

The Ancient Secrets

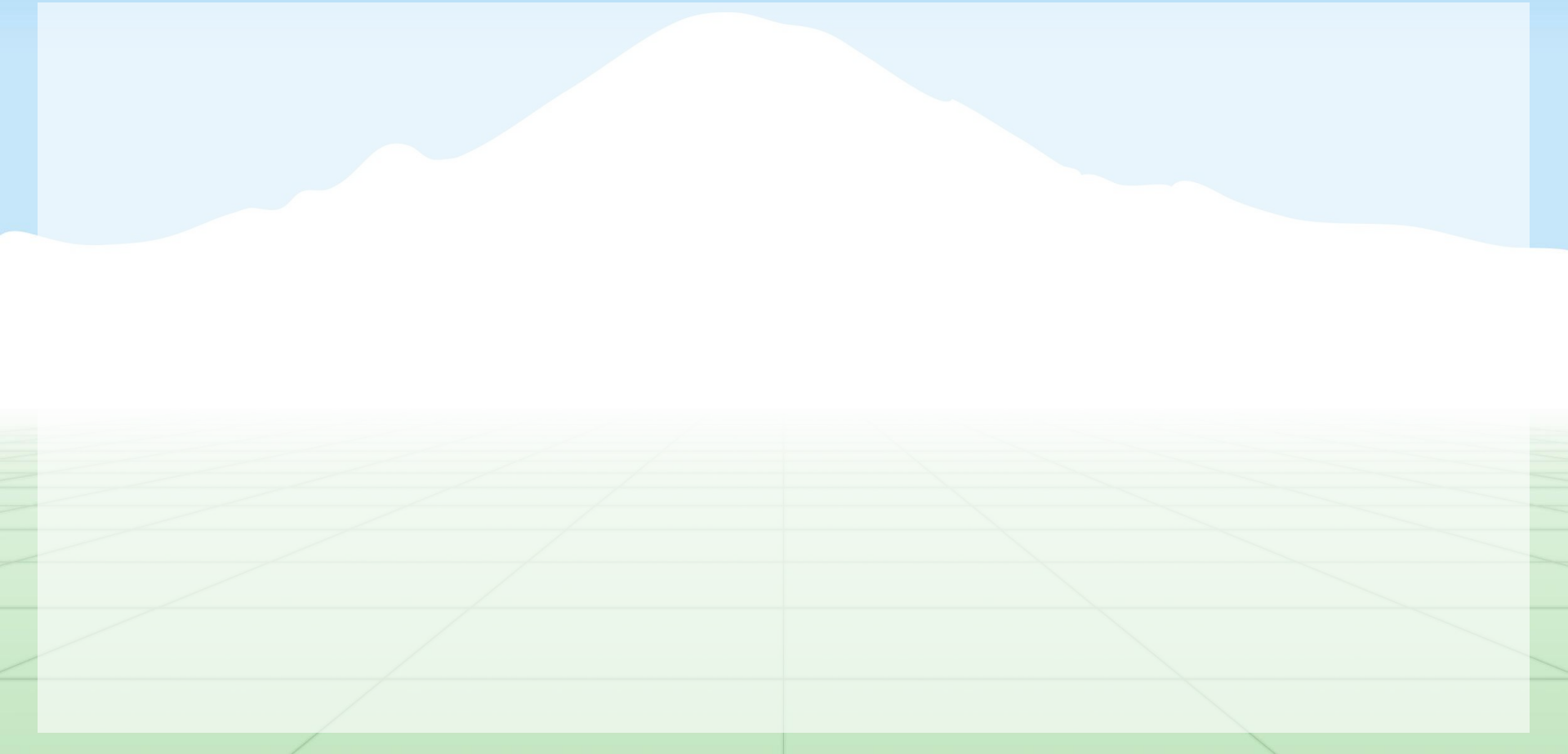


Computer Vision

Topics:

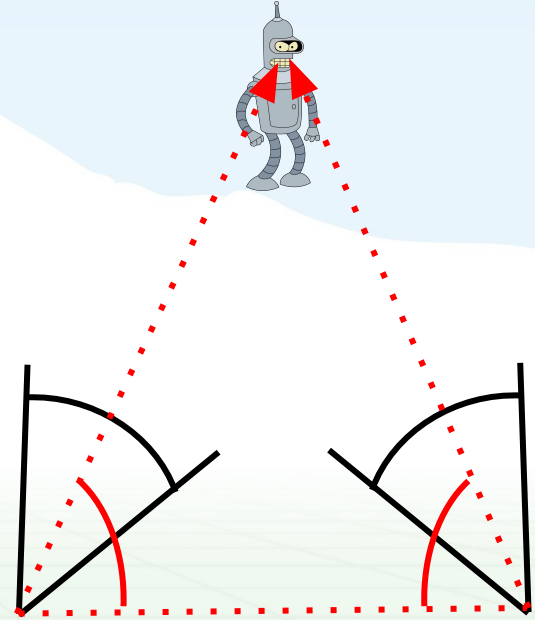
3D, Depth Perception, & Stereo

How do we perceive depth?



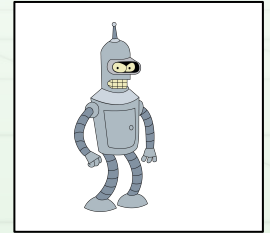
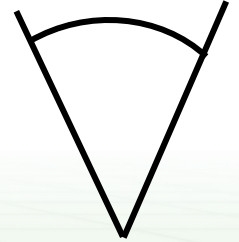
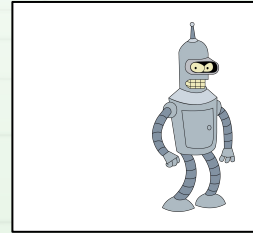
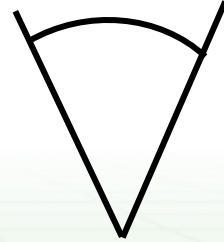
How do we perceive depth?

-Binocular Convergence



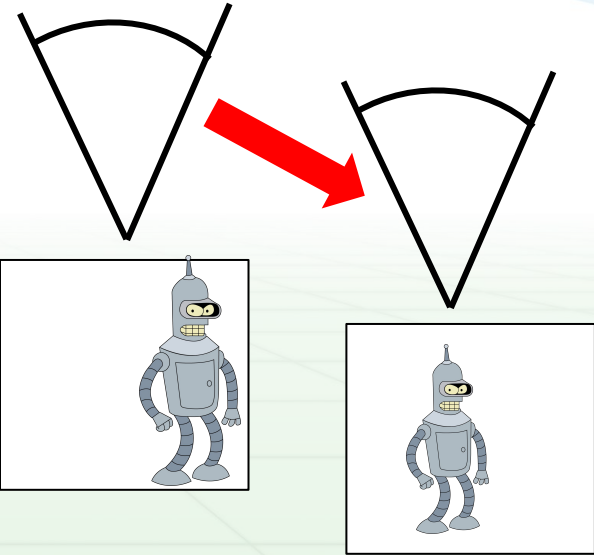
How do we perceive depth?

- Binocular Convergence
- Binocular Parallax



How do we perceive depth?

- Binocular Convergence
- Binocular Parallax
- Monocular Movement Parallax

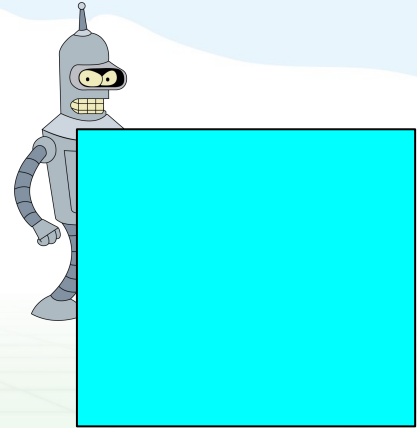


How do we perceive depth?

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

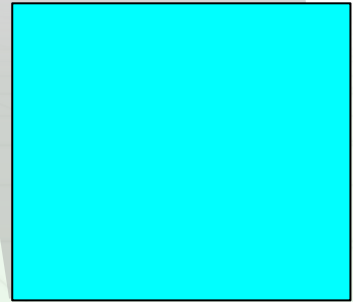
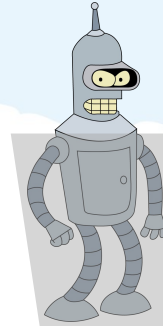
How do we perceive depth?

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

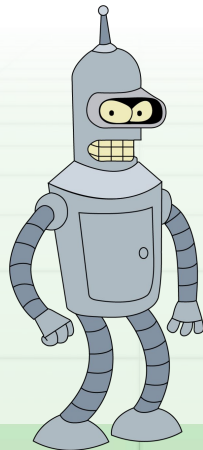


How do we perceive depth?

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

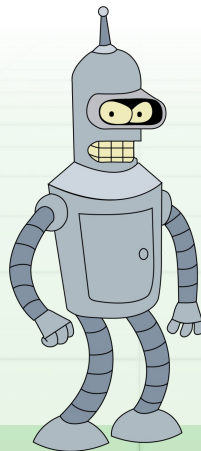


Interactive Demo



Interactive Demo

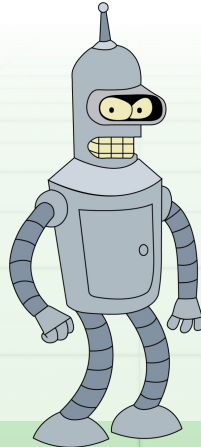
Close one eye



Interactive Demo

Close one eye

Cover Bender with your thumb

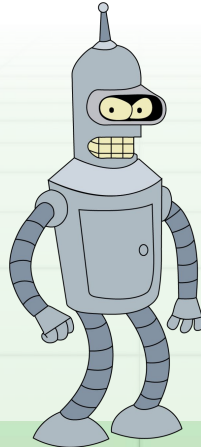


Interactive Demo

Close one eye

Cover Bender with your thumb

Open your other eye and close the first



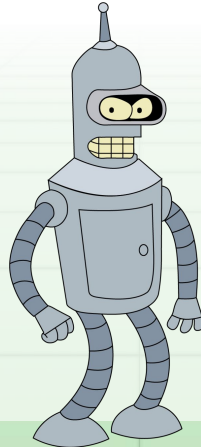
Interactive Demo

Close one eye

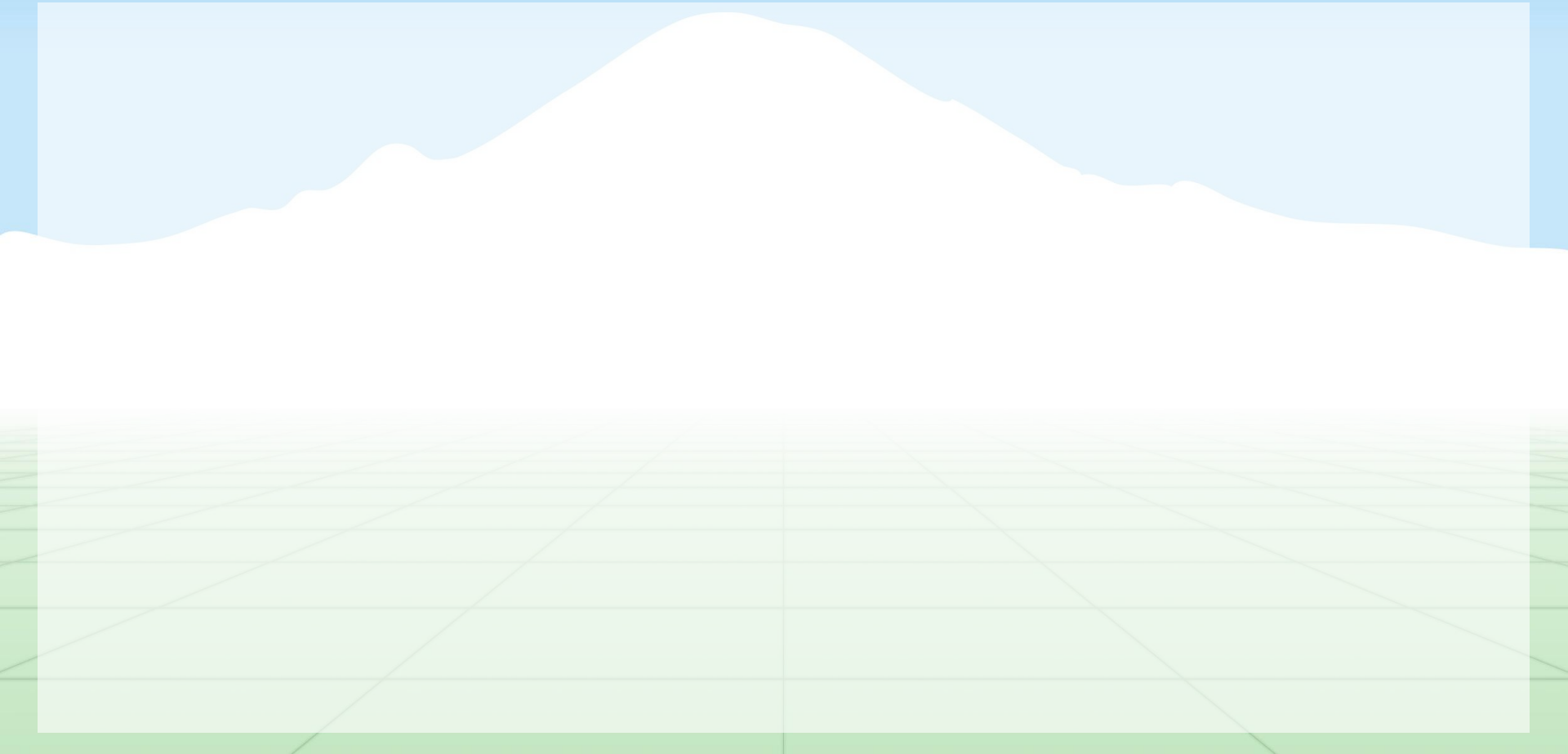
Cover Bender with your thumb

Open your other eye and close the first

Bender moved!



How do computers perceive depth?



How do computers perceive depth?

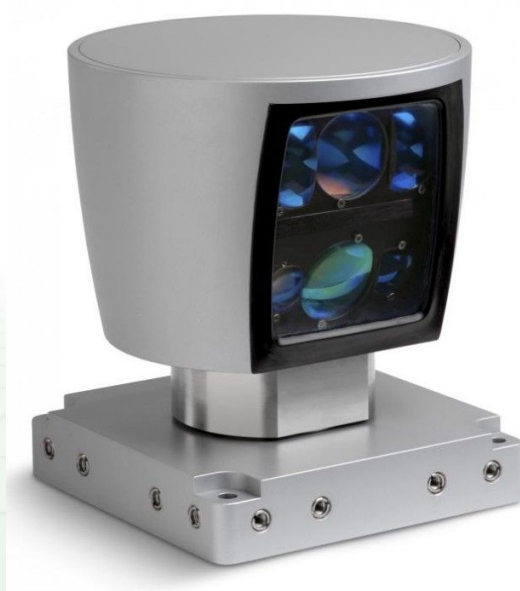
-Stereo



How do computers perceive depth?

-Stereo

-Lidar



How do computers perceive depth?

- Stereo
- Lidar
- Structured light

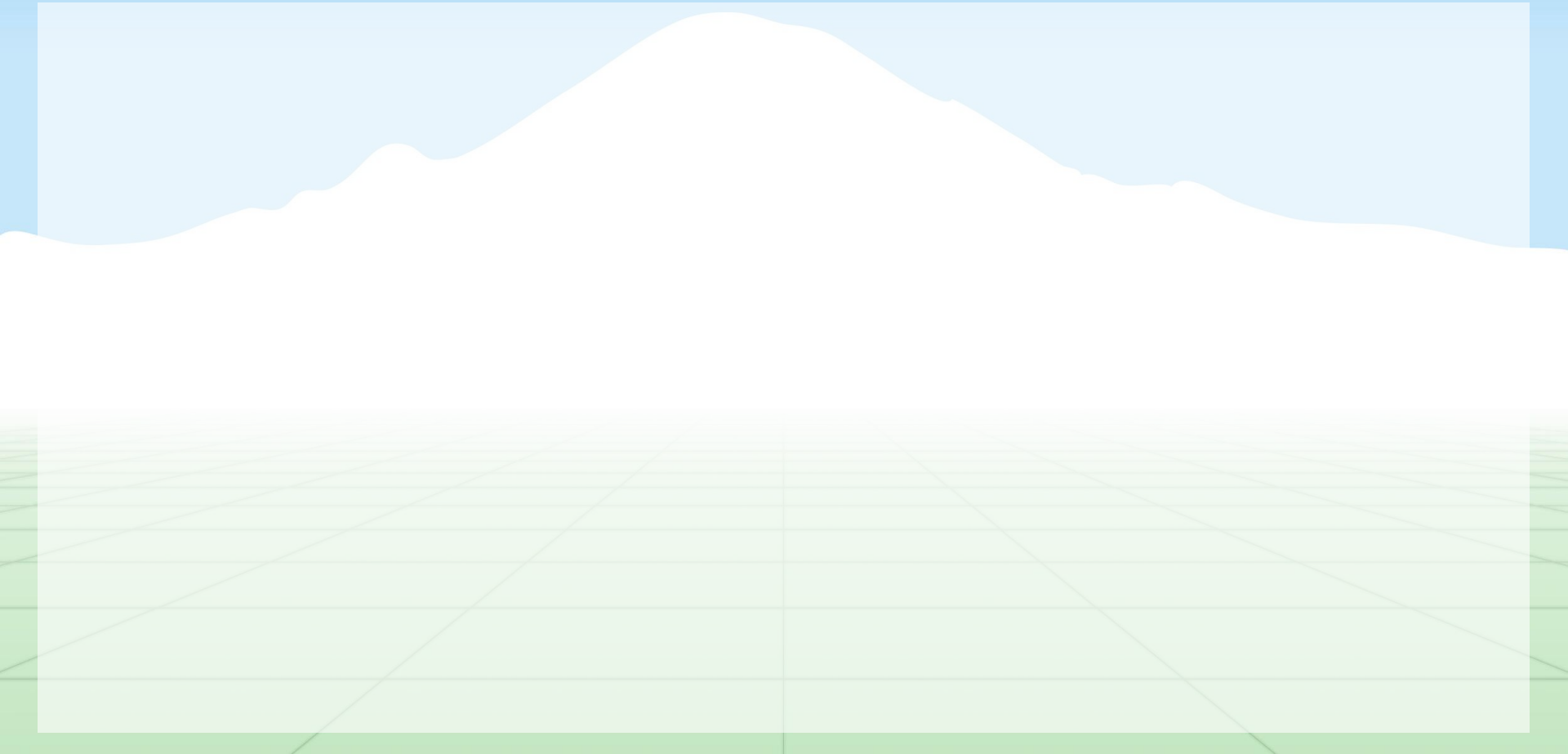


How do computers perceive depth?

- Stereo
- Lidar
- Structured light
- Time of flight

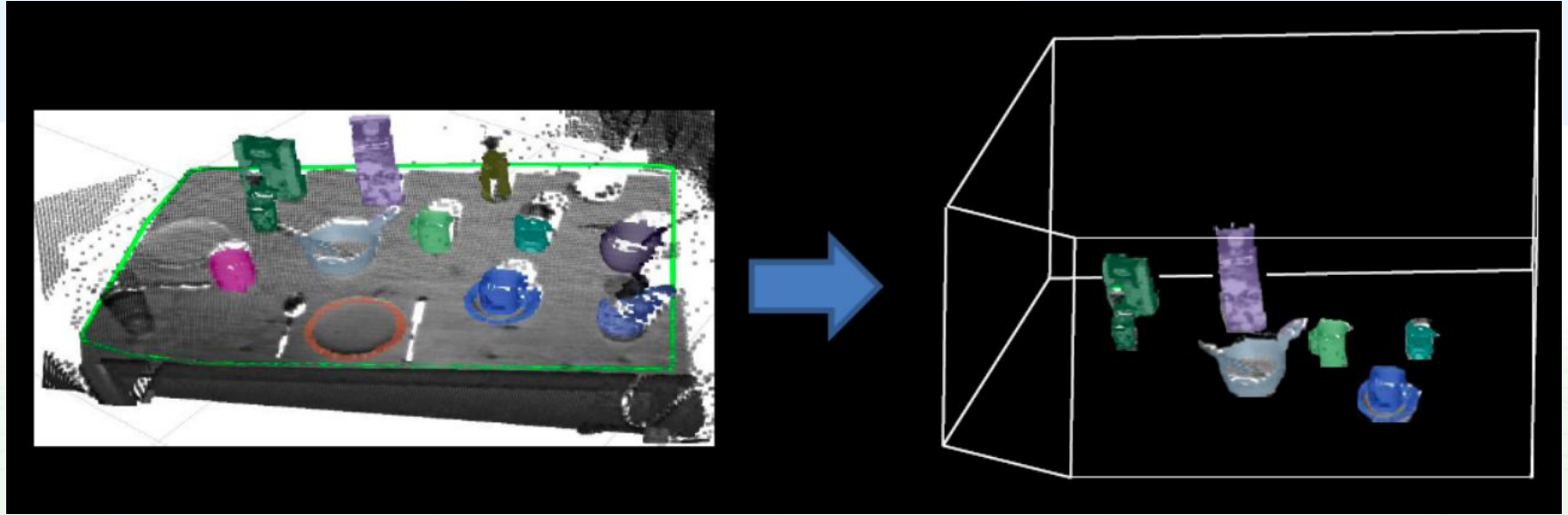


Why do we want depth?



Why do we want depth?

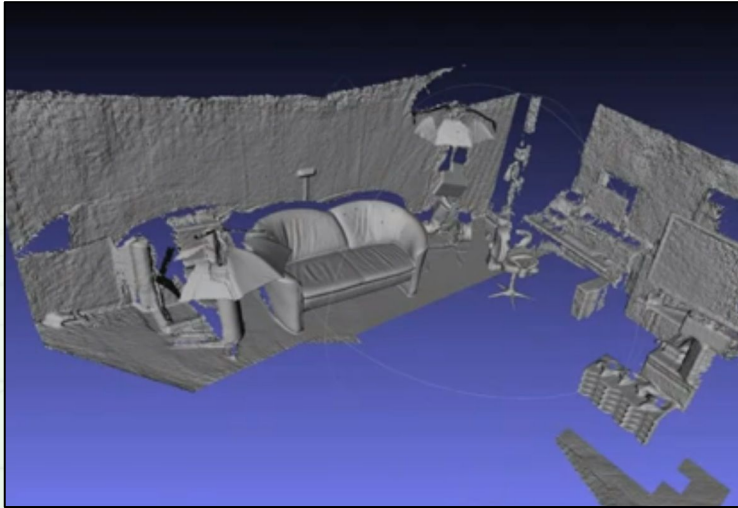
-Object segmentation



Why do we want depth?

- Object segmentation
- 3D reconstruction

KinectFusion



Why do

-Object

-3D re



Live Input Depth Map



Live Model Output



Live RGB Image (unused)



Canonical Model Reconstruction



Warped Model

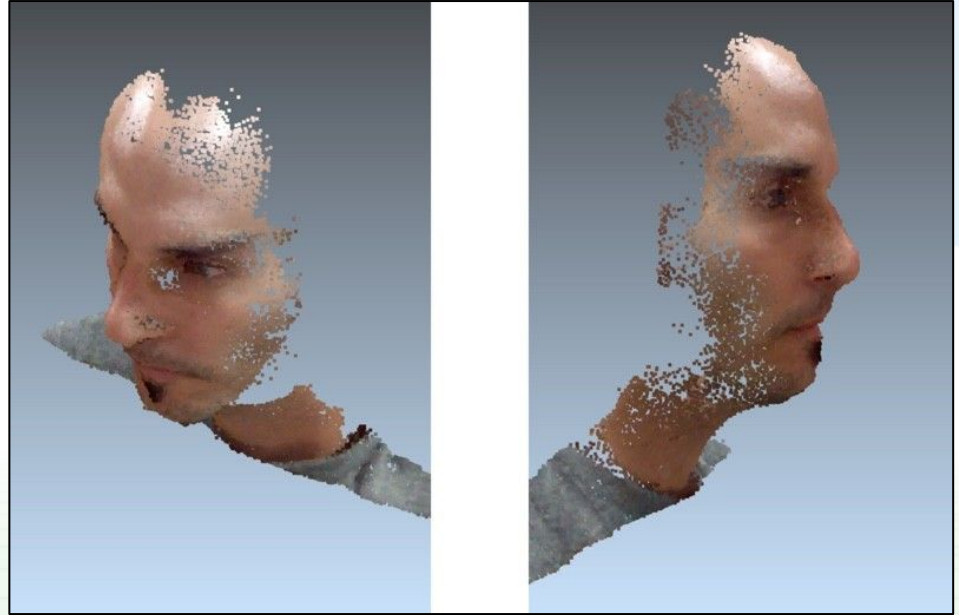
Why do we want depth?

- Object segmentation
- 3D reconstruction
- Navigation



Why do we want depth?

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

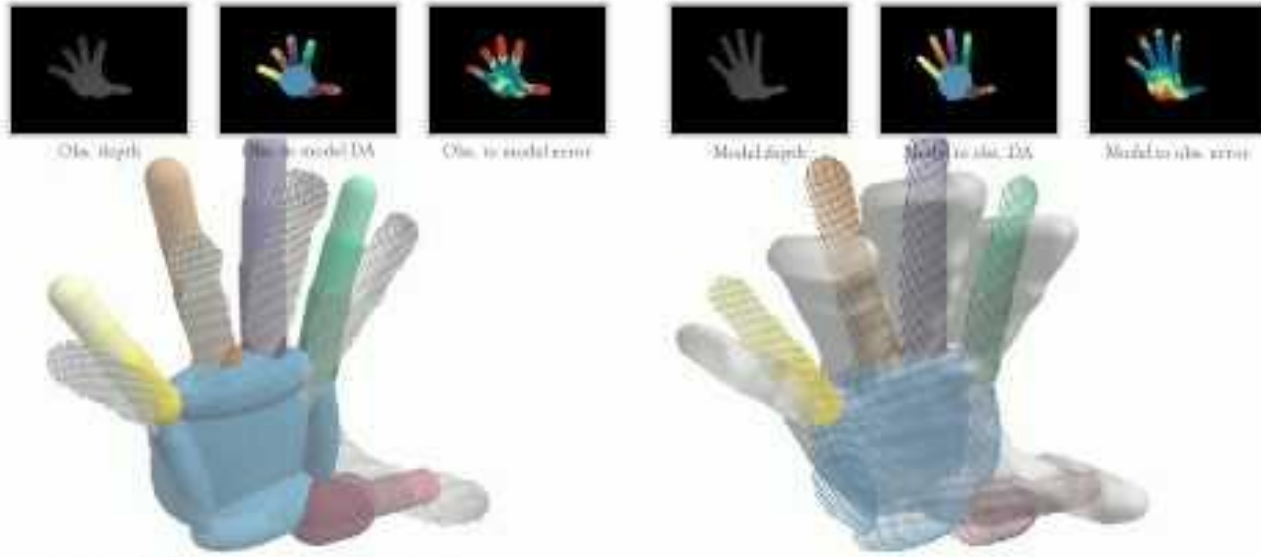


Why do we want depth?

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

Why do

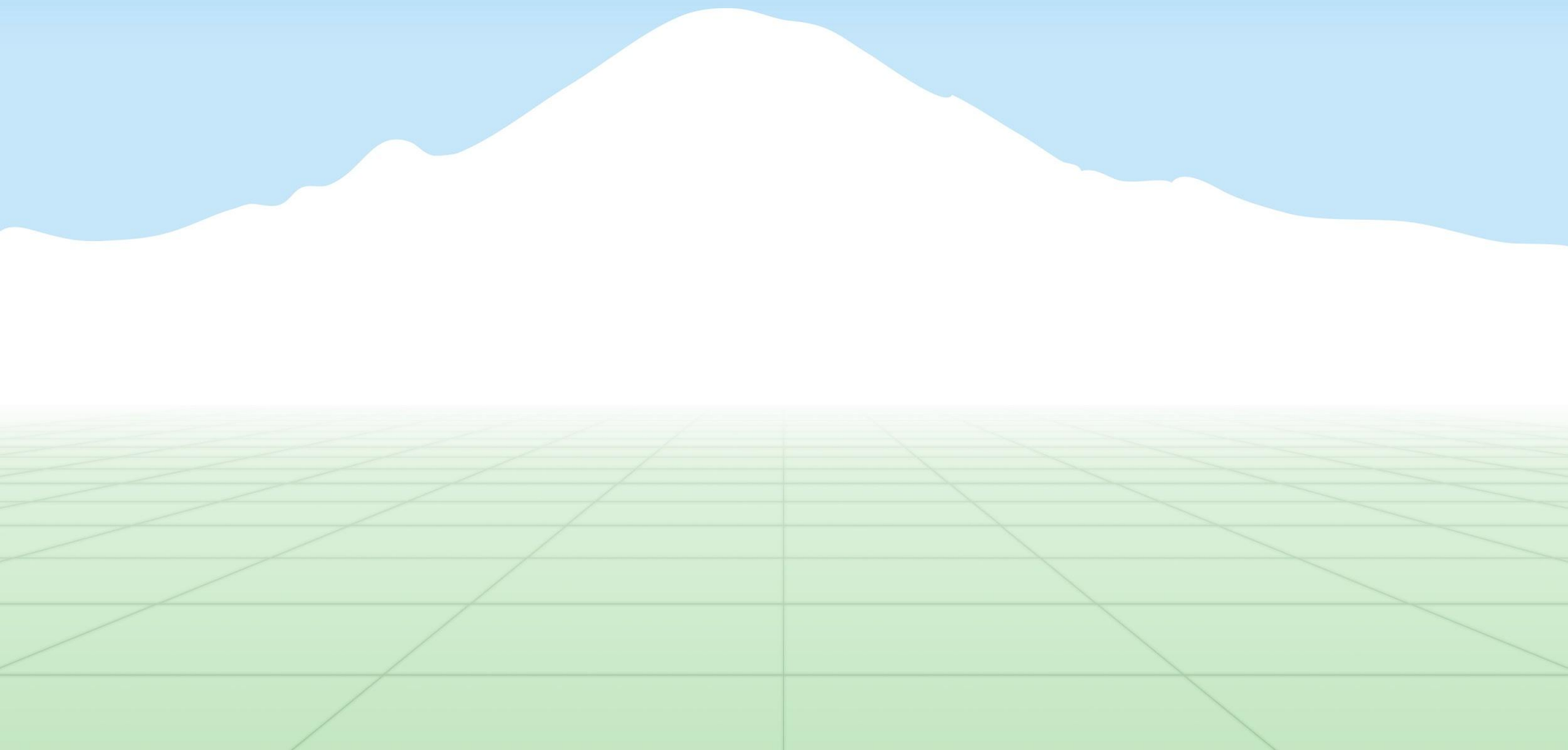
- Object
- 3D rec
- Naviga
- Facial
- Pose t



- The full objective function is given by:

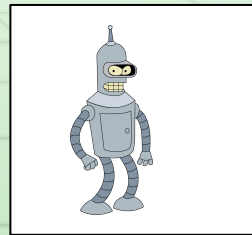
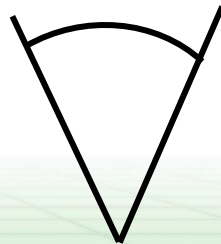
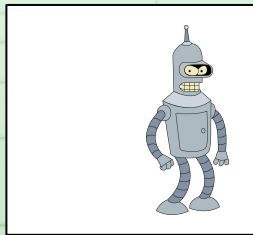
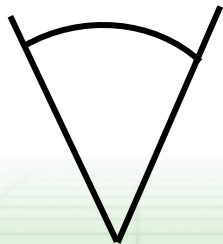
$$\hat{\theta} = \arg \min_{\theta} \sum_{u \in \mathcal{U}} \text{SDF}_{\text{real}}(\mathbf{x}_u; \theta)^2 + \lambda \sum_{u \in \mathcal{U}} \text{SDF}_{\text{obs}}(\hat{\mathbf{x}}_u(\theta); D)^2$$

Stereo



Stereo

aka Binocular Parallax



Stereo

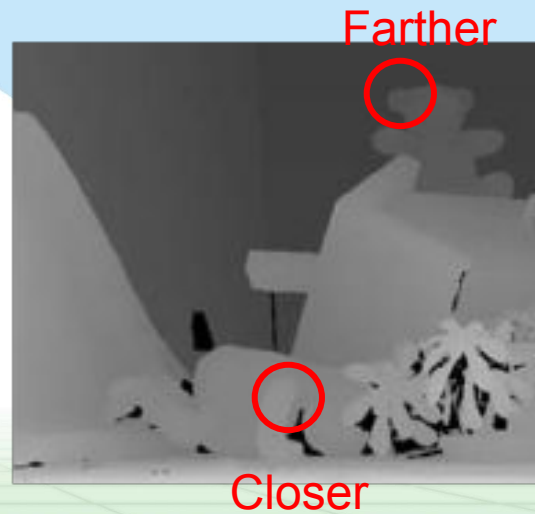


Stereo



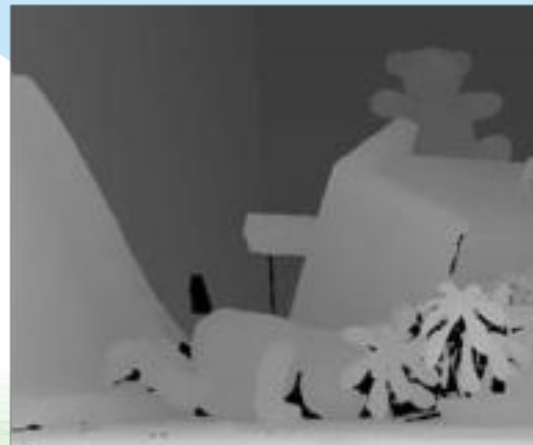
Displacement inversely proportional to distance from the camera.

Stereo



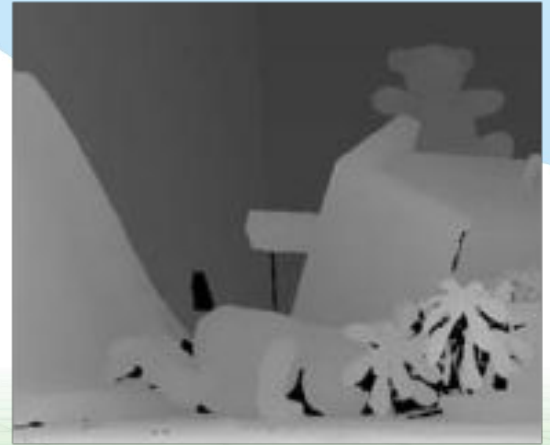
Stereo

How do we align different parts of two images?



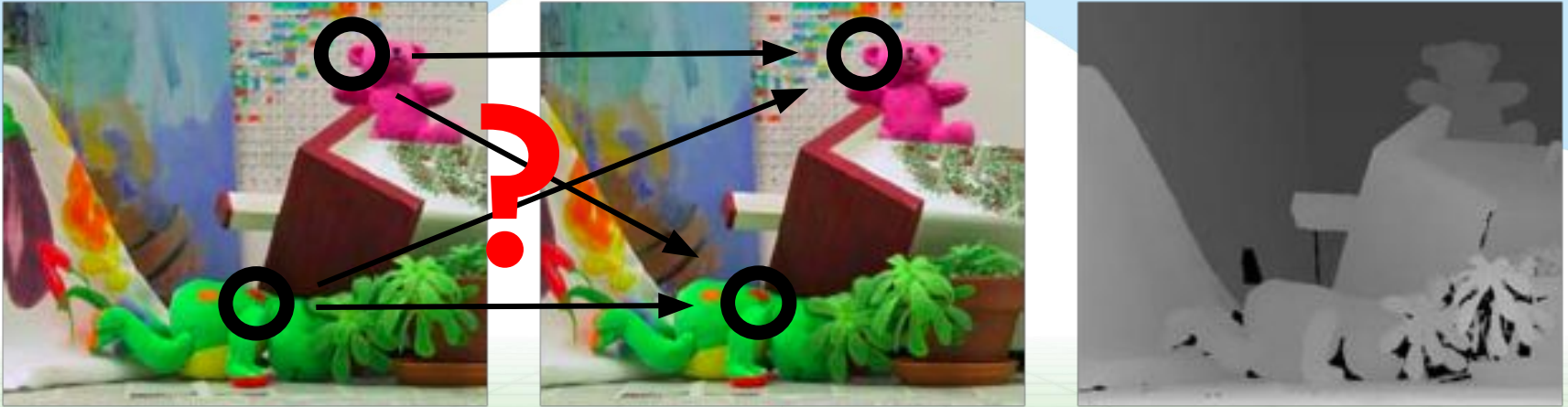
Stereo

Feature matching!



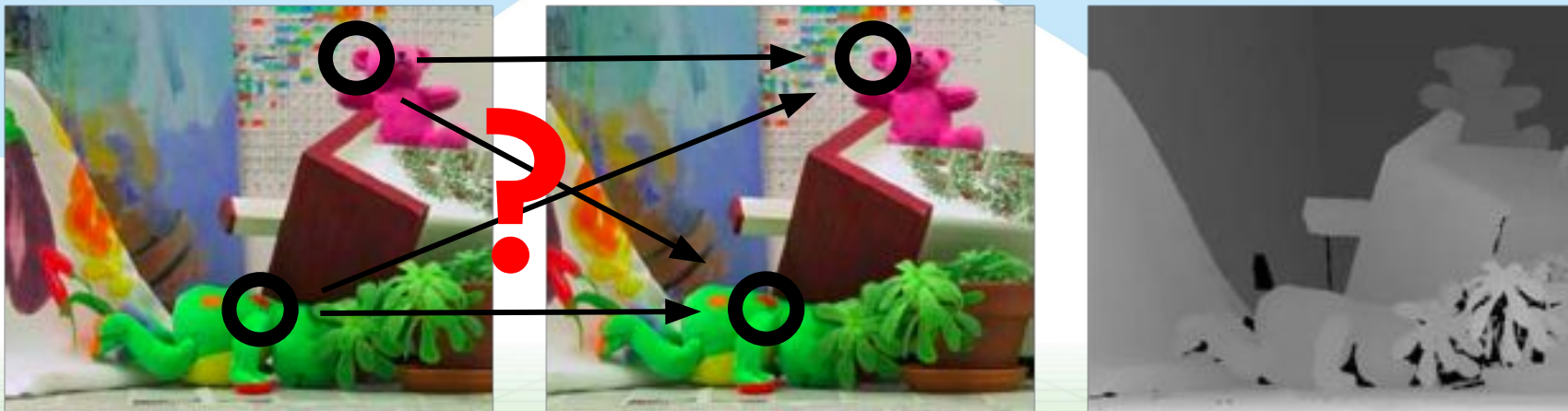
Stereo

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



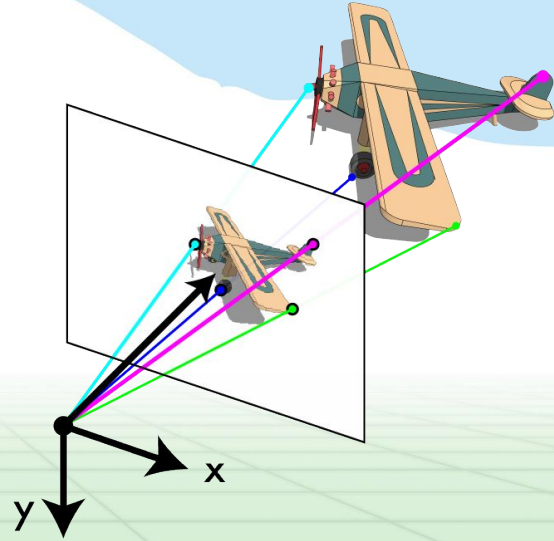
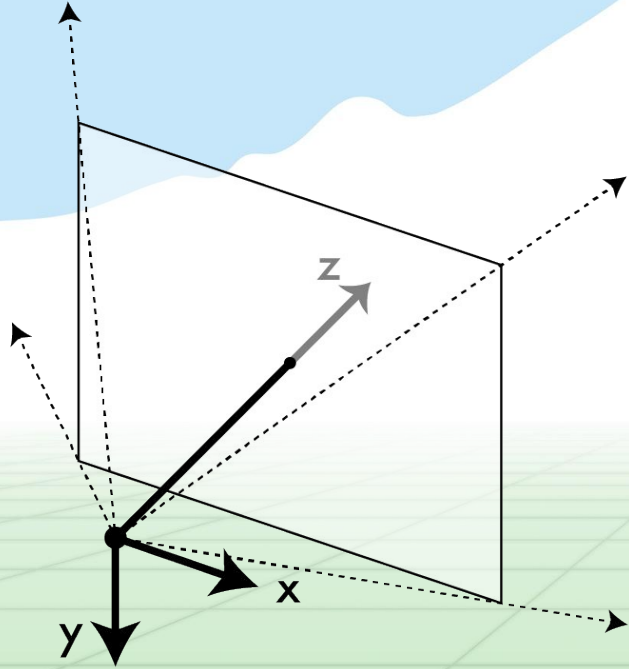
Stereo

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

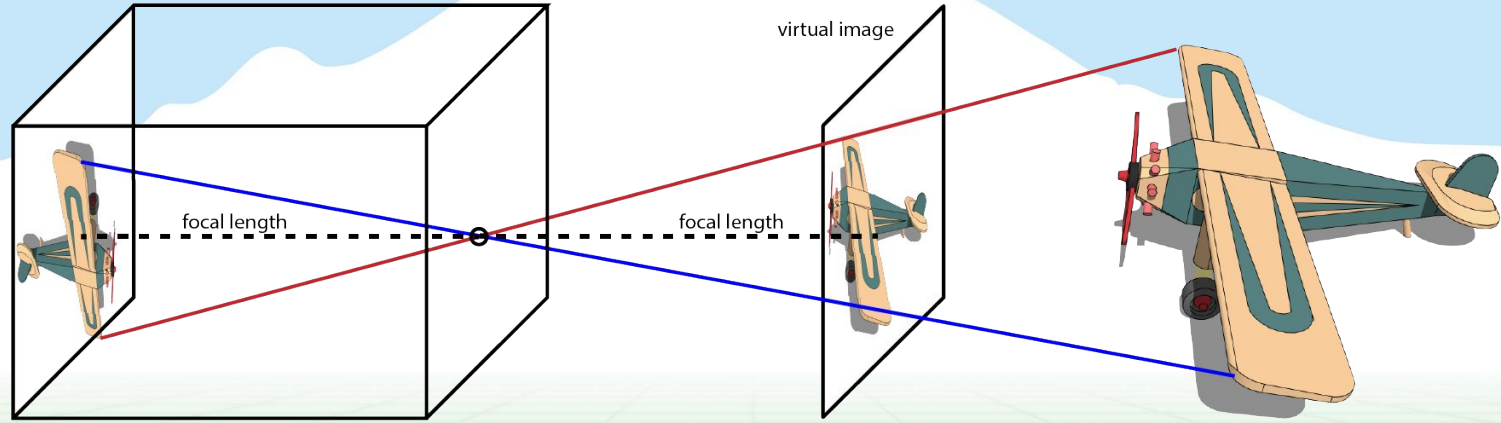


Very difficult, can we take advantage of the structure somehow?

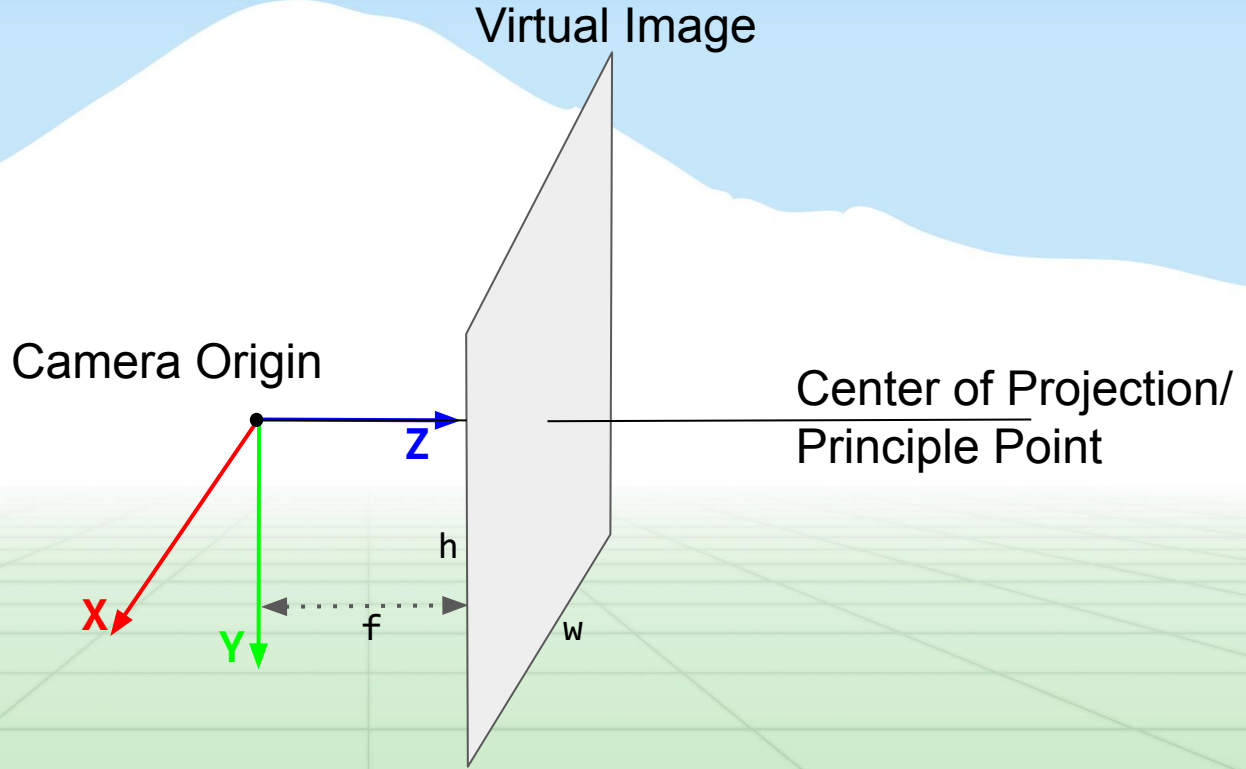
Perspective Projection



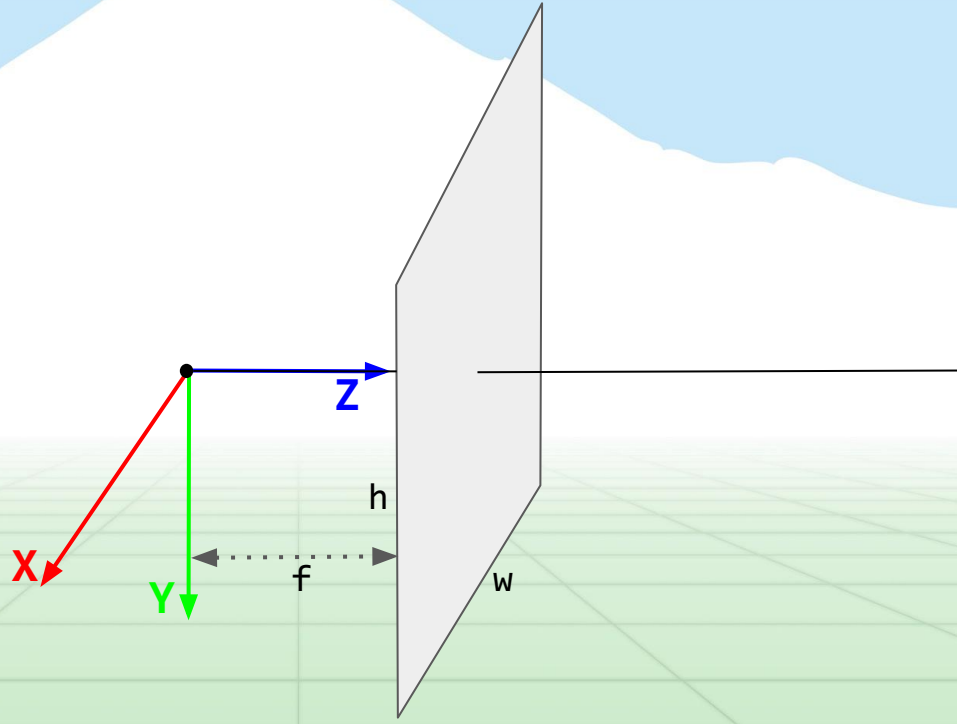
Perspective Projection



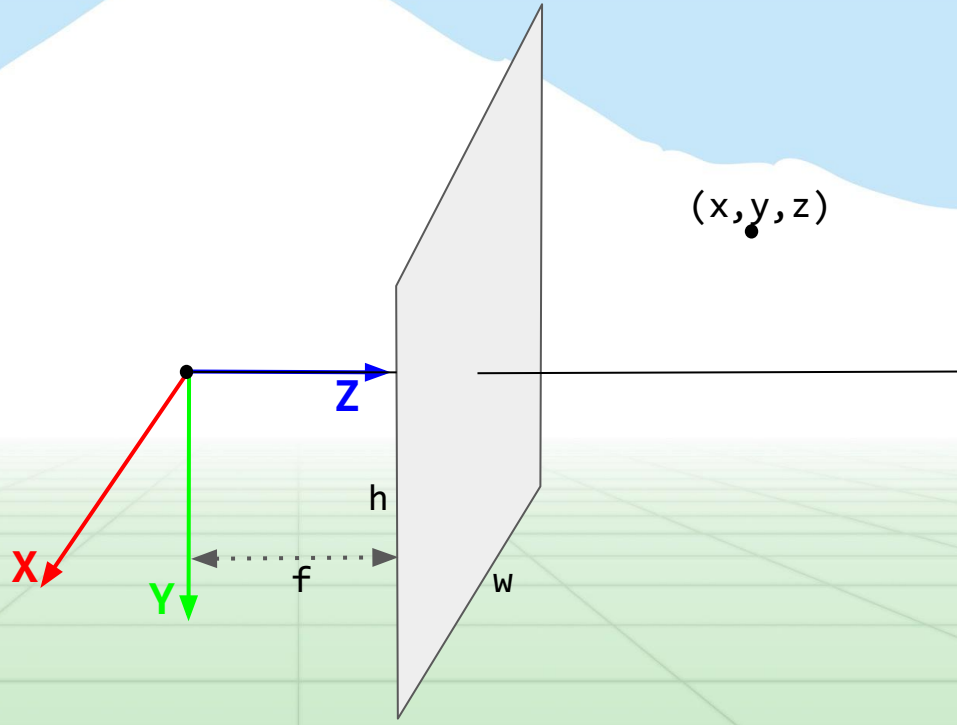
Perspective Projection



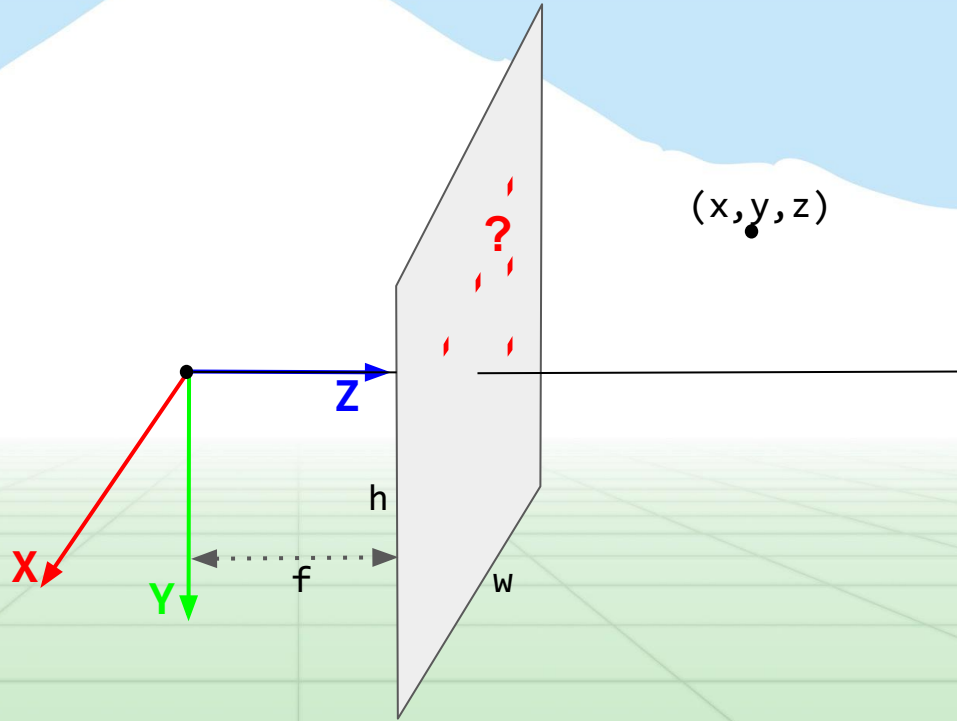
Perspective Projection



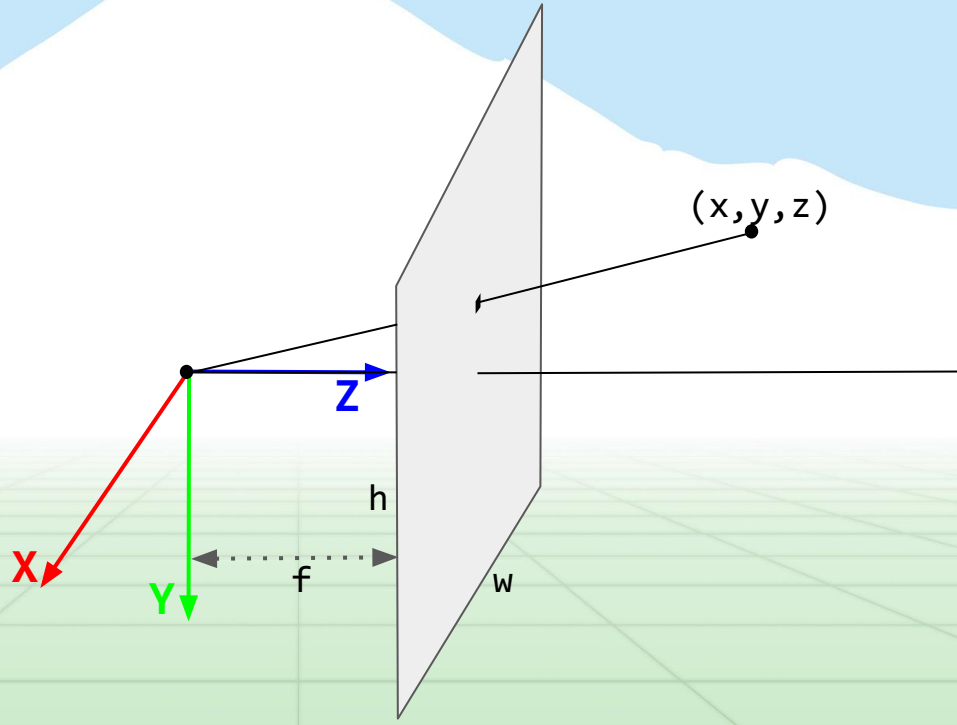
Perspective Projection



Perspective Projection



Perspective Projection

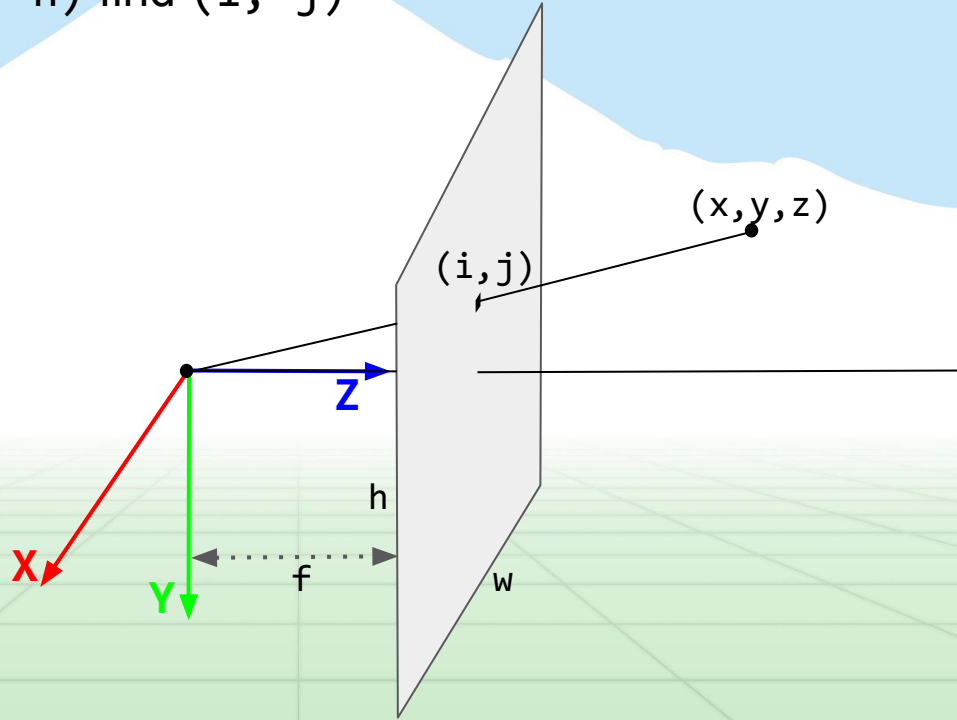


Perspective Projection

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

$i = ?$

$j = ?$

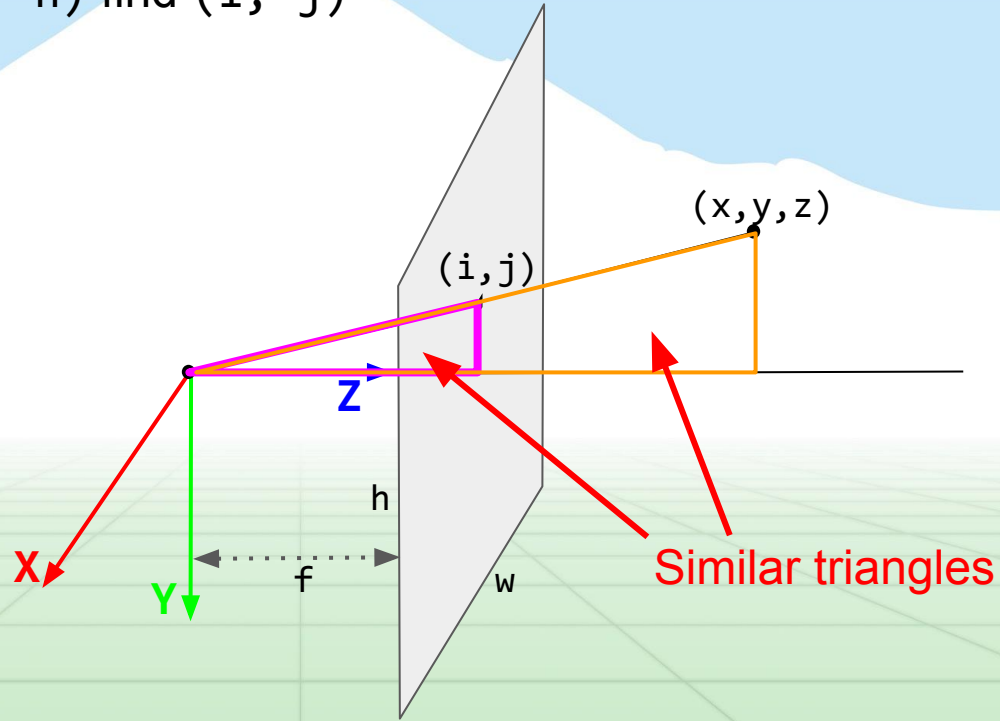


Perspective Projection

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

$i = ?$

$j = ?$

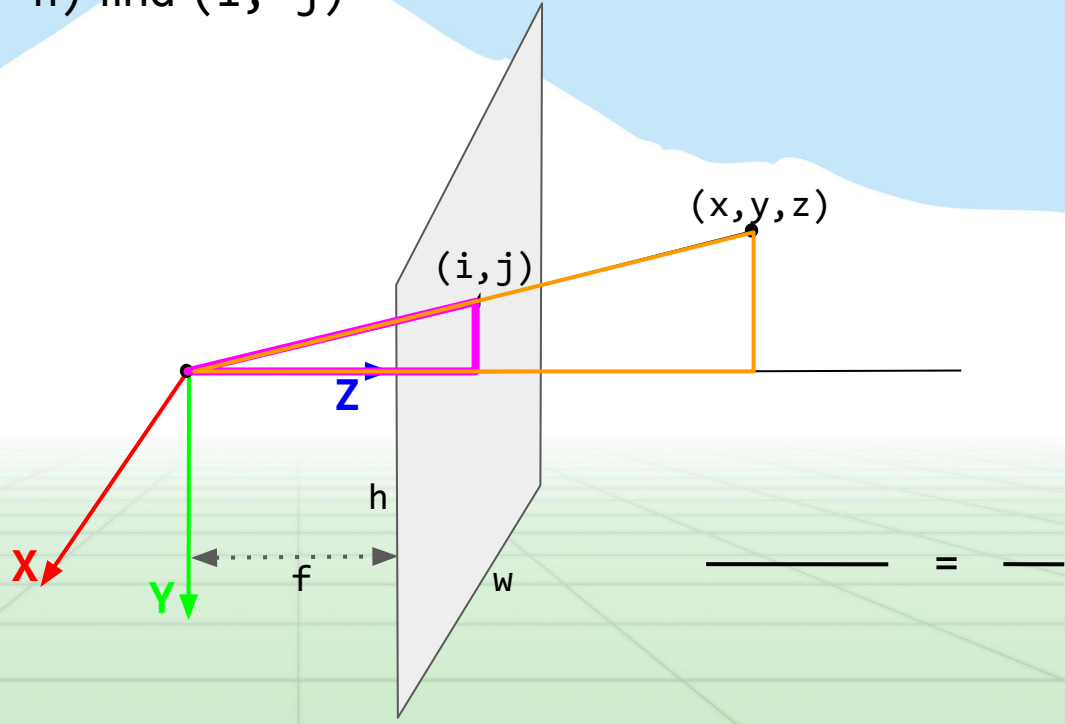


Perspective Projection

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

$i = ?$

$j = ?$

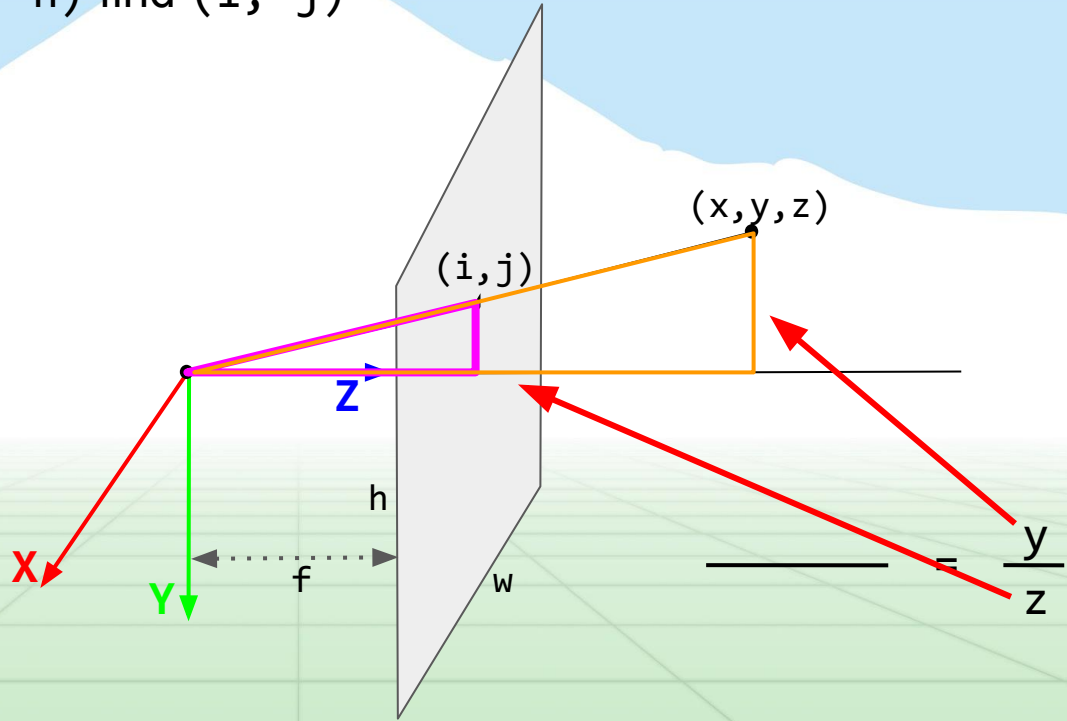


Perspective Projection

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

$$i = ?$$

$$j = ?$$

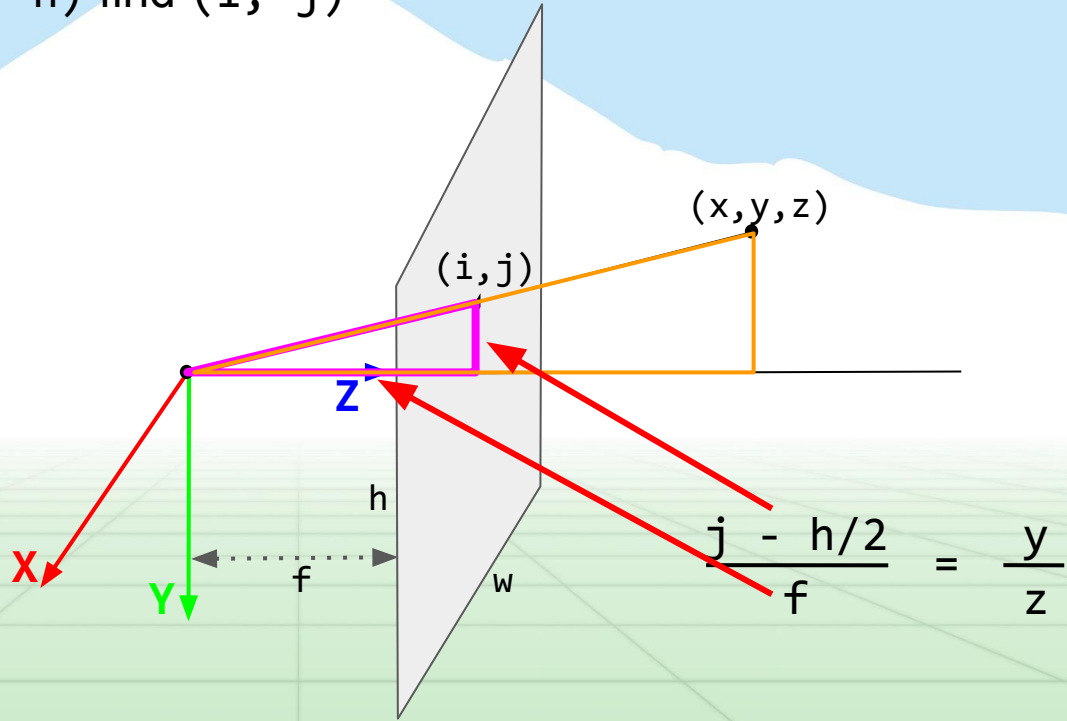


Perspective Projection

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

$i = ?$

$j = ?$

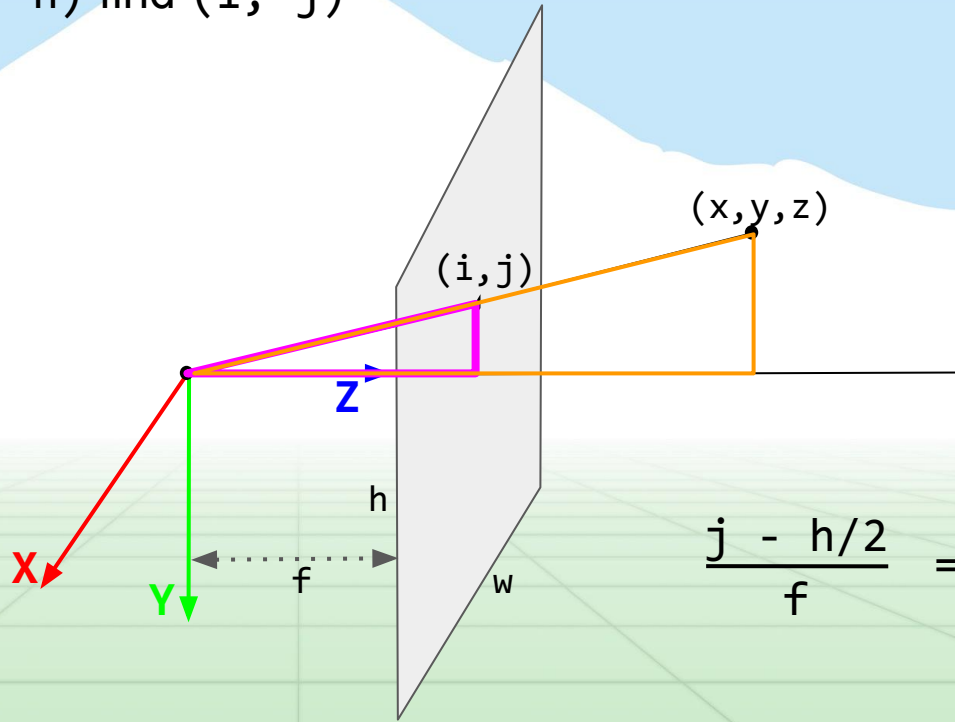


Perspective Projection

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

$i = ?$

$j = ?$



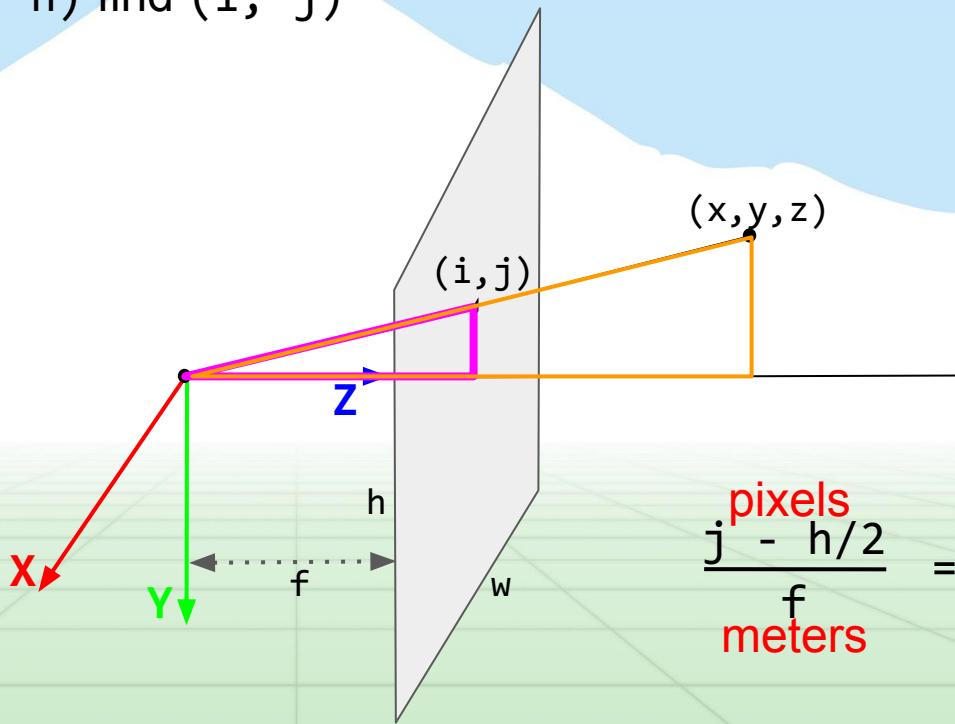
$$\frac{j - h/2}{f} = \frac{y}{z}$$

Perspective Projection

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

$$i = ?$$

$$j = ?$$



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

pixels
meters

Perspective Projection

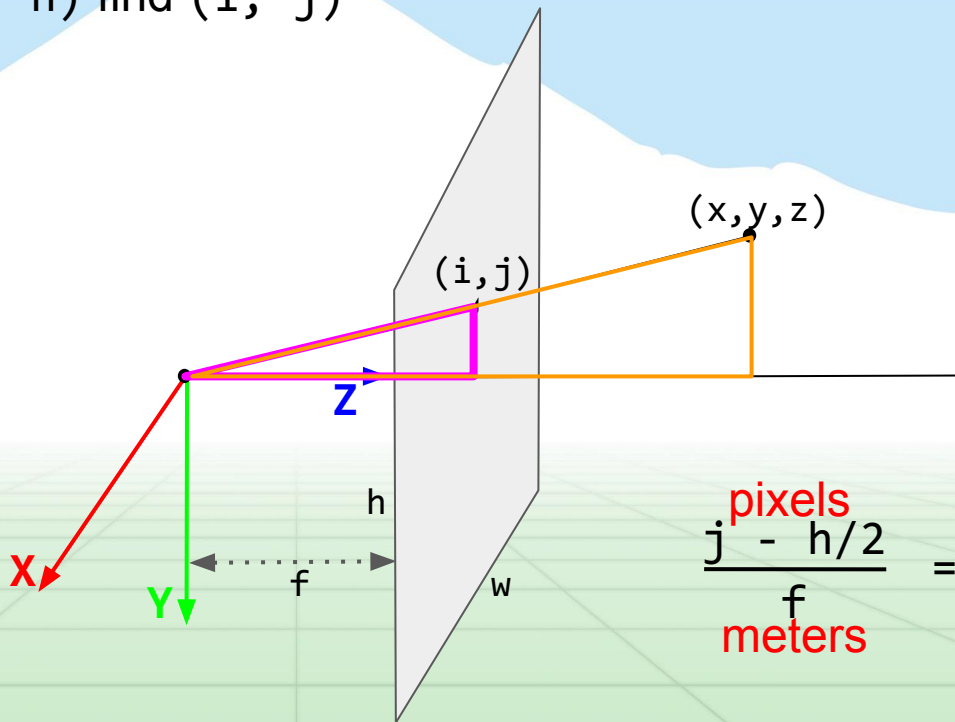
Given (x, y, z) and (f, w, h) find (i, j)

$$i = ?$$

$$j = ?$$

Pixel Pitch:

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



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

pixels
meters

Perspective Projection

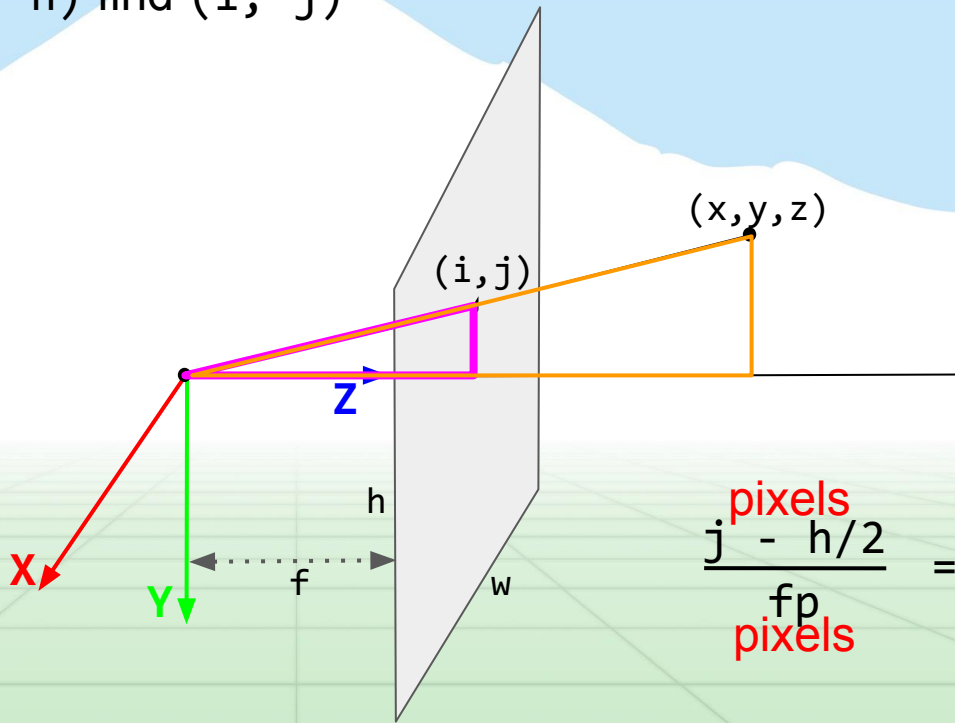
Given (x, y, z) and (f, w, h) find (i, j)

$$i = ?$$

$$j = ?$$

Pixel Pitch:

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



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

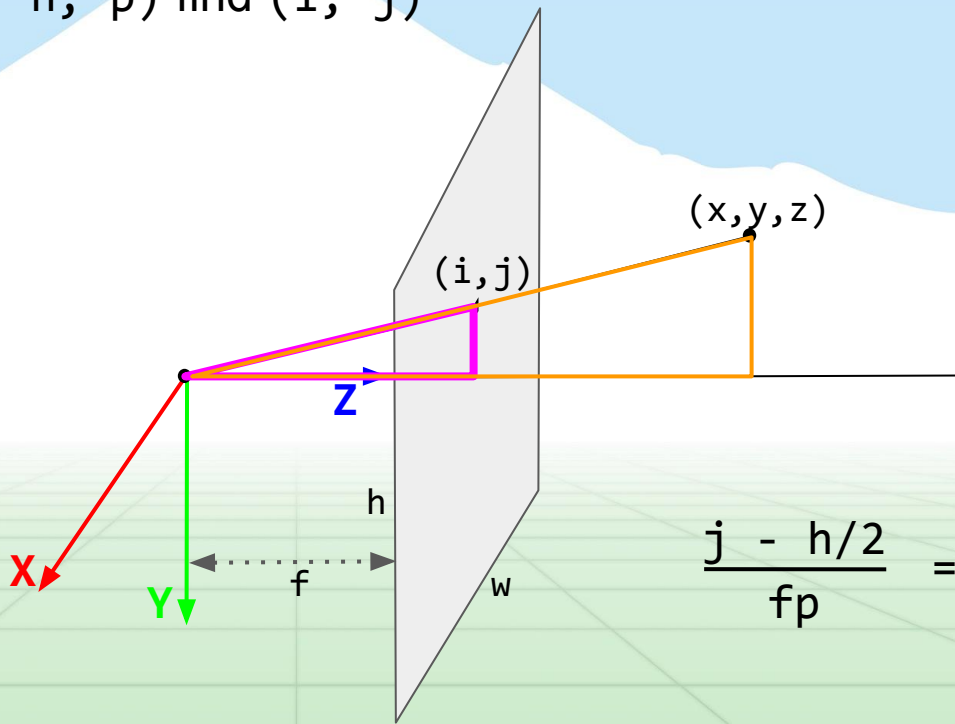
pixels
pixels

Perspective Projection

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

$$i = ?$$

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

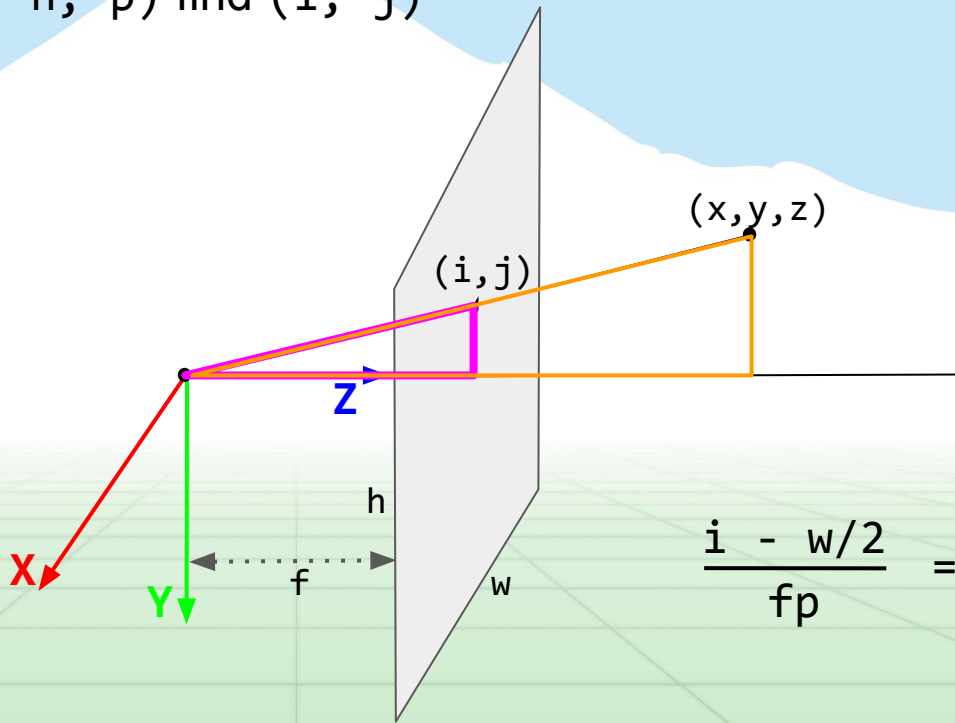


$$\frac{j - h/2}{fp} = \frac{y}{z}$$

Perspective Projection

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

$$i = \frac{fp}{z}x + \frac{w}{2}$$
$$j = \frac{fp}{z}y + \frac{h}{2}$$



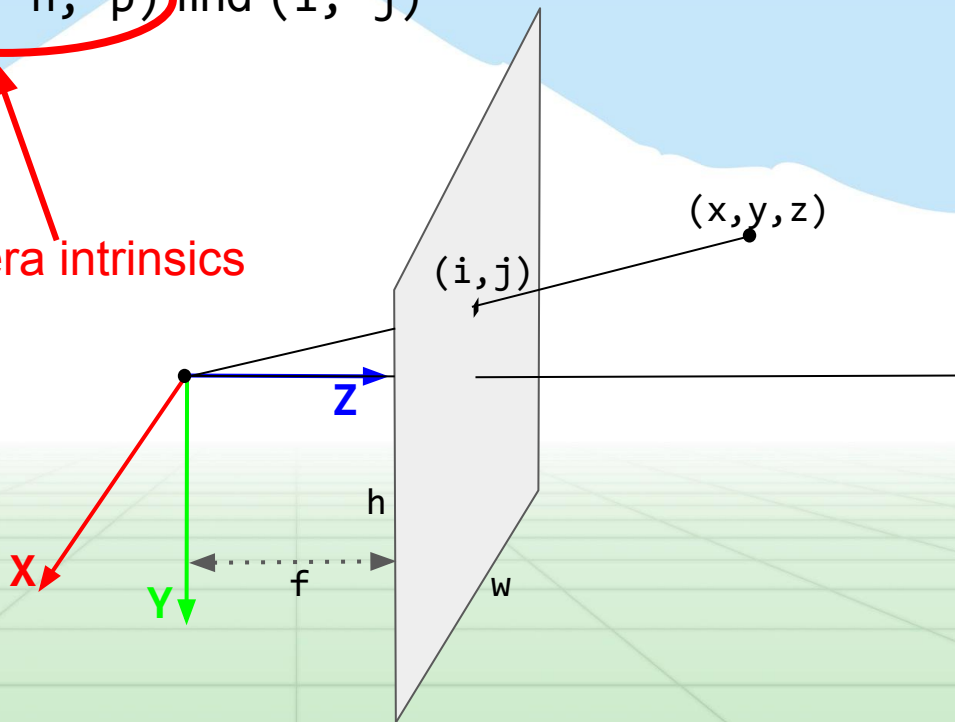
$$\frac{i - w/2}{fp} = \frac{x}{z}$$

Perspective Projection

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

$$i = \frac{fp}{z}x + \frac{w}{2}$$
$$j = \frac{fp}{z}y + \frac{h}{2}$$

Camera intrinsics



Perspective Projection

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

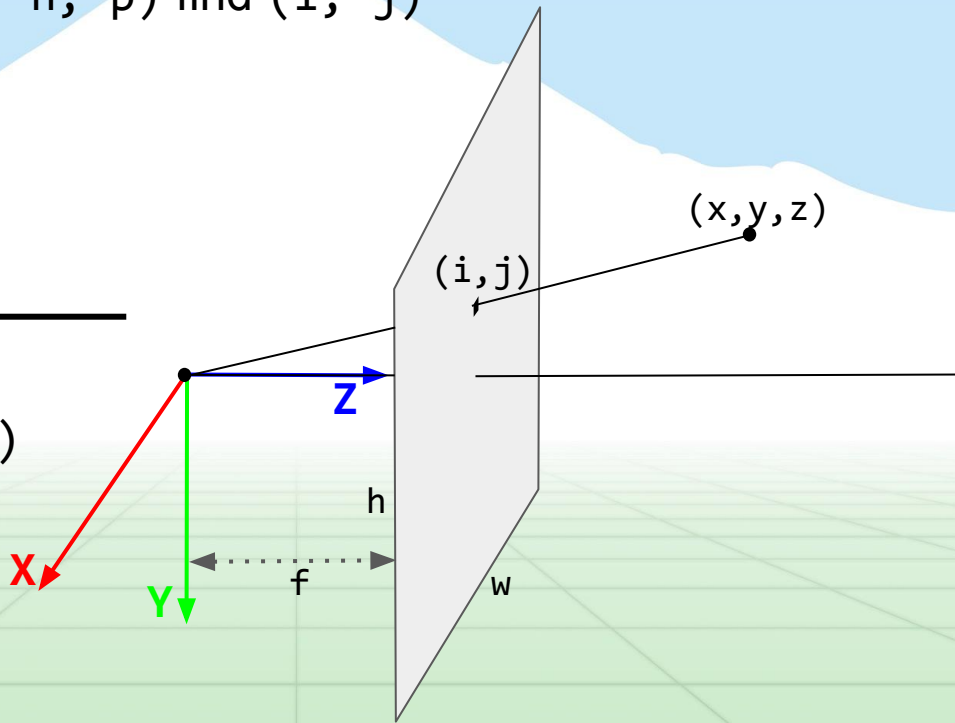
$$i = \frac{fpx}{z} + \frac{w}{2}$$
$$j = \frac{fpy}{z} + \frac{h}{2}$$

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)

$$i = \frac{fp}{z}x + \frac{w}{2}$$
$$j = \frac{fp}{z}y + \frac{h}{2}$$

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

$$x = \frac{(i - \frac{w}{2})z}{f}$$
$$y = \frac{(j - \frac{h}{2})z}{f}$$
$$z = \dots$$



Perspective Projection

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

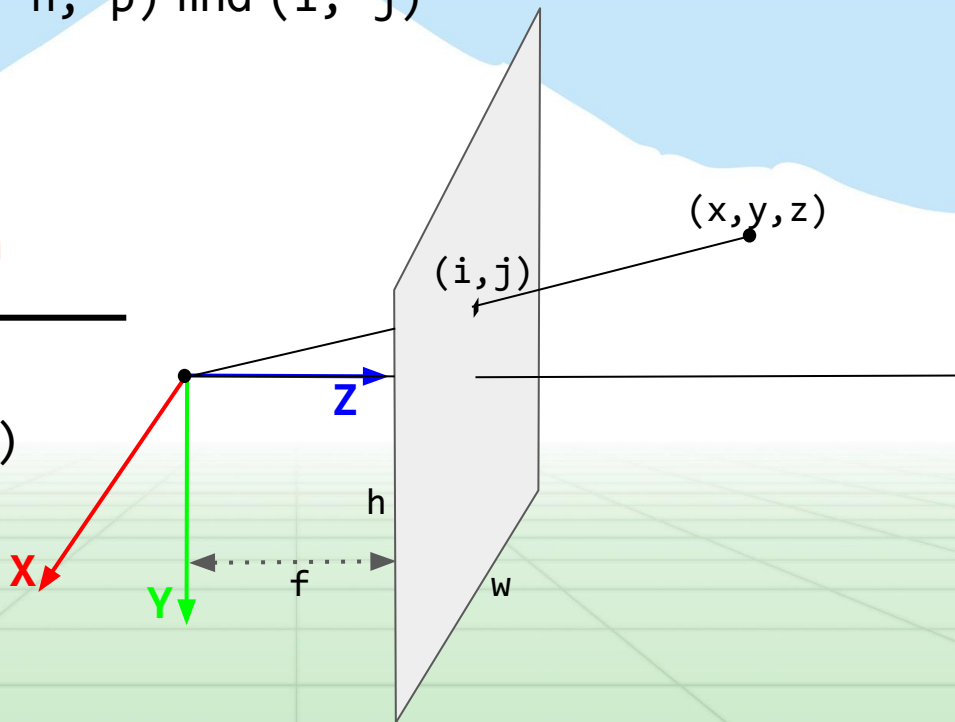
$$i = \frac{fpx}{z} + \frac{w}{2}$$
$$j = \frac{fpy}{z} + \frac{h}{2}$$

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

$$x = ?$$

$$y = ?$$

$$z = ?$$



Perspective Projection

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

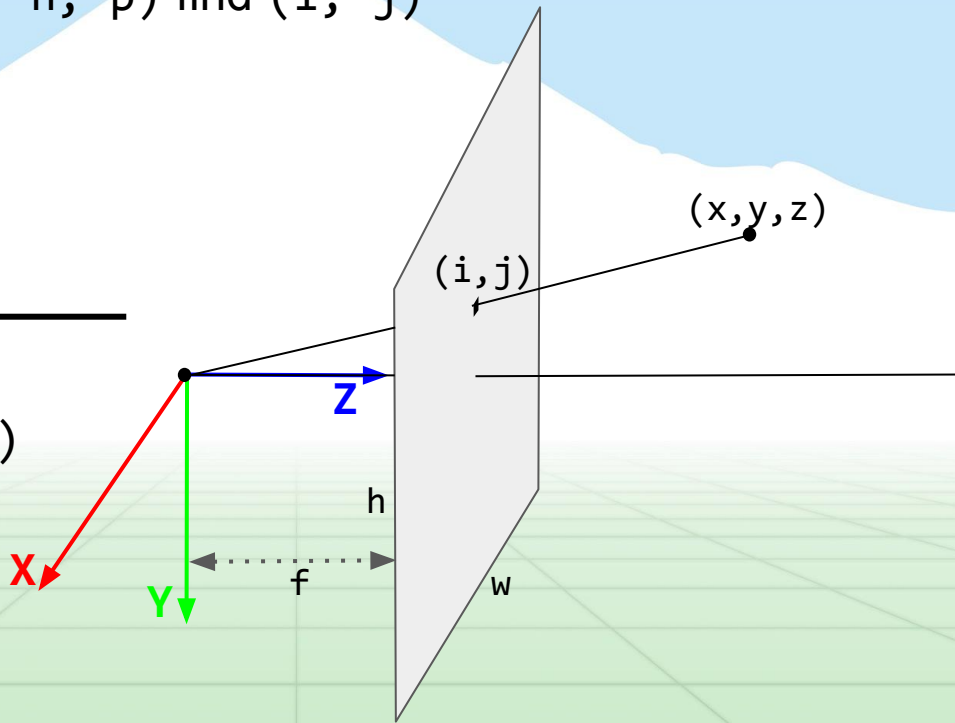
$$i = \frac{fp}{z}x + \frac{w}{2}$$
$$j = \frac{fp}{z}y + \frac{h}{2}$$

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)

$$i = \frac{fp}{z}x + \frac{w}{2}$$
$$j = \frac{fp}{z}y + \frac{h}{2}$$

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

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

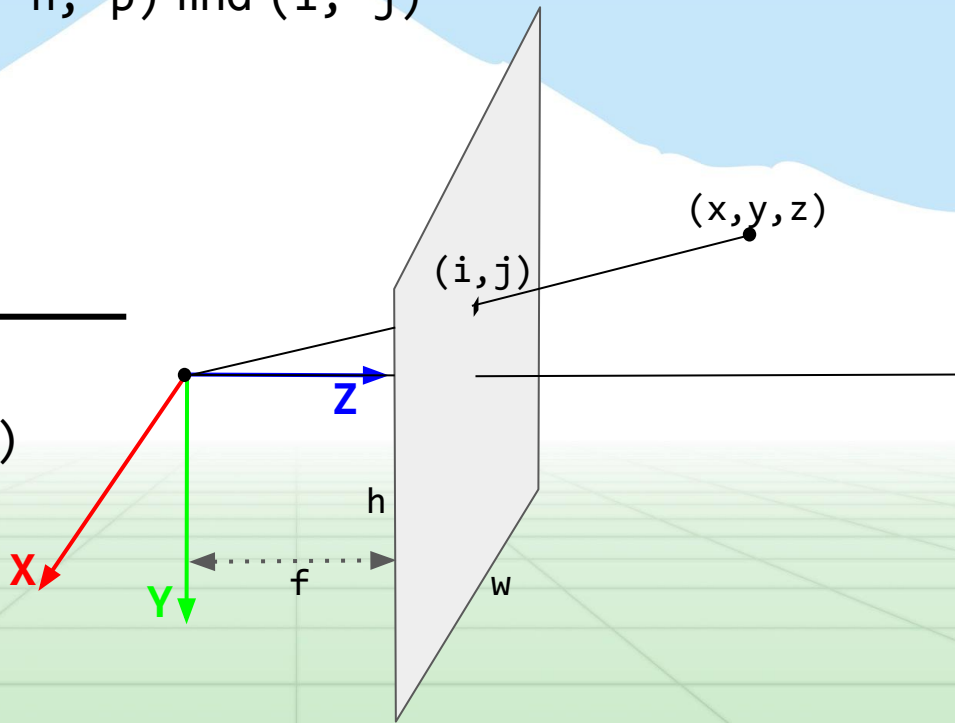


Image Coordinates

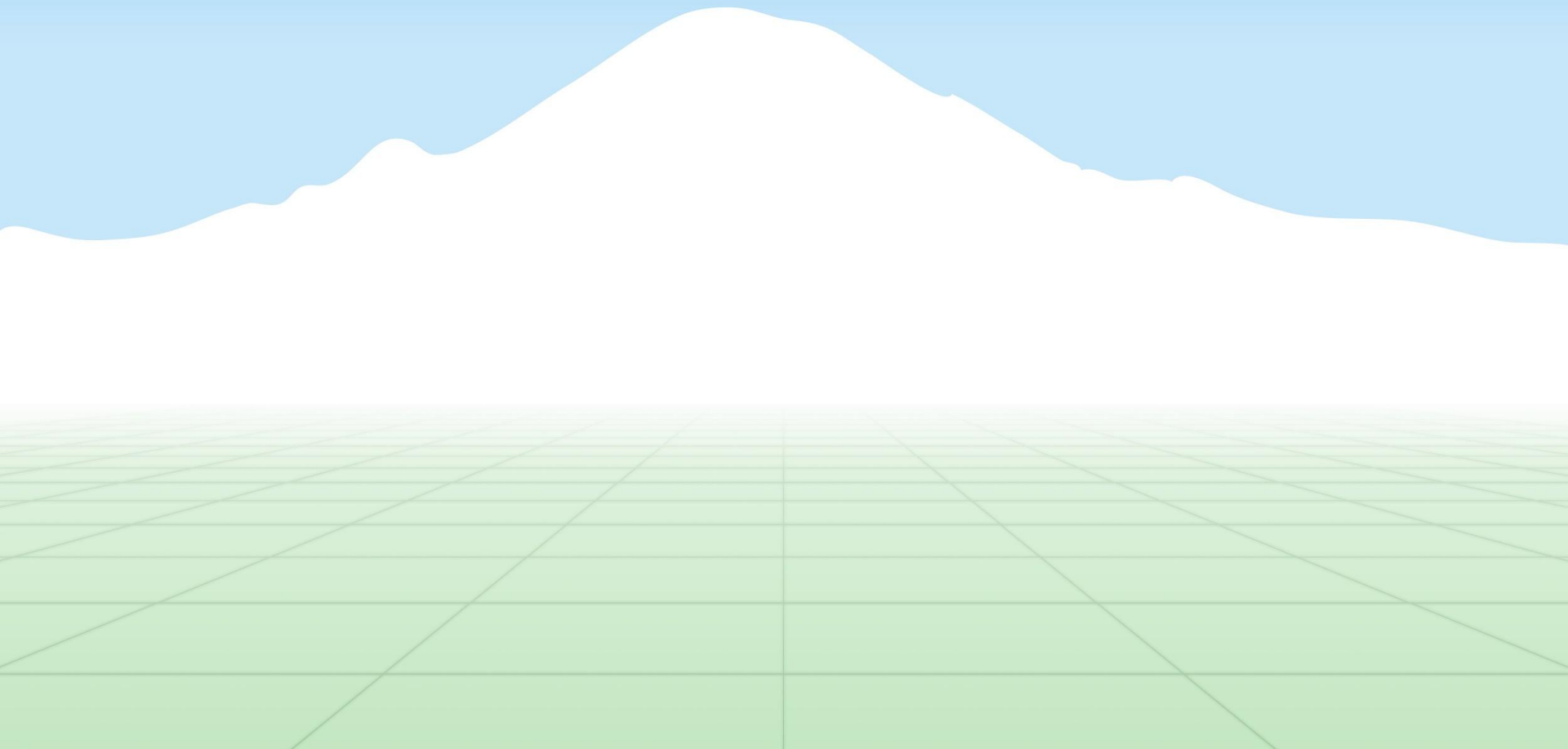


Image Coordinates

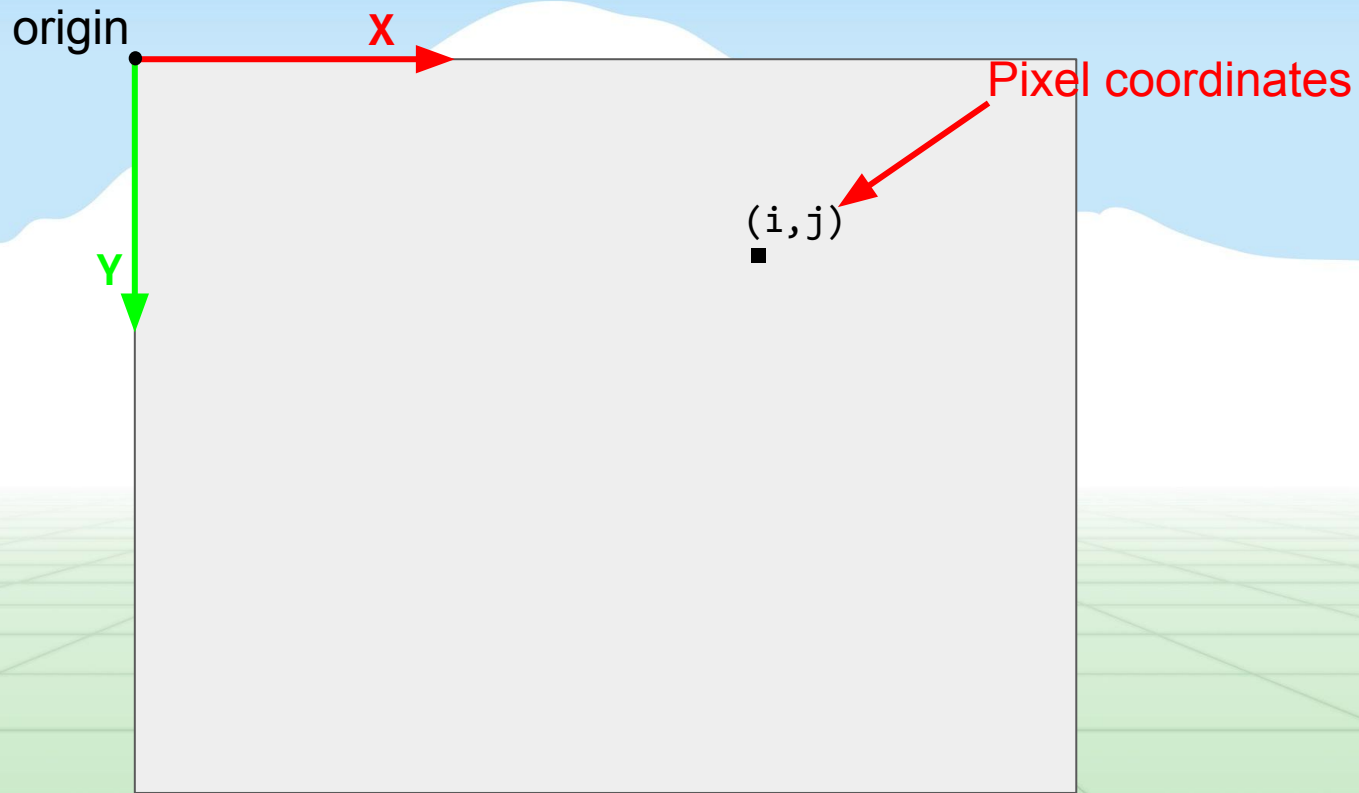


Image Coordinates

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

x' , y' , z' in
meters.

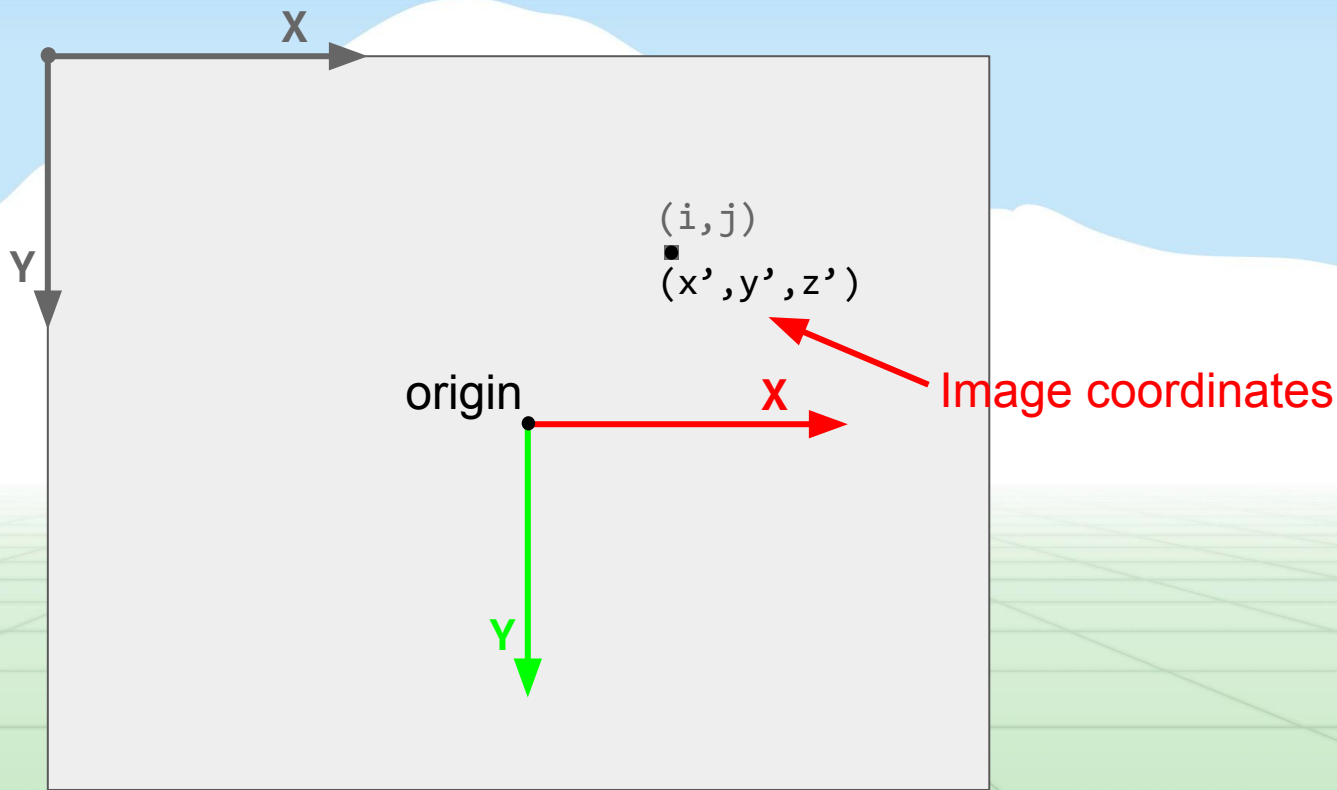
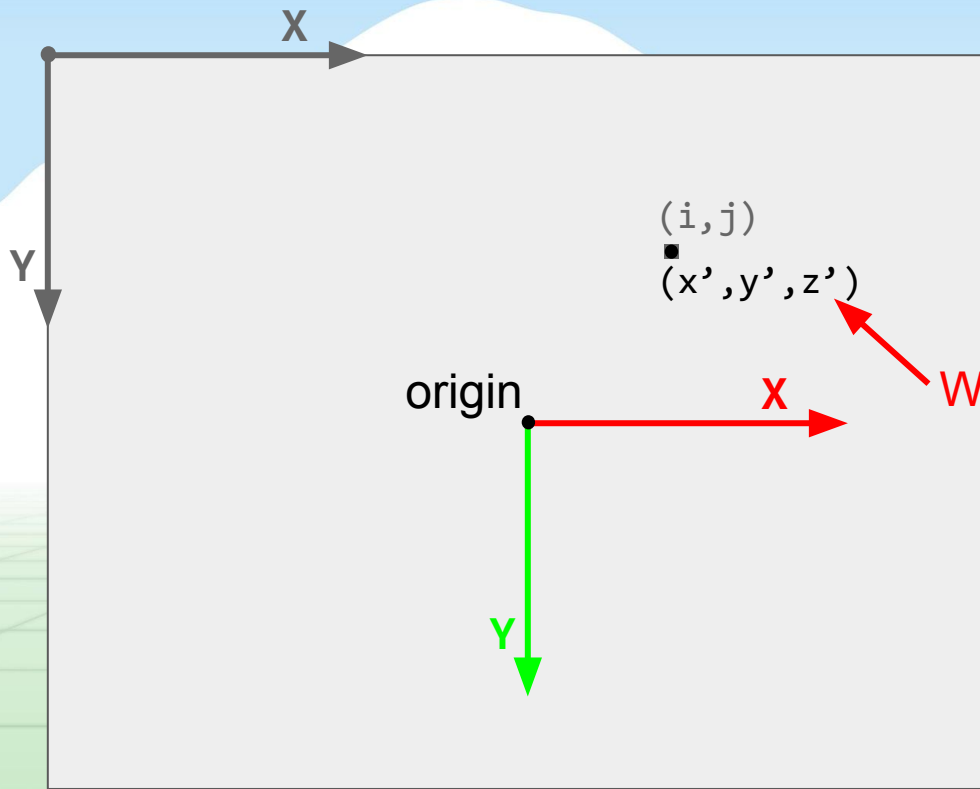


Image Coordinates

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

x' , y' , z' in
meters.



What is z' ?

Image Coordinates

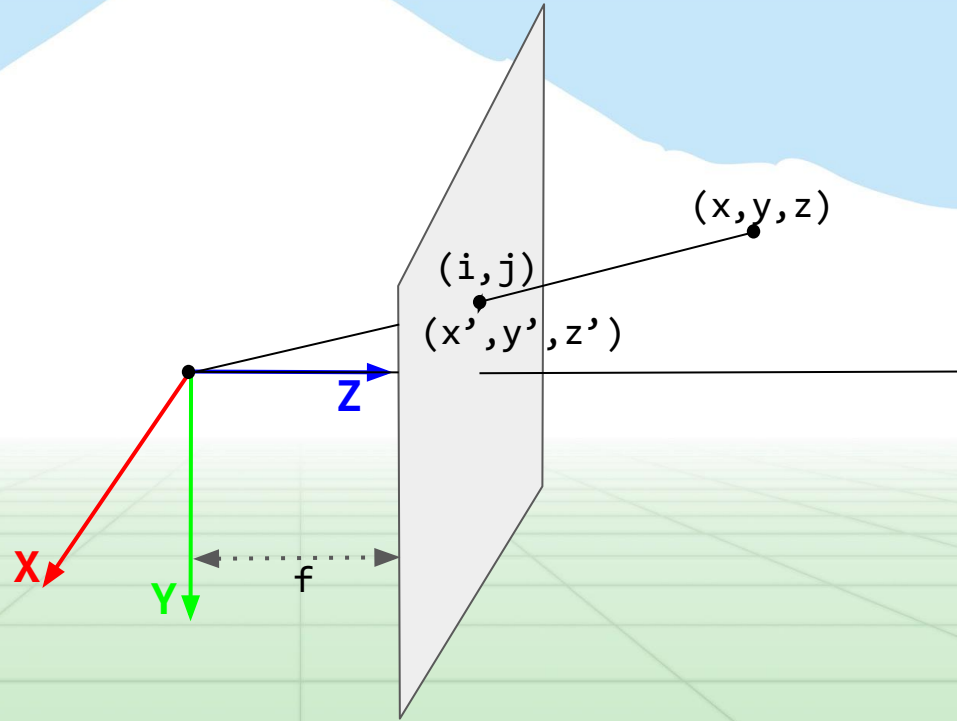


Image Coordinates

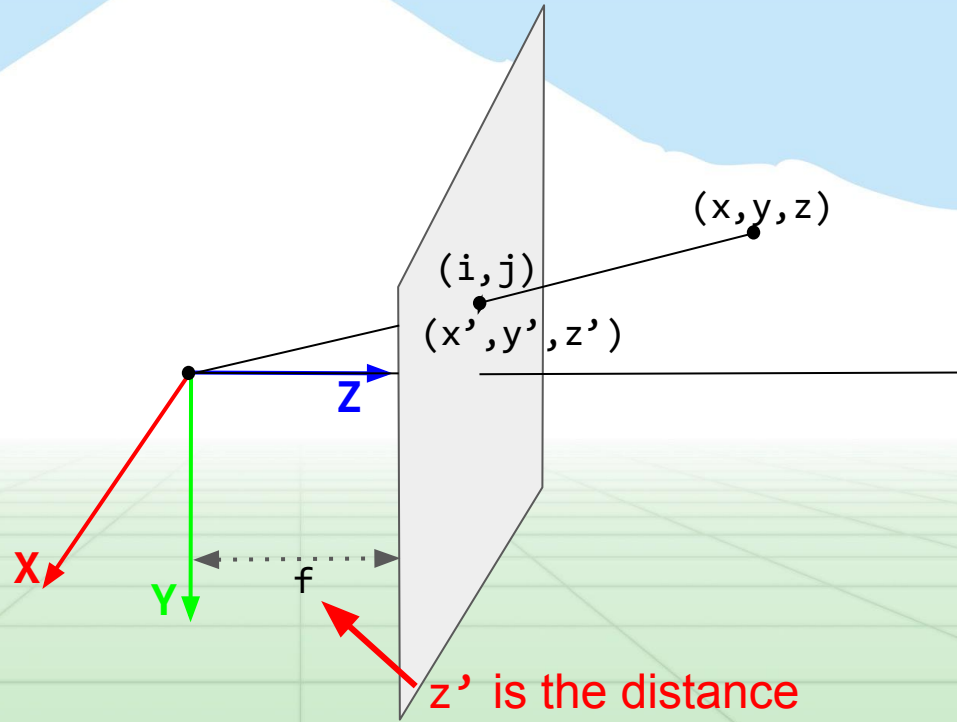


Image Coordinates

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

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

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

$$z' = f$$

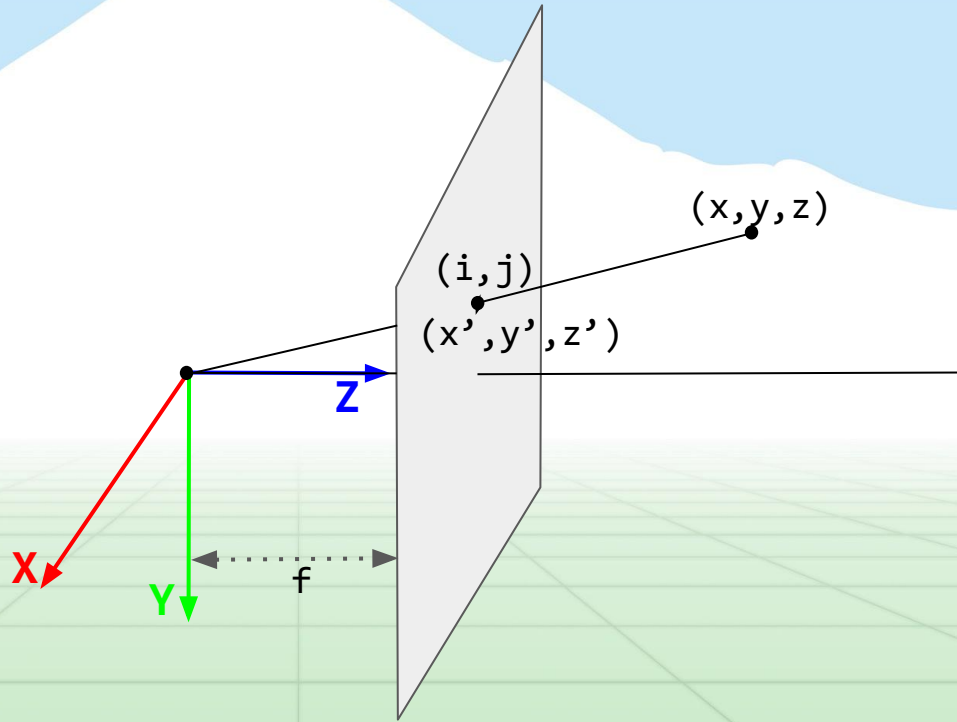


Image Coordinates

Re-scale (meters -> pixels)

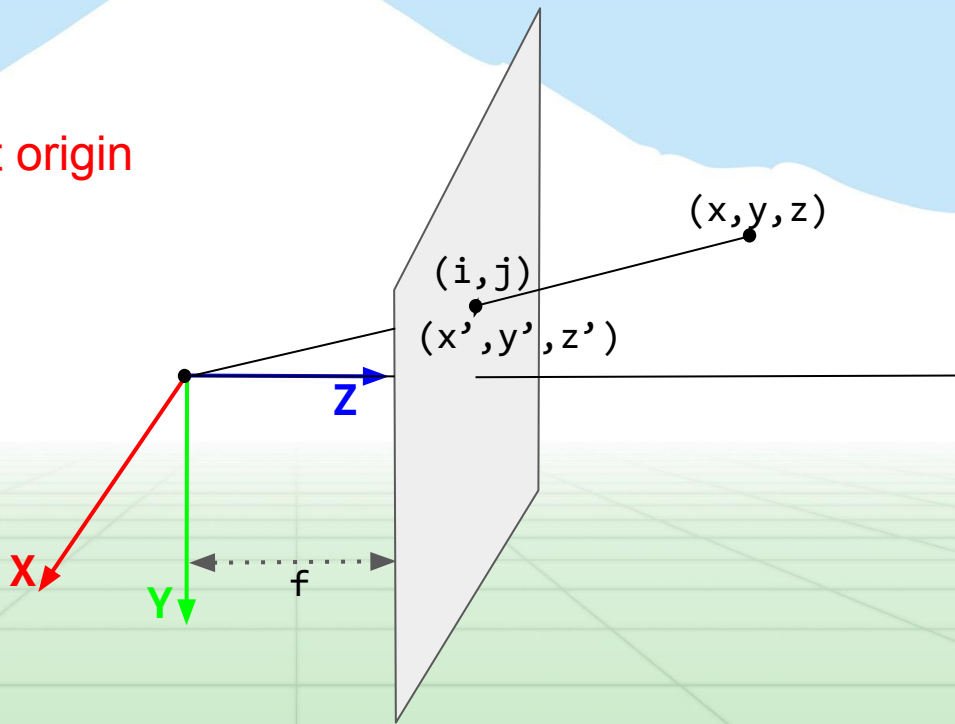
$$\begin{aligned} i &= px' + \frac{w}{2} \\ j &= py' + \frac{h}{2} \end{aligned}$$

Shift origin

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

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

$$z' = f$$



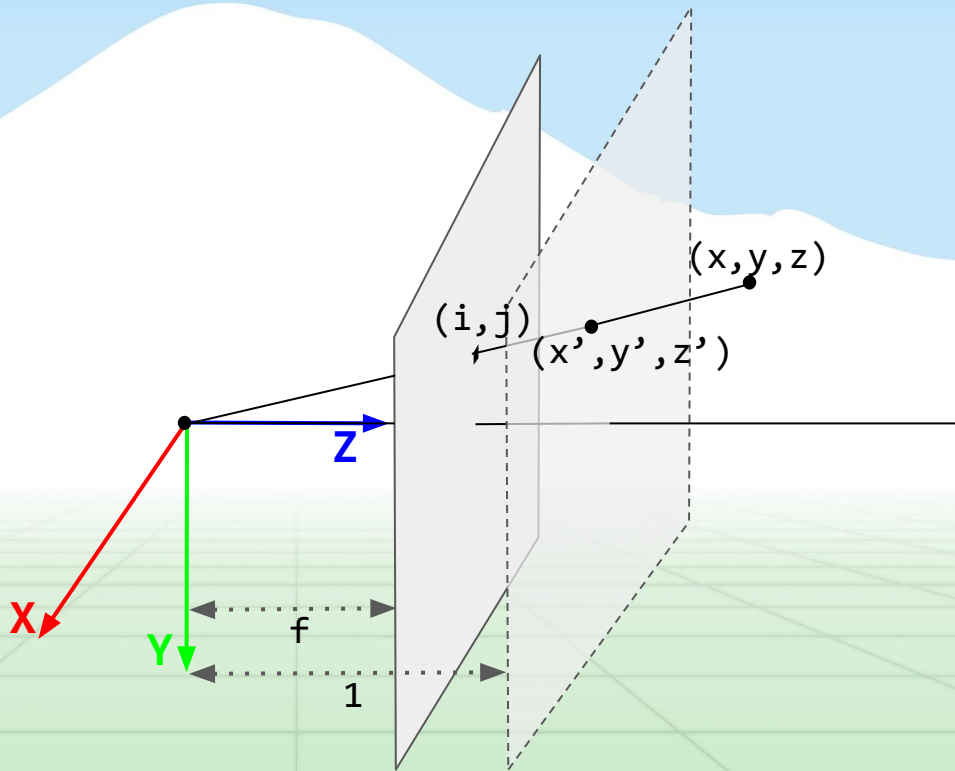
Normalized Image Coordinates

$$i = f p x' + \frac{w}{2}$$
$$j = f p y' + \frac{h}{2}$$

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

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

$$z' = 1$$



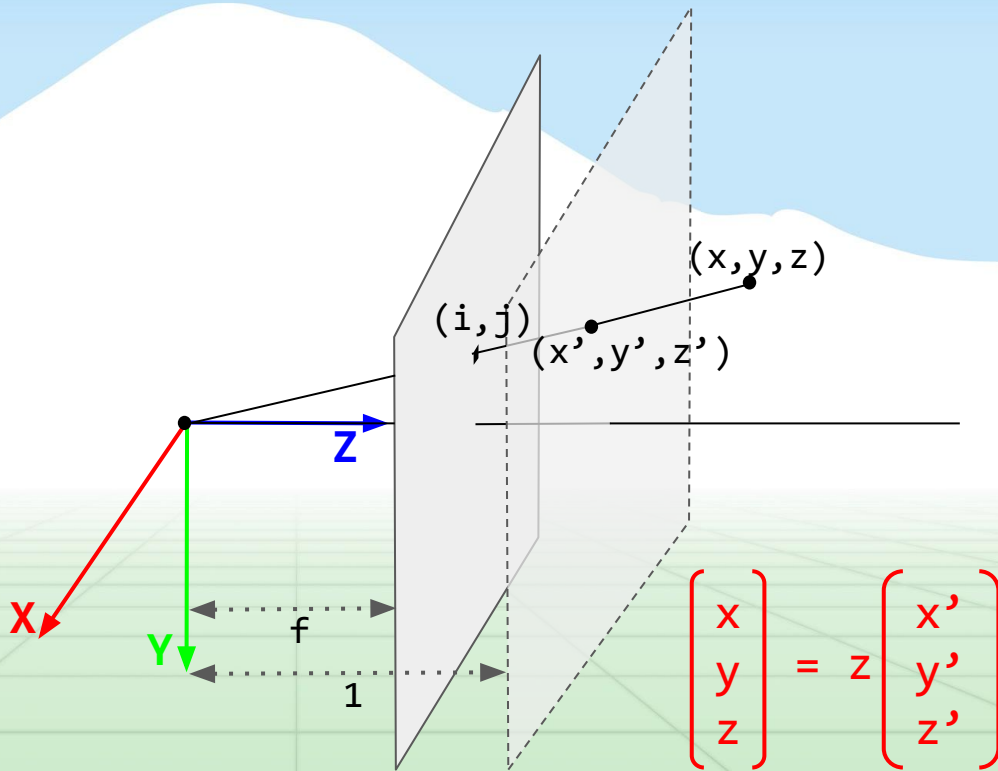
Normalized Image Coordinates

$$i = f p x' + \frac{w}{2}$$
$$j = f p y' + \frac{h}{2}$$

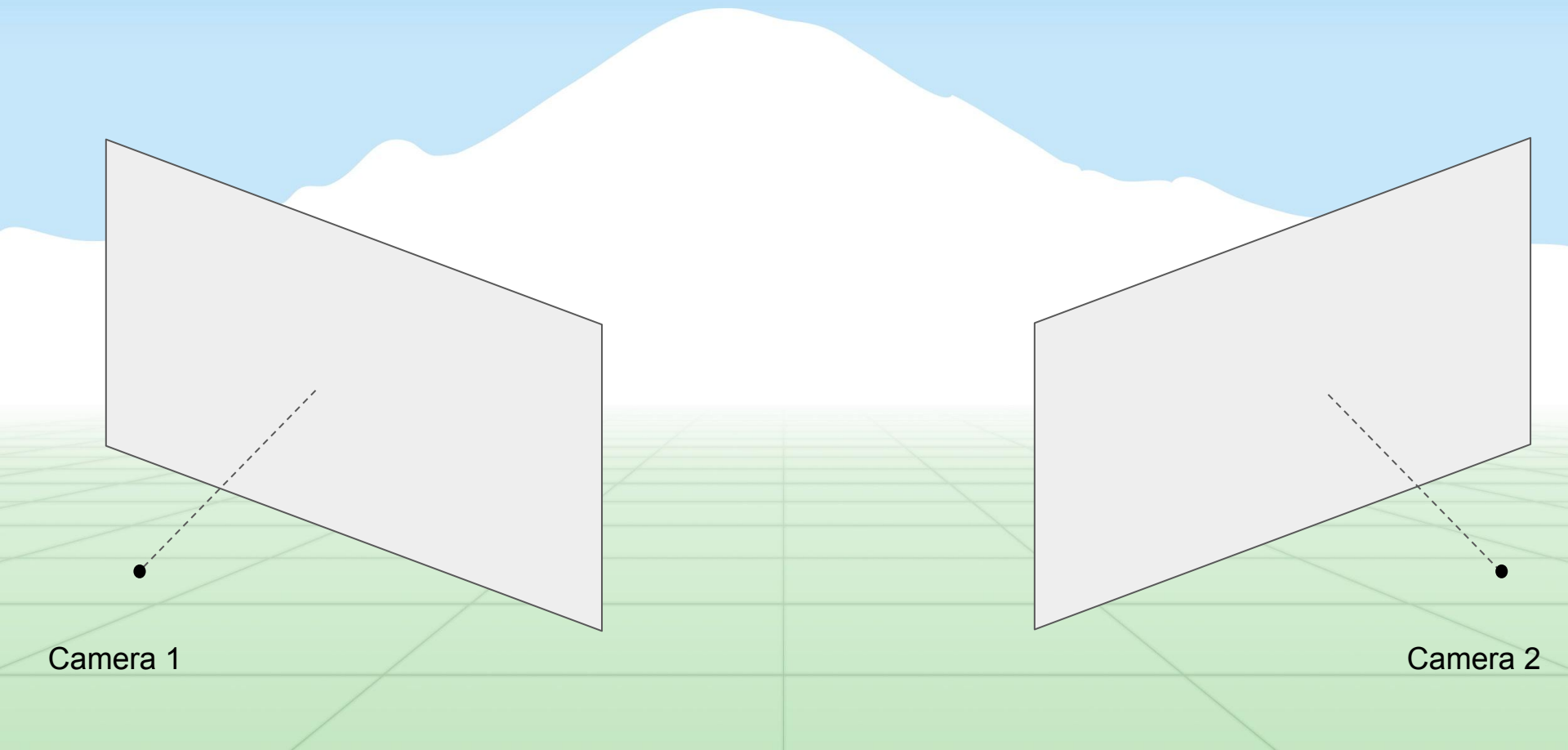
$$x' = \frac{x}{z}$$

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

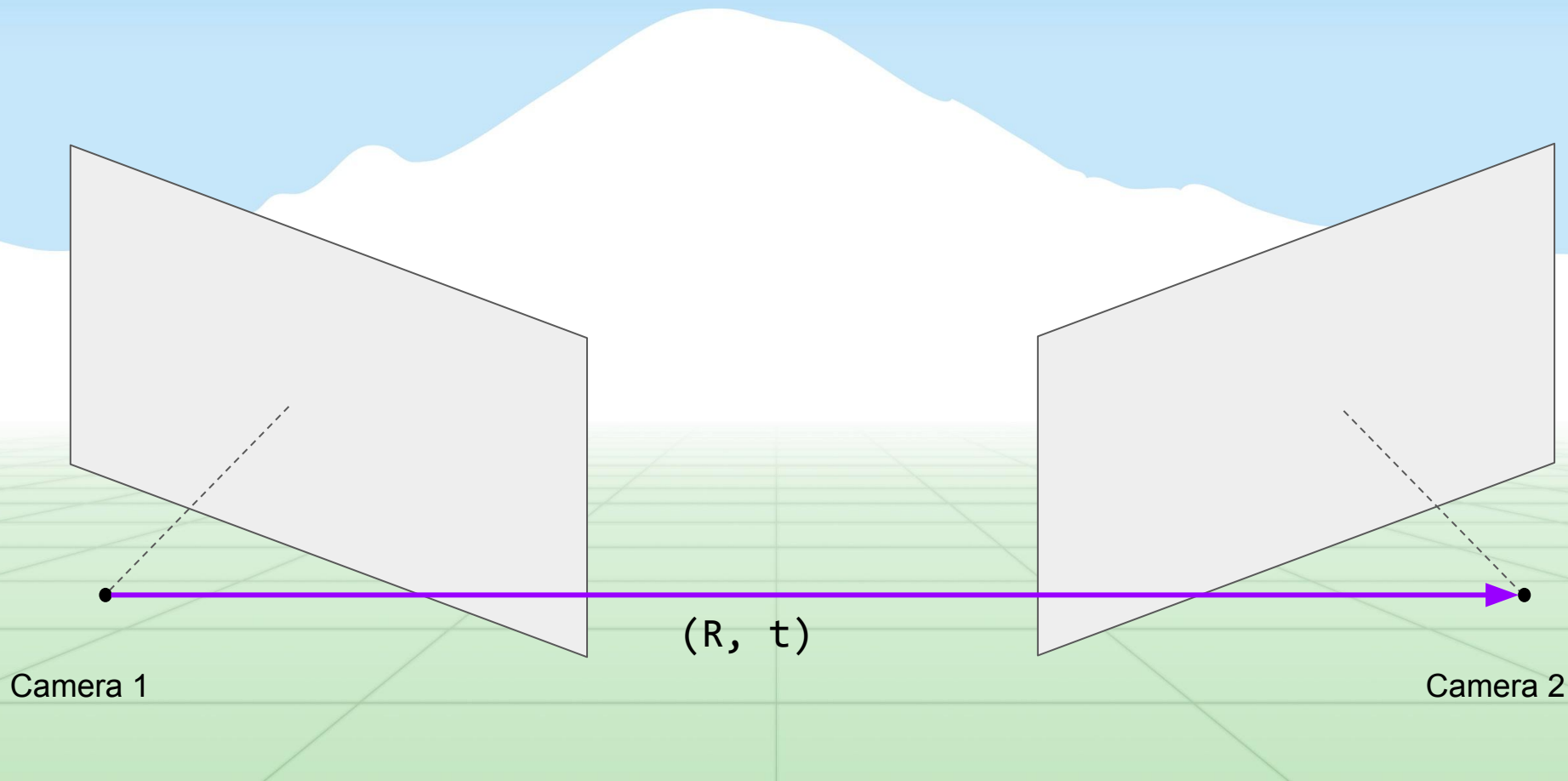
$$z' = 1$$



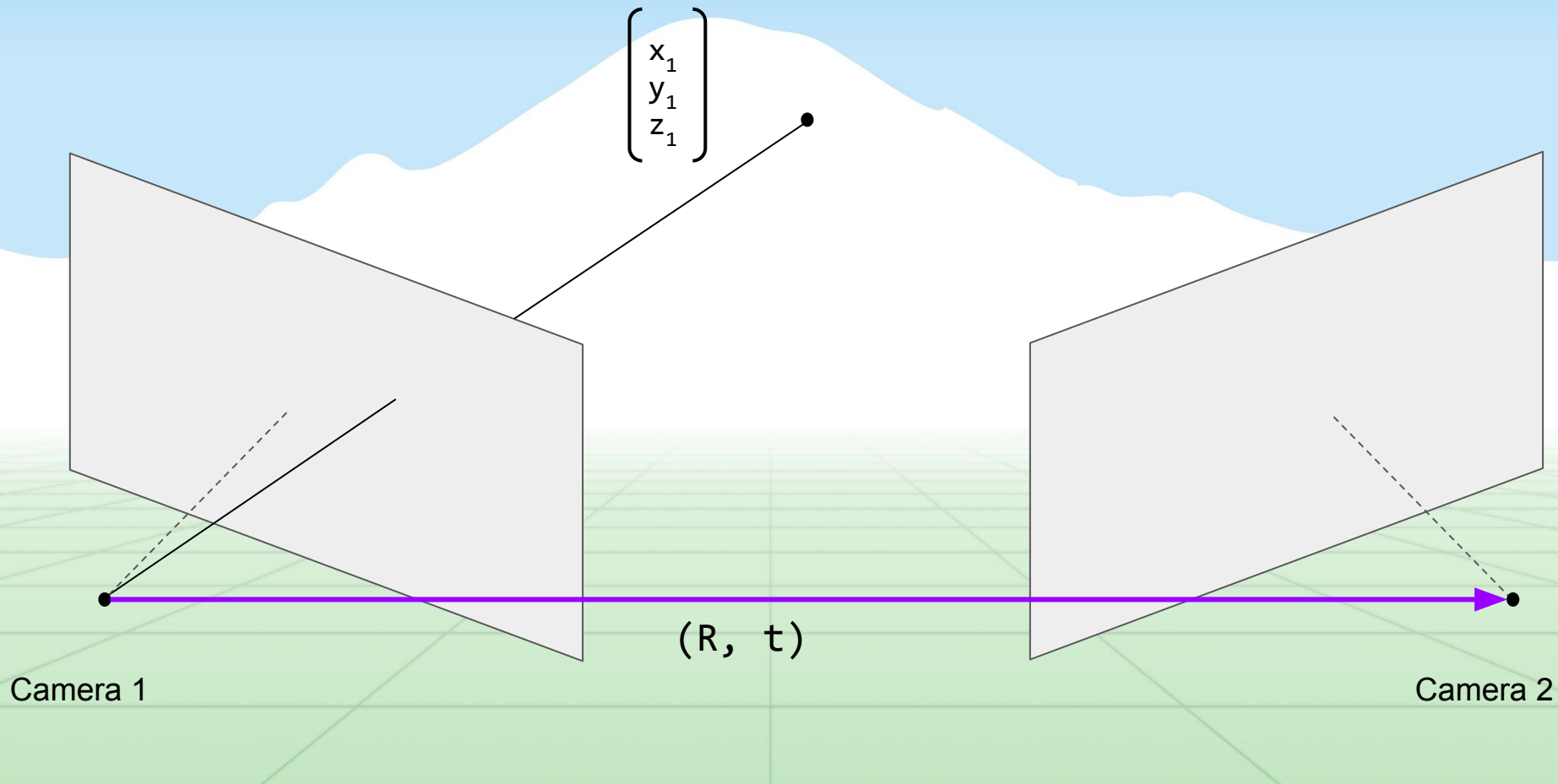
Stereo



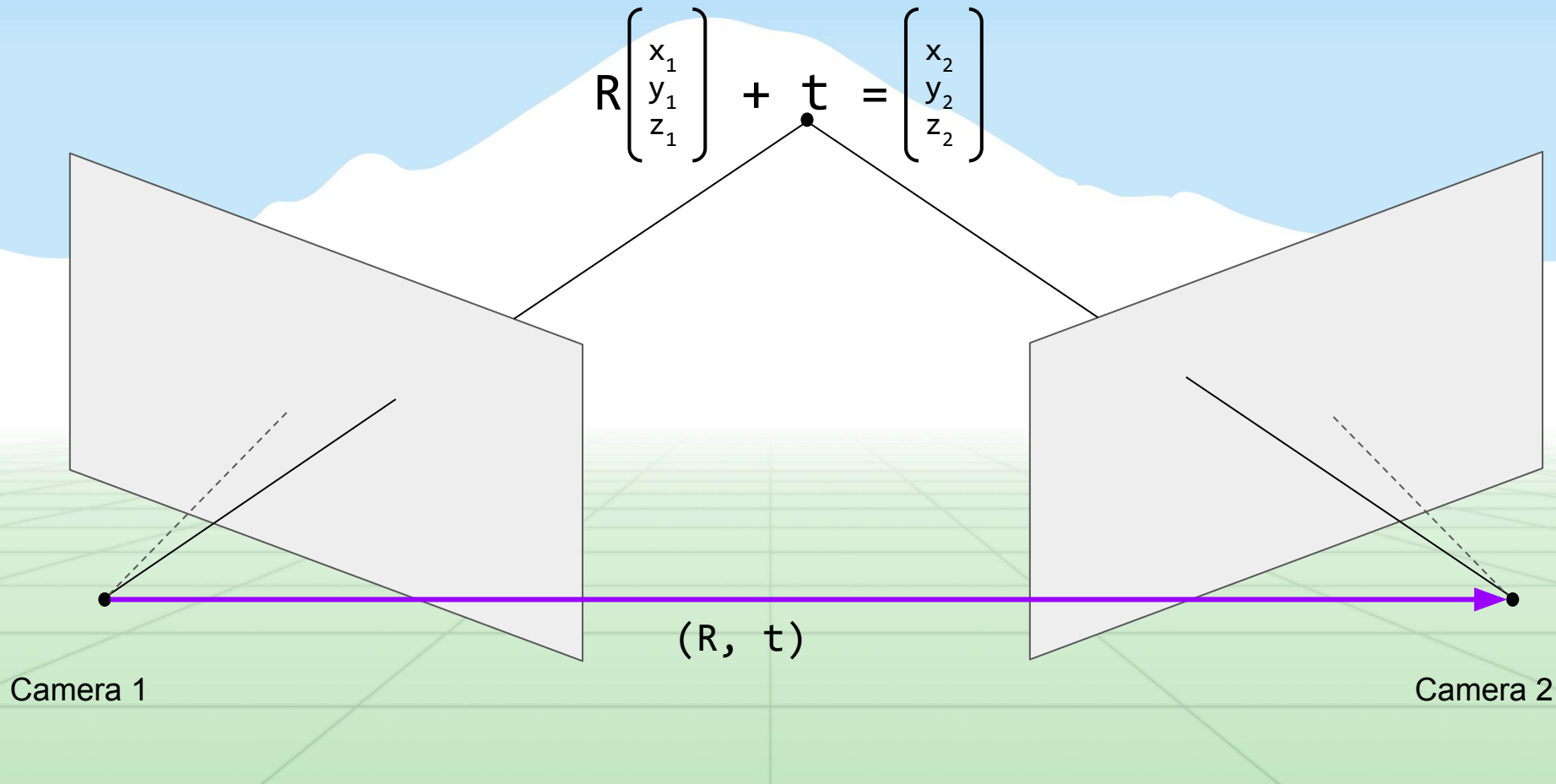
Stereo



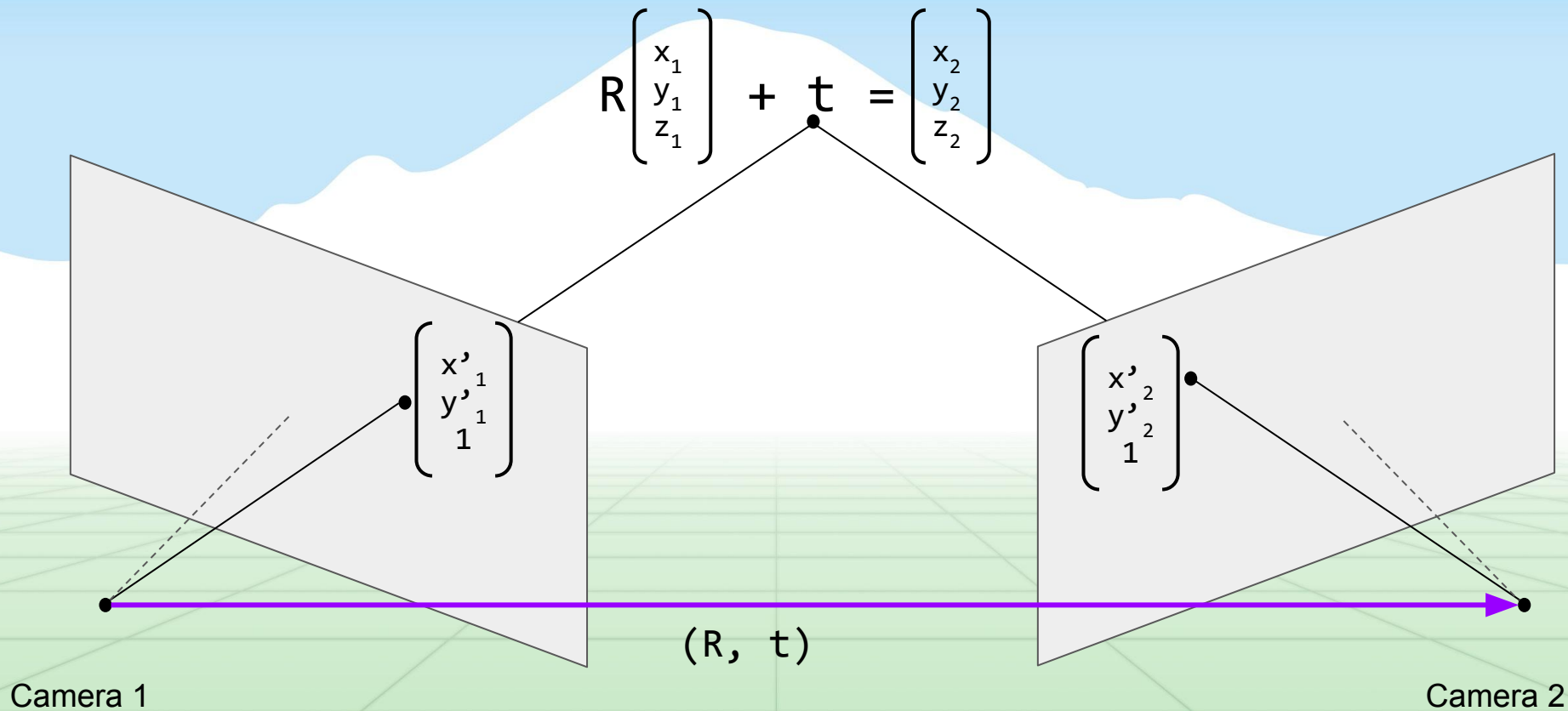
Stereo



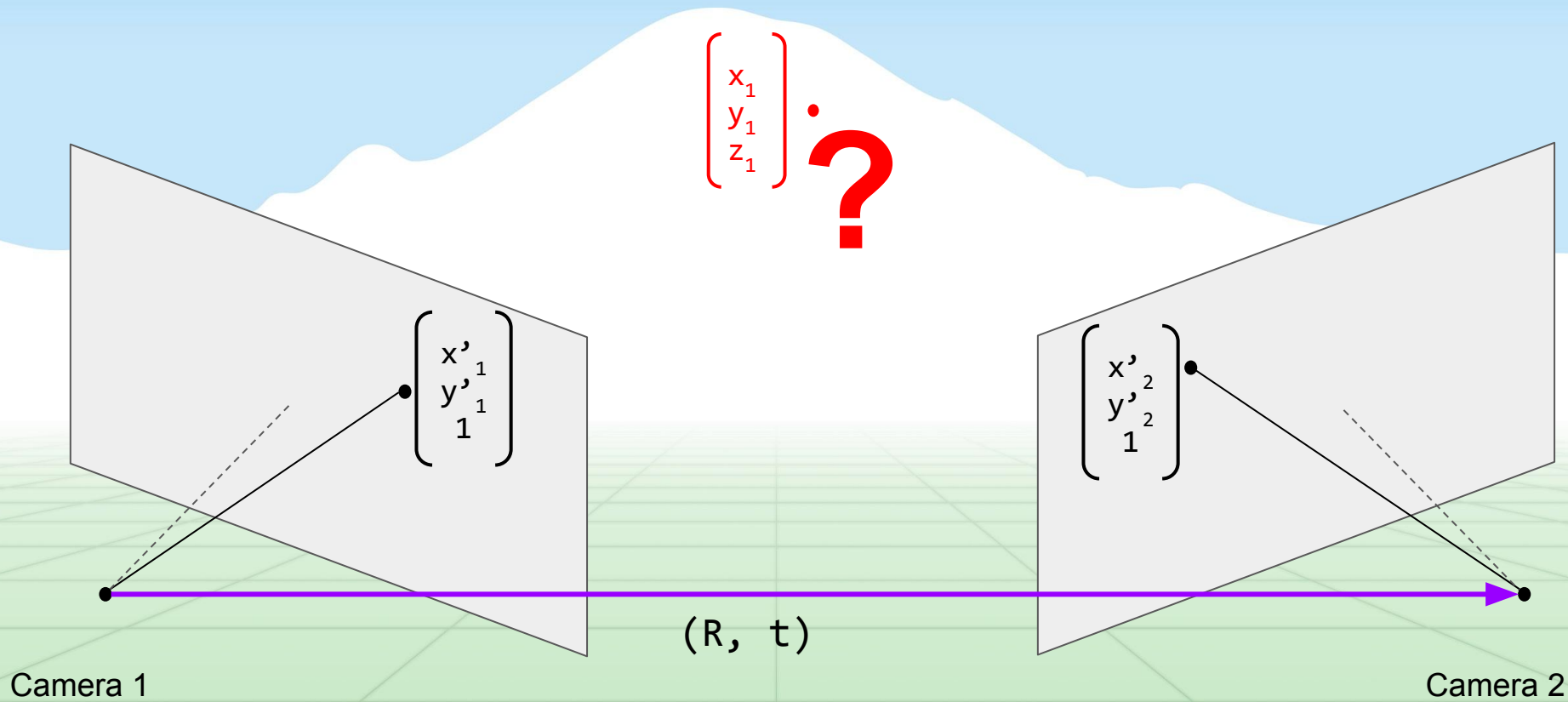
Stereo



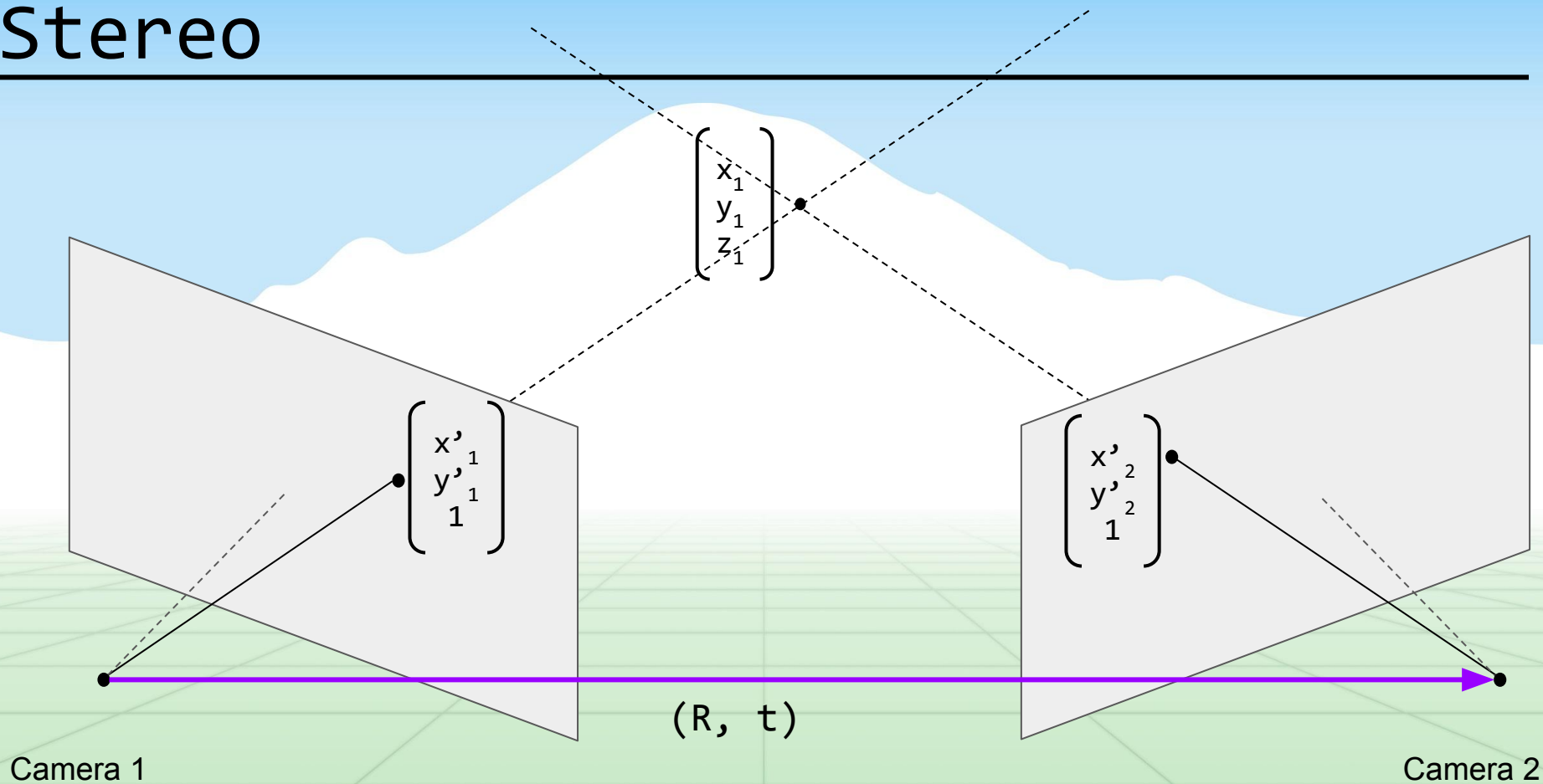
Stereo



Stereo



Stereo



Stereo

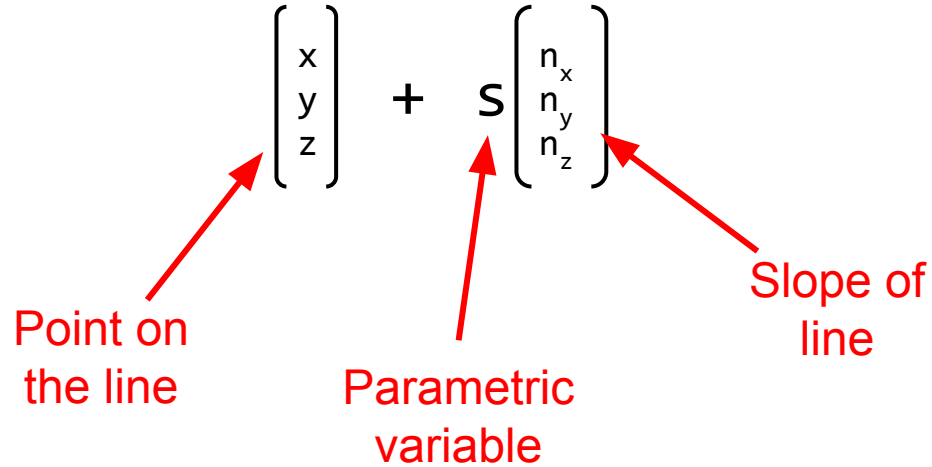
A line in 3D can be represented as:

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

Point on the line

Parametric variable

Slope of line

The diagram shows the parametric equation for a 3D line. A red arrow points from the text 'Point on the line' to the vector [x, y, z]. Another red arrow points from 'Parametric variable' to the scalar 's'. A third red arrow points from 'Slope of line' to the direction vector [n_x, n_y, n_z].

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

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + S_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix} \xrightarrow[\text{1 coordinate frame}]{\text{Transform to camera}} -t + R^{-1} S_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$$

Stereo

A line in 3D can be represented as:

Line 1

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + S_1 \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix}$$

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

Line 2

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + S_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$$

Transform to camera
1 coordinate frame

$$\xrightarrow{\quad} -t + R^{-1} S_2 \begin{bmatrix} x'_2 \\ y'_2 \\ 1 \end{bmatrix}$$

Stereo

A line in 3D can be represented as:

Line 1

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

Line 2

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

Stereo

A line in 3D can be represented as:

Line 1

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

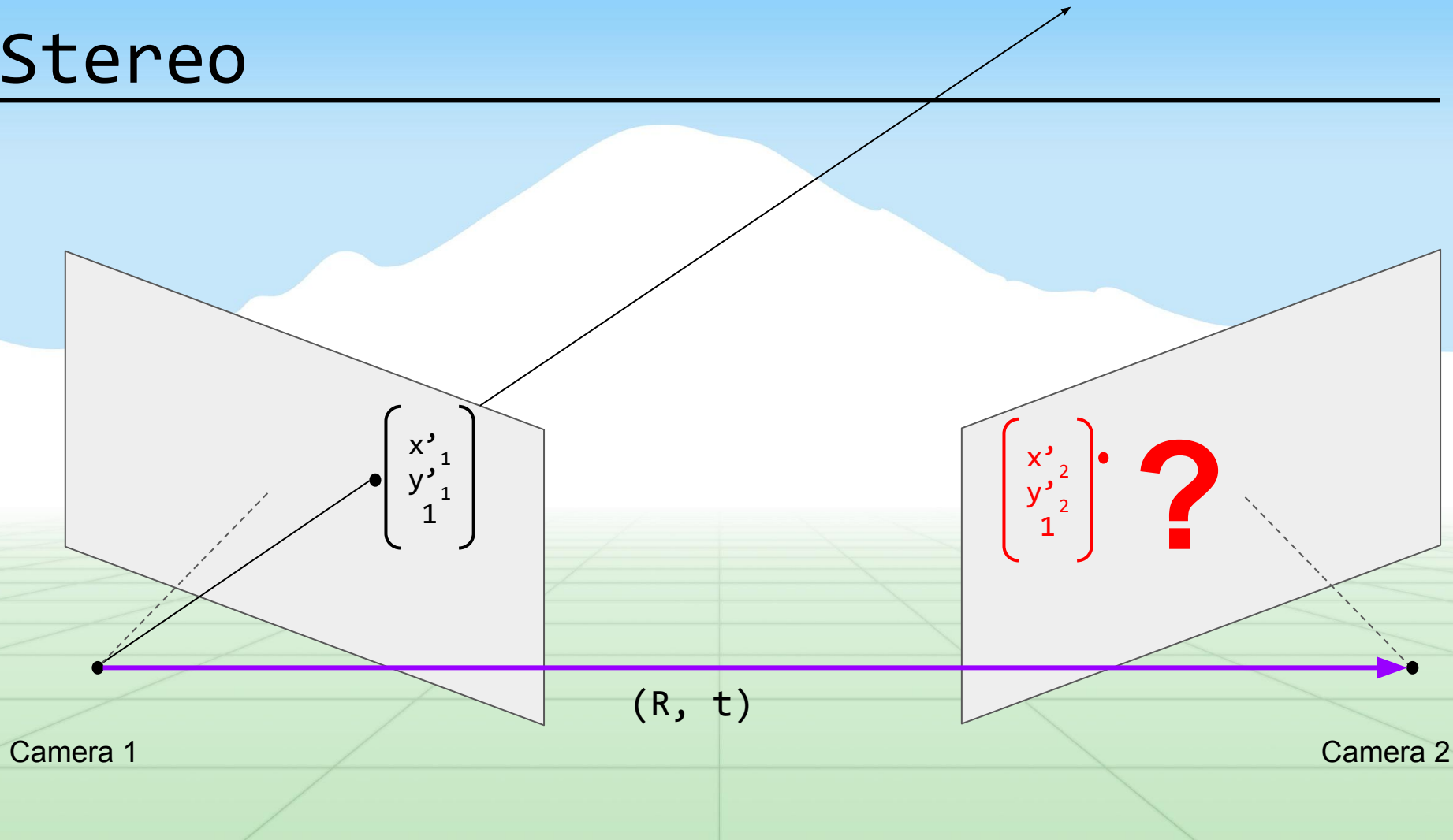
Line 2

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

Solve system of linear equations

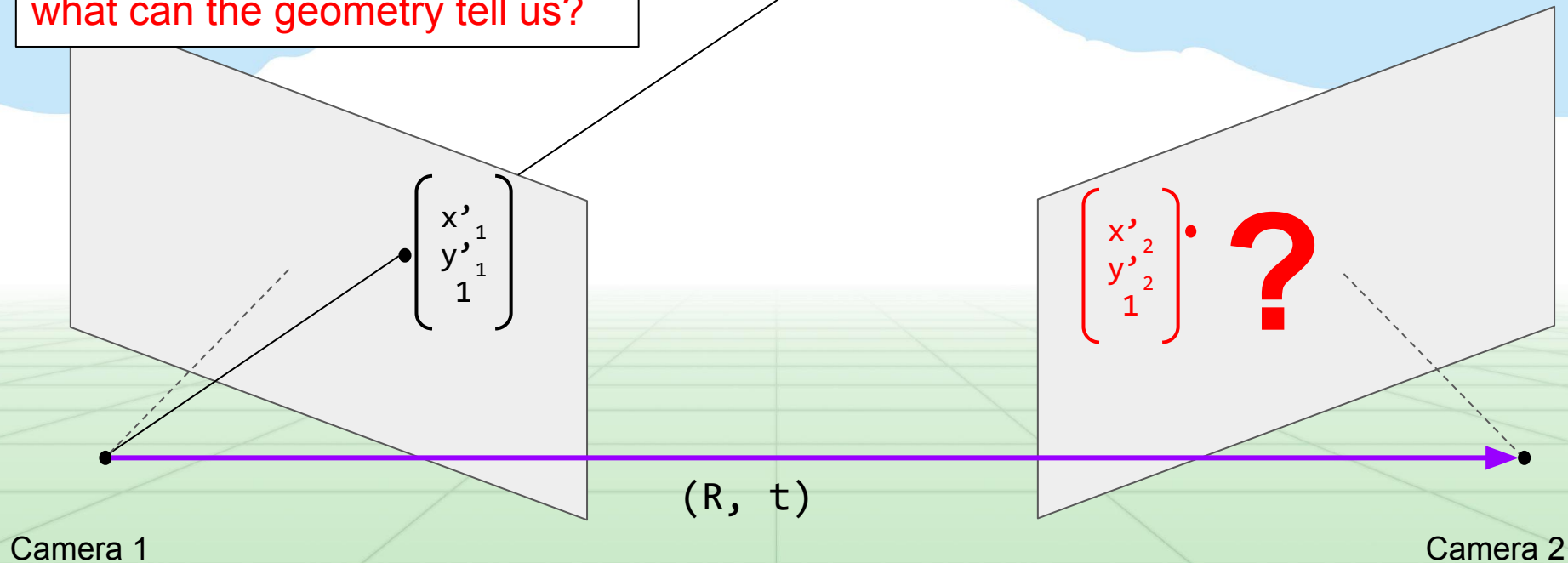
$$z_1 \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} = -\mathbf{t} + \mathbf{R}^{-1} z_2 \begin{pmatrix} x'_2 \\ y'_2 \\ 1 \end{pmatrix}$$

Stereo

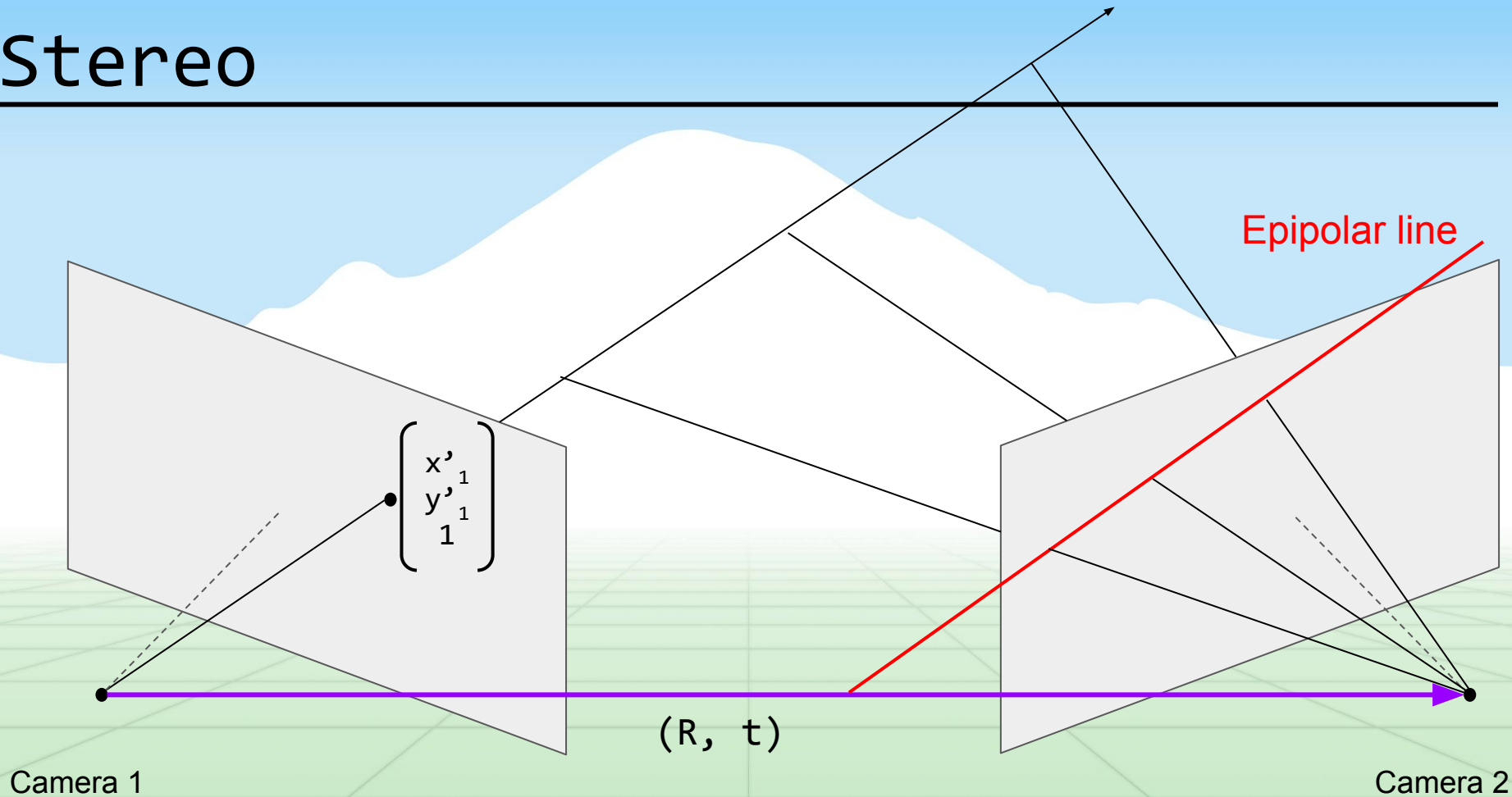


Stereo

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

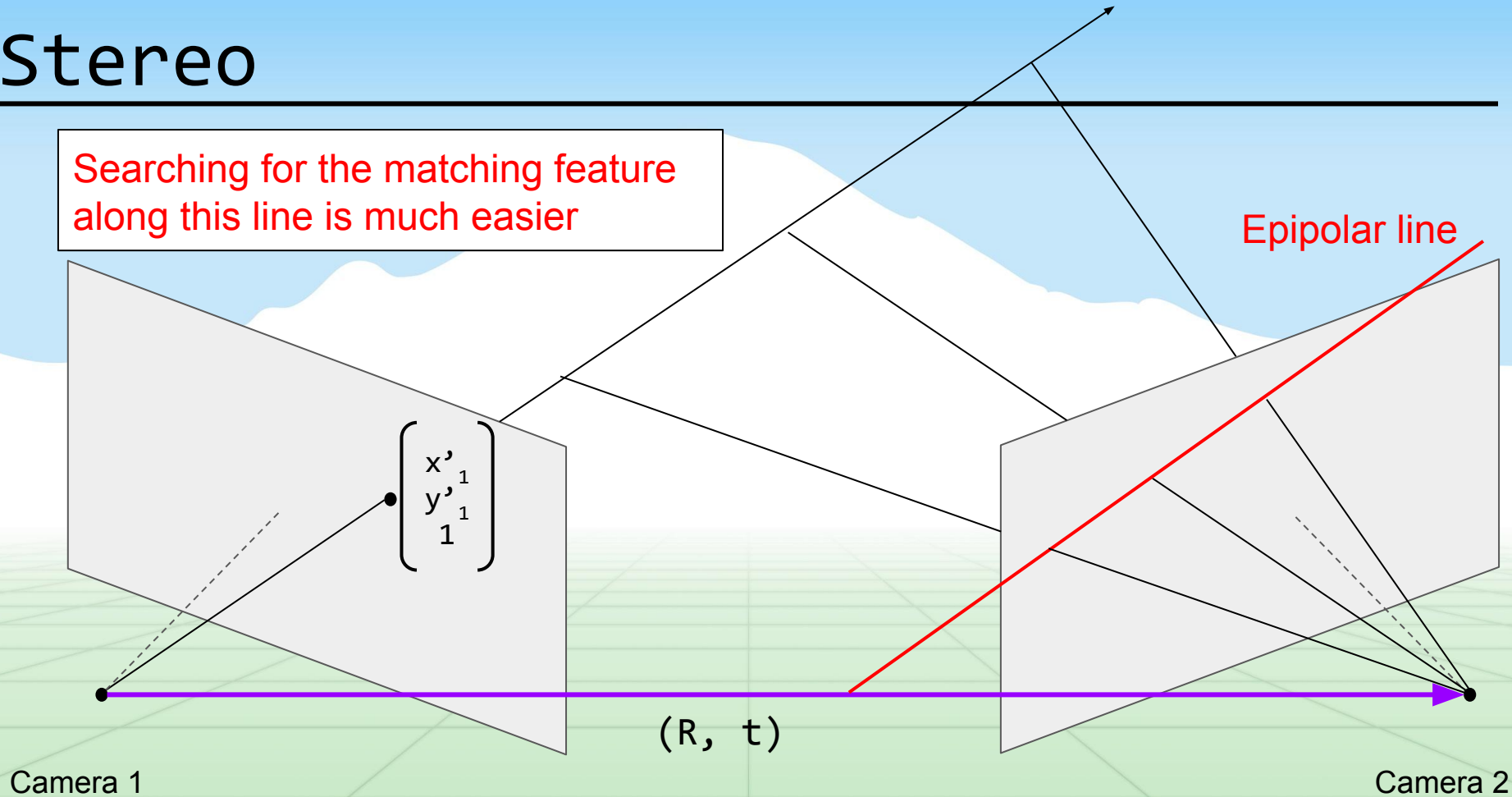


Stereo

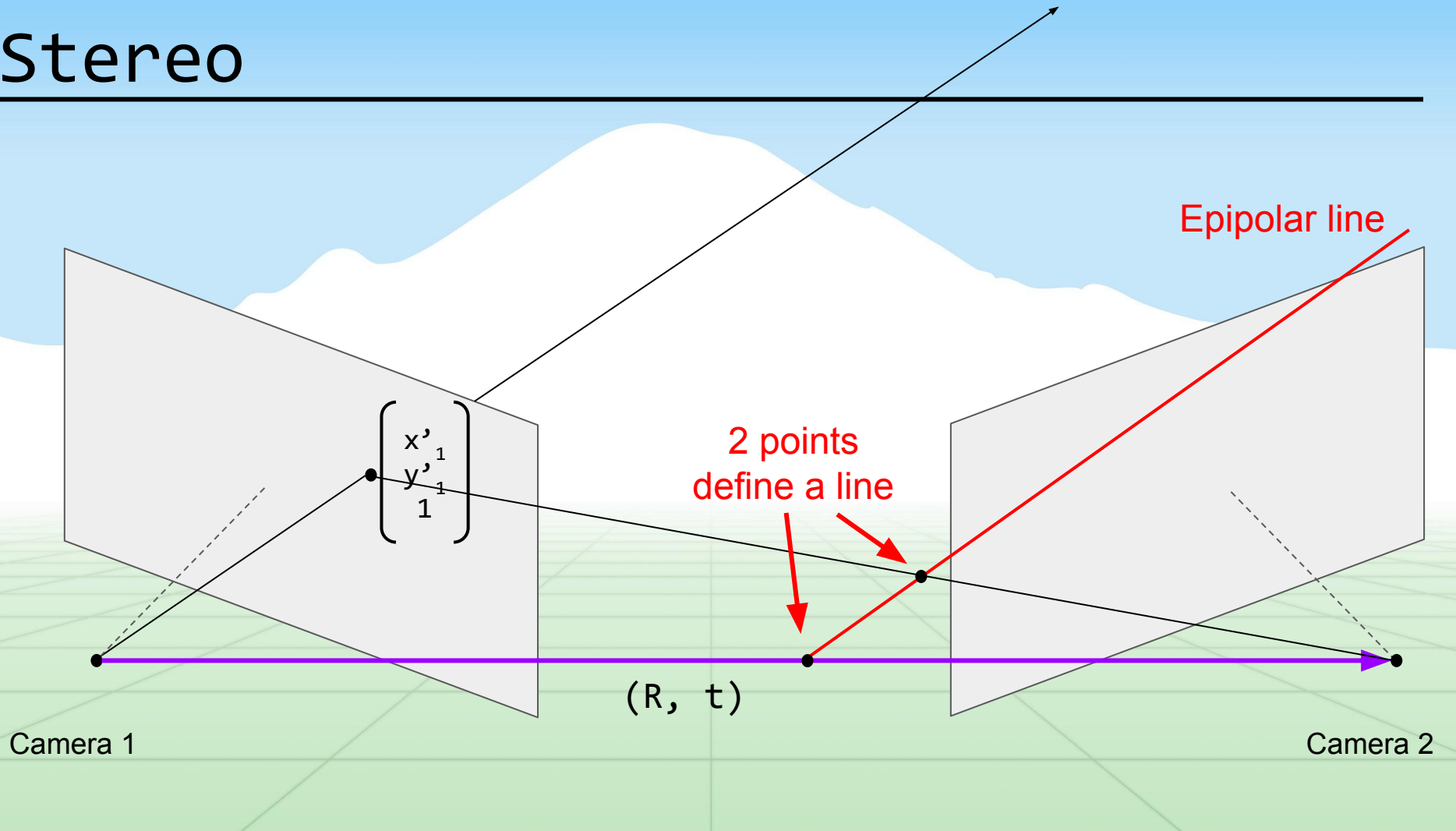


Stereo

Searching for the matching feature along this line is much easier



Stereo



Stereo

$$\begin{bmatrix} t \\ t \\ t \end{bmatrix} + R \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} \frac{1}{z''_1}$$

$$\begin{bmatrix} t \\ t \\ t \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \frac{1}{z'_1}$$

z coordinate
after transform

Stereo

$$\begin{bmatrix} t & + & R \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} \end{bmatrix} \frac{1}{z''_1}$$

$$\begin{bmatrix} t & + & R \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix} \frac{1}{z'_1} \xrightarrow{\text{Simplify}} t \frac{1}{t_z}$$

Stereo

2D equation for a line:

$$y = mx + b$$

Camera

Camera 2

Stereo

2D equation for a line:

$$y = mx + b$$

Doesn't handle
vertical lines

Camera

Camera 2

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

Both our projected points have the format
We can pretend they are points in 2D.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} t \\ t \end{bmatrix} + R \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} \frac{1}{z''_1}$$
$$t \frac{1}{t_z}$$

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

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

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

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

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

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

Stereo

2D equation for a line:

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

$$ax + by + c = 0$$

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

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

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

Camera 2

Stereo

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

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

$$ax + by + c = 0$$

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

Stereo

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

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

$$sax + sby + sc = 0$$

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

Stereo

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

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

$$sax + sby + sc = 0$$

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

Stereo

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

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

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

Stereo

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

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

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

essential matrix
(Longuet-Higgins, 1981)

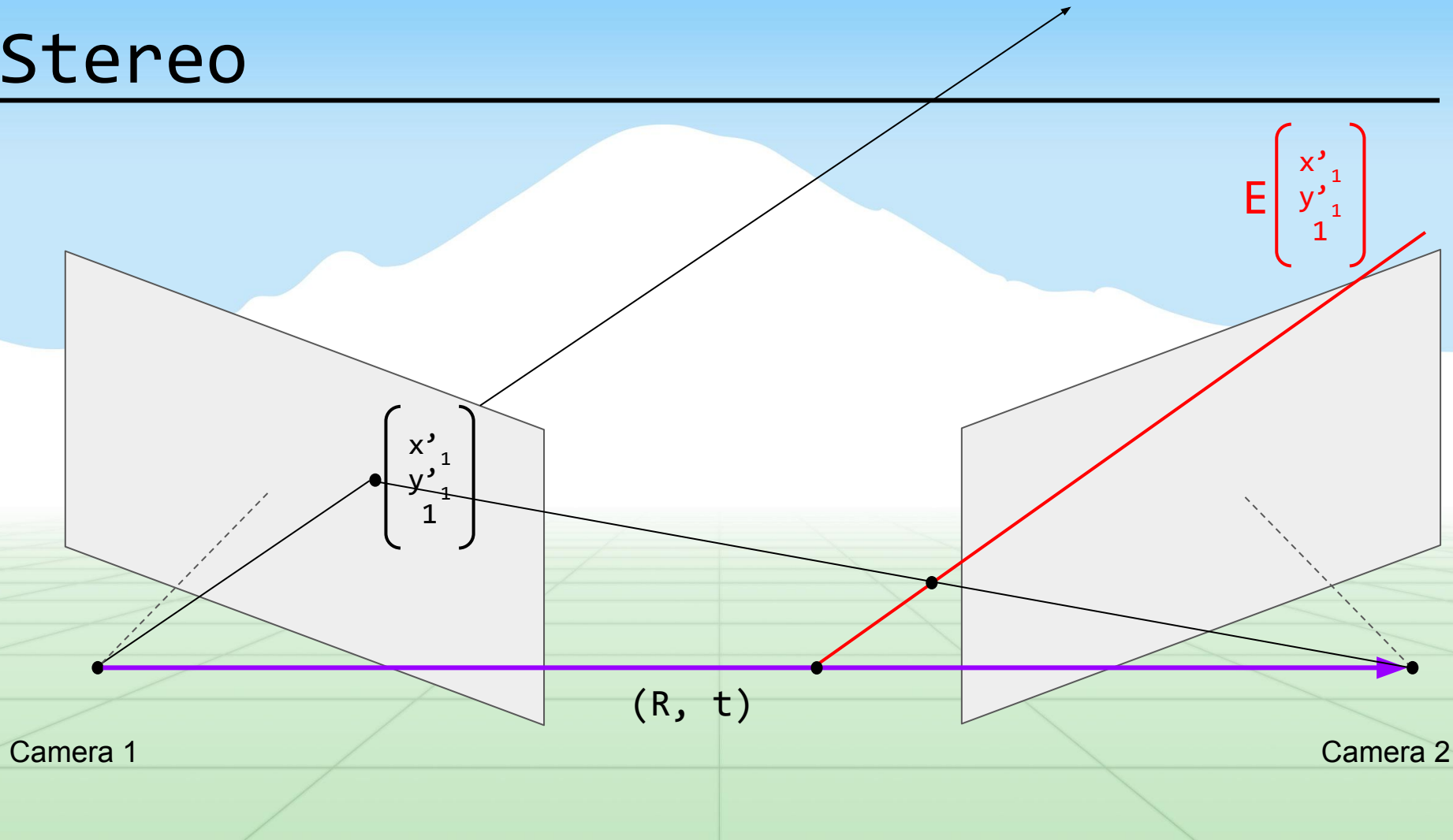
Stereo

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

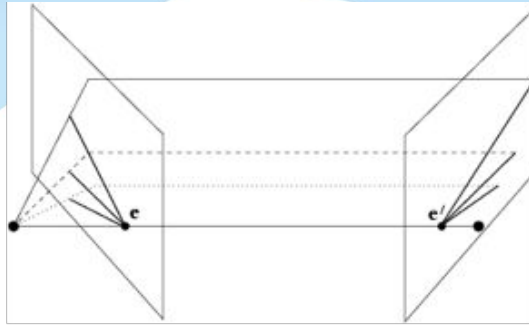
$$\frac{1}{z''_1 t_z} t \times R \begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

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

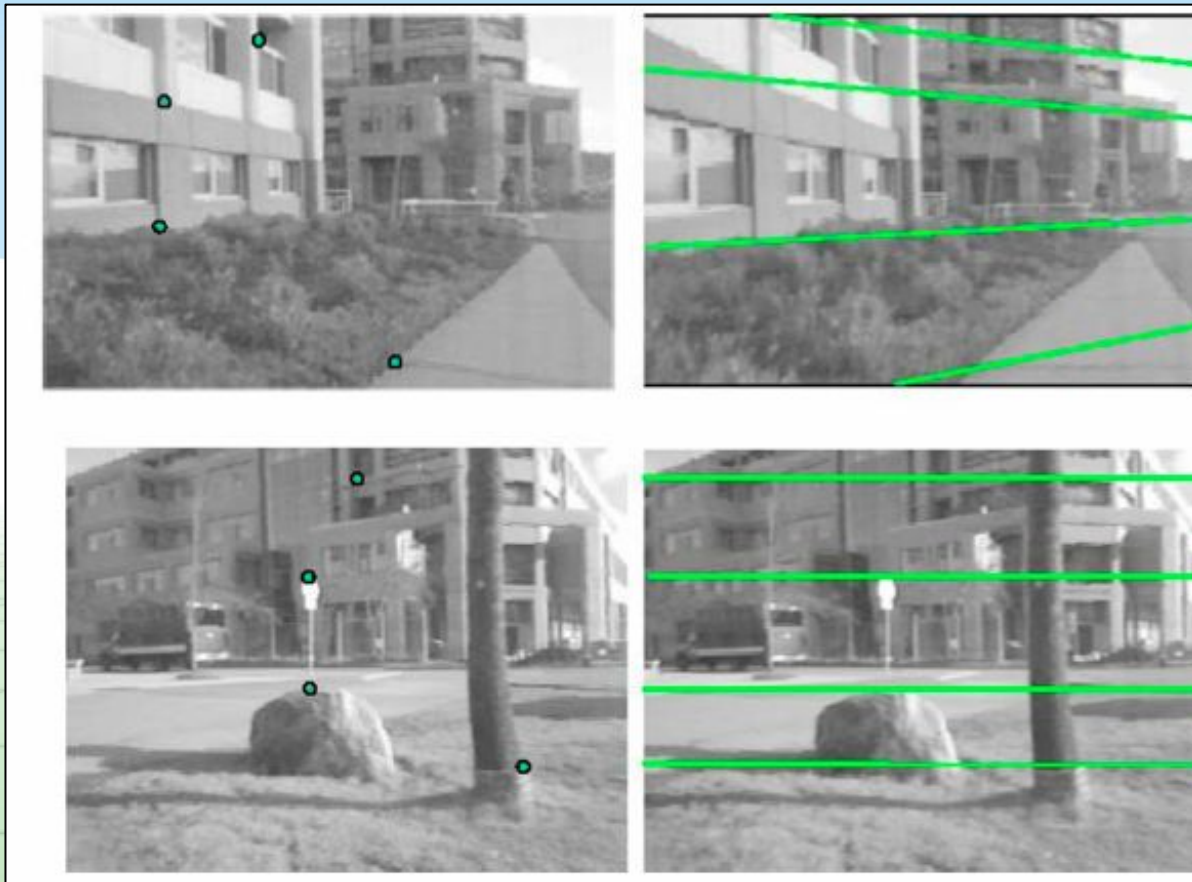
Stereo



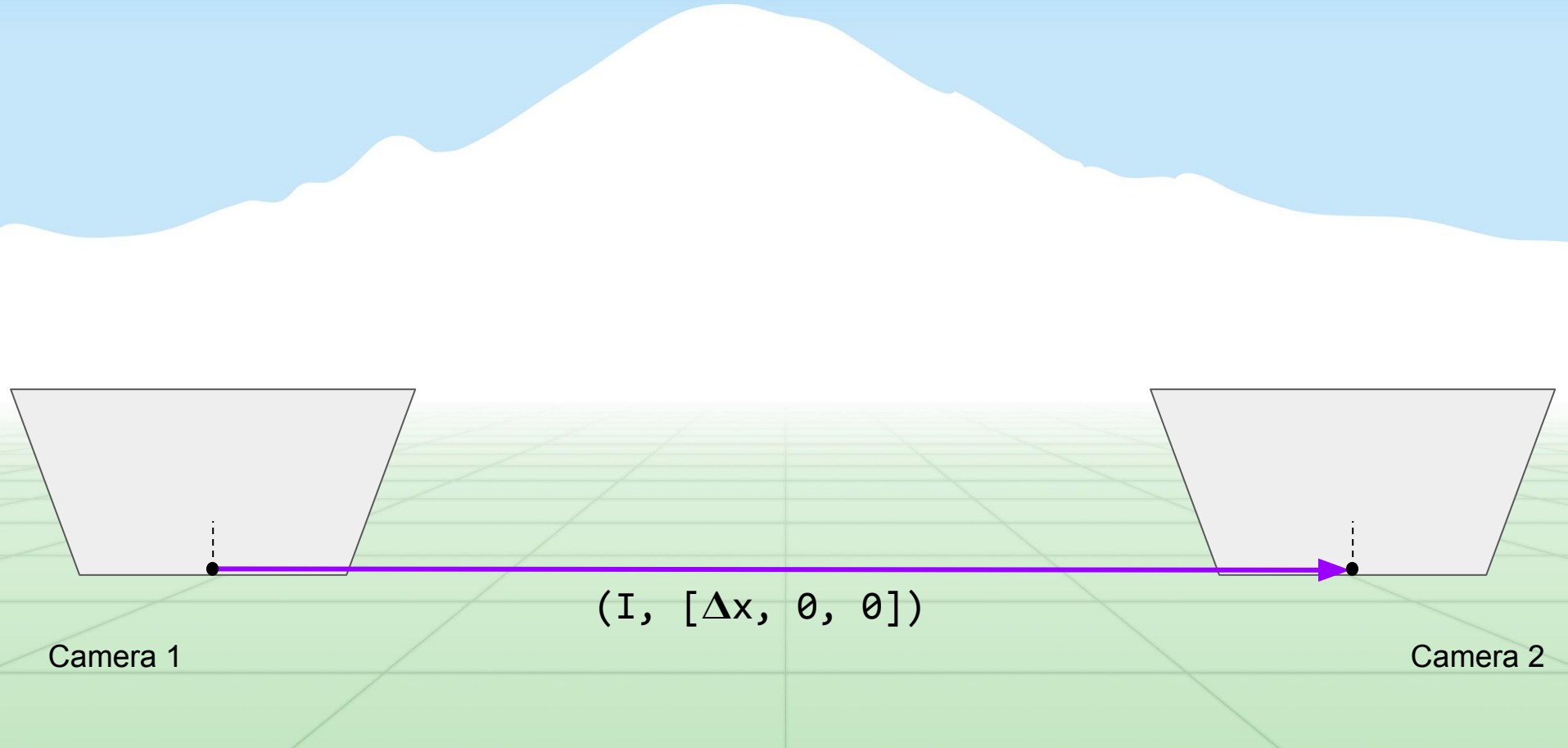
Stereo



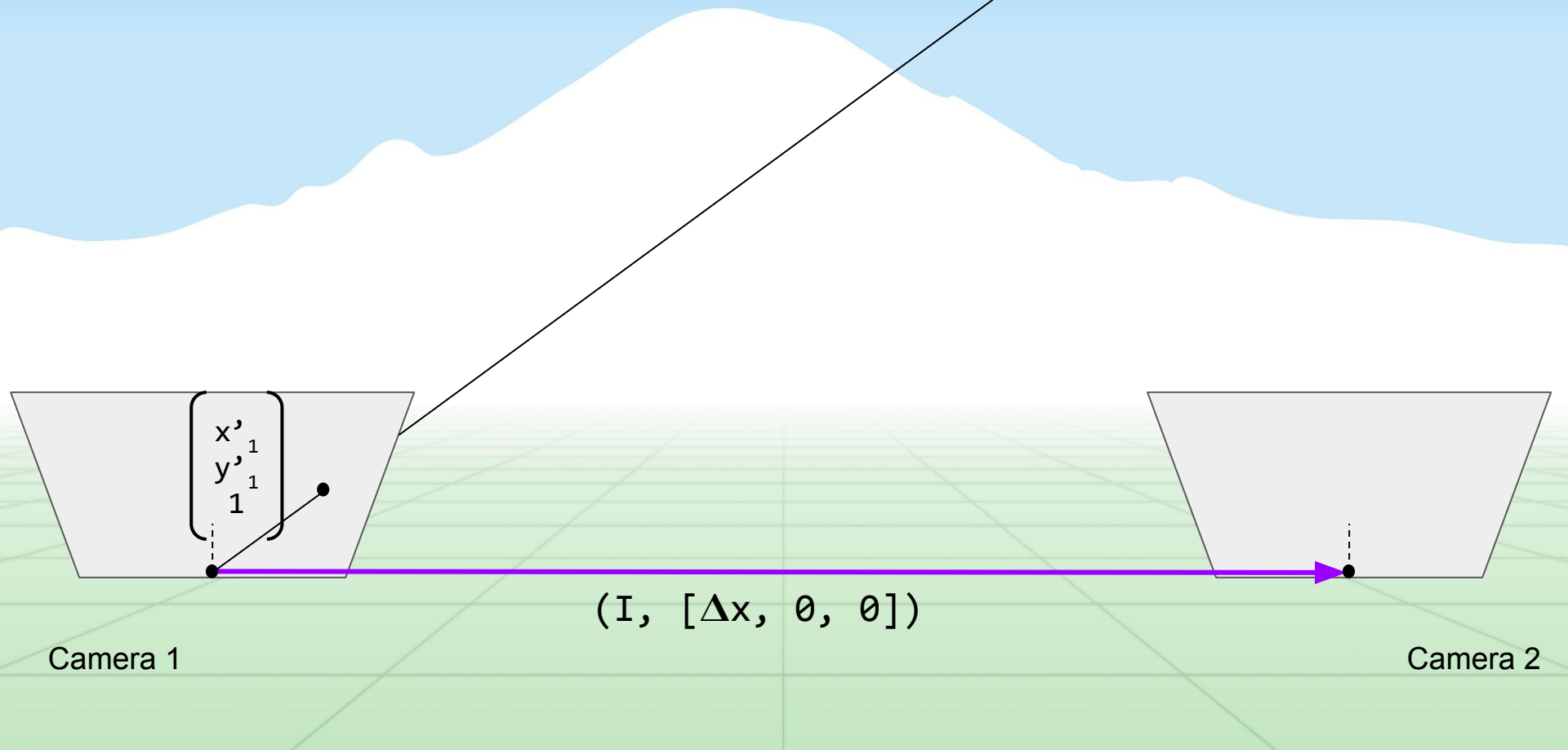
Stereo



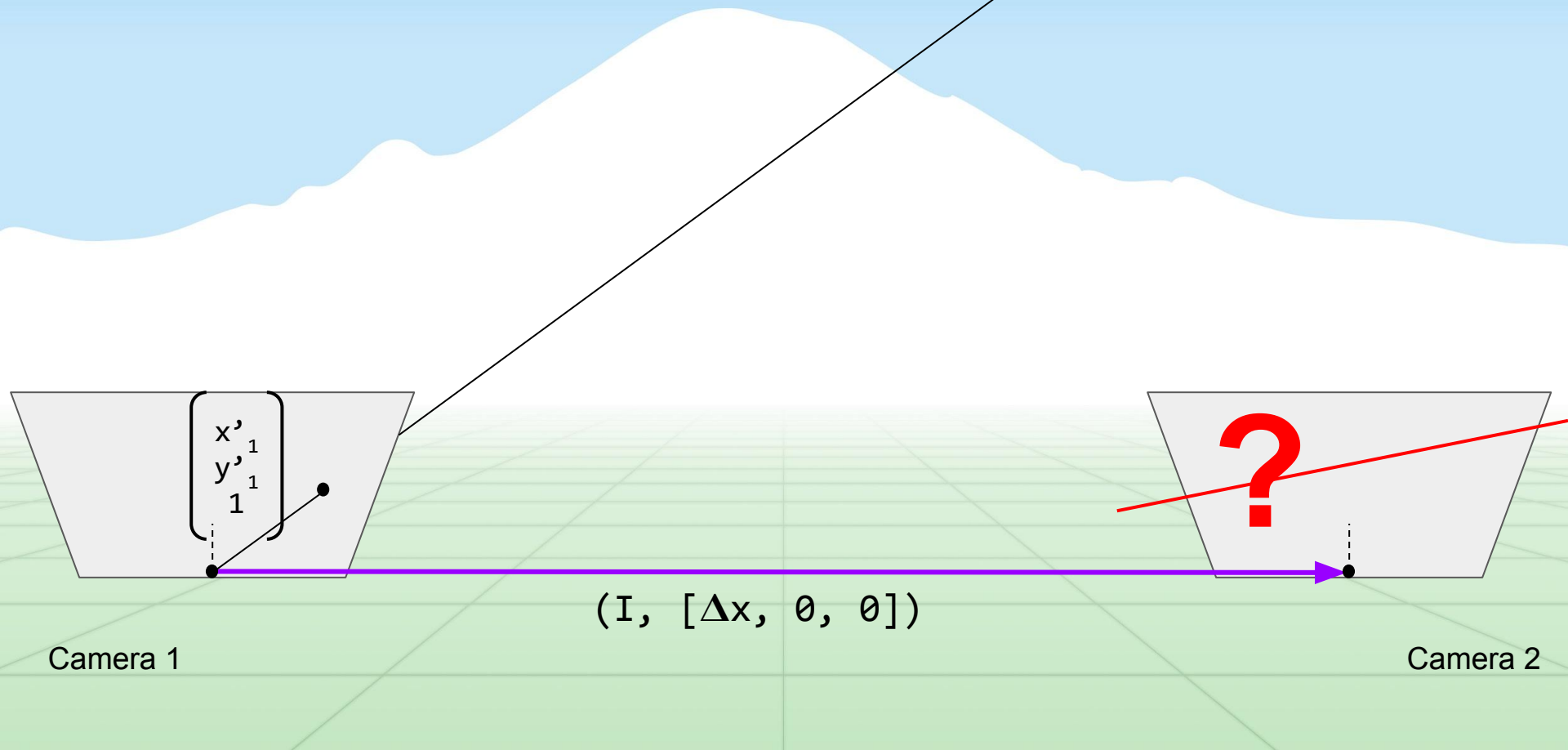
Stereo: Parallel Image Planes



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

Camera 2

Stereo: Parallel Image Planes

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

Stereo: Parallel Image Planes

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

Stereo: Parallel Image Planes

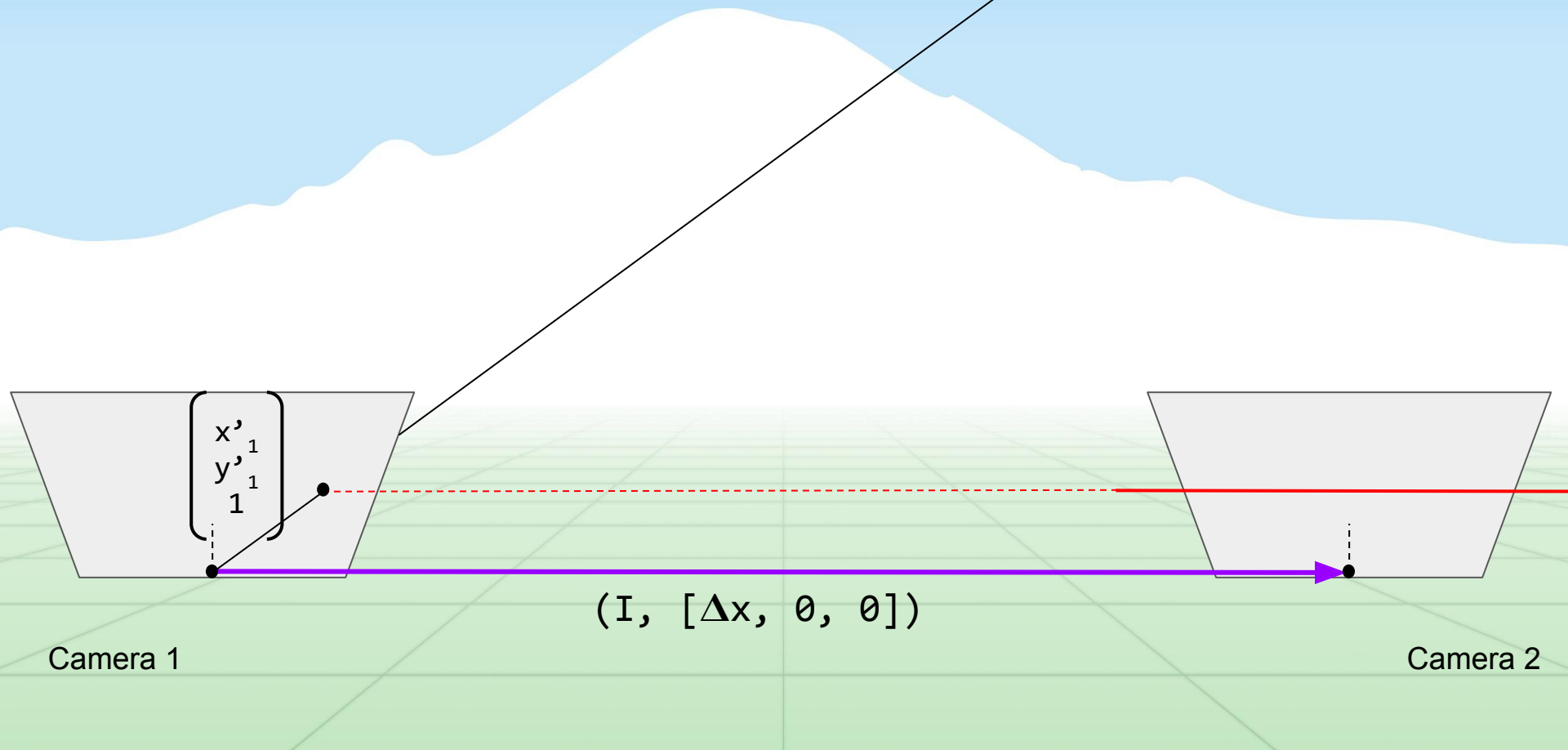
$$\begin{aligned} \begin{pmatrix} \Delta x \\ \theta \\ \theta \end{pmatrix} \times \mathbf{I} \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \\ \begin{pmatrix} \Delta x \\ \theta \\ \theta \end{pmatrix} \times \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \\ \begin{pmatrix} \theta \\ -\Delta x \\ \Delta xy'_1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \end{aligned}$$

Stereo: Parallel Image Planes

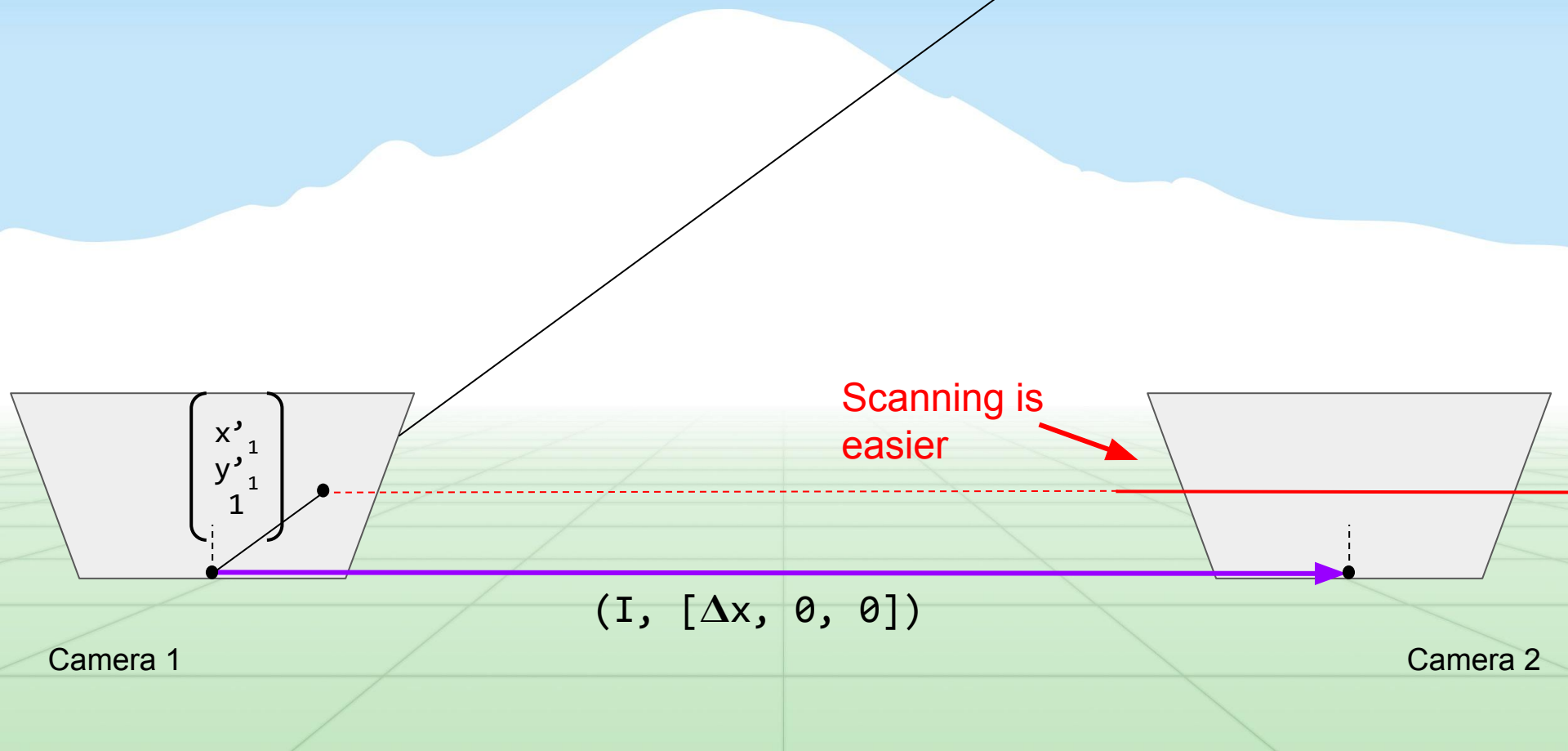
$$\begin{aligned} \begin{pmatrix} \Delta x \\ \theta \\ \theta \end{pmatrix} \times \mathbf{I} \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \\ \begin{pmatrix} \Delta x \\ \theta \\ \theta \end{pmatrix} \times \begin{pmatrix} x'_1 \\ y'_1 \\ 1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \\ \begin{pmatrix} \theta \\ -\Delta x \\ \Delta x y'_1 \end{pmatrix} &= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \end{aligned}$$

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

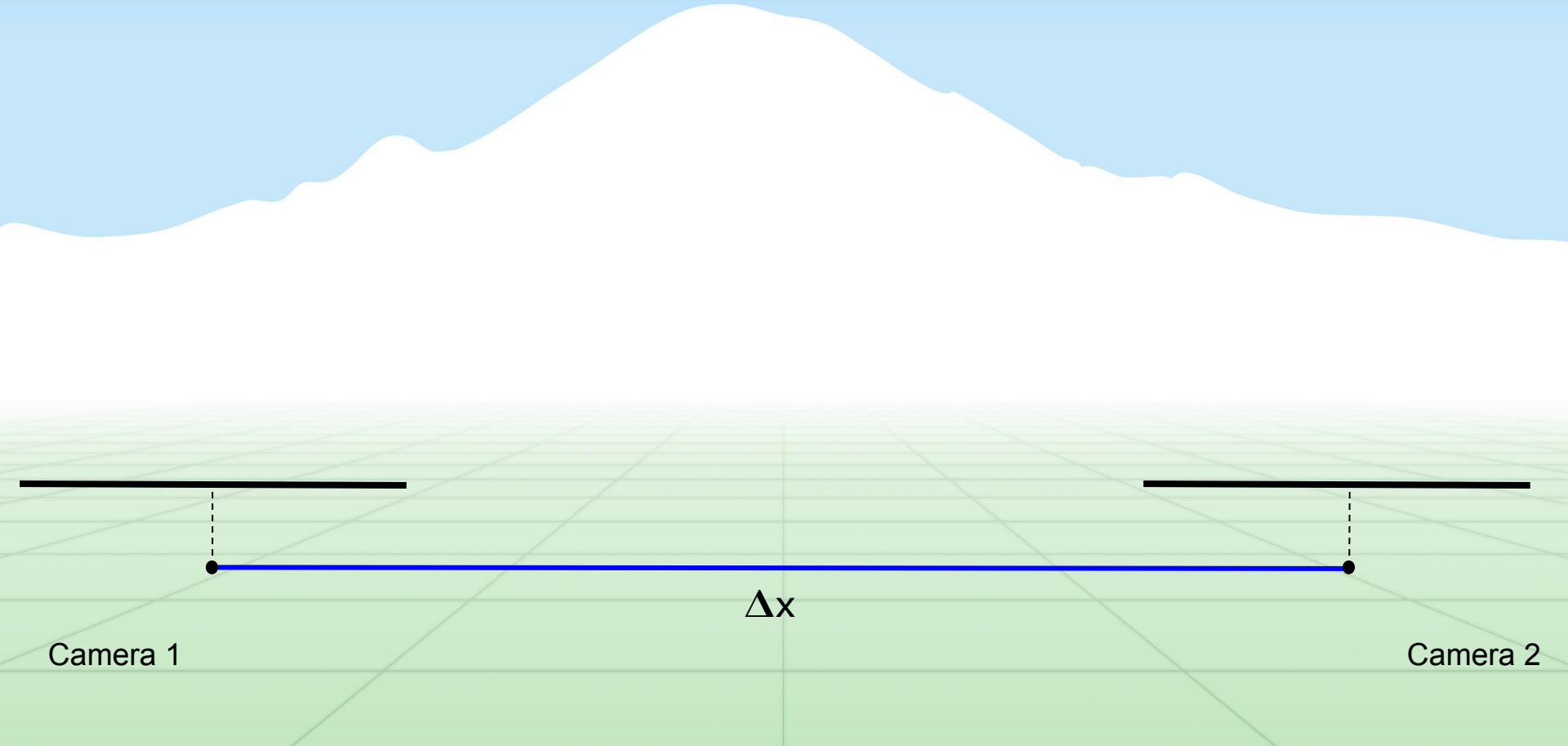
Stereo: Parallel Image Planes



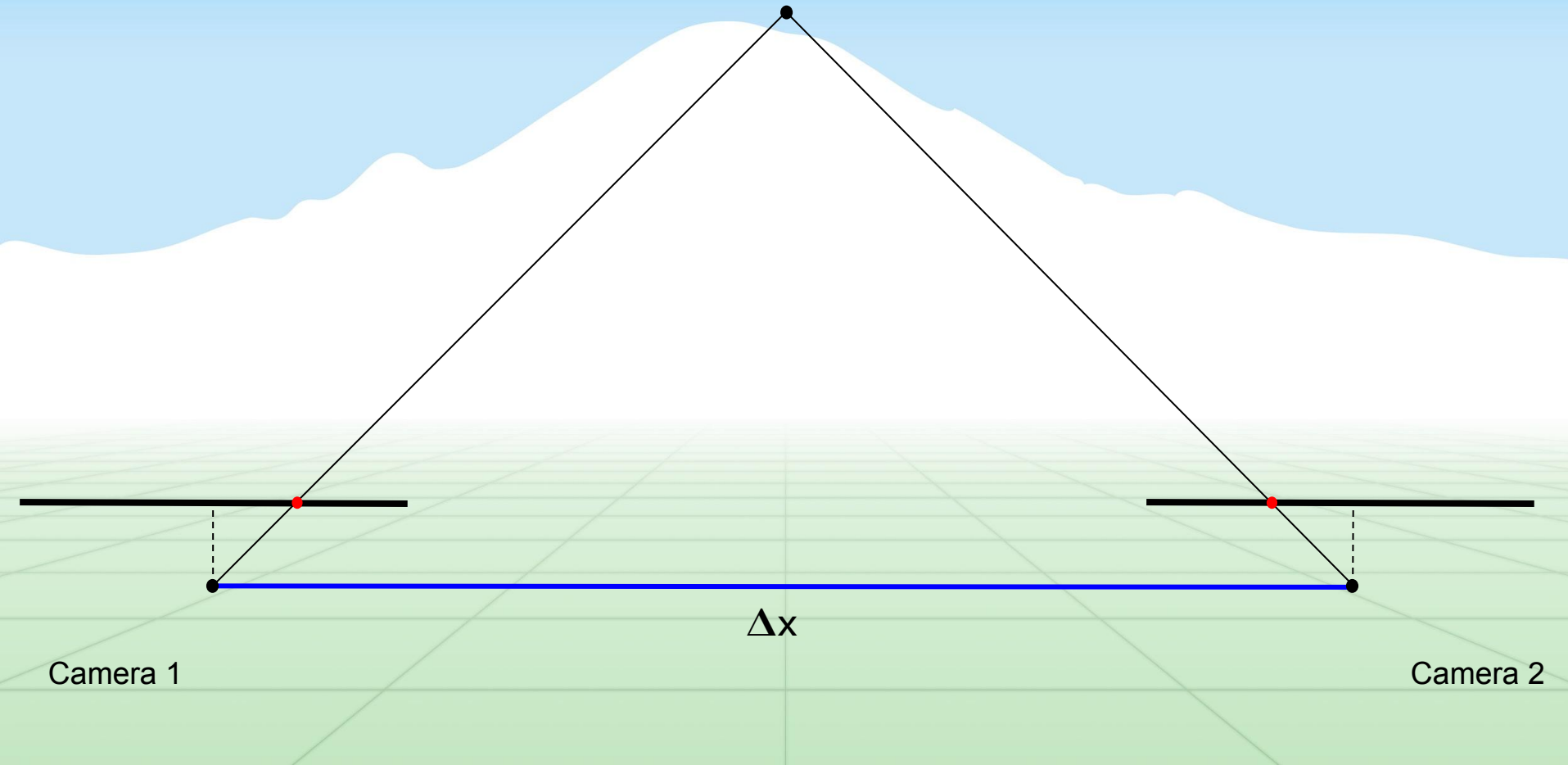
Stereo: Parallel Image Planes



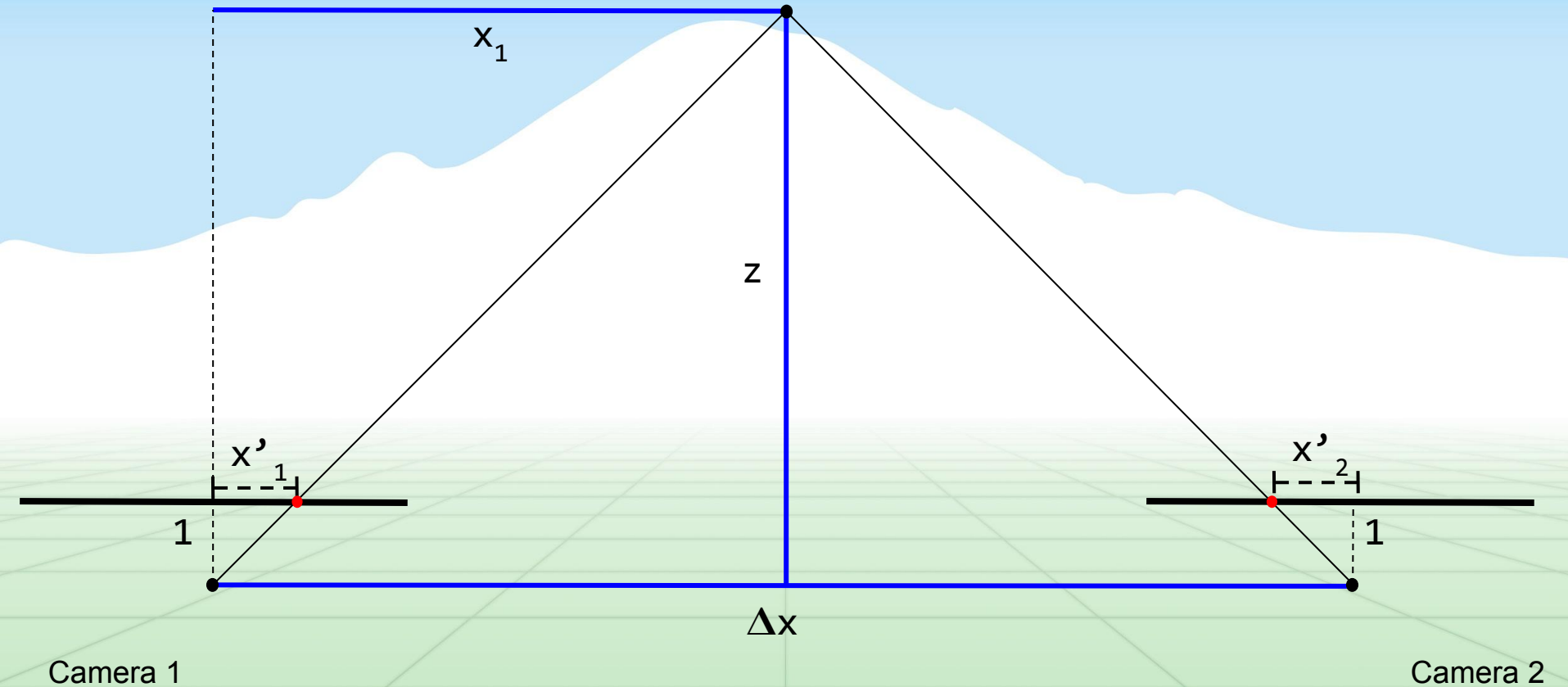
Stereo: Parallel Image Planes



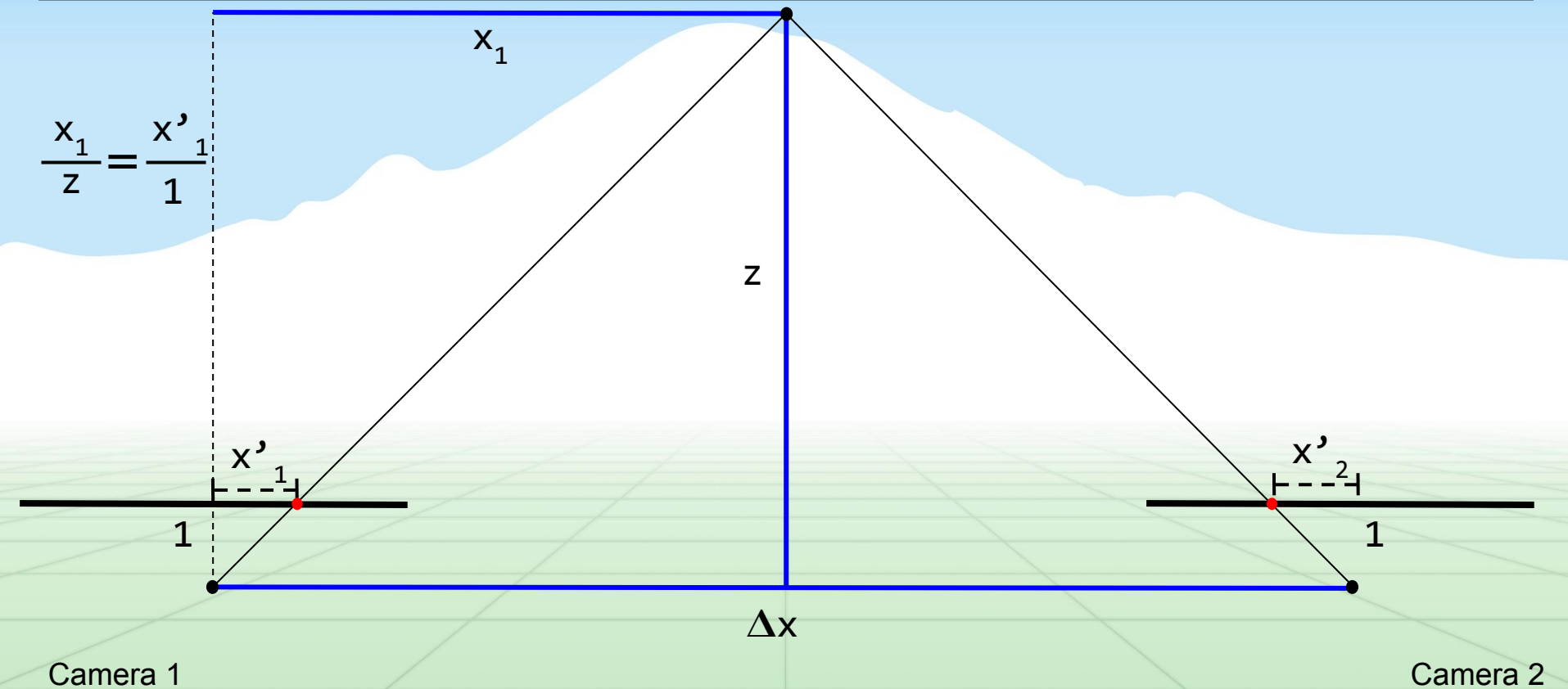
Stereo: Parallel Image Planes



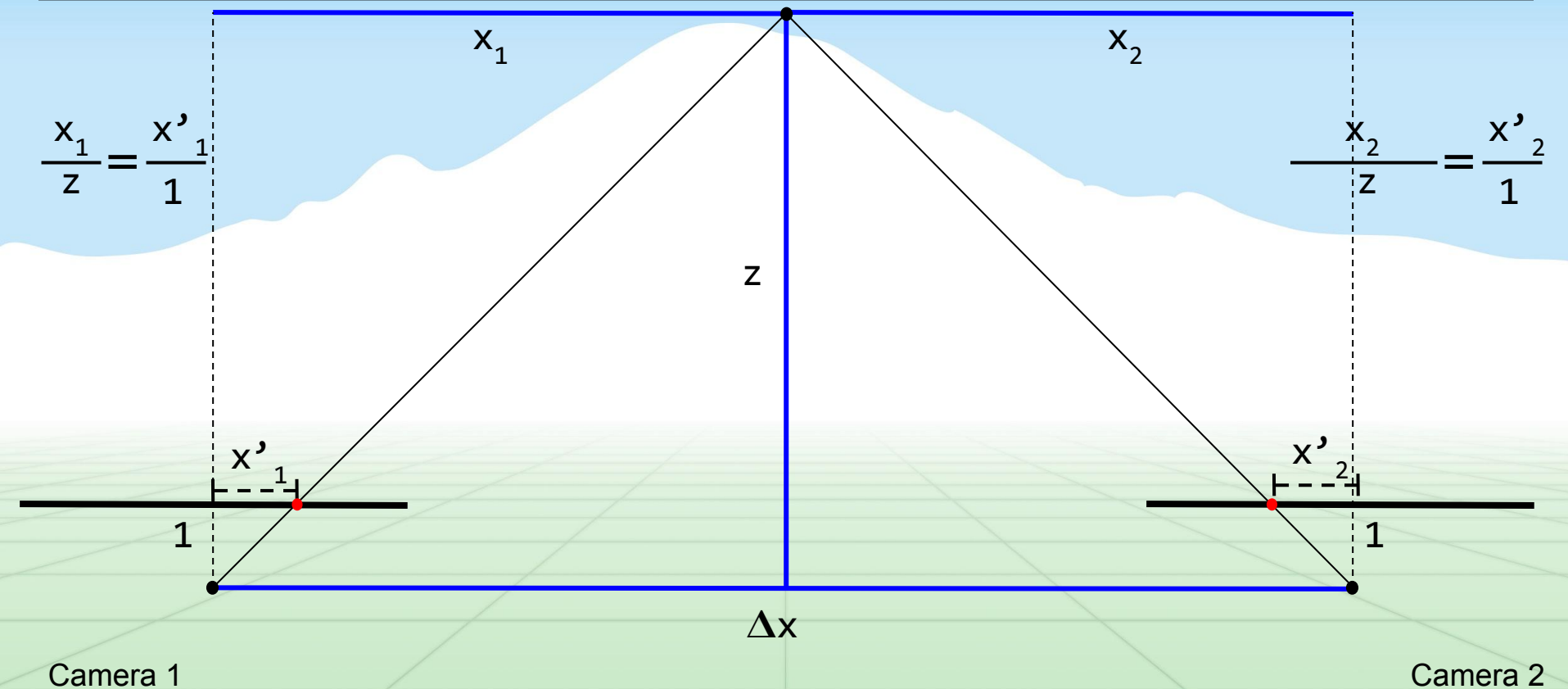
Stereo: Parallel Image Planes



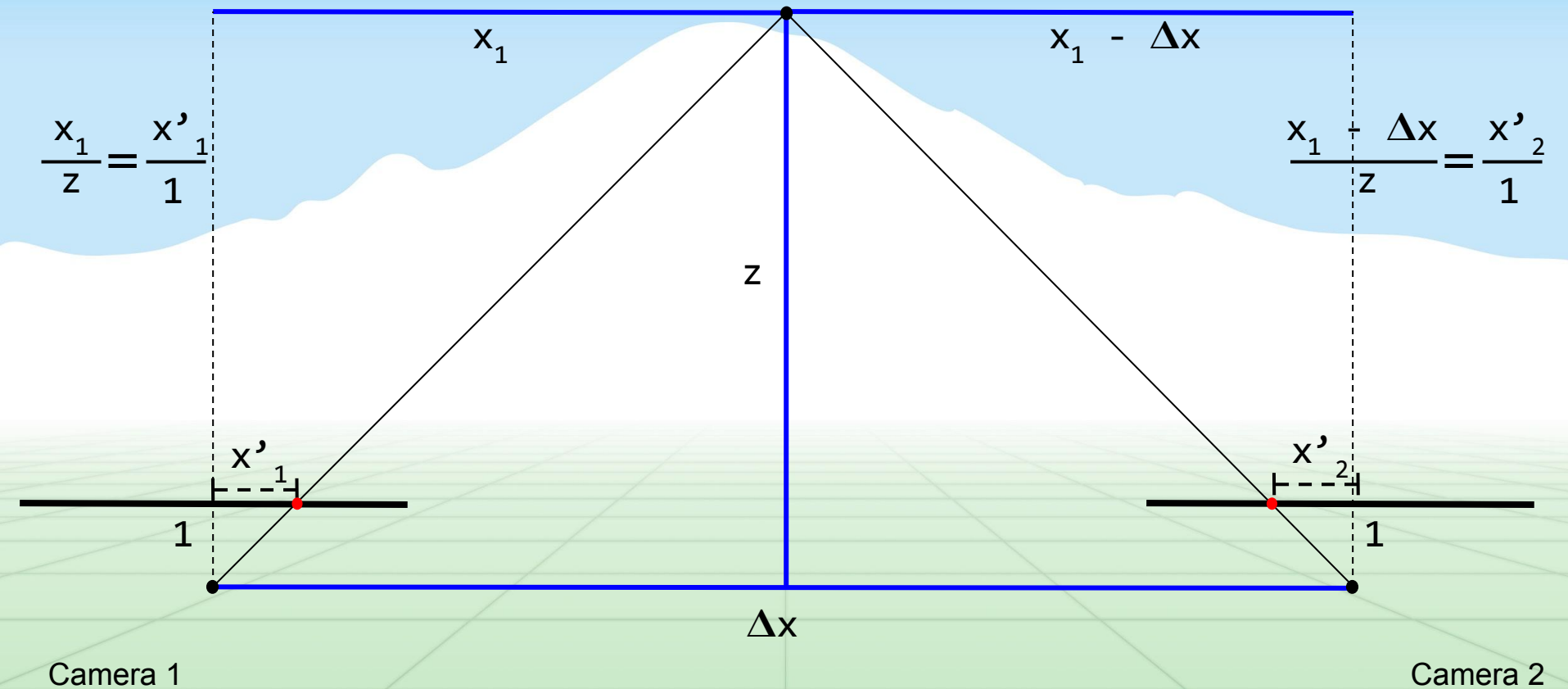
Stereo: Parallel Image Planes



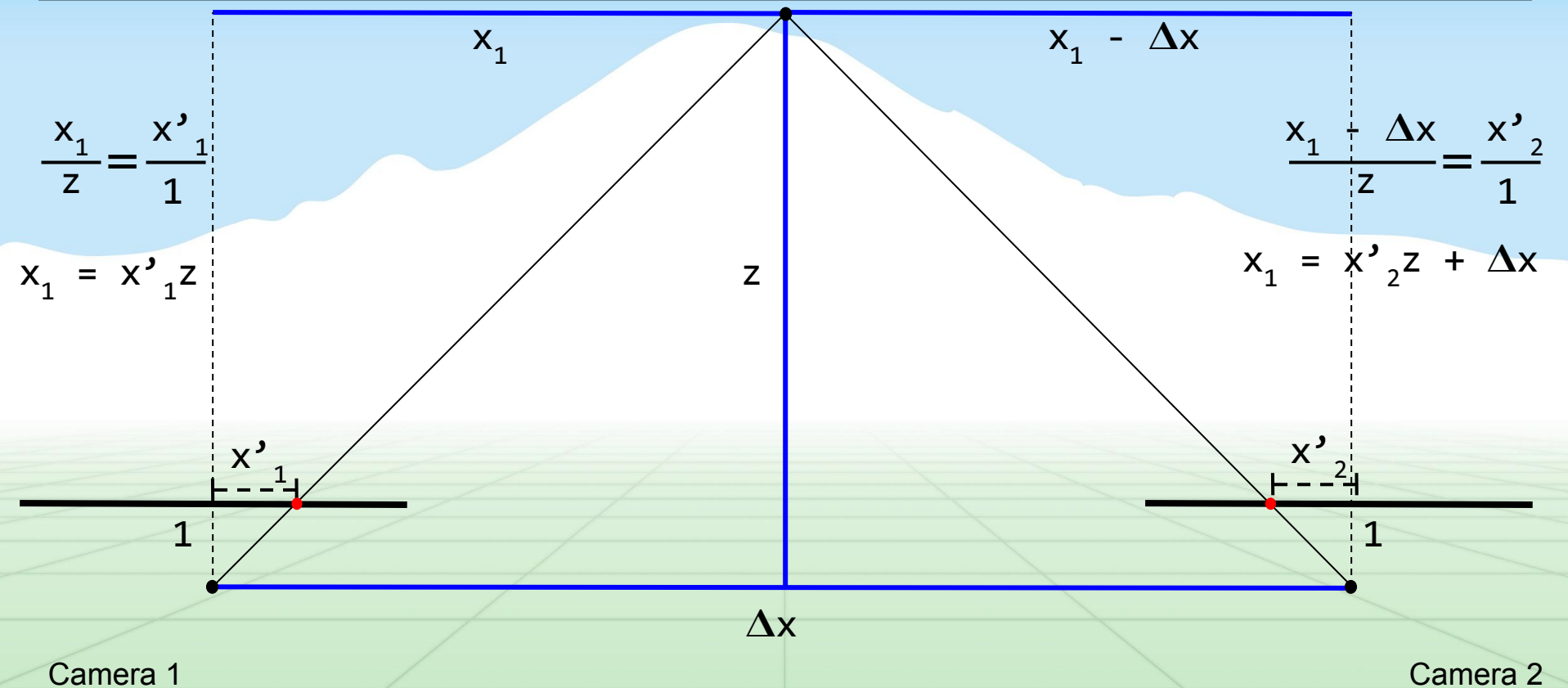
Stereo: Parallel Image Planes



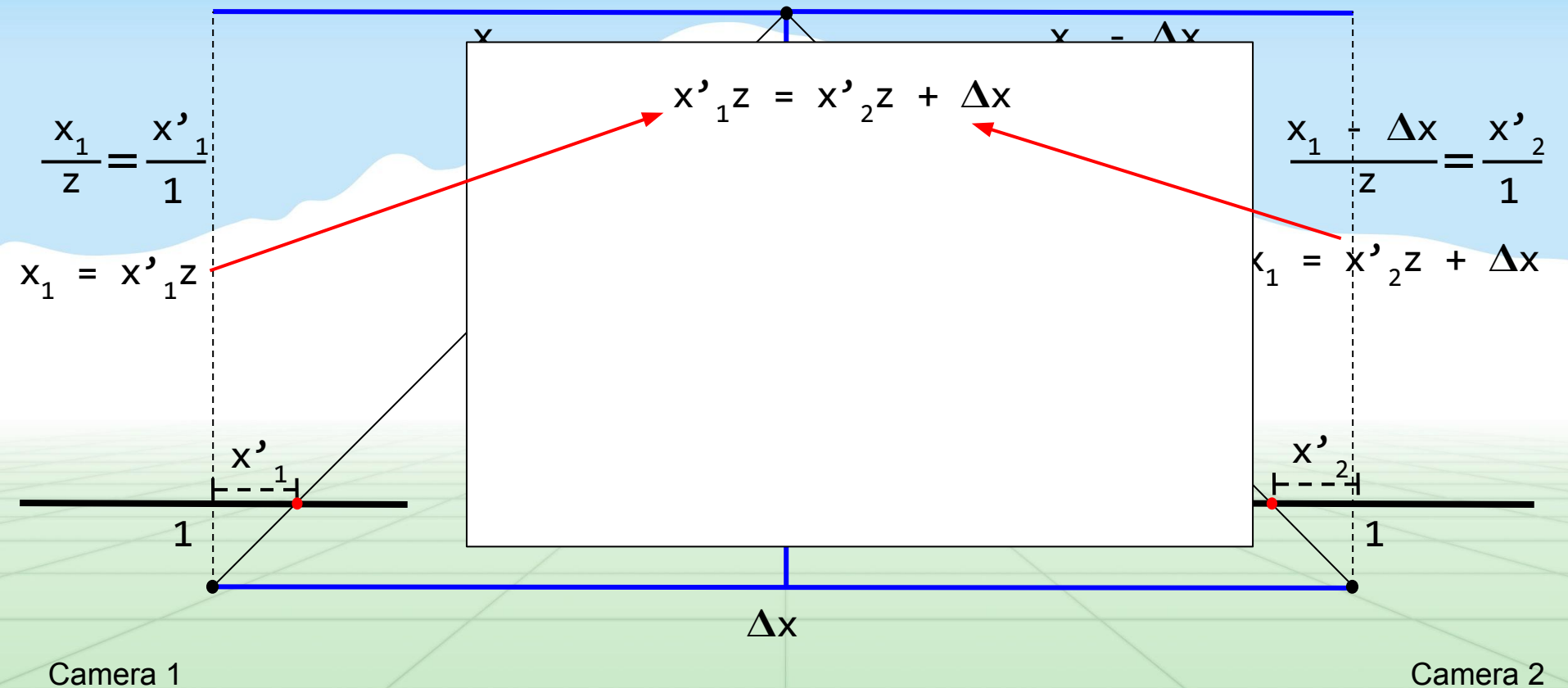
Stereo: Parallel Image Planes



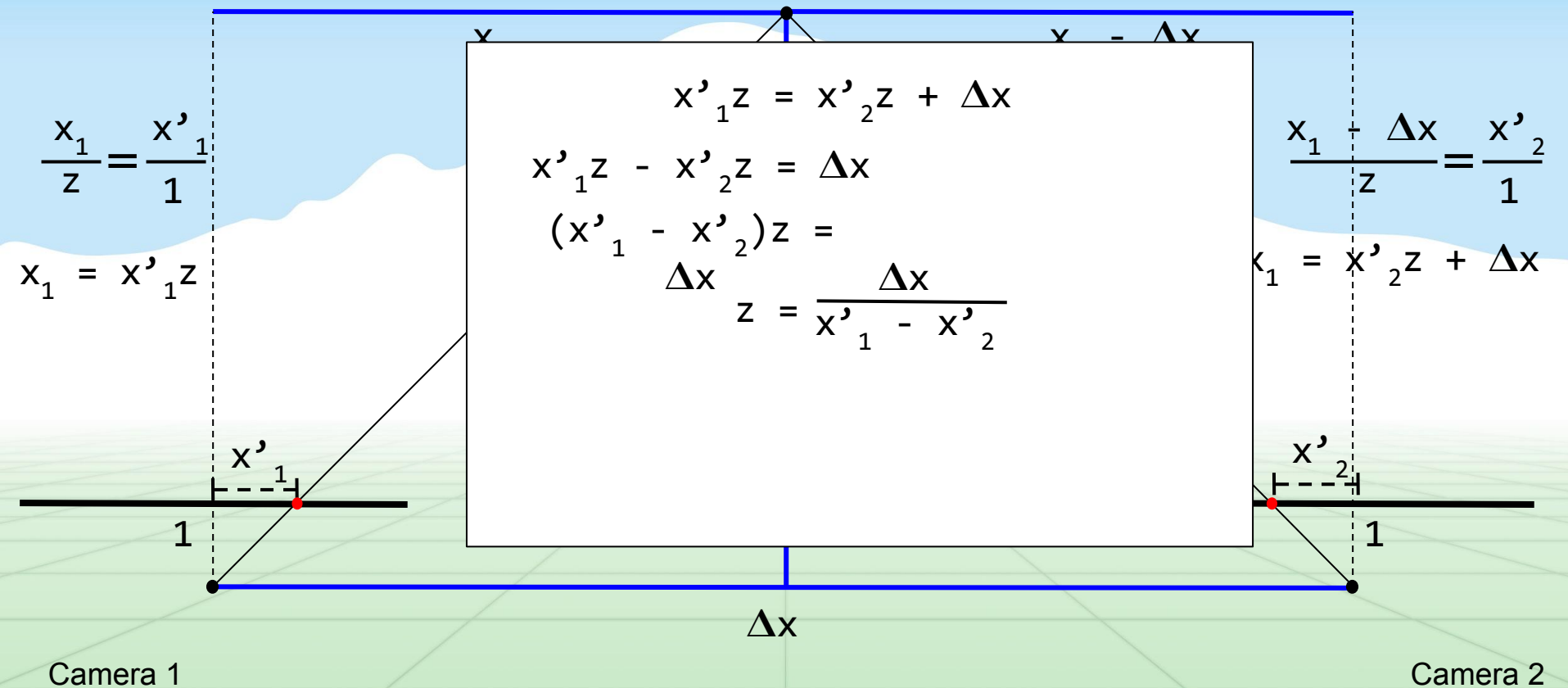
Stereo: Parallel Image Planes



Stereo: Parallel Image Planes

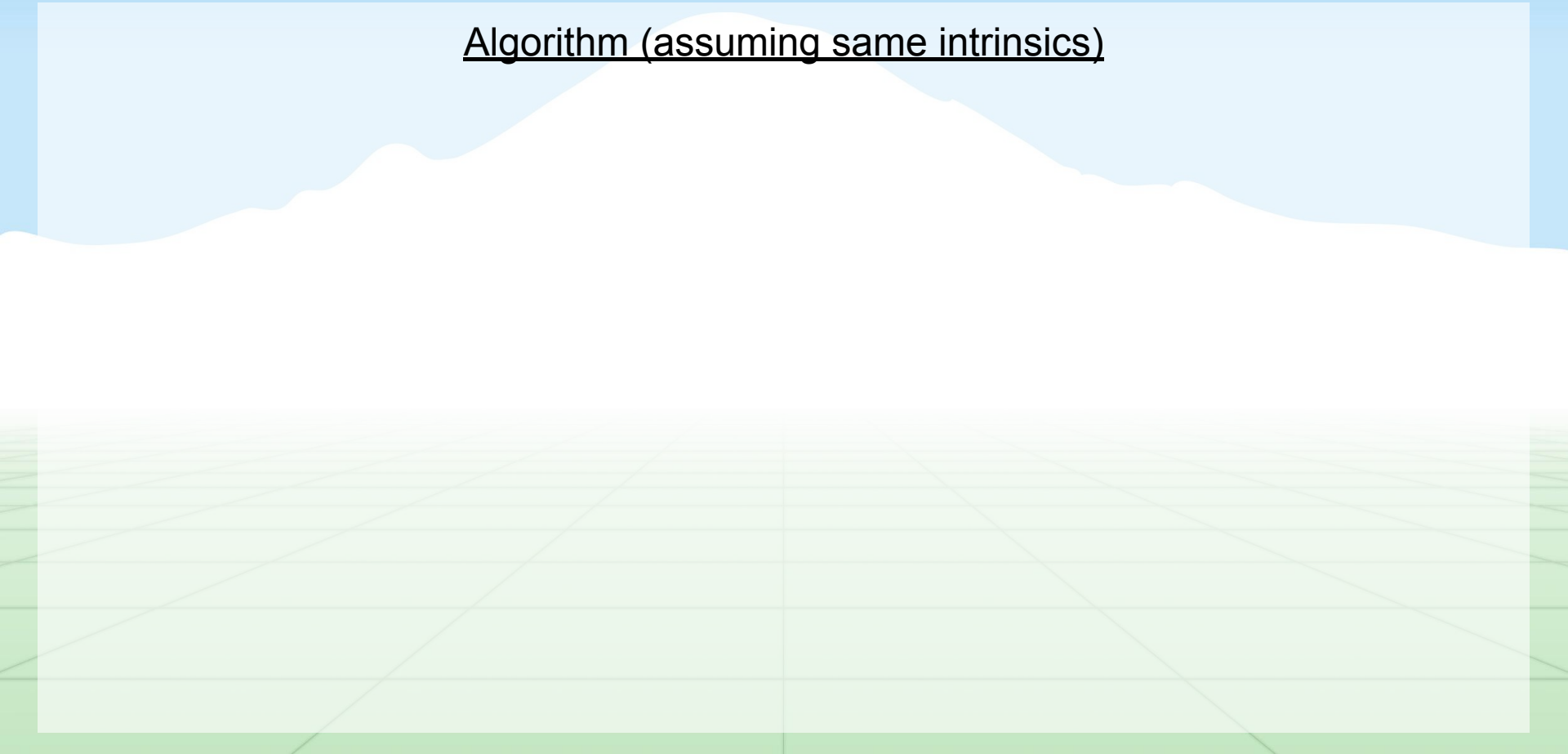


Stereo: Parallel Image Planes



Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)



Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

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

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):  
  depth <- new_image_like( $I_1$ )
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):  
  depth  $\leftarrow$  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( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):  
  depth  $\leftarrow$  new_image_like( $I_1$ )  
  For every pixel  $i, j$  in  $I_1$ :  
     $f_{ij}$  = get_feature_descriptor( $I_1$ ,  $i$ ,  $j$ )  
    best_score  $\leftarrow$  INFINITY  
    best_i  $\leftarrow$  NaN  
    For  $i'$  in columns( $I_2$ ):
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):  
  depth  $\leftarrow$  new_image_like( $I_1$ )  
  For every pixel  $i, j$  in  $I_1$ :  
     $f_{ij}$  = get_feature_descriptor( $I_1$ ,  $i$ ,  $j$ )  
    best_score  $\leftarrow$  INFINITY  
    best_i  $\leftarrow$  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( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):  
  depth  $\leftarrow$  new_image_like( $I_1$ )  
  For every pixel  $i, j$  in  $I_1$ :  
     $f_{ij}$  = get_feature_descriptor( $I_1$ ,  $i$ ,  $j$ )  
    best_score  $\leftarrow$  INFINITY  
    best_i  $\leftarrow$  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[ $i$ ,  $j$ ] =  $fp\Delta x / (i - \text{best\_i})$   
  Return depth
```

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):
```

```
  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[ $i$ ,  $j$ ] =  $fp\Delta x / (i - best\_i)$ 
```

```
  Return depth
```

Parallel image planes
make things easy!

Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

```
compute_depth( $I_1$ ,  $I_2$ ,  $\Delta x$ ,  $f$ ,  $p$ ):
```

```
  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[ $i$ ,  $j$ ] =  $f p \Delta x / (i - \text{best\_i})$ 
```

```
  Return depth
```

What are these?



Stereo: Parallel Image Planes

Algorithm (assuming same intrinsics)

compute
depth
For



I_1, i, j

What are these?

operator(I_2, i', j)

match($f_{ij}, f_{i',j}$)

```
If score < best_score:
```

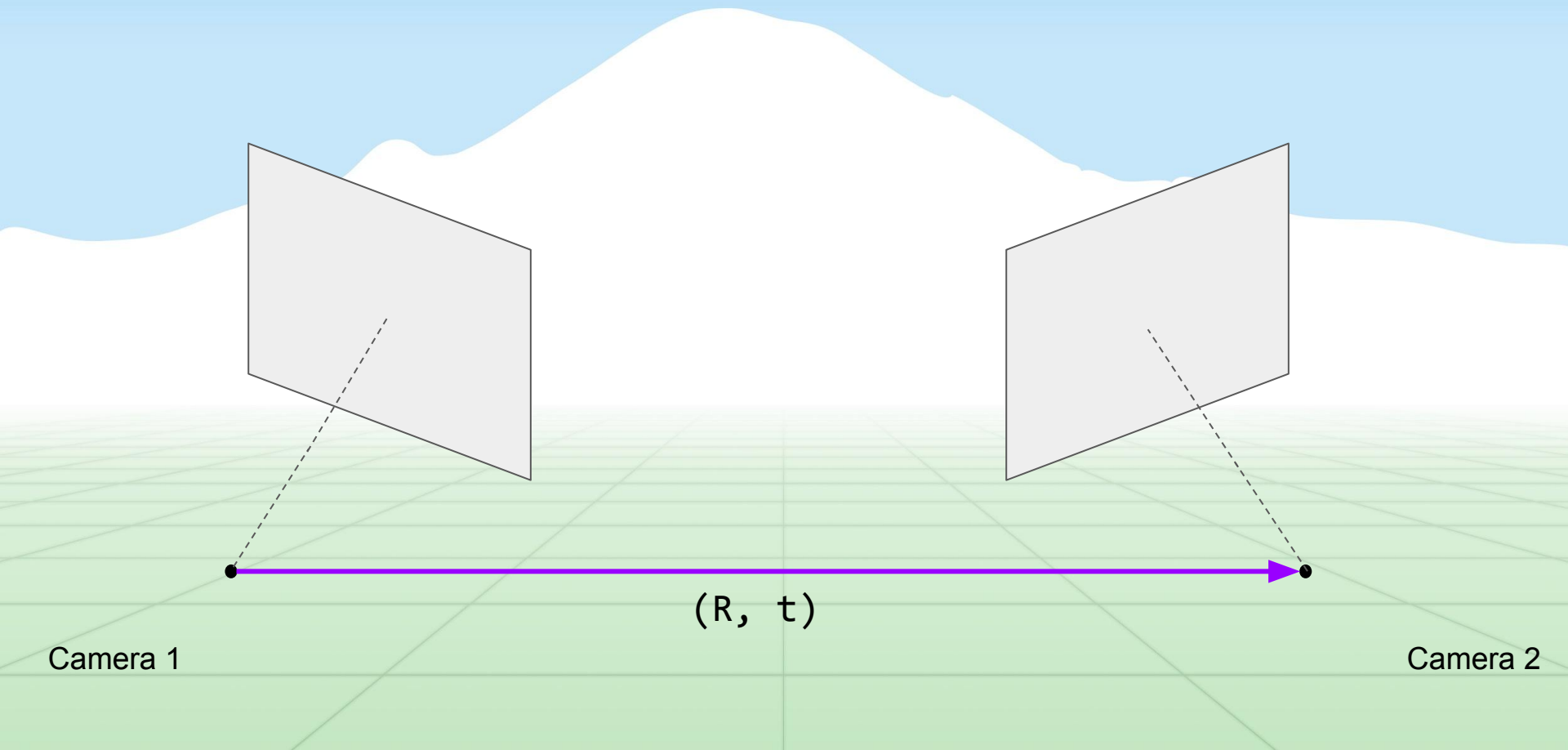
```
    best_score = score
```

```
    best_i = i'
```

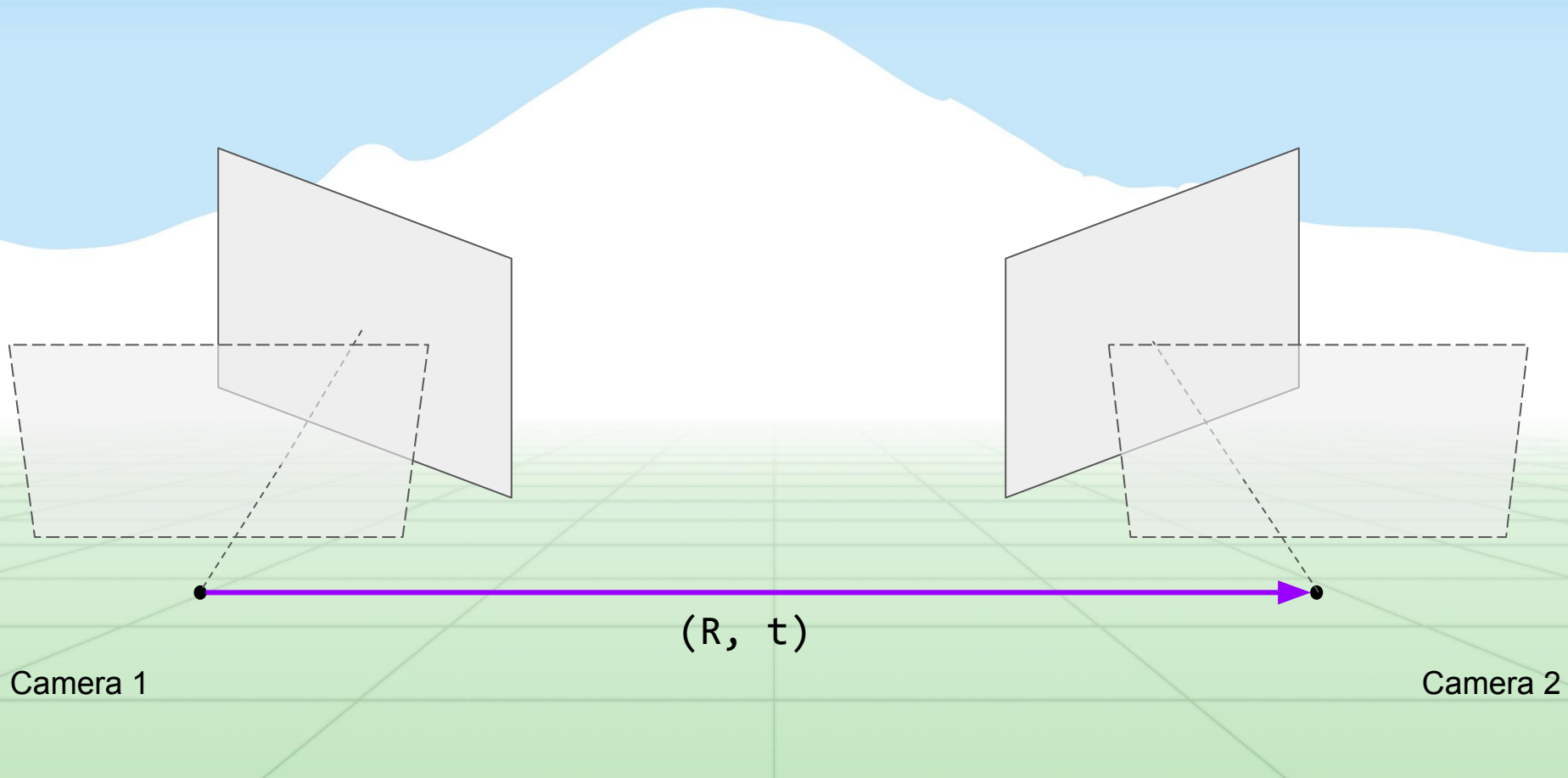
```
    depth[i, j] = fp $\Delta$ x/(i - best_i)
```

```
Return depth
```

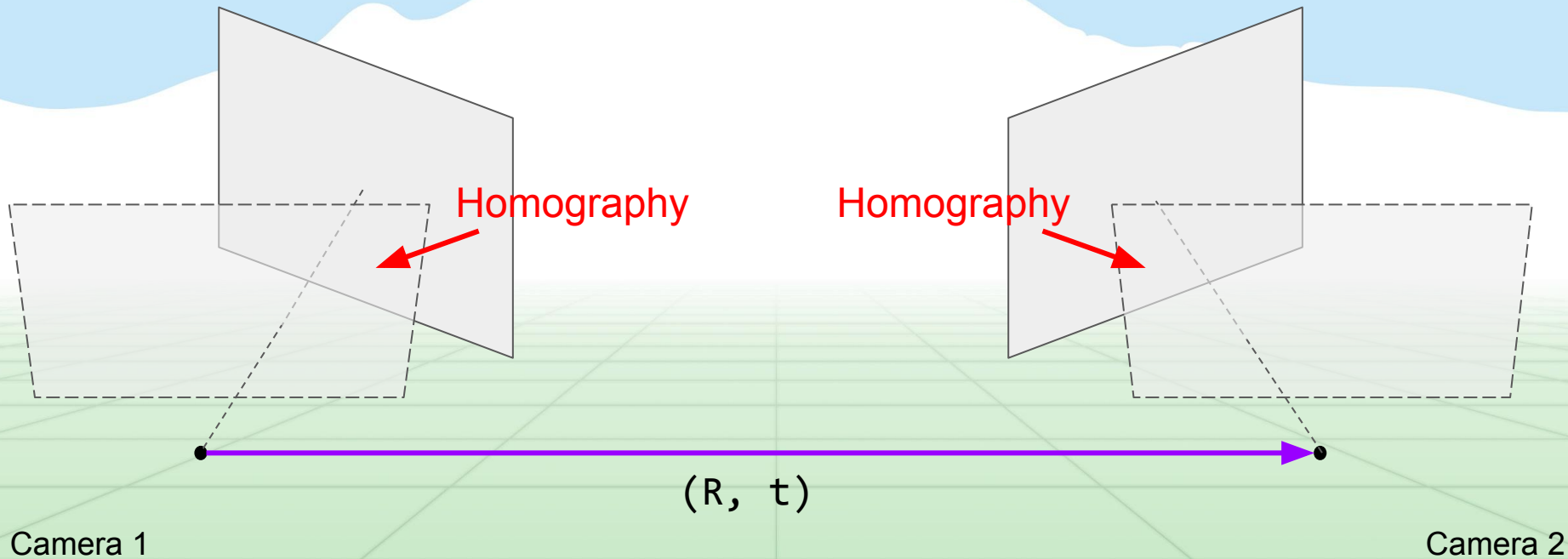
Stereo: Computing Homographies



Stereo: Computing Homographies



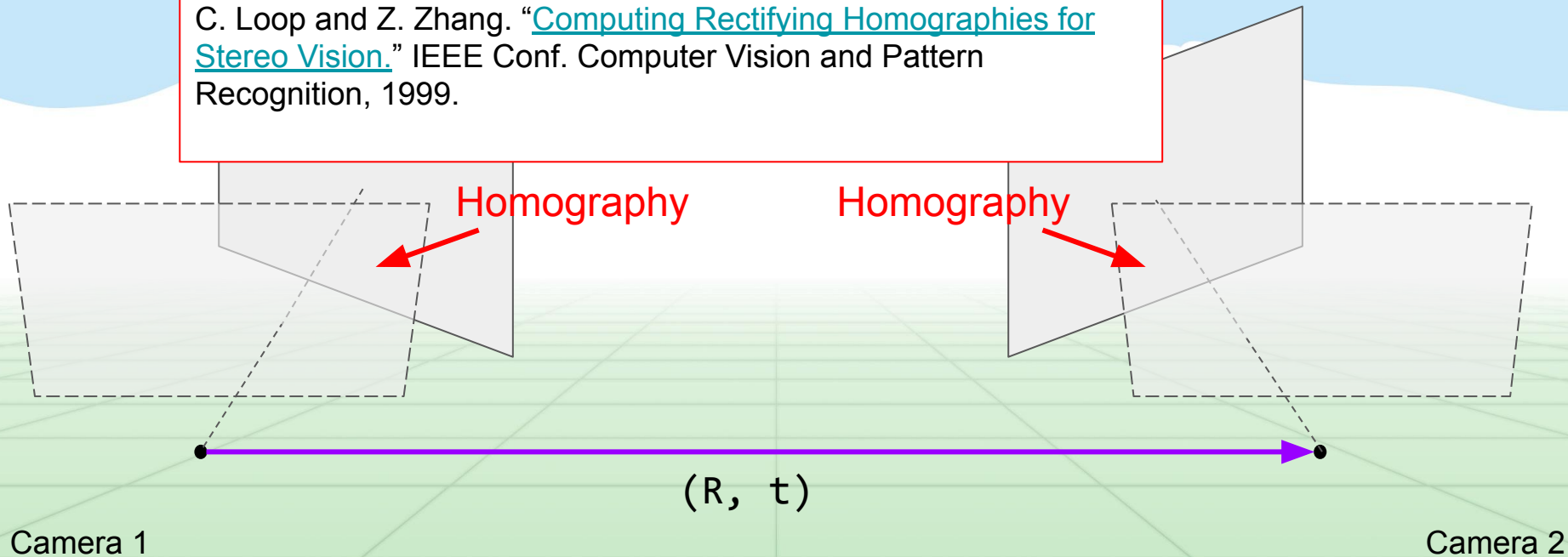
Stereo: Computing Homographies



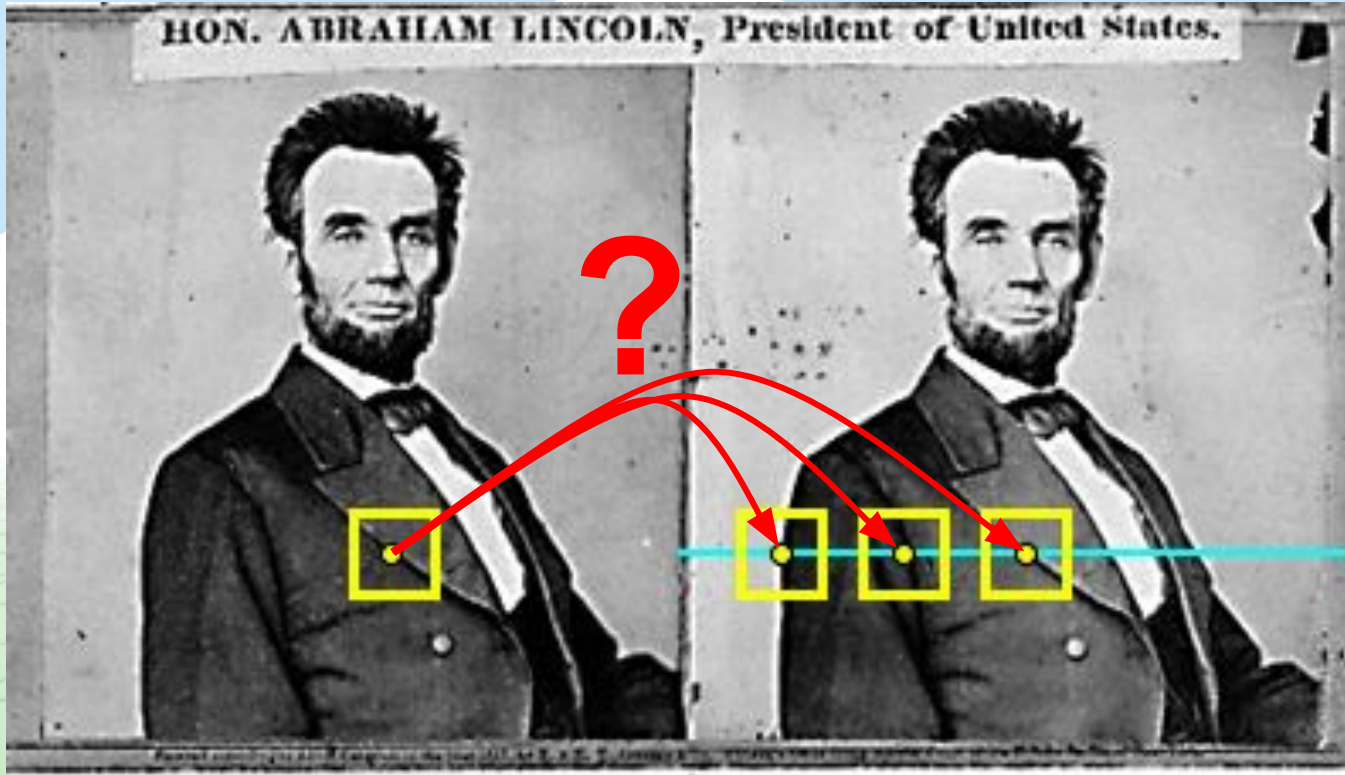
Stereo: Computing Homographies

Stereo image rectification

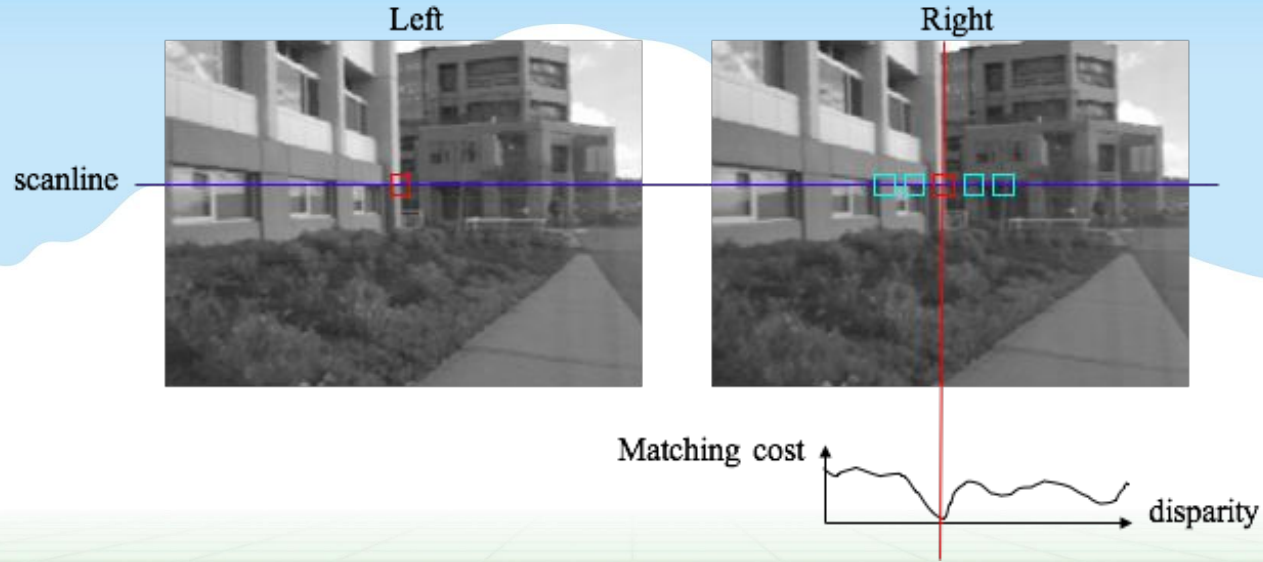
C. Loop and Z. Zhang. “[Computing Rectifying Homographies for Stereo Vision](#).” IEEE Conf. Computer Vision and Pattern Recognition, 1999.



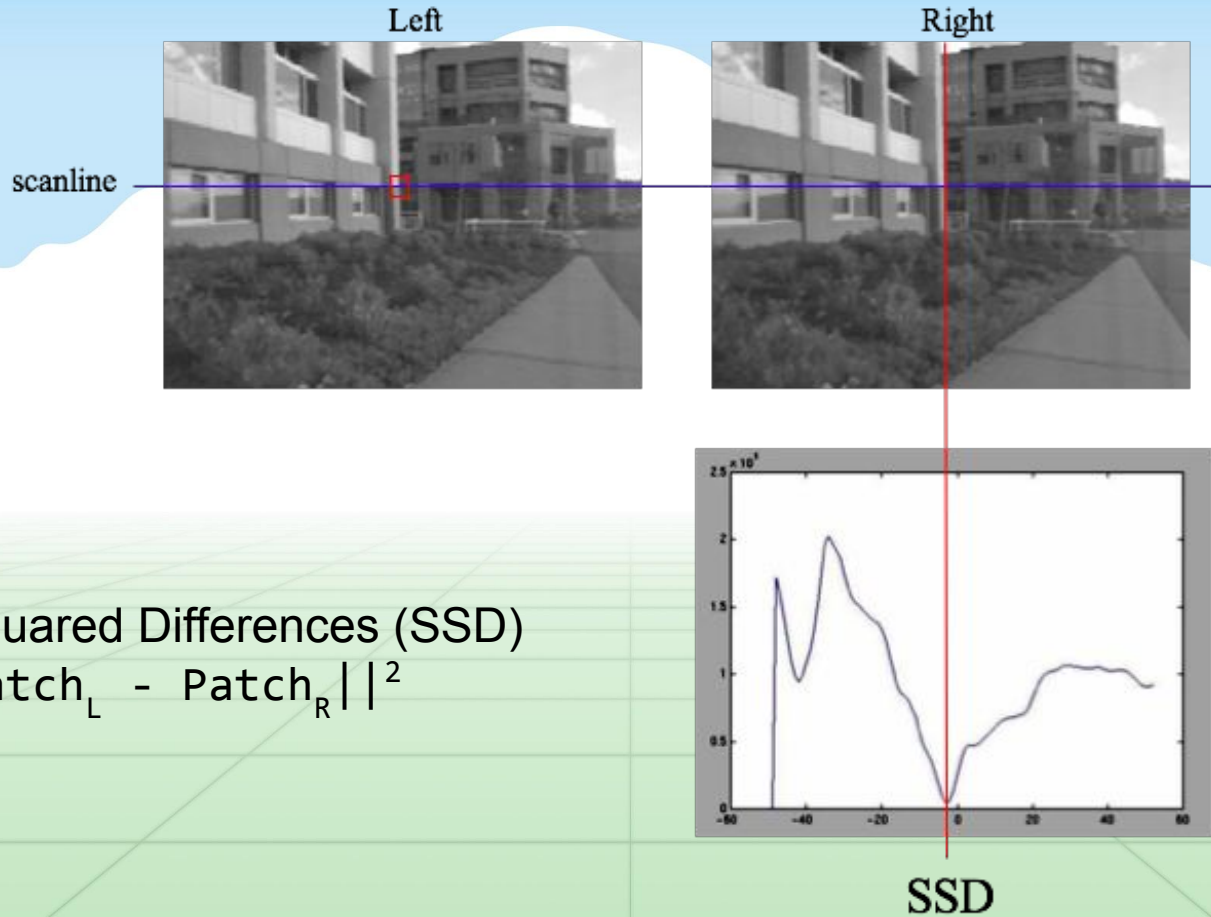
Feature Matching



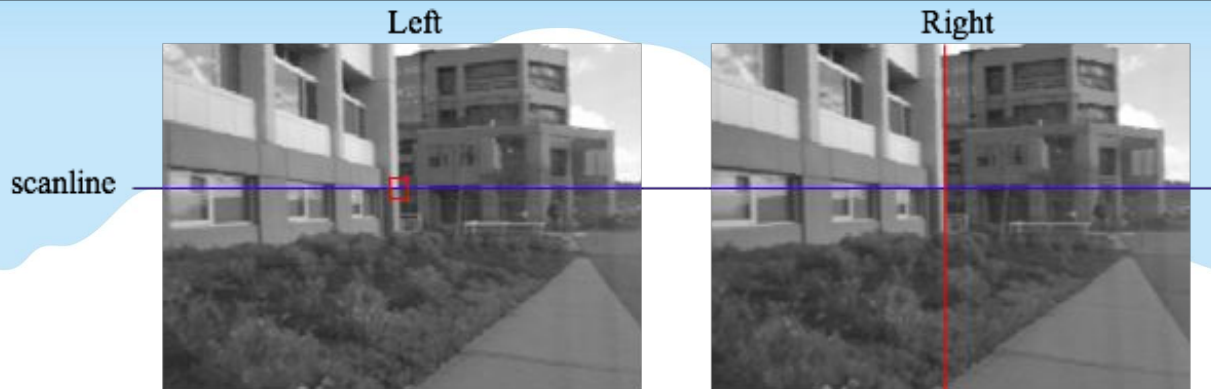
Feature Matching



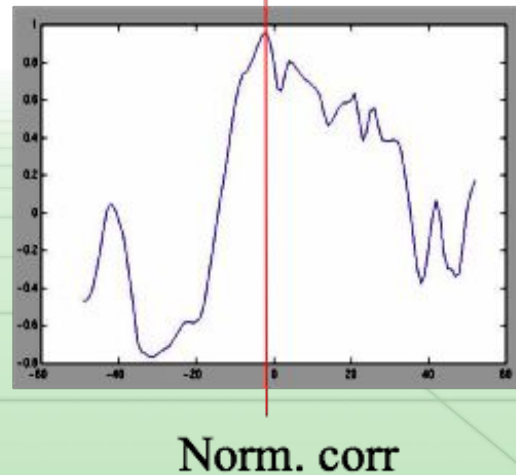
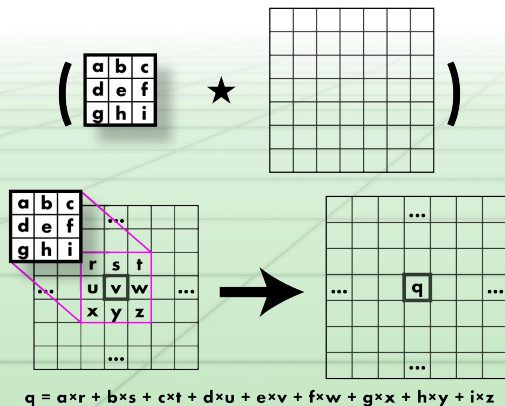
Feature Matching



Feature Matching



Cross-Correlation

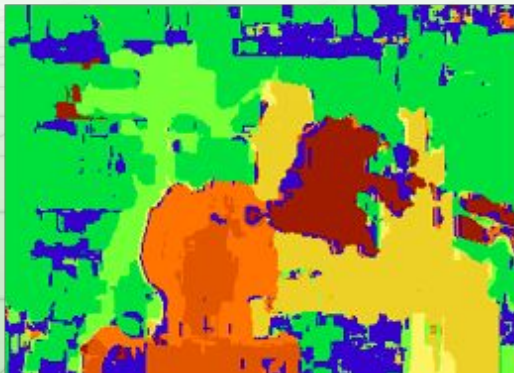


Results with window search

Data



Window-based matching



Ground truth



Results: Mercedes Magic Body Control



Effect of Window Size



Left

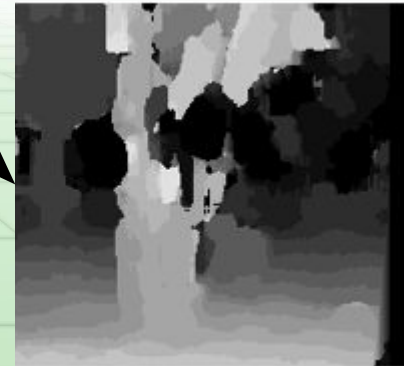


Right

$w = 3$



$w = 20$



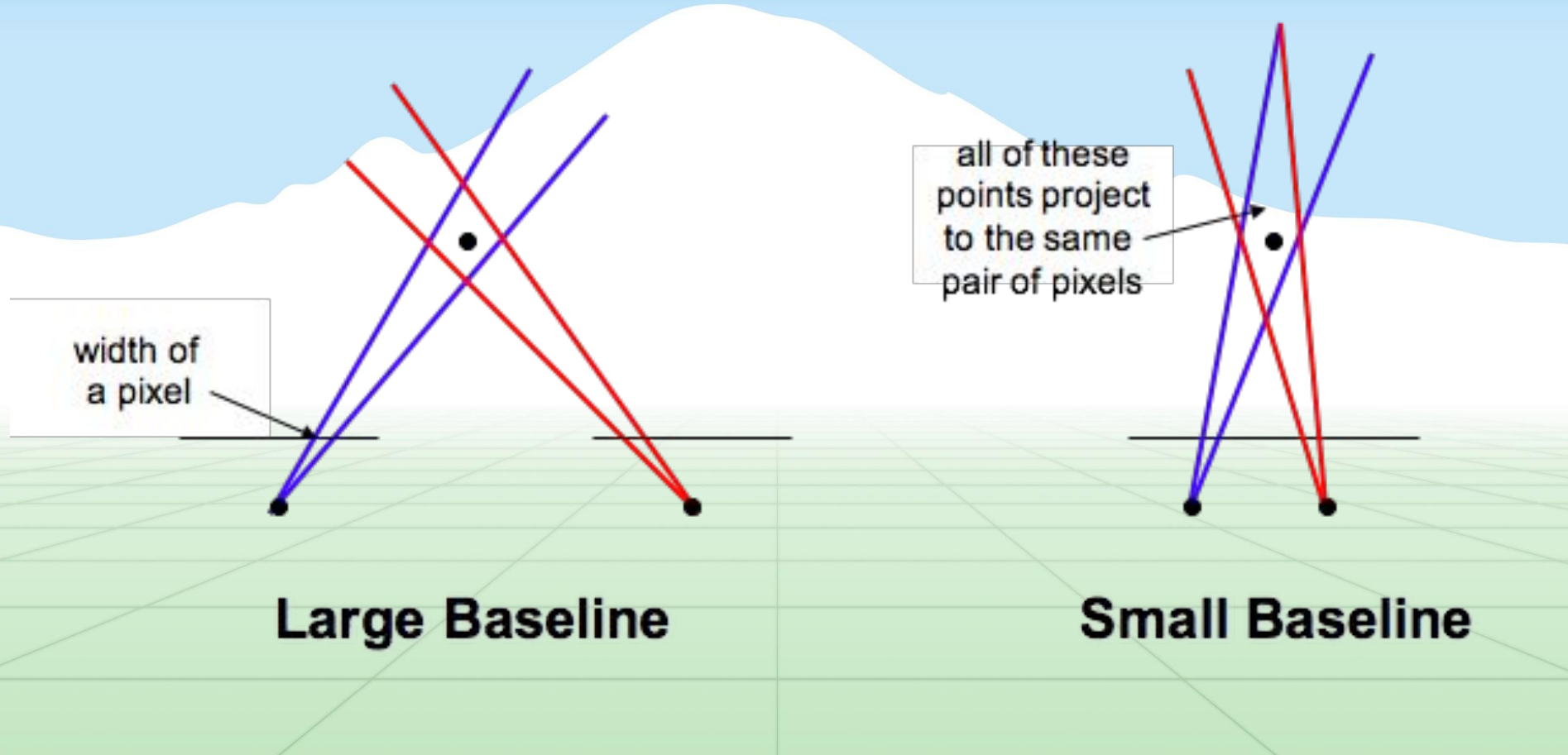
Small window:

- + Good precision, more detail
- Sensitive to noise

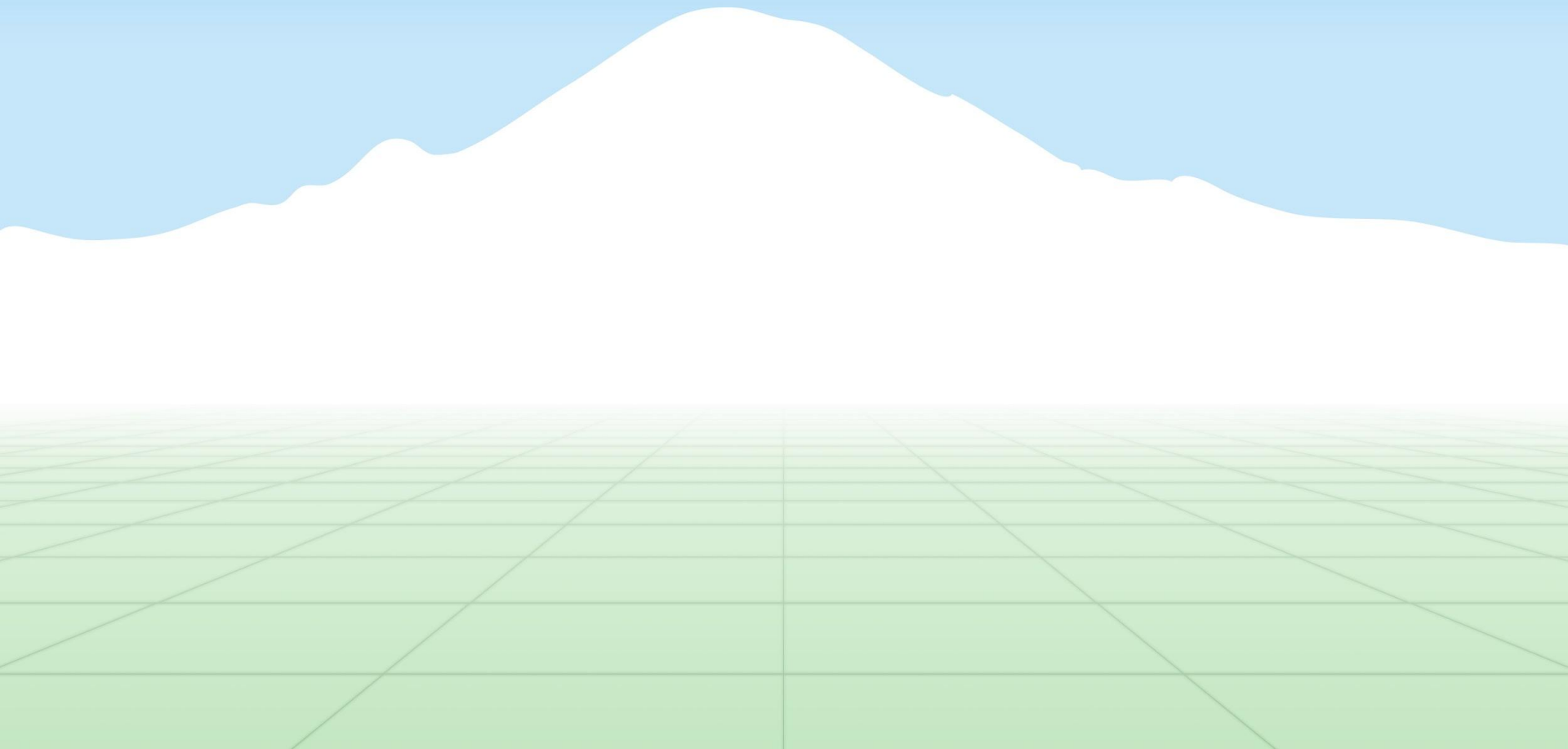
Large window:

- + Robust to noise
- Reduced precision, less detail

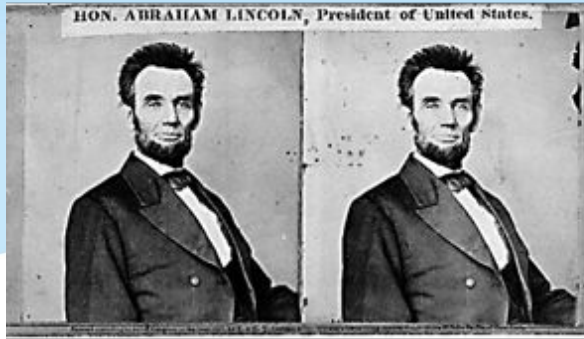
Effect of baseline size



Failure of Correspondence Search



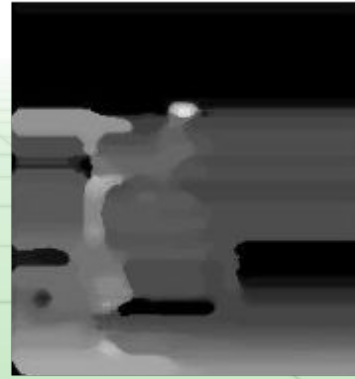
Failure of Correspondence Search



Textureless surfaces



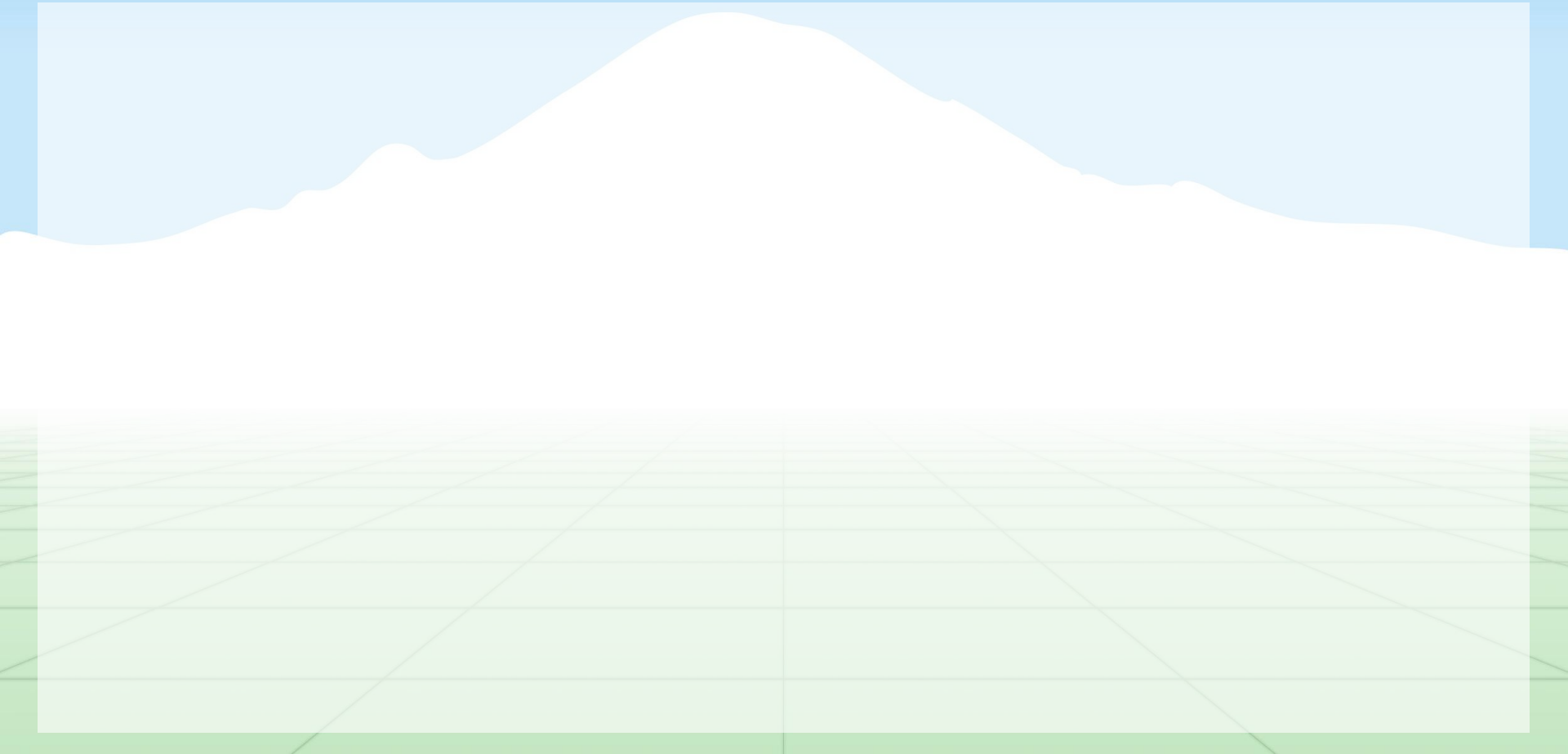
Occlusions, repetition



Non-Lambertian surfaces, specularities

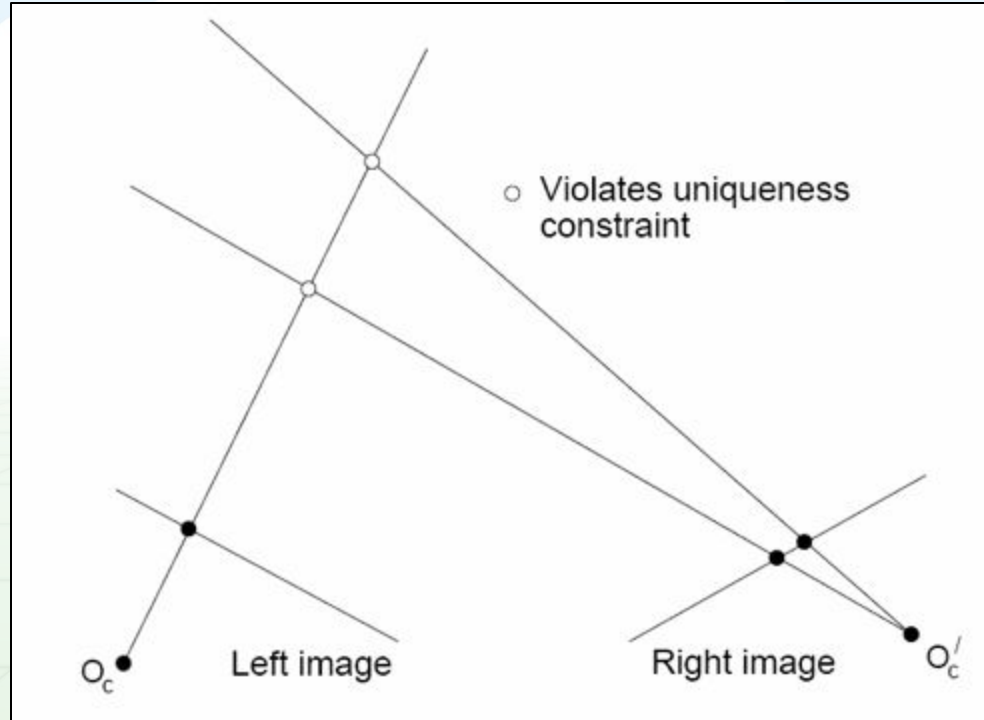
How can we improve the matching?

Stereo Constraints



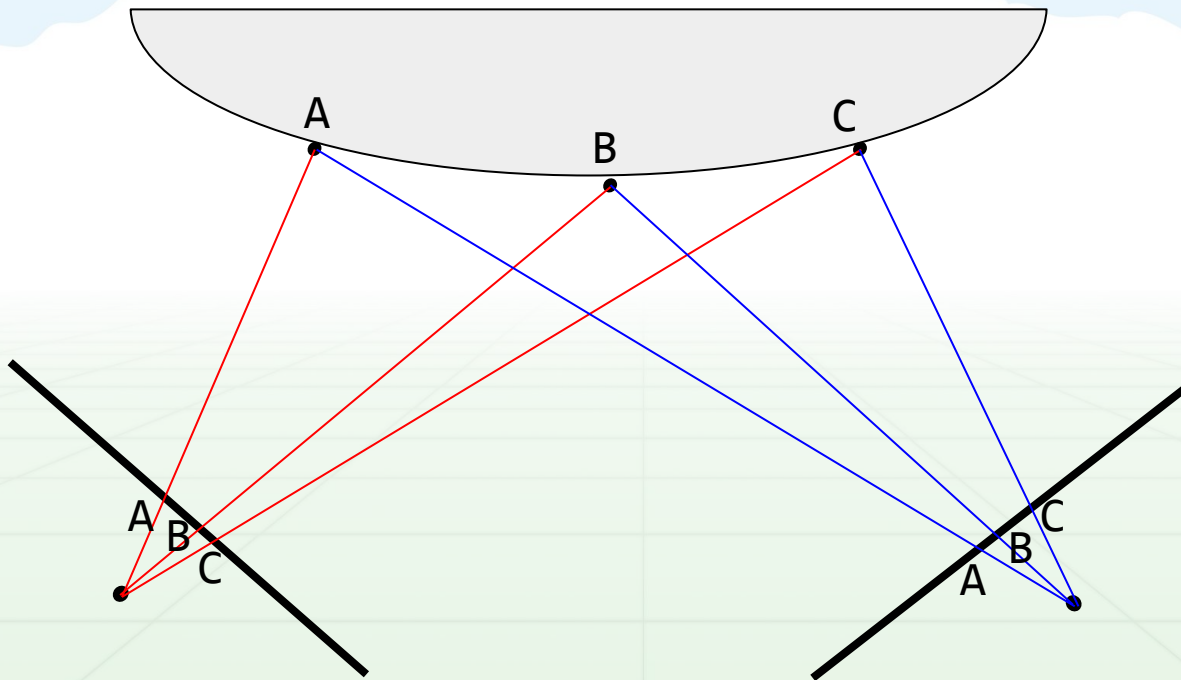
Stereo Constraints

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



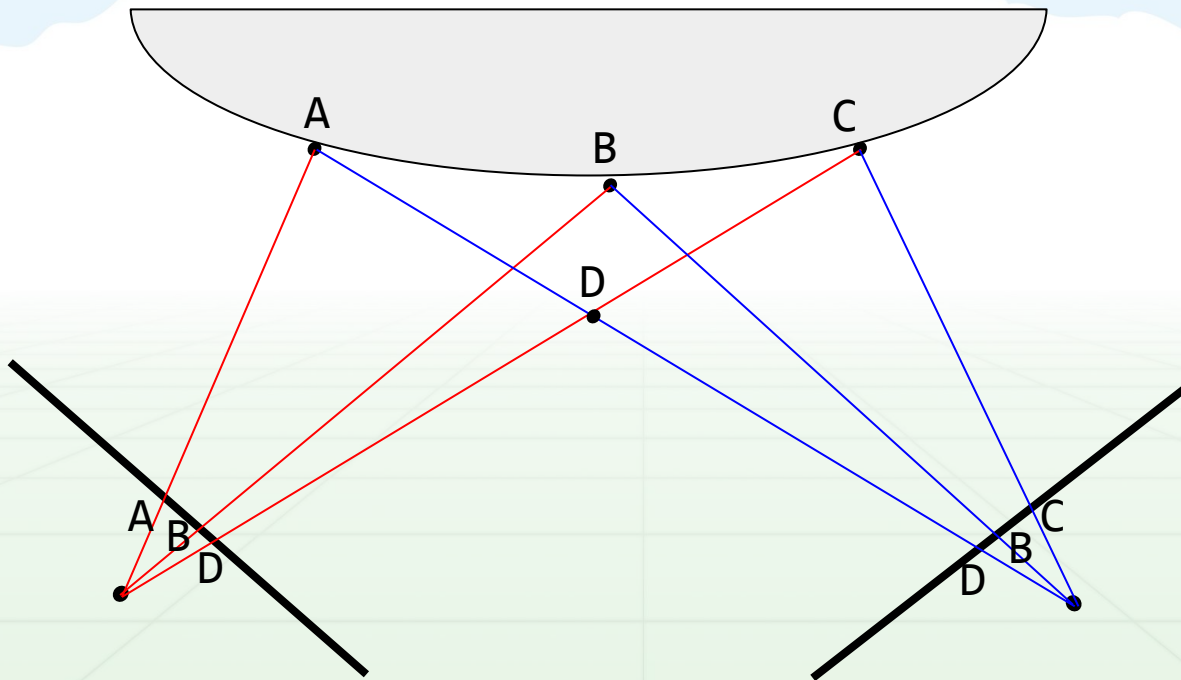
Stereo Constraints

- 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



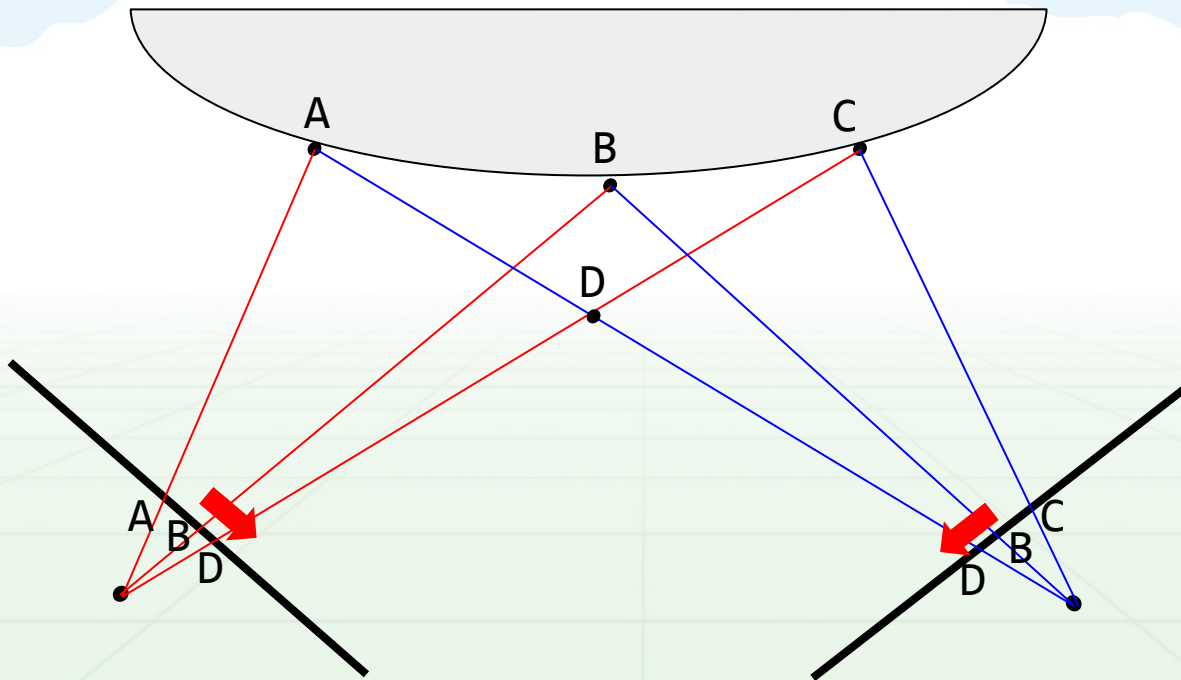
Stereo Constraints

- 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



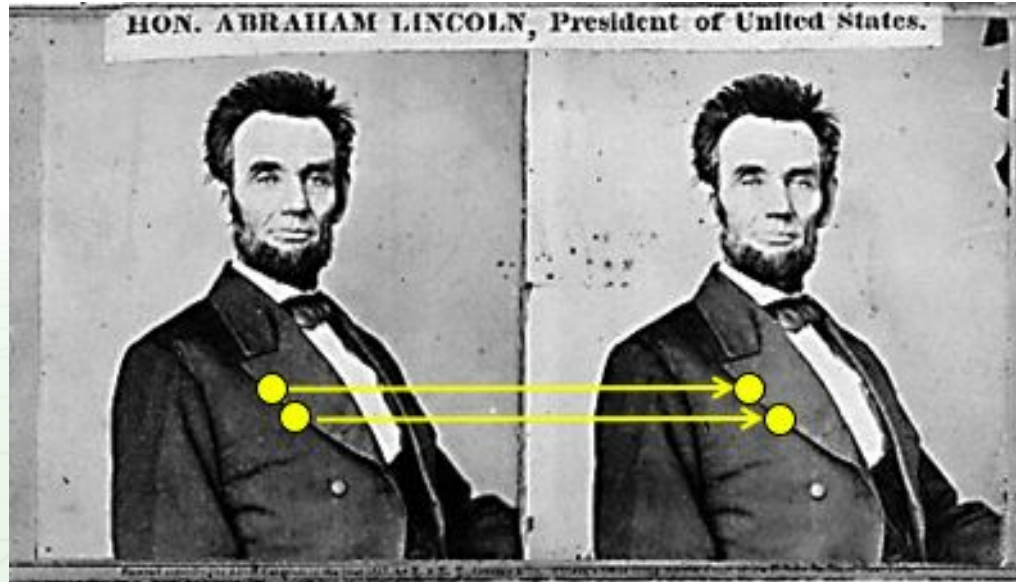
Stereo Constraints

- 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

- 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

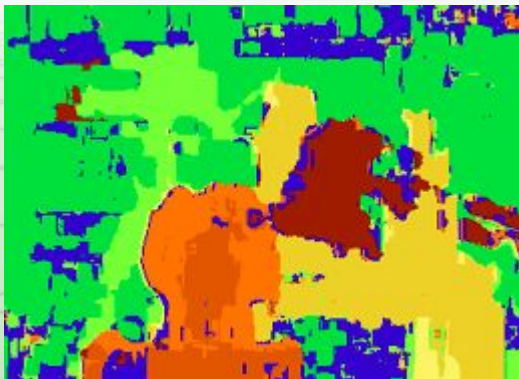
Y. Boykov, O. Veksler, and R. Zabih, [Fast Approximate Energy Minimization via Graph Cuts](#), PAMI 2001

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

