

**Department of Artificial Intelligence**

**College of Computer Science and Information Technology**

**Due Date: Tuesday November 19, 2024 @ 11:59 PM**

**Late Submissions:**

* Q: Can I skip the lab and submit the solution?
  + You will receive a mark of **zero** if you do not attend the lab, even if you complete the exercise. Attending the labs is compulsory for evaluation. If you have a justified excuse, you may receive a partial mark depending on the circumstances. See the next question for information on late submissions.
* **Q:** If I submit it at 12:00am, you’ll still mark it, right?
  + **A:** 11:59pm and earlier is on time. Anything after 11:59pm is late. Anything late will **NOT** be probably marked. If I find you have a legitimate cause, you will be graded according to the following rules (24 hours after deadline 🡪 assignment is marked out of 75% only, 48 hours after deadline 🡪 assignment is marked out of 50% only, 72 hours after deadline 🡪 assignment is marked out of 25% only)

**Objective**

This lab will guide you through the process of enhancing the contrast of RGB images using different methods. You will learn how to manipulate each RGB channel independently, as well as perform contrast adjustments in the HSL color space to better understand color manipulation. The lab will also cover additional image processing tasks for further exploration of multi-channel image manipulation using Python and the skimage library.

**Background**

**1. RGB Color Space**

RGB stands for Red, Green, and Blue, the three primary colors used to represent images on screens. In the RGB model, an image is stored as three separate layers (or channels) for each color. Each pixel in the image has three values, each between 0 and 255 (or normalized to 0.0–1.0), representing the intensities of red, green, and blue. When combined, these channels produce the full-color image.

**2. HSL Color Space**

HSL stands for Hue, Saturation, and Lightness (or value), which represent a different way of encoding color that can sometimes make color manipulation more intuitive.

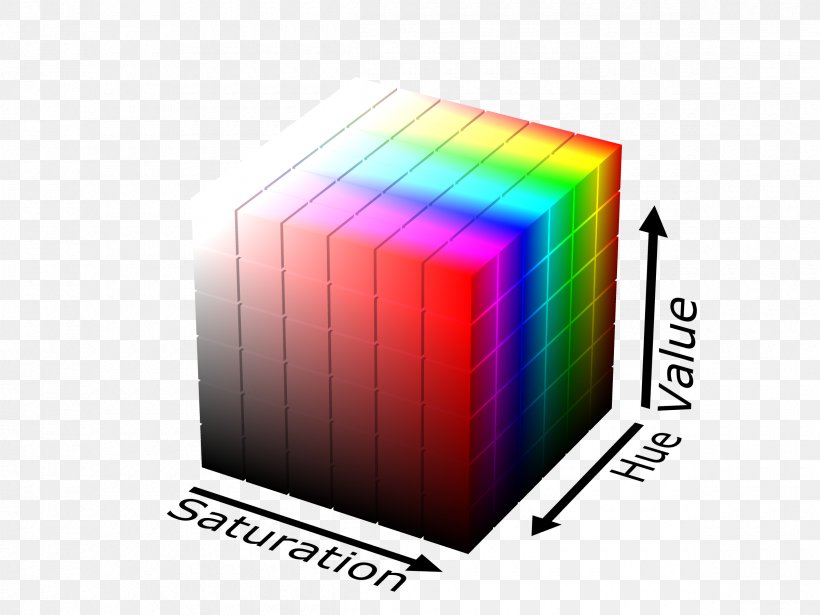
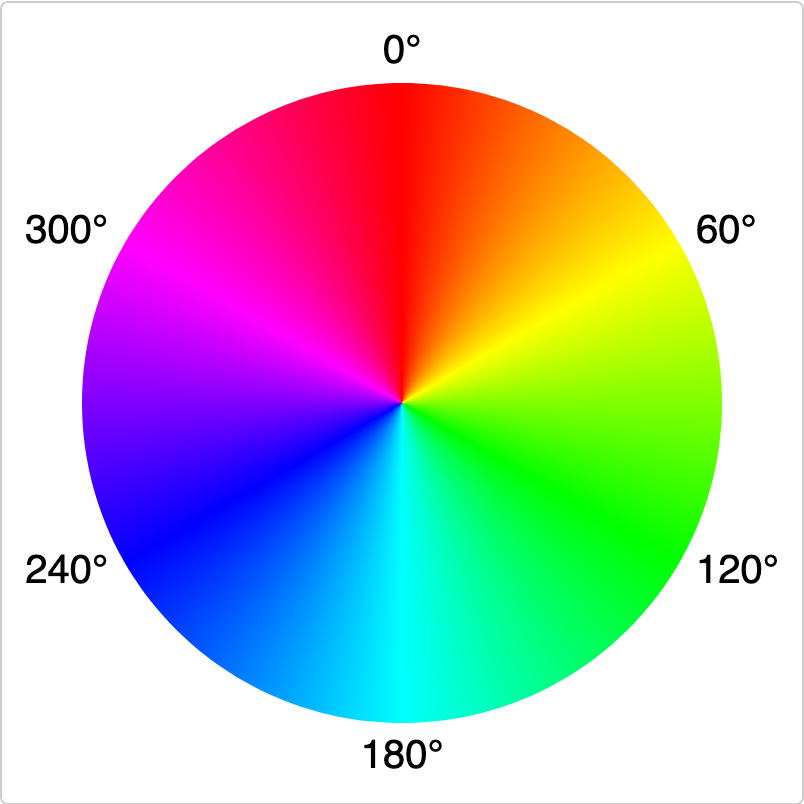
 

Figure 1: Left image: HSL Color Space, right image: Hue color wheel

Source: (<https://favpng.com/png_view/cube-rgb-color-model-hsl-and-hsv-rgb-color-space-cube-png/Fr06aXV2>, <https://developer.mozilla.org/en-US/blog/learn-css-hues-colors-hsl/>)

* **Hue**: Defines the color type, represented as an angle on a color wheel (0–360 degrees), where 0 is red, 120 is green, and 240 is blue.
* **Saturation**: Refers to the intensity or purity of the color. High saturation means vibrant color, while low saturation approaches grayscale.
* **Lightness**: Controls the brightness of the color. A value of 0 gives black, 1 gives white, and values in between represent shades.

To convert an RGB image to HSL, we transform the RGB values into a representation where adjusting the lightness channel directly affects the brightness without altering the color’s hue or intensity. This makes HSL particularly useful for tasks like contrast enhancement.

**Instructions**

**Task 1: RGB Image Contrast Enhancement**

1. **Load an RGB Image**
   * Load a sample RGB image provided in the lab resources or any high-quality RGB image you have.
   * Use skimage.io to read and display the image.
   * Display the image to understand its initial contrast and color distribution.
2. **Enhancing Contrast by RGB Channels**
   * Separate the image into its Red, Green, and Blue channels.
   * For each channel, apply a contrast enhancement technique, such as:
     + **Histogram Equalization**: Redistributes the intensities to cover a wider range, enhancing the contrast.
     + **Contrast Stretching**: Maps the pixel values to span a specified range, improving visibility.
     + You can use functions from skimage.exposure like equalize\_hist or rescale\_intensity.
   * Merge the enhanced channels back into an RGB image using numpy or similar methods.
   * Display the original and enhanced images side by side for comparison.

**Task 2: HSL-Based Contrast Enhancement**

1. **Convert RGB to HSL**
   * Convert the RGB image to HSL using skimage.color.rgb2hsv. Note that in skimage, the HSV color space is similar to HSL, where V (Value) is comparable to Lightness in HSL.
   * Extract the Lightness (L) channel. This channel affects brightness independently, allowing for contrast manipulation without altering the color hue or saturation.
2. **Enhance the Lightness Channel**
   * Apply histogram equalization or contrast stretching to the Lightness channel to enhance contrast.
   * Re-assemble the HSL image with the enhanced Lightness channel.
   * Convert the modified HSL image back to RGB using skimage.color.hsv2rgb.
   * Display the original and HSL-enhanced images side by side.

**Task 3: Additional Image Processing (Choose Any Two)**

1. **Sharpening**
   * Sharpening emphasizes edges, making the image appear clearer. Use an unsharp mask (skimage.filters.unsharp\_mask) to enhance edges.
   * Display the original and sharpened images for comparison.
2. **Blurring**
   * Apply a blurring filter to reduce noise or smooth details. Use Gaussian (skimage.filters.gaussian) or median filtering to achieve this.
   * Experiment with different filter parameters and observe the effect on the image.
3. **Color Channel Adjustment**
   * Modify each RGB channel intensity independently to see its effect on the image's overall tone.
   * For instance, increase the red channel intensity to make the image warmer, or reduce the blue channel to lessen cool tones.
4. **Edge Detection**
   * Edge detection highlights boundaries in the image. Use edge-detection methods like Sobel (skimage.filters.sobel) or Canny (skimage.feature.canny).
   * Display the original and edge-detected images for comparison.

**Detailed Steps for Code Implementation**

1. **Load Image**: Use skimage.io.imread and skimage.io.imshow for loading and displaying the image.
2. **Separate Channels (Task 1)**: Use numpy indexing to isolate R, G, and B channels, apply skimage.exposure functions on each, and then combine channels back using numpy.dstack.
3. **HSL Conversion (Task 2)**: Use skimage.color.rgb2hsv to convert to HSL/HSV, apply enhancement on the Lightness, then use skimage.color.hsv2rgb to convert back.
4. **Display Comparison**: Display results using matplotlib.pyplot to visualize side-by-side comparisons.

Provide an explanation of your code in your own words. This is to ensure that you have a deep understanding of the code you've written and its underlying concepts. You are expected to comments on the main parts and functions of the code.

Guidelines:

* Your explanation should be original and in your own words. Do not copy explanations from textbooks, online resources, or peers.
* Go beyond just describing what the code does. Explain why you chose certain methods or approaches and how they benefit the solution.
* Document your usage of generative AI if any as per the announced policy (see syllabus).

**Assessment**

1. Each student will show all the above parts running as demo to the Lab Instructor **before leaving the lab.** Total marks for the lab is as follows

|  |  |
| --- | --- |
| Task(s) | Marks (demo + report) |
| Task 1 | 4 |
| Task 2 | 4 |
| Task 3 | 2 |
| Total | 10 |

1. Students will prepare a report in which they will submit the snapshots taken while they worked on each part. They will explain the figures to make sure that they understood what they did.