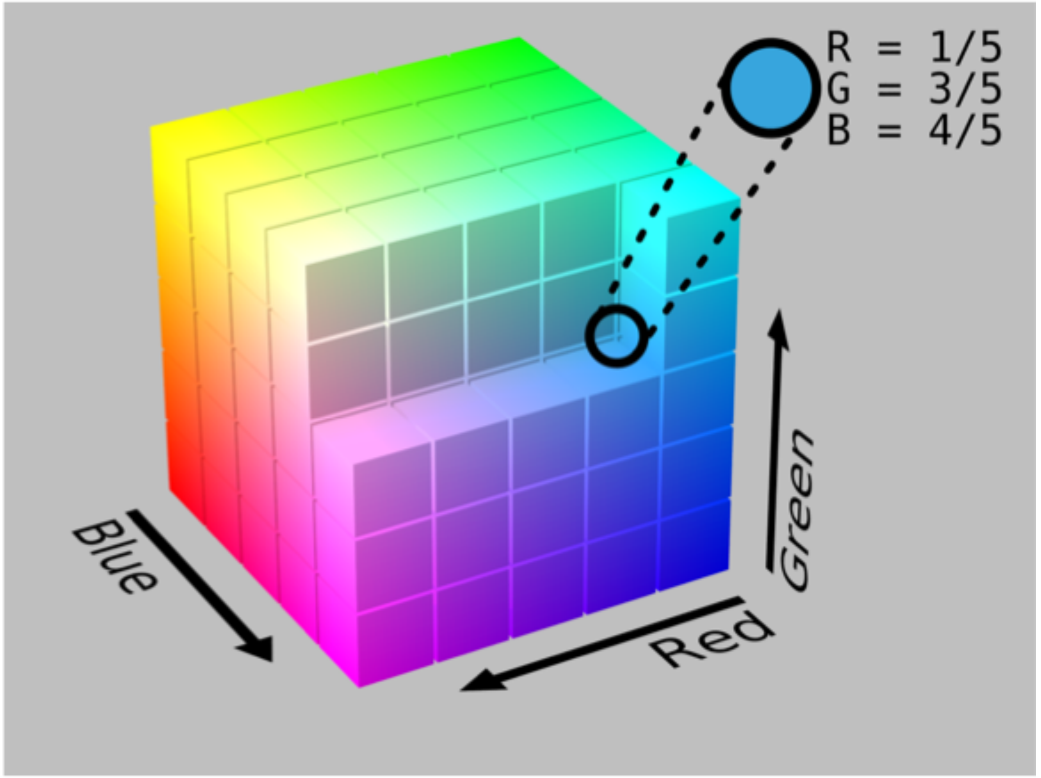
Image Processing & Projective geometry

Arunkumar Byravan

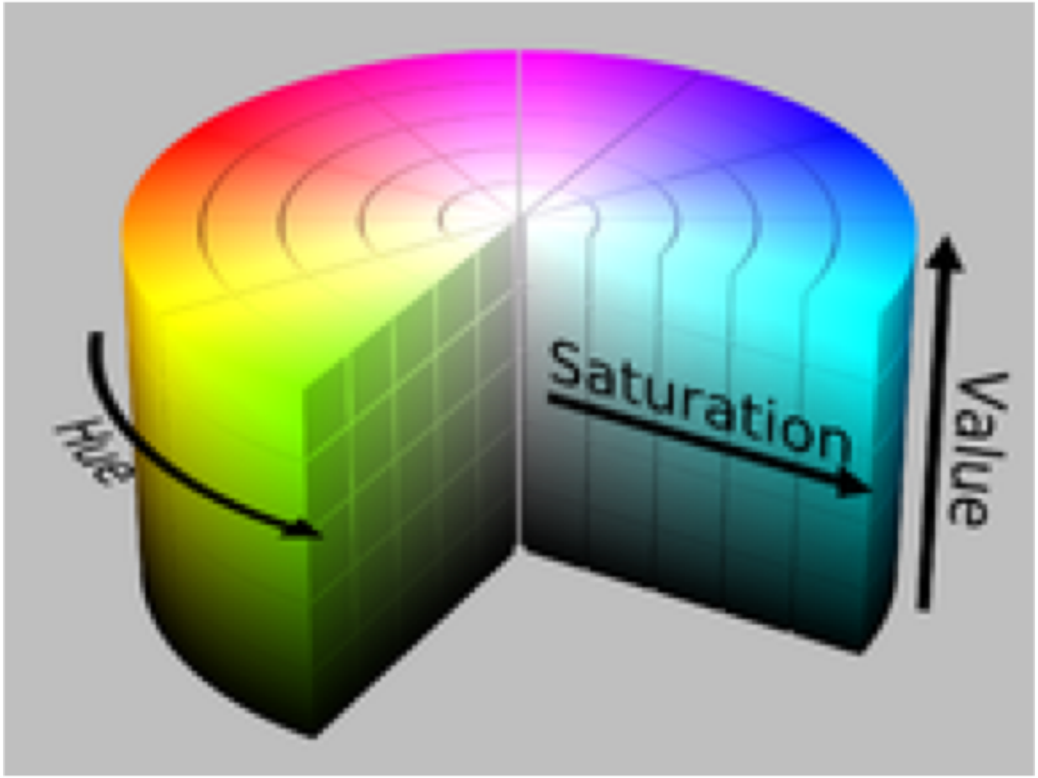
Partial slides borrowed from Jianbo Shi & Steve Seitz

Color spaces

RGB – Red, Green, Blue



HSV – Hue, Saturation, Value

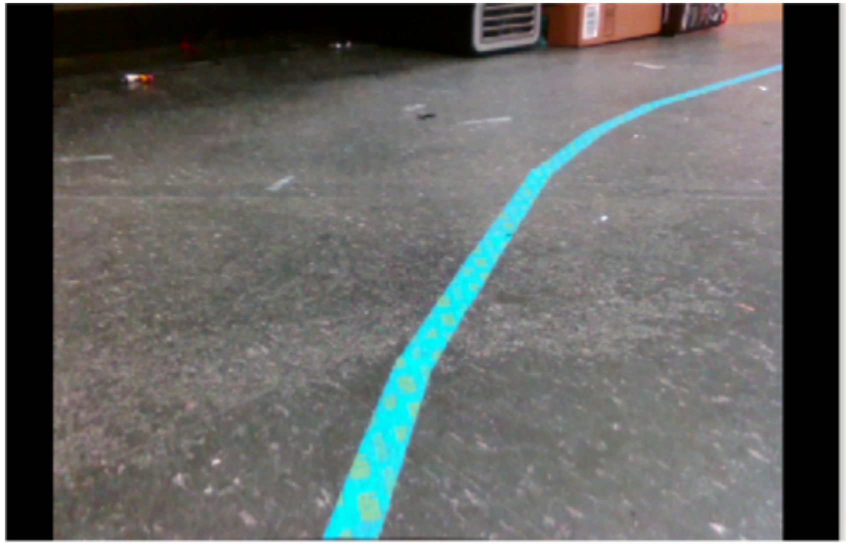


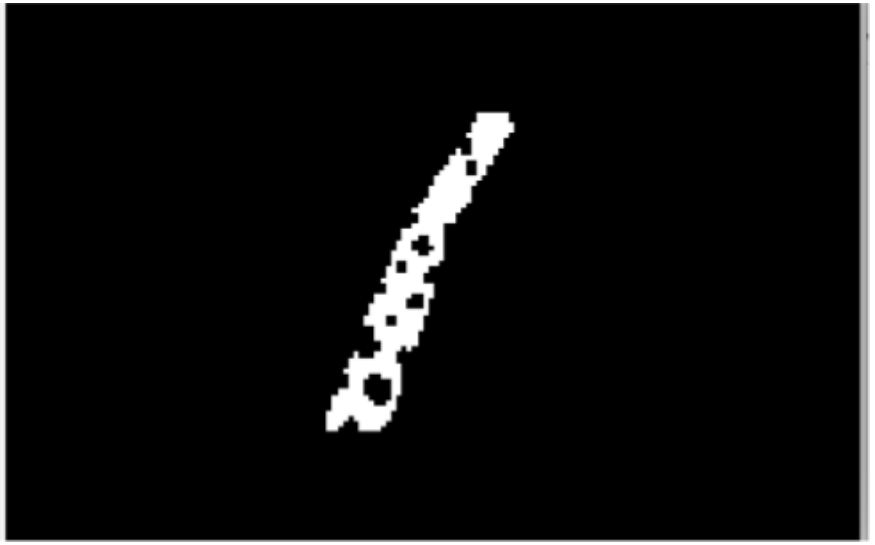
Why HSV?

- HSV separates *luma*, or the image intensity, from *chroma* or the color information
  - Different shades of a color have same hue, different RGBs
- **Advantage:** Robustness to lighting conditions, shadows etc
  - Easy to use for color thresholding!
  - Fast conversion from RGB to HSV *(Python: colorsys)*
- Other relevant color spaces: YCbCr, Lab

HSV example

- GIMP





MPC – Last lecture

- Repeat:
  - Pick a set of controls (Ex: linear velocity, steering angle)
  - Simulate/Rollout using internal model (for time T)
  - Compute error (Ex: distance from desired path, desired fixed point etc.)
  - Choose controls that minimize error
  - Execute control for T’ << T

- Key questions:
  - How to generate rollouts?
  - How to measure error?

MPC - Racecar

- Rollouts – Image templates
  - Pixel tracks of potential paths taken by the car
  - How to generate them?
- Errors – Distance from track in image
  - How to measure error?
  - Other error metrics:
    - Distance to target point
    - Parameterized error (line, spline etc)

How to generate templates?

- Heuristic approach:
  - Generate arcs of varying curvature
  - Associate with a control (linearly interpolate controls & ”invent” a mapping)

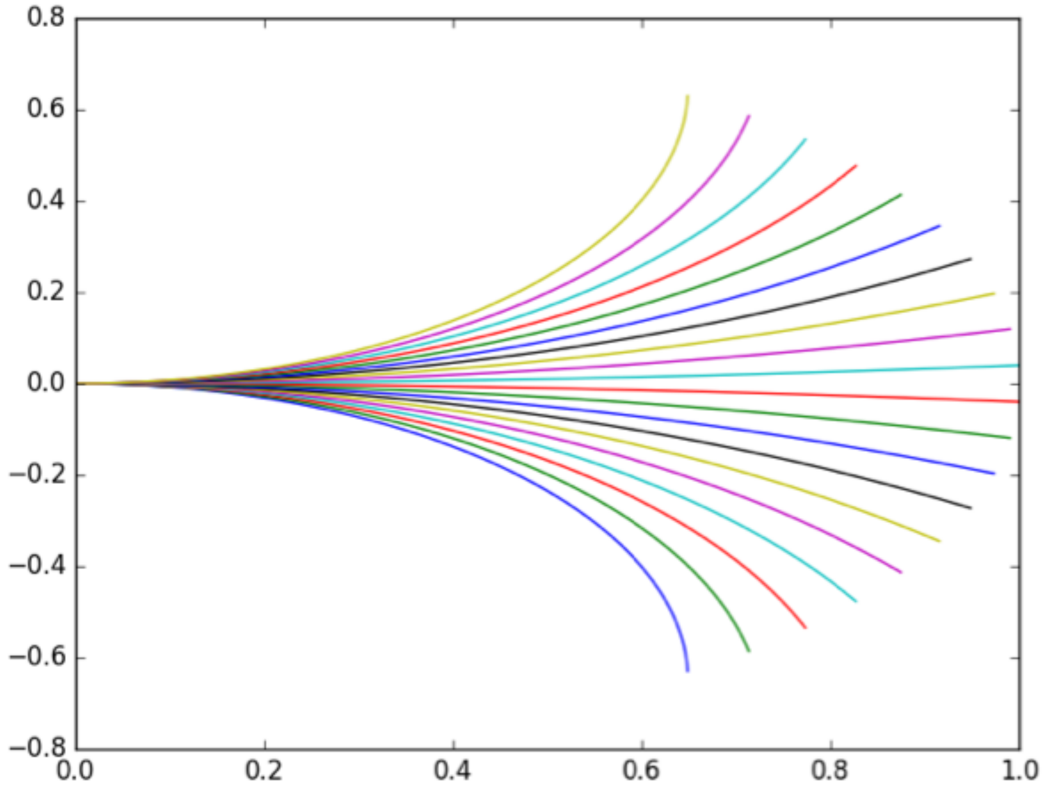
How to generate samples?

- Geometric approach – use motion model

- Generate rollouts from kinematic model based on controls
  - Linearly interpolate controls, rollout a trajectory for fixed horizon “T”

- “Project” rollouts onto image to generate templates
  - Imagine how each rollout would look like when seen from the camera

How to generate templates?

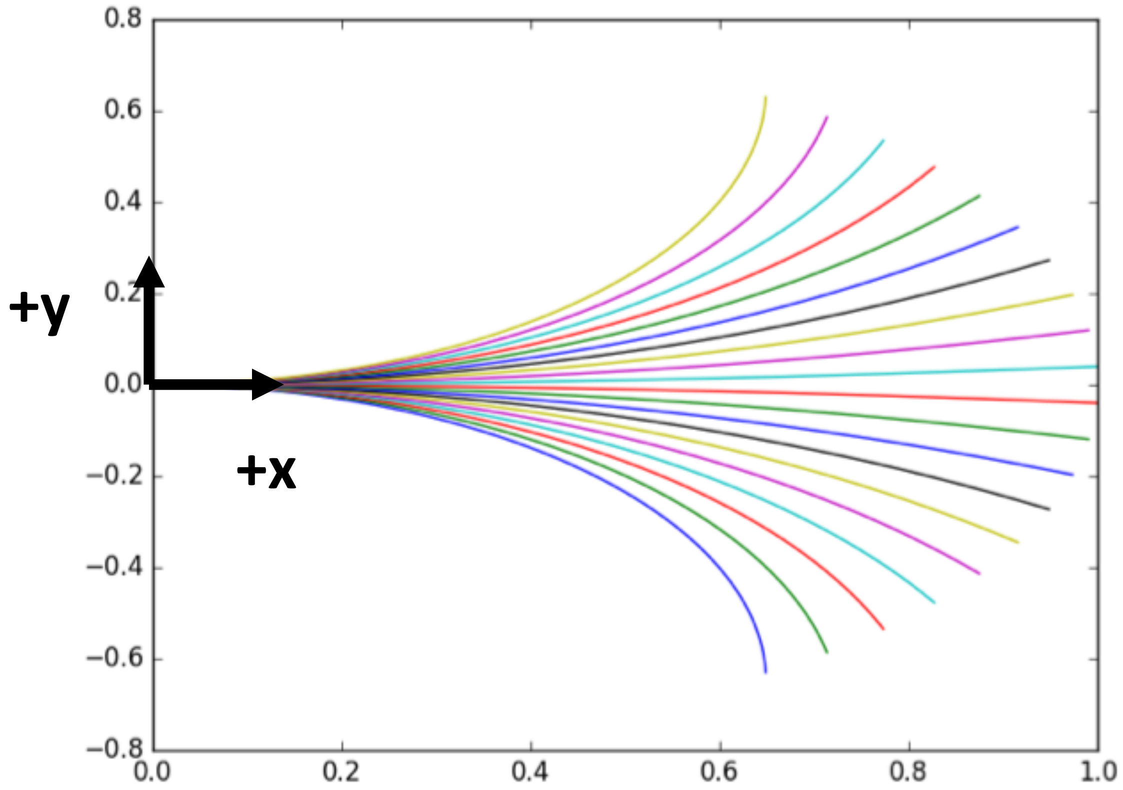


Rollouts from the kinematic motion model, points in frame of car, z = 0

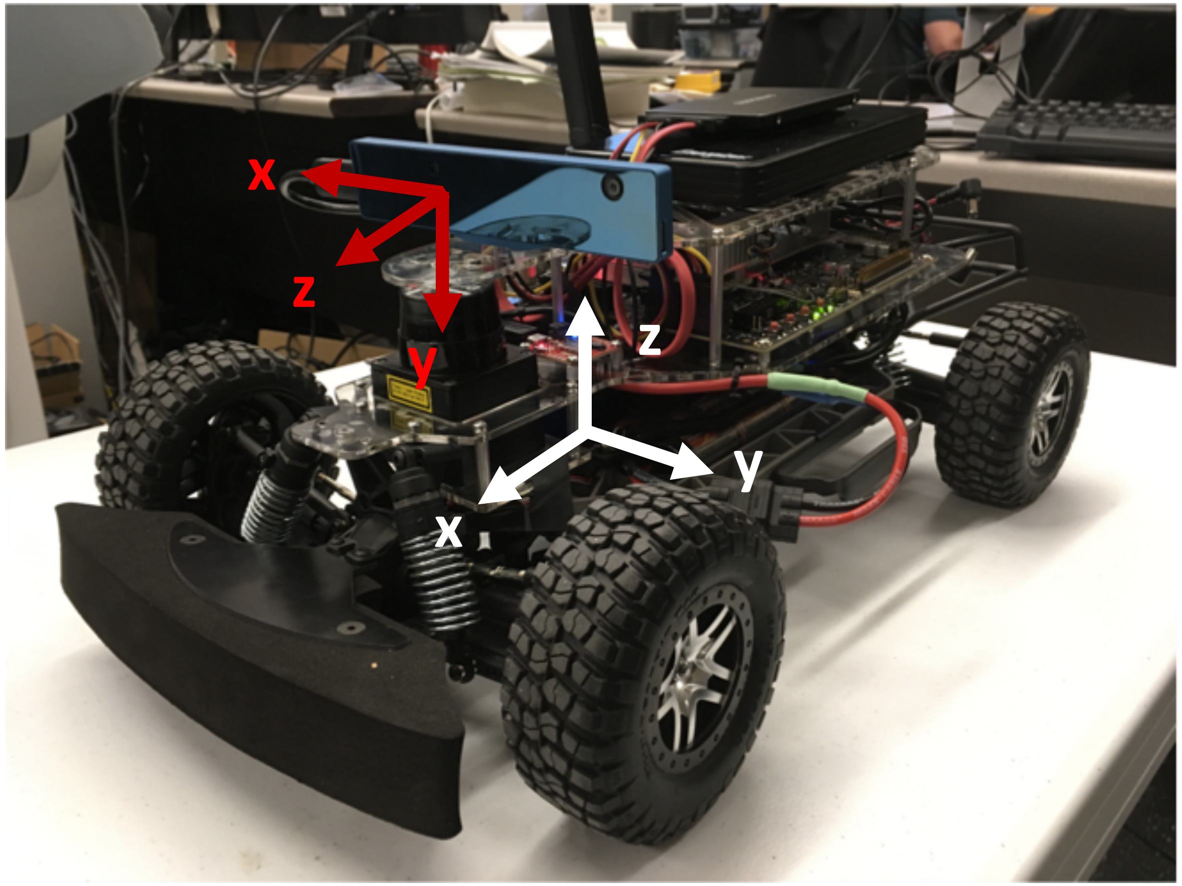
How do we generate image templates from these rollouts?

- Projective geometry!

Camera Extrinsics



Origin frame for rollouts



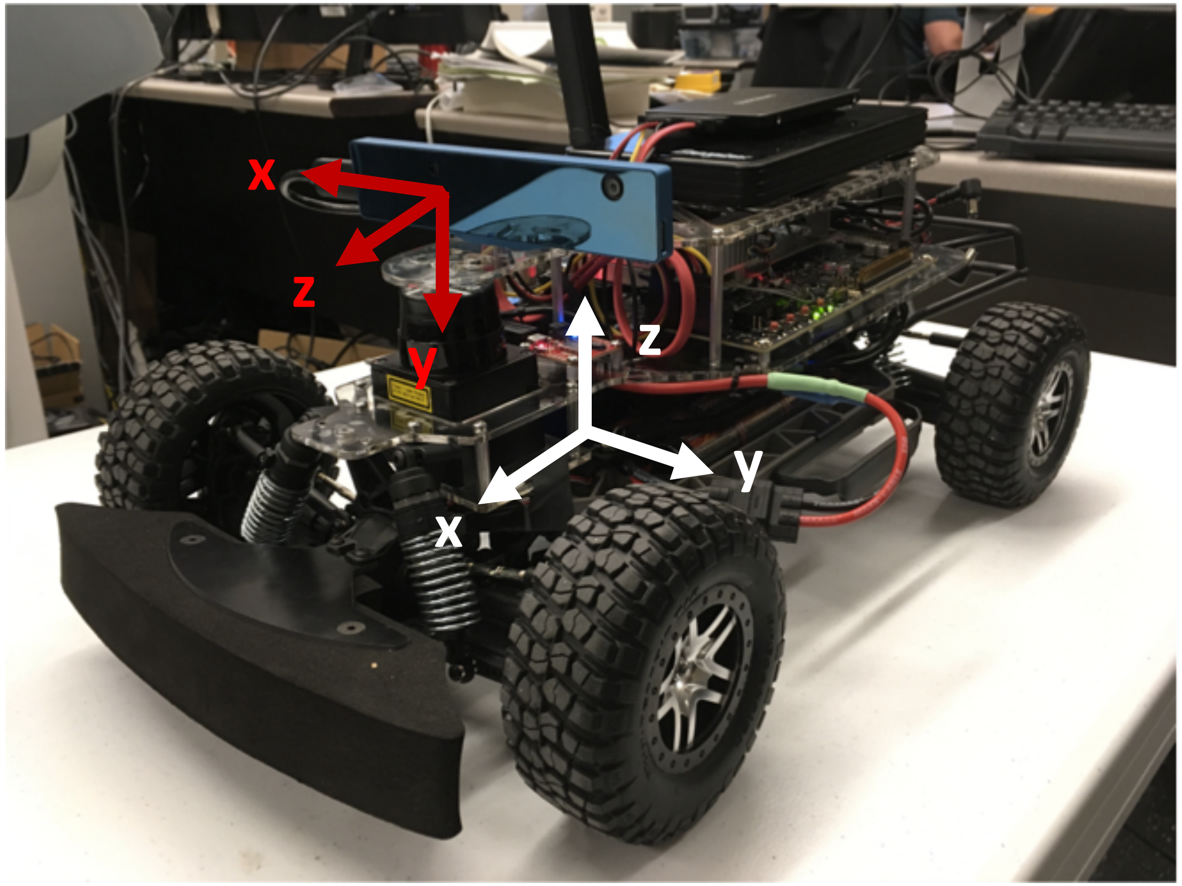
Red is camera frame, white is rollout origin frame

Camera Extrinsics

- We have 3D points in robot frame (white)
- Transform points to camera frame through extrinsics:

Camera frame → $$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$ ← Robot frame

  - **R** = 3x3 rotation matrix
  - **t** = 3x1 translation vector
- For racecar, extrinsics can be measured (and is constant)

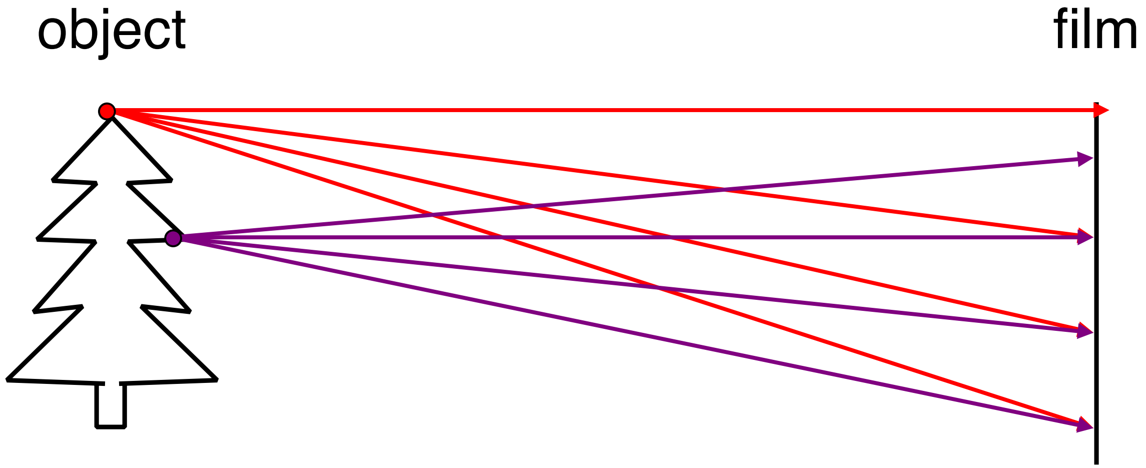


Red is camera frame, white is rollout origin frame

Camera Extrinsics

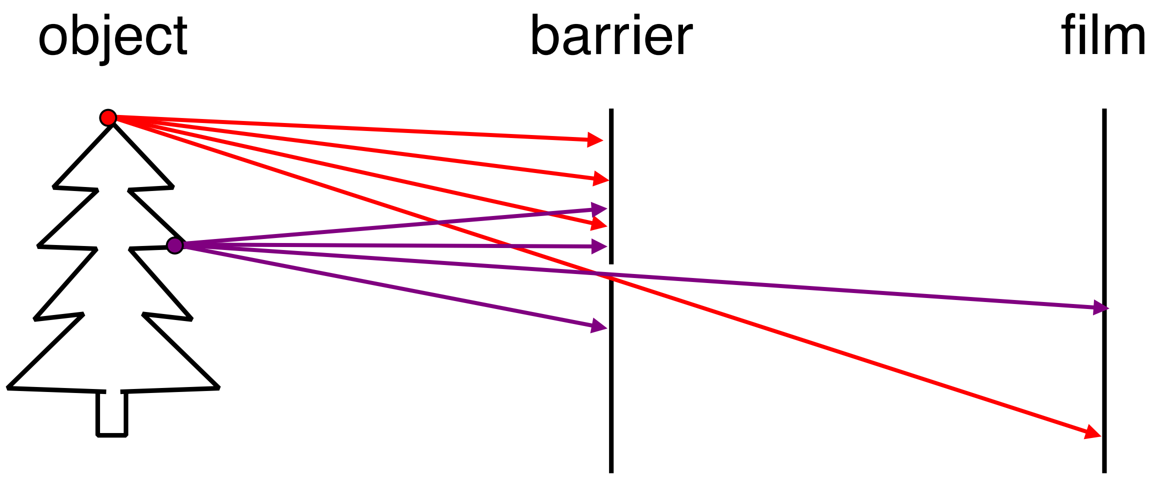
- Extrinsics allow us to transform 3D points to camera frame of reference
- We need to figure out how to get the image of these points as seen by the camera

Image formation process



Slide from Steve Seitz

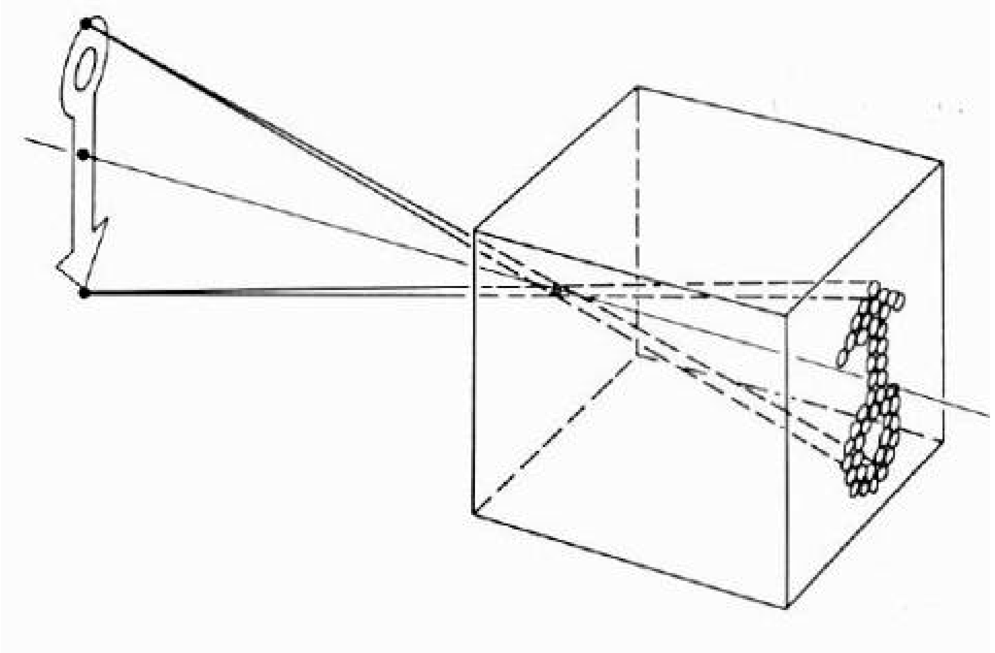
Pinhole camera model



- Add a barrier to block off most of the rays
  - This reduces blurring
  - The opening known as the **aperture**
  - How does this transform the image?

Slide from Steve Seitz

Pinhole camera model



- Pinhole model:
  - Captures **pencil of rays** – all rays through a single point
  - The point is called **Center of Projection (COP)**
  - The image is formed on the **Image Plane**
  - **Effective focal length *f*** is distance from COP to Image Plane

Slide from Steve Seitz

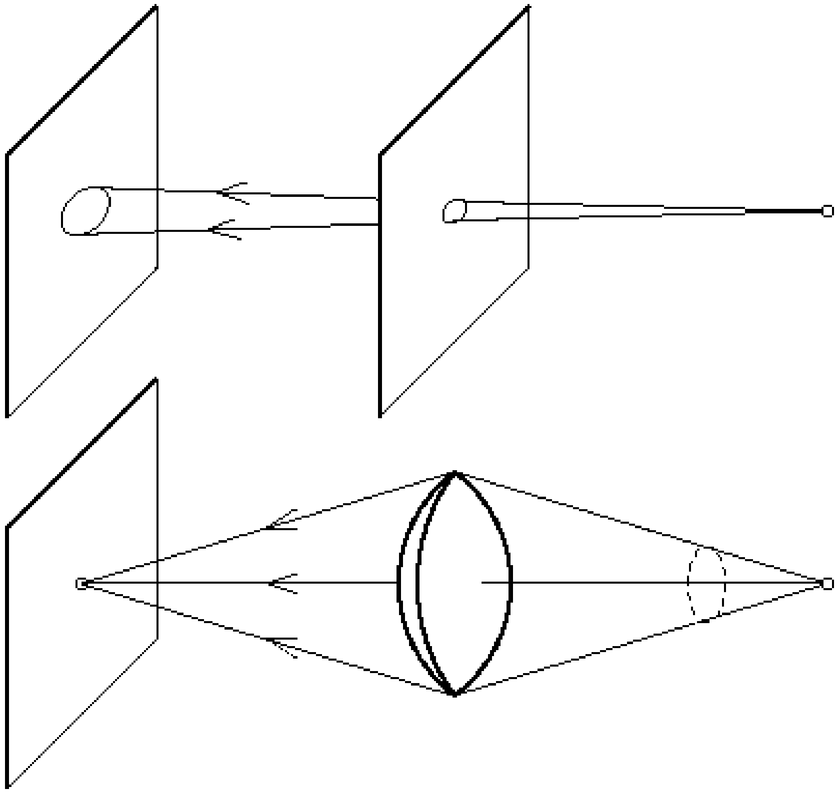
Homemade pinhole camera

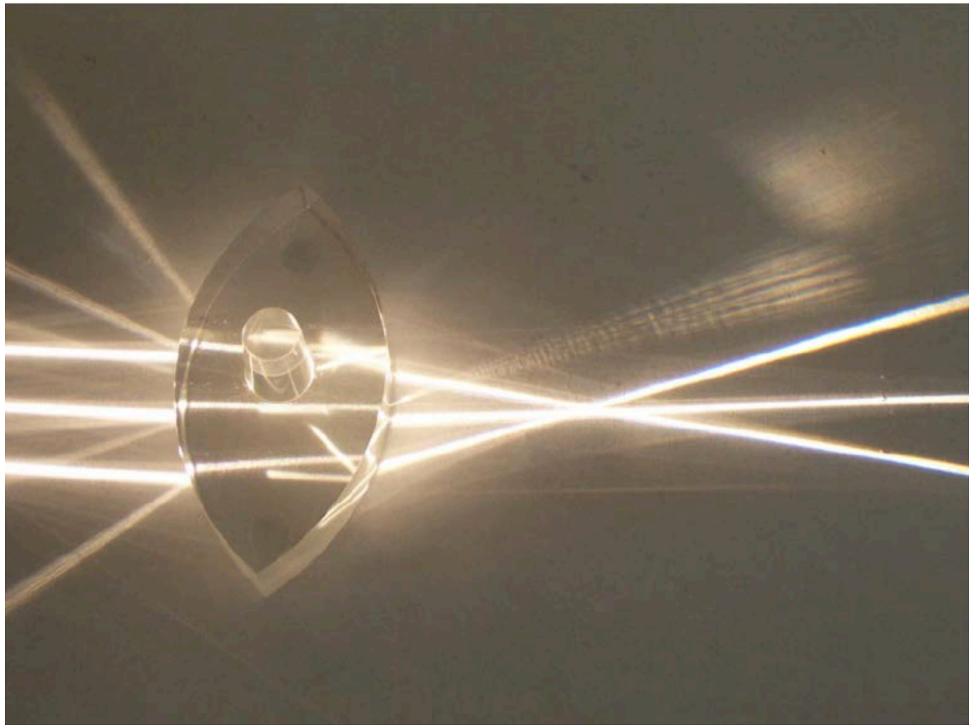
Why so blurry?

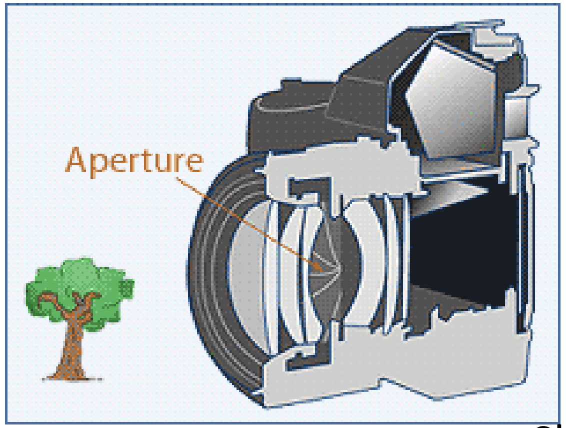
http://www.debevec.org/Pinhole/

Slide from Jianbo Shi

Camera with Lens







Slide from Jianbo Shi

Digital camera

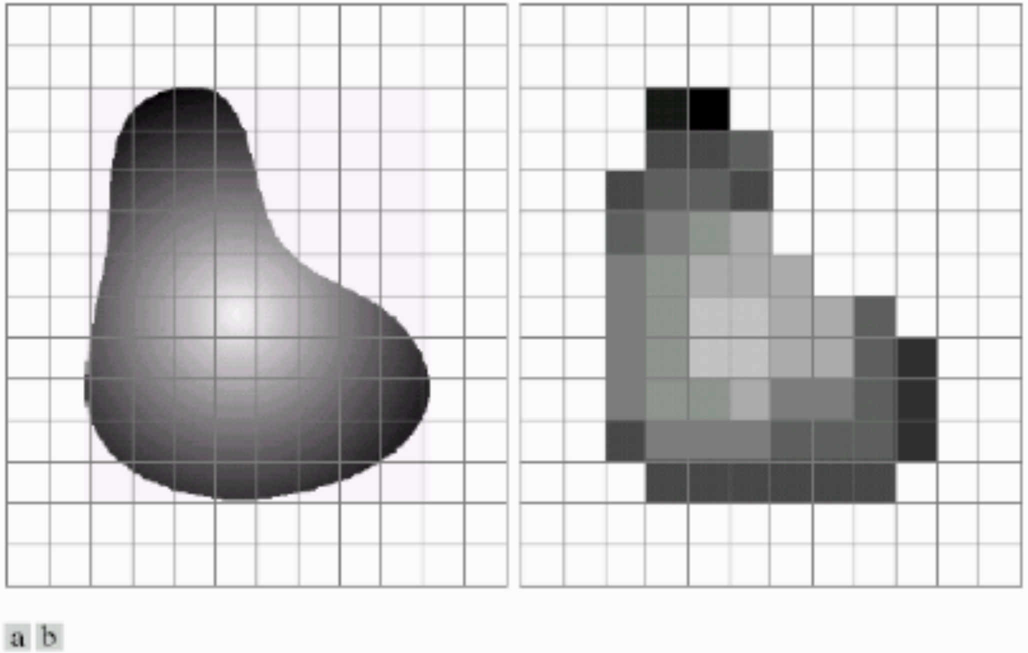
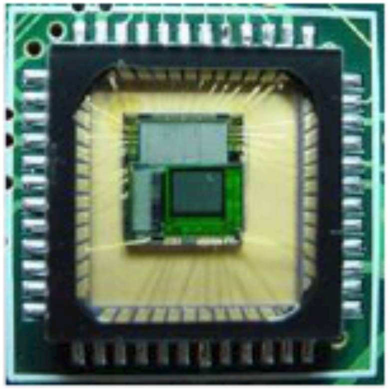


FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.



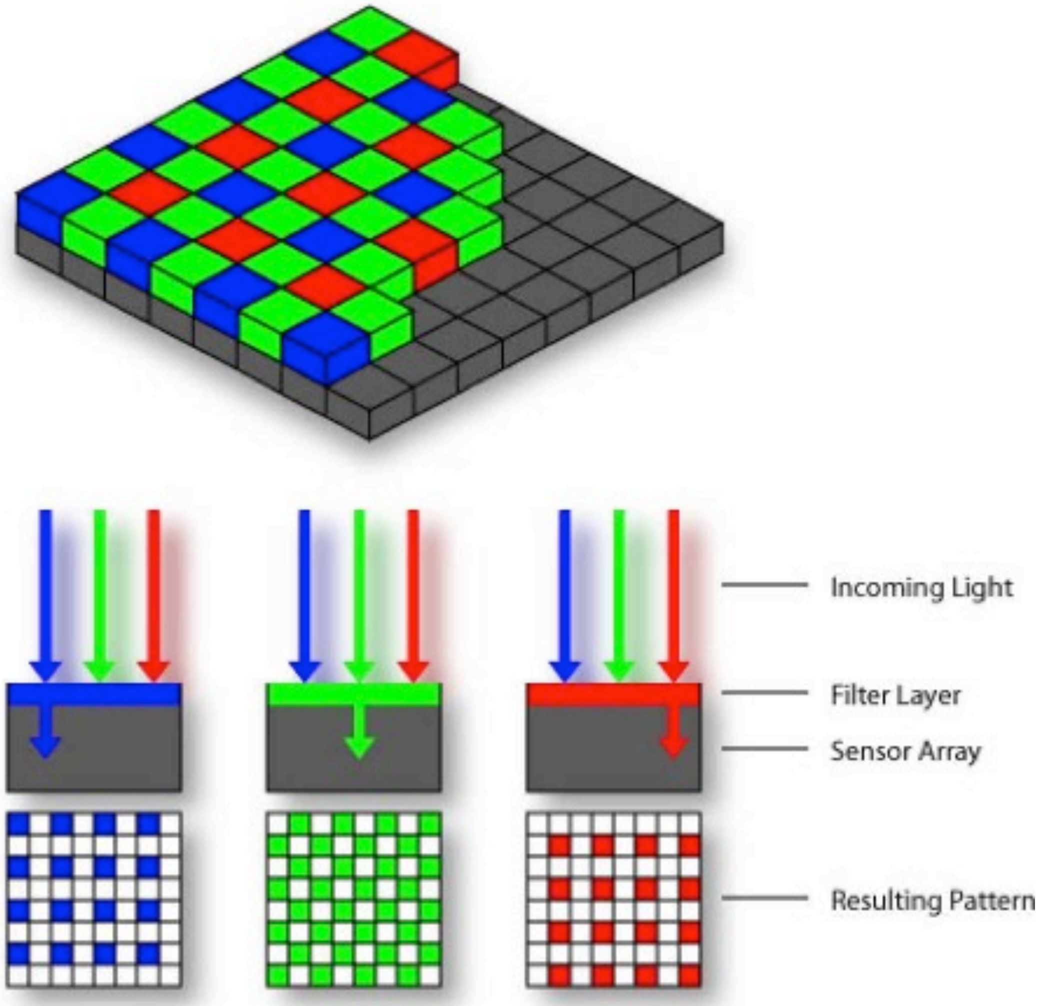
CMOS sensor

Slide from Jianbo Shi

Bayer grid

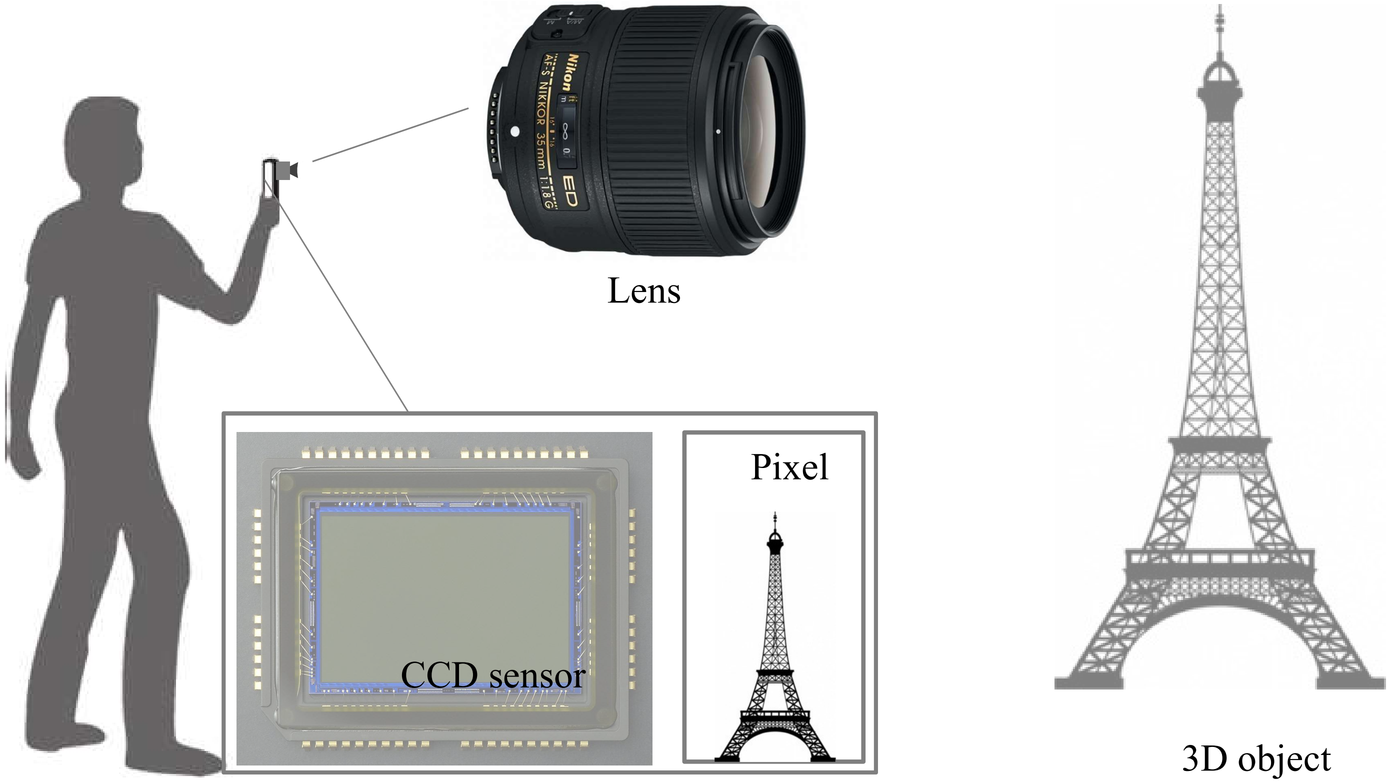
Estimate RGB
at 'G' cels from
neighboring
values

http://www.cooldictionary.com/words/Bayer-filter.wikipedia



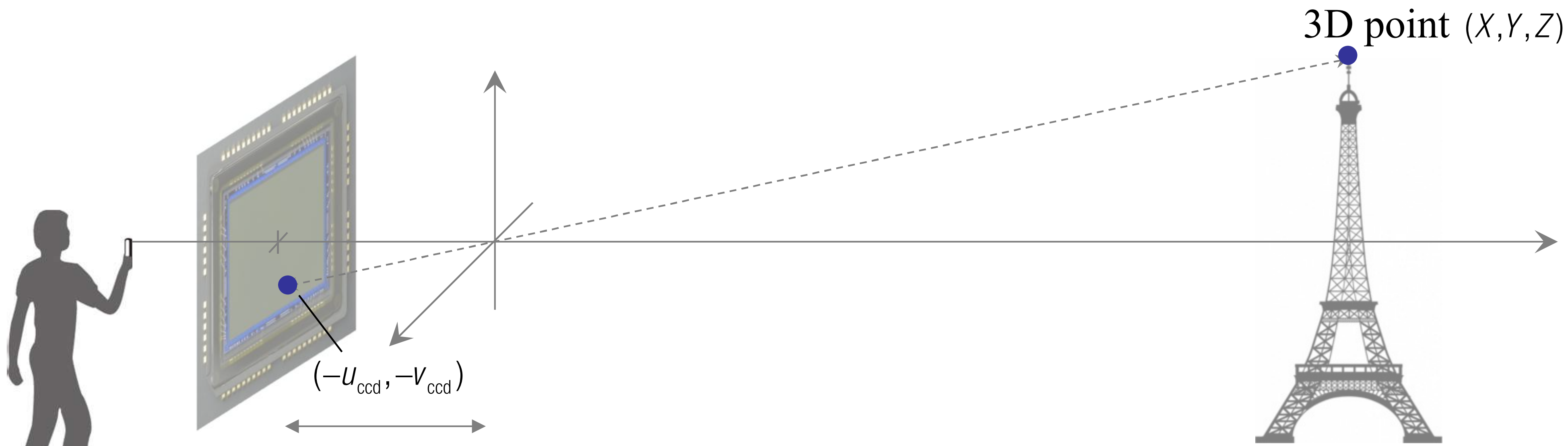
Slide from Jianbo Shi

Projection from 3D to 2D

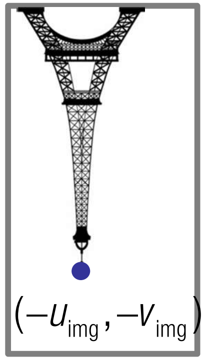


Slide from Jianbo Shi

3D point projection (Metric space)



f_m : Focal length in meter

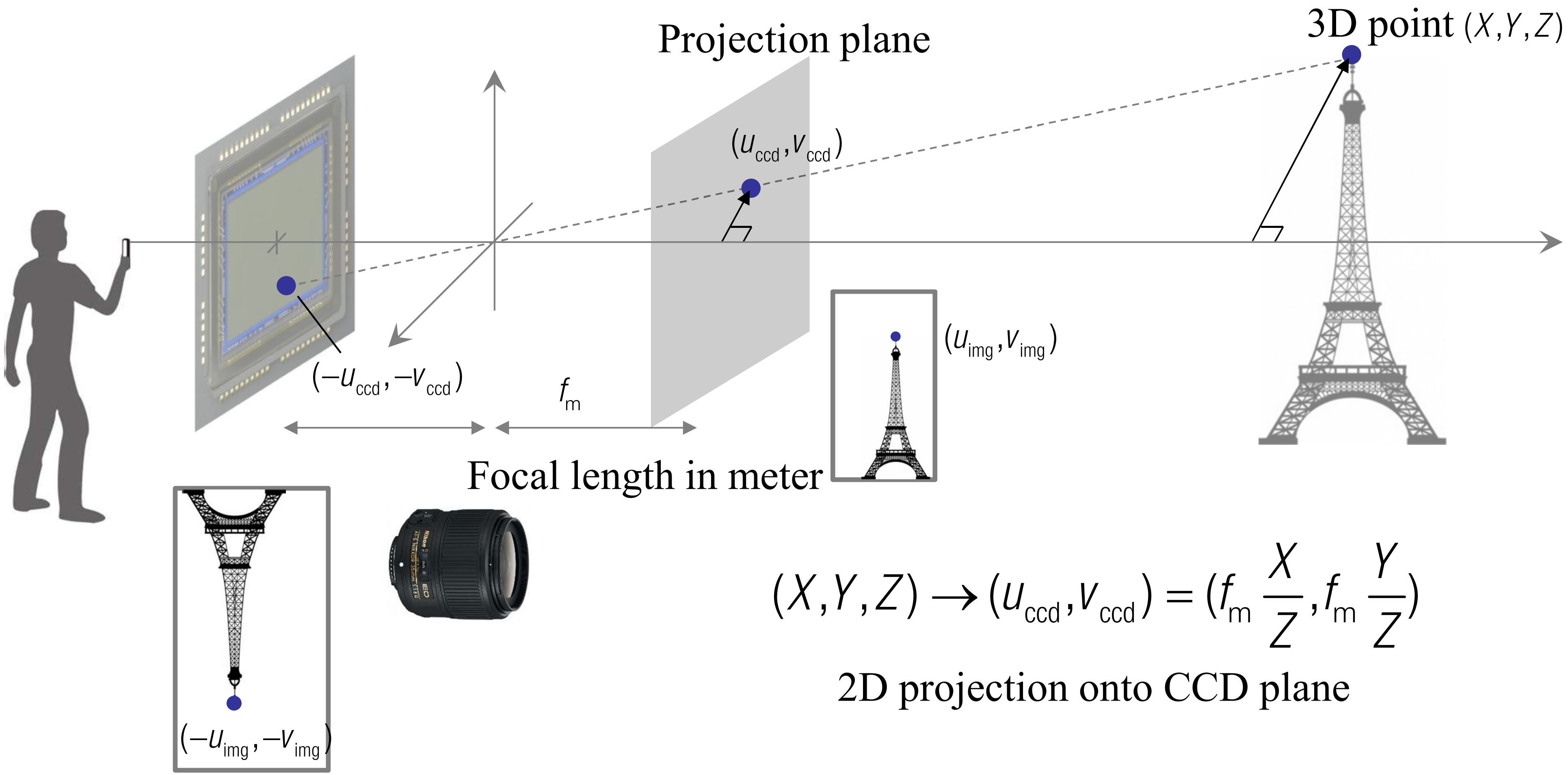


$$(X,Y,Z) \rightarrow (u_{ccd},v_{ccd}) = (f_m \frac{X}{Z}, f_m \frac{Y}{Z})$$

2D projection onto CCD plane

Slide from Jianbo Shi

3D point projection (Metric space)

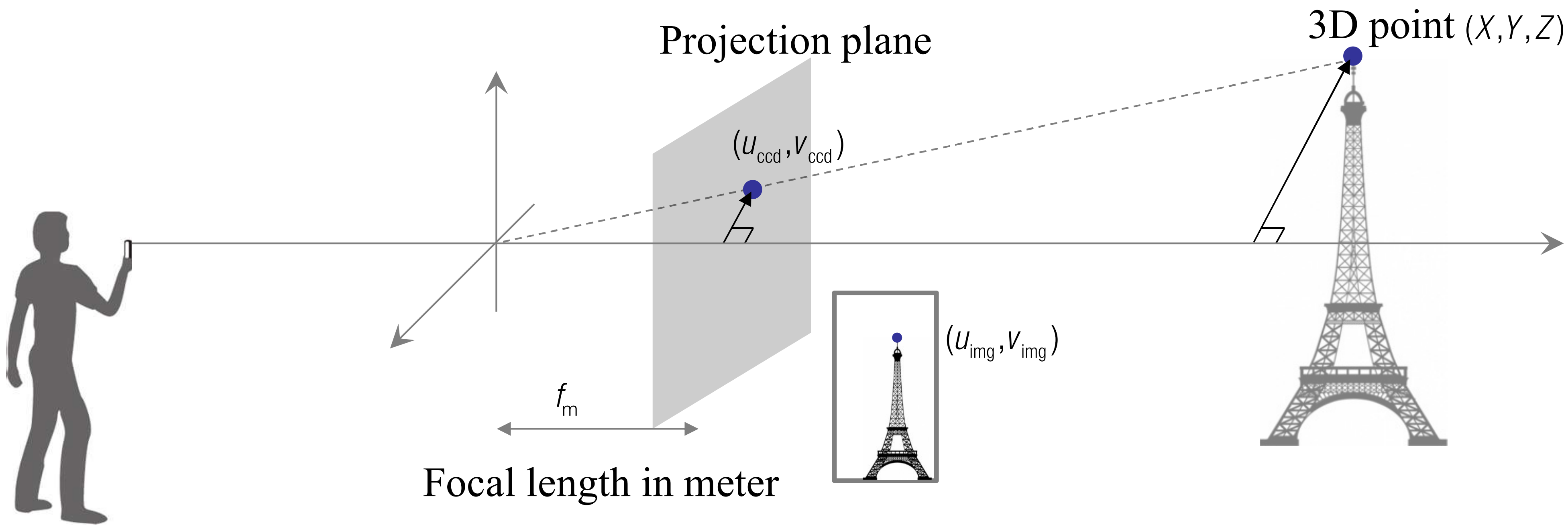


$$(X,Y,Z) \rightarrow (u_{ccd}, v_{ccd}) = (f_m \frac{X}{Z}, f_m \frac{Y}{Z})$$

2D projection onto CCD plane

Slide from Jianbo Shi

3D point projection (Metric space)

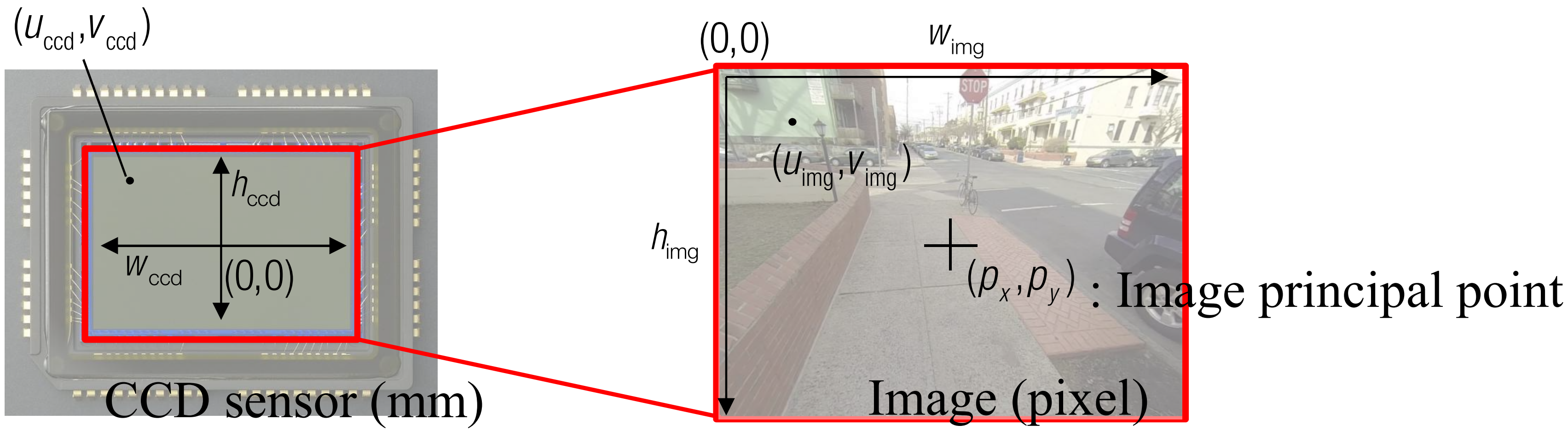


$$(X,Y,Z) \rightarrow (u_{ccd}, v_{ccd}) = (f_m \frac{X}{Z}, f_m \frac{Y}{Z})$$

2D projection onto CCD plane

Slide from Jianbo Shi

3D point projection (Pixel space)

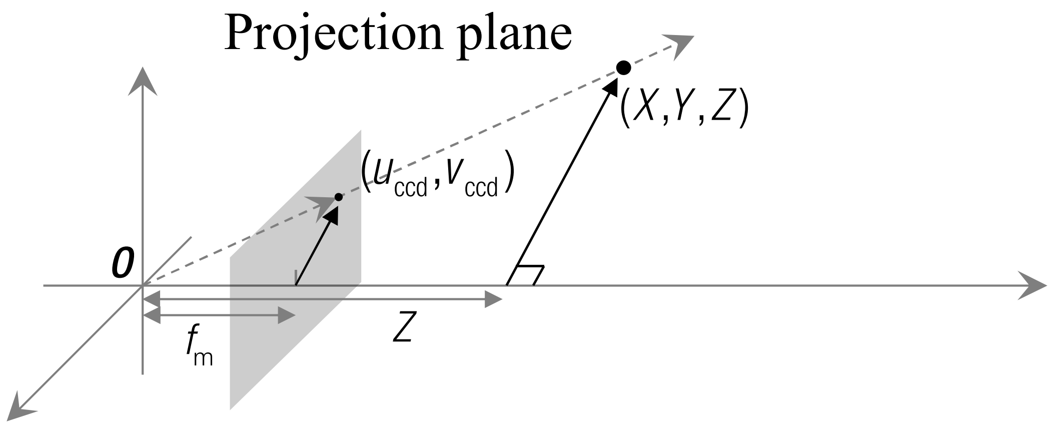


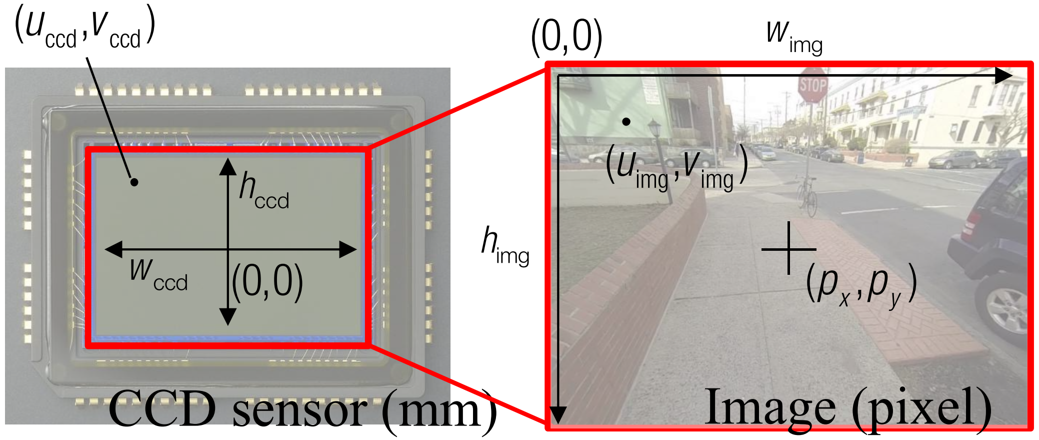
$$\frac{u_{ccd}}{w_{ccd}} = \frac{u_{img} - p_x}{w_{img}} \qquad \frac{v_{ccd}}{h_{ccd}} = \frac{v_{img} - p_y}{h_{img}}$$

$$u_{img} = u_{ccd}\frac{w_{img}}{w_{ccd}} + p_x \qquad v_{img} = v_{ccd}\frac{h_{img}}{h_{ccd}} + p_y$$

Slide from Jianbo Shi

3D point projection (Pixel space)





$$(X,Y,Z) \rightarrow (u_{ccd},v_{ccd}) = (f_m \frac{X}{Z}, f_m \frac{Y}{Z})$$

$$u_{img} = u_{ccd} \frac{w_{img}}{w_{ccd}} + p_x = \underline{f_m \frac{w_{img}}{w_{ccd}}} \frac{X}{Z} + p_x$$

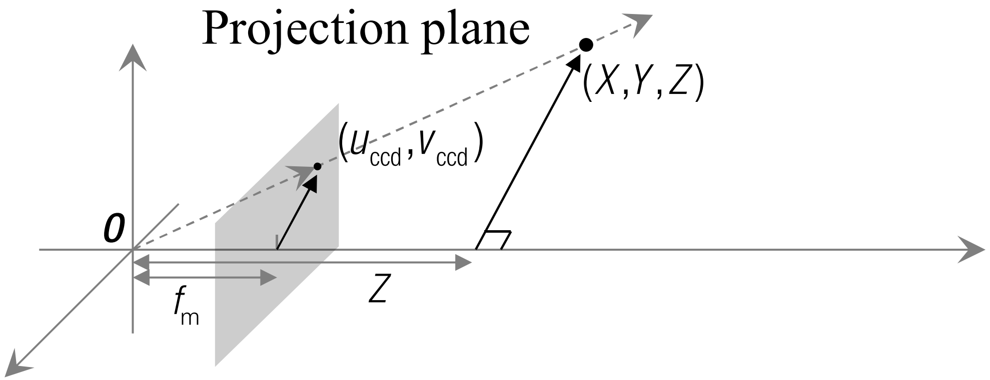
Focal length in pixel

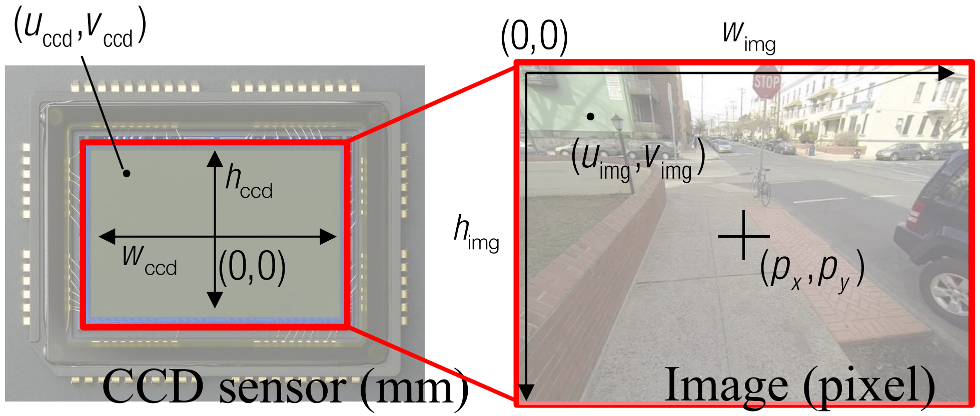
$$v_{img} = v_{ccd} \frac{h_{img}}{h_{ccd}} + p_y = \underline{f_m \frac{h_{img}}{h_{ccd}}} \frac{Y}{Z} + p_y$$

Focal length in pixel

Slide from Jianbo Shi

3D point projection (Pixel space)





$$(X,Y,Z) \rightarrow (u_{ccd}, v_{ccd}) = \left(f_m \frac{X}{Z}, f_m \frac{Y}{Z}\right)$$

$$u_{img} = u_{ccd} \frac{w_{img}}{w_{ccd}} + p_x = f_x \frac{X}{Z} + p_x$$

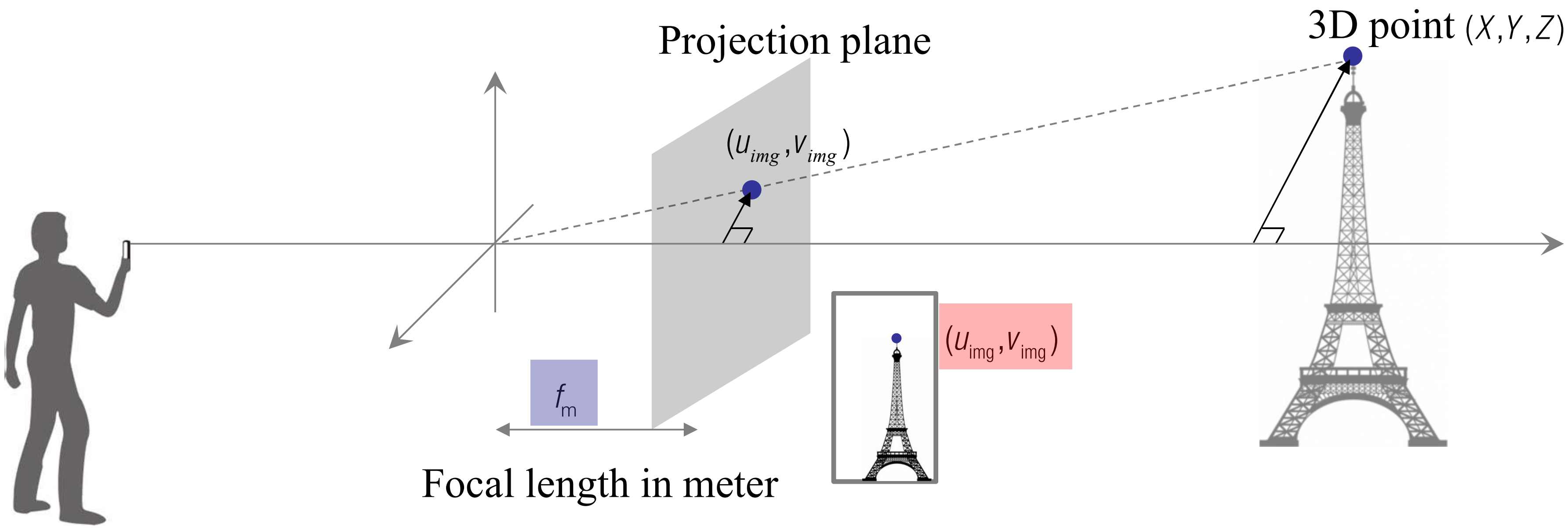
Focal length in pixel

$$v_{img} = v_{ccd} \frac{h_{img}}{h_{ccd}} + p_y = f_y \frac{Y}{Z} + p_y$$

Focal length in pixel

Slide from Jianbo Shi

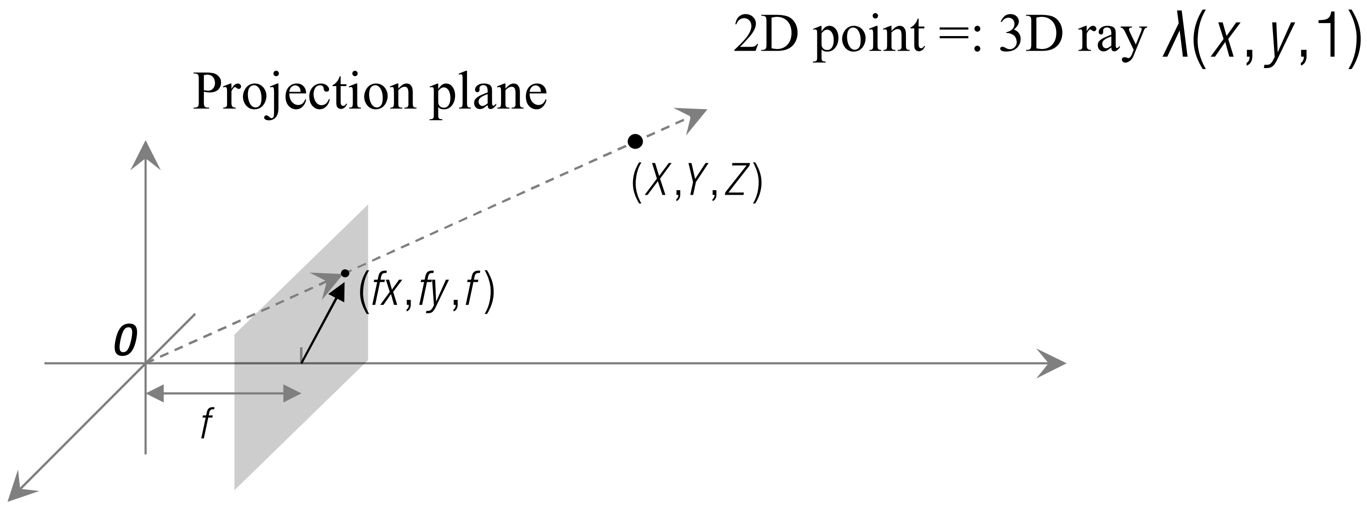
3D point projection (Pixel space)



$$(X,Y,Z) \rightarrow (u_{img}, v_{img}) = \left(f_m \frac{w_{img}}{w_{ccd}} \frac{X}{Z}, f_m \frac{h_{img}}{h_{ccd}} \frac{Y}{Z}\right)$$

Slide from Jianbo Shi

Homogeneous coordinates

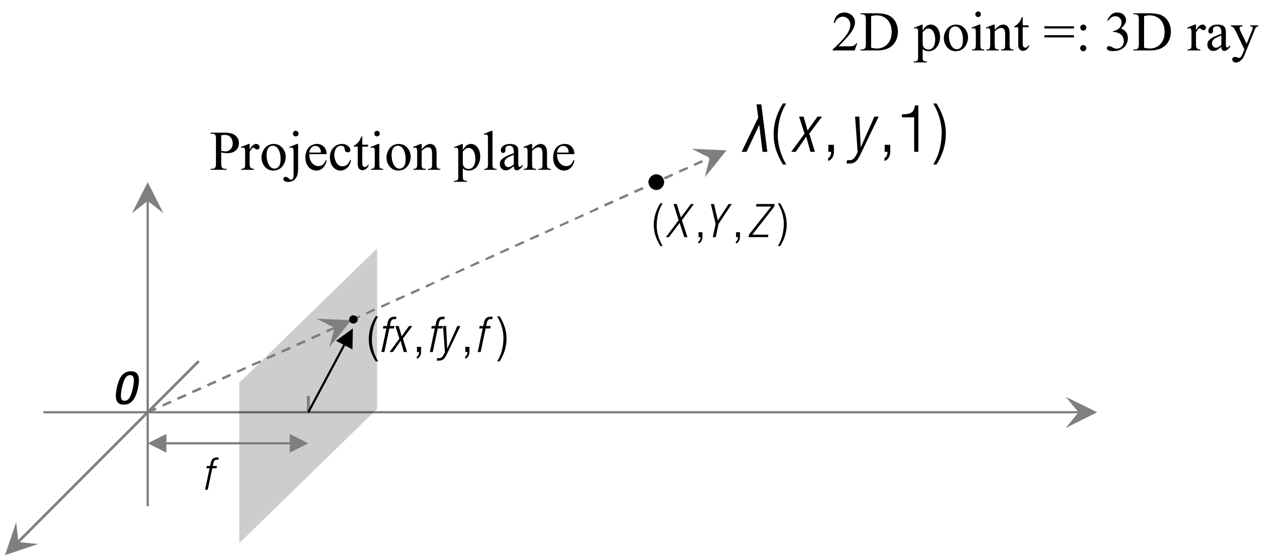


$$(x, y) \rightarrow (x, y, 1)$$
$$= f(x, y, 1)$$
$$= \lambda(x, y, 1)$$

: A point in Euclidean space ($\mathbb{R}^2$) can be represented by a homogeneous representation in Projective space ($\mathcal{P}^2$) (3 numbers).

Slide from Jianbo Shi

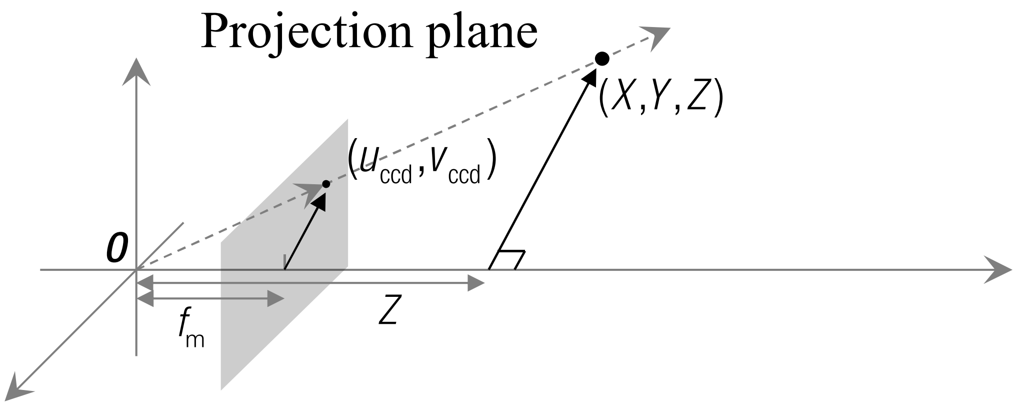
Homogeneous coordinates

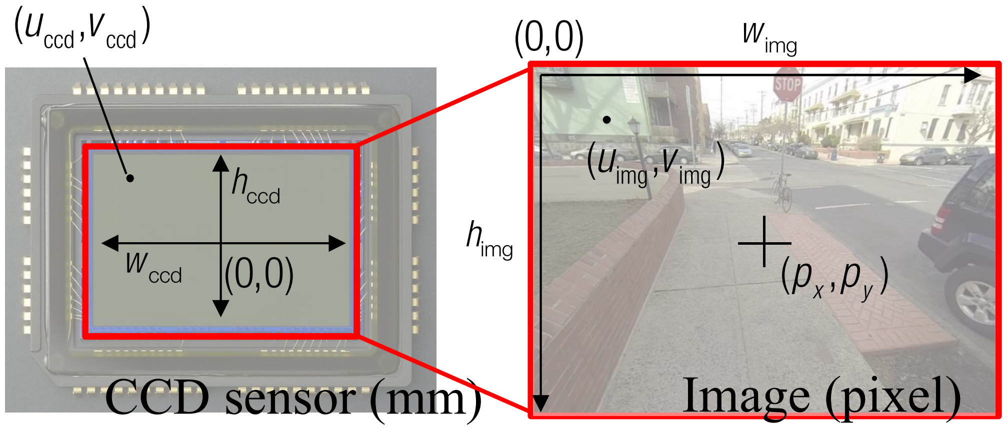


$\underbrace{\lambda(x, y, 1)}_{\text{Homogeneous coordinate}} = (X, Y, Z)$: 3D point lies in the 3D ray passing 2D image point.

Slide from Jianbo Shi

3D point projection (Pixel space)





$$(X,Y,Z) \rightarrow (u_{ccd}, v_{ccd}) = (f_m \frac{X}{Z}, f_m \frac{Y}{Z})$$

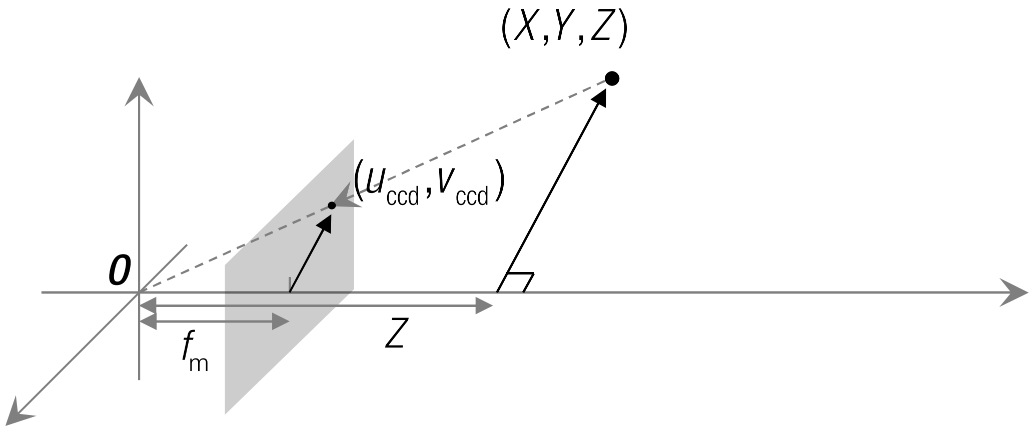
$$u_{img} = f_x \frac{X}{Z} + p_x \qquad v_{img} = f_y \frac{Y}{Z} + p_y$$

$$\lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Homogeneous representation

Slide from Jianbo Shi

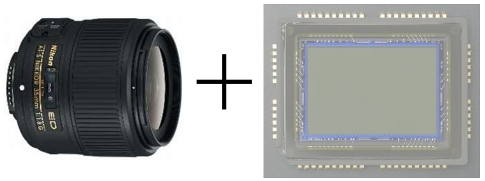
Camera intrinsics



Projection plane

Pixel space | Metric space

$$\lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix}}_{\mathbf{K}} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



Camera intrinsic parameter
: metric space to pixel space

Slide from Jianbo Shi

Putting it all together – Generating templates

- Procedure:
  - Generate rollouts based on kinematic car model (robot frame)
  - Transform points to camera frame based on **camera extrinsics**
  - Project points to pixel space using **camera intrinsics**

Camera frame →

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

← **Robot frame**

Extrinsics

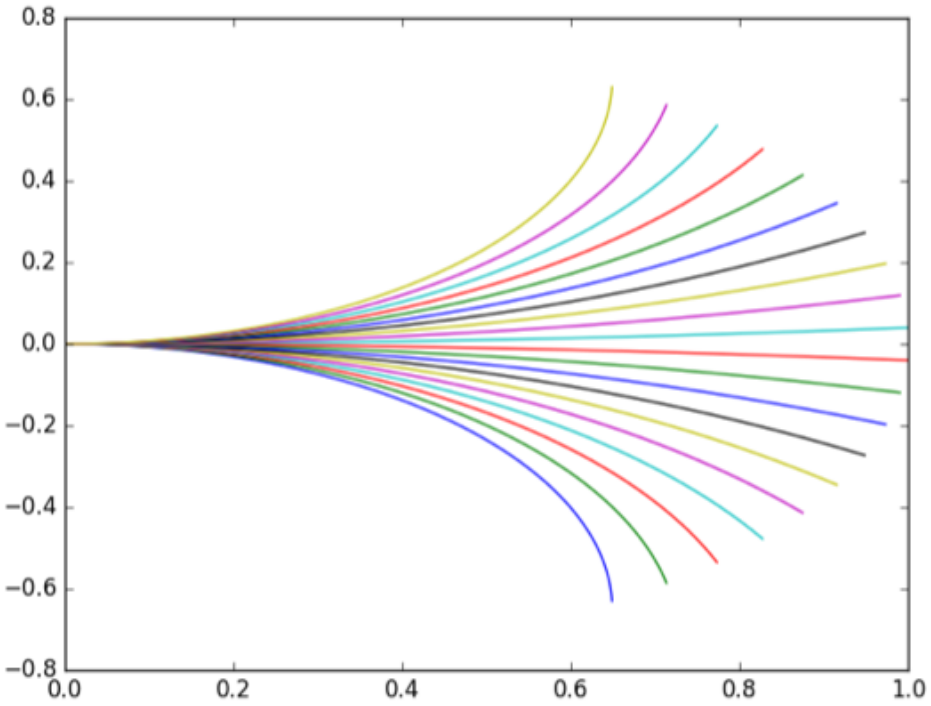
Need to be measured for racecar (Approx values in /tf)

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \begin{pmatrix} \frac{x'}{z'} \\ \frac{y'}{z'} \\ 1 \end{pmatrix}$$

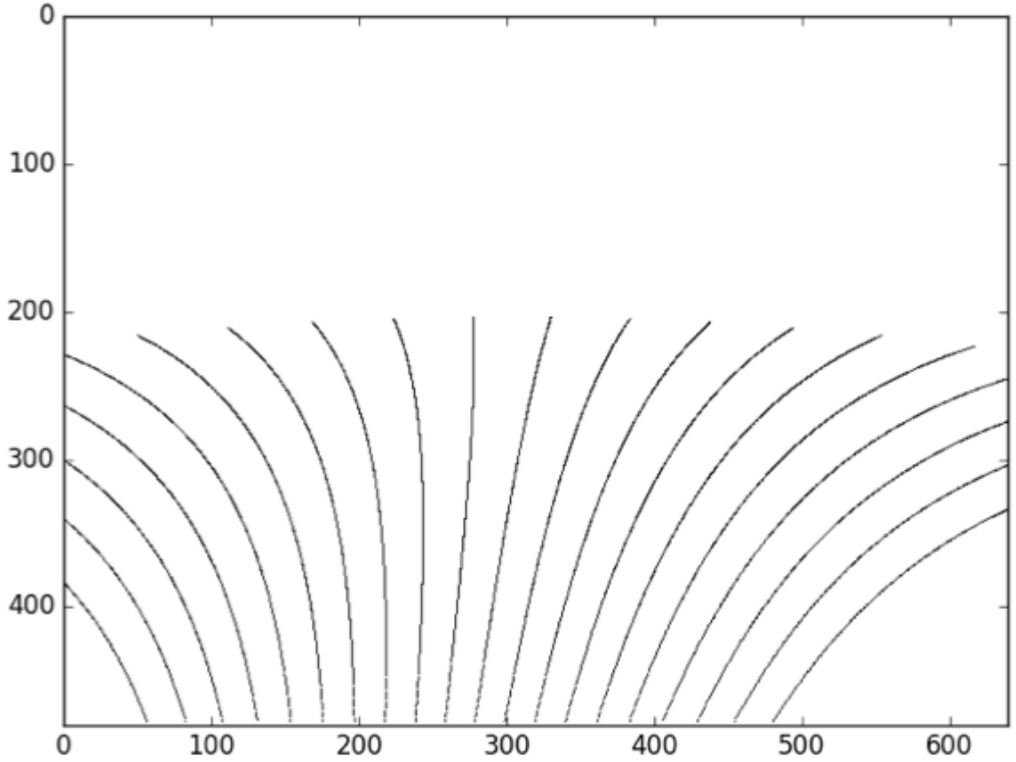
Intrinsics fixed for a camera (for racecar: /camera/color/camera_info)

Intrinsics

Rollout templates



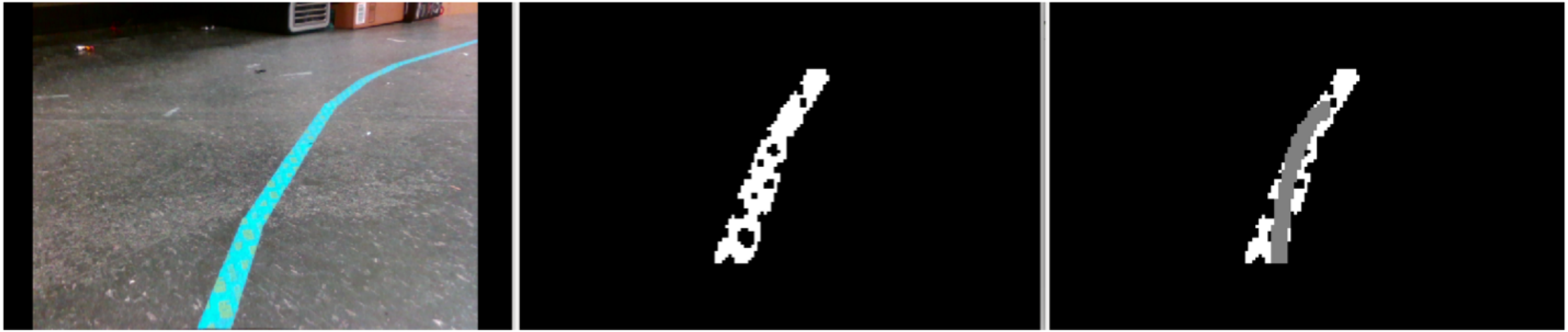
Rollouts from kinematic car model



Projected rollouts

Measuring error (for MPC)

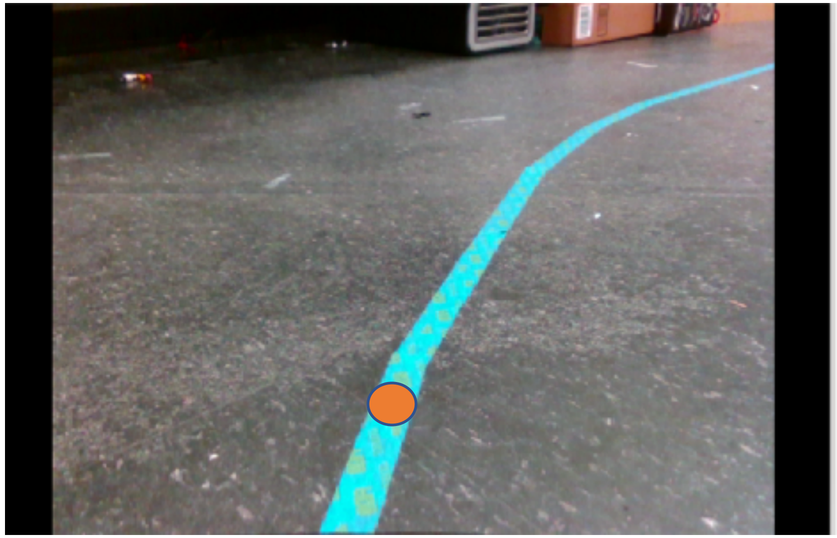
- Template matching using convolution
  - Find template that best matches masked track, choose it



- Issues?

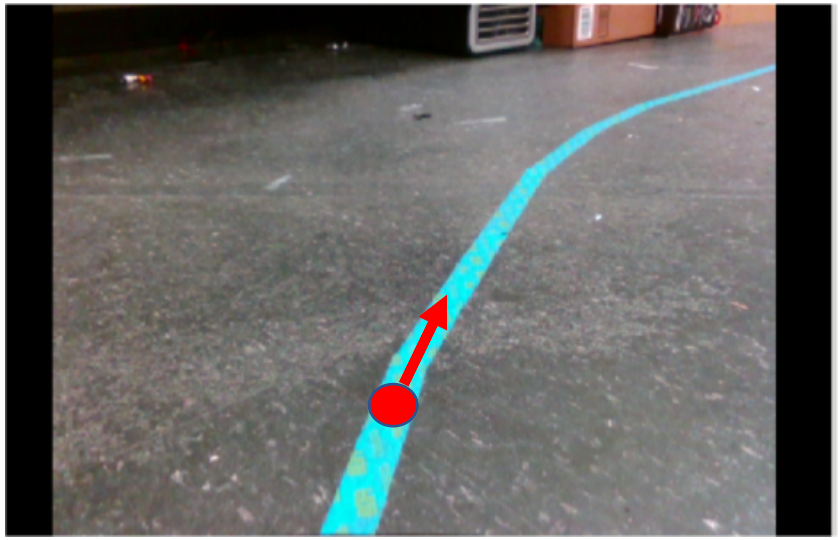
Measuring error (Set-point error)

- Choose set point in image (similar to PID)
  - Find template that gets you closest to set point, choose it



Measuring error (Set-point + Direction error)

- Choose set point in image along with heading (based on track)
  - Find template that gets you closest to set point while oriented correctly
  - Keep track of heading in templates

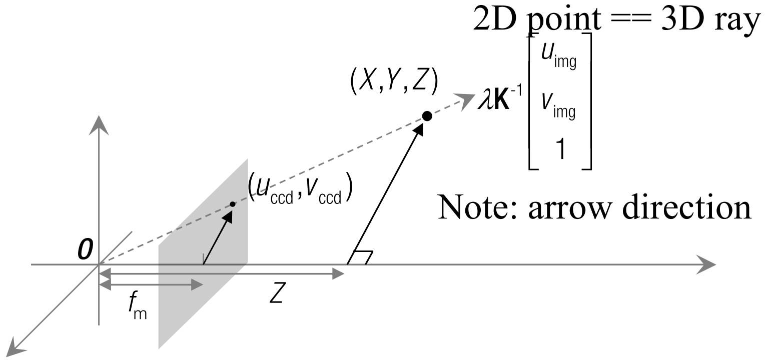


Measuring error (3D error)

- Instead of generating pixelized templates, project masked track (or set point) back to 3D

- How?
  - Each pixel corresponds to ray in 3D
  - We know that all pixels on track lie on ground plane (known)
  - Solve for ray-plane intersection

- Advantage: Reason in 3D!



Projection plane

Pixel space | Metric space

$$\lambda \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} f_x & & p_x \\ & f_y & p_y \\ & & 1 \end{bmatrix}$$

$$\lambda \mathbf{K}^{-1} \begin{bmatrix} u_{img} \\ v_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

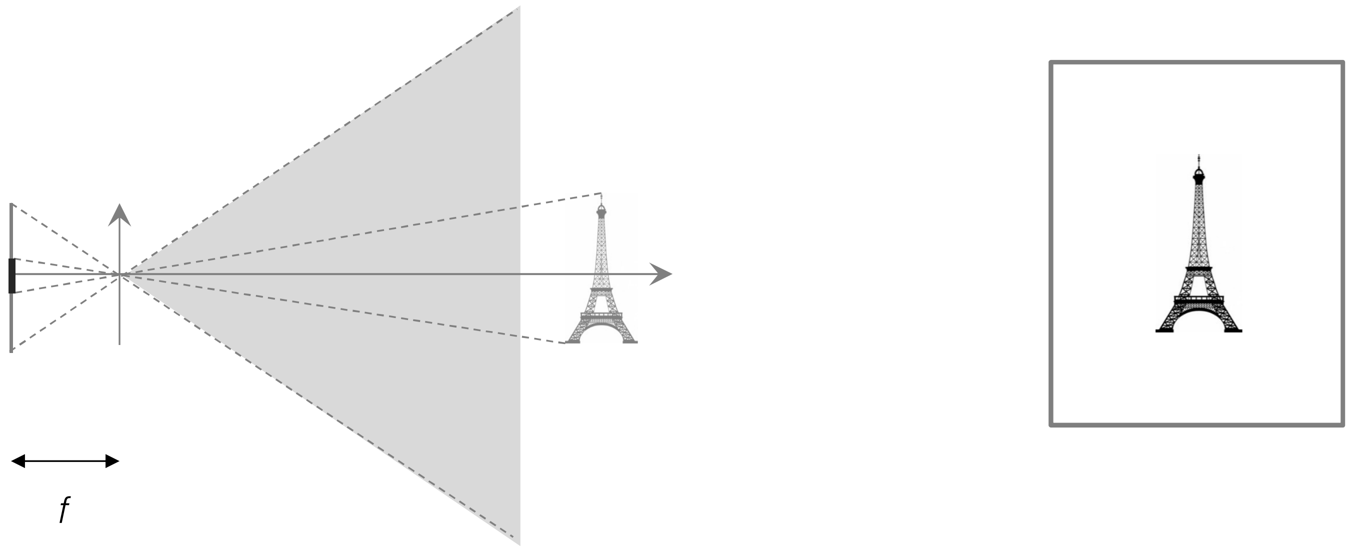
3D ray

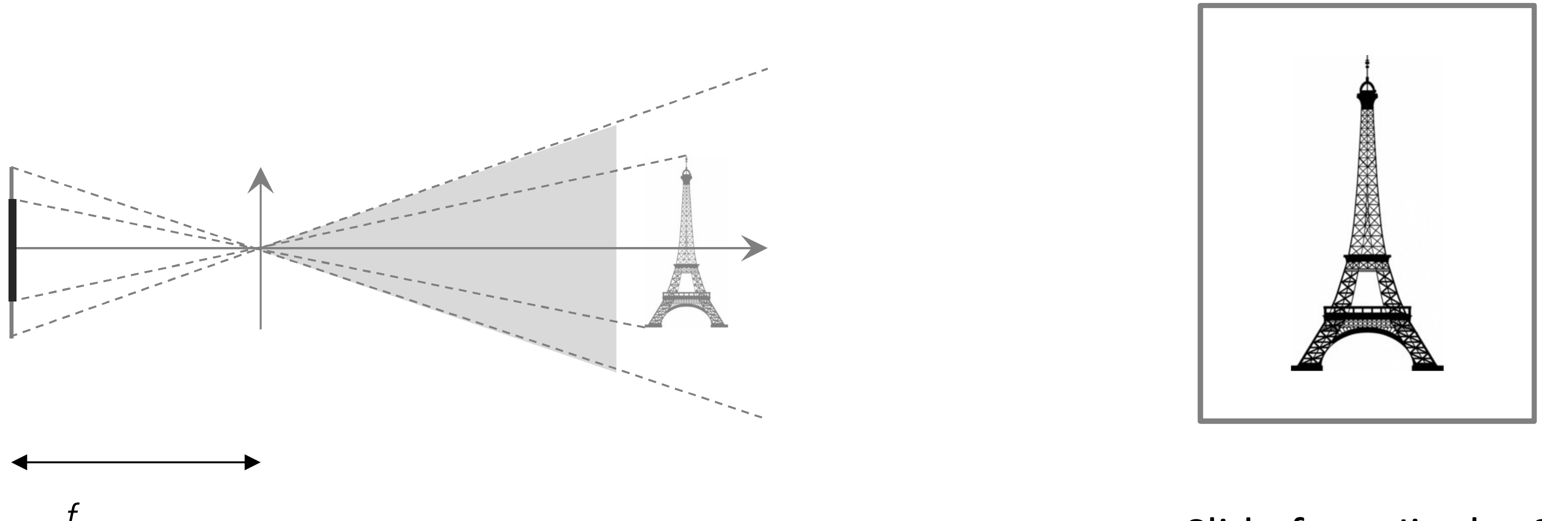
The 3D point must lie in
the *3D ray* passing through the origin and 2D image point.

Measuring error (fancy error metric)

- Fit a line or curve to pixel/3D track points & your rollouts
  - Compare the errors in parametric space (line / curve co-efficients)

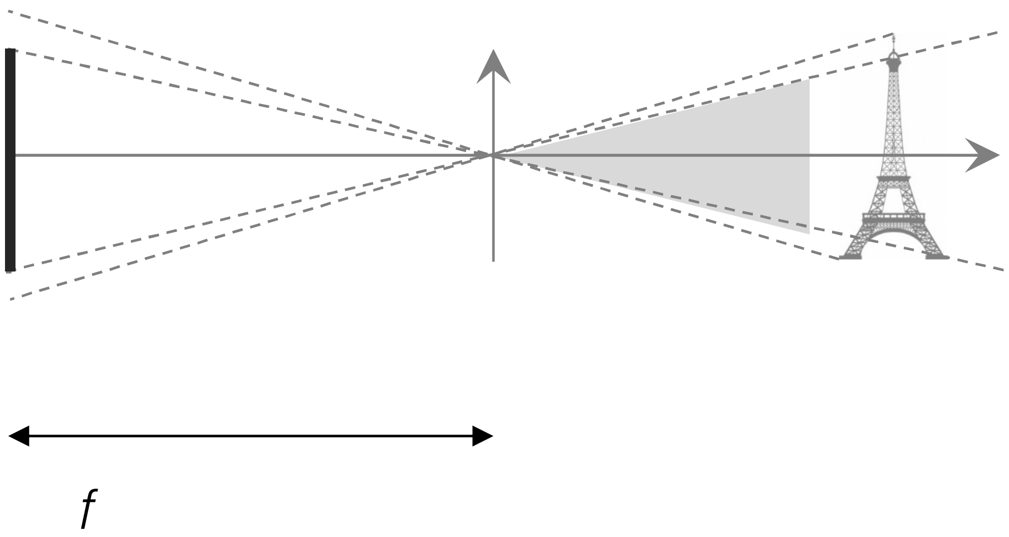
Focal length

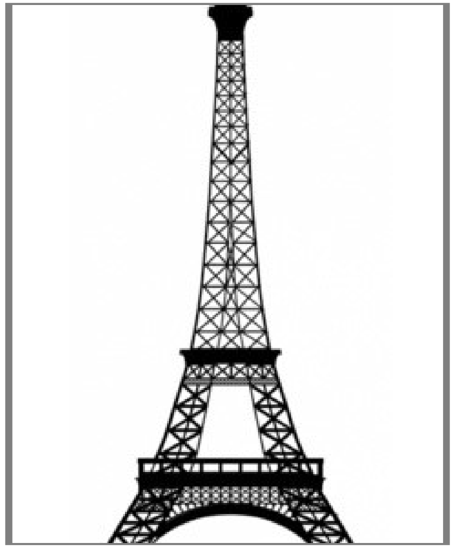




Slide from Jianbo Shi

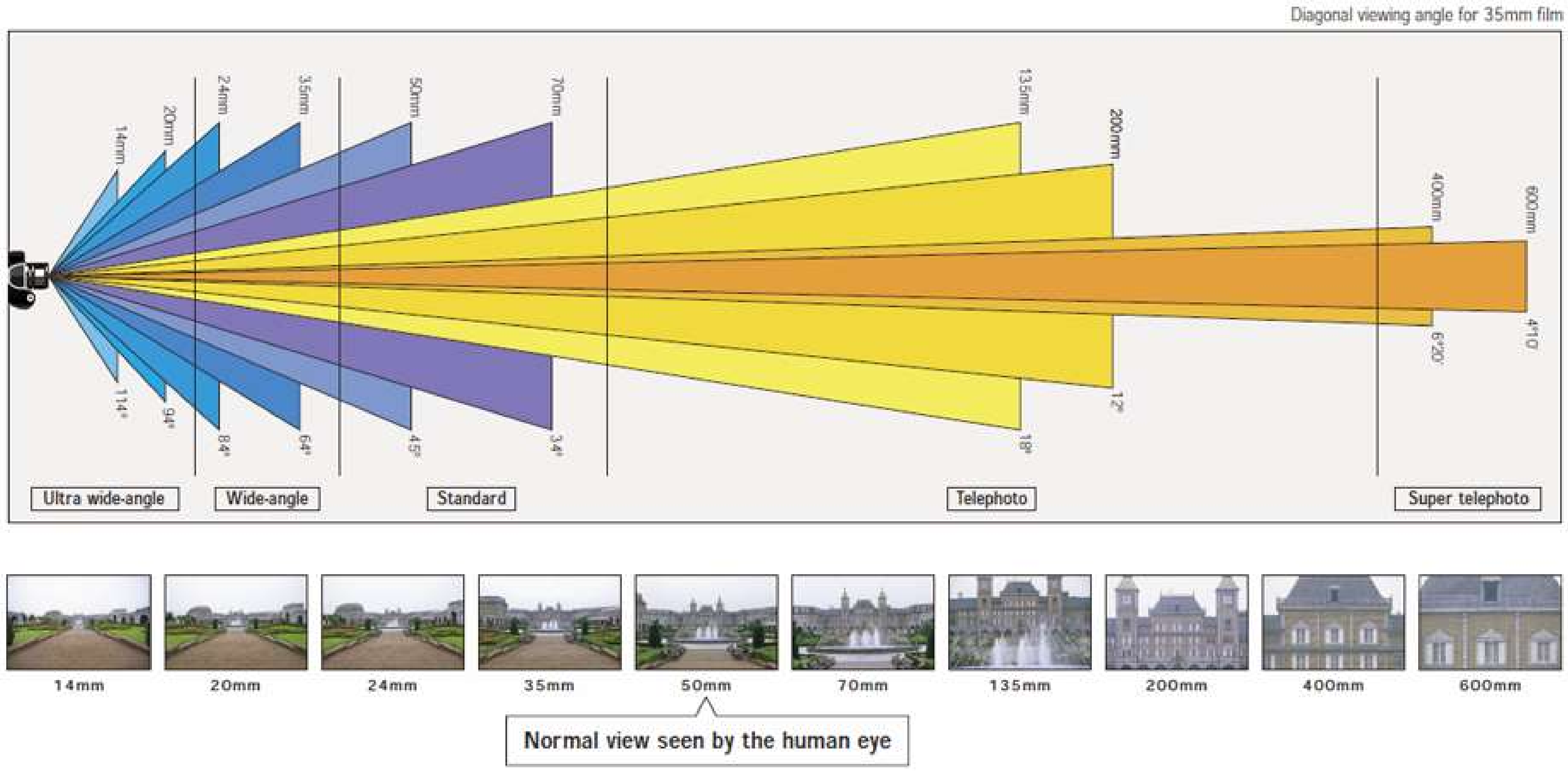
Focal length





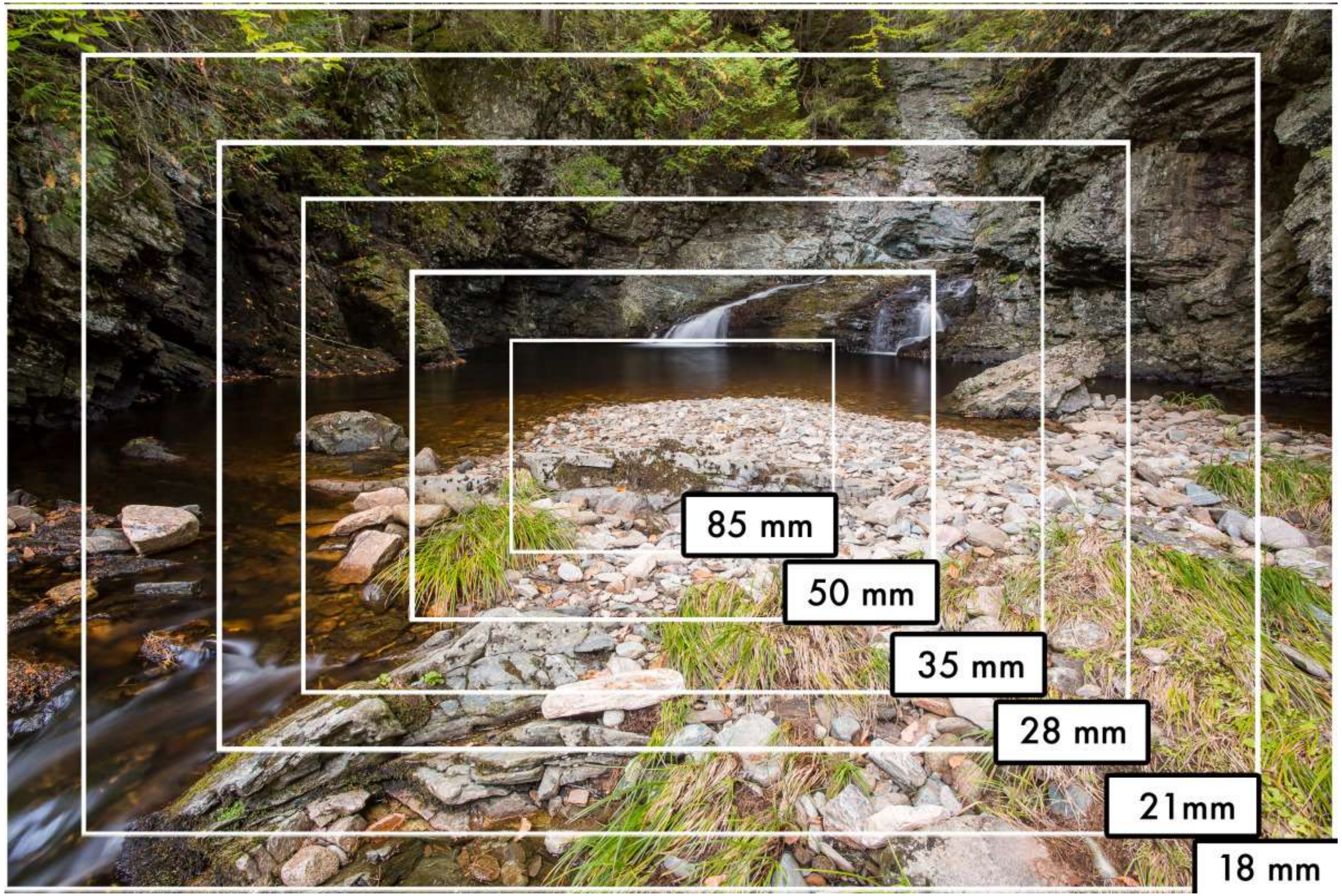
Slide from Jianbo Shi

Focal length



Slide from Jianbo Shi

Focal length

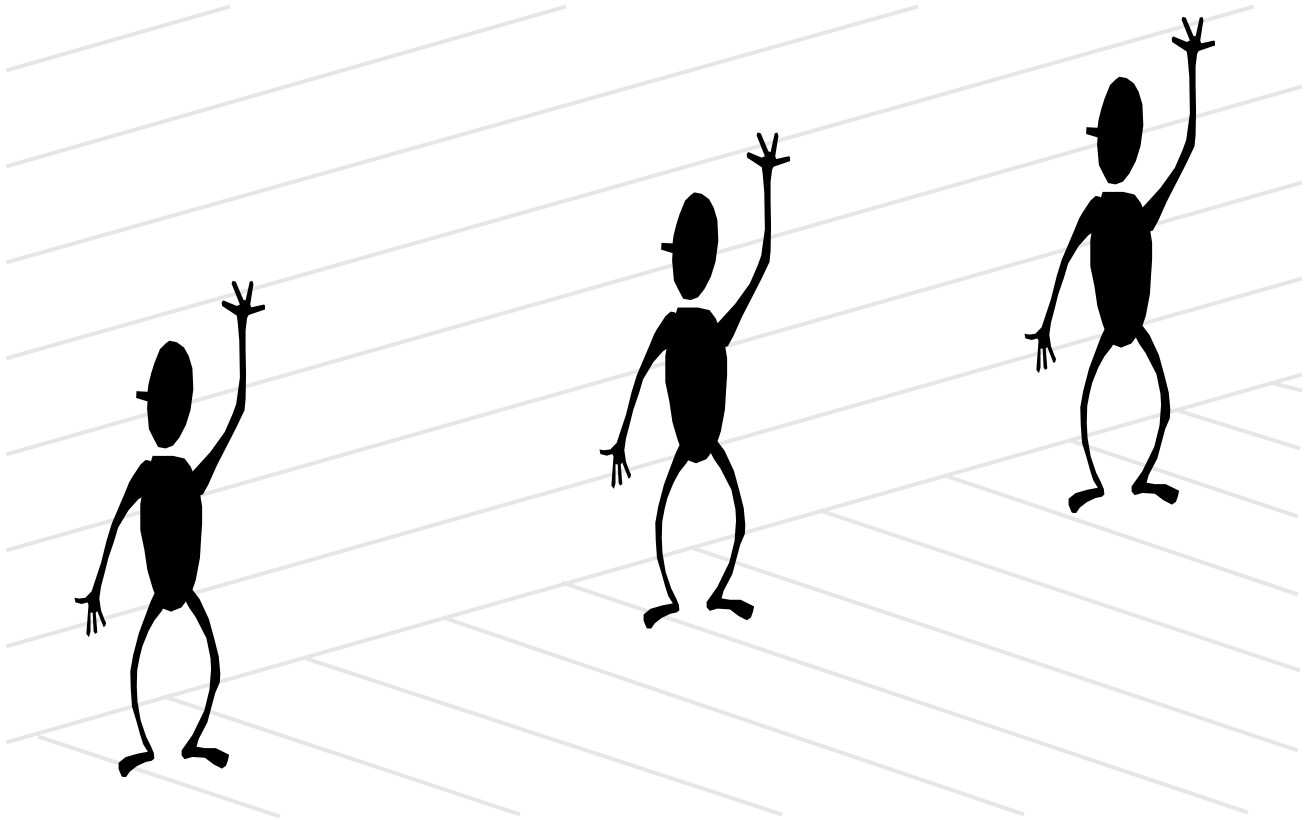


Slide from Jianbo Shi

Dolly zoom

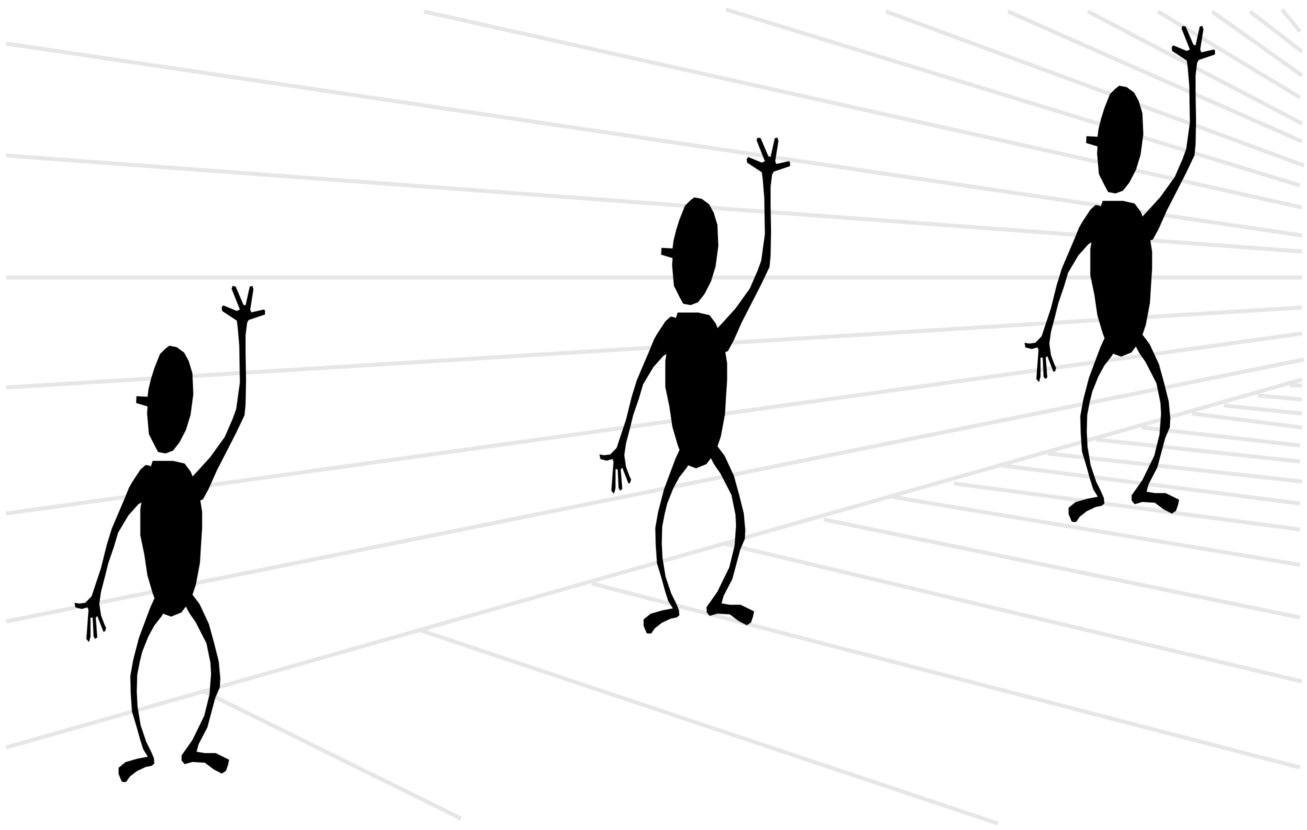
- https://www.youtube.com/watch?v=NB4bikrNzMk

Perspective cues



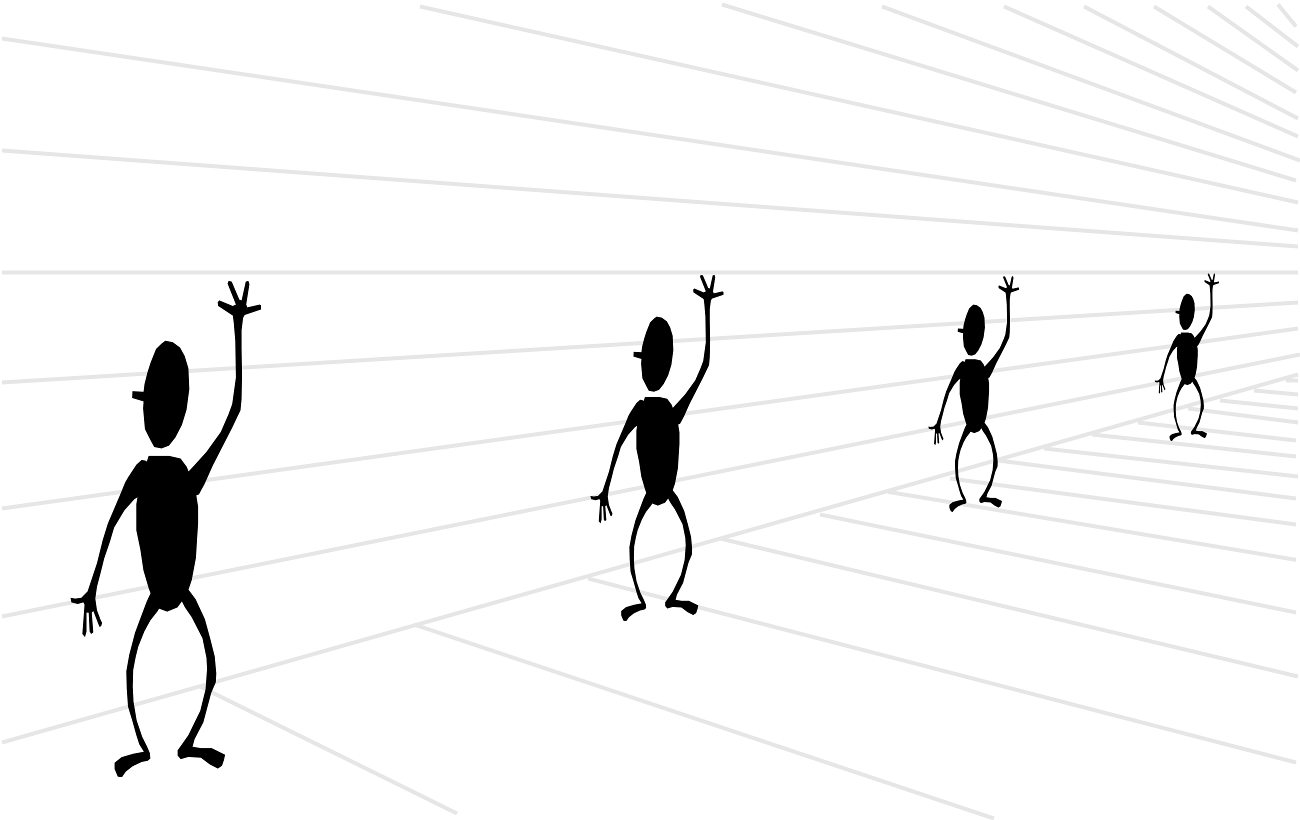
Slide from Steve Seitz

Perspective cues



Slide from Steve Seitz

Perspective cues



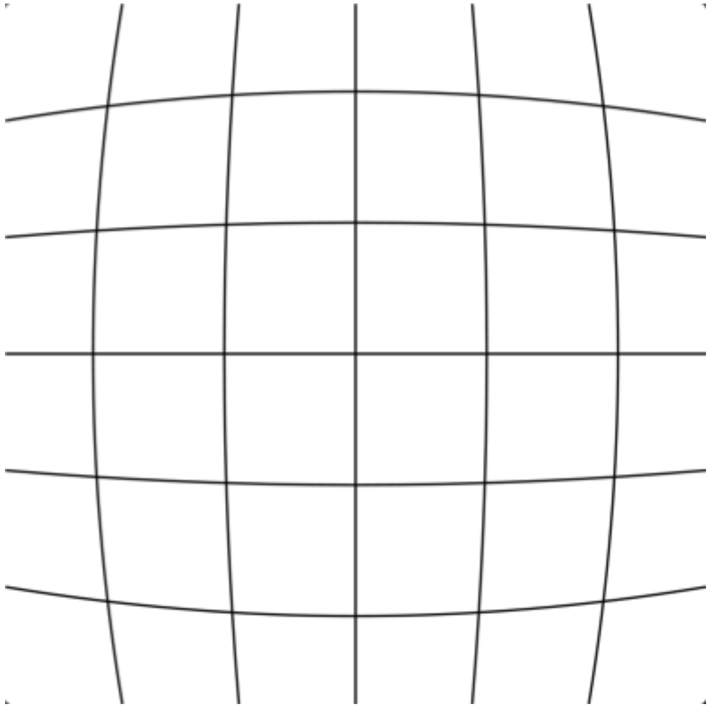
Slide from Steve Seitz

Lens distortion (Fisheye lens)

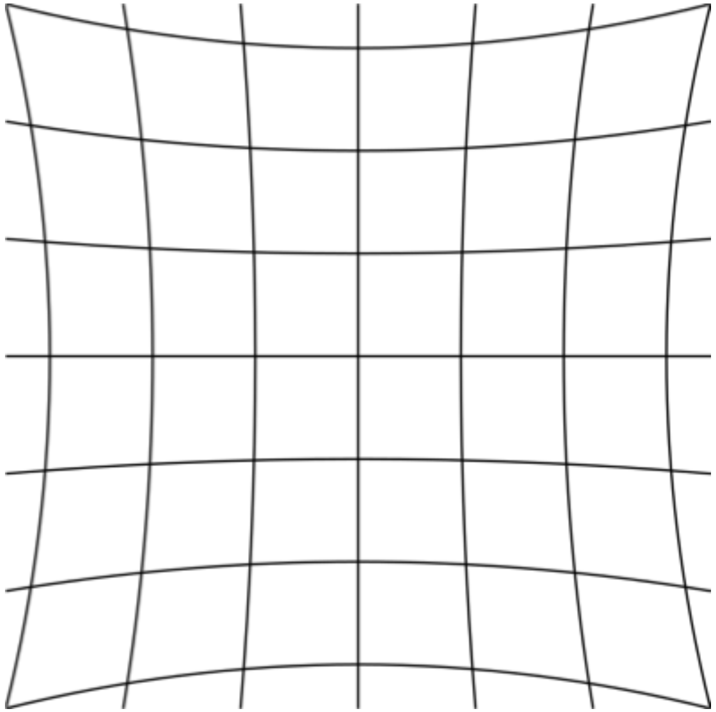
Multiple models to capture distortion, commonly used is Plumb Bob model

Slide from Jianbo Shi

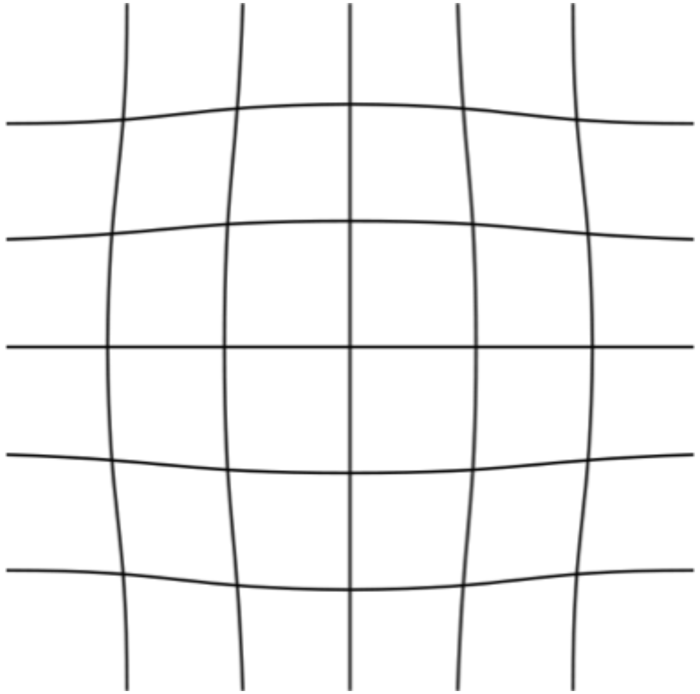
Lens distortion



Barrel distortion



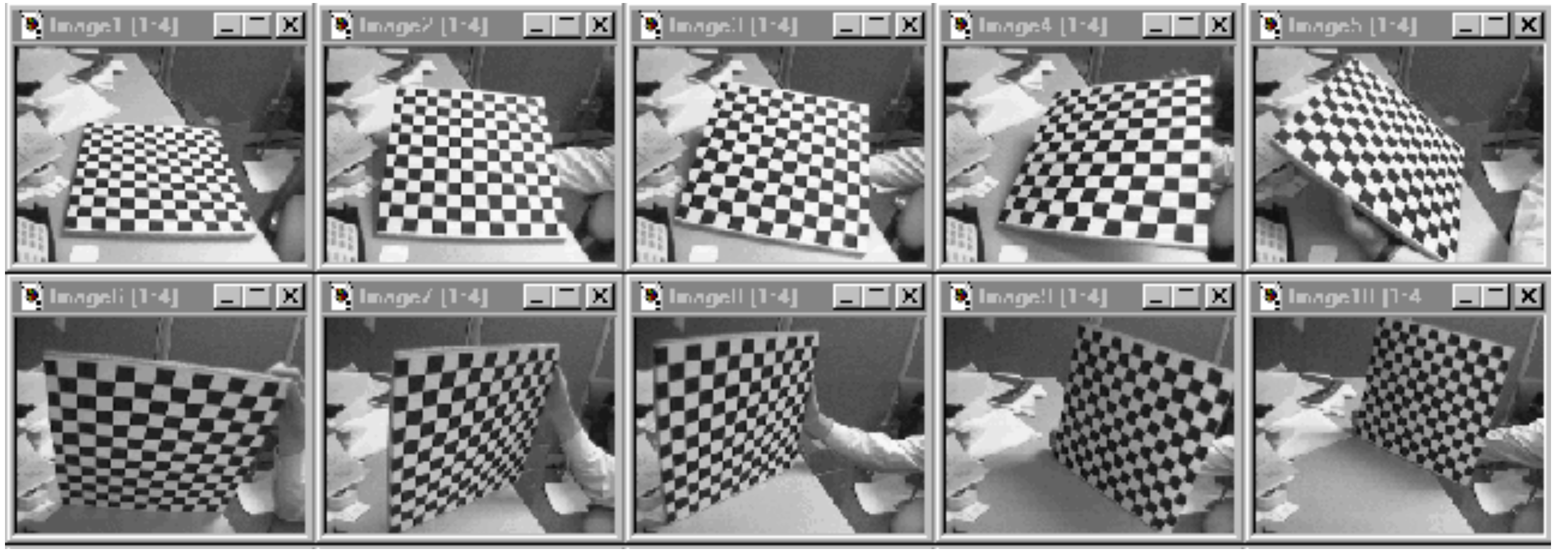
Pincushion distortion



Moustache distortion

Modeled as a function that changes pixel (u,v) after intrinsics + extrinsics based projection

Camera Calibration



Images courtesy Jean-Yves Bouguet, Intel Corp.

- Compute camera intrinsic parameters & distortion
- Compute extrinsics between multiple views/different cameras
- **Key idea:** Use a known object of fixed size & match it across multiple scenes -> provides enough constraints to solve for camera parameters
- Good code available online!
  - Intel's OpenCV library: http://www.intel.com/research/mrl/research/opencv/
  - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang's web site: http://research.microsoft.com/~zhang/Calib/

Slide from Steve Seitz