****

**PES UNIVERSITY**

**Department of Science and Humanities**

**IMAGE PROCESSING**

**Submitted by**

**USN Name**

**01FB14ECS189 Rishabh Soni**

**01FB14ECS194 S K Mayank**

**01FB14ECS207 Sandeep PVN**

**01FB14ECS196 S Varun**

**For the academic year 2015-2016**

**Under the Guidance of**

**Prof. Ranjani Niranjan**

**Assistant Professor**

**Department of Computer Science and Engineering**

**PES UNIVERSITY**

**ACKNOWLEDGMENT**

The success and final outcome of this project required a lot guidance and assistance from many people and I am extremely fortunate to have got this all along the completion of my project work. Whatever I have done is only due to such guidance and assistance and I would not forget to thank them.

We express our deep sense of gratitude to our principle**, Dr. Sridhar**, for providing and excellent academic environment in the college.

I heartily thank **Mr. Nithin Pujari**, Professor and Head of Department of Computer Science and Engineering, for his coordination and encouragement during this project

I owe my profound gratitude to out project guide **Prof. Ranjani Niranjan** Assistant Professor, who took keen interest on our project work and guided us all along till the completion of our project work by providing all necessary information for developing a good system.

I am thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Department of Computer Science which helped us successfully completing our project work. Also, I would like to extend our sincere regard to all the non-teaching staff of Department of Computer Science for their timely support.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| SL.NO | TOPICS | Pg. No |
|  |  |  |
| 1 | Abstract | 4 |
|  |  |  |
| 2 | Concepts | 5 |
|  |  |  |
| 3 | Formula Used | 6 |
|  |  |  |
| 4 | About Octave | 7 |
|  |  |  |
| 5 | Description | 8 |
|  |  |  |
| 6 | Table of Variables | 15 |
|  |  |  |
| 7 | Code | 16 |
|  |  |  |
| 8 | Output | 21 |
|  |  |  |
| 9 | Bibliography | 22 |
|  |  |  |

**ABSTRACT**

**Image processing** is processing of images using mathematical operations by using

any form of signal processing for which the input is an image, a series of images, or a

video, such as a photograph or video frame.

Since our topic is named image processing by machine learning we have taken the numerous images of apple and banana. So how do you identify them?

We are given 10 images of apples and bananas. Each image consists of R, G

and B values which represents Red, Green and Blue respectively. We take the mean of

these values. So this mean represents one image being processed. The same has to be

done with the remaining images and find the mean of all these images. Each image

represents a 3X1 matrix and is a vector.

Take another image for testing in order to find the error with reference to the

derived reference image. The error E is the difference of the test image and the given

image.

The concept of finding the magnitude of this error E we use Linear Regression and

actually detect if the image is an apple or a banana.

**CONCEPTS USED**

* Scaling
* Errors
* Best fit line
* Transposes
* Linear Regression

**FORMULAS USED**

MEAN = ∑ of all variables (Sum)

Number of variables

A.A’ = | | A | | ^2

|| B – A|| = sqrroot (| | B | |^2 + | | A | |^2 – 2 \* B’.A)

Cosθ== B’. A

||B|| ||A||

**ABOUT OCTAVE**

**GNU Octave** is software featuring a high-level programming language, primarily intended for numerical computations. It provides a command-line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. It may also be used as a batch-oriented language. It is part of the GNU Project, it is free software under the terms of the GNU General Public License.

Octave is one of the major [free](https://en.wikipedia.org/wiki/Free_software) alternatives to MATLAB, others being Julia and Scilab. These however put less emphasis on (bidirectional) syntactic compatibility with MATLAB than Octave does.

In addition to use on desktops for personal scientific computing, Octave is used in academia and industry. For example, Octave was used on a massive parallel computer at Pittsburgh supercomputing center to find vulnerabilities related to guessing social security numbers.

**TECHNICAL**

Octave is written in C++ using the C++ standard library.

Octave uses an interpreter to execute the Octave scripting language.

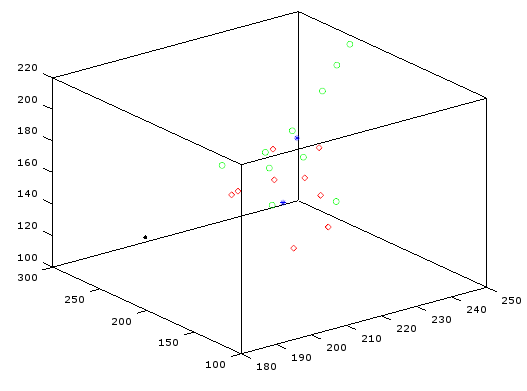
Octave is extensible using dynamically loadable modules.

Octave interpreter has an OpenGL based graphics engine to create plots, graphs and charts and to save or print them. Alternatively, gnu plot can be used for the same purpose.

**DESCRIPTION**

The first step is to take the images of the apple as input, convert them too their appropriate matrices. The code begins with three variables one each for Red, Green and Blue. These are initially zero. They represent the overall RGB values for the variables 'apple'.

The images are specifically named such as to make the input part easier and more flexible. The names of the file are stored in an array 'str' (10 images in our case). The function 'imread' does the above mentioned job for us



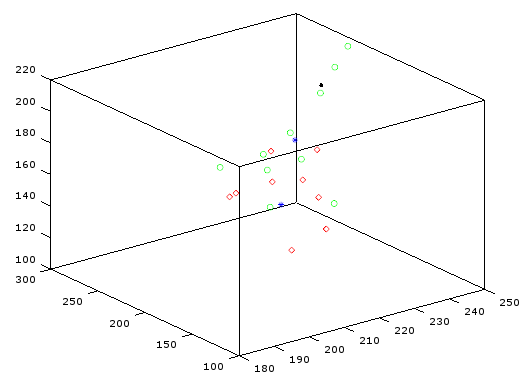
For every image, its respective RGB values are stored in variables with respective names and hence their means are found. This values are plotted then on the graph as seen above. Now these new mean values are added to the respective RGB class values for apple retaining the previous value.

(This step is repeated for all the images in the array ‘str’)

We get the final value of the RGB components of the apple class by dividing the each value by the no. of images the mean is taken over (10 in our case).

Now we have to repeat for the images of the banana as input, convert them too their appropriate matrices. The code begins with three classes one each for Red, Green and Blue. These are initially zero. They represent the overall RGB values for the class 'banana'.

The images are specifically named such as to make the input part easier and more flexible. The names of the file are stored in an array 'str' (10 images in our case). The function 'imread' does the above mentioned job for us.



For every image, its respective RGB values are stored in variables with respective names and hence their means are found. This values are plotted then on the graph as seen above .Now these new mean values are added to the respective RGB class values for banana retaining the previous value.

(This step is repeated for all the images in the array ‘str’)

We get the final value of the RGB components of the banana class by dividing the each value by the no. of images the mean is taken over (10 in our case).

Now getting to the calculations part we have,

Let A be original reference vector of Apple class.

Let A' denote its transpose.

A contains the mean RGB components of the apple class.

∴ A = [class\_apple\_red class\_apple\_green class\_apple\_blue];

The value of the vector A can be defined as, 'norm' of A.

i.e., ||A||^ 2 = A'A

Thus, ||A|| = √ (A'A)

Let B be original reference vector of Banana class.

Let B' denote its transpose.

A contains the mean RGB components of the Banana class.

∴ A = [class\_banana\_red class\_banana\_green class\_banana\_blue];

The value of the vector A can be defined as, 'norm' of B.

i.e., ||B||^ 2 = B'B

Thus, ||B|| = √ (B'B)

Now take in the test image T using 'imread' function. Save its RGB value matrix separately and find the mean for the same. Since we got only a single image this time, no extra calculations are needed to be done.

The test images:

**Image 1: Image 2:**





Now getting to the calculations part we have,

Let T be the Test image vector containing the mean RGB components in it.

Let T' denote its transpose.

||T|| = √ (T'T)

Let EA represent error vector b/w T and A.

Similarly, let EB represent error vector b/w T and B.

⇾ ⇾ ⇾

∴ EA = T - A

⇾ ⇾ ⇾

And, EB = T - B

Now, ||EA|| = ||T-A||

= √ (||T|| + ||A|| – 2.||T||.||A||.cosθ)

Where cosθ = Angle b/w T and A

i.e., cosθ = T’. A

||T|| ||A||

Thus on substitution,

||EA|| = √ (||T|| + ||A|| – 2.T'.A)

Similarly for EB,

||EB|| = √ (||T|| + ||B|| – 2.T'.B)

The above is the final equation that would be used for further calculations.

The only thing left is to compare the errors and validate the lesser error image.

Thus,

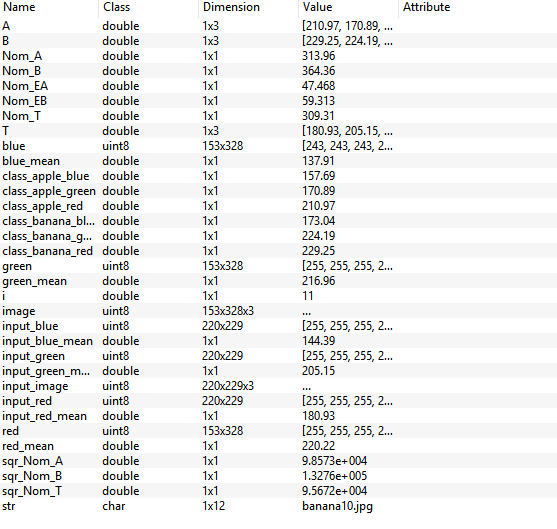
If (||EA|| < ||EB||)

“The image is an apple”

Else

“The image is a banana”

**TABLE OF VARIABLES**

****

**CODE**

**class\_apple\_red=0;**

**class\_apple\_green=0;**

**class\_apple\_blue=0;**

**for i=1:10**

**str= ["apple" int2str (i) ".jpg"]; #input of machine learning images**

**image=imread (str);**

**red=image(:,:, 1);**

**red\_mean=mean (mean (red));**

**green=image (:,:, 2);**

**green\_mean=mean (mean (green));**

**blue=image (:,:, 3);**

**blue\_mean=mean (mean (blue));**

**plot3 (red\_mean, green\_mean, blue\_mean, ‘rd');**

**hold on;**

**class\_apple\_red=class\_apple\_red+red\_mean;**

**class\_apple\_green=class\_apple\_green+green\_mean;**

**class\_apple\_blue=class\_apple\_blue+blue\_mean;**

**i=i+1;**

**end**

**class\_apple\_red=class\_apple\_red/10;**

**class\_apple\_green=class\_apple\_green/10;**

**class\_apple\_blue=class\_apple\_blue/10;**

**plot3 (class\_apple\_red, class\_apple\_green, class\_apple\_blue,'b\*');**

**hold on;**

**#Let A be original reference vector of Apple**

**A= [class\_apple\_red class\_apple\_green class\_apple\_blue];**

**# A transpose is A'**

**# ||A||^2 == A'A, ||A||^2 be sqr\_Nom\_A**

**sqr\_Nom\_A = A\*A';**

**Nom\_A=sqr\_Nom\_A\*\*0.5;**

**class\_banana\_red=0;**

**class\_banana\_green=0;**

**class\_banana\_blue=0;**

**for i=1:10**

**str= ["banana" int2str (i) ".jpg"];**

**Image=imread (str);**

**red=image (:,:, 1);**

**red\_mean=mean (mean (red));**

**green=image (:,:, 2);**

**green\_mean=mean (mean (green));**

**blue=image (:,:, 3);**

**blue\_mean=mean (mean (blue));**

**plot3 (red\_mean, green\_mean, blue\_mean, ‘go');**

**hold on;**

**class\_banana\_red=class\_banana\_red+red\_mean;**

**class\_banana\_green=class\_banana\_green+green\_mean;**

**class\_banana\_blue=class\_banana\_blue+blue\_mean;**

**i=i+1;**

**end**

**class\_banana\_red=class\_banana\_red/10;**

**class\_banana\_green=class\_banana\_green/10;**

**class\_banana\_blue=class\_banana\_blue/10;**

**plot3 (class\_banana\_red, class\_banana\_green, class\_banana\_blue,'b\*');**

**hold on;**

**#Let B be original reference vector of Banana**

**B= [class\_banana\_red class\_banana\_green class\_banana\_blue];**

**# B transpose is B'**

**# ||B||^2 == B'B, ||B||^2 be sqr\_Nom\_B , Nom\_B**

**sqr\_Nom\_B = B\*B';**

**Nom\_B=sqr\_Nom\_B\*\*0.5;**

**input\_image=imread ("testimage2.jpg");**

**input\_red=input\_image (:,:, 1);**

**input\_red\_mean=mean (mean (input\_red));**

**input\_green=input\_image (:,:, 2);**

**input\_green\_mean=mean (mean (input\_green));**

**input\_blue=input\_image (:,:, 3);**

**input\_blue\_mean=mean (mean (input\_blue));**

**plot3 (input\_red\_mean, input\_green\_mean, input\_blue\_mean,'k.');**

**hold on;**

**#Let T be test image vector**

**T= [input\_red\_mean input\_green\_mean input\_blue\_mean];**

**# T transpose is T'**

**# ||T||^2 == T'T, ||T||^2 be sqr\_Nom\_T**

**sqr\_Nom\_T = T\*T';**

**Nom\_T=sqr\_Nom\_T\*\*0.5;**

**#Let EA represent error vector between T and A vectors**

**# vectorEA = vectorT - vectorA**

**# Magnitude of EA is value of error ==> ||EA|| -> Nom\_EA**

**# Nom\_EA = ||T-A|| == sqrroot (sqr\_Nom\_T + sqr\_Nom\_A + 2\*Nom\_T\*Nom\_A\*cos@)**

**# cos@ is angle between T and A ==> (T'\*A)/ (NOM\_T\*NOM\_A)**

**# Hence, on substitution and division, Nom\_EA = sqrroot (sqr\_Nom\_T + sqr\_Nom\_A + 2\*T'\*A)**

**Nom\_EA= (sqr\_Nom\_T + sqr\_Nom\_A - 2\*A\*T') \*\*0.5;**

**#Let EB represent error vector between T and B vectors**

**# vectorEB = vectorT - vectorB**

**# Magnitude of EB is value of error ==> ||EB|| -> Nom\_EB**

**# Nom\_EB = ||T-B|| == sqrroot (sqr\_Nom\_T + sqr\_Nom\_B + 2\*Nom\_T\*Nom\_B\*cos@)**

**# cos@ is angle between T and B ==> (T'\*B)/ (NOM\_T\*NOM\_B)**

**# Hence, on substitution and division, Nom\_EB = sqrroot (sqr\_Nom\_T + sqr\_Nom\_B + 2\*T'\*B)**

**Nom\_EB= (sqr\_Nom\_T + sqr\_Nom\_B - 2\*T\*B') \*\*0.5;**

**#Compare the error values of both linearly regressed lines and detect if it’s an apple or not.**

**If (Nom\_EA<Nom\_EB)**

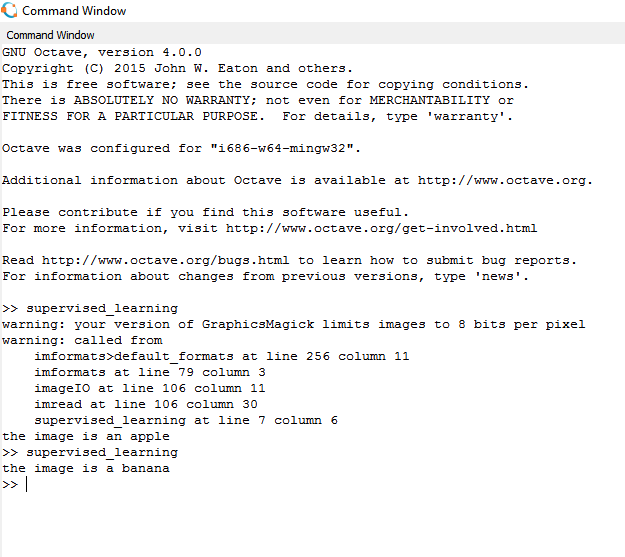
**printf ("the image is an apple \n");**

**else**

**printf ("the image is a banana \n");**

**end**

**OUTPUT**

****

**BIBLIOGRAPHY**