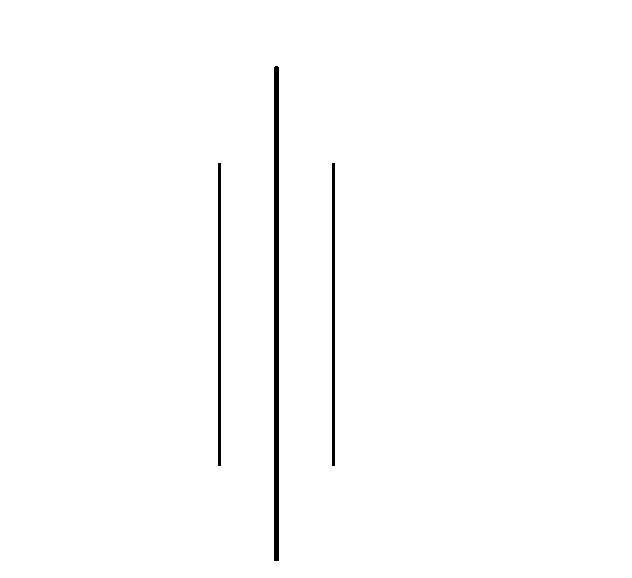
NEPAL ENGINEERING COLLEGE

( Affiliated To Pokhara University )

Changunarayan, Bhaktapur



Report on

# Lab 4: Histogram Equalization

SUBMITTED BY : SUBMITTED TO:

Name: Rishi Raj Pandey Department of Computer Roll no: 021 – 366 Science and Engineering Date: 01 / 03 / 2082 Department of Computer Science and Engineering

**Objectives**: To implements histogram equalization, negative transformation and gamma correction to improve image contract and brightness.

**Convert the Given RGB image into**

* Load Image
* Histogram Equalization
* Gamma Corrected
* Negative
* Save image in different formats

**Code**

namespace lab4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Bitmap originalBitmap;

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

{

OpenFileDialog ofd = new OpenFileDialog();

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

{

originalBitmap = new Bitmap(ofd.FileName);

pictureBox1.Image = originalBitmap;

ShowImage(originalBitmap, "Original Image");

ShowImage(DrawHistogram(originalBitmap), "Original Histogram");

}

}

static void ShowImage(Bitmap bmp, string title)

{

Form form = new Form();

form.Text = title;

form.ClientSize = new Size(bmp.Width, bmp.Height);

PictureBox pb = new PictureBox();

pb.Dock = DockStyle.Fill;

pb.Image = bmp;

pb.SizeMode = PictureBoxSizeMode.StretchImage;

form.Controls.Add(pb);

form.Show();

}

private Bitmap NegativeTransformation(Bitmap bmp)

{

Bitmap result = new Bitmap(bmp.Width, bmp.Height);

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

{

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

{

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

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

result.SetPixel(x, y, negative);

}

}

return result;

}

private Bitmap GammaCorrection(Bitmap bmp, double gamma)

{

Bitmap result = new Bitmap(bmp.Width, bmp.Height);

byte[] gammaLUT = new byte[256];

for (int i = 0; i < 256; i++)

{

gammaLUT[i] = (byte)(Math.Min(255,

(int)((255.0 \* Math.Pow(i / 255.0, gamma)) + 0.5)));

}

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

{

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

{

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

Color corrected = Color.FromArgb(

gammaLUT[pixel.R],

gammaLUT[pixel.G],

gammaLUT[pixel.B]);

result.SetPixel(x, y, corrected);

}

}

return result;

}

static Bitmap DrawHistogram(Bitmap img)

{

int[] hist = new int[256];

int width = img.Width;

int height = img.Height;

for (int y = 0; y < height; y++)

for (int x = 0; x < width; x++)

hist[img.GetPixel(x, y).R]++;

int histHeight = 200;

Bitmap bmp = new Bitmap(256, histHeight);

int max = 0;

foreach (int val in hist)

if (val > max) max = val;

using (Graphics g = Graphics.FromImage(bmp))

{

g.Clear(Color.White);

for (int i = 0; i < 256; i++)

{

float pct = (float)hist[i] / max;

g.DrawLine(Pens.Black, new Point(i, histHeight),

new Point(i, histHeight - (int)(pct \* histHeight)));

}

}

return bmp;

}

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

{

if (pictureBox1.Image != null)

{

pictureBox2.Image = NegativeTransformation((Bitmap)pictureBox1.Image);

ShowImage((Bitmap)pictureBox2.Image, "Negative Image"); }

}

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

{

if (pictureBox1.Image != null)

{

pictureBox2.Image = GammaCorrection((Bitmap)pictureBox1.Image, 2.2);

ShowImage((Bitmap)pictureBox2.Image, "Gamma Corrected Image");

}

}

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

{

if (pictureBox1.Image != null)

{

Bitmap equalized = HistogramEqualizations((Bitmap)pictureBox1.Image);

pictureBox2.Image = equalized;

ShowImage(equalized, "Equalized Image");

ShowImage(DrawHistogram(equalized), "Equalized Histogram");

Bitmap grayImage = ToGrayscale(originalBitmap);

}

}

static Bitmap ToGrayscale(Bitmap input)

{

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

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

{

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

{

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

int grayVal = (int)(0.299 \* pixel.R + 0.587 \* pixel.G + 0.114 \* pixel.B);

gray.SetPixel(x, y, Color.FromArgb(grayVal, grayVal, grayVal));

}

}

return gray;

}

private Bitmap HistogramEqualizations(Bitmap bmp)

{

Bitmap result = new Bitmap(bmp.Width, bmp.Height);

int width = bmp.Width;

int height = bmp.Height;

int totalPixels = width \* height;

int[] histogram = new int[256];

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

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

int gray = (pixel.R + pixel.G + pixel.B) / 3;

histogram[gray]++;

}

}

int[] cdf = new int[256];

cdf[0] = histogram[0];

for (int i = 1; i < 256; i++)

cdf[i] = cdf[i - 1] + histogram[i];

byte[] equalized = new byte[256];

for (int i = 0; i < 256; i++)

{

equalized[i] = (byte)((cdf[i] - cdf[0]) \* 255 / (totalPixels - cdf[0]));

}

for (int y = 0; y < height; y++)

{

for (int x = 0; x < width; x++)

{

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

int gray = (pixel.R + pixel.G + pixel.B) / 3;

byte eqGray = equalized[gray];

result.SetPixel(x, y, Color.FromArgb(eqGray, eqGray, eqGray));

}

}

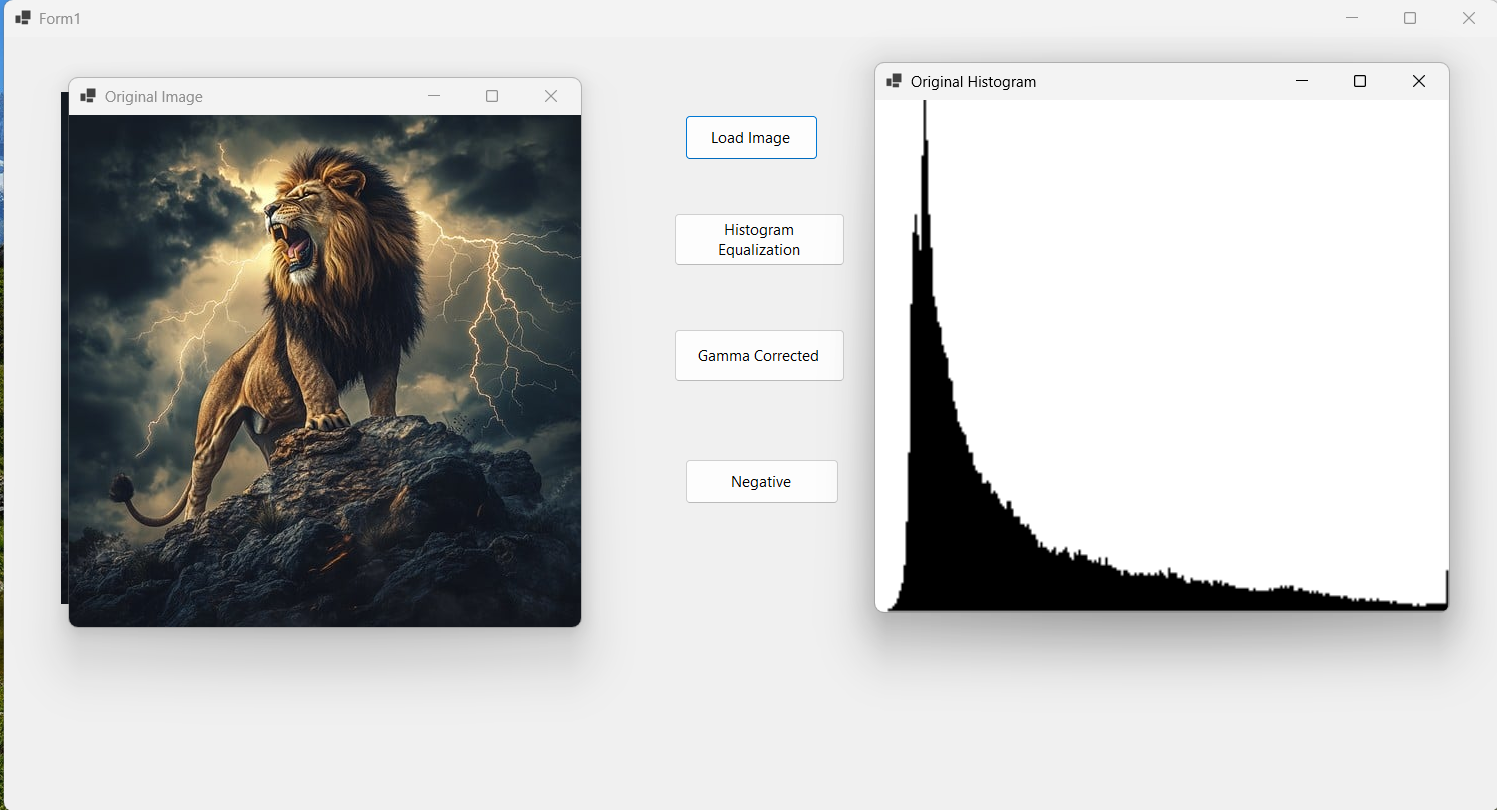
return result;

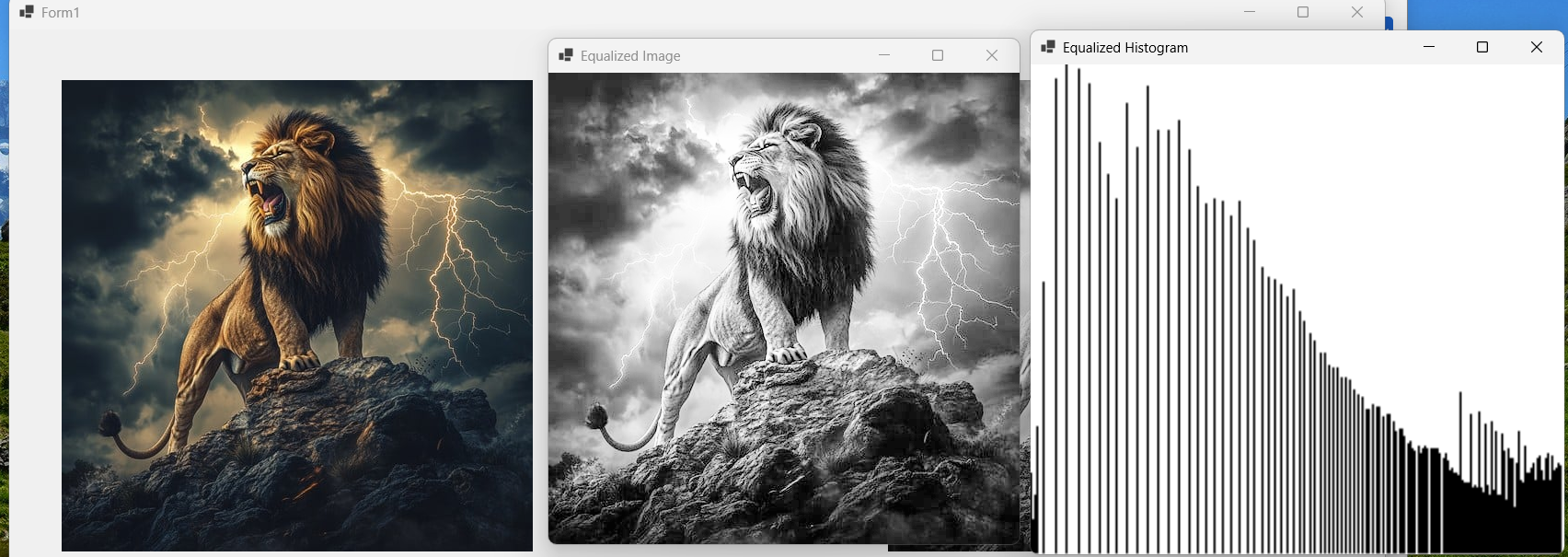
}

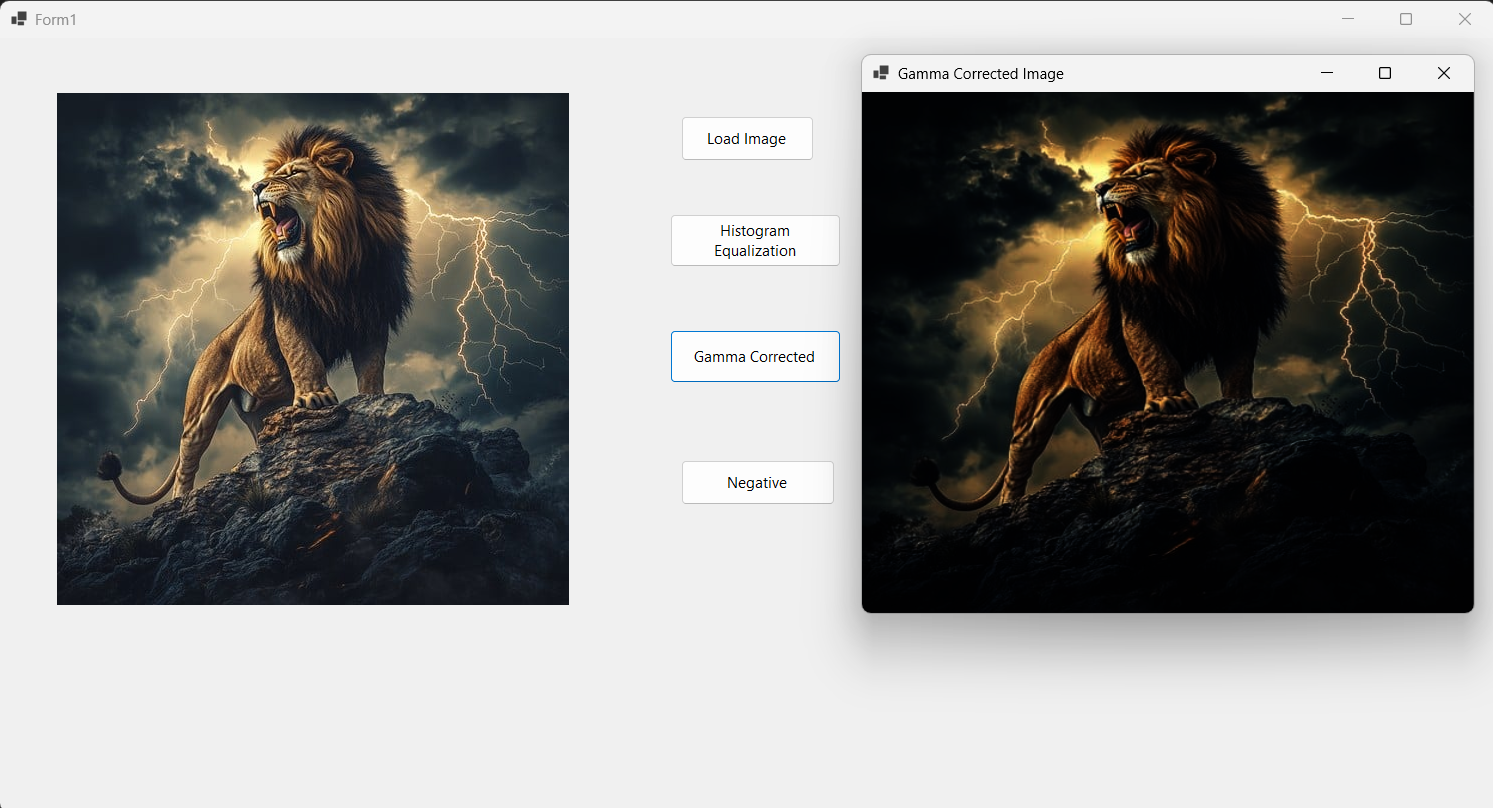
}

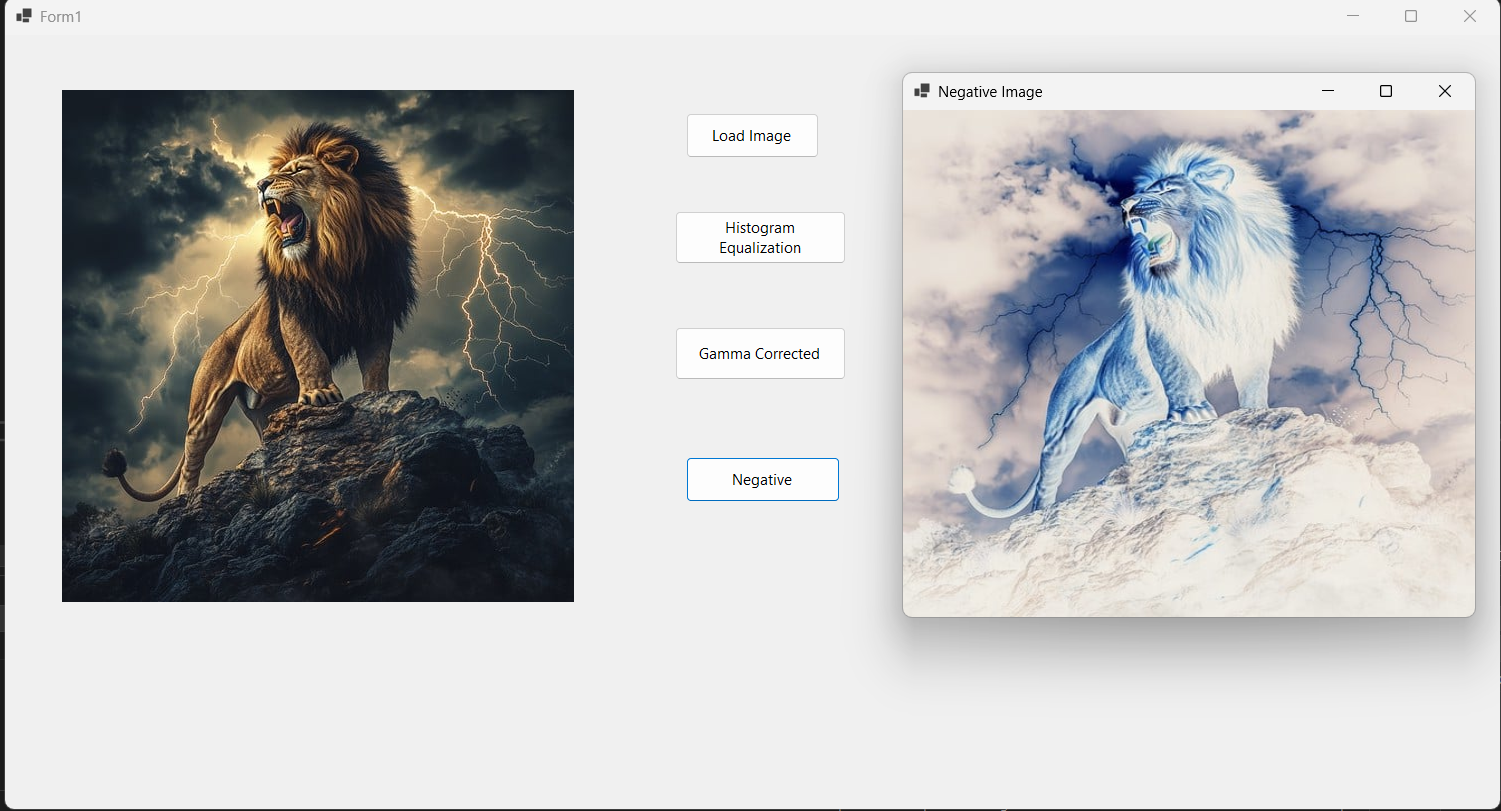
}

**OUTPUT**

****

****

****

****

**OBSERVATION**

The RGB image was successfully loaded using libraries like OpenCV or PIL, preserving its color channels for further processing.

Histogram Equalization improved the image contrast by redistributing pixel intensity values, making hidden details more visible.

Gamma Correction adjusted image brightness based on the gamma value, enhancing visibility in overexposed or underexposed regions.

Negative Transformation inverted the pixel values, highlighting bright areas and offering an alternative visual interpretation.

All processed images were saved in formats like PNG, JPEG, and BMP, with slight differences in quality and file size.

**CONCLUSION**

The lab effectively demonstrated image enhancement using histogram equalization, gamma correction, and negative transformation. Each method served to improve contrast, brightness, or visibility.

Saving images in different formats showed the impact of format choice on quality and storage. Overall, the experiment provided useful insights into practical image enhancement techniques.