General Sir John Kotelawala Defense University

Faculty of Computing

BSC DEGREE IN COMPUTER SCIENCE/ COMPUTER ENGINEERING/ SOFTWARE ENGINEERING

Image Processing and Computer Vision – CS3042

INTAKE 36

ASSIGNMENT – 03

AMRNVB Pethiyagoda

D/BCE/19/0002

Suggest an algorithm using image processing techniques and develop a MAT LAB code For

“Vehicle Number Plate Detection and Recognition”.

Include following sections to your report

a) System Architecture
b) Activity Diagram (Process flow chart)
c) Image Processing Techniques

MAT LAB Code

Below code is for create_templates.m file

```
A=imread('A.bmp');
B=imread('B.bmp');
C=imread('C.bmp');
D=imread('D.bmp');
E=imread('E.bmp');
F=imread('F.bmp');
G=imread('G.bmp');
H=imread('H.bmp');
I=imread('I.bmp');
J=imread('J.bmp');
K=imread('K.bmp');
L=imread('L.bmp');
M=imread('M.bmp');
N=imread('N.bmp');
O=imread('O.bmp');
P=imread('P.bmp');
Q=imread('Q.bmp');
R=imread('R.bmp');
S=imread('S.bmp');
T=imread('T.bmp');
U=imread('U.bmp');
V=imread('V.bmp');
W=imread('W.bmp');
X=imread('X.bmp');
Y=imread('Y.bmp');
Z=imread('Z.bmp');
Afill=imread('fillA.bmp');
Bfill=imread('fillB.bmp');
Dfill=imread('fillD.bmp');
```

```
Ofill=imread('fillO.bmp');
Pfill=imread('fillP.bmp');
Qfill=imread('fillQ.bmp');
Rfill=imread('fillR.bmp');

one=imread('1.bmp');
two=imread('2.bmp');
three=imread('3.bmp');
four=imread('4.bmp');
five=imread('5.bmp');
six=imread('6.bmp');
seven=imread('7.bmp');
eight=imread('8.bmp');
nine=imread('9.bmp');
zero=imread('0.bmp');
zerofill=imread('fill0.bmp');
fourfill=imread('fill4.bmp');
sixfill=imread('fill6.bmp');
sixfill2=imread('fill6_2.bmp');
eightfill=imread('fill8.bmp');
ninefill=imread('fill9.bmp');
ninefill2=imread('fill9_2.bmp');

letter=[A Afill B Bfill C D Dfill E F G H I J K L M...
    N O Ofill P Pfill Q Qfill R Rfill S T U V W X Y Z];

number=[one two three four fourfill five...
    six sixfill sixfill2 seven eight eightfill nine ninefill ninefill2 zero
zerofill];

character=[letter number];

NewTemplates=mat2cell(character,42,[24 24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24 ...
    24 24 24 24 24 24 24]);

save ('NewTemplates','NewTemplates')
clear all
```

Below code is for App.m file (Vehicle Number Plate Detection and Recognition GUI)

```
function varargout = App(varargin)
% APP MATLAB code for App.fig
%      APP, by itself, creates a new APP or raises the existing
%      singleton*.
%
%      H = APP returns the handle to a new APP or the handle to
%      the existing singleton*.
%
%      APP('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in APP.M with the given input arguments.
%
%      APP('Property','Value',...) creates a new APP or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before App_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to App_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 App

% Last Modified by GUIDE v2.5 11-Jun-2021 19:22:52

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
    'gui_Singleton',  gui_Singleton, ...
    'gui_OpeningFcn', @App_OpeningFcn, ...
    'gui_OutputFcn',  @App_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 App is made visible.
function App_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 App (see VARARGIN)
```

```
set(handles.OriginalImage,'visible', 'off');
% Choose default command line output for App
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes App wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = App_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('*.jpg','Select Image');
fullpath=strcat(Pathname,Filename);
f=imread(fullpath);
axes(handles.OriginalImage);
imshow(f);
f=imresize(f,[400 NaN]);
g=rgb2gray(f);
g=medfilt2(g,[3 3]);
se=strel('disk',1);
gi=imdilate(g,se);
ge=imerode(g,se);
gdiff=imsubtract(gi,ge);
gdiff=mat2gray(gdiff);
gdiff=conv2(gdiff,[1 1;1 1]);
gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1);
B=logical(gdiff);
er=imerode(B,strel('line',50,0));
out1=imsubtract(B,er);
F=imfill(out1,'holes');
H=bwmorph(F,'thin',1);
H=imerode(H,strel('line',3,90));
final=bwareaopen(H,100);
Iprops=regionprops(final,'BoundingBox','Image');
NR=cat(1,Iprops.BoundingBox);

r=controlling(NR);
if ~isempty(r)
    I={Iprops.Image};
```

```
    noPlate=[];
    for v=1:length(r)
        N=I{1,r(v)};
        letter=readLetter(N);
        while letter=='O' || letter=='0'
            if v<=3
                letter='O';
            else
                letter='0';
            end
            break;
        end
        noPlate=[noPlate letter];
    end
    %msgbox(strcat('Vehicle Registraction Number :',noPlate));
    set(handles.ExtractedEditText,'string',noPlate);
else
    msgbox('Unable to extract the characters from the number plate.\n');
    msgbox('The characters on the number plate might not be clear or touching
with each other or boundries.\n');
end


function r=controlling(NR)

[Q,W]=hist(NR(:,4));
ind=find(Q==6);

for k=1:length(NR)
    C_5(k)=NR(k,2) * NR(k,4);
end
NR2=cat(2,NR,C_5');
[E,R]=hist(NR2(:,5),20);
Y=find(E==6);
if length(ind)==1
    MP=W(ind);
    binsize=W(2)-W(1);
    container=[MP-(binsize/2) MP+(binsize/2)];
    r=takeboxes(NR,container,2);
elseif length(Y)==1
    MP=R(Y);
    binsize=R(2)-R(1);
    container=[MP-(binsize/2) MP+(binsize/2)];
    r=takeboxes(NR2,container,2.5);
elseif isempty(ind) || length(ind)>1
    [A,B]=hist(NR(:,2),20);
    ind2=find(A==6);
    if length(ind2)==1
        MP=B(ind2);
        binsize=B(2)-B(1);
        container=[MP-(binsize/2) MP+(binsize/2)];
        r=takeboxes(NR,container,1);
    else
        container=guessthesix(A,B,(B(2)-B(1)));
        if ~isempty(container)
            r=takeboxes(NR,container,1);
```

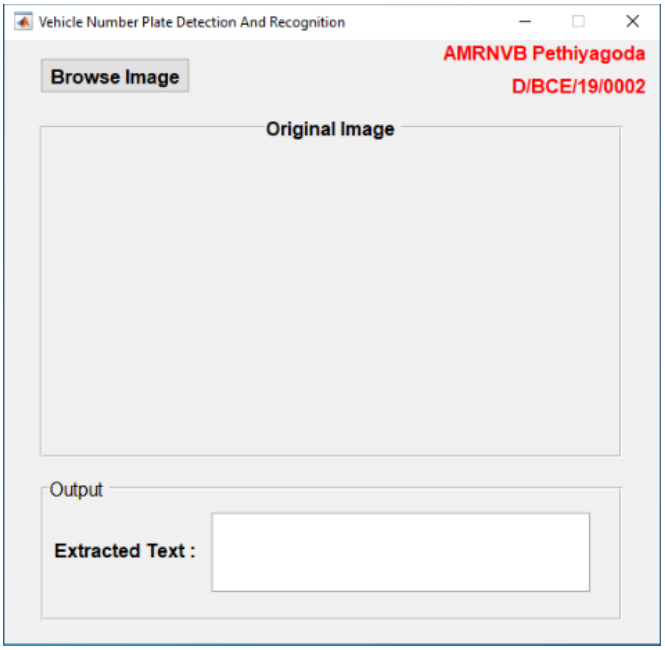
```
        elseif isempty(container)
            container2=guessthesix(E,R,(R(2)-R(1)));
            if ~isempty(container2)
                r=takeboxes(NR2,container2,2.5);
            else
                r=[];
            end
        end
    end
end


function container=guessthesix(Q,W,bsize)

for l=5:-1:2
    val=find(Q==l);
    var=length(val);
    if isempty(var) || var == 1
        if val == 1
            index=val+1;
        else
            index=val;
        end
        if length(Q)==val
            index=[];
        end
        if Q(index)+Q(index+1) == 6
            container=[W(index)-(bsize/2) W(index+1)+(bsize/2)];
            break;
        elseif Q(index)+Q(index-1) == 6
            container=[W(index-1)-(bsize/2) W(index)+(bsize/2)];
            break;
        end
    else
        for k=1:1:var
            if val(k)==1
                index=val(k)+1;
            else
                index=val(k);
            end
            if length(Q)==val(k)
                index=[];
            end
            if Q(index)+Q(index+1) == 6
                container=[W(index)-(bsize/2) W(index+1)+(bsize/2)];
                break;
            elseif Q(index)+Q(index-1) == 6
                container=[W(index-1)-(bsize/2) W(index)+(bsize/2)];
                break;
            end
        end
        if k~=var
            break;
        end
    end
end
```

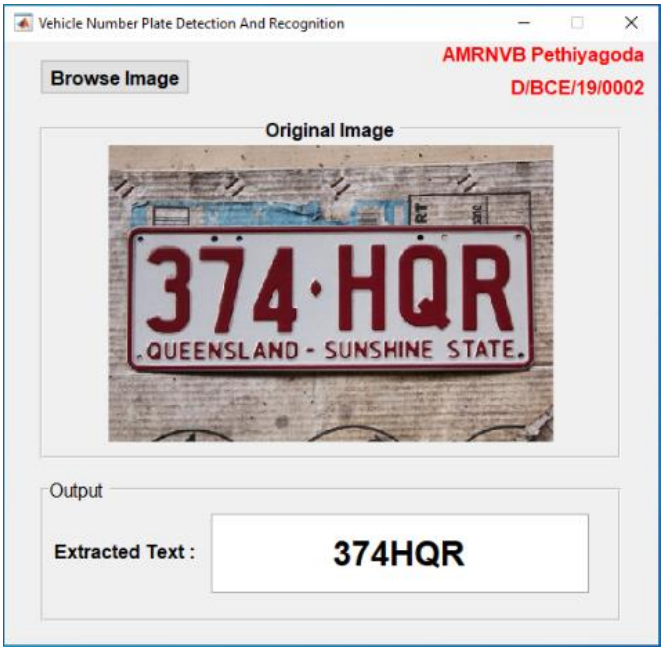
```
if l==2
    container=[];
end

function letter=readLetter(snap)

load NewTemplates
snap=imresize(snap,[42 24]);
comp=[ ];
for n=1:length(NewTemplates)
    sem=corr2(NewTemplates{1,n},snap);
    comp=[comp sem];
end
vd=find(comp==max(comp));
if vd==1 || vd==2
    letter='A';
elseif vd==3 || vd==4
    letter='B';
elseif vd==5
    letter='C';
elseif vd==6 || vd==7
    letter='D';
elseif vd==8
    letter='E';
elseif vd==9
    letter='F';
elseif vd==10
    letter='G';
elseif vd==11
    letter='H';
elseif vd==12
    letter='I';
elseif vd==13
    letter='J';
elseif vd==14
    letter='K';
elseif vd==15
    letter='L';
elseif vd==16
    letter='M';
elseif vd==17
    letter='N';
elseif vd==18 || vd==19
    letter='O';
elseif vd==20 || vd==21
    letter='P';
elseif vd==22 || vd==23
    letter='Q';
elseif vd==24 || vd==25
    letter='R';
elseif vd==26
    letter='S';
elseif vd==27
    letter='T';
elseif vd==28
    letter='U';
```

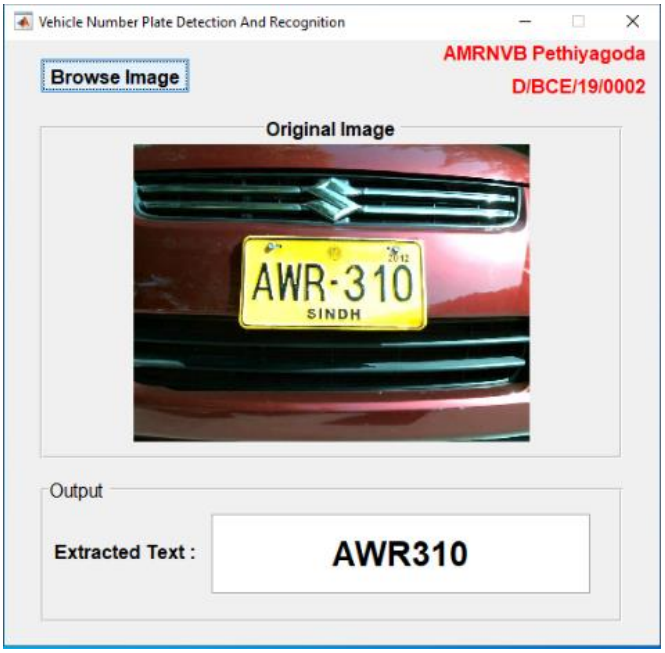
```
elseif vd==29
    letter='V';
elseif vd==30
    letter='W';
elseif vd==31
    letter='X';
elseif vd==32
    letter='Y';
elseif vd==33
    letter='Z';

elseif vd==34
    letter='1';
elseif vd==35
    letter='2';
elseif vd==36
    letter='3';
elseif vd==37 || vd==38
    letter='4';
elseif vd==39
    letter='5';
elseif vd==40 || vd==41 || vd==42
    letter='6';
elseif vd==43
    letter='7';
elseif vd==44 || vd==45
    letter='8';
elseif vd==46 || vd==47 || vd==48
    letter='9';
else
    letter='0';
end


function r=takeboxes(NR,container,chk)

takethisbox=[];
for i=1:size(NR,1)
    if NR(i,(2*chk))>=container(1) && NR(i,(2*chk))<=container(2)
        takethisbox=cat(1,takethisbox,NR(i,:));
    end
end
r=[];
for k=1:size(takethisbox,1)
    var=find(takethisbox(k,1)==reshape(NR(:,1),1,[]));
    if length(var)==1
        r=[r var];
    else
        for v=1:length(var)
            M(v)=NR(var(v),(2*chk))>=container(1) &&
NR(var(v),(2*chk))<=container(2);
        end
        var=var(M);
        r=[r var];
    end
end
```

```
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (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 edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (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 ExtractedEditText_Callback(hObject, eventdata, handles)
% hObject    handle to ExtractedEditText (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 ExtractedEditText as text
%        str2double(get(hObject,'String')) returns contents of
ExtractedEditText as a double


% --- Executes during object creation, after setting all properties.
function ExtractedEditText_CreateFcn(hObject, eventdata, handles)
% hObject    handle to ExtractedEditText (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
```

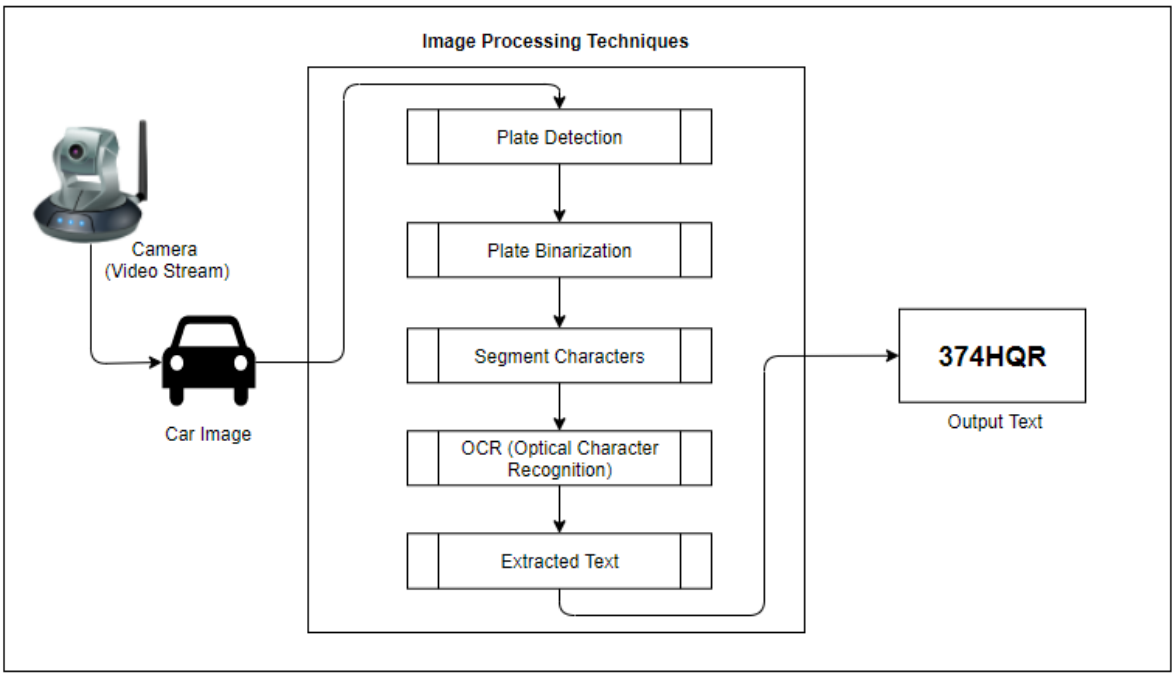
Output


Vehicle Number Plate Detection And Recognition
Browse Image
AMRNVB Pethiyagoda
D/BCE/19/0002
Original Image
Output
Extracted Text :


Vehicle Number Plate Detection And Recognition
Browse Image
AMRNVB Pethiyagoda
D/BCE/19/0002
Original Image
374·HQR
QUEENSLAND - SUNSHINE STATE.
Output
Extracted Text :
374HQR

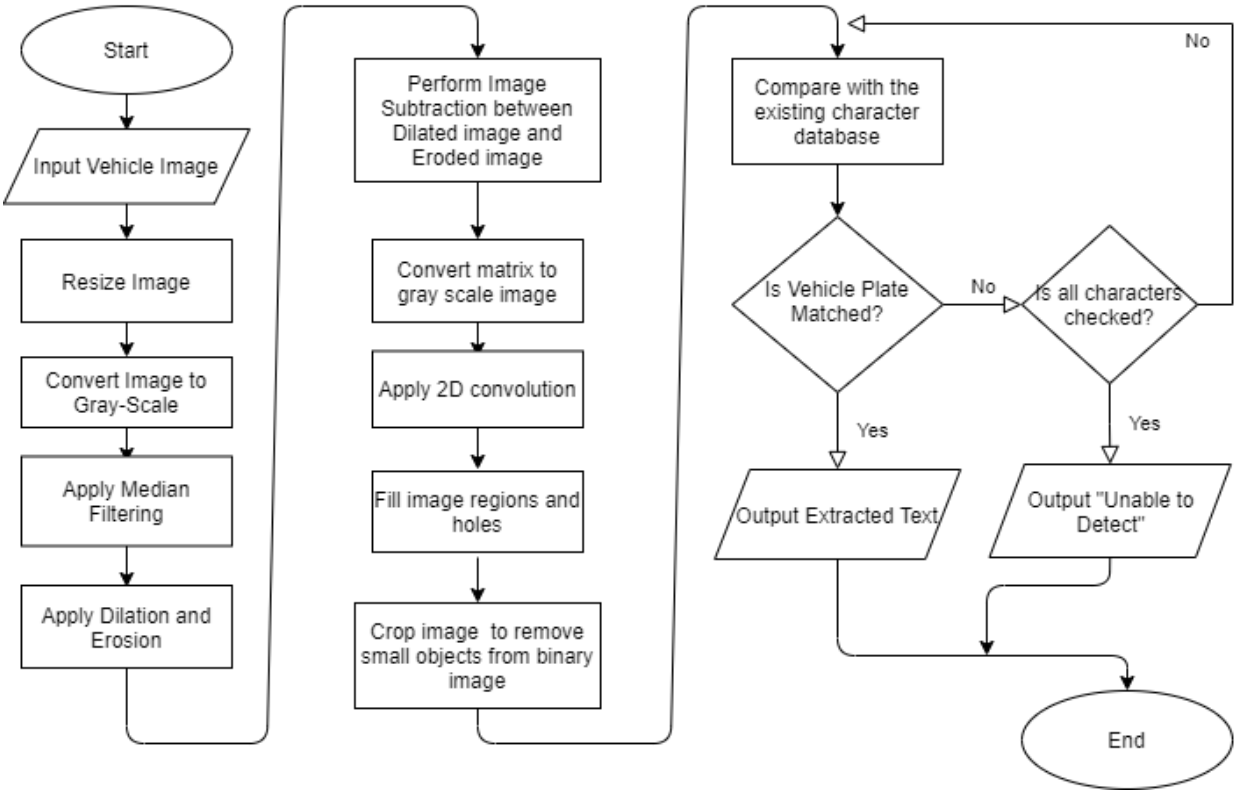

Vehicle Number Plate Detection And Recognition
Browse Image
AMRNVB Pethiyagoda
D/BCE/19/0002
Original Image
AWR-310
SINDH
Output
Extracted Text :
AWR310

a) System Architecture



Vechicle Number Plate Detection and Recognition Archiecture

b) Activity Diagram (Process flow chart)



c) Image Processing Techniques

The vehicle image is given as an image, the image is then resized and converts the true color image RGB to the **grayscale image**. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. Then performs **median filtering**, where each output pixel contains the median value in the **3x3 neighborhood** around the corresponding pixel in the input image.

A strel object is used to represent a flat **morphological structuring element**, which is an essential part of **morphological dilation and erosion operations**. Then **Image Dilation and Erosion** is applied. The two obtained images are **subtracted** with each elesment in array ge from the corresponding element in array gi and returns the difference in the corresponding element of the output array gdiff. Then sets the values to the minimum and maximum values in gdiff.

The **two-dimensional convolution** of matrices is returned. The intensity values of image are adjusted. The image regions and holes are filled. The image is then **cropped** removing small objects from binary image.

Finally, the characters are cropped to obtain them individually and compare them with the generated alphanumeric characters to see if they match. The process continues until all characters are checked. If the suitable character is not found, then display a message that the number could not be identified. If all the characters are identified successfully then extract the individual characters concatenate them into a string and display it.