Color Tracking
Demo
Color Tracking: A Quick Overview

- Color Representations
- Choosing a Color to Track
- How to Find the Target
RGB vs HSV

- RGB is very sensitive to brightness
- HSV (Hue, Saturation, Value)
 HSV

- **Hue:** expressed as a number from 0 to 360 degrees representing hues of red (starts at 0), yellow (starts at 60), green (starts at 120), cyan (starts at 180), blue (starts at 240), and magenta (starts at 300).

- **Saturation:** How "pure" the color is. The closer to 0%, the more grey the color looks.

- **Value:** (or Brightness) works in conjunction with saturation and describes the brightness or intensity of the color from 0% to 100%.

```python
hsv_image = rgb2hsv(rgb_image)
```
Choosing a Color

Use a patch of pixels to determine target HS values rather than a single pixel.
Eroding and Dilating a Binary Image

**erode:**
The value of the output pixel is the *minimum* value of all the pixels in the input pixel's neighborhood

**dilate:**
The value of the output pixel is the *maximum* value of all the pixels in the input pixel's neighborhood
Connected-Component Labeling (a.k.a. Blob Extraction)
def get_target(hsv_image):

    # get pixels within threshold of target patch
    masked_image = mask_image(hsv_image, h_thresh, s_thresh)

    # morphologically open the image (i.e. erode and dilate it)
    opened_image = open_image(masked_image)

    # find the largest connected component (largest blob)
    cc_mask = get_largest_cc(opened_image)

    # x,y of the target center ("center of mass" of the target pixels)
    centroid = get_centroid(cc_mask)

    # use to determine how close we are to the object
    area = get_area(cc_mask)

    return centroid, area
Center the Target

- Turn to center the Target's Centroid

- Move Forward/Backward, and Turn Faster/Slower based on Target Area (i.e. distance to Target)