Computer Vision Project

Virtual Background Creator

Smit Sheth shs572@nyu.edu

Akshat Khare ak7674@nyu.edu

Utkarsh Nath un270@nyu.edu

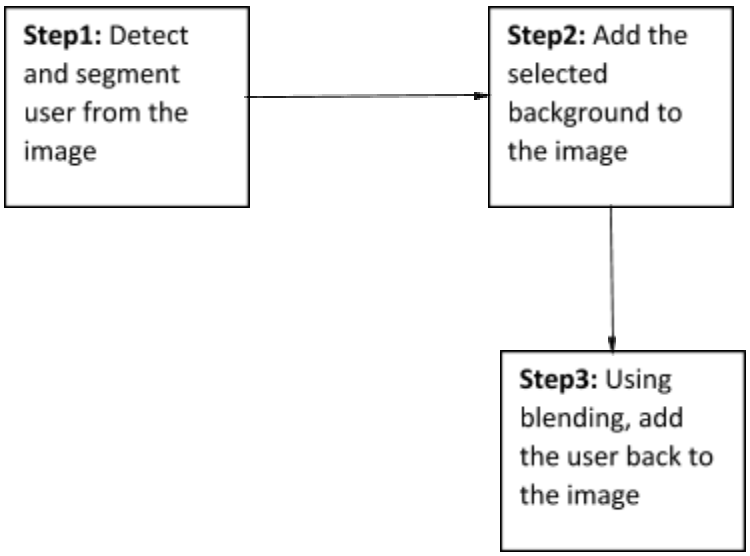
Vishnu Thakral vvt223@nyu.edu

Project Description:

Millions of people have taken to online video applications to work and study from home during this pandemic. Whether you are taking classes from home, working from a messy, cluttered room, or simply don't want to show your work background, the virtual background feature will help you maintain your privacy. The feature will allow you to set the background either as outer space or as a Brooklyn bridge and have your colleagues only see you and not whatever is behind you.

To achieve the best virtual background effect, Video conferencing apps like Zoom recommend having a solid color background preferably a green one. Along with higher quality cameras and uniform lighting to get a better virtual background. Like in any other basic image separation technique it is good to have the foreground (dress in this case) and background of different colors. We try to use the Neural Networks to segment the user from the image thus helping us overcome some of the factors like high resolution cameras which might not be in control of the user. As a result we achieve a better virtual background which stays consistent and thus ensures private and comfortable video calls.

Methodology:



Step1: The first step in our methodology is to segment the human present in the frame of a video stream using segmentation technique. Only the first person using the application should be segmented and the rest of the humans in the frame should be ignored. This is ensured by taking into account the frame to person ratio for each human in the image. In group conference calls this works equally fine when the ratio of frame occupied is almost the same for all the humans. We presently focus on one person segmentation.

If the segmentation is not done perfectly the human present in the frame will not be segmented properly, something similar to what we encountered in our video call on Zoom as shown in Figure 1.

Figure 1

Step2: Once the person in the frame is segmented we add the virtual background behind the selected person in the frame. This is done by doing a basic overlap of 2 images, but the result is not that great as we get uneven edges in the segmented image.We have also added a sample of such unevenly blended images with a virtual background.

Step3: We smoothen the images by using alpha blending technique for smoothing effect when the person's masked image is to be added to the virtual background image.

Expected results:

Fig.2(a)

Fig.2(b)

Fig.2(c)

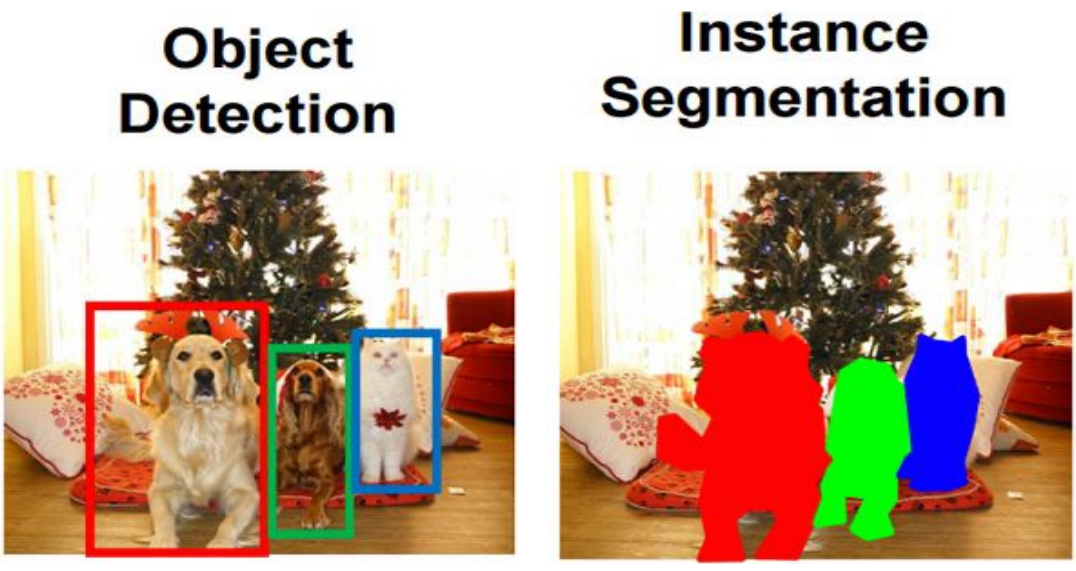
Fig.2(d)

From figure 2a we can observe a user who does not use any background. The user in fig 2b) uses a background but the image segmentation has not been done perfectly thus resulting in improper final image. Fig 2c) is decent from the part of segmentation but has a rough blending thus resulting in us seeing white boundary behind the person's face outline. Fig 2d) is a perfect use case of virtual background even in low light. This is something that we target to achieve, i.e to have a good blended virtual background even in low light conditions (the person in the image is not well lit).

Only the person should be visible in the video and the background should be the one that is selected by the user. The video should look such that the person is sitting in the virtual background and taking the video call.

Object Detection and Instance Segmentation:

Explaining instance segmentation is best done with a visual example — refer to the figure above where we have an example of object detection on the left and instance segmentation on the right



Looking at these two examples we can clearly see a difference between the two.

When performing object detection we are:

1. Computing the bounding box (x, y)-coordinates for each object
2. And then associating a class label with each bounding box as well.

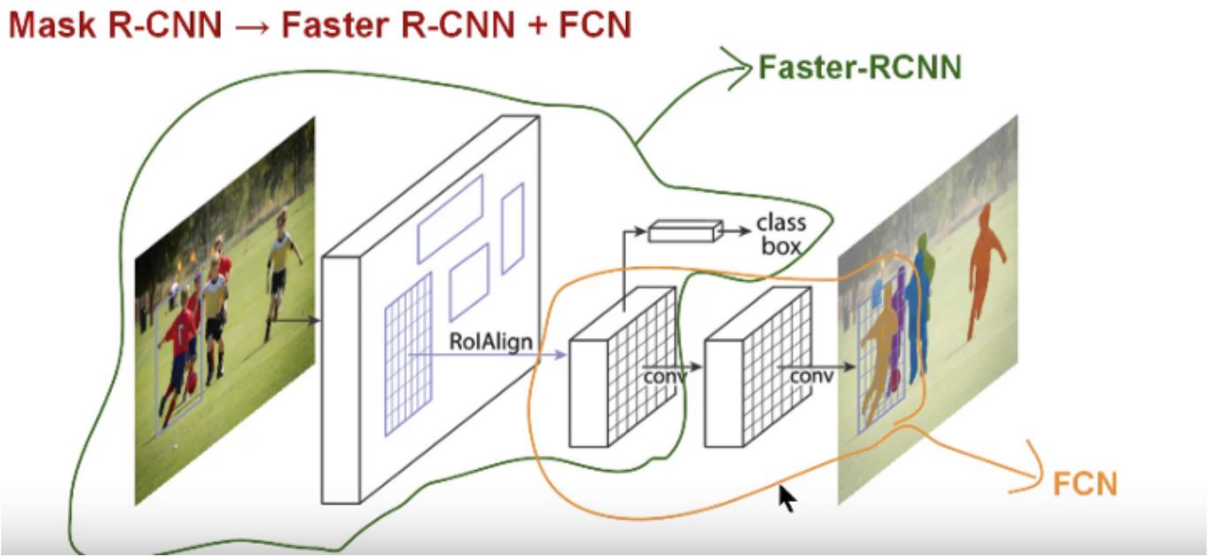
The problem is that object detection tells us nothing regarding the shape of the object itself — all we have is a set of bounding box coordinates. Instance segmentation, on the other hand, computes a pixel-wise mask for each object in the image.
Even if the objects are of the same class label, such as the two dogs in the above image, our instance segmentation algorithm still reports a total of three unique objects: two dogs and one cat.

Using instance segmentation we now have a more granular understanding of the object in the image — we know specifically which (x, y)-coordinates the object exists in.

Furthermore, by using instance segmentation we can easily segment our foreground objects from the background.

We'll be using a Mask R-CNN for instance segmentation in our project.

Training the model:



The Mask R-CNN model can be divided into two parts, the first part is computing the bounding box around the region of interest (in this project ROI is human) and the second part is extracting features from the obtained bounding box and performing classification and regression.

The model gives three outputs
1. Class label (Our case it's only human and other background)

2. A bounding box (4 coordinates)

3. A list of coordinates need to define the segmented portion.

The loss for this model is the total loss in doing classification, generating bounding box and generating the mask.

We used the Mask RCNN model implemented in [1] to perform training on only person images from MS COCO dataset. There were just 2 classes: one person and another background. We use pre-trained weights previously trained on MS COCO dataset and fine-tuned the last layers of the network to achieve better accuracy.

Results:

We were successfully able to train the Mask RCNN model on human images available in MS COCO dataset. Then we used the model to detect a person in a given image frame. After that we successfully overlay the segmented person back into a virtual background of the user's choice and blur the background if none is given.

Link to our result :

https://drive.google.com/open?id=1BZs6sK-LPr7LRw3ZNau0iJXsLJrV7-aw

Limitations:

The first limitation is the most obvious one — our OpenCV instance segmentation implementation is too slow to run in real-time. In order to obtain true real-time instance segmentation performance, we would need to leverage our GPU. But therein lies the problem: OpenCV's GPU support for its DNN module is fairly limited. Currently, it mainly supports Intel GPUs.

Conclusions:

In this project we learned how to perform instance segmentation using OpenCV, Deep Learning and Python. We used our self trained Mask RCNN model with OpenCV to perform instance segmentation. Our results were similar to Zoom virtual background features. However, we could not obtain true real-time performance since OpenCV's GPU support for the DNN module is currently quite limited.

We also tried to use the Kalman filter which is a computationally efficient, recursive, discrete, linear filter to work on frames captured over time i.e it gives estimates of past, present and future states of a system even when the underlying model is imprecise or unknown. The filter theoretically removes statistical noise and other inaccuracies, thus can help boost us in tracking a person who does not move much from one frame to another. This method did not provide us with any major boost in tracking the person from one frame to another and maintaining the segment generated similar to the previous frame as in our case with very low fps the variations happen at a faster rate.

Future Scope:

1. Testing for the results produced by various tracking algorithms and see if they are helpful in tracking movement which will reduce the number of times we have to detect a person in a given frame.
2. To be able to provide the virtual background with 2 or more relevant people in the frame/scene.
3. For faster inference we could experiment with methods like layer pruning or quantization.

Link to our repository:

https://github.com/SmitSheth/Virtual-Background-Maker

References:

[1] https://github.com/matterport/Mask_RCNN