# Lab 2A

# Color Images

This lecture is part of the RACECAR-MN introductory robotics course.
You can visit the course webpage at mitll-racecar-mn.readthedocs.io.

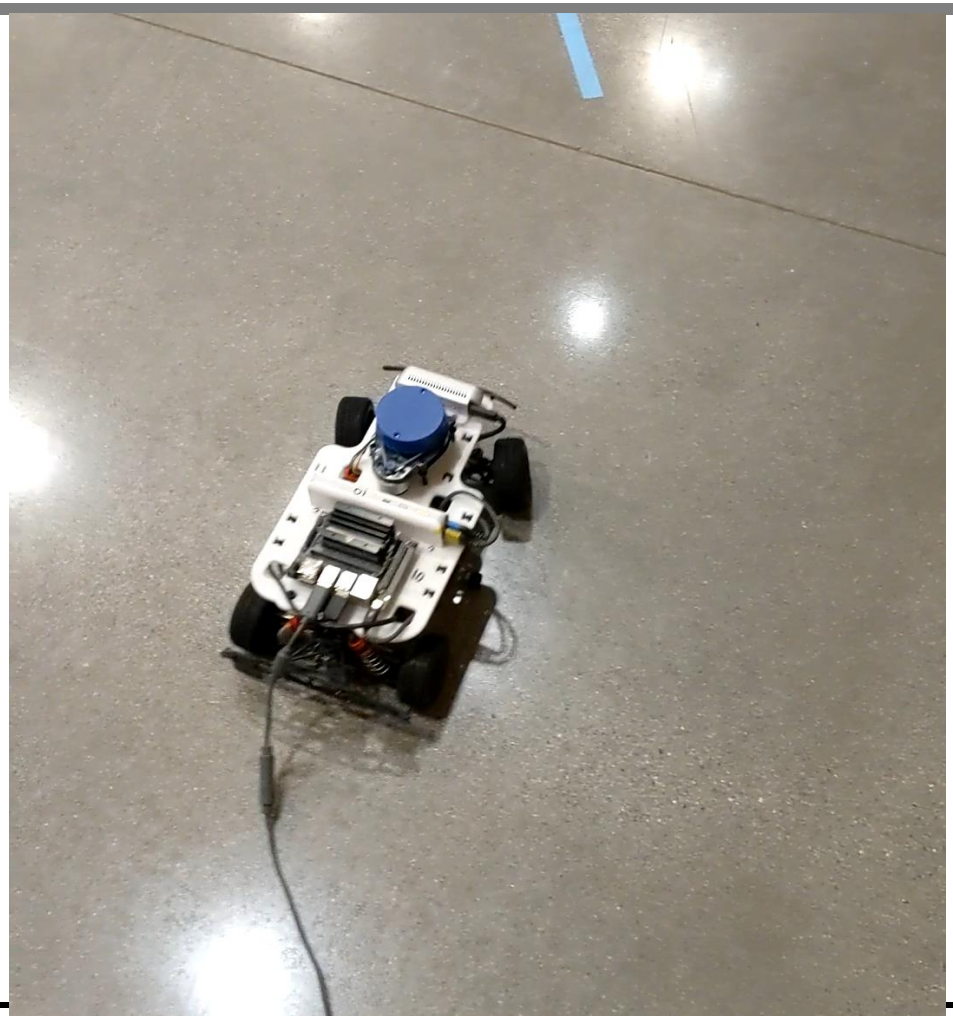
# Objectives

**Main Objective**: Identify objects in a color image based on color

## Learning Objectives

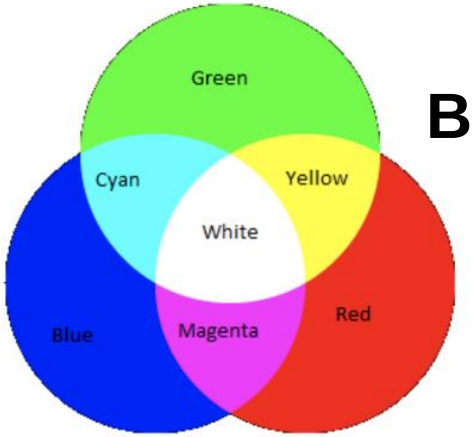
- Learn how to use Jupyter notebooks
- Use OpenCV functions to build image processing functions
- Understand and implement proportional control

# Lab 2 Demo



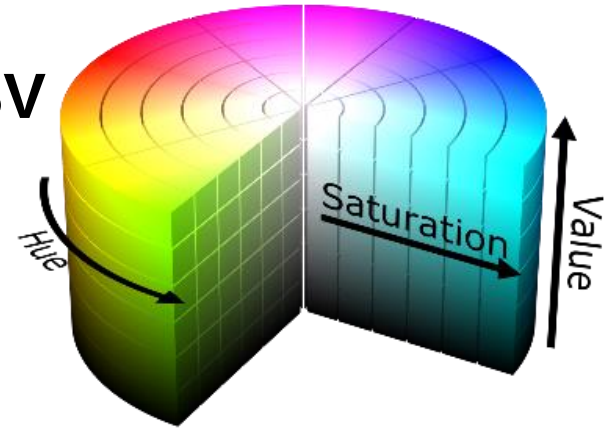
# Color Space

- Color models map input values to colors
- **Color space** is a range of colors a **color model** covers
- **BGR** models use measurements of blue, green, red



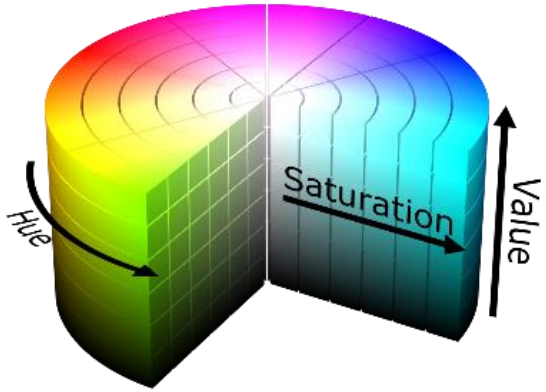

**BGR**

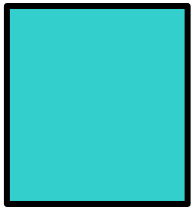
**HSV**



# HSV

- **Hue**: the base color
  - Range: 0°  - 180°
- **Saturation**: inverse the amount of white
  - Range: 0 - 255
- **Value:** inverse the amount of black
  - Range: 0 - 255

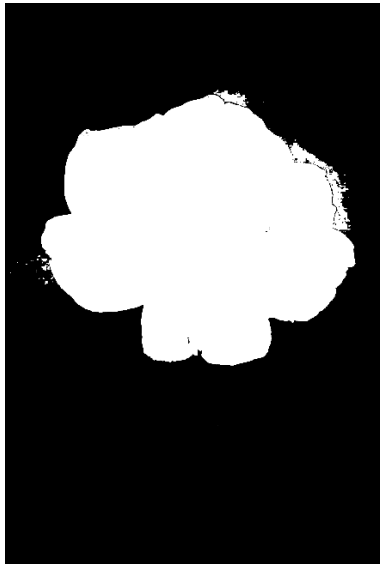


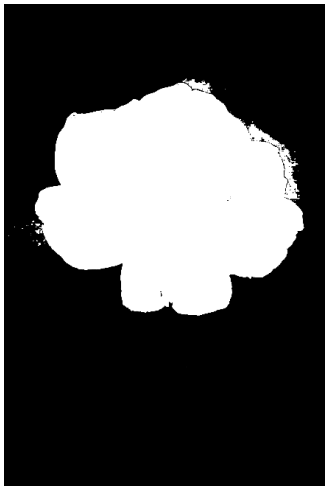
- **H:** 90°
- **S:** 194
- **V:** 207

# OpenCV

- Using a color **threshold**, we can make masks from images
- **BLACK** out the part of the image you want to remain hidden
- WHITE out the part of the image you want to be seen

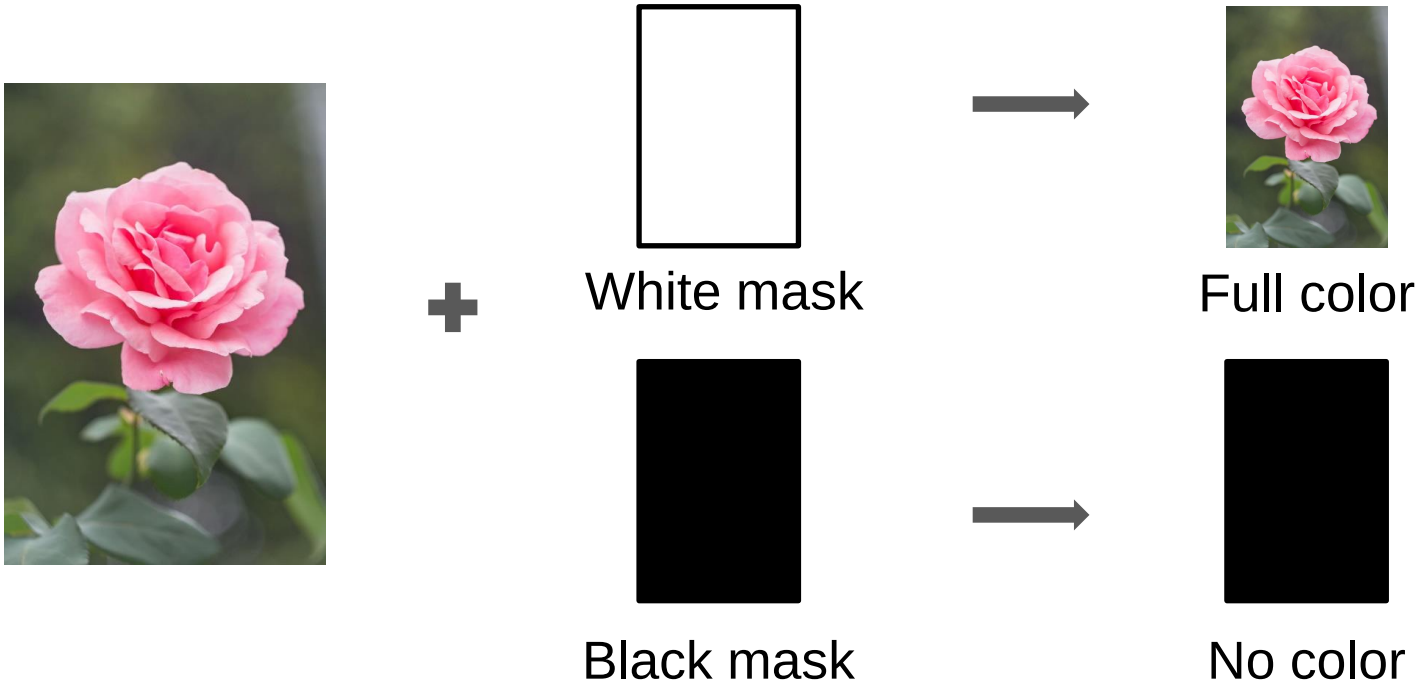


# Color Masking


- **Combine** the color image with the black and white mask to make a **color mask**

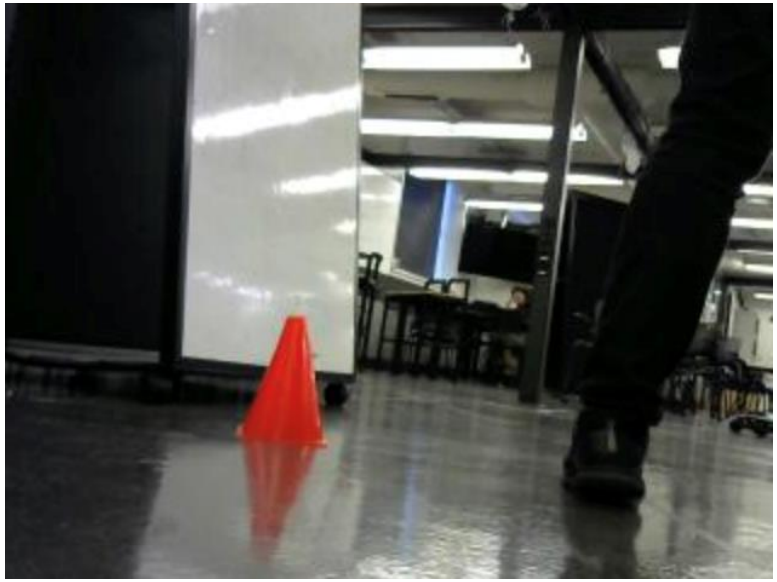
# Color Masking




- **Color masks** can be made with any black and white image.

# Contour Drawing

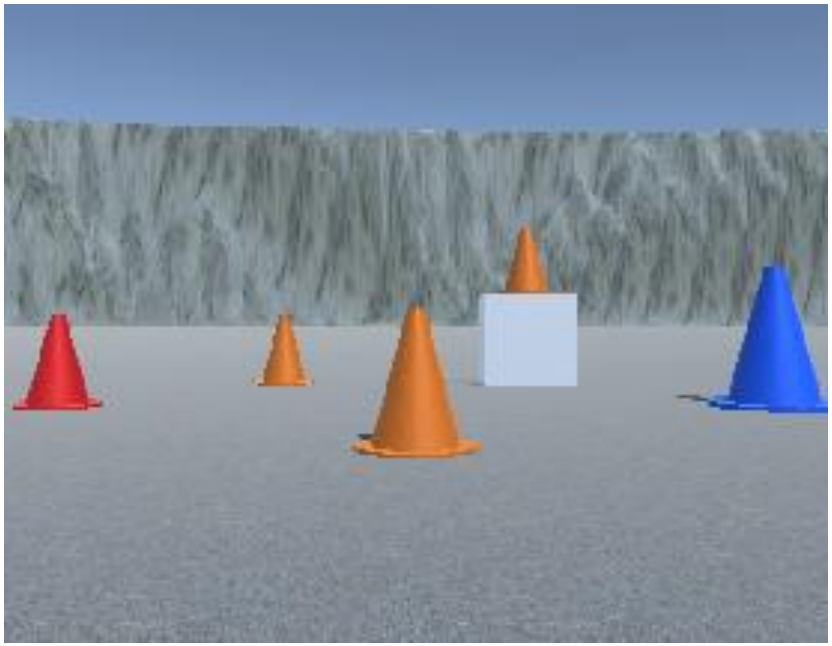
- With **color masks**, we can identify objects and create **contour drawings**



# Common Issues

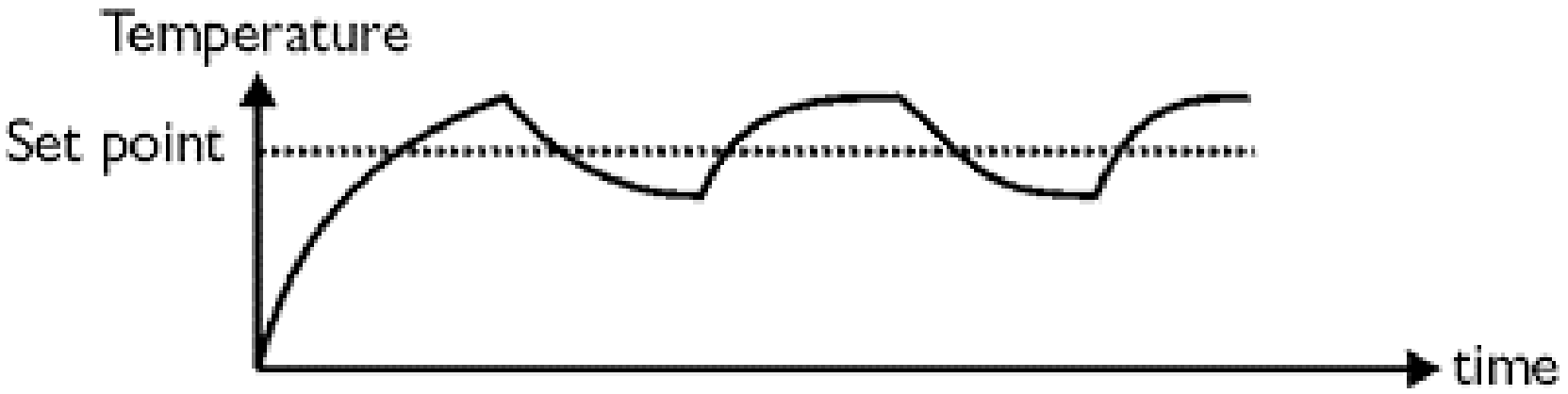
Group activity

- What issues might we have with these images and how would we address them?



# Bang-Bang Control

- **Bang-Bang control** uses feedback from the environment to switch between two **states**. This method of control oscillates around the set point in the acceptable bounds.



# Proportional Control

- **Proportional control** uses feedback from the environment to proportionally set output values. Input is linearly proportional to output

# Camera Module

- Retrieves color and depth images from the camera
- Public Interface
  - `get_color_image()`
  - `get_width()`
  - `get_height()`
  - `get_depth_image()` (we'll use this next time)

# Display Module

- Displays data and images to the screen
  - Simulation: Creates a window on your computer
  - RACECAR: creates a window on the mini-monitor
- Public Interface
  - `show_color_image()`
  - `show_depth_image()` (we'll use this later)
  - `show_lidar()` (we'll use this later)

# Example 1

Group activity

```
image = rc.camera.get_color_image()

for r in range(0, rc.camera.get_height()):
  for c in range(0, rc.camera.get_width()):
    foo = (image[r][c][0] + image[r][c][1] + image[r][c][2]) // 3
    image[r][c][0] = foo
    image[r][c][1] = foo
    image[r][c][2] = foo

rc.display.show_color_image(image)
```

# Example 2

Group activity

```
image = rc.camera.get_color_image()

foo = 0
bar = (0, 0)

for r in range(0, rc.camera.get_height()):
  for c in range(0, rc.camera.get_width()):
    if image[r][c][0] > foo:
      foo = image[r][c][0]
      bar = (r, c)

print(bar)
```

# Lab 2 Objectives

- **Jupyter Notebook**: Write helper functions which use masks/contouring to identify objects based on color
- **Lab 2A**: Implement line following
- **Lab 2B**: Cone parking

# racecar_utils Library

- Helper functions to process inputs/outputs of racecar_core
- You will write everything in racecar_utils during the Jupyter notebook activates throughout the course
- Documentation
- Implementation

# Jupyter Notebooks

Example: lab2.ipynb