A white humanoid robot with a pink vest and a red ball on a soccer field.

# **CMSC426**

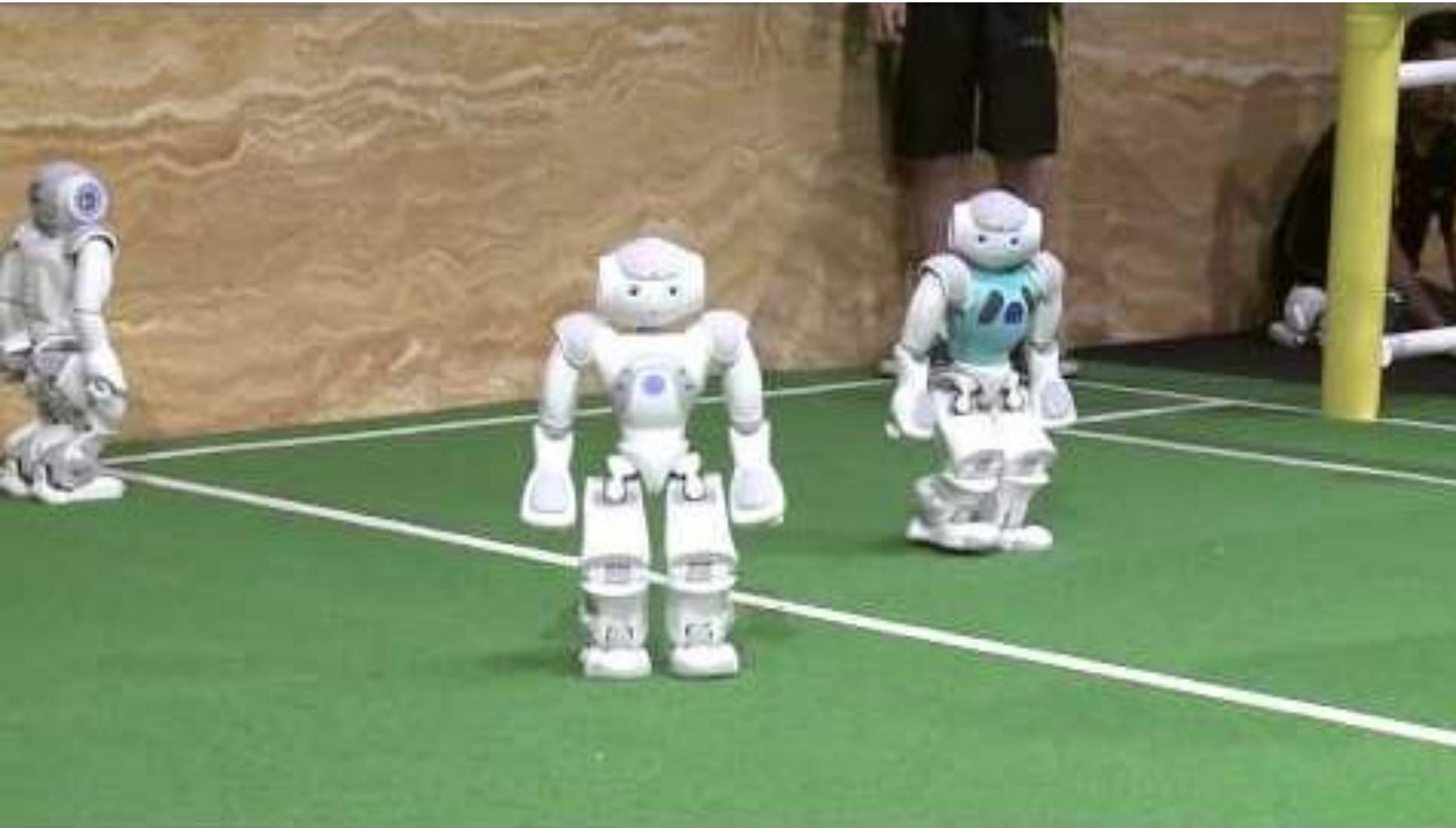
# **Project 1: Color Segmentation**

# **Using GMM for RoboSoccer**

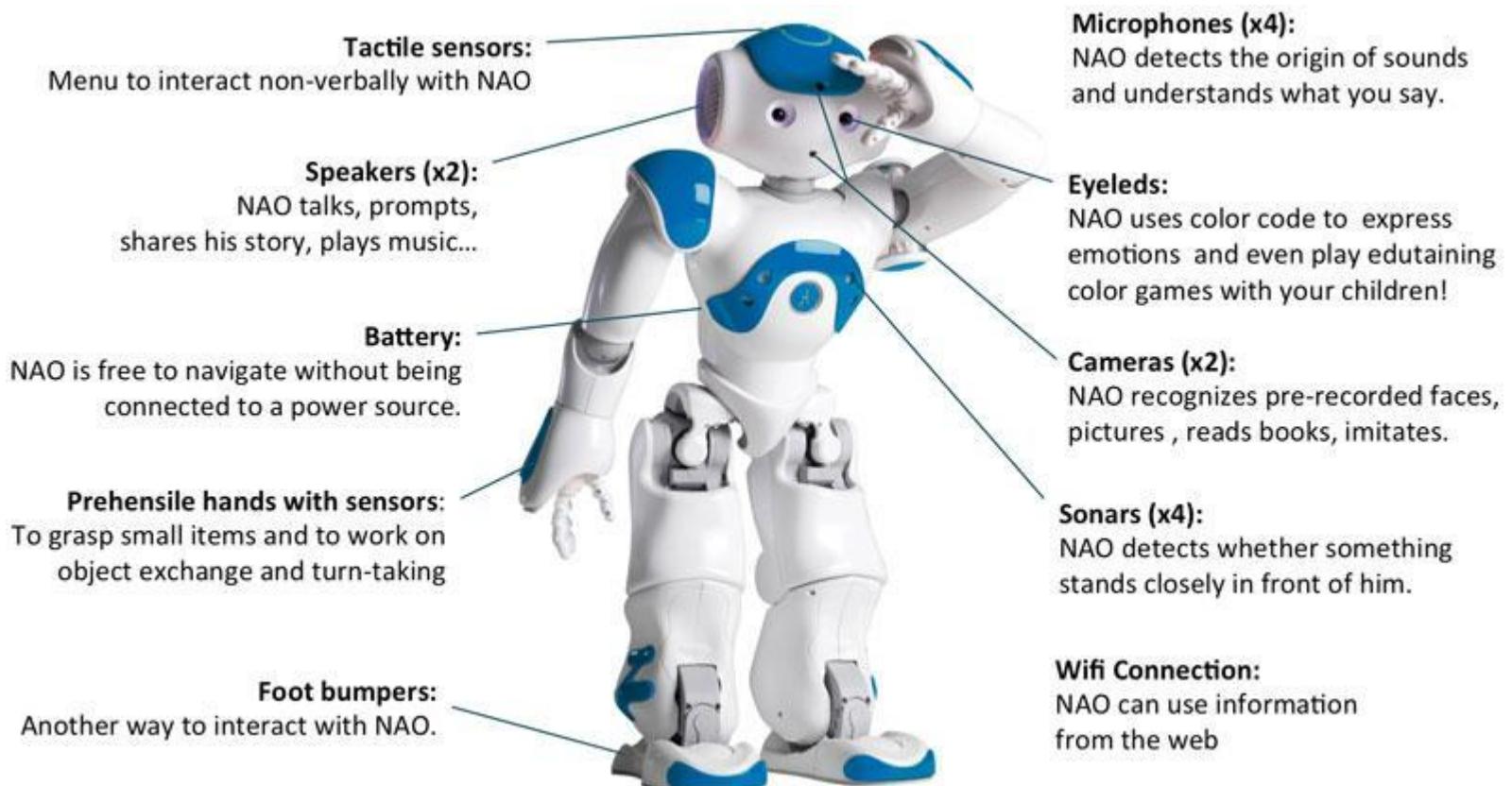
# RoboSoccer



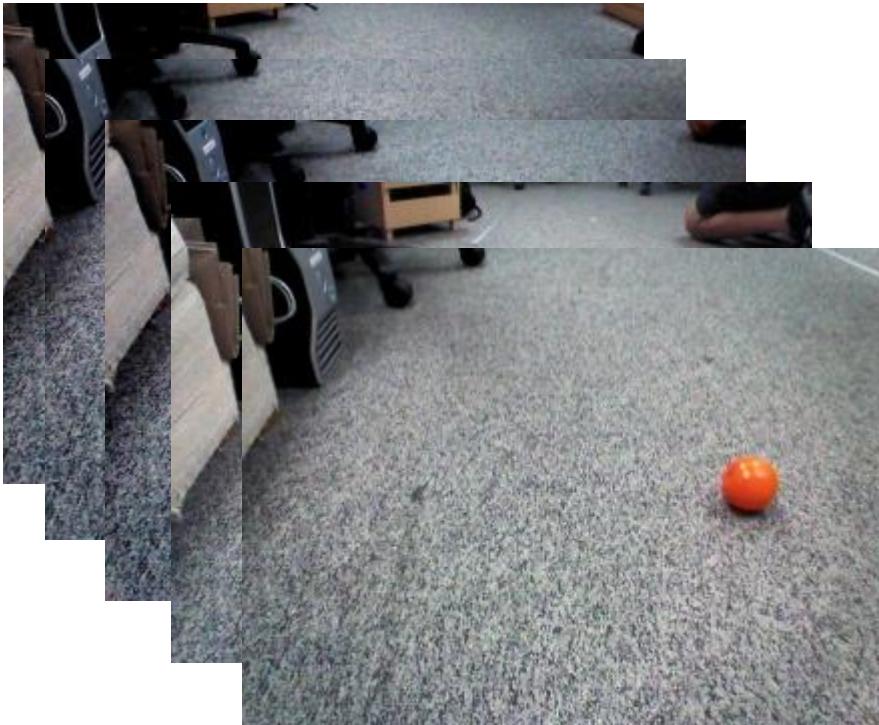
# RoboSoccer



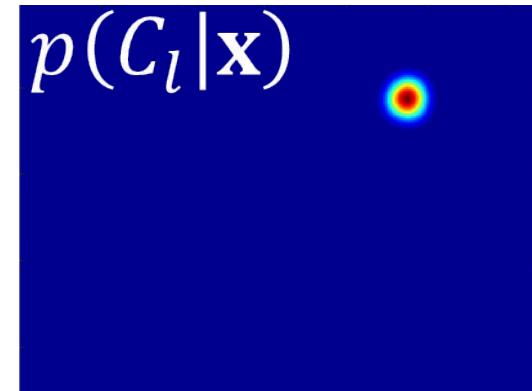
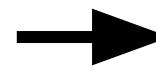
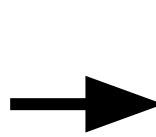
# Data: From the Nao Robot



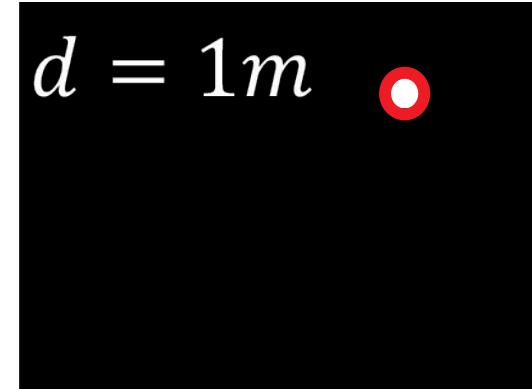
# Color Segmentation: IO for Training



# Color Segmentation: IO for Testing



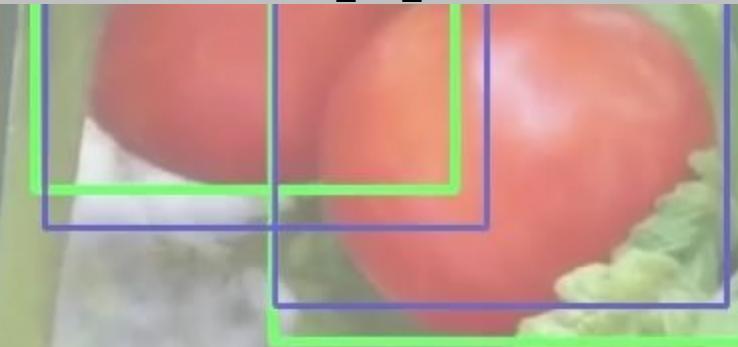
Ball localizer and  
Distance predictor

A thick black arrow pointing downwards from the probability map towards the final output box.

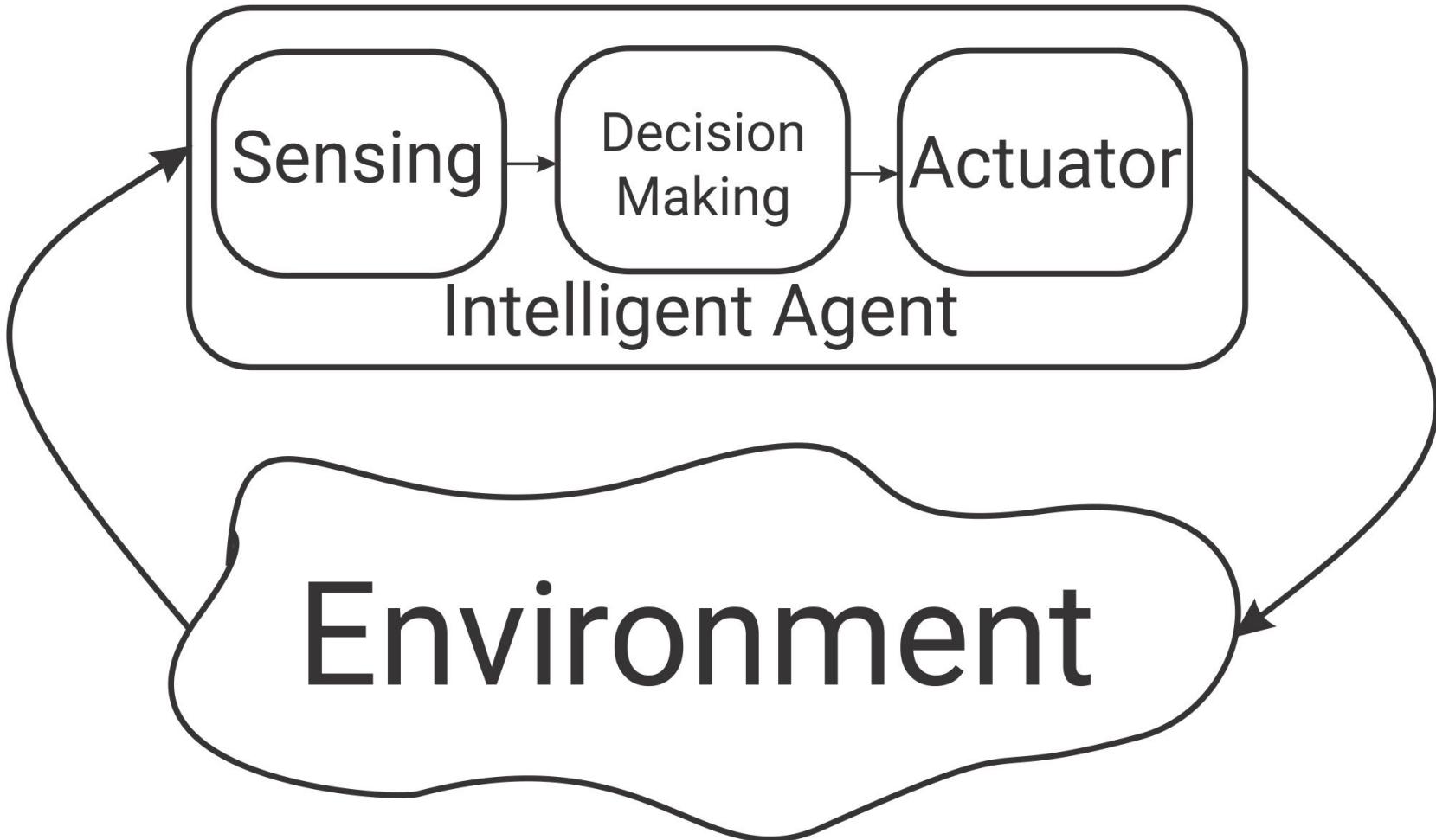
Frame No : 246

Count : 9 / Weight : 1.107kg

# Why? Any Real World Applications?



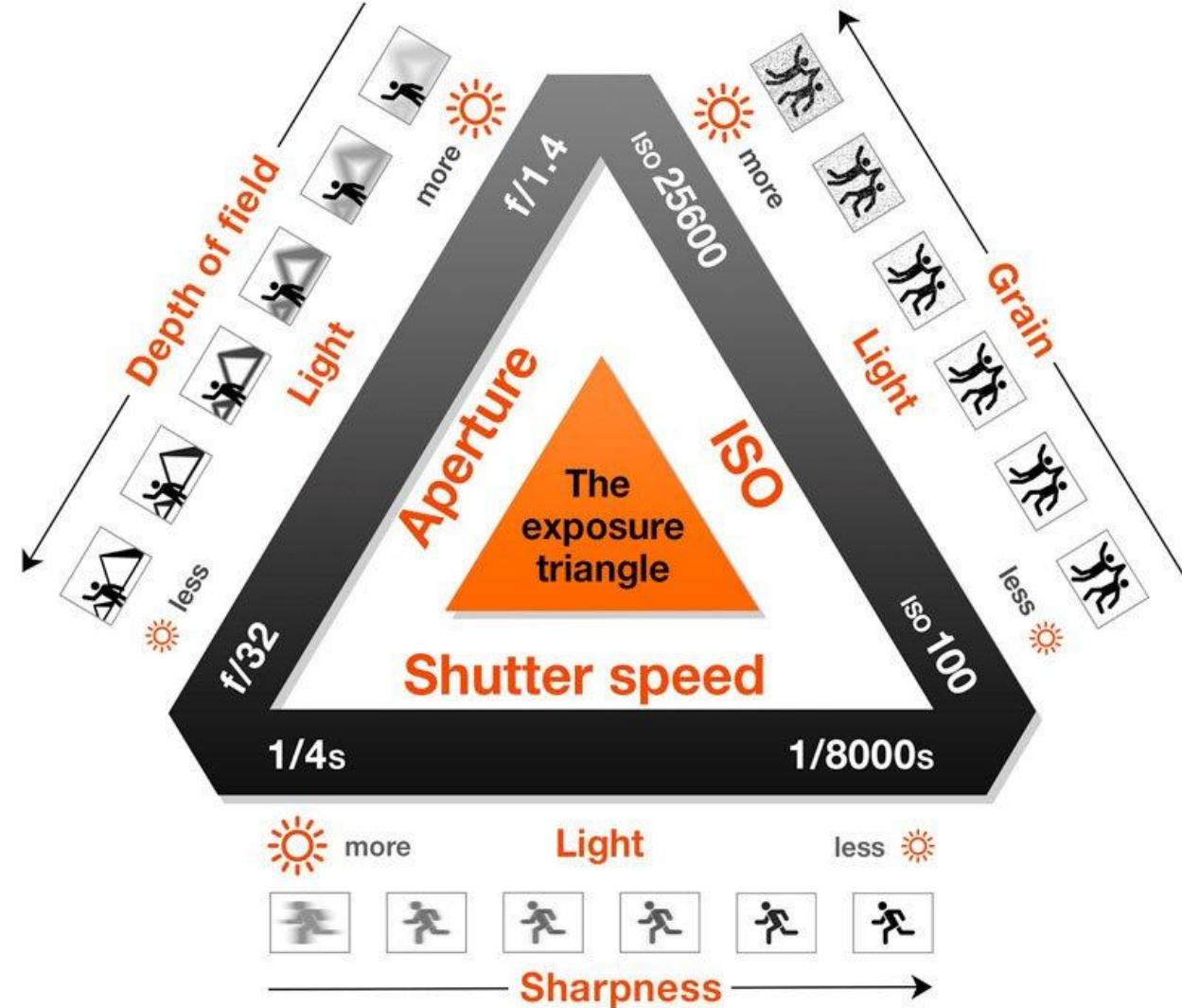
# Intelligent Systems



# Imaging

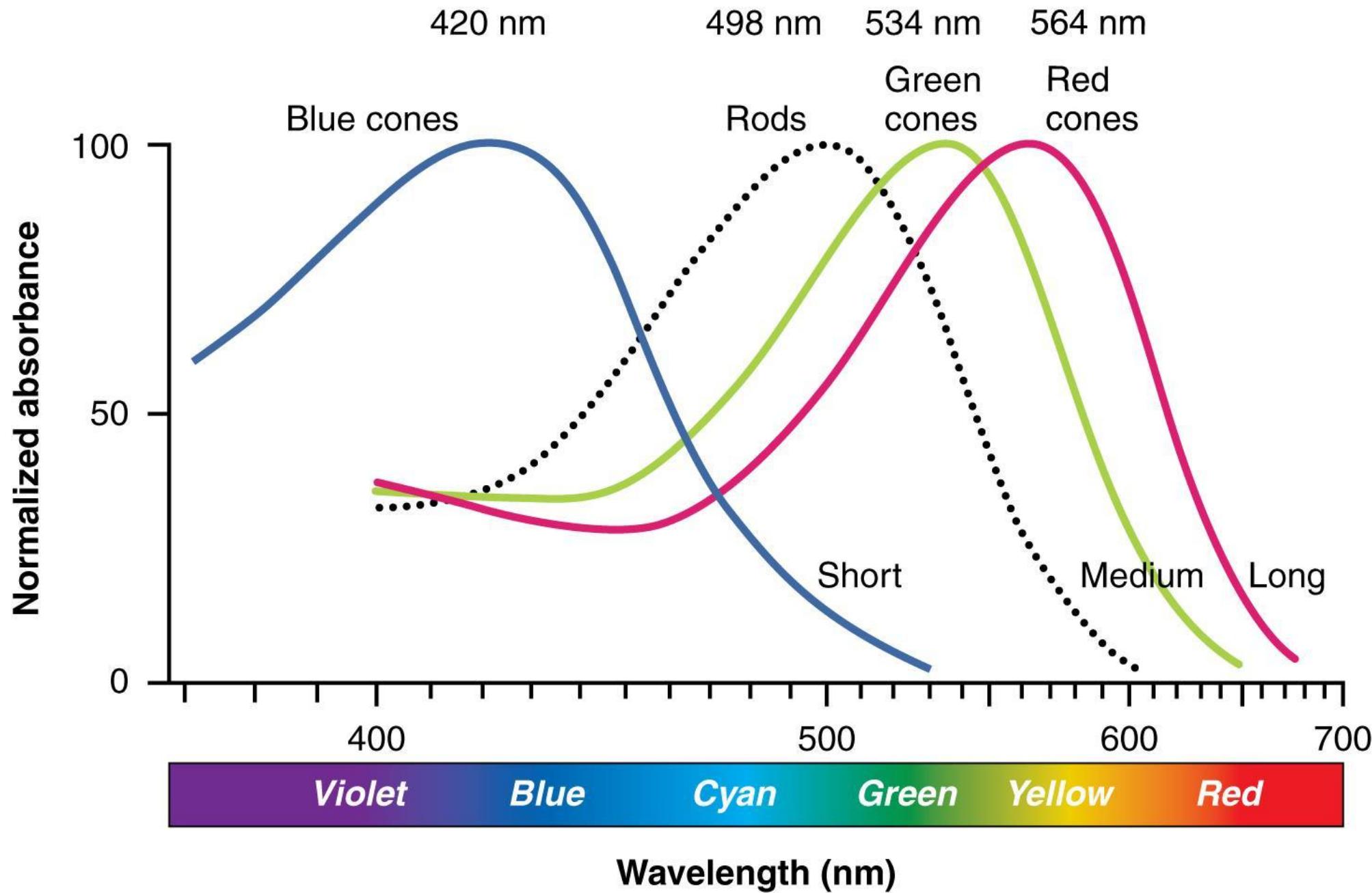
Measures:  $\frac{\text{#Photons}}{\text{Sampling Time}}$

# Exposure Triangle

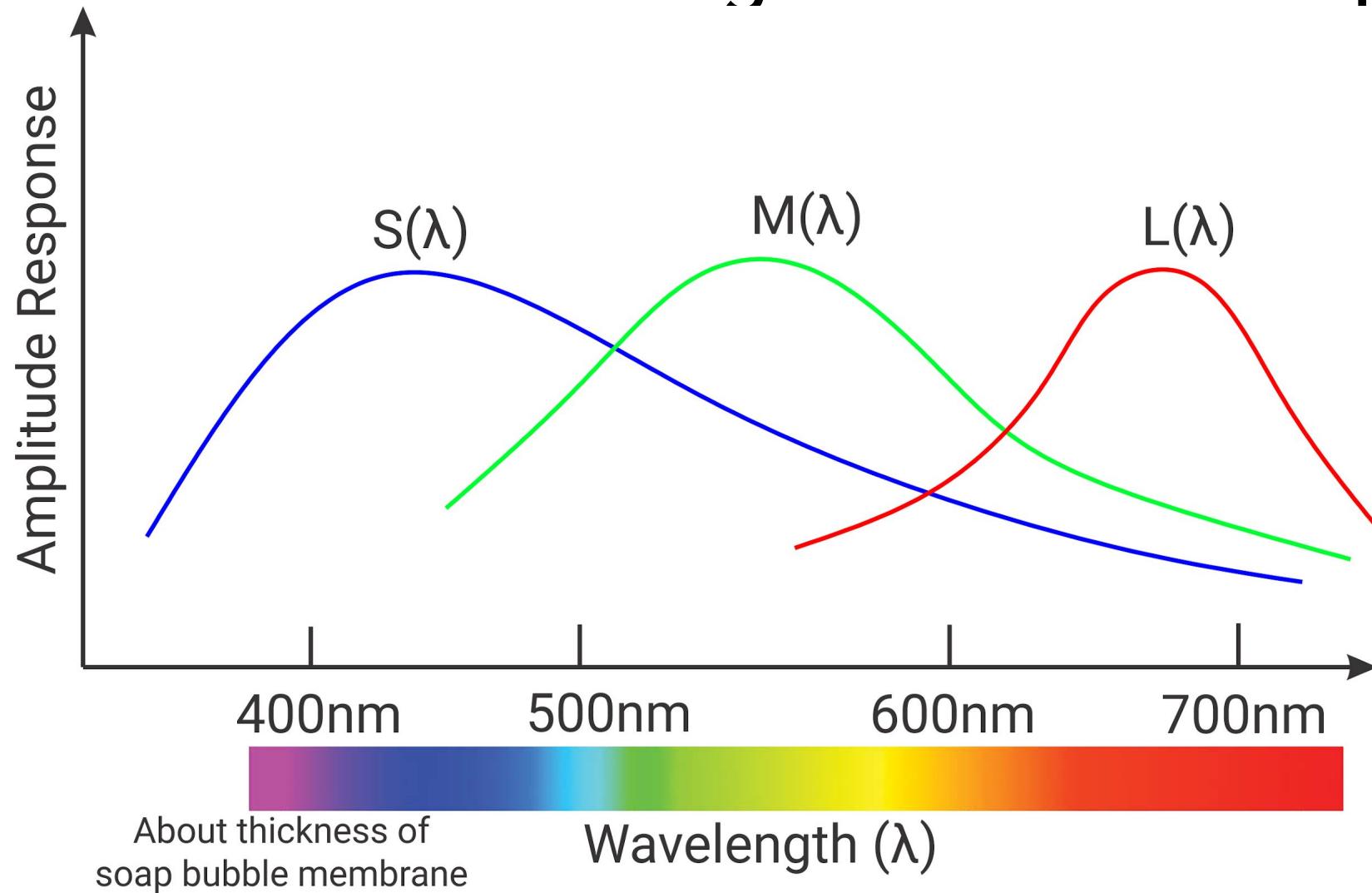


# RGB Color Imaging

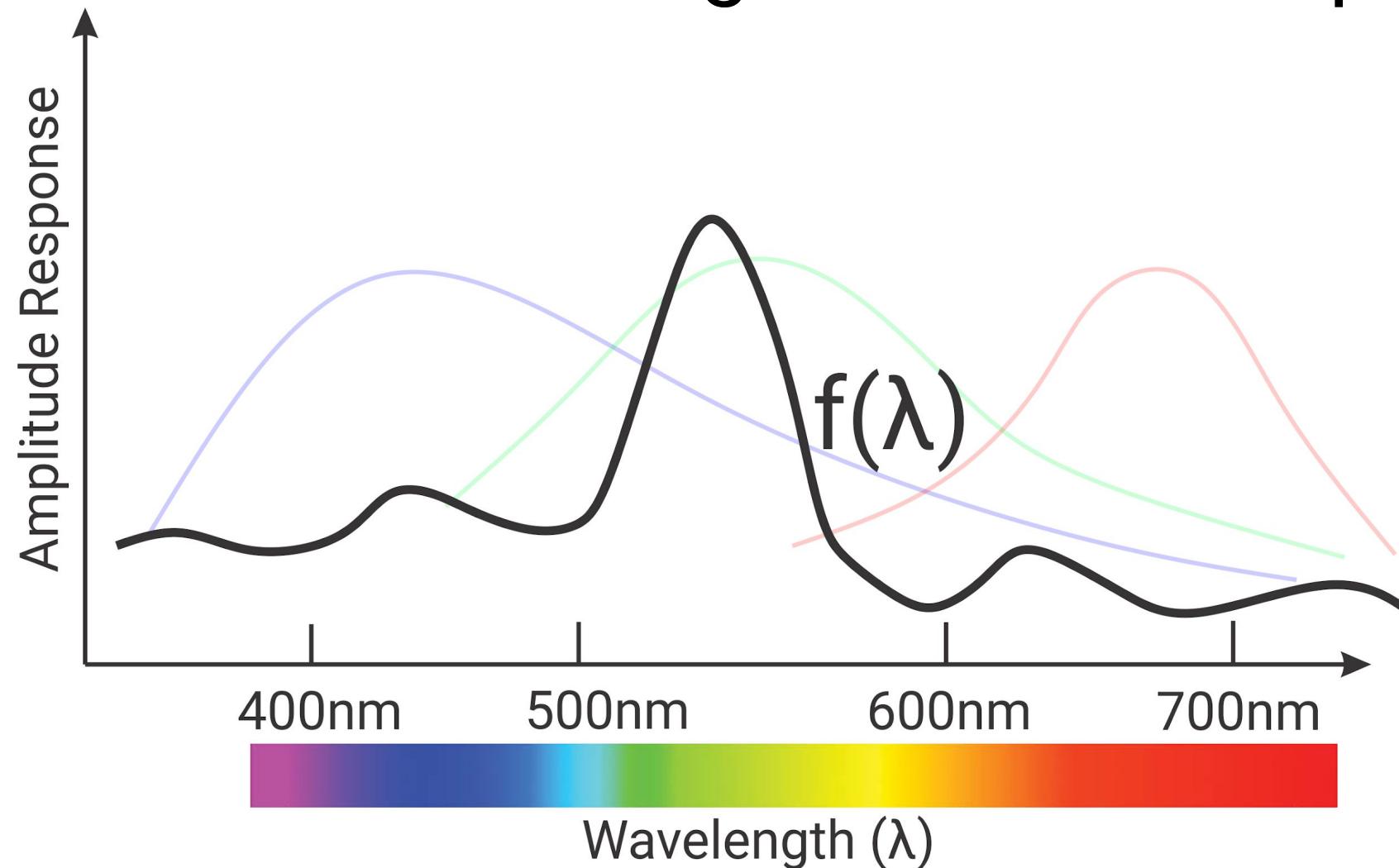
Why **RGB**?



# Mathematical Modelling of Color Perception



# Mathematical Modelling of Color Perception



# Mathematical Modelling of Color Perception

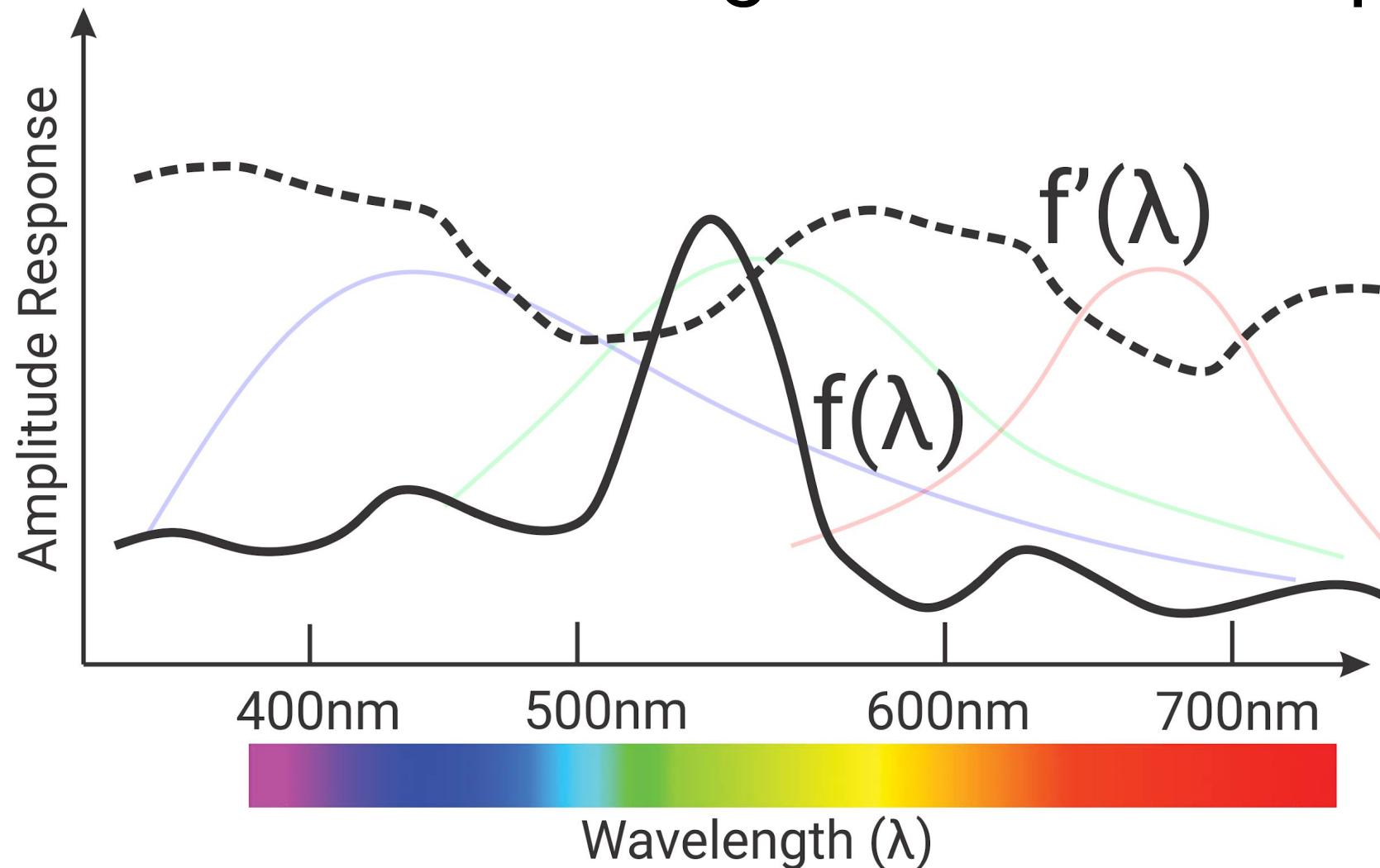
$$S_{res} = \int S(\lambda) f(\lambda) d\lambda$$

$$M_{res} = \int M(\lambda) f(\lambda) d\lambda$$

$$L_{res} = \int L(\lambda) f(\lambda) d\lambda$$

$$\mathbb{R}^\infty \rightarrow \mathbb{R}^3$$

# Mathematical Modelling of Color Perception



# Color Illusion



**ORIGINAL**

(Blue and Gold)  
+0% brightness, +0%  
contrast



**BRIGHTER**

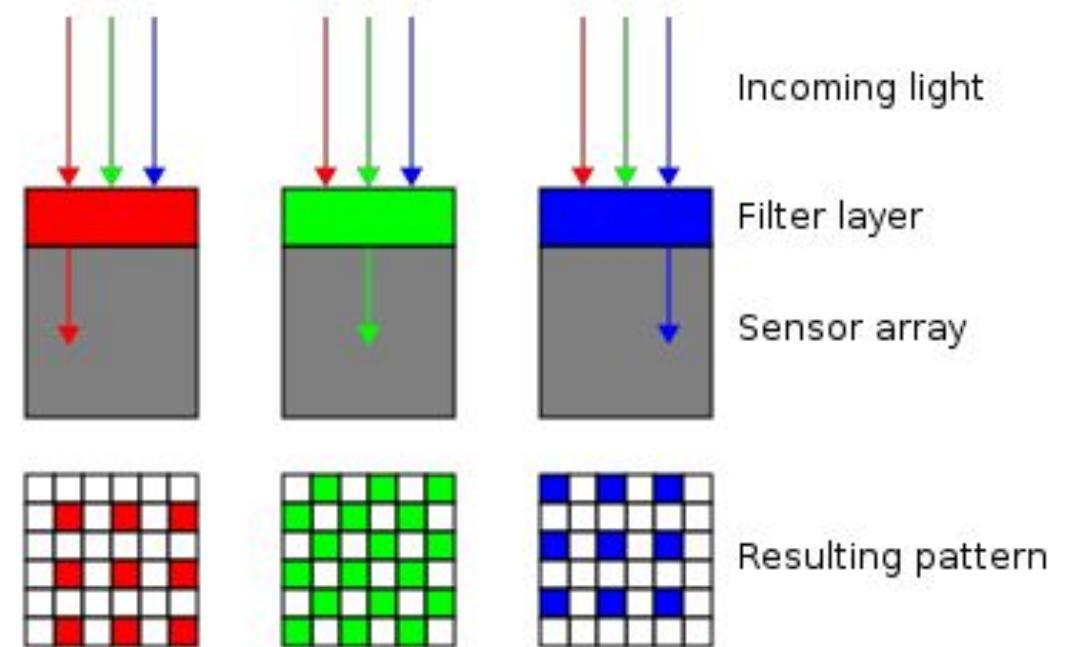
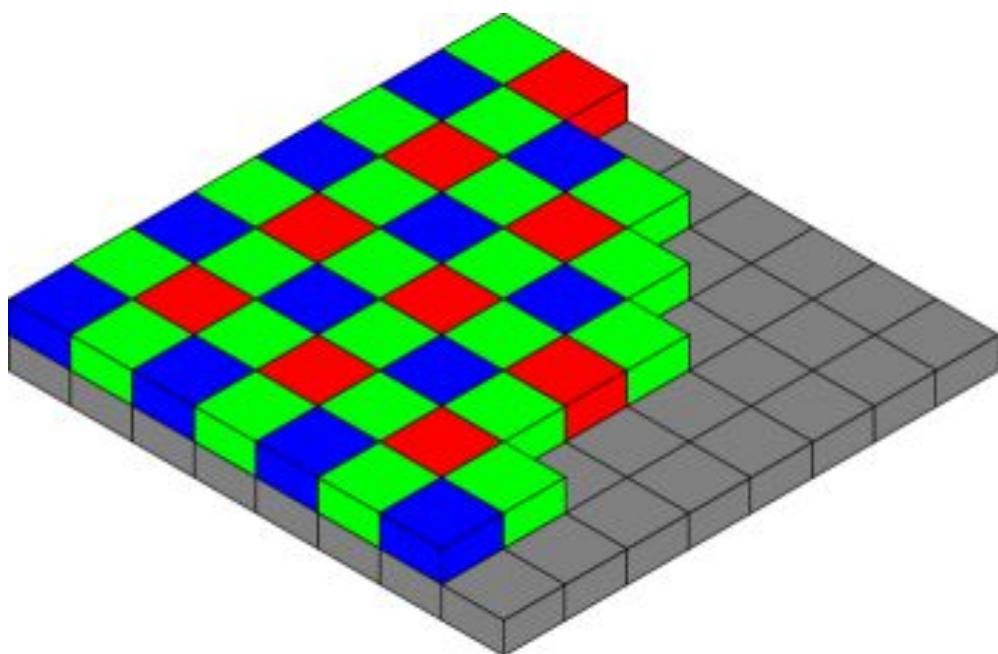
(White and Gold)  
+40% brightness, +40%  
contrast



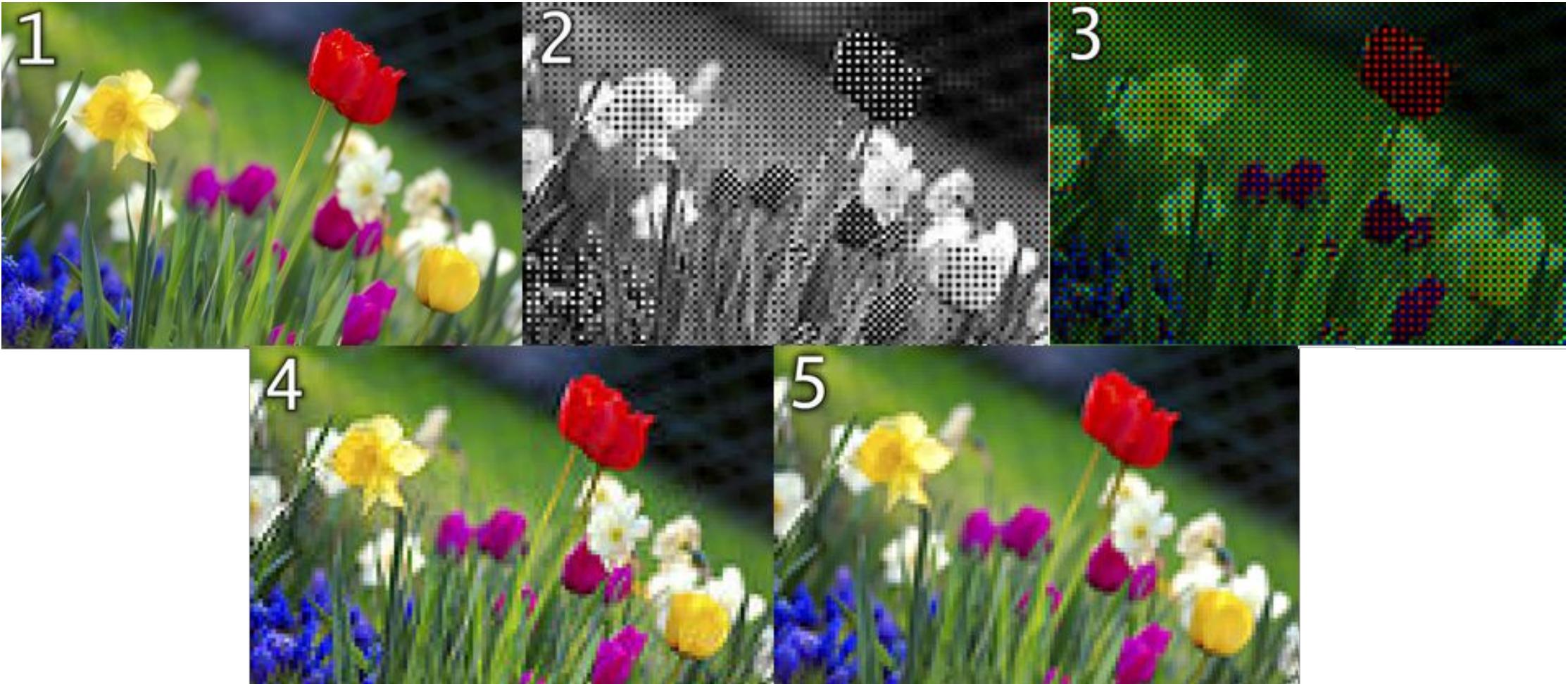
**DARKER**

(Blue and Black)  
-30% brightness, +40%  
contrast

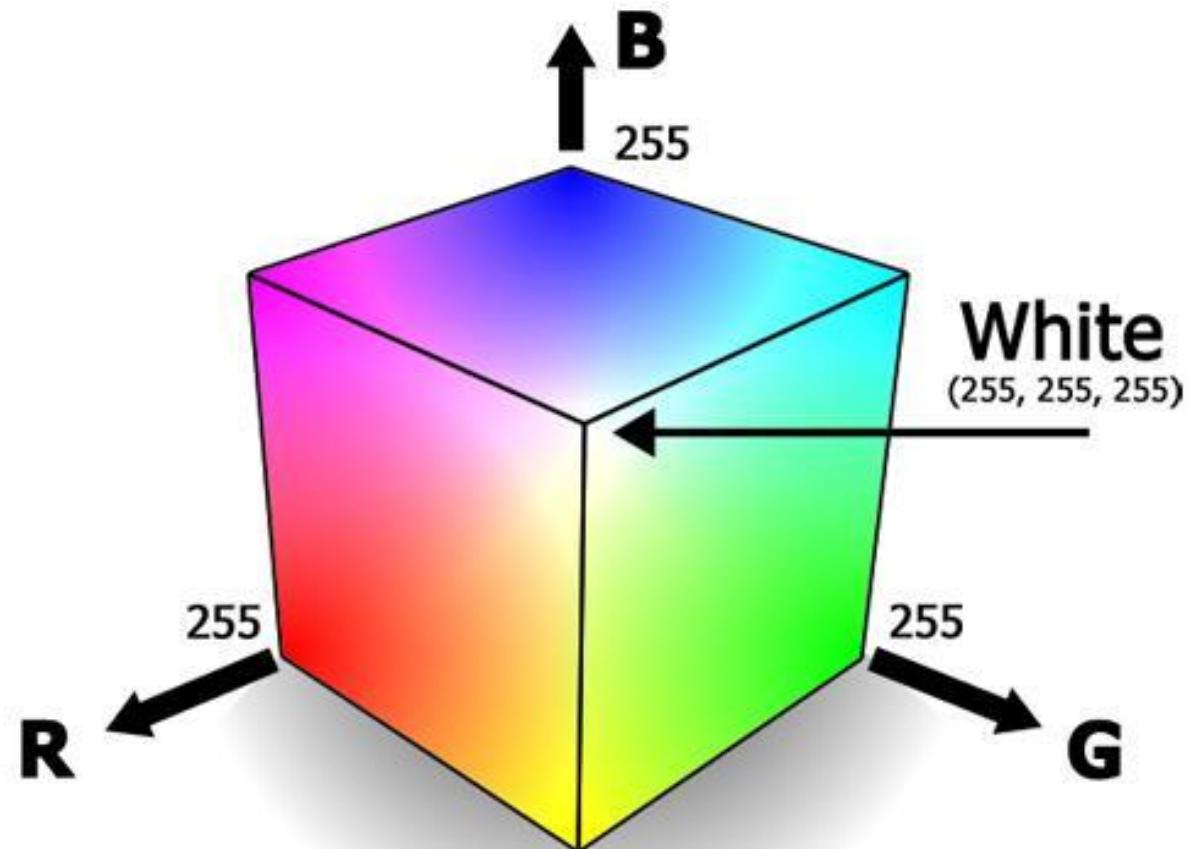
# How is RGB captured?



# Bayer Pattern



# RGB Color Cube



# Different Color Spaces: HSV

$$R' = \frac{R}{255}$$

$$G' = \frac{G}{255}$$

$$B' = \frac{B}{255}$$

$$C_{max} = \max(R', G', B')$$

$$C_{min} = \min(R', G', B')$$

$$\Delta = C_{max} - C_{min}$$

Hue is calculated as follows:

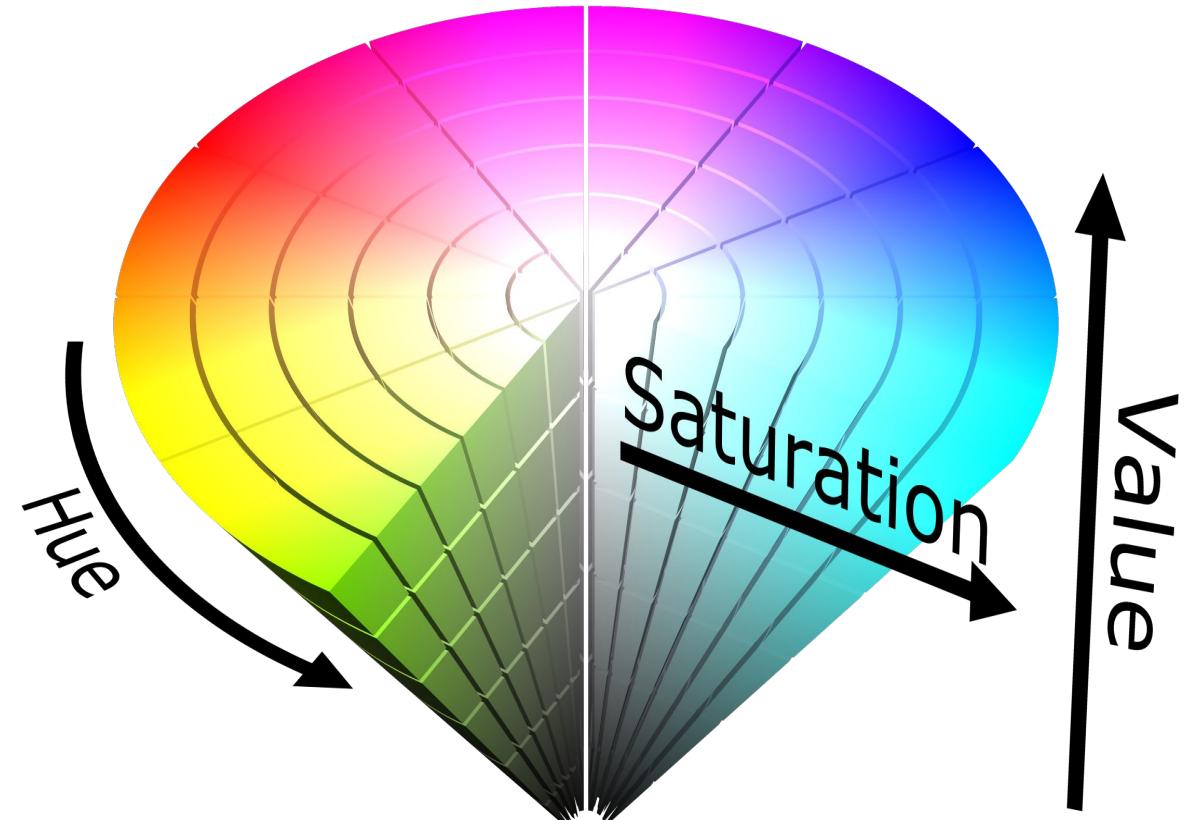
$$\begin{cases} 0^\circ & \text{if } \Delta = 0 \\ 60^\circ \times \frac{G' - B'}{\Delta} & \text{if } C_{max} = G' \\ 60^\circ \times \frac{B' - R'}{\Delta} + 2 & \text{if } C_{max} = B' \\ 60^\circ \times \frac{R' - G'}{\Delta} + 4 & \text{if } C_{max} = R' \end{cases}$$

Saturation is calculated as follows:

$$\begin{cases} 0 & \text{if } C_{max} = 0 \\ \frac{\Delta}{C_{max}} & \text{if } C_{max} \neq 0 \end{cases}$$

Value is calculated as follows:

$$V = C_{max}$$

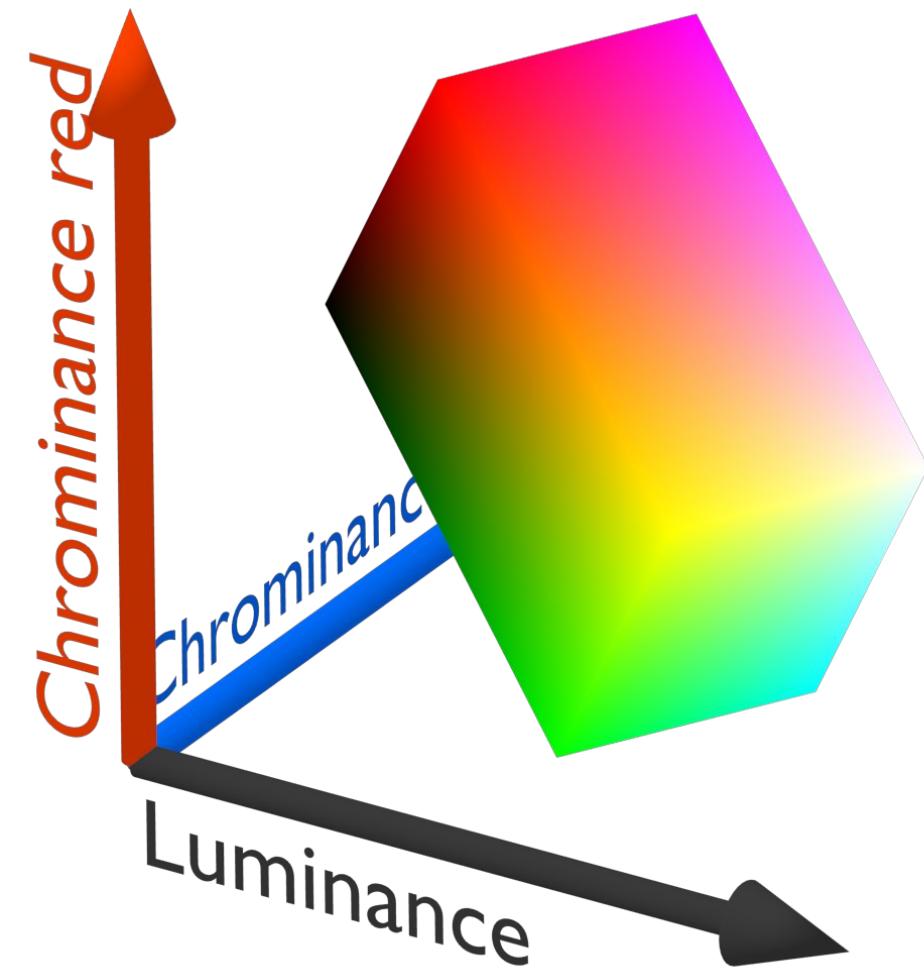


# Different Color Spaces: YCbCr

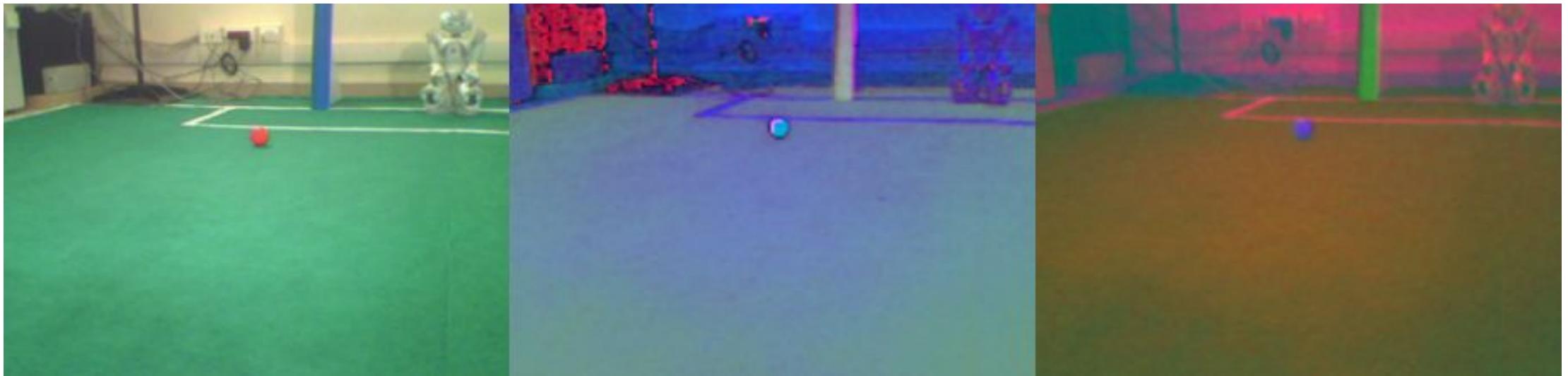
$$Y = 0.299R + 0.587G + 0.114B$$

$$Cb = -0.169R + -0.331G + 0.500B + 128$$

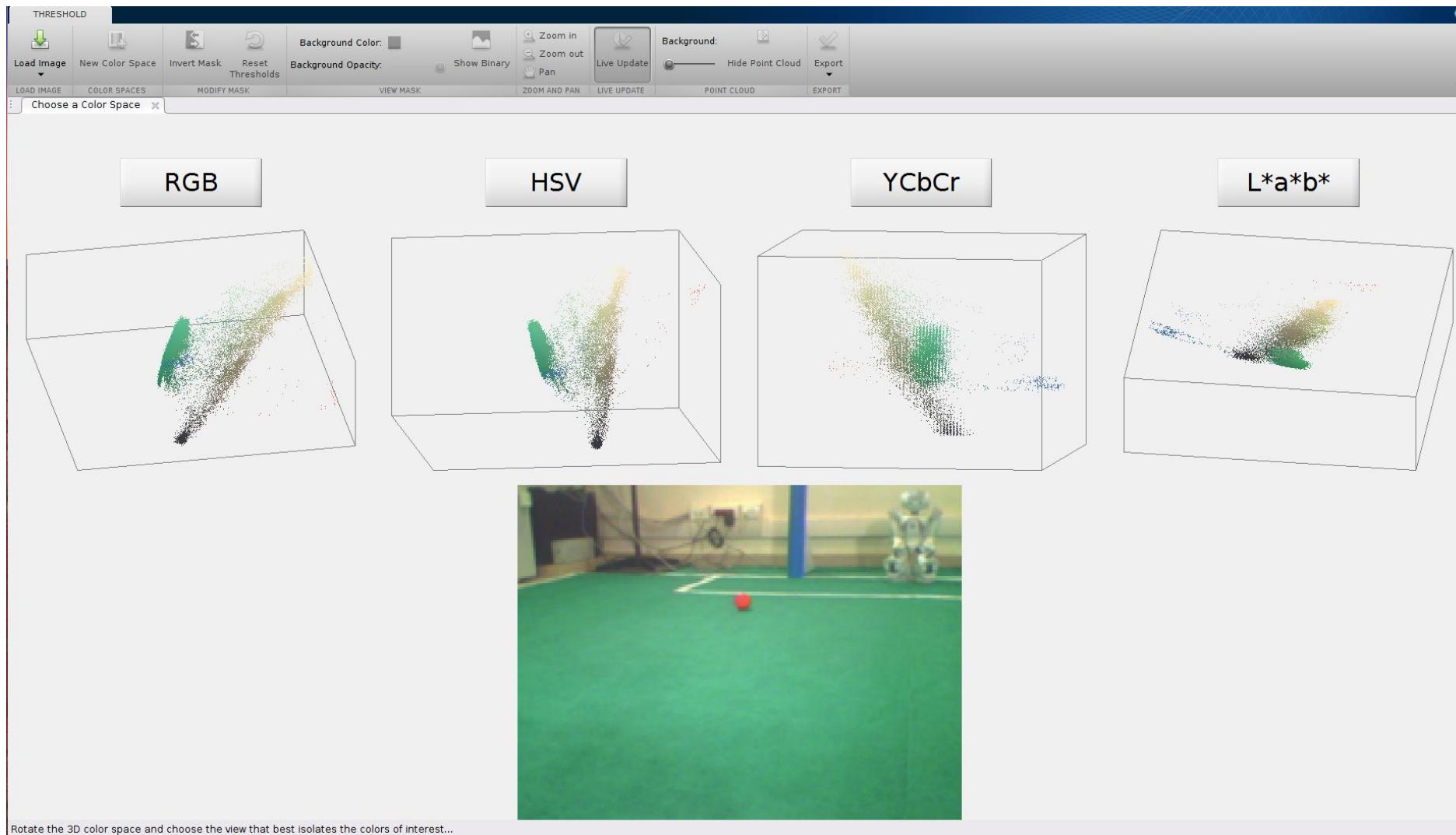
$$Cr = 0.500R + -0.419G + -0.081B + 128$$



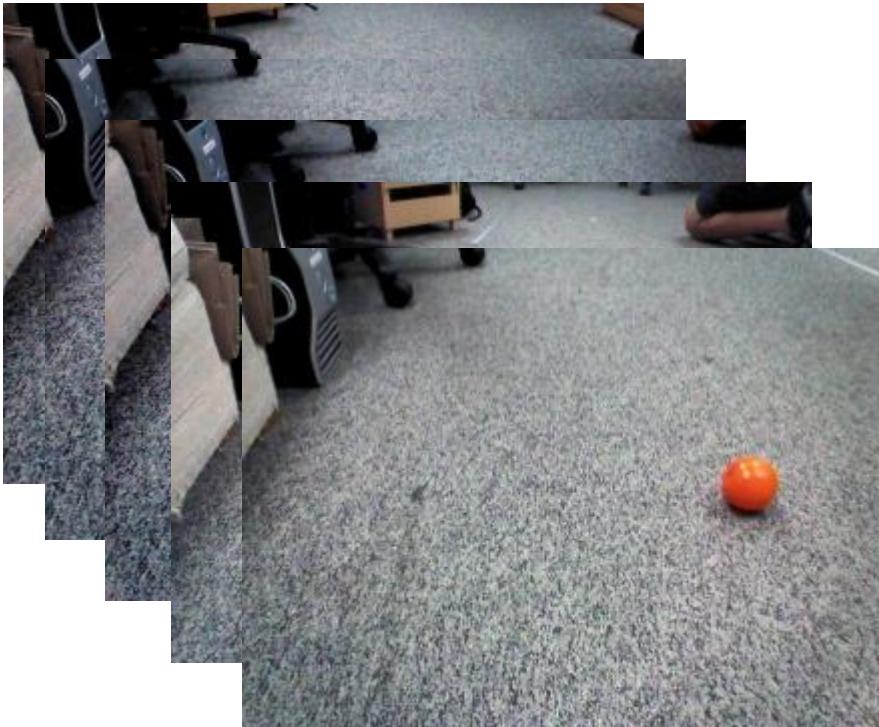
# Robosoccer in RGB, HSV and YCbCr

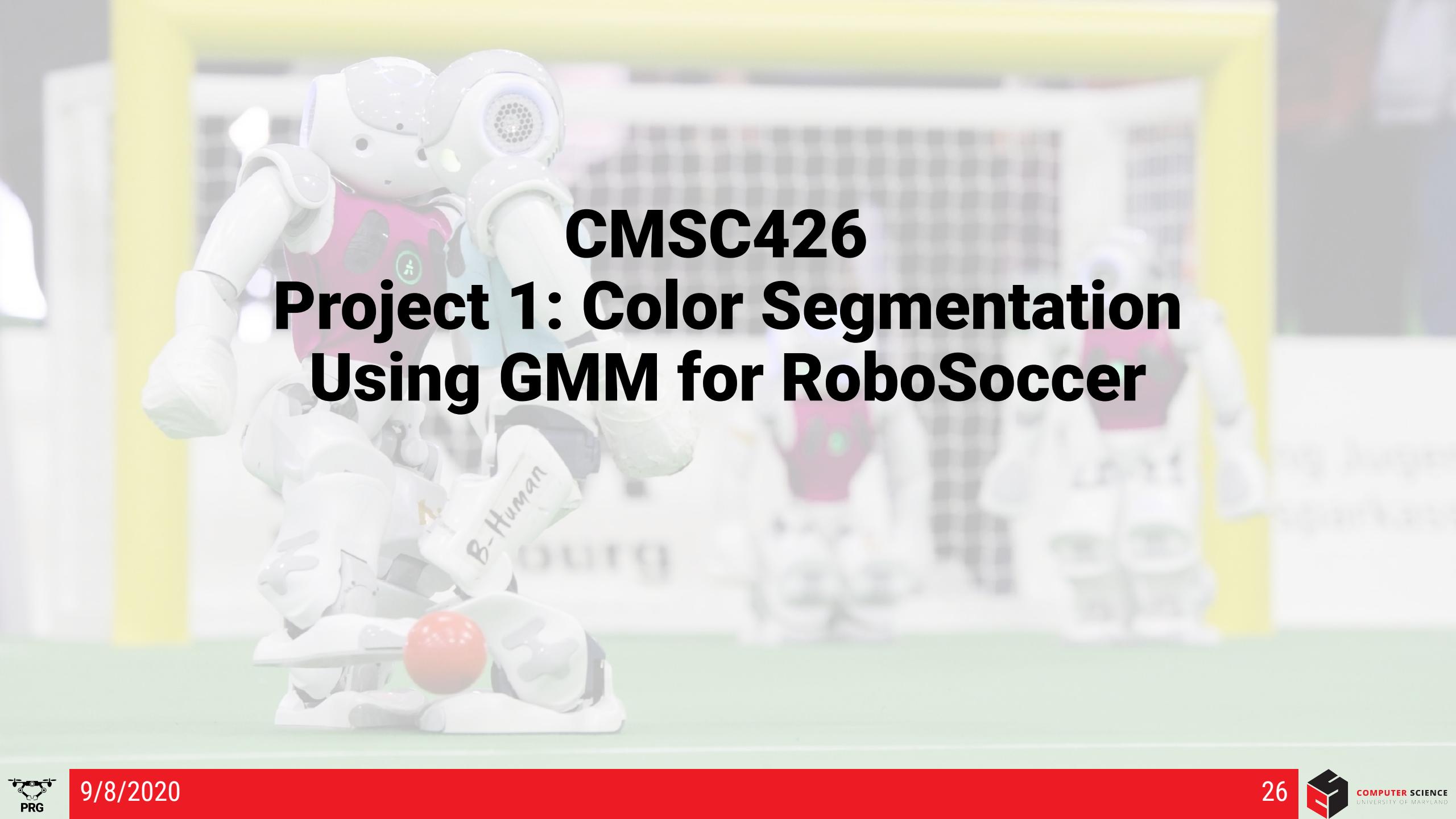


# Mathematical View of Color Thresholding



# Color Segmentation: IO for Training

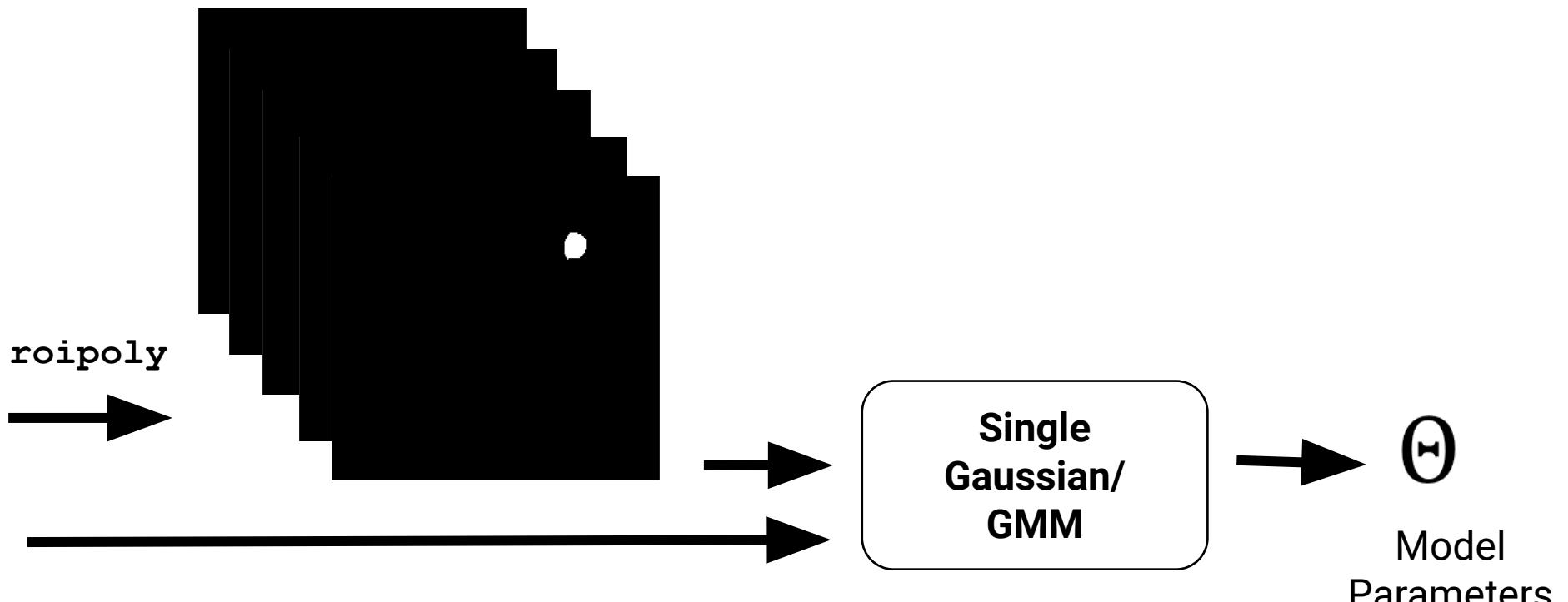


A white humanoid robot with a pink vest and a red ball on a soccer field.

# CMSC426

## Project 1: Color Segmentation Using GMM for RoboSoccer

# Project Pipeline



# Mathematical View of Color Thresholding

$\mathbf{x}$  is a pixel's **RGB** value

$p(\mathbf{x}|C_l)$ : Likelihood

$p(C_l|\mathbf{x})$ : Posterior

How do we get  $p(C_l|\mathbf{x})$ ?

Bayes' Rule!

$$p(C_l|\mathbf{x}) = \frac{p(\mathbf{x}|C_l)p(C_l)}{\sum_{i=1}^L p(\mathbf{x}|C_i)p(C_i)}$$

↑  
Prior

# Color Thresholding Using Single Gaussian

What is function we should use for  $p(C_l)$ ?

Any function of your choice!

Generally Uniform, i.e.,  $p(C_l) = \frac{1}{L}$

One could model more common colors with higher  $p(C_l)$ !

# Color Thresholding Using Single Gaussian

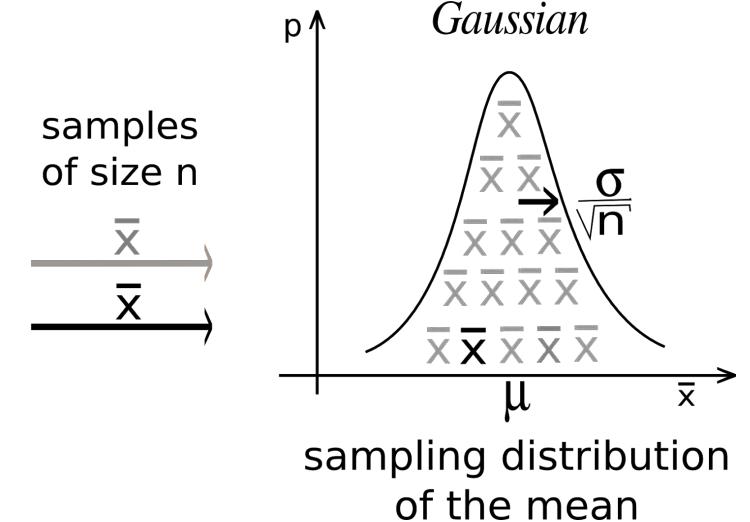
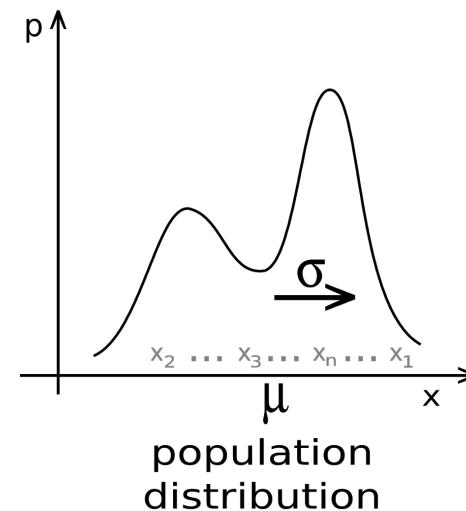
What is function we should use for  $p(\mathbf{x}|C_l)$ ?

$$p(\mathbf{x}|C_l) \sim \mathcal{N}(\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} e^{\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})}$$

or a Normal Distribution

Why Gaussian?

Central Limit Theorem



# Color Thresholding Using Single Gaussian

What do we need to obtain  $\Theta = [\mu, \Sigma]$ ?

$\mathbf{x}$ : Data samples

$$\boldsymbol{\mu} = \frac{1}{N} \mathbf{x} \text{ Empirical Mean}$$

$$\boldsymbol{\Sigma} = \frac{1}{N} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \text{ Empirical Co-variance}$$

# Programming Tips:

$$p(C_l | \mathbf{x}) \propto p(\mathbf{x} | C_l) p(C_l)$$

$$p(C_l) = \frac{1}{L}$$

$$p(\mathbf{x} | C_l) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} e^{\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$$

or a Normal Distribution

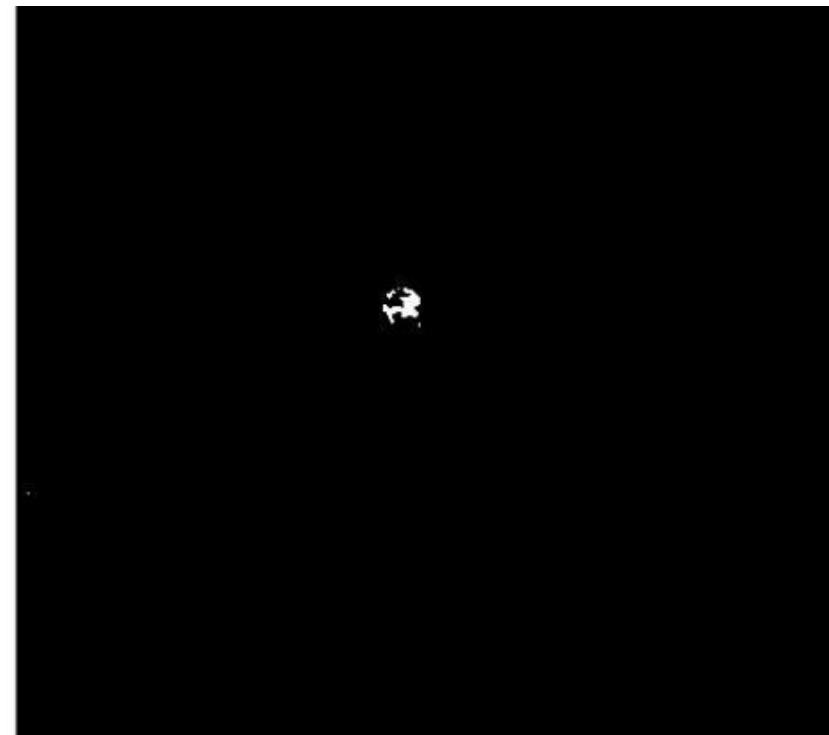
$$\boldsymbol{\mu} = \frac{1}{N} \mathbf{x} \text{ Empirical Mean}$$

$$\boldsymbol{\Sigma} = \frac{1}{N} (\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T \text{ Empirical Co-variance}$$

$\mathbf{x}$  is a pixel's **RGB** value

For Orange Pixel's only

# Sample Output

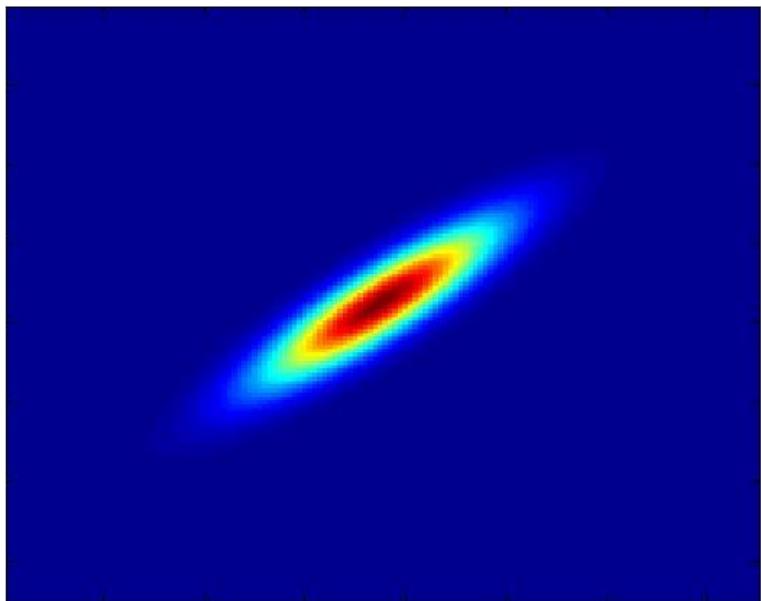


$$p(C_l|x) \geq \tau$$

# $N$ -D Gaussian

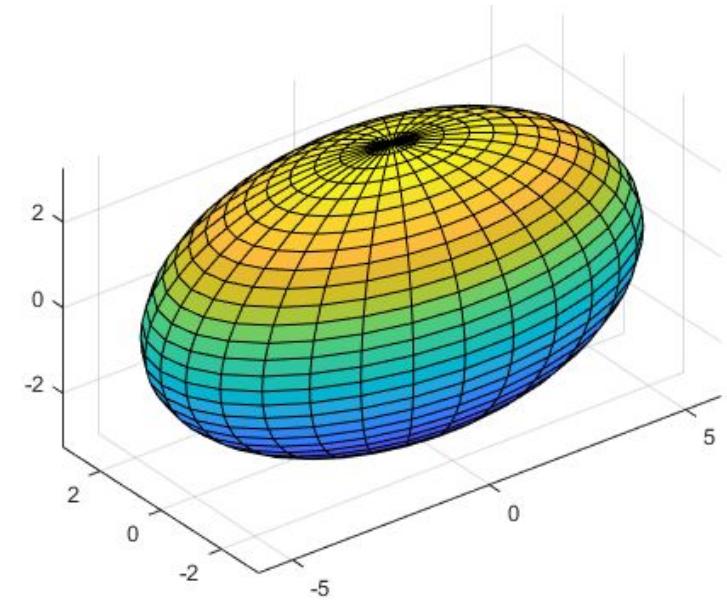
What is the shape of a 2D Gaussian?

Ellipse

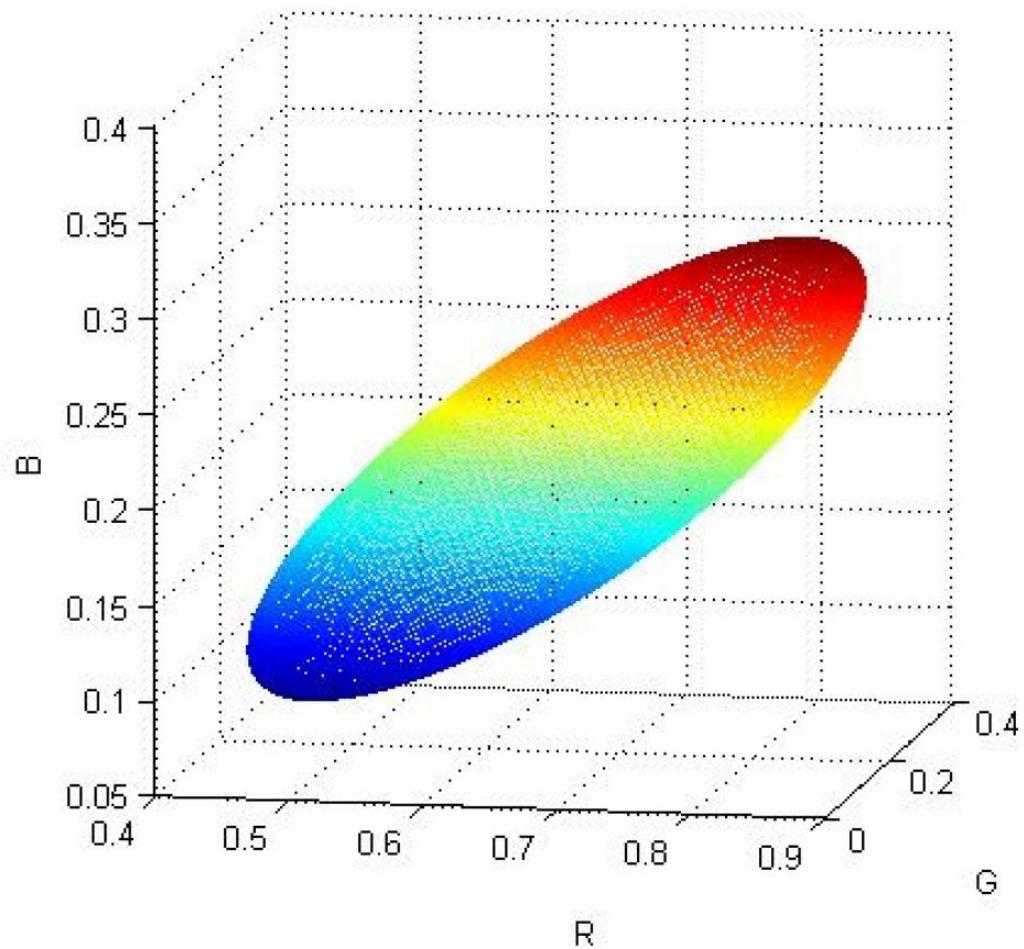
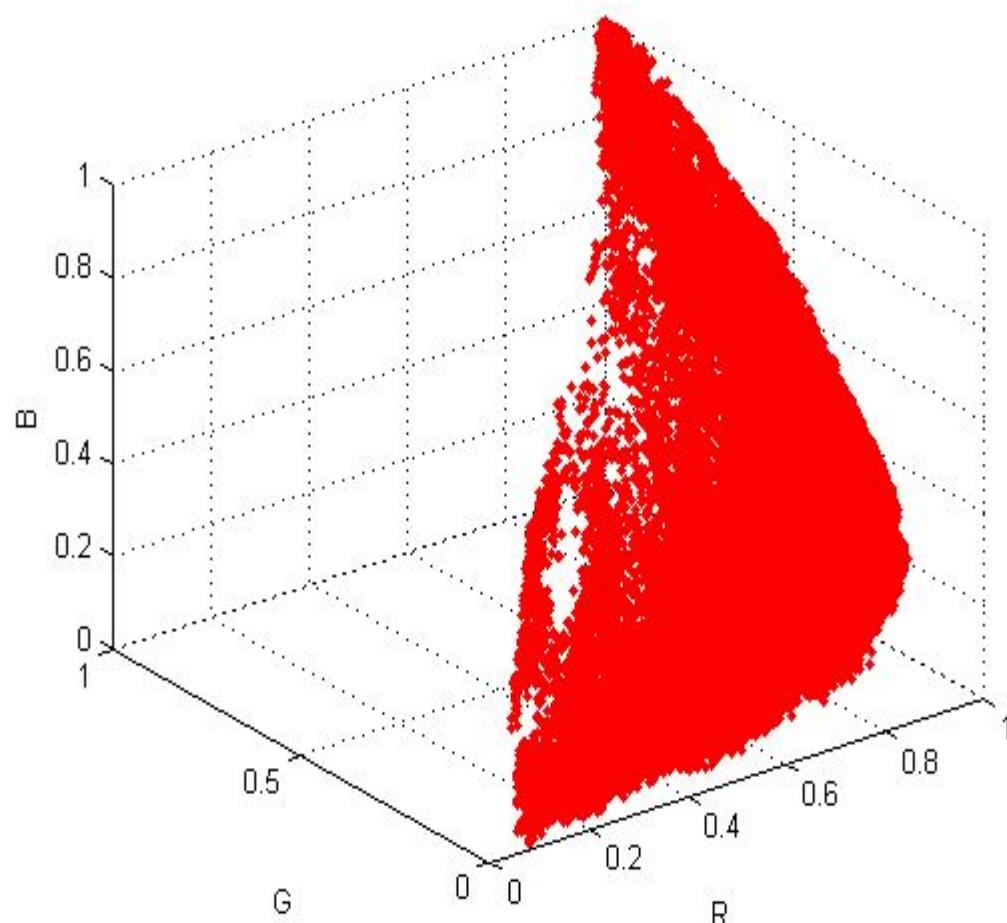


What is the shape of a 3D Gaussian?

Ellipsoid

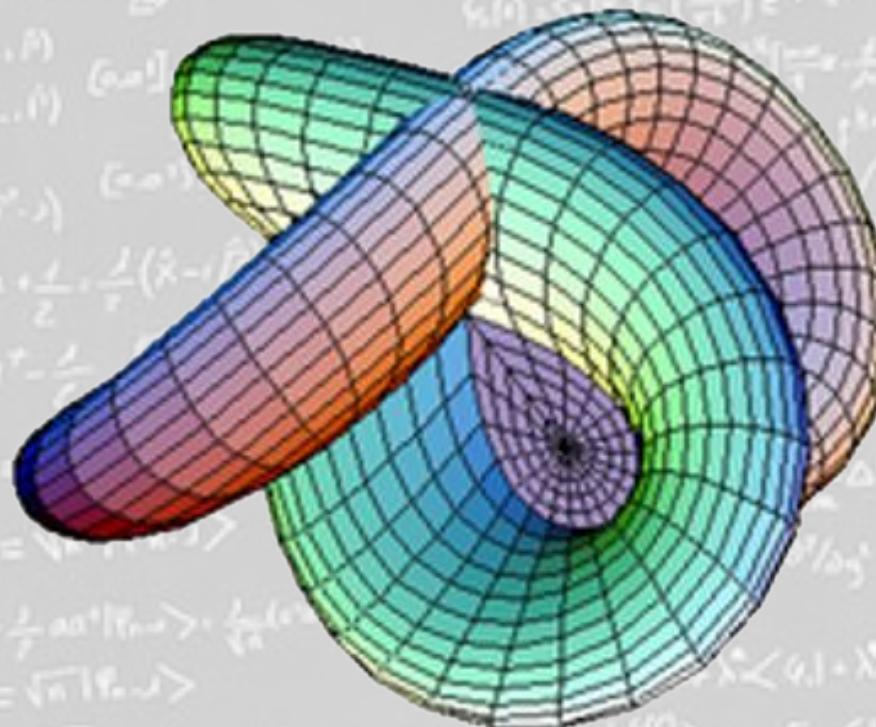


# Problems with Single Gaussian



# Solution?

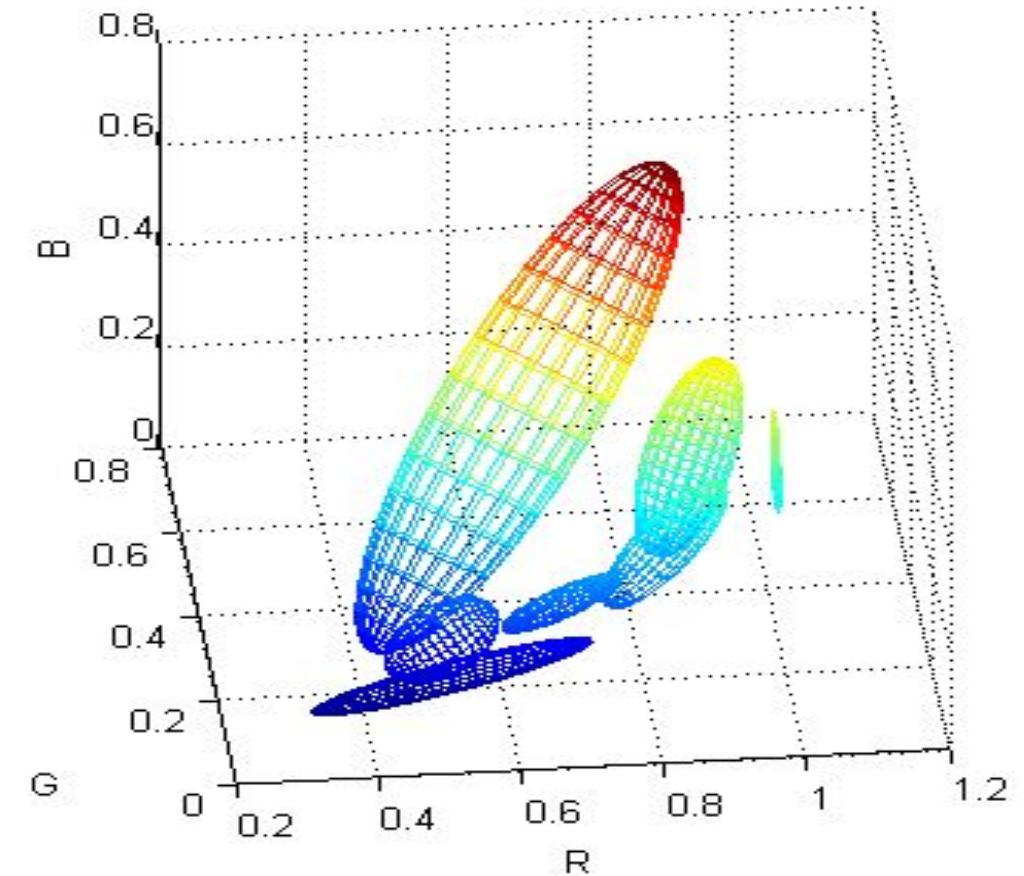
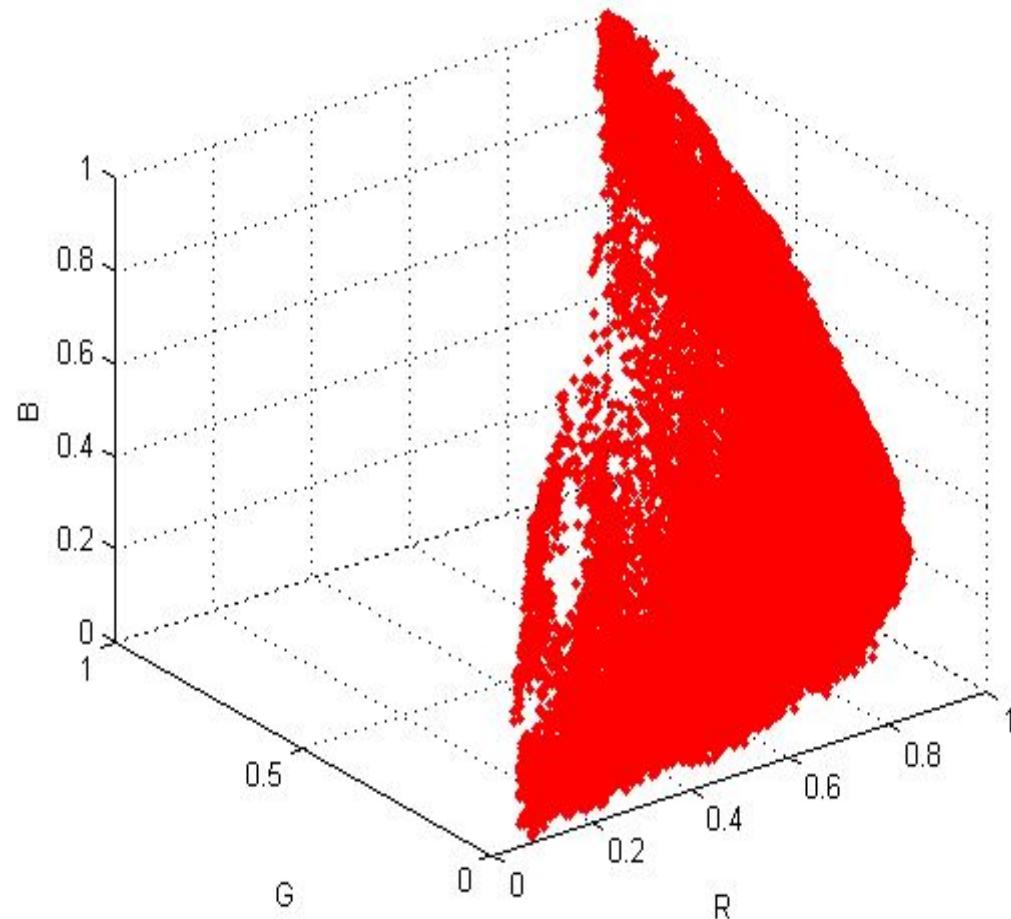
Fancy function/Manifold



Too Complicated!

Use sum of known function as an efficient and easy approximation!

# Gaussian Mixture Models to the rescue!



# Math Behind Awesome GMMs

$$p(\mathbf{x}|C_l) \sim \sum_{i=1}^K \pi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$$

Given  $\mathbf{x}$ , we want to find  $\Theta = [\pi_i, \boldsymbol{\mu}_i, \Sigma_i] \forall i \in [1, K]$

More formally, we want to maximize the correctness of  $p(C_l|\mathbf{x})$

$$\operatorname{argmax}_{\Theta} \sum_{i=1}^N \log p(C_l|\mathbf{x})$$

Maximum Log Likelihood

# Expectation Maximization for GMM

**Initialization:**

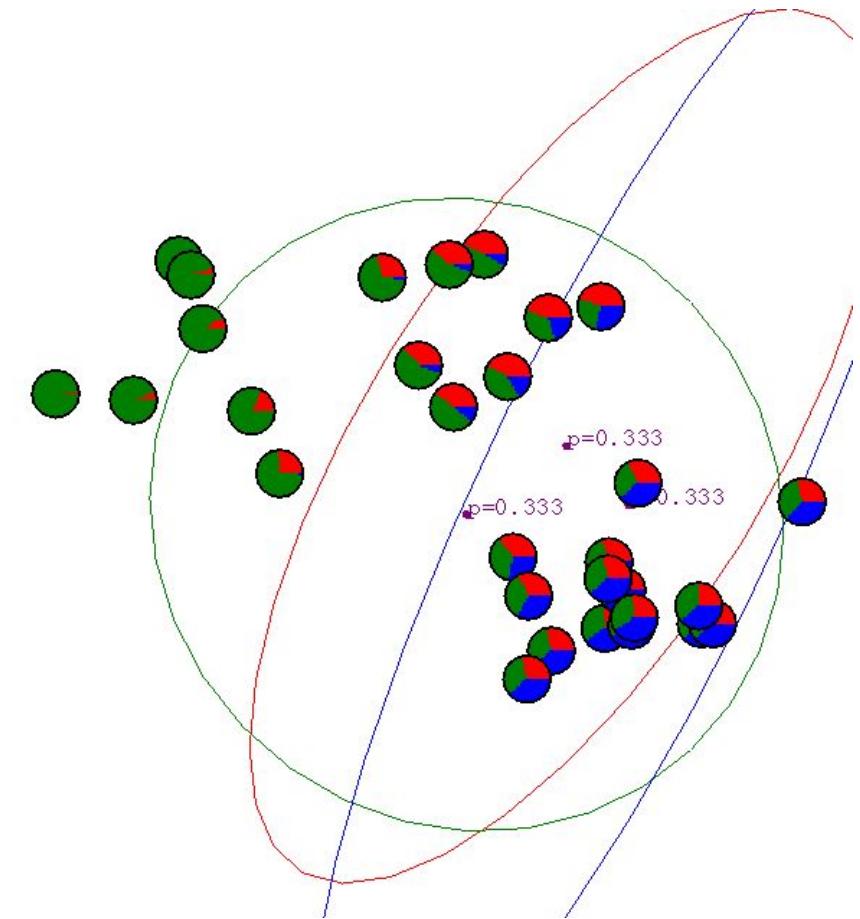
Randomly choose  $\pi_i, \mu_i, \Sigma_i \forall i \in [1, K]$

**E-Step:**

Evaluate Model/ Assign points to model

$$\text{Cluster Weight } \alpha_{i,j} = \frac{\pi_j p(\mathbf{x}_i | C_j)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_i | C_j)}$$

$i$  is the data point index,  $j$  is the cluster index



# Expectation Maximization for GMM

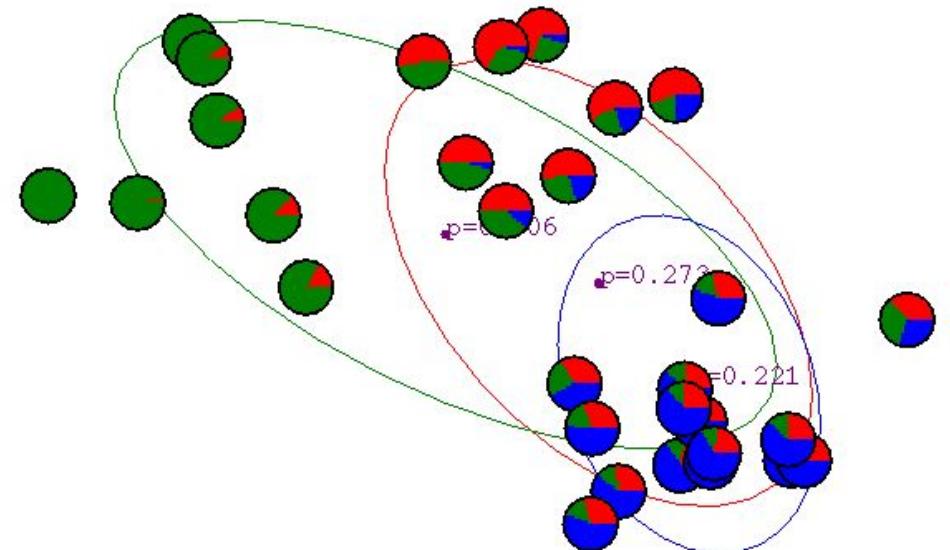
M-Step:

Evaluate best model to fit points

$$\mu_j = \frac{\sum_{i=1}^N \alpha_{i,j} \mathbf{x}_i}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\Sigma_j = \frac{\sum_{i=1}^N \alpha_{i,j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\pi_j = \frac{1}{N} \sum_{i=1}^N \alpha_{i,j}$$



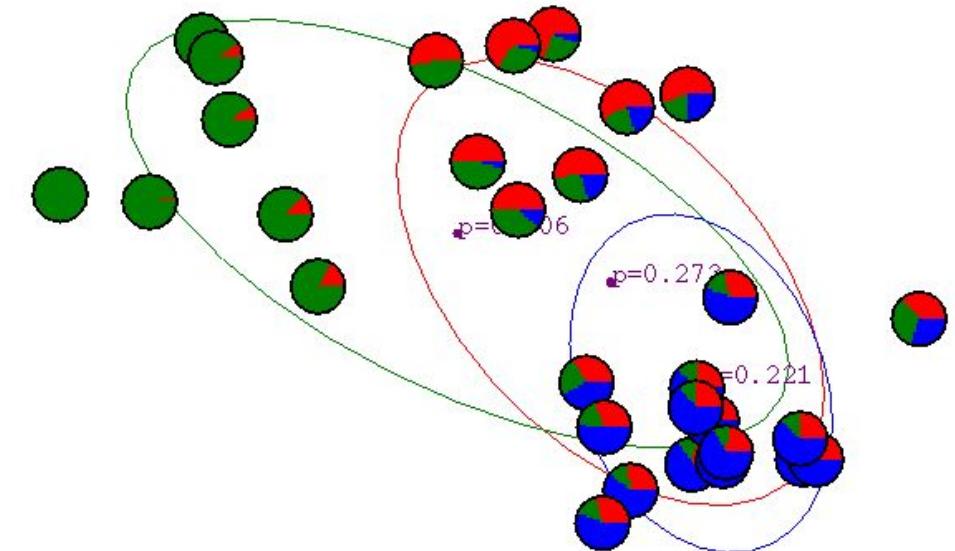
# Expectation Maximization for GMM

E-Step:

Evaluate Model/ Assign points to model

$$\text{Cluster Weight } \alpha_{i,j} = \frac{\pi_j p(\mathbf{x}_i | C_j)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_i | C_j)}$$

$i$  is the data point index,  $j$  is the cluster index



# Expectation Maximization for GMM

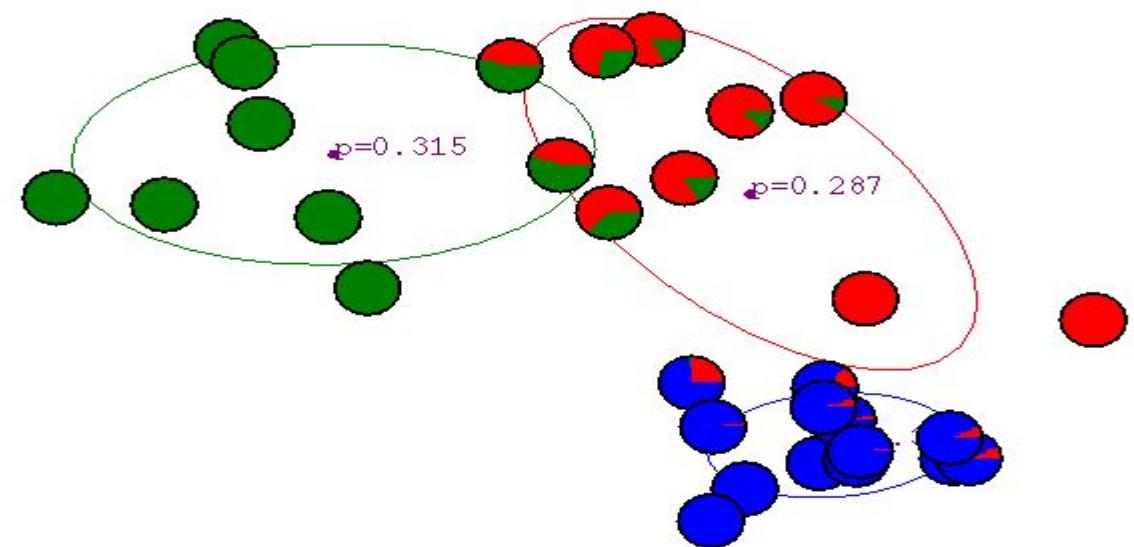
M-Step:

Evaluate best model to fit points

$$\mu_j = \frac{\sum_{i=1}^N \alpha_{i,j} \mathbf{x}_i}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\Sigma_j = \frac{\sum_{i=1}^N \alpha_{i,j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\pi_j = \frac{1}{N} \sum_{i=1}^N \alpha_{i,j}$$

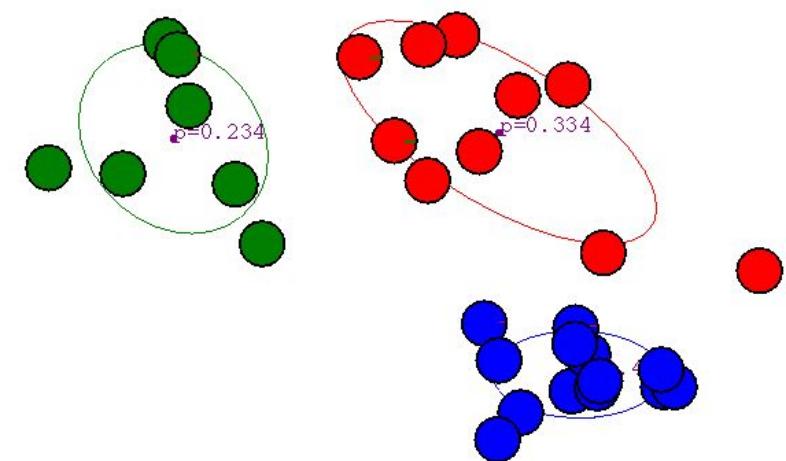


# Expectation Maximization for GMM

Repeat E and M steps until convergence

Convergence measure:

$$\|\mu^t - \mu^{t-1}\|$$



# Programming tips(Pseudocode)

---

## Algorithm 1 GMM

---

```
1: procedure GMM
2:   set  $\tau$                                 ▷ threshold for clustering.
3:   set  $K$                                 ▷ number of gaussians
4:   if training then
5:     load train_images
6:     trainGMM( $K$ )
7:   else if then
8:     load test_images
9:     cluster = testGMM(Model,  $\tau$ )
10:    end if
11:     $d$  = measureDepth(cluster)
12:    plotGMM(Model)                  ▷ visualize GMM ellipsoids
13:    return cluster,  $d$                 ▷ cluster and cluster depth
14:
15: end procedure
```

---

# Training GMM (Only for Orange Pixels)

Randomly choose  $\pi_i, \mu_i, \Sigma_i \forall i \in [1, K]$

Evaluate Model/ Assign points to model

$$\text{Cluster Weight } \alpha_{i,j} = \frac{\pi_j p(\mathbf{x}_i | c_j)}{\sum_{j=1}^K \pi_j p(\mathbf{x}_i | c_j)}$$

$i$  is the data point index,  $j$  is the cluster index

## Algorithm 2 trainGMM

```
1: procedure TRAINGMM( $K$ )
2:   set  $\epsilon$ 
3:   Initialize Model            $\triangleright$  initialize scaling factor, gaussian mean and covariance
4:   while  $i \leq max\_iters$  and  $abs(Mean - prevMean) > \epsilon$  do
5:     expectation_step
6:     maximization_step
7:     increment  $i$ ;
8:   end while
9:   save Model                 $\triangleright$  save scaling factor, gaussian mean and covariance
10:  end procedure
```

$$\mu_j = \frac{\sum_{i=1}^N \alpha_{i,j} \mathbf{x}_i}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\Sigma_j = \frac{\sum_{i=1}^N \alpha_{i,j} (\mathbf{x}_i - \mu_j)(\mathbf{x}_i - \mu_j)^T}{\sum_{i=1}^N \alpha_{i,j}}$$

$$\pi_j = \frac{1}{N} \sum_{i=1}^N \alpha_{i,j}$$

---

**Algorithm 3** testGMM

```
1: procedure TESTGMM(Model,  $\tau$ )
2:   load Model                                 $\triangleright$  load scaling factor, gaussian mean and covariance
3:   computePosterior(Model)
4:   getCluster( $\tau$ )                                $\triangleright$  use thresholding to get the orange ball
5: end procedure
```

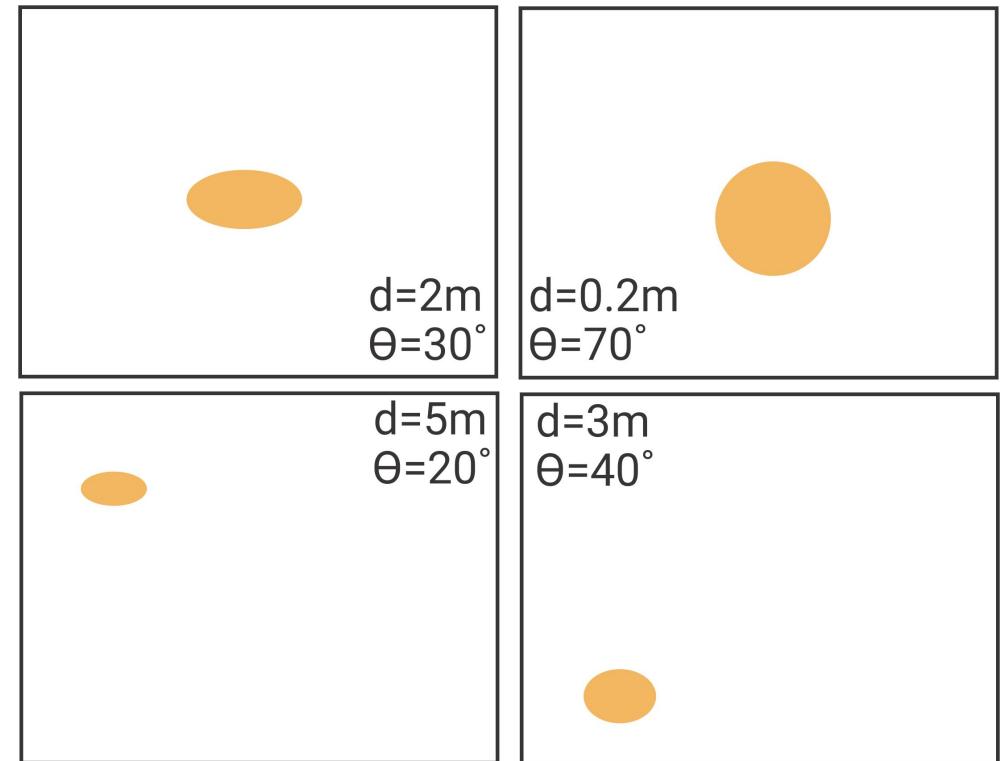
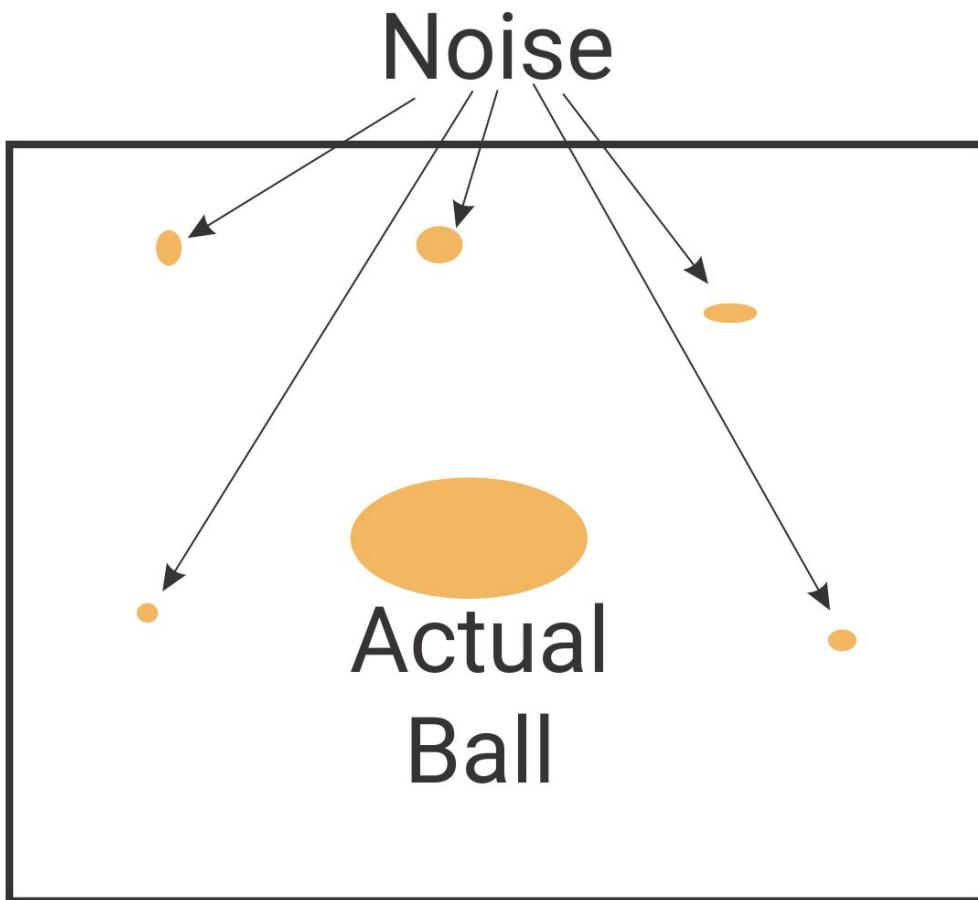
---

$$p(C_l|\mathbf{x}) \propto p(\mathbf{x}|C_l)p(C_l)$$
$$p(C_l) = \frac{1}{L}$$

$$\forall \mathbf{x} \quad p(\mathbf{x}|C_l) = \sum_{i=1}^K \pi_i \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_i, \Sigma_i)$$

**x** is a pixel's **RGB** value

# Estimating Distance To The Ball



`bwmorph, regionprops, fit`

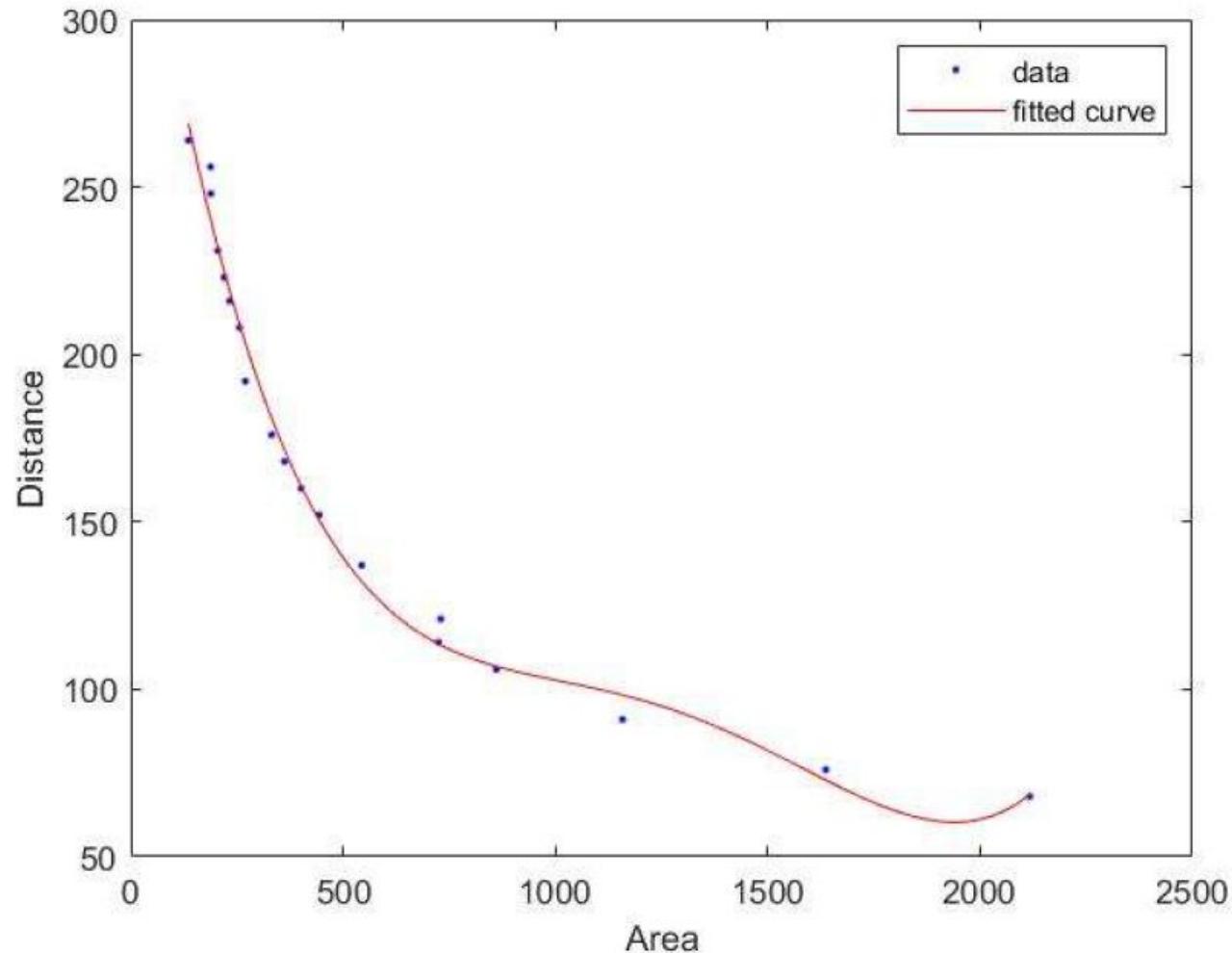


Figure: Distance vs Area Curve

# Train and Test Set

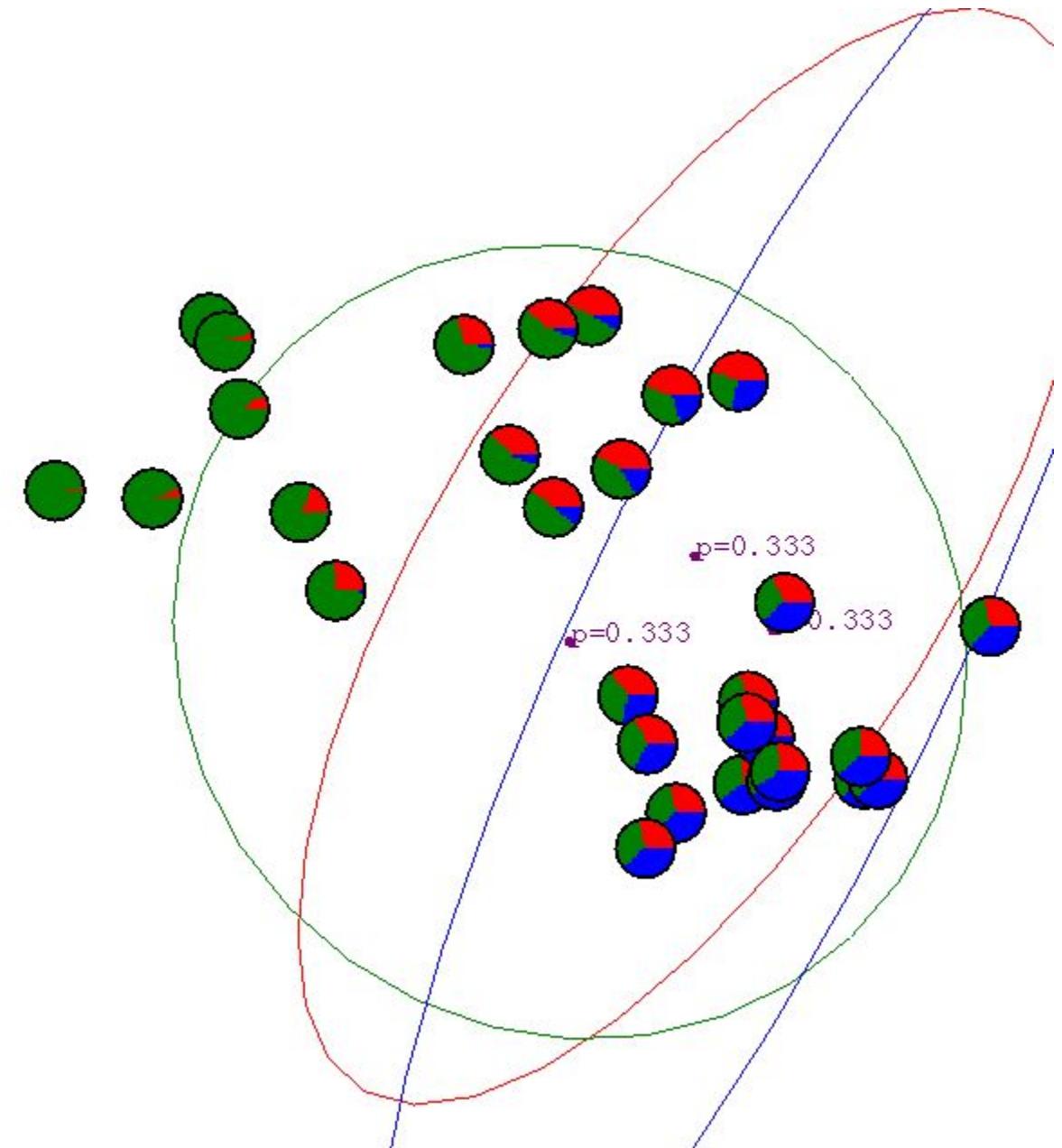
- Few images released with project assignment: Train set.  
Use these images to train and validate your algorithm.
- Since the testing data hasn't been released to you yet, you should do a 80/20 split of your training data itself, and leave the 20% for testing/validating. It's not a good idea to use the same data for training and testing, since that generally leads to overfitting.
- 24hrs before the deadline, new images will be released.  
Only predict on these images, **DO NOT** train/ fit model on these images!

# Improving Robustness

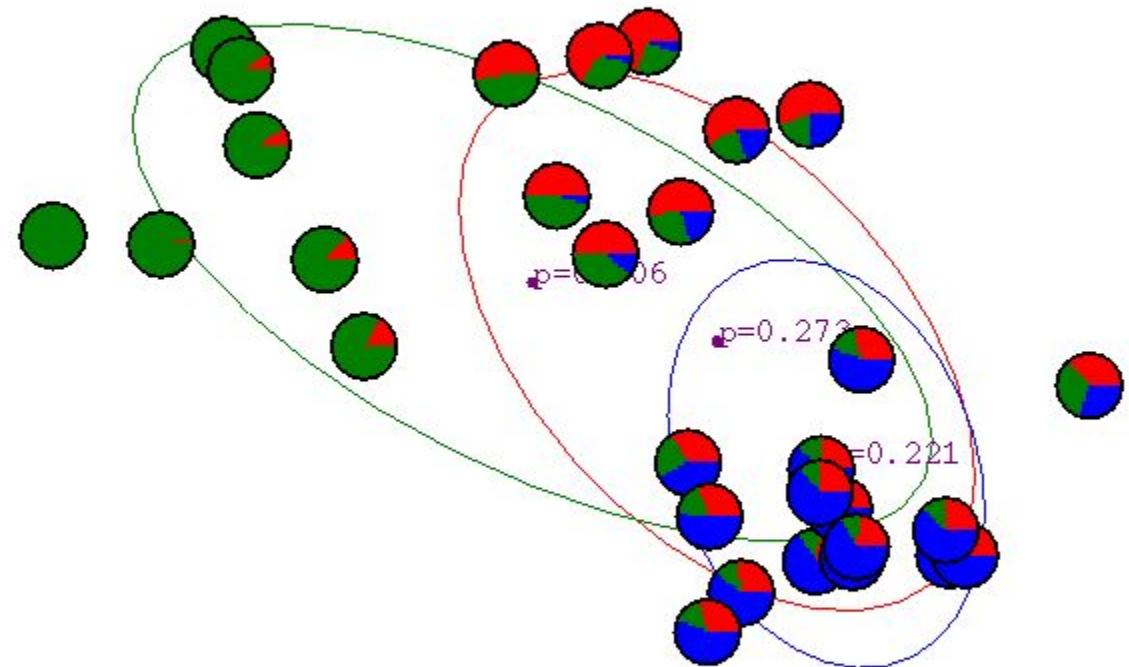
- Think of alternative colorspaces and custom colorspaces
- Hough circles to find circular blobs
- Think of methods to combine stray pixels, i.e., ball split into smaller parts/blobs
- Augment Training set to account for noise and illumination changes
- Using 3D geometry to estimate distance

**Good Luck!**

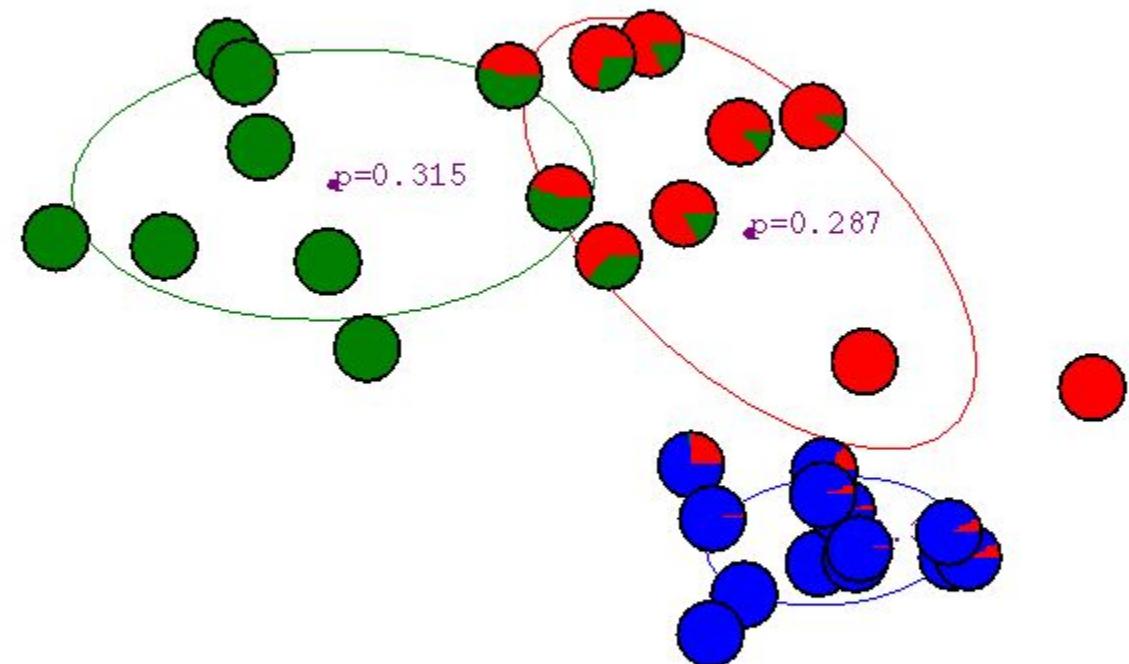
# GMM Intuition



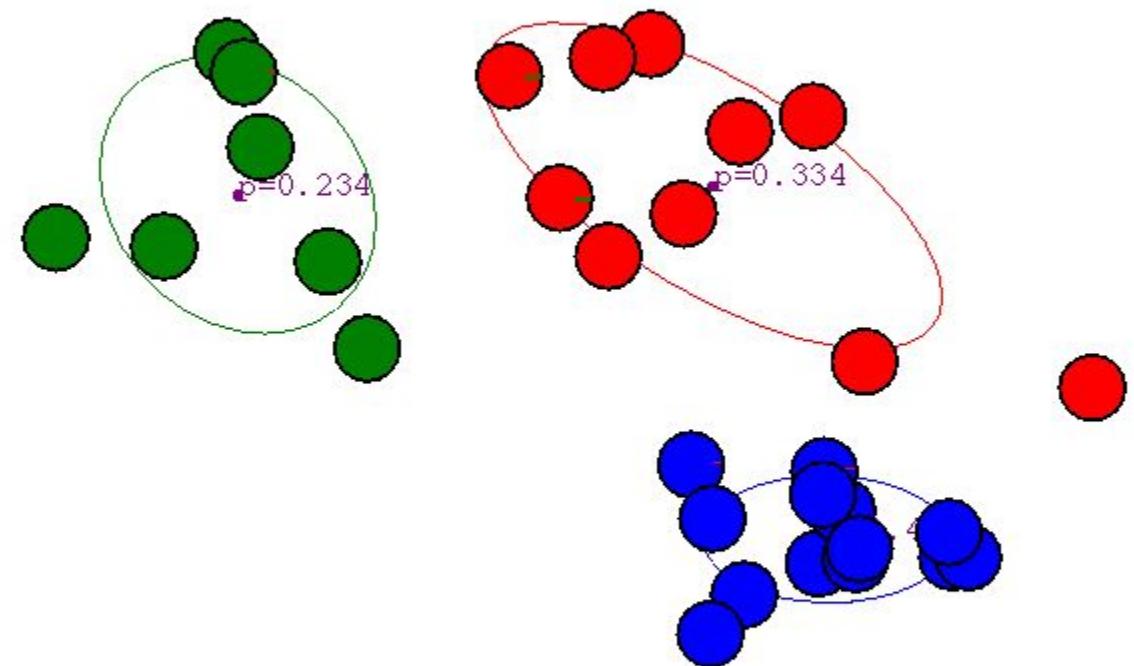
# GMM Intuition



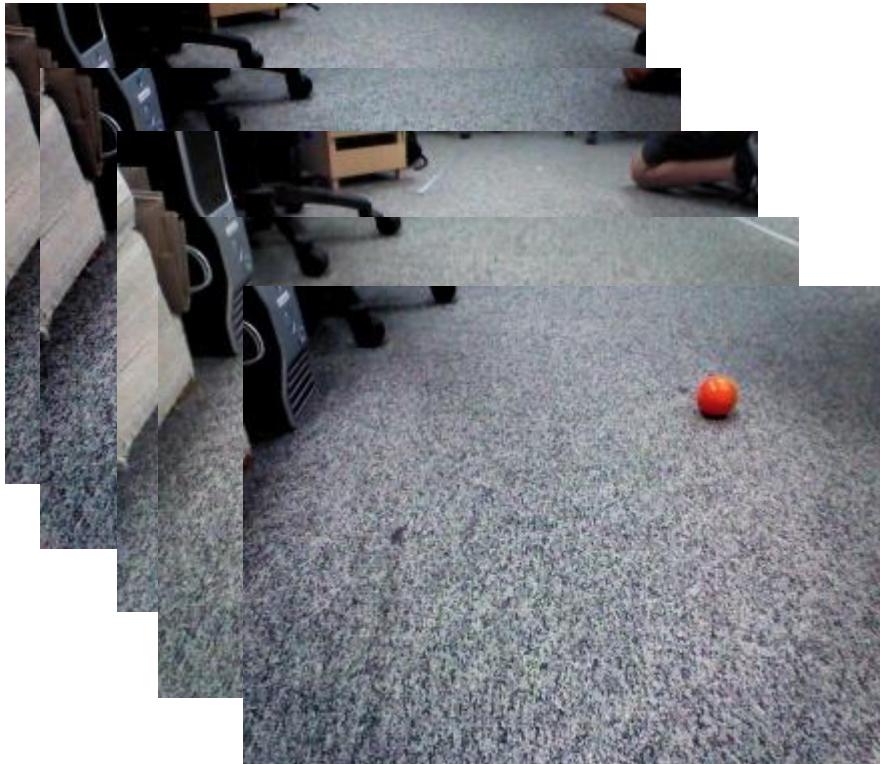
# GMM Intuition



# GMM Intuition



# Color Segmentation using GMM



**roipoly**  
→



# Color Segmentation using GMM

For all training images:

$$p(C_l | \mathbf{x}) \propto p(\mathbf{x} | C_l) p(C_l)$$

$$p(C_l) = \frac{1}{L}$$

$$p(\mathbf{x} | C_l) = \sum_{i=1}^K \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \Sigma_i)$$

$$\Theta = [\pi_i, \boldsymbol{\mu}_i, \Sigma_i] \quad \forall i \in [1, K]$$

Posterior

Prior

Likelihood

Estimated using EM

For any/each test image:

$$\forall \mathbf{x} \quad p(\mathbf{x} | C_l) = \sum_{i=1}^K \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \Sigma_i)$$

$\mathbf{x}$  is a pixel's **RGB** value

Predict using GMM