Homework 5: Report

CSCI 5722: Computer Vision

Rohit Kharat and Reid Glaze

03.19.2022

A. Feature Vector Transform Methods

1. Color: Simple feature vector for a pixel because it is the vector of colors for that pixel. Color information provides an additional perception advantage to help us understand the form and structure of the scene. For example, for parameters (clusteringMethod = 'hac', k = 15; normalize = true; featureFn = 'ColorFeatures'; maxPixels = 1000), we got the mean accuracy of 0.8906. Keeping the same parameters, for KMeans we got the mean accuracy of 0.8975.

2. Color & Position: This is another simple feature vector for a pixel within an image because it concatenates the color vector and position vector. For example, for a pixel in an image (R, G, B) situated at a position (x, y), the feature vector would be (R, G, B, x, y). For parameters (clusteringMethod = 'hac', k = 15; normalize = true; maxPixels = 1000) but changing featureFn = 'PositionColorFeatures', we got mean accuracy of 0.9181. Keeping the same parameters, for KMeans we got the mean accuracy of 0.9173. To run HAC in a considerable time, the maxPixels value had to keep lower.

3. Normalization: This feature vector was used to combine different types of features into a single feature vector. For our case, the RGB channels were in the range [0, 1) while the position of pixel had a much wider range. Clustering algorithms perform poorly if the features are not present on an even scaling. To perform normalization, features were modified such that we had zero mean and unit variance. For parameters (clusteringMethod = 'kmeans', k = 5; normalize = true; featureFn = 'PositionColorFeatures'; maxPixels = 1000), we got the mean accuracy of 0.8700. Keeping the same parameters, but changing normalize = false, we got 0.8618 as the mean accuracy. Normalization did not seem to affect the mean accuracy.

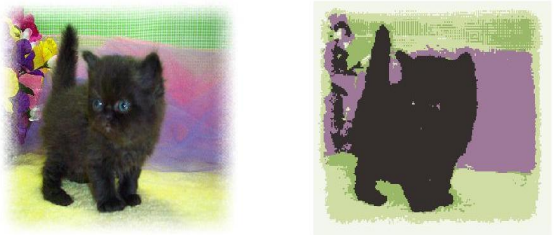
4. Gradient: Gradient is the steepness of the slope at a point and is given by the magnitude of the gradient vector. A gradient can also be used to measure the change in

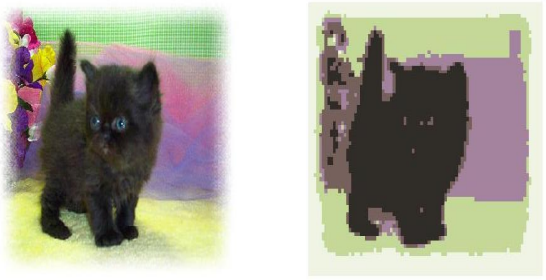
pixel or color with respect to directions. The default Sobel technique was used to define the magnitude and the direction of the gradient. Canny edge detector uses image gradient for edge detection in image processing. The issue with this feature method is that it performs poorly if there is less variation in pixel intensity between the object and the background. For parameters (clusteringMethod = 'hac', k = 10; normalize = true; featureFn = 'GradientFeatures'; maxPixels = 1000) we got mean accuracy of 0.7529.

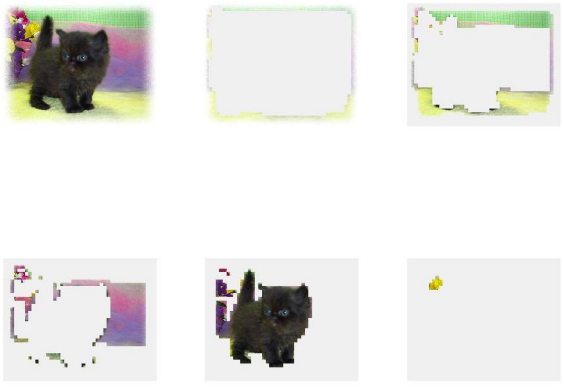
5. Edges: As discussed in the above section, image gradient helps in edge detection. I have used the Canny edge detection method, to aid clustering algorithms for better separation of objects of interest from the background. This feature transform similar to the gradient method performs poorly. For example, for parameters (clusteringMethod = 'hac', k = 10; normalize = true; featureFn = 'EdgeDetectionFeatures'; maxPixels = 1000) we got mean accuracy of 0.7265 which is worse than the gradient. Adding the number of clusters also does not help improve the accuracy. However, using all feature transforms increase the accuracy to 0.8926 because of utilizing color and position feature transforms.

B. Visualization of Segmentation Algorithms

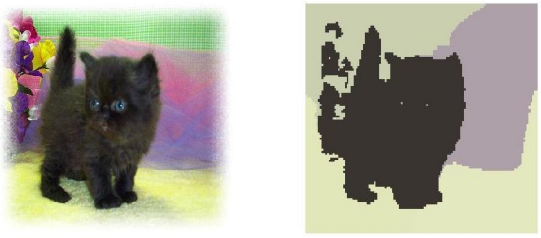
Successful implementation with different parameters: This section shows the successful implementation of segmentation algorithms using different parameters. Experimentally, k = 5 works for most of the feature transforms. Color transforms is able to give decent segmentation instead of going for more complex feature transforms.

<table>
<tr><td colspan="2">clusteringMethod = 'kmeans'; k = 5; normalize = True;
feature = ColorFeatures; resize = 0.5</td></tr>
<tr><td>Mean Color Image</td><td>Segments</td></tr>
<tr><td>  </td><td>  </td></tr>
</table>

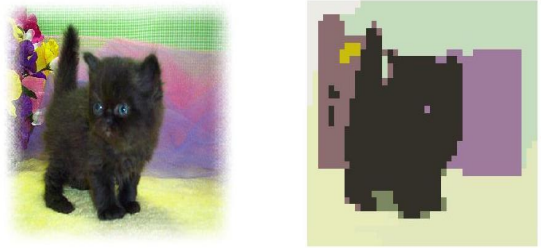
clusteringMethod = ‘kmeans’; k = 5; normalize = True; feature = All (Color+Position+Gradient+Edge); resize = 0.2	
Mean Color Image	Segments
	

clusteringMethod = ‘hac’; k = 5; normalize = True; feature = ColorFeatures; resize = 0.1	
Mean Color Image	Segments
	

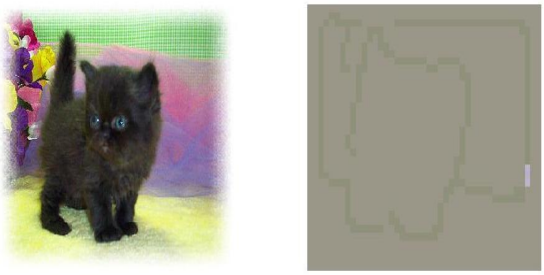
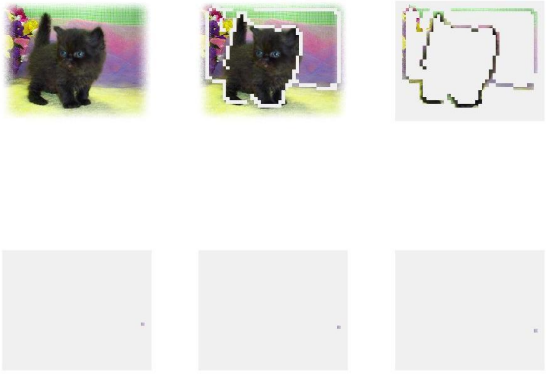
clusteringMethod = ‘kmeans’; k = 5; normalize = True;
feature = PositionColorFeatures; resize = 0.4

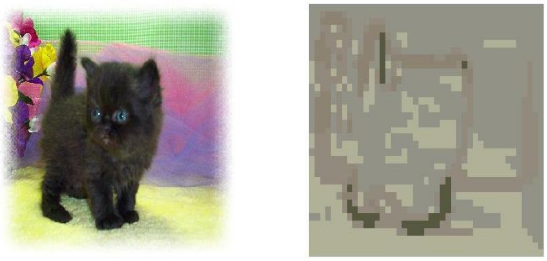
Mean Color Image	Segments
	

clusteringMethod = ‘hac’; k = 7; normalize = True;
feature = PositionColorFeatures; resize = 0.1

Mean Color Image	Segments
	

Unsuccessful implementation with different parameters: This section shows the unsuccessful implementation of segmentation algorithms using different parameters. We noticed that gradient and edge detection alone performs poorly in segmentation. However, when we choose number of clusters too large, we get a large number of segments, and the quality of segmentation is hampered.

clusteringMethod =’hac’; k = 5; normalize = False; feature = EdgeDetectionFeatures; resize = 0.1	
Mean Color Image	Segments
	

clusteringMethod = ‘hac’; k = 5; normalize = True; feature = GradientFeatures; resize = 0.1	
Mean Color Image	Segments
	

clusteringMethod = 'kmeans'; k = 30; normalize = False; feature = All (Color+Position+Gradient+Edge; resize = 0.2	
Mean Color Image	Segments
	

C. Effects of Parameters

1. **What effect does each of the segmentation parameters (feature transform, feature normalization, number of clusters, clustering method, resize) have on the quality of the final segmentation?**

Ans: Color feature transform performs well on giving quality final segmentation images. Color position feature transform also performs well. Other feature transforms, such as gradient and the edge do not give good segmentation images. Normalizing an image does not seem to affect the final output. We got the same quality of images with or without normalization. HAC seems to perform better than KMeans but other parameters also need to be considered for giving a concrete comparison. The resizing of images certainly helps in the case of the time complexity of algorithms. Particularly, in the case of HAC, resizing helps a lot in getting images quickly. Also, as resize value increases, the quality of the segmentation enhances and we got good output images.

2. **How do each of these parameters affect the speed of computing a segmentation?**

Ans: KMeans is a lot faster than the HAC algorithm because HAC starts with forming its own cluster for each pixel and progressively goes on further with segmentation. For

example, KMeans for ‘black-white-kitten’ image with parameters (k = 5, normalize = True, featureFn = ColorFeatures) gives final images at 2.69 seconds. On the other hand, HAC with the same parameters takes 75.41 seconds. KMeans without normalization with parameters (k = 5, featureFn = ColorFeatures) takes less time i.e. 2.47 seconds. The resizing of images certainly helps in the case of the time complexity of algorithms. Particularly, in the case of HAC, resizing helps a lot in getting images quickly otherwise MATLAB gives memory allocation errors. Increasing the number of clusters takes more time, for example, KMeans with k = 7 with parameters (normalize = True, featureFn = ColorFeatures) takes 3.15 seconds.

3. **How do the properties of an image affect the difficulty of computing a good segmentation for that image?**

Ans: The results of segmentation of image varies with the image properties such as image size and resolution. We got good accuracy for images where the algorithm founds better separation between the objects and the background. But, suffers heavily where it is difficult to distinguish the object from the background or otherwise. For example, young-calico-cat performs poorly because the object takes a lot of space in the image, and also the object blends with the background. Furthermore, the algorithm performs poorly at the cat-mouse images because it has a complicated background. Lastly, some images have pixel colors of objects similar to colors of background, which also affects the performance of the algorithm.

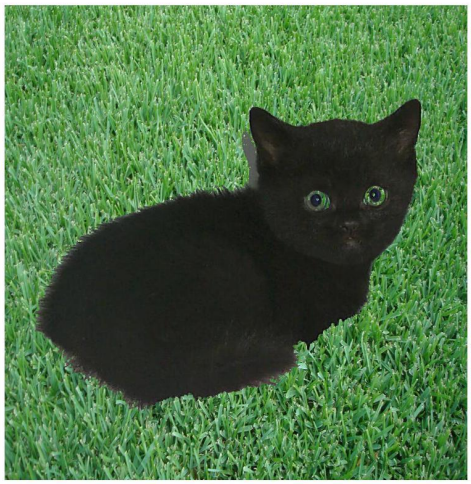
For parameters (clusteringMethod = ‘kmeans’, k = 5, normalize = true, featureFn = ColorFeatures, resize = 1), below is the accuracy of each image:

black-white-kittens2.jpg	0.9009
black_kitten.jpg	0.9603
black_kitten_star.jpg	0.9908
cat-jumping-running-grass.jpg	0.9521
cat_bed.jpg	0.9417
cat_grumpy.jpg	0.8607
cat_march.jpg	0.9888

cat_mouse.jpg	0.6592
cutest-cat-ever-snoopy-sleeping.jpg	0.8950
grey-american-shorthair.jpg	0.9817
grey-cat-grass.jpg	0.9345
kitten16.jpg	0.8586
kitten9.jpg	0.8251
stripey-kitty.jpg	0.8089
the-black-white-kittens.jpg	0.7268
tortoiseshell_shell_cat.jpg	0.8037
young-calico-cat.jpg	0.7302
The mean accuracy for all images	0.8717

D. Composite Images: Transferring Segments

Below are the few composite images after transferring segments to a different background. The parameters used are mentioned below the images. We noticed that the color of the cat's eyes in some images is missing and they take the color of the background. Also, for the black_kitten image, there are some residual segments transferring with the cat. For the cat_bed image, some pixels on the cat's face are missing. Also, keeping the parameters and images the same, running the algorithm gives different results each time. An image with a black background and black object performs poorly in segmentation. Also, lower k, results background taking the shape of the object keeping the same pixel colors of the background. This looked kind of inverse segmentation.

	
Input Images: **Foreground**: black_kitten_star **Background**: grass Segmentation Parameters: clusteringMethod = ‘kmeans’, k = 3, normalize = true, featureFn = ColorFeatures, resize = 0.8	Input Images: **Foreground**: black_kitten **Background**: desert Segmentation Parameters: clusteringMethod = ‘kmeans’, k = 5, normalize = true, featureFn = ColorFeatures, resize = 0.8

	
Input Images: **Foreground**: black_kitten **Background**: earth Segmentation Parameters: clusteringMethod = ‘kmeans’, k = 5, normalize = true, featureFn = PositionColorFeatures, resize = 0.8	Input Images: **Foreground**: Cat_Bed **Background**: beach Segmentation Parameters: clusteringMethod = ‘kmeans’, k = 5, normalize = true, featureFn = PositionColorFeatures, resize = 0.5

E. Parameter Evaluation

	Feature Transform	Feature Normalization	Clustering Method	Number of Clusters	Resize (or max pixels)	Mean Accuracy
1	color	true	k-means	5	1000	0.8577
2	color	true	k-means	10	1000	0.8849
3	color	true	k-means	15	1000	0.8975
4	color	true	k-means	5	8000	0.8700
5	color	true	k-means	10	8000	0.8959
6	color	true	k-means	15	8000	0.9062
7	color	false	k-means	5	1000	0.8602
8	color	false	k-means	10	1000	0.8865
9	color	false	k-means	15	1000	0.8946
10	Color & position	true	k-means	5	1000	0.8700
11	Color & position	true	k-means	10	1000	0.9068
12	Color & position	true	k-means	15	1000	0.9173
13	Color & position	false	k-means	5	1000	0.8618
14	Color & position	false	k-means	10	1000	0.8917
15	Color & position	false	k-means	15	1000	0.9062
16	All	true	k-means	5	1000	0.7294
17	All	true	k-means	10	1000	0.7355

18	All	true	k-means	15	1000	0.7417
19	edge	false	k-means	5	1000	0.7022
20	edge	false	k-means	10	1000	0.6997
21	edge	false	k-means	15	1000	0.7044
22	Color & position	true	HAC	10	1000	0.8981
23	Color & position	true	HAC	15	1000	0.9181
25	All	true	HAC	10	1000	0.8771
26	All	true	HAC	15	1000	0.8926
27	color	true	HAC	10	1000	0.8785
28	color	true	HAC	15	1000	0.8906
29	Color & position	false	HAC	10	1000	0.8878
30	Color & position	false	HAC	15	1000	0.8986
31	gradient	true	HAC	10	1000	0.7529
32	gradient	true	HAC	15	1000	0.7668
33	edge	true	HAC	10	1000	0.7265
34	edge	true	HAC	15	1000	0.7265
35	edge	true	k-means	10	1000	0.7055
36	edge	true	k-means	15	1000	0.7031

F. Qualitative Assessment

1. Based on your quantitative experiments, how do each of the segmentation parameters affect the quality of the final foreground-background segmentation?

Maximum Pixel size
We compared rows 1-3 to rows 4-6 and determined that the maximum pixel size increases the mean accuracy but not by a large margin. This was determined by comparing a maximum pixel size of 1000 to a size of 8000. While this difference may mean something, we decided to isolate the variable because the program takes a very long time to run with a high maximum pixel number, especially when HAC is used.

Number of Clusters
We tested the number of clusters as 5, 10, and 15 for KMeans. We only test 10 and 15 for HAC because a lower number of clusters resulted in an extremely long time for the program to run. For the most part, we found that a higher number of clusters resulted in a higher overall mean accuracy. The one exception to this would be edges, demonstrated in rows 19-21. In this case, the number of clusters does not have an effect.

Feature Normalization
Using k-means, we found that turning off feature normalization does not have a significant effect on the mean accuracy. It was either increased or decreased but not by a large margin. This was determined by comparing the results of rows 1-3 to rows 7-9. When using HAC, we found that turning off feature normalization slightly decreased the mean accuracy. This was demonstrated by comparing rows 29-30 to rows 22-23.

Feature Transform
It was found that the mean accuracy was highest when the color & position features were used. This was found to be true for KMeans and HAC cases. This was demonstrated in rows 10-12 for KMeans. This was demonstrated in rows 22-23 for HAC.

Clustering Method
It was found that the HAC clustering method was generally more accurate than the k-means method. This is demonstrated using all features when comparing rows 25-26 to rows 17-18. However, this difference was not evident when using the color & position feature transform. This was determined by comparing rows 22-23 to rows 11-12.

2. Are some images simply more difficult to segment correctly than others? If so, what are the qualities of these images that cause segmentation algorithms to perform poorly?

In theory, segmentation should work better in images where there is a lot of contrast in between the background and foreground. If the boundary is clear, this should result in higher accuracy.

Using k-means clustering, color & position feature normalization, a cluster size of 15, and a maximum pixel size of 1000, we got a mean accuracy of 0.9173. This was the highest mean accuracy of our 30 trials. The images for the corresponding trial are below.

black-white-kittens2.jpg	0.9158
black_kitten.jpg	0.9569
black_kitten_star.jpg	0.9601
cat-jumping-running-grass.jpg	0.9648
cat_bed.jpg	0.9514
cat_grumpy.jpg	0.8915
cat_march.jpg	0.9568
cat_mouse.jpg	0.8664
cutest-cat-ever-snoopy-sleeping.jpg	0.9056
grey-american-shorthair.jpg	0.9550
grey-cat-grass.jpg	0.9812
kitten16.jpg	0.9076
stripey-kitty.jpg	0.8539
the-black-white-kittens.jpg	0.8399
tortoiseshell_shell_cat.jpg	0.9180
young-calico-cat.jpg	0.8647

From analyzing these results, it appears that images which have a clear boundary between the background and foreground also have a higher accuracy. This would confirm the theory mentioned earlier.

3. Also feel free to point out or discuss any other interesting observations that you made.

Since we limited our maximum pixel size to 1000 for computational purposes, this may not be the most accurate representation of the optimal trial. If the pixel size had been increased, our most accurate trial may have been done through the HAC method. This is probably because HAC tends to yield more accurate results in the other trials.