International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 07 | July - 2022 Impact Factor: 7.185 ISSN: 2582-3930

REPPURTUM OF LISENCE PLATE WITH IOT

Mrs M Buvanesvari[1], Mr R Vinoth Kumar[2], K Priyanga[3],V Preethi [4]

[1]Department of information technology & Rajiv Gandhi college of engineering and technology
[2]Department of information technology & Rajiv Gandhi college of engineering and technology
[3]Department of information technology & Rajiv Gandhi college of engineering and technology
[4]Department of information technology & Rajiv Gandhi college of engineering and technology

---

Abstract - This work addresses the problem of vehicle identification through non-overlapping cameras. As our main contribution, we introduce a novel dataset for vehicle identification, called Vehicle-Rear, that contains more than three hours of high-resolution videos, with accurate information about the make, model, color and year of nearly 3,000 vehicles, in addition to the position and identification of their license plates

Key Words: Vehicle identification, vehicle matching, multi-stream neural networks, feature fusion.

1.INTRODUCTION

The term digital image refers to processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two dimensional data. A digital image is an array of real or complex numbers represented by a finite number of bits. An image given in the form of a transparency, slide, photograph or an X-ray is first digitized and stored as a matrix of binary digits in computer memory. This digitized image can then be processed and/or displayed on a high-resolution television monitor. For display, the image is stored in a rapid-access buffer memory, which refreshes the monitor at a rate of 25 frames per second to produce a visually continuous display.

2. SCOPE OF PROJECT

With the ease and the advancement of communication nowadays, anyone can be a creator. In today's digital ecosystem, any digital content can be copied and distributed without loss of quality. The existing digital rights management system has not substantially reduced copyright infringements.

To track the content uniqueness and ownership we implementing a new decentralized system for registration, licensing and distribution of digital content using a blockchain solution. The execution of the system is taking place without the involvement of intermediaries. The solution utilizes the key features of the blockchain technology, Ethereum smart contracts, as well as the distributed ledger to achieve a decentralized, trusted, traceable, secure delivery of the digital content, with automatic payment and dispute handling.

3. MODULE DESCRIPTION

A. Image Processor

An image processor does the functions of image acquisition, storage, pre-processing, segmentation, representation, recognition and interpretation and finally displays or records the resulting image. The following block diagram gives the fundamental sequence involved in an image processing system.

B. Digital Computer

Mathematical processing of the digitized image such as convolution, averaging, addition, subtraction, etc. are done by the computer.

C. Mass Storage

The secondary storage devices normally used are floppy disks, CD ROMs etc

D. Hard Copy Device

The hard copy device is used to produce a permanent copy of the image and for the storage of the software involved.

E. Operator Console

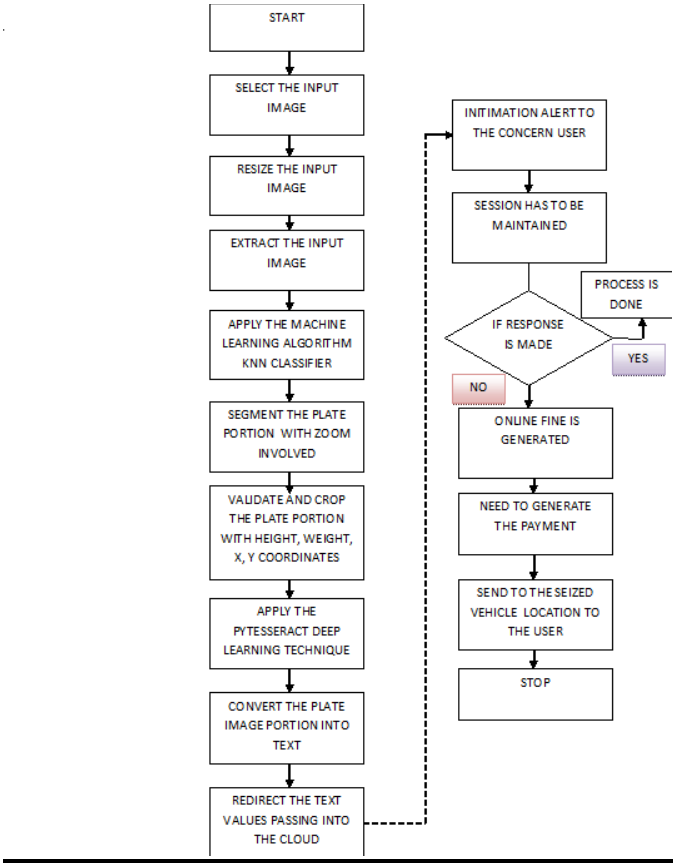
The operator console consists of equipment and arrangements for verification of intermediate results and for alterations in the software as and when require. The operator is also capable of checking for any resulting errors and for the entry of requisite data.

4. SYSTEM DESIGN

In this system design it has the 2 sections
• Flow Diagram
• Architecture Diagram

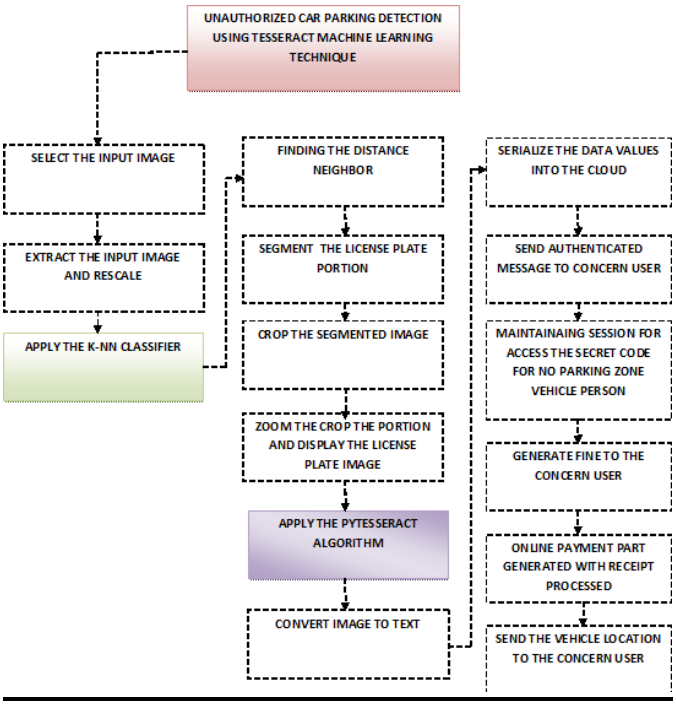
A. Flow Diagram

The flow diagram represents the process of detection of input image, applying Machine Learning, segmentation, applying the Deep Learning, conversion and identifying owner.



B. Architecture Diagram

The flow diagram represents the process of detection of input image, applying Machine Learning, segmentation, applying the Deep Learning, conversion and identifying owner.



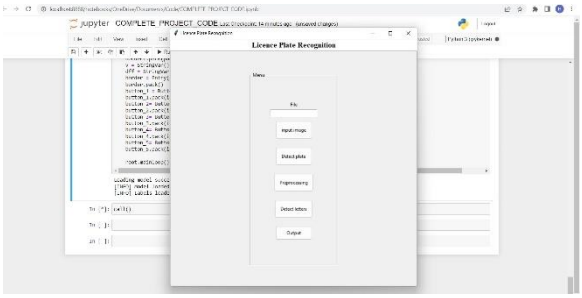
5. IMPLEMENTATION

```
# pre-processing input images and pedict with model
def predict_from_model(image,model,labels):
    image = cv2.resize(image,(80,80))
    image = np.stack((image,)*3, axis=-1)
    prediction = labels.inverse_transform([np.argmax(model.predict(image[np.newaxis,:]))])
    return prediction
wpod_net_path = "wpod-net.json"
wpod_net = load_model(wpod_net_path)
# Load model architecture, weight and labels
json_file = open('MobileNets_character_recognition.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
model = model_from_json(loaded_model_json)
model.load_weights("License_character_recognition_weight.h5")
print("[INFO] Model loaded successfully...")

labels = LabelEncoder()
labels.classes_ = np.load('license_character_classes.npy')
print("[INFO] Labels loaded successfully...")
def call():
    global f
    global LpImg
    global dff
    def import_csv_data():
        f = askopenfilename(
        parent=root,
        title='Choose file')
        v.set(f)



    def Detect_plate():

        f=v.get()

        vehicle, LpImg,cor = get_plate(f)
        # Create a figure of specific size
        #figure = Figure(figsize=(3, 3), dpi=100)

        # Define the points for plotting the figure
        #plot = figure.add_subplot(2, 3, 1)
        #plt.imshow(vehicle)
        #plot = figure.add_subplot(2, 3, 2)
        #plt.imshow(LpImg[0])

        # Add a canvas widget to associate the figure with canvas
        ##canvas = FigureCanvasTkAgg(figure, root)
        ##canvas.get_tk_widget().pack(ipadx= 10 ,ipady = 12,padx=11, pady=11)
        fig = plt.figure(figsize=(12,6))
        grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
        fig.add_subplot(grid[0])
        plt.axis(False)
        plt.imshow(vehicle)
        grid = gridspec.GridSpec(ncols=2,nrows=1,figure=fig)
        fig.add_subplot(grid[1])
        plt.axis(False)
        plt.imshow(LpImg[0])

    def preprocess():
        f=v.get()
        vehicle, LpImg,cor = get_plate(f)
```

```
    if (len(LpImg)): #check if there is at least one license image
        # Scales, calculates absolute values, and converts the result to 8-bit.
        plate_image = cv2.convertScaleAbs(LpImg[0], alpha=(255.0))

        # convert to grayscale and blur the image
        gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray,(7,7),0)

        # Applied inversed thresh_binary
        binary = cv2.threshold(blur, 180, 255,
                               cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]

        kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
        thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)


    # visualize results
    fig = plt.figure(figsize=(12,7))
    plt.rcParams.update({"font.size":18})
    grid = gridspec.GridSpec(ncols=2,nrows=3,figure = fig)
    plot_image = [plate_image, gray, blur, binary,thre_mor]
    plot_name = ["plate_image","gray","filter","binary","Enhancement"]

    for i in range(len(plot_image)):
        fig.add_subplot(grid[i])
        plt.axis(False)
        plt.title(plot_name[i])
        if i ==0:
            plt.imshow(plot_image[i])
        else:
            plt.imshow(plot_image[i],cmap="gray")

def character_reg():
    f=v.get()
    vehicle, LpImg,cor = get_plate(f)
    if (len(LpImg)): #check if there is at least one license image
        # Scales, calculates absolute values, and converts the result to 8-bit.
        plate_image = cv2.convertScaleAbs(LpImg[0], alpha=(255.0))

        # convert to grayscale and blur the image
        gray = cv2.cvtColor(plate_image, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray,(7,7),0)

        # Applied inversed thresh_binary
        binary = cv2.threshold(blur, 180, 255,
                               cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
        kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
        thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
        cont, _ = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

        # creat a copy version "test_roi" of plat_image to draw bounding box
        test_roi = plate_image.copy()

        # Initialize a list which will be used to append charater image
        crop_characters = []

        # define standard width and height of character
        digit_w, digit_h = 30, 60

        for c in sort_contours(cont):
            (x, y, w, h) = cv2.boundingRect(c)
            ratio = h/w
            if 1<=ratio<=3.5: # Only select contour with defined ratio
                if h/plate_image.shape[0]>=0.5: # Select contour which has the height larger than 50% of the plate
                    # Draw bounding box arroung digit number
                    cv2.rectangle(test_roi, (x, y), (x + w, y + h), (0, 255,0), 2)

                    # Sperate number and gibe prediction
                    curr_num = thre_mor[y:y+h,x:x+w]
                    curr_num = cv2.resize(curr_num, dsize=(digit_w, digit_h))
                    _, curr_num = cv2.threshold(curr_num, 220, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
                    crop_characters.append(curr_num)

        print("Detect {} letters...".format(len(crop_characters)))
        fig = plt.figure(figsize=(10,6))
        plt.axis(False)
        plt.imshow(test_roi)
        #plt.savefig('grab_digit_contour.png',dpi=300)
        fig = plt.figure(figsize=(14,4))
        grid = gridspec.GridSpec(ncols=len(crop_characters),nrows=1,figure=fig)

        for i in range(len(crop_characters)):
            fig.add_subplot(grid[i])
            plt.axis(False)
            plt.imshow(crop_characters[i],cmap="gray")
        #plt.savefig("segmented_leter.png",dpi=300)
        fig = plt.figure(figsize=(15,3))
        cols = len(crop_characters)
        grid = gridspec.GridSpec(ncols=cols,nrows=1,figure=fig)
        df=[]
        final_string = ''
        for i,character in enumerate(crop_characters):
            fig.add_subplot(grid[i])
            title = np.array2string(predict_from_model(character,model,labels))
            plt.title('{}'.format(title.strip("'[]"),fontsize=20))
            final_string+=title.strip("'[]")
            plt.axis(False)
            plt.imshow(character,cmap='gray')
```

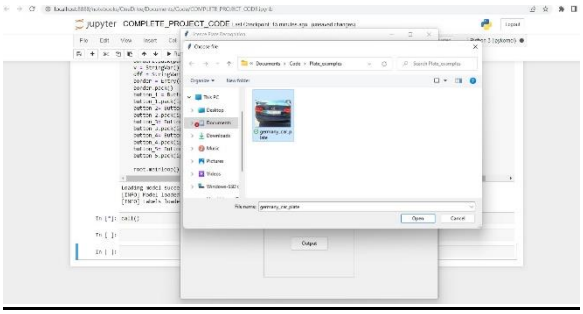
```
        print(final_string)
        df.append(final_string)
        dff.set(final_string)

    def detectowner():
        ds = dff.get()
        data =pd.read_csv("db.csv",header=None)
        print(data[1].where(data[0] ==ds ))
        print(ds)
        values = data[1].where(data[0] ==ds )
        #messagebox.showinfo("showinfo",
data[1].where(data[0] ==ds ))


    root = Tk()
    root.geometry('650x650+450+100')
    root.title('Licence Plate Recognition')
    pane = PanedWindow(root,orient=HORIZONTAL)
    pane.pack(fill=BOTH, expand=1)
    frame = Frame(root, relief=GROOVE , borderwidth=5)
    frame.pack(fill=BOTH, expand=True)
    row1 = tk.Label(pane, text="Licence Plate Recognition",
bg="white",fg="black", cursor="hand2")
    pane.add(row1)
    labelfont = ('times', 15, 'bold')
    row1.config(font=labelfont)
    checkGroup = LabelFrame(frame, text = "Menu", padx=50,
pady=55)
    checkGroup.pack(padx=70, pady=50)
    borderL = tk.Label(checkGroup, text="File")
    borderL.pack(padx=3, pady=3)
    v = StringVar()
    dff = StringVar()
    border = Entry(checkGroup, textvariable=v)
    border.pack()
    button_1 = Button(checkGroup,text='input
image',command=import_csv_data, cursor="hand2")
    button_1.pack(ipadx= 10 ,ipady = 11,padx=11 ,pady=11)
    button_2= Button(checkGroup,text='Detect plate',
command=Detect_plate, cursor="hand2")
    button_2.pack(ipadx= 10 ,ipady = 11,padx=11, pady=11)
    button_3= Button(checkGroup,text='Preprocessing',
command=preprocess, cursor="hand2")
    button_3.pack(ipadx= 10 ,ipady = 11,padx=11, pady=11)
    button_4= Button(checkGroup,text='Detect letters',
command=character_reg, cursor="hand2")
    button_4.pack(ipadx= 10 ,ipady = 11,padx=11, pady=11)
    button_5= Button(checkGroup,text='Output',
command=detectowner, cursor="hand2")
    button_5.pack(ipadx= 10 ,ipady = 11,padx=11, pady=11)

    root.mainloop()
```

6. SCREENSHOTS



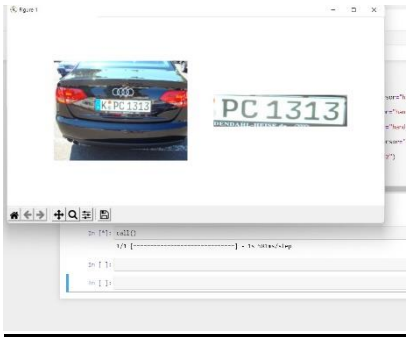
OUTPUT PAGE

Run-through of code and the opening page is excecuted in a new tab.

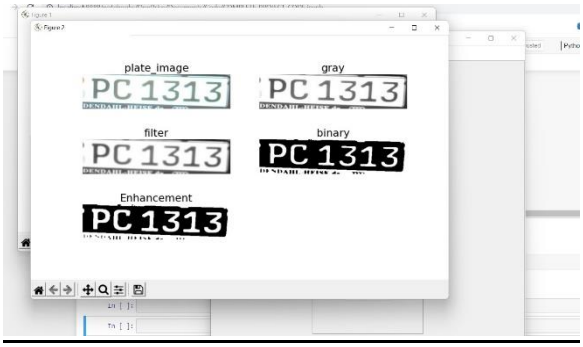


DATASET STORED

Choosing Input Image of liscence plate from “INPUT IMAGE” module

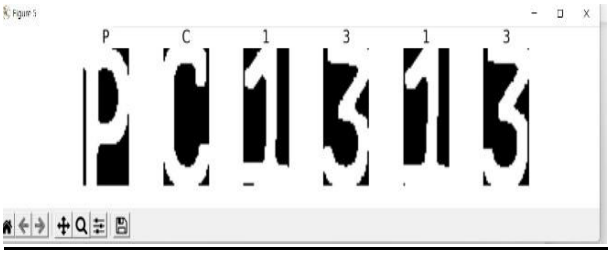


CROPING OF IMAGE



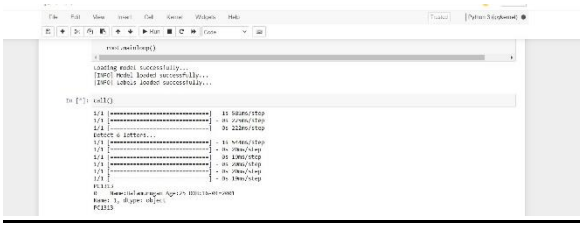
CONVERTING IMAGE TO TEXT

The Detected Image ie. Number Plate undergoes preprocessing and the text from the plate is extracted and displayed below.



CONVERTED TEXT

The Extracted Image of elements in the plate is detected individually and displayed separately when we use "DETECT LETTERS".



END RESULT DETAILS

The detected elements are run through the dataset and the details of the owner of vehicle will be mentioned below.

7. RESULTS AND DISCUSSION

All smart cities commonly face a key problem related to parking facilities and traffic management systems. Projected work does not simply reduce the traffic jam, but it is collectively giving user authentication, cost-efficient and real-time. This method is enforced by putting geolocation, real-time cloud server, a microprocessor for instant data assortment and no-parking detection mobile-application into service. There will be no wastage time anymore searching for a parking space because the parking space will find them.

8. CONCLUSIONS

Smart parking systems typically get information only regarding accessible parking areas. All smart cities commonly face a key problem related to parking facilities and traffic management systems. Due to vehicles in a no-parking zone, several folks ought to face traffic problems. Therefore, to reduce this, projected work combines IoT and cloud. The target of the projected work is to create a model, "Detection and Identification of Vehicle's No-Parking area using IoT and cloud", that helps the drivers to acknowledge the present parked space. No parking area is detected by making use of geolocation of vehicles with the assistance of a real-time cloud server.

ACKNOWLEDGEMENT

We would also like to extend our sincere gratitude and grateful thanks to our principal **Dr. E R VIJAYA KRISHNA** for having extended the research and development facilities of the department.

We are grateful to our chairman **Shri. M K RAJAGOPALAN**. He has been a constant source of inspiration right from the beginning.

We would like to express our faithful and grateful thanks to our administrator officer, **Shri. V BHASKARAN** for his support

We also sincerely thank our head of the department, **Mr. R VINOTH KUMAR** whose continuous encouragement and sufficient comments enabled us to complete our project report.

We thank our project coordinator, **Mr M Bhuvaneswari** and **all our staff members** who have been by our side always and helped us with our project. We also sincerely thank **all the lab technicians** for their help as in the course of our project development.

we are very thankful and grateful to our beloved guide,**MR. Bhuvaneswari, MRS. M INDUMATHY** whose great support in valuable advices, suggestions and tremendous help enabled us in completing our project. they has been a great source of inspiration to us.

We wish to thank our **family members and friends** for their constant encouragement, constructive criticisms and suggestion that has helped us in timely completion of this project.

REFERENCES

[1] Hong Yamin. The two generation License Plate Segmentation and Recognition photo identification [J]. Journal of Lanzhou industry college, 2021,06:18-22.

[2] Zhao Xingwang, Li Tianyang, Wang Liang, Zhou Jing. The two generation License Plate Segmentation and Recognition number recognition system of [J]. computer and modernization based on digital equipment, 2021,06:132-136.

[3] Fu Ronghui. The research and design of vehicle license plate recognition ystem in traffic management system [J]. International Journal of Signal Processing, Image Processing and Pattern Recognition, v 9, n 3, p 445-456, 2020.

[4] Yingyon Zou, Jian Zhai, Yongde Zhang, Xinyan Cao, Guangbin Yu, Juhui, Chen. Research on algorithm for automatic license plate recognition system[J]. International Journal of Multimedia and Ubiquitous Engineering, v 10, n 1, p 101-108, 2019.

[5] Tao Hong, Gopalakrishnam A.K. License plate extraction and recognition of a Thai vehicle based on MSER and BPNN[J]. Proceedings of the 2015 7th International Conference on Knowledge and Smart Technology (KST), p 48-53, 2019.

[6] Dong Jingwei,Sun, Meiting,Liang Gengrui,Jin Kui. The improved neural network algorithm of license plate recognition[J]. International Journal of Signal Processing, Image Processing and Pattern Recognition, v 8, n 5, p 49-54, 2019.

[7] Qu Zhong, Chang Qing-li, Chen Chang-zhi, Lin Li-dan. An improved character recognition algorithm for license plate based on BP neural network[J].Open Electrical and Electronic Engineering Journal, v 8, n 1, p 202-207, 2020.