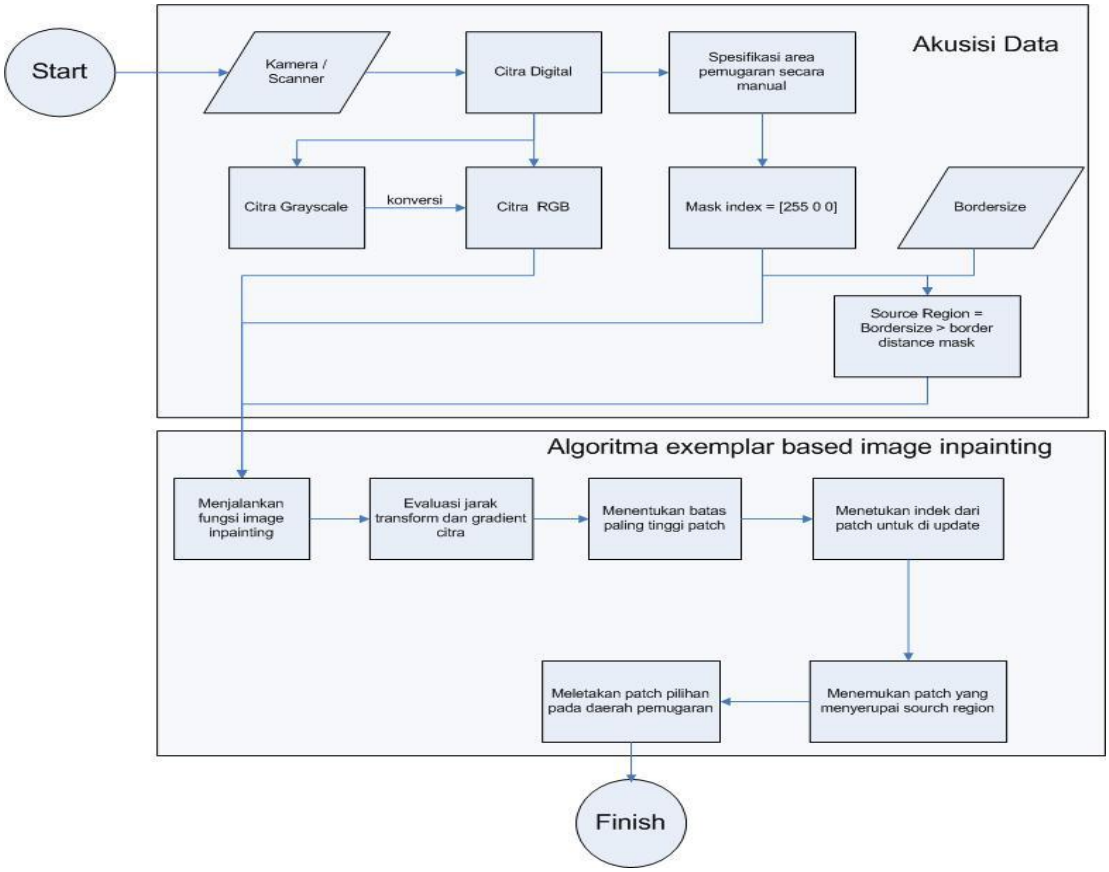
BAB IV

HASIL DAN PEMBAHASAN

4.1 Perancangan Aplikasi

Secara umum aplikasi pemugaran citra *digital* terbagi menjadi dua proses, yaitu proses akusisi data dan algoritma *exemplar-based image inpainting.* Alur kerja dapat dilihat pada Gambar 4.1.



Gambar 4.1 Diagram Alir.

4.1.1 Akusisi Data

Tahap pertama untuk menjalankan sistem penghilang *noise* yaitu akuisisi data. Akuisisi data adalah proses mengubah data *analog* menjadi data *digital* agar mampu dianalisis menggunakan komputer. Dengan berkembangnya ilmu pengetahuan dan teknologi, citra *analog* dapat dikonversi menjadi data *digital* dengan menggunakan alat bantu berupa *scanner*. Selain itu data *digital* juga didapatkan dari kamera *digital*. Data objek yang digunakan dapat berupa *filetype* JPG, PNG, dan TIFF. Citra *digital* yang diproses berupa citra RGB, jika citra yang di*input*kan berupa citra *grayscale*, citra tersebut harus dikonversi terlebih dahulu menjadi citra RGB.

Setelah citra *digital* di*input*kan, proses selanjutnya adalah mengindentifikasi derau secara *manual*. Derau yang dihilangkan dapat berupa derau yang disengaja ataupun tidak. Pada proses ini juga terdapat indentifikasi objek yang dihilangkan dan mengganti dengan piksel yang baru. Identifikasi ini dilakukan secara *manual*, yaitu dengan membuat *mask index* yang ditandai dengan warna merah [255 0 0].

Nilai *border size* digunakan sebagai identifikasi *source region*, Informasi *source region* digunakan untuk mengisi atau menggantikan piksel citra yang dipugar. Setelah data diakusisi dan disimpan dalam media penyimpanan, data dibuka untuk diubah menjadi matrik piksel dengan ukuran yang sama dengan ukuran gambar.

4.1.2 Algoritma *Exemplar-Based Image Inpainting*

Setelah citra yang dipugar dan citra *mask* di*input*kan proses selanjutnya adalah masuk pada tahap pemugaran citra dengan menggunakan algoritma *exemplar-based image inpainting*. Tahap algoritma ini pertama adalah evaluasi jarak dan *gradient* citra dengan menggunakan perkalian konvolusi dua dimensi antar citra dengan *kernel*nya. Berikut merupakan implementasi program dari algoritma yang dibahas pada bab sebelumnya.

```
% evaluasi jarak transform dan gradient citra
   D = bwdist(M);
   Nx = convn(convn(I.gray,GI.g','same'),GI.gp,'same');
   Ny = convn(convn(I.gray,GI.gp','same'),GI.g,'same');
   Nm = Nx.^2 + Ny.^2;
```

Gambar 4.2 *Source Code* Evaluasi Jarak dan *Gradient.*

Code diatas merupakan implementasi program dalam bahasa *MATLAB* untuk:

$$f(x)*g(x)=\sum_{\alpha=-\sim} f(x)g(x-a)$$

Perintah *convn* pada *code* merupakan bagian dari *toolbox image processing* pada *MATLAB*. *Convn* digunakan untuk menghitung nilai konvolusi gambar dengan nilai matriks yang sama. Selanjutnya nilai itu digunakan untuk menentukan jarak *transform* dan *gradient* citra. Langkah selanjutnya adalah menentukan batas paling tinggi *patch* yang digunakan. *Code* di bawah ini adalah implementasi program untuk rumus:

$$C(\boldsymbol{P}) = \frac{\sum q \in \Psi_p \cap (\mathrm{I} - \Omega) C(\mathbf{q})}{|\Psi_{\mathrm{P}}|} \quad \text{dan} \quad \mathrm{D}(\boldsymbol{P}) = \frac{\left|\nabla \mathrm{I}_p^{\perp} . n_p\right|}{\alpha}$$

```
    % menentukan batas prioritas paling tinggi patch
B.normal     = zeros(B.length,2);
B.isophote   = zeros(B.length,2);
B.confidence = zeros(B.length,1);
B.data       = zeros(B.length,1);
B.priority   = zeros(B.length,1);
for i = 1:B.length

   % Mengevaluasi penggunaan normal batas jarak transformasikan
   Mp = ~M(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   Dp = D(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   B.normal(i,:) = [GN.x(Mp) GN.y(Mp)]'*Dp(Mp);
   if norm(B.normal(i,:)) ~= 0
      B.normal(i,:) = B.normal(i,:)/norm(B.normal(i,:));
   end

   % Ekstrak isophote mempergunakan maksimum gradien diantara window
   Nxp = Nx(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   Nyp = Ny(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   Nmp = Nm(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   [maxValue,maxIndex] = max(Nmp(:));
   B.isophote(i,:) = [-Nyp(maxIndex) Nxp(maxIndex)];

   % Evaluasi prioritas patch
   Cp = C(B.boundary(i,1)+pIndex,B.boundary(i,2)+pIndex);
   B.confidence(i) = sum(sum(Cp(Mp)))/(2*pSize+1)^2;
   B.data(i) = abs(sum(B.normal(i,:).*B.isophote(i,:)))/dataAlpha;
   if useIsophote
      B.priority(i) = B.confidence(i)*B.data(i);
   else
      B.priority(i) = B.confidence(i);
   end

end
```

Gambar 4.3 *Source Code* Penentuan Batas Prioritas Paling Tinggi *Patch.*

Pada tahap penentuan batas prioritas *patch* paling tinggi, terdapat beberapa proses di bawahnya yaitu dengan mengevaluasi penggunaan batas normal jarak yang ditansformasikan. Dilanjutkan mengekstrak *ishopote* menggunakan maksimum *gradient* di antara *window*. Proses terakhir dari penentuan batas prioritas *patch* adalah mengevaluasi

prioritas *patch* dengan menggunakan *ishopote*. Langkah selanjutnya adalah menentukan indeks dari *patch* untuk di*update.*

```
[maxPriority,blockIndex] = max(B.priority);
```

Gambar 4.4 *Source Code* Penentuan Indeks *Patch.*

Code di atas merupakan implementasi dari algoritma:

$$p = \arg\max_{P \in \Omega} P$$

Setelah *patch* di*update* proses selanjutnya menemukan *patch* yang menyerupai *source region*. *Code* di bawah mneyatakan proses pada algoritma:

$$C(P) = C(P') \qquad \forall p \ \psi_p \cap \Omega$$

```
block = I.rgb(B.boundary(blockIndex,1)+pIndex,...
              B.boundary(blockIndex,2)+pIndex,:);
mask  = M(B.boundary(blockIndex,1)+pIndex,...
          B.boundary(blockIndex,2)+pIndex);
frame = F(B.boundary(blockIndex,1)+pIndex,...
          B.boundary(blockIndex,2)+pIndex);
match = findmatch(I.rgb,S,block,mask,frame);
```

Gambar 4.5 *Source Code* Menemukan *Patch* Menyerupai *Source Region.*

Pada proses di atas, terdapat pemanggilan fungsi *findmatch*. Langkah yang terakhir adalah meletakan *patch* pilihan pada daerah pemugaran dan proses *inpainting* akan berhenti setelah semua daerah pemugaran terisi.

4.1.3 Hasil Akhir Aplikasi

Setelah proses pemugaran citra *digital*, sistem menghasilkan keluaran berupa perbandingan citra sebelum dipugar dengan citra hasil. Citra hasil dapat disimpan dalam bentuk *filetype* JPG, PNG, dan TIFF. Selain itu hasil akhir aplikasi ini dilengkapi juga dengan lama waktu pemrosesan.

4.2 Tampilan Sistem

Aplikasi pemugaran citra *digital* dengan menggunakan algoritma *exemplar-based image inpainting* dan metode sintesis tekstur ini dibuat serta dikembangkan berdasarkan pengetahuan yang diperoleh dan ditambah dengan berbagai referensi. Secara umum aplikasi ini memiliki tiga *menu* utama yaitu *File*, *Toolbox*, dan *Help and About* yang tergabung dalam satu *file* bernama pemugaran.m.

Gambar 4.6 Tampilan Utama Aplikasi.

Aplikasi ini sendiri memiliki beberapa modul yang terintegrasi dengan sistem, antara lain *open* citra, *open mask*, menyimpan citra hasil, keluar, *toolbox* citra *input*, *toolbox* citra *output*, *manual book*, *about*, *input* citra, *input mask* ,*process* , *reset*, *save* citra, dan waktu proses.

4.2.1 *Menu File Toolbar*

Gambar 4.7 *Menu* File Toolbar.

Menu file terdiri dari:

1. *Open Citra*

   Menu ini berfungsi memanggil data gambar dengan format JPEG, PNG, dan TIFF yang telah diakuisisi dan ditampilkan di *Axes* 1 yang berada pada bagian kiri atas. Setiap gambar yang dipanggil secara otomatis mengikuti ukuran *axes* 1 tersebut *(auto scaling)*.

2. *Open Mask*

   Menu ini berfungsi memanggil data gambar *Mask* dengan format JPEG, PNG, dan TIFF yang telah diakuisisi dan ditampilkan di *axes* 2 yang disediakan. Seperti halnya data

gambar, pada tahap ini juga secara otomatis gambar yang dipanggil disesuaikan dengan ukuran *axes* 2 tersebut *(auto scaling)*.

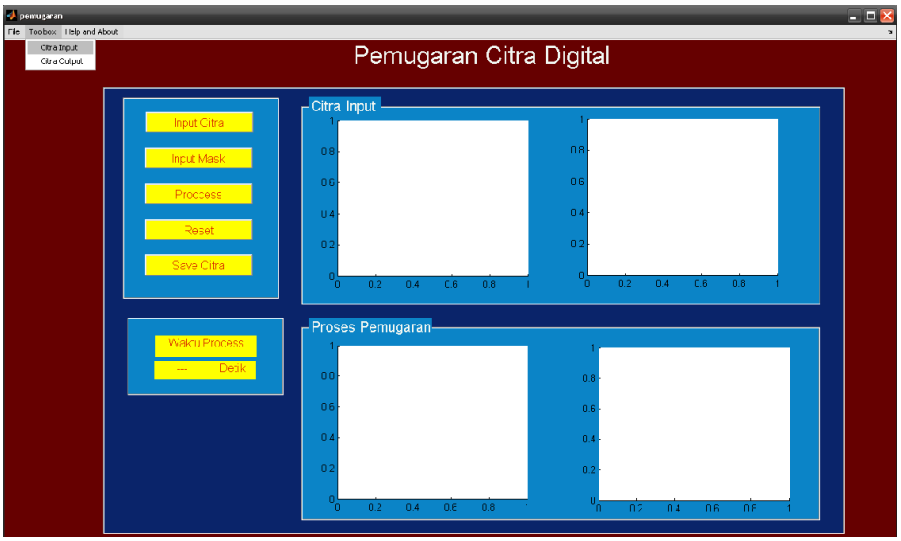
3. Menyimpan Hasil Citra

   Menu ini memiliki fungsi menyimpan gambar yang telah berhasil diperbaiki ke dalam media penyimpanan. Format penyimpanan *file* dapt berupa berbagai tipe data yaitu JPEG, PNG, dan TIFF. F*ile* hasil proses dapat disimpan sebagai file baru atau di*overwrite* pada file yang sama.

4. Keluar

   Menu keluar seperti namanya, berfungsi *menu*tup aplikasi setelah digunakan. Untuk *menu*tup aplikasi juga dapat dilakukan dengan meng*click* tombol silang di pojok kanan atas aplikasi.

4.2.2 *Menu Toolbox*



Gambar 4.8 *Menu Toolbox.*

Menu ini berfungsi untuk menampilkan rincian data pada citra. *Submenu* yang terdapat pada *menu* ini adalah:

1. *Toolbox* Citra *Input*

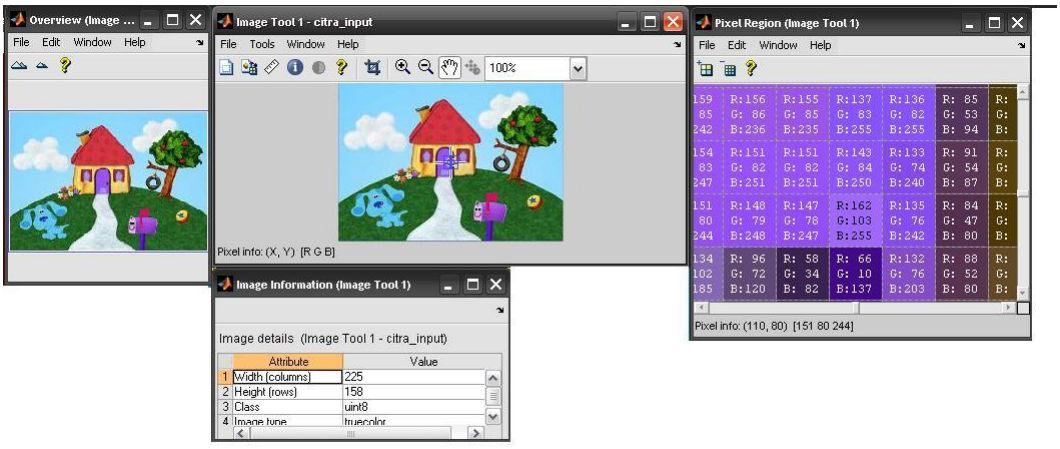
   Menu ini memiliki fungsi menampilkan *toolbox* citra *input*, yang digunakan untuk meneliti lebih jauh tentang derau pada citra. *Menu* ini akan aktif jika gambar pada *axes* 1 tampil.

2. *Toolbox* Citra *Output*

   Menu ini memiliki fungsi menampilkan *toolbox* citra *output*, yang digunakan untuk memeriks proses penghilangan derau pada citra. *Menu* ini baru aktif setelah gambar pada *axes* 3 tampil atau setelah pemrosesan *inpainting* selesai.

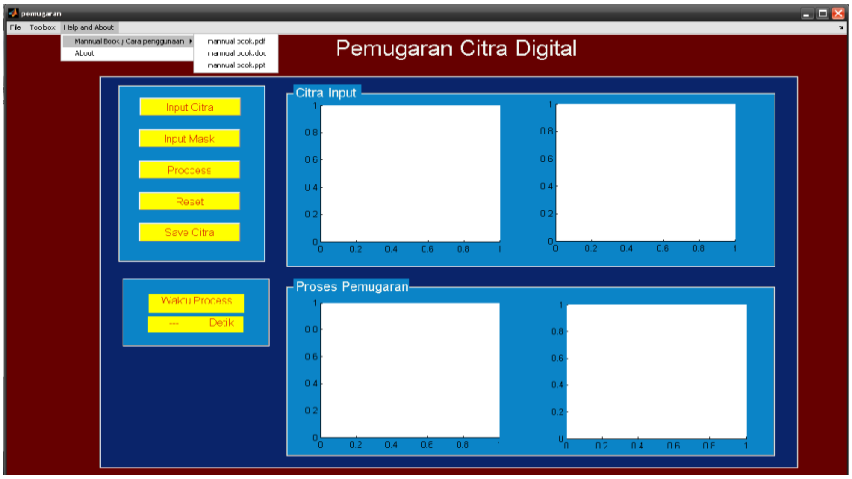
Menu toolbox memiliki beberapa fungsi pendukung seperti:

1. *Image properties.*
2. *Pixel region.*
3. *Zoom in / zoom out*



Gambar 4.9 Fungsi *Menu Toolbox.*

4.2.3 Menu Help and About



Gambar 4.10 *Menu Help and About*

Menu ini berfungsi untuk penjelasan penggunaan aplikasi pemugaran citra *digital*. *Submenu* yang terdapat pada *menu* ini adalah:

1. *Manual Book*

   Menu ini berfungsi untuk membuka buku panduan penggunaan aplikasi pemugaran citra *digital* ini. *Manual* book pada aplikasi ini berupa *file* bertipe .doc, .ppt, dan .pdf.

2. *About*

   Menu about merupakan *menu* yang menampilkan informasi yang berkaitan dengan aplikasi yang dibuat, seperti nama aplikasi, pembuat, informasi pembuat, dan referensi dalam membangun aplikasi.

4.2.4 *Direct Menu*

Gambar 4.11 *Direct Menu.*

Menu ini bertujuan untuk memudahkan penggunaan aplikasi pemugaran citra digital. *Submenu* yang terdapat pada *menu* ini adalah:

1. *Input* Citra

   Menu ini merupakan tombol yang memiliki fungsi sama dengan *menu Open* Citra, yaitu membuka data gambar yang diproses. Pembuatan tombol ini bertujuan agar user yang tidak terbiasa menggunakan fasilitas *menu* dapat langsung mengakses data masukan yang diinginkan dengan lebih mudah. Tombol dengan nama *pushbutton1* memanggil proses yang sama dengan *menu Open* Citra Sehingga menggunakan fasilitas *menu Open* Citra maupun tombol *Input* Citra menghasilkan *output* yang sama.

2. *Input Mask*

   Menu ini merupakan tombol yang memiliki fungsi sama dengan *menu Open mask*, yaitu membuka data gambar *mask* yang diproses. Tombol dengan nama *pushbutton2* memanggil proses yang sama dengan *menu Open Mask* sehingga menggunakan fasilitas *menu Open Mask* maupun tombol *Input Mask* menghasilkan *output* yang sama.

3. *Process*

   Tombol *Process* dalam aplikasi ini adalah tombol utama yang mengaktifkan fungsi pemugaran citra *digital* pada gambar masukan. Tombol dengan nama *pushbutton3* ini memanggil fungsi *inpaint* pada *file* inpaint.m yang berfungsi melakukan pemugaran citra *digital*.

4. *Reset*

   Tombol *reset* merupakan tombol yang digunakan untuk mengembalikan aplikasi ke kondisi awal. Tombol ini mengakibatkan *axes1, axes2, axes3, axes4,* dan *timel* menjadi kosong.

5. *Save* Citra

   Tombol *Save* citra seperti halnya dengan *menu* menyimpan citra hasil memiliki fungsi yang sama yaitu menyimpan gambar yang telah diperbaiki ke dalam media penyimpanan.

6. Waktu *Process*

Menu ini menampilkan berapa lama waktu yang digunakan pada saat pemugaran citra *digital*. Waktu dihitung dari awal setelah tombol *Process* ditekan dan berhenti setelah citra selesai dipugar.

4.3 Pembahasan

4.3.1 Alur Proses Aplikasi

Aplikasi pemugaran citra *digital* dimulai dengan membuka data masukan berupa *file* gambar dengan tipe JPEG, PNG, dan TIFF. Cara pemanggilan gambar dapat dilakukan dengan dua cara, melalui *menu Open* Citra dan tombol *Input* citra dengan *code*:

```
function pushbutton1_Callback(hObject, eventdata, handles)
I = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png','pick a jpge
file');
axes(handles.axes1)
imshow(I);
```

Gambar 4.12 *Source Code* Membuka Citra dengan Tombol *Input* Citra.

```
function mOCitra_Callback(hObject, eventdata, handles)
I = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png','pick a jpge
file');
axes(handles.axes1)
imshow(I);
```

Gambar 4.13 *Source Code* Membuka Citra dengan *Menu Open* Citra.

Pada gambar *source code* sebelumnya terdapat beberapa perintah dari pemrograman MATLAB seperti perintah:

1. *Uigetfile*

   Perintah ini digunakan untuk mendapatkan file citra yang akan dipugar.

2. *Imshow*

   Perintah *Imshow* berfungsi untuk menampilkan citra.

Setelah perintah buka *file* dipanggil, *file* gambar yang diperbaiki diletakkan pada *axes*1, tampilan aplikasi setelah gambar dipanggil menjadi:

Gambar 4.14. *Input* Citra.

Setelah gambar di*input*kan langkah selanjutnya adalah meng*input*kan *mask* citra yang dimulai dengan membuka data masukan berupa *file* gambar citra dengan tipe JPEG, PNG, dan TIFF. Cara

pemanggilan gambar dapat dilakukan dengan dua cara, melalui *menu Open mask* dan tombol *Input mask* dengan *code*:

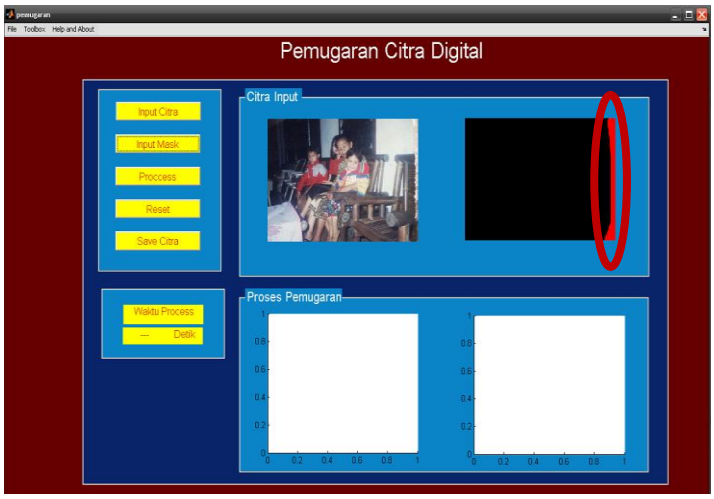
```
function pushbutton2_Callback(hObject, eventdata, handles)
M = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png','pick a png
file');
axes(handles.axes2)
imshow(M);
```

Gambar 4.15 *Source Code* Membuka Gambar dengan Tombol *Input Mask.*

```
function mO_Mask_Callback(hObject, eventdata, handles)
M = uigetfile('*.jpg;*.tiff;*.ppm;*.pgm;*.png','pick a png
file');
axes(handles.axes2)
imshow(M);
```

Gambar 4.16 *Source Code* Membuka Gambar dengan *Menu Open Mask.*

Setelah perintah buka *file* dipanggil, *file* gambar yang diperbaiki diletakkan pada *axes*2, tampilan aplikasi setelah gambar dipanggil menjadi:



Gambar 4.17 Citra *Mask*

Setelah citra dan *mask* di*input*kan, tahap selanjutnya adalah menekan tombol *Process* untuk memproses pemugara citra. Berikut adalah *source code* pada tombol *Process*.

```
myusaha = guidata(gcbo);
```

Fungsi guidata ini khusus digunakan untuk pemrograman menggunakan *GUIDE*. guidata digunakan untuk mendapatkan *object handle* (*tag*) dari komponen *form* yang aktif.

```
tic   % Starting Stopwatch Timer
set(handles.timel,'string','proses');

set(handles.timel,'string',toc)
```

Fungsi *tic* digunakan untuk mengatur *stopwatch* yang digunakan untuk menghitung waktu proses pemugaran citra. Sebaliknya, fungsi *toc* berguna untuk menghentikan pencatatan waktu proses pemugaran pada saat proses selesai.

```
I = im2double(getimage(myusaha.axes1));
nrows = size(I,1);
ncols = size(I,2);

x = 1:ncols;
y = 1:nrows;
[X,Y] = meshgrid(x,y);
```

Fungsi di atas digunakan untuk mengubah citra *input* menjadi citra berganda dan mengindentifikasi piksel dengan (x,y). Perintah *im2double* pada *code* di atas digunakan untuk mengkonversi gambar menjadi citra berganda dengan dua presisi.

```
M = getimage(myusaha.axes2);
[fillImg,fillRegion] =
loadimgs(getimage(myusaha.axes2),[255 0 0]);
M =fillRegion;

function [fillImg,fillRegion] =
loadimgs(getimage,fillColor)
fillImg =getimage;fillRegion = fillImg(:,:,1)==fillColor(1)
& ...
fillImg(:,:,2)==fillColor(2) &
fillImg(:,:,3)==fillColor(3);
```

Fungsi di atas digunakan untuk memproses *mask*, yang mana pada citra *mask* diidentifikasikan dengan nilai RGB [255 0 0]. Artinya daerah yang dipugar ditandai akan bernilai ff0000 atau berwarna merah. Perintah *getimage* digunakan untuk mendapatkan citra *mask* yang sebelumnya telah di*input*kan.

```
borderSize  = 10;
S = bwdist(M) < borderSize;
                S(M) = 0;
```

Fungsi di atas digunakan menandakan *border size* yang digunakan serta, pengenalan *source region* yang digunakan sebagai sumber area untuk mengisi daerah yang dipugar. *Border size* berfungsi untuk menandai daerah sumber.

```
if size(I,3) == 1
          I = repmat(I,[1 1 3]);
end

[J,C] = inpaint(I,M,S);
```

Fungsi sebelumnya digunakan untuk mengindentifikasi citra, jika citra itu merupakan citra *grayscale* maka akan dijadikan citra RGB. Perintah

repmat berfungsi untuk mengkonversi citra menjadi citra RGB. Tahap berikutnya adalah memanggil fungsi *inpaint*.

```
figure(2); clf;
set(gcf,'Name','Hasil Perbandingan Sebelum Dan Sesudah');
set(gcf,'NumberTitle','off','DoubleBuffer','on');
subplot(1,2,1); imagesc(I);
axis image;
set(gca,'YDir','reverse','XTick',[],'YTick',[]);
subplot(1,2,2); imagesc(J.rgb);
axis image;
set(gca,'YDir','reverse','XTick',[],'YTick',[]);
```

Fungsi di atas digunakan untuk menampilkan figur dua yang berisikan perbandingan citra sebelum dipugar dan hasil citra yang sudah dipugar. Gambar-gambar berikut adalah tampilan aplikasi saat pemrosesan pemugaran citra *digital*.

Gambar 4.18 Pemrosesan Pemugaran Citra *digital*.

Gambar 4.19 Hasil Pemrosesan Pemugaran Citra *Digital*.

Pemrosesan citra *digital* berhenti jika semua piksel yang terdapat dalam *mask* selesai diproses. Hasil proses ditunjukan dengan tampilnya waktu *process* serta hasil citra yang sudah diproses pada *axes* 3. Setelah semua tahap selesai tampil figur dua yang berisikan perbandingan citra sebelum dan sesudah.

Gambar 4.20 Hasil Pemugaran Citra *Digital* pada Figure Dua.

Output yang diperoleh kemudian disimpan ke dalam media penyimpanan yang tersedia dengan format JPEG, PNG, dan TIF. Gambar yang telah disimpan tersebut dapat digunakan baik untuk langsung dicetak ataupun untuk data masukan kembali jika hasil yang diperoleh kurang memuaskan.

4.3.2 Observasi

Observasi aplikasi dilakukan dengan *sample* yang berbeda-beda. Hasil yang ditunjukkan dalam pengujian ini adalah gambar yang diujicobakan pada aplikasi sehingga dapat dilihat secara langsung perbedaan sebelum dan sesudah proses pereduksian.

1. Observasi menghilangkan noise berupa jamur pada citra tua

Gambar 4.21 Hasil Pemugaran pada Citra Tua.

Pada citra tersebut dapat dilihat bahwa proses pemugaran citra berjalan dengan baik. Derau yang terjadi akibat noda jamur pada citra tua secara berangsur-angsur hilang dan diganti dengan piksel-piksel baru.

2. Observasi menghilangkan objek yang tidak diinginkan

Gambar 4.22 Hasil Pemugaran Citra Mengilangkan Objek.

Percobaan dilakukan pada citra untuk mengilangkan objek yang tidak diinginkan. Hasil proses dapat memperlihatkan bahwa pemugaran citra berjalan dengan baik.

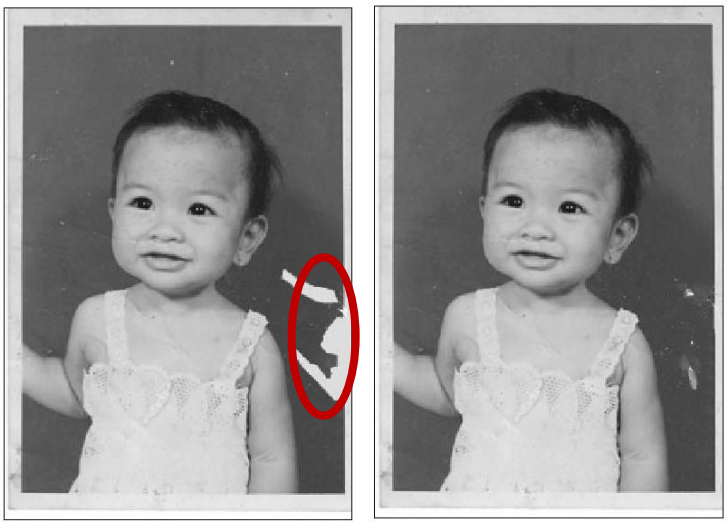
3. Observasi menghilangkan objek berupa teks

Gambar 4.23 Hasil Pemugaran Citra Mengilangkan Objek Berupa Teks.

Percobaan dilakukan pada citra yang terdapat tanggal pembuatan yang merupkan *default* dari kamera yang digunakan. Berdasarkan hasil proses pengamatan pemugaran citra berjalan dengan baik.

Tanggal yang tertera pada citra dapat dihilangkan dan digantikan dengan piksel-piksel baru.

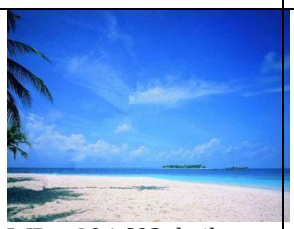
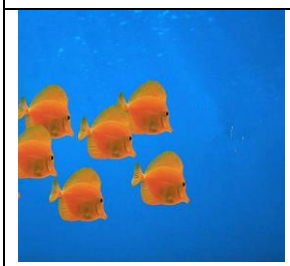
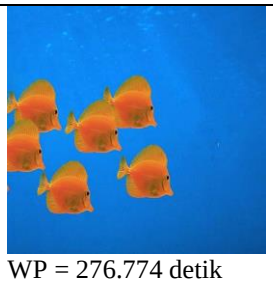
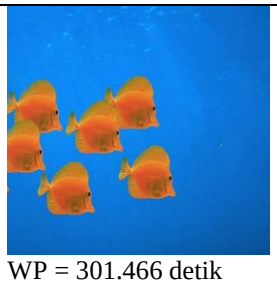
4. Observasi citra pada citra *grayscale*



Gambar 4.24 Hasil Pemugaran Citra *Grayscale.*

Percobaan dilakukan pada citra *grayscale*s Berdasarkan perbandingan gambar di atas dapat dilihat bahwa proses perbaikan berjalan dengan kurang baik. Karena tidak semua daerah yang *mask* tersisi dengan sempurna. Hal ini dikarenakan letak derau terdapat pada tepian sehingga daerah sumber yang tidak sempurna.

5. Observasi berdasarkan *border size* yang digunakan dan lama pemrosesan.

Border size 10	*Border size* 15	*Border size* 20	Citra asli
 WP = 169.637 detik	 WP = 204.693 detik	 WP = 239.299 detik	
 WP = 86.2377 detik	 WP = 98.3002 detik	 WP = 105.549 detik	
 WP = 155.441 detik	 WP = 180.433 detik	 WP = 185.843 detik	
 WP = 305.703 detik	 WP = 396.172 detik	 WP = 404.253 detik	
 WP = 229.263 detik	 WP = 276.774 detik	 WP = 301.466 detik	

Gambar 4.25 Observasi Berdasarkan Perbedaan *Border Size*

Observasi ini dimulai dengan mengatur *border size* masing-masing bernilai 10, 15, dan 20. Berdasarkan hasil yang didapatkan perbedaan *border size* mempengaruhi lama waktu pemrosesan. Minimum nilai *border size* menjadi minimum waktu proses, sebaliknya maksimum *border size* menjadikan waktu proses maksimum. Hal ini dikarenakan nilai *border size* mempengaruhi diameter daerah sumber.

Berdasarkan hasil yang didapatkan perbedaan *border size* juga mempengaruhi kualitas citra hasil, yaitu citra yang dihasilkan *border size* minimum menghasilkan citra hasil yang kualitasnya kurang baik jika dibandingkan dengan hasil dari citra dengan nilai *border size* di atasnya atau maksimum.