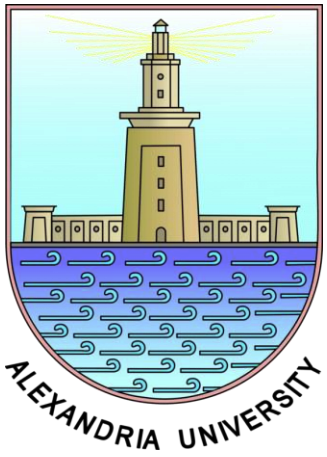
Alexandria University- Faculty of Engineering



Computer and Communication Engineering Department

Image Segmentation Assignment

Pattern Recognition Course

Eyad Salama 7128

Nour Hesham 7150

Abstract

Image segmentation is critical in digital image processing. Image decomposition is the process of separating an image into its component parts, which are referred to as image objects. This helps to reduce the complexity of an image so that it can be examined more thoroughly. It is intended that the final product will be a picture that is free of any noise or distortion. In addition, it is utilised to denote the location of objects as well as their boundaries. In the past, numerous segmentation strategies that took a variety of approaches and differed only slightly from one another were presented. This study conducts an in-depth investigation into a number of different methods of segmentation, including edge-based segmentation, region-based segmentation, clustering, thresholding, and soft computing-based segmentation.

Table of Contents

Table of Figures

Chapter 1: Introduction

1.1 About The Dataset Used

"When you can measure what you are speaking about and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of the meagre and unsatisfactory kind." **-Lord Kelvin**

The purpose of the benchmark is to produce a score for the boundaries of an algorithm for two reasons:

(1) So that different algorithms can be compared to one another

(2) So that progress toward human-level performance can be tracked over time. The score for an algorithm's boundaries is the goal of the benchmark. We have invested a significant amount of time into the development of a relevant border detection benchmark, which we will discuss in more detail in the following paragraphs. For further information, please refer to our publications presented at NIPS 2002 and PAMI.

It is important to keep in mind that the approach that we have decided to go with may be utilized with any boundary dataset; it is not limited to simply our dataset of human segmented natural photos.

The setup is as described below. The photos of humans that have been segmented offer the ground truth limits. Any border that was marked by a human subject is considered to be valid by our organization. The accumulation of these human-marked boundaries is what defines the ground truth since each image has been divided into various segments by a variety of subjects, and we have access to all of these segments. After that, the result of some algorithm being applied to an image is shown to us. Let us suppose that the output is a soft boundary map with boundaries that are one pixel wide and values ranging from zero to one, where higher values indicate a greater degree of certainty in the existence of a boundary. Our job right now is to figure out how well this soft boundary map comes to matching the ground truth boundaries.

Historically, one would "binarize" the border map by picking a certain threshold. This was done. When thresholding a border map, there are two difficulties that might arise: (1) The ideal threshold varies depending on the application, and we'd like the benchmark to be applicable to a variety of use cases; on the other hand, we think that setting a threshold on a low-level feature like boundaries is probably not the best idea for most applications because doing so eliminates a lot of information. Our standard is based on a boundary map that does not include thresholds for any of these reasons.

However, in order to compare the boundary map to the real-world borders, it is necessary for us to threshold the map first, but we do this at a number of various levels, such as 30. We build a precision-recall curve for the method by computing two values, namely precision and recall, at each level of the algorithm. These two numbers are precision and recall. Precision and recall are axes of ROC curves that are comparable to one another but distinct from one another. The term "precision" refers to the likelihood that a pixel created by a machine represents an actual border pixel. The chance that an actual boundary pixel is identified is referred to as recall. These axes are rational and obvious in their application. The amount of noise that is present in the detector's output is a measurement of its precision. The amount of the ground truth that can be recalled is referred to as the recall percentage. As the detector threshold is adjusted, the curve illustrates the intrinsic trade-off that occurs between these two values, namely the trade-off that occurs between false positives and false negatives.

It is nevertheless preferable to condense the performance of an algorithm into a single number, despite the fact that the precision-recall curve for an algorithm represents a rich description of its performance. This is something that can be done in a meaningful way for algorithms whose curves are parallel to one another and do not overlap with one another. When two precision-recall curves do not meet, the one that is further away from the origin tends to have more influence than the other. A measure of this distance is provided by the summary statistic that we make use of. The F-measure is the one to choose since it represents the harmonic mean of accuracy and recall. On the precision-recall curve, the F-measure is specified in all of the different spots. The summary statistic that we offer for an algorithm is the F-measure value that is highest at any point along the precision-recall curve.

Why do we plot precision-recall curves instead of receiver operating characteristic (ROC) curves?

Both precision-recall curves and receiver operating characteristic (ROC) curves exhibit, qualitatively speaking, the same trade-off between false positives and false negatives as does the precision-recall curve. Nevertheless, ROC curves are not the right tool to use when attempting to quantify border detection. The terms "fallout" and "recall" are used as the axes of a ROC curve. Hit rate and recall are the same thing. Recall is simply another name for hit rate. The fallout, also known as the false alarm rate, is the risk that an accurate negative result was misinterpreted as a positive result. Due to the fact that it is not independent of the resolution of the picture, this is not a valid number to use for a border detector. If we make the pixels' radius n times smaller so that the total number of pixels increases at the rate of n squared, then the number of true negatives will increase at a rate that is quadratic in n while the number of true positives will only increase at a rate that is linear in n. Due to the fact that borders are one-dimensional objects, the number of false positives will almost certainly increase linearly with the value of n. As a result, the fallout will decrease by a factor of 1/n. Due to the fact that precision is normalized based on the number of true positives rather than the number of true negatives, this issue is not present in the measure of precision.

1.2 Overview

A digital image is made up of various components that must be "analyzed," to use a technical term, and the "analysis" performed on such components can reveal a wealth of hidden information. This data can assist us in addressing a wide range of business issues, which is one of the many end goals associated with image processing.

Image segmentation is the process of partitioning a digital image into various subgroups (of pixels) known as Image Objects, which can reduce the complexity of the image and thus make image analysis easier.

We use image segmentation algorithms to split and group a specific set of pixels from an image. By doing so, we are actually assigning labels to pixels, and pixels with the same label fall into a category where they share something.

We can use these labels to specify boundaries, draw lines, and separate the most important objects in an image from the rest of the less important ones. In the following example, we try to get the major components, e.g., chair, table, etc., from a main image on the left, and thus all the chairs are colored uniformly. We discovered instances that talk about individual objects in the following tab, so all of the chairs have assorted colors.

This is how various image segmentation methods work in varying degrees of complexity and yield varying levels of output.

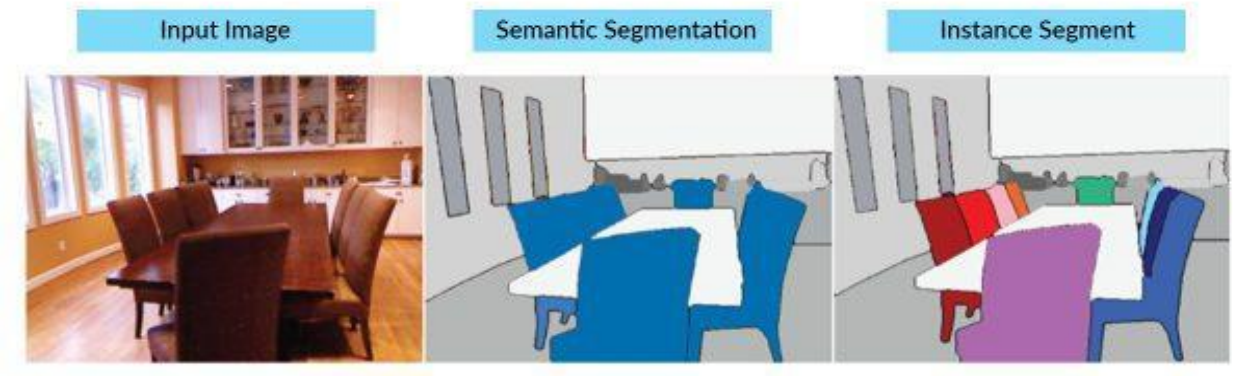


Figure 1: Image Segmentation Example

1.3 The Need for Image Segmentation

Many business problems have been addressed by the concept of partitioning, dividing, fetching, labeling, and then using that information to train various ML models. Let us try to grasp what difficulties Image Segmentation solves in this part.

A facial recognition system uses image segmentation to identify an employee and automatically mark their attendance. In the medical industry, image segmentation is used for efficient and faster diagnosis, detecting diseases, tumors, and cell and tissue patterns from various medical imagery generated by radiography, MRI, endoscopy, thermography, ultrasonography, and so on.

Satellite images are processed to identify patterns, objects, geographical contours, soil information, and so on, which can then be used in agriculture, mining, geo-sensing, and so on. Image segmentation has several applications in robotics, such as RPA and self-driving automobiles. Security photos can be analyzed to identify dangerous items, threats, individuals, and situations. For the procedure, image segmentation implementations in Python, MATLAB, and other languages are widely used.

A remarkably interesting case I came across was a television show about a certain food processing factory, where tomatoes on a fast-moving conveyer belt were being inspected by a computer. It was taking high-speed images with a suitable camera and sending them to a suction robot, which was picking up rotten, unripe, or otherwise damaged tomatoes while leaving the good ones alone.

This is a simple, but important, application of Image Classification in which the algorithm was able to capture only the necessary components from an image, and those pixels were later classified as good, bad, and ugly by the system. A simple system was having a massive impact on that company, eliminating human effort, human error, and increasing efficiency.

1.4 Types of Image Segmentation

When attempting to get a bird's eye view of the Image Segmentation tasks, one is able to observe a critical process that occurs here - object identification. Everything is based on object detection, from simple to complex application areas.

And, as previously stated, detection is made possible because image segmentation algorithms attempt to collect similar pixels together and separate out dissimilar pixels. This is accomplished through the use of two approaches based on image properties:

1.4.1 Similarity Detection

This basic method detects comparable pixels in a picture using a threshold, region expanding, region spreading, and region merging. Clustering and classification, both of which identify similarity based on a pre-defined (known) collection of characteristics, rely on this method of similarity detection on an unknown set of features.

1.4.2 Discontinuity Detection

This is the polar opposite of the similarity detection strategy, in which the program looks for discontinuity. This technique is followed by image segmentation algorithms such as edge detection, point detection, and line detection, which detect edges based on various metrics of discontinuity such as intensity, etc.

Figure 2: Edge vs Region Methods

1.5 Types of Techniques

There are various techniques used in the design of Image Segmentation Algorithms based on the two approaches. These techniques are used depending on the type of image that needs to be processed and analyzed, and they are divided into three broad categories, as shown below:

1.5.1 Techniques for Structural Segmentation

These algorithms require us to first understand the structural information of the image under scanner. Pixels, pixel density, distributions, histograms, color distribution, and so on are examples of this. Second, we need structural information about the region that we are about to retrieve from the image - this section is concerned with identifying our target area, which is highly specific to the business problem that we are attempting to solve. In these algorithms, a similarity-based approach will be used.

1.5.2 Techniques for Stochastic Segmentation

the primary information required for these algorithms is to know the discrete pixel values of the entire image, rather than pointing out the structure of the required portion of the image. This is useful when dealing with a large group of images and there is a high degree of uncertainty about the required object within an object. This method is used by ANN and Machine Learning algorithms such as k-means.

1.5.3 Hybrid Methodologies

As the name implies, these image segmentation algorithms use a combination of structural and stochastic methods, i.e., they use both the structural information of a region and the image's discrete pixel information.

Chapter 2: Assignment Progress

2.1 Problem Statement

We plan to do picture segmentation. Image segmentation refers to the ability to group similar pixels together and assign the same label to these grouped pixels. The grouping problem is also known as the clustering problem. We would like to investigate the use of K-means on the Berkeley Segmentation Benchmark. The actions required to complete the assignment are outlined below.

2.2 Downloading The Dataset

We first downloaded the dataset from the given link which was the official link of the BSR_BSD500 Berkeley dataset.

It was highlighted that the dataset has 500 images, 200 as a test set and we are going to report our results on the first 50 images only.

2.3 Visualizing The Original Image and The Associated Ground Truth

2.3.1 Writing our own function that reads an image and display the image with its associated ground truth segmentation(s).

```
1.  def _visualize_random():
2.      n = random.randint(0, 199)
3.  #     n = 0
4.      print(n)
5.      gt_cnt = len(dataset['gt'][n][1])
6.      fig,axes = plt.subplots(1, gt_cnt+2) # +1 for original image, +1 for edges
7.      image = dataset["imgs"][n]
8.      if dataset["gt"][n][0].shape[0] != 321:
9.          image = cv2.rotate(image, cv2.ROTATE_90_COUNTERCLOCKWISE)
10.     axes[0].imshow(image)
11.     axes[1].imshow(dataset['gt'][n][0])
12.     for i in range(0,gt_cnt):
13.         axes[i + 2].imshow(dataset['gt'][n][1][i])
14.
15.     plt.show()
16.
```

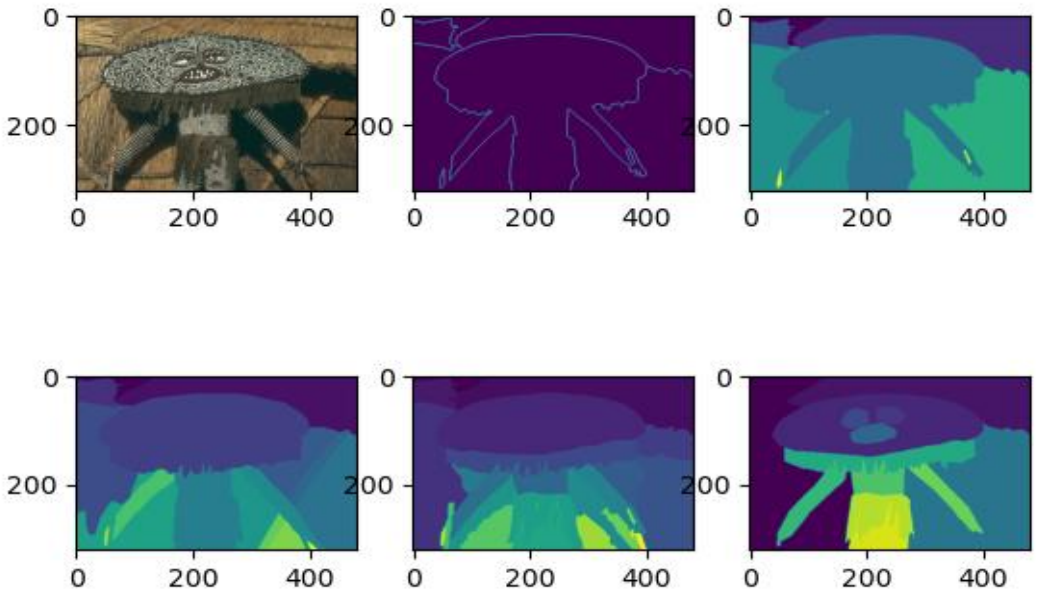


Figure 3: Image and Associated GT

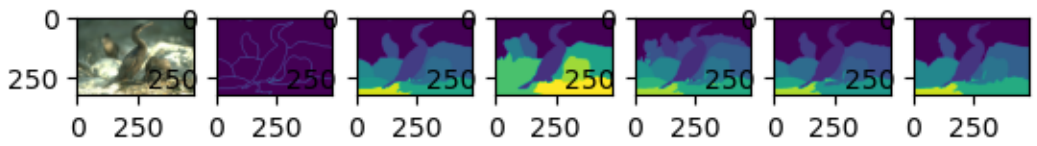


Figure 4: Image and Associated GT 1

2.4 Segmentation using K-means

2.4.1 K-means Output

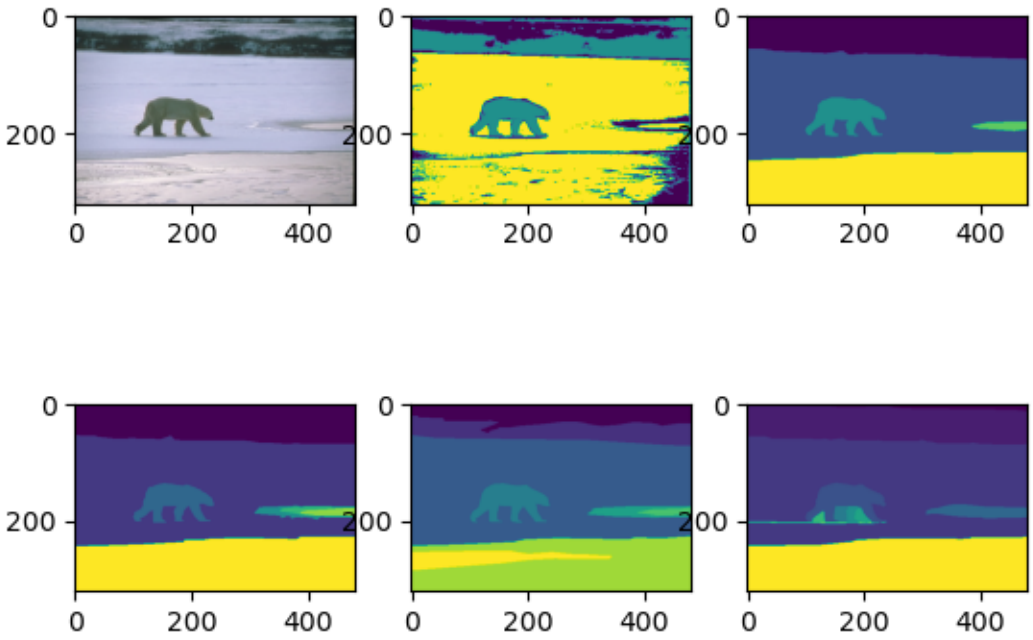


Figure 5: K-means Output

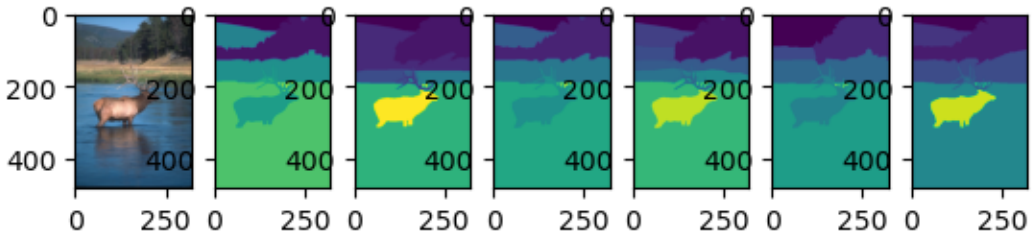


Figure 6: K-means Output 2

2.4.2 Evaluation of Segmentation Results

The following is the Conditional Entropy of Clustering C.

$$H(\mathcal{T}|\mathcal{C}) = \sum_{i=1}^{r} \frac{n_i}{n} H(\mathcal{T}|C_i)$$

Figure 7: Conditional Entropy Rule

Conditional entropy with respect to cluster(i):

$$H(\mathcal{T}|C_i) = -\sum_{j=1}^{k} \left(\frac{n_{ij}}{n_i}\right) \log\left(\frac{n_{ij}}{n_i}\right)$$

Figure 8: Conditional Entropy Ci

Before measuring the conditional entropy and the F-measure of each segmentation we used a max matching algorithm so that the ground truth labels, and the K-means clustering have the same numerical value.

2.4.3 Displaying Results

For each K we took the maximum of conditional entropy so that we can overview the bad examples or the bad segmentation that occurred.

2.4.3.1 Best Segmentation Scores

As we can see from the original picture, the overall RGB values in the original photographs are difficult to discern by the computer in any case, therefore it was believed from the start that segmentation of this picture would be difficult due to the lack of variation in the colours in the picture.

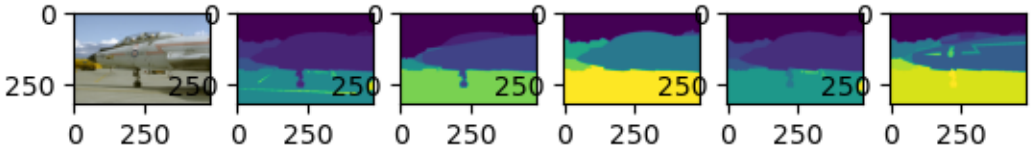


Figure 9: Best Entropy 1

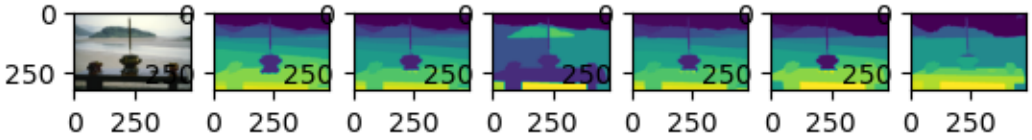


Figure 10: Best Entropy 2

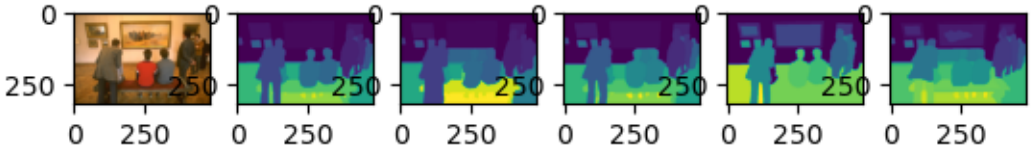


Figure 11: Best Entropy 3

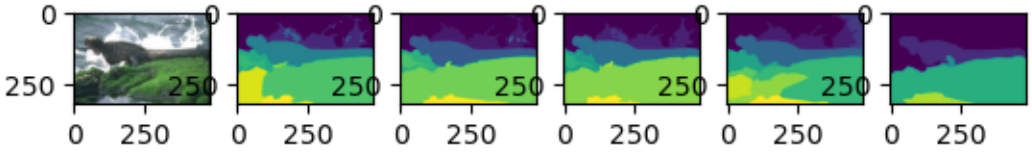


Figure 12: Best Entropy 4

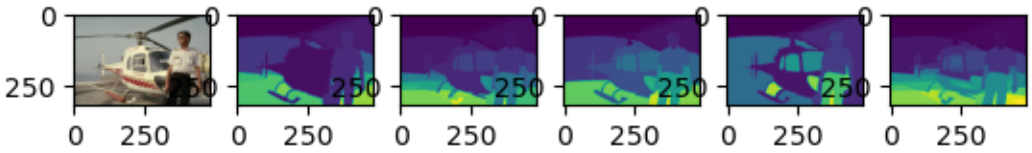


Figure 13: Best Entropy 5

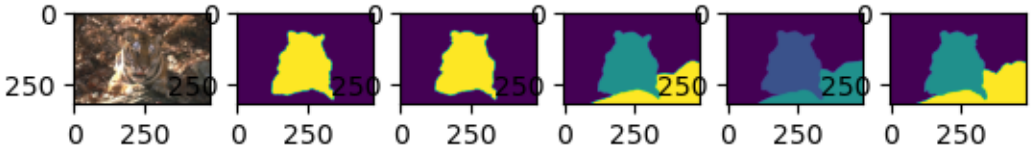


Figure 14: Best fscore 1

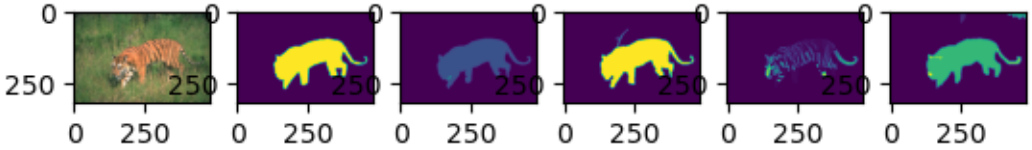


Figure 15: Best fscore 2

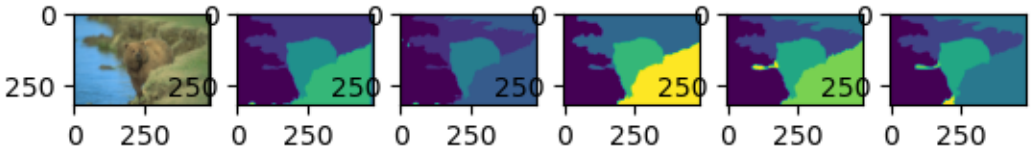


Figure 16: Best fscore 3

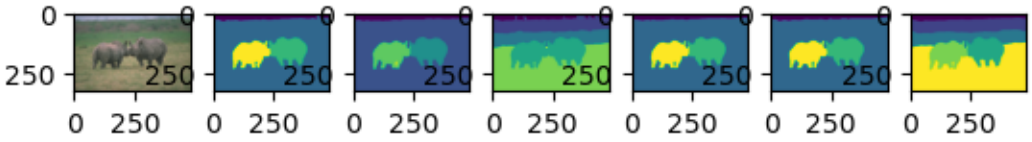


Figure 17: Best fscore 4

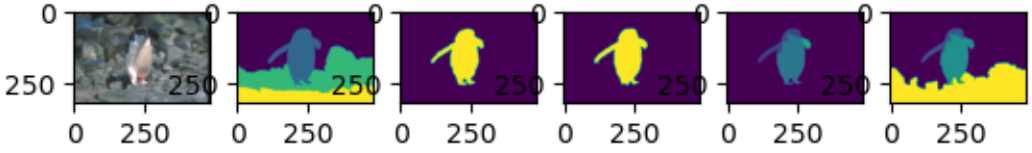


Figure 18: Best fscore 5

2.4.3.2 Worst Segmentations Scores

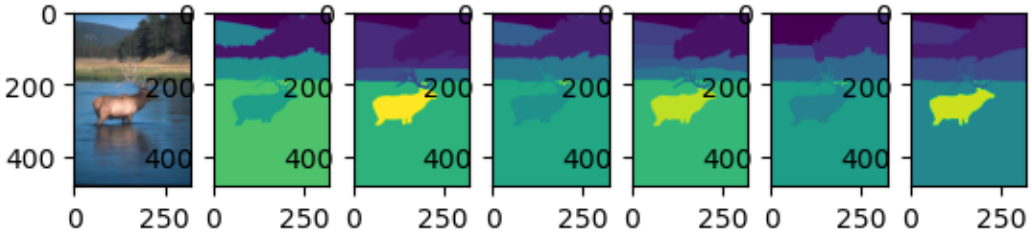


Figure 19:Bad Entropy 1

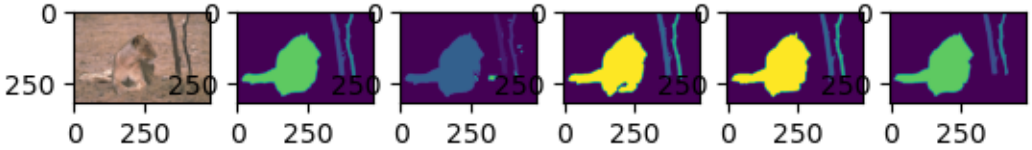


Figure 20: Bad Entropy 2

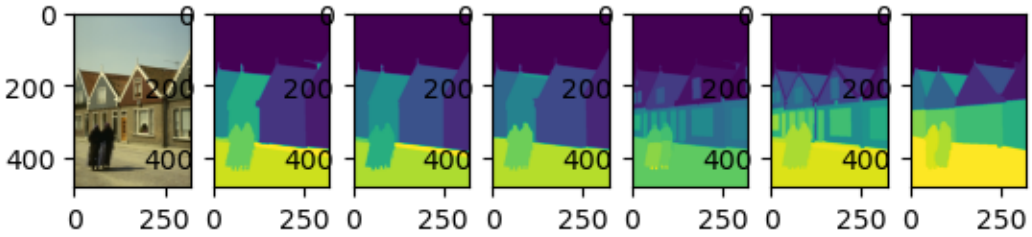


Figure 21: Bad Entropy 3

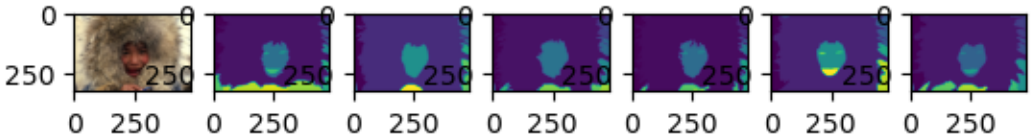


Figure 22: Bad Entropy 4

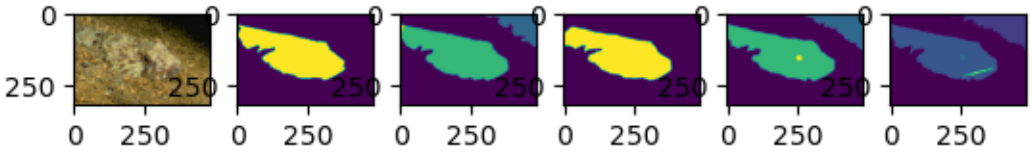


Figure 23: Bad Entropy 5

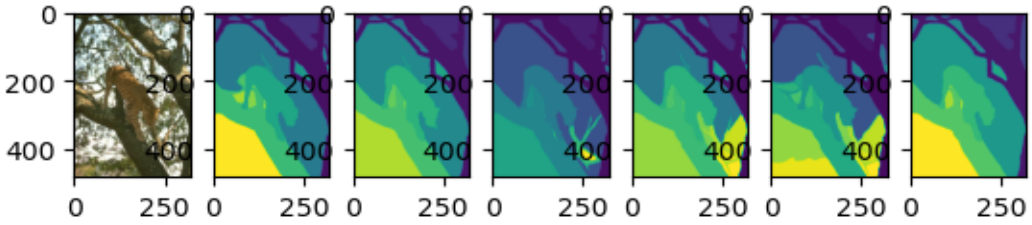


Figure 24: Worst fscore 1

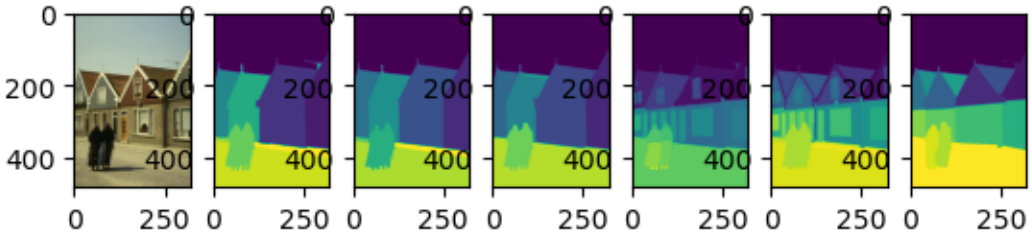


Figure 25: Worst fscore 2

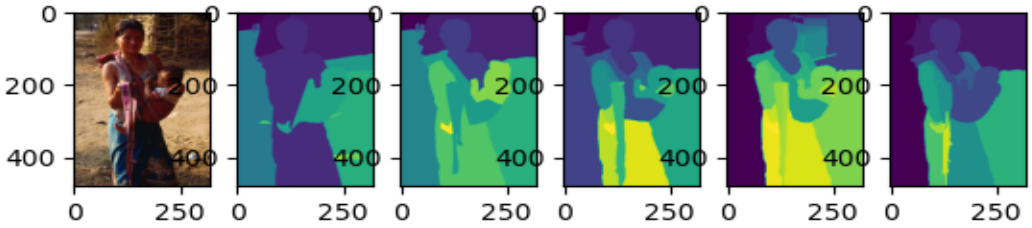


Figure 26: Worst fscore 3

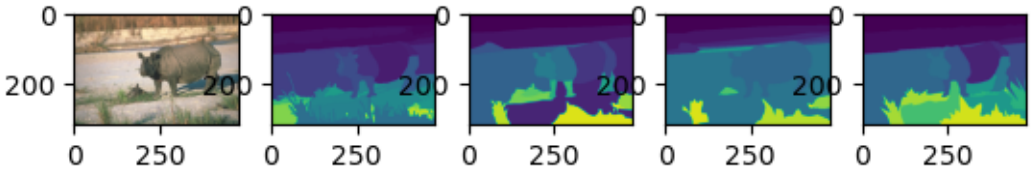


Figure 27: Worst fscore 4

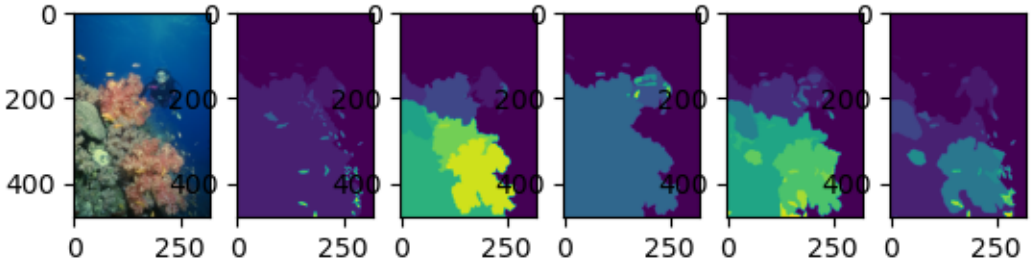


Figure 28: Worst fscore 5

2.5 Big Picture

2.5.1 Comparing Ground Truth and K-means

2.5.1.1 First Picture

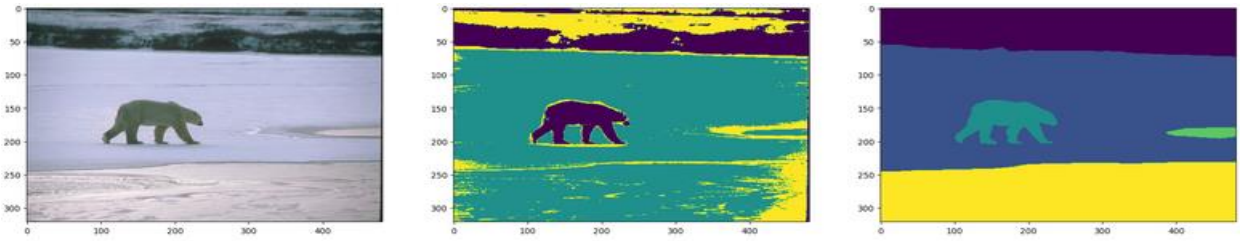


Figure 29: GT vs KM P1

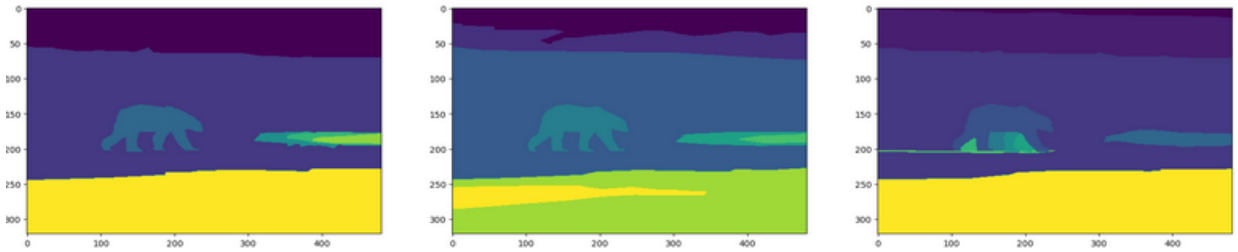


Figure 30: GT vs KM P1

2.5.1.2 Second Picture

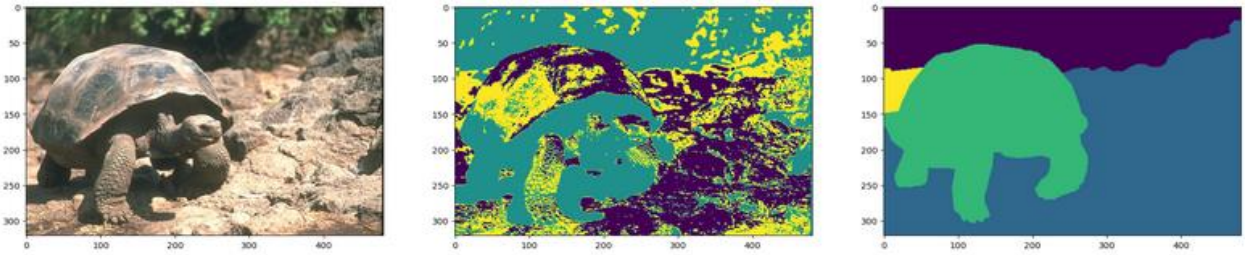


Figure 31: GT vs KM P2

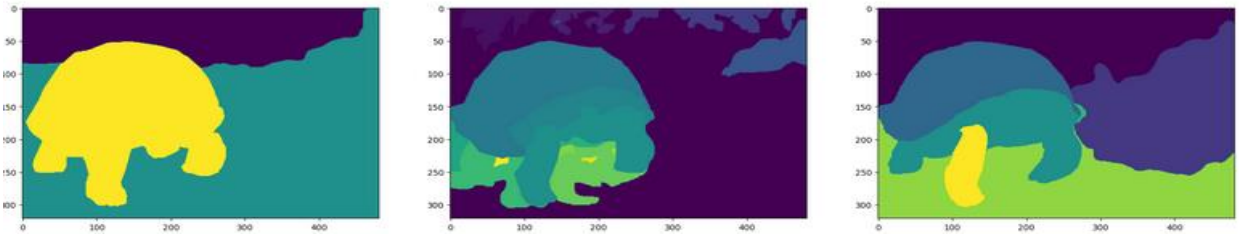


Figure 32: GT vs KM P2

2.5.1.3 Third Picture

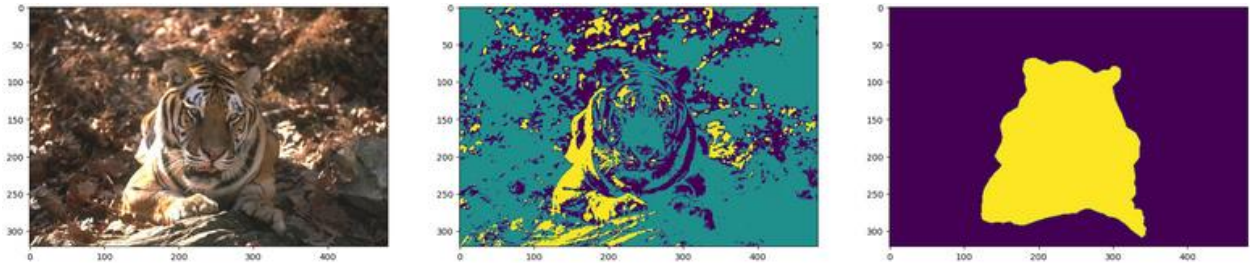


Figure 33: GT vs KM P3

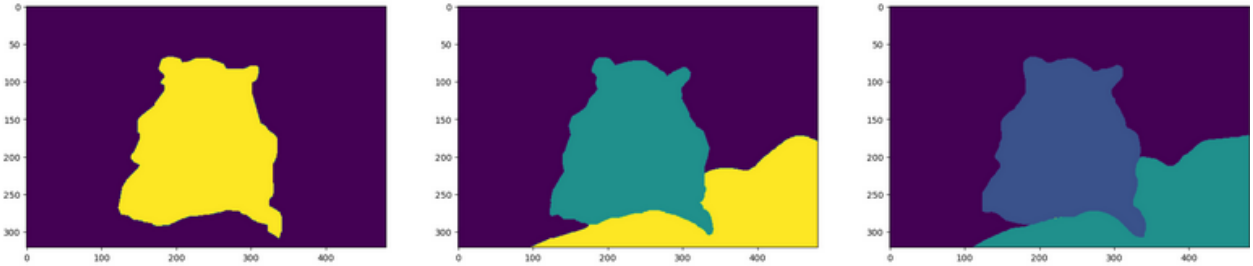


Figure 34: GT vs KM P3

2.5.1.4 Fourth Picture

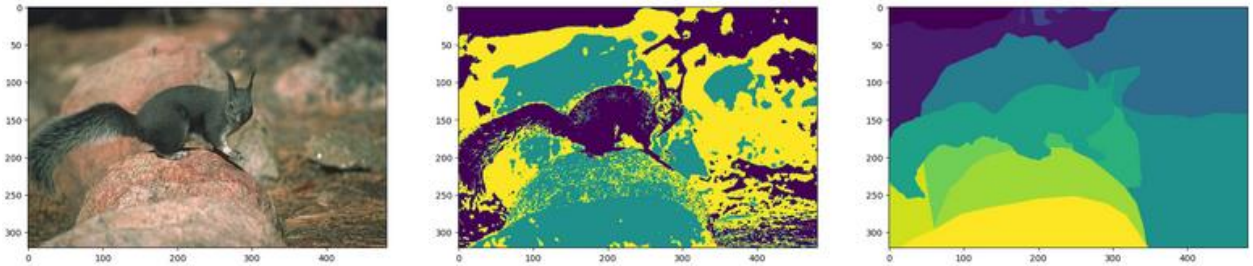


Figure 35: GT vs KM P4

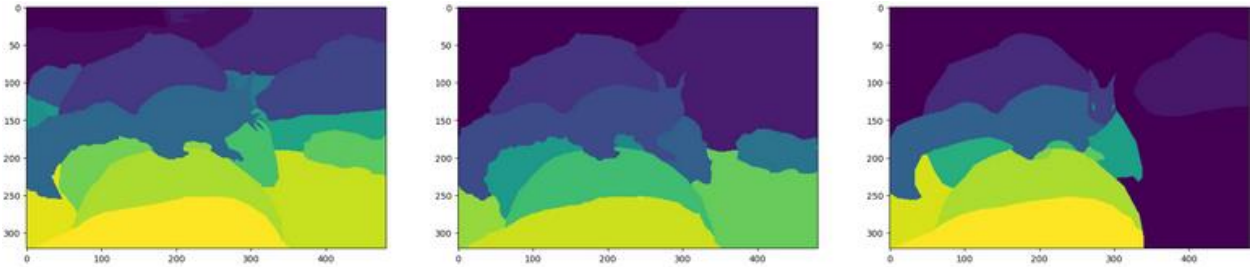


Figure 36: GT vs KM P4

2.5.1.5 Fifth Picture

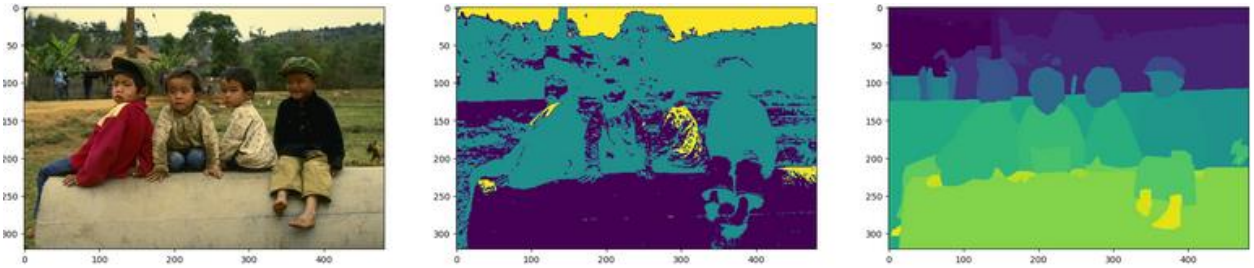


Figure 37: GT vs KM P5

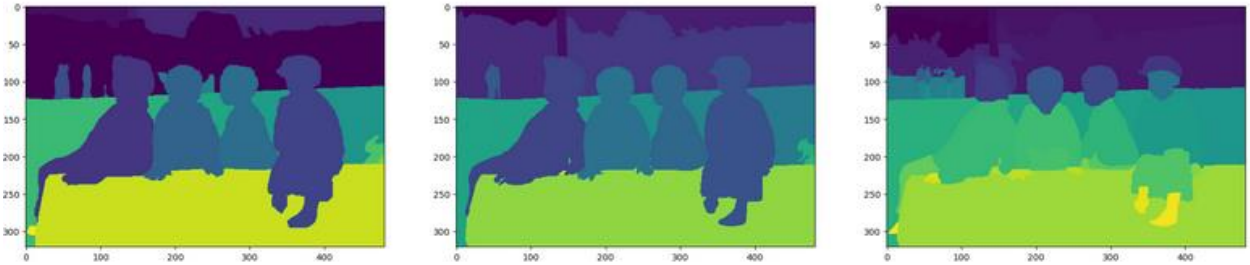


Figure 38: GT vs KM P5

2.5.2 Comparing Ground Truth and Normalized Cut

2.5.2.1 First Picture

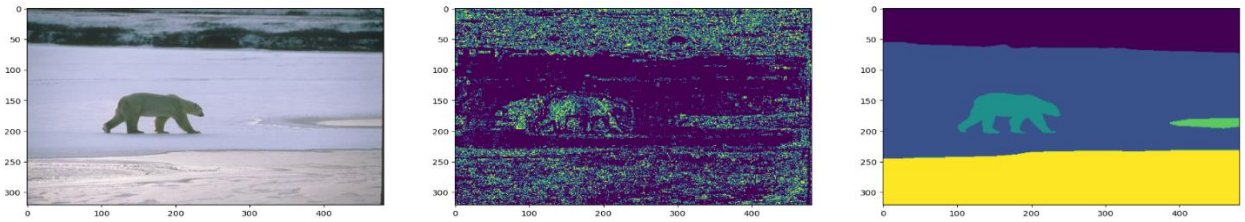


Figure 39: GT vs NC P1

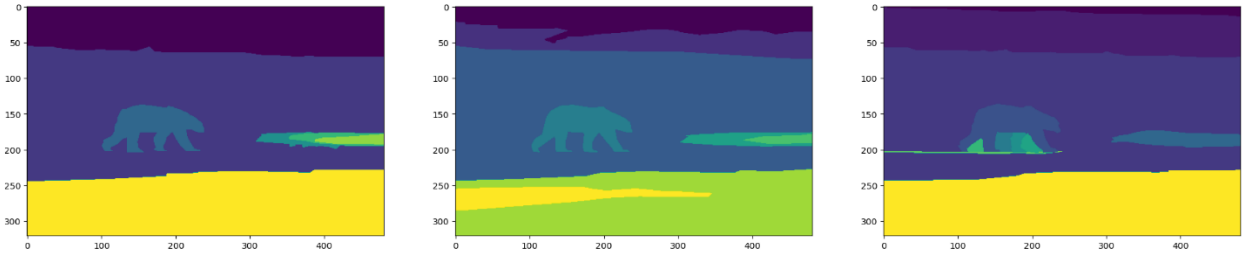


Figure 40: GT vs NC P1

2.5.2.2 Second Picture

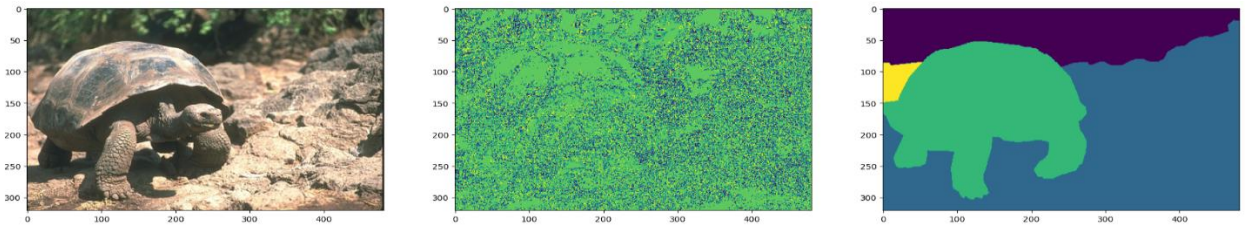


Figure 41: GT vs NC P2

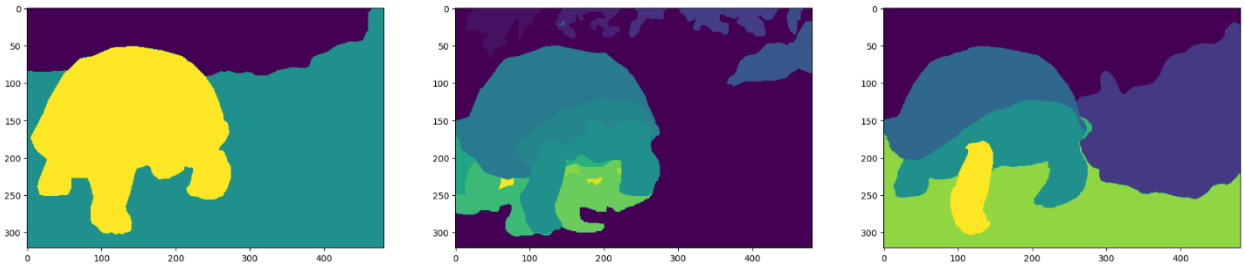


Figure 42: GT vs NC P2

2.5.2.3 Third Picture

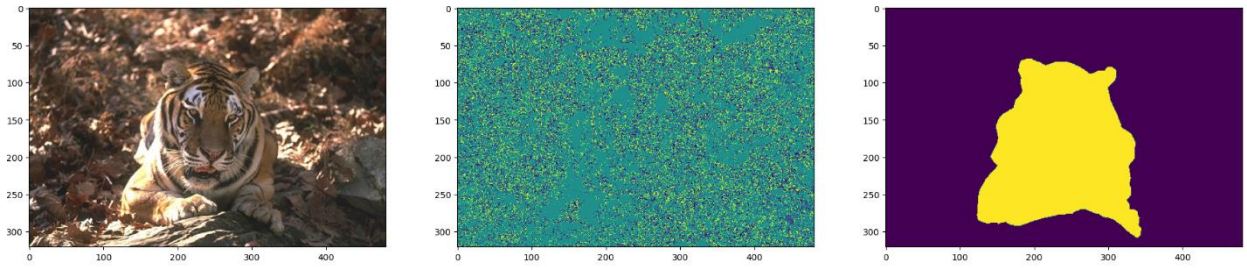


Figure 43: GT vs NC P3

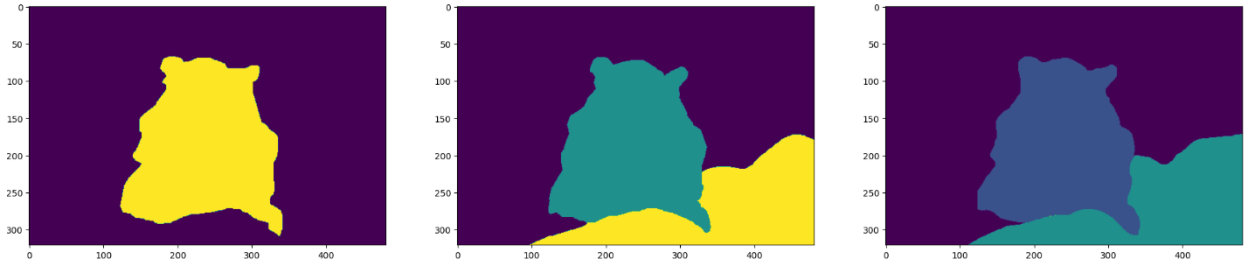


Figure 44: GT vs NC P3

2.5.2.4 Fourth Picture

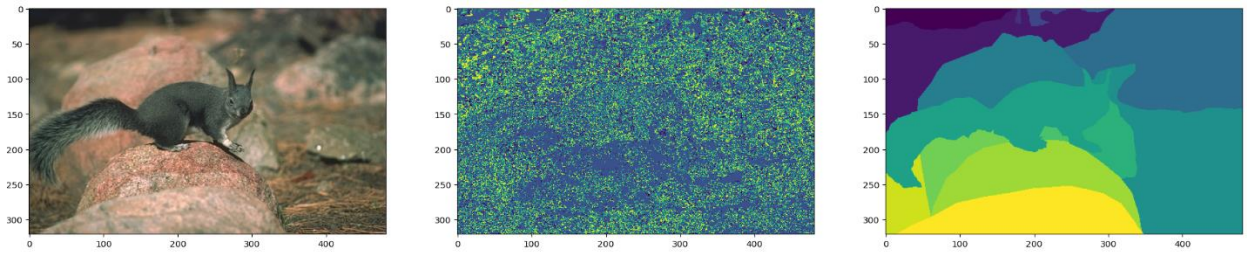


Figure 45: GT vs NC P4

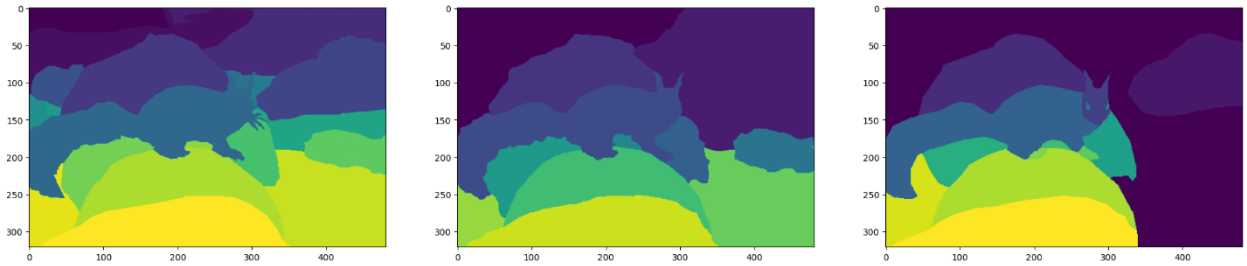


Figure 46: GT vs NC P4

2.5.2.5 Fifth Picture

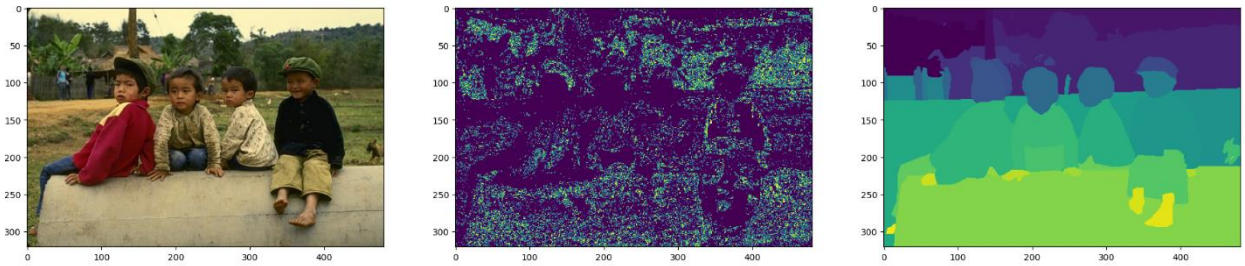


Figure 47: GT vs NC P5

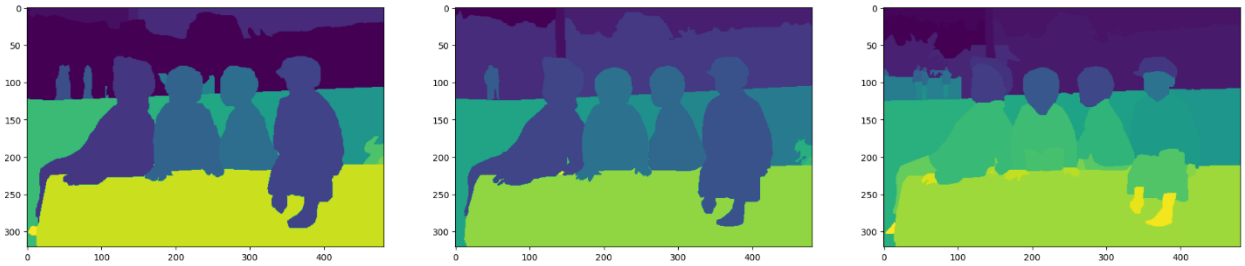


Figure 48: GT vs NC P5

We also Down sampled the images as sometimes the Spectral Clustering algorithm was just taking too much time to be computed (Available in Jupyter notebook).

2.6 Comparing K-means and N-cut

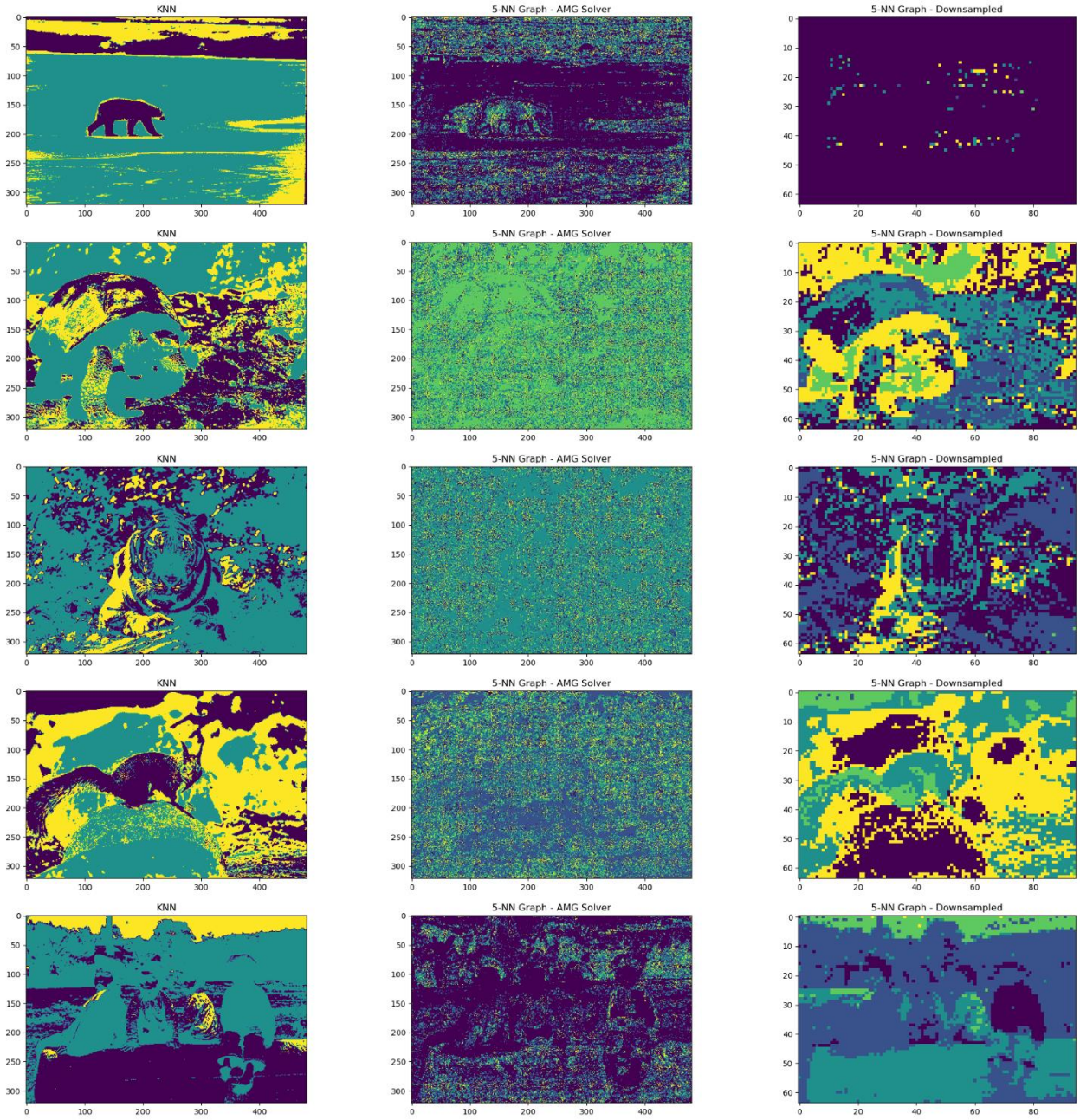


Figure 49: K-means vs N-cut

2.7 Adding Spatial Layout to K-means Algorithm.

In the previous parts, we used the colour features RGB. We did not encode the layout of the pixels. We want to modify that for K-means clustering to encode the spatial layout of the pixels.

2.7.1 Adding New Features

Our idea is to add two extra elements to our feature vector, **`x_pos`** and ***`y_pos`*** as a way to positionally and spatially encode our image, in such a way that items that are closer together on the x axis or the y axis or both are more similar than items that are further away.

This way, our feature vector's shape becomes **`(1, 5)`** instead of the original **`(1, 3)`** feature vector encoding only the RGB channels.

```
1.  seg_res_modified = []
2.
3.  for img_idx in selected_images:
4.      image = dataset["imgs"][img_idx]
5.      km = KMeans_modified(n_clusters=5, max_iter=75, distribution='uniform')
6.      km.fit(image)
7.      indices = km.predict(image)
8.      seg_res_modified.append(indices)
9.
```

2.7.2 Results

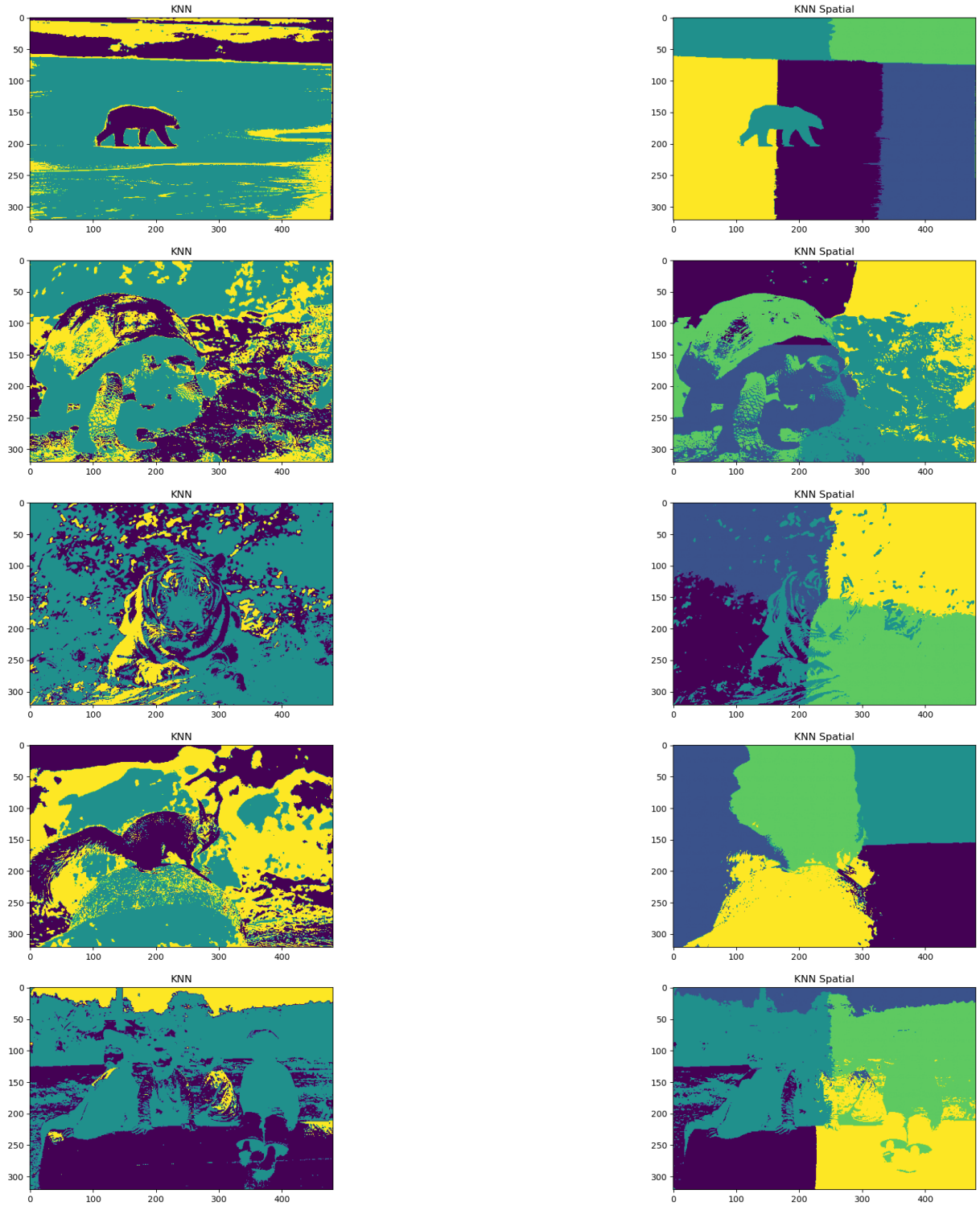


Figure 50: K-means vs Spatial K-means