Di balik Digital Painting Software: Penerapan Dekomposisi Matriks dan Vektor di Ruang Euclidian pada Blurring dan Blending Modes

Muhammad Zahran Ramadhan Ardiana - 13523104[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13523104@std.stei.itb.ac.id

***Abstract*— Digital painting memanfaatkan teknologi untuk menciptakan karya visual kreatif dengan dukungan perangkat lunak seperti Photoshop dan CSP. Teknik Gaussian Blur digunakan untuk menghasilkan transisi halus, sedangkan blending modes seperti Multiply, Overlay, dan Color Dodge memungkinkan manipulasi visual kompleks melalui operasi matematika antara lapisan dasar dan campuran. Penelitian ini menganalisis penerapan matematika, termasuk dekomposisi matriks dan vector di ruang Euclidean, dalam proses digital painting. Hasilnya menunjukkan bahwa integrasi seni dan matematika menghasilkan karya yang estetis dan efisien secara komputasi. Studi ini diharapkan dapat membuka peluang inovasi algoritma digital painting di masa depan.**

***Keywords*— Digital painting, Gaussian Blur, Blending modes, Matematika visual.**

I. PENDAHULUAN

Digital painting merupakan sebuah alat yang menggunakan teknologi digital untuk membuat karya visual baik dua dimensi maupun tiga dimensi. Perangkat lunak digital mampu menghasilkan spektrum warna yang sangat luas dengan model pewarnaan RGB dan Depth menjadikannya sebagai alat kreatif yang mampu menciptakan karya berwarna yang sangat bervariasi.

Dalam konteks pembuatan karya 2D, Adobe Photoshop, Krita, dan Clip Studio Paint merupakan contoh digital painting sofware yang menyediakan banyak kemudahan dan kecepatan dalam memproses gambar. Melalui sistem *layering* yang tersedia, seniman dapat menggabungkan gambar yang berbeda dengan efek berbeda sehingga tercipta visual yang lebih berwarna. Ketersediaan Filter, *blending*, *libraries*, dan lain-lain juga memberikan seniman kebebasan yang luas untuk berkreasi [1].

Di balik antarmuka yang ramah pengguna dan fitur intuitif yang disediakan oleh perangkat lunak tersebut, terdapat fondasi matematika yang kompleks, terutama dalam memproses efek visual seperti *blurring* dan *blending modes*. Operasi ini tidak sekadar manipulasi warna pada permukaan piksel, tetapi melibatkan pemrosesan data gambar dalam bentuk matriks dan vektor di ruang Euclidean.

Salah satu metode *blurring* yang paling umum adalah *Gaussian Blur*. Penggunaan efek *Gaussian Blur* dalam karya sering dipakai untuk membuat efek yang lebih terfokuskan ke objek atau subjek tertentu. Teknik *blurring* ini bekerja dengan menggunakan kernel Gaussian, yaitu sebuah matriks filter yang memiliki distribusi Gaussian dua dimensi. Operasi ini diterapkan melalui konvolusi pada matriks piksel, di mana setiap piksel baru dihitung berdasarkan bobot dari piksel-piksel tetangganya. Proses ini memungkinkan terciptanya efek blur yang halus dan realistis, yang sangat penting dalam berbagai aplikasi, seperti mengurangi noise, menciptakan efek kedalaman, atau menghasilkan transisi yang lebih mulus antara elemen gambar.

Di sisi lain, *blending modes* seperti *Multiply*, *Color Dodge*, dan *Overlay* memungkinkan seniman digital untuk menciptakan komposisi visual yang lebih dinamis. Setiap mode blending memiliki rumus matematis spesifik yang menentukan bagaimana dua lapisan gambar berinteraksi satu sama lain. Misalnya, mode *Multiply* bekerja dengan mengalikan nilai warna piksel dari dua lapisan, menghasilkan warna yang lebih gelap, sedangkan *Color Dodge* bekerja dengan meningkatkan kecerahan berdasarkan perbandingan nilai warna antara dua lapisan. Mode *Overlay* menggabungkan pendekatan dari *Multiply* dan *Screen*, tergantung pada tingkat kecerahan warna dasar.

Studi ini bertujuan untuk menganalisis bagaimana prinsip dekomposisi matriks dan vektor di ruang Euclidean diterapkan dalam algoritma *Gaussian Blur* dan *blending modes* (*Multiply*, *Color Dodge*, dan *Overlay*). Dengan memahami dasar matematika di balik fitur ini, diharapkan dapat membuka wawasan tentang bagaimana perangkat lunak digital painting memadukan estetika dengan efisiensi komputasi, serta memberikan peluang untuk pengembangan algoritma yang lebih optimal di masa depan.

II. KAJIAN TEORI

A. *Dekomposisi Matriks dan Vektor di Ruang Euclidian*

1. Dekomposisi Matriks

Dekomposisi matriks adalah proses memecah suatu matriks menjadi beberapa matriks komponen yang lebih

sederhana untuk analisis atau komputasi lebih lanjut. Teknik ini digunakan secara luas dalam aljabar linear, pengolahan sinyal, dan machine learning untuk menyelesaikan sistem persamaan linear, reduksi dimensi, dan ekstraksi fitur [8].

Jenis-jenis Dekomposisi Matriks

a) Dekomposisi LU (Lower-Upper Decomposition):
Metode dekomposisi LU adalah metode yang menguraikan suatu matriks *A* menjadi hasil kali dua buah matriks yaitu matriks segitiga bawah (Lower triangular) *L* dan matriks segitiga atas (Upper triangular) *U* sehingga:

$$A = L \cdot U.$$

b) Dekomposisi QR:
Metode dekomposisi QR adalah metode yang memfaktorkan suatu matriks *A* berukuran m x n menjadi hasil kali matriks orthogonal *Q* dan matriks segitiga atas *R* sehingga:

$$A = Q \cdot R.$$

c) Dekomposisi SVD:
Metode dekomposisi SVD adalah metode yang memfaktorkan suatu matriks A berukuran m x n menjadi hasil kali matriks *U*, $\sum$, dan V^T. Di mana *U* dan V merupakan matriks orthogonal yang secara terurut berukuran m x m dan n x n. Sedangkan matriks $\sum$ merupakan matriks berukuran m x n yang hanya mengandung nilai-nilai singular dari matriks *A* pada diagonal utamanya sehingga:

$$A = U \cdot \sum \cdot V^T.$$

Implementasi dekomposisi SVD seringkali digunakan untuk mereduksi dimensi guna mempercepat proses operasi.

2. Vektor di Ruang Euclidian

Vektor merupakan suatu kuantitas fisik yang memiliki nilai besaran dan arah, sedangkan ruang vektor merupakan tempat vektor di definisikan [8]. Secara geometris, vektor direpresentasikan sebagai panah yang memiliki ttik awal dan ujung. Vektor dalam ruang Euclidean adalah kuantitas fisik matematika yang merepresentasikan arah dan besaran dalam ruang yang berhingga (1D, 2D, 3D, atau lebih tinggi). Vektor biasanya didefinisikan sebagai elemen dari ruang vektor R^n, di mana *n* merupakan dimensi ruang Euclidian atau dituliskan sebagai berikut:

$$\mathbf{v} = (v_1, v_2, \ldots, v_n) \text{ atau } \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$$

Vektor-vektor di ruang Euclidian dapat dioperasikan menggunakan operasi dasar sebagai berikut:

a) Penjumlahan dan Pengurangan Vektor
Jika $v = (v_1, v_2, \ldots, v_n)$ dan $w = (w_1, w_2, \ldots, w_n)$, maka:

$$v \pm w = (v_1 \pm w_1, v_2 \pm w_2, \ldots, v_n \pm w_n).$$

b) Perkalian Skalar
Jika $v = (v_1, v_2, \ldots, v_n)$ dan skalar $k \in R$, maka

$$k \cdot v = (k \cdot v_1, k \cdot v_2, \ldots, k \cdot v_n).$$

c) Norma vector (Magnitudo)
Panjang atau magnitude untuk suatu vector $v = (v_1, v_2, \ldots, v_n)$ adalah:

$$\| v \| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}.$$

d) Dot Product (Perkalian titik)
Jika $v = (v_1, v_2, \ldots, v_n)$ dan $w = (w_1, w_2, \ldots, w_n)$, maka:

$$v \cdot w = (v_1 \cdot w_1 + v_2 \cdot w_2 + \ldots + v_n \cdot w_n).$$

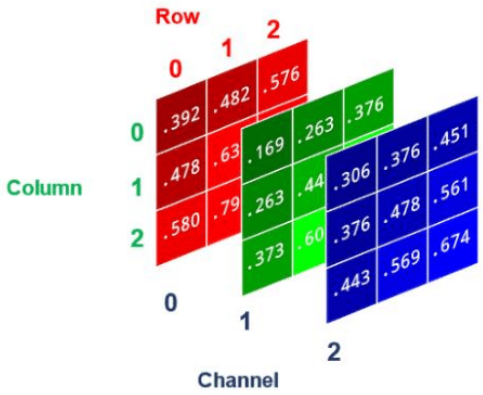
B. Gambar Digitial dan Warna

1. Gambar Digital

Gambar digital adalah representasi visual yang disimpan dalam format digital dan sering digunakan dalam berbagai aplikasi seperti fotografi, grafik komputer, dan pengolahan citra. Secara matematis, gambar digital biasanya berupa kumpulan nilai angka riil atau kompleks yang direpresentasikan dalam bentuk bit tertentu. Umumnya, gambar ini memiliki bentuk persegi panjang dengan tinggi dan lebar sebagai dimensinya.

Dalam konteks ini, citra digital dapat dianggap sebagai fungsi intensitas cahaya, di mana dan adalah koordinat spasial, dan nilai fungsi tersebut pada setiap titik menunjukkan tingkat kecerahan citra pada titik tersebut. Ukuran gambar digital diwakili oleh matriks berukuran baris dan kolom, dengan setiap elemen disebut piksel atau "picture element". Proses digitalisasi koordinat dikenal sebagai pencuplikan gambar (*image sampling*), sedangkan proses digitalisasi derajat keabuan disebut kuantisasi derajat keabuan (*gray-level quantization*).

Gambar berwarna dalam model RGB direpresentasikan sebagai matriks tiga dimensi (saluran), di mana tiap-tiap piksel pada gambar mengandung unsur intensitas atau *depth* dari tiga warna utama: Merah, Hijau, dan Biru. Ilustrasi matriks gambar berwarna dapat dilihat sebagai berikut:



Gambar 2.1 Ilustrasi gambar digital berwarna dalam RGB
(Sumber: https://www.researchgate.net/figure/Numpy-Array-RGB-Channel_fig1_351929771)

Gambar grayscale adalah gambar yang hanya memiliki satu saluran (channel), atau dengan kata lain direpresentasikan sebagai matriks dua dimensi, di mana setiap piksel dalam matriks ini hanya merepresentasikan jumlah cahaya dalam gambar. Hal ini memberikan

informasi intensitas pada rentang tertentu (misalnya 0–255 untuk format gambar 8-bit).

Proses konversi suatu gambar dari RGB ke Grayscale, yang juga dikenal sebagai reduksi dimensi, dapat mempermudah komputr atau alat digital untuk menanganinya karena memerlukan lebih sedikit memori dan operasi. Beberapa metode dapat digunakan untuk mengubah gambar berwarna menjadi gambar grayscale, seperti Metode Rerata dan Metode Berbobot.

Proses konversi yang mudah menggunakan metode rerata dengan formula

$$g = \frac{1}{3}(R + G + B)$$

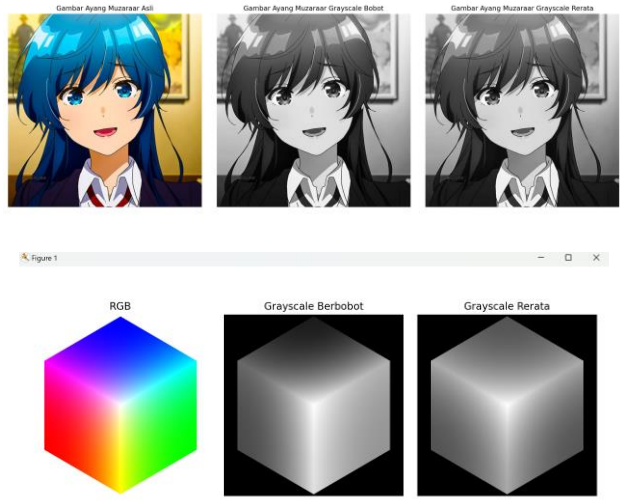
dengan nilai g merupakan representasi nilai intensitas pada suatu pixel.

Namun, metode rerata dianggap kurang sesuai dengan persepsi mata manusia terhadap warna, sehingga pembobotan untuk setiap warna perlu disesuaikan. Sebagai opsi lain, karena mata manusia lebih sensitif terhadap warna hijau, diikuti oleh warna merah, dan kurang sensitif terhadap warna biru, para peneliti telah mengembangkan formula lain yaitu Metode Berbobot, dengan formula:

$$g = 0{,}299R + 0{,}587G + 0{,}114B$$

Koefisien ini (0,299, 0,587, 0,114) dipilih untuk memperkirakan persepsi manusia terhadap intensitas warna [4].

Perbedaan konversi dari RGB ke Grayscale dengan menggunakan formula yang berbeda dapat dilihat pada gambar di bawah ini:



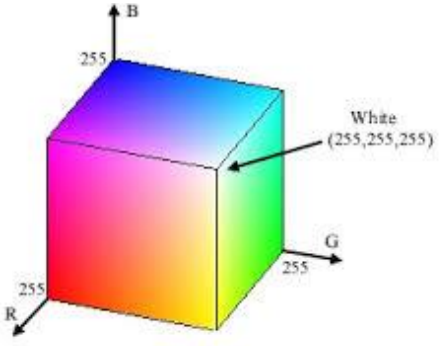
Gambar 2.2 Perbedaan grayscale berbobot dan rerata pada gambar digital
(Sumber: Arsip Penulis)

Warna biru pada gambar akan terlihat lebih gelap pada grayscale berbobot. Sedangkan Warna hijau akan terlihat lebih terang pada grayscale berbobot dibandingkan dengan grayscale hasil Metode Rerata.

2. Warna

Manusia mampu melihat radiasi elektromagnetik pada rentang panjang gelombang 400 hingga 700 nanometer sebagai warna. Penglihatan ini menjadi dasar sistem representasi warna yang digunakan pada perangkat digital.

Sistem yang umum digunakan untuk merepresentasikan warna adalah sistem RGB (Red, Green, Blue). RGB merupakan sistem penggabungan warna primer (*additive primary colors*), yaitu merah, hijau, dan biru, untuk menghasilkan warna tertentu. Tiap saluran biasanya bernilai 8-bit *depth*, atau sebanyak 2^8 kombinasi warna. Karena tiap piksel menampilkan warna dari tiap tiga saluran, maka ada sekitar 2^{8x3} = 16,77 juta warna berbeda yang dapat ditampilkan. Banyak kombinasi dari tiap saluran warna dapat direpresentasikan sebagai ruang vector 3D



Gambar 2.3 Kombinasi warna dalam vector 3D
(Sumber: http://matlab.izmiran.ru/help/toolbox/images/color4.html)

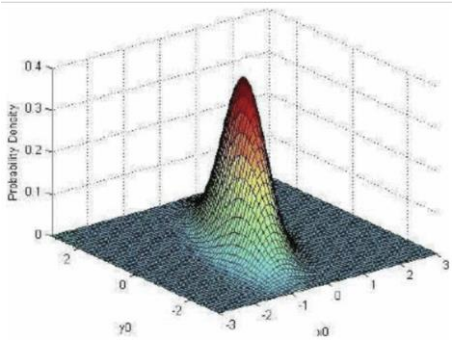
Semakin besar *bit depth*, semakin banyak variasi warna yang dapat dihasilkan, serta transisi yang lebih mulus dalam gradien warna. Akan tetapi, mata manusia hanya mampu membedakan sekitar 10 juta warna, sehingga perbedaan antara gambar dengan warna 10-bit dan 12-bit tidak terlalu signifikan bagi manusia. Namun, perbedaan antara gambar dengan warna 4-bit dan 8-bit dapat terlihat dengan jelas [5].

Nilai intensitas warna dalam sistem RGB juga dapat direpresentasikan dengan kode heksadesimal. Setiap nilai angka dari 0 hingga 255 dikonversi menjadi simbol angka (0–9) dan huruf (A–F). Penggunaan sistem RGB ini memberikan fleksibilitas dalam menghasilkan berbagai warna dengan mengatur intensitas masing-masing warna primer. Sistem ini mendasari representasi warna pada berbagai perangkat elektronik dan aplikasi grafis.

C. *Blurring dan Blending Modes*

1. Blurring

Blurring dalam konteks perangkat lunak digital painting adalah teknik manipulasi gambar untuk mengurangi ketajaman detail dengan meratakan perbedaan nilai piksel di sekitarnya. Proses ini dilakukan dengan menerapkan filter yang melibatkan perhitungan pada sekelompok piksel di sekitar piksel target. Bobot distribusi menentukan kontribusi setiap piksel tetangga pada hasil akhirnya. Salah satu metode paling umum untuk mencapai efek ini adalah Gaussian Blur, yang menggunakan kernel Gaussian untuk menghitung nilai piksel baru berdasarkan distribusi Gaussian di area sekitarnya.



Gambar 2.4 Distribusi Gaussian
(Sumber: https://ieeexplore.ieee.org/document/6044249/metrics#metrics)

a) Kernel

Kernel adalah matriks kecil yang digunakan dalam operasi konvolusi untuk memproses data berupa matriks yang lebih besar, seperti gambar. Kernel sering disebut juga sebagai filter atau mask. Kernel Gaussian mengikuti distribusi Gaussian, di mana piksel di pusat memiliki bobot tertinggi, sedangkan bobot semakin berkurang ke tepi. Distribusi ini memastikan efek blur yang merata dan alami. Rumus Gaussian Kernel 2D adalah:

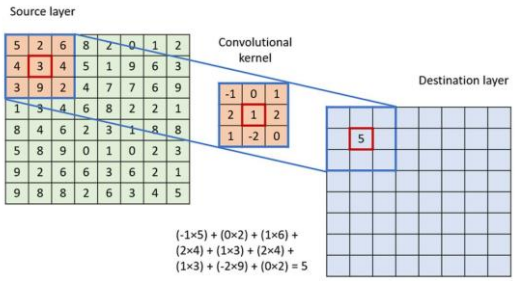
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Di mana adalah koordinat piksel relatif terhadap pusat kernel, yang dianggap berada di posisi (0,0). Sebagai contoh, untuk kernel dengan radius 3, koordinat x dan y berkisar dari -3 hingga 3 [3].

Simpangan baku (σ) memengaruhi seberapa signifikan piksel tetangga terhadap piksel pusat dalam memengaruhi hasil komputasi. Nilai yang kecil membuat distribusi bobot lebih terfkus di pusat, menghasilkan efek blur yang lebih tajam (local blur). Sebaliknya, yang besar menyebarkan efek blur lebih luas. Sedangkan Radius berperan untuk menentukan ukuran kernel. Jika radius dibesarkan, area yang dipertimbangkan untuk bluring semakin besar dan menyebar ke area yang lebih luas untuk nilai sigma yang besar juga. Tapi jika nilai sigma kecil efek blur tetap terkonsentrasi ditengah walaupun cakupan lebih besar, sehingga efek blur hanya menyebar sedikit.

b) Konvolusi

Secara definisi konvolusi adalah proses menerapkan filter pada data untuk mengekstrak fitur tertentu. Dalam pemrosesan gambar, distribusi Gaussian perlu didekati dengan kernel konvolusi. Proses tersebut melibatkan penerapan konvolusi antara matriks gambar dengan kernel Gaussian [2].



Gambar 2.5 Operasi konvolusi pada suatu piksel
(Sumber: https://viso.ai/deep-learning/convolution-operations/)

Seperti pada gambar di atas, konvolusi dilakukan dengan cara mengkali elemen-elemen kernel dengan piksel yang sesuai pada gambar, lalu menjumlahkan hasilnya atau dapat dibuat digambarkan dengan formula:

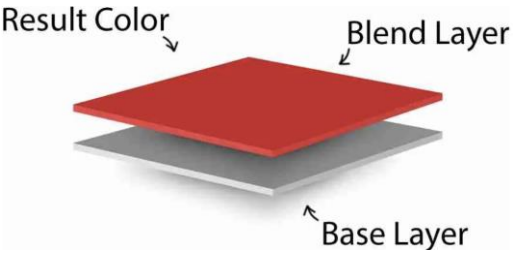
$$G(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} I(i+m, j+n) \cdot K(m,n)$$

- *G(i,j)*: nilai piksel hasil konvolusi di titik i,j
- *I(i+m, j+n)*: nilai piksel pada area gambar
- *K(m,n*: nilai kernel di posisi *(m,n)*
- *k*: Radius kernel

2. Blending Modes

Blending Mode adalah metode atau persamaan matematika yang digunakan untuk menggabungkan lapisan gambar dengan cara tertentu berdasarkan hue, saturasi, luminisitas, atau kombinasi komponen-komponen tersebut sehingga menghasilkan efek visual yang unik. Fungsi ini sering digunakan untuk memberikan tekstur, overlay, atau menyesuaikan efek gambar pada area tertentu tanpa memerlukan layer mask.

Agar blending mode dapat berfungsi, diperlukan minimal dua lapisan, yaitu lapisan dasar/latar belakang (Base Layer) dan lapisan campuran (Blend Layer). Lapisan dasar berfungsi sebagai latar belakang, sedangkan lapisan campuran adalah lapisan yang diterapkan di atasnya [6].



Gambar 2.6 Illustrasi lapisan *blending modes*
(Sumber: https://photoshoptrainingchannel.com/blending-modes-explained/)

Ketika bekerja dengan blending mode pada perangkat lunak digital painting, piksel dari lapisan campuran diproses melalui operasi pencampuran dengan piksel yang sesuai pada lapisan dasar. Pencampuran ini diterapkan pada setiap piksel secara individual untuk menghasilkan hasil akhir yang optimal. Proses pencampuran ini bersifat parametris, artinya perubahan yang dilakukan tidak akan

merusak gambar asli, dan pengaturan mode pencampuran dapat disesuaikan kembali kapan saja sesuai kebutuhan.

Contoh blending mode yang cukup sering digunakan adalah Color Dodge, Multiply, dan Overlay. Formula operasi pada blending mode tersebut direpresentasikan dalam bentuk C_{result} sebagai piksel hasil blending, C_f sebagai piksel pada lapisan blend, dan C_b piksel pada lapisan base [7].

a) Multiply

Multiply kerap kali diguanakan untuk membuat gambar menjadi lebih memadu dengan latar atau sebagai efek bayangan pada suatu objek. Multiply menghasilkan efek lebih gelap karena mengalikan setiap komponen warna.

$$C_{result} = C_f \cdot C_b$$

Jika salah satu warna mendekati hitam (0), hasilnya akan lebih gelap. Namun, Jika salah satu warna mendekati putih (1), warna lainnya tetap sama.

b) Overlay

Overlay biasa digunakan untuk menambah efek warna yang lebih kontras atau membuat warna tertentu menjadi lebih cerah pada gambar. Overlay merupakan kombinasi dari metode Multiply dan Screen, tergantung pada nilai warna/piksel latar belakang. Jika piksel latar belakang memiliki nilai gelap (kurang dari atau sama dengan 0,5), maka metode Multiply digunakan. Sebalknya, jika nilai piksel lebih terang (lebih dari 0,5), metode Screen diaplikasikan.

$$C_{result} = \begin{cases} 2 \cdot C_f \cdot C_b, & C_b \leq 0.5 \\ 1 - 2 \cdot (1 - C_f) \cdot (1 - C_b), & C_b > 0.5 \end{cases}$$

c) Color Dodge

Color Dodge memberikan efek yang lebih cerah dibandingkan Screen dengan cara mengurangi kontras antara lapisan dasar dan lapisan pencampur [6]. Efek ini sering digunakan oleh seniman digital untuk menciptakan efek cahaya yang bersinar pada objek atau hasil bayangan.

$$C_{result} = \frac{C_b}{1 - C_f}$$

Jika C_f mendekati 1, hasil mendekati putih. Sebaliknya, jika C_f mendekati 0, hasil mendekati warna background/latar belakang.

III. IMPLEMENTASI

A. Blurring

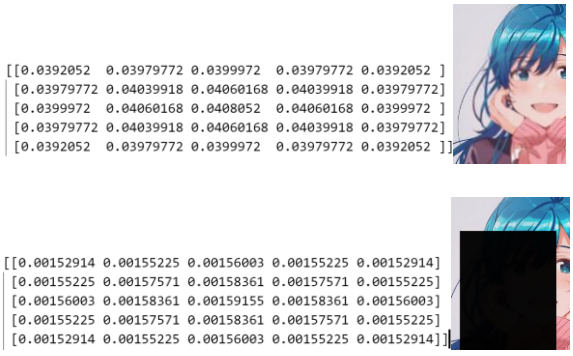
Implementasi efek blur diperlukan gambar keluaran yang terpisah. Memodifikasi gambar dari sumber secara langsung dapat memengaruhi nilai-nilai piksel berdekatan dan mengacaukan perhitungan pada iterasi selanjut sehingga efek blur menjadi tidak stabil dan sesuai. Tepi pada efek blur juga perlu ditangani karena suatu kernel akan melampui batas gambar dan akan menduplikasi tepi/membungkus gambar.

Nilai-nilai pada kernel yang akan digunakan pada efek blur juga perlu dinormalkan. Jika tidak, gambar akan menjadi gelap karena nilai-nilai total akan berjumlah sedikit kurang dari 1 [3]. Selain itu, ukuran kernel biasanya ganjil untuk memastikan adanya satu piksel yang menjadi pusatnya. Berikut kode untuk pembuatan kernel gaussian:

```python
def create_gaussian_kernel(radius, sigma):
    kernel_size = 2 * radius + 1
    gaussian_kernel = np.zeros((kernel_size, kernel_size), dtype=np.float32)
    for x in range(-radius, radius + 1):
        for y in range(-radius, radius + 1):
            #rumus gaussian kernel
            gaussian_kernel[x + radius, y + radius] = (1 / (2 * np.pi * sigma**2)) \
                * np.exp(-(x**2 + y**2) / (2 * sigma**2))

    # Normalisasi kernel
    gaussian_kernel /= np.sum(gaussian_kernel)
    print(gaussian_kernel)
    return gaussian_kernel
```

Gambar 3.1 Kode pembuatan kernel gaussian
(Sumber: Arsip Penulis)

Perbandingan nilai pada kernel dan hasil blurring yang menggunakanan normalisasi dan tidak pada radius 2px dan simpangan sebesar 10, dapat dilihat pada gambar berikut:



Gambar 3.2 Perbandingan nilai kernel dengan normalisasi (atas) dan tidak (bawah)
(Sumber: Arsip Penulis)

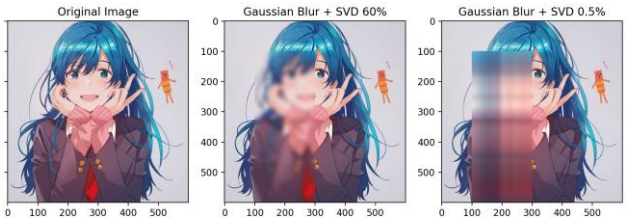
Setelah kernel Gaussian berhasil dibuat, proses konvolusi digunakan untuk menerapkan efek blur pada area gambar yang telah dipilih. Operasi ini hanya dilakukan pada region tertentu, seperti koordinat piksel (x: 100–300, y: 100–600).

```python
def apply_gaussian_blur(image, kernel, region):
    x1, y1, x2, y2 = region
    sub_image = image[y1:y2, x1:x2]
    k = kernel.shape[0] // 2
    padded_sub_image = cv2.copyMakeBorder(sub_image, k, k, k, k, borderType=cv2.BORDER_REFLECT)
    blurred_sub_image = np.zeros_like(sub_image)
    # Konvolusi
    for i in range(k, padded_sub_image.shape[0] - k):
        for j in range(k, padded_sub_image.shape[1] - k):
            for c in range(3):
                region = padded_sub_image[i - k:i + k + 1, j - k:j + k + 1, c]

                #perkalian dot product antara 2 vektor berbentuk matriks
                blurred_sub_image[i - k, j - k, c] = np.sum(region * kernel)

    result_image = image.copy()
    result_image[y1:y2, x1:x2] = blurred_sub_image
    return result_image

def apply_gaussian_blur_with_svd(image, kernel, region, svd_strength):
    result_image = apply_gaussian_blur(image, kernel, region)
    x1, y1, x2, y2 = region
    blurred_sub_image = result_image[y1:y2, x1:x2]

    for c in range(3):
        # Nilai singular values merepresentasikan tingkat kontribusi setiap komponen dalam representasi gambar.
        U, S, Vt = svd(blurred_sub_image[:, :, c], full_matrices=False)
        k_svd = int(len(S) * svd_strength)
        S[k_svd:] = 0
        reconstructed = np.dot(U, np.dot(np.diag(S), Vt))
        blurred_sub_image[:, :, c] = reconstructed.clip(0, 255).astype(np.uint8)

    result_image[y1:y2, x1:x2] = blurred_sub_image
    return result_image
```

Gambar 3.3 Kode Implementasi *blurring* dengan konvolusi dan dekomposisi SVD.
(Sumber: Arsip Penulis)

Metode SVD digunakan untuk menyimpan sebagian besar informasi penting pada matriks singular, seperti

tingkat kontribusi blur dari setiap komponen. Nilai singular terbesar membawa informasi utama, sedangkan nilai singular kecil merepresentasikan detail halus atau noise. Dengan mengatur nilai singular, kita dapat mengontrol tingkat detail yang hilang pada proses blurring.



Gambar 3.4 Hasil *blurring* dengan radius 15px dan simpangan sebesar 7
(Sumber: Arsip Penulis)

Setelah implementasi selesai, hasil blur dapat dibandingkan dengan hasil dari software seperti Photoshop dengan cara membandingkan beberapa piksel yang bersesuaian untuk mengevaluasi akurasi dari perhitungan.

```
[[[ 83  76  94]      [[[ 98  87 104]      [[[ 98  87 104]
  [ 92  81  97]       [105  92 109]       [105  92 109]
  [104  90 105]]      [114  98 111]]      [116  99 115]]

 [[ 83  74  91]      [[ 96  87 104]      [[ 98  87 104]
  [ 93  82  98]       [106  94 108]       [104  93 109]
  [105  91 104]]      [116 100 113]]      [118 102 115]]

 [[ 83  74  91]      [[ 97  88 105]      [[ 99  88 105]
  [ 96  84  98]       [107  95 109]       [106  95 111]
  [111  95 106]]]     [117 101 114]]]     [119 103 116]]]
```

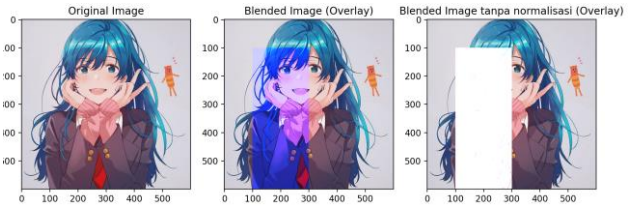
Gambar 3.5 Test hasil *blurring* yang paling mendekati dengan photshop dengan SVD 100%
(Sumber: Arsip Penulis)

Piksel yang dibandingkan dalam gambar memiliki ukuran 3 x 3 piksel. Hasil efek blur pada Photoshop diperlihatkan pada bagian kanan gambar menggunakan radius Gaussian Blur sebesar 5 piksel. Pada contoh ini, array RGB di bagian kiri merupakan hasil blur dari program dengan SVD 100%, radius 5 piksel, dan nilai simpangan 5. Namun, hasil di tengah dengan radius 11 piksel dan simpangan sebesar 5,5 dengan SVD yang sama menunjukkan nilai RGB yang lebih mendekati nilai yang dihasilkan Photoshop.

Setelah berbagai percobaan menggunakan radius sebesar 2r+1 pada program dengan r merupakan radius blur pada photoshop, tidak ditemukan perbedaan yang signifikan dengan selisih lebih besar dari 5 pada setiap pikselnya. Hal ini menjadi asumsi bahwa Photoshop menerapkan radius kernel sebesar r+1 lebih besar dari radius pada program dan menggunakan nilai simpangan sebesar setengah dari ukuran kernel yang terbentuk.

B. Blending Modes

Pada implementasi blending mode seperti Overlay, Color Dodge, dan Multiply, normalisasi dilakukan seperti pada kernel Gaussian untuk mencegah overflow pada nilai bit warna. Nilai piksel dinormalisasikan dalam rentang 0.0–1.0. Operasi blending mode sebagian besar menggunakan perkalian antara nilai piksel. Jika tidak dilakukan normalisasi, hasilnya dapat melebihi 255 dan menjadi full white saat diimplementasikan seperti pada gambar di bawah.



Gambar 3.6 Perbandingan hasil *blend* dengan normalisasi dan tidak
(Sumber: Arsip Penulis)

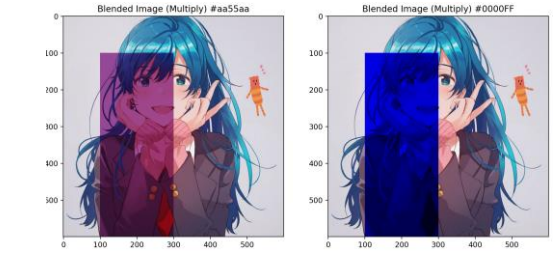
Setelah proses blending selesai, hasil akhirnya perlu dikembalikan dalam rentang valid [0–255]. Hal ini memastikan hasil dapat divisualisasikan atau disimpan dengan benar. Kode blending dari ketiga operasi dapat dilihat pada gambar berikut:

```python
def blending_modes(image, region, hex_color, mode):
    # Konversi warna hex ke RGB
    blend_color = np.array(hex_to_rgb(hex_color), dtype=np.float32)

    x1, y1, x2, y2 = region
    sub_image = image[y1:y2, x1:x2].astype(np.float32)

    # Normalisasi warna blending dan image (0-1)
    blend_color /= 255.0
    sub_image /= 255.0

    # Terapkan blending mode
    if mode == "multiply":
        result = sub_image * blend_color
    elif mode == "overlay":
        mask = sub_image <= 0.5
        result = np.where(mask, 2 * sub_image * blend_color, 1 - 2 * (1 - sub_image) * (1 - blend_color))
    elif mode == "colordodge":
        result = np.minimum(1.0, sub_image / (1 - blend_color + 1e-6))

    # Denormalisasi hasil (0-255) dan konversi kembali
    result = (result * 255).clip(0, 255).astype(np.uint8)
    # Mengganti hasil blend ke gambar asli
    blended_image = image.copy()
    blended_image[y1:y2, x1:x2] = result
    return blended_image
```

Gambar 3.7 Kode implementasi *blending mode*
(Sumber: Arsip Penulis)

a) Multiply

Proses blending multiply mencampurkan piksel lapisan base dengan piksel lapisan blend melalui perkalian langsung. Hasilnya menghasilkan efek gelap yang sesuai dengan komponen warna masing-masing.



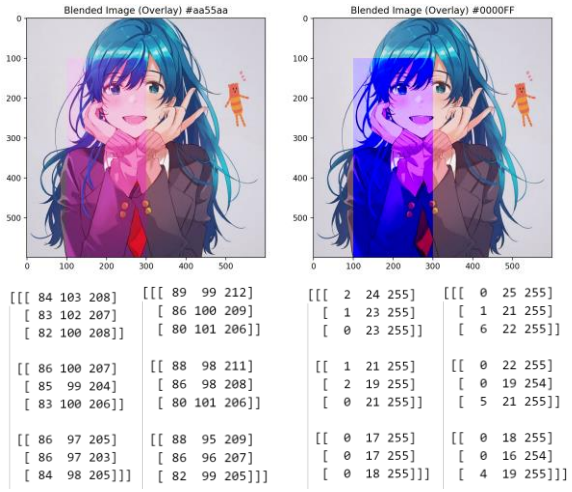

```
[[[ 44  47 126]      [[[ 41  48 128]      [[[  0   0 186]      [[[  0   0 186]
  [ 43  46 125]       [ 42  48 126]       [  0   0 186]       [  0   0 184]
  [ 43  44 124]]      [ 41  47 123]]      [  0   0 182]]      [  0   0 182]]

 [[ 43  46 125]      [[ 40  47 127]      [[  0   0 184]      [[  0   0 186]
  [ 43  46 125]       [ 41  47 125]       [  0   0 184]       [  0   0 184]
  [ 43  44 124]]      [ 41  47 123]]      [  0   1 180]]      [  0   0 182]]

 [[ 43  46 123]      [[ 41  46 127]      [[  1   0 182]      [[  0   0 184]
  [ 44  46 123]       [ 40  46 124]       [  1   0 182]       [  0   0 182]
  [ 43  45 122]]]     [ 42  45 122]]]     [  1   0 180]]]     [  0   1 180]]]
```

Gambar 3.8 Perbandingan piksel RGB hasil program (kiri) dan photoshop (kanan) pada *multiply blending*

(Sumber: Arsip Penulis)

b) Overlay

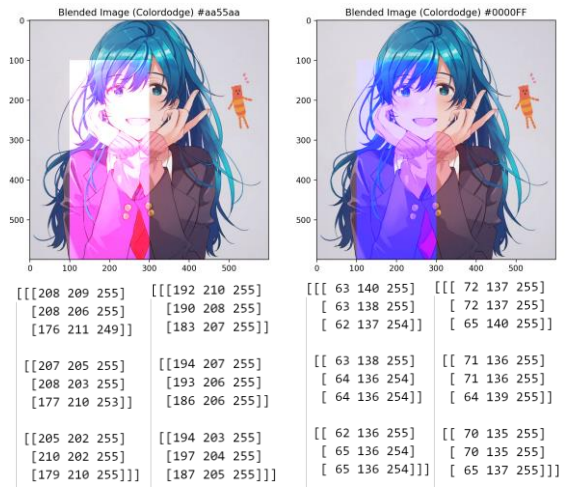
Proses blending overlay dengan cara piksel dengan intensitas rendah diperkuat melalui perkalian langsung, sementara piksel dengan intensitas tinggi disesuaikan menggunakan inversi dan perkalian.



Gambar 3.9 Perbandingan piksel RGB hasil program (kiri) dan photoshop (kanan) pada *overlay blending*
(Sumber: Arsip Penulis)

c) Color Dodge

Proses blending Color Dodge meningkatkan kecerahan gambar dengan membagi nilai piksel lapisan base dengan inversi piksel lapisan blend, dengan perhatian khusus pada kondisi dimana nilai denominator mendekati nol untuk menghindari nilai tak terhingga sehingga harus diberikan offset kecil seperti 1e-6.



Gambar 3.10 Perbandingan piksel RGB hasil program (kiri) dan photoshop (kanan) pada *color dodge blending*
(Sumber: Arsip Penulis)

IV. KESIMPULAN

Digital Painting Software merupakan tools penting bagi para seniman yang senang membuat karya secara digital. Fitur-fitur yang diberikan oleh software digital membuat para seniman menjadi lebih mudah dalam membuat karya yang lebih berkualitas dan bervariasi. Dibalik pembuatan karya digital tersebut, ternyata terdapat operasi matematis seperti SVD dan perkalian vector dalam penerapannya. Bahasan sebelumnya telah menunjukan tiap operasi matematis yang diterapkan pada gambar dengan efek tertentu, tidak jauh berbeda dengan yang dihasilkan oleh digital painting software tersebut seperti photoshop. Sehingga hal ini menjadi bukti bahwa penerapan matematis dalam dunia digital painting benar adanya.

IV. UCAPAN TERIMA KASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa atas rahmat dan kelancaran yang diberikan sehingga penulis dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Penulis juga menyampaikan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T., selaku dosen pengampu mata kuliah IF2123 Aljabar dan Geometri Linear 2024 Kelas 02, atas ilmu dan bimbingan yang telah diberikan selama perkuliahan berlangsung. Ilmu yang beliau sampaikan sangat membantu penulis dalam memahami materi kuliah serta menyelesaikan tugas makalah ini.

Tak lupa, penulis juga mengucapkan terima kasih kepada kedua orang tua yang telah memberikan dukungan selama proses perkuliahan. Penulis berharap makalah ini dapat memberikan kontribusi positif dalam memahami lebih dalam materi yang telah dipelajari.

REFERENSI

[1] L. Guo, "Digital Painting Art Design Based on Computer Aided Technology," 2022 IEEE 2nd International Conference on Mobile Networks and Wireless Communications (ICMNWC), Tumkur, Karnataka, India, 2022, pp. 1-5.

[2] E. S. Gedraite and M. Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation," Proceedings ELMAR-2011, Zadar, Croatia, 2011, pp. 393-396.

[3] A. Sharda, "Image Filters: Gaussian Blur," Medium, 2021. [Online]. Tersedia: https://aryamansharda.medium.com/image-filters-gaussian-blur-eb36db6781b1. [Diakses: 30-Des-2024].

[4] Md. Jewel, "Understanding Digital Images for Image Processing and Computer Vision (Part 1)," Medium, 2023. [Online]. Tersedia: https://medium.com/@md-jewel/understanding-digital-images-for-image-processing-and-computer-vision-part-1-cc42be78cca1. [Diakses: 30-Des-2024].

[5] Snapshot, "Videography FAQ: What is bit depth? How does it affect my video?" Canon Asia, 2022. [Online]. Tersedia: https://snapshot.canon-asia.com/id/article/indo/videography-faq-what-is-bit-depth-how-does-it-affect-my-video. [Diakses: 31-Des-2024].

[6] Blending Modes Explained," Photoshop Training Channel, 18 Agustus 2022. [Online]. Tersedia: https://photoshoptrainingchannel.com/blending-modes-explained/. [Diakses: 1-Jan-2025].

[7] R. Thomas, "Photoshop Blend Modes Explained," Photoblogstop. [Online]. Tersedia: https://photoblogstop.com/photoshop/photoshop-blend-modes-explained. [Diakses: 1-Jan-2025].

[8] R. Munir, "Aljabar Geometri – Tahun Ajaran 2024/2025," Institut Teknologi Bandung (ITB). [Online]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/algeo24-25.htm. [Diakses: 30-Des-2024].

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, 02 Januari 2025

Muhammad Zahran Ramadhan Ardiana
13523104