UJIAN AKHIR SEMESTER

PENGOLAHAN CITRA DIGITAL

" Program Pengolahan Citra Digital - Segmentasi Citra "

Disusun Oleh :

Nama	: Andi Maha Izzah Ghazwani
Nim / Stambuk	: F551 20 114
Kelas	: C

PROGRAM STUDI S1 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TADULAKO

2022 / 2023

I. ALAT DAN BAHAN

A. Laptop / PC

B. Modul

C. Aplikasi *Matlab r2016a*

II. TEORI DASAR

Pengolahan Citra Digital (*Digital Image Processing*) merupakan bidang ilmu yang mempelajari tentang bagaimana suatu citra itu dibentuk, diolah, dan dianalisis sehingga menghasilkan informasi yang dapat dipahami oleh manusia.

Berdasarkan bentuk sinyal penyusunnya, citra dapat digolongkan menjadi dua jenis yaitu citra analog dan citra digital. Citra analog adalah citra yang dibentuk dari sinyal analog yang bersifat *continue*, sedangkan citra digital adalah citra yang dibentuk dari sinyal digital yang bersifat *diskrit*.

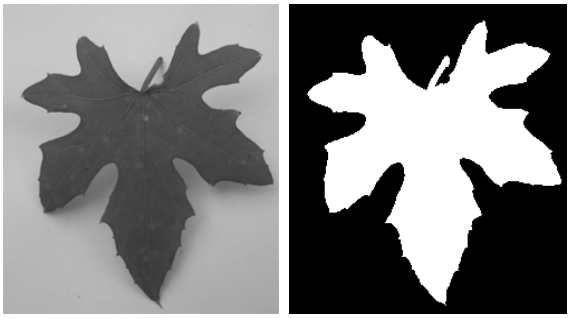
Citra analog dihasilkan dari alat akuisisi citra analog, contohnya adalah mata manusia dan kamera analog. Gambaran yang tertangkap oleh mata manusia dan foto atau film yang tertangkap oleh kamera analog merupakan contoh dari citra analog. Citra tersebut memiliki kualitas dengan tingkat kerincian (resolusi) yang sangat baik tetapi memiliki kelemahan di antaranya adalah tidak dapat disimpan, diolah, dan diduplikasi di dalam komputer. Citra digital merupakan representasi dari fungsi intensitas cahaya dalam bentuk *diskrit* pada bidang dua dimensi. Citra tersusun oleh sekumpulan piksel (*picture element*) yang memiliki koordinat *(x,y)* dan amplitudo *f(x,y)*. Koordinat *(x,y)* menunjukkan letak/posisi piksel dalam suatu citra, sedangkan amplitudo *f(x,y)* menunjukkan nilai intensitas warna citra. Representasi citra digital beserta piksel penyusunnya.

A. Citra *RGB (Red, Green, Blue)* merupakan citra yang nilai intensitas pikselnya tersusun oleh tiga kanal warna yaitu merah, hijau, dan biru.

B. Citra *Grayscale* adalah citra yang nilai intensitas pikselnya berdasarkan derajat keabuan.

C. Citra *Biner* adalah citra yang hanya memiliki dua nilai intensitas yaitu 0 (hitam) dan 1 (putih).

Segmentasi citra merupakan proses yang ditujukan untuk mendapatkan objek-objek yang terkandung di dalam citra atau membagi citra ke dalam beberapa daerah dengan setiap objek atau daerah memiliki kemiripan atribut. Pada citra yang mengandung hanya satu objek, objek dibedakan dari latarbelakangnya. Contoh Segmentasi Citra dapat di lihat sebagai beriku :



(b) Citra daun (a) Hasil segmentasi dalam bentuk biner

Gambar 1, Segmentasi Citra Dalam Bentuk Biner

Segmentasi juga biasa dilakukan sebagai langkah awal untuk melaksanakan klasifikasi objek. Gambar 1 memperlihatkkan hal ini. Setelah segmentasi citra dilaksanakan, fitur yang terdapat pada objek diambil. Sebagai contoh, fitur objek dapat berupa perbandingan lebar dan panjang objek, warna rata-rata objek, atau bahkan tekstur pada objek. Selanjutnya, melalui pengklasifikasi, jenis objek dapat ditentukan. Sebagai contoh, pengklasifikasi menyatakan bahwa daun termasuk golongan *Aglaonema*.

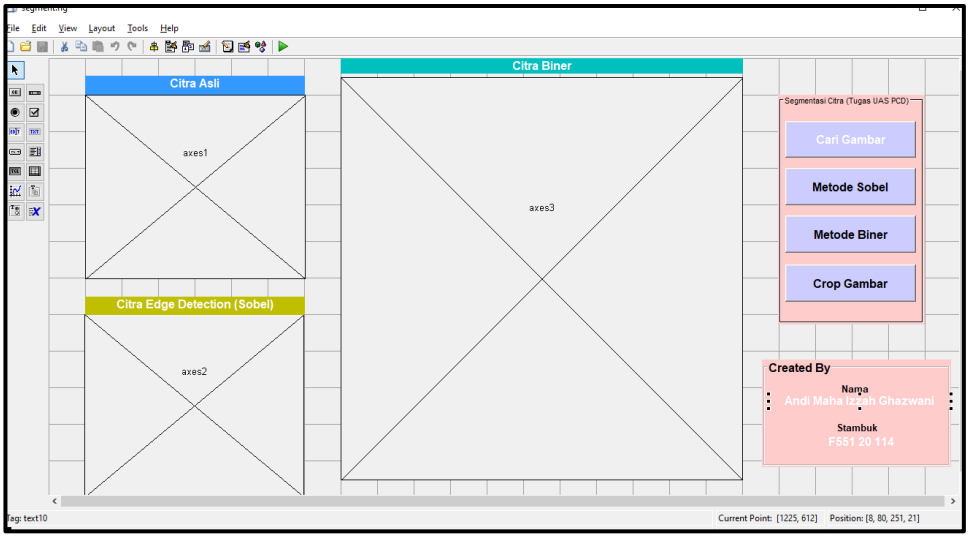
III. SCENARIO PENGOLAHAN GAMBAR

Pada program segmentasi citra kali ini agak berbeda, biasanya gambar yang akan di proses langsung di inputkan melalui *source code*/Kode program(Coding), akan tetapi pada program kali ini, gambar yang di ingin di proses/olah, pengguna cukup melakukan *load* gambar di *directory* perangkat masing-masing, setelah gambar selesai di pilih oleh *user*, selanjutnya *user* tersebut bisa melakukan pengolahan citra dengan menekan beberapa tombol pengolahan citra yang telah di siapkan oleh program. Adapun 3 tombol yang dapat digunakan untuk mengelolah citra yaitu tombol Metode *Sobel* (edge

detection), tombol metode biner dan juga tombol untuk melakukan crop gambar (*citra*). program ini mengandalkan desain GUI *(grapich user interface)*. Dimana komponen yang ada dalam program ini yaitu 3 buah label yang digunakan sebagai judul dari tiap-tiap kotak,

IV. Desain GUI, Source Code Dan Hasil Program

A. Desain GUI



B. Source Code

segment.m

```
1   function varargout = segment(varargin)
2   % SEGMENT M-file for segment.fig%...%
22
23  % Edit the above text to modify the response to help segment
24
25  % Last Modified by GUIDE v2.5 12-Jan-2015 20:52:20
26
27  % Begin initialization code - DO NOT EDIT
28  gui_Singleton = 1;
29  gui_State = struct('gui_Name',       mfilename, ...
30                     'gui_Singleton',  gui_Singleton, ...
31                     'gui_OpeningFcn', @segment_OpeningFcn, ...
32                     'gui_OutputFcn',  @segment_OutputFcn, ...
33                     'gui_LayoutFcn',  [] , ...
34                     'gui_Callback',   []);
35  if nargin && ischar(varargin{1})
36      gui_State.gui_Callback = str2func(varargin{1});
37  end
38
39  if nargout
40      [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
41  else
42      gui_mainfcn(gui_State, varargin{:});
43  end
44  % End initialization code - DO NOT EDIT
```

```
47      % --- Executes just before segment is made visible.
48      function segment_OpeningFcn(hObject, eventdata, handles, varargin)
49      % This function has no output args, see OutputFcn.
54
55      % Choose default command line output for segment
56      handles.output = hObject;
57
58      % Update handles structure
59      guidata(hObject, handles);
60
61      % UIWAIT makes segment wait for user response (see UIRESUME)
62      % uiwait(handles.figure1);
63
64
65      % --- Outputs from this function are returned to the command line.
66      function varargout = segment_OutputFcn(hObject, eventdata, handles)
67      % varargout  cell array for returning output args (see VARARGOUT);
71
72      % Get default command line output from handles structure
73      varargout{1} = handles.output;
74
77      function pushbutton1_Callback(hObject, eventdata, handles)
78      % hObject    handle to pushbutton1 (see GCBO)
81      [name_file1,name_path1] = uigetfile( ...
82      {'*.bmp;*.jpg;*.tif','Files of type (*.bmp,*.jpg,*.tif)';
83      '*.bmp','File Bitmap (*.bmp)';...
84      '*.jpg','File jpeg (*.jpg)';
85      '*.tif','File Tif (*.tif)';
86      '*.*','All Files (*.*)'},...
87      'Open Image');
88
89      if ~isequal(name_file1,0)
90      handles.data1 = imread(fullfile(name_path1,name_file1));
91      guidata(hObject,handles);
92      axes(handles.axes1);
93      imshow(handles.data1);
94      else
95      return;
96      end
98      % --- Executes on button press in pushbutton2.
99      function pushbutton2_Callback(hObject, eventdata, handles)
100     % hObject    handle to pushbutton2 (see GCBO)
103     image1 = handles.data1;
104     B=rgb2gray(image1);
105     C=double(B);
106     for i=1:size(C,1)-2
107         for j=1:size(C,2)-2
108             %Sobel mask for x-direction:
109             Gx=((2*C(i+2,j+1)+C(i+2,j)+C(i+2,j+2))-(2*C(i,j+1)+C(i,j)+C(i,j+2)));
110             %Sobel mask for y-direction:
111             Gy=((2*C(i+1,j+2)+C(i,j+2)+C(i+2,j+2))-(2*C(i+1,j)+C(i,j)+C(i+2,j)));
112
113             %The gradient of the image
114             %B(i,j)=abs(Gx)+abs(Gy);
115             B(i,j)=sqrt(Gx.^2+Gy.^2);
116
117         end
118     end
119     handles.data2 = B;
120     guidata(hObject,handles);
121     axes(handles.axes2);
122     imshow(handles.data2);
124     % --- Executes on button press in pushbutton3.
125     function pushbutton3_Callback(hObject, eventdata, handles)
126     % hObject    handle to pushbutton3 (see GCBO)
129     I = handles.data2;
130     B = graythresh(I);
131     BW = im2bw(handles.data1,B);
132     BW2 = not(BW);
133     BW3 = bwareaopen(BW2, 0);
134     handles.data3 = BW3;
135     guidata(hObject,handles);
136     axes(handles.axes3);
137     imshow(handles.data3);
```

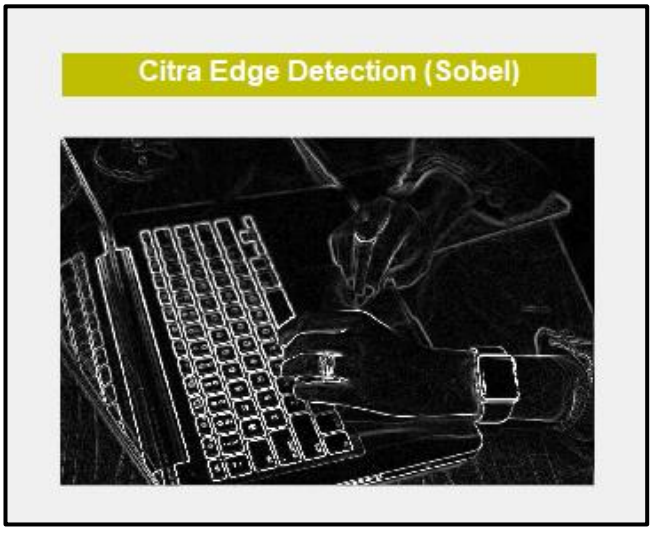
```
139      % --- Executes on button press in pushbutton4.
140      function pushbutton4_Callback(hObject, eventdata, handles)
141      % hObject    handle to pushbutton4 (see GCBO) %...%
144 -    img = handles.data1;
145 -    BW3 = handles.data3;
146 -    a=2000;
147 -    z=4000;
148 -    BW4 = xor(bwareaopen(BW3,a),bwareaopen(BW3,z));
149 -    row = size(img,1);
150 -    col = size(img,2);
151 -    for i = 1 : row
152 -        for j = 1 : col
153 -            if BW4(i,j) == 0
154 -                I(i,j) = 0;
155 -            elseif BW4(i,j) == 1
156 -                I(i,j) = img(i,j);
157 -            end
158 -        end
159 -    end
160 -    imMasked = uint8(I);
161 -    figure, imshow(imMasked), title('Hasil crop Citra Asli');
```

C. Hasil Program

A. Metode Sobel *(edge detection)*.

1. Citra Asli

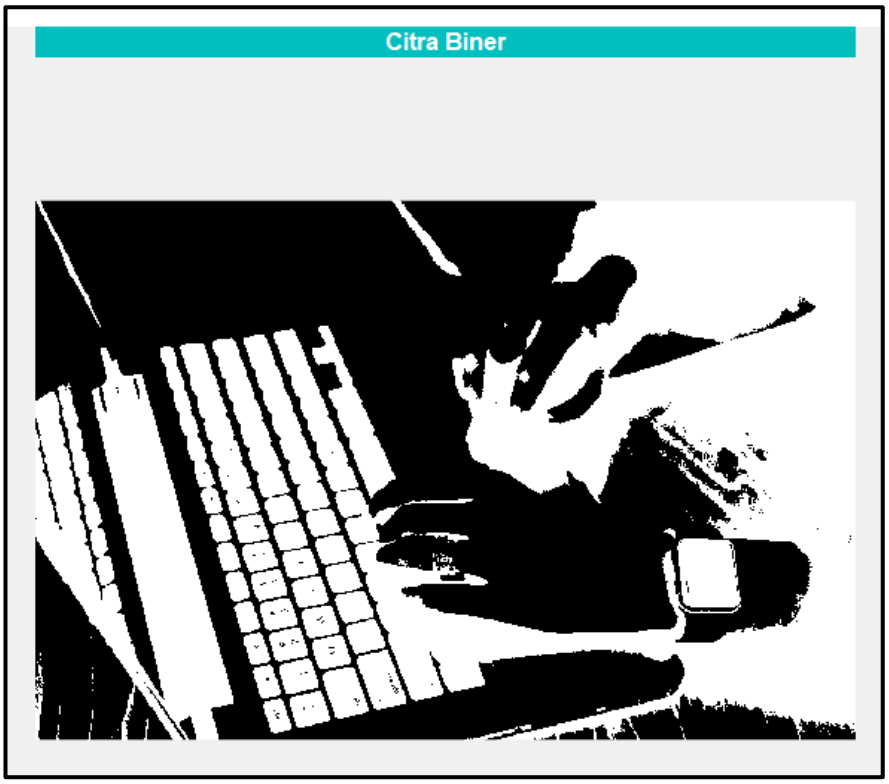
2. Hasil Metode Sobel



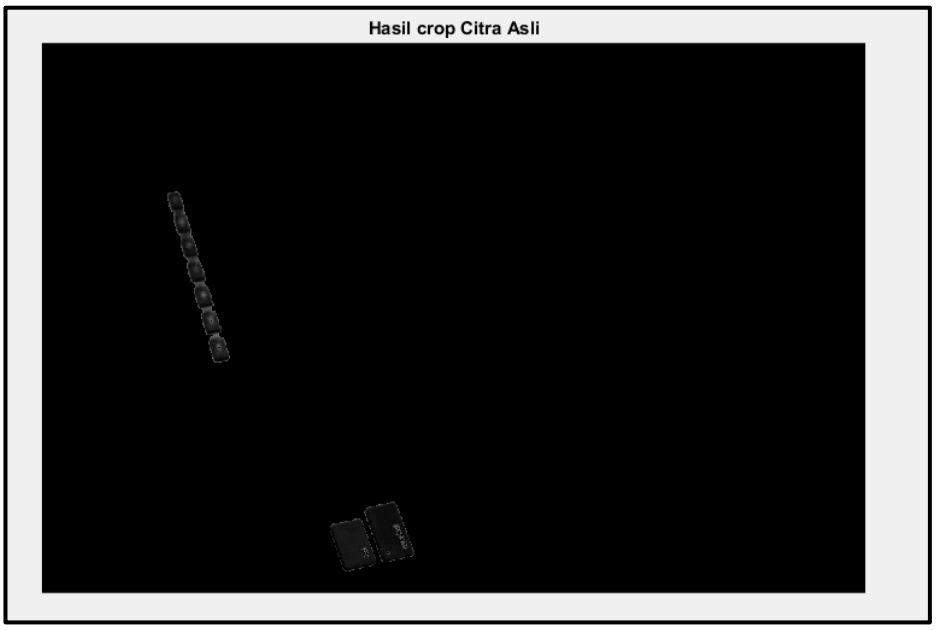
B. Metode Biner

1. Citra Asli

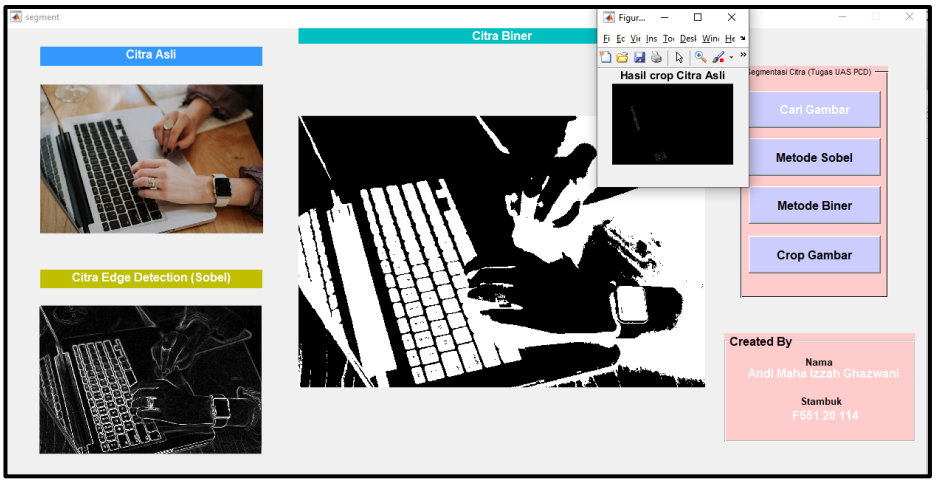
2. Hasil Metode Biner



C. Crop Citra (Gambar)



D. Hasil Running Keseluruhan



V. ANALISIS KODE PROGRAM

Pada program pengolahan citra kali ini terdapat beberapa perintah yang akan di eksekusi Ketika program ini di jalankan, Adapun kode program yang terdapat pada *pushButton1* yaitu berfungsi untuk melakukan load file citra yang dimana file citra yang akan di pakai yaitu file citra yang hanya memiliki *ekstensi* yang di tentukan oleh programmer aplikasi pengolahan citra digital ini lalu data citra tersebut akan di simpan ke dalam sebuah *variable* dengan nama "*data1*".

Kemudian untuk perintah yang ada di *pushButton2* terdapat sebuah *variable* baru dengan nama "*image1*" yang dimana *variable* ini digunakan untuk mengambil nilai yang berada di dalam *variable* "*data1*", kemudian nilai tersebut akan di olah dengan menggunakan rumus tertentu yang mendefinisikan operasi pengolahan citra dengan metode *edge detection* atau sering disebut dengan metode Sobel lalu data olahan citra tersbut akan di kirimkan ke dalam variable baru dengan nama "*data2*" sehingga hasil citra yang telah diolah tersebut akan di tampilkan kedalam kotak/komponen *axes* pada aplikasi GUI *matlab*. Pada *pushButton3* sama halnya dengan komponen *pushButton2* akan tetapi pada *pushButton3* bukan lagi mangambil nilai dari variable awal (data1) akan tetapi nilai yang akan di olah sekarang yaitu nilai hasil dari olahan operasi Sobel atau nilai yang sudah berada di dalam *variable* "*data2*". Nilai tersbut di olah dengan menggunakan perintah atau fungsi dati "*graythresh*" lalu datanya akan di kirim lagi kedalam *variable* baru dengan nama "*data3*" untuk di tampilkan kedalam kotak yang telah di sediakan dalam desain GUI lebih tepatnya kotak komponen *axes*. sehingga keluaran dari olah citra ini menghasilkan gambar dengan resolusi biner atau hitam-putih. Untuk *pushButton4* perintah yang digunakan berfungsi untuk melakukan pemanggilan data yang berada didalam *variable* "*data1*" dan "*data3*", kemudian nilai tersebut diolah dengan menggunakan rumus yang biasa disebut dengan *cropping citra* sehingga keluaran dari hasil pengolahan ini yaitu gambar crop dengan resolusi biner atau hitam-putih.