

---

# **EE6221 Robotics and Intelligent Sensors (Part 3)**

## **Lecture 2: Vision Sensors and Systems**

**Guoqiang Hu**  
School of EEE, NTU

# Outline

---

- Introduction
- Application scenarios
- Cameras
- Pinhole camera model
- Camera calibration
- For reference reading:
  - Color space representations
  - Useful image processing methods

# Application Scenarios of Camera Sensors

---

- Vision-based control
- Vision-based estimation
- Vision-based navigation
- .....

# Academic Examples



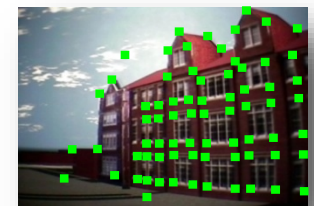
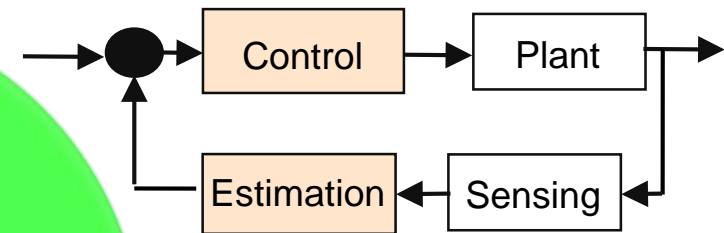
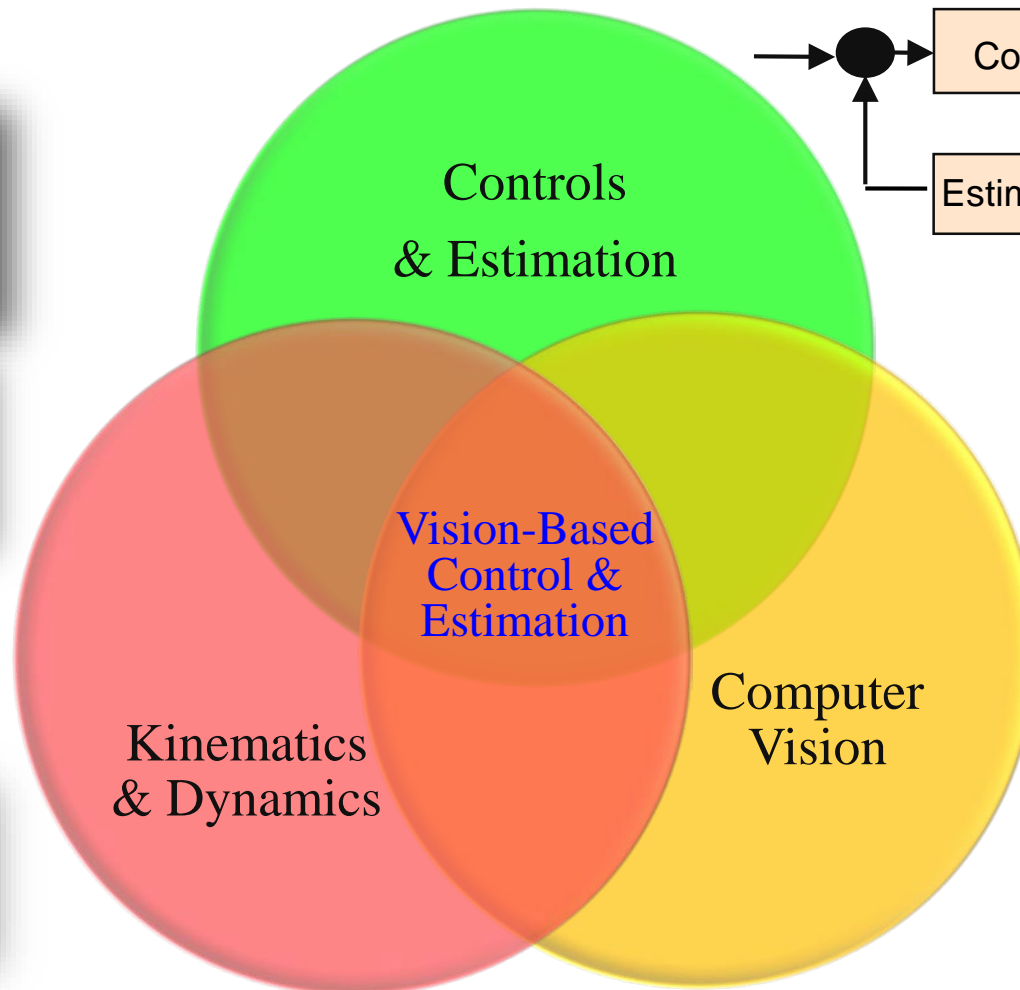
Vision-guided tracking examples



Vision-guided robot navigation

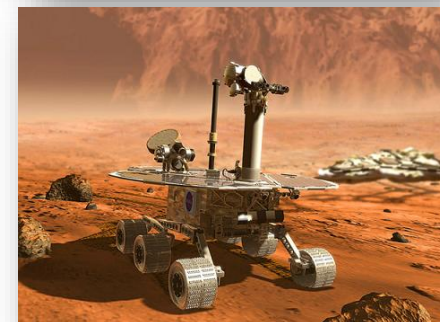
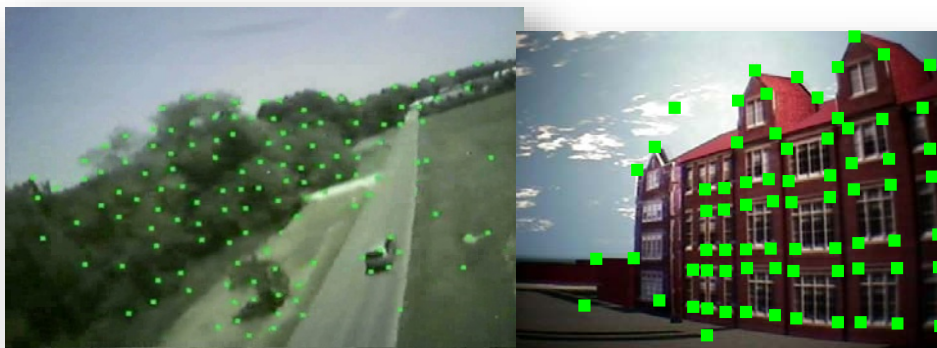
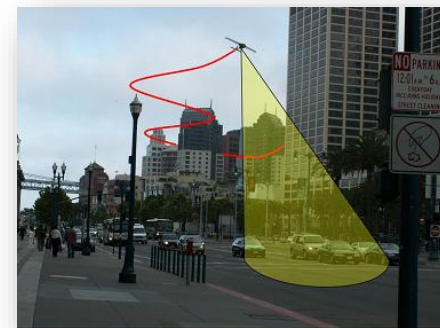
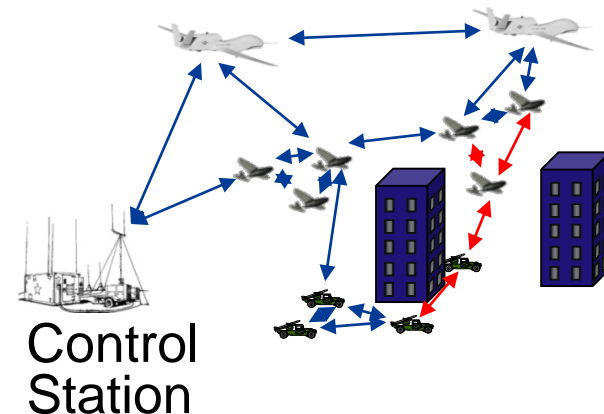
# Vision-Based Control & Estimation

- **Vision-based control and estimation:** use of image data in feedback control and state estimation of moving agents (e.g., robot manipulators, mobile robots, or unmanned vehicles, etc).



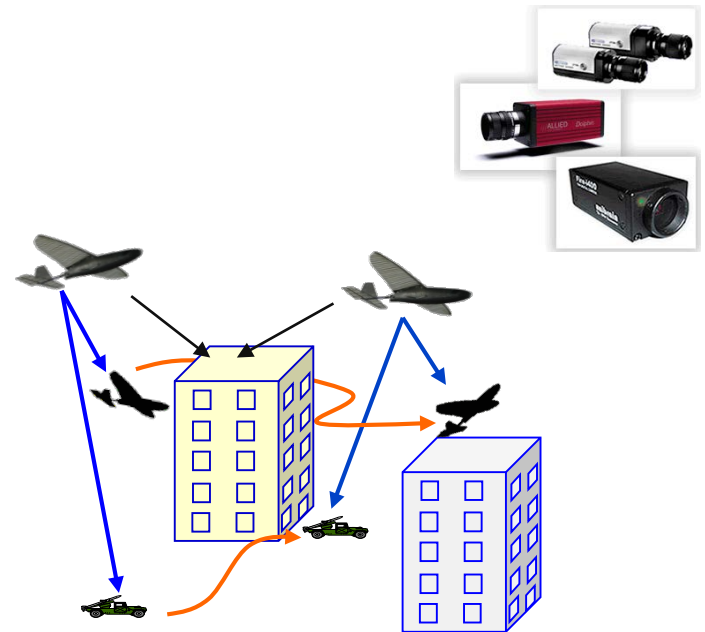
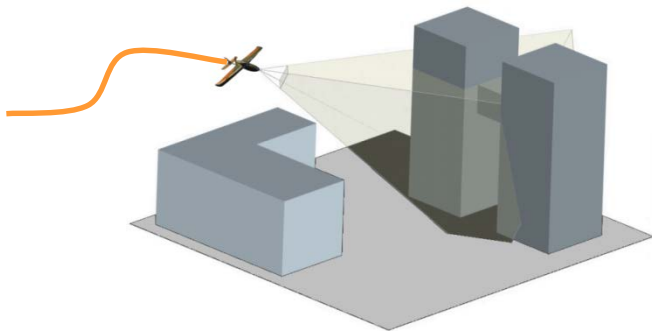
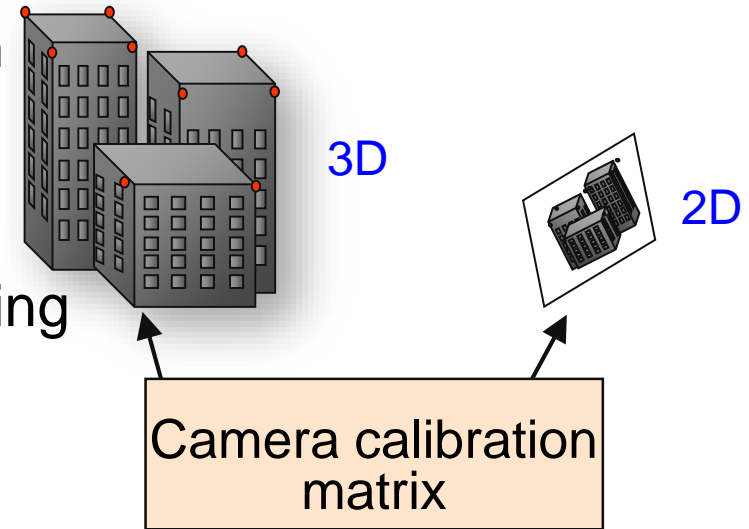
# Motivation – Vision-based Control

- **Why study vision-based control (also called visual servo control)?**
  - Position & orientation is typically required in navigation and control of autonomous vehicles
  - GPS may not be available, IMU has error drift
- **Advantages of vision:**
  - Vision is a rich data set, a lot of potential information
  - Vision is a passive sensor, can't be detected like sonar, radar, laser, etc.
  - Vision is intuitive to humans
  - Cameras are relatively cheap and versatile



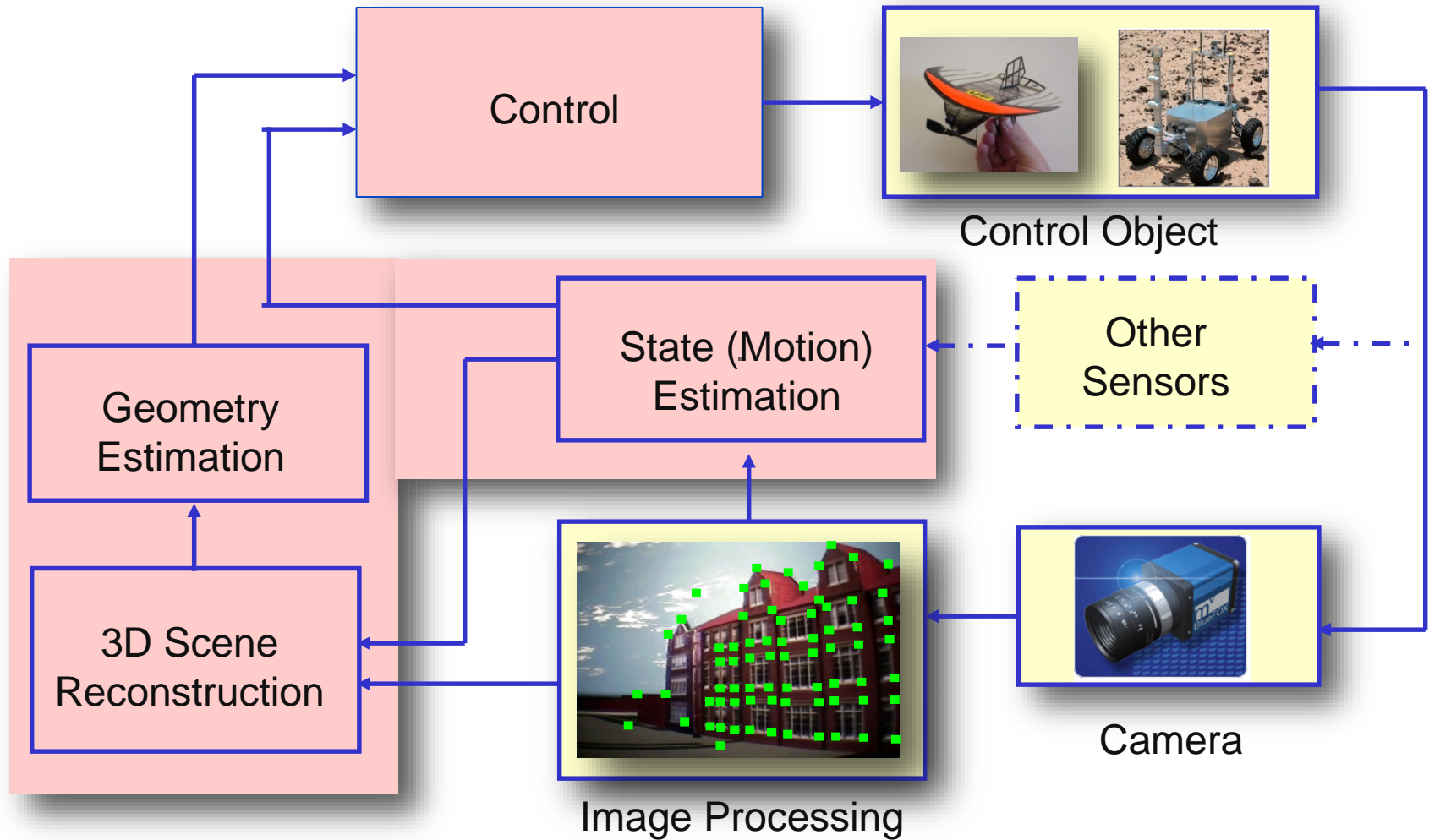
# Challenges – Vision-based Control

- Robust and real-time vision estimation
- Feedback through a transformation: from 3D to 2D
- Loss of depth information during imaging projection
- Camera calibration uncertainty
- Limited field of view
- Relative motion between targets
- Nonlinear, multivariable differential equations with unmeasurable states





# Vision-Based Control & Estimation

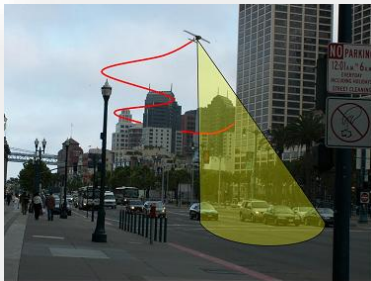
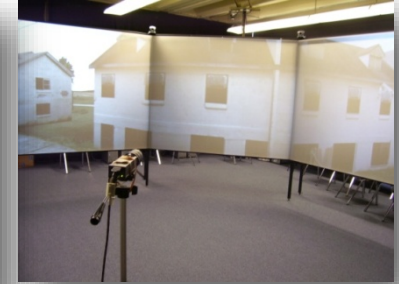




# Applications

## Systems:

- Autonomous ground vehicles
- Smart cars
- Unmanned air vehicles
- Mobile sensor networks
- Nano/micro manipulation
- Mobile manipulators
- .....



# Imaging-Introduction

---

- Cameras are composed of several key components
  - The lens collects and focuses light
  - Imaging surface measures intensity/frequency of light
- We will discuss **simple models for imaging** that are accurate especially when using quality cameras
- We will discuss **camera calibration**, which is essential for image based pose and structure estimation, and can increase the accuracy of simple models, even for low quality cameras
- We will discuss common, simple **image processing routines** that may prove useful

# Imaging-Introduction

---

## Reference Books:

- Computer Vision: A Modern Approach, by Forsyth and Ponce
- Computer and Robot Vision, by Haralick and Shapiro
- An Invitation to 3-D Vision by Ma, Soatto, Kosecka and Sastry

# Imaging-Introduction

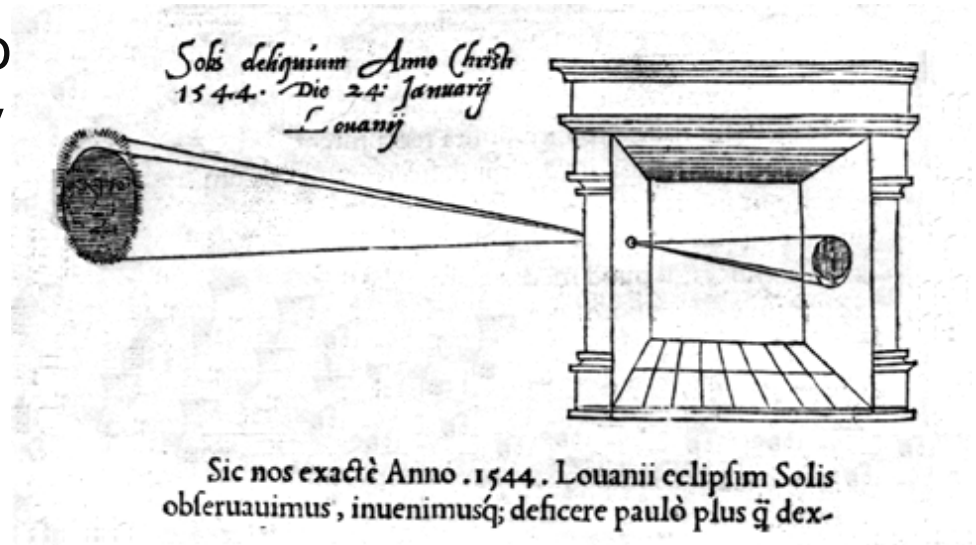
---

## Free software resources for image processing:

- Intel Open Source Computer Vision Library for C++ (OpenCV)
  - <http://www.intel.com/technology/computing/opencv/>
- Machine Vision Toolbox for Matlab
  - <http://www.petercorke.com/Machine%20Vision%20Toolbox.html>
- Camera Calibration Toolbox for Matlab
  - [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)
- Image Processing Toolbox for Matlab
  - Included in most full versions of Matlab
- GNU Image Manipulation Program (Image Processing)
  - <http://www.gimp.org/>
- Virtual Dub (Video Processing/Editing)
  - <http://www.virtualdub.org/index>

# Camera History

- The **Pinhole Camera Model** was discussed by Aristitotle and Euclid in 3<sup>rd</sup> Century BC
- Ibn al-Haitham is credited with building the first pinhole camera (camera obscura) in 10<sup>th</sup> Century AD
- Giovanni Battista della Porta first added a lens to focus light in 15<sup>th</sup> Century
- Boyle and Hooke made portable models in 17<sup>th</sup> Century
- First photograph was by Niépce in 1826
- First color photo by Maxwell in 1861



Reinerus Gemma-Frisius, 1544

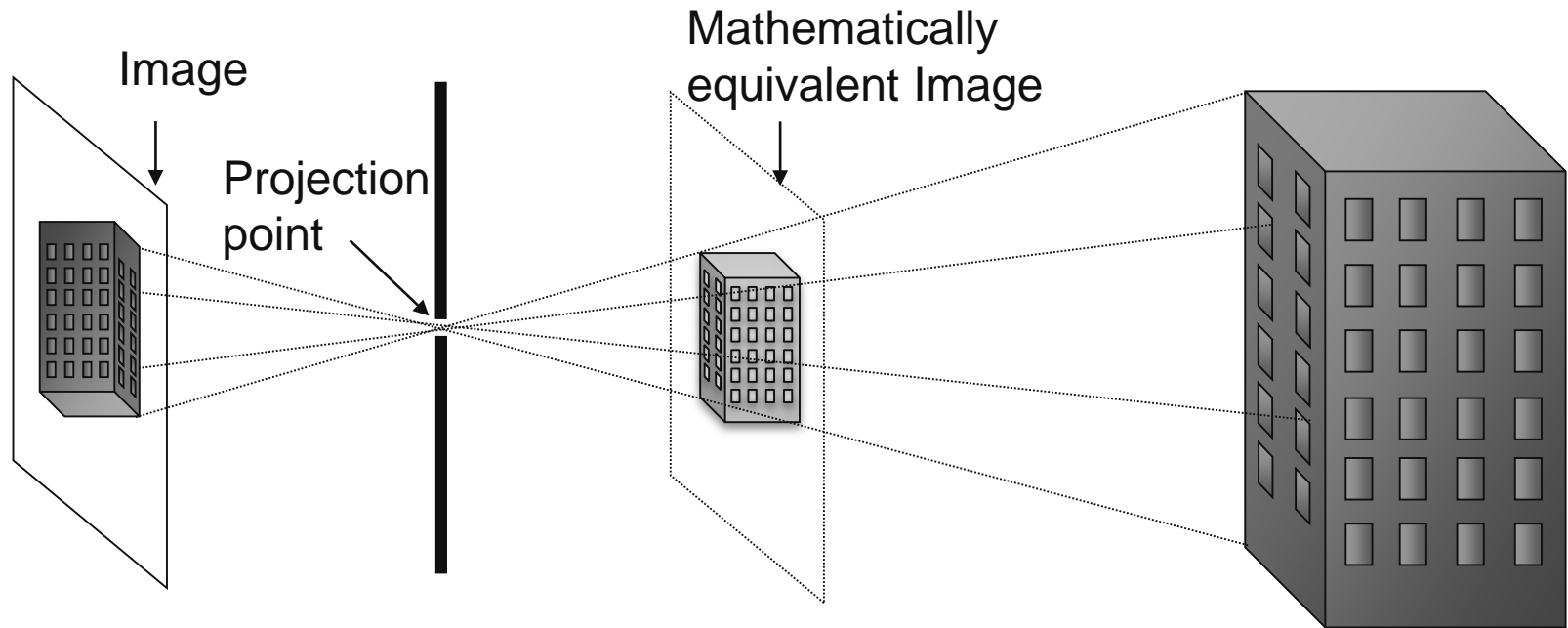
# Camera History

- First **flexible film** developed by Eastman Kodak 1885
- First electronic camera (precursor to TV cameras) developed by Farnsworth in 1927
- Video tape recorders introduced in 1951 by Bing Crosby Enterprises
- First digital camera built by Sasson w/ Eastman Kodak in 1973
- First commercial **digital camera** by Eastman Kodak in 1991 (\$13K!)



Kodak DCS-100, 1991

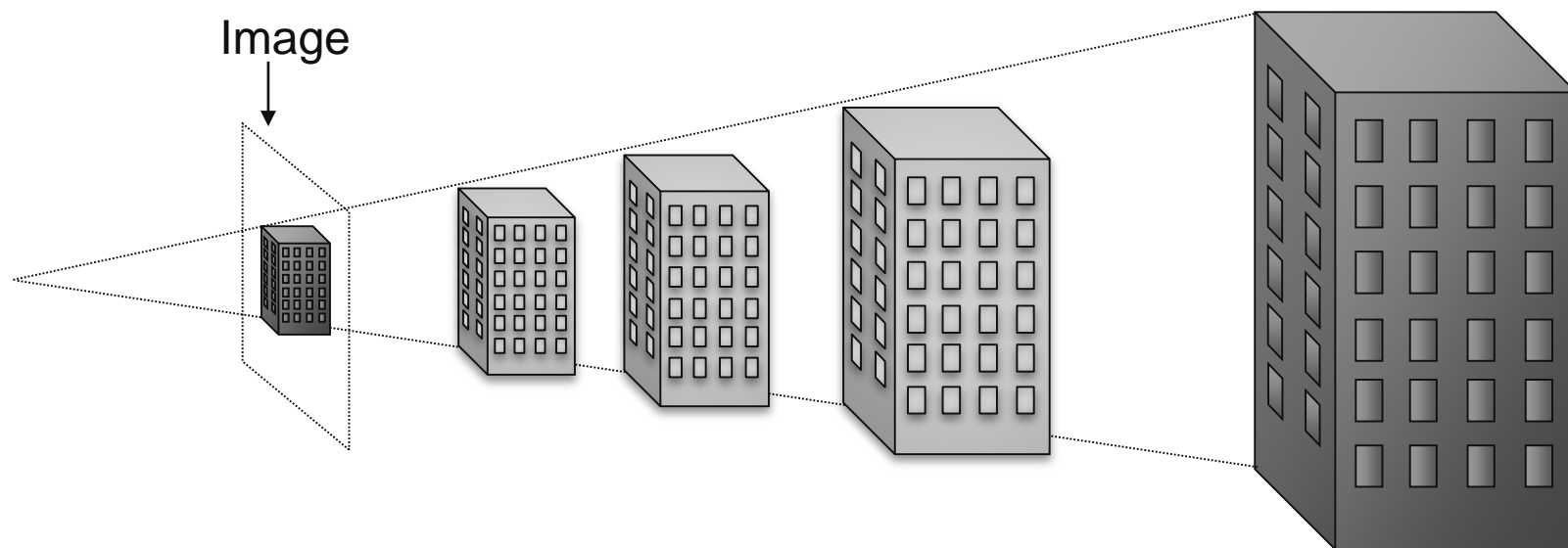
# Pinhole Projection Model



- Rays from every point in the scene intersect at the projection point and continue through to the imaging surface
- Results in an inverted image
- Mathematically, we can consider an imaging surface in front of the projection point, which is not inverted

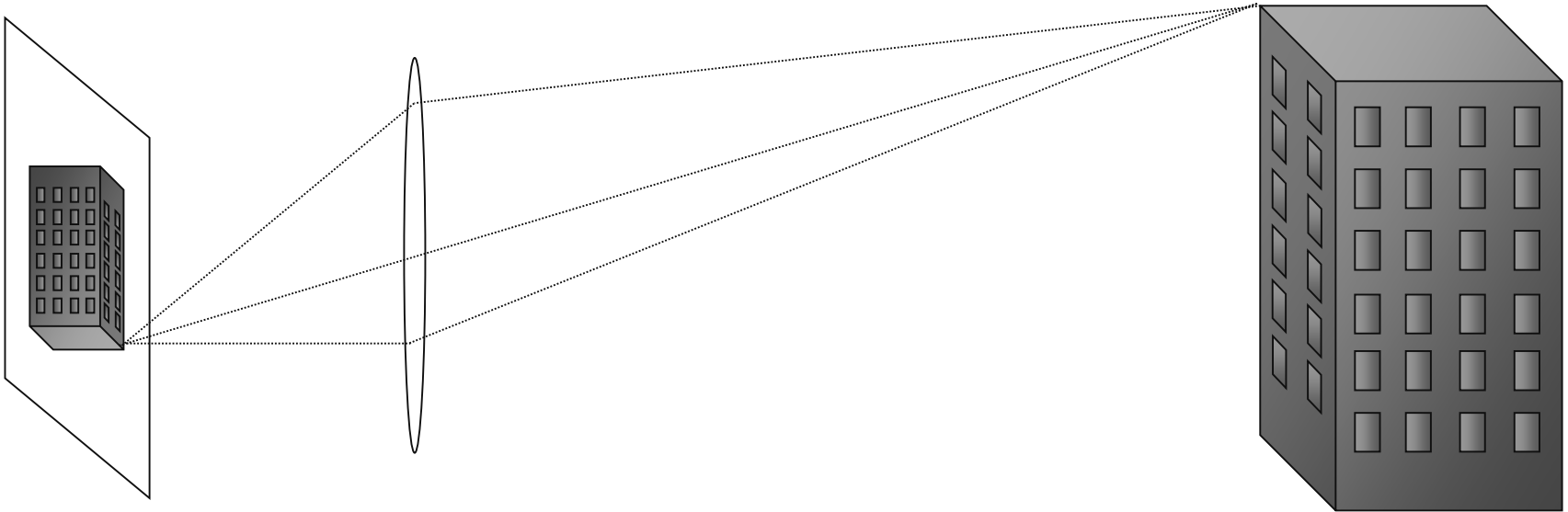


# Pinhole Projection Model



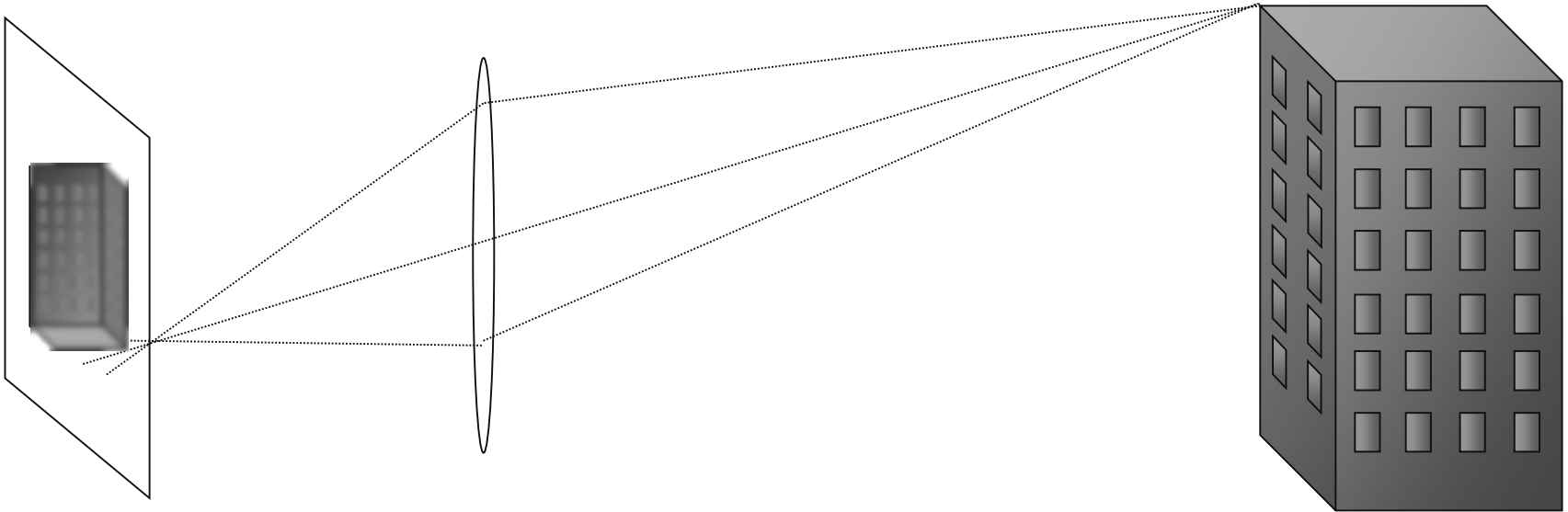
- All point along the rays intersect the image plane at the same point
- This results in a loss of size and depth information
- A priori knowledge or additional sensors/measurements can determine scale
- Fundamental drawback to imaging as a sensor

# Perfect Lens Projection Model



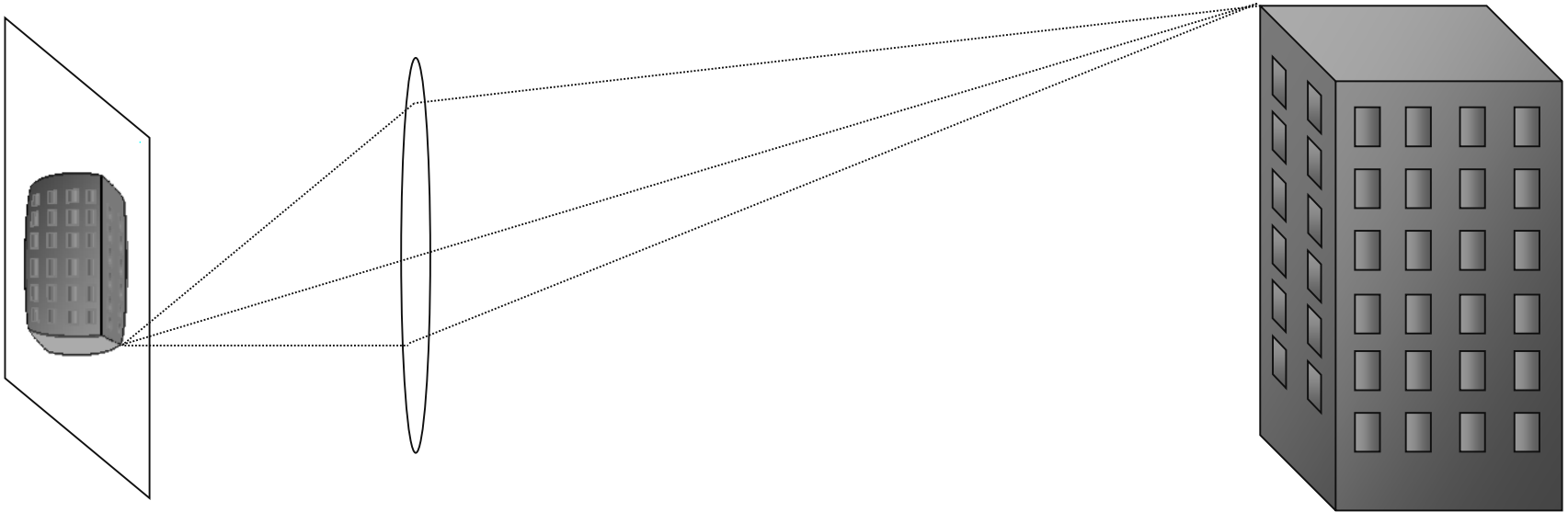
- A lens collects light by focusing many rays from a 3D point to same point on image plane
- A perfect lens is identical to the pinhole camera model at a specific focal length

# Imperfect Lens Projection



- If the distance between the lens and imaging surface is not correct, rays from a 3D point will not be focused to a single point on the image
- This results in a blurred image
- Images can be sharpened, but not of much use in estimation and control

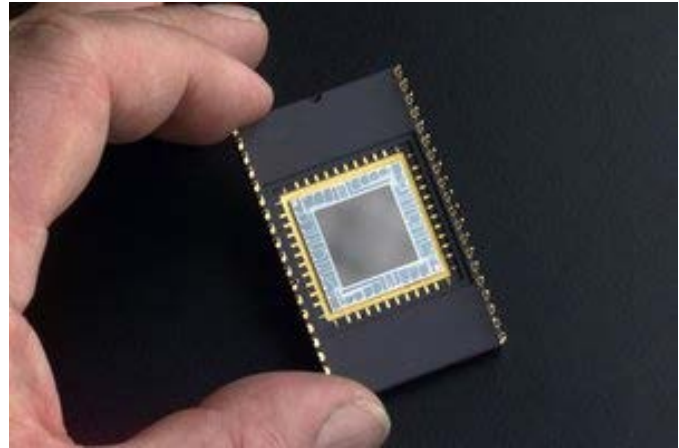
# Imperfect Lens Projection



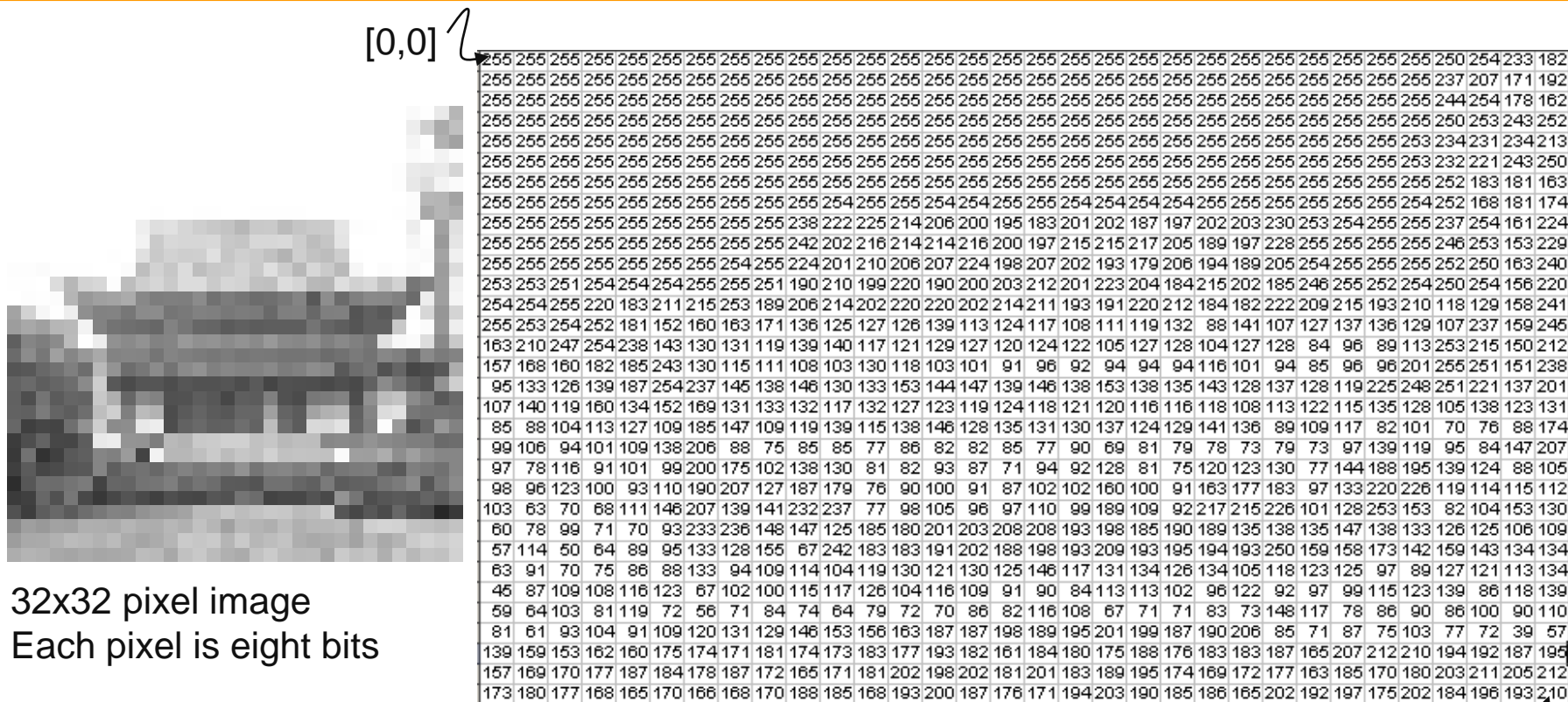
- Real lens will have imperfections which lead to misfocus (blur) or distortion
- Radial distortion occurs when 3D lines are not projected to 2D lines
- Calibration can learn distortion parameters and remove them from an image – approximating pinhole camera model

# Digital Imaging

- In digital cameras, the imaging surface is an array of light sensitive elements (e.g. CCD's)
- The amount of incident light on each element is recorded as a number
- The image is discretized in two ways
  - The image is broken into discrete pixels with coordinates  $[u,v]$
  - The value of each pixel takes discrete values in some range
- Video is further discretized temporally
- This results in some amount of quantization noise



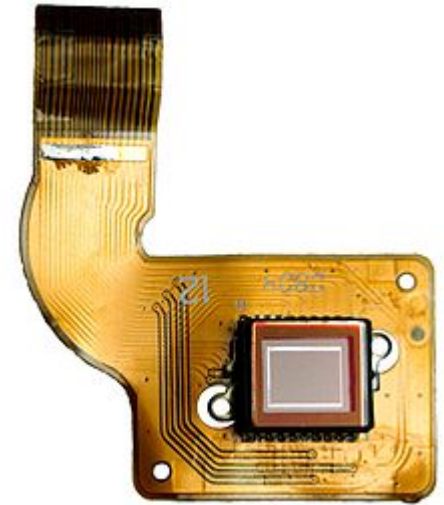
# Digital Imaging



- For an eight-bit image, pixels take integer values in range  $0 \dots 255$  [31,31]
- Typically pixel coordinates are counted from top left to bottom right, though some cameras count from bottom left

# Digital Imaging

- “Charge coupled device” (CCD) and “complementary metal oxide semiconductor” (CMOS) are typical light sensing technologies
- Performance and cost of the two are converging, but typically:
  - CCD more prevalent
  - CCD delivers less noisy image than CMOS
  - Light sensitivity for CCD better than CMOS
  - CMOS requires less power than CCD
  - CMOS cheaper to manufacture than CCD



A CCD image sensor on a flexible circuit board



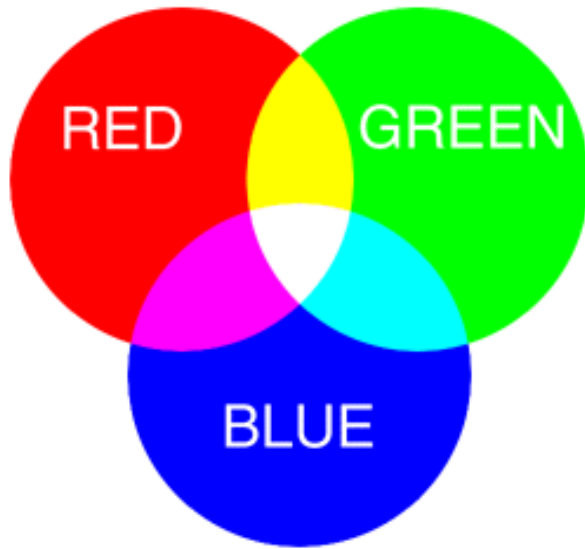
# Digital Imaging

- Digital Cameras using Firewire or USB2.0 are capable of transmitting rates of 50-100Mb/s (1280x1024 1 byte per pixel @ 30-60 Hz)
- Digital frame grabbers (internal PCI Cards) are capable of 400 Mbs
- PCI Express capable of 4 Gb/s!
- Analog video signals (such as RCA or BNC) require an analog frame grabber or external A/D converter (such as RCA-USB) and operate at 30Hz.



# Digital Imaging

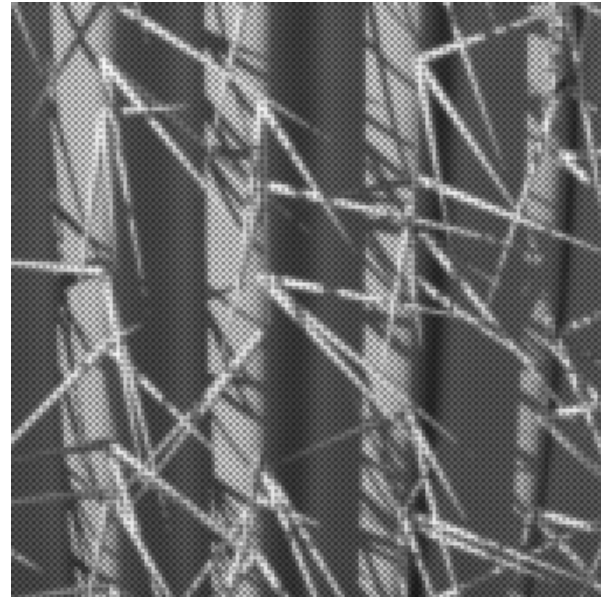
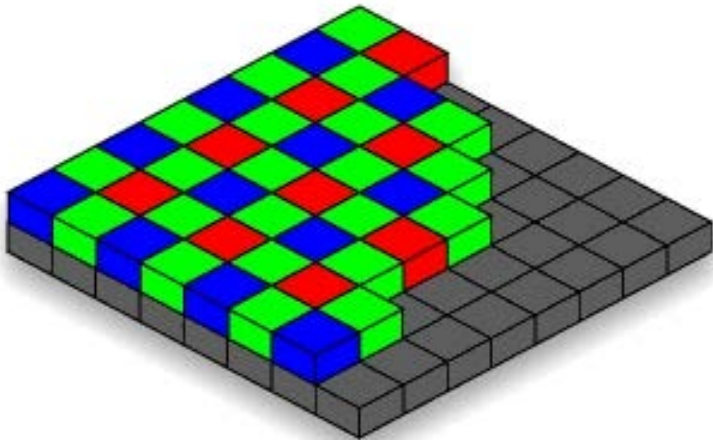
- In grayscale images, pixel has a single measurement of light intensity from pitch black to pure white
- In color images, each pixel has three measurements associated with it, typically red, green and blue (RGB), which indicate the intensity of light at that frequency



- RGB are primary color of light and combine to form other colors
- The human eye is more sensitive to green light, so color cameras generally are more sensitive to green as well.
- RGB to grayscale conversion is not simply the norm of RGB values

# Digital Imaging

- Note that most color cameras do not have separate receptors for each color
- A pattern of color filters, known as “Bayer Pattern” are placed over a single surface
- Resulting image is filtered by camera software/firmware to provide RGB image, but high speed camera may deliver raw Bayer image



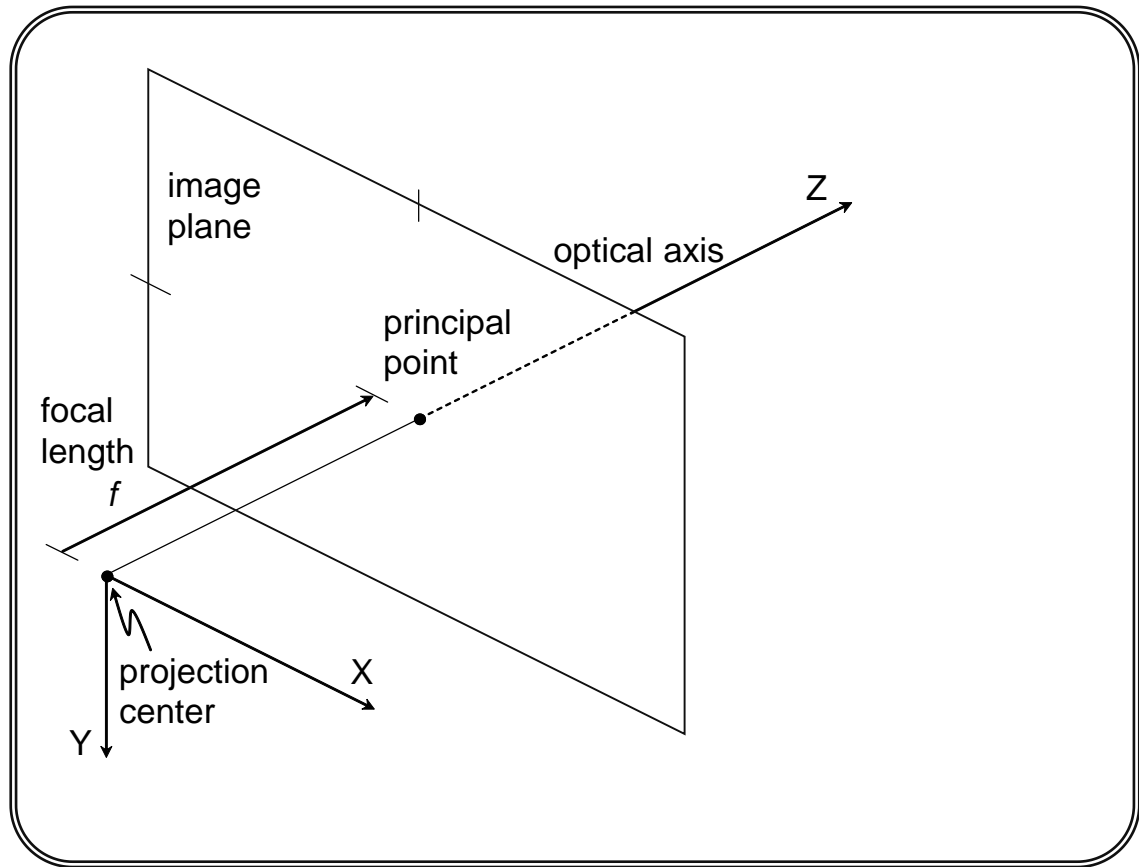
# Pinhole Camera Model

---

- The pinhole camera model is a first order approximation of camera optics
- Widely used and essential for structure from motion algorithms
- Mathematics are simple and accurate for most quality cameras
- If higher order optical effects are pervasive, proper camera calibration can often make the first order approximation valid

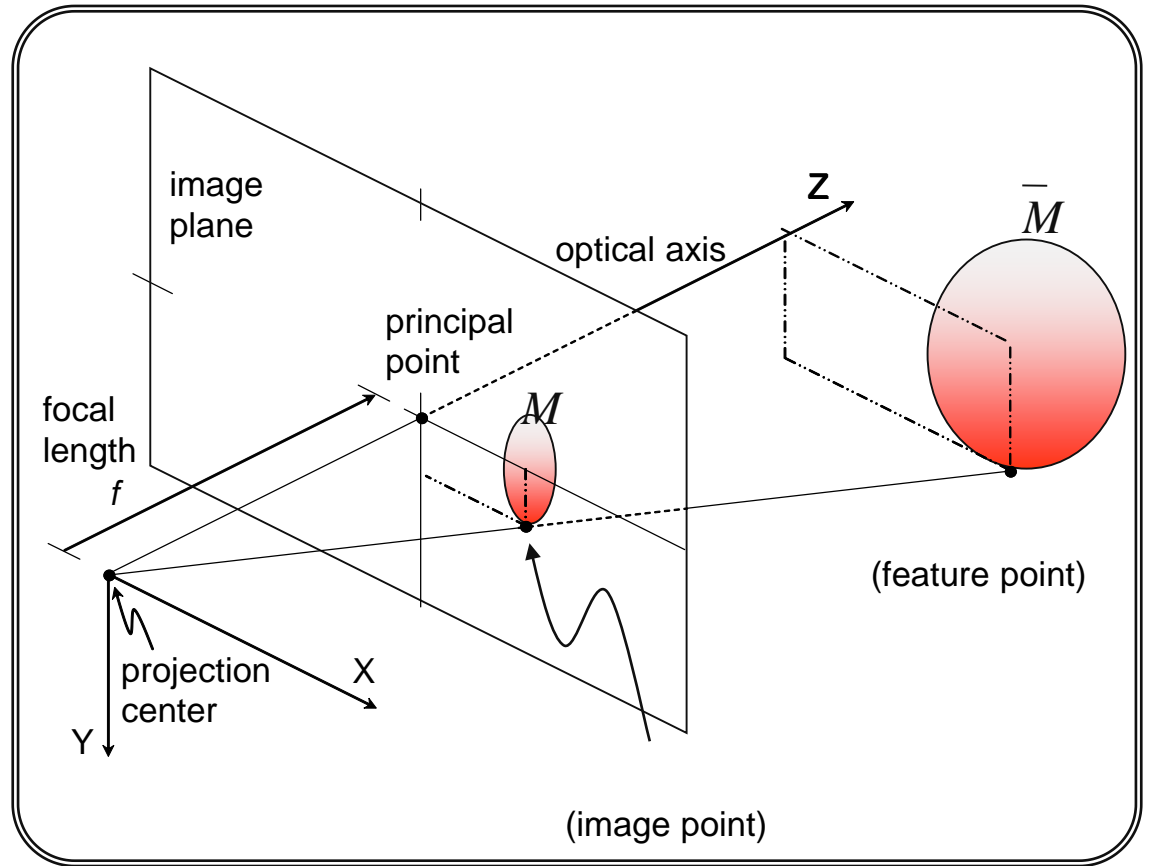
# Pinhole Camera Model

- Reference frame  $\mathcal{F}$  is attached to the camera.
- The origin of  $\mathcal{F}$  is the “projection center”
- The imaging surface is a plane orthogonal to the  $z$  axis and intersects the  $z$  axis at a distance  $f$
- Point of intersection is the “principal point”
- $f$  is the “focal length” of the camera, for simplicity assume  $f=1$



# Pinhole Camera Model

- A feature point has 3D coordinates  $\bar{m} = [x, y, z]^T$  in the camera frame
- It projects to a point in the image plane with coordinates
- The projection of a point can be denoted by function  $m = \pi(\bar{m}) = \frac{1}{z} \bar{m}$
- A curve or surface given as a set of 3D points  $\bar{M}$  projects to  $M = \pi(\bar{M})$



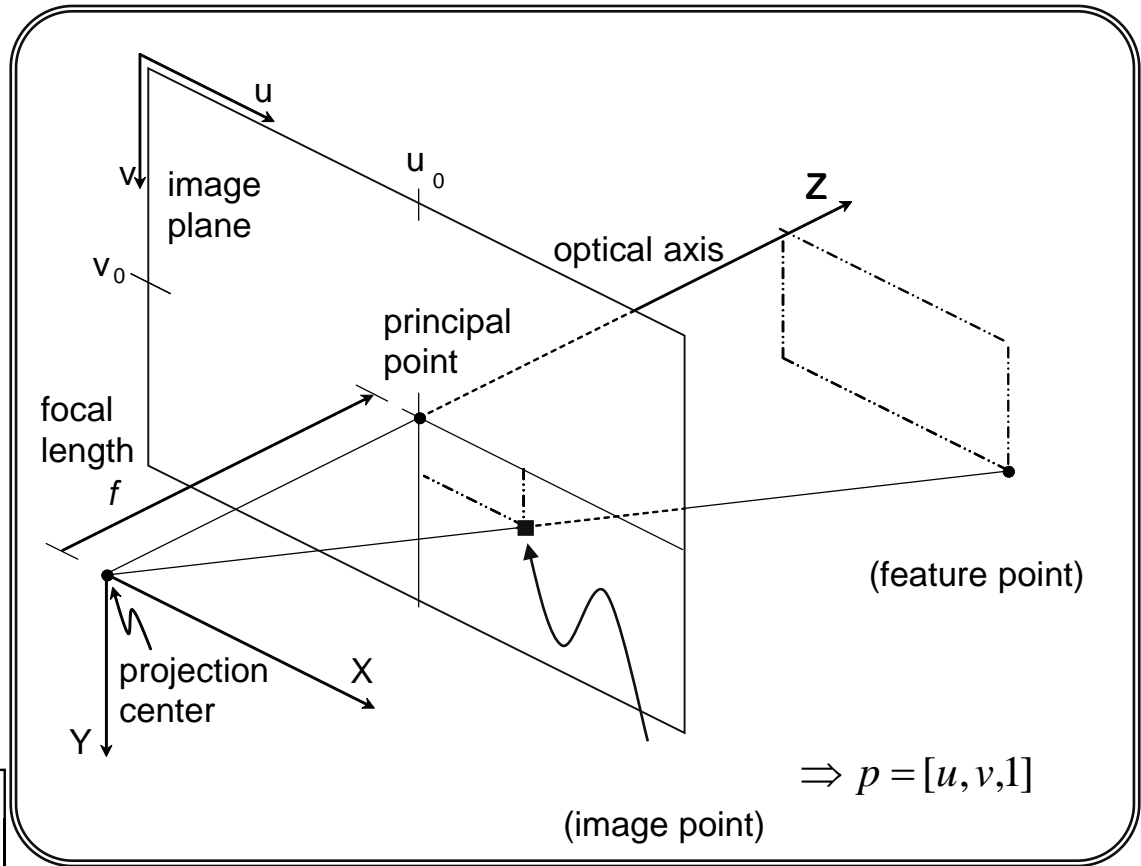
# Pinhole Camera Model

- When dealing with a digital camera, we must account for the number of pixels, shape of pixels, and size of pixel elements

- Image point  $m$  is discretized to pixel coordinates  $p = [u, v, 1]^T$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\sigma_x & -f\sigma_x \tan \alpha & u_0 \\ 0 & f\sigma_y \sec \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix}$$

$$p = Am$$



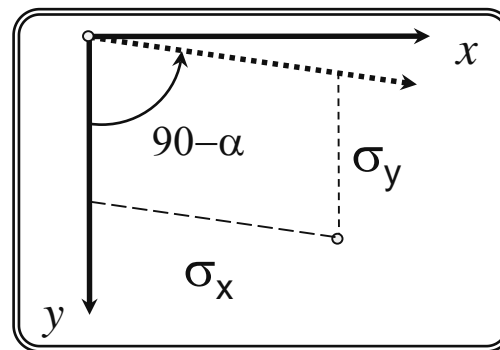
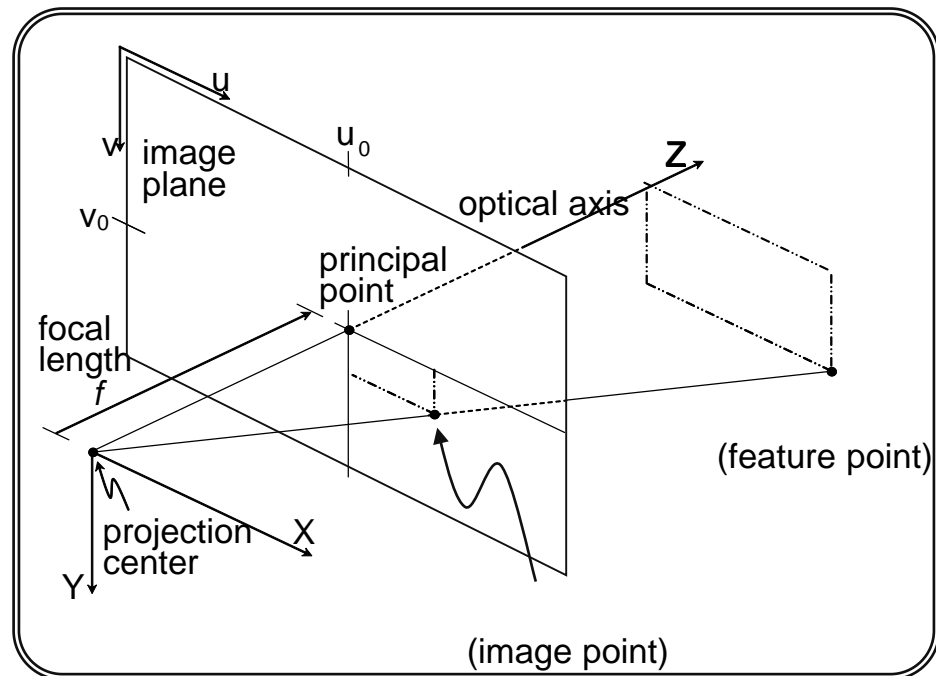


# Pinhole Camera Model

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\sigma_x & -f\sigma_x \tan \alpha & u_0 \\ 0 & f\sigma_y \sec \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{x}{z} \\ \frac{y}{z} \\ 1 \end{bmatrix}$$

$$p = Am$$

- $A$  is intrinsic calibration matrix
- $f$  is camera focal length
- $\sigma_x, \sigma_y$  are size of pixels in focal lengths
- $[u_0, v_0]$  is the principal point in pixels
- $\alpha$  is skew angle, usually  $\approx 0$



# Pinhole Camera Model (Summary)

- 3D feature point with **Euclidean coordinates** in camera frame  $\mathcal{F}$

$$\bar{m} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

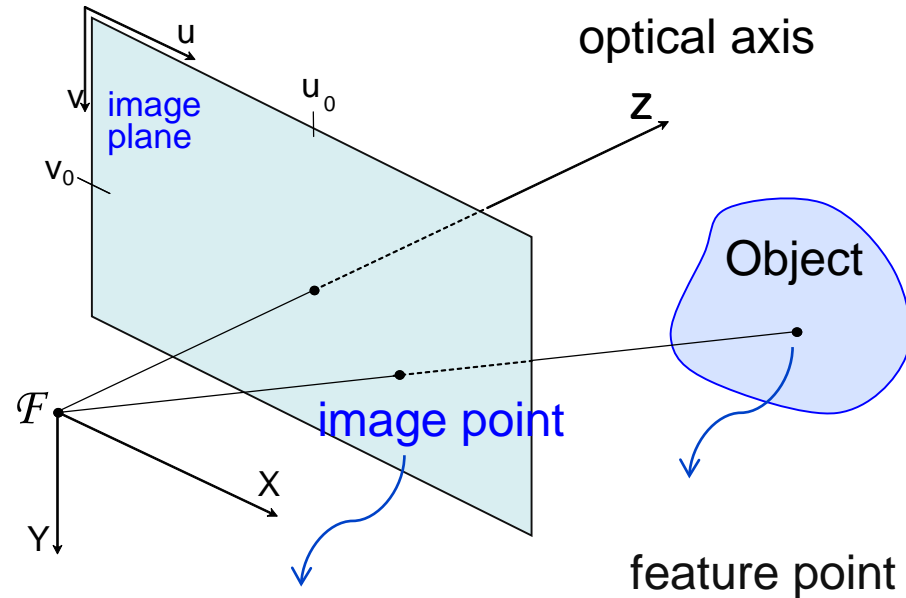
- Projected to image plane with **normalized coordinates** in  $\mathcal{F}$

$$m = \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix} = \frac{\bar{m}}{z} = \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix}$$

- Mapped to **pixel coordinates** by Calibration Matrix

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = Am = \begin{bmatrix} f\sigma_x & -f\sigma_x \tan \alpha & u_0 \\ 0 & f\sigma_y \sec \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_x \\ m_y \\ 1 \end{bmatrix}$$

**A:** camera calibration matrix



# Pinhole Camera Model (Remarks)

---

- Many estimation and control schemes require normalized coordinates for points
- Given pixel coordinates  $p$  of a point in the image and knowledge of calibration matrix  $A$ , recover normalized coordinates


$$m = A^{-1} p$$


- There is no way to recover the Euclidean coordinates without additional information. Depth ambiguity is a consequence of imaging.


# Camera Calibration

- Consider an inertial, world reference frame  $\mathcal{F}_w$
- The camera frame  $\mathcal{F}_c$  has translation  $T$  and rotation  $R$  w.r.t.  $\mathcal{F}_w$ .
- A point has coordinates  $m_w = [x_w, y_w, z_w]^T$  in the world frame.
- Pixel coordinates of the point in the image are given by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\sigma_x & -f\sigma_x \tan \alpha & u_0 \\ 0 & f\sigma_y \sec \alpha & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/z & 0 & 0 & 0 \\ 0 & 1/z & 0 & 0 \\ 0 & 0 & 1/z & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

  
 Intrinsic  
calibration matrix

  
 Projection  
matrix

  
 Extrinsic  
calibration matrix

- Calibration is the processing of recovering these parameters

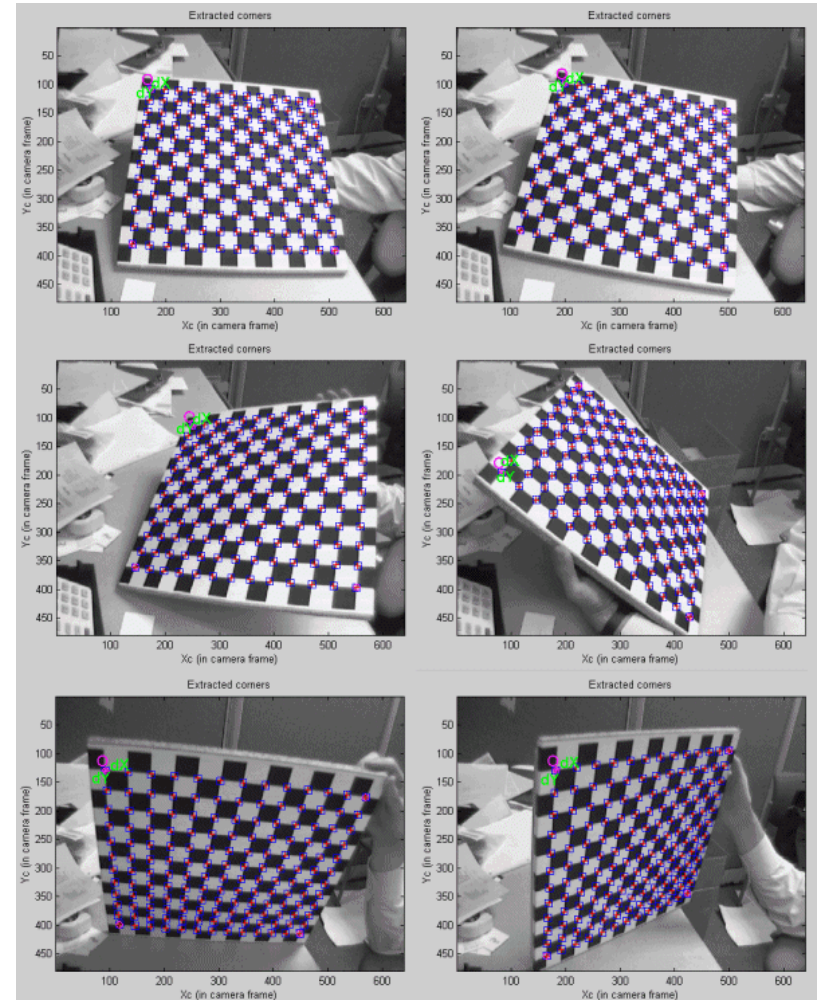
# Camera Calibration

---

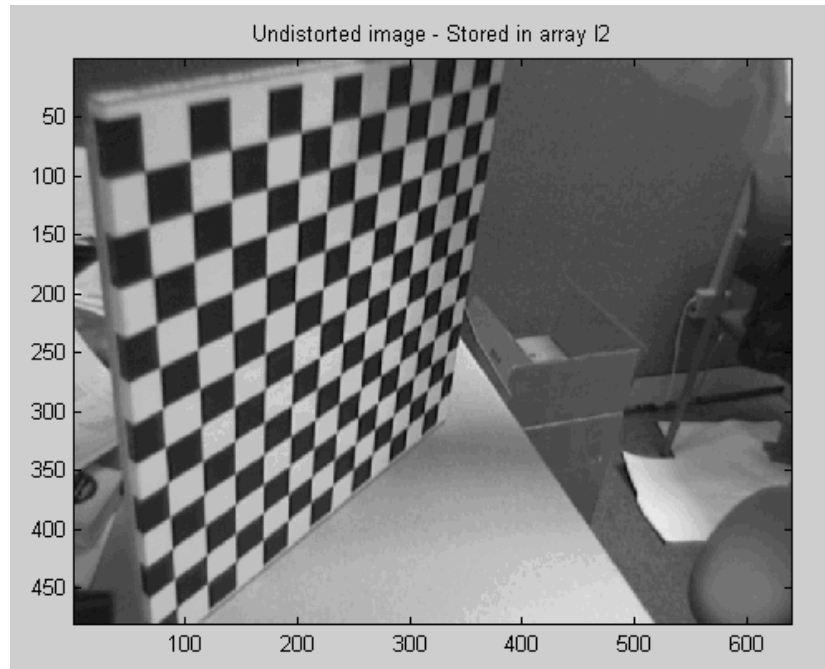
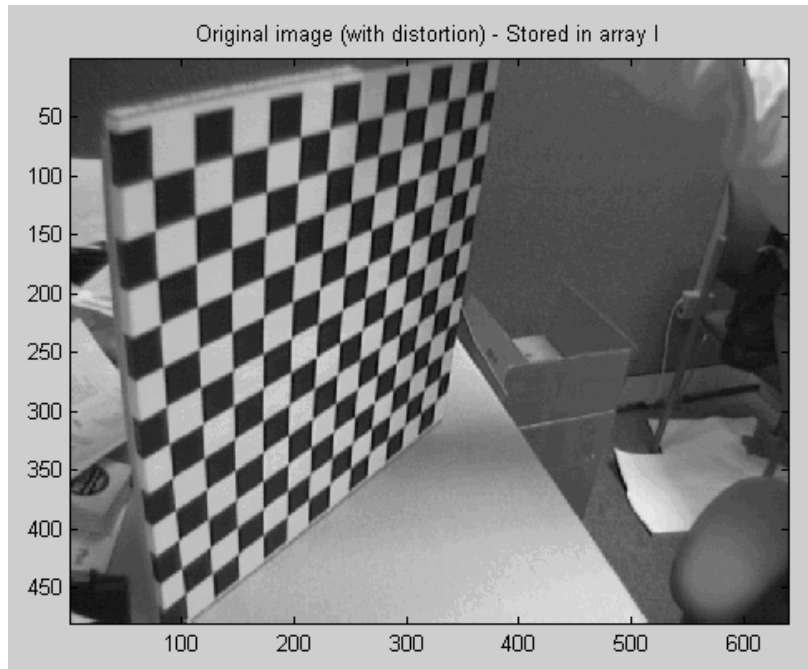
- There are many calibration techniques, and many free resources
- Intel Open Source Computer Vision Library
  - <http://www.intel.com/technology/computing/opencv/>
- Camera Calibration Toolbox for Matlab
  - [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)
- The DLR Camera Calibration Toolbox
  - <http://www.dlr.de/rm-neu/desktopdefault.aspx/tabid-3925/>

# Camera Calibration

- Most calibration methods require some sort of known target such as a calibration board
- Numerous images of target are taken
- Features, such as corners, are extracted
- Comparison of known target and image points allows solving of intrinsic and extrinsic camera parameters
- Intrinsic calibration parameters are usually solved simply as the elements of the calibration matrix  $A$



# Camera Calibration



- In addition to intrinsic calibration matrix, most calibration methods also give nonlinear distortion parameters which can undistort images

$$m_d = \left(1 + k_1 r^2 + k_2 r^4 + k_5 r^6\right) \begin{bmatrix} m_x \\ m_y \end{bmatrix} + \begin{bmatrix} 2k_3 m_x m_y + k_4 (r^2 + 2m_x^2) \\ 2k_4 m_x m_y + k_3 (r^2 + 2m_y^2) \end{bmatrix}$$



# Exercise (not mandatory)

---

**Try one of the calibration example by yourself using matlab**

There are many calibration techniques, and many free resources  
Intel Open Source Computer Vision Library

<http://www.intel.com/technology/computing/opencv/>

Camera Calibration Toolbox for Matlab

[http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)

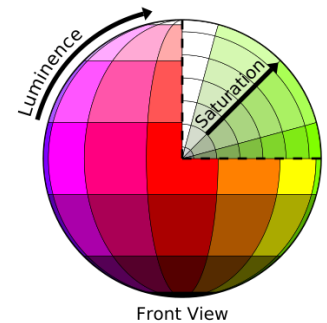
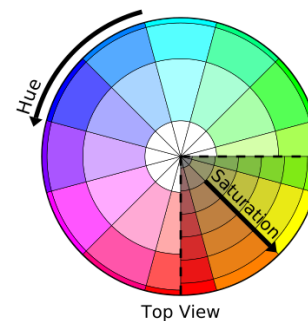
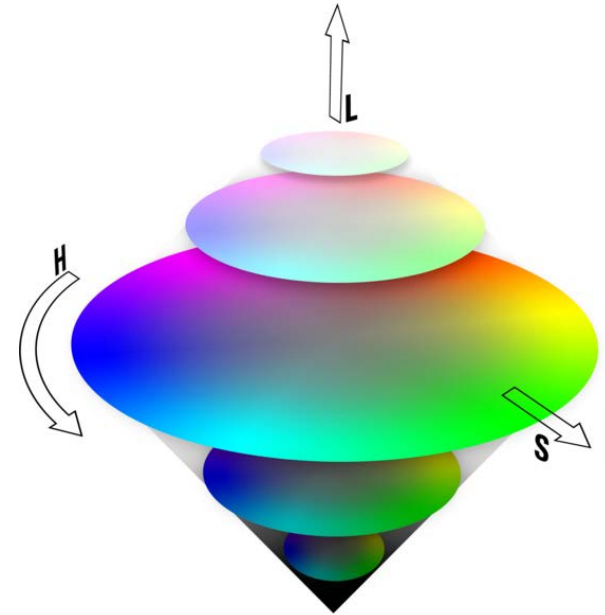
The DLR Camera Calibration Toolbox

<http://www.dlr.de/rm-neu/desktopdefault.aspx/tabid-3925/>

- 
- The following slides are for reference reading if interested

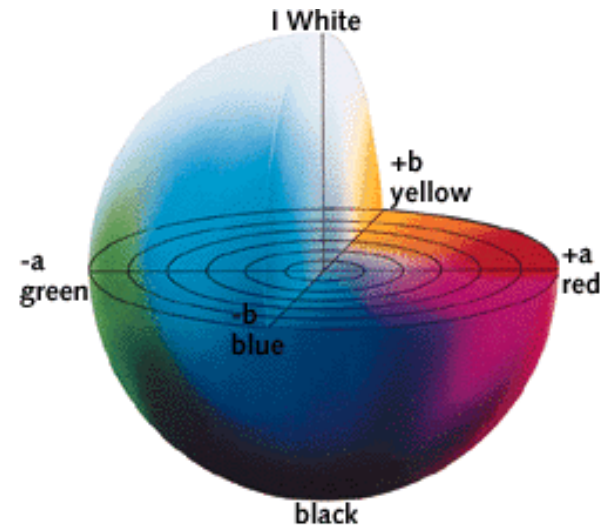
# Useful Imaging Methods - Color Spaces

- Several useful alternatives to RGB color space
- Hue Saturation Luminescence (HSL) color space
  - Color mapped to Hue value which is cyclical
  - Light intensity (white to black) maps to Luminescence
  - Color intensity (grey to bright) maps to Saturation
- Particularly useful for finding objects of similar colors



# Useful Imaging Methods - Color spaces

- CIELAB color space
  - Light intensity (white to black) maps to Luminescence
  - A and B denote color and intensity
- A “linear” color space
- Useful when “distance” between colors is needed to differentiate objects



# Useful Imaging Methods - Image Processing

- In filtering a small 2D “window” or “kernel” of values is convolved with each pixel of the image
- For each pixel in the image, take the sum of products of multiplying the kernel by the pixel values

$$\frac{1}{21} * \begin{bmatrix} 1 & 3 & 1 \\ 3 & 5 & 3 \\ 1 & 3 & 1 \end{bmatrix}$$

## 3x3 Gaussian Kernel

[illegible]

# What is Image Filtering?

Modify the pixels in an image based on some function of a local neighborhood of the pixels

10	5	3
4	5	1
1	1	7

Some function



	7	

# Linear Filtering

- Linear case is simplest and most useful
  - Replace each pixel with a linear combination of its neighbors.
- The prescription for the linear combination is called the convolution kernel.

10	5	3
4	5	1
1	1	7

 $\otimes$ 

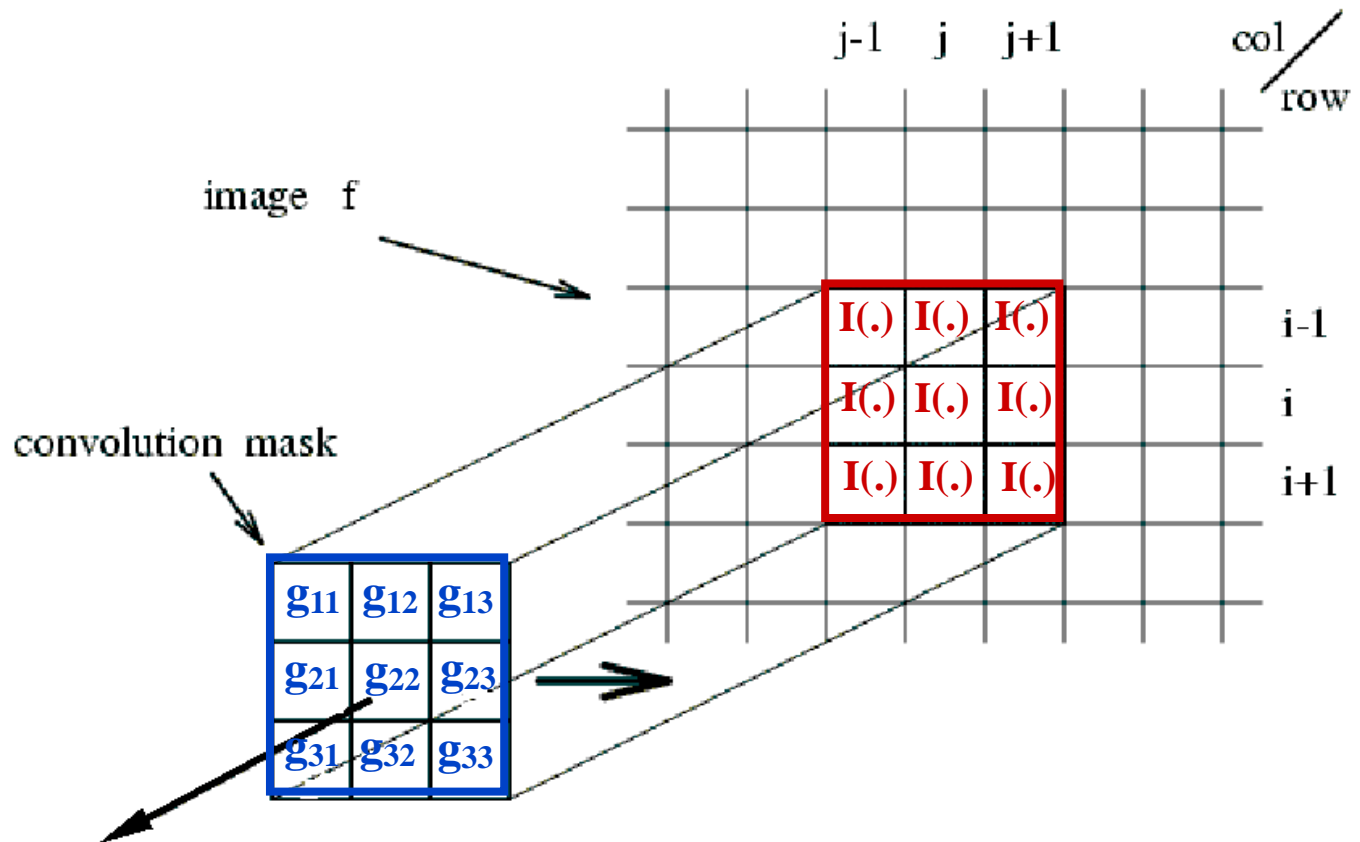
0	0	0
0	0.5	0
0	1.0	0.5

 $=$ 

	7	

kernel

# Linear Filter = Convolution



$$\begin{aligned}
 f(i, j) = & \quad g_{11} I(i-1, j-1) \quad + \quad g_{12} I(i-1, j) \quad + \quad g_{13} I(i-1, j+1) \quad + \\
 & g_{21} I(i, j-1) \quad + \quad g_{22} I(i, j) \quad + \quad g_{23} I(i, j+1) \quad + \\
 & g_{31} I(i+1, j-1) \quad + \quad g_{32} I(i+1, j) \quad + \quad g_{33} I(i+1, j+1)
 \end{aligned}$$



# Linear Filter = Convolution

---

$$f[m, n] = I \otimes g = \sum_{k, l} I[m - k, n - l] g[k, l]$$

with  $\sum_{k, l} g[k, l] = 1$

# Useful Imaging Methods - Image Processing

- Gaussian Filtering convolves a low-pass filter with the image, which gives a smoothed image
- This can reduce pixilation effects and aid in false positives for corner detection
- Too much results in blurred image which can make corner detection hard

$$\frac{1}{21} \begin{matrix} 1 & 3 & 1 \\ 3 & 5 & 3 \\ 1 & 3 & 1 \end{matrix}$$

3x3 Gaussian Kernel



Original Image



3x3 Gaussian Filter

# Image Smoothing With Gaussian

```
figure(3);  
sigma = 3;  
width = 3 * sigma;  
support = -width : width;  
gauss2D = exp( - (support / sigma).^2 / 2);  
gauss2D = gauss2D / sum(gauss2D);  
smooth = conv2(conv2(bw, gauss2D, 'same'), gauss2D, 'same');  
image(smooth);  
colormap(gray(255));  
  
gauss3D = gauss2D' * gauss2D;  
tic ; smooth = conv2(bw,gauss3D, 'same'); toc
```

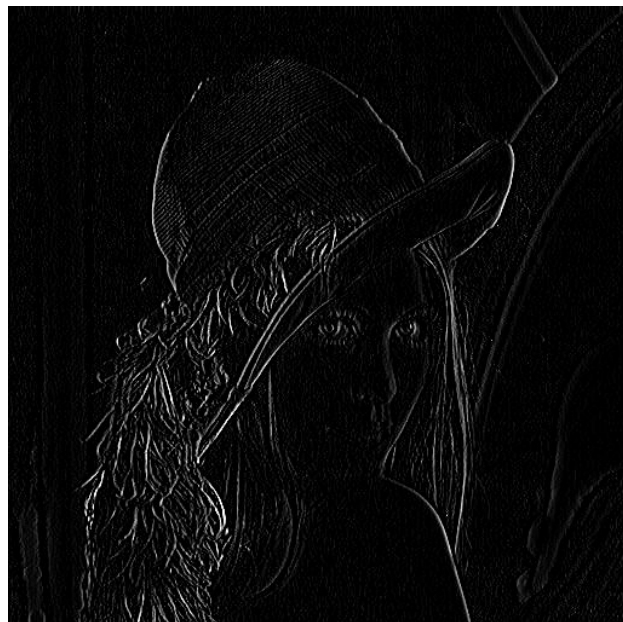
# Useful Imaging Methods - Image Processing

- High Pass filtering can give a simple edge detection
- Subsequent edge detection in two directions gives a simple point detection

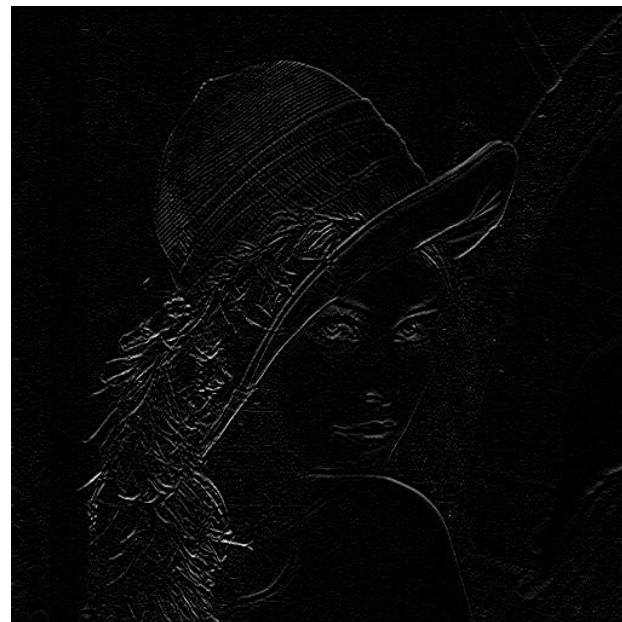
0 0 0  
-1 2 -1  
0 0 0  
3x3 vertical  
edge detector

0 -1 0  
0 2 0  
0 -1 0

3x3 horizontal  
edge detector



Vertical Edge



Horizontal Edge

# Edge Detection With Smoothed Images

---

```
figure(4);  
[dx,dy] = gradient(smooth);  
gradmag = sqrt(dx.^2 + dy.^2);  
gmax = max(max(gradmag));  
imshow(gradmag);  
colormap(gray(gmax));
```

```
figure(6)  
edge(bw, 'sobel')
```

# Useful Imaging Methods - Image Processing

- Median Filtering is a nonlinear filter where the median value in the kernel window is given to the pixel
- Removes “salt and pepper” noise and jittering
- Results in a loss of texture



Noisy Image



3x3 Gaussian Filter



3x3 Median Filter

# Useful Imaging Methods - Image Processing

- In thresholding, all pixels with a value below a certain level are rounded to 0, all pixels above are rounded to max
- Thresholding is often a prerequisite to segmentation algorithms which find connected regions
- Centroids of segments make excellent feature points

