Praktikum

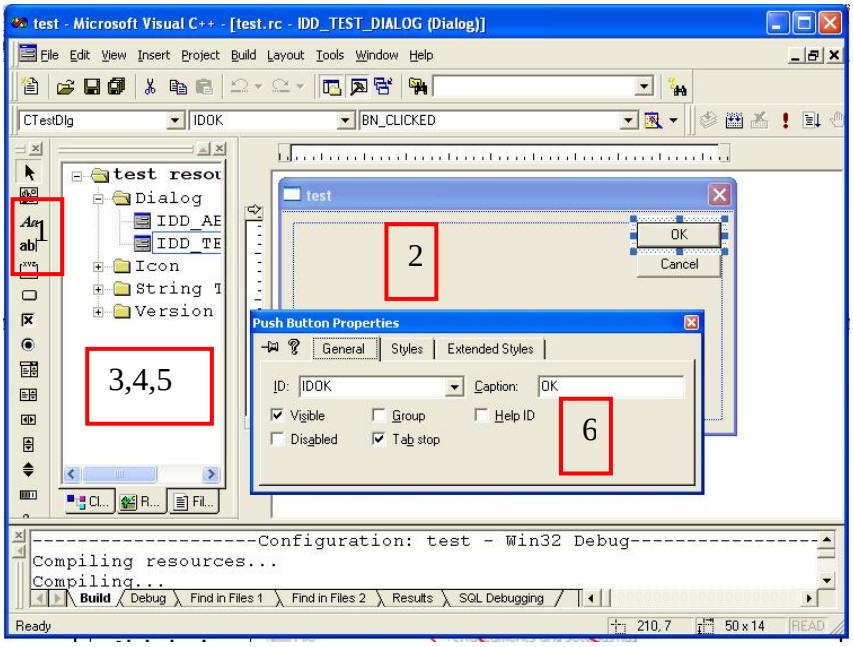
# 1 Pendahuluan

## 1.1. Tujuan:

1. Mahasiswa dapat membuat program pengolahan citra menggunakan Visual C++ dengan MFC.
2. Mahasiswa dapat membuat dialog menggunakan Visual C++ dengan MFC.

## 1.2. Dasar Teori:

IDE (Integrated Development Environtment) Visual C++ 6 dapat dilihat pada gambar 1 . Dengan macam-macam komponen di dalamnya dapat dilihat pada tabel 1.




Gambar 1.1. IDE Visual C++ 6

Macam-macam komponen yang terdapat pada IDE visual C++ 6 sbb :

Tabel 1.1.

Macam-Macam Komponen IDE VC++ 6

| No | IDE Komponen | Deskripsi |
|---|---|---|
| 1 | Toolbox | Bermacam-macam control yang dapat digunakan di project Visual C++ (pada prak Citra yang biasa dipakai : picture, textbox, button,slider dsb). |
| 2 | Dialog Editor (MFC) | Pada area ini tempat membuat atau mengedit tampilan dialog box bukan form window. |
| 3 | Solution Explorer | Menampilkan organisasi project yang dibuat |
| 4 | Class View | Menampilkan simbol code project seperti : namespaces, classes, methods, dan functions |
| 5 | Resource View | Menampilkan resource file dari project |
| 6 | Properties Window | Window yang dapat dipakai untuk mengedit atau merubah sifat control pada waktu disain |

Image processing atau sering disebut dengan pengolahan citra digital merupakan suatu proses dari gambar asli menjadi gambar lain yang sesuai dengan keinginan kita. Misal suatu gambar yang kita dapatkan terlalu gelap maka dengan image processing gambar tersebut bisa kita proses sehingga mendapat gambar yang jelas. Secara garis besar bisa kita gambarkan seperti blok diagram pada gambar 1.2 dibawah ini:


Gambar 1.2 Blok Diagram Pengolahan Citra

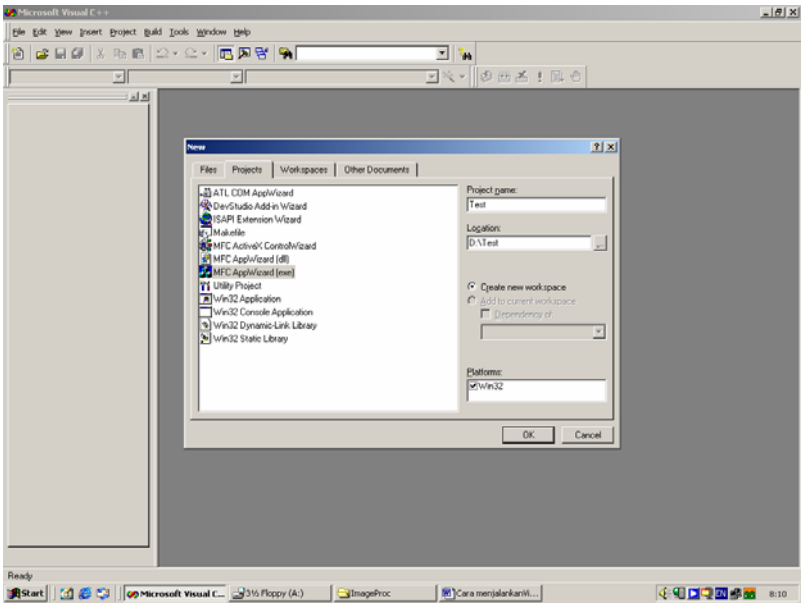
## 1.3. Tugas Pendahuluan:

1. Tuliskan tujuan praktikum
2. Gambarkan blok diagram pengolahan citra
3. Buatkan ringkasan cara menjalankan Visual C++ dengan MFC
4. Buatkan ringkasan cara membuat dialog

## 1.4. Percobaan:

### 1.4.1. Menjalankan Visual C++ dengan MFC

1. Membuka Visual C++ 6.0
   - Pilih menu : Start->Programs->Microsoft Visual Studio 6.0->Microsoft Visual C++ 6.0
2. Memberi nama program
   - Pilih menu : File->New->Projects->MFC AppWizard(exe)
   - Isi Project name misalnya dengan: **Test** (lihat gambar 1.3)
   - Tekan tombol OK




Gambar 1.3 Memberi nama program

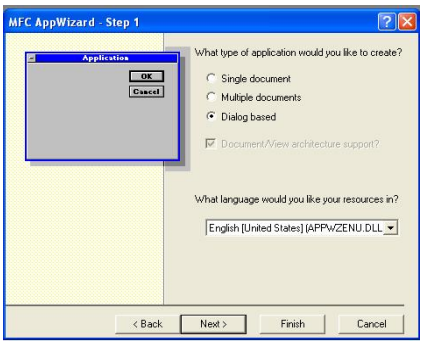
3. Memilih isi program

- Step 1 : What tipe of application would you like to create
  (aplikasi apa yang ingin dibuat) pilih *dialog based* untuk membuat aplikasi memakai dialog. Perhatikan gambar 1.4. a dialog step 1. Tekan tombol Next.
- Step 2 :
  1. What features would you like to include?
     (fitur apa saja yang akan dimasukkan) pilih *About box* dan *3D controls* untuk pilihan standar.
  2. What features would you like to include?
     (mendukung aplikasi apa ?) pilih *ActiveX controls* untuk pilihan standar.
  3. Would you like to include WOSA support ? *Window Sockets* tidak dipilih untuk pilihan standard.
  4. Please enter a title for your dialog ? Title sudah berisi text sama dengan nama project untuk pilihan standard

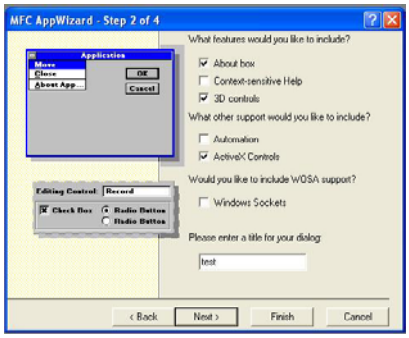
  Perhatikan gambar 1.4.b. dialog step 2. Tekan tombol Next.
- Step 3 :
  1. What style of project would you like ?
     Pilihan standard dan yang aktif hanya *MFC Standard.*
  2. Would you like to generate source file comments ? pilihan standard Yes, Please
  3. How would you like to use the MFC Library ? pilihan standard As a shared DLL

  Perhatikan gambar 1.4.c. dialog step 3. Tekan tombol Next
- Step 4 : AppWizard creates the following classes for you
  (AppWizard akan membuat class seperti dibawah ini)

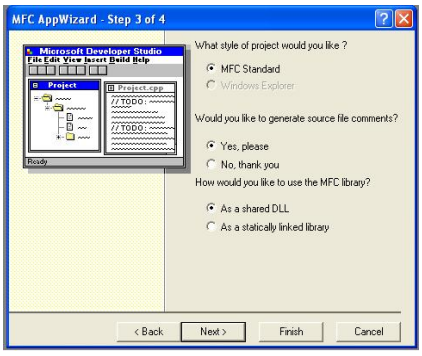
  CtestApp
  CtestDlg
  Perhatikan gambar 1.4.d. dialog step 4.



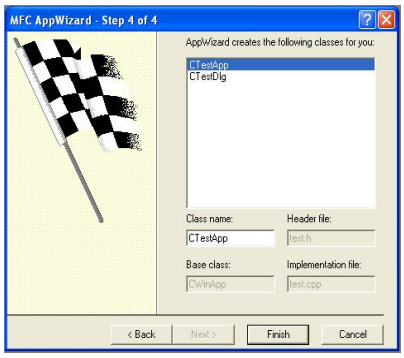

(a)




(b)



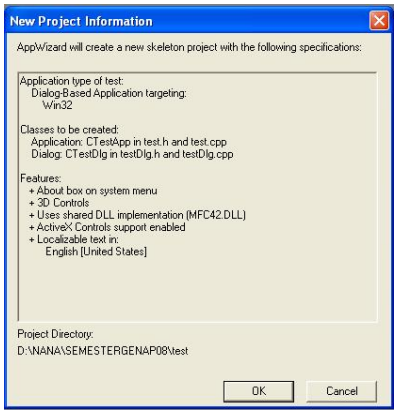

(c)




(d)

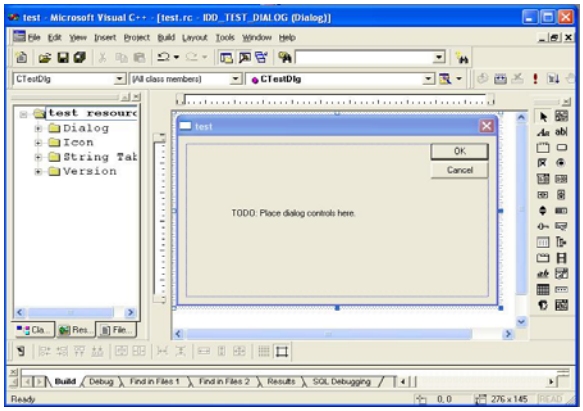
Gambar 1.4. Memilih Isi Program
(a) Dialog Step 1 (b) Dialog Step 2 (c) Dialog Step 3 (d) Dialog Step 4

- Tekan tombol Finish akan muncul New Project Information. Perhatikan gambar 1.5.



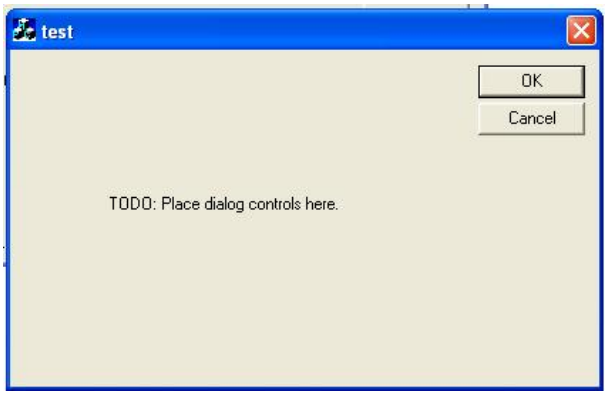

Gambar 1.5. New Project Information

- Kemudian tekan tombol OK. Tampilan awal dialog akan dihasilkan perhatikan gambar 1.6.




Gambar 1.6. Awal Dialog

4. Cara menjalankan program
   - Pilih menu : Build->Execute (!), perhatikan gambar 1.7. Dialog Hasil Execute
   - Tekan tombol Yes, untuk menutup aplikasi.



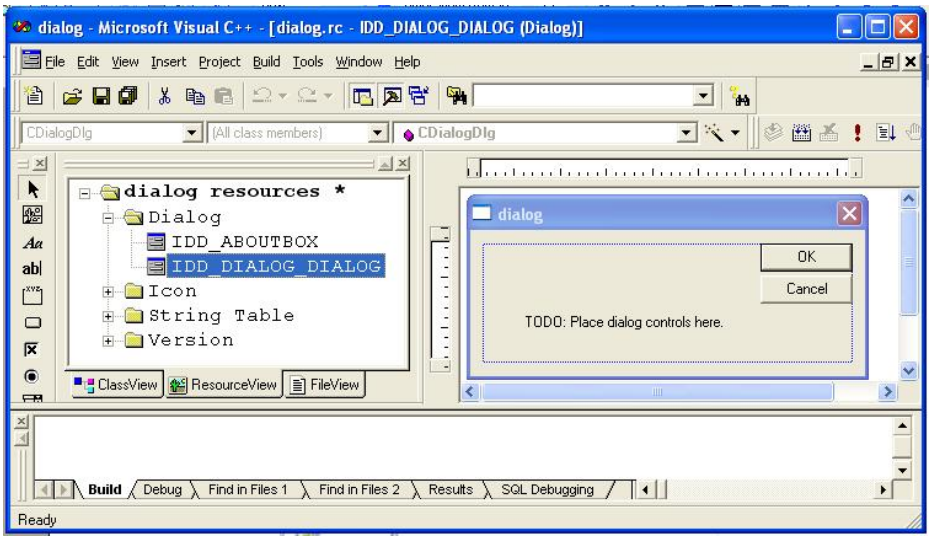

Gambar 1.7. Dialog Hasil Execute

### 1.4.2. Cara Mendisain Dialog

*Catatan* : Jika Dialog Editor tidak tampak buka pada Resource View (Ctrl-Shift-E) double click pada *IDD_TEST_DIALOG*.

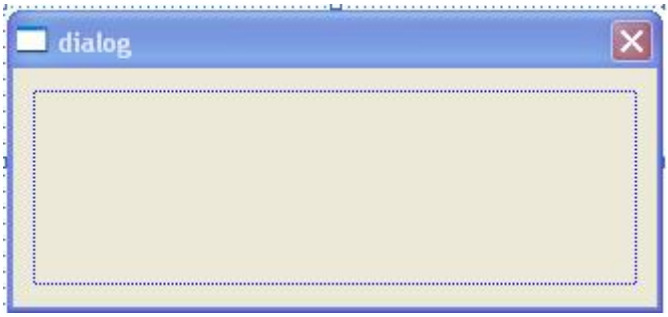
1. Cara membuat dialog
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Dialog
   - Pilih ResourceView pada workspace
   - Pilih folder paling atas dengan cara klik pada tanda +
   - Pilih folder dialog dengan cara klik pada tanda +

- Klik 2 kali IDD_DIALOG_DIALOG seperti gambar 1.4

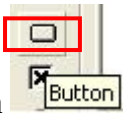



Gambar 1.8 Membuat Dialog

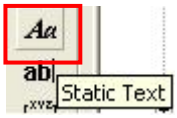
2. Menghapus dan menambah control pada dialog
   - Untuk menghapus control : aktifkan control dengan cara klik pada area control lanjutkan dengan menekan tombol delete. Cobalah untuk menghapus control : text TODO, button OK dan Cancel. Hasil akhir pada gambar 1.9.




Gambar 1.9. Dialog dengan Semua Control Dihapus

   - Untuk menambah control: (button) pilih icon 

control pada toolbox drag drop ke editor dialog.

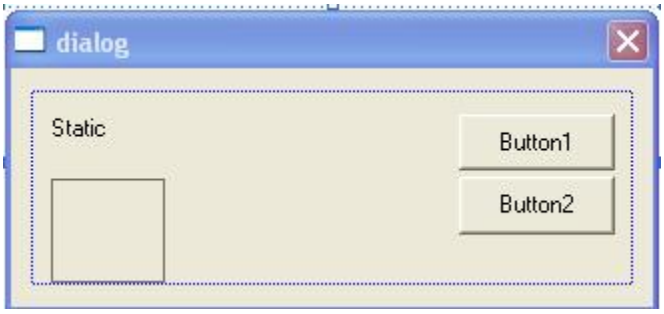
   (static text) pilih icon 

control pada toolbox drag drop ke editor dialog.

(picture box) pilih icon 

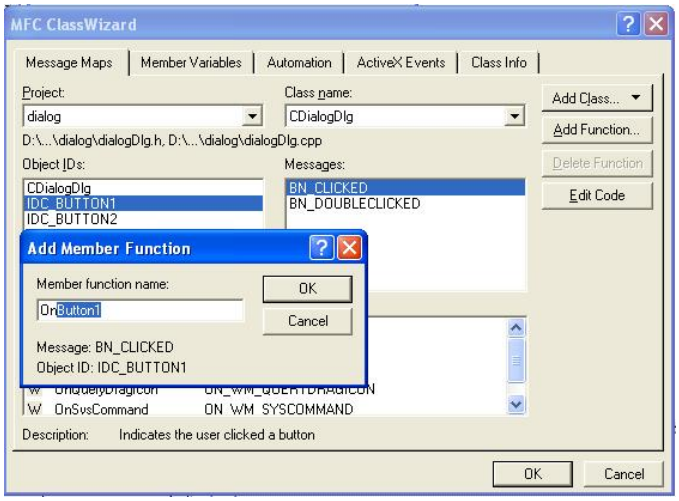
control pada toolbox drag drop ke editor dialog.

- Tambahkan control (dua button, 1 static text, 1 picture) pada dialog editor seperti hasil pada gambar 1.10.



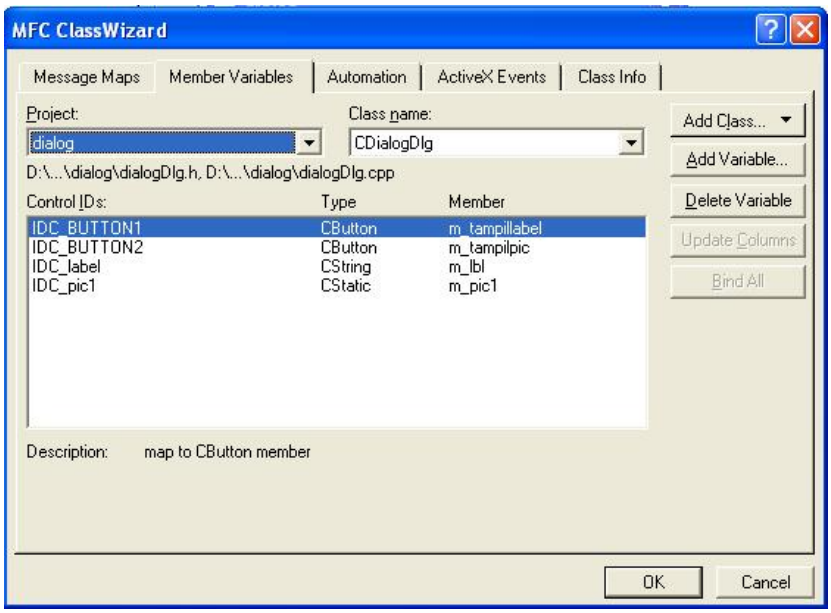

Gambar 1.10 Disain Dialog

3. Mengaktifkan control
   - Pilih control button1, double clik, pada dialog Add Member Function klik OK.
   - Atau klik kanan, pilih klik ClassWizard, pada MFC Class Wizard Dialog Messages aktifkan BN_CLICKED, double clik, pada dialog Add Member Function klik OK. Gambar 1.11.




Gambar 1.11. Mengaktifkan Control Button1

4. Rubah terlebih dahulu ID dari static text = ID_label dan ID dari picture = ID_pic1, dengan cara aktifkan control dan klik kanan pilih properti.
5. Pada dialog MFC classwizard,Member Variabels edit type dan member dari control sebagai berikut gambar 1.12.




Gambar 1.12. MFC ClassWizard pada Member Variabel

6. Menulis fungsi pada button1 clik
   - Tambahkan program untuk menampilkan pesan pada static text seperti dibawah ini

```
void CDialogDlg::OnButton1()
{
// TODO: Add your control notification handler code here
SetDlgItemText(IDC_label,"Belajar Prak Citra dgn VC++ dan MFC");
}
```

   Jalankan program dengan memilih menu Build->Execute (!)

7. Tambahkan deklarasi kelas CBitmap pada File View – Header Files – dialogDlg.h

```
// Construction
public:
CDialogDlg(CWnd* pParent = NULL);        // standard constructor
CBitmap m_bmpBitmap;
//
```

8. Menulis fungsi pada button2 clik

```
void CDialogDlg::OnButton2()
{
// TODO: Add your control notification handler code here
      CDC* pDC = m_pic1.GetDC();//
      CDC dcMem1;
      CRect rect;//kotak di picture
      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);
  pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem1,
        0,0,bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

Jalankan program dengan memilih menu Build->Execute (!)

Hasil akhir program dijalankan pada gambar 1.12.

Gambar 1.13. Hasil Execute Prak 1.

## 1.5. Latihan:

1. Buat program untuk menampilkan Tulisan “Selamat Belajar Pengolahan Citra” dengan menggunakan MessageBox bila submenu yang dipilih.
2. Buatlah program untuk menampilkan Tulisan “Ini adalah Teks” pada fungsi OnDraw(CDC* pDC) dengan menggunakan pDC->TextOut(10,10,"Ini adalah Teks");
3. Buatlah program untuk mengambar sumbu x dan sumbu y pada fungsi OnDraw(CDC* pDC) dengan menggunakan pDC->MoveTo(10,10) dan pDC->LineTo(10,100)
4. Buatlah program untuk menampilkan persamaan linear y=x pada fungsi OnDraw(CDC* pDC) dimana nilai x dari 0 sampai 100
5. Buatlah program untuk menampilkan persamaan kuadrat $y=x^2$ dimana nilai x dari 0 sampai 100 bila submenu yang dibuat dipilih.

## 1.6. Laporan Resmi:

Buatlah laporan resmi dari latihan-latihan diatas dengan cara membuat analisa dan kesimpulan.

Praktikum

2

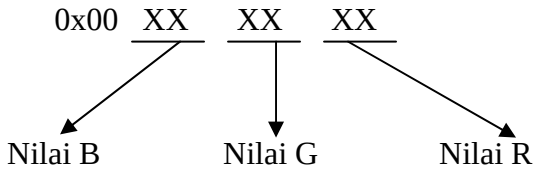
# Dasar Pengolahan Citra (1)

## 2.1. Tujuan:

1. Mahasiswa dapat membuat program untuk menampilkan gambar
2. Mahasiswa dapat membuat program untuk memproses gambar dengan mengambil warna RGB
3. Mahasiswa dapat membuat program untuk memproses gambar dengan meletakkan warna RGB pada lokasi x dan y

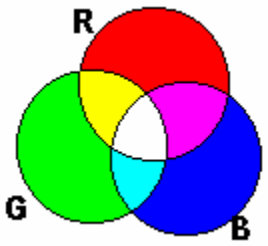
## 2.2. Dasar Teori:

Dasar dari pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Dalam pengolahan citra warna dipresentasikan dengan nilai hexadesimal dari 0x00000000 sampai 0x00ffffff. Warna hitam adalah 0x00000000 dan warna putih adalah 0x00ffffff. Definisi nilai warna di atas seperti gambar 2.1, variabel 0x00 menyatakan angka dibelakangnya adalah hexadecimal.




Gambar 2.1 Nilai warna RGB dalam hexadesimal

Terlihat bahwa setiap warna mempunyai range nilai 00 (angka desimalnya adalah 0) dan ff (angka desimalnya adalah 255), atau mempunyai nilai derajat keabuan 256 = $2^8$. Dengan demikian range warna yang digunakan adalah $(2^8)(2^8)(2^8) = 2^{24}$ (atau yang dikenal dengan istilah True Colour pada Windows). Nilai warna yang digunakan di atas merupakan gambungan warna cahaya merah, hijau dan biru seperti yang terlihat pada gambar 2.2. Sehingga untuk menentukan nilai dari suatu warna yang bukan warna dasar digunakan gabungan skala kecerahan dari setiap warnanya.




Gambar 2.2 Komposisi warna RGB

Dari definisi diatas untuk menyajikan warna tertentu dapat dengan mudah dilakukan, yaitu dengan mencampurkan ketiga warna dasar RGB, table 1. berikut memperlihatkan contoh-contoh warna yang bisa digunakan

Tabel 1. Contoh-contoh warna dalam hexadesimal

| Nilai | Warna |
|---|---|
| 0x00000000 | Hitam |
| 0x000000FF | Merah |
| 0x0000FF00 | Hijau |
| 0x00FF0000 | Biru |
| 0x0000FFFF | Kuning |
| 0x00FF00FF | Magenta |
| 0x00FFFF00 | Cyan |
| 0x00FFFFFF | Putih |

| nilai | Warna |
|---|---|
| 0x0000AAFF | Orange |
| 0x00888888 | Abu-abu |
| 0x00FF00AA | Ungu |
| 0x00AAFF00 | Hijau Muda |
| 0x00AA00FF | Merah Muda |
| 0x00AAFFFF | Kuning Muda |
| 0x000088AA | Coklat |
| 0x00AA0088 | Ungu |

Untuk mengetahui kombinasi warna, perlu dibuat suatu program yang dapat menampilkan warna sesuai dengan nilai yang dimasukkan sehingga dapat dicoba berbagai macam kombinasi warna RGB seperti gambar 2.2.

## 2.3. Tugas Pendahuluan:

1. Tuliskan tujuan praktikum
2. Jelaskan nilai warna RGB dalam hexadesimal
3. Sebutkan tiga komposisi warna dasar
4. Buatkan ringkasan mengenai class CfileDialog, CBitmap, CDC dan metoda setPixel dan getPixel di MSDN

## 2.4. Percobaan:

### 2.4.1. Menampilkan File Gambar

1. Cara membuka file
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan OpenFile
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test dan submenunya OpenFile

- Untuk mengedit isi program tekan tombol Edit Code atau buka file OpeFileView.cpp
- Tambahkan program untuk membuka file seperti dibawah ini

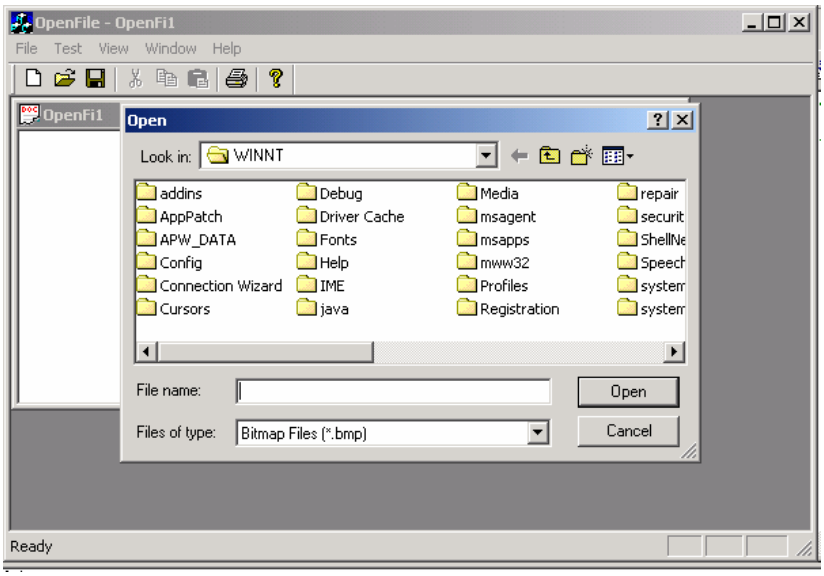
```
/////////////////////////////////////////////////////////////////////////////
// COpenFileView message handlers
// Menampilkan file yang akan dibuka
void COpenFileView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void COpenFileView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file
   - Buka file OpenFileView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
	COpenFileDoc* GetDocument();
	CString name;
	CBitmap m_bmpBitmap;
// Operations
public:
```

```
void LoadGambar(void);
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile ->pilih salah satu gambar misalnya gambar.bmp
   - Hasilnya seperti gambar 2.3




Gambar 2.3 Membuka file gambar

### 2.4.2. Cara Memproses Gambar

1. Cara memproses gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Proses
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan Proses

- Untuk mengedit isi program tekan tombol Edit Code atau buka file ProsesView.cpp
- Tambahkan program untuk memproses gambar seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////////
// CProsesView message handlers

void CProsesView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CProsesView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
void CProsesView::OnTestProses()
{
	// TODO: Add your command handler code here
	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j;
	long int warna;
	char str[5];
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	for(i=0;i<210;i++)
		for(j=0;j<250;j++)
		{
			// memberi warna pada titik
			dcMem.SetPixel(j,i,0x000000ff);
```

```
            }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
        // membaca warna pada titik
        warna=dcMem.GetPixel(5,5);
        sprintf(str,"%ld",warna);
        pDC->TextOut(10,10,str);
}
```

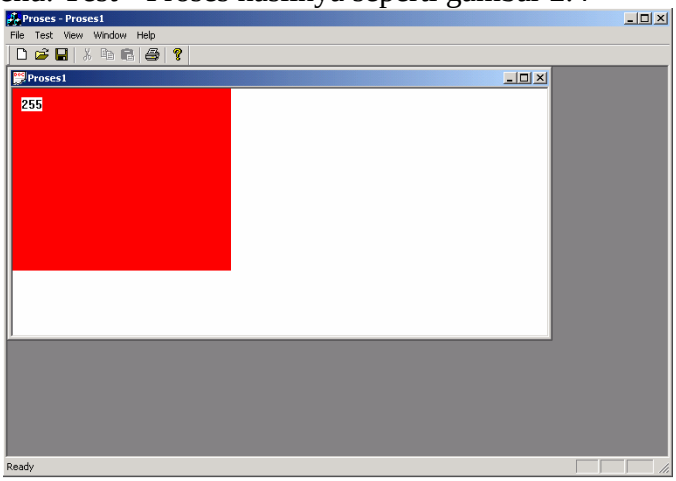
2. Menambah header file

   - Buka file ProsesView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CProsesDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;
// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program

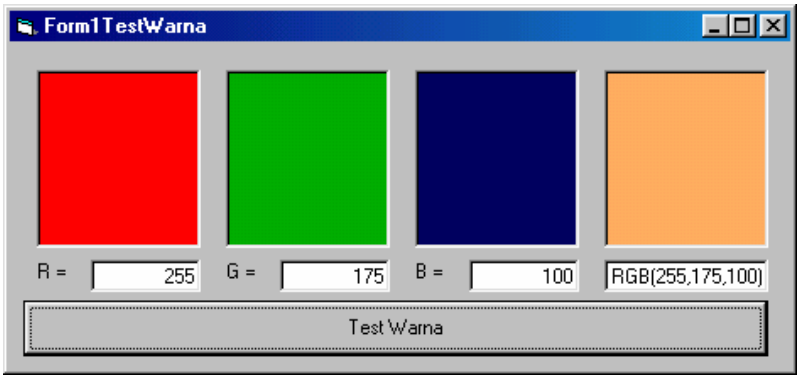
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Proses hasilnya seperti gambar 2.4




Gambar 2.4 Memproses gambar

## 2.5. Latihan:

1. Buatlah program untuk mengubah warna 3 buah picture-box dengan tiga macam nilai RGB yang masing-masing bernilai 0-255, sedangkan picture box yang keempat bernilai RGB yang merupakan kombinasi nilai-nilai R, G dan B seperti gambar berikut ini:



Gambar 2.5. Contoh program test warna

Cobalah mengisi dengan kombinasi berikut ini, apakah hasil dari kombinasi warna RGB berikut ini:

(a) R=0, G=255, B=128

(b) R=128, G=128, B=50

(c) R=100, G=100, B=255

2. Buatlah program untuk melakukan perputaran citra yang setiap baris pada picture-box 1 menjadi kolom pada picture-box 2 dan setiap kolom pada picture box 1 menjadi baris pada picture box 2. Tampilan formnya sebagai berikut.

Gambar 2.6. Pembalikan gambar

## 2.6. Laporan Resmi:

Buatlah laporan resmi dari latihan-latihan diatas dengan cara membuat analisa dan kesimpulan.

Praktikum

3

# Dasar Pengolahan Citra (2)

## 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

## 3.2. Dasar Teori:

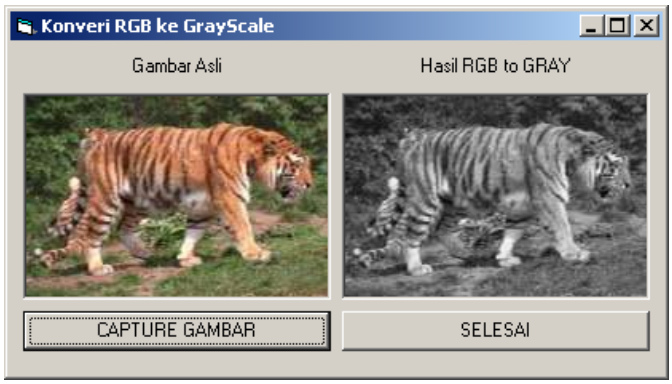
### 3.2.1. Mengubah Citra Berwarna Menjadi Gray-Scale

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 3.1



Gambar 3.1. Contoh form untuk menangkap citra.

### 3.2.2. 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:

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

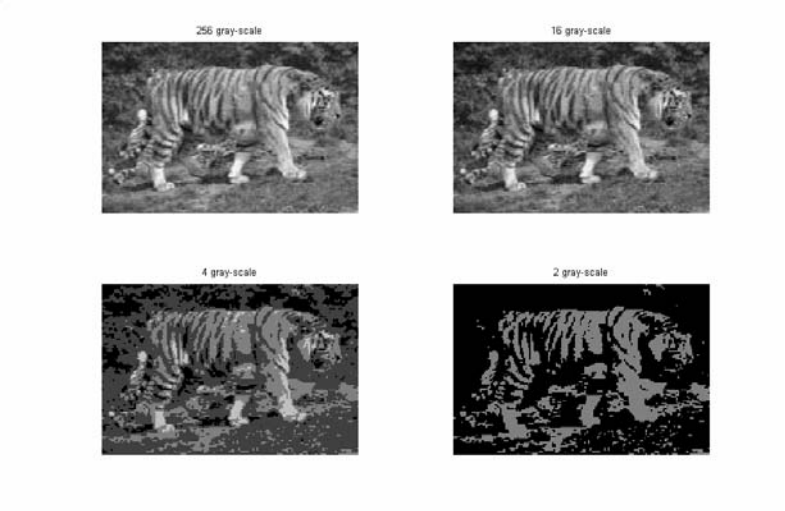
dimana :

w adalah nilai derajat keabuan sebelum thresholding

x adalah nilai derajat keabuan setelah thresholding

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

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 tresholding 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.

## 3.3. Tugas Pendahuluan:

1. Tuliskan tujuan praktikum
2. Jelaskan cara merubah citra berwarna menjadi Gray-Scale
3. Jelaskan cara mengatur jumlah derajat keabuan pada citra dengan Thresholding

## 3.4. Percobaan:

### 3.4.1. Mengubah Citra Berwarna Menjadi Gray-Scale

1. Cara mengubah citra warna menjadi gray-scale
    - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan GrayScale
    - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan GrayScale
    - Untuk mengedit isi program tekan tombol Edit Code atau buka file GrayScaleView.cpp

- Tambahkan program untuk mengubah citra warna menjadi gray-scale seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////////
// CGrayScaleView message handlers

void CGrayScaleView::OnTestLoadgambar()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";
	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CGrayScaleView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CGrayScaleView::OnTestGrayscale()
{
	// TODO: Add your command handler code here
	long int warna;
```

```
        int j,k,red,green,blue,gray;
        CDC* pDC = GetDC();
        CDC dcMem;
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);
        for(j=0;j<210;j++)
                for(k=0;k<250;k++)
                {
                        warna=dcMem.GetPixel(k,j);
                        // merubah data pixel ke RGB
                        WarnaToRGB(warna,&red,&green,&blue);

                        // mengubah warna menjadi Gray-Scale
                        gray=(red+green+blue)/3;

                        //merubah RGB ke data pixel
                        warna=RGBToWarna(gray,gray,gray);
                        dcMem.SetPixel(k,j,warna);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

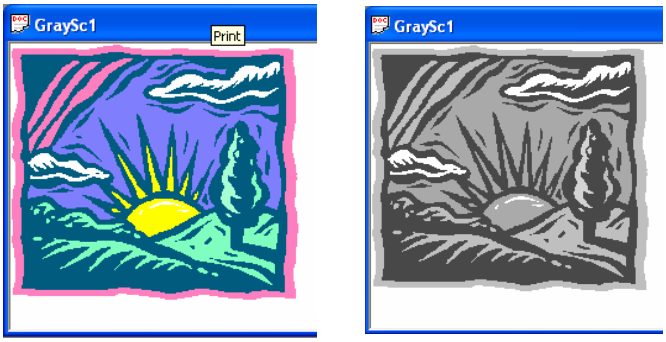
2. Menambah header file

   - Buka file GrayScaleView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CGrayScaleDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;
// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program

   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->GrayScale hasilnya seperti gambar 3.4




Gambar 3.4 Mengubah Citra Berwarna Menjadi Gray-Scale

### 3.4.2. Thresholding

1. Cara Treesholding gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan TreesHold
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan Treesholding
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file TreesHoldView.cpp
   - Tambahkan program untuk Treesholding seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////////
// CTreesholdView message handlers

void CTreesholdView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CTreesholdView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
				m_bmpBitmap.Detach();
			m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}


#include<math.h>
```

```
// merubah data pixel ke RGB
void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
	int x,i;
	x=1;
	*Red=0;
	for(i=0;i<8;i++)
	{
		if((warna&x)!=0)
			*Red=*Red+pow(2,i);
		x=x<<1;
	}
	*Green=0;
	for(i=0;i<8;i++)
	{
		if((warna&x)!=0)*Green=*Green+pow(2,i);
		x=x<<1;
	}
	*Blue=0;
	for(i=0;i<8;i++)
	{
		if((warna&x)!=0)*Blue=*Blue+pow(2,i);
		x=x<<1;
	}
}

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	long int Warna;
	int x,i;
	x=1;
	Warna=0;
	for(i=0;i<8;i++)
	{
		if((Red&x)!=0)
			Warna=Warna+pow(2,i);
		x=x<<1;
	}
	x=1;
	for(i=8;i<16;i++)
	{
		if((Green&x)!=0)
			Warna=Warna+pow(2,i);
		x=x<<1;
	}
	x=1;
	for(i=16;i<24;i++)
	{
		if((Blue&x)!=0)
			Warna=Warna+pow(2,i);
		x=x<<1;
	}
	return(Warna);
}
```

```
void CTreesholdView::OnTestTreesholding()
{
	// TODO: Add your command handler code here
	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j,r,g,b,OutMax=100,OutMin=0,res=10;
	long int w;
	double InMin,InMax,resultr,resultg,resultb;
	double InMinr,InMing,InMinb,InMaxr,InMaxg,InMaxb;
	double scaler,scaleg,scaleb,color;
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);

	InMin=dcMem.GetPixel(0,0);
	WarnaToRGB(InMin,&r,&g,&b);
	InMinr=r;
	InMing=g;
	InMinb=b;
	InMax=dcMem.GetPixel(0,0);
	WarnaToRGB(InMax,&r,&g,&b);
	InMaxr=r;
	InMaxg=g;
	InMaxb=b;
	for(i=0;i<200;i++)
		for(j=0;j<200;j++)
		{
			w=dcMem.GetPixel(j,i);
			WarnaToRGB(w,&r,&g,&b);
			resultr=r;
			resultg=g;
			resultb=b;
			if(InMinr>resultr)InMinr=resultr;
			if(InMing>resultg)InMing=resultg;
			if(InMinb>resultb)InMinb=resultb;
			if(InMaxr<resultr)InMaxr=resultr;
			if(InMaxg<resultg)InMaxg=resultg;
			if(InMaxb<resultb)InMaxb=resultb;
		}
	scaler=(OutMax-OutMin)/(InMaxr-InMinr);
	scaleg=(OutMax-OutMin)/(InMaxg-InMing);
	scaleb=(OutMax-OutMin)/(InMaxb-InMinb);
	color=(OutMax-OutMin)/res;
	for(i=0;i<200;i++)
		for(j=0;j<200;j++)
		{
			w=dcMem.GetPixel(j,i);
			WarnaToRGB(w,&r,&g,&b);
			resultr=r;
			resultg=g;
			resultb=b;
			resultr=scaler*(resultr-InMinr);
			resultg=scaleg*(resultg-InMing);
			resultb=scaleb*(resultb-InMinb);
			resultr=(resultr/color)*color;
			resultg=(resultg/color)*color;
```

```
                        resultb=(resultb/color)*color;
                        resultr=resultr+OutMin;
                        resultg=resultg+OutMin;
                        resultb=resultb+OutMin;
                        w=RGBToWarna(resultr,resultg,resultb);
                        dcMem.SetPixel(j,i,w);
                    }
            pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}
```

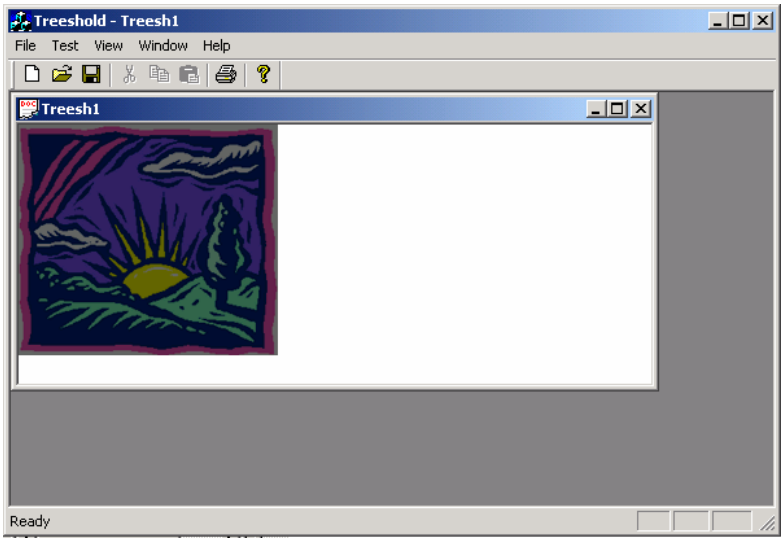
2. Menambah header file

   - Buka file TreesHoldView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CTreesHoldDoc* GetDocument();
        void LoadGambar(void);
// Operations
public:
        CString name;
        CBitmap m_bmpBitmap;
```

3. Cara menjalankan program

   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu : Test->TreesHolding -> hasilnya seperti gambar 3.5




Gambar 3.5 TreesHolding 100 warna resolusi 10

## 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 threshloding 2 menghasilkan citra yang sama ?

## 3.6. Laporan Resmi:

Buatlah laporan resmi dari latihan-latihan diatas dengan cara membuat analisa dan kesimpulan.

Praktikum

4

# Perbaikan Citra (Enhancement 1)

## 4.1. Tujuan:

1. Mahasiswa dapat membuat program untuk menampilkan histogram dari citra
2. Mahasiswa dapat membuat program untuk memperjelas citra (brighness) dan kontras

## 4.2. Dasar Teori:

### 4.2.1. Histogram

Banyak sekali proses pengolahan citra yang melibatkan distribusi data, seperti pada contoh konversi biner di atas. Bahkan dalam image enhancement (perbaikan citra), distribusi dari nilai derajat keabuan pada citra menjadi suatu acuan dasar. Untuk menyatakan distribusi data dari nilai derajat keabuan ini dapat digunakan nilai histogram. Histogram adalah suatu fungsi yang menyatakan jumlah kemunculan dari setiap nilai. Misalkan diketahui data sebagai berikut:

X = 1 3 2 5 3 0 2 1 2 4 2 3

Maka histogramnya adalah munculnya setiap nilai, yaitu: nilai 0 muncul 1 kali, nilai 1 muncul 2 kali, nilai 2 muncul 4 kali, nilai 3 muncul 3 kali, nilai 4 muncul 1 kali dan nilai 5 muncul 1 kali. Karena citra mempunyai derajat keabuan 256 yaitu (0-255) maka histogram menyatakan jumlan kemunculan setiap nilai 0-255.

### 4.2.2. Brightness

Brightness adalah proses penambahan kecerahan dari nilai derajat keabuan. Proses brightness ini dilakukan dengan me-nambahkan nilai derajat keabuan dengan suatu nilai penambah.

$$xb = x + b$$

dimana

x = adalah nilai derajat keabuan

b = nilai penambah

xb = hasil brightness

#### 4.2.3. Mengubah Kontras

Mengubah kontras dari suatu citra adalah proses pengaturan nilai range interval pada setiap nilai derajat keabuan, dan didefinisikan dengan :

$xk = k\,x$

dimana

x = nilai derajat keabuan

k = nilai kontras

xk = nilai setelah pengaturan kontras

## 4.3. Tugas Pendahuluan:

1. Tuliskan tujuan praktikum
2. Jelaskan cara membuat histogram
3. Jelaskan cara mengubah citra menjadi cerah
4. Jelaskan cara mengubah citra menjadi kontras

## 4.4. Percobaan:

#### 4.4.1. Histogram

1. Cara membuat Histogram
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan HistoGram
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile, Histogram
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file HistogramView.cpp
   - Tambahkan program untuk membuat histogram seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////////
// CHistoGramView message handlers

void CHistoGramView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	 OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CHistoGramView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
	name, IMAGE_BITMAP, 0, 0,
	LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CHistoGramView::OnTestHistogram()
{
	// TODO: Add your command handler code here
	int histRed[256],histGreen[256],histBlue[256];
	long int warna;
```

```
        int i,j,k,red,green,blue;
        float accRed=0.0,accGreen=0.0,accBlue=0.0;
        int tempRed[256],tempGreen[256],tempBlue[256];
        CDC* pDC = GetDC();
        CDC dcMem;
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);
        for(i=0;i<256;i++)
        {
                histRed[i]=0;
                histGreen[i]=0;
                histBlue[i]=0;
        }
        for(j=0;j<210;j++)
                for(k=0;k<250;k++)
                {
                        warna=dcMem.GetPixel(k,j);
                        WarnaToRGB(warna,&red,&green,&blue);
                        histRed[red]=histRed[red]+1;
                        histGreen[green]=histGreen[green]+1;
                        histBlue[blue]=histBlue[blue]+1;
                }
        for(j=0;j<210;j++)
                for(k=0;k<250;k++)
                {
                        warna=dcMem.GetPixel(k,j);
                        WarnaToRGB(warna,&red,&green,&blue);
                warna=RGBToWarna(tempRed[red],tempGreen[green],tempBlue[blue]);
                        dcMem.SetPixel(k,j,warna);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

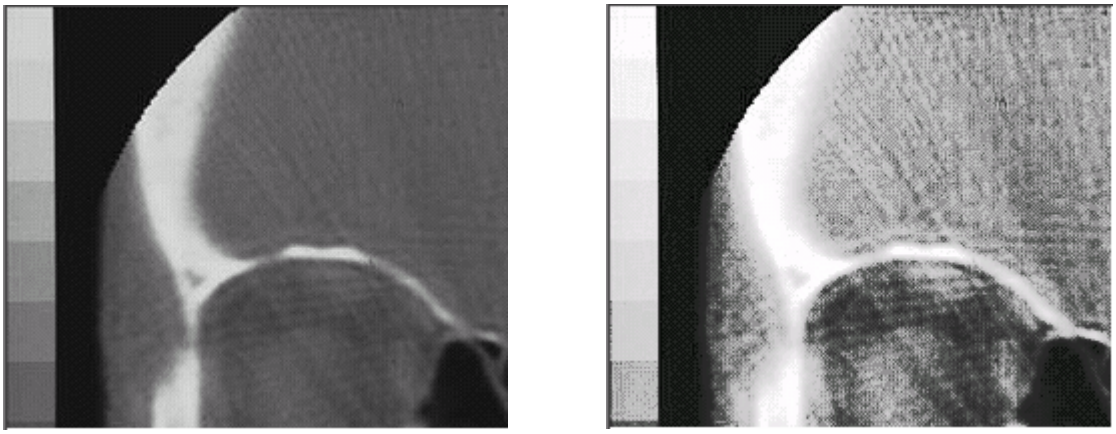
2. Menambah header file

   - Buka file HistogramView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CHistogramDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;

// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program

   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya xray.bmp
   - Pilih menu: Test->Histogram hasilnya seperti gambar 4.1



Gambar 4.1 Histogram dari citra

### 4.4.2. Brighness (Memperjelas Gambar)

1. Cara memperjelas gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Terang
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan Terang
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file TerangView.cpp
   - Tambahkan program untuk memperjelas gambar seperti dibawah ini

```
////////////////////////////////////////////////////////////////////////////
// CTerangView message handlers

void CTerangView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CTerangView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
```

```
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CTerangView::OnTestTerang()
{
	// TODO: Add your command handler code here
	long int warna;
	int j,k,red,green,blue;
	CDC* pDC = GetDC();
	CDC dcMem;
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	for(j=0;j<210;j++)
		for(k=0;k<250;k++)
		{
			warna=dcMem.GetPixel(k,j);
			// merubah data pixel ke RGB
			WarnaToRGB(warna,&red,&green,&blue);

			// memperjelas gambar dengan menambah 50
			red=red+50;
			if(red>255)red=255;
			green=green+50;
			if(green>255)green=255;
			blue=blue+50;
			if(blue>255)blue=255;

			//merubah RGB ke data pixel
			warna=RGBToWarna(red,green,blue);
			dcMem.SetPixel(k,j,warna);
		}
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
```

```
}
```

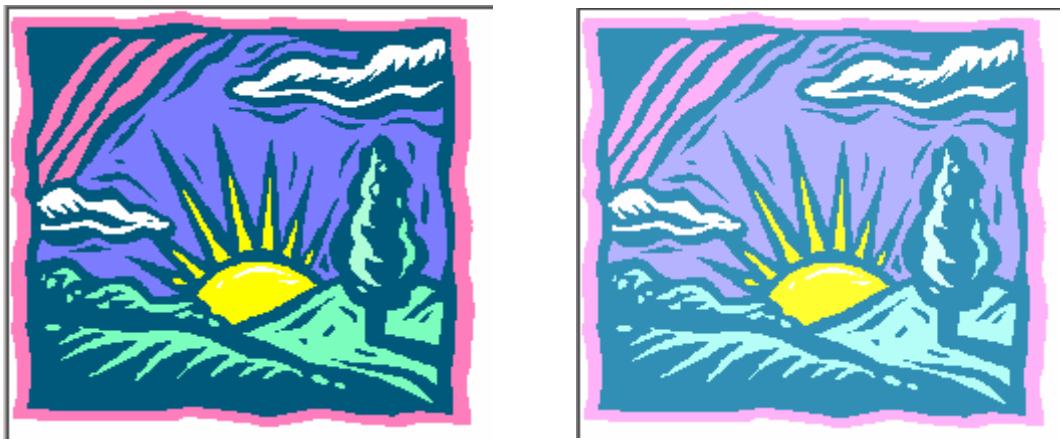
2. Menambah header file

   - Buka file TerangView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CTerangDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;

// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program

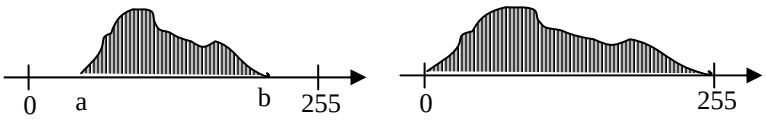
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Terang hasilnya seperti gambar 4.2



Gambar 4.2 Brighness (Memperjelas gambar)

## 4.5. Latihan:

1. Dengan menggunakan proses brightness dan kontras bagaimana cara menjamin distribusi derajat keabuan pada suatu citra merupakan distribusi derajat keabuan yang memenuhi nilai 0-255?




Gambar 4.3 Contoh penyebaran merata histogram

2. Buatlah program untuk dapat melakukan proses kontras dan kemudian inversi gambar? Apa yang terjadi bila dilakukan kontras yang maksimal sehingga memenuhi ruang derajat keabuan 0-255 dan kemudian diinvers?
3. Bagaimana langkah-langkah untuk menghitung histogram derajat keabuan suatu citra?
4. Jelaskan apa kegunaan mengetahui histogram dari derajat keabuan dari suatu citra?

## 4.6. Laporan Resmi:

Buatlah laporan resmi dari latihan-latihan diatas dengan cara membuat analisa dan kesimpulan.

Praktikum

5

# Perbaikan Citra (Enhancement 2)

## 5.1. Tujuan:

1. Mahasiswa dapat membuat program untuk menginvers dari citra
2. Mahasiswa dapat membuat program untuk memperjelas citra dengan histogram Equalization

## 5.2. Dasar Teori:

### 5.2.1. Inversi Citra

Inversi citra adalah proses negatif pada citra, misalkan pada photo, dimana setiap nilai citra dibalik dengan acuan threshold yang diberikan. Proses ini banyak digunakan pada citra-citra medis seperti USG dan X-Ray. Untuk citra dengan derajat keabuan 256, proses inversi citra didefinisikan dengan:

$$xn = 255 - x$$

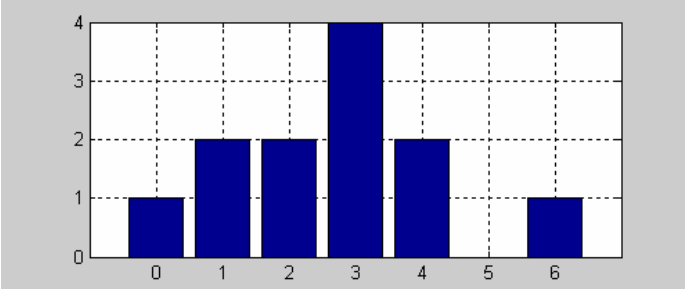
### 5.2.2. Histogram Equalization

Histogram Equalization adalah suatu proses perataan histogram, dimana distribusi nilai derajat keabuan pada suatu citra dibuat rata. Untuk dapat melakukan histogram equalization ini diperlukan suatu fungsi distribusi kumulatif yang merupakan kumulatif dari histogram.

Misalkan diketahui data sebagai berikut:

2 4 3 1 3 6 4 3 1 0 3 2

Maka histogram dari data di atas adalah:

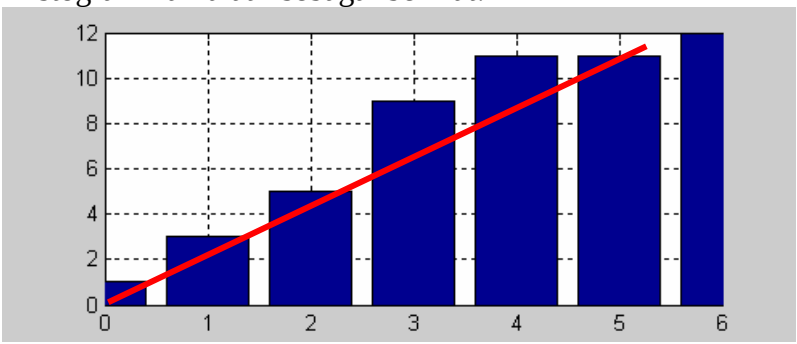



Gambar 5.1 Contoh histogram

Proses perhitungan distribusi kumulatif dapat dijelaskan dengan tabel berikut:

| Nilai | Histogram | Dsitribusi kumulatif |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 1+2=3 |
| 2 | 2 | 3+2=5 |
| 3 | 4 | 5+4=9 |
| 4 | 2 | 9+2=11 |
| 5 | 0 | 11+0=11 |
| 6 | 1 | 11+1=12 |

Dan diperoleh histogram kumulatif sebagai berikut:




Gambar 5.2 Histogram kumulatif

Histogram equalization (perataan histogram) adalah suatu proses dimana histogram diratakan berdasarkan suatu fungsi linier (garis lurus) seperti terlihat pada gambar 5.2. Teknik perataan histogram adalah sebagai berikut:

| Nilai asal | Histogram Kumulatif | Nilai hasil |
|---|---|---|
| 0 | 1 | ½ → 0 |
| 1 | 3 | 3/2 → 1 |
| 2 | 5 | 5/2 → 2 |
| 3 | 9 | 9/2 → 4 |
| 4 | 11 | 11/2 → 5 |
| 5 | 11 | 11/2 → 5 |
| 6 | 12 | 12/2 → 6 |

Nilai hasil histogram equalization adalah sebagai berikut:

$$w = \frac{c_w.th}{n_x n_y}$$

dimana

w = nilai keabuan hasil histogram equalization

$c_w$ = histogram kumulatif dari w

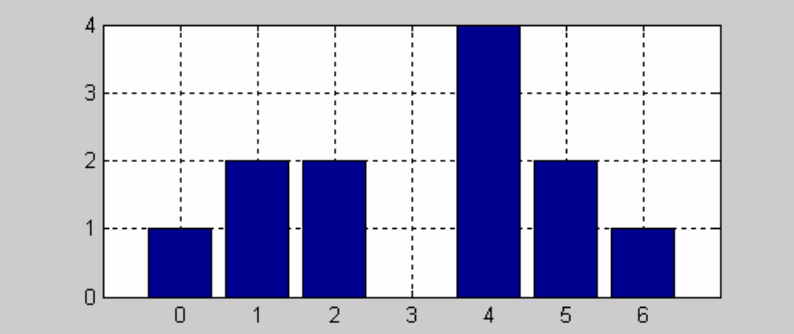
th = threshold derajat keabuan (256)

$n_x$ dan $n_y$ = ukuran gambar

Hasil setelah histogram equalization adalah sebagai berikut:

2 5 4 1 4 6 5 4 1 0 4 2

Histogram dari hasil histogram equalization:




Gambar 5.3 Histogram dari hasil histogram equalization

## 5.3. Tugas Pendahuluan:

1. Tuliskan tujuan praktikum
2. Jelaskan cara mengubah citra menjadi inversnya
3. Jelaskan cara mengubah citra menjadi cerah dengan histogram equalization

## 5.4. Percobaan:

### 5.4.1. Inversi

1. Cara menginversi gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Inversi
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile, Inversi

- Untuk mengedit isi program tekan tombol Edit Code atau buka file InversiView.cpp
- Tambahkan program untuk menginversi gambar seperti dibawah ini

```
//////////////////////////////////////////////////////////////////////////
// CInversiView message handlers
void CInversiView::OnTestLoadgambar()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";
	CFileDialog m_ldFile(TRUE, "*.bmp", name,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}
// Menampilkan gambar hasil dari open file
void CInversiView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
// merubah data pixel ke RGB
void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
	*Red = warna & 0x000000FF;
	*Green = (warna & 0x0000FF00) >> 8;
	*Blue = (warna & 0x00FF0000) >> 16;
}

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CInversiView::OnTestInversi()
{
	// TODO: Add your command handler code here
	long int warna;
	int j,k,red,green,blue;
```

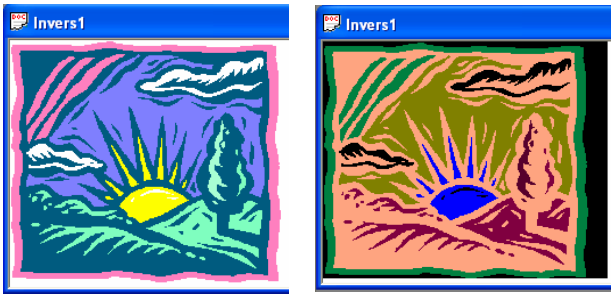
```
        CDC* pDC = GetDC();
        CDC dcMem;
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);
        for(j=0;j<210;j++)
                for(k=0;k<250;k++)
                {
                        warna=dcMem.GetPixel(k,j);
                        // merubah data pixel ke RGB
                        WarnaToRGB(warna,&red,&green,&blue);

                        // memperjelas gambar dengan menambah 50
                        red=255-red;
                        green=255-green;
                        blue=255-blue;

                        //merubah RGB ke data pixel
                        warna=RGBToWarna(red,green,blue);
                        dcMem.SetPixel(k,j,warna);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file

   - Buka file InversiView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CInversiDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;
// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Inversi hasilnya seperti gambar 5.4




Gambar 5.4 Inversi Citra

### 5.4.1. Histogram Equalization

1. Cara memperjelas gambar dengan Histogram Equalization
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan HistoGram
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile, Histogram
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file HistogramView.cpp
   - Tambahkan program untuk memperjelas gambar seperti dibawah ini

```
////////////////////////////////////////////////////////////////////////////
// CHistoGramView message handlers

void CHistoGramView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	 OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

// Menampilkan gambar hasil dari open file
void CHistoGramView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
	name, IMAGE_BITMAP, 0, 0,
	LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

// merubah data pixel ke RGB
void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
```

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CHistoGramView::OnTestHistogram()
{
	// TODO: Add your command handler code here
	int histRed[256],histGreen[256],histBlue[256];
	long int warna;
	int i,j,k,red,green,blue;
	float accRed=0.0,accGreen=0.0,accBlue=0.0;
	int tempRed[256],tempGreen[256],tempBlue[256];
	CDC* pDC = GetDC();
	CDC dcMem;
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	for(i=0;i<256;i++)
	{
		histRed[i]=0;
		histGreen[i]=0;
		histBlue[i]=0;
	}
	for(j=0;j<210;j++)
		for(k=0;k<250;k++)
		{
			warna=dcMem.GetPixel(k,j);
			WarnaToRGB(warna,&red,&green,&blue);
			histRed[red]=histRed[red]+1;
			histGreen[green]=histGreen[green]+1;
			histBlue[blue]=histBlue[blue]+1;
		}
	for(i=0;i<256;i++)
	{
		accRed=accRed+(float)histRed[i]/(250*210);
		accGreen=accGreen+(float)histGreen[i]/(250*210);
		accBlue=accBlue+(float)histBlue[i]/(250*210);
		tempRed[i]=int(accRed*255);
		tempGreen[i]=int(accGreen*255);
		tempBlue[i]=int(accBlue*255);
		if(tempRed[i]>255)tempRed[i]=255;
		if(tempGreen[i]>255)tempGreen[i]=255;
		if(tempBlue[i]>255)tempBlue[i]=255;
	}
	for(j=0;j<210;j++)
		for(k=0;k<250;k++)
		{
			warna=dcMem.GetPixel(k,j);
			WarnaToRGB(warna,&red,&green,&blue);
```

```
            warna=RGBToWarna(tempRed[red],tempGreen[green],tempBlue[blue]);
                        dcMem.SetPixel(k,j,warna);
                }
            pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

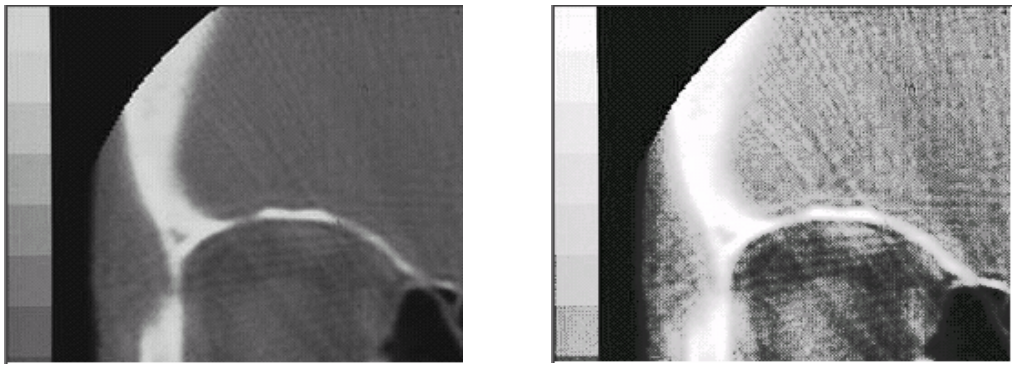
2. Menambah header file

   - Buka file HistogramView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CHistogramDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;

// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program

   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya xray.bmp
   - Pilih menu: Test->Histogram hasilnya seperti gambar 5.5



Gambar 5.5 Memperjelas gambar dengan Histogram Equalization

## 5.5. Latihan:

1. Buatlah program untuk dapat melakukan proses kontras dan kemudian inversi gambar ? Apa yang terjadi bila dilakukan kontras yang maksimal sehingga memenuhi ruang derajat keabuan 0-255 dan kemudian diinvers?
2. Jelaskan langkah-langkah dari proses histogram equalization dengan menggunakan algoritma.

## 5.6. Laporan Resmi:

Buatlah laporan resmi dari latihan-latihan diatas dengan cara membuat analisa dan kesimpulan.

Praktikum

# 6 Filtering

## 6.1. Tujuan:

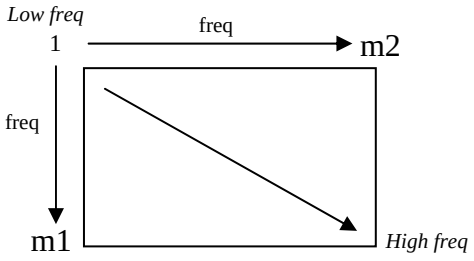
1. Mahasiswa dapat memahami prinsip-prinsip filtering pada gambar
2. Mahasiswa dapat menggunakan proses konvolusi untuk melakukan filter pada gambar
3. Mahasiswa dapat menggunakan filter transparans pada dua citra

## 6.2. Dasar Teori:

### 6.2.1. Prinsip-Prinsip Filtering

Filtering adalah suatu proses dimana diambil sebagian sinyal dari frekwensi tertentu, dan membuang sinyal pada frekwensi yang lain. Filtering pada citra juga menggunakan prinsip yang sama, yaitu mengambil fungsi citra pada frekwensi-frekwensi tertentu dan membuang fungsi citra pada frekwensi-frekwensi tertentu.

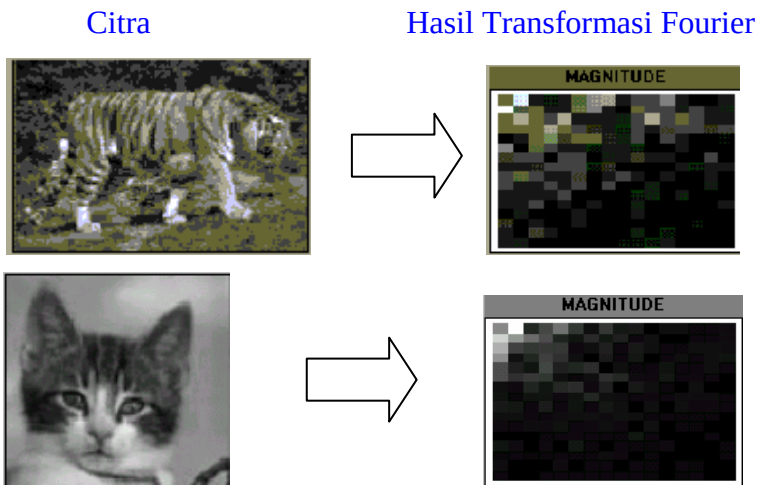
Berdasarkan sifat transformasi fourier dari suatu citra dan format koordinat frekwensi seperti gambar 6.1. berikut ini:




Gambar 6.1. Format koordinat frekwensi pada citra

Berikutnya kita perhatikan bagaimana pengaruh frekwensi renda dan frekwensi tinggi pada citra dengan memanfaatkan hasil dari transformasi fourier. Dimana frekwensi pada

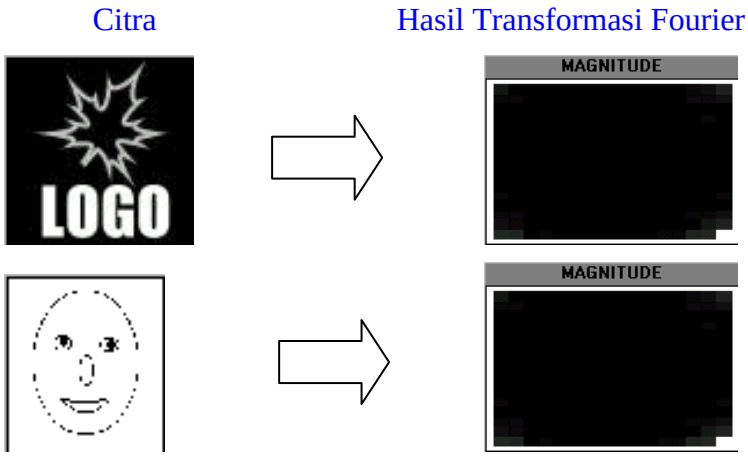
citra dipengaruhi oleh gradiasi warna yang ada pada citra tersebut. Perhatikan hasil transformasi fourier dari beberapa citra berikut 6.2. berikut.




Gambar 6.2. Contoh transformasi fourier citra bergradiasi tinggi

Perhatikan bahwa warna putih (terang) pada gambar hasil transformasi fourier menunjukkan level atau nilai fungsi yang tinggi dan warna hitam (gelap) menunjukkan level atau nilai fungsi yang rendah. Berdasarkan hal ini dan format koordinat frekwensi (pada gambar 5.1) terlihat bahwa pada gambar 5.2 nilai-nilai yang tinggi berada pada frekwensi rendah, ini menunjukkan bahwa citra dengan gradiasi (level threshold) tinggi cenderung berada pada frekwensi rendah. **Sehingga dapat disimpulkan bahwa citra dengan gradiasi tinggi berada pada frekwensi rendah**.

Berikutnya dengan menggunakan citra-citra yang bergradiasi rendah seperti gambar logo data sketsa dimana nilai treshold yang digunakan merupakan nilai-nilai yang kecil dapat dilihat pada gambar 6.3. berikut. Pada gambar 6.3. terlihat bahwa hasil transformasi fourier menunjukkan nilai fungsi hanya berada pada frekwensi tinggi. Dengan demikian dapat disimpulkan bahwa **citra yang bergradiasi rendah berada pada frekwensi tinggi**. Demikian pula citra biner, citra dengan threshold tertentu merupakan citra-citra yang bergradiasi rendah, dan citra-citra ini berada pada frekwensi tinggi.




Gambar 6.3. Contoh transformasi fourier citra bergradiasi rendah

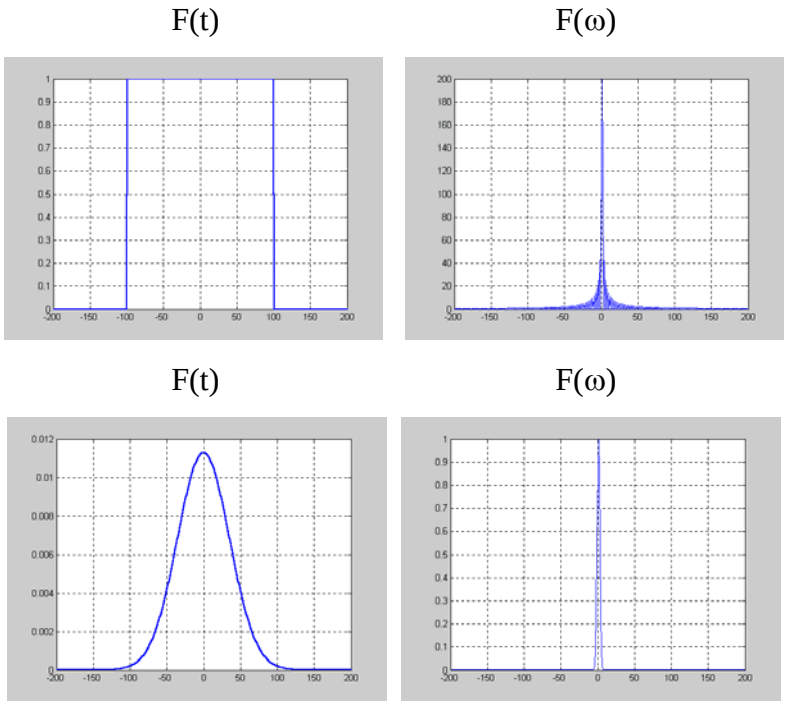
Dari sifat-sifat citra pada bidang frekwensi, maka prinsip-prinsip filtering dapat dikembangkan adalah sebagai berikut:

(1) Bila ingin mempertahankan gradiasi atau banyaknya level warna pada suatu citra, maka yang dipertahankan adalah frekwensi rendah dan frekwensi tinggi dapat dibuang atau dinamakan dengan Low Pass Filter. Hal ini banyak digunakan untuk reduksi noise dan proses blur.

(2) Bila ingin mendapatkan threshold atau citra biner yang menunjukkan bentuk suatu gambar maka frekwensi tinggi dipertahankan dan frekwensi rendah dibuang atau dinamakan dengan High Pass Filter. Hal ini banyak digunakan untuk menentukan garis tepi (*edge*) atau sketsa dari citra.

(3) Bila ingin mempertahankan gradiasi dan bentuk, dengan tetap mengurangi banyaknya bidang frekwensi (bandwidth) dan membuang sinyal yang tidak perlu maka frekwensi rendah dan frekwensi tinggi dipertahankan, sedangkan frekwensi tengahan dibuang atau dinamakan dengan Band Stop Filter. Teknik yang dikembangkan dengan menggunakan Wavelet Transform yang banyak digunakan untuk kompresi, restorasi dan denoising.

Teknik-teknik filtering pada citra dikembangkan menggunakan prinsip-prinsip di atas. Perhatikan beberapa transformasi fourier dari fungsi yang semuanya positif sebagai berikut:




Gambar 6.4. transformasi fourier untuk fungsi positif integer

Dari gambar 6.4. di atas yang masing-masing menyatakan transformasi fourier dari fungsi rata-rata dan fungsi gaussian, dapat dilihat bahwa fungsi-fungsi ini menghasilkan transformasi fourier yang nilainya hanya berada pada frekwensi rendah. Hal ini dapat dinyatakan bahwa filter yang berjenis Low Pass Filter dapat diperoleh dengan suatu kernel filter H (koefisien filter dalam spasial atau koordinat x dan y) yang semua nilainya positif (positif absolut).

## 6.2.2. Konvolusi

Konvolusi adalah perkalian total dari dua buah fungsi f dan f yang didefinisikan dengan:

$$f * h = \int_0^T f(t)h(T-t)dt$$

Untuk fungsi f dan h yang berdimensi 2, maka konvolusi dua dimensi didefinisikan dengan:

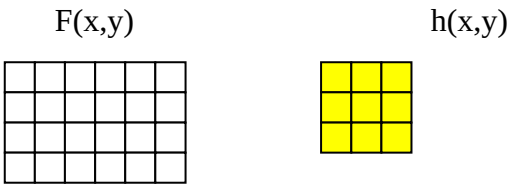
$$f * h = \int_0^{T_x} \int_0^{T_y} f(x,y)h(T_x - x, T_y - y)dxdy$$

Konvolusi 2D inilah yang banyak digunakan pengolahn citra digital, sayangnya rumus diatas sangat sulit diimplementasikan menggunakan komputer, karena pada dasarnya komputer hanya bisa melakukan perhitungan pada data yang diskrit sehingga tidak dapat digunakan untuk menghitung intregral di atas☹

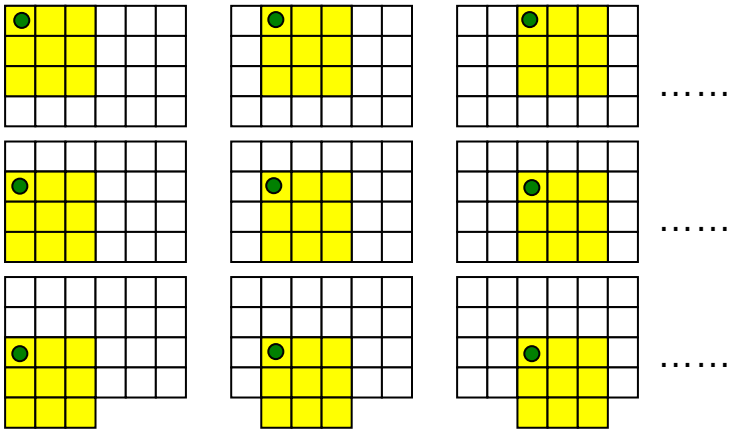
Konvolusi pada fungsi diskrit f(n,m) dan h(n,m) didefinisikan dengan:

$$y(k1,k2) = \sum_{n=1}^{Tn} \sum_{m=1}^{Tm} f(k1+n, k2, m)h(n,m)$$

Perhitungan konvolusi semacam ini dapat digambarkan dengan:




Bila ingin dihitung y = f * h, maka proses perhitungannya dapat dilakukan dengan:




Gambar 5.6. Perhitungan konvolusi secara grafis

Filter pada citra pada bidang spasial dapat dilakukan dengan menggunakan konvolusi dari citra (I) dan fungsi filternya (H), dan dituliskan dengan:

I’ = H ⊗ I

Dan dirumuskan dengan:

$$I'(x,y) = \sum_{i=-n}^{n} \sum_{j=-m}^{m} h(i,j)I(x+i, y+j)$$

dimana :

m,n adalah ukuran dari fungsi filter dalam matrik

Rumus konvolusi di atas digunakan untuk berbagai macam proses filter yang akan dijelaskan pada sub bab-sub bab berikut.

### 6.2.3. Low Pass Filter

Seperti telah dijelaskan di atas bahwa low pass filter adalah proses filter yang mengambil citra dengan gradiasi intensitas yang halus dan perbedaan intensitas yang tinggi akan dikurangi atau dibuang. Ciri-ciri dari fungsi low-pass filter adalah sebagai berikut: $\sum_j \sum_i H(i,j) = 1$

Sebagai contoh dibuat program Low Pass Filter dengan fungsi filter rata-rata sebagai berikut:

$$H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Hasil dari program Low Pass Filter, ini untuk beberapa macam gambar adalah sebagai berikut:


Gambar 6.5 Hasil LPF untuk gambar kucing dan komputer

Dari kedua hasil di atas dapat dilihat bahwa Low Pass Filter menyebabkan gambar menjadi lebih halus dan lebih blur.

### 6.2.4. High Pass Filter

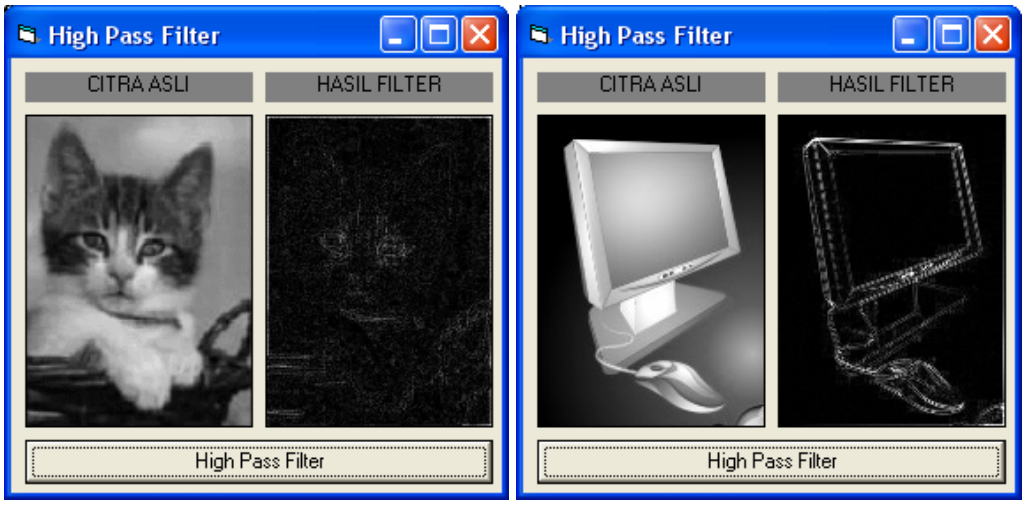
Seperti telah dijelaskan di atas bahwa high pass filter adalah proses filter yang mengambil citra dengan gradiasi intensitas yang tinggi dan perbedaan intensitas yang rendah akan dikurangi atau dibuang. Ciri-ciri dari fungsi low-pass filter adalah sebagai berikut:

$$\sum_j \sum_i H(i, j) = 0$$

Sebagai contoh dibuat program Low Pass Filter dengan fungsi filter rata-rata sebagai berikut:

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Hasil dari program High Pass Filter, ini untuk beberapa macam gambar adalah sebagai berikut:




Gambar 5.8 Hasil HPF untuk gambar kucing dan komputer

Dari kedua hasil di atas dapat dilihat bahwa High Pass Filter menyebabkan gambar hanya diambil atau ditampilkan pada daerah-daerah yang berbeda misalkan pada tepi-tepi

gambar. Pada gambar kucing perbedaan yang muncul tidak begitu jelas karena gambarnya mempunyai gradiasi yang tinggi (halus), sedangkan pada gambar komputer tepi-tepi gambar tampak jelas karena perbedaannya tinggi.

## 6.3. Tugas Pendahuluan:

(1) Tuliskan tujuan praktikum

(2) Tuliskan prinsip-prinsip fitering pada citra

(3) Hitunglah konvolusi dari: $f = \begin{bmatrix} 3 & 6 & 4 \\ 0 & 3 & 7 \end{bmatrix}$ dan $h = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$

(4) Hitunglah konvolusi dari: $f = \begin{bmatrix} 3 & 6 & 4 \\ 0 & 3 & 7 \end{bmatrix}$ dan $h = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$

(5) Tuliskan apa ciri-ciri Low Pass Filter pada citra, dan bagaimana hasilnya

(6) Tuliskan apa ciri-ciri High Pass Filter pada citra, dan bagaimana hasilnya

## 6.3. Percobaan:

Percobaan yang dilakukan antara lain adalah:

(1) Konvolusi untuk Filter pada Citra

(2) Transparansi

### 6.3.1. PERCOBAAN 1: KONVOLUSI

**Percobaan ini melakukan proses filter dengan proses konvolusi dengan kernel filter:**

$$H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

1. Cara membersihkan gambar

- Buat aplikasi AppWizard seperti pada percobaan 1.1 dan beri nama project dengan Bersih
- Buat Menu seperti pada percobaan 1.2 dengan tambahan Test sedangkan submenunya OpenFile, Asal dan Hasil

- Untuk mengedit isi program tekan tombol Edit Code atau buka file BersihView.cpp
- Tambahkan program untuk membersihkan gambar seperti dibawah ini

```
void CBersihView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

// Menampilkan gambar hasil dari open file
void CBersihView::LoadGambar(void)
{
```

```
		CDC* pDC = GetDC();
		CDC dcMem;
		long int w,w1;
		int i,j,r,g,b,x;
		HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
		if(hBitmap)
		{
			if(m_bmpBitmap.DeleteObject())
				m_bmpBitmap.Detach();
			m_bmpBitmap.Attach(hBitmap);
		}
		dcMem.CreateCompatibleDC(pDC);
		dcMem.SelectObject(&m_bmpBitmap);
		//pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

		// Proses RGB to GRAY-SCALE
		for(i=0;i<210;i++)
			for(j=0;j<250;j++) {
				w=dcMem.GetPixel(j,i);
				WarnaToRGB(w,&r,&g,&b);
				x=int((r+g+b)/3);
				w1=RGBToWarna(x,x,x);
				dcMem.SetPixel(j,i,w1);
			}
		pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

	}

	void CBersihView::OnTestSetelahfilter()
	{
		// TODO: Add your command handler code here

		CDC* pDC = GetDC();
		CDC dcMem;
		int i,j,r,g,b;
		int resultr,resultg,resultb;
```

```
long int w,mat[3][3];
float h[3][3],hr,hg,hb;

dcMem.CreateCompatibleDC(pDC);
dcMem.SelectObject(&m_bmpBitmap);

// Proses Konvolusi
int nh=3; // Menyatakan ukuran filter
// Penentuan kernel filter
h[0][0]=(float)1/9; h[0][1]=(float)1/9; h[0][2]=(float)1/9;
h[1][0]=(float)1/9; h[1][1]=(float)1/9; h[1][2]=(float)1/9;
h[2][0]=(float)1/9; h[2][1]=(float)1/9; h[2][2]=(float)1/9;

for(i=0;i<210;i++)
        for(j=0;j<250;j++){
                mat[0][0]=dcMem.GetPixel(j-1,i-1);
                mat[0][1]=dcMem.GetPixel(j,i-1);
                mat[0][2]=dcMem.GetPixel(j+1,i-1);
                mat[1][0]=dcMem.GetPixel(j-1,i);
                mat[1][1]=dcMem.GetPixel(j,i);
                mat[1][2]=dcMem.GetPixel(j+1,i);
                mat[2][0]=dcMem.GetPixel(j-1,i+1);
                mat[2][1]=dcMem.GetPixel(j,i+1);
                mat[2][2]=dcMem.GetPixel(j+1,i+1);
                hr=0;hg=0;hb=0;
                for(int u=0;u<nh;u++)
                        for(int v=0;v<nh;v++){
                                WarnaToRGB(mat[u][v],&r,&g,&b);
                                hr+=(float)r*h[u][v];
                                hg+=(float)g*h[u][v];
                                hb+=(float)b*h[u][v];
                        }
                resultr=(int)hr;
                resultg=(int)hg;
                resultb=(int)hb;
                if(resultr>255)resultr=255;
                if(resultg>255)resultg=255;
```

```
                    if(resultb>255)resultb=255;
                    w=RGBToWarna(resultr,resultg,resultb);
                    dcMem.SetPixel(j,i,w);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

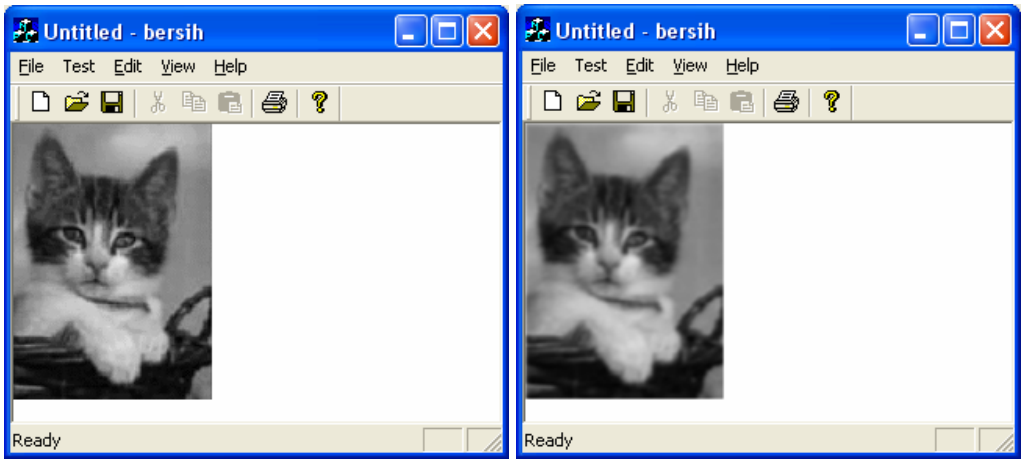
2. Menambah header file

   - Buka file BersihView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CBersihDoc* GetDocument();
        CString name;
        CBitmap m_bmpBitmap;

// Operations
public:
        void LoadGambar(void);
        void HighIntenPassFilterring(int filter);
```

3. Cara menjalankan program

   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Kotor hasilnya seperti gambar 6.9




Gambar 6.9. Contoh hasil filter dengan konvolusi

### 6.3.1. PERCOBAAN 2: FILTER TRANSPARANS

1. Cara mentransparansi gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Transparan
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dengan popup gambar1 dan gambar2, Transparan
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file TransparanView.cpp
   - Tambahkan program untuk Transparansi gambar seperti dibawah ini

```
////////////////////////////////////////////////////////////////////////////
// CTransparanView message handlers

void CTransparanView::OnTestOpenfileGambar1()
{
	// TODO: Add your command handler code here

	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name1,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name1=m_ldFile.GetPathName();
		LoadGambar1();
	}
}

// Menampilkan gambar hasil dari open file
void CTransparanView::LoadGambar1(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name1,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap1.DeleteObject())
```

```
                        m_bmpBitmap1.Detach();
                m_bmpBitmap1.Attach(hBitmap);
        }
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap1);
        pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}

void CTransparanView::OnTestOpenfileGambar2()
{
        // TODO: Add your command handler code here
        static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

        CFileDialog m_ldFile(TRUE, "*.bmp", name2,
        OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
        if(m_ldFile.DoModal()==IDOK)
        {
                name2=m_ldFile.GetPathName();
                LoadGambar2();
        }
}

// Menampilkan gambar hasil dari open file
void CTransparanView::LoadGambar2(void)
{
        CDC* pDC = GetDC();
        CDC dcMem;
        HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name2,
        IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
        if(hBitmap)
        {
                if(m_bmpBitmap2.DeleteObject())
                        m_bmpBitmap2.Detach();
                m_bmpBitmap2.Attach(hBitmap);
        }
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap2);
        pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}
// merubah data pixel ke RGB
void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
        *Red = warna & 0x000000FF;
        *Green = (warna & 0x0000FF00) >> 8;
        *Blue = (warna & 0x00FF0000) >> 16;
}

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
        return(Red+(Green<<8)+(Blue<<16));
```

```
}

void CTransparanView::OnTestTransparan()
{
	// TODO: Add your command handler code here
	int i,j,r,g,b,r1,g1,b1,r2,g2,b2;
	long int w;
	CDC* pDC = GetDC();
	CDC dcMem1,dcMem2;
	dcMem1.CreateCompatibleDC(pDC);
	dcMem1.SelectObject(&m_bmpBitmap1);
	dcMem2.CreateCompatibleDC(pDC);
	dcMem2.SelectObject(&m_bmpBitmap2);
	for(i=0;i<200;i++)
		for(j=0;j<200;j++)
		{
			w=dcMem1.GetPixel(j,i);
			WarnaToRGB(w,&r1,&g1,&b1);
			w=dcMem2.GetPixel(j,i);
			WarnaToRGB(w,&r2,&g2,&b2);
			r=(0.5*r1)+(0.5*r2);
			g=(0.5*g1)+(0.5*g2);
			b=(0.5*b1)+(0.5*b2);
			w=RGBToWarna(r,g,b);
			dcMem1.SetPixel(j,i,w);

		}
	pDC->BitBlt(0,0,200,200,&dcMem1,0,0,SRCCOPY);
}
```

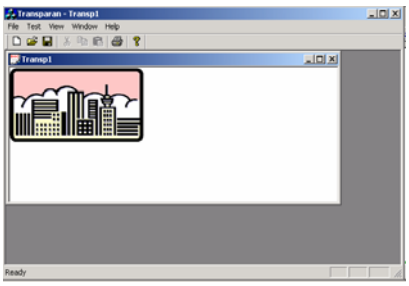
2. Menambah header file

   - Buka file TransparanView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
	CTransparanDoc* GetDocument();
	void LoadGambar1(void);
	void LoadGambar2(void);

// Operations
public:
	CString name1, name2;
	CBitmap m_bmpBitmap1,m_bmpBitmap2;
```

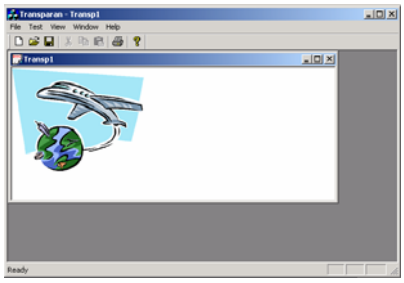
3. Cara menjalankan program

- Pilih menu : Build->Execute (!)
- Pilih menu : Test->OpenFile -> gambar1-> pilih salah satu gambar misalnya gambar1.bmp hasilnya seperti gambar 6.10



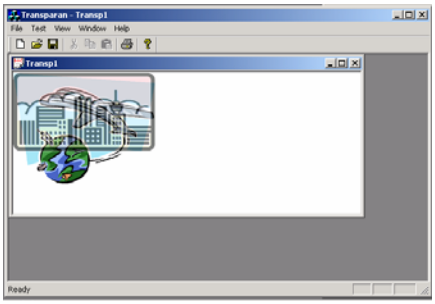
Gambar 6.10. gambar pertama

- Pilih menu : Test->OpenFile -> gambar2-> pilih salah satu gambar misalnya gambar2.bmp hasilnya seperti gambar 6.11



Gambar 6.11. gambar kedua

- Pilih menu: Test->Transparan hasilnya seperti gambar 6.12



Gambar 6.12. Transparansi Filtering

## 6.4. LAPORAN RESMI

### 6.4.1. Hasil Percobaan

(1) Tuliskan prinsip-prinsip fitering pada citra

(2) Tuliskan apa ciri-ciri Low Pass Filter pada citra, dan bagaimana hasilnya

(3) Dengan menggunakan program di atas, ubahlah filter dengan kernel filter di bawah ini dan jelaskan apa hasilnya.

$$h = \begin{bmatrix} 0.1 & 0.1 & 0.2 \\ 0 & 0.2 & 0 \\ 0.1 & 0.1 & 0.2 \end{bmatrix}$$

(4) Tuliskan apa ciri-ciri High Pass Filter pada citra, dan bagaimana hasilnya

(5) Dengan menggunakan program di atas, ubahlah filter dengan kernel filter di bawah ini dan jelaskan apa hasilnya.

$$h = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}$$

(6) Dengan program filter transparans, bagaimana hasilnya bila diberi dua gambar blok wara merah dan biru?

**6.4.2. Latihan**

(1) Pada program konvolusi di atas digunakan untuk ukuran filter 3, Buatlah program dari proses konvolusi dengan ukuran filter yang bebas.

(2) Pada program filter transparans, besar nilai transparansi $\alpha$=0.5, Buatlah program filter transparans dengan memasukkan nilai $\alpha$ dimana nilai $\alpha$ antara 0 sampai dengan 1.

Praktikum

# 7 Mereduksi Noise

## 7.1. TUJUAN:

1. Mahasiswa dapat memahami prinsip-prinsip noise dan cara mereduksi noise
2. Mahasiswa dapat membangkitkan bermacam-macam noise
3. Mahasiswa dapat menggunakan low pass filter untuk mengurangi noise

## 7.2. DASAR TEORI:

### 7.2.1. Noise Pada Citra

Pada saat proses *capture* (pengambilan gambar), ada beberapa gangguan yang mungkin terjadi, seperti kamera tidak fokus atau munculnya bintik-bintik yang bisa jadi disebabkan oleh proses *capture* yang tidak sempurna. Setiap gangguan pada citra dinamakan dengan noise. Noise pada citra tidak hanya terjadi karena ketidak-sempurnaan dalam proses capture, tetapi bisa juga disebabkan oleh kotoran-kotoran yang terjadi pada citra. Berdasarkan bentuk dan karakteristiknya, noise pada citra dibedakan menjadi beberapa macam yaitu:

(1) Gaussian
(2) Speckle
(3) Salt & Pepper

Macam-macam noise ini dapat dilihat pada gambar 6.1 berikut ini:

(a) (b) (c)

Gambar 7.1. Macam-macam noise (a) gaussian (b) speckle dan (c) salt & pepper

Noise gaussian merupakan model noise yang mengikuti distribusi normal standard dengan rata-rata nol dan standard deviasi 1. Efek dari gaussian noise ini, pada gambar muncul titik-titik berwarna yang jumlahnya sama dengan prosentase noise. Noise speckle merupakan model noise yang memberikan warna hitam pada titik yang terkena noise. Sedangkan noise salt & pepper seperti halnya taburan garam, akan memberikan warna putih pada titik yang terkena noise.

Pada beberapa pengolahan citra, terkadang untuk menguji suatu algoritma untuk dapat mereduksi noise, maka noise dihasilkan dari proses pembangkitan noise. Untuk membangkitkan noise digunakan suatu bilangan acak sebagai pengganti noise yang dihasilkan.

### 7.2.2. Membangkitkan Noise Uniform

Noise Uniform seperti halnya noise gausssian dapat dibangkitkan dengan cara membangkitkan bilangan acak [0,1] dengan distribusi uniform. Kemudian untuk titik-titik yang terkena noise, nilai fungsi citra ditambahkan dengan nilai noise yang ada, atau dirumuskan dengan:

$$y(i, j) = x(i, j) + p.a$$

dimana: a = nilai bilangan acak berditribusi uniform dari noise

p = prosentase noise

y(i,j) = nilai citra terkena noise.

x(i,j) = nilai citra sebelum terkena noise.

*Catatan:*

*Fungsi rnd dalam visual basic merupakan fungsi pembangkitan bilangan acak berdistribusi uniform, sehingga noise dapat dibangkitkan secara langsung dengan mengalikannya dengan besaran noise yang dibutuhkan.*

Noise uniform ini merupakan noise sintesis yang sebenarnya dalam penerapannya jarang digunakan, tetapi secara pemrograman pembangkitan noise uniform ini merupakan jenis pembangkitan noise yang paling mudah.


Gambar 7.2. Bebarapa contoh noise uniform dengan prosentase 10%, 20%, 30%, 50%, 75% dan 90%.

### 7.2.3. Membangkitkan Noise Gaussian

Noise gaussian dapat dibangkitkan dengan cara membangkitkan bilangan acak [0,1] dengan distribusi gaussian. Kemudian untuk titik-titik yang terkena noise, nilai fungsi citra ditambahkan dengan nilai noise yang ada, atau dirumuskan dengan:

$$y(i, j) = x(i, j) + p.a$$

dimana: a = nilai bilangan acak berdistribusi gaussian

p = prosentase noise

y(i,j) = nilai citra terkena noise.

x(i,j) = nilai citra sebelum terkena noise.

Untuk membangkitkan bilangan acak berdistribusi gaussian, tidak dapat langsung menggunakan fungsi rnd, tetapi diperlukan suatu metode yang digunakan untuk mengubah distribusi bilangan acak ke dalam fungsi f tertentu. Dalam buku ini digunakan metode rejection untuk memudahkan dalam alur pembuatan programnya. Metode rejection dikembangkan dengan cara membangkitkan dua bilangan acak (x,y) dan ditolak bila y > f(x).


Gambar 7.3. Bebarapa contoh noise gaussian dengan prosentase 10%, 20%, 30%, 50%, 75% dan 90%.

### 7.2.4. Membangkitkan Noise Salt & Pepper

Noise *salt & pepper* dapat dibangkitkan dengan cara membangkitkan bilangan 255 (warna putih) pada titik-titik yang secara probabilitas lebih kecil dari nilai probabilitas noise, dan dirumuskan dengan:

$$F(x,y)=255 \text{ jika } p(x,y) < ProbNoise$$

Dimana: F(x,y) adalah nilai gray-scale pada titik (x,y)

p(x,y) adalah probabilitas acak


Gambar 7.4. Bebarapa contoh noise salt & pepper dengan prosentase 10%, 20%, 30%, 50%, 75% dan 90%.

### 7.2.5. Reduksi Noise Menggunakan Filter Rata-Rata

Ada berbagai macam teknik untuk mengurangi (reduksi) noise, salah satunya menggunakan filter rata-rata. Dalam pengertian noise sebagai suatu nilai yang berbeda dengan semua tetangganya maka dapat dikatakan noise merupakan nilai-nilai yang berada pada frekwensi tinggi, untuk mengurangi noise digunakan Low Pass Filter (LPF). Salah satu dari bentuk LPF adalah filter rata-rata.

Filter rata-rata merupakan filter H dalam bentuk matrik yang berukuran mxn, dan nilainya adalah sama untuk setiap eleme, dan karena bersifat LPF maka jumlah seluruh elemn adalah satu, dan dituliskan dengan:

$$H(i,j) = \frac{1}{m.n}, 1 \le i \le m, 1 \le j \le n$$

Filter rata-rata berukuran 3x3 adalah:

$$H = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad \text{atau ditulis} \quad H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 9$$

Filter rata-rata berukuran 5x5 adalah:

$$H = \begin{bmatrix} 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \\ 1/25 & 1/25 & 1/25 & 1/25 & 1/25 \end{bmatrix}$$

Hasil perhitungan untuk masing-masing filter rata-rata dengan ukuran 3x3, 5x5, dan 7x7 ditunjukkan oleh gambar 7.5, 7.6, dan 7.7 berikut.


Gambar 7.5. Hasil filter rata-rata ukuran 3x3


Gambar 7.6. Hasil filter rata-rata ukuran 5x5


Gambar 7.7. Hasil filter rata-rata ukuran 7x7

Peningkatan ukuran filter memang akan semakain banyak mengurangi filter tetapi terjadi proses blur yang tidak dapat dihindari, hall ini menyebabkan nilai SNR juga akan semakin rendah.

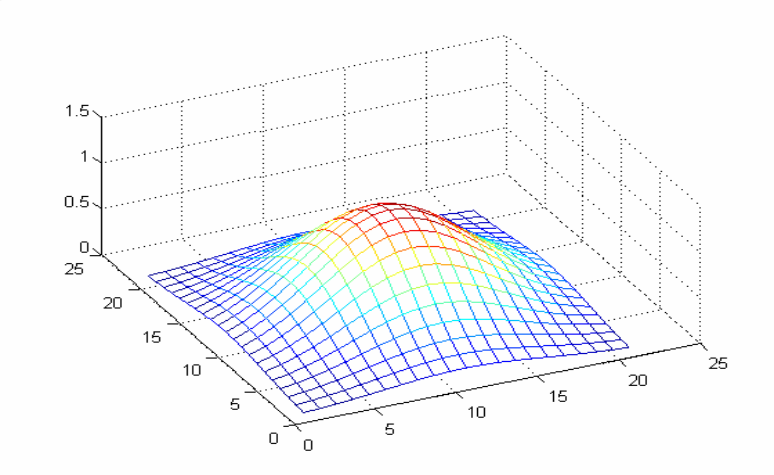
### 7.2.6. Reduksi Noise Menggunakan Filter Gaussian

Filter lain yang banyak digunakan dalam mereduksi noise selain filter rata-rata adalah filter gaussian. Filter gaussian ini sebenarnya hampir sama dengan filter rata-rata hanya ada nilai bobot yang tidak rata seperti pada filter rata-rata, tetapi mengikuti fungsi gaussian sebagai berikut:

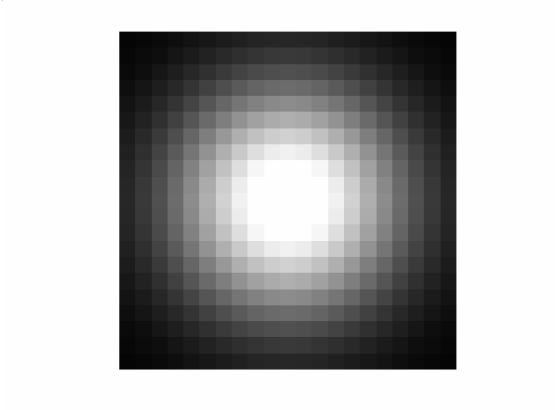
$$G(x,y) = \frac{1}{s\sqrt{\pi}} e^{-\left((x-m_x)^2+(y-m_y)^2\right)/2s}$$

dimana: s adalah sebaran dari fungsi gaussian

($m_x$,$m_y$) adalah titik tengah dari fungsi gaussian




Gambar 7.8 Model Fungsi Gaussian dalam ruang.



Gambar 7.9. Model contour dari filter gaussian.

Berdasarkan rumus dari fungsi gaussian di atas untuk ukuran 3x3 akan diperoleh matrik kernel filter gaussian:

$$H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 1 \\ 1 & 1 & 1 \end{bmatrix} / 13 \quad \text{atau} \quad H = \begin{bmatrix} 0.077 & 0.077 & 0.077 \\ 0.077 & 0.308 & 0.077 \\ 0.077 & 0.077 & 0.077 \end{bmatrix}$$

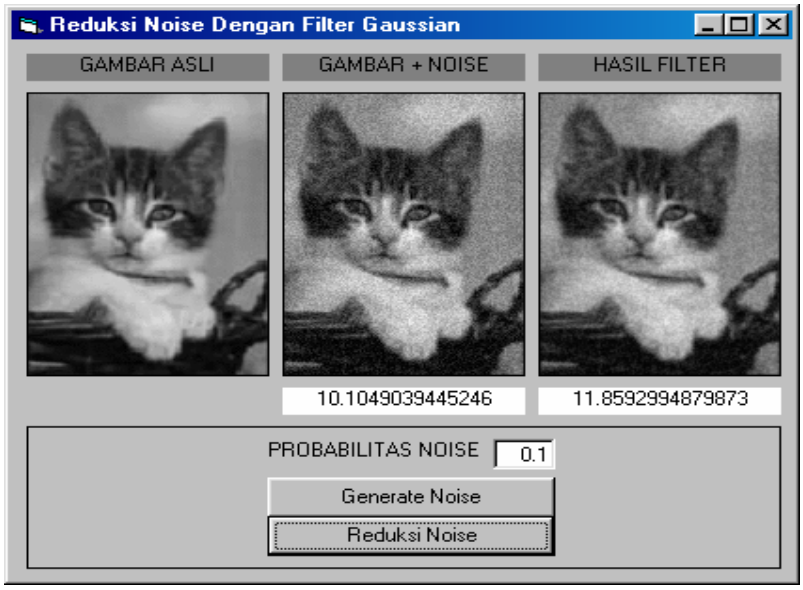
Kernel filter gaussian untuk ukuran 5x5 adalah:

```
H =    0.0030   0.0133   0.0219   0.0133   0.0030
       0.0133   0.0596   0.0983   0.0596   0.0133
       0.0219   0.0983   0.1621   0.0983   0.0219
       0.0133   0.0596   0.0983   0.0596   0.0133
       0.0030   0.0133   0.0219   0.0133   0.0030
```

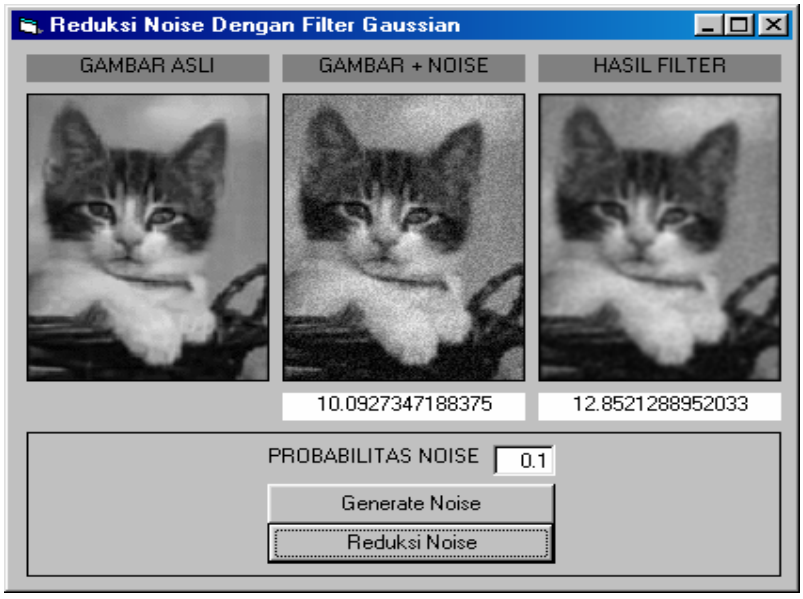
Kernel filter gaussian untuk ukuran 7x7 adalah:

```
0.0013 0.0041 0.0079 0.0099 0.0079 0.0041 0.0013
0.0041 0.0124 0.0241 0.0301 0.0241 0.0124 0.0041
0.0079 0.0241 0.0470 0.0587 0.0470 0.0241 0.0079
0.0099 0.0301 0.0587 0.0733 0.0587 0.0301 0.0099
0.0079 0.0241 0.0470 0.0587 0.0470 0.0241 0.0079
0.0041 0.0124 0.0241 0.0301 0.0241 0.0124 0.0041
0.0013 0.0041 0.0079 0.0099 0.0079 0.0041 0.0013
```

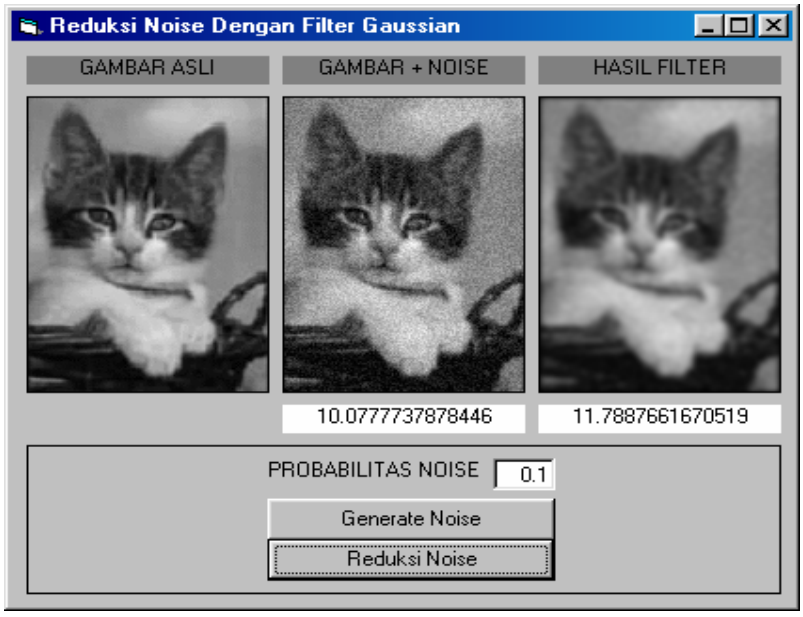
Berikut ini hasil dari program filter gaussian di atas dengan ukuran kernel yang diubah-ubah, yaitu 3x3, 5x5 dan 7x7. Hall ini dapat dilakukan dengan mengubah nilai Nfilter yang ada pada even form_load dengan nilai 3, 5 atau 7 secara manual, dan kemudian jalankan programnya. Perhatikan bagaimana hasil dari masing-masing ukuran filter gaussian dan bandingkan dengan dengan hasil dari filter rata-rata.




Gambar 7.10. Hasil filter gaussian dengan ukuran 3x3




Gambar 7.11. Hasil filter gaussian dengan ukuran 5x5




Gambar 7.12. Hasil filter gaussian dengan ukuran 7x7

## 7.3. TUGAS PENDAHULUAN

(1) Tuliskan tujuan praktikum

(2) Tuliskan karakteristik dari noise uniform, noise gaussian, noise salt & pepper, dan noise speckle.

(3) Tuliskan teknik-teknik pembangkitan noise uniform, noise gaussian, noise salt & pepper, dan noise speckle.

(4) Mengapa filter rata-rata merupakan Low Pass Filter?

(5) Sebutkan macam-macam filter untuk keperluan reduksi noise.

## 7.4. PERCOBAAN

Percobaan yang dilakukan antara lain adalah:

(1) Membangkitkan Noise Uniform

(2) Membangkitkan Noise Gaussian

(3) Mereduksi Noise Dengan Filter Rata-rata.

### 7.4.1. Percobaan 1 : Membangkitkan Noise Uniform

1. Cara Membangkitkan Noise Uniform
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Noise1
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan Generate Uniform Noise
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file Noise1View.cpp
   - Tambahkan program untuk mean filtering gambar seperti dibawah ini

```
void CNoise1View::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}

}

// merubah data pixel ke RGB
void WarnaToRGB(long int warna,int *Red, int *Green, int *Blue)
{
	*Red = warna & 0x000000FF;
	*Green = (warna & 0x0000FF00) >> 8;
```

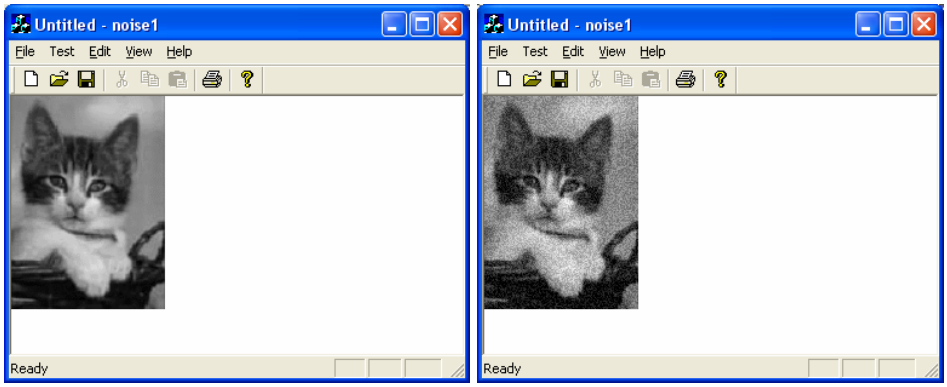
```
	*Blue = (warna & 0x00FF0000) >> 16;
}

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

// Menampilkan gambar hasil dari open file
void CNoise1View::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	long int w,w1;
	int i,j,r,g,b,x;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	//pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

	// Proses RGB to GRAY-SCALE
	for(i=0;i<210;i++)
		for(j=0;j<250;j++) {
			w=dcMem.GetPixel(j,i);
			WarnaToRGB(w,&r,&g,&b);
			x=int((r+g+b)/3);
			w1=RGBToWarna(x,x,x);
			dcMem.SetPixel(j,i,w1);
		}
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

}

void CNoise1View::OnTestGennoise()
{
	// TODO: Add your command handler code here

	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j,r,g,b,nr,x;
	long int w,w1;

	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);

	for(i=0;i<210;i++)
		for(j=0;j<250;j++) {
			w=dcMem.GetPixel(j,i);
```

```
                    WarnaToRGB(w,&r,&g,&b);
                    x=int((r+g+b)/3);
                    //Membangkitkan noise uniform
                    nr=64*rand()/RAND_MAX-32;
                    x=x+nr;
                    if(x>255) x=255;
                    if(x<0) x=0;
                    w1=RGBToWarna(x,x,x);
                    dcMem.SetPixel(j,i,w1);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file
   - Buka file MeanFilterView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CMeanFilterDoc* GetDocument();
        void LoadGambar(void);
// Operations
public:
        CString name;
        CBitmap m_bmpBitmap;
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu : Test->GenerateUniformNoise -> hasilnya seperti gambar 7.13




Gambar 7.13. Hasil sebelum dan setelah pembangkitan noise uniform

7.4.2. Percobaan 2 : Membangkitkan Noise Gaussian

1. Cara Membangkitkan Noise Gaussian
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan Noise2
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan Generate Uniform Gaussian
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file Noise2View.cpp
   - Tambahkan program untuk mean filtering gambar seperti dibawah ini

```
void CNoise2View::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}

}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

// Menampilkan gambar hasil dari open file
void CNoise2View::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	long int w,w1;
	int i,j,r,g,b,x;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
```

```
        if(hBitmap)
        {
                if(m_bmpBitmap.DeleteObject())
                        m_bmpBitmap.Detach();
                m_bmpBitmap.Attach(hBitmap);
        }
        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);
        //pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

        // Proses RGB to GRAY-SCALE
        for(i=0;i<210;i++)
                for(j=0;j<250;j++) {
                        w=dcMem.GetPixel(j,i);
                        WarnaToRGB(w,&r,&g,&b);
                        x=int((r+g+b)/3);
                        w1=RGBToWarna(x,x,x);
                        dcMem.SetPixel(j,i,w1);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

}

void CNoise2View::OnTestGennoise()
{
        // TODO: Add your command handler code here

        CDC* pDC = GetDC();
        CDC dcMem;
        int i,j,r,g,b,nr,x;
        long int w,w1;

        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);

        for(i=0;i<210;i++)
                for(j=0;j<250;j++) {
                        w=dcMem.GetPixel(j,i);
                        WarnaToRGB(w,&r,&g,&b);
                        x=int((r+g+b)/3);
                        //Membangkitkan noise uniform
                        nr=64*rand()/RAND_MAX-32;
                        x=x+nr;
                        if(x>255) x=255;
                        if(x<0) x=0;
                        w1=RGBToWarna(x,x,x);
                        dcMem.SetPixel(j,i,w1);
                }
        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

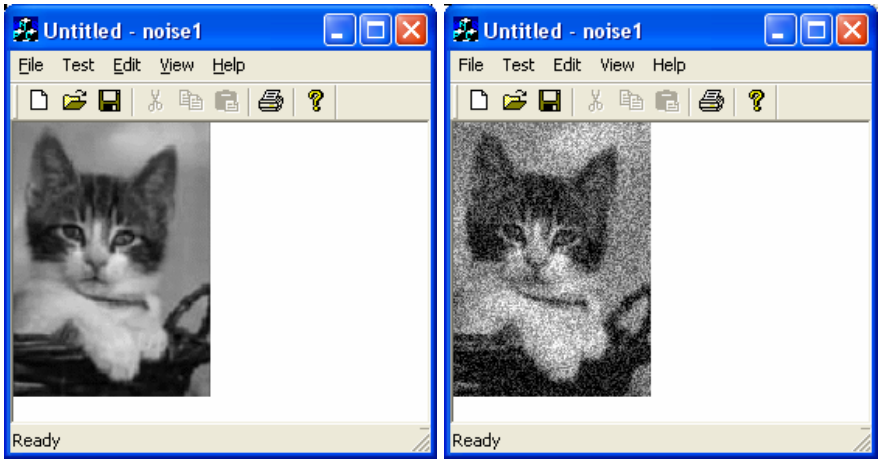
2. Menambah header file

   - Buka file MeanFilterView.h

- Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CMeanFilterDoc* GetDocument();
        void LoadGambar(void);
// Operations
public:
        CString name;
        CBitmap m_bmpBitmap;
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu : Test->GenerateUniformGaussian -> hasilnya seperti gambar 7.14




Gambar 7.14. Hasil sebelum dan setelah pembangkitan noise uniform

### 7.4.3. Percobaan 3 : Mereduksi Noise Dengan Filter Rata-rata

1. Cara mean filtering gambar
   - Buat aplikasi AppWizard seperti pada praktikum 1 dan beri nama project dengan MeanFilter
   - Buat Menu seperti pada praktikum 2 dengan tambahan Test sedangkan submenunya OpenFile dan MeanFilter
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file MeanFilterView.cpp

- Tambahkan program untuk mean filtering gambar seperti dibawah ini

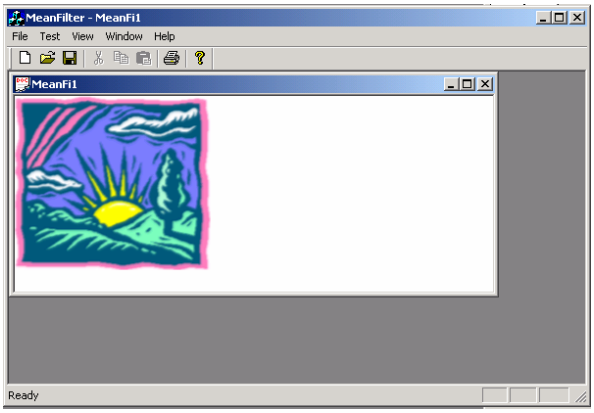
```
/////////////////////////////////////////////////////////////////////////////
// CMeanFilterView message handlers

void CMeanFilterView::OnTestOpenfile()
{
	// TODO: Add your command handler code here
	static char BASED_CODE szFilter[]="Bitmap Files (*.bmp)|*.bmp||";

	CFileDialog m_ldFile(TRUE, "*.bmp", name,
	OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
	if(m_ldFile.DoModal()==IDOK)
	{
		name=m_ldFile.GetPathName();
		LoadGambar();
	}

}

// Menampilkan gambar hasil dari open file
void CMeanFilterView::LoadGambar(void)
{
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(), name,
	IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
				m_bmpBitmap.Detach();
			m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
```

```
	return(Red+(Green<<8)+(Blue<<16));
}

void CMeanFilterView::OnTestMeanfilter()
{
	// TODO: Add your command handler code here
	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j,k,l,r,g,b;
	int resultr,resultg,resultb;
	long int w, mat[3][3];
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	for(i=0;i<200;i++)
		for(j=0;j<200;j++)
		{
			mat[0][0]=dcMem.GetPixel(j-1,i-1);
			mat[0][1]=dcMem.GetPixel(j,i-1);
			mat[0][2]=dcMem.GetPixel(j+1,i-1);
			mat[1][0]=dcMem.GetPixel(j-1,i);
			mat[1][1]=dcMem.GetPixel(j,i);
			mat[1][2]=dcMem.GetPixel(j+1,i);
			mat[2][0]=dcMem.GetPixel(j-1,i+1);
			mat[2][1]=dcMem.GetPixel(j,i+1);
			mat[2][2]=dcMem.GetPixel(j+1,i+1);
			resultr=0;resultg=0;resultb=0;
			for(k=0;k<3;k++)
				for(l=0;l<3;l++)
				{
					WarnaToRGB(mat[k][l],&r,&g,&b);
					resultr=resultr+r;
					resultg=resultg+g;
					resultb=resultb+b;
				}
			resultr=resultr/9;
			resultg=resultg/9;
			resultb=resultb/9;
			if(resultr>255)resultr=255;
			if(resultg>255)resultg=255;
			if(resultb>255)resultb=255;
			w=RGBToWarna(resultr,resultg,resultb);
			dcMem.SetPixel(j,i,w);
		}
	pDC->BitBlt(0,0,200,200,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file
   - Buka file MeanFilterView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
        CMeanFilterDoc* GetDocument();
        void LoadGambar(void);
// Operations
public:
        CString name;
        CBitmap m_bmpBitmap;
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu : Test->MeanFilter -> hasilnya seperti gambar 7.15




Gambar 7.15. Hasil Filter rata-rata

## 7.5. LAPORAN RESMI

(1) Tuliskan karakteristik dari noise uniform, noise gaussian, noise salt & pepper, dan noise speckle.

(2) Tuliskan teknik-teknik pembangkitan noise uniform, noise gaussian, noise salt & pepper, dan noise speckle.

(3) Dengan menggunakan program pembangkitan noise gaussian di atas, hitung SNR untuk prosentase noise 10%, 20%, 30% sampai dengan 90%, dan gambarkan hubungan prosentase noise dengan nilai SNR.

(4) Dengan menggunakan program pembangkitan noise salt & pepper di atas, hitung SNR untuk prosentase noise 10%, 20%, 30% sampai dengan 90%, dan gambarkan hubungan prosentase noise dengan nilai SNR.

(5) Dengan menggunakan program pembangkitan noise speckle di atas, hitung SNR untuk prosentase noise 10%, 20%, 30% sampai dengan 90%, dan gambarkan hubungan prosentase noise dengan nilai SNR.

(6) Dengan program reduksi noise rata-rata menggunakan filter rata-rata 3x3, hitunglah perbaikan SNR bila noise dibangkitkan dengan prosentase noise 5%, 10%, 15%, 20%, sampai dengan 50%. (Gunakan program dari latihan no.1)

(7) Dengan program reduksi noise gaussian menggunakan filter gaussian 3x3, hitunglah perbaikan SNR bila noise dibangkitkan dengan prosentase noise 5%, 10%, 15%, 20%, sampai dengan 50%. (Gunakan program dari latihan no.2)

**LATIHAN:**

(1) Tuliskan listing program untuk mereduksi noise dengan filter rata-rata menggunakan teknik konvolusi.

(2) Tuliskan listing program untuk mereduksi noise dengan filter gaussian menggunakan teknik konvolusi.

Praktikum

8

# Deteksi Tepi (Edge Detection)

## 8.1. TUJUAN:

1. Mahasiswa dapat memahami prinsip-prinsip deteksi tepi pada citra
2. Mahasiswa dapat melakukan deteksi tepi pada citra dengan metode Robert
3. Mahasiswa dapat melakukan deteksi tepi pada citra dengan metode Prewitt
4. Mahasiswa dapat melakukan deteksi tepi pada citra dengan metode Sobel

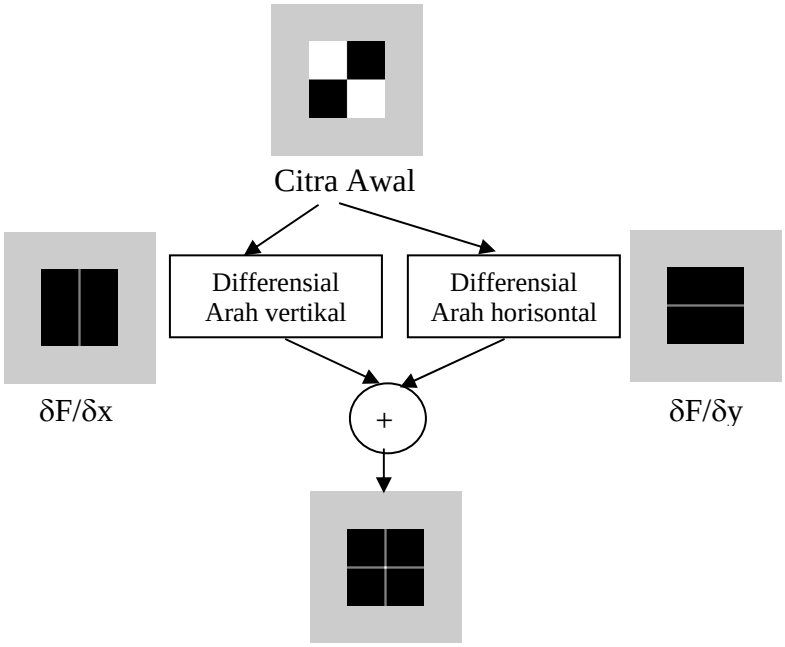
## 8.2. DASAR TEORI:

### 8.2.1. Prinsip-Prinsip Deteksi Tepi

Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah :

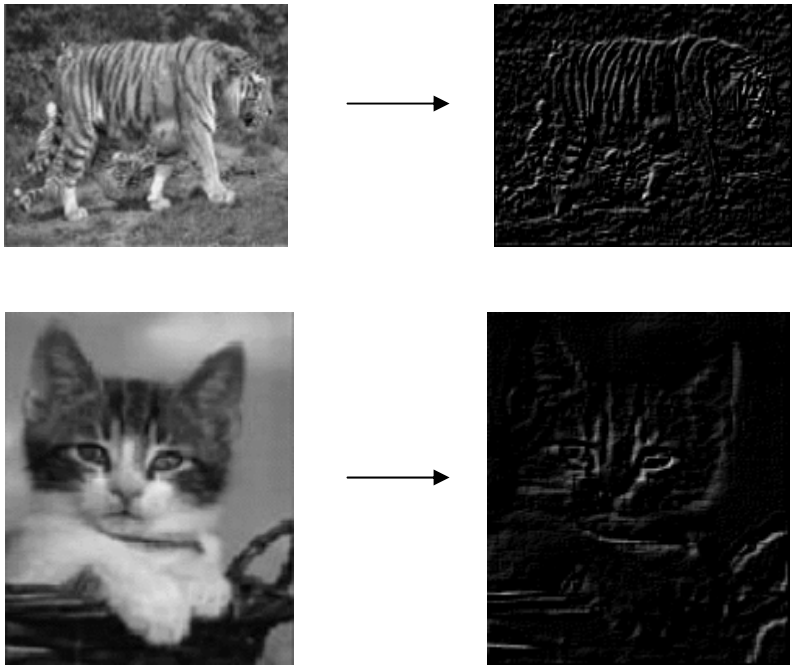
- Untuk menandai bagian yang menjadi detail citra
- Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena error atau adanya efek dari proses akuisisi citra

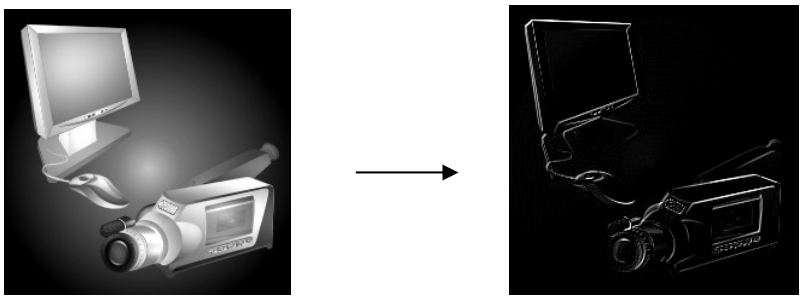
Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar 8.1 berikut ini menggambarkan bagaimana tepi suatu gambar diperoleh.




Gambar 8.1. Proses Deteksi Tepi Citra

Perhatikan hasil deteksi dari beberapa citra menggunakan model differensial di atas:





Gambar 8.2. Hasil beberapa deteksi tepi

Pada gambar 8.2. terlihat bahwa hasil deteksi tepi berupa tepi-tepi dari suatu gambar. Bila diperhatikan bahwa tepi suatu gambar terletak pada titik-titik yang memiliki perbedaan tinggi. Berdasarkan prinsip-prinsip filter pada citra maka tepi suatu gambar dapat diperoleh menggunakan High Pass Filter (HPF), yang mempunyai karakteristik:

$$\sum_{y}\sum_{x} H(x,y) = 0$$

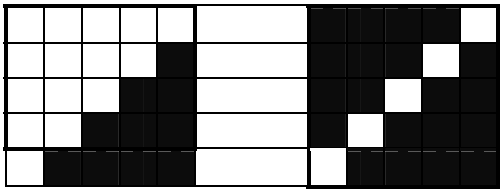
Contoh:
Diketahui fungsi citra f(x,y) sebagai berikut:

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |

Dengan menggunakan filter : $H(x,y) = \begin{bmatrix} -1 & 1 \end{bmatrix}$
Maka Hasil filter adalah :

| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |

Bila digambarkan maka proses filter di atas mempunyai masukan dan keluaran sebagai berikut



Catatan:

Untuk mencoba perhitungan di atas dapat dilakukan dengan cara manual menggunakan perhitungan konvolusi yang telah dibahas pada bab 5, atau dengan memanfaatkan program konvolusi.

Macam-macam metode untuk proses deteksi tepi ini, antara lain:

(1) Metode Robert
(2) Metode Prewitt
(3) Metode Sobel

Metode yang banyak digunakan untuk proses deteksi tepi adalah metode Robert, Prewitt dan Sobel, Gonzalez[1].

### 8.2.1. Metode Robert

Metode Robert adalah nama lain dari teknik differensial yang dikembangkan di atas, yaitu differensial pada arah horisontal dan differensial pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan differensial. Teknik konversi biner yang disarankan adalah konversi biner dengan meratakan distribusi warna hitam dan putih [5], seperti telah dibahas pada bab 3. Metode Robert ini juga disamakan dengan teknik DPCM (*Differential Pulse Code Modulation*)

Kernel filter yang digunakan dalam metode Robert ini adalah:

$$H = \begin{bmatrix} -1 & 1 \end{bmatrix} \quad \text{dan} \quad H = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

### 8.2.2. Metode Prewitt

Metode Prewitt merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF.

Kernel filter yang digunakan dalam metode Prewitt ini adalah:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad H = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

### 8.2.3. Metode Sobel

Metode Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi.

Kernel filter yang digunakan dalam metode Sobel ini adalah:

$$H = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad H = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

## 8.3. TUGAS PENDAHULUAN

(1) Tuliskan tujuan praktikum

(2) Tuliskan prinsip-prinsip deteksi tepi pada citra

(3) Tuliskan beberapa macam metode deteksi tepi yang umum digunakan

(4) Tuliskan filter mask (=H) yang digunakan pada masing-masing metode deteksi tepi

(5) Diketahui sebuah fungsi citra biner f(x,y) sebagai berikut :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Dengan metode Robert, Prewitt, dan Sobel dapatkan tepi citra diatas.

## 8.4. PERCOBAAN

Percobaan yang dilakukan antara lain adalah:

(1) Deteksi Tepi dengan Metode Robert

(2) Deteksi Tepi dengan Metode Prewitt

(3) Deteksi Tepi dengan Metode Sobel

### 8.4.1. PERCOBAAN 1: Deteksi Tepi dengan Metode Robert

**Percobaan ini melakukan proses deteksi tepi dengan kernel filter:**

$$H(x,y) = \begin{bmatrix} -1 & 1 \end{bmatrix}$$

**8. . .** Cara deteksi tepi :

- Buat aplikasi AppWizard seperti pada percobaan 1.1 dan beri nama project dengan Edge
- Buat Menu seperti pada percobaan 1.2 dengan tambahan Test sedangkan submenunya Load Gambar pada sub menu ini gambar RGB sekaligus dikonversi menjadi gambar gray scale, dan sub menu Edge Robert
- Untuk mengedit isi program tekan tombol Edit Code atau buka file EdgeView.cpp
- Tambahkan program untuk deteksi tepi seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////
///////////
// CEdgeView message handlers

void CedgeView:: OnTestLoadgambar ()
{
      // TODO: Add your command handler code here
      static char BASED_CODE szFilter[]="Bitmap Files
(*.bmp)|*.bmp||";

      CfileDialog m_ldFile(TRUE, "*.bmp", name,
      OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
      if(m_ldFile.DoModal()==IDOK)
      {
            name=m_ldFile.GetPathName();
            LoadGambar();
      }
}



// Menampilkan gambar hasil dari open file
void CedgeView::LoadGambar(void)
{
```

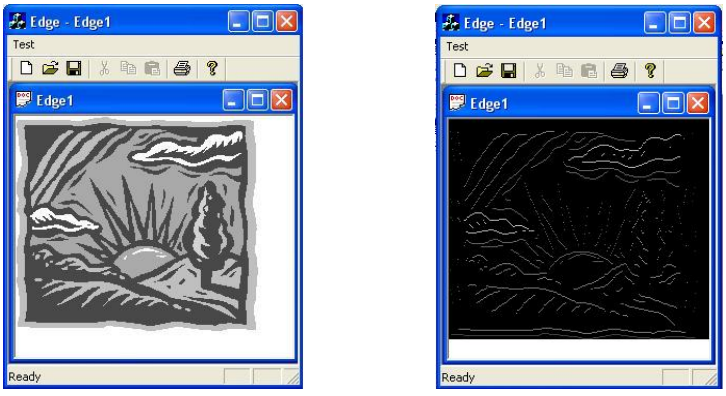
```
	CDC* pDC = GetDC();
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
	name, IMAGE_BITMAP, 0, 0,
	LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	// Proses RGB to GRAY-SCALE
	for(i=0;i<210;i++)
		for(j=0;j<250;j++) {
			w=dcMem.GetPixel(j,i);
			WarnaToRGB(w,&r,&g,&b);
			x=int((r+g+b)/3);
			w1=RGBToWarna(x,x,x);
			dcMem.SetPixel(j,i,w1);
		}
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CedgeView::On TestEdgerobert ()
{
	// TODO: Add your command handler code here
	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j,r,g,b;
	int resultr,resultg,resultb;
	long int w,mat[1][2];
	int h[1][2],hr,hg,hb;

	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);


	// Proses Konvolusi
	int nh=2; // Menyatakan ukuran filter
	// Penentuan kernel filter
	h[0][0]=-1; h[0][1]=1;
	int u=0;//Menyatakan ukuran filter •orizontal
	for(i=0;i<210;i++)
		for(j=0;j<250;j++){
			mat[0][0]=dcMem.GetPixel(j,i);
			mat[0][1]=dcMem.GetPixel(j,i+1);
			hr=0;hg=0;hb=0;
				//Menyatakan ukuran filter vertikal
```

```
                        for(int v=0;v<2;v++){
                              WarnaToRGB(mat[u][v],&r,&g,&b);
                              hr+=r*h[u][v];
                              hg+=g*h[u][v];
                              hb+=b*h[u][v];
                        }
                  resultr=hr;
                  resultg=hg;
                  resultb=hb;
                  if(resultr>255)resultr=255;
                  if(resultg>255)resultg=255;
                  if(resultb>255)resultb=255;
                  w=RGBToWarna(resultr,resultg,resultb);
                  dcMem.SetPixel(j,i,w);
            }
      pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file
   - Buka file EdgeView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
      CedgeDoc* GetDocument();
      Cstring name;
      Cbitmap m_bmpBitmap;

// Operations
public:
      void LoadGambar(void);
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->Load Gambar-> pilih salah satu gambar misalnya gambar.bmp hasilnya seperti gambar 8.3 a.
   - Pilih menu: Test->Edge Robert hasilnya seperti gambar 8.3. b.




Gambar 8.3.

a. Gambar Asal (gambar.bmp) Langsung -> Gray Scale

b. Contoh hasil deteksi tepi Robert

## 8.4.2. PERCOBAAN 2 : Deteksi Tepi dengan Metode Prewitt

**Percobaan ini melakukan proses deteksi tepi dengan kernel filter:**

$$H_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad H_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

1. Cara deteksi tepi :
   - Buat aplikasi AppWizard seperti pada percobaan 1.1 dan beri nama project dengan Edge
   - Buat Menu seperti pada percobaan 1.2 dengan tambahan Test sedangkan submenunya Load Gambar pada sub menu ini gambar RGB sekaligus dikonversi menjadi gambar gray scale, dan sub menu Edge Prewitt
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file EdgeView.cpp
   - Tambahkan program untuk deteksi tepi seperti dibawah ini

```
////////////////////////////////////////////////////////////////////
///////////
// CEdgeView message handlers
void CEdgeView::OnTestLoadgambar()
{
      // TODO: Add your command handler code here
      static char BASED_CODE szFilter[]="Bitmap Files
(*.bmp)|*.bmp||";

      CFileDialog m_ldFile(TRUE, "*.bmp", name,
      OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
      if(m_ldFile.DoModal()==IDOK)
      {
            name=m_ldFile.GetPathName();
            LoadGambar();
      }
}

// Menampilkan gambar hasil dari open file
void CEdgeView::LoadGambar(void)
{
      CDC* pDC = GetDC();
```

```
	CDC dcMem;
	HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
	name, IMAGE_BITMAP, 0, 0,
	LR_LOADFROMFILE|LR_CREATEDIBSECTION);
	if(hBitmap)
	{
		if(m_bmpBitmap.DeleteObject())
			m_bmpBitmap.Detach();
		m_bmpBitmap.Attach(hBitmap);
	}
	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);
	// Proses RGB to GRAY-SCALE
	for(i=0;i<210;i++)
		for(j=0;j<250;j++) {
			w=dcMem.GetPixel(j,i);
			WarnaToRGB(w,&r,&g,&b);
			x=int((r+g+b)/3);
			w1=RGBToWarna(x,x,x);
			dcMem.SetPixel(j,i,w1);
		}
	pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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



//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
	return(Red+(Green<<8)+(Blue<<16));
}

void CEdgeView::OnTestEdgeprewitt()
{
	// TODO: Add your command handler code here
	CDC* pDC = GetDC();
	CDC dcMem;
	int i,j,r,g,b;
	int resultr,resultg,resultb;

	long int w,mat[3][3];
	int hy[3][3],hx[3][3],Gxy[3][3],hr,hg,hb;

	dcMem.CreateCompatibleDC(pDC);
	dcMem.SelectObject(&m_bmpBitmap);

	// Proses Konvolusi
	int nh=3; // Menyatakan ukuran filter

	// Penentuan kernel filter hy
	hy[0][0]=-1; hy[0][1]=0; hy[0][2]=1;
	hy[1][0]=-1; hy[1][1]=0; hy[1][2]=1;
	hy[2][0]=-1; hy[2][1]=0; hy[2][2]=1;

	// Penentuan kernel filter hx
	hx[0][0]=-1; hx[0][1]=-1; hx[0][2]=-1;
```

```
        hx[1][0]=0; hx[1][1]=0; hx[1][2]=0;
        hx[2][0]=1; hx[2][1]=1; hx[2][2]=1;

        for(i=0;i<210;i++)
              for(j=0;j<250;j++){
                    mat[0][0]=dcMem.GetPixel(j-1,i-1);
                    mat[0][1]=dcMem.GetPixel(j,i-1);
                    mat[0][2]=dcMem.GetPixel(j+1,i-1);
                    mat[1][0]=dcMem.GetPixel(j-1,i);
                    mat[1][1]=dcMem.GetPixel(j,i);
                    mat[1][2]=dcMem.GetPixel(j+1,i);
                    mat[2][0]=dcMem.GetPixel(j-1,i+1);
                    mat[2][1]=dcMem.GetPixel(j,i+1);
                    mat[2][2]=dcMem.GetPixel(j+1,i+1);
                    hr=0;hg=0;hb=0;
                    for(int u=0;u<nh;u++)
                          for(int v=0;v<nh;v++){
                                WarnaToRGB(mat[u][v],&r,&g,&b);
                                Gxy[u][v]=hy[u][v]+hx[u][v];
                                hr+= r * Gxy[u][v];
                                hg+= g * Gxy[u][v];
                                hb+= b * Gxy[u][v];
                          }
                    resultr=abs(hr);
                    resultg=abs(hg);
                    resultb=abs(hb);

                    if(resultr>255)resultr=255;
                    if(resultg>255)resultg=255;
                    if(resultb>255)resultb=255;
                    w=RGBToWarna(resultr,resultg,resultb);
                    dcMem.SetPixel(j,i,w);
              }

        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}
```

2. Menambah header file
   - Buka file EdgeView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
      CEdgeDoc* GetDocument();
      CString name;
      CBitmap m_bmpBitmap;

// Operations
public:
      void LoadGambar(void);
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Edge hasilnya seperti gambar 8.4.



a) b)

Gambar 8.4.

a). Gambar Asal (gambar.bmp) Langsung -> Gray Scale

b). Contoh hasil deteksi tepi Robert

Gambar 8.4. Contoh hasil deteksi tepi Prewitt

## 8.4.3. PERCOBAAN 3 : Deteksi Tepi dengan Metode Sobel

**Percobaan ini melakukan proses deteksi tepi dengan kernel filter:**

$$H_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{dan} \quad H_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

1. Cara deteksi tepi :
   - Buat aplikasi AppWizard seperti pada percobaan 1.1 dan beri nama project dengan Edge
   - Buat Menu seperti pada percobaan 1.2 dengan tambahan Test sedangkan submenunya Load Gambar pada sub menu ini gambar RGB sekaligus dikonversi menjadi gambar gray scale, dan sub menu Edge Sobel
   - Untuk mengedit isi program tekan tombol Edit Code atau buka file EdgeView.cpp
   - Tambahkan program untuk deteksi tepi seperti dibawah ini

```
/////////////////////////////////////////////////////////////////////////////
// CEdgeView message handlers

void CEdgeView::OnTestLoadambar()
{
	// TODO: Add your command handler code here
```

```
      static char BASED_CODE szFilter[]="Bitmap Files
(*.bmp)|*.bmp||";

      CFileDialog m_ldFile(TRUE, "*.bmp", name,
      OFN_HIDEREADONLY|OFN_OVERWRITEPROMPT, szFilter);
      if(m_ldFile.DoModal()==IDOK)
      {
            name=m_ldFile.GetPathName();
            LoadGambar();
      }
}

// Menampilkan gambar hasil dari open file
void CEdgeView::LoadGambar(void)
{
      CDC* pDC = GetDC();
      CDC dcMem;
      HBITMAP hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),
      name, IMAGE_BITMAP, 0, 0,
      LR_LOADFROMFILE|LR_CREATEDIBSECTION);


      if(hBitmap)
      {
            if(m_bmpBitmap.DeleteObject())
                  m_bmpBitmap.Detach();
            m_bmpBitmap.Attach(hBitmap);
      }
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);

      // Proses RGB to GRAY-SCALE
      for(i=0;i<210;i++)
            for(j=0;j<250;j++) {
                  w=dcMem.GetPixel(j,i);
                  WarnaToRGB(w,&r,&g,&b);
                  x=int((r+g+b)/3);
                  w1=RGBToWarna(x,x,x);
                  dcMem.SetPixel(j,i,w1);
            }
      pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);
}

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

//merubah RGB ke data pixel
long int RGBToWarna(int Red, int Green, int Blue)
{
      return(Red+(Green<<8)+(Blue<<16));
}

void CedgeView::OnTestEdge()
{
      // TODO: Add your command handler code here
      CDC* pDC = GetDC();
      CDC dcMem;
      int i,j,r,g,b;
      int resultr,resultg,resultb;
```

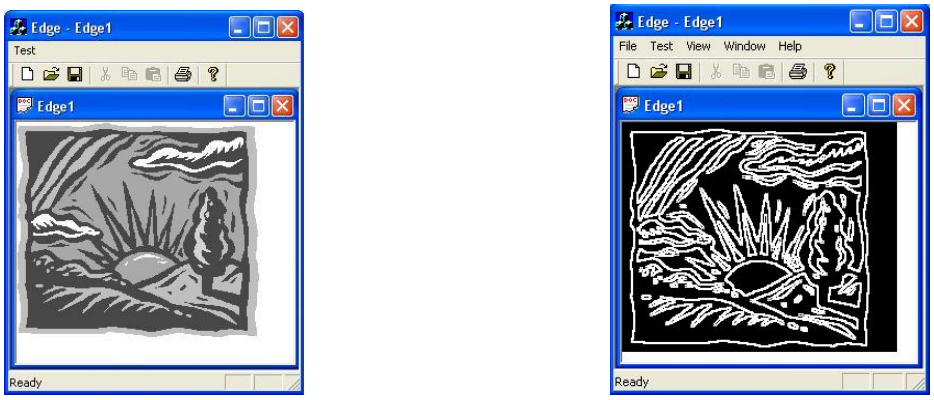
```
        long int w,mat[3][3];
        int hy[3][3],hx[3][3],Gxy[3][3],hr,hg,hb;

        dcMem.CreateCompatibleDC(pDC);
        dcMem.SelectObject(&m_bmpBitmap);

        // Proses Konvolusi
        int nh=3; // Menyatakan ukuran filter

        // Penentuan kernel filter hy
        hy[0][0]=-1; hy[0][1]=0; hy[0][2]=1;
        hy[1][0]=-1; hy[1][1]=0; hy[1][2]=1;
        hy[2][0]=-1; hy[2][1]=0; hy[2][2]=1;

        // Penentuan kernel filter hx
        hx[0][0]=-1; hx[0][1]=-1; hx[0][2]=-1;
        hx[1][0]=0; hx[1][1]=0; hx[1][2]=0;
        hx[2][0]=1; hx[2][1]=1; hx[2][2]=1;

        for(i=0;i<210;i++)
              for(j=0;j<250;j++){
                    mat[0][0]=dcMem.GetPixel(j-1,i-1);
                    mat[0][1]=dcMem.GetPixel(j,i-1);
                    mat[0][2]=dcMem.GetPixel(j+1,i-1);
                    mat[1][0]=dcMem.GetPixel(j-1,i);
                    mat[1][1]=dcMem.GetPixel(j,i);
                    mat[1][2]=dcMem.GetPixel(j+1,i);
                    mat[2][0]=dcMem.GetPixel(j-1,i+1);
                    mat[2][1]=dcMem.GetPixel(j,i+1);
                    mat[2][2]=dcMem.GetPixel(j+1,i+1);
                    hr=0;hg=0;hb=0;
                    for(int u=0;u<nh;u++)
                          for(int v=0;v<nh;v++){
                                WarnaToRGB(mat[u][v],&r,&g,&b);
                                Gxy[u][v]=hy[u][v]+hx[u][v];
                                hr+= r * Gxy[u][v];
                                hg+= g * Gxy[u][v];
                                hb+= b * Gxy[u][v];
                          }
                    resultr=abs(hr);
                    resultg=abs(hg);
                    resultb=abs(hb);
                    if(resultr>255)resultr=255;
                    if(resultg>255)resultg=255;
                    if(resultb>255)resultb=255;
                    w=RGBToWarna(resultr,resultg,resultb);
                    dcMem.SetPixel(j,i,w);
              }

        pDC->BitBlt(0,0,250,210,&dcMem,0,0,SRCCOPY);

}
```

2. Menambah header file
   - Buka file EdgeView.h
   - Tambahkan program seperti dibawah ini

```
// Attributes
public:
      CEdgeDoc* GetDocument();
      CString name;
```

```
        CBitmap m_bmpBitmap;

// Operations
public:
        void LoadGambar(void);
```

3. Cara menjalankan program
   - Pilih menu : Build->Execute (!)
   - Pilih menu : Test->OpenFile -> pilih salah satu gambar misalnya gambar.bmp
   - Pilih menu: Test->Edge hasilnya seperti gambar 8.5.




a) b)

Gambar 8.5.

a). Gambar Asal (gambar.bmp) Langsung -> Gray Scale

b). Contoh hasil deteksi tepi Sobel

## 8.4. LAPORAN RESMI

### 8.4.1. Hasil Percobaan

(1) Tuliskan prinsip-prinsip deteksi tepi pada citra

(2) Tuliskan beberapa macam metode deteksi tepi yang umum digunakan

(3) Tuliskan filter mask (=H) yang digunakan pada masing-masing metode deteksi tepi

(4) Gunakan citra dengan bentuk yang menonjol misalnya : gedung1, gedung2, pesawat1, pesawat2, seperti di gambar 8.6. Ttampilkan hasil deteksi tepi dari keempat gambar ini pada semua metode deteksi tepi yang sudah dilakakukan di percobaan. Amati apa yang dapat anda simpulkan.

Gambar 8.6. Citra dengan Bentuk yang Jelas

(3) Gunakan citra dengan bentuk yang seperti noise misalnya : pohon semak1, pohon semak2, sungai1, sungai 2 seperti di gambar 8.7. Ttampilkan hasil deteksi tepi dari keempat gambar ini pada semua metode deteksi tepi yang sudah dilakakukan di percobaan. Amati apa yang dapat anda simpulkan.

Gambar 8.7. Citra dengan Bentuk Noise

(5) Dengan menggunakan program di atas, ubahlah filter mask deteksi tepi yang digunakan pada percobaan dengan filter mask di bawah dan jelaskan apa perbedaan hasil deteksi tepi yang terjadi. Gunakan citra pada gambar 8.6 dan citra pada gambar 8.7. untuk pengamatan.

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ dan } h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

## 8.4.2. Latihan

(1) Cobalah memodifikasi program deteksi tepi di atas agar dapat digunakan untuk ukuran filter yang bebas.

(2) Buatlah program untuk melakukan deteksi tepi citra dengan metode laplacian dengan filter kernel :

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \text{ dan } h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$