Laporan Resmi Praktikum Pengolahan Citra

Flip Image, Kuantisasi Citra & Adaptive Thresholding

Dimas Rizky H.P. – 2110141011 – 3 D4 IT A

***Flip Image***

* ***Source Code***

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace FlipImage

{

public partial class Form1 : Form

{

Bitmap objBitmap;

Bitmap objBitmap1;

public Form1()

{

InitializeComponent();

}

private void button3\_Click(object sender, EventArgs e)

{

DialogResult d = openFileDialog1.ShowDialog();

if (d == DialogResult.OK)

{

objBitmap = new Bitmap(openFileDialog1.FileName);

pictureBox1.Image = objBitmap;

}

}

private void button1\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

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

{

Color w = objBitmap.GetPixel(x, y);

objBitmap1.SetPixel(objBitmap.Width - 1 - x, y, w);

}

}

pictureBox2.Image = objBitmap1;

}

private void button2\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

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

{

Color w = objBitmap.GetPixel(x, y);

objBitmap1.SetPixel( x, objBitmap.Height - y - 1, w);

}

}

pictureBox2.Image = objBitmap1;

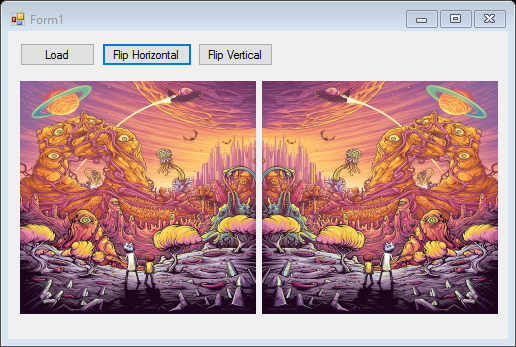
}

}

}

* ***Analisa***

Pada dasarnya proses flip Horizontal adalah meletakkan titik horisontal secara berkebalikan dimana setiap pixel x pada gambar asal akan diletakkan pada pixel width-(x+1) pada gambar hasil. Proses flip horizontal dapat dilihat pada *gambar 1.0* berikut ini



*Gambar 1.0 Flip Horizontal*

Untuk flip vertikal, sumbu yang dibalik adalah menggunakan patokan sumbu y, dimana pada dasarnya adalah pembalikan dan peletakan pixel pada height-(x+1). Contohnya dapat dilihat pada *gambar 1.1*.

**

*Gambar 1.1 Flip Vertical*

***Kuantisasi Citra***

* ***Source Code***

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace KuantisasiCitra

{

public partial class Form1 : Form

{

Bitmap objBitmap;

Bitmap objBitmap1;

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

DialogResult d = openFileDialog1.ShowDialog();

if(d == DialogResult.OK)

{

objBitmap = new Bitmap(openFileDialog1.FileName);

pictureBox1.Image = objBitmap;

}

}

private void button2\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for(int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

Color wb = Color.FromArgb(xg, xg, xg);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button3\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for(int y = 0; y< objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xbw = 0;

if (xg >= 128) xbw = 255;

Color wb = Color.FromArgb(xbw, xbw, xbw);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button4\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for(int y = 0; y <objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xk = 16 \* (int)(xg/16);

Color wb = Color.FromArgb(xk, xk, xk);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

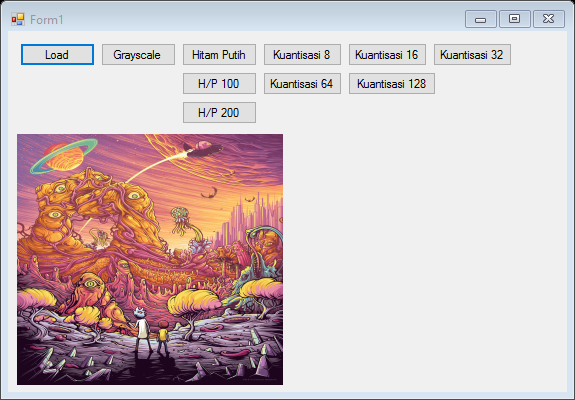
}

}

}

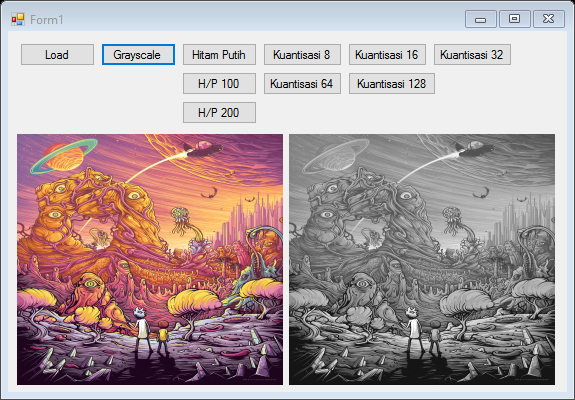
* ***Output Program***

Program tersebut akan menghasilkan interface yang dapat digunakan untuk melakukan kuantisasi citra dan memodifikasi citra yang ada. Berikut adalah hasil program dengan fitur lengkap yang akan digunakan.



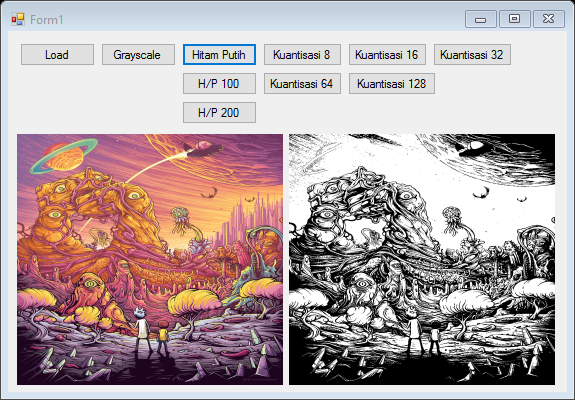
*Gambar 2.0 Interface program*

Terdapat beberapa button diantaranya adalah button grayscale, hitam putih dan kuantisasi 16 (default program sebelum ada tambahan program). Masing masing akan mengubah citra hasil load, menjadi citra masing masing fiturnya. Berikut adalah hasil programnya



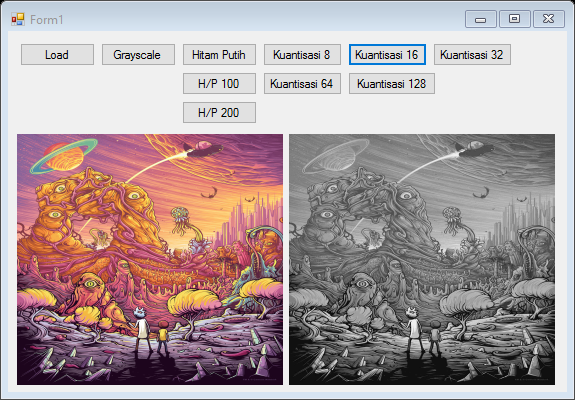
*Gambar 2.1 Fitur Grayscale*

Fitur grayscale akan merubah warna dimasing-masing pixel gambar input menjadi grayscale dengan nilai grayscale merupakan rata-rata dari nilai RGB per pixelnya. Hasil citra yang dihasilkan berupa image citra dengan mode grayscale

**

Gambar 2.2 Citra Hitam Putih dengan value threshold 128

Pada fitur hitam putih, citra yang dihasilkan adalah sebuah citra yang berisi warna hitam atau putih. Citra ini biasa digunakan untuk melakukan thresholding atau pendeteksian tepi dengan nilai valu threshold dapat di setting sesuai dengan tingkat detail dari gambar.



*Gambar 2.3 Citra kuantisasi 16*

Kuantisasi citra merupakan proses mengubah jumlah nilai keabuan, misalnya dari citra awal diubah menjadi 8 nilai derajat keabuan atau 16 nilai derajat keabuan. Pada *gambar 2.3* dapat dilihat hasil dari kuantisasi 16 membuat citra output memiliki nilai keabuan pada rentang 16.

* ***Analisa perbedaan grayscale dari nilai rata2 RGB dan layer RGB biasa***

Citra derajat keabuan adalah sebuah gambar yang hanya berisi terang dan gelap tanpa menggunakan warna. Semakin besar nilai derajat keabuan sebuah titik, maka titik akan terlihat semakin terang. Sebaliknya semakin kecil nilai derajat keabuan sebuah titik, titik akan terlihat semakin gelap.

Ada beberapa cara untuk mendapat citra derajat keabuan, yaitu menggunakan basis RGB dan rata2 RGB itu sendiri. Perbedaan yang mendasar antara dua pendekatan ini adalah, jika menggunakan basis masing2 RGB untuk mendapatkan citra derajat keabuan, maka hanya dalam rentang frekuensi warna tertentu saja yang mendapatkan grayscale sempurna yang artinya nilai grayscale nya tidak rata untuk setiap warna dasarnya. Sedangkan jika menggunakan rata rata RGB, hasil dari grayscale citra akan sama rata untuk setiap warna dasar yang ada.

* ***Program hitam putih dengan threshold 100, dan 200 & analisa***

Penambahan program diberikan dalam bentuk button callback, bernama *H/W 100* dan *H/W 200* dengan kode callback yang dipanggil seperti berikut :

private void button9\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xbw = 0;

if (xg >= 100) xbw = 255;

Color wb = Color.FromArgb(xbw, xbw, xbw);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button10\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xbw = 0;

if (xg >= 200) xbw = 255;

Color wb = Color.FromArgb(xbw, xbw, xbw);

objBitmap1.SetPixel(x, y, wb);

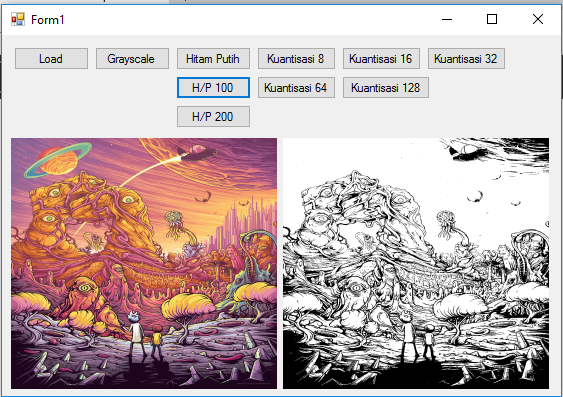
}

}

pictureBox2.Image = objBitmap1;

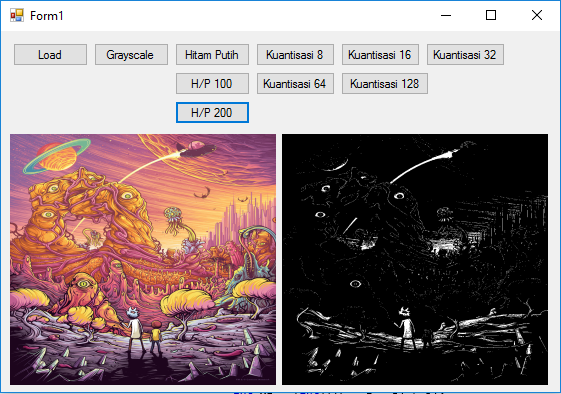
}

Pada button 9 diberikan threshold dengan value bernilai 100, sedangkan button 10 di assign dengan threshold bernilai 200, hasilnya dapat dilihat pada gambar 2.5 dan 2.6 berikut ini



*Gambar 2.5 Hitam Putih Dengan Threshold 100*

Threshold 100 artinya titik piksel yang nilainya kurang dari 100 akan diubah menjadi 0 atau menjadi warna hitam, sedangkan titik piksel yang nilainya di atas atau sama dengan 128 akan diubah menjadi 255 atau putih

**

Gambar 2.6 Hitam Putih Dengan Threshold 200

Sama halnya dengan threshold 100, threshold 200 akan mengubah warna seluruh pixel yang nilainya dibawah 200 menjadi 0 atau warna hitam dan mengubah yang nilainya di atas 200 atau sama dengan 200 menjadi 255 atau warna putih.

***Program kuantisasi citra 8, 32, 64 dan 128 & analisa***

Kuantisasi citra pada program tambahan ini berupa beberapa button untuk melakukan kuantisasi citra 8, 32, 64, dan 128. Berikut merupakan source code hasil penambahan button

private void button6\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xk = 8 \* (int)(xg / 8);

Color wb = Color.FromArgb(xk, xk, xk);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button7\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xk = 32 \* (int)(xg / 32);

Color wb = Color.FromArgb(xk, xk, xk);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button5\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xk = 64 \* (int)(xg / 64);

Color wb = Color.FromArgb(xk, xk, xk);

objBitmap1.SetPixel(x, y, wb);

}

}

pictureBox2.Image = objBitmap1;

}

private void button8\_Click(object sender, EventArgs e)

{

objBitmap1 = new Bitmap(objBitmap);

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

{

for (int y = 0; y < objBitmap1.Height; y++)

{

Color w = objBitmap.GetPixel(x, y);

int r = w.R;

int g = w.G;

int b = w.B;

int xg = (int)((r + g + b) / 3);

int xk = 128 \* (int)(xg / 128);

Color wb = Color.FromArgb(xk, xk, xk);

objBitmap1.SetPixel(x, y, wb);

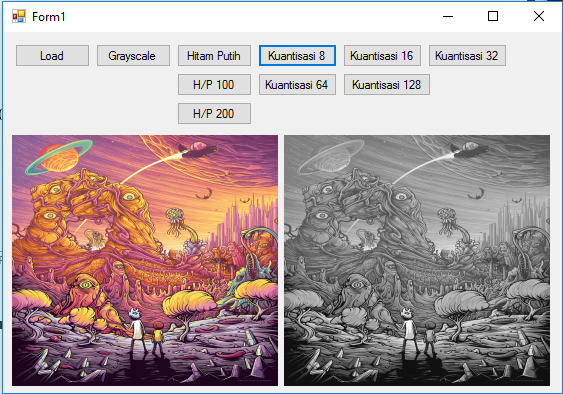
}

}

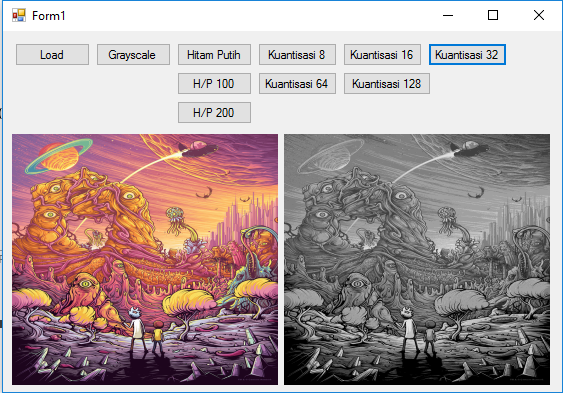
pictureBox2.Image = objBitmap1;

}

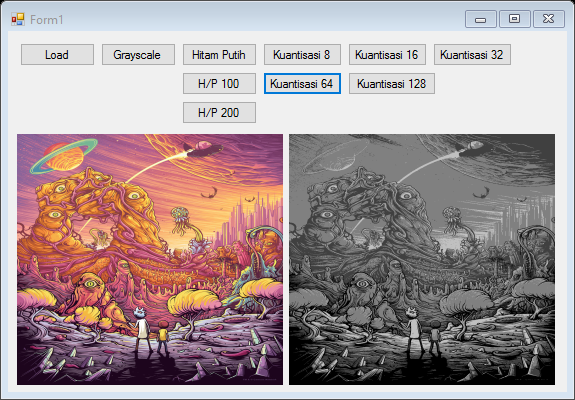
Kuantisasi masing masing nilai menghasilkan hasil output citra yang berbeda beda, perbedaan antara outputnya adalah sebagai berikut ini.



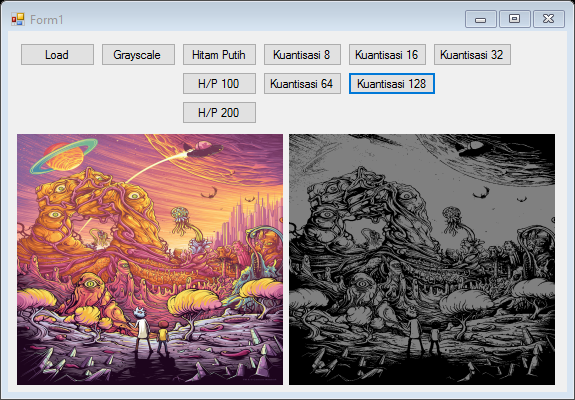
*Gambar 2.7 Kuantisasi 8*

**

*Gambar 2.8 Kuantisasi 32*

**

*Gambar 2.9 Kuantisasi 64*

**

*Gambar 2.10 Kuantisasi 128*

Dari beberapa kuantisasi tersebut terdapat perbedaan output yang terlihat, yaitu semakin besar nilai kuantisasinya, akan semakin sempit jarak warna keabuan yang dapat difilter pada gambar output yang dihasilkan dari proses kuantisasi citra gambar input.

Dengan semakin sempitnya range warna keabuan, semakin besar kuantisasi citra yang dilakukan akan semakin membuat gambar output menjadi lebih gelap

***Adaptive Thresholding***

* ***Source Code***

private void button3\_Click(object sender, EventArgs e)

{

gray = new Bitmap(asli);

gray2 = new int[gray.Width, gray.Height];

sumimage2 = new int[gray.Width , gray.Height ];

output = new int[gray.Width, gray.Height];

graynya = new int[gray.Width+1, gray.Height+1];

sumimage = new int[gray.Width+1, gray.Height+1];

thesum = new int[gray.Width + 1, gray.Height + 1];

mean = new int[gray.Width + 1, gray.Height + 1];

thresholdnya = new int[gray.Width + 1, gray.Height + 1];

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

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

{

Color w = gray.GetPixel(x, y);

int a = w.A;

int r = w.R;

int g = w.G;

int b = w.B;

int avg = ((r + g + b) / 3);

graynya[x+1,y+1]=avg;

sumimage[x+1, y+1] = avg;

gray2[x, y] = avg;

//sumimage2[x, y] = avg;

gray.SetPixel(x, y, Color.FromArgb(a, avg, avg, avg));

}

pictureBox1.Image = gray;

}

Berikut merupakan source code untuk merubah citra menjadi grayscale, untuk setelahnya akan dilakukan integral2sum untuk mempersiapkan adaptive thresholding. Untuk selanjutnya adalah koding dari integral2sum

private void button8\_Click(object sender, EventArgs e)

{

int sum ;

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

{

sum = 0;

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

{

sum = sum + gray2[x, y];

if (x == 0)

{

sumimage2[x, y] = sum;

}

else {

sumimage2[x, y] =sumimage2[x-1, y]+ sum;

}

}

}

label1.Text = "" + sumimage2[0, 0] + " " + sumimage2[0, 1] + " " + sumimage2[0, 2];

label2.Text = "" + sumimage2[1, 0] + " " + sumimage2[1, 1] + " " + sumimage2[1, 2];

}

Setelah dilakukan integral2sum, maka citra grayscale akan memiliki range derajat keabuan yang tidak terbatas hanya dalam range yang sama namun juga memiliki range yang dinamis bergantung dengan nilai sebaran integral2sum nya. Hal ini memungkinkan untuk dilakukannya adaptive thresholding, berikut adalah koding untuk adaptive thresholding

private void button9\_Click(object sender, EventArgs e)

{

int x1, x2, y1, y2,count;

int sum;

// ini aku ganti, aslinya 7,35

for (int x = s / 2 + 1; x < gray.Width - s / 2; x++) //4,3

{

for (int y = s / 2 + 1; y < gray.Height - s / 2; y++) //4,3

{

x1 = x - s / 2;

x2 = x + s / 2;

y1 = y - s / 2;

y2 = y + s / 2;

count = (x2 - x1) \* (y2 - y1);

sum = sumimage2[x2, y2] - sumimage2[x2, y1 - 1] - sumimage2[x1 - 1, y2] + sumimage2[x1 - 1, y1 - 1];

if (gray2[x, y] \* count <= (sum \* (100 - t) / 100))

{

output[x, y] = 0;

}

else {

output[x, y] = 255;

}

}

}

}

Penggunaan adaptive thresholding ini akan mendeteksi tepi tepi object yang sebelumnya sudah dibuah menjadi grayscale dan di lakukan integral2sum, sehingga thresholdingnya lebih akurat dan dinamis. Selanjutnya adalah fungsi final untuk menerapkan semua perubahan pada citra yang ada

private void button10\_Click(object sender, EventArgs e)

{

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

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

{

if (x < s / 2 + 1 || x > (gray.Width - s / 2 - 1) || y < s / 2 + 1 || y > (gray.Height - s / 2 - 1))

{

gray.SetPixel(x, y, Color.FromArgb(255, 255, 255));

}

else {

gray.SetPixel(x, y, Color.FromArgb(output[x, y], output[x, y], output[x, y]));

}

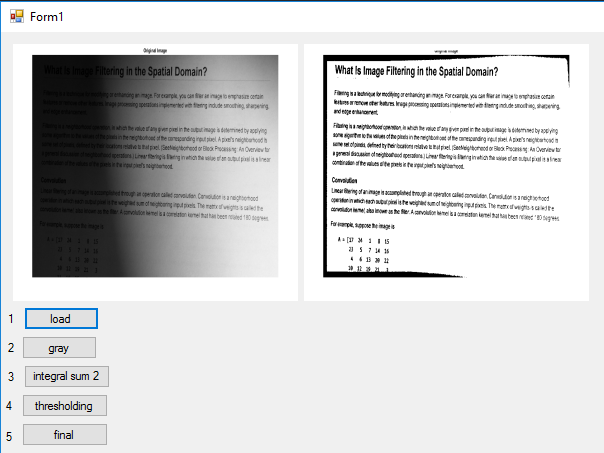
}

pictureBox2.Image = gray;

File = gray;

}

Button final digunakan untuk melakukan setting perubahan pada setiap langkahnya untuk penerpaan adaptive thresholding pada sebuah citra, hasil dari adaptive thresholding ini dapat dilihat pada *gambar 3.1* di bawah ini.



*Gambar 3.1 Hasil dari local adaptive threshold*

* ***Analisa***

Thresholding merupakan proses modifikasi citra berderajat keabuan menjadi citra biner atau citra hitam putih, sehingga dapat diketahui daearah mana yang termasuk objek dan mana yang termasuk background. Citra hasil thresholding biasa digunkan untuk mengenali suatu objek ataupun digunakan untuk ekstraksi fiktur.

Adaptive thresholding sendiri merupakan pengembangan dari local thresholding yang mana algoritmanya dapat mendeteksi objek meskipun tingkat keterangan atau iluminasi sebuah citranya tidak sama rata (nonuniform). Adaptive thresholding sangat berguna jika ingin mengenali suatu objek dengan keadaan pencahayaan yang kurang ataupun pencahayaan yang berubah ubah.