2021 - AI 825 - Visual Recognition
International Institute of Information Technology, Bangalore

Assignment 2

Shourabh Payal (MT2020054)

1. **Compute the fore-ground segmentation mask for the attached images: Image1, Image2 & Image3**

   We will be using a system with 2 stages (This system **doesn't** use user interaction)

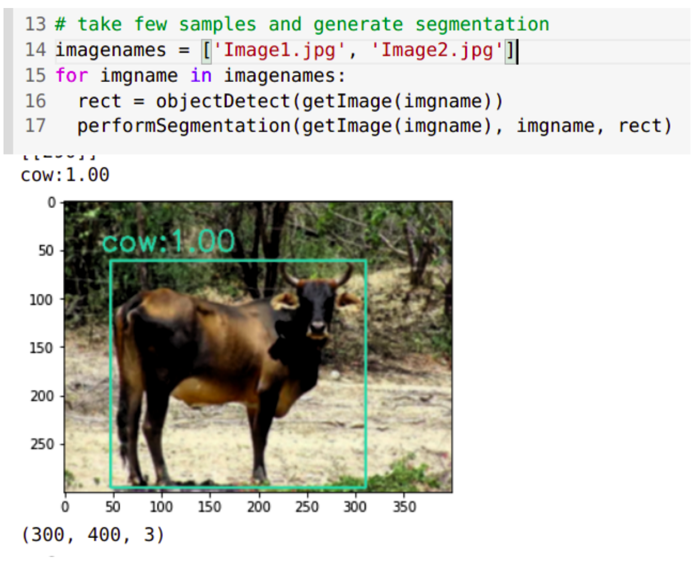
   (a) **Stage 1: Detect an object (YOLO bounding box)**

   In this stage we will make use of YOLO detector to identify objects in our images. Since we don not have any data at hand for training the model from scratch, we will be using pre trained weights for the model which was trained on COCO dataset.

   The YOLO detector will also provide us with a bounding box around our object. The classification class of the object should not necessarily be correct as long as the bounding box provides a good measure of the objects in the image.

   This bounding box can be fed to our next stage to extract foreground. In addition to normal detection of the bounding boxes using YOLO we also perform non max suppression to get rid of redundant boxes for the same object.

   Below is one of the intermediate results produced by this stage.



```
# take few samples and generate segmentation
imagenames = ['Image1.jpg', 'Image2.jpg']
for imgname in imagenames:
  rect = objectDetect(getImage(imgname))
  performSegmentation(getImage(imgname), imgname, rect)
```

```
cow:1.00
```

```
(300, 400, 3)
```

(b) **Stage 2: GrabCut algorithm**

Input to our second stage is an image and a bounding box that was detected by the stage 1 yolo detector.

We will use the grabcut algorithm provided by opencv with cv.GC_INIT_WITH_RECT option and keeping the interactive mask to all zeros signifying no user interaction.

We also perform some additional tweaks like applying gaussian blur to the image before feeding it to stage 2. We apply dfs fill algorithm (performDfsCorrection) which makes an bounded black region inside a white region white. Below is one such example on cow image.

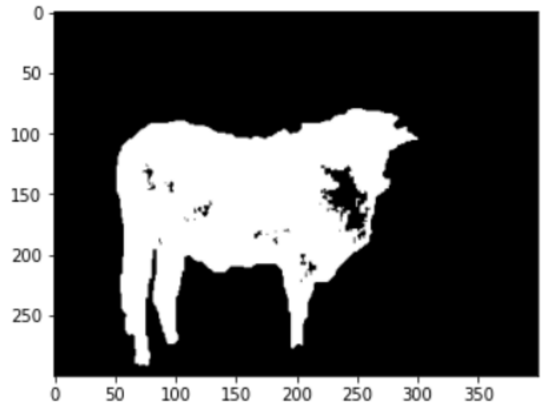


Figure 1: Before dfs flood fill correction

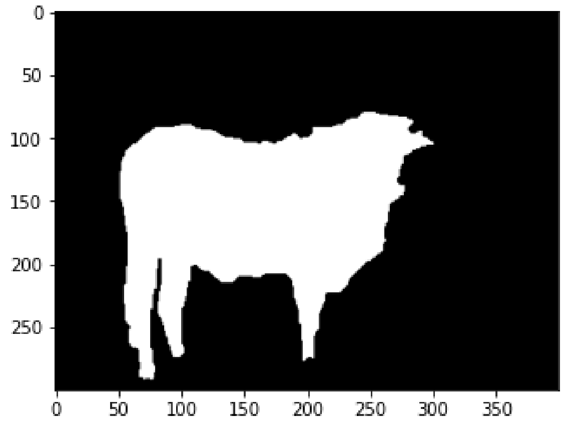
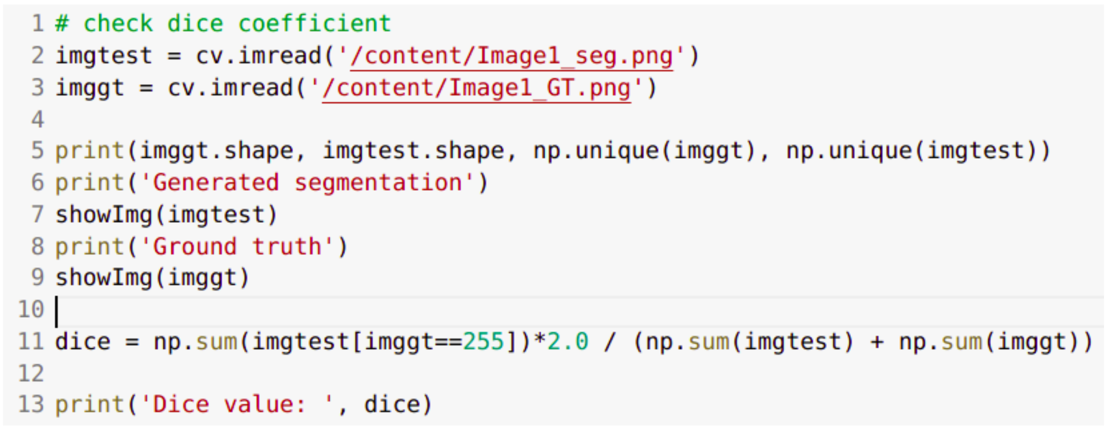


Figure 2: After dfs flood fill correction

(c) **Results**

We were able to achieve a dice score of 0.959 against the cow ground truth image provided.



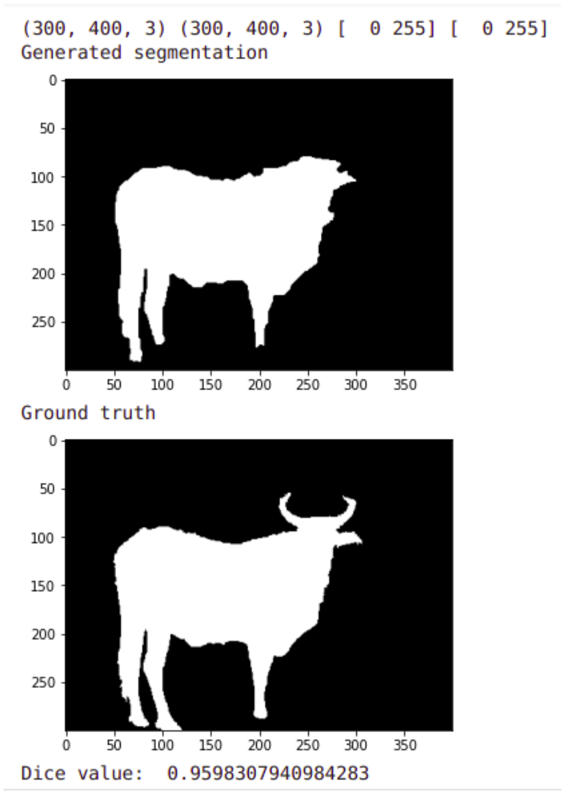
```python
# check dice coefficient
imgtest = cv.imread('/content/Image1_seg.png')
imggt = cv.imread('/content/Image1_GT.png')

print(imggt.shape, imgtest.shape, np.unique(imggt), np.unique(imgtest))
print('Generated segmentation')
showImg(imgtest)
print('Ground truth')
showImg(imggt)

dice = np.sum(imgtest[imggt==255])*2.0 / (np.sum(imgtest) + np.sum(imggt))

print('Dice value: ', dice)
```

Figure 3: Calculate dice score



```
(300, 400, 3) (300, 400, 3) [  0 255] [  0 255]
Generated segmentation
```

```
Ground truth
```

```
Dice value:  0.9598307940984283
```

Figure 4: Results for ground truth cow image

Here are some other results