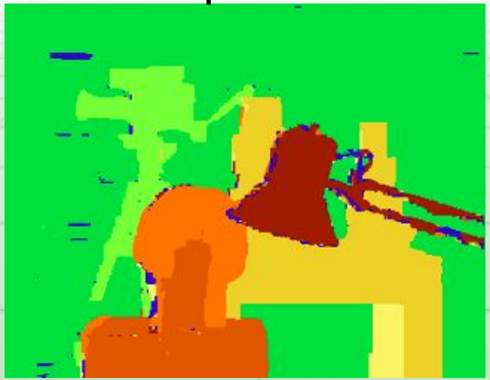
Results with graph cut

Y. Boykov, O. Veksler, and R. Zabih, Fast Approximate Energy Minimization via Graph Cuts, PAMI 2001

Before



Graph Cuts



Ground Truth