Vision 1
15-491 CMRoboBits

Manuela Veloso
And
Brett Browning

Fall 2007

Outline

- Vision overview
- Camera sensors
- Image representations
- Challenges in vision
- Calibration
- Low-level vision algorithms
- Summary

What is Robot Vision?

- Machine Vision:
  - The study of techniques that obtain higher level information from images
- Robot Vision:
  - Techniques to extract information about the world from images obtained from the robot's cameras to enable a robot to perform its tasks
  - Many constraints known a priori
  - Real-time performance often crucial

Vision Algorithms Overview

- For robots, vision used for two key problems
- Finding objects
  - Object detection, recognition, and tracking
- Understanding structure
  - Structure from motion/stereo
  - SLAM
  - Structure/shape from texture

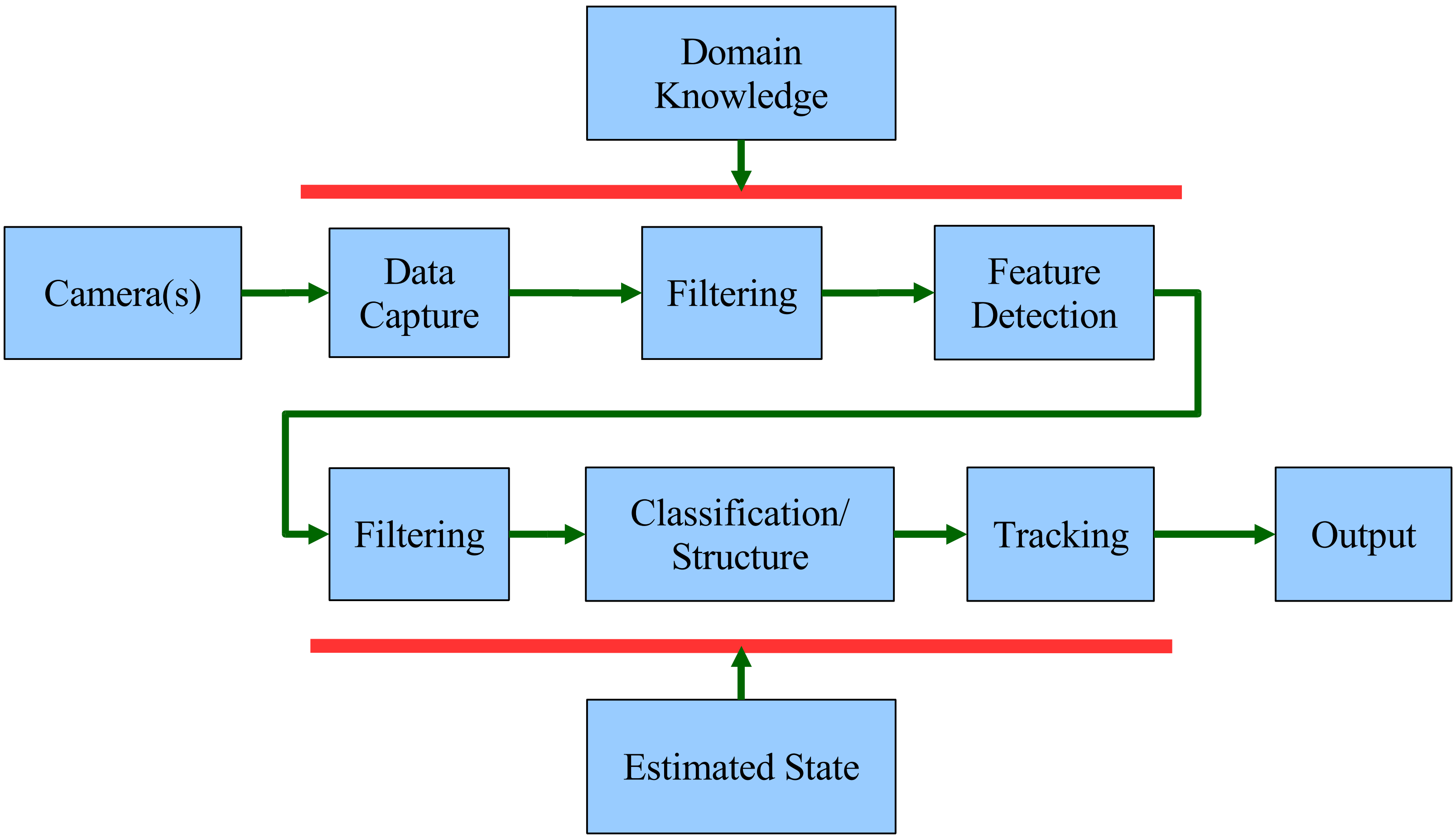
Vision Examples

- Tracking videos:
  - http://www.umiacs.umd.edu/~ramani/movies/list.html
- Face detection:
  - http://www.merl.com/projects/FaceDetection/
- Car tracking:
  - http://www.mobileye-vision.com/
- Stereo:
  - http://labvisione.deis.unibo.it/~smattoccia/stereo.htm
- SFM:
  - http://www.vis.uky.edu/~dnister/Research/research.html
- SLAM:
  - http://www.doc.ic.ac.uk/~ajd/

Many Techniques Exist

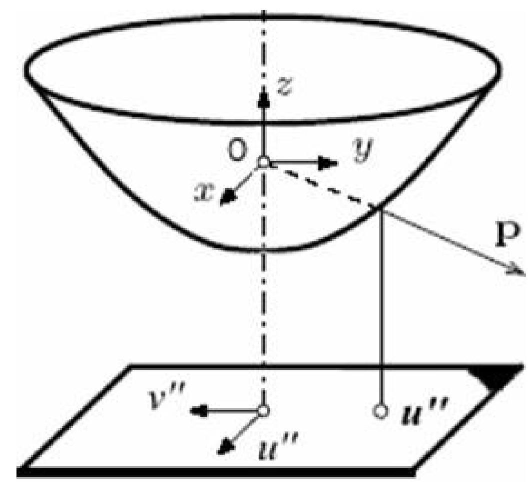
- Low level vision
  - Filtering, thresholding, segmentation, classification
  - Feature detection (corner, edge, SIFT, region, ...)
  - Optical flow, feature tracking, stereo
- High level vision
  - Learned object classifiers
  - Graphical models for detection, tracking, SLAM, scene understanding
  - Multi-view geometry for structure for motion, stereo
  - Many more...

Typical Parts of a Vision System

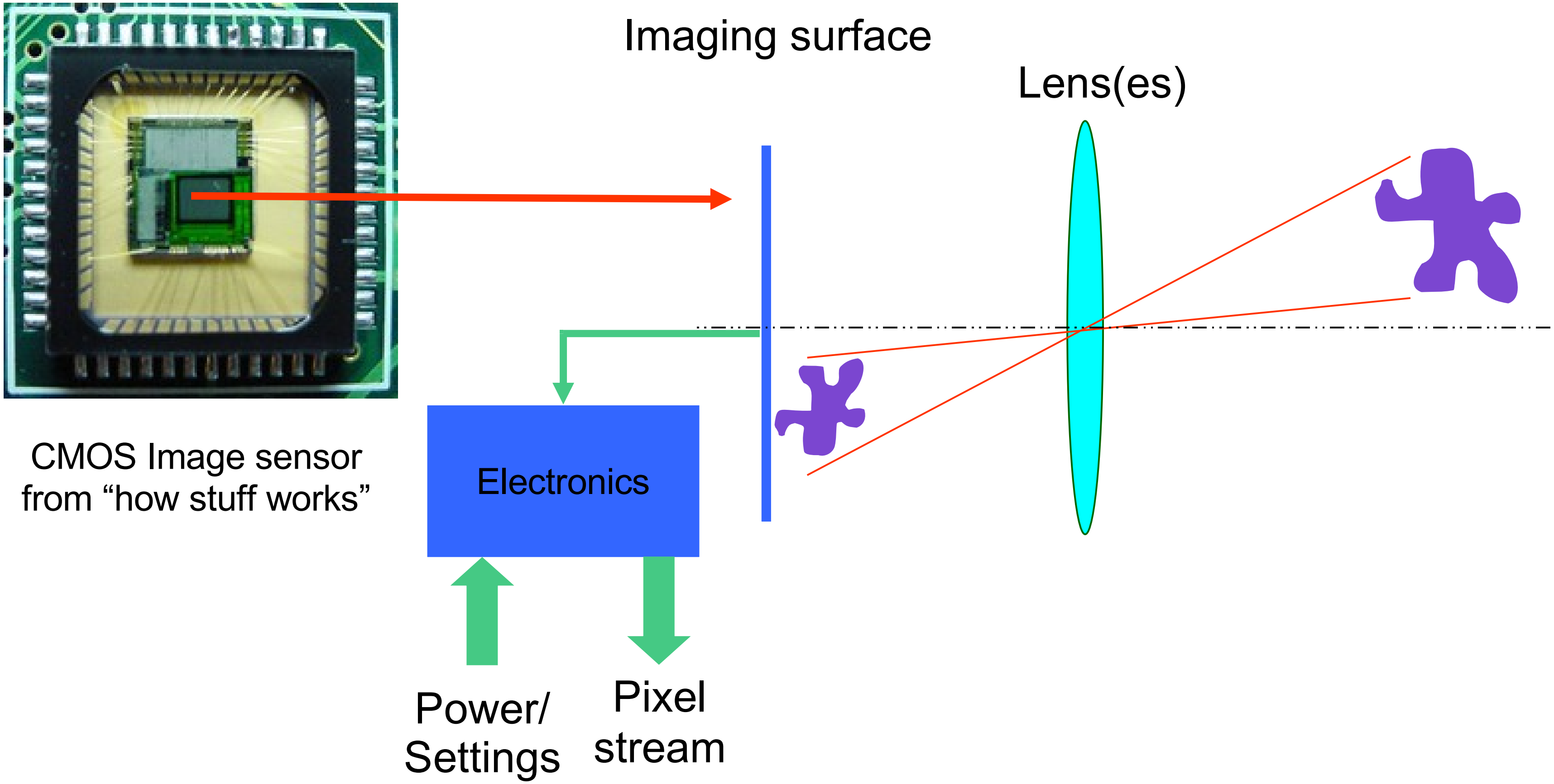


Cameras As Sensors

- Most machine vision cameras consist of
  - Photon sensitive sensor elements with filters
  - Mirror(s) and/or lens(es) to manipulate light
  - Digital frame capture electronics
  - Optionally structured light sources

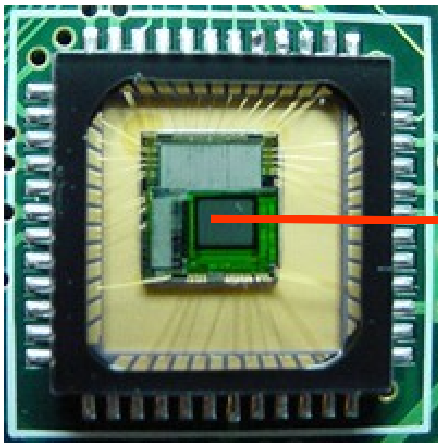


Parts of a Digital Camera



Variations

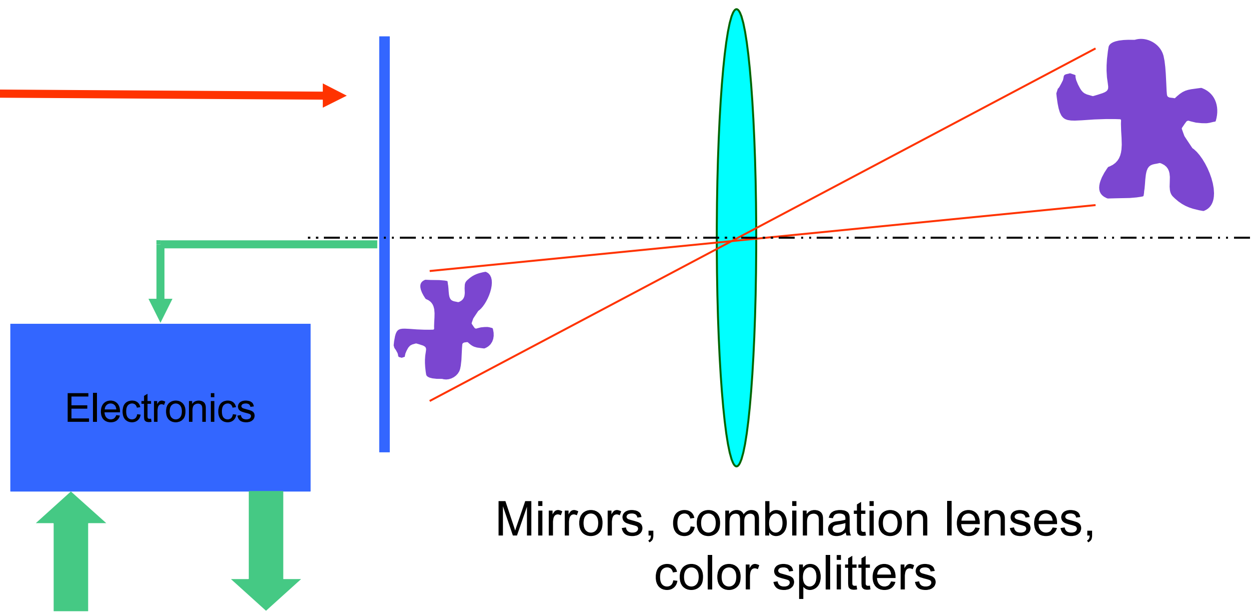
CCD, CMOS imagers



CMOS Image sensor
from “how stuff works”

Imaging surface

Lens(es)



Mirrors, combination lenses,
color splitters

Power/
Settings

Pixel
stream

IEEE1394, USB, S-video,
coax, camera-link

What's in an Image?

- Usually a 2D array of pixels
  - A pixel is usually 8-bits but can be more
  - Sometimes color filters, and/or more than 1 array
- Conversion to a 2D array in a standard color space

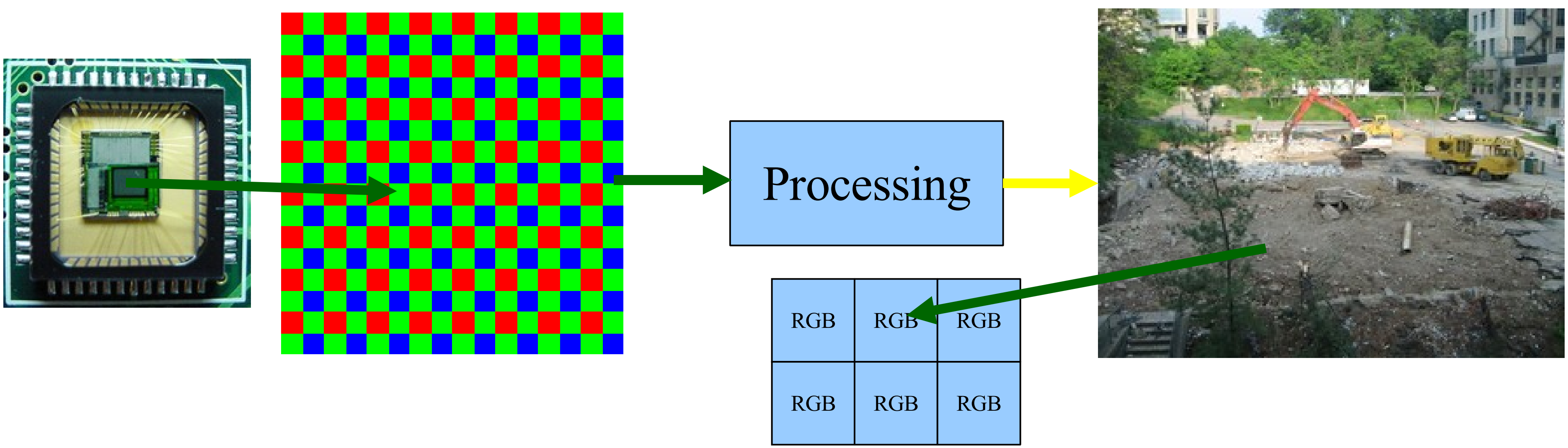


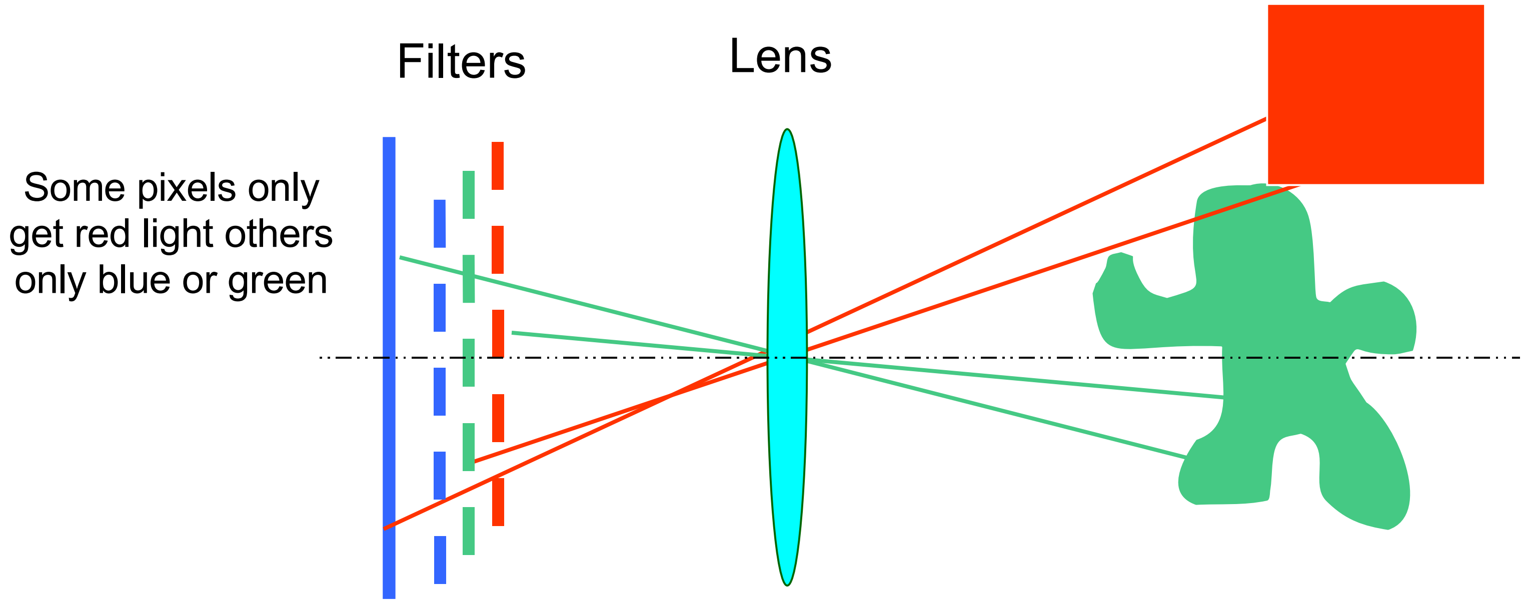
Image Example

Some Vision Challenges

- Lighting changes
  - Intensity and color is a function of surface, lighting conditions, and camera optics
- Scale
  - Distance to object changes scale (pixel area)
- Loss of information
  - Camera is a 3D to 2D mapping, so depth (or scale) is lost in the transformation
- Occlusion
- Noise

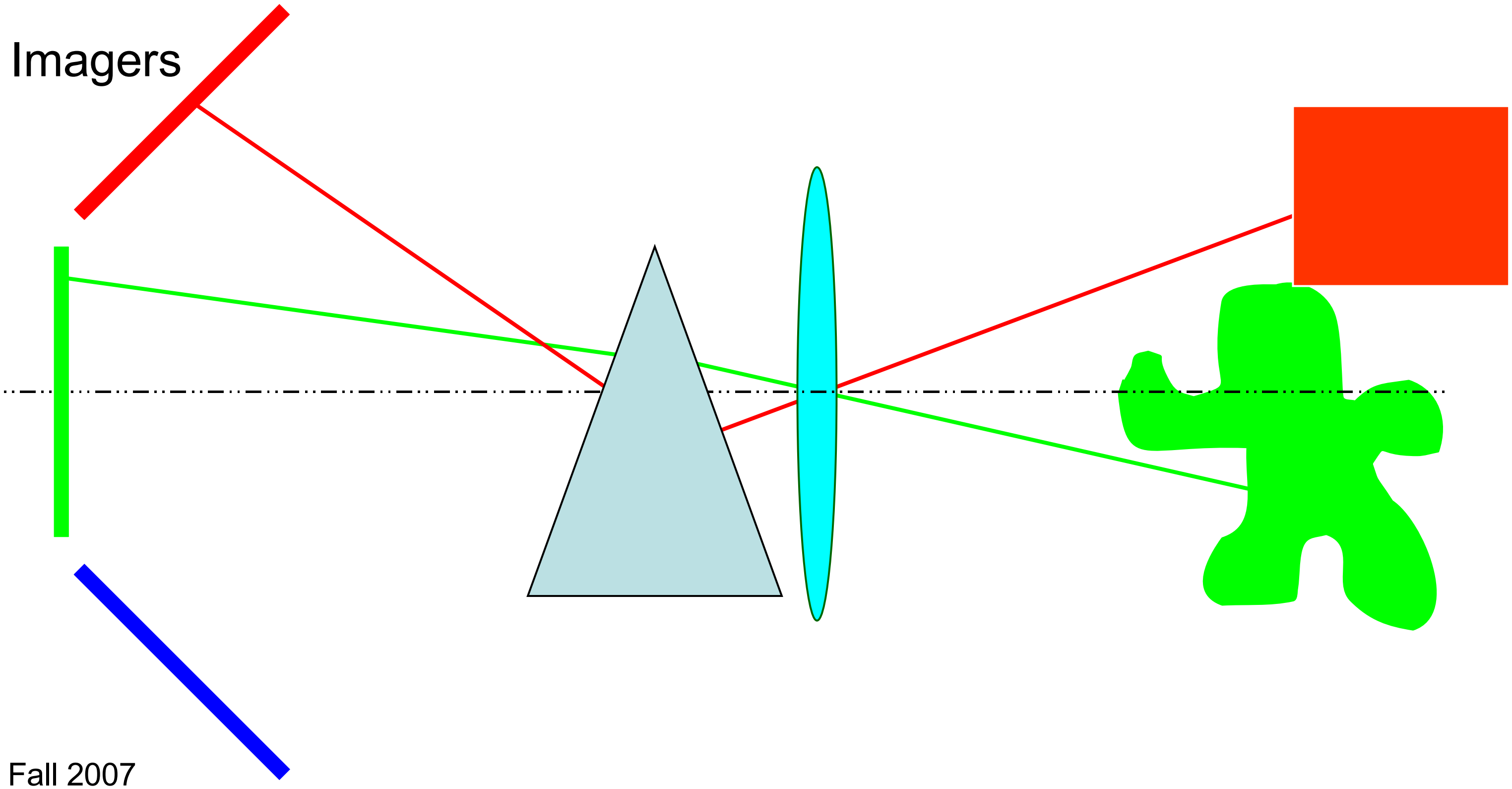
Color Images

- White light really consists of many colors
  - Remember Isaac Newton!!!
  - Let some pixels collect light of only one color
  - Must combine and filter to get full color image



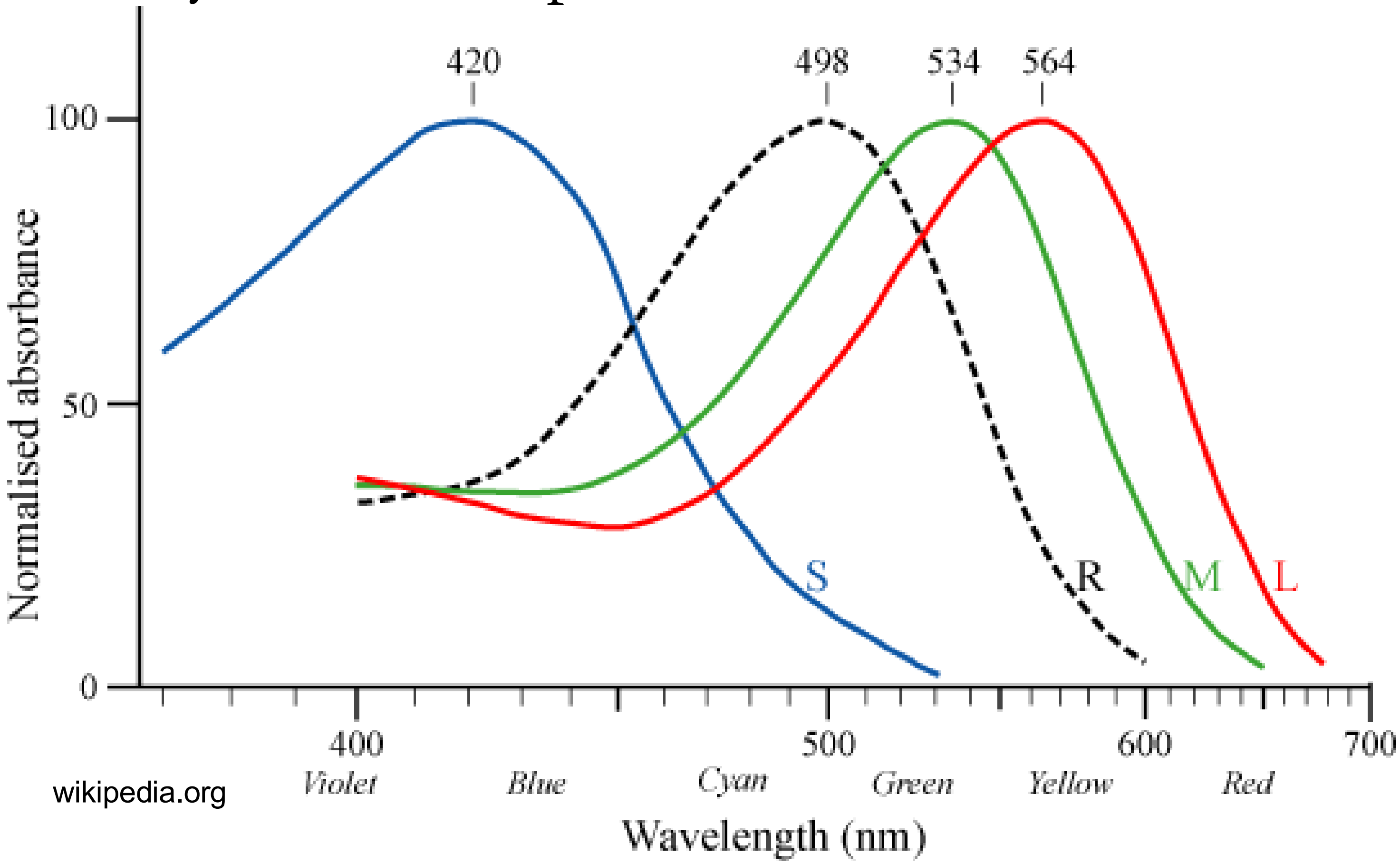
3 CCD Imagers

- An alternative is to *split* light by color and to use multiple image sensors, one for each color



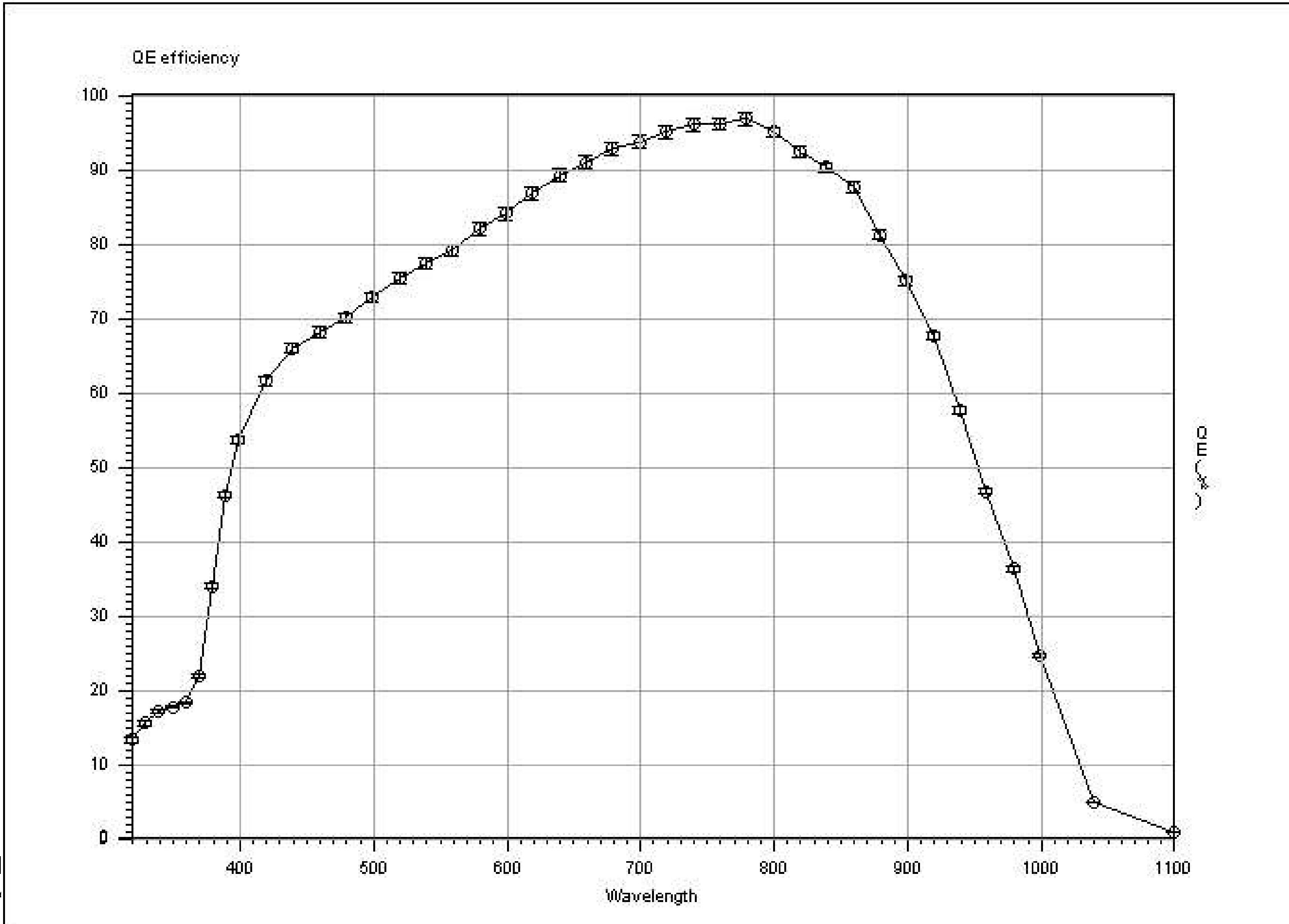
Why RGB?

- Humans (mostly) have rods & 3 x cones
  - Rods: Grayscale, cones provide color "RGB"

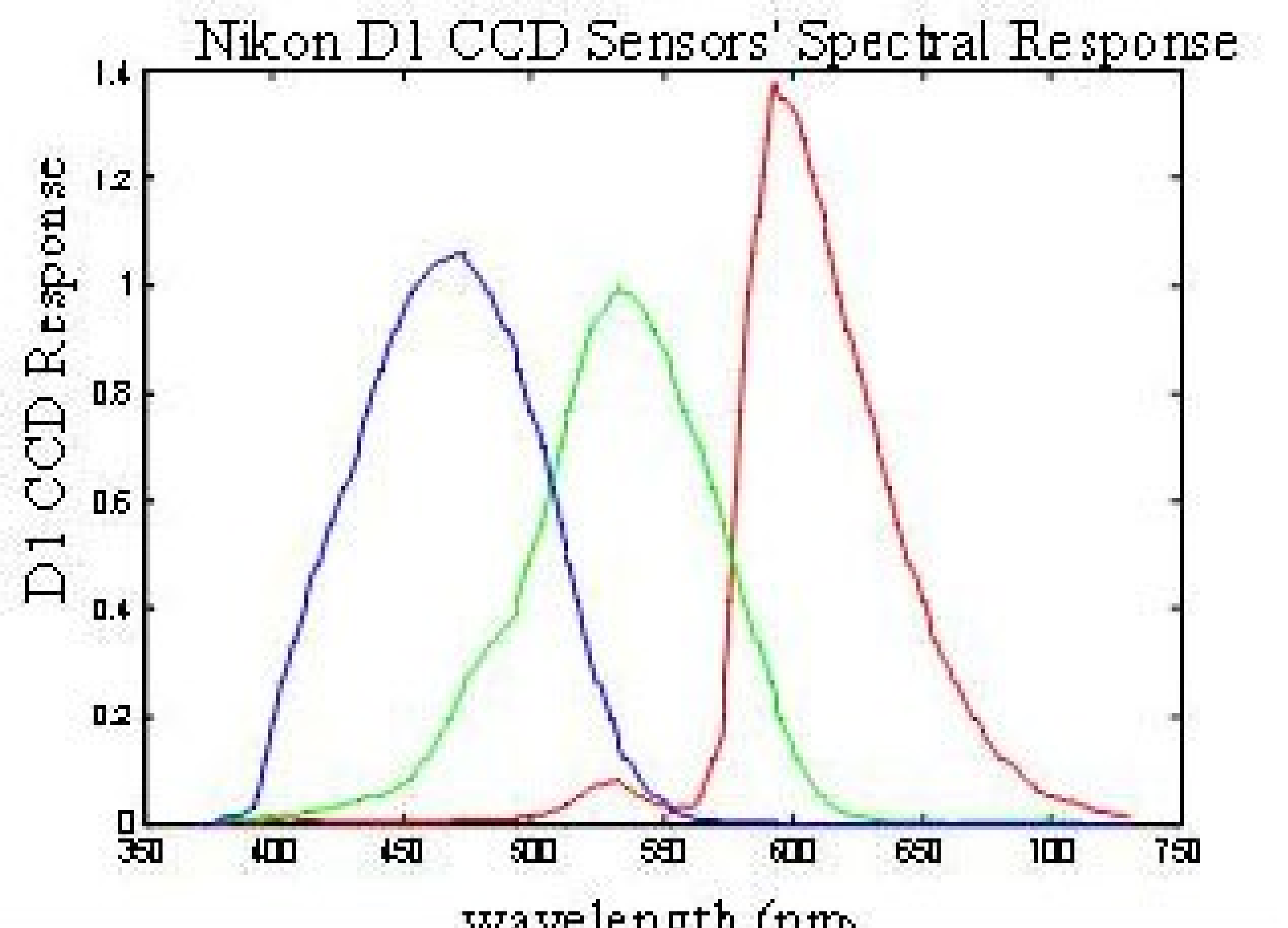


wikipedia.org

CCD Quantum Efficiency

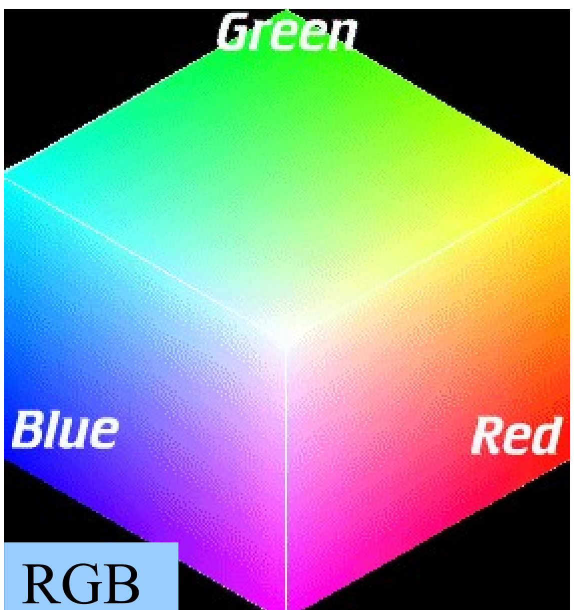


Camera Spectral Response

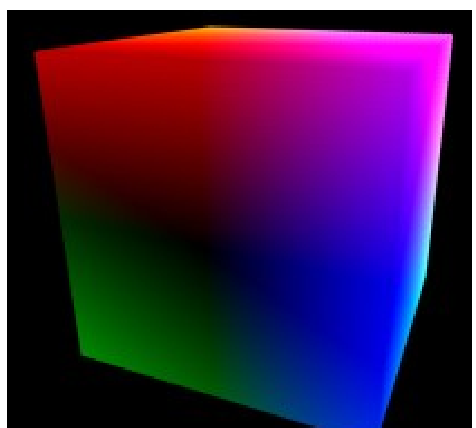


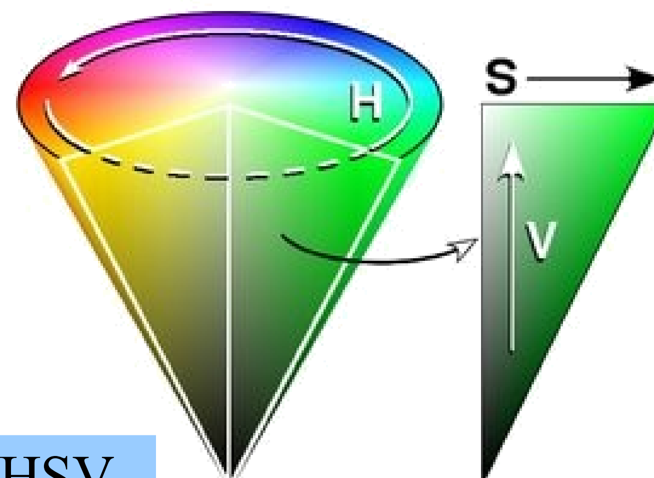
Color Spaces

- Many ways to represent color
  - RGB (red, green, blue), nRGB
  - YUV, or Y Cr Cb (luminance + chroma)
  - HSV or HSL (hue, saturation, value)



RGB


YUV



HSV

Why Different Color Spaces

- Its all about invariance
  - Color is a function of surface properties and lighting
  - Example: Red surface in green light
- The problem
  - As light changes color changes
  - Different color spaces have different sensitivity to lighting changes
  - e.g. HSV => V = "grayness", HS = "2D color"
  - No perfect approach however

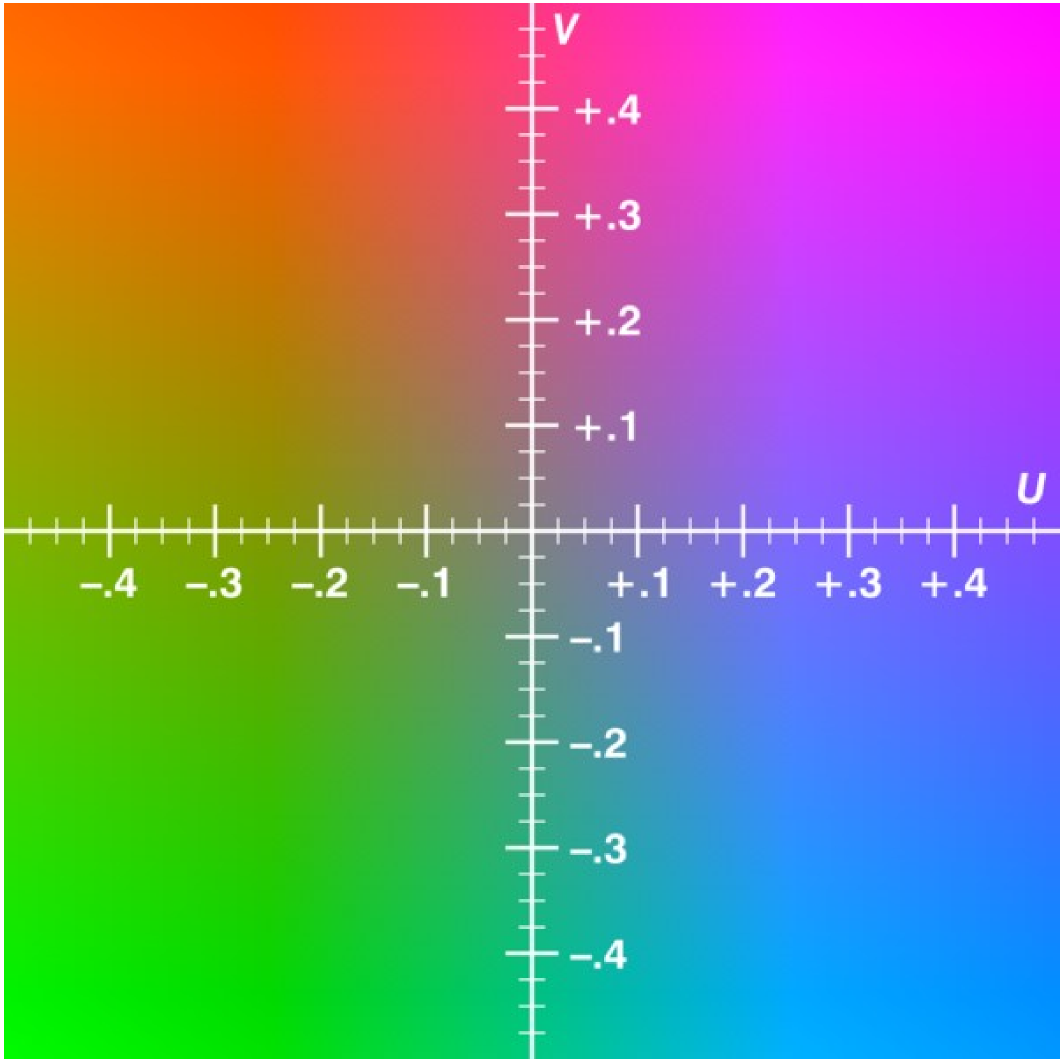
Common Color Spaces

- RGB: Common output from camera
- nRGB: normalize each color channel

$$\hat{R}=\frac{R}{R+G+B},\quad \hat{G}=\frac{G}{R+G+B},\quad V=\frac{1}{3}\left(R+G+B\right)$$

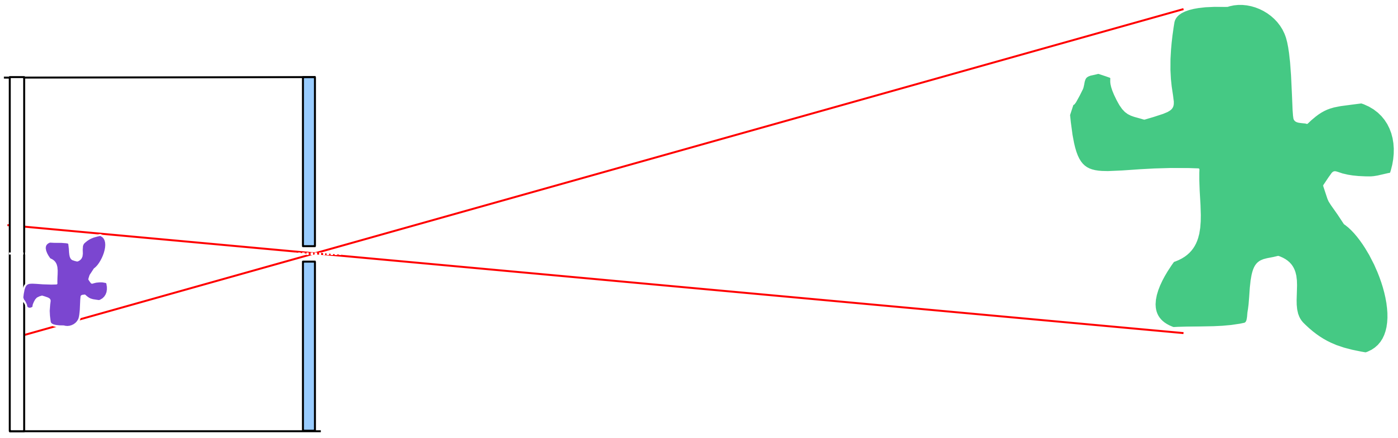
- YUV: linear transform of RGB,
  - Y= Luminance, U,V= Chrominance (color)
  - Really called YCbCr

YUV Plane



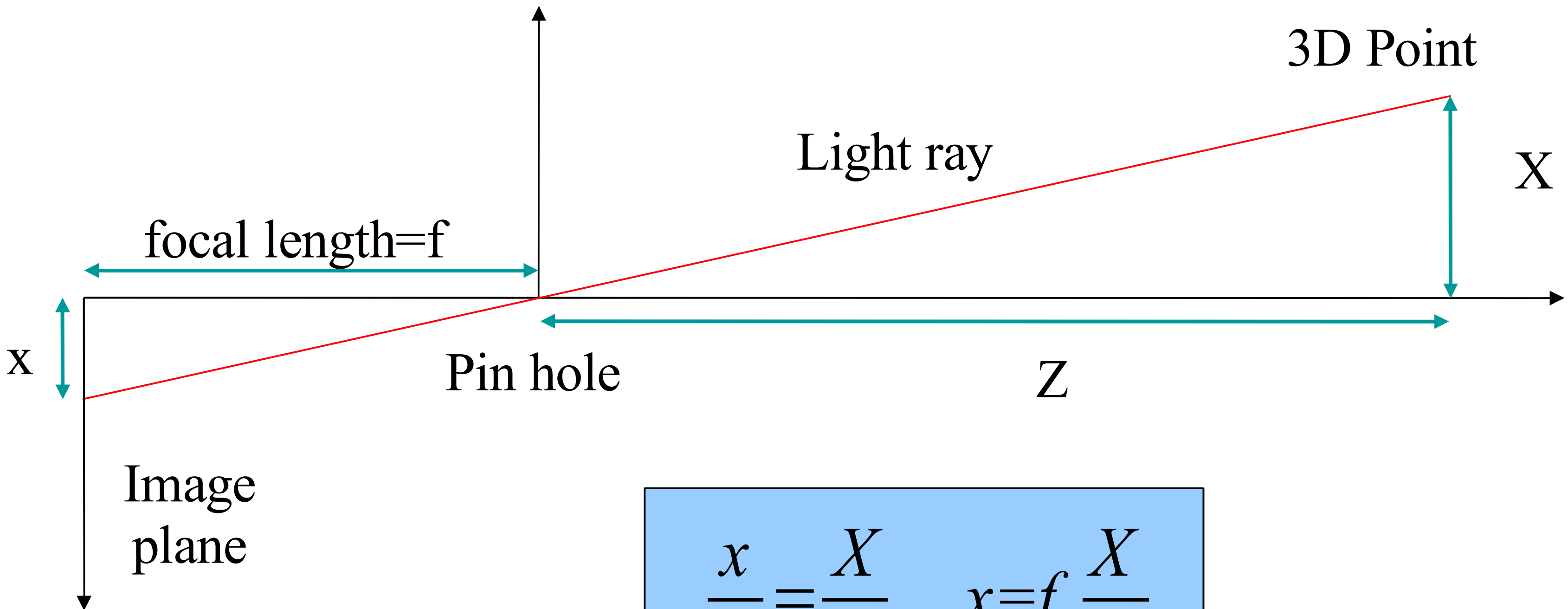
Camera Geometry

- A Camera maps from points in 3D to a point on the 2D image plane
- To describe it mathematically let's look at a simple camera: an ideal pin-hole camera



2D Pin-hole Camera

- Using similar triangles



$$\frac{x}{f}=\frac{X}{Z}, \quad x=f\,\frac{X}{Z}$$

Ideal Camera Transformation

- Extending to 3D we have for an ideal camera

$$x = f\,\frac{X}{Z}, \quad y = f\,\frac{Y}{Z}$$

- Note:
  - This assumes the coordinate systems are aligned
  - All 3D points on the same light ray map to the same 2D image coordinate

In More Detail

- A little manipulation we get

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \propto \begin{pmatrix} f\alpha_x & 0 & c_x \\ 0 & f\alpha_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

$$x \propto KX$$

Image coordinates

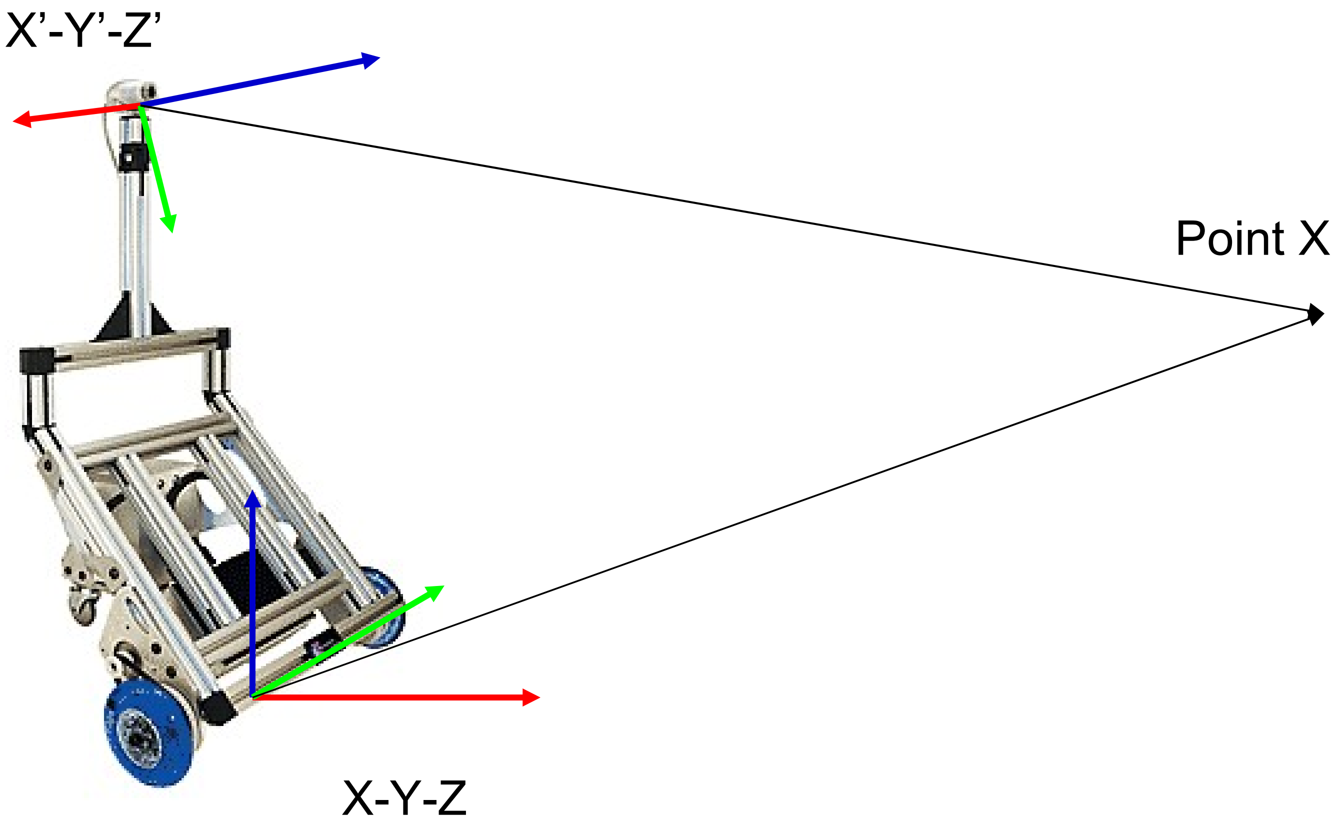
World coordinate

Camera matrix

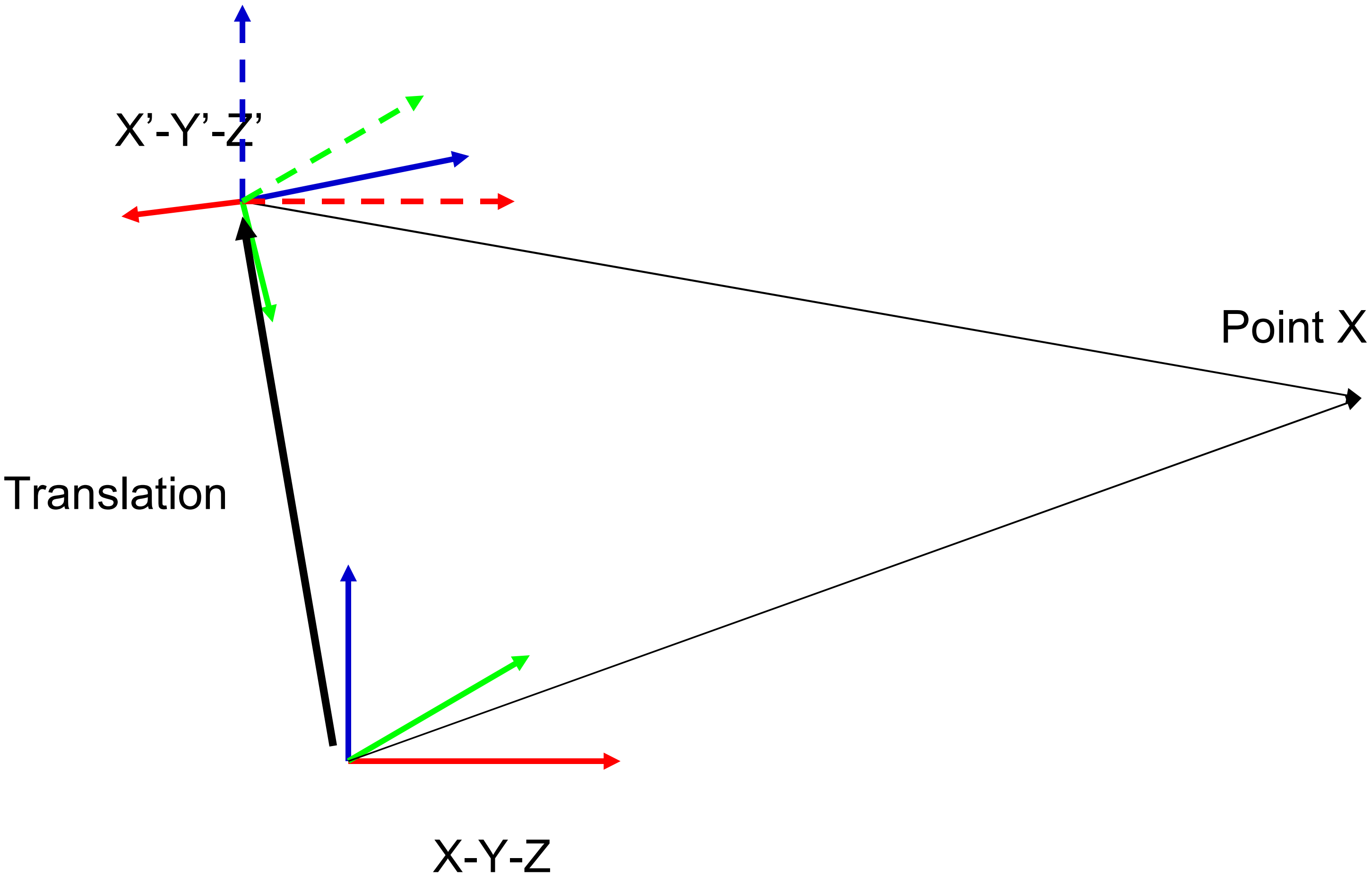
Focal length

Camera center

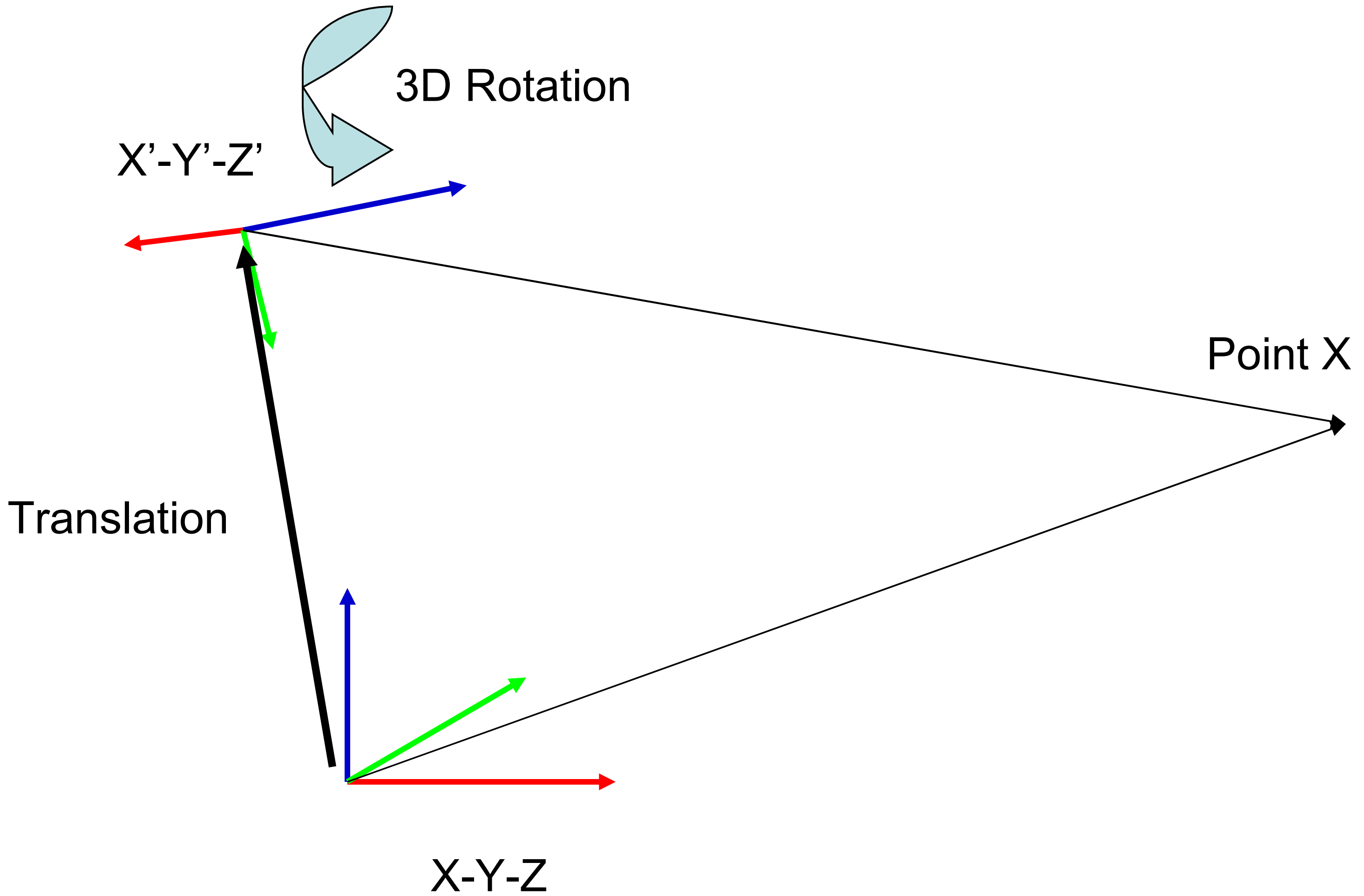
Coordinate Frames



Coordinate Frames



Coordinate Frames



3D Translations

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} = \vec{X} - \vec{X}_0$$

3D Rotations

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R\,X$$

3D Rotation and Translation

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{pmatrix} = R\left(X - X_0\right)$$

All Together

Extrinsic parameters

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \propto KR\left(X - X_0\right)$$

Intrinsic parameters

$$K = \begin{pmatrix} f\alpha_x & 0 & c_x \\ 0 & f\alpha_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Camera Calibration

- Intrinsic calibration
  - Put *known* pattern in many orientations
  - Solve for *K*, (and distortion parameters)
- Extrinsic calibration
  - Put *known* pattern in *known* location, solve for R, X_0
- Calibration is a big optimization problem
- Can also model noise/distortion characteristics

Low-Level Vision

- Goal
  - Cleanup/filter image, transform image space
  - Extract useful information for high-level detection
- Examples
  - Smoothing, edge detection, corner detection, feature detection (more later)
  - Color space transformation
  - Segmentation

Low-Level Image Processing

- Why?
  - More compact representation
  - More invariant to lighting, viewpoint changes, etc.
  - Easier to process
  - Essential for performing higher level operations

Filtering 101

- Goal is to remove noise from an image
- Usually achieved by *convolution* of a smoothing *operator* with the image
- Many feature detectors also use convolution
- So let's look at convolution!

Convolution Example

- Operator is a matrix
- Result is another image

-1	0	1
-2	0	2
-1	0	1

Operator Window

Image

3	4	20	21	20	19	0	1
3	2	3	20	20	19	2	1
2	0	3	19	21	20	3	0
2	1	1	25	24	19	21	5
5	10	19	20	23	3	2	0
1	3	10	20	19	24	1	1

Convolution Example

- Operator is a matrix
- Result is another image

-1	0	1
-2	0	2
-1	0	1

Operator Window

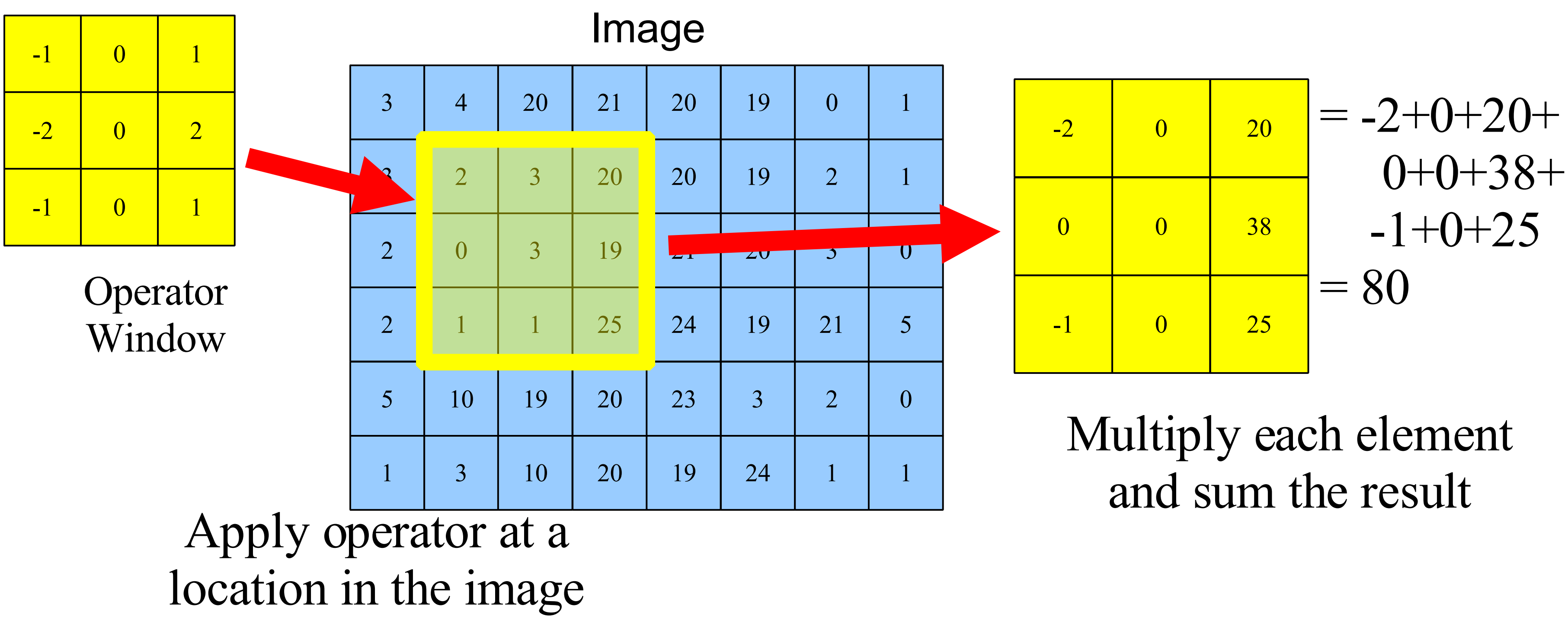
Image

3	4	20	21	20	19	0	1
3	2	3	20	20	19	2	1
2	0	3	19	21	20	3	0
2	1	1	25	24	19	21	5
5	10	19	20	23	3	2	0
1	3	10	20	19	24	1	1

Apply operator at a location in the image

Convolution Example

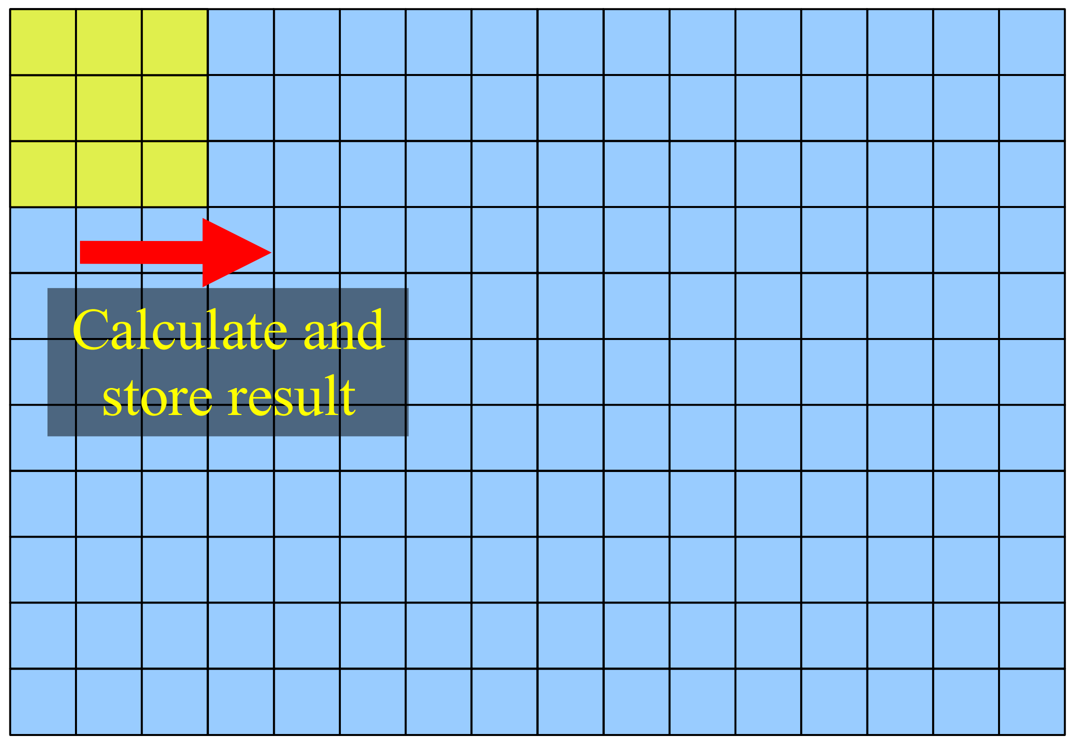
- Operator is a matrix
- Result is another image



Motion of the Operator

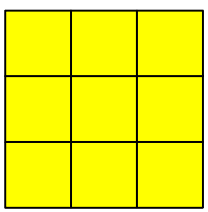
- Start in the top left corner
- Different ways to handle border regions

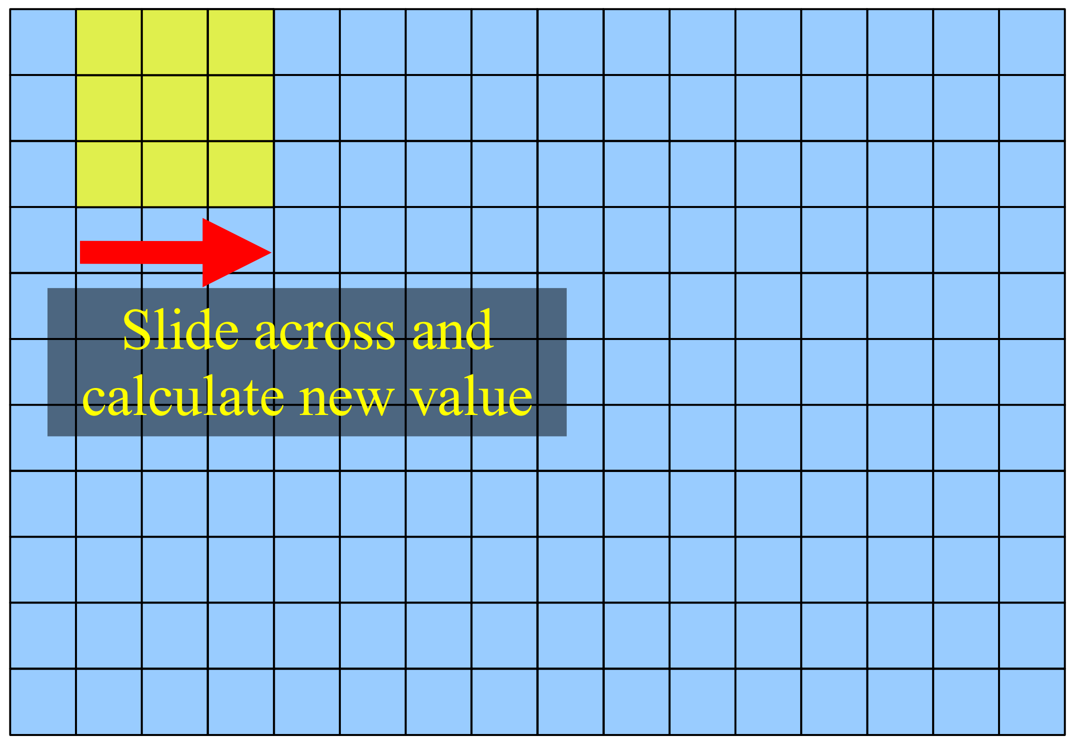
-1	0	1
-2	0	2
-1	0	1



Motion of the Operator

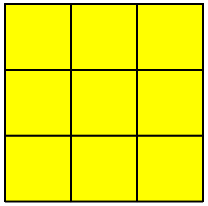
- Slide the operator across the row, storing the output in the result image as you go

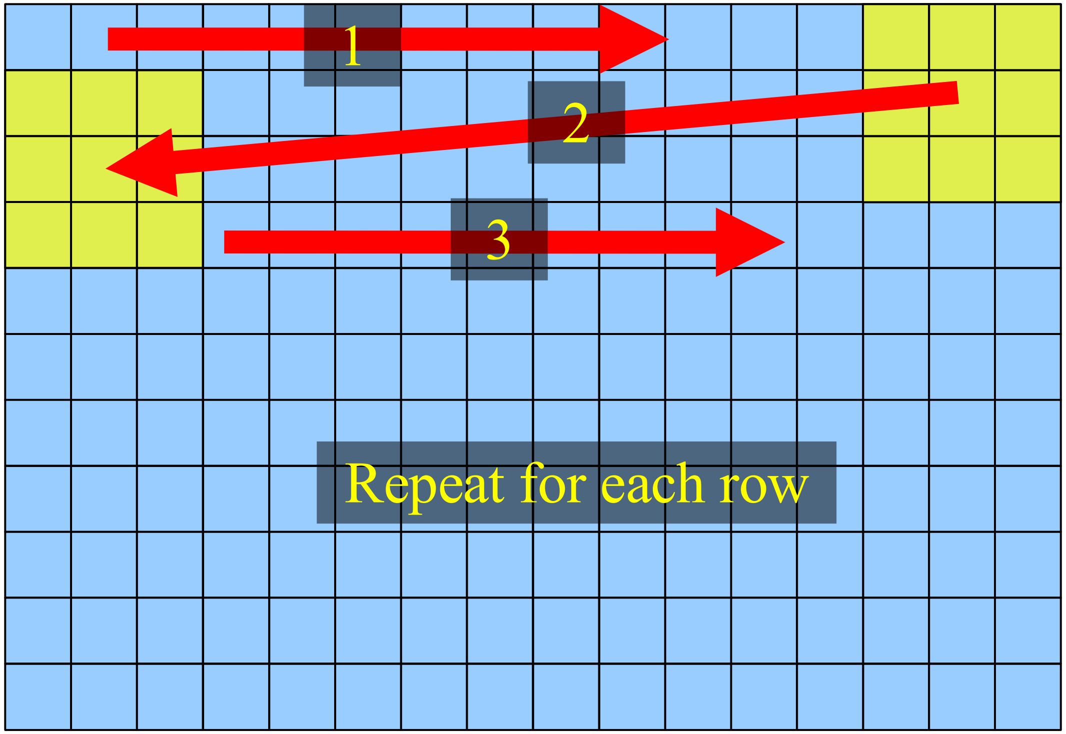




Motion of the Operator

- Repeat for the next row, and so on, until we complete the image





Smoothing/Filtering

- Operator mask size/values changes output
  - Smoothing, differentials
  - Related to discretized mathematical operators (e.g. spatial derivatives)
- Experiment with Gimp!

Some Common Filters

- Blurring

1	1	1
1	1	1
1	1	1

- Gaussian smoothing

1	2	1
2	4	2
1	2	1

- Sobel edge detection
  – Gradient operators

-1	0	1
-2	0	2
-1	0	1

-1	-2	-1
0	0	0
1	2	1

- Laplacican

0	-1	0
-1	4	-1
0	-1	0

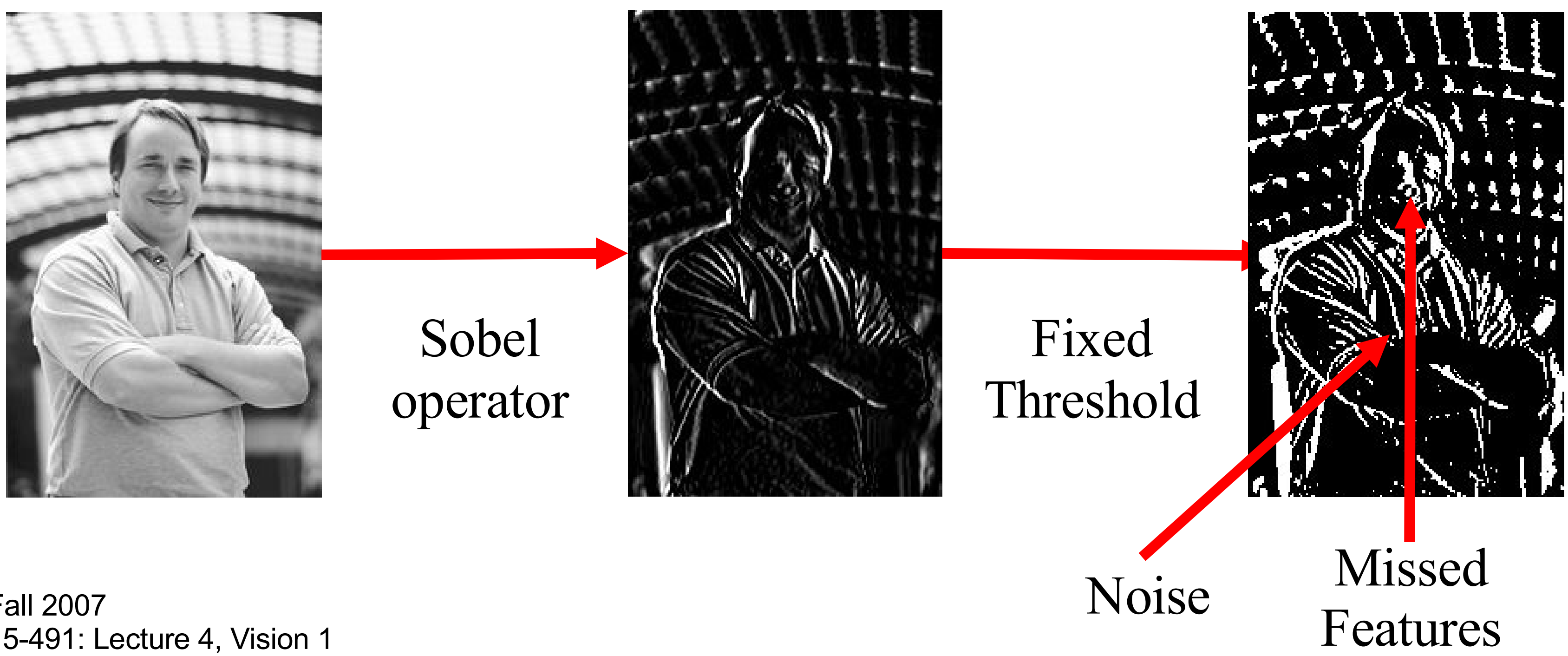
Derivative operators

Feature Detection

- Goal is to extract *interesting* parts of the image
- Many, many approaches and variations
- Usual approach
  - Apply some filters e.g. smoothing, edge detection
  - Apply specialized operator e.g. corner detector
  - Threshold result
  - Find extrema (also called non-maxima suppression)

Simple Example

- Threshold in Gimp with
  - Menu item *Layers* => *Colors* => *Threshold*
- Experiment with different thresholds



Harris Corner Detection

From google image search

Segmentation

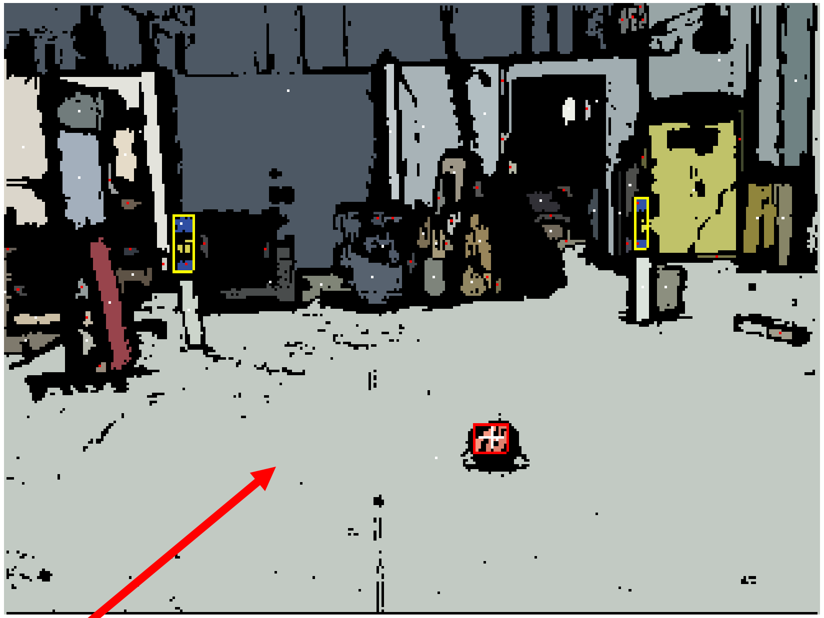
Samples from http://vision.ece.ucsb.edu/

Segmentation

- *Partition* image pixels into groups/clusters
- How do we decide similarity?
  - Intensity, color, texture, …
- How do we decide which pixel belongs to which blob?
  - This is a hard problem…
  - Spectral clustering techniques, watershed algorithms
- An alternative
  - Label pixels as 1 of N classes
  - Group identical pixels

Fast Color Segmentation

- Classify each pixel based on color using pre-defined tables, then group pixels into blobs



Symbolic color value: {Unknown, Red, Yellow, Blue, Green, Floor}

Two Part Process

Label each pixel based on class

Group neighboring pixels of same label

List of color blobs with area, bounding box, statistics, label

Labeling Pixels

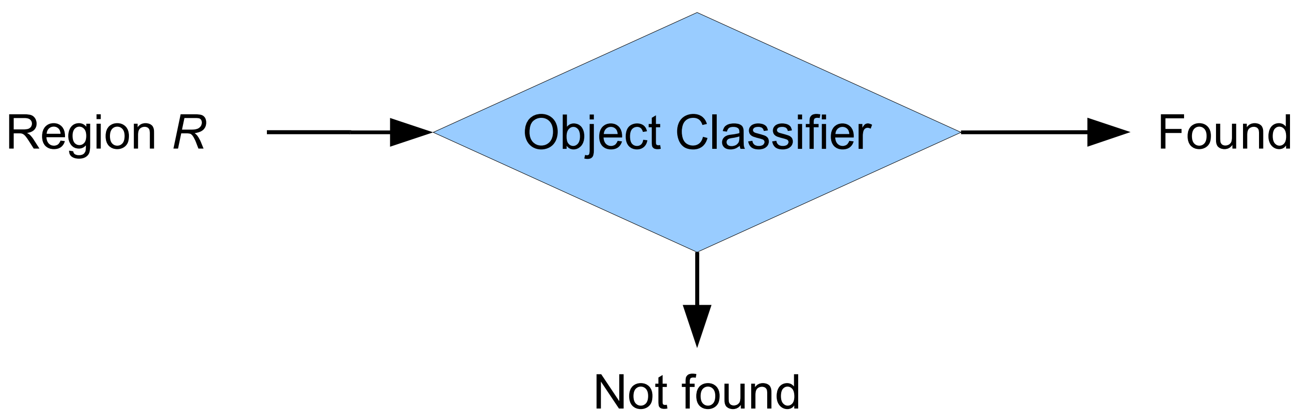
- Use your favorite/fast classifier
  - Nearest neighbors (e.g. Gaussian, look up table)
  - Naïve Bayes classifier
  - Decision trees, etc.
- Performance depends on number of colors, separability, sensitivity to lighting variations
- Can use different color spaces

Growing Regions

- Really a connected components analysis
- Fast algorithm: CMVision [Bruce et al. 00]
  - Run length encoding of rows
  - Tree-based Union Find with path compression of runs to produce final "blobs"
  - http://www.cs.cmu.edu/~jbruce/cmvision
  - Paper provided on web page

Simple Object Detection

- Single colored object
  - Look for blob of right size/shape
- We're back to classification!



Questions?