Penerapan Matriks dalam Implementasi Gaussian Filter

Orvin Andika, 13523017[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13523017@mahasiswa.itb.ac.id, [2]orvin.andika@gmail.com

***Abstrak*—Filter Gaussian merupakan salah satu metode yang digunakan untuk mengurangi noise dan menghasilkan citra yang lebih halus. Makalah ini membahas tentang hubungan antara ukuran matriks kernel, sigma , dan citra yang dihasilkan. Metode yang digunakan adalah perhitungan matriks pada distribusi Gaussian dan pengujian di beberapa jenis citra dengan tingkat noise yang berbeda. Semakin besar ukuran matriks kernel dan sigma maka citra yang dihasilkan akan semakin halus dan efek blurnya juga semakin tinggi. Makalah ini membahas tentang konsep dasar Filter Gaussian dan bagaimana matriks digunakan dalam implementasinya.**

***Kata Kunci*—Filter Gaussian, blur, kernel, sigma**

I. Pendahuluan

Dalam dunia teknologi dan informasi pengolahan gambar merupakan salah satu bidang yang berkembang pesat. Pada pengolahan citra terdapat penggunaan filter untuk mengolah data citra agar menghasilkan informasi yang lebih berguna. Banyak teknik yang dikembangkan untuk meningkatkan dan menyederhanakan sebuah gambar, salah satunya adalah penggunaan Filter Gaussian. Filter ini digunakan untuk mengurangi noise pada gambar sehingga kualitas gambar dapat meningkat.

Filter Gaussian di proses dengan menggunakan fungsi Gaussian. Fungsi tersebut adalah fungsi matematika berbentuk lonceng dan menghasilkan bobot piksel berdasar dengan jarak relatifnya ke pusat piksel. Dalam implementasinya, ada salah satu aspek penting yang harus diperhatikan yaitu penggunaan matriks dalam pemrosesannya. Matriks memiliki peran penting dalam proses untuk menerapkan filter pada gambar. Proses ini juga memerlukan pemahaman tentang operasi matriks seperti perkalian dan yang lainnya. Konsep dasar matriks ini sangat penting untuk dipahami agar dapat menerapkan Filter Gaussian dengan efektif.

Filter Gaussian menggunakan fungsi Gaussian untuk menghasilkan kernel filter. Fungsi ini berdasar pada distribusi normal dan memiliki sifat simetri dan sentralitas, yang memungkinkannya mencapai efek penghalusan secara optimal.

Makalah ini membahas tentang konsep dasar Filter Gaussian dan bagaimana matriks digunakan dalam implementasinya.

II. Landasan Teori

A. Matriks

Matriks adalah sebuah elemen dua dimensi yang tersusun dalam baris dan kolom. Matriks biasanya dinyatakan dengan huruf kapital seperti A dan elemennya dinyatakan dengan aij, dengan i adalah baris dan j adalah kolom.

Matriks digunakan untuk merepresentasikan data atau hubungan antar elemen dalam berbagai bidang. Dalam pemrosesan citra, matriks digunakan untuk merepresentasikan suatu gambar, dengan tiap elemen matriks adalah nilai tiap piksel dalam gambar tersebut.

B. Citra

Citra atau gambar adalah representasi visual dari suatu objek atau peristiwa yang berbentuk digital. Dalam bentuk digital, citra direpresentasikan sebagai matriks yang tiap elemennya adalah nilai piksel dari citra tersebut.

Citra memiliki beberapa jenis. Citra grayscale adalah citra yang hanya mengandung tingkat keabu-abuan. Sementara itu, citra berwarna dapat dibagi menjadi tiga matriks terpisah yaitu matriks merah, hijau, dan biru. Informasi ini memungkinkan komputer untuk memahami suatu citra sebelum diproses lebih lanjut. Untuk implementasi filter Gaussian, hanya dibutuhkan citra grayscale dari suatu gambar. Citra memiliki beberapa komponen utama, antara lain kecerahan, kontur, kontras, dan warna.

C. Piksel

Piksel adalah elemen terkecil yang menyusun suatu citra digital. Piksel menyimpan informasi tentang tentang warna atau tingkat kecerahan dalam suatu citra. Pada gambar grayscale, piksel tidak memiliki informasi tentang warna dan hanya memiliki informasi tentang tingkat kecerahan saja. Sementara itu, pada gambar berwarna piksel memiliki tiga nilai yang merepresentasikan intensitas warna merah, hijau, dan biru.

Semakin banyak piksel suatu gambar, maka akan semakin tinggi resolusinya. Resolusi yang tinggi akan membuat gambar semakin tajam dan detail.

Piksel adalah dasar dari semua operasi pemrosesan gambar. Setiap operasi pemrosesan gambar, seperti filter Gaussian ini pasti akan dilakukan dengan manipulasi pada nilai piksel.

D. Noise

Noise adalah gangguan atau penyimpangan pada suatu citra digital. Noise mungkin disebabkan oleh pencahayaan yang buruk, kompresi data, atau kualitas kamera yang buruk. Ada beberapa jenis noise yang sering muncul. Gaussian noise adalah noise yang disebabkan oleh distribusi acak, Salt-and-Pepper noise adalah noise yang muncul sebagai titik-titik hitam atau putih dalam gambar, dan Speckle noise yang sering terjadi dalam citra ultrasonik.

Noise dapat mengurangi kualitas gambar. Kualitas gambar yang rendah akan membuat pemrosesan citra menjadi lebih sulit. Oleh karena itu, penghilangan noise perlu dilakukan sebelum melanjutkan pemrosesan gambar.

E. Pemrosesan Citra

Pemrosesan citra adalah cabang ilmu komputer dan matematika yang berkaitan dengan manipulasi, analisis, dan pengolahan gambar digital. Tujuan dari pemrosesan citra adalah untuk meningkatkan kualitas gambar, mengidentifikasi informasi penting, dan mempersiapkan citra untuk analisis lebih lanjut.

Pemrosesan citra dibagi menjadi beberapa tahap. Tahapan pertama adalah *preprocessing.* Tahapan ini bertujuan untuk mengurangi noise dan meningkatkan kualitas citra. Filter Gaussian bisa dimanfaatkan untuk tahapan ini. Kedua adalah *enhancement.* Tahapan ini bertujuan untuk meningkatkan fitur tertentu agar lebih mudah dianalisis lebih lanjut. Ketiga adalah *segmentation.* Tahapan ini bertujuan untuk memisahkan antara objek di dalam citra dengan latar belakangnya. Keempat adalah *feature extraction.* Tahapan ini bertujuan untuk mengambil informasi-informasi dari dalam gambar. Terakhir adalah *recognition.* Ini adalah tahap untuk mengenali pola atau objek di dalam gambar

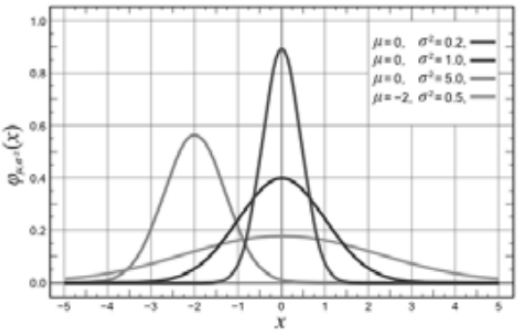
F. Filter Gaussian

Filter Gaussian adalah filter yang sering digunakan dalam pemrosesan gambar untuk menghaluskan dan mengurangi noise. Filter ini menghitung nilai piksel baru berdasarkan kontribusi piksel-piksel di sekitarnya. Kontribusi ini dihitung dari bobot yang ditentukan dengan fungsi Gaussian[3].

Untuk pemrosesan gambar, fungsi Gaussian yang digunakan adalah fungsi Gaussian dua dimensi[2]. Rumus fungsi tersebut adalah:

$$h(m,n) = \frac{1}{2\pi\sigma^2} e^{\frac{-(m^2+n^2)}{2\sigma^2}}$$

Dengan σ adalah standar deviasi distribusi normal yang digunakan untuk menentukan lebar distribusi Gaussian, semakin besar standar deviasi maka akan semakin banyak titik tetangga yang dimasukkan dalam perhitungan. Sementara itu, m dan n merupakan koordinat relatif terhadap pusat kernel.



Gambar 2.1 Kurva Distribusi Normal, diambil dari [1]

G. Konvolusi

Dalam konteks pengolahan citra, konvolusi adalah teknik untuk menghaluskan atau memperjelas citra dengan menggantikan nilai piksel dengan sejumlah nilai piksel yang sesuai atau berdekatan dengan piksel aslinya. Untuk pengolahan gambar, fungsi konvolusi yang digunakan adalah fungsi diskrit karena nilai piksel suatu gambar adalah nilai-nilai diskrit[4] . Berikut adalah rumus konvolusi yang digunakan dalam pengolahan citra dua dimensi.

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{a=-\infty}^{\infty} f(a,b) \cdot g(x-a, y-b)$$

III. Implementasi

Dengan rumus distribusi Gaussian, bisa dibuat sebuah algoritma yang menerima input ukuran dan sigma untuk membuat matriks kernel secara otomatis dalam bahasa pemrograman Python. Python dipilih karena banyaknya library yang bisa digunakan untuk mempermudah pembuatan algoritma. Misalnya, library numpy yang bisa digunakan untuk mempermudah perhitungan dan pembuatan matriks. Selain itu, library cv2 dapat mempermudah dalam memproses, menyimpan, dan

membuka file gambar sementara library scipy dapat digunakan untuk mempermudah proses konvolusi.

Implementasi matriks kernel dilakukan dengan fungsi gaussian kernel yang menerima parameter size dan sigma. Size adalah ukuran matriks kernel yang diharapkan sementara sigma adalah standar deviasinya. Fungsi ini menggunakan library numpy untuk mempermudah perhitungan. Di akhir perhitungan, dilakukan normalisasi dengan membagi setiap elemen dengan jumlah seluruh elemen agar jumlah seluruh elemen sama dengan 1.

```python
def gaussian_kernel(size, sigma):
    """
    Membuat kernel Gaussian.
    :param size: Ukuran kernel (size x size)
    :param sigma: Standar deviasi Gaussian
    :return: Matriks kernel Gaussian
    """
    k = size // 2
    x, y = np.meshgrid(np.arange(-k, k + 1), np.arange(-k, k + 1))
    kernel = (1 / (2 * np.pi * sigma**2)) * np.exp(-(x**2 + y**2) / (2 * sigma**2))
    return kernel / kernel.sum()
```

Gambar 3.1 Implementasi Fungsi Matriks Kernel

Untuk perbandingan dibutuhkan lima matriks kernel masing-masing dengan size dan sigma yang dibedakan.

Matriks kernel pertama memiliki ukuran 3x3 dengan sigma 1. Hasil perhitungan matriks dengan fungsi gaussian_kernel menghasilkan matriks berikut:

```
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
```

Gambar 3.2 Matriks Kernel 3x3 Sigma 1

Matriks kernel kedua memiliki ukuran 5x5 dengan sigma 1. Hasil perhitungan matriks dengan fungsi gaussian_kernel menghasilkan matriks berikut:

```
[[0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]
 [0.01330621 0.0596343  0.09832033 0.0596343  0.01330621]
 [0.02193823 0.09832033 0.16210282 0.09832033 0.02193823]
 [0.01330621 0.0596343  0.09832033 0.0596343  0.01330621]
 [0.00296902 0.01330621 0.02193823 0.01330621 0.00296902]]
```

Gambar 3.3 Matriks Kernel 5x5 Sigma 1

Matriks kernel ketiga memiliki ukuran 7x7 dengan sigma 1. Hasil perhitungan matriks dengan fungsi gaussian_kernel menghasilkan matriks berikut:

```
[[1.96519161e-05 2.39409349e-04 1.07295826e-03 1.76900911e-03
  1.07295826e-03 2.39409349e-04 1.96519161e-05]
 [2.39409349e-04 2.91660295e-03 1.30713076e-02 2.15509428e-02
  1.30713076e-02 2.91660295e-03 2.39409349e-04]
 [1.07295826e-03 1.30713076e-02 5.85815363e-02 9.65846250e-02
  5.85815363e-02 1.30713076e-02 1.07295826e-03]
 [1.76900911e-03 2.15509428e-02 9.65846250e-02 1.59241126e-01
  9.65846250e-02 2.15509428e-02 1.76900911e-03]
 [1.07295826e-03 1.30713076e-02 5.85815363e-02 9.65846250e-02
  5.85815363e-02 1.30713076e-02 1.07295826e-03]
 [2.39409349e-04 2.91660295e-03 1.30713076e-02 2.15509428e-02
  1.30713076e-02 2.91660295e-03 2.39409349e-04]
 [1.96519161e-05 2.39409349e-04 1.07295826e-03 1.76900911e-03
  1.07295826e-03 2.39409349e-04 1.96519161e-05]]
```

Gambar 3.4 Matriks Kernel 7x7 Sigma 1

Matriks kernel keempat memiliki ukuran 5x5 dengan sigma 0.5. Hasil perhitungan matriks dengan fungsi gaussian_kernel menghasilkan matriks berikut:

```
[[6.96247819e-08 2.80886418e-05 2.07548550e-04 2.80886418e-05
  6.96247819e-08]
 [2.80886418e-05 1.13317669e-02 8.37310610e-02 1.13317669e-02
  2.80886418e-05]
 [2.07548550e-04 8.37310610e-02 6.18693507e-01 8.37310610e-02
  2.07548550e-04]
 [2.80886418e-05 1.13317669e-02 8.37310610e-02 1.13317669e-02
  2.80886418e-05]
 [6.96247819e-08 2.80886418e-05 2.07548550e-04 2.80886418e-05
  6.96247819e-08]]
```

Gambar 3.5 Matriks Kernel 5x5 Sigma 0.5

Matriks kernel kelima memiliki ukuran 5x5 dengan sigma 2. Hasil perhitungan matriks dengan fungsi gaussian_kernel menghasilkan matriks berikut:

```
[[0.02324684 0.03382395 0.03832756 0.03382395 0.02324684]
 [0.03382395 0.04921356 0.05576627 0.04921356 0.03382395]
 [0.03832756 0.05576627 0.06319146 0.05576627 0.03832756]
 [0.03382395 0.04921356 0.05576627 0.04921356 0.03382395]
 [0.02324684 0.03382395 0.03832756 0.03382395 0.02324684]]
```

Gambar 3.6 Matriks Kernel 5x5 Sigma 2

Implementasi penerapan filter gaussian dilakukan melalui gaussian_filter. Fungsi ini menerima parameter image dan kernel. Image adalah gambar asli yang ingin diproses dengan filter gaussian. Kernel adalah matriks kernel yang telah dibuat di fungsi sebelumnya sesuai ukuran dan standar deviasinya. Fungsi ini memanfaatkan library scipy untuk memudahkan proses konvolusi.

```python
def gaussian_filter(image, kernel):
    """
    Menerapkan Gaussian filter pada citra.
    :param image: Citra input
    :param kernel: Kernel Gaussian
    :return: Citra hasil filter
    """
    return convolve(image, kernel)
```

Gambar 3.7 Implementasi Gaussian Filter

Contoh pemanggilan kedua fungsi dapat dilihat pada gambar berikut.

```python
image = cv2.imread('path_to_image', cv2.IMREAD_GRAYSCALE)
size = 5
sigma = 2
kernel = gaussian_kernel(size, sigma)
print(kernel)
filtered_image = gaussian_filter(image, kernel)
output_filename = f'filtered_image_kernel_{size}_sigma_{sigma}.jpg'
cv2.imwrite(output_filename, filtered_image)

cv2.imshow('Original Image', image)
cv2.imshow('Filtered Image', filtered_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Gambar 3.8 Implementasi Pemanggilan Gaussian Filter

Implementasi dilakukan dengan memanfaatkan library cv2. Library cv2 digunakan dalam beberapa tahap. Pertama, library cv2 digunakan dalam tahap membuka file gambar yang ingin diubah. Kedua, library cv2 dimanfaatkan untuk mengubah gambar tadi menjadi gambar grayscale agar lebih mudah dilakukan proses selanjutnya. Size dan sigma dapat diganti sesuai nilai yang diharapkan. Kemudian, fungsi gaussian_kernel dipanggil dengan parameter size dan sigma dan hasilnya disimpan dalam variabel kernel. Setelah itu, fungsi gaussian_filter dipanggil dengan parameter image asli dan kernel. Hasil dari fungsi tersebut disimpan dalam variabel filtered_image. Image asli dan filtered_image kemudian ditampilkan dalam window popup dengan memanfaatkan library cv2.

IV. Hasil dan Pembahasan

Berikut adalah gambar asli yang digunakan untuk mengetes algoritma filter gaussian.

Gambar 4.1 Gambar Asli

Percobaan pertama dilakukan dengan ukuran matriks kernel sebagai variabel bebas dan sigma sebagai variabel kontrol.

Berikut adalah hasil dari tes dengan ukuran 3x3 dan sigma 1 untuk mengetes algoritma filter gaussian.

Gambar 4.2 Gambar Kernel 3x3 Sigma 1

Terlihat bahwa noise pada gambar tersebut sudah berkurang jika dibandingkan dengan gambar asli. Terlihat juga bahwa efek blurnya masih rendah. Namun, noise belum sepenuhnya hilang.

Berikut adalah hasil dari tes dengan ukuran 5x5 dan sigma 1 untuk mengetes algoritma filter gaussian.

Gambar 4.3 Gambar Kernel 5x5 Sigma 1

Terlihat bahwa noise pada gambar tersebut sudah berkurang jika dibandingkan dengan gambar asli tetapi perbedaannya dengan gambar dengan ukuran kernel 3x3 tidak terlalu signifikan. Hal ini mungkin terjadi karena hasil dari fungsi dengan ukuran kernel 3x3 sudah cukup halus dan tidak mengandung banyak noise sehingga ukuran kernel yang lebih besar tidak memiliki pengaruh yang besar.

Berikut adalah hasil dari tes dengan ukuran 7x7 dan

sigma 1 untuk mengetes algoritma filter gaussian.

Gambar 4.4 Gambar Kernel 7x7 Sigma 1

Terlihat bahwa noise pada gambar tersebut sudah berkurang jika dibandingkan dengan gambar asli tetapi perbedaannya dengan gambar dengan ukuran kernel 3x3 dan 5x5 tidak terlalu signifikan. Perbedaan yang cukup terlihat mungkin hanya dari efek blurnya yang sedikit meningkat. Hal ini mungkin terjadi karena hasil dari fungsi dengan ukuran kernel 3x3 sudah cukup halus dan tidak mengandung banyak noise sehingga ukuran kernel yang lebih besar tidak memiliki pengaruh yang besar.

Percobaan kedua dilakukan dengan ukuran matriks kernel sebagai variabel kontrol dan sigma sebagai variabel bebas.

Berikut adalah hasil dari tes dengan ukuran 5x5 dan sigma 0.5 untuk mengetes algoritma filter gaussian.

Gambar 4.5 Gambar Kernel 5x5 Sigma 0.5

Terlihat gambar masih terlihat sangat kasar dan noise belum banyak hilang. Hal ini dikarenakan semakin kecil nilai sigma maka efek penghalusan akan semakin kecil dan hanya akan memengaruhi piksel yang lebih dekat dengan pusat kernel.

Berikut adalah hasil dari tes dengan ukuran 5x5 dan sigma 1 untuk mengetes algoritma filter gaussian.

Gambar 4.6 Gambar Kernel 5x5 Sigma 1

Terlihat bahwa gambar terlihat lebih halus jika dibandingkan dengan gambar asli dan gambar dengan sigma 0.5. Hal ini disebabkan oleh nilai sigma yang membesar sehingga distribusi bobot akan lebih menyebar sehingga piksel yang lebih jauh akan memiliki bobot yang lebih besar. Ini menyebabkan efek penghalusan yang semakin kuat.

Berikut adalah hasil dari tes dengan ukuran 5x5 dan sigma 2 untuk mengetes algoritma filter gaussian.

Gambar 4.7 Gambar Kernel 5x5 Sigma 2

Terlihat bahwa gambar ini memiliki efek blur yang lebih kuat dibandingkan gambar-gambar sebelumnya. Hal ini juga terjadi karena nilai sigmanya yang membesar sehingga distribusi bobot akan lebih menyebar sehingga piksel yang lebih jauh akan memiliki bobot yang lebih besar. Ini menyebabkan efek efek penghalusan yang sangat kuat dan menghilangkan ketajaman gambar.

Dari kedua percobaan, terlihat bahwa pengaruh dari standar deviasi lebih besar daripada ukuran kernel. Nilai standar deviasi mengontrol seberapa lebar distribusi Gaussian dalam kernel. Semakin besar standar deviasi, maka semakin lebar distribusi bobotnya sehingga akan semakin kuat pula efek penghalusannya. Sementara itu, ukuran kernel mengontrol seberapa banyak piksel yang dihitung dalam proses konvolusi. Semakin besar ukuran kernel maka jumlah piksel yang dihitung dalam satu operasi akan semakin banyak juga. Meskipun jumlah piksel yang dihitung banyak, jika standar deviasinya kecil tidak selalu menghasilkan efek penghalusan yang kuat. Kernel yang lebih besar hanya akan menyebarkan bobot lebih luas tetapi yang lebih menentukan kuat atau tidaknya efek penghalusan adalah distribusi bobot yang diatur oleh besarnya standar deviasi.

V. Kesimpulan

Eksperimen yang dilakukan menunjukkan bahwa filter Gaussian merupakan metode yang bagus untuk menghilangkan noise dan memperhalus gambar.

Untuk mendapatkan hasil maksimal, perlu mengatur ukuran kernel dan besar standar deviasinya. Jika terlalu kecil maka gambar akan tetap kasar dan banyak noise. Akan tetapi jika terlalu besar, efek blurnya akan terlalu kuat sehingga detail gambar akan hilang.

Pengaruh standar deviasi terhadap efek penghalusan lebih tinggi daripada ukuran kernel. Ukuran kernel yang besar hanya berarti menyebarkan bobot lebih luas. Distribusi bobot ditentukan oleh besarnya standar deviasi sehingga pengaruhnya lebih besar

Penelitian dapat dikembangkan dengan mengombinasikan filter Gaussian dengan metode lain seperti deteksi tepi atau segmentasi citra untuk mendapatkan hasil yang lebih komprehensif.

VI. Ucapan Terima Kasih

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa
2. Dosen pengampu mata kuliah Aljabar Linier dan Geometri
3. Teman-teman penulis
4. Pihak-pihak lain

yang telah membantu penulis dalam menyelesaikan makalah ini

References

[1] N. K. Majid, M. Hariadi, and S. Mardi, "Distribusi Gaussian Perilaku Tarung NPC Prajurit pada Game Peperangan Menggunakan Metode Box-Muller," in *Seminar Nasional Pascasarjana X*, Surabaya, Indonesia, Aug. 2010, ISBN: 979-545-0270-1.

[2] H. Sunandar, "Perbaikan kualitas citra menggunakan metode Gaussian Filter," *MEANS (Media Informasi Analisa dan Sistem)*, vol. 2, no. 1, p. 19, June 2017, ISSN: 2548-6985.

[3] M. Mafi, H. Martin, M. Cabrerizo, J. Andrian, A. Barreto, and M. Adjouadi, "A comprehensive survey on impulse and Gaussian denoising filters for digital images," *Signal Processing*, vol. 157, pp. 236-260, Apr. 2019. DOI: 10.1016/j.sigpro.2018.12.006.

[4] R. Rambe, "Perbaikan kualitas citra digital menggunakan metode kernel konvolusi," *Terapan Informatika Nusantara (TIN)*, vol. 1, no. 11, pp. 557-562, Apr. 2021. [Online]. Available: https://ejurnal.seminar-id.com/index.php/tin.

[5] R. Munir, 'Homepage Rinaldi Munir'. https://informatika.stei.itb.ac.id/~rinaldi.munir/. [Diakses 29 Desember 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.

Yogyakarta, 29 Desember 2024

Orvin Andika
13523017