

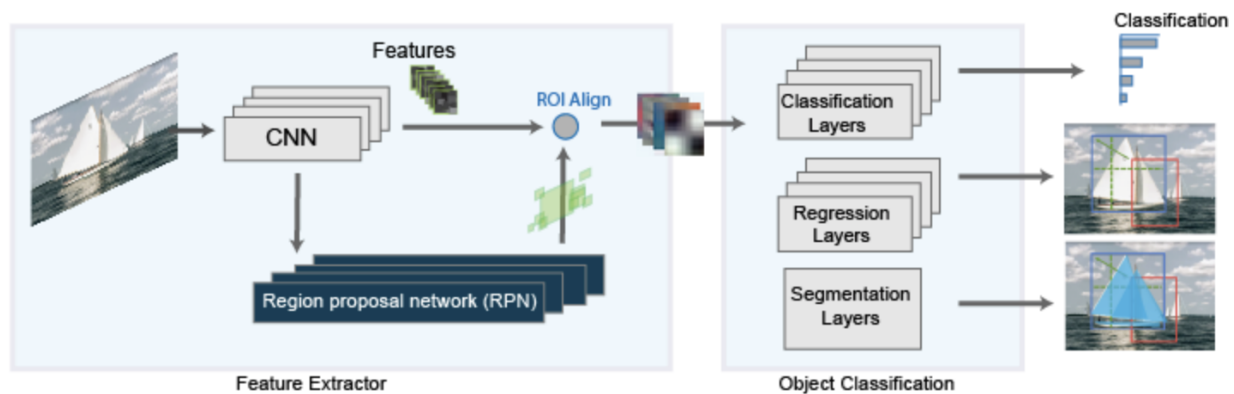
1- Image Segmentation through Matlab

1.1- Matlab Mask-RCNN

This repository demonstrates training a Mask-RCNN network to perform instance segmentation and running an inference on a few test images. The network is trained on two classes - 'Person' and 'Car' using the COCO 2014 dataset.

Instance segmentation is a computer vision technique through which we can detect and localize objects while simultaneously generating a segmentation map for each of the detected instances. The Mask-RCNN network belongs to the RCNN family of networks and builds on the Faster-RCNN network to perform pixel level segmentation on the detected objects. The Mask-RCNN network uses a Faster-RCNN network with,

1. A more accurate sub-pixel level ROI pooling layer - ROIAlign
2. A Mask branch for pixel level object segmentation.



Mask R-CNN → Faster R-CNN + FCN

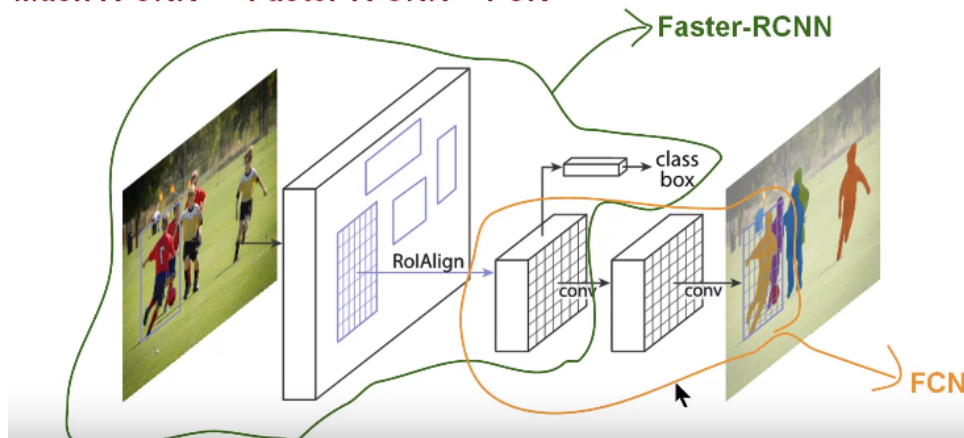


Figure 1.1A: Overview of Mask-RCNN

To download, first navigate to the src folder inside the catkin workspace. From here, we can clone the repository by:

```
$ git clone https://github.com/manjrekara/Mask-RCNN-with-ROS.git
```

For our purposes, we are interested in predicting if the images from Gazebo can be successfully segmented by the deep learning algorithm. One note to consider is that **training** an algorithm is gpu intensive and requires a lot of patience as well as high computation speeds. Due to this, we will instead **predict** certain images from Gazebo and see if the algorithm can correctly segment them based on either a ‘Car’ or a ‘Person.’

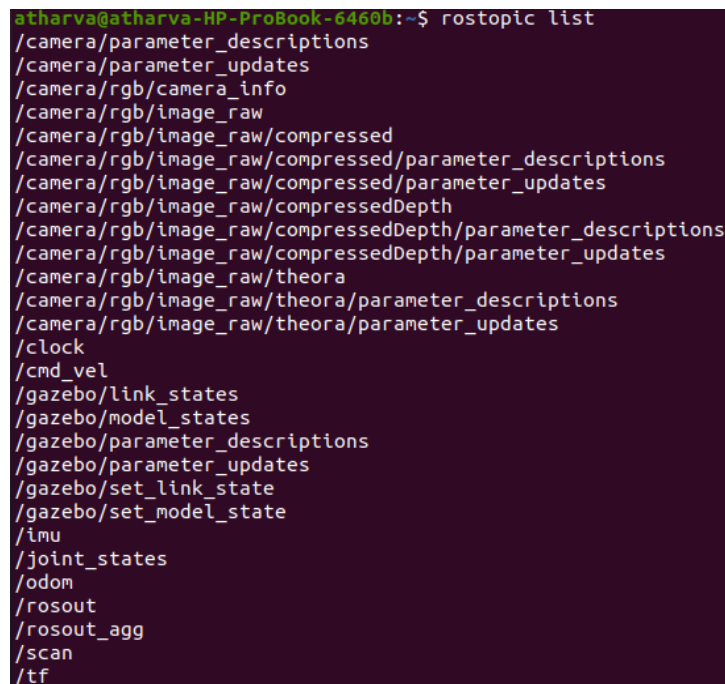
1.2- Accessing camera topic from Gazebo

In order to access specific topics we first need to list any topics by typing the command

```
$ rostopic list
```

This will list the current topics that roscore is able to pin-point as shown in Figure 1.2A below.

Note, you need to make sure you have launched a robot in the Gazebo UI via ROS for the **rostopic list** command to list the current topics.



```
atharva@atharva-HP-ProBook-6460b:~$ rostopic list
/camera/parameter_descriptions
/camera/parameter_updates
/camera/rgb/camera_info
/camera/rgb/image_raw
/camera/rgb/image_raw/compressed
/camera/rgb/image_raw/compressed/parameter_descriptions
/camera/rgb/image_raw/compressed/parameter_updates
/camera/rgb/image_raw/compressedDepth
/camera/rgb/image_raw/compressedDepth/parameter_descriptions
/camera/rgb/image_raw/compressedDepth/parameter_updates
/camera/rgb/image_raw/theora
/camera/rgb/image_raw/theora/parameter_descriptions
/camera/rgb/image_raw/theora/parameter_updates
/clock
/cmd_vel
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/imu
/joint_states
/odom
/rosout
/rosout_agg
/scan
/tf
```

Figure 1.2A: Robot Topics

For the purposes of this section, we are interested in any camera or sensor topics from our robot. Once we have located such topics we can call them by using a series of commands in the Matlab terminal.

```
$ imgSub = rossubscriber('/camera/rgb/image_raw');  
$ imgMsg = receive(imgSub);  
$ img_curr = readImage(imgMsg);  
$ imshow(img_curr);
```

The **imshow** command will show us an output image as received by the camera on the robot in Gazebo. Now that we are able to access such a camera topic, we can use it to run through several deep learning algorithms present in Matlab. As shown in the Figure 1.2B below, we placed a standing person in front of the robot camera. The **imshow** command shows us what the robot sees through it's camera sensor.

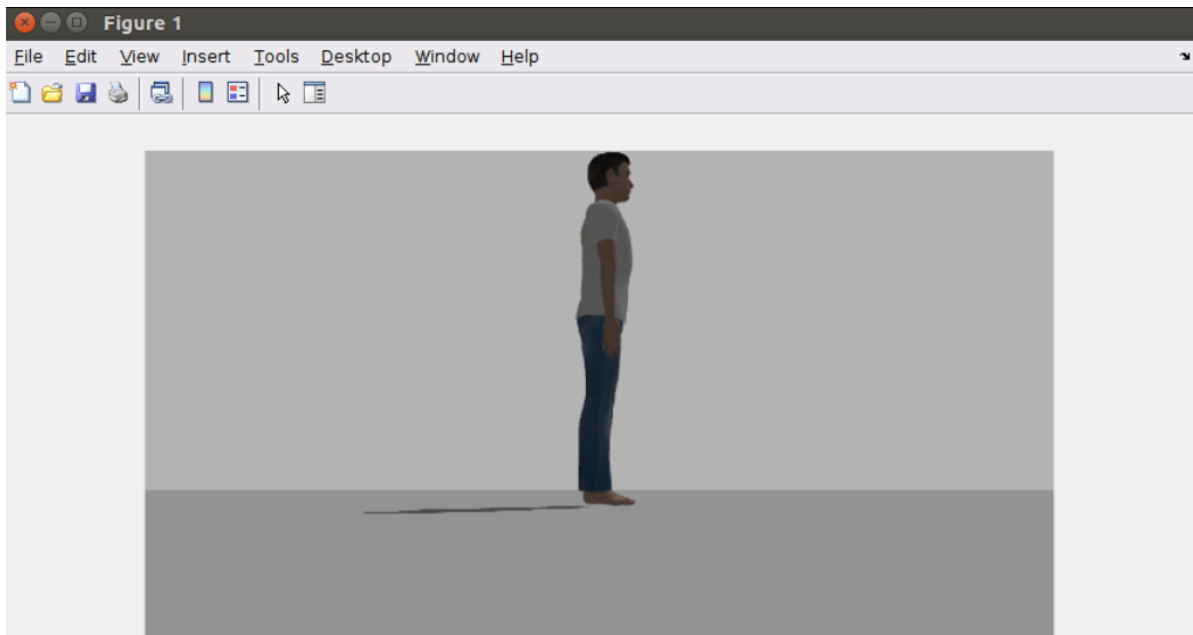


Figure 1.2B: Standing person in Gazebo captured through a robot camera

3.3- Image Segmentation

The test.m file that I wrote is shown in Figure 3.3A below. This test.m file will communicate with the detectMaskRCNN.m file which is downloaded when we clone the repository.

```

1 % Read the image for inference
2
3 img = imread(img_curr);
4
5 % Define the target size of the image for inference
6 targetSize = [700 700 3];
7
8 % Resize the image maintaining the aspect ratio and scaling the largest
9 % dimension to the target size.
10 imgSize = size(img);
11 [~, maxDim] = max(imgSize);
12 resizeSize = [NaN NaN];
13 resizeSize(maxDim) = targetSize(maxDim);
14
15 img = imresize(img, resizeSize);
16
17 % detect the objects and their masks
18 [boxes, scores, labels, masks] = detectMaskRCNN(net, maskSubnet, img, params, executionEnvironment);
19
20 % Visualize Predictions
21
22 % Overlay the detected masks on the image using the insertObjectMask
23 % function.
24 if isempty(masks)
25     overlayedImage = img;
26 else
27     overlayedImage = insertObjectMask(img, masks);
28 end
29 figure, imshow(overlayedImage)
30
31 % Show the bounding boxes and labels on the objects
32 showShape("rectangle", gather(boxes), "Label", labels, "LineColor", 'g')

```

Figure 1.3A: Matlab Test Script that runs the Mask R-CNN algorithm

```

% Read the image for inference
img = imread(img_curr);
% Define the target size of the image for inference
targetSize = [700 700 3];
% Resize the image maintaining the aspect ratio and scaling the largest dimension
to the target size.
imgSize = size(img);
[~, maxDim] = max (imgSize);
resizeSize = [NaN NaN];
resizeSize (maxDim) = targetSize (maxDim);
img = imresize(img, resizeSize);
% detect the objects and their masks
[boxes, scores, labels, masks] = detectMaskRCNN(net, maskSubnet, img, params,
executionEnvironment);
% Visualize Predictions
% Overlay the detected masks on the image using the insertObjectMask ** function,
if (isempty(masks))
overlayedImage = img;
else
overlayedImage = insertObjectMask(img, masks);
end
figure, imshow(overlayedImage)
% Show the bounding boxes and labels on the objects
showShape(*rectangle", gather(boxes), 'Label*', labels, 'LineColor','g')

```

Figure 1.3B shown below displays the limitations of the algorithm as the car's label is output; however, the person's label is not shown. We think this is due to the camera angle and the algorithm not being able to fully decipher what object is in the image because of this non ideal angle.

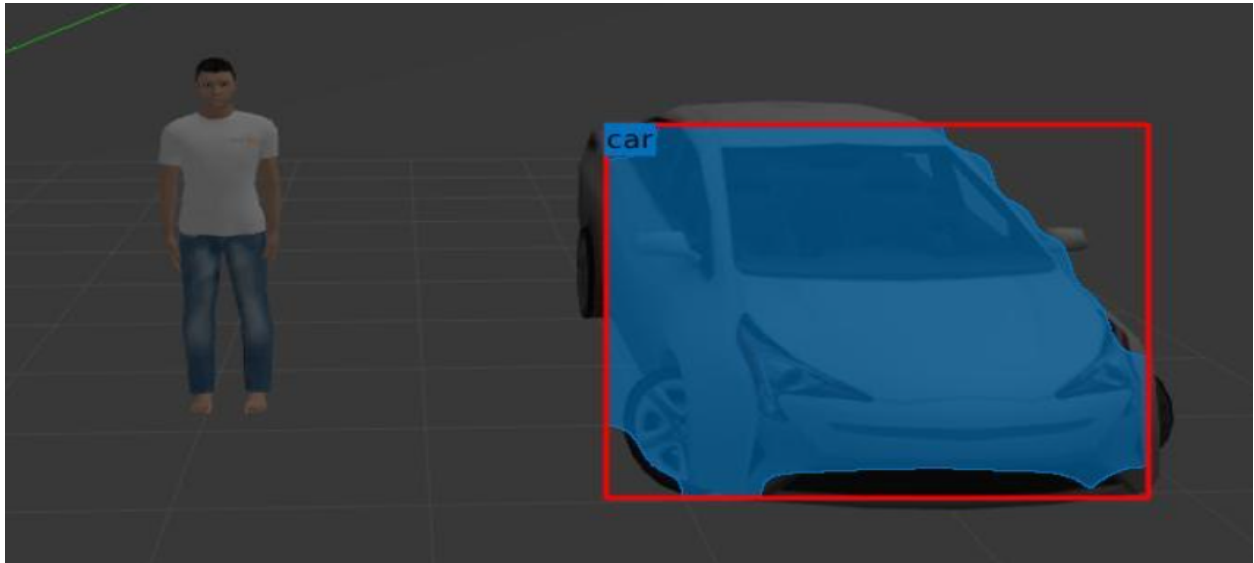


Figure 1.3B: Incorrect segmentation of a person due to a non ideal angle

Figure 1.3C shown below is another example of incorrect instance segmentation. Here, the SUV's lighting is clashing with the Gazebo background causing no label to be displayed for the SUV. The sedan has a uniform background color which helps with an accurate segmentation.

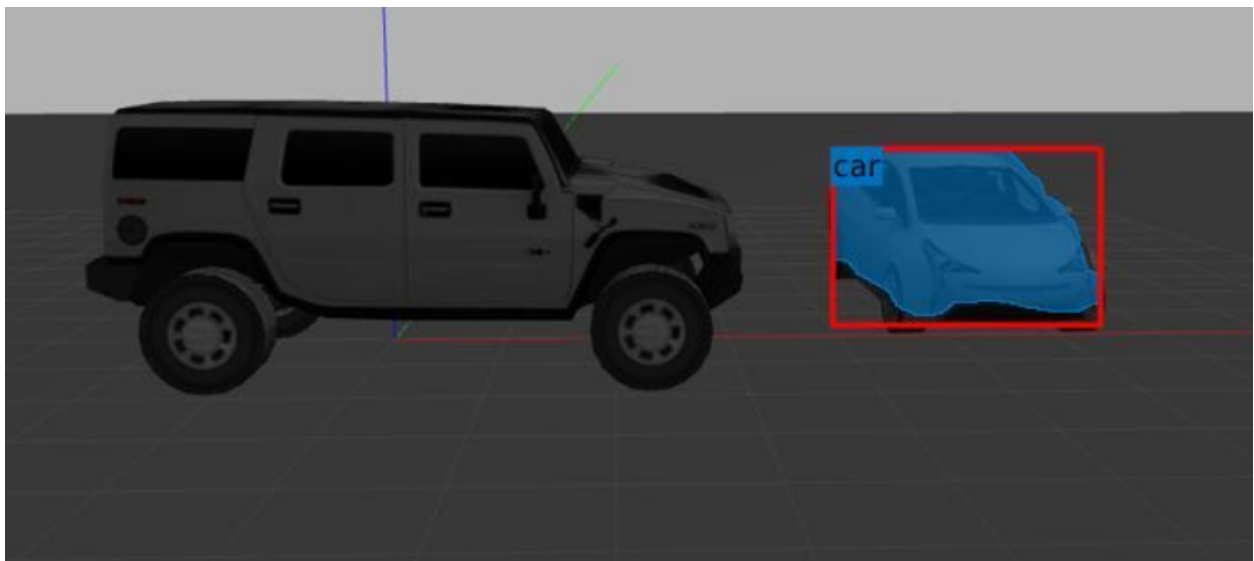


Figure 1.3C: Incorrect instance segmentation of the SUV due to background discoloration

Figure 1.3D shown below is the output that we get when the camera is focused to the point of center for both person and car. Here, the segmented output is accurate and both classes are being identified by the algorithm.

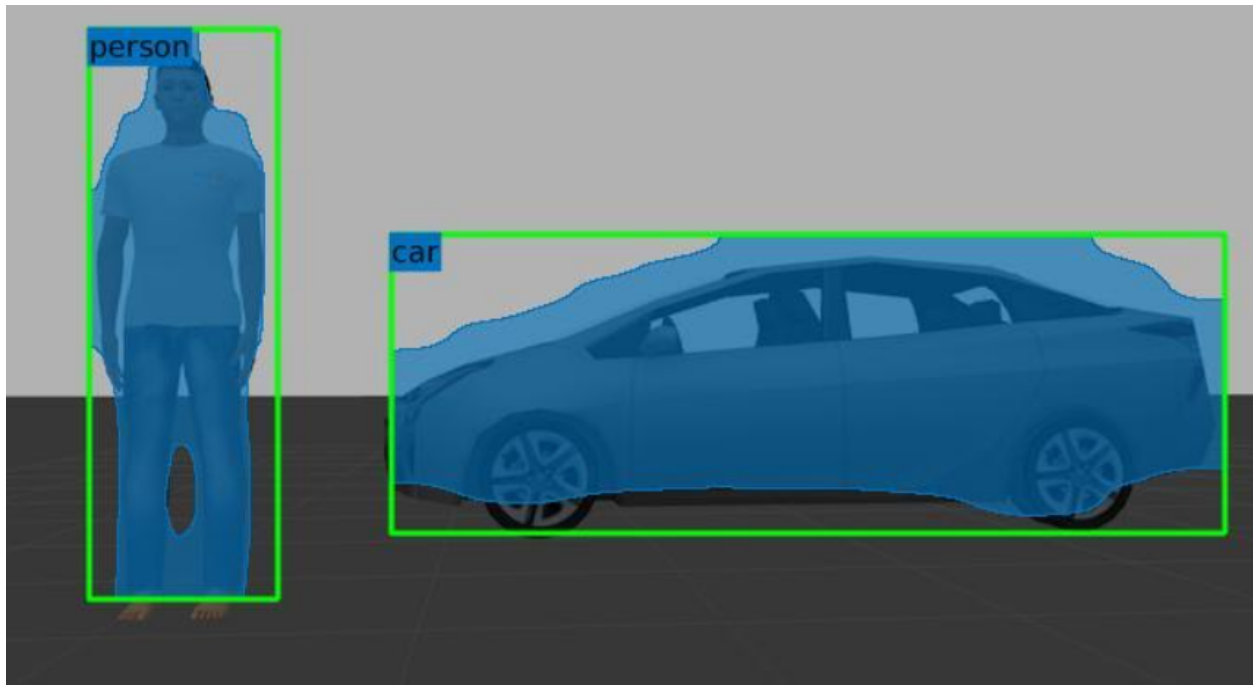


Figure 1.3D: Correct image segmentation on person and car

References

[Getting Started with Mask R-CNN for Instance Segmentation - MATLAB & Simulink \(mathworks.com\)](#)