PixelLib

Release 0.1.0

May 27, 2020

Contents:

PixelLib is a library created for performing image segmentation using few lines of code. It is a flexible library created to allow easy integration of image segmentation into software solutions.

PixelLib requires python's version 3.5-3.7, Download python

It requires pip's version >= 19.0

Install pip with:

```
pip3 install pip
```

Install PixelLib and its dependencies:

Install the latest version of tensorflow(Tensorflow 2.0+) with:

```
pip3 install tensorflow
```

Install Opencv-python with:

```
pip3 install opencv-python
```

Install Pillow with:

```
pip3 install pillow
```

Install scikit-image with:

```
pip3 install scikit-image
```

Install PixelLib with:

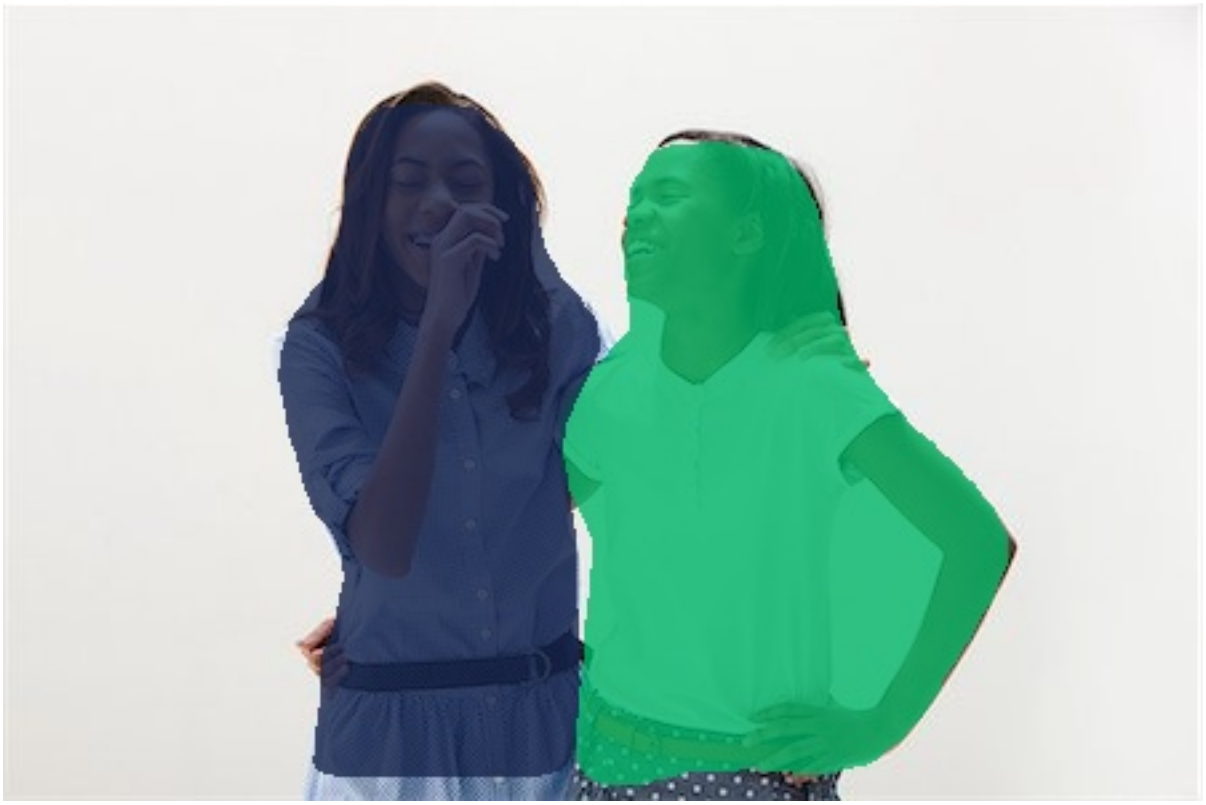
```
pip3 install pixellib
```

PixelLib supports the two major types of segmentation:

1 **Semantic segmentation**: Objects in an image with the same pixel values are segmented with the same colormaps.

SEMANTIC SEGMENTATION WITH PIXELLIB

2 **Instance segmentation**: Instances of the same object are segmented with different color maps.



INSTANCE SEGMENTATION WITH PIXELLIB

CHAPTER 1

SEMANTIC SEGMENTATION WITH PIXELLIB

PixelLib is implemented with Deeplabv3+ framework to perform semantic segmentation. Xception model trained on pascalvoc dataset is used for semantic segmentation.

Download the xception model from here.

Code to implement semantic segmentation:

```
import pixellib
from pixellib.semantic import semantic_segmentation

segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.h5")
segment_image.segmentAsPascalvoc("path_to_image", output_image_name = "path_to_output_
↪image")
```

We shall take a look into each line of code.

```
import pixellib
from pixellib.semantic import semantic_segmentation

#created an instance of semantic segmentation class
segment_image = semantic_segmentation()
```

The class for performing semantic segmentation is imported from pixellib and we created an instance of the class.

```
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.h5")
```

We called the function to load the xception model trained on pascal voc.

```
segment_image.segmentAsPascalvoc("path_to_image", output_image_name = "path_to_output_
↪image")
```

This is the line of code that performs segmentation on an image and the segmentation is done in the pascalvoc's color format. This function takes in two parameters:

path_to_image: the path to the image to be segemented.

path_to_output_image: the path to save the output image. The image will be saved in your current working directory.

Sample1.jpg

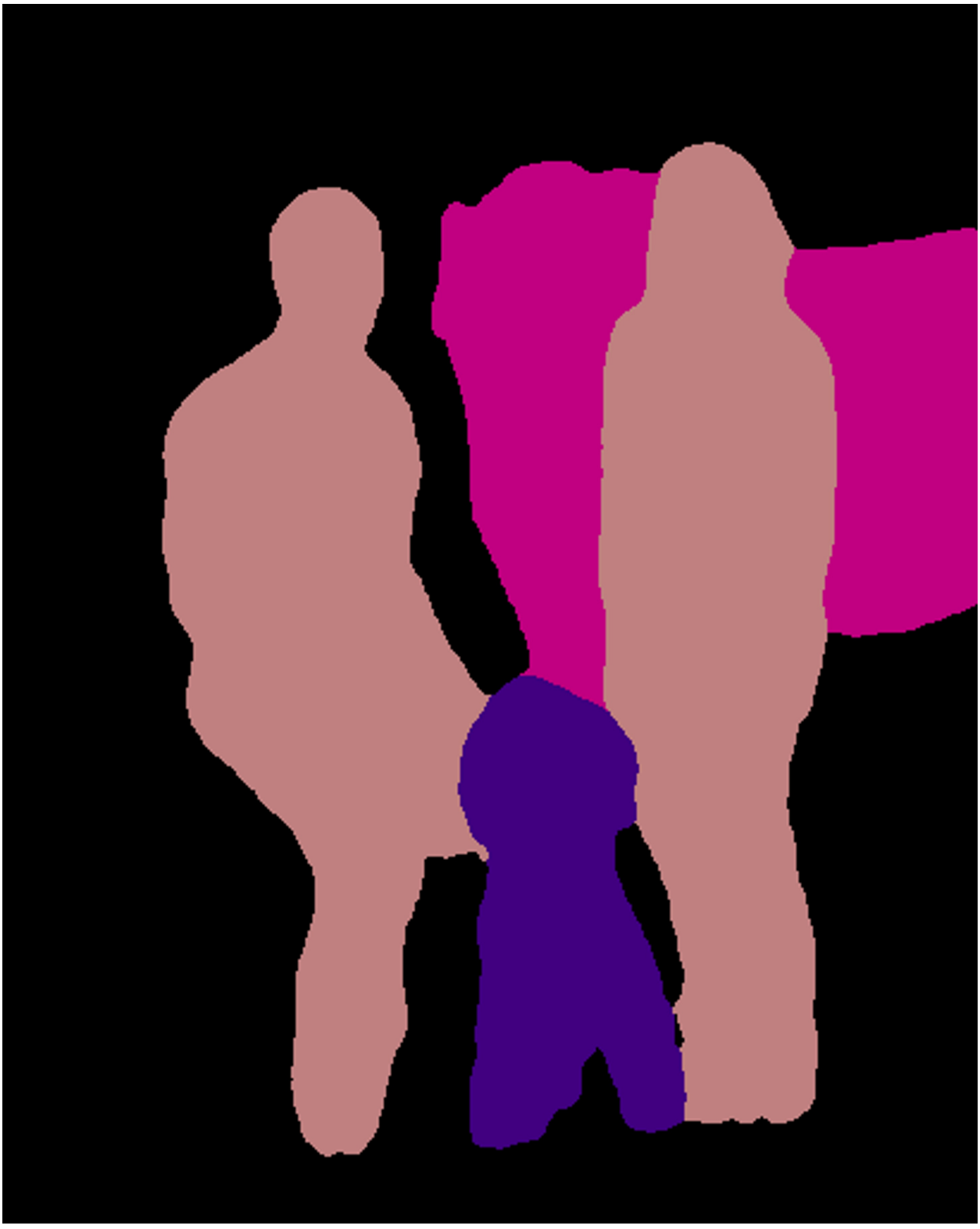
Image's source: Pinterest

```
import pixellib
from pixellib.semantic import semantic_segmentation
```

(continues on next page)

(continued from previous page)

```
segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("deeplabv3_xception_tf_dim_ordering_tf_kernels.h5")
segment_image.segmentAsPascalvoc("sample1.jpg", output_image_name = "image_new.jpg")
```



Your saved image with all the objects present segmented.

You can obtain an image with segmentation overlay on the objects with a modified code below.

```
segment_image.segmentAsPascalvoc("sample1.jpg", output_image_name = "image_new.jpg",
↪overlay = True)
```

We added an extra parameter **overlay** and set it to **true**, we produced an image with segmentation overlay.

- You can check the inference time required for performing segmentation by modifying the code below..

```
import pixellib
from pixellib.semantic import semantic_segmentation
import time

segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("pascal.h5")

start = time.time()
segment_image.segmentAsPascalvoc("sample1.jpg", output_image_name= "image_new.jpg")

end = time.time()
print(f"Inference Time: {end-start:.2f}seconds")
```

```
Inference Time: 8.19seconds
```

It took 8.19 seconds to run semantic segmentation on the image.

Specialised uses of PixelLib may require you to return the array of the segmentation's output.

- Obtain the array of the segmentation's output by using this code,

```
output, segmap = segment_image.segmentAsPascalvoc()
```

- You can test the code for obtaining arrays and print out the shape of the output by modifying the semantic segmentation code below.

```
import pixellib
from pixellib.semantic import semantic_segmentation
import cv2

segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("pascal.h5")
output, segmap = segment_image.segmentAsPascalvoc("sample1.jpg")
cv2.imwrite("img.jpg", output)
print(output.shape)
```

- Obtain both the output and the segmentation overlay's arrays by using this code,

```
output, segoverlay = segment_image.segmentAsPascalvoc(overlay = True)
```

```
import pixellib
from pixellib.semantic import semantic_segmentation
import cv2

segment_image = semantic_segmentation()
segment_image.load_pascalvoc_model("pascal.h5")
segmap, segoverlay = segment_image.segmentAsPascalvoc("sample1.jpg", overlay= True)
cv2.imwrite("img.jpg", segoverlay)
print(segoverlay.shape)
```

This xception model is trained on pascal voc dataset, a dataset with 20 object categories.

Objects and their corresponding colormaps.


Aeroplane
Bicycle
Bird
Boat
Bottle
Bus
Car
Dog
Chair
Cow
Diningtable
Cat
Horse
Motorbike
Person
Pottedplant
Sheep
Sofa
Train
Tvmonitor

CHAPTER 2

INSTANCE SEGMENTATION WITH PIXELLIB

Instance segmentation with PixelLib is based on MaskRCNN framework.

Download the mask rcnn model from here

Code to implement instance segmentation:

```
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("path_to_image", output_image_name = "output_image_path")
```

Observing each line of code:

```
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
```

The class for performing instance segmentation is imported and we created an instance of the class.

```
segment_image.load_model("mask_rcnn_coco.h5")
```

This is the code to load the mask rcnn model to perform instance segmentation.

```
segment_image.segmentImage("path_to_image", output_image_name = "output_image_path")
```

This is the code to perform instance segmentation on an image and it takes two parameters:

path_to_image: The path to the image to be predicted by the model.

output_image_name: The path to save the segmentation result. It will be saved in your current working directory.

Sample2.jpg

Image's source:Wikicommons

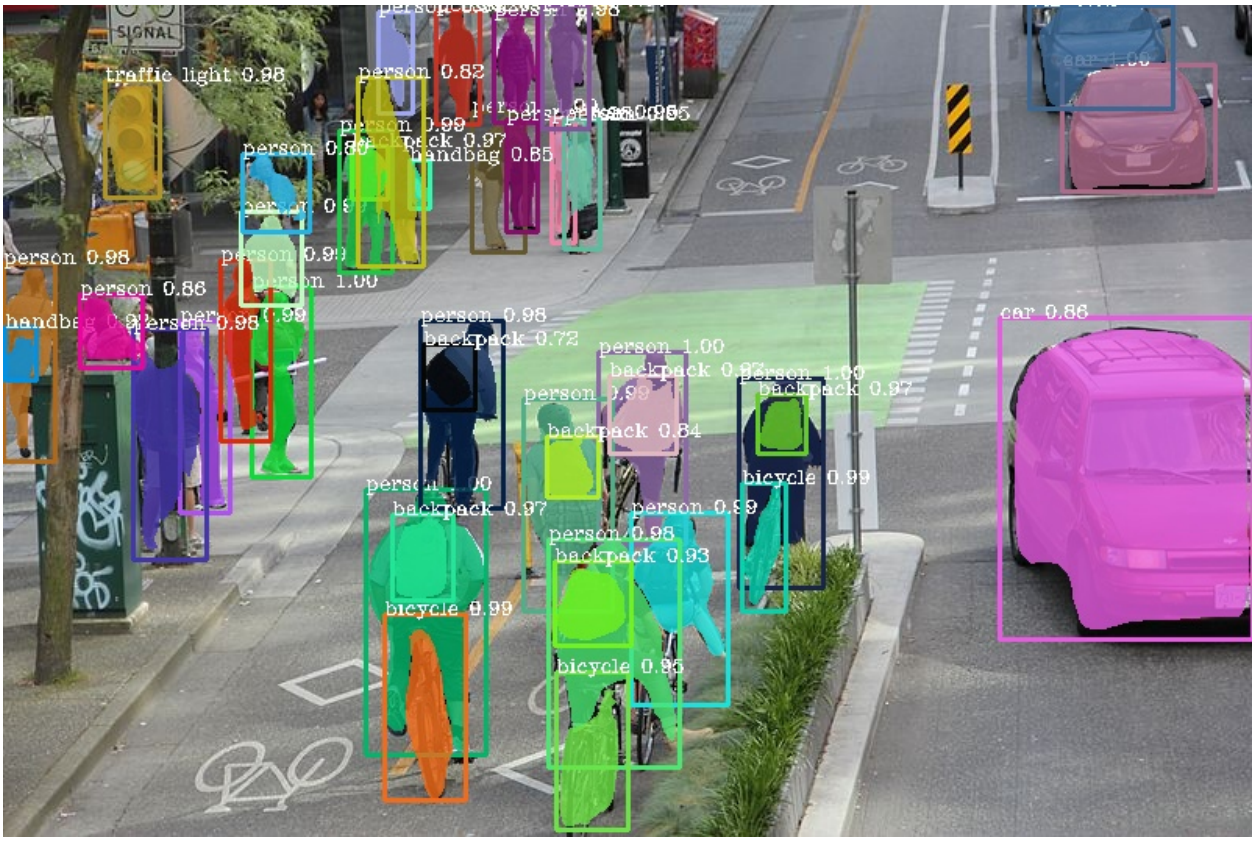
```
import pixellib
from pixellib.instance import instance_segmentation

segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")
segment_image.segmentImage("sample2.jpg", output_image_name = "image_new.jpg")
```

This is the saved image in your current working directory.

You can implement segmentation with bounding boxes. This can be achieved by modifying the code.

```
segment_image.segmentImage("sample2.jpg", output_image_name = "image_new.jpg", show_
↪bboxes = True)
```

We added an extra parameter **show_bboxes** and set it to **true**, the segmentation masks are produced with bounding boxes.



You get a saved image with both segmentation masks and bounding boxes.

- You can check the inference time required for performing segmentation by modifying the code below..

```
import pixellib
from pixellib.instance import instance_segmentation
import time

segment_image = instance_segmentation()
segment_image.load_model("mask_rcnn_coco.h5")

start = time.time()
segment_image.segmentImage("former.jpg", output_image_name= "image_new.jpg")

end = time.time()
print(f"Inference Time: {end-start:.2f}seconds")
```

```
Inference Time: 12.87seconds
```

It took 12.87 seconds to run instance segmentation on the image.

Specialised uses of PixelLib may require you to return the array of the segmentation's output.

Obtain the following arrays:

-Detected Objects' arrays

-Objects' corresponding class_ids' arrays

-Segmentation masks' arrays

-Output's array

By using this code

```
segmask, output = segment_image.segmentImage()
```

- You can test the code for obtaining arrays and print out the shape of the output by modifying the instance segmentation code below.

```
import pixellib
from pixellib.instance import instance_segmentation
import cv2

instance_seg = instance_segmentation()
instance_seg.load_model("mask_rcnn_coco.h5")
segmask, output = instance_seg.segmentImage("sample2.jpg")
cv2.imwrite("img.jpg", output)
print(output.shape)
```

- Obtain arrays of segmentation with bounding boxes by including the parameter *show_bboxes*.

```
segmask, output = segment_image.segmentImage(show_bboxes = True)
```

```
import pixellib
from pixellib.instance import instance_segmentation
import cv2

instance_seg = instance_segmentation()
instance_seg.load_model("mask_rcnn_coco.h5")
segmask, output = instance_seg.segmentImage("sample2.jpg", show_bboxes= True)
cv2.imwrite("img.jpg", output)
print(output.shape)
```

CONTACT INFO:

olafenwaayoola@gmail.com

Github.com

Twitter.com

Facebook.com

Linkedin.com