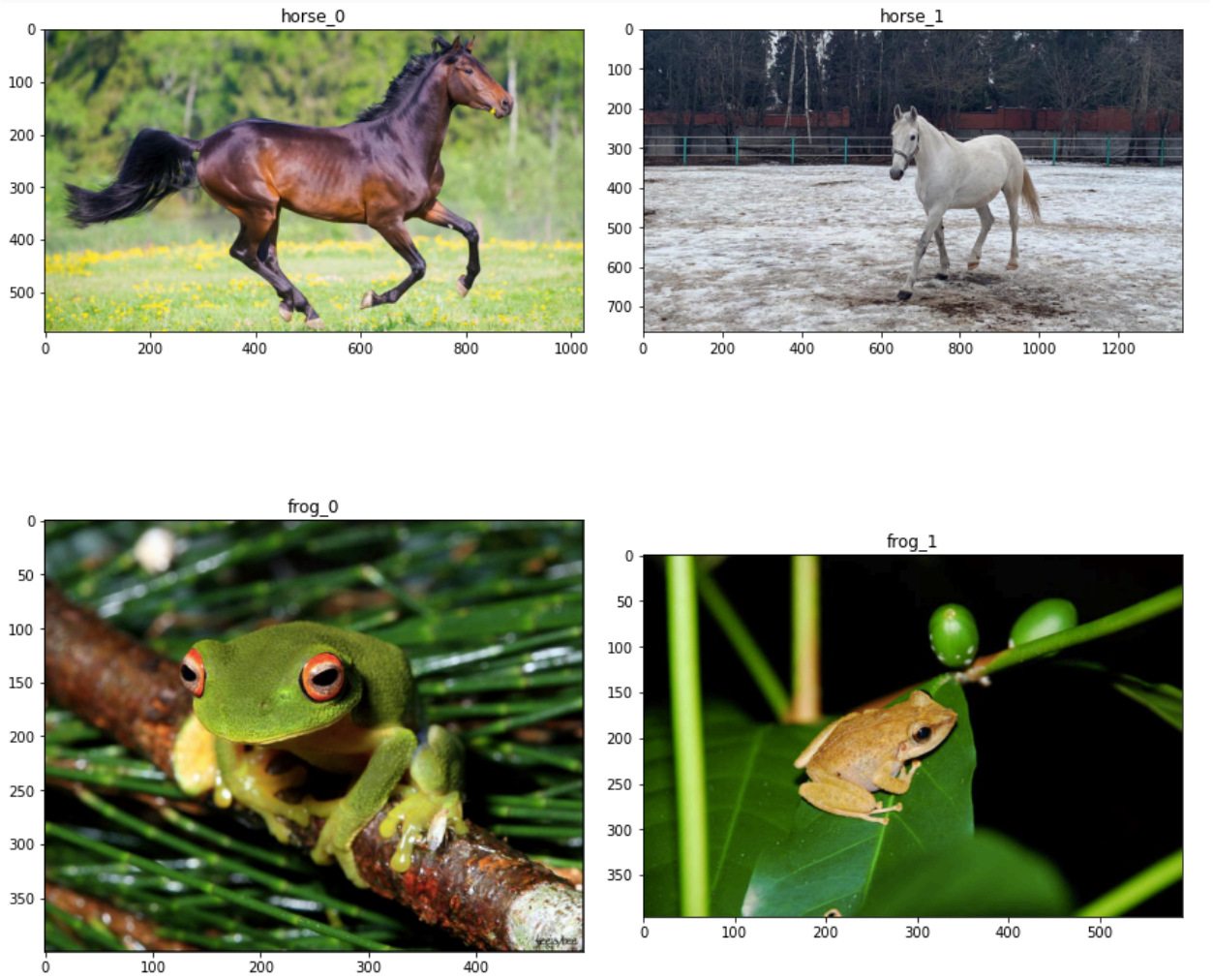
אלגוריתמים ויישומים בראייה ממוחשבת – 046746

תרגיל בית 3

Ido Nutov: 305242968
Sarah Buzaglo: 931161798

Part 1 - Classic Vs. Deep Learning-based Semantic Segmentation

1. Display the images

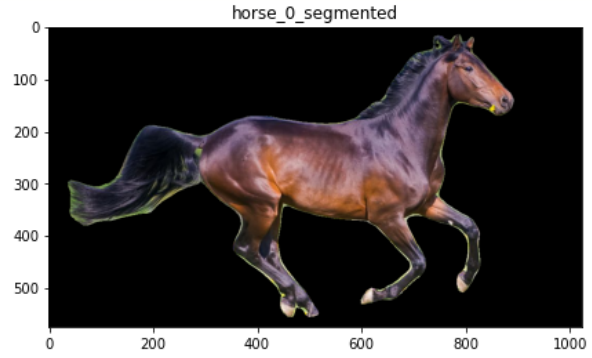


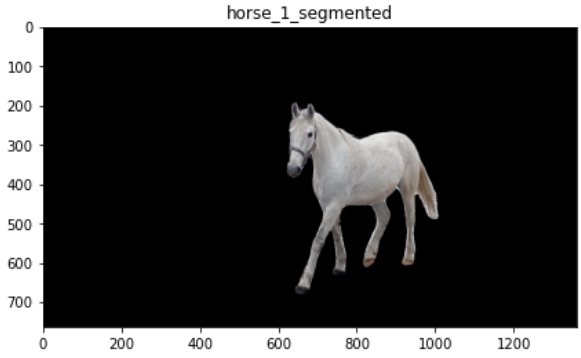
2. Segmentation

One deep learning method: `getModelDeepLabV3()`

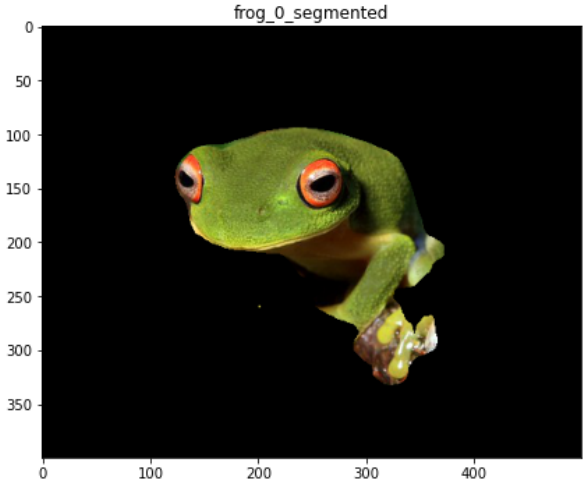
Summarize of the method

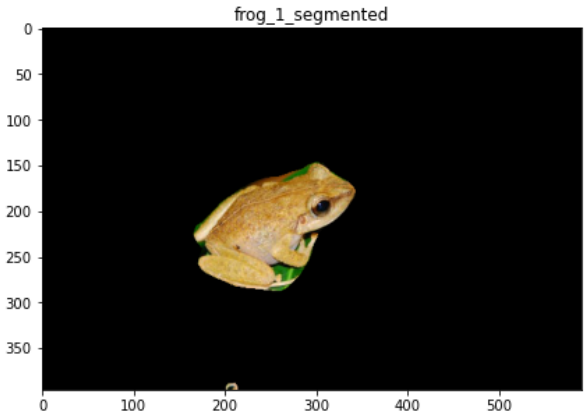
We used the pre trained neural network deeplabv3 as our deep segmentation method.
The network was trained on the PascalVoc dataset and has an encoder-decoder architecture.





We can see the network did a good job segmenting the horses, horse is one of the output categories (index 13) hence we can conclude that the dataset the network was trained on has many horse images in a variety of environments and backgrounds.





We can see that the network didn't do so well on the left frog and did reasonably well on the right frog, frog isn't one of the output categories hence both frogs were classified as birds (index 3) we suspect the network performed better on the right frog due to its color and surroundings is more plausible for a bird.

Advantages:

- A good runtime on gpu (also improves runtime as gpu's develop)
- Gives us the type of the object segmented (can be valuable information i.e. a person captured in a self drinig car camera)
- We can improve performance and increase number of categories by collecting more data

- Additional synthetic data can be automatically generated and labeled using game engines such as UE4, and training can improve on this kind of data by using a variety of training techniques such as domain adaptation technique.

Disadvantages:

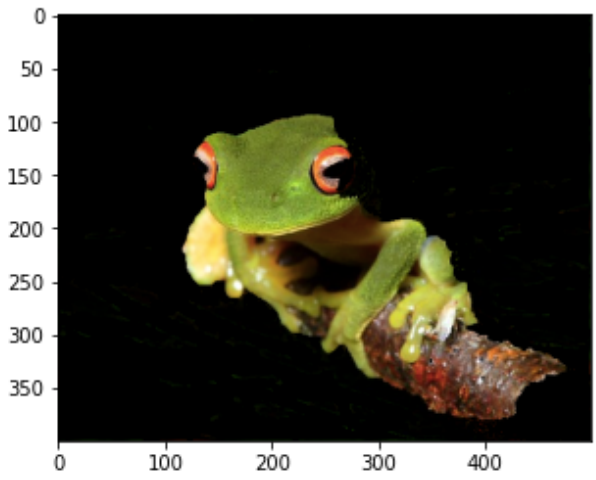
- Long training time
- For a reasonable training time and a good runtime need for gpu which is expensive
- For good results on deeplab model we need large amount of labeled data, which is hard and expensive to collect, even with synthetic data we still need an expert in a game engine to create the automatically labeled dataset
- Can give false semantics (see frog examples)
- May not give any segmentation on an object which is not in the category list of the labels of the training data (see old typing machine example)

One classic method: `cv2.grabCut`

Summarize of the method

User inputs the rectangle as a parameter in the cv2.grabCut function. Everything outside this rectangle will be taken as sure background. Everything inside rectangle is unknown. Computer does an initial labelling depending on the data we gave: it labels the foreground and background pixels. Then, a Gaussian Mixture Model (GMM) is used to model the foreground and background. Depending on the data we gave, GMM learns and create new pixel distribution. That is, the unknown pixels are labelled either probable foreground or probable background.

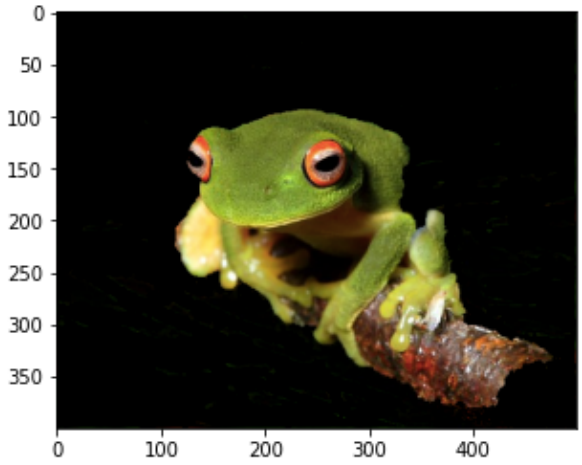
A graph is built from this pixel distribution. Nodes in the graphs are pixels. Additional two nodes are added, Source node and Sink node. Every foreground pixel is connected to Source node and every background pixel is connected to Sink node. The weights of edges connecting pixels to source node/end node are defined by the probability of a pixel being foreground/background. The weights between the pixels are defined by the edge information or pixel similarity. If there is a large difference in pixel color, the edge between them will get a low weight. Then a **mincut algorithm** is used to segment the graph. It cuts the graph into two separating source node and sink node with minimum cost function. The cost function is the sum of all weights of the edges that are cut. After the cut, all the pixels connected to Source node become foreground and those connected to Sink node become background. The process is continued until the classification converges.

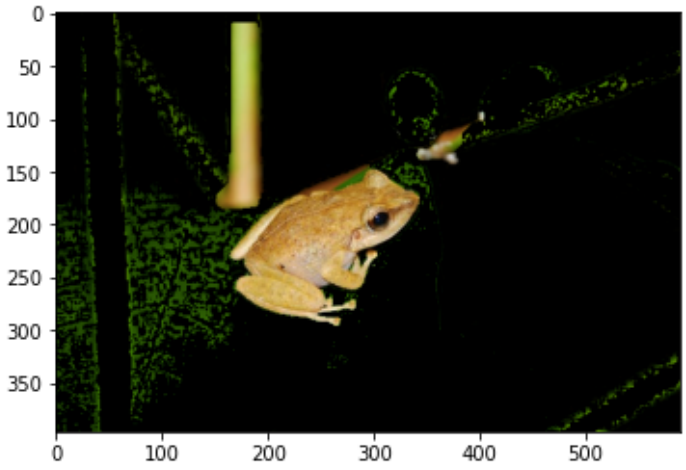


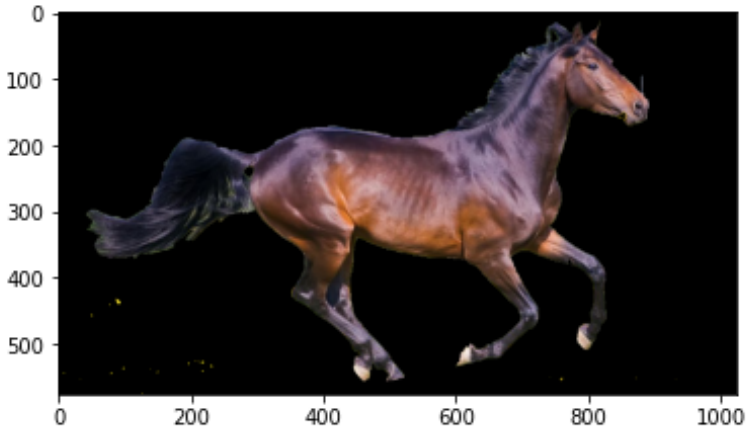
Improvement

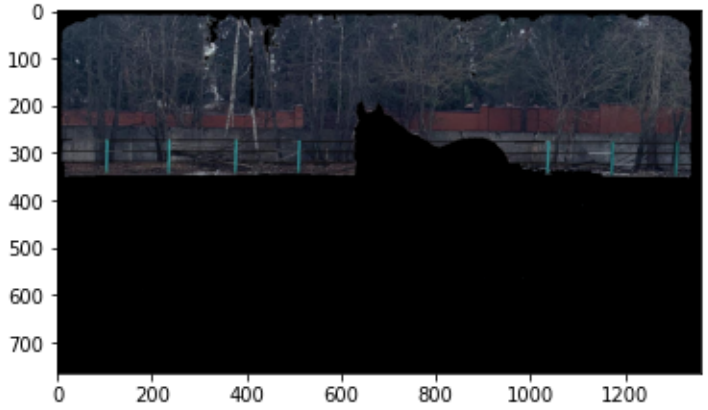
From the running of simple cv2.grabCut, we noticed that the algorithm got better results on images in which there is a sharp transition between the object and the background. So, we can think at a preprocessing step like histogram equalization (=increase the contrast) in order to get more accurate segments.

Ex: On the first frog, we get more of the back of the animal.









Advantages

- In adaptated case (sharp transition between the object and the background + one element without holes), the algorithm is very accurate (horse number one)
- Grabcut doesn't need a training on a consequent amount of data (ex: grabcut is more efficient on the typing machine that the deep learning)

Disadvantages:

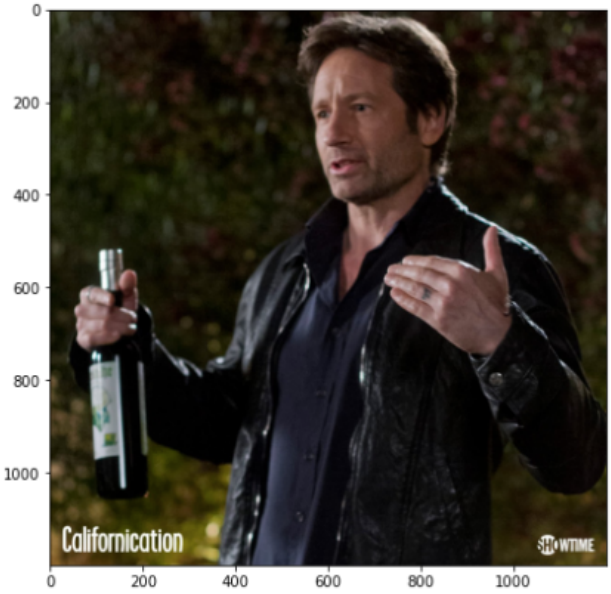
- Need initial condition on the picture: a sharp transition between the object and the background (the horse 2 isn't identified because of that)

- Need some knowledge on the picture because of the manual adjustments (size of the rectangle, touch up)
- No semantics compared to the deep learning model (we cannot tell which object was segmented)

3. **Pick 3 images and display them:**
4. **Apply each method on the 3 images.**

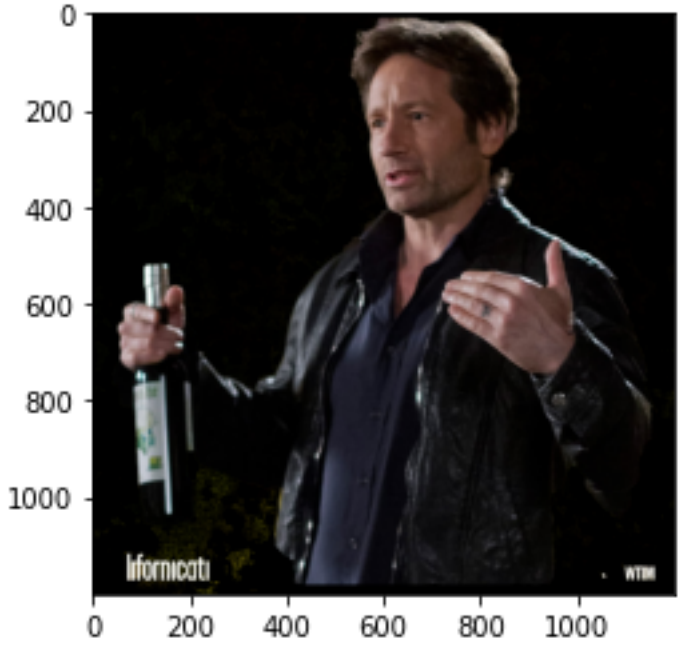
One image of a living being: Hank Moody from Californication

- Image



- `getModelDeepLabV3()`

- `cv2.grabCut`



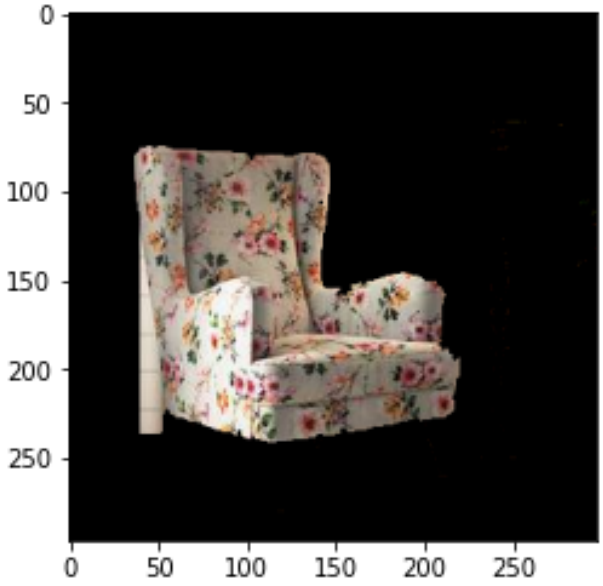
The deep learning method and the classic worked pretty well on this image but will prefer the result of the deep learning method because this one also allow a semantic segmentation and make the difference between the bottle and the human being.

One image of commonly-used object：old chair

- Image

- `getModelDeepLabV3()`

- `cv2.grabCut`

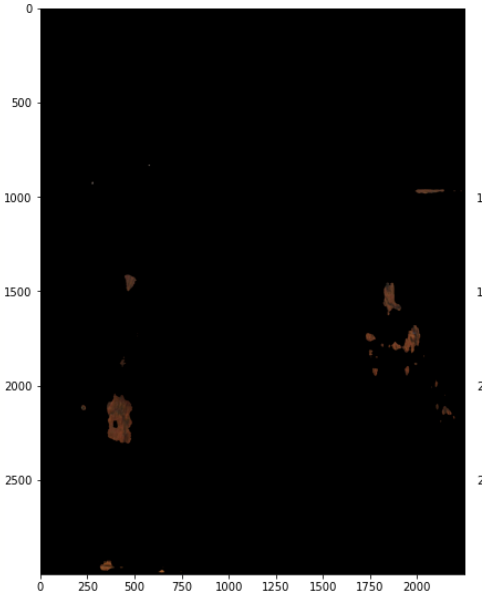


In this case, the deep learning method and the classic worked pretty well too but the deep learning model is more precise and also identify the legs of the chair. So, we will prefer it. This identification of a detail is due to the fact that the model trained with chairs.

One image of not-so-commonly-used object：typing-machine

- Image

- `getModelDeepLabV3()`



- `cv2.grabCut`

For this image, the deep learning model can't recognize the object because it didn't train on this category of objects and grabcut does a better job identifying it.

In a more general way, we can notice that the deep learning method performs a better segmentation on objects which it has trained on (compared to unfamiliar objects).
The classic grabcut method perform well and, in a case of an unfamiliar object to the deep learning model, it will perform better that the deep learning.

5. **Segmentation can be rough around the edges (ex: the mask is not perfect and may be noisy around the edges). What can be done to fix or at least alleviate this problem?**

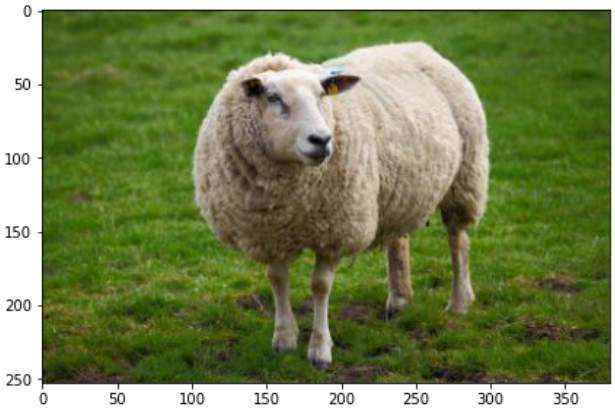
Edge effect can occur due to noise so filtering the image night be helpful, we tried this method with the chair example and got a better segmentation of the forward left leg of the chair:

Another thing we can use are morphological operators like the close operator which will fill small holes and jigsaw pattern on the edges in the segmentation mask.

6. **Load a pre-trained classifier**

Code. Done.

7. **Pick an image of an animal in its natural habitat. Display the image you chose and feed- forward it to the pre-trained network. What is the network's prediction?**



```
'ram'
```

The network prediction is a ram.

8. **Segment the animal and display the result.**

9. **Put the animal in a different habitat, i.e., use the mask to place the animal on a different background. Display the result.**

```
res = classify(classifier , beach)
labels[str(res.cpu().numpy())][1]
```

```
'sandbar'
```

10. **Feed forward the new image to the pre-trained network. Was the prediction different than Q7? If so, discuss the reasons for that to happen.**

Now, the prediction is different. It is identified as a sandbar.

We think a reason for that is that there are more pixels in the image related to ‘sandbar’ than there are to sheep.
This reasoning is further enhanced when we look at the top 3 predictions

```
[42]: torch.topk(probs, 3)

[42]: torch.return_types.topk(
      values=tensor([[0.0582, 0.0196, 0.0150]], device='cuda:0'),
      indices=tensor([[977, 978, 348]], device='cuda:0'))

[ ]:
```

which are sandbar, seashore and ram respectively.
If we want to further investigate we can create class activation maps to show us which pixels participated in the prediction of each class.

Part 2 - Jurrasic Fishbach

We worked on Google Colab environment.

2.1-2.2: We filmed Ido during a barbecue in which the camera is held still, we cut the top part of the video to be 540 pixels and padded with zeros so the final resolution was 540x960 .
We used DeepLabV3 neural network (which was discussed in the previous section) to perform Ido's segmentation.

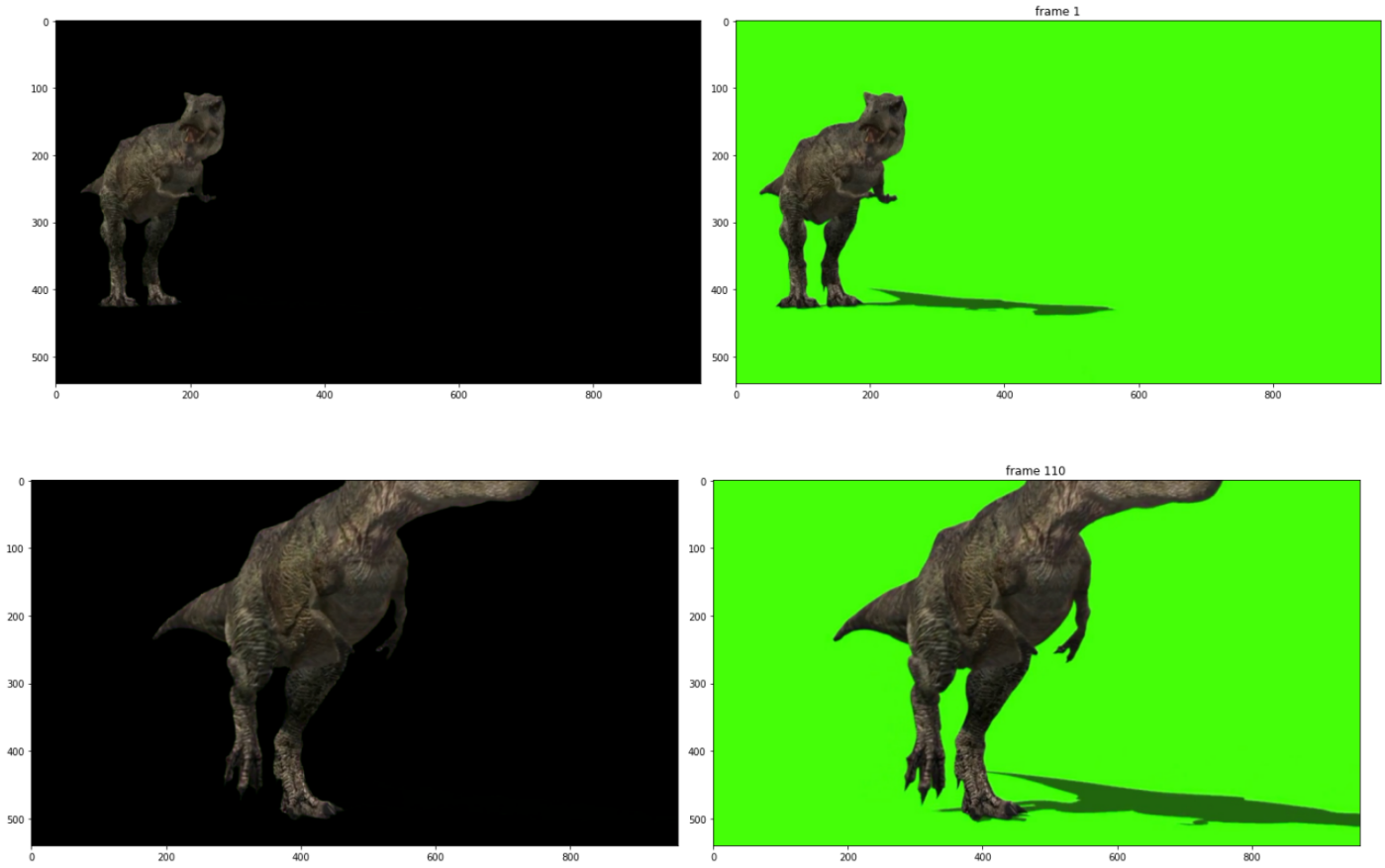
We noticed that there isn't a smooth transition between the boundaries of the segmented object and the background.

2.3:

We picked the video of the dinosaur, for green screen segmentation we used a modification to Petro Vlahos original green screen matting method to perform the green screen segmentation. We created the dinosaur segmentation mask with the following equation:

$$\boldsymbol{mask = 1 - a_1\left(I_g - a_2 I_b\right)}$$

With the image (I) scaled between 0 to 1 and I_g, I_b are the green and blue channels respectively and a_1, a_2 are tunable parameters which we optimized by a trial and error process then we clipped the mask to 0 to 1 scale and took the elementwise power of 5 of the mask for smoother transition. We can see this method gave us smooth transition between the object and the background.



We can see the results are pretty good and the dinosaur is separated smoothly from the green background.
Today there are more advanced variations of Vlahos method which includes Bayesian optimization for each frame to find the best a_1, a_2 but because we are dealing with a very short video and segmenting only the dinosaur we tuned the parameters and kept them constant during the dinosaur segmentation.

2.4: We chose the Dubbo star observatory in Australia as our background and first we blended the dinosaur into the environment and then Ido , our final result: