Praktikum 9

Reduksi Noise

POKOK BAHASAN :

- ✓ Noise
- ✓ Low Pass Filter

TUJUAN BELAJAR :

Setelah melakukan praktikum pada bab ini, mahasiswa diharapkan mampu:

- ✓ Membuat program generate bermacam noise
- ✓ Membuat program mereduksi noise dengan low pass filter

DASAR TEORI :

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 9.1 berikut ini:

(a) (b) (c)

Gambar 9.1. 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.

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 rand() dalam C 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 9.2. Bebarapa contoh noise uniform dengan prosentase 10%, 20%, 30%, 50%, 75% dan 90%.

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

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 9.4. 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:

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

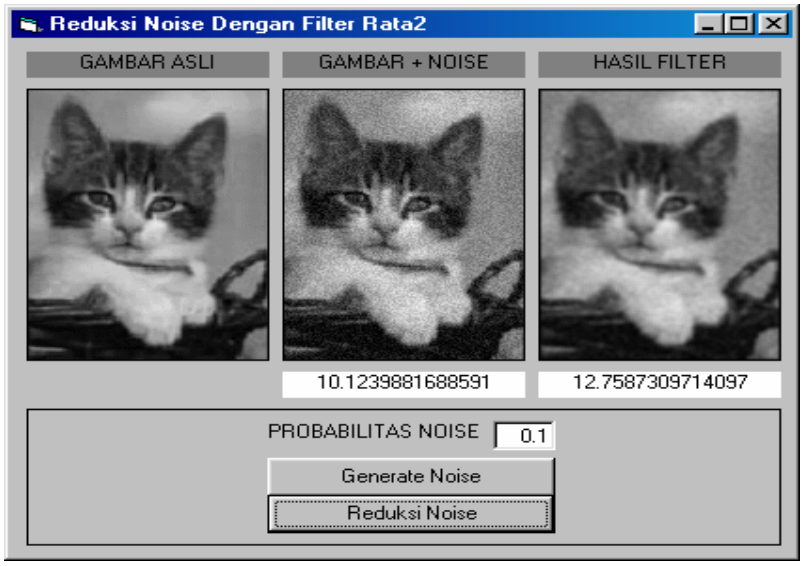
Filter rata-rata berukuran 3x3 adalah:

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

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 9.5, 9.6, dan 9.7 berikut.



Gambar 9.5. Hasil filter rata-rata ukuran 3x3

Gambar 9.6. Hasil filter rata-rata ukuran 5x5

Gambar 9.7. 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.

Reduksi Noise Menggunakan Filter Gaussian

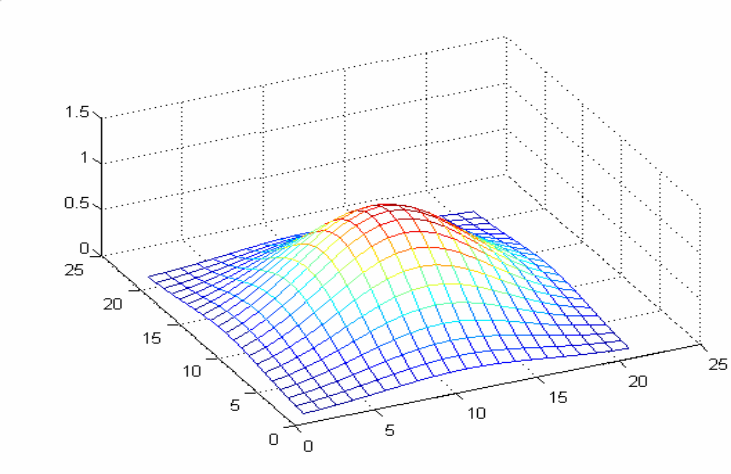
Filter lain yang banyak digunakan dalam mereduksi noise selain filter rata-rata adalah filter gaussian. Filter gaussian ini sebenarnya hampir sama dengan filter rata-rata

hanya ada nilai bobot yang tidak rata seperti pada filter rata-rata, tetapi mengikuti fungsi gaussian sebagai berikut:

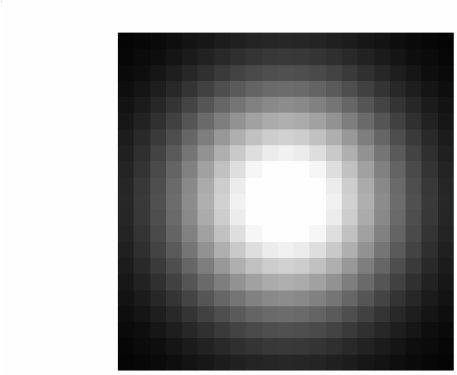
$$G(x,y) = \frac{1}{s\sqrt{\pi}} e^{-\left((x-m_x)^2+(y-m_y)^2\right)/2s}$$

dimana: s adalah sebaran dari fungsi gaussian

(m_x,m_y) adalah titik tengah dari fungsi gaussian



Gambar 9.8 Model Fungsi Gaussian dalam ruang.



Gambar 9.9. Model contour dari filter gaussian.

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

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

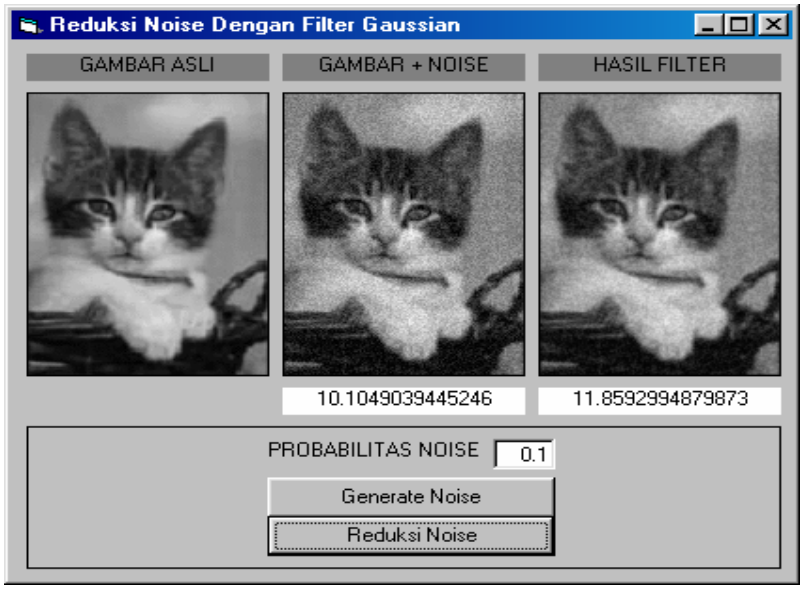
Kernel filter gaussian untuk ukuran 5x5 adalah:

```
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
```

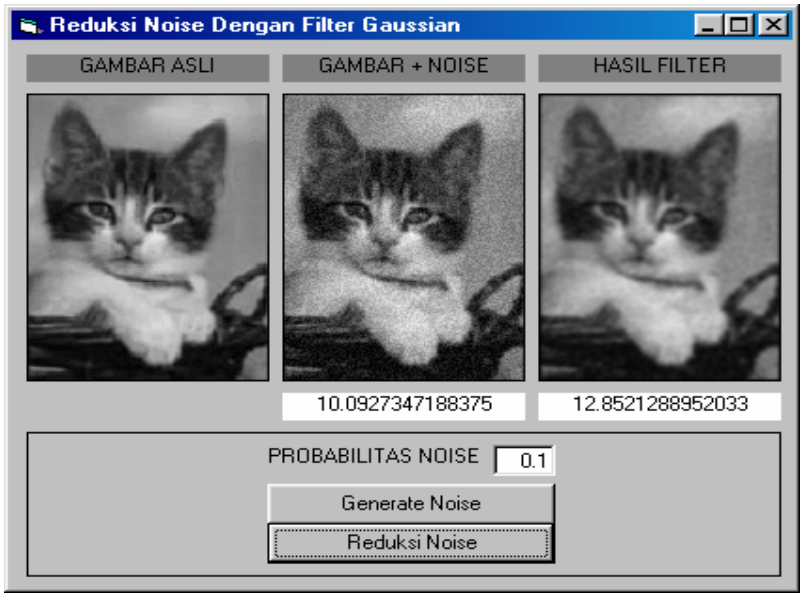
Kernel filter gaussian untuk ukuran 7x7 adalah:

```
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.



Gambar9.10. Hasil filter gaussian dengan ukuran 3x3



Gambar 9.11. Hasil filter gaussian dengan ukuran 5x5

Gambar 9.12. Hasil filter gaussian dengan ukuran 7x7

TUGAS PENDAHULUAN :

(1) Tuliskan tujuan praktikum

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

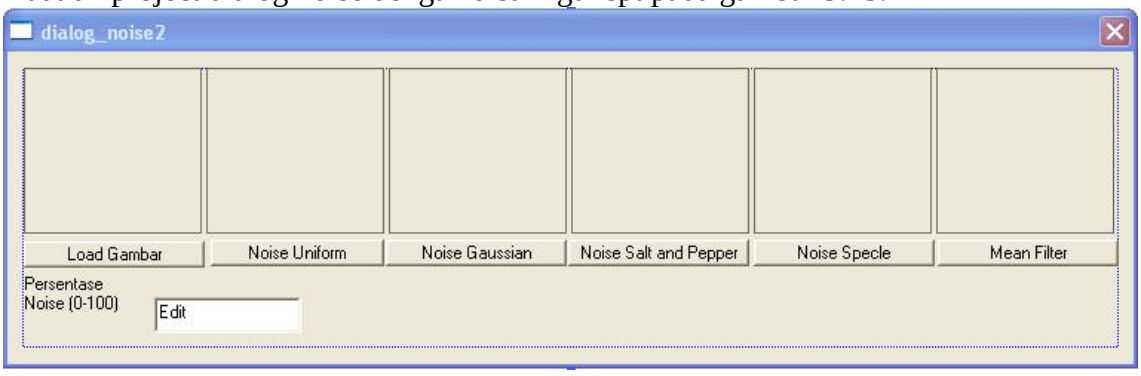
(3) Tuliskan teknik-teknik pembangkitan noise uniform, noise gaussian, noise salt & pepper, dan noise speckle.

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

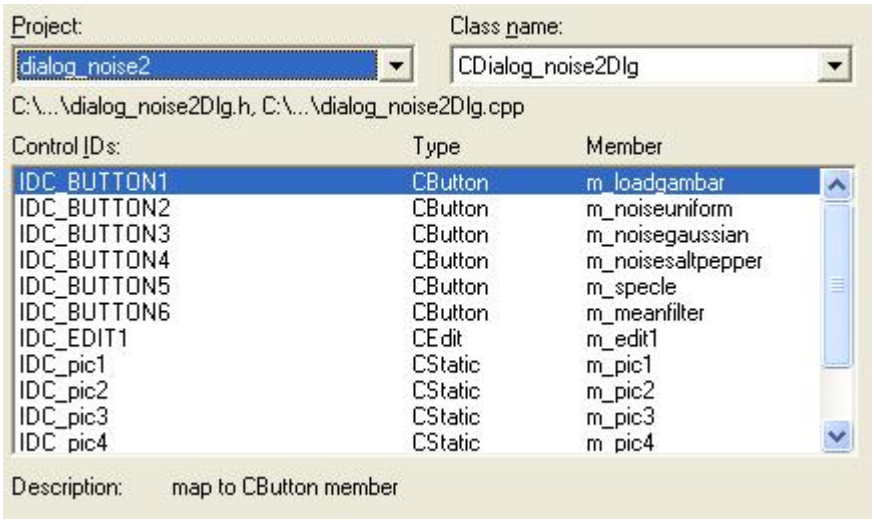
(5) Sebutkan macam-macam filter untuk keperluan reduksi noise.

PERCOBAAN :

Buatlah project dialog noise dengan disain gui spt pada gambar 9.13.



Gambar 9.13. Disain GUI Dialog Noise



Gambar 9.14. Member Variabel Dialog Noise

Fungsi mengubah warna ke rgb

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

Fungsi mengubah rgb ke warna

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

Tambahan pada header file

```
public:
	CBitmap m_bmpBitmap;
```

```
void CDialog_noise2Dlg::OnButton1()
{
      int j,k,gray,red,green,blue;
      long int warna;
      CDC* pDC = m_pic1.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
HBITMAP
hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),"kucing.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);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {
                  warna=dcMem.GetPixel(k,j);
                  WarnaToRGB(warna,&red,&green,&blue
                  gray=(red+green+blue)/3;
                  warna=RGBToWarna(gray,gray,gray);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
                  bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

```
void CDialog_noise2Dlg::OnButton2()
{
      int j,k,gray,red,green,blue,persentase,nr;
      long int warna;
      float fpersen;
      CDC* pDC = m_pic2.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
HBITMAP
hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),"kucing.bmp
",
IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
      if(hBitmap)
      {
            if(m_bmpBitmap.DeleteObject())
                  m_bmpBitmap.Detach();
            m_bmpBitmap.Attach(hBitmap);
      }
      time_t seconds;
      time(&seconds);
      srand((unsigned int) seconds);
      m_pic2.GetClientRect(rect);//
      m_bmpBitmap.GetBitmap(&bm);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      persentase=GetDlgItemInt(IDC_EDIT1);
      fpersen=float (persentase/100.);
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {
                  warna=dcMem.GetPixel(k,j);
                  WarnaToRGB(warna,&red,&green,&blue);
                  gray=(red+green+blue)/3;
                  nr=rand()%256;
                  nr=int (fpersen*nr);
                  gray=gray+nr;
                  if(gray>255) gray=255;
                  if(gray<0) gray=0;
                  warna=RGBToWarna(gray,gray,gray);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
                  bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

```
void CDialog_noise2Dlg::OnButton3()
{
      int j,k,gray,red,green,blue,persentase,nr,sw,x,y;
      long int warna;
      float fpersen;
      CDC* pDC = m_pic3.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
HBITMAP
hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),"kucing.bmp
",IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
      if(hBitmap)
      {
            if(m_bmpBitmap.DeleteObject())
                  m_bmpBitmap.Detach();
            m_bmpBitmap.Attach(hBitmap);
      }
      time_t seconds;
      time(&seconds);
      srand((unsigned int) seconds);
      m_pic3.GetClientRect(rect);//
      m_bmpBitmap.GetBitmap(&bm);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      persentase=GetDlgItemInt(IDC_EDIT1);
      fpersen=float (persentase/100.);
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {     warna=dcMem.GetPixel(k,j);
                  WarnaToRGB(warna,&red,&green,&blue);
                  gray=(red+green+blue)/3;
                  sw = 0;
                  while (sw == 0)
                  {
                        x = int(rand()%256) ;
                        y = rand();
                        if (y < exp(pow(-x,2)))
                              sw = x;
                  }
                  nr=sw;
                  nr=int (fpersen*nr);
                  gray=gray+nr;
                  if(gray>255) gray=255;
                  if(gray<0) gray=0;
                  warna=RGBToWarna(gray,gray,gray);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
                 bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

```
void CDialog_noise2Dlg::OnButton4()
{
      int j,k,gray,red,green,blue,persentase;
      long int warna;
      float fpersen,nr;
      CDC* pDC = m_pic4.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
      HBITMAP
      hBitmap=(HBITMAP)::LoadImage(AfxGetInstanceHandle(),"kuci
      ng.bmp",IMAGE_BITMAP, 0, 0,
      LR_LOADFROMFILE|LR_CREATEDIBSECTION);
      if(hBitmap)
      {
            if(m_bmpBitmap.DeleteObject())
                  m_bmpBitmap.Detach();
            m_bmpBitmap.Attach(hBitmap);
      }
      time_t seconds;
      time(&seconds);
      srand((unsigned int) seconds);
      m_pic4.GetClientRect(rect);//
      m_bmpBitmap.GetBitmap(&bm);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      persentase=GetDlgItemInt(IDC_EDIT1);
      fpersen=float (persentase/100.);
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {
                  warna=dcMem.GetPixel(k,j);
                  WarnaToRGB(warna,&red,&green,&blue);
                  gray=(red+green+blue)/3;
                  nr=float(rand()%256)/100;
                  if (nr<fpersen)
                        gray=255;
                  else
                        gray=int(gray);
                  if(gray>255) gray=255;
                  if(gray<0) gray=0;
                  warna=RGBToWarna(gray,gray,gray);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
                  bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

```
void CDialog_noise2Dlg::OnButton5()
{     int j,k,gray,red,green,blue,persentase;
      long int warna;
      float fpersen,nr;
      CDC* pDC = m_pic5.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
      HBITMAP hBitmap
      =(HBITMAP)::LoadImage(AfxGetInstanceHandle(),"kucing.bmp"
,
      IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE|LR_CREATEDIBSECTION);
      if(hBitmap)
      {
            if(m_bmpBitmap.DeleteObject())
                  m_bmpBitmap.Detach();
            m_bmpBitmap.Attach(hBitmap);
      }
      time_t seconds;
      time(&seconds);
      srand((unsigned int) seconds);
      m_pic5.GetClientRect(rect);//
      m_bmpBitmap.GetBitmap(&bm);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      persentase=GetDlgItemInt(IDC_EDIT1);
      fpersen=float (persentase/100.);
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {
                  warna=dcMem.GetPixel(k,j);
                  WarnaToRGB(warna,&red,&green,&blue);
                  gray=(red+green+blue)/3;
                  nr=float(rand()%256)/100;
                  if (nr<fpersen)
                        gray=0;
                  else
                        gray=(int)(gray);
                  if(gray>255) gray=255;
                  if(gray<0) gray=0;
                  warna=RGBToWarna(gray,gray,gray);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
                  bm.bmWidth,bm.bmHeight,SRCCOPY);//
}
```

```
void CDialog_noise2Dlg::OnButton6()
{     int i,j,k,red,green,blue,resultr,resultg,resultb;
      long int warna,mat[3][3];
      float h[3][3],hr,hg,hb;
      CDC* pDC = m_pic6.GetDC();//
      CDC dcMem;
      CRect rect;//
      BITMAP bm;//
      m_pic6.GetClientRect(rect);//
      m_bmpBitmap.GetBitmap(&bm);//
      dcMem.CreateCompatibleDC(pDC);
      dcMem.SelectObject(&m_bmpBitmap);
      int nh=3; // Menyatakan ukuran filter
      // Penentuan kernel filter
      for(i=0;i<nh;i++)
            for(j=0;j<nh;j++)
                  h[i][j]=(float)1/9;
      for(j=0;j<bm.bmHeight;j++)
            for(k=0;k<bm.bmWidth;k++)
            {
                  mat[0][0]=dcMem.GetPixel(k-1,j-1);
                  mat[0][1]=dcMem.GetPixel(k,j-1);
                  mat[0][2]=dcMem.GetPixel(k+1,j-1);
                  mat[1][0]=dcMem.GetPixel(k-1,j);
                  mat[1][1]=dcMem.GetPixel(k,j);
                  mat[1][2]=dcMem.GetPixel(k+1,j);
                  mat[2][0]=dcMem.GetPixel(k-1,j+1);
                  mat[2][1]=dcMem.GetPixel(k,j+1);
                  mat[2][2]=dcMem.GetPixel(k+1,j+1);
                  hr=0;hg=0;hb=0;
                  //proses konvolusi
                  for(int u=0;u<nh;u++)
                        for(int v=0;v<nh;v++){
                        WarnaToRGB(mat[u][v],&red,&green,&blue);
                              hr+=(float)red*h[u][v];
                              hg+=(float)green*h[u][v];
                              hb+=(float)blue*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;
                  warna=RGBToWarna(resultr,resultg,resultb);
                  dcMem.SetPixel(k,j,warna);
            }
pDC->StretchBlt(0,0,rect.Width(),rect.Height(),&dcMem,0,0,
      bm.bmWidth,bm.bmHeight,SRCCOPY);//

}
```

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, cobalah prosentase noise 10%, 20%, 30% sampai dengan 90%, amati hasil citra bernoise, buatlah kesimpulan dari generate noise gaussian.

(4) Dengan menggunakan program pembangkitan noise salt & pepper di atas, cobalah prosentase noise 10%, 20%, 30% sampai dengan 90%, amati hasil citra bernoise, buatlah kesimpulan dari generate noise salt & pepper.

(5) Dengan menggunakan program pembangkitan noise speckle di atas, cobalah prosentase noise 10%, 20%, 30% sampai dengan 90%, amati hasil citra bernoise, buatlah kesimpulan dari generate noise speckle.

(6) Dengan program reduksi noise rata-rata menggunakan filter rata-rata 3x3, bila noise dibangkitkan dengan prosentase noise 5%, 10%, 15%, 20%, sampai dengan 50%. Bagaimana hasil citra yang telah direduksi noisenya ? berilah kesimpulan.

(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%. Bagaimana hasil citra yang telah direduksi noisenya ? berilah kesimpulan.