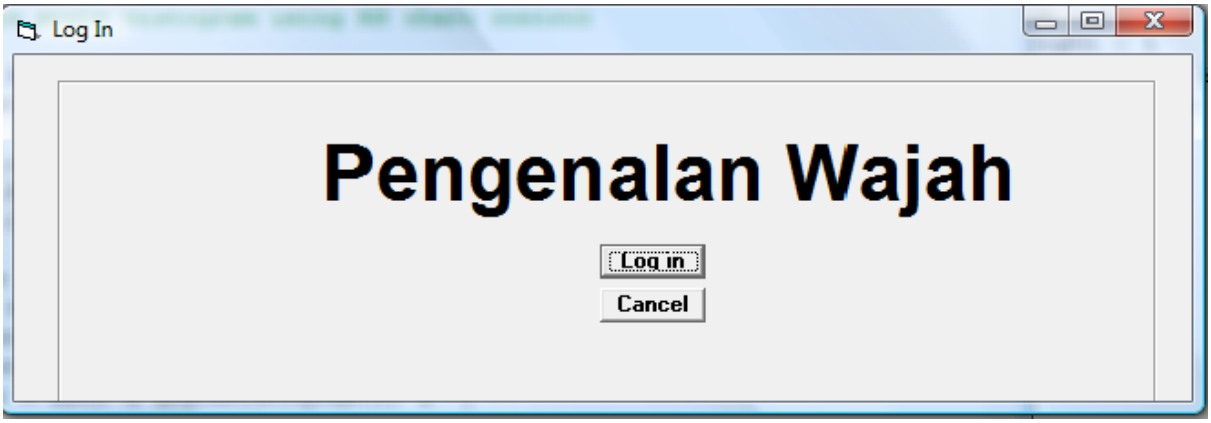
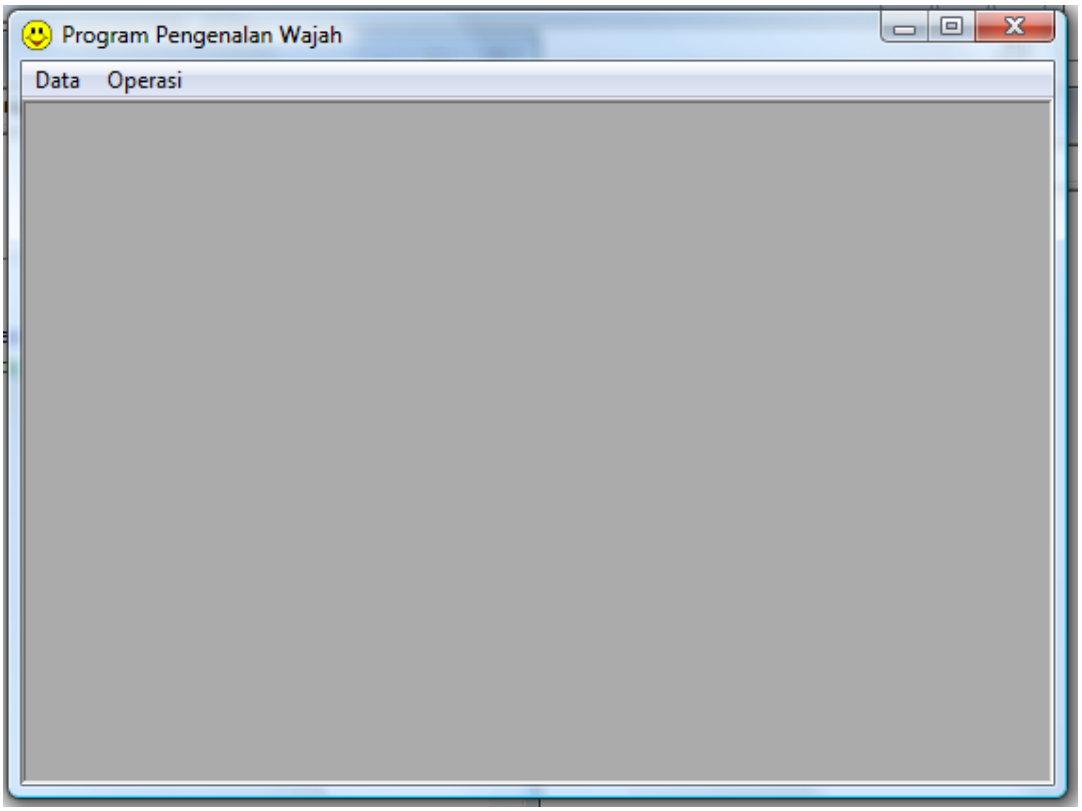
LAMPIRAN A

1. Form Log In

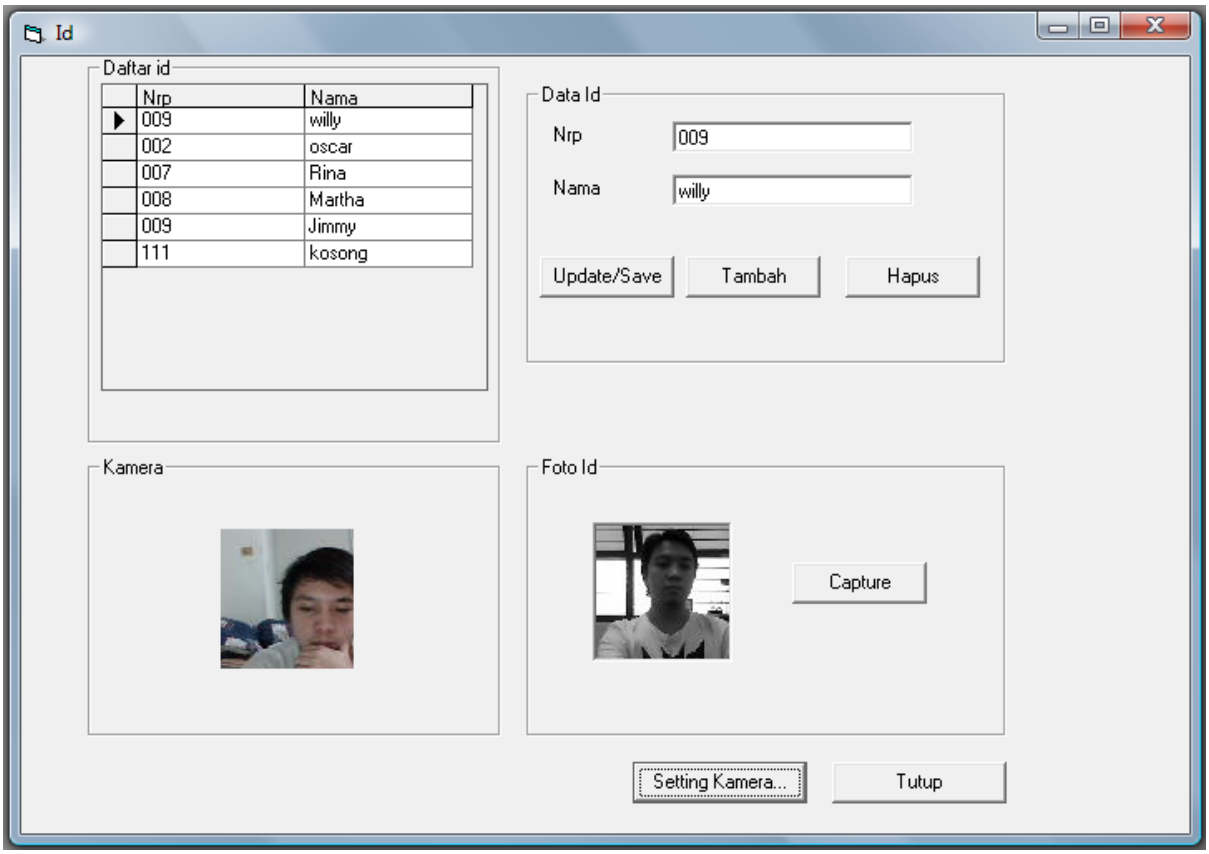


2. Menu

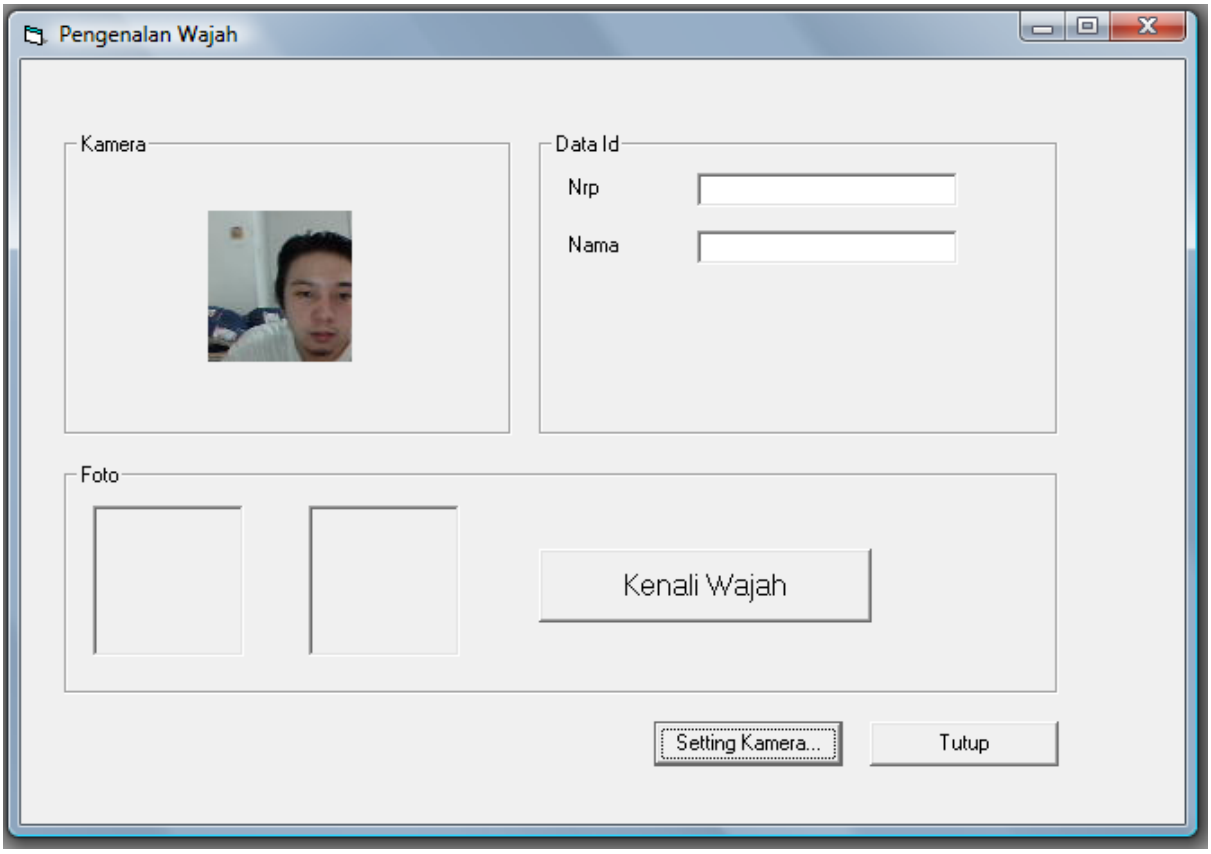


3. Form Setting

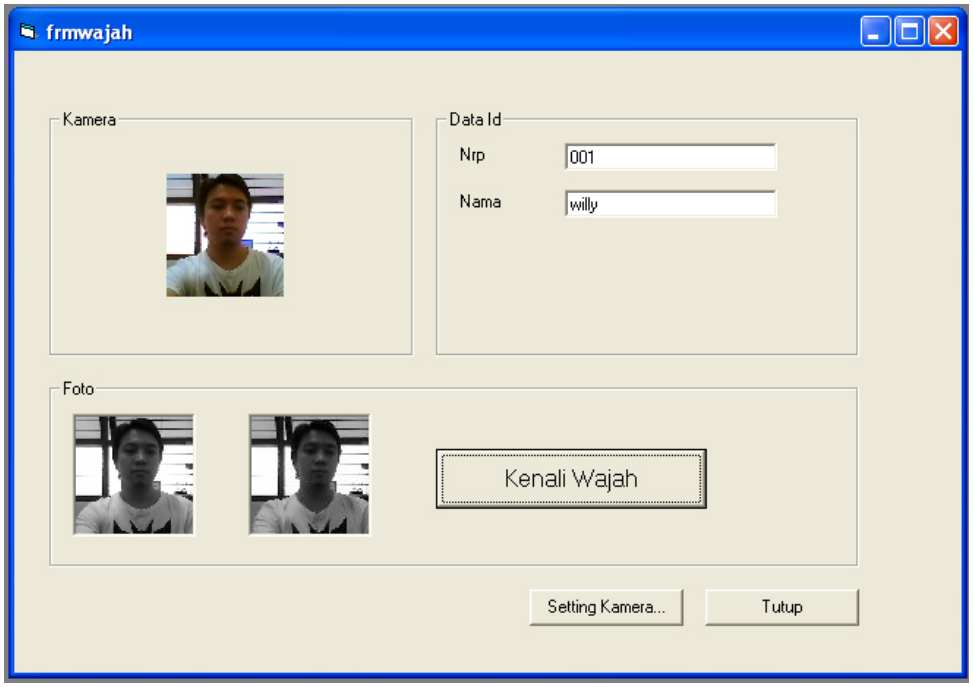
3. Form Input Data

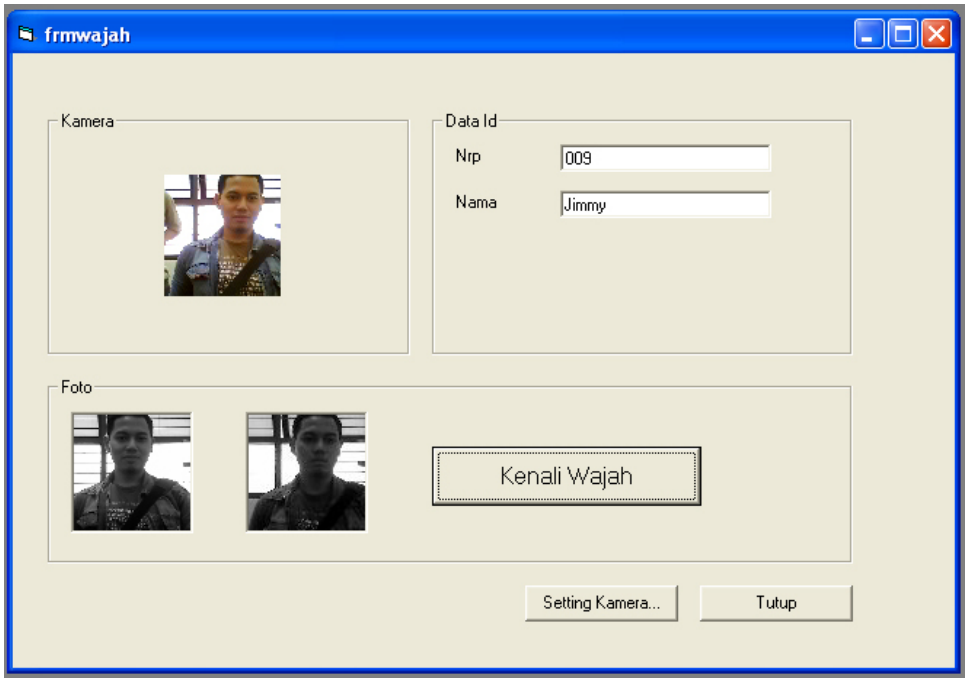


4. Form Pengenalan Wajah



5. Hasil Pengenalan Wajah yang Teridentifikasi

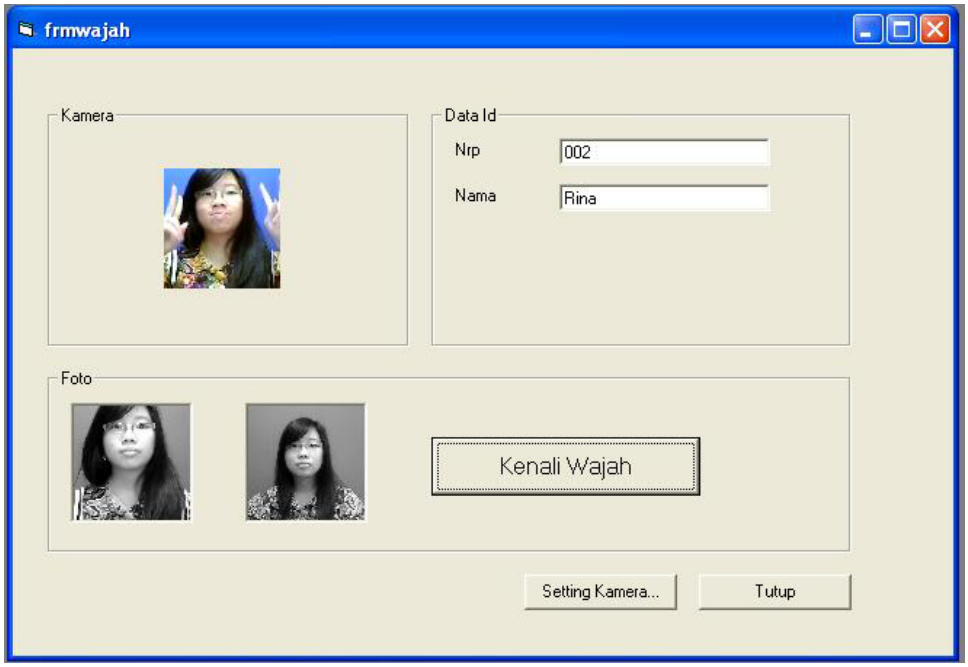



frmwajah
Kamera
Data Id
Nrp
009
Nama
Jimmy
Foto
Kenali Wajah
Setting Kamera...
Tutup


frmwajah
Kamera
Data Id
Nrp
003
Nama
Oscar
Foto
Kenali Wajah
Setting Kamera...
Tutup

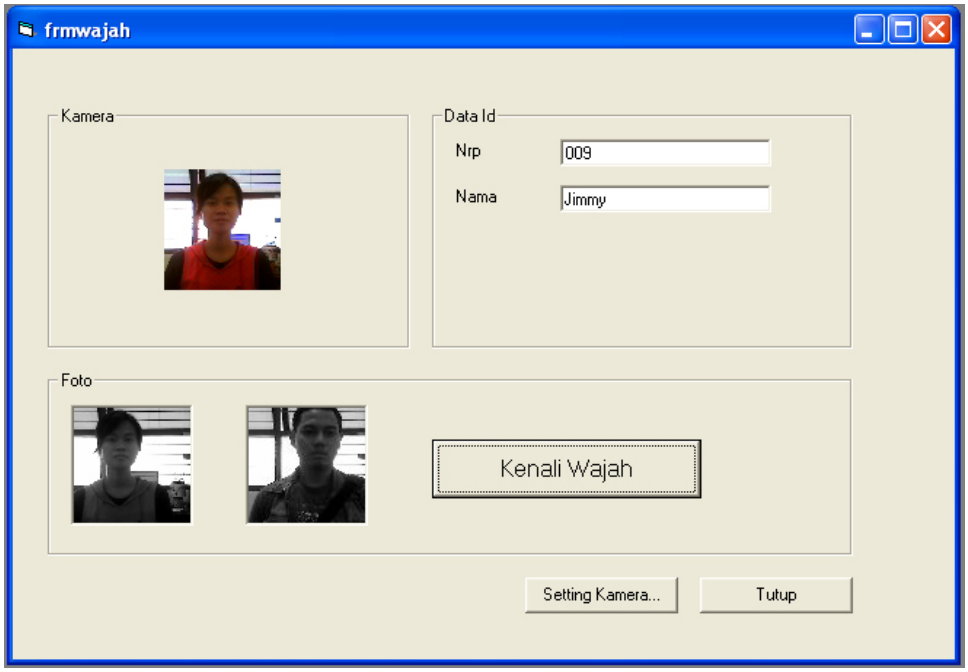

frmwajah
Kamera
Data Id
Nrp
004
Nama
Insan
Foto
Kenali Wajah
Setting Kamera...
Tutup

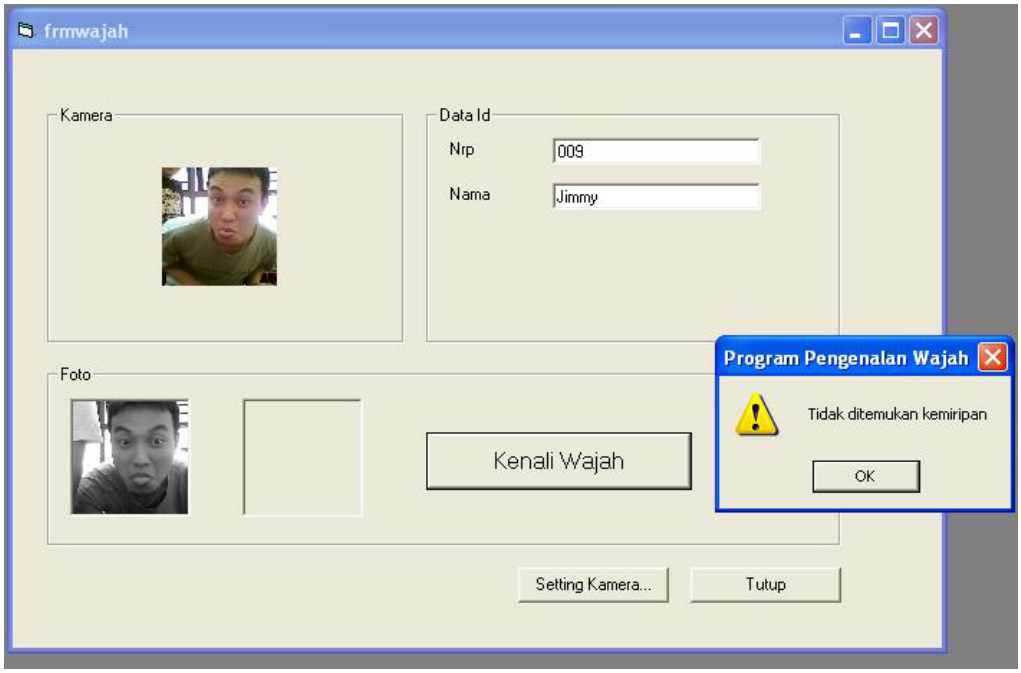

frmwajah
Kamera
Data Id
Nrp
005
Nama
Willy
Foto
Kenali Wajah
Setting Kamera...
Tutup


frmwajah
Kamera
Data Id
Nrp
002
Nama
Rina
Foto
Kenali Wajah
Setting Kamera...
Tutup


frmwajah
Kamera
Data Id
Nrp
001
Nama
Martha
Foto
Kenali Wajah
Setting Kamera...
Tutup

6. Hasil Pengenalan Wajah yang Tidak Teridentifikasi




frmwajah
Kamera
Data Id
Nrp
009
Nama
Jimmy
Foto
Kenali Wajah
Setting Kamera...
Tutup
Program Pengenalan Wajah
Tidak ditemukan kemiripan
OK

LAMPIRAN B

1. Source Code Form LogIn

```
Private Sub cmdCancel_Click()
   Unload Me
End Sub

Private Sub cmdlg_Click()

        Loading
        MDIMain.show
        Unload Me

End Sub

Private Sub Loading()
   Dim fname As String
   Dim lastIndex As Integer
   Dim query As String
   Dim i As Integer
   Dim rsid As ADODB.Recordset

   Picture1.height = 1253
   Picture1.width = 1253
   Picture1.ScaleWidth = 80
   Picture1.ScaleHeight = 80
   Picture1.Visible = True

   Call faces.init(image_Width, image_Height)

   If cn.State <> adStateOpen Then cn.Open strCon

   query = "select id, Nrp, Nama from id"
   Set rsid = New ADODB.Recordset
   rsid.CursorLocation = adUseClient
   rsid.Open query, cn, adOpenStatic, adLockOptimistic

   rsid.MoveLast 'cari index id terakhir
   lastIndex = rsid.Fields(0).value
   rsid.MoveFirst

   ProgressBar1.Visible = True
   ProgressBar1.Min = 0
   ProgressBar1.max = lastIndex
   ProgressBar1.value = 0

   For i = 0 To lastIndex
```

```
        fname = Dir$(App.Path & PictName & i & ".jpg")

        If fname <> "" Then
            Picture1.Picture = LoadPicture(App.Path & PictName & i & ".jpg")
            Call faces.addFace(Picture1, Trim(Str(i)))
        End If
        ProgressBar1.value = i
        DoEvents
    Next
    Loaded = True
End Sub
```

2. Source Code Form id

```
Option Explicit
Dim rsid As ADODB.Recordset


Private Sub cmdAdd_Click()
    If MsgBox("Tambah id?", vbYesNo, "Program Pengenalan Wajah") = vbYes Then
        rsid.AddNew
        Picture6.Picture = Nothing
    End If
End Sub

Private Sub cmdCapture_Click()
    If MsgBox("Ganti foto id?", vbYesNo, "Program Pengenalan Wajah") = vbYes Then
        Picture6.Picture = Nothing
        Capture ezVC
        Crop Picture6, App.Path & CaptName
        Kill App.Path & CaptName
        GrayScale Picture6
        SavePhoto Picture6, rsid.Fields(0).value
        Call faces.addFace(Picture6, rsid.Fields(0).value)
    End If
End Sub

Private Sub cmdClose_Click()
    Unload Me
End Sub

Private Sub cmdDelete_Click()
    If MsgBox("Hapus id ini?", vbYesNo, "Program Pengenalan Wajah") = vbYes Then
        Kill App.Path & PictName & rsid.Fields(0).value & ".jpg"
        rsid.Delete
        rsid.update
```

```
    End If
End Sub

Private Sub cmdSetting_Click()
    If ezVC.HasDlgFormat Then
        ezVC.ShowDlgVideoFormat
    Else
        MsgBox "Kamera ini tidak memiliki setting...", vbOKOnly, "Program Pengenalan Wajah"
    End If
End Sub

Private Sub cmdUpdate_Click()
    On Error Resume Next
    rsid.update
    dgdid.Refresh

End Sub

Private Sub dgdid_RowColChange(LastRow As Variant, ByVal LastCol As Integer)
    On Error Resume Next
    Picture6.Picture = Nothing
    Picture6.Picture = LoadPicture(App.Path & PictName & rsid.Fields(0).value & ".jpg")

End Sub

Private Sub Form_Load()
    On Error Resume Next

    ezVC.width = 2400
    ezVC.height = 1800

    Dim query As String

    If cn.State <> adStateOpen Then cn.Open strCon

    'id
    query = "select id, Nrp, Nama from id"
    Set rsid = New ADODB.Recordset
    rsid.CursorLocation = adUseClient
    rsid.Open query, cn, adOpenDynamic, adLockOptimistic

    dgdid.Columns(0).Caption = "Nrp"
    dgdid.Columns(0).DataField = "Nrp"
    dgdid.Columns(1).Caption = "Nama"
    dgdid.Columns(1).DataField = "Nama"
```

```
    Set dgdid.DataSource = rsid

    txtNrp.DataField = "Nrp"
    Set txtNrp.DataSource = rsid

    txtNama.DataField = "Nama"
    Set txtNama.DataSource = rsid


    Picture6.Picture = LoadPicture(App.Path & PictName & rsid.Fields(0).value & ".jpg")


End Sub

Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next

    rsid.Close

End Sub
```

3. Source Code Form Setting

```
Option Explicit

Dim rsSetting As ADODB.Recordset

Private Sub cmdClose_Click()
    Unload Me
End Sub

Private Sub cmdSave_Click()
    rsSetting.update
    MinimumEigen = Int(txtMinEigen.Text)
    MsgBox "Data telah disimpan", vbInformation, "Program Pengenalan Wajah"
End Sub

Private Sub Form_Load()
    Dim query As String

    query = "select nilai from setting where nama='MinEigen'"
    Set rsSetting = New ADODB.Recordset
    rsSetting.CursorLocation = adUseClient
    rsSetting.Open query, cn, adOpenStatic, adLockOptimistic

    txtMinEigen.DataField = "nilai"
```

```
    Set txtMinEigen.DataSource = rsSetting
End Sub
```

4. Source Code Form Pengenalan Wajah

```
Option Explicit
Dim rsid As ADODB.Recordset
Private Sub cmdwajah_Click()
    On Error Resume Next
    Dim index As Integer

        'stage 1 - capture camera
        Capture ezVC
        Crop Picture6, App.Path & CaptName
        Kill App.Path & CaptName
        GrayScale Picture6

        'stage 2 - matching picture
        index = MatchPict(Picture6, Picture5)

        'stage 3 - fill the form
        rsid.MoveFirst
        rsid.Find "id=" & index
        txtNrp.Text = rsid.Fields(1).value
        txtNama.Text = rsid.Fields(2).value


End Sub
```

5. Source Code Data Module

```
Public Const dbname = "id.mdb"
Public Const strCon = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & dbname &
";Persist Security Info=False"
Public cn As New ADODB.Connection
```

6. Source Code Mengubah Ekstensi .bmp Menjadi .jpg

```
Public Declare Function getDesktop Lib "JPGUtils.dll" (ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal blnJpeg As Boolean, ByVal JPGCompressQuality As Integer, ByVal strFileName As String) As Integer
Public Declare Function ConvertBMPtoJPG Lib "JPGUtils.dll" (ByVal strFileName As String, ByVal JPGCompressQuality As Integer, ByVal blnKeepBMP As Boolean) As Integer
Public Declare Function ConvertJPGtoBMP Lib "JPGUtils.dll" (ByVal strFileName As String, ByVal blnKeepJPG As Boolean) As Integer
```

7. Source Code Library

```
Option Explicit

Public Const CaptName = "/capt.bmp"
Public Const PictName = "/images/pict"

Dim ImagePixels(0 To 2, 0 To 120, 0 To 160) As Integer
Dim pixel As Long
Dim x, y As Integer
Dim i, j As Integer
Dim red, green, blue, gray As Integer

Public faces As New ClassFaceRecogniser
Public Const image_Width = 80
Public Const image_Height = 80

Public UserName As String
Public UserType As String
Public Loaded As Boolean
Public MinimumEigen As Integer


Public Sub Capture(ezVC As ezVidCap)
   On Error Resume Next
   ezVC.CapSingleFrame
   If (ezVC.SaveDIB(App.Path & CaptName)) Then
      'MsgBox "Gambar tercapture"
   Else
      MsgBox "Gagal"
   End If
   ezVC.Preview = True
End Sub

Public Sub Crop(pict As PictureBox, filename As String)
   On Error Resume Next
```

```
    pict.Picture = LoadPicture(filename)

    x = pict.ScaleWidth
    y = pict.ScaleHeight


    For i = 20 To 99
        For j = 40 To 119
            pixel = pict.Point(j, i)
            red = pixel& Mod 256
            green = ((pixel And &HFF00) / 256&) Mod 256&
            blue = (pixel And &HFF0000) / 65536
            ImagePixels(0, i, j) = red
            ImagePixels(1, i, j) = green
            ImagePixels(2, i, j) = blue
        Next
    Next

    pict.height = 1253
    pict.width = 1253

    pict.ScaleWidth = 80
    pict.ScaleHeight = 80

    pict.Picture = Nothing

    For i = 20 To 99
        For j = 40 To 119
            pict.PSet (j - 40, i - 20), RGB(ImagePixels(0, i, j), ImagePixels(1, i, j), ImagePixels(2, i,
j))
        Next
        pict.Refresh
    Next

End Sub

Public Sub GrayScale(pict As PictureBox)
    On Error Resume Next

    x = pict.ScaleWidth
    y = pict.ScaleHeight

    For i = 0 To y - 1
        For j = 0 To x - 1
```

```
            pixel = pict.Point(j, i)
            red = pixel& Mod 256
            green = ((pixel And &HFF00) / 256&) Mod 256&
            blue = (pixel And &HFF0000) / 65536

            ' my algorithm to grayscale picture
            ' is it correct?

            red = (red * 5) \ 10
            green = (green * 8) \ 10
            blue = (blue * 3) \ 10
            gray = ((red + green + blue) * 10) \ 16

            pict.PSet (j, i), RGB(gray, gray, gray)
        Next
        pict.Refresh
    Next

End Sub

Public Sub SavePhoto(pict As PictureBox, index As Integer)
    On Error Resume Next
    SavePicture pict.image, App.Path & PictName & index & ".bmp"
    ConvertBMPtoJPG App.Path & PictName & index & ".bmp", 80, False
    MsgBox "File telah tersimpan", , App.Title
End Sub

Public Function MatchPict(pict1 As PictureBox, pict2 As PictureBox) As Integer
    Dim closestImage As Integer

    pict2.Picture = Nothing
    closestImage = Val(faces.Identify(pict1))

    If ((closestImage <> 0) And (Dir$(App.Path & PictName & closestImage & ".jpg") <> "")) 
Then
        pict2.Picture = LoadPicture(App.Path & PictName & closestImage & ".jpg")
    Else
        MsgBox "Tidak ditemukan kemiripan", vbExclamation, "Program Pengenalan Wajah"
    End If
    MatchPict = closestImage
End Function
```

8. Source Code ClassFaceRecognisier

```
Option Explicit

Public imageWidth As Integer
Public imageHeight As Integer

Public NoOfFaces As Integer
Dim Face(1000) As classImageProcessing
Dim EigenFace(1000) As classImageProcessing
Dim NameOfFace(1000) As String

Dim faceTemplate() As Single

Dim testFace As classImageProcessing
Dim testEigenFace As classImageProcessing

Public Identity As classImageProcessing


Public Sub init(image_Width As Integer, image_Height As Integer)
  imageWidth = image_Width
  imageHeight = image_Height
  ReDim faceTemplate(imageWidth, imageHeight)
End Sub


Public Sub addFace(facePicture As PictureBox, faceName As String)
  Set Face(NoOfFaces) = New classImageProcessing
  Set EigenFace(NoOfFaces) = New classImageProcessing
  Set testFace = New classImageProcessing
  Set testEigenFace = New classImageProcessing
  Call Face(NoOfFaces).init(imageWidth, imageHeight)
  Call EigenFace(NoOfFaces).init(imageWidth, imageHeight)
  Call testFace.init(imageWidth, imageHeight)
  Call testEigenFace.init(imageWidth, imageHeight)

  Call Face(NoOfFaces).update(facePicture)
  NameOfFace(NoOfFaces) = faceName
  NoOfFaces = NoOfFaces + 1

  Call updateFaceTemplate
  Call updateEigenFaces
End Sub
```

```
Private Sub updateFaceTemplate()
'calculates an average face template
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer

  For i = 0 To NoOfFaces - 1
    For x = 0 To imageWidth - 1
      For y = 0 To imageHeight - 1
        If (i > 0) Then
          faceTemplate(x, y) = faceTemplate(x, y) + Face(i).getPoint(x, y)
          Else
          faceTemplate(x, y) = Face(i).getPoint(x, y)
        End If
      Next
    Next
  Next

  For x = 0 To imageWidth - 1
    For y = 0 To imageHeight - 1
      faceTemplate(x, y) = Int(faceTemplate(x, y) / NoOfFaces)
    Next
  Next
End Sub


Private Sub updateEigenFaces()
'updates all the eigenfaces
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer
  Dim df As Integer

  For i = 0 To NoOfFaces - 1
    For x = 0 To imageWidth - 1
      For y = 0 To imageHeight - 1
        df = Face(i).getPoint(x, y) - faceTemplate(x, y)
        If (df < 0) Then
          df = 0
        End If
        Call EigenFace(i).setPoint(x, y, CByte(df))
      Next
    Next
  Next
End Sub
```

```
Public Function Identify(facePicture As PictureBox) As String
'identifies the given image
  Dim i As Integer
  Dim x As Integer
  Dim y As Integer
  Dim df As Integer
  Dim Distance As Long
  Dim minDistance As Long
  Dim retval As String
  Dim a As Integer
  Dim b As Integer

  retval = ""
  Call testFace.update(facePicture)

  'calculate the eigenface
  For x = 0 To imageWidth - 1
    For y = 0 To imageHeight - 1
      df = testFace.getPoint(x, y) - faceTemplate(x, y)
      If (df < 0) Then
        df = 0
      End If
      Call testEigenFace.setPoint(x, y, CByte(df))
    Next
  Next

  'compare it to other eigenfaces
  minDistance = ((100 - MinimumEigen) * 99999999#) / 100

  'MsgBox minDistance

  For i = 0 To NoOfFaces - 1
    Distance = 0
    For x = 0 To imageWidth - 1
      For y = 0 To imageHeight - 1
        a = EigenFace(i).getPoint(x, y)
        b = testEigenFace.getPoint(x, y)
        df = Abs(a - b)
        Distance = Distance + df
      Next
    Next

    If (Distance < minDistance) Then
```

```
      minDistance = Distance
      retval = NameOfFace(i)
      Set Identity = Face(i)
    End If
  Next

  Identify = retval

End Function
```

9. Source Code ClassImageProcessing

```
Option Explicit

Public width As Integer
Public height As Integer
Dim image() As Byte

Dim edgeTraced() As Boolean
Dim temp() As Boolean
Public TraceEdgesThresh As Integer
Public minEdgeLength As Integer
Dim traceDirection As Single
Dim traceRadius As Integer
Dim traceX As Single
Dim traceY As Single

Dim angleHistogram(18) As Integer

Public edgesWidth As Integer
Public edgesHeight As Integer
Dim Edges() As Byte

Public processType As Integer

Public EdgeThreshold As Single
Dim averageContrast As Double

Const IMAGE_RAW = 0
Const IMAGE_RED = 1
Const IMAGE_GREEN = 2
Const IMAGE_BLUE = 3
Const IMAGE_EDGES = 4
Const IMAGE_MOVEMENT = 5
```

```
'masks used for edge detection
Const NO_OF_EDGE_MASKS = 14
Dim EdgeMask(NO_OF_EDGE_MASKS)
Const NO_OF_EDGE_TYPES = 5
Dim EdgeHistogram(NO_OF_EDGE_TYPES) As Integer

Const EDGE_VECTOR_LENGTH = 200
Dim EdgeVector(5, EDGE_VECTOR_LENGTH) As Single
Dim currEdgeVector As Integer
Dim maxEdgeVectorIntensity As Integer



Private Function traceSearch(Optional beginSearch As Boolean) As Boolean
'move the trace point in a curcular motion until a new feature is found
'returns TRUE when a new feature is located

  Dim tx As Integer
  Dim ty As Integer

  traceSearch = False

  If (beginSearch) Then
    traceDirection = 0
    traceRadius = 90
  End If

  traceX = traceX + Cos((traceDirection / 180) * 3.14)
  traceY = traceY + Sin((traceDirection / 180) * 3.14)
  traceDirection = traceDirection + traceRadius
  If (traceDirection > 360) Then
    traceDirection = 0
    traceRadius = traceRadius - 1
    If (traceRadius < 0) Then
      traceRadius = 0
    End If
  End If

  If (traceX < 0) Then
    traceX = 0
  End If
  If (traceX >= width) Then
    traceX = width - 1
  End If
```

```
  If (traceY < 0) Then
    traceY = 0
  End If
  If (traceY >= height) Then
    traceY = height - 1
  End If

  tx = Int(traceX)
  ty = Int(traceY)

  If ((image(tx, ty) > TraceEdgesThresh) And (Not edgeTraced(tx, ty))) Then
    traceSearch = True
  End If
End Function


Private Sub calcAngleHistogram()
'calculates a histogram from the angles of edge traces
  Dim i As Integer
  Dim dx As Integer
  Dim dy As Integer
  Dim length As Integer
  Dim angle As Single
  Dim intensity As Single


  For i = 0 To 17
    angleHistogram(i) = 0
  Next

  For i = 0 To currEdgeVector - 1
    dx = EdgeVector(0, i) - EdgeVector(2, i)
    dy = Abs(EdgeVector(1, i) - EdgeVector(3, i))
    length = Sqr((dx * dx) + (dy * dy))
    If (length > 0) Then
      angle = (Cos(dy / length) / 3.14) * 180
      If (dx < 0) Then
        angle = 180 - angle
      End If
      angle = Int(angle / 10)
      intensity = 1 'EdgeVector(4, i) / 255
      angleHistogram(angle) = angleHistogram(angle) + (length * intensity)
    End If
  Next
```

```
End Sub


Private Sub initEdgeMasks()
'defines edge masks
' 1 = horizontal
' 2 = vertical
' 3 = diagonal left
' 4 = diagonal right
' 5 = cross

  Dim mask
  Dim i As Integer
  Dim mstr As String

  'Lines -
  EdgeMask(0) = Array(1, 1, 1, _
                      0, 0, 0, _
                      0, 0, 0, _
                      1)
  EdgeMask(1) = Array(0, 0, 0, _
                      1, 1, 1, _
                      0, 0, 0, _
                      1)
  EdgeMask(2) = Array(0, 0, 0, _
                      0, 0, 0, _
                      1, 1, 1, _
                      1)
  'Lines |
  EdgeMask(3) = Array(1, 0, 0, _
                      1, 0, 0, _
                      1, 0, 0, _
                      2)
  EdgeMask(4) = Array(0, 1, 0, _
                      0, 1, 0, _
                      0, 1, 0, _
                      2)
  EdgeMask(5) = Array(0, 0, 1, _
                      0, 0, 1, _
                      0, 0, 1, _
                      2)
  'Diagonals
  EdgeMask(6) = Array(0, 0, 1, _
                      0, 1, 0, _
                      1, 0, 0, _
```

```
                3)
  EdgeMask(7) = Array(0, 1, 0, _
                1, 0, 0, _
                0, 0, 0, _
                3)
  EdgeMask(8) = Array(0, 0, 0, _
                0, 0, 1, _
                0, 1, 0, _
                3)
  EdgeMask(9) = Array(1, 0, 0, _
                0, 1, 0, _
                0, 0, 1, _
                4)
  EdgeMask(10) = Array(0, 1, 0, _
                0, 0, 1, _
                0, 0, 0, _
                4)
  EdgeMask(11) = Array(0, 0, 0, _
                1, 0, 0, _
                0, 1, 0, _
                4)
  'Crosses
  EdgeMask(12) = Array(1, 0, 1, _
                0, 1, 0, _
                1, 0, 1, _
                5)
  EdgeMask(13) = Array(0, 1, 0, _
                1, 1, 1, _
                0, 1, 0, _
                5)
  'nothing
  'EdgeMask(14) = Array(0, 0, 0, _
  '                0, 0, 0, _
  '                0, 0, 0, _
  '                0) 'last number indicates edge type
  'EdgeMask(15) = Array(1, 1, 1, _
  '                1, 1, 1, _
  '                1, 1, 1, _
  '                0)
End Sub


Private Sub initEdgeMasks_old()
'defines edge masks
' 0 = horizontal
```

```
' 1 = vertical
' 2 = diagonal left
' 3 = diagonal right
' 4 = cross

  Dim mask
  Dim i As Integer
  Dim mstr As String

  'Lines -
  EdgeMask(0) = Array(1, 1, 1, _
                      0, 0, 0, _
                      0, 0, 0, _
                      1)
  EdgeMask(1) = Array(0, 0, 0, _
                      1, 1, 1, _
                      0, 0, 0, _
                      1)
  EdgeMask(2) = Array(0, 0, 0, _
                      0, 0, 0, _
                      1, 1, 1, _
                      1)
  'Lines double -
  EdgeMask(3) = Array(1, 1, 1, _
                      1, 1, 1, _
                      0, 0, 0, _
                      1)
  EdgeMask(4) = Array(0, 0, 0, _
                      1, 1, 1, _
                      1, 1, 1, _
                      1)
  'Lines |
  EdgeMask(5) = Array(1, 0, 0, _
                      1, 0, 0, _
                      1, 0, 0, _
                      2)
  EdgeMask(6) = Array(0, 1, 0, _
                      0, 1, 0, _
                      0, 1, 0, _
                      2)
  EdgeMask(7) = Array(0, 0, 1, _
                      0, 0, 1, _
                      0, 0, 1, _
                      2)
  EdgeMask(8) = Array(1, 1, 0, _
```

```
                1, 1, 0, _
                1, 1, 0, _
                2)
EdgeMask(9) = Array(0, 1, 1, _
                0, 1, 1, _
                0, 1, 1, _
                2)
'Diagonals
EdgeMask(10) = Array(0, 0, 1, _
                0, 1, 0, _
                1, 0, 0, _
                3)
EdgeMask(11) = Array(0, 0, 1, _
                0, 1, 1, _
                1, 1, 0, _
                3)
EdgeMask(12) = Array(0, 1, 1, _
                1, 1, 0, _
                1, 0, 0, _
                3)
EdgeMask(13) = Array(1, 0, 0, _
                0, 1, 0, _
                0, 0, 1, _
                4)
EdgeMask(14) = Array(1, 1, 0, _
                0, 1, 1, _
                0, 0, 1, _
                4)
EdgeMask(15) = Array(1, 0, 0, _
                1, 1, 0, _
                0, 1, 1, _
                4)
'Crosses
EdgeMask(16) = Array(1, 0, 1, _
                0, 1, 0, _
                1, 0, 1, _
                5)
EdgeMask(17) = Array(0, 1, 0, _
                1, 1, 1, _
                0, 1, 0, _
                5)
'nothing
EdgeMask(18) = Array(0, 0, 0, _
                0, 0, 0, _
                0, 0, 0, _
```

```
                0) 'last number indicates edge type
  EdgeMask(19) = Array(1, 1, 1, _
                1, 1, 1, _
                1, 1, 1, _
                0)
End Sub


Public Sub traceEdges()
'traces edges within the image
  Dim finished As Boolean
  Dim x As Integer
  Dim y As Integer
  Dim traced As Boolean

  finished = False
  traced = False
  x = 0
  y = 0
  While (Not finished)
    x = x + 1
    If (x = width) Then
      y = y + 1
      x = 0
    End If
    If (y < height) Then
      If ((edgeTraced(x, y) = False) And (image(x, y) > TraceEdgesThresh)) Then
        traced = traceEdgesFromPoint(x, y, 0)
      End If
      Else
      x = 0
      y = 0
      If (Not traced) Then
        finished = True
      End If
      traced = False
    End If
  Wend

  Call sortEdgeVector
  Call calcAngleHistogram

End Sub
```

```
Public Sub traceEdges_old()
'traces edges within the image

  Dim x As Integer
  Dim y As Integer

  traceX = 0
  traceY = 0
  Call traceSearch(True)
  While (traceRadius > 0)
    If (traceSearch()) Then
      traceRadius = 90
      x = Int(traceX)
      y = Int(traceY)
      If (traceEdgesFromPoint(x, y, 0)) Then
        traceX = x
        traceY = y
      End If
    End If
  Wend

  'Call sortEdgeVector
  Call calcAngleHistogram

End Sub


Private Sub diffuseEdges()
'diffuses edges information
'this allows edge tracing to be more noise tollerant
  Dim x As Integer
  Dim y As Integer
  Dim i As Integer
  Dim value As Integer

  For i = 0 To 1
    For x = 1 To width - 2
      For y = 1 To height - 2
        If (image(x, y) > TraceEdgesThresh) Then
          image(x, y) = 255

          'value = image(x - 1, y - 1)
          'value = value + image(x - 1, y)
          'value = value + image(x - 1, y + 1)
          'value = value + image(x + 1, y - 1)
```

```
        'value = value + image(x + 1, y)
        'value = value + image(x + 1, y + 1)
        'value = value + image(x, y + 1)
        'value = value + image(x, y - 1)
        'value = value / 8
        'image(x, y) = value
       End If
     Next
   Next
 Next

End Sub


Public Function traceEdgesFromPoint(ByRef x As Integer, ByRef y As Integer, ByRef edgeLength As Integer) As Boolean
'traces along edges starting at the given point
  Dim i As Integer
  Dim j As Integer
  Dim sx As Integer
  Dim sy As Integer
  Dim xx As Integer
  Dim yy As Integer
  Dim pathFound As Boolean
  Dim initialEdgeLength As Integer
  Dim mindirection As Single
  Dim maxdirection As Single
  Dim initialX As Integer
  Dim initialY As Integer
  Dim max As Integer
  Dim value As Integer
  Dim intensity As Single
  Dim direction As Integer
  Static averagedirection As Single
  Dim directionDifference As Integer
  Dim thresh As Integer

  initialX = x
  initialY = y
  xx = initialX
  yy = initialY
  initialEdgeLength = edgeLength
  intensity = 0
  thresh = 0 ' TraceEdgesThresh / 2
```

```
If (initialEdgeLength = 0) Then
  For i = 0 To width - 1
    For j = 0 To height - 1
      temp(i, j) = False
    Next
  Next
End If

averagedirection = 0
traceEdgesFromPoint = False
While ((image(xx, yy) > thresh) And (temp(xx, yy) = False))
  sx = xx
  sy = yy
  temp(xx, yy) = True
  edgeLength = edgeLength + 1
  If (edgeTraced(xx, yy) = False) And (edgeLength > minEdgeLength) Then
    traceEdgesFromPoint = True
  End If
  pathFound = False
  max = 0

  If (sy > 0) Then
    value = image(sx, sy - 1)
    If ((value > thresh) And (temp(sx, sy - 1) = False)) Then
      If (value > max) And ((averagedirection > 270) Or (averagedirection < 90)) Then
        max = value
        xx = sx
        yy = sy - 1
        direction = 0
      End If
    End If
  End If

  If (sx < width - 1) Then

    If (sy > 0) Then
      value = image(sx + 1, sy - 1)
      If ((value > thresh) And (temp(sx + 1, sy - 1) = False)) Then
        If (value > max) And ((averagedirection > 315) And (averagedirection < 135)) Then
          max = value
          xx = sx
          yy = sy - 1
          direction = 45
        End If
      End If
```

```
  End If

  value = image(sx + 1, sy)
  If ((value > thresh) And (temp(sx + 1, sy) = False)) Then
    If (value > max) And ((averagedirection > 0) And (averagedirection < 180)) Then
      max = value
      xx = sx + 1
      yy = sy
      direction = 90
    End If
  End If


  If (sy < height - 1) Then
    value = image(sx + 1, sy + 1)
    If ((value > thresh) And (temp(sx + 1, sy + 1) = False)) Then
      If (value > max) And ((averagedirection > 45) And (averagedirection < 225)) Then
        max = value
        xx = sx
        yy = sy + 1
        direction = 135
      End If
    End If
  End If

End If

If (sy < height - 1) Then
  value = image(sx, sy + 1)
  If ((value > thresh) And (temp(sx, sy + 1) = False)) Then
    If (value > max) And ((averagedirection > 90) And (averagedirection < 270)) Then
      max = value
      xx = sx
      yy = sy + 1
      direction = 180
    End If
  End If
End If

If (sx > 0) Then

  If (sy < height - 1) Then
    value = image(sx - 1, sy + 1)
    If ((value > thresh) And (temp(sx - 1, sy + 1) = False)) Then
      If (value > max) And ((averagedirection > 135) And (averagedirection < 315)) Then
```

```
        max = value
        xx = sx - 1
        yy = sy + 1
        direction = 225
       End If
     End If
   End If

   value = image(sx - 1, sy)
   If ((value > thresh) And (temp(sx - 1, sy) = False)) Then
     If (value > max) And ((averagedirection > 180) Or (averagedirection = 0)) Then
       max = value
       xx = sx - 1
       yy = sy
       direction = 270
     End If
   End If

   If (sy > 0) Then
     value = image(sx - 1, sy - 1)
     If ((value > thresh) And (temp(sx - 1, sy - 1) = False)) Then
       If (value > max) And ((averagedirection > 225) Or (averagedirection < 45)) Then
         max = value
         xx = sx - 1
         yy = sy - 1
         direction = 315
       End If
     End If
   End If
 End If

 If (averagedirection > 0) Then
   intensity = (intensity + max) / 2
   directionDifference = Abs(averagedirection - direction)
   If (directionDifference > 180) Then
     directionDifference = 360 - directionDifference
   End If
   averagedirection = averagedirection - (directionDifference / 2)
   If (averagedirection < 0) Then
     averagedirection = 360 + averagedirection
   End If
   If (averagedirection > 360) Then
     averagedirection = averagedirection - 360
   End If
```

```
    If ((edgeLength > 3) And (directionDifference > 20) And (traceEdgesFromPoint)) Then
      Call addEdgeVector(initialX, initialY, xx, yy, intensity)
      initialX = xx
      initialY = yy
    End If

    Else
    intensity = max
    averagedirection = direction
  End If

 Wend

 If (traceEdgesFromPoint = True) Then
  Call addEdgeVector(initialX, initialY, xx, yy, intensity)
 End If

 If (initialEdgeLength = 0) Then
  'If (edgeLength > minEdgeLength) Then
   For i = 0 To width - 1
    For j = 0 To height - 1
     If (temp(i, j) = True) Then
      edgeTraced(i, j) = True
     End If
    Next
   Next
  'End If
 End If

 x = xx
 y = yy
 traceDirection = direction

End Function


Private Sub addEdgeVector(x1 As Integer, y1 As Integer, x2 As Integer, y2 As Integer, intensity
As Single)
'adds a new edge vector

  If (currEdgeVector < EDGE_VECTOR_LENGTH) Then
   EdgeVector(0, currEdgeVector) = x1
   EdgeVector(1, currEdgeVector) = y1
   EdgeVector(2, currEdgeVector) = x2
   EdgeVector(3, currEdgeVector) = y2
```

```
    EdgeVector(4, currEdgeVector) = intensity
    If (intensity > maxEdgeVectorIntensity) Then
      maxEdgeVectorIntensity = intensity
    End If
    currEdgeVector = currEdgeVector + 1
  End If
End Sub


Private Sub sortEdgeVector()
'sorts the edge vector by distance

  Dim dx As Integer
  Dim dy As Integer
  Dim length As Long
  Dim mindist As Long
  Dim closest As Integer
  Dim vect As Single
  Dim i As Integer
  Dim j As Integer

  For i = 0 To currEdgeVector - 2
    mindist = 99999
    closest = 0
    For j = i + 1 To currEdgeVector - 1
      dx = EdgeVector(2, i) - EdgeVector(0, j)
      dy = EdgeVector(3, i) - EdgeVector(1, j)
      length = (dx * dx) + (dy * dy)
      If (length < mindist) Then
        mindist = length
        closest = j
      End If
    Next
    If ((closest > 0) And (closest <> i + 1)) Then
      'swap
      For j = 0 To 4
        vect = EdgeVector(j, i + 1)
        EdgeVector(j, i + 1) = EdgeVector(j, closest)
        EdgeVector(j, closest) = vect
      Next
    End If
  Next
```

```
  'For i = 0 To currEdgeVector - 1
  '  For j = 0 To currEdgeVector - 1
  '    If (i <> j) Then
  '      dx = EdgeVector(0, i) - EdgeVector(0, j)
  '      dy = EdgeVector(1, i) - EdgeVector(1, j)
  '      length = ((dx * dx) + (dy * dy))
  '      If (length < 3 * 3) Then
  '        EdgeVector(0, i) = EdgeVector(0, i) - (dx / 2)
  '        EdgeVector(1, i) = EdgeVector(1, i) - (dy / 2)
  '        EdgeVector(0, j) = EdgeVector(0, j) + (dx / 2)
  '        EdgeVector(1, j) = EdgeVector(1, j) + (dy / 2)
  '      End If
  '      dx = EdgeVector(2, i) - EdgeVector(2, j)
  '      dy = EdgeVector(3, i) - EdgeVector(3, j)
  '      length = ((dx * dx) + (dy * dy))
  '      If (length < 3 * 3) Then
  '        EdgeVector(2, i) = EdgeVector(2, i) - (dx / 2)
  '        EdgeVector(3, i) = EdgeVector(3, i) - (dy / 2)
  '        EdgeVector(2, j) = EdgeVector(2, j) + (dx / 2)
  '        EdgeVector(3, j) = EdgeVector(3, j) + (dy / 2)
  '      End If
  '    End If
  '  Next
  'Next

End Sub


Private Function dist(x1 As Single, y1 As Single, x2 As Single, y2 As Single) As Single
  Dim dx As Single
  Dim dy As Single

  dx = x1 = x2
  dy = y1 - y2
  dist = Sqr((dx * dx) + (dy * dy))
End Function



Public Sub getEdges()
'updates the edges

  Dim mask
  Dim i As Integer
  Dim j As Integer
```

```
Dim x As Integer
Dim y As Integer
Dim xx As Integer
Dim yy As Integer
Dim diff As Long
Dim thresh As Integer
Dim diff2 As Long
Dim estr As String
Dim minDiff As Long
Dim winner As Integer
Dim ex As Integer
Dim ey As Integer
Dim av As Integer

thresh = 100

For i = 0 To NO_OF_EDGE_TYPES - 1
  EdgeHistogram(i) = 0
Next

x = 0
ex = 0
While (x < width - 2)
  y = 0
  ey = 0
  While (y < height - 2)
    Edges(ex, ey) = 0
    minDiff = 9999999
    winner = -1
    For i = 0 To NO_OF_EDGE_MASKS - 1
      mask = EdgeMask(i)
      diff = 0
      j = 0
      av = 0
      For yy = y To y + 2
        For xx = x To x + 2
          av = av + image(xx, yy)
          diff2 = Abs((mask(j) * 255) - image(xx, yy))
          diff = diff + diff2
          j = j + 1
        Next
      Next
      If (av / 9 > 30) Then

        'edge
```

```
          diff = diff / 9
          If (diff < minDiff) And (diff < thresh) Then
            winner = mask(9)
            minDiff = diff
            Edges(ex, ey) = winner
          End If

          Else

          'blank
          winner = 0
          Edges(ex, ey) = winner

        End If
      Next
      'Edges(ex, ey) = Rnd * 5  'test
      If (winner > 0) Then
        EdgeHistogram(winner - 1) = EdgeHistogram(winner - 1) + 1
      End If
      ey = ey + 1
      y = y + 2
    Wend
    ex = ex + 1
    x = x + 2
  Wend

  'fill in the gaps
  Call getEdges_secondary

End Sub


Public Sub getEdges_secondary()
'fills in edges where they "should" appear
  Dim x As Integer
  Dim y As Integer

  For x = 1 To edgesWidth - 1
    For y = 1 To edgesHeight - 1
      'horizontal
      If ((Edges(x - 1, y) > 0) And (Edges(x + 1, y) > 0)) Then
        Edges(x, y) = 1
        Else
        'vertical
```

```
        If ((Edges(x, y - 1) > 0) And (Edges(x, y + 1) > 0)) Then
          Edges(x, y) = 2
          Else
         'diagonal
         If ((Edges(x - 1, y - 1) > 0) And (Edges(x + 1, y + 1) > 0)) Then
           'Edges(x, y) = 4
           Else
           'diagonal
           If ((Edges(x + 1, y - 1) > 0) And (Edges(x - 1, y + 1) > 0)) Then
             'Edges(x, y) = 3
           End If
         End If
        End If
      End If

      If ((Edges(x + 1, y) <> 1) And (Edges(x + 1, y) = Edges(x, y))) Then
        Edges(x, y) = 0
      End If
      If ((Edges(x, y + 1) <> 2) And (Edges(x, y + 1) = Edges(x, y))) Then
        Edges(x, y) = 0
      End If

      'surrounded by edges
      If ((Edges(x - 1, y - 1) > 0) And (Edges(x - 1, y) > 0) And (Edges(x - 1, y + 1) > 0) And
(Edges(x, y - 1) > 0) And (Edges(x, y + 1) > 0) And (Edges(x + 1, y - 1) > 0) And (Edges(x + 1, y)
> 0) And (Edges(x + 1, y + 1) > 0)) Then
        Edges(x, y) = 0
      End If

    Next
  Next

End Sub


Public Sub init(imageWidth As Integer, imageHeight As Integer)
  width = imageWidth
  height = imageHeight
  ReDim image(width, height)

  ReDim edgeTraced(width, height)
  ReDim temp(width, height)
  minEdgeLength = 10

  edgesWidth = width / 2
```

```
  edgesHeight = height / 2
  ReDim Edges(edgesWidth, edgesHeight)
  EdgeThreshold = 0
  processType = 0
  Call initEdgeMasks
  averageContrast = 1
  ReDim picked(width, height)
End Sub


Private Sub calcEdgeVector()
'calculates the edge vector for the image
  Dim i As Integer

  For i = 0 To EDGE_VECTOR_LENGTH - 1

  Next

End Sub


Public Sub whiteNoise()
  Dim x As Integer
  Dim y As Integer

  For x = 0 To width - 1
    For y = 0 To height - 1
      image(x, y) = Rnd * 255
    Next
  Next
End Sub


Public Function getPoint(x As Integer, y As Integer) As Byte
  getPoint = image(x, y)
End Function


Public Function setPoint(x As Integer, y As Integer, value As Byte)
  image(x, y) = value
End Function


Public Sub update(canvas As PictureBox, Optional left As Variant, Optional top As Variant,
Optional wdth As Variant, Optional hght As Variant)
```

```
'import a picture
'processtype = 0   greyscale
'              1   red
'              2   green
'              3   blue
'              4   edges
'              5   movement

  Dim x As Integer
  Dim y As Integer
  Dim screenX As Integer
  Dim screenY As Integer
  Dim w As Integer
  Dim h As Integer
  Dim xx As Integer
  Dim yy As Integer
  Dim value As Double
  Dim RGBval As Long
  Dim pixels As Double
  Dim maxCol As Long
  Dim edgeValue As Single
  Dim screenWidth As Single
  Dim screenHeight As Single
  Dim screenLeft As Single
  Dim screenTop As Single

  If (Not IsMissing(left)) And (Not IsMissing(top)) Then
    screenLeft = left
    screenTop = top
    screenWidth = wdth
    screenHeight = hght
    Else
    screenLeft = 0
    screenTop = 0
    screenWidth = canvas.ScaleWidth
    screenHeight = canvas.ScaleHeight
  End If

  w = CInt(screenWidth / width)
  If (w < 1) Then
    w = 1
  End If
  h = CInt(screenHeight / height)
  If (h < 1) Then
    h = 1
```

```
  End If

  pixels = w * h
  maxCol = RGB(255, 255, 255)
  For x = 0 To width - 1
    For y = 0 To height - 1
      edgeTraced(x, y) = False
      screenX = screenLeft + ((x / width) * screenWidth)
      screenY = screenTop + ((y / height) * screenHeight)
      value = 0
      For xx = screenX To screenX + w - 1
        For yy = screenY To screenY + h - 1

          RGBval = canvas.Point(xx, yy)
          Select Case processType
            Case 0  'greyscale
              value = value + (RGBval / maxCol)
            Case 1  'red
              value = value + ((RGBval And 255) / 255)
            Case 2  'green
              value = value + ((RGBval And 65280) / 65280)
            Case 3  'blue
              value = value + ((RGBval And 16711680) / 16711680)
          End Select

        Next
      Next
      value = (value / pixels) * 255

      image(x, y) = value
    Next
  Next

End Sub


Public Sub getImageEdges(rawImage As classImageProcessing)
'extracts edges from the given image
  Dim x As Integer
  Dim y As Integer
  Dim value As Single
  Dim scalex As Single
  Dim scaley As Single
  Dim xx As Integer
  Dim yy As Integer
```

```
Dim p1 As Integer
Dim p2 As Integer
Dim avContrast As Double

scalex = rawImage.width / width
scaley = rawImage.height / height

currEdgeVector = 0
maxEdgeVectorIntensity = 0

avContrast = 0
For x = 1 To width - 1
  For y = 1 To height - 1
    edgeTraced(x, y) = False
    xx = x * scalex
    yy = y * scaley
    If ((xx >= 1) And (yy >= 1)) Then
      p1 = rawImage.getPoint(xx, yy)
      p2 = rawImage.getPoint(xx - 1, yy)
      value = Abs(p1 - p2)
      p2 = rawImage.getPoint(xx, yy - 1)
      value = value + Abs(p1 - p2)
      value = value / (255 * 2)
      avContrast = avContrast + value
      'If (Abs(value - averageContrast) < EdgeThreshold) Then
      If (value < EdgeThreshold) Then
        value = 0
        Else
        value = 255 * value
      End If
      image(x, y) = value
    End If
  Next
Next

'calc average contast
avContrast = avContrast / (width * height)
averageContrast = avContrast
If (averageContrast < 0.01) Then
  averageContrast = 0.01
End If

'calc threshold used for tracing along edges
TraceEdgesThresh = (averageContrast * 255) * 0.1
```

```
  'Call diffuseEdges

  'Call getEdges
  Call traceEdges

End Sub


Public Sub getImageContours(rawImage As classImageProcessing)
'extracts edges from the given image
  Dim x As Integer
  Dim y As Integer
  Dim value As Single
  Dim scalex As Single
  Dim scaley As Single
  Dim xx As Integer
  Dim yy As Integer
  Dim p1 As Integer
  Dim p2 As Integer
  Dim value2 As Single
  Dim max As Single

  scalex = rawImage.width / width
  scaley = rawImage.height / height

  currEdgeVector = 0
  maxEdgeVectorIntensity = 0
  max = 1 - EdgeThreshold

  For x = 1 To width - 1
    For y = 1 To height - 1
      edgeTraced(x, y) = False
      xx = x * scalex
      yy = y * scaley
      If ((xx >= 1) And (yy >= 1)) Then
        p1 = rawImage.getPoint(xx, yy)
        p2 = rawImage.getPoint(xx - 1, yy)
        value = Abs(p1 - p2)
        p2 = rawImage.getPoint(xx, yy - 1)
        value = value + Abs(p1 - p2)
        value = value / (255 * 2)
        value2 = value - EdgeThreshold
        If (value2 < 0) Then
          value = 0
          Else
```

```
        value = 255 - (255 * (value2 / max))
       End If
       image(x, y) = value
      End If
    Next
  Next

End Sub



Public Sub show(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer

  If (processType <> 4) Then

  canvas.FillStyle = 0
  For x = 0 To width - 1
    For y = 0 To height - 1
      value = image(x, y)
      Select Case processType
        Case 1 'red
          c = RGB(value, 0, 0)
        Case 2 'green
          c = RGB(0, value, 0)
        Case 3 'blue
          c = RGB(0, 0, value)
        Case 4 'edges
          value = 255 - value
          c = RGB(value, value, value)
        Case Else
          c = RGB(value, value, value)
      End Select
      canvas.FillColor = c
      screenX(0) = (x / width) * canvas.ScaleWidth
      screenY(0) = (y / height) * canvas.ScaleHeight
      screenX(1) = ((x + 1) / width) * canvas.ScaleWidth
      screenY(1) = ((y + 1) / height) * canvas.ScaleHeight
```

```
      canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c, B
    Next
  Next

  Else

    'Call showEdges(canvas)
    canvas.Cls
    Call showEdgeTraces(canvas)

  End If

End Sub


Public Sub showEdgeTraces(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer

  'canvas.Cls
  canvas.FillStyle = 0
  For x = 0 To width - 1
    For y = 0 To height - 1
      If (edgeTraced(x, y) = True) Then
        c = RGB(230, 230, 230)
        canvas.FillColor = c
        screenX(0) = (x / width) * canvas.ScaleWidth
        screenY(0) = (y / height) * canvas.ScaleHeight
        screenX(1) = ((x + 1) / width) * canvas.ScaleWidth
        screenY(1) = ((y + 1) / height) * canvas.ScaleHeight
        canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c, B
      End If
    Next
  Next

  Call showEdgeVector(canvas)

End Sub

Public Sub showEdgeVector(canvas As PictureBox)
```

```
  Dim x1 As Integer
  Dim y1 As Integer
  Dim x2 As Integer
  Dim y2 As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
  Dim value As Byte
  Dim c As Long
  Dim i As Integer
  Dim radius As Integer

  'canvas.Cls
  canvas.FillStyle = 0
  canvas.DrawWidth = 1
  radius = (canvas.ScaleWidth / width) / 2
  For i = 0 To currEdgeVector - 1
    x1 = EdgeVector(0, i)
    y1 = EdgeVector(1, i)
    x2 = EdgeVector(2, i)
    y2 = EdgeVector(3, i)

    'c = RGB((EdgeVector(4, i) / maxEdgeVectorIntensity) * 255, 0, 0)
    c = RGB(i, 0, 0)
    canvas.FillColor = c
    screenX(0) = (x1 / width) * canvas.ScaleWidth
    screenY(0) = (y1 / height) * canvas.ScaleHeight
    screenX(1) = (x2 / width) * canvas.ScaleWidth
    screenY(1) = (y2 / height) * canvas.ScaleHeight
    If (i > 0) Then
      canvas.Line -(screenX(0), screenY(0)), c
    End If
    canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c
    'canvas.Circle (screenX(0), screenY(0)), radius, c
    'canvas.Circle (screenX(1), screenY(1)), radius, c
  Next

End Sub



Public Sub showEdges(canvas As PictureBox)
  Dim x As Integer
  Dim y As Integer
  Dim screenX(2) As Single
  Dim screenY(2) As Single
```

```
  Dim edgeType As Byte
  Dim c As Long
  Dim i As Integer

  canvas.Cls
  canvas.FillStyle = 0
  c = RGB(0, 0, 0)
  For x = 0 To edgesWidth - 1
    For y = 0 To edgesHeight - 1

      screenX(0) = (x / edgesWidth) * canvas.ScaleWidth
      screenY(0) = (y / edgesHeight) * canvas.ScaleHeight
      screenX(1) = ((x + 1) / edgesWidth) * canvas.ScaleWidth
      screenY(1) = ((y + 1) / edgesHeight) * canvas.ScaleHeight

      edgeType = Edges(x, y)
      Select Case edgeType
        Case 1 'horizontal line
          canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(0)), c
        Case 2 'vertical line
          canvas.Line (screenX(0), screenY(0))-(screenX(0), screenY(1)), c
        Case 3 'diagonal /
          canvas.Line (screenX(0), screenY(1))-(screenX(1), screenY(0)), c
        Case 4 'diagonal \
          canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(1)), c
        Case 5 'cross
          canvas.Line (screenX(0), screenY(0))-(screenX(1), screenY(0)), c
          canvas.Line (screenX(0), screenY(0))-(screenX(0), screenY(1)), c
      End Select
    Next
  Next

End Sub


Public Sub showEdgeHistogram(chart As Object)
'displays edge histogram using MS chart control

  Dim i As Integer
  Dim estr As String

  chart.chartType = 7
  chart.RowCount = NO_OF_EDGE_TYPES
  chart.ColumnCount = 1
```

```
  estr = ""
  For i = 0 To chart.RowCount - 1
    chart.Row = i + 1
    chart.Data = EdgeHistogram(i)
    estr = estr & EdgeHistogram(i) & ", "
  Next
  chart.Refresh
  'MsgBox estr

End Sub


Public Sub showAngleHistogram(chart As Object)
'displays angle histogram using MS chart control

  Dim i As Integer
  Dim estr As String

  chart.chartType = 7
  chart.RowCount = 18
  chart.ColumnCount = 1

  estr = ""
  For i = 0 To chart.RowCount - 1
    chart.Row = i + 1
    chart.Data = angleHistogram(i)
    estr = estr & angleHistogram(i) & ", "
  Next
  chart.Refresh
  'MsgBox estr

End Sub
```