Title: README Setup Mask RCNN TF2 CannyEdge MacOS

Document Number: 102d-1 (New ID for CV100)

190l-n (Older ID, retired on June 9, 2022)

CTI One Corporation

Table 1a. Document History

2022-05-12	Establish this document, document archive: /media/harry/easystore/backup-2020-2-15/CTI/3proejcts/3-8-smart-tech/3-8-4-CTI/3-8-4-6-products/AIV200/190-robots-health/190a-unity-simulation/190a-5-moving-objects (retired)	YY, HL
2022-06-06	Update this document /media/harry/easystore/backup-2020-2-15/CTI/3proejcts/3-8-smart-tech/3-8-4-CTI/3-8-4-6-products/CV100/102-user-guide-source-code/102d-openCV-Mask-RCNN-Mac/102d-0-readme-mask-rcnn$ (New ID, folder update by HL, June 9, 2022)	ZW

Table 1b. Testing and Release Approval Form

2022-06-09	Tested by ZW and approved for release by HL	Pending for testing and approval

Table 2. References

Number	Name and URL	Note

1.	Mask R-CNN for Object Detection and Segmentation using TensorFlow 2.0 https://github.com/ahmedfgad/Mask-RCNN-TF2	
2.	Mask R-CNN release history for downloading the model file in the h5 format https://github.com/matterport/Mask_RCNN/releases	
3	Fixing the Protobuf problem https://stackoverflow.com/questions/72441758/typeerror-descriptors-cannot-not-be-created-directly	

Table 3. Prerequisite

Software Prerequisite No.	Description and Version	Note
1.	MacOS version: Bigsur or later	
2.	Anaconda version 4.7.12 or later	
3.	git version 2.17.1 or later	
Hardware Prerequisite No.	Description and Version	
1.	NVIDIA GPU (any Nvidia GPU? If not, remove this) Comments: (1) Please add the Macbook Type/Model, with memory size? (2) any GPU on the MacBook, if not, remove it;	HL June 9, 2022

Mask R-CNN using TensorFlow 2, Display concatenated image using OpenCV

1. Setup The Mask RCNN Environment

1.1. Create the Anaconda environment;

conda create -n maskrcnn2-gpu python=3.7.13 (the name can be changed)

1.2. Activate the Anaconda environment;

conda activate maskrcnn2-gpu

1.3. Clone the GitHub folder;

git clone https://github.com/ahmedfgad/Mask-RCNN-TF2.git

1.4. Download the pre-trained model;

https://github.com/matterport/Mask_RCNN/releases

Download mask_rcnn_coco.h5 from Mask_RCNN_2.0 Assets.

Copy mask_rcnn_coco.h5 to the "Mask-RCNN-TF2" folder and the "Mask-RCNN-TF2/samples" folder created at Step 1.4.

1.5. Setup the Mask-RCNN package into Python virtual environment;

cd Mask-RCNN-TF2

check the requirement.txt file, make sure the information is:

==========

numpy

scipy

Pillow

cython

matplotlib

scikit-image==0.16.2

tensorflow==2.2.0

protobuf==3.20.*

keras==2.3.1

opencv-python

h5py==2.10.0

imgaug

Ipython[all]

==========

if not, delete the texts of requirement. Copy and paste the above text to the requirement.text.

And Execute the following command:

$ pip3 install -r requirements.txt

$ python setup.py install

2. Run Mask R-CNN

2.1. Copy a JPEG file from Mask-RCNN-TF2/images/ to Mask-RCNN-TF2/samples/ and change the file name to “sample_image.jpg”

2.2. Navigate to the “samples” folder on the terminal;

cd samples/

2.3. Activate the Anaconda environment;

conda activate maskrcnn2-gpu

2.4. Set the environment variables for TensorFlow

export TF_XLA_FLAGS=--tf_xla_enable_xla_devices YY: Not Need for CPU

export TF_FORCE_GPU_ALLOW_GROWTH='true'

2.5. Execute a sample program;

python3 mask-rcnn-prediction.py

Figure 1. Original Image

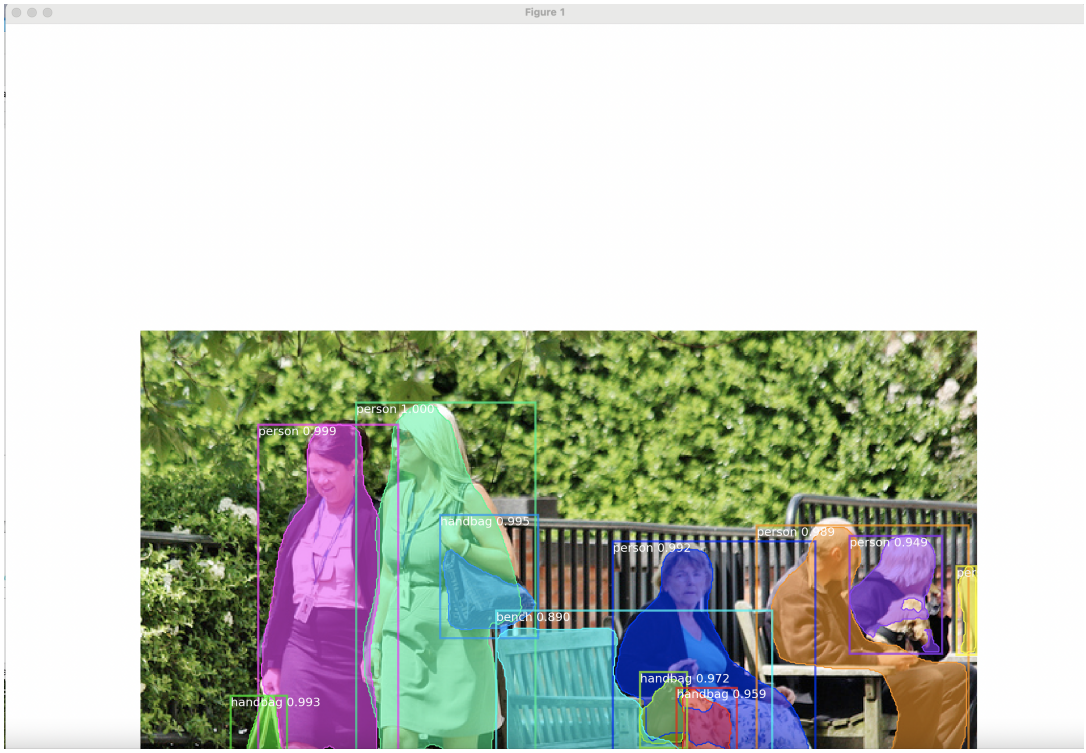

Figure 2. The Result Image

3. Create A Python Code For Concatenating Mask R-CNN And Original Images With Canny Edge Detection Based On OpenCV

3.1. Copy samples/mask-rcnn-prediction.py and create mask-rcnn-prediction_canny.py

3.2. Change mask-rcnn-prediction_canny.py as the below;

==========

```
import mrcnn
import mrcnn.config
import mrcnn.model
import mrcnn.visualize
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
```

```
# load the class label names from disk, one label per line
# CLASS_NAMES = open("coco_labels.txt").read().strip().split("\n")

CLASS_NAMES = ['BG', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack', 'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush']

class SimpleConfig(mrcnn.config.Config):
    # Give the configuration a recognizable name
    NAME = "coco_inference"

    # set the number of GPUs to use along with the number of images per GPU
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1

    # Number of classes = number of classes + 1 (+1 for the background). The background class is named BG
    NUM_CLASSES = len(CLASS_NAMES)


# Initialize the Mask R-CNN model for inference and then load the weights.
# This step builds the Keras model architecture.
model = mrcnn.model.MaskRCNN(mode="inference",
                             config=SimpleConfig(),
                             model_dir=os.getcwd())

# Load the weights into the model.
model.load_weights(filepath="mask_rcnn_coco.h5",
                   by_name=True)

# load the input image, convert it from BGR to RGB channel
image_org = cv2.imread("sample_image.jpg")
```

```
image = cv2.cvtColor(image_org, cv2.COLOR_BGR2RGB)

# Perform a forward pass of the network to obtain the results
r = model.detect([image], verbose=0)

# Get the results for the first image.
r = r[0]

# figsize = (16, 16)        # YY: figsize(16, 16) provides the image size as 1600x1600
figsize = (12.8, 7.2)       # YY: figsize(12.8, 7.2) provides the image size as 1280x720

fig, ax = plt.subplots(1, figsize=figsize)

# Visualize the detected objects.
mrcnn.visualize.display_instances(image=image,
                                  boxes=r['rois'],
                                  masks=r['masks'],
                                  class_ids=r['class_ids'],
                                  class_names=CLASS_NAMES,
                                  scores=r['scores'],
                                  ax=ax)

# convert matplotlib figure to cv2 image
fig.tight_layout(pad=0)
fig.canvas.draw()

img_detected = np.array(fig.canvas.renderer.buffer_rgba())
img_detected = cv2.cvtColor(img_detected, cv2.COLOR_RGBA2BGR)

# CannyEdge of the original image
img_org_gray = cv2.cvtColor(image_org, cv2.COLOR_BGR2GRAY)
img_org_edge = cv2.Canny(img_org_gray, 100, 200)
img_org_edge = cv2.cvtColor(img_org_edge, cv2.COLOR_GRAY2BGR)          # To concatenate with color images

# CannyEdge of the detected image
```

img_detect_gray = cv2.cvtColor(img_detected, cv2.COLOR_BGR2GRAY)

img_detect_edge = cv2.Canny(img_detect_gray, 100, 200)

img_detect_edge = cv2.cvtColor(img_detect_edge, cv2.COLOR_GRAY2BGR) # To concatenate with color images

vconcat_frame_1 = cv2.vconcat([cv2.resize(image_org, (image_org.shape[1], image_org.shape[0]), cv2.INTER_AREA),
cv2.resize(img_detected, (image_org.shape[1], image_org.shape[0]), cv2.INTER_AREA)])
vconcat_frame_2 = cv2.vconcat([cv2.resize(img_org_edge, (image_org.shape[1], image_org.shape[0]), cv2.INTER_AREA),
cv2.resize(img_detect_edge, (image_org.shape[1], image_org.shape[0]), cv2.INTER_AREA)])

hconcat_frame = cv2.hconcat([vconcat_frame_1, vconcat_frame_2])

concatenated_img = cv2.resize(hconcat_frame, (1920, 1080), cv2.INTER_AREA)

cv2.namedWindow("OpenCV Window", cv2.WINDOW_NORMAL) # cv2.WINDOW_NORMAL or cv2.WINDOW_AUTOSIZE

cv2.imshow('OpenCV Window', concatenated_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

==========

3.3. Modify mrcnn/visualize.py

Change Line 126, 127 and comment out Line 129

 # ax.set_ylim(height + 10, -10)

 # ax.set_xlim(-10, width + 10)

ax.set_ylim(height, 0)

ax.set_xlim(0, width)

 # ax.set_title(title)

```
127     # ax.set_ylim(height + 10, -10)
128     # ax.set_xlim(-10, width + 10)
129     ax.set_ylim(height, 0)
130     ax.set_xlim(0, width)
131
132     ax.axis('off')
133     # ax.set_title(title)
```

Figure 3. Modifed visualize.py

3.4. Run setup program again

Change the directory from Mask-RCNN-TF2/samples/ to Mask-RCNN-TF2

cd ..

Run the setup code

python setup.py install

4. Run The Python Program Of Mask R-CNN With Canny Edge Detection

4.1. Copy a JPEG file from Mask-RCNN-TF2/images/ to Mask-RCNN-TF2/samples/ and change the file name to “sample_image.jpg”

4.2. Navigate to the “samples” folder on the terminal;

cd samples/

4.3. Activate the Anaconda environment;

conda activate maskrcnn2-gpu

4.4. Set the environment variables for TensorFlow

export TF_XLA_FLAGS=--tf_xla_enable_xla_devices

export TF_FORCE_GPU_ALLOW_GROWTH='true'

4.5. Execute a sample program;

python3 mask-rcnn-prediction_canny.py

Figure 4. The result of Modifed mask-rcnn-prediction_canny.py

(END)