Pemanfaatan Algoritma Divide and Conquer pada Median Filtering dengan Perbandingan SSIM dalam Citra Digital

Moh Fairuz Alauddin Yahya - 13522057
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13522057@std.stei.itb.ac.id

***Abstrak*— Dalam pemrosesan citra digital, pengurangan noise merupakan langkah penting untuk meningkatkan kualitas gambar. Salah satu teknik yang umum digunakan adalah Median Filtering, yang bekerja dengan mengambil nilai median piksel dalam suatu area tertentu di sekitar setiap piksel dalam gambar. Teknik ini efektif dalam mengurangi noise sambil mempertahankan detail penting dalam gambar. Namun, implementasi Median Filtering seringkali memerlukan komputasi yang intensif. Dalam usaha untuk meningkatkan efisiensi, penggunaan algoritma Divide and Conquer telah diusulkan. Algoritma ini membagi citra menjadi bagian-bagian kecil, menerapkan Median Filtering pada setiap bagian, dan kemudian menggabungkan hasilnya. Pendekatan ini diharapkan dapat meningkatkan kinerja proses denoising.**

Kata kunci*— *Denoising; Median Filtering; Divide and Conquer; pemrosesan citra digital.

I. PENDAHULUAN

Dalam dunia pemrosesan citra digital, pengurangan noise atau denoising adalah proses penting untuk meningkatkan kualitas gambar. Noise, yang dapat muncul akibat gangguan sensor, kompresi data, atau kondisi lingkungan, dapat mengaburkan detail penting dalam gambar dan mengurangi ketajaman serta kejelasannya.

Dalam upaya untuk mengatasi tantangan ini, berbagai teknik telah dikembangkan untuk mengurangi atau menghilangkan noise dari citra digital. Salah satu teknik yang umum digunakan adalah Median Filtering. Teknik ini bekerja dengan cara mengambil nilai median dari piksel-piksel dalam suatu area tertentu di sekitar setiap piksel dalam gambar. Dengan cara ini, nilai yang ekstrem atau noise yang mencolok dapat dihilangkan, sementara detail dan struktur gambar yang penting tetap dipertahankan.

Tujuan utamanya adalah untuk mempertahankan informasi penting sambil mengurangi gangguan yang tidak diinginkan. Median Filtering telah terbukti efektif dalam mengurangi berbagai jenis noise dalam gambar digital, menjadikannya salah satu alat penting dalam toolbox pemrosesan citra.

Penggunaan algoritma Divide and Conquer bertujuan untuk meningkatkan efisiensi dan kinerja proses filtering. Teknik Divide and Conquer membagi masalah besar menjadi submasalah yang lebih kecil, memprosesnya secara terpisah, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir. Dalam konteks Median Filtering, ini dapat dilakukan dengan membagi citra menjadi bagian-bagian kecil, menerapkan median filtering pada masing-masing bagian secara terpisah, dan kemudian menggabungkan hasilnya.

Proses ini mengurangi kompleksitas perhitungan dengan membatasi jangkauan pencarian untuk setiap piksel, karena algoritma hanya perlu mempertimbangkan piksel di sekitarnya dalam submasalah yang lebih kecil. Hal ini dapat mengurangi waktu komputasi secara signifikan, terutama pada gambar dengan resolusi tinggi.

Penelitian ini bertujuan untuk mencari teknik yang dapat memperbaiki kualitas gambar dengan cara menghilangkan noise tanpa mengorbankan detail yang penting dalam gambar, serta memberikan kontribusi pada pengembangan teknik denoising yang lebih baik dan lebih cepat dalam pemrosesan gambar digital.

II. LANDASAN TEORI

A. *Divide and Conquer*

Algoritma Divide and Conquer adalah paradigma pemecahan masalah yang melibatkan pemecahan masalah besar menjadi bagian-bagian yang lebih kecil yang kemudian dipecahkan secara terpisah. Setiap bagian yang lebih kecil dipecahkan secara rekursif sampai menjadi masalah yang cukup sederhana untuk dipecahkan dengan mudah. Setelah bagian-bagian kecil tersebut dipecahkan, solusi-solusi mereka digabungkan kembali untuk memberikan solusi keseluruhan dari masalah yang besar. [1]

Langkah-langkah umum dalam algoritma Divide and Conquer adalah sebagai berikut:

1. Divide (Pembagian): Masalah utama dibagi menjadi beberapa masalah yang lebih kecil dan terpisah.

Langkah ini dilakukan hingga masalah menjadi cukup kecil untuk diselesaikan secara langsung.

2. Conquer (Penaklukan): Setiap masalah yang lebih kecil dipecahkan secara rekursif hingga mencapai ukuran yang dapat ditangani secara langsung. Biasanya, langkah ini mencakup proses pemecahan masalah yang sama pada setiap bagian masalah yang lebih kecil.
3. Combine (Penggabungan): Setelah masalah-masalah kecil tersebut diselesaikan, solusi-solusi mereka digabungkan kembali untuk membentuk solusi dari masalah awal yang lebih besar.

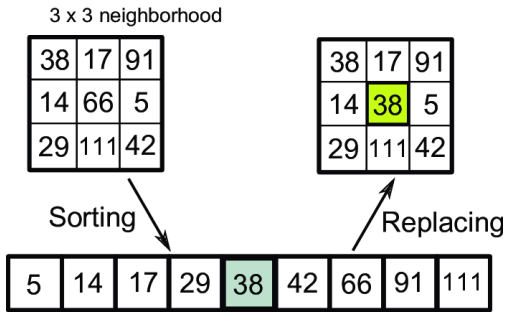
Kelebihan dari algoritma Divide and Conquer adalah kemampuannya dalam menyelesaikan masalah yang kompleks dengan efisiensi yang baik, terutama ketika masalah dapat dibagi menjadi bagian-bagian yang seimbang dan independen. Namun, kelemahannya terletak pada overhead rekursi yang tinggi dan kompleksitas memori tambahan yang dibutuhkan untuk menyimpan hasil dari setiap pemecahan masalah kecil.

B. Denoising

Denoising dalam citra adalah proses menghilangkan atau mengurangi noise dari gambar digital sehingga gambar tersebut menjadi lebih jelas dan berkualitas. Noise merupakan gangguan acak yang ditambahkan ke gambar selama proses akuisisi, transfer, atau penyimpanan data. Noise dapat muncul dalam berbagai bentuk, termasuk titik-titik berwarna yang terpencar-pencar (salt-and-pepper noise), noise Gaussian, atau noise yang terkait dengan kondisi lingkungan saat pengambilan gambar. [3]

Tujuan utama dari denoising adalah untuk menghasilkan gambar yang lebih bersih dan lebih jelas tanpa mengorbankan detail yang penting dalam gambar tersebut. Denoising biasanya dilakukan menggunakan berbagai teknik pemrosesan gambar, yang dapat diterapkan secara spasial maupun frekuensi.

C. Median Filtering



Gambar 1 Median Filtering Ilustration [2]

Median Filtering adalah salah satu teknik pengolahan citra digital yang digunakan untuk mengurangi noise atau gangguan dalam gambar. Teknik ini bertujuan untuk mempertahankan detail struktural dan menghilangkan noise dengan cara mengganti nilai setiap piksel dengan nilai median dari nilai-nilai piksel di sekitarnya.

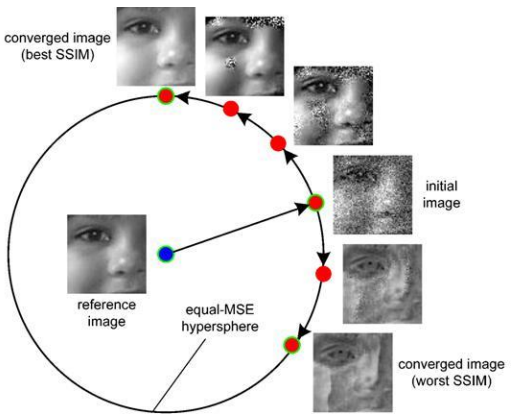
Konsep utama Median Filtering adalah penggunaan nilai median, bukan rata-rata, untuk mengganti nilai piksel. Hal ini karena nilai median lebih tahan terhadap nilai ekstrem atau pencilan daripada rata-rata. Dengan menggunakan nilai median, piksel yang terganggu oleh noise dapat dihilangkan tanpa mengorbankan detail penting dari gambar.

Kelebihan dari Median Filtering adalah kemampuannya untuk mengurangi berbagai jenis noise tanpa merusak detail penting dalam gambar. Namun, pada gambar dengan noise yang sangat tinggi, Median Filtering mungkin tidak memberikan hasil yang optimal, dan teknik denoising lain mungkin lebih cocok.

Median Filtering biasanya lebih cocok digunakan untuk mengatasi jenis noise yang disebut salt and pepper noise. Salt and pepper noise adalah jenis noise yang terdiri dari titik-titik putih (salt) dan titik-titik hitam (pepper) yang tersebar acak dalam gambar. Noise ini muncul sebagai hasil dari kesalahan dalam perekaman atau pemrosesan data, dan dapat mengganggu kualitas gambar dengan cara yang berbeda-beda, tergantung pada intensitasnya.

Algoritma Median Filtering sangat efektif dalam menghilangkan salt and pepper noise karena sifatnya yang mempertahankan detail struktural dalam gambar sambil menghilangkan nilai yang ekstrem atau pencilan. Ketika jendela Median Filtering ditempatkan di sekitar piksel yang terganggu oleh noise salt and pepper, nilai median dari piksel-piksel di sekitarnya akan dihitung. Karena salt and pepper noise biasanya muncul sebagai nilai yang ekstrem dan tidak sesuai dengan konteks lokal, nilai median yang dihitung akan cenderung merepresentasikan nilai yang sebenarnya dari lingkungan sekitarnya.

D. SSIM



Gambar 2 Ilustrasi Proses SSIM [4]

SSIM atau Structural Similarity Index Measure adalah metrik yang digunakan untuk mengukur seberapa baik suatu gambar denoised atau restorasi mempertahankan struktur, detail, dan warna dari gambar asli. Metrik ini membantu dalam mengevaluasi kualitas visual dari gambar hasil

denoising dengan memperhatikan aspek-aspek struktural dari gambar tersebut.

Langkah-langkah umum dalam menghitung SSIM adalah sebagai berikut:

1. Penghitungan Statistik: Gambar asli dan gambar hasil denoising diubah menjadi domain intensitas (biasanya dalam bentuk grayscale). Kemudian, statistik seperti rata-rata (mean), deviasi standar (standard deviation), dan kovarians (covariance) dari kedua gambar dihitung.
2. Perhitungan Kemiripan Struktural: SSIM memperhitungkan kemiripan struktural antara gambar asli dan gambar denoised dengan membandingkan tekstur, kontras, dan struktur lokal dari kedua gambar. Ini dilakukan dengan menghitung indeks kemiripan struktural antara dua blok piksel yang saling berdekatan dalam gambar.
3. Perhitungan Kemiripan Luminansi: SSIM juga memperhitungkan kemiripan dalam luminansi (tingkat kecerahan) antara kedua gambar dengan membandingkan rata-rata dan varian dari intensitas piksel di kedua gambar.
4. Gabungan dari Kemiripan Struktural dan Kemiripan Luminansi: Setelah mendapatkan nilai kemiripan struktural dan kemiripan luminansi, kedua nilai tersebut digabungkan untuk menghasilkan nilai akhir SSIM.

Skala nilai SSIM berkisar antara -1 hingga 1, di mana nilai 1 menunjukkan kesamaan sempurna antara kedua gambar, sedangkan nilai yang lebih rendah menunjukkan adanya distorsi atau perbedaan yang signifikan antara kedua gambar. Semakin tinggi nilai SSIM, semakin mirip gambar denoised dengan gambar asli.

SSIM sangat berguna dalam mengevaluasi kualitas visual dari hasil denoising, karena mempertimbangkan aspek struktural dan luminansi dari gambar. Ini membuat SSIM menjadi metrik yang penting dalam memilih dan membandingkan teknik-teknik denoising yang berbeda.

III. Implementasi dan Pembahasan

A. Median Filtering

Proses pada algoritma *Median Filtering* yang digunakan pada dasarnya terbagi menjadi 4 tahap iterative:

1) Pemilihan Jendela

   Untuk setiap piksel dalam gambar, sebuah jendela (window) berukuran tertentu diposisikan di sekitarnya. Ukuran jendela ini bisa bervariasi tergantung pada aplikasi dan tingkat noise yang ingin dihilangkan. Jendela ini berfungsi sebagai area tempat kita mencari nilai median. Nilai jendela dipilih 1 piksel disekitar.

2) Pengurutan Nilai

   Setelah jendela ditempatkan di sekitar piksel yang sedang diproses, nilai dari semua piksel dalam jendela diurutkan dari yang terkecil hingga yang terbesar. Pengurutan ini dilakukan untuk memudahkan penentuan nilai median.

3) Pengambilan Nilai Median

   Setelah nilai-nilai piksel diurutkan, nilai median dari semua nilai piksel dalam jendela diambil sebagai nilai baru untuk piksel yang sedang diproses. Nilai median dihitung dengan menemukan nilai yang berada di tengah-tengah urutan nilai piksel. Jika jumlah piksel dalam jendela adalah ganjil, maka nilai median adalah nilai piksel yang berada di tengah. Jika jumlah piksel dalam jendela adalah genap, maka nilai median adalah rata-rata dari dua nilai yang berada di tengah.

4) Penggantian Nilai Piksel

   Nilai median yang telah dihitung digunakan untuk menggantikan nilai piksel asli dalam gambar. Piksel yang sedang diproses akan diperbarui dengan nilai median yang telah dihitung.

Proses ini diulangi untuk setiap piksel dalam gambar, sehingga noise dapat dihilangkan secara keseluruhan.

B. Penerapan Algoritma Divide and Conquer

Penggunaan algoritma Divide and Conquer bertujuan untuk meningkatkan efisiensi komputasi dengan membagi masalah besar menjadi submasalah yang lebih kecil, memprosesnya secara terpisah, dan kemudian menggabungkan hasilnya untuk mendapatkan solusi akhir. Proses ini dapat dijelaskan dengan lebih detail sebagai berikut:

1) Pemisahan Gambar (*Divide*)

   Pertama, gambar yang akan diproses dibagi menjadi bagian-bagian yang lebih kecil, gambar dapat dibagi menjadi matriks pikesl terkecil, dan setiap matriks ini kemudian akan diproses secara terpisah.

2) Proses Median Filtering pada Setiap Bagian (*Conquer)*

   Setelah gambar dibagi, algoritma Median Filtering diterapkan pada setiap bagian gambar secara terpisah. Proses Median Filtering dilakukan sesuai dengan langkah-langkah yang telah dijelaskan sebelumnya, di mana nilai median dihitung untuk setiap piksel dalam bagian gambar.

3) Penggabungan Hasil (*Combine)*

   Setelah Median Filtering selesai diterapkan pada setiap bagian gambar, hasil dari setiap bagian digabungkan kembali untuk membentuk gambar hasil akhir dengan mensubtitusikan hasil proses median ke piksel yang bersesuaian.

Pseudocode Algoritma Divide and Conquer

```
function divide(image, y, x, kernel_half,
kernel_size, height, width):
```

```
    if kernel_size equals 1:
            return [image[y, x]]
    neighborhood = []
    for each i in range(-kernel_half,
kernel_half + 1):
    for each j in range(-kernel_half,
kernel_half + 1):
      ny = min(max(y + i, 0), height - 1)
      nx = min(max(x + j, 0), width - 1)
      neighborhood.append(image[ny, nx])
      return neighborhood
function conquer(neighborhood):
     return median value of neighborhood
function combine(result, y, x,
median_value):
      result[y, x] = median_value
function parallel_median_filter(image,
kernel_size):
       height, width = shape of image[:2]
       kernel_half = kernel_size / 2
       result = create an array like image
filled with zeros
function process_pixel(y, x):
        neighborhood = divide(image, y, x,
kernel_half, kernel_size, height, width)
            median_value =
conquer(neighborhood)
            combine(result, y, x,
median_value)
    process_pixel_threads
```

C. *Pengujian*

Pada makalah ini, proses pengujian dilakukan menggunakan gambar-gambar dengan perbesaran yang berbeda secara berturut-turut, yaitu 0.5, 1, dan 2 kali lipat. Semua dataset sumber memiliki resolusi awal 512x512 piksel, sehingga data yang diolah adalah dengan resolusi yang sudah diperbesar sesuai dengan faktor tersebut. Setiap perbesaran mewakili sebuah kasus yang berbeda: kasus pertama untuk gambar dengan resolusi kecil, kasus kedua untuk gambar dengan resolusi normal, kasus ketiga untuk gambar dengan resolusi besar, dan kasus keempat untuk gambar dengan resolusi paling besar.

Dalam pengujian ini, setiap gambar akan diuji menggunakan dua jenis algoritma: algoritma yang bersifat naif dan algoritma divide conquer. Complexitas waktu dari kedua algoritma akan dibandingkan untuk melihat performa masing-masing algoritma pada berbagai resolusi gambar.

Selain itu, pengujian akan melibatkan analisis kesesuaian hasil algoritma dengan fungsi filtering bawaan dari library OpenCV. Pemilihan window untuk analisis ini adalah 1 piksel sekeliling. Selanjutnya, akan dilakukan pengukuran efektivitas hasil algoritma dengan menggunakan Structural Similarity Index (SSIM). SSIM akan dihitung dengan menggunakan eksternal library dari scikit-learn (sklearn) untuk melihat sejauh mana kesesuaian hasil algoritma dengan fungsi filtering bawaan dari OpenCV.

1) *Pengujian Pertama*

Digunakan gambar individu bermain basket dalam ruangan, diambil dari lingkungan indoor, dengan perbesaran 0.5 kali dari ukuran asli, setara dengan resolusi 256x256 piksel.

Gambar 3 Pengujian 1 Orang Bermain Basket [5]
(Sumber: Kaggle.com)

Berikut adalah hasil median filtering menggunakan library bawaan dari OpenCV dan metode Divide conquer.

Gambar 4 Hasil Image Denoising Pengujian 1

```
==================Test Case 1====================
Time for naive: 3.94096302986145
Time for divide and conquer: 1.1604235172271729
SSIM before denoising: 61.13802363802605 %
SSIM with Divide and Conquer: 83.32845456197772 %
SSIM with OpenCV Median Filter: 83.32845456197772 %
=================================================
```

Gambar 5 Data Hasil Pengujian 1

Pengujian menunjukkan perbedaan waktu eksekusi yang signifikan, yaitu 1,16 detik untuk Divide conquer dan 3,94 detik untuk algoritma naif. Dari hasil tersebut, terbukti bahwa Divide conquer mampu memberikan kinerja yang lebih efisien.

Dapat dilihat juga bahwa nilai Structural Similarity Index (SSIM) yang dihasilkan oleh kedua pendekatan,

baik Divide conquer maupun OpenCV, sebanding. Hal ini menunjukkan bahwa algoritma tersebut berhasil dan sama dalam proses denoising, yang tercermin dari peningkatan nilai SSIM yang signifikan, yaitu sekitar 22%.

2) Pengujian Kedua

Digunakan gambar individu bermain basket dalam ruangan, diambil dari lingkungan indoor dengan ukuran asli yaitu resolusi 512x512 piksel.

Gambar 6 Pengujian 2 Wanita [5]
(Sumber: Kaggle.com)

Berikut adalah hasil median filtering menggunakan library bawaan dari OpenCV dan metode Divide conquer.

Gambar 7 Hasil Image Denoising Pengujian 2

```
==================Test Case 2=====================
Time for naive: 17.31080389022827
Time for divide and conquer: 4.666355133056641
SSIM before denoising: 30.13211577289922 %
SSIM with Divide and Conquer: 86.29486513357644 %
SSIM with OpenCV Median Filter: 86.29486513357644 %
```

Gambar 8 Data Hasil Pengujian 2

Hasil pengujian menunjukkan adanya perbedaan waktu eksekusi yang signifikan antara dua pendekatan yang diuji. Waktu eksekusi untuk algoritma Divide conquer tercatat sebesar 4,6 detik, sedangkan algoritma naif memerlukan waktu sebanyak 17,3 detik. Perbedaan waktu yang mencolok ini menunjukkan bahwa algoritma Divide conquer mampu memberikan kinerja yang jauh lebih efisien dibandingkan dengan pendekatan naif.

Selain itu, analisis juga dilakukan terhadap nilai Structural Similarity Index (SSIM) yang dihasilkan oleh kedua pendekatan, yaitu Divide conquer dan OpenCV. Hasilnya menunjukkan bahwa kedua pendekatan ini memiliki nilai SSIM yang sebanding. dan menunjukkan bahwa baik algoritma Divide conquer maupun OpenCV berhasil dalam proses denoising, yang tercermin dari peningkatan nilai SSIM yang signifikan, yakni sekitar 56%.

3) Pengujian Kedua

Digunakan gambar sekumpulan bebek yang berada di outdoor, dengan perbesaran gambar 2 kali lipat, yaitu memiliki resolusi 1024x1024 piksel.

Gambar 9 Pengujian 2 Sekumpulan Bebek [5]
(Sumber: Kaggle.com)

Berikut adalah hasil median filtering menggunakan library bawaan dari OpenCV dan metode Divide conquer.

Gambar 10 Hasil Image Denoising Pengujian 3

```
==================Test Case 3=====================
Time for naive: 73.97346067428589
Time for divide and conquer: 18.68701672554016
SSIM before denoising: 59.81204669729513 %
SSIM with Divide and Conquer: 72.8808825193021 %
SSIM with OpenCV Median Filter: 72.8808825193021 %
==================================================
```

Gambar 11 Data Hasil Pengujian 3

Hasil pengujian menunjukkan perbedaan waktu eksekusi yang cukup signifikan antara dua pendekatan yang diuji. Algoritma Divide conquer mencatat waktu eksekusi sebesar 18 detik, sementara algoritma naif memerlukan waktu sebanyak 73 detik. Perbedaan waktu yang mencolok ini menegaskan bahwa algoritma Divide conquer mampu memberikan kinerja yang jauh lebih efisien dibandingkan dengan pendekatan naif.

Selain itu, dilakukan analisis terhadap nilai Structural Similarity Index (SSIM) yang dihasilkan oleh kedua pendekatan, yakni Divide conquer dan OpenCV. Hasilnya menunjukkan bahwa keduanya memiliki nilai

SSIM yang sebanding, menandakan bahwa baik algoritma Divide conquer maupun OpenCV berhasil dalam proses denoising. Peningkatan nilai SSIM yang signifikan, sekitar 13%, meskipun sudah menurun dari sebelumnya, menunjukkan bahwa perbaikan perlu dilakukan pada pemilihan window untuk scaling yang besar. Hal ini tidak mengindikasikan penurunan kualitas algoritma, tetapi menekankan perlunya evaluasi ulang untuk memastikan pemilihan median dapat menghasilkan kandidat yang lebih baik lagi.

IV. Kesimpulan Dan Saran

Berdasarkan hasil pengujian, dapat disimpulkan bahwa metode Divide conquer menunjukkan keunggulan dalam efisiensi waktu eksekusi dibandingkan dengan pendekatan naif. Hal ini terbukti dari perbedaan waktu eksekusi yang signifikan pada berbagai skenario pengujian. Selain itu, kedua pendekatan berhasil meningkatkan kualitas gambar melalui proses denoising, dengan peningkatan yang cukup besar dalam nilai Structural Similarity Index (SSIM).

Meskipun demikian, terdapat beberapa area yang perlu mendapatkan perhatian lebih lanjut. Evaluasi lebih lanjut diperlukan terhadap pemilihan window untuk scaling yang besar guna memastikan peningkatan kualitas gambar tanpa mengorbankan efisiensi waktu eksekusi yang telah terbukti efisien.

Tentu saja, algoritma yang telah diimplementasikan oleh penulis masih memiliki ruang untuk pengembangan lebih lanjut. Hal ini dapat dilakukan dengan memperbaiki algoritma Divide and Conquer yang ada atau dengan mencoba algoritma yang berbeda untuk mencapai solusi yang lebih optimal.

Link Repositori

My github repository

Video Link at Youtube

Demo

Ucapan Terima kasih

Penulis ingin menyatakan rasa terima kasih kepada Dr. Ir. Rinaldi, M.T. dan semua pengajar IF2211 Strategi Algoritma atas kesempatan yang diberikan kepada penulis dalam pembuatan makalah ini. Tak lupa juga, penulis juga mengungkapkan rasa syukur kepada orang tua, keluarga, dan teman-teman atas dukungan dan doa mereka.

V. References

[1] R. Munir, "https://informatika.stei.itb.ac.id/," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2023-2024/Algoritma-Divide-and-Conquer-(2024)-Bagian1.pdf. [Accessed 12 6 2024].

[2] N. Maturi, "Concept of median filtering in image-processing," 11 2013. [Online]. Available: https://www.researchgate.net/figure/Concept-of-median-filtering-in-image-processing_fig17_281534044. [Accessed 12 6 2024].

[3] "Glematic," [Online]. Available: https://gleematic.com/indonesia/apa-itu-image-denoising-dan-bagaimana-teknologi-ini-meningkatkan-kualitas-pemrosesan-dokumen/. [Accessed 12 6 2024].

[4] Z. Wang, "The SSIM Index for Image Quality Assesment," 11 2 2011. [Online]. Available: https://www.cns.nyu.edu/~lcv/ssim/. [Accessed 12 6 2024].

[5] R. Bansal, "Kaggle," 4 2024. [Online]. Available: https://www.kaggle.com/datasets/rajneesh231/salt-and-pepper-noise-images. [Accessed 12 6 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, 12 Juni 2024

Moh Fairuz Alauddin Yahya 13522057