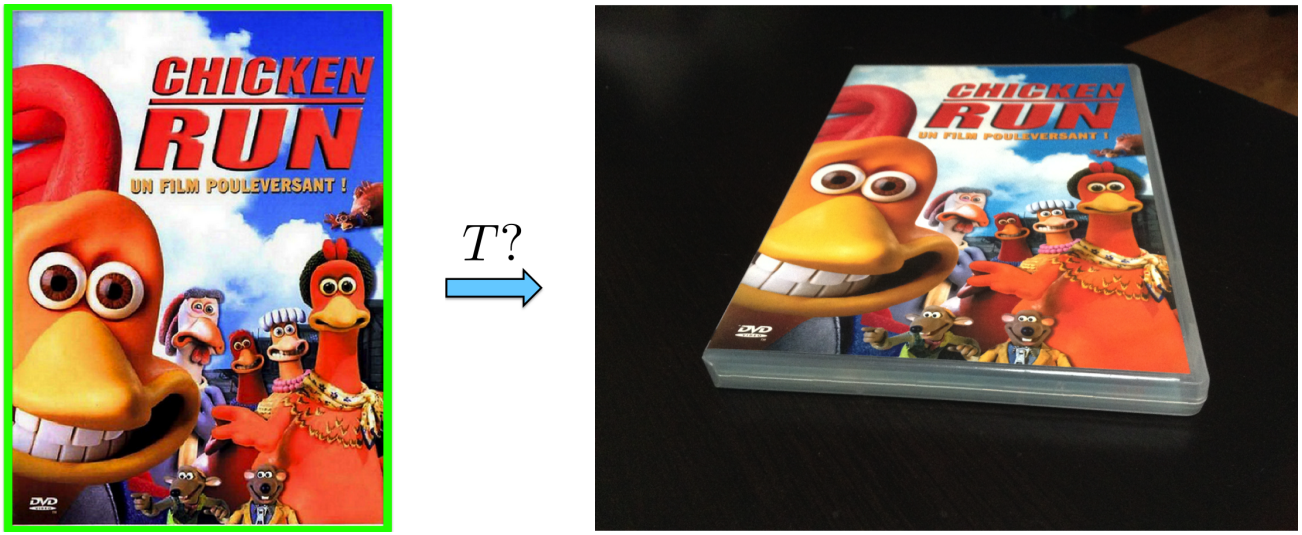
Back to the Homography: The Why

Homography



- 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.

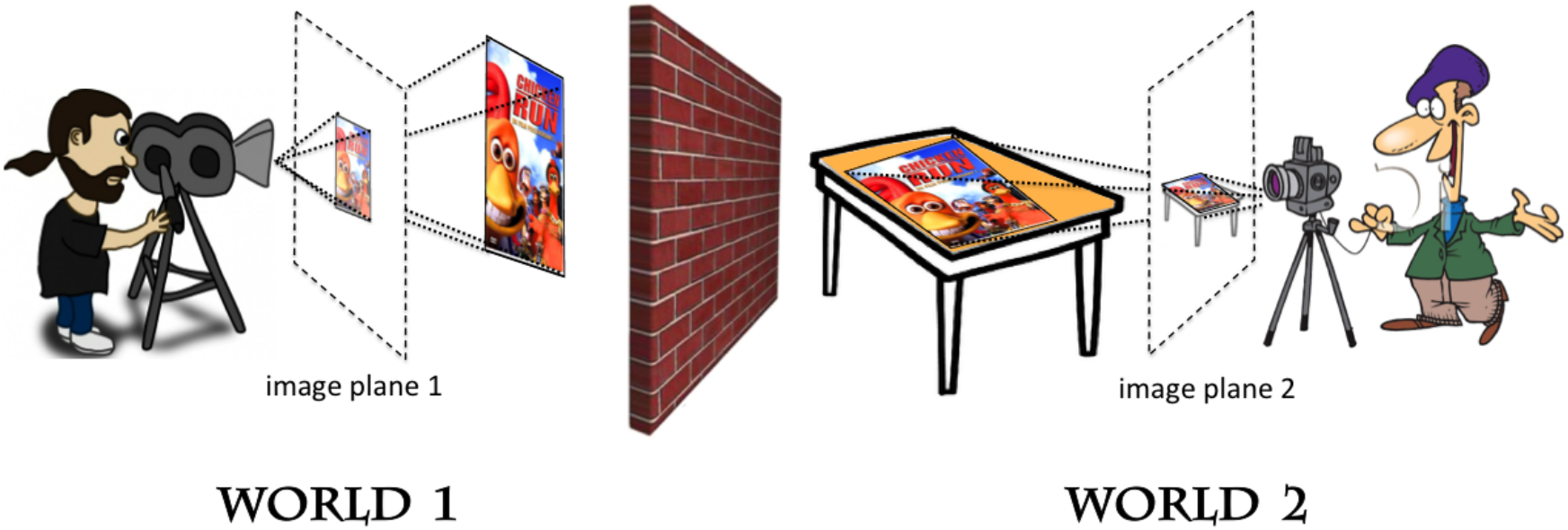


Figure: We have our object in two different worlds, in two different poses relative to camera, two different photographers, and two different cameras.

Homography

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

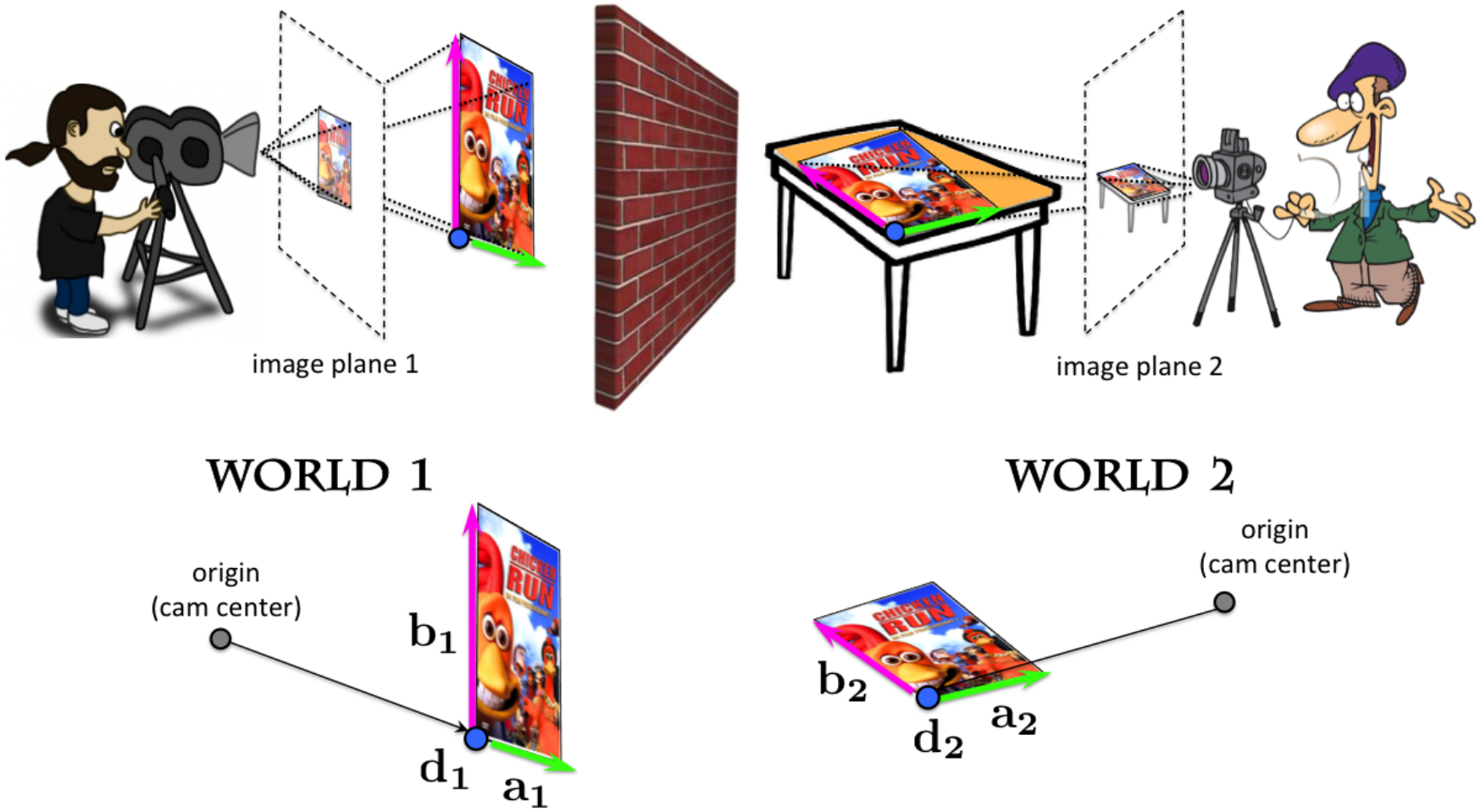


Figure: Our object is a plane. Each plane is characterized by one point **d** on the plane and two independent vectors **a** and **b** on the plane.

Homography

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

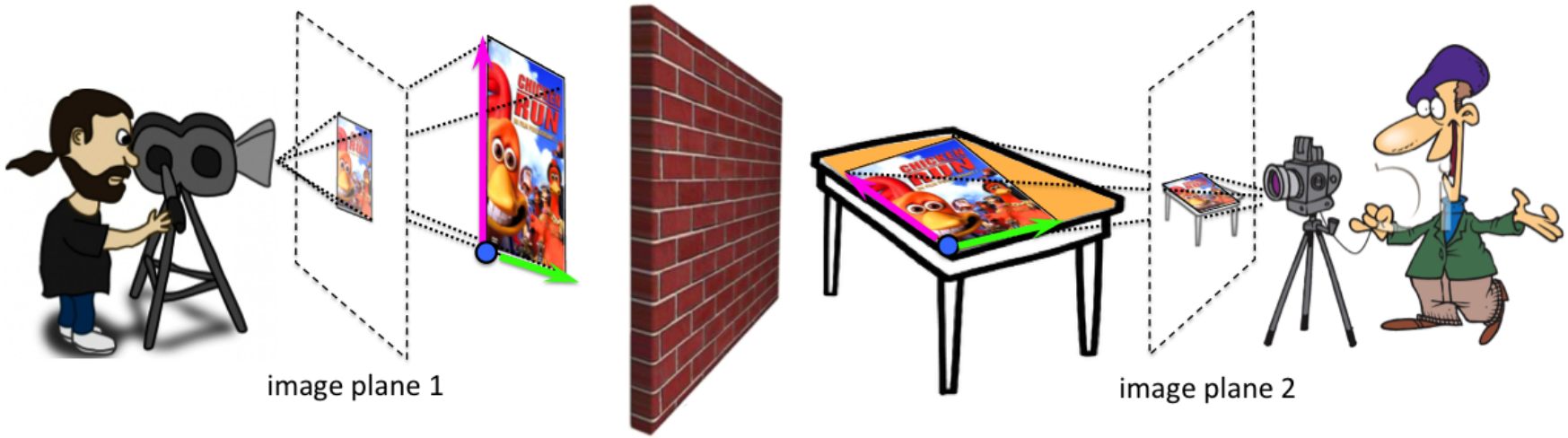


Figure: Then any other point $\mathbf{X}$ on the plane can be written as: $\mathbf{X} = \mathbf{d} + \alpha\mathbf{a} + \beta\mathbf{b}$.

Homography

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

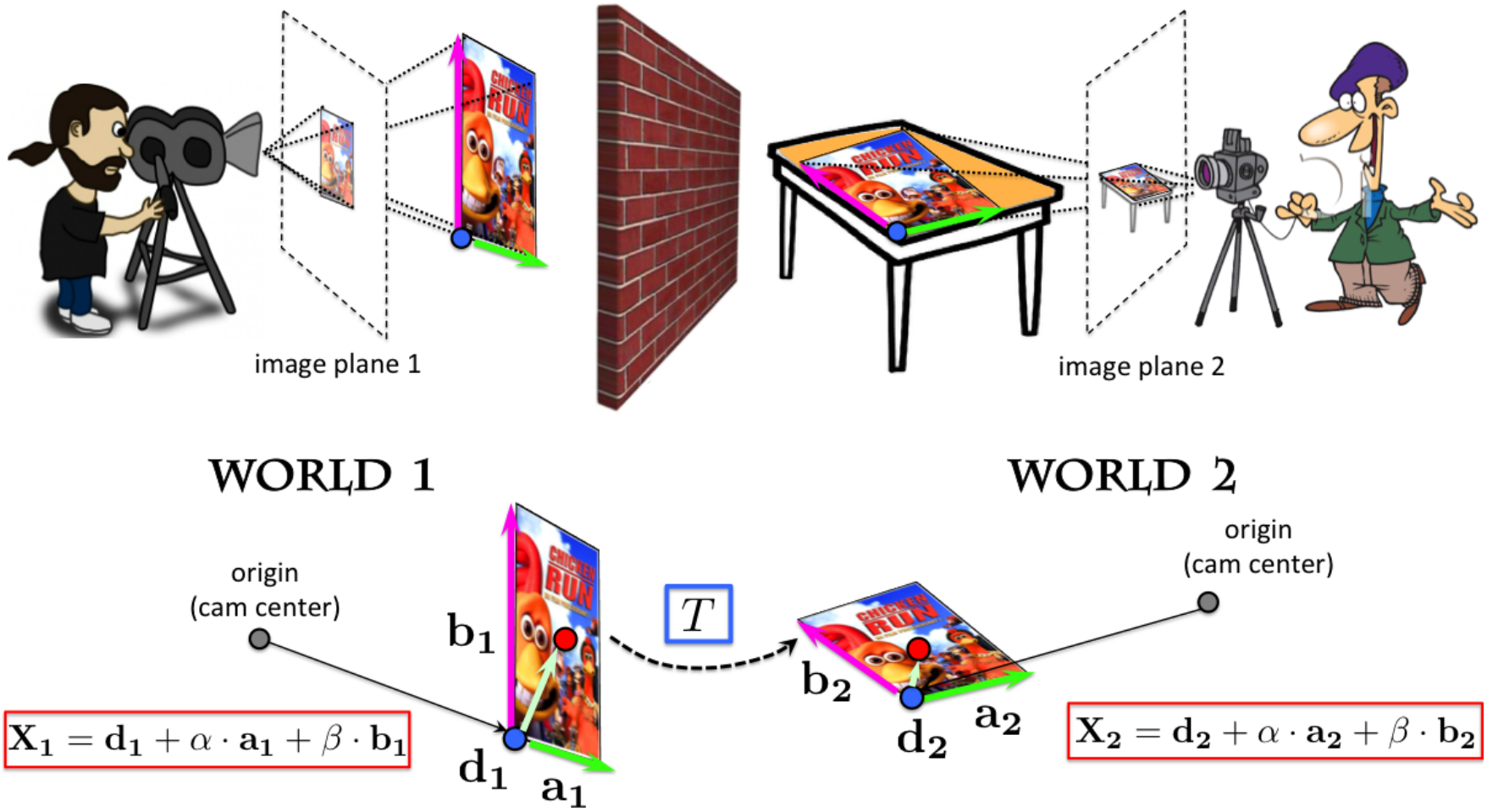


Figure: Any two Chicken Run DVDs on our planet are related by some transformation T. We'll compute it, don't worry.

Homography

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

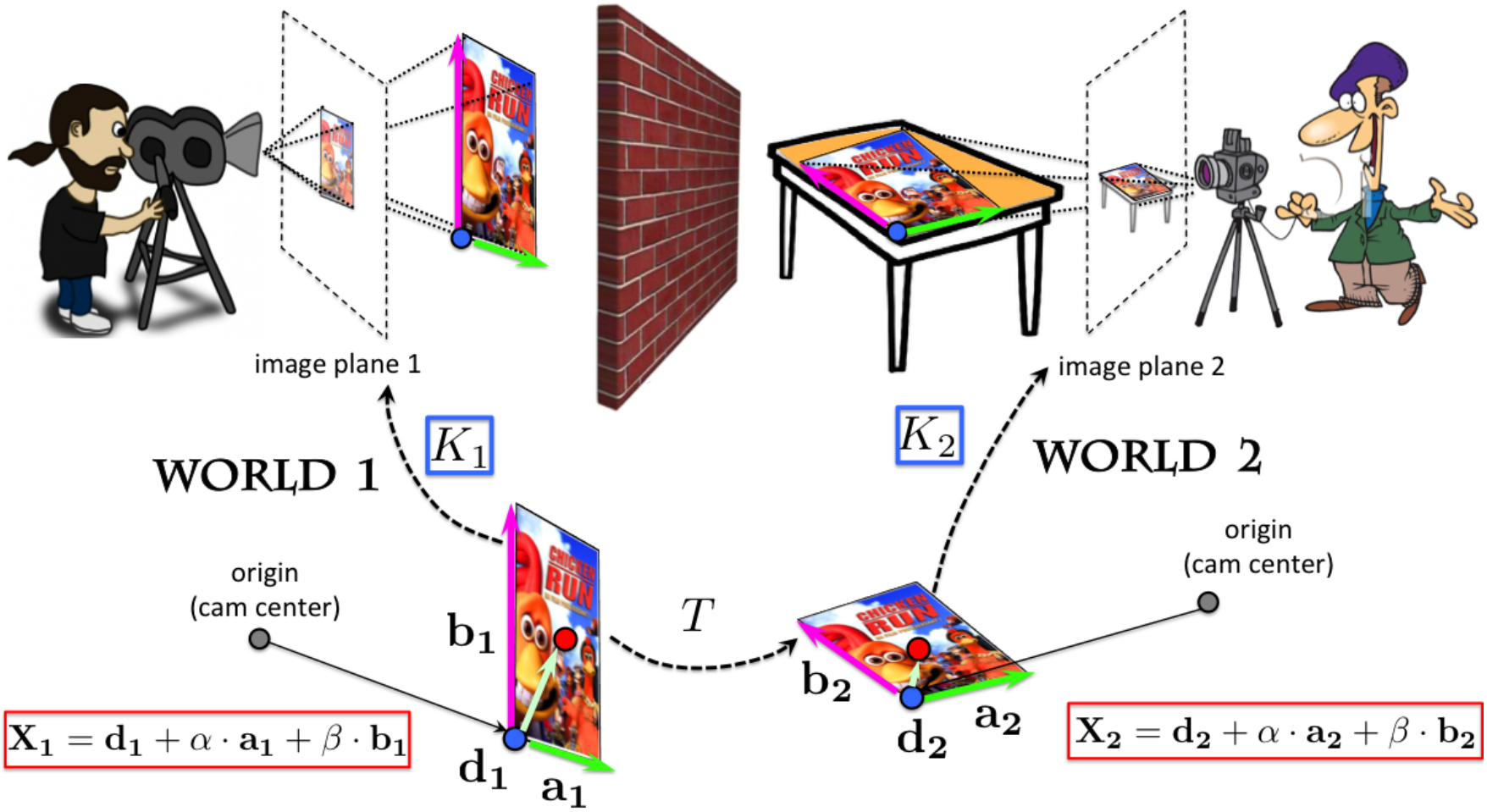


Figure: Each object is seen by a different camera and thus projects to the corresponding image plane with different camera intrinsics.

Homography

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

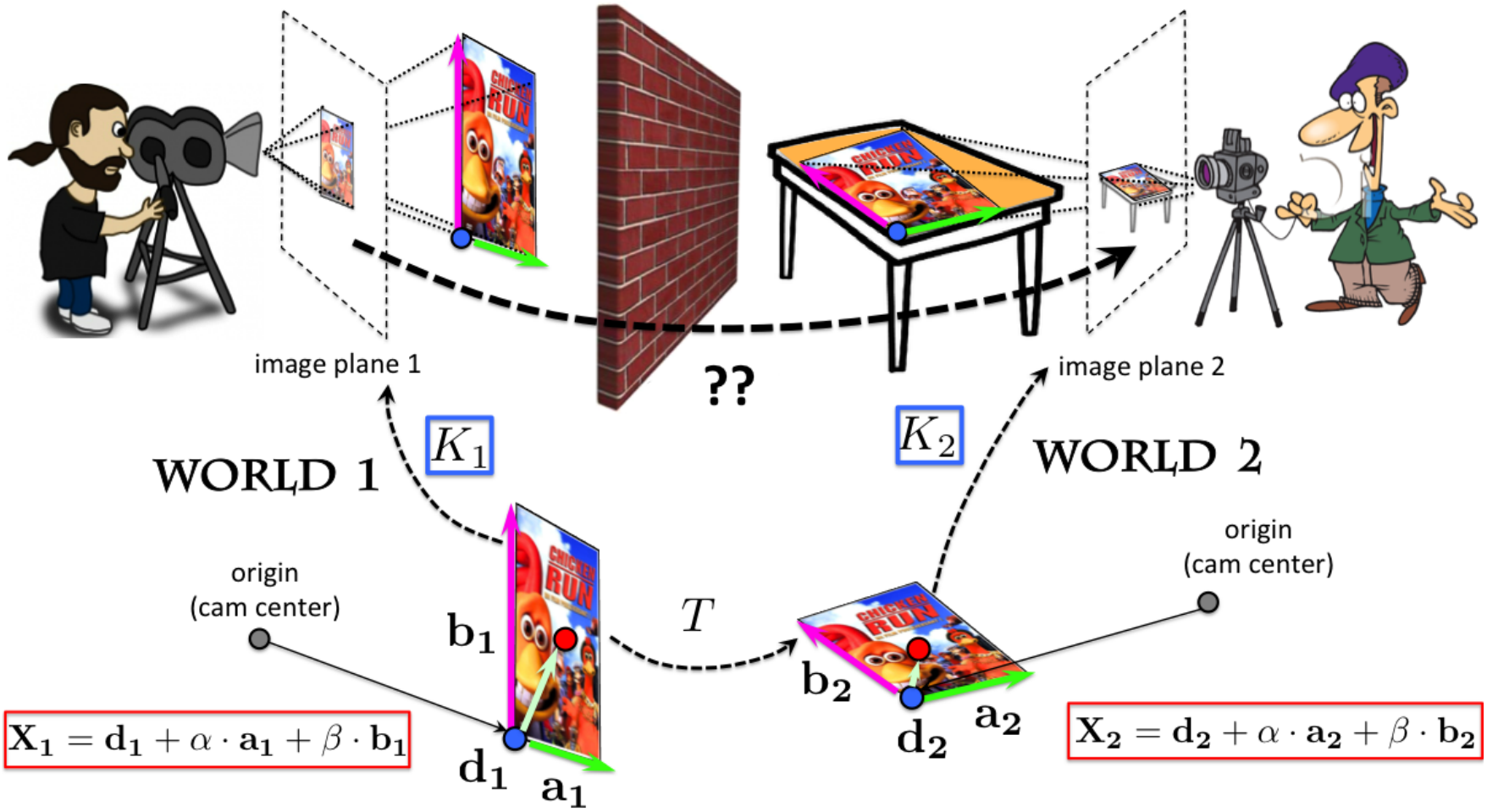


Figure: Given this, the question is what's the transformation that maps the DVD on the first image to the DVD in the second image?

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:

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

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

- We can rewrite this using homogeneous coordinates:

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

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

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:

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

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

- We can rewrite this using homogeneous coordinates:

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

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

- 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:

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

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

- We can rewrite this using homogeneous coordinates:

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

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

- 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:

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

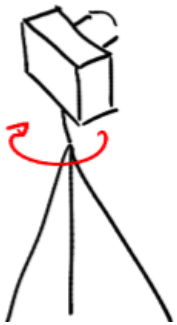
Homography

- 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.

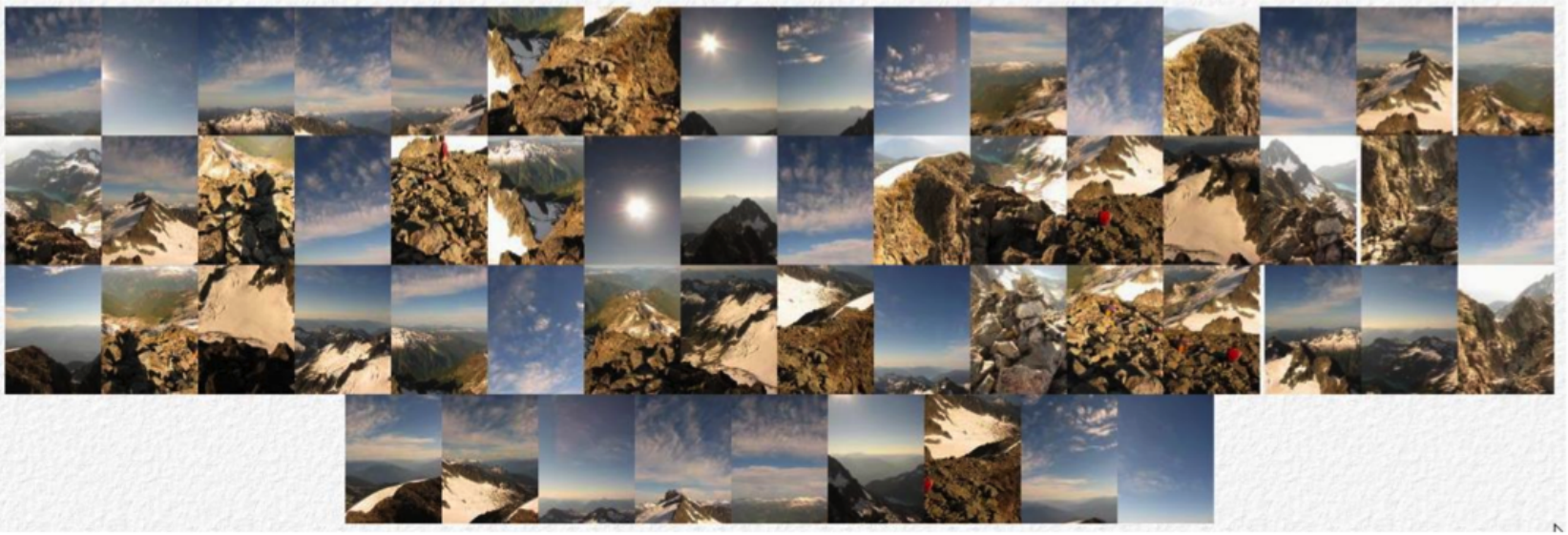
Remember Panorama Stitching from Lecture 9?

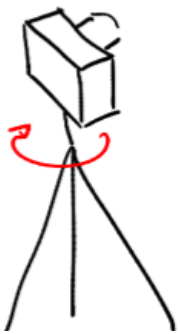


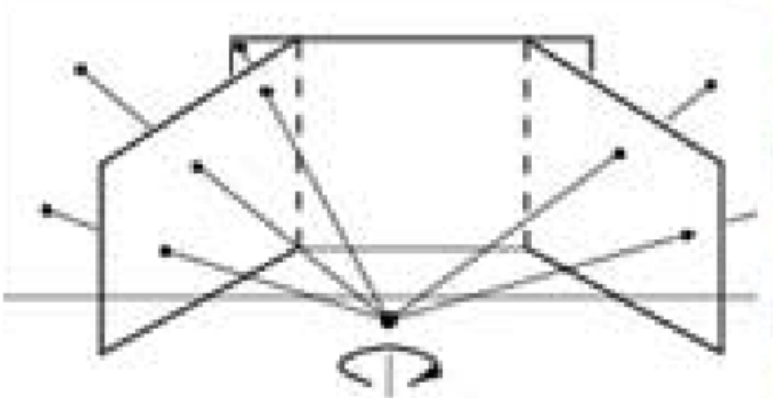
Take a tripod, rotate camera and take pictures

[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 $T = R$:

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

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

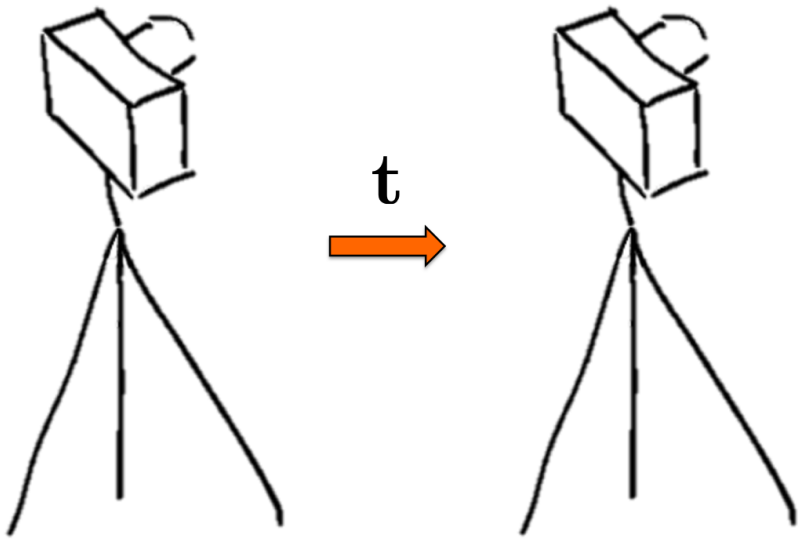
- And this is a homography

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

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

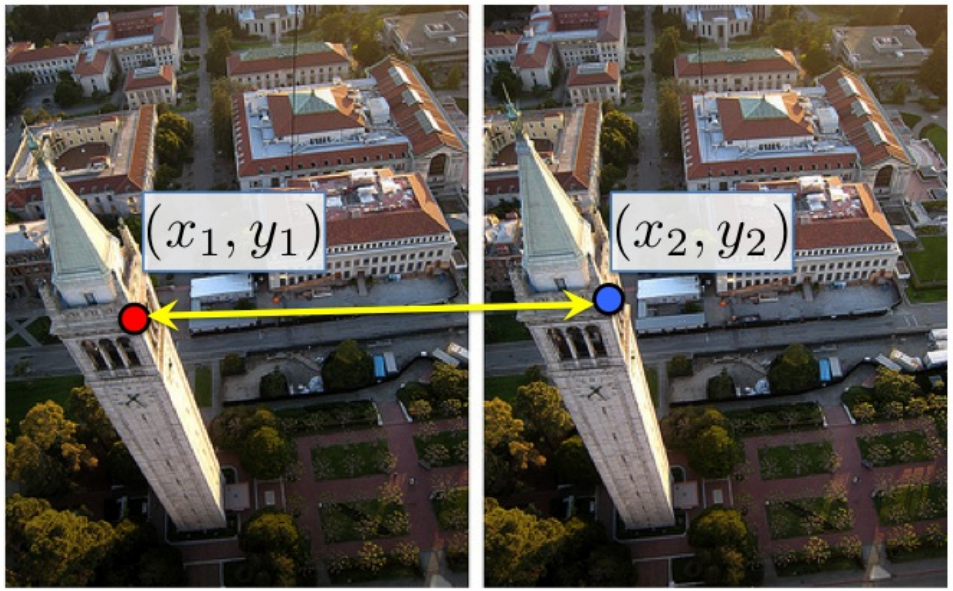
  we know that different w_1 mean different points $\mathbf{X_1}$ on the projective line
- Where (x_1, y_1) maps to in the 2nd image depends on the 3D location of $\mathbf{X_1}$!

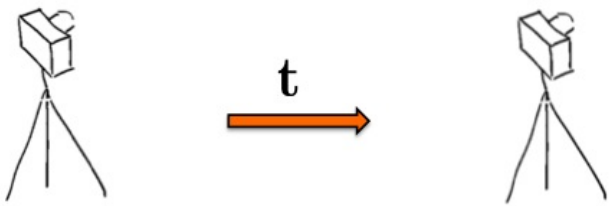
What If I Move the Camera?

- **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?



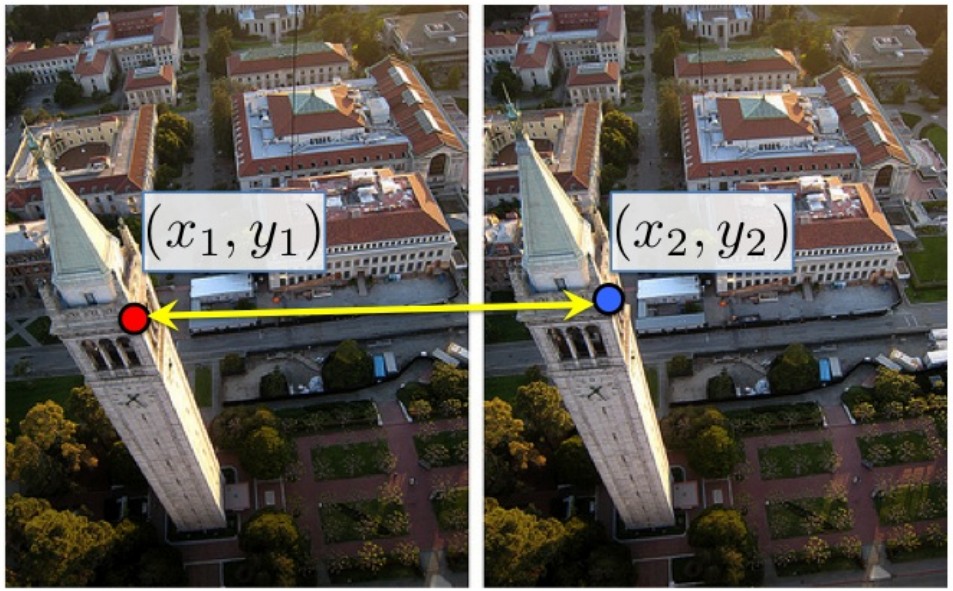


$$w_2 \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = w_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} - K\mathbf{t}$$

We know this

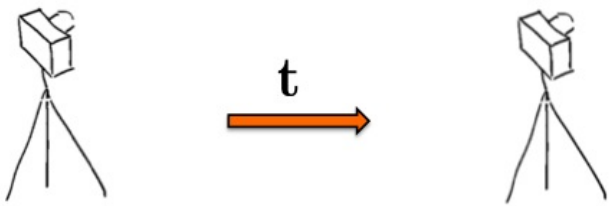
What If I Move the Camera?

- **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:

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

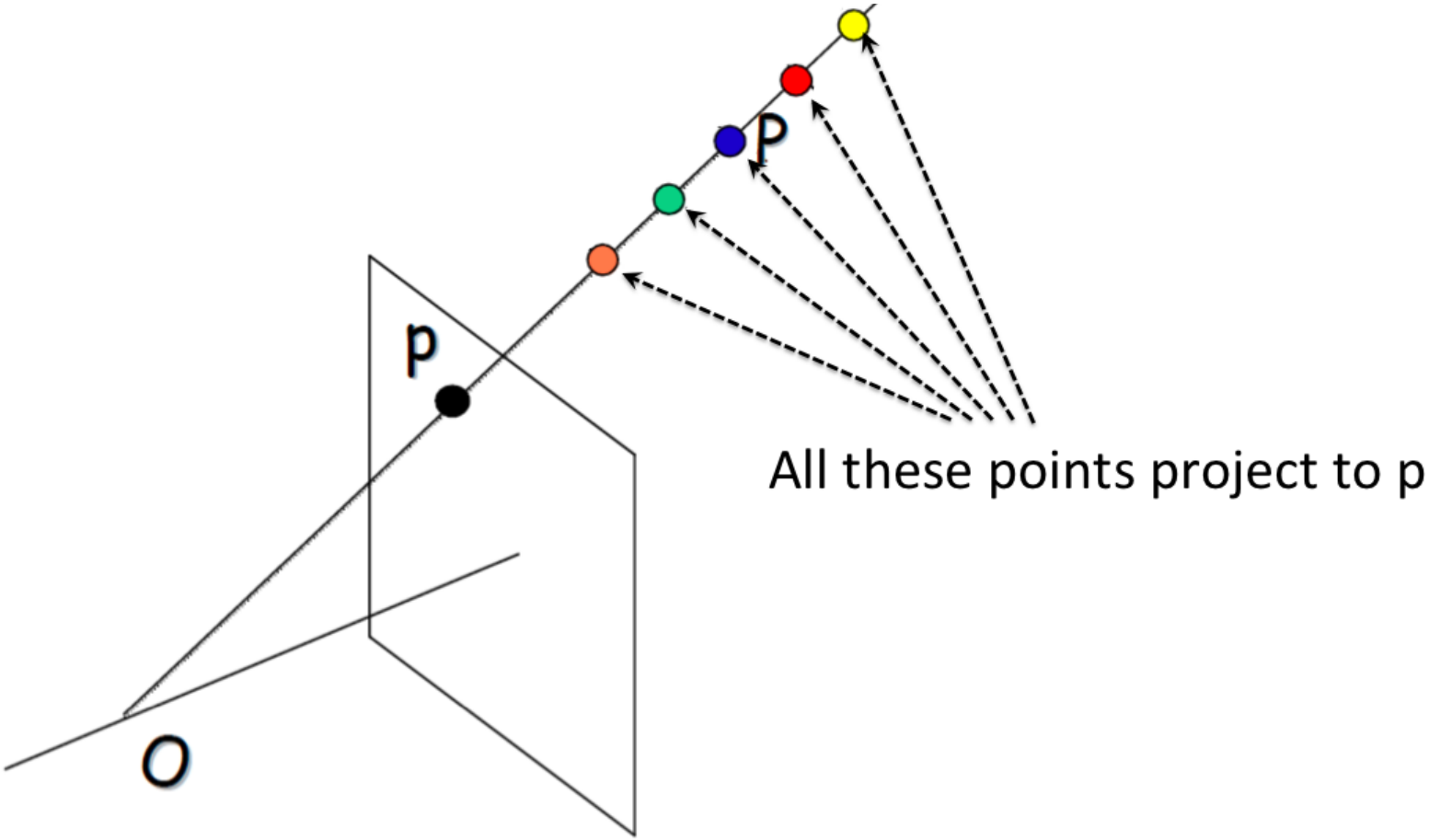
- Parallel lines in 3D are parallel in the image

Depth

Getting the 3D Scene Behind the Image

Depth from a Monocular Image

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



[Pic adopted from: J. Hays]

Depth from a Monocular Image

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

[Pic from: S. Lazebnik]

Depth from a Monocular Image

Nothing is impossible!

Depth from a Monocular Image

- When present, we can use certain cues to get depth (3D) from one image
- Can you come up with (at least) 8 ways of getting depth from a single image?

Depth from a Monocular Image

1

Shape from Shading

Depth from a Monocular Image

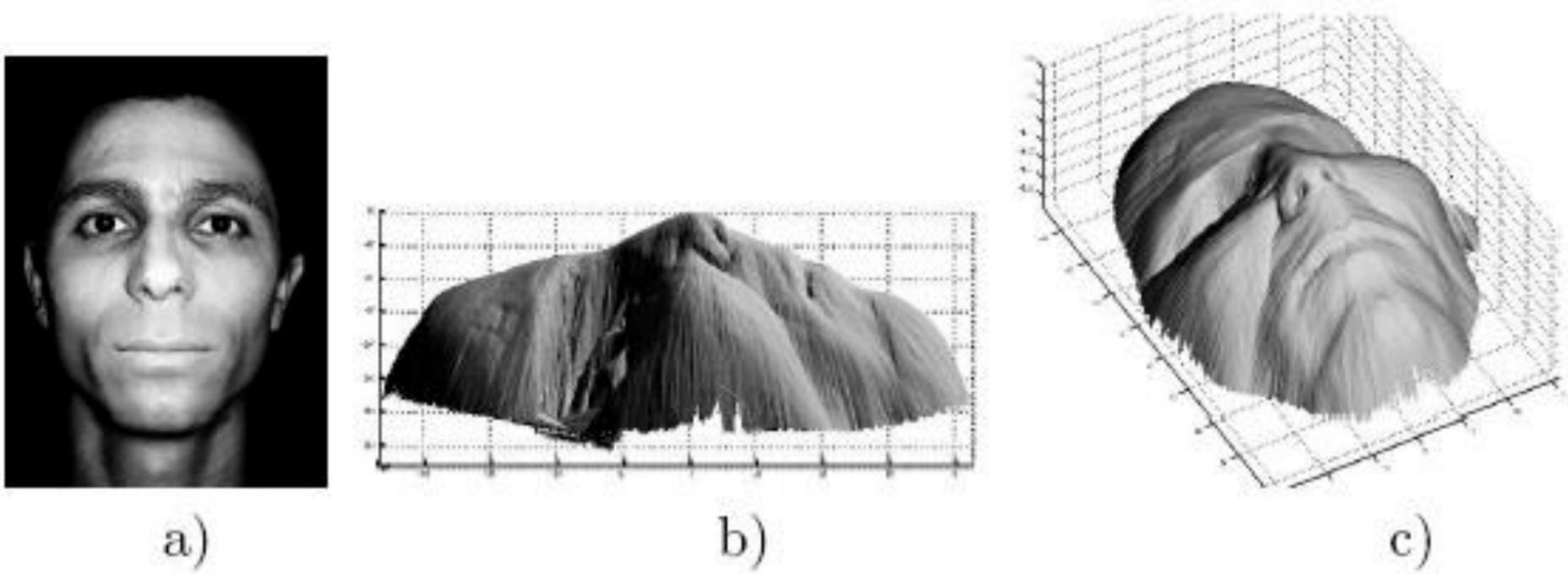


Figure: Shape from Shading

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

②

Shape from Texture

Depth from a Monocular Image

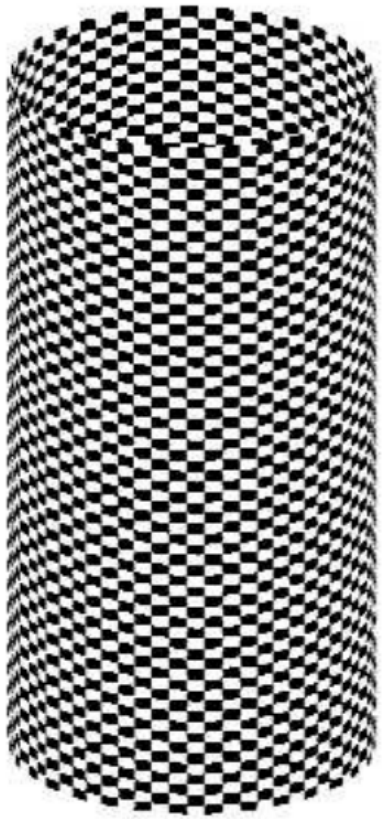
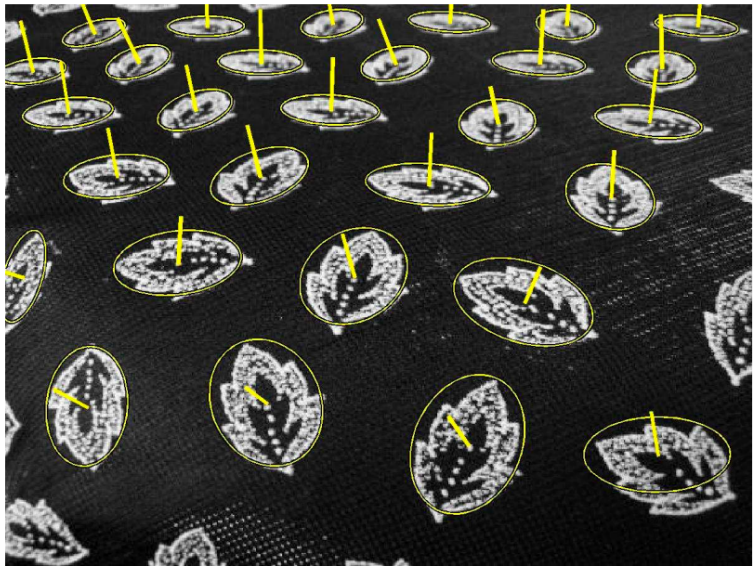
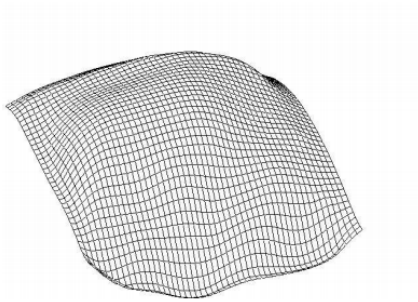


Figure: Shape from Texture: What do you see in the image?

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

Depth from a Monocular Image





(a) Estimated surface shape

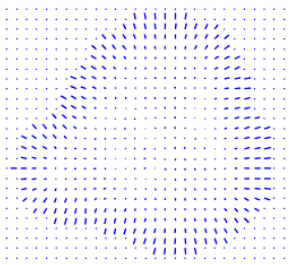
(b) Texture projected onto surface

Figure: Shape from Texture

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

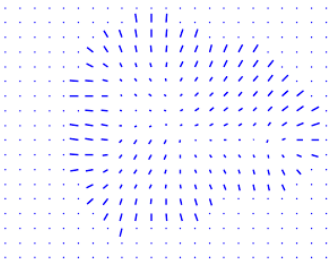
Depth from a Monocular Image

(a) Segmented image

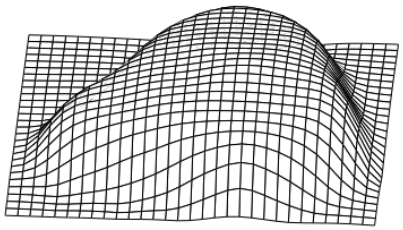


(b) Needle diagram

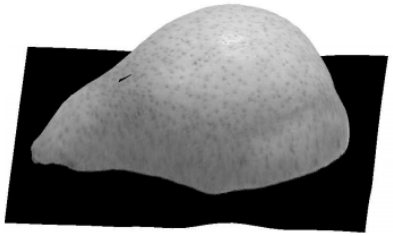
(e) Segmented image



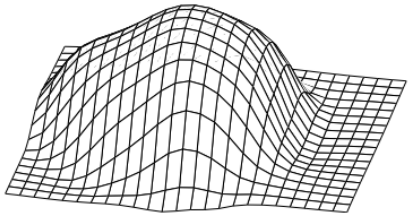
(f) Needle diagram



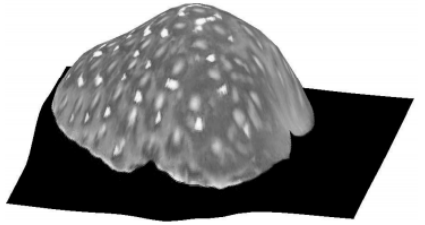
(c) Mesh surface



(d) New view of pear



(g) Mesh surface



(h) New view of strawberry

Figure: Shape from Texture

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

Depth from a Monocular Image

Waterlilies:
non-homogeneous

Donut:
non-stationary

Woven lamp:
anisotropic

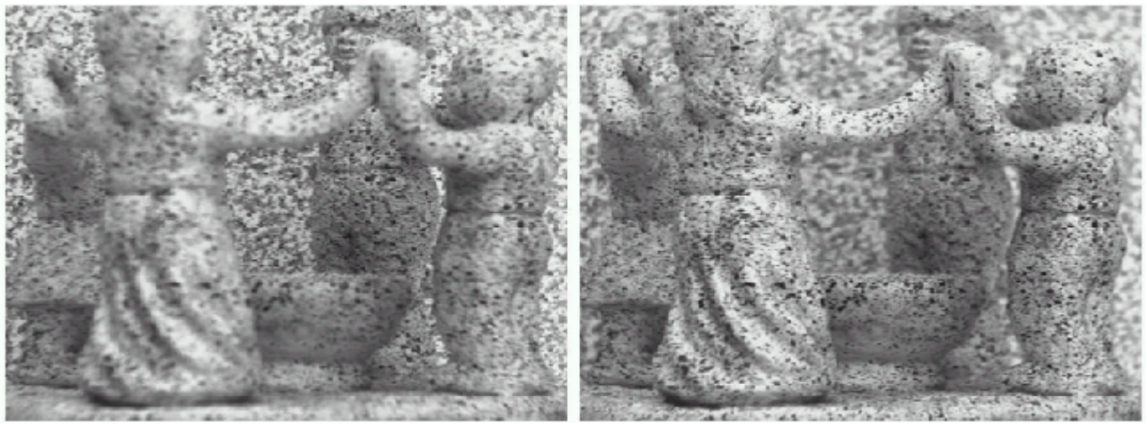
Figure: Shape from Texture: And quite a lot of stuff around us is textured

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

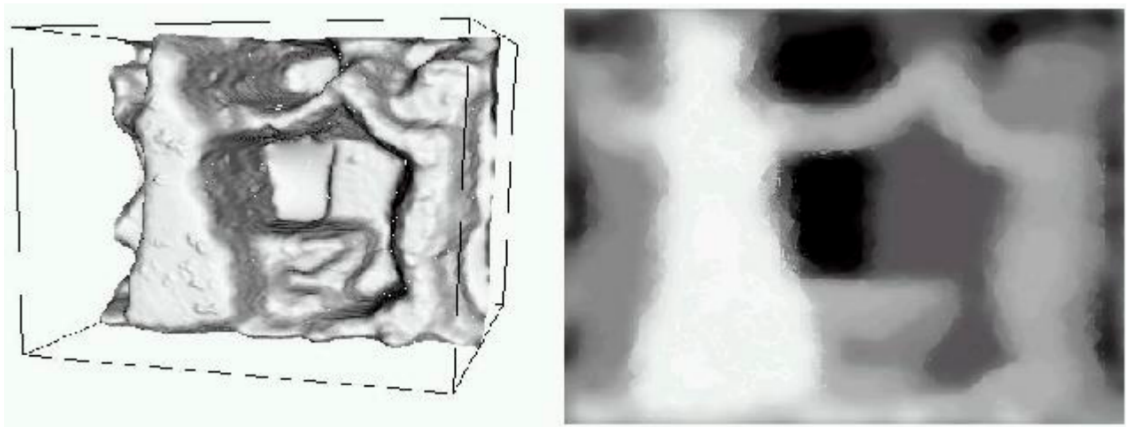
3

Shape from Focus/De-focus

Depth from a Monocular Image



Images from same point of view, different camera parameters



3d shape / depth estimates

Figure: Shape from Focus/De-focus

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

4

Depth Ordering via Occlusion Reasoning

Depth from a Monocular Image

Depth from a Monocular Image

Y and T junctions are great indicators of occlusion

Depth from a Monocular Image

Y and T junctions are great indicators of occlusion

Depth from a Monocular Image

Non-occluding pumpkins are a problem

Depth from a Monocular Image

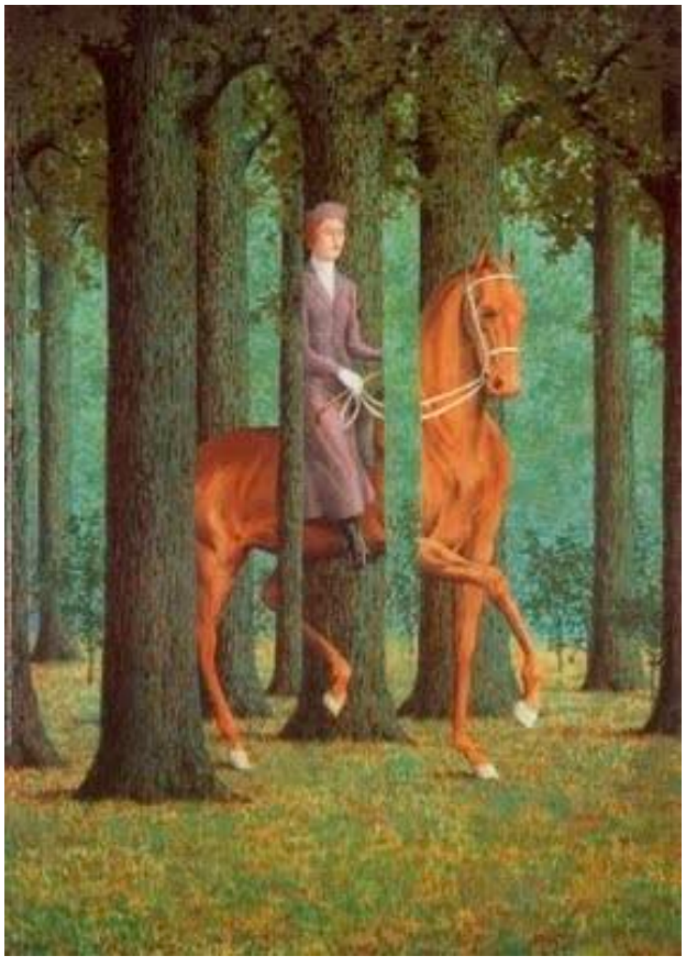


Figure: Occlusion gives us **ordering** in depth

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

5

Depth from Google

Depth from a Monocular Image

Figure: We go to Italy and take this picture.

[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 a Monocular Image

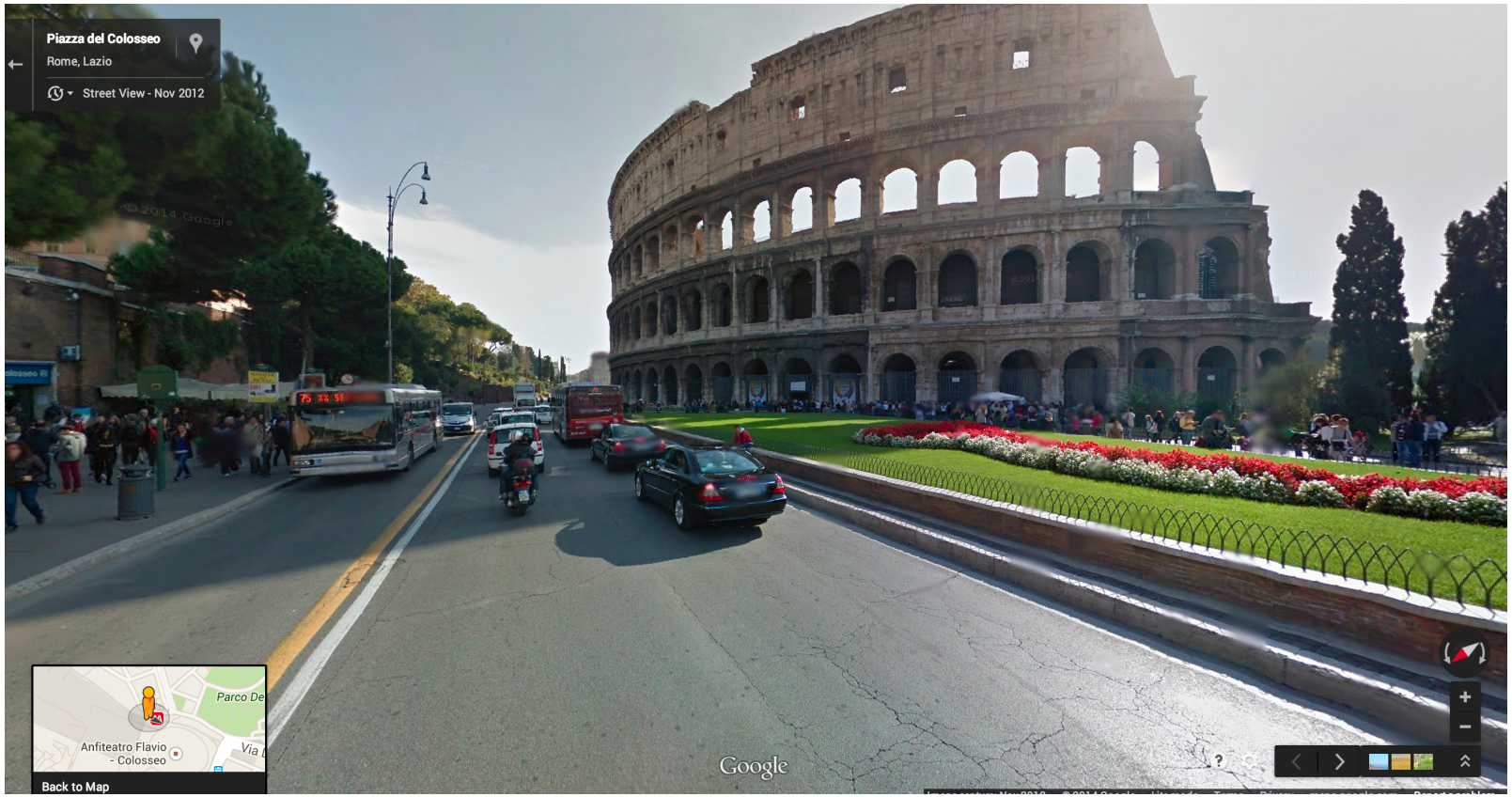


Figure: We can match it to Google Street View (compute accurate location and viewing angle). See paper below.

[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 a Monocular Image

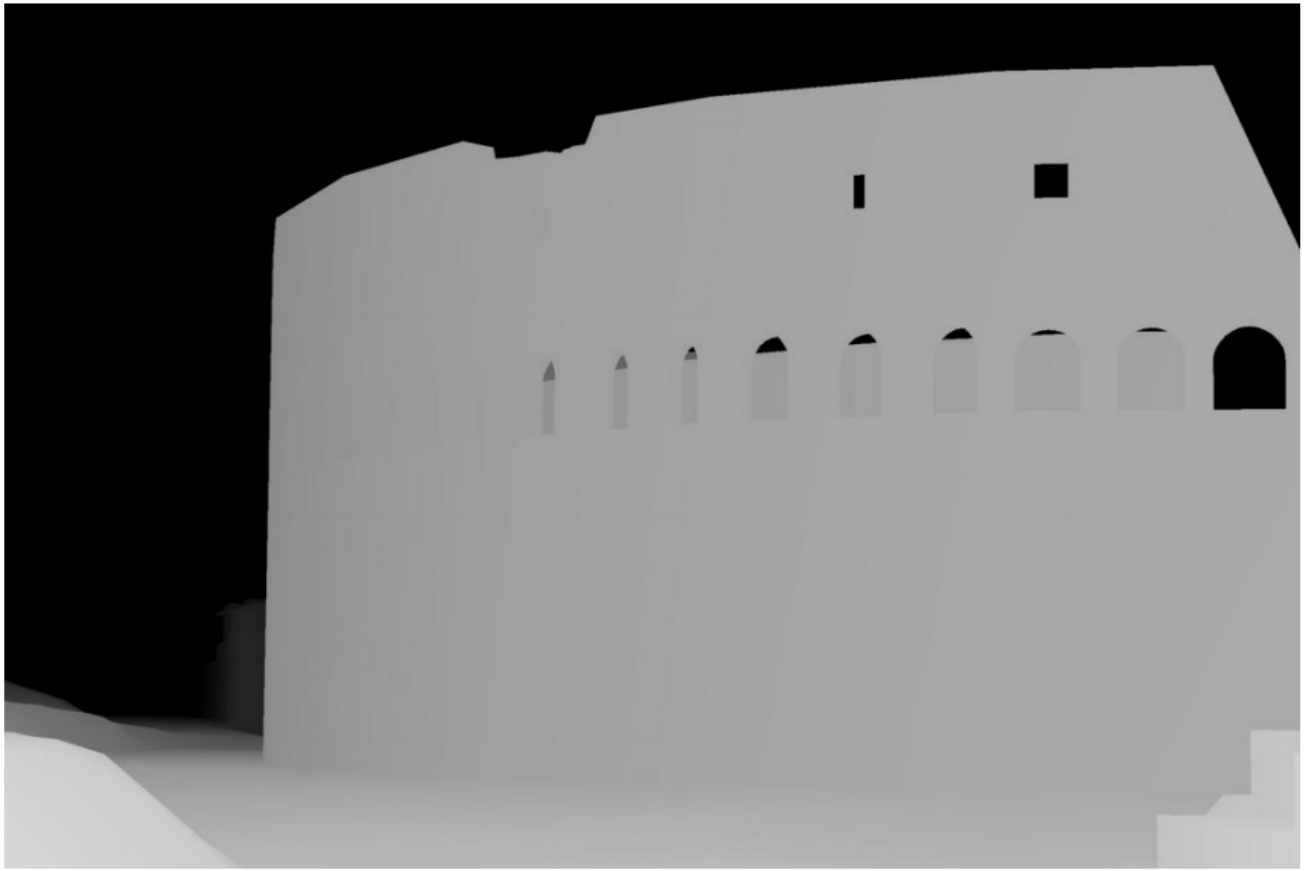


Figure: Depth from Google: "Borrow" depth from Google's Street View Z-buffer.

[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 a Monocular 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*

Depth from a Monocular Image

Figure: Depth from Google: "Borrow" depth from Google's Street View Z-buffer

http://inear.se/urbanjungle/

Depth from a Monocular Image

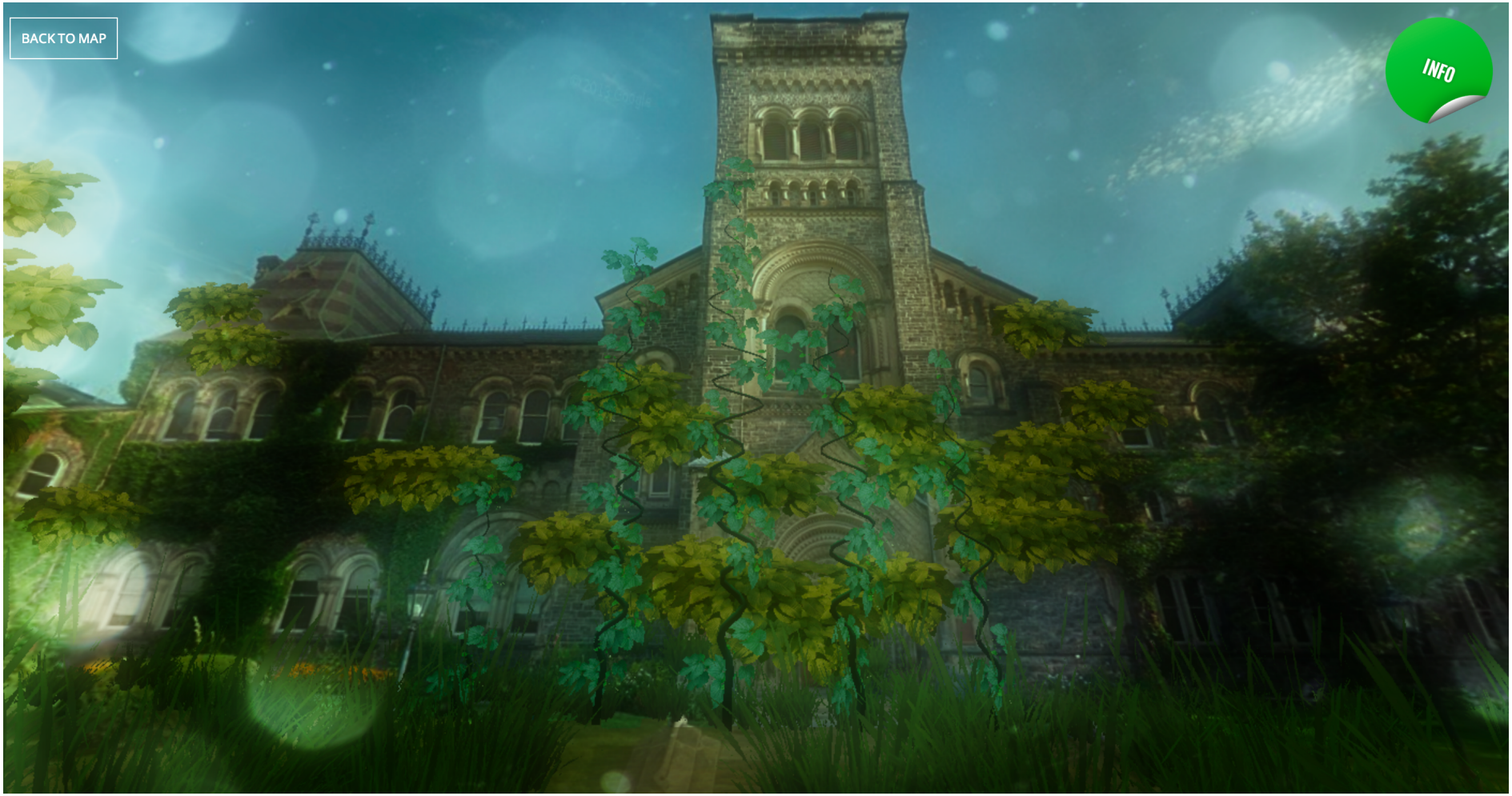


Figure: Depth from Google: Once you have depth you can render cool stuff

http://inear.se/urbanjungle/

Depth from a Monocular Image

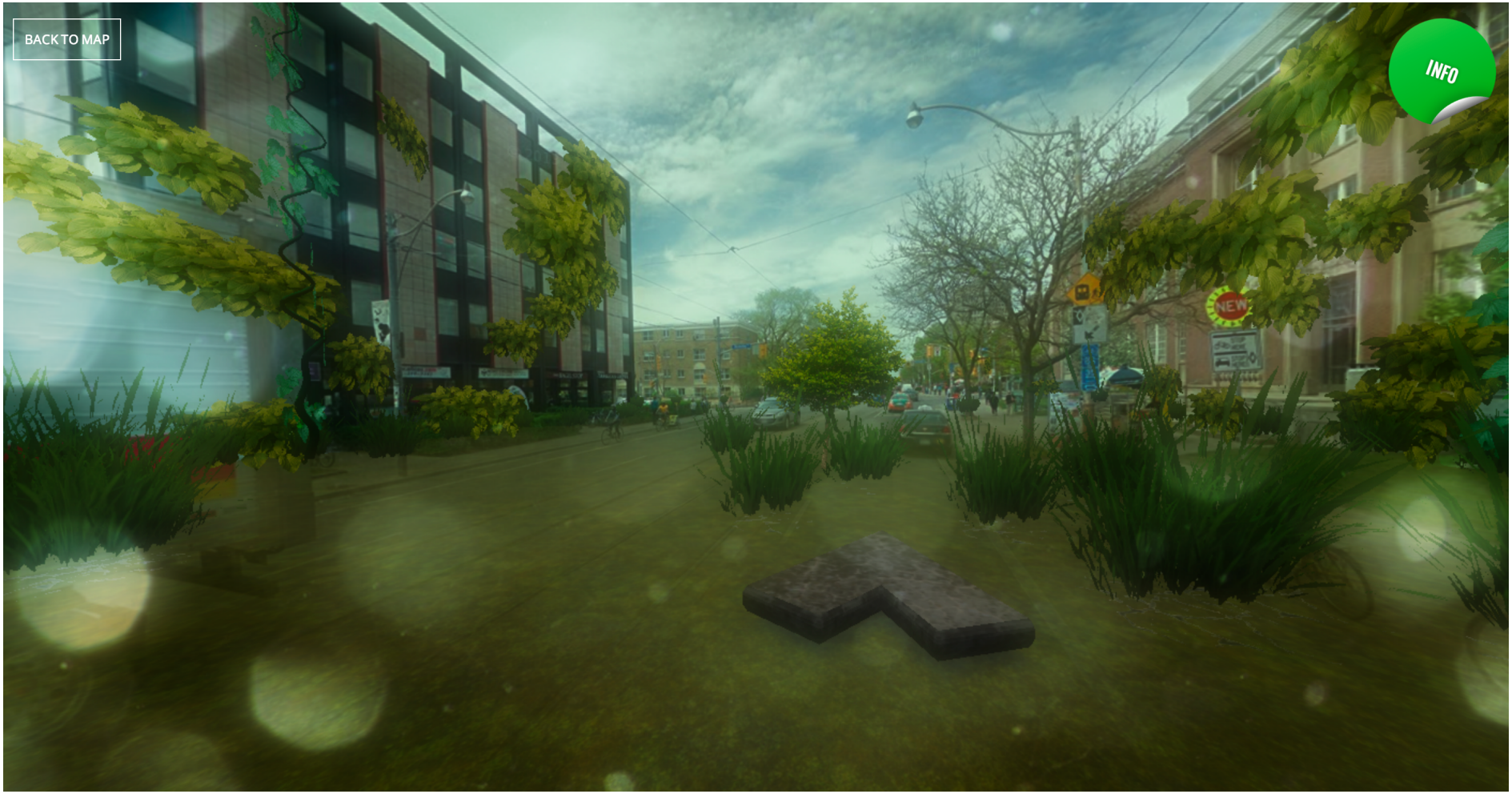


Figure: Depth from Google: Recognize this?

http://inear.se/urbanjungle/

6

Depth via Recognition

Depth from a Monocular Image

- Get 3D models of objects (lots available online, e.g. Google 3D Warehouse)

Figure: CAD models of IKEA furniture from Lim et al.

[Joseph J. Lim, Hamed Pirsiavash, Antonio Torralba. Parsing IKEA Objects: Fine Pose Estimation. ICCV'13]

Depth from a Monocular Image

- Match (align) 3D models with image (estimate the projection matrix P)

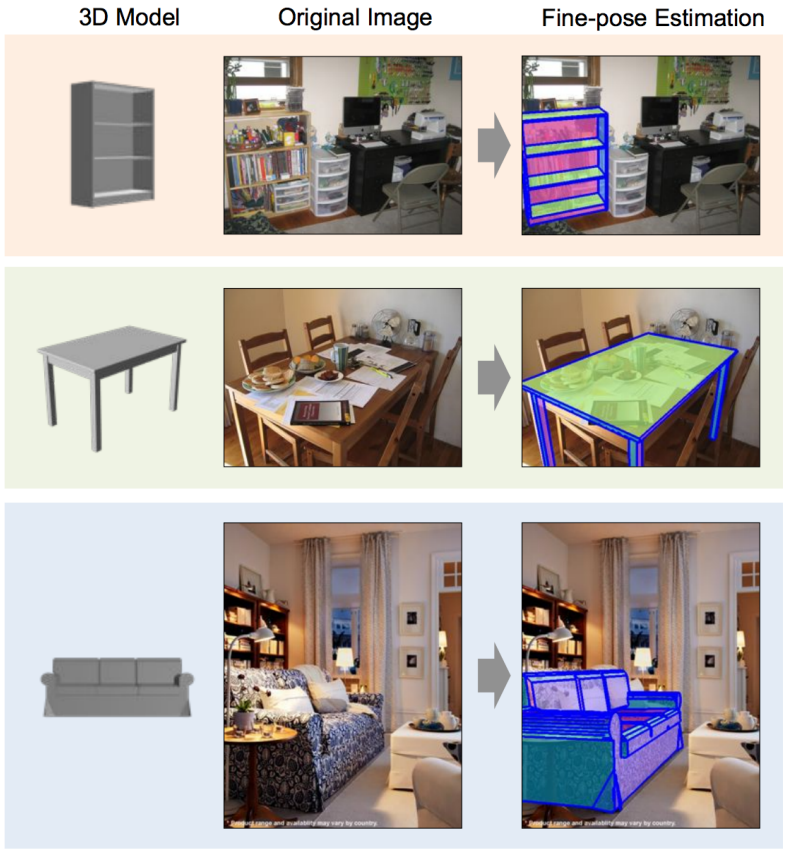


Figure: Match CAD models to image

[Joseph J. Lim, Hamed Pirsiavash, Antonio Torralba. Parsing IKEA Objects: Fine Pose Estimation. ICCV'13]

Depth from a Monocular Image

- Match (align) 3D models with image (estimate the projection matrix P)

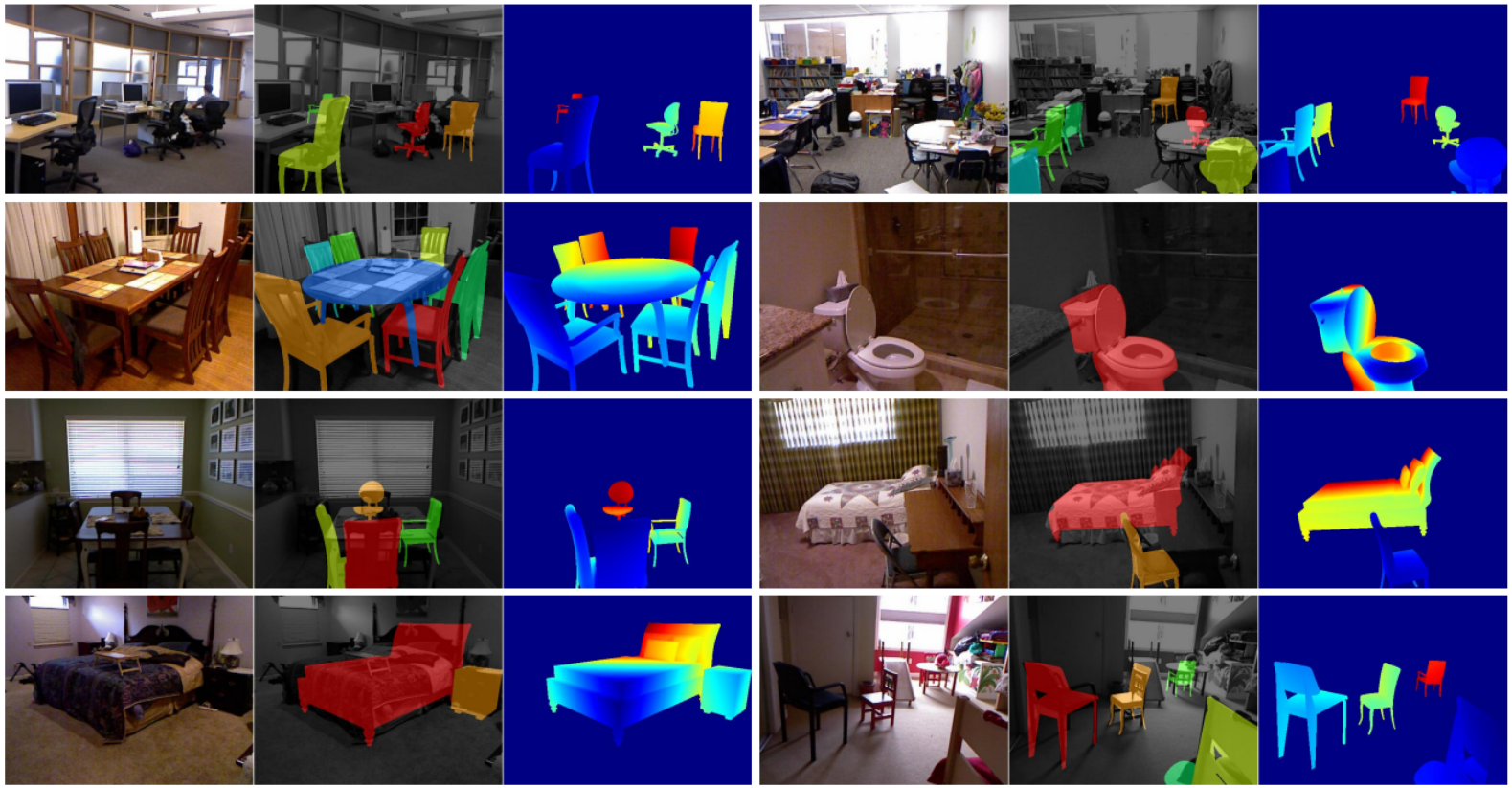


Figure: Render depth from the CAD model.

[Saurabh Gupta, Pablo Arbelaez, Ross Girshick, Jitendra Malik. Aligning 3D Models to RGB-D Images of Cluttered Scenes. CVPR'15]

7

Depth via Machine Learning

Depth from a Monocular Image

- Collect training data: for example RGB-D data acquired by Kinect
- Train classifiers/regressors

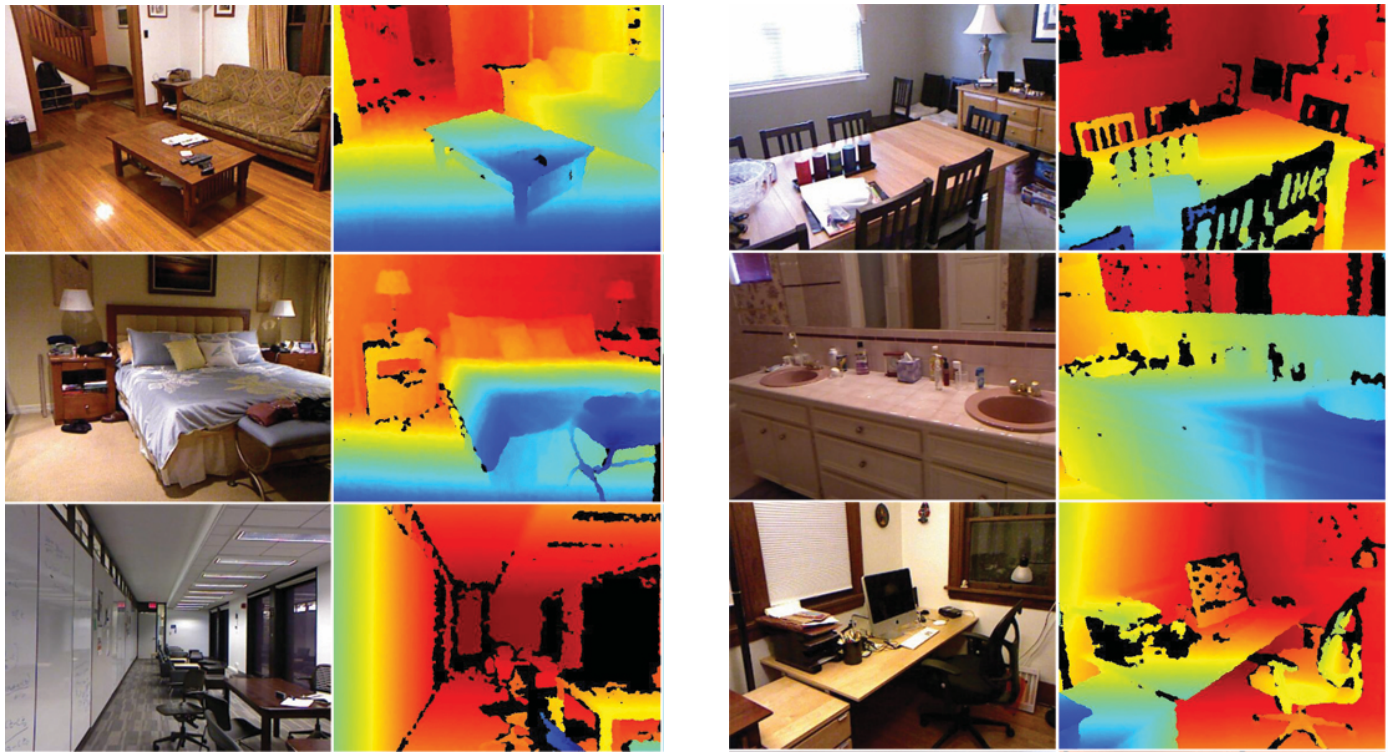


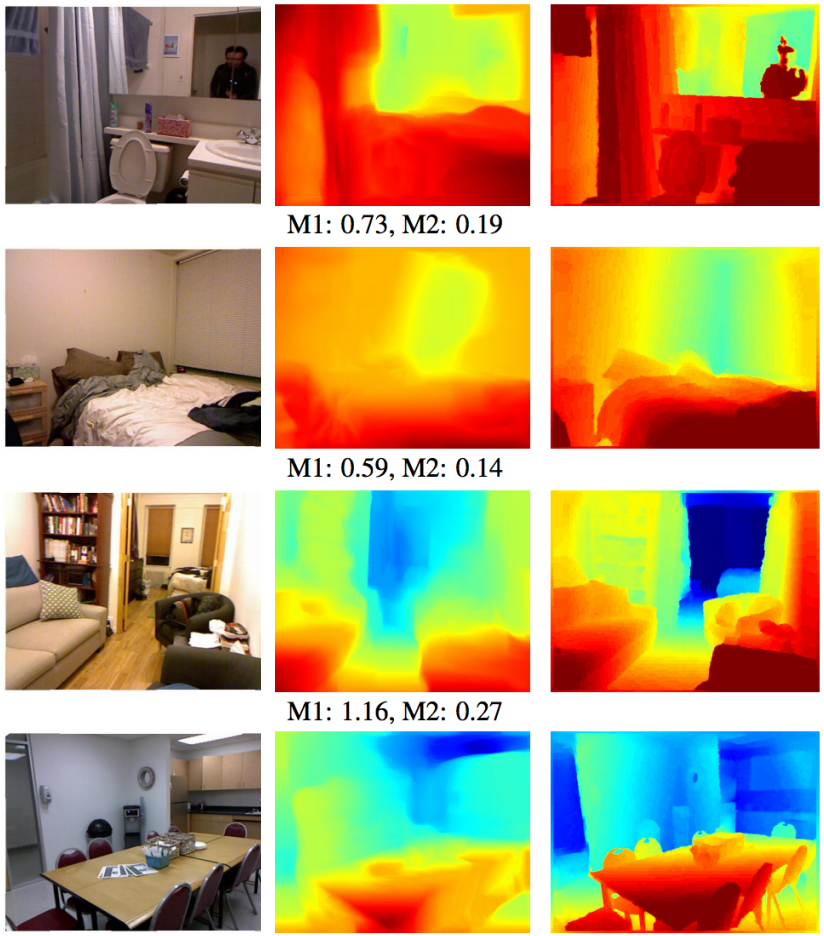
Figure: The NYUv2 dataset: RGB-D images collected with Kinect

[Nathan Silberman, Pushmeet Kohli, Derek Hoiem, Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. ECCV'12.]

http://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

Depth from a Monocular Image

(Red color: pixels closer to the camera, blue: pixels further away from camera.)



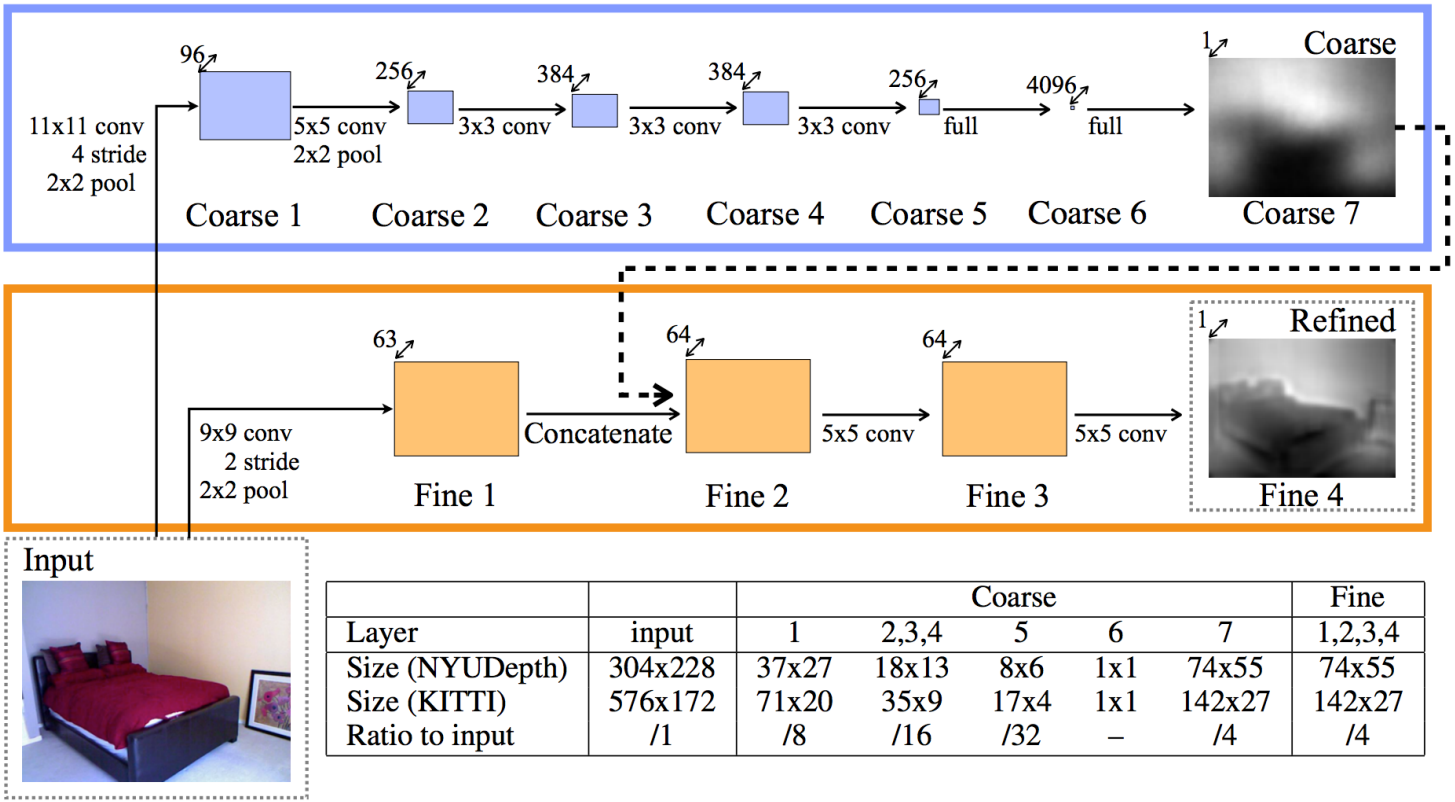
(a) image (b) predicted depth (c) ground-truth depth

Figure: Obtain impressive results.

[Christian Hane, L'ubor Ladicky, Marc Pollefeys. Direction Matters: Depth Estimation with a Surface Normal Classifier. CVPR'15]

Depth from a Monocular Image

- Train Convolutional Neural Nets (CNNs)



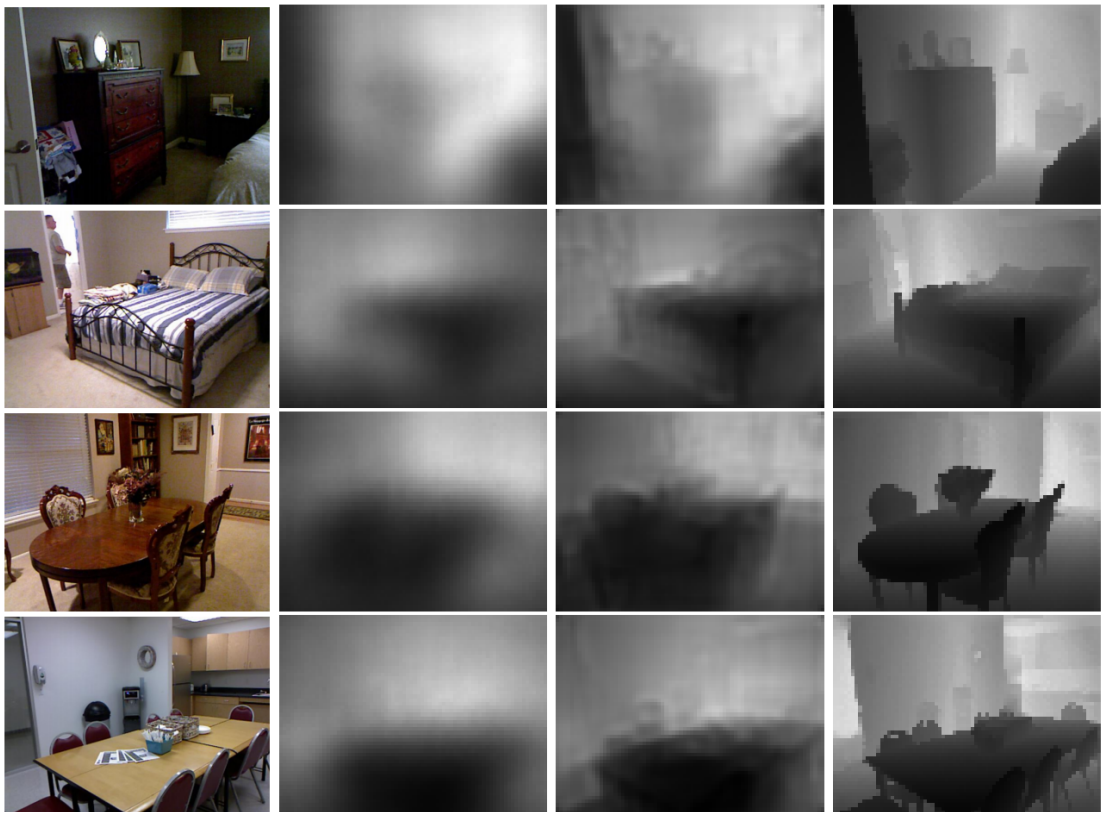
		Coarse					Fine
Layer	input	1	2,3,4	5	6	7	1,2,3,4
Size (NYUDepth)	304x228	37x27	18x13	8x6	1x1	74x55	74x55
Size (KITTI)	576x172	71x20	35x9	17x4	1x1	142x27	142x27
Ratio to input	/1	/8	/16	/32	–	/4	/4

Figure: CNN architecture from Eigen et al.

[David Eigen, Christian Puhrsch, Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. NIPS'14]
Code: http://www.cs.nyu.edu/~deigen/depth/

Depth from a Monocular Image

- Train Convolutional Neural Nets (CNNs)



(a) image (b) predicted depth (c) ground-truth depth

Figure: Results from Eigen et al.

[David Eigen, Christian Puhrsch, Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. NIPS'14]
Code: http://www.cs.nyu.edu/~deigen/depth/

Depth from a Monocular Image

- Train Convolutional Neural Nets (CNNs) to predict **surface normals** instead of **depth** (works very well!)

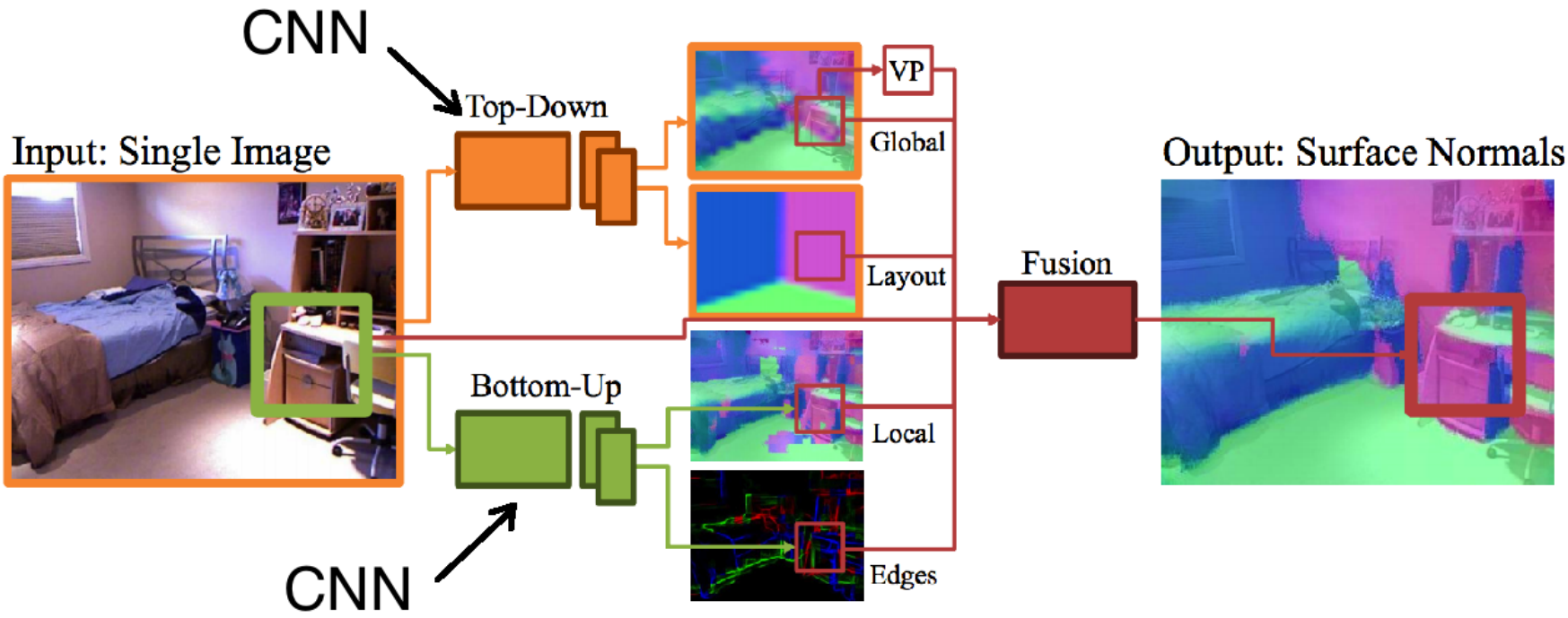


Figure: Predict surface normals via CNNs

[Xiaolong Wang, David F. Fouhey, Abhinav Gupta. Designing Deep Networks for Surface Normal Estimation. CVPR'15]

Depth from a Monocular Image

- Train Convolutional Neural Nets (CNNs) to predict **surface normals** instead of **depth** (works very well!)

Figure: Results

[Xiaolong Wang, David F. Fouhey, Abhinav Gupta. Designing Deep Networks for Surface Normal Estimation. CVPR'15]

8

Depth by Tricking Your Brain

Depth from a Monocular Image

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

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

Depth from a Monocular Image

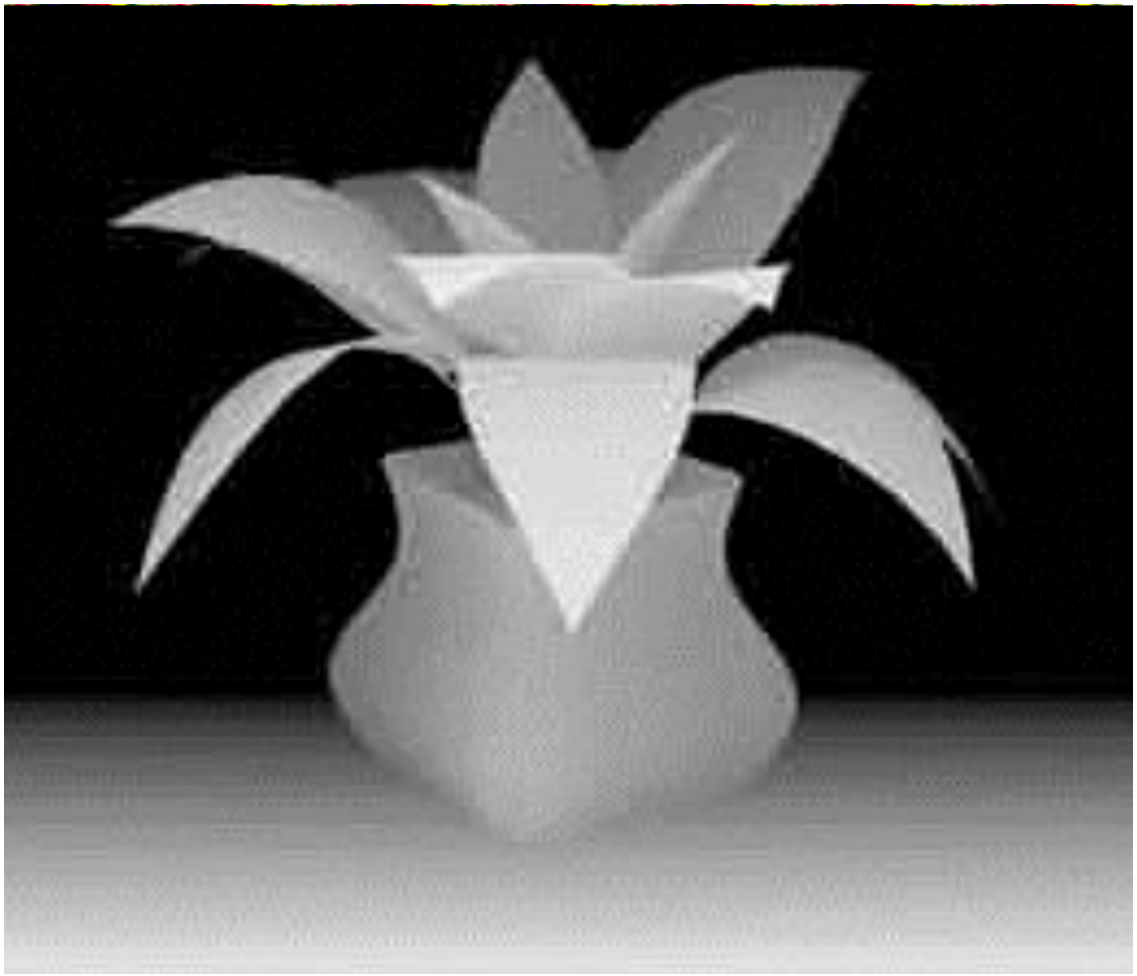


Figure: Depth by tricking the brain

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

Next Time:
Depth from Stereo