Image Segmentation and Classification Using Texture

Rodrigo Daudt
rcdaudt@gmail.com

Songyou Peng
psy920710@gmail.com

I. INTRODUCTION

This report describes the texture segmentation and classification based on the usage of texture descriptors obtained from the gray level co-occurrence matrices (GLCM). The design, implementation, and result analysis of the usage of this method for image segmentation and classification will be discussed on the following sessions.

II. ALGORITHM ANALYSIS

GLCM is a matrix that counts how often different transitions between pixel gray scale values occur in an image or subimage for a given offset [1]. Every element m_{ij} in the GLCM corresponds to the number of times that two points with a given offset and with quantized gray values i and j exist in the image. That is to say, GLCM considers the spatial relationship of two pixels to build a form of directional transition histogram.

After acquiring the GLCM for a given distance and angle, we can easily compute statistical properties of the matrix, such as homogeneity, correlation, and contrast. As we know, the texture of an image is defined by the way the gray levels of the pixels vary in space. Different textures frequently lead to different values of homogeneity, correlation, and contrast, especially if the GLCM is calculated using different offsets. Therefore, the information obtained from the statistical analysis of the GLCM can be used to help with the segmentation and classification of textured images.

III. DESIGN AND IMPLEMENTATION FOR SEGMENTATION

The segmentation of the images using texture features was divided in a few steps. After loading the images and converting them to double precision, each image was processed separately since they each demanded different parameters or slightly different procedures in order to achieve a good segmentation result, but the workflow stayed mostly the same for all the images.

The first step was to calculate the texture descriptors for each pixel of the image. This was done by transforming the image to grayscale, creating a window around each pixel, calculating the GLCM of that sub-image, and then extracting the texture descriptors from the GLCM. Smaller windows were used along the edges of the images to avoid the need to pad the images. A function called *compute_descriptors* was programmed to calculate the descriptors for each image, given a window size and an offset. This function used the built-in MATLAB functions *graycomatrix* and *graycoprops*. The features chosen to be used in the classification process were contrast, energy and homogeneity.

After the texture descriptors were calculated, a new N-layer image was created by stacking selected layers from the original image and the texture descriptor images. Once this new image with more information was created, it was then preprocessed in a number of ways, including Gaussian blur, median filter, value clipping, value stretching, and histogram equalization, The parameters and order for each of these preprocessing techniques was tuned in order to obtain good segmentation results.

The segmentation itself was performed by the region growing function that was implemented during previous lab session. The only modification to this function was to move from Manhattan distance to Euclidean distance, since it seemed to perform better for our results, perhaps due to the variable dimensionality of the input images.

IV. SEGMENTATION RESULTS ANALYSIS

During the implementation, we found out texture features are really useful for segmentation, which can be seen from the Fig. 1. If we only use the color feature for segmentation, the results show a large number of small regions on the background because of the texture in the original image. When we provide texture information to the image segmentation algorithm, the fragments disappear and pixels in the same regions join together.

It is worth mentioning that we can acquire even better results if we use prior knowledge of the image for segmenting. Fig. 2 illustrates this point. We notice that, instead of using all RGB 3 channels information, we can just use the blue channel and texture descriptors because we know the background is totally blue, therefore the blue values are very different between the background and the foreground which helps the segmentation process. The result in Fig. 2(d) shows less fragments and better integrity.

However, it should also be noted that adding the texture features is likely to cause the segmentation to be less precise along the boundaries between regions. Comparing the original images with the results of segmentation with texture feature in Fig. 1 and Fig. 2, we can notice that the resulting boundaries in the results of segmentation using texture features are not very precise.

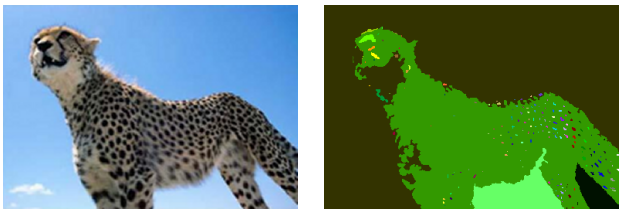
Besides, for a 256×256 grayscale image, in the environment of Mac OS X 10.10.5, 2,7 GHz Intel Core i5, 8 GB 1867 MHz DDR3, it takes more than 60 seconds to calculate all the texture feature images if the mask size is 11×11. The implementation speed of the algorithm is so slow that it is hard to be applied to real-time applications, which is a strong limitation.

(a) Original

(b) RGB

(c) RGB+Texture

Fig. 1. Comparison of the segmentation results without and with texture features



(a) Original (b) RGB

(c) RGB+Texture (d) B+Texture

Fig. 2. Comparison of the segmentation results without and with texture features

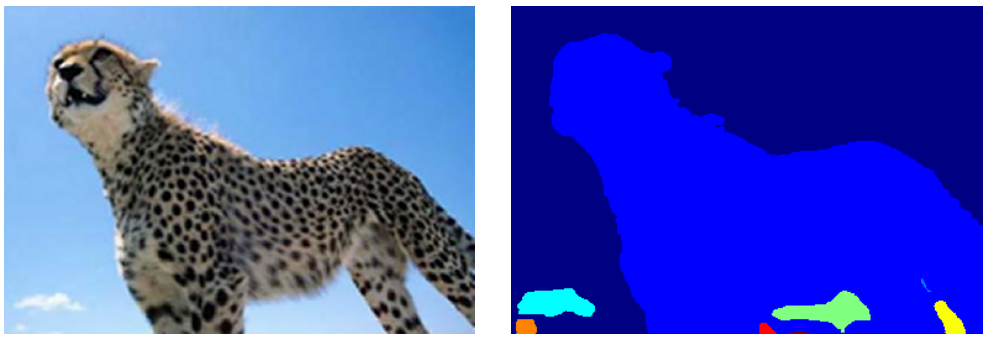
Figures 3, 4, 5, and 6 show side by side the original images and the best obtained result for segmentation using texture features for each provided image. These images are the ones obtained after the final tuning of the parameters of the preprocessing steps mentioned in Section III.

The image feli.tif shown in Fig. 3(a) had its segmentation results vastly improved by the use of texture descriptors, since the cheetah's fur has a very distinctive texture, while the sky behind it is almost completely plain. We can see in Fig. 3(b) that the cheetah was completely segmented from the sky with minor mistakes made by the algorithm in the lower part of the image.

The hand2.tif image, which is shown in 4(a), proved itself to be more challenging to segment. Given the similar hue between the background and the hand, this image relied heavily on texture descriptors to be segmented. But given that texture descriptors are not very precise with respect to position, the small objects and borders in the image required a more precise tuning of the segmentation parameters. The best result obtained with our method can be found in Fig. 4(b).

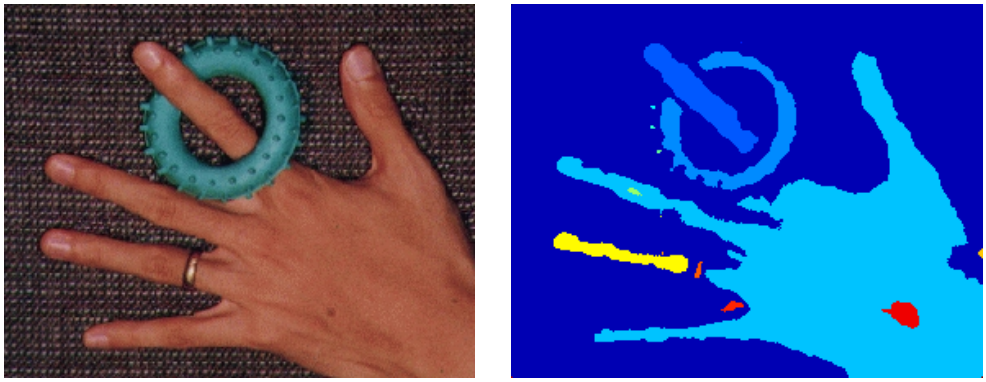
The mosaic8.tif image, shown in Fig. 5(a), proved itself to be the most challenging to segment. The image is composed of four textures with similar colors, and therefore it relied heavily on the use of texture descriptors for the segmentation. GLCMs with two different direction offsets had to be used to achieve a decent segmentation of this image, and a very fine tuning of the segmentation and preprocessing parameters was necessary to achieve the segmentation found in Fig. 5(b).

Finally, the pingpong2.tif image, found in Fig. 6(a), was also challenging to segment, given the similarity between the colors of some of the objects in the image. The fine tuning of the segmentation parameters led to good segmentation results, but it was still not perfect for differentiating between the background, the playing hand and the racket handle, as can be seen in Fig. 6(b).



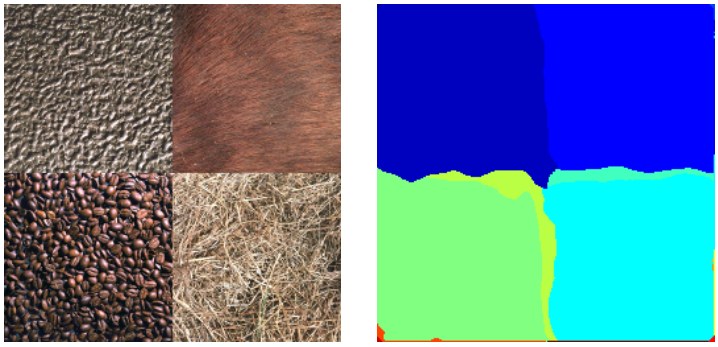
(a) Original (b) Segmentation result

Fig. 3. Comparison of original images to the segmentation results for feli.tif



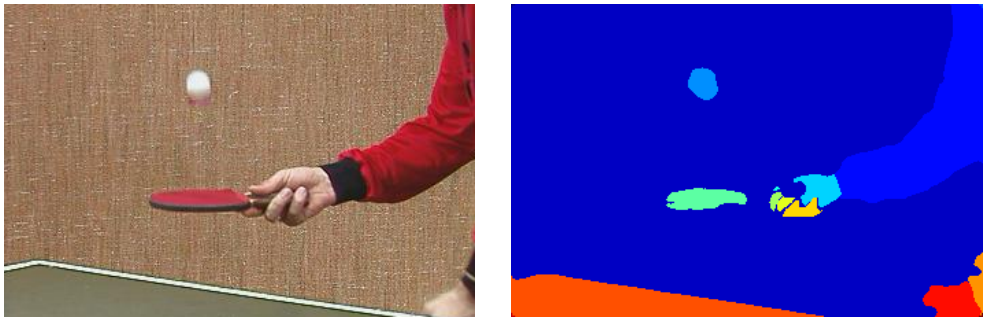
(a) Original (b) Segmentation result

Fig. 4. Comparison of original images to the segmentation results for hand2.tif



(a) Original (b) Segmentation result

Fig. 5. Comparison of original images to the segmentation results for mosaic8.tif



(a) Original (b) Segmentation result

Fig. 6. Comparison of original images to the segmentation results for pingpong2.tif

V. DESIGN AND IMPLEMENTATION FOR CLASSIFICATION

In the classification part, instead of calculating the texture features for each pixel, what we want is to produce a feature vector using texture descriptors for each image in our dataset.

First, we calculate the co-occurrence matrix for each image and acquire different statistic values from the matrix. Here we use all 4 statistics properties: energy, homogeneity, correlation and contrast. Therefore, a 1×4 vector is acquired for the current image. And then, we change the angle or/and distance to acquire a new co-occurrence matrix and its corresponding 1×4 vector. Finally, all the vectors are concatenated to one single vector, which is the feature vector of the image. We repeat the same process for all the images in both training and testing set.

After calculating the feature vector for each image, we send it to Weka and apply random forest to train and evaluate our model. Based on the testing results, the final accuracy rate is calculated.

VI. CLASSIFICATION RESULTS ANALYSIS

When only texture features are used, the highest accuracy rate we obtained was 92.5%. This can only be obtained as all 4 statistics properties are calculated in the distances of 1, 12, 23 along 8 directions (all multiples of $45°$, starting at $0°$). We are also able to achieve a 93.75% classification accuracy by adding the mean gray value of the image, and 98.75% classification accuracy by adding the mean values for the R, G and B layers.

The thing that we observed was that scaling the features in each dimension to the range of $[0, 1]$ led to a worse result rather than a better one. We scaled our data as follows:

$$data\_normalized_i = \frac{data_i - min_{data_i}}{max_{data_i} - min_{data_i}}$$

where max_{data_i} and min_{data_i} are the maximum and minimum values among all the images (training set + testing set) along dimension i, e.g. contrast on the direction $[1, 0]$.

From [2], we know the ranges of contrast and correlation are $[0, (size(GLCM, 1)-1)^2]$ and [-1, 1] respectively, while energy and homogeneity are already in the range [0, 1]. Therefore, one possible explanation that scaling leads to worse results is that scaling will reduce the differences of contrast and correlation among different images, which makes it harder for the classifier to achieve a good result.

VII. PROJECT MANAGEMENT

It was decided at the beginning of the execution of this project that, for learning reasons, each of the members would develop their own implementations of the both the segmentation and the classification parts. After two working versions existed, features of both versions were combined in order to produce a final version that produced the best results we could. The Github platform was used as a collaborative tool for the code development[1], and the report was written using the online LaTeX platform Overleaf.

[1]Rodrigo Daudt & Songyou Peng, SSI-Labs, (2016), GitHub repository, `https://github.com/rcdaudt/SSI-Labs`

VIII. CONCLUSIONS

In this session we observed the utility of using texture descriptors derived from the GLCM of an image in the processes of segmentation and classification of images. These descriptors provide us with more information than a simple RGB analysis would yield, and this information can be essential when analyzing images where textures can be observed.

We were also able to observe that the calculations of texture descriptors are computationally demanding, and therefore must be used wisely. Depending on the way they are applied, the time these calculations take to finish may be too long for these techniques to be used in real time applications.

REFERENCES

[1] Hall-Beyer, Mryka. "Tutorial: GLCM Texture". Fp.ucalgary.ca. N.p., 2016. Web. 18 Mar. 2016. `http://www.fp.ucalgary.ca/mhallbey/the_glcm.htm`.

[2] "Documentation." Properties of Gray-level Co-occurrence Matrix. Accessed March 17, 2016. `http://fr.mathworks.com/help/images/ref/graycoprops.html`.