Eigenvalue-Driven Scoring System for Readibility and Visual Comfort in UI Color Palettes

Noumisyifa Nabila Nareswari - 13523058[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*13523058@std.stei.itb.ac.id*

***Abstract*— Color plays a crucial role in determining the quality and usability of user interfaces (UI) design, determining both its aesthetics and functionality. This paper introduces an algorithm for scoring color palettes based on the principles of Material Design 3 (Material You) by Google. By representing colors as vectors in the HCT (Hue, Chroma, Tone) space, the algorithm generates tonal variations for specific UI roles, and then the algorithm incorporates Euclidean distance to evaluate the similarity between the input color palette. The algorithm later dynamically generates an ideal color palette and compare its result to the input using Euclidean dstance formula approach. Results demonstrate the algorithm's effectiveness in scoring the color palette, which could potentially help many development of applications in user-centric UI design workflows**

***Keywords*— Material Design, Euclidean Distance, HCT Color Space, User Interface Design.**

I. Introduction

Color coordination plays an important role as a deciding factor in determining the overall quality of any user interface. Studies in human-computer interaction state that a well-coordinated color scheme could significantly boosts the readability, usability, user engagement [3], and aesthetic of an interface which also enhances overall user experience across any digital platforms. Not just that, color coordination also holds an important role in emotion control of users, operation logic, and presentation of information [4]. By building and organizing the right color scheme, digital platform developers could make their product effectively reach the expected goal.

Designing optimal color palettes can now be approached systematically using quantified and theoretical methods, thanks to the development of numerous color theories and rules over recent years. Many of these theories and rules of thumb for determining a color scheme are derived from earlier foundational concepts. For instance, in 1810, Johann Wolfgang von Goethe introduced the standard color wheel, which remains widely used today [6]. Similarly, the Munsell Color System is based on a three-dimensional model, where colors are represented by three attributes: hue, value, and chroma [7]. However, the focus of this paper is on the color theory and algorithms derived from Material Design's color algorithms, specifically from Material 3, Google's open-source design system, which will be explored and utilized in the development of the proposed algorithm.

Euclidean distance is an important fundamental used in the field of vectors within Euclidean space. Euclidean distance is the difference between two norm vectors in the space of R^n (Euclidean space). Or it could also be interpreted as the straight-line distance between two points in multi-dimensional space. In computational application context, color can be represented as a vector. Each color attribute value will be represented as a component of its vector. This will allow further assessment on many color schemes.

Despite its importance, achieving color scheme with optimal harmony for building interfaces remains a challenge. To address this challenge, this study introduces an algorithm designed to evaluate the quality of a color palette or scheme for interface design by measuring its Euclidean distance from an ideal color palette. This will help interface designers to get more objective evaluation of its color palette/scheme before applied to the design. The algorithm will give a score with a range from 1-100 and give the ideal color palette/scheme similar to its input but with some adjustment. This algorithm serves as a practical tool to assess ad optimizing color palette/scheme in interface design context.

II. Theoretical Background

A. Basic Color Theories

a. Hue

Hue represents the angular position of a color in a color wheel measured in degrees (0-360). Hue is one of many basic attribute of a color. It is what typically defines which color name is a certain color. For example, 0° represents the color red, 120° represents the color green, and 240° represents the color blue.

b. Chroma

Chroma quantifies the degree of a color's saturation. Higher chroma value represents a vivid color while lower chroma value represents a more muted-toned-color. It could also be interpreted as how much a color deviates from gray. For example, colors like neon pink and bright orange have high chroma value while pastel colors have lower chroma value.

c. Tone

Tone indicates the relative luminance/brightness of a color on a scale from black to white. In short, tone measures the lightness of a color. For example, dark colors like charcoal have low tone value while light colors like off-white have high tone value. In design, this attribute helps creating depth and making effective contrast between the background and the elements.

d. Contrast

Contrast refers to the perceptual difference between two colors that could determine a certain design element visibility and creating emphasis to certain elements. Contrast could be used in many design contexts such as hue contrast, chroma contrast, tone contrast. All of those basically representing the value difference between two colors color attribute.

B. Material Design's Color Algorithm

Material Design, specifically Material Design 3 (Material You) is a design language introduced by Google to help many designers to emphasize clarity, boldness, and intentionality, in their interface designs. One of its important component in their whole design language is its color system. Material Design's color algorithm introduces a systematic and quantifiable approach to color scheme generation and harmonization in interface design context. The algorithm mainly operates with using HCT (Hue, Chroma, Tone) color space approach to represent color, a uniform system derived from CAM16. It got its own advantages compared to using RGB and HSL color system as it enables more precise control over color relationship and consistency over various display conditions. Here is a more extensive explanation on its key components.

a. Role-Based Color Palettes

Material Design's color algorithm generates its color palettes and assign it to some specific roles. Each role has its own place in UI design. Some roles are designated to certain components in UI:

- Primary colors: Represents the main theme color.
- Secondary colors: Complementary colors to support the primary colors.
- Tertiary colors: Works for accents throughout the interface design.
- Neutral colors: Usually applied to backgrounds.
- Error Palette: Assigned for components that states an error in system, usually high in contrast and is usually red.

b. Dynamic Color Generation

Material Design's color algorithm implement dynamic color generation functionality to make the color palettes tailored to the wanted brand or personality of its user, enabling personalization and also accessibility to certain display condition. The algorithm starts with a seed color than later expanded into a scheme then organized into a solid ready-to-use palette. Some points to be highlighted are:

- Hue adjustments: The algorithm able to adjust the other color (secondary, tertiary, error, and neutral) to its primary color that is originated to its seed color.
- Chroma adjustments: Reduces chroma for the neutral palettes to make sure the other color that work as a component doesn't subdued.
- Tone mapping: Assign tones to palettes based on each of the role and ensuring its accessibility and contrast to each other.

C. Vector in Euclidean Space

Euclidian space is named after the Greek Mathematician named Euclid. An n-dimensional Euclidean space, denoted as Rn, is a vector space consisting of all ordered n-tuples of real numbers.

$$Rn=\{(x1,x2,\ldots,xn)|xi\in R,i=1,2,\ldots,n\}$$

Each xi in the formula represents a coordinate of a point in the space and the dimension n indicates the number of axes in the space.

In Euclidean space, a vector is defined as an ordered collection of numbers that represents direction of a component relative to the coordinate system (or relative to the Euclidean space) and the magnitude of each direction. In n-dimensional space (Euclidean space), all vectors written originated from the origin point (O point) as followed:

$$v=(v1,v2,\ldots,vn)$$

These are some key attributes of vector:

a. Magnitude

Magnitude or length of a vector is calculated

using the Euclidean norm with the formula as followed:

$$\|v\|=v12+v22+\cdots+vn2$$

This formula is derived from the Pythagorean theorem, representing the straight-line distance of two vectors.

b. Unit vector

Unit vector is a vector that has a magnitude of exactly 1. This attribute defines the direction of a vector without regard to its magnitude. The unit vector of a vector can be calculated through this formula:

$$v\hat{} = v / \|v\|$$

In this formula, v represents the original vector and ‖v‖ represents the magnitude of vector v. Standard basis vectors are one of many example of a unit vectors. For example, in 3 dimensional-space,

$$i\hat{}= (1,0,0)$$
$$j\hat{}= (0,1,0)$$
$$k\hat{}= (0,0,1)$$

D. Color as a Vector in Material Design's Color Algorithm Context

In the Material Design's color algorithm, each color represented as a three-dimensional vector that represents of the HCT (Hue, Chroma, Tone) value. Then, in this context, color represented as followed:

$$C = (h,c,t)$$

With C represents the color vector, h represents the hue value of the color, c represents the chroma value of the color, and t represents the tone value of the color. This way of color representation leverages easier calculation using vector operations for color generation, adjustment, and comparison (about comparison later discussed in Euclidean Distance). Addition and subtraction operation serves to shift colors by modifying one or more of its component. This operation performed using this formula:

$$c'=c+\Delta c$$

In the formula, c′ represents the modified color vector, c represents the initial color vector before modification, and Δc represents the modification applied to the initial vector.

E. Euclidean Distance

Euclidean distance is a concept in linear algebra and vector mathematics used to measure the distance between two vectors in Euclidean space. The Euclidean distance between vector A(u_1, u_2, …, u_n) and vector B (v_1, v_2, …, v_n) is formulated as followed:

This formula calculates the straight-line distance between two vectors with generalized application of Pythagorean theorem. Here, u_i and v_i represent the value of point A and B in the i-th base/dimension. This concept widely used to measure similarity or dissimilarity between objects. Suppose there are set of objects represented as feature vector of w_1, w_2, …, w_n. Each vector represented in the same Euclidean space (same dimension) with n amount of components. Given a query vector z, to determine which object of the set is most similar to the query, analyze the Euclidean distance of each object with the given query then choose the object with the smallest distance that indicates that it is the most similar object.

F. Importance of Color Harmony in Interface Designs

Author names and affiliations are to be centered beneath the title and printed in Times 11-point, non-boldface type. Multiple authors may be shown in a two- or three-column format, with their affiliations below their respective names. Affiliations are centered below each author name, italicized, not bold. Include e-mail addresses if possible. Follow the author information by three blank lines before main text.

III. PROPOSED ALGORITHMS

The proposed algorithm is mainly focused on the implementation of Euclidean distance between the input color palette with the desirable color palette generated through the Material 3 Design's color algorithm principles for scoring functionality. The key steps including input processing, dynamic palette generation, palette scoring, and the more desirable palette according Material 3 Design's color algorithm palette generation.

A. Input Color Processing

The algorithm starts with collecting user input of hex colors that divided into color roles defined by Material 3 Design's color principles which are primary, secondary, tertiary, surface, and error. User can input several color for each role, acting like a color palette because basically each role itself has its own more intricate sub-role for more specific use in UI design.

```python
def get_colors_for_role(role):
    """
    Collects user input for a specific color role.
    """
    colors = []
    print(f"Enter colors for {role}. Input -999 to finish:")
    while True:
        color_input = input(f"{role} color: ").strip()
        if color_input == "-999":
            break
        try:
            colors.append(hex_to_argb(color_input))
        except ValueError:
            print("Invalid hex format. Please try again.")
    return colors
```

Image 1. get_colors_for_role function implementation

Later the input will be breakdown and converted into color vector of HCT for later processing.

B. Dynamic Palette Generation

For each role the program will generates a tonal palette based on the seed color provided from the input. The algorithm will adjust so that it generates a color palette that doesn't go so off of the original seed color but still able to fulfill the Material 3 Design's color principal. In general, for each palette these are the adjustments being made to the original color seed:

- Primary color: Retains the input hue and chroma and generates more tonal variations to fill in the required sub-role color.
- Secondary and tertiary: Applies hue shifts +30° for secondary colors and +60° for tertiary colors.
- Surface: Determines the right surface tone based on the contrast of other color role palettes.
- Error: Uses a predefined red hue with little chroma adjustments based off of the color seed.

```python
def generate_dynamic_palette(source_color, role, all_colors=None):
    source_hct = Hct.fromArgb(source_color)

    if role == "primary":
        return TonalPalette.fromHueAndChroma(source_hct.hue, source_hct.chroma).generate_tonal_palette()
    elif role == "secondary":
        adjusted_hue = (source_hct.hue + 30) % 360
        adjusted_chroma = max(source_hct.chroma - 20, 8)
        return TonalPalette.fromHueAndChroma(adjusted_hue, adjusted_chroma).generate_tonal_palette()
    elif role == "tertiary":
        adjusted_hue = (source_hct.hue + 60) % 360
        adjusted_chroma = max(source_hct.chroma - 40, 4)
        return TonalPalette.fromHueAndChroma(adjusted_hue, adjusted_chroma).generate_tonal_palette()
    elif role == "surface":
        return _generate_surface_palette(source_hct, all_colors)
    elif role == "error":
        return TonalPalette.fromHueAndChroma(25.0, 84.0).generate_tonal_palette()
    elif role == "neutral":
        neutral_hct = Hct.fromHueChromaTone(source_hct.hue, 12.0, 90)
        return TonalPalette.fromHct(neutral_hct).generate_tonal_palette([60, 70, 80, 90, 95])
    elif role == "neutralVariant":
        neutral_variant_hct = Hct.fromHueChromaTone(source_hct.hue, 16.0, 90)
        return TonalPalette.fromHct(neutral_variant_hct).generate_tonal_palette([60, 70, 80, 90, 95])
    else:
        raise ValueError(f"Unknown role: {role}")
```

Image 2. Generate_dynamic_palette function implementation

The image gives general view of how the palette generator algorithm works. The vectors result of this algorithm later processed for scoring and color palette generation.

C. Palette Scoring

In this part of the code, the program will compare the original color palette with the more desirable generated color palette. The algorithm mainly using the Euclidean distance formula between two HCT color vectors.

```python
def euclidean_distance(color1, color2):
    """
    Calculates Euclidean distance between two colors in HCT space.
    """
    return sqrt(sum((a - b) ** 2 for a, b in zip(color1, color2)))
```

Image 3. Euclidean_distance function implementation

```python
def score_palette(input_palette, generated_palette):
    input_hct = [Hct.fromArgb(color) for color in input_palette]

    # Ensure only valid color values are processed
    generated_palette = [
        color if isinstance(color, int) else hex_to_argb(color)
        for color in generated_palette if isinstance(color, (int, str))
    ]
    generated_hct = [Hct.fromArgb(color) for color in generated_palette]

    total_distance = 0
    for input_color, generated_color in zip(input_hct, generated_hct):
        total_distance += euclidean_distance(
            (input_color.hue, input_color.chroma, input_color.tone),
            (generated_color.hue, generated_color.chroma, generated_color.tone),
        )

    max_distance = len(input_palette) * 100
    return max(0, 100 - (total_distance / max_distance) * 100)
```

Image 4. Score_palette function implementation

```python
def score_palettes(input_palettes, generated_palettes):
    scores = {}
    for role, input_colors in input_palettes.items():
        if role not in generated_palettes:
            scores[role] = 0.0
            continue

        generated_colors = generated_palettes[role]

        # Debugging: Print palettes for validation
        print(f"\nScoring {role}:")
        print(f"Input Colors: {input_colors}")
        print(f"Generated Colors: {generated_colors}")

        scores[role] = score_palette(input_colors, generated_colors)
    return scores
```

Image 5. Score_palettes function implementation

The algorithm mainly working in these three functions. The score_palettes mainly do scoring in the role level and this function will call score_palette that also do scoring but in the smaller scale whish is in color palette level. The score_palette then also will call the euclidean_distance function that will calculate the Euclidean of each two compared HCT color vector to generate the score of the input palette. This resulting a palette scoring system in a role level. Program will generate five scores which are the primary palette score, secondary palette score, tertiary palette score, surface palette score, and error palette score.

D. Palette Generation

After finishing the palette data generation, the program will generate the PNG images for better visualization of the corrected color palette. Each role's tonal palette will

be displayed in one image of color horizontal strip using this algorithm.

```python
def save_palette_to_folder(palettes, folder):
    """
    Saves all generated palettes to a specified folder.
    Each role is saved as a single PNG file.
    """
    for role, colors in palettes.items():
        filename = os.path.join(folder, f"{role}.png")
        save_palette_to_png(colors, filename)
```

```python
def save_palette_to_png(palette, filename):
    """
    Saves a tonal palette as a PNG file.
    Ensures all colors are ARGB integers.
    """
    img = Image.new("RGB", (100 * len(palette), 100))
    for i, color in enumerate(palette):
        if isinstance(color, str):
            color = hex_to_argb(color)

        r = (color >> 16) & 0xFF
        g = (color >> 8) & 0xFF
        b = color & 0xFF
        for x in range(i * 100, (i + 1) * 100):
            for y in range(100):
                img.putpixel((x, y), (r, g, b))
    img.save(filename)
```

Image 6. Save_palette_to_folder and save_palette_to_png function implementation

All the color palette horizontal strips will be put into one folder inside the device.

IV. Results and Discussions

This section presents the outcomes of the proposed algorithm, demonstrating its ability to generate a color palettes suitable for UI designs and aligned with Material Design principles. Pictures (screenshots) of example testing are included to illustrate the results visually

The algorithm was tested using various color palette inputs. For eah role, the input palette served as the base for generating dynamic palettes. The following are the color palette data input used for testing

Test	Primary	Secondary	Tertiary	Surface	Err
1.	#32c75c	#ff9c33	#198ae1	#ffffff	#c
	#31a379				#e
2.	#112299	#5533aa	#eecc44	#000011	#ee2211

Table 1. Testing colors data

Later the algorithm generated tonal variations of each role based on the input colors. Here are the testing results starting from test number 1.

```
Enter colors for primary (hex format, e.g., #4285F4). Input -999 to finish:
primary color: #32c75c
primary color: #31a379
primary color: -999
Enter colors for secondary (hex format, e.g., #4285F4). Input -999 to finish:
secondary color: #ff9c33
secondary color: -999
Enter colors for tertiary (hex format, e.g., #4285F4). Input -999 to finish:
tertiary color: #198ae1
tertiary color: -999
Enter colors for surface (hex format, e.g., #4285F4). Input -999 to finish:
surface color: #ffffff
surface color: -999
Enter colors for error (hex format, e.g., #4285F4). Input -999 to finish:
error color: #c43b5d
error color: #ed3b3b
error color: -999

Generating desired palettes...
```

Image 7. Input testing for test case 1

```
Primary Score: 44.07
Secondary Score: 24.26
Tertiary Score: 20.03
Surface Score: 79.41
Error Score: 0.00
```

Image 8. Palette score testing for test case 1

Here shown the score of each color palette role and followed are the color palette that generated from the algorithm.

Image 9. Primary palette for test case 1

Image 10. Secondary palette for test case 1

Image 11. Tertiary palette for test case 1

Image 12. Surface palette for test case 1

Image 13. Error palette for test case 1

Now, these are the testing results starting from test number 2.

```
Enter colors for primary (hex format, e.g., #4285F4). Input -999 to finish:
primary color: #112299
primary color: -999
Enter colors for secondary (hex format, e.g., #4285F4). Input -999 to finish:
secondary color: #5533aa
secondary color: -999
Enter colors for tertiary (hex format, e.g., #4285F4). Input -999 to finish:
tertiary color: #eecc44
tertiary color: -999
Enter colors for surface (hex format, e.g., #4285F4). Input -999 to finish:
surface color: #000011
surface color: -999
Enter colors for error (hex format, e.g., #4285F4). Input -999 to finish:
error color: #ee2211
error color: -999

Generating desired palettes...
```

Image 14. Input testing for test case 2

```
Primary Score: 74.85
Secondary Score: 47.13
Tertiary Score: 0.15
Surface Score: 10.89
Error Score: 66.34
```

Image 15. Palette score testing for test case 2

Here shown the score of each color palette role and followed are the color palette that generated from the algorithm.

Image 16. Primary palette for test case 2

Image 17. Secpndary palette for test case 2

Image 18. Tertiary palette for test case 2

Image 19. Surface palette for test case 2

Image 20. Error palette for test case 2

The testing results showing that it is possible and feasible to use Euclidean distance formula to do color palette scoring by calculating its distance from the ideal color palette and then treating the distance as a score.

V. Conclusion

This study presents an algorithm for generating and scoring color palettes based on Material Design principles, utilizing the HCT (Hue, Chroma, Tone) color space to ensure perceptual uniformity and harmony. The algorithm dynamically adjusts colors for specific UI roles, including primary, secondary, surface, and error palettes, while maintaining visual contrast and accessibility. Later the generated palette compared using Euclidean distance formula. It is proves that it is effective using the said method.

VI. Acknowledgment

First and foremost, I would like to express my deepest gratitude to Mr. Rinaldi Munir, my lecturer for the Linear Algebra and Geometry course. Thanks to his guidance, expertise, and dedication in teaching us throughout the semester, I manage to finish this project and understand many new knowledge beyond my imagination.

I would also like to extend my heartfelt thanks to my parents, whose unwavering support and encouragement have been a constant source of motivation during my academic journey.

References

[1] IEEE Xplore. (n.d.). *Document on color research*. Retrieved January 2, 2025, from https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9035402file:///C:/Users/mia/Downloads/266-537-1-SM.pdf

[2] Unknown author. (n.d.). *Research paper on color analysis* [PDF]. Retrieved December 30, 2024, from 266-537-1-SM.pdf

[3] ResearchGate. (n.d.). *Does color matter on web user interface design*. Retrieved December 31, 2024, from https://www.researchgate.net/publication/320546804_Does_Color_Matter_on_Web_User_Interface_Design

[4] IEEE Xplore. (n.d.). *Document on accessibility in design*. Retrieved January 3, 2025, from https://ieeexplore.ieee.org/document/9708614.com

[5] Color Matters. (n.d.). *Why color matters*. Retrieved January 1, 2025, from https://www.colorcom.com/research/why-color-matters

[6] ResearchGate. (n.d.). *Goethe's theory of color in the arts*. Retrieved December 29, 2024, from https://www.researchgate.net/publication/331906749_Goethe's_Theory_of_Color_in_the_Arts

[7] ScienceDirect. (n.d.). *Munsell color system*. Retrieved January 4, 2025, from https://www.sciencedirect.com/topics/engineering/munsell-system#:~:text=Munsell%20Color%20System,-Albert%20Munsell%2C%20a&text=He%20used%20the%20three%20color,in%20100%20equally%20spaced%20hues

[8] Munir, R. (2024). *Aljabar Geometri: Material & references for geometry* [Online course material]. Retrieved January 5, 2025, from https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/algeo24-25.htm#Makalah

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, 27 Desember 2024

Noumisyifa Nabila Nareswari
13523058