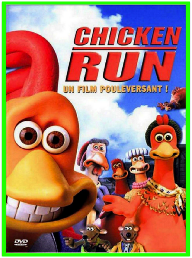
Back to the Homography: The Why

Homography



$\xrightarrow{T?}$

- In Lecture 9 we said that a homography is a transformation that maps a projective plane to another projective plane.
- We shamelessly dumped the following equation for homography without explanation:

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

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

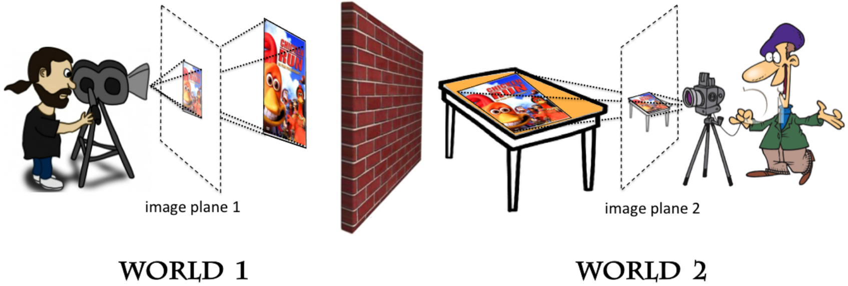


Figure: We have our object in two different worlds, in two different poses relative to camera, two different photographers, and two different cameras.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

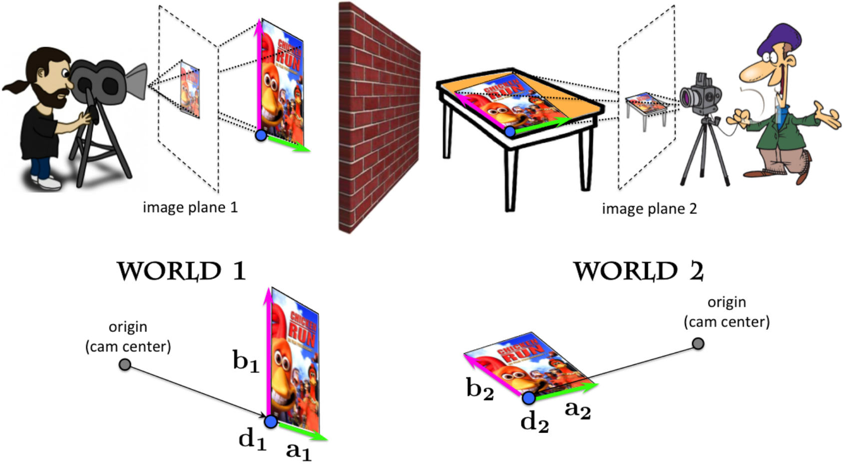


Figure: Our object is a plane. Each plane is characterized by one point **d** on the plane and two independent vectors **a** and **b** on the plane.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

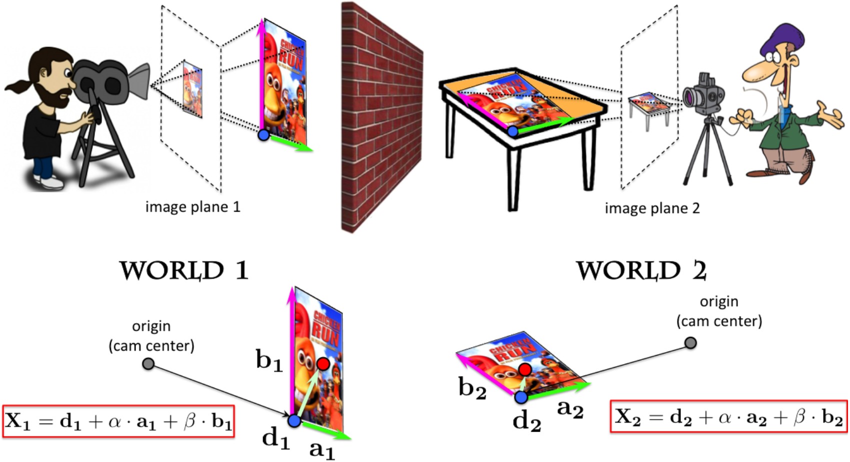


Figure: Then any other point **X** on the plane can be written as: $\mathbf{X} = \mathbf{d} + \alpha\mathbf{a} + \beta\mathbf{b}$.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

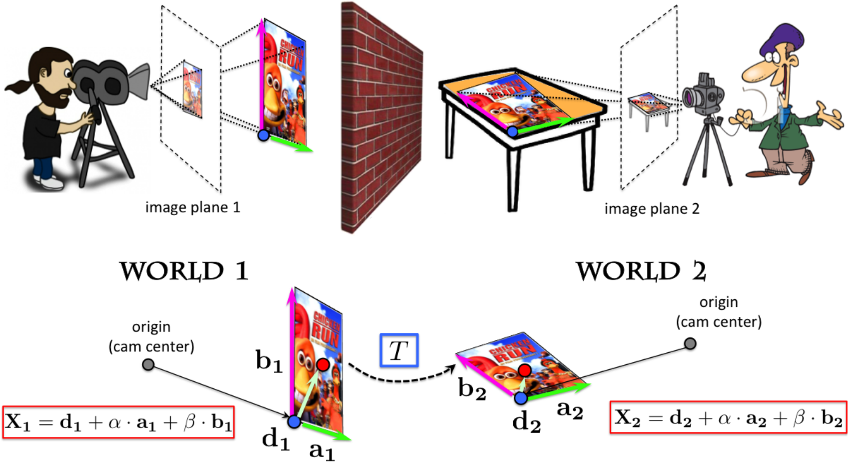


Figure: Any two Chicken Run DVDs on our planet are related by some transformation T. We'll compute it, don't worry.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

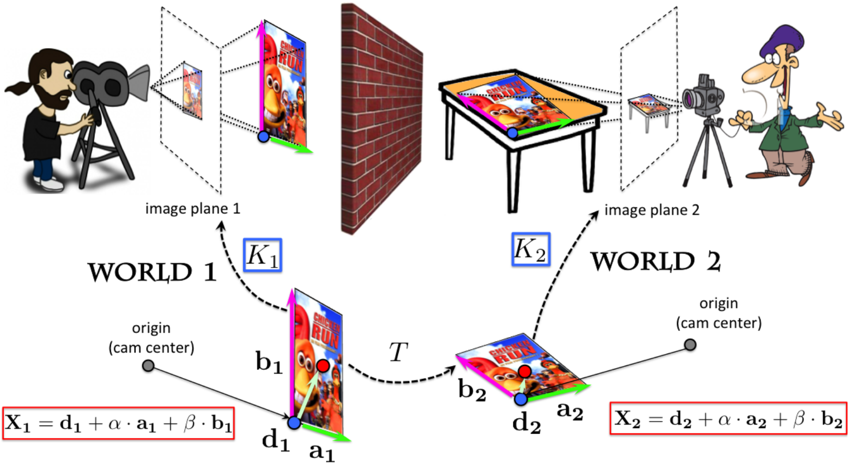


Figure: Each object is seen by a different camera and thus projects to the corresponding image plane with different camera intrinsics.

Homography

- Let's revisit our transformation in the (new) light of perspective projection.

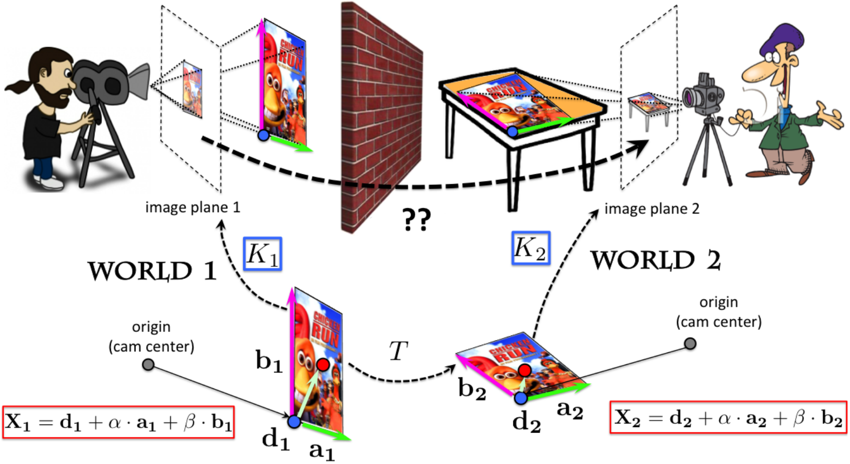


Figure: Given this, the question is what's the transformation that maps the DVD on the first image to the DVD in the second image?

Homography

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

$$\text{First plane}: \quad \mathbf{X_1} = \mathbf{d_1} + \alpha \cdot \mathbf{a_1} + \beta \cdot \mathbf{b_1}$$
$$\text{Second plane}: \quad \mathbf{X_2} = \mathbf{d_2} + \alpha \cdot \mathbf{a_2} + \beta \cdot \mathbf{b_2}$$

Via α and β, the two points $\mathbf{X_1}$ and $\mathbf{X_2}$ are in the same location relative to each plane.

Homography

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

$$\text{First plane}: \ \mathbf{X_1} = \mathbf{d_1} + \alpha \cdot \mathbf{a_1} + \beta \cdot \mathbf{b_1}$$
$$\text{Second plane}: \ \mathbf{X_2} = \mathbf{d_2} + \alpha \cdot \mathbf{a_2} + \beta \cdot \mathbf{b_2}$$

Via α and β, the two points $\mathbf{X_1}$ and $\mathbf{X_2}$ are in the same location relative to each plane.

- We can rewrite this using homogeneous coordinates:

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

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

Homography

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

$$\text{First plane}: \ \mathbf{X_1} = \mathbf{d_1} + \alpha \cdot \mathbf{a_1} + \beta \cdot \mathbf{b_1}$$
$$\text{Second plane}: \ \mathbf{X_2} = \mathbf{d_2} + \alpha \cdot \mathbf{a_2} + \beta \cdot \mathbf{b_2}$$

  Via α and β, the two points $\mathbf{X_1}$ and $\mathbf{X_2}$ are in the same location relative to each plane.
- We can rewrite this using homogeneous coordinates:

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

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

- Careful: $A_1 = \begin{bmatrix} \mathbf{a_1} & \mathbf{b_1} & \mathbf{d_1} \end{bmatrix}$ and $A_2 = \begin{bmatrix} \mathbf{a_2} & \mathbf{b_2} & \mathbf{d_2} \end{bmatrix}$ are 3×3 matrices.

Homography

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

$$\text{First plane}: \ \mathbf{X_1} = \mathbf{d_1} + \alpha \cdot \mathbf{a_1} + \beta \cdot \mathbf{b_1}$$
$$\text{Second plane}: \ \mathbf{X_2} = \mathbf{d_2} + \alpha \cdot \mathbf{a_2} + \beta \cdot \mathbf{b_2}$$

  Via α and β, the two points $\mathbf{X_1}$ and $\mathbf{X_2}$ are in the same location relative to each plane.
- We can rewrite this using homogeneous coordinates:

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

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

- Careful: $A_1 = \begin{bmatrix} \mathbf{a_1} & \mathbf{b_1} & \mathbf{d_1} \end{bmatrix}$ and $A_2 = \begin{bmatrix} \mathbf{a_2} & \mathbf{b_2} & \mathbf{d_2} \end{bmatrix}$ are 3×3 matrices.

Homography

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

$$X_2 = T\,X_1$$

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

- Expand it:

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

- 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.

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

Homography

- From previous slide:

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

- Insert $\mathbf{X_2} = T\,\mathbf{X_1}$ into equality on the right:

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

- And finally:

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

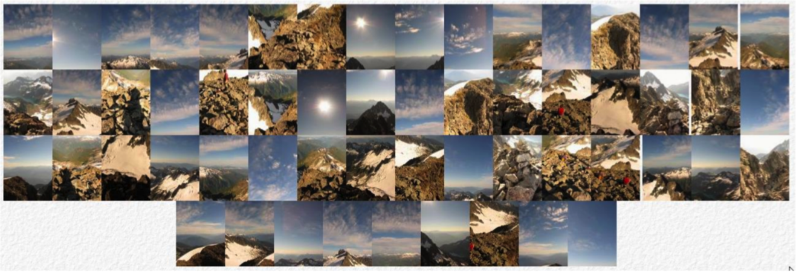
Homography

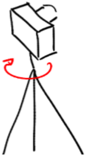
- The nice thing about homography is that once we have it, we can compute where any point from one projective plane maps to on the second projective plane. We do not need to know the 3D location of that point. We don't even need to know the camera parameters.
- We still owe one more explanation for Lecture 9.

Homography

- The nice thing about homography is that once we have it, we can compute where any point from one projective plane maps to on the second projective plane. We do not need to know the 3D location of that point. We don't even need to know the camera parameters.
- We still owe one more explanation for Lecture 9.

Remember Panorama Stitching from Lecture 9?

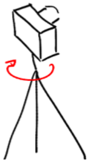


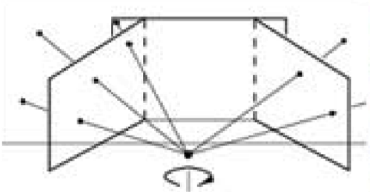


Take a tripod, rotate camera and take pictures

[Source: Fernando Flores-Mangas]

Remember Panorama Stitching from Lecture 9?





- 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):

$$\mathbf{X_2} = R^T \mathbf{X_1}$$

  where $\mathbf{X_1}$ is a 3D point in the coordinate system of the first camera and $\mathbf{X_2}$ the 3D point in the coordinate system of the rotated camera.

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

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K\mathbf{X_1} \qquad \text{and} \qquad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K\mathbf{X_2}$$

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

Rotating the Camera

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

$$\mathbf{X_2} = R^T \mathbf{X_1}$$

  where $\mathbf{X_1}$ is a 3D point in the coordinate system of the first camera and $\mathbf{X_2}$ the 3D point in the coordinate system of the rotated camera.

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

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K\mathbf{X_1} \qquad \text{and} \qquad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K\mathbf{X_2}$$

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

- And this is a homography

Rotating the Camera

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

$$\mathbf{X_2} = R^T \mathbf{X_1}$$

  where $\mathbf{X_1}$ is a 3D point in the coordinate system of the first camera and $\mathbf{X_2}$ the 3D point in the coordinate system of the rotated camera.

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

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K\mathbf{X_1} \qquad \text{and} \qquad w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = K\mathbf{X_2}$$

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

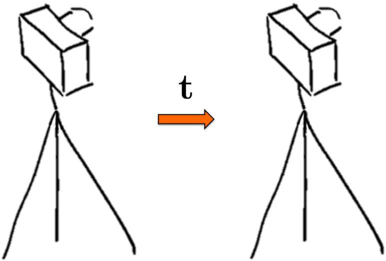
- And this is a homography

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 $\mathbf{t}$, then: $\mathbf{X_2} = \mathbf{X_1} - \mathbf{t}$. Let's try the same trick again:

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

What If I Move the Camera?

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

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

What If I Move the Camera?

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

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

What If I Move the Camera?

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

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

What If I Move the Camera?

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

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

- Hmm... Different values of w_1 give me different points in the second image.
- So even if I have K and $\mathbf{t}$ 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 $\mathbf{t}$, then: $\mathbf{X_2} = \mathbf{X_1} - \mathbf{t}$. Let's try the same trick again:

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

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

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K\mathbf{X_1}$$

  we know that different w_1 mean different points $\mathbf{X_1}$ on the projective line

What If I Move the Camera?

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

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

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

$$w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = K\mathbf{X_1}$$

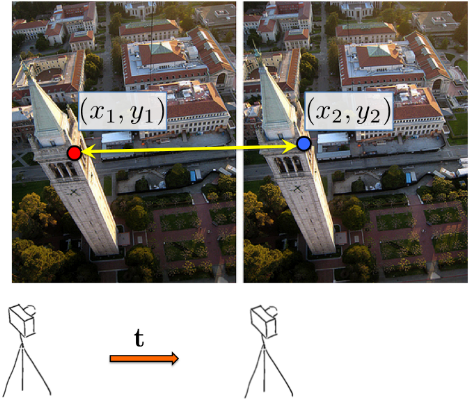
  we know that different w_1 mean different points $\mathbf{X_1}$ on the projective line
- Where (x_1, y_1) maps to in the 2nd image depends on the 3D location of $\mathbf{X_1}$!

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?

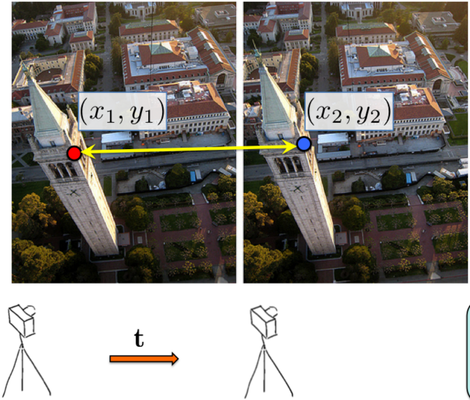


$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - K\mathbf{t}$$

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?



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

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 $\mathbf{Q}$ is in camera's coordinate system:
  - $\mathbf{Q} = (X, Y, Z)^T \quad \rightarrow \quad \mathbf{q} = \left(\frac{f \cdot X}{Z} + p_x, \frac{f \cdot Y}{Z} + p_y\right)^T$
  - Same as: $\mathbf{Q} = (X, Y, Z)^T \quad \rightarrow \quad \begin{bmatrix} w \cdot x \\ w \cdot y \\ w \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \rightarrow \quad \mathbf{q} = \begin{bmatrix} x \\ y \end{bmatrix}$

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

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

$$\begin{bmatrix} w \cdot x \\ w \cdot y \\ w \end{bmatrix} = \underbrace{\mathbf{K}\left[\mathbf{R} \mid \mathbf{t}\right]}_{\mathbf{P}} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

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:

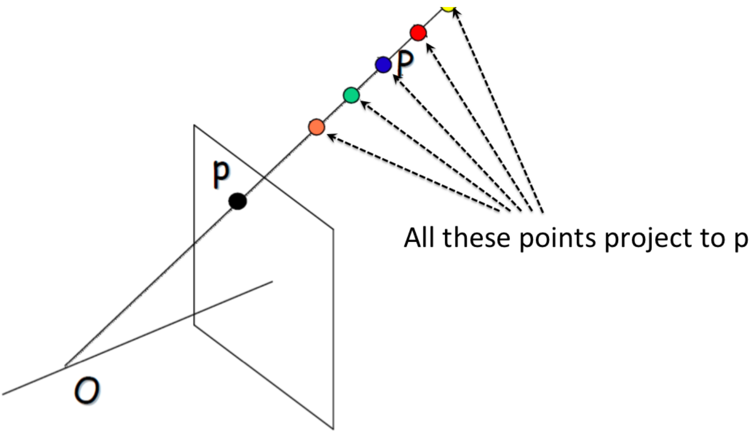
$$\mathbf{Q} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \qquad \rightarrow \qquad \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Parallel lines in 3D are parallel in the image

Stereo

Depth from Monocular Image

- We know that it's impossible to get depth from a single image



[Pic adopted from: J. Hays]

Depth from Monocular Image

- We know that it's impossible to get depth from a single image

[Pic from: S. Lazebnik]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

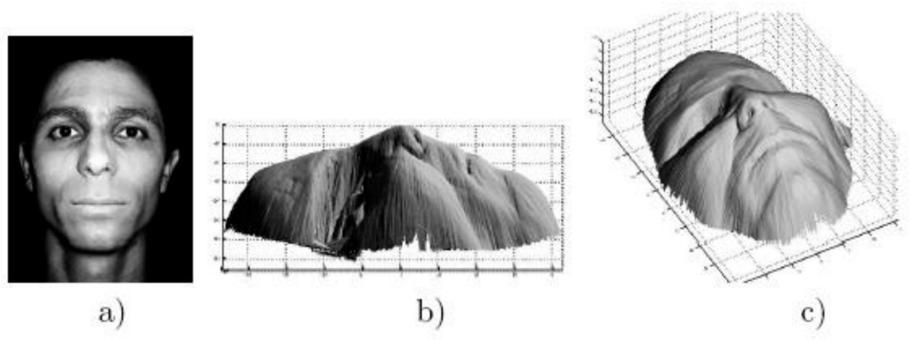


Figure: Shape from Shading

[Slide credit: J. Hays, pic from: Prados & Faugeras 2006]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

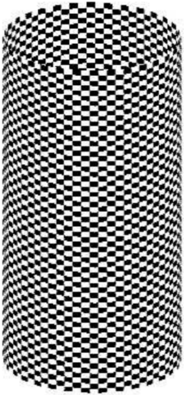
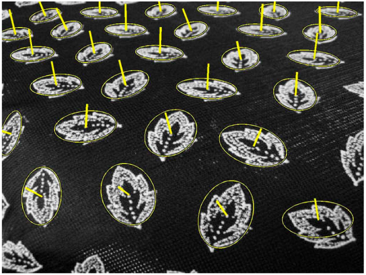


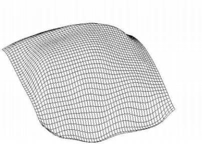
Figure: Shape from Texture: What do you see in the image?

[From the PhD Thesis: A.M. Loh. The recovery of 3-D structure using visual texture patterns]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image





(a) Estimated surface shape

(b) Texture projected onto surface

Figure: Shape from Texture

[From the PhD Thesis: A.M. Loh. The recovery of 3-D structure using visual texture patterns]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

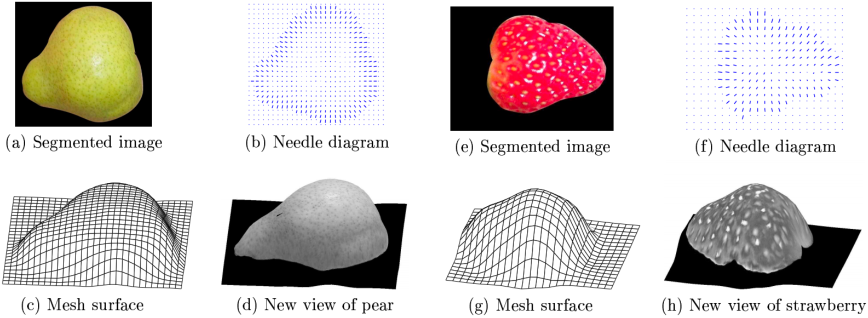


Figure: Shape from Texture

[From the PhD Thesis: A.M. Loh. The recovery of 3-D structure using visual texture patterns]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

Waterlilies: non-homogeneous | Donut: non-stationary | Woven lamp: anisotropic

Figure: Shape from Texture: And quite a lot of stuff around us is textured

[From the PhD Thesis: A.M. Loh. The recovery of 3-D structure using visual texture patterns]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

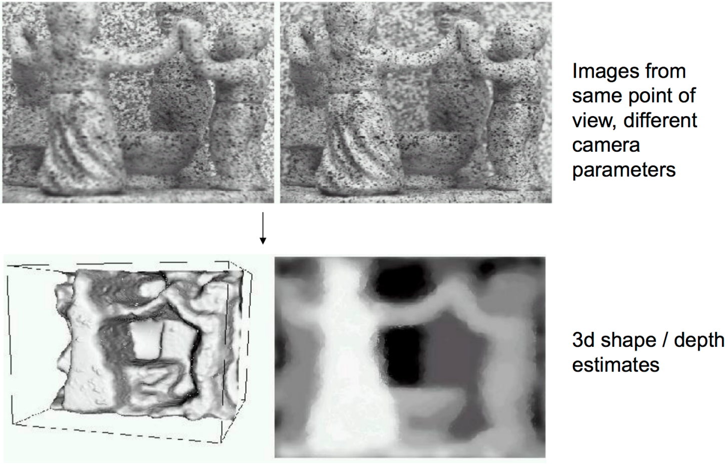


Figure: Shape from Focus/De-focus

[Slide credit: J. Hays, pics from: H. Jin and P. Favaro, 2002]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

Figure: Occlusion gives us **ordering** in depth

[Slide credit: J. Hays, Painting: Rene Magritt'e *Le Blanc-Seing*]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

Figure: Depth from Google: "Borrow" depth from Google's Street View Z-buffer [Paper: C. Wang, K. Wilson, N. Snavely, *Accurate Georegistration of Point Clouds using Geographic Data*, 3DV 2013. http://www.cs.cornell.edu/projects/georegister/docs/georegister_3dv.pdf]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

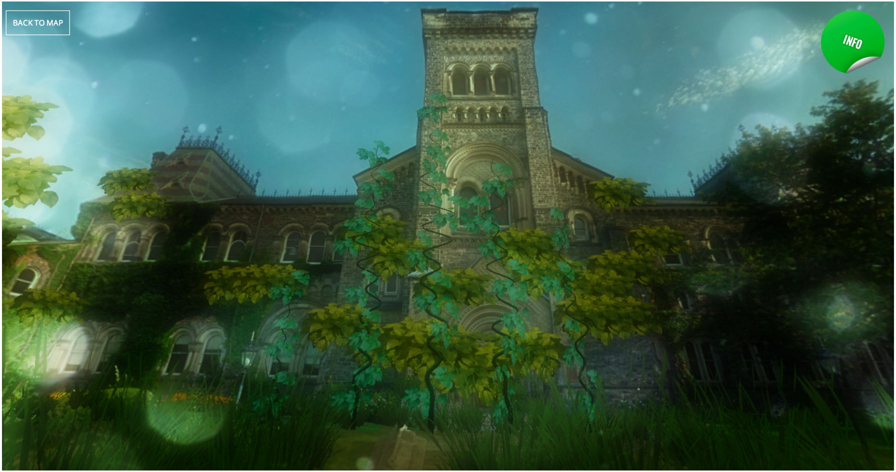


Figure: Depth from Google: Once you have depth you can render cool stuff

http://inear.se/urbanjungle/

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

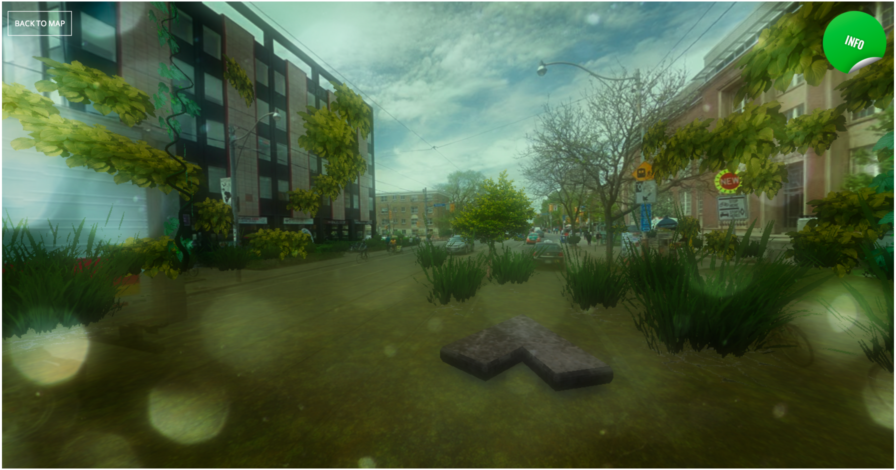


Figure: Depth from Google: Recognize this?

http://inear.se/urbanjungle/

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

Figure: Depth by tricking the brain: do you see the 3D object?

[Source: J. Hays, Pics from: `http://magiceye.com`]

Depth from Monocular Image

- But when present, we can use certain cues to get depth (3D) from one image

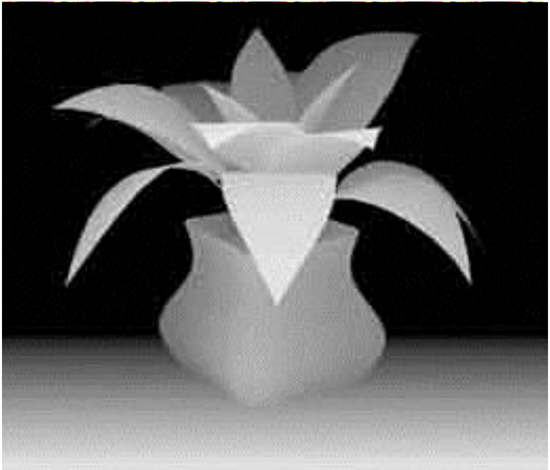


Figure: Depth by tricking the brain

[Source: J. Hays, Pics from: `http://magiceye.com`]

Depth from Two Views: Stereo

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

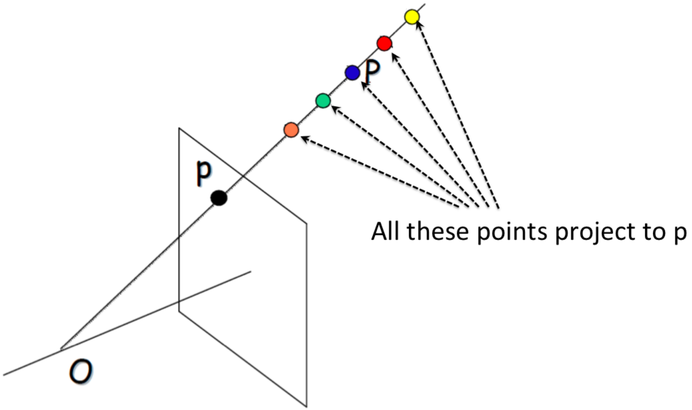


Figure: One camera

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

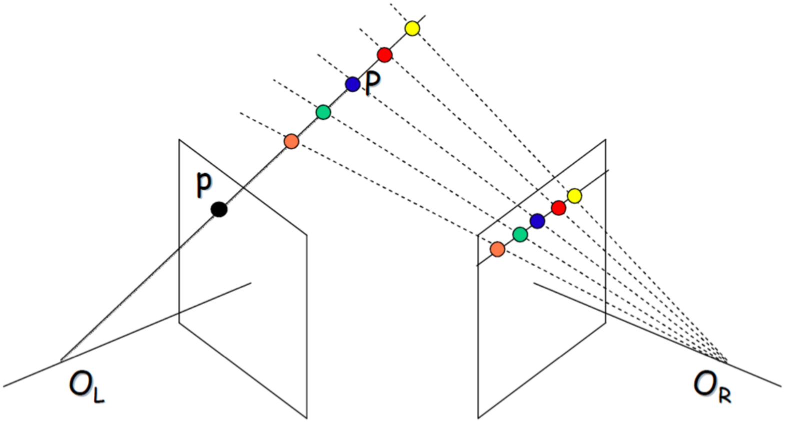


Figure: Add another camera

Depth from Two Views: Stereo

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

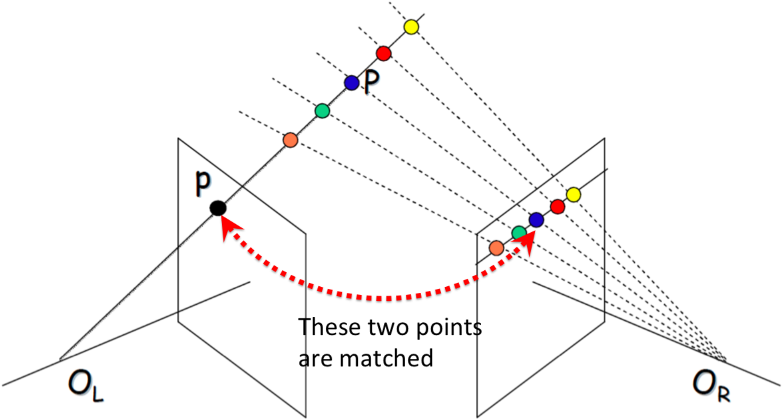


Figure: If I am able to find corresponding points in two images...

Depth from Two Views: Stereo

- I can get 3D!

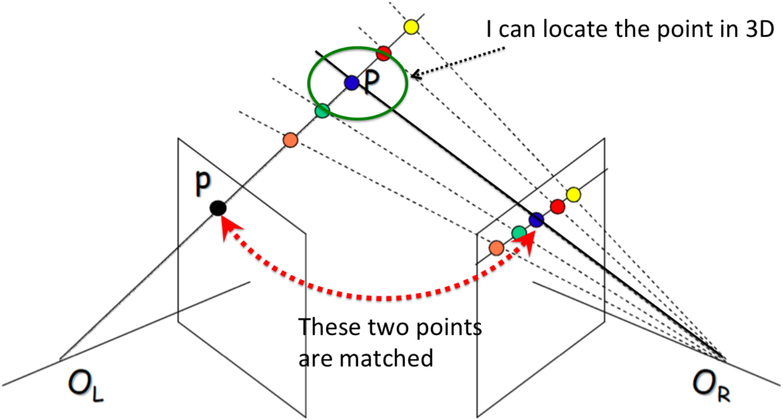
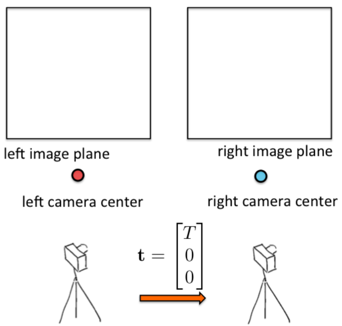


Figure: I can get a point in 3D by triangulation!

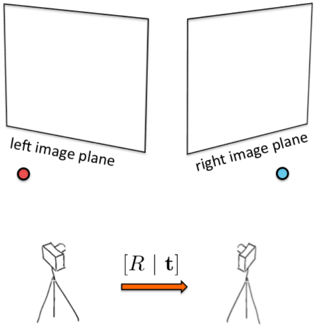
Epipolar geometry

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

Parallel stereo cameras:

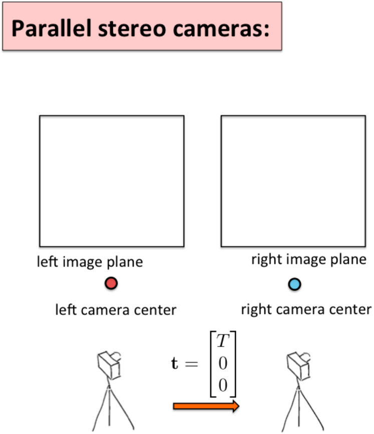


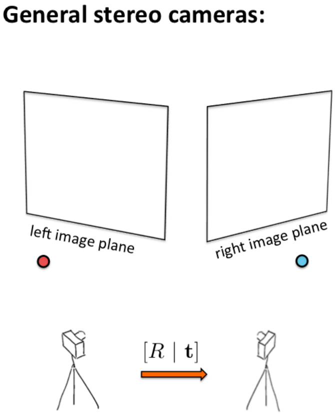
General stereo cameras:



Epipolar geometry

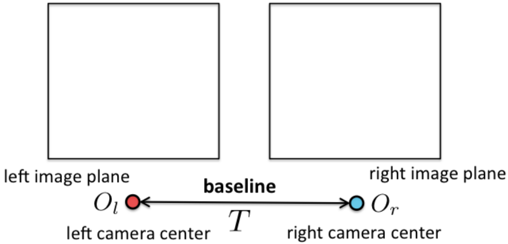
- Case with two cameras with parallel optical axes ← **First this**
- General case

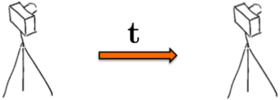




Stereo: Parallel Calibrated Cameras

- 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**.



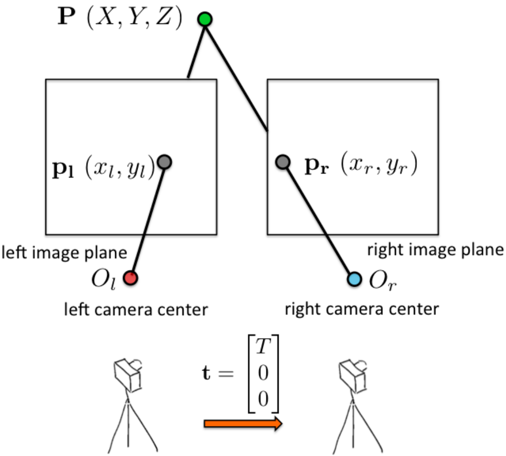


$$\mathbf{t} = \begin{bmatrix} T \\ 0 \\ 0 \end{bmatrix}$$

The right camera is shifted to the right in X direction

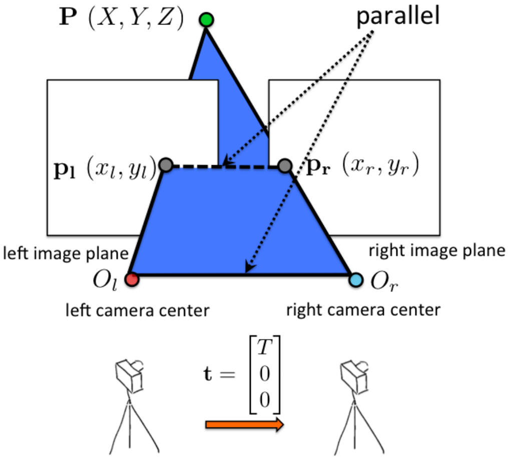
Stereo: Parallel Calibrated Cameras

- Pick a point P in the world



Stereo: Parallel Calibrated Cameras

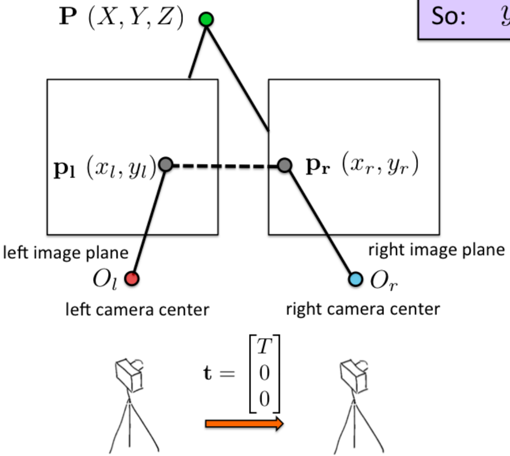
- Points $\mathbf{O_l}$, $\mathbf{O_r}$ and $\mathbf{P}$ (and $\mathbf{p_l}$ and $\mathbf{p_r}$) lie on a plane. Since two image planes lie on the same plane (distance f from each camera), the lines $\mathbf{O_lO_r}$ and $\mathbf{p_lp_r}$ are parallel.



Stereo: Parallel Calibrated Cameras

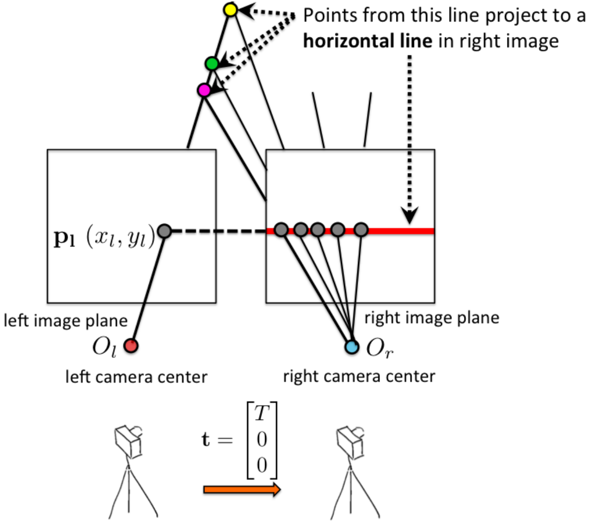
- Since lines $\mathbf{O_l O_r}$ and $\mathbf{p_l p_r}$ are parallel, and $\mathbf{O_l}$ and $\mathbf{O_r}$ have the same y, then also $\mathbf{p_l}$ and $\mathbf{p_r}$ have the same y: $y_r = y_l$!

So: $y_r = y_l$



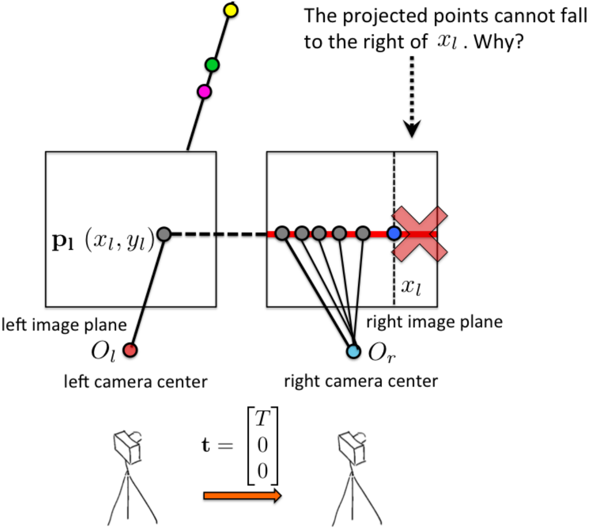
Stereo: Parallel Calibrated Cameras

- 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 remember this.



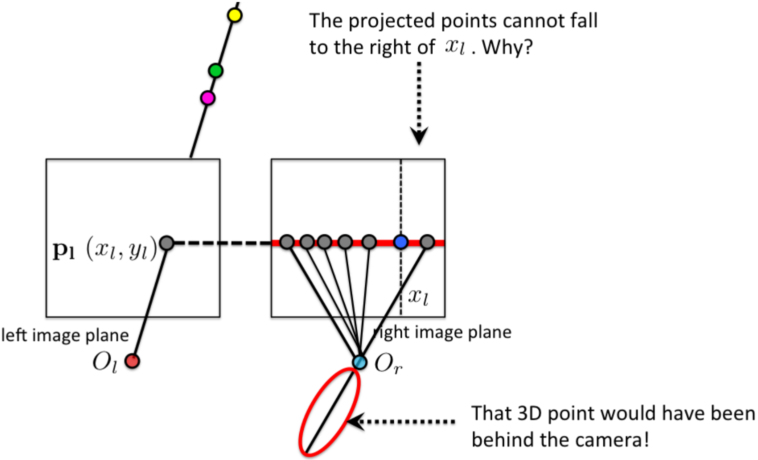
Stereo: Parallel Calibrated Cameras

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



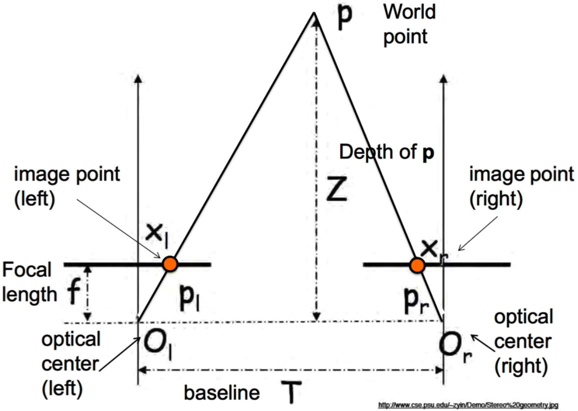
Stereo: Parallel Calibrated Cameras

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



Stereo: Parallel Calibrated Cameras

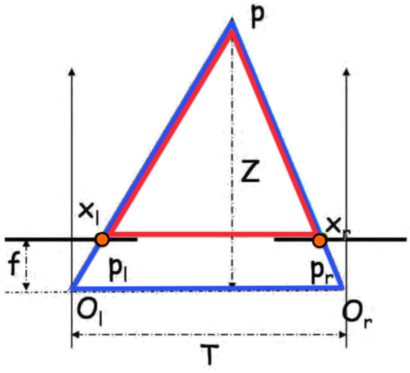
- Since our points $\mathbf{p_l}$ and $\mathbf{p_r}$ lie on a horizontal line, we can forget about y_l for a moment (it doesn't seem important). Let's look at the camera situation from the birdseye perspective instead. Let's see if we can find a connection between x_l, x_r and Z (because Z is what we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



[Adopted from: J. Hays]

Stereo: Parallel Calibrated Cameras

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



Similar triangles:

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

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

baseline

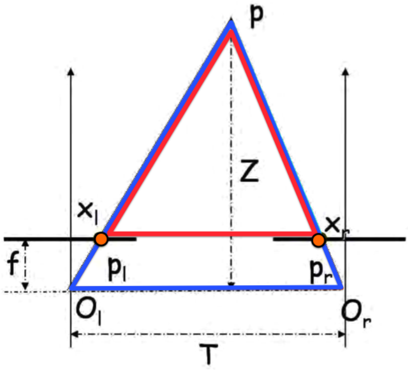
focal length

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:

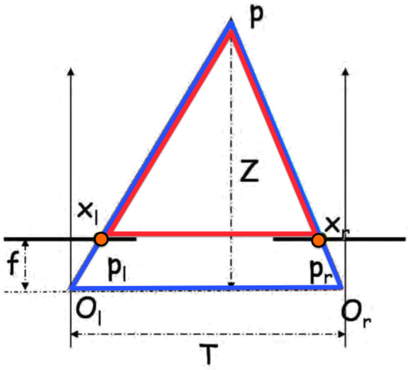
$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

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

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:

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

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

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

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

Stereo: Parallel Calibrated Cameras

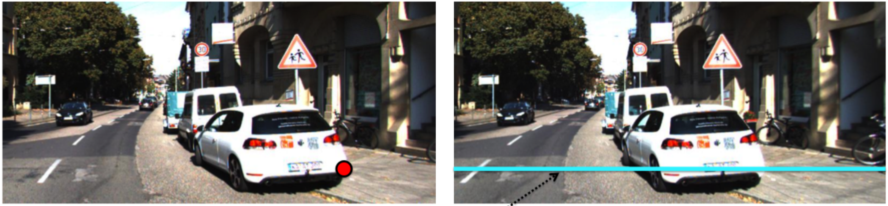
- For each point x_l, how do I get x_r?

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

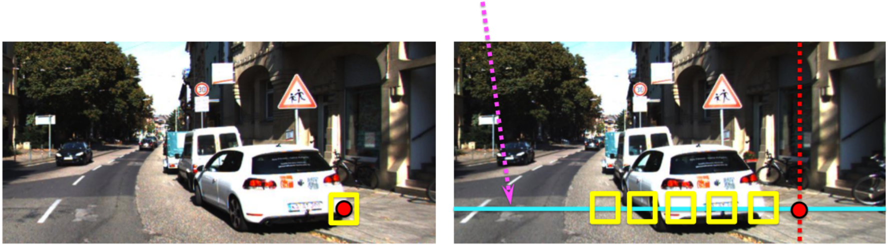
the match will be **on the left** of x_l

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

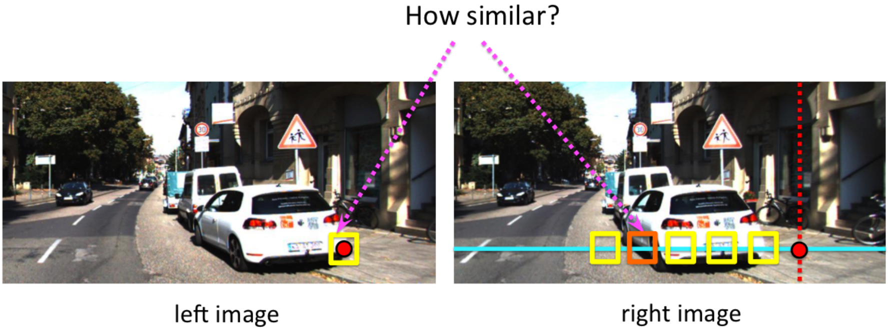
right image

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.



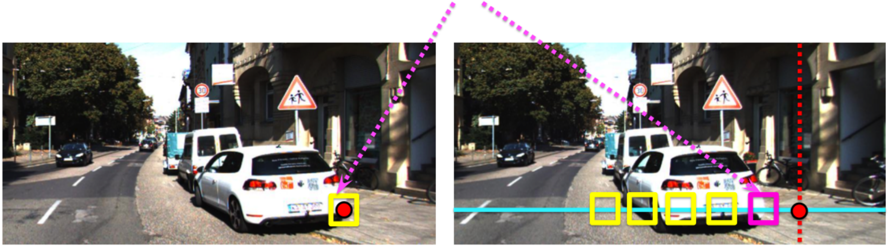
we **scan** the line and **compare** patches to the one in the left image

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!



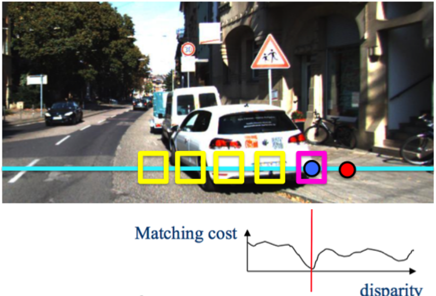
left image | right image

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

- 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



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

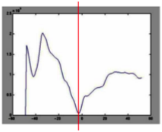
Matching cost: **SSD** or **normalized correlation**

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



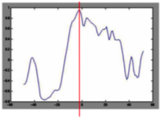
Compute a matching cost

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

- 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

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

Stereo: Parallel Calibrated Cameras

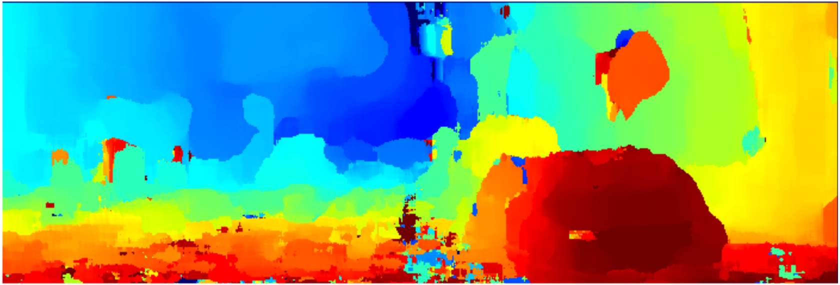
- For each point x_l, how do I get x_r? By matching. Patch around x_r should look similar to the patch around x_l.

left image

Do this for all the points in the left image!

Stereo: Parallel Calibrated Cameras

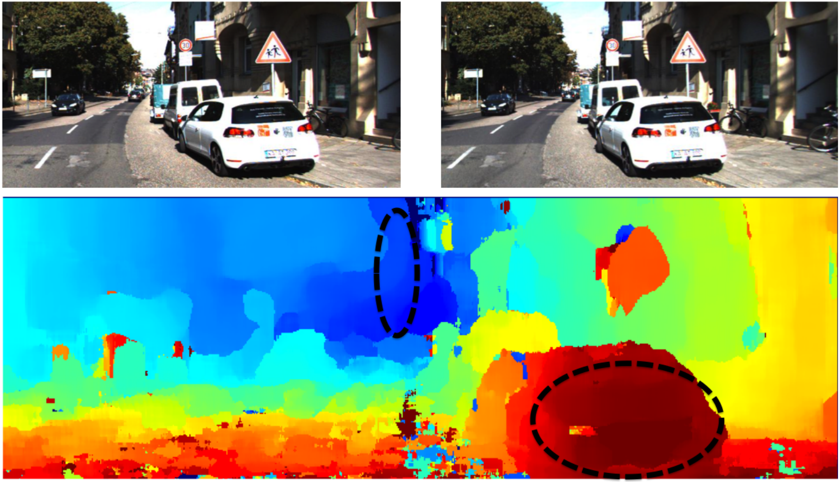
- We get a disparity map as a result



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

Stereo: Parallel Calibrated Cameras

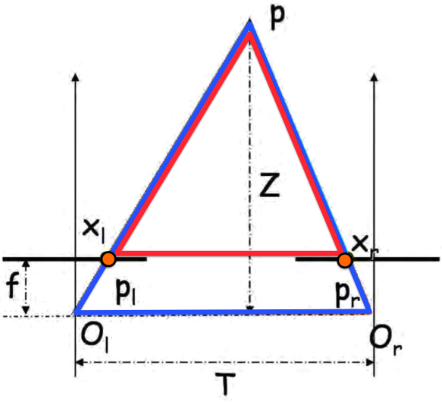
- We get a disparity map as a result



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:

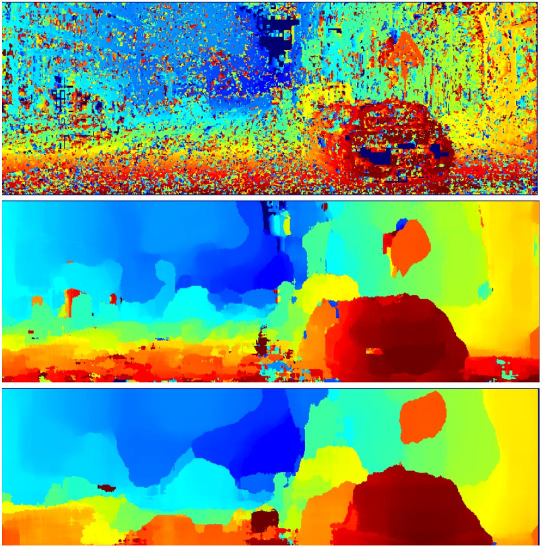
$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

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

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

Stereo: Parallel Calibrated Cameras

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



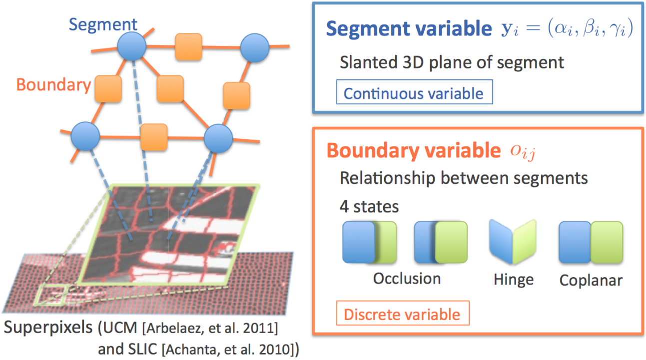
patch size $= 5$

patch size $= 35$

patch size $= 85$

You Can Do It Much Better...

- With Energy Minimization on top, e.g., a Markov Random Field (MRF)



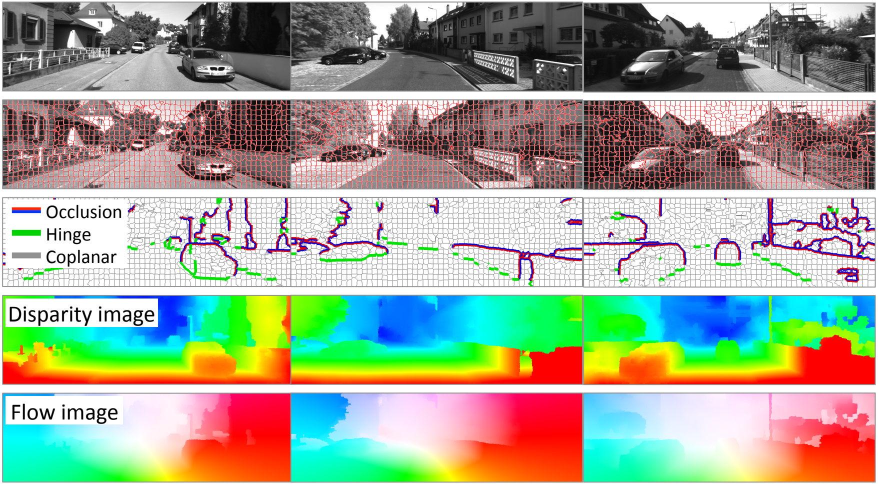
K. Yamaguchi, D. McAllester, R. Urtasun, *Efficient Joint Segmentation, Occlusion Labeling, Stereo and Flow Estimation*, ECCV 2014

Paper: `http://www.cs.toronto.edu/~urtasun/publications/yamaguchi_et_al_eccv14.pdf`

Code: `http://ttic.uchicago.edu/~dmcallester/SPS/index.html`

You Can Do It Much Better...

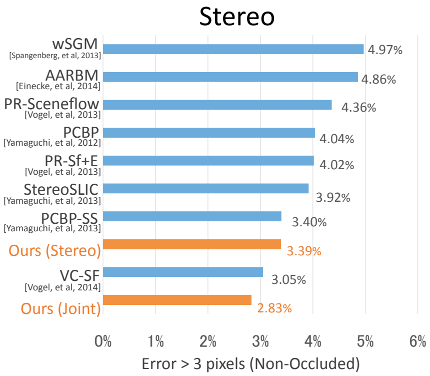
[K. Yamaguchi, D. McAllester and R. Urtasun, ECCV 2014]

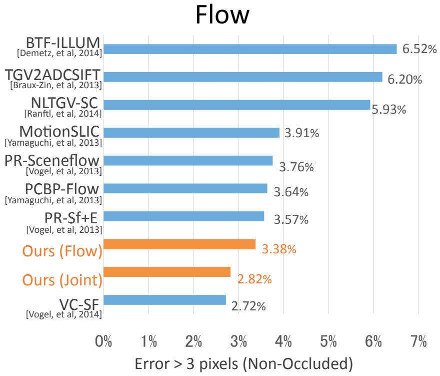


Look at State-of-the-art on KITTI

Where "Ours" means: [K. Yamaguchi, D. McAllester and R. Urtasun, ECCV 2014]

- How can we evaluate the performance of a stereo algorithm?





- Autonomous driving dataset KITTI: `http://www.cvlibs.net/datasets/kitti/`

From Disparity We Get...

- Depth: Once you have disparity, you have 3D

Figure: K. Yamaguchi, D. McAllester and R. Urtasun, ECCV 2014

From Disparity We Get...

- Money ;)

Epipolar geometry

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