A Learning-Based Data Collection Tool For Human Segmentation

Robert Jiang
***Mentors**: Changfan Chen, Jin Guo*

Summary
The current ME full body segmentation model does not perform well when the person is not facing the camera in a typical standing position. One possibility for this performance loss could be a lack of good training data with difficult human poses such as bending over, crouching, or sitting down. Aggravated by model quantization distortion while running on the NPU, the current model failed to accurately detect humans in these challenging positions. To address the problem of training data inadequacy, this project employs a learning-based data collection system. First, OpenPose[1] is used to filter a video for suitable frames. Then by feeding the filtered frames to a start-of-art segmentation teacher model, Mask R-CNN[2], and comparing its segmentation masks with those from the current ME model, images that performed poorly on ME can be identified and collected as training data. A total of 19,084 images containing a wide variety of human poses were collected. This report will present the system design and implementation along with sample results that have been collected.

1. Introduction
Image segmentation is the process of partitioning an image into different regions that contain pixels that belong to the same class. This project will focus on human segmentation, or extracting a person from an image, using a convolutional neural network. In particular, this network will be run on DTEN ME to generate a virtual background during Zoom meetings.

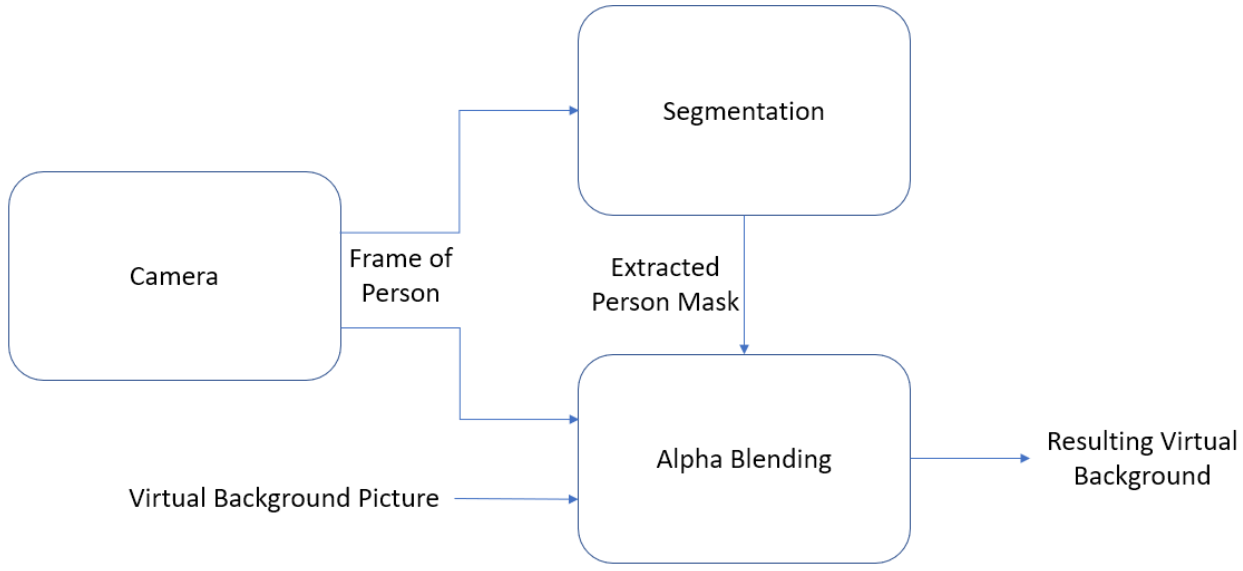

Fig. 1. Outline of Virtual Background Process.

As with all deep learning models, in order to achieve an accurate and robust model, large amounts of data need to be collected to train the model. For human segmentation, one way to generate data is by taking pictures and labeling each one manually. However, this method is very costly as it is slow and requires lots of resources and people.

This project explores another option to use automation to collect data from internet videos. This should reduce the time and resources required to help generate a train dataset. However, not all data can be used, as the data has to be tailored for the application. For this particular case, only images that contain a single full-body person are needed. To improve the ME model train dataset, a pre-trained teacher model is used to identify what sorts of images perform badly on ME segmentation.

The goal is to develop a tool that can take videos from the internet and process them automatically to select frames that are suitable to be used as train data.

2. Problem Statement

Currently, the ME model performs reasonably well on images where the person is in a typical standing position facing the camera. However, when the person is not in a neutral standing position(crouched, sideways, bending over…), the ME model struggles.

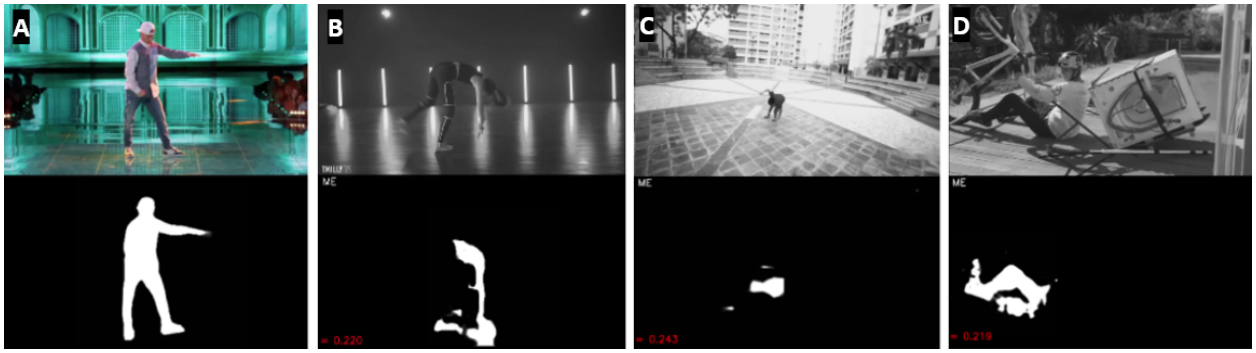

Fig. 2. A: Good Segmentation, B: Missing legs and torso, C: Missing arms and legs, D:Missing upper body

One possible reason this occurs is that there is not enough data of people in those positions to train the model with. Another reason is due to quantization of the model in order to run segmentation on ME NPU, effectively decreasing the accuracy of the model. Thus we need more data to make the model more robust on all kinds of data.

Manually gathering these kinds of data would require a team to collect and label the train data. This process is slow and requires lots of resources. But a large amount of pre-existing data can be found on the internet in videos. In particular, sports videos such as dancing, gymnastics, or parkour include many frames of people in all kinds of positions including those that are missing from the train dataset.

3. Proposed Solution

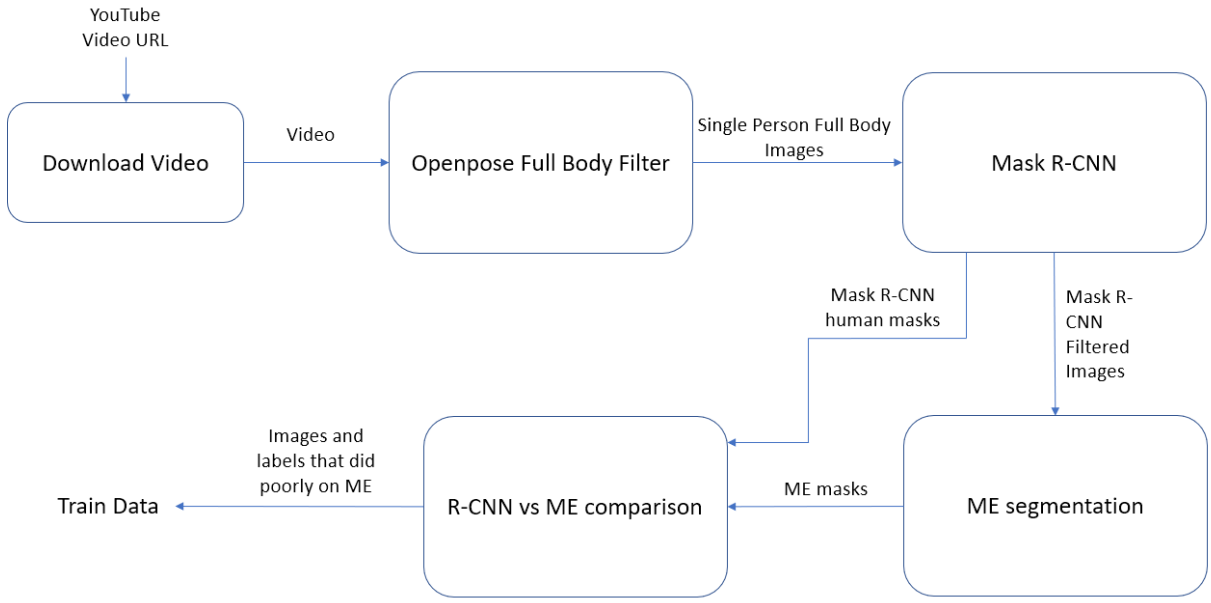

Fig. 3. Solution Outline

We propose a solution that will use pre-existing data from internet videos combined with a filtering and learning technique to enhance the train dataset. OpenPose is a human pose estimation module that will be used as the primary filtering mechanism to select frames that contain a single full-body person. The selected images will then be passed to a pre-trained teacher model, Mask R-CNN, to do a second round of filtering and generate human masks. Then the twice-filtered images will be sent to ME to generate ME masks. The learning technique primarily comes from the final step, R-CNN vs ME comparison. Here, the masks generated by ME will be compared against the masks generated by Mask-RCNN. For any given image, we want the ME mask to be similar to the Mask R-CNN mask. If the two masks differ by a certain threshold, then that image and label will be saved as training data.

4. Implementation Details

The entire project is implemented in Python. OpenPose source code is written in C++ but the Python wrapper has been used to integrate it into the program.

4.1 OpenPose Full Body Filter

OpenPose will run its detection algorithm and output a matrix of body key points that it detects in the image. These key points can then be analyzed to determine whether the image contains a single full-body person.

The points that need to be detected are the head, hands, and feet points. If all of these points are present, then most likely the complete body of the person is in the given image.

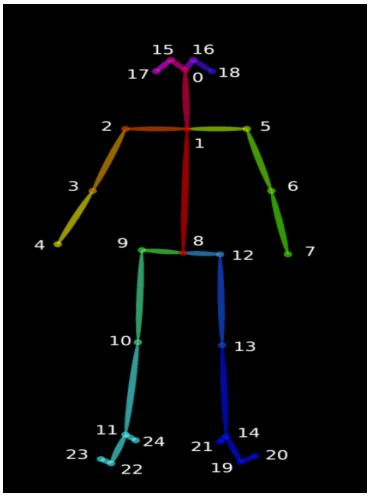

Fig. 4.1. Body 25 Key Points

Fig. 4.2. ***Left****: Head, hands, and feet points are present.* ***Middle:*** *Missing feet points.* ***Right****: More than one person.*

Looking at Fig. 4.1, the first three points to check are points 0 4 and 7, the head and both hands. Then either 19 or 20 and 22 or 23, the toes of both feet, must be present. If all these points are found then the image will be considered good.

To prevent saving frames that are too similar, after detecting and saving a good image, a certain number of frames is skipped. Assuming for a given 30 frames per second video, all frames contain a full body single person, 2 frames out of the 30 for a given second will be saved.

One major issue I ran into was with the python wrapper of OpenPose. When running OpenPose on many images with no people, around 10k in this case, OpenPose will terminate with a deallocating none error and the program will crash.

Since the iteration that errors is constant, one solution is to restart the entire process before it reaches that iteration. This can be done using multiprocessing and running OpenPose in its own

process. However, this would be inefficient since time would be wasted on restarting the process over and over again.

A better solution was found by modifying OpenPose source code. The problem occurs when the keypoints variable which contains pose information is empty, i.e., there is no person in the image, and accessing that variable through Python would return a None value. The error was resolved by modifying the C++ source code to return a non-empty array when keypoints is empty. I wrote a more detailed explanation here[3].

4.2 Mask Generation

Mask R-CNN is used to generate a bitmask to use as ground truth for training data. The pre-trained weights are trained on the COCO dataset to detect and segment 81 possible different classes. Humans are class number 1.

Additionally, this also serves as the second phase of filtering since Mask R-CNN can count how many people are in the image. This is able to filter out some more images that OpenPose incorrectly identifies, one of the problems that are discussed in section 6.1.

4.3 RCNN vs ME Comparison

To identify what sorts of images should be used to train the ME model, an Intersection Over Union, or IOU, is calculated between the Mask R-CNN generated masks and the ME generated masks. The IOU is a measure of similarity between two masks that ranges from 0 to 1. 0 means completely different and 1 means exactly the same. Those with an IOU score of less than a certain threshold, in this case, 0.5, can be saved as training data. The IOU parameter can be adjusted by the user. The higher the value, the more images that will be saved.

$$IOU = \frac{\sum(A * B) + \varepsilon}{\sum(A + B) - \sum(A * B) + \varepsilon}$$

Fig. 4.3. IOU calculation for two masks A and B. $*$ *and* $+$ *are element-wise operations.* $\varepsilon = 1 * 10^{-7}$ *is used to prevent division by 0.*

4.4 Automation

Initially, the entire process was done manually. I downloaded videos from YouTube then ran OpenPose filtering on each video to save good images. Then run Mask R-CNN and ME segmentation to generate masks. And finally use an IOU comparison to find the images to save as training data. However, this is too slow as it takes too much manual work.

All of image gathering and mask generation can be automated into a single application. Pytube is a python library that is used to download YouTube videos to the machine. Then using OpenCV we look at each frame one by one and apply OpenPose to it and save those frames that we want.

Once all videos are processed, Mask R-CNN will start up and generate masks for the saved images. So all that is needed from the user is a list of YouTube links and the program will output images with their corresponding person masks.

5. Results

I have collected a total of 55,152 images from various sports YouTube videos which include humans in all kinds of different poses. 19,084 of those scored an IOU of less than 0.5 and were saved as training data.

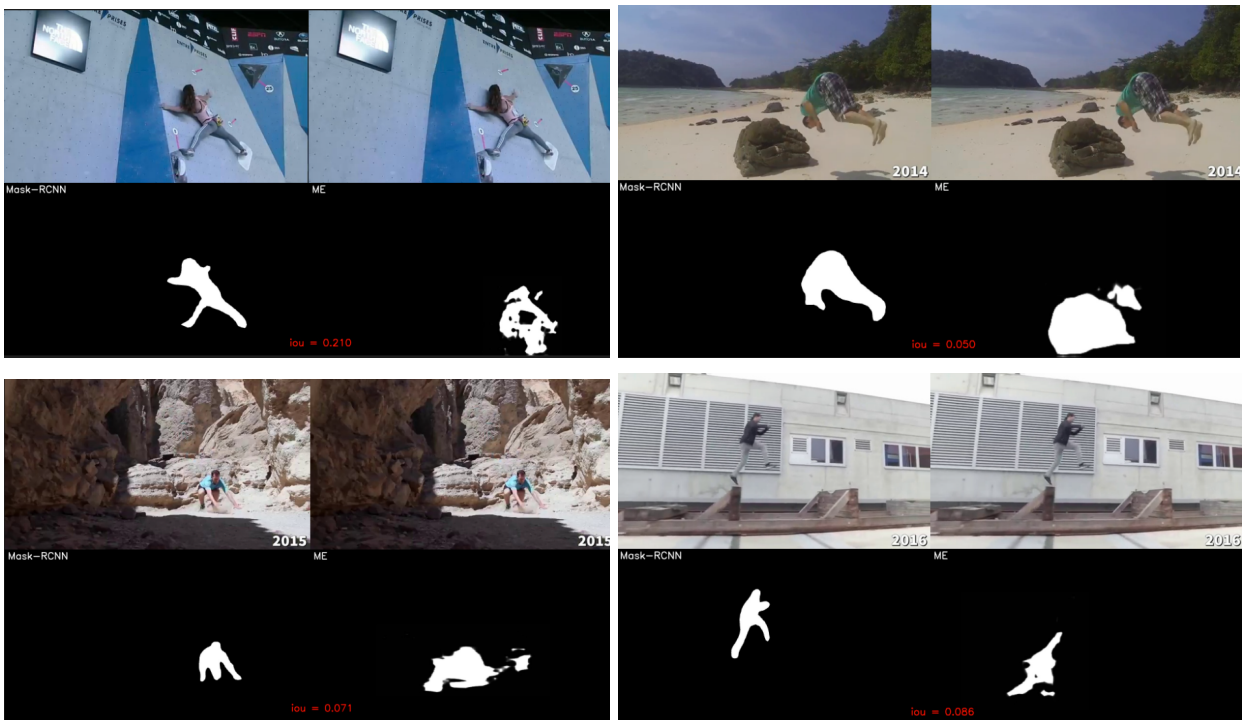

Fig. 5.1. Examples of different poses that perform poorly on ME. In each image, bottom right is the ME mask and bottom left is Mask R-CNN.

However, not all detections were accurate. In certain images, OpenPose is unable to accurately detect the person. Some examples are shown in Fig. 5.2.

Fig. 5.2. ***Left:*** *Did not detect the second person.* ***Middle:*** *Both feet falsely detected due to movement.* ***Right:*** *False detections.*

All 3 of these pictures were saved as training data but none of them meet the restriction of a single full-body person. Possible solutions are discussed in section 6.1.

6. Discussion

6.1 OpenPose

With regards to the accuracy of OpenPose, I have tried modifying the net resolution and using a maximum accuracy configuration provided by OpenPose authors. However, this did not result in better detections. The 3 images in Fig. 5.2 are detected with maximum accuracy configuration but still yield inaccurate results.

One solution could be to improve the filter itself. We can run the images through more human detection algorithms to further filter out bad images. Or we could look for better algorithms than OpenPose and Mask R-CNN to get better results.

Another solution could be to filter out the bad images by hand. This would take more time than doing it with a computer. However, using this tool to do the preliminary filtering and mask generation will save much more time than doing the entire data collection process manually. This is because the program will have already filtered out a subset of the entire video and generated the labels. The only manual work would be to look through the subset and remove those that do not meet the restrictions.

6.2 Pytube

Pytube is not supported by Google and sometimes errors will occur when Google changes Youtube internally. The only way around this is to manually download and wait for a Pytube fix.

6.3 Possible Optimizations

Currently, OpenPose and Mask R-CNN run one after the other, meaning the images that OpenPose selects have to be saved in a temporary folder before passing on to Mask R-CNN. The reasoning behind this choice is because there is not enough GPU memory to run both OpenPose and Mask R-CNN simultaneously on the current machine with 12GB. If memory allows for, a more efficient method is to run OpenPose and Mask R-CNN together, removing the need for a temporary folder.

Additionally, since videos can have varying frame rates, dynamically setting the number of frames to skip after each good image is saved can yield better results.

Another area of improvement would be to automate Mask R-CNN vs ME comparison. Currently, this step involves manually sending the images onto ME and then bringing the results back to run

the comparison. Automating this step would increase efficiency. One way to do this could be to emulate ME segmentation model on the same machine that is running data collection so that it will automatically run ME segmentation after collecting data. Another possibility is creating a script to automatically send the data to ME and back.

7. Conclusion

Collecting good train data is a difficult task that requires lots of time and resources. Although this project does not completely remove the need for manual work, it provides a way to automate some parts of the process. More work will need to be done in order to use this program as a standalone data collection tool. As of now, this tool can be used to collect large amounts of potential train data from internet videos to create a more diverse train dataset.

8. References

[1]CMU Perceputal Computing Lab, OpenPose, Github Repository, https://github.com/CMU-Perceptual-Computing-Lab/OpenPose

[2]Matterport, Mask R-CNN, Github Repository, https://github.com/matterport/Mask_RCNN

[3]OpenPose Deallocating None Solution, Github, https://github.com/CMU-Perceptual-Computing-Lab/OpenPose/issues/1967#issuecomment-880255953