Application of Linear Transformations for Color and Brightness Manipulation in Movie Poster Images

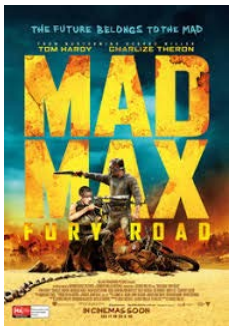
Mikhael Benrael Tampubolon 13524009[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524009@mahasiswa.itb.ac.id, [2]mikhaelbenrael@gmail.com

***Abstract*— Digital images play a central role in modern media, particularly in film promotion. Despite their visual complexity, images can be represented mathematically as structured numerical data. This paper examines how basic concepts of Linear Algebra can be applied to manipulate the visual properties of a movie poster image. Each pixel is modeled as a vector in three-dimensional Euclidean space R^3, corresponding to the Red, Green, and Blue (RGB) color channels. By treating the set of pixel colors as a vector space, common image processing operations such as brightness adjustment and color modification can be described as linear transformations. Brightness changes are achieved through scalar multiplication, while color filtering and channel mixing are implemented using matrix multiplication. The results show that fundamental Linear Algebra offers a simple yet effective framework for basic digital image manipulation.**

***Keywords*— Image Processing, Linear Transformation, RGB Color Model, Vector Space**

I. Introduction

In the digital era, visual information plays a central role in communication across entertainment, advertising, and interactive media. Digital images have become the dominant medium through which information and artistic expression are conveyed. In the film industry, movie posters are a particularly important form of visual media, serving not only as promotional material but also as a means of communicating the tone, genre, and emotional atmosphere of a film through carefully designed color schemes and visual contrast.



While a movie poster appears to humans as an artistic arrangement of shapes and colors, from a computational perspective it is fundamentally a structured collection of numerical data. Each image consists of a grid of pixels, where visual properties such as color and brightness are encoded numerically. This representation enables images to be processed and modified using mathematical tools. Although modern image editing systems may appear complex, their basic operations are grounded in Linear Algebra.

Linear Algebra provides a natural framework for modeling digital images. Color information can be represented as vectors in Euclidean space, and collections of pixels can be organized into structured arrays. Fundamental operations such as scalar multiplication and matrix multiplication, which are core topics in undergraduate Linear Algebra, can be directly interpreted as adjustments to image brightness and color composition.

This paper aims to connect abstract Linear Algebra concepts with practical computer science applications. Specifically, we use a movie poster image as a matrix of color vectors in $\mathbb{R}^3$ and apply linear transformations to manipulate its brightness and color balance. The scope of this work is limited to low-level pixel operations, focusing on how vectors in Euclidean space and linear transformations can be used to achieve modifications.

II. Theoretical Background

To perform mathematical manipulation of digital images, color data must be represented using precise algebraic objects. This requires the use of vectors in Euclidean space, vector spaces, and linear transformations.

A. Vectors in Euclidean Space

A vector in Euclidean space is defined as an ordered tuple of real numbers. The space $\mathbb{R}^n$ is given by

$$\mathbb{R}^n = \{(v1, v2, \dots, v_n) | \, v_i \in \mathbb{R}\}$$

An element $\boldsymbol{v} \in \mathbb{R}^n$ is written as

$$\boldsymbol{v} = (v1, v2, \dots, v_n)$$

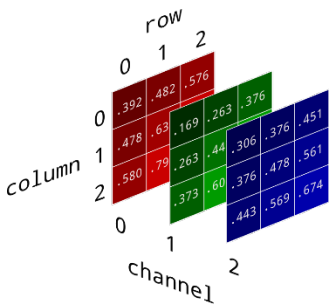
In the RGB color model, each pixel color is represented by three components. Therefore, a color vector is an

element of $\mathbb{R}^3$:

$$c = (r, g, b), \quad r, g, b \in \mathbb{R}$$

In practice, the typical value ranges are

$$r, g, b \in [0,1] \quad or \quad r, g, b \in \{0,1, \dots, 255\}$$



This representation allows color values to be interpreted geometrically. The set of all possible colors occupies a cube in R3, with vertices including

$$(0,0,0), (1,0,0), (0,1,0), (0,0,1), (1,1,1)$$

The magnitude of a color vector may also be considered using the Euclidean norm

$$||\boldsymbol{c}|| = \sqrt{(r^2 + g^2 + b^2)}$$

which provides an approximate measure of perceived brightness, although human perception is not strictly linear.

B. Vector Spaces and Linear Combinations

A vector of space V over $\mathbb{R}$ is a set equipped with vectors closed under vector addition and scalar multiplication. For all **u, v** ∈ V and α, β ∈ $\mathbb{R}$, written as:

$$\boldsymbol{u} + \boldsymbol{v} \in \mathrm{V}, \quad \mathrm{k}\mathbf{u} \in \mathrm{V}$$

The RGB color space inherits the vector space structure of $\mathbb{R}^3$. As a result, any finite linear combination of color vectors is mathematically valid:

$$w = \sum_{\{i=1\}}^{\{n\}} k_i v_i, \quad k_i \in \mathbb{R}$$

Common image operations can be expressed using this formulation. Brightness scaling is given by

$$\boldsymbol{c}' = \alpha \boldsymbol{c}, \quad \alpha > 0,$$

Where $\alpha > 1$ increases brightness and $0 < \alpha < 1$ decreases brightness. Color interpolation and blending can be expressed as

$$\boldsymbol{c} = \lambda \boldsymbol{c}_1 + (1 - \lambda)\boldsymbol{c}_2, \quad 0 \le \lambda \le 1.$$

This expression is frequently used to model transparency and image overlay effects. Although practical implementations may impose constraints such as clipping values to a displayable range, these constraints do not alter the underlying linear structure.

C. Linear Transformations and Matrices

A linear transformation $T : \mathbb{R}^n \to \mathbb{R}^m$ satisfies the following properties for all **u, v** ∈ $\mathbb{R}^n$ and α ∈ $\mathbb{R}$,

$$T(\boldsymbol{u} + \boldsymbol{v}) = T(\boldsymbol{u}) + T(\boldsymbol{v}),$$

$$T(\alpha \boldsymbol{u}) = \alpha T(\boldsymbol{u}).$$

Every linear transformation between Euclidean spaces can be represented by matrix multiplication. In color manipulation, a transformation applied to a pixel color vector $x \in R^3$ is written as $\boldsymbol{x}' = A\boldsymbol{x}$ where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \in R^{3x3}$$

Explicitly,

$$r' = a_{11}r + a_{12}g + a_{13}b,$$
$$g' = a_{21}r + a_{22}g + a_{23}b,$$
$$b' = a_{31}r + a_{32}g + a_{33}b,$$

Special cases include:

- Identity transformation:

$$T(\alpha \boldsymbol{u}) = \alpha T(\boldsymbol{u}).$$

- Channel scaling:

$$A = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \beta & 0 \\ 0 & 0 & \gamma \end{bmatrix}$$

- Grayscale transformation:

$$A = \begin{bmatrix} w_r & w_g & w_b \\ w_r & w_g & w_b \\ w_r & w_g & w_b \end{bmatrix}, \quad w_r + w_g + w_b = 1.$$

Because matrix multiplication is applied independently to each pixel, the same linear transformation can be uniformly applied across an entire image. This property makes linear transformations an effective and mathematically transparent tool for basic color and brightness manipulation.

III. MODELING IMAGE COLOR USING LINEAR ALGEBRA

A. Digital Image as a Matrix of Vectors

A digital image with resolution of R x C consists of R rows and C columns of pixels. Each pixel contains color information encoded using the RGB model. Algebraically, instead of treating the image as a matrix of scalar values, each pixel can be represented as a vector in $\mathbb{R}^3$.

Let I denote an image. The pixel located at row *i* and column *j* is represented as

$$\boldsymbol{I}_{\mathrm{i,j}} = \begin{bmatrix} \mathrm{r_{i,j}} \\ \mathrm{g_{i,j}} \\ \mathrm{b_{i,j}} \end{bmatrix} \in \mathbb{R}^3 ,$$

Where $r_{i,j}, g_{i,j},$ and $b_{i,j}$ denote the red, green, and blue channel intensities, respectively. The image can be viewed as a matrix whose entries are vectors:

$$\boldsymbol{I} \in (\mathbb{R}^3)^{R\mathrm{x}C}$$

For computational convenience, the image is often reshaped into a collection of vectors. By flattening the image, all pixels are arranged into a matrix.

$$X = [v1v2 \ldots v_n] \in \mathbb{R}^{3\mathrm{x}N}, \quad \mathrm{N} = \mathrm{R\,x\,C}$$

where each column $\mathbf{v}_k \in \mathbb{R}^3$ corresponds to a single pixel. This representation enables linear transformations to be applied simultaneously to all pixels using matrix multiplication.

B. Brightness Manipulation as Scalar Multiplication

Brightness manipulation can be modeled as scaling the magnitude of each pixel's color vector. In Linear Algebra, changing the magnitude of a vector while preserving its direction is achieved through scalar multiplication.

Let a pixel color be represented by

$$v = (r, g, b) \in \mathbb{R}^3$$

And let $\alpha \in R_{>0}$ be a scalar. The brightness adjusted pixel is defined as

$$\mathrm{v}' = \alpha v = (\alpha r, \alpha g, \alpha b).$$

Two cases are of particular interest:

- **Brightness increase:**

$$\alpha > 1 \rightarrow ||v'|| > ||v||$$

- **Brightness decrease:**

$$0 < \alpha < 1 \rightarrow ||v'|| < ||v||$$

Geometrically, this operation moves the color vector along a straight line passing through the origin (0,0,0). Since the ratios r : g : b remain unchanged, the hue of the color is preserved while its intensity is modified. This operation satisfies the properties of a linear transformation and can be applied uniformly to all pixels in the image.

C. Color Transformation as Matrix Multiplication

While scalar multiplication adjusts intensity, it does not allow interaction between color channels. To modify color balance, mix channels, or apply stylistic effects, a more general linear transformation is required.

Let $\mathbf{v} \in \mathbb{R}^3$ be a pixel color vector. A color transformation is defined using a matrix, explicitly written as:

$$\begin{bmatrix} \mathrm{r'} \\ \mathrm{g'} \\ \mathrm{b'} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} \mathrm{r} \\ \mathrm{g} \\ \mathrm{b} \end{bmatrix}$$

Each output channel is a linear combination of the original RGB components. Applying the same matrix to every pixel yields a global color transformation:

$$\mathrm{X'} = \mathrm{AX}.$$

Example 1: Red Channel Emphasis

To enhance the red channel while keeping the other channels unchanged, a diagonal matrix is used:

$$A_{\mathrm{red}} = \begin{bmatrix} 1{,}5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This transformation scales only the red component, making the image appear warmer.

Example 2: Grayscale Conversion

Grayscale conversion can be modeled as a linear transformation that projects color vectors onto the gray diagonal in $\mathbb{R}^3$. A commonly used transformation is

$$A_{\mathrm{gray}} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.299 & 0.587 & 0.114 \\ 0.299 & 0.587 & 0.114 \end{bmatrix}$$

Applying this matrix results in

$$\mathrm{r' = g' = b'} = 0{,}299\mathrm{r} + 0.587\mathrm{g} + 0.114\mathrm{b},$$

IV. EXPERIMENTAL IMPLEMENTATION

To test the theoretical model, a single movie poster image was processed using direct linear algebra operations. The goal was to demonstrate how scalar multiplication and matrix-based color transformations affect brightness and overall color composition.

A. Experimental Setup

- **Input Image:** "poster.jfif" (standard movie poster)
- **Tools:** Python 3 with NumPy and Pillow (PIL) for image handling.
- **Operations**: Only scalar multiplication and matrix multiplication. No pre-built filters or brightness functions were used.

The operations apply directly to pixel vectors, reflecting the models described in II and III.

B. Implementation Example

The Python code:

```python
import numpy as np
from PIL import Image

# Load image and normalize RGB to [0,1]
img = Image.open("poster.jfif")
vectors = np.array(img) / 255.0
R, C, _ = vectors.shape

# Brightness adjustment
alpha = 1.5
bright_vectors = np.clip(vectors * alpha, 0, 1)

# Sepia transformation matrix
sepia_A = np.array([
    [0.393, 0.769, 0.189],
    [0.349, 0.686, 0.168],
    [0.272, 0.534, 0.131]
])
sepia_vectors = vectors.reshape(-1, 3).dot(sepia_A.T)
sepia_vectors = np.clip(sepia_vectors, 0, 1)
sepia_vectors = sepia_vectors.reshape(R,C,3)

# Red channel enhancement
red_A = np.array([
    [1.5, 0, 0],
    [0, 1, 0],
    [0, 0, 1]
])
red_vectors = vectors.reshape(-1, 3).dot(red_A.T)
red_vectors = np.clip(red_vectors, 0, 1)
red_vectors = red_vectors.reshape(R, C, 3)

# Convert to images and save
Image.fromarray((bright_vectors*255).astype(np.uint8)).save("poster_bright.jfif")
Image.fromarray((sepia_vectors*255).astype(np.uint8)).save("poster_sepia.jfif")
Image.fromarray((red_vectors*255).astype(np.uint8)).save("poster_red.jfif")
```

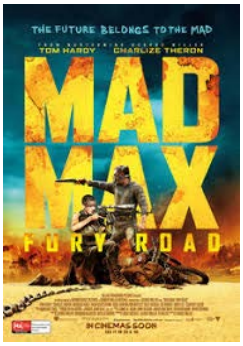
This section of the code begins by loading the movie poster image from the file "poster.jfif" and converting it into a NumPy array. The pixel values are normalized to the range [0,1] to represent vectors in $\mathbb{R}^3$. Next, a brightness adjustment is applied by multiplying every pixel vector by the scalar alpha = 1.5, effectively scaling the magnitude of each vector without changing its direction. The code then applies a sepia transformation by defining a 3×3 matrix sepia_A, which specifies the weighted combination of the original red, green, and blue channels. Each pixel vector is reshaped and multiplied by the transpose of sepia_A to perform the linear transformation across all pixels simultaneously. Finally, the transformed vectors are clipped to ensure all values remain within the [0,1] range, and reshaped back to the original image dimensions in preparation for conversion back to image format

This section of the code focuses on enhancing the red channel. A 3×3 diagonal matrix red_A is defined, scaling only the red component of each pixel by 1.5 while leaving green and blue unchanged. The pixel vectors are flattened to apply matrix multiplication efficiently across the entire image, then clipped to the valid [0,1] range. After the transformation, the array is reshaped back to the original image dimensions, ready for conversion into an image object. Finally, all three processed image arrays—brightness-adjusted, sepia-transformed, and red-enhanced—are converted back to standard image format for saving.

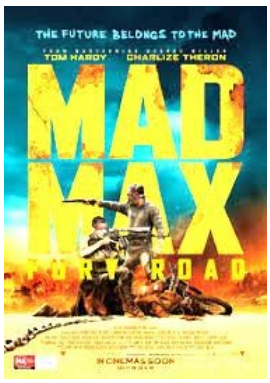
C. Result Visualization

The results of the experimental implementation demonstrate the practical effects of linear algebra on image data:

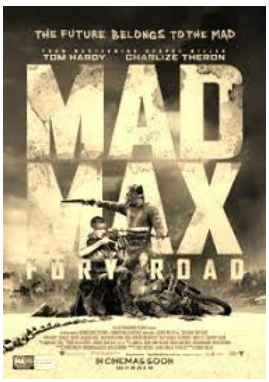
1. **Original Image:** The movie poster displays a wide range of vibrant colors and contrast. This serves as the baseline for comparison.



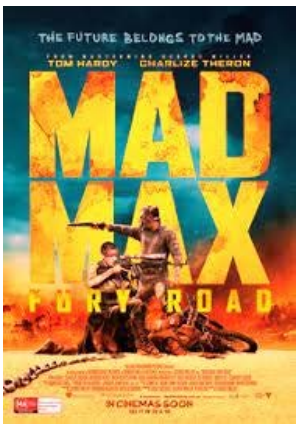
2. **Brightness-Adjusted Image:** Applying α=1.5 resulted in a global increase in pixel intensity. Dark regions became more visible, mid-tones were lifted, and bright areas approached saturation. This confirms that scalar multiplication effectively scales the vector norms of pixel colors while preserving their hue.



3. **Color-Transformed Image (Sepia):** The sepia transformation reduced the contribution of green and blue channels relative to red. The resulting image exhibits a warm, monochromatic tone while maintaining the spatial arrangement of original features.



4. **Red Channel Enhanced Image:** The code amplified the red channel while leaving green and blue unchanged resulted in a poster with noticeably warmer tones. Areas dominated by red became more vivid, while cooler regions (green and blue) appeared relatively muted.



V. Discussion

The experimental results confirm that a digital image can be effectively treated as a vector space. Representing each pixel as a vector in $\mathbb{R}^3$ allows manipulation of brightness and color directly using scalar and matrix operations.

A. Linear Behavior of Color Adjustments

Brightness adjustment via scalar multiplication proportionally increased all color channels while preserving their ratios, resulting in a brighter image without changing hues. Matrix transformations, such as the sepia filter and red channel enhancement, produced predictable changes: the sepia matrix combined RGB channels to create warm tones, while the red enhancement amplified the red component across the image. These operations demonstrate that linear transformations provide consistent and interpretable ways to modify visual features.

B. Practical Limitations

While linear algebra operations are effective for global adjustments, certain limitations emerged during experimentation:

1. **Clipping:** Pixel values must be clipped to [0,1] for display, introducing a minor non-linearity.
2. **No Semantic Awareness:** Transformations apply uniformly; they cannot target specific objects or regions.
3. **Global Effects Only:** Adjustments affect all pixels, which may unintentionally alter areas not meant to be emphasized.

VI. Conclusion

This study demonstrated the practical application of Linear Algebra in digital image processing using a movie poster as an example. By modeling the image as a matrix of vectors in $\mathbb{R}^3$, each representing an RGB pixel, we showed that brightness adjustments and color grading can be expressed as scalar and matrix multiplications.

Scalar multiplication effectively altered pixel intensity, while matrix multiplication enabled color transformations such as sepia toning and selective channel enhancement, preserving image structure while modifying perceptual color. These operations highlight how abstract concepts—basis vectors, linear independence, and linear transformations—manifest in real-world visual effects.

Despite their efficiency, linear methods have limitations: they cannot perform content-aware edits or semantic adjustments, and digital displays require clipping to maintain valid color ranges.

Overall, this work confirms that simple linear algebraic operations provide a robust and interpretable framework for basic image manipulation, bridging theoretical mathematics with practical visual computing. For computer science students, mastering these operations is

essential to understanding how computers perceive and modify images

REFERENCES

[1] R. Munir, *Ruang Vektor Umum (Bagian 1)*, Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025.

[2] R. Munir, *Ruang Vektor Umum (Bagian 2)*, Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025.

[3] R. Munir, *Ruang Vektor Umum (Bagian 3)*, Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025.

[4] R. Munir, *Vektor di Ruang Euclidean (Bagian 2)*, Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025.

[5] R. Munir, *Vektor di Ruang Euclidean (Bagian 3)*, Bahan Kuliah IF2123 Aljabar Linier dan Geometri, Program Studi Teknik Informatika, STEI-ITB, 2025

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 24 Desember 2025

Mikhael Benrael Tampubolon 13524009