A REPORT

ON

CERVICAL CANCER DETECTION USING MACHINE LEARNING, TOOTH CAVITY SEGMENTATION USING IMAGE PROCESSING & INTERFACING NEOPIXEL RING USING ADAFRUIT FLORA V3

BY

PIYUSH MISHRA	**2016B2AA0633H**
PRANSHU KABRA	**2016B3A70595H**

AT

CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE, PILANI

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

JULY 2018

A REPORT
ON

CERVICAL CANCER DETECTION USING MACHINE LEARNING, TOOTH CAVITY SEGMENTATION USING IMAGE PROCESSING & INTERFACING NEOPIXEL RING USING ADAFRUIT FLORA V3

BY

PIYUSH MISHRA	**2016B2AA0633H**	**MSC. CHEMISTRY + ELECTRONICS AND COMMUNICATION**
PRANSHU KABRA	**2016B3A70595H**	**MSC. ECONOMICS+COMPUTER SCIENCE**

Prepared in partial fulfilment of the Practice School-I Course
AT

CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE, PILANI

A Practice School-I station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
JULY 2018

Acknowledgements

We would like to thank Dr. Santanu Chaudhury (Director, CEERI) and PS Division, BITS Pilani for giving us the opportunity to work in a top-notch research organization like CEERI. We are grateful to Dr. J.L. Raheja for guiding us through the projects and giving us insights into his own work. We would also like to thank Ms. Shalini Nehra who was our mentor for these projects and has constantly guided us and helped us get past the roadblocks. We are also grateful to Mr. Pawan Sharma, our PS-I instructor whose constant advice and support throughout the program has made the PS-1 experience a much better learning experience. Finally, we would like to thank Dr. S. Murugesan who has been our PS instructor in the absence of Mr. Pawan Sharma.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)
Practice School Division

Station: CENTRAL ELECTRONICS ENGINEERING RESEARCH INSTITUTE

Centre: PILANI

Duration: 1 MONTH   Date of Start: 22/05/18   Date of Submission 20/06/18

Titles of the Projects: CERVICAL CANCER DETETION USING MACHINE LEARNING,

TOOTH CAVITY SEGMENTATION USING IMAGE PROCESSING &

INTERFACING NEOPIXEL RING USING ADAFRUIT FLORA V3

PIYUSH MISHRA	2016B2AA0633H	MSC. CHEMISTRY + B.E. ELECTRONICS AND COMMUNICATION ENGG.
PRANSHU KABRA	2016B3A70595H	MSC. ECONOMICS + B.E. COMPUTER SCIENCE AND ENGG.

Name and designation of the expert: DR. J.L. RAHEJA (Head, Control and Automation group, CEERI Pilani)

Name of the PS Faculty: MR. PAWAN SHARMA

Key Words: CERVICAL CANCER, FEATURE EXTRACTION, SVM, GABOR FILTER, TOOTH CAVITY, THRESHOLDING, MICRO-CONTROLLER, ARDUINO

Project Areas: IMAGE PROCESSING, MACHINE LEARNING, ARDUINO

PROJECT 1 - CERVICAL CANCER DETECTION USING MACHINE LEARNING

Abstract: Cervical cancer, being the fourth most common women cancer in the world, researchers throughout the world are gravely concerned for finding a solution to it so that an early and timely diagnosis of this disease could be realized and as such the overall mortality rate could be mitigated. In this project we worked on building an SVM classifier using Visual Studio 12 configured with OpenCV 2.4.6. The input to our classifier is a photograph of the cervix taken through a vaginal speculum. Based on the features extracted from the images, the classifier predicts whether the image is of a cancerous cervix or a non-cancerous one. The model achieved an accuracy of 95.56% on the test set comprising of 45 images (25 cancerous+20 non-cancerous).

Signatures of Students	Signature of PS Faculty
Date:	Date:

CONTENTS

1. Introduction

Uterine Cervical Cancer is one of the most common forms of cancer in woman worldwide. It is ranked 11th in incidence and 13th in mortality in the developed countries, due to the ability to detect the precancerous lesions through government-sponsored Cervical Cancer Screening Program. This success in the developed countries has been achieved through a synergy between Cervical Cytology Screening for Cervical Intra-epithelial Neoplasia (CIN) before they become invasive and their effective treatment directed by colposcopy.

Colposcopy is a very effective, non-invasive diagnostic tool. In colposcopy, the cervix is examined non-invasively by a colposcope which is a specially designed binocular stereomicroscope. The abnormal cervical regions turn to be white after application of 5% acetic acid and are called Acetowhite (AW) lesions which are then biopsied under colposcopic guidance for confirmation by histopathological examination. Modern colposcopes can produce a digital image of the cervix. Colposcopy today is considered the gold standard for detection and treatment of pre-cancerous lesions of the cervix. However, there is currently a void in specialized image processing software which has the ability to process images acquired in colposcopy. Nevertheless, trained personnel are required for evaluation of the results.

We have presented a novel feature screening algorithm by deriving relevance measures from the decision boundary of Support Vector Machines.

The proposed relevance measures have several advantages:

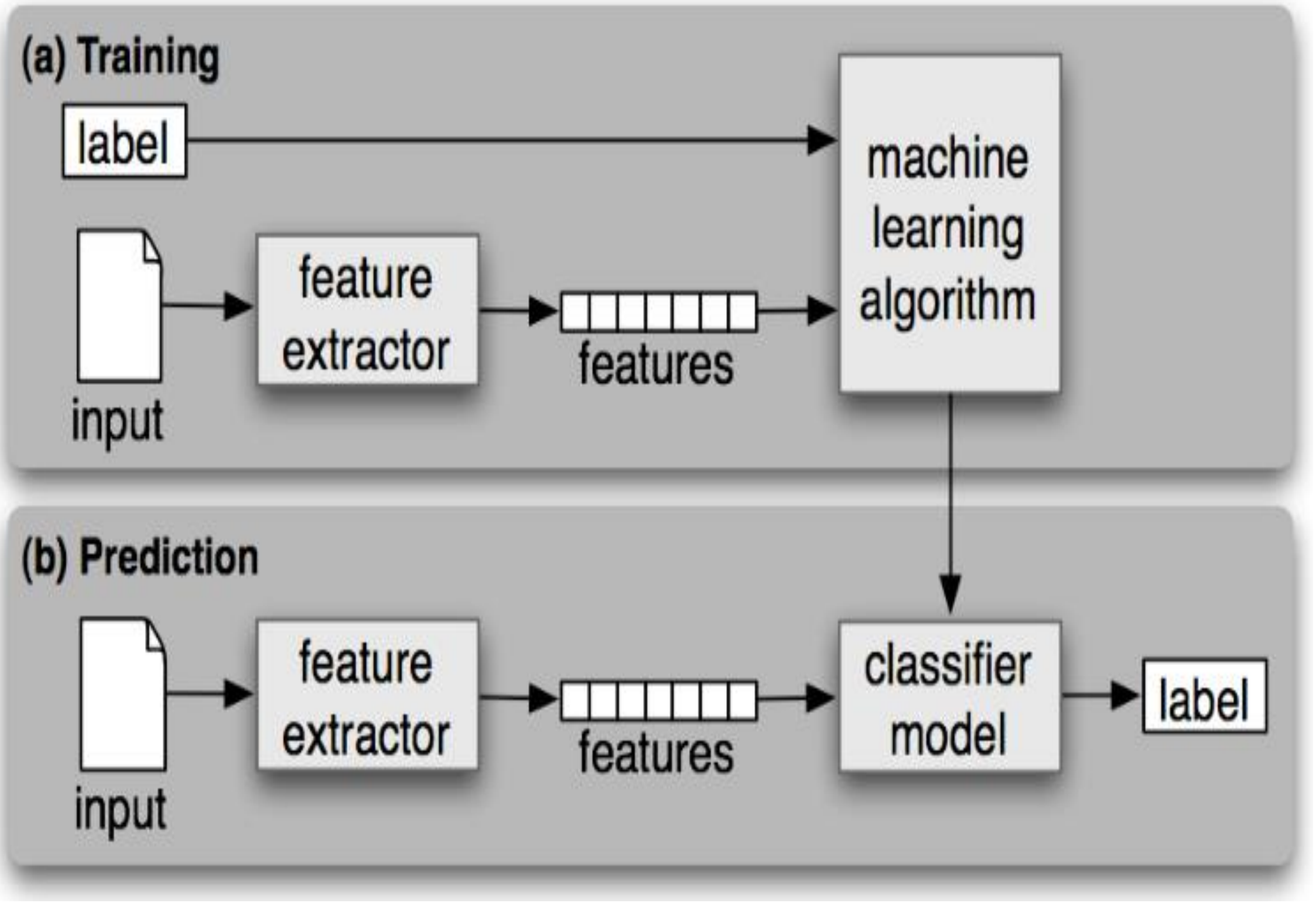
1) As derived simultaneously for all dimensions, they do not only focus on single dimension as most existing measures do.

2) As the maximum margin boundary of SVM has been proven to be optimal in a structural risk minimization sense, they may better indicate the discriminative power of features.

3) As efficient routines for SVM training are available that can readily deal with huge number of features and samples, they do not sacrifice in computational cost. Our experimental results show significant improvements in pixel-level classification accuracy by using the proposed method.

2. Workflow

The basic workflow of the project is given below. During training the **SVM** (Support Vector Machines) model, an image is taken as an input which is preprocessed to give features using feature extraction. The features are stored in an array to form a feature vector. This feature vector is passed into SVM model to train. During testing, an image is again taken as an input from the user, through which features are extracted to give us the specified image's feature vector. This feature vector is passed into the classifier model to predict whether a person has cervical cancer or not.



3. Dataset

The dataset included a selected number of images out of many images given by Dr JL. Raheja. The images were taken from all angles. In some images, the cervix was at the corners of the images. The dataset included images, only where the cervix was at the center. Some images were so zoomed out that it was not possible to see the cervix in the images. Those images also had to be omitted. Moreover, many images were blurred and thus could not have been used in the SVM model. Both the training and the testing model consisted a total of 45 images which composed of 25 cancerous and 20 non-cancerous images.

4. Image Pre-processing

Image pre-processing refers to the basic operations with images at the lowest level of abstraction- both the input and the output are intensity images. The aim of pre-processing is an improvement of the image data that suppresses the unwanted distortions and enhances some image features important for further processing.

In this project, the main image pre-processing techniques that were used are described below:

4.1. Cropping the image

All the images in the dataset had a common property that the primarily important region of interest(ROI) was centered near the centre of the image. Therefore, it made sense to crop out the central area which would be used for further operations. The following piece of code was used to do this:

```
Mat croppedImage = im(Rect(140,180,430,300));
```

This resulted in a cropped image with dimensions as specified using the parameters.

4.2. Converting image into grayscale

In OpenCV 2.4.6, inside the imgproc_ module, there is a built-in function called cvtColor that was made use of for this purpose. It takes as arguments the Mat object img(which is to be converted to grayscale), Mat object gray (the empty image matrix), and the conversion code, cv::COLOR_BGR2GRAY.

4.3. Adjusting the contrast and brightness of the image

The images provided in the dataset varied significantly in terms of contrast and brightness, therefore it was important that we applied some kind of normalization to the images.

Two commonly used point processes are multiplication and addition with a constant:

g(x) = alpha*f(x) + beta

The parameters alpha > 0 and beta are often called the gain and bias parameters. These parameters are said to control contrast and brightness respectively.

We can think of f(x) as the source image pixels and g(x) as the output image pixels. Then, more conveniently we can write the expression as:

g(i,j) = alpha*f(i,j) + beta

where i and j indicates that the pixel is located in the i-th row and j-th column.

After trying with various values of alpha and beta the optimal values that gave best results were obtained as : alpha=1.5, beta=0

4.4. Normalizing the image intensities for all images

Using the image intensity of a reference image in the dataset as reference, we normalised the intensities of all the images.

4.5. Applying Gabor filter

In image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for texture analysis, which means that it basically analyses whether there are any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. They have been found to be particularly appropriate for texture representation and discrimination. In the spatial domain, a 2D Gabor filter is a Gaussian kernel function modulated by a sinusoidal plane wave. Following is the code used for this operation:

```
cv::Mat kernel = cv::getGaborKernel(cv::Size(kernel_size,kernel_size), sig, th, lm, gm,
ps);
```

cv::filter2D(new_image, dest, CV_32F, kernel); //new_image is converted to dest after applying the kernel.

Following values of the parameters were found to be optimal:

int kernel_size = 31;

double sig = 1, th = 0, lm = 1.0, gm = 0.02, ps = 0;

4.6. Binary thresholding

The final pre-processing step involved the conversion of the grayscale image into binary thresholded image. This is one of the most basic image segmentation techniques. In this technique, to differentiate the pixels we are interested in from the rest (which will eventually be rejected), we perform a comparison of each pixel intensity value with respect to a threshold. If the intensity of given pixel is greater than the threshold, it is changed to the value 0(black), if it's lesser, it is changed to the value 255(white).

5. Feature Extraction:

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called *feature selection*. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data. Following features were selected for extraction in the project:

5.1. Entropy

Image entropy is a quantity which is used to describe the `business' of an image, i.e. the amount of information which must be coded for by a compression algorithm. Low entropy images, such as those containing a lot of black sky, have very little contrast and large runs of pixels with the same or similar DN values. An image that is perfectly flat will have an entropy of zero. Consequently, they can be compressed to a relatively small size. On the other hand, high entropy images such as an image of heavily cratered areas on the moon have a great deal of contrast from one pixel to the next and consequently cannot be compressed as much as low entropy images. Basically, entropy is a statistical measure of randomness that can be used to characterize the texture of the input image.

$$Entropy = \sum_{i,j=0}^{N-1} -\ln(P_{ij})P_{ij}$$

The entropy of the image is calculated before thresholding. The entropy of an image has been determined approximately from the histogram of the image. The histogram shows the different grey level probabilities in the image. The histogram of the image can be calculated in the way given below:-

```
calcHist(&new_image, 1, 0, Mat(), hist, 1, &histSize, &histRange, uniform, accumulate );
```

Finally, the entropy is calculated using this formula.

```
float entropy = -1*sum(hist.mul(logP)).val[0];
```

The different grey level probabilities of the histogram are multiplied with their natural algorithms. The sum of these products gives us entropy.

5.2. Homogeneity

Homogeneity relates to the validity of the often convenient assumption that the statistical properties of any one part of an overall dataset are the same as any other part. Homogeneity is used to look for further properties that might need to be treated as varying within a dataset once some initial types of non-homogeneity have been dealt with. Homogeneity returns a value that measures the closeness of the distribution of elements in the GLCM to the GLCM diagonal.

$$Homogeneity = \sum_{i,j=0}^{N-1} \frac{P_{ij}}{1+(i-j)^2}$$

We find and compare the homogeneity of the images before they are thresholded. We compute the homogeneity of the image by successively blurring the images using the following function:-

```
GaussianBlur(new_image,blur_non,Size(11, 11), 2);
```

Then we calculate the mean square difference between the pixels of the tenth and eleventh blurred images. Finally, we compute homogeneity using the following formula:-

```
homogeneity = MSD * MSD / last_blur_non.total();
```

5.3. Number of black pixels

The number of black pixels may signify the **area** of all the black patches inside the thresholded image. The black patches in the image converge to the cervix and the cancer in the image. Thus, an image which does not show cancer would show less area of the black patches as it has only the cervix in it. Whereas an image showing cancer would have higher area as it has both the cervix and the cancer in the image.

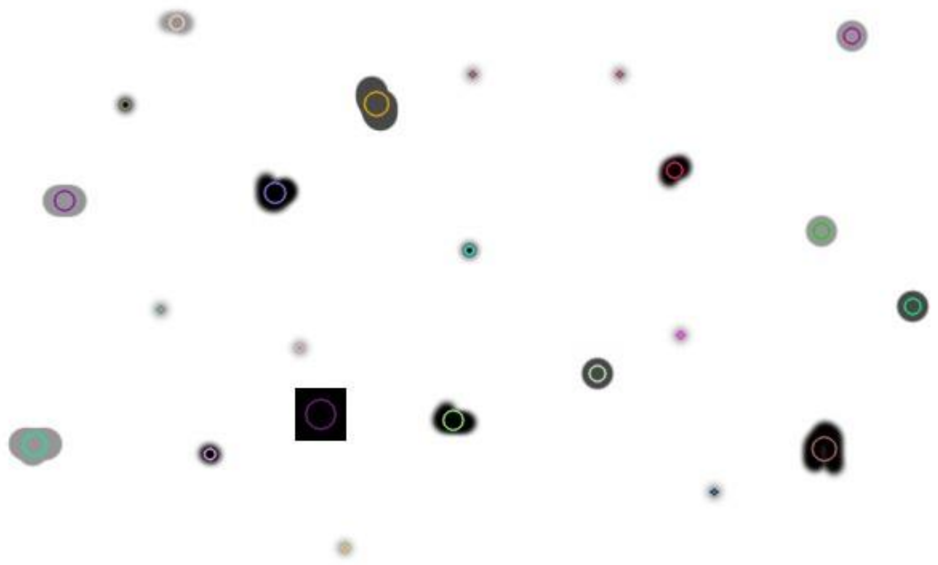
As we know, the intensity of a black pixel is 0 whereas the intensity of the white pixel is 255. Thus to find the area, we access all the pixels in the image and check whether their in intensity is 0.

```
if ( dst.at<uchar>(y,x) == 0 ) {
```

If the following condition is true we increase the variable specified by area by 1.

5.4. Number of Blobs

A Blob is a group of connected pixels in an image that share some common property (E.g grayscale value). In the image above, the dark connected regions are blobs, and the goal of blob detection is to identify and mark these regions.



The algorithm is controlled by parameters (shown in bold below) and has the following steps.

Thresholding: Convert the source images to several binary images by thresholding the source image with thresholds starting at **minThreshold**. These thresholds are incremented by **thresholdStep** until **maxThreshold**. So, the first threshold is minThreshold, the second is minThreshold + thresholdStep, the third is minThreshold + 2 x thresholdStep, and so on.

Grouping: In each binary image, connected white pixels are grouped together. Let's call these binary blobs.

Merging: The centers of the binary blobs in the binary images are computed, and blobs located closer than **minDistBetweenBlobs** are merged.

6. Feature Vectors

A set of numeric features can be conveniently described by a feature vector. An example of reaching a two-way classification from a feature vector (related to the perceptron) consists of calculating the scalar product between the feature vector and a vector of weights, comparing the result with a threshold, and deciding the class based on the comparison.

In pattern recognition and machine learning, a **feature vector** is an n-dimensional vector of numerical features that represent some object. Many algorithms in machine learning require a numerical representation of objects, since such representations facilitate processing and statistical analysis. When representing images, the feature values might correspond to the pixels of an image, while when representing texts the features might be the frequencies of occurrence of textual terms. Feature vectors are equivalent to the vectors of explanatory variables used in statistical procedures such as linear regression. Feature vectors are often combined with weights using a dot product in order to construct a linear predictor function that is used to determine a score for making a prediction.

The vector space associated with these vectors is often called the **feature space**. In order to reduce the dimensionality of the feature space, a number of dimensionality reduction techniques can be employed.

A vector is a series of numbers, like a matrix with one column but multiple rows, that can often be represented spatially. A **feature** is a numerical or symbolic property of an aspect of an object. A **feature vector** is a vector containing multiple elements about an object. Putting feature vectors for objects together can make up a **feature space**.

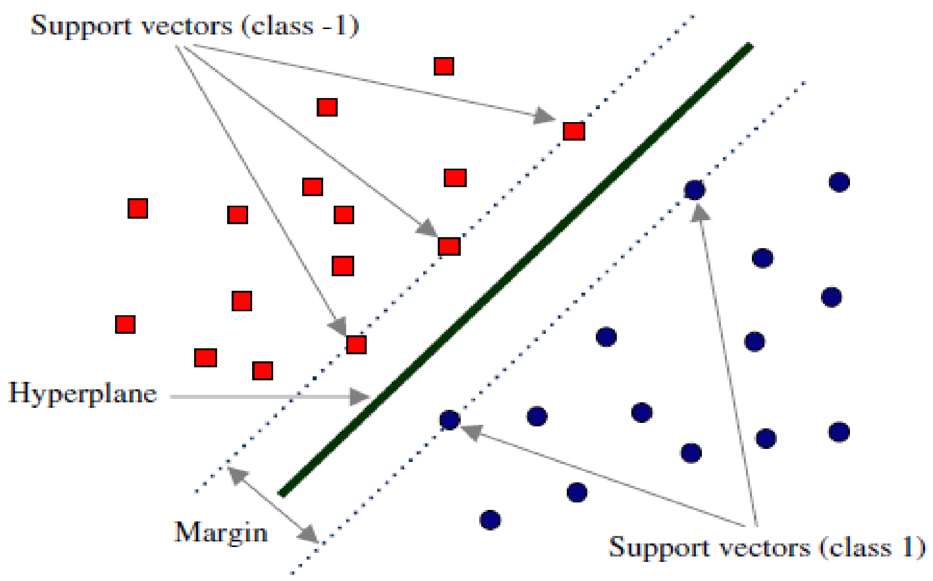
In our project, the feature vector is a 2-D array or a matrix with dimension n and 4. n is the number of images we give as input and 4 is the number of features, namely Entropy, Homogeneity, Number of black pixels and Number of blobs. We test and train our model based on this matrix, that is, feature vector.

7. SVM

In machine learning, support vector machines (SVMs) are considered to be the best "off the shelf" supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories (or classes), an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

The original SVM algorithm was invented by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963 and it worked only for linearly separable data. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

A detailed discussion on how an SVM calculates the optimal hyperplane (for linearly-separable data) is provided in the appendices section of the report.



Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

In this project we tried both these variants, starting with the SVM for linearly separable data. For the given dataset the application of this variant of SVM did not yield satisfactory results. The accuracy on a randomly selected (test+train) dataset of 60 images (30 in test and train each) yielded an accuracy of 50%.

Then we moved on to SVM for non-linearly separable data(see the appendices section for details), which gave much better results.

7.1. Coding the SVM:

Coding the SVM involved 3 major steps:

7.1.1. Setting up SVM's parameters

In the most simple case, when the training examples are spread into two classes that are linearly separable, there was not much to do with. However, while using SVMs with non-linearly separable data, there was a lot of adjustments to be made to the parameters (a SVM using a kernel function to raise the dimensionality of the examples, etc). As a consequence of this, we had to define some parameters before training the SVM. These parameters are stored in an object of the class CvSVMParams .

Following is the set of parameters along with their optimal values found during the project:

```
CvSVMParams params;

params.svm_type    = CvSVM::C_SVC;

params.kernel_type = CvSVM::RBF;

param.C           = 1000000;

param.gamma       = 0.000001;

param.term_crit    = TermCriteria(CV_TERMCRIT_ITER, (int)1e7, 1e-6);
```

Type of SVM. We choose here the type CvSVM::C_SVC that can be used for n-class classification (here n=2). This parameter is defined in the attribute CvSVMParams.svm_type. The important feature of the type of SVM CvSVM::C_SVC deals with imperfect separation of classes (i.e. when the training data is non-linearly separable. We chose this SVM type only for being the most commonly used.

Type of SVM kernel. The kernel functions are not interesting for the case of linearly separable training data. But in the case of non-linearly separable data, the choice of kernel can make a whole lot of difference in the results. Let's explain briefly now the main idea behind a kernel function. It is a mapping done to the training data to improve its resemblance to a linearly separable set of data. This mapping consists of increasing the dimensionality of the data and is done efficiently using a kernel function. We experimented with 3 types of kernels during the project:

i) LINEAR

ii) RBF

iii) POLY

We chose here the type CvSVM::RBF which gave the best results amongst the three. This parameter is defined in the attribute CvSVMParams.kernel_type.

Termination criteria of the algorithm. The SVM training procedure is implemented solving a constrained quadratic optimization problem in an iterative fashion(see the Appendix A for details). Here we specify a maximum number of iterations and a tolerance error so we allow the algorithm to finish in less number of steps even if the optimal hyperplane has not been computed yet. This parameter is defined in a structure cvTermCriteria.

7.1.2. Training the SVM

We call the method CvSVM::train to build the SVM model as follows:

```
CvSVM SVM;

SVM.train(trainingDataMat, labelsMat, Mat(), Mat(), params);
```

7.1.3. Testing the SVM

The method CvSVM::predict is used to classify an input sample using a trained SVM.

Following is the piece of code which was written for this purpose:

```
cv::Mat predicted(1,1, CV_32F);

for(int i = 0; i <testDataMat.rows; i++) {

   cv::Mat sample = testDataMat.row(i);

   float p = SVM.predict(sample);

///

}
```

8. ACCURACY AND RESULT

The final accuracy obtained on the test set was 95.56%. Out of the 45 images that were used in the test set, our model correctly predicted the labels of 43 images.

9. Conclusions and Recommendations

Final remarks:

The final model that we arrived at was a result of a series of continuous refinements, some altering it drastically while some involved fine-tuning the existing parameters. In our opinion, an accuracy of >95% makes this model fit for use in real-life situations.

Recommendations:

We worked on Visual studio 2012 configured with OpenCV 2.4.6 which is an outdated tool for this purpose. If the state of the art deep learning models can be made available, the results may improve. Also, an enlarged and refined dataset can yield better results.

10. Appendices

10.1. How is the optimal hyperplane computed?

Let’s introduce the notation used to define formally a hyperplane:

f(x) = β0 + βTx,

where β is known as the weight vector and β0 as the bias.

The optimalhyperplanecanberepresentedinaninfinitenumberofdifferentwaysbyscalingof β and β0. As a matter of convention, among all the possible representations of the hyperplane, the one chosen is

|β0 + βTx| = 1

where x symbolizes the training examples closest to the hyperplane. In general, the training examples that are closest to the hyperplane are called support vectors. This representation is known as the canonical hyperplane. Now, we use the result of geometry that gives the distance between a point x and a hyperplane (β,β0):

$$\mathrm{distance} = \frac{|\beta_0 + \beta^T x|}{||\beta||}.$$

.

In particular, for the canonical hyperplane, the numerator is equal to one and the distance to the support vectors is

$$\mathrm{distance}_{\text{support vectors}} = \frac{|\beta_0 + \beta^T x|}{||\beta||} = \frac{1}{||\beta||}.$$

Recall that the margin introduced in the previous section, here denoted as M, is twice the distance to the closest examples:

$$M = \frac{2}{||\beta||}$$

Finally, the problem of maximizing M is equivalent to the problem of minimizing a function L(β) subject to some constraints. The constraints model the requirement for the hyperplane to classify correctly all the training examples xi. Formally,

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \, \forall i,$$

where yi represents each of the labels of the training examples. This is a problem of Lagrangian optimization that can be solved using Lagrange multipliers to obtain the weight vector β and the bias β0 of the optimal hyperplane.

10.2. Support Vector Machines for Non-Linearly Separable Data

Extension of the Optimization Problem

We know that using SVMs we obtain a separating hyperplane. Therefore, since the training data is now non-linearly separable, we must admit that the hyperplane found will misclassify some of the samples. This misclassification is a new variable in the optimization that must be taken into account. The new model has to include both the old requirement of finding the hyperplane that gives the biggest margin and the new one of generalizing the training data correctly by not allowing too many classification errors. We start here from the formulation of the optimization problem of finding the hyperplane which maximizes the margin (this is explained in the previous section):

$$\min_{\beta,\beta_0} L(\beta) = \frac{1}{2}||\beta||^2 \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 \, \forall i$$

There are multiple ways in which this model can be modified so it takes into account the misclassification errors. For example, one could think of minimizing the same quantity plus a constant times the number of misclassification errors in the training data, i.e.:

$$\min ||\beta||^2 + C(\#\ \text{misclassication errors})$$

However, this one is not a very good solution since, among some other reasons, we do not distinguish between samples that are misclassified with a small distance to their appropriate decision region or samples that are not. Therefore, a better solution will take into account the distance of the misclassified samples to their correct decision regions, i.e.:

$$\min ||\beta||^2 + C(\text{distance of misclassified samples to their correct regions})$$

For each sample of the training data a new parameter ξi is defined. Each one of these parameters contains the distance from its corresponding training sample to their correct decision region. The following picture shows non-linearly separable training data from two classes, a separating hyperplane and the distances to their correct regions of the samples that are misclassified.

$$\min_{\beta, \beta_0} L(\beta) = ||\beta||^2 + C \sum_i \xi_i \text{ subject to } y_i(\beta^T x_i + \beta_0) \geq 1 - \xi_i \text{ and } \xi_i \geq 0\ \forall i$$

How should the parameter C be chosen? It is obvious that the answer to this question depends on how the training data is distributed. Although there is no general answer, it is useful to take into account these rules:

• Large values of C give solutions with less misclassification errors but a smaller margin. Consider that in this case it is expensive to make misclassification errors. Since the aim of the optimization is to minimize the argument, few misclassifications errors are allowed.

• SmallvaluesofCgivesolutionswithbiggermarginandmoreclassificationerrors. In this case the minimization does not consider that much the term of the sum so it focuses more on finding a hyperplane with big margin.

11. References

1. The OpenCV Tutorials 2.4.13.6
2. Cervical Cancer Detection Using SVM Based Feature Screening by Jiayong Zhang and Yanxi Liu
3. Classification of Cervical Cancer using Artificial Neural Networks by M. Anousouya Devi∗, S. Ravi∗, J. Vaishnavi and S. Punitha
4. DeepPap: Deep Convolutional Networks for Cervical Cell Classification by Ling Zhang, Le Lu, Senior Member, IEEE, Isabella Nogues, Ronald M. Summers, Shaoxiong Liu, and Jianhua Yao

Glossary

Gabor Filter: a linear filter used for texture analysis, which means that it basically analyses whether there are any specific frequency contents in the image in specific directions in a localized region around the point or region of analysis.

Thresholding: the simplest thresholding methods replace each pixel in an image with a black pixel if the image intensity is less than some fixed constant T or a white pixel if the image intensity is greater than that constant. In the example image on the right, this results in the dark tree becoming completely black, and the white snow becoming completely white.

Entropy: lack of order or predictability

Homogeneity: the validity of the often convenient assumption that the statistical properties of any one part of an overall dataset are the same as any other part.

Feature vector: an n-dimensional vector of numerical features that represent some object

SVM: Support vector machine; a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible

Project 2 – Tooth Cavity Segmentation

Abstract: In the era of minimal invasive dentistry, every effort should be directed to preserve the maximum tooth structure during cavity preparation. However, while making cavities, clinicians usually get indecisive at what point caries excavation should be stopped, so as to involve only the infected dentin. Apparent lack of valid clinical markers, difficulties with the use of caries detector dyes and chemo mechanical caries removal systems carve out a need for an improved system, which would be helpful to differentiate between the healthy and infected dentin during caries excavation. Light induced fluorescence evaluation is a novel concept implicated for caries detection and for making decisions while cavity preparation. Autofluorescence masking effect was found to be helpful for caries detection and the red fluorescence in the treatment mode was found helpful in deciding 'when to stop the excavation process.' Light induced fluorescence evaluation - Diagnosis - Treatment concept can be used as a guide for caries detection and excavation. It also facilitates decision making for stopping the caries excavation so as to involve infected dentin only.

The **aim** of the project is to extract just the tooth region of the image and remove the unwanted blurred distortions in the image, also known as noise. Basically, the aim of the project is to refine the image so that it would be easier to determine whether the cavity is removed from the tooth or not.

Signatures of Students:	Signature of PS Faculty:
Date:	Date:

CONTENTS

1. **Introduction**

Once the caries has reached the cavitary stage, the only treatment option left is to restore the lesion, because of the irreversible nature of the disease. Till yester years, cavities were made using GV Black's principles that dictate extension for prevention. This classical approach to treat dentinal lesions mandates removing all the infected and affected dentin. However today, there is paradigm shift in the manner in which the cavitated dentinal lesions are handled. With changes in the materials and restorative principles, the concept of 'Minimally Invasive Dentistry (MID)' has taken the lead. And, it is now well accepted to extend the cavities only to involve infected dentin and leaving the affected dentin as such. However, while making cavities, it is usually difficult to know at what point caries excavation should be stopped, such as to involve only the infected dentin. This is due to the lack of valid clinical markers to differentiate between the infected and affected dentin.

Many subjective factors like consistency of the tissue and color are being used throughout to differentiate between the healthy and diseased dentinal tissue, and thus serve as a guide for the termination of the excavation process. Other methods used to facilitate this excavation process include: use of caries detector dyes; use of chemo mechanical measures; smart prep burs, etc. But there are concerns with their usage, like inadvertent cutting of sound dentin, increased time to excavate etc. Fluorescence has also been used to guide the excavation process. Based on the property of fluorescence of dental tissues, four different fluorescence visions can be observed: green fluorescence indicating healthy tissues; black green fluorescence indicating infected dentin; bright red colour indicting the margin of infected/affected dentin; and, acid green fluorescence indicating sound dentin at the end of excavation. These different fluorescence signals aid in caries detection and decision making during cavity preparation. This concept has been termed as Light Induced Fluorescence Evaluation - Diagnosis - Treatment concept (Life D.T concept).

Recently a new fluorescence based camera system that works on the principle of Life D.T has been launched to aid caries detection and to guide cavity preparation [SoproLife (Sopro, La Ciotat, France)]. The camera captures the images in three different modes that is, daylight, diagnosis and treatment mode. Capturing in the day light provides a white light image with a magnification of more than 50 times than the tooth surface. The other two modes of the camera work on the principle of autofluorescence. In the diagnostic mode, the camera uses a visible blue light frequency (wavelength 450 nm) to illuminate the surface of the teeth, and provides an anatomic image overlay of the green fluorescence image on the "white light" image. This green fluorescence is considered as an indicator of healthy dental tissues; while carious lesions could be detected by variation in the auto fluorescence of its tissues in relation to a healthy area of the same tooth.

In addition to the green fluorescence, red fluorescence may also be seen in some diagnostic mode images. This red fluorescence may represent deep dentinal caries; however, at the same time it might be a false signal coming from the organic deposits covering the tooth. Terrer E. *et al.* found a correlation between this red signal and organic deposits in the bottom of the groove. Hence, if a red fluorescent signal is encountered in the diagnostic mode images, it needs to be validated. For validation, the area showing the red fluorescence should be washed off with sodium bicarbonate or pumice, and if the fluorescence persists, then only it is considered to be representative of infected dentin. The fluorescence would no longer be there, if the source is simply the organic deposits on the tooth surface.

The third mode is the treatment mode, and the red fluorescence captured in this mode is considered as an indicator to differentiate between infected and affected dentin.

Important points to remember: -

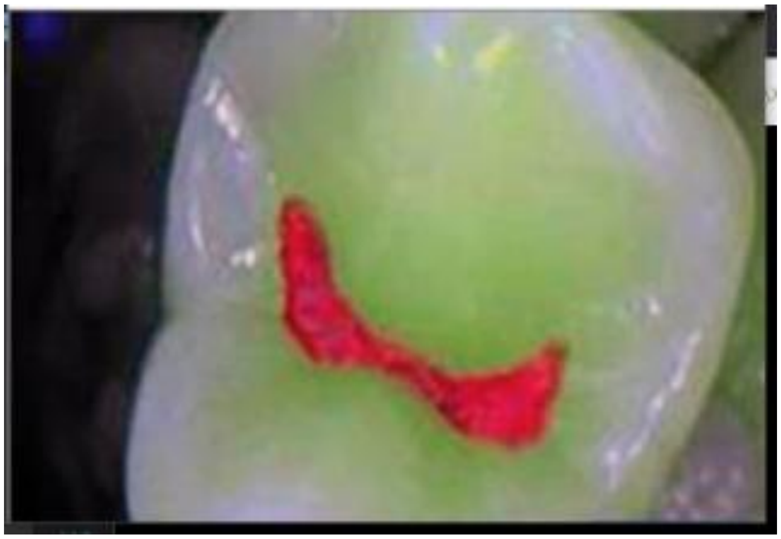
1.Green fluorescence in the diagnostic mode of the camera is considered an indicator of healthy tooth and loss of green fluorescence (black green fluorescence) indicates infected dentin.

2. Excavation is done till acid green fluorescence is achieved.

3. Bright red fluorescence indicates infected/ affected dentin.

2. **Image Processing**

We take the BGR image as an input from the user.



BGR pixels is a type of subpixel rendering. **Subpixel rendering** is a way to increase the apparent resolution of a computer's liquid crystal display (LCD) or organic light-emitting diode (OLED) display by rendering pixels to take into account the screen type's physical properties. It takes advantage of the fact that each pixel on a color LCD is actually composed of individual red, green, and blue or other color subpixels to anti-alias text with greater detail or to increase the resolution of all image types on layouts which are specifically designed to be compatible with subpixel rendering.

We created a white screen whose width and height of the image is equal to the width and height of the input image.

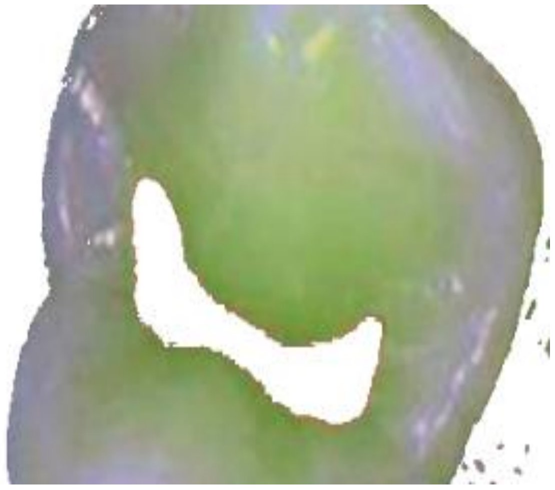
Thresholding:-The simplest segmentation method. It separates out regions of an image corresponding to objects which we want to analyze. This separation is based on the variation of intensity between the object pixels and the background pixels. To differentiate the pixels we are interested in from the rest (which will eventually be rejected), we perform a comparison of each pixel intensity value with respect to a *threshold* (determined according to the problem to solve). Once we have separated properly the important pixels, we can set them with a determined value to identify them (i.e. we can assign them a value of 0 (black), 255(white) or any value that suits your needs).

After taking the BGR image as an input from the user, we use basic thresholding operations using the OpenCV **cv::inRange** function. The function helps us to detect an object based on the range of pixel values.

OpenCV's inRange() function is very similar to threshold(). It also takes a grayscale image and converts it into a "binary" image: one where every pixel is either all-white or all-black based on whether or not the pixel is within the given range of of values. Where threshold() selects for pixels that are above a given value, inRange() lets you specify a maximum as well as a minimum value. Like most of the other OpenCV filter functions, inRange() can be used on any single-channel image, not just grayscale versions of color images. In particular, inRange() is especially useful when applied to the Hue channel.

inRange(int min, int max) takes two arguments: the first represents the lower bound of the range, the second represents the upper bound. It will result in a binary OpenCV output image with the same pixel value where the pixels were in-range and white where they were out-of-range.

We used the inRange() function to generate the mask on the input image that has a value of the same pixel value for pixels where the RGB values fall within the range (0-255, 120-255, 0-255) and a value of 255 for pixels whose values don't lie in this interval. Then we copied the mask to the white screen. This helped us extract the tooth portion of the image without the red region.

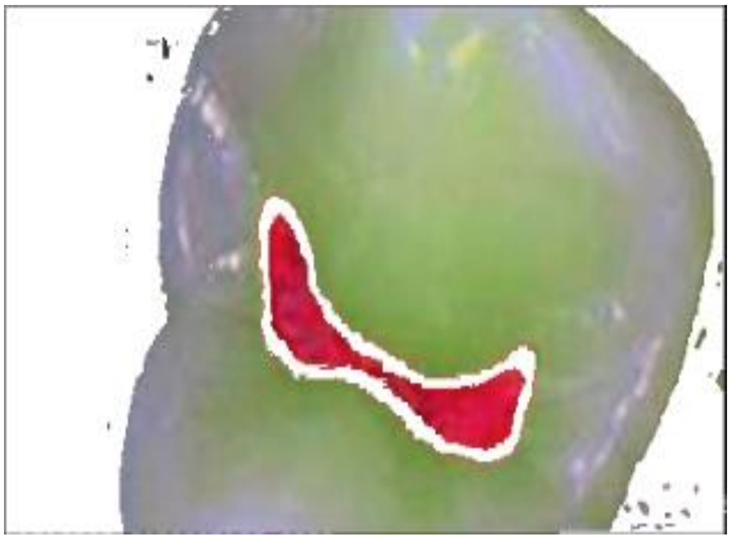


To extract the red region and add to the image we first converted the BGR image to an HSV image.

HSV (hue, saturation, value) colorspace is a model to represent the colorspace similar to the RGB color model. Since the hue channel models the color type, it is very useful in image processing tasks that need to segment objects based on its color. Variation of the saturation goes from unsaturated to represent shades of gray and fully saturated (no white component). Value channel describes the brightness or the intensity of the color. Next image shows the HSV cylinder. Since colors in the RGB colorspace are coded using the three channels, it is more difficult to segment an object in the image based on its color.

The hue ranges from 150-180 and 0-20 for red colour in HSV images. The value ranges from 0-50 for black colour in HSV images.

After converting the image to HSV, we used the inRange() function to generate the mask on the input image that has a value of the same pixel value for pixels where the HSV values fall within the range (150-180, 50-255, 50-255) and a value of 255 for pixels whose values don't lie in this interval. Then we copied the mask to the previous image. This helped us to add the lighter red portion of the input image to the image of the tooth.

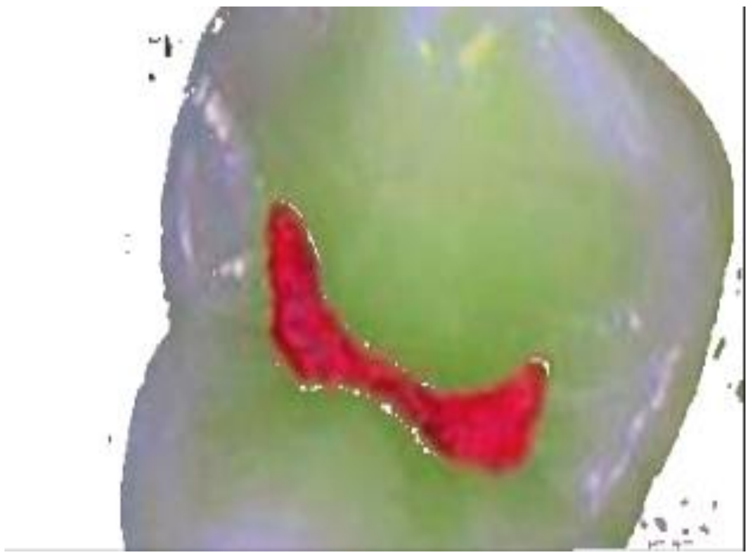


We again used the inRange() function to generate the mask on the input image that has a value of the same pixel value for pixels where the HSV values fall within the range (0-20, 50-255, 50-255) and a value of 255 for pixels whose values don't lie in this interval.

Then we copied the mask to the previous image. This helped us to add the darker red portion of the input image to the image of the tooth.

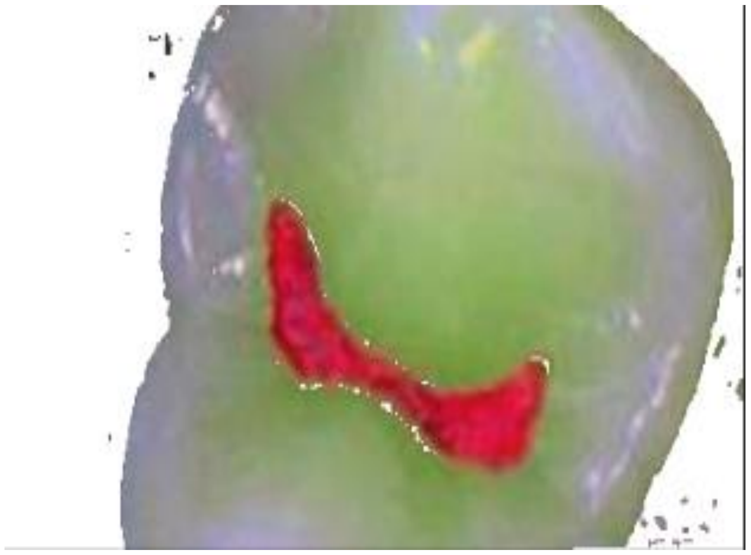
The reason that we varied the saturation from 50-255 and not 0-255 was to avoid the black colour objects getting added to our resultant image.

This is our final image.



3. **Result**

The final image is shown below. Only the tooth is shown in the image and the surrounding noise has been removed from the image.



4. **Conclusion**

Using the final image, it is really easy to determine the presence of cavity in the tooth as we have removed the surrounding noise.

5. References

1. Light induced fluorescence evaluation: A novel concept for caries diagnosis and excavation by Neeraj Gugnani, IK Pandit, Nikhil Srivastava, Monika Gupta, Shalini Gugnani

2. The OpenCV Tutorials 2.4.13.6

PROJECT 3: <u>INTERFACING NEOPIXEL RING WITH ADAFRUIT FLORA V3</u>

Abstract: The NeoPixel ring, developed by Adafruit industries comes as a single ring with 16 individually addressable RGB LEDs assembled in a circle with 1.75" (44.5mm) outer diameter. The rings are 'chainable' - the output pin of one can be connected to the input pin of another. Using only one microcontroller pin we can control as many as you can LEDs as we can chain together. The microcontroller board used is Adafruit FLORA v3 board which is arduino compatible. A user friendly interface is developed using the windows form application feature of Visual Studio 2012.

Signature of the students: Signature of PS faculty:

Date: Date:

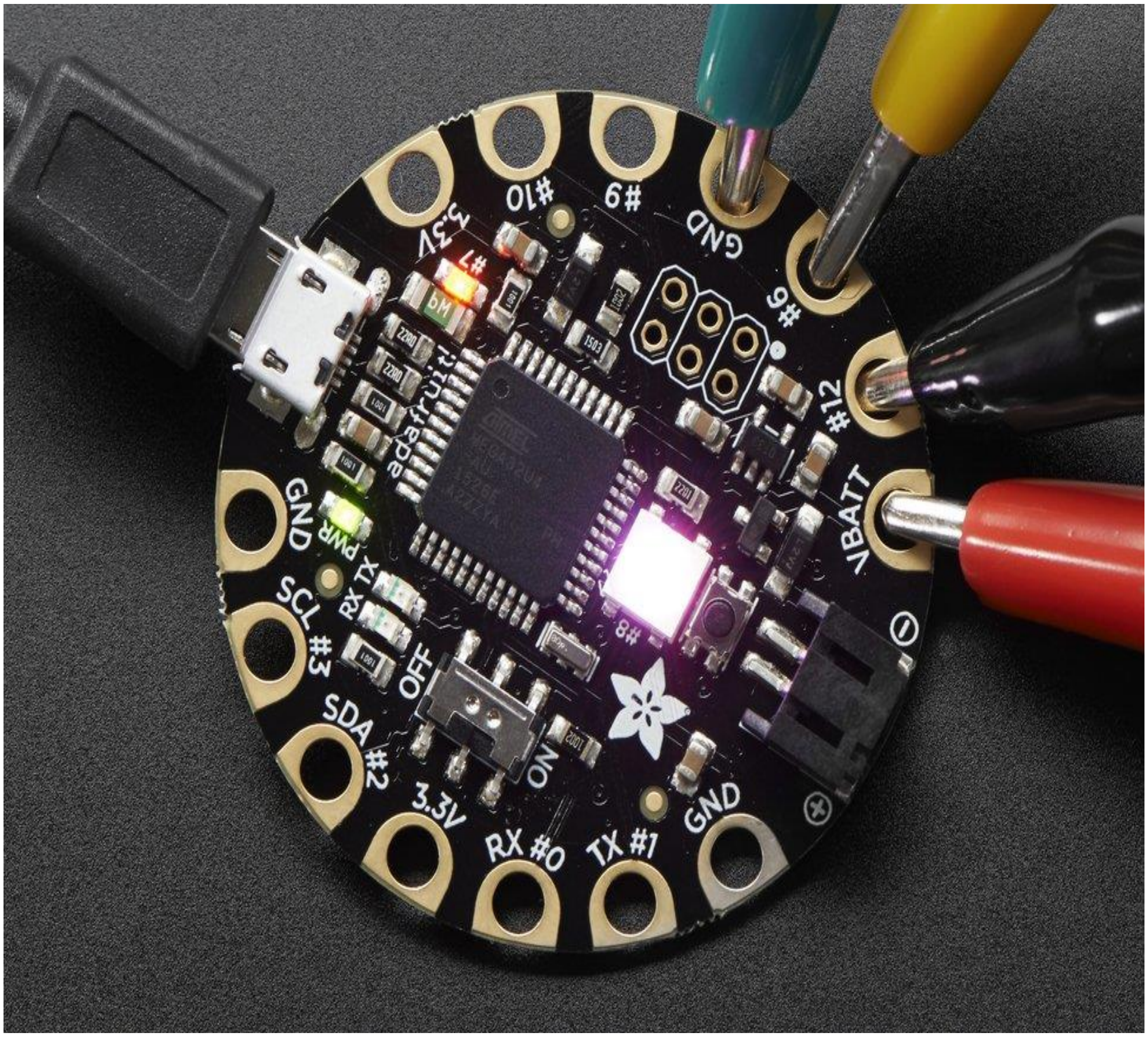
CONTENTS

1. INTRODUCTION:

1.1 NeoPixel Ring

The NeoPixel ring, developed by Adafruit industries comes as a single ring with 16 individually addressable RGB LEDs assembled in a circle with 1.75" (44.5mm) outer diameter. The rings are 'chainable' - the output pin of one can be connected to the input pin of another. Using only one microcontroller pin we can control as many as you can LEDs as we can chain together. Each LED is addressable as the driver chip is inside the LED. Each one has ~18mA constant current drive so the color will be very consistent even if the voltage varies, and no external choke resistors are required making the design slim. The whole ring can be powered with 5VDC (4-7V works). There is a single data line with a very timing-specific protocol. Since the protocol is very sensitive to timing, it requires a real-time microconroller such as an AVR, Arduino, PIC, mbed, etc. It cannot be used with a Linux-based microcomputer or interpreted microcontroller such as the netduino or Basic Stamp. Various Arduino based platforms can be used to control the ring as it comes with the NeoPixel library which is arduino compatible.

1.2 Adafruit FLORA v3 board



FLORA is Adafruit's fully-featured wearable electronics platform. It's a round, sewable, Arduino-compatible microcontroller designed to empower various wearables projects. The FLORA is small (1.75" diameter, weighing 4.4 grams). The FLORA has 4 indicator LEDs: power good, digital signal LED for bootloader feedback, data rx/tx. Also onboard is an ICSP connector for easy re-programming for advanced users. Flora v3 has an RGB NeoPixel for even more colorful lighting. The FLORA works great with the Arduino IDE and is easy to install support with IDE 1.6.4 or later.

2. PROGRAMMING THE FLORA BOARD

We worked on the latest version of arduino IDE, 1.8.5. It doesn't come with pre-installed NeoPixel library, but it is easy to install it from Adafruit's NeoPixel github repository. After installing the library, it can be integrated in the code simply by including the header:

```
#include <Adafruit_NeoPixel.h>
```

The official documentation is a good source to learn about the various functionalities supported. Also, it comes with a lot of exemplar projects which provide great insight as to how the project can be structured.

Initializing and Parameters

```
#define PIN 6  //Arduino pin number

 int brightness;      //Controls pixel intensity values

// Parameter 1 = number of pixels in strip

// Parameter 2 = Arduino pin number

// Parameter 3 = pixel type flags, add together as needed:

//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)

//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)

//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)

//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)

Adafruit_NeoPixel strip = Adafruit_NeoPixel (16, PIN,  NEO_RGBW + NEO_KHZ800);
```

<u>Set the Pixel Color</u>

The following code snippet sets the color of ith pixel to c, where c is the GRB value representation of the desired color. For example for green color, c= (255, 0, 0), for red color, c= (0, 255, 0) and so on.

```
strip.setPixelColor(i,c);
```

<u>Set the pixel brightness</u>

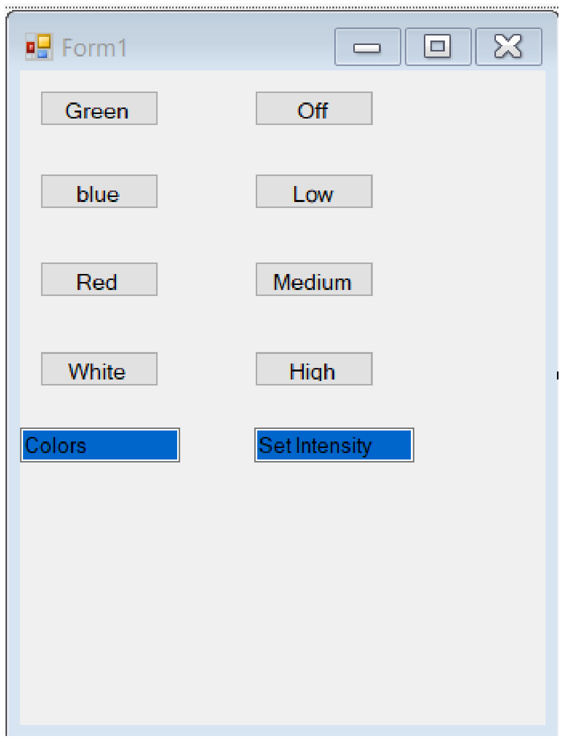
The NeoPixel library comes with an inbuilt function setBrightness, which expects as an argument a brightness value, ranging from 0 (off) to 255 (maximum intensity). Following code shows its usage:

```
int brightness;

strip.setBrightness(brightness);
```

3. Windows form application as the UI

Instead of having to give inputs from the console, we made the interface between the user and the executor (FLORA) more interactive by making a windows forms application which contains a set of colors and set of brightness values which can both be controlled independently of each other.



4. RESULT AND CONCLUSION

The NeoPixel ring has been successfully interfaced with the FLORA board and the windows form application can be used to control the color of the ring as well as it’s brightness (independently).

5. REFERENCES:

1.Adafruit NeoPixel Überguide

2.Arduino Programming Notebook- Arduino Playground

7. GLOSSARY:

Arduino – An open source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

FLORA – Adafruit's fully featured wearables electronic platform with Arduino compatible microcontroller on board.