CS 477 HW #11

Questions completed: All undergrad level questions in MATLAB

A1. Implementing K-means clustering.

In this section, the K-means clustering segmentations are shown. For the K-means stopping criterion, the approach of waiting until the clustering assignments do not change was chosen with a fallback max iteration number of 200. Luckily, this did not have to be reached for any of the image segmentations. My thoughts on this are that the method we choose does not really matter since we are only working with trivial examples in this assignment. It is a good point however that for real world usage, you might have a break case if you know that bad starting point clusters were selected. The ideal solution in contrast is just to have enough computing resources so that you can keep the workflow as simple as possible.

In Figure 1 below, the original sunset.tiff image, the K=5 segmentation, and K=10 segmentation are shown below.

(a) Original sunset.tiff image.

(b) Segmentation with K=5.

(c) Segmentation with K=10.

Figure 1: Segmentation of sunset.tiff.

In Figure 2 below, the original tiger-1.tiff image, the K=5 segmentation, and K=10 segmentation are shown below.

(a) Original tiger-1.tiff image.

(b) Segmentation with K=5.

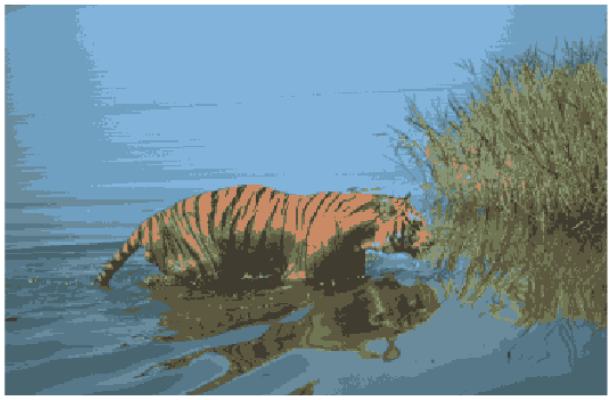
(c) Segmentation with K=10.

Figure 2: Segmentation of tiger-1.tiff.

In Figure 3 below, the original tiger-2.tiff image, the K=5 segmentation, and K=10 segmentation are shown below.

(a) Original tiger-2.tiff image.

(b) Segmentation with K=5.



(c) Segmentation with K=10.

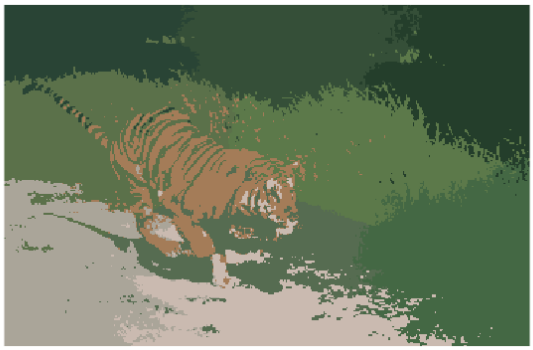
Figure 3: Segmentation of tiger-2.tiff.

A2. 5-dimensional feature vector.

In Figure 4 below, segmentation using the specified 5-dimensional feature vector of $(R, G, B, \lambda * X, \lambda * Y)$ is shown for the three images using $\lambda = 1$ and $\lambda = 10$ for $K = 10$ clusters.

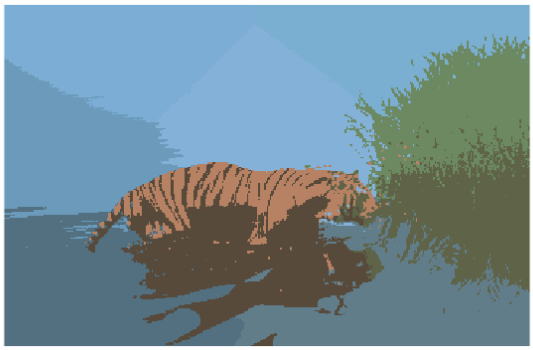
(a) sunset.tiff with $\lambda = 1$.

(b) sunset.tiff with $\lambda = 10$.



(c) tiger-1.tiff with $\lambda = 1$.

(d) tiger-1.tiff with $\lambda = 10$.



(e) tiger-2.tiff with $\lambda = 1$.

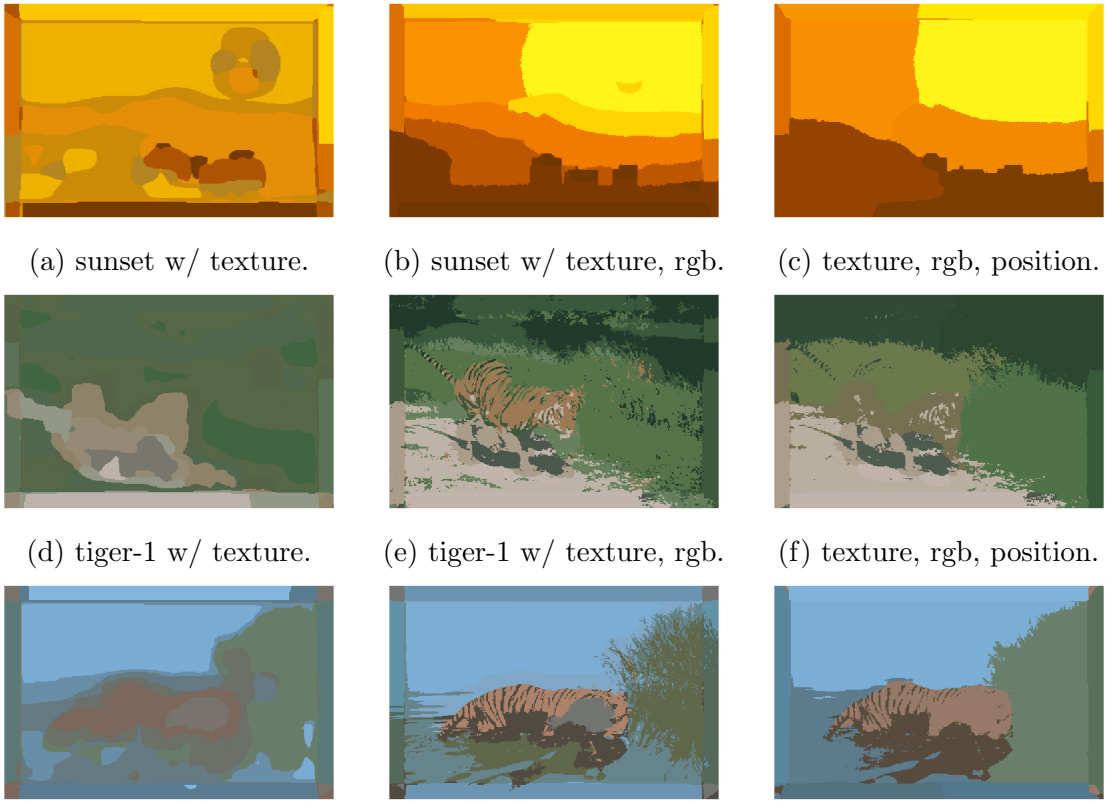
(f) tiger-2.tiff with $\lambda = 10$.

Figure 4: Segmentation of images (K=10) with 5-dimensional feature vector.

Based on Figure 4 above, it appears that increasing λ which is the weight placed on the pixel position results in segmentation that is more sensitive to the position of the pixels. This is seen from the difference between the left and right images of Figure 4 for every row.

A3. Using texture features.

Below in Figure 5, results for the three different feature vectors for each of the three images are shown. The parameters are $W = 10 \times 10$ window size, $K = 10$ clusters, and $\lambda = 1$ as the pixel position weight. Additionally, the root mean squared value in the window W was taken instead of the mean squared value. All features in the feature vector were scaled to the range of 0 to 255.



(a) sunset w/ texture. (b) sunset w/ texture, rgb. (c) texture, rgb, position.

(d) tiger-1 w/ texture. (e) tiger-1 w/ texture, rgb. (f) texture, rgb, position.

(g) tiger-2 w/ texture. (h) tiger-2 w/ texture, rgb. (i) texture, rgb, position.

Figure 5: Segmentation of images (K=10) with 5-dimensional feature vector.

From Figure 5 above, we can see that using just texture features alone (first column) gives an interesting segmentation that is better than just placing a high weight on position with RGB but worse than segmentation with RGB alone. Using texture features plus RGB (second column) also gives an interesting segmentation, but it seems

that texture features corrupts more than helps segmentation that would have been done with just RGB. Finally, using texture features plus RGB plus position (third column) gives another interesting trio of segmentations, but it is perhaps worse than texture features plus RGB. For these images, it is perhaps better to cluster on just RGB alone as the groupings are mostly color-based, but for a different set of images that are mostly the same color, texture features may improve the segmentation.

Another thing of note are the visual artifacts in the images around the borders that creates a second kind of border within the images. I ran into this issue in assignment 8, but ultimately I do not know where this exactly comes from. My best guess is that it arises from the convolution of the images used to average the texture features in the window W because I observed that the visual artifacts around the border changes as I change the window size, so perhaps it is just the way that I am convolving images in MATLAB that produces this unwanted visual artifact.