CSCI 5722 Assignment 4

Name: Ajay Kumar Kedia
Identity Key: ajke2555

Solution 6

a) In this project, I tried multiple feature extraction

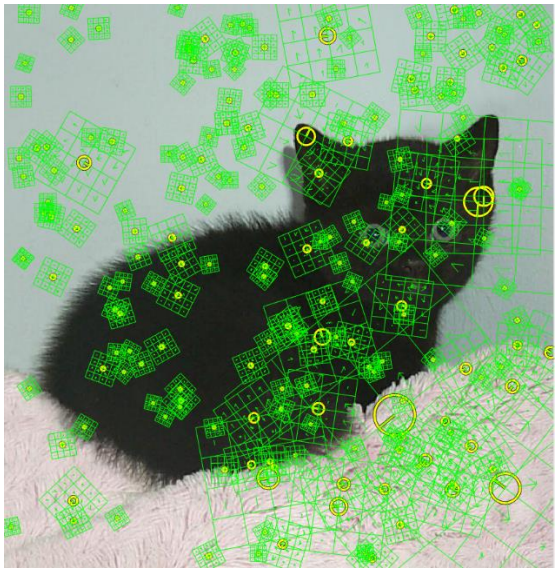
1) Color Features
2) Color and Position Features
3) Gradient Features
4) Edge Detection Features
5) VL SIFT Features

Let's take these one by one.

1) **Color Features:** In this feature, I computed the color of all the pixels in my image. This feature helps to identify the foreground from the image and separate the foreground and background of the image. Generally, if you remove the saturation of the image by removing the color intensity by taking away using double function on the image, it makes the image more inclined towards the separation from its background. Accuracy = 87.10% with 5 clusters and Resize = 1 and Normalization = true
2) **Color and Position Features:** In this feature, I was calculating the color and position of the pixel in the image. So, for each pixel, I was computing a feature vector with r, g, b, x and y where (r, g, b) is the color of the pixel and (x, y) is the position of the pixel in the image. It also helps in increasing the accuracy in K-means clustering and HAC clustering. Because based on the position and color feature, it helps me to separate its foreground from its background in the image. It didn't boost our accuracy so much, but it decreased it by 1% which is not as good as color feature. Some of the images overfitted that's why I got less accuracy. I created features in 5 dimension whose size is equal to original image and then in first three dimension, I stored the image after removing the saturation and rest two dimension, I stored its position by playing with width and height. In one feature, I fixed the width as image height and took height from 1 to image width as the feature vector's height. And, in second feature, I took complement of my image height and stored it in my feature vector by fixing its width to 1. Accuracy = 85.99% with 5 clusters and Resize = 1 and Normalization = true
3) **Gradient Features:** In this feature, I was calculating the gradient of the image that how fast its dx and dy changing based on the middle pixel in the section of the box. I started with 3x3 box and find out the change in the gradient and then find out the direction of it's changed. Based on that, I made feature using its magnitude and direction of change. I didn't get the much accuracy when I tried to run it on using HAC cluster with 0.01 resize image with 5 cluster, it came out 74.63% because I resized the image to 0.01 but when I increased the resized the image to 0.1, the accuracy boosted up and it came out 86.32 and if I keep on increasing the size of cluster and image resize, the accuracy will keep on increasing because it will have more intense pixel rather than compressing the pixels which changing its gradient ratios.
4) **Edge Detection Features**: In this feature, I was detecting the edge of the foreground in the image. My intention was to find out the edge so that I can separate it from it background. As we know

that, background is not having any sharp turn. They are static where as foreground image has sharp edges, and which can help me to separate it from it's background. I was trying different features like Canny, Prewitt, Sobel, Roberts and approxcanny to detect the images from its background. The edge works out very well, if I have difference in the contrast of foreground and background. If both foreground and background of the same intensity, then it will not detect the feature very well and come out with noises and which reduced my final accuracy. I played with multiple images of different intensity and the high contrast images win the race and turn out a good feature but not as good like other features, in detecting the foreground from the image. But when I run it with all the images, it gave me bad accuracy and it didn't help much to separate most of the images foreground. Accuracy = 70.41% with 5 clusters and resize = 1 and Normalization = true.

5) **VL SIFT Features:** In SIFT features, we have a circular orientation which describes the geometric frame of 4 parameters: Center coordinates X and Y, its scale (the radius of the region) and its orientation (an angle expressed in radians). The SIFT detector uses blobs which resample to image structures. By searching for blobs at multiple scales and positions, the SIFT detector is invariant (or, more accurately, covariant) to translation, rotation and re-scaling of the image. By using the features, I find out the orientation and scale of the image with different permutation of the features. So, vl_sift in MATLAB, gives us two things, one is feature and one is the descriptor. Both has the same height (width x height) but different width. I walked through the features in vl_sift and find out that it's width is 4 which has position as (x, y), orientation and scale. Using the position, I created features using orientation and scales at different dimension and run our K-means and HAC cluster. Accuracy = 86.21% with 5 clusters and resize = 1 and Normalization = true. One snapshot of the image using VLSIFT feature



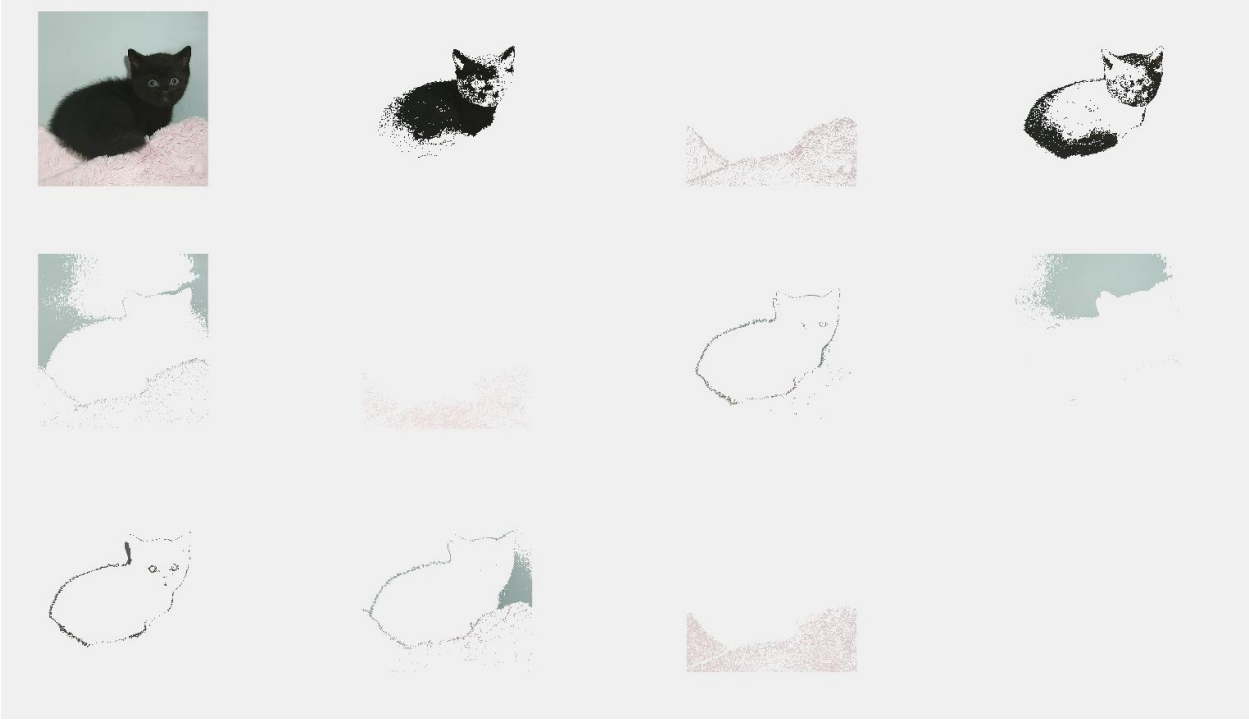
As we can see in the image, that the orientation feature is telling us that in which direction we have to move to get foreground and its separating the foreground from the background if we move in the direction of the arrow. In this image, I worked on 200 features with different permutation to come out with this result. If you zoom out this image, then you will see that the scale and orientation will be a good feature to separate both, foreground and background.

6) **Morphological Openings Features:** In Morphological openings features, I was trying to binarize the image which estimates the background illumination. Morphological opening is an erosion followed by a dilation, using the same structuring element for both operations. The opening operation has the effect of removing objects that cannot completely contain the structuring element. So, I first convert my image in binary format then to get the background, I call the strel function to create a disk-shaped structuring element with a radius of 10 and calls the imerode to compliment the background image and fill the hole based on the disk size or to remove the holes from the image, I can also use imfill operation and fill the background image. Then, I passed this image as feature. I used this feature with VL SIFT and got Accuracy = 96.21% with 5 clusters and resize = 1 and Normalization = true.

Solution Part b)

So, I tried with different segmentation with different clusters. Here are the images of successful and unsuccessful segmentation with different features.

1) Black_kitten_star.jpg



I used this segmentation on my background image and try to mask this image and the final result I got is given below

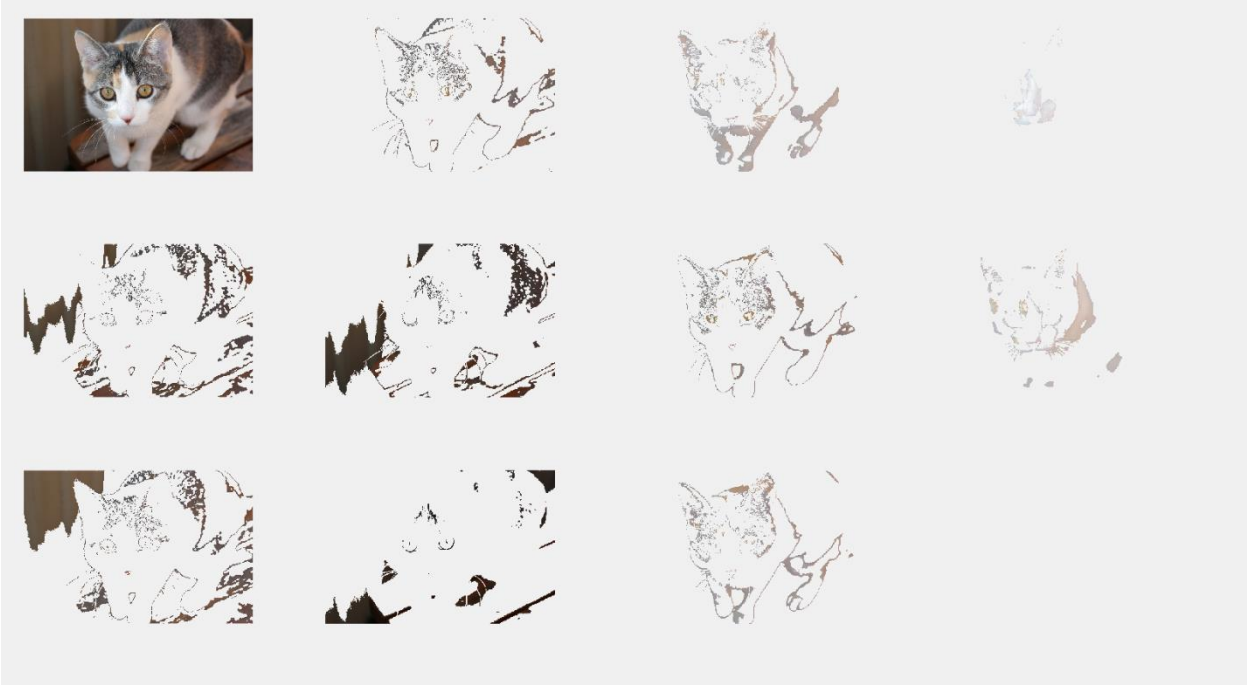
Parameters:
Clustering: K-means
Normalization: True
Resize = 1
K = 10
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 98.73%

2) Black_kitten.jpg

I used this segmentation on my background image and try to mask this image and the final result I got is given below

Parameters:
Clustering: HAC
Normalization: True
Resize = 0.1
K = 5
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 92.31%

3) tortoiseshell_shell_cat.jpg



I used this segmentation on my background image and try to mask this image and the final result I got is given below

Parameters:
Clustering: K-means
Normalization: True
Resize = 0.8
K = 50
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 89.96%

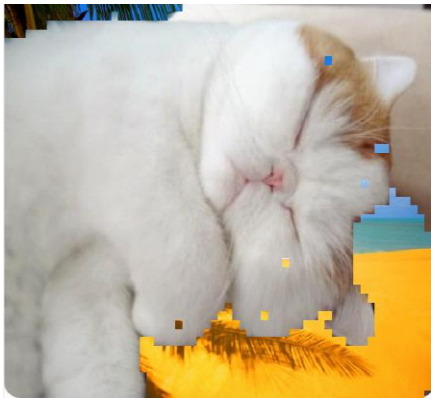
4) black-white-kittens2.jpg

I used this segmentation on my background image and try to mask this image and the final result I got is given below

Parameters:
Clustering: HAC
Normalization: True
Resize = 0.1
K = 10
Features: ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 64.34%

5) cutest-cat-ever-snoopy-sleeping.jpg

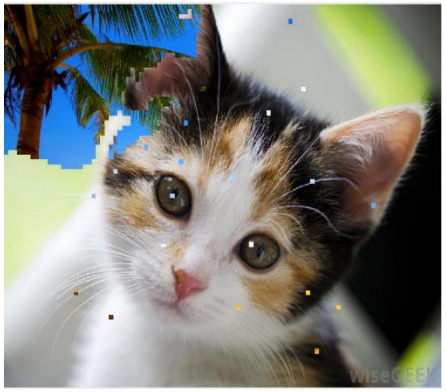
I used this segmentation on my background image and try to mask this image and the final result I got is given below



Parameters:
Clustering: K-means
Normalization = True
Resize = 1
K = 40
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 65.22%

6) young-calico-cat.jpg

I used this segmentation on my background image and try to mask this image and the final result I got is given below



Parameters:
Clustering: K-means
Normalization: True
Resize = 0.8
K = 15
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 59.32%

So, from the above examples, 1-3 are successful and gave me the good accuracy and most accurately separated the foreground from its background and did the masking on another background images where as image from 4-6 were very bad in separating the foreground from its background as most of the pictures is having background attached to it in the final result. I tried with different parameters but couldn't remove the pieces of background from the image and hence, we got unsuccessful masking in the result images.

Well, from the above 6 examples (3 Successful, 3 Unsuccessful), one thing I noted the contrast between the images. As first 3 examples are having solid contrast between its foreground and background part and hence our algorithms were helpful in separating foreground from its background while in last 3 images, it was not the case, white cat is mostly looks similar to its background part in terms of intensity, texture and color and hence our features couldn't work out very well to turn the features ups and down and hence, we got poor masking in the result.

Solution c)

1) Effect of Segmentation Parameters
    a. **Feature Transform:**
    Different Feature transforms played a different role when we are trying to separate foreground from the background image. In my solution 1, I provided you different accuracy for different features.
        1) Color Features: Accuracy = 87.10% with 5 clusters and Resize = 1 and Normalization = true
        2) Color Position Features: Accuracy = 85.99% with 5 clusters and Resize = 1 and Normalization = true
        3) Gradient Features: Accuracy = 74.63% with 5 clusters and Resize = 0.01 and Normalization = True
        Accuracy = 86.32% with 10 clusters and Resize = 0.1 and Normalization = True
        4) Edge Detection Features: Accuracy = 70.41% with 5 clusters and resize = 1 and Normalization = true
        5) VL SIFT Features: Accuracy = 86.21% with 5 clusters and resize = 1 and Normalization = true
        6) Combined all Features: Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means
        Accuracy = 86.14% with 10 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC
        Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means
    b. **Feature Normalization:**
    Feature Normalization also played a good role to differentiate foreground and make it noisy free. I tried making it true and false and recorded its behavior. So while using normalization as True, I got better accuracy on my segmentation and without normalization, I got less accuracy compared to when it's true.
    Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means
    Accuracy = 82.23% with 10 clusters and resize = 1 and Normalization = false and Clustering Method = K-means
    c. **Number of Clusters:**
    Number of clusters also played a good role to differentiate foreground from its background but when you increase the cluster, process became slow as its has to divided out parameters in different number of cluster, so speed was inversely proportional to the

number of clusters, but accuracy was directly proportional to the number of cluster. As the number of clusters were increasing, my accuracy of segmentation was kept on increasing.

Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 86.14% with 10 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

d. Clustering Method:

In **HAC**, there are two different approaches that fall under this name: top-down and bottom-up. In top-down hierarchical clustering, we divide the data into 2 clusters (using K-means with K=2). Then, for each cluster, we can repeat this process, until all the clusters are too small or too similar for further clustering to make sense, or until we reach a preset number of clusters.

In bottom-up hierarchical clustering, we start with each data item having its own cluster. We then look for the two items that are most similar and combine them in a larger cluster. We keep repeating until all the clusters we have left are too dissimilar to be gathered together, or until we reach a preset number of clusters.

In **K-means** clustering, we try to identify the best way to divide the data into K sets simultaneously. A good approach is to take K items from the data set as initial cluster representatives, assign all items to the cluster whose representative is closest, and then calculate the cluster mean as the new representative, until it converges (all clusters stay the same).

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 92.98% with 50 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC

I can't check it on resize = 1 because the array size was too huge, and it was throw memory segmentation error. But, if we have more RAM, then HAC will give better result than K-means.

e. Resize:

Yes, resize was also playing a good role because if I resized the image to 0.001 then I will get the bad accuracy as it's very hard to play with pixel because we have very less number of pixel to compare and that's why bad accuracy. If we keep on increasing the resize for some limit, we will get better accuracy but after some point of time, our accuracy starts decreasing because we can't make a 150 KB image to convert it into 10MB image as it will lose its intensity and texture and hence, will get a bad accuracy in separating the foreground from its background.

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 91.63% with 50 clusters and resize = 0.1 and Normalization = true and Clustering Method = K-means

Accuracy = 82.74% with 50 clusters and resize = 0.01 and Normalization = true and Clustering Method = K-means

2) Speed of Computing Segmentation
    a. **Feature Transform:** As the number of features increased, speed of computing started to decrease because now it will take time to make features based on parameters passed and computation of our process becomes slow. More memory is used, and CPU kept on playing with virtual memory.
    Increase of Feature Transform is inversely proportional to speed of computing
    b. **Feature Normalization:** Normalization slow down our speed of computing but gives better result. As our CPU is busy to do extracting the noises from the images which slow down the computing power of our system and makes the process slow.
    Normalization is inversely proportional to the speed of computing
    c. Number of Clusters: As the number of clusters increase speed of computation of foreground from the image goes down because now we have to extract features all the clusters and hence speed of computing slow down.
    Number of clusters inversely proportional to the speed of computing.
    d. **Clustering Method:** In **HAC** is quite slow then K-means because for each cluster, we can repeat this process, until all the clusters are too small or too similar for further clustering to make sense, or until we reach a preset number of clusters where in **K-means** clustering, we try to identify the best way to divide the data into K sets simultaneously. So, the speed of computing is slow in HAC compared to K-means.
    e. **Resize:** In Resize, speed of computing increases as the resize decreases.
    Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means
    Accuracy = 91.63% with 50 clusters and resize = 0.1 and Normalization = true and Clustering Method = K-means
    Accuracy = 82.74% with 50 clusters and resize = 0.01 and Normalization = true and Clustering Method = K-means
    So, decrease in resize increase the speed of computation in separating of our foreground from its background

3) Properties of an Image:
    a. **Contrast of the Image:** As the contrast difference between foreground and background is more, it's easy to get a good segmentation of the image. Example is given in solution 2. A good contrast is helpful to detect the edges and VL Sift of the image and also help in finding out the gradient of the image as good as possible and boost our feature extraction.
    b. **Texture of the Image:** A good texture image is easy to separate its foreground from its background. Edge detection feature will extremely play a good role in this case and increase our accuracy by many folds. On a good texture, we apply a threshold value (Apply different threshold value until or unless we can separate the foreground from the background) and then apply edge detection to separate both.
    c. Color of the Image: Color of the pixel also play a good role in separating the foreground from its background image and hence, gives us a good segmenting image. Remove the saturation of the image and then use this image as a feature extraction will help in differentiating its foreground and background.

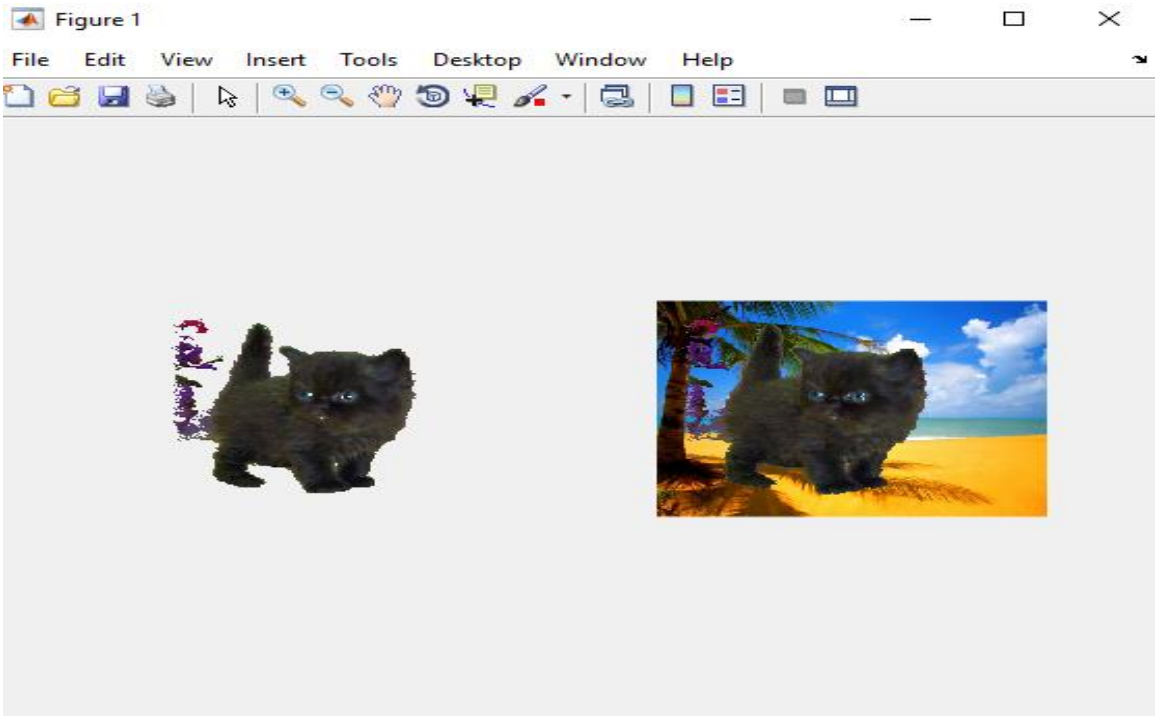
d. **Position of the pixel:** Combination of color and position can play a role in extracting its foreground from its background but alone as a feature, there is no sense to use it. It doesn't give any information how foreground is separated from its background because the image is just a matrix with numbers and the numbers don't give any information about the foreground. So, its hard to detect it from the image.

Solution d)

Let's take example one by one:
Example 1)
Input Image = black_kitten.jpg

Background = beach.jpg

Output Image:



Parameters Used:

Clustering: K-means
Normalization: True

Resize = 0.1
K = 5
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 91.29%

So, I applied K-means clustering and used above parameters to separate the image of cat from its background and then transfer it to another background. I also applied the HAC algorithm which you can see in solution 2 where I used

Parameters:
Clustering: HAC
Normalization: True
Resize = 0.1
K = 5
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 92.31%

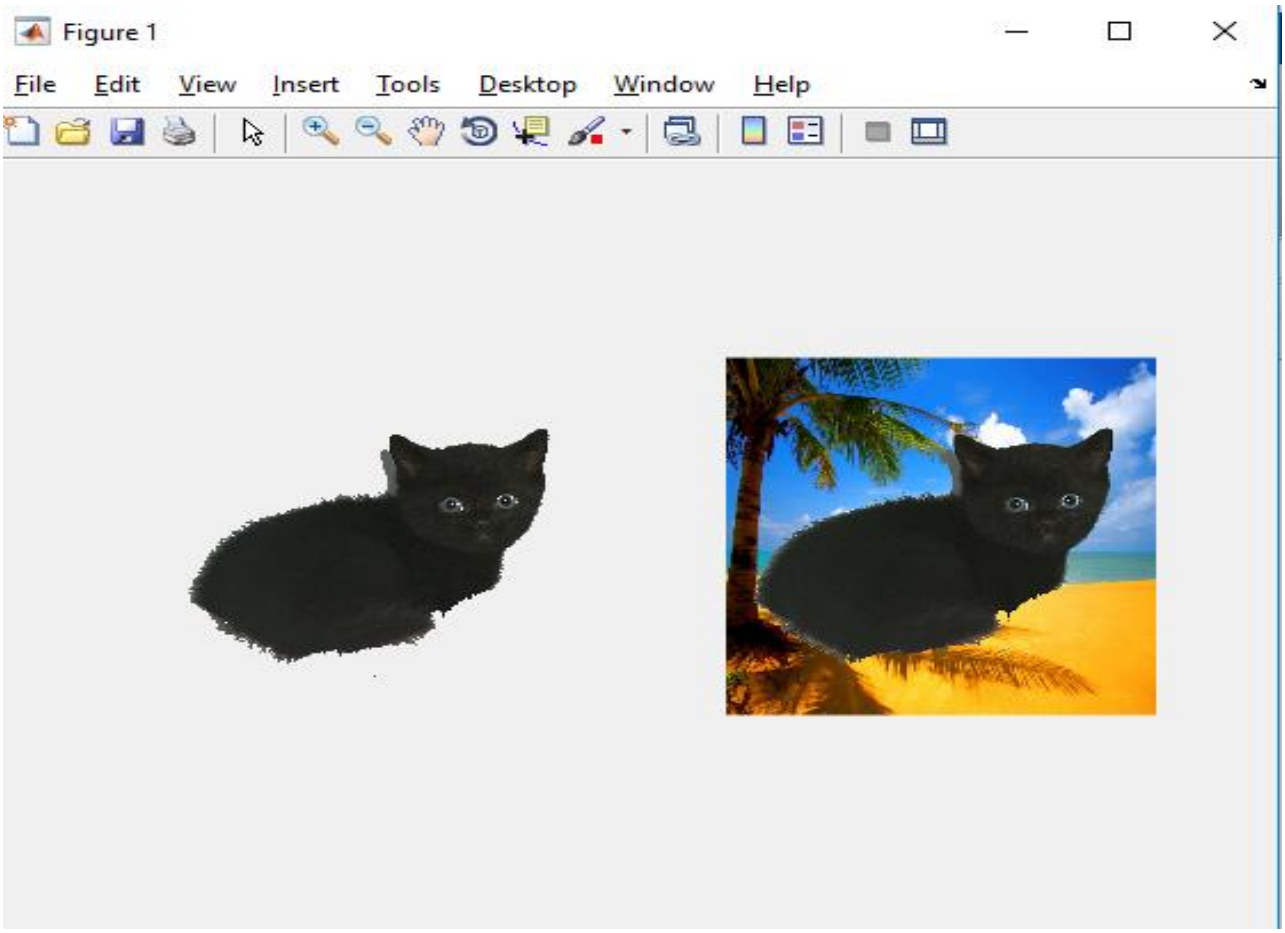
So, both accuracy turns out different, as K-means affected by K and Resize value because HAC is quite slow if we increase both the value.

Example 2)

Input Image = black_kitten.jpg

Background = beach.jpg

Output Image:



Parameters Used:

Clustering: K-means
Normalization: True
Resize = 1
K = 10
Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 98.73%

So, I applied K-means clustering and used above parameters to separate the image of cat from its background and then transfer it to another background. I also applied the HAC algorithm

Parameters Used:

Clustering: HAC
Normalization: True
Resize = 0.1
K = 5

Features = ComputeFeatures (consists of Compute Color Position Feature, Compute Color Feature, Compute Edge Detection Feature, Compute VL SIFT Feature, Compute Gradient Feature)
Accuracy = 95.71%

So, both accuracy turns out different, as K-means gave me the best show than HAC because in HAC, we are using Resize = 0.1 and K = 5 for the same features and it was quite slow if we increase both the value.

Solution e)

Feature Transform	Feature Normalization	Clustering Method	Number of Clusters	Resize (or Max Pixels)	Mean Accuracy
Color	N	KMEANS	5	1	0.8321
Color	N	KMEANS	4	0.8	0.8260
Color	N	HAC	3	0.1	0.8013
Color	Y	KMEANS	5	1	0.8710
Color	Y	KMEANS	4	0.8	0.8598
Color	Y	HAC	3	0.1	0.8420
Color Position	N	KMEANS	5	1	0.8476
Color Position	N	KMEANS	4	0.8	0.8410
Color Position	N	HAC	4	0.1	0.8426
Color Position	Y	KMEANS	50	1	0.9324
Color Position	Y	KMEANS	40	1	0.9193
Color Position	Y	HAC	40	0.1	0.9220
Gradient	N	KMEANS	5	1	0.8312
Gradient	N	KMEANS	4	1	0.8419
Gradient	N	HAC	3	0.1	0.8290
Gradient	Y	KMEANS	5	1	0.8432
Gradient	Y	KMEANS	4	1	0.8399
Gradient	Y	HAC	3	0.1	0.8294
Edge	N	KMEANS	5	1	0.6841
Edge	N	KMEANS	4	1	0.6792
Edge	N	HAC	3	0.1	0.6733
Edge	Y	KMEANS	5	1	0.7041
Edge	Y	KMEANS	4	1	0.7012
Edge	Y	HAC	3	0.1	0.7003
VL SIFT	N	KMEANS	5	1	0.8481
VL SIFT	N	KMEANS	4	1	0.8312
VL SIFT	N	HAC	3	0.1	0.8345
VL SIFT	Y	KMEANS	5	1	0.8621
VL SIFT	Y	KMEANS	4	1	0.8511
VL SIFT	Y	HAC	3	0.1	0.8501
Combined	Y	KMEANS	50	1	0.9374
Combined	Y	HAC	50	0.1	0.9293
Morphological	Y	KMEANS	10	1	0.8992
Morphological	Y	HAC	5	0.1	0.8803

Solution Part e.1) I have already explained it in part C

a. **Feature Transform:**

Different Feature transforms played a different role when we are trying to separate foreground from the background image. In my solution 1, I provided you different accuracy for different features.

7) Color Features: Accuracy = 87.10% with 5 clusters and Resize = 1 and Normalization = true
8) Color Position Features: Accuracy = 85.99% with 5 clusters and Resize = 1 and Normalization = true
9) Gradient Features: Accuracy = 74.63% with 5 clusters and Resize = 0.01 and Normalization = True
   Accuracy = 86.32% with 10 clusters and Resize = 0.1 and Normalization = True
10) Edge Detection Features: Accuracy = 70.41% with 5 clusters and resize = 1 and Normalization = true
11) VL SIFT Features: Accuracy = 86.21% with 5 clusters and resize = 1 and Normalization = true
12) Combined all Features: Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means
    Accuracy = 86.14% with 10 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC
    Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

b. **Feature Normalization:**

Feature Normalization also played a good role to differentiate foreground and make it noisy free. I tried making it true and false and recorded its behavior. So while using normalization as True, I got better accuracy on my segmentation and without normalization, I got less accuracy compared to when it's true.

Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 82.23% with 10 clusters and resize = 1 and Normalization = false and Clustering Method = K-means

c. **Number of Clusters:**

Number of clusters also played a good role to differentiate foreground from its background but when you increase the cluster, process became slow as its has to divided out parameters in different number of cluster, so speed was inversely proportional to the number of clusters, but accuracy was directly proportional to the number of cluster. As the number of clusters were increasing, my accuracy of segmentation was kept on increasing.

Accuracy = 87.23% with 10 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 86.14% with 10 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

d. Clustering Method:

In **HAC**, there are two different approaches that fall under this name: top-down and bottom-up. In top-down hierarchical clustering, we divide the data into 2 clusters (using K-means with K=2). Then, for each cluster, we can repeat this process, until all the clusters are too small or too similar for further clustering to make sense, or until we reach a preset number of clusters.

In bottom-up hierarchical clustering, we start with each data item having its own cluster. We then look for the two items that are most similar and combine them in a larger cluster. We keep repeating until all the clusters we have left are too dissimilar to be gathered together, or until we reach a preset number of clusters.

In **K-means** clustering, we try to identify the best way to divide the data into K sets simultaneously. A good approach is to take K items from the data set as initial cluster representatives, assign all items to the cluster whose representative is closest, and then calculate the cluster mean as the new representative, until it converges (all clusters stay the same).

I can't check it on resize = 1 because the array size was too huge, and it was throw memory segmentation error. But, if we have more RAM, then HAC will give better result than K-means. K-means is linear in the number of data objects i.e. O(n), where *n* is the number of data objects. The time complexity of most of the hierarchical clustering algorithms is quadratic i.e. $O(n^2)$. As the number of records increase the performance of hierarchical algorithm goes decreasing and time for execution increased. K-mean algorithm also increases its time of execution but as compared to hierarchical algorithm its performance is better. Hierarchical algorithm shows more quality as compared to k-mean algorithm. As a general conclusion, k-mean algorithm is good for large dataset and hierarchical is good for small datasets. Comparison between these algorithms can be implemented on the basis of normalization, by taking normalized and un-normalized data will give different results.

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 92.98% with 50 clusters and resize = 0.1 and Normalization = true and Clustering Method = HAC

e. **Resize:**

Yes, resize was also playing a good role because if I resized the image to 0.001 then I will get the bad accuracy as it's very hard to play with pixel because we have very less number of pixel to compare and that's why bad accuracy. If we keep on increasing the resize for some limit, we will get better accuracy but after some point of time, our accuracy starts decreasing because we can't make a 150 KB image to convert it into 10MB image as it will lose its intensity and texture and hence, will get a bad accuracy in separating the foreground from its background.

Accuracy = 93.74% with 50 clusters and resize = 1 and Normalization = true and Clustering Method = K-means

Accuracy = 91.63% with 50 clusters and resize = 0.1 and Normalization = true and Clustering Method = K-means
Accuracy = 82.74% with 50 clusters and resize = 0.01 and Normalization = true and Clustering Method = K-means.

Solution Part e.2) Yes, there are some images which are hard to separate from their background. I have already given this solution in Part C.

a. **Contrast of the Image:** As the contrast difference between foreground and background is more, it's easy to get a good segmentation of the image. Example is given in solution 2. A good contrast is helpful to detect the edges and VL Sift of the image and help in finding out the gradient of the image as good as possible and boost our feature extraction. And there is some image which are hard to separate from their background because their background and foreground both have the same contrast. So, our clustering algorithm quite faced problem based on the feature we have provided.
b. **Texture of the Image:** A good texture image is easy to separate its foreground from its background. Edge detection feature will extremely play a good role in this case and increase our accuracy by many folds. On a good texture, we apply a threshold value (Apply different threshold value until or unless we can separate the foreground from the background) and then apply edge detection to separate both. Some of the images have the same texture and which is quite hard to analyze that we are targeting to edge or not and because of that our features turned out very poor feature extraction and got a poor accuracy.
c. **Color of the Image:** Color of the pixel also play a good role in separating the foreground from its background image and hence, gives us a good segmenting image. Remove the saturation of the image and then use this image as a feature extraction will help in differentiating its foreground and background. Some of the images had the quite same color as its background and its very hard to remove the foreground and we don't have much difference in foreground and background in terms of pixel color and got a poor accuracy in my final result. Gradient features affected a lot as we are getting very less difference in the orientation.
d. **Position of the pixel:** Combination of color and position can play a role in extracting its foreground from its background but alone as a feature, there is no sense to use it. It doesn't give any information how foreground is separated from its background because the image is just a matrix with numbers and the numbers don't give any information about the foreground. So, it's hard to detect it from the image. But some of the image affected by the combination of the color and position as if we have same contrast and same intensity in color, it quite hard for out clustering algorithm to separate the foreground from its background and hence, got very less accuracy and poor segmenting image as my result.

Solution Part e.3)

One interesting observation I made while working on this object is that use independent features as much as possible because they will provide you an extra source to separate your foreground image from it's background and will not affect much on the speed as the dependent one. Dependent one cause overfitting

or underfitting and hence gave us poor result. Apart from that, if we have good contrast image, we can use threshold value to separate it from its background and our features turn out as good source of separation.