1-Data Türü ve Yapısı:

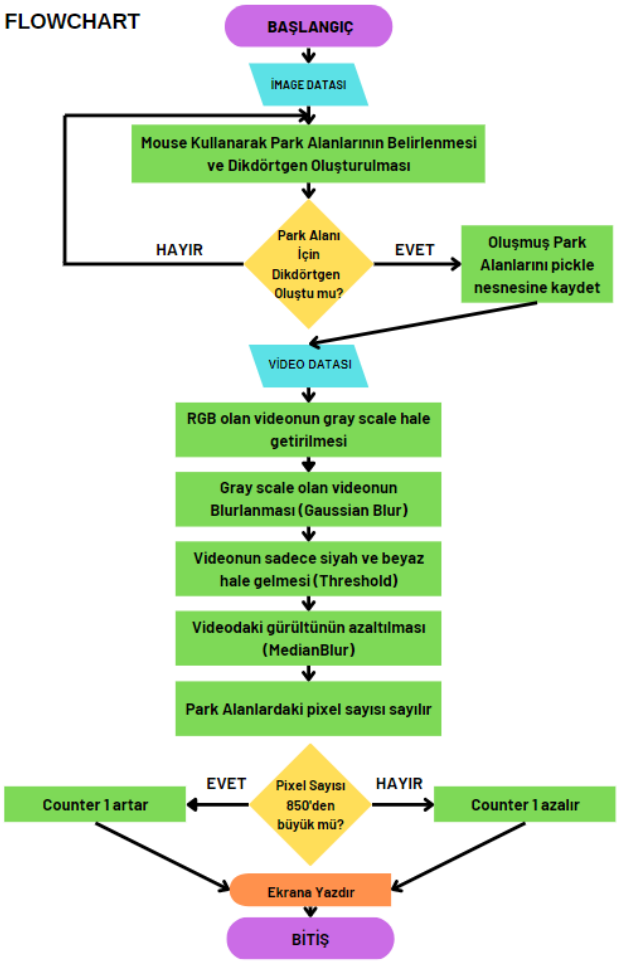
Projede çalışılan data bir videodur ve uzantısı **.mp4** uzantılıdır. Figure1.1'de OpenCV kullanarak videonun özelliklerini gözlenmiştir.

```
##########################################################################################################################
#1- Data türü ve yapısı:
def get_video_properties(video_path):
    # Videoyu aç
    video = cv2.VideoCapture(video_path)
    # Video özelliklerini al
    frame_count = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    fps = video.get(cv2.CAP_PROP_FPS)
    duration_sec = frame_count / fps
    # Video çözünürlüğünü al
    width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))

    # Videoyu kapat
    video.release()

    return frame_count, fps, duration_sec , width , height

video_path = 'carPark.mp4'
frame_count, fps, duration_sec ,width, height = get_video_properties(video_path)
# Sonuçlarıları yazdıralım:
print("***Çerçeve Sayısı:", frame_count , "***FPS:", fps , "***Süre (saniye):", duration_sec)
print("Çözünürlük: {}x{}".format(width, height))
##########################################################################################################################
```

```
C:\Users\doggu\PycharmProjects\CarSpaceOpenCv\venv\Scripts\python.exe C:/Users/doggu/PycharmProjects/CarSpaceOpenCv/anaS
***Çerçeve Sayısı: 679 ***FPS: 24.0 ***Süre (saniye): 28.291666666666668
Çözünürlük: 1100x720
```

(Figure1.1)

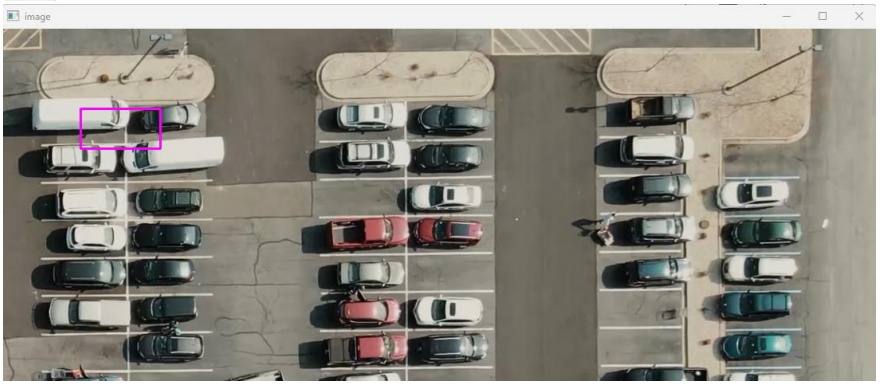
2- Algoritmanın FlowChart Tasarımı:



3- Kodlama ve Çıktıları:

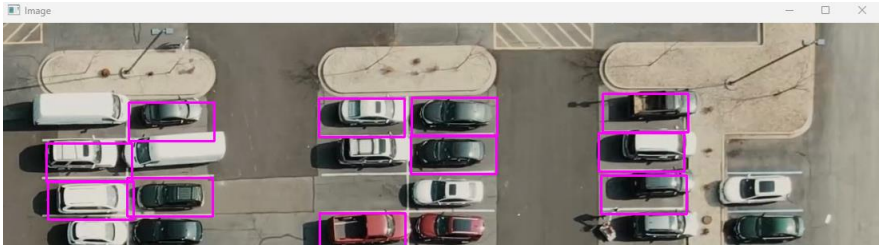
I. Resim Datasının Okunması ve Resim Üzerinde Dikdörtgen Çizilmesi:

```python
import cv2
import pickle

img = cv2.imread("carParkImg.png")

while True:
    #cv2.rectangle(image, start_point, end_point, color, thickness)
    cv2.rectangle(img , (100,100) , (200,150), (255,0,255) , 2)
    cv2.imshow("image" , img)
    cv2.waitKey(1)
```



II. Mouse Click ile Dikdörtgen Çizimi ve Dikdörtgenin Kenar Uzunluğunun Belirlenmesi:

```python
width , height = 107,48   #dikdörtgenin uzun ve kısa kenarını belirledik
posList = []

def mouseClick(events ,x ,y ,flags , params):
    if events ==cv2.EVENT_LBUTTONDOWN:
        posList.append((x, y))

while True:

    for pos in posList:

        #cv2.rectangle(image, start_point, end_point, color, thickness)
        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)
```



III. Pickle Kütüphanesi ile Dikdörtgenleri Tutan Nesne Oluşturulması:

```python
width , height = 107,48   #dikdörtgenin uzun ve kısa kenarını belirledik

try :
    with open('CarParkPos', 'rb') as f:    #önceki dikdörtgenleri tutar
        posList = pickle.load(f)
except:
    posList = []

def mouseClick(events ,x ,y ,flags , params):
    if events ==cv2.EVENT_LBUTTONDOWN:   # sol click ile kare çizer
        posList.append((x, y))
    if events ==cv2.EVENT_RBUTTONDOWN:     #sağ click ile kareyi siler
        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:  #picke nesnesine ekler dikdörtgeni
        pickle.dump(posList , f)
```

IV. Parking Space Algılanması:

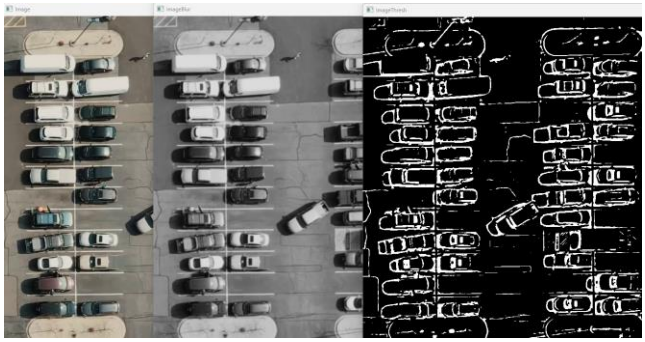
```
#video
cap = cv2.VideoCapture('carPark.mp4')
width , height = 107,48   #dikdörtgenin uzun ve kısa
def checkParkingSpace():
    for pos in posList:
        x , y =pos

        imgCrop = img[y:y+height,x:x+width]
        cv2.imshow(str(x+y) , imgCrop)

with open('CarParkPos', 'rb') as f:  # önceki dikdörtgenleri tutar
    posList = pickle.load(f)
```

V. RGB olan videonun GrayScale ve sonrasında GaussianBlur olarak ayarlanması:

```
imgGray = cv2.cvtColor(img , cv2.COLOR_BGR2GRAY)
imgBlur = cv2.GaussianBlur(imgGray , (3,3) , 1)
```

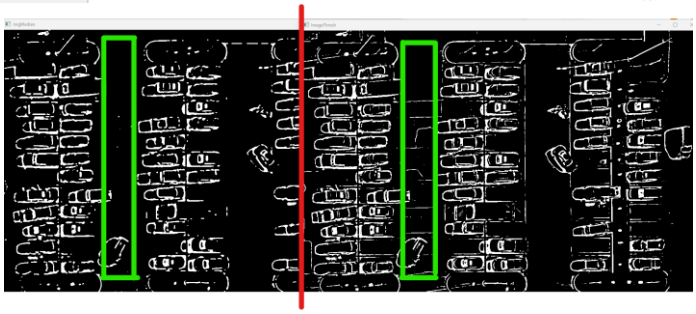
VI. Threshold ile videonun sadece siyah ve beyaz renkleri(0 veya 255) olarak ayarlanması:

```
#Binary image yapma
imgThreshold = cv2.adaptiveThreshold(imgBlur ,255 , cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                     cv2.THRESH_BINARY_INV,25,16)
```



VII. MedianBlur ile Noise Temizlendi. Daha doğru sonuç verir:

```
imgMedian = cv2.medianBlur(imgThreshold,5)
```



VIII. Her dikdörtgen içindeki pixel sayısının yazdırılması:

```
#oluşturulan dikdörtgen içinde ne kadar pixel dolu , çok ise arac vardır.
count = cv2.countNonZero(imgCrop)
cvzone.putTextRect(img , str(count) , (x,y+height-10) , scale= 1 ,thickness=3 , offset=0)
```

IX. Boş Park alanlarının ve dolu park alanlarının gösterilmesi:

```
19          cvzone.putTextRect(img , str(count) , (x,y+height-3) , scale= 1.5
20                             ,thickness=2 , offset=0, colorR=(0,0,250))
21
22
23          if count < 850:
24              color= (0,255,0) #red
25              thickness=5
26          else:
27              color = (0,0,255) #green
28              thickness = 5
29          cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
30
```

X. Ekranda Sonuç Gösterilmesi:

```
23
24          if count < 850:
25              color= (0,255,0) #red
26              thickness=5
27              spaceCounter +=1
28          else:
29              color = (0,0,255) #green
30              thickness = 5
31          cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
32
33      #skorları yazdırmak için
34      cvzone.putTextRect(img, f'Free :{spaceCounter}/{len(posList)}', (100, 50), scale=3
35                         , thickness=5, offset=20, colorR=(0, 200, 0))
```

4- Algoritma değerlendirmesi: (grafikleri, blok diyagramları, karşılaştırma vs.)

Algoritmada herhangi bir grafik gerektirecek uygulamaya gereksinim duyulmamıştır. Algoritma çalışma mantığında video kısmına geçmeden önce dikdörtgenlerin (park alanlarının oluşması) için video üzerinden ekran görüntüsü alınıp işlemler yapılmıştır. Algoritmanın düzenli çalışması ve dinamik olması için pickle nesnesi kullanılmıştır. Algoritma sonucu dört farklı çıktı oluşmuştur. Threshold kullanılarak pixel sayısı sayılmıştır. Pixel sayısına göre park alanı boşluk durumu döndüren bir algoritma yazılmıştır.

5- Yorumlama

Algoritma sonucu 4 farklı sonuç çıkmıştır. Aşağıdaki gibidir:

1. RGB video: Algoritmanın çalışması için uygun değildir ve bu yüzden üzerinde farklı bir işlem yapılmıştır.
2. GrayScaleVideo: Gray scale video amacı RGB olan resmi 0-255 arasında siyah beyaz tonlaması yapılır , pixel sayarken daha kolay olma
3. Threshold Siyah Beyaz video: Threshold ile gray scale olan video 0 veya 255 değerleri almıştır. Algoritma için bu şekilde olması gerekmektedir.
4. MedianBlur Video : Amaç algoritmanın pixel sayarken noise olan kısımları saymaması ve daha doğru sonuç vermesidir.

Son çıktı olarak ise 69 park alanında boş olanlar çıkarılmıştır ve ekrana yazdırılmıştır. OpenCV kütüphaneleri kullanılarak gerçek zamanlı park alanı projesi yapılmıştır.

Kaynaklar:

https://docs.opencv.org/4.x/
https://www.computervision.zone/courses/parking-space-counter/
https://docs.python.org/3/library/pickle.html
https://en.wikipedia.org/wiki/Gaussian_blur