Segmentasi Citra Digital Menggunakan Otsu Thresholding dengan Optimasi Genetic Algorithm

Muhammad Syahnabil Hammam Sungkar
Fakultas Teknologi Maju dan Multidisplin
Teknik Robotika dan Kecerdasan Buatan
Surabaya, Indonesia
muhammad.syahnabil.hammam-2021@ftmm.unair.ac.id

Saddam Al Fattah Putra Adi
Fakultas Teknologi Maju dan Multidisplin
Teknik Robotika dan Kecerdasan Buatan
Surabaya, Indonesia
saddam.al.fattah-2021@ftmm.unair.ac.id

Lakki Taj Roid
Fakultas Teknologi Maju dan Multidisplin
Teknik Robotika dan Kecerdasan Buatan
Surabaya, Indonesia
lakki.taj.roid-2021@ftmm.unair.ac.id

***Abstract*—Penelitian ini bertujuan untuk mengimplementasikan metode Otsu Thresholding dalam segmentasi citra digital dengan pendekatan optimasi menggunakan algoritma genetika. Metode ini digunakan untuk menentukan nilai ambang intensitas optimal yang memisahkan latar belakang dan objek pada citra grayscale. Hasil pengujian menunjukkan bahwa algoritma genetika mampu menemukan nilai ambang yang mendekati metode Otsu klasik dengan fleksibilitas parameter dan waktu komputasi yang masih dapat diterima.**

Kata kunci— Genetic Algorithm, Otsu Thresholding, Segmentasi Citra, Optimasi, Pengolahan Citra Digital

I. PENDAHULUAN

Segmentasi citra merupakan salah satu tahapan penting dalam pengolahan citra digital yang bertujuan untuk memisahkan objek relevan dari latar belakangnya. Salah satu metode segmentasi yang paling umum digunakan adalah *thresholding*, yaitu proses menentukan nilai ambang (*threshold*) untuk mengklasifikasikan piksel menjadi dua kelas, yakni latar belakang (*background*) dan objek utama (*foreground*). Metode thresholding sederhana namun efektif dalam banyak aplikasi seperti deteksi tepi, pengenalan karakter, dan analisis medis.

Salah satu teknik *thresholding* yang banyak digunakan adalah metode Otsu, yang dikembangkan oleh Nobuyoku Otsu pada tahap Otsu pada tahun 1979 [1]. Metode ini bekerja dengan memilih nilai ambang yang memaksimalkan variansi antar kelas (*inter-class variance*) dan meminimalkan variansi dalam kelas (*intra-class variance*), sehingga menghasilkan segmentasi yang optimal. Namun, metode Otsu secara klasik melakukan pencarian nilai ambang secara *brute-force,* yang dapat menjadi kurang efisien untuk citra beresolusi tinggi atau dalam kasus multi-thresholding.

Untuk mengatasi keterbatasan tersebut, pendekatan berbasis optimasi dapat digunakan. Salah satu metode optimasi yang populer adalah Algoritma Genetika (*Genetic Algorithm*), sebuah teknik heuristik yang meniru proses evolusi biologi untuk mencari solusi optimal. Dalam konteks *thresholding*, algoritma genetika dapat digunakan untuk mencari nilai ambang terbaik berdasarkan fungsi objektif dari metode Otsu [2].

Penelitian ini bertujuan untuk mengimplementasikan metode Otsu *Thresholding* yang dioptimalkan menggunakan algoritma genetika dalam proses segmentasi citra digital. Dengan pendekatan ini, diharapkan diperoleh nilai ambang yang optimal secara efisien serta menghasilkan segmentasi citra yang akurat.

II. TINJAUAN PUSTAKA

A. Library

Langkah pertama dalam mengimplementasi metode Otsu *Thresholding* yaitu mengimport semua *library* (pustaka) Python yang dibutuhkan. Setiap *library* memiliki peran spesifik dalam percobaan kali ini. Berikut merupakan beberapa *library* yang diimpor dalam percobaan ini:

- OpenCV: Digunakan untuk memuat dan memproses gambar, termasuk membaca gambar input dan mengubahnya dari BGR ke format grayscale. OpenCV menyediakan alat yang efisien untuk manipulasi dan *preprocess* gambar [3].
- Numpy: Digunakan untuk operasi numerik, seperti manipulasi array, perhitungan varians, dan penanganan nilai intensitas piksel dalam gambar [4].
- Pandas: Digunakan untuk mengatur dan meringkas hasil eksperimen dalam format terstruktur, memungkinkan analisis statistik kinerja algoritma [5].
- Random: Menyediakan fungsi untuk menghasilkan angka acak, yang penting untuk menginisialisasi populasi dan melakukan operasi persilangan dan mutasi dalam algoritma genetik [6].
- Time: Digunakan untuk mengukur waktu proses algoritma genetik, yang memungkinkan evaluasi efisiensi komputasi pada berbagai pengaturan parameter [7].
- Matplotlib.pyplot: Digunakan untuk memvisualisasikan gambar asli, histogram intensitas piksel, dan gambar tersegmentasi, membantu dalam analisis hasil segmentasi [8].

B. Otsu Thresholding

Otsu Thresholding adalah metode thresholding otomatis yang bekerja dengan prinsip memaksimalkan variansi antar kelas (*between-class variance*) dalam histogram intensitas citra. Dalam konteks segmentasi biner, Otsu berupaya mencari nilai ambang T yang dapat memisahkan citra menjadi dua kelas piksel: latar belakang (*background*) dan objek (*foreground*) [9]. Secara matematis, Otsu mendefinisikan variansi antar kelas sebagai berikut:

$$\sigma_b^2(T) = \omega_0(T) \cdot \omega_1(T) \cdot [\mu_0(T) - \mu_1(T)]^2 \quad (1)$$

dengan:

- $\omega_0(T), \omega_1(T)$: Probabilitas kelas 0 dan 1 berdasarkan *threshold* T,
- $\mu_0(T), \mu_1(T)$: Rata-rata intensitas piksel pada masing-masing kelas.

Link Menuju Project: https://github.com/TajRoid/SKRL_UAS

Metode ini mengasumsikan bahwa histogram citra memiliki dua puncak (bimodal), sehingga pemisahan antara kedua kelas dapat dilakukan secara efektif [2].

Keunggulan utama dari Otsu *Thresholding*, yaitu kemampuannya untuk bekerja tanpa parameter tambahan, bersifat otomatis dan *non-parametrik*, serta efisien dalam implementasi komputasional. Namun demikian, metode ini memiliki keterbatasan, terutama pada citra dengan histogram yang tidak *bimodal* atau citra yang dipengaruhi oleh *noise*, cahaya tidak merata, atau kontras rendah [10].

C. Genetic Algorithm (GA)

Algoritma genetika (*Genetic Algorithm*) adalah salah satu metode dalam kecerdasan buatan yang berfungsi sebagai teknik optimasi berdasarkan prinsip evolusi biologis seperti seleksi alam dan pewarisan genetik [11]. Setiap individu dalam populasi direpresentasikan dalam bentuk kromosom, biasanya dinotasikan sebagai vector:

$$X = [x_1, x_2, \dots, x_n] \quad (2)$$

di mana x_i merepresentasikan gen dan dapat berupa bilangan biner atau kontinu tergantung pada jenis masalah. Evaluasi performa tiap individu dilakukan dengan menggunakan fungsi fitness $f(X)$, yang mementukan seberapa baik Solusi tersebut:

$$f(X) = \text{nilai fitness individu } X \quad (3)$$

Metode ini bekerja dengan membuat populasi awal yang terdiri dari solusi-solusi acak, kemudian solusi tersebut dievaluasi menggunakan fungsi *fitness* untuk menilai kualitasnya [12]. Selanjutnya, solusi dengan nilai *fitness* terbaik akan dipilih untuk melalui proses *crossover* dan *mutasi*, sehingga menghasilkan populasi generasi berikutnya yang lebih baik [13].

III. METODOLOGI

Pada penelitian ini, metode yang digunakan adalah gabungan antara *Otsu Thresholding* dan *Genetic Algorithm* (GA) sebagai pendekatan optimasi yang digunakan untuk segmentasi citra digital. Tujuan dari metode ini adalah menemukan nilai ambang (*threshold*) yang dapat memisahkan objek dari latar belakang.

A. Akuisisi Citra

Tahap pertama dalam penelitian ini adalah akuisisi citra, yaitu proses pengambilan atau pengumpulan data citra digital yang akan digunakan sebagai objek uji. Citra diperoleh dari berbagai sumber, seperti dataset terbuka di internet maupun koleksi pribadi yang relevan dengan topik segmentasi objek. Format citra yang bisa digunakan adalah .jpg, .jpeg, dan .png.

Citra yang dipilih memiliki karakteristik visual yang bervariasi, baik dari segi pencahayaan, kontras, hingga kompleksitas latar belakang, dengan tujuan untuk menguji keandalan metode segmentasi yang diusulkan. Sebelum digunakan dalam tahap pemrosesan, citra diubah ke dalam *grayscale* agar sesuai dengan prinsip dasar *thresholding* Otsu. Konversi ini dilakukan menggunakan fungsi *cv2.cvtColor()* dari pustaka OpenCV.

B. Perhitungan Threshold dengan GA-Otsu

Proses pencarian nilai ambang optimal dalam penelitian ini menggunakan kombinasi antara metode Otsu dan pendekatan optimasi berbasis algoritma genetika (*genetic algorithm* / GA). Proses dimulai dengan konversi citra *grayscale* menjadi citra biner. Setelah citra dilakukan binerisasi, dilakukan penghitungan nilai variansi yang bertujuan untuk dua kelompok piksel, yaitu piksel latar depan (*foreground*) dan latar belakang (*background*). Adapan rumusnya adalah:

$$\sigma_B^2(k) = \omega_0(k)\,\sigma_0^2(k) + \omega_1(k)\,\sigma_1^2(k) \quad (2)$$

dengan:

- $\omega_0(k)$ dan $\omega_1(k)$: Proporsi jumlah piksel pada masing-masing kelas,
- $\sigma_0^2(k)$ dan $\sigma_1^2(k)$: Variansi intensitas pada masing-masing kelas.

Untuk mengadaptasi nilai tersebut dalam kerangka algoritma genetika dibuat sebuah fungsi yang digunakan untuk menghitung nilai *fitness* dari setiap individu (*threshold*) dengan formula:

$$f(th) = \frac{1}{\sigma_0^2(th) + \varepsilon} \quad (3)$$

dengan ε adalah bilangan kecil untuk menghindari pembagian dengan nol.

Setelah dilakukan seluruh penghitungan, dilakukan pembuatan fungsi pada *code* untuk mengintegerasi sekumpulan *threshold* dengan acak sebagai populasi awal. Di setiap generasi, fitness tiap individu dihitung, lalu dilakukan seleksi berbasis probabilitas proporsional terhadap nilai fitness. Proses *crossover* dilakukan menggunakan *one-point crossover* pada representasi biner 8-bit dari *threshold*, sedangkan mutasi dilakukan dengan membalik salah satu bit secara acak dengan peluang tertentu. Setelah iterasi selama sejumlah generasi, individu dengan nilai *fitness* tertinggi dipilih sebagai solusi, yaitu nilai *threshold* yang kemudian digunakan untuk melakukan segmentasi citra. Fungsi ini juga mencatat waktu proses untuk evaluasi performa algoritma.

C. Segmentasi Citra

Proses ini bertujuan untuk memisahkan objek utama dari latar belakang dalam citra berdasarkan *threshold* yang telah diproses sebelumnya. Citra yang telah dikonversi ke *grayscale* kemudian dibagi menjadi dua piksel kelas menggunakan teknik *thresholding biner*. Piksel dengan nilai intensitas lebih besar atau sama dengan nilai ambang akan diklasifikasikan sebagai objek (f*oreground*), sedangkan piksel dengan intensitas di bawah nilai ambang akan diklasifikasikan sebagai latar belakang (*background*). Secara matematis, segmentasi dapat dirumuskan sebagai berikut:

$$S(x,y) = \begin{cases} 1, \text{jika } I(x,y) \geq T \\ 0, \text{jika } I(x,y) < T \end{cases} \quad (4)$$

dengan:

- $S(x,y)$: Hasil segmentasi pada piksel koordinat (x,y)
- $I(x,y)$: Nilai intensitas pada piksel (x,y)
- T : Nilai Ambang hasil optimasi dengan GA-Otsu

D. *Analisis Parameter GA*

Untuk mengevaluasi kinerja metode GA-Otsu dalam proses segmentasi citra, dilakukan analisis terhadap beberapa parameter penting dalam algoritma genetika. Parameter-parameter ini berpengaruh langsung terhadap kemampuan algoritma dalam menemukan nilai ambang optimal dan juga terhadap efisiensi komputasi. Ada dua parameter yang divariasikan, yaitu:

1. Laju Mutasi
   Menentukan seberapa besar peluang terjadinya perubahan acak (bit-flip) pada individu baru. Mutasi bertujuan untuk menjaga keragaman populasi agar algoritma tidak terjebak pada solusi lokal.
2. Jumlah Generasi
   Menentukan berapa kali proses evolusi dilakukan. Semakin banyak generasi yang dijalankan, semakin besar peluang untuk menemukan solusi optimal, meskipun dengan risiko waktu komputasi yang lebih lama.

Setiap kombinasi parameter diuji pada citra yang sama untuk menjaga konsistensi uji. Untuk setiap variasi, dilakukan beberapa kali uji ulang untuk memperoleh rata-rata (*mean*) dan standar deviasi dari nilai *fitness* akhir. Indikator yang diamati dalam penelitian ini adalah nilai *fitness* tertinggi, waktu komputasi, dan satabilitas hasil.

E. *Visualisasi Hasil*

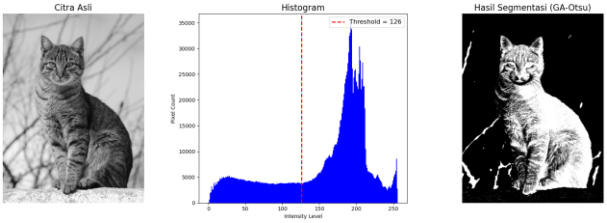
Visualisasi bertujuan untuk membandingkan secara langsung antara citra asli, informasi intensitas piksel dalam bentuk histogram, dan hasil segmentasi akhir, sehingga efektivitas metode yang digunakan dapat diamati secara kualitatif.

IV. ANALISIS DAN PEMBAHASAN

Setelah seluruh proses segmentasi citra dilakukan menggunakan metode GA-Otsu, dilakukan analisis dan pembahasan terhadap hasil yang diperoleh, baik dari sisi visual, numerik, maupun terhadap pengaruh parameter-parameter dalam algoritma genetika.

A. *Visual Segmentasi*

Metode GA-Otsu mampu menghasilkan segmentasi yang cukup baik dalam memisahkan objek utama dari latar belakang. Ini menunjukkan bahwa threshold yang dihasilkan dari optimasi dengan GA dapat mengatasi tantangan pada distribusi histogram citra yang kompleks. Perbandingan dengan histogram intensitas juga menunjukkan bahwa nilai ambang yang diperoleh berada di area pemisah antara dua puncak histogram (*bimodal*), sebagaimana yang menjadi prinsip dasar metode Otsu.



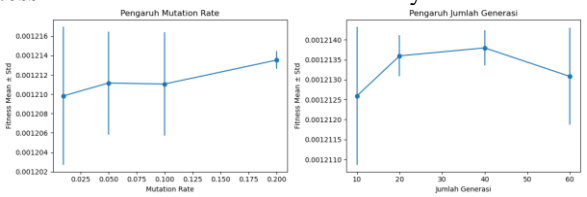
Gambar 1. Hasil Segmentasi Citra oleh GA-Otsu

B. *Evaluasi Nilai Fitness dan Waktu Komputasi*

Nilai *fitness* tertinggi dari *threshold* yang diperoleh menunjukkan tingkat optimalitas dari pemisahan dua kelas piksel. Semakin tinggi nilai fitness, semakin besar jarak antara rata-rata intensitas piksel dari dua kelas (*foreground* dan *background*), yang berarti segmentasi semakin baik. Di sisi lain, waktu komputasi yang dibutuhkan untuk mencapai solusi optimal juga diamati sebagai indikator efisiensi. Secara umum, terlihat bahwa parameter seperti laju mutasi dan jumlah generasi berpengaruh terhadap hasil akhir.

C. *Pengaruh Parameter GA*

Parameter algoritma genetika memiliki pengaruh terhadap performa segmentasi citra yang diukur berdasarkan nilai *fitness*. Pada Gambar 2, ditampilkan grafik pengaruh variasi *mutation rate* dan jumlah generasi terhadap nilai *fitness* rata-rata beserta standar deviasinya



Gambar 2. Pengaruh Parameter GA

a) Pengaruh Mutation Rate
Pada grafik sebelah kiri, terlihat bahwa variasi *mutation rate* antara 0.01, 0.05, 0.1, dan 0.2 menghasilkan nilai *fitness* yang relatif stabil dengan perbedaan yang sangat kecil. Namun demikian, *mutation rate* 0.2 memberikan nilai *fitness* tertinggi dibandingkan yang lain. Hal ini menunjukkan bahwa mutasi yang lebih tinggi memberikan keragaman solusi yang lebih luas dalam proses evolusi, sehingga meningkatkan peluang mendapatkan ambang yang lebih optimal.

Akan tetapi, *mutation rate* 0.1 menunjukkan standar deviasi (*error bar*) tertinggi, yang berarti hasil segmentasinya kurang stabil antar percobaan. Sementara mutation rate 0.05 dan 0.2 relatif lebih stabil, yang menunjukkan bahwa pengaturan *mutation rate* memerlukan keseimbangan antara eksplorasi (keragaman solusi) dan konsistensi hasil.

b) Pengaruh Jumlah Generasi
Pada grafik sebelah kanan, variasi jumlah generasi antara 10, 20, 40, dan 60 memperlihatkan tren peningkatan nilai *fitness* seiring bertambahnya generasi, dengan nilai tertinggi diperoleh pada generasi ke-40. Hal ini menunjukkan bahwa semakin panjang proses evolusi, semakin optimal solusi *threshold* yang ditemukan.

Namun, saat jumlah generasi ditambah hingga 60, terjadi sedikit penurunan pada nilai *fitness*, meskipun tidak signifikan. Standar deviasi juga terlihat menurun hingga generasi ke-40, namun sedikit meningkat pada generasi ke-60. Ini memperkuat indikasi bahwa peningkatan generasi tidak selalu menjamin peningkatan kualitas segmentasi.

V. KESIMPULAN DAN SARAN

Penelitian ini menunjukkan bahwa metode GA-Otsu adalah metode gabungan yang bisa dilakukan untuk segmentasi citra, dengan parameter algoritma genetika yang berpengaruh signifikan terhadap hasil. *Mutation rate* 0.2

menghasilkan nilai *fitness* terbaik, sedangkan jumlah generasi 40 memberikan keseimbangan antara akurasi dan efisiensi waktu komputasi. Visualisasi hasil membuktikan bahwa metode ini mampu memisahkan objek dari latar belakang dengan baik. Untuk pengembangan selanjutnya, disarankan penggunaan adaptasi parameter secara otomatis, eksplorasi pendekatan *multi-threshold*, serta uji coba pada beragam jenis citra guna meningkatkan generalisasi metode.

REFERENCES

[1] N. Ahmed and K. R. Rao, "A_Threshold_Selection_Method_from_Gray-Level_Histograms," 1979.

[2] Vignesh Gopalakrishnan, "Image segmentation using otsu threshold selection method," 2023. [Online]. Available: https://medium.com/@vignesh.g1609/image-segmentation-using-otsu-threshold-selection-method-856ccdacf22

[3] B. Thorne and R. Grasset, "Python for Prototyping Computer Vision Applications."

[4] M. Bauer and M. Garland, "Legate NumPy," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA: ACM, Nov. 2019, pp. 1–23. doi: 10.1145/3295500.3356175.

[5] M. Vagizov, A. Potapov, K. Konzhgoladze, S. Stepanov, and I. Martyn, "Prepare and analyze taxation data using the Python Pandas library," *IOP Conf Ser Earth Environ Sci*, vol. 876, no. 1, p. 012078, Oct. 2021, doi: 10.1088/1755-1315/876/1/012078.

[6] A. Novikau, "Analysis of Pseudorandom Number Generator in Python," *International Journal of Science and Engineering Applications*, Nov. 2024, doi: 10.7753/IJSEA1312.1001.

[7] P. O. Lucas, O. Orang, P. C. L. Silva, E. M. A. M. Mendes, and F. G. Guimarães, "A Tutorial on Fuzzy Time Series Forecasting Models: Recent Advances and Challenges," *Learning and Nonlinear Models*, vol. 19, no. 2, pp. 29–50, Dec. 2022, doi: 10.21528/lnlm-vol19-no2-art3.

[8] S. Cao, Y. Zeng, S. Yang, and S. Cao, "Research on Python Data Visualization Technology," *J Phys Conf Ser*, vol. 1757, no. 1, p. 012122, Jan. 2021, doi: 10.1088/1742-6596/1757/1/012122.

[9] W. K. P. Barros, L. A. Dias, and M. A. C. Fernandes, "Fully Parallel Implementation of Otsu Automatic Image Thresholding Algorithm on FPGA," *Sensors*, vol. 21, no. 12, p. 4151, Jun. 2021, doi: 10.3390/s21124151.

[10] D. T. Anggraeni, "Perbaikan Citra Dokumen Hasil Pindai Menggunakan Metode Simple, Adaptive-Gaussian, dan Otsu Binarization Thresholding," *EXPERT: Jurnal Manajemen Sistem Informasi dan Teknologi*, vol. 11, no. 2, p. 71, Dec. 2021, doi: 10.36448/expert.v11i2.2170.

[11] E. S. Eriana, S. Kom, M. Kom, and D. A. Zein, "ARTIFICIAL INTELLIGENCE (AI) PENERBIT CV. EUREKA MEDIA AKSARA."

[12] A. M. , I. K. M. O. Poltak Rikinaldo T.M, "Aplikasi Warta Jemaat GPDI El-Shaddai Makassar Berbasis WebService Menggunakan Algoritma Genetika," 2022.

[13] A. Fitriyani, H. Lubis, A. A. Hendharsetiawan, and J. A. L. Hutahaean, "SISTEM PENJADWALAN MATA PELAJARAN MENGGUNAKAN ALGORITMA GENETIKA SMK PGRI RAWALUMBU," *Jurnal Manajamen Informatika Jayakarta*, vol. 5, no. 2, p. 205, Apr. 2025, doi: 10.52362/jmijayakarta.v5i2.1881.