Real-Time Multiple Color Detection using Naïve Bayes

Dr.SAFIA NAVEED.S Associate Professor
Department of Computer Science and Engineering
KCG College of Technology, Chennai

Ibrahim Gadli
student of Department of Computer Science and Engineering
KCG College of Technology, Chennai

ABSTRACT - The main objective of this application is the methodology for identifying the shades of colors with an exact prediction with their names. A study says a normal human can identify nearly 1 million shades of colors. But in the case of humans having "Dichromacy", they would be able to see only 1% (i.e.10,000 colors) of the normal humans. While painting pictures, a painter needs to identify the color patterns exactly, or else the image's reality is unclear. Color detection is the process of detecting the name of the color. This is an easy task for humans to perform but the computer on the other hand cannot detect the color easily.Detecting colors can be a tricky task for computers, which is why we developed this project to make it easier and more accurate. We're always looking for ways to improve our technology and make it more efficient for our users. Numerous projects and research papers have been dedicated to addressing this issue. Python programming language utilizes the Naive Bayes algorithm, Pandas, and OpenCV libraries to tackle the problem. Naive Bayes is a straightforward method for creating classifiers that assign class labels to problem instances, represented as vectors of feature values, where the category labels are drawn from some finite set. However, there may be room for improvement in terms of optimizing the algorithm and libraries for better performance. Various algorithms can be utilized for training a naive Bayes classifier, but they all share the assumption that the value of a specific feature is unrelated to the value of any other feature, provided the class variable is known. The Open Source Computer Vision Library(OpenCV) was created with computational efficiency in mind and with a strong focus on real-time applications. It is an excellent tool for developers who need to implement video and image processing algorithms quickly and efficiently. Additionally, the Panda platform offers cloud-based video and audio encoding infrastructure, making it an ideal solution for developers who need to process large amounts of video data. Together, these tools provide a powerful set of capabilities for developers working on real-time video processing applications.

Keywords: Color Detection, OpenCV, Naive Bayes, Real-Time,

I. Introduction

Color detection is the process of detecting the name of any color. Simple isn't it? Well, for humans this is an extremely easy task but for computers, it is not straightforward. Human eyes and brains work together to translate light into color. Light receptors that are present in our eyes transmit the signal to the brain. Our brain then recognizes the color. Since childhood, we have mapped certain lights with their color names. We will be using a similar strategy to detect color names.

Before going into the speculations of the project it is important to know the definition of color detection. It is simply the process of identifying the name of any color. It is obvious that humans perform this action naturally and do not put any effort into doing so. While it is not the case for computers.

Human eyes and brain work in coordination to translate light into color. Light receptors that are present in the eyes transmit the signal to the brain which in turn recognizes the color. There is no exaggeration in saying that humans have mapped certain lights with their color names since childhood. This same strategy is useful in detecting color names in this project.

The fundamentals of computer vision are being utilized to track three distinct colors: Red, Green, and Blue. Upon successful compilation and execution of the code, an image is displayed in a new window, with its path provided as an argument.

We acquire both the name of the color and the combination of its red, blue, and green values for recognizing colors and in the field of robotics. One of the applications of color detection by computer vision is in driverless cars. This system is useful in detecting traffic and vehicle backlights and decides to stop, start and continue driving. This can also be applied in industry to pick and place different colored objects by the robotic arm. Color detection is also used as a tool in various image editing and drawing apps.

II. Literature Survey

A. Current Knowledge of OpenCV and Naïve Bayes

OpenCV may be a library of programming functions mainly aimed toward real-time computer vision. In simple language, it's a library used for Image Processing. It's mainly used to do all the operations associated with Images. You download the library on your computer and then begin writing your code which will make use of the various features in OpenCV. You will build your code and run it to perform the task you described. OpenCV is a Computer Vision library with APIs that permit you to find out a pipeline for your Computer Vision project. OpenCV-Python is a library of Python bindings designed to unravel computer vision problems. OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays.

Naive Bayes is based on the idea that features of measurement are independent of each other. This is often naive because it's (almost) never true. Naive Bayes uses an identical method to predict the probability of various classes which are supported by various attributes. This algorithm is usually utilized in text classification and problems that have multiple classes. Naive Bayes is an eager learning classifier and it sure is fast. Thus, it might be used for creating predictions in real-time multi-class Prediction: This algorithm is documented for predicting probabilities of multiple classes in the target variable.

B. How we use OpenCV and Pandas for Color detection:

Here we describe how we use OpenCV and Pandas libraries. OpenCV, Pandas, and Numpy are very much necessary to be installed in Python for this project. We can directly give an image path from the command prompt. The panda's library is very useful when we need to perform various operations on data files like CSV.pd.read_csv () which reads the CSV file into the pandas DataFrame.

We have assigned each column a name for easy access. OpenCV Library is useful for detecting the image, color values, and names. It can compute the RGB value of the pixels. The function parameters have the event name and (X, Y) coordinates of the position. This is called the draw function. Which was created by OpenCV and Pandas.

C. Optimization Techniques

Safia et al.[1] suggested a few optimization techniques such as Particle Swarm Optimization (PSO), Discrete Particle Swarm Optimization (DPSO), and Fractional Order Discrete Particle Swarm Optimization(FODPSO) Techniques based on which best optimum features from the face of the chauffeur can be selected to denote his drowsiness.

Safia et al. [2] also suggested that the region of interest can be captured and the same region can be inspected thoroughly in terms of pixel values, evaluating the degree of the noise present, analyzing the position of boundaries to study the region of interest, analyze the fluctuating intensities across the forehead region of the face and also consider the edge effects of the eyes.

Kun Fang et al [3] examines the use of the Naive Bayes approach in photos and compares its features, demonstrating that SURF feature description can improve classification outcomes when used to describe images.

Neal N. Xiong et al [4] The authors of this work proposed a revolutionary real-time color image segmentation technique based on RGB color space color similarity. The technique was tested on several fire photos and is resistant to noise and variations in illumination. The outcomes demonstrated that the approach is capable of successfully segmenting fire zones in time.

Kuan-Yu Chou et al [5] proposes a real time multi facefaces system based on Naive Bayesian classifier using field programmable gate array

Diah Harnoni Apriyanti et al [6] developed a new color detection model where it uses color labels and deep learning for color detection in flowers.

real time object detection is a challenging task so Varsha S.Futane et al [7] deduce a method to track real time object in video data set using Naïve Bayes Classifier

III. Methodology

We have the R, G, and B values. Now we need another function that will return us the color name from the RGB values. We do this by calculating distance (d) for the colors and each of the color names in our dataset. The color name with the least distance is the color name to the original color.

A. Working on Color Detection with OpenCV and Naïve Bayes Algorithm:

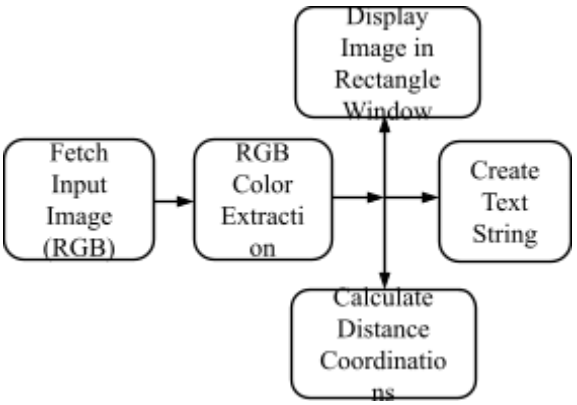


Figure 1: Architecture Diagram

The above architecture shows the capability for the project. It consists of a well-defined sequence diagram that is abstracted from the source code. It leverages the rich capabilities of the technology such as the OpenCV library in Python. The above architecture makes the process more efficient based on principles and properties related to each other. As we know that Red, Green, and Blue are the primary colors that can be mixed to produce different colors. The present color detection project takes the path of an image as an input and looks for the composition of three different colors red, green and blue in the given image.

Naive Bayes Classification on DataSet: It is a classification-based technique based on the Bayes theorem. In simple terms, a Naïve Bayes classifier assumes that the presence of a particular feature in a given Class *Fig.2,* is unrelated to the presence of any feature.

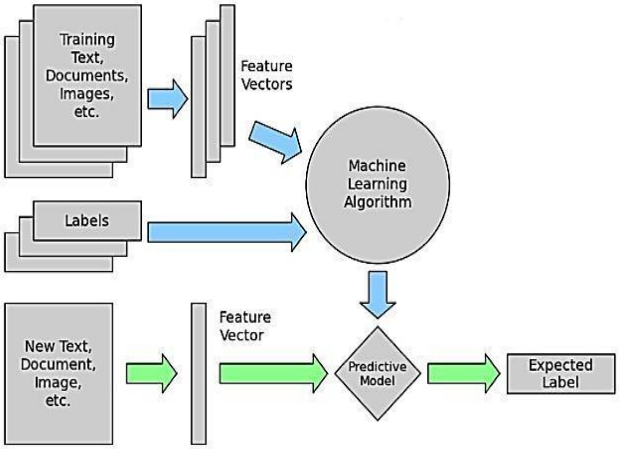


Figure 2: Class is unrelated to the presence of any feature

Naïve Bayes is the most efficient algorithm of machine learning and data mining. Its competitive performance in classification [8] is surprising because the conditional independence assumption on which it is based is always true in the real world.

B. The Dataset:

Colors are made up of 3 primary colors: red, green, and blue. In computers, we define each color value within a range of 0 to 255. So in how many ways can we define a color? The answer is 256*256*256 = 16,581,375.

There are approximately 16.5 million different ways to represent a color. In our dataset, we need to map each color's values with their corresponding names. But don't worry because we don't need to map all the values. We will be using a dataset that contains RGB values with their corresponding names.

The three components in the RGB color space, which is highly relevant and will be changed accordingly as long as the brightness is changed. RGB is a non-uniform color space, so the perception of differences (color) between the two colors cannot stand for the distance between two points in the color space.

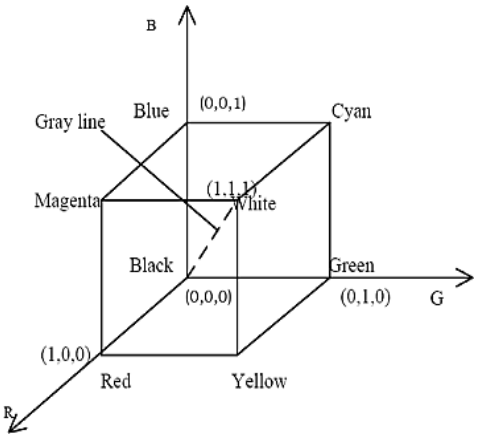


figure 3: RGB Color Model

Thus, the RGB color space is often converted to the other color spaces, such as HSI, HSV, the CIE, and Lab, by using linear or nonlinear transform in image processing. However, the original image we have collected usually is the RGB space, color space conversion will increase the amount of computation. There are many segmentation methods by using RGB color space, for example, license location [16] gets the license plate area accurately by calculating the contrast in the RGB components, reducing the calculated amount.

C. OpenCV:

OpenCV (Open Source Computer Vision) is a library for computer vision that includes numerous highly optimized algorithms that are used in Computer vision tasks. The library has more than 2500 algorithms and is capable of processing images and videos to detect faces, identify objects, classify human actions, track moving objects, color detection, pattern recognition and many more.

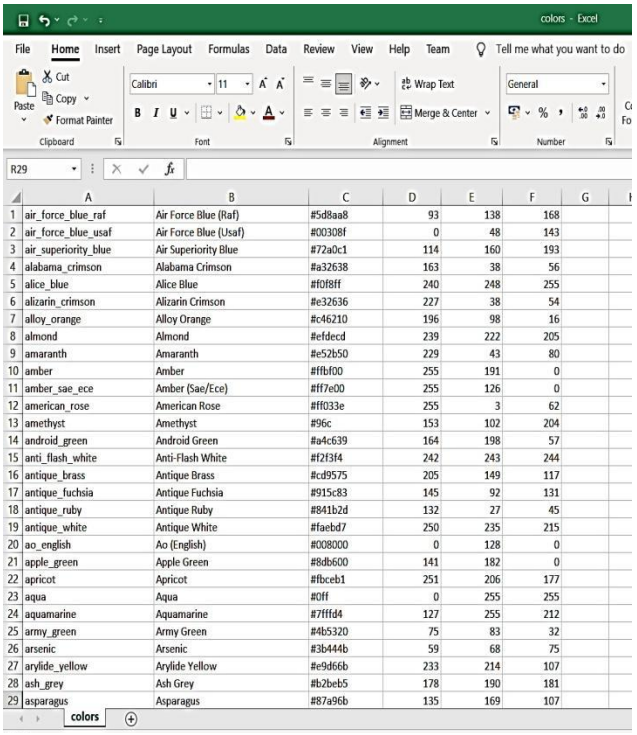
OpenCV is supported by multiple platforms including Windows, Linux, and macOS and is available for use in multiple languages as well (C, C++, Java, Python, etc.). OpenCV's library is widely used in Python for building real-time Machine Learning and Deep Learning applications.

Its cross-platform support and availability in multiple programming languages allow us to develop applications that can be used on different systems. OpenCV is also fast and efficient as compared to the other libraries for computer vision (Caffe, Keras, etc.).

Pandas pd:

Pandas (all lowercase) is a popular Python-based data analysis toolkit which can be imported using import pandas as pd. It presents a diverse range of utilities, ranging from parsing multiple file formats to converting an entire data table into a NumPy matrix array.

IV. The Color Detection Dataset:



	A	B	C	D	E	F	G
1	air_force_blue_raf	Air Force Blue (Raf)	#5d8aa8	93	138	168	
2	air_force_blue_usaf	Air Force Blue (Usaf)	#00308f	0	48	143	
3	air_superiority_blue	Air Superiority Blue	#72a0c1	114	160	193	
4	alabama_crimson	Alabama Crimson	#a32638	163	38	56	
5	alice_blue	Alice Blue	#f0f8ff	240	248	255	
6	alizarin_crimson	Alizarin Crimson	#e32636	227	38	54	
7	alloy_orange	Alloy Orange	#c46210	196	98	16	
8	almond	Almond	#efdecd	239	222	205	
9	amaranth	Amaranth	#e52b50	229	43	80	
10	amber	Amber	#ffbf00	255	191	0	
11	amber_sae_ece	Amber (Sae/Ece)	#ff7e00	255	126	0	
12	american_rose	American Rose	#ff033e	255	3	62	
13	amethyst	Amethyst	#96c	153	102	204	
14	android_green	Android Green	#a4c639	164	198	57	
15	anti_flash_white	Anti-Flash White	#f2f3f4	242	243	244	
16	antique_brass	Antique Brass	#cd9575	205	149	117	
17	antique_fuchsia	Antique Fuchsia	#915c83	145	92	131	
18	antique_ruby	Antique Ruby	#841b2d	132	27	45	
19	antique_white	Antique White	#faebd7	250	235	215	
20	ao_english	Ao (English)	#008000	0	128	0	
21	apple_green	Apple Green	#8db600	141	182	0	
22	apricot	Apricot	#fbceb1	251	206	177	
23	aqua	Aqua	#0ff	0	255	255	
24	aquamarine	Aquamarine	#7fffd4	127	255	212	
25	army_green	Army Green	#4b5320	75	83	32	
26	arsenic	Arsenic	#3b444b	59	68	75	
27	arylide_yellow	Arylide Yellow	#e9d66b	233	214	107	
28	ash_grey	Ash Grey	#b2beb5	178	190	181	
29	asparagus	Asparagus	#87a96b	135	169	107	

colors

figure 4: Data Set

The dataset we'll use, for this python project, will be called colors.csv. This dataset has 865×6 entries, with this

865 colors we're going to approximately find 1.6 million colors.

The first column contains the name of the color, the second contains the name of the color which will be displayed to the user, the third columns have the Hexadecimal values and fourth, fifth and sixth column has the RGB values.

V. Steps for Detecting Colors in Python

A. Make necessary Libraries:

import cv2
import pandas as pd

```
import cv2
import pandas as pd
```

figure 5: Importing Libraries

B. Path to a picture:

The first step is to fetch a high-quality image with good resolution. To load an image from a file, we use Cv2.imread().

The image should be in the working directory or the full path of the image should be provided Img=cv2.imread(img path).

img_path=r'C:/Users/lavanya/OneDrive/Documents/Lava Documents/Measi/ml/colorpic.jpg'
img = cv2.imread(img_path)

```
img_path = r'C:/Users/Lavanya/OneDrive/Documents/Lava Documents/Measi/ml/colorpic.jpg'
img = cv2.imread(img_path)
```

figure 6: Setting Path To A Picture

C. Declaring Global Variables:

The variable clicked which is boolean is to make sure we have clicked on a picture or not to a RGB values and the X and Y positions. Where r, g, b is used to store the RGB value of the pixel that's clicked on.Variable x_pos and y_pos is used to store x and y coordinates that user clicked on.

clicked = False
r = g = b = x_pos = y_pos = 0

```
# declaring global variables (are used later on)
clicked = False
r = g = b = x_pos = y_pos = 0
```

figure 7: Declaring Global Variable

D. Reading csv file with pandas and giving names to each column:

In this phase, the 3 layered colors are extracted from the input image. All the color images on screens such as televisions, computers, monitors, laptops and mobile screens are produced by the combination of Red, Green and Blue light.

Each primary color takes an intensive value 0 (lowest) to 255 (highest). When mixing 3 primary colors at different intensity levels a variety of colors are produced. For Example: If the intensity value of the primary colors is 0, this linear combination corresponds to black. If the intensity value of the primary colors is 1, this linear combination corresponds to white.

index = ["color", "color_name", "hex", "R", "G", "B"]
csv =pd.read_csv('colors.csv', names=index, header=None

```
# Reading csv file with pandas and giving names to each column
index = ["color", "color_name", "hex", "R", "G", "B"]
csv = pd.read_csv('colors.csv', names=index, header=None)
```

figure 8: Reading Csv File With Pandas and giving names to column

E. Calculating minimum distance from all colors and get the most matching color:

The minimum distance is calculated by considering moving towards the origin point from all colors to get the most matching color.

The pandas library serves as an important utility to perform various operations on comma-separated values, pd.read_csv() reads the csv file and loads it into the pandas data frame.

defget_color_name(R, G, B):
minimum = 10000
for i in range(len(csv)):
d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))
if d <= minimum:
minimum = d
cname = csv.loc[i, "color_name"]
return cname

```
# function to calculate minimum distance from all colors and get the most matching color
def get_color_name(R, G, B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.
        if d <= minimum:
            minimum = d
            cname = csv.loc[i, "color_name"]
    return cname
```

figure 9: To Calculate the minimum distance

F. Function to get x,y coordinates of mouse double click:

The rectangle window is used to display the image with shades of color. After the double-click is triggered, the RGB values and color name is updated.

To display an image Cv2.imshow () method is used. By using cv2.rectangle and cv2.putText() functions, the color name and its intensity level can be obtained.

defdraw_function(event, x, y, flags, param):
if event == cv2.EVENT_LBUTTONDBLCLK:
global b, g, r, x_pos, y_pos, clicked
clicked = True
x_pos = x
y_pos = y
b, g, r = img[y, x]
b = int(b)
g = int(g)
r = int(r)
cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_function)
while True:
cv2.imshow("image", img)

if clicked:

 # cv2.rectangle(image, start point, endpoint, color, thickness)-1 fills entire rectangle

cv2.rectangle(img, (20, 20), (750, 60), (b, g, r), -1)

 # Creating text string to display(Color name and RGB values)

text = get_color_name(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)

#

cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType)

cv2.putText(img, text, (50, 50), 2, 0.8, (255, 255, 255), 2, cv2.LINE_AA)

 # For very light colours we will display text in black colour

if r + g + b >= 600:

cv2.putText(img, text, (50, 50), 2, 0.8, (0, 0, 0), 2, cv2.LINE_AA)

clicked = False

```
# function to get x,y coordinates of mouse double click
def draw_function(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b, g, r, x_pos, y_pos, clicked
        clicked = True
        x_pos = x
        y_pos = y
        b, g, r = img[y, x]
        b = int(b)
        g = int(g)
        r = int(r)


cv2.namedWindow('image')
cv2.setMouseCallback('image', draw_function)

while True:

    cv2.imshow("image", img)
    if clicked:

        # cv2.rectangle(image, start point, endpoint, color, thickness)-1 fills entire rectar
        cv2.rectangle(img, (20, 20), (750, 60), (b, g, r), -1)

        # Creating text string to display( Color name and RGB values )
        text = get_color_name(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)

        # cv2.putText(img,text,start,font(0-7),fontScale,color,thickness,lineType )
        cv2.putText(img, text, (50, 50), 2, 0.8, (255, 255, 255), 2, cv2.LINE_AA)

        # For very light colours we will display text in black colour
        if r + g + b >= 600:
            cv2.putText(img, text, (50, 50), 2, 0.8, (0, 0, 0), 2, cv2.LINE_AA)

        clicked = False
```

figure 10: Function To Get x,y Coordinates Of Mouse When Double Click

G. Break the loop when user hits 'esc' key:

The Escape key to close the image window.

if cv2.waitKey(20) & 0xFF == 27:

break

```
# Break the loop when user hits 'esc' key
if cv2.waitKey(20) & 0xFF == 27:
    break
```

figure 11: To Break The Loop When ESC Key Is Pressed

H. Output:

Double click on the window to know the name of the pixel color.

figure 12: Output Image With Color Intensity RGB Values As R=252 G=42 B=17 For Red

figure 13: Output Image With Color Intensity RGB Values As R=252 G=229 B=13 For Golden Yellow

Figure 13: Output Image With Color Intensity Rgb Values As R=3 G=9 B=97 For Royal Blue

VI. Conclusion

In this Research area with all methodology and results we learned about colors and therefore the way we will extract RGB values and the color name of a pixel. We learned the way to handle events like double clicking on the window and saw the way to read CSV files with pandas and apply the OpenCV library to perform operations on data. This is often utilized in numerous image editing and drawing apps, this is an easy task for humans to detect the color and choose one. But the computer on the other hand cannot detect the color easily. It is a tough task for a computer to detect the colors. So that's the reason behind choosing this project. Naive Bayes algorithm, Pandas and OpenCV libraries are used in python languages. Naive Bayes may be a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of

feature values, where the category labels are drawn from some finite set.

References

1. Early Diabetes Discovery From Tongue Images , Safia Naveed S, Geetha G and Leninisha S, The Computer Journal, 2020, https://doi.org/10.1093/comjnl/bxaa022

2. Intelligent Diabetes Detection System based on Tongue Datasets, Safia Naveed S, GurunathanGeetha ,Current Medical Imaging Reviews 2019;15(7):672-678, doi: 10.2174/1573405614666181009133414

3. Naive Bayes Image Classification Based on Multiple Features Kun Fang Computer Software and Media Applications (2019) Volume 2 doi: 10.24294/csma.v2i1.1171

4. Color Sensor And Their Application Based On Real-Time Color Image Segmentation For Cyber Physical System Neal N. Xiong Yang Shen, Kangye Yang, Changhoon Lee and Chunxue Wu* Xiong *et al. EURASIP Journal on Image and Video Processing* (2018) 2018:23 doi: 10.1186/s13640-018-0258-x

5. Real-Time and Low-Memory Multi-Faces Detection System Design With Naive Bayes Classifier Implemented on FPGAKuan-Yu Chou and Yon-Ping Chen IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 30, NO. 11, NOVEMBER 2020 doi:10.1109/TCSVT.2019.2955926

6. Automated color detection in orchids using color labels and deep learning Diah Harnoni Apriyanti 12*, Luuk J. Spreeuwers', Peter J. F. Lucas[13], Raymond N. J. Veldhuis[1] plos journal doi:10.1371/journal.pone.0259036

7. Proficient Use of Naïve Bayes Classifier in Object Tracking Varsha S. Futane Anilkumar N. Holambe Asian Journal of Convergence in Technology Volume1, Issue 6 Issn No.:2350-1146,I.F-2.71