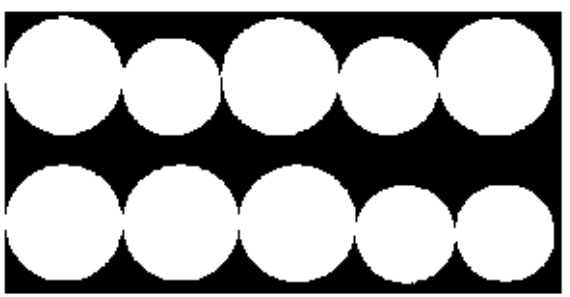
Lecture 2:
Color and Contrast

Object Detection

- The goal of object detection is to identify which pixels belong to a certain object or class (and which pixels do not).
- A detection mask is a binary image which assigns a 0 or 1 to each pixel:
  - 1=object detected (positive / WHITE)
  - 0=object not detected (negative / BLACK)



Grayscale Thresholding

- The simplest way to perform detection on a grayscale image is to threshold the pixels at some level T.

$$D(i,j) = \begin{cases} 1 & \text{if } A(i,j) < T \\ 0 & \text{if } A(i,j) \geq T \end{cases}$$

- For example, the coins are dark objects on a white background. So we would choose a threshold less than pure white, say T=200.

- If the coins were bright objects on a dark background, we may threshold the other way: $\mathrm{D(i,j)} = 1 \text{ if } A(i,j) > T$.

Logicals

- We create a 0-1 logical matrix in Matlab by putting a boolean statement in square brackets.
- This creates a matrix with value 1 where the statement is true, 0 where it is false.

Ex D = [A > 100];

A=

201	0	6	255
255	199	202	15
253	0	200	100

D=

1	0	0	1
1	1	1	0
1	0	1	0

Logicals

- Basic Matlab boolean operators

==	Equality	&	And
~	Not	|	Or

- Detect objects with a specific gray value:

```
D = [A == 153];
```

- Detect dark objects:

```
D = [A<115];
```

- Detect light objects:

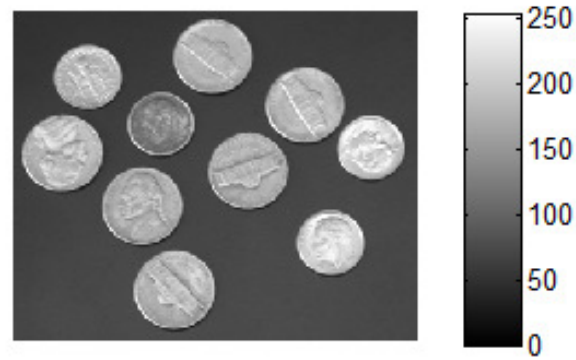
```
D = [A>180];
```

- Detect gray objects in a specific range:

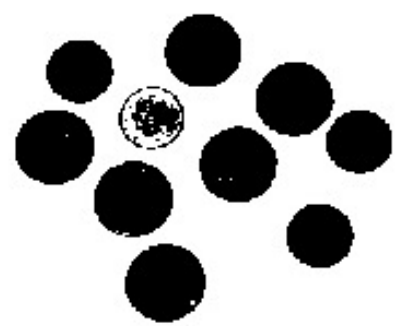
```
D = [ A>83 & A<192];
```

Selecting the Threshold

- Picking the right threshold value is a bit of an art form. It may require some trial and error.
- Thresholding is subjective. Different people may prefer different thresholds.
- It may help to add a colorbar to your image.
- The Matlab command graythresh uses *Otsu's algorithm* to pick a good dividing line, normalized to the range [0,1]. You may use this as a starting point.




```
A=imread('coins.png');
level = 255*graythresh(A)
                    126
subplot(121); imshow(A); colorbar;
subplot(122); imshow([A<126]);
```



Binary Image Operations

- After you get your thresholded binary image, you can clean it up using Matlab's binary image operations.
  - imdilate -- Make the blobs a little bigger.
  - imerode -- Make the blobs a little smaller.
  - bwareaopen -- Remove small blobs.
  - imfill -- Fill in the holes in the blobs.
  - watershed -- Separate overlapping blobs.

```
D = [A<200];
subplot(121);  imshow(D);
```

```
D2 = imfill(D, 'holes');
subplot(122);  imshow(D2);
```

Color Image Thresholding

- Thresholding color images is more difficult because we have 3 channels to consider.
- Essentially we have to set 6 threshold values.

```
D = [ A(:,:,1) > Rmin & A(:,:,1) < Rmax
    & A(:,:,2) > Gmin & A(:,:,2) < Gmax
    & A(:,:,3) > Bmin & A(:,:,3) < Bmax ];
```

Color Image Thresholding

- Ex Find the yellow M&Ms.

R _____

G _____

B _____

Alternative Color Spaces

- The standard RGB format is sometimes hard to work with.
- Several other schemes for storing color information have been proposed. These are called color spaces.
  - HSV
  - IHS
  - YIQ
  - YCbCr
  - CIE Lab
- It takes 3 numbers to make a specific color.

HSV Color Space

- To convert from RGB to the HSV space, we calculate

Let C=max(R,G,B) - min(R,G,B).

$V = \max(R, G, B)$

$$S = \begin{cases} \frac{C}{V} & \text{if } V \neq 0 \\ 0 & \text{if } V = 0 \end{cases}$$

$$H = \frac{1}{6} * \begin{cases} \frac{G - B}{C} mod\ 6 & \text{if } \max(R,G,B) = R \\ \frac{B - R}{C} + 2 & \text{if } \max(R,G,B) = G \\ \frac{R - G}{C} + 4 & \text{if } \max(R,G,B) = B \end{cases}$$

- The Matlab commands rgb2hsv and hsv2rgb will do the conversion for us.
- The HSV values are normalized to be in the range [0,1].
- Note the imshow and imagesc commands are built for RGB images. Do *not* try to display the HSV image.

RGB vs. HSV

A = imread('ash.png');

Red A(:,:,1)

Green A(:,:,2)

Blue A(:,:,3)

B = rgb2hsv(A);

Hue B(:,:,1)

What is the basic color?

Saturation B(:,:,2)

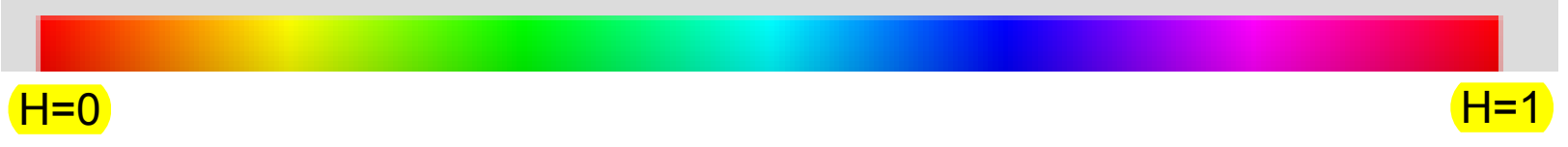
How vibrant is the color?

Value B(:,:,3)

How bright is the color?

Hue (H)

- The first channel H describes the "basic" color of that pixel.
- Note that the H colors are periodic.



- To shift the colors along the Hue scale by 0.2:

```
B=rgb2hsv(A);  B(:,:,1)=B(:,:,1)+0.2;  imshow(hsv2rgb(B));
```

Hue Thresholding

- If we want to threshold a color image, picking out specific RGB colors can be a pain.
- It is sometimes easier to pick out the Hue. Then we only need to threshold one channel.
- Ex Find the yellow M&Ms.

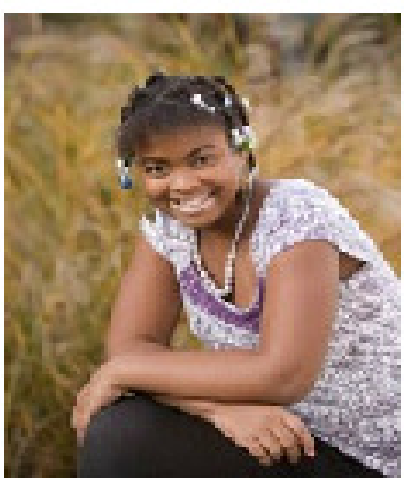

```
B=rgb2hsv(A);     D = [B(:,:,1)>0.1 & B(:,:,1)<0.2];
```

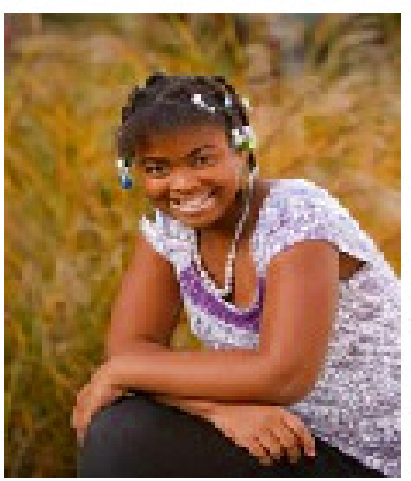
Saturation (S)

- The second channel S measures how vibrant the color is.
- Low saturation means the color is close to white.

- To increase the saturation by 50%

```
B=rgb2hsv(A);    B(:,:,2) = 1.5 * B(:,:,2);    imshow(hsv2rgb(B));
```





Value (V)

- The 3rd channel V measures the intensity or brightness of the color.
- Low value means the color is close to black.

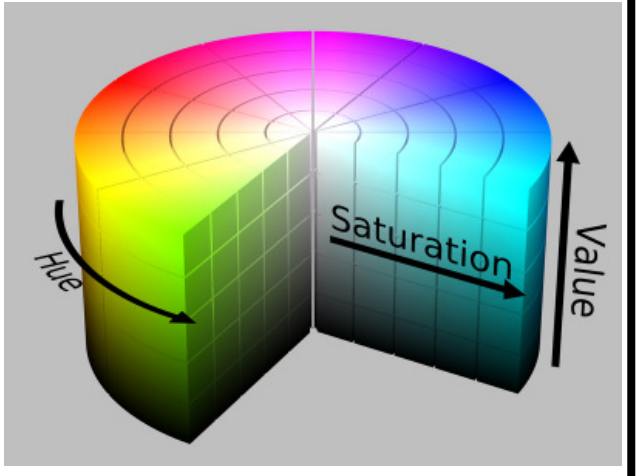
V=0 V=1

- To increase an image's value by 50%

```
B=rgb2hsv(A);   B(:,:,3)=1.5 * B(:,:,3);  imshow(hsv2rgb(B));
```

Advantages of HSV

- We can visualize the HSV color space as a cylinder.



- Compared to RGB, it is easier to do some things in the HSV color space.
  - Threshold a color image just on the Hue channel.
  - Make an image more vivid by increasing the Saturation channel.
  - Make an image brighter or darker by manipulating the Value channel.

Contrast

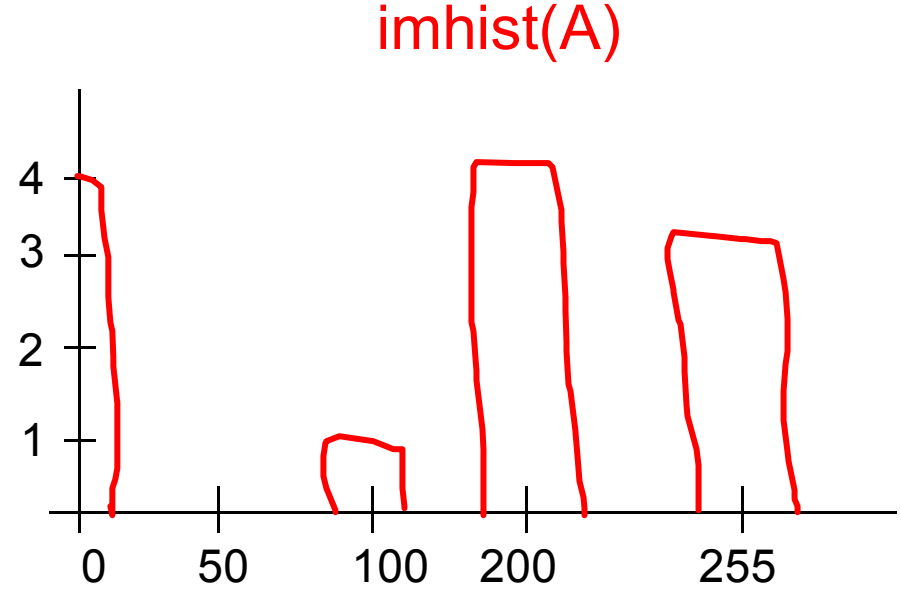
- Contrast is the difference in the image's brightness (or color) values.
- Objects are more visible in images with high contrast.
- Improving the contrast can make some tasks easier, such as thresholding to detect objects.

Image Histograms

- An image histogram displays the frequency of each gray value in the range [0,255].
- The Matlab command imhist displays the image histogram of a grayscale uint8 image.

A =

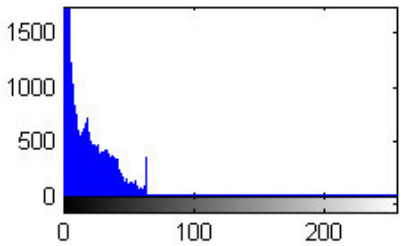
200	0	0	255
255	200	200	0
255	0	200	100



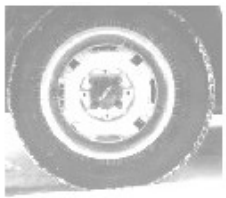
Analyzing Histograms

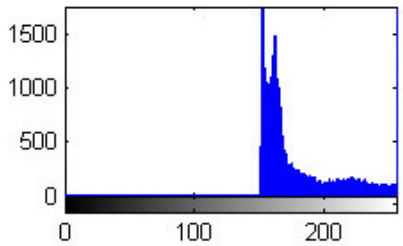
- The histogram reveals why the contrast on some images is poor.
- An image with good contrast should use all the gray values to a roughly equal extent (*uniform distribution*).

Dark image

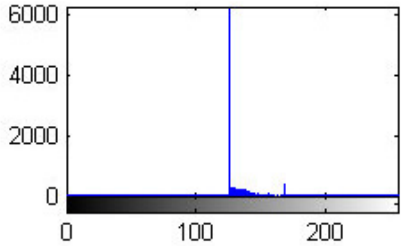


Light image

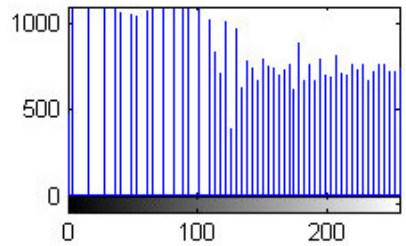




Low contrast



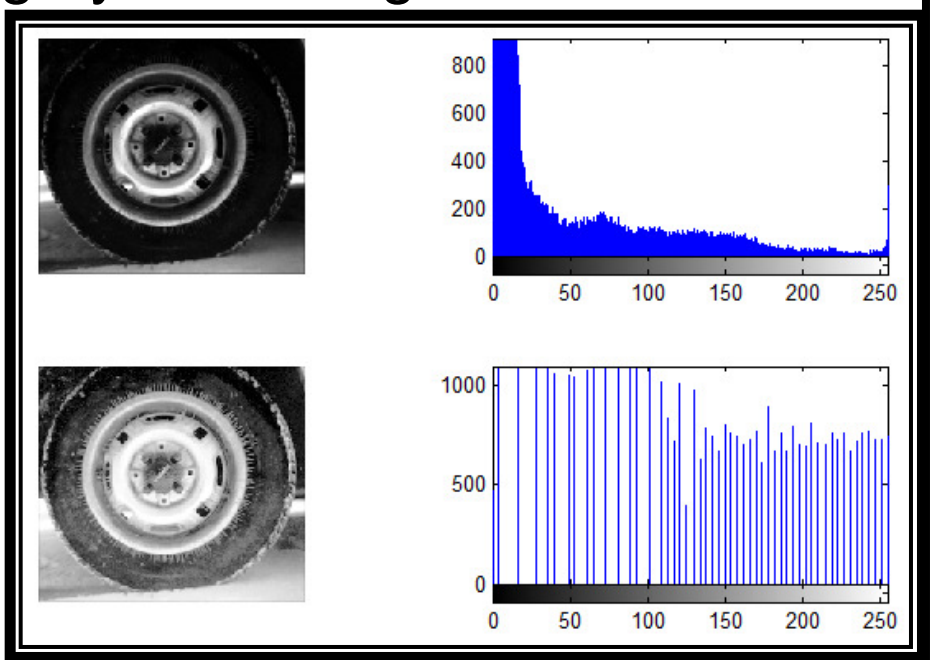
High contrast



Histogram Equalization

- The goal of histogram equalization is to redistribute the gray values of the image so that the histogram is flatter (more uniform).
- The Matlab command histeq produces the histogram equalized image of a uint8 grayscale image.

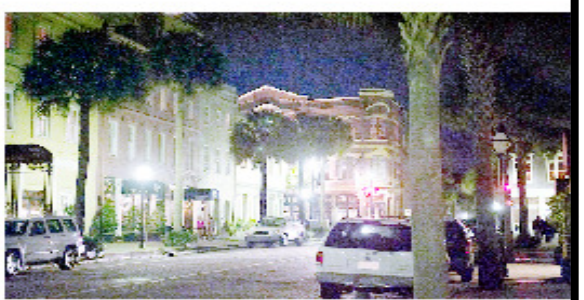
```
A=imread('tire.tif');
B=histeq(A);

subplot(221); imshow(A);
subplot(222); imhist(A);
subplot(223); imshow(B);
subplot(224); imhist(B);
```



Color Histogram Equalization

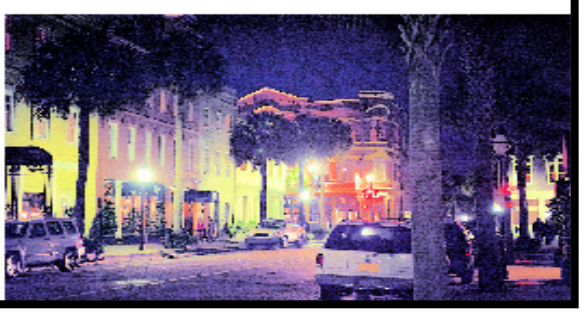
- The histeq method only works on a grayscale image.
- We could apply histogram equalization to each RGB channel.

```
for i=1:3
        B(:,:,i) = histeq(A(:,:,i));
end
```



- Or we could just apply histogram equalization to the Value channel.

```
B = rgb2hsv(A);
B(:,:,3) = histeq(B(:,:,3));
B = hsv2rgb(B);
```



Histogram Matching

- Instead of flattening the histogram, we could transform the histogram to a desired shape.
- Histogram matching is the process of making one image's histogram more closely resemble another image's histogram.
- The Matlab command imhistmatch takes two uint8 images as input and makes the first image's histogram match the second's.

```
A=imread('night.jpg');   B=imread('surf.jpg');   C=imhistmatch(A,B);
```

A

B

C