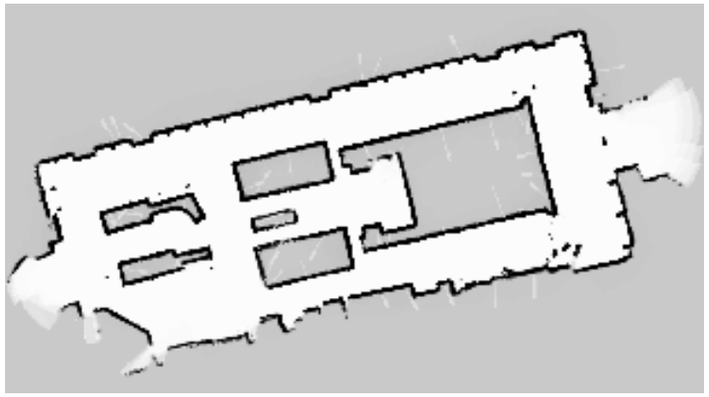
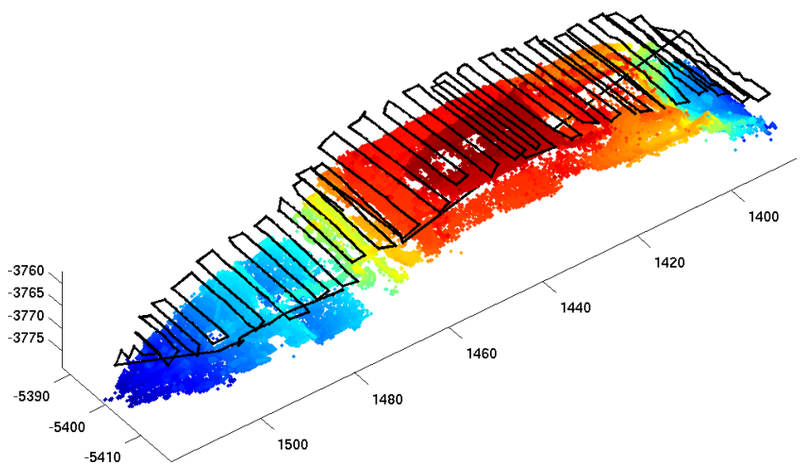
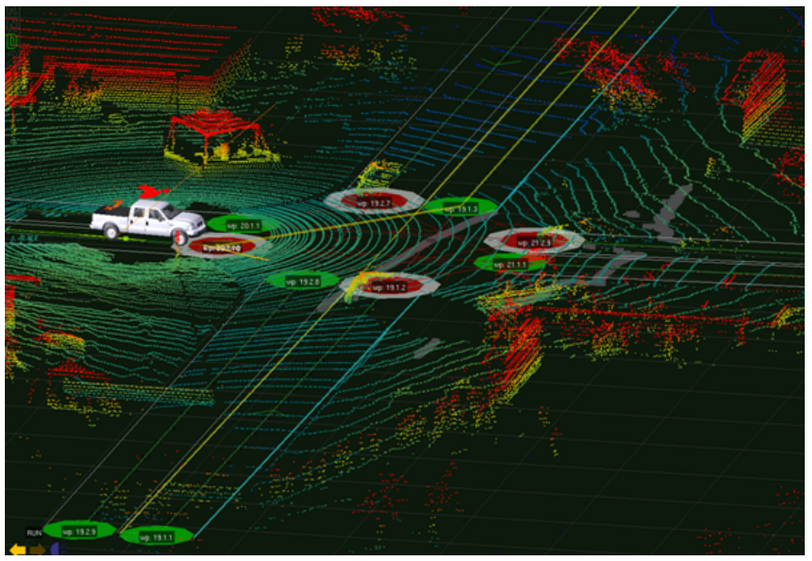




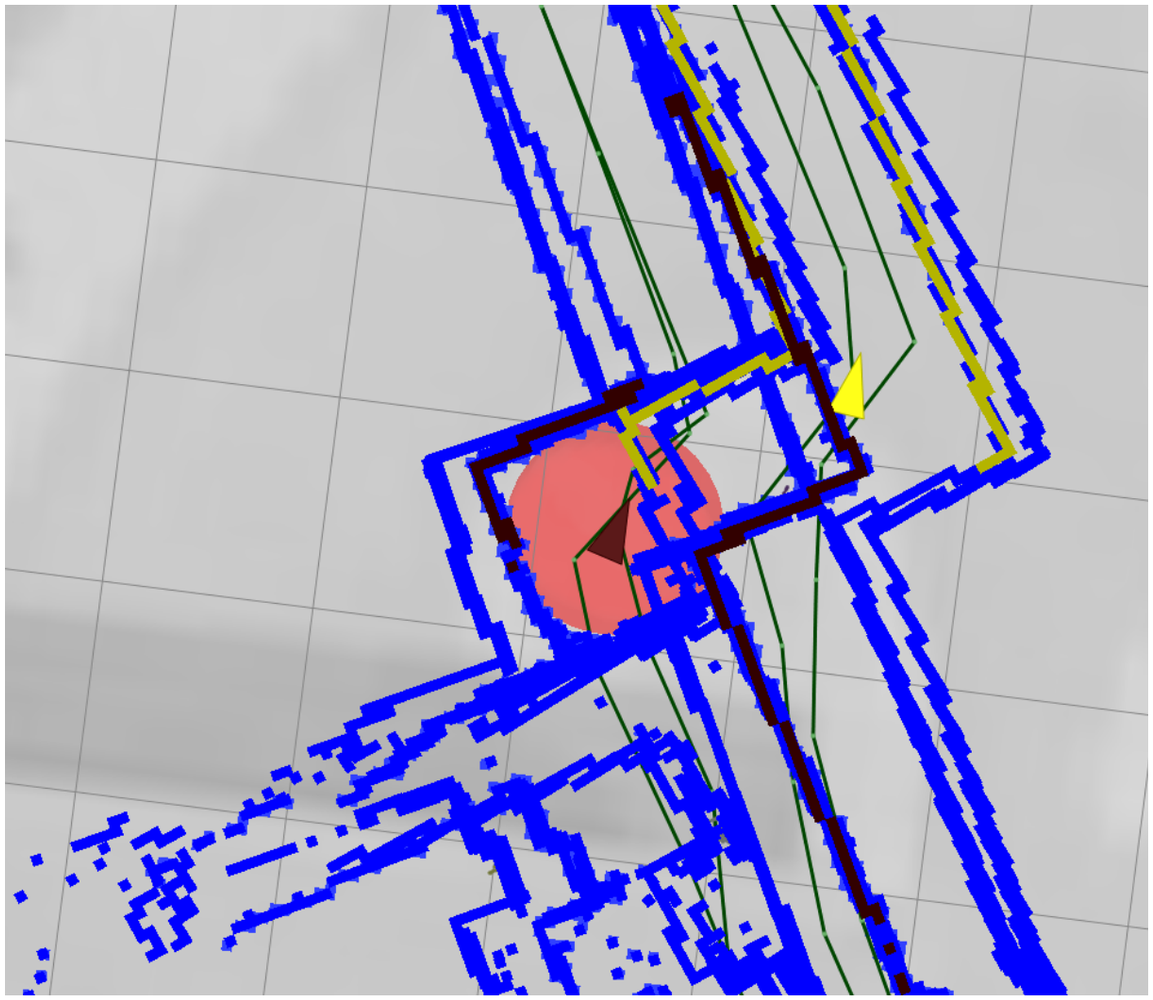


EECS568 Mobile Robotics: Methods and Principles

Prof. Edwin Olson

L16. Scan Matching and Image Formation

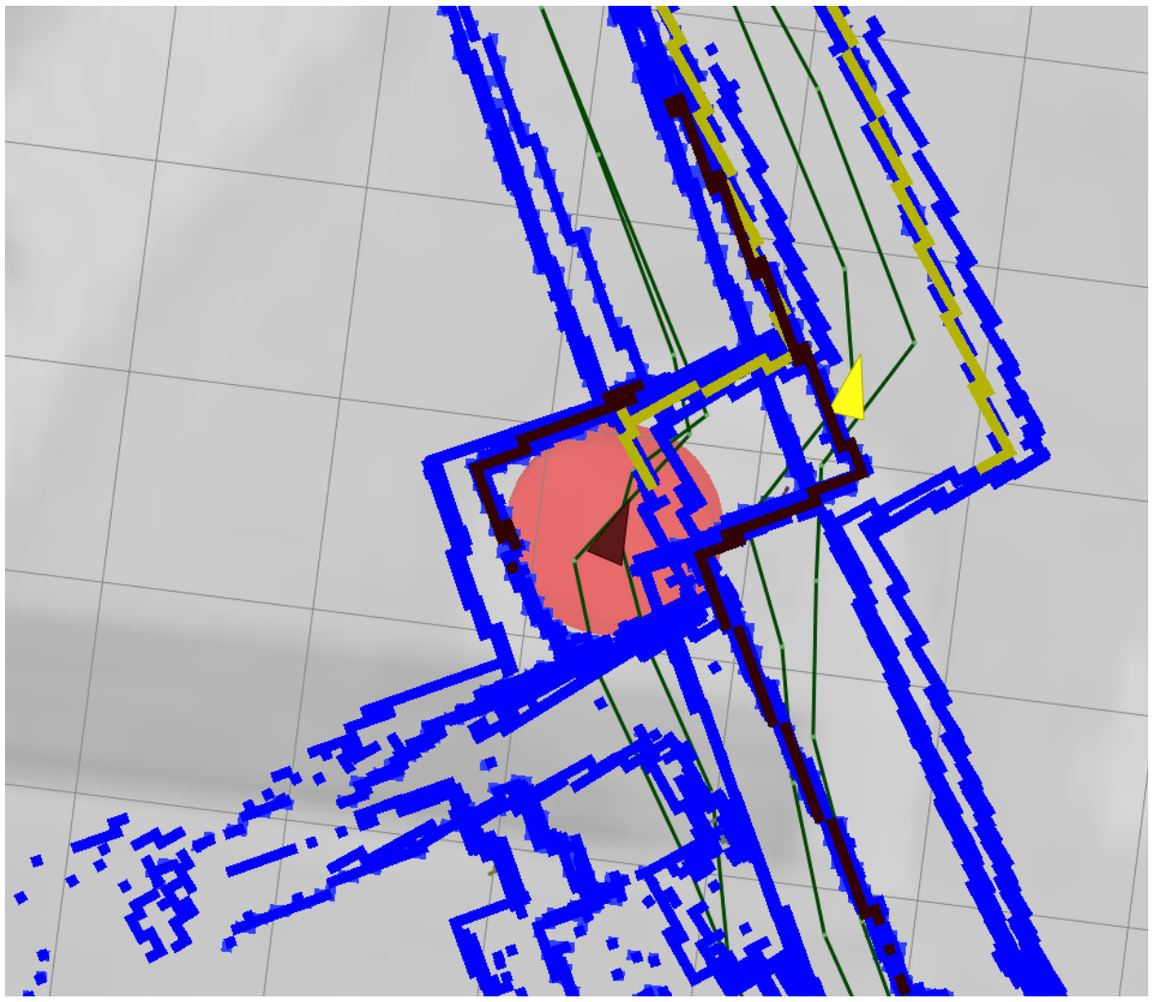
Scan Matching



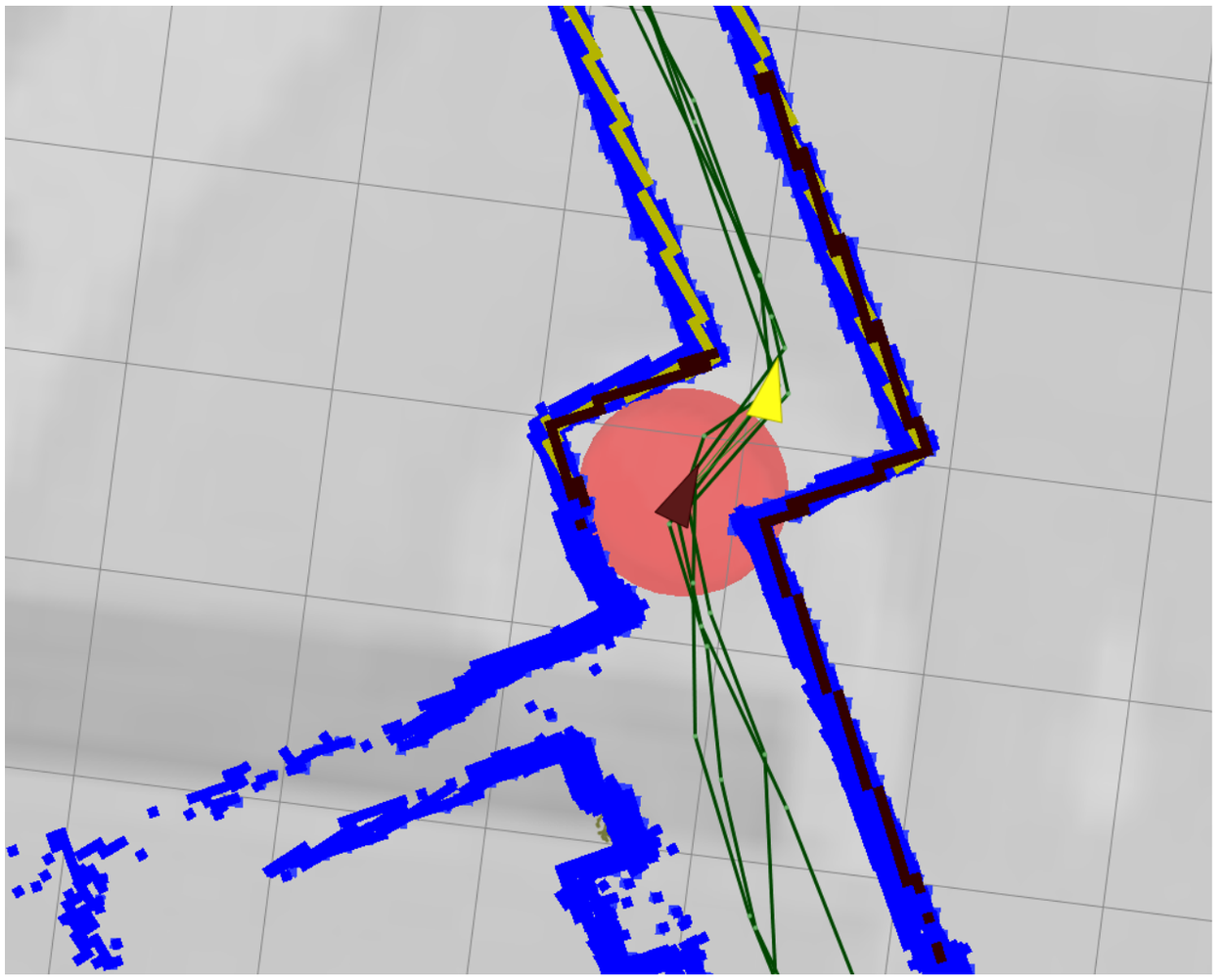
Before

After

Scan Matching



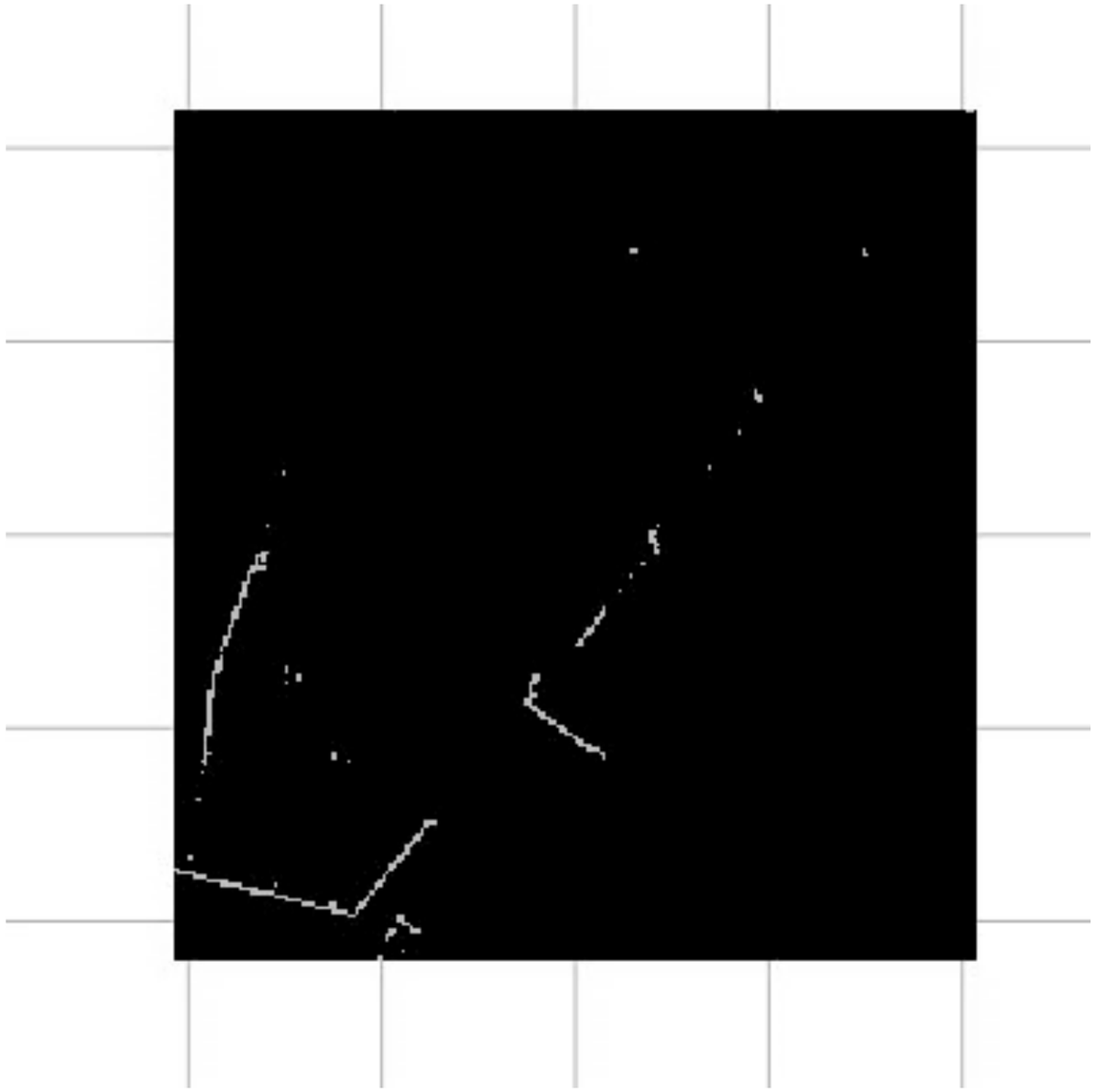
Before



After

Map matching has to be *fast*

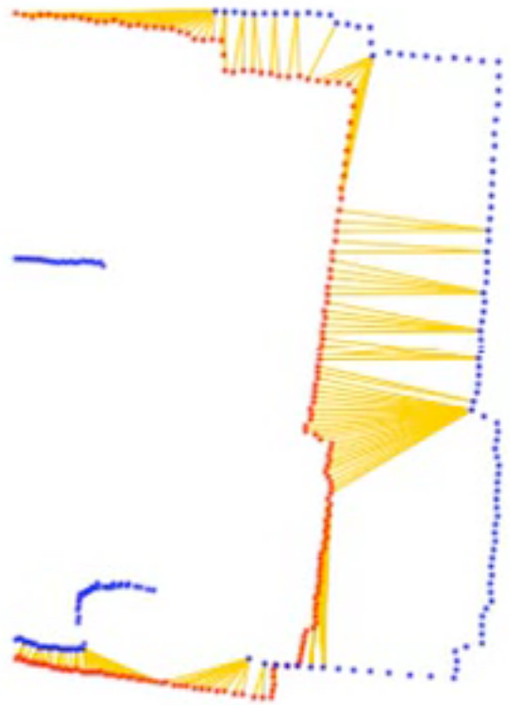
- 14 robots generate new maps at ~5 Hz.
- Must attempt matches against as many other possibly-overlapping maps as possible



- Our previous best methods manage ~50Hz.
  - Fast, but not fast enough!

Classic Method: ICP

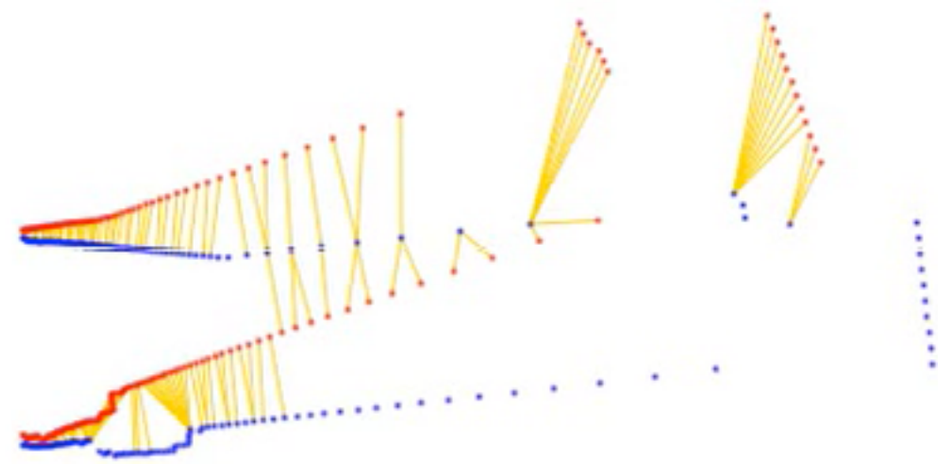
- Associate each point in scan A with its nearest neighbor in scan B
  - Compute rigid-body transformation
- Repeat.



15.2 FPS

Classic Method: ICP

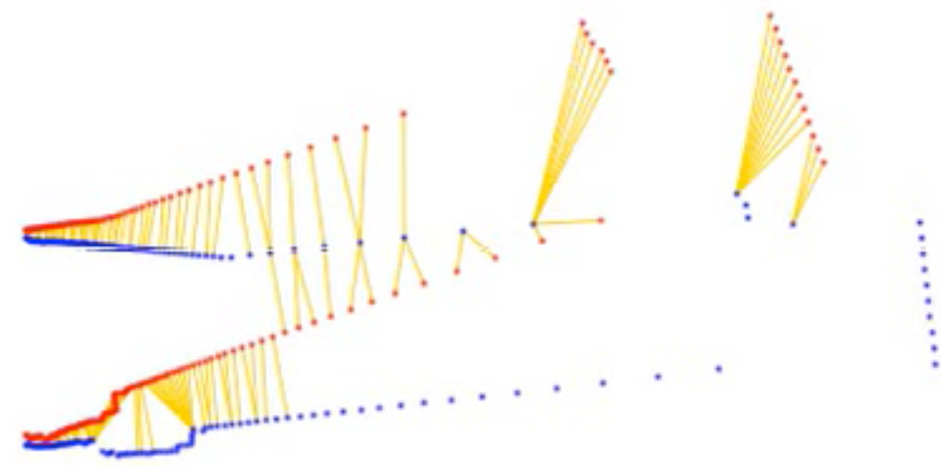
- Not probabilistically grounded
- Often fails to converge to even a "sensible" solution.



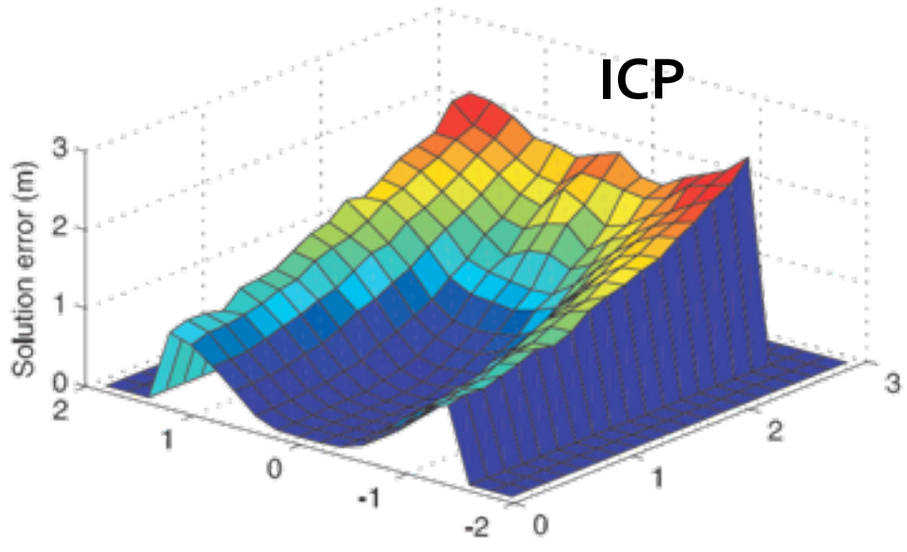
15.1 FPS

Classic Method: ICP

- Not probabilistically grounded
- Often fails to converge to even a "sensible" solution.



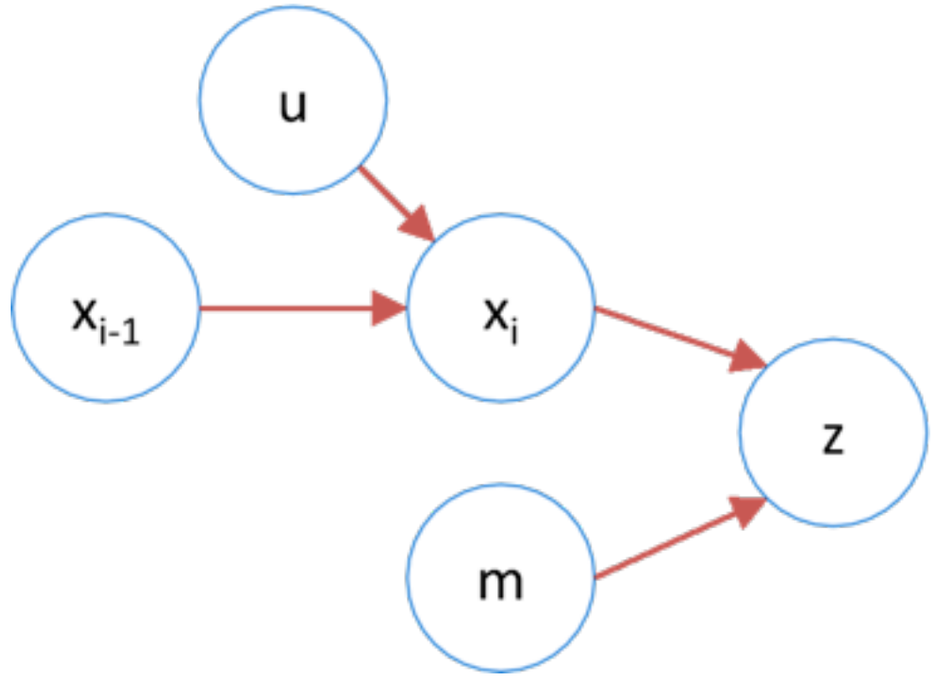
15.1 FPS



Probabilistic Formulation

- What we want:

$$p(x_i|x_{i-1}, u, z, m)$$



- Apply Bayes' law and exploit conditional independencies:

$$p(x_i|x_{i-1}, u, z, m) \propto p(z|x_i, m)p(x_i|x_{i-1}, u)$$

observation model *motion model*

- Basic idea: search for x_i that maximizes probability.

Observation model

- How do we compute $p(z|x_i, m)$?

- Lie #1: Assume independent laser measurements

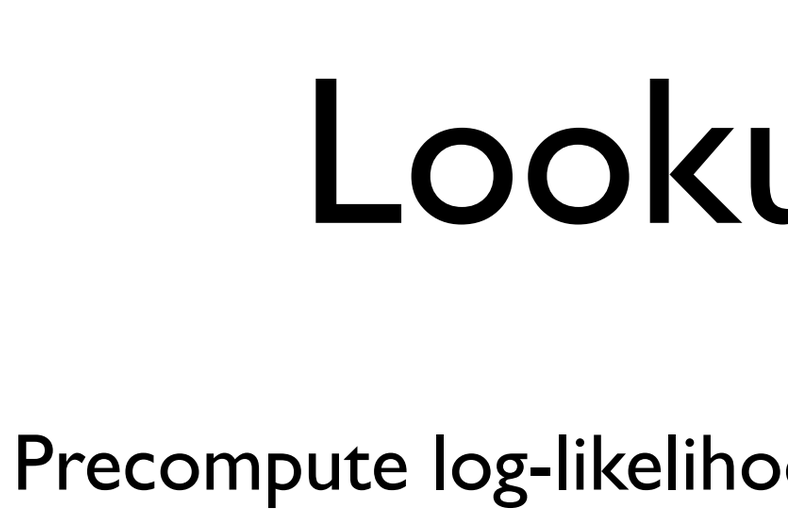
$$p(z|x_i, m) = \prod_j p(z_j|x_i, m)$$

- Lie #2: Ignore occlusion/visibility effects
- Lie #3: Assume radially symmetric Gaussian noise model

- *We can use a simple 2D lookup table for the probability of each point!*

Lookup Table

- Precompute log-likelihood of observing a point at each location
  - Approximately a convolution of a Gaussian with the "reference" scan



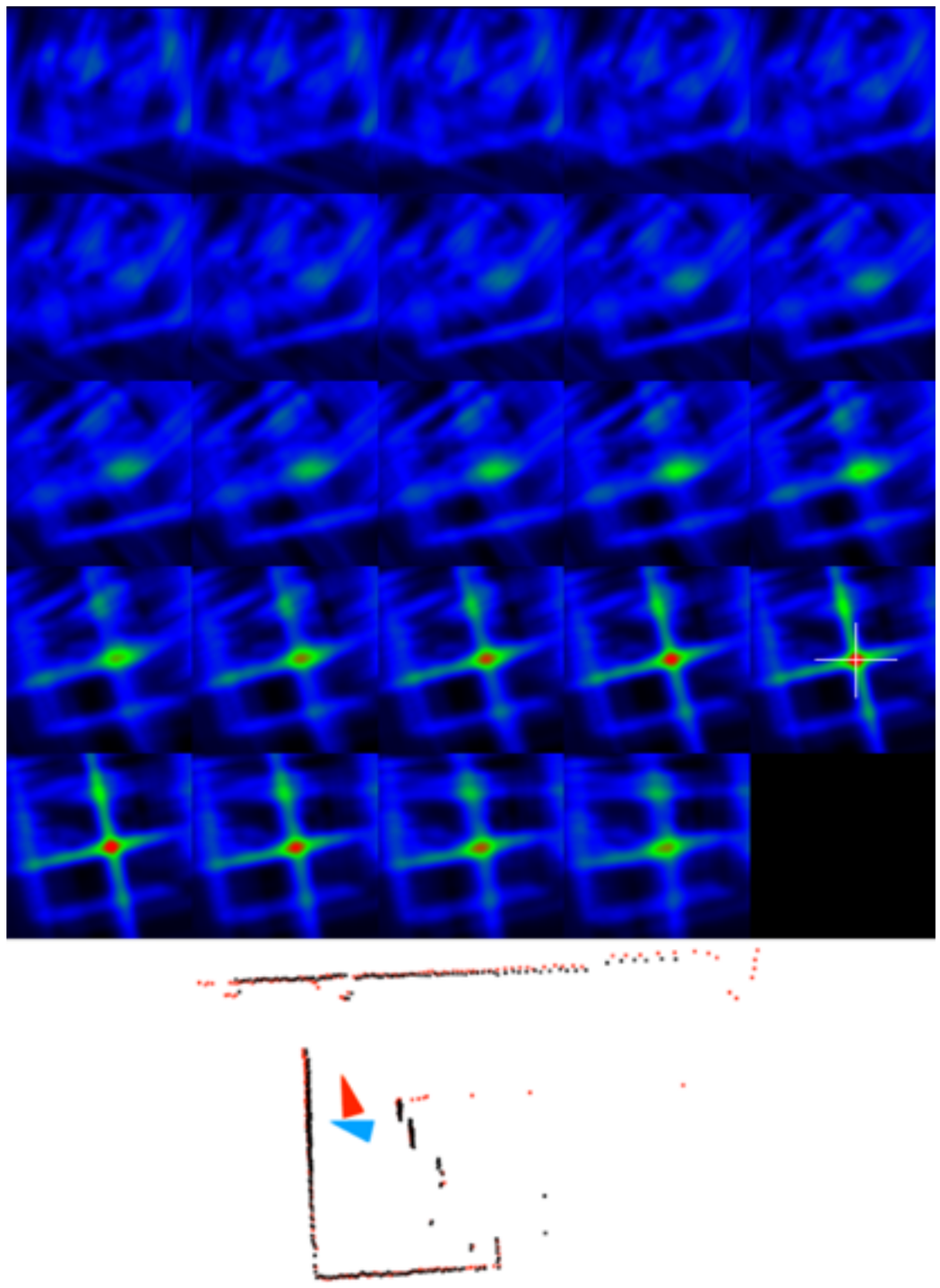
Reference Scan

Lookup Table

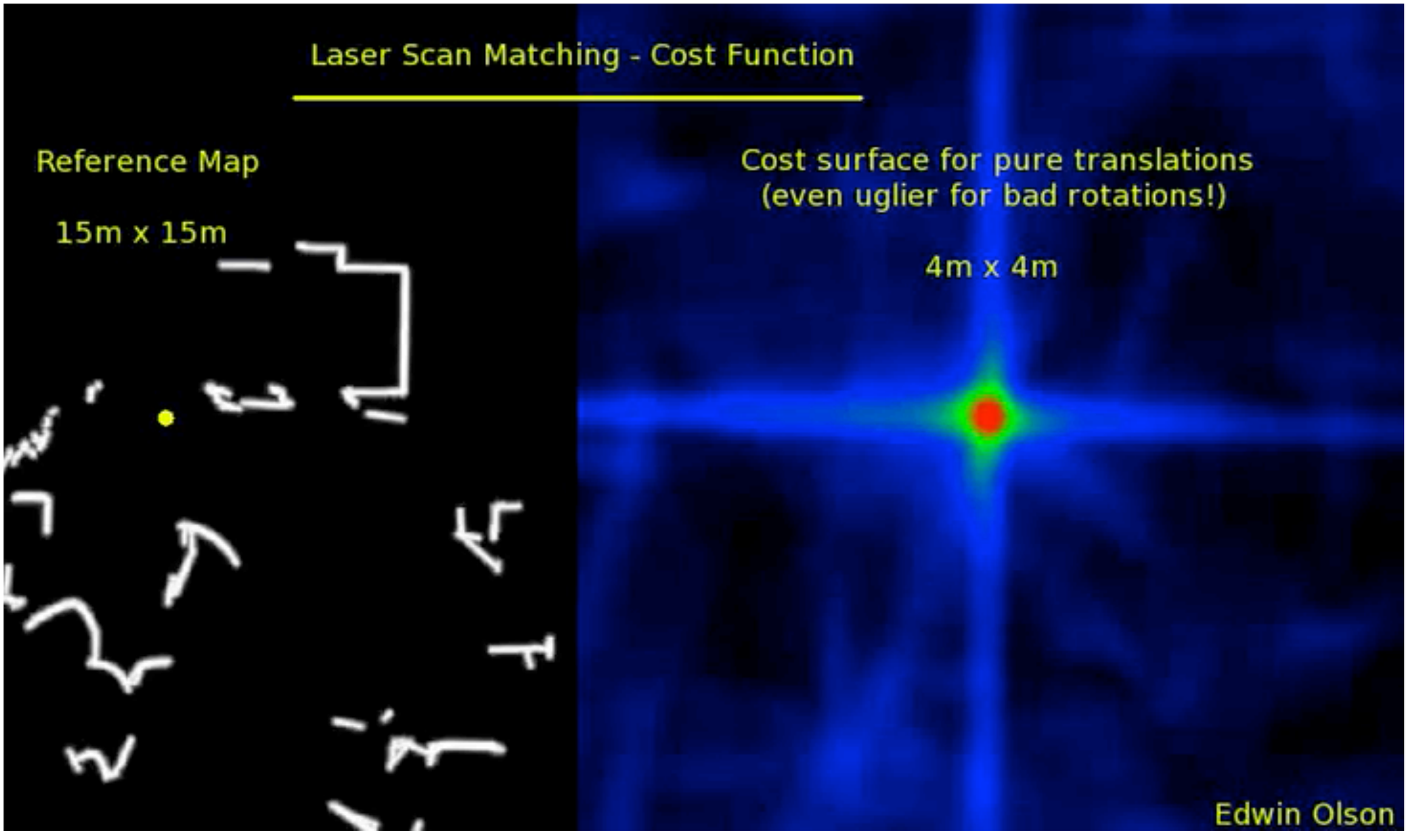
Correlative Scan Matching

- If we exhaustively search over x_i's, we just pick out the maximum likelihood solution.

- This is quite slow!
  - Runtime grows rapidly with amount of uncertainty



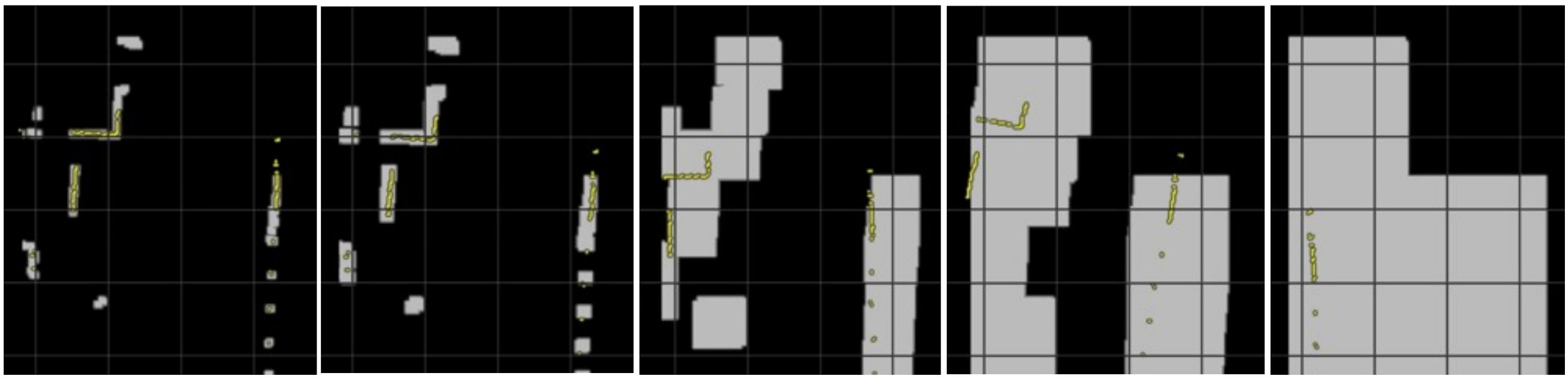
Scan Matching in Action



Multi-Resolution scan matching

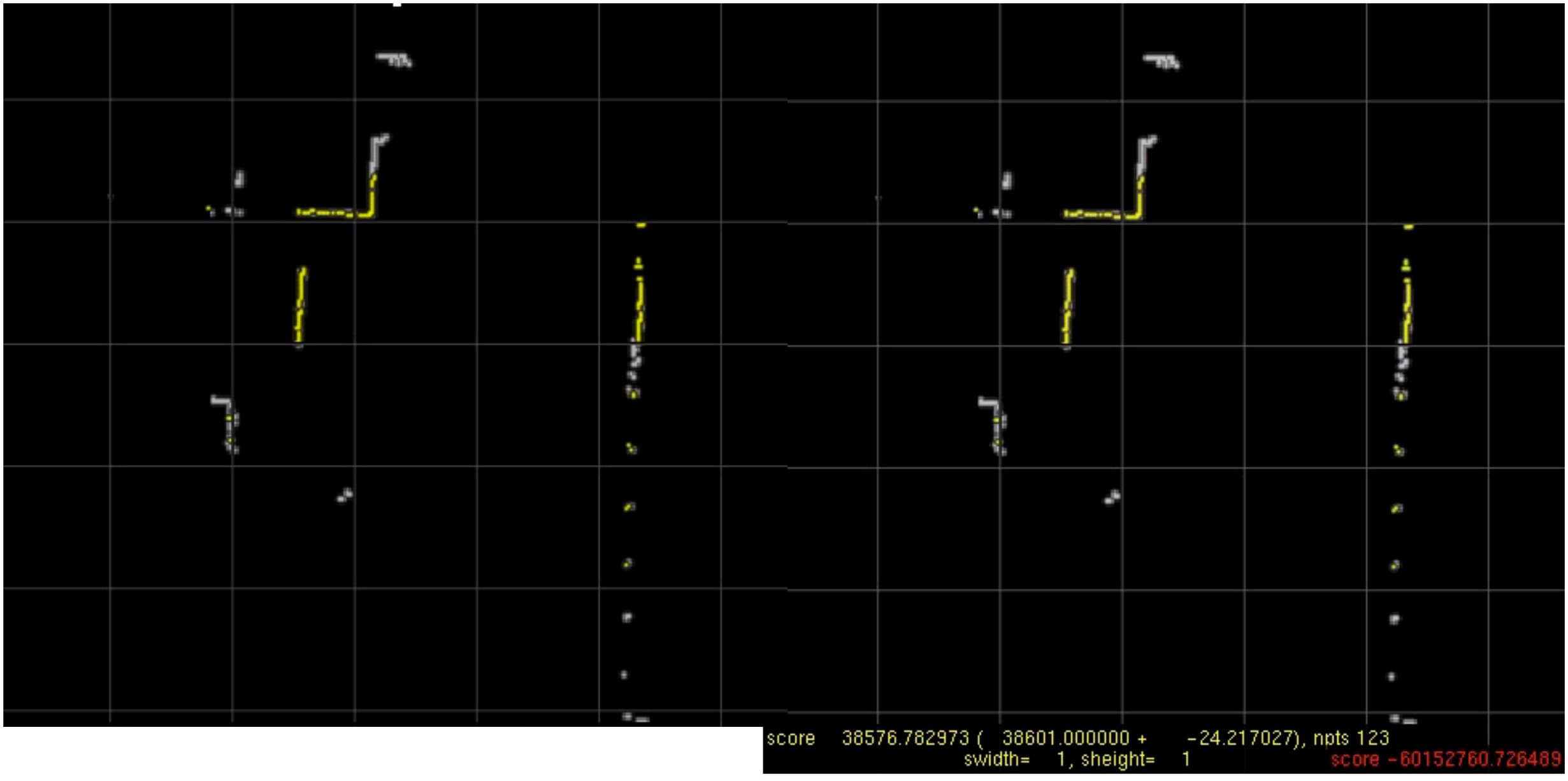
- Initialize a heap
- Populate heap with an exhaustive search at very coarse resolution
  - Use low-resolution versions of lookup tables so that we don't suffer from aliasing, and so we don't miss the optimum
- while true
  - Extract most promising solution, s
  - if full-resolution(s)
    - return s
  - else
    - Resample s at finer resolution, add to heap.

Multi-resolution matching



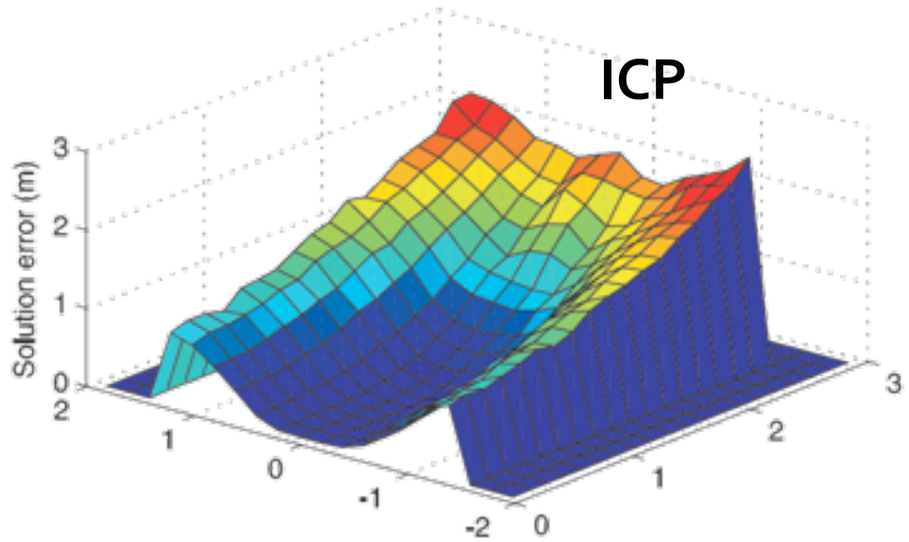
- Result of max-convolutions
  - Guarantees that score computed by low-resolution version is at least as good as any higher-resolution version in the same area.

Multi-resolution matching



Matching Results

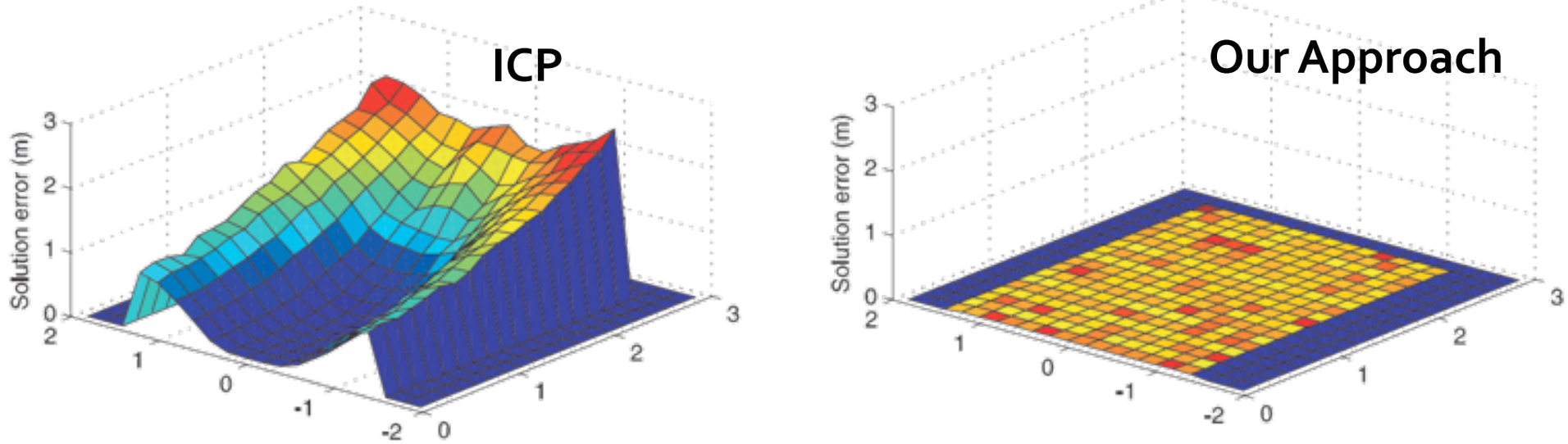
- Immunity to local minima (versus ICP)



- Search time is largely independent of search window
  - Low-probability alignments are quickly ruled out
  - Typical matching in ~2 ms

Matching Results

- Immunity to local minima (versus ICP)

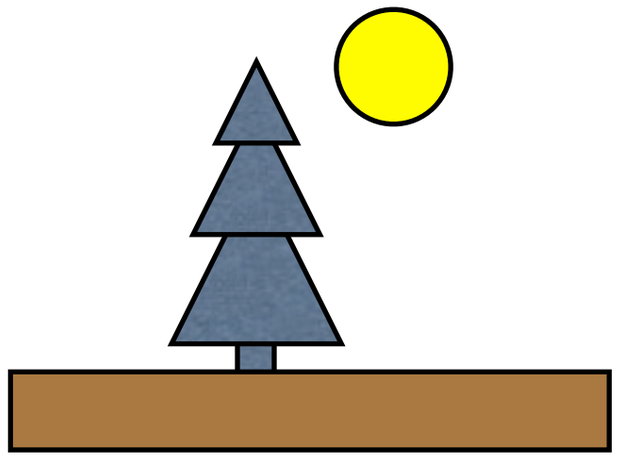


- Search time is largely independent of search window
  - Low-probability alignments are quickly ruled out
  - Typical matching in ~2 ms

Cameras and Image Formation

World Simplest Camera?

The world

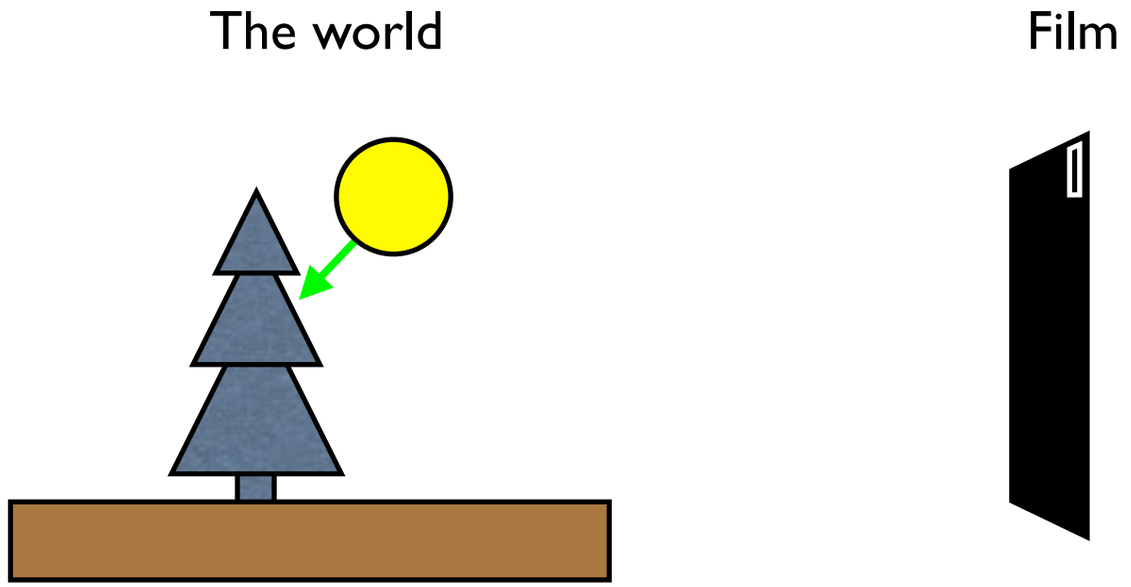


Film

- Just hold up a piece of film

- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

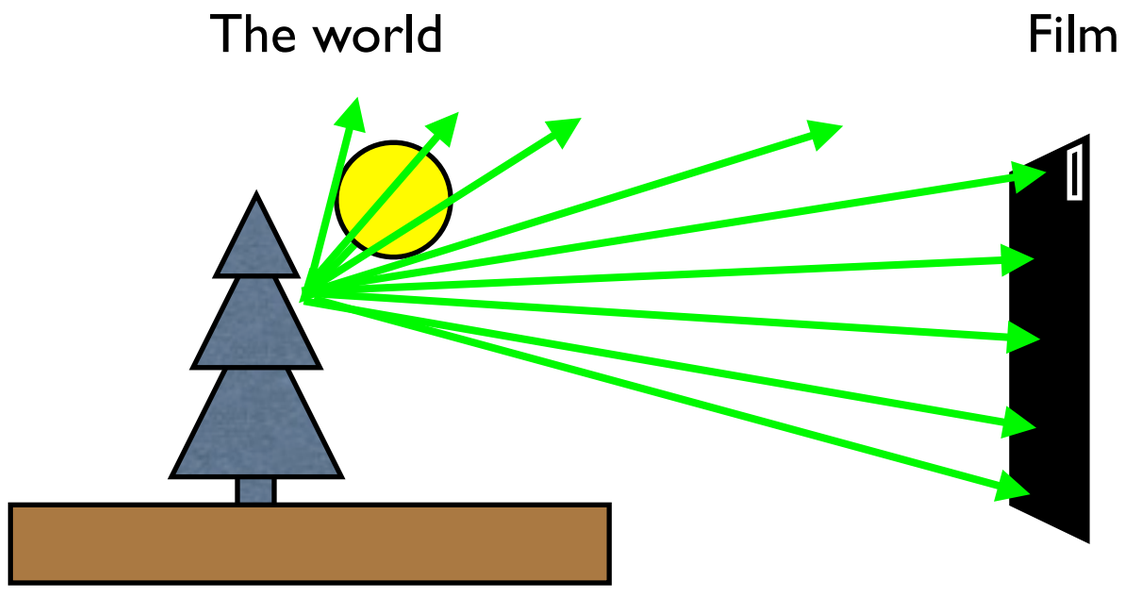
World Simplest Camera?



- Just hold up a piece of film

- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

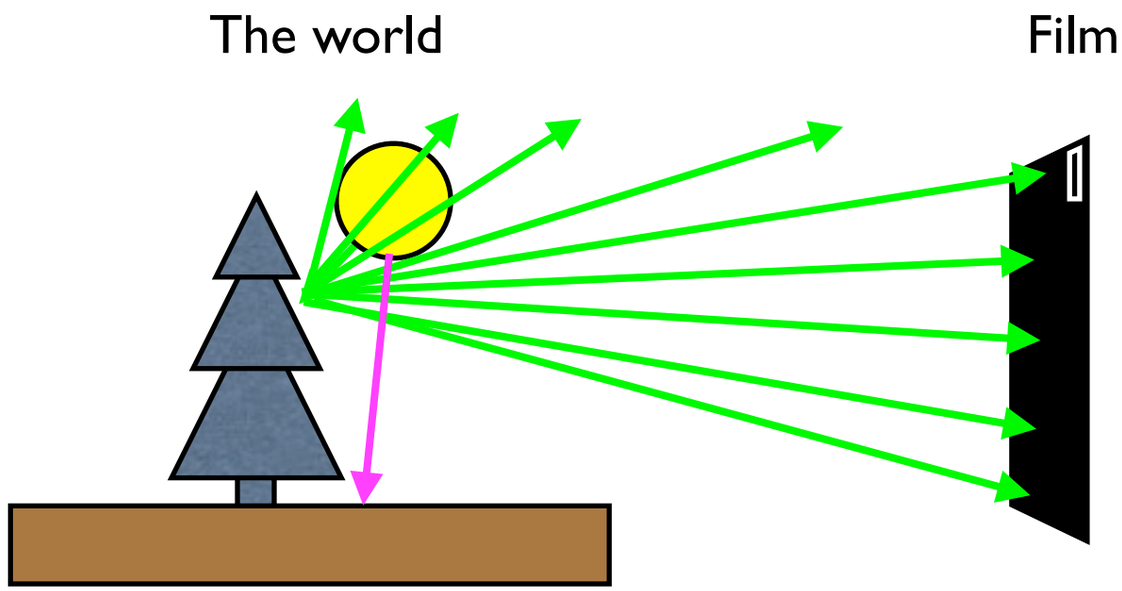
World Simplest Camera?



- Just hold up a piece of film

- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

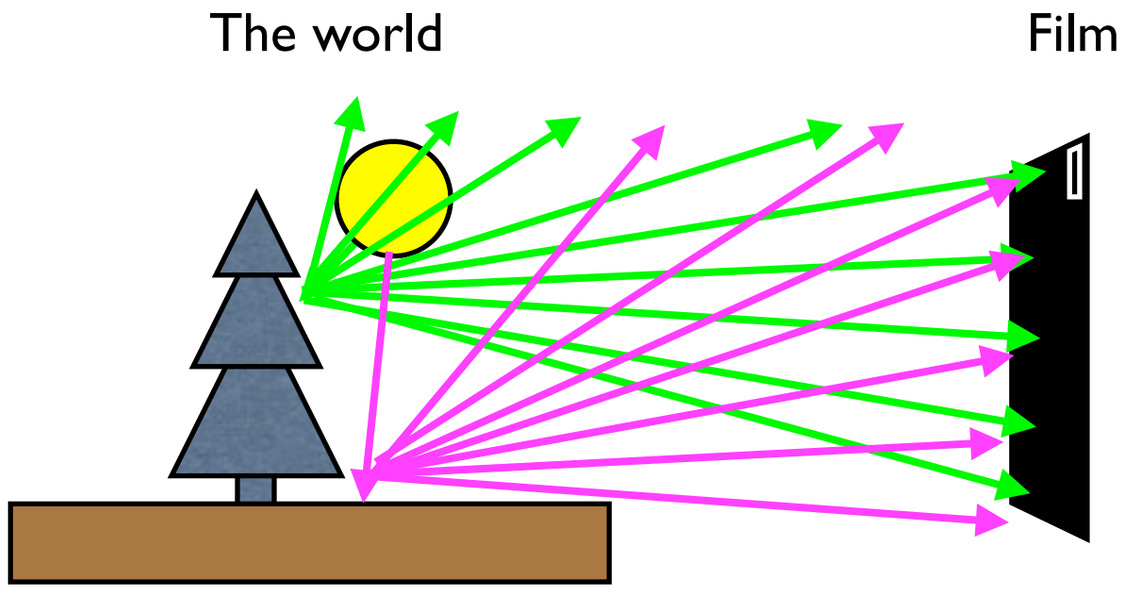
World Simplest Camera?



- Just hold up a piece of film

- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

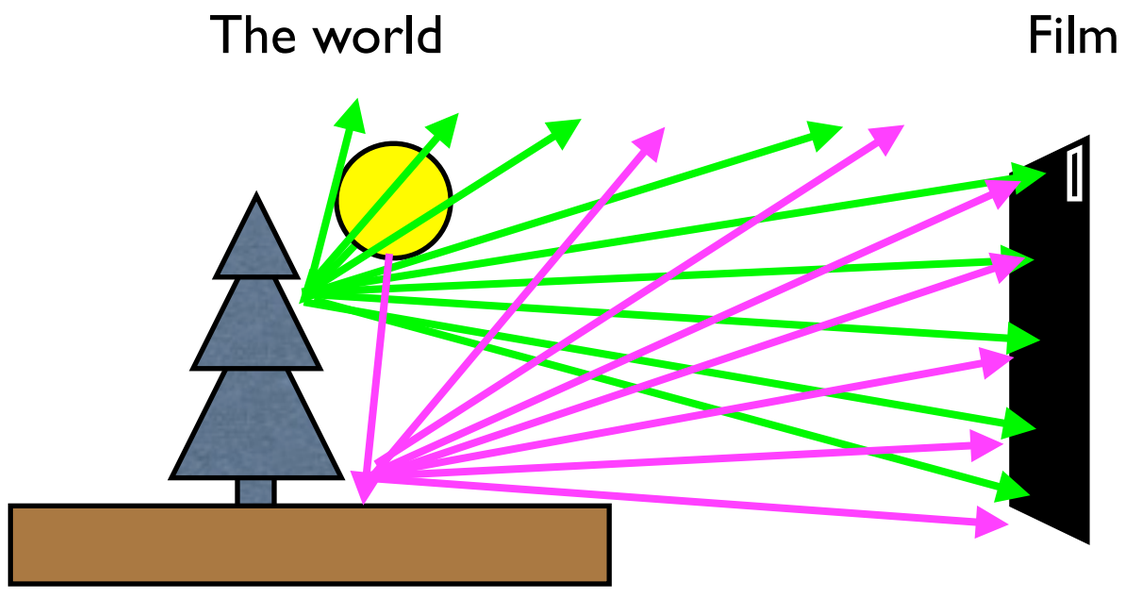
World Simplest Camera?



- Just hold up a piece of film

- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

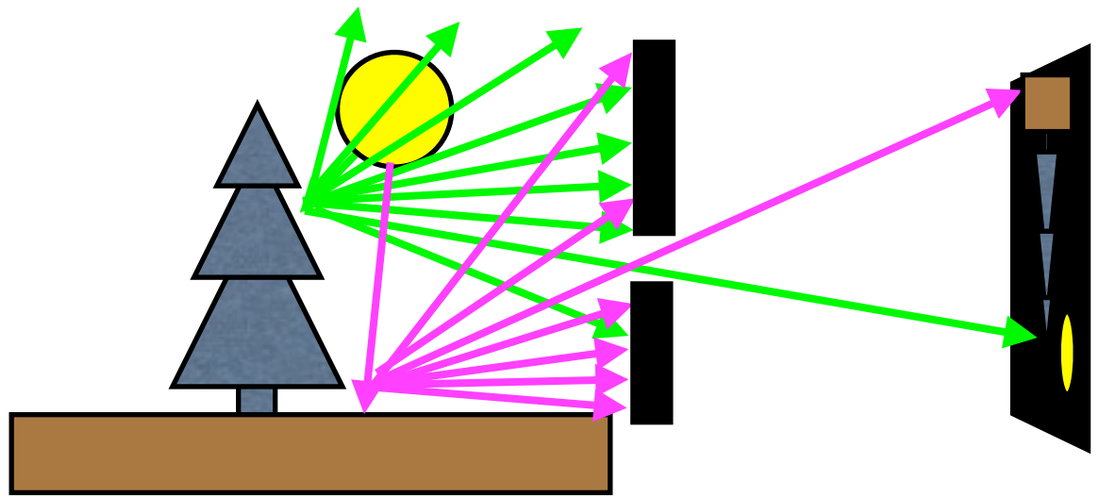
World Simplest Camera?



- Just hold up a piece of film

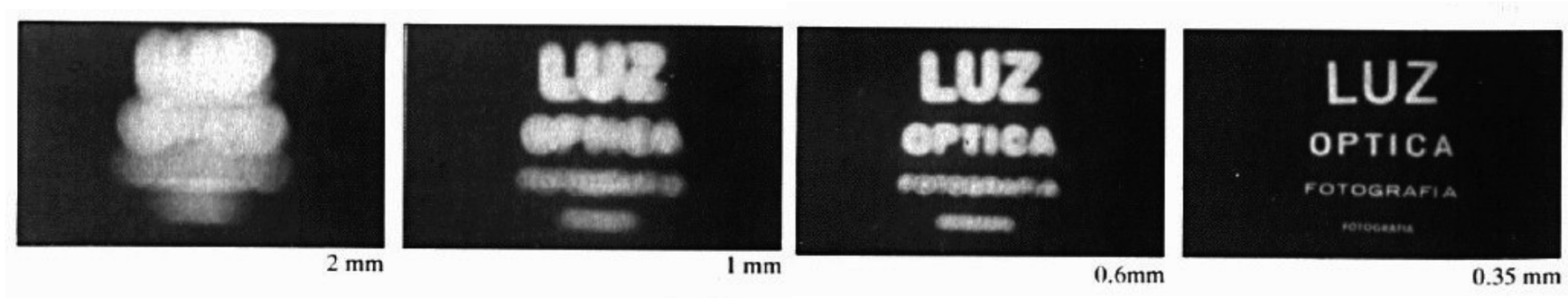
- Do we get an image on the film?
  - For each piece of the film, where do the photons come from?

Let's add an aperture



- An aperture blocks all but a small subset of the rays
  - Causes the image to appear in focus!

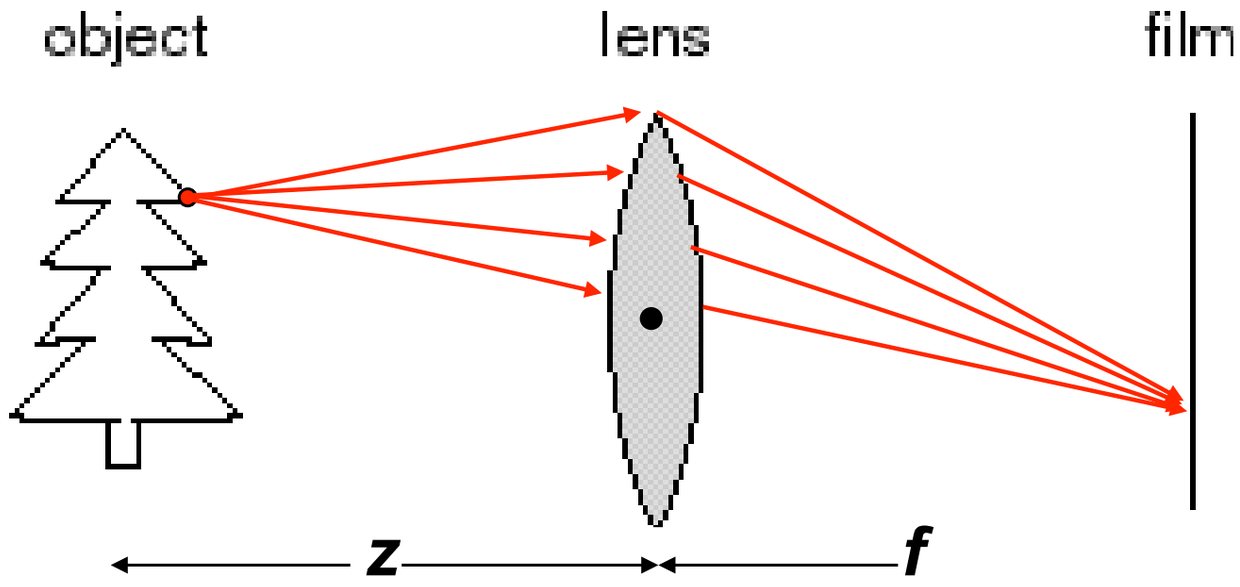
Aperture Size



- Why not make the aperture super small?
  - A “pin-hole” lens.
  - Not enough light to “register” on our film
- What happens when the aperture is bigger?
  - More rays can fit through--- blurrier image

- Is there any way of getting a sharp image, but allow more light through?
  - Yes! A *lens*.

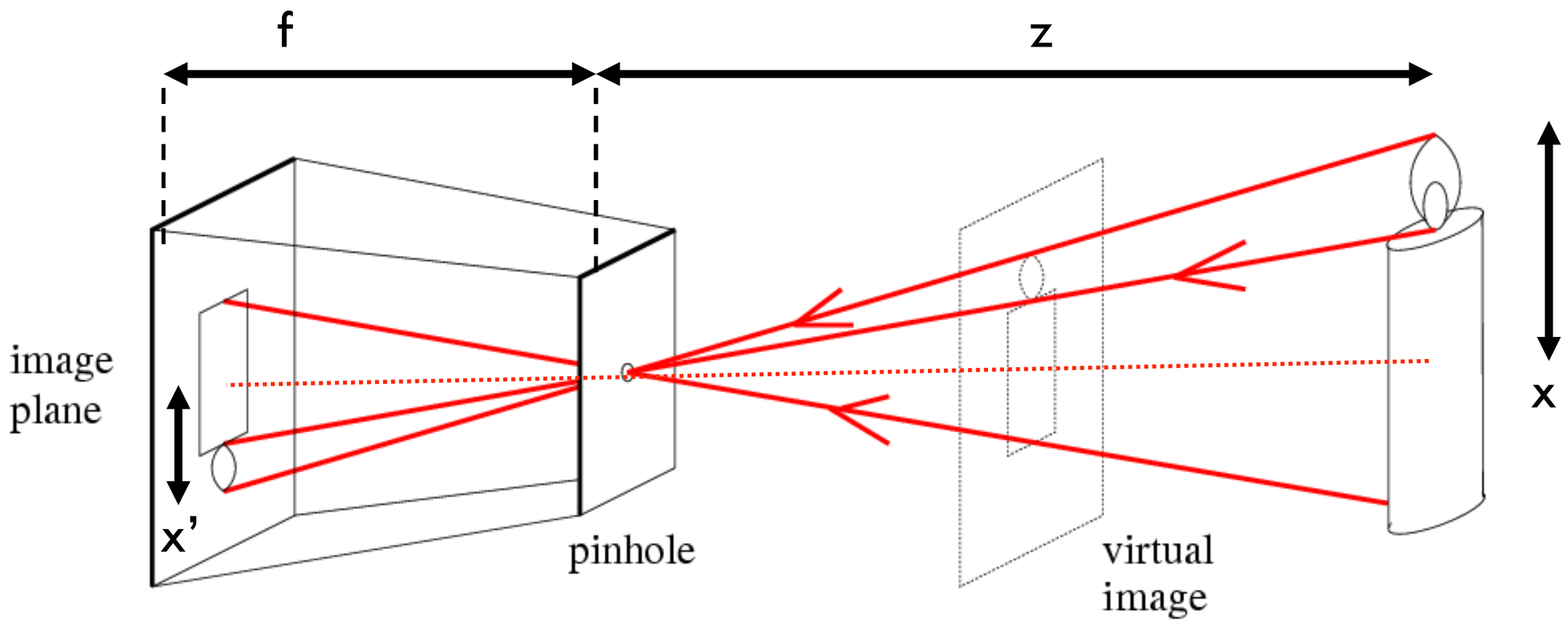
Lenses



- A lens collects rays with a particular divergence and refocuses them to a point.
  - But points at the "wrong" distance won't be refocused exactly.
- *Depth of field:* how much of the scene is in focus

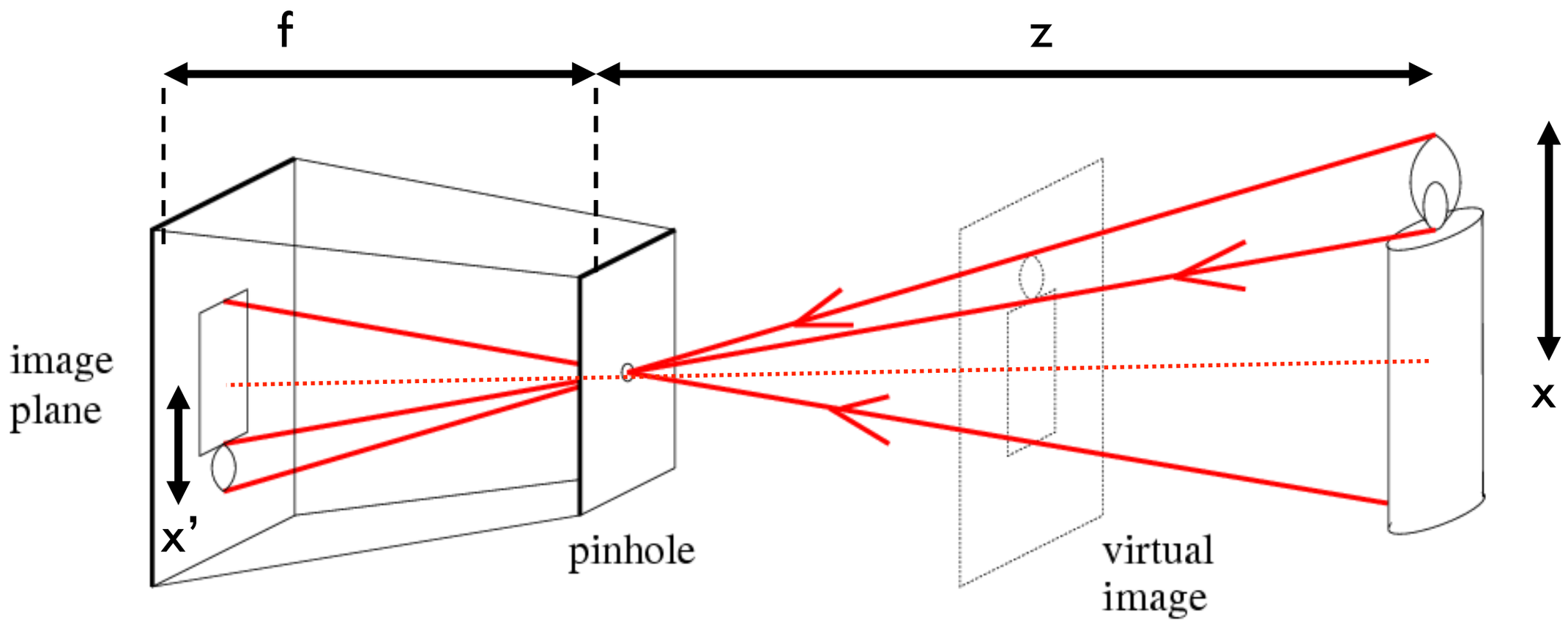
- We're going to ignore this today, however--- we're going to assume a "pin-hole" model.

Perspective Projection



- The pinhole creates two similar triangles
  - Allows us to determine x' in terms of x

Perspective Projection

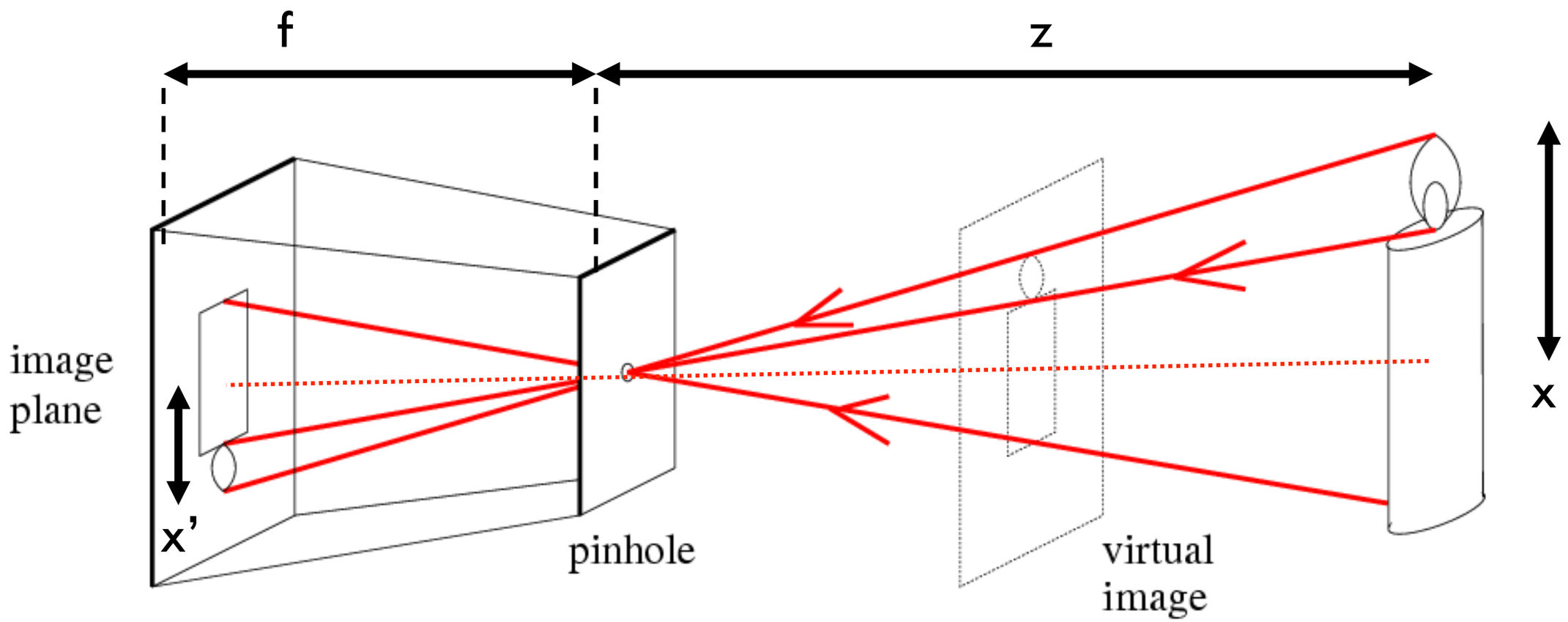


- The pinhole creates two similar triangles
  - Allows us to determine x' in terms of x

$$x' = -xf/z$$

(why is it negative? we'll assume from here on out that the camera "unflips" the image.)

Perspective Projection



- What are the pixel coordinates where the flame appears?
  - $x' = fx/z + c$
  - Measure f in "pixels" and add an offset (so that the "middle" pixel is in the middle of the image)

The Perspective Matrix

- Suppose we write a point in the world (like the position of the candle flame) as a vector:

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

the matrix M is a function of z! We want M to work for any point p.

We also are unable to handle cx, cy

- Can we write a matrix so that p' = Mp?

$$p' = \begin{bmatrix} fx/z + c_x \\ fy/z + c_y \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} f/z & 0 & 0 \\ 0 & f/z & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

The Perspective Matrix

- Suppose we write a point in the world (like the position of the candle flame) as a vector:

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

the matrix M is a function of z! We want M to work for any point p.

We also are unable to handle cx, cy

- Can we write a matrix so that p' = Mp?

$$p' = \begin{bmatrix} fx/z + c_x \\ fy/z + c_y \end{bmatrix} \stackrel{?}{=} \begin{bmatrix} f/z & 0 & 0 \\ 0 & f/z & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

What's wrong with this?

Homogenous Coordinates

- We'll introduce a new convention, *homogenous coordinates*.
- We write points just the way we did before, but add an extra row:
  - The extra row is a *scale factor* for the whole vector.

there's a whole related set of geometry, but all we need to know in this lecture is that there's a funny scale factor.

Homogenous Coordinates

- We'll introduce a new convention, *homogenous coordinates*.
- We write points just the way we did before, but add an extra row:
  - The extra row is a *scale factor* for the whole vector.

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{becomes} \quad p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

there's a whole related set of geometry, but all we need to know in this lecture is that there's a funny scale factor.

Homogenous Coordinates

- We'll introduce a new convention, *homogenous coordinates*.
- We write points just the way we did before, but add an extra row:
  - The extra row is a *scale factor* for the whole vector.

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{becomes} \quad p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

What point does this correspond to?

$$\begin{bmatrix} 10 \\ 20 \\ 15 \\ 5 \end{bmatrix}$$

there's a whole related set of geometry, but all we need to know in this lecture is that there's a funny scale factor.

Do homogeneous coordinates help?

- Eureka!

$$p' = \begin{bmatrix} fx/z + c_x \\ fy/z + c_y \\ 1 \end{bmatrix} = \begin{bmatrix} fx + c_x z \\ fy + c_y z \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2d homogeneous coordinates

2d homogeneous coordinates

perspective transform

3d homogeneous coordinates

Moving the camera

- Basic idea:
  - Moving the camera is exactly the same thing as moving the world in the opposite way.

- But how do we represent motion?
  - With a matrix!

Translation

- Suppose I want to shift all objects by Tx, Ty, Tz:

$$\begin{bmatrix} x + T_x \\ y + T_y \\ z + Tz \\ 1 \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

?

Translation

- Suppose I want to shift all objects by Tx, Ty, Tz:

$$\begin{bmatrix} x + T_x \\ y + T_y \\ z + Tz \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

- How about a rotation of 90 degrees around the Z axis?

$$\begin{bmatrix} -y \\ x \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} ? \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation

- How about a rotation of 90 degrees around the Z axis?

$$\begin{bmatrix} -y \\ x \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation Matrices: Intuition

- What do those R's mean?
  - They correspond to the directions in the direction!

You can easily verify that the [1 0 0 1] vector is transformed to [R00 R10 R20 1]

$$\begin{matrix} & \hat{x}' & \hat{y}' & \hat{z}' & \\ \end{matrix}$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation Matrices: Intuition

- What do those R's mean?
  - They correspond to the directions in the direction!

You can easily verify that the [1 0 0 1] vector is transformed to [R00 R10 R20 1]

$$\begin{matrix} & \hat{x}' & \hat{y}' & \hat{z}' & \\ \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} & = & \begin{bmatrix} R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \end{matrix}$$

Important Properties:

1. Each column is a unit vector
2. Each column is perpendicular to the others

Rigid-Body Transformations

- The product of two rigid-body transformations is ***always*** another rigid-body transformation!
- So no matter how the object has been translated or rotated, we can describe its position with a single 4x4 matrix, which has the structure:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{00} & R_{01} & R_{02} & T_x \\ R_{10} & R_{11} & R_{12} & T_y \\ R_{20} & R_{21} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Putting it all together

$$\begin{bmatrix} x' \\ y' \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & R_{02} & T_x \\ R_{10} & R_{11} & R_{12} & T_y \\ R_{20} & R_{21} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2d homogeneous pixel (camera) coordinates

Focal length and focal center of camera

Rigidly move every object in the world to simulate the camera's true position

3d homogenous (world) coordinates

Putting it all together

$$\begin{bmatrix} x' \\ y' \\ s \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & R_{02} & T_x \\ R_{10} & R_{11} & R_{12} & T_y \\ R_{20} & R_{21} & R_{22} & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

2d homogeneous pixel (camera) coordinates

Focal length and focal center of camera

"Intrinsics"

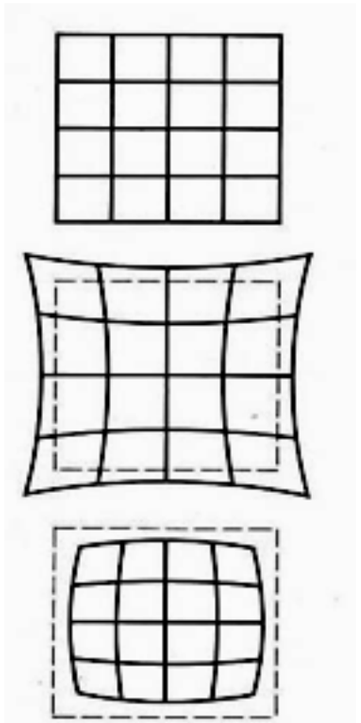
Rigidly move every object in the world to simulate the camera's true position

"Extrinsics"

3d homogenous (world) coordinates

Lens distortions

- Unfortunately, real (imperfect) lenses further complicate life.



Undistorted

Pin cushion

Barrel (common)

Correcting for lens distortion

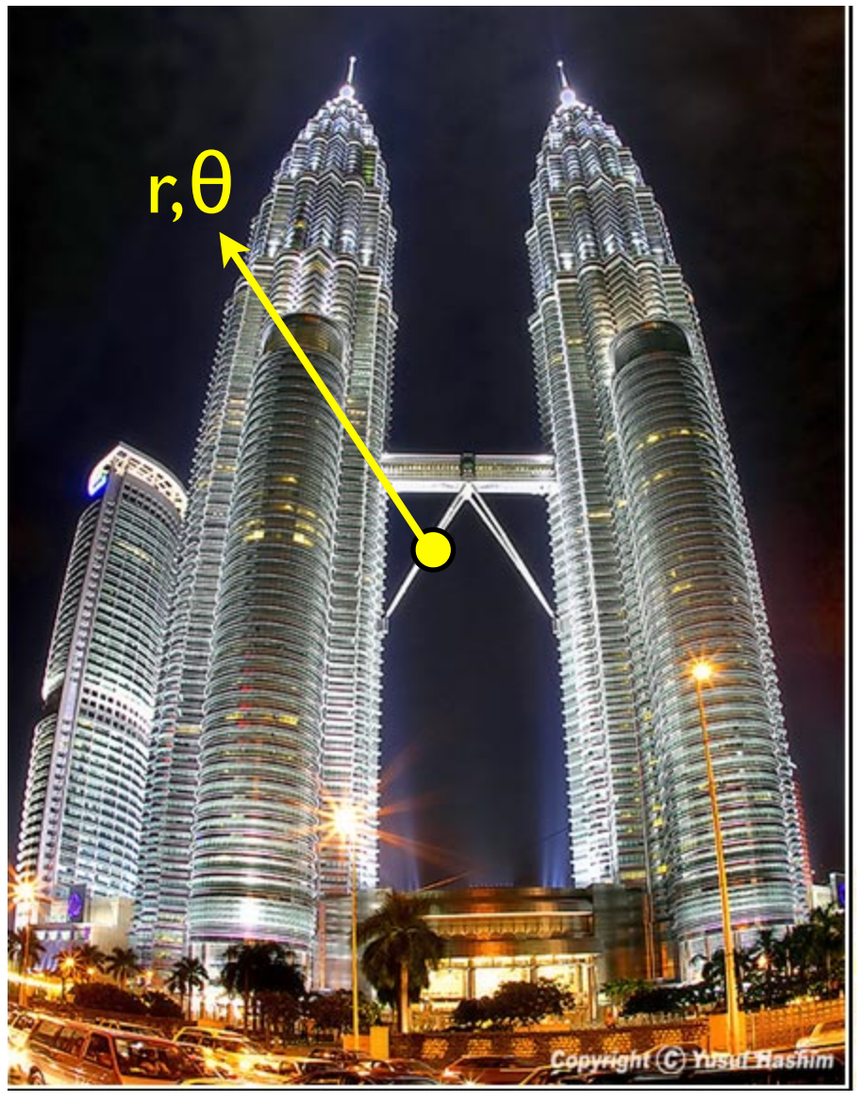
- Radial Distortion

  1. Compute the pixel coordinates assuming the lens is undistorted

  2. Convert to polar form

  3. Compute r' = f(r)

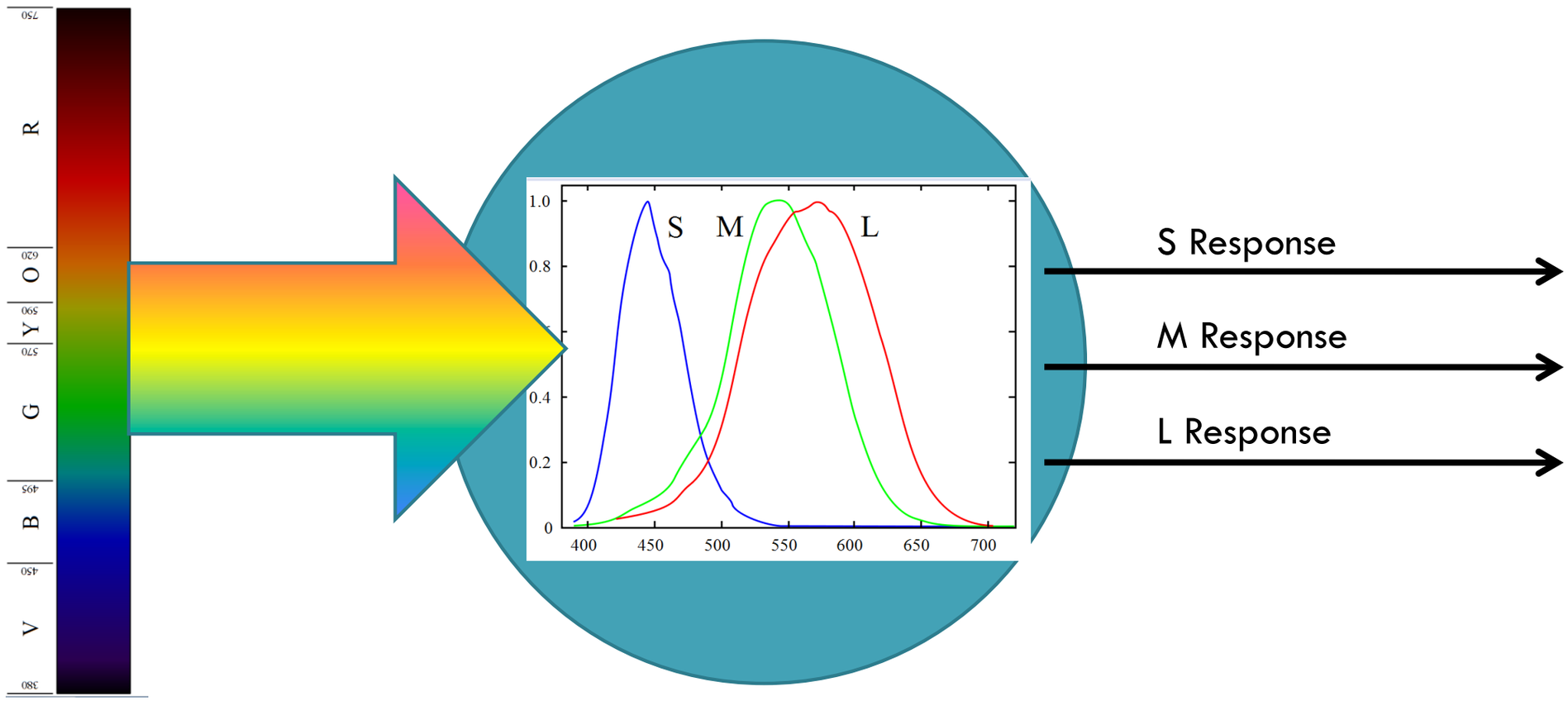
  4. Convert r' and θ back to Cartesian coordinates.



- Function f() is typically nasty polynomial functions.
  - We find the parameters by using non-linear optimization algorithms

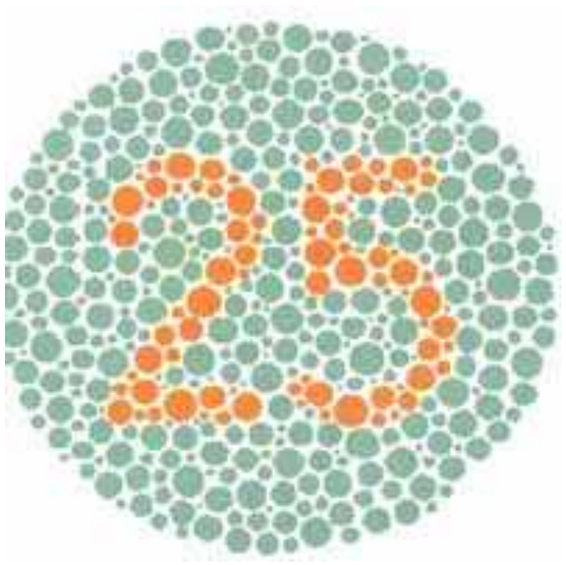
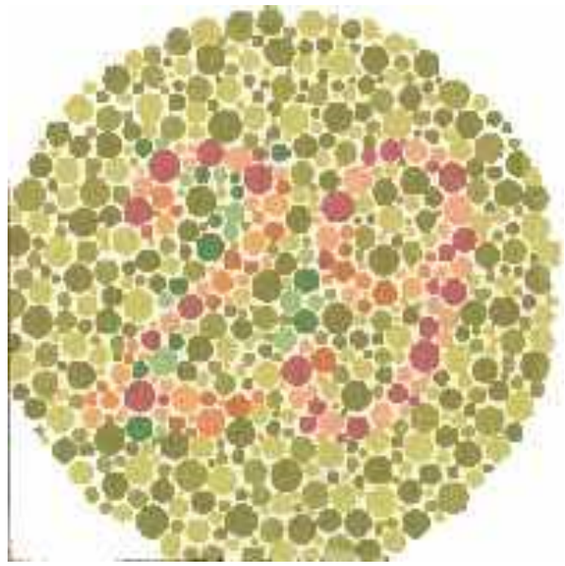
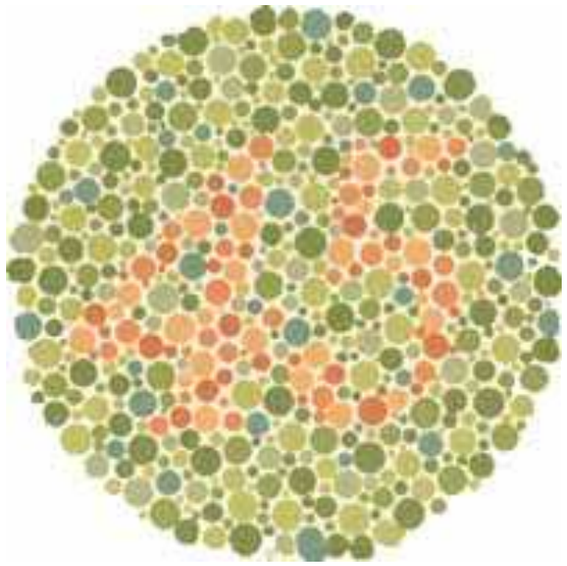
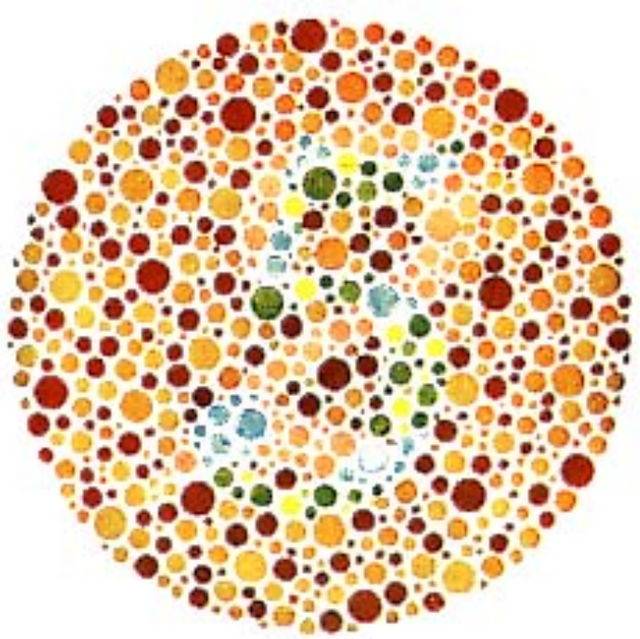
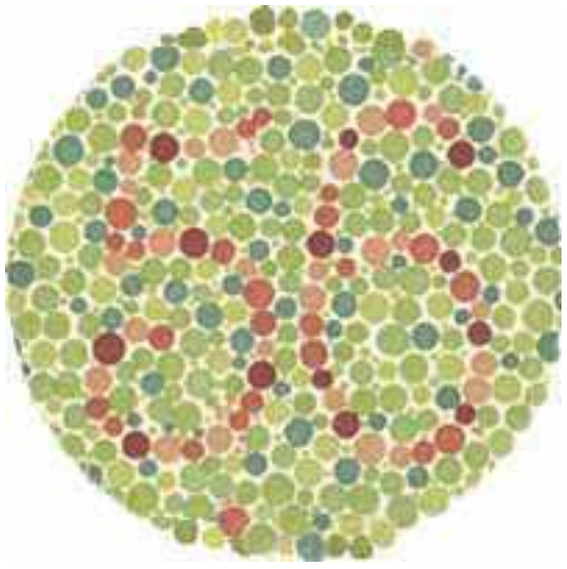
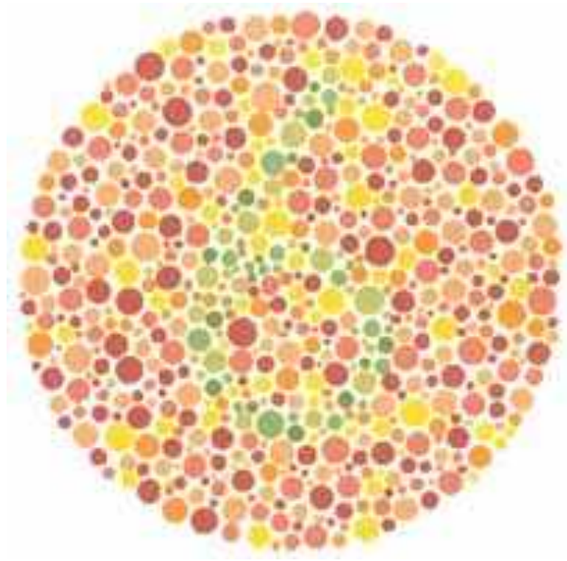
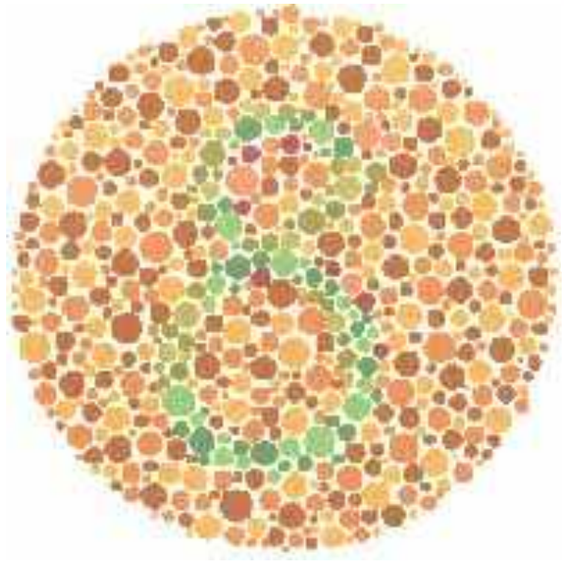
Color Cameras

- Incoming light is described in terms of a *power spectral density*
- *"Color"* isn't a physical property of light
  - It's made up by our eyes and brain!
  - Different types of incoming light can have the same "color"

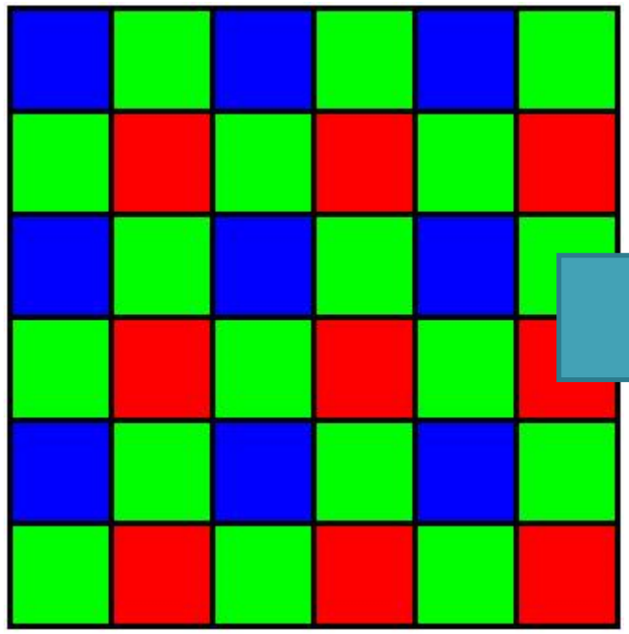


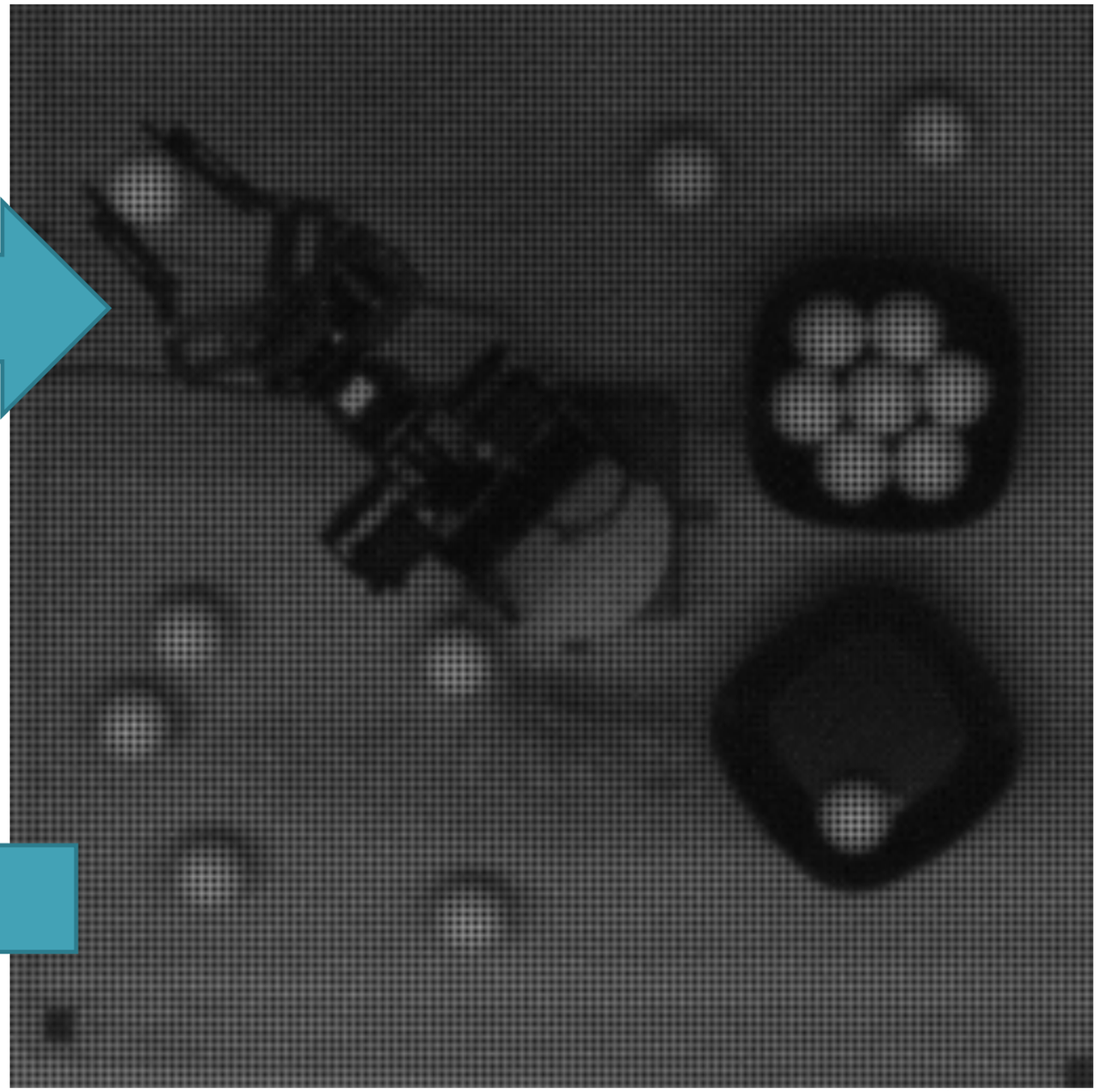
Eye

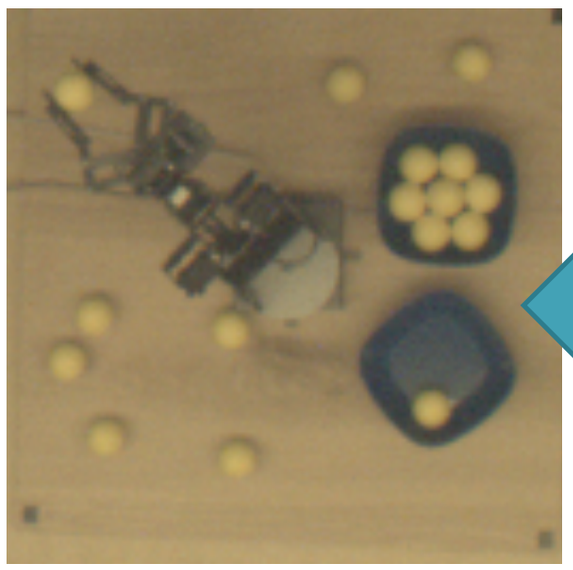
Just for fun...

Bayer Patterns

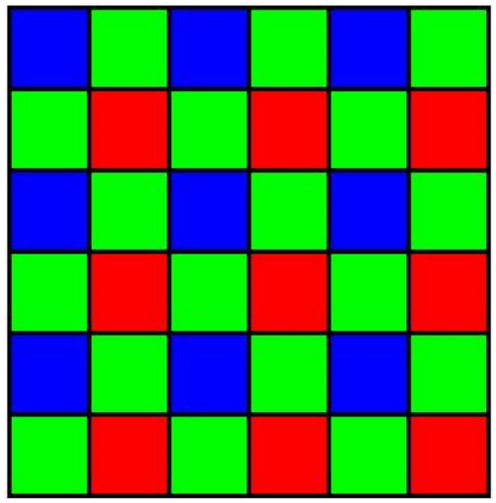






Bayer Patterns

- Why does this matter?
  - At each pixel, two color channels are interpolated based on nearby pixels



- Thus, a color camera is more blurry than a monochrome camera.
  - e.g., Monochrome cameras give slightly better results for AprilTags