Analisis Konsep Vektor dalam Pemrosesan Gambar dengan Filter Brave Pink Hero Green

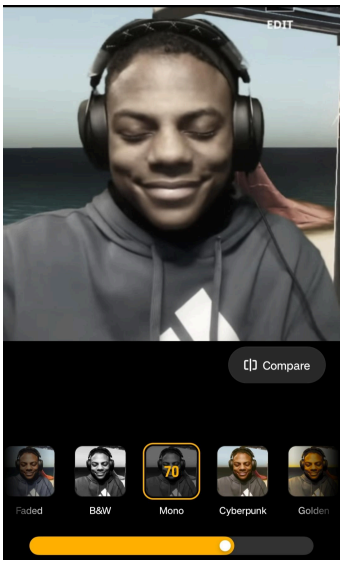
Jonathan Levi - 13523132
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523132@mahasiswa.itb.ac.id, [2]jonathanlevi12345678@gmail.com

***Abstract*—Peningkatan *trend* aktivisme digital sering disertai dengan simbol yang unik. Salah satunya yang viral pada tahun 2025 adalah penggunaan filter "*brave pink* dan *hero green*". Banyak orang, termasuk artis dan *influencer*, yang menge-*post* foto mereka dengan filter ini di sosial media seperti Instagram, X, dan YouTube. Hal ini dilakukan sebagai simbol solidaritas, wujud protes dari rakyat terhadap pemerintah Republik Indonesia, dan untuk melayangkan tuntutan 17+8 rakyat. Gambar tersebut dapat berupa *post* di sosial media maupun sebagai gambar profil dari akun mereka. Namun, apakah Anda tahu bagaimana cara kerja filter ini di balik layar?**

***Keywords*—Brave pink hero green, efek duotone, pemrosesan gambar, vektor.**

I. Pendahuluan

Ada berbagai hal yang dapat dilakukan untuk memanipulasi gambar saat mengedit gambar. Beberapa fitur edit gambar di antaranya adalah *crop* gambar, rotasi gambar, dan filter. Aplikasi pengedit gambar (*image editor*), termasuk aplikasi pengedit bawaan ponsel pintar, memberikan *template-template* filter yang dapat digunakan secara langsung, dengan *setting* untuk mengatur intensitas dari filter tersebut. Contohnya adalah filter *monochromatic*, *warm*, *vibrant*, natural, dan sebagainya.



Gambar 1.1. Fitur pengaturan filter gambar pada aplikasi pengedit gambar bawaan ponsel pintar. Sumber: Dokumen Pribadi, https://youtu.be/ZIPCptw_Otg?t=2h8m17s

Filter "*brave pink* dan *hero green*" sendiri merupakan jenis filter *duotone*. Filter *duotone* adalah filter yang fokus pada dua warna tertentu dan *shade-shade* dari warna itu (warna-warna lain yang mirip atau mendekati salah satu dari kedua warna utama).

Gambar 1.2. Contoh foto Jerome Polin dengan filter "*brave hero* dan *pink green*". Sumber: https://www.inilah.com/ini-cara-ganti-profil-brave-pink-hero-green-di-medsos-simbol-perlawanan-dan-solidaritas

Setiap filter memiliki operasi matematika yang terjadi di balik layar dan diabstraksikan (disembunyikan) dari pengguna agar lebih sederhana (*user-friendly*). Operasi matematika ini melibatkan vektor. Oleh karena itu, pada makalah ini, akan dibahas cara kerja filter ini secara matematis.

II. Teori Dasar

A. Vektor di Ruang Euclidean

Vektor adalah besaran yang memiliki nilai dan arah. Ada berbagai cara untuk merepresentasikan vektor dengan suatu simbol, di antaranya adalah $\vec{v}$ dan v. Ruang vektor R^n yang paling sering digunakan adalah vektor di R^2 (vektor dua dimensi) dan vektor di R^3 (vektor tiga dimensi) [1]. Vektor yang akan dibahas adalah vektor tiga

dimensi, karena setiap piksel pada gambar terdiri dari tiga bagian, yaitu piksel R (*red*/merah), G (*green*/hijau), dan B (*blue*/biru).

1. Notasi dan Komponen Vektor

   Vektor dapat disimbolkan sebagai berikut:

   - (v_1, v_2, v_3)
   - $\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$
   - $\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$
   - $v_i\vec{i} + v_2\vec{j} + v_3\vec{k}$

   Vektor yang ditulis dengan bentuk seperti ini adalah vektor yang berawal dari titik O, atau dalam kata lain, vektor yang titik awalnya (0,0,0) [1].

2. Panjang/Norma/*Magnitude* Vektor

   Panjang vektor atau norma vektor atau magnitude vektor disimbolkan sebagai $||\vec{v}||$ atau $||v||$.

   - Jika $\vec{v} = (v_1, v_2, v_3)$, maka:

$$||\vec{v}|| = \sqrt{v_1^2 + v_2^2 + v_3^2}.$$

3. Penjumlahan dan Pengurangan Vektor

   Misalnya, ada dua buah vektor

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

   Penjumlahan vektor adalah operasi menjumlahkan komponen ke-n setiap vektor. Penjumlahan vektor-vektor tersebut adalah

$$\vec{u} + \vec{v} = \begin{pmatrix} u_1 + v_1 \\ u_2 + v_2 \\ u_3 + v_3 \end{pmatrix}.$$

   Pengurangan vektor adalah operasi mengurangi komponen ke-n satu vektor dengan vektor lainnya. Pengurangan vektor-vektor tersebut adalah

$$\vec{u} - \vec{v} = \begin{pmatrix} u_1 - v_1 \\ u_2 - v_2 \\ u_3 - v_3 \end{pmatrix}.$$

4. Perkalian Vektor

   Perkalian Vektor adalah perkalian antara sebuah vektor dengan vektor lainnya atau dengan konstanta (skalar). Perkalian vektor terbagi menjadi dua jenis, yaitu:

   - Perkalian vektor dengan skalar.
   - Perkalian titik (*dot product*)
   - Perkalian silang (*cross product*)

   Perkalian vektor dengan skalar adalah operasi yang mengalikan sebuah bilangan (nilai skalar) dengan sebuah vektor. Misalnya, diketahui sebuah skalar k dan sebuah vektor

$$\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix},$$

   maka perkalian antara skalar dan vektor tersebut adalah

$$k\vec{v} = \begin{pmatrix} k \times v_1 \\ k \times v_2 \\ k \times v_3 \end{pmatrix}$$

   [1].

   Perkalian titik (dot product) adalah operasi yang mengalikan komponen ke-n setiap vektor, lalu menjumlahkan hasil perkalian tersebut. Perkalian titik disimbolkan dengan tanda "$\cdot$" di antara kedua buah vektor yang akan dikalikan titik. Misalnya, ada dua buah vektor

$$\vec{u} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}, \vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix},$$

   maka perkalian titik dari vektor-vektor tersebut adalah

$$\vec{u} \cdot \vec{v} = (u_1 \times v_1) + (u_2 \times v_2) + (u_3 \times v_3)$$

   [1].

   Perkalian silang (*cross product*) adalah operasi yang mengalikan kedua panjang vektor dengan sin dari sudut yang mengapit kedua vektor. Perkalian silang disimbolkan dengan tanda "×" di antara kedua buah vektor yang akan dikalikan silang. Misalnya, ada dua buah vektor

$$\vec{u} = u_1 i + u_2 j + u_3 k,$$
$$\vec{v} = v_1 i + v_2 j + v_3 k,$$

   maka

$$\vec{u} \times \vec{v} = \begin{bmatrix} i & j & k \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{bmatrix}$$
$$= (u_2 v_3 - u_3 v_2) i - (u_1 v_3 - u_3 v_1) j + (u_1 v_2 - u_2 v_1) k.$$

5. Normalisasi Vektor

   Normalisasi vektor adalah pembagian sebuah vektor dengan panjang atau magnitude vektor

tersebut. Berdasarkan definisinya, proses normalisasi vektor dapat dirumuskan sebagai

$$\vec{u} = \frac{\vec{v}}{||\vec{v}||},$$

yang mana $\vec{v} = (v_1, v_2, v_3)$.

Rumus tersebut dapat dijabarkan menjadi

$$\vec{u} = \left(\frac{v_1}{||\vec{v}||}, \frac{v_2}{||\vec{v}||}, \frac{v_3}{||\vec{v}||}\right),$$

yang mana panjang vektor $||\vec{u}|| = 1$, yaitu

$$||\vec{u}|| = \sqrt{u_1^2 + u_2^2 + u_3^2} = 1,$$

atau lebih tepatnya

$$\sqrt{\left(\frac{v_1}{||\vec{v}||}\right)^2 + \left(\frac{v_2}{||\vec{v}||}\right)^2 + \left(\frac{v_3}{||\vec{v}||}\right)^2} = 1$$

[1].

III. Pengimplementasian Konsep Vektor pada Filter Brave Pink Hero Green

A. Konversi Gambar Menjadi Kumpulan Piksel

Setiap gambar memiliki unit terkecil yang disebut piksel. Agar sebuah gambar dapat diproses, gambar ini harus diubah menjadi sebuah data angka terlebih dahulu. Data angka ini berupa kumpulan piksel. Piksel-piksel ini dapat disimbolkan sebagai vektor R^3 (R, G, B).

Sebagai contoh, gambar dengan resolusi 1920 × 1080 dapat direpresentasikan sebagai *array* dengan 1920 baris. Setiap baris tersebut memiliki 1080 buah vektor (R, G, B). Visualisasi dari format data yang akan dibentuk adalah sebagai berikut:

```
img_array = [
    # Baris 0
  [
    [R0_0,G0_0,B0_0], # Piksel Kolom 0
    [R0_1,G0_1,B0_1], # Piksel Kolom 1
    # ...,          # Piksel Kolom 2-1918
    [R0_1919,G0_1919,B0_1919]
      # Piksel Kolom 1919
  ],

  # Baris 1
  [
    [R1_0,G1_0,B1_0], # Piksel Kolom 0
    [R1_1,G1_1,B1_1], # Piksel Kolom 1
    # ...,          # Piksel Kolom 2-1918
    [R1_1919,G1_1919,B1_1919]
      # Piksel Kolom 1919
    ],

  # Baris 2 - 1078
  # ...,

  # Baris 1079
  [
    [R1079_0,G1079_0,B1079_0],
      # Piksel Kolom 0
    [R1079_1,G1079_1,B1079_1],
      # Piksel Kolom 1
    # ...,          # Piksel Kolom 2-1918
    [R1079_1919,G1079_1919,B1079_1919]
      # Piksel Kolom 1919
  ]
]
```

Pada kode implementasi, gambar di-*load* menggunakan Pillow atau Python Imaging Library (PIL). Data `img_array` tersebut akan dibentuk secara otomatis melalui *function* `np.array(image)`. `Img_array` berukuran $(H, W, 3)$ dengan H sebagai tinggi gambar (*height*), W sebagai lebar gambar (*width*). Piksel-piksel ini kemudian dinormalisasi menjadi 0-1 dengan cara mengalikan tiap komponen dalam vektor (R, G, B) dengan skalar $\frac{1}{255}$. Numpy otomatis melakukan operasi perkalian/pembagian masing-masing komponen dalam semua vektor (dalam kasus ini, yakni R, G, dan B) dengan skalar.

```
img =
Image.open(input_path).convert('RGB')
# Ubah ke 0 (x1) - 1 (x2)
img_array = np.array(img) / 255.0
```

B. Konversi Piksel ke Grayscale

Kumpulan vektor-vektor piksel yang didapatkan sebelumnya akan diubah menjadi *grayscale*. Hal ini dilakukan karena efek *duotone* nanti bekerja dengan cara melihat intensitas terang (*brightness*). Semakin mendekati nilai 0, semakin mendekati warna hijau piksel tersebut; semakin mendekati nilai 1, semakin mendekati warna *pink* piksel tersebut.

Dalam kasus *grayscale*, semakin mendekati nilai 0, semakin mendekati warna hitam piksel tersebut; semakin mendekati nilai 1, semakin mendekati warna putih piksel tersebut. Konversi piksel gambar menjadi *grayscale* dapat dilakukan dengan cara melakukan perkalian titik (*dot product*) antara tiap vektor (R, G, B) dengan vektor *linear luminance*, yaitu

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} \cdot \begin{pmatrix} 0.2126 \\ 0.7152 \\ 0.0722 \end{pmatrix},$$

atau jika dijabarkan secara langsung:

$$luminance = R \times 0.2126 + G \times 0.7152 + B \times 0.0722$$

$$luminance = 0.2126R + 0.7152G + 0.0722B$$

[2].

Dalam kode implementasi, perkalian titik ini menggunakan *function* `dot()` dari *library* Numpy.

```
luma_weights = np.array([0.2126,
                         0.7152,
                         0.0722])
luminance = img_array.dot(luma_weights)
```

C. Kontras Otomatis (Normalisasi Min-max)

Mekanisme kontras otomatis meregangkan (*stretch*) kecerahan piksel (data dari variabel `luminance`). Tujuan mekanisme ini adalah untuk memaksimalkan kontras pada gambar hasil. Cara kerjanya adalah "menarik" nilai-nilai kecerahan piksel sehingga nilai kecerahan minimum menjadi 0 dan nilai kecerahan maksimum menjadi 1. Nilai-nilai lainnya di antara nilai minimum dan maksimum akan "tertarik" mendekati ke nilai 0 atau 1. Mekanisme ini menggunakan konsep normalisasi min-max (*min-max normalization*), yang dapat dirumuskan sebagai

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}.$$

dengan syarat $x_{max} - x_{min} \neq 0$ [3].

Dalam konteks mekanisme kontras otomatis, rumus ini dapat disimbolkan sebagai

$$luminance' = \frac{luminance - min\_lum}{max\_lum - min\_lum},$$

dengan syarat $max\_lum - min\_lum \neq 0$. Jika $max\_lum - min\_lum = 0$, maka mekanisme ini dapat dilewatkan.

Dalam kode implementasi, Numpy akan otomatis mengurangi dan membagi tiap nilai dalam `luminance` dengan nilai skalar `min_lum` dan `max_lum - min_lum`.

```
min_lum = np.min(luminance)
max_lum = np.max(luminance)
if min_lum != max_lum:
    luminance = (luminance - min_lum) /
(max_lum - min_lum)
luminance = np.clip(luminance, 0, 1)
```

D. Interpolasi Linear

Data `luminance` yang telah melewati proses normalisasi min-max (*auto contrast*) belum cukup untuk memberikan filter "*brave pink* dan *hero green*" ke gambar. Hal ini dikarenakan pada data tersebut, hanya diketahui dua buah data yang pasti, yakni:

- Nilai kecerahan piksel 0: konversi warna menjadi *hero green* (*hex code* `#1B602F` atau `rgb(27, 96, 47)`)
- Nilai kecerahan piksel 1: konversi warna menjadi *brave pink* (*hex code* `#F784C5` atau `rgb(247, 132, 197)`)
- Nilai kecerahan piksel lain (misalnya 0,82): konversi menjadi vektor (R, G, B) berapa?

Untuk mengatasi hal tersebut, digunakan interpolasi linear. Interpolasi linear bertujuan untuk mendapatkan nilai dari data-data lain di antara dua buah data yang nilainya diketahui. Interpolasi linear dapat dirumuskan sebagai berikut:

$$y = y_1 + (x - x_1) \times \frac{y_2 - y_1}{x_2 - x_1}$$

[4].

Dalam hal ini, x_1 dan x_2 adalah dua data nilai kecerahan piksel yang diketahui, yaitu masing-masing 0 dan 1. y_1 dan $y_2$adalah pemetaan (konversi) dari data nilai kecerahan piksel yang diketahui, yaitu masing-masing vektor $(27, 96, 47)$ dan $(247, 132, 197)$. x adalah data yang ingin dikonversi menjadi y, yaitu sebuah vektor t. y adalah $duotone$.

Jika variabel-variabel dari rumus tersebut disubstitusikan dengan variabel pada implementasi program, maka rumus tersebut adalah:

$$duotone = SHADOW\_COLOR + (t - 0) \times \frac{HIGHLIGHT\_COLOR - SHADOW\_COLOR}{1 - 0}$$

$$duotone = \begin{pmatrix} 27 \\ 96 \\ 47 \end{pmatrix} + t \times \left(\begin{pmatrix} 247 \\ 132 \\ 197 \end{pmatrix} - \begin{pmatrix} 27 \\ 96 \\ 47 \end{pmatrix} \right)$$

$$duotone = \begin{pmatrix} 27 \\ 96 \\ 47 \end{pmatrix} + t \times \begin{pmatrix} 220 \\ 36 \\ 150 \end{pmatrix}$$

Agar $duotone$ bisa dihasilkan sebagai sebuah kumpulan piksel dengan struktur data yang sama dengan `img_array` pada Bagian A (ukuran $(H, W, 3)$), maka t perlu dikonversi dari ukuran (H, W) menjadi *array* tiga dimensi berukuran $(H, W, 1)$ terlebih dahulu. Numpy akan otomatis mengalikan setiap bagian "1" dari *array* tersebut dengan vektor tiga dimensi dan menghasilkan $duotone$ berukuran $(H, W, 3)$. Hal ini dilakukan agar kumpulan piksel `duotone` tersebut dapat dikonversi menjadi gambar nantinya.

Kode implementasi dari interpolasi linearnya adalah:

```
# Hero Green (#1B602F, rgb(27,96,47))
SHADOW_COLOR = np.array([27, 96, 47])
# Brave Pink (#F784C5, rgb(247,132,197))
HIGHLIGHT_COLOR = np.array([247, 132,
197])

// ...

# 4. Duotone Mapping
# Ubah dimensi (H, W) jadi (H, W, 1)
t = luminance[:, :, np.newaxis]
```

```
# Interpolasi linear
# y = y1 + (x-x1) * ((y2-y1) / (x2-x1))
duotone = SHADOW_COLOR + (t-0) *
((HIGHLIGHT_COLOR-SHADOW_COLOR) / (1-0))
```

E. Konversi Kumpulan Piksel duotone *Menjadi Gambar*

Data `duotone` tersebut masih berupa kumpulan piksel. Data tersebut akan dikonversi balik menjadi sebuah gambar. Namun, data tersebut tidak bisa langsung dikonversi menjadi gambar, karena memiliki komponen-komponen bertipe data *float* dalam vektor (R, G, B) tersebut. Setiap komponen dari vektor tersebut harus dikonversi menjadi **integer** dahulu. Vektor (R, G, B) tidak memiliki komponen negatif, hanya nol dan positif, sehingga jenis integer harus ***unsigned***. Bit yang dipilih adalah **8 bit**, karena komponen vektor (R, G, B) berkisar dari 0 sampai 255 (8 bit dapat menyimpan $2^8 = 256$ nilai). Oleh karena itu, komponen/nilai yang ada dalam `duotone` disimpan sebagai tipe data `np.uint8`.

Implementasinya adalah sebagai berikut:

```
result = Image.fromarray(
            duotone.astype(np.uint8)
         )
result.save(output_path)
```

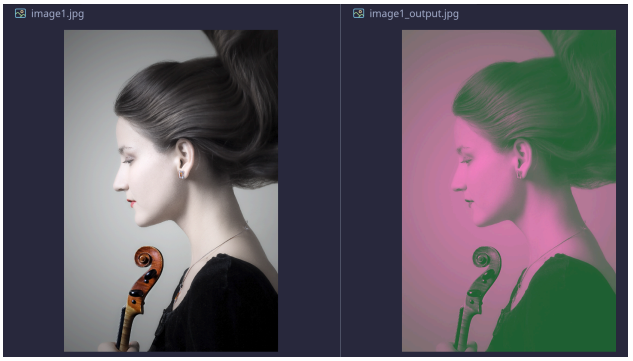
F. Hasil Akhir dan Implementasi Program

Setelah semua proses tersebut selesai, kode tersebut dapat digabung menjadi sebuah program Python yang cukup sederhana. Berikut hasil implementasi programnya menggunakan *library* Numpy dan PIL:

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

# Hero Green (#1B602F, rgb(27, 96, 47))
SHADOW_COLOR = np.array([27, 96, 47])
# Brave Pink (#F784C5, rgb(247, 132,
197))
HIGHLIGHT_COLOR = np.array([247, 132,
197])

input_path = sys.argv[1]
output_path = sys.argv[2]

# 1. Load gambar
img =
Image.open(input_path).convert('RGB')
# Ubah ke 0 (x1) - 1 (x2)
img_array = np.array(img) / 255.0

# 2. Konversi ke grayscale (Rec. 709)
luma_weights = np.array([0.2126,
                         0.7152,
                         0.0722])
luminance = img_array.dot(luma_weights)

# 3. Auto-Contrast
min_lum = np.min(luminance)
max_lum = np.max(luminance)
if min_lum != max_lum:
    luminance = (luminance - min_lum) /
(max_lum - min_lum)
luminance = np.clip(luminance, 0, 1)

# 4. Duotone Mapping
# Ubah dimensi (H, W) jadi (H, W, 1)
t = luminance[:, :, np.newaxis]

# Interpolasi linear
# y = y1 + (x-x1) * ((y2-y1) / (x2-x1))
duotone = SHADOW_COLOR + (t-0) *
((HIGHLIGHT_COLOR-SHADOW_COLOR) / (1-0))

# 5. Save
result = Image.fromarray(
            duotone.astype(np.uint8)
         )
result.save(output_path)
print(f"Gambar berhasil diproses:
'{input_path}' -> '{output_path}'")
```

IV. Percobaan

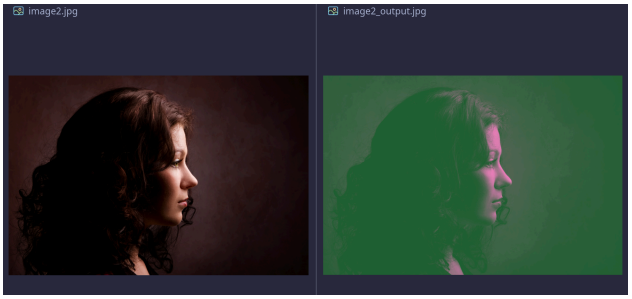
Percobaan akan menggunakan 3 kasus percobaan (*test case*). Kasus pertama adalah menggunakan gambar yang dominan terang. Kasus kedua adalah menggunakan gambar yang dominan gelap. Kasus ketiga adalah menggunakan gambar yang bagian gelap dan terangnya seimbang.



Gambar 4.1. Hasil percobaan dengan menggunakan gambar yang dominan terang. Sumber: https://www.stockvault.net/photo/102431/ill-give-you-all-my-soul, Dokumen Pribadi

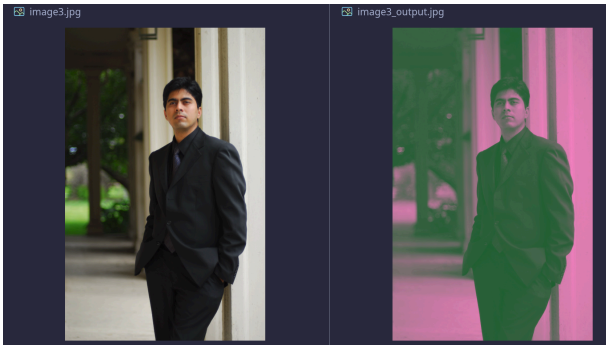
Untuk kasus pertama, gambar cenderung memiliki

shade merah muda (pada bagian yang terang), sedangkan bagian yang gelap seperti rambut dan pakaian memiliki *shade* hijau tua.



Gambar 4.2. Hasil percobaan dengan menggunakan gambar yang dominan gelap. Sumber: https://www.stockvault.net/photo/121745/beautiful-woman, Dokumen Pribadi

Untuk kasus kedua, gambar cenderung memiliki *shade* hijau tua (pada bagian yang gelap), sedangkan bagian yang terang seperti wajah memiliki *shade* merah muda.



Gambar 4.3. Hasil percobaan dengan menggunakan gambar yang memiliki bagian terang dan gelap yang seimbang. Sumber: https://www.stockvault.net/photo/104980/business-man, Dokumen Pribadi

Untuk kasus ketiga, latar belakang (*background*) dan wajah memiliki *shade* merah muda karena bagian tersebut cenderung terang, sedangkan rambut, pakaian, dan pohon memiliki *shade* hijau tua karena bagian tersebut cenderung gelap.

V. Kesimpulan

Konsep-konsep vektor seperti penjumlahan vektor, pengurangan vektor, perkalian vektor dengan skalar, perkalian titik dapat digunakan untuk membuat implementasi dari filter *duotone*, termasuk filter *brave hero* dan *pink green*. Konversi gambar menjadi vektor menggunakan konsep perkalian vektor dengan skalar. Konversi piksel menjadi *grayscale* menggunakan konsep perkalian titik. Mekanisme kontras otomatis (normalisasi min-max) melibatkan konsep pengurangan vektor dan perkalian vektor dengan skalar. Interpolasi linear melibatkan pengurangan vektor, perkalian vektor dengan skalar, serta penjumlahan vektor.

VI. Ucapan Terima Kasih

Penulis mengucapkan rasa syukur dan terima kasih kepada Tuhan Yang Maha Esa karena telah menyertai dan memberkati penulis dalam menyelesaikan makalah ini. Penulis berterima kasih kepada orang tua penulis karena telah memotivasi penulis untuk membuat makalah ini. Penulis juga mengucapkan terima kasih kepada dosen-dosen dan teman-teman penulis karena telah membimbing dan mengajari penulis mata kuliah ini, serta memberi fasilitas dan lingkungan yang kondusif sehingga penulis dapat megembangkan makalah dan implementasi program ini.

Referensi

[1] H. Anton and C. Rorres, *Elementary linear algebra: Applications Version*, 11th ed. 2014, ch. 3.

[2] T. Shields, "'Standard' RGB to grayscale conversion," *Stack Overflow*, Jul. 12, 2013. https://stackoverflow.com/a/17619494 (accessed Dec. 09, 2025).

[3] D. Jain, "Data normalization in data mining," *GeeksforGeeks*, Nov. 29, 2025. https://www.geeksforgeeks.org/machine-learning/data-normalization-in-data-mining/ (accessed Dec. 20, 2025).

[4] J.-K. Chou and C.-K. Yang, "Simulation of face/hairstyle swapping in photographs with skin texture synthesis," *Multimedia Tools and Applications*, vol. 63, no. 3, pp. 729–756, Oct. 2011, doi: 10.1007/s11042-011-0891-1.

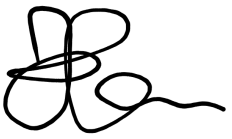
Lampiran

- *Repository* kode sumber: https://github.com/RealNath/brave-pink-hero-green-filter
- Video demonstrasi: https://youtu.be/rI4zkKNpMBA

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



Jonathan Levi. 13523132