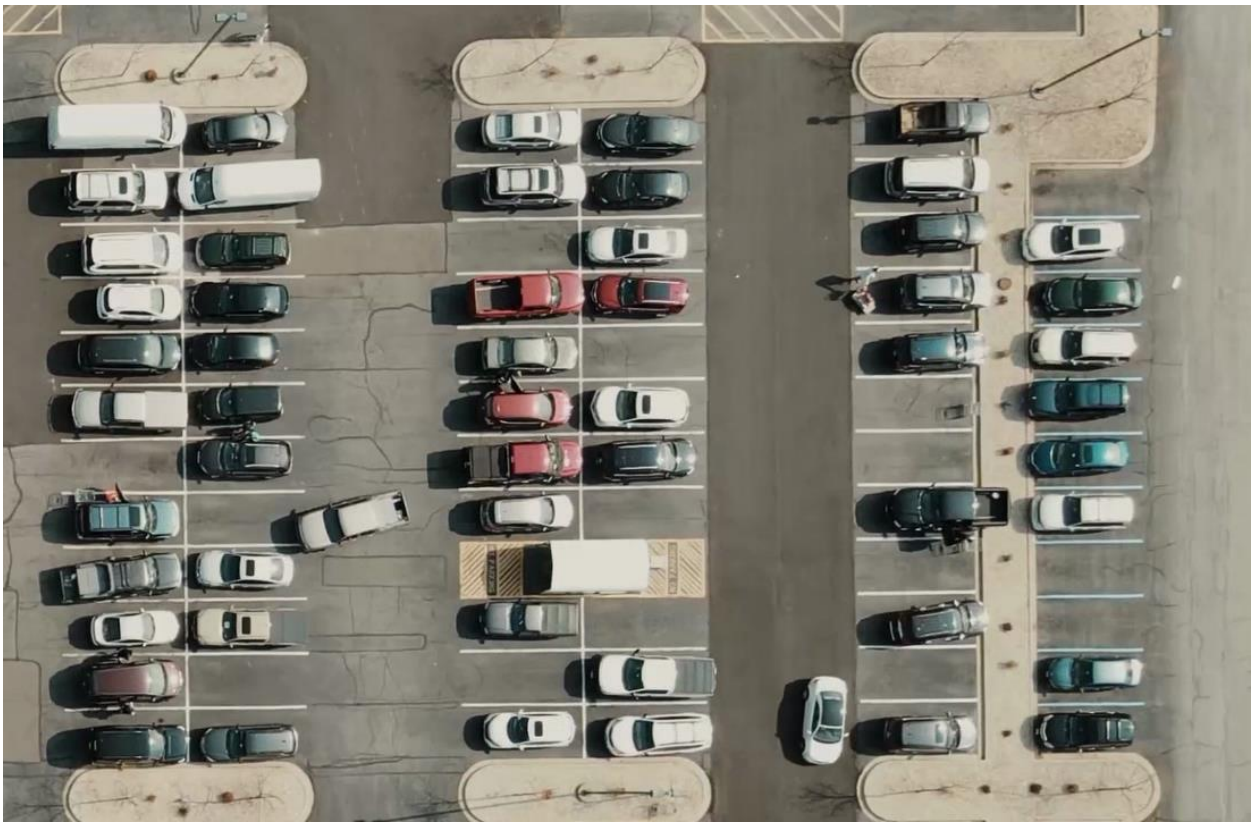
DATE	**17/06/2023**
TEAM ID	**NM2023TMID06916**
CODE	**PYTHON**

AI ENABLED CAR PARKING SYSTEM

An AI-enabled car parking system is a solution that leverages artificial intelligence to automate and optimize the process of parking vehicles in a parking lot or garage. The system uses sensors, cameras, and other technologies to monitor and analyze the available parking spaces, detect the presence of vehicles, and guide drivers to vacant spots.

The problem that this system aims to solve is the inefficiency and frustration associated with traditional parking systems, which often result in wasted time, fuel, and resources. Conventional parking lots and garages typically rely on manual processes and human attendants to manage the flow of vehicles, which can be slow, error-prone, and prone to congestion.

An AI-enabled car parking system can address these issues by providing real-time information on available parking spots, optimizing the use of space, and minimizing the time and effort required for parking. This can improve the overall customer experience, reduce traffic congestion and emissions, and increase revenue for parking lot operators.

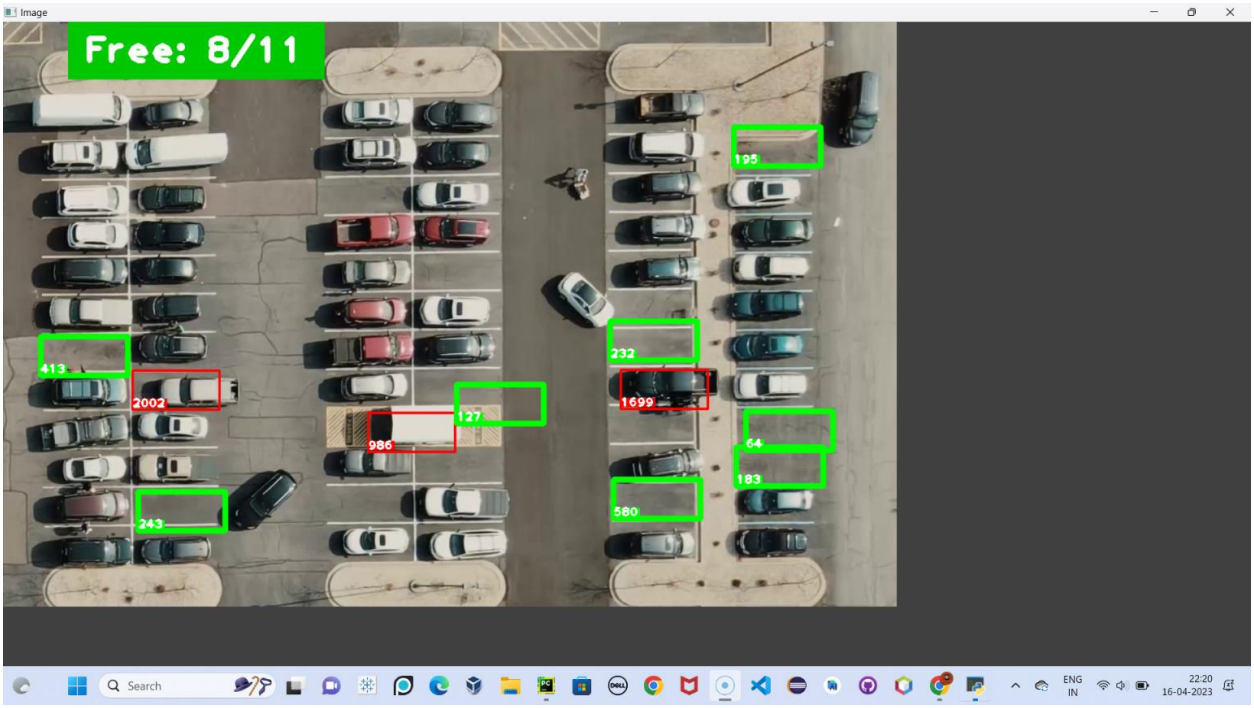


MAIN.PY

```
import cv2
import pickle


width, height = 107, 48


try:
    with open('CarParkPos', 'rb') as f:
        posList = pickle.load(f)
except:
```

```
    posList = []


def mouseClick(events, x, y, flags, params):
    if events == cv2.EVENT_LBUTTONDOWN:
        posList.append((x, y))
    if events == cv2.EVENT_RBUTTONDOWN:
        for i, pos in enumerate(posList):
            x1, y1 = pos
            if x1 < x < x1 + width and y1 < y < y1 + height:
                posList.pop(i)

    with open('CarParkPos', 'wb') as f:
        pickle.dump(posList, f)


while True:
    img = cv2.imread('carParkImg.png')
    for pos in posList:
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), (255, 0, 255), 2)

    cv2.imshow("Image", img)
    cv2.setMouseCallback("Image", mouseClick)
    cv2.waitKey(1)
```

OUTPUT FOR MAIN.PY



Main_Trackbars.py

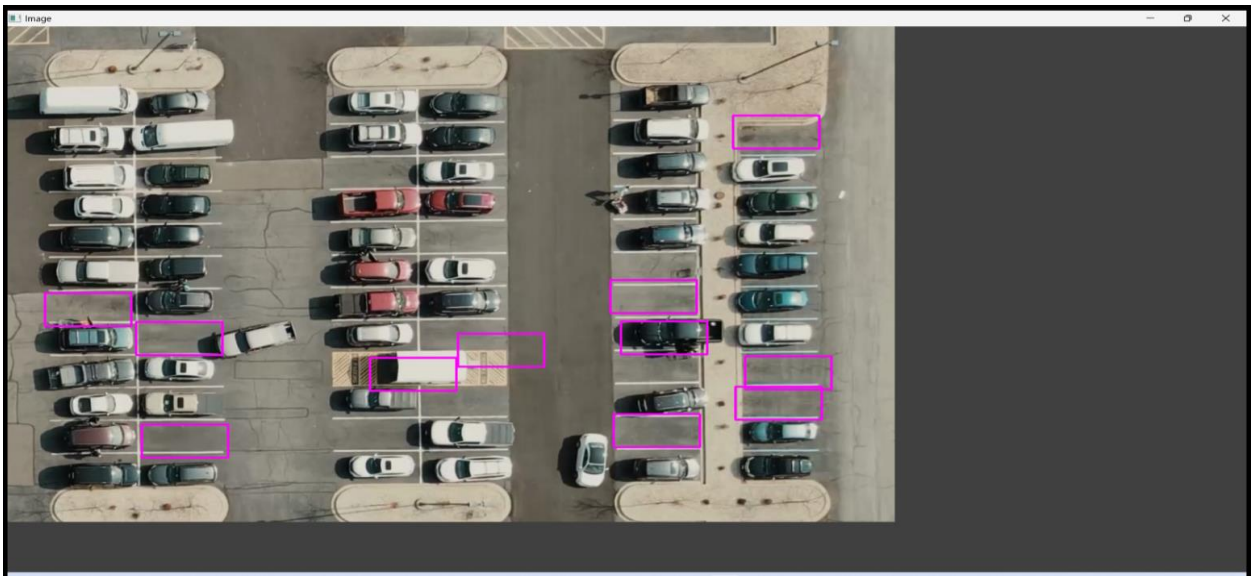
```
import cv2
import pickle
import cvzone
import numpy as np

cap = cv2.VideoCapture('carPark.mp4')
width, height = 103, 43
with open('polygons', 'rb') as f:
    posList = pickle.load(f)



def empty(a):
    pass
cv2.namedWindow("Vals")
cv2.resizeWindow("Vals", 640, 240)
cv2.createTrackbar("Val1", "Vals", 25, 50, empty)
cv2.createTrackbar("Val2", "Vals", 16, 50, empty)
cv2.createTrackbar("Val3", "Vals", 5, 50, empty)



def checkSpaces():
```

```
    spaces = 0
    for pos in posList:
        x, y = pos
        w, h = width, height

        imgCrop = imgThres[y:y + h, x:x + w]
        count = cv2.countNonZero(imgCrop)

        if count < 900:
            color = (0, 200, 0)
            thic = 5
            spaces += 1

        else:
            color = (0, 0, 200)
            thic = 2

        cv2.rectangle(img, (x, y), (x + w, y + h), color, thic)

        cv2.putText(img, str(cv2.countNonZero(imgCrop)), (x, y + h - 6),
cv2.FONT_HERSHEY_PLAIN, 1,
                    color, 2)

    cvzone.putTextRect(img, f'Free: {spaces}/{len(posList)}', (50, 60), thickness=3,
offset=20,
```

```
                colorR=(0, 200, 0))



while True:


    # Get image frame
    success, img = cap.read()
    if cap.get(cv2.CAP_PROP_POS_FRAMES) ==
cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    # img = cv2.imread('img.png')
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    # ret, imgThres = cv2.threshold(imgBlur, 150, 255, cv2.THRESH_BINARY)


    val1 = cv2.getTrackbarPos("Val1", "Vals")
    val2 = cv2.getTrackbarPos("Val2", "Vals")
    val3 = cv2.getTrackbarPos("Val3", "Vals")
    if val1 % 2 == 0: val1 += 1
    if val3 % 2 == 0: val3 += 1
    imgThres = cv2.adaptiveThreshold(imgBlur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                     cv2.THRESH_BINARY_INV, val1, val2)
    imgThres = cv2.medianBlur(imgThres, val3)
    kernel = np.ones((3, 3), np.uint8)
```

```
imgThres = cv2.dilate(imgThres, kernel, iterations=1)

checkSpaces()
# Display Output

cv2.imshow("Image", img)
# cv2.imshow("ImageGray", imgThres)
# cv2.imshow("ImageBlur", imgBlur)
key = cv2.waitKey(1)
if key == ord('r'):
    pass
```

Output For Main_Trackbars.py



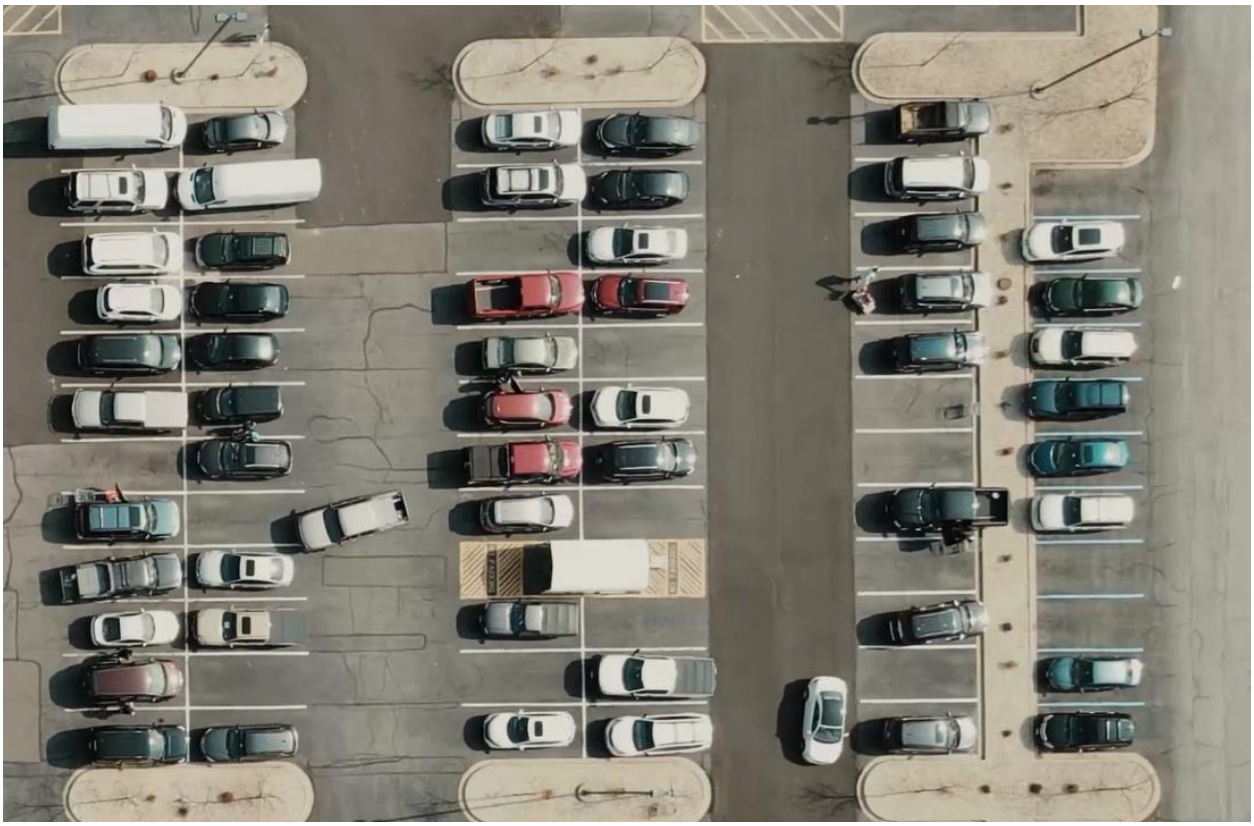
ParkingSpacePicker.py

```
import cv2
import pickle
import cvzone
import numpy as np

# Video feed
cap = cv2.VideoCapture('carPark.mp4')

with open('CarParkPos', 'rb') as f:
    posList = pickle.load(f)

width, height = 107, 48
def checkParkingSpace(imgPro):
    spaceCounter = 0
    for pos in posList:
        x, y = pos
        imgCrop = imgPro[y:y + height, x:x + width]
        # cv2.imshow(str(x * y), imgCrop)
        count = cv2.countNonZero(imgCrop)
        if count < 900:
            color = (0, 255, 0)
```

```
            thickness = 5
            spaceCounter += 1
        else:
            color = (0, 0, 255)
            thickness = 2
        cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
        cvzone.putTextRect(img, str(count), (x, y + height - 3), scale=1,
                           thickness=2, offset=0, colorR=color)

    cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50),
scale=3,
                       thickness=5, offset=20, colorR=(0,200,0))
while True:

    if cap.get(cv2.CAP_PROP_POS_FRAMES) ==
cap.get(cv2.CAP_PROP_FRAME_COUNT):
        cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
    success, img = cap.read()
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
    imgThreshold = cv2.adaptiveThreshold(imgBlur, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                         cv2.THRESH_BINARY_INV, 25, 16)
    imgMedian = cv2.medianBlur(imgThreshold, 5)
    kernel = np.ones((3, 3), np.uint8)
```

```
imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)


checkParkingSpace(imgDilate)
cv2.imshow("Image", img)
# cv2.imshow("ImageBlur", imgBlur)
# cv2.imshow("ImageThres", imgMedian)
cv2.waitKey(10)
```

OUTPUT FOR CARPARKLMG.PNY



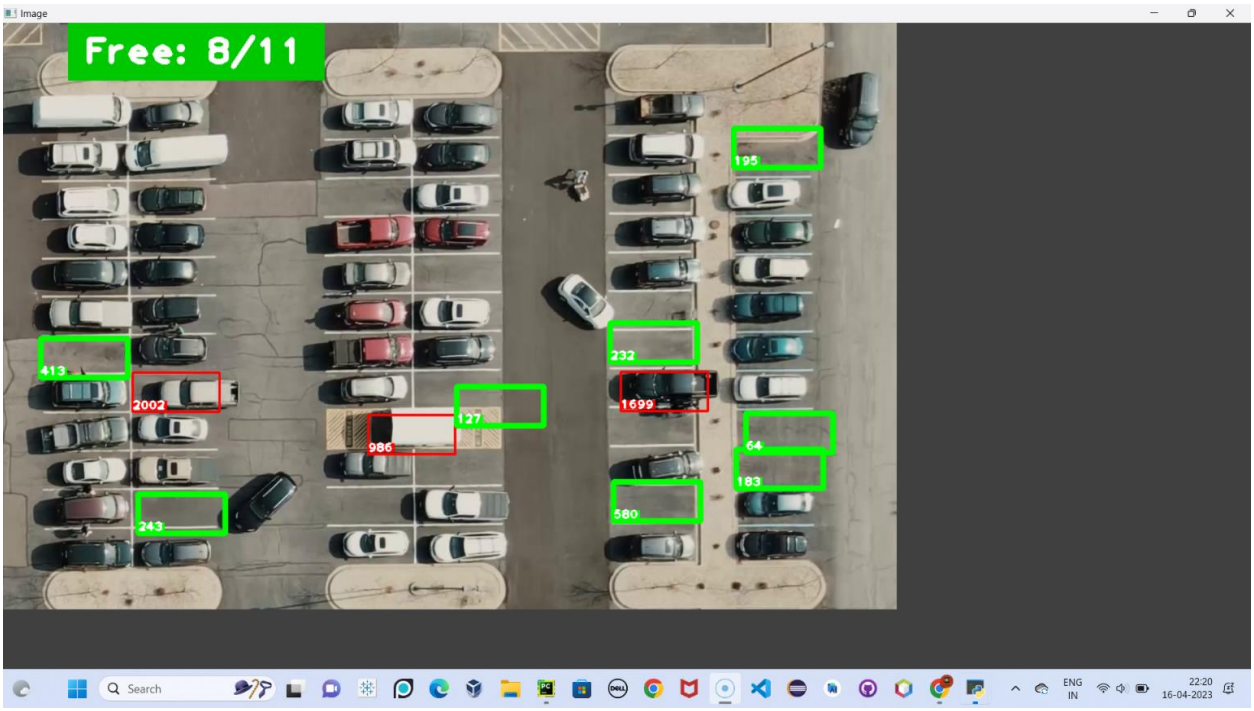


Image
Free: 8/11
195
413
232
2002
1699
127
986
64
183
580
243
Search
ENG
IN
22:20
16-04-2023