**MATLAB PROGRAM FOR IMAGE PROCESSING GUI**

function varargout = imageProc(varargin)

% IMAGEPROC M-file for imageProc.fig

% IMAGEPROC, by itself, creates a new IMAGEPROC or raises the existing

% singleton\*.

%

% H = IMAGEPROC returns the handle to a new IMAGEPROC or the handle to

% the existing singleton\*.

%

% IMAGEPROC('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in IMAGEPROC.M with the given input arguments.

%

% IMAGEPROC('Property','Value',...) creates a new IMAGEPROC or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before imageProc\_OpeningFunction gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to imageProc\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Copyright 2002-2003 The MathWorks, Inc.

% Edit the above text to modify the response to help imageProc

% Last Modified by GUIDE v2.5 14-Apr-2014 17:14:22

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @imageProc\_OpeningFcn, ...

'gui\_OutputFcn', @imageProc\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before imageProc is made visible.

function imageProc\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to imageProc (see VARARGIN)

% Choose default command line output for imageProc

clc;

handles.fileLoaded = 0;

handles.fileLoaded2 = 0;

set(handles.axes1,'Visible','off');

set(handles.axes2,'Visible','off');

set(handles.axesHist1,'Visible','off');

set(handles.axesHist2,'Visible','off');

set(handles.editPath, 'Visible', 'off');

set(handles.editSize, 'Visible', 'off');

set(handles.editComment, 'Visible', 'off');

set(handles.textHist1, 'Visible', 'off');

set(handles.textHist2, 'Visible', 'off');

set(handles.sliderBright, 'Enable', 'off');

set(handles.sliderContrast, 'Enable', 'off');

set(handles.editBright,'String', sprintf('%10s:%4.0f%%', 'Brightness', 100\*get(handles.sliderBright,'Value')));

set(handles.editContrast,'String', sprintf('%10s:%4.0f%%', 'Contrast', 100\*get(handles.sliderContrast,'Value')));

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.

function varargout = imageProc\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in LoadButton.

function LoadButton\_Callback(hObject, eventdata, handles)

% hObject handle to LoadButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[FileName,PathName] = uigetfile({'\*.\*'},'Load Image File');

if (FileName==0) % cancel pressed

return;

end

handles.fullPath = [PathName FileName];

[a, b, Ext] = fileparts(FileName); %returns the path, filename, extension

availableExt = {'.bmp','.jpg','.jpeg','.tiff','.png','.gif'}; %supported image formats in MATLAB

FOUND = 0;

for i=1:length(availableExt)

if (strcmpi(Ext, availableExt{i}))

FOUND=1;

break;

end

end

if (FOUND==0)

h = msgbox('File type not supported!','Error','error');

return;

end

set(handles.editPath, 'Visible', 'on');

set(handles.editSize, 'Visible', 'on');

set(handles.editComment, 'Visible', 'on');

info = imfinfo(handles.fullPath);

if (~isempty(info.Comment))

% save current image comment (to be used later in image save)

handles.currentImageComment = info.Comment{1};

else

handles.currentImageComment = '';

end

set(handles.editSize, 'String', sprintf('SIZE (W x H) : %d x %d', info.Width, info.Height));

set(handles.editComment, 'String', sprintf('COMMENT: %s', handles.currentImageComment));

set(handles.editPath, 'String', handles.fullPath);

set(handles.sliderBright, 'Enable', 'on');

set(handles.sliderContrast, 'Enable', 'on');

RGB = imread(handles.fullPath);

handles.RGB = RGB;

handles.RGB2 = RGB;

handles.RGBtmp = RGB;

handles.fileLoaded = 1;

handles.fileLoaded2 = 0;

set(handles.axes1,'Visible','off'); set(handles.axes2,'Visible','off');

set(handles.axesHist1,'Visible','off'); set(handles.axesHist2,'Visible','off');

set(handles.textHist1, 'Visible', 'off');

axes(handles.axesHist2); cla; %cla clears current axis

set(handles.textHist2, 'Visible', 'off');

axes(handles.axes1); cla; imshow(RGB);

axes(handles.axes2); cla;

set(handles.axes1,'Visible','on');

handles = updateHistograms(handles);

% Update handles structure

guidata(hObject, handles);

% --- Executes on button press in CopyButton.

function CopyButton\_Callback(hObject, eventdata, handles)

% hObject handle to CopyButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded==1)

handles.RGB2 = handles.RGB;

axes(handles.axes2); imshow(handles.RGB2);

handles.fileLoaded2 = 1;

handles = updateHistograms(handles);

guidata(hObject, handles);

else

h = msgbox('No primary file has been loaded!','Error','error');

end

% --- Executes on button press in MedianButton.

function MedianButton\_Callback(hObject, eventdata, handles)

% hObject handle to MedianButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded2==1)

[M,N,ttt] = size(handles.RGB2);

handles.RGBtmp = handles.RGB2;

elseif(handles.fileLoaded==1)

[M,N,ttt] = size(handles.RGB);

handles.RGBtmp = handles.RGB;

else

h = msgbox('No primary file has been loaded!','Error','error');

return;

end

RUN = 1;

while (RUN==1)

prompt = {'Enter Median Row Factor (0-5%):','Enter Median Column Factor (0-5%):'};

dlg\_title = 'Enter Median Parameters:';

num\_lines = 1; %number of lines for text box

def = {'2','2'}; %default input values

answer = inputdlg(prompt,dlg\_title,num\_lines,def); %answer taken as a cell-array

if (isempty(answer))

return;

end

M1 = str2num(answer{1})/100;

M2 = str2num(answer{2})/100;

if ((str2num(answer{1})>=0) && (str2num(answer{1})<=5)) && ((str2num(answer{2})>=0) && (str2num(answer{2})<=5))

RUN = 0;

end

end

M1 = round(M1 \* M);

M2 = round(M2 \* N);

w = waitbar(0, 'Median filtering ... Please wait ...');

handles.RGB2(:,:,1) = medfilt2(handles.RGBtmp(:,:,1),[M1 M2]); % Median filtering for Red component

waitbar(1/3, w);

handles.RGB2(:,:,2) = medfilt2(handles.RGBtmp(:,:,2),[M1 M2]); % Median filtering for Green component

waitbar(2/3, w);

handles.RGB2(:,:,3) = medfilt2(handles.RGBtmp(:,:,3),[M1 M2]); % Median filtering for Blue component

close(w);

axes(handles.axes2); imshow(handles.RGB2);

handles.fileLoaded2 = 1;

handles = updateHistograms(handles);

guidata(hObject, handles);

% --- Executes on button press in SharpButton.

function SharpButton\_Callback(hObject, eventdata, handles)

% hObject handle to SharpButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded2==1)

handles.RGBtmp = handles.RGB2;

elseif (handles.fileLoaded==1)

handles.RGBtmp = handles.RGB;

else

h = msgbox('No primary file has been loaded!','Error','error');

return

end

H = fspecial('unsharp');

handles.RGB2(:,:,1) = imfilter(handles.RGBtmp(:,:,1),H,'replicate');

handles.RGB2(:,:,2) = imfilter(handles.RGBtmp(:,:,2),H,'replicate');

handles.RGB2(:,:,3) = imfilter(handles.RGBtmp(:,:,3),H,'replicate');

axes(handles.axes2); imshow(handles.RGB2);

handles.fileLoaded2 = 1;

handles = updateHistograms(handles);

guidata(hObject, handles);

% --- Executes on button press in GrayButton.

function GrayButton\_Callback(hObject, eventdata, handles)

% hObject handle to GrayButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded2==1)

handles.RGBtmp = handles.RGB2;

elseif (handles.fileLoaded==1)

handles.RGBtmp = handles.RGB;

else

h = msgbox('No primary file has been loaded!','Error','error');

return

end

Gray = rgb2gray(handles.RGBtmp);

handles.RGB2(:,:,1) = Gray;

handles.RGB2(:,:,2) = Gray;

handles.RGB2(:,:,3) = Gray;

axes(handles.axes2); imshow(handles.RGB2);

handles.fileLoaded2 = 1;

handles = updateHistograms(handles);

guidata(hObject, handles);

% --- Executes on button press in SaveButton.

function SaveButton\_Callback(hObject, eventdata, handles)

% hObject handle to SaveButton (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded2==1)

[file,path] = uiputfile('\*.jpg','Save Secondary Image As');

imwrite(handles.RGB2,[path file],'jpg');

else

h = msgbox('No secondary file has been loaded!','Save Error','error');

end

function editSize\_Callback(hObject, eventdata, handles)

% hObject handle to editSize (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editSize as text

% str2double(get(hObject,'String')) returns contents of editSize as a double

% --- Executes during object creation, after setting all properties.

function editSize\_CreateFcn(hObject, eventdata, handles)

% hObject handle to editSize (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function editComment\_Callback(hObject, eventdata, handles)

% hObject handle to editComment (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editComment as text

% str2double(get(hObject,'String')) returns contents of editComment as a double

% --- Executes during object creation, after setting all properties.

function editComment\_CreateFcn(hObject, eventdata, handles)

% hObject handle to editComment (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function editPath\_Callback(hObject, eventdata, handles)

% hObject handle to editPath (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editPath as text

% str2double(get(hObject,'String')) returns contents of editPath as a double

% --- Executes during object creation, after setting all properties.

function editPath\_CreateFcn(hObject, eventdata, handles)

% hObject handle to editPath (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on slider movement.

function sliderBright\_Callback(hObject, eventdata, handles)

% hObject handle to sliderBright (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.editBright,'String', sprintf('%10s:%4.0f%%', 'Brightness', 100\*get(handles.sliderBright,'Value')));

% --- Executes during object creation, after setting all properties.

function sliderBright\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderBright (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on slider movement.

function sliderContrast\_Callback(hObject, eventdata, handles)

% hObject handle to sliderContrast (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

set(handles.editContrast,'String', sprintf('%10s:%4.0f%%', 'Contrast', 100\*get(handles.sliderContrast,'Value')));

% --- Executes during object creation, after setting all properties.

function sliderContrast\_CreateFcn(hObject, eventdata, handles)

% hObject handle to sliderContrast (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.

if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor',[.9 .9 .9]);

end

% --- Executes on button press in pushbuttonContrastBrightness.

function pushbuttonContrastBrightness\_Callback(hObject, eventdata, handles)

% hObject handle to pushbuttonContrastBrightness (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

if (handles.fileLoaded2==1)

handles.RGBtmp = handles.RGB2;

elseif (handles.fileLoaded==1)

handles.RGBtmp = handles.RGB;

else

h = msgbox('No primary file has been loaded!','Error','error');

return

end

handles.RGB2 = changeBrightness(handles.RGBtmp, get(handles.sliderBright, 'Value'), get(handles.sliderContrast, 'Value'));

axes(handles.axes2); imshow(handles.RGB2);

handles.fileLoaded2 = 1;

handles = updateHistograms(handles);

guidata(hObject, handles);

function handlesNew = updateHistograms(handles)

handlesNew = handles;

if (handles.fileLoaded == 1)

set(handles.textHist1, 'Visible', 'on');

axes(handlesNew.axesHist1);

cla;

ImageData1 = reshape(handlesNew.RGB(:,:,1), [size(handlesNew.RGB, 1) \* size(handlesNew.RGB, 2) 1]);

ImageData2 = reshape(handlesNew.RGB(:,:,2), [size(handlesNew.RGB, 1) \* size(handlesNew.RGB, 2) 1]);

ImageData3 = reshape(handlesNew.RGB(:,:,3), [size(handlesNew.RGB, 1) \* size(handlesNew.RGB, 2) 1]);

[H1, X1] = hist(ImageData1, 1:5:256);

[H2, X2] = hist(ImageData2, 1:5:256);

[H3, X3] = hist(ImageData3, 1:5:256);

hold on;

plot(X1, H1, 'r');

plot(X2, H2, 'g');

plot(X3, H3, 'b');

axis([0 256 0 max([H1 H2 H3])]);

end

if (handlesNew.fileLoaded2 == 1)

set(handles.textHist2, 'Visible', 'on');

axes(handlesNew.axesHist2);

cla;

ImageData1 = reshape(handlesNew.RGB2(:,:,1), [size(handlesNew.RGB2, 1) \* size(handlesNew.RGB2, 2) 1]);

ImageData2 = reshape(handlesNew.RGB2(:,:,2), [size(handlesNew.RGB2, 1) \* size(handlesNew.RGB2, 2) 1]);

ImageData3 = reshape(handlesNew.RGB2(:,:,3), [size(handlesNew.RGB2, 1) \* size(handlesNew.RGB2, 2) 1]);

[H1, X1] = hist(ImageData1, 1:5:256);

[H2, X2] = hist(ImageData2, 1:5:256);

[H3, X3] = hist(ImageData3, 1:5:256);

hold on;

plot(X1, H1, 'r');

plot(X2, H2, 'g');

plot(X3, H3, 'b');

axis([0 256 0 max([H1 H2 H3])]);

end

function editBright\_Callback(hObject, eventdata, handles)

% hObject handle to editBright (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editBright as text

% str2double(get(hObject,'String')) returns contents of editBright as a double

% --- Executes during object creation, after setting all properties.

function editBright\_CreateFcn(hObject, eventdata, handles)

% hObject handle to editBright (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function editContrast\_Callback(hObject, eventdata, handles)

% hObject handle to editContrast (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of editContrast as text

% str2double(get(hObject,'String')) returns contents of editContrast as a double

% --- Executes during object creation, after setting all properties.

function editContrast\_CreateFcn(hObject, eventdata, handles)

% hObject handle to editContrast (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

**MATLAB PROGRAM FOR SLIDE SHOW CONTROL**

clear; clc;

%image url for capture may change according to connection

url = 'http://100.72.61.138:8080/shot.jpg';

%start ActiveX server and connect with PowerPoint Application

ppt = actxserver('powerpoint.application');

ppt.Visible = 1;

h = ppt.Presentations.Open('C:\Users\SAMSUNG\Desktop\abc.ppt');

pause(5);

while 1

%Read image from camera

ss = imread(url);

%Segmentation

k=ss(:,:,1)<=135 & ss(:,:,1)>=100 & ss(:,:,2)<=40 & ss(:,:,2)>=10 & ss(:,:,3)<=40 & ss(:,:,3)>=10;

%Dilation followed by erosion and filling

se=strel('disk', 5);

e=imclose(k, se);

f=imfill(e, 'holes');

%count number of connected components

[stat num]=bwlabel(f);

disp(num);

pause(0.5);

if num==1

%move to next slide

ppt.ActivePresentation.SlideShowWindow.View.Next;

elseif num==2

%move to previous slide

ppt.ActivePresentation.SlideShowWindow.View.Previous;

elseif num==3

%exit the slideshow

ppt.ActivePresentation.SlideShowWindow.View.Exit;

elseif num==4

break;

else

end

pause(1);

end

%Save and close the Presentation

ppt.ActivePresentation.Save;

ppt.ActivePresentation.Close;

%disconnect from server and delete the object

ppt.Quit;

ppt.delete;

**MATLAB PROGRAM FOR WINDOWS MEDIA PLAYER CONTROL**

1. **Building the GUI**

function varargout = act\_ive\_gui\_new(varargin)

% ACT\_IVE\_GUI\_NEW M-file for act\_ive\_gui\_new.fig

% ACT\_IVE\_GUI\_NEW, by itself, creates a new ACT\_IVE\_GUI\_NEW or raises the existing

% singleton\*.

%

% H = ACT\_IVE\_GUI\_NEW returns the handle to a new ACT\_IVE\_GUI\_NEW or the handle to

% the existing singleton\*.

%

% ACT\_IVE\_GUI\_NEW('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in ACT\_IVE\_GUI\_NEW.M with the given input arguments.

%

% ACT\_IVE\_GUI\_NEW('Property','Value',...) creates a new ACT\_IVE\_GUI\_NEW or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before act\_ive\_gui\_new\_OpeningFunction gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to act\_ive\_gui\_new\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help act\_ive\_gui\_new

% Last Modified by GUIDE v2.5 18-Apr-2014 11:20:06

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @act\_ive\_gui\_new\_OpeningFcn, ...

'gui\_OutputFcn', @act\_ive\_gui\_new\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin & isstr(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before act\_ive\_gui\_new is made visible.

function act\_ive\_gui\_new\_OpeningFcn(hObject, eventdata, handles,varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to act\_ive\_gui\_new (see VARARGIN)

% Choose default command line output for act\_ive\_gui\_new

clc;

handles.output = hObject;

%set(gcf,'Color',[0 0 0]);

pos=[50 200 550 400];

MovieControl = actxcontrol('WMPlayer.OCX',pos);

set(MovieControl,'uiMode','none');

assignin('base','MovieControl',MovieControl);

set(MovieControl.settings,'autoStart',0);

handles.MovieControl = MovieControl;

set(handles.edit2,'ForegroundColor',[0 1 0]);

set(handles.edit2,'BackgroundColor',[0 0 0]);

workdir=pwd;

handles.workdir=workdir;

[a,map]=imread('play.jpg');

[r,c,d]=size(a);

x=ceil(r/18);

y=ceil(c/18);

g=a(1:x:end,1:y:end,:);

g(g==255)=0.8\*255;

set(handles.pushbutton2,'CData',g);

[a,map]=imread('stop.jpg');

[r,c,d]=size(a);

x=ceil(r/20);

y=ceil(c/20);

g=a(1:x:end,1:y:end,:);

g(g==255)=0.8\*255;

set(handles.pushbutton3,'CData',g);

[a,map]=imread('open\_files.jpg');

[r,c,d]=size(a);

x=ceil(r/18);

y=ceil(c/18);

g=a(1:x:end,1:y:end,:);

g(g==255)=0.8\*255;

set(handles.pushbutton1,'CData',g);

[a,map]=imread('pause.jpg');

[r,c,d]=size(a);

x=ceil(r/20);

y=ceil(c/20);

g=a(1:x:end,1:y:end,:);

g(g==255)=0.8\*255;

set(handles.pushbutton4,'CData',g);

% Update handles structure

guidata(hObject, handles);

% UIWAIT makes act\_ive\_gui\_new wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = act\_ive\_gui\_new\_OutputFcn(hObject, eventdata,handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in pushbutton1.

function pushbutton1\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[filename pathname] = uigetfile('\*.\*','Please select a file');

if ~filename

return

end;

file1=[pathname filename];

[path,name,ext] = fileparts(file1);

set(handles.listbox1,'String',[name,ext],'Value',1);

mp = handles.MovieControl;

mp.URL=[pathname filename];

dir\_path=pathname;

% x.name=dir\_path;

% dir\_path=x.name;

% x.name1=file1;

handles.dir\_path=dir\_path;

handles.pan=file1;

handles.numind=1;

numind=handles.numind;

guidata(hObject,handles);

set(handles.listbox1,'UserData',dir\_path);

% --- Executes on button press in pushbutton2.

function pushbutton2\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

mp = handles.MovieControl;

mp.controls.play;

pause(1);

% currpos=mp.controls.currentPosition;

pause(1);

r=get(mp.currentMedia,'duration');

set(handles.slider3,'max',r)

% for i=currpos:r

currpos1=mp.controls.currentPosition;

% %index\_selected1 = get(handles.listbox1,'Value');

% pause(1);

r=get(mp.currentMedia,'duration');

remaint=r-currpos1;

set(handles.edit2,'String',remaint);

set(handles.slider3,'Value',currpos1);

% end

% --- Executes on button press in pushbutton3.

function pushbutton3\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

mp = handles.MovieControl;

% stopval=get(handles.pushbutton3,'Value');

% if stopval==1

mp.controls.stop

% end

% --- Executes on button press in pushbutton4.

function pushbutton4\_Callback(hObject, eventdata, handles)

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

mp = handles.MovieControl;

mp.controls.pause

% --- Executes during object creation, after setting all properties.

function slider2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to slider2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change

% 'usewhitebg' to 0 to use default. See ISPC and COMPUTER.

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

% --- Executes on slider movement.

function slider2\_Callback(hObject, eventdata, handles)

% hObject handle to slider2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

mp = handles.MovieControl;

k=get(handles.slider2,'Value');

set(mp.settings,'Volume',k);

% --- Executes during object creation, after setting all properties.

function slider3\_CreateFcn(hObject, eventdata, handles)

% hObject handle to slider3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change

% 'usewhitebg' to 0 to use default. See ISPC and COMPUTER.

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

% --- Executes on slider movement.

function slider3\_Callback(hObject, eventdata, handles, varargin)

% hObject handle to slider3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

mp = handles.MovieControl;

j=get(hObject,'Value');

set(mp.controls,'CurrentPosition',j);

% --- Executes during object creation, after setting all properties.

function listbox1\_CreateFcn(hObject, eventdata, handles)

% hObject handle to listbox1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc

set(hObject,'BackgroundColor',[0 0 0]);

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

% varargout{1} = handles.output;

% --- Executes on selection change in listbox1.

function listbox1\_Callback(hObject, eventdata, handles)

% hObject handle to listbox1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns listbox1 contents as cell array

% contents{get(hObject,'Value')} returns selected item from listbox1

mp = handles.MovieControl;

index\_selected = get(handles.listbox1,'Value')

dir\_path=get(handles.listbox1,'UserData');

set(handles.listbox1,'Value',index\_selected)

file\_list1 = get(handles.listbox1,'String')

assignin('base','file\_list1',file\_list1)

k=iscell(file\_list1)

k1=ischar(file\_list1)

if k==1 & k1==0

filename1= file\_list1{index\_selected};

elseif k1==1 & k==0

file\_list1=cellstr(file\_list1);

filename1=file\_list1{index\_selected};

elseif k==0

filename1=handles.pan

end

[path,name,ext,ver] = fileparts(filename1);

g=fullfile(dir\_path,[name ext ver]);

mp.URL=g;

pause(0.05);

h=get(handles.figure1,'SelectionType');

h1=strcmp(h,'open');

if h1

mp.controls.play

else

return;

end

currpos=mp.controls.currentPosition;

pause(1);

r=get(mp.currentMedia,'duration');

set(handles.slider3,'max',r)

for i=currpos:r

currpos1=mp.controls.currentPosition;

index\_selected1 = get(handles.listbox1,'Value');

pause(1);

r=get(mp.currentMedia,'duration');

remaint=r-currpos1;

set(handles.edit2,'String',remaint);

posend=0;

if remaint <=0 || posend==r

break;

end

end

mp = handles.MovieControl;

index\_selected = get(handles.listbox1,'Value')

dir\_path=get(handles.listbox1,'UserData');

set(handles.listbox1,'Value',index\_selected)

file\_list1 = get(handles.listbox1,'String')

assignin('base','file\_list1',file\_list1)

k=iscell(file\_list1)

k1=ischar(file\_list1)

if k==1 & k1==0

filename1= file\_list1{index\_selected};

elseif k1==1 & k==0

file\_list1=cellstr(file\_list1);

filename1=file\_list1{index\_selected};

elseif k==0

filename1=handles.pan

end

[path,name,ext,ver] = fileparts(filename1);

g=fullfile(dir\_path,[name ext ver]);

j=get(handles.edit1,'String');

set(mp.settings,'playCount',str2double(j));

mp.URL=g;

pause(0.05);

h=get(handles.figure1,'SelectionType');

h1=strcmp(h,'open');

if h1 || index\_selected1~=index\_selected

mp.controls.play

else

return;

end

currpos=mp.controls.currentPosition;

pause(1);

r=get(mp.currentMedia,'duration');

set(handles.slider3,'max',r)

for i=currpos:r

currpos1=mp.controls.currentPosition;

index\_selected1 = get(handles.listbox1,'Value');

pause(1);

r=get(mp.currentMedia,'duration');

remaint=r-currpos1;

set(handles.edit2,'String',remaint);

posend=0;

if remaint <=0 || posend==r

break;

end

end

set(handles.listbox1,'Value',index\_selected+1);

% --- Executes during object creation, after setting all properties.

function edit2\_CreateFcn(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc

set(hObject,'BackgroundColor','white');

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

function edit2\_Callback(hObject, eventdata, handles)

% hObject handle to edit2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text

% str2double(get(hObject,'String')) returns contents of edit2 as a

% double

1. **Code to control the Windows Media Player**

**MATLAB PROGRAM FOR CONTROL OF ROOM LIGHT AND/OR LED LIGHTS OR OTHER APPLIANCES**

clc; clear all;

last\_state=[0 0 0 0]; %Stores last state of four LEDs

a=arduino ('COM44'); %object constructor for Arduino control

pin=[12 11 10 9]; %Pin number on Arduino board

%declare pin mode as output for the Digital Pins on Arduino board

a.pinMode(pin(1),'output');

a.pinMode(pin(2),'output');

a.pinMode(pin(3),'output');

a.pinMode(pin(4),'output');

%initialising the video object with image capturing device

v=videoinput('winvideo', 2);

while 1

i=getsnapshot(v);

y=ycbcr2rgb(i);

%thresolding or feature extraction

k=y(:,:,1)<=255 & y(:,:,1)>=59 & y(:,:,2)<=255 & y(:,:,2)>=53 & y(:,:,3)<=255 & y(:,:,3)>=32;

f=imfill(k,'holes');

se=strel('disk',10);

e=imerode(f,se);

%count number of Region of Interest

[stat num]=bwlabel(e);

%Take a decision based on the calculated Region of Interest

if num==1

disp('1');

if last\_state(1)==0

a.digitalWrite(pin(1),1);

last\_state(1)=1;

elseif last\_state(1)==1

a.digitalWrite(pin(1),0);

last\_state(1)=0;

end

elseif num==2

disp('2');

if last\_state(2)==0

a.digitalWrite(pin(2),1);

last\_state(2)=1;

elseif last\_state(2)==1

a.digitalWrite(pin(2),0);

last\_state(2)=0;

end

elseif num==3

disp('3');

if last\_state(3)==0

a.digitalWrite(pin(3),1);

last\_state(3)=1;

elseif last\_state(3)==1

a.digitalWrite(pin(3),0);

last\_state(3)=0;

end

elseif num==4

disp('4');

if last\_state(4)==0

a.digitalWrite(pin(4),1);

last\_state(4)=1;

elseif last\_state(4)==1

a.digitalWrite(pin(4),0);

last\_state(4)=0;

end

elseif num==5

a.digitalWrite(pin(1),0);

a.digitalWrite(pin(2),0);

a.digitalWrite(pin(3),0);

a.digitalWrite(pin(4),0);

last\_state(1:4)=0;

break;

else

disp(0);

end %end of if

end % end of while

delete(a); % delete the object constructor

**MATLAB function ‘COLORSEG’ for performing colour segmentation of RGB images**

function I = colorseg(varargin)

%COLORSEG Performs segmentation of a color image.

% S = COLORSEG('EUCLIDEAN ' , F, T, M) performs segmentation of color image F % using a Euclidean measure of similarity. M is a 1-by-3 vector representing the % average color used for segmentation (this is the center of the sphere in Fig. 6.26 % of DIPUM). T is the threshold against which the distances are compared.

%

% S = COLORSEG( ' MAHALANOBIS ' , F, T, M, C) performs segmentation ofcolor % image F using the Mahalanobis distance as a measure ofsimilarity. C is the 3-by -% 3 covariance matrix of the sample color vectors of the class of interest. See % function covmatrix for the computation of C and M.

%

% S is the segmented image (a binary matrix) in which 0s denote the background.

% Preliminaries.

% Recall that varargin is a cell array.

f = varargin{2};

if(ndims(f) ~= 3) || (size(f, 3) ~= 3)

error ( ' Input image must be RGB . ' ) ;

end

M = size(f, 1); N = size(f, 2);

% Convert f to vector format using function imstack2vectors.

f = imstack2vectors(f);

f = double(f);

% Initialize I as a column vector. It will be reshaped later into an image.

I = zeros(M\*N, 1);

T = varargin{3};

m = varargin{4};

m = m(:)'; % Make sure that m is a row vector.

if length(varargin) == 4

method = ' euclidean ' ;

elseif length(varargin) == 5

method = 'mahalanobis';

else

error( 'Wrong number of inputs. ');

end

switch method

case 'euclidean'

% Compute the Euclidean distance between all rows of X and m. See Section 12.2 % of DIPUM for an explanation of the following expression. D(i) is the Euclidean % distance between vector X(i,:) and vector m.

p = length(f);

D = sqrt(sum(abs(f - repmat(m, p, 1)).^2, 2));

case 'mahalanobis'

C = varargin{5};

D = mahalanobis(f, C, m);

otherwise

error( ' Unknown segmentation method. ')

end

% D is a vector of size MN-by-1 containing the distance computationsfrom all the color % pixels to vector m. Find the distances <= T.

J = D <= T;

% Set the values of I(J) to 1. These are the segmented color pixels.

I (J) = 1;

% Reshape I into an M-by-N image.

I = reshape(I, M, N);

**MATLAB function ‘mahalanobis’ for computing the mahalanobis distance when using colour segmentation**

function D = mahalanobis(varargin)

% MAHALANOBIS computes the mahalanobis distance.

% D = MAHALANOBIS (Y, X) computes the mahalanobis Distance between each vector % in Y to the mean (centroid) of the vector in X, and outputs the result in vector D, % whose length is size(Y,1). The vectors in X and Y are assumed to be organized as % rows. The input data can be real or complex. The outputs are real quantities.

%

% D = MAHALANOBIS (Y, CX, MX) computes the mahalanobis Distance between % each vector in Y and the given mean vector MX. The result are output in vector D% whose length is Size(Y, 1). The vectors in Y are assumed to be organized as the% rows of this array. The input data can be real or complex. The outputs are real% quantities. In addition to the mean vector MX, the covariance matrix CX of a% population of vectors X must be provided also. Use function COVMATRIX(Section% 11.5) to compute MX and CX.

% Reference: Acklam, P.J.[2002]. “MATLAB Array Manipulation tips and tricks,” % available at

% Home.online.no/-pjacklam/matlab/doc/mtt/index.html

% Or in the Tutorials section at

% www.imageprocessingplace.com

param = varargin; % keep in mind that param is a cell array.

Y = param(1);

if length(param) == 2

X = param{2};

%compute the mean vector and covariance matrix of the vector in X.

[Cx, mx] = covmatrix(X);

elseif length(param) == 3 % cov. Matrix and mean vector provided.

Cx = param{2};

mx=param{3};

else

error('Wrong number of inputs')

end

mx=mx(:)'; % Make sure that mx is a row vector for the next step.

% subtract the mean vector from each vector in Y.

Yc = bsxfun(@minus, Y, mx);

% compute the mahalanobis distances.

D= real(sum(Yc/Cx.\*conj(Yc), 2));

**MATLAB function for Adaptive Median Filtering of Input Image**

function f = adpmedian(g, Smax)

% ADPMEDIAN Perform adaptive median filtering.

% F = ADPMEDIAN(G, SMAX) performs adaptive median filtering of

% image G. The median filter starts at size 3-by-3 and iterates

% up to size SMAX - by - SMAX. SMAX must be an odd integer greater

% than 1.

% SMAX must be an odd, positive integer greater than 1.

if (Smax <= 1) || (Smax/2 == round(Smax/2)) || (Smax ~= round(Smax))

error(' SMAX must be an odd integer > 1.')

end

% Initial setup.

f = g;

f(:) = 0;

alreadyProcessed = false(size(g));

% Begin filtering.

for k = 3:2:Smax

zmin = ordfilt2(g, 1, ones(k, k), 'symmetric');

zmax = ordfilt2(g, k \* k, ones(k, k), 'symmetric');

zmed = medfilt2(g, [k k], 'symmetric');

processUsingLevelB = (zmed > zmin) & (zmax > zmed) & ~alreadyProcessed;

zB = (g > zmin) & (zmax > g);

outputZxy = processUsingLevelB & zB;

outputZmed = processUsingLevelB & ~zB;

f(outputZxy) = g(outputZxy);

f(outputZmed) = zmed(outputZmed);

alreadyProcessed = alreadyProcessed | processUsingLevelB;

if all(alreadyProcessed(:))

break;

end

end

% Output zmed for any remaining unprocessed pixels. Note that %this zmed was computed using a window of size Smax-by- Smax, %which is the final value of k in the loop.

f( ~alreadyProcessed) = zmed(~alreadyProcessed);