CV for Surgical Applications - Final Project

Amit Zalle 213126402
Sergey Ilizirov 211624788

November 2024

Abstract

Under the course CV for Surgical Applications in the Technion University, we were given as an assignment the following instructions.

Given objects files of needle driver and tweezers, camera.json file with camera's intrinsics and background photos of kind hdri and coco2017 we need to generate synthetic data and train a segmentation model on it. than refine it using domain adaptation with 2 videos of surgical operation. With this model we need to predict on other surgical videos.

We will use blender to render the synthetic data, and yolov8-seg as the model, with pseudo labeling as the domain adaptation.

1 Exploratory Data Analysis:

Our test videos are of surgical operations of sewing an hand after a surgery. During most of the data we have two hands of the surgeon operating the sewing. one of his hand holding a tweezers, and the other a needle-driver. at times other empty hands of another surgeon might get inside the frame. Due to that we have three classes of the hands: "Tweezers", "Needle-Driver", "Empty". and we want to mark the hands with a boundary boxes - each with a confidence level of the model, and the class we believe the hand belong to.

We can separate the data into two parts:

1.1 Synthetic Data Creation

For the synthetic data creation we have several items. we have object files which are used to render the needle driver or tweezers in the new image. we have 15 of the first type and 10 of the second. we have a camera.json file which contain the settings(intrinsics) of the camera capturing the image. this includes focal length on both axises, principle-point and height and weight. finally we have the backgrounds. Firstly we have 736 hdris files, and 32612 coco images. From all this files we can render synthetic data.

1.2 Tune Videos

After the creation of the synthetic data, we need to apply domain adaptatin on the model. to do so, we must use pseudo labeling on data that is similar to the test data. For that we have videos for tuning. We have two larger videos with 1.5M and 887K KB and two smaller ones with 9K and 11K.

(a) HDRI background

(b) COCOC2017 background

Figure 1: Examples of backgrounds given

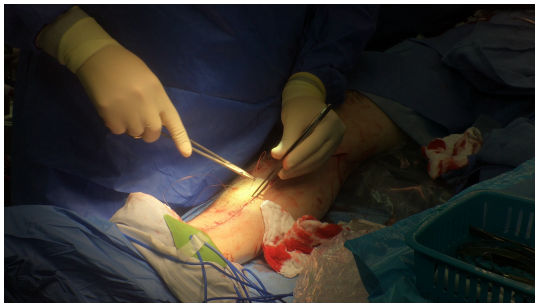


Figure 2: example of frame from video

2 Experiments

2.1 Data Generation

To render the synthetic data we used blenderproc 2.7.1. We used the given parameters for the object, background and camera and randomized the other settings in a manner we will discuss later. To use this we also created an environment called synth of conda with python 3.10.

We created 255 images with only needle driver object and hdri background, and 255 of the same kind with coco background - 255 since we have 15 types of objects so we created 17 images each. We created 250 images for each background type for only tweezers object - 25 each object. and finally created 450 for each background type with both objects - 3 for each combination of objects. We saved them all in coco format that we will change to yolo in the future. We did all that by creating a python script that go in loops over the objects files and call a command for the terminal to run a python script that create the necessary number of frames for this object and type of background.

The script of the generation of frames extract the variables that consistent to all the frames for this run - such as the object file, camera file, type of background... and randomize the rest of the parameters to create different frames. We randomize:

- Objects - We randomized the position of the objects. but we allowed overlap between them only with probability of 0.1 and otherwise assigned the second object to be randomized in the size of the image that has more space(meaning, if the first object is to the left than we randomize to the right).
- Light - We assigned a light to each object in the image and not just one light total, in order to see both objects better.
  - type- We randomize the type of the light to be either point, sun or spot.

  - position- We randomized the position of the light around the object using a shell with the center on the object, radius between 1 and 4 and elevation between 1 to 60.
  - energy- We randomized the energy of the light between 150 to 900.
  - color- We randomized the color of the light by randomizing its three channels of RGB.

- Camera - We assigned one camera to the image, and made sure all objects in the image can be seen with it after its set.
  - position- We randomized the position of the camera around the objects using a shell with the center on the object or between the objects if theres more than one, radius between 10 and 18 and elevation between -85 to 85.
  - look at point- Instead of looking directly at the object, we randomized a point close to it - offset between -0.5 to 0.5 in each direction.
  - rotation- when calculation the rotation matrix of the camera we added some uniform(-0.75,0.75) in plane rotation.

- Mist - We added some mist to the image with probability of 0.1, to create more randomization and give more complexity to the dataset.

- Background - We sampled a random image from the images of the type of the background. notice, for the coco type, we didn't render the background with the image, but overlay it later on the image.

Figure 3: Example of synthetic data

2.2 Format Correction

Firstly, as said before for the coco background we added the background as an overlay to the image, which changes the type from RGB to RGBA, and created problem with the ICCP. we ran a python script that fixed both those problems.

Additionaly, since we need the data to be in a yolo format and it is in coco format, we used the code from the "https://github.com/Koldim2001/COCO_to_YOLOv8.git" git - with some changes - to convert the dataset.

Finally, we seperated them into 4 yolo datasets. one with the images with only needle, one with only tweezers, one with only both, and one including all.

2.3 Training Pipeline

1. PreTrained Model - We start with the original Pre-Trained YOLOv8n-seg model.
2. Refine Original Data - After taking the pre-trained model, we Fine-Tune it with different yolo dataset, based on the model - needle only, tweezers only, both only, all. For that we used 50 epochs and 16 batch size.

3. Augmentations - We did not apply augmentations, since the augmentation has been done automaticly in the creation of the images, by letting the camera rotate and be able to not capture all the object. And because yolo apply augmentations automatically.

4. We examine the 4 models on the tune video, and chose the All model since it seemed to bring the best results.

The model did not seem to bring good results, especially on the tune data. it seems to confuse a lot between the needle driver and tweeers and with other objects. We believe this is because there is similarity between the shape of the two, especially when it is partially hidden by the hands. Solution we believe will work is to add an hand object to the rendering - with it we can hide part of the object and the model will learn better(we did not apply this change in our model since we assumed only the given obj are allowed to use(to examine the ability with only those) and because most models we found costs money). It is also important to mention we tried many different model, from tring the larger yolo models, changing the amounts of epochs between 10,20,30,50, changing the batch size between 8,16,32,64. trying different mixes of the images. fine tuning one model with the data of the other. using different model for the tweezers and for the needle driver. removing the "Empty" label from the classes and more. all seem to not capable to handle the task of segmenting the objects.

2.4 Domain Adaptation

For this part we used similar method as for HW1, meaning pseudo labeling with the vids_tune data. We used the All model to predict on the vids and save the frames who had conf bigger than 0.6 for the needles and bigger than 0.25 for the tweezers. Then we saved the mask by converting the binary mask to normalized polygons points and saving them.

Notice, we could see the model is predicting many tweezers near the edge of the image - which dont make sense - so we added a condition the prediction need to be near the center of the frame.

We also added all the images from the original generated data, to keep the model from getting too much based on pseudo data.

Then we refined the former model on this pseudo-data with the original rendered data as validation. We got bad result on the model - which make sense since the former model did not predict well which cause the model to train on bad predictions, and the masked that create was saved with some offset to the yolo format. so we deigned a code that predict on the test video with the pseudo and the original model, and since the pseudo failed it is very similar to the original model.

3 Discussion and Conclusions

In this project we learned a lot about rendering synthetic data using blenderproc. from the technical capabilities and operation through the different setting you can use to create wider distribution of the data.

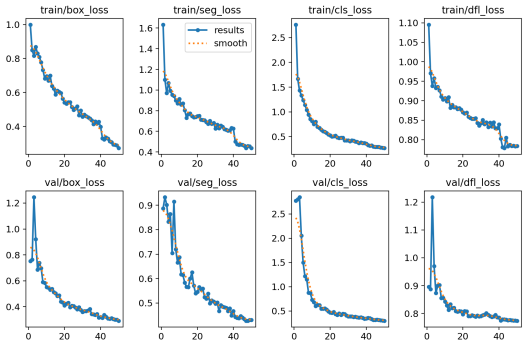
We also again worked with pseudo labeling, and this time due to bad performance seen how the effect of this on the model is very high, and dependent on the success of the original model.

We learned to operate the technical side of coco dataset and segmentation and how to convert those to yolo model, and how to save segmentation in yolo format.

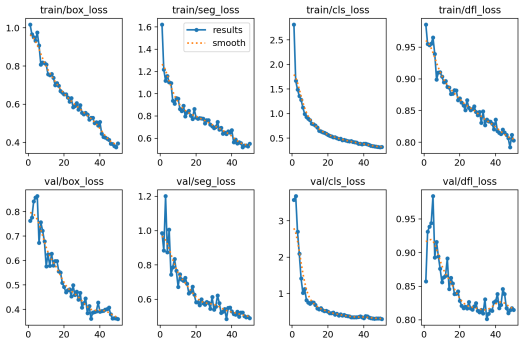
The model itself didn't show high result, but we believe with adding of hand object to the rendering the pipeline will work well and will show high results.

The code itself can be seen in the Github repository: "https://github.com/amitZalle/cvsa_FinalProject.git"

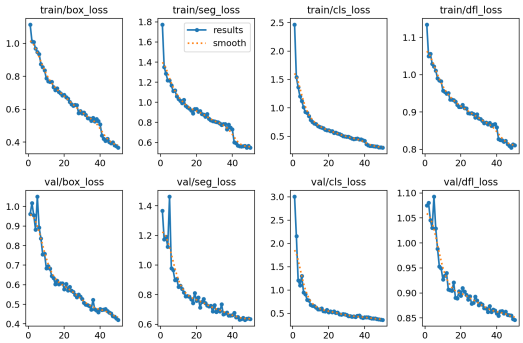
You can see the runs of the different model below:



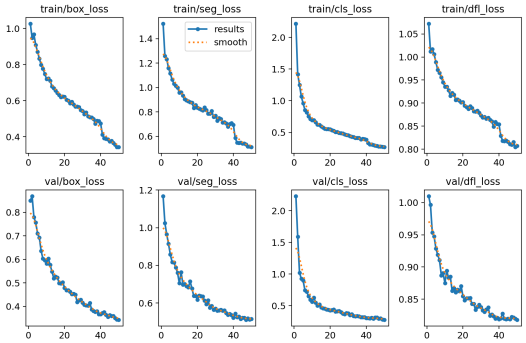
(a) trained on only needle



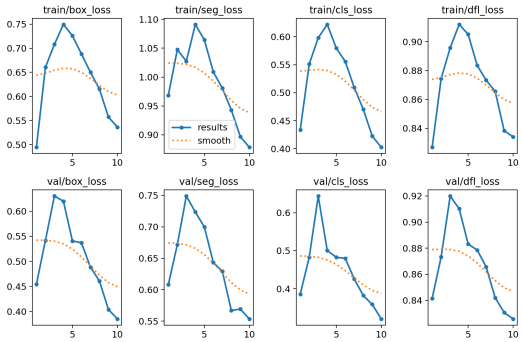
(b) trained on only tweezers



(c) trained on only both



(d) trained on all



(e) Model with pseudo data

Figure 4: Metrics over train and val of the models