CS371 Digital Image Processing
Exercise 5
Unsupervised Image Segmentation Algorithms

Alexandros Angelakis
csd4334@csd.uoc.gr

January 8, 2023

1 Part A: Comprehension Questions

K-means vs Mean Shift

Q: What are the advantages and disadvantages of the two algorithms?
A: First of all, let's talk about the **K-Means algorithm**. It does have a lot of advantages but also some disadvantages what we should have in mind.

Let's first start with its advantages. To begin with, K-means guarantees to converge in a finite number of iterations, meaning it always converges regardless of the size or the structure of the data sets we run the algorithm with, thus it easily adapts to new examples. Secondly, considering the previous advantage, we can tell that it is capable of scaling to large data sets. Since it is relatively simple to implement, is it often faster than the other common clustering methods, like hierarchical clustering, and we can also warm-start the positions of centroids, in order to have the algorithm find the "best" clusters for our data faster.

Let's now talk about the disadvantages of the K-Means algorithm. The most significant drawback of this algorithm is that it is extremely sensitive to initialization, thus it has poor convergence speed and bad overall clustering. Since K-Means is a heuristic algorithm, it means that it requires initial means. The problem is that it doesn't learn the K from the data, so the user has to manually choose the number of clusters. There are some ways to find an ideal value of K (elbow method, trial and error), but it's often difficult to determine whether a given value is incorrect. We previously mentioned in the advantages that it always converges regardless the data. This is not always the best solution since it can cluster data that are not balanced. Lastly, the algorithm is scaling with high dimensions, meaning that as the number of dimensions increases, a distance-based similarity measure converges to a constant value between any given examples.

Our next unsupervised clustering algorithm is the **Mean Shift algorithm**. This algorithm also has some advantages and disadvantages that are worth mentioning.

Let's begin with the advantages. The Mean Shift algorithm automatically selects the number of clusters K, since it initially creates as many clusters as the number of data-points and then it deletes the overlapping ones, contrary to other clustering algorithms, like the aforementioned K-means. As a result of that, the output of Mean Shift is not dependent on the initialization part. Lastly, it can also model the complex clusters which have non convex shape, and it works very well on spherical-shaped data.

One of the drawbacks of this algorithm is that it is not highly scalable, as it requires multiple nearest neighbor searches, in order to find the the mean of all the data-points in all the sliding windows, during its execution. It has a complexity of $O(n^2)$. Furthermore, since we manually choose the size of each sliding window in our data set (bandwidth), this can lead to major problems in our output, and selecting a wrong value can lead to very bad results. Lastly, because of the fact that we do not have any direct control on the number of clusters, this will be a problem in some applications where we need a specific number of clusters.

Q: What are the elements of the K-Means algorithm that should be properly defined because they determine its performance, and what are those of the mean shift algorithm?
A: For the **K-Means algorithm**, we should first properly define the number of clusters we want to segment our data with, in order to have the best result, good overall clustering. Another element that is very important is in the initialization step. How we initialize the positions of the centroids is crucial for the execution time of the algorithm, how fast the algorithm converges.

Lastly, distance metric plays a significant role in identifying the similar data points and forming respective clusters. Although K-Means uses the euclidean distance as its default distance metric, other distance measures are also important for some data sets, and can have very different results.

For the **Mean Shift algorithm**, we only need the size of the regions we are going to consider as our sliding windows, called bandwidth. Choosing the right bandwidth/radius is really important, if we consider a small value we will have an extreme case that each points is its own cluster and if we consider a large value there will only be one cluster containing all the data-points. Since it fails to find the correct clusters for some data-sets, the structure of our data points is also an element we should consider before using this algorithm. The distance metric here also plays a crucial role as we mentioned for the K-Means algorithm.

2 Part B: Segmentation Algorithm Implementation

In this section, we had to implement two segmentation algorithms, by exploiting for each image given 3 data types: their colour, colour + spatial coordinates and their depth map. Using the **K-Means clustering algorithm** for each data type I mentioned earlier, and for $k = 3, 5, 7$ we are able to partition the image into some very interesting parts. Let's see the results of the partitioning and make some comments.

We had three images to work with, one image that depicts a bedroom, an image that depicts an office and an image with a table room. Let's begin with the office image using the default distance metric, the euclidean distance.

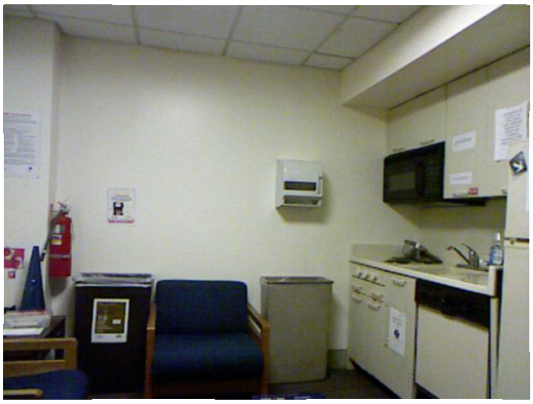

Figure 1: Original office image

Figure 2: Labeled office colour

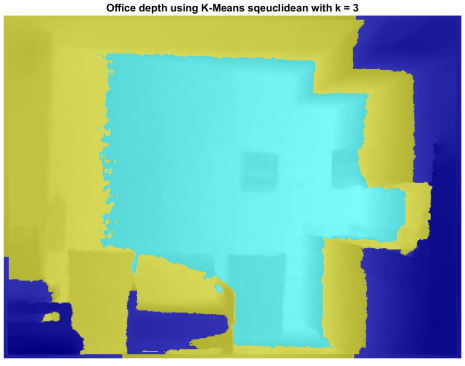

Figure 3: Labeled office depth

Figure 4: Labeled office colour + coord

By partitioning the office image with 3 clusters, we can see that for each data type we have a very different result. For the colour data type, we can see that with 3 clusters is difficult to not group together some colours that are not that similar. Darker colours of the image have been assigned a light blue colour, less darker pixels blue and bright parts of the image yellow colour. Fore the depth of the image, things are really interesting. We can see that for "deeper" parts of the image (things that are far from the camera) are labeled with light blue colour, less far from the camera have been assigned yellow colour, and

things that are closer to us are labeled with blue colour. For the colour + spatial coordinates, things are a bit complicated. The image has been partitioned into 3 segments, mostly based on the spatial coordinates, colour does not affect it much. We can see that similar things that are close to one another, are grouped together.

Let's now see for 5 clusters, how the resulting images are affected.

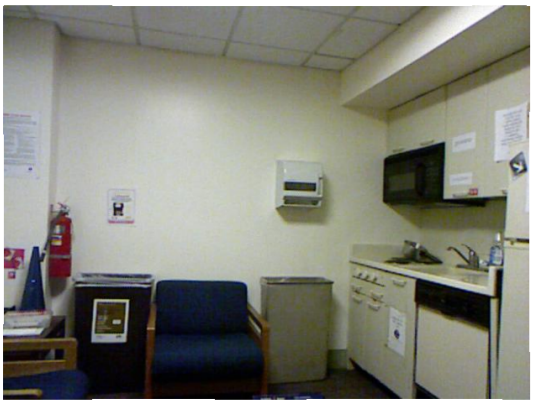

Figure 5: Original office image

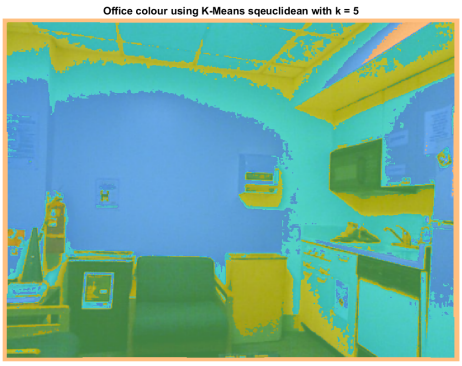

Figure 6: Labeled office colour

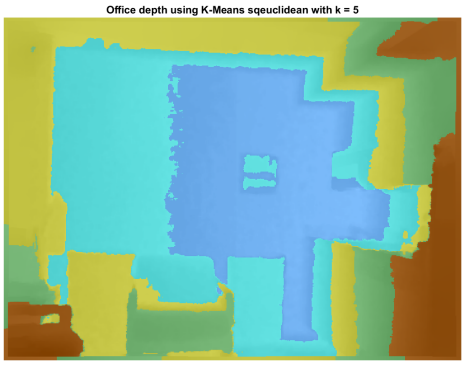

Figure 7: Labeled office depth

Figure 8: Labeled office colour + coord

We can easily say that the new image is grouped better with 5 clusters, instead of just using 3. In the image with only the colour, we can see that now the colours that are grouped together are very similar, from the colour of the light on the wall, to the colour of the chair. In the image with only the depth, we can now see that the depth of the image is depicted better, each colour corresponds to deeper things in the image. For the colour + spatial coordinates, it somehow grouped together some pixels with similar colours, but again the spatial coordinates of those pixels are more effective here. Again, every piece of furniture that is next to one another and similar, is grouped together.

Lastly, let's see the results with 7 clusters.

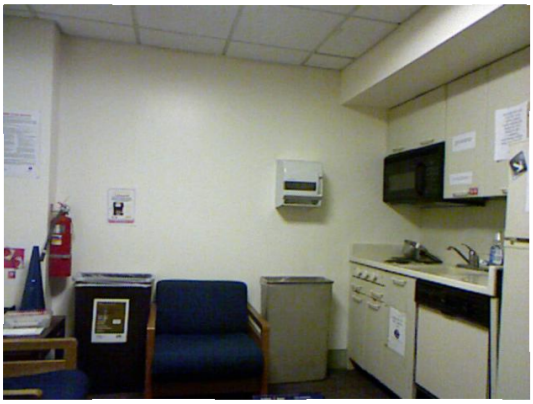

Figure 9: Original office image

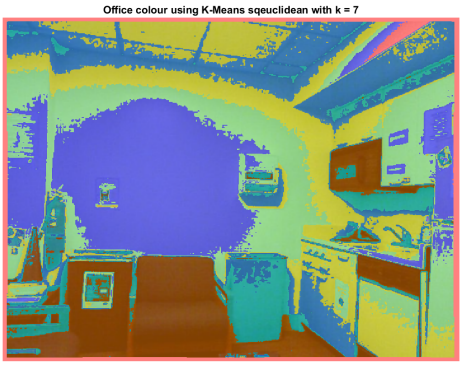

Figure 10: Labeled office colour

Figure 11: Labeled office depth

Figure 12: Labeled office colour + coord

For the colour labeled image, it perfectly grouped together every similar colour of the image, from the light on the wall/ceiling, to every piece of furniture depicted. For the depth labeled image, it also did a pretty good job showing the depth of the image, thus the construction and the area in the office. Lastly, for the colour + spatial coordinates labeled image, we can again see that pixels that are close to another with a little big of colour similarity between them, have been grouped together.

In my opinion, using $k = 7$ clusters, the K-Means algorithm performs better on our images. The rest of the images (tableroom, bedroom) will be at the end of this report, in the demonstrations section. We have the same results on the grouping part, thus I will not make any extensive comments on them.

Since 7 clusters give us the best result, let's run the K-Means algorithm again, with different comparison metrics, cityblock and cosine. Let's start with the cityblock metric.

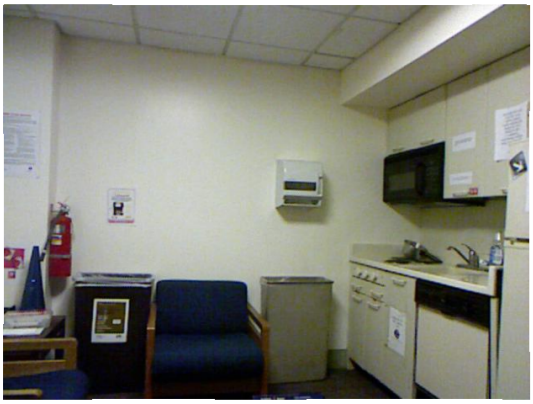

Figure 13: Original office image

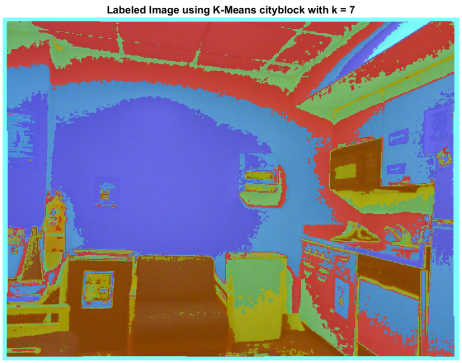

Figure 14: Labeled office colour

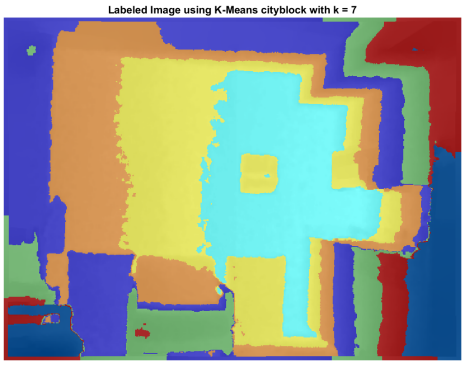

Figure 15: Labeled office depth

Figure 16: Labeled office colour + coord

For the colour labeled image, there are minor differences with the euclidean metric, for example on the bottom right of the image, some colours are differently grouped together, but other than that, there differences are not very perceivable. The same goes for the depth labeled image. The only major difference between them is in the center of the image, the colours on the wall are grouped together into one cluster, instead of breaking it into two. I think this is better, because a wall can't have parts that are deeper in the image. The most perceivable differences between the euclidean and the cityblock distance metric is in the colour + spatial coordinate labeled image, since the shapes of the clusters are different, but the grouping is kind of the same.

To finish with the K-Means algorithm, let's now run the algorithm with the cosine distance metric and comment on the results.

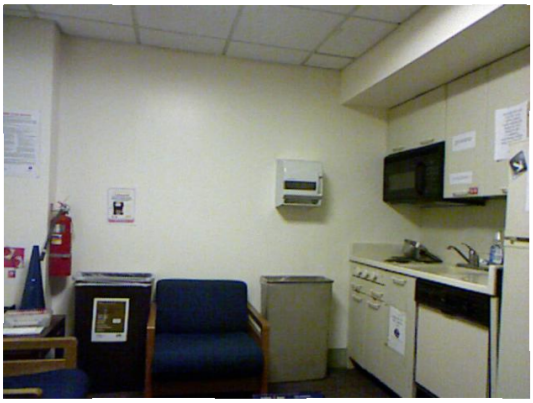

Figure 17: Original office image

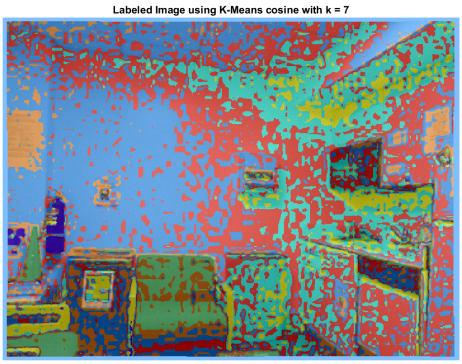

Figure 18: Labeled office colour

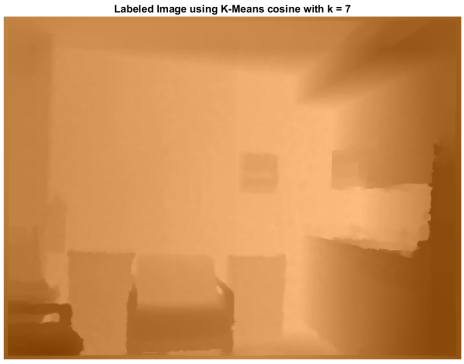

Figure 19: Labeled office depth

Figure 20: Labeled office colour + coord

We can easily say that the cosine distance metric is not useful for our images. The grouping is all over the place and it feels like the new labeled image has noise. However, there are some part of the image that is grouped together well, some parts on the wall, ceiling, furniture. If we take a look at the depth labeled image, we can see that K-Means was not able to partition the image, resulting in having only 1 cluster. Finally, for the colour + spatial coordinates, we can see a "rainbow" of colours, starting from the top left of the image. This means that it only groups together pixels of the image that are close to one another in a piece of triangle, starting from the top left.

COSINE BONUS: If we take a look at the cosine similarity between two vectors, the mathematical formula is the following:

$$Cosine(u, v) = \frac{u^T v}{\|u\|\|v\|}$$

When we read our image, we immediately change its values to be within the range of $[0 - 1]$. This has as an effect that some pixels will have very low relative magnitudes, making them effectively zero. This results in the denominator of the mathematical formula to almost be eliminated, very close to zero. To avoid this problem, we just adding 1 to every value in our image, and the problem is fixed.

We covered the K-Means algorithm in an extensive way, let's now look at an other clustering algorithm named **Mean-Shift**. I'm again going to show the office image, the rest of the images are going to be in the "Demonstrations" section. We had three bandwidth to play with $bw = 0.2, 0.35, 0.6$. We are going to run the algorithms with the data types of the image's colour and their colour + spatial coordinates. For the last data type, things have been very irritating, since the given algorithm wasn't converging, even after 7+ hours, so in order to find a way to converge, I had to resize all the images to 128x128 pixels. Let's begin with the bandwidth of 0.2.

Figure 21: Labeled office colour

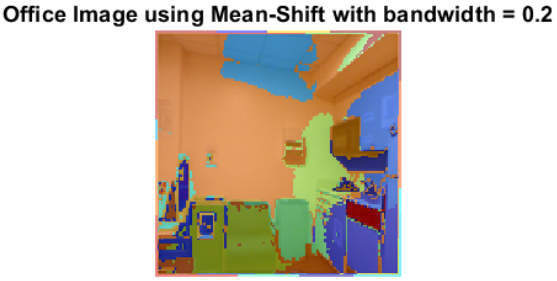

Figure 22: Labeled office colour + coord

As we can see, the bandwidth of 0.2 is a little too low for the Mean-Shift algorithm to perfectly group together the different colours in the image. We can see that it only has 4 clusters, but the grouping of the colours is not quite there. We can confirm that by looking the colour of the chair and the colour of the trash can, completely different colours grouped together. For the colour + spatial coordinates, we can see that the coordinates of the pixels affect the clustering. Things that have the same colour and are close to each other have been assigned to the same cluster.

Let's run the algorithm now with $bandwidth = 0.35$.

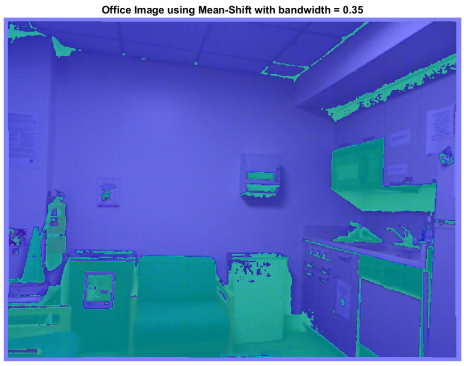

Figure 23: Labeled office colour

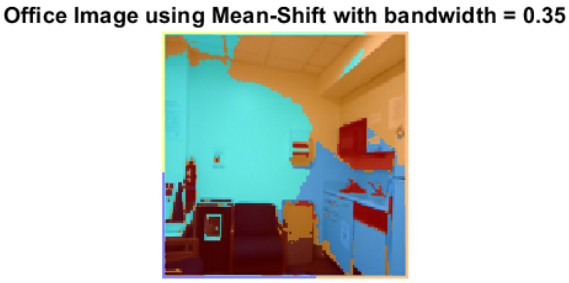

Figure 24: Labeled office colour + coord

Now the grouping is even worse, because we increased the number of the bandwidth and thus, we decreased the number of possible clusters in our grouping. We only have two number of clusters, and as we can see it grouped together the bright colours and the dark colours of the image. For the colour + coordinates, we can see that again, the coordinates of the pixels have affected the clustering and also, there are more clusters here than the colour.

Lastly, let's use as bandwidth 0.6.

Figure 25: Labeled office colour

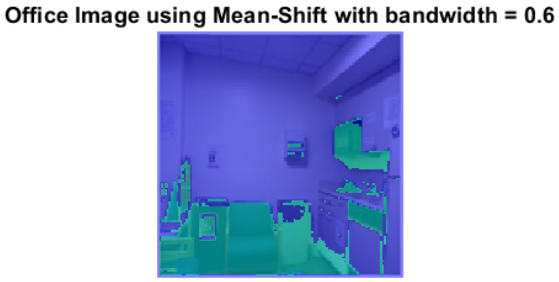

Figure 26: Labeled office colour + coord

The worst clustering we've seen so far, we only have one cluster, since the bandwidth is very large. Nothing to comment on that. On the other hand, we can see a very good clustering from the colour + coordinates, because we have 2 clusters (the bandwidth is large) and we can see that the bright colours of the image have been assigned a dark blue colour and the dark colours a light blue.

For this image, the best bandwidth was 0.2, but for the general other images, the best was 0.35, and that's what I'm going to use for the cityblock mean-shift algorithm. Let's take a look of how it run.

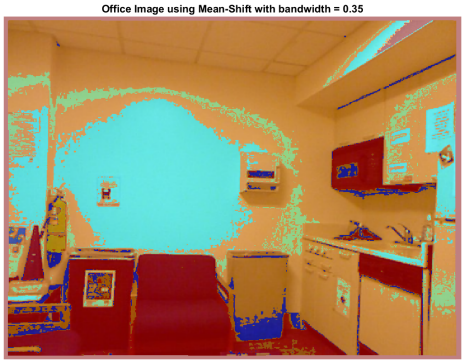

Figure 27: Labeled office colour

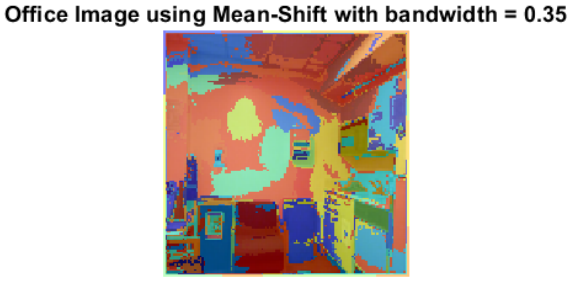

Figure 28: Labeled office colour + coord

As we can see, by using as distance metric the cityblock it has better grouping. It is not the best but surely better than having only 2 number of clusters. The colour on the walls has been assigned orange colour and even more brighter spots of the image have been assigned light blue and purple. The dark spots have been assigned black. On the other hand, we can see that for the colour + coordinates, the number of clusters is insanely large (I can count at least 9). We can see that the clustering is a lot different from the colour. For some reason, giving to this algorithm more information about our image, the colour is the one data type that is affected the most, and for that, we can see that slightly changes on the colours are assigned to different clusters.

3 Part C: Analysis of Results

1. The data type that leads to the best results is only the colour. This occurs because when we look at an image, the first thing that our eye captures is the colours and the details of the image. Since every image can have many different colours in it, grouping them together will have a better visual result, because the details of the image can also be shown.
2. The algorithm that gave me the best result is the K-Means algorithm. This happens because by doing trial and error for the number of clusters, we found the best number that gives us the best grouping. Yes, we don't initialize the positions of the centroids but it is still faster than the mean-shift for our example images.
3. Of course, the colour characteristics of the objects presented in the image and in relation to the background pattern play a significant role in the result of the segmentation. Some objects depicted in the image have a weird pattern, with different colours on them, but the clustering algorithms managed to group together the pixels that have very similar colours.
4. Increasing the complexity of the data is never a good idea, especially when we have images of that size. To be more precise, the images that we are working with have the size of 480x640x3 meaning that if we simultaneously use as input to the segmentation algorithms all these data types, the size of the data will be $480 * 640 * 8 = 2.457.600$, a large number for such a small image. Yes, it might improve the performance of the segmentation, but it's not worth the computational cost.

4 BONUS

1. Throughout this course, we've seen various filters and their effects on the images, from smoothing, to edge detection etc. What if we take the original images and pass them through a Gaussian filter, meaning we remove any possible noise in the image with the cost of blurriness if we are not careful, and then try to run the same segmentation algorithms to get (possibly) a better result? I'm going to use the image with the office, and compare the result after filtering it with the Gaussian filter and before filtering it. I'm only going to use the colour here, because the colour + spatial coordinates have been a pain in the ass, especially for the Mean-Shift algorithm.
2. Let's filter the image with standard deviation of 2 and see the results. As we can see, the filtered image has been smoothed out, some noise has been reduces but also some regions of the image have been blurred. Let's now test our two algorithms on the new image.

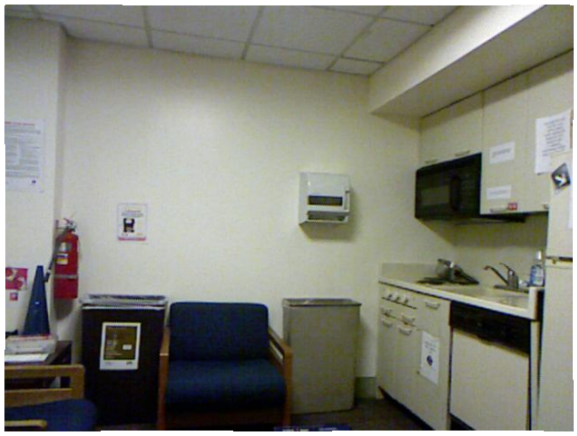

Figure 29: Official office image

Figure 30: Smoothed office image with sigma = 2

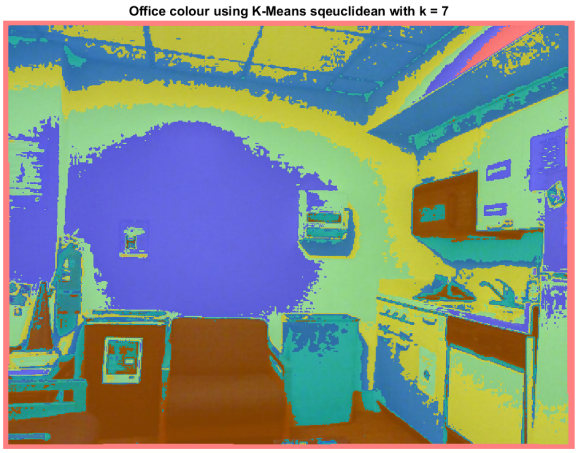

Figure 31: Labeled office colour

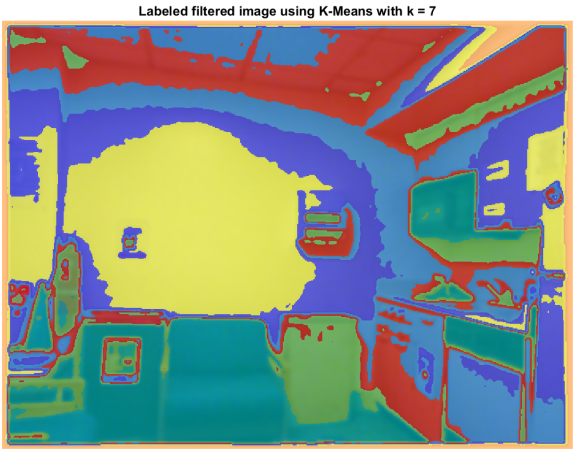

Figure 32: Labeled smoothed office colour

Figure 33: Labeled office colour

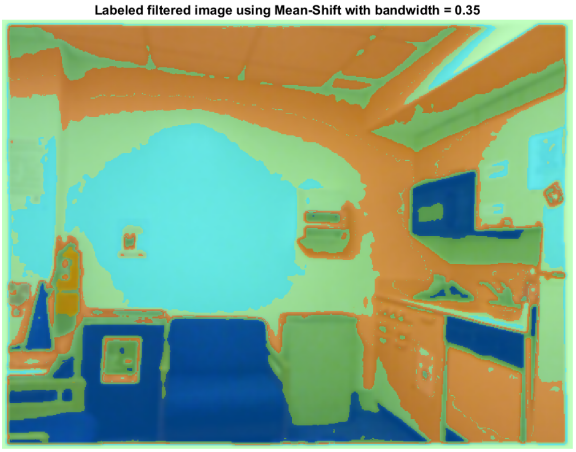

Figure 34: Labeled smoothed office colour

We can see some interesting effects of the Gaussian filter on the new smoothed labeled image. You can see in the labeled image with filtering it, some areas of the image that have some little dots that don't quite fit. For example, on the roof of the office there are no dots and is very smooth, same at the top right corner of the image. There are no weird pointy labeled colours in the new image, because of the smoothing we did to it. I think, the smoothing we did actually has better results, showing better the details on the walls and on the furniture, because all the noise from the original image has been eliminated.

The biggest difference is in the Mean-Shift algorithm, where the green group (like noise) of the official image has been completely eliminated. Looking at the new labeled image we can not see any noise grouped together, everything is smoothed and perfectly segmented together, meaning that again by applying a Gaussian filter to this image, we are having better results in our segmentation.

To sum up, there are a lot of ways to transform our input data (image) and play with these algorithms. I was curious about applying a Gaussian filter to this image and it turned out having better results. By smoothing the image, the noise that is hardly seen with our bear eyes is eliminated, so the grouping has been more efficient and more smoothed, for both segmentation algorithms. The fact that the image is a little bit blurrier than the original didn't affect the clustering at all.

5 Demonstrations

All the rest images and the grouped images using the K-Means algorithm. The same that I mentioned earlier applies here.

Figure 35: Original bedroom image

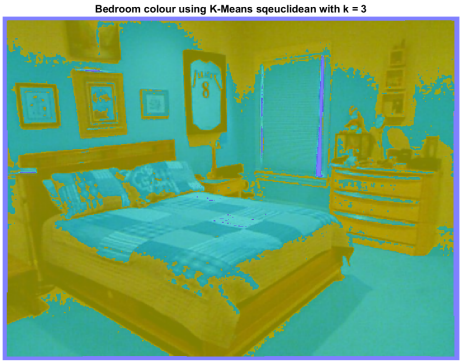

Figure 36: Labeled bedroom colour

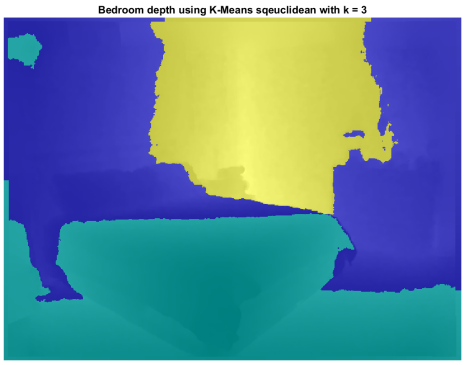

Figure 37: Labeled bedroom depth

Figure 38: Labeled bedroom colour + coord

Figure 39: Original bedroom image

Figure 40: Labeled bedroom colour

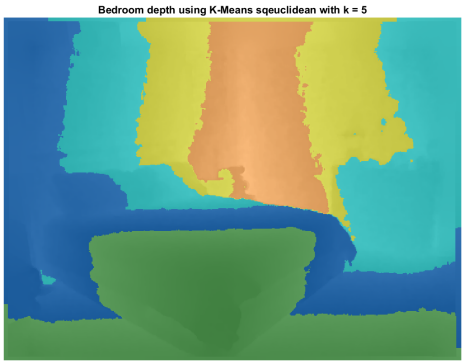

Figure 41: Labeled bedroom depth

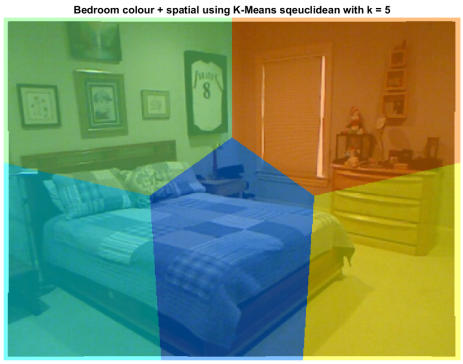

Figure 42: Labeled bedroom colour + coord

Figure 43: Original bedroom image

Figure 44: Labeled bedroom colour

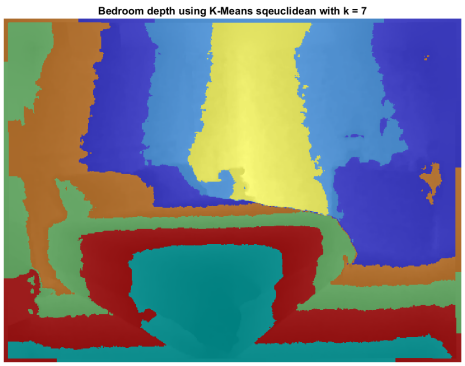

Figure 45: Labeled bedroom depth

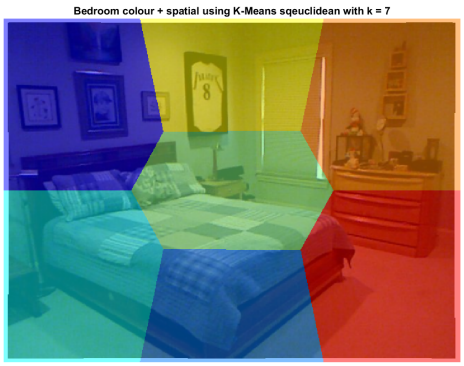

Figure 46: Labeled bedroom colour + coord

Figure 47: Original tableroom image

Figure 48: Labeled tableroom colour

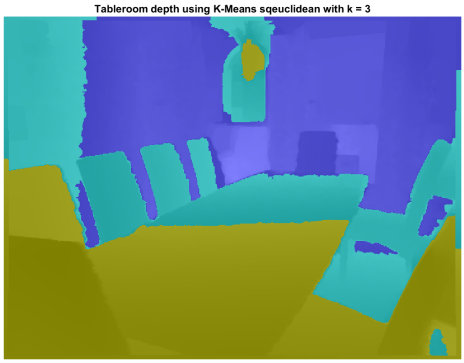

Figure 49: Labeled tableroom depth

Figure 50: Labeled tableroom colour + coord

Figure 51: Original tableroom image

Figure 52: Labeled tableroom colour

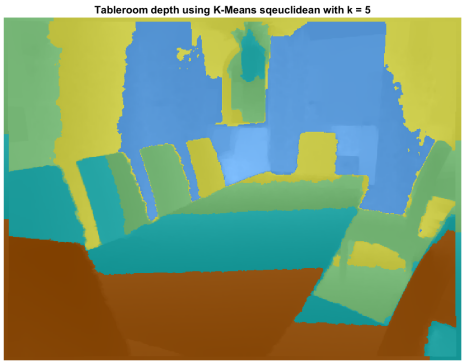

Figure 53: Labeled tableroom depth

Figure 54: Labeled tableroom colour + coord

Figure 55: Original tableroom image

Figure 56: Labeled tableroom colour

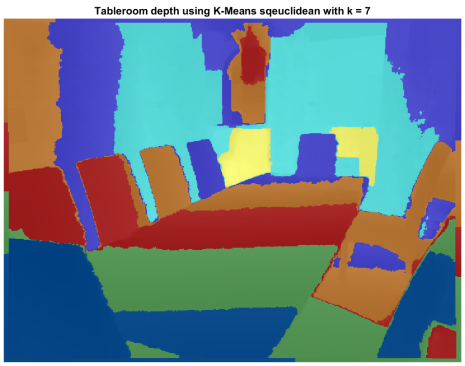

Figure 57: Labeled tableroom depth

Figure 58: Labeled tableroom colour + coord

Figure 59: Original bedroom image

Figure 60: Labeled bedroom colour

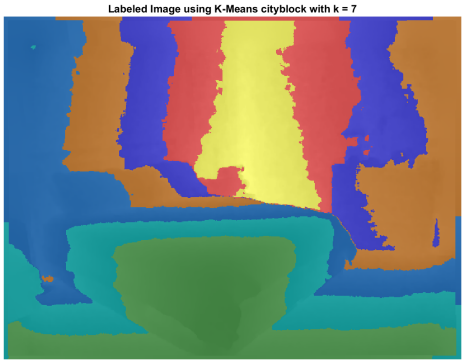

Figure 61: Labeled bedroom depth

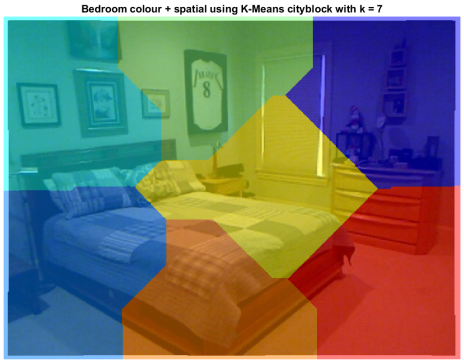

Figure 62: Labeled bedroom colour + coord

Figure 63: Original bedroom image

Figure 64: Labeled bedroom colour

Figure 65: Labeled bedroom depth

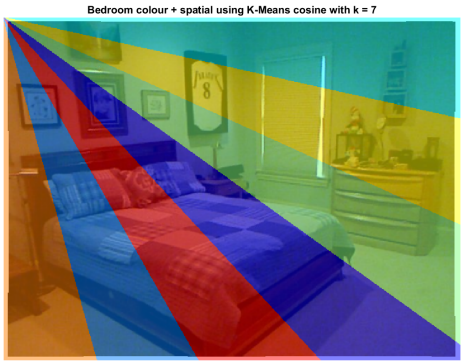

Figure 66: Labeled bedroom colour + coord

Figure 67: Original tableroom image

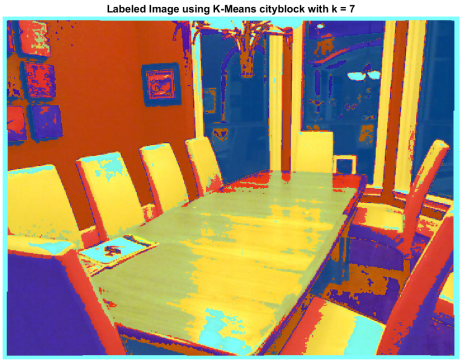

Figure 68: Labeled tableroom colour

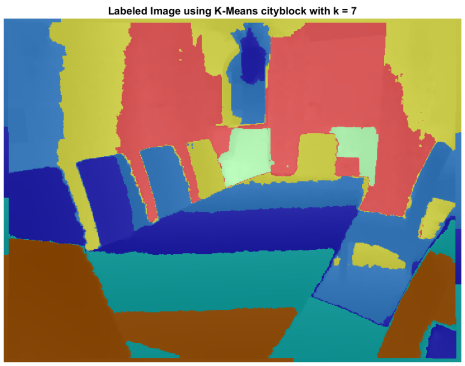

Figure 69: Labeled tableroom depth

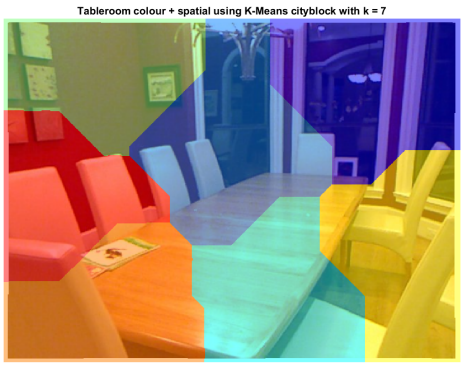

Figure 70: Labeled tableroom colour + coord

Figure 71: Original tableroom image

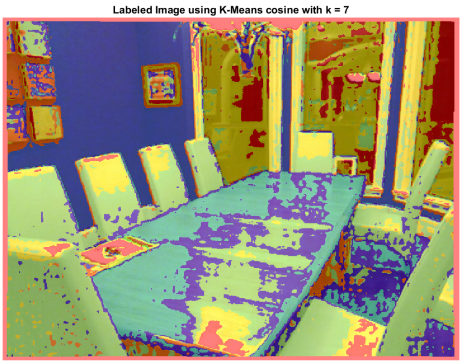

Figure 72: Labeled tableroom colour

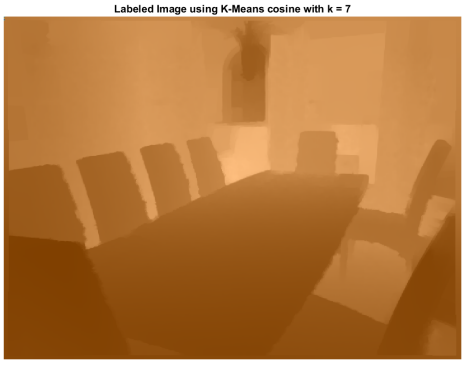

Figure 73: Labeled tableroom depth

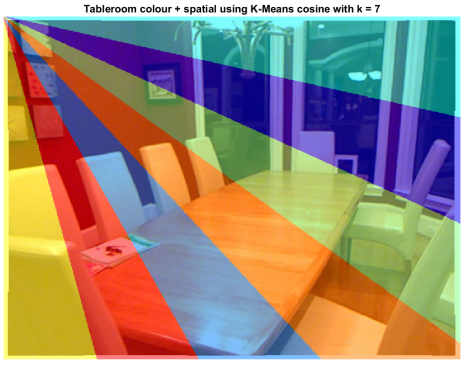

Figure 74: Labeled tableroom colour + coord

All the rest images and the grouped images using the Mean-Shift algorithm. The same that I mentioned earlier applies here.

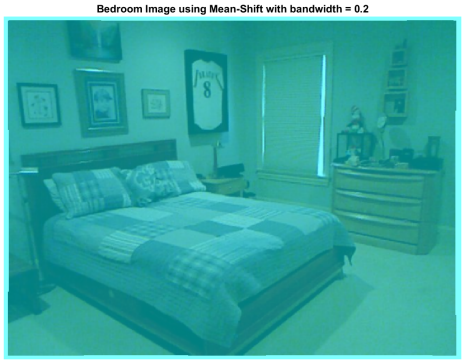

Figure 75: Labeled bedroom colour

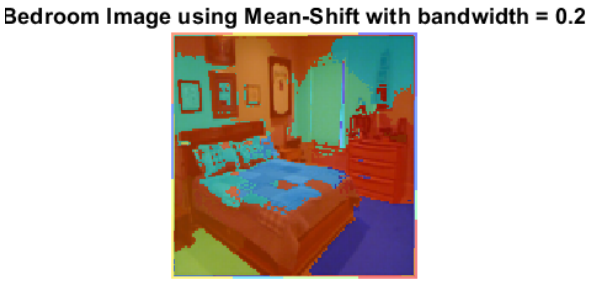

Figure 76: Labeled bedroom colour + coord

Figure 77: Labeled bedroom colour

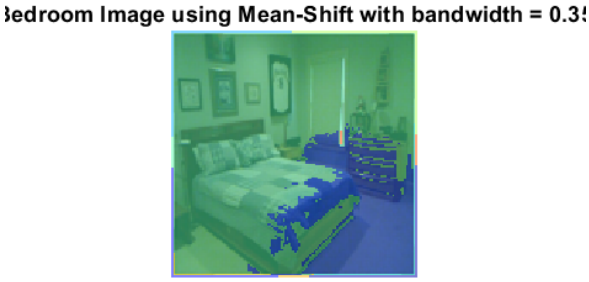

Figure 78: Labeled bedroom colour + coord

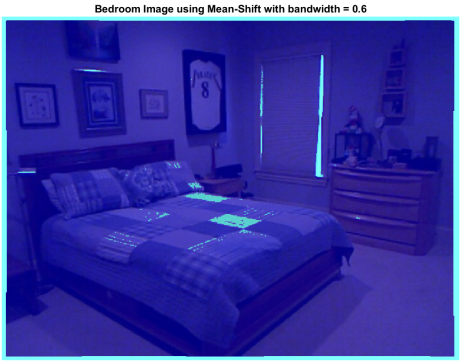

Figure 79: Labeled bedroom colour

Figure 80: Labeled bedroom colour + coord

Figure 81: Labeled tableroom colour

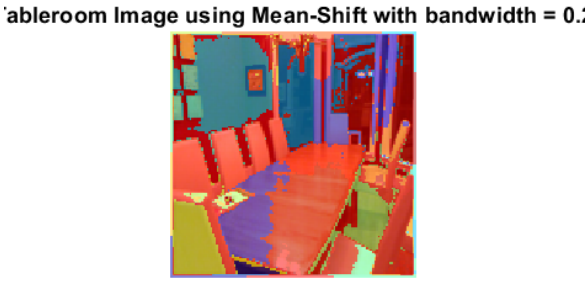

Figure 82: Labeled tableroom colour + coord

Figure 83: Labeled tableroom colour

Figure 84: Labeled tableroom colour + coord

Figure 85: Labeled tableroom colour

Figure 86: Labeled tableroom colour + coord

Figure 87: Labeled bedroom colour

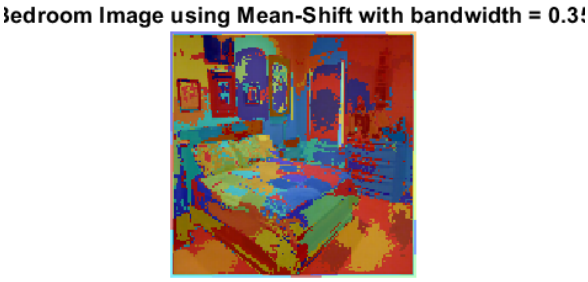

Figure 88: Labeled bedroom colour + coord

Figure 89: Labeled tableroom colour

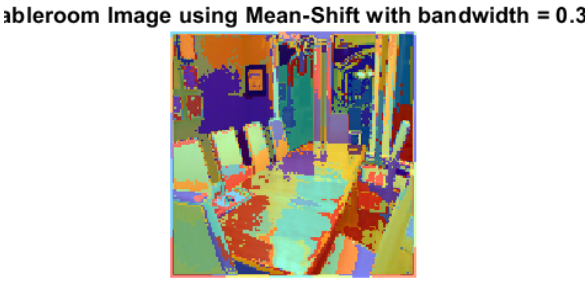

Figure 90: Labeled tableroom colour + coord