Scene Segmentation and Interpretation

Lab 2 Report: Image Characterization using Texture

Mohammad Rami Koujan
M.Sc. VIBOT
University of Girona

1 Introduction and problem definition

A texture is a set of elements occurring in some regular or repeated pattern. It is, in fact, another feature that can help to segment images into regions of interest and to classify those regions. In some images, it can be the defining characteristic of regions and critical in obtaining a correct analysis. The image in Figure 1 has four distinct textures. The texture of the top left, top right, bottom left, bottom right squares. These textures can be quantified and used to identify the object classes they represent.

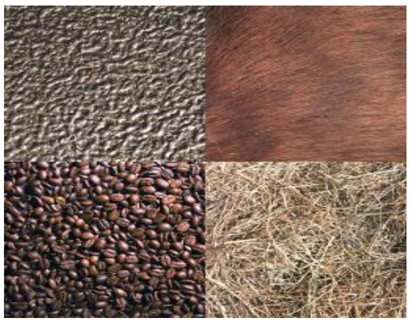


Figure 1: image with four regions characterized by different textures.

A number of methods can be employed in order to characterize images using texture information. Some of these are:

1- Statistical methods

   - Co-occurrence matrices
   - Energy (Laws masks)
   - Parametric masks
   - Local Binary Patterns (LPB)

2- Structural methods
3- Modelization methods

   -Markov Random Fields

4- Space-frequency filtering methods

   - Gabor filters

   - Wavelets

Therefore, the use of these methods to describe texture regions will ease the task of segmenting or classifying these regions.

2 Texture algorithm analysis

One of the efficient statistical methods that is used usually to characterize textures is the co-occurrence matrix. A co-occurrence matrix, GLCM, is a two-dimensional array in which both rows and the columns represent a set of possible image values V. For example, for gray-tone images, V can be the set of possible gray tones and for color images V can be the set of possible colors. The value of this matrix at the index (i,j) indicates how many times value i co-occurs with value j in some designated spatial relationship. For example, the spatial relationship might be that value i occurs immediately to the right of value j.

To be more precise, the interest during this lab is specifically at the case where the set V is a set of gray tones, and the spatial relationship is given by a matrix that specifies the displacement between all possible combinations of the intensities included in the gray image. Indeed, the value of the co-occurrence matrix is determined by a number of pre-defined parameters, e.g. distance between neighbors, number of intensity levels, and orientations. The distance and orientation specify the pixel separation and location of neighbors relative to each other. For example, when these two parameters are defined as follow: [D -D], the co-occurrence matrix will contain the number of occurrences of all the possible pairs of pixels, according to their intensity, that are at a row separation of D and column separation of –D, or at distance D and angle 135 degree. However, the number of intensity levels parameter determines the size of the co-occurrence matrix. Hence, for 256 different intensity levels, the size of the co-occurrence matrix is 256*256. This means that GLCM matrices are always square with the same dimensionality as the number of grey-levels.

When GLCM matrix is normalized, it represents the probability that two points with grey-levels i and j exist in a distanced and orientation theta. Consequently, a number of statistical features can be computed from this matrix like energy, entropy, contrast, homogeneity, and correlation.

Depending on the desired application, those feature descriptors are usually computed to aid the classification or segmentation procedure. For segmentation purposes, the mentioned features should be calculated locally using a window that slides on the entire image. Thereafter, the obtained values are used as extra dimensions in addition to the image RGB colors, for instance, to add information about the texture regions inside the image. This means that each pixel in this case has a vector of features that includes GLCM statistical parameters and its color or intensity value. While for the aim of classification, those descriptors are computed globally for the entire image, which means all pixels in the image are used to compute a final vector of features, and used for the sake of training and testing steps during classification.

3 Design and implementation of the proposed solution for the segmentation problem

In this section, an implementation of a region-growing segmentation algorithm based on texture and color information is explained in details. Furthermore, the proposed method is tested on four images with distinct textural characteristics.

As explained in the previous section, in order to take advantage of the texture information, the feature descriptors should be calculated locally on the neighborhood of a pixel.

This results in a vector of features for each pixel of the image. Then, the resultant vector of features should be combined with a vector of RGB values of each pixel so that this bigger vector can be passed to the region-growing segmentation function, instead of only using the color values, and consequently used for segmenting the image.

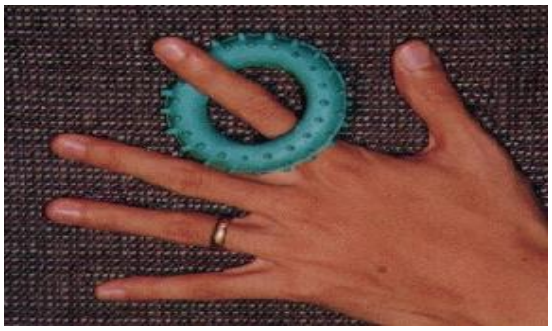
The implemented function takes the window size, offset vector, original image, value of the segmentation threshold, and the number of intensity levels inside each window as input arguments and returns the segmented image and the number of regions as output arguments. The function initially starts padding the image according to window_size variable because a window of this size will be centered on each pixel while calculating the feature vector at each iteration and consequently to take boarder pixels into account the image should be padded. Next, a loop iterates on every pixel opening a window and passing the internal pixels' intensities to the "graycomatrix" function in order to compute the GLCM matrix. In addition to the local window content, the offset vector, which determines the orientations to be taken into account during the calculation of the GLCM matrix, and the number of intensity levels, which determines the size of the GLCM matrix, are passed to "graycomatrix" as well. The GLCM matrix then is passed to "graycoprops" function which has the responsibility of computing the statistical parameters, energy, contrast, and homogeneity. However, the entropy is calculated separately by another function. At the end of this loop, a number of images are obtained. Each of them represents one feature descriptor. Moreover, there could be more than one image for each feature depending on the number of rows of the offset variable (number of orientations). Before sending those images to the region growing algorithm, a normalization step takes place to ensure that all the images have values within the same range (0 to 1). Otherwise, they will represent different weights in the segmentation function and consequently some of them will have more impact on the final result than the others. In fact, those images are considered as additional dimensions of the original image.

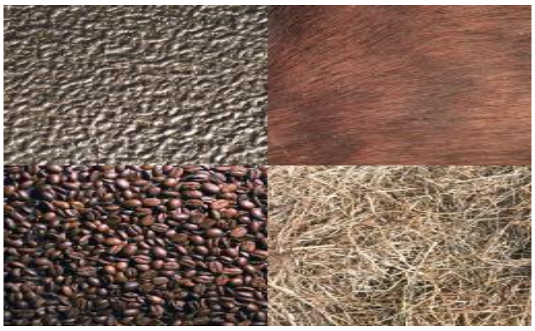
The idea of the segmentation algorithm is to start from a random pixel, the first one in the provided code, by giving it a label and then obtain the neighbors of this pixel, 4-neighbors connectivity is considered here. Those neighbors will be checked by a separate function, called IsCheckedBefore, if they have any other label or being processed and then inserted to a queue, the aim of which is to organize the procedure in terms of controlling the shape of the region being grown, which is done for example by inserting neighbors clockwise, growing more effectively level by level around the seed, and easing the implementation. Then, each pixel inside the queue will be retrieved and tested to determine if it satisfies the aggregation condition or not. The testing condition is based on taking the norm of the difference between the feature descriptor vector of the current pixel and the mean vector. It is noteworthy that the size of these two vectors, in case one orientation is considered while computing the GLCM matrix, is 7*1 (energy, contrast, homogeneity, entropy, red, green, blue) and this number will increase by four for each new orientation.

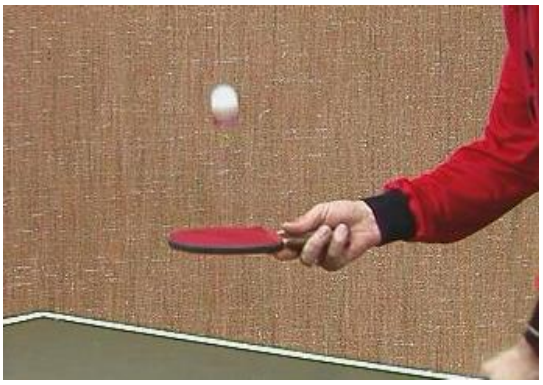
If the testing condition is satisfied, the pixel under study will be given the region label and its neighbors will be obtained and passed to IsCheckedBefore function in order to insert them at the back of the queue. Otherwise, the retrieved pixel will not be assigned any label. This process ends when the queue becomes empty which means the current region is not able to grow any more. Subsequently, the previous steps will start again with new unassigned pixel until all the pixels have labels. One final comment concerning the implementation is that, the image is filtered with Gaussian filter at the begging of the segmentation procedure and then its colors are normalized between 0 and 1.

4 Experimental section and results analysis for the segmentation problem.

This section presents the results of testing the previously described algorithm on the four following images:


(a)


(b)


(c)

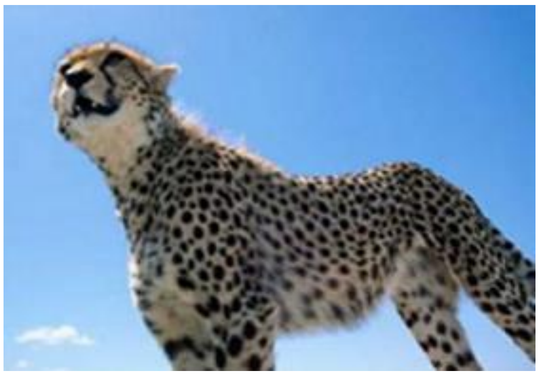
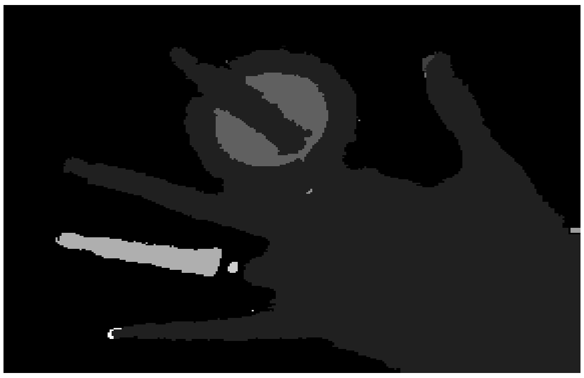
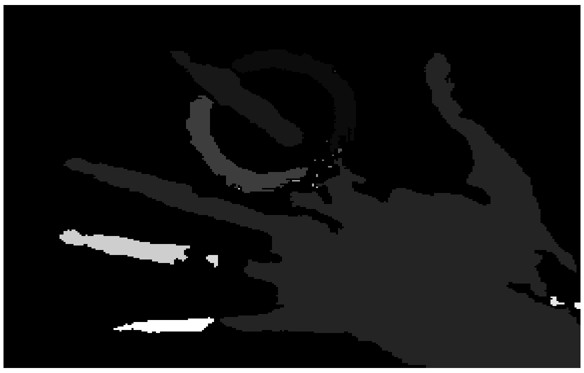
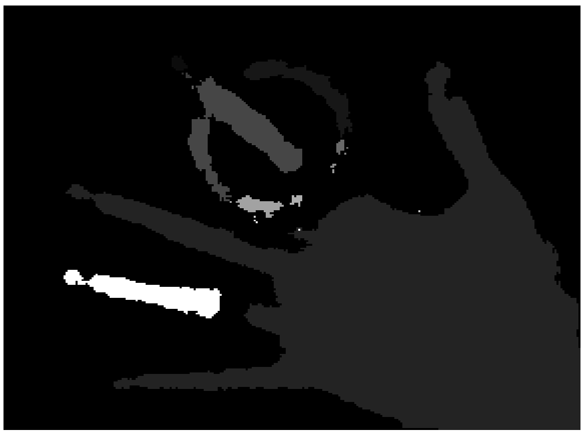

(d)

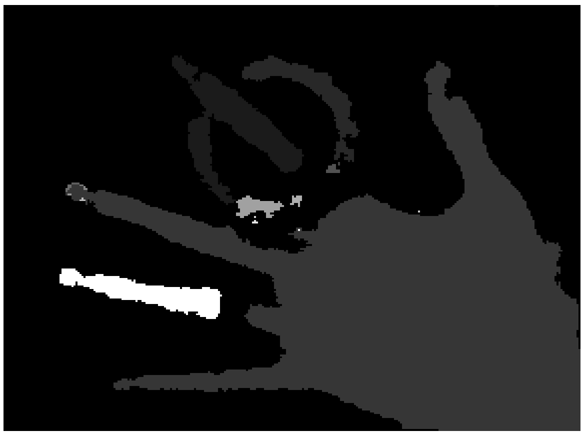
Figure 2: (a) hand image (288*228 pixels), (b) Feli image (382*265 pixels), (c) ping pong image (344*228 pixels), (d) mosaic image (256*256 pixels).

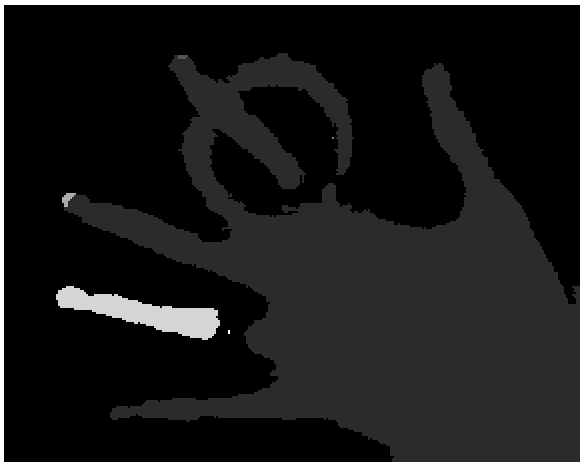
The parameters that are varied during the test process are the window size, number of intensity levels considered during the calculation of the GLCM matrix, the number of orientations (number of rows of the offset vector) and the threshold. In addition, during comparison the following shorthand notation is used: Image name__number of orientations__window size__threshold value. Figures 3 to 6 show the results of testing the region growing algorithm based on the color and texture information on the images in figure 2. The values of the previously mentioned parameters with the same described notation is written under each image.

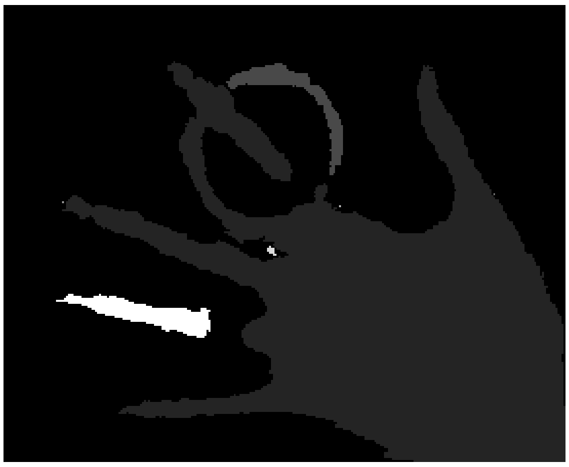

hand_2O_11W_32L_1.8T

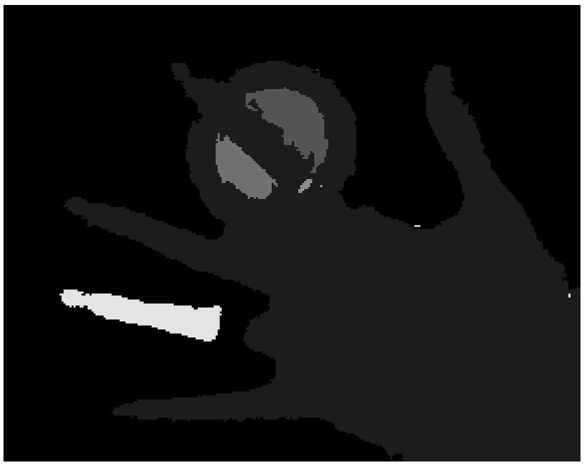

hand_2O_11W_32L_2.1T

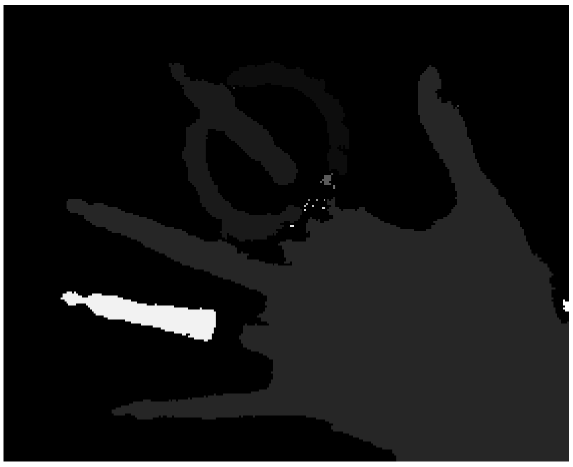

hand_2O_11W_64L_1.9T

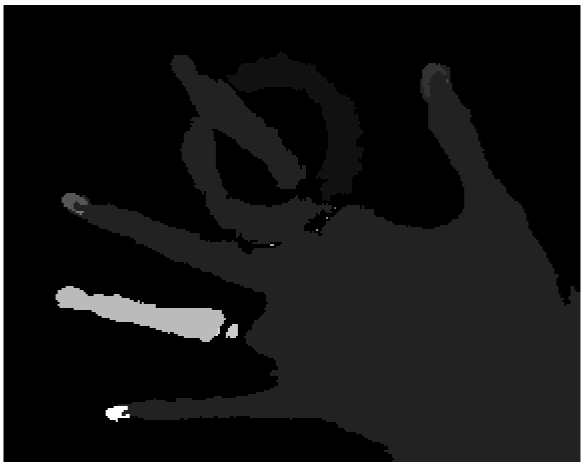

hand_2O_11W_64L_2T

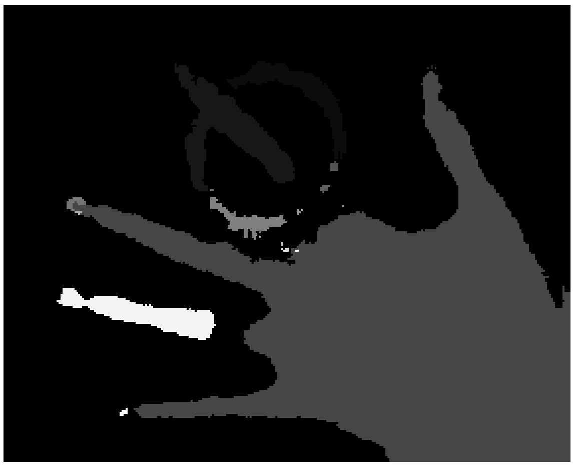

hand_2O_15W_32L_1.85T

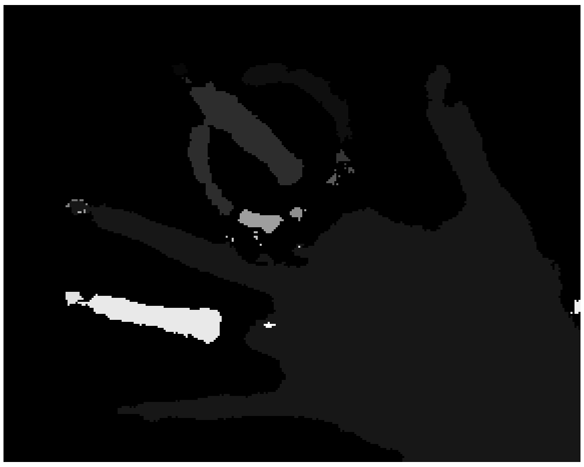

hand_2O_15W_32L_2.1T

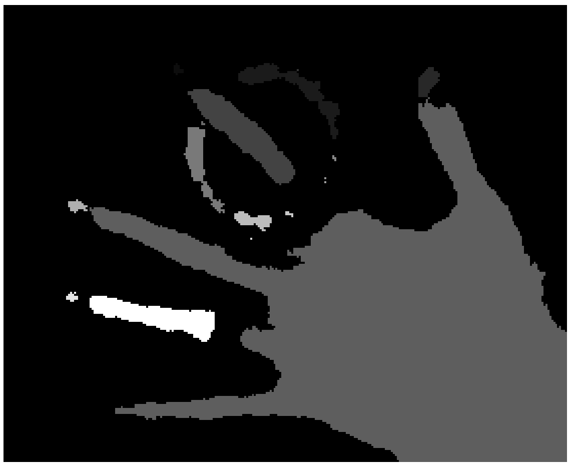

hand_2O_15W_64L_1.75T

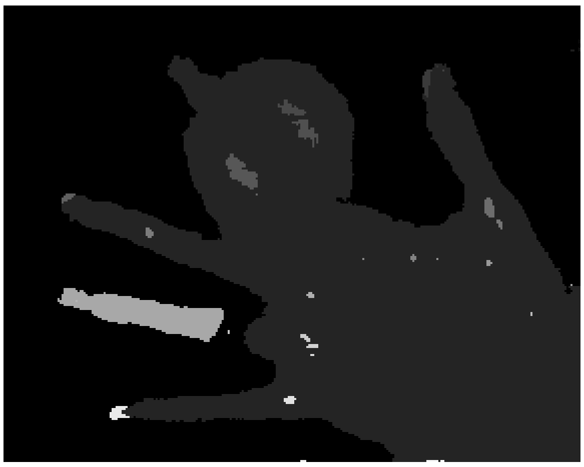

hand_2O_15W_64L_1.95T

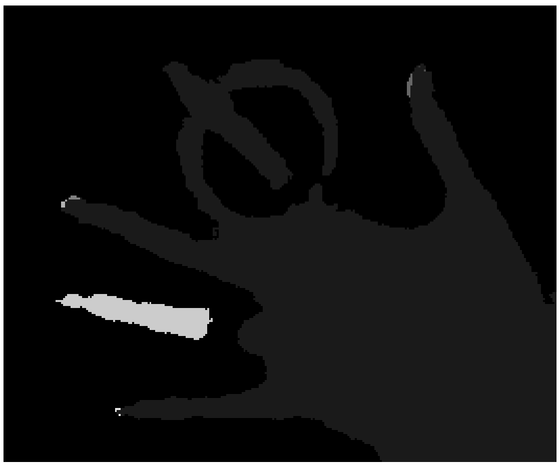

hand_4O_11W_32L_2.3T

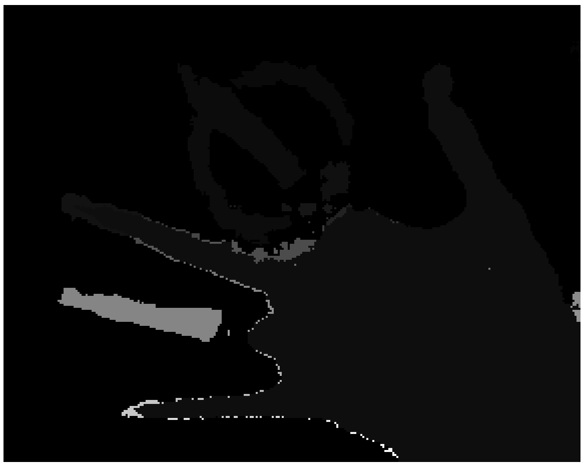

hand_4O_11W_32L_2.6T


hand_4O_11W_64L_2.3.1T


hand_4O_11W_64L_2.6.1T


hand_4O_15W_32L_2.1T


hand_4O_15W_32L_2.3T


hand_4O_15W_64L_2.1T

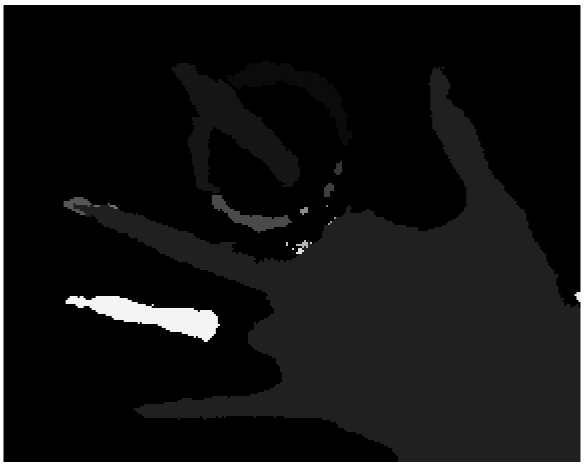
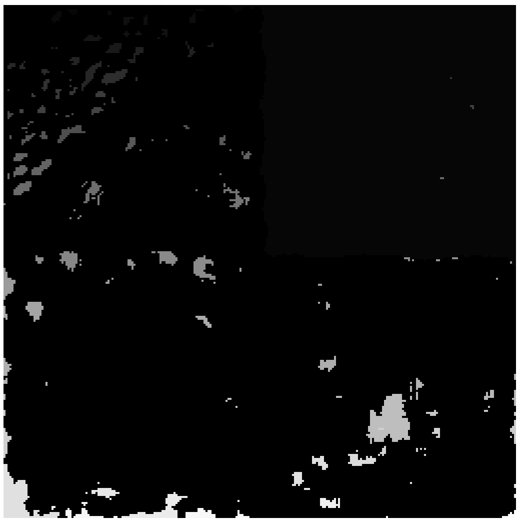
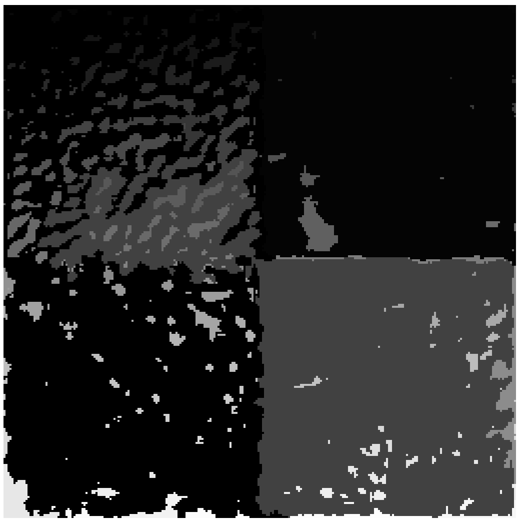
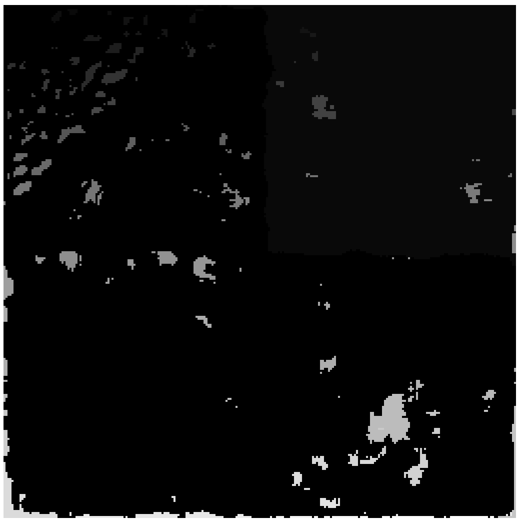
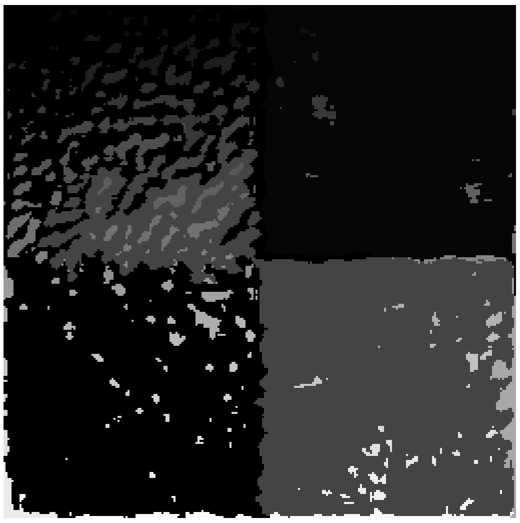

hand_4O_15W_64L_2.3T

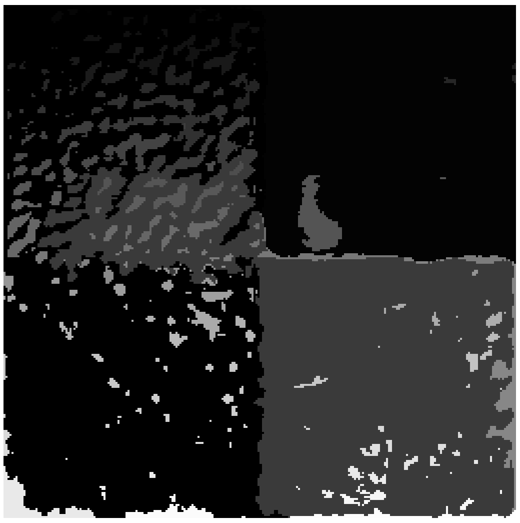
Figure 3: results of testing the segmentation algorithm on the hand image

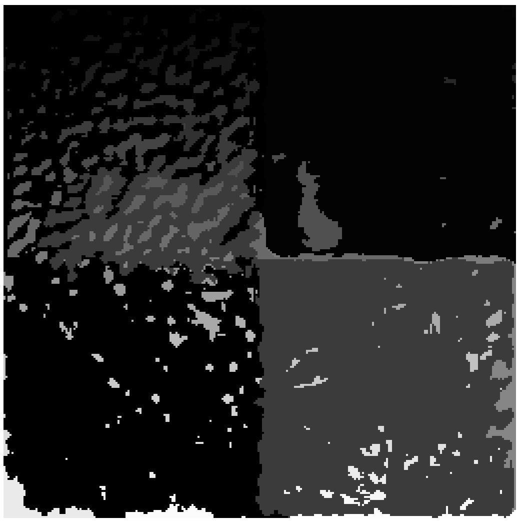

mos_2O_11W_32L_1.6T

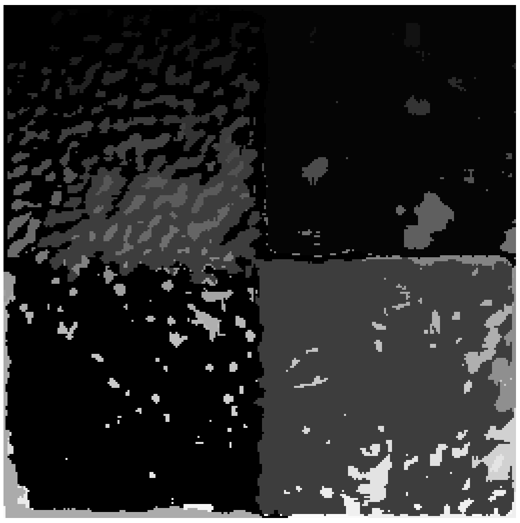

mos_2O_11W_32L_1.47T

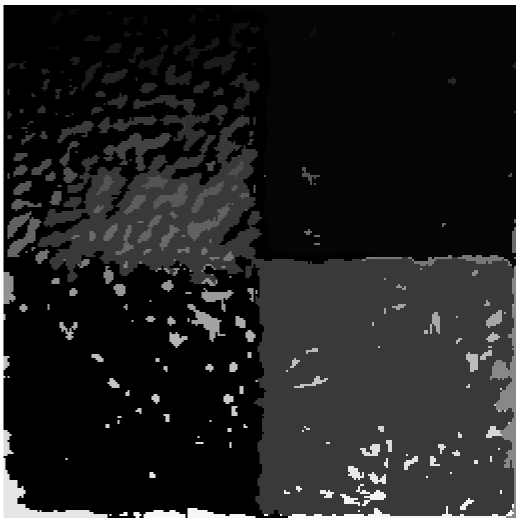

mos_2O_11W_64L_1.6T

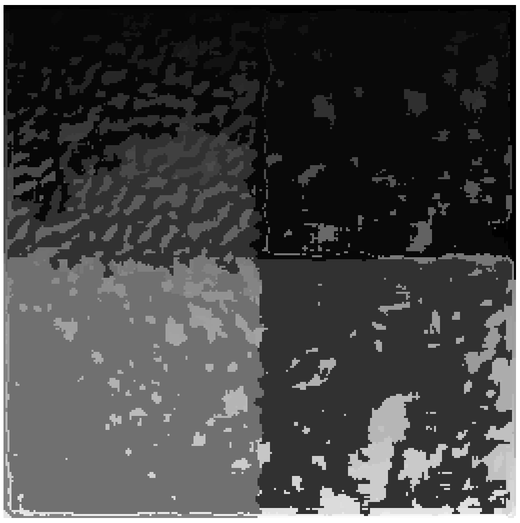

mos_2O_11W_64L_1.55T

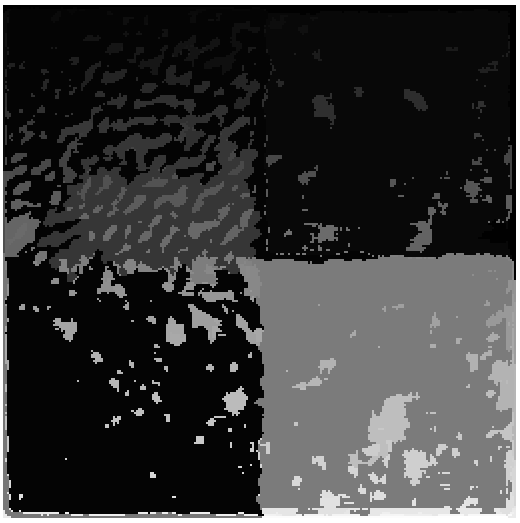

mos_2O_15W_32L_1.5T

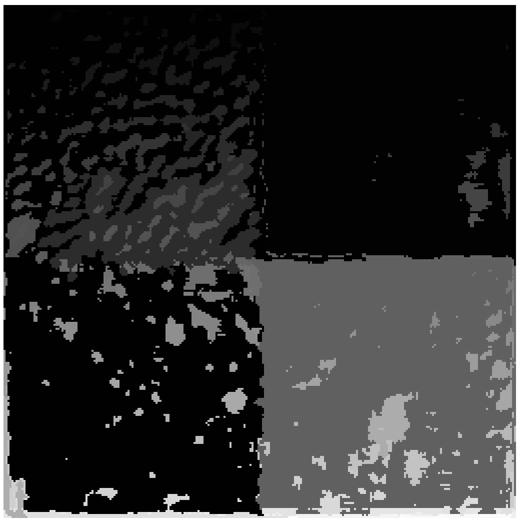

mos_2O_15W_32L_1.45T

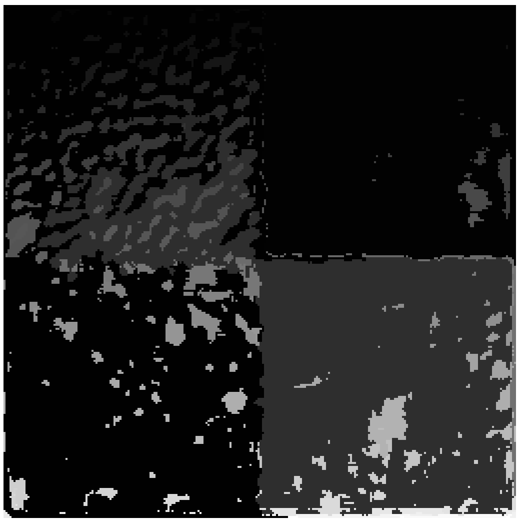

mos_2O_15W_64L_1.35T

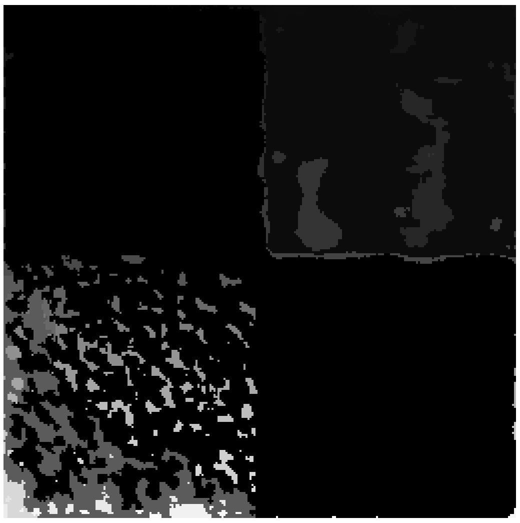

mos_2O_15W_64L_1.45T

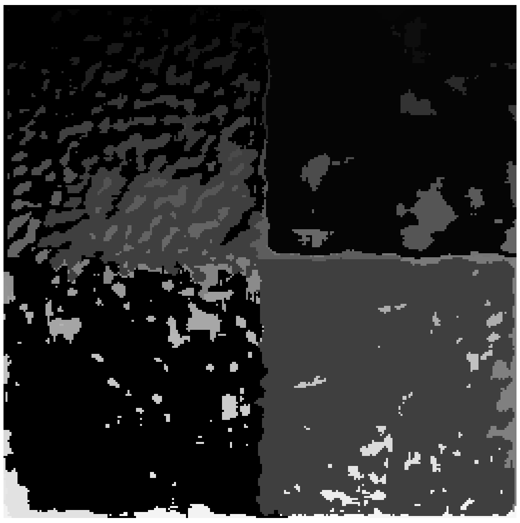

Mos_4O_11W_64L_1.3T

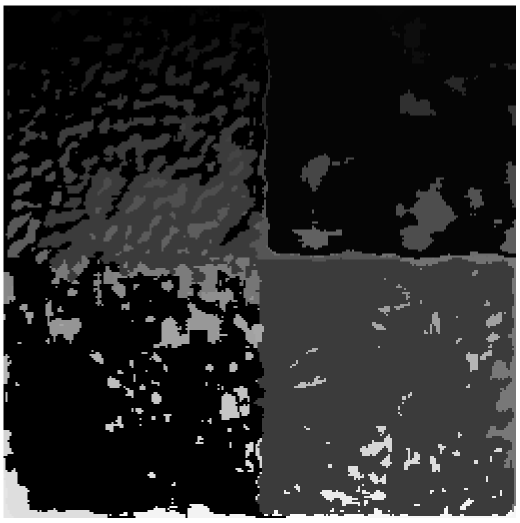

Mos_4O_11W_64L_1.45T


Mos_4O_11w_L32_1.50T


Mos_4O_11w_L32_1.53T


mos_4O_15W_32L_1.65T


mos_4O_15W_32L_1.57T


mos_4O_15W_64L_1.5T

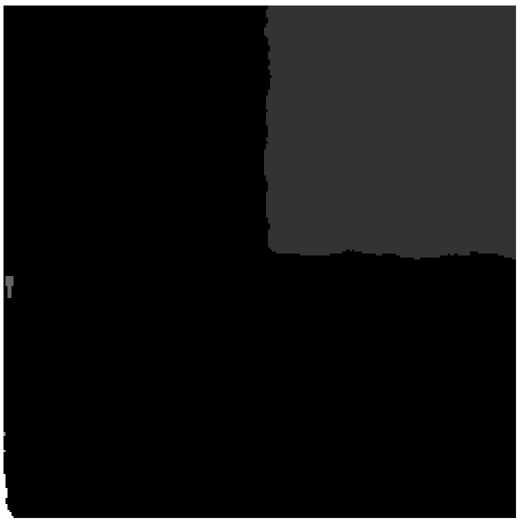
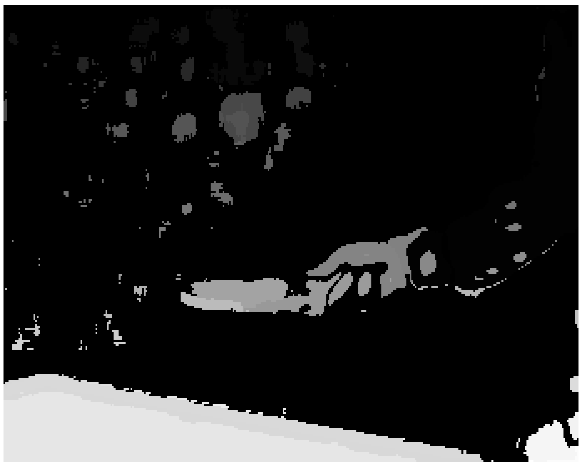
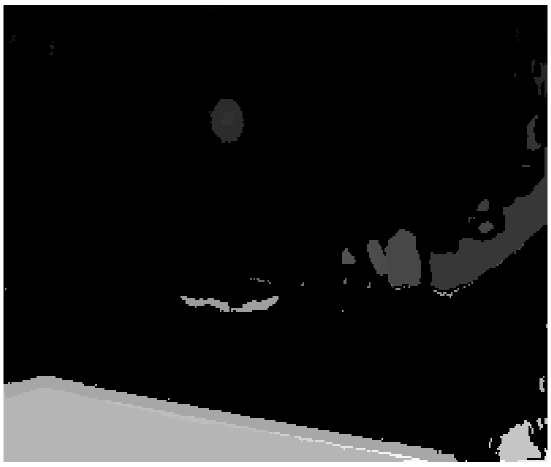
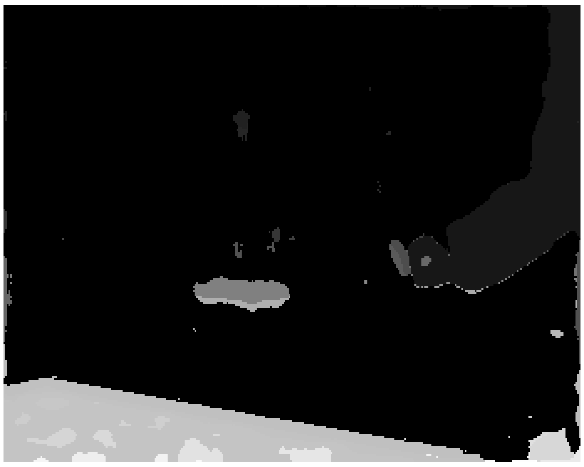
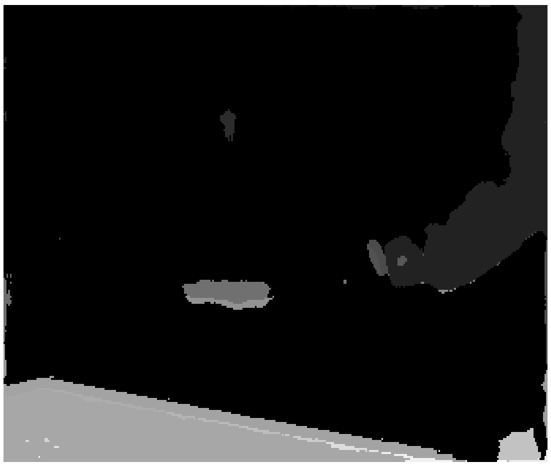

mos_4O_15W_64L_2.5T

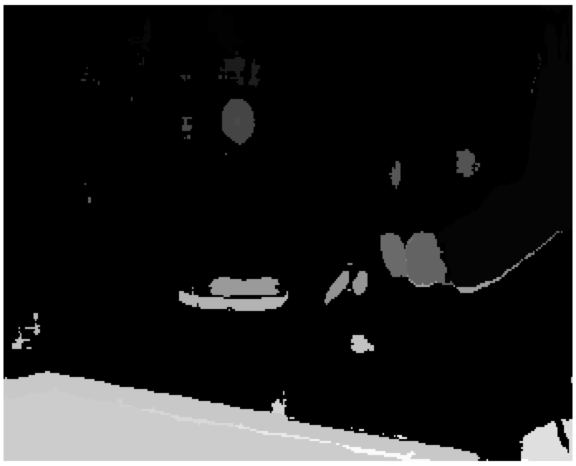
Figure 4: results of testing the segmentation algorithm on the mosaic image

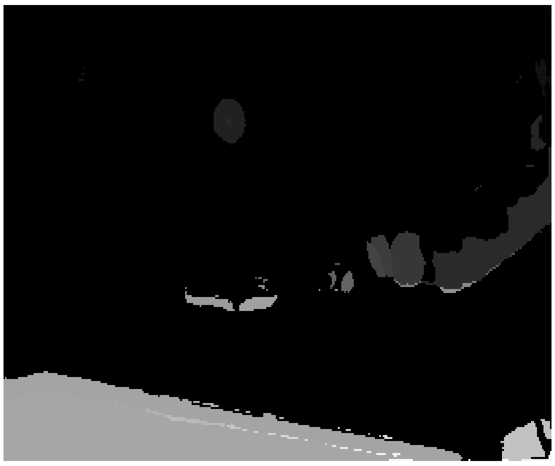

ping_2O_11W_32L_1.4T

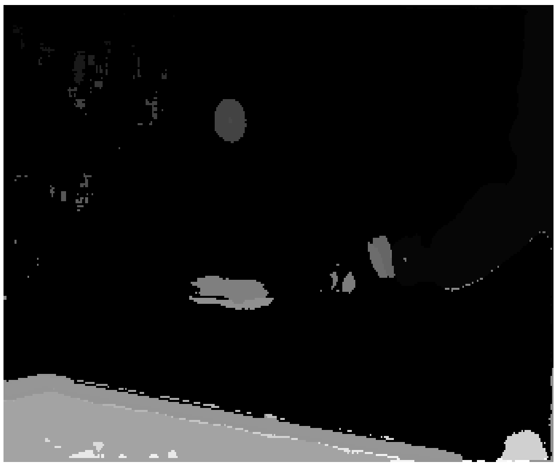

ping_2O_11W_32L_1.6T

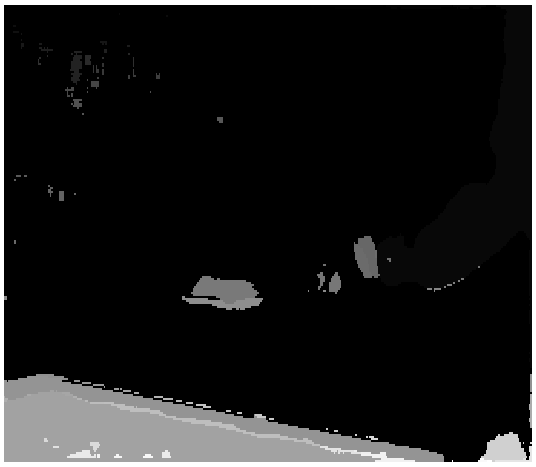

ping_2O_11W_64L_1.35T

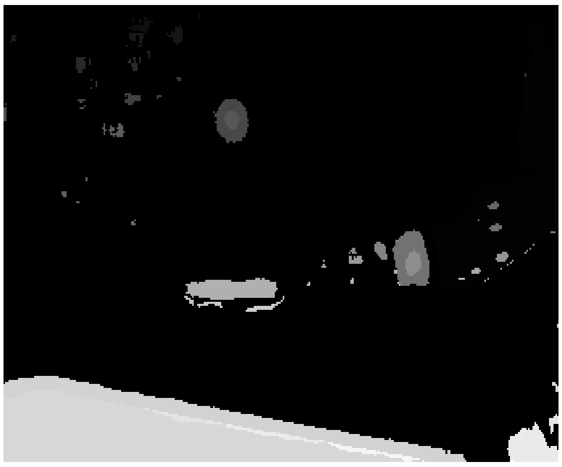

ping_2O_11W_64L_1.43T

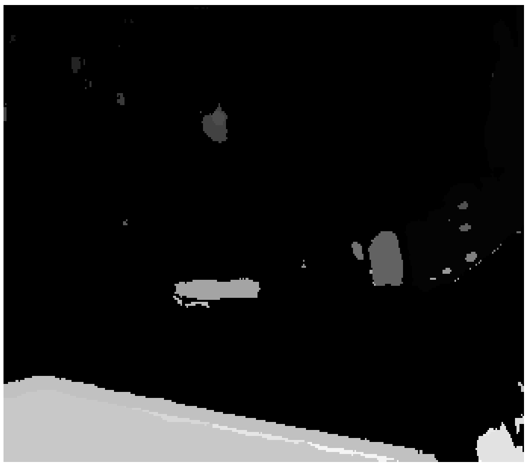

ping_2O_15W_32L_1.47T

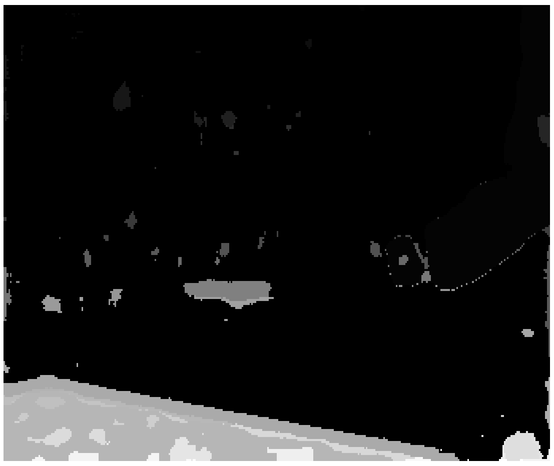

ping_2O_15W_32L_1.57T

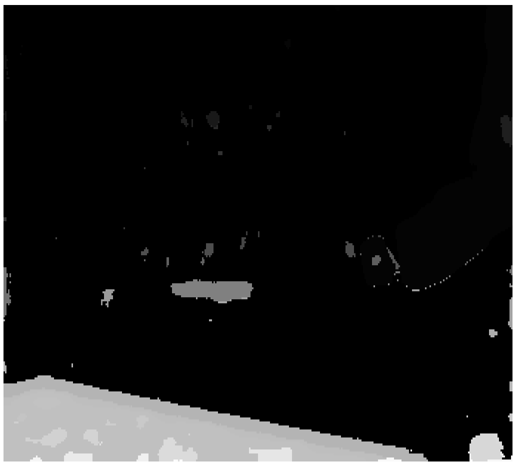

ping_2O_15W_64L_1.35T

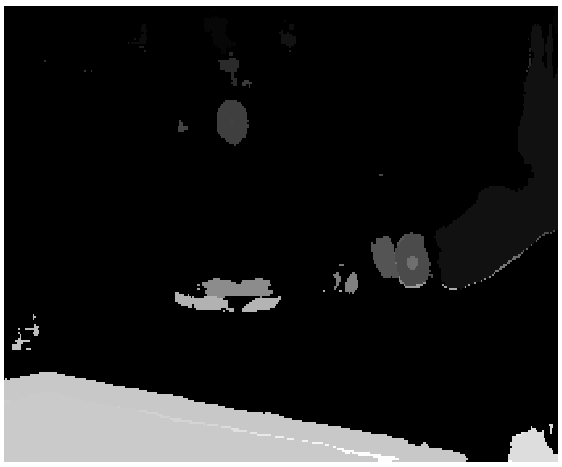

ping_2O_15W_64L_1.38T

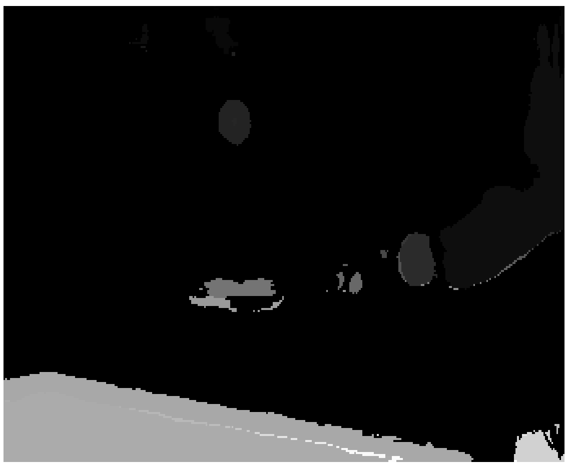

ping_4O_11W_32L_1.95T

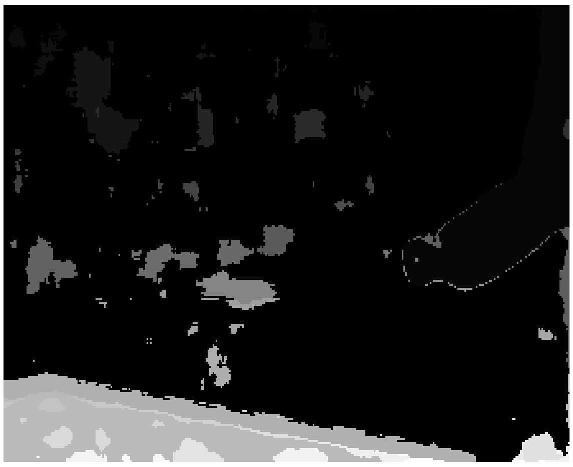

ping_4O_11W_32L_2T


ping_4O_11W_64L_1.7T


ping_4O_11W_64L_1.75T


ping_4O_15W_32L_1.95T


ping_4O_15W_32L_2T


ping_4O_15W_64L_1.63T

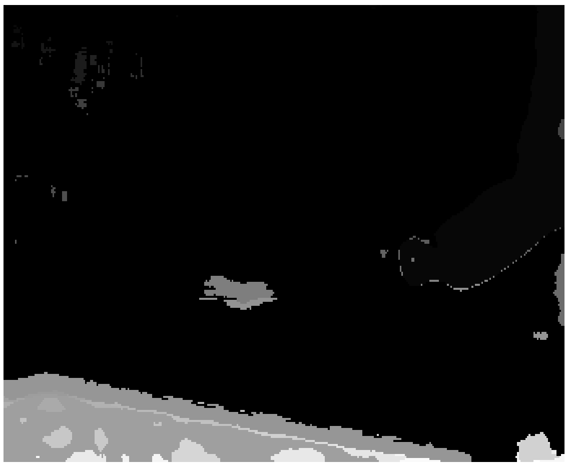
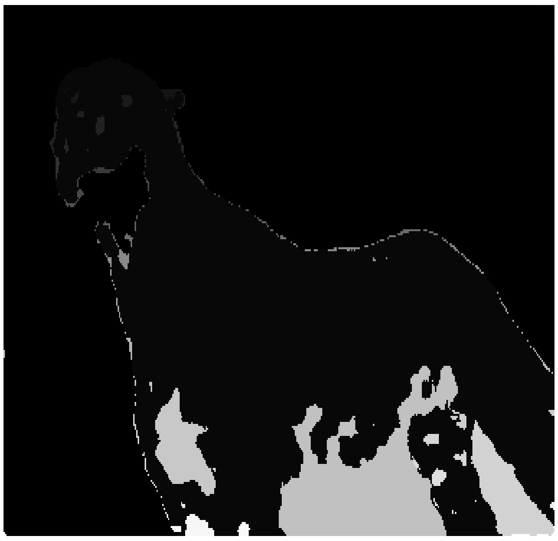
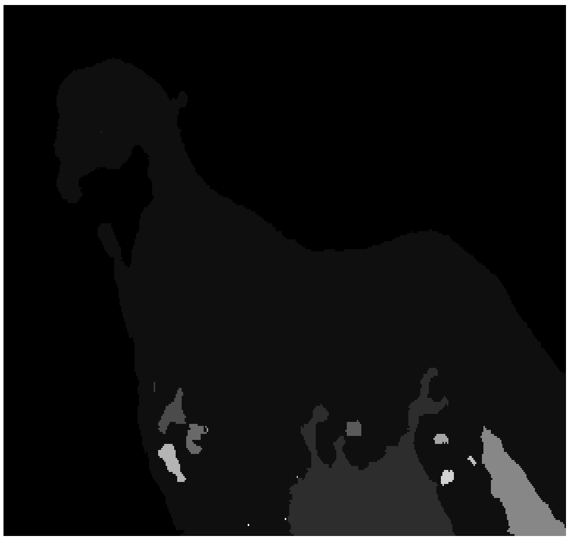
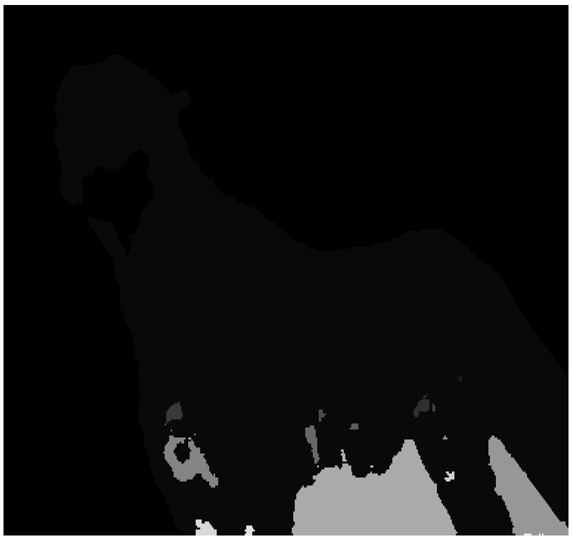
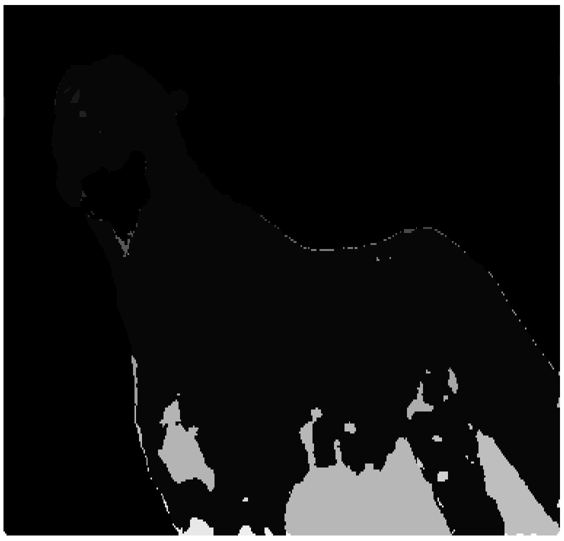

ping_4O_15W_64L_1.64T

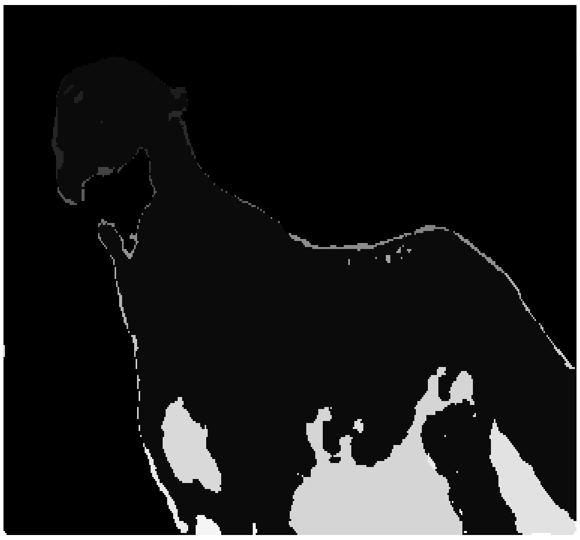
Figure 5: results of testing the segmentation algorithm on the ping pong image

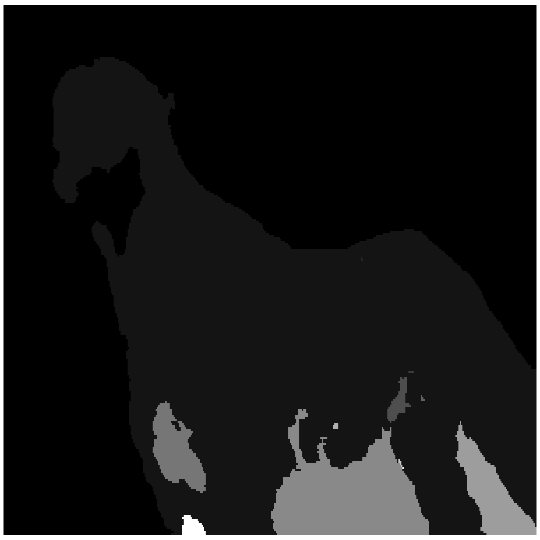

Feli_2O_11W_32L_1.75T

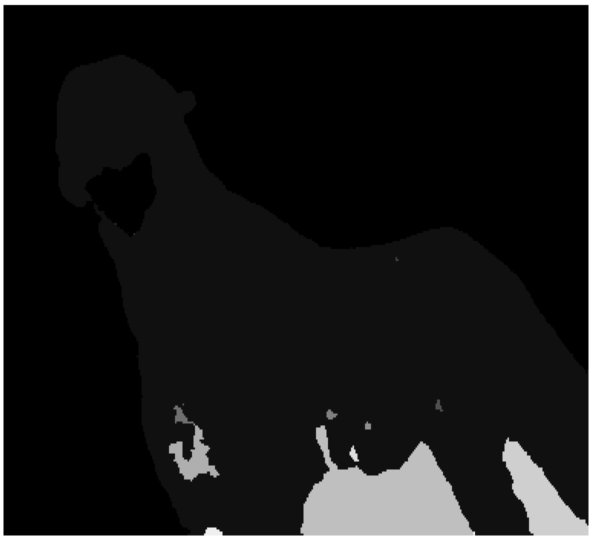

Feli_2O_11W_32L_1.95T

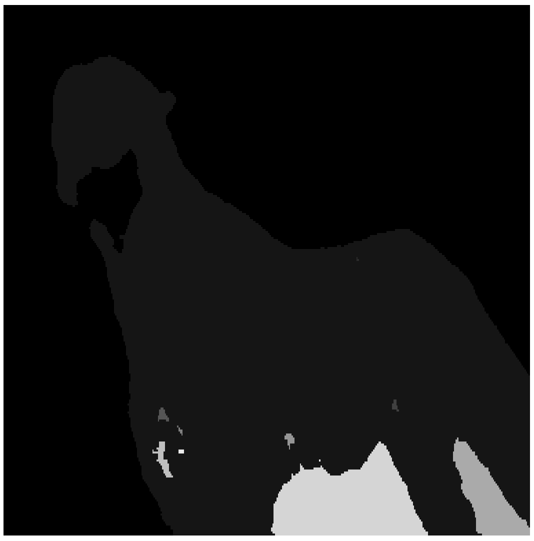

Feli_2O_11W_64L_1.9T

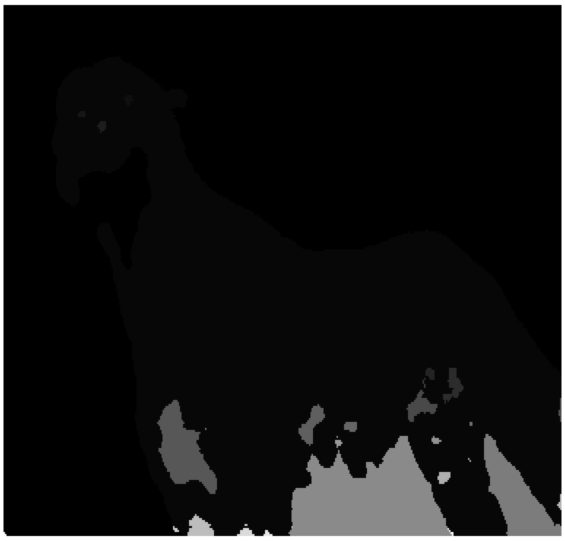

Feli_2O_11W_64L_1.65T

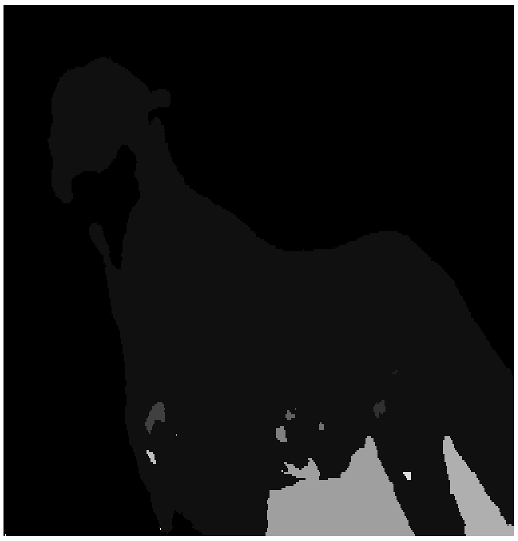

Feli_2O_15W_32L_1.7T

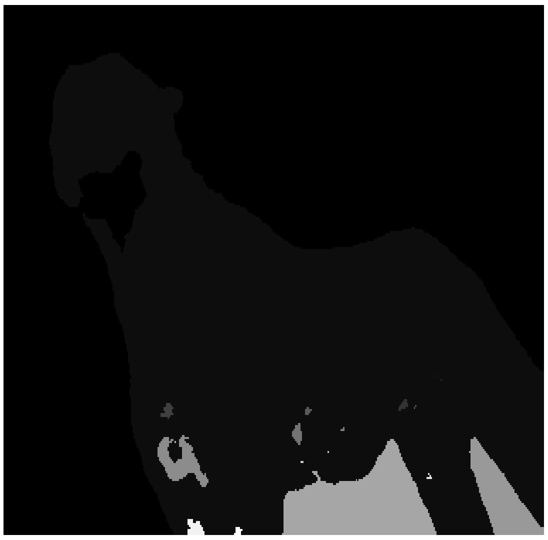

Feli_2O_15W_32L_1.9T

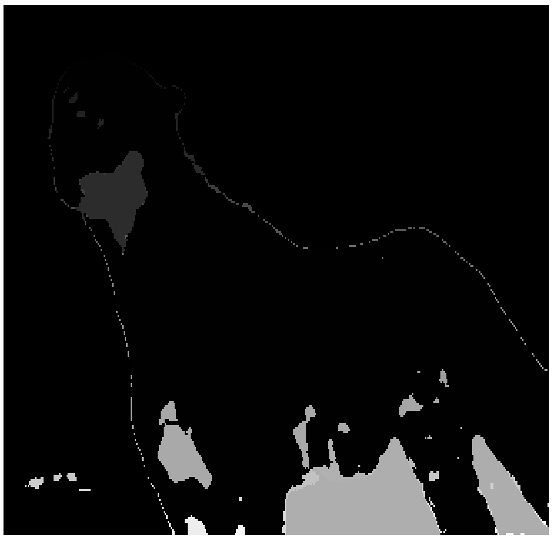

Feli_2O_15W_64L_1.8T

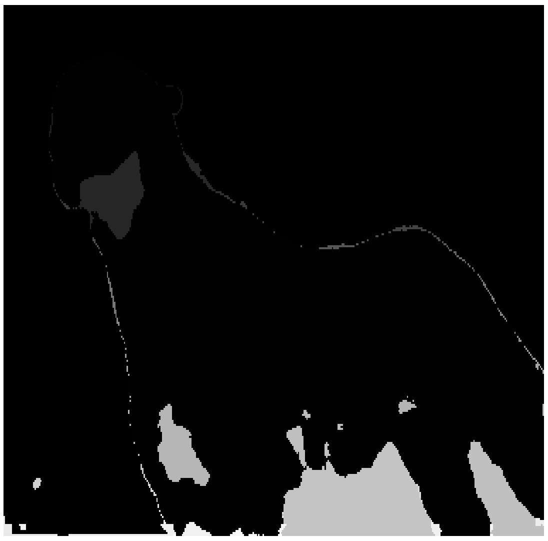

Feli_2O_15W_64L_2T

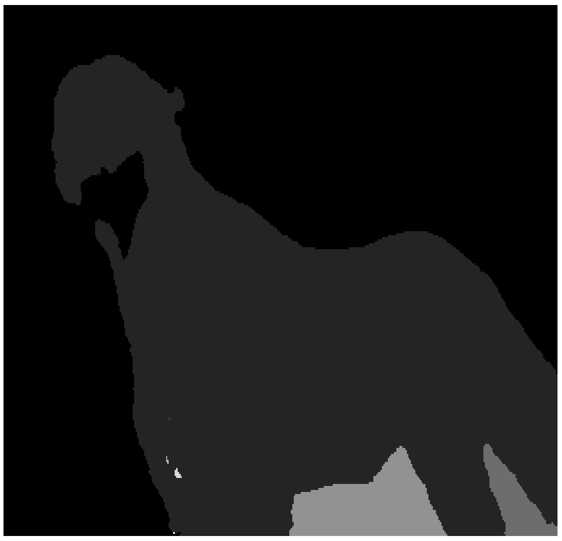

Feli_4O_11W_32L_2.6T


Feli_4O_11W_32L_2.8T


Feli_4O_11W_64L_2.5T


Feli_4O_11W_64L_2T


Feli_4O_15W_32L_2.2T


Feli_4O_15W_32L_2.8T

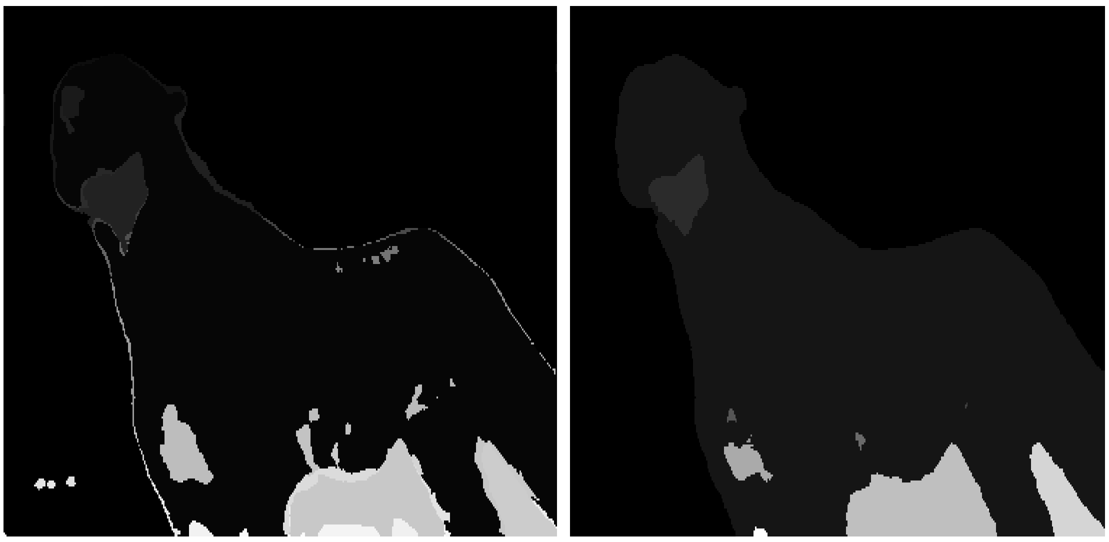


Figure 6: results of testing the segmentation algorithm on the Feli image

Looking at the images of figure 3 reveals a number of important characteristics. First, for every configuration of the varied parameters, the two chosen threshold values give different number of segmented regions depending on the image itself. Some of those images are over-segmented, others are more acceptable. However, in all of them the background, which is of a textural type, is segmented successfully as one region, as opposed to the case of working based on only color information. Secondly, increasing the number of the desired orientations has the effect of increasing the threshold value in order to get comparable results. This is because incorporating more orientations leads to feature descriptor with higher dimension. Furthermore, discontinuous regions like the one inside the ring has been segmented in some images as a part of the background because a tiny part of the ring is broken while the index finger is separated from the rest of the hand in all the presented images. Another noticeable feature is that putting up the value of the window size and the number of intensity levels has slightly better effect in terms of the boundary of the resultant regions but at the cost of much higher computational time. Therefore, choosing the lower values of parameters for the tested case would give very similar results. In order to comprehend the benefits more thoroughly, the following figure shows the result of segmenting the previous hand image using region growing algorithm but based on only color values:

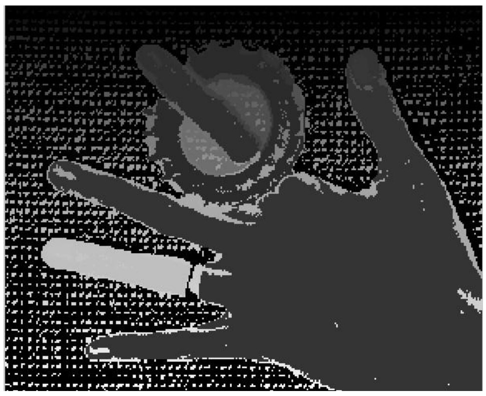


Figure 7: result of applying RG algorithm based on only color values

The previous figure clearly demonstrates the benefits of using textural information since the background here are segmented into a large number of tiny regions.

Visually speaking, the images in figure 4 have not been segmented into clear four separate parts even though a number of different parameter values were used. The top right square of this image has the best segmentation results among the other three. This can be understood well by inspecting the feature descriptor images. Figure 8 shows one of the resultant features images, energy, entropy, homogeneity, and contrast, for one chosen configuration.

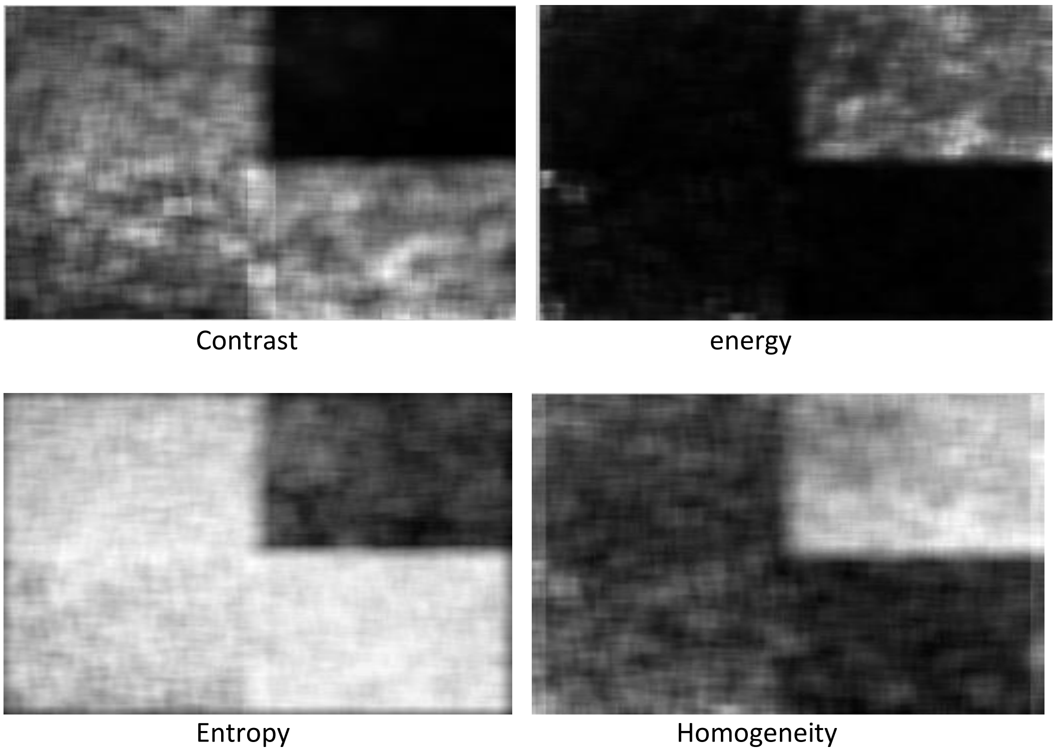


Figure 8: feature descriptors images for the mosaic image

It is evident from the preceding figures how the top right square has always distinct values in the four shown features than the others. The remaining regions have very similar values in the four presented images which explains why it is difficult to separate them entirely. Thus, choosing a quite high value of the threshold in some of the presented configurations, has the effect of merging the two diagonal squares first, because they are more similar than the third one, and then for even higher threshold the three are considered as one region. Again varying the used parameters within the chosen range for each one did not have a high impact on the quality of the segmented image, just higher time response of the algorithm.

The images in figure 5 and 6 give somewhat visually better results compared to the previous expectations. In these two cases, increasing the number of orientations and intensity levels improved rather the results in terms of the number of the segmented regions and the boundary smoothness.

Tables 1 to 4 demonstrate the same previous results along with the computational cost in terms of time.

Image Name	Number of Levels	Number of orientations	Window size	Time cost
ping_2O_11W_32L	32	2	11	391.334 s
ping_2O_11W_64L	64	2	11	541.692 s
ping_2O_15W_32L	32	2	15	313.666 s

ping_2O_15W_64L	64	2	15	591.893 s
ping_4O_11W_32L	32	4	11	425.173 s
ping_4O_11W_64L	64	4	11	425.173 s
ping_4O_15W_32L	32	4	15	577.031 s
ping_4O_15W_64L	64	4	15	949.405 s

Table 1: results of testing the ping_pong image

Image Name	Number of orientations	Window size	Number of Levels	Time cost
mos_2O_11W_32L-	2	11	32	371.468 s
mos_2O_11W_64L-	2	11	64	414.209 s
mos_2O_15W_32L-	2	15	32	246.328 s
mos_2O_15W_64L-	2	15	64	444.775 s
Mos_4O_11W_64L	4	11	32	422.173 s
Mos_4O_11w_L32	4	11	64	450.173 s
mos_4O_15W_32L-	4	15	32	344.547 s
mos_4O_15W_64L-	4	15	64	748.490 s

Table 2: results of testing the mosaic image

Image Name	Number of orientations	Window size	Number of Levels	Time cost
hand_2O_11W_32L-	2	11	32	334.656 s
hand_2O_11W_64L-	2	11	64	366.945 s
hand_2O_15W_32L-	2	15	32	351.232 s
hand_2O_15W_64L-	2	15	64	397.155 s
hand_4O_11W_32L-	4	11	32	421.727 s
hand_4O_11W_64L-	4	11	64	837.708 s
hand_4O_15W_32L-	4	15	32	420.460 s
hand_4O_15W_64L-	4	15	64	727.005 s

Table 3: results of testing the hand image

Image Name	Number of orientations	Window size	Number of Levels	Time cost
Feli_2O_11W_32L	2	11	32	579.023 s
Feli_2O_11W_64L	2	11	64	770.874 s
Feli_2O_15W_32L	2	15	32	595.786 s
Feli_2O_15W_64L	2	15	64	863.974 s
Feli_4O_11W_32L	4	11	32	515.173 s
Feli_4O_11W_64L	4	11	64	525.173 s
Feli_4O_15W_32L	4	15	32	600.031 s
Feli_4O_15W_64L	4	15	64	1289.267 s

Table 4: results of testing the Feli image

Inspecting the preceding tables carefully reveals that the time required for calculating the feature descriptors is highly dependent on the following parameters listed from highest to lowest time dependency in this example:

1- Image size.
2- The number of levels, which determines the GLCM matrix size.

3- Number of orientations to be considered while calculating the GLCM matrix.
4- window size.

Finally, the use of the texture information is quite significant as it is shown by the results presented in this section. Using color or intensity values alone is not sufficient to segment texture regions effectively and could lead to an image with high number of segmented regions and low number of pixels inside each of these regions when an algorithm like region growing based on colors is used, figure 7. Nevertheless, the use of those descriptors is at the cost of the computational cost. A region growing algorithm that benefits from texture features takes much more time to compute the results than just using color values. Indeed, the time cost could be reduced by choosing the appropriate values for the Offset, NumLevels, and Window_size parameters since increasing them is not always sufficient to obtain better results.

5 Design and implementation of the proposed solution for the classification problem

The second application in which texture information is employed and exploited is the classification of objects. The implemented Matlab files that carry out this task of classification work as follows:

The scr_classify file, actually, has the responsibility of building the training and testing sets. This is done by iterating through 20 folders. Each one of these folders contains 6 images with the same type of textural information. The first two images of each folder are incorporated in the training set. This is accomplished by calling another function called "ComputeFeatureVector", the aim of which is to extract the feature descriptors globally from these images. By globally, it is meant that the output of this function for each input image will give a vector of features of size 1*N, where N is the number of features. The same thing is achieved for the rest four images inside each folder but for building the testing set, by also extracting their global features using the "ComputeFeatureVector" function. Thereafter, the Matlab data are converted into weka sets, used for training a random forest classifier, and classifies the testing set. Finally, the scr_classify file returns the confusion matrix and the correct percentage of classification. The function that is used for doing the data set conversion is "convertWekaDataset".

The functions "graycomatrix" and "graycoprops" are called inside "ComputeFeatureVector" to calculate the GLCM matrix and the feature descriptors (energy, homogeneity, correlation, and contrast). Moreover, the entropy is computed separately and inserted in the feature vector to be used in the classification step. Thus, if all of the mentioned features are used with four different offsets, the size of the output vector will be 4*5=20 features.

6 Experimental section and results analysis for the classification problem

This sections summarizes the results of testing the classification functions described in the preceding section with distinct features and other "graycomatrix" function parameters. Table 5 presents these results with the various parameters.

Offset vector	Computational time (sec)	Correct classification
[0 1]	7.683 s	0.7625
[-1 0]	8.039 s	0.8375
[-1 -1]	8.132 s	0.8625

[-1 1]	6.798 s	0.9125
[0 2]	7.472 s	0.7500
[-2 0]	7.810 s	0.7375
[-2 -2]	8.356 s	0.7000
[-2 2]	6.715 s	0.6875
[0 1;-1 0]	10.659 s	0.8375
[0 1;-1 -1]	11.982 s	0.8625
[0 1;-1 1]	11.160 s	0.8125
[-1 0;-1 -1]	9.756 s	0.8875
[-1 0;-1 1]	11.517 s	0.7750
[-1 -1;-1 1]	12.330 s	0.8125
[0 1;-1 -1;-1 1]	14.267 s	0.8250
[0 1;-1 0;-1 1]	13.943 s	0.8125
[0 1;-1 1;-1 0]	13.941 s	0.8500
[-1 -1;-1 1;-1 0]	15.331 s	0.8875
[-1 -1;-1 1;-1 0;0 1]	15.879 s	0.8500
[-1 -1;-1 0;-1 1;0 1]	16.307 s	0.7625
[0 1;-1 0;-1 1;-1 -1]	17.208 s	0.8500
[-1 0;0 1;-1 1;-1 -1]	15.850 s	0.9125
[0 1;-1 -1;-1 1;-1 0]	18.116 s	0.9000

Table 5: results of applying the classification experiment.

Actually, a number of conclusions can be drawn from the previous table. First and foremost, increasing the size of the offset vector has the impact of raising the computational time and the accuracy in most of the studied cases. This is, in fact, an expected result because the growing in the offset vector size puts up the number of features and consequently more computations are required by the classifier to compare between them and better accuracy would be achieved. Secondly, increasing the distance inside the offset vector to 2 has slightly worse results. This depends on the nature of the texture under test and no general rules can be made from this. Additionally, changing the order of the rows inside the offset vector results in a different accuracies. Indeed, this is not a predicted thing. However, it can be referred to the nature of the random forest algorithm which seems to be reliant on the order the features. Another significant point to state is that in the previous table 5 different features were used. In case of less number of features, the accuracy and the computational time will change. Finally, Accuracy is not always a reliable metric for the real performance of a classifier, because it will yield misleading results if the data set is unbalanced (that is, when the number of samples in different classes vary greatly) which is not the case here.

7 Organization and development of the coursework

As a first step, it was essential for me to review what I had learnt previously in the lecture about image characterization using texture information. Then, I started to think about and write down the practical steps of the possible implementation. Initially, I implemented the segmentation part with just one feature descriptor and in one orientation, which did not add much information, just to be familiar at the beginning with the required computational time and try to inspect the resultant feature image in details. Next, I began to incorporate more features and taking into account some important steps like normalization at different steps and the Gaussian filter kernel characteristics. Therefore, the first session was spent to implement the first version of the segmentation part of this assignment and then I spent the week after that session for analyzing and testing more images. During the second session, I implemented the classification part and put the results in tables and figures for the sake of comparison. The final step was to write the report in order to explain what I had done so far.

8 Conclusions

During this lab sessions, I have successfully designed, analyzed and implemented a Region Growing algorithm and a classification procedure based on the texture information in Matlab program. A number of tests were carried out on different types and sizes of color images with distinct texture regions showing the significance of using textural features in two dissimilar application fields, segmentation and classification. Subsequently, comparisons were drawn analyzing and criticizing the results.