Universitat Politècnica de Catalunya
Universitat de Barcelona
Universitat Rovira i Virgili

Master in Artificial Intelligence

Computing vision

# Image segmentation: bottom-top

*Authors:*
Alejandro Suárez Hernández
Johannes Heidecke

November 2016

# Contents



# List of Figures

# 1 k-means segmentation

We can see in figures 1 and 2 the result of segmenting the image `animals.jpg` with k-means with K=5 and K=10, respectively. The K parameter denotes the number of different clusters (i.e. regions in which the image is segmented). No transformations have been applied up to this point.

Whether the results are satisfactory or not depends on the application. If, for instance, we wish to separate the animals from the background we would like that the labels assigned to the pixels that belong to the animals are different than those assigned to the vegetation that surrounds them. Should this be case it turns out that with K=5 some pixels from the animals are assigned labels that correspond to the background (check the primate and the tigers).

On the other hand K=10 does a better job distinguishing the animals from the background, although (arguably) at the price of having some over-segmentation (the unnecessary distinction between different shades of the animals hair).

In both cases we can see that the assignment of labels to the lion and the grass is somewhat erratic and noisy. This is due to the texture of those regions. We will see later that this phenomenon can be counteracted by means of scaling down (losing image details), by taking into account the spatial coordinates of each pixel, or doing both.

It is also worth mentioning how the algorithm can have artistic purposes, serving as a technique to achieve the 'posterization' effect present in some photographic suites such as GIMP or Photoshop.

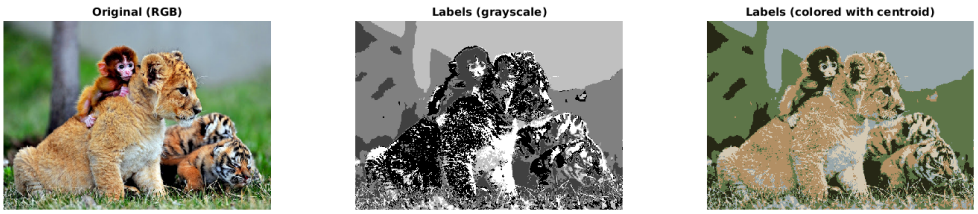


**Figure 1:** Segmentation of `animals.jpg` with K-means (K = 5)

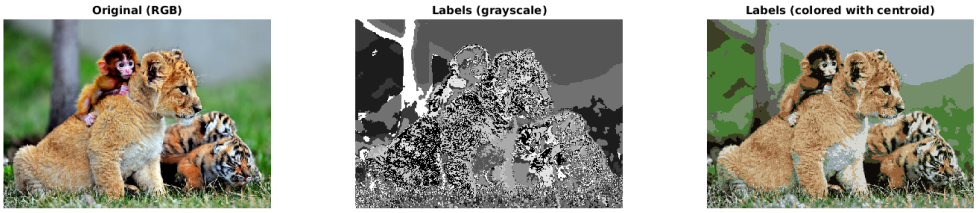


**Figure 2:** Segmentation of `animals.jpg` with K-means (K = 10)

We can see more results for other images in figures 3 and 4.

The algorithm may or may not be deterministic depending upon the choice of parameters. The default start method, *kmeans++*, is stochastic (although it applies a certain heuristic in order to have the initial clusters as far apart from each other as possible). The other named methods (*sample*, *uniform*, *cluster*) are stochastic as well. There is an additional start method which consists in accepting from the user a matrix with the initial clusters' centroids. Should these clusters be decided deterministically the algorithm will perform the same steps at each run. For the purpose of our analysis we will stick to *kmeans++*.

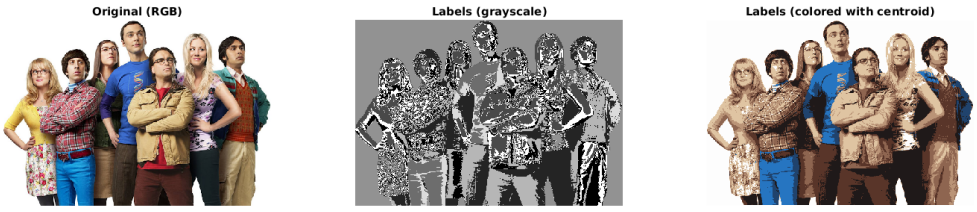


**Figure 3:** Segmentation of `bigbangfamily.png` with K-means (K = 8)

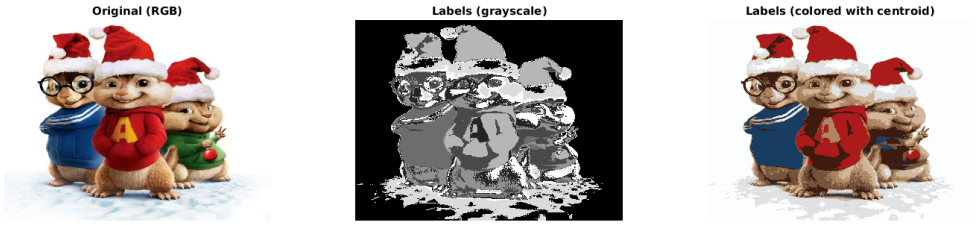


**Figure 4:** Segmentation of `alwin2.jpg` with K-means (K = 8)

We can see how the results can (and will, generally) be different between successive runs of the algorithm with the same arguments and the same image in figure 5.

Of course, when using any of the stochastic methods we can force the results to be the same at each run by fixing the seed of the Random Number Generator (RNG).

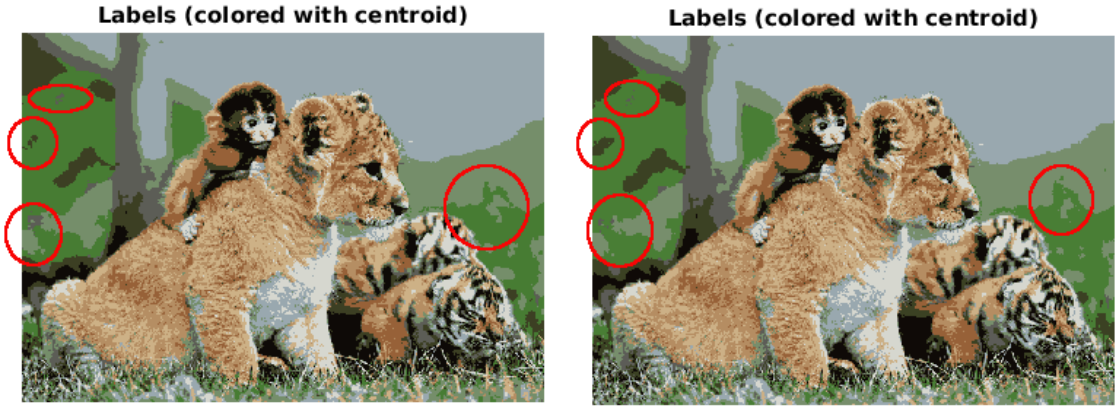


**Figure 5:** Results of successive runs of the k-means algorithm for `animals.jpg` with K=10. The most noticeable differences are marked with a red circle.

In order to check whether the results are the same when applying transformations to the image, and bearing in mind the previous explanation, we need to fix the state of the RNG (otherwise the results would be different no matter there is a transformation or not). Even doing so we can anticipate that we will obtain different results since these transformations have an influence over the initialization method (whenever it is one of the named stochastic methods). In the case of scaling the image this becomes obvious when we take the limit case and scale the image down to just one pixel (e.g. the average of all the pixels in the image). In general scaling down the image removes details and can result in redundant clusters and/or less noisy labels. On the other hand scaling the image up can magnify noisy sections of the image that can fool the algorithm into assigning a cluster to those spurious regions. Finally rotating the image has an impact on the order in which the data

is presented to the algorithm.

In figure 6 we can appreciate that these geometrical transformations have indeed an impact over the outcome of the algorithm. However it is worth noticing that under reasonable circumstances (i.e. no extreme or contrived transformations and not too much noise) the results are fairly similar from a qualitative point of view. Perhaps the most significant change is seen when we scale down the image to 0.25 of the original dimensions (i.e. 16 times less area), since the label assigment to the lion and the grass section is not so noisy.


**(a)** Original


**(b)** Scaled down (s=0.5)


**(c)** Scaled up (s=2)


**(d)** Rotated

**Figure 6:** Comparison of k-means' (K=10) results after applying geometric transformations

## 1.1 Adding spatial information to the k-means algorithm

In figure 7 and 8 we can see what the results for `animals.jpg` when we include the spatial coordinates in the data passed to the k-means algorithm. We can see that now the pixel coordinates has a significant weight when determining the cluster it belongs to. This helps to fight some of the problems we had before: now the labels assigned to the grass section do not seem so erratic and spurious.

On the other hand we can see as a disadvantage that large entities being cut asunder whenever they are too large (like the baby lion in the middle). Not only that, now the size of the image have a great weight over the segmentation process. The bigger the image the more will it resemble a Voronoi diagram rather than an image segmented by colors. Figures 9 and 10 illustrates this even better.

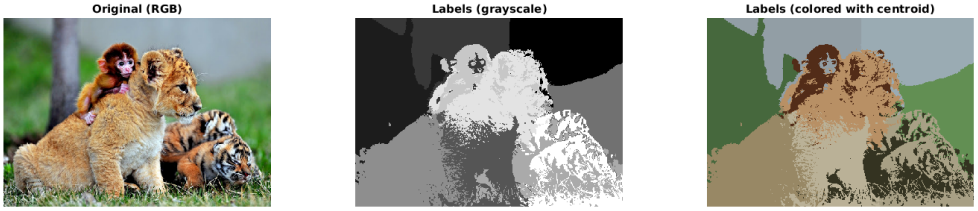


**Figure 7:** Segmentation of `animals.jpg` with K-means (K = 10) and spatial coordinates

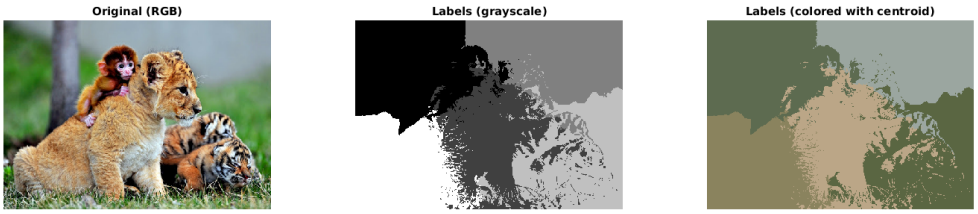


**Figure 8:** Segmentation of `animals.jpg` with K-means (K = 5) and spatial coordinates

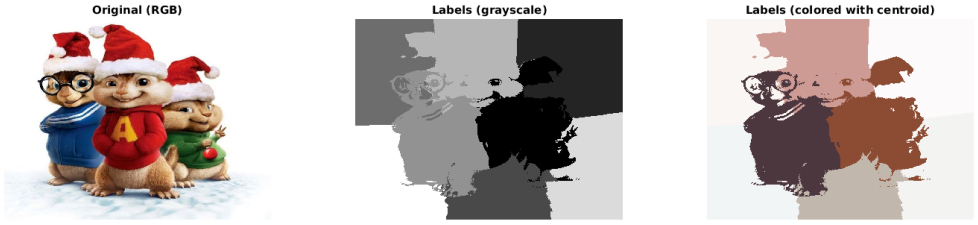


**Figure 9:** Segmentation of `alwin2.jpg` with K-means (K = 8) and spatial coordinates

The spatial coordinates can help to discriminate between two entities that have similar colors but are distant from each other. This means that the number of clusters would ideally be chosen to match the number of expected single-color compact entities in the image, instead of the number of colors. In general this is difficult. Moreover, when using Euclidean distance with k-means we are assuming spherical clusters which in general do not adjust well to the figures in our image (e.g. the elongated shape of the lion).

Now the size of the image has to be chosen well. Scaling down `animals.jpg` before applying k-means with K=10 and spatial coordinates, as shown in figure 11, renders a much better segmentation than that showed in figure 7. One could think that the clusters are even better than in figure 2 since the baby lion is segmented much more uniformly and the grass region presents less noise now that the spatial coordinates are taken into account.

## 1.2 Additional observations

Alternatively, instead of scaling the image down, we could introduce a variant of the Euclidean distance that weights differently the distance in the color space and the geometrical distance. In this same line of thought one could even weight differently the difference of each attribute without caring if it is a color attribute or a spatial one (e.g. giving more importance to the red channel). Other possibilities include using color spaces like CIEL*a*b that take into account the human perception and in which the distances are more meaning-

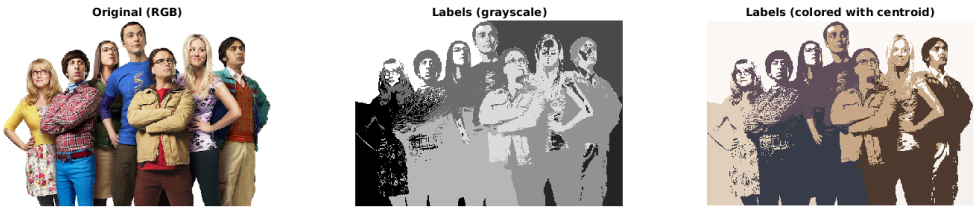


**Figure 10:** Segmentation of `bigbangfamily.png` with K-means (K = 8) and spatial coordinates

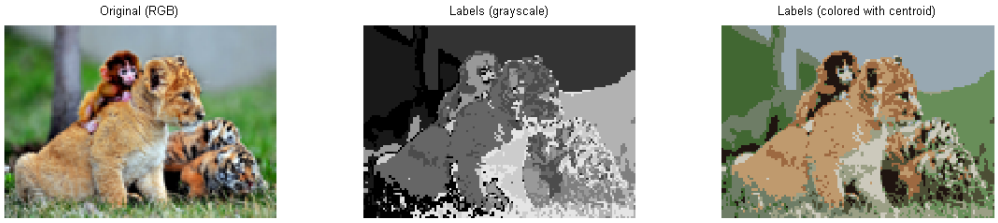


**Figure 11:** Segmentation of `animals.jpg` (scaled down, s=0.25) with K-means (K = 10) and spatial coordinates

ful in a perceptual sense. In this color space the L* channel represents the luminosity and can be thrown away if our purpose is to segment by perceived color minimizing the impact of light changes. All these ideas can be applied to the following algorithm we will cover (Mean-shift) as well, although they will not be covered in this document.

## 2 Mean-shift segmentation

Next we analyze the Mean-shift algorithm. One of its main advantages with respect to k-means is that we do not get to decide the number of clusters. Instead there is only one parameter to tweak: the bandwidth, which decides the size of the neighborhood that is considered to compute the direction towards which the currently selected active point should move. The number of clusters is decided for each image depending on the data and on the bandwidth value. This makes it quite more adaptable to different images. However its main disadvantage became apparent when performing the experiments for this section: it is notably slower than k-means because it starts with as many active points as pixels there are in the image. For the purpose of our experiments we have found that a bandwidth of 30 yields good results in a reasonable amount of time, so this is the employed value in all the examples (unless stated otherwise).

We can see in figure in figure 12 that the segmentation is quite good, segmenting the baby lion almost as a whole. One inconvenient is that the tigers and part of the primate are included in the same cluster, which is reasonable given than they have very similar colors and we are using a segmentation technique that is purely driven by color information. Another disadvantage is that since we are not including spatial information, there is low uniformity in the grass section. However if our purpose is to separate the foreground from the background, Mean-shift is an effective technique to do so.

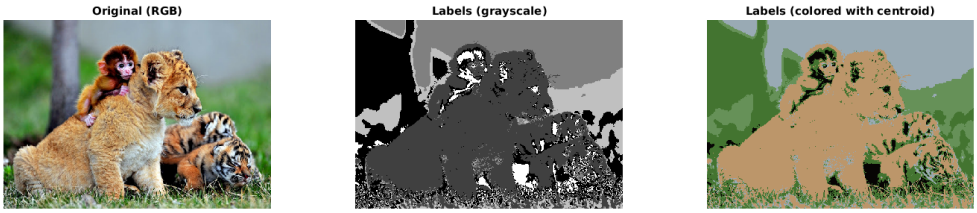


**Figure 12:** Segmentation of `animals.jpg` with Mean-shift

Other examples of segmentation with Mean-shift are shown in figures 13 and in 14. We can appreciate a good segmentation quality in general. There are some effects that could possibly be avoided with a lower bandwidth, like some pieces of clothing being assigned to the same cluster as the skin and the cheeks of the squirrels being assigned to the same cluster as the background.

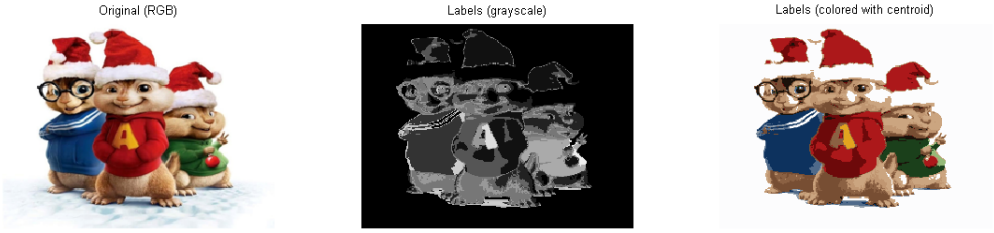


**Figure 13:** Segmentation of `alwin2.jpg` with Mean-shift


**Figure 14:** Segmentation of `bigbangfamily.png` with Mean-shift

The algorithm is not deterministic because at each update step the next point to update (i.e. move towards the center of mass of its neighbourhood) is selected randomly. This is done in the line 47 of the provided `MeanShiftCluster.m` file (`tempInd = ceil((numInitPts-1e-6)*rand);`). This also means that two executions of the Mean-shift algorithm do not yield necessarily the same value (unless the seed of the random generator is fixed). This can be observed in figure 15

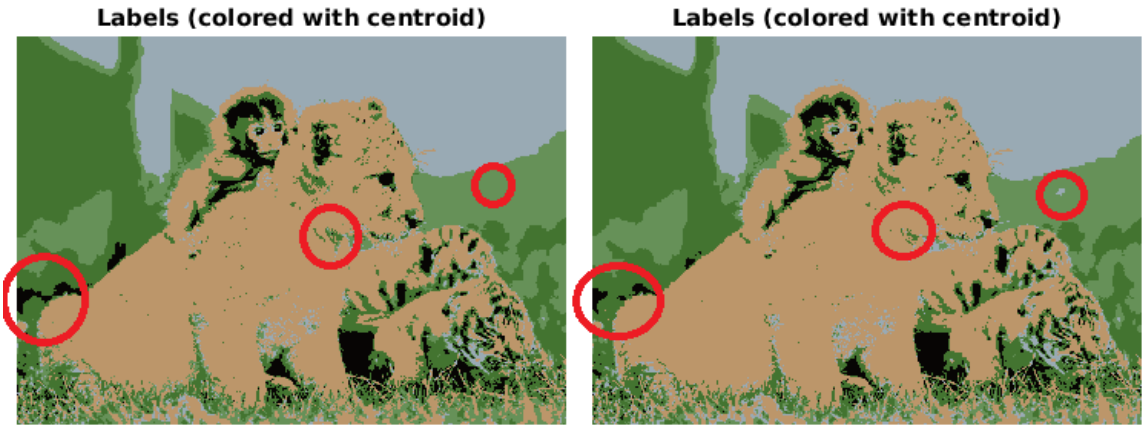


**Figure 15:** Results of two successive runs of Mean-shift. The most noticeable differences are marked with a red circle.

Much like with k-means, geometric transformations lead to different results. This is observed in figures 16 and 17, because the order in which the active points are picked to be shifted is dependent on the structure of the data passed to the algorithm.

It is also interesting to observe the effect of the bandwidth on the segmentation. We

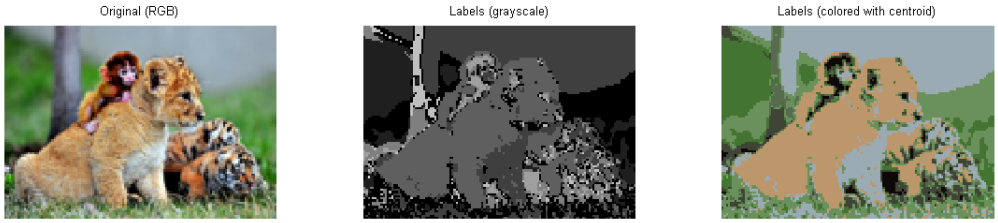


**Figure 16:** Segmentation of `animals.jpg` (scaled down, s=0.25) with Mean-shift

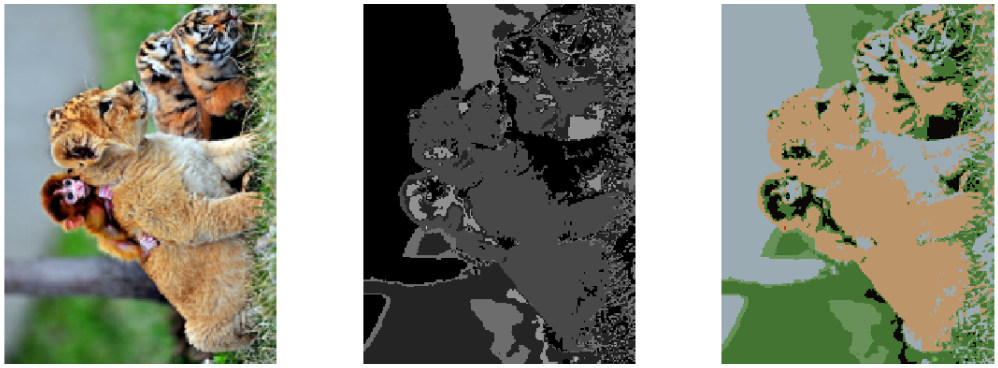

**Figure 17:** Segmentation of `animals.jpg` (rotated) with Mean-shift

expect that a high bandwidth leads to grouping more active points in the same cluster, reducing the total number of clusters. On the other hand a higher bandwidth helps to reduce the number of active points earlier in the algorithm, which means that the runtime is lower. Therefore the bandwidth should be increased either to fight over-segmentation problems. To illustrate this explanation we provide figure 18 which shows a result generated with a bandwidth of 50 and has less clusters than the segmentation showed in figure 14. Figure 19 is another example that in addition shows how the bandwidth can be used to produce a somewhat less noisy segmentation (compare it to figure 12).

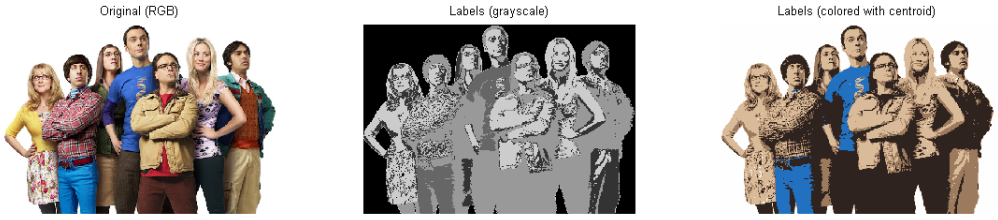


**Figure 18:** Segmentation of `bigbangfamily.png` with Mean-shift. BW = 50

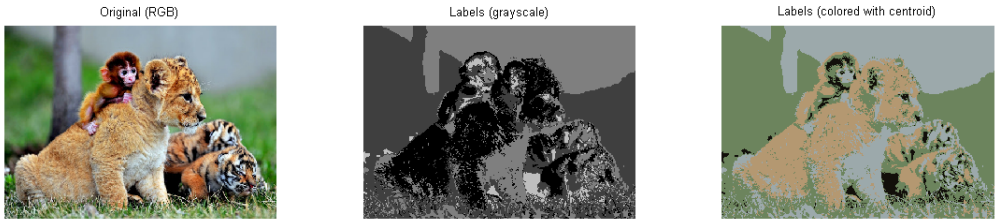


**Figure 19:** Segmentation of `animals.jpg` with Mean-shift. BW = 50

## 2.1 Adding spatial information to the Mean-shift segmentation

Finally we discuss the effect of adding spatial information to the data passed to the Mean-shift algorithm. In figure 20 we can see the `animals.jpg` scaled down image segmented with spatial coordinates and a bandwidth of 30 (for this section we will used either scaled down images or high bandwidths in order to produce results in a reasonable amount of time). Much like in k-means, the effect of considering the spatial coordinates is obtaining smoother regions but with the possibility of dividing large objects. More examples are shown in figures 21 and 22. The lion of figure 23 is a particularly good example of how a big entity is cut asunder into several regions.

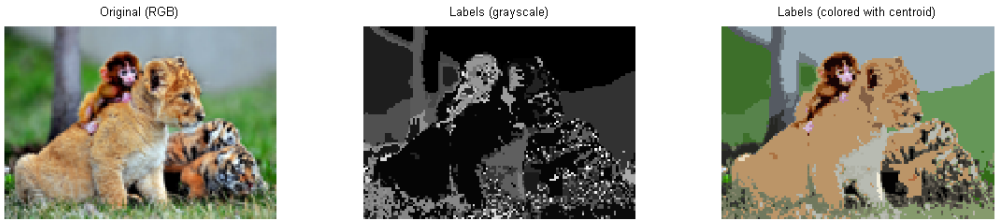


**Figure 20:** Segmentation of scaled down (s=0.5) `animals.jpg` with Mean-shift and spatial information.

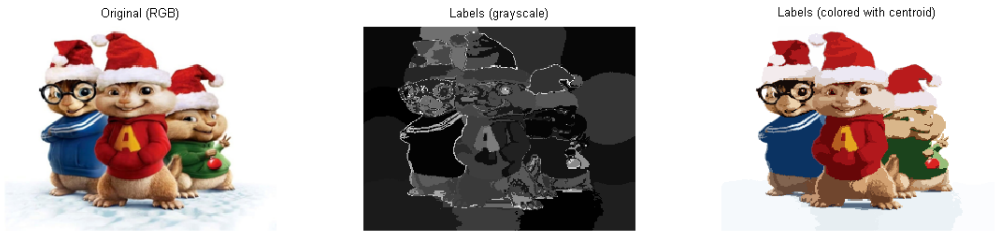


**Figure 21:** Segmentation of scaled down (s=0.25) `alwin2.jpg` with Mean-shift and spatial information.


**Figure 22:** Segmentation of scaled down (s=0.5) `bigbangfamily.png` with Mean-shift and spatial information.

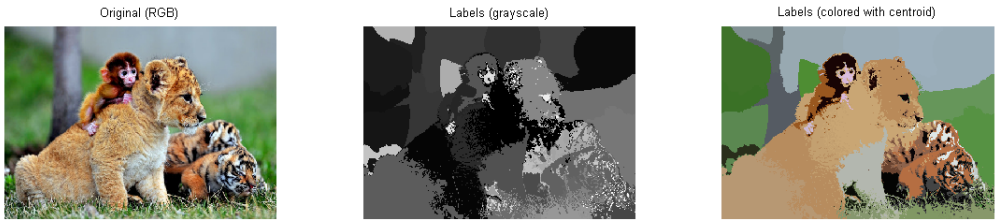


**Figure 23:** Segmentation of full-size `animals.jpg` with Mean-shift and spatial information. BW=50

# Appendices

## A Annex

Here we list all the delivered script and function files delivered for this practice.

We include relevant observations. For full insight, we refer the reader to the source code.

- `show_results.m`: a method that takes a RGB image, an array of labels (one per each pixel of the image) and the centroids associated to each cluster (sorted from lowest label to greatest). It plots the original image, the labels with a grayscale colormap and the labels colored with the mean color of the cluster.
- `kmeans_exercise.m`: script that executes the k-means algorithm once and plots the results. We have used this script to generate the plots of the first section of this document.
- `mshift_exercise.m`: idem for Mean-shift algorithm