Brandon Carr
CVIP Homework 2

Foreground-Background Texture-Based Segmentation via Clustering

---

Implemented Solution

The segmentation process involves identifying points within the foreground and section them off from the background. Once we do this, we can easily transplant foreground elements into whichever image we desire.

Execution starts with the formation of a filter bank. A function is called that creates and stores a bank of filters, a three-dimensional matrix containing 48 unique filters. Our input image is then convolved with each of these filters, and the filter responses are then stored in a second three-dimensional matrix. The data point matrix X is then constructed. We have as many rows in X as we have pixels in our input image. Each element in a row is one pixel's filter response for a specific filter. X is then passed off to a k-means clustering algorithm.

K-Means Algorithm Implementation

For this assignment, I have chosen to implement my own version of the k-means algorithm to do the clustering of points. The function itself requires two things; the data point matrix "X" and the number of cluster centers to be calculated, "k." The user can also include a matrix called "centers" as an optional parameter.

The data matrix X is a stacked matrix in which each row contains one data point from the original image "I." The optional matrix, "centers" is a matrix containing the user-chosen cluster centers of the data points. If no such matrix is included, the function will randomly select k data points from X and create a cluster centers matrix automatically.

The main task the function performs is to cycle between all points in X and calculate their distance to the chosen cluster centers. Each point receives a cluster assignment which is stored in a new vector, "idx." The cluster centers are then re-assigned to be the mean values of their data point members. This entire process will continue until either one of two things happens: the cluster center reassignments do not change past a certain distance threshold, or the number of total iterations exceeds 100. This allows us to be sure that the cluster center assignments are as close to the actual centers of the clusters as they can be.

The vector idx is structured so that row 'm' of idx contains the value of "centers" that point 'm' in X was assigned to. At the end, the function returns idx, the final cluster assignment vector, to be used to construct our index image.

Final Steps

Using our index image, we are now able to determine which clustered pixels belong to the foreground and background. With this knowledge, we can determine our foreground segmentation vector. This is simply the values of the indices that are in the foreground of our segmented image. Putting this vector, idx, the source image, and the target image into the function transferImg() will return a final image of the animal set in front of a background image.

There is also an issue with automatically obtaining the foreground vector. Currently, the code does not accomplish this. The foreground vector must be obtained by the user and then fed into the transferImg() function. This is, however, not an impossible feature. One could implement this automation by counting the instances of each index in our index image's border. The index with the most occurrences within the borders of the image is most likely the background index.

Limitations

Using a pixel-wise calculation results in a mostly-accurate index image to be produced. Edges of the foreground are almost flawlessly identified. However, there are some instances where parts of the foreground are assigned to a background cluster. This results in the segmented image having small clusters of pixels removed from the foreground. Using a window as opposed to a single pixel when assigning parts of the image to clusters will likely result in a clearer segmentation with fewer "holes."

In this particular implementation, we are categorizing pixels based on texture. If we were to take other factors into account, such as color or interest points, we would surely obtain more desirable results.

Results

The following pages contain the results of our implementation using the given sample images. The k values used are k=2 for all images.

Zebra

Gecko

Dalmatian

Cheetah