EKSTRAKSI LATAR DEPAN PADA CITRA IKAN DENGAN METODE *GRABCUT* YANG DIAUTOMASI MENGGUNAKAN *SALIENCY MAP*

Skripsi

Disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Komputer

Bagus Nugraha
1313617002

PROGRAM STUDI ILMU KOMPUTER
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS NEGERI JAKARTA
2022

LEMBAR PERSETUJUAN HASIL SIDANG SKRIPSI

EKSTRAKSI LATAR DEPAN PADA CITRA IKAN DENGAN METODE GRABCUT YANG DIAUTOMASI MENGGUNAKAN SALIENCY MAP

Nama : **Bagus Nugraha**

No. Registrasi : **1313617002**

	Nama	Tanda Tangan	Tanggal
Penanggung Jawab			
Dekan	: Prof. Dr. Muktiningsih N, M.Si..	.....	.....
	NIP. 196405111989032001		
Wakil Penanggung Jawab			
Wakil Dekan I	: Dr. Esmar Budi, S.Si., MT.	.....	.....
	NIP. 197207281999031002		
Ketua	: Ir. Fariani Hermin Indiyah, MT.	.....	22-08-2022
	NIP. 196002111987032001		
Sekretaris	: Med Irzal, M.Kom.	.....	23-08-2022
	NIP. 197706152003121001		
Penguji	: Ria Arafiyah, M.Si.	.....	24-08-2022
	NIP. 197511212005012004		
Pembimbing I	: Drs. Mulyono, M.Kom.	.....	24-08-2022
	NIP. 196605171994031003		
Pembimbing II	: Muhammad Eka Suryana, M.Kom.	.....	24-08-2022
	NIP. 198512232012121002		

Dinyatakan lulus ujian skripsi tanggal: 18 Agustus 2022

LEMBAR PERNYATAAN

Saya menyatakan dengan sesungguhnya bahwa skripsi dengan judul **"Ekstraksi Latar Depan pada Citra Ikan dengan Metode GrabCut yang Diautomasi Menggunakan *Saliency Map*"** yang disusun sebagai syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Ilmu Komputer Universitas Negeri Jakarta adalah karya ilmiah saya dengan arahan dari dosen pembimbing.

Sumber informasi yang diperoleh dari penulis lain yang telah dipublikasikan dan disebutkan dalam teks skripsi ini, telah dicantumkan dalam Daftar Pustaka sesuai dengan norma, kaidah dan etika penulisan ilmiah.

Jika dikemudian hari ditemukan sebagian besar skripsi ini bukan hasil karya saya sendiri dalam bagian-bagian tertentu, saya bersedia menerima sanksi pencabutan gelar akademik yang saya sanding dan sanksi-sanksi lainnya sesuai dengan peraturan perundang-undangan yang berlaku.

Jakarta, 30 Maret 2022

Bagus Nugraha

1313617002

HALAMAN PERSEMBAHAN

Untuk Ibu, Bapak,
dan Adik-adikku tercinta.

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena dengan rahmat dan karunia-Nya, penulis dapat menyelesaikan skripsi yang berjudul **"Ekstraksi Latar Depan pada Citra Ikan dengan Metode GrabCut yang Diautomasi Menggunakan *Saliency Map*"**.

Keberhasilan dalam menyusun skripsi ini tidak lepas dari bantuan berbagai pihak yang mana dengan tulus dan ikhlas memberikan masukan guna sempurnanya skripsi ini. Oleh karena itu dalam kesempatan ini, dengan kerendahan hati penulis mengucapkan banyak terima kasih kepada:

1. Yth. Para petinggi di lingkungan FMIPA Universitas Negeri Jakarta.
2. Yth. Ibu Ir. Fariani Hermin Indiyah, M.T selaku Koordinator Program Studi Ilmu Komputer.
3. Yth. Bapak Drs. Mulyono, M.Kom selaku Dosen Pembimbing I yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
4. Yth. Bapak Muhammad Eka Suryana, M.Kom selaku Dosen Pembimbing II yang telah membimbing, mengarahkan, serta memberikan saran dan koreksi terhadap skripsi ini.
5. Yth. Seluruh Dosen Pengajar dan Staff di Prodi Ilmu Komputer Universitas Negeri Jakarta.
6. Ibu dan Bapak yang selama ini telah sabar membimbing, mengarahkan, dan mendoakan penulis tanpa kenal lelah untuk selama-lamanya.
7. Aufi Rayesa dan Hafizun Alim sebagai rekan belajar yang telah membantu dan mendukung penulis dalam menyusun tugas akhir ini.

8. Teman-teman Program Studi Ilmu Komputer 2017 yang telah membantu dan mendukung, tugas akhir ini dapat diselesaikan.
9. Dan semua pihak yang juga telah membantu dengan tidak mengurangi rasa hormat penulis yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa penyusunan tugas akhir ini masih jauh dari sempurna karena keterbatasan ilmu dan pengalaman yang dimiliki. Oleh karenanya, kritik dan saran yang bersifat membangun akan penulis terima dengan senang hati. Akhir kata, penulis berharap tugas akhir ini bermanfaat bagi semua pihak khususnya penulis sendiri. Semoga Allah SWT senantiasa membalas kebaikan semua pihak yang telah membantu penulis dalam menyelesaikan tugas akhir ini.

Jakarta, 30 Maret 2022

Bagus Nugraha

ABSTRAK

BAGUS NUGRAHA. Ekstraksi Latar Depan pada Citra Ikan dengan Metode GrabCut yang Diautomasi Menggunakan *Saliency Map*. Skripsi. Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Negeri Jakarta. 2022. Dibawah bimbingan Drs. Mulyono, M.Kom dan Muhammad Eka Suryana, M.Kom

Indonesia adalah negara kepulauan terbesar dengan keanekaragaman hayati yang tinggi. Diperkirakan 8500 spesies ikan hidup di perairan Indonesia, proses identifikasi jenis ikan masih dilakukan secara manual sehingga timbul masalah pada akurasi, waktu pengenalan dan hasil pengenalan yang subjektif. Tujuan penelitian ini adalah pemisahan latar depan dari latar belakang pada citra ikan agar identifikasi jenis ikan secara otomatis lebih efisien dan tinggi tingkat akurasinya. Dalam penelitian ini dilakukan ekstraksi latar depan pada citra ikan dengan metode GrabCut yang telah diautomasi. Tahap pertama dilakukan proses konversi ruang warna Lab. Tahap kedua dilakukan proses segmentasi dengan metode *simple linear iterative clustering* (SLIC). Tahap ketiga dilakukan proses deteksi *saliency*. Tahap keempat dilakukan ekstraksi latar depan dengan metode GrabCut. Kemudian dilakukan perhitungan nilai *image similarity* dari citra hasil ekstraksi metode ini. Rata-rata nilai *image similarity* yang dihasilkan dari metode ini adalah 1.13 x 10^4 jika citra uji hanya terdapat satu objek dan 20.34 x 10^4 jika citra uji terdapat lebih dari satu objek. Metode GrabCut yang diautomasi dengan *saliency map* sangat baik dalam proses pemisahan latar depan dari latar belakang pada citra uji yang hanya terdapat satu objek dan tidak disarankan jika citra uji terdapat lebih dari satu objek.

Kata kunci : Ekstraksi Latar Depan, Citra Ikan, GrabCut, Deteksi *Saliency*, *Image Similarity*.

ABSTRACK

BAGUS NUGRAHA. *Foreground Extraction of Fish Image with Automated GrabCut Method Using Saliency Map. Thesis. Faculty of Mathematics and Science, State University of Jakarta . 2022. Under supervised by* Drs. Mulyono, M.Kom *and* Muhammad Eka Suryana, M.Kom

Indonesia is the largest archipelagic country with high biodiversity. It is estimated that 8500 species of fish live in Indonesian waters, the identification process of fish species is still done manually so that problems arise in accuracy, recognition time and subjective recognition results. The purpose of this study is to separate the foreground from the background in the fish image so that automatic identification of fish species is more efficient and has a high level of accuracy. In this study, foreground extraction of fish images was carried out using the automated GrabCut method. The first stage is the Lab color space conversion process. The second stage is the segmentation process using the simple linear iterative clustering (SLIC) method. The third stage is the saliency detection process. The fourth stage is foreground extraction using the GrabCut method. Then the image similarity value is calculated from the image extracted by this method. The average image similarity value generated from this method is 1.13 x 10^4 *if the test image contains only one object and 20.34 x* 10^4 *if the test image contains more than one object. The GrabCut method which is automated with a saliency map is very good in the process of separating the foreground from the background on a test image that contains only one object and is not recommended if the test image contains more than one object.*

Keywords : *Foreground Extraction, Fish Image, GrabCut, Saliency Detection, Image Similarity.*

DAFTAR ISI

LEMBAR PENGESAHAN **ii**

LEMBAR PERNYATAAN **iii**

HALAMAN PERSEMBAHAN **iv**

KATA PENGANTAR **v**

ABSTRAK **vii**

ABSTRACK **viii**

DAFTAR ISI **xi**

DAFTAR TABEL **xii**

DAFTAR GAMBAR **xiv**

DAFTAR LAMPIRAN **xiv**

I PENDAHULUAN **1**

A. Latar Belakang 1
B. Rumusan Masalah 5
C. Batasan Masalah 5
D. Tujuan Penelitian 6
E. Manfaat Penelitian 6

II KAJIAN PUSTAKA **8**

A. Pengertian Citra Digital 8
B. Pembentukan Citra Digital 9
1. Akuisisi Citra 9
2. *Sampling* 11
3. Kuantisasi 11
C. Pengolahan Citra Digital 11
D. Jenis Citra Digital 12
1. Citra Berwarna 12

2. Citra Berskala Keabuan (*Grayscale Image*) 13
3. Citra Biner 13
E. Histogram Citra 14
F. Segmentasi Citra 15
G. Ekstraksi Latar Depan 16
H. *Superpixel* 16
I. *Simple Linear Iterative Clustering* (SLIC) 17
J. *Superpixel-based Saliency Detection* 19
K. GrabCut 23
L. *Closed-Form Matting* 26

III METODOLOGI PENELITIAN **34**
A. Sumber Data 34
B. Deskripsi Sistem 35
C. Perancangan Sistem 36
1. Proses Input Citra 39
2. Data Keluaran 39
3. Proses SLIC 40
4. Proses Deteksi *Saliency* 41
5. Proses GrabCut 42
6. Proses *Closed-Form Matting* 44
7. *Image Similarity* 44
D. Skenario Pengujian 46
1. Ekstraksi Latar Depan Dengan GrabCut Menggunakan *Saliency Map* 47
2. Ekstraksi Latar Depan Dengan *Closed-Form Matting* 47
3. Ekstraksi Latar Depan Dengan *Background Removal AI* 47
4. Menghitung Nilai *Image Similarity* 48

IV HASIL DAN PEMBAHASAN **49**
A. Pengujian Data Masukan 49
1. Konversi Citra ke Ruang Warna Lab 50
2. Proses Segmentasi Citra Menggunakan SLIC 51
3. Proses Deteksi *Saliency* 54
4. Proses Ekstraksi Latar Depan dengan GrabCut 56
B. Pengujian data masukan dengan metode pembanding 66
C. *Image Similarity* 71

V PENUTUP **77**
A. Kesimpulan 77
B. Saran 78

DAFTAR PUSTAKA **81**

LAMPIRAN **82**

DAFTAR TABEL

Tabel 4.1 Kriteria data citra masukan 49
Tabel 4.2 Hasil pengubahan ruang warna RGB ke ruang warna Lab pada citra masukan 50
Tabel 4.3 Hasil segmentasi citra Lab menggunakan metode *Simple Linear Iterative Clustering* (SLIC) 52
Tabel 4.4 Hasil proses deteksi *saliency* 54
Tabel 4.5 Inisialisasi persegi dalam metode GrabCut 57
Tabel 4.6 Pengujian metode GrabCut yang diautomasi pada citra dengan satu objek 58
Tabel 4.7 Pengujian metode GrabCut yang diautomasi pada citra dengan objek lebih dari satu 59
Tabel 4.8 Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi dengan nilai *compactness* SLIC yang berbeda 60
Tabel 4.9 Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi pada citra dengan satu objek 62
Tabel 4.10 Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi pada citra dengan objek lebih dari satu . . . 64
Tabel 4.11 Kriteria citra *scribbles* 67
Tabel 4.12 Hasil ekstraksi latar depan menggunakan metode Closed-Form Matting 68
Tabel 4.13 Kriteria citra referensi 71
Tabel 4.14 Perbandingan hasil ekstraksi latar depan antara metode GrabCut yang diautomasi, Closed-Form Matting dan remove.bg 72
Tabel 4.15 Perbandingan nilai *image similarity* 74

DAFTAR GAMBAR

Gambar 2.1 Sistem koordinat pada citra digital 9
Gambar 2.2 Ilustrasi proses akuisisi citra 10
Gambar 2.3 (a) Citra kontinu diproyeksikan ke sensor *array* dan (b) hasil *sampling* dan kuantisasi. 10
Gambar 2.4 Citra berwarna 12
Gambar 2.5 Citra berskala keabuan 13
Gambar 2.6 Citra biner 14
Gambar 2.7 Contoh histogram citra 15
Gambar 2.8 Batas wilayah pencarian SLIC 17
Gambar 2.9 Hasil deteksi *saliency* berbasis *superpixel* 20
Gambar 2.10 Ilustrasi GrabCut dan Graph Cut 24
Gambar 2.11 (a) Lokal *window* yang dipilih dari citra dan (b) plot RGB dari distribusi warnanya. 28
Gambar 2.12 Ilustrasi *Closed-Form Matting* 32

Gambar 3.1 Citra ikan "Abudefduf" dan "Chaetodon" 34
Gambar 3.2 Citra ikan dengan lebih dari satu objek dan tampak depan . . 35
Gambar 3.3 Citra *scribbles* 36
Gambar 3.4 Diagram alir metode GrabCut menggunakan *Saliency Map* . 37
Gambar 3.5 Diagram alir perbandingan metode 38
Gambar 3.6 Diagram alir algoritma SLIC 40
Gambar 3.7 Diagram alir algoritma *Saliency Map* 42
Gambar 3.8 Diagram alir GrabCut 43
Gambar 3.9 Diagram alir *Closed-Form Matting* 45

Gambar 4.1 Potongan sampel kode pengubahan ruang warna RGB ke ruang warna Lab 51
Gambar 4.2 Hasil segmentasi SLIC dengan nilai *compactness* (a) 0.01, (b) 0.1, (c) 1, (d) 10 dan (e) 100 53
Gambar 4.3 Potongan sampel kode segmentasi citra Lab menggunakan metode *Simple Linear Iterative Clustering* (SLIC) 53
Gambar 4.4 Hasil *saliency map* dengan nilai *compactness* (a) 0.01, (b) 0.1, (c) 1, (d) 10 dan (e) 100 55

Gambar 4.5 Hasil *saliency map* dengan mengatur nilai *compactness* sebesar 10 pada tahap segmentasi SLIC 55
Gambar 4.6 Potongan sampel kode proses deteksi *saliency* 56
Gambar 4.7 Ilustrasi tahap transisi yang terdapat diantara tahap deteksi *saliency* dan tahap ekstraksi latar depan dengan GrabCut . . . 65
Gambar 4.8 Potongan sampel kode ekstraksi latar depan menggunakan GrabCut 66
Gambar 4.9 Potongan sampel kode Closed-Form Matting 70
Gambar 4.10 Potongan sampel kode *image similarity* 76

DAFTAR LAMPIRAN

Lampiran 1 Daftar Citra Masukan 82
Lampiran 2 Daftar Citra *Scribbles* 86
Lampiran 3 Daftar Citra Hasil Ekstraksi GrabCut yang Diautomasi 90
Lampiran 4 Daftar Citra Hasil Ekstraksi Metode Closed-Form Matting 94
Lampiran 5 Daftar Citra Hasil Ekstraksi Menggunakan Website remove.bg . . . 98
Lampiran 6 Sampel Kode *Superpixel-based Saliency Detection* 102
Lampiran 7 Sampel Kode GrabCut 105

BAB I

PENDAHULUAN

A. Latar Belakang

Republik Indonesia merupakan negara kepulauan terbesar di dunia dan sangat kaya akan keanekaragaman hayati. Secara geografis, Indonesia terletak di wilayah garis khatulistiwa dan beriklim tropis. Iklim tropis membuat populasi kehidupan akuatik lebih beragam daripada di negara-negara dengan iklim subtropis yang dingin. Menurut informasi yang diunggah di situs web Direktorat Jenderal Pengelolaan Ruang Laut, luas wilayah perairan Indonesia dari Sabang hingga Merauke adalah 5,8 juta kilometer persegi. Indonesia dikenal sebagai negara maritim karena wilayah perairannya lebih luas dari wilayah daratannya. Perairan Indonesia sangat luas, dengan perkiraan 8.500 spesies ikan hidup di perairan Indonesia atau 45 persen dari total dunia, menurut Kementerian Kelautan dan Perikanan Indonesia. Setiap jenis ikan memiliki bentuk dan ukuran yang berbeda untuk setiap jenis ikan. Karena perbedaan jenis dan karakteristik ikan yang begitu besar, sulit bagi kita untuk mengenali keanekaragaman ikan tersebut tanpa pengetahuan tentang perikanan (Ramadhani dan Murti, 2018).

Pada penelitian yang dilakukan oleh (Saputra dkk., 2019), jenis-jenis ikan yang terdapat pada Sungai Dong Sandar dan Sungai Rempangi di Kecamatan Sungai Laur Kabupaten Ketapang Kalimantan Barat terdiri dari 50 spesies dalam 18 famili. Kesamaan jenis ikan dan terbatasnya jumlah ahli ikan di Indonesia membuat proses identifikasi jenis ikan di Indonesia semakin sulit (Prasmatio dkk., 2020). Proses identifikasi dilakukan secara manual dengan memeriksa ciri-ciri ikan dan memerlukan pengamatan mata manusia untuk dapat mengidentifikasi jenis ikan tersebut. Keahlian mengidentifikasi jenis ikan cuma bisa dipunyai oleh sebagian kecil orang yang mempunyai pengetahuan di bidang terpaut seperti biolgi kelautan serta perikanan (Kumaseh dkk., 2013). Proses identifikasi jenis ikan masih bersifat manual dan membutuhkan tebakan manusia untuk mengidentifikasi jenis ikan

sehingga menimbulkan masalah pada akurasi, waktu pengenalan dan hasil pengenalan yang subjektif. Teknologi visi komputer (*computer vision*) adalah salah satu teknologi yang dapat membantu manusia untuk mengidentifikasi berbagai jenis ikan dengan mudah (Pratama dkk., 2020). Dengan berkembangnya ilmu komputer di bidang visi komputer, manusia kini dapat lebih akurat mengenali berbagai jenis ikan air tawar melalui citra digital (Fauzi dkk., 2019). Oleh karena itu, identifikasi jenis ikan secara komputerisasi dan otomatis semakin dibutuhkan. Visi komputer adalah seperangkat teknologi yang memungkinkan perangkat lunak untuk menganalisis dan memproses citra.

Pengolahan citra adalah bagian dari visi komputer. Pengolahan citra banyak digunakan di berbagai bidang seperti bidang pertanian, perikanan dan kedokteran. Dalam pengolahan citra, citra diproses sedemikian rupa sehingga dapat digunakan untuk aplikasi lain (Muwardi dan Fadlil, 2018). Identifikasi jenis ikan dapat dimulai dengan pengolahan citra yaitu segmentasi pada citra digital. Keuntungan melakukan segmentasi pada citra digital adalah mempermudah proses identifikasi dan kelanjutannya dapat digunakan dalam klasifikasi (Fatimah dkk., 2017). Deteksi objek dan segmentasi dapat meningkatkan akurasi pengenalan citra (Hua dkk., 2021). Secara umum, segmentasi citra adalah langkah pertama dari pengolahan citra, menyisakan informasi penting dari citra yang disebut latar depan dan membuang bagian tidak berguna yang disebut sebagai latar belakang (Deng, 2018). Citra ikan harus melalui proses pemisahan antara objek ikan dengan latar belakang citra, bertujuan untuk memperjelas karakteristik ikan di dalam citra tersebut. Mengidentifikasi jenis ikan dibutuhkan proses pengenalan ikan. Pengenalan ikan adalah suatu cara untuk mengidentifikasi ikan berdasarkan ciri-ciri tertentu, seperti gambaran bentuk, pola tubuh, warna atau ciri-ciri lainnya (Darmanto, 2019). Untuk dapat mengidentifikasi jenis ikan, ciri bentuk merupakan fitur yang mendasar untuk dilakukannya proses segmentasi citra.

Ekstraksi latar depan dalam segmentasi citra telah menjadi isu penting selama beberapa dekade terakhir. Ekstraksi latar depan berperan penting dalam aplikasi visi komputer, seperti pengeditan citra, peningkatan citra (*image enhancement*), klasifikasi citra, pengartian citra dan video (*image and video understanding*) dan sistem pengawasan (*surveillance systems*) (Kitov dan Ponomareva, 2021). Dalam jurnal (Feng dan He, 2019) telah dijelaskan bahwa tujuan dari ekstraksi latar depan adalah untuk membagi citra menjadi dua bagian yaitu latar depan dan latar belakang yang memainkan peranan penting dalam analisis dan pengenalan citra. Dalam jurnal lainnya, ekstraksi latar depan didefinisikan sebagai tugas untuk mengidentifikasi dan mengekstraksi objek berbasis fitur yang berbeda dari gambar dunia nyata dan implementasinya meliputi pengenalan objek, pelacakan visual, perbaikan citra, pencitraan medis, klasifikasi dan lainnya (Li dkk., 2018). Ada berbagai macam metode ekstraksi latar depan pada citra digital seperti, GrabCut dan *image matting*.

Teknik GrabCut yang dikembangkan oleh (Rother dkk., 2004), dianggap sebagai salah satu teknik segmentasi semi otomatis paling canggih untuk segmentasi citra. Teknik tersebut adalah teknik yang sangat kuat untuk segmentasi citra. Disinilah citra dapat disegmentasikan menjadi wilayah latar belakang dan latar depan. GrabCut merupakan metode segmentasi objek yang memerlukan interaksi pengguna untuk menginisialisasi proses segmentasi yaitu membuat persegi panjang di sekitar objek yang diinginkan pada citra untuk mengekstraknya.

Image matting adalah proses mengekstraksi objek dari latar belakang dalam satu citra dan menggabungkan objek tersebut dengan latar belakang yang berbeda, yang merupakan pekerjaan penting dalam pemrosesan citra digital dan pengeditan video (Wang dkk., 2020). Salah satu metode *image matting* adalah *Closed-Form Matting*.

Ditemukan beberapa penelitian sebelumnya tentang segmentasi citra digital dengan ikan sebagai objeknya. Penelitian (Noviyanti dkk., 2019) melakukan segmentasi citra digital pada citra dengan objek ikan arwana *super red* berbasis

deteksi tepi menggunakan algoritma *canny*. Dalam penelitian (Fadllullah dkk., 2016) melakukan segmentasi pada citra ikan tuna dengan memakai metode *Gradient-Barrier Watershed* berdasarkan analisis hierarki klaster dan *Regional Credibility Merging*. Hasil eksperimen pada penelitian tersebut dengan menggunakan 25 citra ikan tuna menunjukkan metode tersebut berhasil menyelesaikan segmentasi dengan nilai rata-rata relatif kesalahan pada bagian latar depan 4,77%, kesalahan klasifikasi 0,63% dan waktu penyelesaian 11,61 detik.

Penelitian tentang segmentasi citra digital pada objek lainnya juga sudah dilakukan sebelumnya, seperti pada penelitian (Na dkk., 2014), melakukan segmentasi otomatis label produk pada botol dengan metode GrabCut. Penelitian tersebut mengemukakan metode untuk membangun *trimap* awal yang akurat untuk metode GrabCut tanpa memerlukan interaksi manusia. *Trimap* awal tersebut dibangun dengan menerapkan *saliency map* untuk menemukan area yang menonjol (*region of interest*) dari citra. Hasil eksperimen untuk 130 citra dengan objek botol *wine* menunjukkan bahwa metode yang dikemukakan berhasil melakukan segmentasi label produk pada botol dengan akurasi rata-rata 92,31%.

Dalan jurnal (Kim dkk., 2014) yang berjudul "*An Improved GrabCut Using a Saliency Map*" menggunakan visual *saliency* berbasis *superpixel* dalam mendeteksi wilayah target untuk inisialisasi GrabCut yang efektif dan mendapatkan hasil penelitian yaitu tingkat akurasi rata-rata dalam mengekstraksi latar depan pada citra digital dengan menggunakan GrabCut konvensional mencapai 82,9% sedangkan dengan menggunakan GrabCut yang ditingkatkan menggunakan *saliency* dapat mencapai 96,6%.

Dalam penelitian (Ye dkk., 2018) melakukan deteksi kualitas penampilan mentimun pada citra dengan latar belakang yang kompleks. Penelitiannya menggunakan metode GrabCut yang telah ditingkatkan dengan deteksi *saliency*. Deteksi *saliency* dalam penelitian tersebut digunakan untuk mendeteksi posisi target latar depan dalam citra dan kemudian dijadikan batas-batas *rectangle box* dalam algoritma GrabCut. GrabCut itu sendiri bertugas untuk mengekstrak batas garis

mentimun atau mengekstrak objek mentimun dari citra yang diberikan. Metode tersebut secara signifikan meningkatkan akurasi pengenalan dan efisiensi algoritma dan disebutkan bahwa algoritma GrabCut yang ditingkatkan menggunakan saliency lebih optimal dari algoritma GrabCut konvensional. Dengan eksperimen segmentasi menggunakan kurang lebih 300 citra asli mentimun, hasilnya menunjukkan bahwa metode GrabCut yang ditingkatkan dapat secara efektif mengekstrak objek mentimun pada citra dengan dengan tingkat ketepatan 93,88%, tingkat keberhasilan 99,35% dan tingkat kesalahan 5,84%.

Dari permasalahan tersebut diusulkan sebuah metode yaitu ekstraksi latar depan pada citra ikan dengan metode GrabCut yang diautomasi menggunakan *saliency map*. *Saliency map* disini bertujuan untuk menggantikan interaksi pengguna pada algoritma GrabCut yaitu membuat persegi panjang di sekitar objek yang diinginkan sebagai inisialisasi proses segmentasi. Penelitian ini berfokus pada ekstraksi latar depan yang dihasilkan dari segmentasi citra menggunakan metode GrabCut. Diharapkan hasil dari segmentasi citra ikan akan digunakan untuk identifikasi jenis ikan yang sesuai dan hasil dari segmentasi citra ikan membuat identifikasi jenis ikan menjadi efisien dan tinggi tingkat akurasinya.

B. Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka rumusan masalah penelitian ini adalah “Bagaimana membangun suatu sistem yang dapat melakukan ekstraksi latar depan pada citra ikan dengan metode GrabCut yang diautomasi menggunakan *saliency map*?”.

C. Batasan Masalah

Penelitian ini memiliki beberapa batasan agar tidak menyimpang dari permasalahan di atas. Adapun batasan masalah dalam penelitian ini adalah:

1. Hanya melakukan pemisahan latar depan dari latar belakang pada citra ikan.

2. Citra yang digunakan dalam penelitian ini adalah citra dengan ikan tampak samping, ikan tampak depan dan objek lebih dari satu ikan.
3. Bagian objek setidaknya harus berada di tengah citra.
4. Jenis citra yang digunakan adalah citra digital dengan ruang warna RGB.
5. Ukuran citra digital yang digunakan berukuran 420×320 piksel.
6. Bahasa pemrograman yang dipakai dalam penelitian ini adalah Python 3.

D. Tujuan Penelitian

Tujuan dari penelitian ini adalah membangun sistem deteksi objek pada citra ikan dengan metode GrabCut yang telah diautomasi menggunakan *saliency map*.

E. Manfaat Penelitian

Adapun manfaat yang diharapkan penulis dalam penelitian ini adalah sebagai berikut:

1. Bagi Penulis
   - Menambah pengetahuan dan pengalaman langsung tentang cara mengekstraksi latar depan pada citra ikan dengan metode GrabCut yang diautomasi menggunakan *saliency map*.
   - Untuk memenuhi salah satu syarat kelulusan strata satu (S1) Program Studi Ilmu Komputer Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Negeri Jakarta.
2. Bagi Program Studi Ilmu Komputer
   - Sebagai referensi pada penelitian-penelitian selanjutnya yang berhubungan dengan pengolahan citra digital lebih tepatnya segmentasi citra digital.

- Dapat digunakan sebagai bahan pertimbangan dalam menentukan metode ekstraksi latar depan yang tepat untuk memisahkan objek dari latar belakang pada citra digital.
- Sebagai basis pengetahuan untuk seluruh mahasiswa khususnya mahasiswa program studi Ilmu Komputer Universitas Negeri Jakarta tentang penggunaan metode GrabCut yang diautomasi dengan *saliency map* dalam memisahkan latar depan dari latar belakang pada citra ikan.

BAB II

KAJIAN PUSTAKA

A. Pengertian Citra Digital

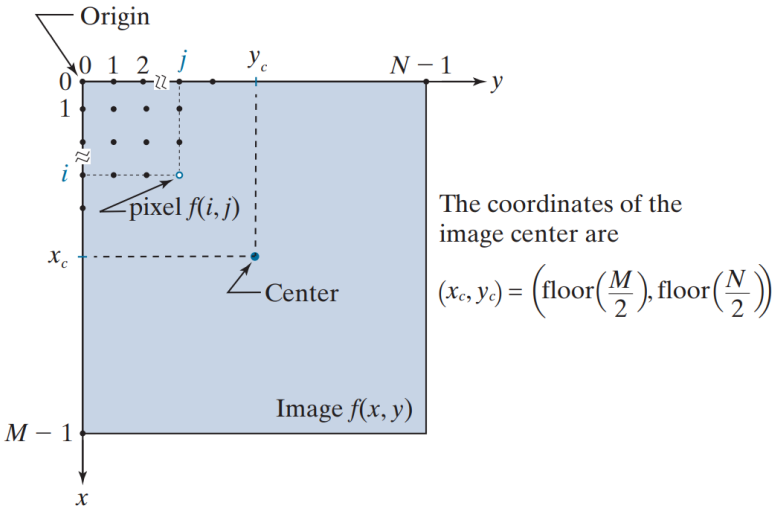
Sebuah citra dapat didefinisikan sebagai fungsi dua dimensi $f(x, y)$, dengan x dan y adalah koordinat spasial (*plane*) dan amplitudo f pada setiap pasangan koordinat (x, y) disebut intensitas atau tingkat keabuan (*gray level*) citra pada koordinat tersebut. Jika x, y dan nilai intensitas f seluruhnya berhingga (*finite*) dan bernilai diskrit, maka citra tersebut adalah citra digital (Gonzalez dan Woods, 2018).

Didefinisikan fungsi $f(s, t)$ sebagai fungsi citra kontinu dari dua variabel kontinu s dan t. Fungsi tersebut diubah menjadi citra digital melalui proses pengambilan sampel (*sampling*) dan kuantisasi. Dengan proses tersebut dapat mengubah sampel citra kontinu menjadi citra digital $f(x, y)$, yang berukuran M baris dan N kolom, dimana (x, y) adalah koordinat diskrit. Untuk notasi koordinat diskrit $x = 0, 1, 2, \ldots, M - 1$ dan $y = 0, 1, 2, \ldots, N - 1$. Citra digital dapat ditulis dalam bentuk matriks (*array*) yang terdiri dari nilai numerik $f(x, y)$, ini merupakan representasi yang digunakan untuk pemrosesan komputer (Gonzalez dan Woods, 2018). Dalam bentuk persamaan, representasi dari *array* numerik $M \times N$ dapat ditulis sebagai.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \ldots & f(0, N-1) \\ f(1,0) & f(1,1) & \ldots & f(1, N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \ldots & f(M-1, N-1) \end{bmatrix} \tag{2.1}$$

Sisi kanan persamaan di atas adalah citra digital yang direpresentasikan sebagai *array* bilangan riil. Citra digital terdiri dari sejumlah elemen yang berhingga (*finite*), yang masing-masing memiliki lokasi dan nilai tertentu. Elemen-elemen ini

disebut sebagai *picture elements*, *image elements*, *pels* dan *pixels*. Piksel adalah istilah yang paling banyak digunakan untuk menunjukkan elemen citra digital (Gonzalez dan Woods, 2018). Gambar 2.1 menunjukkan sistem koordinat yang digunakan untuk merepresentasikan citra digital.



Gambar 2.1: Sistem koordinat pada citra digital

Sumber : Gonzalez dan Woods (2018)

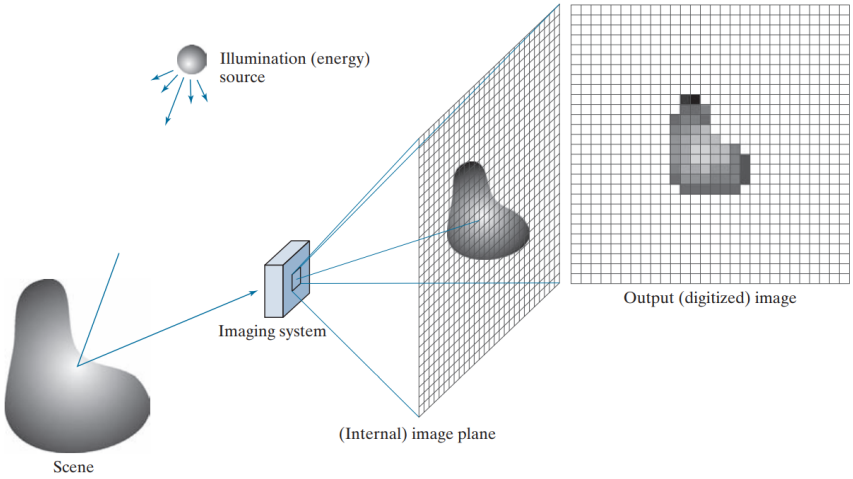
B. Pembentukan Citra Digital

Pembentukan citra digital (diskrit) melalui beberapa tahapan, yaitu akuisisi citra, *sampling* dan kuantisasi (Gonzalez dan Woods, 2018).

1. Akuisisi Citra

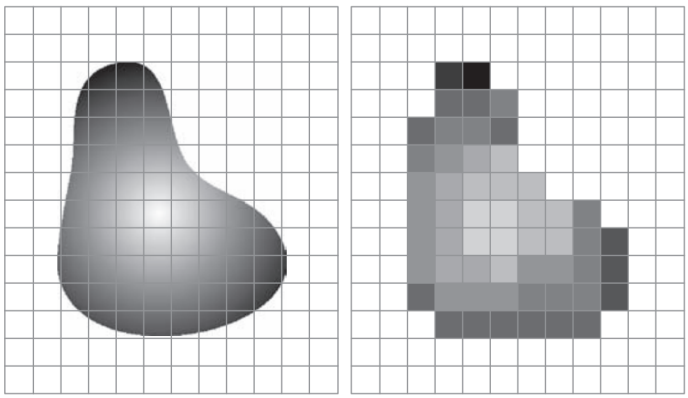
Citra analog tidak dapat dengan mudah diinterpretasikan oleh komputer dan dengan demikian representasi alternatifnya adalah dengan menggunakan citra digital. Ada beberapa jenis sensor yang digunakan untuk akuisisi citra yaitu sensor tunggal (*single sensor*), sensor garis (*sensor strip*) dan sensor larik (*sensor array*). Ada banyak cara untuk akuisisi citra, tetapi tujuan keseluruhannya sama, yaitu menghasilkan citra digital dari data yang dihasilkan sensor. Keluaran sebagian besar sensor adalah bentuk gelombang tegangan kontinu yang nilai amplitudo dan elemen

spasialnya terkait dengan fenomena fisik yang ditangkap sensor. Dapat disimpulkan bahwa akuisisi citra adalah penggunaan sensor untuk memetakan fenomena fisik ke dalam citra kontinu. Untuk membuat citra digital, data penginderaan yang kontinu harus diubah ke dalam format digital. Ini membutuhkan dua proses, pengambilan sampel dan kuantisasi (Gonzalez dan Woods, 2018).



Gambar 2.2: Ilustrasi proses akuisisi citra

Sumber : Gonzalez dan Woods (2018)



Gambar 2.3: (a) Citra kontinu diproyeksikan ke sensor *array* dan (b) hasil *sampling* dan kuantisasi.

Sumber : Gonzalez dan Woods (2018)

2. *Sampling*

Sampling adalah proses mendigitalkan nilai koordinat. Keluaran yang dihasilkan oleh sebagian besar sensor berupa citra kontinu yang dapat dinotasikan dengan fungsi kontinu $f(x, y)$. Fungsi tersebut merupakan sinyal kontinu pada nilai x, y dan juga amplitudonya (intensitas). Nilai koordinat x dan y yang kontinu akan ditransformasikan ke dalam bentuk diskrit. Proses *sampling* akan menghasilkan nilai koordinat yang diskrit pada setiap posisi di koordinat kontinu (Gonzalez dan Woods, 2018).

3. Kuantisasi

Kuantisasi adalah proses mendigitalkan nilai amplitudo (intensitas). Proses kuantisasi adalah proses pengubahan nilai amplitudo kontinu menjadi nilai amplitudo diskrit. Nilai amplitudo yang terkuantisasi adalah nilai dalam koordinat diskrit hasil proses *sampling* (Gonzalez dan Woods, 2018).

C. Pengolahan Citra Digital

Pengolahan citra digital mengacu pada pengolahan citra digital oleh komputer digital. Pengolahan citra digital adalah proses menerima masukan citra digital dan menghasilkan keluaran citra digital, meliputi beberapa proses seperti pra-pemrosesan citra, ekstraksi atribut dari citra (segmentasi) dan pengenalan objek yang terdapat pada citra (Gonzalez dan Woods, 2018).

Pengolahan citra digital dapat dibagi menjadi beberapa tingkatan yaitu rendah, menengah dan tinggi (Gonzalez dan Woods, 2018). Pengolahan tingkat rendah mencakup operasi dasar seperti pra-pemrosesan citra untuk menghilangkan *noise*, meningkatkan kontras dan ketajaman citra. Pengolahan tingkat rendah dicirikan oleh masukan dan keluarannya berupa citra. Pengolahan tingkat menengah pada citra mencakup beberapa pemrosesan seperti segmentasi (mempartisi citra ke dalam wilayah atau objek), mendeskripsikan objek dalam citra dan pengenalan objek. Pengolahan tingkat menengah dicirikan dengan masukan yang umumnya

berupa citra, sedangkann keluarannya berupa atribut yang dihasilkan dari citra tersebut (misalnya, tepi, kontur dan identitas objek). Terakhir, pengolahan citra dalam tingkat yang lebih tinggi seperti pengenalan objek yang terdapat pada citra.

D. Jenis Citra Digital

Ada tiga jenis citra digital yang umum digunakan dalam pemrosesan citra. Ketiga jenis citra tersebut yaitu citra berwarna, citra berskala keabuan dan citra biner.

1. Citra Berwarna

Citra yang direpresentasikan dalam model warna RGB terdiri dari tiga komponen yaitu *red*, *green* dan *blue*. Jumlah bit yang digunakan untuk merepresentasikan setiap piksel dalam ruang RGB disebut *pixel depth*. Meninjau dari ketiga komponen citra RGB dengan masing-masing komponen yaitu *red*, *green* dan *blue* mempunyai ukuran 8-bit. Dalam kondisi ini, setiap piksel warna RGB (yaitu nilai triplet (R,G,B)) memiliki kedalaman 24-bit (3 bidang citra dikalikan jumlah bit per bidang). Jumlah total warna yang mungkin dalam citra RGB 24-bit adalah $(2^8)^3 = 16.777.216$ (Gonzalez dan Woods, 2018).

Gambar 2.4: Citra berwarna

Sumber : Gonzalez dan Woods (2018)

2. Citra Berskala Keabuan (*Grayscale Image*)

Cahaya yang tidak berwarna disebut cahaya monokromatik (atau akromatik). Satu-satunya atribut cahaya monokromatik adalah intensitasnya. Karena intensitas cahaya monokromatik dianggap bervariasi dari hitam ke abu-abu dan akhirnya putih, istilah tingkat keabuan biasanya digunakan untuk menunjukkan intensitas monokromatik. Intensitasnya berkisar antara 0 sampai dengan 255. Kisaran nilai cahaya monokromatik dari hitam ke putih biasanya disebut skala keabuan dan citra monokromatik sering disebut sebagai citra berskala keabuan (Gonzalez dan Woods, 2018).

Gambar 2.5: Citra berskala keabuan

Sumber : Gonzalez dan Woods (2018)

3. Citra Biner

Setiap piksel dalam citra biner hanya dinyatakan dengan sebuah nilai dari dua buah nilai yaitu 0 dan 1, nilai 0 menyatakan warna hitam dan nilai 1 menyatakan warna putih. Citra biner menampilkan piksel *foreground* dengan warna putih dan piksel *background* dengan warna hitam. Biasanya citra biner dihasilkan dari proses pengolahan citra digital seperti segmentasi citra, *thresholding*, morfologi, ataupun *dithering*(Gonzalez dan Woods, 2018).

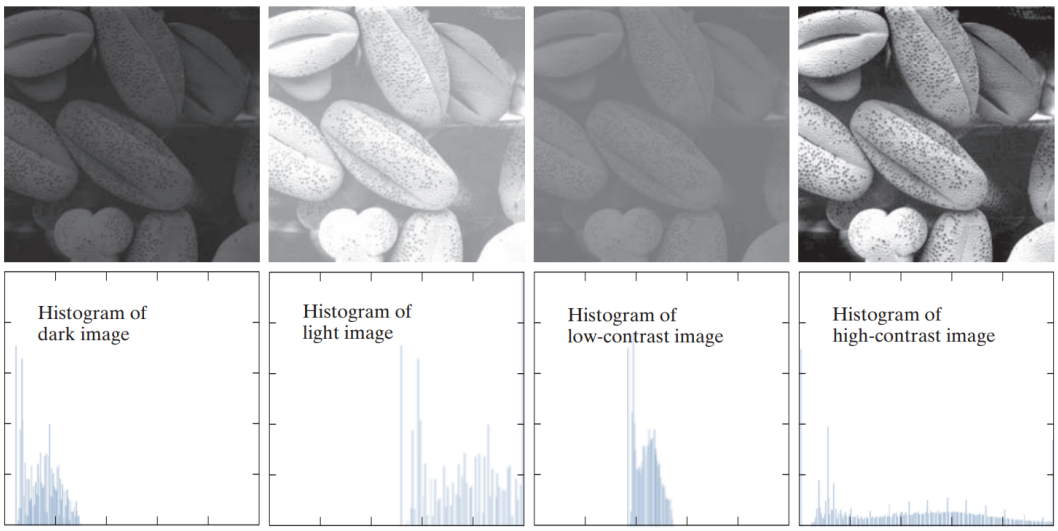
Gambar 2.6: Citra biner

Sumber : Gonzalez dan Woods (2018)

E. Histogram Citra

Seperti histogram pada umumnya, histogram citra juga menunjukkan frekuensi. Namun histogram citra menunjukkan frekuensi nilai intensitas piksel. Dalam histogram citra, sumbu x menunjukkan intensitas tingkat keabuan dan sumbu y menunjukkan frekuensi intensitas tersebut (Gonzalez dan Woods, 2018). Bentuk histogram berhubungan dengan tampilan citra. Misalnya, Gambar 2.7 menunjukkan citra dengan empat karakteristik intensitas dasar yaitu gelap, terang, kontras rendah dan kontras tinggi.

Dalam citra dengan intensitas gelap, kelas-kelas (*bins*) histogram banyak terpopulasi di ujung bawah skala intensitas. Demikian pula, kelas-kelas histogram paling padat dari citra dengan intensitas terang berada di ujung skala yang lebih tinggi. Citra dengan kontras rendah memiliki histogram sempit yang biasanya terletak di tengah skala intensitas. Dapat dilihat bahwa komponen histogram dari citra dengan kontras tinggi mencakup rentang skala intensitas yang luas, distribusi piksel tidak terlalu jauh dengan kelas tetangganya dan terdapat beberapa kelas yang jauh lebih tinggi daripada yang lain. Secara intuitif, masuk akal untuk menyimpulkan bahwa citra yang pikselnya cenderung menempati seluruh rentang tingkat intensitas dan cenderung terdistribusi secara seragam, akan memiliki tampilan kontras tinggi dan akan menunjukkan variasi abu-abu yang besar.



Gambar 2.7: Contoh histogram citra

Sumber : Gonzalez dan Woods (2018)

F. Segmentasi Citra

Segmentasi citra adalah proses membagi citra menjadi beberapa segmen untuk membuat tampilan citra lebih bermakna dan lebih mudah untuk dianalisis. Segmentasi citra adalah proses membagi citra digital menjadi beberapa segmen. Artinya, membagi piksel di area yang sama menurut beberapa kriteria homogenitas, seperti kumpulan piksel, warna, intensitas dan tekstur, sehingga dapat ditemukan dan diidentifikasi batas-batas objeknya itu dalam citra (R.Dass dkk., 2012).

Dalam praktiknya, segmentasi citra digunakan untuk *filtering* citra yang *noisy*, aplikasi medis (menemukan tumor dan patologi lainnya, mengukur volume jaringan, bedah dengan panduan komputer, diagnosis, perencanaan perawatan, studi struktur anatomi), menemukan objek dalam citra satelit (jalan, hutan dan sebagainya), pengenalan wajah, pengenalan sidik jari dan lain-lain (R.Dass dkk., 2012). Berdasarkan teknologi yang berbeda, pendekatan segmentasi citra dibagi menjadi dua kategori yang berdasarkan dua sifat citra :

1. Deteksi diskontinuitas (*Discontinuities*)

   Deteksi diskontinuitas berarti mempartisi citra berdasarkan perubahan intensitas yang curam (*abrupt*). Ini termasuk algoritma segmentasi citra seperti deteksi tepi.

2. Deteksi kesamaan (*Similarities*)

   Deteksi kesamaan berarti mempartisi citra ke dalam wilayah yang serupa menurut seperangkat kriteria yang telah ditentukan. Ini termasuk algoritma segmentasi citra seperti *thresholding*, *region growing* dan *region splitting and merging*.

G. Ekstraksi Latar Depan

Ekstraksi latar depan didefinisikan sebagai tugas untuk mengidentifikasi dan mengekstraksi objek berbasis fitur yang berbeda dari gambar dunia nyata (Li dkk., 2018). Ekstraksi latar depan bertujuan untuk membagi citra menjadi dua bagian yaitu latar depan dan latar belakang (Feng dan He, 2019).

Teknik ekstraksi latar depan diawali dengan mendeteksi objek pada citra dan dilanjutkan dengan mengekstrak atau memperoleh objek yang terdeteksi dari citra aslinya. Dengan kemajuan penelitian di berbagai bidang ilmu pengetahuan dan teknologi, teknik ini telah menemukan pengaplikasiannya di berbagai bidang yang berbeda yaitu deteksi penyakit medis, ekstraksi objek satelit, *image retrival*, pengenalan objek dan lain-lain (Soni dkk., 2014).

H. *Superpixel*

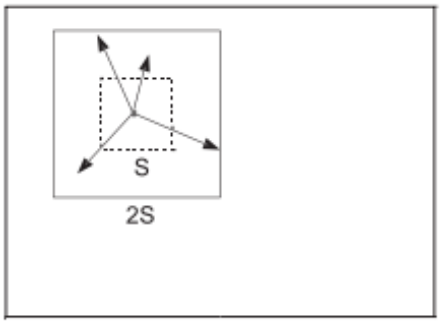
Pengelompokan setiap piksel yang serupa ke dalam grup-grup tertentu di dalam citra disebut *superpixel*. Karena persepsi pengelompokan piksel mengandung lebih banyak informasi daripada piksel individu dan menyelaraskan lebih baik ke batas objek daripada struktur berbasis *grid* lainnya, ekstraksi *superpixel* digunakan

sebagai langkah pra-pemrosesan yang terkenal di berbagai aplikasi visi komputer, seperti segmentasi citra berbasis grafik, pengenalan objek dan lain-lain (Cahskan dkk., 2015).

Superpixel digunakan untuk mengurangi ukuran citra guna mengurangi biaya pemrosesan. Karena *superpixel* adalah wilayah yang terlalu banyak tersegmentasi dan setiap *superpixel* terdiri dari piksel yang berdekatan, menggunakan perwakilan (*representative*) dari setiap *superpixel* akan mengurangi efek *noise* pada citra (Cahskan dkk., 2015).

I. *Simple Linear Iterative Clustering* (SLIC)

SLIC adalah *iterative clustering* yang sederhana dan efisien. Algoritma yang ditingkatkan berdasarkan K-means *clustering*. Algoritma ini digunakan untuk segmentasi otomatis citra berwarna dan memperoleh kecepatan dalam *clustering* yang cepat, tepi lokasi yang halus dan tepat, dan jumlah blok piksel segmentasi yang dapat dikontrol. Algoritma ini telah banyak digunakan dalam pra-pemrosesan rekayasa citra warna.



Gambar 2.8: Batas wilayah pencarian SLIC

Sumber : Achanta dkk. (2012)

Dua parameter perlu diinisialisasi dalam algoritma SLIC yaitu jumlah *superpixel* K dan koefisien *compact* m. Jumlah blok area yang terbagi dapat dikontrol dengan mengatur parameter K. Tingkat kesesuaian batas dapat diubah dengan mengubah koefisien *compact* m. Semakin besar angkanya, semakin rapi

blok pikselnya. Di awal algoritma, citra diubah menjadi ruang warna CIELab. Kemudian, menurut jumlah *superpixel* K, citra dibagi menjadi *superpixel* dengan ukuran $S = \sqrt{N/K}$ dan jumlah pusat *cluster* dapat diperoleh dari ukuran *superpixel* S. Pusat *cluster* dari setiap *superpixel* diwakili oleh lima parameter yaitu L, a, b dari ruang warna CIELab dan x, y dari informasi posisinya. Nilai spesifik blok *superpixel* yang sesuai dihitung dengan ukuran blok piksel dan jumlah pusat *cluster* per baris dan per peringkat pada citra. Semua jarak piksel $d(i)$ diatur ke ∞ dan tanda klasifikasi $I(i)$ diinisialisasi ke -1. Piksel di wilayah $2S \times 2S$ diasosiasikan dengan pusat *cluster* terdekat saat ini dan setiap jarak piksel D didefinisikan sebagai berikut.

$$D = \sqrt{{d_c}^2 + \left(\frac{d_s}{S}\right)^2 m^2} \tag{2.2}$$

$$d_c = \sqrt{(L_j - L_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2} \tag{2.3}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{2.4}$$

i dan j mewakili dua piksel, m adalah koefisien *compact*, S adalah panjang sisi yang diharapkan dari *superpixel*, d_c adalah nilai perbedaan warna dalam ruang warna CIELab dan d_s adalah jarak ruang antar piksel. Koefisien m memungkinkan pengguna mengontrol kekompakan *superpixel*. Semakin besar nilai m, *superpixel* yang dihasilkan lebih kompak atau memiliki rasio area ke perimeter yang lebih rendah. Jika nilai m kecil, *superpixel* yang dihasilkan memiliki ukuran dan bentuk yang kurang teratur. Untuk menghindari kesalahan pada pengelompokan di tepi citra, algoritma ini akan menghitung ulang lokasi pusat *cluster* dan pusat *cluster* baru harus dipindahkan ke posisi gradien terendah. Gradien citra dihitung sebagai berikut.

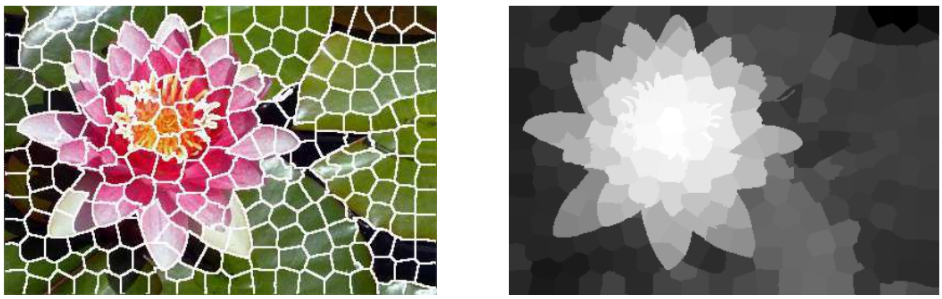
$$G(x,y) = \|I(x+1,y) - I(x-1,y)\|^2 - \|I(x,y+1) - I(x,y-1)\|^2 \quad (2.5)$$

$I(x,y)$ adalah vektor Lab yang sesuai dengan piksel pada posisi (x,y) dan $||.||$ adalah *L2 norm* yang berarti perhitungan akar kuadrat dari jumlah nilai vektor kuadrat. Setelah semua piksel diasosiasikan dengan pusat *cluster* terdekat, pusat *cluster* baru dihitung sebagai rata-rata vektor $Labxy$ dari semua piksel yang termasuk dalam *cluster*. Proses asosiasi piksel dengan pusat *cluster* terdekat dan menghitung ulang pusat *cluster* dilakukan secara berulang hingga konvergensi, tetapi dalam jurnal (Achanta dkk., 2012) telah menemukan bahwa 10 iterasi sudah cukup untuk sebagian besar citra.

J. *Superpixel-based Saliency Detection*

Visual *saliency* adalah kualitas persepsi yang membuat beberapa objek dalam citra menonjol dari daerah sekitarnya dan dengan demikian menarik perhatian visual manusia. Mekanisme tersebut memungkinkan manusia dengan mudah mengidentifikasi objek yang menonjol secara visual bahkan dalam citra yang kompleks.

Deteksi *saliency* berbasis *superpixel* adalah proses untuk mendeteksi objek penting atau *region of interest* (ROI) dalam sebuah citra digital. Dalam jurnal (Liu dkk., 2013), melakukan perbandingan model *saliency* berbasis *superpixel* dengan 6 (enam) model *saliency* yaitu *frequency tuned*, *distibution of color and orientation*, *context-aware*, *region contrast*, *kernal density estimation* dan *over-segmentation*. Hasil perbandingan secara objektif yaitu model *saliency* berbasis *superpixel* mempunyai nilai presisi diatas 90% dan model *saliency* lainnya mempunyai nilai presisi dibawah 90%.



(a) Hasil segmentasi *superpixel* **(b)** *Saliency map* yang dihasilkan

Gambar 2.9: Hasil deteksi *saliency* berbasis *superpixel*

Sumber : Liu dkk. (2013)

Metode deteksi *saliency* berbasis *superpixel* memiliki beberapa keuntungan seperti ketahanan terhadap derau dalam citra, pemrosesan yang cepat dan memberikan nilai *saliency* yang sama ke piksel yang memiliki karakteristik yang serupa. Metode ini memisahkan citra yang diberikan menjadi beberapa *superpixel* dan menggunakan informasi warna dan juga informasi spasial untuk menghitung nilai *saliency* dari setiap *superpixel*. Deteksi *saliency* berbasis *superpixel* dapat digunakan untuk membuat *initial rectangle* dalam algoritma GrabCut dengan maksud menggantikan interaksi manusia yang tidak stabil atau dengan kata lain menjadikan metode GrabCut menjadi otomatis (*unsupervised*) tanpa memerlukan interaksi manusia. Deteksi *saliency* berbasis *superpixel* diaplikasikan untuk manghasilkan ROI yang dapat digunakan sebagai batas-batas *rectangle* dalam inisialisasi metode GrabCut.

Tahapan dalam deteksi *seliency* berbasis *superpixel* dibagi menjadi beberapa tahapan. Pertama, segmentasi *superpixel* dan kuantisasi warna adaptif digunakan untuk menyederhanakan (*simplify*) citra masukan. Kemudian, *inter-superpixel similarity* 2.6 dihitung berdasarkan perbedaan (*intersection*) histogram dan jarak spasialnya. Tujuan menghitung *inter-superpixel similarity* yaitu untuk menghasilkan *saliency map* tingkat *superpixel*. Dalam penelitian (Liu dkk., 2013) disebutkan bahwa untuk menyederhanakan citra masukan melalui proses segmentasi *superpixel* dan kuantisasi warna adaptif, untuk mewakili citra menggunakan pengurangan

jumlah *superpixel* dan warna representatif. Citra masukan pertama kali ditransformasikan ke dalam ruang warna Lab, dengan *luminance channel* dan dua *chrominance channel* dipisahkan.

Langkah pertama dalam proses menemukan *saliency map* yaitu dengan menggunakan metode SLIC untuk mempartisi citra asli menjadi sejumlah *superpixel*, yang biasanya memiliki bentuk yang teratur dan kompak dengan batas yang lebih baik. Parameter ukuran awal *superpixel* disetel ke $\sqrt{N/200}$. Dimana N menunjukkan jumlah piksel pada citra dan jumlah *superpixel* yang dihasilkan kurang lebih 200, yang umumnya cukup untuk menjaga batas yang berbeda dengan baik. Contoh segmentasi *superpixel* pada citra ditunjukkan pada gambar 2.9a, dimana batas-batas *superpixel* yang berbeda digambarkan menggunakan garis putih.

Untuk kuantisasi warna adaptif, setiap *channel* warna dalam citra masukan dikuantisasi terlebih dahulu menjadi histogram citra h_i dengan jumlah *bins*/interval sebanyak q. Histogram citra yang telah dibuat mempunyai ukuran $q \times q \times q$. Warna terkuantisasi dari setiap *bins* qc_k dihitung sebagai warna rata-rata piksel yang termasuk dalam *bins* tersebut. m *bins* tertinggi yang mencakup lebih dari jumlah piksel sebanyak $a \cdot N$ dijadikan warna sebagai representasi setiap segmennya. Setiap *bins* yang tersisa $q^3 - m$ digabungkan menjadi salah satu m *bins* terpilih dengan perbedaan terkecil antara warna terkuantisasi mereka dan warna terkuantisasi dari semua *bins* diperbarui untuk menghasilkan tabel kuantisasi warna Q dengan m entri. Parameter q dan a diset menjadi 16 dan 0.95.

Pada penelitian (Liu dkk., 2013), dikarenakan warna objek yang menonjol didistribusikan secara kompak di tengah citra dibandingkan dengan yang lain, *inter-superpixel similarity* atau nilai *saliency* S untuk setiap *superpixel* dihitung sebagai berikut.

$$S(i) = \frac{\sum_{j=1}^{n} \Psi(i,j) \cdot D(j)}{\sum_{j=1}^{n} \Psi(i,j)} \tag{2.6}$$

Dalam persamaan di atas, $D(j)$ adalah jarak spasial antara posisi tengah *superpixel* ke-j dengan posisi tengah citra, $\Psi\left(i,j\right)$ adalah *similarity value* yang dihitung dengan menggunakan *color similarity* $\Psi_c\left(i,j\right)$ dan *spatial similarity* $\Psi d\left(i,j\right)$ antara *superpixel* ke-i dan ke-j. Nilai kesamaan setiap pasangan *superpixel* ke-i dan *superpixel* ke-j, didefinisikan sebagai berikut.

$$\Psi\left(i,j\right) = \Psi_c\left(i,j\right).\Psi_d\left(i,j\right) \tag{2.7}$$

$$\Psi_c\left(i,j\right) = \sum_{k=1}^{m} \min\left\{H_i\left(k\right), H_j\left(k\right)\right\} \tag{2.8}$$

$$\Psi_d\left(i,j\right) = 1 - \left(\frac{D_{i,j}}{d}\right) \tag{2.9}$$

Pada persamaan 2.8, *color similarity* $\Psi_c\left(i,j\right)$ diartikan sebagai penjumlahan dari perpotongan (*intersection*) histogram warna *superpixel* ke-i dan histogram warna *superpixel* ke-j, dihitung dan disederhanakan berdasarkan *adaptive color quantization* dan dinormalisasi menjadi $\sum_{k=1}^{m} H_i\left(k\right) = 1$. Dalam persamaan *spatial similarity* $\Psi_d\left(i,j\right)$, $D_{i,j}$ adalah jarak spasial antara pusat *superpixel* ke-i dan ke-j, dan d adalah panjang diagonal dari citra (Liu dkk., 2013).

Selanjutnya yaitu menghitung nilai *saliency* setiap *superpixel* menggunakan persamaan 2.6. Dalam persamaan 2.8 yang didefinisikan sebagai jumlah perpotongan antar dua histogram. Histogram tersebut adalah histogram warna tiap *superpixel* yang telah dikuantisasi sebelumnya dan dinormalisasi menjadi ($\sum_{k=1}^{m} H_i\left(k\right) = 1$).

Setelah menghitung nilai *saliency* setiap *superpixel* dengan menggunakan persamaan 2.6 , maka dihasilkanlah *saliency map* seperti pada gambar 2.9b. Untuk mengevaluasi penerapan *saliency map* dalam mendeteksi objek yang menonjol (*region of interest*) secara lebih eksplisit maka dilakukan operasi *thresholding* adaptif pada *saliency map* menggunakan metode Otsu untuk mendapatkan *mask* citra biner dari objek yang menonjol (Liu dkk., 2013).

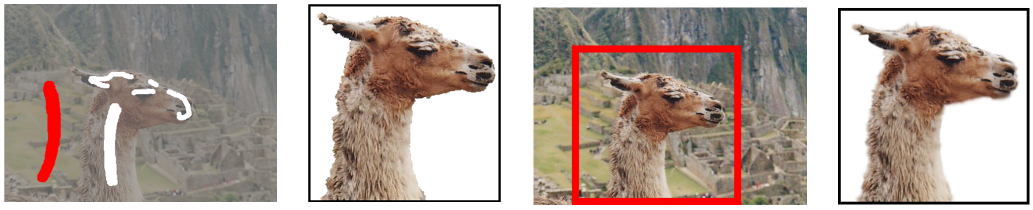
K. GrabCut

GrabCut merupakan metode segmentasi citra digital yang disempurnakan dari metode sebelumnya yaitu *Graph Cut* (Rother dkk., 2004) dimana citra dibagi menjadi dua yaitu latar depan dan latar belakang. Teknik GrabCut mengatasi masalah ekstraksi objek latar depan yang efisien dan interaktif dalam lingkungan yang kompleks dimana latar belakang sulit untuk dipisahkan. Algoritma GrabCut bertujuan untuk mencapai kinerja tinggi dalam segmentasi dengan sedikit interaksi dari pengguna. Pengguna menggambar persegi panjang pada citra. Seluruh piksel yang berada di luar persegi panjang dikelompokkan sebagai latar belakang dan seluruh piksel piksel yang berada di dalam persegi panjang dikelompokkan sebagai kemungkinan latar depan. *Gaussian Mixture Models* (GMM) membuat kelas latar depan dan latar belakang. Kelas latar belakang adalah piksel di luar persegi panjang dan piksel di dalam persegi panjang, tetapi dengan warna yang sama dengan piksel di luar persegi panjang. Piksel yang tersisa dikelompokkan di latar depan. Terdapat beberapa kelebihan metode GrabCut yaitu lebih optimal dari metode *Graph Cut*, menyederhanakan interaksi pengguna, mendapatkan hasil terbaik dan meningkatkan hasil kliping latar depan (Ngurah Sanditya Riantama dkk., 2019). Berikut adalah beberapa perbedaan antara GrabCut dengan *Graph Cut*:

1. Target (objek) dan model latar belakang pada *Graph Cut* adalah histogram keabuan sedangkan pada GrabCut digantikan oleh model GMM dengan 3 *channel* RGB.

2. Minimisasi energi (segmentasi) *Graph Cut* dicapai pada satu waktu sedangkan pada GrabCut digantikan oleh proses iteratif interaktif dari estimasi segmentasi berkelanjutan dan pembelajaran parameter model.

3. *Graph Cut* mengharuskan pengguna untuk menentukan beberapa titik dari target dan latar belakang sedangkan GrabCut hanya perlu memilih target berdasarkan persegi.



Inisialisasi Graph Cut | Hasil Graph Cut | Inisialisasi GrabCut | Hasil GrabCut

Gambar 2.10: Ilustrasi GrabCut dan Graph Cut

Sumber : Rother dkk. (2004)

Untuk sebuah algoritma segmentasi, *cost function* adalah sebuah keharusan, dinyatakan dalam formula 2.10 - 2.12, dengan k_n berarti piksel ke-n milik komponen GMM ke-k dan a_n hanya mempunyai nilai 0 dan 1 yang berarti piksel ke-n termasuk dalam latar belakang atau latar depan. θ adalah parameter dari GMM, komponen dalam θ diantaranya adalah koefisien pembobotan, nilai rata-rata dan matriks *covariance*. z_n adalah nilai piksel dari piksel ke-n. $COV(a_n, k_n)$ merujuk pada matriks *covariance* dari komponen yang dimiliki oleh latar belakang atau latar depan dan berkesinambungan dengan piksel ke-n.

$$U\left(a, k, \theta, z\right) = \sum_{n} D\left(a_n, k_n, \theta, z_n\right) \tag{2.10}$$

$$\begin{aligned} D\left(a_n, k_n, \theta, z_n\right) = & -\log\left[\pi\left(a_n, kn\right)\right] + \frac{1}{2}\log\left\{\det\left[COV\left(a_n, k_n\right)\right]\right\} \\ & + \frac{1}{2}\left[z_n - \mu\left(a_n, k_n\right)\right]^T\left[COV\left(a_n, k_n\right)\right]^{-1}\left[z_n - \mu\left(a_n, k_n\right)\right] \end{aligned} \tag{2.11}$$

$$\theta = \left\{\pi\left(a, k\right), \mu\left(a, k\right), COV\left(a, k\right), a = 0, 1, k = 1, \ldots, k\right\} \tag{2.12}$$

U menunjukkan kecocokan distribusi a terhadap z dalam bentuk penjumlahan logaritma negatif dari fungsi probabilitas. Fungsi D 2.10 - 2.11 adalah

multidimensi *gaussian probability distribution*. Maka dari itu, fungsi D berarti seberapa mungkin sebuah piksel berada di latar belakang ataupun latar depan. Jumlah *cost function* dinyatakan dalam formula 2.13 yang memuat fungsi U dan V. Fungsi V ditulis sebagai 2.14.

$$E\left(a,k,\theta,z\right)=U\left(a,k,\theta,z\right)+V\left(a,z\right) \tag{2.13}$$

$$V\left(a,z\right)=\gamma\sum_{(m,n)\epsilon c}f\left(a\right)\exp\left(-\beta\left\|z_m-z_n\right\|^2\right)\ where f\left(a\right)=\begin{cases}0, a_n=a_m\\ 1, a_n\neq a_m\end{cases} \tag{2.14}$$

$$\beta=\left(2\langle\left(z_m-z_n\right)^2\rangle\right)^{-1} \tag{2.15}$$

V adalah istilah kehalusan, menunjukkan penalti untuk diskontinuitas antara piksel ke-n dan piksel ke-m. Dengan a adalah label piksel, z adalah RGB *value* atau intensitas piksel, maka z_m berarti intensitas piksel ke-m dan z_n adalah intensitas piksel ke-n, c adalah himpunan pasangan piksel (*pairs of neighboring pixels*) yaitu piksel ke-n dan piksel ke-m, jika nilai a piksel ke-n sama dengan nilai a piksel ke-m maka nilai $f(a)$ adalah 0 dan sebaliknya maka nilai $f(a)$ adalah 1. γ dan β konstan dengan nilai γ sebesar 50 dan nilai β didapat dari persamaan 2.15. Jika dua piksel yang berdekatan punya perbedaan nilai yang besar, mereka bisa dipertimbangkan sebagai latar belakang dan latar depan dengan *cost* yang rendah. Jika nilai dua piksel yang bertetangga itu dekat, maka jumlah untuk mengalokasikannya berbeda latar belakang atau latar depan akan sangat besar. Segmentasi tersebut dijalankan dengan teori *max flow min cut* dan *cost function* 2.13 (Dung dkk., 2018). Berikut adalah algoritma dalam metode GrabCut yang dijelaskan lebih sederhana (Ngurah Sanditya Riantama dkk., 2019):

1. Pengguna membuat *trimap* terlebih dahulu dengan membuat persegi panjang di sekitar objek. Piksel di dalam persegi panjang didefinisikan sebagai tidak terdefinisi (*unknown*) dan piksel di luar persegi panjang didefinisikan sebagai latar belakang.

2. Seluruh piksel yang tidak terdefinisi ditetapkan ke kelas latar depan dan seluruh piksel yang didefinisikan sebagai latar belakang ditetapkan ke kelas latar belakang.

3. *Gaussian Mixture Model* (GMM) dibentuk untuk kelas latar depan dan latar belakang pertama.

4. Setiap piksel di kelas latar depan ditetapkan ke komponen GMM latar depan dan setiap piksel di kelas latar belakang ditetapkan ke komponen GMM latar belakang.

5. GMM yang baru dibentuk dari sekelompok piksel yang dibentuk pada GMM sebelumnya.

6. Menjalankan operasi *Graph Cut* untuk mendapatkan bagian latar depan dan latar belakang baru yang diklasifikasikan.

7. Langkah 4-6 diulangi sampai klasifikasi piksel tidak berubah.

L. *Closed-Form Matting*

Ekstraksi latar depan dan latar belakang, atau biasa disebut *matting*, masih banyak digunakan dalam aplikasi *computer vision*. Sebagian besar *matting* merupakan metode *supervised* yang biasanya membutuhkan *input user* tambahan (*constraints*), seperti *trimaps* atau *scribbles*. Namun, lebih efisien jika dilakukan tanpa campur tangan manusia. Banyak aplikasi pembuat film khususnya dalam animasi efek visual mengalami kesulitan dalam menempatkan objek pada adegan latar belakang yang sebenarnya. Teknik ekstraksi latar depan telah digunakan untuk mendapatkan objek. Oleh karena itu, dapat dikomposisikan dengan latar belakang dan diambil dalam kondisi nyata. Proses pemisahan antara latar depan dan latar belakang dapat dilakukan dengan menggunakan beberapa teknik, seperti segmentasi

citra. Namun demikian, segmentasi citra, hasilnya masih belum dapat memisahkan objek secara akurat. *Image matting* telah diusulkan sebagai solusi untuk memecahkan masalah pemisahan objek ini.

Natural image matting dan *compositing* sangat penting dalam pengeditan citra dan video. Secara formal, metode *image matting* mengambil masukan pengguna yaitu citra I, yang diasumsikan sebagai gabungan dari citra latar depan F dan citra latar belakang B. Warna piksel ke-I diasumsikan sebagai kombinasi linear dari warna latar depan dan latar belakang yang sesuai.

$$I_i = \alpha_i F_i + (1 - \alpha_i) B_i \tag{2.16}$$

α_i adalah *opacity* piksel latar depan. Dalam *natural image matting*, semua kuantitas di sisi kanan persamaan *compositing* 2.16 tidak diketahui. Pada setiap piksel, sistem persamaan linear seperti di atas dapat diperoleh, tetapi ada terlalu banyak parameter yang tidak diketahui, lebih dari jumlah persamaan. Ini merupakan kesulitan dalam *matting*, sehingga perlu ditambahkan masukan pengguna tambahan (*constraints*).

Pada jurnal (Levin dkk., 2008), berasumsi bahwa F dan B konstan di setiap *small window*. Dalam *small window* masing-masing F dan B adalah campuran linear dari dua warna. Dengan kata lain, warna Fg dan Bg di dalam *small window* terletak pada garis lurus di ruang warna RGB. F_1, F_2 dan F_i menyatakan warna, berlaku juga untuk B. β berubah untuk setiap piksel di dalam *window*.

$$F_i = \beta_i F_1 + (1 - \beta_i) F_2 \tag{2.17}$$

$$B_i = \gamma_i B_1 + (1 - \gamma_i) B_2 \tag{2.18}$$

Dalam persamaan *matting* membutuhkan beberapa asumsi tentang F, B dan/atau α. Pada jurnal (Levin dkk., 2008) mengasumsikan bahwa F dan B diperkirakan konstan pada *small window* disekitar setiap piksel. Memungkinkan

Gambar 2.11: (a) Lokal *window* yang dipilih dari citra dan
(b) plot RGB dari distribusi warnanya.
Sumber : Levin dkk. (2008)

untuk menulis ulang persamaan *compositing* 2.16, dengan menyatakan α sebagai fungsi linear dari citra I.

$$\alpha_i \approx aI_i + b, \forall i \in w \tag{2.19}$$

$$a = \frac{1}{F - B} \quad dan \quad b = -\frac{B}{F - B} \tag{2.20}$$

$\forall i \in w$ adalah untuk semua piksel di *window* w dan w adalah *small window*. Jika warna latar depan dan latar belakang di *window* memenuhi *color line model*, maka persamaan 2.19 berlaku. Untuk membuktikannya, dengan mensubstitusi *linear combination* F_i dan B_i ke dalam persamaan 2.16.

$$
\begin{aligned}
I_i &= \alpha_i F_i + (1-\alpha_i) B_i \\
&= \alpha_i (\beta_i F_1 + (1-\beta_i) F_2) + (1-\alpha_i)(\gamma_i B_1 + (1-\gamma_i) B_2) \\
&= \alpha_i \beta_i F_1 + \alpha_i F_2 - \alpha_i \beta_i F_2 + \gamma_i B_i - a_i \gamma_i \beta_1 + B_2 - \alpha_i B_2 - \gamma_i B_2 + \alpha_i \gamma_i B_2
\end{aligned} \tag{2.21}
$$

Selanjutnya, memindahkan B_2 dari ruas kanan ke ruas kiri persamaan dan dituliskan dalam bentuk perkalian matriks.

$$
\begin{aligned}
I_i - B_2 &= [F_2 - B_2, F_1 - F_2, B_1 - B_2, B_2 - B_1] \begin{bmatrix} \alpha_i \\ \alpha_i \beta_i \\ \gamma_i \\ \alpha_i \gamma_i \end{bmatrix} \\
&= [F_2 - B_2, F_1 - F_2, B_1 - B_2] \begin{bmatrix} \alpha_i \\ \alpha_i \beta_i \\ \gamma_i - \alpha_i \gamma_i \end{bmatrix}
\end{aligned} \tag{2.22}
$$

Setelah memindahkan B_2 ke ruas kiri persamaan, selanjutnya menyederhanakan persamaan yang sudah didapat sebelumnya dan dapat ditulis kembali menjadi:

$$
H \begin{bmatrix} \alpha_i \\ \alpha_i \beta_i \\ \gamma_i - \alpha_i \gamma_i \end{bmatrix} = I_i - B_2 \tag{2.23}
$$

I_i dan B_2 adalah vektor 3×1, mewakili 3 *color channel*. a_1, a_2, a_3 sebagai komponen pada baris pertama dalam H^{-1} dan b sebagai hasil perkalian skalar dari baris pertama H^{-1} dengan vektor B_2, kemudian didapatkanlah persamaan 2.19.

Relasi pada persamaan 2.19 memberikan petunjuk untuk menemukan α, a dan b yang meminimalkan *cost function*. *Cost function* dapat ditulis sebagai berikut.

$$J(\alpha, a, b) = \sum_{j \in i} \left(\sum_{i \in w_j} (\alpha_i - a_j - b_j)^2 + \epsilon a_j^2 \right) \tag{2.24}$$

w_j adalah *small window* di sekitar piksel j. Dalam implementasinya ukuran *small window* hanya menggunakan ukuran sebesar 3x3 piksel. ϵ adalah konstanta kecil untuk istilah regularisasi pada a_j, salah satu alasan untuk menambahkan regularisasi ini adalah stabilitas numerik. *Cost function* adalah kuadrat dalam α, a dan b, dengan $3N$ *unknowns* untuk citra dengan ukuran N piksel. Untungnya, dalam implementasinya a dan b dapat dieliminasi dari persamaan 2.24, menjadi *quadratic cost* hanya dalam N *unknowns* : nilai *alpha* piksel.

$$\begin{aligned} J(\alpha) &= \min_{a,b} J(\alpha, a, b). \\ J(\alpha) &= \alpha^T L \alpha \end{aligned} \tag{2.25}$$

α adalah vektor $N \times 1$ dan L adalah Matting Laplacian $N \times N$, dimana entri ke-(i, j) nya dapat didefinisikan dalam persamaan:

$$L_{i,j} = \sum_{k|(i,j) \in w_k} \delta_{ij} - \frac{1}{|\omega_k|} \left(1 + (I_i - \mu_k) \left(\Sigma_k + \frac{\epsilon}{|\omega_k|} I3 \right)^{-1} (I_j - \mu_k) \right) \tag{2.26}$$

δ_{ij} adalah *kronecker delta*, I_i dan I_j adalah dua pasang piksel ((x_i, y_i),(x_j, y_j)) di dalam 3×3 *small window*.

$$\begin{aligned} i_k &= (x + x_i - r) + (y + y_i - r) * w \\ j_k &= (x + x_j - r) + (y + y_j - r) * w \end{aligned} \tag{2.27}$$

x dan y adalah piksel citra, w adalah lebar dari citra, r adalah radius dari ukuran *window* lokal (nilainya 1). Jika $i_k = j_k$ maka nilai *kronecker delta*-nya 1 selain itu

nilainya 0. μ_k adalah *mean* dari intensitas dalam *window* di sekitar piksel k (vektor 3×1), $|\omega_k|$ adalah jumlah piksel dalam *window* disekitar piksel k, Σ_k adalah *covariance* matriks (3×3) pada *color channel*. Nilai ϵ sebesar 0.1^7 dan $I3$ adalah matriks identitas 3×3.

Selanjutnya menulis ulang persamaan 2.24 dalam bentuk matriks untuk membuktikan *cost function* pada persamaan 2.25, maka persamaannya dapat didefinisikan sebagai berikut.

$$\begin{aligned} J(\alpha, a, b) &= \sum_{j\in i}\left(\sum_{i\in w_j}(\alpha_i - a_j - b_j)^2 + \epsilon a_j^2\right) \\ &= \sum_{j=1}^{N}\left\|\begin{bmatrix} I_1^{j^T} & 1 \\ I_2^{j^T} & 1 \\ \dots & \dots \\ I_w^{j^T} & 1 \\ \sqrt{\varepsilon} & 0 \end{bmatrix}\begin{bmatrix} a_j \\ b_j \end{bmatrix} - \begin{bmatrix} \alpha_1^j, \\ \alpha_2^j \\ \dots \\ \alpha_w^j \\ 0 \end{bmatrix}\right\|^2 \end{aligned} \tag{2.28}$$

Setelah menulis ulang persamaan 2.24 dalam bentuk matriks, didapatkan persamaan:

$$J(\alpha, a, b) = \sum_k \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\|^2 \tag{2.29}$$

Untuk setiap *alpha matte* yang diberikan, pasangan optimal a_k^*, b_k^* disetiap *windows* w_k adalah solusi untuk *least squares problem*.

$$\begin{aligned} (a_k^*, b_k^*) &= argmin \left\| G_k \begin{bmatrix} a_k \\ b_k \end{bmatrix} - \bar{\alpha}_k \right\| \\ &= \left(G_k^T G_k\right)^{-1} G_k^T \bar{\alpha}_k \end{aligned} \tag{2.30}$$

Dari persamaan di atas, a_k^* dan b_k^* adalah fungsi dari *alpha*, jadi seluruh masalah ada pada fungsi dari *alpha* tersebut. Dengan melakukan substitusi solusi di atas ke dalam persamaan 2.29 akan menghasilkan persamaan *cost function* seperti pada persamaan 2.25.

$$
\begin{aligned}
\bar{G}_k &= I - G_k \left(G_k^T G_k\right)^{-1} G_k^T \\
J(\alpha) &= \sum_k \bar{\alpha}_k^T \bar{G}_k^T \bar{G}_k \bar{\alpha}_k \\
J(\alpha) &= \sum_{k=1}^{N} \left\| G_k \left(G_k^T G_k\right)^{-1} G_k^T \alpha_k - \alpha_k \right\|^2 \\
&= \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}^T L \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} \\
&= \alpha^T L \alpha
\end{aligned}
\tag{2.31}
$$

(a) Sribbles **(b)** Alpha matte **(c)** Citra latar depan

Gambar 2.12: Ilustrasi *Closed-Form Matting*

Sumber : Levin dkk. (2008)

Diketahui bahwa *constraints* yang disediakan pengguna pada *matte* dapat melalui *scribbles* atau *trimap*. Pengguna menggunakan *background brush* (contoh: warna hitam) untuk mengindikasikan sebagai piksel *background* ($\alpha = 0$) dan *foreground brush* (contoh: warna putih) untuk mengindikasikan sebagai piksel

foreground ($\alpha = 1$). Untuk mengekstrak *alpha matte* (gambar 2.12a) yang *matching* dengan *constraints* yang diberikan oleh pengguna maka digunakanlah persamaan berikut.

$$\alpha = (L + \lambda D)^{-1} \lambda b \tag{2.32}$$

L adalah *Laplacian matting* $N \times N$, N jumlah seluruh piksel dalam citra masukan, *lambda* yang dipakai dalam (Levin dkk., 2008) sebesar 100, α adalah vektor $N \times 1$, D adalah matriks diagonal yang elemen diagonalnya 1 untuk *contrained* piksel dan 0 untuk piksel lainnya, b adalah vektor yang berisi nilai α yang ditentukan untuk *contrained* piksel dan 0 untuk semua piksel lainnya, *constrained* piksel yang dimaksud contohnya adalah piksel-piksel yang terkena *brush* hitam ataupun putih dimana nilai α pada piksel tersebut dipaksa untuk bernilai 0 ataupun 1. Persamaan akhir untuk menemukan *alpha matte* digunakan persamaan linear sebagai berikut.

$$\begin{aligned} AX &= B \\ A &= L + \lambda D \\ B &= \lambda b \end{aligned} \tag{2.33}$$

X adalah α, A berbentuk matriks dan B berbentuk vektor. Selanjutnya melakukan proses ekstraksi latar belakang dengan menggunakan *alpha matte* yang sudah ditemukan dengan cara mengkombinasikan RGB *color channel* dan *alpha channel*. RGB *color channel* didapat dari citra masukan sedangkan *alpha channel* didapat dari *alpha matte* yang sebelumnya sudah ditemukan. Kombinasi kedua *channel* yang dimaksud adalah menumpuk kedua citra tersebut di sumbu ketiga. Hasil ekstraksi latar depan dapat dilihat pada gambar 2.12c.

BAB III

METODOLOGI PENELITIAN

A. Sumber Data

Data yang akan digunakan untuk mengimplementasikan penelitian ini berupa data citra digital dengan objek ikan yang diunduh dari situs GitHub dan situs jejaring sosial tempat orang mengunggah dan berbagi foto untuk dilihat orang lain. Citra yang diunduh adalah data yang bebas untuk digunakan ataupun diedit. Meskipun sumber data yang digunakan dalam penelitian ini terdiri dari beberapa sumber yang berbeda, akan tetapi jenis ikannya sama, itu dilakukan untuk menguji apakah metode yang diusulkan dapat bekerja dengan baik atau tidak, dengan kualitas citra digital yang berbeda. Semua citra yang akan digunakan harus diubah terlebih dahulu ukuran citranya. Ukuran citra yang dipakai untuk penelitian ini adalah 420×320 piksel. Tingkat piksel yang lebih kecil akan lebih mempercepat proses ekstraksi latar depan dengan metode GrabCut menggunakan *Saliency Map* maupun metode *Closed-Form Matting*. Citra yang digunakan dalam penelitian ini mempunyai beberapa kriteria untuk objek ikannya seperti, citra dengan ikan tampak samping, citra dengan ikan tampak depan dan citra dengan objek lebih dari satu ikan (*multiple object*).

Gambar 3.1: Citra ikan "Abudefduf" dan "Chaetodon"
Sumber : https://github.com/mekas/fishR, diakses pada 29 Agustus 2021

Sumber :
www.flickr.com/
Kevin Bryant

Sumber :
wikimedia.org/
Kora27

Sumber :
www.flickr.com/
Derek Keats

Sumber :
www.flickr.com/
Ryan McMinds

Sumber :
www.flickr.com/
Francois Libert

Sumber :
www.flickr.com/
Andreas Marz

Sumber :
www.flickr.com/
Andreas Marz

Sumber :
www.flickr.com/
Peter Fuller

Gambar 3.2: Citra ikan dengan lebih dari satu objek dan tampak depan
Diakses pada 29 Agustus 2021

Metode *Closed-Form Matting* adalah sebuah metode *supervised* yang membutuhkan dua input data. Data pertama adalah citra digital yang akan diekstrak latar depannya dan data kedua adalah *scribbles*. *Scribbles* adalah citra dari data pertama yang sudah diberi tanda dengan menggunakan *brush*. Latar belakang diberi tanda *brush* berwarna hitam sedangkan latar depan ditandai dengan *brush* berwarna putih. Data *scribbles* yang akan digunakan untuk mengimplementasikan metode *Closed-Form Matting* berupa data yang dibuat dengan *software* paint. Data ini berupa *file* yang berformat *portable network graphics* (*.png).

B. Deskripsi Sistem

Dalam penelitian ini sistem yang akan dibuat adalah sistem yang akan dipakai untuk ekstraksi latar depan pada citra ikan dengan metode GrabCut yang telah diautomasi menggunakan metode deteksi *saliency* berbasis *superpixel*. Fokus penulis dari penelitian ini adalah memisahkan objek ikan dengan latar belakangnya. Citra yang akan digunakan oleh penulis untuk menguji sistem ekstraksi latar depan pada penelitian ini berasal dari situs GitHub dan situs jejaring sosial tempat orang mengunggah dan berbagi foto.

Gambar 3.3: Citra *scribbles*

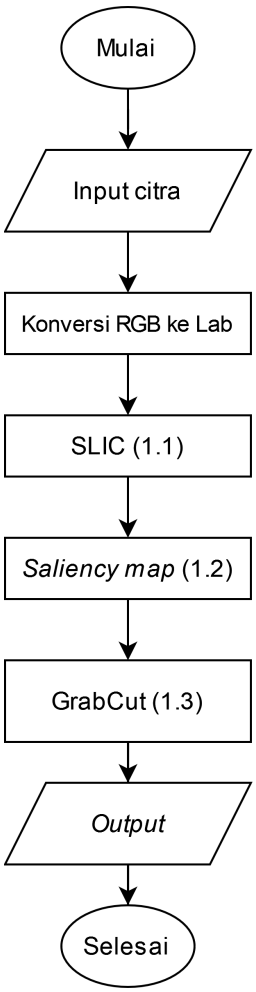
Bahasa pemrograman yang akan digunakan dalam membuat sistem adalah Python versi 3. Tujuan dalam penelitian adalah mendapatkan hasil segmentasi objek ikan atau ekstraksi latar depan pada sebuah citra dengan objek ikan yang mana akan dihasilkan hanya objek ikannya saja dengan latar belakang yang transparan.

Tahapan proses dalam ekstraksi latar depan dengan metode GrabCut menggunakan *saliency map* adalah menginputkan citra ikan, mendeteksi objek ikan menggunakan *saliency map* berbasis *superpixel* dan segmentasi citra dengan GrabCut. Terdapat beberapa tahap dalam proses deteksi objek ikan menggunakan *saliency map* berbasis *superpixel* di antaranya yaitu konversi citra RGB menjadi Lab, melakukan operasi SLIC dan menghitung nilai *saliency* dari setiap *superpixel*.

C. Perancangan Sistem

Pada subbab ini akan dijelaskan desain proses yang digunakan untuk mengetahui proses apa saja yang dilakukan untuk ekstraksi latar depan citra menggunakan metode GrabCut yang diautomasi dengan deteksi *saliency* berbasis *superpixel*. Perancangan sistem dalam penelitian ini adalah implementasi dari

ekstraksi latar depan citra digital dengan metode GrabCut yang diautomasi menggunakan deteksi *saliency* berbasis *superpixel*. Proses pertama yang dilakukan adalah menginputkan data yaitu citra dengan objek ikan di dalamnya, dilanjutkan dengan proses deteksi objek ikan pada citra (*region of interest*) menggunakan deteksi *saliency* berbasis *superpixel*, proses ini bertujuan untuk membuat *initial rectangle* dalam algoritma GrabCut. Setelah proses tersebut selesai, dilanjutkan dengan proses segmentasi menggunakan metode GrabCut.

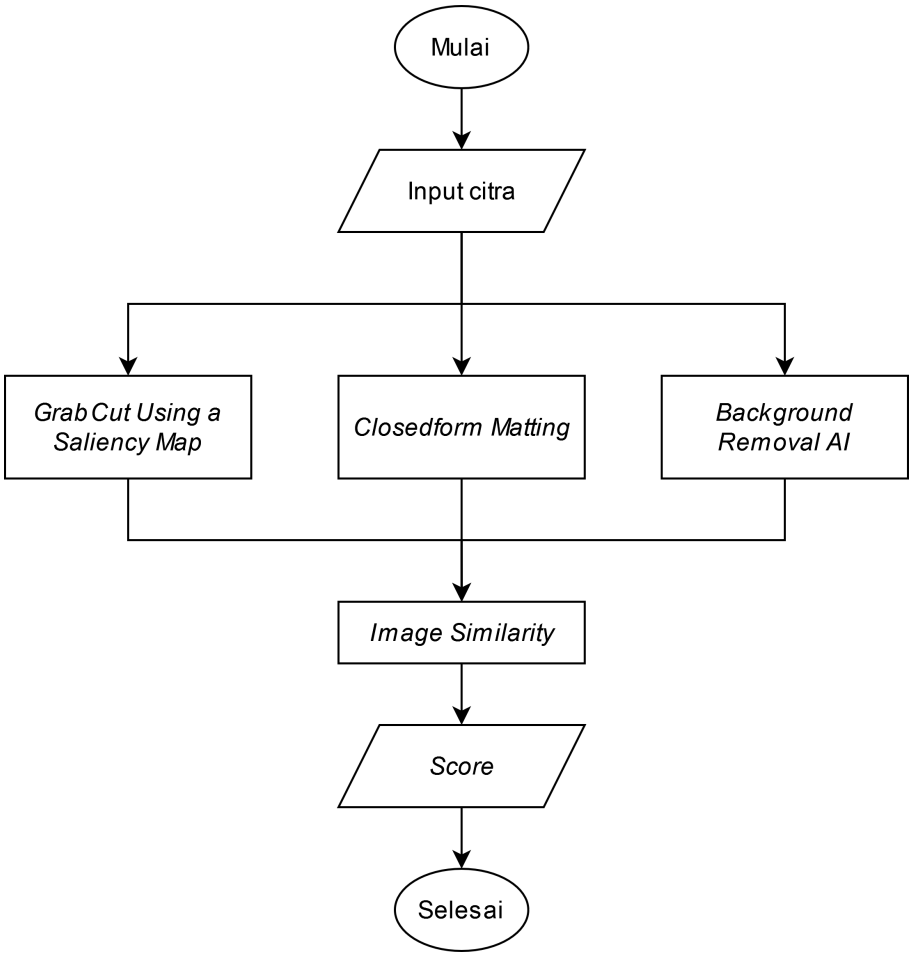


Gambar 3.4: Diagram alir metode GrabCut menggunakan *Saliency Map*

Untuk melihat proses sistem yang mencakup proses sistem dari awal yaitu proses input citra sampai terbentuknya keluaran (*output*) berupa hasil ekstraksi latar

depan dapat dilihat dengan jelas dalam diagram alir (*flowchart*). Setelah melakukan pengujian terhadap satu citra, dilakukan proses ekstraksi kembali dengan menggunakan citra selanjutnya hingga semua citra telah terekstrak latar depannya. Gambar 3.4 adalah desain langkah penelitian atau proses yang dilakukan dalam sistem pada penelitian ini.

Dalam penelitian ini dilakukan perbandingan antara dua metode ekstraksi latar depan yang berbeda untuk mengetahui sejauh mana hasil dari metode ekstraksi latar depan yang diusulkan. Metode pertama adalah GrabCut yang diautomasi menggunakan *saliency map* dan metode kedua adalah *Closed-Form Matting*. Metode pertama bersifat *unsupervised* sedangkan metode kedua bersifat *supervised*.



Gambar 3.5: Diagram alir perbandingan metode

Pada gambar 3.4, dapat dilihat yang pertama dilakukan yaitu menginputkan citra ikan. Kemudian dilakukan proses mendeteksi *saliency* berbasis *superpixel* pada citra tersebut dan langkah terakhir adalah segmentasi citra dengan metode GrabCut dengan inisialisasi input dari hasil proses deteksi *saliency* atau *saliency map* yang telah dibuat pada langkah sebelumnya. Pada proses ini, citra disegmentasi menjadi dua wilayah yaitu, wilayah latar depan dan latar belakang. Hasil segmentasi citra menggunakan metode GrabCut yaitu berupa latar depannya saja dan latar belakangnya menjadi transparan.

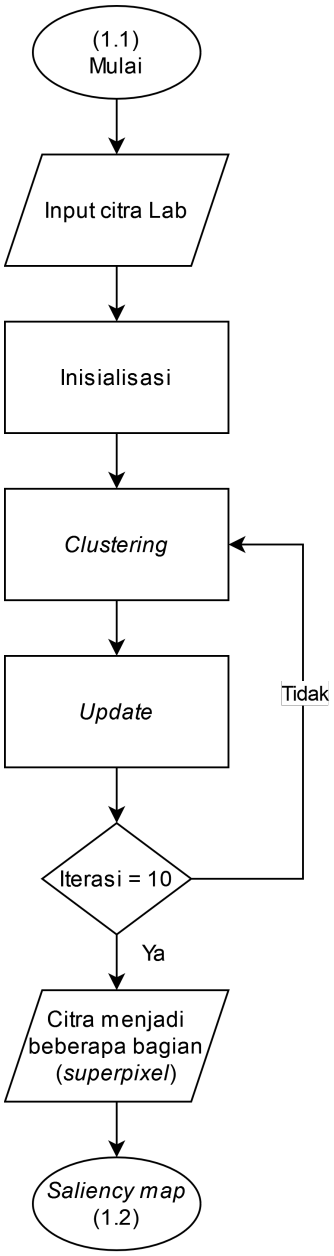
1. Proses Input Citra

Data yang akan digunakan untuk mengimplementasikan penelitian ini adalah berupa data yang diunduh dari situs GitHub dan diunduh dari situs jejaring sosial tempat orang mengunggah dan berbagi foto dengan kata kunci jenis ikan "Abudefduf" dan "Chaetodon". Terdapat beberapa citra masukan yang digunakan dalam proses ekstraksi latar depan dengan metode GrabCut menggunakan *saliency map*, yaitu citra dengan objek ikan tampak samping, ikan tampak depan dan lebih dari satu ikan. Dalam proses mendeteksi *saliency* (ROI) berbasis *superpixel*, data masukan yang berupa citra ikan dengan ruang warna RGB harus dilakukan proses konversi ruang warna menjadi Lab. Dikarenakan ruang warna Lab lebih linear secara persepsi daripada ruang warna lainnya. Persepsi linear tersebut memiliki arti bahwa perubahan dalam jumlah yang sama dalam nilai warna akan menghasilkan perubahan dengan kepentingan visual yang hampir sama.

2. Data Keluaran

Data yang akan dihasilkan dari ekstraksi latar depan dengan metode GrabCut yang diautomasi menggunakan deteksi *saliency* berbasis *superpixel* berupa citra latar depan atau citra dengan latar belakang yang transparan.

3. Proses SLIC



Gambar 3.6: Diagram alir algoritma SLIC

Langkah pertama dalam proses menemukan *saliency map* yaitu dengan menggunakan metode SLIC untuk mempartisi citra masukan menjadi sejumlah *superpixel*, yang biasanya memiliki bentuk yang teratur dan kompak dengan batas yang lebih baik. Tahap pertama pada proses SLIC adalah inisialisasi pusat *cluster*

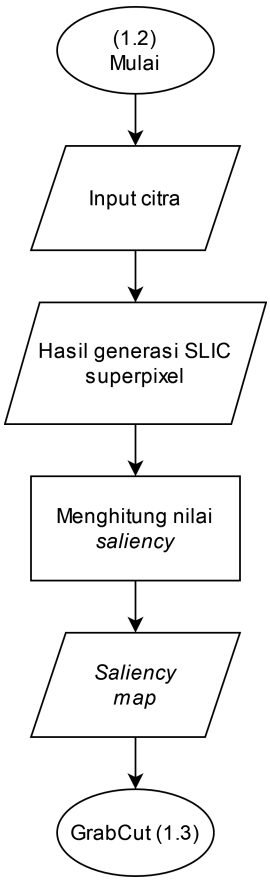
dengan ukuran awal *superpixel* disetel ke $N/200$ dan jarak antar pusat *cluster* awal $S = \sqrt{N/200}$. Dimana N menunjukkan jumlah piksel pada citra dan jumlah *superpixel* yang dihasilkan kurang lebih 200, yang umumnya cukup untuk menjaga batas (*boundaries*) yang berbeda dengan baik. Contoh segmentasi *superpixel* pada citra ditunjukkan pada gambar 2.9a, dimana batas-batas *superpixel* yang berbeda digambarkan menggunakan garis putih.

Selanjutnya memindahkan pusat *cluster* ke lokasi piksel yang memiliki nilai gradien (persamaan 2.5) terendah di lingkungan 3×3 pusat *cluster* tersebut. Tahap berikutnya yaitu menempatkan piksel ke pusat *cluster* terdekat (terletak dalam area $2S \times 2S$ di sekitar pusat *cluster*) menggunakan persamaan 2.2. Proses menempatkan piksel dengan pusat *cluster* terdekat dan menghitung ulang pusat *cluster*, diulang sampai tidak ada perubahan. Tetapi 10 iterasi sudah cukup untuk sebagian besar citra.

4. Proses Deteksi *Saliency*

Deteksi *saliency* berbasis *superpixel* adalah proses untuk mendeteksi objek penting atau *region of interest* (ROI) dalam sebuah citra digital. Tujuan deteksi *saliency* berbasis *superpixel* dalam penelitian ini adalah untuk membuat *initial rectangle* dalam algoritma GrabCut dengan maksud menggantikan interaksi manusia yang tidak stabil atau dengan kata lain menjadikan metode GrabCut menjadi otomatis tanpa memerlukan interaksi manusia. Deteksi *saliency* di sini dilakukan untuk menghasilkan ROI yang dapat digunakan sebagai batas-batas *rectangle* dalam inisialisasi metode GrabCut. Gambar 3.7 adalah diagram alir secara umum deteksi *saliency* berbasis *superpixel*.

Selanjutnya yaitu menghitung nilai *saliency* setiap *superpixel* menggunakan persamaan 2.6. Dalam persamaan 2.8 yang didefinisikan sebagai jumlah perpotongan antar dua histogram, histogram tersebut adalah histogram warna tiap *superpixel* yang telah dikuantisasi dan dinormalisasi sebelumnya. Setelah menghitung nilai *saliency* setiap *superpixel*, maka dihasilkanlah *saliency map*

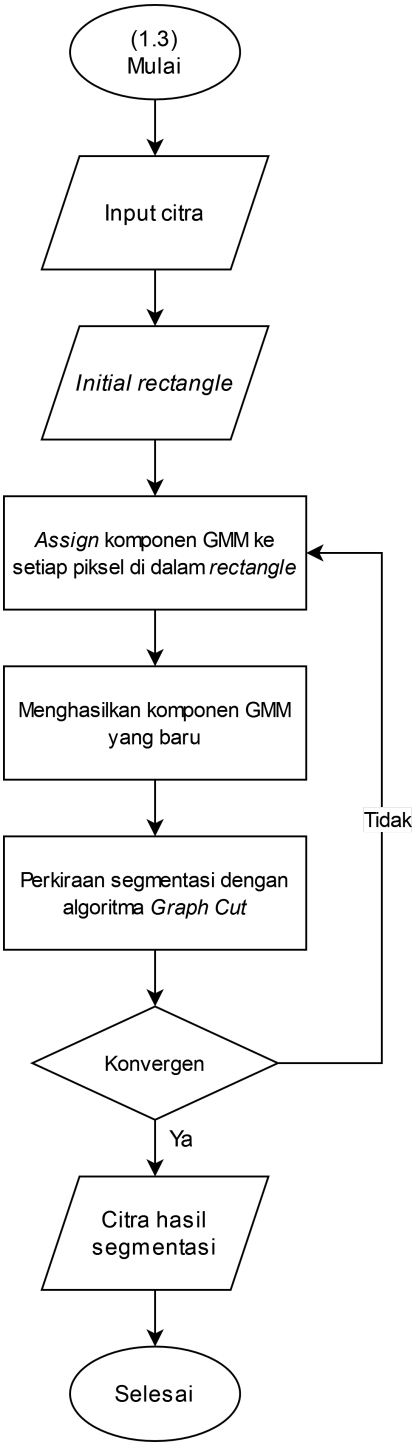


Gambar 3.7: Diagram alir algoritma *Saliency Map*

5. Proses GrabCut

Segmentasi dalam penelitian ini adalah memisahkan objek ikan dengan latar belakangnya pada citra digital. Citra digital yang telah ditemukan *saliency map*-nya, dilakukan segmentasi citra menggunakan metode GrabCut dengan inisialisasi *rectangle* yang telah terbentuk dari *saliency map* dalam citra tersebut. Gambar 3.8 adalah proses bagaimana metode ini berjalan.

Proses segmentasi citra menggunakan metode GrabCut bisa dibagi menjadi beberapa tahap. Pertama, menandai daerah yang melingkupi objek dengan *rectangle* dan piksel di dalam *rectangle* dapat digunakan untuk membangun *foreground* GMM (persamaan 2.11) dan di saat yang sama piksel yang berada di luar *rectangle* digunakan untuk membangun *background* GMM. Tahap kedua, GMM ini digunakan untuk membentuk *cost function U* (persamaan 2.10) dan segmentasi akan berjalan



Gambar 3.8: Diagram alir GrabCut

(persamaan 2.13). Tahap ketiga, hasil segmentasi membentuk *background* dan *foreground* GMM yang baru, perlu memastikan jika GMM yang baru serupa dengan yang lama atau tidak. Jika mereka tidak berubah, maka segmentasinya berhenti, lalu program akan selesai. Jika mereka berubah, program akan kembali ke tahap kedua sampai mereka tidak mengalami perubahan.

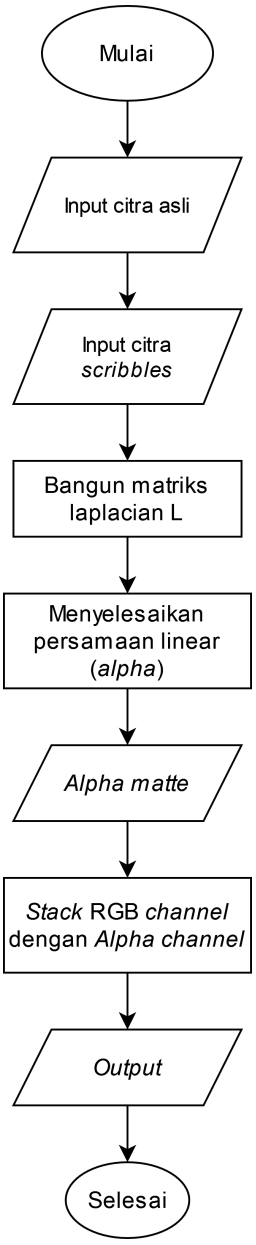
6. Proses *Closed-Form Matting*

Closed-Form Matting adalah metode ekstraksi latar depan yang bersifat *supervised*. Metode ini dipakai sebagai metode pembanding dalam penelitian ini. *Closed-Form Matting* membutuhkan dua data masukan yaitu citra yang ingin diektraksi latar depannya dan citra *scribbles*. Citra *scribbles* adalah citra yang sudah diberi tanda dengan menggunakan *brush*, latar belakang ditandai dengan *brush* berwarna hitam sedangkan latar depan ditandai dengan *brush* berwarna putih.

Langkah pertama dalam metode *Closed-Form Matting* adalah membangun *Laplacian* matriks (persamaan 2.26) berdasarkan data masukan citra asli bukan *scribbles*. Kemudian dilanjutkan dengan menyelesaikan sistem persamaan linear (persamaan 2.32) untuk menemukan *alpha matte*. Citra latar depan dapat dihasilkan dengan menambahkan metode ekstensi yaitu menumpuk RGB *channel* yang berasal dari data masukan citra asli dengan *alpha channel* (*opacity* piksel latar depan) yang dihasilkan dari *Closed-Form Matting*. Hasil dari metode ekstensi adalah citra asli dengan dimensi ketiganya bertambah 1 *layer* (*alpha channel*) dimana dapat disebut sebagai *filter*, jika nilai *alpha*-nya 1 maka warna dalam piksel tersebut ditampilkan dan jika nilai *alpha*-nya 0 maka warna dalam piksel tersebut menjadi transparan. Dengan demikian hasilnya dapat dikatakan dengan citra latar depan dikarenakan hanya menampilkan objek dari citra asli.

7. *Image Similarity*

Proses pencocokan kemiripan atau kemiripan karakteristik suatu citra salah satunya dengan menggunakan metode pengukuran jarak. Salah satu metode



Gambar 3.9: Diagram alir *Closed-Form Matting*

pengukuran jarak dalam mengukur kemiripan atau kemiripan suatu citra adalah dengan menggunakan metode *Euclidean Distance*, sehingga dapat diketahui kemiripan tekstur citra uji dengan citra referensi (Dewi dkk., 2018).

Untuk menilai kesamaan citra hasil dari metode GrabCut menggunakan *saliency map* dengan hasil validasi menggunakan *background removal AI* dan citra hasil metode *Closed-Form Matting* dengan hasil validasi menggunakan *background removal AI* maka digunakanlah metode *Euclidean Distance* untuk menilai kesamaan atau kemiripan antara dua citra yang diuji dengan satu citra referensi (Agrawal, 2021).

$$d = \left((p_1 - q_1)^2 + (p_2 - q_2)^2\right)^{1/2} \tag{3.1}$$

Langkah pertama untuk menghitung *image similarity* menggunakan *Euclidean Distance* adalah menginputkan citra uji dan citra referensi sebagai *array*. Karena citra masukan adalah citra berwarna yang mempunyai tiga saluran warna yaitu *array* tiga dimensi. Maka langkah selanjutnya adalah meratakan *array* tersebut sehingga menjadi *array* satu dimensi. Kemudian membuat histogram untuk setiap citra masukan, dimana untuk setiap indeks 0 sampai 255 akan dihitung kemunculan nilai piksel tersebut dalam citra. Setelah memiliki histogram, maka digunakanlah *Euclidean Distance* untuk mengetahui perbedaan kedua histogram. Berdasarkan jarak antara histogram citra uji dan citra referensi, semakin kecil nilainya maka semakin mirip citra yang diuji dengan citra referensi.

D. Skenario Pengujian

Pada subbab ini akan dibahas mengenai skenario pengujian yang akan dilakukan pada penelitian ini. Secara keseluruhan pengujian dalam penelitian ini akan membandingkan hasil dari ekstraksi latar depan menggunakan metode yang diusulkan dengan hasil dari ekstraksi latar depan menggunakan metode *Closed-Form Matting* yang bertujuan untuk mengetahui sebaik apa keluaran (*output*) yang

dihasilkan dari metode GrabCut menggunakan *saliency map*. Untuk memvalidasi hasil dari kedua metode tersebut, penulis menggunakan *background removal AI* dan keluarannya dijadikan sebagai acuan pembanding untuk hasil dari kedua metode tersebut. Dalam menilai hasil dari metode GrabCut menggunakan *saliency map* dan hasil dari metode *Closed-Form Matting* digunakan nilai *image similarity*.

1. Ekstraksi Latar Depan Dengan GrabCut Menggunakan *Saliency Map*

Metode GrabCut menggunakan *saliency map* membutuhkan satu citra masukan untuk melakukan proses ekstraksi latar depan. Citra masukan yang digunakan untuk menguji metode GrabCut menggunakan *saliency map* diantaranya citra dengan objek ikan tampak samping, citra dengan objek ikan tampak depan dan citra dengan objek lebih dari satu ikan (*multiple objects*). Setiap citra yang digunakan memiliki resolusi 420×320 piksel. Proses ekstraksi latar depan dengan GrabCut menggunakan *saliency map* dilakukan ke setiap citra masukan. Keluaran (*output*) yang dihasilkan pada metode ini berupa citra latar depan atau citra dengan latar belakang yang transparan.

2. Ekstraksi Latar Depan Dengan *Closed-Form Matting*

Proses ekstraksi latar depan dengan *Closed-Form Matting* membutuhkan dua citra masukan. Citra masukan yang pertama sama seperti citra masukan yang dijelaskan pada metode sebelumnya. Citra masukan yang kedua yaitu citra *scribbles*, dapat dilihat pada gambar 3.3. Proses ekstraksi latar depan dengan metode *Closed-Form Matting* dilakukan ke setiap pasang citra masukan. Keluaran yang dihasilkan pada metode ini berupa citra latar depan atau citra dengan latar belakang yang transparan.

3. Ekstraksi Latar Depan Dengan *Background Removal AI*

Background removal AI yang digunakan dalam penelitian ini berasal dari internet yang beralamatkan https://www.remove.bg/. *Background removal AI*

digunakan untuk mendapatkan citra dengan latar belakang yang transparan dari citra yang diinputkan. Citra yang dihasilkan oleh *background removal AI* diasumsikan sebagai citra latar depan yang hasilnya paling mendekati objeknya sebagaimana mata manusia melihat objek dalam sebuah citra. Citra yang digunakan dalam proses ini sama seperti citra masukan yang telah dijelaskan pada metode sebelumnya. Keluaran yang dihasilkan oleh *background removal AI* dijadikan sebagai acuan pembanding pada metode GrabCut menggunakan *saliency map* dan metode *Closed-Form Matting*.

4. Menghitung Nilai *Image Similarity*

Dalam menilai hasil ekstraksi latar depan dari kedua metode yang akan dibandingkan, yaitu metode GrabCut menggunakan *saliency map* dan metode *Closed-Form Matting* dengan mencari nilai *image similarity* menggunakan *Euclidean Distance*. Untuk menemukan nilai *image similarity*, dibutuhkan dua input yaitu, citra uji dan citra referensi. Tujuannya untuk mengetahui seberapa mirip citra uji dengan citra referensi. Citra uji pada tahap ini adalah citra latar depan yang dihasilkan dari metode GrabCut menggunakan *saliency map* dan metode Closed-Form Matting, sedangkan citra referensi didapat dari hasil *background removal AI*. Pertama, menghitung nilai *image similarity* dari citra hasil GrabCut dan citra referensi. Kedua, menghitung nilai *image similarity* dari citra hasil Closed-Form Matting dan citra referensi. Selanjutnya, kedua nilai tersebut dibandingkan. Semakin kecil nilai *Euclidean Distance* yang dihasilkan, maka semakin mirip citra yang diuji dengan citra referensi.

BAB IV

HASIL DAN PEMBAHASAN

A. Pengujian Data Masukan

Data masukan yang digunakan untuk pengujian dalam penelitian ini adalah citra dengan objek ikan tampak samping, citra dengan objek ikan tampak depan dan citra dengan beberapa objek ikan. Data masukan yang digunakan dalam pengujian sebanyak 18 citra ikan. Data masukan yang digunakan dalam pengujian bertipe jpg dan jpeg. Kriteria citra ikan yang digunakan dapat dilihat pada tabel di bawah ini.

Tabel 4.1: Kriteria data citra masukan

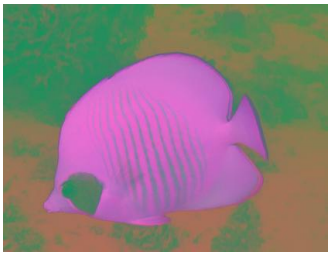
No	Nama *file*	Ukuran panjang dan lebar citra (piksel)	Ukuran *file*
1.	Abudefduf1.jpeg	420 X 320	41 KB
2.	Abudefduf2.jpeg	420 X 320	15 KB
3.	Abudefduf3.jpg	420 X 320	84 KB
4.	Abudefduf4.jpg	420 X 320	24 KB
5.	Abudefduf5.jpg	420 X 320	66 KB
6.	Abudefduf_frontlook.jpg	420 X 320	104 KB
7.	Abudefduf_multiple1.jpg	420 X 320	133 KB
8.	Abudefduf_multiple2.jpg	420 X 320	96 KB
9.	Abudefduf_multiple3.jpg	420 X 320	43 KB
10.	Chaetodon1.jpg	420 X 320	21 KB
11.	Chaetodon2.jpeg	420 X 320	27 KB
12.	Chaetodon3.jpeg	420 X 320	83 KB
13.	Chaetodon4.jpeg	420 X 320	72 KB
14.	Chaetodon5.jpeg	420 X 320	111 KB
15.	Chaetodon_frontlook.jpg	420 X 320	54 KB

No	Nama *file*	Ukuran panjang dan lebar citra (piksel)	Ukuran *file*
16.	Chaetodon_multiple1.jpg	420 X 320	163 KB
17.	Chaetodon_multiple2.jpg	420 X 320	41 KB
18.	Chaetodon_multiple3.jpg	420 X 320	40 KB

1. Konversi Citra ke Ruang Warna Lab

Citra digital yang diuji memiliki ruang warna RGB. Citra yang sudah masuk ke dalam sistem kemudian dilanjutkan dengan pengubahan ruang warna RGB ke ruang warna Lab. Tabel 4.2 menunjukkan beberapa hasil konversi ruang warna Lab dan gambar 4.1 adalah potongan kode yang memanggil fungsi konversi ruang warna RGB ke ruang warna Lab pada citra masukan.

Tabel 4.2: Hasil pengubahan ruang warna RGB ke ruang warna Lab pada citra masukan

No	Citra asli	Citra Lab
1.		
2.		

No	Citra asli	Citra Lab
3.		
4.		

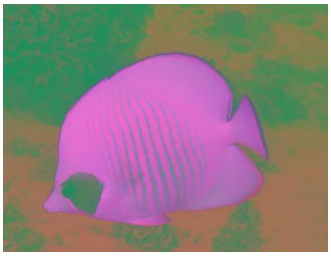
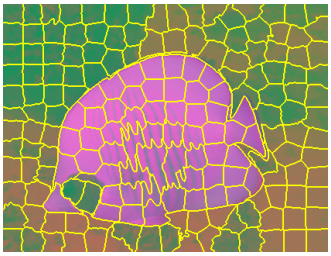
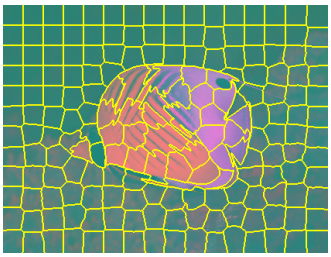
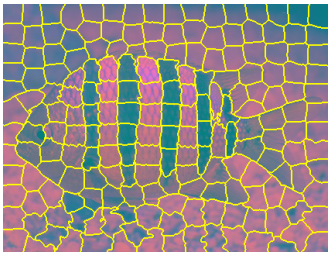
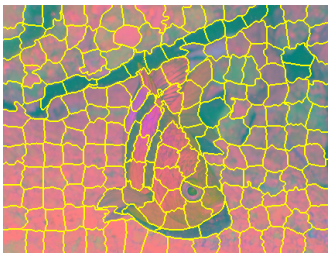
```
from skimage import io, color

def rgbToLab(img_rgb):
    img_lab = color.rgb2lab(img_rgb / 255)
    img_lab_scaled = (img_lab + [0, 128, 128]) / [100, 255, 255]
    return img_lab_scaled

im = io.imread("input/Abudefduf3.jpg")
im_lab = rgbToLab(im)
```

Gambar 4.1: Potongan sampel kode pengubahan ruang warna RGB ke ruang warna Lab

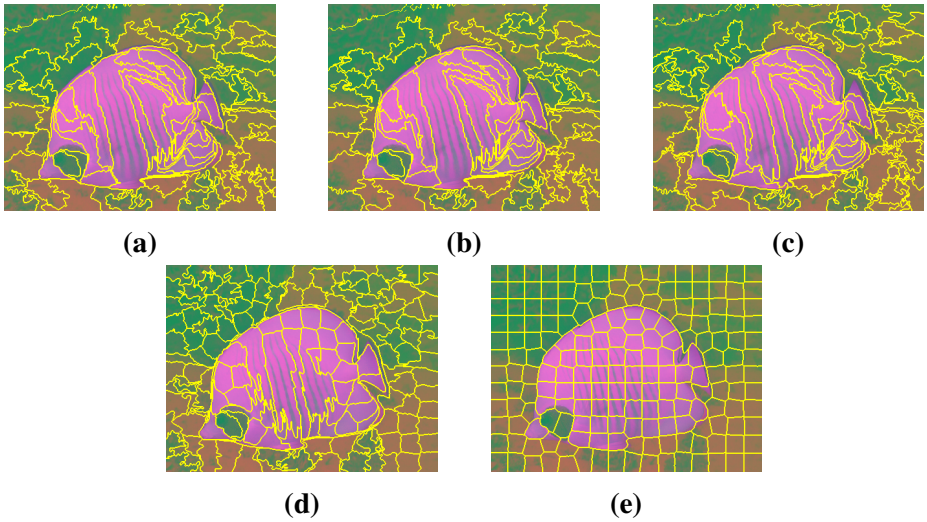
2. Proses Segmentasi Citra Menggunakan SLIC

Citra yang telah diubah ruang warnanya menjadi Lab pada tahap sebelumnya, kemudian digunakan sebagai data masukan pada saat segmentasi citra menggunakan SLIC. Hasil dari proses ini adalah mendapatkan sejumlah segmen atau dapat disebut dengan *superpixel*. Tabel 4.3 menunjukkan beberapa hasil dari proses segmentasi citra menggunakan SLIC.

Tabel 4.3: Hasil segmentasi citra Lab menggunakan metode *Simple Linear Iterative Clustering* (SLIC)

No	Citra Lab	Hasil segmentasi SLIC
1.		
2.		
3.		
4.		

Perkiraan jumlah segmen yang diproduksi mendekati 200. Dalam pengujian, beberapa percobaan dilakukan dengan nilai *compactness* yang berbeda. Gambar 4.2 adalah hasil penggunaan SLIC untuk melakukan segmentasi citra yang memiliki satu objek dengan beberapa nilai *compactness* dalam skala log.



Gambar 4.2: Hasil segmentasi SLIC dengan nilai *compactness* (a) 0.01, (b) 0.1, (c) 1, (d) 10 dan (e) 100

Ketika nilai *compactness* kurang dari atau sama dengan satu, akan dihasilkan segmen yang tidak beraturan, segmen tersebut tidak tumpang tindih atau segmen yang dihasilkan memang jauh lebih sedikit dari nilai segmen yang ditentukan. Nilai *compactness* yang lebih tinggi membuat setiap segmen yang dihasilkan lebih membentuk persegi. Nilai *compactness* sangat bergantung pada kontras citra dan bentuk objek pada citra. Gambar 4.3 adalah potongan kode untuk melakukan segmentasi citra menggunakan SLIC.

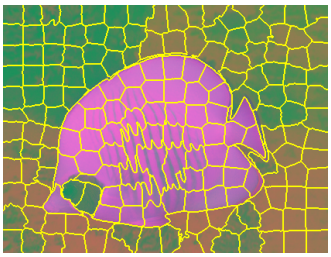
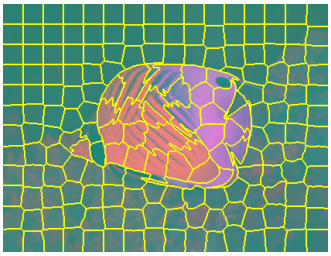
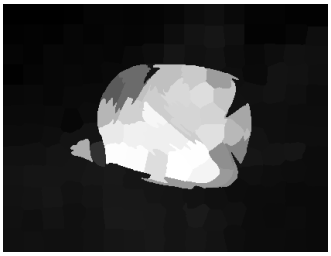
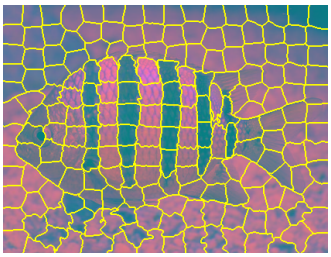
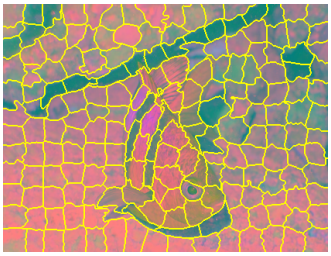
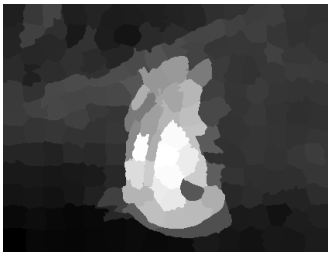
```
from skimage import segmentation

def slic(img_lab, n_segments, compactness):
    segments = segmentation.slic(img_lab, n_segments=n_segments,
                                 compactness=compactness,
                                 sigma=1, start_label=1)
    return segments

segments = slic(im_lab,n_segments=200,compactness=30)
```

Gambar 4.3: Potongan sampel kode segmentasi citra Lab menggunakan metode *Simple Linear Iterative Clustering* (SLIC)

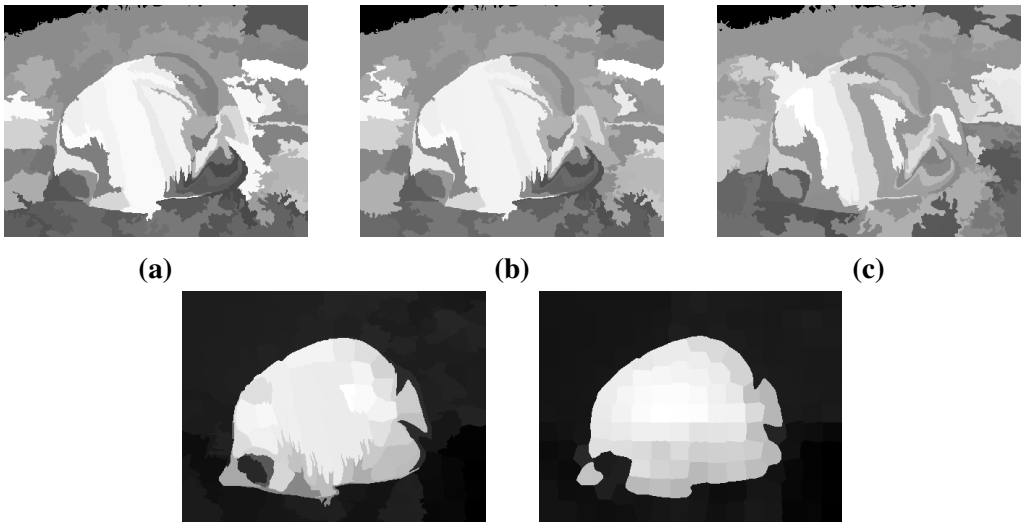
3. Proses Deteksi *Saliency*

Setelah melakukan proses segmentasi citra menggunakan SLIC, dilanjutkan dengan menghitung nilai *saliency* setiap segmennya untuk menghasilkan *saliency map* dari citra masukan. Tabel 4.4 menunjukkan hasil deteksi *saliency* dengan data segmen yang dihasilkan pada tahap sebelumnya.

Tabel 4.4: Hasil proses deteksi *saliency*

No	Hasil segmentasi SLIC	*Saliency map*
1.		
2.		
3.		
4.		

Hasil dalam proses ini bergantung pada segmen yang dihasilkan selama proses segmentasi citra menggunakan SLIC. Dalam pengujian dilakukan beberapa percobaan dengan nilai *compactness* yang berbeda pada proses segmentasi citra menggunakan SLIC. Gambar 4.4 menunjukkan hasil *saliency map* pada satu citra yang memiliki satu objek dengan mengatur nilai *compactness* dalam skala log pada tahap sebelumnya.



(a) **(b)** **(c)**

(d) **(e)**

Gambar 4.4: Hasil *saliency map* dengan nilai *compactness* (a) 0.01, (b) 0.1, (c) 1, (d) 10 dan (e) 100

Pada gambar 4.4, nilai *compactness* yang kurang dari atau sama dengan satu, rata-rata *saliency map* yang dihasilkan tidak beraturan atau tidak membentuk objeknya. Ketika latar belakang dan latar depan pada citra uji sangat kontras, nilai *compactness* sebesar 10 cukup untuk mendeteksi objek yang menonjol pada citra uji. Gambar 4.6 adalah potongan kode untuk mendeteksi *saliency*.

Gambar 4.5: Hasil *saliency map* dengan mengatur nilai *compactness* sebesar 10 pada tahap segmentasi SLIC

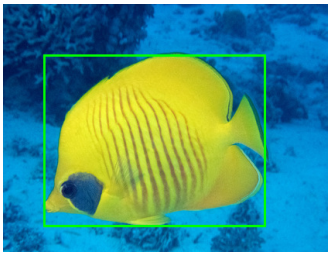
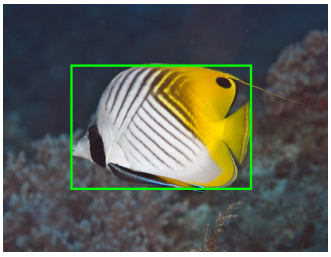
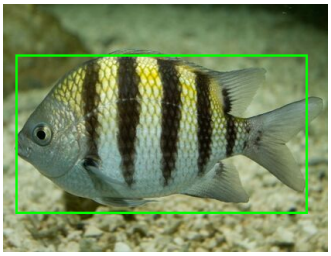
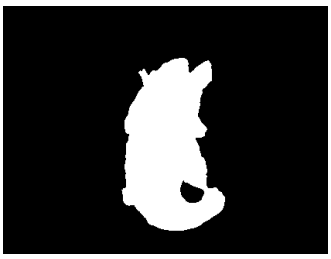
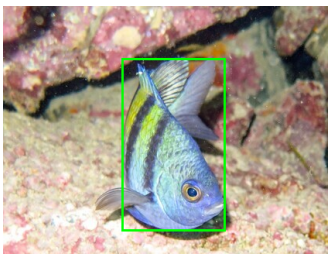
```
from skimage import filters, measure
import histogram as hist
import numpy as np

class saliency_map:
    def __init__(self, img_rgb, segments):
        ...
        self.sp_saliency_score()
        self.build_saliency_map()
        self.otsu()

    def similarity_value(self, i, j, d, norm_hist_sp, list_centroid):
        hist_i = norm_hist_sp[i-1]
        hist_j = norm_hist_sp[j-1]
        color_similarity = hist.hist_intersection(hist_i, hist_j)
        D = n_dimensional_euclidean_distance(
            list_centroid[i-1], list_centroid[j-1])
        spatial_similarity = 1 - (D/d)
        return color_similarity*spatial_similarity

    def D_j(self, j, center, list_centroid):
        return (n_dimensional_euclidean_distance(list_centroid[j-1], center))

    def sp_saliency_score(self):...
    def build_saliency_map(self):
        sa_img = self.segments.copy().astype(float)
        for i in range(np.amax(self.segments)):
            mask = self.segments == i+1
            sa_img[mask] = self.sp_score[i]
        result = np.interp(sa_img, (sa_img.min(), sa_img.max()), (0, 255))
        self.saliency_map = result.astype("uint8")

    def otsu(self):...

sp_sal = saliency_map(im, segments)
```

Gambar 4.6: Potongan sampel kode proses deteksi *saliency*

4. Proses Ekstraksi Latar Depan dengan GrabCut

Setelah tahap deteksi *saliency* dilakukan, kemudian dilanjutkan dengan proses konversi citra hasil deteksi *saliency* menjadi citra biner. Saat mendeteksi objek yang menonjol (*region of interest*) secara lebih eksplisit, *thresholding* adaptif dilakukan pada *saliency map* yang dihasilkan pada proses sebelumnya menggunakan metode Otsu untuk mendapatkan citra biner dari objek yang menonjol tersebut. Tabel 4.5 adalah inisialisasi persegi dalam metode GrabCut dari citra biner yang dihasilkan.

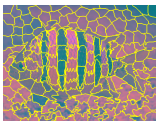
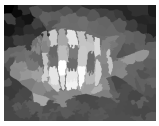
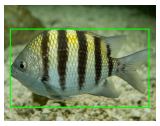
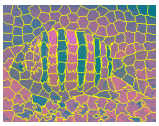
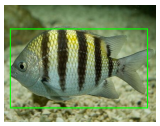
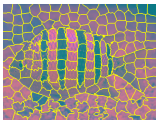
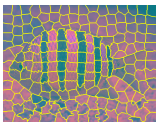
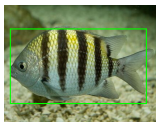
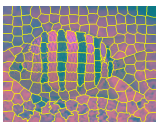
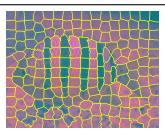
Tabel 4.5: Inisialisasi persegi dalam metode GrabCut

No	Citra biner	Inisialisasi persegi
1.		
2.		
3.		
4.		

Citra biner yang dihasilkan dari *thresholding* adaptif menggunakan metode Otsu dapat menghasilkan koordinat persegi di sekitar area objek. Proses mendapatkan koordinat persegi adalah dengan mencari titik terluar di kiri dan atas bagian putih pada citra biner dan titik terluar di kanan dan bawah bagian putih pada citra biner. Koordinat ini kemudian dapat digunakan sebagai inisialisasi persegi dalam metode ekstraksi latar depan menggunakan GrabCut. Berikut adalah hasil pengujian metode

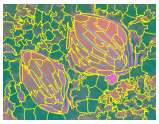
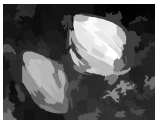
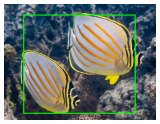
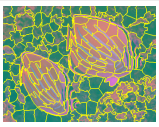
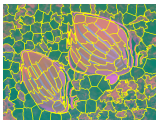
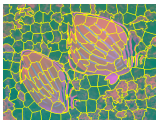
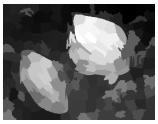
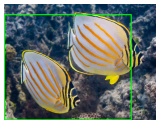
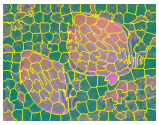
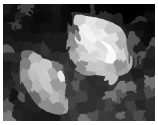
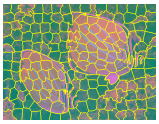
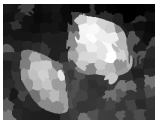
GrabCut yang diautomasi menggunakan *saliency map* pada dua buah citra uji yaitu, citra dengan satu objek dan citra dengan objek lebih dari satu.

Tabel 4.6: Pengujian metode GrabCut yang diautomasi pada citra dengan satu objek

Citra masukan	SLIC		*Saliency map*	GrabCut	
	Comp	Hasil		*Rect*	Hasil
	10				
	15				
	20				
	25				
	30				
	35				

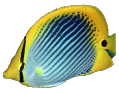
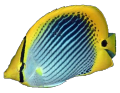
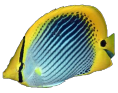
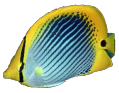
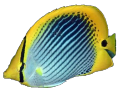
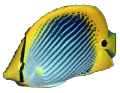
Jika ukuran persegi sesuai dengan ukuran objek (*region of interest*), metode GrabCut dapat menangani ekstraksi latar depan dengan sangat baik, dapat dilihat pada tabel 4.6, tetapi jika ukuran persegi tidak sesuai atau melebar dari ukuran objeknya, maka hasil ekstraksi latar depan (*foreground*) yang dihasilkan kurang baik. Tabel di bawah ini menunjukkan bahwa dalam citra yang berisi beberapa objek ikan, sebagian besar inisialisasi persegi tidak sesuai atau melebar dari ukuran objek. Dikarenakan inisialisasi persegi dalam metode GrabCut hanya satu, maka dalam citra yang mempunyai objek lebih dari satu yang terekstraksi hanya satu objek saja.

Tabel 4.7: Pengujian metode GrabCut yang diautomasi pada citra dengan objek lebih dari satu

Citra masukan	SLIC		*Saliency map*	GrabCut	
	Comp	Hasil		*Rect*	Hasil
	10				
	15				
	20				
	25				
	30				
	35				

Dikarenakan hasil ekstraksi latar depan dengan menggunakan metode GrabCut sangat dipengaruhi oleh inisialisasi persegi, hasil inisialisasi persegi dipengaruhi oleh hasil deteksi *saliency* dan deteksi *saliency* dipengaruhi oleh nilai *compactness* pada segmentasi SLIC, maka dilakukan pengujian nilai *compactness* pada segmentasi SLIC terhadap hasil ekstraksi latar depan dengan menggunakan metode GrabCut. Berikut adalah hasil pengujian pada seluruh citra uji dengan mengatur nilai *compactness* dari rentang 10 sampai 35 dengan kelipatan 5 pada tahap segmentasi SLIC.

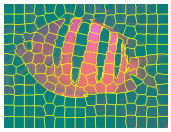
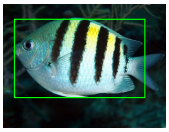
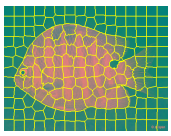
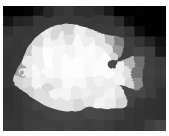
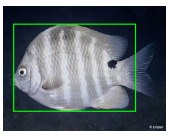
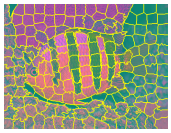
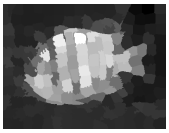
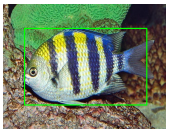
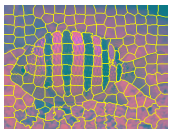
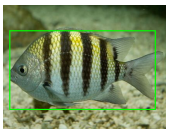
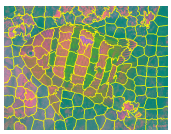
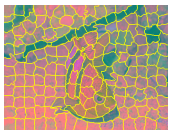
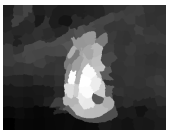
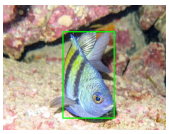
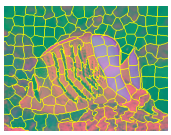
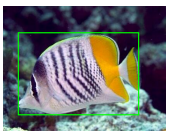
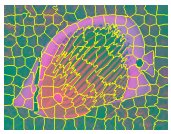
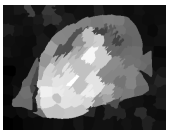
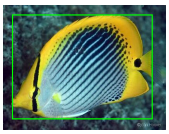
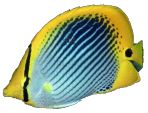
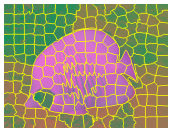
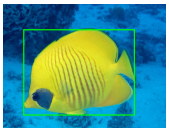
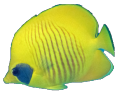
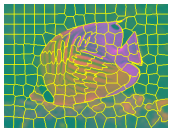
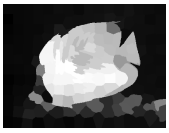
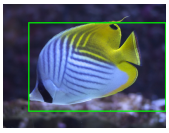
Tabel 4.8: Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi dengan nilai *compactness* SLIC yang berbeda

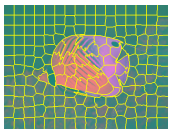
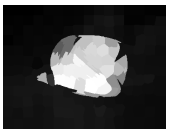
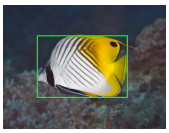
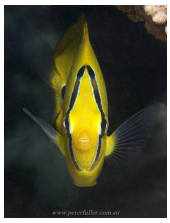
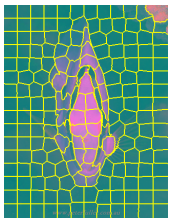
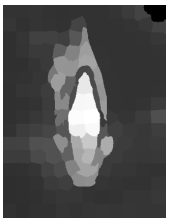
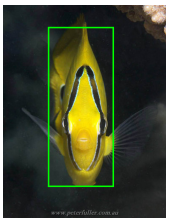
No	Nilai *compactness* pada segmentasi SLIC					
	10	15	20	25	30	35
1.						
2.						
3.						
4.						
5.						
6.						
7.						
8.						
9.						
10.						
11.						
12.						

No	Nilai *compactness* pada segmentasi SLIC					
	10	15	20	25	30	35
13.						
14.						
15.						
16.						
17.						
18.						

Hasil ekstraksi latar depan pada citra ketiga dan keempat dalam tabel 4.8 dengan nilai *compactness* lebih kecil dari 30, masih terdapat sedikit bagian latar belakang yang ikut terekstraksi bersama objeknya. Ketika latar belakang dan objek pada citra yang diuji sangat kontras, nilai *compactness* sebesar 10 sudah cukup untuk melakukan ekstraksi latar depan dengan baik, akan tetapi ketika kontrasnya rendah, membutuhkan nilai *compactness* lebih dari atau sama dengan 30 untuk hasil ekstraksi yang lebih baik. Hasil ekstraksi pada citra dengan objek lebih dari satu tidak sebaik hasil ekstraksi pada citra dengan satu objek. Dari tabel 4.8 hasil ekstraksi dengan nilai *compactness* sebesar 30 dipilih sebagai citra yang akan dibandingkan dengan hasil ekstraksi pada metode Closed-Form Matting. Berikut hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi dengan nilai *compactness* yang diatur menjadi 30 pada tahap segmentasi SLIC.

Tabel 4.9: Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi pada citra dengan satu objek

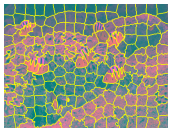
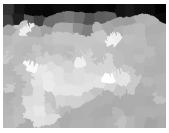
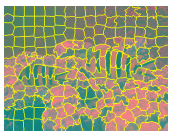
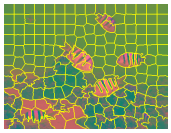
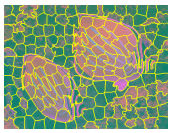
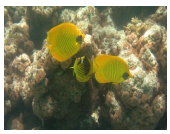
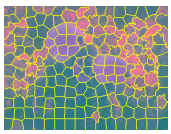
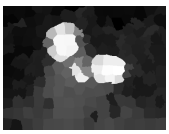
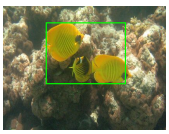
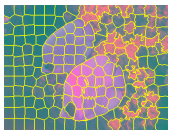
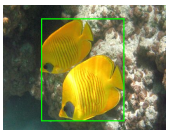
No	Citra masukan	SLIC	*Saliency map*	GrabCut	
				Rectangle	Hasil
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					

No	Citra masukan	SLIC	*Saliency map*	GrabCut	
				Rectangle	Hasil
11					
12					

Tabel 4.9 menunjukkan hasil ekstraksi latar depan pada citra yang hanya mempunyai satu objek di dalamnya. Dalam tabel tersebut, terdapat beberapa hasil ekstraksi latar depan yang bagian tubuh ikannya (seperti ekor dan sirip) tidak ikut terekstraksi dan juga ada hasil inisialisasi citra yang melebar seperti pada citra kesepuluh tetapi GrabCut dapat mengatasi ekstraksinya dengan baik. Secara keseluruhan citra hasil ekstraksi latar depan menggunakan GrabCut yang diautomasi pada citra dengan satu objek yang terdapat pada tabel 4.9 menunjukkan hasil yang cukup baik.

Dapat dilihat dalam tabel 4.9 pada tahap deteksi *saliency* terdapat bagian tubuh ikan yang terdeteksi tetapi tidak terlalu menonjol (terang) hasil *saliency* pada bagian tersebut, hal itu membuat adanya bagian tubuh ikan yang tidak termasuk ke dalam persegi pada inisialisasi metode GrabCut, menyebabkan terdapat bagian tubuh ikan yang tidak ikut terekstraksi. Hasil ekstraksi latar depan pada tabel tersebut masih terdapat sedikit latar belakang yang ikut terekstraksi, hal itu dapat disebabkan oleh beberapa faktor seperti, ukuran persegi dalam inisialisasi metode GrabCut tidak sesuai dengan ukuran besar objek dan kualitas citra masukan juga berpengaruh terhadap hasil ekstraksi. Ketika warna tubuh ikan dan warna latar belakangnya jauh berbeda dan tepi dari objek terlihat jelas (tajam) maka GrabCut dapat lebih mudah mengenali batas-batas dari objeknya itu sendiri sehingga lebih mudah dalam melakukan ekstraksi latar depan.

Tabel 4.10: Hasil ekstraksi latar depan menggunakan metode GrabCut yang diautomasi pada citra dengan objek lebih dari satu

No	Citra masukan	SLIC	*Saliency map*	GrabCut	
				Rectangle	Hasil
1					
2					
3					
4					
5					
6					

Ketika metode ekstraksi ini digunakan untuk mengekstraksi citra dengan objek lebih dari satu, hasil ekstraksinya tidak sebaik dengan hasil ekstraksi pada citra dengan satu objek, hal tersebut disebabkan oleh ketidakcocokan dalam tahap peralihan yang terdapat di antara tahap deteksi *saliency* dan tahap ekstraksi latar depan dengan GrabCut. Dalam penelitian ini, tahap peralihan yang dimaksud adalah *thresholding* adaptif menggunakan metode Otsu dan inisialisasi persegi yang digunakan sebagai data masukan dalam metode GrabCut.

Gambar 4.7: Ilustrasi tahap transisi yang terdapat diantara tahap deteksi *saliency* dan tahap ekstraksi latar depan dengan GrabCut

Thresholding adaptif digunakan untuk mendapatkan citra biner dari citra yang dihasilkan dalam tahap deteksi *saliency*. Dalam citra biner tersebut dapat dibedakan menjadi dua yaitu, piksel yang bernilai 0 sebagai bagian dari latar belakang dan piksel yang bernilai 1 sebagai bagian dari latar depan. Kemudian untuk menentukan besarnya persegi dalam inisialisasi metode GrabCut adalah dengan mencari titik terluar sebelah kiri dan atas bagian putih (piksel yang bernilai 1) pada citra biner dan titik terluar sebelah kanan dan bawah bagian putih pada citra biner dengan kata lain inisialisasi persegi mencakup seluruh piksel yang bernilai 1 pada citra biner yang dihasilkan.

Penyebab hasil ekstraksi latar depan pada citra dengan objek lebih dari satu kurang baik adalah karena hasil deteksi *saliency* menunjukkan objek yang menonjol bukan hanya latar depannya saja melainkan latar belakangnya juga ikut terdeteksi menjadikan piksel yang bernilai 1 atau bagian putih pada citra binernya menyelimuti semua yang terdeteksi dalam deteksi *saliency* dan pelabelan yang dilakukan dalam penelitian ini hanya terdapat satu untuk bagian latar depan oleh karena itu objek-objek yang terdapat pada citra tersebut dihitung sebagai satu objek saja. Seluruh bagian putih pada citra biner masuk ke dalam inisialisasi persegi yang menyebabkan hasil ekstraksinya bukan hanya latar depan saja yang terekstraksi melainkan latar belakangnya juga ikut terekstraksi. Gambar 4.8 adalah potongan kode untuk melakukan ekstraksi latar depan dengan metode GrabCut yang diautomasi menggunakan *saliency map*.

```
import numpy as np
import helper as hp

class GC:
    def __init__(self, img, rect):
        ...
        self.tempMask = np.zeros(self.im.shape[:2], dtype=np.uint8)
        self.tempMask[rect[1]:rect[1]+rect[3],
                      rect[0]:rect[0]+rect[2]]=GROUP_PR_FG['sign']
        self.classify_pixels()
        self.smoothness()
        self.init_GMMs()
        self.run()
        self.mask = np.where((self.tempMask == 1) +
                             (self.tempMask == 3), 255, 0)
                             .astype('uint8')
        self.mask3D = np.zeros_like(self.im)
        for i in range(3):
            self.mask3D[:, :, i] = self.mask.copy()
        self.output()
        self.outBgTransparent = hp.make_transparent_bgd(self.out)

    def smoothness(self):...
    def classify_pixels(self):...
    def init_GMMs(self):...
    def run(self):
        self.assign_GMMs_components()
        self.learn_GMMs()
        self.construct_graph()
        self.estimate_segmentation()

    def output(self):
        self.out = (self.im & self.mask3D)

grabCut = GC(im2,rect)
result = grabCut.outBgTransparent
```

Gambar 4.8: Potongan sampel kode ekstraksi latar depan menggunakan GrabCut

B. Pengujian data masukan dengan metode pembanding

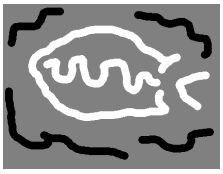
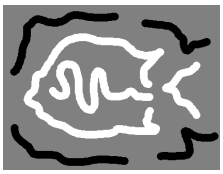
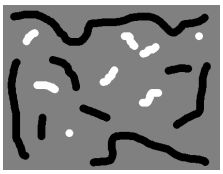
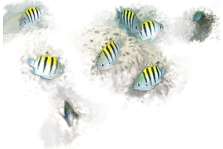
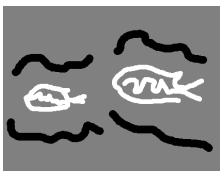
Dalam melakukan ekstraksi latar depan menggunakan metode Closed-Form Matting, diperlukan dua data masukan, yaitu citra asli dan citra *scribbles*. Format *file* citra *scribbles* adalah png. Kriteria citra *scribbles* yang digunakan dapat dilihat pada tabel di bawah ini.

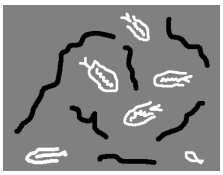
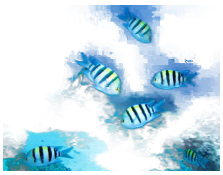
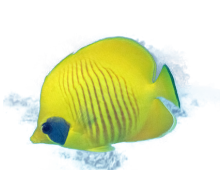
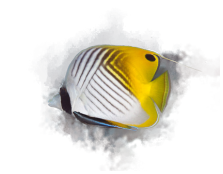
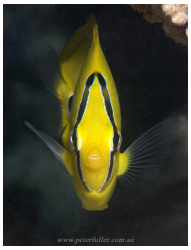
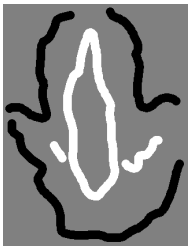
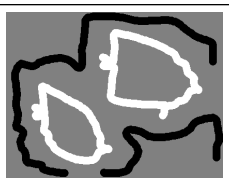
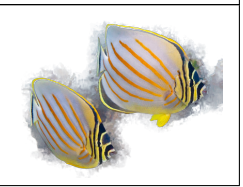
Tabel 4.11: Kriteria citra *scribbles*

No	Nama *file*	Ukuran panjang dan lebar citra (piksel)	Ukuran *file*
1.	Abudefduf1_scribbles.png	420 X 320	6 KB
2.	Abudefduf2_scribbles.png	420 X 320	6 KB
3.	Abudefduf3_scribbles.png	420 X 320	6 KB
4.	Abudefduf4_scribbles.png	420 X 320	7 KB
5.	Abudefduf5_scribbles.png	420 X 320	6 KB
6.	Abudefduf_frontlook_scribbles.png	420 X 320	5 KB
7.	Abudefduf_multiple1_scribbles.png	420 X 320	5 KB
8.	Abudefduf_multiple2_scribbles.png	420 X 320	5 KB
9.	Abudefduf_multiple3_scribbles.png	420 X 320	6 KB
10.	Chaetodon1_scribbles.png	420 X 320	6 KB
11.	Chaetodon2_scribbles.png	420 X 320	6 KB
12.	Chaetodon3_scribbles.png	420 X 320	6 KB
13.	Chaetodon4_scribbles.png	420 X 320	6 KB
14.	Chaetodon5_scribbles.png	420 X 320	5 KB
15.	Chaetodon_frontlook_scribbles.png	420 X 320	5 KB
16.	Chaetodon_multiple1_scribbles.png	420 X 320	6 KB
17.	Chaetodon_multiple2_scribbles.png	420 X 320	5 KB
18.	Chaetodon_multiple3_scribbles.png	420 X 320	5 KB

Hasil ekstraksi latar depan dengan menggunakan metode Closed-Form Matting terbilang sangat baik, namun masih terdapat *noise* di sekitar objeknya dan citra *scribbles* yang dibutuhkan masih dibuat secara manual menggunakan aplikasi *editing* citra digital seperti Paint atau Adobe Photoshop. Tabel 4.12 adalah hasil ekstraksi latar depan menggunakan Closed-Form Matting dan gambar 4.9 adalah potongan kode metode Closed-Form Matting.

Tabel 4.12: Hasil ekstraksi latar depan menggunakan metode Closed-Form Matting

No	Citra asli	Citra *scribbles*	Hasil ekstraksi
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			

No	Citra asli	Citra *scribbles*	Hasil ekstraksi
9.			
10.			
11.			
12.			
13.			
14.			
15.			
16.			

No	Citra asli	Citra *scribbles*	Hasil ekstraksi
17.			
18.			

```
import numpy as np
import scipy.sparse
import scipy.sparse.linalg
from PIL import Image

def CF(image, scribbles):
    i, j, v = matting_laplacian(image)
    h, w = scribbles.shape
    L = scipy.sparse.csr_matrix((v, (i, j)), shape=(w*h, w*h))

    A, b = system_build(L, scribbles)
    alpha = scipy.sparse.linalg.spsolve(A, b).reshape(h, w)

    cutout = np.concatenate([image, alpha[:, :, np.newaxis]], axis=2)
    cutout = np.clip(cutout*255, 0, 255).astype(np.uint8)
    alpha = np.clip(alpha*255, 0, 255).astype(np.uint8)
    return alpha, cutout

im = './input/Abudefduf3.jpg'
scr = './scribbles/Abudefduf3_scribbles.png'
im  = np.array(Image.open(im).convert("RGB"))/255.0
scr = np.array(Image.open(scr).convert(  "L"))/255.0
alpha_matte, cutout = CF(im, scr)
```

Gambar 4.9: Potongan sampel kode Closed-Form Matting

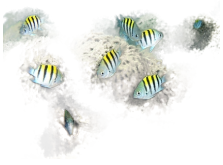
C. *Image Similarity*

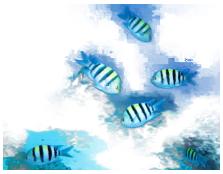
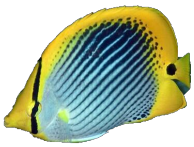
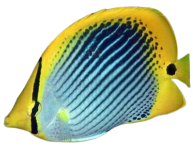
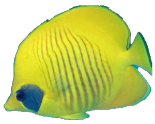
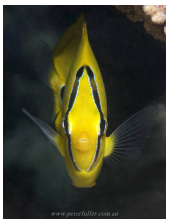
Citra hasil ekstraksi latar depan dari kedua metode yaitu, metode GrabCut menggunakan *saliency map* dan metode Closed-Form Matting, dibandingkan jarak histogram warnanya dari citra referensinya dengan menggunakan Euclidean Distance atau dapat disebut dengan nilai *image similarity*. Kriteria citra referensi yang digunakan dapat dilihat pada tabel di bawah ini.

Tabel 4.13: Kriteria citra referensi

No	Nama *file*	Ukuran panjang dan lebar citra (piksel)	Ukuran *file*
1.	Abudefduf1-removebg.png	420 X 320	136 KB
2.	Abudefduf2-removebg.png	420 X 320	109 KB
3.	Abudefduf3-removebg.png	420 X 320	143 KB
4.	Abudefduf4-removebg.png	420 X 320	129 KB
5.	Abudefduf5-removebg.png	420 X 320	141 KB
6.	Abudefduf_frontlook-removebg.png	420 X 320	73 KB
7.	Abudefduf_multiple1-removebg.png	420 X 320	34 KB
8.	Abudefduf_multiple2-removebg.png	420 X 320	55 KB
9.	Abudefduf_multiple3-removebg.png	420 X 320	45 KB
10.	Chaetodon1-removebg.png	420 X 320	107 KB
11.	Chaetodon2-removebg.png	420 X 320	162 KB
12.	Chaetodon3-removebg.png	420 X 320	106 KB
13.	Chaetodon4-removebg.png	420 X 320	110 KB
14.	Chaetodon5-removebg.png	420 X 320	82 KB
15.	Chaetodon_frontlook-removebg.png	420 X 320	69 KB
16.	Chaetodon_multiple1-removebg.png	420 X 320	139 KB
17.	Chaetodon_multiple2-removebg.png	420 X 320	48 KB
18.	Chaetodon_multiple3-removebg.png	420 X 320	86 KB

Tabel 4.14: Perbandingan hasil ekstraksi latar depan antara metode GrabCut yang diautomasi, Closed-Form Matting dan remove.bg

No	Citra asli	GrabCut yang diautomasi	Closed-Form Matting	Citra referensi
1.				
2.				
3.				
4.				
5.				
6.				
7.				

No	Citra asli	GrabCut yang diautomasi	Closed-Form Matting	Citra referensi
8.				
9.				
10.				
11.				
12.				
13.				
14.				
15.				

No	Citra asli	GrabCut yang diautomasi	Closed-Form Matting	Citra referensi
16.				
17.				
18.				

Citra referensi pada tabel 4.14 dihasilkan oleh sistem *background removal* pada website https://www.remove.bg/ dan tabel 4.15 adalah nilai *image similarity* yang dihasilkan dari citra uji dan citra referensi.

Tabel 4.15: Perbandingan nilai *image similarity*

No	Nama *file*	Nilai *similarity*	
		GrabCut	Closed-Form Matting
1.	Abudefduf1	2,35 x 10^4	27,85 x 10^4
2.	Abudefduf2	3,06 x 10^4	25,37 x 10^4
3.	Abudefduf3	2 x 10^4	30,38 x 10^4
4.	Abudefduf4	2,11 x 10^4	26,53 x 10^4
5.	Abudefduf5	0,29 x 10^4	29,29 x 10^4
6.	Abudefduf_frontlook	0.35 x 10^4	37,25 x 10^4
7.	Abudefduf_multiple1	44,92 x 10^4	43,71 x 10^4
8.	Abudefduf_multiple2	30,42 x 10^4	39,03 x 10^4
9.	Abudefduf_multiple3	24,72 x 10^4	41,32 x 10^4

No	Nama *file*	Nilai *similarity*	
		GrabCut	Closed-Form Matting
10.	Chaetodon1	0,38 x 10^4	29,57 x 10^4
11.	Chaetodon2	0,34 x 10^4	23,75 x 10^4
12.	Chaetodon3	0.19 x 10^4	26,96 x 10^4
13.	Chaetodon4	0.34 x 10^4	29,95 x 10^4
14.	Chaetodon5	0.62 x 10^4	35,86 x 10^4
15.	Chaetodon_frontlook	1,48 x 10^4	37,85 x 10^4
16.	Chaetodon_multiple1	19,52 x 10^4	28,68 x 10^4
17.	Chaetodon_multiple2	1,14 x 10^4	37,59 x 10^4
18.	Chaetodon_multiple3	1,32 x 10^4	32,72 x 10^4

Pada tabel 4.15 dapat dilihat bahwa rata-rata nilai *image similarity* dari hasil ekstraksi menggunakan GrabCut lebih kecil daripada hasil ekstraksi menggunakan Closed-Form Matting. Ketika citra uji adalah citra dengan objek lebih dari satu ikan yang diekstraksi menggunakan metode GrabCut, nilai *image similarity* yang dihasilkan lebih tinggi daripada citra yang hanya memiliki satu objek ikan didalamnya. Nilai *image similarity* pada citra uji yang dihasilkan oleh metode Closed-Form Matting lebih besar, mungkin itu disebabkan hasil ekstraksinya masih terdapat *noise* disekitar objeknya.

Berdasarkan hasil di atas, dapat dilihat bahwa perbedaan jarak histogram antara citra uji yang dihasilkan menggunakan metode GrabCut dan citra referensi jauh lebih kecil daripada perbedaan jarak histogram antara citra uji yang dihasilkan menggunakan Closed-Form Matting dan citra referensi. Dapat diartikan bahwa citra hasil ekstraksi menggunakan GrabCut lebih mendekati atau mirip dengan citra referensinya. Gambar 4.10 adalah potongan kode untuk menghitung nilai *image similarity* antara citra uji dengan citra referensi.

```
import numpy as np

def L2Norm(H1, H2):
    distance = 0
    for i in range(len(H1)):
        distance += (H1[i]-H2[i])**2
    return np.sqrt(distance)

def imageSimilarity(ref_img, test_img):
    H1 = imgHist(ref_img)
    H2 = imgHist(test_img)
    dist_test_ref = L2Norm(H1, H2)
    return ("%.2f" % dist_test_ref

imageSimilarity(ref_im,test_im)
```

Gambar 4.10: Potongan sampel kode *image similarity*

BAB V

PENUTUP

A. Kesimpulan

Dari perancangan, implementasi dan uji coba sistem ekstraksi latar depan dengan metode GrabCut yang diautomasi menggunakan *saliency map*, diperoleh kesimpulan sebagai berikut:

1. Rata-rata nilai *image similarity* yang dihasilkan dengan metode ini adalah 1.13 x 10^4 untuk citra dengan satu objek dan 20.34 x 10^4 untuk citra dengan objek lebih dari satu. Dapat disimpulkan bahwa metode ini mampu mengekstraksi latar depan pada citra dengan satu objek karena nilai *image similarity*-nya cukup kecil dibandingkan dengan citra dengan objek lebih dari satu.

2. Ketika latar depan (*region of interest*) dan latar belakang pada citra uji memiliki kontras yang tinggi, hasil ekstraksinya sangat baik daripada citra uji yang memiliki latar depan dan latar belakang dengan kontras yang rendah.

3. Rata-rata nilai *image similarity* yang dihasilkan dengan metode Closed-Form matting adalah 30.05 x 10^4 untuk citra dengan satu objek dan 37.18 x 10^4 untuk citra dengan objek lebih dari satu. Nilai *image similarity* yang dihasilkan dari metode ini jauh lebih kecil daripada metode Closed-Form Matting, dapat disimpulkan bahwa citra hasil ekstraksi latar depan dengan metode ini lebih baik, karena semakin kecil nilainya maka semakin mirip hasil ekstraksi dari citra uji dengan citra referensinya.

B. Saran

Saran untuk penelitian selanjutnya adalah melakukan perubahan dalam tahap transisi yang terdapat di antara tahap deteksi *saliency* menuju tahap ekstraksi dengan GrabCut seperti, menggunakan metode selain *thresholding* adaptif Otsu dan mengganti metode untuk menemukan letak dan besar persegi pada inisialisasi metode GrabCut atau melakukan peningkatan pada metode GrabCut itu sendiri agar dapat melakukan ekstraksi latar depan pada citra dengan objek lebih dari satu.

DAFTAR PUSTAKA

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., dan Süsstrunk, S. 2012. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282.

Agrawal, P. 2021. A Beginners' Guide to Image Similarity using Python. `https://www.analyticsvidhya.com/blog/2021/03/a-beginners-guide-to-image-similarity-using-python/` [6 Oktober 2021].

Cahskan, A., Koz, A., dan Alatan, A. A. 2015. Hyperspectral superpixel extraction using boundary updates based on optimal spectral similarity metric. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, volume 2015-Novem, pages 1020–1023.

Darmanto, H. 2019. Introduction To Fish Species Based on Otolith Conture Using the Convolutional Neural Network. *Joined jurnal*, 2(1):41–59.

Deng, L. L. 2018. Pre-detection Technology of Clothing Image Segmentation Based on GrabCut Algorithm. *Wireless Personal Communications*, 102(2):599–610.

Dewi, M. P., Silfina, dan Murni, D. 2018. Similarity measurement of segmented image using euclidean distance method. *Advances in Social Science, Education and Humanities Research (ASSEHR)*, 285.

Dung, L.-R., Yang, Y.-M., dan Wu, Y.-Y. 2018. A Hierarchical Grab Cut Image Segmentation Algorithm. *Journal of Computer and Communications*, 06(02):48–55.

Fadllullah, A., Arifin, A. Z., dan Navastara, D. A. 2016. Segmentasi Citra Ikan Tuna Menggunakan Gradient-Barrier Watershed Berbasis Analisis Hierarki Klaster dan Regional Credibility Merging. *Jurnal Buana Informatika*, 7(3):225–234.

Fatimah, F., Laxmi, G. F., dan Eosina, P. 2017. Pengubahan Data Image Ikan Air Tawar ke Data Vektor menggunakan Edge Detection Metode Canny. *Jurnal Riset Pendidikan Matematika*, 9:55–60.

Fauzi, S., Eosina, P., dan Laxmi, G. F. 2019. Implementasi Convolutional Neural Network Untuk Identifikasi Ikan Air Tawar. *Seminar Nasional Teknologi Informasi*, pages 163–167.

Feng, B. dan He, K. 2019. Improved Grab Cut with Human Visual Perception. *2019 IEEE 4th International Conference on Image, Vision and Computing, ICIVC 2019*, (1):50–54.

Gonzalez, R. C. dan Woods, R. E. 2018. *Digital Image Processing, Fourth Edition, Global Edition*. Pearson Education.

Hua, J., Hao, T., Zeng, L., dan Yu, G. 2021. YOLOMask, an instance segmentation algorithm based on complementary fusion network. *Mathematics*, 9(15):1–12.

Kim, K. S., Yoon, Y. J., Kang, M. C., Sun, J. Y., dan Ko, S. J. 2014. An improved GrabCut using a saliency map. *2014 IEEE 3rd Global Conference on Consumer Electronics, GCCE 2014*, pages 317–318.

Kitov, V. dan Ponomareva, L. 2021. Image foreground extraction and its application to neural style transfer. *Information Technologies and Mathematical Methods in Economics and Management*, 2830.

Kumaseh, M. R., Latumakulita, L., Nainggolan, N., dan Citra, S. 2013. Segmentasi Citra Digital Ikan Menggunakan Digital Fish Image Segmentation by Thresholding Method. *Jurnal Ilmiah Sains*, 13.

Levin, A., Lischinski, D., dan Weiss, Y. 2008. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242.

Li, X., Liu, K., dan Dong, Y. 2018. Superpixel-Based Foreground Extraction with Fast Adaptive Trimaps. *IEEE Transactions on Cybernetics*, 48(9):2609–2619.

Liu, Z., Meur, O. L., dan Luo, S. 2013. Superpixel-based saliency detection. *Proc. Int. Workshop Image Analytics. Multimedia Interactive Services (WIAMS)*, pages 1–4.

Muwardi, F. dan Fadlil, A. 2018. Sistem Pengenalan Bunga Berbasis Pengolahan Citra dan Pengklasifikasi Jarak. *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, 3(2):124.

Na, I. S., Chen, Y. J., dan Kim, S. H. 2014. Automatic Segmentation of Product Bottle Label Based on GrabCut Algorithm. *International Journal of Contents*, 10(4):1–10.

Ngurah Sanditya Riantama, G., Nyoman Piarsa, I., dan Made Arya Sasmita, G. 2019. Pengaruh Segmentasi Terhadap Hasil Rotasi Citra Menggunakan Metode Minimum Area Rectangle. *Jurnal Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi)*, 7(2):95.

Noviyanti, P., Kusrini, K., dan Kurniawan, M. P. 2019. Segmentasi Citra Ikan Arwana Super Red Berdasarkan Deteksi Tepi Menggunakan Algoritma Canny. *Jurnal Teknologi Informasi*, 3(2):200.

Prasmatio, R. M., Rahmat, B., dan Yuniar, I. 2020. Deteksi Dan Pengenalan Ikan Menggunakan Algoritma Convolutional Neural Network. *Jurnal Informatika dan Sistem Informasi (JIFoSI)*, 1(2 SE - Articles):510–521.

Pratama, S. P., Rahmat, B., dan Anggraeny, F. T. 2020. Deteksi Ikan Dengan Menggunakan Algoritma Adaboost-SVM. *Jurnal Informatika dan Sistem Informasi (JIFoSI)*, 1(2):458–466.

Ramadhani, M. dan Murti, D. H. 2018. Klasifikasi Ikan Menggunakan Oriented Fast and Rotated Brief (Orb) Dan K-Nearest Neighbor (Knn). *JUTI: Jurnal Ilmiah Teknologi Informasi*, 16(2):115.

R.Dass, Priyanka, dan Devi, S. 2012. Image Segmentation Techniques. *International Journal of Electronis and Communication Technology*, 7109(1):66–70.

Rother, C., Kolmogorov, V., dan Blake, A. 2004. "GrabCut" - Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314.

Saputra, O., Anwari, M. S., dan Herawatiningsih, R. 2019. Keanekaragaman Jenis Ikan Air Tawar Di Sungai Dong Sandar Dan Sungai Rempangi Di Kecamatan Sungai Laur Kabupaten Ketapang. *Jurnal Hutan Lestari*, 7(1):21–31.

Soni, A., Pandey, N., dan Halarnkar, P. 2014. Review on Image Object Extraction. *International Journal of Current Engineering and Technology*, 4(2):864–869.

Wang, X., Li, S., Sui, L., dan Wang, J. 2020. Quick automatic head image matting method based on segmentation and propagation. *Pattern Recognition Letters*, 130:30–37.

Ye, H., Liu, C., dan Niu, P. 2018. Cucumber appearance quality detection under complex background based on image processing. *International Journal of Agricultural and Biological Engineering*, 11(4):171–177.

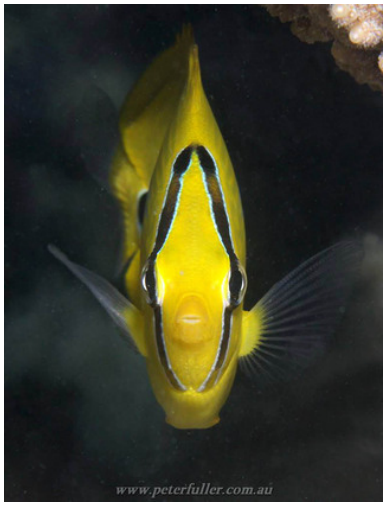
LAMPIRAN

Lampiran 1 Daftar Citra Masukan

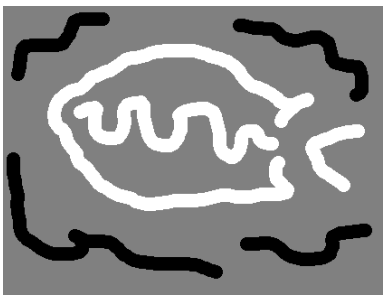
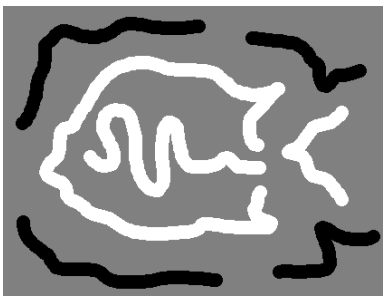
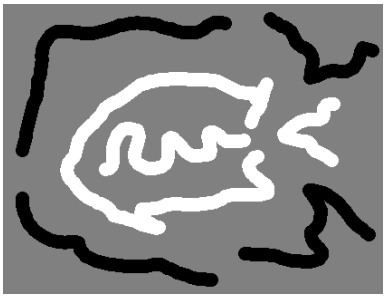
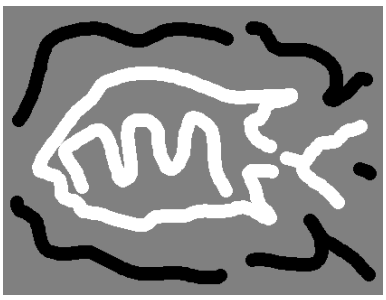
No	Nama file	Citra
1.	Abudefduf1.jpeg	
2.	Abudefduf2.jpeg	
3.	Abudefduf3.jpg	
4.	Abudefduf4.jpg	

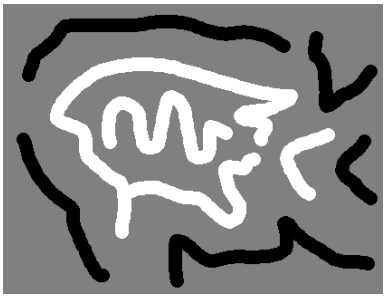
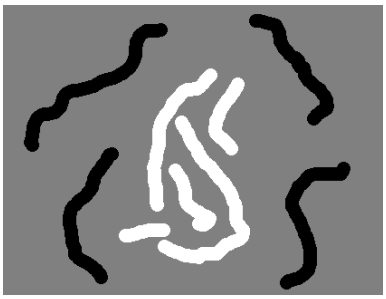
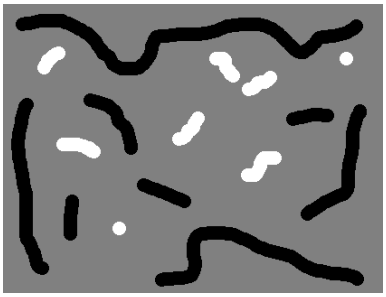
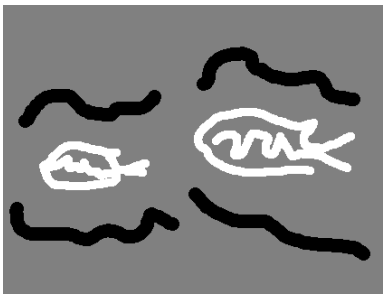
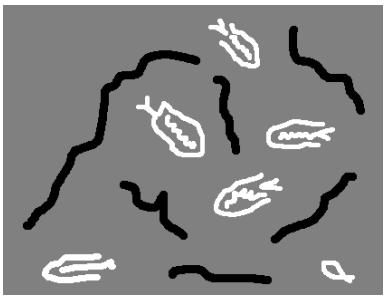
No	Nama file	Citra
5.	Abudefduf5.jpg	
6.	Abudefduf_frontlook.jpg	
7.	Abudefduf_multiple1.jpg	
8.	Abudefduf_multiple2.jpg	
9.	Abudefduf_multiple3.jpg	

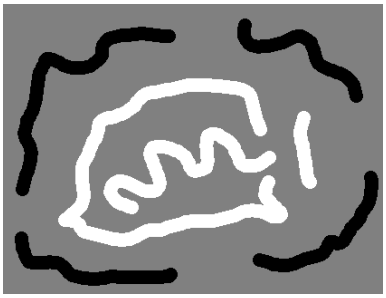
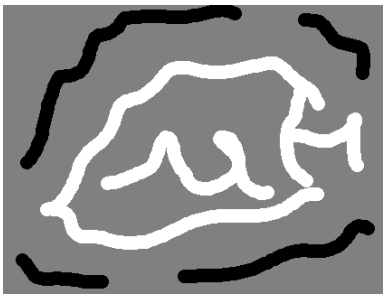
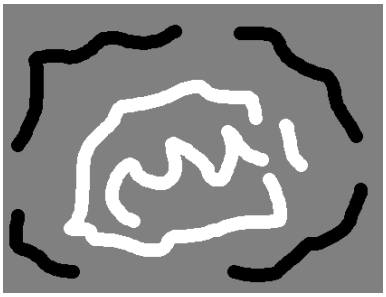
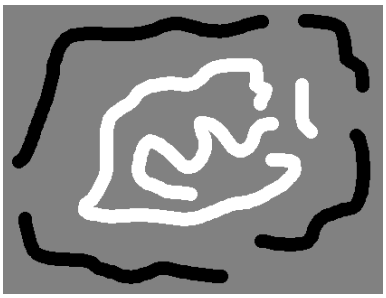
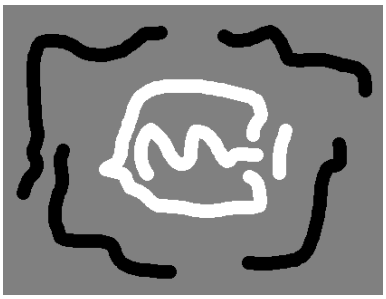
No	Nama file	Citra
10.	Chaetodon1.jpg	
11.	Chaetodon2.jpeg	
12.	Chaetodon3.jpeg	
13.	Chaetodon4.jpeg	
14.	Chaetodon5.jpeg	

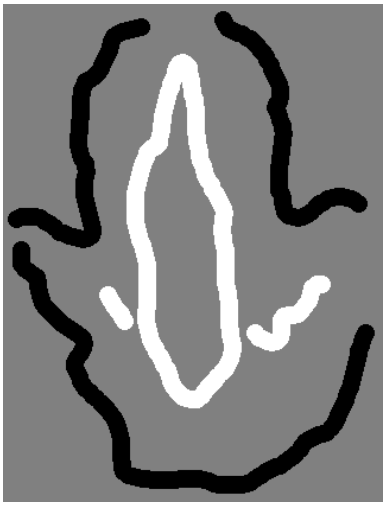
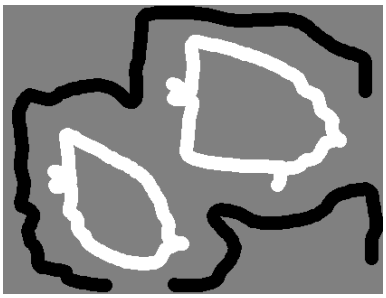
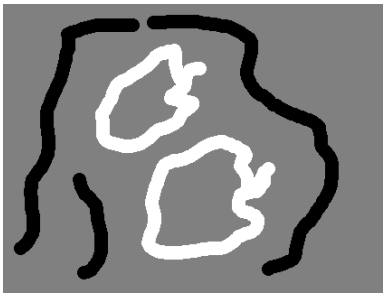
No	Nama file	Citra
15.	Chaetodon_frontlook.jpg	
16.	Chaetodon_multiple1.jpg	
17.	Chaetodon_multiple2.jpg	
18.	Chaetodon_multiple3.jpg	

Lampiran 2 Daftar Citra *Scribbles*

No	Nama file	Citra
1.	Abudefduf1 _scribbles.png	
2.	Abudefduf2 _scribbles.png	
3.	Abudefduf3 _scribbles.png	
4.	Abudefduf4 _scribbles.png	

No	Nama file	Citra
5.	Abudefduf5 _scribbles.png	
6.	Abudefduf_frontlook _scribbles.png	
7.	Abudefduf_multiple1 _scribbles.png	
8.	Abudefduf_multiple2 _scribbles.png	
9.	Abudefduf_multiple3 _scribbles.png	

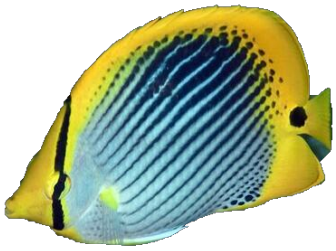
No	Nama file	Citra
10.	Chaetodon1 _scribbles.png	
11.	Chaetodon2 _scribbles.png	
12.	Chaetodon3 _scribbles.png	
13.	Chaetodon4 _scribbles.png	
14.	Chaetodon5 _scribbles.png	

No	Nama file	Citra
15.	Chaetodon_frontlook _scribbles.png	
16.	Chaetodon_multiple1 _scribbles.png	
17.	Chaetodon_multiple2 _scribbles.png	
18.	Chaetodon_multiple3 _scribbles.png	

Lampiran 3 Daftar Citra Hasil Ekstraksi GrabCut yang Diautomasi

No	Nama file	Citra
1.	Abudefduf1 _output.png	
2.	Abudefduf2 _output.png	
3.	Abudefduf3 _output.png	
4.	Abudefduf4 _output.png	

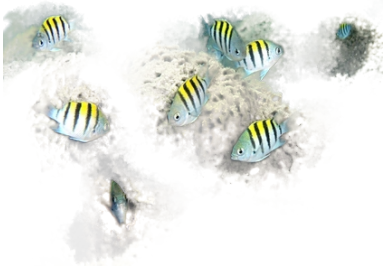
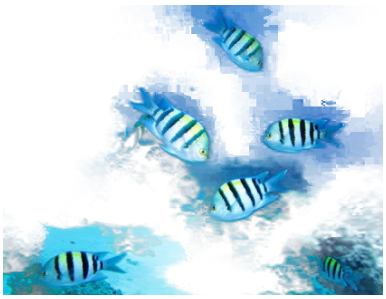
No	Nama file	Citra
5.	Abudefduf5 _output.png	
6.	Abudefduf_frontlook _output.png	
7.	Abudefduf_multiple1 _output.png	
8.	Abudefduf_multiple2 _output.png	
9.	Abudefduf_multiple3 _output.png	

No	Nama file	Citra
10.	Chaetodon1 _output.png	
11.	Chaetodon2 _output.png	
12.	Chaetodon3 _output.png	
13.	Chaetodon4 _output.png	
14.	Chaetodon5 _output.png	

No	Nama file	Citra
15.	Chaetodon_frontlook _output.png	
16.	Chaetodon_multiple1 _output.png	
17.	Chaetodon_multiple2 _output.png	
18.	Chaetodon_multiple3 _output.png	

Lampiran 4 Daftar Citra Hasil Ekstraksi Metode Closed-Form Matting

No	Nama file	Citra
1.	Abudefduf1 _cutout.png	
2.	Abudefduf2 _cutout.png	
3.	Abudefduf3 _cutout.png	
4.	Abudefduf4 _cutout.png	

No	Nama file	Citra
5.	Abudefduf5 _cutout.png	
6.	Abudefduf_frontlook _cutout.png	
7.	Abudefduf_multiple1 _cutout.png	
8.	Abudefduf_multiple2 _cutout.png	
9.	Abudefduf_multiple3 _cutout.png	

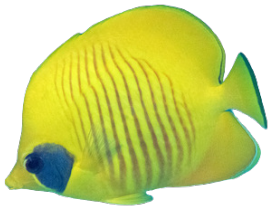
No	Nama file	Citra
10.	Chaetodon1 _cutout.png	
11.	Chaetodon2 _cutout.png	
12.	Chaetodon3 _cutout.png	
13.	Chaetodon4 _cutout.png	
14.	Chaetodon5 _cutout.png	

No	Nama file	Citra
15.	Chaetodon_frontlook _cutout.png	
16.	Chaetodon_multiple1 _cutout.png	
17.	Chaetodon_multiple2 _cutout.png	
18.	Chaetodon_multiple3 _cutout.png	

Lampiran 5 Daftar Citra Hasil Ekstraksi Menggunakan Website remove.bg

No	Nama file	Citra
1.	Abudefduf1 -removebg.png	
2.	Abudefduf2 -removebg.png	
3.	Abudefduf3 -removebg.png	
4.	Abudefduf4 -removebg.png	

No	Nama file	Citra
5.	Abudefduf5 -removebg.png	
6.	Abudefduf_frontlook -removebg.png	
7.	Abudefduf_multiple1 -removebg.png	
8.	Abudefduf_multiple2 -removebg.png	
9.	Abudefduf_multiple3 -removebg.png	

No	Nama file	Citra
10.	Chaetodon1 -removebg.png	
11.	Chaetodon2 -removebg.png	
12.	Chaetodon3 -removebg.png	
13.	Chaetodon4 -removebg.png	
14.	Chaetodon5 -removebg.png	

No	Nama file	Citra
15.	Chaetodon_frontlook -removebg.png	
16.	Chaetodon_multiple1 -removebg.png	
17.	Chaetodon_multiple2 -removebg.png	
18.	Chaetodon_multiple3 -removebg.png	

Lampiran 6 Sampel Kode *Superpixel-based Saliency Detection*

https://github.com/bagusn13/Automatic-GrabCut

```
class saliency_map:
    def __init__(self, img_rgb, segments):
        self.img_rgb = np.asarray(img_rgb,
                                  dtype=np.float64)
        self.height, self.width = self.img_rgb.shape[:2]
        self.numpixel = self.height * self.width
        self.center_of_image = np.asarray([self.width/2,
                                           self.height/2])
        self.d = np.sqrt((self.img_rgb.shape[0]**2) +
                         (self.img_rgb.shape[1]**2))

        self.segments = segments
        self.list_centroid = None
        self.sp_score = None
        self.saliency_map = None
        self.otsu_img = None

        self.sp_saliency_score()
        self.build_saliency_map()
        self.otsu()

    def sp_idx(self, index=True):
        u = np.unique(self.segments)
        return [np.where(self.segments == i) for i in u]

    def similarity_value(self, i, j, d,
                         norm_hist_sp, list_centroid):
        hist_i = norm_hist_sp[i-1]
        hist_j = norm_hist_sp[j-1]
        color_similarity = hist.hist_intersection(
                           hist_i, hist_j)
        D = n_dimensional_euclidean_distance(
```

```
        list_centroid[i-1], list_centroid[j-1])
    spatial_similarity = 1 - (D/d)
    return color_similarity*spatial_similarity

def D_j(self, j, center, list_centroid):
    return (n_dimensional_euclidean_distance(
            list_centroid[j-1], center))

def sp_saliency_score(self):
    (segments_ids, pix_counts) = np.unique(
          self.segments, return_counts=True)

    regions = measure.regionprops(self.segments)
    self.list_centroid = np.array([[i.centroid[1],
                 i.centroid[0]] for i in regions])

    superpixel_list = self.sp_idx(self.segments)
    superpixel = [self.img_rgb[idx]
                 for idx in superpixel_list]
    hist_sp = hist.histogram_each_superpixel(
                                  superpixel)
    norm_hist_sp = hist.normalize_histograms(
                                     hist_sp)

    sal_val = []
    for sp_i in segments_ids:
      sigma_a = 0
      sigma_b = 0
      for sp_j in segments_ids:
        sigma_a += (self.similarity_value(
                    sp_i, sp_j,
                    self.d, norm_hist_sp,
                    self.list_centroid) *
                    self.D_j(sp_j,
                    self.center_of_image,
```

```
                    self.list_centroid))
        sigma_b += self.similarity_value(sp_i, sp_j,
                    self.d, norm_hist_sp,
                    self.list_centroid)
      if sigma_b == 0:
        sal_val.append(0)
      else:
        sal_val.append(sigma_a/sigma_b)

    ssmax = np.max(sal_val)
    ssmin = np.min(sal_val)
    nss_val = (sal_val - ssmax)/(ssmin-ssmax)

    result = []
    for sp_i in segments_ids:
      sigma_a = 0
      sigma_b = 0
      for sp_j in segments_ids:
        sigma_a += (self.similarity_value(
                    sp_i, sp_j,
                    self.d, norm_hist_sp,
                    self.list_centroid) *
                    nss_val[sp_j-1])
        sigma_b += self.similarity_value(
                    sp_i, sp_j,
                    self.d, norm_hist_sp,
                    self.list_centroid)
      if sigma_b == 0:
        result.append(0)
      else:
        result.append(sigma_a/sigma_b)
    self.sp_score = result

  def build_saliency_map(self):
    sa_img = self.segments.copy().astype(float)
```

```
        for i in range(np.amax(self.segments)):
          mask = self.segments == i+1
          sa_img[mask] = self.sp_score[i]
        result = np.interp(sa_img, (sa_img.min(),
                 sa_img.max()), (0, 255))
        self.saliency_map = result.astype("uint8")

    def otsu(self):
        thresh = filters.threshold_otsu(
                 self.saliency_map)
        binary_tmp = self.saliency_map > thresh
        self.otsu_img = binary_tmp.astype(np.uint8)
```

Lampiran 7 Sampel Kode GrabCut

```
import numpy as np
import igraph as ig
from GaussianMixture import GMM
import helper as hp

np.seterr(divide='ignore')
GROUP_BG = {'sign': 0}
GROUP_FG = {'sign': 1}
GROUP_PR_BG = {'sign': 2}
GROUP_PR_FG = {'sign': 3}

class GC:
  def __init__(self, img, rect, gmm_comp=5):
    self.im = img.copy()
    self.img = np.asarray(img, dtype=np.float64)
    self.rows, self.cols, _ = img.shape
    self.tempMask = np.zeros(self.im.shape[:2],
                    dtype=np.uint8)
    self.out = np.zeros(self.im.shape, np.uint8)
```

```
self.tempMask[rect[1]:rect[1] + rect[3],
rect[0]:rect[0] + rect[2]] = GROUP_PR_FG['sign']
self.classify_pixels()

self.gmm_comp = gmm_comp
self.gamma = 50
self.beta = 0

self.bgd_gmm = None
self.fgd_gmm = None
self.comp_idxs = np.empty((self.rows, self.cols),
                  dtype=np.uint32)

self.left_V = np.empty((self.rows,
               self.cols - 1))
self.upleft_V = np.empty((self.rows - 1,
                 self.cols - 1))
self.up_V = np.empty((self.rows - 1, self.cols))
self.upright_V = np.empty((self.rows - 1,
                  self.cols - 1))

self.gc_graph = None
self.gc_graph_capacity = None
self.gc_source = self.cols * self.rows
self.gc_sink = self.gc_source + 1

self.smoothness()
self.init_GMMs()
self.run()

self.mask = np.where((self.tempMask == 1) +
(self.tempMask == 3), 255, 0).astype('uint8')
self.mask3D = np.zeros_like(self.im)
for i in range(3):
self.mask3D[:, :, i] = self.mask.copy()
```

```
    self.output()
    self.outBgTransparent = hp.make_transparent_bgd(
                                self.out)

  def classify_pixels(self):
    self.bgd_indexes = np.where(np.logical_or(
    self.tempMask == GROUP_BG['sign'],
    self.tempMask == GROUP_PR_BG['sign']))
    self.fgd_indexes = np.where(np.logical_or(
    self.tempMask == GROUP_FG['sign'],
    self.tempMask == GROUP_PR_FG['sign']))
    assert self.bgd_indexes[0].size > 0
    assert self.fgd_indexes[0].size > 0

  def init_GMMs(self):
    self.bgd_gmm = GMM(self.img[self.bgd_indexes])
    self.fgd_gmm = GMM(self.img[self.fgd_indexes])

  def smoothness(self):
    _left_diff = self.img[:, 1:] - self.img[:, :-1]
    _upleft_diff = self.img[1:, 1:] -
                   self.img[:-1, :-1]
    _up_diff = self.img[1:, :] - self.img[:-1, :]
    _upright_diff = self.img[1:, :-1] -
                    self.img[:-1, 1:]

    self.beta = np.sum(np.square(_left_diff)) +
                np.sum(np.square(_upleft_diff)) + \
                np.sum(np.square(_up_diff)) + \
                np.sum(np.square(_upright_diff))
    self.beta = 1 / (2 * self.beta / (4 *
    self.cols * self.rows - 3 * self.cols - 3 *
    self.rows + 2))

    self.left_V = self.gamma * np.exp(-self.beta *
```

```
                    np.sum(np.square(_left_diff),
                    axis=2))
    self.upleft_V = self.gamma / np.sqrt(2) *
    np.exp(-self.beta * np.sum(np.square(
    _upleft_diff), axis=2))
    self.up_V = self.gamma * np.exp(-self.beta *
    np.sum(np.square(_up_diff), axis=2))
    self.upright_V = self.gamma / np.sqrt(2) *
    np.exp(-self.beta * np.sum(np.square(
    _upright_diff), axis=2))

  def assign_GMMs_components(self):
    self.comp_idxs[self.bgd_indexes] =
    self.bgd_gmm.which_comp(
    self.img[self.bgd_indexes])
    self.comp_idxs[self.fgd_indexes] =
    self.fgd_gmm.which_comp(
    self.img[self.fgd_indexes])

  def learn_GMMs(self):
    self.bgd_gmm.fit(self.img[self.bgd_indexes],
    self.comp_idxs[self.bgd_indexes])
    self.fgd_gmm.fit(self.img[self.fgd_indexes],
    self.comp_idxs[self.fgd_indexes])

  def construct_graph(self):
    bgd_indexes = np.where(
    self.tempMask.reshape(-1) == GROUP_BG['sign'])
    fgd_indexes = np.where(
    self.tempMask.reshape(-1) == GROUP_FG['sign'])
    pr_indexes = np.where(np.logical_or(
    self.tempMask.reshape(-1) == GROUP_PR_BG['sign'],
    self.tempMask.reshape(-1) == GROUP_PR_FG['sign']))

    edges = []
```

```
self.gc_graph_capacity = []

edges.extend(
list(zip([self.gc_source] * pr_indexes[0].size,
pr_indexes[0])))
_D = - \
np.log(self.bgd_gmm.prob(self.img.reshape(-1, 3)
[pr_indexes]))
self.gc_graph_capacity.extend(_D.tolist())
assert len(edges) == len(self.gc_graph_capacity)

edges.extend(
list(zip([self.gc_sink] * pr_indexes[0].size,
pr_indexes[0])))
_D = - \
np.log(self.fgd_gmm.prob(self.img.reshape(-1, 3)
[pr_indexes]))
self.gc_graph_capacity.extend(_D.tolist())
assert len(edges) == len(self.gc_graph_capacity)

edges.extend(
list(zip([self.gc_source] * bgd_indexes[0].size,
bgd_indexes[0])))
self.gc_graph_capacity.extend([0] *
bgd_indexes[0].size)
assert len(edges) == len(self.gc_graph_capacity)

edges.extend(
list(zip([self.gc_sink] * bgd_indexes[0].size,
bgd_indexes[0])))
self.gc_graph_capacity.extend([9 * self.gamma] *
bgd_indexes[0].size)
assert len(edges) == len(self.gc_graph_capacity)

edges.extend(
```

```
list(zip([self.gc_source] * fgd_indexes[0].size,
fgd_indexes[0])))
self.gc_graph_capacity.extend([9 * self.gamma] *
fgd_indexes[0].size)
assert len(edges) == len(self.gc_graph_capacity)

edges.extend(
list(zip([self.gc_sink] * fgd_indexes[0].size,
fgd_indexes[0])))
self.gc_graph_capacity.extend([0] *
fgd_indexes[0].size)
assert len(edges) == len(self.gc_graph_capacity)

img_indexes = np.arange(self.rows * self.cols,
dtype=np.uint32).reshape(self.rows, self.cols)
mask1 = img_indexes[:, 1:].reshape(-1)
mask2 = img_indexes[:, :-1].reshape(-1)
edges.extend(list(zip(mask1, mask2)))
self.gc_graph_capacity.extend(
self.left_V.reshape(-1).tolist())
assert len(edges) == len(self.gc_graph_capacity)

mask1 = img_indexes[1:, 1:].reshape(-1)
mask2 = img_indexes[:-1, :-1].reshape(-1)
edges.extend(list(zip(mask1, mask2)))
self.gc_graph_capacity.extend(
self.upleft_V.reshape(-1).tolist())
assert len(edges) == len(self.gc_graph_capacity)

mask1 = img_indexes[1:, :].reshape(-1)
mask2 = img_indexes[:-1, :].reshape(-1)
edges.extend(list(zip(mask1, mask2)))
self.gc_graph_capacity.extend(
self.up_V.reshape(-1).tolist())
assert len(edges) == len(self.gc_graph_capacity)
```

```
    mask1 = img_indexes[1:, :-1].reshape(-1)
    mask2 = img_indexes[:-1, 1:].reshape(-1)
    edges.extend(list(zip(mask1, mask2)))
    self.gc_graph_capacity.extend(
    self.upright_V.reshape(-1).tolist())
    assert len(edges) == len(self.gc_graph_capacity)
    assert len(edges) == 4 * self.cols * self.rows -
    3 * (self.cols + self.rows) + 2 + \
    2 * self.cols * self.rows
    self.gc_graph = ig.Graph(self.cols *
    self.rows + 2)
    self.gc_graph.add_edges(edges)

  def estimate_segmentation(self):
    mincut = self.gc_graph.st_mincut(
    self.gc_source, self.gc_sink,
    self.gc_graph_capacity)
    pr_indexes = np.where(np.logical_or(
    self.tempMask == GROUP_PR_BG['sign'],
    self.tempMask == GROUP_PR_FG['sign']))
    img_indexes = np.arange(self.rows * self.cols,
    dtype=np.uint32).reshape(self.rows, self.cols)
    self.tempMask[pr_indexes] = np.where(np.isin(
    img_indexes[pr_indexes], mincut.partition[0]),
    GROUP_PR_FG['sign'], GROUP_PR_BG['sign'])
    self.classify_pixels()

  def run(self):
    self.assign_GMMs_components()
    self.learn_GMMs()
    self.construct_graph()
    self.estimate_segmentation()

  def output(self):
```

```
self.out = (self.im & self.mask3D)
```