Pembuatan Filter *Cut Out* pada Pengolahan Citra melalui Segmentasi Citra dengan Memanfaatkan Algoritma Kruskal

Chiquita Ahsanunnisa - 13521129
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521129@std.stei.itb.ac.id

***Abstract*—Filter *cut out* adalah salah satu filter citra yang cukup populer dalam dunia fotografi dan desain grafis. Filter ini dapat memberikan tampilan yang lebih grafis atau kartun pada gambar dengan menghilangkan beberapa detail dan menghasilkan bentuk-bentuk sederhana dan kontras yang kuat. Dasar dari pembuatan filter ini adalah segmentasi citra. Salah satu algoritma yang dapat dimanfaatkan untuk mensegmentasi citra adalah dengan mengaplikasikan algoritma Kruskal—yang sejatinya digunakan untuk memecahkan persoalan pohon merentang minimum—pada citra berbasis graf.**

Keywords—filter cut out; segmentasi citra; algoritma Kruskal; pengolahan citra

I. Pendahuluan

Dalam era digital yang semakin maju, teknologi telah membawa perubahan signifikan dalam berbagai aspek kehidupan manusia, termasuk dalam bidang fotografi dan media sosial. Salah satu tren yang semakin populer adalah penggunaan filter citra. Filter citra adalah alat atau teknik yang digunakan untuk memodifikasi tampilan visual sebuah citra, dengan tujuan menciptakan tampilan yang unik, menarik, atau dramatis.

Salah satu filter citra yang cukup populer dalam dunia fotografi dan desain grafis adalah filter *cut out*. Filter *cut out* adalah filter atau efek yang digunakan untuk mempertegas atau menyoroti kontur objek dalam gambar dengan menghapus detail halus dan menghasilkan garis kontur yang tegas. Filter ini dapat memberikan tampilan yang lebih grafis atau kartun pada gambar dengan menghilangkan beberapa detail dan menghasilkan bentuk-bentuk sederhana dan kontras yang kuat.

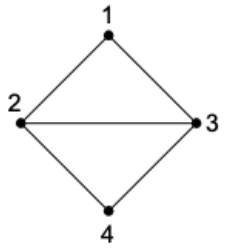
Ada banyak cara untuk membuat filter *cut out*. Namun, pada dasarnya, langkah utama dari pembuatan filter ini adalah segmentasi citra. Salah satu algoritma yang dapat dimanfaatkan untuk mensegmentasi citra adalah dengan mengaplikasikan algoritma Kruskal—yang sejatinya digunakan untuk memecahkan persoalan pohon merentang minimum—pada citra berbasis graf.

II. Landasan Teori

A. Graf dan Pohon

Graf adalah struktur data yang digunakan untuk merepresentasikan hubungan antara objek-objek diskrit. Secara formal, sebuah graf didefinisikan dengan *tuple* (V, E), dengan V adalah himpunan tidak kosong dari simpul-simpul (*vertices*) dan E adalah himpunan sisi-sisi (*edges*) yang menghubungkan sepasang simpul.

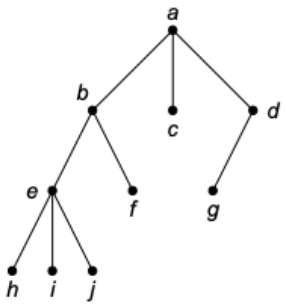
Pada graf, dikenal istilah lintasan (path) dan sirkuit (cycle). Lintasan yang panjangnya n dari simpul awal v_0 ke simpul tujuan v_n di dalam graf G ialah barisan berselang-seling simpul-simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, v_n$ sedemikian sehingga $e_1 = (v_0, v_1), e_2 = (v_1, v_2), ..., e_n = (v_{n-1}, v_n)$ adalah sisi-sisi dari graf *G*. Sirkuit adalah lintasan yang berawal dan berakhir pada simpul yang sama.



Gambar 1. Graf G_1 (Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/)

Pada graf G_1 di atas, jalur 1-3-4 adalah sebuah lintasan, sedangkan jalur 1-3-4-2-1 adalah sebuah sirkuit.

Pohon adalah graf yang terhubung dan tidak mengandung sirkuit. Selain itu, ada juga istilah pohon berakar. Pohon berakar (*rooted tree*) adalah pohon yang salah satu simpulnya diperlakukan sebagai akar dan sisi-sisinya memiliki arah (merupakan graf berarah juga). Pada Gambar 1, a merupakan akar dari pohon tersebut.



Gambar 2. Pohon Berakar (Sumber: informatika.stei.itb.ac.id/~rinaldi.munir/)

Pada pohon berakar, dikenal istilah simpul anak (*child*) dan simpul orangtua (*parent*). Pada Gambar 2, a adalah orangtua dari b, c, dan d. Begitupun sebaliknya, b, c, dan d adalah anak dari a.

B. Persoalan Pohon Merentang Minimum

Pohon merentang dari sebuah graf yang terubung adalah upagraf merentang yang berupa pohon. Pohon merentang diperoleh melalui pemutusan sirkuit di dalam graf.

Sebuah graf terhubung yang berbobot bisa jadi memiliki lebih dari satu pohon merentang. Di antara kemungkinan tersebut, ada satu pohon merentang yang memiliki total bobot minimum. Pohon tersebut disebut pohon merentang minimum atau *minimum spanning tree* (MST).

Persoalan pohon merentang minimum memiliki banyak aplikasi. Di bidang pengolahan citra, persoalan ini dapat diaplikasikan untuk segmentasi pada citra berbasis graf.

C. Algoritma Kruskal

Algoritma Kruskal merupakan algoritma yang digunakan untuk memecahkan persoalan pohon merentang minimum. Algoritma ini tergolong dalam jenis algoritma *greedy*. Strategi *greedy* yang digunakan pada algoritma ini yaitu dengan memilih sisi yang memiliki bobot minimum pada setiap langkah. Jika sisi tersebut tidak membentuk sirkuit pada pohon merentang yang sedang dibangun, tambahkan sisi tersebut ke pohon merentang yang sedang dibangun.

Misalkan T adalah himpunan yang merepresentasikan pohon merentang minimum yang akan dibangun dari graf G yang memiliki n simpul. Berikut merupakan langkah-langkah dari algoritma Kruskal.

1. Urutkan sisi-sisi graf G secara menaik berdasarkan bobotnya.
2. Pilih sisi dengan bobot minimum yang tidak membentuk sirkuit pada T. Tambahkan sisi tersebut ke dalam T.
3. Ulangi langkah 2 sebanyak n-1 kali.

D. Algoritma Union-Find

Pada algoritma Kruskal yang sudah dipaparkan di bagian sebelumnya, diperlukan pengecekan sirkuit dan penambahan sisi pada pohon merentang yang akan dibangun. Langkah ini juga merupakan sebuah persoalan yang tidak mudah.

Untuk itu, diperlukan suatu algoritma menyelesaikan persoalan ini. Salah satu algoritma yang dapat digunakan untuk pengecekan sirkuit dan penambahan sisi pada pohon merentang adalah algoritma Union-Find yang diaplikasikan pada struktur data *disjoint set union*.

Berikut adalah langkah-langkah dari algoritma Union-Find.

1. Buat himpunan-himpunan *disjoint* untuk setiap simpul pada graf.
2. Untuk setiap sisi (u,v) pada graf:
   - Cari akar dari himpunan tempat u dan v berada.
   - Jika u dan v memiliki akar yang sama dari himpunan *disjoint*-nya, sisi tersebut menyebabkan graf mengandung sirkuit.
   - Jika sebaliknya, lakukan *union* dari himpunan *disjoint*-nya.

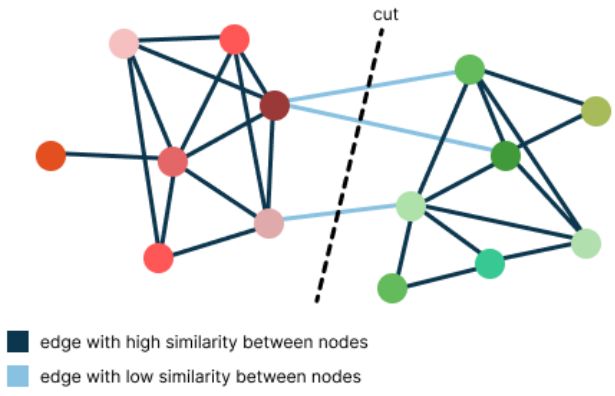
E. Klasterisasi Graf

Klasterisasi graf adalah proses pemecahan graf menjadi beberapa upagraf yang mewakili kluster. Setiap upagraf terdiri atas simpul-simpul yang memiliki kemiripan karakteristik yang tinggi.

Prinsip dari klasterisasi graf adalah mengelompokkan graf dengan melakukan pemotongan (*graph cut*). Kriteria dari pemotongan graf yaitu sebagai berikut.

- Simpul yang bersisian dari upagraf yang sama memiliki kemiripan karakteristik yang tinggi.
- Simpul yang bersisian dari upagraf yang berbeda memiliki kemiripan karakteristik yang rendah.

Proses pemotongan pada klasterisasi graf diilustrasikan pada Gambar 3.



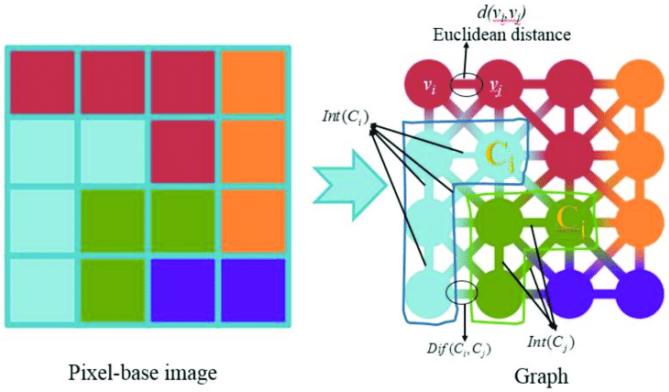
Gambar 3. Klasterisasi Graf (Sumber: Dokumentasi Penulis)

Jika ukuran yang digunakan untuk membandingkan hubungan antarsimpul adalah ukuran perbedaan karakteristik (lawan dari ukuran kemiripan karakteristik), dapat dilihat bahwa total ukuran perbedaan pada tiap sisi dalam satu upagraf harus minimum.

Klasterisasi graf dapat diaplikasikan pada banyak hal. Dalam konteks pengolahan citra, klasterisasi graf dapat digunakan untuk segmentasi pada citra berbasis graf.

F. *Citra Berbasis Graf*

Pada umumnya, citra direpresentasikan melalui struktur data matriks. Namun, citra juga dapat direpresentasikan melalui struktur data graf.



Gambar 4. Citra Berbasis Graf

Graf yang merepresentasikan citra memiliki simpul berupa piksel pada gambar dan sisi yang menghubungkan piksel-piksel tersebut. Simpul mempunyai data warna dalam notasi vektor RGB (*red*, *green*, *blue*). Sisi mempunyai bobot yang merepresentasikan satuan perbedaan antara kedua simpul yang dihubungkan.

Misalkan terdapat simpul v1 yang memiliki vektor warna (r_1, g_1, b_1) dan simpul v2 yang memiliki vektor warna (r_2, g_2, b_2). Perbedaan (d) antara kedua simpul tersebut dapat dihitung melalui jarak euclidean antara kedua vektor RGB dengan rumus sebagai berikut.

$$d^2 = (r_1 - r_2)^2 + (g_1 - g_2)^2 + (b_1 - b_2)^2 \quad (1)$$

Nilai d di atas lah yang menjadi bobot pada sisi graf. Sebagai catatan, untuk meminimumkan proses kalkulasi pada program, dapat juga digunakan d^2 sebagai ukuran perbedaan karena nilai d^2 sebanding dengan d. Hal ini berlaku karena ukuran perbedaan d memiliki nilai berupa bilangan nonnegatif.

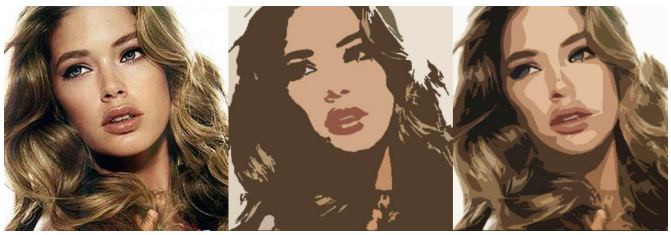
G. *Segmentasi Citra*

Segmentasi citra adalah proses pemisahan suatu citra menjadi segmen-segmen berdasarkan atribut karakteristiknya, seperti warna, tekstur, bentuk, atau kecerahan. Segmentasi citra pada umumnya bertujuan untuk menyederhanakan citra sehingga lebih bermakna dan lebih mudah dianalisis.

Salah satu jenis segmentasi citra yang umum digunakan adalah segmentasi region. Segmentasi region membagi citra menjadi region-region yang memiliki kriteria yang mirip. Salah satu teknik yang umum digunakan untuk segmentasi region pada pengolahan citra adalah segmentasi berbasis graf (*graph based segmentation*). Teknik ini diaplikasikan pada citra berbasis graf. Prinsip dari teknik ini adalah klusterisasi graf berdasarkan karakteristik piksel.

H. *Filter Cut Out pada Pengolahan Citra*

Dalam konteks pengolahan citra, filter *cut out* adalah filter atau efek yang digunakan untuk mempertegas atau menyoroti kontur objek dalam gambar dengan menghapus detail halus dan menghasilkan garis kontur yang tegas. Filter ini dapat memberikan tampilan yang lebih grafis atau kartun pada gambar dengan menghilangkan beberapa detail dan menghasilkan bentuk-bentuk sederhana dan kontras yang kuat.



Gambar 5. Filter *Cut Out*

Konsep dari filter *cut out* sebenarnya adalah dengan membuat warna pada citra menjadi *flat*. Hal ini dilakukan dengan memperkecil variasi warna pada citra dengan mengubah warna menjadi warna rerata untuk setiap segmennya.

Misalkan sebuah segmen dari citra memiliki n simpul yang dilambangkan dengan vi. Warna rataan (dalam vektor RGB) untuk segmen tersebut dihitung dengan rumus berikut.

$$warna\ rataan = \frac{\sum_{i=1}^{n} v_i}{n} \quad (2)$$

III. Implementasi

Secara keseluruhan, citra yang akan diberikan filter *cut out* akan melalui lima tahap proses. Berikut adalah lima tahap proses tersebut.

1. Tahap Preproses
2. Tahap Transformasi Citra menjadi Graf
3. Tahap Segmentasi Citra
4. Tahap Aplikasi Filter *Cut Out*
5. Tahap Penyesuaian Citra Hasil Filter *Cut Out*

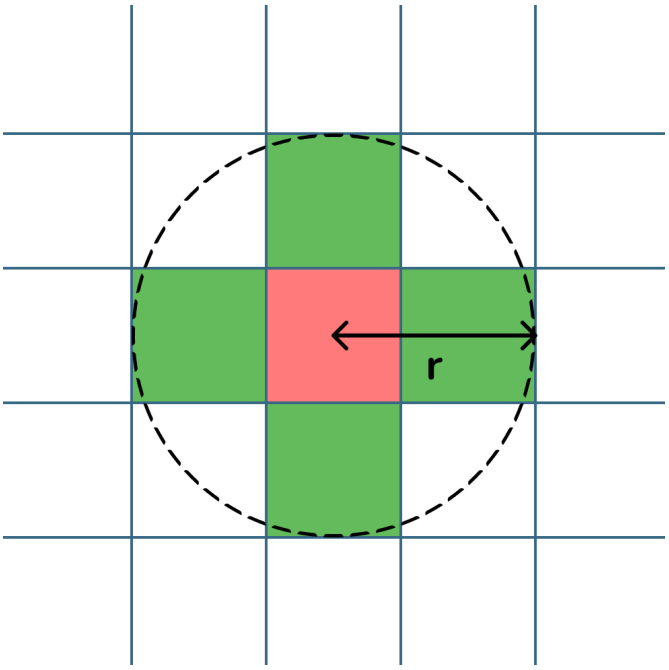
Setiap tahap di atas dirincikan pada bagian di bawah ini.

A. *Tahap Preproses*

Pada tahap preproses, citra masukan direpresentasikan dalam struktur data matriks berisi elemen vektor RGB untuk tiap pikselnya. Kemudian, citra masukan diberi filter *smoothing*. Hal ini bertujuan untuk mengurangi *noise* yang ada pada citra masukan sehingga hasil pemberian filter *cut out* dapat lebih akurat. Selain itu, pada tahap ini dilakukan pengubahan ukuran citra jika ukuran citra masukan sangat besar (panjang atau lebarnya lebih dari 500 piksel). Hal ini dilakukan karena ukuran citra yang terlalu besar menyebabkan penggunaan memori pada program semakin besar. Pengubahan ukuran dilakukan dengan perhitungan perbandingan biasa.

B. *Tahap Transformasi Citra menjadi Graf*

Pada tahap ini, citra yang awalnya direpresentasikan dalam struktur data matriks ditransformasikan menjadi graf yang terdiri atas simpul dan sisi. Simpul-simpul graf sudah terwakilkan oleh data matriks. Untuk data sisi graf, tidak perlu dilakukan konstruksi sisi untuk setiap pasangan simpul yang ada. Konstruksi sisi cukup dilakukan untuk simpul yang berada pada radius *r* dari simpul (lihat Gambar 6). Hal ini bertujuan agar perhitungan yang dilakukan lebih efektif dan memori pada *runtime* tidak dihabiskan untuk menyimpan sisi yang tidak akan dipakai. Bobot pada sisi dihitung dengan persamaan (1).



Gambar 6. Pembuatan Sisi Graf berdasarkan Radius (Sumber: Dokumentasi Penulis)

C. *Tahap Segmentasi Citra*

Pada tahap ini, citra dipecah-pecah menjadi beberapa segmen region. Proses segmentasi ini memanfaatkan algoritma Kruskal dan algoritma Find-Union.

Pada bagian Klusterisasi Graf, telah disebutkan bahwa untuk melakukan klusterisasi graf menjadi segmen-segmen upagraf, total ukuran perbedaan pada tiap sisi dalam satu upagraf harus minimum. Dalam kasus segmentasi citra, hal ini dapat diakali dengan membuat pohon merentang yang minimum pada tiap segmen. Pemotongan yang membatasi segmen-segmen ditentukan berdasarkan suatu nilai *threshold*. Jika bobot sisi bernilai lebih dari *threshold* yang ditentukan, sisi tersebut dipotong.

Berikut adalah langkah-langkah proses segmentasi citra yang digunakan.

1. Urutkan sisi-sisi pada graf berdasarkan bobot (perbedaan) secara menaik.
2. Bangun pohon merentang pada tiap segmen dengan cara menambahkan sisi yang memiliki bobot minimum dan tidak membentuk sirkuit. Gunakan algoritma Find-Union untuk melakukan langkah ini.
3. Ulangi langkah 2 hingga ditemukan sisi dengan bobot lebih besar daripada *threshold*.

D. *Tahap Aplikasi Filter Cut Out*

Pada tahap ini, dilakukan perhitungan warna rataan untuk setiap segmen. Warna rataan dihitung berdasarkan persamaan (2). Setelah itu, warna dari citra sebelumnya diganti dengan warna rataan dari segmen terkait.

E. *Tahap Penyesuaian Citra Hasil Filter Cut Out*

Pada tahap preproses, ada kemungkinan ukuran citra berubah karena ada pengubahan ukuran, yaitu jika ukuran citra terlalu besar. Pada tahap terakhir ini, ukuran disesuaikan kembali dengan ukuran awal citra masukan.

F. *Implementasi dalam Kode*

Penulis telah membuat program pembuatan filter *cut out* dalam bahasa Python. Program dibuat dengan memanfaatkan library Pillow untuk pengolahan citra serta library NumPy untuk perhitungan. Program menerima masukan *path* dari citra yang akan diberikan filter *cut out*, radius, *threshold*, serta *path* untuk menyimpan citra hasil filter *cut out*. Program menampilkan dialog citra hasil filter *cut out* dan menyimpan citra tersebut sesuai *path* yang diberikan di masukan. Selengkapnya tentang program dapat dilihat pada pranala GitHub yang terlampir di bagian Lampiran.

IV. Pengujian dan Analisis

Untuk melihat hasil dari program yang telah dibuat, dilakukan pengujian dengan masukan citra sebagai berikut.

Gambar 7. Citra Uji 1 (Sumber: *www.photopoly.net/wp-content/uploads/18042012/High-Contrast-Photos-3.jpg*)

Gambar 8. Citra Uji 2 (Sumber: *https://i.pinimg.com/564x/0e/4f/72/0e4f72288a4fa34decad052b243baa03--vibrant-colors-high.jpg*)

Gambar 9. Citra Uji 3 (Sumber: *https://upload.wikimedia.org/wikipedia/commons/d/da/Purple_flower_%284764445139%29.jpg*)

A. Hasil Pengujian dengan Threshold 200

Berikut adalah hasil pengujian program dengan *threshold* sebesar 200.

Gambar 10. Hasil Pengujian Program dengan *Threshold* sebesar 200 untuk Citra Uji 1 (Sumber: Dokumentasi Penulis)

Gambar 11. Hasil Pengujian Program dengan *Threshold* sebesar 200 untuk Citra Uji 2 (Sumber: Dokumentasi Penulis)

Gambar 12. Hasil Pengujian Program dengan *Threshold* sebesar 200 untuk Citra Uji 3 (Sumber: Dokumentasi Penulis)

B. *Hasil Pengujian dengan Threshold 300*

Berikut adalah hasil pengujian program dengan *threshold* sebesar 300.

Gambar 13. Hasil Pengujian Program dengan *Threshold* sebesar 300 untuk Citra Uji 1 (Sumber: Dokumentasi Penulis)

Gambar 14. Hasil Pengujian Program dengan *Threshold* sebesar 300 untuk Citra Uji 2 (Sumber: Dokumentasi Penulis)

Gambar 15. Hasil Pengujian Program dengan *Threshold* sebesar 300 untuk Citra Uji 3 (Sumber: Dokumentasi Penulis)

C. *Analisis*

Secara umum, berdasarkan gambar-gambar di atas, dengan membandingkan gambar dari sumber dan gambar hasil filter *cut out*, dapat terlihat bahwa filter berhasil diaplikasikan. Dapat dilihat bahwa citra yang telah diaplikasikan filter *cut* out memiliki warna yang kurang bervariasi dan lebih *flat* dibanding citra aslinya. Yang menjadi kekurangan adalah hasil filter masih terlihat kurang rapi karena ada banyak *noise*.

Pada pengujian ini, dilakukan perbandingan untuk melihat bagaimana efek nilai *threshold* pada hasil filter *cut out*. Berdasarkan pengujian di atas, dapat dilihat bahwa semakin besar *threshold*, warna yang dihasilkan oleh filter semakin *flat*. Hal ini terjadi karena nilai *threshold* menentukan besar toleransi perbedaan karakteristik tertinggi yang terdapat dalam suatu segmen. Makin besar *threshold*-nya, toleransi perbedaan pun makin besar. Akibatnya, segmen-segmen yang dibentuk semakin besar dan penempatan warna rataan dari filter *cut out* makin besar. Dampak yang terlihat secara visual adalah variasi warna semakin sedikit (warna semakin *flat*).

V. SIMPULAN DAN SARAN

Berdasarkan pemaparan dan pengujian yang telah dilakukan, penulis menyimpulkan beberapa hal sebagai berikut.

1. Algoritma Kruskal yang sejatinya merupakan algoritma untuk memecahkan persoalan pohon merentang minimum dapat dimanfaatkan untuk membuat filter *cut out* pada pengolahan citra berbasis graf.
2. Nilai *threshold* menentukan besar toleransi perbedaan karakteristik tertinggi yang terdapat dalam suatu segmen. Makin besar *threshold*-nya, toleransi perbedaan pun makin besar. Akibatnya, segmen-segmen yang dibentuk semakin besar dan penempatan warna rataan dari filter *cut out* makin besar.

Selain itu, berdasarkan pengerjaan makalah ini, penulis juga memberikan saran sebagai berikut.

1. Dari sisi konsep, untuk pengembangan selanjutnya, dapat dicoba penentuan warna rataan dengan cara lain, misalnya dengan median, konversi RGB ke *hue* dan *saturation*, atau cara lainnya.
2. Dari sisi program, untuk pengembangan selanjutnya, sebaiknya gunakan bahasa yang dapat memproses kode dengan lebih cepat seperti C++ atau Java.
3. Dari sisi algoritma, untuk pengembangan selanjutnya, sebaiknya mengaplikasikan *preprocessing* citra lebih baik dalam rangka mengurangi *noise* dari citra yang dihasilkan.

LAMPIRAN

Implementasi Algoritma pada GitHub:

https://github.com/ashnchiquita/image-cut-out-filter-using-kruskal-algorithm

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada:

1. Tuhan Yang Maha Esa,
2. orang tua penulis yang senantiasa membantu dan menyemangati penulis,
3. Dr. Ir. Rinaldi Munir, M.T. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma, dan
4. teman-teman yang sudah mendukung penulis hingga saat ini.

REFERENSI

[1] Acharya, Akruti. 2022. *Guide to Image Segmentation in Computer Vision: Best Practices*. Dilansir dari *https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/*. Diakses pada 21 Mei 2023.

[2] Algorithms for Competitive Programming. 2023. *Disjoint Set Union*. Dilansir dari *https://cp-algorithms.com/data_structures/disjoint_set_union.html*. Diakses pada 21 Mei 2023.

[3] Felzenszwalb, Pedro F., dan Huttenlocher, Daniel P. 2004. *Efficient Graph-Based Image Segmentation*. *International journal of computer vision*, 59, 167-181.

[4] Long, Xiaodong dan Sun, Jian. 2020. *Image segmentation based on the minimum spanning tree with a novel weight*. *Optik-International Journal for Light and Electron Optics*, 221.

[5] Munir, Rinaldi. 2020. *Graf (Bagian 1)*. Dilansir dari *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf*. Diakses pada 21 Mei 2023.

[6] Munir, Rinaldi. 2020. *Pohon (Bagian 1)*. Dilansir dari *https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Pohon-2020-Bag1.pdf*. Diakses pada 21 Mei 2023.

[7] Munir, Rinaldi. 2021. *Algoritma Greedy (Bagian 2)*. Dilansir dari *https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf*. Diakses pada 21 Mei 2023.

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, 22 Mei 2023

13521129 Chiquita Ahsanunnisa