LAMPIRAN A

LISTING PROGRAM APLIKASI

A.1. Deklarasi Variabel-Variabel yang Dipakai dalam Program Aplikasi (modVar.bas)

```
Public bytByteArray() As Byte 'Byte array of the image file
Public lngSizeOfFile As Long 'Byte length of the image file
Public blnFileOpened As Boolean 'True = image file is opened
Public strFileName As String 'File name
Public lngWidth As Long 'Width of image
Public lngHeight As Long 'Height of image
Public lngTotalPixel As Long 'Width x Height of image (total sum of pixel in the image)
Public colR() As Double '1D original R color
Public colG() As Double '1D original G color
Public colB() As Double '1D original B color

Public lngPixel As Long
Public lngActualWidth As Long 'actual width of picture box
Public lngActualHeight As Long 'actual height of picture box
Public Y() As Double '1D Luminance
Public Cb() As Double '1D Chrominance Blue
Public Cr() As Double '1D Chrominance Red
Public Act1() As Double '1D processed R color/Luminance
Public Act2() As Double '1D processed G color/Chrominance Blue
Public Act3() As Double '1D processed B color/Chrominance Red
Public sngStart As Single 'Start of processed time needed
Public sngStop As Single 'Stop of processed time needed

Public blnResult As Boolean 'True = form result is opened
Public Const clngWidth As Long = 4000 'Picture box width if it's opened not in image's actual size
Public Const clngHeight As Long = 3000 'Picture box width if it's opened not in image's actual size

Public col2DR() As Double '2D original R color
Public col2DG() As Double '2D original R color
Public col2DB() As Double '2D original R color
Public Act2D1() As Double '2D processed R color/Luminance
Public Act2D2() As Double '2D processed G color/Chrominance Blue
Public Act2D3() As Double '2D processed B color/Chrominance Red

Public strDimension As String 'choose between 1D process or 2D process
Public blnNormal As Boolean 'choose between normalized transform/inverse process or non normalized
transform/inverse process

Public lngOriHistR() As Double 'histogram of original image's R color
Public lngOriHistG() As Double 'histogram of original image's G color
Public lngOriHistB() As Double 'histogram of original image's B color
Public lngOriMax As Double 'max number of histogram of original image's any color
Public lngOriMax2 As Double 'max number of histogram of original image's any color

Public lngRslHistR() As Double 'histogram of processed image's R color
Public lngRslHistG() As Double 'histogram of processed image's G color
Public lngRslHistB() As Double 'histogram of processed image's B color
Public lngRslMax As Double 'max number of histogram of processed image's any color
Public lngRslMax2 As Double 'max number of histogram of processed image's any color

Public blnOriHisto As Boolean 'True = form original histogram is opened
Public blnRslHisto As Boolean 'True = form result histogram is opened

Public strKomponen As String '"Komponen1"=hanya memproses komponen1,"All"=memproses ketiga
komponen

Public strProccess As String
'"YCbCr"=LuminanceCbCr,"RGB"=RedGreenBlue,"TW/IWT"=Inverse/Transformasi Wavelet
```

```
Public dblMinValue As Double 'Store min value for YCbCr/TW/IWT histogram
Public dblMaxValue As Double 'Store max value for YCbCr/TW/IWT histogram

Public dblBoundary As Double 'Store boundary value
Public dblN As Double '2^dblN = lngTotalPixel

Public lngWidthC As Long 'Width of comparison image
Public lngHeightC As Long 'Height of comparison image
Public lngTotalPixelC As Long 'Width x Height of comparison image (total sum of pixel in the image)
```

A.2. Prosedur untuk Membuka File Citra Digital (mdiMain.frm)

```
Private Sub OpenFile(FileName As String)
On Error GoTo errLoad
Dim Fno As Integer
   blnFileOpened = False
   Fno = FreeFile
   Open FileName For Binary As #Fno
      lngSizeOfFile = LOF(Fno)
      ReDim bytByteArray(1 To lngSizeOfFile) As Byte
      Get #Fno, , bytByteArray
   Close #Fno
   'cek validasi format file
   If ((bytByteArray(1))) & ((bytByteArray(2))) <> "6677" Then
      Unload frmOriImage
      pbarP.Visible = False
      MsgBox "Format citra tidak sesuai dengan format yang ditentukan yaitu 24-bit BMP", vbCritical, "Validasi
resolusi citra"
      Exit Sub
   End If
   'cek validasi lebar dan tinggi citra
   lngWidth = HexToDec(Format(Hex(bytByteArray(22)), "00") & Format(Hex(bytByteArray(21)), "00") &
Format(Hex(bytByteArray(20)), "00") & Format(Hex(bytByteArray(19)), "00"))
   lngHeight = HexToDec(Format(Hex(bytByteArray(26)), "00") & Format(Hex(bytByteArray(25)), "00") &
Format(Hex(bytByteArray(24)), "00") & Format(Hex(bytByteArray(23)), "00"))
   If lngWidth > 512 Then
      Unload frmOriImage
      pbarP.Visible = False
      MsgBox "Lebar resolusi dari citra melebihi batas yang ditentukan yaitu 512", vbCritical, "Validasi resolusi
citra"
      Exit Sub
   ElseIf InStr(1, Log(lngWidth) / Log(2), ",") <> 0 Or InStr(1, Log(lngWidth) / Log(2), ".") Then
      Unload frmOriImage
      pbarP.Visible = False
      MsgBox "Lebar resolusi dari citra tidak memenuhi syarat yaitu harus merupakan 2 pangkat x" _
         & vbNewLine & "dimana x harus merupakan bilangan bulat", vbCritical, "Validasi resolusi citra"
      Exit Sub
   ElseIf lngHeight > 512 Then
      Unload frmOriImage
      pbarP.Visible = False
      MsgBox "Tinggi resolusi dari citra melebihi batas yang ditentukan yaitu 512", vbCritical, "Validasi resolusi
citra"
      Exit Sub
   ElseIf InStr(1, Log(lngHeight) / Log(2), ",") <> 0 Or InStr(1, Log(lngHeight) / Log(2), ".") <> 0 Then
      Unload frmOriImage
      pbarP.Visible = False
      MsgBox "Tinggi resolusi dari citra tidak memenuhi syarat yaitu harus merupakan 2 pangkat x" _
         & vbNewLine & "dimana x harus merupakan bilangan bulat", vbCritical, "Validasi resolusi citra"
      Exit Sub
   End If
   'ambil rgb dari tiap pixel citra
   If funcGetRGB(strDimension) = False Then Exit Sub
   sngStop = Timer
   blnFileOpened = True
   subDisplayFileProp
   Exit Sub
errLoad:
   sbarStatus.Panels("Status").Text = "Error pada saat membuka file citra digital"
End Sub
```

A.3. Prosedur untuk Mendapatkan Array R, G, B dari Citra Digital (modProcFunc.bas)

```
Public Function funcGetRGB(Dimension As String) As Boolean
Dim j As Long
   On Error GoTo errLoad
   funcGetRGB = False
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   lngTotalPixel = lngWidth * lngHeight
   Select Case Dimension
   Case "1D"
      dblN = Log(lngTotalPixel) / Log(2)
      mdiMain.txtDepth.ToolTipText = "Min res = 1, Max res = " & dblN
   Case "2D"
      If lngWidth > lngHeight Then
         dblN = Log(lngHeight) / Log(2)
         mdiMain.txtDepth.ToolTipText = "Min res = 1, Max res = " & dblN
      ElseIf lngWidth < lngHeight Then
         dblN = Log(lngWidth) / Log(2)
         mdiMain.txtDepth.ToolTipText = "Min res = 1, Max res = " & dblN
      Else
         dblN = Log(lngWidth) / Log(2)
         mdiMain.txtDepth.ToolTipText = "Min res = 1, Max res = " & dblN
      End If
   End Select

Dim dblBTmp As Double
Dim dblGTmp As Double
Dim dblRTmp As Double

   lngPixel = 55
   Select Case Dimension
   Case "1D"
      ReDim colR(1 To lngTotalPixel)
      ReDim colG(1 To lngTotalPixel)
      ReDim colB(1 To lngTotalPixel)
      ReDim Act1(1 To lngTotalPixel)
      ReDim Act2(1 To lngTotalPixel)
      ReDim Act3(1 To lngTotalPixel)

      mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
      For j = 1 To lngTotalPixel
         If j Mod 10000 = 0 Then
            mdiMain.pbarP.Max = mdiMain.pbarP.Max + 1
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
         End If
         DoEvents

         dblBTmp = bytByteArray(lngPixel)
         colB(j) = dblBTmp
         Act3(j) = dblBTmp
         lngPixel = lngPixel + 1

         dblGTmp = bytByteArray(lngPixel)
         colG(j) = dblGTmp
         Act2(j) = dblGTmp
         lngPixel = lngPixel + 1

         dblRTmp = bytByteArray(lngPixel)
         colR(j) = dblRTmp
         Act1(j) = dblRTmp
         lngPixel = lngPixel + 1
      Next j
   Case "2D"
      ReDim col2DR(1 To lngWidth, 1 To lngHeight)
      ReDim col2DG(1 To lngWidth, 1 To lngHeight)
      ReDim col2DB(1 To lngWidth, 1 To lngHeight)
      ReDim Act2D1(1 To lngWidth, 1 To lngHeight)
      ReDim Act2D2(1 To lngWidth, 1 To lngHeight)
```

```
        ReDim Act2D3(1 To lngWidth, 1 To lngHeight)

        Dim lng2DWidth As Long
        Dim lng2DHeight As Long

        lng2DWidth = 1
        lng2DHeight = 1
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        For j = 1 To lngTotalPixel
           If j Mod 10000 = 0 Then
              mdiMain.pbarP.Max = mdiMain.pbarP.Max + 1
              mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
           End If
           DoEvents

           dblBTmp = bytByteArray(lngPixel)
           lngPixel = lngPixel + 1
           dblGTmp = bytByteArray(lngPixel)
           lngPixel = lngPixel + 1
           dblRTmp = bytByteArray(lngPixel)
           lngPixel = lngPixel + 1

           If (j Mod lngWidth) <> 0 Then
              lng2DWidth = (j Mod lngWidth)
              col2DB(lng2DWidth, lng2DHeight) = dblBTmp
              col2DG(lng2DWidth, lng2DHeight) = dblGTmp
              col2DR(lng2DWidth, lng2DHeight) = dblRTmp
              Act2D3(lng2DWidth, lng2DHeight) = dblBTmp
              Act2D2(lng2DWidth, lng2DHeight) = dblGTmp
              Act2D1(lng2DWidth, lng2DHeight) = dblRTmp
           Else
              lng2DWidth = lngWidth
              col2DB(lng2DWidth, lng2DHeight) = dblBTmp
              col2DG(lng2DWidth, lng2DHeight) = dblGTmp
              col2DR(lng2DWidth, lng2DHeight) = dblRTmp
              Act2D3(lng2DWidth, lng2DHeight) = dblBTmp
              Act2D2(lng2DWidth, lng2DHeight) = dblGTmp
              Act2D1(lng2DWidth, lng2DHeight) = dblRTmp
              lng2DHeight = lng2DHeight + 1
           End If
        Next j
     End Select
     funcGetRGB = True
     Exit Function
errLoad:
End Function
```

A.4. Prosedur untuk Mengkonversi Array RGB menjadi Array YCbCr (modProcFunc.bas)

```
Public Function funcRGBtoYCbCr(Dimension As String) As Boolean
     mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
     On Error GoTo errLoad
     mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
     funcRGBtoYCbCr = False
     mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
     If blnFileOpened = False Then Exit Function
     mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
Dim dblYTmp As Double
Dim dblCbTmp As Double
Dim dblCrTmp As Double
Dim dblAdd As Double
     dblAdd = 128
     If Dimension = "1D" Then
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        For j = 1 To lngTotalPixel
           DoEvents
```

```
            If j Mod 10000 = 0 Then
               mdiMain.pbarP.Max = mdiMain.pbarP.Max + 1
               mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
            End If
            dblYTmp = (0.299 * Act1(j)) + (0.587 * Act2(j)) + (0.114 * Act3(j))
            dblCbTmp = (-0.16874 * Act1(j)) + (-0.33126 * Act2(j)) + (0.5 * Act3(j)) + dblAdd
            dblCrTmp = (0.5 * Act1(j)) + (-0.41869 * Act2(j)) + (-0.08131 * Act3(j)) + dblAdd
            Act1(j) = dblYTmp
            Act2(j) = dblCbTmp
            Act3(j) = dblCrTmp
         Next j
      ElseIf Dimension = "2D" Then
         For i = 1 To lngHeight
            For j = 1 To lngWidth
               dblYTmp = (0.299 * Act2D1(j, i)) + (0.587 * Act2D2(j, i)) + (0.114 * Act2D3(j, i))
               dblCbTmp = (-0.16874 * Act2D1(j, i)) + (-0.33126 * Act2D2(j, i)) + (0.5 * Act2D3(j, i)) + dblAdd
               dblCrTmp = (0.5 * Act2D1(j, i)) + (-0.41869 * Act2D2(j, i)) + (-0.08131 * Act2D3(j, i)) + dblAdd
               Act2D1(j, i) = dblYTmp
               Act2D2(j, i) = dblCbTmp
               Act2D3(j, i) = dblCrTmp
            Next j
         Next i
      End If
      funcRGBtoYCbCr = True
errLoad:
End Function
```

A.5. Prosedur untuk Mengkonversi Array YCbCr menjadi Array RGB (modProcFunc.bas)

```
Public Function funcYCbCrToRGB(Dimension As String) As Boolean
   On Error GoTo errLoad
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   funcYCbCrToRGB = False
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   If blnFileOpened = False Then Exit Function
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
Dim dblAdd1 As Double
Dim dblAdd2 As Double
Dim dblAdd3 As Double
Dim dblR As Double
Dim dblG As Double
Dim dblB As Double
   dblAdd1 = -179.2
   dblAdd2 = 135.42
   dblAdd3 = -227.07
   If Dimension = "1D" Then
      For j = 1 To lngTotalPixel
         DoEvents
         If j Mod 10000 = 0 Then
            mdiMain.pbarP.Max = mdiMain.pbarP.Max + 1
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
         End If
```

```
            dblR = (1 * Act1(j)) + (1.402 * Act3(j)) + dblAdd1
            dblG = (1 * Act1(j)) + (-0.34414 * Act2(j)) + (-0.71414 * Act3(j)) + dblAdd2
            dblB = (1 * Act1(j)) + (1.772 * Act2(j)) + dblAdd3
            Act1(j) = dblR
            Act2(j) = dblG
            Act3(j) = dblB
        Next j
    ElseIf Dimension = "2D" Then
        For i = 1 To lngHeight
            For j = 1 To lngWidth
                dblR = (1 * Act2D1(j, i)) + (1.402 * Act2D3(j, i)) + dblAdd1
                dblG = (1 * Act2D1(j, i)) + (-0.34414 * Act2D2(j, i)) + (-0.71414 * Act2D3(j, i)) + dblAdd2
                dblB = (1 * Act2D1(j, i)) + (1.772 * Act2D2(j, i)) + dblAdd3
                Act2D1(j, i) = dblR
                Act2D2(j, i) = dblG
                Act2D3(j, i) = dblB
            Next j
        Next i
    End If
    funcYCbCrToRGB = True
errLoad:
End Function
```

A.6. Prosedur untuk Melakukan Transformasi Wavelet Terhadap Array Citra Digital Masukan (modProcFunc.bas)

```
Public Function funcWaveletTransform(Dimension As String, blnNormalized As Boolean, Choose As String)
As Boolean
    On Error GoTo errLoad
Dim ytmp() As Double

    mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
    funcWaveletTransform = False
    mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
    If blnFileOpened = False Then Exit Function
    mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
    Select Case Dimension
    Case "1D"
        ReDim ytmp(1 To lngTotalPixel)

        subDecomposition lngTotalPixel, Act1, ytmp, blnNormalized
        Act1 = ytmp
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        If Choose = "All" Then
            subDecomposition lngTotalPixel, Act2, ytmp, blnNormalized
            Act2 = ytmp
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1

            subDecomposition lngTotalPixel, Act3, ytmp, blnNormalized
            Act3 = ytmp
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        End If
    Case "2D"
        ReDim ytmp(1 To lngWidth, 1 To lngHeight)

        sub2DDecomposition Act2D1, ytmp, blnNormalized
        Act2D1 = ytmp
        mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
```

```
        If Choose = "All" Then
            sub2DDecomposition Act2D2, ytmp, blnNormalized
            Act2D2 = ytmp
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1

            sub2DDecomposition Act2D3, ytmp, blnNormalized
            Act2D3 = ytmp
            mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
        End If
    End Select
    funcWaveletTransform = True
errLoad:
End Function

(modWaveletTransform.bas)
Public Sub subDecomposition(N As Long, Xin() As Double, XOut() As Double, blnNormalized As Boolean)
Dim i As Long
Dim Xtmp1() As Double
ReDim Xtmp1(1 To N)
Dim Xtmp2() As Double
ReDim Xtmp2(1 To N)
Dim intN As Long
    If blnNormalized = True Then
        For i = 1 To N
            Xtmp1(i) = Xin(i) / Sqr(N) 'normalize input coefficient
        Next i
    Else
        Xtmp1 = Xin
    End If
    intN = N
    While intN >= 2
        subDecompositionStep intN, Xtmp1, Xtmp2, blnNormalized
        Xtmp1 = Xtmp2
        intN = intN / 2
    Wend
    XOut = Xtmp1
End Sub

(modWaveletTransform.bas)
Public Sub subDecompositionStep(N As Long, Xin() As Double, XOut() As Double, blnNormalized As
Boolean)
Dim i As Long
Dim intN As Long
    If blnNormalized = True Then
        For i = 1 To N / 2
            XOut(i) = (Xin((2 * i) - 1) + Xin(2 * i)) / Sqr(2)
            XOut((N / 2) + i) = (Xin((2 * i) - 1) - Xin(2 * i)) / Sqr(2)
        Next i
    Else
        For i = 1 To N / 2
            XOut(i) = (Xin((2 * i) - 1) + Xin(2 * i)) / 2
            XOut((N / 2) + i) = (Xin((2 * i) - 1) - Xin(2 * i)) / 2
        Next i
    End If
End Sub

(modWaveletTransform.bas)
Public Sub sub2DDecomposition(Xin() As Double, XOut() As Double, blnNormalized As Boolean)
Dim i As Long
Dim j As Long
Dim Xtmp1() As Double
Dim Xtmp2() As Double
Dim Xtmp2D1() As Double
ReDim Xtmp1(1 To lngWidth)
ReDim Xtmp2(1 To lngWidth)
ReDim Xtmp2D1(1 To lngWidth, 1 To lngHeight)
    Xtmp2D1 = Xin
    For i = lngHeight To 1 Step -1
        For j = 1 To lngWidth
            Xtmp1(j) = Xtmp2D1(j, i)
```

```
      Next j
      subDecomposition lngWidth, Xtmp1, Xtmp2, blnNormalized
      For j = 1 To lngWidth
         Xtmp2D1(j, i) = Xtmp2(j)
      Next j
   Next i
ReDim Xtmp1(1 To lngHeight)
ReDim Xtmp2(1 To lngHeight)
   For j = 1 To lngWidth
      For i = lngHeight To 1 Step -1
         Xtmp1(i) = Xtmp2D1(j, i)
      Next i
      subDecomposition lngHeight, Xtmp1, Xtmp2, blnNormalized
      For i = 1 To lngHeight
         Xtmp2D1(j, i) = Xtmp2(i)
      Next i
   Next j
   XOut = Xtmp2D1
End Sub
```

A.7. Prosedur untuk Melakukan Invers Transformasi Wavelet Terhadap Array Koefisien-Koefisien Hasil Transformasi (modProcFunc.bas)

```
Public Function funcInverseWaveletTransform(Dimension As String, blnNormalized As Boolean, Choose As
String) As Boolean
   On Error GoTo errLoad
Dim ytmp() As Double

   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   funcInverseWaveletTransform = False
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   If blnFileOpened = False Then Exit Function
   mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
   Select Case Dimension
   Case "1D"
      ReDim ytmp(1 To lngTotalPixel)

      subReconstruction lngTotalPixel, Act1, ytmp, blnNormalized
      Act1 = ytmp
      mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
      If Choose = "All" Then
         subReconstruction lngTotalPixel, Act2, ytmp, blnNormalized
         Act2 = ytmp
         mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1

         subReconstruction lngTotalPixel, Act3, ytmp, blnNormalized
         Act3 = ytmp
         mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
      End If
   Case "2D"
      ReDim ytmp(1 To lngWidth, 1 To lngHeight)

      sub2DReconstruction Act2D1, ytmp, blnNormalized
      Act2D1 = ytmp
      mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
      If Choose = "All" Then
         sub2DReconstruction Act2D2, ytmp, blnNormalized
         Act2D2 = ytmp
         mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1

         sub2DReconstruction Act2D3, ytmp, blnNormalized
         Act2D3 = ytmp
         mdiMain.pbarP.Value = mdiMain.pbarP.Value + 1
      End If
   End Select
   funcInverseWaveletTransform = True
errLoad:
End Function
```

```
(modWaveletTransform.bas)
Public Sub subReconstruction(N As Long, Xin() As Double, XOut() As Double, blnNormalized As Boolean)
Dim i As Long
Dim Xtmp1() As Double
ReDim Xtmp1(1 To N)
Dim Xtmp2() As Double
ReDim Xtmp2(1 To N)
Dim intN As Long
   intN = 2
   Xtmp1 = Xin
   While intN <= N
      subReconstructionStep intN, Xtmp1, Xtmp2, blnNormalized
      For i = 1 To intN
         Xtmp1(i) = Xtmp2(i)
      Next i
      intN = intN * 2
   Wend
   If blnNormalized = True Then
      For i = 1 To N
         XOut(i) = Xtmp1(i) * Sqr(N) 'undo normalization
      Next i
   Else
      XOut = Xtmp1
   End If
End Sub

(modWaveletTransform.bas)
Public Sub subReconstructionStep(N As Long, Xin() As Double, XOut() As Double, blnNormalized As
Boolean)
Dim i As Long
Dim intN As Long
   If blnNormalized = True Then
      For i = 1 To N / 2
         XOut((2 * i) - 1) = (Xin(i) + Xin((N / 2) + i)) / Sqr(2)
         XOut(2 * i) = (Xin(i) - Xin((N / 2) + i)) / Sqr(2)
      Next i
   Else
      For i = 1 To N / 2
         XOut((2 * i) - 1) = (Xin(i) + Xin((N / 2) + i))
         XOut(2 * i) = (Xin(i) - Xin((N / 2) + i))
      Next i
   End If
End Sub

(modWaveletTransform.bas)
Public Sub sub2DReconstruction(Xin() As Double, XOut() As Double, blnNormalized As Boolean)
Dim i As Long
Dim j As Long
Dim Xtmp1() As Double
Dim Xtmp2() As Double
Dim Xtmp2D1() As Double
ReDim Xtmp1(1 To lngHeight)
ReDim Xtmp2(1 To lngHeight)
ReDim Xtmp2D1(1 To lngWidth, 1 To lngHeight)
   Xtmp2D1 = Xin
   For j = 1 To lngWidth
      For i = lngHeight To 1 Step -1
         Xtmp1(i) = Xtmp2D1(j, i)
      Next i
      subReconstruction lngHeight, Xtmp1, Xtmp2, blnNormalized
      For i = 1 To lngHeight
         Xtmp2D1(j, i) = Xtmp2(i)
      Next i
   Next j
ReDim Xtmp1(1 To lngWidth)
ReDim Xtmp2(1 To lngWidth)
   For i = lngHeight To 1 Step -1
      For j = 1 To lngWidth
         Xtmp1(j) = Xtmp2D1(j, i)
      Next j
```

```
      subReconstruction lngWidth, Xtmp1, Xtmp2, blnNormalized
      For j = 1 To lngWidth
         Xtmp2D1(j, i) = Xtmp2(j)
      Next j
   Next i
   XOut = Xtmp2D1
End Sub
```

A.8. Prosedur/Fungsi Matematis yang Diperlukan untuk Perhitungan dalam Program Aplikasi (modMath.bas & modSort.bas)

```
(modMath.bas)
Public Function ln(X As Double) As Double
   ln = Log(X) / Log(Exp(1))
End Function

(modSort.bas)
Public Sub subAbsValue(ArrayInput() As Double)
Dim i As Long
   For i = LBound(ArrayInput) To UBound(ArrayInput)
      ArrayInput(i) = Abs(ArrayInput(i))
   Next i
End Sub

(modSort.bas)
Public Sub subBubbleSort(ArrayInput() As Double)
Dim i As Long
Dim j As Long
Dim dblTmp As Double
   For i = 2 To UBound(ArrayInput)
      For j = UBound(ArrayInput) To i Step -1
         DoEvents
         If ArrayInput(j) < ArrayInput(j - 1) Then
            dblTmp = ArrayInput(j)
            ArrayInput(j) = ArrayInput(j - 1)
            ArrayInput(j - 1) = dblTmp
         End If
      Next j
   Next i
End Sub

(modSort.bas)
Public Sub subQuickSort(ArrayInput() As Double)
Dim L As Long
Dim R As Long
Dim P() As Long
Dim minStr As Double
Dim maxStr As Double
Dim i As Long
Dim ArrayTmp() As Double
   L = LBound(ArrayInput)
   R = UBound(ArrayInput)
   ReDim ArrayTmp(L - 1 To R + 1)
   ReDim P(L To R)
   minStr = 0
   maxStr = 0
   For i = L To R
      If maxStr < ArrayInput(i) Then maxStr = ArrayInput(i)
      P(i) = i
   Next i
   minStr = maxStr
   For i = L To R
      If minStr > ArrayInput(i) Then minStr = ArrayInput(i)
   Next i
   For i = L To R
      ArrayTmp(i) = ArrayInput(i)
   Next i
   'We put "sentinel" values flanking the real keys to avoid an extra test in
   ' the inner loop.
```

```
    ArrayTmp(L - 1) = minStr
    ArrayTmp(R + 1) = maxStr
    'We mostly sort the list with QuickSort.
    subQuick L, R, ArrayTmp, P
    'Then we finish up with low overhead InsertionSort
    subInsert L, R, ArrayTmp, P
    For i = L To R
       ArrayTmp(i) = ArrayInput(P(i))
    Next i
    For i = L To R
       ArrayInput(i) = ArrayTmp(i)
    Next i
End Sub

(modSort.bas)
Private Sub subQuick(L As Long, R As Long, A() As Double, P() As Long)
    Dim MED As Long
    Dim LP As Long
    Dim RP As Long
    Dim Pivot As String
    Dim TMP As Long

    'Sublists <= 12 keys will be finished by running the whole list once thru
    ' InsertionSort.
    If R - L > 12 Then
    'Get the median pointer...
       MED = (L + R) \ 2
    'and swap it to the leftmost position.
       TMP = P(MED)
       P(MED) = P(L)
       P(L) = TMP
    'Now compare the leftmost, next leftmost & rightmost to choose a median of
    ' 3...
       If A(P(L + 1)) > A(P(R)) Then
          TMP = P(L + 1)
          P(L + 1) = P(R)
          P(R) = TMP
       End If
       If A(P(L)) > A(P(R)) Then
          TMP = P(L)
          P(L) = P(R)
          P(R) = TMP
       End If
       If A(P(L + 1)) > A(P(L)) Then
          TMP = P(L + 1)
          P(L + 1) = P(L)
          P(L) = TMP
       End If
    'and use its key as our pivot.
       Pivot = A(P(L))
    'Now work inward from each end.
       LP = L
       RP = R + 1
       Do
       'Scan right for a pointer whose key >= Pivot.  In case Pivot is the
       ' largest key, we have
       'a sentinel value of MaxStr in A(R + 1) that will end a runaway loop.
       ' Using the sentinel
       'avoids having a second test in the inner loop,
       '  so it can be as fast as possible.
          DoEvents
          Do
             LP = LP + 1
          Loop While A(P(LP)) < Pivot
       'Scan left for a pointer whose key <= Pivot.  Again,
       '  we have a sentinel value of MinStr
       'in A(L - 1) to stop the loop if Pivot is the smallest value in the
       ' list.
          Do
             RP = RP - 1
```

```
        Loop While A(P(RP)) > Pivot
      'If the pointers have crossed we're done.
        If RP <= LP Then Exit Do
      'Otherwise, swap the pair we've identified.
        TMP = P(LP)
        P(LP) = P(RP)
        P(RP) = TMP
      Loop
   'Swap the pointer of the Pivot value back into place.
      TMP = P(L)
      P(L) = P(RP)
      P(RP) = TMP
   'Sort the shorter sublist first so the recursion stack is limited to
   ' logarithmic depth.
      If (RP - 1) - L <= R - LP Then
        subQuick L, RP - 1, A, P
        subQuick LP, R, A, P
      Else
        subQuick LP, R, A, P
        subQuick L, RP - 1, A, P
      End If
   End If
End Sub

(modSort.bas)
Private Sub subInsert(L As Long, R As Long, A() As Double, P() As Long)
   Dim LP As Long
   Dim RP As Long
   Dim TMP As Long
   Dim T As String

   For RP = L + 1 To R
      TMP = P(RP)
      T = A(TMP)
      For LP = RP To L + 1 Step -1
        If T < A(P(LP - 1)) Then P(LP) = P(LP - 1) Else Exit For
      Next LP
      P(LP) = TMP
   Next RP
End Sub
```

A.9. Prosedur-Prosedur Penghalusan (modIE.bas)

```
Option Explicit
Dim i As Long
Dim j As Long
Dim k As Long

Public dblMask() As Double
Public strJumlahTitik As String

'Uniform smoothing 'Choose="Komponen1"/"All"
Public Sub subIESmooth(Resolution As Integer, ChooseKomponen As String, ChooseIEType As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim dblTmp As Double
Dim lngCount As Long
Dim dblBoundary As Double
Dim u As Integer
Dim v As Integer
Dim uMin As Integer
Dim uMax As Integer
   Select Case ChooseIEType
   Case "5titik"
      dblTmp = 1 / 5
      uMin = -1
      uMax = 1
      ReDim dblMask(uMin To uMax, uMin To uMax)
      dblMask(-1, -1) = 0
```

```
    dblMask(0, -1) = dblTmp
    dblMask(1, -1) = 0
    dblMask(-1, 0) = dblTmp
    dblMask(0, 0) = dblTmp
    dblMask(1, 0) = dblTmp
    dblMask(-1, 1) = 0
    dblMask(0, 1) = dblTmp
    dblMask(1, 1) = 0
Case "9titik"
    dblTmp = 1 / 9
    uMin = -1
    uMax = 1
    ReDim dblMask(uMin To uMax, uMin To uMax)
    For u = uMin To uMax
        For v = uMin To uMax
            dblMask(v, u) = dblTmp
        Next v
    Next u
Case "25titik"
    dblTmp = 1 / 25
    uMin = -2
    uMax = 2
    ReDim dblMask(uMin To uMax, uMin To uMax)
    For u = uMin To uMax
        For v = uMin To uMax
            dblMask(v, u) = dblTmp
        Next v
    Next u
Case "49titik"
    dblTmp = 1 / 49
    uMin = -3
    uMax = 3
    ReDim dblMask(uMin To uMax, uMin To uMax)
    For u = uMin To uMax
        For v = uMin To uMax
            dblMask(v, u) = dblTmp
        Next v
    Next u
End Select
lngCount = 0
dblTmp = 0
Select Case strDimension
Case "1D"
    ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
    Dim lng2DWidth As Long
    Dim lng2DHeight As Long

    lng2DWidth = 0
    lng2DHeight = 1
    dblBoundary = lngTotalPixel / (2 ^ Resolution)
    For i = 1 To lngTotalPixel
        If i <= dblBoundary Then
            lng2DWidth = lng2DWidth + 1
            tmpRo(lng2DWidth, lng2DHeight) = Act1(i)
            tmpGo(lng2DWidth, lng2DHeight) = Act2(i)
            tmpBo(lng2DWidth, lng2DHeight) = Act3(i)
            If (i Mod lngWidth) = 0 Then
                lng2DHeight = lng2DHeight + 1
                lng2DWidth = 0
            End If
        Else
            lng2DWidth = lngWidth
            lng2DHeight = lng2DHeight - 1
            Exit For
        End If
    Next i
    ReDim Preserve tmpRo(1 To lng2DWidth, 1 To lng2DHeight)
    ReDim Preserve tmpGo(1 To lng2DWidth, 1 To lng2DHeight)
```

```
ReDim Preserve tmpBo(1 To lng2DWidth, 1 To lng2DHeight)
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
For i = 1 To lng2DHeight
   For j = 1 To lng2DWidth
      dblTmpR = 0
      dblTmpG = 0
      dblTmpB = 0
      For u = uMin To uMax
         For v = uMin To uMax
            If j - v < 1 Then
               dblTmpR = dblTmpR + 0
            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
               dblTmpR = dblTmpR + 0
            ElseIf i - u < 1 Then
               dblTmpR = dblTmpR + 0
            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
               dblTmpR = dblTmpR + 0
            Else
               dblTmpR = dblTmpR + tmpRo(j - v, i - u) * dblMask(v, u)
            End If
            If ChooseKomponen = "All" Then
               If j - v < 1 Then
                  dblTmpG = dblTmpG + 0
               ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                  dblTmpG = dblTmpG + 0
               ElseIf i - u < 1 Then
                  dblTmpG = dblTmpG + 0
               ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                  dblTmpG = dblTmpG + 0
               Else
                  dblTmpG = dblTmpG + tmpGo(j - v, i - u) * dblMask(v, u)
               End If

               If j - v < 1 Then
                  dblTmpB = dblTmpB + 0
               ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                  dblTmpB = dblTmpB + 0
               ElseIf i - u < 1 Then
                  dblTmpB = dblTmpB + 0
               ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                  dblTmpB = dblTmpB + 0
               Else
                  dblTmpB = dblTmpB + tmpBo(j - v, i - u) * dblMask(v, u)
               End If
            End If
         Next v
      Next u
      tmpRo(j, i) = dblTmpR
      tmpGo(j, i) = dblTmpG
      tmpBo(j, i) = dblTmpB
   Next j
Next i
Dim tmpRo1() As Double
Dim tmpGo1() As Double
Dim tmpBo1() As Double
ReDim tmpRo1(1 To lngTotalPixel)
ReDim tmpGo1(1 To lngTotalPixel)
ReDim tmpBo1(1 To lngTotalPixel)

lngCount = 0
For i = 1 To lng2DHeight
   For j = 1 To lng2DWidth
      lngCount = lngCount + 1
      tmpRo1(lngCount) = tmpRo(j, i)
      tmpGo1(lngCount) = tmpGo(j, i)
      tmpBo1(lngCount) = tmpBo(j, i)
   Next j
Next i
```

```
    For i = dblBoundary + 1 To lngTotalPixel
        tmpRo1(i) = Act1(i)
        tmpGo1(i) = Act2(i)
        tmpBo1(i) = Act3(i)
    Next i
    Act1 = tmpRo1
    If ChooseKomponen = "All" Then
        Act2 = tmpGo1
        Act3 = tmpBo1
    End If
Case "2D"
    ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
    dblBoundary = lngTotalPixel / (2 ^ Resolution)
    For i = 1 To lngHeight
        For j = 1 To lngWidth
            If j <= lngWidth / (2 ^ Resolution) And i <= lngHeight / (2 ^ Resolution) Then
                dblTmp = 0
                For u = uMin To uMax
                    For v = uMin To uMax
                        If j - v < 1 Then
                            dblTmp = dblTmp + 0
                        ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                            dblTmp = dblTmp + 0
                        ElseIf i - u < 1 Then
                            dblTmp = dblTmp + 0
                        ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                            dblTmp = dblTmp + 0
                        Else
                            dblTmp = dblTmp + Act2D1(j - v, i - u) * dblMask(u, v)
                        End If
                    Next v
                Next u
                tmpRo(j, i) = dblTmp
            Else
                tmpRo(j, i) = Act2D1(j, i)
            End If
            If ChooseKomponen = "All" Then
                If j <= dblBoundary And i <= dblBoundary Then
                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            Else
                                dblTmp = dblTmp + Act2D2(j - v, i - u) * dblMask(u, v)
                            End If
                        Next v
                    Next u
                    tmpGo(j, i) = dblTmp

                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            Else
```

```
                        dblTmp = dblTmp + Act2D3(j - v, i - u) * dblMask(u, v)
                    End If
                Next v
            Next u
            tmpBo(j, i) = dblTmp
        Else
            tmpGo(j, i) = Act2D2(j, i)
            tmpBo(j, i) = Act2D3(j, i)
        End If
    End If
  Next j
 Next i
 Act2D1 = tmpRo
 If ChooseKomponen = "All" Then
  Act2D2 = tmpGo
  Act2D3 = tmpBo
 End If
 End Select
End Sub

'Gaussian smoothing 'Choose="Komponen1"/"All"
Public Sub subIEGSmooth(Resolution As Integer, ChooseKomponen As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim dblTmp As Double
Dim lngCount As Long
Dim dblBoundary As Double
Dim u As Integer
Dim v As Integer
Dim uMin As Integer
Dim uMax As Integer
 uMin = -4
 uMax = 4
 ReDim dblMask(uMin To uMax, uMin To uMax)
 dblMask(-4, 4) = 0#
 dblMask(-3, 4) = 0.000001
 dblMask(-2, 4) = 0.000007
 dblMask(-1, 4) = 0.000032
 dblMask(0, 4) = 0.000053
 dblMask(1, 4) = 0.000032
 dblMask(2, 4) = 0.000007
 dblMask(3, 4) = 0.000001
 dblMask(4, 4) = 0#
 dblMask(-4, 3) = 0.000001
 dblMask(-3, 3) = 0.00002
 dblMask(-2, 3) = 0.000239
 dblMask(-1, 3) = 0.001072
 dblMask(0, 3) = 0.001768
 dblMask(1, 3) = 0.001072
 dblMask(2, 3) = 0.000239
 dblMask(3, 3) = 0.00002
 dblMask(4, 3) = 0.000001
 dblMask(-4, 2) = 0.000007
 dblMask(-3, 2) = 0.000239
 dblMask(-2, 2) = 0.002915
 dblMask(-1, 2) = 0.013064
 dblMask(0, 2) = 0.021539
 dblMask(1, 2) = 0.013064
 dblMask(2, 2) = 0.002915
 dblMask(3, 2) = 0.000239
 dblMask(4, 2) = 0.000007
 dblMask(-4, 1) = 0.000032
 dblMask(-3, 1) = 0.001072
 dblMask(-2, 1) = 0.013064
 dblMask(-1, 1) = 0.05855
 dblMask(0, 1) = 0.096532
 dblMask(1, 1) = 0.05855
 dblMask(2, 1) = 0.013064
 dblMask(3, 1) = 0.001072
```

```
dblMask(4, 1) = 0.000032
dblMask(-4, 0) = 0.000053
dblMask(-3, 0) = 0.001768
dblMask(-2, 0) = 0.021539
dblMask(-1, 0) = 0.096532
dblMask(0, 0) = 0.159155
dblMask(1, 0) = 0.096532
dblMask(2, 0) = 0.021539
dblMask(3, 0) = 0.001768
dblMask(4, 0) = 0.000053
dblMask(-4, -1) = 0.000032
dblMask(-3, -1) = 0.001072
dblMask(-2, -1) = 0.013064
dblMask(-1, -1) = 0.05855
dblMask(0, -1) = 0.096532
dblMask(1, -1) = 0.05855
dblMask(2, -1) = 0.013064
dblMask(3, -1) = 0.001072
dblMask(4, -1) = 0.000032
dblMask(-4, -2) = 0.000007
dblMask(-3, -2) = 0.000239
dblMask(-2, -2) = 0.002915
dblMask(-1, -2) = 0.013064
dblMask(0, -2) = 0.021539
dblMask(1, -2) = 0.013064
dblMask(2, -2) = 0.002915
dblMask(3, -2) = 0.000239
dblMask(4, -2) = 0.000007
dblMask(-4, -3) = 0.000001
dblMask(-3, -3) = 0.00002
dblMask(-2, -3) = 0.000239
dblMask(-1, -3) = 0.001072
dblMask(0, -3) = 0.001768
dblMask(1, -3) = 0.001072
dblMask(2, -3) = 0.000239
dblMask(3, -3) = 0.00002
dblMask(4, -3) = 0.000001
dblMask(-4, -4) = 0#
dblMask(-3, -4) = 0.000001
dblMask(-2, -4) = 0.000007
dblMask(-1, -4) = 0.000032
dblMask(0, -4) = 0.000053
dblMask(1, -4) = 0.000032
dblMask(2, -4) = 0.000007
dblMask(3, -4) = 0.000001
dblMask(4, -4) = 0#

lngCount = 0
dblTmp = 0
Select Case strDimension
Case "1D"
   ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
   ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
   ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
   Dim lng2DWidth As Long
   Dim lng2DHeight As Long

   lng2DWidth = 0
   lng2DHeight = 1
   dblBoundary = lngTotalPixel / (2 ^ Resolution)
   For i = 1 To lngTotalPixel
      If i <= dblBoundary Then
         lng2DWidth = lng2DWidth + 1
         tmpRo(lng2DWidth, lng2DHeight) = Act1(i)
         tmpGo(lng2DWidth, lng2DHeight) = Act2(i)
         tmpBo(lng2DWidth, lng2DHeight) = Act3(i)
         If (i Mod lngWidth) = 0 Then
            lng2DHeight = lng2DHeight + 1
            lng2DWidth = 0
         End If
```

```
    Else
      lng2DWidth = lngWidth
      lng2DHeight = lng2DHeight - 1
      Exit For
    End If
Next i
ReDim Preserve tmpRo(1 To lng2DWidth, 1 To lng2DHeight)
ReDim Preserve tmpGo(1 To lng2DWidth, 1 To lng2DHeight)
ReDim Preserve tmpBo(1 To lng2DWidth, 1 To lng2DHeight)
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
For i = 1 To lng2DHeight
   For j = 1 To lng2DWidth
      dblTmpR = 0
      dblTmpG = 0
      dblTmpB = 0
      For u = uMin To uMax
         For v = uMin To uMax
            If j - v < 1 Then
               dblTmpR = dblTmpR + 0
            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
               dblTmpR = dblTmpR + 0
            ElseIf i - u < 1 Then
               dblTmpR = dblTmpR + 0
            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
               dblTmpR = dblTmpR + 0
            Else
               dblTmpR = dblTmpR + tmpRo(j - v, i - u) * dblMask(v, u)
            End If
            If ChooseKomponen = "All" Then
               If j - v < 1 Then
                  dblTmpG = dblTmpG + 0
               ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                  dblTmpG = dblTmpG + 0
               ElseIf i - u < 1 Then
                  dblTmpG = dblTmpG + 0
               ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                  dblTmpG = dblTmpG + 0
               Else
                  dblTmpG = dblTmpG + tmpGo(j - v, i - u) * dblMask(v, u)
               End If

               If j - v < 1 Then
                  dblTmpB = dblTmpB + 0
               ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                  dblTmpB = dblTmpB + 0
               ElseIf i - u < 1 Then
                  dblTmpB = dblTmpB + 0
               ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                  dblTmpB = dblTmpB + 0
               Else
                  dblTmpB = dblTmpB + tmpBo(j - v, i - u) * dblMask(v, u)
               End If
            End If
         Next v
      Next u
      tmpRo(j, i) = dblTmpR
      tmpGo(j, i) = dblTmpG
      tmpBo(j, i) = dblTmpB
   Next j
Next i
Dim tmpRo1() As Double
Dim tmpGo1() As Double
Dim tmpBo1() As Double
ReDim tmpRo1(1 To lngTotalPixel)
ReDim tmpGo1(1 To lngTotalPixel)
ReDim tmpBo1(1 To lngTotalPixel)

lngCount = 0
```

```
    For i = 1 To lng2DHeight
        For j = 1 To lng2DWidth
            lngCount = lngCount + 1
            tmpRo1(lngCount) = tmpRo(j, i)
            tmpGo1(lngCount) = tmpGo(j, i)
            tmpBo1(lngCount) = tmpBo(j, i)
        Next j
    Next i
    For i = dblBoundary + 1 To lngTotalPixel
        tmpRo1(i) = Act1(i)
        tmpGo1(i) = Act2(i)
        tmpBo1(i) = Act3(i)
    Next i
    Act1 = tmpRo1
    If ChooseKomponen = "All" Then
        Act2 = tmpGo1
        Act3 = tmpBo1
    End If
Case "2D"
    ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
    dblBoundary = lngTotalPixel / (2 ^ Resolution)
    For i = 1 To lngHeight
        For j = 1 To lngWidth
            If j <= lngWidth / (2 ^ Resolution) And i <= lngHeight / (2 ^ Resolution) Then
                dblTmp = 0
                For u = uMin To uMax
                    For v = uMin To uMax
                        If j - v < 1 Then
                            dblTmp = dblTmp + 0
                        ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                            dblTmp = dblTmp + 0
                        ElseIf i - u < 1 Then
                            dblTmp = dblTmp + 0
                        ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                            dblTmp = dblTmp + 0
                        Else
                            dblTmp = dblTmp + Act2D1(j - v, i - u) * dblMask(u, v)
                        End If
                    Next v
                Next u
                tmpRo(j, i) = dblTmp
            Else
                tmpRo(j, i) = Act2D1(j, i)
            End If
            If ChooseKomponen = "All" Then
                If j <= dblBoundary And i <= dblBoundary Then
                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            Else
                                dblTmp = dblTmp + Act2D2(j - v, i - u) * dblMask(u, v)
                            End If
                        Next v
                    Next u
                    tmpGo(j, i) = dblTmp

                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
```

```
                        dblTmp = dblTmp + 0
                    ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                        dblTmp = dblTmp + 0
                    ElseIf i - u < 1 Then
                        dblTmp = dblTmp + 0
                    ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                        dblTmp = dblTmp + 0
                    Else
                        dblTmp = dblTmp + Act2D3(j - v, i - u) * dblMask(u, v)
                    End If
                Next v
            Next u
            tmpBo(j, i) = dblTmp
        Else
            tmpGo(j, i) = Act2D2(j, i)
            tmpBo(j, i) = Act2D3(j, i)
        End If
    End If
  Next j
 Next i
 Act2D1 = tmpRo
 If ChooseKomponen = "All" Then
  Act2D2 = tmpGo
  Act2D3 = tmpBo
 End If
End Select
End Sub

'Uniform smoothing dengan nilai ambang 'Choose="Komponen1"/"All"
Public Sub subIESmoothAmbang(Resolution As Integer, Value As Integer, ChooseKomponen As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim dblTmp As Double
Dim lngCount As Long
Dim dblBoundary As Double
Dim u As Integer
Dim v As Integer
Dim uMin As Integer
Dim uMax As Integer
 dblTmp = 1 / 25
 uMin = -2
 uMax = 2
 ReDim dblMask(uMin To uMax, uMin To uMax)
 For u = uMin To uMax
  For v = uMin To uMax
   dblMask(v, u) = dblTmp
  Next v
 Next u

 lngCount = 0
 dblTmp = 0
 Select Case strDimension
 Case "1D"
  ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
  ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
  ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
  Dim lng2DWidth As Long
  Dim lng2DHeight As Long

  lng2DWidth = 0
  lng2DHeight = 1
  dblBoundary = lngTotalPixel / (2 ^ Resolution)
  For i = 1 To lngTotalPixel
   If i <= dblBoundary Then
    lng2DWidth = lng2DWidth + 1
    tmpRo(lng2DWidth, lng2DHeight) = Act1(i)
    tmpGo(lng2DWidth, lng2DHeight) = Act2(i)
    tmpBo(lng2DWidth, lng2DHeight) = Act3(i)
    If (i Mod lngWidth) = 0 Then
```

```
            lng2DHeight = lng2DHeight + 1
            lng2DWidth = 0
        End If
    Else
        lng2DWidth = lngWidth
        lng2DHeight = lng2DHeight - 1
        Exit For
    End If
Next i
ReDim Preserve tmpRo(1 To lng2DWidth, 1 To lng2DHeight)
ReDim Preserve tmpGo(1 To lng2DWidth, 1 To lng2DHeight)
ReDim Preserve tmpBo(1 To lng2DWidth, 1 To lng2DHeight)
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
For i = 1 To lng2DHeight
    For j = 1 To lng2DWidth
        dblTmpR = 0
        dblTmpG = 0
        dblTmpB = 0
        For u = uMin To uMax
            For v = uMin To uMax
                If j - v < 1 Then
                    dblTmpR = dblTmpR + 0
                ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                    dblTmpR = dblTmpR + 0
                ElseIf i - u < 1 Then
                    dblTmpR = dblTmpR + 0
                ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                    dblTmpR = dblTmpR + 0
                Else
                    dblTmpR = dblTmpR + tmpRo(j - v, i - u) * dblMask(v, u)
                End If

                If ChooseKomponen = "All" Then
                    If j - v < 1 Then
                        dblTmpG = dblTmpG + 0
                    ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                        dblTmpG = dblTmpG + 0
                    ElseIf i - u < 1 Then
                        dblTmpG = dblTmpG + 0
                    ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                        dblTmpG = dblTmpG + 0
                    Else
                        dblTmpG = dblTmpG + tmpGo(j - v, i - u) * dblMask(v, u)
                    End If

                    If j - v < 1 Then
                        dblTmpB = dblTmpB + 0
                    ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                        dblTmpB = dblTmpB + 0
                    ElseIf i - u < 1 Then
                        dblTmpB = dblTmpB + 0
                    ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                        dblTmpB = dblTmpB + 0
                    Else
                        dblTmpB = dblTmpB + tmpBo(j - v, i - u) * dblMask(v, u)
                    End If
                End If
            Next v
        Next u
        If Abs(dblTmpR - tmpRo(j, i)) < Value Then
            tmpRo(j, i) = tmpRo(j, i)
        Else
            tmpRo(j, i) = Round(dblTmpR)
        End If

        If Abs(dblTmpG - tmpGo(j, i)) < Value Then
            tmpGo(j, i) = tmpGo(j, i)
        Else
```

```
            tmpGo(j, i) = Round(dblTmpG)
        End If

        If Abs(dblTmpB - tmpBo(j, i)) < Value Then
            tmpBo(j, i) = tmpBo(j, i)
        Else
            tmpBo(j, i) = Round(dblTmpB)
        End If
    Next j
Next i
Dim tmpRo1() As Double
Dim tmpGo1() As Double
Dim tmpBo1() As Double
ReDim tmpRo1(1 To lngTotalPixel)
ReDim tmpGo1(1 To lngTotalPixel)
ReDim tmpBo1(1 To lngTotalPixel)

lngCount = 0
For i = 1 To lng2DHeight
    For j = 1 To lng2DWidth
        lngCount = lngCount + 1
        tmpRo1(lngCount) = tmpRo(j, i)
        tmpGo1(lngCount) = tmpGo(j, i)
        tmpBo1(lngCount) = tmpBo(j, i)
    Next j
Next i
For i = dblBoundary + 1 To lngTotalPixel
    tmpRo1(i) = Act1(i)
    tmpGo1(i) = Act2(i)
    tmpBo1(i) = Act3(i)
Next i
Act1 = tmpRo1
If ChooseKomponen = "All" Then
    Act2 = tmpGo1
    Act3 = tmpBo1
End If
Case "2D"
ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
dblBoundary = lngTotalPixel / (2 ^ Resolution)
For i = 1 To lngHeight
    For j = 1 To lngWidth
        If j <= lngWidth / (2 ^ Resolution) And i <= lngHeight / (2 ^ Resolution) Then
            dblTmp = 0
            For u = uMin To uMax
                For v = uMin To uMax
                    If j - v < 1 Then
                        dblTmp = dblTmp + 0
                    ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                        dblTmp = dblTmp + 0
                    ElseIf i - u < 1 Then
                        dblTmp = dblTmp + 0
                    ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                        dblTmp = dblTmp + 0
                    Else
                        dblTmp = dblTmp + Act2D1(j - v, i - u) * dblMask(u, v)
                    End If
                Next v
            Next u

            If Abs(dblTmp - Act2D1(j, i)) < Value Then
                tmpRo(j, i) = Act2D1(j, i)
            Else
                tmpRo(j, i) = Round(dblTmp)
            End If
        Else
            tmpRo(j, i) = Act2D1(j, i)
        End If
        If ChooseKomponen = "All" Then
```

```
                If j <= dblBoundary And i <= dblBoundary Then
                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            Else
                                dblTmp = dblTmp + Act2D2(j - v, i - u) * dblMask(u, v)
                            End If
                        Next v
                    Next u

                    If Abs(dblTmp - Act2D2(j, i)) < Value Then
                        tmpGo(j, i) = Act2D2(j, i)
                    Else
                        tmpGo(j, i) = Round(dblTmp)
                    End If

                    dblTmp = 0
                    For u = uMin To uMax
                        For v = uMin To uMax
                            If j - v < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf j - v > lngWidth / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u < 1 Then
                                dblTmp = dblTmp + 0
                            ElseIf i - u > lngHeight / (2 ^ Resolution) Then
                                dblTmp = dblTmp + 0
                            Else
                                dblTmp = dblTmp + Act2D3(j - v, i - u) * dblMask(u, v)
                            End If
                        Next v
                    Next u

                    If Abs(dblTmp - Act2D3(j, i)) < Value Then
                        tmpBo(j, i) = Act2D3(j, i)
                    Else
                        tmpBo(j, i) = Round(dblTmp)
                    End If
                Else
                    tmpGo(j, i) = Act2D2(j, i)
                    tmpBo(j, i) = Act2D3(j, i)
                End If
            End If
        Next j
    Next i
    Act2D1 = tmpRo
    If ChooseKomponen = "All" Then
        Act2D2 = tmpGo
        Act2D3 = tmpBo
    End If
  End Select
End Sub

'Noise reduction with median mask 'Choose="Komponen1"/"All"
Public Sub subIENoiseReductionMedian(Resolution As Integer, ChooseKomponen As String, ChooseIEType
As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim lngCount As Long
Dim dblBoundary As Double
Dim u As Integer
```

```
Dim v As Integer
Dim uM As Integer
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
Dim urutanR() As Double
Dim urutanG() As Double
Dim urutanB() As Double
   ReDim urutanR(0)
   ReDim urutanG(0)
   ReDim urutanB(0)
   urutanR(0) = 0
   urutanG(0) = 0
   urutanB(0) = 0
   ReDim urutanR(1 To 121)
   ReDim urutanG(1 To 121)
   ReDim urutanB(1 To 121)
   Select Case ChooseIEType
   Case "9titik"
      uM = 1
   Case "25titik"
      uM = 2
   Case "49titik"
      uM = 3
   Case "121titik"
      uM = 5
   End Select
   lngCount = 0
   Select Case strDimension
   Case "1D"
      ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
      ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
      ReDim tmpBo(1 To lngWidth, 1 To lngHeight)
      Dim lng2DWidth As Long
      Dim lng2DHeight As Long

      lng2DWidth = 0
      lng2DHeight = 1
      dblBoundary = lngTotalPixel / (2 ^ Resolution)
      For i = 1 To lngTotalPixel
         If i <= dblBoundary Then
            lng2DWidth = lng2DWidth + 1
            tmpRo(lng2DWidth, lng2DHeight) = Act1(i)
            tmpGo(lng2DWidth, lng2DHeight) = Act2(i)
            tmpBo(lng2DWidth, lng2DHeight) = Act3(i)
            If (i Mod lngWidth) = 0 Then
               lng2DHeight = lng2DHeight + 1
               lng2DWidth = 0
            End If
         Else
            lng2DWidth = lngWidth
            lng2DHeight = lng2DHeight - 1
            Exit For
         End If
      Next i
      ReDim Preserve tmpRo(1 To lng2DWidth, 1 To lng2DHeight)
      ReDim Preserve tmpGo(1 To lng2DWidth, 1 To lng2DHeight)
      ReDim Preserve tmpBo(1 To lng2DWidth, 1 To lng2DHeight)

      For i = 1 + uM To (lng2DHeight / (2 ^ Resolution)) - uM
         For j = 1 + uM To (lng2DWidth / (2 ^ Resolution)) - uM
            dblTmpR = 0
            dblTmpG = 0
            dblTmpB = 0
            For u = -uM To uM
               For v = -uM To uM
                  urutanR((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = tmpRo(j - v, i - u)
                  If ChooseKomponen = "All" Then
                     urutanG((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = tmpGo(j - v, i - u)
                     urutanB((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = tmpBo(j - v, i - u)
```

```
                End If
            Next v
        Next u
        For u = 2 To (2 * uM + 1) * (2 * uM + 1)
            For v = (2 * uM + 1) * (2 * uM + 1) To u Step -1
                If urutanR(v) < urutanR(v - 1) Then
                    dblTmpR = urutanR(v)
                    urutanR(v) = urutanR(v - 1)
                    urutanR(v - 1) = dblTmpR
                End If
                If ChooseKomponen = "All" Then
                    If urutanG(v) < urutanG(v - 1) Then
                        dblTmpG = urutanG(v)
                        urutanG(v) = urutanG(v - 1)
                        urutanG(v - 1) = dblTmpG
                    End If
                    If urutanB(v) < urutanB(v - 1) Then
                        dblTmpB = urutanB(v)
                        urutanB(v) = urutanB(v - 1)
                        urutanB(v - 1) = dblTmpB
                    End If
                End If
            Next v
        Next u
        tmpRo(j, i) = urutanR((2 * uM + 1) * (2 * uM + 1) / 2)
        If ChooseKomponen = "All" Then
            tmpGo(j, i) = urutanG((2 * uM + 1) * (2 * uM + 1) / 2)
            tmpBo(j, i) = urutanB((2 * uM + 1) * (2 * uM + 1) / 2)
        End If
    Next j
Next i
Dim tmpRo1() As Double
Dim tmpGo1() As Double
Dim tmpBo1() As Double
ReDim tmpRo1(1 To lngTotalPixel)
ReDim tmpGo1(1 To lngTotalPixel)
ReDim tmpBo1(1 To lngTotalPixel)

lngCount = 0
For i = 1 To lng2DHeight
    For j = 1 To lng2DWidth
        lngCount = lngCount + 1
        tmpRo1(lngCount) = tmpRo(j, i)
        tmpGo1(lngCount) = tmpGo(j, i)
        tmpBo1(lngCount) = tmpBo(j, i)
    Next j
Next i
For i = dblBoundary + 1 To lngTotalPixel
    tmpRo1(i) = Act1(i)
    tmpGo1(i) = Act2(i)
    tmpBo1(i) = Act3(i)
Next i
Act1 = tmpRo1
If ChooseKomponen = "All" Then
    Act2 = tmpGo1
    Act3 = tmpBo1
End If
Case "2D"
    ReDim tmpRo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpGo(1 To lngWidth, 1 To lngHeight)
    ReDim tmpBo(1 To lngWidth, 1 To lngHeight)

    For i = 1 + uM To (lngHeight / (2 ^ Resolution)) - uM
        For j = 1 + uM To (lngWidth / (2 ^ Resolution)) - uM
            dblTmpR = 0
            dblTmpG = 0
            dblTmpB = 0
            For u = -uM To uM
                For v = -uM To uM
                    urutanR((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = Act2D1(j - v, i - u)
```

```
                If ChooseKomponen = "All" Then
                    urutanG((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = Act2D2(j - v, i - u)
                    urutanB((2 * uM + 1) * u + v + ((2 * uM + 1) * (2 * uM + 1) + 1) / 2) = Act2D3(j - v, i - u)
                End If
            Next v
        Next u
        For u = 2 To (2 * uM + 1) * (2 * uM + 1)
            For v = (2 * uM + 1) * (2 * uM + 1) To u Step -1
                If urutanR(v) < urutanR(v - 1) Then
                    dblTmpR = urutanR(v)
                    urutanR(v) = urutanR(v - 1)
                    urutanR(v - 1) = dblTmpR
                End If
                If ChooseKomponen = "All" Then
                    If urutanG(v) < urutanG(v - 1) Then
                        dblTmpG = urutanG(v)
                        urutanG(v) = urutanG(v - 1)
                        urutanG(v - 1) = dblTmpG
                    End If
                    If urutanB(v) < urutanB(v - 1) Then
                        dblTmpB = urutanB(v)
                        urutanB(v) = urutanB(v - 1)
                        urutanB(v - 1) = dblTmpB
                    End If
                End If
            Next v
        Next u
        tmpRo(j, i) = urutanR((2 * uM + 1) * (2 * uM + 1) / 2)
        If ChooseKomponen = "All" Then
            tmpGo(j, i) = urutanG((2 * uM + 1) * (2 * uM + 1) / 2)
            tmpBo(j, i) = urutanB((2 * uM + 1) * (2 * uM + 1) / 2)
        End If
    Next j
Next i
For i = 1 + uM To (lngHeight / (2 ^ Resolution)) - uM
    For j = 1 + uM To (lngWidth / (2 ^ Resolution)) - uM
        Act2D1(j, i) = tmpRo(j, i)
        If ChooseKomponen = "All" Then
            Act2D2(j, i) = tmpGo(j, i)
            Act2D3(j, i) = tmpBo(j, i)
        End If
    Next j
Next i
End Select
End Sub

'Noise reduction with median wavelet filter 'Choose="Komponen1"/"All"
Public Sub subIEWaveletFilter(ChooseKomponen As String, ChooseType As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
Dim lngCount As Long
Dim dblDevMedR As Double
Dim dblDevMedG As Double
Dim dblDevMedB As Double
Dim dblNoiseThresholdR As Double
Dim dblNoiseThresholdG As Double
Dim dblNoiseThresholdB As Double
Dim dblDiv As Double
    dblDiv = 0.6745
    Select Case strDimension
    Case "1D"
        ReDim tmpRo(1 To lngTotalPixel)
        ReDim tmpGo(1 To lngTotalPixel)
        ReDim tmpBo(1 To lngTotalPixel)

        tmpRo = Act1
```

```
    tmpGo = Act2
    tmpBo = Act3

    For i = 2 To lngTotalPixel
       For j = lngTotalPixel To i Step -1
          If Abs(tmpRo(j)) < Abs(tmpRo(j - 1)) Then
             dblTmpR = Abs(tmpRo(j))
             tmpRo(j) = Abs(tmpRo(j - 1))
             tmpRo(j - 1) = dblTmpR
          End If

          If ChooseKomponen = "All" Then
             If Abs(tmpGo(j)) < Abs(tmpGo(j - 1)) Then
                dblTmpG = Abs(tmpGo(j))
                tmpGo(j) = Abs(tmpGo(j - 1))
                tmpGo(j - 1) = dblTmpG
             End If
             If Abs(tmpBo(j)) < Abs(tmpBo(j - 1)) Then
                dblTmpB = Abs(tmpBo(j))
                tmpBo(j) = Abs(tmpBo(j - 1))
                tmpBo(j - 1) = dblTmpB
             End If
          End If
       Next j
    Next i

    dblDevMedR = tmpRo(lngTotalPixel / 2 + 1) / dblDiv
    If ChooseKomponen = "All" Then
       dblDevMedG = tmpGo(lngTotalPixel / 2 + 1) / dblDiv
       dblDevMedB = tmpBo(lngTotalPixel / 2 + 1) / dblDiv
    End If

    dblNoiseThresholdR = dblDevMedR * Sqr(ln(CDbl(lngTotalPixel)))
    If ChooseKomponen = "All" Then
       dblNoiseThresholdG = dblDevMedG * Sqr(ln(CDbl(lngTotalPixel)))
       dblNoiseThresholdB = dblDevMedB * Sqr(ln(CDbl(lngTotalPixel)))
    End If

    For i = 1 To lngTotalPixel
       If ChooseType = "HardThresholding" Then
          If Act1(i) <= dblNoiseThresholdR Then Act1(i) = 0
          If ChooseKomponen = "All" Then
             If Act2(i) <= dblNoiseThresholdG Then Act2(i) = 0
             If Act3(i) <= dblNoiseThresholdB Then Act3(i) = 0
          End If
       ElseIf ChooseType = "SoftThresholding" Then
          If Act1(i) <= dblNoiseThresholdR Then
             Act1(i) = 0
          Else
             Act1(i) = Act1(i) - dblNoiseThresholdR
          End If
          If ChooseKomponen = "All" Then
             If Act2(i) <= dblNoiseThresholdG Then
                Act2(i) = 0
             Else
                Act2(i) = Act2(i) - dblNoiseThresholdG
             End If
             If Act3(i) <= dblNoiseThresholdB Then
                Act3(i) = 0
             Else
                Act3(i) = Act3(i) - dblNoiseThresholdB
             End If
          End If
       End If
    Next i
 Case "2D"
    ReDim tmpRo(1 To lngTotalPixel)
    ReDim tmpGo(1 To lngTotalPixel)
    ReDim tmpBo(1 To lngTotalPixel)
    lngCount = 0
```

```
For i = 1 To lngHeight
   For j = 1 To lngWidth
      lngCount = lngCount + 1
      tmpRo(lngCount) = Act2D1(j, i)
      If ChooseKomponen = "All" Then
         tmpGo(lngCount) = Act2D2(j, i)
         tmpBo(lngCount) = Act2D3(j, i)
      End If
   Next j
Next i

For i = 2 To lngTotalPixel
   For j = lngTotalPixel To i Step -1
      If Abs(tmpRo(j)) < Abs(tmpRo(j - 1)) Then
         dblTmpR = Abs(tmpRo(j))
         tmpRo(j) = Abs(tmpRo(j - 1))
         tmpRo(j - 1) = dblTmpR
      End If

      If ChooseKomponen = "All" Then
         If Abs(tmpGo(j)) < Abs(tmpGo(j - 1)) Then
            dblTmpG = Abs(tmpGo(j))
            tmpGo(j) = Abs(tmpGo(j - 1))
            tmpGo(j - 1) = dblTmpG
         End If
         If Abs(tmpBo(j)) < Abs(tmpBo(j - 1)) Then
            dblTmpB = Abs(tmpBo(j))
            tmpBo(j) = Abs(tmpBo(j - 1))
            tmpBo(j - 1) = dblTmpB
         End If
      End If
   Next j
Next i

dblDevMedR = tmpRo(lngTotalPixel / 2 + 1) / dblDiv
If ChooseKomponen = "All" Then
   dblDevMedG = tmpGo(lngTotalPixel / 2 + 1) / dblDiv
   dblDevMedB = tmpBo(lngTotalPixel / 2 + 1) / dblDiv
End If

dblNoiseThresholdR = dblDevMedR * Sqr(ln(CDbl(lngTotalPixel)))
If ChooseKomponen = "All" Then
   dblNoiseThresholdG = dblDevMedG * Sqr(ln(CDbl(lngTotalPixel)))
   dblNoiseThresholdB = dblDevMedB * Sqr(ln(CDbl(lngTotalPixel)))
End If

For i = 1 To lngHeight
   For j = 1 To lngWidth
      If ChooseType = "HardThresholding" Then
         If Act2D1(j, i) <= dblNoiseThresholdR Then Act2D1(j, i) = 0
         If ChooseKomponen = "All" Then
            If Act2D2(j, i) <= dblNoiseThresholdG Then Act2D2(j, i) = 0
            If Act2D3(j, i) <= dblNoiseThresholdB Then Act2D3(j, i) = 0
         End If
      ElseIf ChooseType = "SoftThresholding" Then
         If Act2D1(j, i) <= dblNoiseThresholdR Then
            Act2D1(j, i) = 0
         Else
            Act2D1(j, i) = Act2D1(j, i) - dblNoiseThresholdR
         End If
         If ChooseKomponen = "All" Then
            If Act2D2(j, i) <= dblNoiseThresholdG Then
               Act2D2(j, i) = 0
            Else
               Act2D2(j, i) = Act2D2(j, i) - dblNoiseThresholdG
            End If
            If Act2D3(j, i) <= dblNoiseThresholdB Then
               Act2D3(j, i) = 0
            Else
               Act2D3(j, i) = Act2D3(j, i) - dblNoiseThresholdB
```

```
                End If
            End If
        End If
    Next j
  Next i
 End Select
End Sub

'Noise reduction with mean wavelet filter 'Choose="Komponen1"/"All"
Public Sub subIEWaveletFilterM(ChooseKomponen As String, ChooseType As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim dblDevMeanR As Double
Dim dblDevMeanG As Double
Dim dblDevMeanB As Double
Dim dblNoiseThresholdR As Double
Dim dblNoiseThresholdG As Double
Dim dblNoiseThresholdB As Double
Dim dblTotalR As Double
Dim dblTotalG As Double
Dim dblTotalB As Double
  dblTotalR = 0
  dblTotalG = 0
  dblTotalB = 0
  Select Case strDimension
  Case "1D"
    ReDim tmpRo(1 To lngTotalPixel)
    ReDim tmpGo(1 To lngTotalPixel)
    ReDim tmpBo(1 To lngTotalPixel)
    tmpRo = Act1
    tmpGo = Act2
    tmpBo = Act3
    subAbsValue tmpRo
    If ChooseKomponen = "All" Then
      subAbsValue tmpGo
      subAbsValue tmpBo
    End If
    For i = 1 To lngTotalPixel
      dblTotalR = dblTotalR + tmpRo(i)
      If ChooseKomponen = "All" Then
        dblTotalG = dblTotalG + tmpGo(i)
        dblTotalB = dblTotalB + tmpBo(i)
      End If
    Next i

    dblDevMeanR = dblTotalR / lngTotalPixel
    If ChooseKomponen = "All" Then
      dblDevMeanG = dblTotalG / lngTotalPixel
      dblDevMeanB = dblTotalB / lngTotalPixel
    End If

    dblNoiseThresholdR = dblDevMeanR * Sqr(ln(CDbl(lngTotalPixel)))
    If ChooseKomponen = "All" Then
      dblNoiseThresholdG = dblDevMeanG * Sqr(ln(CDbl(lngTotalPixel)))
      dblNoiseThresholdB = dblDevMeanB * Sqr(ln(CDbl(lngTotalPixel)))
    End If

    For i = ((lngTotalPixel / 2) + 1) To lngTotalPixel
      If ChooseType = "HardThresholding" Then
        If Act1(i) <= dblNoiseThresholdR Then Act1(i) = 0
        If ChooseKomponen = "All" Then
          If Act2(i) <= dblNoiseThresholdG Then Act2(i) = 0
          If Act3(i) <= dblNoiseThresholdB Then Act3(i) = 0
        End If
      ElseIf ChooseType = "SoftThresholding" Then
        If Act1(i) <= dblNoiseThresholdR Then
          Act1(i) = 0
        Else
          Act1(i) = Act1(i) - dblNoiseThresholdR
```

```
            End If
            If ChooseKomponen = "All" Then
               If Act2(i) <= dblNoiseThresholdG Then
                  Act2(i) = 0
               Else
                  Act2(i) = Act2(i) - dblNoiseThresholdG
               End If
               If Act3(i) <= dblNoiseThresholdB Then
                  Act3(i) = 0
               Else
                  Act3(i) = Act3(i) - dblNoiseThresholdB
               End If
            End If
         End If
      Next i
   Case "2D"
      For i = 1 To lngHeight
         For j = 1 To lngWidth
            dblTotalR = dblTotalR + Act2D1(j, i)
            If ChooseKomponen = "All" Then
               dblTotalG = dblTotalG + Act2D2(j, i)
               dblTotalB = dblTotalB + Act2D3(j, i)
            End If
         Next j
      Next i

      dblDevMeanR = dblTotalR / lngTotalPixel
      If ChooseKomponen = "All" Then
         dblDevMeanG = dblTotalG / lngTotalPixel
         dblDevMeanB = dblTotalB / lngTotalPixel
      End If

      dblNoiseThresholdR = dblDevMeanR * Sqr(ln(CDbl(lngTotalPixel)))
      If ChooseKomponen = "All" Then
         dblNoiseThresholdG = dblDevMeanG * Sqr(ln(CDbl(lngTotalPixel)))
         dblNoiseThresholdB = dblDevMeanB * Sqr(ln(CDbl(lngTotalPixel)))
      End If

      For i = 1 To lngHeight
         For j = 1 To lngWidth
            If ChooseType = "HardThresholding" Then
               If Act2D1(j, i) <= dblNoiseThresholdR Then Act2D1(j, i) = 0
               If ChooseKomponen = "All" Then
                  If Act2D2(j, i) <= dblNoiseThresholdG Then Act2D2(j, i) = 0
                  If Act2D3(j, i) <= dblNoiseThresholdB Then Act2D3(j, i) = 0
               End If
            ElseIf ChooseType = "SoftThresholding" Then
               If Act2D1(j, i) <= dblNoiseThresholdR Then
                  Act2D1(j, i) = 0
               Else
                  Act2D1(j, i) = Act2D1(j, i) - dblNoiseThresholdR
               End If
               If ChooseKomponen = "All" Then
                  If Act2D2(j, i) <= dblNoiseThresholdG Then
                     Act2D2(j, i) = 0
                  Else
                     Act2D2(j, i) = Act2D2(j, i) - dblNoiseThresholdG
                  End If
                  If Act2D3(j, i) <= dblNoiseThresholdB Then
                     Act2D3(j, i) = 0
                  Else
                     Act2D3(j, i) = Act2D3(j, i) - dblNoiseThresholdB
                  End If
               End If
            End If
         Next j
      Next i
   End Select
End Sub
```

```
'Noise reduction with median wavelet filter 'Choose="Komponen1"/"All"
Public Sub subIEWaveletFilterMed(ChooseKomponen As String, Dimension As String, ChooseType As String)
Dim tmpRo() As Double
Dim tmpGo() As Double
Dim tmpBo() As Double
Dim tmpRo1() As Double
Dim tmpGo1() As Double
Dim tmpBo1() As Double
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
Dim lngCount As Long
Dim dblDevMedR As Double
Dim dblDevMedG As Double
Dim dblDevMedB As Double
Dim dblNoiseThresholdR As Double
Dim dblNoiseThresholdG As Double
Dim dblNoiseThresholdB As Double
Dim dblDiv As Double
   dblDiv = 0.6745
   Select Case Dimension
   Case "1D"
      ReDim tmpRo(1 To lngTotalPixel / 2)
      ReDim tmpGo(1 To lngTotalPixel / 2)
      ReDim tmpBo(1 To lngTotalPixel / 2)

      For i = ((lngTotalPixel / 2) + 1) To lngTotalPixel
         tmpRo(i - (lngTotalPixel / 2)) = Act1(i)
         If ChooseKomponen = "All" Then
            tmpGo(i - (lngTotalPixel / 2)) = Act2(i)
            tmpBo(i - (lngTotalPixel / 2)) = Act3(i)
         End If
      Next i

      subAbsValue tmpRo
      If ChooseKomponen = "All" Then
         subAbsValue tmpGo
         subAbsValue tmpBo
      End If

      subQuickSort tmpRo
'      subBubbleSort tmpRo
      If ChooseKomponen = "All" Then
         subQuickSort tmpGo
         subQuickSort tmpBo
'         subBubbleSort tmpGo
'         subBubbleSort tmpBo
      End If

      dblDevMedR = tmpRo((lngTotalPixel / 4) + 1) / dblDiv
      If ChooseKomponen = "All" Then
         dblDevMedG = tmpGo1((lngTotalPixel / 4) + 1) / dblDiv
         dblDevMedB = tmpBo1((lngTotalPixel / 4) + 1) / dblDiv
      End If

      dblNoiseThresholdR = dblDevMedR * Sqr(ln(CDbl(lngTotalPixel / 2)))
      If ChooseKomponen = "All" Then
         dblNoiseThresholdG = dblDevMedG * Sqr(ln(CDbl(lngTotalPixel / 2)))
         dblNoiseThresholdB = dblDevMedB * Sqr(ln(CDbl(lngTotalPixel / 2)))
      End If

      For i = 1 To lngTotalPixel
         If ChooseType = "HardThresholding" Then
            If Act1(i) <= dblNoiseThresholdR Then Act1(i) = 0
            If ChooseKomponen = "All" Then
               If Act2(i) <= dblNoiseThresholdG Then Act2(i) = 0
               If Act3(i) <= dblNoiseThresholdB Then Act3(i) = 0
            End If
         ElseIf ChooseType = "SoftThresholding" Then
```

```
            If Act1(i) <= dblNoiseThresholdR Then
               Act1(i) = 0
            Else
               Act1(i) = Act1(i) - dblNoiseThresholdR
            End If
            If ChooseKomponen = "All" Then
               If Act2(i) <= dblNoiseThresholdG Then
                  Act2(i) = 0
               Else
                  Act2(i) = Act2(i) - dblNoiseThresholdG
               End If
               If Act3(i) <= dblNoiseThresholdB Then
                  Act3(i) = 0
               Else
                  Act3(i) = Act3(i) - dblNoiseThresholdB
               End If
            End If
         End If
      Next i
   Case "2D"
      ReDim tmpRo(1 To lngTotalPixel)
      ReDim tmpGo(1 To lngTotalPixel)
      ReDim tmpBo(1 To lngTotalPixel)
      lngCount = 0
      For i = 1 To lngHeight
         For j = 1 To lngWidth
            lngCount = lngCount + 1
            tmpRo(lngCount) = Act2D1(j, i)
            If ChooseKomponen = "All" Then
               tmpGo(lngCount) = Act2D2(j, i)
               tmpBo(lngCount) = Act2D3(j, i)
            End If
         Next j
      Next i

      For i = 2 To lngTotalPixel
         For j = lngTotalPixel To i Step -1
            If Abs(tmpRo(j)) < Abs(tmpRo(j - 1)) Then
               dblTmpR = Abs(tmpRo(j))
               tmpRo(j) = Abs(tmpRo(j - 1))
               tmpRo(j - 1) = dblTmpR
            End If

            If ChooseKomponen = "All" Then
               If Abs(tmpGo(j)) < Abs(tmpGo(j - 1)) Then
                  dblTmpG = Abs(tmpGo(j))
                  tmpGo(j) = Abs(tmpGo(j - 1))
                  tmpGo(j - 1) = dblTmpG
               End If
               If Abs(tmpBo(j)) < Abs(tmpBo(j - 1)) Then
                  dblTmpB = Abs(tmpBo(j))
                  tmpBo(j) = Abs(tmpBo(j - 1))
                  tmpBo(j - 1) = dblTmpB
               End If
            End If
         Next j
      Next i

      dblDevMedR = tmpRo(lngTotalPixel / 2 + 1) / dblDiv
      If ChooseKomponen = "All" Then
         dblDevMedG = tmpGo(lngTotalPixel / 2 + 1) / dblDiv
         dblDevMedB = tmpBo(lngTotalPixel / 2 + 1) / dblDiv
      End If

      dblNoiseThresholdR = dblDevMedR * Sqr(ln(CDbl(lngTotalPixel)))
      If ChooseKomponen = "All" Then
         dblNoiseThresholdG = dblDevMedG * Sqr(ln(CDbl(lngTotalPixel)))
         dblNoiseThresholdB = dblDevMedB * Sqr(ln(CDbl(lngTotalPixel)))
      End If
```

```
      For i = 1 To lngHeight
         For j = 1 To lngWidth
            If ChooseType = "HardThresholding" Then
               If Act2D1(j, i) <= dblNoiseThresholdR Then Act2D1(j, i) = 0
               If ChooseKomponen = "All" Then
                  If Act2D2(j, i) <= dblNoiseThresholdG Then Act2D2(j, i) = 0
                  If Act2D3(j, i) <= dblNoiseThresholdB Then Act2D3(j, i) = 0
               End If
            ElseIf ChooseType = "SoftThresholding" Then
               If Act2D1(j, i) <= dblNoiseThresholdR Then
                  Act2D1(j, i) = 0
               Else
                  Act2D1(j, i) = Act2D1(j, i) - dblNoiseThresholdR
               End If
               If ChooseKomponen = "All" Then
                  If Act2D2(j, i) <= dblNoiseThresholdG Then
                     Act2D2(j, i) = 0
                  Else
                     Act2D2(j, i) = Act2D2(j, i) - dblNoiseThresholdG
                  End If
                  If Act2D3(j, i) <= dblNoiseThresholdB Then
                     Act2D3(j, i) = 0
                  Else
                     Act2D3(j, i) = Act2D3(j, i) - dblNoiseThresholdB
                  End If
               End If
            End If
         Next j
      Next i
   End Select
End Sub
```

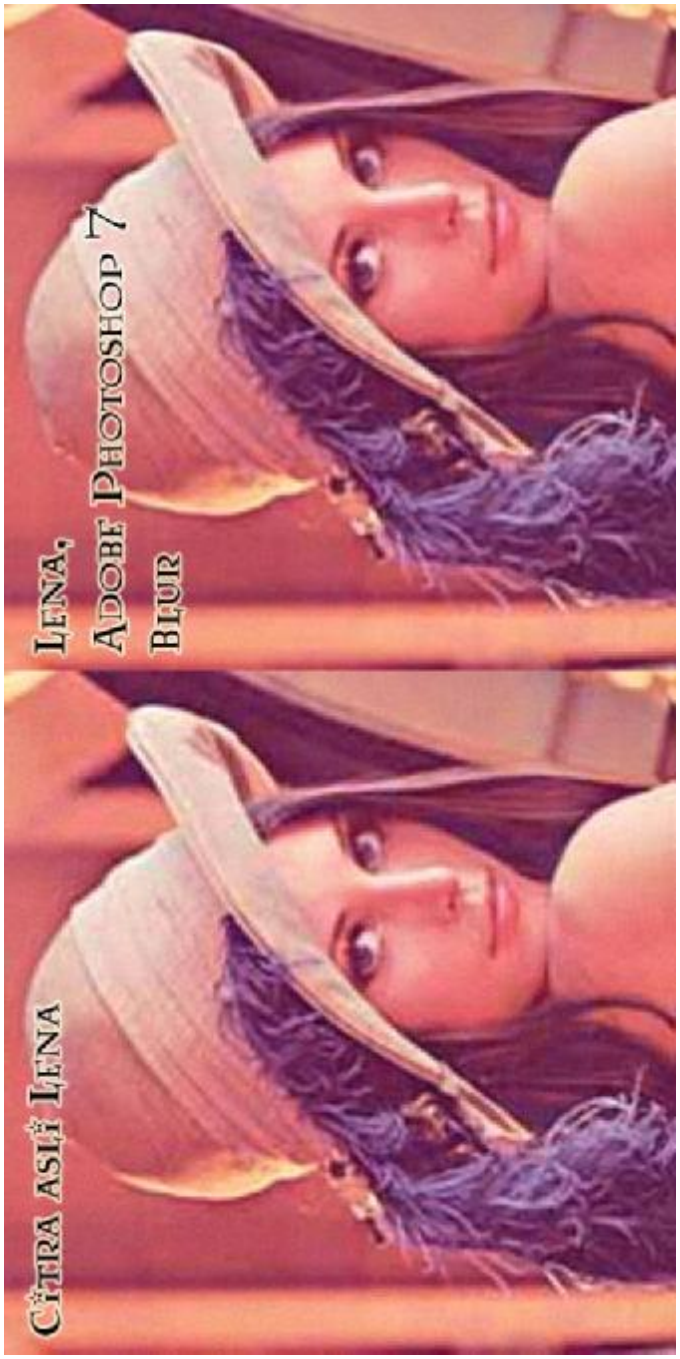
A.10. Fungsi Penghitungan MSE (modMSE.bas)

```
Public Function funcMSE(Dimension As String) As Double
Dim i As Long
Dim j As Long
Dim dblTmpR As Double
Dim dblTmpG As Double
Dim dblTmpB As Double
   dblTmpR = 0
   dblTmpG = 0
   dblTmpB = 0
   Select Case Dimension
   Case "1D"
      For i = 1 To lngTotalPixel
         dblTmpR = dblTmpR + ((colR(i) - Act1(i)) ^ 2)
         dblTmpG = dblTmpG + ((colG(i) - Act2(i)) ^ 2)
         dblTmpB = dblTmpB + ((colB(i) - Act3(i)) ^ 2)
      Next i
   Case "2D"
      For i = 1 To lngHeight
         For j = 1 To lngWidth
            dblTmpR = dblTmpR + ((col2DR(j, i) - Act2D1(j, i)) ^ 2)
            dblTmpG = dblTmpG + ((col2DR(j, i) - Act2D1(j, i)) ^ 2)
            dblTmpB = dblTmpB + ((col2DR(j, i) - Act2D1(j, i)) ^ 2)
         Next j
      Next i
   End Select
   dblTmpR = dblTmpR / lngTotalPixel
   dblTmpG = dblTmpG / lngTotalPixel
   dblTmpB = dblTmpB / lngTotalPixel
   funcMSE = (dblTmpR + dblTmpG + dblTmpB) / 3
End Function
```

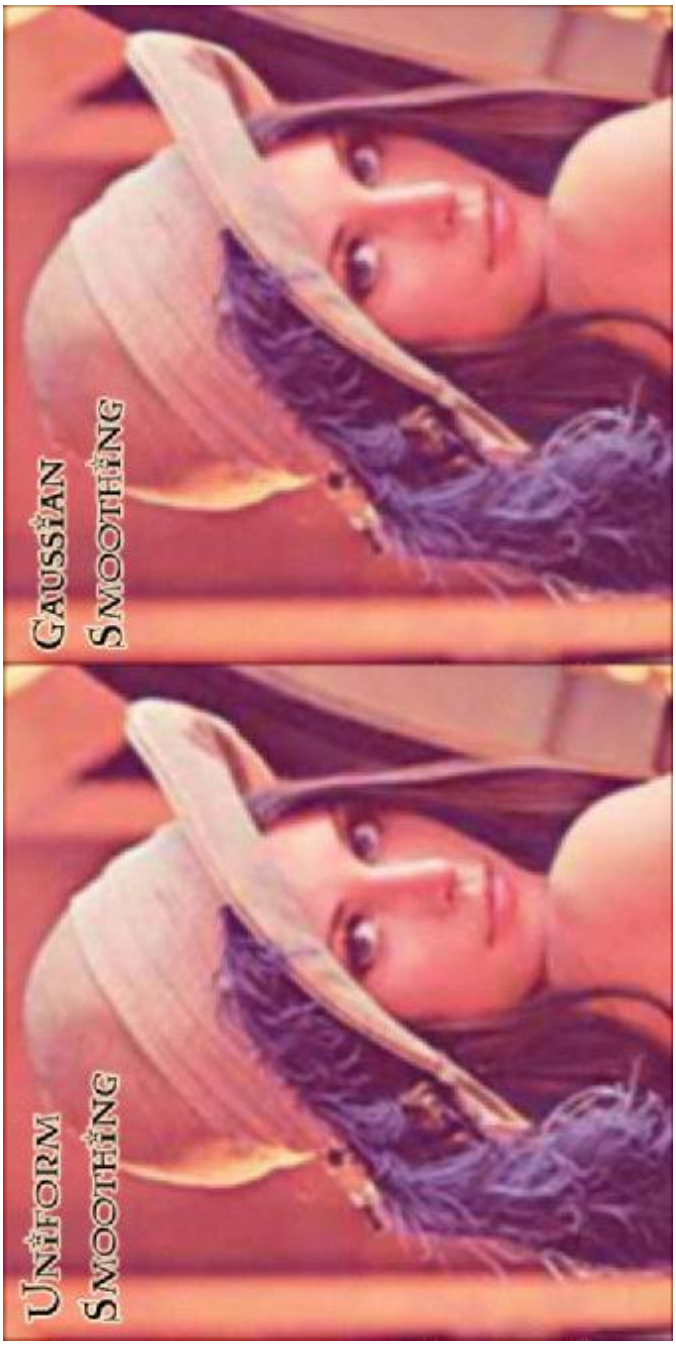
LAMPIRAN B

CITRA ASLI DAN HASIL PENGOLAHANNYA

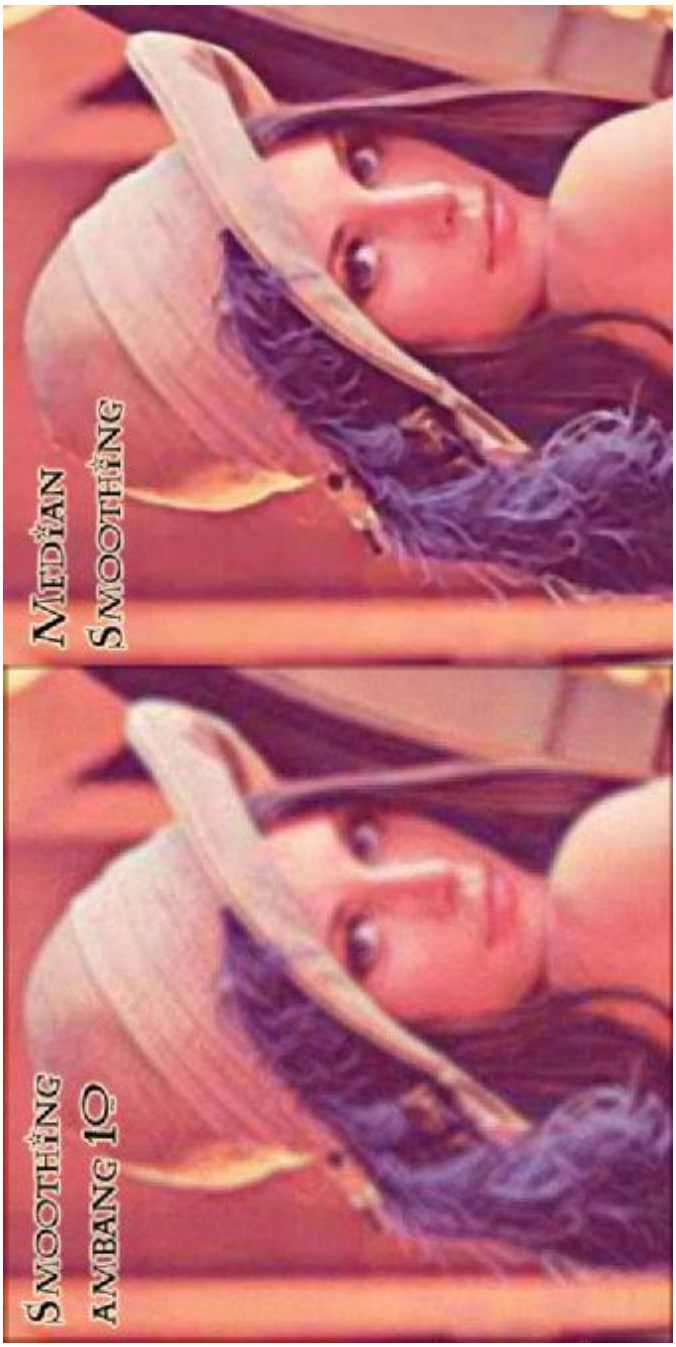
B.1. Citra Asli Lena dan *Blur* dengan Adobe Photoshop 7



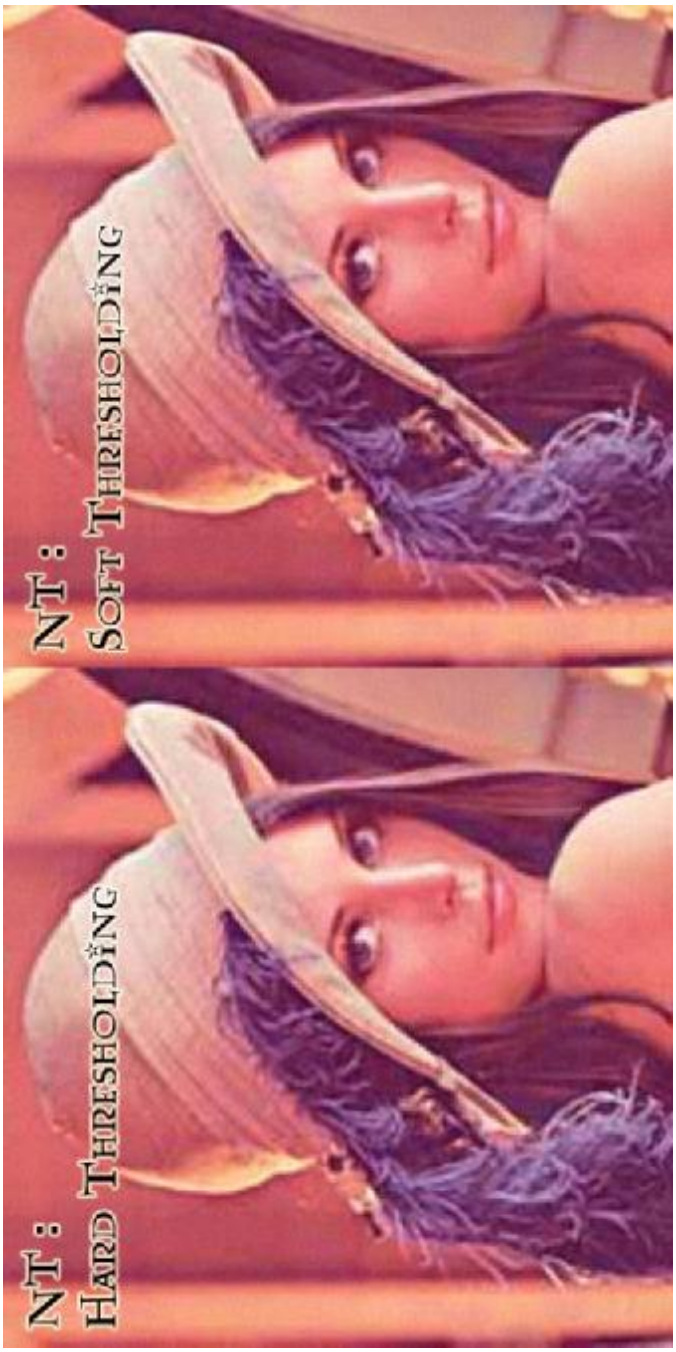
B.2. Citra Lena dengan *Uniform Smoothing* dan *Gaussian Smoothing*



B.3. Citra Lena dengan *Smoothing* Ambang 10 dan *Median Smoothing*



B.4. Citra Lena dengan *Noise Thresholding*; *Hard Thresholding* dan *Soft Thresholding*



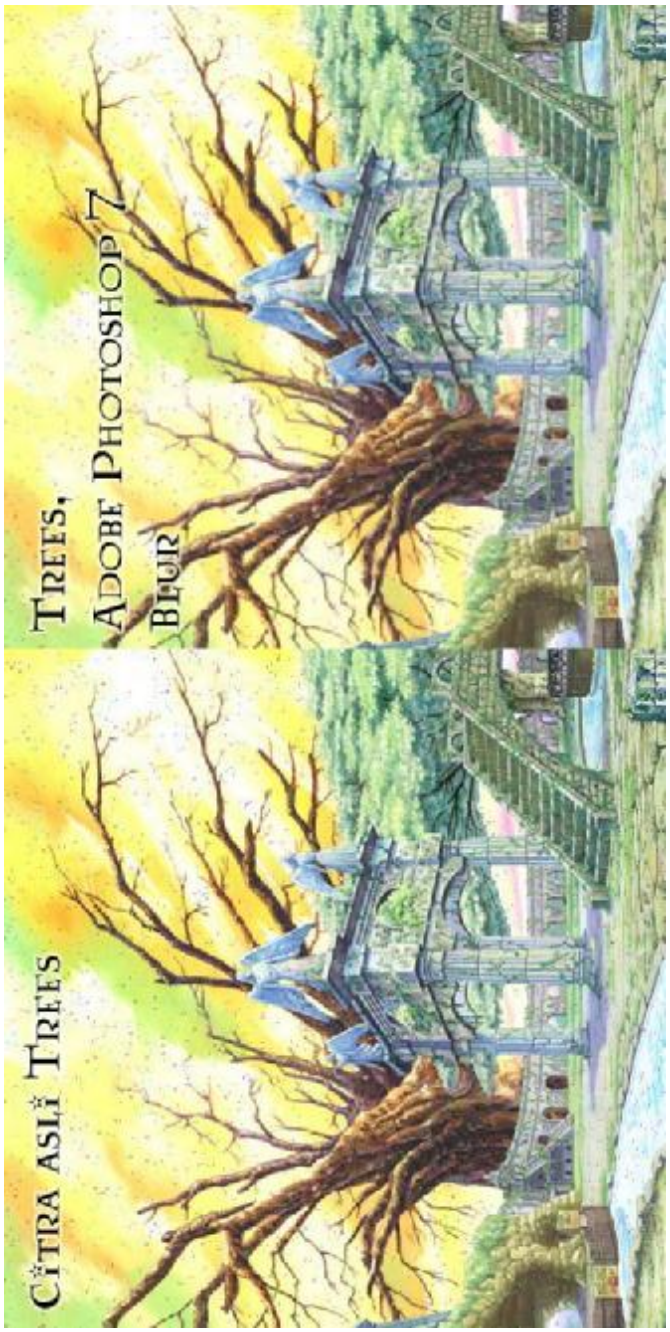
B.5. Citra Asli Cartoon dan *Blur* dengan Adobe Photoshop 7

B.6. Citra Cartoon dengan *Uniform Smoothing* dan *Gaussian Smoothing*

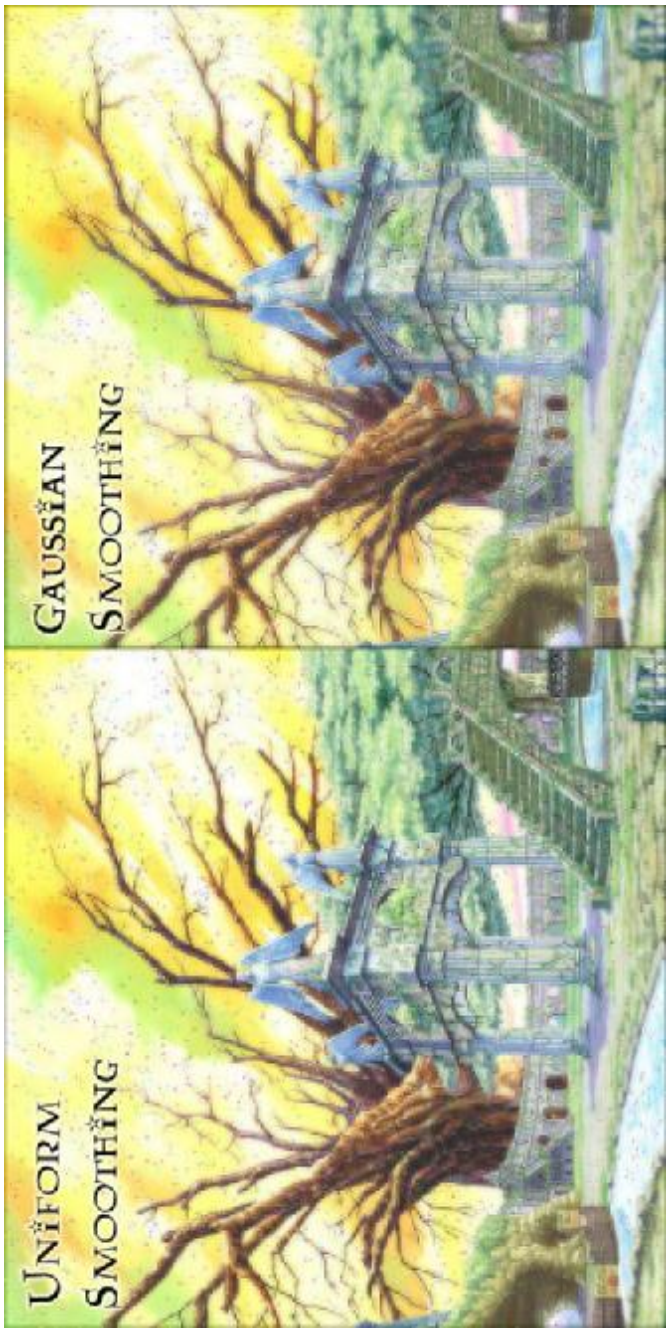
B.7. Citra Cartoon dengan *Smoothing* Ambang 10 dan *Median Smoothing*

B.8. Citra Cartoon dengan *Noise Thresholding*; *Hard Thresholding* dan *Soft Thresholding*

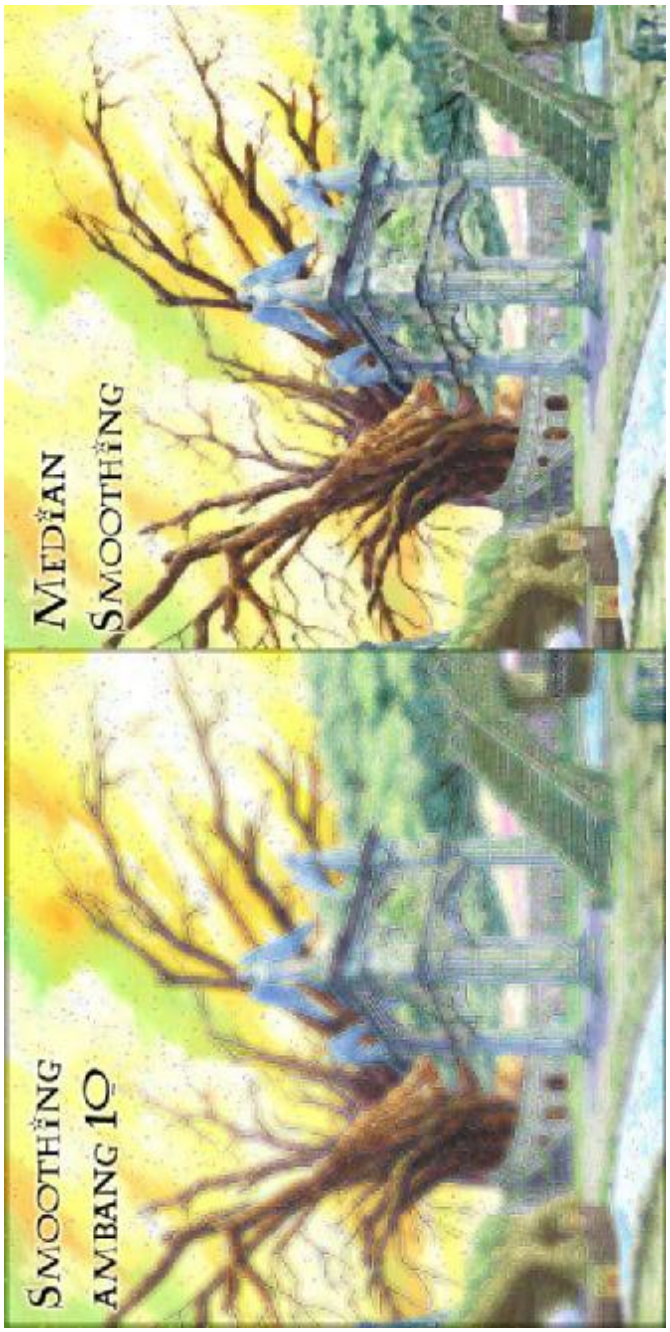
B.9. Citra Asli Trees dan *Blur* dengan Adobe Photoshop 7



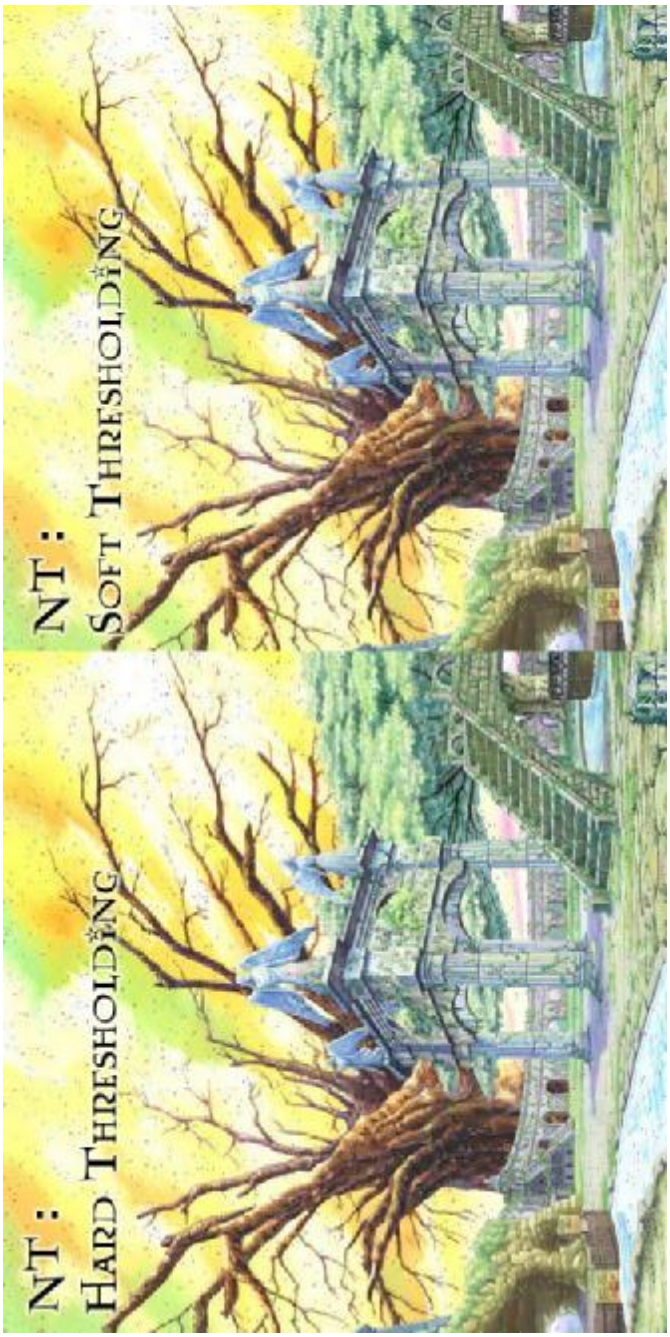
B.10. Citra Trees dengan *Uniform Smoothing* dan *Gaussian Smoothing*



B.11. Citra Trees dengan *Smoothing* Ambang 10 dan *Median Smoothing*



B.12. Citra Trees dengan *Noise Thresholding*; *Hard Thresholding* dan *Soft Thresholding*



LAMPIRAN C

DATA PENGAMATAN HASIL SURVEI

C.1. Survei 1

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.2. Survei 2

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.3. Survei 3

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.4. Survei 4

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.5. Survei 5

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.6. Survei 6

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.7. Survei 7

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.8. Survei 8

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.9. Survei 9

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.10. Survei 10

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.11. Survei 11

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.12. Survei 12

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.13. Survei 13

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.14. Survei 14

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.15. Survei 15

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.16. Survei 16

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.17. Survei 17

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.18. Survei 18

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.19. Survei 19

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.20. Survei 20

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.21. Survei 21

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.22. Survei 22

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.23. Survei 23

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.24. Survei 24

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.25. Survei 25

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.26. Survei 26

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.27. Survei 27

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.28. Survei 28

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.29. Survei 29

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,

C.30. Survei 30

Nama :

Citra Masukan	Metode		Nilai
lena512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Cartoon512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	
Trees512x512NG.bmp	**Multiresolusi**		
		Uniform Smoothing 5 titik	
		Gaussian Smoothing	
		Smoothing dengan nilai ambang = 10	
		Median Smoothing 9 titik	
	Noise Thresholding		
		Hard Thresholding	
		Soft Thresholding	

Keterangan:
Batasan nilai adalah bilangan bulat tanpa pecahan yang berkisar 1 – 5,
1 = paling jelek
5 = paling bagus

Yang menilai,