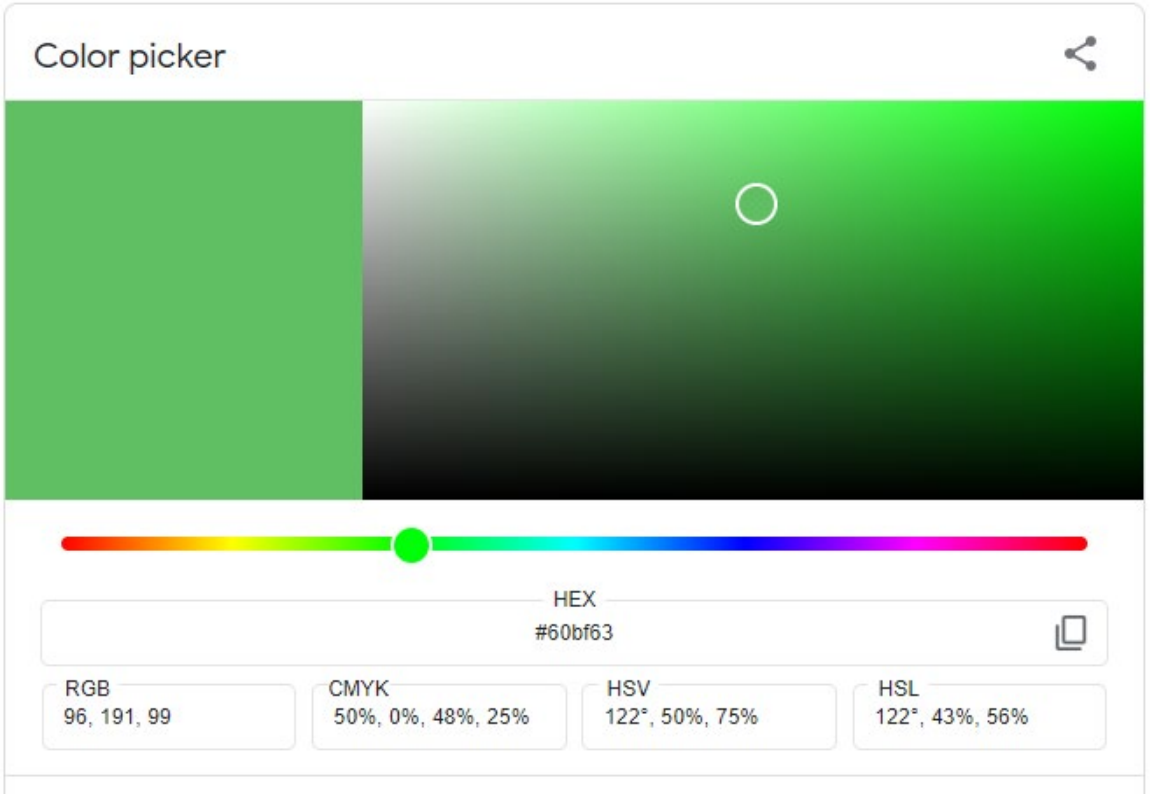
Computer Vision Color Spaces

MRE/EME 5983 Robot Operating Systems

Color Spaces

- Color spaces are a formal mechanism for organizing colors

- For example, google color picker: https://g.co/kgs/nHnZpd



Colors Spaces In Computer Vision

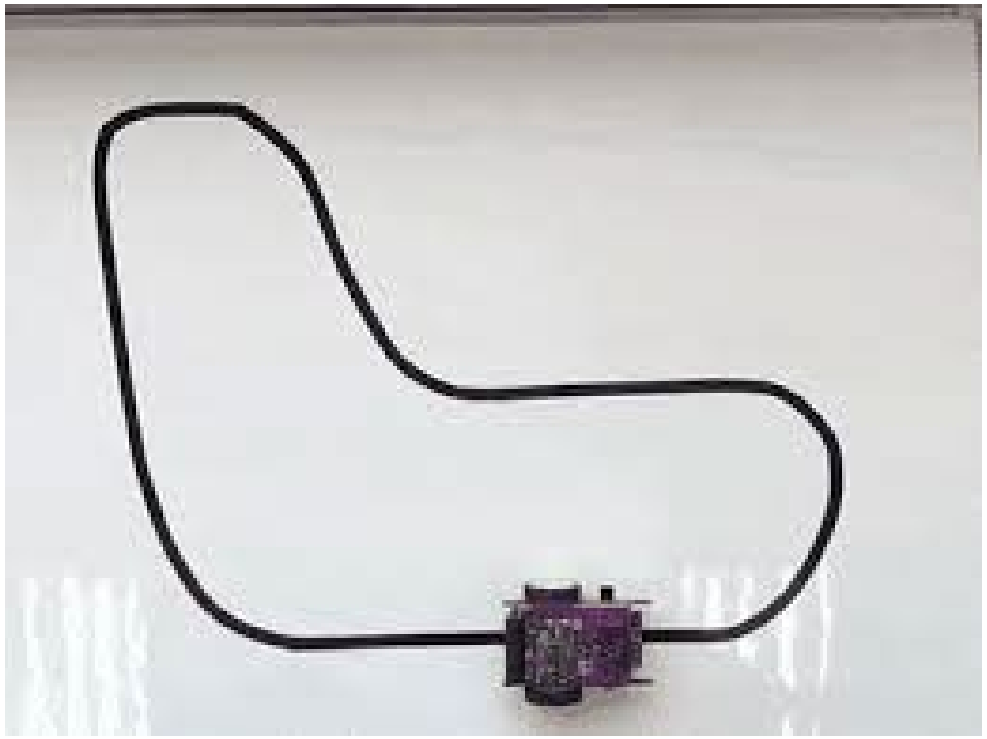
- Which color spaces are used in computer vision?

- OpenCV supports many color spaces, but we will leverage:
  - RGB/BGR: Red, Green, Blue / Blue, Green, Red
  - GRAY: Grayscale
  - Binary: Black and white
  - HSV: Hue, Saturation, Value (sometimes call HSB: Hue, Saturation, Brightness)
  - HLS: Hue, Lightness, Saturation

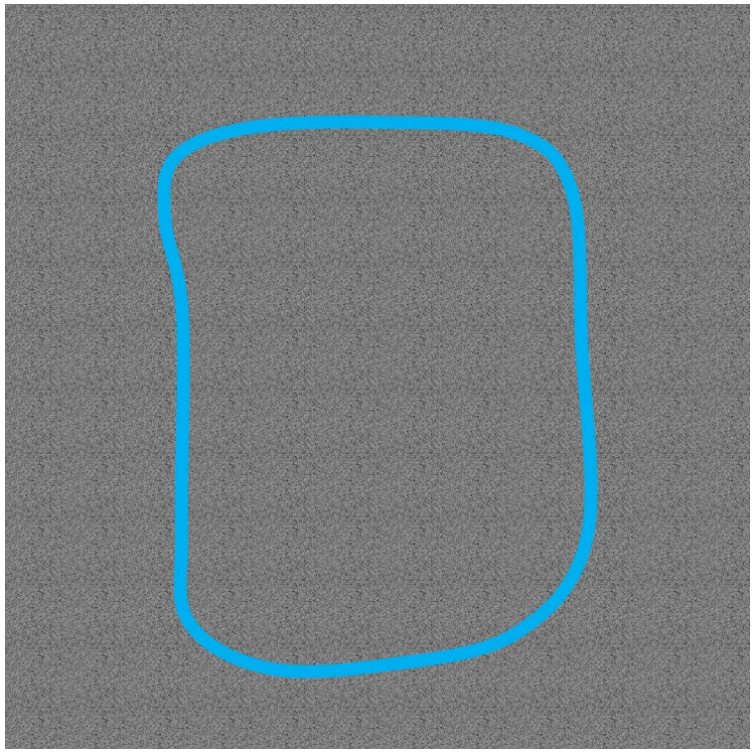
Colors Spaces In Computer Vision

- Motivation
  - Take a simple problem. Program a robot to follow a line…

Case 1



Case 2



Use BGR Color Space To Find A Line

opencv_bgr.py

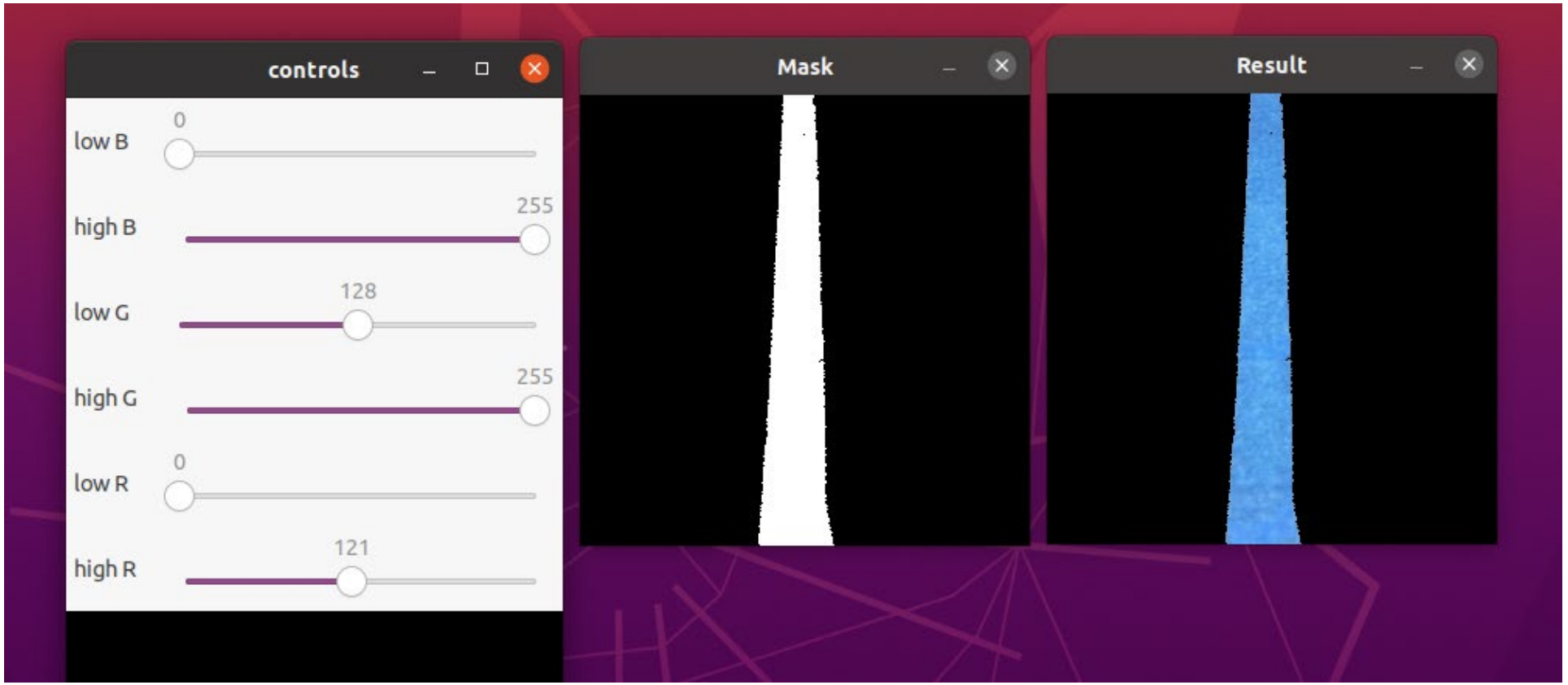
```
import cv2 as cv
import numpy as np
import sys

# Color ranges
B_low = 0
B_high = 255
G_low= 0
G_high = 255
R_low= 0
R_high = 255

# Trackbar callback fucntion to update BGR value
def callback(x):
    global R_low, B_high, G_low, G_high, R_low, R_high
    B_low = cv.getTrackbarPos('low B','controls')
    B_high = cv.getTrackbarPos('high B','controls')
    G_low = cv.getTrackbarPos('low G','controls')
    G_high = cv.getTrackbarPos('high G','controls')
    R_low = cv.getTrackbarPos('low R','controls')
    R_high = cv.getTrackbarPos('high R','controls')
    return


# Create the a controls window
cv.namedWindow('controls',2)

# Create trackbars for Low and High B, G, R
cv.createTrackbar('low B','controls',    0, 255, callback)
cv.createTrackbar('high B','controls', 255, 255, callback)
cv.createTrackbar('low G','controls',    0, 255, callback)
cv.createTrackbar('high G','controls', 255, 255, callback)
cv.createTrackbar('low R','controls',    0, 255, callback)
cv.createTrackbar('high R','controls', 255, 255, callback)
```

opencv_bgr.py

```
# Read the image
img_orig = cv.imread(sys.argv[1])

# Loop for edits
while(1):

    # Set up bounds
    bgr_low = np.array([B_low, G_low, R_low], np.uint8)
    bgr_high = np.array([B_high, G_high, R_high], np.uint8)

    # Get filter image
    mask = cv.inRange(img_orig, bgr_low, bgr_high)
    res = cv.bitwise_and(img_orig, img_orig, mask=mask)

    # Show images
    cv.imshow('Mask',mask)
    cv.imshow('Result',res)

    # Exit on key enter
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break

# Close all windows
cv.destroyAllWindows()
```

Use BGR Color Space To Find A Line – Example 1

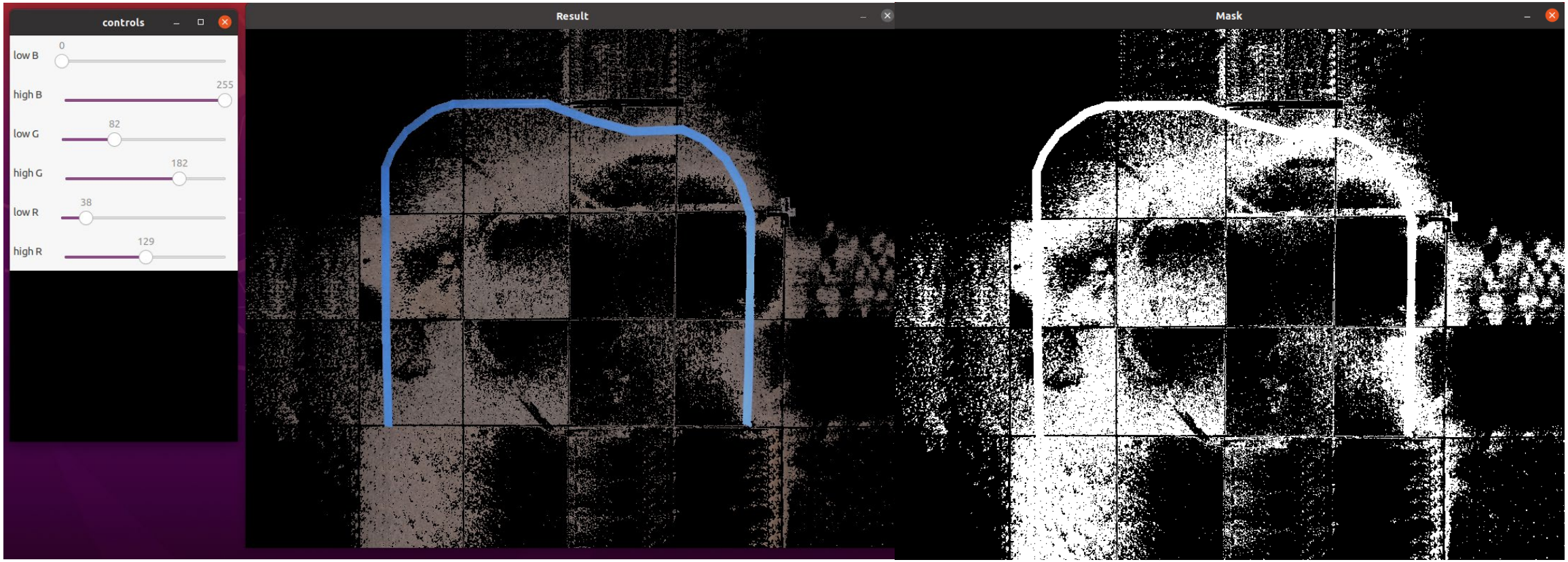
- python3 opencv_bgr.py ../img/blue_line_floor_1.png



- The BGR color space can be used to successfully find the blue line

Use BGR Color Space To Find A Line – Example 2

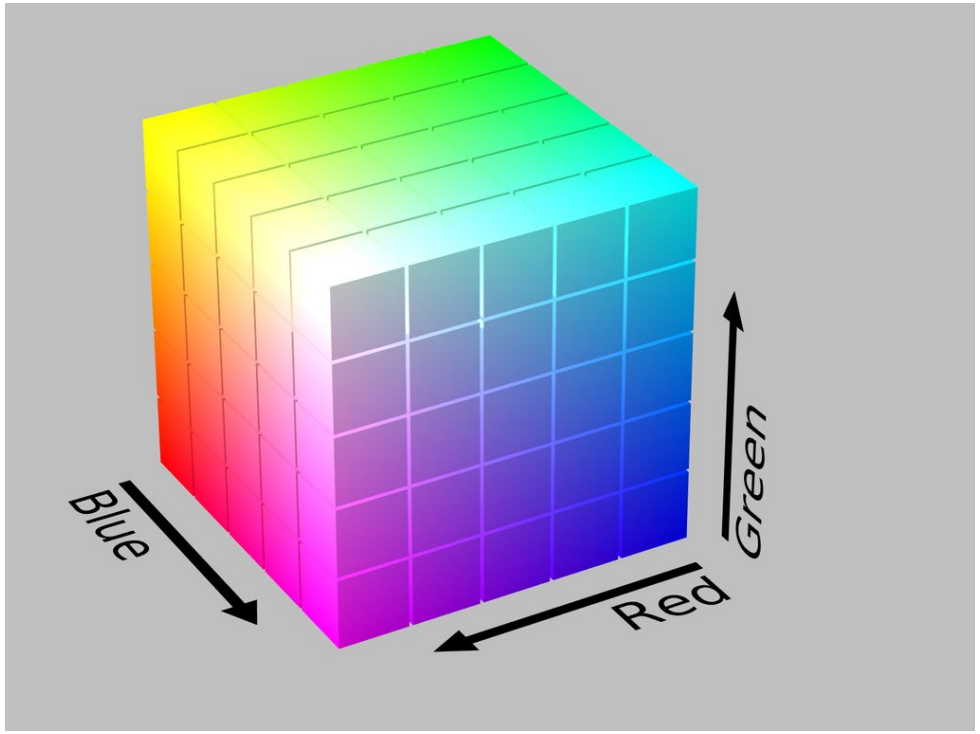
- python3 opencv_bgr.py ../img/blue_line_floor_2.png

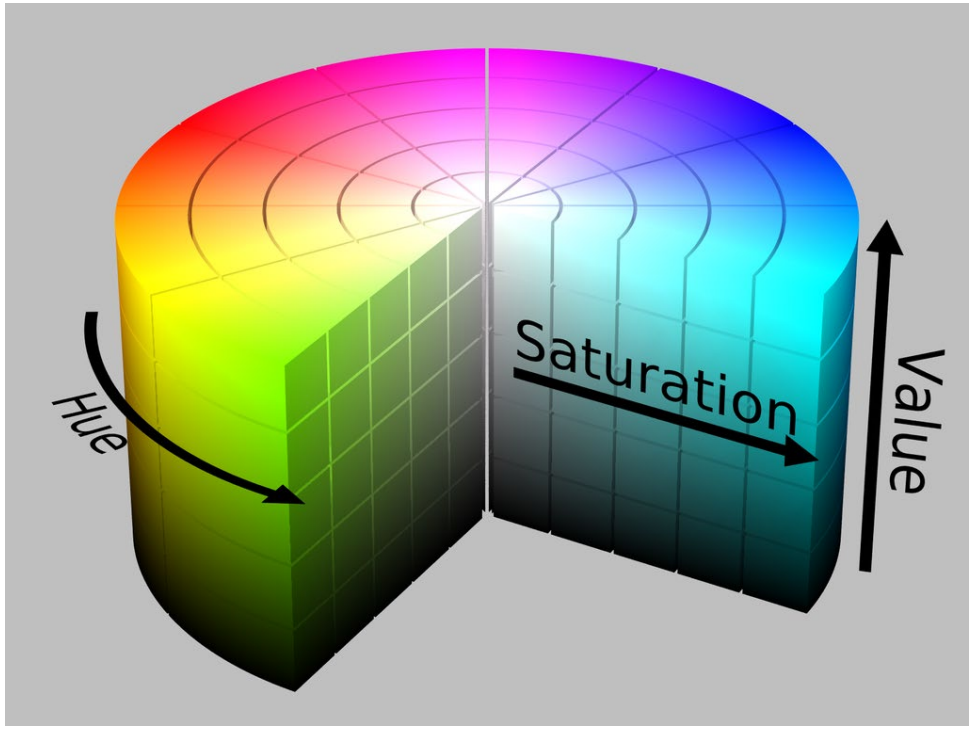


- The BGR color space cannot be used to successfully find the blue line!

Introducing A Different Color Spaces

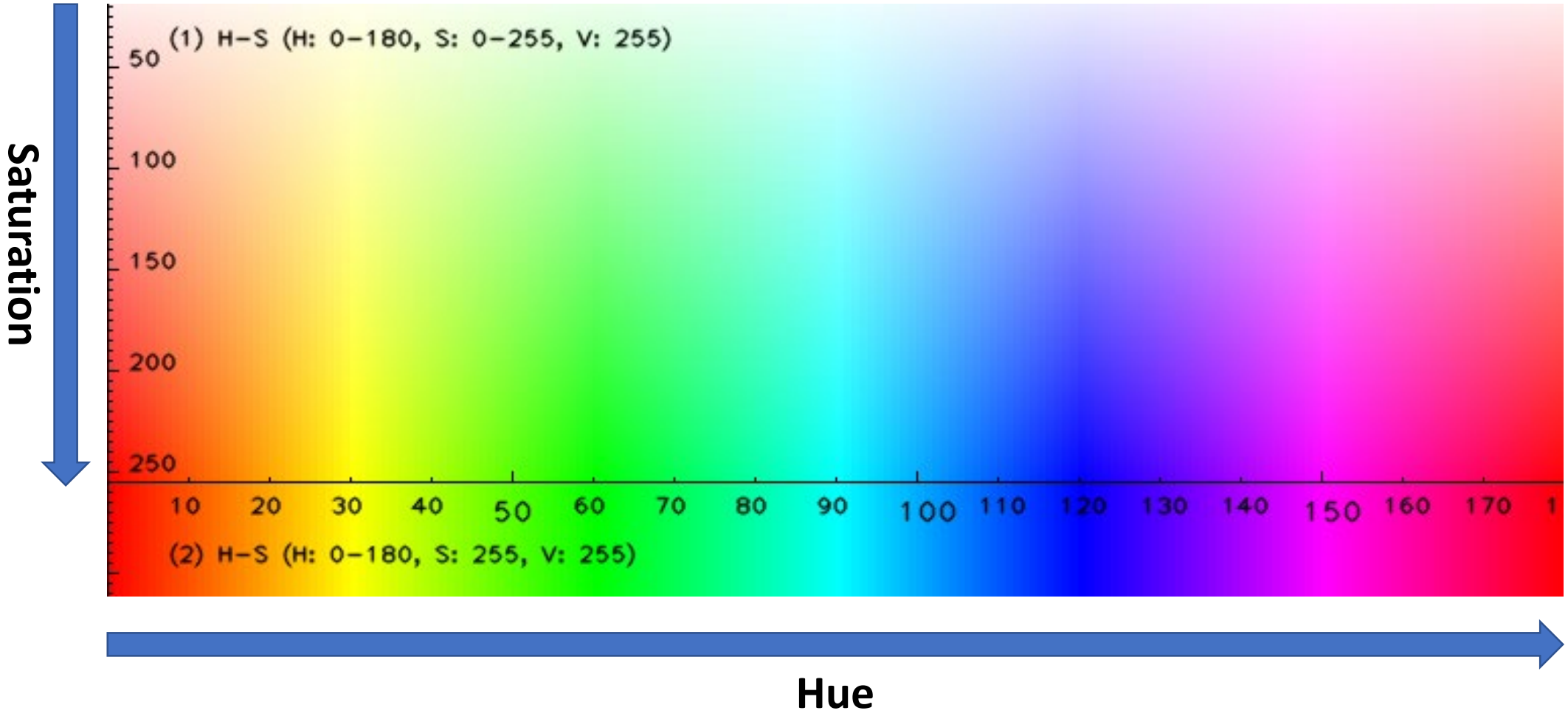
- Remember, color spaces are a mechanism for organizing colors
- We are not introducing new colors, just a different way of representing them





HSV two dimensional view

- For two dimensions, we can fix one variable…



Let's Use HSV Color Space To Find A Line

opencv_hsv.py

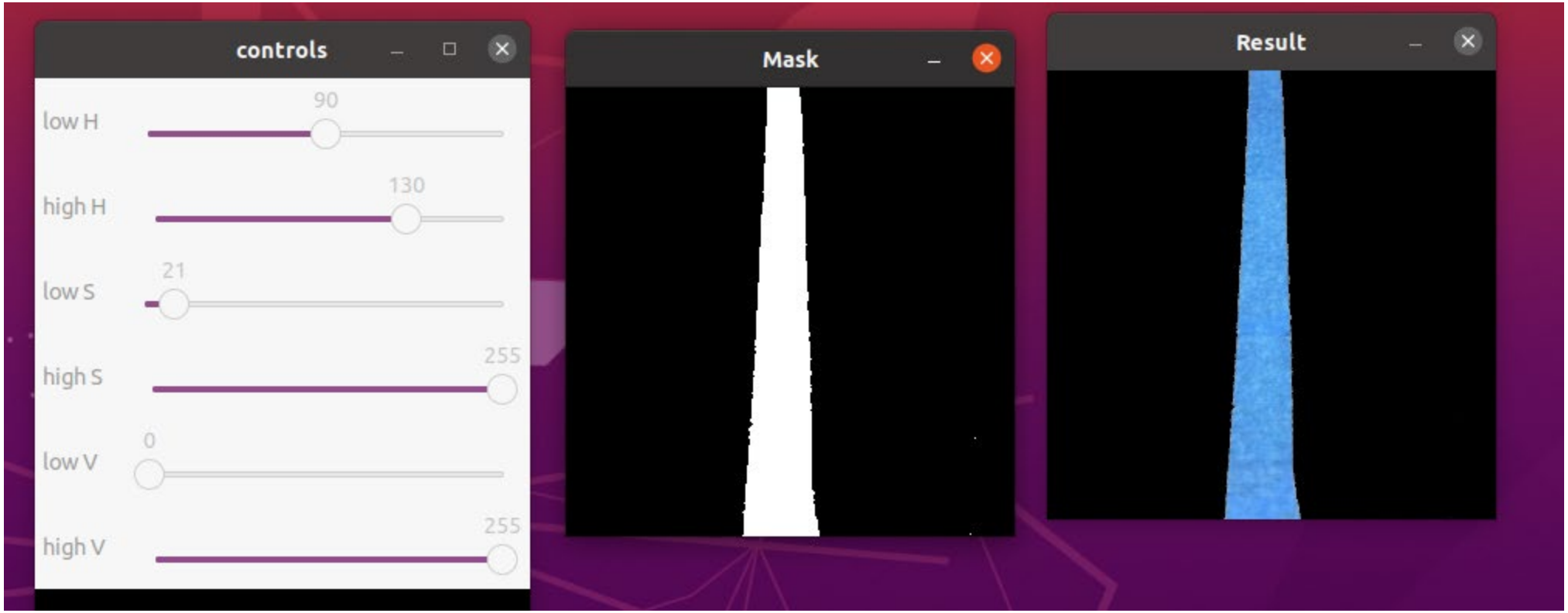
```
import cv2 as cv
import numpy as np
import sys

# Color ranges
H_low = 0
H_high = 179
S_low= 0
S_high = 255
V_low= 0
V_high = 255

# Trackbar callback fucntion to update HSV value
def callback(x):
    global H_low, H_high, S_low, S_high, V_low, V_high
    H_low = cv.getTrackbarPos('low H','controls')
    H_high = cv.getTrackbarPos('high H','controls')
    S_low = cv.getTrackbarPos('low S','controls')
    S_high = cv.getTrackbarPos('high S','controls')
    V_low = cv.getTrackbarPos('low V','controls')
    V_high = cv.getTrackbarPos('high V','controls')
    return


# Create the a controls window
cv.namedWindow('controls',2)

# Create trackbars for Low and High B, G, R
cv.createTrackbar('low H','controls',    0, 179, callback)
cv.createTrackbar('high H','controls', 179, 179, callback)
cv.createTrackbar('low S','controls',    0, 255, callback)
cv.createTrackbar('high S','controls', 255, 255, callback)
cv.createTrackbar('low V','controls',    0, 255, callback)
cv.createTrackbar('high V','controls', 255, 255, callback)
```

opencv_hsv.py

```
# Read the image
img_orig = cv.imread(sys.argv[1])

# Loop for edits
while(1):

    # Conver to HSV
    img_hsv = cv.cvtColor(img_orig, cv.COLOR_BGR2HSV)

    # Set up bounds
    hsv_low = np.array([H_low, S_low, V_low], np.uint8)
    hsv_high = np.array([H_high, S_high, V_high], np.uint8)

    # Get filter image
    mask = cv.inRange(img_hsv, hsv_low, hsv_high)
    res = cv.bitwise_and(img_orig, img_orig, mask=mask)

    # Show images
    cv.imshow('Mask',mask)
    cv.imshow('Result',res)

    # Exit on key enter
    k = cv.waitKey(1) & 0xFF
    if k == 27:
        break

# Close all windows
cv.destroyAllWindows()
```

Use BHSV Color Space To Find A Line – Example 1

- python3 **opencv_hsv**.py ../img/blue_line_floor_1.png



Use HSV Color Space To Find A Line – Example 2

- python3 **opencv_hsv**.py ../img/blue_line_floor_2.png

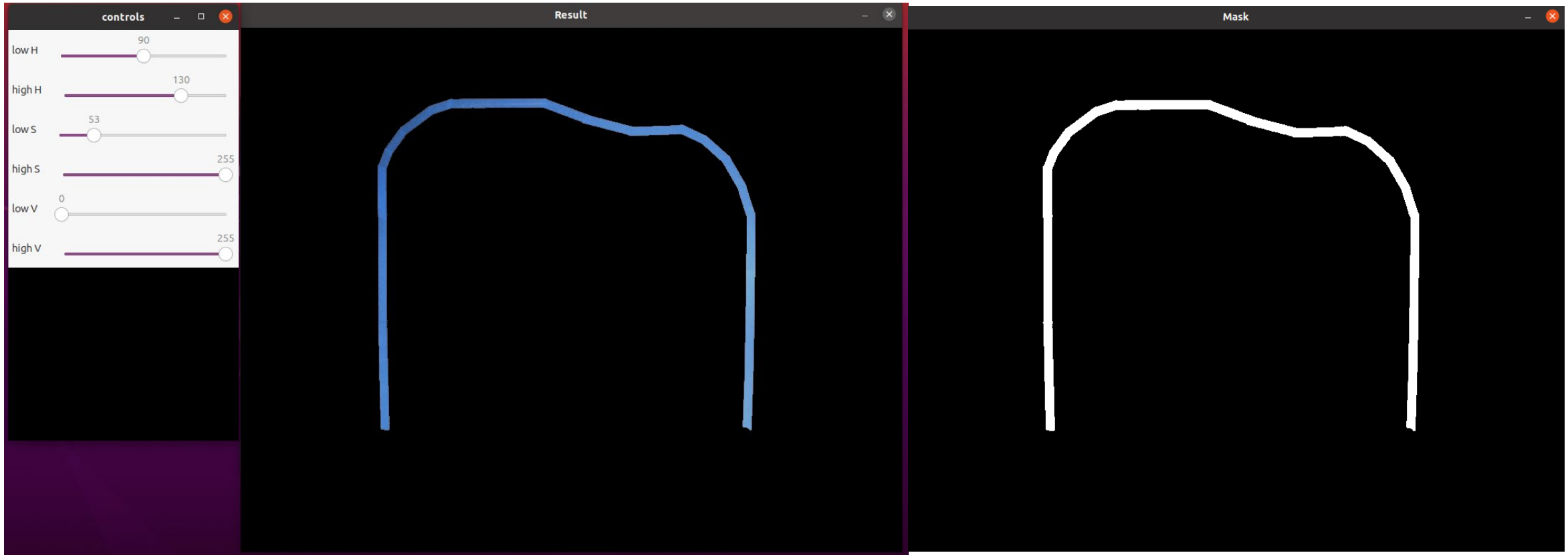


Image Processing With Dynamic Reconfigure

- We can use ROS dynamic reconfigure to create tunable parameters for filtering images

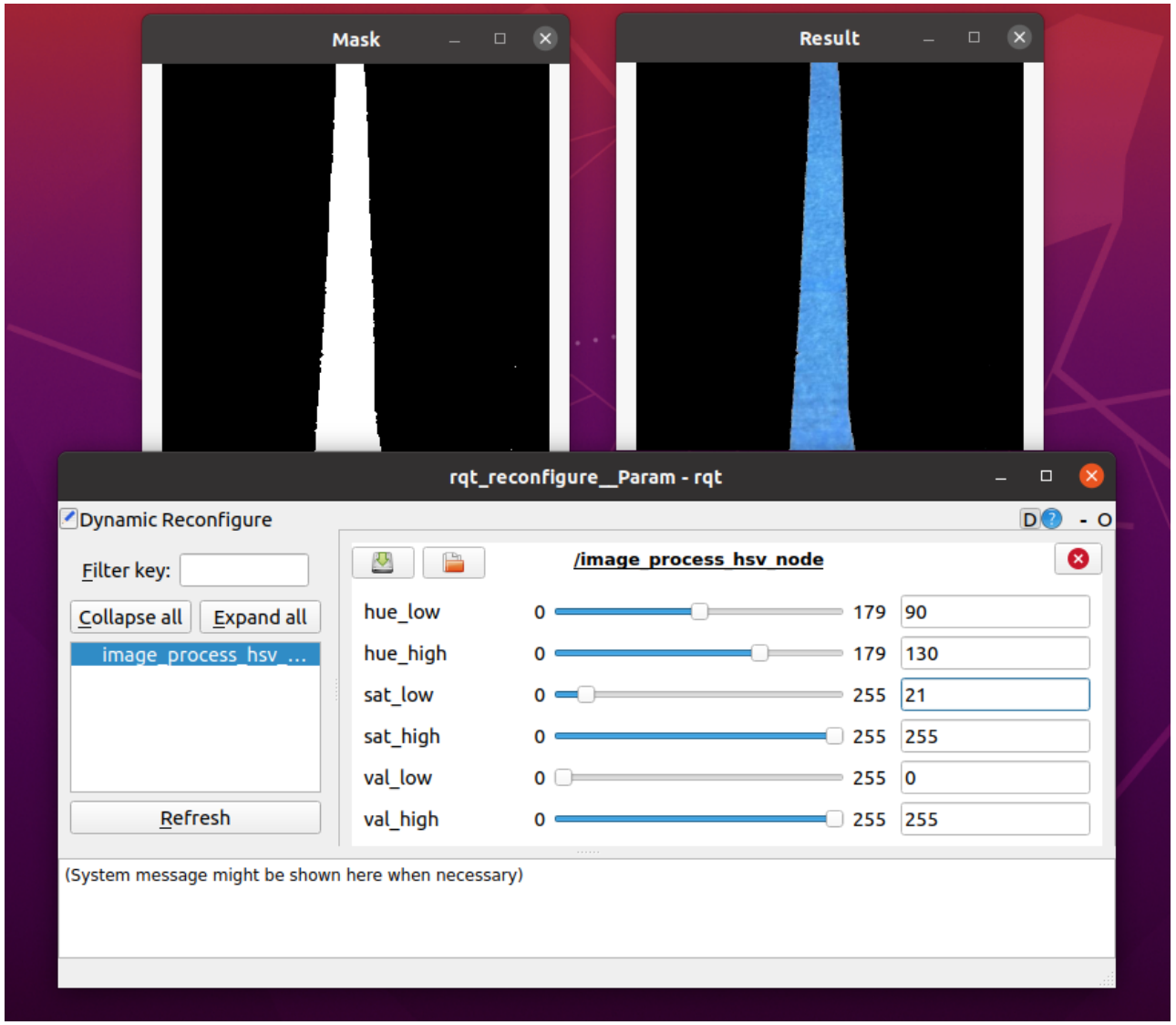
- We can add the following the our dynamic reconfigure parameters

```
gen.add('hue_low',        int_t,   0, 'Hue Low',          0,   0,  179)
gen.add('hue_high',       int_t,   0, 'Hue High',       179,   0,  179)
gen.add('sat_low',        int_t,   0, 'Sat Low',          0,   0,  255)
gen.add('sat_high',       int_t,   0, 'Sat High',       255,   0,  255)
gen.add('val_low',        int_t,   0, 'Value Low',        0,   0,  255)
gen.add('val_high',       int_t,   0, 'Value High',     255,   0,  255)
```

Example HSV Image Processing In ROS

- There is an example of HSV image processing in the course_tutorials package

$ roslaunch course_tutorials hsv_image_process.launch



Summary

- We reviewed how we can leverage color space to enhance the performance of computer vision

- We will use color spaces to assist in programing robots to line and or lane follow