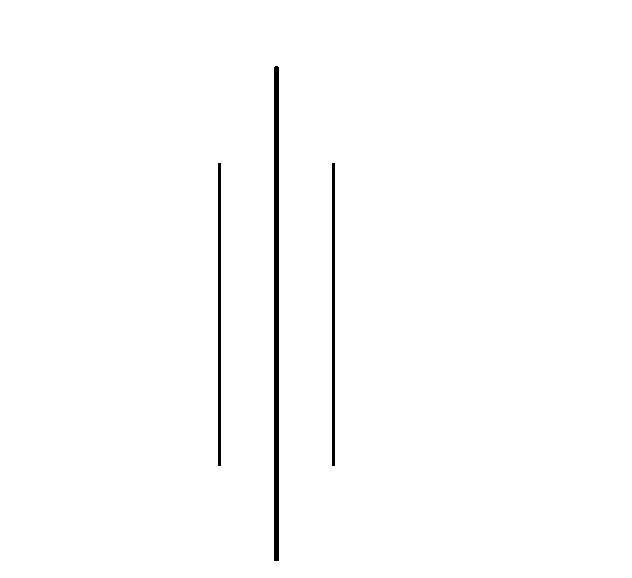
NEPAL ENGINEERING COLLEGE

( Affiliated To Pokhara University )

Changunarayan, Bhaktapur



Report on

# Lab 1a: Image Acquisition and Display

SUBMITTED BY : SUBMITTED TO:

Name: Rishi Raj Pandey Department of Computer Roll no: 021 – 366 Science and Engineering Date: 08 / 01 / 2082 Department of Computer Science and Engineering

**Objectives**: To read an RGB image, display it, convert it to grayscale, and generate variations including copy, negative, grayscale, red/green/blue saturated versions, and save each result.

1. **Convert the Given RGB image into**

* Copy Image
* Negative Image
* Gray Image
* Save Image

**Code**

namespace Lab1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

Bitmap originalImage;

private void btnLoadAndProcess\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp";

if (open.ShowDialog() == DialogResult.OK)

{

// Load Original Image

originalImage = new Bitmap(open.FileName);

pictureBox1.Image = originalImage;

// Copy Image

Bitmap copyImage = new Bitmap(originalImage);

pictureBox2.Image = copyImage;

copyImage.Save("copy\_image.png");

// Negative Image

Bitmap negativeImage = MakeNegative(copyImage);

pictureBox3.Image = negativeImage;

negativeImage.Save("negative\_image.png");

// Grayscale Image

Bitmap grayImage = MakeGrayscale(copyImage);

pictureBox4.Image = grayImage;

grayImage.Save("gray\_image.png");

MessageBox.Show("Images saved successfully.");

}

}

private Bitmap MakeNegative(Bitmap image)

{

Bitmap negative = new Bitmap(image.Width, image.Height);

for (int y = 0; y < image.Height; y++)

{

for (int x = 0; x < image.Width; x++)

{

Color pixel = image.GetPixel(x, y);

Color negPixel = Color.FromArgb(255 - pixel.R, 255 - pixel.G, 255 - pixel.B);

negative.SetPixel(x, y, negPixel);

}

}

return negative;

}

private Bitmap MakeGrayscale(Bitmap image)

{

Bitmap gray = new Bitmap(image.Width, image.Height);

for (int y = 0; y < image.Height; y++)

{

for (int x = 0; x < image.Width; x++)

{

Color pixel = image.GetPixel(x, y);

int grayVal = (int)(0.3 \* pixel.R + 0.59 \* pixel.G + 0.11 \* pixel.B);

Color grayPixel = Color.FromArgb(grayVal, grayVal, grayVal);

gray.SetPixel(x, y, grayPixel);

}

}

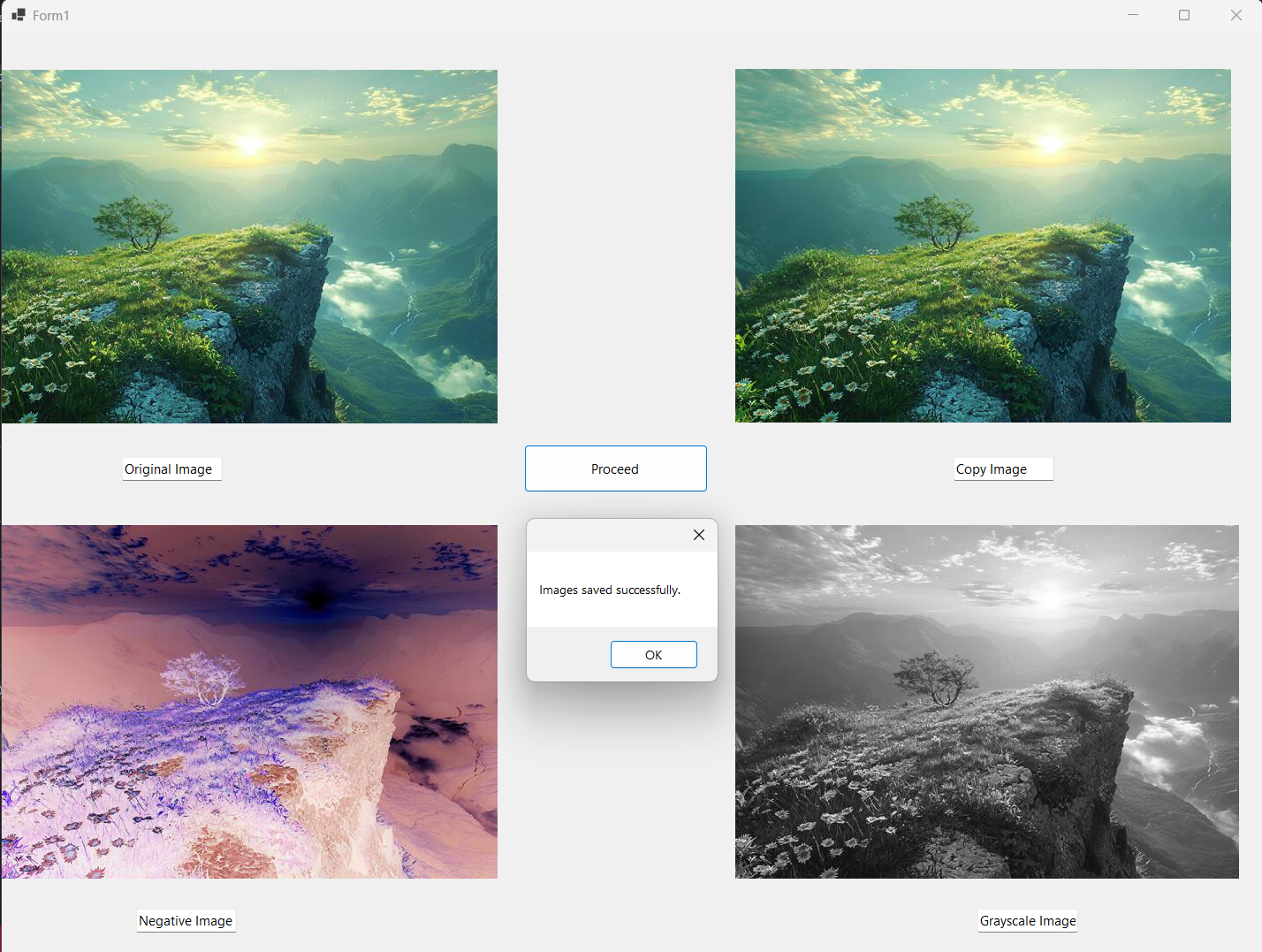
return gray;

}

}

}

**OUTPUT**



1. **Convert the Given RGB image into**

* Red Saturation Image
* Blue Saturation Image
* Green Saturation Image
* Save Image

**Code**

namespace Lab1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

Bitmap originalImage;

private void btnLoadAndProcess\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

open.Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp";

if (open.ShowDialog() == DialogResult.OK)

{

// Load Original Image

originalImage = new Bitmap(open.FileName);

pictureBox1.Image = originalImage;

// Red Saturation

Bitmap redImage = SaturateColor(originalImage, "red");

pictureBox2.Image = redImage;

redImage.Save("red\_saturation.png");

// Green Saturation

Bitmap greenImage = SaturateColor(originalImage, "green");

pictureBox3.Image = greenImage;

greenImage.Save("green\_saturation.png");

// Blue Saturation

Bitmap blueImage = SaturateColor(originalImage, "blue");

pictureBox4.Image = blueImage;

blueImage.Save("blue\_saturation.png");

MessageBox.Show("Saturation images saved successfully.");

}

}

private Bitmap SaturateColor(Bitmap image, string color)

{

Bitmap saturated = new Bitmap(image.Width, image.Height);

for (int y = 0; y < image.Height; y++)

{

for (int x = 0; x < image.Width; x++)

{

Color pixel = image.GetPixel(x, y);

int r = 0, g = 0, b = 0;

switch (color.ToLower())

{

case "red":

r = pixel.R;

break;

case "green":

g = pixel.G;

break;

case "blue":

b = pixel.B;

break;

}

saturated.SetPixel(x, y, Color.FromArgb(r, g, b));

}

}

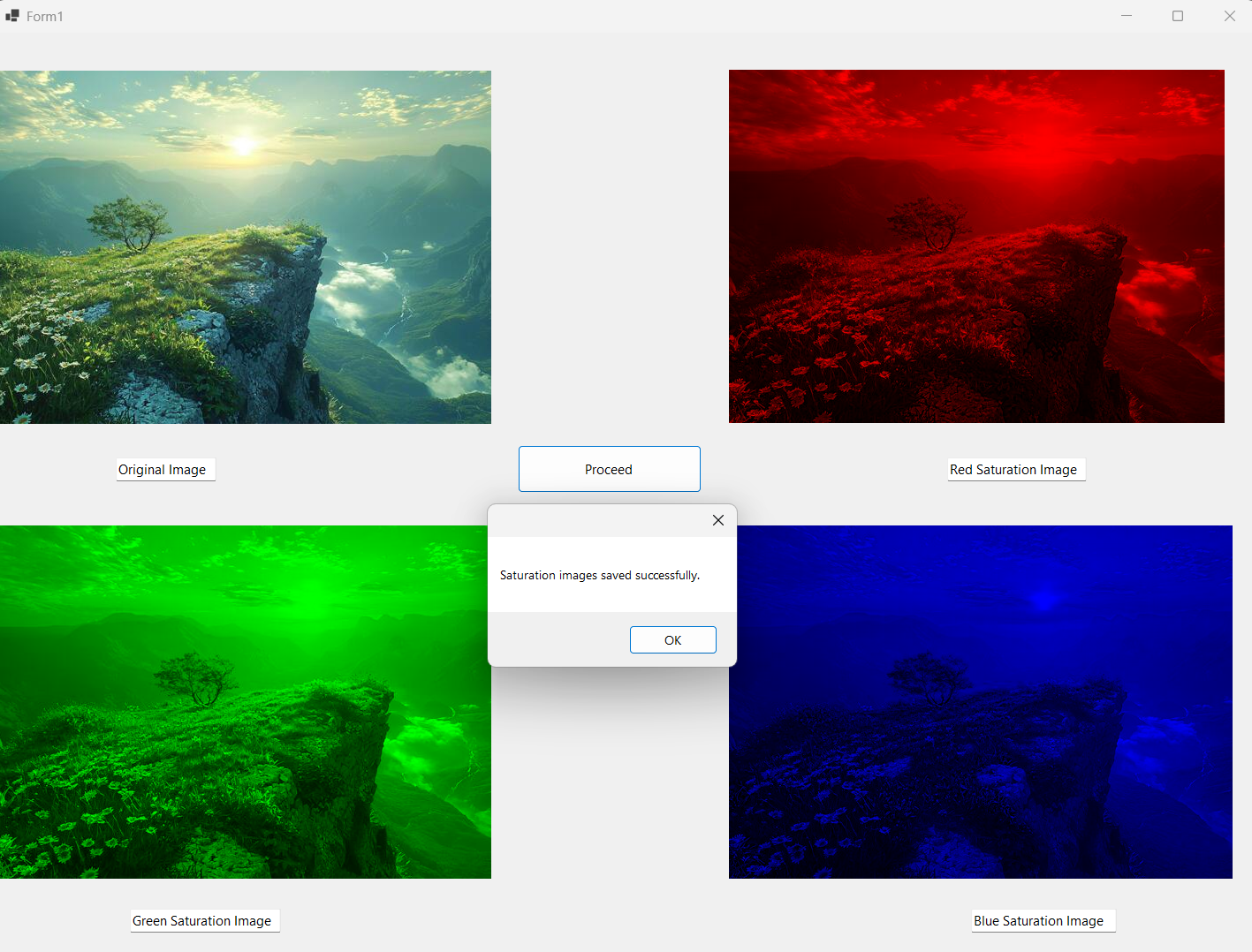
return saturated;

}

}

}

**OUTPUT**

****

**OBSERVATION**

Upon executing the image processing tasks, the original RGB image was successfully read and displayed. Various transformations were applied, including creating a copy, generating a negative, converting to grayscale, and producing red, green, and blue saturation images. Each transformed image was displayed correctly and saved to the desired location, confirming that the image manipulation functions worked as intended

**CONCLUSION**

The experiment demonstrated effective techniques for basic image acquisition and processing using programming. It successfully highlighted how RGB images can be manipulated to extract grayscale and color-specific information. These operations are fundamental in image processing and lay the groundwork for more advanced techniques in computer vision and digital image analysis.