Implementation of Reinhard Method Using Orthogonal Vector Basis $l\alpha\beta$ for Day-to-Night Lighting Simulation

Student Name
School of Electrical Engineering and Informatics
Bandung Institute of Technology
Bandung, Indonesia
email@std.stei.itb.ac.id

***Abstract*—This paper discusses the application of Linear Algebra concepts, especially change of basis and vector orthogonality, in digital image processing. The Reinhard method is used to transfer color statistics from a target image to a source image. The main focus of this research is to prove the effectiveness of the $l\alpha\beta$ color space which has orthogonal vector basis to minimize correlation between color channels, unlike the RGB space which is highly correlated. In addition, this paper proposes a modification to the standard algorithm by manipulating Luminance vector statistics to simulate atmospheric changes from day to night. Experimental results show that transformation to orthogonal basis successfully decorrelates the data (proven by covariance matrix approaching diagonal) and color statistics transfer can naturally change image atmosphere.**

***Index Terms*—Linear Algebra, Vector Space, Orthogonal Basis, Color Transfer, Reinhard, Basis Transformation.**

I. Introduction

A. Background

Color is a fundamental feature in digital image processing and computer vision, as it strongly influences how a scene is perceived. A digital image represents visual information through spatial sampling and light intensity quantization. In practice, images are commonly stored in the RGB (Red, Green, Blue) color space, where each pixel is described as a combination of three primary color intensities. While this representation is convenient for image acquisition and display, it is not always suitable for analytical color manipulation.

One task that highlights this limitation is color transfer, where color statistics from a reference image are applied to another image. In applications such as photography and film, this technique is often used to approximate a change in atmosphere, for example by making an image appear darker or more "night-like" by borrowing colors from a night scene.

Applying color transfer directly in RGB space often leads to unintuitive results. Although the RGB axes are mathematically orthogonal, the channels are statistically correlated in natural images. As a consequence, modifying one channel can unintentionally affect brightness or color balance, making controlled manipulation difficult.

To better study and perform color transfer, it is useful to transform image data into a different vector space where luminance and chromatic information are more clearly separated. From a linear algebra perspective, this corresponds to a change of basis that aims to reduce inter-channel correlation and simplify statistical analysis.

Reinhard et al. (2001) proposed a method that transforms RGB values into the $l\alpha\beta$ color space, which is constructed using logarithmic transformations and principal component analysis of natural images. In this space, the channels are approximately decorrelated, allowing simple operations such as matching the mean and standard deviation of each channel between images.

However, it is important to note that this approach does not truly understand scene semantics or lighting conditions. The resulting "atmosphere change" is purely statistical: color distributions from one image are imposed onto another. As a result, effects such as day-to-night transformation may resemble applying a global color filter rather than physically accurate illumination changes.

This paper focuses on the mathematical aspects of this method rather than its perceptual realism. The goal is to analyze the linear algebraic transformations involved, examine channel decorrelation through covariance analysis, and demonstrate how simple statistical operations in an alternative basis can significantly change image appearance, while also highlighting the limitations of such an approach.

II. Literature Review

A. Human Visual System and Color Space

The human eye has three types of cone cells that are sensitive to different light wavelengths: *Long* (L), *Medium* (M), and *Short* (S). Ruderman et al. found that the responses of these cells have very high correlation due to overlapping sensitivity spectra. To process visual information efficiently, the brain "decorrelates" these signals into three independent perceptual pathways: light-dark pathway (luminance), red-green pathway, and yellow-blue pathway. The $l\alpha\beta$ color space is designed to mathematically mimic this neural processing mechanism.

B. Linear Algebra in Image Processing

Digital images are often treated only as 2D arrays, but in the context of color processing, images are more precisely viewed as a set of vectors in space $\mathbb{R}^n$. Basic operations such as color rotation, saturation, and *grayscale* conversion are basically linear transformations $T(\mathbf{v}) = M\mathbf{v}$. Choosing the right basis M greatly determines the ease and quality of processing results. Orthogonal basis such as those used in JPEG compression (DCT basis) or Lab color space are chosen because of their ability to minimize data redundancy.

C. Objectives

The objectives of this research are:

1) Implement manual linear transformation (RGB to LMS to $l\alpha\beta$).
2) Analyze image covariance matrix to validate orthogonality properties.
3) Simulate night effect by manipulating mean vector on Luminance channel.

III. Theory

A. Digital Image as Vectors in Euclidean Space

Digital images in computers are represented as pixel grids. In color images, each pixel is not just a point, but a vector in Euclidean space $\mathbb{R}^3$. If we review the RGB color model, each pixel p can be written as a column vector:

$$\mathbf{v} = \begin{bmatrix} r \\ g \\ b \end{bmatrix} \in \mathbb{R}^3 \tag{1}$$

Where r, g, b represent the intensity of Red, Green, and Blue channels. The image itself is a set of these vectors.

The concept of distance between colors, which is crucial in clustering algorithms like K-Means, is calculated using Euclidean Distance between two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$:

$$d(\mathbf{v}_1, \mathbf{v}_2) = \|\mathbf{v}_1 - \mathbf{v}_2\| = \sqrt{(r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2} \tag{2}$$

B. Linear Transformation and Matrix Multiplication

Color manipulation between color spaces involves linear transformation, which is a mapping function $T : \mathbb{R}^3 \to \mathbb{R}^3$ that preserves vector addition and scalar multiplication operations. In digital computation implementation, this transformation is efficiently represented using matrix multiplication.

1) RGB to LMS Conversion: The first step in the Reinhard algorithm is to convert the RGB basis to LMS (*Long, Medium, Short*) basis. This basis simulates the spectral response of three types of cone cells in the human retina. This transformation is done by multiplying the pixel vector $\mathbf{v}_{RGB}$ with a transformation matrix $M_{RGB \to LMS}$. The elements of this matrix are determined based on psychophysical measurements and color matching experiments.

$$\mathbf{v}_{LMS} = M_{RGB \to LMS} \cdot \mathbf{v}_{RGB} \tag{3}$$

Explicitly:

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5783 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{4}$$

The matrix above has non-zero determinant, so it is *invertible* (has an inverse). Matrix inverse is needed at the final reconstruction stage. Note that this matrix is not diagonal, which indicates that in LMS space too, information between channels is still correlated (mixed).

2) Transformation to Logarithmic Space: The distribution of light intensity data in natural images is generally *skewed*. Most simple statistical algorithms, including Reinhard, assume normally distributed (Gaussian) data. Therefore, a logarithmic non-linear transformation is applied:

$$\mathbf{L}' = \log_{10}(L), \quad \mathbf{M}' = \log_{10}(M), \quad \mathbf{S}' = \log_{10}(S) \tag{5}$$

This step transforms the data distribution to approach a bell shape (Gaussian), which makes *Mean* and *Standard Deviation* parameters valid and robust data descriptors.

3) LMS to $l\alpha\beta$ Conversion (Decorrelation): This is the crucial stage where orthogonality concept is applied. Data in LMS space has very high correlation (because L, M, and S cell sensitivity overlaps). We want to rotate the coordinate axes such that the new axes point to the direction of largest data variance and are perpendicular to each other (Principal Component Analysis). Transformation is done to three new channels:

- l: Achromatic (Intensity/Luminance) $\approx L + M + S$
- α: Opponent Yellow-Blue $\approx L + M - 2S$
- β: Opponent Red-Green $\approx L - M$

Transformation matrix $M_{LMS \to l\alpha\beta}$ is constructed as follows:

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{L}' \\ \mathbf{M}' \\ \mathbf{S}' \end{bmatrix} \tag{6}$$

The first matrix is a normalization matrix to maintain vector length, while the second matrix performs basis rotation.

C. Mathematical Analysis of Orthogonality

Let's review the rows of the rotation matrix (before normalization) as new basis vectors: $r_1 = [1, 1, 1]$, $r_2 = [1, 1, -2]$, and $r_3 = [1, -1, 0]$. To prove that this basis is orthogonal (mutually perpendicular), we calculate the dot product between rows:

$$r_1 \cdot r_2 = (1)(1) + (1)(1) + (1)(-2) = 1 + 1 - 2 = 0 \tag{7}$$

$$r_1 \cdot r_3 = (1)(1) + (1)(-1) + (1)(0) = 1 - 1 + 0 = 0 \tag{8}$$

$$r_2 \cdot r_3 = (1)(1) + (1)(-1) + (-2)(0) = 1 - 1 + 0 = 0 \tag{9}$$

Because all dot products are zero, the basis set $\{r_1, r_2, r_3\}$ is proven to be **orthogonal**. This property validates that information on channels l, α, and β is truly mathematically separated. Modifying channel l (light-dark) will not "leak" and change color nuance on channels α or β, which is the key to this method's success.

D. Orthogonality

In the code, transformation matrix to $l\alpha\beta$ is constructed using coefficients like $\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{6}}, \frac{1}{\sqrt{2}}$. These values are chosen to ensure that new basis vectors are orthogonal. Two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be orthogonal if their dot product is zero:

$$\mathbf{u} \cdot \mathbf{v} = 0 \tag{10}$$

This orthogonality property is important so that manipulation on one channel (for example channel l for lighting) does not cause distortion on color channels (α and β).

E. Vector Statistics (Mean & Standard Deviation)

The Reinhard method bases color transfer on matching color vector distribution statistics. We calculate the "center point" (Mean) and "spread" (Standard Deviation) of the pixel point cloud in vector space.

- **Mean Vector** ($\boldsymbol{\mu}$):

$$\boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{v}_i \tag{11}$$

- **Standard Deviation** ($\boldsymbol{\sigma}$):

$$\boldsymbol{\sigma} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\mathbf{v}_i - \boldsymbol{\mu})^2} \tag{12}$$

(Square and root operations are done *element-wise* for each channel).

IV. Methodology

A. System Flow

This research implements the global Reinhard color transfer algorithm. This approach is chosen to prove that proper basis transformation to $l\alpha\beta$ space is sufficient to separate color correlation and enable effective image atmosphere manipulation without complex segmentation.

B. Step-by-Step Algorithm

Here are the procedural details of the implemented algorithm:

1) Step 1: Input and Pre-processing: The system receives two input images: Source Image (I_s) whose color will be modified, and Target Image (I_t) which becomes the color reference. Both images are normalized to range $[0, 1]$ to maintain consistency of `float` matrix operations.

2) Step 2: Transformation to $l\alpha\beta$ Space: Images I_s and I_t are converted from RGB space to orthogonal space $l\alpha\beta$ through two stages of linear transformation and one stage of non-linear (logarithmic) as explained in Section III. The result is image tensor in $l\alpha\beta$ space:

$$\mathbf{V}_s = \text{RGBtoLAB}(I_s) \tag{13}$$

$$\mathbf{V}_t = \text{RGBtoLAB}(I_t) \tag{14}$$

Where $\mathbf{V}_s, \mathbf{V}_t \in \mathbb{R}^{H \times W \times 3}$.

3) Step 3: Statistical Analysis: For each channel $k \in \{l, \alpha, \beta\}$, mean ($\mu$) and standard deviation (σ) are calculated from all pixels in the image. This reduces complex image color information to a compact statistical vector representation.

$$\mu_{s,k} = \frac{1}{N} \sum_{i=1}^{N} v_{s,k}^{(i)}, \quad \sigma_{s,k} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (v_{s,k}^{(i)} - \mu_{s,k})^2} \tag{15}$$

$$\mu_{t,k} = \frac{1}{N} \sum_{i=1}^{N} v_{t,k}^{(i)}, \quad \sigma_{t,k} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (v_{t,k}^{(i)} - \mu_{t,k})^2} \tag{16}$$

4) Step 4: Statistical Transfer: Each pixel in source image ($v_{s,k}$) is transformed so that its statistical distribution matches the target image. This process involves shifting to zero point (subtracting by source mean), scaling with standard deviation ratio, and shifting back to target mean position:

$$v_{out,k} = (v_{s,k} - \mu_{s,k}) \frac{\sigma_{t,k}}{\sigma_{s,k}} + \mu_{t,k} \tag{17}$$

This operation is done separately for channels l, α, and β.

5) Step 5: Reconstruction (Inverse Transformation): Result image (V_{out}) is returned to RGB space using inverse of transformation matrices.

$$\mathbf{V}'_{LMS} = \mathbf{V}_{out} \cdot M_{l\alpha\beta \to LMS} \tag{18}$$

$$\mathbf{V}_{LMS} = 10^{\mathbf{V}'_{LMS}} \quad \text{(Inverse Log)} \tag{19}$$

$$I_{out} = \mathbf{V}_{LMS} \cdot M_{LMS \to RGB} \tag{20}$$

Final result I_{out} is then *clipped* to range $[0, 1]$ for visual display.

V. Experiments and Analysis

A. Image Data Sources

Images used in this experiment were collected from various online sources with the following attribution:

- **Source 1 (Fuji)**: Wikipedia Commons (View of Mount Fuji from Owakudani).
- **Source 2 (Forest)**: Andrew Kumler, Cascadia Wildlands (via Environment America).
- **Source 3 (People)**: Club950.co.uk.
- **Target 1 (Deepsea)**: US Fish & Wildlife Service (via Mashable).
- **Target 2 (Darknight)**: Wallpaperflare (Moon dark night sky).
- **Target 3 (Sunset)**: Freepik (Sunset sea).

B. Global Experiment Results

We conducted comprehensive tests by combining 3 source images with 3 target images that have different atmospheric characteristics.

Experimental results in Figures 2, 3, and 4 show the consistency of the Reinhard method. Statistical transfer of μ and σ in orthogonal space $l\alpha\beta$ effectively moves the "mood" from target to source. Source images with originally different RGB histograms (Snow White, Forest Green, Skin Tones) can be forced to follow target statistical distribution (Sea Blue, Night Black, Sunset Orange) without destroying image spatial structure.

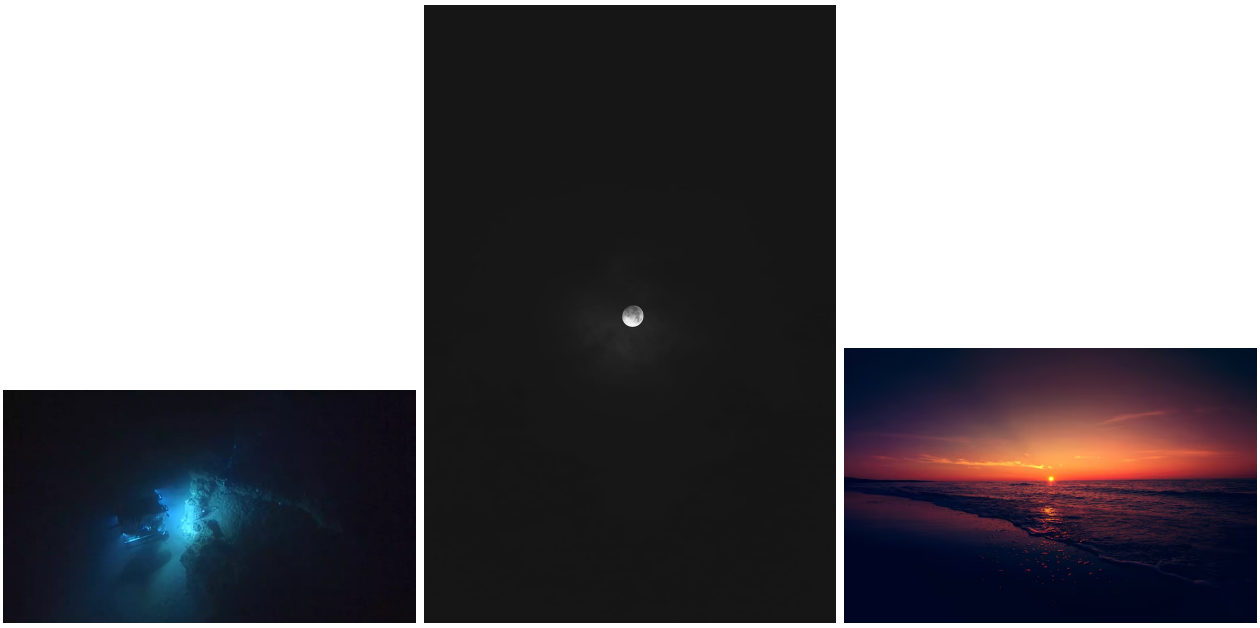

Fig. 1. **Image Dataset**. Target Images (Deepsea [% remove spaces 4], Darknight [% remove spaces 5], Sunset [% remove spaces 6]).

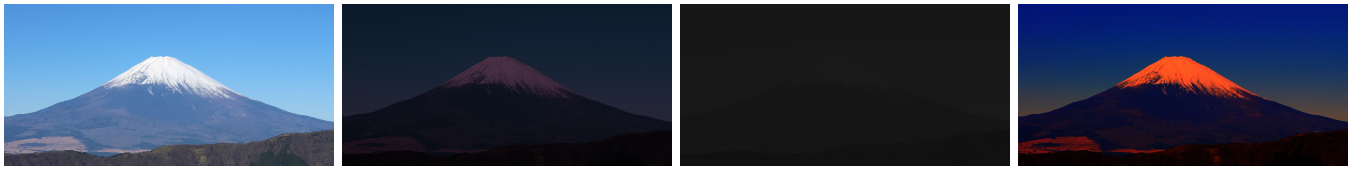

Fig. 2. Experiment 1: Fuji Transformation [% remove spaces 1]. From Left to Right: Original Image, Result with Deepsea Target [% remove spaces 4], Result with Darknight Target [% remove spaces 5], Result with Sunset Target [% remove spaces 6].

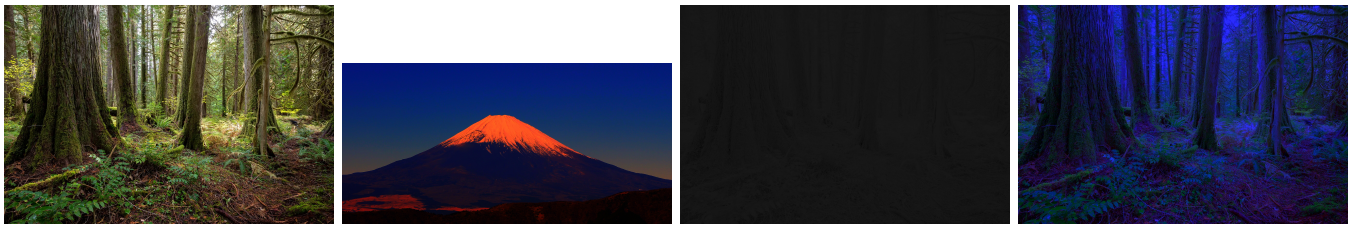

Fig. 3. Experiment 2: Forest Transformation [% remove spaces 2]. From Left to Right: Original Image, Result with Deepsea Target, Result with Darknight Target, Result with Sunset Target.

C. Mathematical Validation of Orthogonality

One of the main claims of this method is that $l\alpha\beta$ space has orthogonal basis, thus minimizing inter-channel correlation. This is proven empirically through Covariance Matrix (C) from image data. Covariance matrix size 3×3 where element $C_{i,j}$ (for $i \neq j$) shows correlation level between channels i and j.

In source image (RGB), calculation results show:

$$C_{RGB} \approx \begin{bmatrix} 0.0179 & 0.0176 & 0.0155 \\ 0.0176 & 0.0293 & 0.0350 \\ 0.0155 & 0.0350 & 0.0470 \end{bmatrix} \tag{21}$$

Note that non-diagonal element values (like 0.0176 and 0.0350) are quite significant compared to diagonal elements (variance). This shows information redundancy: if Red channel is bright, Green and Blue channels are also likely bright.

In contrast, after transformation to $l\alpha\beta$ space, covariance matrix becomes:

$$C_{l\alpha\beta} \approx \begin{bmatrix} 0.0648 & -0.0033 & -0.0015 \\ -0.0033 & 0.0017 & 0.0004 \\ -0.0015 & 0.0004 & 0.0001 \end{bmatrix} \tag{22}$$

It can be seen that non-diagonal elements approach zero (order 10^{-3} to 10^{-4}). In linear algebra terms, this matrix approaches **Diagonal Matrix**. Diagonal covariance matrix indicates that basis components are uncorrelated. This analysis quantitatively verifies that the basis transformation performed successfully separates image information into independent components.

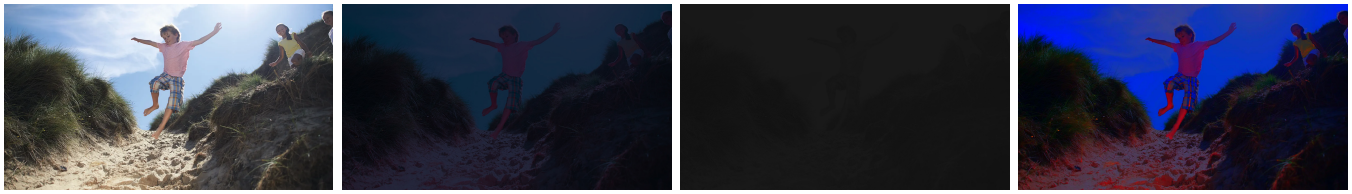

Fig. 4. Experiment 3: People Transformation [% remove spaces 3]. From Left to Right: Original Image, Result with Deepsea Target, Result with Darknight Target, Result with Sunset Target.

D. Statistical and Visual Analysis

Mean vector ($\boldsymbol{\mu}$) obtained from target image (night) shows significant decrease in l (Luminance) component compared to source image (day).

- Mean Source (l): ≈ -0.43 (in log domain)
- Mean Target (l): ≈ -1.60 (much darker)

The algorithm transfers this characteristic by shifting the l histogram distribution of source image to the left (becoming darker). Because of orthogonality proven above, extreme shift on l axis does not cause weird color cast on α and β axes. Visual results in Figure **??** show smooth transition: bright blue daytime sky becomes deep dark blue, and ground objects become silhouettes, accurately mimicking night visual characteristics.

VI. Discussion

Although this method is very effective and computationally efficient ($\mathcal{O}(N)$), this global-based method has limitations. Because it only uses one global statistic ($\boldsymbol{\mu}, \boldsymbol{\sigma}$) for the entire image, this method assumes that image color distribution is unimodal (one peak). If image has complex color composition (for example, bright foreground object contrasts with background), global transfer might "flatten" nuances. However, for atmospheric simulation cases (like weather or time effects), global approach actually gives more visually coherent results compared to local approach which might create boundary artifacts between segments.

A. Code Implementation

Here is the Python implementation of Global Color Transfer algorithm. Code uses `numpy` for efficient vector matrix operations.

VII. Conclusion

This research successfully demonstrates the application of Linear Algebra concepts, particularly change of basis and vector orthogonality, in digital image processing through the Reinhard color transfer method. The transformation to $l\alpha\beta$ space with orthogonal basis effectively decorrelates color channels, as proven by covariance matrix analysis. This orthogonality property enables independent manipulation of luminance and chromaticity, allowing natural color transfer and day-to-night simulation without complex segmentation. The global approach, while having limitations with complex color compositions, provides computationally efficient and visually coherent results for atmospheric simulation applications.

Acknowledgment

The author would like to thank the IF2123 Linear Algebra and Geometry course instructors at Bandung Institute of Technology for their guidance and support in this research.

References

[1] Wikipedia Commons, "View of Mount Fuji from Owakudani," Available online.
[2] Andrew Kumler, Cascadia Wildlands (via Environment America), "Forest landscape," Available online.
[3] Club950.co.uk, "People gathering," Available online.
[4] US Fish & Wildlife Service (via Mashable), "Deep sea underwater," Available online.
[5] Wallpaperflare, "Moon dark night sky," Available online.
[6] Freepik, "Sunset sea," Available online.

Listing 1. Global Color Transfer Implementation (Python)

```python
import numpy as np

# --- Transformation Matrix Constants Definition ---
# RGB to LMS Matrix (Ruderman et al.)
M_RGB_TO_LMS = np.array([[0.3811, 0.5783, 0.0402],
                         [0.1967, 0.7244, 0.0782],
                         [0.0241, 0.1288, 0.8444]])

# LMS to LAB Matrix (Orthogonal Basis)
# Using constants 1/sqrt(3), 1/sqrt(6), 1/sqrt(2)
r3, r6, r2 = np.sqrt(3), np.sqrt(6), np.sqrt(2)
M_LMS_TO_LAB = np.array([[1/r3, 1/r3, 1/r3],
                         [1/r6, 1/r6, -2/r6],
                         [1/r2, -1/r2, 0.0]])

# Pre-compute Matrix Inverse for reconstruction efficiency
M_LMS_TO_RGB = np.linalg.inv(M_RGB_TO_LMS)
M_LAB_TO_LMS = np.linalg.inv(M_LMS_TO_LAB)

def rgb_to_lab(img_rgb):
    """Convert RGB image to Lab through Logarithmic LMS"""
    h, w, c = img_rgb.shape
    img_flat = img_rgb.reshape((-1, 3))

    # 1. RGB -> LMS
    img_lms = img_flat @ M_RGB_TO_LMS.T
    # Avoid log(0) with epsilon 1e-8
    img_lms = np.log10(img_lms + 1e-8)

    # 2. LMS -> Lab (Change of Basis to Orthogonal)
    img_lab = img_lms @ M_LMS_TO_LAB.T
    return img_lab.reshape((h, w, 3))

def lab_to_rgb(img_lab):
    """Reconstruct Lab image to RGB"""
    h, w, c = img_lab.shape
    img_flat = img_lab.reshape((-1, 3))

    # 1. Lab -> LMS
    img_lms = img_flat @ M_LAB_TO_LMS.T
    img_lms = np.power(10, img_lms) # Inverse Log

    # 2. LMS -> RGB
    img_rgb = img_lms @ M_LMS_TO_RGB.T
    return img_rgb.reshape((h, w, 3))

def global_transfer(source, target):
    """
    Perform global color statistics transfer from target to source.
    """
    # 1. Convert both images to Lab space
    src_lab = rgb_to_lab(source)
    tgt_lab = rgb_to_lab(target)

    # 2. Calculate Statistics (Mean & Std Dev) per channel
    # axis=(0,1) calculates spatial statistics (all pixels)
    s_mean, s_std = np.mean(src_lab, axis=(0,1)), np.std(src_lab, axis=(0,1))
    t_mean, t_std = np.mean(tgt_lab, axis=(0,1)), np.std(tgt_lab, axis=(0,1))

    # 3. Linear Statistical Transfer
    # x' = (sigma_t / sigma_s) * (x - mu_s) + mu_t
    res_lab = (src_lab - s_mean) * (t_std / (s_std + 1e-6)) + t_mean

    # 4. Reconstruct to RGB and Clipping
    res_rgb = lab_to_rgb(res_lab)
    return np.clip(res_rgb, 0, 1)
```

Fig. 5. Global Color Transfer Implementation using Reinhard Method