Analisis Pemanfaatan Pendeteksian Tepi dan Segmentasi Objek untuk fitur *Image Clipper* pada ponsel Samsung

Jason Rivalino - 13521008
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521008@std.stei.itb.ac.id

***Abstrak*—Makalah ini akan membahas mengenai proses analisis terkait dengan pemanfaatan Pendeteksian Tepi dan Segmentasi Objek untuk pengimplementasian fitur *Image Clipper* yang ada pada ponsel Samsung. Fitur *Image Clipper* merupakan fitur yang terdapat di dalam galeri pada ponsel Samsung yang dapat membantu pengguna untuk memisahkan gambar inti dari latar belakangnya dengan lebih mudah. Pada pengimplementasian fitur ini, terdapat sejumlah proses pengolahan citra seperti pendeteksian tepi dan segmentasi objek serta terdapat pemrosesan lebih lanjut untuk menciptakan hasil pemisahan gambar yang lebih presisi.**

Kata Kunci—Pendeteksian Tepi, Segmentasi Objek, Image Clipper, fitur Samsung

I. PENDAHULUAN

Sebagai salah satu perusahaan ponsel terbesar yang ada di dunia pada saat ini, Samsung tentunya terus melakukan perkembangan secara signifikan pada setiap produk ponsel yang baru dikeluarkannya. Terdapat berbagai inovasi dan kreasi yang dihadirkan untuk menjadikan ponsel Samsung semakin canggih dari waktu ke waktu. Hal ini tentunya bertujuan untuk semakin membantu kenyamanan pengguna dalam menggunakan ponsel sehingga dapat meningkatkan nilai penjualan ponsel Samsung pada pasar industri elektronik secara global.

Ponsel Samsung dengan keluaran terbaru sudah memiliki beragam fitur menarik yang dapat membantu penggunanya dalam menggunakan ponsel ini secara lebih interaktif jika dibandingkan dengan keluaran ponsel terdahulu. Fitur-fitur yang muncul tentunya menyesuaikan dengan kebutuhan pengguna dan kondisi perkembangan teknologi terbaru. Salah satu fitur yang hadir pada ponsel Samsung keluaran terbaru yaitu fitur *Image Clipper* yang terdapat pada galeri yang dapat membantu pengguna dalam mengelola gambar yang dimilikinya.

Fitur *Image Clipper* yang terdapat pada ponsel Samsung hadir mulai dari keluaran versi One UI 5.1 yang diluncurkan pada April 2023[1]. Fitur ini memungkinkan pengguna ponsel Samsung untuk melakukan proses *editing* pada gambar yang terdapat di dalam galeri untuk melakukan pemisahan antara gambar inti yang ingin diekstrak dengan latar belakang yang terdapat di dalam gambar itu sendiri. Gambar yang sudah diekstrak pun kemudian dapat disimpan secara tersendiri dan dibagikan kepada pengguna lain ataupun dibagikan melalui aplikasi lain. Adanya fitur *Image Clipper* dapat membantu pengguna untuk melakukan proses *editing* pemisahan objek pada gambar secara lebih mudah tanpa memerlukan bantuan aplikasi tambahan lainnya.

Dalam proses pengimplementasian fitur *Image Clipper* yang ada pada ponsel Samsung, terdapat sejumlah proses pengolahan citra yang diterapkan untuk dapat melakukan pemisahan antara lain pendeteksian tepi yang kemudian digunakan untuk melakukan segmentasi objek beserta adanya pemrosesan lebih lanjut untuk membantu proses pemisahan gambar dengan akurasi yang lebih baik. Analisis tahapan proses implementasi inilah yang akan menjadi bahasan utama pada makalah ini.

II. LANDASAN TEORI

A. Pendeteksian Tepi *(Edge Detection)*

Tepi (*edge*) yang terdapat di dalam citra merupakan perubahan pada nilai keabuan yang terjadi dalam jarak yang singkat. Pendeteksian tepi umumnya digunakan di dalam citra untuk meningkatkan penampakan garis batas atau objek di dalam citra dan untuk mengenali objek yang terdapat di dalam citra[2].

Pendeteksian tepi merupakan bagian dari proses analisis citra yang bertujuan untuk mengidentifikasi objek yang terdapat di dalam citra[2]. Proses analisis citra pada umumnya terdiri atas tiga tahapan yaitu.

1. Ekstraksi ciri: tahapan pendeteksian keberadaan tepi dari objek di dalam citra.
2. Segmentasi: proses mereduksi citra menjadi bagian-bagian *region* tertentu.
3. Klasifikasi: pemetaan segmen-segmen yang berbeda ke dalam kelas objek yang berbeda juga.

Dalam proses pendeteksian tepi, terdapat berbagai jenis operator yang dapat diimplementasikan untuk melakukan pendeteksian antara lain sebagai berikut.

1. Operator Gradien
   Pendeteksian tepi dengan operator Gradien akan memanfaatkan pendekatan kalkulus diferensial dengan perhitungan kemiringan fungsi memanfaatkan turunan pertama. Hasil turunan parsial yang didapat akan berbentuk dua buah *mask* konvolusi sebagai berikut[2].

$$G_x = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \quad G_y = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

2. Operator Laplace
   Pendeteksian tepi dengan operator Laplace akan memanfaatkan operator turunan kedua untuk mendeteksi tepi yang lebih akurat, khususnya pada tepi curam. Bentuk umum dari *mask* konvolusi yang didapatkan dari operator Laplace adalah sebagai berikut[2].

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

3. Operator LoG (*Laplacian of Gaussian*)
   Pendeteksian tepi dengan operator LoG secara umum mirip dengan operator Laplace. Namun, sebelum dilakukan operator Laplace, terdapat proses yaitu penghalusan dengan fungsi Gaussian yang bertujuan untuk mengurangi adanya kemunculan tepi palsu.

4. Operator Sobel
   Pendeteksian tepi dengan operator Sobel akan menggunakan nilai konstanta c=2 dengan bentuk umum *mask* konvolusi sebagai berikut[3].

$$S_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

5. Operator Prewitt
   Pendeteksian tepi dengan operator Prewitt akan menggunakan nilai konstanta c=1 dengan bentuk umum *mask* konvolusi sebagai berikut[3].

$$P_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad P_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

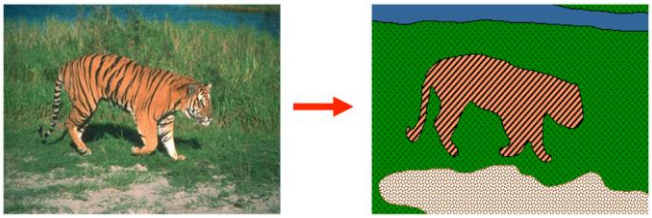
6. Operator Roberts
   Pendeteksian tepi dengan operator Roberts akan menggunakan bentuk umum *mask* konvolusi sebagai berikut[3].

$$R_+ = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad R_- = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

7. Operator Canny
   Pendeteksian tepi dengan operator Canny dapat menghasilkan tepi dengan ketebalan sebesar 1 *pixel*. Terdapat beberapa tahapan dalam implementasi operator Canny antara lain sebagai berikut.
   a. Menghaluskan citra dengan menggunakan penapis Gaussian.
   b. Menghitung gradien dan arah gradien setiap *pixel* dengan salah satu operator antara Sobel, Prewitt, dan Roberts.
   c. Menentukan kondisi berdasarkan nilai mutlak gradien suatu *pixel*. Jika melebihi nilai ambang T, maka *pixel* tersebut termasuk ke dalam *pixel* tepi[3].

B. Segmentasi Citra

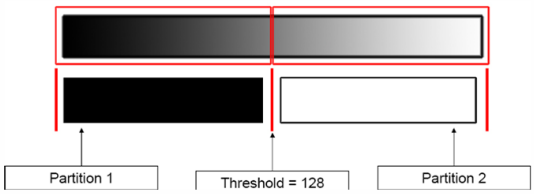
Segmentasi Citra merupakan proses untuk mempartisi citra menjadi sejumlah bagian *region* yang setiap bagiannya terdiri atas sekumpulan *pixel* yang saling terhubung satu sama lainnya. Tujuan dari adanya proses segmentasi citra yaitu untuk membagi citra menjadi segmen yang berbeda-beda ataupun untuk memisahkan objek dengan latar belakang sehingga pemrosesan hanya akan dilakukan pada segmen tertentu di dalam citra[4].



Gambar 2.1 Contoh Segmentasi Citra
(Sumber: https://ai.stanford.edu/~syyeung/cvweb/tutorial3.html)

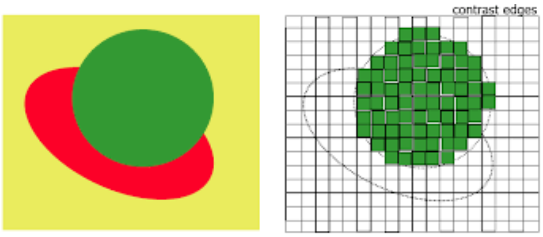
Terdapat dua kelompok dari pendekatan metode yang dapat diimplementasikan dalam segmentasi citra antara lain sebagai berikut.

1. Diskontinuitas: partisi citra berdasarkan pendeteksian tepi[4].
2. *Similarity*: partisi citra berdasarkan kemiripan area menurut properti yang ditentukan. Terdapat beberapa jenis metode segmentasi yang menggunakan jenis pendekatan ini antara lain.
   a. Pengambangan (*thresholding*)
      Metode untuk pemisahan citra berdasarkan batas nilai ambang T yang dapat ditentukan dengan tiga teknik yaitu.
      i. Pengambangan global
         Penentuan nilai ambang berdasarkan keseluruhan nilai-nilai *pixel*.
      ii. Pengambangan lokal
         Penentuan nilai ambang berdasarkan pada *pixel-pixel* yang saling bertetangga.
      iii. Pengambangan adaptif
         Kondisi nilai ambang akan mengalami perubahan secara dinamis bergantung pada kondisi pencahayaan di dalam citra[4].



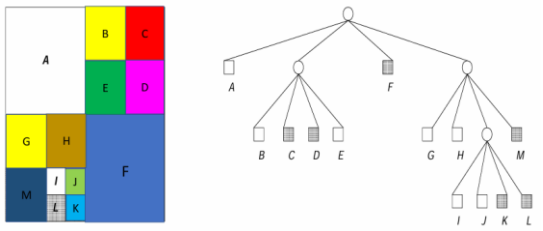
Gambar 2.2 Implementasi metode pengambangan pada segmentasi citra
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/22-Segmentasi-Citra-Bagian1-2024.pdf)

b. *Region Growing*
Metode untuk pemisahan citra dengan mengelompokkan *pixel* menjadi *sub-region* yang kemudian akan bertumbuh menjadi *region* yang lebih besar[4].



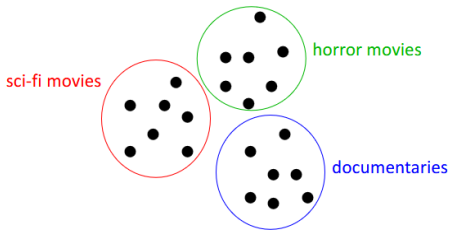
Gambar 2.3 Implementasi metode *Region Growing* pada segmentasi citra
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/22-Segmentasi-Citra-Bagian1-2024.pdf)

c. *Split and merge*
Metode untuk pemisahan citra dengan menggunakan algoritma *divide and conquer* untuk membagi citra menjadi sejumlah *region* yang bersifat *disjoint*. Setelahnya akan dilakukan proses penggabungan untuk *region* tetangga yang bersifat homogen[5].



Gambar 2.4 Implementasi metode *Split and merge* pada segmentasi citra
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/23-Segmentasi-Citra-Bagian2-2024.pdf)

d. *Clustering*
Metode untuk pemisahan citra dengan mengelompokkan (*clustering*) pada berbagai titik yang memiliki kemiripan ke dalam jenis kelompok yang sama. Pada segmentasi citra, titik-titik akan dinyatakan sebagai vektor fitur yang dikelompokkan ke dalam k *cluster*[5].



Gambar 2.5 Implementasi metode *Clustering* pada segmentasi citra
(Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/23-Segmentasi-Citra-Bagian2-2024.pdf)

C. Samsung

Samsung merupakan sebuah perusahaan yang berasal dari Korea Selatan dan bergerak dalam bidang penyediaan perangkat elektronik. Skala Perusahaan Samsung merupakan salah satu yang terbesar di dunia dalam bidang industri elektronik. Perusahaan Samsung sendiri didirikan pada tanggal 1 Marer 1938 oleh Byung-cull Lee di Taegu. Samsung awalnya bergerak dalam industri ekspor barang dagangan (*trading*) dan kemudian merambah ke dalam bidang industry lainnya dengan salah satu yang terbesar yaitu dalam bidang Samsung elektronik[6].

Gambar 2.6 Logo Perusahaan Samsung
(Sumber: https://www.samsung.com/id/about-us/brand-identity/logo/)

Samsung pertama kali mengeluarkan ponsel *Mobile* pada dekade 90-an dan terus mengeluarkan berbagai jenis ponsel yang mengikuti kondisi perkembangan zaman. Kesuksesan Samsung pada industri *smartphone* berawal pada tanggal 27 April 2009 ketika Samsung meluncurkan ponsel *Android* pertamanya yaitu Samsung i7500 yang kemudian menjadi awal dari kesuksesan untuk ponsel-ponsel Samsung selanjutnya[6].

Pada saat ini, perkembangan ponsel Samsung mengalami perkembangan yang sangat pesat dan sudah dikenal sebagai produsen ponsel terbesar di dunia. Ponsel *Samsung Android* banyak diminati oleh konsumen dikarenakan sejumlah faktor seperti desain ponsel yang elegan dan mewah, harga yang cukup sesuai, hingga kelengkapan spesifikasi dan fitur-fitur yang terdapat di dalam ponsel yang mengikuti dengan perkembangan zaman[6].

D. *Image Clipper*

Image Clipper merupakan salah satu fitur yang dihadirkan pada ponsel Samsung sebagai inovasi dalam pengolahan gambar yang dimiliki oleh pengguna. Fitur ini mulai

diperkenalkan pada keluaran versi One UI 5.1 yang diluncurkan pada April 2023[1]. Pada fitur ini, pengguna dapat melakukan pengolahan gambar yang terdapat di dalam galeri ponsel untuk memudahkan pengguna dalam melakukan pemisahan antara gambar inti yang ingin diekstrak dengan latar belakang yang terdapat dalam gambar.

Gambar 2.7 Fitur *Image Clipper* yang terdapat pada ponsel Samsung
(Sumber: https://www.sammobile.com/news/work-image-clipper-magic-samsung-galaxy-device/)

Hasil gambar yang sudah berhasil diekstrak kemudian dapat disimpan dalam bentuk foto tersendiri yang terpisah di dalam galeri. Selain itu, gambar juga dapat dibagikan kepada pengguna ponsel lainnya ataupun disimpan pada aplikasi lain. Contohnya yaitu gambar hasil ekstraksi dapat dibagikan sebagai stiker pada grup Whatsapp ataupun menyimpan gambar pada Samsung Notes[1].

Penggunaan fitur ini pada galeri ponsel dapat dilakukan dengan memilih gambar yang terdapat di dalam galeri kemudian tahan pada objek gambar yang ingin dipotong. Sistem ponsel kemudian akan otomatis bekerja untuk mengidentifikasi objek dalam gambar dan melakukan pemotongan dari latar belakang[1].

III. Penerapan Pendeteksian Tepi dan Segmentasi Objek pada Fitur *Image Clipper*

Implementasi yang akan dilakukan untuk fitur *Image Clipper* akan berupa simulasi yang memanfaatkan dua pendekatan yaitu dengan memanfaatkan pendeteksitan tepi (diskontinuitas) dan kemiripan objek (*similarity*). Adapun tujuan dari penerapan fitur *Image Clipper* melalui dua pendekatan yaitu untuk membandingkan hasil yang didapatkan untuk dilakukan analisis terkait pemilihan proses ekstraksi gambar yang lebih baik.

Pada implementasi fitur *Image* Clipper dengan pendeteksian tepi, implementasi hanya akan dilakukan dengan memanfaatkan operator Canny dikarenakan operator ini memiliki hasil keluaran dan akurasi pendeteksian tepi yang paling baik jika dibandingkan dengan operator lainnya. Hal ini terjadi karena seperti penjelasan sebelumnya bahwa operator Canny dapat menghasilkan pendeteksian tepi dengan ukuran ketebalan sebesar satu *pixel* saja. Implementasi untuk fitur *Image Clipper* dengan pendeteksian tepi akan dilakukan dengan menggunakan bahasa pemrograman Matlab dan akan langsung memanfaatkan *library* edge yang sudah tersedia.

Sedangkan untuk implementasi fitur *Image Clipper* dengan kemiripan objek, implementasi hanya akan dilakukan dengan menggunakan salah satu metode yaitu dengan metode *Clustering* yang menerapkan pengelompokan titik berdasarkan kemiripan dengan memanfaatkan salah satu algoritma *clustering* yang paling popular yaitu *K-Means Clustering*. Adapun untuk implementasi *Image Clipper* dengan kemiripan onjek akan dilakukan dengan menggunakan bahasa pemrograman Python.

A. Implementasi *Image Clipper* dengan metode pendeteksian tepi

Proses untuk implementasi *Image Clipper* dengan metode pendeteksian tepi dilakukan dengan beberapa tahapan proses sebagai berikut.

1. Melakukan pemilihan gambar yang terdapat di dalam folder untuk dilakukan pemrosesan.
2. Menerapkan pendeteksian tepi dengan operator Canny untuk mendapatkan kondisi gambar yang telah mengandung tepian.
3. Segmentasi objek kemudian dilakukan dengan mengubah citra yang telah mengandung tepian ke dalam citra biner untuk kemudian dilakukan pengisian dengan metode *closing* dan pembersihan dengan metode *opening*.
4. Pembuatan mask berdasarkan segmentasi objek dan dikembalikan ke dalam gambar asli untuk ekstraksi objek berdasarkan segmentasi.

Berikut merupakan contoh kode untuk implementasi dari fitur *Image Clipper* dengan metode pendeteksian tepi menggunakan bahasa pemrograman Matlab.

select_image.m (Pemilihan gambar)

```
function filename = select_image()
    % Function to open a file explorer and select an image file
    % Output:
    %   filename - full path of the selected image file

    % File types filter for common image formats
    [file, path] = uigetfile({'*.jpg;*.jpeg;*.png;*.bmp;*.tiff', 'Image Files (*.jpg, *.jpeg, *.png, *.bmp, *.tiff)';
                              '*.*', 'All Files (*.*)'}, ...
                             'Select an Image File');

    % Check if the user canceled the selection
    if isequal(file, 0)
        error('No file was selected.');
    end

    % Construct the full file path
    filename = fullfile(path, file);
end
```

Gambar 3.1 *Screenshoot Source Code* Program untuk select_image.m
(Sumber: Dokumentasi Penulis)

read_image.m (Pembacaan gambar)

```
function img = read_image(filename)
    % Membaca gambar dengan nama spesifik
    fprintf('Membaca gambar input dari file: %s\n', filename);
    img = imread(filename);
end
```

Gambar 3.2 *Screenshoot Source Code* Program untuk

read_image.m
(Sumber: Dokumentasi Penulis)

canny_edge_detection.m (Pendeteksian Tepi Canny)

```
function image_edge_detection = canny_edge_detection(img)
    % Konversi Image ke Grayscale
    if size(img, 3) == 3
        img = rgb2gray(img);
    end

    % Mendeteksi tepi menggunakan pilihan metode
    image_edge_detection = edge(img, 'canny');

    % Change to double
    image_edge_detection = double(image_edge_detection);
```

Gambar 3.3 *Screenshoot Source Code* Program untuk canny_edge_detection.m
(Sumber: Dokumentasi Penulis)

segmentation_using_edge.m (Segmentasi gambar)

```
function image_segmentation_result = segmentation_using_edge(img, image_edge_detection, radius, min_pixel_area)
    % Binerisasi citra hasil deteksi tepi
    binarize_result = imbinarize(image_edge_detection, graythresh(image_edge_detection));

    % Operasi closing pixel
    cleaned_edges = imclose(binarize_result, strel('disk', radius));

    % Operasi untuk menghilangkan border
    cleaned_edges = imclearborder(cleaned_edges);

    % Operasi untuk mengisi lubang
    cleaned_edges = imfill(cleaned_edges, 'holes');

    % Operasi opening pixel
    cleaned_edges = imopen(cleaned_edges, strel('disk', radius));

    % Operasi untuk menghilangkan region kecil
    cleaned_edges = bwareaopen(cleaned_edges, min_pixel_area);

    % Hasil akhir segmentasi objek
    largest_region_mask = cleaned_edges;

    % Membuat mask
    mask = uint8(largest_region_mask);

    % Mengalikan mask dengan citra asli
    image_segmentation_result = img .* mask;
end
```

Gambar 3.4 *Screenshoot Source Code* Program untuk canny_edge_detection.m
(Sumber: Dokumentasi Penulis)

main.m (Program utama)

```
% Pilih dan load program
filename = select_image();
img = read_image(filename);

% Deteksi tepi pada citra
image_edge_detection = canny_edge_detection(img);

% Melakukan segmentasi objek berdasarkan hasil deteksi tepi
image_segmentation_result = segmentation_using_edge(img, image_edge_detection, 2, 4000);

% Menampilkan hasil sebelum dan sesudah deteksi tepi
figure;
subplot(1, 3, 1);
imshow(img);
title("Gambar Asli");

subplot(1, 3, 2);
imshow(image_edge_detection);
title("Gambar Hasil Deteksi Tepi");

subplot(1, 3, 3);
imshow(image_segmentation_result);
title("Gambar Hasil Segmentasi Objek");
```

Gambar 3.5 *Screenshoot Source Code* Program untuk main.m
(Sumber: Dokumentasi Penulis)

Pada implementasi *Image Clipper* dengan pendeteksian tepi, terdapat sejumlah parameter penentu yang diperlukan untuk menghasilkan segmentasi gambar secara lebih akurat. Beberapa diantaranya adalah sebagai berikut.

1. Radius: ukuran seberapa besar jarak untuk proses penutupan (*closing*) dan pembukaan (*opening*). Pada contoh disini digunakan angka radius sebesar 2.
2. Strel: salah satu jenis morfologi yang digunakan untuk proses penutupan (*closing*) dan pembukaan (*opening*).
3. Pixel area minimal: area pixel minimal yang akan dihilangkan pada proses pembukaan (*opening*). Pada contoh disini digunakan angka sebesar 4000.

Berikut merupakan contoh hasil yang didapat dari proses segmentasi objek dengan pemanfaatan pendeteksian tepi.

Gambar 3.6 *Screenshoot* pengujian *image clipper* untuk gambar koin dengan pendeteksian tepi
(Sumber: Dokumentasi Penulis)

Gambar 3.7 *Screenshoot* pengujian *image clipper* untuk gambar cameramen dengan pendeteksian tepi
(Sumber: Dokumentasi Penulis)

B. Implementasi *Image Clipper* dengan metode *Clustering*

Proses untuk implementasi *Image Clipper* dengan metode *Clustering* akan dilakukan dengan beberapa tahapan proses sebagai berikut.

1. Melakukan pemilihan gambar yang terdapat di dalam folder untuk dilakukan pemrosesan.
2. Melakukan konversi gambar ke dalam format sebelum dilakukan pemrosesan.
3. Menentukan kriteria konvergensi dan jumlah *cluster* yang diinginkan untuk proses *Image Clipper*
4. Menentukan *threshold* berdasarkan ukuran *mask* pada *Cluster*.
5. Pembuatan *mask* untuk tiap *cluster* dengan ukuran yang lebih kecil dari *threshold*.
6. Pengubahan *mask* ke gambar yang sebenarnya untuk ditampilkan.

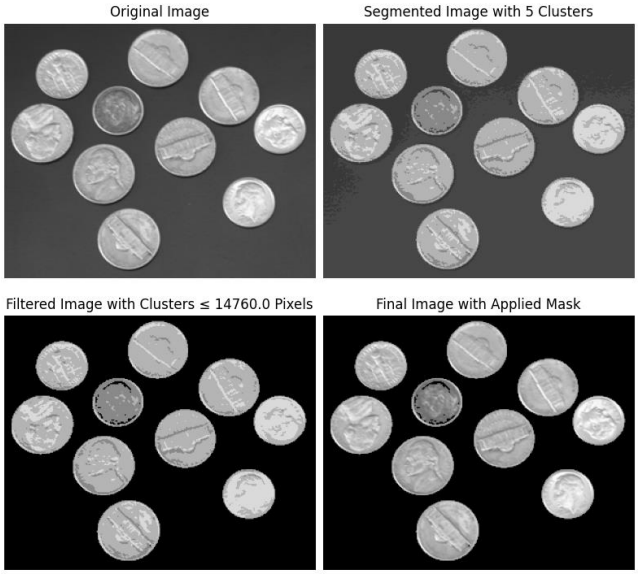
Berikut merupakan contoh kode untuk implementasi dari fitur *Image Clipper* dengan metode *Clustering* menggunakan bahasa pemrograman Python.

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from tkinter import Tk
from tkinter.filedialog import askopenfilename

# File dialog untuk memilih gambar
def choose_image():
    Tk().withdraw()  # Hide the main tkinter window
    file_path = askopenfilename(filetypes=[("Image Files", "*.jpg;*.jpeg;*.png;*.bmp")])
    return file_path

# Pemilihan gambar
image_path = choose_image()
if not image_path:
    print("No image selected!")
    exit()

# Load dan konversi gambar ke RGB
img = cv2.imread(image_path)
image = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Reshape gambar ke format 2D array dan konversi ke float32
pixel_vals = image.reshape((-1, 3))
pixel_vals = np.float32(pixel_vals)
# Menentukan kriteria konvergensi dan jumlah cluster
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 100, 0.2)

# Jumlah cluster yang diinginkan
k = 5
_, labels, centers = cv2.kmeans(pixel_vals, k, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

# Konversi nilai centroid ke tipe data integer
centers = np.uint8(centers)
segmented_data = centers[labels.flatten()]
segmented_image = segmented_data.reshape(image.shape)

# Hitung ukuran rata-rata cluster
threshold_size = np.mean([np.sum(labels == i) for i in range(k)])

# Filter cluster berdasarkan ukuran
filtered_image = np.zeros_like(image)

for i in range(k):
    # Membuat mask untuk setiap cluster
    mask = (labels.flatten() == i).reshape(image.shape[:2])

    # Menghitung ukuran cluster
    cluster_size = np.sum(mask)

    if cluster_size <= threshold_size:
        # Assign cluster untuk cluster yang lebih kecil dari threshold
        filtered_image[mask] = centers[i]

    # Visualisasi cluster
    cluster_image = np.zeros_like(image)
    cluster_image[mask] = centers[i]

# Buat mask dari filtered image
mask = np.any(filtered_image > 0, axis=-1).astype(np.uint8)

# Implementasi mask ke gambar asli
result_image = img * mask[..., np.newaxis]

# Konversi hasil ke format RGB
result_image_rgb = cv2.cvtColor(result_image, cv2.COLOR_BGR2RGB)
# Menampilkan hasil
plt.figure(figsize=(12, 12))

plt.subplot(2, 2, 1)
plt.imshow(image)
plt.title("Original Image")
plt.axis("off")

plt.subplot(2, 2, 2)
plt.imshow(segmented_image)
plt.title(f"Segmented Image with {k} Clusters")
plt.axis("off")

plt.subplot(2, 2, 3)
plt.imshow(filtered_image)
plt.title(f"Filtered Image with Clusters ≤ {threshold_size} Pixels")
plt.axis("off")

plt.subplot(2, 2, 4)
plt.imshow(result_image_rgb)
plt.title("Final Image with Applied Mask")
plt.axis("off")

plt.tight_layout()
plt.show()
```

Gambar 3.8 *Screenshoot Source Code* Program untuk segmentation_using_clustering.py
(Referensi: https://www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/)

Pada implementasi *Image Clipper* dengan pendeteksian tepi, terdapat parameter yaitu nilai k untuk menentukan jumlah *Cluster* yang akan terbentuk dalam pemrosesan program. Pada program ini ditentukan angka *default* untuk jumlah nilai k adalah 5.

Berikut merupakan contoh hasil yang didapat dari proses segmentasi objek dengan pemanfaatan *Clustering*.



Gambar 3.9 *Screenshoot* pengujian *image clipper* untuk gambar koin dengan *Clustering*
(Sumber: Dokumentasi Penulis)

Gambar 3.10 *Screenshoot* pengujian *image clipper* untuk gambar kameramen dengan *Clustering*
(Sumber: Dokumentasi Penulis)

C. Analisis Perbandingan Metode Implementasi *Image Clipper*

Berdasarkan hasil penerapan yang ada diatas, secara umum kedua metode diatas sudah bisa untuk menampilkan implementasi terkait dengan fitur *Image Clipper* dalam melakukan segmentasi objek. Implementasi yang ada sudah menampilkan hasil segmentasi yang cukup sesuai untuk gambar dengan latar belakang polos seperti contoh koin, tetapi masih belum terlalu bagus untuk segmentasi dengan latar belakang yang ramai seperti contoh kameramen.

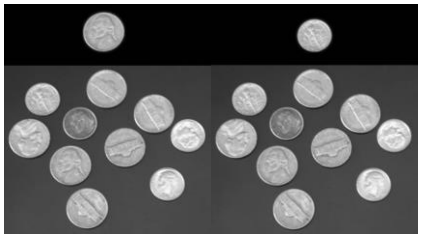
Pada implementasi segmentasi objek dengan pendeteksian tepi, terdapat banyak parameter penentu untuk dapat menghasilkan gambar hasil segmentasi yang sesuai seperti radius, jenis *morphological*, hingga minimal pixel area. Sedangkan, untuk implementasi dengan *Clustering*, parameter yang diperlukan hanya nilai k saja yang menentukan banyaknya jumlah *cluster* yang akan dibentuk.

Meskipun sudah bisa menampilkan hasil untuk segmentasi objek, hasil yang didapat dari implementasi dengan kedua metode ini masih belum terlalu bagus jika dibandingkan dengan fitur *Image Clipper* yang sudah diimplementasikan pada ponsel Samsung. Hal ini tentunya dikarenakan dalam pemrosesan gambar yang dilakukan pada Samsung, sudah terdapat berbagai implementasi pengolahan gambar yang lebih canggih dan bahkan memanfaatkan AI untuk pendeteksian gambar yang ingin diekstrak sehingga sudah menghasilkan akurasi untuk ekstrak gambar yang lebih baik.

Sebagai contoh untuk perbandingan *Image Clipper* yang terdapat pada ponsel Samsung untuk gambar koin dan kameramen, sudah didapatkan hasil seperti berikut.

Gambar 3.11 *Image Clipper* untuk gambar kameramen pada ponsel Samsung
(Sumber: Dokumentasi Penulis)

Untuk gambar koin, bahkan sudah bisa dilakukan ekstraksi untuk tiap koin secara terpisah seperti berikut.



Gambar 3.11 *Image Clipper* untuk gambar koin pada ponsel Samsung
(Sumber: Dokumentasi Penulis)

Berdasarkan hasil yang didapat pada ponsel Samsung, hasil ekstraksi yang ada sudah sangat presisi dan sesuai dengan kebutuhan pengguna ponsel untuk dapat mengekstraksi gambar. Hal ini tentunya menjadi saran untuk perbaikan dan pengembangan dari pengerjaan makalah ini agar dapat mengimplementasikan fitur untuk *Image Clipper* dengan lebih baik.

IV. KESIMPULAN DAN SARAN

A. Kesimpulan

Penerapan pendeteksian tepi dan metode *Clustering* dapat digunakan untuk melakukan pemrosesan pada gambar untuk menerapkan fitur *Image Clipper* seperti yang terdapat pada ponsel Samsung untuk melakukan ekstraksi terhadap objek tertentu yang ada di dalam gambar. Hasil ekstraksi yang didapat pun kemudian dapat diambil secara tersendiri dan dipisahkan dari latar belakangnya.

Meskipun gambaran simulasi penerapan *Image Clipper* yang terdapat pada ponsel Samsung sudah berhasil diterapkan menggunakan bahasa pemrograman Matlab dan Python, masih terdapat sejumlah kekurangan yang ada seperti penerapan segmentasi yang memerlukan sejumlah parameter yang memiliki nilai berubah-ubah untuk tiap kondisi gambar yang berbeda hingga hasil yang didapat juga masih belum presisi seperti yang ada pada fitur dalam ponsel Samsung. Walapun demikian, penelitian yang dilakukan sudah berhasil untuk menunjukkan bahwa metode pendeteksian tepi dan *Clustering* dapat dimanfaatkan untuk melakukan segmentasi terhadap objek.

B. Saran

Saran yang ada terkait dengan pengerjaan ini adalah agar kedepannya implementasi terkait dengan fitur *Image Clipper* dapat dikembangkan menjadi lebih presisi dengan memanfaatkan berbagai metode pengolahan citra yang lebih baik sehingga kondisi akan semakin sesuai menyesuaikan dengan yang terdapat pada ponsel Samsung sekarang.

SOURCE CODE

Untuk lampiran pranala kode pengerjaan dapat diakses pada *repository* Github dengan link berikut.

https://github.com/jasonrivalino/MakalahPCD

UCAPAN TERIMA KASIH

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmatnya, penulis dapat menyelesaikan makalah ini. Penulis juga menyampaikan terima kasih kepada keluarga dan teman-teman yang telah membantu dan memberikan dukungan kepada penulis dalam proses pembuatan makalah ini. Tidak lupa juga, penulis menyampaikan terima kasih terhadap Bapak Rinaldi Munir selaku pengampu mata kuliah IF4073 Pemrosesan Citra Digital yang senantiasa membimbing dan mendidik dalam mata kuliah ini. Terakhir, penulis juga meminta maaf jika makalah yang ada masih terdapat kesalahan dan belum sempurna.

REFERENSI

[1] SamMobile, “Work the Image Clipper’s magic on your Samsung Galaxy device”, https://www.sammobile.com/news/work-image-clipper-magic-samsung-galaxy-device/, 2023. [Diakses pada tanggal 9 Januari 2025].

[2] Program Studi Teknik Informatika STEI-ITB, "Homepage Rinaldi Munir," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/18-Pendeteksian-Tepi-Bagian1-2024.pdf [Diakses pada tanggal 10 Januari 2025].

[3] Program Studi Teknik Informatika STEI-ITB, "Homepage Rinaldi Munir," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/19-Pendeteksian-Tepi-Bagian2-2024.pdf [Diakses pada tanggal 11 Januari 2025].

[4] Program Studi Teknik Informatika STEI-ITB, "Homepage Rinaldi Munir," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/22-Segmentasi-Citra-Bagian1-2024.pdf [Diakses pada tanggal 11 Januari 2025].

[5] Program Studi Teknik Informatika STEI-ITB, "Homepage Rinaldi Munir," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2024-2025/23-Segmentasi-Citra-Bagian2-2024.pdf [Diakses pada tanggal 11 Januari 2025].

[6] UIN Suska Riau, https://repository.uin-suska.ac.id/19605/9/9.%20BAB%20IV%20%281%29.pdf [Diakses pada tanggal 11 Januari 2025].

[7] Geeksforgeeks, “Image Segmentation using K Means Clustering”, https://www.geeksforgeeks.org/image-segmentation-using-k-means-clustering/, 2023. [Diakses pada tanggal 15 Januari 2025].

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 15 Januari 2024

Jason Rivalino
13521008