Individual project-Based Image Processing Assignment 03
Color Transfer Between Images

S.Y.D.Senadheera
Reg No: 21/ENG/128
Department of Electrical and Electronic Engineering
Faculty of Engineering
University of Sri Jayawardenapura

***Abstract*—This project studies using Python to apply color transfer techniques to improve the visual aesthetics of photographs. The objective of this project is to generate visually pleasing and cohesive outcomes by applying an innovative process that makes it easier to transfer color attributes from a target image to five source images at once.The technique first analyzes the color distributions of the source and target photos. Then, mathematical adjustments are applied. Color integrity and artistic quality are assessed on the resultant images, showcasing color transfer's potential in digital art, photography, and film, among other industries. The project finishes with saved outputs of the altered photos, offering a thorough foundation for other image processing applications in the future.**

1. Introduction

An essential component of image processing is color manipulation, which is frequently used to remove unwanted color casts from photos, such as the yellow tones that are frequently present in photos taken in incandescent lighting. By transferring the color attributes of one image to another—for example, by applying the rich hues of a sunset photograph to a daylight rendering—this study investigates a revolutionary approach to color correction as shown like in figure 2 .

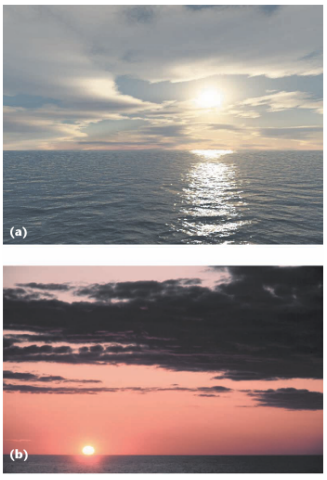

Figure 1. Color transfer technique application

Choosing a color space that minimizes correlations between color channels is essential to achieving good color transfer. The l alpha beta color space, which was created by Ruderman et al., shows promise in this situation because of its orthogonal characteristics and compatibility with human vision. The objective of this study is to utilize the benefits of the l alpha beta space for smooth color changes, improving the visual coherence of processed images. . [1] so here in this project it aims to do the color transfer between images which is very important in image processing application and thse are mainly used in photo editing and film making indsutry as the color quality is necssary for that.

Figure 2. Color transfer technique application

Our main tactic is to select an appropriate color space and then perform basic operations there in order to achieve our goal of doing so using a straightforward algorithm. There will be correlations between the values of the various channels when a typical three channel image is represented in any of the most widely used color spaces. For instance, if the blue channel is large in RGB space, the majority of pixels will have large values for the red and green channels. This suggests that altering one color channel at a time is

necessary to alter all color channels simultaneously in order to change a pixel's look in a consistent manner. This makes the process of changing a hue more difficult. An orthogonal color space devoid of axis correlations is what we are after.

1.1. Why BMP is used

An uncompressed digital image can be stored in a bitmap file (BMP), which is a raster image format that retains all of the pixel data without any loss. Because of this, BMP is perfect for applications since it guarantees high-quality images and accurate color representation. Because they're easy to use and lossless, BMP (Bitmap) files are frequently utilized in image processing applications. BMP files are perfect for activities requiring exact color manipulation and analysis, in contrast to JPG or JPEG formats, which use lossy compression that might reduce image quality. It is especially helpful in color transfer situations when it's important to preserve the original color integrity of the source image. It is required to convert JPEG or JPEG photos to BMP format prior to processing if working with them, as they are more suited for storage.

If we have jpg or JPEG it willl not be a problem as we can convert easily into BMP format. I also made a python program which make it easy to do so.

```
PS C:\Acedemics\EE3253 Introduction to Image Processing\Project\Covert to BMP> python convert_to_BMP.py bird.jpg
Image converted to BMP format and saved as bird.bmp
```

Figure 3. python output of converting result

```python
from PIL import Image
import os

def convert_to_bmp(input_file, output_file):
    # Open the input image
    with Image.open(input_file) as img:
        # Convert the image to BMP format and save it
        img.save(output_file, 'BMP')

if __name__ == '__main__':
    import argparse

    # Set up argument parser
    parser = argparse.ArgumentParser(description='Convert JPG/JPEG images to BMP format.')
    parser.add_argument('input_file', help=r'C:\Acedemics\EE3253 Introduction to Image Processing\Project\Covert to BMP')
    parser.add_argument('output_file', nargs='?', help=r'C:\Acedemics\EE3253 Introduction to Image Processing\Project\Covert to BMP')

    args = parser.parse_args()

    # Get the output file path
    if not args.output_file:
        # If no output file path is provided, use the input file name with .bmp extension
        base_name, _ = os.path.splitext(args.input_file)
        output_file = f'{base_name}.bmp'
    else:
        output_file = args.output_file

    # Convert the image
    try:
        convert_to_bmp(args.input_file, output_file)
        print(f'Image converted to BMP format and saved as {output_file}')
    except FileNotFoundError:
        print(f'Error: The file {args.input_file} does not exist.')
    except Exception as e:
        print(f'An error occurred: {e}')
```

Listing 1. Convert to BMP

1.2. Background

Color transfer techniques have various applications that are essential for ensuring color consistency and enhancing the visual quality of images across different domains use in the current world

- Photography:
  Adjusting colors to achieve specific moods or artistic effects. Correcting color casts caused by lighting conditions. Enhancing photos to make them visually appealing and consistent with a desired aesthetic
- Film and Animation:
  Ensuring consistent color grading across scenes. Maintaining harmony in pictures and situations by using uniform color grading. producing noticeable outcomes and establishing the atmosphere or tone for specific scenes. emulating the color scheme of a specific time period, place, or scene.
- Digital Art:
  Allowing artists to apply color styles from one artwork to another. Assisting in the restoration of old or damaged artworks by matching colors to a reference
- Scientific Visualization:
  Applying color maps to data visualizations to highlight patterns, trends, and anomalies. Enhancing the clarity and interpretability of complex datasets.

1.3. Used Image processing teqniques

- Color Space Conversion:
  The conversion of images from the RGB color space to the LAB color space. LAB color space separates luminance (lightness) from chrominance (color information), which allows for more effective and perceptually accurate color manipulation.
- Statistical Analysis:
  Calculation of the mean and standard deviation for each color channel (L, A, B) in both the source and target images. These statistics provide insight into the color distribution and are essential for the subsequent adjustment steps.

- Pixel-wise Adjustment:
  Each pixel in the source image is adjusted based on the mean and standard deviation of the source and target images. The formula used typically aligns the source image's color distribution with that of the target image by scaling and shifting the pixel values.
- Clipping and Normalization:
  After adjustment, pixel values are clipped to ensure they fall within the valid range (0 to 255) for 8-bit images. This step prevents color overflow and underflow, which can lead to artifacts.
- Inverse Color Space Conversion:
  The adjusted image is converted back from the LAB color space to the RGB color space for final output. [2] This step ensures that the image can be displayed and used in standard RGB-compatible environments.
- Image Visualization:
  Visualization techniques such as displaying the source, target, and result images side by side using libraries like Matplotlib to assess the effectiveness of the color transfer.

1.4. Methodology:

1.4.1. Crucial actions make up the implementation:.

- Reading Pictures:
  In order to prepare them for processing, the source and target photos are read from their respective directories and transformed from BGR to LAB color space

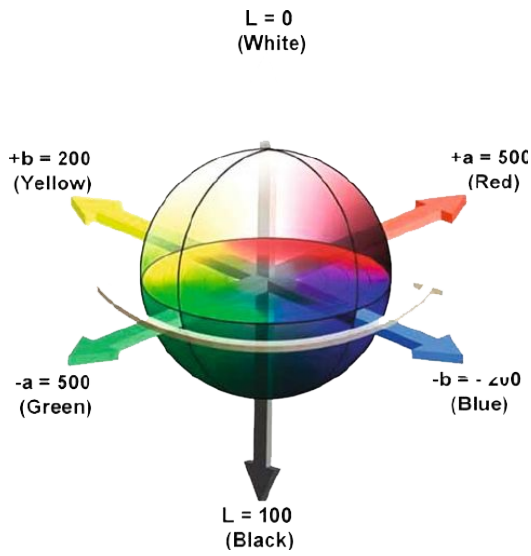

Figure 4. LAB color space

- Finding the Mean and Standard Deviation:
  For both the source and target images, the mean and standard deviation of each color channel (L, A, and B) are calculated. In order to modify the color distribution, these statistics are essential.

In the context of color images, we can compute the mean for each color channel (e.g., red, green, and blue) separately. Mathematically, the mean of a set of values is given by below equation .The mean (average) of a set of values $x_1, x_2, \ldots, x_n$ is given by the formula:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \tag{1}$$

where n is the number of values, and x_i represents each individual value.

The standard deviation of a set of values $x_1, x_2, \ldots, x_N$ is given by the formula:

$$\text{Standard Deviation} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x_i - \text{Mean})^2} \tag{2}$$

When we impose mean and standard deviation onto data points in a color image, we aim to match the statistical properties of the image to a desired target. To impose mean and standard deviation, we normalize the color channels of an image The normalized value of a data point x_i is given by the formula:

$$\text{Normalized Value} = \frac{x_i - \text{Mean}}{\text{Standard Deviation}} \tag{3}$$

- Color adjustment:
  Using the computed statistics as a basis, every pixel in the original image is changed. The formula guarantees that the target image's color distribution matches the source image's.
- Output Generation:
  After processing, the image is saved and reverted to BGR color space. Every image pair's results are saved for future comparison.
- Visualization:
  Matplotlib is used to display the output along with the original source and target images.

1.4.2. Datasets.

- Source images
- Target images

1.4.3. Libraries.

1) NumPy:
   - Use: For manipulating arrays and performing numerical calculations.
   - Installing –'pip install numpy'

2) OpenCV:

- Use: To read, write, and process pictures. OpenCV has routines to compute statistics like mean and standard deviation as well as convert between different color spaces.
- Installing –'pip install opencv-python

3) Tkinter:

- Use: For creating the graphical user interface (GUI) of the application.

4) Matplotlib:

- Use: For charting outcomes and showing images.
- Installing –'pip install matplotlib'

5) Threading:

- To run the image processing in a separate thread, keeping the UI responsive.

6) OS:

- Use: To perform operations on the operating system, like opening folders and listing files in directories.

7) Pyinstaller

- APython library that serves as a tool to convert Python scripts into standalone executables. It allows you to bundle a Python application and all its dependencies into a single package.

1.4.4. Tools.

- Python
- Tkinter GUI Elements
- File Dialogs
- Integrated Development Environment (IDE)-'VScode'
- Image Viewer
  For viewing the processed images. OpenCV's cv2.imshow function and the default image viewer of the operating system are used.
- Command Line or Terminal

1.5. Code

```
import numpy as np
import cv2
import os
from tkinter import filedialog, Tk, Button, Label, messagebox, DoubleVar
from tkinter.ttk import Progressbar, Style
from matplotlib import pyplot as plt
from threading import Thread
from tkinter import Tk, Text, END

#functions
def read_file(sn, tn):
    s = cv2.imread(sn)
    s = cv2.cvtColor(s, cv2.COLOR_BGR2LAB)
    t = cv2.imread(tn)
    t = cv2.cvtColor(t, cv2.COLOR_BGR2LAB)
    return s, t

def get_mean_and_std(x):
    x_mean, x_std = cv2.meanStdDev(x)
    x_mean = np.hstack(np.around(x_mean, 2))
    x_std = np.hstack(np.around(x_std, 2))
    return x_mean, x_std

def color_transfer(sources, targets, output_folder):
    results = []  # List to store the source, target, and result images for display

    for n in range(len(sources)):
        print("Converting picture " + str(n+1) + "...")
        s, t = read_file(sources[n], targets[n])
        s_mean, s_std = get_mean_and_std(s)
        t_mean, t_std = get_mean_and_std(t)

        height, width, channel = s.shape
        for i in range(height):
            for j in range(width):
                for k in range(channel):
                    x = s[i, j, k]
                    x = ((x - s_mean[k]) * (t_std[k] / s_std[k])) + t_mean[k]
                    x = round(x)
                    x = 0 if x < 0 else x
                    x = 255 if x > 255 else x
                    s[i, j, k] = x
        result = cv2.cvtColor(s, cv2.COLOR_LAB2BGR)
        result_path = os.path.join(output_folder, 'r' + str(n+1) + '.bmp')
        cv2.imwrite(result_path, result)

        # Store images for later display
        results.append((cv2.cvtColor(cv2.imread(sources[n]), cv2.COLOR_BGR2RGB),
                        cv2.cvtColor(cv2.imread(targets[n]), cv2.COLOR_BGR2RGB),
                        cv2.cvtColor(result, cv2.COLOR_BGR2RGB)))

        # Update the progress bar
        progress_var.set((n + 1) / len(sources) * 100)
        root.update_idletasks()

    return results

def show_images(results):
    for idx, (source, target, result) in enumerate(results):
        plt.figure(figsize=(15, 5))

        plt.subplot(1, 3, 1)
        plt.imshow(source)
        plt.title(f'Source Image {idx+1}')
        plt.axis('off')

        plt.subplot(1, 3, 2)
        plt.imshow(target)
```

```
        plt.title(f'Target Image {idx+1}')
        plt.axis('off')

        plt.subplot(1, 3, 3)
        plt.imshow(result)
        plt.title(f'Result Image {idx+1}')
        plt.axis('off')

        plt.tight_layout()
        plt.show()

def select_source_files():
    global source_files
    source_files = filedialog.askopenfilenames(title='Select Source Images', filetypes=[("Image files", "*.bmp;*.jpg;*.jpeg;*.png")])

    if not source_files:
        messagebox.showerror("Error", "No source files selected.")
    else:
        source_label.config(text=f"Selected {len(source_files)} source files")

def select_target_files():
    global target_files
    target_files = filedialog.askopenfilenames(title='Select Target Images', filetypes=[("Image files", "*.bmp;*.jpg;*.jpeg;*.png")])

    if not target_files:
        messagebox.showerror("Error", "No target files selected.")
    else:
        target_label.config(text=f"Selected {len(target_files)} target files")

def select_output_folder():
    global output_folder
    output_folder = filedialog.askdirectory(title='Select Output Folder')

    if not output_folder:
        messagebox.showerror("Error", "No output folder selected.")
    else:
        output_label.config(text=f"Output folder selected: {output_folder}")

def start_conversion():
    if not source_files or not target_files:
        messagebox.showerror("Error", "Please select source and target files first.")
        return

    if len(source_files) != len(target_files):
        messagebox.showerror("Error", "The number of source and target files must be the same.")
        return

    if not output_folder:
        messagebox.showerror("Error", "Please select an output folder.")
        return

    # Start the conversion in a separate thread to keep the UI responsive
    conversion_thread = Thread(target=run_conversion)
    conversion_thread.start()

def run_conversion():
    results = color_transfer(source_files, target_files, output_folder)
    show_images(results)

if __name__ == "__main__":
    source_files = []
    target_files = []
    output_folder = ""

    root = Tk()
    root.title("Color Transfer Application")

    style = Style()
    style.configure("TButton", font=("Arial", 12), width=20)
    style.configure("TLabel", font=("Arial", 12), foreground="blue")
    style.configure("TProgressbar", thickness=30)  # Increase the height of the progress bar

    # Create a Text widget
    text_widget = Text(root, height=2, width=40, font=("Arial", 16, "bold"), bd=0, bg=root.cget("bg"), wrap="word")
    text_widget.pack(pady=10)

    # Insert text with colored word "color"
    text_widget.insert(END, "Welcome to ", "color_black")
    text_widget.insert(END, "C", "color_red")
    text_widget.insert(END, "O", "color_green")
    text_widget.insert(END, "L", "color_blue")
    text_widget.insert(END, "O", "color_orange")
    text_widget.insert(END, "R", "color_purple")
    text_widget.insert(END, " transfer app", "color_black")

    # Add tags for colors
    #different colors to make it nice
    text_widget.tag_configure("color_black", foreground="black", font=("Arial", 16, "bold"))
    text_widget.tag_configure("color_red", foreground="red", font=("Arial", 20, "bold"))
    text_widget.tag_configure("color_green", foreground="green", font=("Arial", 20, "bold"))
    text_widget.tag_configure("color_blue", foreground="blue", font=("Arial", 20, "bold"))
    text_widget.tag_configure("color_orange", foreground="orange", font=("Arial", 20, "bold"))
    text_widget.tag_configure("color_purple", foreground="purple", font=("Arial", 20, "bold"))

    # Make the Text widget read-only
    text_widget.configure(state="disabled")

    # Center the text using padding
    text_widget.tag_add("center", "1.0", END)  # Add a tag to center the text
    text_widget.tag_configure("center", justify="center")
```

```
    #Source file selection
    Label(root, text="Select source and target
        images", font=("Arial", 12, "bold"), fg="
        green").pack(pady=10)
    Button(root, text="Select Source Files",
        command=select_source_files).pack(pady=10)
    source_label = Label(root, text="", font=("
        Arial", 12), fg="red")
    source_label.pack(pady=5)

    #Target file selection
    Button(root, text="Select Target Files",
        command=select_target_files).pack(pady=10)
    target_label = Label(root, text="", font=("
        Arial", 12), fg="red")
    target_label.pack(pady=5)
    Label(root, text="", height=2).pack()

    #Output folder selection
    Label(root, text="Select where to save result
        images", font=("Arial", 12, "bold"), fg="
        green").pack(pady=10)
    Button(root, text="Select Output Folder",
        command=select_output_folder).pack(pady=10
        )
    output_label = Label(root, text="", font=("
        Arial", 12), fg="red")
    output_label.pack(pady=5)
    Label(root, text="", height=2).pack()

    #Button to Start the operation
    Button(root, text="Start", command=
        start_conversion, width=20, height=2,
       bg="red", fg="white", font=("Arial", 14, "
           bold")).pack(pady=10)

    #progress bar
    progress_var = DoubleVar()
    progress_bar = Progressbar(root, variable=
        progress_var, maximum=100, style="
        TProgressbar")
    progress_bar.pack(pady=10, padx=20, fill="x")

    # Add credentials at the bottom
    Label(root, text="Yashmika Senadheera\n21/ENG/
        128", font=("Arial", 10), fg="purple").
        pack(pady=20)

    root.mainloop()
```

Listing 2. Python Script for my project

1.6. Application software

I made an application which ask for source files,target files and also the location to save the converted images. there is an human friendly interface which anyone can easily do there color transfer requirements to images.

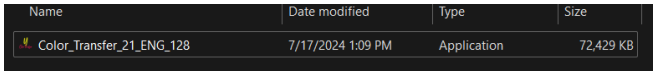

Figure 5. application details

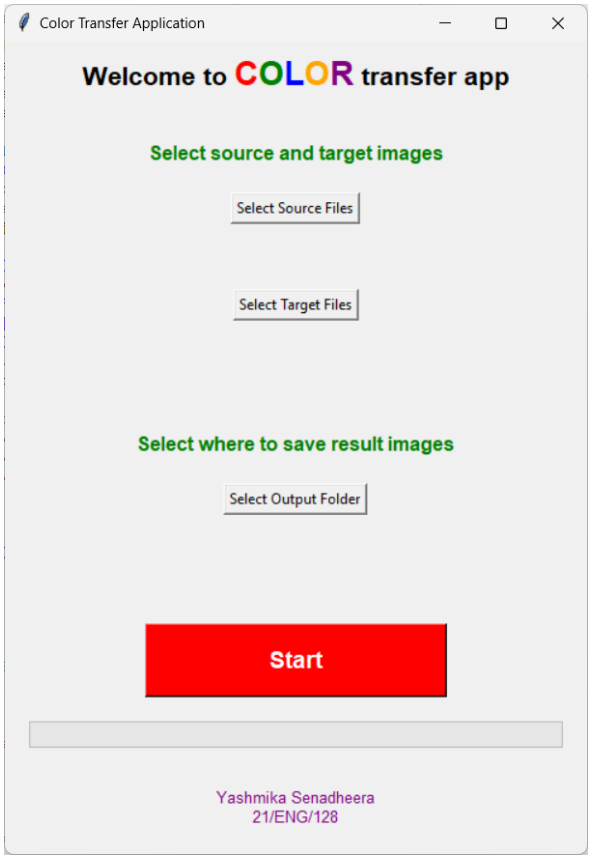

Figure 6. User interface of the app

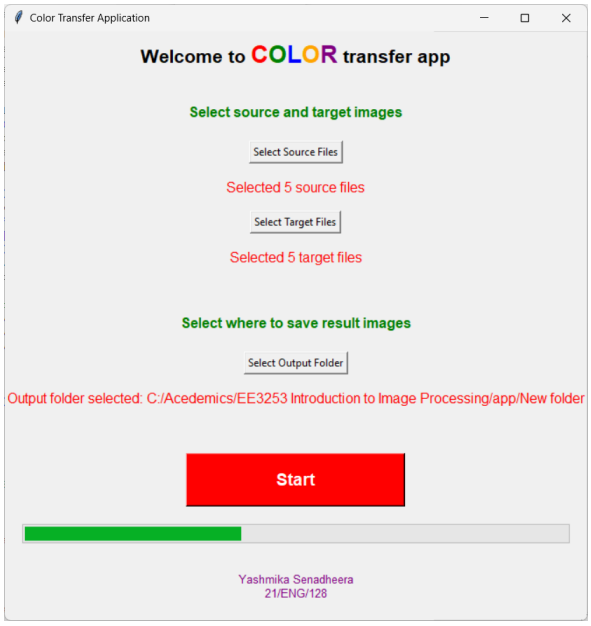

Figure 7. processing interface of the app

1.7. Results

from the programme below results can be obtained and save the image inside the folder for later use.

- Source Image: The original image with its inherent color characteristics.
- Target Image: The image whose color characteristics are to be transferred.
- Result Image: The source image with colors adapted to match the target image.

The color palettes of the source and target images are successfully adjusted by the color transfer algorithm. This conversion imparts the color tone and mood of the target photos while preserving the general composition and level of detail of the source photographs.

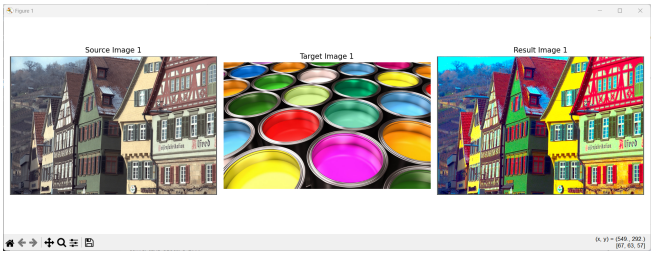

Figure 8. example converted output 1

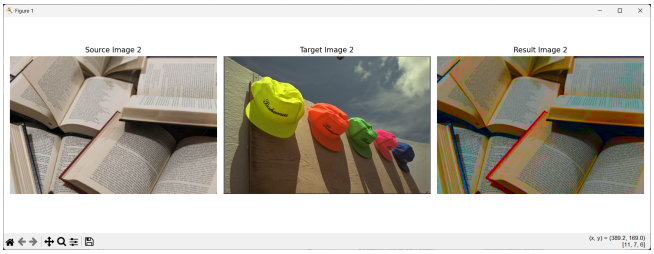

Figure 9. example converted output 2

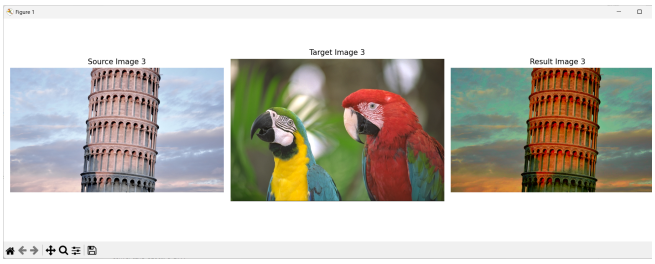

Figure 10. example converted output 3

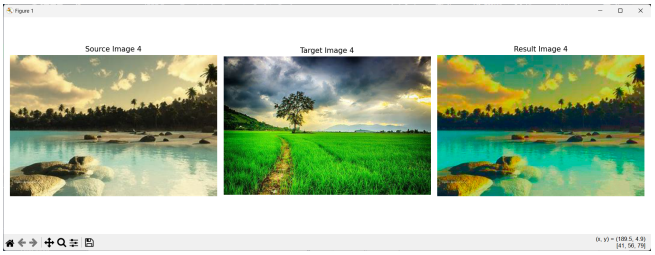

Figure 11. example converted output 4

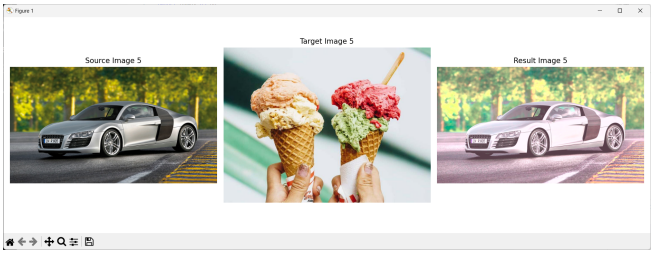

Figure 12. example converted output 5

1.8. Limitations

Although the approach is successful, there are a few drawbacks:

- Lighting: Different lighting setups can have an impact on color accuracy.
- Image Content: When intricate textures are present, unexpected outcomes can happen
- File format; Handling different image formats (BMP, JPG, JPEG) and converting them appropriately.

1.9. Future Plans

Potential improvements in the future might be:
applying cutting-edge methods like color transfer based on deep learning. investigating real-time video processing applications.

Real-Time Processing:Develop real-time color transfer capabilities for live video processing. Apply real-time processing techniques for applications in film production, live streaming, and virtual reality.

2. Conclusion

This article shows that one effective tool for working with color images is a color space [3] with decorrelated axes. Applying the mean and standard deviation to the data points is an easy process that, with appropriate input photos, provides believable output images. Applications for this technique range from modest image postprocessing to enhance appearance to more dramatic changes, such turning an image taken during the day into a scene at night. We can limit the color choices to hues that are most likely to appear

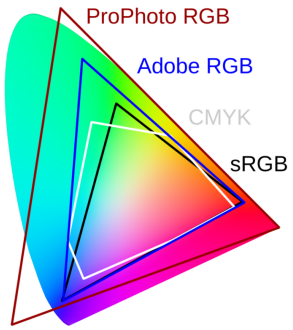

Figure 13. Color space

in nature by using the measurement of color spectrum in photos. Because of this method's simplicity, we may use it as a plug-in for a variety of commercial graphics packages. Lastly, we predict that l alpha beta color space will be successfully applied by researchers to additional problems like color quantization.

Acknowledgments

I would like to thank Erik Reinhard, Michael Ashikhmin, Bruce Gooch and Peter Shirley who done implementation of the paper Color Transfer between Images and everyone who supported to improve and make a working application which is important for many fields. Finally, I would like to acknowledge the authors and developers of the various Python libraries and tools that were utilized in this project, including OpenCV, NumPy, and Matplotlib. Their contributions to the open-source community have been invaluable

References

[1] E. Reinhard, M. Adhikhmin, B. Gooch, and P. Shirley, "Color transfer between images," *IEEE Computer graphics and applications*, vol. 21, no. 5, pp. 34–41, 2001.

[2] L. Busin, N. Vandenbroucke, and L. Macaire, "Color spaces and image segmentation," *Advances in imaging and electron physics*, vol. 151, no. 1, p. 1, 2008.

[3] E. Reinhard and T. Pouli, "Colour spaces for colour transfer," in *Computational Color Imaging: Third International Workshop, CCIW 2011, Milan, Italy, April 20-21, 2011. Proceedings 3*. Springer, 2011, pp. 1–15.