

Dasar Pengolahan Citra (3)

1. Tujuan:

1. Mahasiswa dapat membuat program untuk merubah citra warna RGB menjadi Gray-Scale
2. Mahasiswa dapat membuat program thresholding atau mengatur jumlah derajat keabuan yang ada pada citra

2. Dasar Teori:

2.1. Mengubah Citra Berwarna Menjadi Gray-Scale

Proses awal yang banyak dilakukan dalam *image processing* adalah mengubah citra berwarna menjadi citra gray-scale, hal ini digunakan untuk menyederhanakan model citra. Seperti telah dijelaskan di depan, citra berwarna terdiri dari 3 layer matrik yaitu R-layer, G-layer dan B-layer. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer matrik gray-scale dan hasilnya adalah citra gray-scale. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing r, g dan b menjadi citra gray scale dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai r, g dan b sehingga dapat dituliskan menjadi:

$$s = \frac{r + g + b}{3}$$

Untuk mencoba proses konversi citra berwarna menjadi citra gray-scale ini dapat dibuat program seperti gambar 1



Gambar 1. Contoh form untuk menangkap citra.

2.2. Thresholding

Thresholding digunakan untuk mengatur jumlah derajat keabuan yang ada pada citra. Dengan menggunakan thresholding maka derajat keabuan bisa diubah sesuai keinginan, misalkan diinginkan menggunakan derajat keabuan 16, maka tinggal membagi nilai derajat keabuan dengan 16. Proses thresholding ini pada dasarnya adalah proses pengubahan kuantisasi pada citra, sehingga untuk melakukan thresholding dengan derajat keabuan dapat digunakan rumus:

$$x = b.\text{int}\left(\frac{w}{b}\right)$$

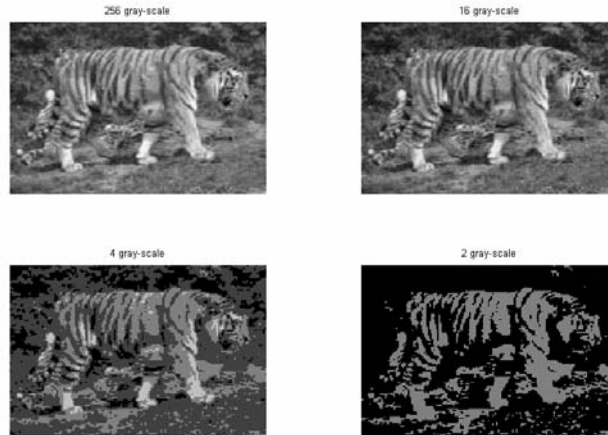
dimana :

w adalah nilai derajat keabuan sebelum thresholding

x adalah nilai derajat keabuan setelah thresholding

$$b = \text{int}\left(\frac{256}{a}\right)$$

Berikut ini contoh thresholding mulai di 256, 16, 4 dan 2.



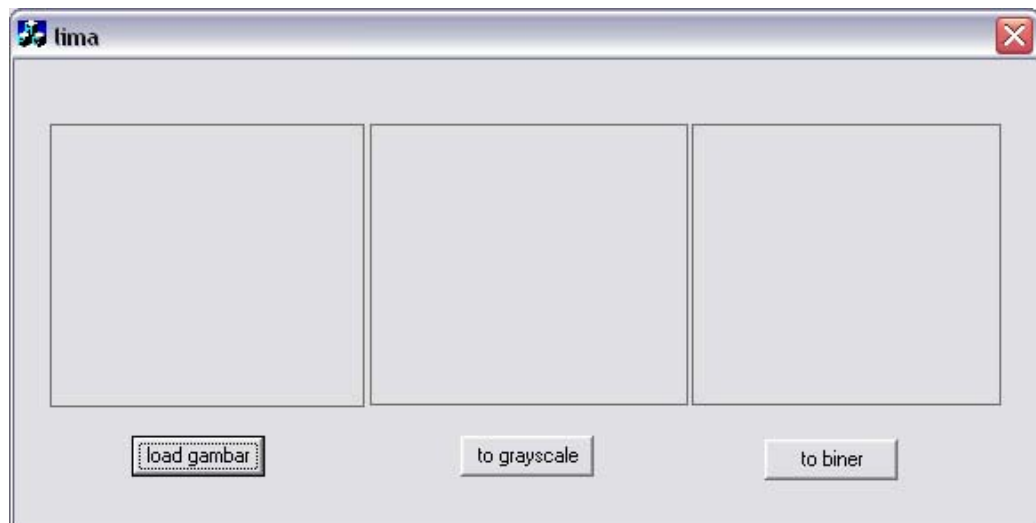
Gambar 3.3. Contoh thresholding

Untuk mencoba melakukan proses thresholding, perlu dibuat program untuk dapat mengubah-ubah nilai thresholding sesuai keinginan. Sehingga perlu ditampilkan dua citra, yaitu citra asli (gray-scale) dan hasil thresholdingnya dengan nilai thresholding yang ditentukan melalui input seperti terlihat pada gambar 3.3.

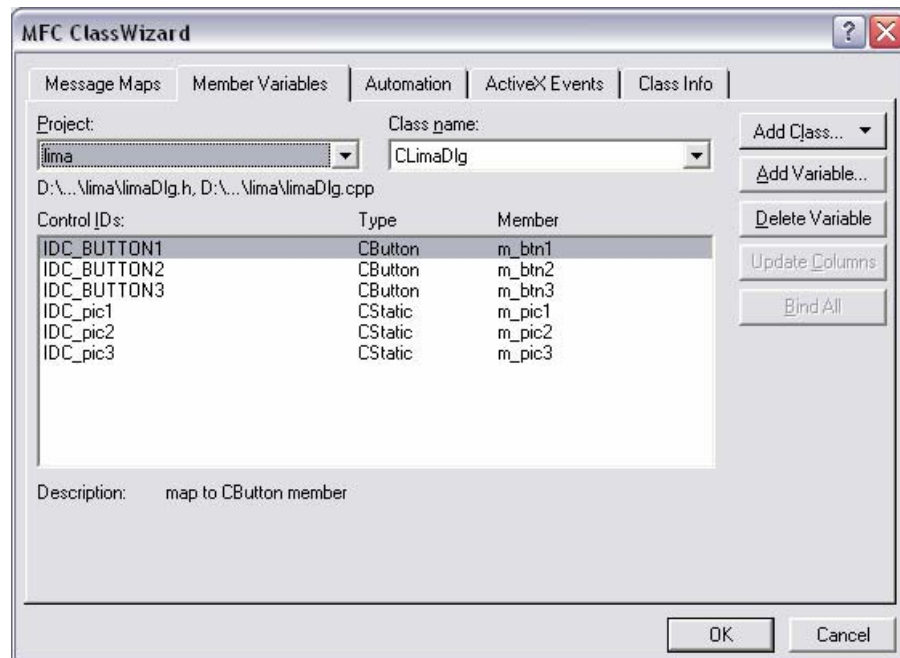
3. Percobaan:

3.1. Mengubah Citra Berwarna Menjadi Gray-Scale

1. Cara mengubah citra warna menjadi gray-scale
 - Buat aplikasi AppWizard seperti pada praktikum 1
 - Buat Menu dengan desain seperti berikut:



Member variabel



Isikan program pada button load gambar

```
void CLimaDlg::OnButton1()
{
    CDC* pDC = m_pic1.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
    "satu.bmp",IMAGE_BITMAP, 0, 0,
    LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic1.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
}
```

Isikan program pada button to graySclae

```

void CLimaDlg::OnButton2()
{
    int i,j,red,green,blue,gray;
    long int warna,warnagray;
    CDC* pDC = m_pic2.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
    "satu.bmp", IMAGE_BITMAP,
    0,
    0,
    LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic2.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {
            warna=dcMem1.GetPixel(j,i);
            WarnaToRGB(warna,&red,&green,&blue);
            gray=int(red+green+blue)/3;
            warnagray=RGBToWarna(gray,gray,gray);
            dcMem1.SetPixel(j,i,warnagray);
        }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
}

```

Isikan program pada button toBiner

```

void CLimaDlg::OnButton3()
{
    int i,j,red,green,blue,gray;
    long int warna,warnagray,ratagray;
    CDC* pDC = m_pic3.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
    "satu.bmp", IMAGE_BITMAP,
    0,
    0,
    LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
    }
}

```

```

        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic3.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {
            warna=dcMem1.GetPixel(j,i);
            WarnaToRGB(warna,&red,&green,&blue);
            gray=int(red+green+blue)/3;
            ratagray+=gray;
        }
        ratagray=ratagray/(bm.bmHeight*bm.bmWidth);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {
            warna=dcMem1.GetPixel(j,i);
            WarnaToRGB(warna,&red,&green,&blue);
            gray=int(red+green+blue)/3;
            if(gray<128) gray=0;
            else gray=255;
            warnagray=RGBToWarna(gray,gray,gray);
            dcMem1.SetPixel(j,i,warnagray);
        }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
}

```

Fungsi mengubah warna ke rgb

```

void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
    *Red = warna & 0x000000FF;
    *Green = (warna & 0x0000FF00) >> 8;
    *Blue = (warna & 0x00FF0000) >> 16;
}

```

Fungsi mengubah rgb ke warna

```

long int RGBToWarna(int Red, int Green, int Blue)
{
    return(Red+(Green<<8)+(Blue<<16));
}

```

Tambahan pada header file

```
public:  
    CBitmap m_bmpBitmap;
```

Penjelasan Program :

1. fungsi mengubah warna ke rgb

```
void WarnaToRGB(long int warna,int *Red, int *Green,  
int *Blue)  
{  
    *Red = warna & 0x000000FF;  
    *Green = (warna & 0x0000FF00) >> 8;  
    *Blue = (warna & 0x00FF0000) >> 16;  
}
```

Sebuah gambar akan diambil informasi mengenai 3 warna dasar tiap pixelnya, yaitu merah biru dan hijau, fungsi ini akan memecah gambar menjadi nilai-nilai warna dasarnya

2. fungsi mengubah rgb ke warna

```
long int RGBToWarna(int Red, int Green, int Blue)  
{  
    return(Red+(Green<<8)+(Blue<<16));  
}
```

Setiap pixel pada gambar akan diberikan campuran dari 3 warna dasar yang sebelumnya telah dipecah sehingga setiap pixel akan terdiri dari 3 warna dasar tergantung pada intensitasnya

3. merubah gambar ke grayscale

```
for(i=0;i<bm.bmHeight;i++)  
    for(j=0;j<bm.bmWidth;j++)  
    {  
        warna=dcMem1.GetPixel(j,i);  
        WarnaToRGB(warna,&red,&green,&blue);  
        gray=int(red+green+blue)/3;  
        warnagray=RGBToWarna(gray,gray,gray);  
        dcMem1.SetPixel(j,i,warnagray);  
    }
```

Pada perubahan sebuah gambar menjadi grayscale dapat dilakukan dengan cara mengambil semua pixel pada gambar kemudian warna tiap pixel akan diambil informasi mengenai 3 warna dasar yaitu merah, biru dan hijau (melalui fungsi warnatoRGB), ketiga warna dasar ini akan dijumlahkan kemudian dibagi tiga sehingga didapat nilai rata-rata. Nilai rata-rata inilah yang akan dipakai untuk memberikan warna pada pixel

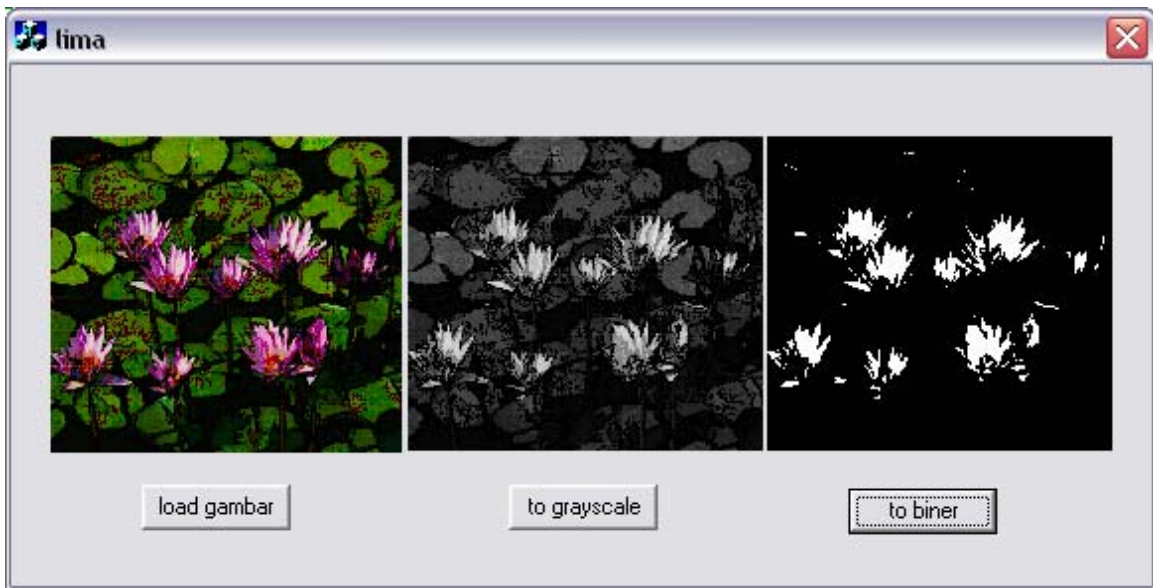
gambar sehingga warna menjadi grayscale, tiga warna dasar dari sebuah pixel akan diset menjadi nilai rata-rata (melalui fungsi RGBtoWarna)

4. merubah gambar ke biner

```
for(i=0;i<bm.bmHeight;i++)
    for(j=0;j<bm.bmWidth;j++)
    {
        warna=dcMem1.GetPixel(j,i);
        WarnaToRGB(warna,&red,&green,&blue);
        gray=int(red+green+blue)/3;
        if(gray<128) gray=0;
        else gray=255;
        warnagray=RGBToWarna(gray,gray,gray);
        dcMem1.SetPixel(j,i,warnagray);
    }
```

Mengubah gambar ke biner prosesnya hampir sama dengan mengubah gambar ke grayscale, bedanya warna rata-rata akan dikelompokkan menjadi dua, jika intensitas warna dimulai dari 0 sampai dengan 255 maka diambil nilai tengahnya yaitu 128, jika dibawah 128 maka warna akan cenderung hitam dan diatas 128 warna akan cenderung putih

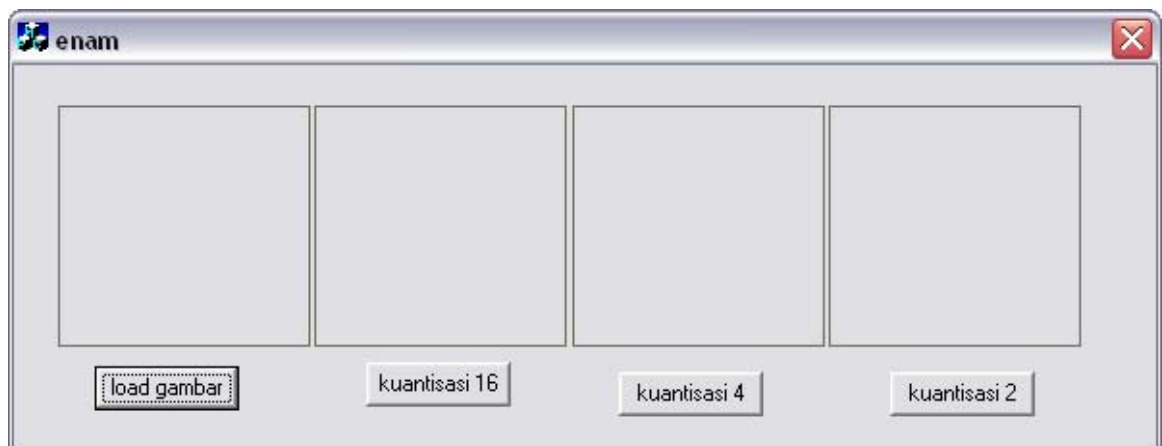
Menjalankan program



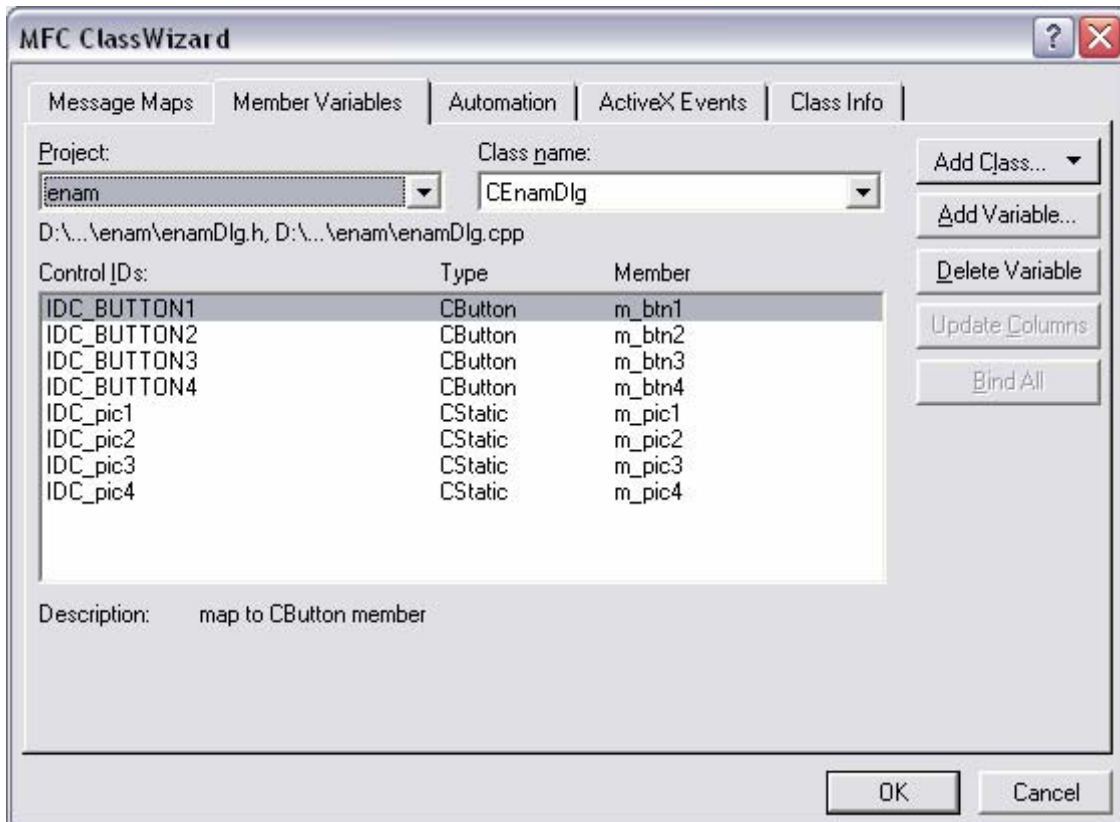
3.2. Thresholding

1. Cara Thresholding gambar

- Buat aplikasi AppWizard
- Buat Menu dengan desain sebagai berikut



Member variabel:



Isikan program pada button load gambar (grayscale)

```

void CEnamDlg::OnButton1()
{
    int i,j,red,green,blue,gray;
    long int warna,warnagray;
    CDC* pDC = m_pic1.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
    "pens.bmp",IMAGE_BITMAP, 0, 0,
    LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic1.GetClientRect(rect);//
    m_bmpBitmap.GetBitmap(&bm);//
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {

```

```

        warna=dcMem1.GetPixel(j,i);
        WarnaToRGB(warna,&red,&green,&blue);
        gray=int(red+green+blue)/3;
        warnagray=RGBToWarna(gray,gray,gray);
        dcMem1.SetPixel(j,i,warnagray);
    }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);//
}

```

Isikan program pada button kuantisasi 16

```

void CEnamDlg::OnButton2()
{
    int i,j,red,green,blue,th;
    long int warna,wgray,xgray;
    CDC* pDC = m_pic2.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
    "pens.bmp",IMAGE_BITMAP, 0, 0,
    LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic2.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    th= int (256/16);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {
            warna=dcMem1.GetPixel(j,i);
            WarnaToRGB(warna,&red,&green,&blue);
            wgray=(red+green+blue)/3;
            xgray=th*int (wgray/th);
            warna=RGBToWarna(xgray,xgray,xgray);
            dcMem1.SetPixel(j,i,warna);
        }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
}

```

Isikan program pada button kuantisasi 4

```

void CEnamDlg::OnButton3()
{
    int i,j,red,green,blue,th;
    long int warna,wgray,xgray;
    CDC* pDC = m_pic3.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
"pens.bmp",IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {
        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic3.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    th= int (256/4);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
        {
            warna=dcMem1.GetPixel(j,i);
            WarnaToRGB(warna,&red,&green,&blue);
            wgray=(red+green+blue)/3;
            xgray=th*int (wgray/th);
            warna=RGBToWarna(xgray,xgray,xgray);
            dcMem1.SetPixel(j,i,warna);
        }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);//
}

```

Isikan program pada button kuantisasi 2

```

void CEnamDlg::OnButton4()
{
    int i,j,red,green,blue,th;
    long int warna,wgray,xgray;
    CDC* pDC = m_pic4.GetDC();
    CDC dcMem1;
    CRect rect;
    BITMAP bm;
    HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
"pens.bmp",IMAGE_BITMAP, 0, 0,
LR_LOADFROMFILE|LR_CREATEDIBSECTION);
    if(hBitmap)
    {

```

```

        if(m_bmpBitmap.DeleteObject())
            m_bmpBitmap.Detach();
        m_bmpBitmap.Attach(hBitmap);
    }
    m_pic4.GetClientRect(rect);
    m_bmpBitmap.GetBitmap(&bm);
    dcMem1.CreateCompatibleDC(pDC);
    dcMem1.SelectObject(&m_bmpBitmap);
    th= int (256/2);
    for(i=0;i<bm.bmHeight;i++)
        for(j=0;j<bm.bmWidth;j++)
            {
                warna=dcMem1.GetPixel(j,i);
                WarnaToRGB(warna,&red,&green,&blue);
                wgray=(red+green+blue)/3;
                xgray=th*int (wgray/th);
                warna=RGBToWarna(xgray,xgray,xgray);
                dcMem1.SetPixel(j,i,warna);
            }
    pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
    0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);
}

```

Fungsi mengubah warna ke rgb

```

void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
    *Red = warna & 0x000000FF;
    *Green = (warna & 0x0000FF00) >> 8;
    *Blue = (warna & 0x00FF0000) >> 16;
}

```

Fungsi mengubah rgb ke warna

```

long int RGBToWarna(int Red, int Green, int Blue)
{
    return(Red+(Green<<8)+(Blue<<16));
}

```

Tambahan pada header file

```

public:
    Cbitmap m_bmpBitmap;

```

Penjelasan program :

```
th= int (256/x); // x = kuantisasi, nilai derajat keabuan, 16, 4, 2
for(i=0;i<bm.bmHeight;i++)
    for(j=0;j<bm.bmWidth;j++)
    {
        warna=dcMem1.GetPixel(j,i);
        WarnaToRGB(warna,&red,&green,&blue);
        wgray=(red+green+blue)/3;
        xgray=th*int (wgray/th);
        warna=RGBToWarna(xgray,xgray,xgray);
        dcMem1.SetPixel(j,i,warna);
    }
```

Proses kuantisasi hampir sama dengan grayscale, bedanya warna pada setiap pixel yang telah dirata-rata akan dikalikan dengan nilai derajat keabuan (th), sehingga terjadi pengelompokan warna mejadi beberapa kelompok sesuai dengan nilai kuantisasinya

Menjalankan program:



3.5. Latihan:

1. Sebutkan proses utama pada proses konversi citra berwarna menjadi citra gray-scale?

Apa perbedaan antara pemakaian rumus rata-rata $x = (r + g + b) / 3$ dan rumus RGB optimal $x = 0.42r + 0.32g + 0.28b$.

2. Ubahlah program konversi citra berwarna menjadi citra gray scale di atas dengan mengubah nilai gray scale dengan

$$x = 0.5r + 0.2g + 0.3b$$

$$x = 0.2r + 0.5g + 0.3b$$

$$x = 0.2r + 0.2g + 0.5b$$

$$x = 0.5r + 0.5g + 0b$$

$$x = 0.5r + 0g + 0.5b$$

Perhatikan bagaimana perbedaan hasil konversi dengan tiga macam rumus di atas.

3. Jelaskan apa pengertian dari thresholding, dan bagaimana prosesnya ?
Apa hubungan thresholding dengan kuantisasi pada citra?
4. Jelaskan apakah konversi citra ke citra biner dan proses thresholding dengan nilai thresholding 2 menghasilkan citra yang sama ?