**TUGAS KELOMPOK**

**[RKS309] SISTEM KONTROL ROBOTIKA LANJUT**

**Topik:** Penerapan Algoritma Genetika untuk Optimalisasi *Otsu Thresholding* pada Citra

**Kelas:** RK – A1



**KELOMPOK 7**

1. Holina Natalia ( 162012233005 )
2. Alexander Sebastian Axel Wahyudi ( 162012233028 )
3. Wisnu Imam Satrio ( 162112233068 )
4. Maharani Yassar Dewanti ( 162112233079 )

**PROGRAM STUDI TEKNIK ROBOTIKA DAN KECERDASAN BUATAN**

**FAKULTAS TEKNOLOGI MAJU DAN MULTIDISIPLIN**

**UNIVERSITAS AIRLANGGA**

**2024**

# Pendahuluan

*Thresholding* adalah teknik fundamental dalam pengolahan citra digital yang digunakan untuk segmentasi gambar, yaitu memisahkan objek dari latar belakang berdasarkan tingkat intensitas piksel [1]. *Otsu Thresholding* adalah metode populer yang menentukan nilai ambang optimal dengan meminimalkan variansi dalam kelas dari histogram gambar, sehingga menghasilkan segmentasi yang lebih akurat tanpa memerlukan parameter tambahan [1]. Teknik ini sangat penting dalam aplikasi seperti pengenalan objek, deteksi tepi, dan analisis medis karena membantu mengurangi kompleksitas data dan meningkatkan kualitas gambar dengan memisahkan fitur penting dari *noise* [2, 3].

Algoritma Genetika (GA) adalah metode optimasi berbasis evolusi yang menggunakan prinsip seleksi alam untuk menemukan solusi optimal [4]. Dalam konteks *Otsu Thresholding*, GA dapat digunakan untuk menemukan ambang yang meminimalkan variansi dalam kelas lebih efisien dibandingkan metode iteratif tradisional [5]. Dengan menggunakan operasi seleksi, crossover, dan mutasi, GA mengembangkan populasi ambang kandidat untuk mencapai solusi terbaik. Penerapan GA dalam *Otsu Thresholding* bertujuan untuk meningkatkan efisiensi dan akurasi proses segmentasi gambar, menjadikannya alat yang efektif dalam berbagai aplikasi pengolahan citra digital [5].

# Metodologi

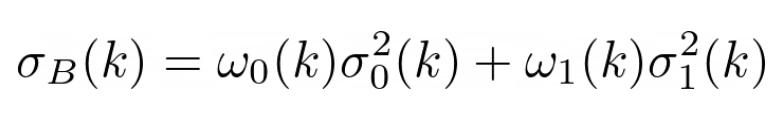
**2.1 Algoritma Genetika**

Algoritma Genetika (*Genetic Algorithm*) adalah metode optimasi berbasis populasi yang terinspirasi oleh proses evolusi biologis. Algoritma ini menggunakan mekanisme seleksi alamiah, *crossover* (pertukaran informasi genetik), dan mutasi untuk menemukan solusi optimal [4]. Pada dasarnya, proses optimasi dimulai dengan menginisialisasi populasi awal yang terdiri dari sekumpulan kandidat solusi (individu). Setiap individu diukur menggunakan fungsi objektif untuk menentukan kualitasnya. Kemudian, individu-individu terbaik dipilih untuk bereproduksi, menghasilkan generasi baru melalui proses *crossover* dan mutasi. Proses ini berulang hingga tercapai kondisi penghentian, seperti jumlah generasi yang telah ditentukan atau konvergensi pada solusi optimal. Algoritma Genetika cocok digunakan dalam masalah *thresholding* citra karena mampu mencari nilai *threshold* optimal secara efisien meskipun ruang pencarian besar dan kompleks [5].

**2.2 *Otsu Thresholding***

*Otsu Thresholding* adalah metode segmentasi citra yang digunakan untuk membagi citra menjadi dua kelas, yaitu latar belakang (*background*) dan objek (*foreground*), berdasarkan nilai intensitas piksel [5]. Metode ini mencari nilai ambang (*threshold*) yang meminimalkan varian intra-kelas dan memaksimalkan varian antar-kelas, yang secara efektif memisahkan dua kelas tersebut [6]. Prosesnya melibatkan perhitungan histogram citra untuk menentukan distribusi intensitas piksel dan kemudian menghitung antara kelas varians pada setiap kemungkinan nilai *threshold* [6]. Nilai *threshold* optimal adalah nilai yang memaksimalkan antara kelas varians. Metode ini sangat efektif untuk citra dengan bimodal histogram, di mana intensitas piksel dari latar belakang dan objek memiliki distribusi yang terpisah [5].

**2.3 Fungsi Objektif dari *Otsu Thresholding***



# Implementasi

**3.1 Alur Program dan Flowchart**

Algoritma *Otsu Thresholding* bagaikan seorang peneliti yang ingin memisahkan dua kelompok benda dalam sebuah kotak berdasarkan ukurannya. Ia menggunakan histogram, bagaikan grafik yang menunjukkan jumlah benda berdasarkan ukurannya, untuk menganalisis distribusi benda di dalam kotak. Tujuan *Otsu* adalah menemukan garis pemisah yang tepat di histogram, bagaikan garis batas di taman bermain yang memisahkan area anak-anak dan area orang dewasa. Garis ini harus meminimalkan variasi ukuran benda di setiap kelompok, bagaikan memastikan anak-anak bermain di area yang sesuai dengan ukuran mereka dan orang dewasa di area yang sesuai dengan ukuran mereka. Dengan meminimalkan variasi ini, *Otsu* dapat memastikan bahwa benda-benda di setiap kelompok memiliki ukuran yang serupa, bagaikan memastikan anak-anak di area anak-anak memiliki ukuran yang tidak jauh berbeda dan orang dewasa di area orang dewasa memiliki ukuran yang tidak jauh berbeda. Setelah menemukan garis pemisah, *Otsu* kemudian mengelompokkan benda berdasarkan ukurannya. Benda yang ukurannya di bawah garis pemisah dikategorikan sebagai kelompok kecil, bagaikan anak-anak yang bermain di area anak-anak, dan benda yang ukurannya di atas garis pemisah dikategorikan sebagai kelompok besar, bagaikan orang dewasa yang bermain di area orang dewasa. Proses ini memungkinkan peneliti untuk memisahkan dan mengelompokkan benda-benda di dalam kotak dengan cara yang efisien dan akurat, bagaikan memisahkan anak-anak dan orang dewasa di taman bermain dengan cara yang aman dan terorganisir.

Citra digital tersusun atas kumpulan piksel, di mana setiap piksel memiliki nilai intensitas yang berkisar antara 0 hingga 255. Dalam sebuah citra, dimungkinkan beberapa piksel memiliki nilai intensitas yang sama [5]. Contohnya, pada citra *grayscale*, beberapa piksel mungkin memiliki nilai intensitas 255. Peluang kemunculan nilai intensitas tertentu dalam citra dapat dihitung dengan membagi jumlah piksel yang memiliki intensitas tersebut dengan total jumlah piksel dalam citra. Dengan menghitung peluang kemunculan nilai intensitas yang berbeda, kita dapat menganalisis distribusi intensitas piksel dalam citra dan memperoleh wawasan tentang karakteristik citra tersebut [6]. Secara matematis, peluang kemunculan piksel dengan nilai intensitas i dalam sebuah citra dapat dihitung dengan rumus berikut [1]:

di mana:

: Peluang kemunculan piksel dengan nilai intensitas i.

: Jumlah piksel dalam citra yang memiliki nilai intensitas i.

: Total jumlah piksel dalam citra.

Rumus ini menghitung peluang dengan membagi jumlah piksel dengan intensitas spesifik (Ni) dengan total jumlah piksel (N), menghasilkan nilai antara 0 dan 1 [1]. Semakin dekat nilai ini dengan 1, semakin tinggi peluang kemunculan intensitas spesifik tersebut dalam citra. Contohnya, jika sebuah citra memiliki 10.000 piksel dan 200 di antaranya memiliki nilai intensitas 128, maka peluang kemunculan piksel dengan intensitas 128 dalam citra tersebut adalah:

P(128) = 200 piksel / 10.000 piksel = 0,02

Artinya, terdapat peluang 2% untuk menemukan piksel dengan intensitas 128 dalam citra tersebut. *Otsu Thresholding* merupakan algoritma yang dirancang untuk memisahkan objek dari latar belakang dalam gambar *grayscale* (hitam putih). Algoritma ini bekerja dengan mencari nilai ambang batas (*threshold*) optimal yang meminimalkan varians antar kelas (varians antara objek dan latar belakang) dan memaksimalkan varians intra-kelas (varians dalam objek dan latar belakang) [1]. Langkah-langkah *Otsu Thresholding* [1]:

1. Membuat Histogram

Pertama, *Otsu Thresholding* membuat histogram gambar. Histogram ini menunjukkan frekuensi kemunculan setiap nilai intensitas (tingkat kecerahan) piksel dalam gambar. Informasi ini penting untuk memahami distribusi intensitas piksel di seluruh gambar.

1. Iterasi Melalui Nilai *Threshold*

Algoritma ini kemudian mengiterasi melalui semua nilai ambang batas (k) yang mungkin. Pada setiap iterasi, nilai k digunakan untuk membagi piksel dalam gambar menjadi dua kelas:

* 1. Kelas 0 (Latar Belakang)

Piksel dengan intensitas ≤ k. Piksel-piksel ini dianggap sebagai bagian dari latar belakang gambar.

* 1. Kelas 1 (Objek)

Piksel dengan intensitas > k. Piksel-piksel ini dianggap sebagai bagian dari objek yang ingin dipisahkan dari latar belakang.

1. Perhitungan Varian

Setelah membagi piksel, *Otsu Thresholding* menghitung dua jenis varians:

* 1. Varians Intra-kelas ( dan )

Varians ini mengukur seberapa tersebar nilai intensitas piksel di dalam setiap kelas (latar belakang dan objek). Semakin kecil nilai varians ini, semakin homogen intensitas piksel di dalam kelasnya.

* 1. Varians Antar-kelas ()

Varians ini mengukur seberapa berbeda nilai intensitas piksel antar kelas (latar belakang dan objek). Semakin besar nilai varians ini, semakin berbeda karakteristik intensitas antara objek dan latar belakang, sehingga memudahkan pemisahannya.

1. Penentuan Nilai (k) Optimal

*Otsu Thresholding* memilih nilai ambang batas (k) yang menghasilkan varians antar-kelas () terbesar. Nilai k ini dianggap sebagai ambang batas optimal karena memaksimalkan perbedaan intensitas antara objek dan latar belakang, sehingga menghasilkan segmentasi gambar yang lebih baik.

1. Rumus Perhitungan (Fungsi Objektif)

Rumus-rumus yang digunakan dalam *Otsu Thresholding* untuk menghitung probabilitas kelas, rata-rata kelas, dan varians dijelaskan dalam jawaban sebelumnya. Di sini, kita akan fokus pada rumus untuk menghitung varians antar-kelas ():

dimana:

: Varians antar-kelas

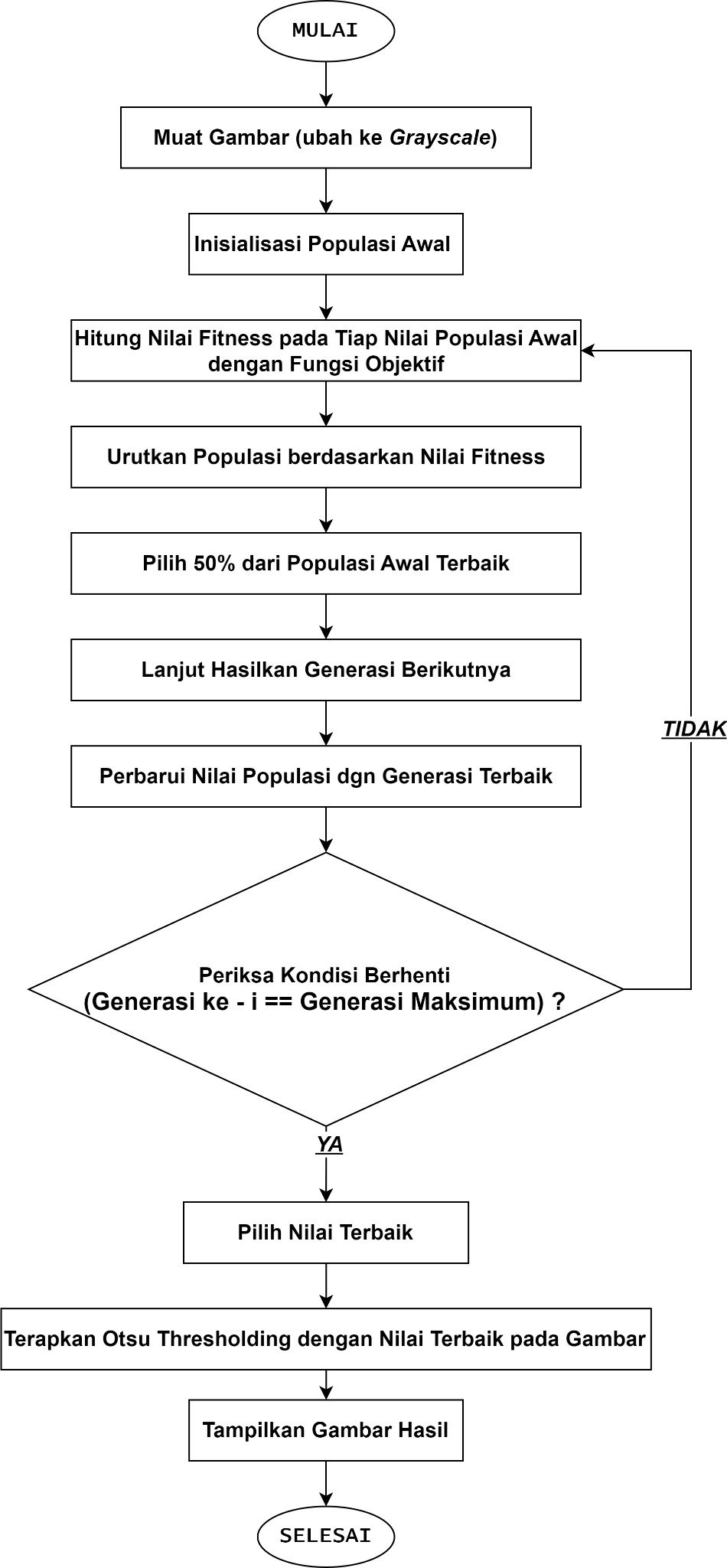
: Rata-rata kelas latar belakang (kelas 0)

: Rata-rata kelas objek (kelas 1)

: Probabilitas kelas latar belakang (kelas 0)

: Probabilitas kelas objek (kelas 1)

Rumus ini menunjukkan bahwa varians antar-kelas bergantung pada perbedaan rata-rata intensitas antara kedua kelas dan probabilitas masing-masing kelas ( dan ). Nilai yang tinggi menunjukkan bahwa terdapat perbedaan intensitas yang signifikan antara objek dan latar belakang, sehingga memungkinkan pemisahan yang lebih baik.



Pada *flowchart* di atas, langkah-langkah detail yang telah diterapkan pada *Otsu Thresholding* dengan Algoritma Genetika adalah sebagai berikut:

1. Mulai

Inisialisasi semua parameter, seperti jumlah populasi, jumlah generasi, tingkat crossover, dan tingkat mutasi.

2. Muat Gambar

Baca gambar dari *path* yang sudah ditentukan, dan lakukan konversi gambar ke skala abu-abu (*grayscale*) karena hal ini penting dalam *Otsu Thresholding*.

3. Inisialisasi Populasi

Buat populasi awal dengan sejumlah kromosom (individu) yang setara dengan jumlah populasi. Setiap kromosom merepresentasikan nilai *threshold* (ambang batas) (k) yang acak dalam rentang yang valid.

4. Hitung *Fitness* untuk Setiap (k)

Untuk setiap kromosom (*threshold*) dalam populasi:

1. Terapkan nilai k pada gambar *grayscale*.
2. Hitung varians intra-kelas ( dan ) untuk kedua kelas (latar belakang dan objek).
3. Hitung varians antar-kelas () menggunakan rumus *Otsu Thresholding*.
4. Gunakan nilai sebagai nilai *fitness* (kesesuaian) kromosom.

5. Urutkan Populasi Berdasarkan *Fitness*

Urutkan populasi berdasarkan nilai *fitness* () dari yang terkecil (terbaik) hingga terbesar (terburuk).

6. Pilih Setengah Populasi Terbaik

Pilih 50% teratas dari populasi (kromosom dengan nilai *fitness* terbaik) untuk melanjutkan ke generasi berikutnya, sehingga setengah populasi lainnya akan dibuang karena dianggap mati dan tidak menghasilkan keturunan yang lebih baik.

7. Hasilkan Keturunan

Menggunakan operasi *crossover* dan mutasi untuk menghasilkan keturunan baru.

* *Crossover*: Pilih dua kromosom dari populasi terbaik secara acak. Gabungkan bagian-bagian kromosom tersebut untuk menghasilkan dua kromosom baru (keturunan).
* Mutasi: Ubah nilai bit acak pada beberapa kromosom dengan probabilitas yang ditentukan (tingkat mutasi).

8. Perbarui Populasi

Menggantikan populasi lama dengan populasi baru yang terdiri dari kromosom terbaik dari generasi sebelumnya dan keturunan yang baru dihasilkan.

9. Periksa Kondisi Penghentian

Jika jumlah generasi maksimum tercapai, lanjutkan ke langkah berikutnya. Jika tidak, kembali ke langkah 4 untuk memulai generasi berikutnya.

10. Pilih Ambang Terbaik

Memilih kromosom dengan nilai *fitness* terbaik () dari populasi akhir. Ambang batas (k) yang direpresentasikan oleh kromosom ini adalah ambang batas optimal untuk *Otsu Thresholding*.

11. Ambang Gambar

Menerapkan nilai *threshold* k optimal pada gambar *grayscale*.

12. Tampilkan Gambar Terambang

Menampilkan gambar asli dan gambar yang telah diambang batas dengan k optimal.

13. Selesai

Algoritma selesai dan menghasilkan k optimal untuk *Otsu Thresholding*.

**Keterangan:**

* Populasi

Populasi dalam algoritma genetika terdiri dari individu-individu (kromosom) yang mewakili solusi potensial untuk masalah (dalam hal ini, nilai ambang batas untuk *Otsu Thresholding*) [4].

* *Fitness*

*Fitness* adalah nilai yang mengukur seberapa baik setiap individu (kromosom) dalam menyelesaikan masalah [4]. Dalam *Otsu Thresholding*, *fitness* dihitung dengan menggunakan varians antar kelas () [6]. Nilai yang lebih kecil menunjukkan segmentasi gambar yang lebih baik, sehingga individu dengan nilai yang lebih kecil memiliki *fitness* yang lebih tinggi [6].

* *Crossover*

*Crossover* adalah operasi yang menggabungkan bagian-bagian dari dua individu (kromosom). Crossover dilakukan untuk mengeksplorasi ruang solusi dan menghasilkan keturunan dengan karakteristik yang menggabungkan fitur-fitur dari kedua induk. Metode crossover yang sering digunakan termasuk [4]:

1. *Single-point crossover*

Memilih satu titik dalam kromosom dan menukar segmen-segmen di kedua sisi titik tersebut antara dua individu.

1. *Two-point crossover*

Memilih dua titik dalam kromosom dan menukar segmen-segmen di antara titik-titik tersebut.

1. *Uniform crossover*

Menukar bit atau segmen individu berdasarkan probabilitas yang ditentukan.

* Mutasi

Mutasi adalah operasi yang mengubah satu atau lebih gen dalam kromosom untuk menghasilkan variasi dalam populasi [4]. Mutasi dilakukan untuk menghindari konvergensi prematur ke solusi lokal dan untuk memastikan eksplorasi ruang solusi yang lebih luas. Mutasi dapat berupa perubahan nilai bit tunggal dalam kromosom biner atau sedikit perubahan nilai dalam kromosom *real*.

**3.2 Kode Program**

Dalam tugas ini, kami menggunakan total 4 buah program yang menjalankan algoritma genetika dalam implementasinya pada *Otsu thresholding*. Seluruh 4 program yang kami buat, memiliki satu hal yang sama yaitu terkait implementasi fungsi objektifnya.

def compute\_otsu\_criteria(im:np.ndarray,th):

thresholded\_im = threshold\_image(im,th)

nb\_pixels = im.size

nb\_pixels1 = np.count\_nonzero(thresholded\_im)

weight1 = nb\_pixels1/nb\_pixels

weight0 = 1-weight1

if weight1 == 0 or weight0 == 0:

return np.inf

val\_pixels1 = im[thresholded\_im==1]

val\_pixels0 = im[thresholded\_im==0]

var0 = np.var(val\_pixels0) if len(val\_pixels0) > 0 else 0

var1 = np.var(val\_pixels1) if len(val\_pixels1) > 0 else 0

return weight0\*var0 + weight1\*var1

Fungsi ***threshold\_image*** bertujuan untuk melakukan segmentasi citra menjadi dua kelas berdasarkan ambang batas (th). Hasilnya adalah citra biner di mana piksel yang lebih besar dari ambang batas akan disetel ke 1 dan sisanya ke 0. Fungsi ***compute\_otsu\_criteria*** bertujuan untuk menghitung kriteria *Otsu*, yang merupakan kombinasi dari dalam-varians dalam dua kelas yang dihasilkan setelah citra dipisahkan oleh ambang batas tertentu dengan parameter yaitu ***im*** = citra input dalam bentuk array numpy (2D array untuk citra *grayscale*) dan ***th*** = Nilai ambang batas yang digunakan untuk memisahkan citra menjadi dua kelas.

Langkah-langkah dalam fungsi ***compute\_otsu\_criteria*** yaitu :

1. Penghitungan Citra Biner

Memanggil fungsi ***threshold\_image*** untuk menghasilkan citra biner dengan menggunakan ambang batas yang diberikan.

1. Perhitungan Jumlah Piksel

Menghitung jumlah piksel total dalam citra input (***nb\_pixels***).

1. Perhitungan Bobot Kelas

Menghitung jumlah piksel yang termasuk ke dalam kelas 1 (nilai piksel yang lebih besar dari ambang batas) dan menghitung bobot kelas 1 (***weight1***) dan bobot kelas 0 (***weight0***).

1. Penanganan Kasus Khusus

Mengembalikan nilai tak hingga (***np.inf***) jika salah satu dari bobot kelas adalah 0, yang menunjukkan kasus di mana salah satu kelas tidak memiliki piksel.

1. Perhitungan Varians

Menghitung varians piksel untuk setiap kelas dimana varians dihitung menggunakan nilai piksel dari kelas 0 dan kelas 1.

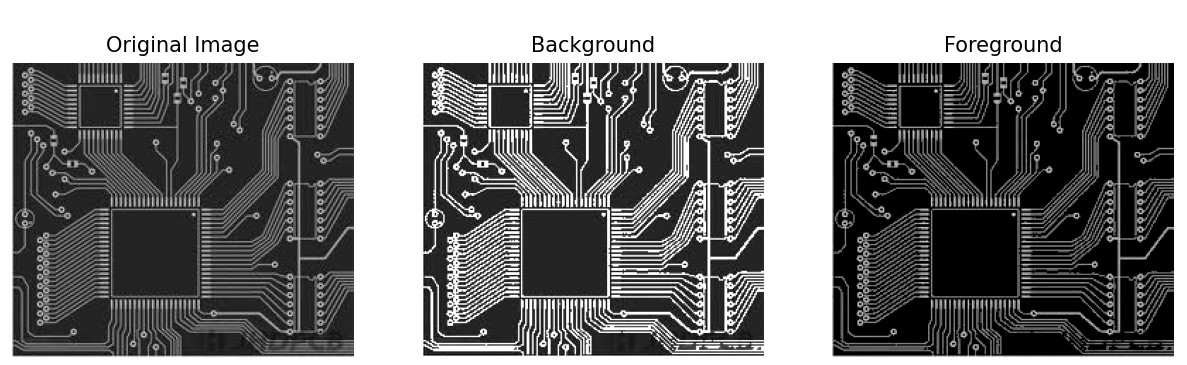
1. Perhitungan Kriteria *Otsu*

Mengembalikan kriteria *Otsu*, yaitu kombinasi dari dalam-varians dalam dua kelas yang telah dihitung sebelumnya.

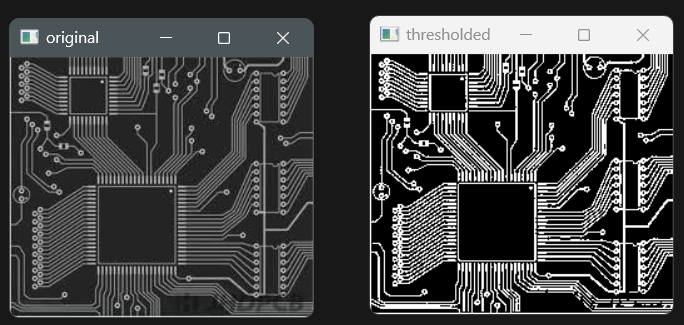
# Hasil dan Pembahasan

**4.1 Hasil Keseluruhan**

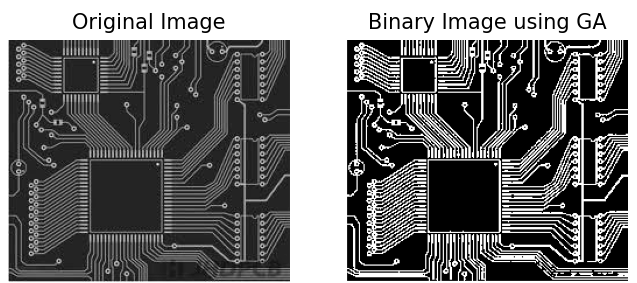
1. k = 84 (Wisnu)



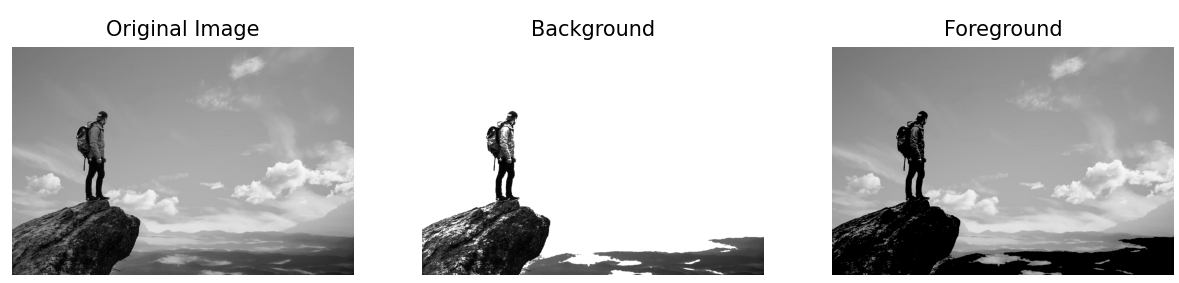
1. k = 84 (Holin)



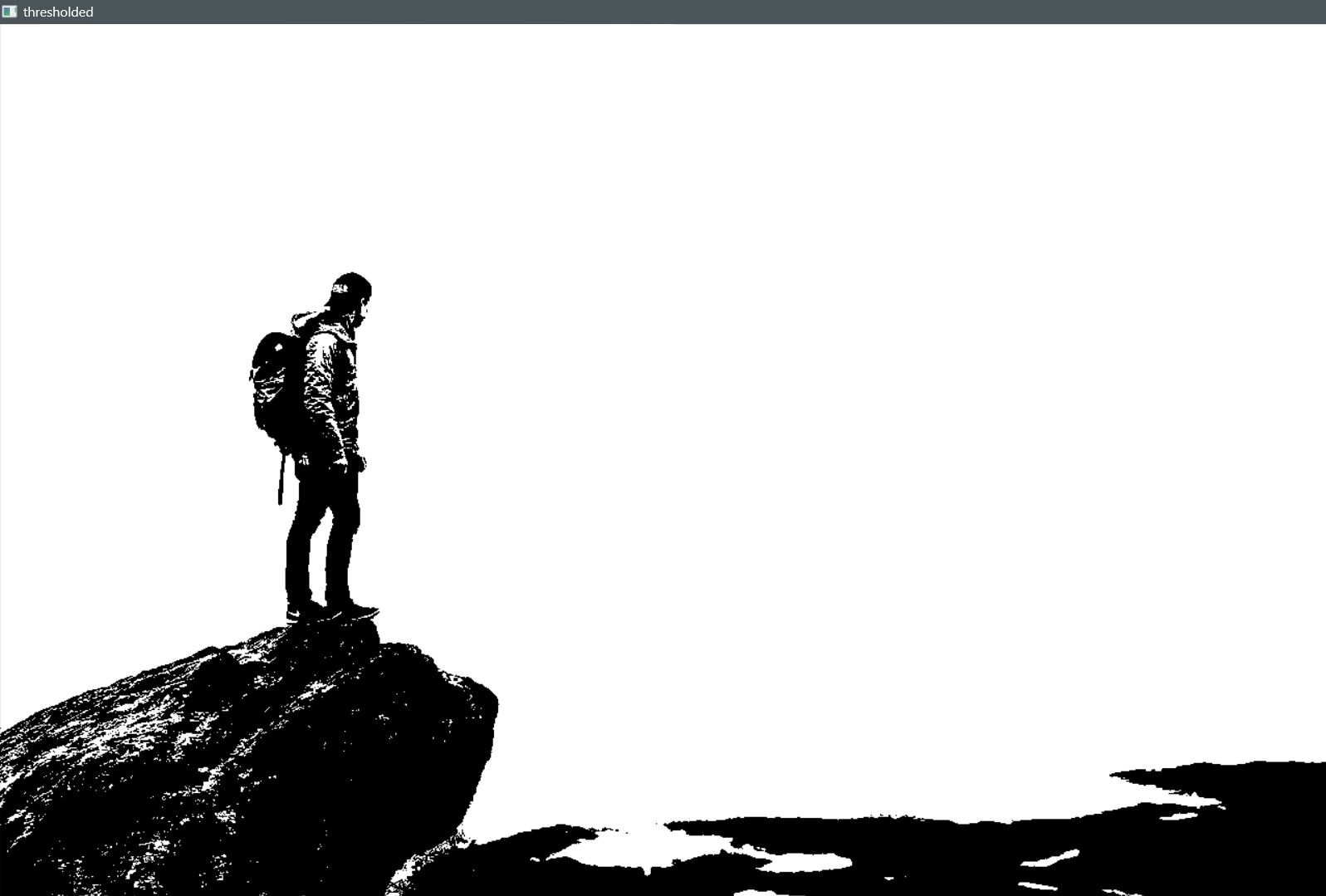
1. k = 83 (Rani)



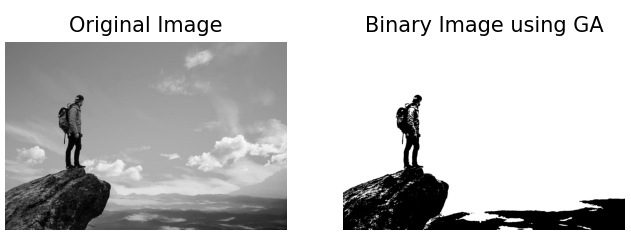
1. k = 108 (Holin)



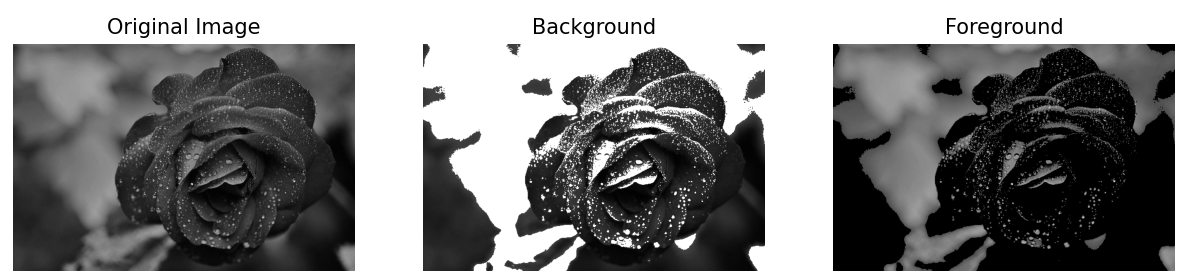
1. k = 108 (Wisnu)

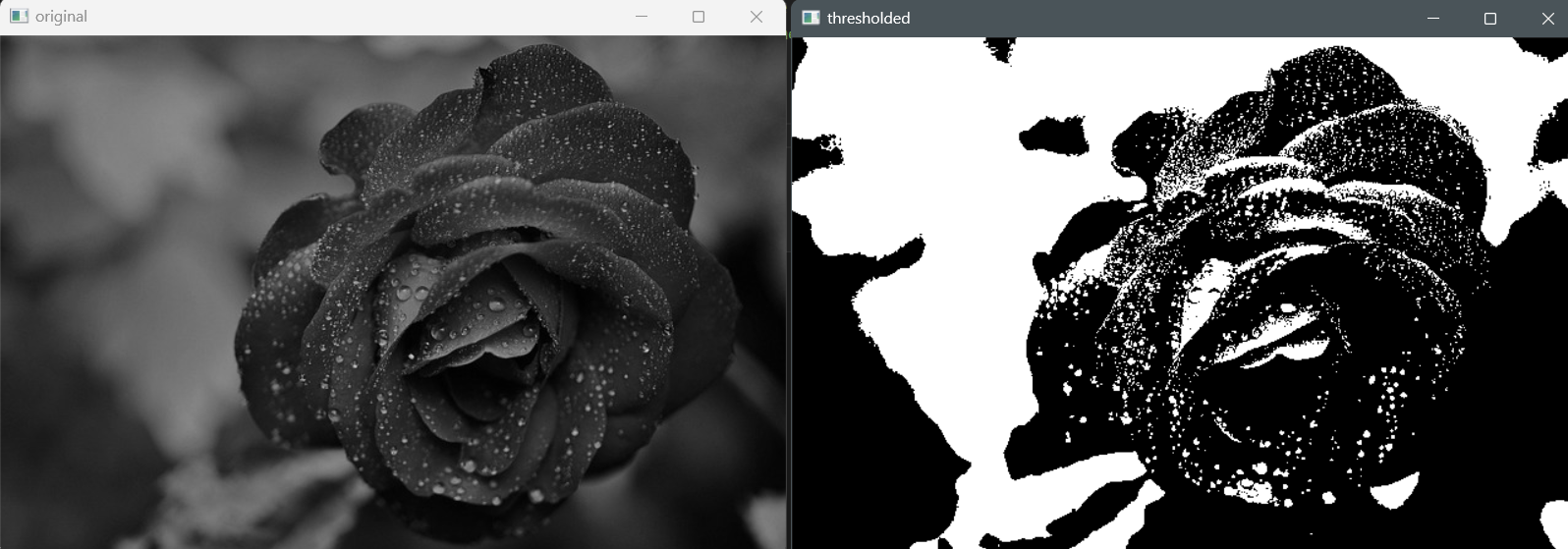
1. k = 107 (Rani)



1. k = 66 (Holin



1. k = 65 (Wisnu)

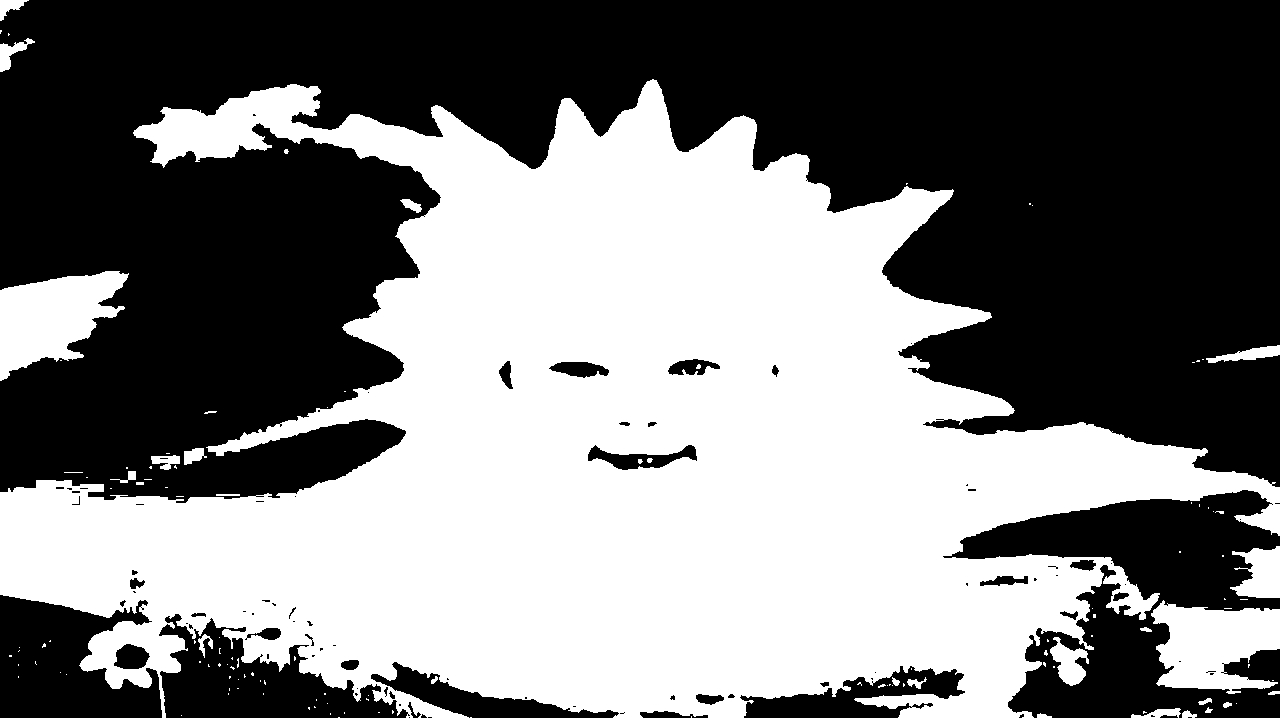


1. k = 66 (Rani)



**4.2 Analisis dan Pembahasan**

Dari berbagai program yang telah dibuat, kami menggunakan program tambahan pada 1 gambar yang digunakan untuk perbandingan sebagai berikut:



Gambar 4.2.1 Gambar Percobaan Fitness *Otsu Threshold*

dan Gambar 4.2.2 Hasil Binerisasi Paling Optimal

Perhitungan fitness dari *Otsu Thresholding* menggunakan variabel besaran variansi antara jumlah angka *pixel* terang dan *pixel* gelap. Rumus perhitungan *fitness* adalah sebagai berikut:

dimana,

* adalah Jumlah *Pixel* yang memiliki value maksimal (255)
* adalah jumlah *Pixel* total dalam gambar yang diinputkan
* adalah variansi dari jumlah *pixel* gelap
* adalah variansi dari jumlah *pixel* terang

Berdasarkan percobaan manual *Otsu Thresholding*, ditemukan variansi *thresholding* terbaik berada pada titik dalam tabel berikut:

| **Titik *Threshold*** | ***Fitness*** |
| --- | --- |
| … | … |
| 158 | 702.6913926 |
| 159 | 702.1030519 |
| 160 | 701.8171778 |
| 161 | 701.9117475 |
| 162 | 702.4162786 |
| … | … |

Dari perhitungan manual, didapatkan bahwa variansi *image* terbesar terdapat pada titik *threshold* 160 dari range {0-255}.

1. **Kode Holin**

Dalam kode metode ini, digunakan parameter sebagai berikut:

* *Population\_Size* : 12
* *Max\_Generations* : 30
* *Mutation\_Rate*  : 0.2

**Hasil Kode Pada Generasi Terakhir:**

| ##################  ##Generation 30 ##  ##################  pop (threshold val), fitness value  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  160 , 701.8171777858365  Best threshold: 160 |
| --- |



1. **Kode Rani**

Dalam kode ini, digunakan parameter sebagai berikut:

* *Population\_Size* : 50
* *Max\_Generations* : 100
* *Mutation\_Rate* : 0.01

***Best threshold found by GA*: 160**



Dapat dilihat bahwa dengan program manapun, dilakukan eksekusi kode secara berulang kali untuk menentukan apakah GA akan menemukan hasil yang sama ketika percobaan diulang. Nilai yang ditemukan adalah nilai maksimal yaitu 160. Hal ini membuktikan bahwa dengan implementasi manapun selama itu menggunakan algoritma genetika, nilai *threshold* yang tepat adalah 160.

# Kesimpulan

Dari hasil implementasi algoritma genetika dalam *Otsu Thresholding* dengan berbagai kode program, ditemukan bahwa nilai *threshold* optimal untuk memisahkan objek dari latar belakang dalam citra *grayscale* cenderung stabil di sekitar 160 dari rentang nilai piksel (0-255). Dengan menggunakan parameter yang telah ditentukan, seperti ukuran populasi, jumlah generasi, dan tingkat mutasi, algoritma genetika mampu secara efisien mengeksplorasi ruang solusi dan menemukan solusi yang baik. Hasil ini konsisten dengan perhitungan manual dan menunjukkan potensi algoritma genetika sebagai pendekatan yang efektif dalam menyelesaikan masalah segmentasi citra. Untuk meningkatkan performa dan validitas hasil, dapat dilakukan penyesuaian parameter dan validasi dengan metode *Otsu Thresholding* lainnya atau perhitungan manual, serta eksplorasi lebih lanjut untuk aplikasi dalam pemrosesan citra yang lebih kompleks.

# Referensi

1. Radillah, T., Ameliza, K., & Fitriyanto, I. (2023). Segmentasi citra menggunakan metode Otsu dalam pengenalan pola sederhana. *Indonesian Journal of Computer Science*, 12(6). https://doi.org/10.33022/ijcs.v12i6.3492
2. Rahmawati, A., Yulianti, I., & Nurajizah, S. (2023). Image segmentation analysis using Otsu thresholding and mean denoising for the identification of coffee plant diseases. *Jurnal Riset Informatika*, 6(1), 7-14. https://doi.org/10.34288/jri.v6i1.261
3. Restuning Pamuji, M. A., & Putra Pamungkas, D. (2023). Segmentation of red onion leaf images using Otsu’s thresholding method. *Nusantara of Engineering (NOE)*, 6(2), 169-174. https://doi.org/10.29407/noe.v6i2.20553
4. Alam, T., Qamar, S., Dixit, A., et al. (2020). Genetic algorithm: Reviews, implementations, and applications. *International Journal of Engineering Pedagogy (iJEP)*. https://doi.org/10.36227/techrxiv.12657173.v1
5. Hadiq, H., Solehatin, S., Djuniharto, D., Muslim, M. A., & Salahudin, S. N. (2023). Comparison of the suitability of the Otsu method thresholding and multilevel thresholding for flower image segmentation. *Journal of Soft Computing Exploration*, 4(4), 242-249
6. Shang, W., & Cheng, Y. (2016). An improved OTSU method based on Genetic Algorithm. In *2016 4th International Conference on Machinery, Materials and Information Technology Applications*. https://doi.org/10.2991/icmmita-16.2016.304
7. **LAMPIRAN**

***GITHUB* :** XXXXXXXXXXXXXX