Image Segmentation for Scene Learning

Student name: Sébastien Lambert

Student number: 416349

Supervisor Name:

Date: Friday, 02 February 2024

Table of Contents

Table of figures

Introduction

In this assignment we are going to look at image segmentation, the different metrics that can be used to assess it regarding if we have a ground truth or not and how they are relevant. We will also compare different algorithms and ways to deal with the task we have in this assignment.

I. Selection of the appropriate segmentation method

To select the appropriate segmentation method, we need to take a closer look at the objectives. As we need to apply segmentation to find areas in an image consisting of similar pixels that can be used as a pre-processing stage for an object detection and tracking while preserving the semantic context of the image as much as possible while reducing dimensionality of the input image for the following stages of processing.

First, we know that there are different types of image segmentation: [1]

- Semantic segmentation

A computer vision task that involves classifying each pixel in an image into a specific object class or category.

- Instance segmentation

A computer vision task that extends the concept of object detection and semantic segmentation by not only identifying and classifying objects in an image but also distinguishing between individual instances of each class.

- Panoptic segmentation

A computer vision task that unifies two previously separate tasks: semantic segmentation and instance segmentation.

Here, we only need semantic segmentation. So, semantic segmentation plus the other criteria above leads us to different choices:

1. Semantic Segmentation with Deep Learning:
2. Colour-based Segmentation:
3. Superpixel Segmentation:
4. Texture-based Segmentation:
5. Graph-based Segmentation:
6. Edge-based Segmentation:
7. Histogram-based Segmentation:
8. PCA-based Dimensionality Reduction:
9. Hybrid Approaches:
10. Region Growing:

All these options can be suitable but, in this case, as we do not really know if there is any pattern or similar features in this data set it would be better to go for unsupervised learning. Moreover, the little size of this data set regarding the complexity of that task for this data set also guides us towards the unsupervised method. Furthermore, we set our trade-off on the complexity to implement and to explain and the estimated performance. Moreover, when we take a closer look to the dataset, we can see there was already research made onto it for the recall and precision:

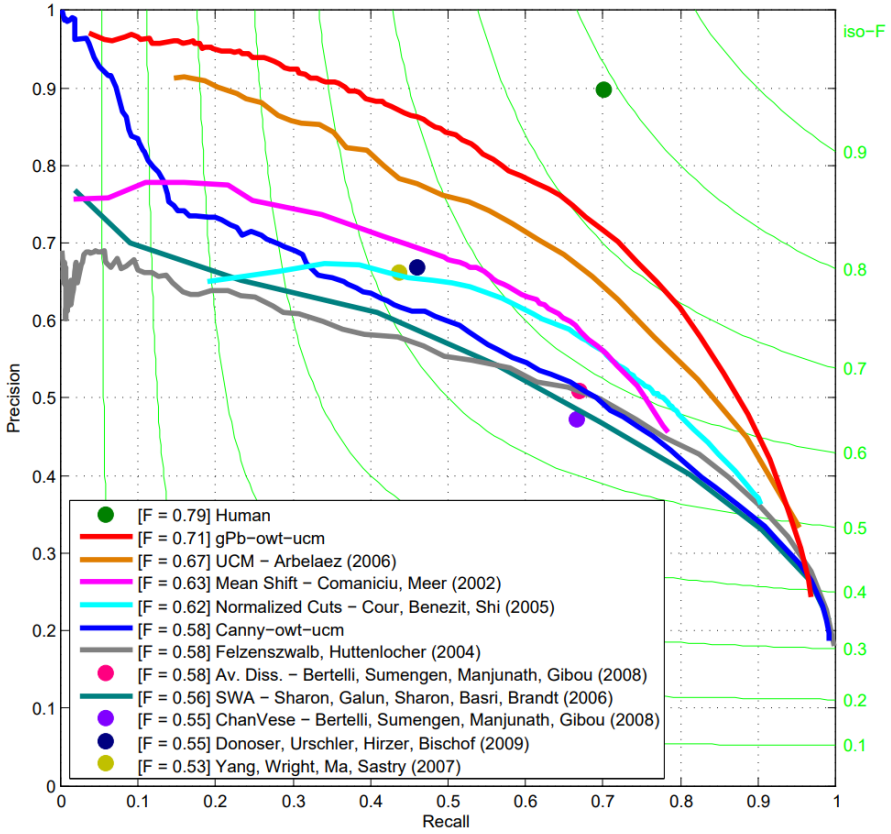


Figure 1: Evaluation of segmentation algorithms on the BSDS300 Benchmark. [2]

Therefore, we can see that among the different method used there is the mean shift algorithm one of the many algorithms we went through during the lectures (as mentioned [3]).

Mean Shift is a non-parametric clustering algorithm that is commonly used for object tracking, image segmentation, and mode seeking. It was introduced by Dorin Comaniciu and Peter Meer in their paper "Mean Shift: A Robust Approach Toward Feature Space Analysis" in 2002. So, it fits perfectly for the task here. As the Mean Shift clustering is to iteratively shift the data points towards the mode (peak) of the data density. The algorithm seeks to find the regions in the data space where the density of data points is the highest. [3]

Nonetheless, it does not reduce the dimensionality still the reduction we have to do here can be achieved out of the main algorithm. However, another possible issue is that we know that the mean shift algorithm has high computational cost but as nothing was mentioned about this in the subject, we can avoid that cons. Still, if we were needed to do the same thing for a real-time application the mean shift algorithm can do so. [4]

II. Implementation of the method

1. Implementation

To implement the method, I created a GitHub repository where I pushed all the data, I did so because I wanted to work on Google Colab at the library and at home.

```
!git clone https://github.com/AVDC-Sebastien/AIAS-Assignment
!unzip /content/AIAS-Assignment/Assignment_data.zip -d /content
!rm -r /content/AIAS-Assignment
```

```
!mv '/content/Assignment data' '/content/Assignment_data'
```
Figure 2: Importing the data set (had an issue with !wget giving me a json instead of a .zip)

Then, I used python to implement the mean shift method. To do so I imported the sklearn library where the mean shift method is implemented and optimized, and where I can find others handy tools to help me in implementing the method such as the estimate bandwidth.

```
bandwidth = estimate_bandwidth(flat_image, quantile=.06, n_samples=3000)
```
Figure 3: Estimating the bandwidth for one image

Because the bandwidth will make a change in the size of the different regions and the sensitivity noise and the computational efficiency. The latter leads us to bin seeding parameter which is a method used to speed up the convergence of the algorithm by reducing the number of data points considered during the mean shift computation. Here after testing on one image, we got from a computational time of 30 seconds to 45 minutes with bin seeding being the only difference.

Then, once the mean shift method as finished running, we need to make this data relevant. So, we made the same zones to be the same colours and try to make them in growing order so it would be better for the analysis part. Moreover, as we should reduce the dimensionality of the input we will output the same kind of data given by the ground truth and to do so we are just going to convert the image to grey scale then to int thus we will have a flat array of the size of the image reducing by 3 the dimensionality.

```
segmented_image = color.rgb2gray(result)
 colors = np.unique(segmented_image)
 for i,number in enumerate(colors):
   segmented_image[segmented_image==number]=i
```
Figure 4: Ordering the data.

After having the image segmented by our mean shift algorithm, we need to import the ground truth which are some MATLAB files.

```
def Get_mat_file(groundTruth_path):
  mat = scipy.io.loadmat(groundTruth_path)
  data = mat['groundTruth']
  all_data = []
  for n,segmented_image_gTruth in enumerate(data[0]):
    all_data.append(data[0,n]['Segmentation'][0,0])
  return all_data
```
Figure 5: importing the data from MATLAB to python.

Once every step is done, we only must tune the different parameters. First, we can try to modify the bandwidth to try obtaining better result.

2. Tuning of the parameters

But before tuning, we to define metrics to know if the tuning is good or can be better. So, let's define them (the utility of each metrics would be precise in the analysis):

- The precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- The recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- The F1 score:

$$\text{F1 Score} = \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}$$

- And the Jaccard index:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- And also the computational time

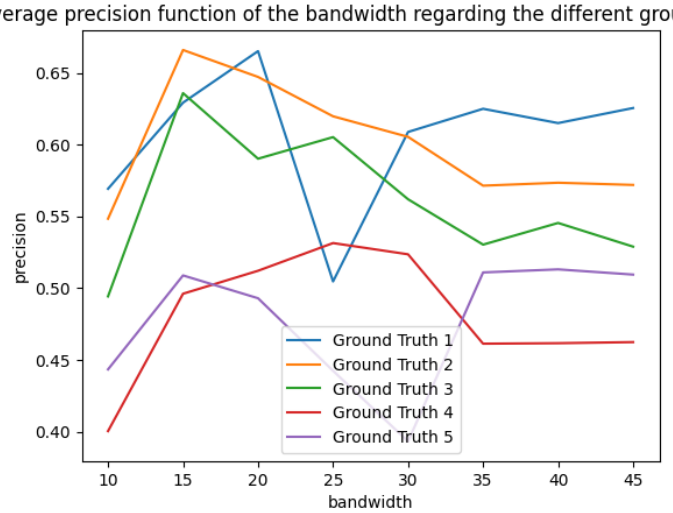
At first, I used the functions given by sklearn, but they were not suitable for this task. So, I needed to make one by myself:

```
def Get_metrics(true_img, pred_img):
  # Get the size of the image
  size = pred_img.shape

  # Initialize the vars
  metrics = []
  Temp_TP, Temp_FP, Temp_FN = 0,0,0

  # Change the image format
  true_img_flat = true_img.flatten()
  pred_img_flat = pred_img.flatten()

  # Get the unique values
  true_segments = np.unique(true_img_flat)
  pred_segments = np.unique(pred_img_flat)

  # Copy the flat image
  temp_pred_segment = pred_img_flat.copy()
  temp_true_segment = true_img_flat.copy()


  for segment in pred_segments:
    if segment == 0:
      temp_pred_segment[temp_pred_segment != 0] = 2
      temp_pred_segment[temp_pred_segment == segment] = 1
      temp_pred_segment[temp_pred_segment == 2] = 0
    else:
      temp_pred_segment[temp_pred_segment != segment] = 0
      temp_pred_segment[temp_pred_segment == segment] = 1

    selected_segment = 0
    for true_seg in true_segments:
      if true_seg == 0:
        temp_true_segment[temp_true_segment!=0]=2
        temp_true_segment[temp_true_segment==true_seg]=1
        temp_true_segment[temp_true_segment==2]=0
      else:
        temp_true_segment[temp_true_segment!=true_seg]=0
        temp_true_segment[temp_true_segment==true_seg]=1


      # Need to make temp copy so we do not overide it
      temp_1 = temp_pred_segment.copy()
      temp_true_1 = temp_true_segment.copy()
      temp_2 = temp_pred_segment.copy()
      temp_true_2 = temp_true_segment.copy()
      temp_3 = temp_pred_segment.copy()
      temp_true_3 = temp_true_segment.copy()
```

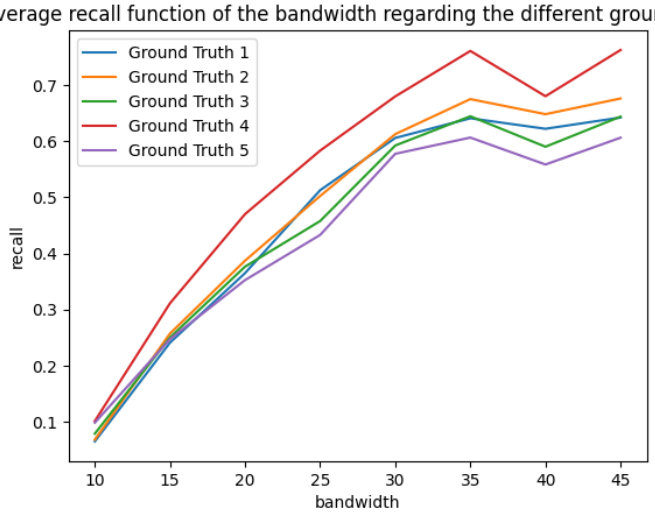
Figure 6: Get the metrics

Augmenting the bandwidth result in a less precise result but reduce the computing time. So, first let's try to figure out what is the bandwidth range I need to look for. I ran my code changing the bandwidth from 0 70 knowing that the bandwidth is usually a small number [5] and that the bandwidth computed by the estimate_bandwidth function is around 20. So, let's take a bit more than the double of it and go for 45. Let's have a look on other images. But first as this time it took 4 min, we might consider doing it with a random sampling big enough to represent the 500 images (because I wanted to include all of them). So, I took a sampling of 30 random image. To choose this amount I followed a rule of thumb I used to use in my past studies.

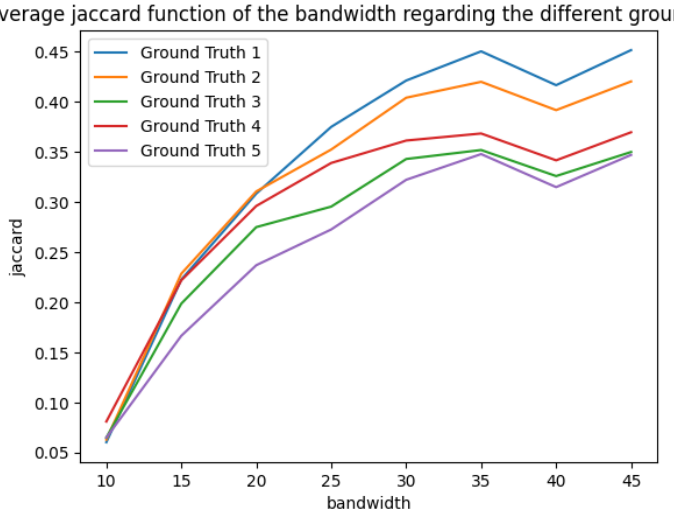
Average precision function of the bandwidth regarding the different ground Truth

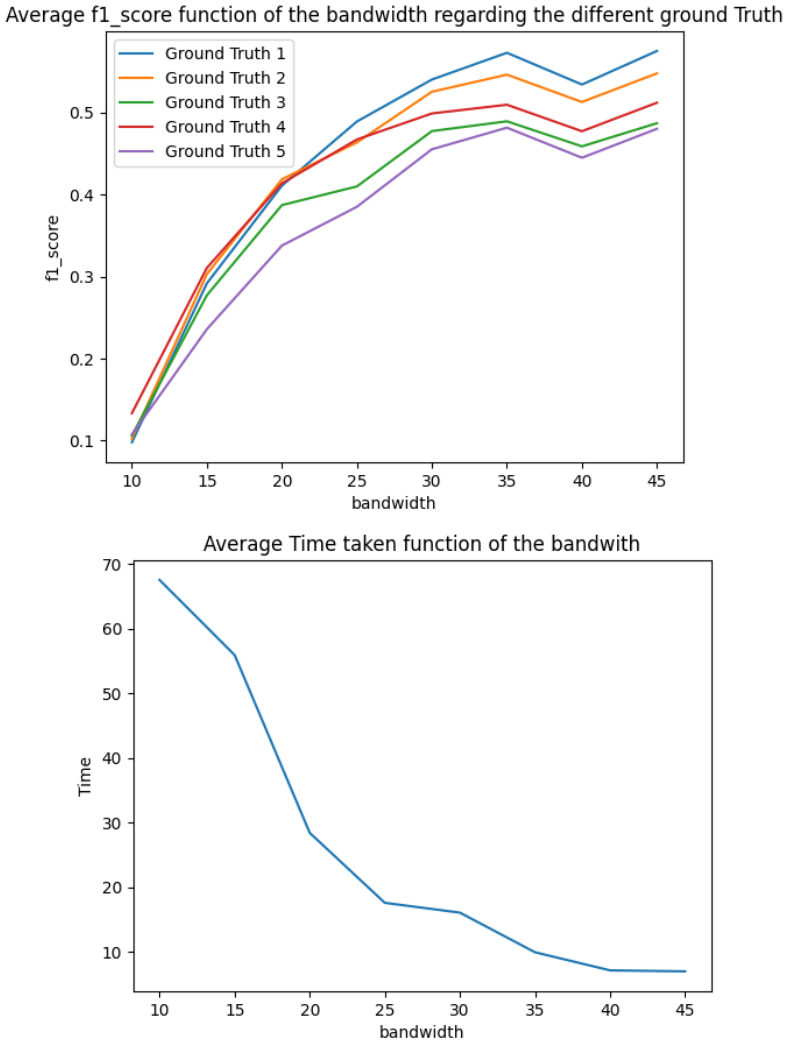


Average recall function of the bandwidth regarding the different ground Truth



Average jaccard function of the bandwidth regarding the different ground Truth





We can see here that when we increase the bandwidth, we lose precision but the recall, Jaccard index and the f1 score are better and the computational time decrease. But the increase in the average Jaccard index, f1 score and recall mean here that we lose data, so the data is more spread, and those metrics increases. As shown here:

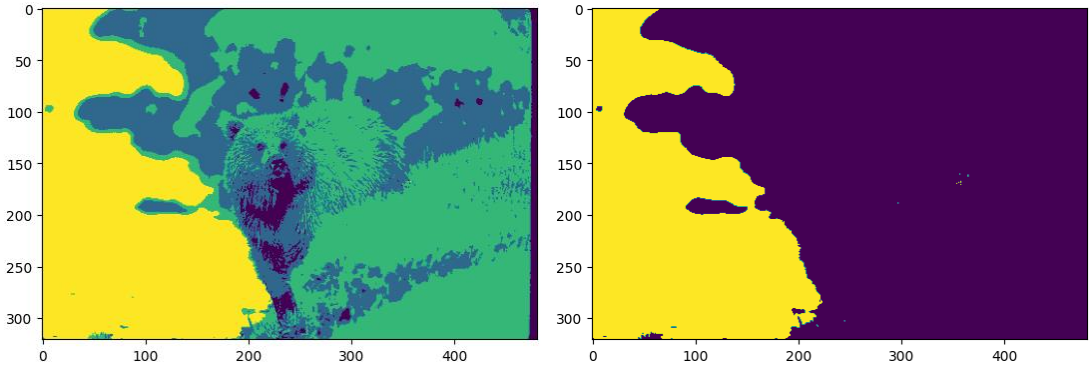


Figure 7: Segmented image 100099 with a bandwidth of 20 and 40

So, to choose the correct bandwidth we got to trade precision for computational time and so we will use the bandwidth of the estimate function from sklearn which always offer the best option because it stays on the value, we get but will adapt if there is a huge shift on the required bandwidth.

One of the issues we see here is the compute time. To reduce it we can change the min_bin_freq parameters to try to speed up the algorithm, so we just try it and try like we did before for the bandwidth. Still, here we have no hints on what is the range we should try so let's aim wide. We can work on a range

from 0 to 60 and do the same as before which is taking the average of the metrics defined before and the computing time for a sampling of 30 random images:

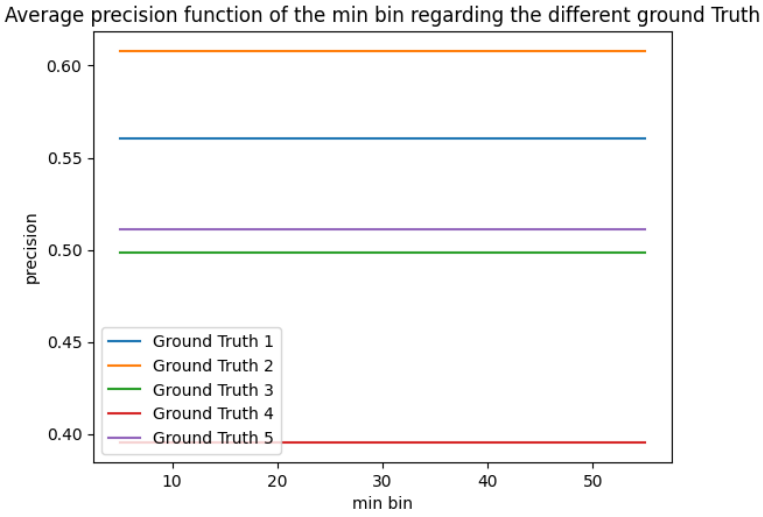


Figure 8: Precision

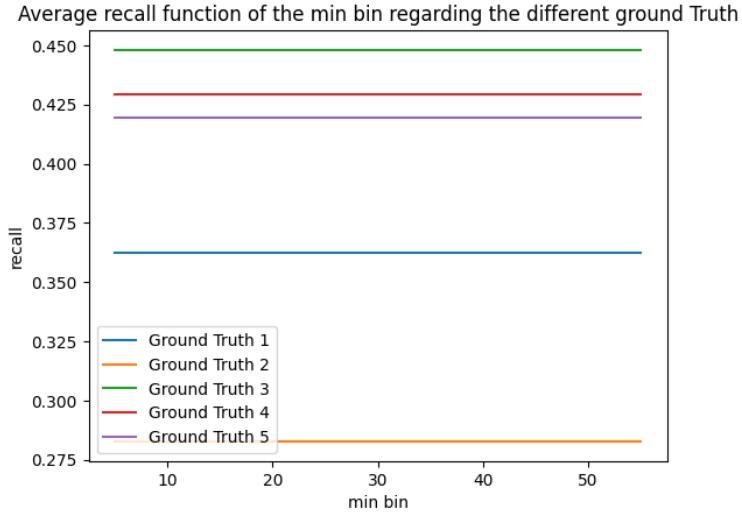


Figure 9: Recall

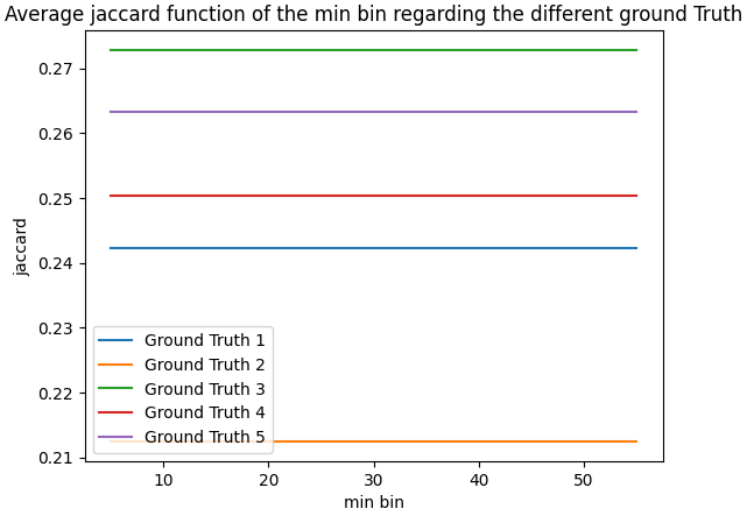


Figure 10: Jaccard index

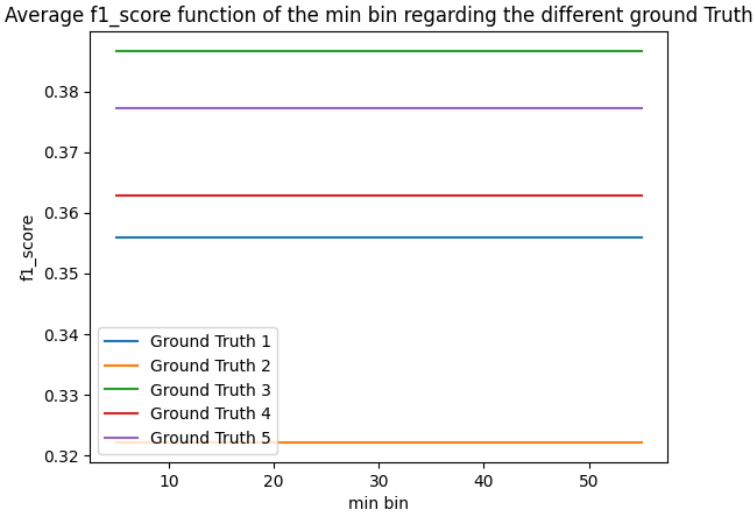


Figure 11: F1 score

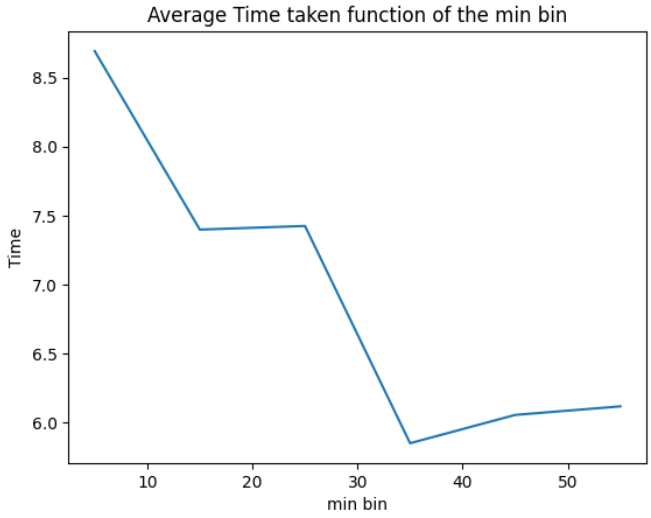
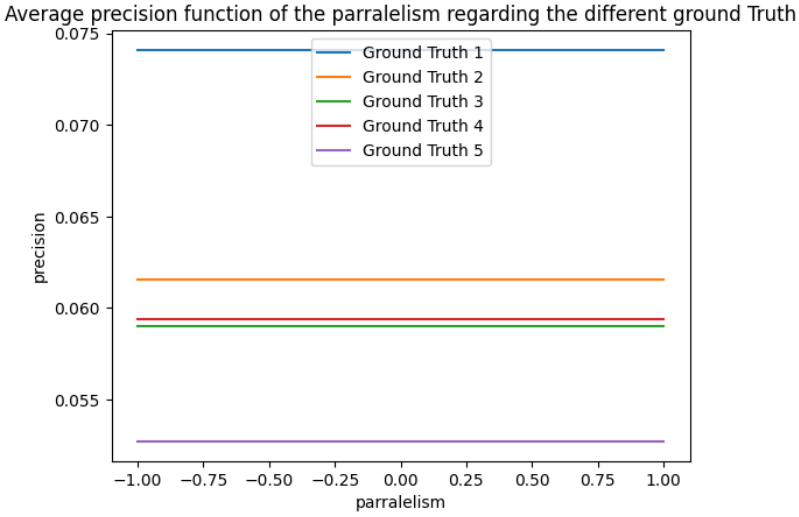
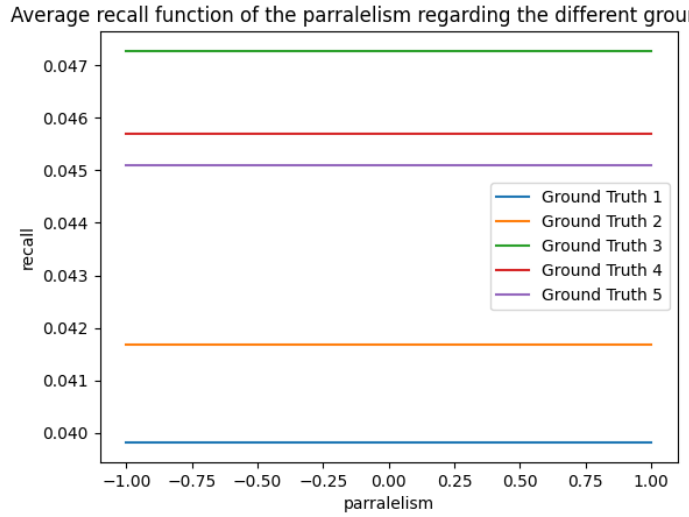


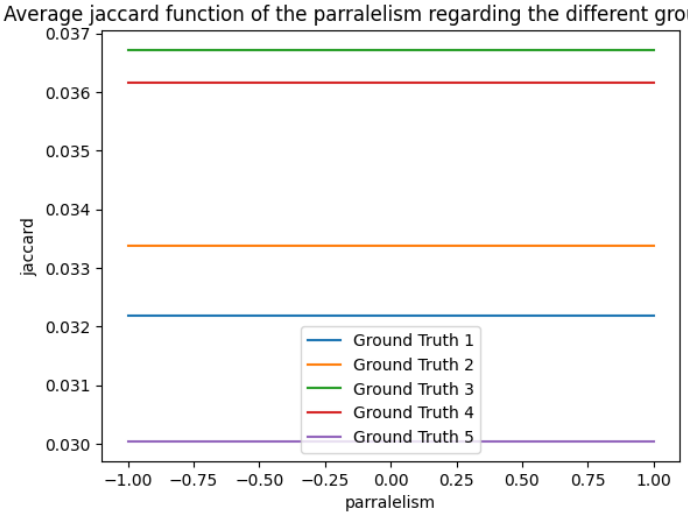
Figure 12: Computational time

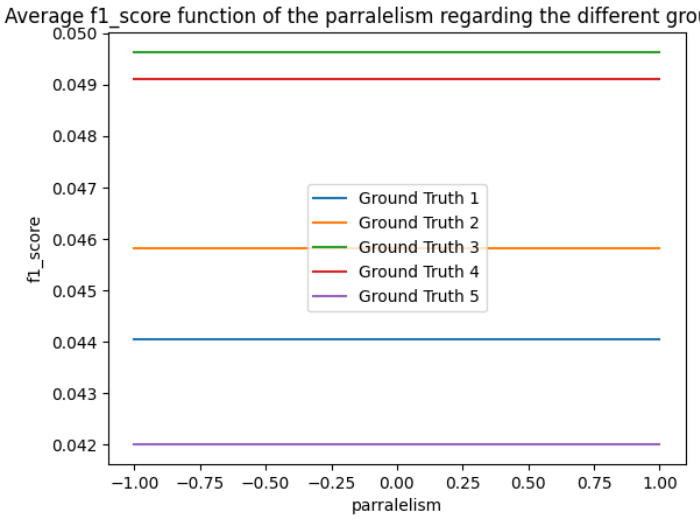
We can see that the min_bin_freq is not affecting any of the different metrics beside the computational time. Moreover, after a min bin frequency of 35 we cannot see a huge change in the computational time so we can stick with 35 for the next part.

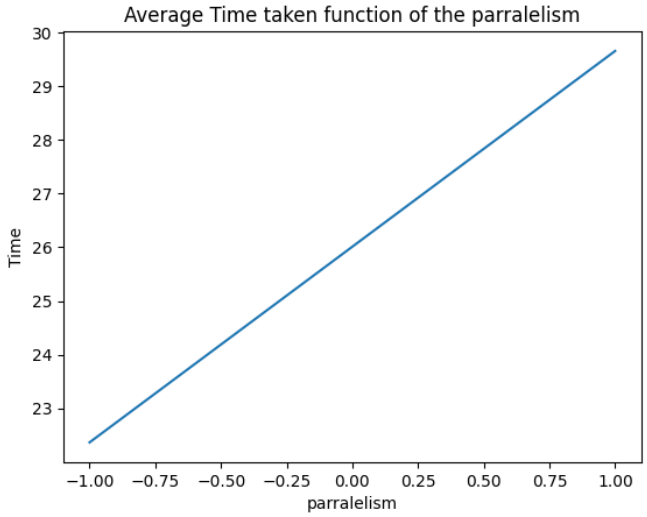
Another parameter which might impact the speed would be the n_jobs which is the number of jobs to use for the computation. So, let's do the same again:




Average recall function of the parralelism regarding the different ground Truth
recall
parralelism
Ground Truth 1
Ground Truth 2
Ground Truth 3
Ground Truth 4
Ground Truth 5


Average jaccard function of the parralelism regarding the different ground Truth
jaccard
parralelism
Ground Truth 1
Ground Truth 2
Ground Truth 3
Ground Truth 4
Ground Truth 5


Average f1_score function of the parralelism regarding the different ground Truth
f1_score
parralelism
Ground Truth 1
Ground Truth 2
Ground Truth 3
Ground Truth 4
Ground Truth 5



Here we can see that using the parallelism does not affect other metrics than the computational time so as it better to consider n_jobs = -1 we are going to use it for the next part.

Now that we have our method implemented and the different parameters, we need are tuned we can move on to analyse our implemented method.

III. Assessment of the performance of the implemented segmentation technique

To assess the performance of the implemented segmentation technique there is multiple choices. First as we can access to the ground truth, we can use it as a reference. Then, even though we do have access to the ground truth some other interesting indicator does not need the ground truth.

It is also important to mention that analysing the output data should always be regarding the objectives. Here, we apply a segmentation routine for a basic task of finding areas in the image, consisting of similar pixels that can be used as a pre-processing stage for the object detection and tracking. One of the requirements here is to preserve semantic context of the image as much as possible, while reducing dimensionality of the input to the following stages of processing.

1. With Ground Truth:

1.1. Quantitative Evaluation:

So, we are going to look at all the metrics discussed before. To do so I let the code run over the night to have all the images segmented and all the metrics extracted, and this is what I got after 6h of run:

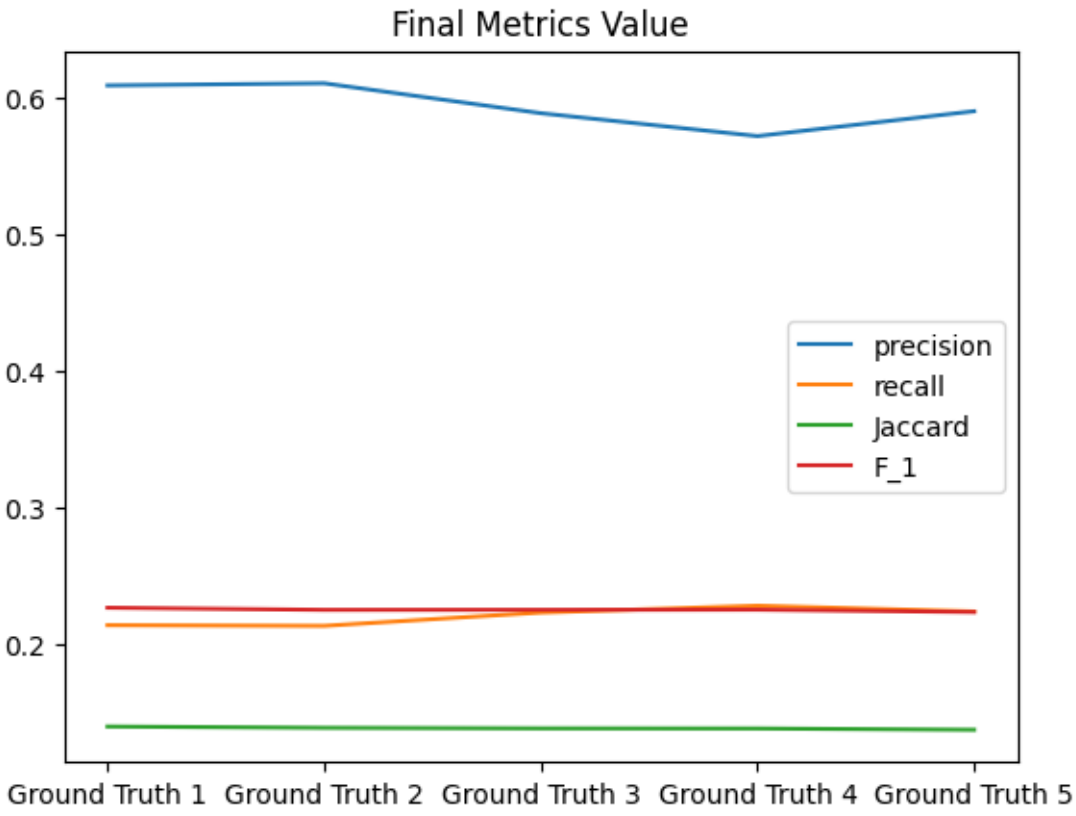
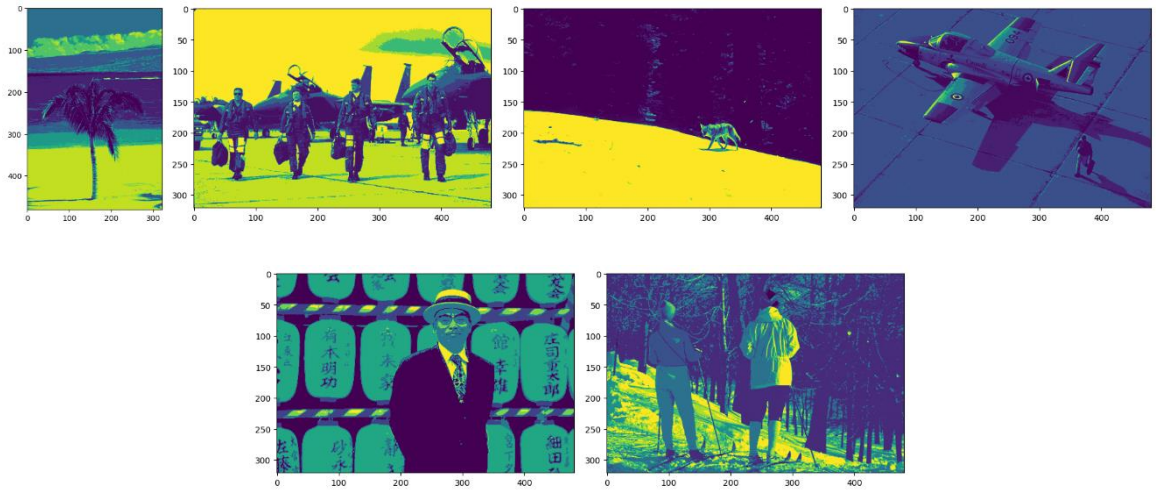


Figure 13: Final 500 images metrics value

So, we can see that the overall precision of around 0.6 indicate that when the algorithm predicts a pixel as part of the object, it is likely to be correct. Which is good because it is what we want! Then we have an overall recall of 0.22 which means that the algorithm is not pretty bad at finding all the relevant pixels belonging to the object. So, it makes the precision score less valuable, because even though the pixels found are found in the right object almost half of them are not found. Then the Jaccard index of an overall of almost 0.17 too also indicates not a bad overlap between the predicted and true positive regions. Finally, the f1 score illustrate what we said just before on the precision and recall as it shows the balance between precision and recall. With an overall score of 0.23 we can see that those two values are not well balanced.

1.2. Visual Inspection:

We could also try visual inspection, superimposing the ground truth to our results as an overlay as a difference. Or just plot different images and see that they are as segmented as they should be.



We can see even if there is some difference with the ground truth, the main “objects” are clearly identified.

1.3. Error Analysis:

Here we are going to look at some case where the results were totally different from the ground truth to see what happened. Let’s look at this picture:

Figure 14: Reference image 69022

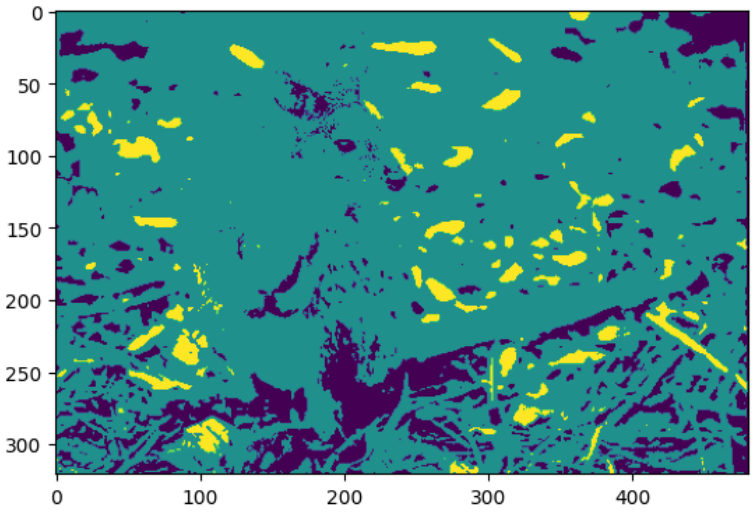


Figure 15: Segmented Image

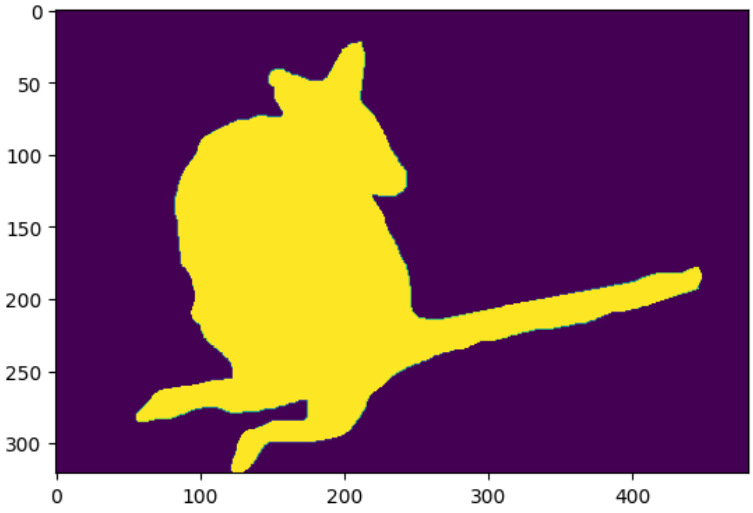


Figure 16: Ground Truth

We can see here that the segmented image is bad. Due to the image in the light the mean shift algorithm can really make any segments. To fix this issue we could try to always have a proper lighted environment. And if we cannot do so we would need to change the segmentation method.

2. Without Ground Truth:

Here as we do not know much about the data set, we can really set application specific metrics. One metrics that is useful if we do not possess the ground truth would be just looking at each picture and visually inspect the segmented output and compare it with the original image. Looking for meaningful segmentations and how well it captures the structures in the image. But as we do have the ground truth it does not make sense doing so. Moreover, the metrics would not have been precise compared to what we did before. Still, we can say that by looking at images we can distinguish even better than before the different object and outlines, for instance:

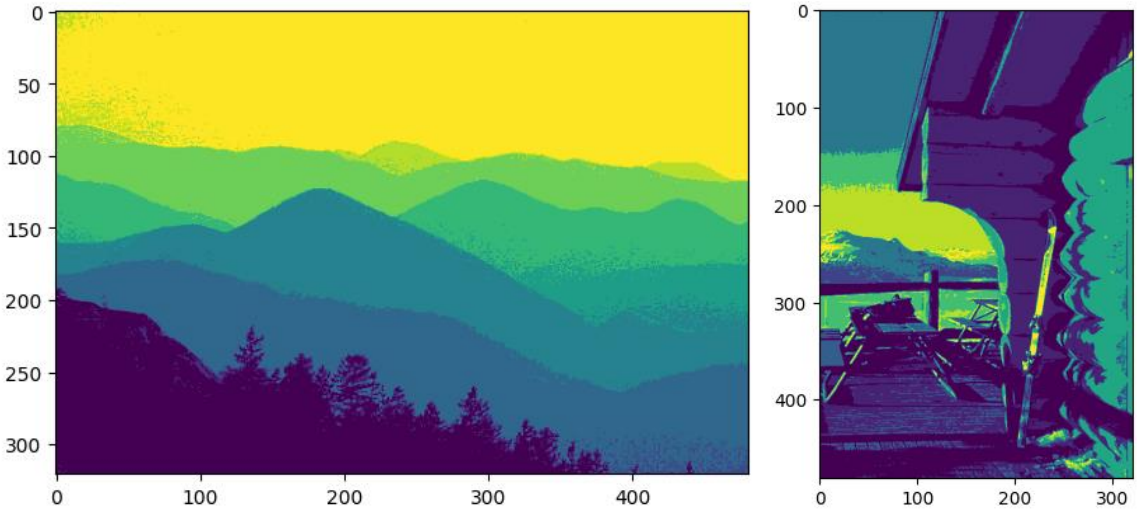


Figure 17: Segmented images from the 500 images

Still, as said before this does not mean much.

2.1. Runtime and Resource Usage:

Even though the runtime or resource usage it is not specified in the task, it is still important to know for further work, in case of an object detection in real time we need to have a fast algorithm and if it is an onboard device, it should require the least amount of resource possible.

In average the mean shift method is using only 3 to 4 GB of RAM and take and average 20 seconds to segment an image. So, I cannot be used on a on board device like this it needs to be changed to fit another task. For instance, the raspberry pi 3 as only 1GB of RAM so it will not be able to run. [6]

2.2. Compactness

As the data base is composed of image, we need to segment into different regions trying to know if each segment is made to be together or in another if the variance of each segment is low or not, so we need to get the compactness of each segmented image. Which give us this:

```
def get_variance(img_path,segmented_image_path):
  image = cv2.imread(img_path)
  flat_image = np.float32(image.reshape((-1,3)))

  segmented_image = np.loadtxt(segmented_image_path).astype(int)
  segments = np.unique(segmented_image)

  variance = []
  for values in segments:
    temp = []
    for i in range(len(segmented_image)):
      for j in range(len(segmented_image[0])):
        if segmented_image[i][j] == values:

          temp.append(flat_image[i])
    mean_rgb = np.mean(temp, axis=0)
    print(len(temp))
    squared_diff = (temp - mean_rgb) ** 2
    print(np.sum(squared_diff, axis=0))
    variance_rgb = np.sum(squared_diff, axis=0) / len(temp)
    variance.append(variance_rgb)
  return np.mean(np.mean(variance,axis=0))
```

Figure 18: Code to get the variance for one image

For instance, in this picture

Figure 19: Image 100099

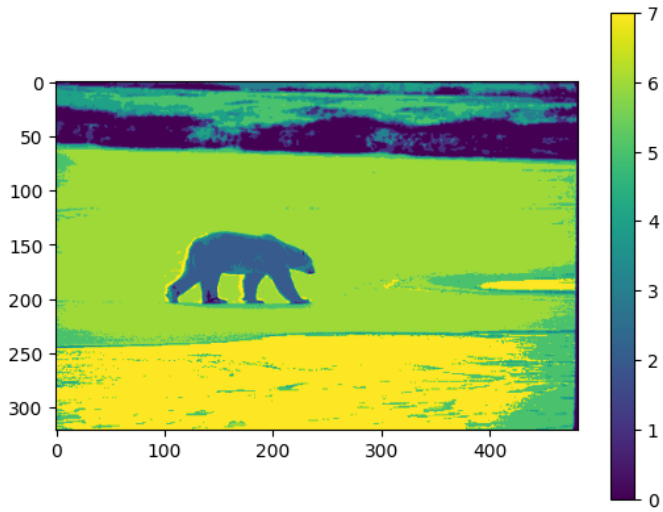


Figure 20: Mean Shift segmented image 100099

We can obtain:

```
[[185.2856 , 132.79655,  74.48746],
 [160.48964 , 119.09381 ,  64.079636],
 [21.217283, 10.773635,  6.866873],
 [444.59055, 330.79218, 223.34488],
 [473.45117, 333.43124, 221.36775],
 [554.0058 , 377.78754, 277.19852],
 [196.07034223, 136.14072889, 135.71058984],
 [69.02823 , 38.503113, 27.005344]]
```

Figure 21: Variance of the RGB image for different segments

So, we can see that the variance is quite high but for the bear the third line the variance is lower. Still, we cannot really generalize this method as we should be looking at every single variance of each segment because taking the average would make not that much sense. Because the algorithm could segment well the object, we want but not other things we are not interested in. However, if we are interested in as much every single segment, we can consider taking the average of the average and it would give us 234.4456 as an average for the 200 images. We cannot really tell a lot with this value due to the unknown final goal.

3. Comparison with other know techniques.

We can compare the implemented method to the other method shown on the paper. We can see on the Figure 1: Evaluation of segmentation algorithms on the BSDS300 Benchmark. that with a F1 the same score as we got of 0.35 that they got with the mean shift algorithm a precision of 0.75 and a recall of 0.2. So, they are more precise than we are with kind of the same values for the other metrics. Furthermore, the main difference with other techniques is computational time as discuss before the Mean Shift algorithm as high computational O(tn2) (with t is the time for processing each data point and n is the number of data points) [7] needs compared to other techniques that run faster than the Mean Shift.

4. Issues faced.

As shown before the major issue I faced was the computational time. I could not make any mistake because each part of the code could take hours to complete so it was quite challenging. Another issue would be the using MATLAB at first because I am more used to do AI in python so quickly moved to python. Furthermore, we had to do a lot of code that we had to test before running due to the long running time so it took me more time than expected and I had to wait sometimes over night to have my code complete but got disconnected during the night so had to start it again and monitor it during the day, I had consider changing methods multiples time but did not risk.

Conclusion

To conclude, we discussed and selected one of the multiples techniques for segmentation and chose the mean shift algorithm based upon the complexity of implementation the difficulty to understand the algorithm, the results it would give, and if we saw it during the lectures. Then, applied it to find areas in an image consisting of similar pixels that can be used as a pre-processing stage for an object detection and tracking while preserving the semantic context of the image as much as possible while reducing dimensionality of the input image for the following stages of processing. And after a long tuning of the different parameters, we could finally assess the quality of the results. And so, with and without the ground truth but revealing a lot flaws in the final model. For further use mean shift does not seem to be the number option regarding the results I obtain compared to what they got on [2]. And, we can see that I used all the images here https://github.com/AVDC-Sebastien/AIAS-Assignment in the data.zip

References

[1] A. Acharya, “Guide to Image Segmentation in Computer Vision: Best Practices,” Encord, 7 November 2022. [Online]. Available: https://encord.com/blog/image-segmentation-for-computer-vision-best-practice-guide/. [Accessed 25 January 2024].

[2] P. Arbelaez, C. Fowlkes, M. Maire and J. Malik, “Contour Detection and Hierarchical Image Segmentation,” 2010. [Online]. Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/papers/amfm_pami2010.pdf. [Accessed 20 January 2024].

[3] D. I. Petrunin. [Online]. Available: https://canvas.cranfield.ac.uk/courses/27433/pages/assignment-materials?module_item_id=536715. [Accessed 20 January 2024].

[4] T. S. a. R. K. N. Oshima, “Real Time Mean Shift Tracking using Optical Flow Distribution,” 2006 . [Online]. Available: https://ieeexplore.ieee.org/document/4108272. [Accessed 20 January 2024].

[5] sklearn, “sklearn.cluster.MeanShift,” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.MeanShift.html. [Accessed 31 January 2024].

[6] PiHut. [Online]. Available: https://thepihut.com/products/raspberry-pi-3-model-b-plus#:~:text=Awesome%20Extras&text=The%20Raspberry%20Pi%203%20Model,significantly%20faster%20300Mbit%2Fs%20ethernet..

[7] Y. C. L. Z. T. T. &. F. Fang Huang, “Implementation of the parallel mean shift-based image segmentation algorithm on a GPU cluster,” [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/17538947.2018.1432709#:~:text=The%20time%20complexity%20of%20the,points%20to%20be%20processed%20increases..