Implementasi Algoritma Kruskal pada *Region Adjacency Graph* untuk *Preprocessing* Segmentasi Gambar Berbasis *Pixel-Grid*

Theresia Estelina Ratu Udju - 13525088
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: theresiaestelina@gmail.com , 13525088@std.stei.itb.ac.id

***Abstrak*— Segmentasi gambar merupakan salah satu tahapan penting dalam pengolahan citra digital yang bertujuan untuk membagi gambar menjadi beberapa region. Salah satu pendekatan berbasis graf yang dapat digunakan dalam proses segmentasi adalah *Region Adjacency Graph* (RAG). Makalah ini mengimplementasikan Algoritma Kruskal pada *Region Adjacency Graph* sebagai tahap *preprocessing* segmentasi gambar berbasis pixel-grid. Implementasi diawali dengan membentuk *superpixel* menggunakan metode SLIC, kemudian menyusun *Region Adjacency Graph* berdasarkan kemiripan warna antarregion yang bertetangga. Selanjutnya, Algoritma Kruskal digunakan untuk menghasilkan *Minimum Spanning Tree* (MST) dengan total bobot yang minimum. Hasil implementasi menunjukkan bahwa MST yang dihasilkan mampu menyederhanakan struktur graf tanpa menghilangkan konektivitas antarregion. Implementasi ini menunjukkan penerapan konsep teori graf dalam pengolahan citra digital serta potensi penggunaan Algoritma Kruskal sebagai metode *preprocessing* pada segmentasi gambar berbasis graf.**

***Kata Kunci*—citra digital, graf, region adjacecy graph, minimum spanning tree, algoritma kruskal, segmentasi gambar, superpixel**

I. PENDAHULUAN

Saat ini pengolahan citra digital sudah berkembang pesat dan dimanfaatkan di berbagai sektor kehidupan seperti fotografi, dunia komunikasi, keamanan, bahkan kedokteran. Pada makalah ini, istilah "gambar" yang digunakan pada judul merujuk pada citra digital, yaitu representasi visual yang terbentuk dari data digital. Salah satu tahap pada pengolahan citra digital adalah segmentasi citra (*image segmentation*). Segmentasi citra merupakan sebuah teknik untuk membagi citra menjadi beberapa region yang terpisah. Dengan mengurai visual yang kompleks menjadi segmen-segmen dengan bentuk tertentu, segmentasi citra membuat proses pengolahan citra menjadi lebih efisien [1].

Segmentasi citra merupakan salah satu masalah paling klasik dan paling banyak dipelajari dalam bidang *computer vision*. Meskipun berbagai metode segmentasi citra telah dikembangkan, kualitas segmentasi yang dihasilkan secara otomatis masih memiliki perbedaan yang cukup signifikan dibandingkan segmentasi yang dilakukan oleh manusia. Selain itu, banyak metode segmentasi modern memerlukan proses optimasi dan analisis hubungan antarwilayah yang kompleks sehingga membutuhkan representasi data yang lebih efisien untuk mendukung proses segmentasi [2].

Salah satu metode umum yang digunakan dalam pengolahan citra adalah representasi berbasis graf. Dalam pendekatan ini, hubungan antarelemen citra dimodelkan dalam bentuk simpul (*node*) dan sisi (*edge*). Region Adjacency Graph (RAG) merupakan salah satu representasi graf yang digunakan untuk menggambarkan hubungan antarwilayah pada citra. Setiap wilayah direpresentasikan sebagai simpul, sedangkan hubungan antarwilayah direpresentasikan sebagai sisi yang memiliki bobot berdasarkan karakteristik tertentu, seperti perbedaan warna atau intensitas. Representasi ini memungkinkan struktur citra dianalisis secara lebih terorganisir dibandingkan analisis langsung pada tingkat piksel [3].

Untuk menyederhanakan suatu graf kompleks yang terbentuk, Algoritma Kruskal dapat digunakan untuk mencari *Minimum Spanning Tree* (MST). Algoritma ini bekerja dengan secara bertahap memilih sisi-sisi dengan bobot minimum tanpa membentuk siklus hingga seluruh simpul terhubung. Hasil MST akan memiliki jumlah sisi yang lebih sedikit daripada graf awal namun konektivitas antarsimpulnya tetap dipertahankan [4]. Dengan begitu, Algoritma Kruskal ini dapat digunakan sebagai penyederhanaan graf sebelum proses segmentasi citra dilakukan.

Berdasarkan pembahasan tersebut, pada makalah ini diimplementasikan Algoritma Kruskal pada *Region Adjacency Graph* untuk preprocessing segmentasi citra berbasis pixel-grid. Implementasi dilakukan menggunakan bahasa pemrograman Python. Proses yang dilakukan meliputi pembentukan region dari citra menggunakan metode *superpixel*, penyusunan *Region Adjacency Graph*, penerapan Algoritma Kruskal untuk menghasilkan *Minimum Spanning Tree*, serta analisis perubahan struktur graf yang dihasilkan.

II. LANDASAN TEORI

A. Citra Digital

Secara sederhana citra dapat didefinisikan sebagai fungsi dua dimensi *f(x,y)*. Variabel *x* dan *y* merepresentasikan titik koordinat posisi, sedangkan nilai *f* merepresentasikan intensitas atau tingkat keabu-abuan citra pada titik tersebut. Ketika titik koordinat dan nilai intensitas tersebut berupa nilai-nilai yang terbatas dan diskrit, maka citra tersebut dikategorikan sebagai citra digital. Citra digital pada dasarnya merupakan representasi visual yang terbentuk dari struktur data digital yang dapat dipahami dan diolah oleh komputer. Citra ini tersusun dari titik-titik yang disebut piksel. Piksel tersebut tersusun dalam baris dan kolom, serta memiliki nilai yang menggambarkan tingkat kecerahan atau warna pada posisi tertentu dalam citra. Citra digital dapat diperolah dari berbagai sumber seperti hasil pemindai (*scanner*) atau kamera digital. Pengolahan citra digital adalah ilmu tentang bagaimana kita memproses citra-citra tersebut dengan bantuan komputer [1].

B. Segmentasi Citra

Segmentasi citra adalah suatu proses membagi-bagi citra menjadi beberapa bagian (segmen). Piksel-piksel yang saling berhubungan atau memiliki kemiripan visual dikelompokan menjadi satu segmen [2]. Dengan kata lain, setiap piksel yang mempunyai karakteristik serupa akan diberi label yang sama. Tujuan dari segmentasi ini adalah untuk menyederhanakan atau mengubah bentuk citra menjadi sesuatu yang lebih mudah untuk dianalisis. Teknik segmentasi citra dapat dikelompokan menjadi dua pendekatan. Pertama, pendekatan klasik, yaitu menggunakan pemisahan atau penggabungan region berdasarkan kriteria lokal piksel bertetangga. Kedua, pendekatan modern, yaitu mengopotimalkan kriteria global, seperti konsistensi internal region (*intra-region consistency*) dan ketidakmiripan batas antarregion (*inter-region dissimilarity*). Teknik-teknik yang telah dikembangkan untuk segmentasi citra, antara lain *active contours, level sets, mean shift, normalized cuts*, dan *graph cuts* [2].

C. SLIC Superpixel

Superpixel merupakan kelompok wilayah kecil pada citra yang terbentuk dari sekumpulan piksel yang memiliki karakteristik warna dan spasial yang serupa. Algoritma *Simple Linear Iterative Clustering* (SLIC) merupakan salah satu metode untuk menghasilkan *superpixel* dengan cepat. Metode ini merupakan hasil modifikasi dari algoritma *k-means* biasa agar bisa bekerja secara linear dan lebih efisien pada piksel citra. Secara umum, SLIC membagi citra menjadi *k* buah *superpixel* yang ukuran tiap wilayahnya hamper sama rata. Dengan menggunakan SLIC, kita menghindari ribuan perhitungan jarak yang redundan. SLIC dirancang secara khusus untuk menyelesaikan permasalahan klasterisasi *superpixel*. Kompleksitas algoritma SLIC adalah *O(N)* [5].

D. Graf

Suatu graf terdiri dari himpunan tak kosong unsur-unsur yang disebut sebagai simpul (*node*) dan himpunan pasangan dari simpul-simpul tersebut yang disebut sisi (*edge*). Graf dapat dilambangkan dengan *G = (V,E)*, yang artinya sebuah graf *G* memiliki pasangan *vertices (V)* dan *edges (E)*. Di dalam graf, setiap sisi berfungsi menghubungkan sepasang simpul, dan suatu graf diperbolehkan untuk tidak mengandung satu buah sisi pun di dalamnya. Dalam konteks pengolahan citra, simpul digunakan sebagai representasi unit visual, baik berupa piksel tunggal maupun kumpulan piksel yang homogen (*superpixel*). Sementara itu, sisi digunakan untuk merepresentasikan hubungan antarpiksel tersebut.

Jika sisi yang menghubungkan pasangan simpul di dalam graf memiliki suatu nilai numerik, maka graf tersebut dikategorikan sebagai graf berbobot (*weighted graph*). Dalam pengolahan citra, nilai bobot pada sisi ini digunakan untuk menyatakan nilai atau tingkat perbedaan karakteristik tertentu.

E. Region Adjacency Graph

Region Adjacency Graph (RAG) adalah sebuah graf tak berarah (*undirected graph*) yang dibuat dengan cara membagi-bagi citra menjadi bagian-bagian wilayah kecil yang saling bersentuhan (biasanya disebut *superpixel*). Di dalam graf ini, setiap potongan wilayah direpresentasikan oleh sebuah simpul, sedangkan sisi merepresentasikan bahwa region-region tersebut saling bertetangga atau menempel secara spasial. Representasi ini memungkinkan struktur citra dianalisis secara lebih efisien dibandingkan analisis langsung pada tingkat piksel yang berjumlah jutaan [3].

Proses pembentukan RAG dimulai setelah gambar dipecah menjadi beberapa wilayah homogen melalui tahapan segmentasi tingkat rendah, seperti algoritma *superpixel* (contohnya SLIC atau SLICO). Berdasarkan wilayah yang diperoleh dari proses tersebut, sebuah graf dibentuk dengan ketentuan satu wilayah direpresentasikan oleh satu simpul yang letak koordinatnya diambil dari nilai Tengah wilayah tersebut. Hubungan ketetanggaan atau posisi spasial antarwilayah yang saling berbagi garis batas kemudian ditandai dengan pemasangan sebuah sisi di antara simpul-simpulnya. Setiap sisi yang menghubungkan simpul di dalam RAG memiliki bobot yang digunakan untuk merepresentasikan nilai tingkat perbedaan atau ketidakmiripan antarwilayah yang berdampingan. Perhitungan bobot sisi ini dilakukan secara praktis dengan menghitung metrik kemiripan piksel, seperti perbedaan nilai rata-rata warna antarwilayah, serta perbedaan nilai fitur tekstur wilayah yang diekstrak menggunakan metode statistik, seperti *Gray-Level Co-occurrence Matrix* (GLCM) [3].

F. Minimum Spanning Tree

Minimum Spanning Tree (MST) atau Pohon Merentang Minimum merupakan suatu subgraph dari graf berbobot yang menghubungkan semua simpul dengan bobot yang minimum tanpa membuat siklus. Dengan mengurangi jumlah sisi yang tidak diperlukan, kompleksitas graf dapat diturunkan tanpa menghilangkan konektivitas antarregion. Secara umum ada dua algoritma klasik yang paling populer untuk mencari MST, yaitu Algoritma Prim dan Algoritma Kruskal. Kedua algoritma ini sama-sama menggunakan pendekatan *greedy* (memilih langkah terbaik di setiap tahap), tetapi memiliki cara kerja yang berbeda. Selain dua algoritma tersebut, ada juga algoritma lain yang digunakan untuk kebutuhan atau struktur graf yang lebih

spesifik, seperti Algoritma Boruvka dan Algoritma Reverse-Delete [4]. Jumlah sisi graf pada MST adalah sebanyak *n-1*, dengan *n* merupakan jumlah simpul.

G. Algoritma Kruskal

Algoritma Kruskal merupakan salah satu metode dengan pendekatan *greedy algorithm* yang sering digunakan untuk menyelesaikan permasalahan MST. Algoritma ini akan selalu memilih dan menambahkan sisi dengan bobot paling minimal ke dalam graf atau pohon yang sedah dibentuk.

Secara teknis, algoritma ini memanfaatkan himpunan saling lepas (*disjoint-set*) untuk mengelola elemen-elemen yang terpisah [4]. Tahap awal algoritma ini dimulai dengan memilih sisi dengan bobot terkecil, lalu memeriksa apaka sisi tersebut membentuk siklus. Jika tidak membentuk siklus, sisi ditambahkan ke MST. Proses diulangi hingga diperoleh MST yang setiap simpulnya terhubung satu sama lain. Secara keseluruhan, kompleksitas Algoritma Kruskal adalah *O(E log E)* [4].

III. Analisis dan Pembahasan

A. Citra Masukan

Gambar 1 Citra Masukan
(Sumber: indonesiana.id)

Citra yang digunakan dalam analisis ini adalah citra berwarna dengan yang menampilkan berbagai jenis buah dan sayuran yang diletakan dalam suatu keranjang. Secara visual, citra memiliki beberapa region yang dapat dibedakan dengan jelas, seperti area keranjang berwarna cokelat, kelompok buah-buahan dan sayuran berwarna merah, kuning, dan hijau. Perbedaan karakteristik warna dan tekstur antar objek tersebut memungkinkan proses pembentukan *superpixel* dan *Region Adjacency Graph* dilakukan dengan baik.

Citra tersebut akan digunakan sebagai data masukan untuk proses pembentukan *superpixel* menggunakan algoritma SLIC. Setiap *superpixel* yang terbentuk selanjutnya direpresentasikan sebagai simpul pada *Region Adjacency Graph*, sedangkan hubungan ketetanggaan antar *superpixel* direpresentasikan sebagai sisi. Struktur graf yang dihasilkan kemudian diproses menggunakan Algoritma Kruskal untuk membentuk *Minimum Spanning Tree* sebagai tahap preprocessing segmentasi citra.

B. Pembentukan Superpixel

Pada analisis ini, proses pembentukan superpixel dilakukan dengan algoritma *Simple Linear Iterative Clustering* (SLIC) yang tersedia pada pustaka Python scikit-image. Tahap ini bertujuan untuk mengelompokkan piksel-piksel yang memiliki karakteristik warna dan Lokasi yang serupa ke dalam satu region. Dengan demikian, piksel tidak lagi diproses secara individual pada tahap-tahap selanjutanya, melainkan diproses berdasarkan region yang telah terbentuk.

Parameter yang digunakan adalah *n_segments* = 100 dan *compactness* = 10. Parameter *n_segments* menentukan jumlah *superpixel* yang akan dibentuk, sedangkan *compactness* mengatur keseimbangan antara kemiripan warna dan kedekatan spasial dalam proses pengelompokan piksel.

Implementasi pembentukan *superpixel* dilakukan menggunakan kode berikut:

```
from skimage import io, segmentation

image = io.imread("input.jpg")

segments = segmentation.slic(
    image,
    n_segments = 100,
    compactness = 10,
    start_label = 1
)
```

Gambar 2 Hasil Pembentukan *Superpixel*

C. Pembentukan Region Adjacency Graph

Setelah pembentukan *superpixel*, tahap selanjutnya adalah pembentukan Region Adjacency Graph (RAG). RAG digunakan untuk menyatakan hubungan antarregion yang dihasilkan dari metode SLIC sebelumnya. Dalam graf ini, setiap *superpixel* direpresentasikan sebagai simpul, sedangkan hubungan ketetanggaan antar-*superpixel* direpresentasikan sebagai sisi.

Pembentukan RAG dilakukan dengan menggunakan fungsi *rag_mean_color()* dari pustaka Python *scikit-image*. Fungsi tersebut menyusun graf berdasarkan hubungan ketetanggaan antarregion dan menghitung bobot sisi berdasarkan perbedaan warna rata-rata dari dua region yang saling bertetangga.

Semakin kecil perbedaan warna antarregion, semakin kecil bobot sisi yang menghubungkan kedua simpul tersebut.

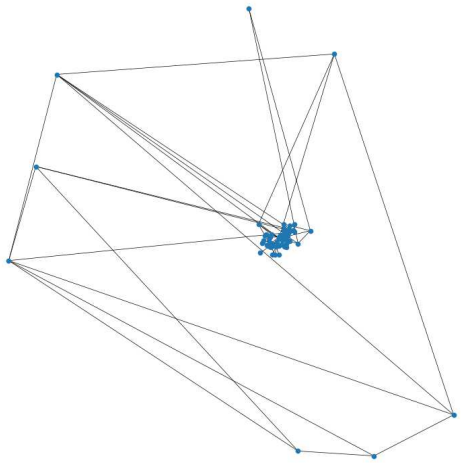
Implementasi pembentukan RAG dilakukan menggunakan kode berikut:

```
from skimage import graph

rag = graph.rag_mean_color(
    image,
    segments,
    mode = 'distance'
)

print("=== Hasil RAG ===")
print("Jumlah Simpul :",
rag.number_of_nodes())
print("Jumlah Sisi :",
rag.number_of_edges())
```

Output

```
=== Hasil RAG ===
Jumlah Simpul : 58
Jumlah Sisi : 138
```

Pada kode tersebut, parameter *image* merupakan citra masukan, sedangkan *segments* merupakan hasil segmentasi *superpixel*. Parameter *mode = 'distance'* digunakan untuk menghitung bobot sisi berdasarkan jarak warna rata-rata antar region.



Gambar 3 Visualisasi Graf Hasil RAG

D. Penerapan Algoritma Kruskal

Tahap selanjutnya adalah penerapan Algoritma Kruskal untuk menghasilkan *Minimum Spanning Tree* (MST). Tujuan dari tahap ini adalah menyederhanakan graf yang kompleks dengan mengurangi jumlah sisinya, tetapi tetap mempertahankan konektivitas tiap simpulnya.

Pada analisis ini, implementasi Algoritma Kruskal dilakukan menggunakan pustaka *NetworkX* pada Python. Graf berbobot hasil pembentukan RAG diproses menggunakan fungsi *minimum_spanning_tree()*.

Implementasi Algoritma Kruskal dilakukan menggunakan kode berikut:

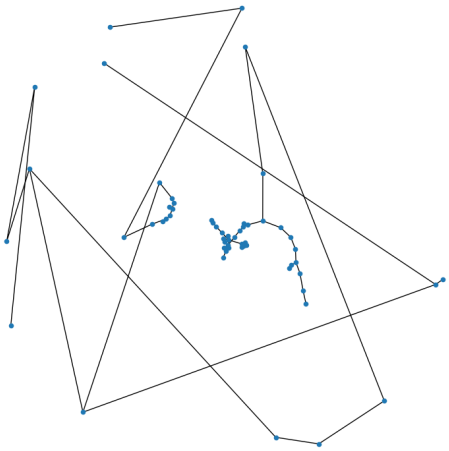
```
import networkx as nx

mst = nx.minimum_spanning_tree(
    rag,
    weight = 'weight',
    algorithm = 'kruskal'
)

print("=== Hasil MST ===")
print("Jumlah Simpul :",
mst.number_of_nodes())
print("Jumlah Sisi :",
mst.number_of_edges())
```

Output

```
=== Hasil MST ===
Jumlah Simpul : 58
Jumlah Sisi : 57
```

Pada kode tersebut, parameter *weight = 'weight'* digunakan untuk menyatakan bahwa pemilihan sisi dilakukan berdasarkan bobot graf, sedangkan parameter *algorithm = 'kruskal'* menunjukkan bahwa proses pembentukan MST dilakukan menggunakan Algoritma Kruskal.

Hasil dari tahap ini merupakan graf baru dengan jumlah simpul yang sama dengan graf awal, namun dengan jumlah sisi yang lebih sedikit. Graf hasil MST mempertahankan hubungan antarregion sehingga setiap simpul tetap terhubung.



Gambar 4 Visualisasi Graf Hasil MST

E. Hasil Segmentasi

Setelah MST berhasil didapatkan, segmentasi citra dapat dilakukan sebagai salah satu tahap untuk pengolahan citra. Pada analisis ini segmentasi dilakukan dengan memutus sisi yang memiliki bobot lebih besar dari nilai *threshold*, yaitu 20.

Setelah itu graf akan dibagi menjadi beberapa *connected components* yang setiap komponennya dianggap menjadi satu segmen dan direpresentasikan kembali dengan bentuk citra menggunakan warna rata-rata segmen.

Segmentasi citra dilakukan menggunakan kode berikut:

```
import matplotlib.pyplot as plt
import numpy as np
import networkx as nx

threshold = 20

edges_to_remove = []

for u, v, data in mst.edges(data=True):

    if data["weight"] > threshold:
        edges_to_remove.append((u, v))

mst.remove_edges_from(edges_to_remove)


components = list(
    nx.connected_components(mst)
)


new_labels = np.zeros_like(
    segments,
    dtype = np.int32
)

for label_id, comp in enumerate(
        components,
        start = 1):

    for node in comp:

        new_labels[
            segments == node
        ] = label_id


segmented = label2rgb(
    new_labels,
    image,
    kind = 'avg'
)

plt.figure(figsize = (8,6))
plt.imshow(segmented)
plt.title("Hasil Segmentasi")
plt.axis("off")
plt.show()
```

Gambar 5 Hasil Segmentasi Citra

Berdasarkan hasil segmentasi tersebut, dapat dilihat bahawa citra berhasil dibagi menjadi beberapa segmen. Region-region yang memiliki kemiripan warna dan berdekatan cenderung menjadi satu segmen yang sama, sedangkan region yang memiliki perbedaan warna signifikan terpisah pada segmen yang berbeda.

Pada citra masukan yang digunakan, masih terdapat beberapa objek yang belum terpisah secara sempurna dan msih terdapat objek-objek berbeda yang tergabung dalam segmen yang sama. Hal ini dapat disebabkan oleh kemiripan warna antarobjek, kode Python yang belum optimal, serta penggunaan nilai *threshold* yang belum optimal. Meskipun begitu, hasil segmentasi menunjukkan bahwa proses pembentukan *superpixel*, penyusunan *Region Adjacency Graph*, dan penerapan Algoritma Kruskal mampu menghasilkan pengelompokan region yang cukup representatif terhadap struktur citra.

F. Analisis Hasil

Berdasarkan implementasi yang dilakukan, *superpixel* yang merepresentasikan kelompok piksel dengan karakteristik serupa berhasil dibentuk dengan metode SLIC. Penggunaan *superpixel* ini membantu tahap-tahap selanjutnya karena analisis yang dilakukan bukan lagi menggunakan piksel tunggal, tetapi piksel-piksel yang telah dikelompokan menjadi region-region. Pada citra masukan

Region yang terbentuk kemudian direpresentasikan menggunakan *Region Adjacency Graph*. Graf yang dihasilkan mampu menggambarkan hubungan antarregion serta memberi nilai numerik atau bobot yang menunjukkan tingkat kemiripan warna antar region. Melalui tahap ini dihasilkan RAG dengan jumlah simpul sebanyak 58 dan jumlah sisi sebanyak 138.

Penerapan Algoritma Kruskal pada RAG berhasil menghasilkan *Minimum Spanning Tree* yang menghubungkan setiap simpul dengan jumlah sisi yang lebih sedikit dengan bobot yang minimum. Hasil yang didapatkan adalah perubahan jumlah sisi graf dari 138 menjadi 57 (sebanyak simpul – 1). Algoritma Kruskal berhasil digunakan untuk menyederhanakan graf yang komplek, namun tetap mempertahankan konektivitas antarsimpulnya.

Melalui hasil segmentasi, dapat diketahui bahwa region-region yang memiliki kemiripan dari segi warna akan cenderung dikelompokan menjadi segmen yang sama. Meskipun pemisahan objek belum sepenuhnya sempurna, proses yang dilakukan telah menunjukkan bagaimana kombinasi *superpixel*, RAG, dan Algoritma Kruskal dapat digunakan sebagai tahap *preprocessing* pada segmentasi gambar berbasis graf. Berdasarkan pembahasan tersebut, didapatkan bahwa implementasi yang dilakukan berhasil menunjukkan penerapan teori graf, khususnya *Region Adjacency Graph* dan Algoritma Kruskal, dalam pengolahan gambar atau citra digital.

IV. KESIMPULAN

Pengolahan citra digital merupakan bidang yang terus berkembang dan banyak dimanfaatkan dalam berbagai sektor. Salah satu bentuk pengolahan citra digital adalah segmentasi citra. Pada makalah ini telah dilakukan implementasi Algoritma Kruskal pada *Region Adjacency Graph* (RAG) sebagai tahap *preprocessing* segmentasi citra berbasis pixel-grid.

Proses diawali dengan membentuk *superpixel* dengan menggunakan metode SLIC untuk pengelompokan piksel mwnjadi beberapa region. Region yang terbentuk kemudian direpresentasikan dalam bentuk RAG dengan setiap region merupakan simpul dan hubungan antarregion merupakan sisi berbobot. Algoritma Kruskal diimplementasikan untuk menyederhanakan RAG. Dari algoritma tersebut berhasil didapatkan *Minimum Spanning Tree* (MST) yang menghubungkan seluruh region dengan jumlah sisi dan bobot yang minimum.

Berdasarkan hasil segmentasi yang diperoleh, region-region dengan karakteristik serupa umumnya tergabung ke dalam kelompok atau segmen yang sama, meskipun masih terdapat beberapa objek yang belum terpisah secara sempurna. Hal ini menunjukan bahwa kualitas segmentasi dipengaruhi oleh parameter *superpixel* dan kualitas citra yang digunakan. Walaupun begitu, implementasi yang dilakukan sudah berhasil menunjukkan bahwa kombinasi *superpixel,* RAG, dan Algoritma Kruskal dapat digunakan sebagai pendekatan *preprocessing* segmentasi citra.

Secara keseluruhan, makalah ini menunjukkan bahwa konsep teori graf, khususnya RAG dan Algoritma Kruskal, dapat diterapkan untuk menyederhanakan representasi gambar dan membantu proses segmentasi dalam pengolahan citra digital.

V. SARAN

Berdasarkan implementasi dan analisis yang telah dilakukan, terdapat beberapa saran yang dapat dipertimbangkan unuk pengembangan lebih lanjut. Pertama, penggunaan jumlah *superpixel* dan nilai *threshold* yang lebih bervariasi, serta optimasi parameter-parameter tersebut untuk memperoleh hasil segmentasi yang lebih baik. Kedua, menggunakan lebih banyak gambar atau citra dengan karakteristik yang berbeda, seperti variasi warna, tekstur, dan kompleksitas objek. Dengan demikian, kinerja metode yang digunakan dapat dievaluasi lebih lanjut. Ketiga, optimasi dan eksplorasi kode Python untuk analisis lebih lanjut dan mendapatkan variasi hasil lain yang lebih baik.

VI. UCAPAN TERIMAKASIH

Penulis mengucapkan syukur kepada Tuhan Yang Maha Esa karena atas segala berkat dan kasih karunia-Nya penulis dapat menyelesaikan makalah ini. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada dosen pengampu mata kuliah matematika diskrit, Dr. Ir. Rinaldi, M.T., yang telah memberikan ilmu dan pengetahuan sehingga dapat penulis terapkan dalam makalah ini. Terakhir, penulis berterima kasih kepada orang tua dan teman-teman penulis yang senantiasa mendukung dan memberikan semangat kepada penulis selama proses pembuatan makalah ini.

VII. LAMPIRAN

Link Repositori Github:
https://github.com/there514/kruskal-rag-image-segmentation/tree/main

REFERENSI

[1] R. C. Gonzalez dan R. E. Woods, Digital Image Processing, 4th ed. New York: Pearson, 2018.

[2] R. Szeliski, Computer Vision: Algorithms and Applications. Springer, 2010.

[3] Emergent Mind, "Region Adjacency Graph (RAG)," 15 Desember 2025. Diperoleh dari: https://www.emergentmind.com/topics/region-adjacency-graph-rag. Diakses pada: 16 Juni 2026.

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Introduction to Algorithms, 3rd ed. Cambridge: MIT Press, 2009.

[5] T. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 11, November 2012.

[6] R. Munir, "Graf (Bagian 1)," Program Studi Teknik Informatika, STEI-ITB, 2026. Diperoleh dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/20-Graf-Bagian1-2026.pdf. Diakses pada: 16 Juni 2026.

[7] R. Munir, "Pohon (Bagian 1)," Program Studi Teknik Informatika, STEI-ITB, 2026. Diperoleh dari: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2025-2026/23-Pohon-Bag1-2026.pdf. Diakses pada: 16 Juni 2026..

[8] NetworkX Developers, "minimum_spanning_tree," NetworkX Documentation. Diperoleh dari: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.tree.mst.minimum_spanning_tree.html. Diakses pada: 16 Juni 2026.

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, 19 Juni 2026

Theresia Estelina Ratu Udju, 13525088