Vacant Parking Space Detection

Ansh Mittal (17ucs028)
Anshul Jain (17ucs029)
Harshit Garg (17ucs064)
Anshu Musaddi (17ucs185)

Abstract

In this paper, we propose a solution for effective use of the main parking space of our campus. Using the piecewise transformation algorithm, we are segmenting area available and the cars present on a given image. Also, the perspective of the image was changed in order to apply the algorithm on it.

Keywords

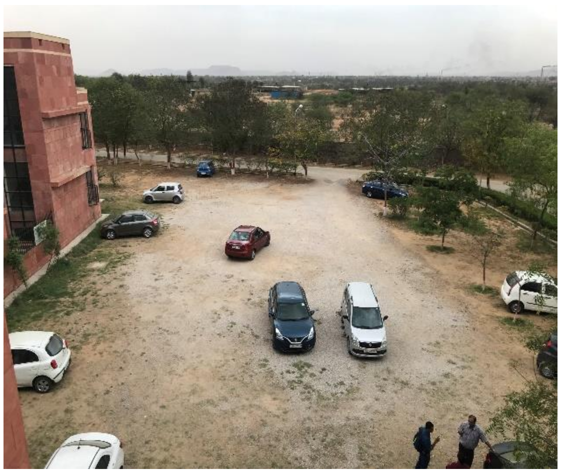
Perspective transformation, range filtering, thresholding, morphological transformation.

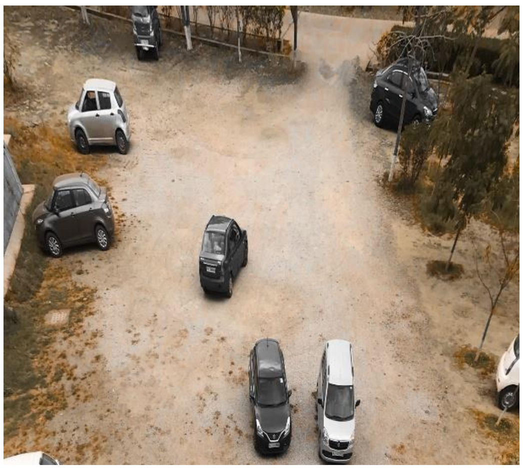
Introduction

The main parking space near the mechanical department in our campus, faces a problem of effective use of the parking area. We, therefore, intend to provide a solution by creating an image processing algorithm which will utilize photographs in order to predict the number of vacant space available in the parking area. We used piecewise transformation algorithm to calculate the total available area for parking. With this, we are able to take out the number of vacant spaces available.

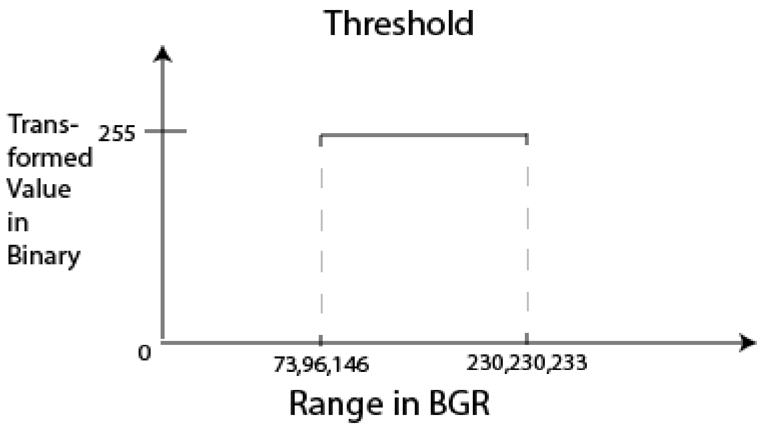
Approach

To start the project, we captured the pictures of the site from above by fixing the camera at the roof in one position. The images did not have the desired perspective and size so we altered the perspective by perspective transformation and resized the image to desired one.

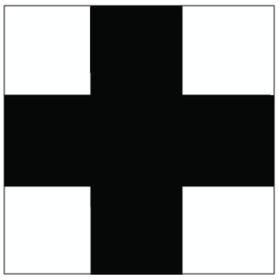




After that, we performed thresholding on the image in order to obtain a binary image. The thresholding was performed using a piecewise linear transformation where the pixels that were between the range (73,96,146) and (230,230,233) were transformed to 255 while all the other pixels were transformed to 0.



The thresholding changed the image from BGR to binary. The obtained binary image helped us in differentiating between the area which can be used for parking and car parked along with the area which can't be used for parking. After performing threshold, we observed some salt and pepper noise in the obtained image. This noise was handled by first performing opening followed by closing with same structing element. The structuring element that was used for performing the opening and closing is shown below –



We chose the above structure keeping in mind that the curves of car are to be maintained as if we increase the edges, the number of cars that can be accommodated increases.

After performing the above steps, we clearly got the empty region denoted by 255 and all the occupied region with 0. We asked the user to input the length of the car he wants. As the actual length differs from that of pixels in the image he can simply click on the car edges already present in the image. Post that we created a window of the specified dimensions. We calculated length*breadth *255(ws) for the whole image. We kept on putting the window at each pixel of the image and calculated the sum of the intensity values in that window(os). We calculated the error that is there in the image and if it is greater than 8%, we ignored the error and made that region complete black and marked that region with green in the original image. Also, with help of counter we counted the number of cars.

Image After Thresholding and Noise Reduction

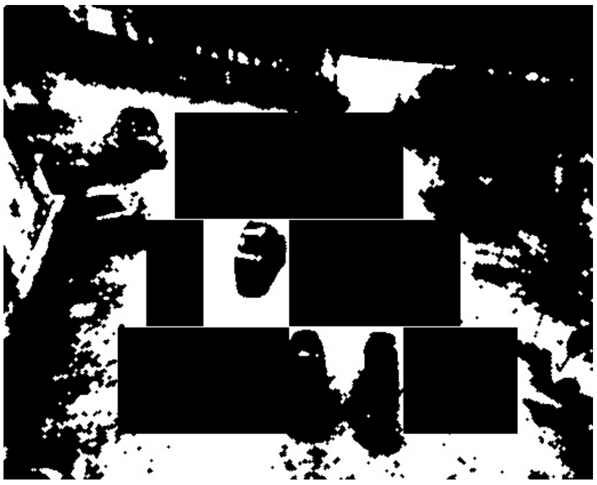
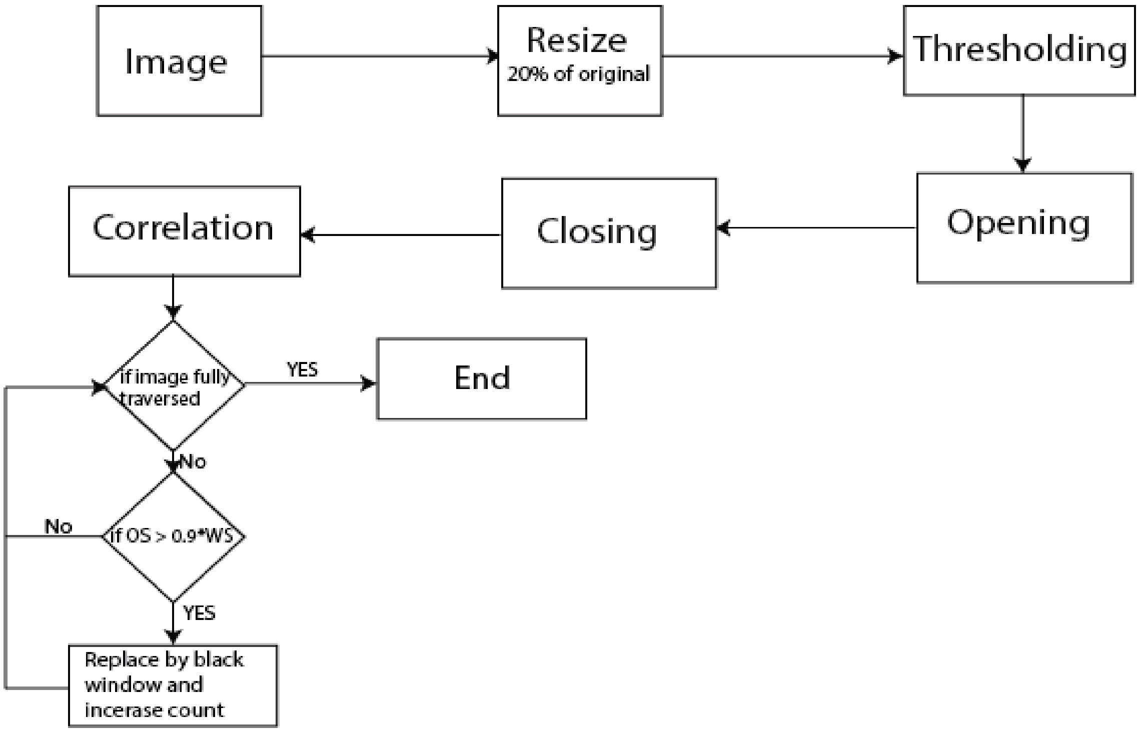


Image After Counting Empty Spots

Worked Performed

- Resizing
- Perspective transformation
- Thresholding
- Opening
- Closing
- Correlation



CODE

Perspective Transformation

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

clicks = []
#load image and resize it
image = cv2.imread('Pictures/IMG_3980.jpg')
image = cv2.resize(image,(1023,1023))

#set perspective parameters which have border points
pts1 = np.float32([[260,330],[738,370],[0,808],[1003,808]])
pts2 = np.float32([[0,0],[200,0],[0,200],[200,200]])

#Perform perspective transform
M = cv2.getPerspectiveTransform(pts1,pts2)
dst = cv2.warpPerspective(image,M,(200,200))

#plotting both the images in a window
plt.subplot(121),plt.imshow(image),plt.title('Input')
plt.subplot(122),plt.imshow(dst),plt.title('Output')
plt.show()
```

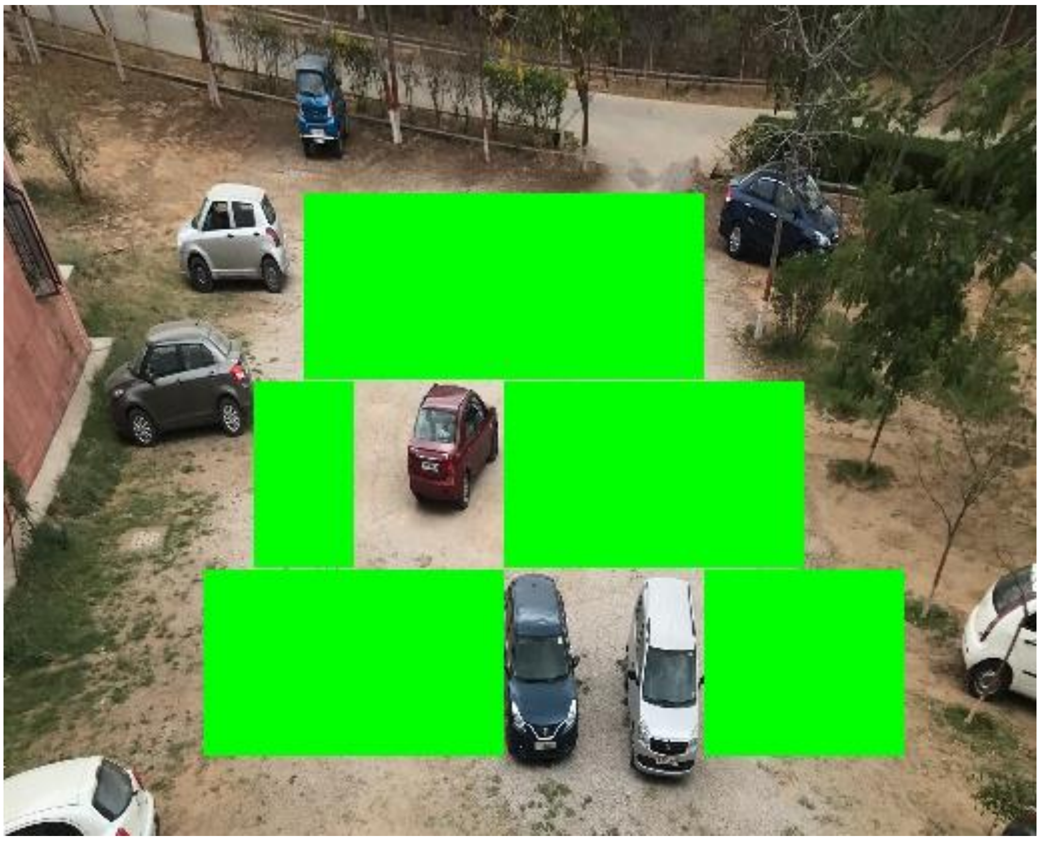
Parking System

```
import cv2
import numpy as np
import os
import sys
import time
import matplotlib.pyplot as plt

markerLst = []
#System function to accept mouseclick position
def on_mouse(event, x, y, flags, params):
    if event == cv2.EVENT_LBUTTONDOWN:
        print(str(x) + ', ' + str(y)+ "  (" +str(img[y,x,0])+" , 
"+str(img[y,x,1])+" , "+str(img[y,x,2])+" )")
        markerLst.append((x,y))

#Applied colour filtering for identifying space and separate out cars
def filter(img):
    outimg = np.zeros((img.shape[0],img.shape[1]),np.uint8)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i][j][0] >= 73 and img[i][j][0] <= 230:
                if img[i][j][1] >= 96 and img[i][j][1] <= 230:
                    if img[i][j][2] >= 146 and img[i][j][2] <= 233:
                        outimg[i][j] = 255
                    else:
                        outimg[i][j] = 0
                else:
                    outimg[i][j] = 0
            else:
                outimg[i][j] = 0

    return outimg
```

```
#User defined window size is stored
def windowSize():
    while len(markerLst)!=4:
        time.sleep(1)
    print(markerLst)
    wnWid = abs(markerLst[0][0]-markerLst[1][0])
    wnLen = abs(markerLst[2][1]-markerLst[3][1])
    return (wnWid,wnLen)

#calculates no of car parking spaces available
def carCalculator(img,colorimg,windowWidth,windowLength):
    count = 0
#    windowLength = 90
#    windowWidth = 60
#    windowLength = windowLength - int(0.08*windowLength)
#    windowWidth = windowWidth - int(0.08*windowWidth)
    window = np.full((windowLength,windowWidth),255,np.uint8)
    userImg = colorimg.copy()
    ws = windowWidth*windowLength*255
    newImage = img.copy()
    for i in range(0,img.shape[0]-windowLength+1,int(windowLength/2)+1):
        for j in range(100,img.shape[1]-
windowWidth+1,int(windowWidth/2)+1):
            os = np.sum(newImage[i:i+windowLength,j:j+windowWidth])
            #if Intensity is more than minimum allowed than it is parking
space
            #increase the count and replace that section by green box
            if os/ws > 0.92:
                count = count+1
                for n in range(i,i+windowLength+1):
                    for m in range(j,j+windowWidth+1):
                        newImage[n][m] = 0
                        userImg[n][m] = [0,255,0]
            cv2.imshow('output',userImg)
    return count

#Image opened and resized
img = cv2.imread('Pictures/Parking(post warp).jpg')
resizedImg = cv2.resize(img,None,fx=0.2, fy=0.2, interpolation =
cv2.INTER_CUBIC)

#Converting coloured image into black and white image using thresholding
and showing it
threshImg = filter(resizedImg)

#Circle shaped structuring element is used for applying opening and closing
and then it is showed
kernel = np.array([[0,1,0],[1,1,1],[0,1,0]],np.uint8)
opnImg = cv2.morphologyEx(threshImg, cv2.MORPH_OPEN, kernel)
clsImg = cv2.morphologyEx(opnImg, cv2.MORPH_CLOSE, kernel)

#User will mark first structuring element's width folowed by its length on
original image
print("\n============================================================
======")
print('****Mark on input image for maximum dimensions(width * length) of
car****')
cv2.namedWindow('Input')
cv2.setMouseCallback('Input', on_mouse, 0 )
cv2.imshow('Input',resizedImg)
cv2.imshow('after opening & closing',clsImg)
```

```
print('Press key to continue:')
cv2.waitKey()
tuples = windowSize()
wWid = tuples[0]
wLen = tuples[1]
print(str(wWid)+'\t'+str(wLen))
count = carCalculator(clsImg,resizedImg,wWid,wLen)
print(str(count)+" Cars can be parked.")
print("===========================================================================
====\n")
cv2.waitKey()
cv2.destroyAllWindows()
```

Results



By performing the above techniques, we reached to a very good model which gave the desired results. There are few issues that can be worked upon more if get some more time that are-

- We couldn’t incorporate the space occupied by the grass and trees it can be incorporated if we add some more colour range in the thresholding.
- We couldn’t incorporate the orientation of the window that we used for calculating the number of available spaces for the car parking.

Discussion

We could have got a better result if we have taken images from a better angle.
We could have also resolved the issues mentioned in the results.

Bibliography

- Stack overflow
- Python3 documentation
- OpenCV documentation

Contribution

Ansh Mittal (17ucs028) – 25% - Perspective Transformation and documentation
Anshul Jain (17ucs029) - 25% - Thresholding and presentation
Harshit Garg (17ucs064) – 25% - Opening and Closing and documentation
Anshu Musaddi (17ucs185) – 25% - Correlation and presentation