*Recognition of vehicle number plate using MATLAB*

Video surveillance nowadays is used everywhere due to the decreasing costs of high quality surveillance systems, it’s used for home security, Military application, banking\ATMs, traffic monitoring etc.

Accordingly, automated systems have been designed for numerous detection tasks.

***I.INTRODUCTION***

Number plates are used around the world to identify vehicles , They’re identified either manually or automatically.

Automatic vehicle identification is done through image processing , automatic identification is used mostly for the purpose of effective traffic control and security application such as access control or restricted areas or tracking wanted vehicles. Number plate recognition (NPR) is hard to develop for general use as each country has it’s own special number plate system and layout.

Experimentation with NPR has been conducted over many years, yet it’s still a challenging task. Number plate detection system investigates an input image to identify some local patches containing the license plate. Since a plate can be anywhere in an image with different sizes. it’s necessary to check every pixel of the image to locate it.

In parking, number plates are used to calculate the duration of the parking (e.g., when a vehicle enters a gate, the number plate is automatically recognized and stored in a database).

In an NPR system spectral analysis approach is used while acquiring the image, extracting the region of interest , segmenting the characters using different feature extraction techniques.

A close up of text on a white background

Description automatically generated

The goal of my project is (as shown in the diagram) to extract the location of a number plate, segment it , recognize the characters and display the number with relevant information .

***Illiterate study***

For this project I read a couple of articles online and Master thesis for inspiration and guidance:

The first one I read was:

International Journal of Computer Applications (0975 – 8887)

Volume 134 – No.1, January 2016

From <https://www.researchgate.net/publication/290787583_Vehicle_Number_Plate_Recognition_System_A_Literature_Review_and_Implementation_using_Template_Matching>

I took the liberty of taking some code from this article(in my template matching and segmentation functions )

And A Thesis presented to the Faculty of the Graduate School at the University of Missouri by XIN LI

Due to the overpopulation in regions like china, India and small geographical areas of some countries like the Netherlands a lot of research has been conducted on the matter concluding that an automatic number plate recognition software is needed to manage the traffic effectively. During the research In china, 2002 they discovered that the same system could be used in a lot of cases like in parking, video surveillance, etc.

Most number plate recognition systems use the same diagram

A close up of a device

Description automatically generated

It’s a very simple yet effective system that has been implemented a lot throughout the years. But the problem with this system is that it’s not 100% accurate, and in some cases it could output the wrong results.

Some of the feasible difficulties they had were:

1.broken or defected number plate

2.blurry image

3.plate not with in legal specifications (personalized font or layout \special characters not with in legal specifications)

4.low resolution when cropping and zooming in on the number plate (bad quality cameras)

5.poor maintenance of the number plate from the owners side which can result in similarities between certain characters (e.g., O and D, 8 and B or 0 and O).

6.moving cars (moving car plate number that results in blurry useless images )

But the government could help by implementing strict laws that dictate how much a person could personalize his number plate(e.g., what fonts to use and what background colours)

The second article I read was:

Wikipedia, <https://en.wikipedia.org/wiki/Automatic_number-plate_recognition>

🡪ANPR was invented in 1976 at the Police Scientific Development Branch in Britain. Prototype systems were working by 1979, and contracts were awarded to produce industrial systems, first at EMI Electronics, and then at Computer Recognition Systems (CRS) in [Wokingham](https://en.wikipedia.org/wiki/Wokingham), UK. Early trial systems were deployed on the [A1 road](https://en.wikipedia.org/wiki/A1_road_(Great_Britain)) and at the [Dartford Tunnel](https://en.wikipedia.org/wiki/Dartford_Tunnel). The first arrest through detection of a stolen car was made in 1981. However, ANPR did not become widely used until new developments in cheaper and easier to use software were pioneered during the 1990s. The collection of ANPR data for future use (*i.e*., in solving then-unidentified crimes) was documented in the early 2000s. The first documented case of ANPR being used to help solve a murder occurred in November 2005, in [Bradford](https://en.wikipedia.org/wiki/Bradford), UK, where ANPR played a vital role in locating and subsequently convicting killers of [Sharon Beshenivsky](https://en.wikipedia.org/wiki/Murder_of_Sharon_Beshenivsky), which was proof of their usefulness when used by the law.

Of course it’s not just identifying killers in runaway vehicles, the automatic number plate recognition system (ANPR) was useful in day to day life cases too like identifying the number plates of vehicles that drove too fast on the highway, this was achieved through the availability of ANPR in compact systems with more processing speed due to the need for quick processing in cases of vehicles driving faster than 160km\h.

The way they’ve done it was through image processing on images taken from traffic cameras, segmenting those photos and matching the number plate with the ones in the data base until they get a high correlation match, Of course each algorithm for ANPR is tailored to the number plates of that specific country.

-The cameras used can be existing road-rule enforcement or closed-circuit television cameras, as well as mobile units, which are usually attached to vehicles. Some systems use infrared cameras to take a clearer image of the plates.

One the biggest concerns of these systems nowadays with the increasing number of cameras on the road and the technological advancements in video surveillance field is privacy:

Is it ethical to survey people everywhere they go, each time they use their cars ? to gather data without their permission ?

And accuracy:

Will it always work? even with low quality cameras and bad conditions (dark images, reflection of the sun or any light source on the number plate )?

and are these systems failproof ?couldn’t somebody just change his number plate illegally to match another persons number plate and cause him trouble ?

***III. concept of operations***

Automatic Number Plate Recognition , ANPR in short, is a system in high demand in todays markets.

Every new shop opening, every house being built ,in every parking , in each intersection or roundabout, there’s at least one camera either for security or managing traffic reasons. If we would implement an ANPR system in some of them it would go a long way in making life easier, more convenient and more secure.

The stakeholders for projects like these would be:

-Companies

-Governments (especially in big cities, like Amsterdam)

-Paid parking

1.Companies would use the ANPR to monitor the entrances, who’s getting in and at what time, it would save them a lot of money because they wouldn’t have to hire a lot of security guards.

2.Goverments would use this technology to monitor the public on the street, for example somebody driving by a red light, they would be able to identify and penalize the driver of that car or in case of a fleeing vehicle, they’ll able to catch the driver really easily, if not they’d have a way to follow up with the case . It can even be used to know if somebody has driven through a specific street at a specified time or not, it would go a long way in helping the police and law officers and making their daily tasks a lot easier.

In conclusion the government would implement the technology in surveillance, security, safety, information gathering and traffic managements.

3.In parking they could use the system to automate the whole parking experience of opening \closing the gates and charging the driver based on how many hours his car stayed inside (by subtracting the time the vehicle got inside from the time it got out).

I followed this diagram during my project:

A screenshot of a cell phone

Description automatically generated

***IV. System requirements***

Computer vision and character recognition, algorithms for

license plate recognition play an important role in video

analysis of the number plate image. Therefore they form the

core modules in any ANPR system.

Vehicle identification has been an active research over the last few years. A number of researches have been carried out

to identify the type of vehicle such as a car, truck, scooter or

motorcycle. My research focuses on identifying European number plates and how would that be implemented in a real life fully realized system .

My project is made as a very condensed version of what the police or security in most countries around the world are using these days to identify wanted vehicles or to look for a specific vehicle.

-Use case diagram:

A close up of a map

Description automatically generated

-Activity diagram:

A close up of a map

Description automatically generated

Acquire image

***V. system design***

Through testing and research I gathered the necessary data and knowledge about the project field and history.

The system for automatic car license plate recognition includes a camera, a frame grabber, a computer, and custom designed software for image processing, analysis and recognition.

The software is based on the article published online:

International Journal of Computer Applications (0975 – 8887)

Volume 134 – No.1, January 2016

The classification algorithm I used is dependent on template matching and correlation between the segmented parts of the license plate image and the templates, I made 6 functions in total

First one (number plate extract.m) is the one responsible for taking the input, processing it, running it through all the other functions and outputting the result at the end.

function numberPlateExtract

%NUMBERPLATEEXTRACT extracts the characters from the input number plate image.

% Reading the number plate image or from a snapshot from video.

% imP =imread('confusion2.jpg','jpg');

vid = videoinput('winvideo',1);

start(vid);

preview(vid);

imP = getsnapshot(vid);

stop(vid);

delete(vid);

subplot (2,1,1);

imshow(imP);

im2=imresize (imP, [400 NaN]); % Resizing the image keeping same aspect ratio

im3=rgb2gray(im2); % Converting the RGB (color) image to gray (intensity).

im3=medfilt2(im3, [3 3]); % Median filtering to remove noise.

se=strel('disk',1); % Structural element (disk of radius 1) for morphological processing.

gi=imdilate(im3, se); % Dilating the gray image with the structural element.

ge=imerode(im3, se); % Eroding the gray image with structural element.

gdiff=imsubtract(gi, ge); % Morphological Gradient for edges enhancement.

gdiff=mat2gray(gdiff); % Converting the class to double.

gdiff=conv2(gdiff,[1 1;1 1]); % Convolution of the double image for brightening the edges.

gdiff=imadjust(gdiff,[0.5 0.7],[0 1],0.1); % Intensity scaling between the range 0 to 1.

Binary=logical(gdiff); % Conversion of the class from double to binary.

% Eliminating the possible horizontal lines from the output image of regiongrow

% that could be edges of license plate.

er=imerode(Binary,strel('line',50,0));

out1=imsubtract(Binary,er);

% Filling all the regions of the image.

F=imfill(out1,'holes');

% Thinning the image to ensure character isolation.

H=bwmorph(F,'thin',1);

H=imerode(H,strel('line',3,90));

% Selecting all the regions that are of pixel area more than 100.

final=bwareaopen(H,100);

% Two properties 'BoundingBox' and binary 'Image' corresponding to these

% Bounding boxes are acquired.

Iprops=regionprops(final,'BoundingBox','Image');

% Selecting all the bounding boxes in matrix of order numberofboxesX4;

NR=cat(1,Iprops.BoundingBox);

% Calling of control function.

r=control(NR); % Function 'control' outputs the array of indices of boxes required for extraction of characters.

if ~isempty(r) % If succesfully indices of desired boxes are achieved.

I={Iprops.Image}; % Cell array of 'Image' (one of the properties of regionprops)

noPlate=[]; % Initializing the variable of number plate string.

for v=1:length(r)

N=I{1,r(v)}; % Extracting the binary image corresponding to the indices in 'r'.

letter=readLetters(N); % Reading the letter corresponding the binary image 'N'.

while letter=='O' || letter=='0' % Since it's not easy to distinguish

if v<=3 % between '0' and 'O' during the extraction of character

letter='O'; % in binary image. Using the characteristic of european plates

else % that starting three characters are alphabets, this code will

letter='0'; % easily decide whether it is '0' or 'O'. The condition for 'if'

end % just need to be changed if the code is to be implemented with some other

break; % cities plates. The condition should be changed accordingly.

end

noPlate=[noPlate letter]; % Appending every subsequent character in noPlate variable.

end

%fid = fopen('noPlate.txt', 'wt'); % This portion of code writes the number plate

%fprintf(fid,'%s\n',noPlate); % to the text file, if executed a notepad file with the

%fclose(fid); % name noPlate.txt will be open with the number plate written.

%winopen('noPlate.txt')

if noPlate == 'RKL65JJ' %if the identified plate is any of these then show the information

disp('the plate number is: ');

disp(noPlate);

disp(' #############')

disp(' issued the 04-january-1997 in Warsaw')

disp(' #############')

disp(' registered under the name: Emilek Dymek')

disp(' #############')

disp(' Model: Marcedes-Benz Sprinter')

im = imread('MarcedesSprinter.jpg','jpg');

subplot(2,1,2);

imshow(im);

elseif noPlate == '8A86365'

disp('the plate number is: ');

disp(noPlate);

disp(' #############')

disp(' issued the 06-may-2011 in Prague')

disp(' #############')

disp(' registered under the name: Jarmila Jana')

disp(' #############')

disp(' Model: Toyota Sienna')

im = imread('ToyotaSienna.jpg','jpg');

subplot(2,1,2);

imshow(im);

elseif noPlate == '1AY9904'

disp('the plate number is: ');

disp(noPlate);

disp(' #############')

disp(' issued the 25-march-2017 in Pilson')

disp(' #############')

disp(' registered under the name : Elon Ida')

disp(' #############')

disp(' Model: Toyota Prius')

im = imread('ToyotaPrius.jpg','jpg');

subplot(2,1,2);

imshow(im);

else

disp('number plate:')

disp(noPlate)

disp('is not registered in the system.')

end

else % If fail to extract the indexes in 'r' this line of error will be displayed.

disp('Unable to extract the characters from the number plate.');

disp('check if the lighting is too bright or too dark or the characters are hard to identify so try with a different ones');

end

end

Second one (creating templates’) is pretty self explanatory: it creates the templates that are going to be used for the identification of the characters.

%CREATING TEMPLATES

%Letters(A..Z)

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');

%Numbers(0..9)

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');

line = imread('--.PNG','PNG');

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 line];

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

Third one (segmentation.m) segments the input image to several with each letter having its own image.

function r=segmentation(NR,container,chk)

% helps in determining the values of indices of interested Bounding boxes.

% outputs the value of indices corresponding to

% the desired Bounding boxes. NR is the numberofregionsx4 matrix of all the

% regions Bounding boxes. CONTAINER is the width of the bin that contain

% all the seven bounding boxes of interest. CHK will determine whether

% bounding boxes are y-dimension width's wise grouped or y-coordinate wise

% grouped. CHK = 2 considers y-dimension width grouping and CHK=1 considers

% y-coordinate grouping.

%segments the input image to several with each letter having its own image

takethisbox=[]; % Initialize the variable with an empty matrix.

for i=1:size(NR,1)

if NR(i,(2\*chk))>=container(1) && NR(i,(2\*chk))<=container(2) % If Bounding box is among the container.

takethisbox=cat(1,takethisbox,NR(i,:)); % then Take that box and concatenate along first dimension.

end

end

r=[];

for k=1:size(takethisbox,1)

var=find(takethisbox(k,1)==reshape(NR(:,1),1,[])); % Finding the indices of the interested boxes among NR

if length(var)==1 % because x-coordinate of the boxes will be unique.

r=[r var];

else % In case if x-coordinate is not unique

for v=1:length(var) % then check which box falls under container condition.

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

end

Fourth one (read letters.m) uses the templates and compares them with the input segmented images, there is a whole vector with correlation values for each letter, this function is the one assigned with outputting the correct character (with the highest correlation value ).

function letter=readLetters(snap)

% reads the character from the character's binary image.

% LETTER=READLETTERS(epsilon) outputs the character in class 'char' from the

% input binary image epsilon.

% matches the letters with the templates made

load NewTemplates % Loads the templates of characters in the memory.

snap=imresize(snap,[42 24]); % Resize the input image so it can be compared with the template's images.

comp=[ ];

for n=1:length(NewTemplates)

sem=corr2(NewTemplates{1,n},snap); % Correlation the input image with every image in the template for best matching.

comp=[comp sem]; % Record the value of correlation for each template's character.

end

vd=find(comp==max(comp)); % Find the index which corresponds to the highest matched character.

% Accodrding to the index assign to 'letter'.

% Alphabets

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';

% Numerals

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

end

Fifth one (guess the seven.m) runs through the seven images of characters once .

function container=guesstheseven(Q,W,bsize)

% guesses the container for seven Bounding boxes.

% CONTAINER=GUESSTHESEVEN(Q,W,BSIZE) outputs the container for the desired

% Bounding boxes from the frequency row vector Q, row vector of mid

% points of classes in W and class size in BSIZE.

% basically goes through the segmented letters and puts them in a vector

for l=5:-1:2

val=find(Q==l); % Find the indices corresponding the value of frequency equals 'l'.

var=length(val); % Check how many indices are found.

if isempty(var) || var == 1 % If no index or one index is found.

if val == 1

index=val+1; % index 0 is not allowed in MATLAB.

else

index=val; % Assign that value to 'index'.

end

if length(Q)==val % In case if the last index value is reached,

index=[]; % then index+1 will be out of Q.

end

if Q(index)+Q(index+1) == 7 % If the sum of frequencies with the subsequent bin equals seven.

container=[W(index)-(bsize/2) W(index+1)+(bsize/2)]; % Calculae container and break looping

break; % for more values.

elseif Q(index)+Q(index-1) == 7 % If the sum of frequencies with the previous bin equals seven.

container=[W(index-1)-(bsize/2) W(index)+(bsize/2)]; % Calculate container and break looping

break; % for more values.

end

else % If more than one index are found.

for k=1:1:var % Repeat the analysis for every value of the bin and checks for the same condition

if val(k)==1

index=val(k)+1; % 0 index is not allowed in MATLAB.

else

index=val(k); % that where the sum of frequencies equals seven.

end

if length(Q)==val(k) % In case if the last index value is reached,

index=[]; % then index+1 will be out of Q.

end

if Q(index)+Q(index+1) == 7

container=[W(index)-(bsize/2) W(index+1)+(bsize/2)]; % Calculate the value of container and break.

break;

elseif Q(index)+Q(index-1) == 7

container=[W(index-1)-(bsize/2) W(index)+(bsize/2)];

break;

end

end

if k~=var % If for any value of index classes frequencies sum to seven then just break.

break;

end

end

end

if l==2 % If looping is done and no frequencies sum to seven then assign container the empty matrix.

container=[];

end

end

Sixth one (control.m) makes sure that there’s 7 characters in the license plate image .

function r=control(NR)

% determines the array of indices of Bounding boxes of interest.

% outputs the row vector R containing the indices of

% the bounding boxes of interest from the matrix NR. NR is the matrix of

% order numberofregionsx4. Numberofregions are the total number of

% regions extracted from the function regionprops.To ensure the order cat(1,...) function could be used.

% The code for this function emphasizes on obtaining the indices of

% Bounding boxes whose width along the y-dimension is nearly same. If

% the approach of y-width doesn't work then Bounding Boxes with nearly

% same y-coordinates are obtained.

% checks if there is seven letters or not

[Q,W]=hist(NR(:,4)); % Histogram of the y-dimension widths of all boxes.

ind=find(Q==7); % Find indices from Q corresponding to frequency '7'.

% Since the number plates of cars in europe have mostly seven characters so

% find(Q==7) is used.

for k=1:length(NR) % using the uniqueness of y-co

C\_5(k)=NR(k,2) \* NR(k,4); % ordinate and y-width.

end

NR2=cat(2,NR,C\_5'); % Appending new column in NR.

[E,R]=hist(NR2(:,5),20);

Y=find(E==6); % Searching for seven characters.

if length(ind)==1 % If seven boxes of interest are succesfully found record

MP=W(ind); % the midpoint of corresponding class.

binsize=W(2)-W(1); % Calculate the container size.

container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete container size.

r=segmentation(NR,container,2);%calling the function segmentation

elseif length(Y)==1

MP=R(Y);

binsize=R(2)-R(1);

container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete container size.

r=segmentation(NR2,container,2.5); % Call function segmentation.

elseif isempty(ind) || length(ind)>1 % If there is no value of '7' in the Q vector.

[A,B]=hist(NR(:,2),20); % Use y-coordinate approach only.

ind2=find(A==7);

if length(ind2)==1

MP=B(ind2);

binsize=B(2)-B(1);

container=[MP-(binsize/2) MP+(binsize/2)]; % Calculating the complete container size.

r=segmentation(NR,container,1);

else

container=guesstheseven(A,B,(B(2)-B(1))); % Call of function guesstheseven.

if ~isempty(container) % If guesstheseven works succesfully.

r=segmentation(NR,container,1); % Call the function segmentation.

elseif isempty(container)

container2=guesstheseven(E,R,(R(2)-R(1)));

if ~isempty(container2)

r=segmentation(NR2,container2,2.5);

else

r=[]; % Otherwise assign an empty matrix to 'r'.

end

end

end

end

end

***VI. Testing***

I tested my system for the first time using condensed 3 digits license plate recognition and it was surprisingly hard, a lot of characters were very similar and some numbers were hard to distinguish but I managed to make it work after changing the templates I used the first time, then I tested the system with 6 digit license plate and it couldn’t figure out the letters that were too close or with different font so I made the segmented area a bit larger but that resulted in mistakenly reading two letters or numbers as one (IJ as D or ea. as O), after resetting the size of the boxes I had some trouble with the conditions of the input image (too bright , too dark, there’s a reflection, etc) with those I hit a rock solid wall, so I tried only testing number plates with neutral conditions, I took some pictures of Dutch number plates while I was hanging around in Amsterdam centrum, but sadly my algorithm even after changing it a lot still hasn’t been able to read dashes (-), so I tested it on other European countries number plates. These were the plates I tested my system on:

A close up of a sign

Description automatically generatedA picture containing car, outdoor

Description automatically generatedA black sign with white text

Description automatically generatedA close up of a sign

Description automatically generated

A close up of a sign

Description automatically generatedA silver car

Description automatically generated

A picture containing car, sitting, outdoor, meter

Description automatically generated

I tested it with several European license plates from all over the region.

I evaluated my algorithm through a predicted to true ratio, it was 71% accurate.

A close up of a building

Description automatically generated

A picture containing white, wall

Description automatically generatedthis confusion matrix is in a separate Excel file attached to this one.

[confusion matrix.xlsx](https://d.docs.live.net/8c1a2dea7793b5cd/Bureau/studies/matlab%20project/confusion%20matrix.xlsx)

I tried to make that number go all the way up to 100% but it was impossible for me because I’m without a group so I had to do all the work on my own with no one to consult or ask for advice.

I didn’t test the ability of the system working in a very brightly (like in sunny day) environment nor a very dim one (like a parking lot).

I didn’t have the time to work on it properly because I was alone on this project and it was really hard for me to do everything by myself and the internet didn’t provide much to help me figure out a way to make it more accurate and work in more situations like detecting and recognizing moving license plates or more challenging conditions.

***VII. Evaluation***

The Automatic License (or number) plate recognition system, through a lot of testing and research I discovered that classification, can be achieved via template matching using correlation values which made it a lot easier for me to focus on segmenting, analysing and comparing the characters to achieve better accuracy.

Through trial and error I discovered that it’s still hard to differentiate between similar letter or numbers, now reflecting on the whole project I think that I achieved some sort of a plate recognition software that works while interacting with the real world, even though it doesn’t work 100% of the time I’m still proud of it , if I had to do it again I’d like having a group so that I can expend on this project and make it work more than 90% of the time, my planning because I was alone wasn’t really that important so I just figured it out as I go. I encountered some technical problems that I didn’t expect like the need for a faster processor to make the program work instantly and for a better camera because the light bouncing from the pictures is making the characters hard to see for the camera.

All in all, it was a great way to test my ability to get work done while under pressure, and I think I did okay.