International Islamic University Islamabad

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

DIGITAL IMAGE PROCESSING

COMPLEX ENGINEERING PROBLEM REPORT

Group Members: (Name & Registration No.)

Haseeb Mehmood 622-FET/BSEE/F19

Mohid Aamir 663-FET/BSEE/F19

Introduction

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class (such as humans, buildings, or cars) in digital images and videos.[1] Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

It is widely used in computer vision tasks such as image annotation,[2] vehicle counting,[3] activity recognition,[4] face detection, face recognition, video object co-segmentation. It is also used in tracking objects, for example tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video.

Every object class has its own special features that helps in classifying the class – for example all circles are round. Object class detection uses these special features. For example, when looking for circles, objects that are at a particular distance from a point (i.e. the center) are sought. Similarly, when looking for squares, objects that are perpendicular at corners and have equal side lengths are needed. A similar approach is used for face identification where eyes, nose, and lips can be found and features like skin color and distance between eyes can be found.

Motivation

Motivation: For the standard number plates the automatic number plate recognition becomes very easy to read and recognizes the character. In India the vehicle number plates has no standard size and font so it become very difficult to read and recognize the characters of the number plate. So flexible algorithm required solve this problem. The first automatic number plate recognition system was developed in 1976 by the Police Scientific Development Branch in Britain. However, since its inception, with the exponential growth in the number of vehicles on the road and high-speed imaging devices, there is now more potential for a number plate recognition system than there ever was.

Number Plate recognition plays an important role in various applications such as traffic monitoring on road, automatic toll payment, parking lots access control, detection of stolen vehicles. To identify a car number plate is effective because of its uniqueness of the car. Real time number plate recognition plays an important role in automatic monitoring of traffic rules. The recognition of car number plate can be used for automatic car parking because each car has its own identification number.

Literature Review

A paper proposed by Al-amri *et al.* [5] in 2010 compares various edge detectors. One of the most important uses of edge detection technique is for image segmentation. An image is divided into various regions which can further be classified as foreground and background. An edge is generally associated with a significant change in the intensity of the image or some discontinuity. Discontinuity in a grayscale image can be of three types - point, line, edge, and gradient.
Soundrapandiyan and Mouli (2015) suggested a novel and adaptive method for pedestrian detection. Further, they separated the foreground objects from the background by image pixel intensities. Subsequently, they used high boost filter for enhancing the foreground edges. The efficacy of the proposed method is evident from the subject evaluation results as well as objective evaluation with around 90% of pedestrian's detection rate compared to the other single image existing methods. In future, they planned to improve the performance of the method with higher detection rate and low false positives on par with sequence image methods.
Ramya and Rajeswari (2016) suggested a modified frame difference method which uses the correlation between blocks of current image and background image to categorize the pixels as foreground and background. The blocks in the current image which are highly correlated with the background image are considered as background. For the other block, the pixel-wise comparison is made to categorize it as foreground or background. The experiments conducted proved this approach improves the frame difference method particularly as finding accuracy with speed. However, this study needs to focus towards other information available in the blocks such as shape and edge can be used to improve the detection accuracy.

Risha and Kumar (2016) suggested an optic flow with the morphological operation for object detection in video. Further applied morphological operation towards *A Review of Detection and Tracking of Object from Image and Video Sequences* 749
Image segmentation method based on region includes: Region growing method, Region splitting & merging method and Clustering method. Segmentation based on edge detection includes: Ray histogram technique, Gradient based method, first order derivative method (Prewitt operator, Sobel operator, Canny operator) and Second order derivative method which has Laplacian operator and Zero crossing. Other type of segmentation is done through thresholding methods like Global thresholding and Local thresholding.

Akanksha et al. [11] proposed another work that represents importance of segmentation for object recognition. Segmentation is defined as separating out the regions with similar textures, intensity etc. with respect to background. Then the objects with same class are kept together, so as a result, the objects are separated according to the classes. There are several image segmenting techniques like Otsu's method and K-means clustering

P. Janani *et al.* [12], image processing techniques for spatial domain are discussed. They proposed that by adding certain type of noises, one can find out which method is efficient for which kind of noise. Different types of noises like Gaussian noise, Salt and pepper noise, Speckle noise, Poisson noise and quantization noise are dealt with several filters like: 1) Median filter: This filter removes the noise while preserving the edges, so each value of the matrix is replaced by the median value. 2) Mean filter: In this filter a mask is applied on each pixel to enhance the image quality. The mask is a collection of pixels that fall under same category. 3) Weiner filter: The Weiner filter is an old approach in which the values of the pixels are optimized to reduce the noise in the signal by comparing it with desired noiseless channel. To reduce overfitting and to improve performance on test images, a large training set is required to train deep learning techniques

The Haar Wavelet

Features used in this thesis for object detection were used for image retrieval (Jacobs et al., 1995) before being applied to pedestrian detection (Oren et al., 1997) and generalised to object detection (Papageorgiou et al., 1998). The features used by the Haar

The Haar Wavelet

Features used in this thesis for object detection were used for image retrieval (Jacobs et al., 1995) before being applied to pedestrian detection (Oren et al., 1997) and generalised to object detection (Papageorgiou et al., 1998). The features used by the Haar

Classifier Cascade algorithm were created for face detection (Viola & Jones, 2001b) but were later used for face recognition (Jones & Viola, 2003b). The common ground is the feature set – images can yield an enormous range of features, few of which are likely to discriminate between the object to detect and everything else that may appear in the image. Feature selection algorithms therefore receive a lot of attention.

One such method, the eigenface algorithm, finds features defining complete image regions. It was originally created for face recognition (Turk & Pentland, 1991), but is also effective at face detection (Popovici & Thiran, 2003).

Two other well-known
feature types are SIFT (Scale-Invariant Feature Transform), usually used for object recognition (Lowe, 1999), and Gabor Filters (Schiele & Crowley, 2000). There are also regular classifier algorithms which have been successfully applied with input provided directly from image pixel regions, such as neural networks (Rowley et al., 1998a).
The Haar Classifier Cascades studied in detail in this thesis use two-dimensional Haar Wavelet Features: simple rectangular patterns of light and dark.

Problem

With the onset of covid and the growing importance of social distancing and minimalizing human to human interactions, there is now an increasing demand for devices or systems that would reduce these interactions to cut down the risk of infection, whether it be covid or from any other disease . Furthermore, increasing the work that is to be made automatic would make it more efficient than having it done through human labour. We have seen a rise in such systems in the forms of cashless or online payments, online ticket booking, online shopping etc . But one sector which is almost completely operated by humans and suffers extremely from disorder and inefficiency is parking in buildings.
We also want to detect pedestarian so that it can be used for the safety of people crossing road ,streets,zebracrossing e.t.c

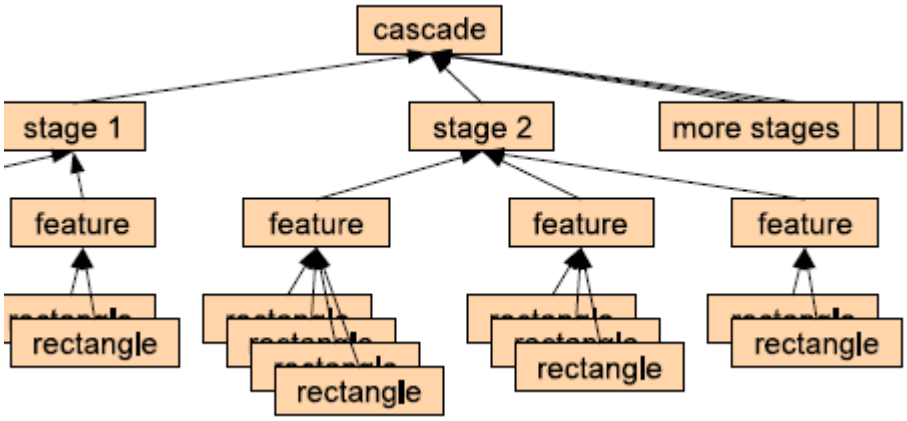
PROPOSED ALGORITHM

In order to recognize CARS, Number Plate and persons in image we are using **OPENCV** library for processing of image in python language.

There are multiple proposed algorithms which are used in this program

HAAR CASCADE METHOD TO RECOGNIZE CARS

1. A *cascade* steps through several *stages* in turn; it stops and returns false if a stage returns false. If all stages return true, the cascade returns true.
2. A *stage* returns true if the sum of its *feature* outputs exceeds a chosen threshold.
3. A *feature* returns the sum of its *rectangle* outputs.
4. A *rectangle* returns the sum of pixel values in the image region bounded by that rectangle, multiplied by a chosen weight.



CANNY FILTER AND CONTOUR TO DETECT NUMBER PLATE

The process of Canny edge detection algorithm can be broken down to five different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise
2. Find the intensity gradients of the image
3. Apply gradient magnitude thresholding or lower bound cut-off suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to str

HOG descriptor for person detection

Let's look at some important aspects of HOG that makes it different from other feature descriptors:

- The HOG descriptor focuses on the structure or the shape of an object. Now you might ask, how is this different from the edge features we extract for images? In the case of edge features, we only identify if the pixel is an edge or not. HOG is able to provide the edge direction as well. This is done by extracting the **gradient and orientation** (or you can say magnitude and direction) of the edges
- Additionally, these orientations are calculated in **'localized' portions**. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. We will discuss this in much more detail in the upcoming sections
- Finally the HOG would generate a **Histogram** for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

FLOW DIAGRAM

SUMMARY

In this we are taking image and converting it into grayscale and then smoothing it.To detect car in picture we are using cars data set to easily classify it sizes .Now the image is compared by this data. If car sized is found in image then a rectangle is made around it to show car is detected and labeled it with car tag. Later we are applying canny filter to find contours easily. If contours are found around the number plate then a rectangle is made around it and a new cropped image is generated also. Finally to detect a person we are using Histogram of oriented gradients to detect a person in image. After all things are detected we are displaying the output image.

LIBRARIES USED

```
import cv2
import imutils
import numpy as np
from matplotlib import pyplot as plt

from imutils.object_detection import non_max_suppression
```

CV2

CV2 is basically OPENCV library used to read and displaying images .Also it contains various filters for image processing.

IMULTIS

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib

NUMPY

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

MATPLOTLIB

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

IMUTILS.OBJECT_DETECTION IMPORT NON_MAX_SUPPRESSION

The image is scanned along the image gradient direction, and if pixels are not part of the local maxima they are set to zero.

CARS DATA SET

```
car_cascade = cv2.CascadeClassifier('cars.xml')
```

We are using opencv library to import cars shape dataset.

READING IMAGE FROM COMPUTER AND RESIZING

```
img = cv2.imread('C:\\Users\\mohid\\Music/sample.jpg',cv2.IMREAD_COLOR)
img = cv2.resize(img, (600,400) )
plt.figure(figsize=(10,20))
plt.imshow(img)


gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- First of all we reading image from Personal computer with the help of **cv2.imread** command.
- Now resizing the image into 600x400 pixels.
- With the help of matplotlib original image is being displayed.
- Later with **cv2.cvtColor** image is converted into gray scale image

CAR DETECTION

```
cars = car_cascade.detectMultiScale(gray, 1.1, 1)
ncars=0
for (x, y, w, h) in cars:
    a=cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 2)
    cv2.putText(a, 'car', (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12), 2)
    ncars = ncars + 1
```

detectMultiScale is used to detects objects of different sizes. Since we are using cars sizes datasets. It will detect any size in image which resembles with given data.

After car is detected we are using for loop to mask multiple car with rectangles if detected.

By **cv2.putText** we are labeling the rectangles with car title with Hershey font.

NUMBER PLATE DETECTION

```
gray = cv2.bilateralFilter(gray, 13, 15, 15)
edged = cv2.Canny(gray, 30, 200)
contours = cv2.findContours(edged.copy(), cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
contours = imutils.grab_contours(contours)
contours = sorted(contours, key = cv2.contourArea, reverse = True)[:10]
screenCnt = None
```

A **bilateral filter** is a non-linear, edge-preserving, and noise-reducing smoothing filter for images. It replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.

The **Canny edge detector** is an edge detection operator that uses a multi-stage algorithm to detect a wide range of edges in images. With opencv tool canny the bilateral image is converted into edges only picture which helps in finding contours.

Contour is an outline representing or bounding the shape or form of something. **cv2.findContours** is used to detect contours in canny image **and imutils.grab_contours** will get or store these contours. Later these contours are arranged in order.

FOR LOOP TO PERFORM OPERATIONS IF CONTOURS IS DETECTED OR NOT

```
for c in contours:

    peri = cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, 0.018 * peri, True)

    if len(approx) == 4:
        screenCnt = approx
        break

if screenCnt is None:
    detected = 0
    print ("No contour detected")
else:
     detected = 1

if detected == 1:
    cv2.drawContours(img, [screenCnt], -1, (0, 0, 255), 3)
    mask = np.zeros(gray.shape,np.uint8)
    new_image = cv2.drawContours(mask,[screenCnt],0,255,-1,)
    new_image = cv2.bitwise_and(img,img,mask=mask)
```

```
(x, y) = np.where(mask == 255)
(topx, topy) = (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))
Cropped = gray[topx:bottomx+1, topy:bottomy+1]
```

cv2.arcLength and **cv2.approxPolyDP** is used to approximate the contours and closing the approximate curves if detected.

If approximation is detected it means one number plate is detected. More the approximations more are the numberplates.

Similarly if no contour is detected a **"No contour detected "** message is displayed.

If contours is detected a border is made around it with **cv2.drawContours**.

Mask is generated with help of zero padding to apply on contour image to extract it as grayscale image.

New image is then bit wise and with original image to subtract all other parts except for numberplate area.

Now using min max from numpy the numberplate area is cropped.

DETECTING PERSON

```
HOGCV = cv2.HOGDescriptor()
HOGCV.c(cv2.HOGDescriptor_getDefaultPeopleDetector())
```

HOG, or Histogram of Oriented Gradients, is a feature descriptor that is often used to extract features from image data. The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features.

SVM works by mapping data to a high-dimensional feature space so that data points can be categorized, even when the data are not otherwise linearly separable.

By using these we are detecting human features from data set

FUNCTION

```
def Detector(frame):
    ## USing Sliding window concept
    rects, weights = HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8, 8),
scale=1.03)
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    pick = non_max_suppression(rects, probs=None, overlapThresh=0.65)
    c = 1
    for x, y, w, h in pick:
        cv2.rectangle(frame, (x, y), (w, h), (139, 34, 104), 2)
        cv2.rectangle(frame, (x, y - 20), (w,y), (139, 34, 104), -1)
        cv2.putText(frame, f'Person{c}', (x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255,
255), 2)
        c += 1

    cv2.putText(frame, f'Total Persons : {c - 1}', (20, 450), cv2.FONT_HERSHEY_DUPLEX,
0.8, (255, 255,255), 2)
    cv2.imshow('output', frame)
    return frame
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Frame is an input image we are providing. This image is then padded and scale for easily matching with data sets.

We are using for loops incase multiple peoples are detected in one frame. Then by using **cv2.rectangle** command we are putting rectangle around people and labeling it.

After with **cv2.imshow** we are displaying the output image.

LASTLY BUT CHIEFLY

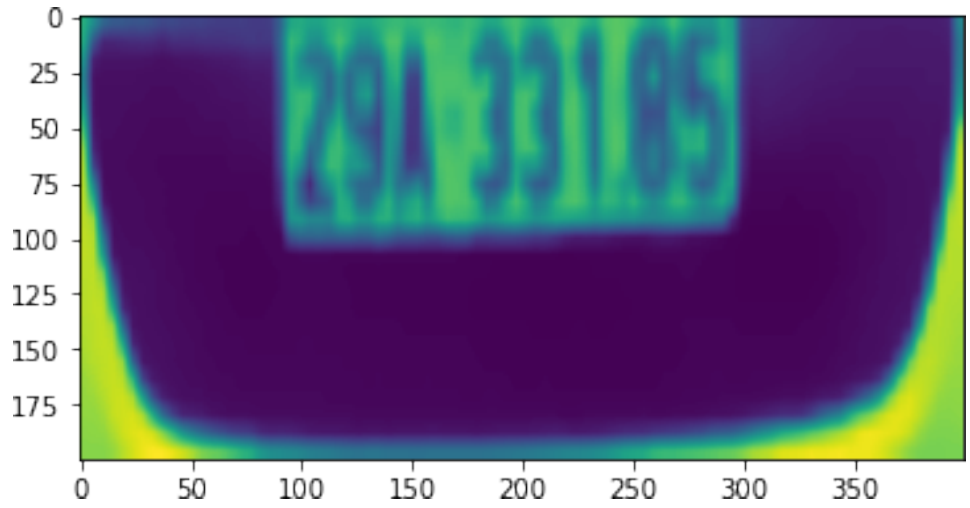
```
img = cv2.resize(img,(500,300))
Cropped = cv2.resize(Cropped,(400,200))
img = Detector(img)

cv2.imshow('Cropped',Cropped)

cv2.waitKey(0)
cv2.destroyAllWindows()
plt.figure(figsize=(10,20))
plt.imshow(img)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

Finally numberplate image is cropped and results are displayed. Also results are displayed using the plt command

INPUT IMAGE

OUTPUT IMAGE

And Number plate cropped image



CONCLUSION

As you can see from the output the algorithm has detected two cars and one person and number plate of front car. Its not 100% efficient as it is not fully AI based rather simple object detection with help of some data sets. As you can see it also detected motorcycle as car and not detected two persons behind it.

Similary numberplate was detected and cropped which appeared in separate image

In [23]:

```
import cv2
import imutils
import numpy as np

from matplotlib import pyplot as plt


from imutils.object_detection import non_max_suppression
```

COMPLETE CODE

In [24]:

```
car_cascade = cv2.CascadeClassifier('cars.xml')
```

In [25]:

```
img = cv2.imread('C:\\Users\\mohid\\Music/sample.jpg',cv2.IMREAD_COLOR)
img = cv2.resize(img, (600,400) )
plt.figure(figsize=(10,20))
plt.imshow(img)

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

In [26]:

```
cars = car_cascade.detectMultiScale(gray, 1.1, 1)
ncars=0
for (x, y, w, h) in cars:
    a=cv2.rectangle(img, (x,y), (x+w,y+h), (0,0,255), 2)

    cv2.putText(a, 'car', (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36,255,12),
2)
    nears = nears + 1
```

In [27]:
```
gray= cv2.bilatera1Filter(gray, 13, 15, 15)
edged= cv2.Canny(gray, 30, 200)
contours= cv2.findContours(edged.copy(), cv2.RETR TREE, cv2.CHAIN_APPROX_SIMPLE)
contours= imutils.grab_contours(contours)
contours= sorted(contours, key= cv2.contourArea, reverse= True)[:10]
screenCnt = None
```

In [28]:
```
for c in contours:


    peri  =  cv2.arclength(c, True)
    approx= cv2.approxPolyDP(c,0.018 * peri, True)

    if len(approx) == 4:
        screenCnt  = approx
        break

if screenCnt is None:
    detected= 0
    print ("No contour  detected")
else:
     detected= 1

if detected== 1:
    cv2.drawContours(img, [screenCnt], -1,(0, 0, 255), 3)
    mask= np.zeros(gray.shape,np.uint8)
    new_image     cv2.drawContours(mask,[screenCnt],0,255,-1,)
    new_image   cv2.bitwise and(img,img,mask=mask)
```

In [29]:
```
(x, y)    np.where(mask ==   255)
(topx, topy) =  (np.min(x), np.min(y))
(bottomx, bottomy) = (np.max(x), np.max(y))
Cropped=  gray[topx:bottomx+1,  topy:bottomy+1]
```

In [335…

In [30]:
```
HOGCV = cv2.HOGDescriptor()
HOGCV.setSVMDetector(cv2.HOGDescriptor getDefaultPeopleDetector())
```

In [31]:

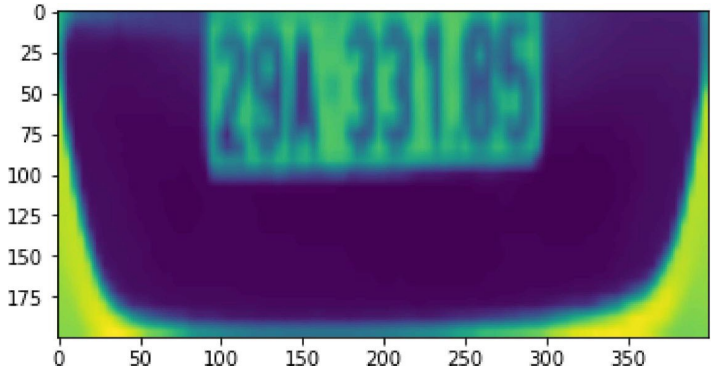
```
def Detector(frame):
    ## using Sliding window concept
    rects, weights= HOGCV.detectMultiScale(frame, winStride=(4, 4), padding=(8,
8), scale=1.03)
    rects = np.array([[x, y, x + w, y + h] for (x, y, w, h) in rects])
    pick= non_max_suppression(rects, probs=None, overlapThresh=0.65)
    c = 1
    for x, y, w, h in pick:
        cv2.rectangle(frame, (x, y),(w, h), (139, 34, 104), 2)
        cv2.rectangle(frame, (x,y - 20), (w,y), (139,34, 104), -1)
        cv2.putText(frame, f'Person{c}',(x, y), cv2.FONT_HERSHEY_SIMPLEX, 0.6,
(255,255,255), 2)
        c += 1


    cv2.putText(frame,£'Total Persons : {c - 1)',(20,450),
cv2.FONT_HERSHEY_DUPLEX, Q  (255,255,255),2)
    plt.imshow(img)
    return frame
```

In [32]:

```
img = cv2.resize(img,(500,300))
Cropped= cv2.resize(Cropped,(400,200))
img = Detector(img)


plt.imshow(Cropped)
plt.figure(figsize=(10,20))
plt.imshow(img)
```

Out[32]:

```
<matplotlib.image.AxesImage at 0x1ff2aa043d0>
```



In [ ]:

ANNEXURE

Above code is part of annexure and I have also added our code on github which I have provided link below

https://github.com/Arigatohaxroot/DIP-PROJECT-663-622-F19

REFERENCES

- https://github.com/duyet/opencv-car-detection/blob/master/Vehicle%20detection.ipynb
  https://github.com/Spidy20/Pedestrian_Detection_OpenCV
- https://medium.com/programming-fever/license-plate-recognition-using-opencv-python-7611f85cdd6c
- https://media.geeksforgeeks.org/wp-content/uploads/20200326001003/blurred.jpg
- https://en.wikipedia.org/wiki/Object_detection
- https://www.researchgate.net/publication/356148587_Automatic_Number_Plate_Recognition_System_for_Parking
- https://www.researchgate.net/publication/356148587_Automatic_Number_Plate_Recognition_System_for_Parking
- https://arxiv.org/pdf/1910.14255
- https://www.ripublication.com/ijcir17/ijcirv13n5_07.pdf
- https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/
- https://www.geeksforgeeks.org/python-haar-cascades-for-object-detection/