# masks.py — Color Mask Definitions

This file defines color masks for use in image processing, specifically in the LAB color space. These masks allow the robot to reliably detect various colors in its environment, which is essential for stripe following, wall detection, and course logic.

## What are Color Masks?

A **color mask** is a set of lower and upper bounds in a color space (here, LAB) used to isolate pixels of a specific color from an image. For example, to find all “orange” pixels, you use the rOrange mask.

## Why LAB Color Space?

* **LAB** color space separates lightness (L) from color channels (A/B), making color detection more robust against lighting changes and shadows compared to RGB.
* Each mask is a pair of NumPy arrays: [lower\_bound, upper\_bound].

## List of Masks Defined

* **rMagenta**: For magenta stripes.
* **rRed**: For red stripes.
* **rGreen**: For green stripes.
* **rBlue**: For blue stripes (used for one course path).
* **rOrange**: For orange stripes (used for the alternate course path).
* **rBlack**: For wall detection (black lines or walls).

All masks are defined as follows:

rMagenta = [np.array([22, 150, 50], np.uint8), np.array([200, 200, 150], np.uint8)]  
rRed = [np.array([0, 150, 120], np.uint8), np.array([255, 220, 180], np.uint8)]  
rGreen = [np.array([0, 50, 0], np.uint8), np.array([255, 120, 255], np.uint8)]  
rBlue = [np.array([0, 120, 60], np.uint8), np.array([255, 160, 120], np.uint8)]  
rOrange = [np.array([0, 117, 150], np.uint8), np.array([255, 200, 200], np.uint8)]  
rBlack = [np.array([0, 105, 105], np.uint8), np.array([80, 151, 151], np.uint8)]

## How are Masks Used?

* **Imported** in functions.py and main.py.
* Passed into image processing functions like find\_contours() to segment and find contours of the specific color within a region of interest (ROI).
* Used in stripe detection, wall following, and turn logic.

## Example Usage

from masks import rOrange, rBlack  
  
# In an image processing function:  
orange\_contours = find\_contours(img\_lab, rOrange, ROI3)  
black\_contours = find\_contours(img\_lab, rBlack, ROI1)

* This finds all orange stripes in the main ROI and all black wall contours in the left ROI.

## Tuning and Updating Masks

**LAB color masks may need to be adjusted for different lighting conditions, cameras, or colored materials.** To update these values:

1. **Use the LAB Mask Tuner Tool (lab\_mask\_tuner.py in this repo):**
   * This script lets you interactively adjust LAB range sliders while viewing a live mask overlay.
   * Place your sample object in the ROI, tune the sliders, and press q to print the final LAB bounds.
2. **Replace the values in masks.py:**
   * Copy the printed lower and upper bounds into your mask definitions.

*This ensures your robot detects colors reliably in your specific environment.*

## Additional Variable

* **lotType**: Set to "light" (purpose: for lighting condition or mode selection, but not actively used in main code).

## Why Is This Important?

Correct color mask definitions are critical for reliable robot vision. - Too narrow: robot may miss stripes/walls under different lighting. - Too broad: robot may detect noise or wrong objects.

## Troubleshooting

* If your robot fails to detect stripes or walls, use the LAB Mask Tuner to find better values for your lighting/camera.
* Use OpenCV tools to sample LAB values from your camera images in your actual environment.

## References

* [LAB Color Space](https://docs.opencv.org/4.x/de/d25/imgproc_color_conversions.html)
* [NumPy Arrays](https://numpy.org/doc/stable/)
* [OpenCV InRange Function](https://docs.opencv.org/4.x/df/d9d/tutorial_py_colorspaces.html)
* [LAB Mask Tuner Tool](./lab_mask_tuner.py)