

Homography (Part II) + Stereo (Parallel Cameras)

Homography

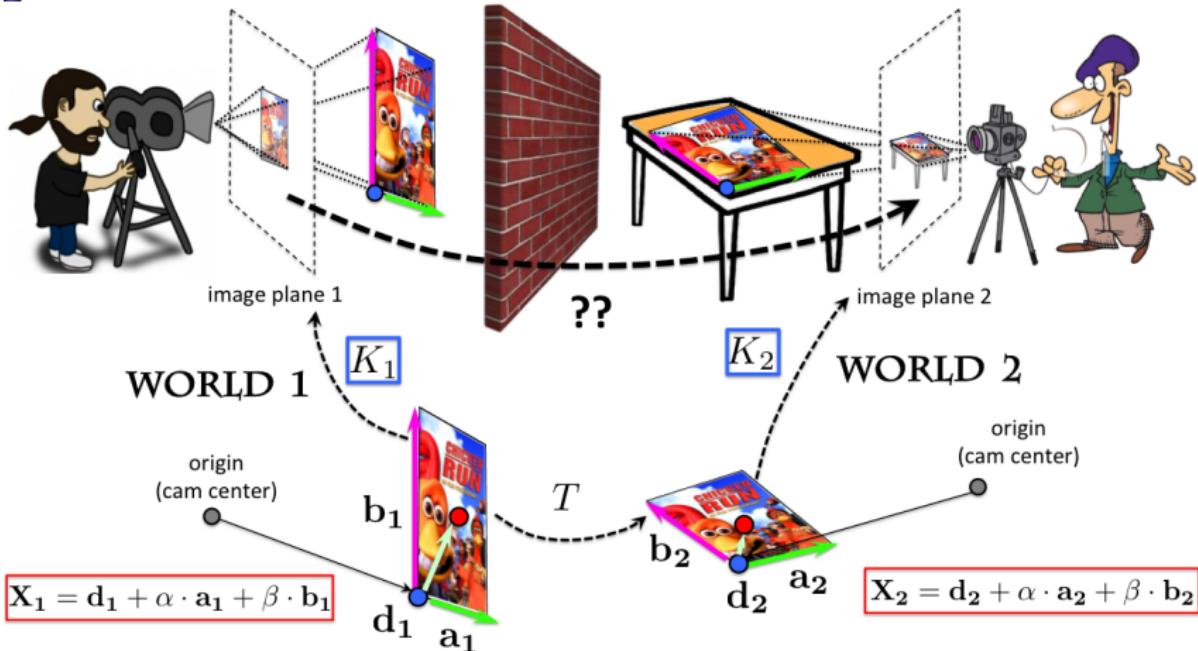


Figure: Given this, the question is what's the transformation that maps the DVD on the first image to the DVD in the second image?

Homography

- Each point on a plane can be written as: $X = d + \alpha \cdot a + \beta \cdot b$, where d is a point, and a and b are two independent directions on the plane.
- Let's have two different planes in 3D:

$$\text{First plane : } X_1 = d_1 + \alpha \cdot a_1 + \beta \cdot b_1$$

$$\text{Second plane : } X_2 = d_2 + \alpha \cdot a_2 + \beta \cdot b_2$$

Via α and β , the two points X_1 and X_2 are in the same location relative to each plane.

- We can rewrite this using homogeneous coordinates:

$$\text{First plane : } X_1 = [a_1 \quad b_1 \quad d_1] \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} = A_1 \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix}$$

$$\text{Second plane : } X_2 = [a_2 \quad b_2 \quad d_2] \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} = A_2 \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix}$$

- Careful: $A_1 = [a_1 \quad b_1 \quad d_1]$ and $A_2 = [a_2 \quad b_2 \quad d_2]$ are 3×3 matrices.

Homography

- In 3D, a transformation between the planes is given by:

$$X_2 = TX_1$$

There is one transformation T between every pair of points X_1 and X_2 .

- Expand it:

$$A_2 \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} = T A_1 \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} \quad \text{for every } \alpha, \beta$$

- Then it follows: $T = A_2 A_1^{-1}$, with T a 3×3 matrix.
- Let's look at what happens in projective (image) plane. Note that we have each plane in a separate image and the two images may not have the same camera intrinsic parameters. Denote them with K_1 and K_2 .

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

Homography

- From previous slide:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

- Insert $X_2 = T X_1$ into equality on the right:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 T X_1$$

- And finally divide through by w_1 :

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Homography

- From previous slide:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

- Insert $X_2 = T X_1$ into equality on the right:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 T X_1 = K_2 T (K_1^{-1} K_1) X_1$$

- And finally divide through by w_1 :

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Homography

- From previous slide:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

- Insert $X_2 = T X_1$ into equality on the right:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 T X_1 = K_2 T (\underbrace{K_1^{-1} K_1}_{w_1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix})$$

- And finally divide through by w_1 :

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Homography

- From previous slide:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

- Insert $X_2 = T X_1$ into equality on the right:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 T X_1 = K_2 T (K_1^{-1} K_1) X_1 = w_1 K_2 T K_1^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

- And finally divide through by w_1 :

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Homography

- From previous slide:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K_1 X_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 X_2$$

- Insert $X_2 = TX_1$ into equality on the right:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K_2 T X_1 = K_2 T (K_1^{-1} K_1) X_1 = w_1 \underbrace{K_2 T K_1^{-1}}_{3 \times 3 \text{ matrix}} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

- And finally divide through by w_1 :

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Remember Panorama Stitching from Lecture 9?



Take a tripod, rotate camera
and take pictures

- Each pair of images is related by homography. **Why?**

[Source: Fernando Flores-Mangas]

Rotating the Camera

- Rotating my camera with R is the same as rotating the 3D points with R^T (inverse of R):

$$X_2 = R^T X_1$$

where X_1 is a 3D point in the coordinate system of the first camera and X_2 the 3D point in the coordinate system of the rotated camera.

- We can use the same trick as before, where we have $T = R^T$:

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = KX_1 \quad \text{and} \quad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = KX_2$$

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = w_1 \underbrace{K R^T K^{-1}}_{3 \times 3 \text{ matrix}} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

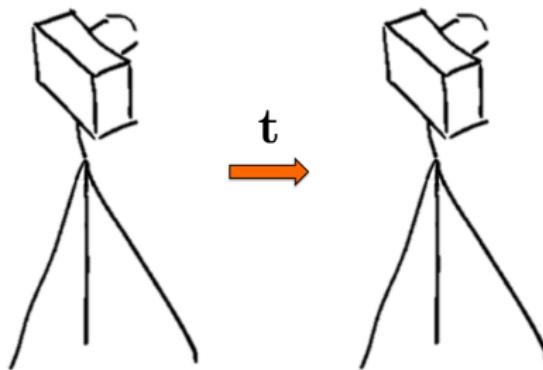
- And this is a homography

What If I Move the Camera?

- So if I take a picture and then rotate the camera and take another picture, the first and second picture are related via homography (assuming the scene didn't change in between)

What If I Move the Camera?

- So if I take a picture and then rotate the camera and take another picture, the first and second picture are related via homography (assuming the scene didn't change in between)
- What if I **move** my camera?



What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K X_2$$

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K X_2 = K (X_1 - t)$$

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K X_2 = \underbrace{K(X_1 - t)}_{w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}}$$

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K X_2 = K (X_1 - t) = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K X_2 = K (X_1 - t) = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$

- Hmm... Different values of w_1 give me different points in the second image.
- So even if I have K and t it seems I can't compute where a point from the first image projects to in the second image.

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = KX_2 = K(X_1 - t) = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$

- Hmm... Different values of w_1 give me different points in the second image.
- So even if I have K and t it seems I can't compute where a point from the first image projects to in the second image.
- From

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = KX_1$$

we know that different w_1 mean different points X_1 on the projective line

What If I Move the Camera?

- If I move the camera by t , then: $X_2 = X_1 - t$. Let's try the same trick again:

$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = KX_2 = K(X_1 - t) = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$

- Hmm... Different values of w_1 give me different points in the second image.
- So even if I have K and t it seems I can't compute where a point from the first image projects to in the second image.
- From

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = KX_1$$

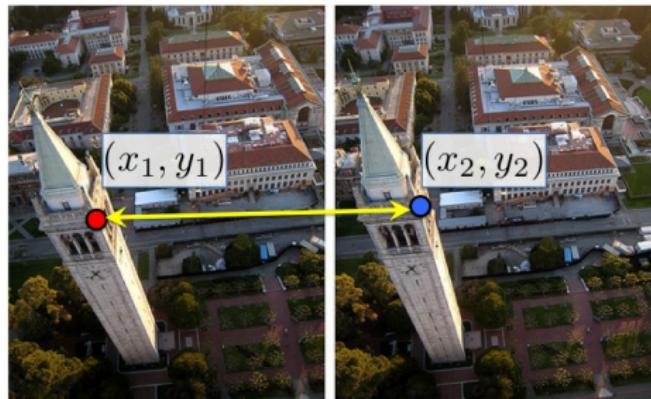
- we know that different w_1 mean different points X_1 on the projective line
- Where (x_1, y_1) maps to in the 2nd image depends on the 3D location of X_1 !

What If I Move the Camera?

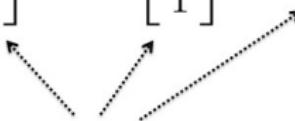
- **Summary:** So if I **move** the camera, I can't easily map one image to the other. **The mapping depends on the 3D scene behind the image.**
- What about the opposite, what if I know that points (x_1, y_1) in the first image and (x_2, y_2) in the second belong to the same 3D point?

What If I Move the Camera?

- **Summary:** So if I **move** the camera, I can't easily map one image to the other. **The mapping depends on the 3D scene behind the image.**
- What about the opposite, what if I know that points (x_1, y_1) in the first image and (x_2, y_2) in the second belong to the same 3D point?



$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$

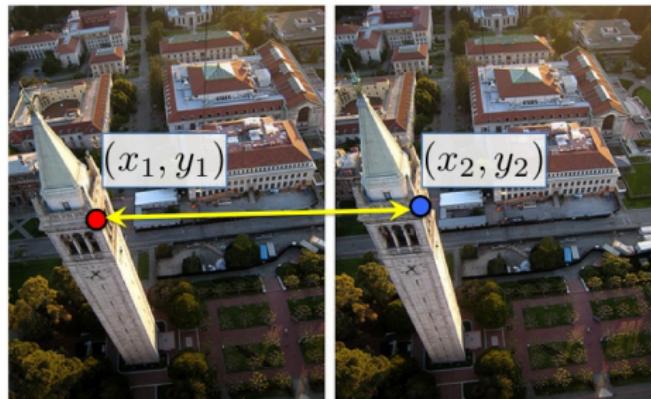


We know this

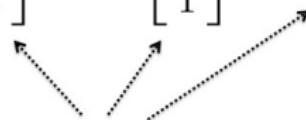


What If I Move the Camera?

- **Summary:** So if I **move** the camera, I can't easily map one image to the other. **The mapping depends on the 3D scene behind the image.**
- What about the opposite, what if I know that points (x_1, y_1) in the first image and (x_2, y_2) in the second belong to the same 3D point?



$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - Kt$$



We know this



- We can compute w_1 and w_2
- We can compute point in 3D!

What If I Move the Camera?

- **Summary:** So if I **move** the camera, I can't easily map one image to the other. **The mapping depends on the 3D scene behind the image.**
- What about the opposite, what if I know that points (x_1, y_1) in the first image and (x_2, y_2) in the second belong to the same 3D point?
- This great fact is called **stereo**
- This brings us to the **two-view** geometry, which we'll look at next

Summary – Stuff You Need To Know

Perspective Projection:

- If point Q is in camera's coordinate system:

$$\bullet \quad Q = (X, Y, Z)^T \rightarrow q = \left(\frac{f \cdot X}{Z} + p_x, \frac{f \cdot Y}{Z} + p_y \right)^T$$

$$\bullet \quad \text{Same as: } Q = (X, Y, Z)^T \rightarrow \begin{bmatrix} w \cdot x \\ w \cdot y \\ w \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow q = \begin{bmatrix} x \\ y \end{bmatrix}$$

where $K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$ is camera intrinsic matrix

- If Q is in world coordinate system, then the full projection is characterized by a 3×4 matrix P:

$$\begin{bmatrix} w \cdot x \\ w \cdot y \\ w \end{bmatrix} = \underbrace{K[R \mid t]}_P \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Summary – Stuff You Need To Know

Perspective Projection:

- All parallel lines in 3D with the same direction meet in one, so-called vanishing point in the image
- All lines that lie on a plane have vanishing points that lie on a line, so-called vanishing line
- All parallel planes in 3D have the same vanishing line in the image

Orthographic Projection

- Projections simply drops the Z coordinate:

$$Q = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Parallel lines in 3D are parallel in the image

Stereo

Depth from Two Views: Stereo

- All points on projective line to P map to p

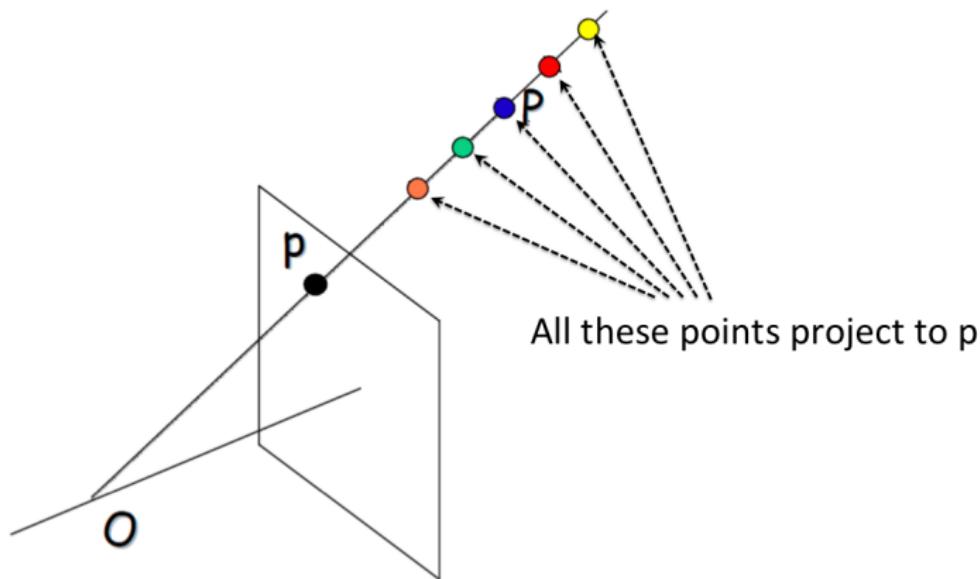


Figure: One camera

Depth from Two Views: Stereo

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

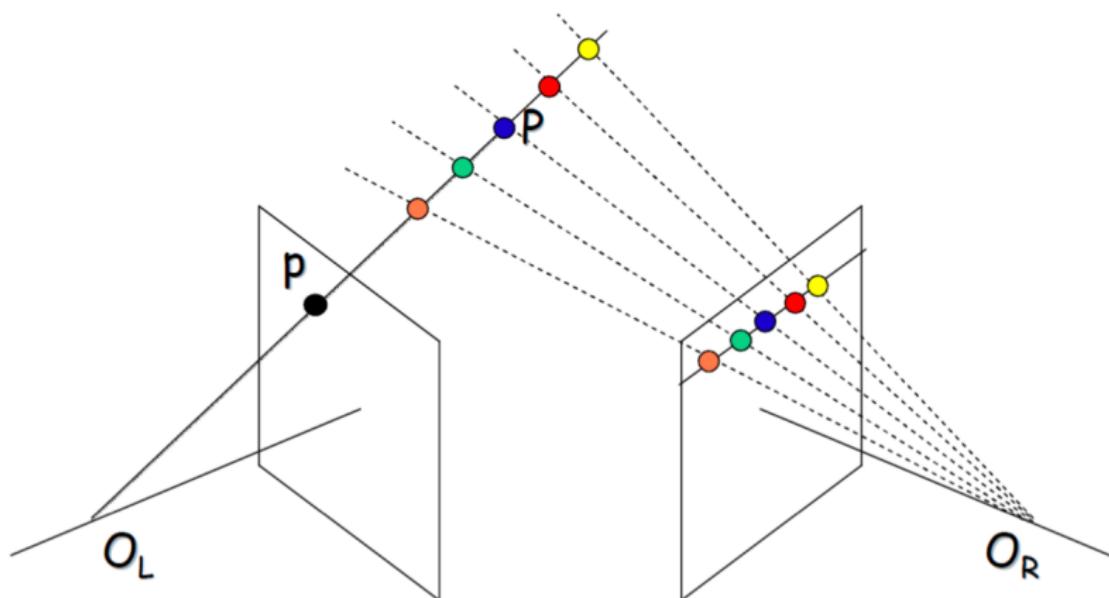


Figure: Add another camera

Depth from Two Views: Stereo

- If I search this line to find correspondences...

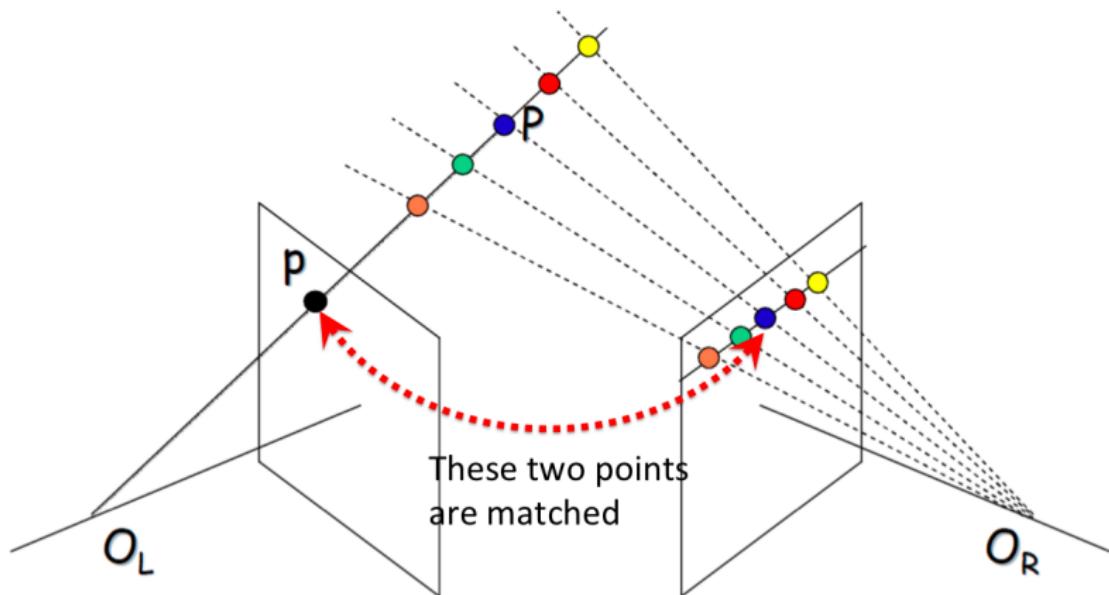


Figure: If I am able to find corresponding points in two images...

Depth from Two Views: Stereo

- I can get 3D!

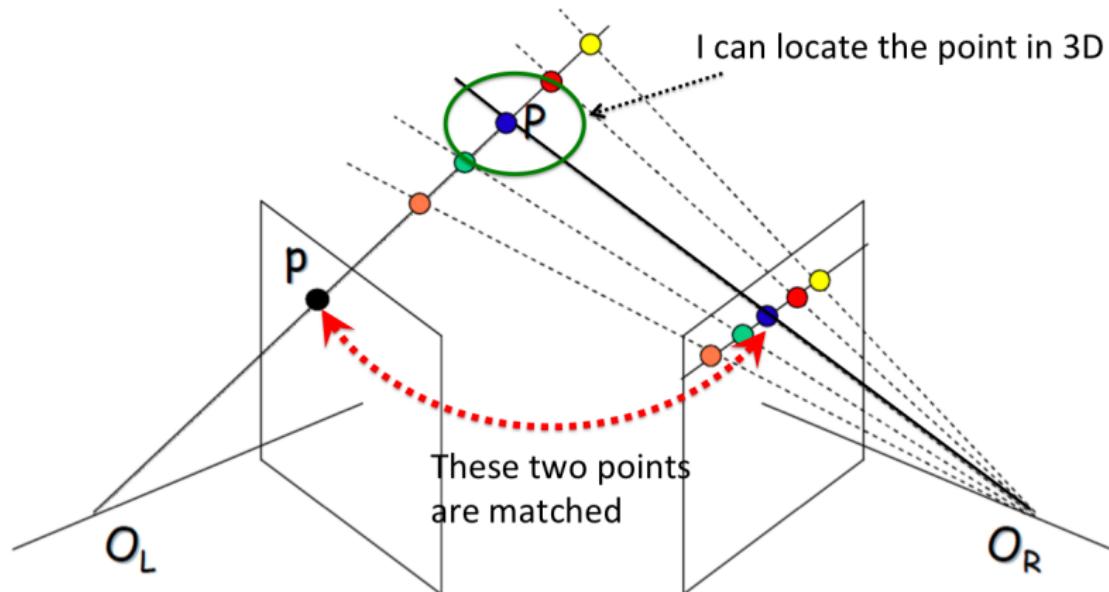


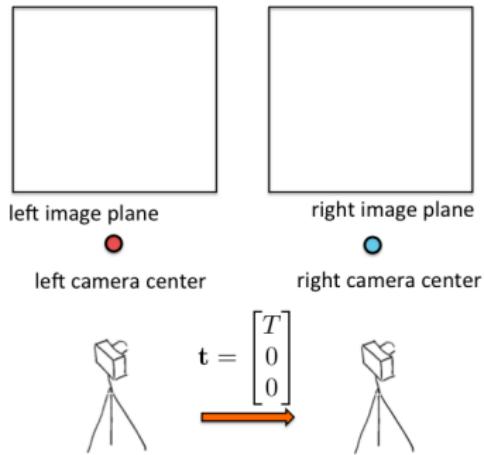
Figure: I can get a point in 3D by triangulation!

Stereo

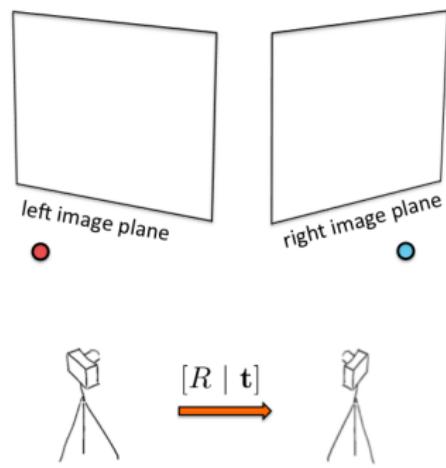
Epipolar geometry

- Case with two cameras with parallel optical axes
- General case

Parallel stereo cameras:



General stereo cameras:

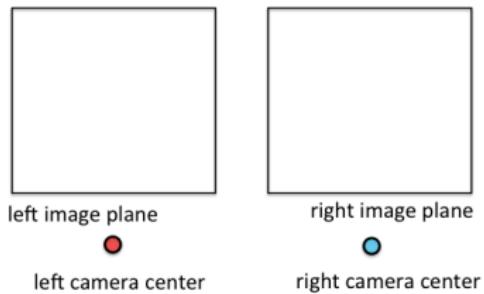


Stereo

Epipolar geometry

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

Parallel stereo cameras:

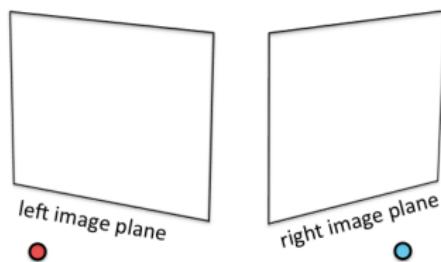


$$\mathbf{t} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$



$$[R \mid t]$$

General stereo cameras:

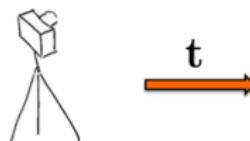
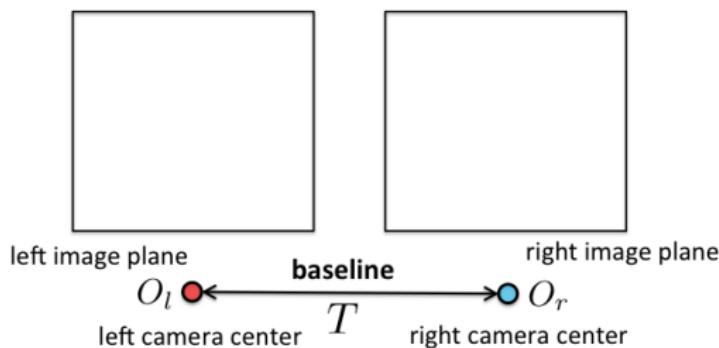


$$[R \mid t]$$



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**.

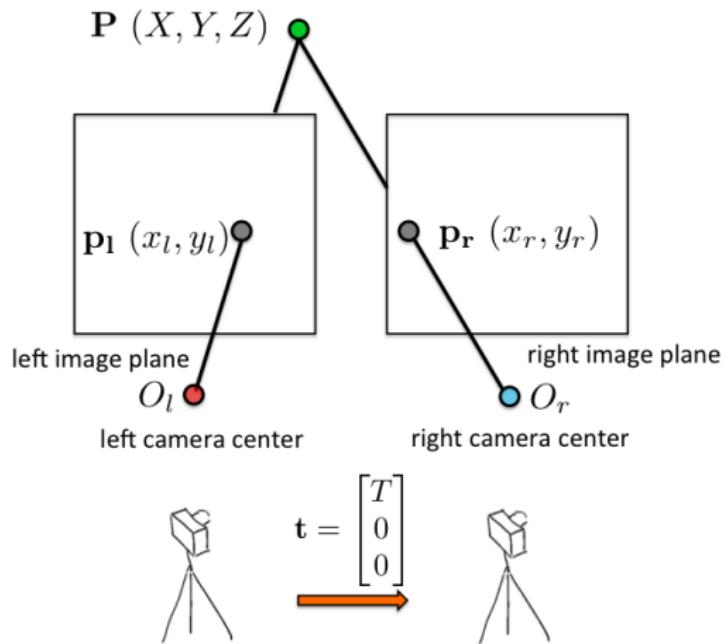


$$\mathbf{t} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

The right camera is shifted to the right in X direction

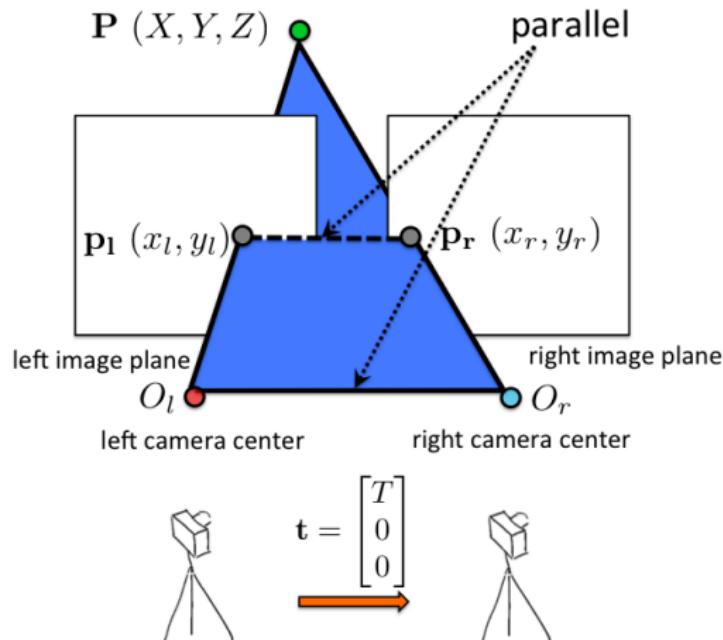
Stereo: Parallel Calibrated Cameras

- Pick a point P in the world



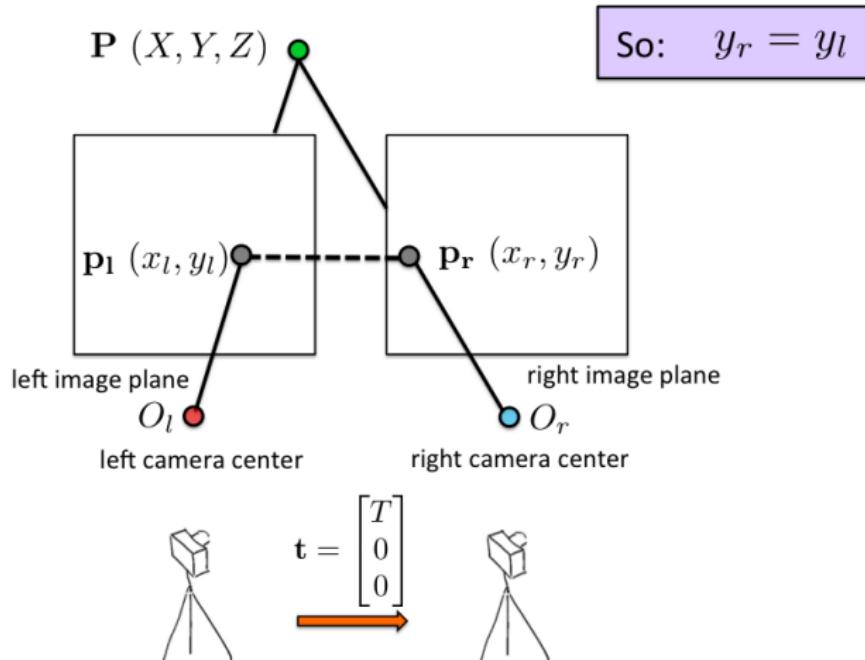
Stereo: Parallel Calibrated Cameras

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



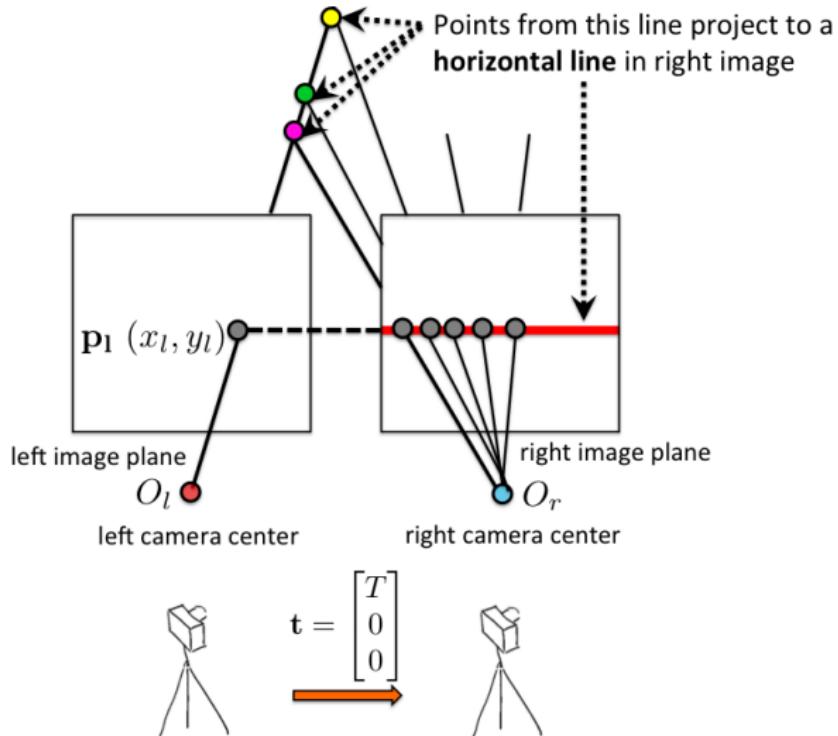
Stereo: Parallel Calibrated Cameras

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



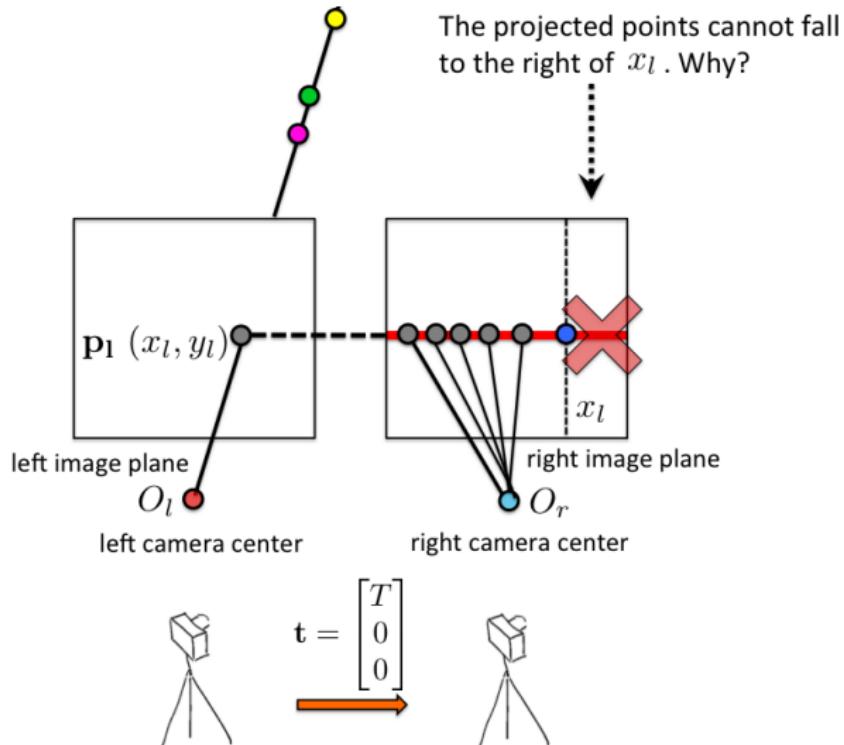
Stereo: Parallel Calibrated Cameras

- So all points on the projective line $O_l p_l$ project to a horizontal line with $y = y_l$ on the right image. This is nice, let's remember this.



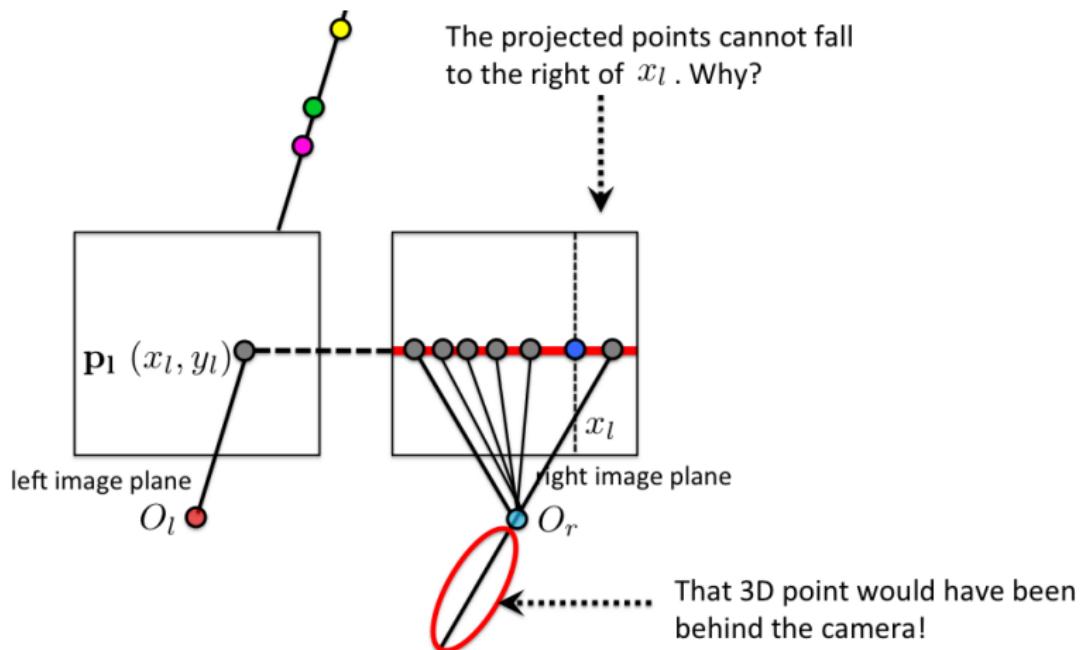
Stereo: Parallel Calibrated Cameras

- Another observation: No point from $O_l p_l$ can project to the right of x_l in the right image. **Why?**



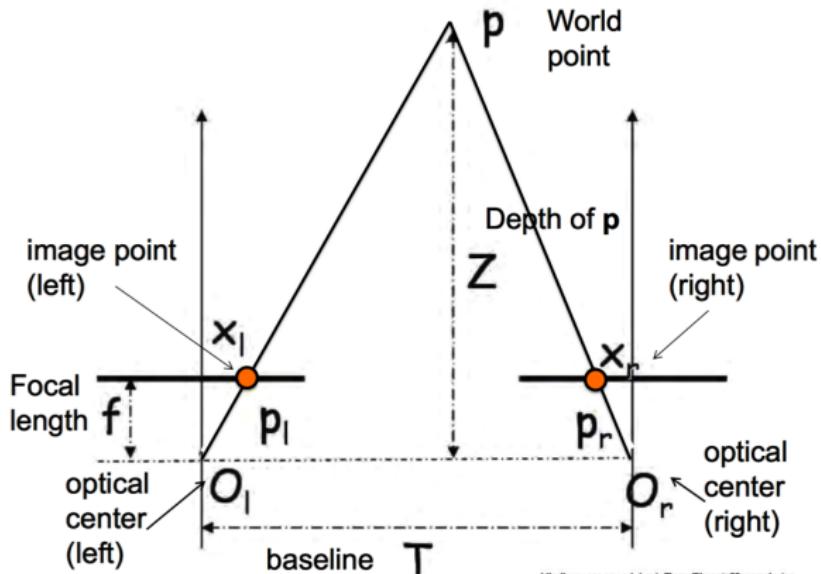
Stereo: Parallel Calibrated Cameras

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



Stereo: Parallel Calibrated Cameras

- Since our points p_l and 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).

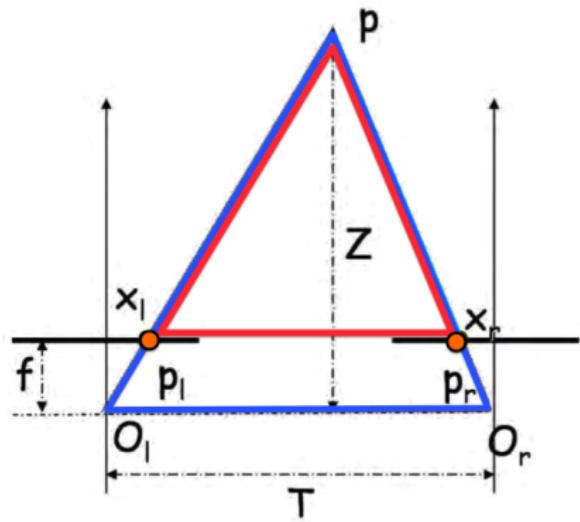


<http://www.cs.psu.edu/~zylin/Demo/Stereo/2Dgeometry.jpg>

[Adopted from: J. Hays]

Stereo: Parallel Calibrated Cameras

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



Similar triangles:

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

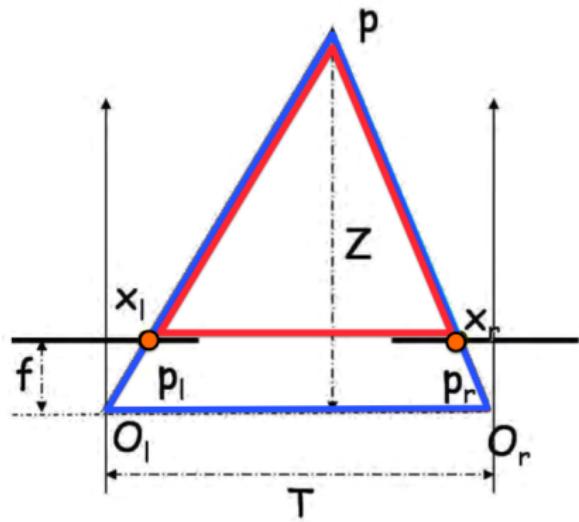
$$Z = \frac{f \cdot T}{x_l - x_r}$$

baseline
focal length
disparity

[Adopted from: J. Hays]

Stereo: Parallel Calibrated Cameras

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



Similar triangles:

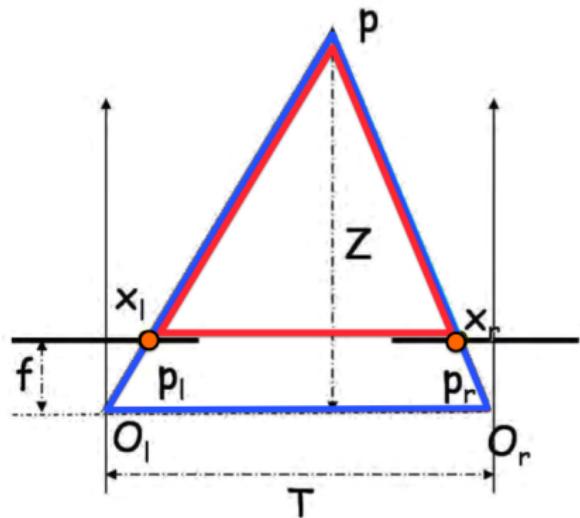
$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

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

So if I know x_l and x_r , then I can compute Z !

Stereo: Parallel Calibrated Cameras

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



Similar triangles:

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

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

$$x = \frac{f \cdot X}{Z} + p_x$$

And if I know Z , I can compute X and Y , which gives me the point in 3D

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ?



left image



right image

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching.



left image



right image

the match will be on this line (same y)

(CAREFUL: this is only true for parallel cameras. Generally, line not horizontal)

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching.

We are looking for this point



left image x_l



right image x_l

the match will be **on the left** of x_l

how do I find it?

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

We call this line a **scanline**



left image



right image

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

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

How similar?



left image



right image

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

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

How similar?



left image



right image

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

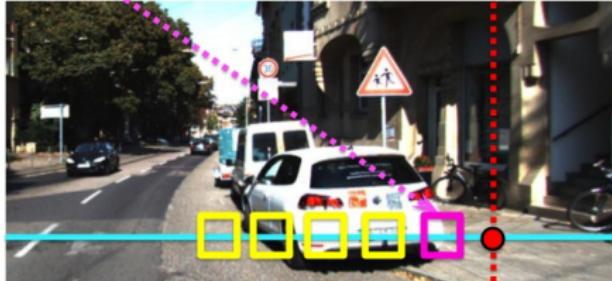
Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

Most similar. A match!



left image



right image

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

Stereo: Parallel Calibrated Cameras

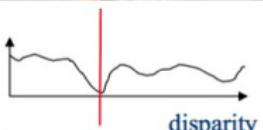
- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .



left image



Matching cost



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

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

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

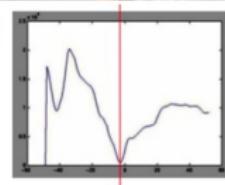
$$SSD(\text{patch}_l, \text{patch}_r) = \sum_x \sum_y (I_{\text{patch}_l}(x, y) - I_{\text{patch}_r}(x, y))^2$$



left image



SSD



disparity

Compute a matching cost

Matching cost: SSD (look for minima)

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .

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

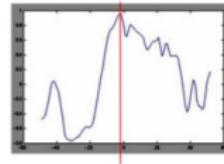


left image



Compute a matching cost

Norm.
Corr.



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

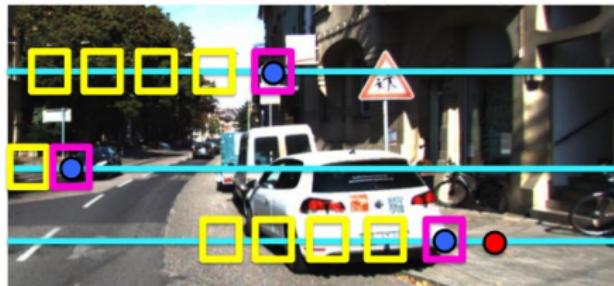
disparity

Stereo: Parallel Calibrated Cameras

- For each point x_l , how do I get x_r ? By matching. Patch around x_r should look similar to the patch around x_l .



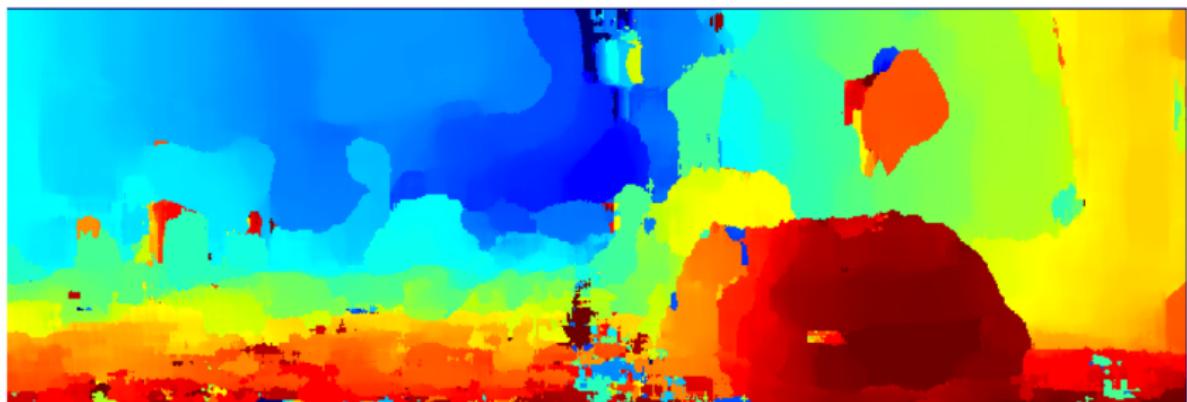
left image



Do this for all the points in the left image!

Stereo: Parallel Calibrated Cameras

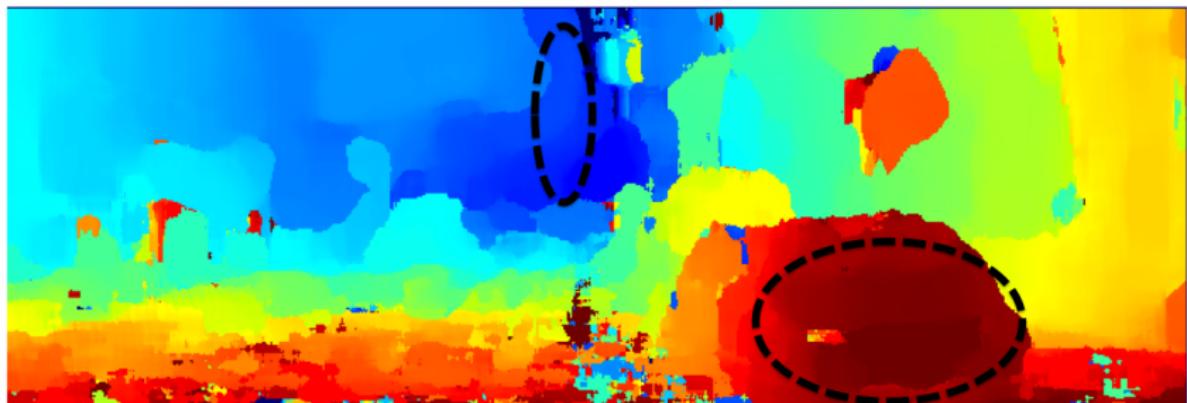
- We get a disparity map as a result



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

Stereo: Parallel Calibrated Cameras

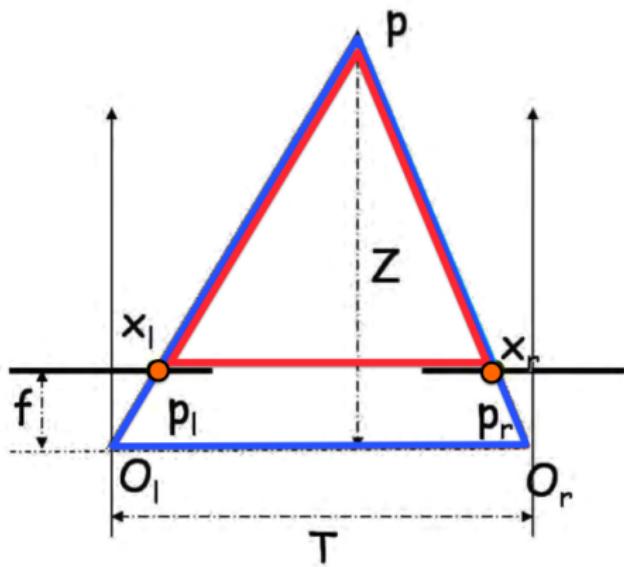
- We get a disparity map as a result



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

Stereo: Parallel Calibrated Cameras

- Depth and disparity are inversely proportional



Similar triangles:

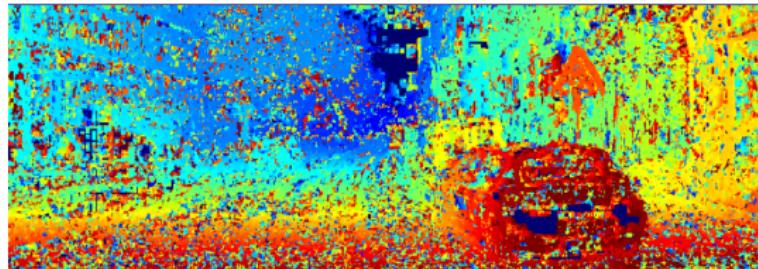
$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

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

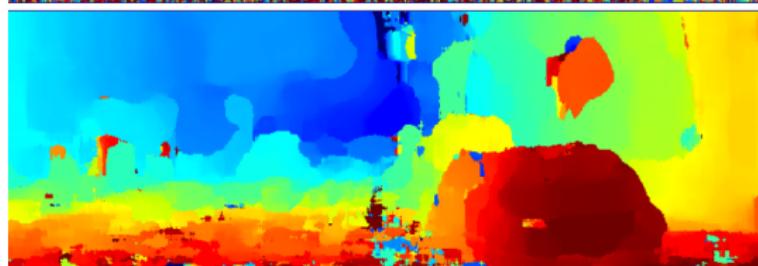
Depth (Z) and disparity are
inversely proportional

Stereo: Parallel Calibrated Cameras

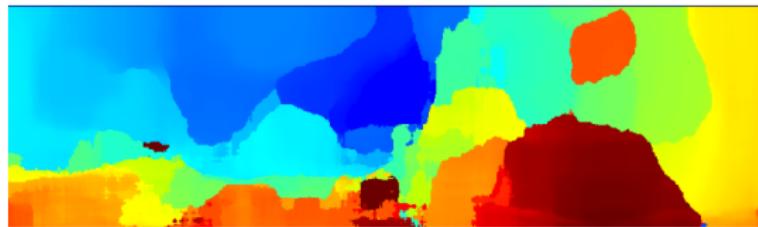
- Smaller patches: more detail, but noisy. Bigger: less detail, but smooth



patch size = 5



patch size = 35



patch size = 85