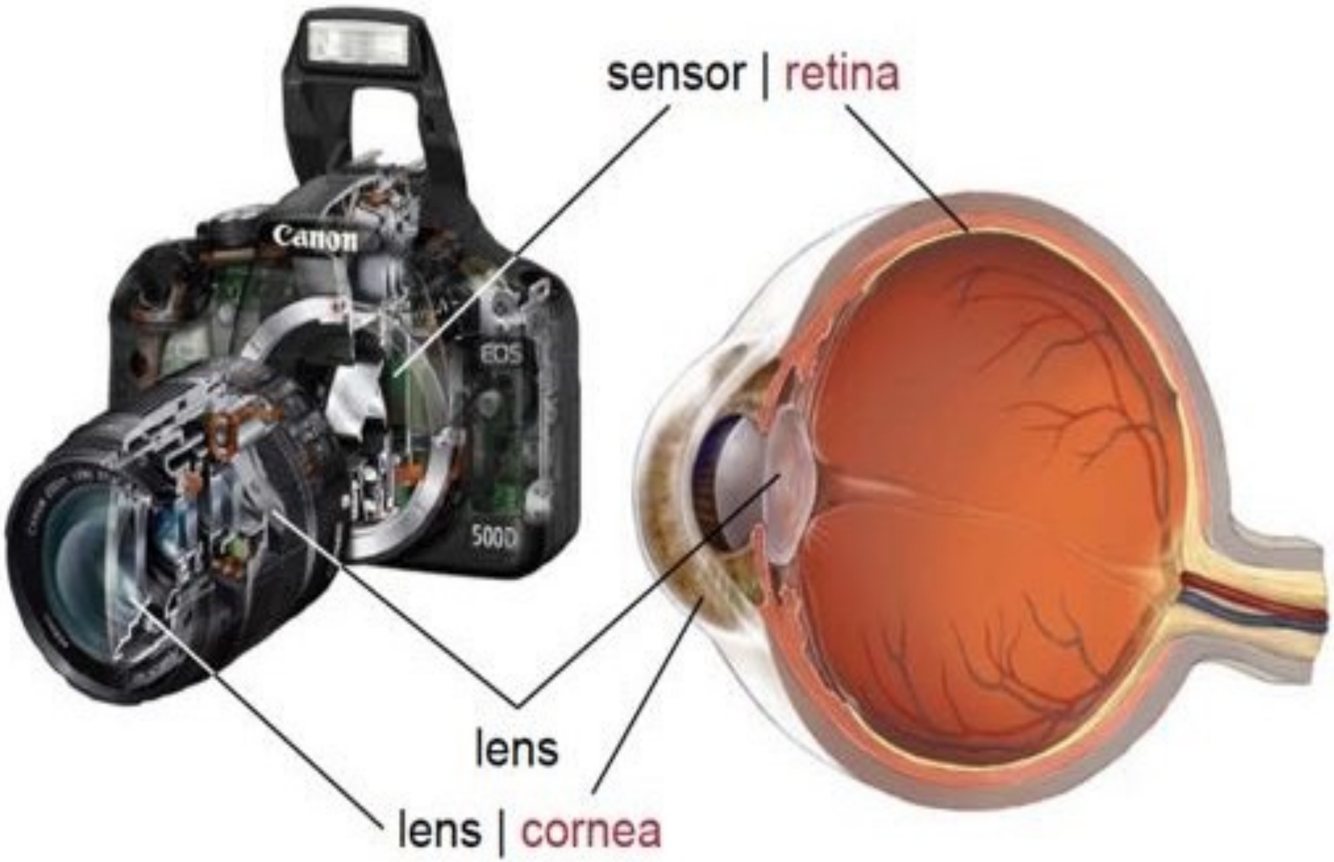
Camera Models and Depth Estimation

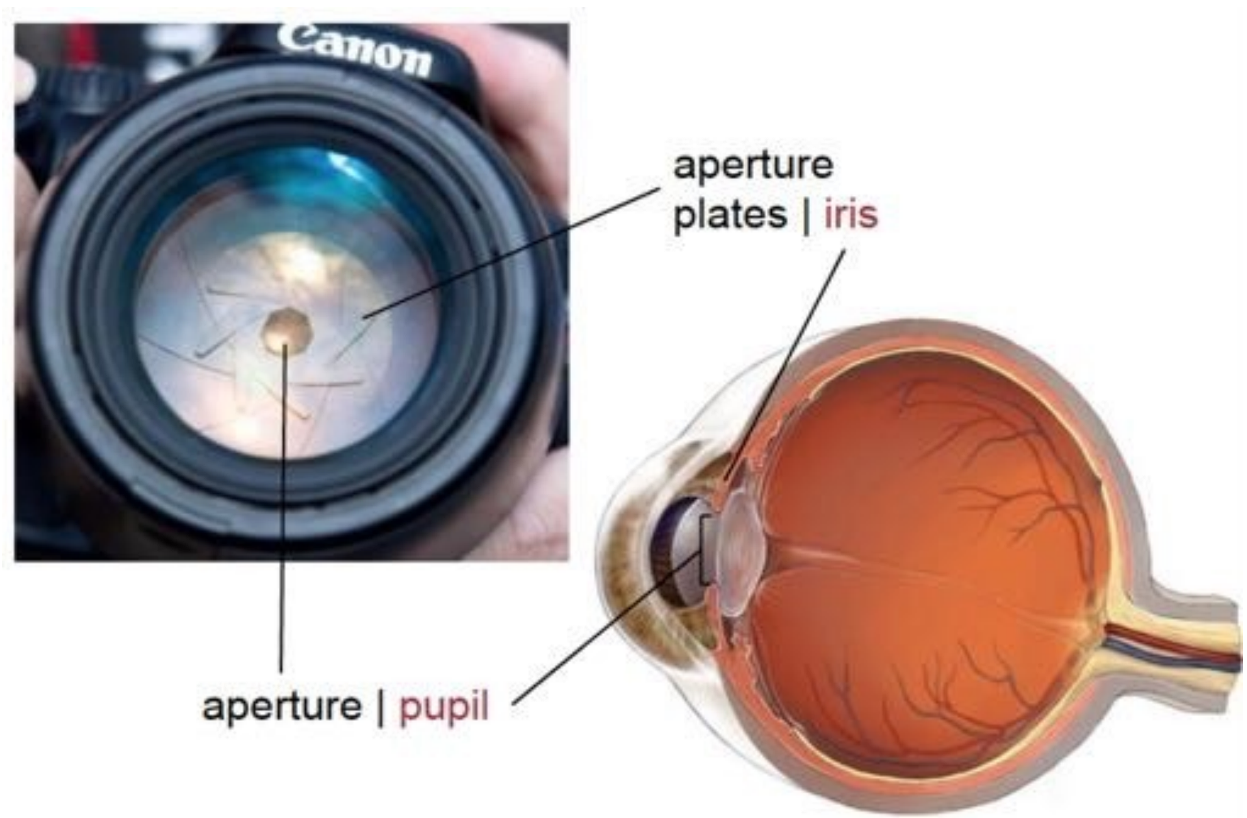
SUBRAHMANYAM MURALA
CVPR Lab
School of Computer Science and Statistics
Trinity College Dublin, Ireland

Source: Sanja Fidler, “CSC420: Intro to Image Understanding Introduction,” University of Toronto (Lectures).

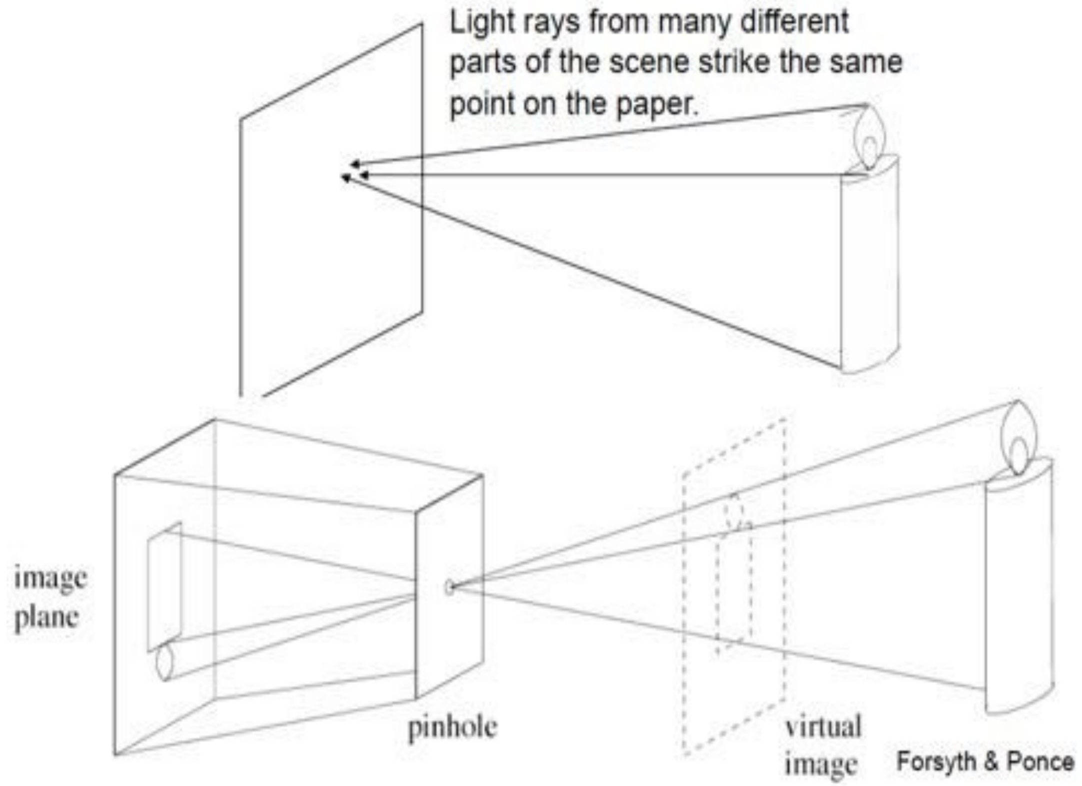
Camera is structurally similar to the eye



Camera is structurally similar to the eye

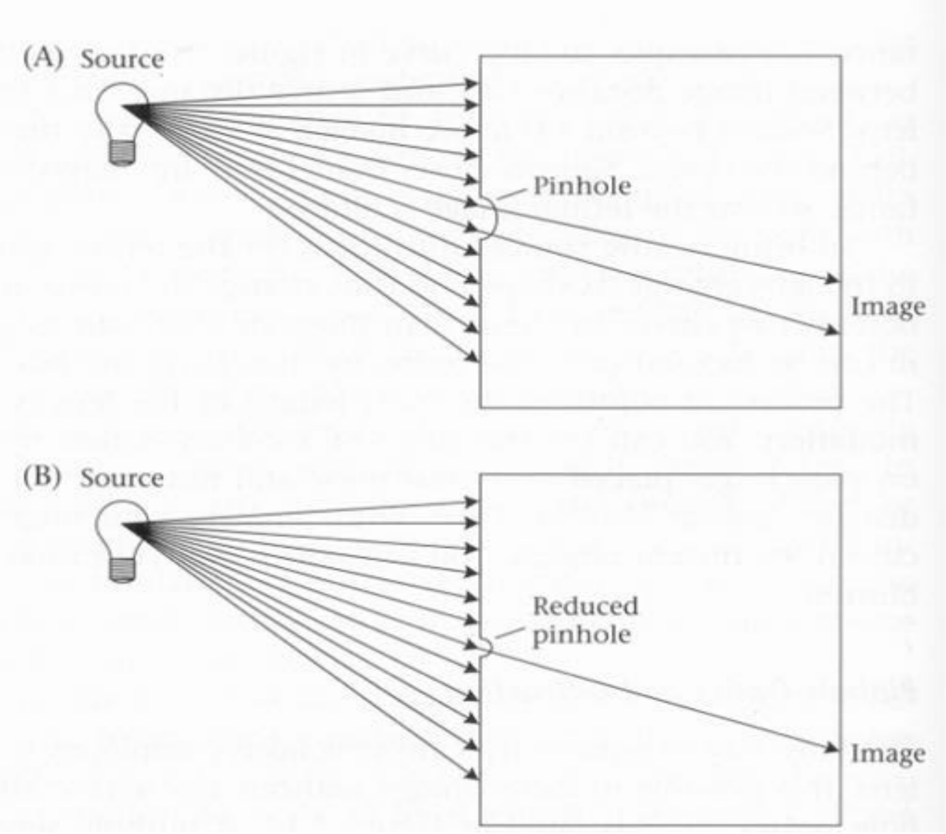


The pinhole camera



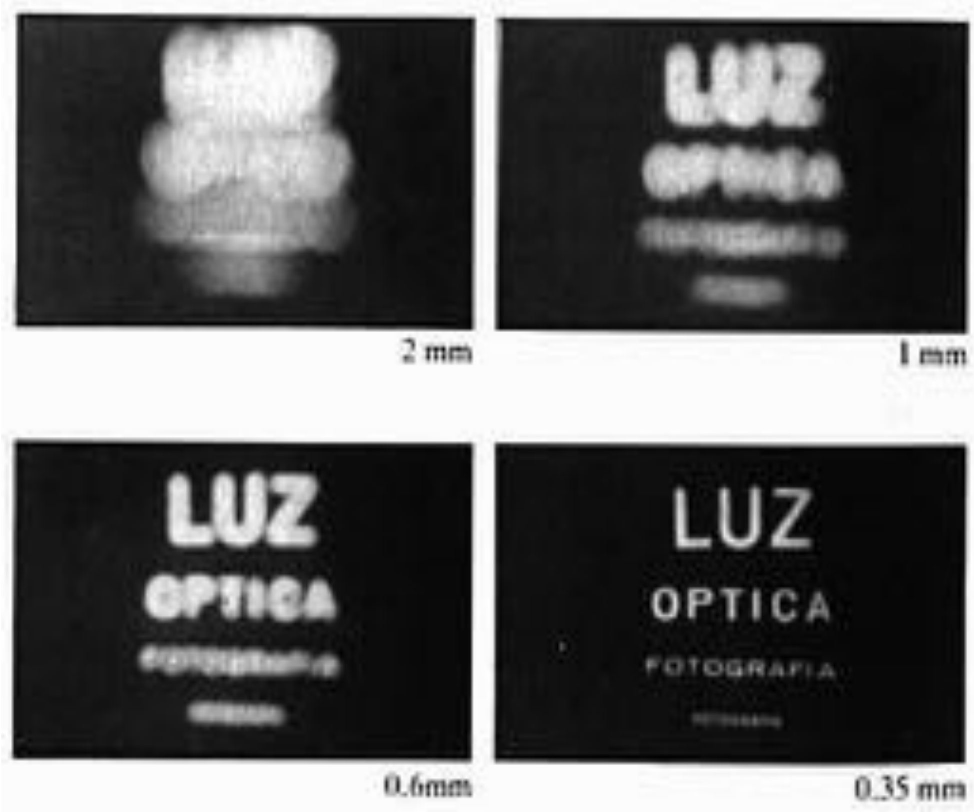
Size of the pinhole is called **aperture**

The pinhole camera



The pinhole camera

Shrinking the Aperture



Why not make the aperture as small as possible?

- Less light gets through
- Diffraction effects...

The pinhole camera

Shrinking the Aperture

Imaging

- Images are 2D projections of real world scene
- Images capture two kinds of information:
  - ✓ Geometric: positions, points, lines, curves, etc.
  - ✓ Photometric: intensity, color
- Complex 3D–2D relationships
- Camera models approximate these relationships

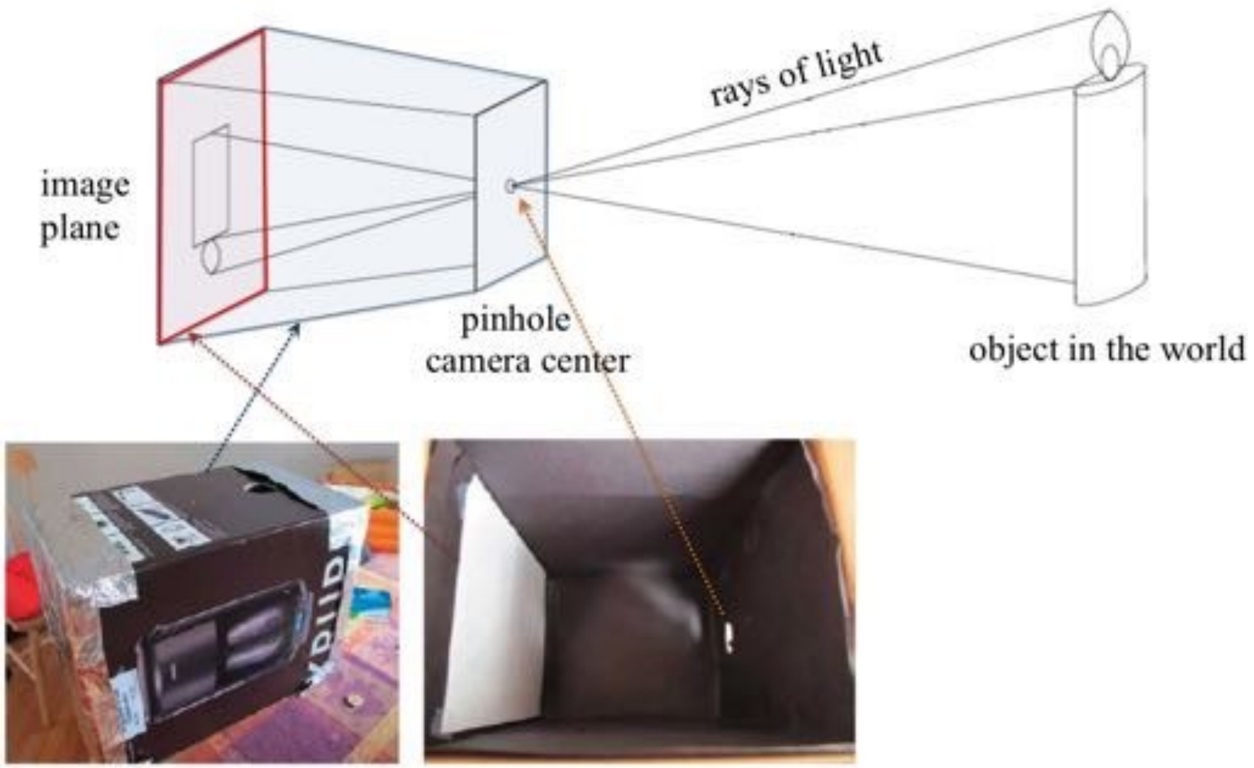
Projection

Projection

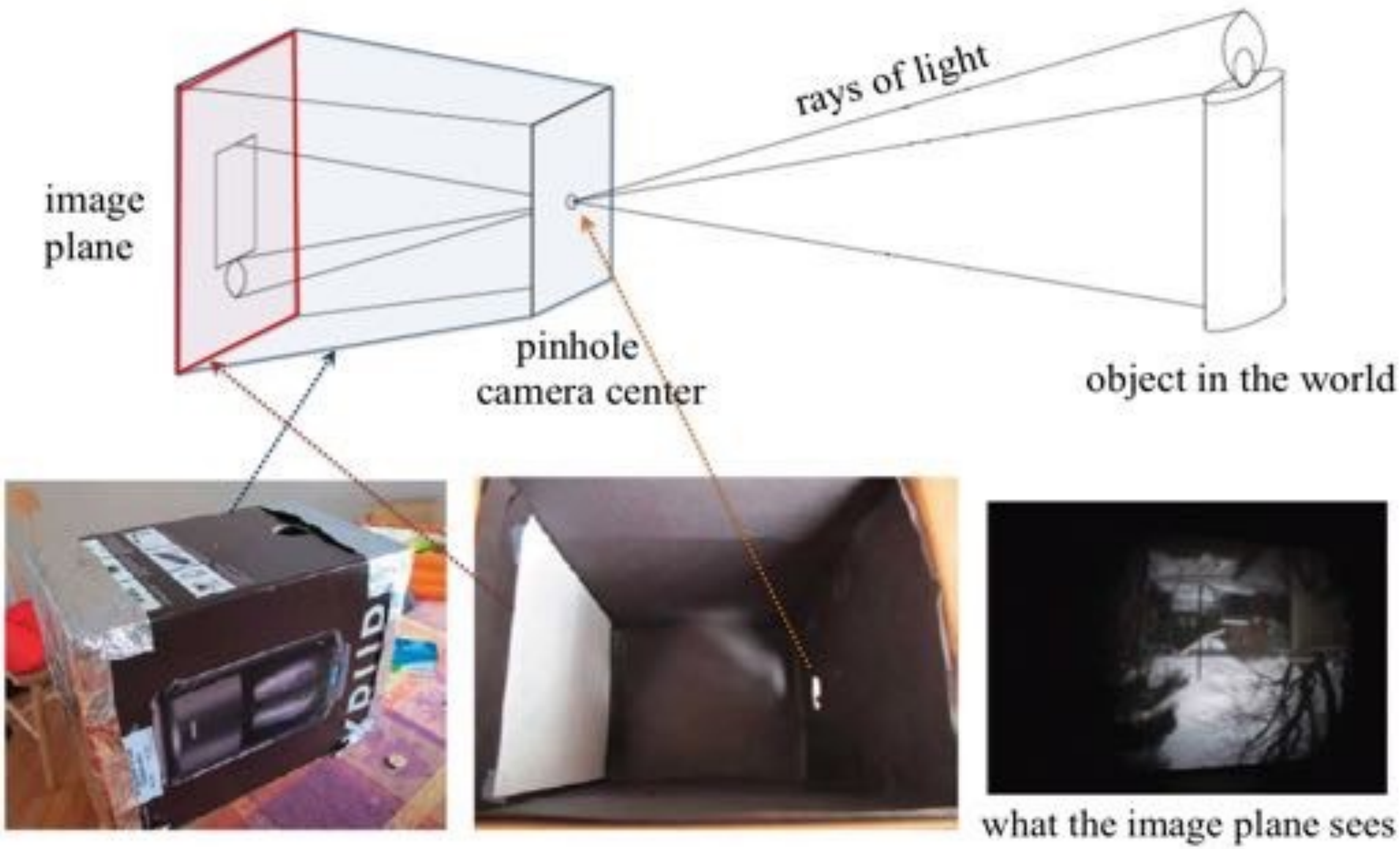
3D to 2D Projection

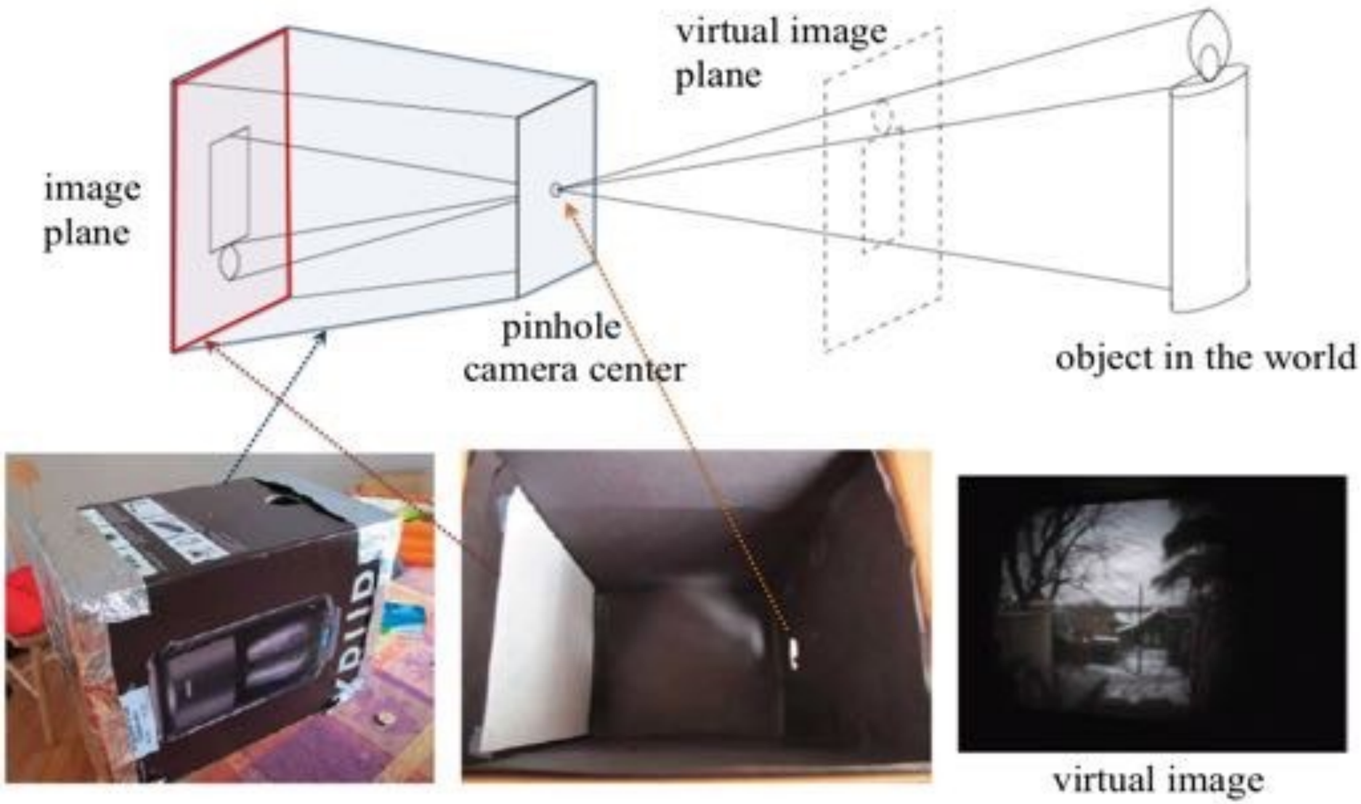
- How are 3D primitives projected onto the image plane?
- We can do this using a linear 3D to 2D projection matrix

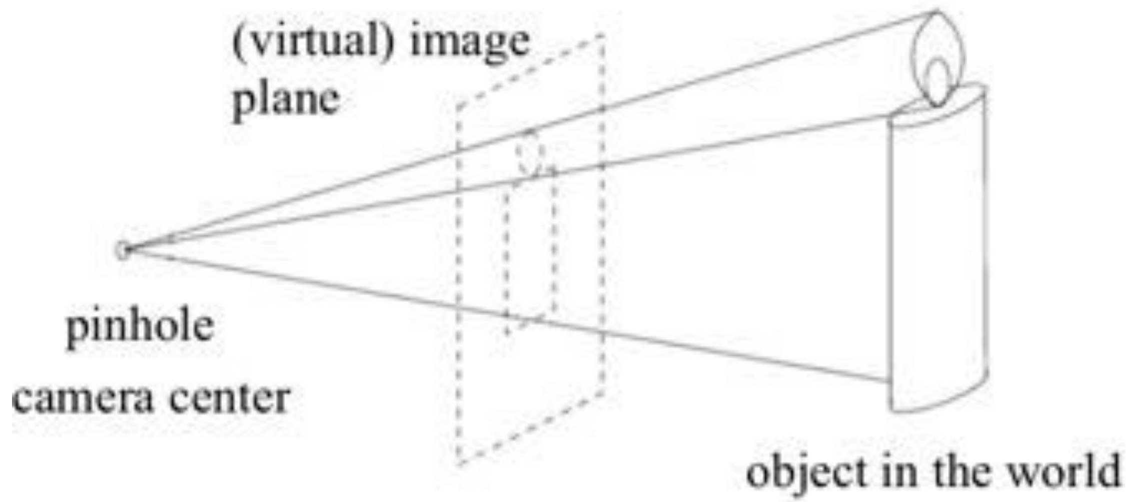
Modeling Projection



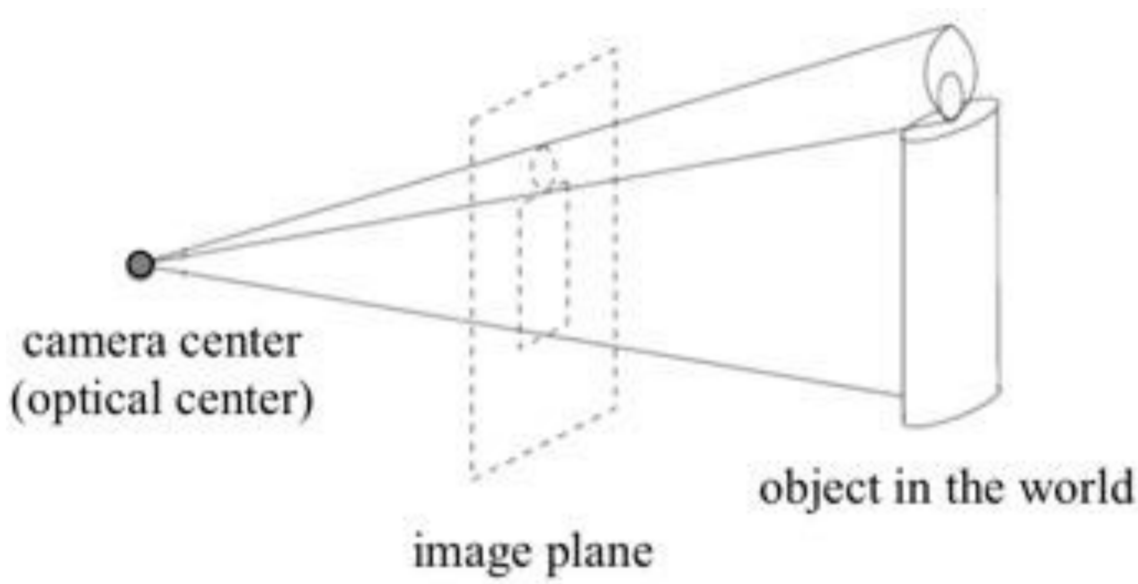
Modeling Projection



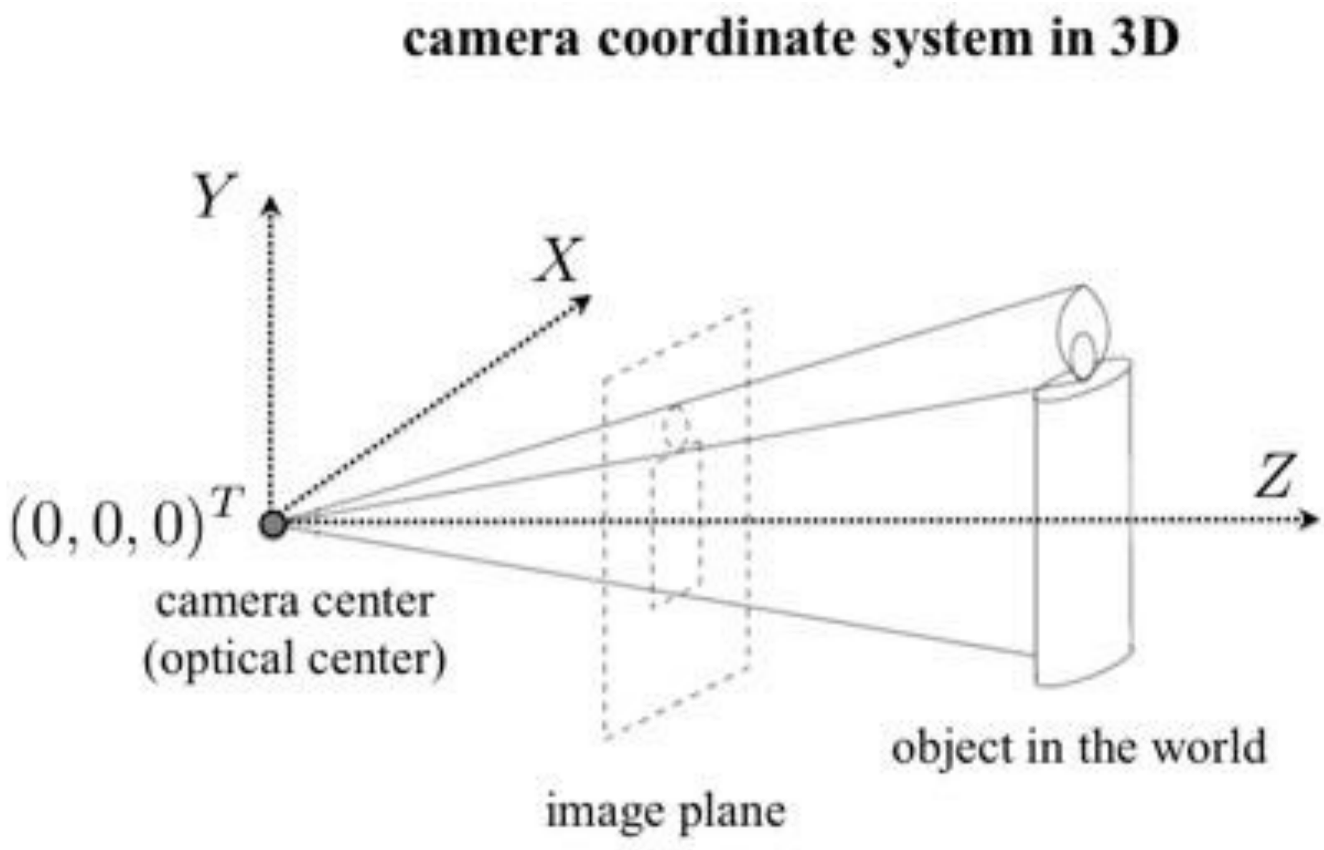

virtual image plane
image plane
pinhole camera center
object in the world
virtual image



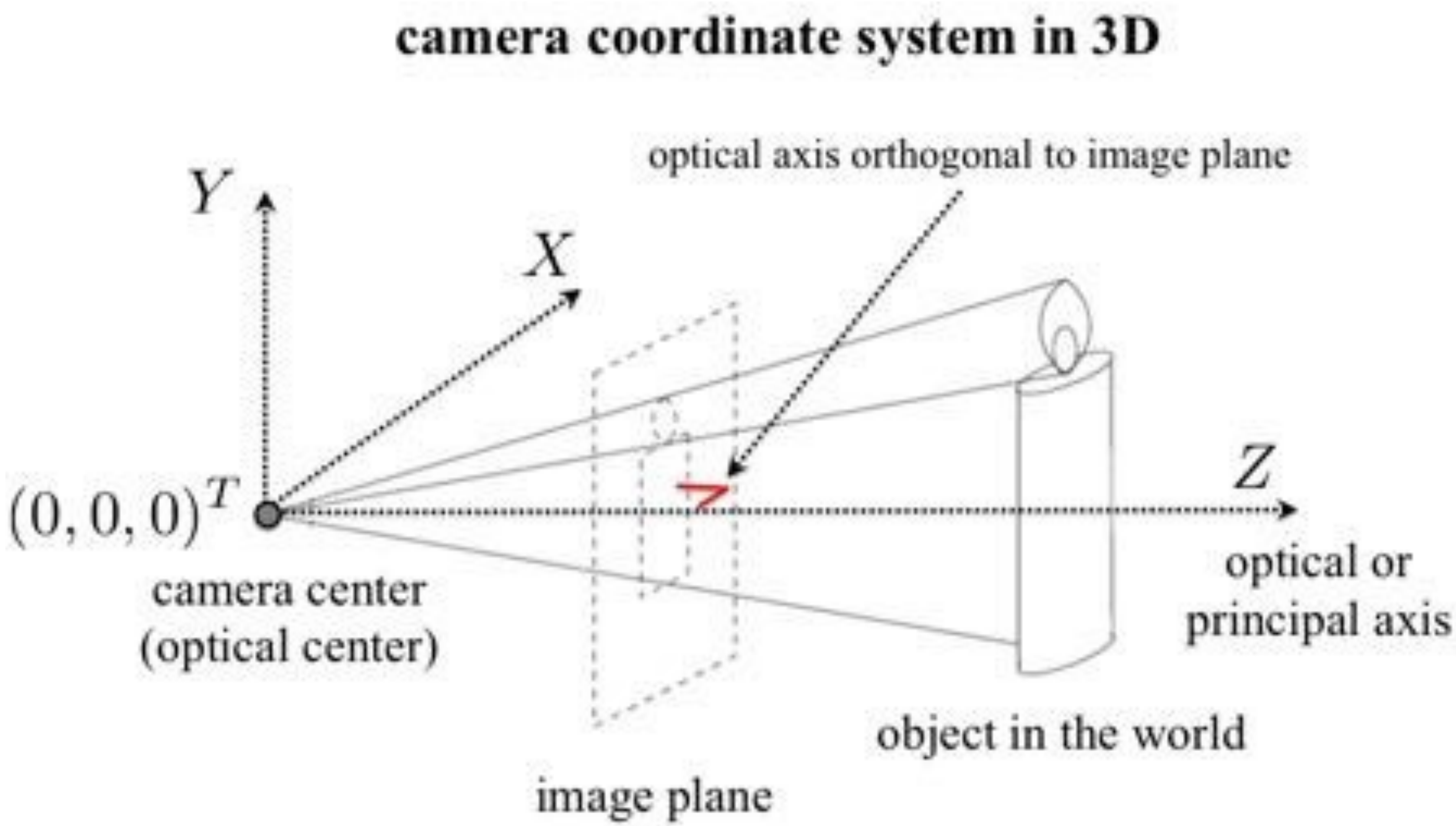
- Since it's easier to think in a non-upsidedown world, we will work with the virtual image plane, and just call it the image plane.
- How do points in 3D project to image plane? If I know a point in 3D, can I compute to which pixel it projects?



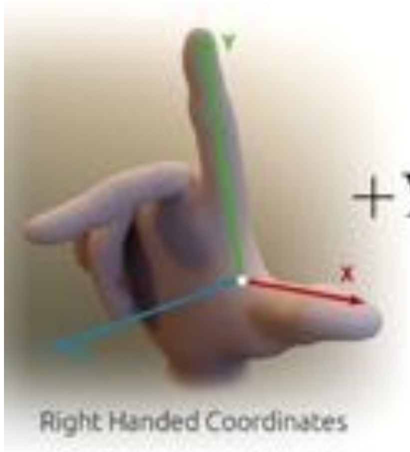
- First some notation which will help us derive the math
- To start with, we need a coordinate system

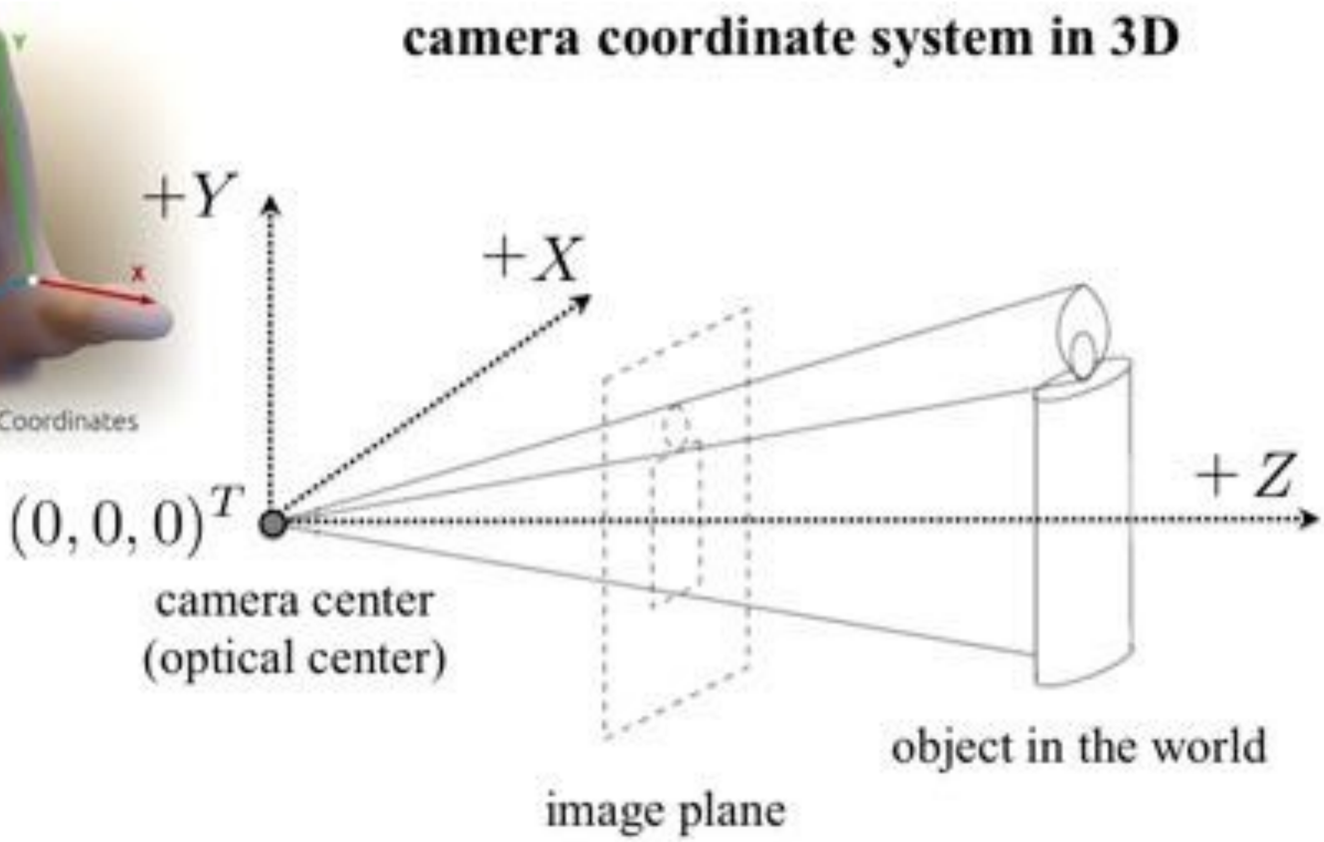


- We place a coordinate system relative to camera: **optical center** or **camera center C** is thus at origin (0, 0, 0).

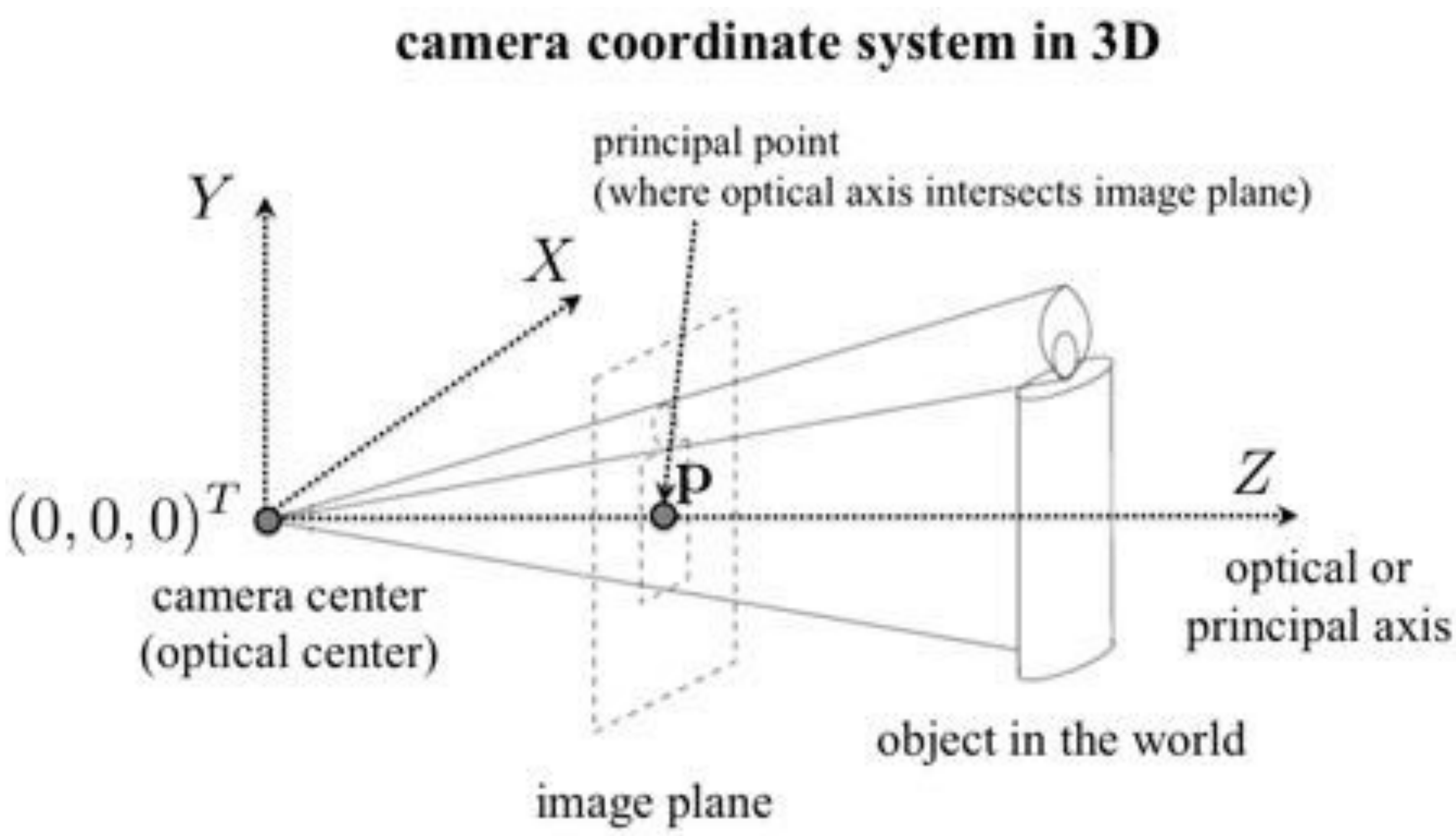


- The Z axis is called the **optical** or **principal axis.** It is orthogonal to the image plane. Axes X and Y are parallel to the image axes.

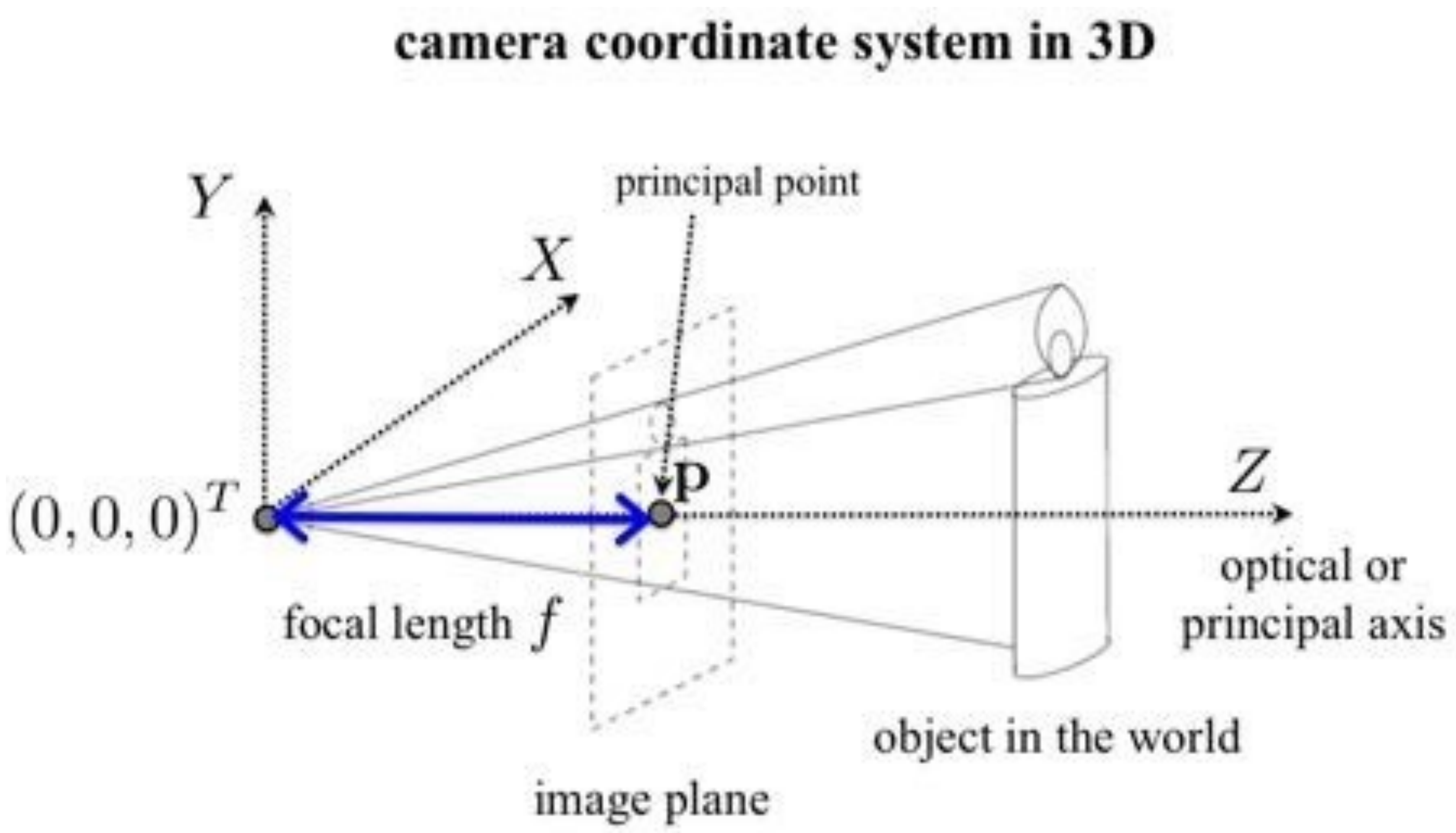




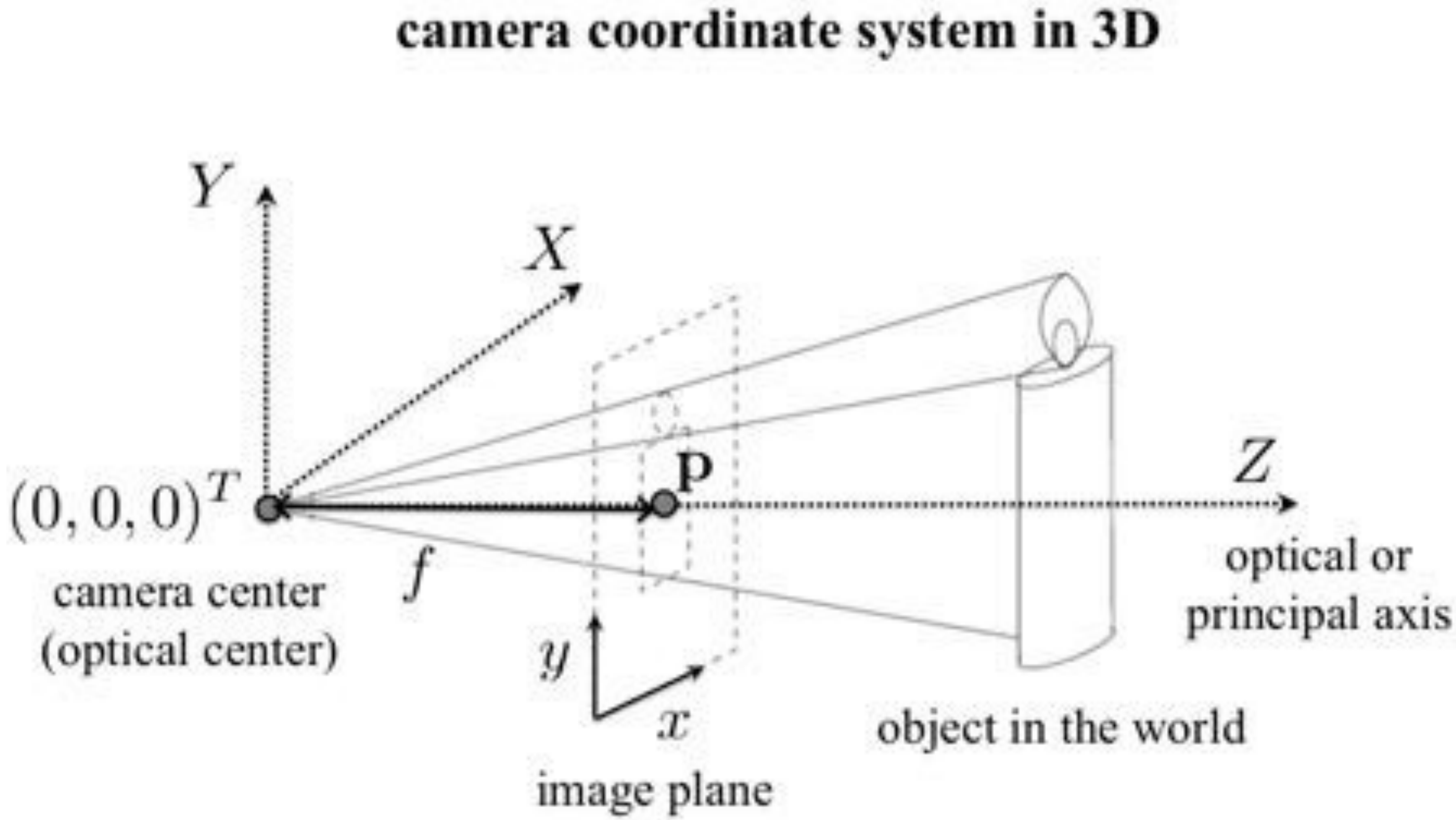
- We will use a **right handed** coordinate system



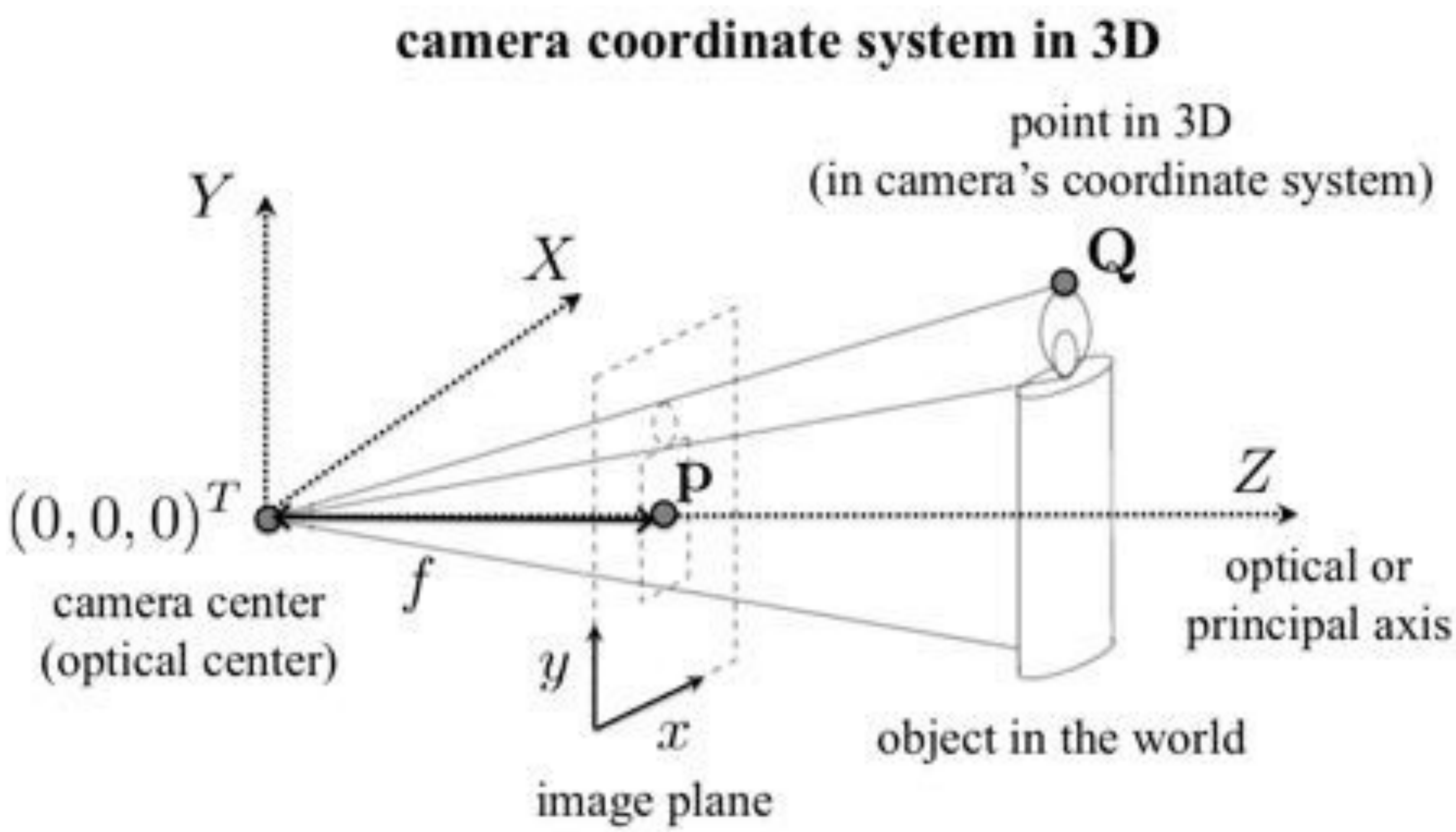
- The optical axis intersects the image plane in a point, **p**. We call this point a **principal point**.



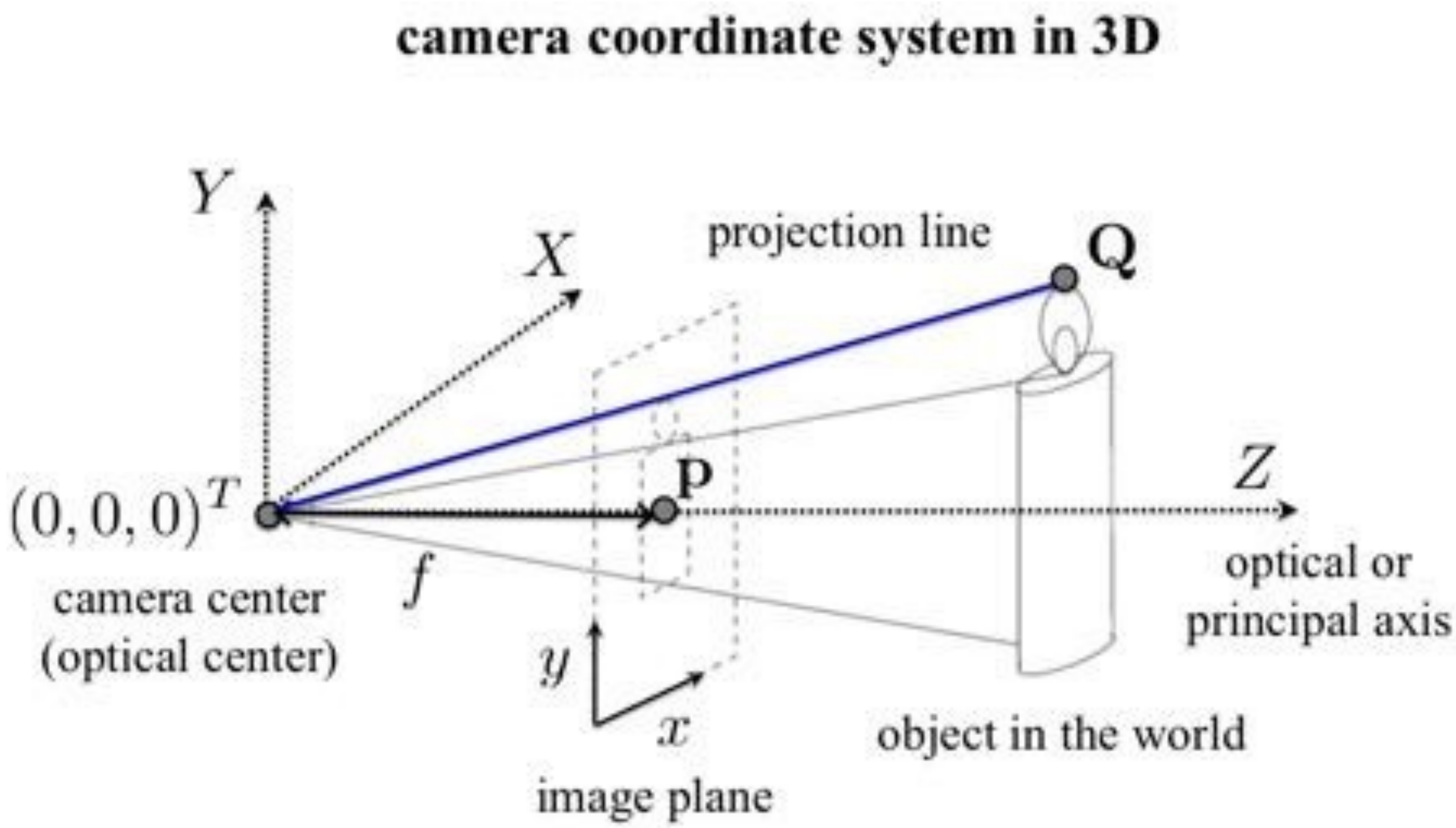
- The distance from the camera center to the principal point is called **focal length**, we will denote it with f .



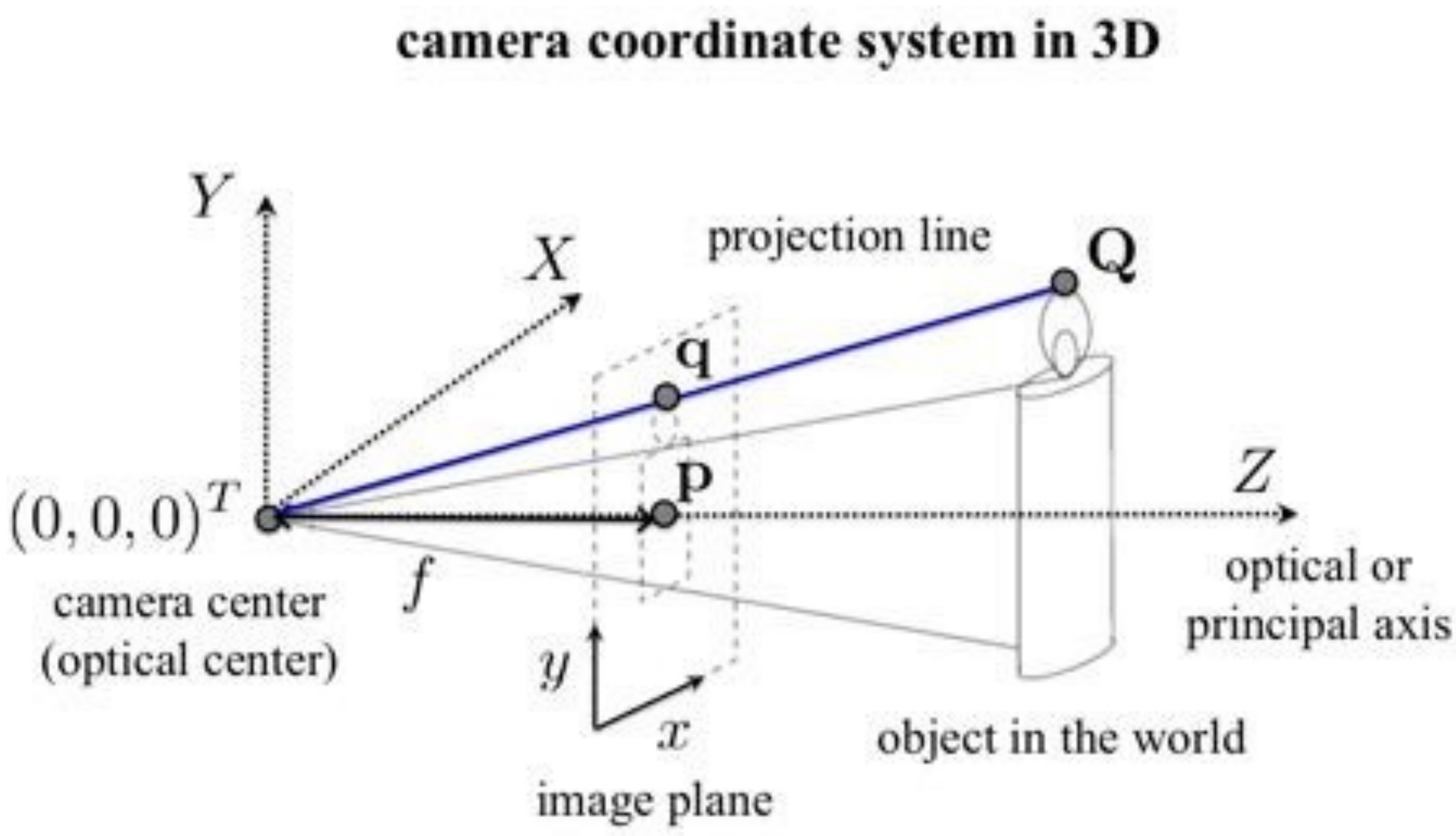
- We'll denote the image axes with *x* and *y* . An image we see is of course represented with these axes. We'll call this an **image coordinate system.**
- The tricky part is how to get from the camera's coordinate system (3D) to the image coordinate system (2D).



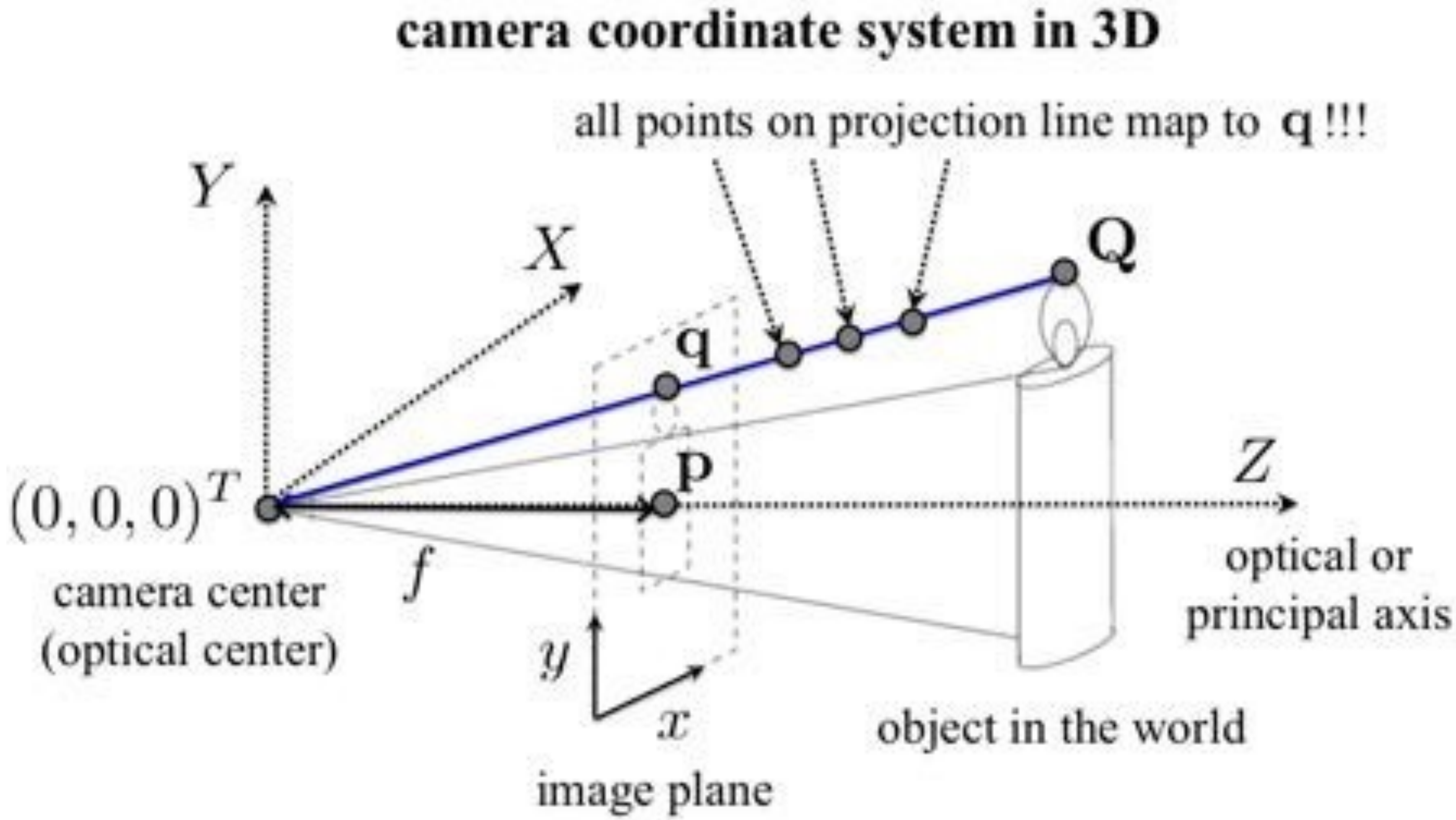
- Let's take some point **Q** in 3D. **Q** "lives" relative to the camera; its coordinates are assumed to be in camera's coordinate system.



- We call the line from **Q** to camera center a **projection line.**



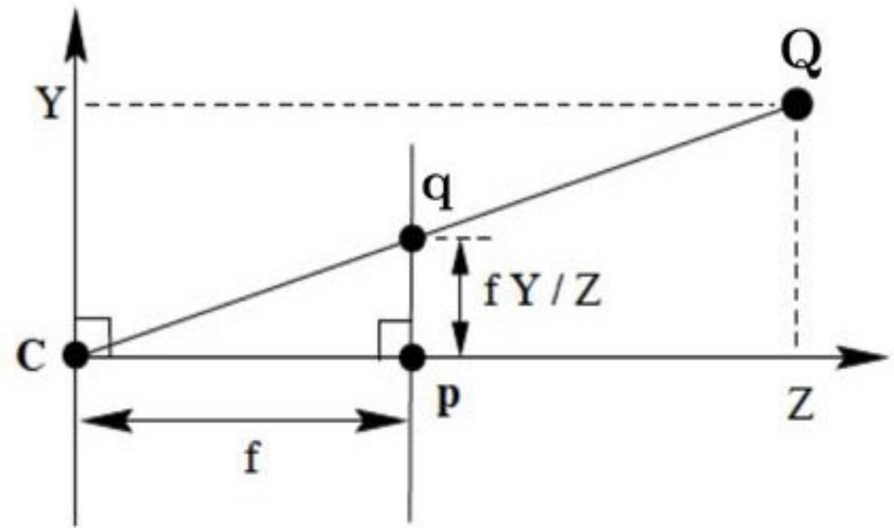
- The projection line intersects the image plane in a point **q**. This is the point we see in our image.



- First thing to notice is that all points from **Q**'s projection line project to the same point **q** in the image!
- **Ambiguity**: It's impossible to know how far a 3D point is from the camera along the projection line by looking only at the image (point **q**).

- **Ambiguity**: It's impossible to know how far a 3D point is from the camera along the projection line by looking only at the image (point **q**).
- It's impossible to know the real 3D size of objects just from an image

  Why did the detective put a dollar bill next to the footprint?
- How would you compute the shoe's dimensions?



Projection Equations

Using similar triangles:

$$\mathbf{Q} = (X, Y, Z)^T \quad \rightarrow \quad \left(\frac{f \cdot X}{Z}, \frac{f \cdot Y}{Z}, f\right)^T$$

Projection properties

Many-to-one: any points along same ray map to same point in image

Points $\rightarrow$ points

Lines $\rightarrow$ lines. Why?

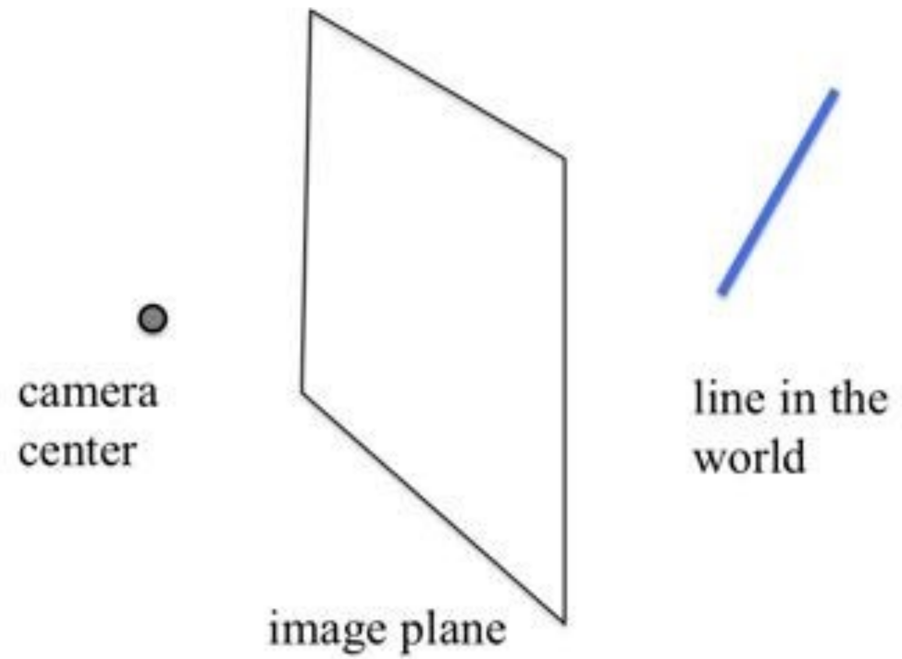


Figure: Proof by drawing

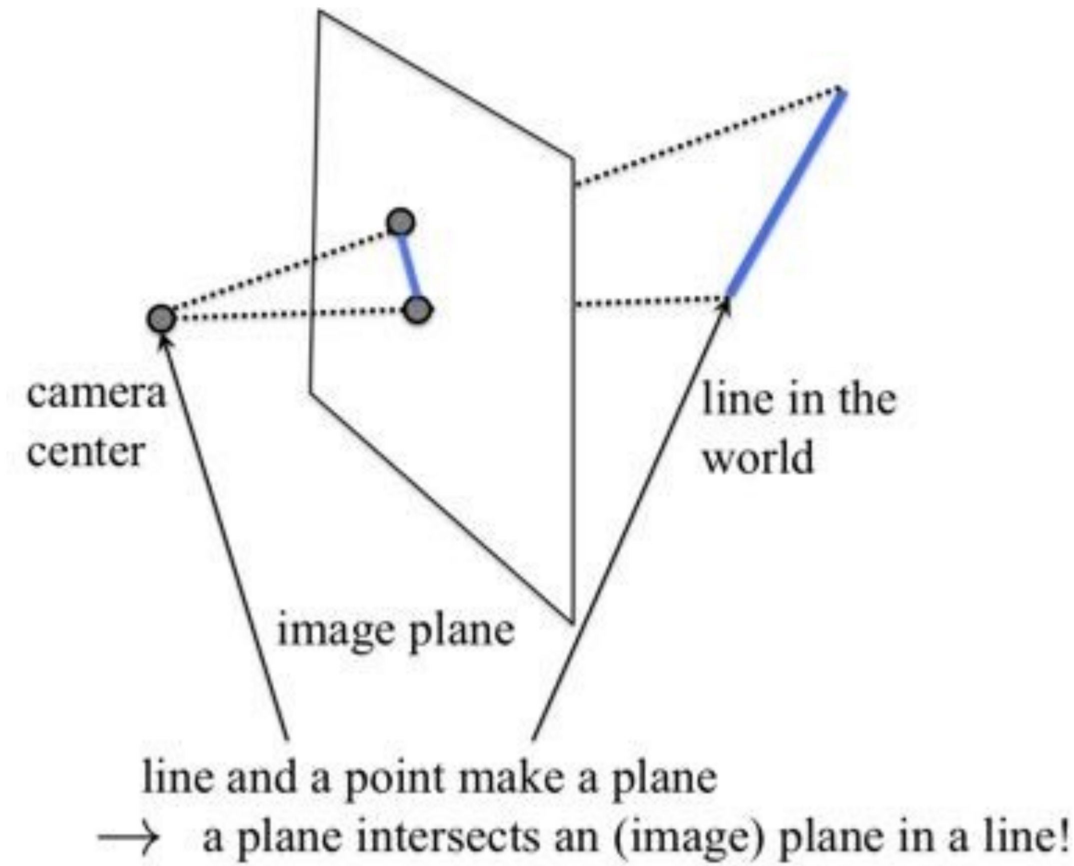


Figure: Proof by drawing

Many-to-one: any points along same ray map to same point in image

Points $\longrightarrow$ points

Lines $\longrightarrow$ lines

But line through principal point projects to a point. Why?

Figure: Can you tell where is the principal point?

Many-to-one: any points along same ray map to same point in image

Points $\longrightarrow$ points

Lines $\longrightarrow$ lines

But line through principal point projects to a point. Why?

Planes $\longrightarrow$ planes

Many-to-one: any points along same ray map to same point in image

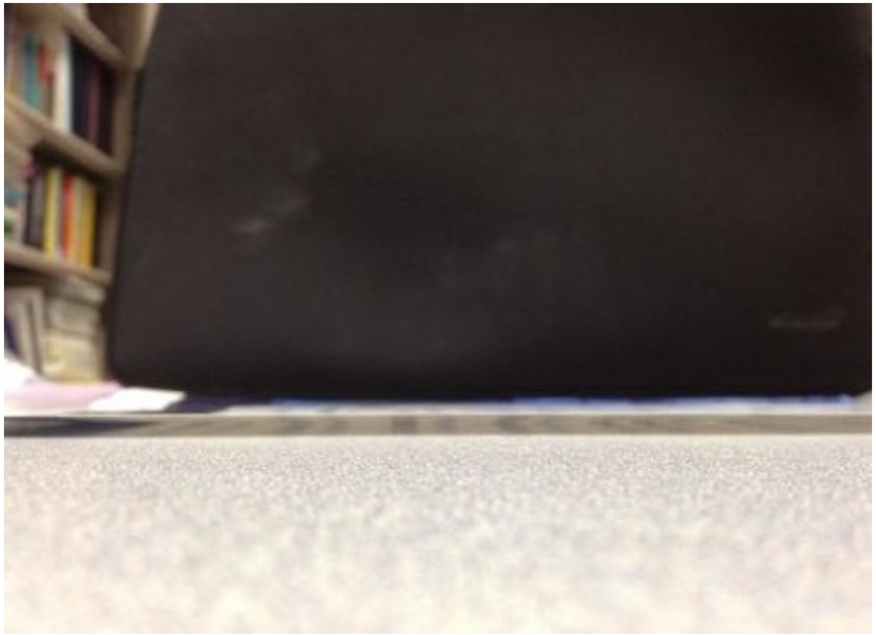
Points $\longrightarrow$ points

Lines $\longrightarrow$ lines

But line through principal point projects to a point. Why?

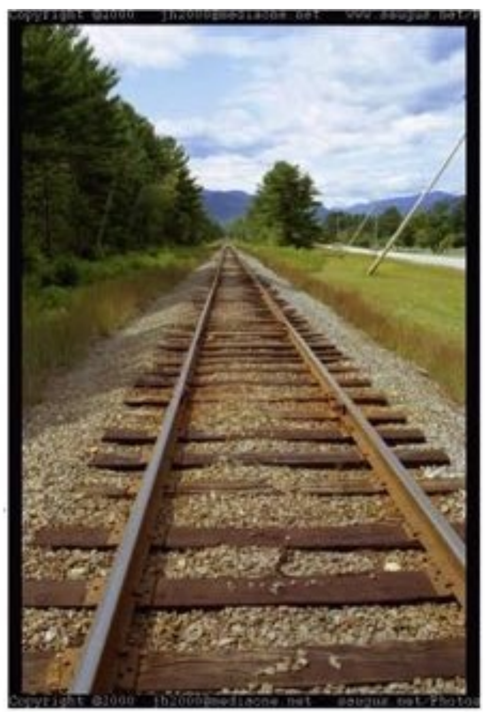
Planes $\longrightarrow$ planes

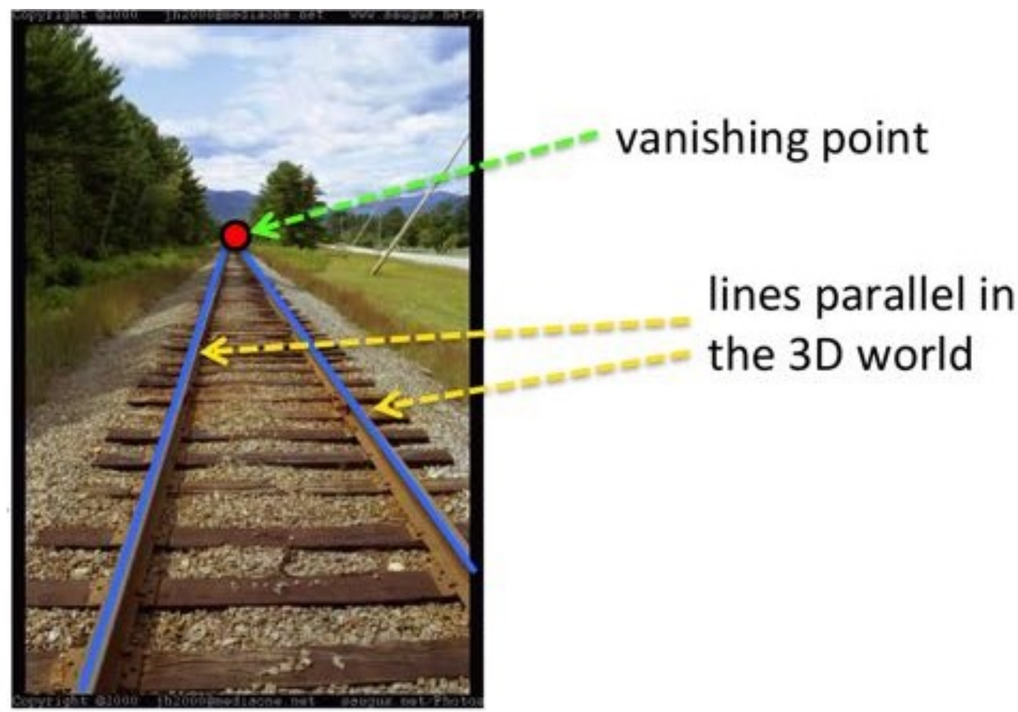
But plane through principal point which is orthogonal to image plane projects to line. Why?



Projection Properties: Cool Facts

- Parallel lines converge at a **vanishing point**
- Each different direction in the world **has its own vanishing point**

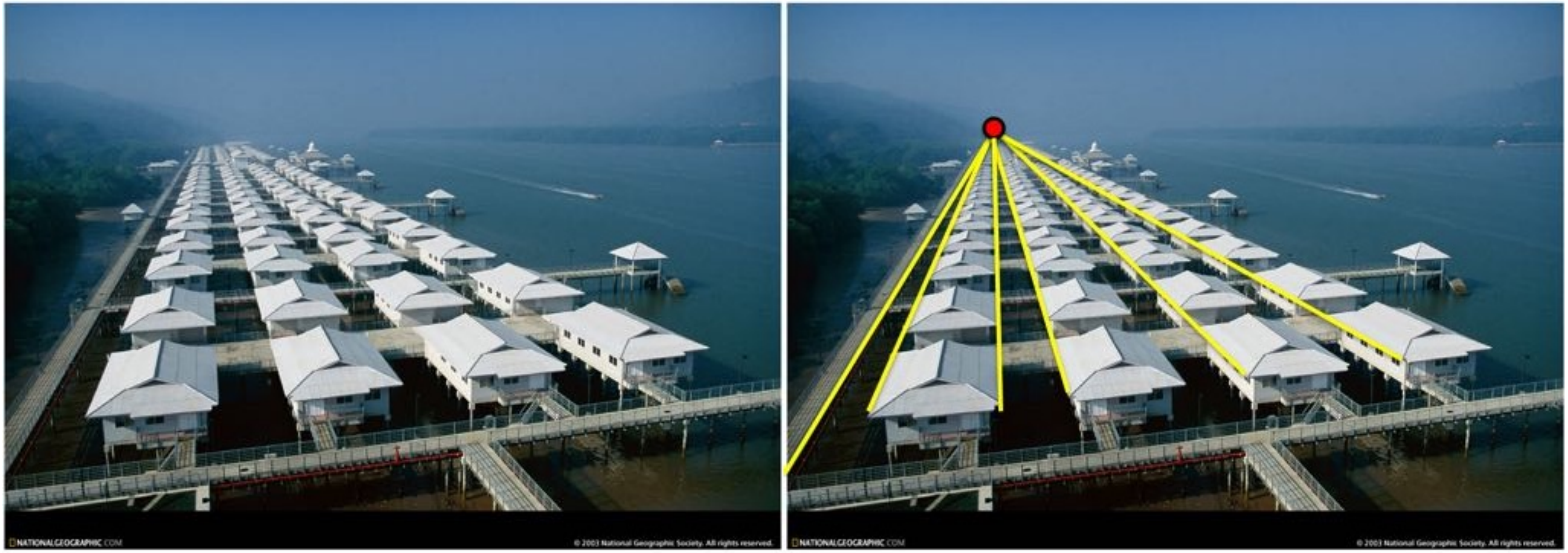




[Adopted from: N. Snavely, R. Urtasun]

Parallel lines converge at a **vanishing point**

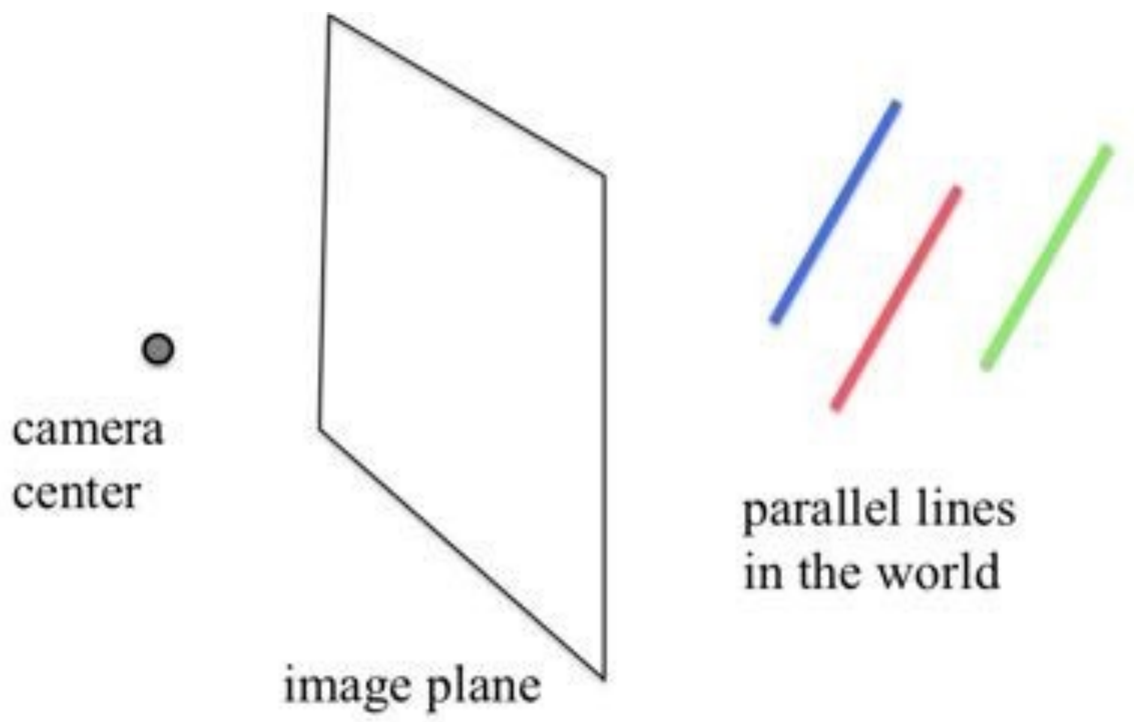
- Each different direction in the world **has its own vanishing point**
- All lines with the same 3D direction intersect at the **same vanishing point**



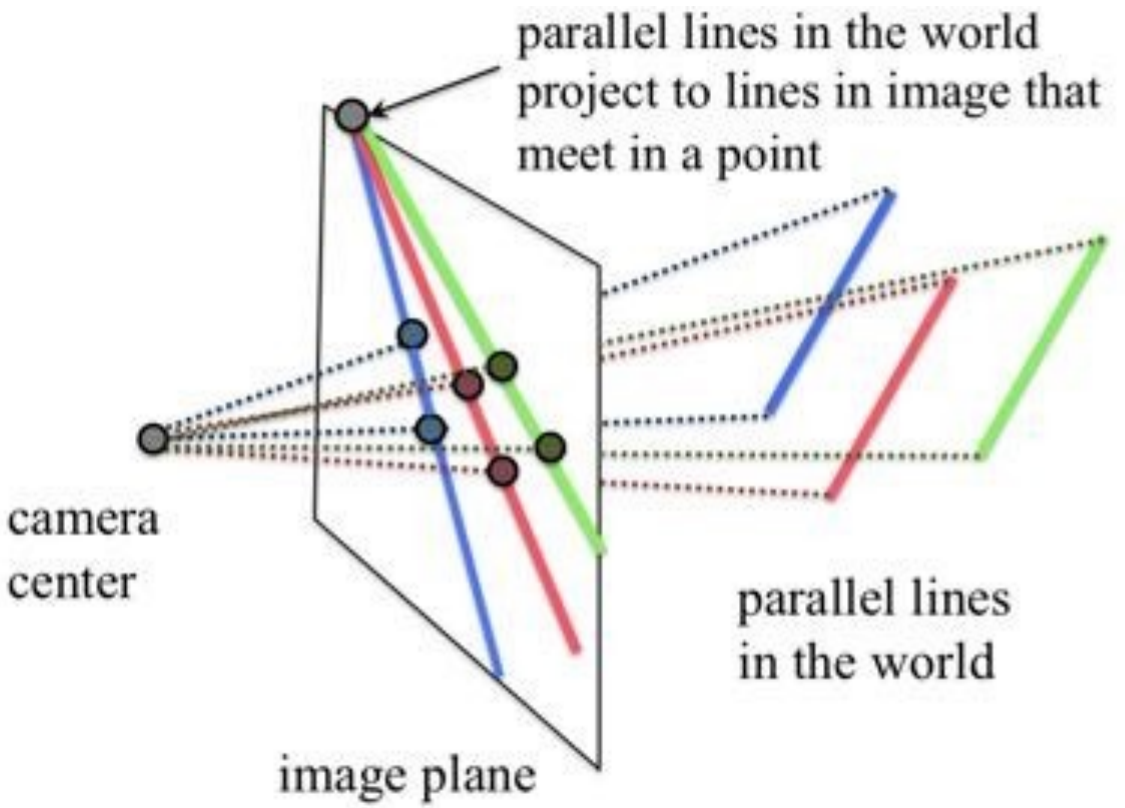
[Pic: R. Szeliski]

Projection Properties: Vanishing Point

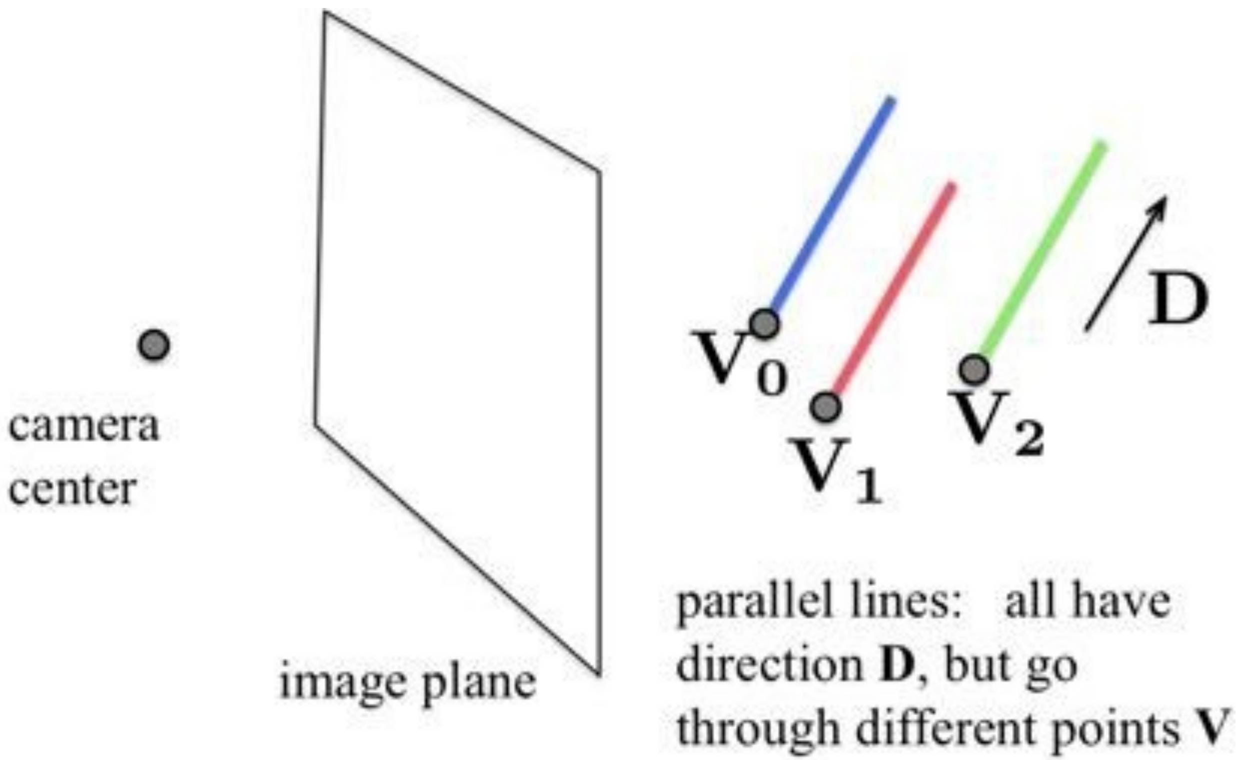
All lines with the same 3D direction intersect at the **same vanishing point. Why?**



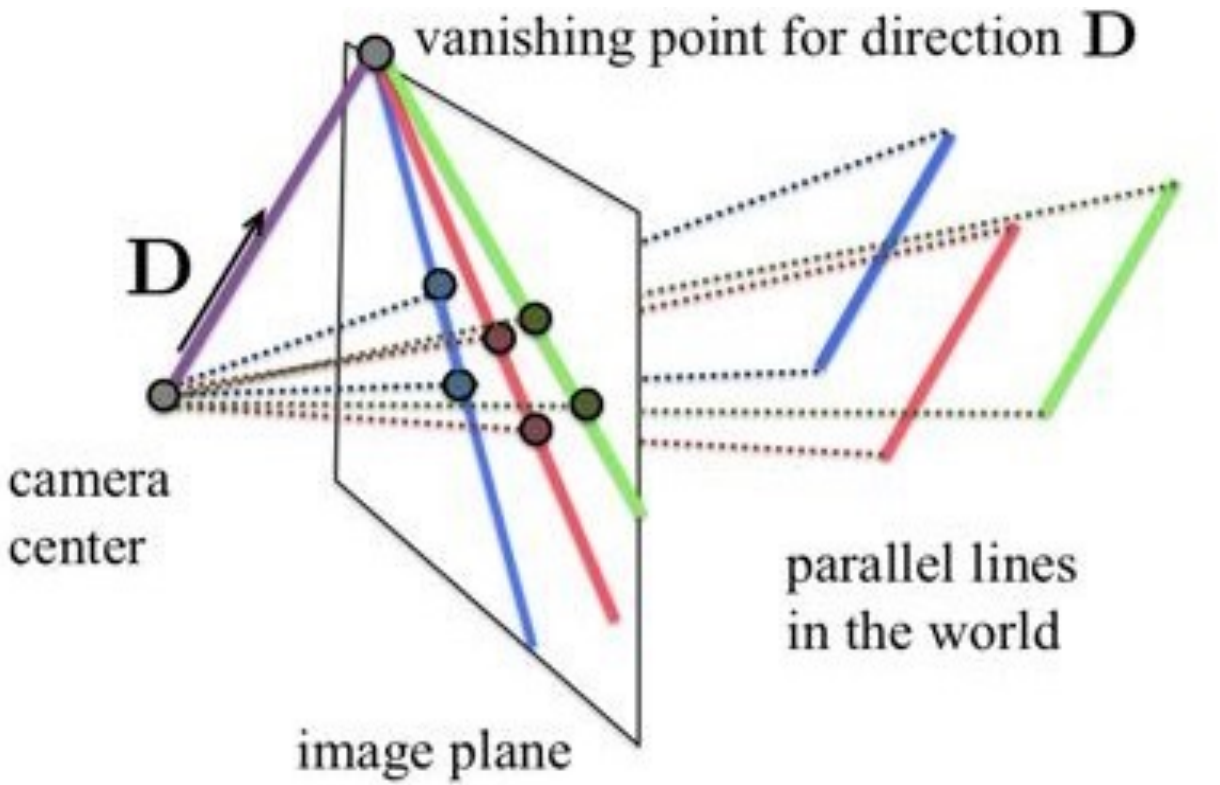
All lines with the same 3D direction intersect at the **same vanishing point. Why?**



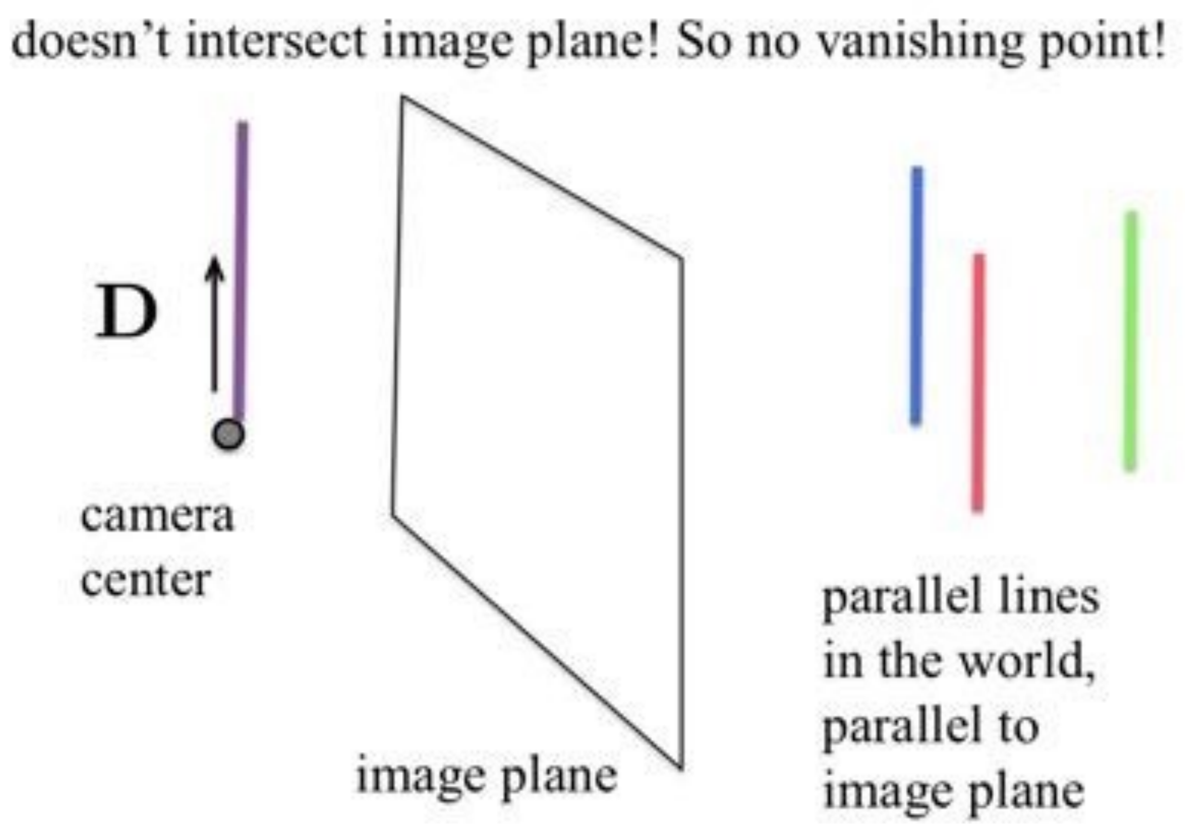
All lines with the same 3D direction intersect at the **same vanishing point. Why?**



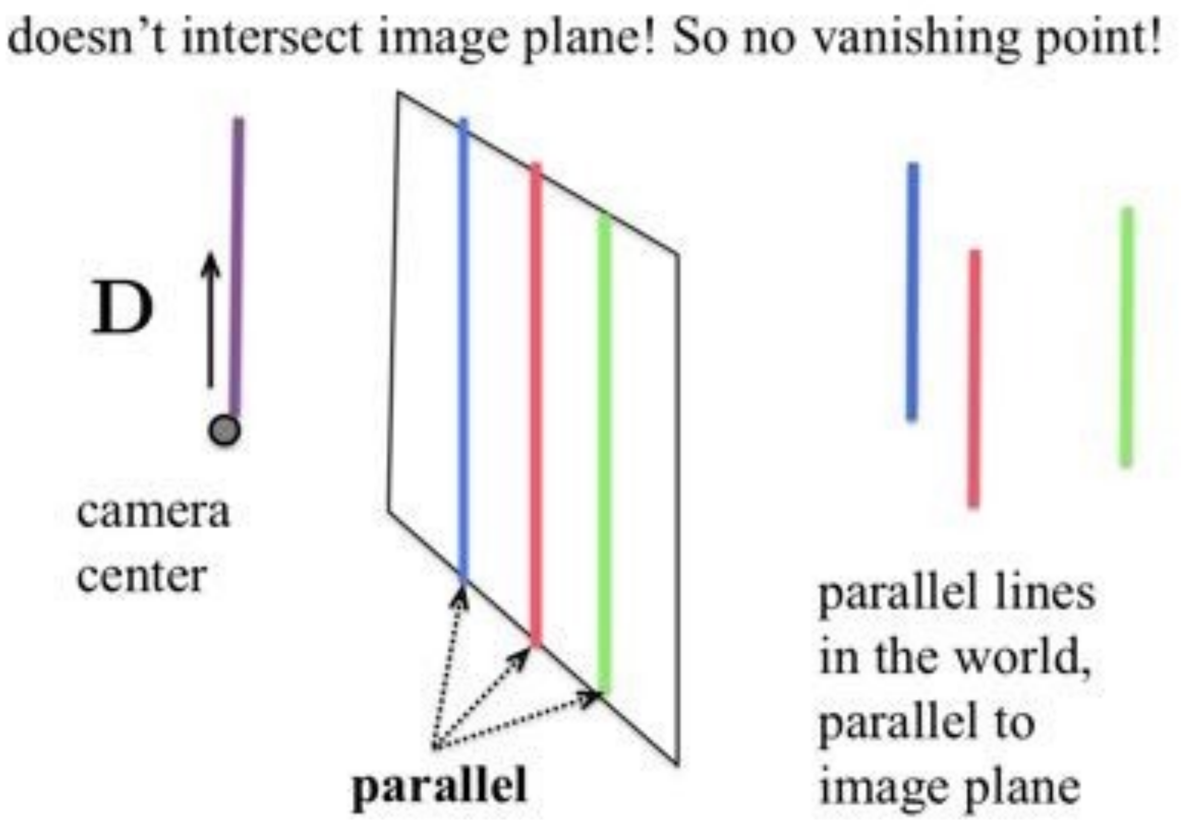
- All lines with the same 3D direction intersect at the **same vanishing point.**
- The easiest way to find this point: Translate line with direction **D** to the camera center. This line intersects the image plane in the vanishing point corresponding to direction **D**!



Lines parallel to image plane are also parallel in the image. We say that they intersect at infinity.



Lines parallel to image plane are also parallel in the image. We say that they intersect at infinity.



Projection Properties: Cool Tricks

- This picture has been recorded from a car with a camera on top. We know the camera intrinsic matrix K .
- Can we tell the incline of the hill we are driving on?
- How?

Can we tell the incline of the hill we are driving on?

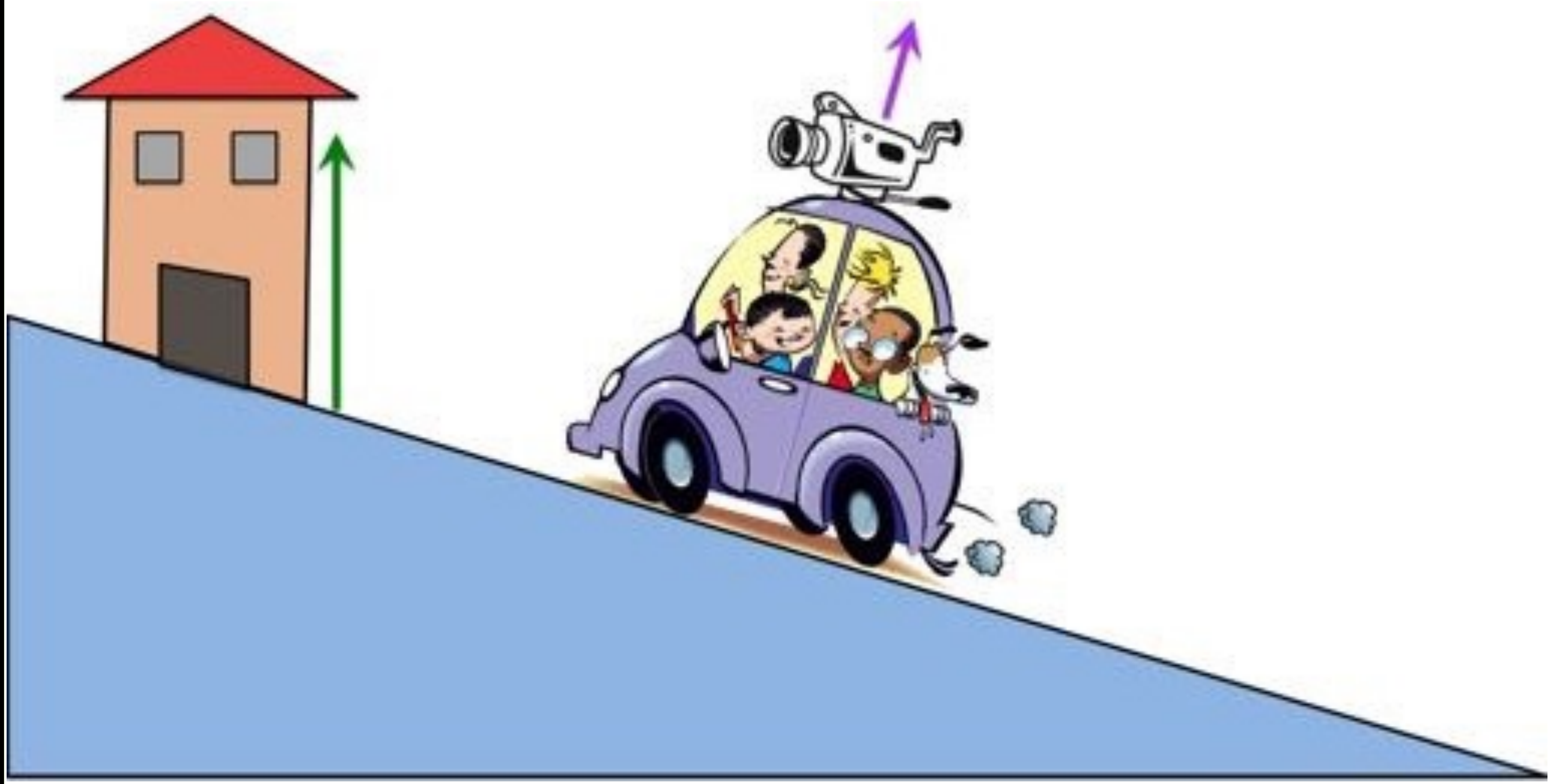


Figure: This is the 3D world behind the picture.

Can we tell the incline of the hill we are driving on?

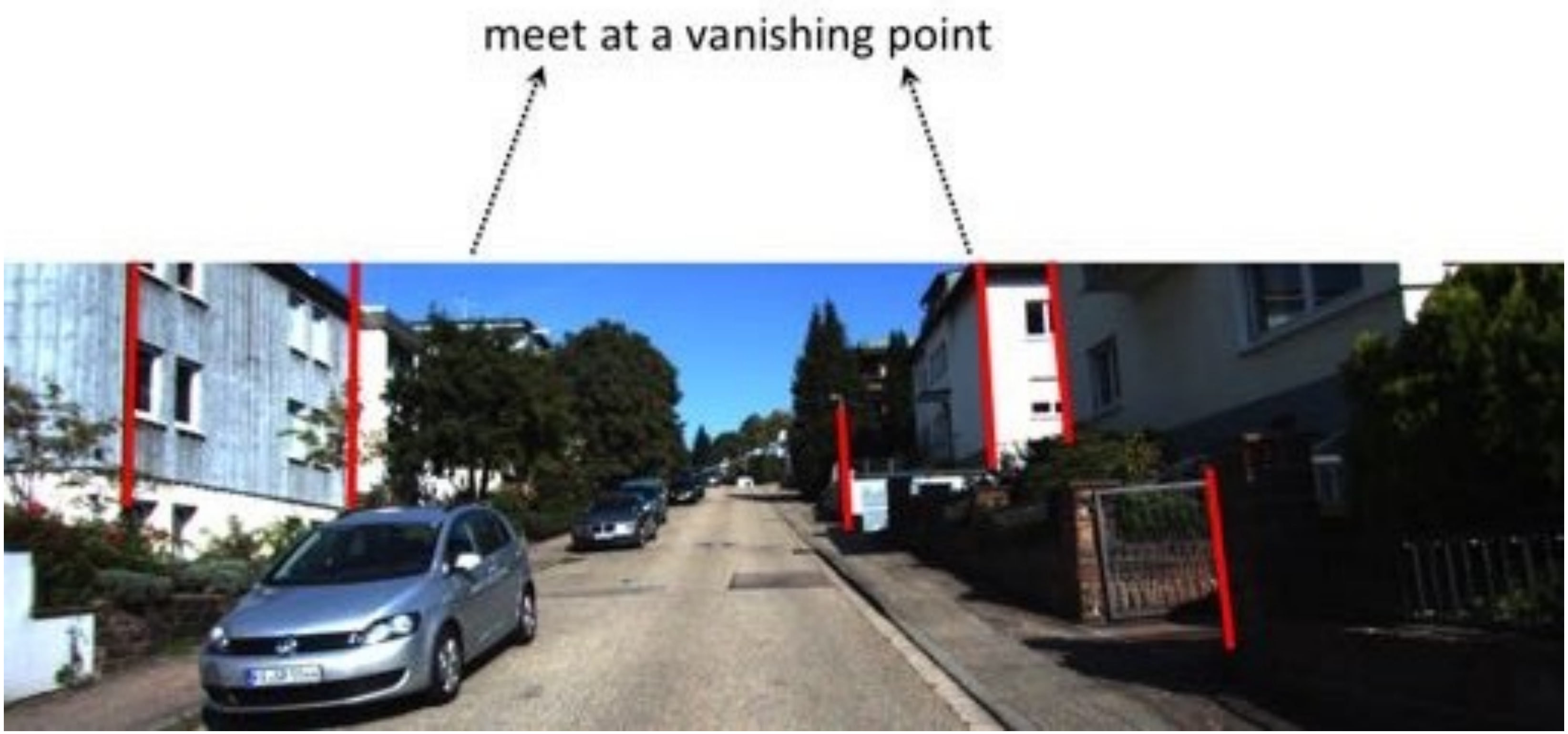


Figure: Extract "vertical" lines and compute vanishing point. How can we compute direction in 3D from vanishing point (if we have K)?

Can we tell the incline of the hill we are driving on?

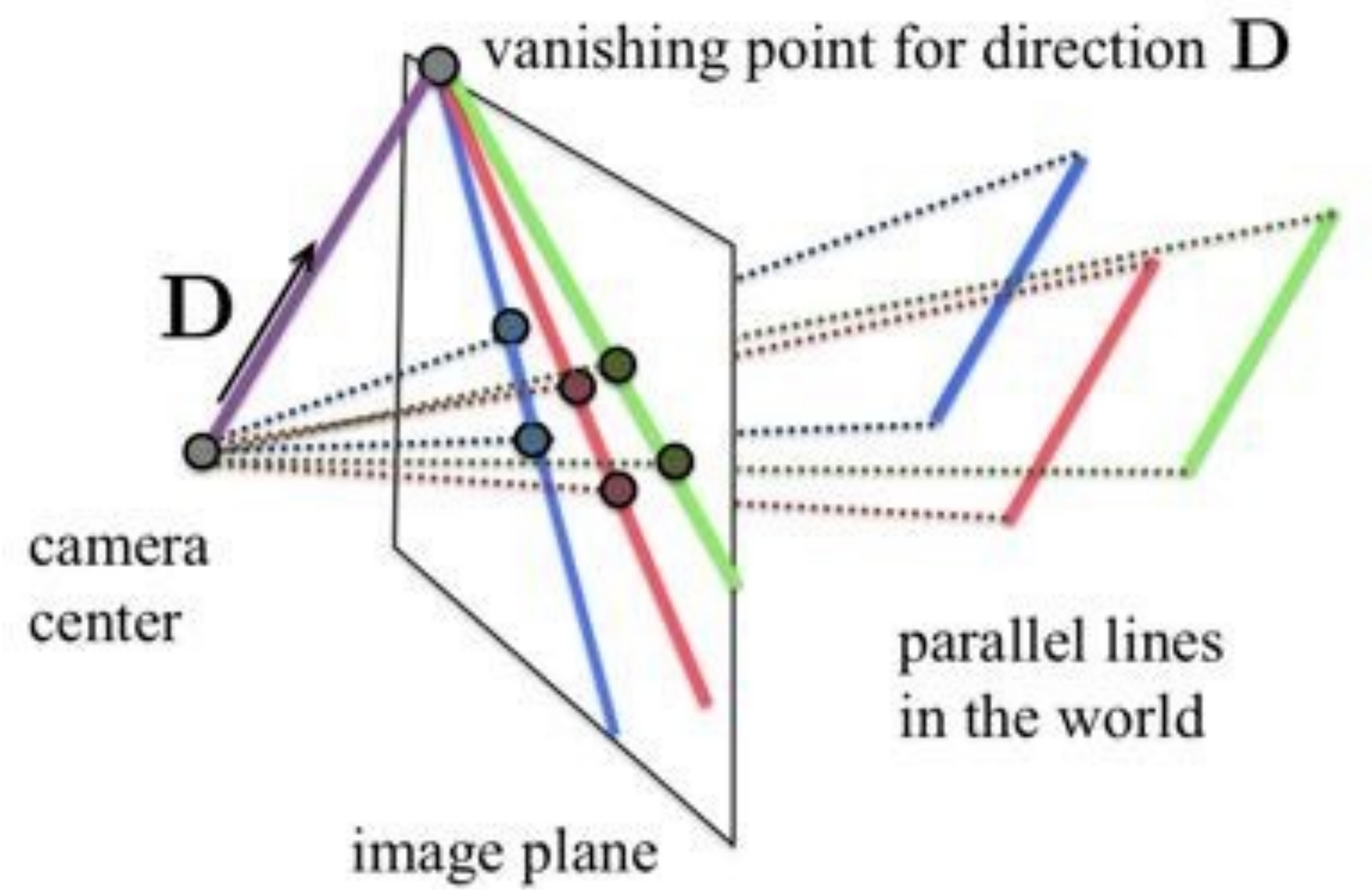
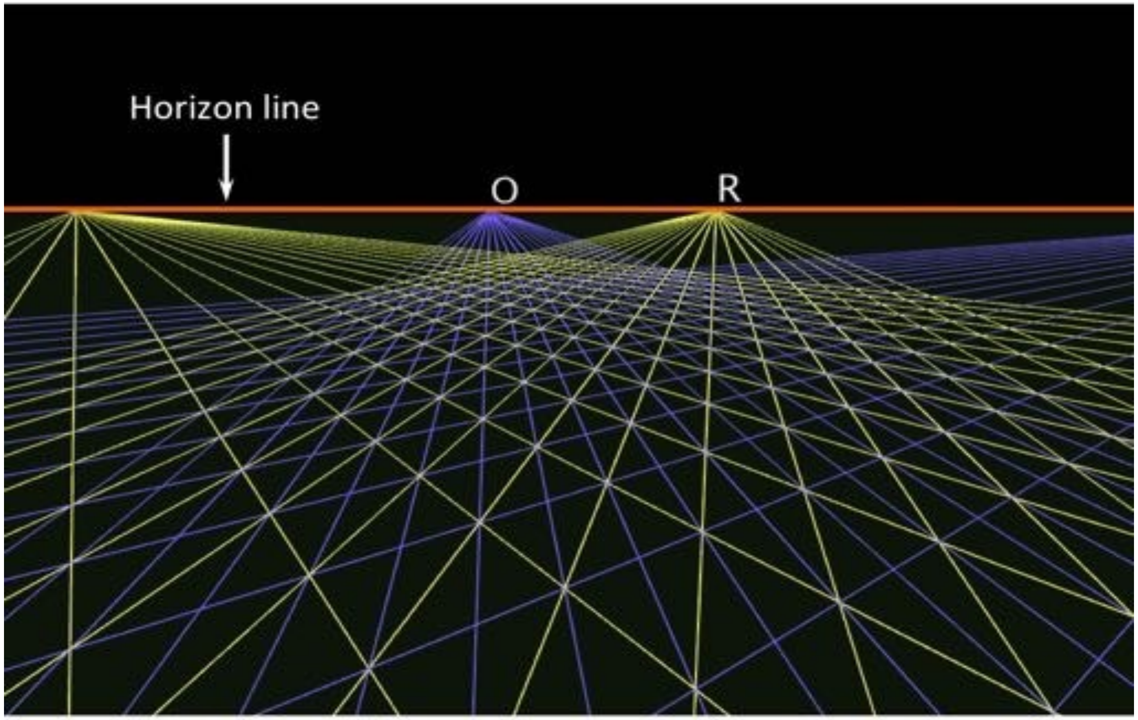


Figure: This picture should help.

Projection Properties: Cool Facts

Parallel lines converge at a **vanishing point**

- Each different direction in the world **has its own vanishing point**
- For lines on the same 3D plane, the vanishing points lie on a **line.** We call it a **vanishing line.** Vanishing line for the ground plane is a **horizon line.**



http://4.bp.blogspot.com/-0Jm9d9j35Tc/T5ESbVpKI7I/AAAAAAAACEk/nVAiTxBuiyc/s1600/perspectiveGrid-01.png

Parallel lines converge at a **vanishing point**

- For lines on the same 3D plane, the vanishing points lie on a **line.** We call it a **vanishing line.** Vanishing line for the ground plane is a **horizon line.**

Punta Cana

Can I tell how much above ground this picture was taken?

Can I tell how much above ground this picture was taken?

Same distance as where the horizon intersects a building

Same distance as where the horizon intersects a building: 50 floors up

- This is only true when the camera (image plane) is orthogonal to the ground plane. And the ground plane is flat.
- A very nice explanation of this phenomena can be find by Derek Hoiem here: https://courses.engr.illinois.edu/cs543/sp2011/materials/3dscene__book_svg.pdf

Depth from Stereo

SUBRAHMANYAM MURALA
CVPR Lab
School of Computer Science and Statistics
Trinity College Dublin, Ireland

Source: Sanja Fidler, "CSC420: Intro to Image Understanding Introduction," University of Toronto (Lectures).

Depth from Two Views: Stereo

All points on the projective line to **P** map to **p**

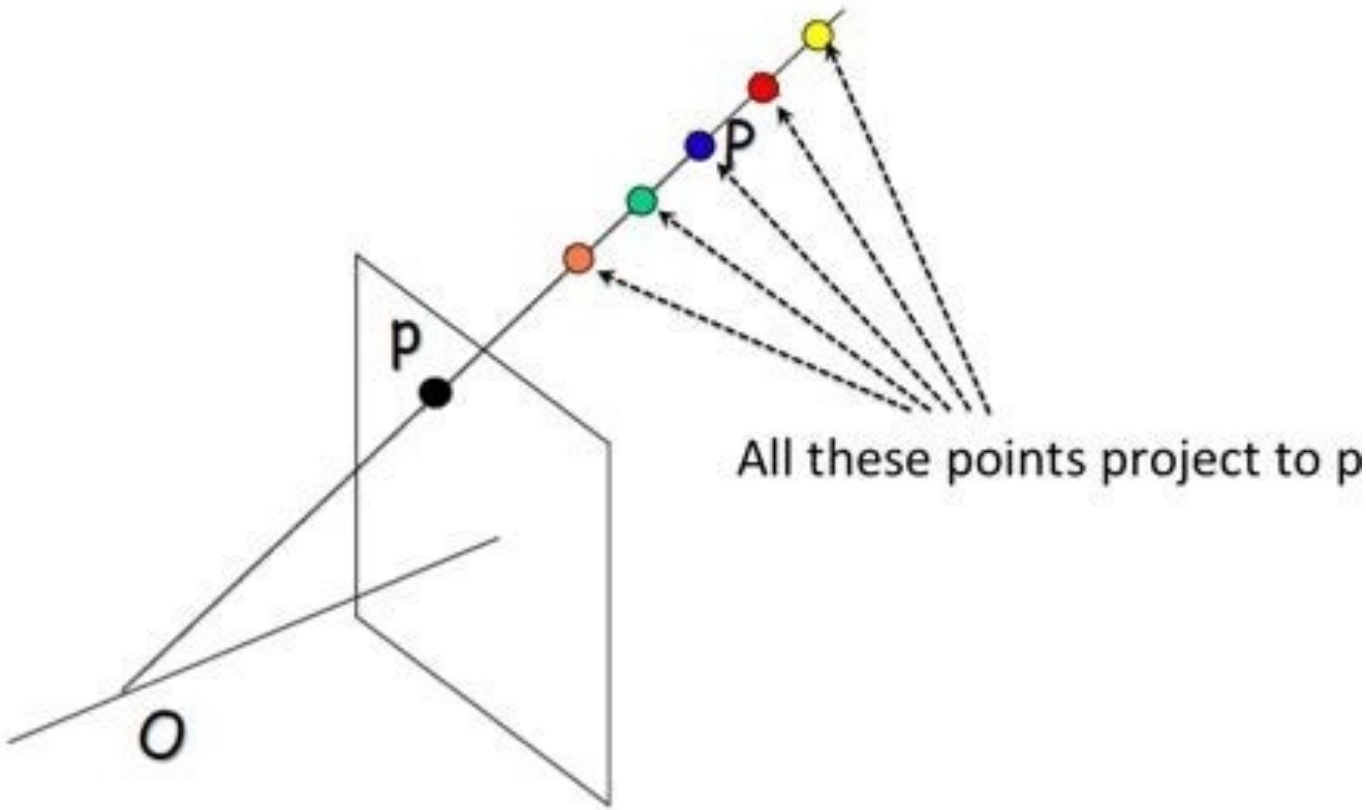


Figure: One camera

All points on projective line to **P** in left camera map to a **line** in the image plane of the right camera

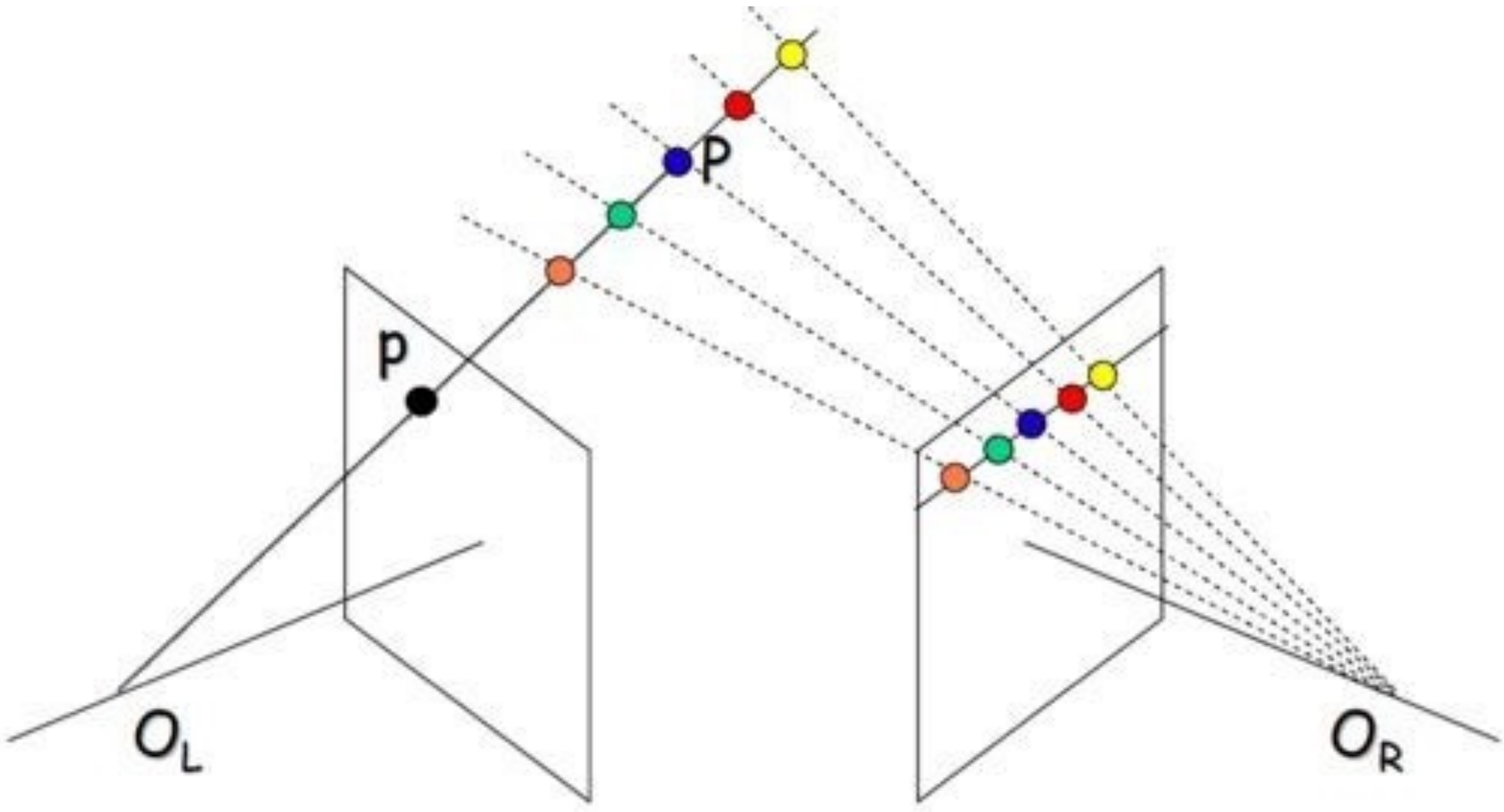


Figure: Add another camera

If I search this line to find correspondences...

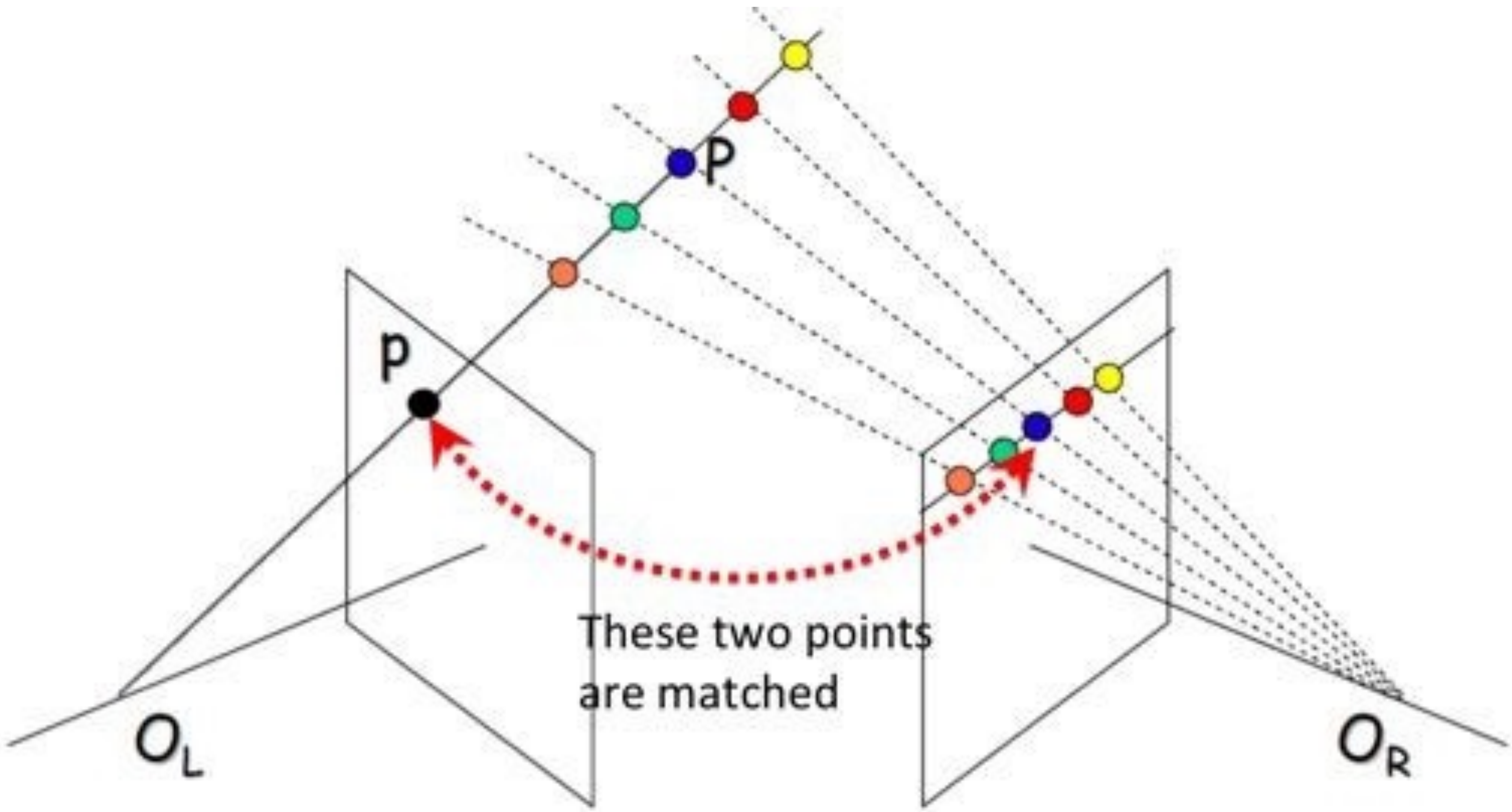


Figure: If I am able to find corresponding points in two images...

I can get 3D!

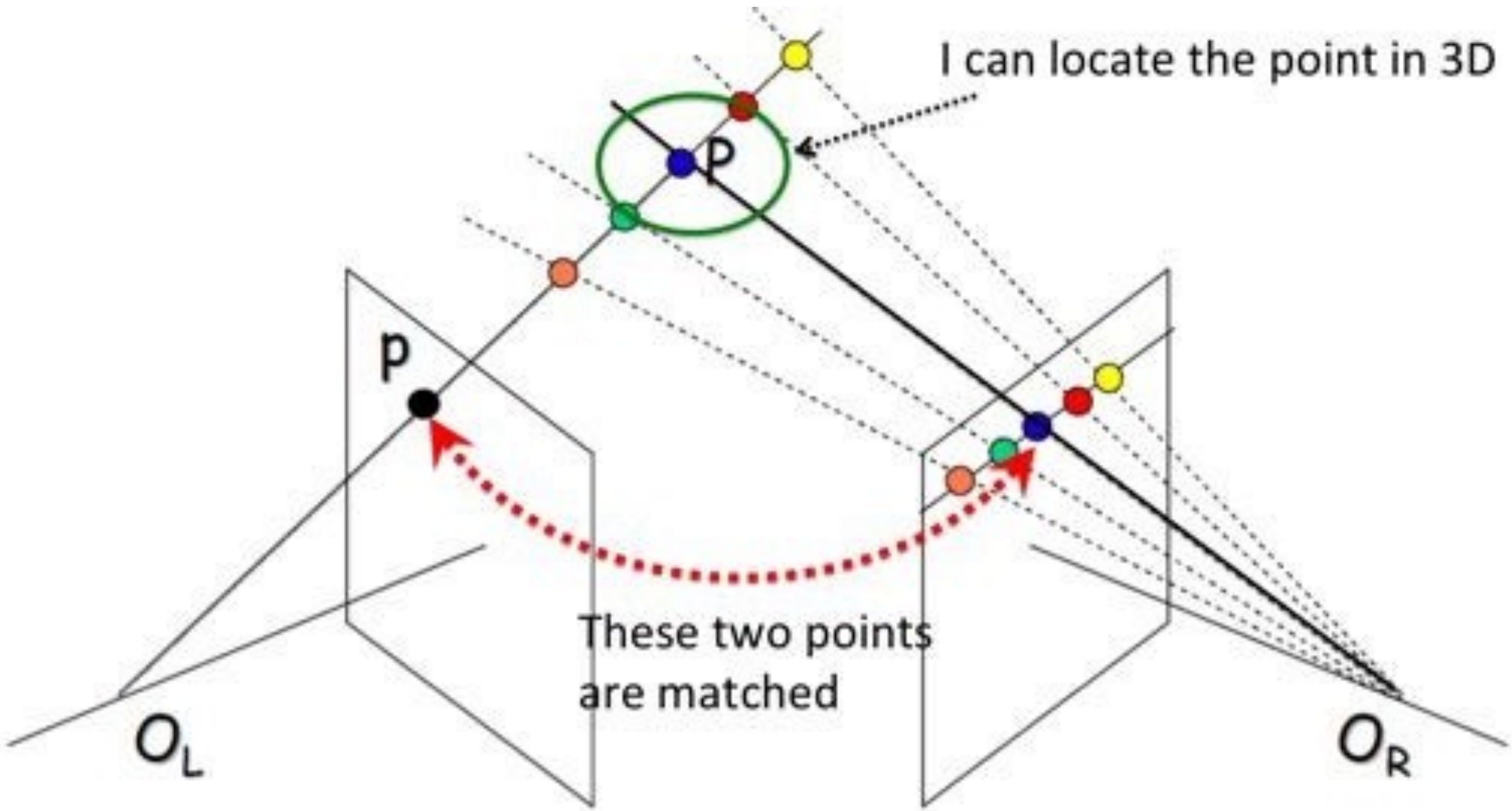


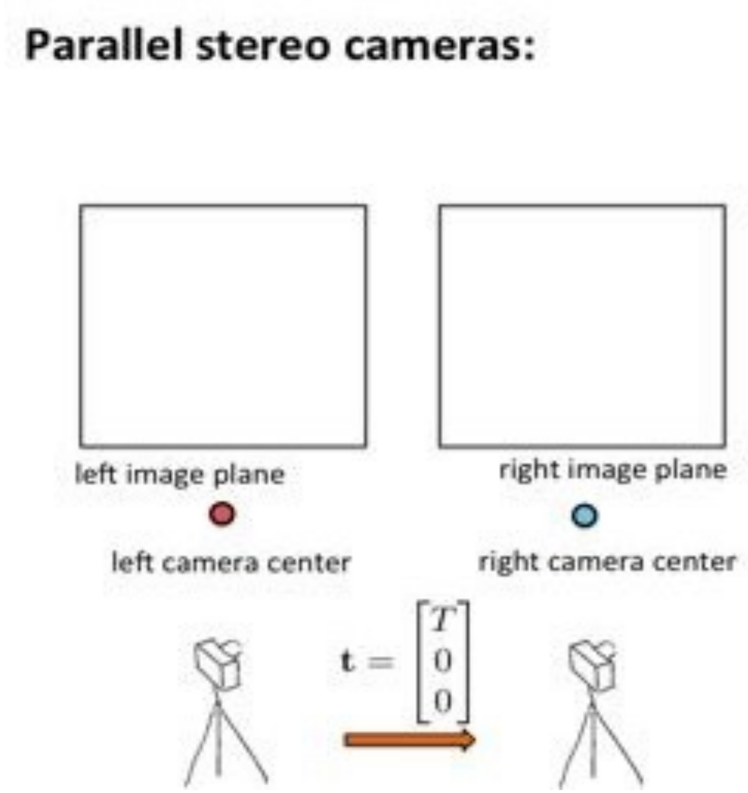
Figure: I can get a point in 3D by triangulation!

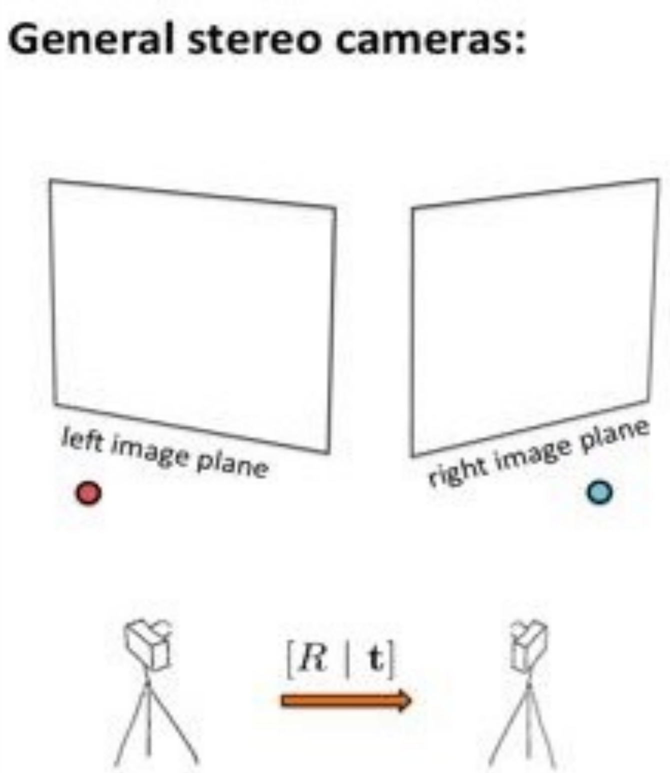
Stereo

Epipolar geometry

Case with two cameras with parallel optical axes

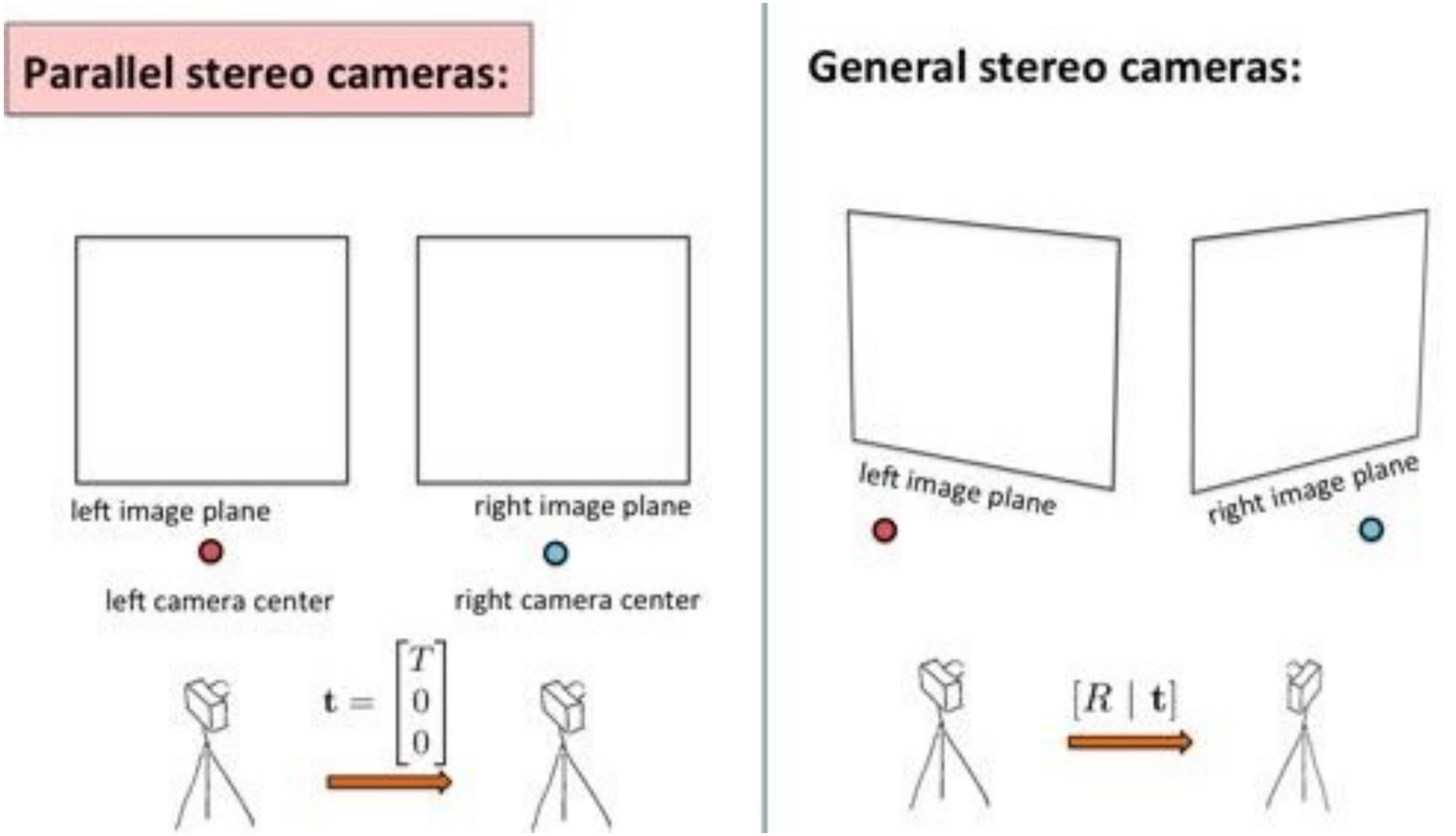
General case





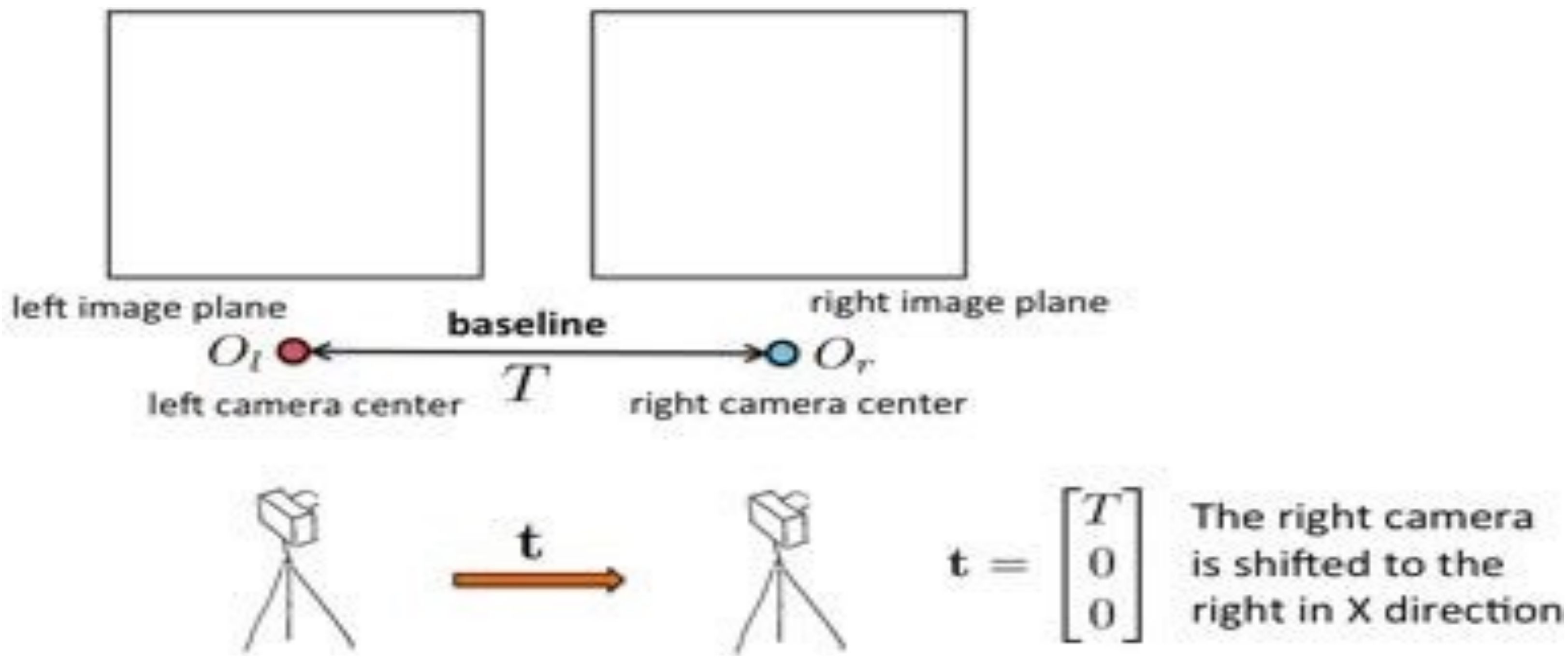
Epipolar geometry

- Case with two cameras with parallel optical axes ← **First this**
- General case

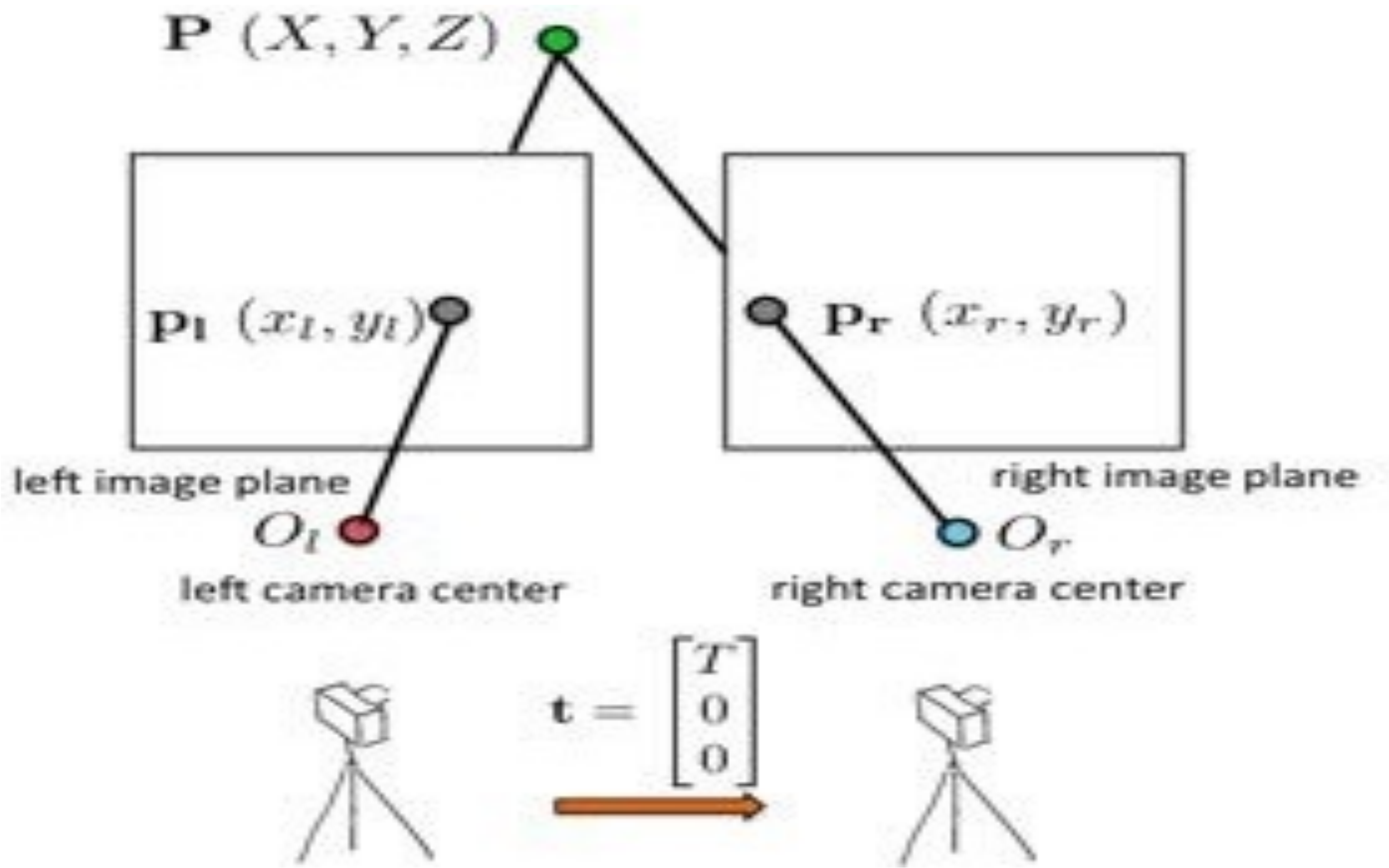


Stereo: Parallel Calibrated Cameras

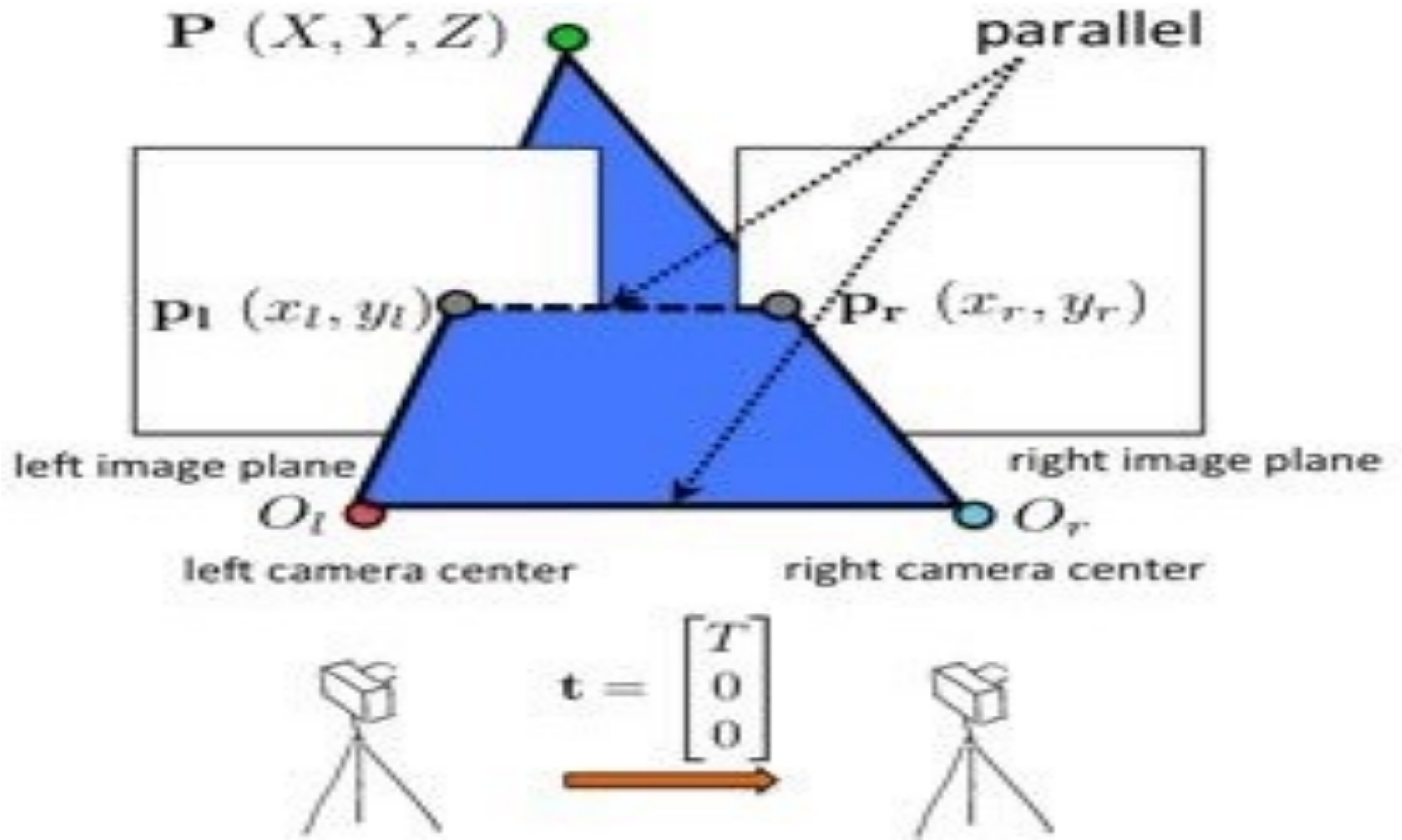
We assume that the two calibrated cameras (we know intrinsics and extrinsics) are parallel, i.e. the right camera is just some distance to the right of left camera. We assume we know this distance. We call it the **baseline**.



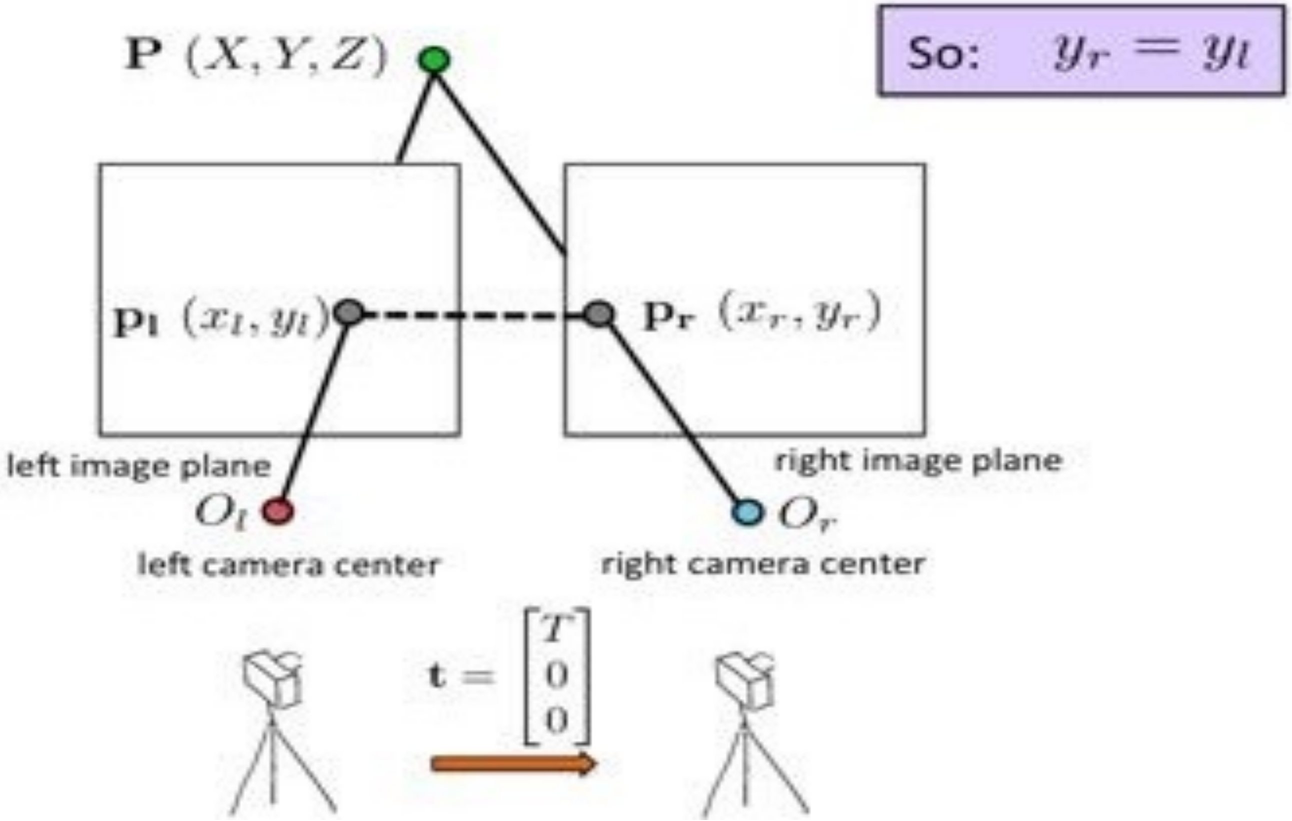
Pick a point *P* in the world



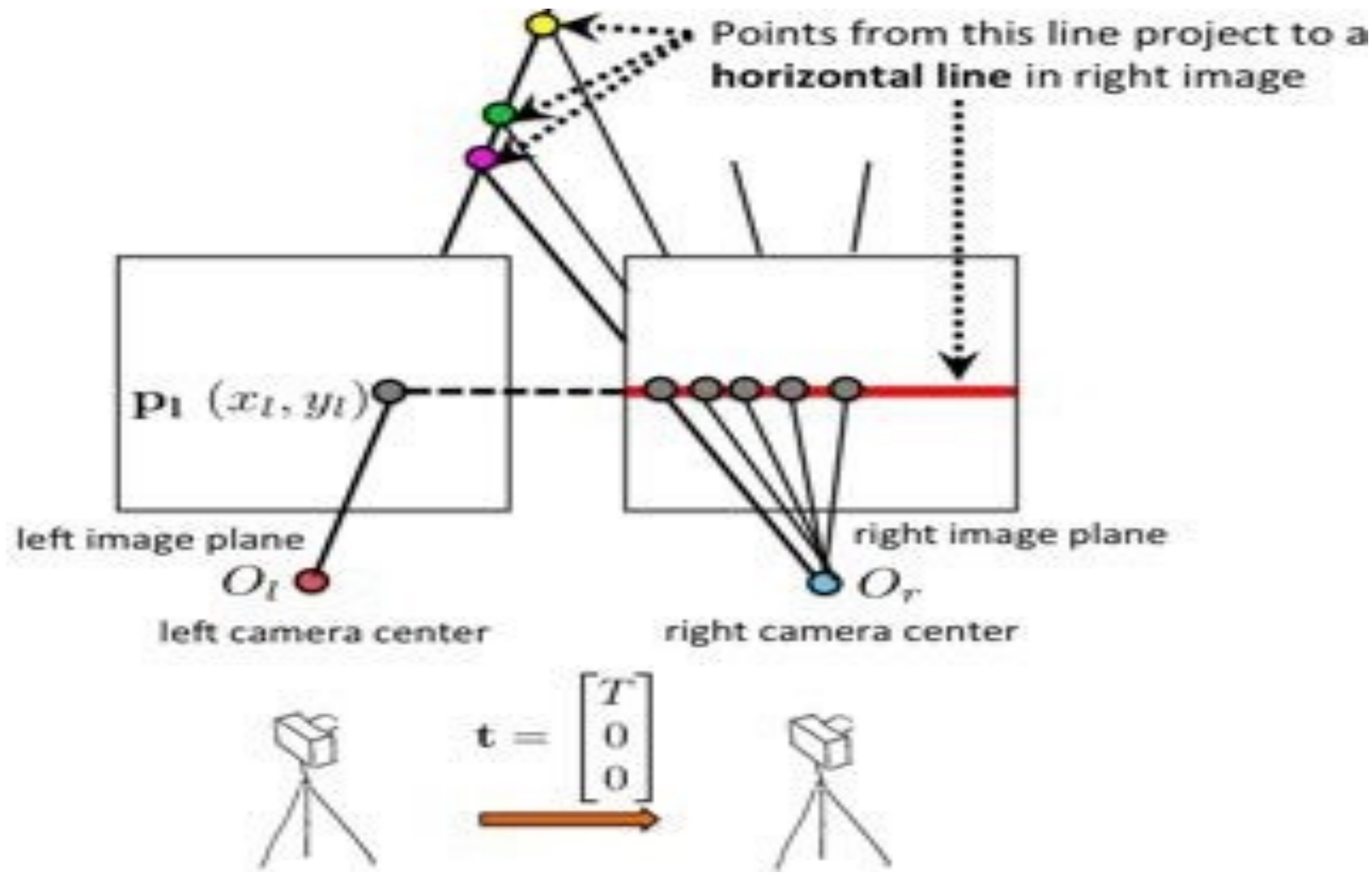
Points $\mathbf{O_l}$, $\mathbf{O_r}$ and **P** (and $\mathbf{p_l}$ and $\mathbf{p_r}$) lie on a plane. Since two image planes lie on the same plane (distance *f* from each camera), the lines $\mathbf{O_lO_r}$ and $\mathbf{p_lp_r}$ are parallel.



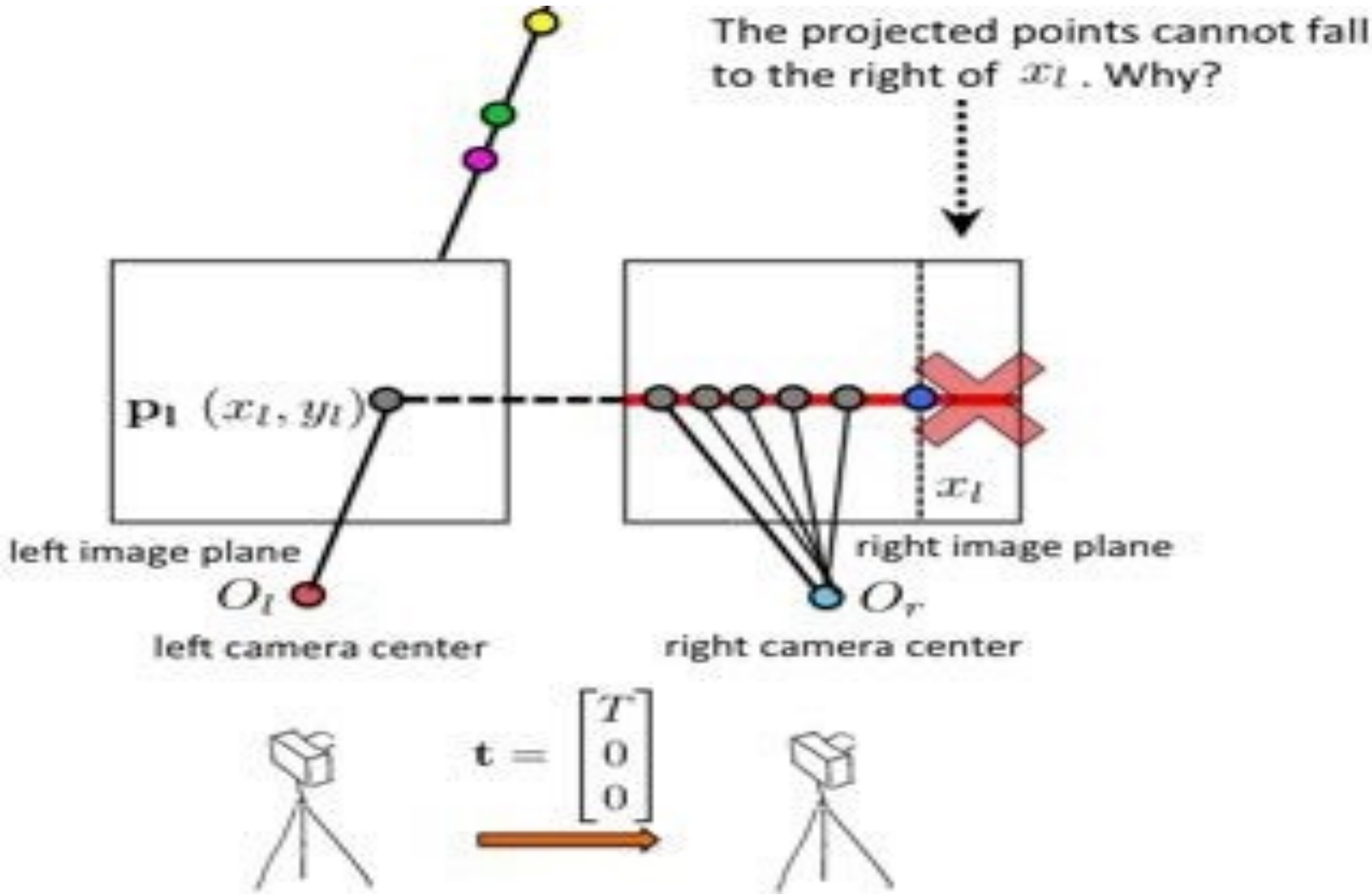
Since lines $\mathbf{O_l O_r}$ and $\mathbf{p_l p_r}$ are parallel, and $\mathbf{O_l}$ and $\mathbf{O_r}$ have the same y , then also $\mathbf{p_l}$ and $\mathbf{p_r}$ have the same y : $y_r = y_l$!



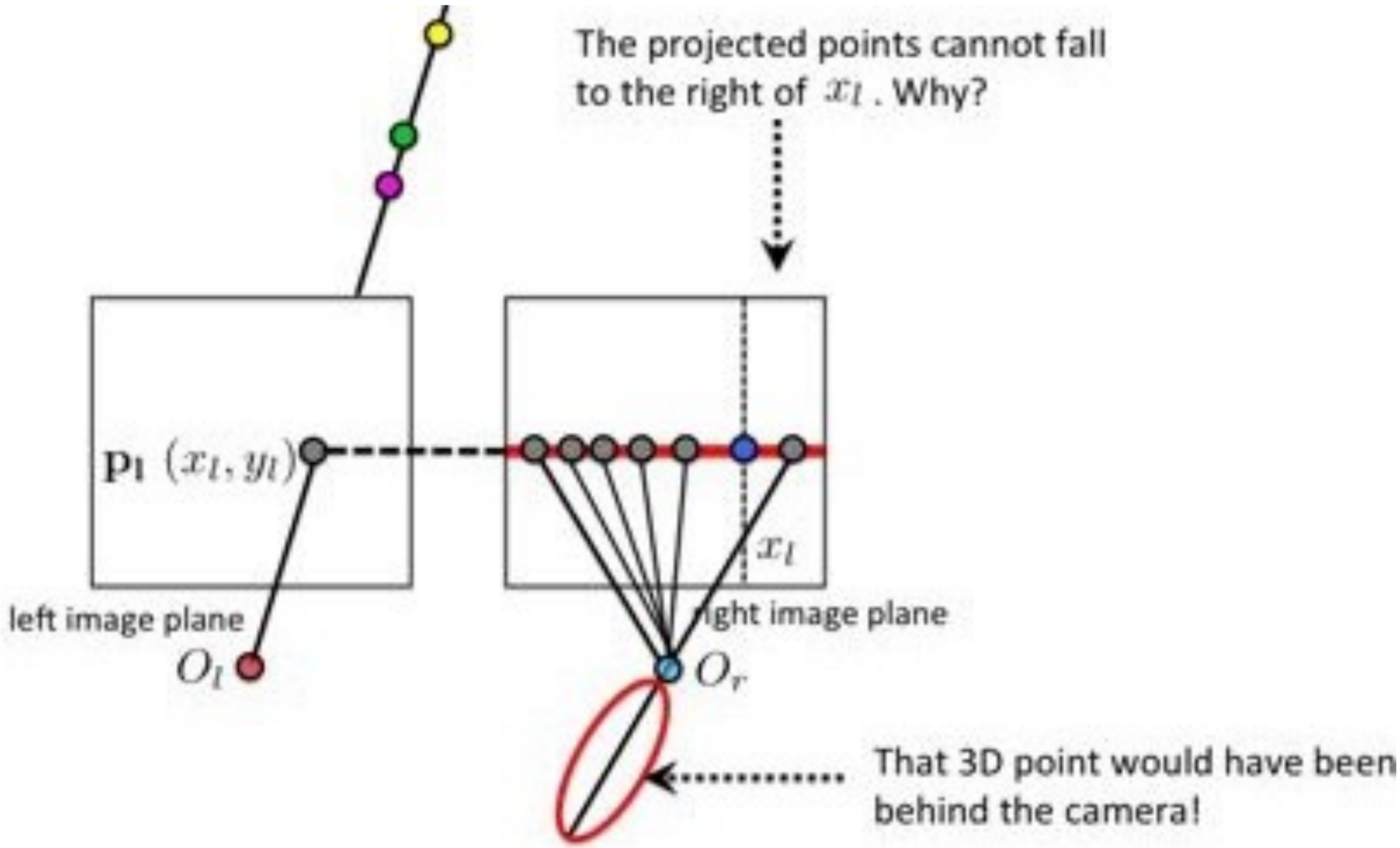
So all points on the projective line $\mathbf{O_l p_l}$ project to a horizontal line with $y = y_l$ on the right image. This is nice, let's rememberthis.



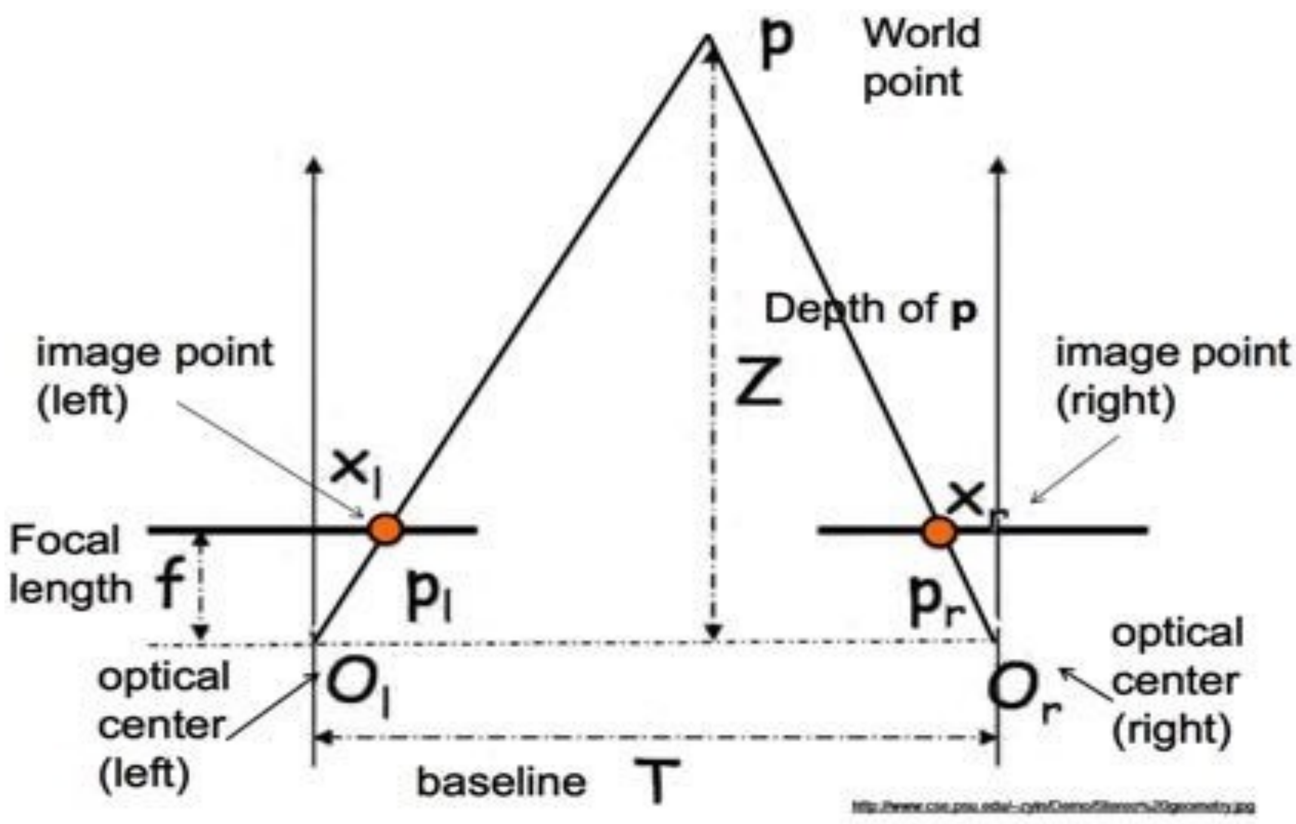
Another observation: No point from **O$_l$p$_l$** can project to the right of x_l in the right image. **Why**?



Because that would mean our image can see behind the camera...

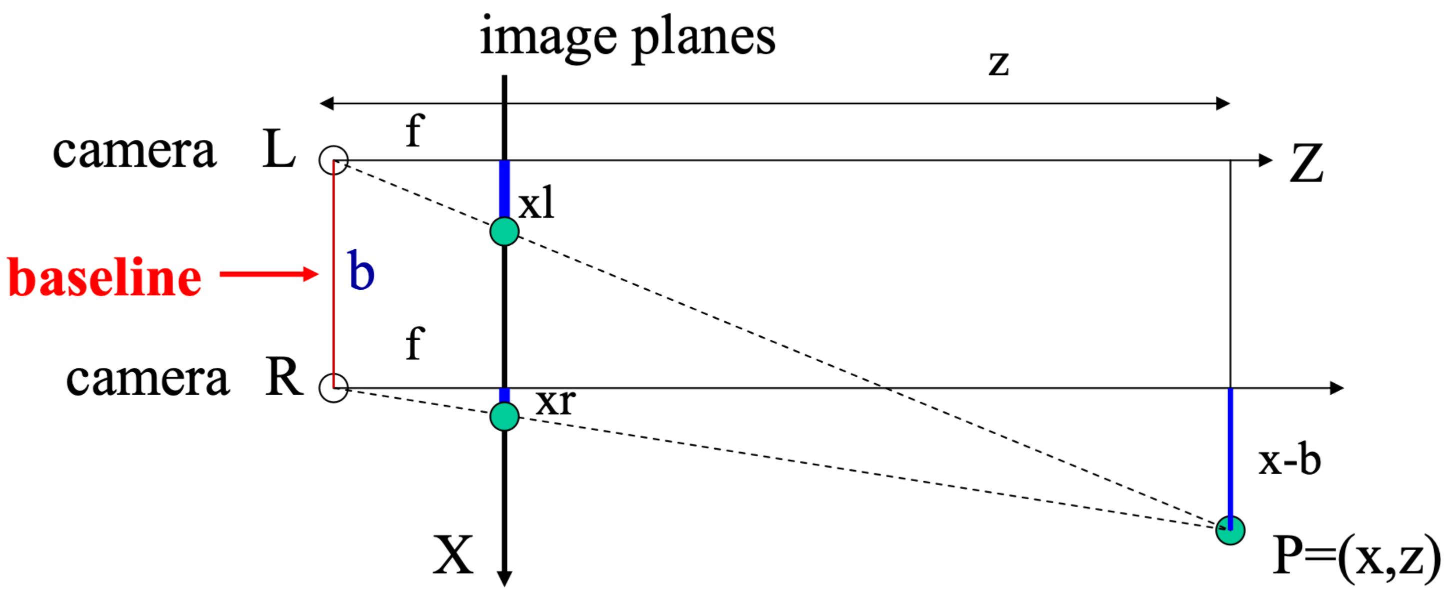


Since our points $\mathbf{p_l}$ and $\mathbf{p_r}$ lie on a horizontal line, we can forget about y_l for a moment (it doesn't seem important). Let's look at the camera situation from the birdseye perspective instead. Let's see if we can find a connection between x_l , x_r and Z (because Z is what we want).



[Adopted from: J. Hays]

We can then use similar triangles to compute the depth of the point *P*



$$\frac{z}{f} = \frac{x}{xl}$$

$$\frac{z}{f} = \frac{x-b}{xr}$$

$$\frac{z}{f} = \frac{y}{yl} = \frac{y}{yr}$$

Y-axis is perpendicular to the page.

(from similar triangles)

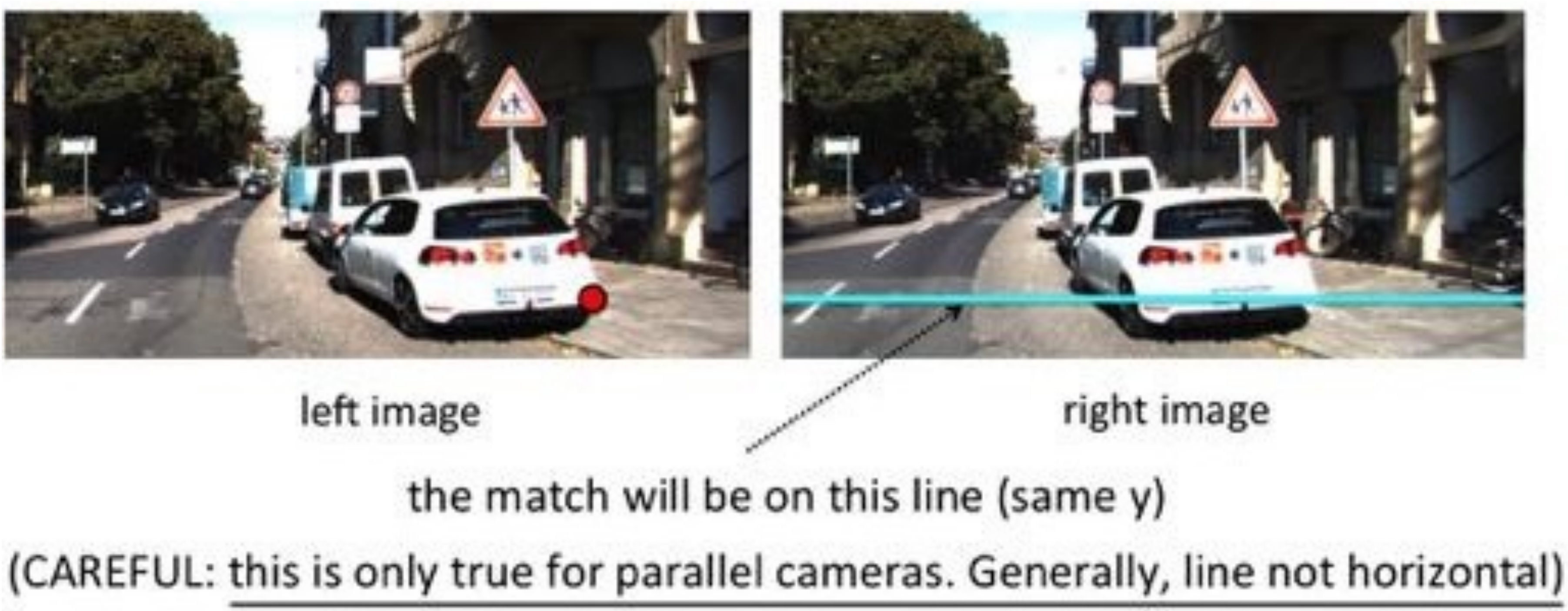
[Adopted from: J. Hays]

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$?

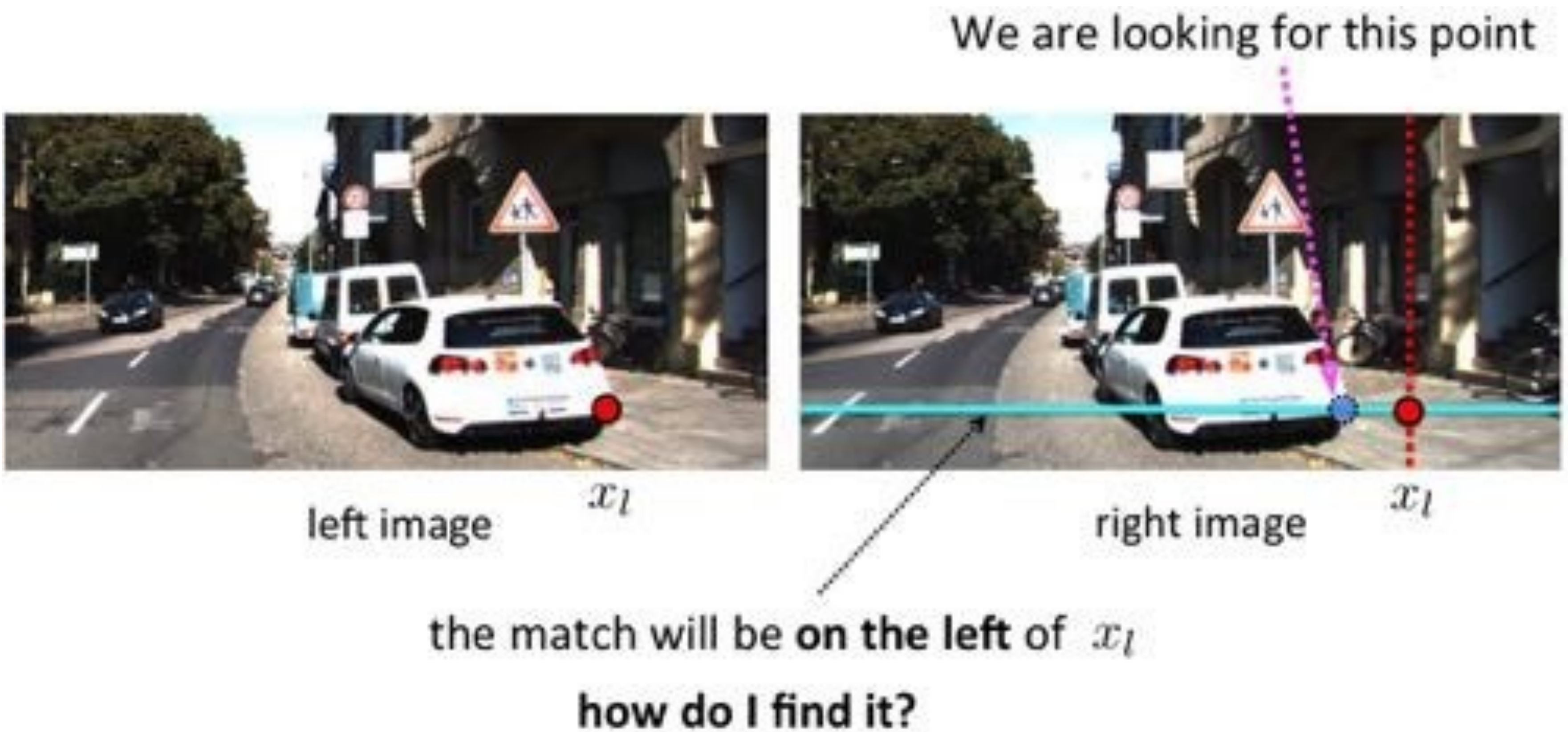
left image

right image

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching on line $y_r = y_l$.

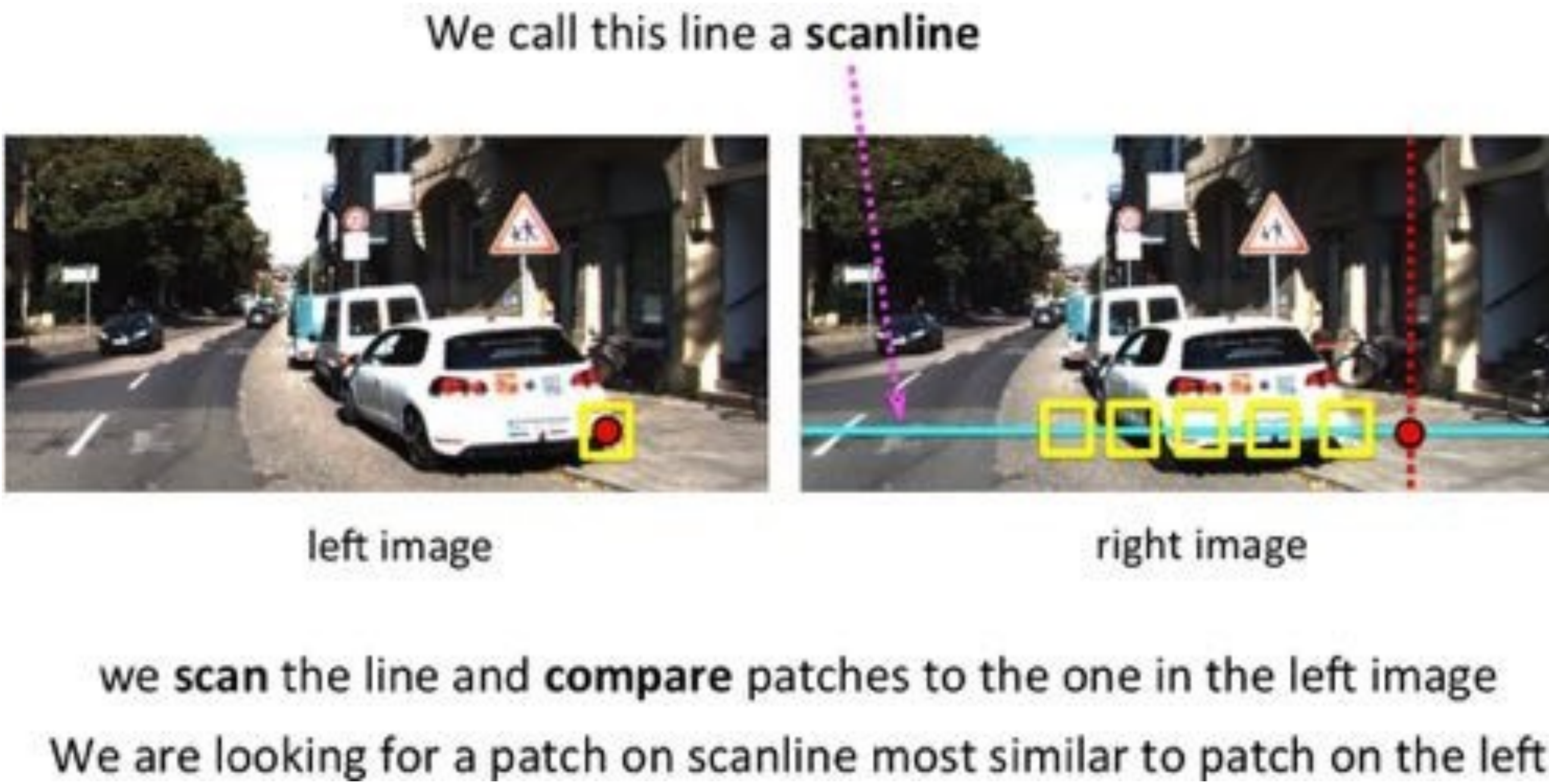


For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching on line $y_r = y_l$.

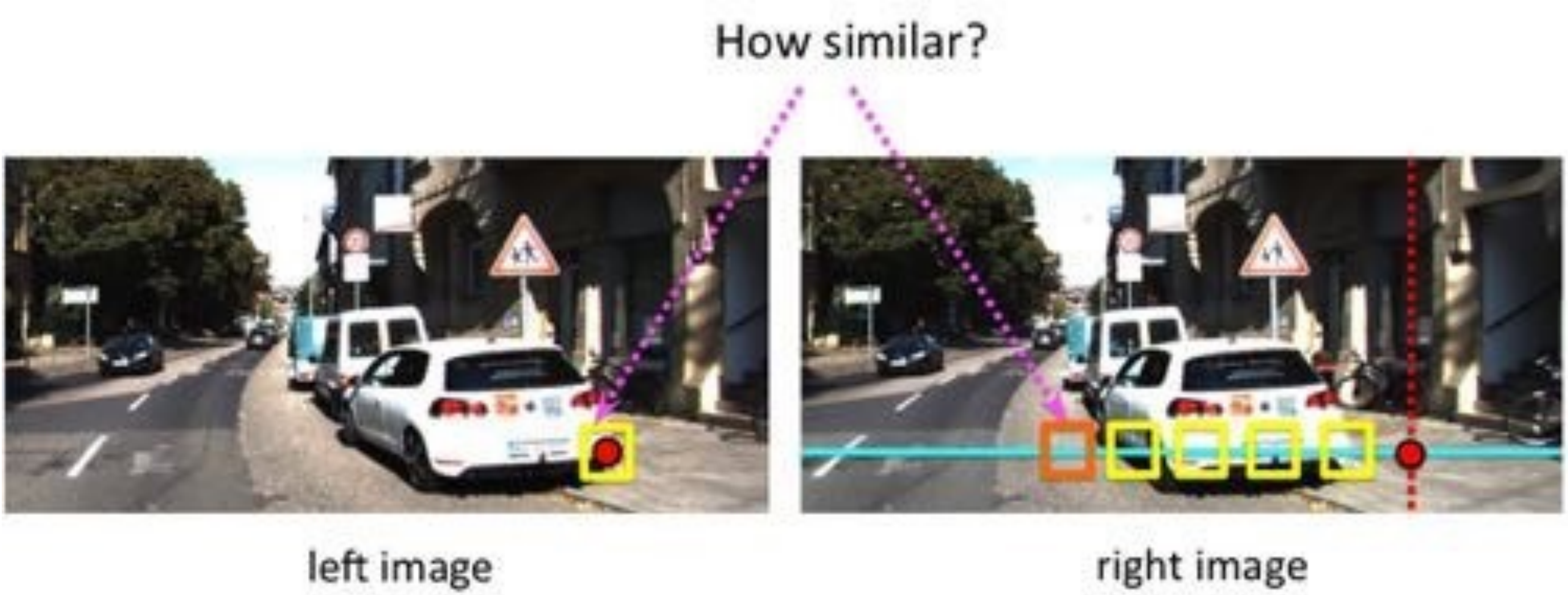


Depth from Stereo

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).



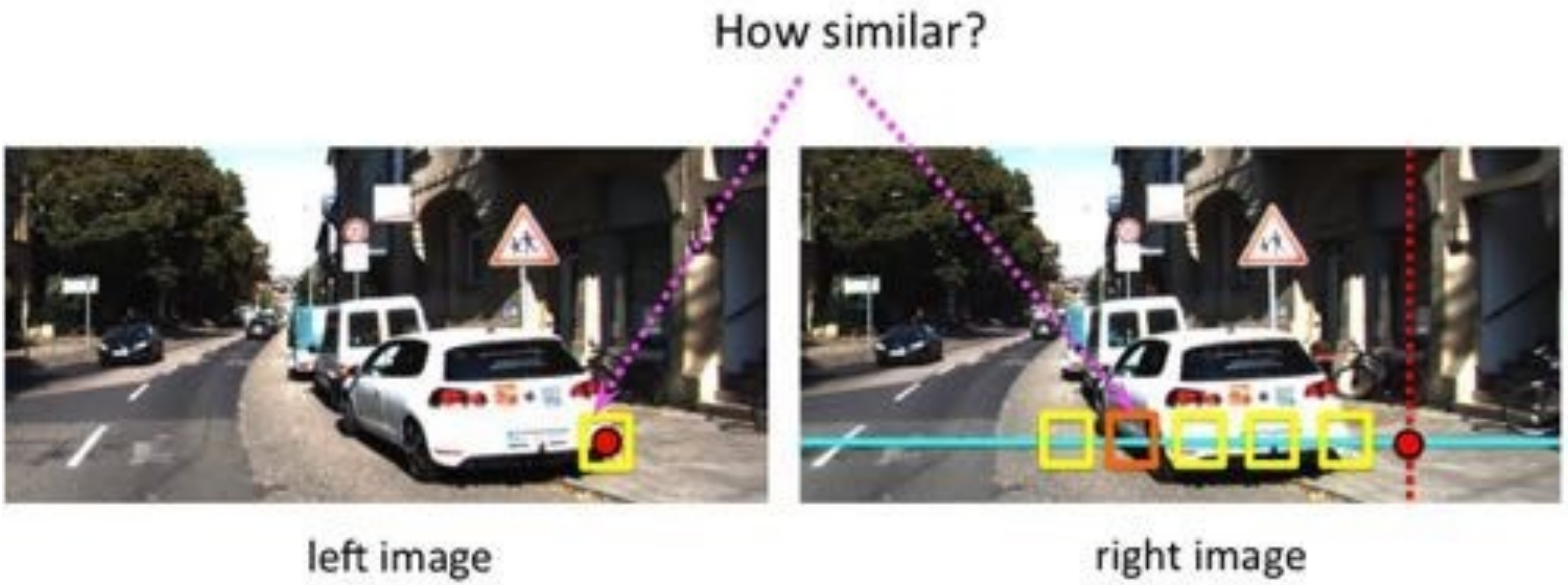
For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).



we **scan** the line and **compare** patches to the one in the left image

We are looking for a patch on scanline most similar to patch on the left

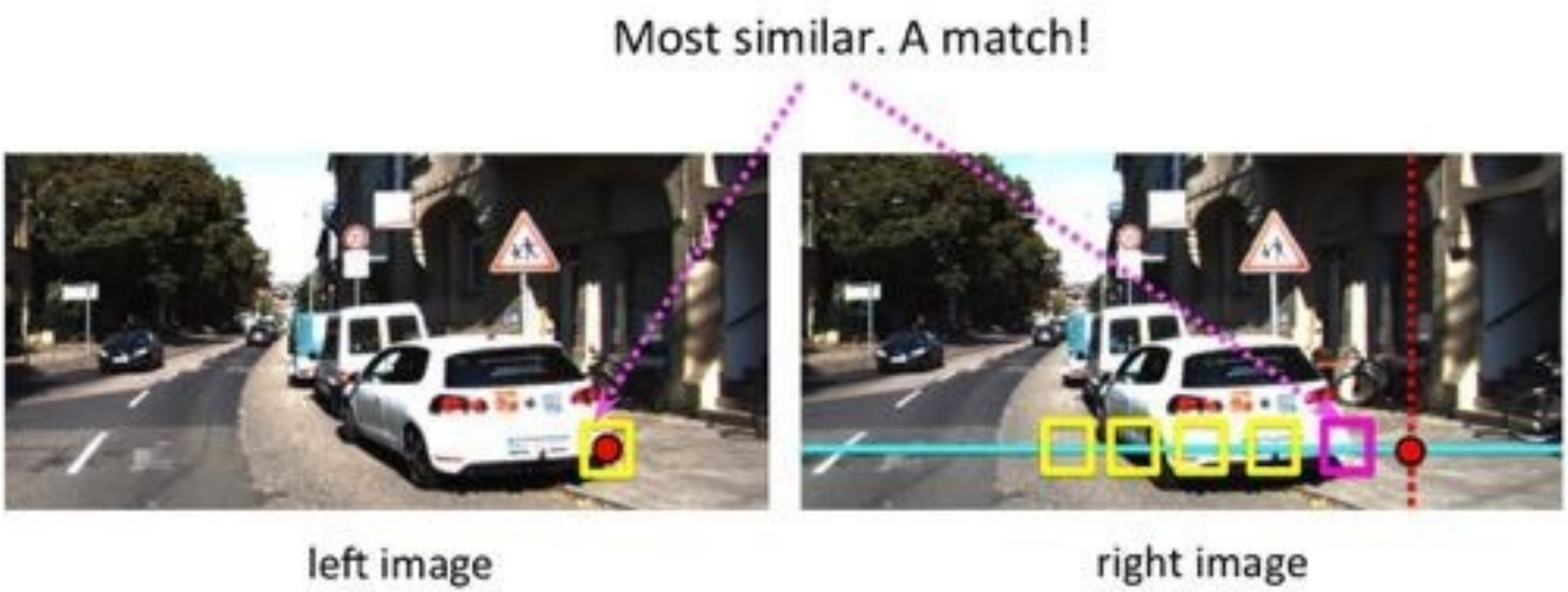
For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).



we **scan** the line and **compare** patches to the one in the left image

We are looking for a patch on scanline most similar to patch on the left

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).



we **scan** the line and **compare** patches to the one in the left image

We are looking for a patch on scanline most similar to patch on the left

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).

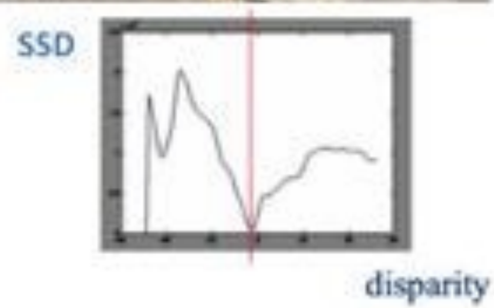
At each point on the scanline: Compute a **matching cost**

Matching cost: **SSD** or **normalized correlation**

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).

$$SSD(\text{patch}_l, \text{patch}_r) = \sum_x \sum_y (I_{\text{patch}_l}(x, y) - I_{\text{patch}_r}(x, y))^2$$

left image



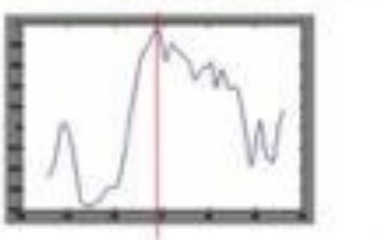
Compute a matching cost

Matching cost: **SSD** **(look for minima)**

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).

$$NC(\text{patch}_l, \text{patch}_r) = \frac{\sum_x \sum_y (I_{\text{patch}_l}(x, y) \cdot I_{\text{patch}_r}(x, y))}{||I_{\text{patch}_l}|| \cdot ||I_{\text{patch}_r}||}$$

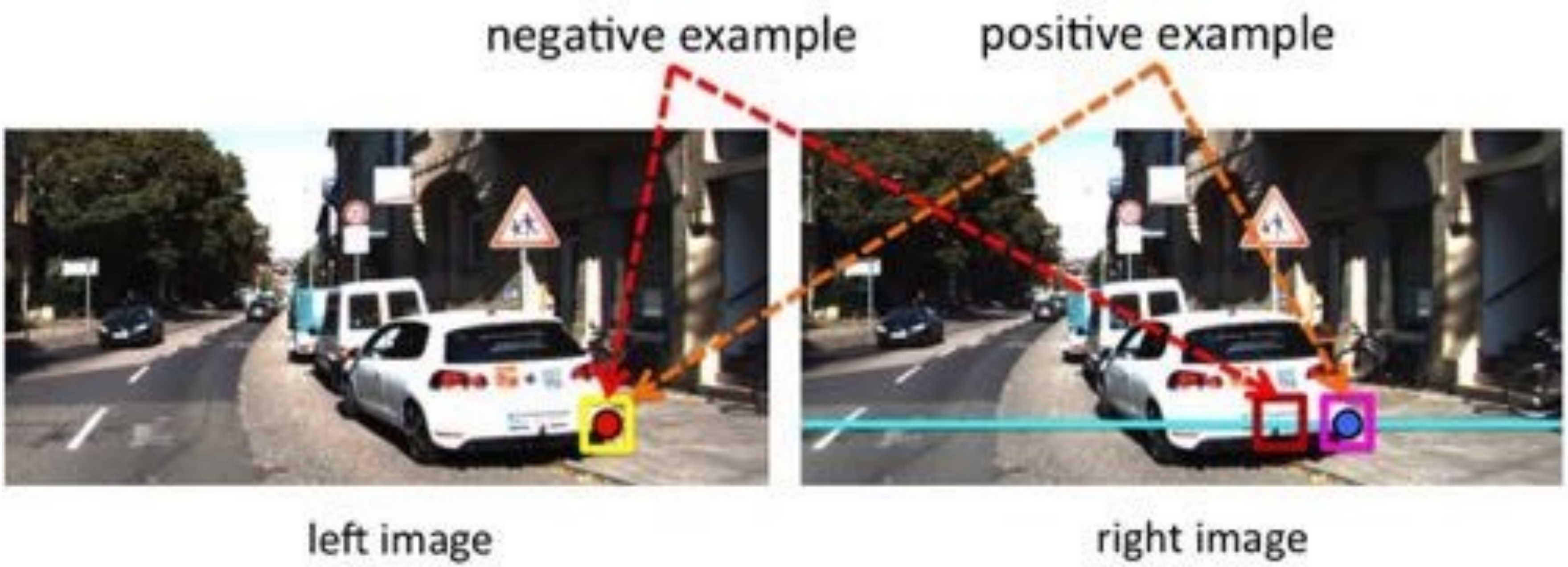
left image



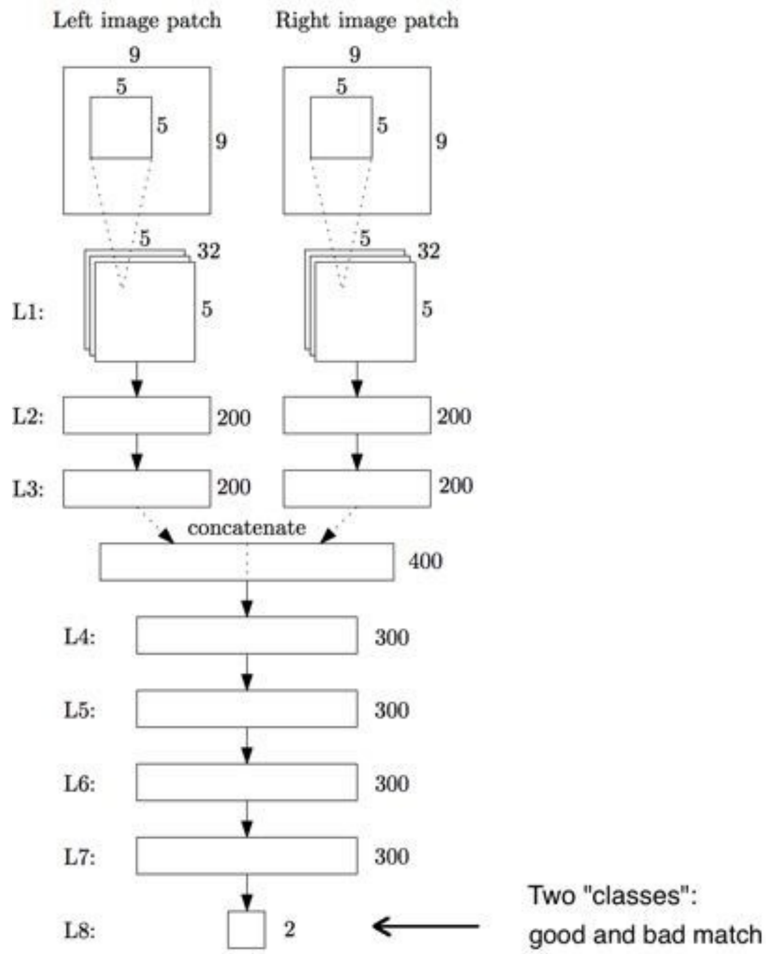
Compute a matching cost

Matching cost: **Normalized Corr. (look for maxima)**

Version'2015: Train a classifier! How can I get ground-truth?



Training examples: get positive and negative matches



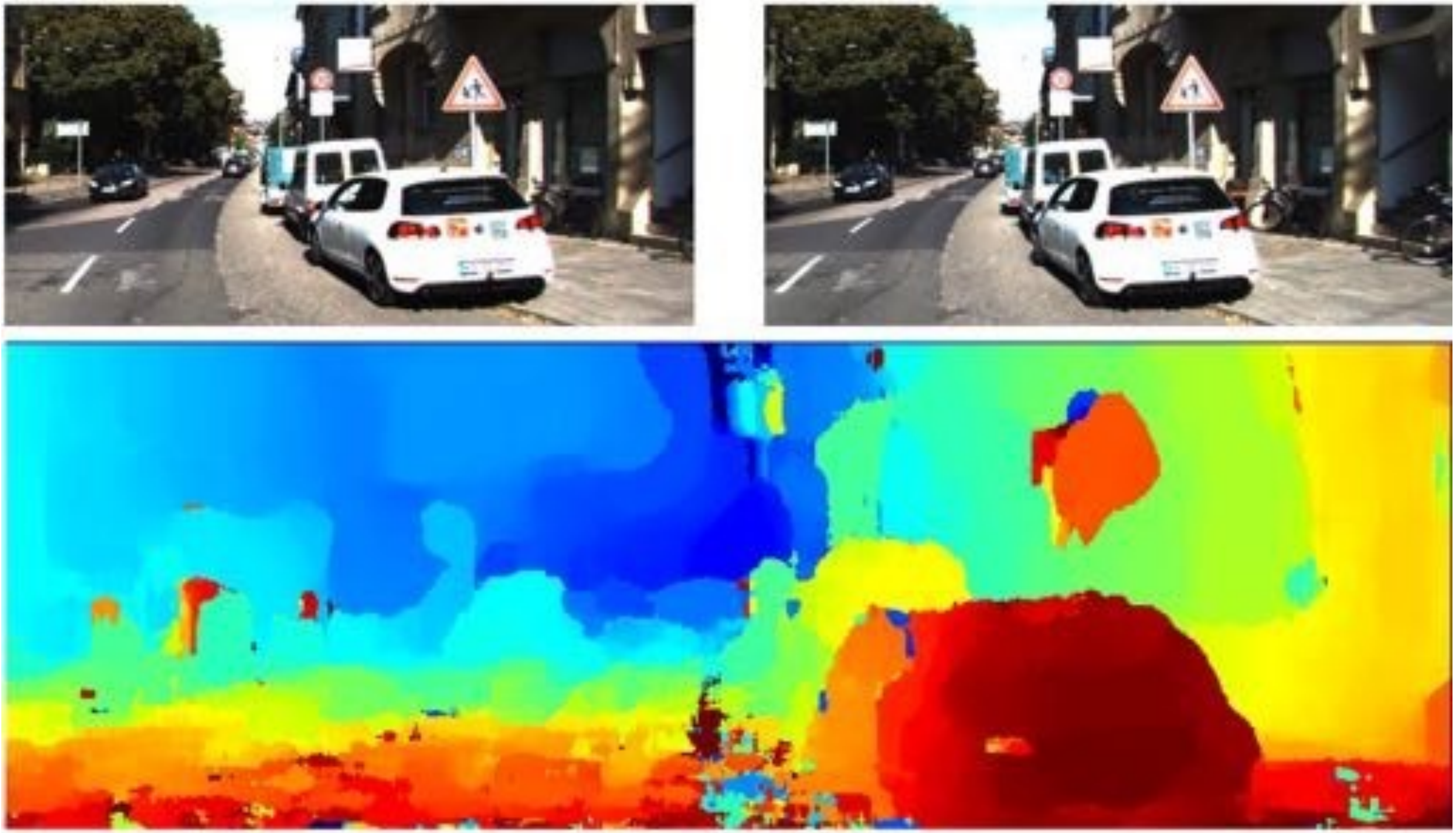
[J. Zbontar and Y. LeCun: Computing the Stereo Matching Cost with a Convolutional Neural Network. CVPR'15]

For each point $\mathbf{p_l} = (x_l, y_l)$, how do I get $\mathbf{p_r} = (x_r, y_r)$? By matching. Patch around (x_r, y_r)) should look similar to the patch around (x_l, y_l).

left image

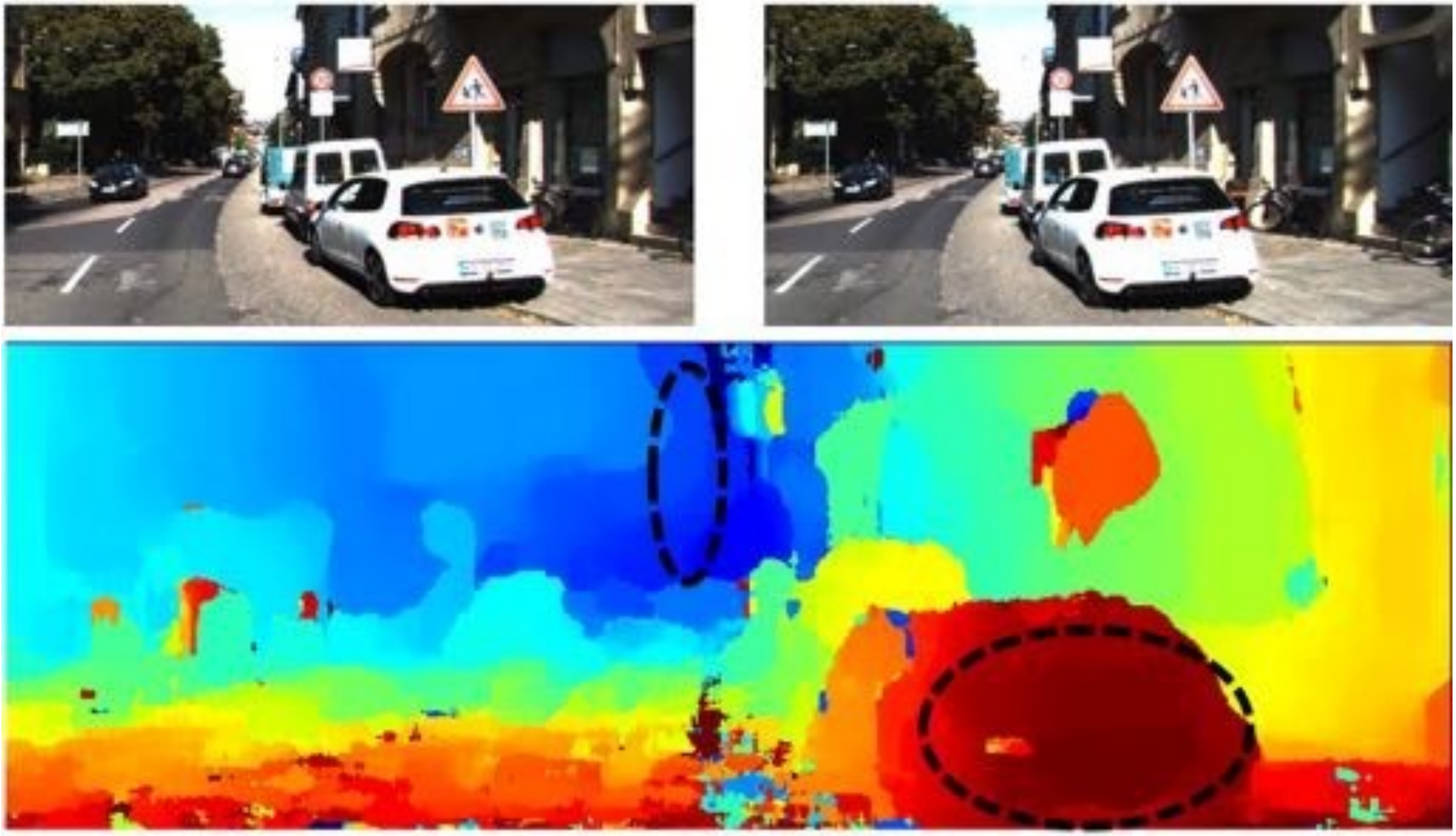
Do this for all the points in the left image!

We get a disparity map as a result



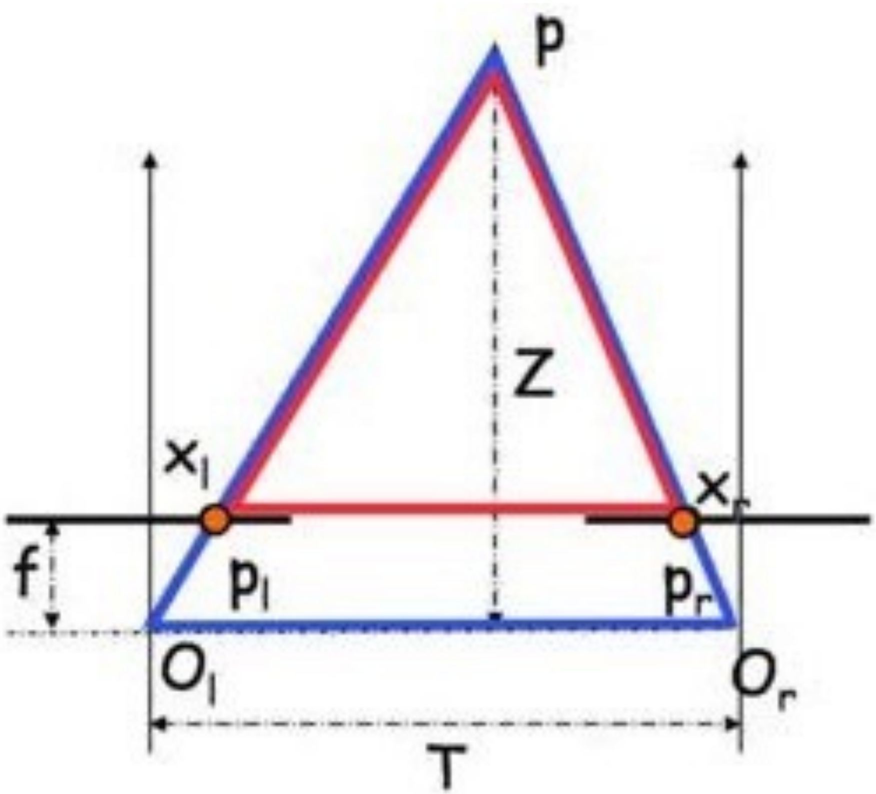
Result: **Disparity map**
(red values large disp., blue small disp.)

We get a disparity map as a result



Things that are closer have **larger disparity** than those that are far away from camera. Why?

Depth and disparity are inversely proportional



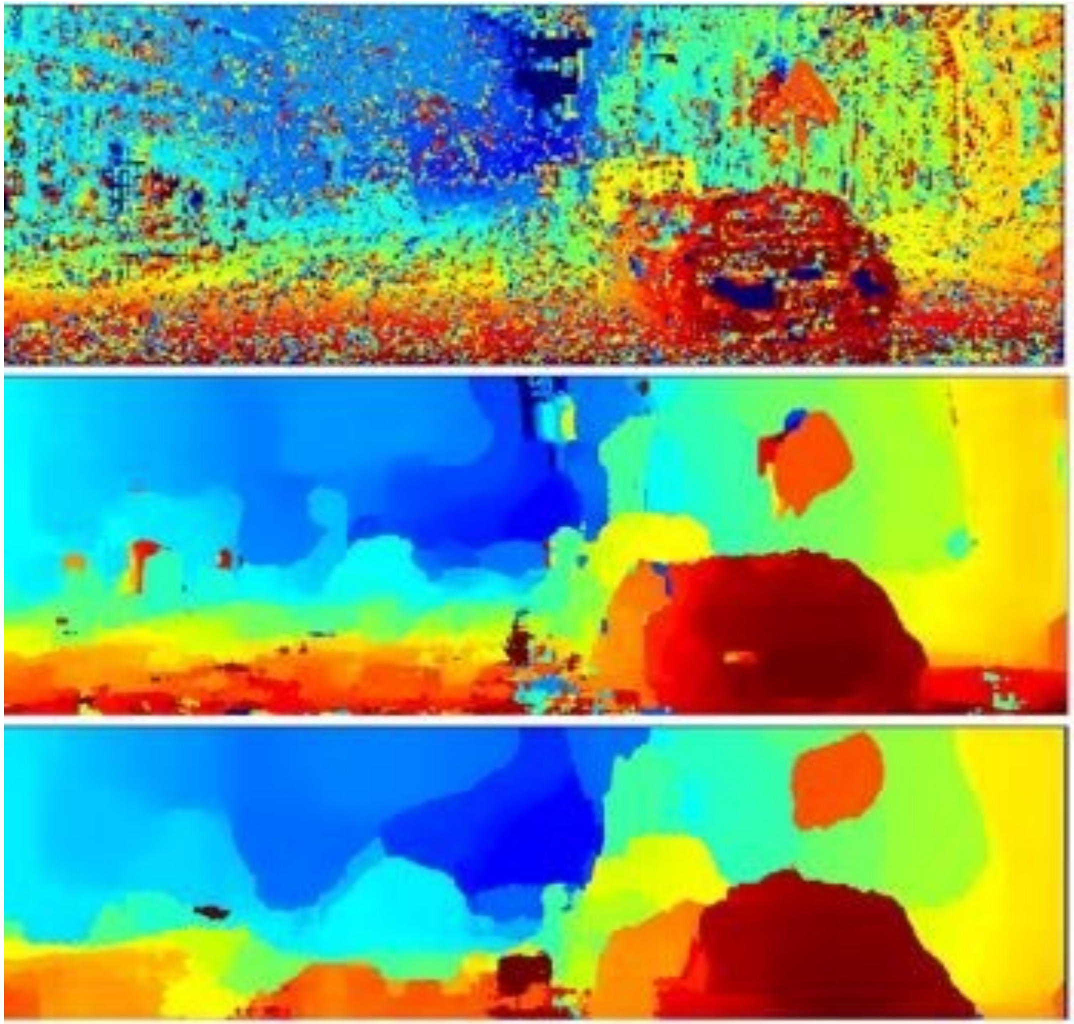
Similar triangles:

$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

$$\boxed{Z = \frac{f \cdot T}{\boxed{x_r - x_l}}}$$

Depth (Z) and disparity are **inversely proportional**

Smaller patches: more detail, but noisy. Bigger: less detail, but smooth



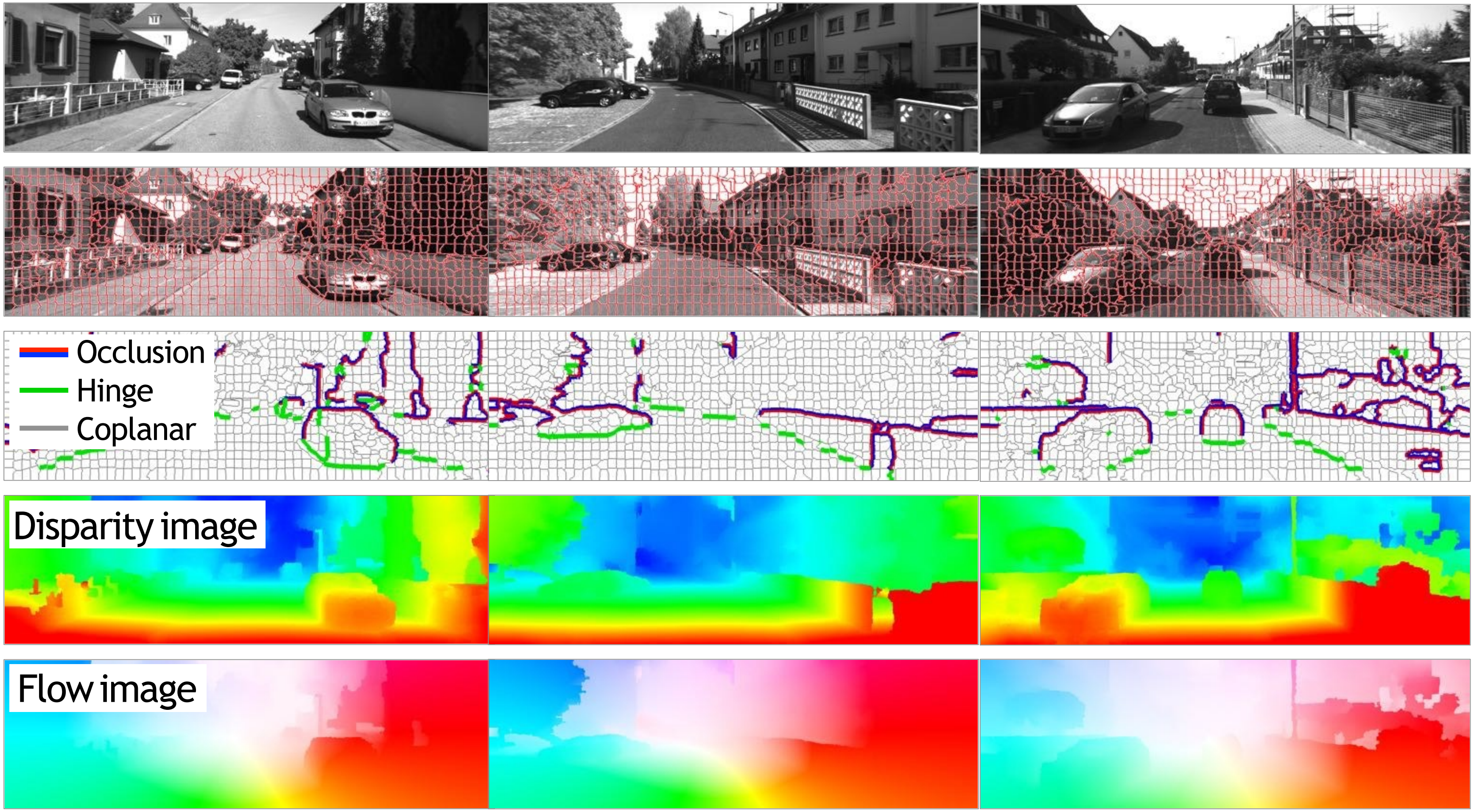
patch size = 5

patch size = 35

patch size = 85

Depth from Stereo

[K. Yamaguchi, D. McAllester and R. Urtasun, ECCV 2014]



Stereo

Epipolar geometry

- Case with two cameras with parallel optical axes
- General case **Next time**