Scene Segmentation and Interpretation

Image Characterization using Texture

SAVINIEN BONHEUR & ALBERT CLÉRIGUES
Universitat de Girona
March 23, 2017

1 Introduction and problem definition

Texture characterization adds a layer of complexity to the characterization paradigm faced so far in this subject, where we have been using the color and intensity to segment similar pixels in regions. However, texture patches have pixels that are very different to each other according to these features. Hence we need a way of computing descriptors that also have into consideration the pixel neighbourhood to characterize pixels belonging to arbitrary textures.

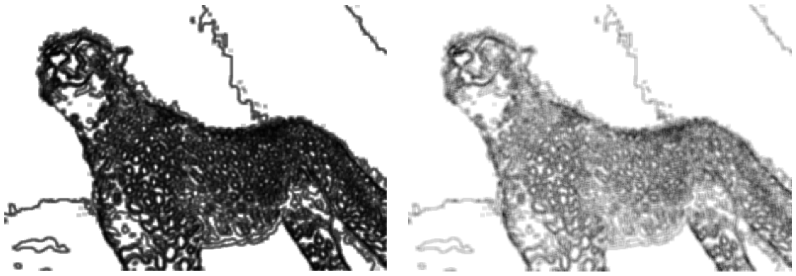
For this matter we will develop descriptors made using co-ocurrence matrices for neighbourhood description, which we will use to enhance the region growing segmentation algorithm and to classify texture patches.

2 Texture descriptor for segmentation

In this section we will explore different parameter configurations for the computation and statistics of co-ocurrence matrices with the goal of getting an optimal local texture descriptor.

First, we need to find the glcm of the neighbourhoods of all pixels in the image, which is computed using the function "graycomatrix" that Matlab provides in its toolbox. This function provides the possibility to tweak differents parameters such as 'Graylimits', 'Offset' and 'Numlevels'. In the following, we will experiment with each of these parameters independently to pick the ones that best suit the segmentation problem.

2.1 Graylimits and NumLevels



(a) NumLevels=2 (b) NumLevels=8

Figure 1: NumLevels comparison of local texture descriptor

The Graylimits parameter controls the range in which the input image will be normalized, this can allow us to get rid of too bright and too dark spots. If the value is too high it will lead to a less general texture descriptor, if too low it will lead to a too specific descriptor for different textures. We decide to keep the default range which will adapt to the image range.
The NumLevels parameters makes reference to the quantization of the image intensities for glcm computation. As we can see in Figure 1, the difference between using 2 or 8 number of levels is really different, and it is better described for our purposes with 2 levels, since it provides sharper edges and more homogeneous regions. Consequently we will stick with Matlab default value of 2 number of levels.

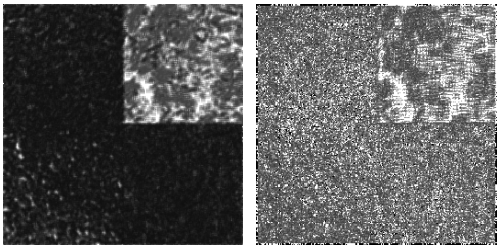
2.2 Offset

The Offset can be thought as composed of two distinct parameters, the distance from the central pixel at which we will compare the intensities and the orientation in which we take the step.

(a) Single orientation (b) Four orientations

Figure 2: Number of orientations comparison in the obtention of the glcm

In figure 2 we can observe that the 4 orientations give a better result, since we are obtaining more information from the neighbourhood of the pixel, which means we are averaging more transitions and getting a more stable descriptor while being more robust to noise.



(a) distance=2 (b) distance=4

Figure 3: Neighborhood's distance comparison in the obtention of the glcm

High frequencies are better described by taking smaller neighborhoods into account, if the considered neighbourhood is too big we can introduce artifacts as seen in Figure 3(b). After seeing the results with different distances we pick an offset distance of 2 which is big enough to avoid considering close pixels that could be too similar, but small enough to correctly describe the neighbourhood.

3 Region Growing segmentation enhancement

After we have adjusted the parameters to obtain a good local texture descriptor for the provided images we will add this descriptors to the region growing algorithm implemented in Coursework 1.

We append the texture descriptors as new channels to our image, with the texture normalized between 0 and 255, along with the color channels, obtaining the following results:

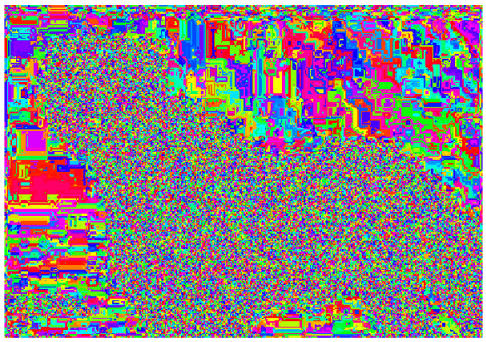


Figure 4: Segmentation result with appended texture descriptor

The result is completely oversegmented and really noisy with all different thresholds. We conclude that the problem lies in the texture and color descriptors, that have a lot of discontinuous regions and high frequencies. To try and solve this issue we applied gaussian filtering with different kernel sizes to try and obtain more homogeneous regions to ease the region grow segmentation.

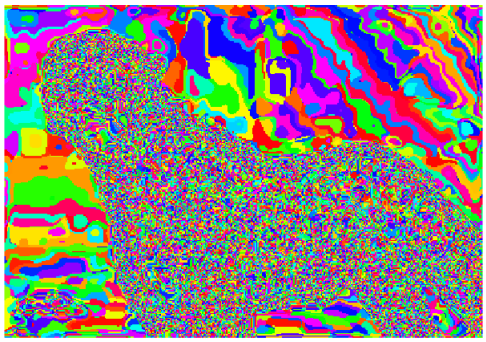


Figure 5: Segmentation result using color and texture descriptors with Gaussian filtering

As it turns out, the Gaussian filter is too aggressive and we can't find a good compromise for the size of the kernel. A smaller one will not achieve enough homogeneity inside regions to avoid over segmentation, and bigger ones blur too much the edges leading to inaccurate results.

We decided to try with a median filter that also gets rid of high frequencies while making a much better job at conserving the edges. As seen in Figure 6, the result is even better than only color information with low-pass filtering.

(a) Color and Texture descriptors (b) Only color descriptors

Figure 6: Results of region grow segmentation with median filter

4 Segmentation results and conclusions

After establishing the parameters for texture descriptor computation, found a way to integrate it within the color information for the region grow algorithm and added median filtering we test our algorithm with the provided images, obtaining the following results:

Figure 7: Original Images

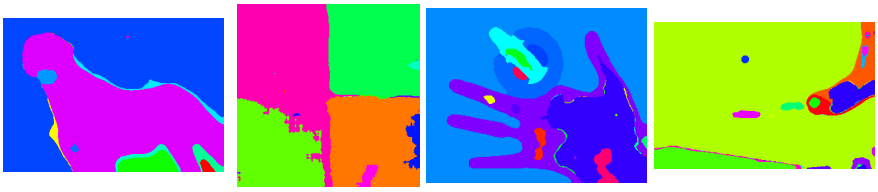


Figure 8: Segmentation result with color information and median filtering

Figure 9: Segmentation result using combined color and texture descriptor with median filtering

From the obtained results, we can conclude that adding textures descriptor work well for the type of images for which it has been tuned with, such as high frequency textures. However, for areas with lack of texture or low frequencies, the texture descriptor will actually lead to over segmentation of those regions compared to using only colors information. We conclude that using texture characterization for segmenting arbitrary images isn't an optimal solution if they introduce low quality information that can mess with the segmentation. Texture descriptors should only be used when colors cannot, all by themselves, describe the different regions properly.

5 Classification using texture descriptors

The classification of different texture patches will be done using a K Nearest Neighbour Classifier (KNNC). For labelling a given sample, it will compare it with all samples from the training set, get the K most similar to it and assign the label of the majority of them.

Figure 10: Examples of textures for classification

In our implementation we start from an existing framework that has already implemented all the code for database loading and classification, being left to implement the computation of the descriptor array. We will use co-ocurrence matrices for global texture information extraction, this is different from the use for segmentation since now we are characterizing the entire image at once. We will then summarize this matrices in a series statistical parameters to create a robust descriptor that should be as different as possible for different textures.

5.1 Co-ocurrence Matrices Computation

The descriptor used will be based on the statistical analysis of co-ocurrence matrices, which are built with the following parameters:

- **Number of Levels**. The number of gray levels correspond to the quantization of the intensity values to count the transitions. By reducing the number of gray levels we will group the transitions on coarse intensity groups and be more robust to noise, while also reducing memory and complexity. However, by using a low number of levels we may oversimplify the description and loose precision.
- **Offsets** The offset parameter in the Matlab function 'graycomat' specifies several locations for intensity comparison with the pixel we are considering. One co-ocurrence matrix will be made for each offset, encoding the transitions of each pixel and a fixed neighbour.

  The offsets will be formed by the 4 basic orientations (0^o, 45^o, 90^o and 135^o) at different distances to obtain a description of the whole neighbourhood of the pixel. Since we aim to obtain a global image

descriptor, the number of offsets considered and its range will be higher than the segmentation descriptor.

5.2 Descriptor computation

We then use the function provided by Matlab 'graycoprops' to compute statistics from the co-ocurrence matrix that encode the characteristics of the texture. Specifically we will be using the four parameters provided by Matlab:

- **Contrast:** measure of the average intensity contrast between a pixel and its neighbor.
- **Correlation** how statistically correlated a pixel is to its neighbor.
- **Energy**: uniformity of the image, is 1 for a constant image.
- **Homogeneity**: measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal. If all elements lie in the diagonal all considered steps transition between same intensities.

These four statistical parameters are computed for each of the co-ocurrence matrices computed for each of the different offsets, which we then concatenate together to form the descriptor array.

6 Classification results and conclusions

Although we have already experimented with the "graycomat" function parameters in the segmentation section the goal was completely different. In this section we will explore the best parameters for global image description, experimenting with a bigger number of offsets and graylevels for accurate classification.

To carry out these tests we have used a slightly modified version of the script to let us use different parameters for co-ocurrence matrix computation. We tried different values of "NumLevels" to see the effect of quantization in classifier accuracy as well as different offset distances in the four basic orientations.

From the test results seen in Figure 11 we conclude that the most accurate and computationally efficient parameters are 8 gray levels and offsets in all 4 directions with distance 1 to 5, which result in 91.25% of classifier accuracy. By using these parameters we obtain a descriptor with 4 elements for each direction and range, which add up to an array of 80 elements.

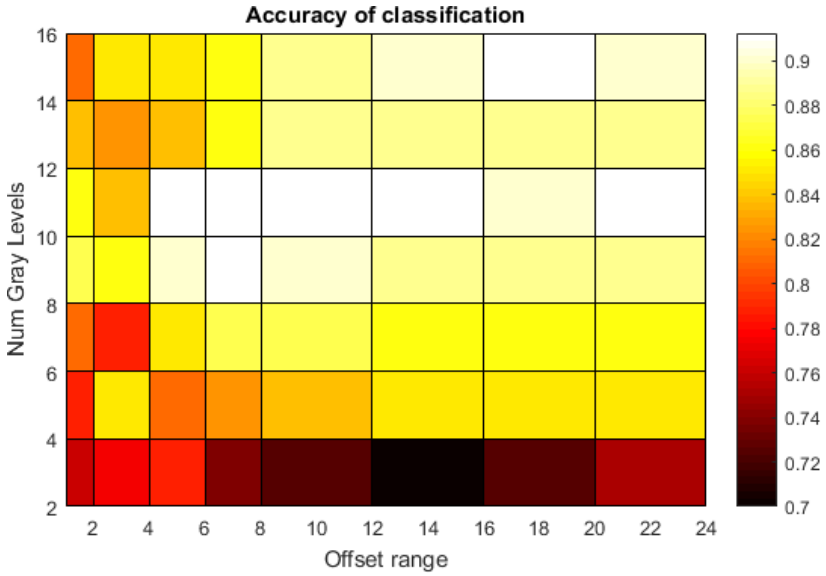


Figure 11: Accuracy of classification with respect to different co-ocurrence matrix parameters

After carrying out the classification with the selected parameters, we check the confusion matrix to try and spot any pattern or trend in the misclassified samples that we could try to solve.

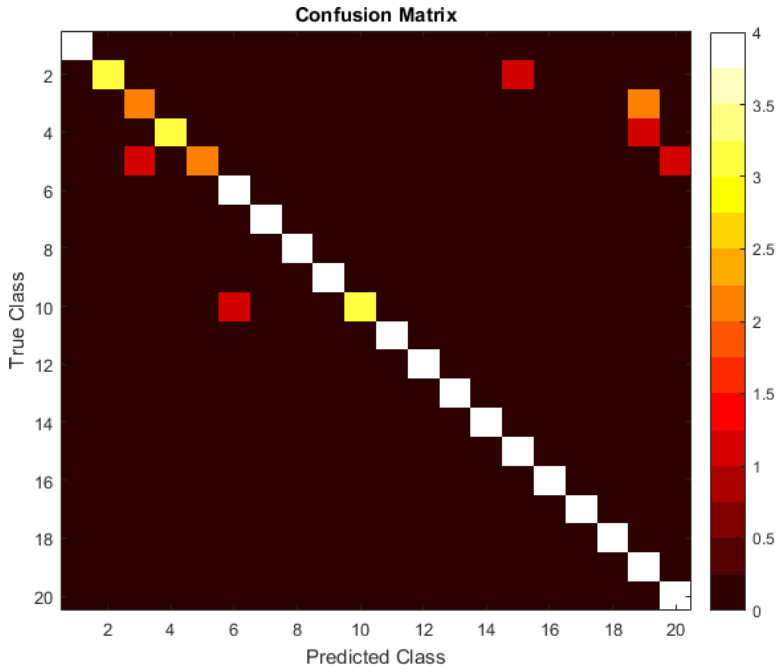


Figure 12: Confusion matrix of classification

The misclassified samples don't seem to follow a pattern, however, upon close inspection we see that some of the misclassified textures have very different colors. The obvious step in trying to increase the accuracy of classification is to also encode color information on our descriptor. We do so by computing one descriptor for each channel of the RGB image and concatenating them to obtain a descriptor with 240 elements.

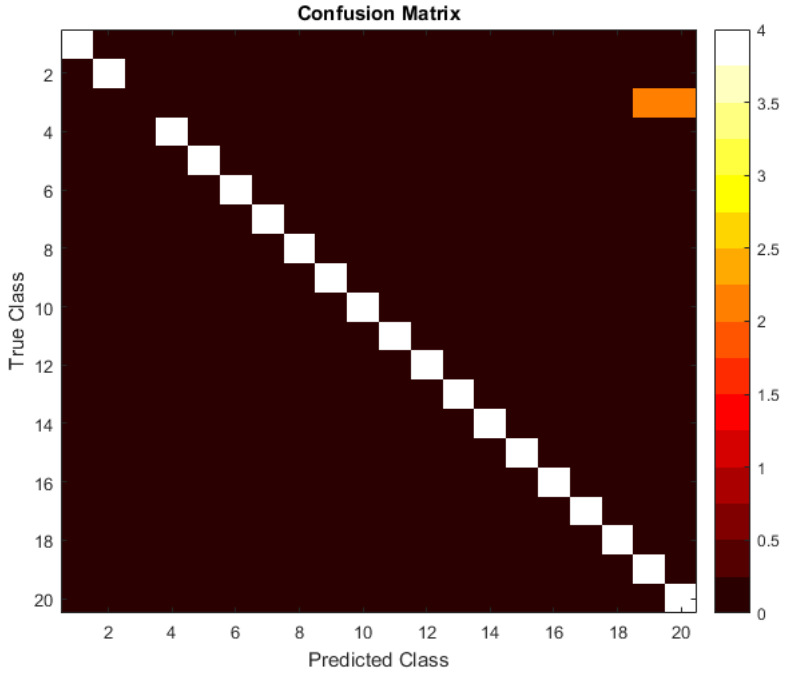


Figure 13: Confusion matrix of classification with color

The classifier accuracy after adding color information into our descriptor increases up to 95%. The confusion matrix shows class number 3, consisting of big green leaves, is being completely misclassified. A possible explanation could be that in this predominantly green texture only the green channel has valuable information for the descriptor, while the red and blue add no valuable information and may introduce errors when computing the similarity with other classes. This problem could be generalized for images whose color information is not distributed evenly among all channels. A possible solution could be to project each image to a common non-correlated color space, but this falls outside of the scope of this coursework.

7 Organization and development

The workload estimation and distribution finally was as follows:

Tasks	**Time**	**In charge**
Descriptor for segmentation.	2h.	Savinien
Descriptor for classification.	2h.	Albert
Addition of texture to Region Growing.	2h.	Savinien
Segmentation testing and tuning.	4h.	Albert & Sav
Classification testing and tuning.	2h.	Albert
Write coursework's report.	8h.	Albert & Sav

Table 1: Workload time estimation and distribution

8 Conclusions

Throughout this coursework we have faced multiple problems while trying to correctly characterize texture.

It is clear that texture has to be characterized by studying the pixel neighbourhood, but it is unclear the size that we should consider since it could vary depending on the size of its features, like the radius of the cheetah's spots. This problem is just a specific instance of the generalization problem, since different parameter configurations will be optimal for some textures but worsen the results to others. This has been seen with the addition of color information to the classifier descriptor, that improved results for all classes but completely ruined it for one of them.

Still both the classification and segmentation parts have seen the results greatly improved by the addition of color to pure texture descriptors. This two observations lead to the conclusion that a robust feature descriptor should try to use as much information available while trying to be general enough to correctly characterize all possibilities.