CSCI 631 Homework 2

1) First, we need to import 48 filters from python file makeLMfilters.py. After we convert the original image to grayscale, we need to convolve the grayscale with each filter. We can store the result in the multi-dimensional array. Here comes the most important part, we should use an implementation from an existing library such as scikit-learn or use my own version. Here I choose to use them both to get k clusters. Finally, we reshape the clustering result into the dimensions of the input image and try different k values to see the difference.

   Advantage of k-means algorithm:

   a) Relatively simple to implement
   b) Scales to large data sets
   c) Guarantees convergence
   d) Can warm-start the positions of centroids
   e) Easily adapts to new examples
   f) Generalized to clusters of different shapes and sizes, such as elliptical clusters

   Disadvantage of k-means algorithm:

   a) Choosing k manually
   b) Being dependent on initial values
   c) Clustering data of varying sizes and density
   d) Clustering outliers
   e) Scaling with number of dimensions

2) I implemented my own version of k-means clustering.

```
'****** my own kmeans ******'
idx = myOwnKmeans('gecko.jpg', 4)
trImg = transferImg([0,1,2,3], idx, 'gecko.jpg', 'bg.jpg')
'****** my own kmeans ******'
```

   You need to call myOwnKmeans function instead of calling segmentImg function.

   If you do this way as the image shown above, my own version of k-means clustering can run normally. The parameter responses of mykmeans function is the result of 48 filters convolve with grayscale. If we use rows, cols=grayscale.shape , then we can get the rows and cols parameters. K is the number of clusters in k-means algorithm.

   Mykmeans function will call the functions as followed:

   eulid(vec1, vec2)

   initCentroids(dataset, k)

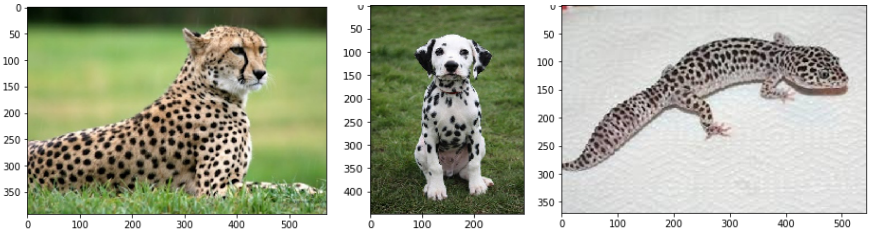
   minDistance(dataset, centroidList, mydic)

   getVar(centroidList, clusterDict)

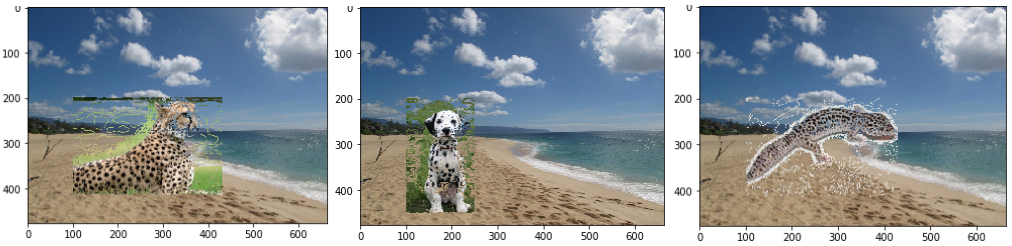
   getCentroids(clusterDict).

3)

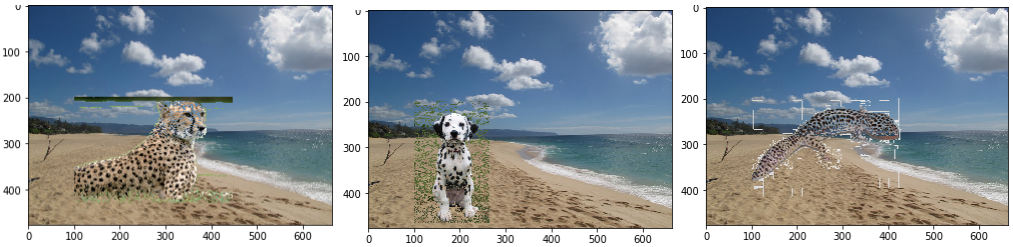
Original Image



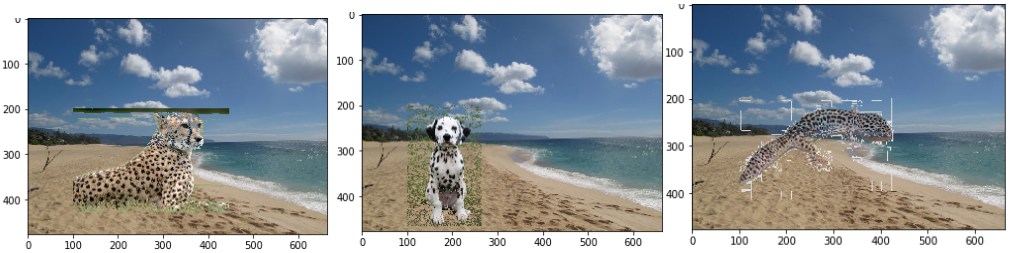
Texture Filter



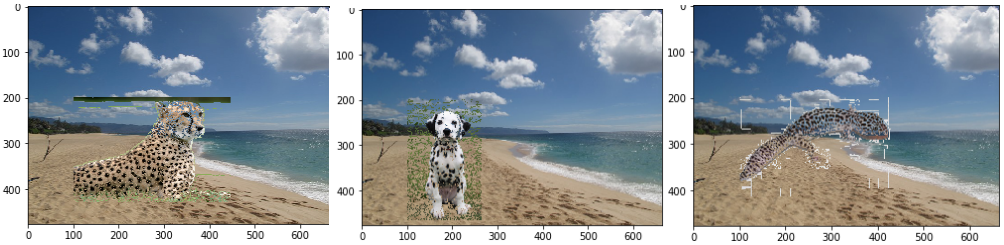
HSV Filter



RGB Filter



LAB Filter



LAB space expresses color as three numerical values. If we use LAB color space, it is perceptually meaningful. Distances in the color space correspond to similar quantities of visually perceived change. HSV is also perceptually meaningful, it has three dimensions: hue, saturation and value. In this homework, the animals have distinct colors. So it is relatively easier to segment the original image, and then transfer it to the background.

Scale-space blob detector can convolve image with scale-normalized Laplacian at several scales.

Find maxima of squared Laplacian response in scale-space. But since Laplacian has strong response along edge, we need to be careful that edge responses need to be eliminated. For this homework, since the animal images have strong edge responses, the final result is less perfect than color space method.

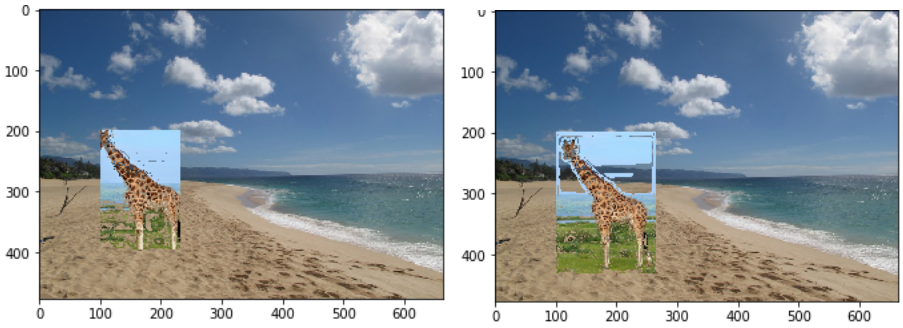
4) The reason why many segmentations have holes is that if our original image doesn't have clear textures such as spots or stripe, our program may treat this part of image as background or foreground. For this case, if we adopt color filters such as LAB, RGB or HSV filters, the quality of image might be improved. Or if we consider combining texture filter with color filter together, then we can get a better result.

   Also, in the given code, I have found an issue here. If we use giraffe.jpg image, the outcome tends to be smaller than the original size. Part of giraffe's head is missing. It is because of the transferImg function.

```
# Crop the source and indexed images
idxImg = idxImg[25:rows-25, 25:cols-25]
sImg = sImg[25:rows-25, 25:cols-25]
rows, cols, clrs = sImg.shape
```

   If we remove the above part, then the problem will be solved.

   Please see the images as followed:



   The image which is on the left loses half of giraffe's head. On the contrary, the image which is on the right is the normal size.