Oceanic Trash

Round 3

ROV ROBO-TECH

MATE ROV COMPETITION

Microsoft Azure

Contents

Abstract

In this challenge we focused mainly on choosing good datasets, training well and choosing a well prediction threshold for the trash in the video.

Phase 1: Data collection

Choosing the dataset wasn't that hard as there were many datasets there and as there were various objects in the video, we labeled our data on fish and trash.

We collected our data from:

- Trash-ICRA19: A Bounding Box Labeled Dataset of Underwater Trash

Phase 2: Preparing the data

Fig.1 The flow of data from the drive to custom vision service

The same as the previous challenges we pulled the data using google drive API and transferred it to the custom vision service then we trained it but this time we consumed so much quota and used many accounts to train the data.

the collected data was in YOLO Format so we made a code that assigns the regions to the objects as you can see in Fig.2

```
#/content/testinh/
def Get_Regions( mianDir ,filename):
  if os.path.exists( mianDir + filename + ".txt" ):
    TextFile = open( mianDir + filename + ".txt", "r").read()
    out_regions = []
    Regions = TextFile.splitlines()
    for imgRegion in Regions :
      tagis = imgRegion[0]

      x,y,w,h = imgRegion.split(" ")[1:]
      x,y,w,h = float(x),float(y),float(w),float(h)
      x,y = x-(w/2) , y-(h/2)
      if (tagis == '0') :

        oneRegion =  Region(tag_id=Trash_tag.id, left=x,top=y,width=w,height=h)
      elif (tagis == '1'):

          oneRegion =  Region(tag_id=bio_tag.id, left=x,top=y,width=w,height=h)
      elif tagis == '2' :

           oneRegion =  Region(tag_id=rov_tag.id, left=x,top=y,width=w,height=h)

      out_regions.append(oneRegion)
    return out_regions
```

Fig.2 assigning the regions to the objects

```
import os
import sys
directory = "/content/drive/MyDrive/Marwan_Trash-ICRA19/trash_ICRA19/dataset/train/" #drive path
tagged_images_with_regions = []
patch_count = 0
total_count = 0
for filename in os.listdir(directory):
  #logging the stuates

  loggingOut = str(total_count) + "/" + "5720" + "    " + str((total_count/5212 )*100)[:3] + "%"
  sys.stdout.write("\r" + loggingOut)
  sys.stdout.flush()

  if filename.endswith(".jpg") :
    total_count += 1
    patch_count += 1
    regions = Get_Regions(directory, filename[:-4] )
    with open(directory  + filename , mode="rb") as image_contents:
        tagged_images_with_regions.append(
            ImageFileCreateEntry(name=filename[:-4],
                                 contents=image_contents.read(), regions=regions))
  if patch_count == 63 : # maximum number of images in one upload is 64
      patch_count = 0
      try:
        Upload_Data(tagged_images_with_regions)
      except:
        print("An exception occurred while uploading")

      tagged_images_with_regions = []

```

Fig.3 Code used to upload to custom vision service

Phase 3: Extracting the trash from the video

The Flow of the Phase

1. As shown in Fig.4 we started by getting the video URL to fetch it.
2. Then cutting it to 30 frames per second, we didn't pass every frame to our custom vision service as it consumed so much quota and time.
3. The output will be the bounding box around the detected trash object, we will loop through every prediction to check its probability if it is high and if it is really a trash object. as shown in Fig.5.

Fig.4

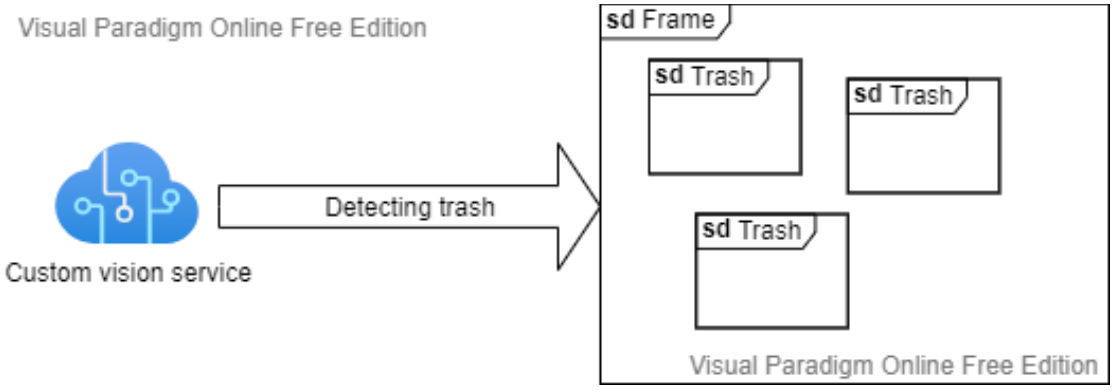

Fig.5

Some Real time Images:

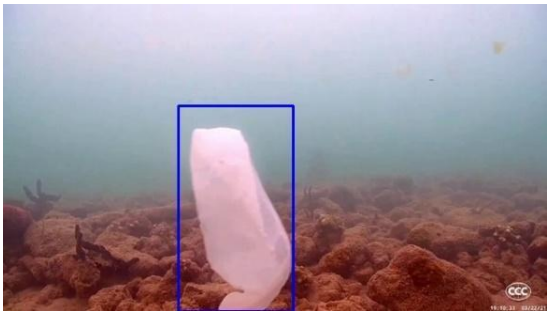

Fig.6

Fig.7

Fig.8

Fig.9

CODE:

```
cap = cv2.VideoCapture("../video.mp4")  #Importing the video
count=0
# TRACKER INITIALIZATION
success, frame = cap.read()

while(cap.isOpened()):
    ret, img = cap.read()
    if ret == False :
      break
    cv2.imwrite("buffer" +str(count)+ ".jpg" ,img )  #Storing input frame
    print("read",count)
    data = countImg(count)  #Pass the frame to be processed

    count +=1
```

Fig.10

```python
#Function to process each input frame
def countImg(count):
  font = cv2.FONT_HERSHEY_SIMPLEX
  with open("buffer" +str(count)+ ".jpg", mode="rb") as test_data:   #Importing input frame
    results = predictor.detect_image_with_no_store(project.id, "Iteration1",test_data)  #Sending it to the model
  image = cv2.imread("buffer" +str(count)+ ".jpg")   #Reading the input frame
  fish=0
  for i in results.predictions:
    if i.probability >pred and i.tag_name in tags:
      #Calculating bounding box size
      box = i.bounding_box
      h,w,_ = image.shape
      start_point =(  math.floor( box.left * w)  , math.floor(box.top * h) )
      end_point =     math.floor( (box.left * w )+ box.width* w) , math.floor( (box.top * h) + box.height* h )
      color = (255, 0, 0)
      thickness = 2
      #Drawing bounding box
      image = cv2.rectangle(image,start_point,end_point,color, thickness)
  cv2.imwrite("../images/" +str(count)+ ".jpg" ,image )   #Exporting output frame
```

Fig.11

Phase 4: Uploading the video

After processing the frames in the custom vision service, we stacked them again to recreate our output video as shown in Fig.12.

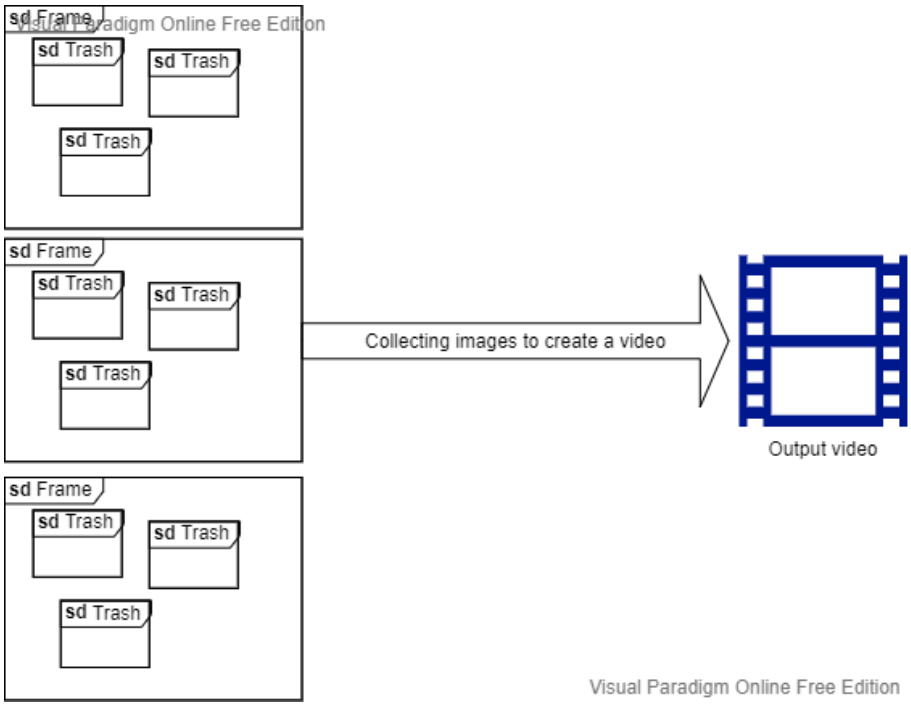

Fig.12

CODE:

```
#Exporting the images to create the final video
img_array = []
scale_percent = 60 # percent of original size
for i in range(count):
    filename=f'images/{i}.jpg'
    img = cv2.imread(filename)
    width = int(img.shape[1] * scale_percent / 100)
    height = int(img.shape[0] * scale_percent / 100)
    dim = (width, height)
    resized = cv2.resize(img, dim, interpolation = cv2.INTER_AREA)
    img_array.append(resized)

out = cv2.VideoWriter('project2.mp4',cv2.VideoWriter_fourcc(*'DIVX'), 30, dim)

for i in range(len(img_array)):
    out.write(img_array[i])
out.release()
```

Fig.13

Lessons learnt

Even though the task wasn't that hard we faced more troubles than the previous challenges as the subscription quota was draining quickly and we had to use less resources to complete our task.

Flowchart of the program