CMPE 465 COMPUTER VISION

IMAGE PROCESSING BASED CANCER CELL DETECTION

Department of Computer Engineering

TED University

Ankara, Turkey

1. Problem Definiton

With the development of technology, it involved in medical areas as well. For years scientists worked on the reasons behind the diseases and some of them are mispredicted. However, with the help of Image Processing features, It has become one of the mechanisms that help to diagnose cells effectively. Cancer is one of the severe diseases that dangers human life and it is essential the diagnose it before it's too late. So it is crucial to make the right diagnosis . That is why having high accuracy programs that help doctors will decrease the rate of mistakes. So usage of image processing will decrease the change of misdiagnosing. So we decided to choose a topic which will affect human life.

Morphological and functional characteristics of the malignant cell. Morphologically, the cancerous cell is characterized by a large nucleus, having an irregular size and shape, the nucleoli are prominent, the cytoplasm is scarce and intensely colored or, on the contrary, is pale. There are some ways to differ from malignant and benign cells. Cancerous cells have different behaviors than healthy cells. Their cell divisions are uncontrolled and grouped in a specific part of the cell. Also, their color, shapes, and distance between nuclei and epicyte(cell wall) differ from each other. The purpose of this project is to find out whether a cell is malignant or benign with the help of a data-set of CT images. The process begins with the acquisition of images and enhancement of them later continues with segmentation, feature extraction, and classification. We mainly aim to train cancerous cells concerning their colors and decide whether our test data include cancerous cells or not.

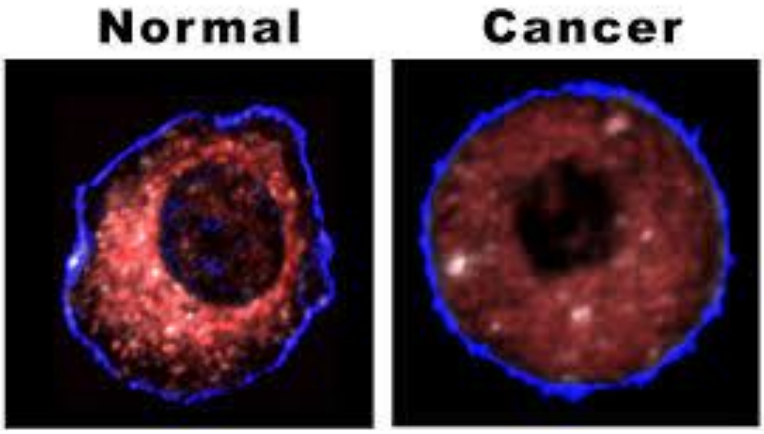

Figure 1 Difference between normal and cancerous cell

1.1 Background Research

There were many cancer detection researches for not only blood cancer but also lung, brain, breast, etc. However, the phases that these researches follow are almost the same. Most researches begin with the preprocessing the image because it requires to have a good quality image to get better results. Here are some examples that are related to our work.

1.1.1 Lung Cancer Detection Using Digital Image Processing Techniques

A research made by MEHWISH BARI from Mehran University about Lung Cancer Detection Using Digital Image Processing Techniques begins its preprocess phases with Spatial Domain Filtering to remove noises from the image and then Gabor and wiener filter is applied to enhance the image. The rest of the process continues with segmentation, feature extraction, and classification. In that research to segment the images k-means clustering is used. However, since there are different ways to segment an image it is not the only method as well. When the image is well-segmented most of the researches continue with the feature extraction phase. Feature are the key points of an image and it helps to define properties of the image as well and it can be done with different frameworks such as SIFT, HOG, etc. This phase aims to have a feature vector to determine the lung's features. Lastly, the classification part which classifies the labels. In his research lung's which has a tumor's features are extracted and trained and when a sample is taken it compares with trained data and it concludes whether it has a tumor or not.

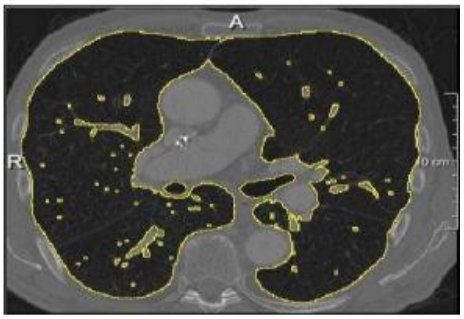

Figure 2 Extracted Features of Lung

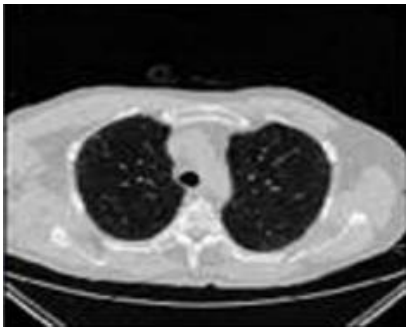

Figure 3 Threshold Applied Lung

Figure 4 Background Removed Lungs

1.1.2 Image Processing based Leukemia Cancer Cell Detection

Different research made by Ashwini Rejintal M. Tech in DEC, Department of ECE about Leukemia Cancer Cell Detection follows the same phases such as preprocessing, segmentation, feature extraction, and classification. Different from other researched GLDM and GLCM are used in the feature extraction phase also. Afterward, input images are first trained and then with the help of SVM it classifies the cancerous and non-cancerous cells then query images by other mean test images are send to SVM to figure out whether cells are cancerous or not.

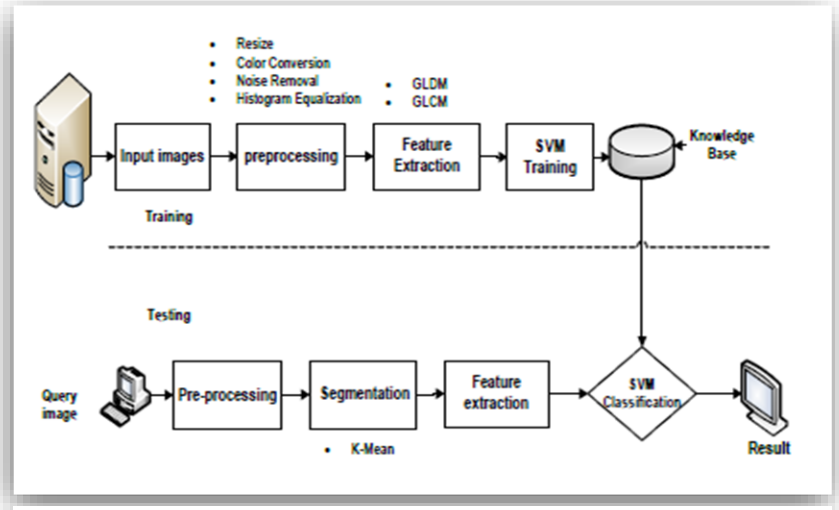

Figure 5 Lukemia cancer cell detection phases

1.1.3 Computer Aided Cancer Detection and Diagnosis Using Image Processing

The last research we examined is also similar to the previous ones. However different from other ones this research paper explains the different methods that can be applied in the segmentation part. It also helped us to determine which segmentation is most appropriate for our topic. With the help of this research paper, we concluded that usage of Watershed segmentation will fit our project because we first aim to clarify the cells in watershed segmentation and it is an edge-based segmentation type that will help us to find cell walls.

With these researches, we concluded that the most crucial part is to extract features and have a clear feature we need to have a good segmented image and train these feature vector and since Matlab provides SVM feature in itself we also decided to use SVM in our project. These researches helped us to create a plan and where to begin and how to begin also. Among these researches, we decided to take one about Leukemia Cancer Cell Detection as a model because before we begin to do some basic researches, the methods that we found seemed familiar with that one. Proposed System

2.1 Proposed System

2.1.1 Image Acquisiton

The first phase of the research began with the acquisition of images. It took a while to find the proper data-set because most of the images were mixed, there was no discrimination whether the cell is cancerous or not. However, during our researches, a proper data-set is found even the quality of the images is poor. Then prepared for the preprocessing phase.

2.1.2 Image Preprocessing

All the images have gone several preprocessing processes such as noise reduction, sharpening, enhancement and background removal to analyze cells in a better manner.

2.1.3 Noise Removal

Having clear cell images is essential for feature extraction so all these noises have to be removed from the image. First RGB images are converted into grayscale then the image contains noises such as white noise or salt and pepper are removed. In ımplementation first, by using fspecial function we created a gaussian filter and then that filter is applied to image with imfilter function.

2.1.4 Image Sharpening

This phase is most required because some of the cells were connected or blurred. When sharpening filter is applied it creates an image that has certain cell walls. It basically helps to determine the difference between cytoplasm and cells. In that part imsharpen method is applied to the image and its value is taken as 2.0 to make the cells clear.

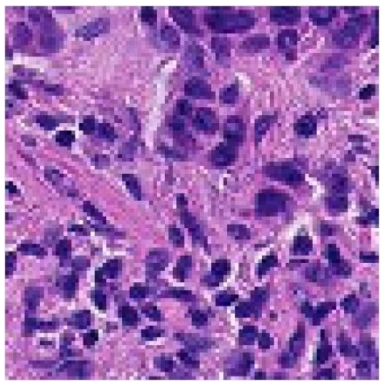

Figure 6 Original Image

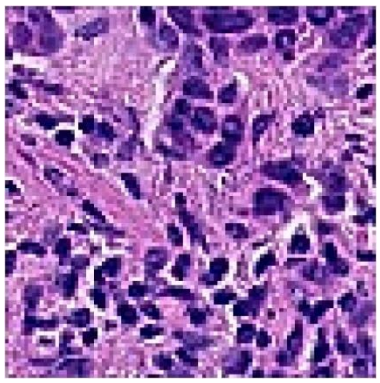

Figure 7 Sharpened Image

2.1.5 Image Enhencement

Image enhancement is called a way to improve the image's quality. It generally changes the image's contrast to have a clear image. Matlab's imadjust function gave a good result and made the cells more selectable. The first image is converted to greyScale and **imadjust** function is applied to it. It returns an image in an appropriate contrast.

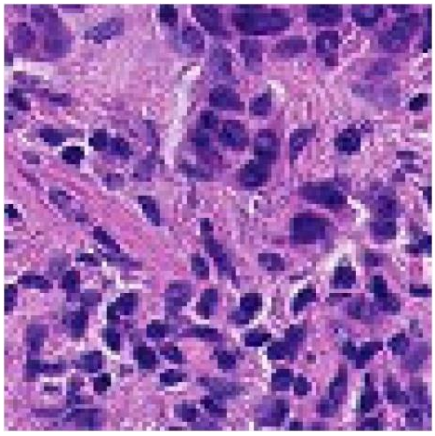

Figure 8 Original Image

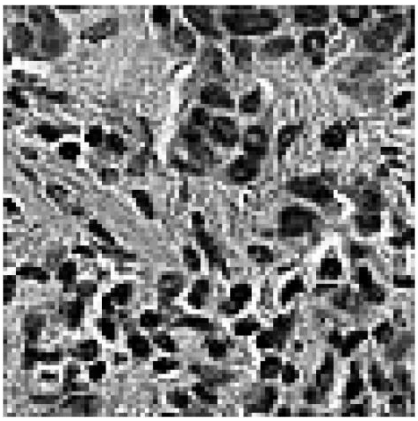

Figure 9 Enhenced Image

2.1.6 Background Removal

Essential part of removing the background is to find proper threshold value. To find the appropriate value otsu's method is used. Matlab's greythresh function computes a global threshold T from the grayscale image. Then the image is binarized background remains white and foreground zero also. After that image is transferred into RGB. In implementation first phase is to find the proper threshold value, with otsu's method it returned a level which is proper threshold value and image imbinarized with that

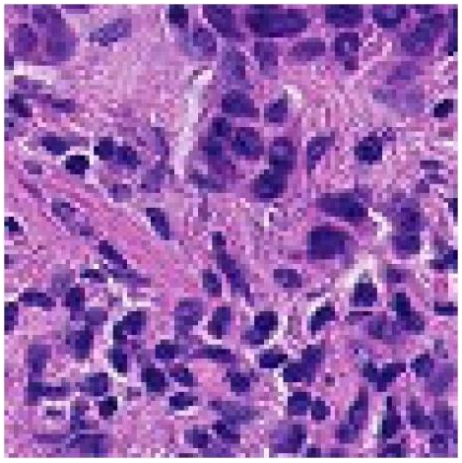

Figure 10 Original Image

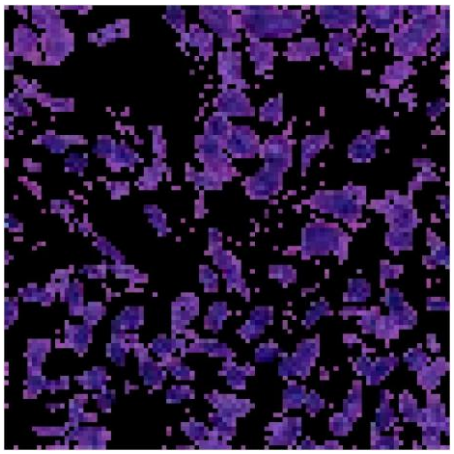

Figure 11 Background Removed Image

2.2 Processing Sta

2.2.1 Watershed Segmentation

In watershed segmentation image can be considered as a topographic surface. If we flood this surface from its minima and if we prevent the merging of the waters coming from different sources, we partition the image into two different sets.

The catchment basins and the watershed lines. If we apply the watershed transformation to the image gradient, the catchment basins should theoretically correspond to the homogeneous gray level regions of this image and applying this method provide to determine cells. However this method might cause over-segmentation problems and these issues can be handles by using opening-closing and SKIZ(skeleton by influence zones).

To prevent over-segmentation, we used morphological Opening-Closing by Reconstruction. Opening is erosion followed by dilation and Closing is dilation followed by erosion. After these operations, we had created a binary image ***bw.*** Using it we have calculated the ridge lines and distance transforms to be used in watershed. The results of some key steps can be seen in the figures(2-4) below.

After the watershed is calculated, each basin is labeled with a value to be used in separation in later steps. The resulting labels of watershed segmentation put over to the inputted preprocessed image can be seen in figure-5. Each different label is separated by using a different color to represent it on the image.

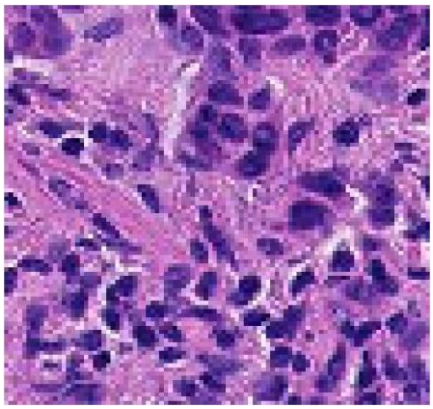

Figure 12 Original Image

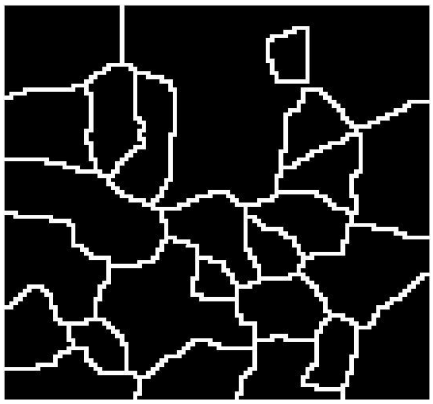

Figure 13 Ridge Lines using a different SKIZ method

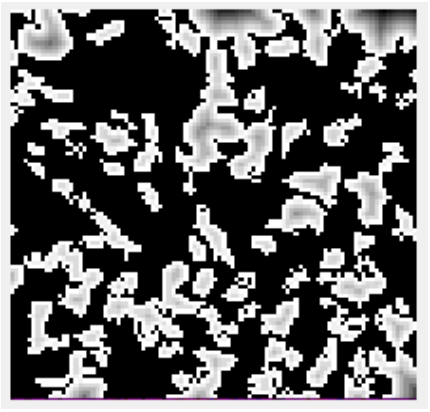

Figure 14 Distance Transform of the Image after applying Morphological Opening-Closing

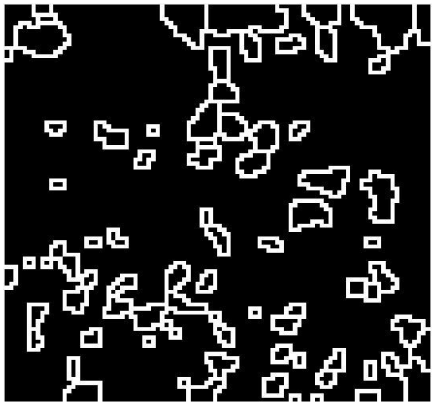

Figure 15 Ridge lines using SKIZ (this version is used in the code)

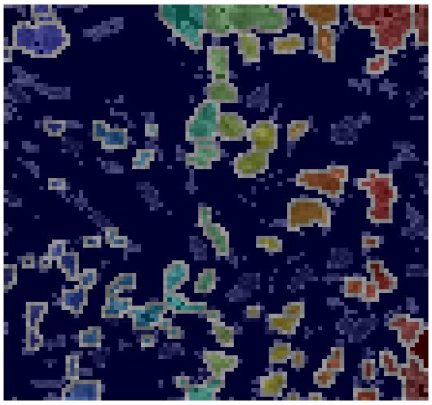

Figure 16 Watershed Applied on top of preprocessed image

2.3 Feature Extraction

2.2.1.1 Color Histograms

To get us started, and to give us an idea, one method was suggested to us for extracting the features from the image, using color histograms of images. This method used the entirety of the image instead of small segments of it. We compared the results coming from this method, and our initally proposed method to see wether we succeeded or not.

In this method, first we seperated each channel to its own array called ***Red***, ***Green***, and ***Blue,*** each one holding the intensity values for one channel. Next, we used imhist() method to create a histogram for each channel.

Last, these histogram values are concatanated using cat and they are stored to be used as our feature_vector in SVM

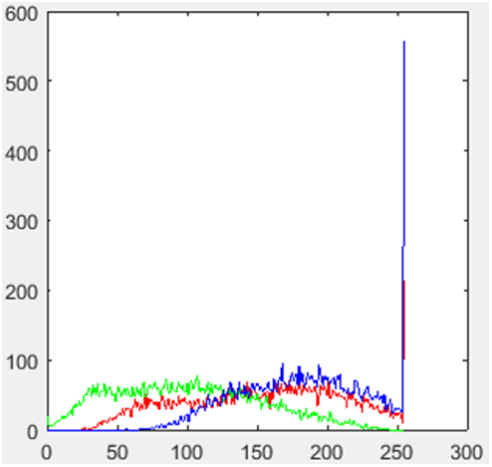

Figure 17 Histogram Graph of Image

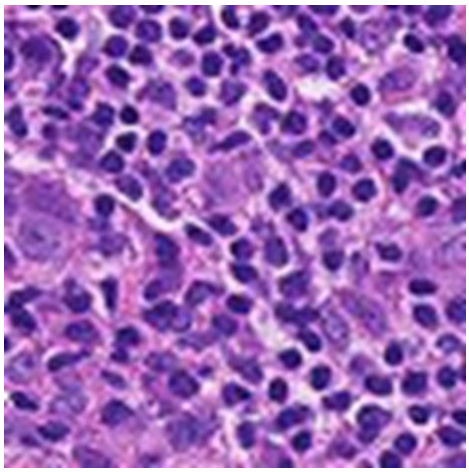

Figure 18 Original Image

2.2.1.2 Our Idea

Our idea was to select cells that appear to be malignant(cancerous) and extract features from these cells to be put to SVM. To do this, we required the segmented image coming from watershed segmentation step. After recieving the image, we seperated the cells and extracted the features from each cell segment.

After the watershed segmentation, most cells were parted from each other and they were labeled, so the individual cells were visibly divided and similar cells had the same labels. To seperate the cells from the image, we implemented a method, which recieved an image and the labels (coming from watershed step). Using the labels, each layer was copied to a new 96*96 image (96*96 is the fixed size of all images of the dataset). After the layers are seperated we had cells that had the same apperances

in the same image. These images were gathered in a M*96*96*3 array (M being the number of layers) called ***cells***.

The size of the images were 96*96*3 and most of this image contained pixels with 0 intensity(black), to reduce the size of our cells we removed each empty row and column until a non-zero intensity was found, using the ***find*** method provided by matlab.

Figure 19 Image size is reduced

Up to this point, layers are seperated and the sizes are reduced, so we moved to the feature extraction step. A cancer cell has many distinguishing attributes, color, size, shape, etc.. We started with color feature to have a general idea on what we were working with. According to our research, cancer cells had more saturated an intense colors, so we believed that the most saturated cell would be a cell that has cancer (if the image contains malignant cells).

To get the color feature, we calculated the average intensity values for each channel, for each cell segment, and divided that value with the overall average intensity values of the image that contained that cell. The obtained **feature_vector** was R,G,B (a feature_vector only contains values of one cell segment). For each cell segment, the feature extraction is repeated and each feature_vector is gathered in an array **feature_Vectors**.

After the features are extracted from each segment, they are compared among each other to find the cell with highest intensity values, and that cell is considered the main feature vector of the entire image. Since our data required only one pixel of cancer tissue to be present in an image (to be labeled as malignant), this approach seemed plausable.

After getting the feature vector from the selected cell, that vector is stored in an array ***traning_fetaure_vectors*** . This process is repeated for each image in the dataset.

2.3 SVM Classification

This is the final step for finding wether an image contains malignant cells or not. To classify our data, we used SVM(Support Vector Machine) since we only required a binary classification. This step uses the features extracted from our previous step, Feature Extraction, which gives out an array of features that contains same attributes for all images (3 color intensities R,G and B for our idea and a color histogram containing all the pixel values for the advised method). We used two functions for classifying our data. ***fitcsvm*** for traning an SVM model and ***predict*** for classifying a new image.

```
fitcsvm(trainingFeatures,trainingLabel,'Standardize',true,'KernelFunction','linear');
```

fitcsvm gets many inputs. First input is trainingFeatures, which is an M*N array where M is the number of images in our train data, and N is the size of our feature vector. Second input is the labels that correspond to each image (0 or 1). The ranking of each label is aligned with the trainingFeatures since each element in trainingLabel and trainingFeatures should correspond to the same image. The other inputs specify which methods are to be used to create a model, these inputs are explained in previous sections of this report.

When fitcsvm method finishes, a SVMModel is returned as a result. That result is our trained Classifier and it is ready for classifying unlabeled images.

For each image in our test data, we ran our preprocessing and processing methods. The result is a feature vector extracted from our test image. The resulting vectors are gathered in testingFeatures, a M*N array where M is the number of testing images and N is the size of feature vector for each image. The testingFeatures array is sent to the **predict** method, which takes 2 inputs. First input is the SVMModel that we trained previously. Second input is the testingFeatures array.

Predict method uses the trained classifier to classify each element of testingFeatures to label the corresponding image as malignant(1) or benign(0). The outputs of predict is test_label_answers which are the predicted labels for each image.

After the predicted labels are obtained, we compare the real values(we have the actual labels for each image) and calulate our accuracy rate. The success rates are discussed in Results section.

2.4 Results

2.4.1 Dataset

Our dataset was obtained from Kaggle.com, Histopathological Cancer Detection. This dataset was part of a kaggle challange and contained more than 200,000 96*96 rgb images. There were 2 categories of images, training and test. The training set contained 220,025 images. We did not use the test images since we had more than enough images in the training set. Since each training image had a label, we were able to calculate our success rate.

In the dataset there are various types of images. There are images that has hundereds of cells while some of them had 5-10. Also, the images does not have the same color variance, as some images have very low saturation others are have very bright and distinctive colors. The background is always white (due to the nature of microscopic images).

The images were labeled as healty(0) and cancerous(1), these labels were given for the entire image and only if the central 32*32 area contained cancer, the image was labeled as cancerous. The labels being given for the entire image made things more challenging as we tried to find a cell and extracted its features.

The seperation of cancerous cells and normal ones are very challenging and as our results show, cannot be decided solely on color intensites because all of the images contain purple,dark-purple cells, but not all of them are cancerous. Our assumption of intense color values meaning cancer, lost its viability after this observation.

2.4.2 Success Rates and Accuracy of Segmentation

We used two different approaches to see if we could manage to find a reasonable success rate. In both approaches, we have trained the data using 5388 images and tested them with 100.

One approach is creating a color histogram using the whole image and sending that histogram as a fetaure vector. The other approach is seperating cells from each other and finding one with the highest color intensity.

First approach was to create a color histogram of the entire image, seperating each channel. Then concatanating the histogram values together to be used as a fetaure vector representing that image.

This approach yielded a success rate of %20. Most images(with many cells) were considered as cancerous even though they were normal cells. This made us understand that there were other attributes that needed to be considered when deciding and that color intensities were not enough on its own.

Second approach, which was our main idea of how we were going to find whether an image contained cancer or not, was to apply watershed segmentation on the image and find a cell that was most likely a cancerous cell.

First problem occured in watershed segmentation part, the segmented images were very small in size (96*96 px) so the pixels that belonged to 2 different cells were too close, this made the cells to be grouped as one cell. The other problem was our method of using watershed, was not able to find all the cells.

Our approach yielded a success rate of %50. There were many false positives using our approach (normal images classified as cancerous), this also showed us that our approach was wrong and needed to consider more attributes such as number of cells clustered together and the texture of cells (which are 2 very distinctive features of cancerous cells). Since we used color intensities to find cancerous cells, and our dataset had very similar looking cells(according to their colors), our approach classified most of the cells as cancerous.

2.5 Method

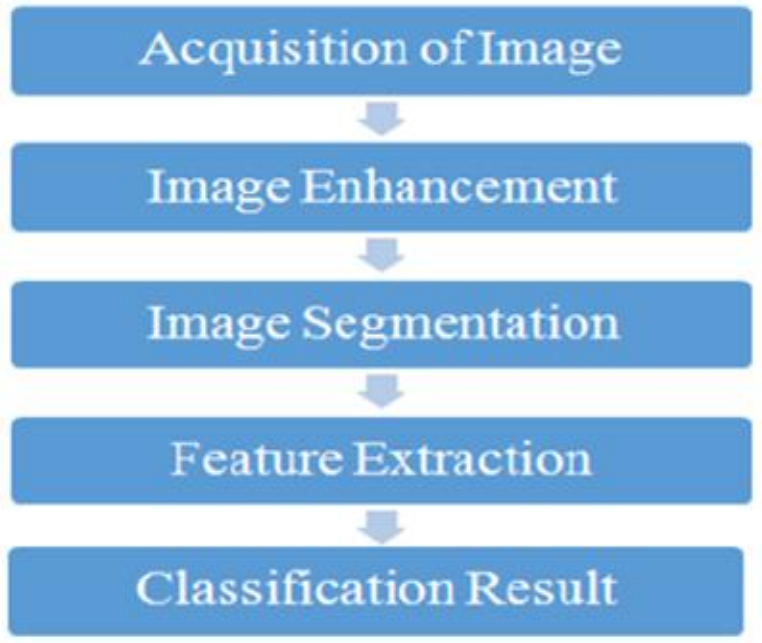

Figure 20 Description of steps

2.5.1 Acquisitionof Image

In this part, we look for the most appropriate data for our research. Most essential phase of the research was the find data which include cancerous cell because having only cell images does not help us in the training part. It was hard to find proper data, only useful data that we could find was lack of pixels but it was grouped with malignant and benign cells so we decided to choose it as our data-set

2.5.2 Image Enhencement

This part is consist where Image preprocessing is applying to the image. Pre-process stage consist of 4 stages, noise reduction, image sharpening, image enhancement and lastly background removal.

Noise Reduction

fspecial(type)=creates a two-dimensional filter h of the specified type. Some of the filter types have optional additional parameters, shown in the following syntaxes. **fspecial** returns h as a correlation kernel, which is the appropriate form to use with **imfilter**

Gaussian filter is created an applied to image with imfilter function.

imfilter(A,h)= filter(A,h) filters the multidimensional array A with the multidimensional filter h and returns the result in B. You optionally can filter a multidimensional array with a 2-D filter using a GPU

Image Sharpening

imsharpen(A,Name,Value)= uses name-value pairs to control aspects of the unsharp masking.

sharpenedImage = imsharpen(noiseRemovedImage, 'Amount', 2.0);

In order to specify the image more, 2.0 value is taken.

Image Enhencement

First image is converted to grayscale and then imadjust function is applied

imadjust(I) = maps the intensity values in grayscale image I to new values in J. By default, imadjust saturates the bottom 1% and the top 1% of all pixel values. This operation increases the contrast of the output image J.

Background Removal

Otsu's method is used to have threshold value **graythresh**(I) computes a global threshold T from grayscale image I then converts the image into a binary image using the threshold with **imbinarize** function

2.5.3 Image Segmentation

imdilate = imdilate(I,SE) dilates the grayscale, binary, or packed binary image I, returning the dilated image, J. SE is a structuring element object or array of structuring element objects, returned by the strel or offsetstrel functions.

İmerode = imerode(I,SE) erodes the grayscale, binary, or packed binary image I, returning the eroded image, J. SE is a structuring element object or array of structuring element objects, returned by the strel or offsetstrel functions.

İmcomplement: J = imcomplement(I) computes the complement of the image I and returns the result in J. J= Image Complement

İmregionalmin : imregionalmin(I) returns the binary image BW that identifies the regional minima in grayscale image I. Regional minima are connected components of pixels with a constant intensity value, surrounded by pixels with a higher value.

İmclose : imclose(I,SE) performs morphological closing on the grayscale or binary image I, returning the closed image, J. SE is a single structuring element object returned by the strel or offsetstrel functions.

İmopen : imopen(I,SE) performs morphological opening on the grayscale or binary image I, returning the opened image, J. SE is a single structuring element object returned by the strel or offsetstrel functions.

Strel: strel('disk',r,n) creates a disk-shaped structuring element, where r specifies the radius and n specifies the number of line structuring elements used to approximate the disk shape.

Bwareaopen : bwareaopen(BW,P) removes all connected components (objects) that have fewer than P pixels from the binary image BW, producing another binary image, BW2. This operation is known as an *area opening*.

İmimposemin: imimposemin(I,BW) modifies the grayscale mask image I using morphological reconstruction so it only has regional minima wherever binary marker image BW is nonzero.

Watershed : returns a label matrix L that identifies the watershed regions of the input matrix A

İmgradient : imgradient(I) returns the gradient magnitude, Gmag, and the gradient direction, Gdir, of the 2-D grayscale or binary image I.

label2rgb : label2rgb(L) converts a label matrix, L, such as those returned by labelmatrix, bwlabel, bwlabeln, or watershed, into an RGB color image for the purpose of visualizing the labeled regions.

watershed_segmentation(rgbimage)

outputs : [L, lrgb]

L = result of applying watershed to distance transform

Lrgb = labels got from L, using label2rgb

Gets an rgb image and applies Opening-Closing operations to the image. Then the resulting image is binarized.

The distance transform is taken and watershed method is used on the distance transform. Output of watershed transform is eroded and assigned to a variable called labels

Cell_seperate(image, labels)

Outputs : cellArray

İmage = the image to be cell seperated

Labels = output lrgb of watershed_segmentation

cellArray = all the seperated cells gathered in one array (96*96 images with most pixels being blank(black))

Gets the labels and the image from previous steps and seperates the image according to the labels, each pixel having the same label is gathered and put on a new image. Those new images are stored in the cellAray.

2.5.4 Feature Extraction

Feature_extract(cells)

Outputs : feature_vector

Cells = cellArray from cell_seperate function

Feature_vector = the average color intensities of each image in the cells array.

Gets the cellArray from cell_seperate and reduces their size using remove_zero_intensity. Then the average color intenisites are calculated for each element of cells array. The features extracted from each cells element is put to the feature_vector to be sent to next function.

Remove_zero_intensity(img)

Outputs : nonZeroImage

nonZeroImage = size reduced version of img

Gets an image and removes the unused parts of the image (the black paddings formed when the cells are seperated) using the find method. The result is returned

Find_relative_cells(image, cellArray)

Outputs : relativeCells

relativeCells = an array of cells that are inside the central 32*32 pixel area.

Gets a cell array and selects the ones that have a pixel that is inside the specified area.

The relative cell images are put to the relativeCells array.

Select_candidate_feature(feature_vectors)

Outputs : selection

Selection = a 1x3 feature vector of color intensities (for each RGB channel)

Gets the feature_vector created in feature_extract function and selects the one with most appropriate feature value. Returns the selected feature.

İmhist : imhist(I) calculates the histogram for the grayscale image I. The imhist function returns the histogram counts in counts and the bin locations in binLocations.

Cat : cat(dim,A,B) concatenates B to the end of A along dimension dim when A and B have compatible sizes (the lengths of the dimensions match except for the operating dimension dim).

2.6 Classification

Predict : redict(filter) returns the predicted state, xpred, and the predicted state estimation error covariance, Ppred, for the next time step of the input tracking filter.

Table2array : table2array(T) converts the table, T, to a homogeneous array, A.

Fitcsvm: returns a support vector machine (SVM) classifier Mdl trained using the sample data contained in the table Tbl. ResponseVarName is the name of the variable in Tbl that contains the class labels for one-class or two-class classification.

2.7 Conclusion

As conclusion, we have seen that our approach for detecting cancer in an image was faulty and needed improvement. Some of our research led us to believe cancerous cells had more intense color values and therefore it could be used to detect cancer. This idea was proven wrong according to our results and the discussions we have made. It showed that color values are not a direct indicator of cancer(alone).

With this project we have seen the importance of selecting correct features for determining results. We have also seen the capabilities of SVM while dealing with a classification problem. We have used different techniques for applying watershed algorithm and seen how these different approaches give different results.

From this point on, we are want to improve this project by adding more in depth feature extraction techniques for determining the malignancy of an image. We also want to improve the current implementation of watershed to be able to find and seperate all of the cells in an given image. We want to improve because by adding more functionalites we will be learining about how to do those things.

2.8 References

Jain, Sanjeev & Patil, Bhagyashri. (2014). Cancer Cells Detection Using Digital Image Processing Methods. International Journal of Latest Research in Science and Technology. VOLUME 3. 45-49.

International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887 Volume 7 Issue IV, Apr 2019- Available at

Jamil, Mutiullah & Bari, Mehwish & Ahmed, Adeel & Sabir, Muhammad & Naveed, Sajid. (2019). Lung Cancer Detection Using Digital Image Processing Techniques: A Review. 10.22581/muet1982.1902.10.

https://www.kaggle.com/c/histopathologic-cancer-detection