Homework 5: Report

Alec Bell, David Lee **CSCI 5722** **Instructor: Fleming**

a) *You should describe all methods of computing feature vectors that you used in the assignment. UG students should use any combinations of color and position data (from Task 2), or add more feature data if they wish to. Graduate students can add gradient, edges, SIFT or SURF features, or any new feature vectors they came up with for Task 3. For each method of computing feature vectors, explain the reason you expect that this feature vector will or will not produce a good segmentation for an image. Think about it as a hypothesis for an experiment: "I believe this combination will give good results because …". Your report will be graded as much on the results as on the reasoning you provide. In addition, you should describe all methods of feature normalization that your clustering method employs (task 2).*

Answer)

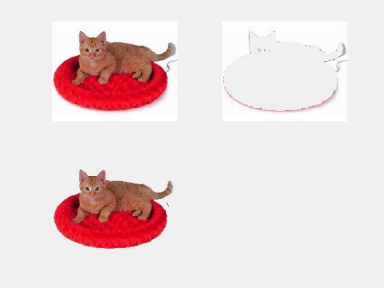
We chose to do a combination of color, position, edge, and gradient data.
Below is a list of the various methods we used along with an explanation for why we thought it would produce a good segmentation for an image.

- Color: @ComputeColorFeatures
    - The most intuitive, we believe color data is a great start for a feature vector. We continued to use this one throughout our experiments because there is a high likelihood that an object in an image (i.e. a cat) will mostly have colors of minorly varying shade or a few colors of minorly varying shade. This allows it to be frequently distinguishable from the rest of the image so long as there aren't other objects of almost the same range of color, so we believe this method is very successful for segmenting images with high contrast among objects.
- Color + Position: @ComputePositionColorFeatures
    - Adding on to the method purely based on color above, including position data is a decent strategy for segmenting an image where there are multiple objects of the same range of color but separated by position. We believe this method to be more successful than the prior since it would help to prevent similarly colored pixels distant from the desired segmentation from being included in the segment.
- Color + Position + Edge (Canny) + Directional Gradient (Prewitt): @ComputeFeatures
    - Lastly, this method includes the previous two features, but also includes edge and directional gradient data.
    - We used the Canny method for edge detection in hopes that including edge data would help the clustering algorithm bring outliers sticking out of a desired segmentation into a cluster. For example, hair sticking out on a cat would be picked up as an edge, then clustered together with other hairs sticking out on the cat also picked up as edges, and eventually these "sticking out cat hair clusters" would be merged in with the main "cat clusters".
    - We used the Prewitt method for directional gradient detection in hopes that areas of similar texture would be clustered together since more textured surfaces in an image (i.e. gravel) will have more changes in directional gradients whereas less textured surfaces (i.e. a blank white wall) will have less changes in directional gradients. The reason more texture would result in more changes in directional gradients is because there would be more color differences due to changes in shade by shadows and varying materials within the real surface.

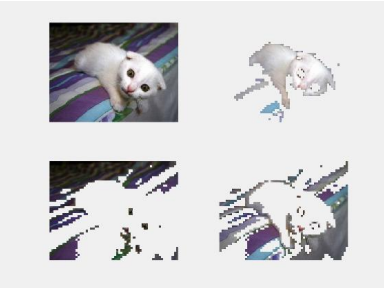
b) *You should include visualizations of at least 6 different segmentations (task 6). At least 3 of these segmentations should be successful, and at least 3 of these segmentations should be unsuccessful. Each of your examples should use different parameters for segmentation, and the parameters for each of your examples should be different.*

Answer)

Successful segmentation examples are black_kitten_star, Cat_Bed, cat_march.



Unsuccessful segmentation examples are cutest-cat-ever-snoopy-sleeping, grey-cat-grass, kitten16.



Example	No. Cluster	Clustering Method	Segmentation Parameters	Norm -alize	Resize
black_kitten_star	2	kmeans	Color	False	0.2
Cat_Bed	2	kmeans	Color+Position+Edge+Gradient	True	0.4
cat_march	2	kmeans	Color+Position	True	0.2
cutest-cat-ever-snoopy-sleeping	2	kmeans	Color+Position	False	0.2
grey-cat-grass	3	kmeans	Color+Position+Edge+Gradient	True	0.2
kitten16	3	kmeans	Color+Position+Edge+Gradient	False	0.2

c) *You should also answer the following questions (a few sentences for each question is sufficient):*

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

Answer)

- Feature Transform.
    - *Color* + *Position* features seemed to yield the best results. It was substantially better than simply *Color* features, but only marginally better than *Color* + *Position* + *Edge* + *Directional Gradient*.
- Normalization.
    - Non-normalized features seemed to yield the best results by a significant margin.
- Number of clusters.
    - This is mostly dependent on the image since each one has a different number of objects within it. However, it did seem that K values between the range of 3-4 tended to yield better results when K=2 or K>=5.
- Clustering method.
    - K-means seemed to yield the best results.
- Resize.
    - We saw slightly better performance as the resize ratio decreased.

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

Answer)

- Feature Transform.
    - It takes a little longer to compute a higher number of features but not much.
- Normalization.
    - This slows downs computation, likely because the algorithm has to work with floating point numbers rather than integers which increases complexity at the hardware level.
- Number of clusters.
    - For HAC, the number of clusters has very minimal impact unless the number of clusters is set to an extremely high value. For K-means, less clusters can significantly increase the speed since it decreases the number of distances that need to be computed.
- Clustering method.
    - In most situations, K-means is substantially faster than HAC. This is because HAC gets much slower as the number of samples increases, and images tend to have lots of samples since they are 2D objects (i.e. 1000x2000 image = 2,000,000 samples).
- Resize.
    - Lower resize values significantly increases the speed for obvious reasons.

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

Answer)

Images with a lot of different objects of varying appearance and coloration, images with objects of very similar color and texture, and images with noise tended to yield the worst segmentations. Whereas images with clearly defined objects of different appearance and low noise tended to yield the best segmentations.

d) *(Extra Credit for Graduate students) Include at least 2 examples of composite images produced by transferring segments from one image to another (task 7). For each composite image explain how you produced it (i.e. describe what the input images were and what segmentation parameters were used).*

Answer)

For this image (black_kitten_star.jpg), we used K-means (K=2), a resize factor of 0.5, the feature transform @ComputeFeatures, and we did not normalize the features.

For this image (Cat_Bed.jpg), we used K-means (K=2), a resize factor of 1.0, the feature transform @ComputeFeatures, and we did normalize the features.

e) *Include a detailed evaluation of the effect of varying segmentation parameters (feature transform, feature normalization, clustering method, number of clusters, resize) on the mean accuracy of foreground-background segmentations on the provided dataset. You should test a minimum of 30 combinations of parameters. To present your results, you might consider*
making a table similar to Table 1.

Feature Transform	Normalization	Clustering Method	K	Resize Factor	Mean Accuracy
ComputePositionColorFeatures	1	K-means	3	0.1	0.9043
ComputePositionColorFeatures	1	K-means	3	0.3	0.9101
ComputeFeatures	1	K-means	3	0.1	0.9097
ComputeFeatures	1	K-means	3	0.3	0.9091
ComputePositionColorFeatures	0	K-means	3	0.1	0.9402
ComputePositionColorFeatures	0	K-means	3	0.3	0.9161
ComputeFeatures	0	K-means	3	0.1	0.9388
ComputeFeatures	0	K-means	3	0.3	0.9161
ComputePositionColorFeatures	1	K-means	4	0.1	0.9039
ComputePositionColorFeatures	1	K-means	4	0.3	0.9060
ComputeFeatures	1	K-means	4	0.1	0.9037
ComputeFeatures	1	K-means	4	0.3	0.9122
ComputePositionColorFeatures	0	K-means	4	0.1	0.9401
ComputePositionColorFeatures	0	K-means	4	0.3	0.9424
ComputeFeatures	0	K-means	4	0.1	0.9350
ComputeFeatures	0	K-means	4	0.3	0.9491
ComputePositionColorFeatures	1	HAC	3	0.1	0.8796
ComputePositionColorFeatures	1	HAC	3	0.3	0.8887
ComputeFeatures	1	HAC	3	0.1	0.8966
ComputePositionColorFeatures	0	HAC	3	0.1	0.9392
ComputeFeatures	0	HAC	3	0.1	0.9392
ComputeColorFeatures	1	K-means	2	0.15	0.9129
ComputeColorFeatures	1	K-means	2	0.25	0.9156
ComputePositionColorFeatures	1	K-means	2	0.15	0.9035
ComputePositionColorFeatures	1	K-means	2	0.25	0.9068
ComputeColorFeatures	0	K-means	5	0.15	0.9454
ComputeColorFeatures	0	K-means	5	0.25	0.9554
ComputePositionColorFeatures	0	K-means	5	0.15	0.9464
ComputePositionColorFeatures	0	K-means	5	0.25	0.9487
ComputeFeatures	0	K-means	5	0.15	0.9452

f) Expand upon the qualitative assessment of Section c) by answering the following questions:

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

Answer)

Note: All answer is based on kmeans method except for clustering method due to lack of dataset collected with hac. For comparison of clustering method, we only applied same combination of kmeans among all dataset as we have with hac.

- Feature Transform:

Feature Transform	AVERAGE of Mean Accuracy
ComputeFeatures	0.9217125
ComputePositionColorFeatures	0.9203875

@ComputeFeatures resulted in slightly better segmentations on average. This is likely because the additional features helped clustering in some edge case scenarios, but overall they were almost equally performant.

- Normalization:

Normalization	AVERAGE of Mean Accuracy
0	0.934725
1	0.90273

Non-normalization resulted in better segmentations on average. This is likely because algorithms like K-means and HAC operate on Euclidean distance, so normalization ensures every feature is weighted equally in determining which cluster a pixel belongs to. In a non-normalized feature vector, however, features that have the potential to have higher magnitudes than other features have a greater influence on the distance to a cluster. This yielded slightly better results for us likely due to the fact that color features were [0,255], position features were [0,height] or [0,width] (depending on which was larger), and edge/gradient features were [0,1]. This, to us, seems to reflect the order of precedence for how these different features should be weighted. Color and position are the most important, and then other minor features like edges and gradients came last in influencing the clustering outcome.

- Number of clusters:

K	AVERAGE of Mean Accuracy
3	0.91127
4	0.92405

4 clusters resulted in better segmentations on average. We believe this is because it provides enough room for the background to be segmented multiple times, but not enough for the foreground to be improperly segmented.

- Clustering method: Kmeans show better performance.

Clustering Method	AVERAGE of Mean Accuracy
hac	0.88415
K-means	0.9072

** Due to lack of HAC dataset which is applied by wide variety of combination, we only compare the result based on some constraints:*

- Feature: ComputePositionColorFeature
- Normalization: 1
- Number of Cluster: 3
- Resize ratio: average of 0.1, 0.3

K-means resulted in better segmentations on average. We were surprised by this after reading some literature comparing the two generally, so we think this was largely influenced by us selecting the “right” parameters outside of the method. It would probably be good to test this on a larger dataset.

- Resize:

Resize Factor	AVERAGE of Mean Accuracy
0.1	0.9219625
0.3	0.9201375

A resize ratio of 0.1 does slightly better than 0.3 on average, although it may be negligible enough to not allow us to make conclusions about it. We think it may have been because images that have been resized by bilinear interpolation may have reduced variance though since it generally takes the mean of surrounding pixels at a location, especially when it’s resized to be very small like 0.1.

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

Answer)

One example black-white-kittens2.jpg and grey-american-shorthair.jpg. These two always had an accuracy range between 0.70~0.80. One characteristic we found on these was that there was a white cat having very similar colors to the background. Even if there is some variation in colors, it becomes very hard to segment if the feature distance between an object and the background is closer than distance between parts within an object. We might be able to improve the quality if we put more features. By putting more features, the effect of RGB color similarity between the object and background will be diminished.

3. *Also feel free to point out or discuss any other interesting observations that you made. Note: Overall for this assignment, the focus is more on your choices, the reasoning you provide, and the explanation & discussion, than on the actual code or results (although these are important too!).*

Answer)

The feasibility of the HAC method would be largely improved with better hardware. We had to resize images to be very small to get a result with this method in any reasonable time frame. We are curious what factors could be improved to improve the speed of the algorithm while also maintaining quality segmentations.

Another observation we made was that finding the right features seemed to have a huge impact on the result of the segmentation. We are curious what additional features may result in better segmentations and why.