

# PROGRAM LISTING 1: VISUAL BASIC 3.0 DEMO PROGRAM

By Malcolm Stagg

## I. FORM LAYOUT

VERSION 2.00

Begin Form frmMain

```
BackColor = &H00FFC0C0&
BorderStyle = 1 'Fixed Single
Caption = "Science Fair 03/04 Method Test Program"
ClientHeight = 4560
ClientLeft = 780
ClientTop = 1485
ClientWidth = 5655
ControlBox = 0 'False
ForeColor = &H00000000&
Height = 4965
KeyPreview = -1 'True
Left = 720
LinkTopic = "Form1"
MaxButton = 0 'False
MinButton = 0 'False
ScaleHeight = 304
ScaleMode = 3 'Pixel
ScaleWidth = 377
Top = 1140
Width = 5775
```

Begin PictureBox Picture3

```
AutoRedraw = -1 'True
AutoSize = -1 'True
Height = 375
Left = 5280
ScaleHeight = 23
ScaleMode = 3 'Pixel
ScaleWidth = 23
TabIndex = 11
Top = 120
Visible = 0 'False
Width = 375
```

End

Begin PictureBox Picture2

```
BackColor = &H00800000&
Height = 3630
Left = 120
ScaleHeight = 240
ScaleMode = 3 'Pixel
ScaleWidth = 7
TabIndex = 10
Top = 120
Width = 135
```

End

Begin PictureBox Picture1

```

AutoRedraw    = -1 'True
BackColor     = &H00FF8080&
Height       = 3630
Left         = 420
ScaleHeight  = 240
ScaleMode    = 3 'Pixel
ScaleWidth   = 320
TabIndex     = 3
Top          = 120
Width        = 4830
End
Begin TextBox txtCommand
  BackColor   = &H00800000&
  ForeColor   = &H00FFFFFF&
  Height      = 285
  Left        = 120
  TabIndex    = 0
  Top         = 4200
  Width       = 4935
End
Begin Line Line4
  BorderColor = &H00FFFFFF&
  X1          = 28
  X2          = 351
  Y1          = 250
  Y2          = 250
End
Begin Line Line3
  BorderColor = &H00FFFFFF&
  X1          = 350
  X2          = 350
  Y1          = 8
  Y2          = 250
End
Begin Line Line2
  BorderColor = &H00808080&
  X1          = 27
  X2          = 27
  Y1          = 7
  Y2          = 251
End
Begin Line Line1
  BorderColor = &H00808080&
  X1          = 27
  X2          = 351
  Y1          = 7
  Y2          = 7
End
Begin Label Button
  Alignment   = 2 'Center
  BackColor   = &H00FF8080&
  BorderStyle = 1 'Fixed Single
  Caption     = "About the Program"

```

```
ForeColor      = &H00800000&
Height        = 255
Index         = 6
Left         = 3360
TabIndex     = 8
Top          = 3840
Width        = 1935
End
Begin Label Button
Alignment     = 2 'Center
BackColor    = &H00FF8080&
BorderStyle  = 1 'Fixed Single
Caption      = "How to Use"
ForeColor    = &H00800000&
Height      = 255
Index       = 4
Left       = 2040
TabIndex  = 6
Top       = 3840
Width    = 1215
End
Begin Label Button
Alignment     = 2 'Center
BackColor    = &H00FF8080&
BorderStyle  = 1 'Fixed Single
Caption      = "List of Commands"
ForeColor    = &H00800000&
Height      = 255
Index       = 2
Left       = 360
TabIndex  = 4
Top       = 3840
Width    = 1575
End
Begin Label Button
Alignment     = 2 'Center
BackColor    = &H00FF8080&
BorderStyle  = 1 'Fixed Single
Caption      = "OK"
ForeColor    = &H00800000&
Height      = 255
Index       = 0
Left       = 5160
TabIndex  = 1
Top       = 4200
Width    = 375
End
Begin Label Button
Alignment     = 2 'Center
BackColor    = &H00808080&
ForeColor    = &H00000000&
Height      = 255
Index       = 1
```

```

Left      = 5175
TabIndex = 2
Top       = 4215
Width     = 375
End
Begin Label Button
Alignment = 2 'Center
BackColor = &H00808080&
ForeColor = &H00000000&
Height    = 255
Index     = 3
Left      = 375
TabIndex  = 5
Top       = 3855
Width     = 1575
End
Begin Label Button
Alignment = 2 'Center
BackColor = &H00808080&
ForeColor = &H00000000&
Height    = 255
Index     = 5
Left      = 2055
TabIndex  = 7
Top       = 3855
Width     = 1215
End
Begin Label Button
Alignment = 2 'Center
BackColor = &H00808080&
ForeColor = &H00000000&
Height    = 255
Index     = 7
Left      = 3375
TabIndex  = 9
Top       = 3855
Width     = 1935
End
End
End

```

## II. SUBROUTINES

```

Dim lastdown As Integer, tmp(320, 240) As Long, n, N2, vec(8) As Integer, sx(320,
240) As Integer, sy(320, 240) As Integer, xn, yn, xn1(4), yn1(4), xs(2, 4), ys(2, 4),
thrs(3, 4), pnt(320, 240) As Long, pnt2(320, 240) As Long
Dim feats(1, 1000) As Integer, pts As Integer, dirs, stuck
Dim linex(1, 1000) As Integer, liney(1, 1000) As Integer, lines As Integer
Dim features(319, 239) As Integer, max1 As Integer
Dim ang(320, 240) As Integer, square(-1 To 10, -1 To 10) As Long
Dim con(22) As Integer, corner(1000, 2), corners, cornnow
Dim x, y, min2max(8)

```

```

Sub addcorn (i, j)

```

Checks to see if the corner (i, j), can be added. If it can, it is added to the list of corners.
--

```

'MsgBox i
'MsgBox j
  minimum = 0
  For c = 1 To corners
    x2 = corner(c, 1)
    y2 = corner(c, 2)
    If Abs(x2 - i) <= 2 And Abs(y2 - j) <= 2 Then
      minimum = -1
    End If
  Next
  If Not minimum Then
    corners = corners + 1
    corner(corners, 1) = i
    corner(corners, 2) = j
    picture1.PSet (i, j), &HFF&
  End If
End Sub

```

```

Sub bincorner ()
  This is an earlier corner-finding algorithm I tried to develop.
  For y = 6 To 233
    For x = 6 To 313
      If pnt(x, y) > &H808080 Then
        foundcorn = -1
        For y1 = y - 6 To y - 1
          If pnt(x, y1) > &H808080 Then foundcorn = 0
        Next
        For y1 = y + 1 To y + 6
          If pnt(x, y1) > &H808080 Then foundcorn = 0
        Next
        foundcorn2 = -1
        For x1 = x - 6 To x - 1
          If pnt(x1, y) > &H808080 Then foundcorn2 = 0
        Next
        For x1 = x + 1 To x + 6
          If pnt(x1, y) > &H808080 Then foundcorn2 = 0
        Next
        If foundcorn Then picture1.PSet (x, y), &HFF&
        If foundcorn2 Then picture1.PSet (x, y), &HFF0000
      End If
    Next
    updat y * 100 / 240
    DoEvents
  Next
End Sub

```

```

Sub binedge ()
  This is an earlier thinning algorithm I tried to develop.
  For y = 0 To 238
    For x = 0 To 318
      a = pnt(x, y) > &H808080
      b = pnt(x + 1, y) > &H808080
      c = pnt(x, y + 1) > &H808080
      d = pnt(x + 1, y + 1) > &H808080
      '-
    Next
  Next

```

```

    '|
    '/'
    '\
    If (a Xor b) Or (a Xor c) Or (a Xor d) Then
    'If (a Xor d) Or (b Xor c) Then
        'pnt(x, y) = &HFFFFFF
        picture1.PSet (x, y), &HFFFFFF'pnt(x, y)
    Else
        'pnt(x, y) = 0
        picture1.PSet (x, y), 0'pnt(x, y)
    End If
Next
DoEvents
updat y * 100 / 240
Next

```

End Sub

```

Sub brokeline ()
    Experimental subroutine to fix broken lines.
For y = 0 To 239
    For x = 0 To 319
    blbk:
        If pnt(x, y) > &H808080 And connected8(0, x, y) = 1 Then
            'getkern x, y
            'If (a And connected8(0, x - 1, y - 1) <> 2) Or (b And connected8(0, x, y -
            1) <> 2) Or (c And connected8(0, x + 1, y - 1) <> 2) Or (d And connected8(0, x -
            1, y) <> 2) Or (f And connected8(0, x + 1, y) <> 2) Or (g And connected8(0, x - 1,
            y + 1) <> 2) Or (h And connected8(0, x, y + 1) <> 2) Or (i And connected8(0, x +
            1, y + 1) <> 2) Then pnt(x, y) = 0: GoTo blbk
            dmin = 255
            good = 0
            For y2 = y - 5 To y + 5
                For x2 = x - 5 To x + 5
                    If dmin > (Abs(x - x2) + Abs(y - y2)) Then
                        dmin = (Abs(x - x2) + Abs(y - y2))
                        minx = x2
                        miny = y2
                        good = -1
                    End If
                Next
            Next
            End If
            If good Then
            On Error Resume Next
            If (miny - y) > (minx - x) Then
                If miny < y Then
                    For y2 = miny To y
                        x2 = minx + (y2 - miny) * (x - minx) / (y - miny)
                        pnt(x, y) = &HFFFFFF
                    Next
                Else
                    For y2 = y To miny
                        x2 = minx + (y2 - miny) * (x - minx) / (y - miny)
                    Next
                End If
            End If
        End If
    Next
End Sub

```

```

        pnt(x, y) = &HFFFFFF
    Next
End If
Else
    If minx < x Then
        For x2 = minx To x
            y2 = miny + (x2 - minx) * (y - miny) / (x - minx)
            pnt(x, y) = &HFFFFFF
        Next
    Else
        For x2 = x To minx
            y2 = miny + (x2 - minx) * (y - miny) / (x - minx)
            pnt(x, y) = &HFFFFFF
        Next
    End If
End If
End If
Next
Next
End Sub

Sub btndown (n) General form-interface related subroutine
    Button(n).Left = Button(n).Left + 1
    Button(n).Top = Button(n).Top + 1
End Sub

Sub btnup (n) General form-interface related subroutine
    Button(n).Left = Button(n).Left - 1
    Button(n).Top = Button(n).Top - 1
End Sub

Sub Button_Click (Index As Integer) General form-interface related subroutine
    Select Case Index
        Case 0
            cmd_exec
        Case 2
            frmcmd.Show
        Case 6
            frmwhat.Show
            frmwhat.Label1.Caption = "This program has been written and designed by
Malcolm Stagg. It was written for demonstration purposes for Science Fair 2003/04."
    End Select
End Sub

Sub Button_Db1Click (Index As Integer) General form-interface related subroutine
    Button_MouseUp lastdown, 1, 0, 0, 0
    lastdown = Index
    btndown Index
End Sub



Sub Button_MouseDown (Index As Integer, Button As Integer, Shift As Integer, x As
Single, y As Single)
    Button_MouseUp lastdown, 1, 0, 0, 0

```

```

    lastdown = Index
    btndown Index
End Sub

```

General form-interface related subroutine
---

```

Sub Button_MouseUp (Index As Integer, Button As Integer, Shift As Integer, x As
Single, y As Single)
    If Index = -1 Then Exit Sub
    lastdown = -1
    btnup Index
End Sub

```

```

Function chord (ni, nj, npt)
    dx = feats(0, 0) - ni
    dy = feats(1, 0) - nj
    dmax = 0
    If dx <> 0 And dy <> 0 Then
        slope = dy / dx ' no abs
        b = nj - slope * ni 'no abs
        'MsgBox "ni: " + Str(ni) + ", nj: " + Str(nj) + ", calculated nj:" + Str(slope *
ni) + ", slope: " + Str(slope) + ", difference: " + Str(b)
    End If
    dsum = 0
    If npt = 0 Then
        chord = 0
        Exit Function
    End If
    For i = 1 To npt
        If dx = 0 Then
            d = Abs(feats(0, i) - ni)
        ElseIf dy = 0 Then
            d = Abs(feats(1, i) - nj)
        Else
            xp = (feats(1, i) - b) / slope ' no abs
            xp2 = ((feats(1, i) - nj) * dx) / dy + ni
            yp = slope * feats(0, i) + b ' no abs
            yp2 = ((feats(0, i) - ni) * dy) / dx + nj
            'MsgBox Str(xp) + "=" + Str(xp2) + " " + Str(yp) + "=" + Str(yp2)
            d = min(Abs(feats(0, i) - xp), Abs(feats(1, i) - yp))
            If d >= dmax Then
                dmax = d
                dmaxx = feats(0, i)
                dmaxy = feats(1, i)
                'If dmax > 3 Then MsgBox "dmax:" & Str(dmax) & "x:" & Str(dmaxx) &
"y:" & dmaxy
            End If
            'picture1.Line (feats(0, i), yp)-(xp, feats(1, i)), &HFF&
            'DoEvents
            'picture1.Line (feats(0, i), yp)-(xp, feats(1, i)), &HFFFFFF
            'MsgBox "xp:" & Str(xp)
            'MsgBox "yp:" & Str(yp)
        End If
        dsum = dsum + d
    Next

```

Chords are used for vectorization. A group of pixels is compared to a line
--

```

'picture1.Line (ni, nj)-(feats(0, 0), feats(1, 0)), &HFF&
'DoEvents
If dsum / pts > 15 Then
    'corners = corners + 1
    'corner(corners, 1) = dmaxx'feats(0, i)
    'corner(corners, 2) = dmaxy'feats(1, i)
    MsgBox dmax
    addcorn dmaxx, dmaxy
    chord = 100000
    Exit Function
End If
'picture1.Line (ni, nj)-(feats(0, 0), feats(1, 0)), &HFFFFFF
chord = 0'dsum
End Function

```

```

Sub cmd_exec ( ) General form-interface related subroutine
'on error Resume Next
test = frm1list.List1.ListCount - 1
If UCase(txtcommand.Text) <> "UNDO" Then
    'picture3.Picture = picture1.Picture
End If
100 Select Case UCase(txtcommand.Text)
    Case "8 TO 4 CONNECTED"
        filter84
    Case "ANGLES"
        edgeangles
    Case "BINARY EDGE"
        binedge
    Case "BINARY CORNER"
        bincorner
    Case "BROKEN LINE"
        brokeline
    Case "CLEAR"
        picture1.Cls
    Case "CLOSE"
        Unload frmcmd
        Unload frmwhat
    Case "COMMAND USE"
        frmcmd.Show
        frmwhat.Show
    Case "COPY"
        clipboard.SetData picture1.Image
    Case "DISPLAY"
        display
    Case "EXIT"
        End
    Case "FEATURE"
        feature
    Case "FILTER"
        filter
    Case "FREI-CHEN"
        freichen
    Case "HELP ABOUT"

```

```

    frmwhat.Show
    frmwhat.Label1.Caption = "This program has been written and designed by
Malcolm Stagg. It was written for demonstration purposes for Science Fair 2003/04."
    Case "HIT AND MISS"
        hitmiss
    Case "LEVEL"
        level
    Case "LINE FOLLOW"
        linefollow
    Case "LIST OF COMMANDS"
        frmcmd.Show
    Case "MEAN"
        mean
    Case "MEDIAN"
        median
    Case "MY VECTOR"
        myvector
    Case "NEW VECTOR"
        vectornew
    Case "NEW SUPPRESSION"
        newsuppress
    Case "NON MAXIMAL SUPPRESSION"
        suppression
    Case "NON MAXIMAL SUPPRESSION 2"
        suppression2
    Case "PASTE"
        picture1.Picture = clipboard.GetData()
    Case "POST PROCESS"
        postprocess
    Case "PREWITT GRADIENT"
        prewitt
    Case "ROBERTS CROSS"
        robcross
    Case "ROBERTS CROSS X"
        robcrossx
    Case "ROOT"
        root
    Case "RUN"
        'on error GoTo 1
        test2 = 1
        For p = 0 To test
            txtcommand.Text = frmlist.List1.List(p)
            GoTo 100
5         Next
        test2 = 0
        GoTo 2
1         Resume 2
2         txtcommand.Text = ""
    Case "SHOW LINES"
        showlines
    Case "SOBEL"
        sobel
    Case "SOBEL 2"

```

```

    sobel2
Case "SOBEL X"
    sobelx
Case "SOBEL Y"
    sobely
Case "STORE"
    For y = 0 To 239
        For x = 0 To 319
            pnt(x, y) = picture1.Point(x, y)
        Next
        updat y / 239 * 100
    Next
Case "SQUARED"
    squared
Case "THINNING"
    thinning
Case "THINNING 2"
    thinning2
Case "THINNING 3"
    thinning3
Case "THINNING 4"
    thinning4
Case "UNDO"
    clipboard.SetData picture1.Image
    picture1.Picture = picture3.Picture
    picture3.Picture = clipboard.GetData()
    txtcommand.Text = ""
    If n = 0 Then
        N2 = frm1list.List1.List(frm1list.List1.ListCount - 1)
        frm1list.List1.RemoveItem (frm1list.List1.ListCount - 1)
        n = 1
    Else
        frm1list.List1.AddItem N2
        n = 0
    End If
Case "VECTOR"
    vector
Case "VECTOR 2"
    vector2
Case "VECTOR DOWN"
    vectordn
End Select
'O1, O2, yc, d, s, e
If Left(UCCase(txtcommand.Text), 3) = "3D " Then
    w = 4
    Do Until Mid(txtcommand.Text, w, 1) = ","
        w = w + 1
        o1s = o1s + Mid(txtcommand.Text, w, 1)
    Loop
    w = w + 1
    Do Until Mid(txtcommand.Text, w, 1) = ","
        w = w + 1
        o2s = o2s + Mid(txtcommand.Text, w, 1)

```

```

Loop
w = w + 1
Do Until Mid(txtcommand.Text, w, 1) = ","
    w = w + 1
    ycs = ycs + Mid(txtcommand.Text, w, 1)
Loop
w = w + 1
Do Until Mid(txtcommand.Text, w, 1) = ","
    w = w + 1
    ds = ds + Mid(txtcommand.Text, w, 1)
Loop
w = w + 1
Do Until Mid(txtcommand.Text, w, 1) = ","
    w = w + 1
    ss = ss + Mid(txtcommand.Text, w, 1)
Loop
w = w + 1
Do Until Len(txtcommand.Text) = w
    w = w + 1
    es = es + Mid(txtcommand.Text, w, 1)
Loop
O1 = Val(o1s)
O2 = Val(o2s)
yc = Val(ycs)
d = Val(ds)
s = Val(ss)
e = Val(es)
threed O1, O2, yc, d, s, e
End If
If Left(UCCase(txtcommand.Text), 11) = "LOAD IMAGE " Then
    On Error Resume Next
    picture1.Picture = LoadPicture(Right(txtcommand.Text,
Len(txtcommand.Text) - 11))
End If
If Left(UCCase(txtcommand.Text), 10) = "LOAD LIST " Then
    Open Right(txtcommand.Text, Len(txtcommand.Text) - 10) For Input As #1
    frm1list.List1.Clear
    Do Until EOF(1)
        Input #1, b
        frm1list.List1.AddItem b
    Loop
    Close
    txtcommand.Text = ""
End If
If Left(UCCase(txtcommand.Text), 8) = "SAVE 3D " Then
    Open Right(txtcommand.Text, Len(txtcommand.Text) - 8) For Output As #1
    save3d
    Close
End If
If Left(UCCase(txtcommand.Text), 10) = "SAVE LIST " Then
    Open Right(txtcommand.Text, Len(txtcommand.Text) - 10) For Output As #1
    For n = 0 To frm1list.List1.ListCount
        If frm1list.List1.List(n) <> "" Then Print #1, frm1list.List1.List(n)
    
```

```

    Next
    Close
End If
If Left(UCase(txtcommand.Text), 6) = "SUSAN " Then
    w = 7
    Do Until Mid(txtcommand.Text, w, 1) = ","
        t1s = t1s + Mid(txtcommand.Text, w, 1)
        w = w + 1
    Loop
    w = w + 1
    t2s = Right(txtcommand, Len(txtcommand) - w)
    susan Val(t1s), Val(t2s)
End If
If Left(UCase(txtcommand.Text), 10) = "THRESHOLD " Then
    threshold Val(Right(txtcommand.Text, Len(txtcommand.Text) - 10))
End If
If Left(UCase(txtcommand.Text), 4) = "SET " Then
    If Right(UCase(txtcommand.Text), Len(txtcommand.Text) - 4) = "RIGHT"
Then
        sets 2
    Else
        sets 1
    End If
End If
If Left(UCase(txtcommand.Text), 9) = "MULTIPLY " Then
    multiply (Right(UCase(txtcommand.Text), Len(txtcommand.Text) - 9))
End If

frm1.Show
If txtcommand.Text <> "" Then frm1.List1.AddItem txtcommand.Text
txtcommand.Text = ""
Me.Show
If test2 = 1 Then GoTo 5
End Sub

```

```

Function connected (img, xn, yn) Finds number of pixels in the 4-connected group of a pixel

```

```

n = 0
If xn = 0 Or yn = 0 Or xn = 319 Or yn = 239 Then
    connected = 0
    Exit Function
End If
If img = 0 Then
    'If (pnt(xn - 1, yn - 1) > &H808080) Then n = n + 1
    'If (pnt(xn, yn - 1) > &H808080) Then n = n + 1
    'If (pnt(xn + 1, yn - 1) > &H808080) Then n = n + 1
    'If (pnt(xn - 1, yn) > &H808080) Then n = n + 1
    'If (pnt(xn + 1, yn) > &H808080) Then n = n + 1
    'If (pnt(xn - 1, yn + 1) > &H808080) Then n = n + 1
    'If (pnt(xn, yn + 1) > &H808080) Then n = n + 1
    'If (pnt(xn + 1, yn + 1) > &H808080) Then n = n + 1
Else
    'If (pnt2(xn - 1, yn - 1) > &H808080) Then n = n + 1

```

```

    If (pnt2(xn, yn - 1) > &H808080) Then n = n + 1
    'If (pnt2(xn + 1, yn - 1) > &H808080) Then n = n + 1
    If (pnt2(xn - 1, yn) > &H808080) Then n = n + 1
    If (pnt2(xn + 1, yn) > &H808080) Then n = n + 1
    'If (pnt2(xn - 1, yn + 1) > &H808080) Then n = n + 1
    If (pnt2(xn, yn + 1) > &H808080) Then n = n + 1
    'If (pnt2(xn + 1, yn + 1) > &H808080) Then n = n + 1
End If
    connected = n
End Function

```

```

Function connected8 (img, xn, yn) Finds number of pixels in the 8-connected group of a pixel

```

```

n = 0
If xn = 0 Or yn = 0 Or xn = 319 Or yn = 239 Then
    connected8 = 0
    Exit Function
End If
If img = 0 Then
    If (pnt(xn - 1, yn - 1) > &H808080) Then n = n + 1
    If (pnt(xn, yn - 1) > &H808080) Then n = n + 1
    If (pnt(xn + 1, yn - 1) > &H808080) Then n = n + 1
    If (pnt(xn - 1, yn) > &H808080) Then n = n + 1
    If (pnt(xn + 1, yn) > &H808080) Then n = n + 1
    If (pnt(xn - 1, yn + 1) > &H808080) Then n = n + 1
    If (pnt(xn, yn + 1) > &H808080) Then n = n + 1
    If (pnt(xn + 1, yn + 1) > &H808080) Then n = n + 1
Else
    If (pnt2(xn - 1, yn - 1) > &H808080) Then n = n + 1
    If (pnt2(xn, yn - 1) > &H808080) Then n = n + 1
    If (pnt2(xn + 1, yn - 1) > &H808080) Then n = n + 1
    If (pnt2(xn - 1, yn) > &H808080) Then n = n + 1
    If (pnt2(xn + 1, yn) > &H808080) Then n = n + 1
    If (pnt2(xn - 1, yn + 1) > &H808080) Then n = n + 1
    If (pnt2(xn, yn + 1) > &H808080) Then n = n + 1
    If (pnt2(xn + 1, yn + 1) > &H808080) Then n = n + 1
End If
    connected8 = n
End Function

```

```

Sub corner1 () Part of 2003 vectorization method

```

```

    Do Until result = 1
        xn = xn + 1
        yn = yn - 1
        result = -(picture1.Point(xn, yn) > &H808080)
        If yn = 10 Then
            inccorner1
        End If
        If xn = 225 Then Exit Sub
    Loop
    'MsgBox Str(xn) + Str(yn)
    xn1(1) = xn
    yn1(1) = yn
End Sub

```

```

Sub corner2 () Part of 2003 vectorization method
  Do Until result = 1
    xn = xn - 1
    yn = yn - 1
    result = -(picture1.Point(xn, yn) > &H808080)
    If yn = 10 Then
      inccorner2
    End If
    If xn = 65 Then Exit Sub
  Loop
  'MsgBox Str(xn) + Str(yn)
  xn1(2) = xn
  yn1(2) = yn
End Sub

```

```

Sub corner3 () Part of 2003 vectorization method
  Do Until result = 1
    xn = xn + 1
    yn = yn + 1
    result = -(picture1.Point(xn, yn) > &H808080)
    If yn = 225 Then
      inccorner3
    End If
    If xn = 225 Then Exit Sub
  Loop
  'MsgBox Str(xn) + Str(yn)
  xn1(3) = xn
  yn1(3) = yn
End Sub

```

```

Sub corner4 () Part of 2003 vectorization method
  Do Until result = 1
    xn = xn - 1
    yn = yn + 1
    result = -(picture1.Point(xn, yn) > &H808080)
    If yn = 225 Then
      inccorner4
    End If
    If xn = 65 Then Exit Sub
  Loop
  'MsgBox Str(xn) + Str(yn)
  xn1(4) = xn
  yn1(4) = yn
End Sub

```

```

Sub cornerchk () Part of current vectorization method. Finds if current pixel is near a corner
  nocorner = -1
  For c = 1 To corners
    x2 = corner(c, 1)
    y2 = corner(c, 2)
    dx = Abs(x - x2)
    dy = Abs(y - y2)

```

```

If (dx < 2 And dy < 2) Then
  If linex(0, lines) <> x2 Or liney(0, lines) <> y2 Then
    'MsgBox "line created"
    linex(1, lines) = x2
    liney(1, lines) = y2
    lines = lines + 1
    'x = corner(cornnow, 1)
    'y = corner(cornnow, 2)
    linex(0, lines) = x2
    liney(0, lines) = y2
    nocorner = 0
    stuck = -1

  End If
End If
Next
'On Error Resume Next
'If nocorner Then pnt2(x, y) = 0
End Sub

```

Function crossing (img, xn, yn) Finds how many groups of pixel are connected to the current

```

ReDim cindex(8)
ci = 0
If img = 0 Then
  cindex(1) = pnt(xn, yn - 1) > &H808080
  ' cindex(2) = pnt(xn + 1, yn - 1) > &H808080
  cindex(3) = pnt(xn + 1, yn) > &H808080
  ' cindex(4) = pnt(xn + 1, yn + 1) > &H808080
  cindex(5) = pnt(xn, yn + 1) > &H808080
  ' cindex(6) = pnt(xn - 1, yn + 1) > &H808080
  cindex(7) = pnt(xn - 1, yn) > &H808080
  ' cindex(8) = pnt(xn - 1, yn - 1) > &H808080
Else
  cindex(1) = pnt2(xn, yn - 1) > &H808080
  'cindex(2) = pnt2(xn + 1, yn - 1) > &H808080
  cindex(3) = pnt2(xn + 1, yn) > &H808080
  'cindex(4) = pnt2(xn + 1, yn + 1) > &H808080
  cindex(5) = pnt2(xn, yn + 1) > &H808080
  'cindex(6) = pnt2(xn - 1, yn + 1) > &H808080
  cindex(7) = pnt2(xn - 1, yn) > &H808080
  'cindex(8) = pnt2(xn - 1, yn - 1) > &H808080
End If
If cindex(1) And cindex(3) Then cindex(2) = -1
If cindex(3) And cindex(5) Then cindex(4) = -1
If cindex(5) And cindex(7) Then cindex(6) = -1
If cindex(7) And cindex(1) Then cindex(8) = -1
cindex(0) = cindex(8)
For n1 = 0 To 7
  If cindex(n1) < cindex(n1 + 1) Then ci = ci + 1
Next
crossing = ci

```

End Function

Some subs don't display the result to save time. This will show what it looks like.

```
Sub display ()
For y = 0 To 239
  For x = 0 To 319
    picture1.PSet (x, y), pnt(x, y)
  Next
  updat y / 239 * 100
  DoEvents
Next
End Sub
```

This is a demo sub to show basically which direction the edges are going in

```
Sub edgeangles ()
For y = 1 To 238
  For x = 1 To 318
    If pnt(x, y) > &H808080 Then
      If ang(x, y) >= -.4 And ang(x, y) < .4 Then picture1.PSet (x, y), &HFF&
      If ang(x, y) >= .4 And ang(x, y) < 2.36 Then picture1.PSet (x, y), &HFF00&
      If ang(x, y) >= 2.36 Or ang(x, y) < -2.36 Then picture1.PSet (x, y),
&HFF0000
      If ang(x, y) >= -2.36 And ang(x, y) < -.4 Then picture1.PSet (x, y),
&HFFFF00
    End If
  Next
  updat y / 239 * 100
  DoEvents
Next
End Sub
```

This was a poor attempt at a line following algorithm

```
Sub feature ()
For y = 0 To 239
  For x = 0 To 319
    features(x, y) = 0
  Next
Next
For y = 1 To 235
  For x = 1 To 315
    'MsgBox X
    'MsgBox Y
    If max16(tmp(x, y), tmp(x + 1, y), tmp(x + 2, y), tmp(x + 3, y), tmp(x, y + 1), tmp(x + 1, y + 1), tmp(x + 2, y + 1), tmp(x + 3, y + 1), tmp(x, y + 2), tmp(x + 1, y + 2), tmp(x + 2, y + 2), tmp(x + 3, y + 2), tmp(x, y + 3), tmp(x + 1, y + 3), tmp(x + 2, y + 3), tmp(x + 3, y + 3)) > 25 Then
      Select Case max1
      Case 1
        features(x, y) = 1
      Case 2
        features(x + 1, y) = 1
      Case 3
        features(x + 2, y) = 1
      Case 4
        features(x + 3, y) = 1
      Case 5
```

```

        features(x, y + 1) = 1
    Case 6
        features(x + 1, y + 1) = 1
    Case 7
        features(x + 2, y + 1) = 1
    Case 8
        features(x + 3, y + 1) = 1
    Case 9
        features(x, y + 2) = 1
    Case 10
        features(x + 1, y + 2) = 1
    Case 11
        features(x + 2, y + 2) = 1
    Case 12
        features(x + 3, y + 2) = 1
    Case 13
        features(x, y + 3) = 1
    Case 14
        features(x + 1, y + 3) = 1
    Case 15
        features(x + 2, y + 3) = 1
    Case 16
        features(x + 3, y + 3) = 1
    End Select
End If
Next
Next
Next
For y = 2 To 237
    For x = 2 To 317
        If features(x, y) = 1 Then
            lin1 = pnt(x - 1, y - 1) + pnt(x - 2, y - 2) + pnt(x + 1, y + 1) + pnt(x +
2, y + 2)
            lin2 = pnt(x, y - 1) + pnt(x, y - 2) + pnt(x, y + 1) + pnt(x, y + 2)
            lin3 = pnt(x + 1, y - 1) + pnt(x + 2, y - 2) + pnt(x - 1, y + 1) + pnt(x -
2, y + 2)
            lin4 = pnt(x - 1, y) + pnt(x - 2, y) + pnt(x + 1, y) + pnt(x + 2, y)
            maxline = 0
            minline = 1000000000
            If lin1 > maxline Then maxline = lin1
            If lin1 < minline Then minline = lin1
            If lin2 > maxline Then maxline = lin2
            If lin2 < minline Then minline = lin2
            If lin3 > maxline Then maxline = lin3
            If lin3 < minline Then minline = lin3
            If lin4 > maxline Then maxline = lin4
            If lin4 < minline Then minline = lin4
            On Error Resume Next
            If minline / maxline >= 1 / 3 Then features(x, y) = 0
        End If
        picture1.PSet (x, y), features(x, y) * &HFFFFFF
    Next
    DoEvents
Next
Next

```

End Sub

Sub filter () These are the great noise filters and 8- to 4- connected filters that I'm using

```
' 000  
' 010  
' 000  
'
```

```
'or  
'00000  
'0 0  
'0 1 0  
'0 0  
'00000
```

```
For yb = 2 To 237
```

```
  For xa = 2 To 317
```

```
    y = yb
```

```
    x = xa
```

```
    'If pnt(x, y) > &H808080 Then
```

```
      th = &H808080
```

```
      h = pnt(x, y - 1) > th
```

```
      l = pnt(x - 1, y) > th
```

```
      m = pnt(x, y) > th
```

```
    If m Or (l And h) Then
```

```
      a = pnt(x - 2, y - 2) > th
```

```
      b = pnt(x - 1, y - 2) > th
```

```
      c = pnt(x, y - 2) > th
```

```
      d = pnt(x + 1, y - 1) > th
```

```
      e = pnt(x + 2, y - 1) > th
```

```
      f = pnt(x - 2, y - 1) > th
```

```
      g = pnt(x - 1, y - 1) > th
```

```
      i = pnt(x + 1, y - 1) > th
```

```
      j = pnt(x + 2, y - 1) > th
```

```
      k = pnt(x - 2, y) > th
```

```
      l = pnt(x - 1, y) > th
```

```
      n = pnt(x + 1, y) > th
```

```
      o = pnt(x + 2, y) > th
```

```
      p = pnt(x - 2, y + 1) > th
```

```
      q = pnt(x - 1, y + 1) > th
```

```
      r = pnt(x, y + 1) > th
```

```
      s = pnt(x + 1, y + 1) > th
```

```
      t = pnt(x + 2, y + 1) > th
```

```
      u = pnt(x - 2, y + 2) > th
```

```
      v = pnt(x - 1, y + 2) > th
```

```
      w = pnt(x, y + 2) > th
```

```
      xc = pnt(x + 1, y + 2) > th
```

```
      yc = pnt(x + 2, y + 2) > th
```

```
    'ABCDE
```

```
    'FGHIJ |GH|
```

```
    'KLMNO |LM|
```

```
'PQRST --
'UVWXY
```

```
If (m And Not a And Not b And Not c And Not d And Not e And Not f And
Not j And Not k And Not o And Not p And Not t And Not u And Not v And Not w And
Not xc And Not yc) Or (Not g And Not h And Not i And Not l And Not n And Not q And
Not r And Not s And m) Then
```

```
    pnt(x - 1, y - 1) = 0
    pnt(x, y - 1) = 0
    pnt(x + 1, y - 1) = 0
    pnt(x - 1, y) = 0
    pnt(x, y) = 0
    pnt(x + 1, y) = 0
    pnt(x - 1, y + 1) = 0
    pnt(x, y + 1) = 0
    pnt(x + 1, y + 1) = 0
```

```
End If
```

```
If g And m Then
```

```
    If Not (c And d And i) And Not (i And n) And Not l And Not (b And c)
```

```
Then
```

```
    pnt(x, y - 1) = &HFFFFFFF
```

```
    ElseIf Not h And Not (r And q) And Not (q And p And k) And Not (k And
```

```
f) Then
```

```
        pnt(x - 1, y) = &HFFFFFFF
```

```
    End If
```

```
    ElseIf l And h Then
```

```
        If Not (b And c) And Not m And Not (k And f) And Not (f And a And b)
```

```
Then
```

```
        pnt(x - 1, y - 1) = &HFFFFFFF
```

```
        ElseIf Not (i And n) And Not (n And s And r) And Not (r And q) And Not
```

```
g Then
```

```
            pnt(x, y) = &HFFFFFFF
```

```
        End If
```

```
    End If
```

```
    'Else
```

```
    'End If
```

```
    End If
```

```
Next xa
```

```
    updat yb / 237 * 100
```

```
Next yb
```

```
For y = 2 To 237
```

```
    For x = 2 To 317
```

```
        'If pnt(x, y) < &H808080 And crossing(0, x, y) = 2 Then
```

```
            'pnt(x, y) = &HFFFFFFF
```

```
        'End If
```

```
        picture1.PSet (x, y), pnt(x, y)
```

```
    Next
```

```
    updat y / 237 * 100
```

```
    DoEvents
```

```
Next
```

```
End Sub
```

```
Sub filter84 ()
```

An earlier attempt at an 8- to 4- connected filter
--

```
t = &H808080
```

```
For y = 0 To 240  
  For x = 0 To 320  
    pnt2(x, y) = pnt(x, y)  
  Next  
Next
```

```
For y = 0 To 237  
  For x = 0 To 317  
    a = pnt(x, y) > t  
    b = pnt(x + 1, y) > t  
    c = pnt(x + 2, y) > t  
    d = pnt(x + 3, y) > t  
    e = pnt(x, y + 1) > t  
    f = pnt(x + 1, y + 1) > t  
    g = pnt(x + 2, y + 1) > t  
    h = pnt(x + 3, y + 1) > t  
    i = pnt(x, y + 2) > t  
    j = pnt(x + 1, y + 2) > t  
    k = pnt(x + 2, y + 2) > t  
    l = pnt(x + 3, y + 2) > t  
    m = pnt(x, y + 3) > t  
    n = pnt(x + 1, y + 3) > t  
    o = pnt(x + 2, y + 3) > t  
    p = pnt(x + 3, y + 3) > t
```

```
    If e And Not (f And g) And h Then ' see 4a  
    'If e And h Then  
    '  b = 0  
    '  c = 0  
    '  f = -1  
    '  g = -1  
    '  j = 0  
    '  k = 0  
    ' ElseIf b And Not (f And j) And n Then '4b  
    'ElseIf b And n Then  
    '  e = 0  
    '  f = -1  
    '  g = 0  
    '  i = 0  
    '  j = -1  
    '  k = 0  
    ' ElseIf a And Not (f And k) And p Then '4c  
    'ElseIf a And p Then  
    '  b = -1  
    '  c = 0  
    '  d = 0  
    '  e = 0  
    '  f = -1  
    '  g = -1  
    '  h = 0
```

```

'      i = 0
'      j = 0
'      k = -1
'      l = -1
'      n = 0
'      o = 0
'      ElseIf d And Not (g And j) And m Then '4d
'      'ElseIf d And m Then
'      a = 0
'      b = 0
'      c = -1
'      e = 0
'      f = -1
'      g = -1
'      h = 0
'      i = -1
'      j = -1
'      k = 0
'      l = 0
'      n = 0
'      o = 0
'      ElseIf a And Not (f And j) And o Then '43a
'      'ElseIf a And o Then
'      b = -1
'      c = 0
'      e = 0
'      f = -1
'      g = 0
'      i = 0
'      j = -1
'      k = 0
'      m = 0
'      n = -1
'      ElseIf c And Not (f And j) And m Then '43b
'      'ElseIf c And m Then
'      a = 0
'      b = -1
'      e = 0
'      f = -1
'      g = 0
'      i = 0
'      j = -1
'      k = 0
'      n = -1
'      o = 0
'      ElseIf a And Not (f And g) And l Then '43c
'      'ElseIf a And l Then
'      b = 0
'      c = 0
'      d = 0
'      e = -1
'      f = -1
'      g = -1

```

```

'      h = -1
'      i = 0
'      j = 0
'      k = 0
'      ElseIf d And Not (f And g) And i Then '43d
'      'ElseIf d And i Then
'      a = 0
'      b = 0
'      c = 0
'      e = -1
'      f = -1
'      g = -1
'      h = -1
'      j = 0
'      k = 0
'      l = 0
'      ElseIf b And Not (f And k) And o Then '42a
'      'ElseIf b And o Then
'      c = 0
'      d = 0
'      e = 0
'      f = -1
'      g = -1
'      h = 0
'      i = 0
'      j = 0
'      k = -1
'      l = 0
'      ElseIf c And Not (g And j) And n Then '42b
'      'ElseIf c And n Then
'      a = 0
'      b = 0
'      e = 0
'      f = -1
'      g = -1
'      h = 0
'      i = 0
'      j = -1
'      k = 0
'      l = 0
'      ElseIf e And Not (f And k) And l Then '42c
'      'ElseIf e And l Then
'      b = 0
'      c = 0
'      d = 0
'      f = -1
'      g = -1
'      h = 0
'      j = 0
'      k = -1
'      n = 0
'      o = 0
'      ElseIf h And Not (g And j) And i Then '42d

```

```

' ElseIf h And i Then
'     a = 0
'     b = 0
'     c = 0
'     e = 0
'     f = -1
'     g = -1
'     j = -1
'     k = 0
'     n = 0
'     o = 0
' ElseIf e And Not f And g Then '3a
" ElseIf e And g Then
'     b = 0
'     f = -1
'     j = 0
'     pnt2(x + 1, y) = b * -&HFFFFFF
'     pnt2(x + 1, y + 1) = f * -&HFFFFFF
'     pnt2(x + 1, y + 2) = j * -&HFFFFFF
' ElseIf b And Not f And j Then '3b
" ElseIf b And j Then
'     e = 0
'     f = -1
'     g = 0
'     pnt2(x, y + 1) = e * -&HFFFFFF
'     pnt2(x + 1, y + 1) = f * -&HFFFFFF
'     pnt2(x + 2, y + 1) = g * -&HFFFFFF
' ElseIf a And Not f And k Then '3c
" ElseIf a And k Then
'     b = 0
'     c = 0
'     e = -1
'     f = -1
'     g = -1
'     i = 0
'     j = 0
'     pnt2(x + 1, y) = b * -&HFFFFFF
'     pnt2(x + 2, y) = c * -&HFFFFFF
'     pnt2(x, y + 1) = e * -&HFFFFFF
'     pnt2(x + 1, y + 1) = f * -&HFFFFFF
'     pnt2(x + 2, y + 1) = g * -&HFFFFFF
'     pnt2(x, y + 2) = i * -&HFFFFFF
'     pnt2(x + 1, y + 2) = j * -&HFFFFFF
' ElseIf c And Not f And i Then '3d
" ElseIf c And i Then
'     a = 0
'     b = 0
'     e = -1
'     f = -1
'     g = -1
'     j = 0
'     k = 0
'     pnt2(x, y) = a * -&HFFFFFF

```

```

'   pnt2(x + 1, y) = b * -&HFFFFFF
'   pnt2(x, y + 1) = e * -&HFFFFFF
'   pnt2(x + 1, y + 1) = f * -&HFFFFFF
'   pnt2(x + 2, y + 1) = g * -&HFFFFFF
'   pnt2(x + 1, y + 2) = j * -&HFFFFFF
'   pnt2(x + 2, y + 2) = k * -&HFFFFFF
'   ElseIf b And Not (f And g) And k Then '32a
'   'ElseIf b And k Then
'       c = 0
'       d = 0
'       e = 0
'       f = -1
'       g = -1
'       h = 0
'       i = 0
'       j = 0
'   ElseIf c And Not (f And g) And j Then '32b
'   'ElseIf c And j Then
'       a = 0
'       b = 0
'       e = 0
'       f = -1
'       g = -1
'       h = 0
'       k = 0
'       l = 0
'   ElseIf g And Not (f And j) And i Then '32c
'   'ElseIf g And i Then
'       a = 0
'       b = 0
'       e = 0
'       f = -1
'       g = -1
'       h = 0
'       k = 0
'       l = 0
'   ElseIf e And Not (f And j) And l Then '32d
'   'ElseIf e And l Then
'       b = 0
'       c = 0
'       f = -1
'       g = 0
'       i = 0
'       j = -1
'       m = 0
'       n = 0
'   ElseIf a And Not b And Not e And f Then '2a
'       b = -1
'       pnt2(x + 1, y) = b * -&HFFFFFF
'   ElseIf b And Not a And Not f And e Then '2b
'       a = -1
'       pnt2(x, y) = a * -&HFFFFFF
End If

```

```

Next
DoEvents
updat y / 2.4
Next
For y = 0 To 240
  For x = 0 To 320
    pnt(x, y) = pnt2(x, y)
  Next
Next

```

```
End Sub
```

```
Sub findstcorner ()
```

Part of vectorization - finds a vertice to start a line with
--

```

For n = 1 To corners
  x = corner(n, 1)
  y = corner(n, 2)
  On Error Resume Next
  ' For y2 = y - 3 To y + 3
  '   For x2 = x - 3 To x + 3
  '     If pnt2(x2, y2) > &H808080 Then GoTo goodfind
  '   Next
  ' Next
  ' 'uh oh
  ' GoTo nogoodfind
'goodfind:
'   x = x2
'   y = y2
'   MsgBox Str(x) & Str(y) & Str(pnt2(x, y))
  stuck = -1
  t = &H808080
  If pnt2(x, y - 1) > t Then
    stuck = 0
    dirs = 1
  End If
  If pnt2(x + 1, y) > t Then
    stuck = 0
    dirs = 2
  End If
  If pnt2(x, y + 1) > t Then
    stuck = 0
    dirs = 3
  End If
  If pnt2(x - 1, y) > t Then
    stuck = 0
    dirs = 4
  End If
  linex(0, lines) = x
  liney(0, lines) = y
  If Not stuck Then
    ' For i = 0 To 320
    '   For j = 0 To 240
    '     pnt2(i, j) = pnt(i, j)
    '   Next

```

```

    ' Next
    'cornnow = n
    'MsgBox Str(x) & Str(y)
    Exit Sub
  End If
Next
' no good starting point
End Sub

```

General form-interface related subroutine

```

Sub Form_KeyPress (KeyAscii As Integer)
  If KeyAscii = 13 Then
    Button_Click (0)
  End If
End Sub

```

```

Sub Form_Load ()
  lastdown = -1
End Sub

```

General form-interface related subroutine

```

Sub Form_Resize ()
  If frmmain.WindowState <> 0 Then frmmain.WindowState = 0
End Sub

```

Frei-Chen edge detection method

```

Sub freichen ()
  For y = 1 To 238
    For x = 1 To 318
      n = retfreichenx(x, y)
      N2 = retfreicheny(x, y)
      n3 = Sqr(n * n + N2 * N2)
      If n3 > 255 Then n3 = 255
      tmp(x, y) = n3
    Next
    updat y / 239 * 100
  Next
  For y = 0 To 239
    For x = 0 To 319
      picture1.PSet (x, y), tmp(x, y) * &H10101
    Next
    updat y / 239 * 100
  Next
End Sub

```

This will store the neighborhood around a pixel to 9 variables

```

Sub getkern (i, j)
  a = pnt(i - 1, j - 1) > &H808080
  b = pnt(i, j - 1) > &H808080
  c = pnt(i + 1, j - 1) > &H808080
  d = pnt(i - 1, j) > &H808080
  e = pnt(i, j) > &H808080
  f = pnt(i + 1, j) > &H808080
  g = pnt(i - 1, j + 1) > &H808080
  h = pnt(i, j + 1) > &H808080
  i = pnt(i + 1, j + 1) > &H808080

```

End Sub

A terrible corner finding algorithm

Sub hitmiss ()

For y = 0 To 239

For x = 0 To 319

a = -(picture1.Point(x - 1, y - 1) > &H808080)

b = -(picture1.Point(x, y - 1) > &H808080)

c = -(picture1.Point(x + 1, y - 1) > &H808080)

d = -(picture1.Point(x - 1, y) > &H808080)

e = -(picture1.Point(x, y) > &H808080)

f = -(picture1.Point(x + 1, y) > &H808080)

g = -(picture1.Point(x - 1, y + 1) > &H808080)

h = -(picture1.Point(x, y + 1) > &H808080)

i = -(picture1.Point(x + 1, y + 1) > &H808080)

If b = 1 And d = 0 And e = 1 And f = 1 And g = 0 And h = 0 Then

tmp(x, y) = 1

ElseIf b = 1 And d = 1 And e = 1 And f = 0 And h = 0 And i = 0 Then

tmp(x, y) = 1

ElseIf b = 0 And c = 0 And d = 1 And e = 1 And f = 0 And h = 1 Then

tmp(x, y) = 1

ElseIf a = 0 And b = 0 And d = 0 And e = 1 And f = 1 And h = 1 Then

tmp(x, y) = 1

Else

tmp(x, y) = 0

End If

If a = 0 And b = 0 And c = 0 And d = 0 And e = 1 And g = 0 Then

tmp(x, y) = 1

ElseIf a = 0 And b = 0 And c = 0 And e = 1 And f = 0 And i = 0 Then

tmp(x, y) = 1

ElseIf c = 0 And e = 1 And f = 0 And g = 0 And h = 0 And i = 0 Then

tmp(x, y) = 1

ElseIf a = 0 And d = 0 And e = 1 And g = 0 And h = 0 And i = 0 Then

tmp(x, y) = 1

ElseIf a = 0 And b = 0 And c = 0 And d = 0 And e = 1 And f = 0 Then

tmp(x, y) = 1

ElseIf a = 0 And b = 0 And d = 0 And e = 1 And g = 0 And h = 0 Then

tmp(x, y) = 1

ElseIf b = 0 And c = 0 And e = 1 And f = 0 And h = 0 And i = 0 Then

tmp(x, y) = 1

ElseIf d = 0 And e = 1 And f = 0 And g = 0 And h = 0 And i = 0 Then

tmp(x, y) = 1

Else

tmp(x, y) = 0

End If

'If a = 1 And b = 1 And c = 1 And d = 1 And e = 0 And g = 1 Then

' tmp(x, y) = 1

'ElseIf a = 1 And b = 1 And c = 1 And e = 0 And f = 1 And i = 1 Then

' tmp(x, y) = 1

'ElseIf c = 1 And e = 0 And f = 1 And g = 1 And h = 1 And i = 1 Then

' tmp(x, y) = 1

'ElseIf a = 1 And d = 1 And e = 0 And g = 1 And h = 1 And i = 1 Then

' tmp(x, y) = 1

'Else

```

        ' tmp(x, y) = 0
    'End If
    Next
    updat y / 239 * 100
Next
For y = 0 To 239
    For x = 0 To 319
        picture1.PSet (x, y), tmp(x, y) * &HFFFFFF
    Next
    updat y / 239 * 100
Next
End Sub

Sub inccorner1 () 
    yn = xn + 1
    xn = 15
End Sub

Sub inccorner2 () 
    yn = (320 - xn) + 1
    xn = 305
End Sub

Sub inccorner3 () 
    yn = (240 - xn) - 1
    xn = 15
End Sub

Sub inccorner4 () 
    yn = (xn - 80) - 1
    xn = 305
End Sub

Sub level () 
For y = 2 To 237
    For x = 2 To 317
        n = retlevelx(x, y)
        N2 = retlevely(x, y)
        n3 = Sqr(n * n + N2 * N2)
        If n3 > 255 Then n3 = 255
        tmp(x, y) = n3
        picture1.PSet (x, y), tmp(x, y) * &H10101
    Next
    updat y / 239 * 100
    DoEvents
Next
End Sub

Sub linefollow () 

End Sub


Function max16 (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p)

```

```

max1 = 1
If b > a Then a = b: max1 = 2
If c > a Then a = c: max1 = 3
If d > a Then a = d: max1 = 4
If e > a Then a = e: max1 = 5
If f > a Then a = f: max1 = 6
If g > a Then a = g: max1 = 7
If h > a Then a = h: max1 = 8
If i > a Then a = i: max1 = 9
If j > a Then a = j: max1 = 10
If k > a Then a = k: max1 = 11
If l > a Then a = l: max1 = 12
If m > a Then a = m: max1 = 13
If n > a Then a = n: max1 = 14
If o > a Then a = o: max1 = 15
If p > a Then a = p: max1 = 16
max16 = a
End Function

```

```

Sub mean () 
'abc
'def
'ghi

```

```

'1/10 1/10 1/10 = .3
'1/10 1/5 1/10 = .4
'1/10 1/10 1/10 = .3
For y = 1 To 238
  For x = 1 To 318
    a = pnt(x - 1, y - 1) And &HFF&
    b = pnt(x, y - 1) And &HFF&
    c = pnt(x + 1, y - 1) And &HFF&
    d = pnt(x - 1, y) And &HFF&
    e = pnt(x, y) And &HFF&
    f = pnt(x + 1, y) And &HFF&
    g = pnt(x - 1, y + 1) And &HFF&
    h = pnt(x, y + 1) And &HFF&
    i = pnt(x + 1, y + 1) And &HFF&
    'tmp(x, y) = Int((a + b + c + d + f + g + h + i) / 10 + e / 5) * &H10101
    tmp(x, y) = Int((a + b + c + d + e + f + g + h + i) / 9) * &H10101
  Next
  updat y / 239 * 100
  'DoEvents
Next
For y = 1 To 238
  For x = 1 To 318
    pnt(x, y) = tmp(x, y)
  Next
Next
End Sub

```

```

Sub median () 

```

```

For y = 1 To 238
  For x = 1 To 318
    a = pnt(x - 1, y - 1) And &HFF&
    b = pnt(x, y - 1) And &HFF&
    c = pnt(x + 1, y - 1) And &HFF&
    d = pnt(x - 1, y) And &HFF&
    e = pnt(x, y) And &HFF&
    f = pnt(x + 1, y) And &HFF&
    g = pnt(x - 1, y + 1) And &HFF&
    h = pnt(x, y + 1) And &HFF&
    i = pnt(x + 1, y + 1) And &HFF&
    'MsgBox Str(a) & Str(b) & Str(c) & Str(d) & Str(e) & Str(f) & Str(g) & Str(h)
    & Str(i)

    For n1 = 1 To 5
      mins = 255
      If a <= mins Then minval = 1: mins = a
      If b <= mins Then minval = 2: mins = b
      If c <= mins Then minval = 3: mins = c

      If d <= mins Then minval = 4: mins = d
      If e <= mins Then minval = 5: mins = e
      If f <= mins Then minval = 6: mins = f

      If g <= mins Then minval = 7: mins = g
      If h <= mins Then minval = 8: mins = h
      If i <= mins Then minval = 9: mins = i

      If minval = 1 Then a = 255
      If minval = 2 Then b = 255
      If minval = 3 Then c = 255
      If minval = 4 Then d = 255
      If minval = 5 Then e = 255
      If minval = 6 Then f = 255
      If minval = 7 Then g = 255
      If minval = 8 Then h = 255
      If minval = 9 Then i = 255
    Next

    'MsgBox Str(min2max(1)) + Str(min2max(2)) + Str(min2max(3)) +
    Str(min2max(4)) + Str(min2max(5)) + Str(min2max(6)) + Str(min2max(7)) +
    Str(min2max(8))
    tmp(x, y) = mins * &H10101

    'tmp(x, y) = Int((a + b + c + d + f + g + h + i) / 10 + e / 5) * &H10101
    'tmp(x, y) = Int((a + b + c + d + e + f + g + h + i) / 9) * &H10101
    '
    'MsgBox Hex(tmp(x, y))
  Next
  updat y / 239 * 100
  'DoEvents
Next
For y = 1 To 238

```

```

For x = 1 To 318
    pnt(x, y) = tmp(x, y)
Next
Next
End Sub

```

```

Function min (a As Double, b As Double)

```

This finds the minimum of two values

```

    If a < b Then
        min = a
    Else
        min = b
    End If
End Function

```

```

Sub multiply (times)

```

Multiplies each point in the image by a value

```

For y = 0 To 239
    For x = 0 To 319
        pnt(x, y) = &H10101 * Int((pnt(x, y) And &HFF&) * times)
    Next
    updat y / 239 * 100
Next
End Sub

```

```

Sub myvector ()

```

Another of my attempts at a vectorization algorithm

```

ReDim mykern(20) As Integer
picture1.Cls
For y = 0 To 229

```

```

    For x = 0 To 309
        tmp(x, y) = 1
    Next
    DoEvents
Next
For y = 0 To 229

```

```

    For x = 0 To 309
        tmp(x, y) = 1
    Next
    DoEvents
Next
For y = 0 To 229

```

```

    For x = 0 To 309
        tmp(x, y) = 1
    Next
    DoEvents
Next
For y = 0 To 229

```

```

    For x = 0 To 309

```

```

        For y = 0 To 229

```

```

            For x = 0 To 309

```

```

                'try kernel of 6 x 6 pixels, not good for complex objects, but fast test

```

```

                If pnt(x, y) > &H808080 Then

```

```

                    MsgBox "x" & Str(x)

```

```

                    MsgBox "y" & Str(y)

```

```

                    mykern(1) = pnt(x, y) > &H808080

```

```

                    mykern(2) = pnt(x + 1, y) > &H808080

```

```

                    mykern(3) = pnt(x + 2, y) > &H808080

```

```

                    mykern(4) = pnt(x + 3, y) > &H808080

```

```

                    mykern(5) = pnt(x + 4, y) > &H808080

```

```

                    mykern(6) = pnt(x + 5, y) > &H808080

```

```

                    mykern(7) = pnt(x + 5, y + 1) > &H808080

```

```

                    mykern(8) = pnt(x + 5, y + 2) > &H808080

```

```

                    mykern(9) = pnt(x + 5, y + 3) > &H808080

```

```

                    mykern(10) = pnt(x + 5, y + 4) > &H808080

```

```

                    mykern(11) = pnt(x + 5, y + 5) > &H808080

```

```

                    mykern(12) = pnt(x + 4, y + 5) > &H808080

```

```

                    mykern(13) = pnt(x + 3, y + 5) > &H808080

```

```

                    mykern(14) = pnt(x + 2, y + 5) > &H808080

```

```

                    mykern(15) = pnt(x + 1, y + 5) > &H808080

```

```

mykern(16) = pnt(x, y + 5) > &H808080
mykern(17) = pnt(x, y + 4) > &H808080
mykern(18) = pnt(x, y + 3) > &H808080
mykern(19) = pnt(x, y + 2) > &H808080
mykern(20) = pnt(x, y + 1) > &H808080
xy = 1
group = 100
break = 0

```

mybk:

```

If mykern(xy) And mykern(xy + 1) Then
  group = group + 1
  mykern(xy) = group
  xy = xy + 1
  mykern(xy) = group
  xy = xy + 1
  If xy < 19 Then
    Do Until ((Not mykern(xy)) And (Not mykern(xy + 1))) Or xy = 19
      mykern(xy) = group
      xy = xy + 1
    Loop
  End If
  If xy = 19 Then
    If mykern(xy + 1) Then
      mykern(xy) = group
      xy = xy + 1
      mykern(xy) = group
      GoTo mynxt
    End If
    If mykern(xy) And mykern(1) > 100 Then
      mykern(xy) = group
      xy = xy + 1
      mykern(xy) = group
      GoTo mynxt
    Else
      If mykern(xy) Then
        mykern(xy) = group
      Else
        mykern(xy) = break
      End If
      xy = xy + 1
      mykern(xy) = break
      GoTo mynxt
    End If
  ElseIf xy > 19 Then
    GoTo mynxt
  End If
End If
If (Not mykern(xy)) And (Not mykern(xy + 1)) Then
  break = break + 1
  mykern(xy) = break
  xy = xy + 1
  mykern(xy) = break
  xy = xy + 1

```

```

If xy < 19 Then
  Do Until (mykern(xy) And mykern(xy + 1)) Or xy = 19
    mykern(xy) = break
    xy = xy + 1
  Loop
End If
If xy = 19 Then
  If Not (mykern(xy + 1)) Then
    mykern(xy) = break
    xy = xy + 1
    mykern(xy) = break
    GoTo mynxt
  End If
  If Not (mykern(xy)) And mykern(1) > 0 And mykern(1) < 100 Then
    mykern(xy) = break
    xy = xy + 1
    mykern(xy) = break
    GoTo mynxt
  Else
    If Not mykern(xy) Then
      mykern(xy) = break
    Else
      mykern(xy) = group
    End If
    xy = xy + 1
    mykern(xy) = group
    GoTo mynxt
  End If
  ElseIf xy > 19 Then
    GoTo mynxt
  End If
  GoTo mybk
End If
If mykern(xy) Or mykern(xy + 1) Then
  'mykern(xy) = -1
  xy = xy + 1
  GoTo mybk
End If
mynxt:
If mykern(1) < 1 Then
  If mykern(2) = mykern(3) Then
    mykern(1) = mykern(2)
  ElseIf (mykern(20) = mykern(19)) Or (mykern(20) > 100) = mykern(1)
Then
    mykern(1) = mykern(20)
  Else
    mykern(1) = 1'just set it to something
  End If
  For n = 1 To 20
    If mykern(n) < 1 Then
      mykern(n) = mykern(1)
    End If
  Next

```

```

End If
'now group together first & last, if nessessary
If (mykern(1) > 100) = (mykern(20) > 100) Then
    mykern(20) = mykern(1)
    If mykern(20) > 100 Then
        group = group - 1
        gb = group
    Else
        break = break - 1
        gb = break
    End If
    For n = 1 To 20
        If mykern(n) = gb + 1 Then
            mykern(n) = mykern(1)
        End If
    Next
End If
If group > 102 Then
    'MsgBox "g" & Str(group - 100)
    'MsgBox "b" & Str(break)
    'MsgBox "x" & Str(x)
    'MsgBox "y" & Str(y)
    If tmp(x, y) = 1 Then
        picture1.PSet (x, y), &HFF&
        For y1 = y - 2 To y + 2
            For x1 = x - 2 To x + 2
                If y1 < 0 Then y1 = 0
                If x1 < 0 Then x1 = 0
                If x1 <> x And y1 <> y Then tmp(x1, y1) = 0
            Next
        Next
    End If

    Else
        tmp(x, y) = 0
    End If
End If
Next
DoEvents
Next

```

End Sub

Function needed (img, xn, yn) Non-existent function, unsure of purpose

```

    If img = 0 Then
    End If
End Function

```

Sub newsuppress () An implementation of Non-Maximal Suppression

```

For y = 2 To 237
    For x = 2 To 317
        n = Abs(sx(x, y))
        N2 = Abs(sy(x, y))
    Next
Next

```

```

n3 = tmp(x, y)
ymax = ((n3 > tmp(x, y - 1)) And (n3 >= tmp(x, y + 1)))
xmax = ((n3 > tmp(x - 1, y)) And (n3 >= tmp(x + 1, y)))
If (ymax And xmax) Or ((ymax And N2 > n) Or (xmax And n > N2)) Then
  'pnt(x, y) = tmp(x, y) * &H10101
Else
  pnt(x, y) = 0
  'tmp(x, y) = 0
End If
Next
updat y / 239 * 100
DoEvents
Next

End Sub

Sub postprocess ()
  My post-processing algorithm for after vectorization
  pi = 3.1415926535
  abc2:
  Do
    again = 0
    For a = 1 To lines
      For b = 1 To lines
        For c = 0 To 1
          For d = 0 To 1
            'linex(0, _) on left
            'liney(0, _) on top
            a1 = linex(c, a) - linex(d, b)
            b1 = liney(c, a) - liney(d, b)
            c1 = Sqr(a1 * a1 + b1 * b1)
            If c1 < 10 And a <> b Then
              On Error Resume Next
              ang1 = Atn((liney(0, a) - liney(1, a)) / (linex(0, a) - linex(1, a)))
              ang2 = Atn((liney(0, b) - liney(1, b)) / (linex(0, b) - linex(1, b)))

              picture1.Line (linex(0, a), liney(0, a))-(linex(1, a), liney(1, a)),
&HFF0000
              picture1.Line (linex(0, b), liney(0, b))-(linex(1, b), liney(1, b)),
&HFF00&
              'MsgBox Str(ang1 * 180 / pi) & Str(ang2 * 180 / pi)
              DoEvents
              picture1.Line (linex(0, a), liney(0, a))-(linex(1, a), liney(1, a)),
&HFFFFFFF
              picture1.Line (linex(0, b), liney(0, b))-(linex(1, b), liney(1, b)),
&HFFFFFFF

              If Abs((ang2 - ang1) * (180 / pi)) <= 5 Then
                MsgBox Str(ang1 * 180 / pi) & Str(ang2 * 180 / pi)
                linex(c, a) = linex(1 - d, b)
                liney(c, a) = liney(1 - d, b)
                lines = lines - 1
                'shift b + 1 and higher left
                For n = b To lines

```

```

        'MsgBox Str(linex(0, n)) + Str(liney(0, n)) + "-" +
Str(linex(1, n)) + Str(liney(1, n))
        linex(0, n) = linex(0, n + 1)
        liney(0, n) = liney(0, n + 1)
        linex(1, n) = linex(1, n + 1)
        liney(1, n) = liney(1, n + 1)
    Next
    GoTo abc2
Else
    linex(c, a) = linex(d, b)
    liney(c, a) = liney(d, b)
End If
End If
Next
Next
Next
Next
Loop While again
abc:
    picture1.Cls
    For n = 1 To lines
abc3:
        a = linex(0, n) - linex(1, n)
        b = liney(0, n) - liney(1, n)
        If Sqr(a * a + b * b) < 20 And lines > 0 Then 'eliminate nonexistant lines
            lines = lines - 1
            For t = n To lines
                linex(0, t) = linex(0, t + 1)
                liney(0, t) = liney(0, t + 1)
                linex(1, t) = linex(1, t + 1)
                liney(1, t) = liney(1, t + 1)
            Next
            GoTo abc3
        End If
        'MsgBox Str(linex(0, n)) + "," + Str(liney(0, n)) + "-" + Str(linex(1, n)) + "," +
Str(liney(1, n))
        picture1.Line (linex(0, n), liney(0, n))-(linex(1, n), liney(1, n)), 0
    Next
    MsgBox Str(lines) + " lines"
End Sub

```

Prewitt gradient edge detection
---------------------------------

```

Sub prewitt ()
For y = 1 To 238
    For x = 1 To 318
        n = retprewittx(x, y)
        N2 = retprewitty(x, y)
        n3 = Sqr(n * n + N2 * N2)
        If n3 > 255 Then n3 = 255
        tmp(x, y) = n3
    Next
    updat y / 239 * 100
Next
For y = 0 To 239

```

```

For x = 0 To 319
    picture1.PSet (x, y), tmp(x, y) * &H10101
Next
updat y / 239 * 100
Next
End Sub

```

Returns the x component of Frei-Chen edge detection

```

Function retfreichenx (x, y)
    a = (pnt(x - 1, y - 1) And &HFF&)
    b = (pnt(x, y - 1) And &HFF&)
    c = (pnt(x + 1, y - 1) And &HFF&)
    d = (pnt(x - 1, y) And &HFF&)
    e = (pnt(x, y) And &HFF&)
    f = (pnt(x + 1, y) And &HFF&)
    g = (pnt(x - 1, y + 1) And &HFF&)
    h = (pnt(x, y + 1) And &HFF&)
    i = (pnt(x + 1, y + 1) And &HFF&)

    a2 = a
    b2 = 0
    c2 = -c
    d2 = d * Sqr(2)
    e2 = 0
    f2 = f * -Sqr(2)
    g2 = g
    h2 = 0
    i2 = -i
    sx(x, y) = a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2
    retfreichenx = sx(x, y)

```

End Function

Returns the y component of Frei-Chen edge detection

```

Function retfreicheny (x, y)
    a = (pnt(x - 1, y - 1) And &HFF&)
    b = (pnt(x, y - 1) And &HFF&)
    c = (pnt(x + 1, y - 1) And &HFF&)
    d = (pnt(x - 1, y) And &HFF&)
    e = (pnt(x, y) And &HFF&)
    f = (pnt(x + 1, y) And &HFF&)
    g = (pnt(x - 1, y + 1) And &HFF&)
    h = (pnt(x, y + 1) And &HFF&)
    i = (pnt(x + 1, y + 1) And &HFF&)
    a2 = -a
    b2 = b * -Sqr(2)
    c2 = -c
    d2 = 0
    e2 = 0
    f2 = 0
    g2 = g
    h2 = h * Sqr(2)
    i2 = i
    sy(x, y) = a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2
    retfreicheny = sy(x, y)

```

End Function

Returns the x component of Level edge detection

Function retlevelx (x, y)

```
a2 = -(pnt(x - 2, y - 1) And &HFF&)  
c2 = (pnt(x + 2, y - 1) And &HFF&)  
d2 = -2 * (pnt(x - 2, y) And &HFF&)  
f2 = 2 * (pnt(x + 2, y) And &HFF&)  
g2 = -(pnt(x - 2, y + 1) And &HFF&)  
i2 = (pnt(x + 2, y + 1) And &HFF&)  
sx(x, y) = Int(a2 + c2 + d2 + f2 + g2 + i2)  
retlevelx = sx(x, y)
```

End Function

Returns the y component of Level edge detection

Function retlevely (x, y)

```
a2 = (pnt(x - 1, y - 2) And &HFF&)  
b2 = 2 * (pnt(x, y - 2) And &HFF&)  
c2 = (pnt(x + 1, y - 2) And &HFF&)  
g2 = -(pnt(x - 1, y + 2) And &HFF&)  
h2 = -2 * (pnt(x, y + 2) And &HFF&)  
i2 = -(pnt(x + 1, y + 2) And &HFF&)  
sy(x, y) = Int(a2 + b2 + c2 + g2 + h2 + i2)  
retlevely = sy(x, y)
```

End Function

Returns the x component of Prewitt edge detection

Function retprewittx (x, y)

```
a = (pnt(x - 1, y - 1) And &HFF&)  
b = (pnt(x, y - 1) And &HFF&)  
c = (pnt(x + 1, y - 1) And &HFF&)  
d = (pnt(x - 1, y) And &HFF&)  
e = (pnt(x, y) And &HFF&)  
f = (pnt(x + 1, y) And &HFF&)  
g = (pnt(x - 1, y + 1) And &HFF&)  
h = (pnt(x, y + 1) And &HFF&)  
i = (pnt(x + 1, y + 1) And &HFF&)  
a2 = a  
b2 = 0  
c2 = -c  
d2 = d  
e2 = 0  
f2 = -f  
g2 = g  
h2 = 0  
i2 = -i  
sx(x, y) = a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2  
retprewittx = sx(x, y)
```

End Function

Returns the y component of Prewitt edge detection

Function retprewitty (x, y)

```
a = (pnt(x - 1, y - 1) And &HFF&)  
b = (pnt(x, y - 1) And &HFF&)  
c = (pnt(x + 1, y - 1) And &HFF&)  
d = (pnt(x - 1, y) And &HFF&)  
e = (pnt(x, y) And &HFF&)  
f = (pnt(x + 1, y) And &HFF&)
```

```

g = (pnt(x - 1, y + 1) And &HFF&)
h = (pnt(x, y + 1) And &HFF&)
i = (pnt(x + 1, y + 1) And &HFF&)
a2 = -a
b2 = -b
c2 = -c
d2 = 0
e2 = 0
f2 = 0
g2 = g
h2 = h
i2 = i
sy(x, y) = a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2
retprewitty = sy(x, y)

```

End Function

Returns the x component of Roberts Cross edge detection
---

Function retrobcrossx (x, y)

```

a = (pnt(x - 1, y - 1) And &HFF&)
b = (pnt(x, y - 1) And &HFF&)
c = (pnt(x + 1, y - 1) And &HFF&)
d = (pnt(x - 1, y) And &HFF&)
e = (pnt(x, y) And &HFF&)
f = (pnt(x + 1, y) And &HFF&)
g = (pnt(x - 1, y + 1) And &HFF&)
h = (pnt(x, y + 1) And &HFF&)
i = (pnt(x + 1, y + 1) And &HFF&)
'a2 = A
'b2 = 0
'c2 = 0
'd2 = 0
'e2 = -e
'f2 = 0
'g2 = 0
'h2 = 0
'i2 = 0

a2 = 0
b2 = 0
c2 = c
d2 = 0
e2 = -e
f2 = 0
g2 = 0
h2 = 0
i2 = 0
'MsgBox e
sx(x, y) = Int(a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2)
retrobcrossx = sx(x, y)

```

End Function

Returns the y component of Roberts Cross edge detection
---

Function retrobcrossy (x, y)

```

a = (pnt(x - 1, y - 1) And &HFF&)
b = (pnt(x, y - 1) And &HFF&)

```

```

c = (pnt(x + 1, y - 1) And &HFF&)
d = (pnt(x - 1, y) And &HFF&)
e = (pnt(x, y) And &HFF&)
f = (pnt(x + 1, y) And &HFF&)
g = (pnt(x - 1, y + 1) And &HFF&)
h = (pnt(x, y + 1) And &HFF&)
i = (pnt(x + 1, y + 1) And &HFF&)

```

```

a2 = a
b2 = 0
c2 = 0
d2 = 0
e2 = -e
f2 = 0
g2 = 0
h2 = 0
i2 = 0

```

```

'a2 = 0
'b2 = 0
'c2 = C
'd2 = 0
'e2 = -e
'f2 = 0
'g2 = 0
'h2 = 0
'i2 = 0

```

```

sy(x, y) = Int(a2 + b2 + c2 + d2 + e2 + f2 + g2 + h2 + i2)
retrobcrossy = sy(x, y)

```

End Function

Returns the x component of Sobel edge detection
---

Function retsobelx (x, y)

```

a2 = -(pnt(x - 1, y - 1) And &HFF&)
c2 = (pnt(x + 1, y - 1) And &HFF&)
d2 = -2 * (pnt(x - 1, y) And &HFF&)
f2 = 2 * (pnt(x + 1, y) And &HFF&)
g2 = -(pnt(x - 1, y + 1) And &HFF&)
i2 = (pnt(x + 1, y + 1) And &HFF&)
sx(x, y) = Int(a2 + c2 + d2 + f2 + g2 + i2)
retsobelx = sx(x, y)

```

End Function

Returns the y component of Sobel edge detection
---

Function retsobely (x, y)

```

a2 = (pnt(x - 1, y - 1) And &HFF&)
b2 = 2 * (pnt(x, y - 1) And &HFF&)
c2 = (pnt(x + 1, y - 1) And &HFF&)
g2 = -(pnt(x - 1, y + 1) And &HFF&)
h2 = -2 * (pnt(x, y + 1) And &HFF&)
i2 = -(pnt(x + 1, y + 1) And &HFF&)
sy(x, y) = Int(a2 + b2 + c2 + g2 + h2 + i2)
retsobely = sy(x, y)

```

End Function

```

Sub robcross ()
  This does Roberts Cross edge detection on the image
  For y = 1 To 238
    For x = 1 To 318
      n = retrobcrossx(x, y)
      N2 = retrobcrossy(x, y)
      n3 = Sqr(n * n + N2 * N2)
      MsgBox n
      If n3 > 255 Then n3 = 255
      tmp(x, y) = n3
    Next
    DoEvents
    updat y / 239 * 100
  Next
  For y = 1 To 238
    For x = 1 To 318
      pnt(x, y) = tmp(x, y) * &H10101
    Next
  Next
End Sub

```

```

Sub robcrossx ()
  This does Roberts Cross detection - x part on the image
  For y = 1 To 238
    For x = 1 To 318
      n3 = retrobcrossx(x, y) / 2 + 128
      If n3 < 0 Then n3 = 0
      If n3 > 255 Then n3 = 255
      tmp(x, y) = n3
    Next
    updat y / 239 * 100
  Next
  For y = 0 To 239
    For x = 0 To 319
      picture1.PSet (x, y), tmp(x, y) * &H10101
    Next
    updat y / 239 * 100
  Next
End Sub

```

```

Sub root ()
  Takes the square root of each pixel in the image
  For y = 0 To 239
    For x = 0 To 319
      pnt(x, y) = Int(Sqr((pnt(x, y) And &HFF&)) * 15) * &H10101
      picture1.PSet (x, y), pnt(x, y)
    Next
    updat y / 239 * 100
    DoEvents
  Next
End Sub

```

```

Sub save3d ()
  Code for saving to my earlier 3D file format

```

```

Print #1, "D"Date$
Print #1, "T"
Print #1, thrds(1, 1)
Print #1, thrds(2, 1)
Print #1, thrds(3, 1)
Print #1, thrds(1, 2)
Print #1, thrds(2, 2)
Print #1, thrds(3, 2)
Print #1, "-"
Print #1, thrds(1, 2)
Print #1, thrds(2, 2)
Print #1, thrds(3, 2)
Print #1, thrds(1, 4)
Print #1, thrds(2, 4)
Print #1, thrds(3, 4)
Print #1, "-"
Print #1, thrds(1, 4)
Print #1, thrds(2, 4)
Print #1, thrds(3, 4)
Print #1, thrds(1, 3)
Print #1, thrds(2, 3)
Print #1, thrds(3, 3)
Print #1, "-"
Print #1, thrds(1, 1)
Print #1, thrds(2, 1)
Print #1, thrds(3, 1)
Print #1, thrds(1, 3)
Print #1, thrds(2, 3)
Print #1, thrds(3, 3)
Print #1, "|"
End Sub

Sub setcorner1 () 
  xn = 10
  yn = 11
End Sub

Sub setcorner2 () 
  xn = 305
  yn = 11
End Sub

Sub setcorner3 () 
  xn = 15
  yn = 224
End Sub

Sub setcorner4 () 
  xn = 305
  yn = 224
End Sub

Sub sets (n) 

```

```

For t = 1 To 4
    xs(n, t) = xn1(t) - 160
    ys(n, t) = yn1(t) - 120
Next
End Sub

Sub showlines ()
    Displays vector lines and vertices on the screen
MsgBox Str(lines) & "lines"
MsgBox Str(corners) & "corners"
For n = 1 To lines - 1
    picture1.Line (linex(0, n), liney(0, n))-(linex(1, n), liney(1, n)), &HFF&
Next
Open "c:\sf2004\corners.txt" For Output As #1
For n = 1 To corners
    picture1.PSet (corner(n, 1), corner(n, 2)), 0
    Print #1, Str(corner(n, 1)); ";"; Str(corner(n, 2))
Next
Close

End Sub

Sub sobel ()
    Runs Sobel edge detection on the image
For y = 1 To 238
    For x = 1 To 318
        n = retsobelx(x, y)
        N2 = retsobely(x, y)
        n3 = Sqr(n * n + N2 * N2)
        If n3 > 255 Then n3 = 255
        tmp(x, y) = n3
    Next
    updat y / 239 * 100
    DoEvents
Next
For y = 1 To 238
    For x = 1 To 318
        pnt(x, y) = tmp(x, y) * &H10101
    Next
Next
End Sub

Sub sobel2 ()
    Runs Sobel & Non-maximal suppression at the same time
ReDim lins(320, 8) As Integer
y2 = 0
'stores first 4 lines of video
For y = 1 To 4
    For x = 0 To 319
        lins(x, y) = pnt(x, y2) And &HFF&
    Next
    y2 = y2 + 1
Next
'sobel on lines 1 - 3, to first line
For x = 1 To 318
    ' y = 1

```

```

' v = lins(x - 1, y) + 2 * lins(x, y) + lins(x + 1, y) - lins(x - 1, y + 2) - 2 * lins(x, y
+ 2) - lins(x + 1, y + 2)
' h = -lins(x - 1, y) - 2 * lins(x - 1, y + 1) - lins(x - 1, y + 2) + lins(x + 1, y) + 2 *
lins(x + 1, y + 1) + lins(x + 1, y + 2)
' lins(x - 1, y) = Sqr(v * v + h * h)

'sobel on lines 2 - 4, to second line
y = 2
v = lins(x - 1, y) + 2 * lins(x, y) + lins(x + 1, y) - lins(x - 1, y + 2) - 2 * lins(x, y
+ 2) - lins(x + 1, y + 2)
h = -lins(x - 1, y) - 2 * lins(x - 1, y + 1) - lins(x - 1, y + 2) + lins(x + 1, y) + 2 *
lins(x + 1, y + 1) + lins(x + 1, y + 2)
lins(x - 1, y) = Sqr(v * v + h * h)
lins(x - 1, 5) = Abs(h) > Abs(v)
lins(x - 1, 6) = lins(x, 2) > lins(x, 1)
Next
'picture1.Line (x - 1, y)-(x - 1, y + 10), lins(x - 1, 7) * -&HFFFFFFF
'find if horizontal maximum
For x = 1 To 318
    lins(x - 1, 7) = (lins(x, 2) > lins(x - 1, 2)) And (lins(x, 2) >= lins(x + 1, 2))
    'picture1.Line (x - 1, y)-(x - 1, y + 10), lins(x - 1, 7) * -&HFFFFFFF
Next
'rotate lines
For x = 0 To 319
    lins(x, 1) = lins(x, 2)
    lins(x, 2) = lins(x, 3)
    lins(x, 3) = lins(x, 4)
    lins(x, 4) = pnt(x, y2) And &HFF&
Next
y2 = y2 + 1

'sobel on lines 2 - 4, to second line
y = 2
For x = 1 To 318
    v = lins(x - 1, y) + 2 * lins(x, y) + lins(x + 1, y) - lins(x - 1, y + 2) - 2 * lins(x, y
+ 2) - lins(x + 1, y + 2)
    h = -lins(x - 1, y) - 2 * lins(x - 1, y + 1) - lins(x - 1, y + 2) + lins(x + 1, y) + 2 *
lins(x + 1, y + 1) + lins(x + 1, y + 2)
    lins(x - 1, y) = Sqr(v * v + h * h)
    lins(x - 1, 8) = Abs(h) > Abs(v)
    lins(x - 1, 6) = (lins(x, 2) > lins(x, 1)) And lins(x - 1, 6)
Next
'do non maximal suppression
For x = 1 To 318
    If (lins(x, 6) And (lins(x + 1, 1) >= lins(x + 1, 2)) And lins(x, 7)) Or (lins(x, 6)
And (lins(x + 1, 1) >= lins(x + 1, 2)) And Not lins(x, 5)) Or (lins(x, 7) And lins(x,
5)) Then
        tmp(x, y2 - 5) = lins(x, 1)
        'picture1.PSet (x, y2 - 4), &HFFFFFFF
    Else
        tmp(x, y2 - 5) = 0
    End If
Next

```

```

repsobel:
'get maximums, h vs v
For x = 1 To 318
    lins(x - 1, 6) = lins(x, 2) > lins(x, 1)
    lins(x - 1, 7) = (lins(x, 2) > lins(x - 1, 2)) And (lins(x, 2) >= lins(x + 1, 2))
    lins(x, 5) = lins(x, 8)
Next
'rotate lines
For x = 0 To 319
    'MsgBox Str(lins(x, 2)) & Str(lins(x, 1))
    lins(x, 1) = lins(x, 2)
    'MsgBox Str(lins(x, 2)) & Str(lins(x, 1))
    lins(x, 2) = lins(x, 3)
    lins(x, 3) = lins(x, 4)
    lins(x, 4) = pnt(x, y2) And &HFF&
Next
y2 = y2 + 1
'sobel on lines 2 - 4, to second line
y = 2
For x = 1 To 318
    v = lins(x - 1, y) + 2 * lins(x, y) + lins(x + 1, y) - lins(x - 1, y + 2) - 2 * lins(x, y
+ 2) - lins(x + 1, y + 2)
    h = -lins(x - 1, y) - 2 * lins(x - 1, y + 1) - lins(x - 1, y + 2) + lins(x + 1, y) + 2 *
lins(x + 1, y + 1) + lins(x + 1, y + 2)
    lins(x - 1, 2) = Sqr(v * v + h * h)
    lins(x - 1, 8) = Abs(h) > Abs(v)
Next
'do non maximal suppression
For x = 1 To 318
    If (lins(x, 6) And (lins(x + 1, 1) >= lins(x + 1, 2)) And lins(x, 7)) Or (lins(x, 6)
And (lins(x + 1, 1) >= lins(x + 1, 2)) And Not lins(x, 5)) Or (lins(x, 7) And lins(x,
5)) Then
        tmp(x, y2 - 5) = lins(x, 1)
    Else
        tmp(x, y2 - 5) = 0
    End If
Next
updat y2 / 238 * 100
If y2 < 238 Then GoTo repsobel

For y = 1 To 238
    For x = 1 To 318
        If tmp(x, y) > 255 Then tmp(x, y) = 255
        pnt(x, y) = tmp(x, y) * &H10101
    Next
Next

End Sub

Sub sobelx () Runs x component of Sobel edge detection
For y = 1 To 238
    For x = 1 To 318

```

```

    n3 = retsobelx(x, y) / 2 + 128
    If n3 < 0 Then n3 = 0
    If n3 > 255 Then n3 = 255
    tmp(x, y) = n3
    picture1.PSet (x, y), tmp(x, y) * &H10101
Next
DoEvents
updat y / 239 * 100
Next
End Sub

```

```

Sub sobely () Runs y component of Sobel edge detection
For y = 1 To 238
    For x = 1 To 318
        n3 = retsobely(x, y) / 2 + 128
        If n3 < 0 Then n3 = 0
        If n3 > 255 Then n3 = 255
        tmp(x, y) = n3
        picture1.PSet (x, y), tmp(x, y) * &H10101
    Next
    updat y / 239 * 100
    DoEvents
Next
End Sub

```

```

Sub squared () Squares each pixel in the image
For y = 0 To 239
    For x = 0 To 319
        pnt(x, y) = Int((pnt(x, y) And &HFF&) * (pnt(x, y) And &HFF&) / 255) *
&H10101
        picture1.PSet (x, y), pnt(x, y)
    Next
    updat y / 239 * 100
    DoEvents
Next
End Sub

```

```

Sub suppression () Original slow form of non-maximal suppression
For y = 1 To 238
    For x = 1 To 318

        n = sx(x, y)
        o = sy(x, y)

        If Abs(o) > Abs(n) Then
            deltay = -1.5
            deltay = -(o > 0) * 3 + deltay
            deltax = (n / o) * deltay
        ElseIf Abs(o) < Abs(n) Then
            deltax = -1.5
            deltax = -(n > 0) * 3 + deltax
            deltay = (o / n) * deltax

```

```

ElseIf n <> 0 Then
    deltax = -1.5
    deltax = -(n > 0) * 3 + deltax
    deltay = -1.5
    deltay = -(o > 0) * 3 + deltay
Else
    deltax = 0
    deltay = 0
End If

If deltax > 0 Then
    deli = 1
ElseIf deltax < 0 Then
    deli = -1
Else
    deli = 0
End If

If deltay > 0 Then
    delj = 1
ElseIf deltay < 0 Then
    delj = -1
Else
    delj = 0
End If

n1 = (n * (1 - deli * deltax) + sx(x + deli, y) * deli * deltax) * (1 - delj *
deltay) + (sx(x, y + delj) * (1 - deli * deltax) + sx(x + deli, y + delj) * deli * deltax)
* delj * deltay
N2 = (o * (1 - deli * deltax) + sy(x + deli, y) * deli * deltax) * (1 - delj *
deltay) + (sy(x, y + delj) * (1 - deli * deltax) + sy(x + deli, y + delj) * deli * deltax)
* delj * deltay
n3 = (n * (1 - deli * deltax) + sx(x - deli, y) * deli * deltax) * (1 - delj *
deltay) + (sx(x, y - delj) * (1 - deli * deltax) + sx(x - deli, y - delj) * deli * deltax) *
delj * deltay
n4 = (o * (1 - deli * deltax) + sy(x - deli, y) * deli * deltax) * (1 - delj *
deltay) + (sy(x, y - delj) * (1 - deli * deltax) + sy(x - deli, y - delj) * deli * deltax) *
delj * deltay

'MsgBox Str(Sqr(n1 * n1 + n2 * n2)) + Str(Sqr(n3 * n3 + n4 * n4)) +
Str(Sqr(n * n + o * o))

If ((n * n + o * o) <= (n1 * n1 + N2 * N2)) Or ((n * n + o * o) <= (n3 * n3 +
n4 * n4)) Then
    sx(x, y) = 0
    sy(x, y) = 0
    n = 0
    o = 0
End If
p = Sqr(n * n + o * o)
If p > 255 Then p = 255
tmp(x, y) = p

```

```

    'picture1.PSet (x, y), tmp(x, y) * &H10101
Next
  updat y / 239 * 100
Next
For y = 1 To 238
  For x = 1 To 318
    picture1.PSet (x, y), tmp(x, y) * &H10101
  Next
  updat y / 239 * 100
DoEvents
Next
End Sub

```

```

Sub suppression2 ()
  For y = 1 To 238
    For x = 1 To 318

```

Another older form of non-maximal suppression

```

      n = sy(x, y)
      o = sx(x, y)

      If Abs(o) > Abs(n) Then
        deltay = -1.5
        deltay = -(o > 0) * 3 + deltay
        deltax = (n / o) * deltay
      ElseIf Abs(o) < Abs(n) Then
        deltax = -1.5
        deltax = -(n > 0) * 3 + deltax
        deltay = (o / n) * deltax
      ElseIf n <> 0 Then
        deltax = -1.5
        deltax = -(n > 0) * 3 + deltax
        deltay = -1.5
        deltay = -(o > 0) * 3 + deltay
      Else
        deltax = 0
        deltay = 0
      End If

      If deltax > 0 Then
        deli = 1
      ElseIf deltax < 0 Then
        deli = -1
      Else
        deli = 0
      End If

      If deltay > 0 Then
        delj = 1
      ElseIf deltay < 0 Then
        delj = -1
      Else
        delj = 0
      End If

```

```

n1 = (n * (1 - deli * deltax) + sy(x + deli, y) * deli * deltax) * (1 - delj *
deltay) + (sy(x, y + delj) * (1 - deli * deltax) + sy(x + deli, y + delj) * deli * deltax)
* delj * deltax

```

```

N2 = (o * (1 - deli * deltax) + sx(x + deli, y) * deli * deltax) * (1 - delj *
deltay) + (sx(x, y + delj) * (1 - deli * deltax) + sx(x + deli, y + delj) * deli * deltax)
* delj * deltax

```

```

n3 = (n * (1 - deli * deltax) + sy(x - deli, y) * deli * deltax) * (1 - delj *
deltay) + (sy(x, y - delj) * (1 - deli * deltax) + sy(x - deli, y - delj) * deli * deltax) *
delj * deltax

```

```

n4 = (o * (1 - deli * deltax) + sx(x - deli, y) * deli * deltax) * (1 - delj *
deltay) + (sx(x, y - delj) * (1 - deli * deltax) + sx(x - deli, y - delj) * deli * deltax) *
delj * deltax

```

```

'MsgBox Str(Sqr(n1 * n1 + n2 * n2)) + Str(Sqr(n3 * n3 + n4 * n4)) +
Str(Sqr(n * n + o * o))

```

```

If ((n * n + o * o) <= (n1 * n1 + N2 * N2)) Or ((n * n + o * o) <= (n3 * n3 +
n4 * n4)) Then

```

```

n = 0

```

```

o = 0

```

```

End If

```

```

p = Sqr(n * n + o * o)

```

```

If p > 255 Then p = 255

```

```

tmp(x, y) = p

```

```

'picture1.PSet (x, y), tmp(x, y) * &H10101

```

```

Next

```

```

updat y / 239 * 100

```

```

Next

```

```

For y = 1 To 238

```

```

For x = 1 To 318

```

```

picture1.PSet (x, y), tmp(x, y) * &H10101

```

```

Next

```

```

updat y / 239 * 100

```

```

Next

```

```

End Sub

```

```

Sub susan (thresh, thresh2) This is the SUSAN corner finder. It worked with poor results

```

```

'thresh = 20

```

```

For y = 1 To 238

```

```

For x = 1 To 318

```

```

o = pnt(x, y) And &HFF&

```

```

a = pnt(x - 1, y - 1) And &HFF&

```

```

b = pnt(x, y - 1) And &HFF&

```

```

c = pnt(x + 1, y - 1) And &HFF&

```

```

d = pnt(x - 1, y) And &HFF&

```

```

e = pnt(x + 1, y) And &HFF&

```

```

f = pnt(x - 1, y + 1) And &HFF&

```

```

g = pnt(x, y + 1) And &HFF&

```

```

h = pnt(x + 1, y + 1) And &HFF&

```

```

'MsgBox -(Abs(o - a) <= thresh)

```

```

    amt = (-(Abs(o - a) <= thresh)) + (-(Abs(o - b)) <= thresh) + (-(Abs(o - c)
<= thresh)) + (-(Abs(o - d) <= thresh)) + (-(Abs(o - e) <= thresh)) + (-(Abs(o - f)
<= thresh)) + (-(Abs(o - g) <= thresh)) + (-(Abs(o - h) <= thresh))
    'MsgBox amt

```

```

    If amt < thresh2 Then
        amt2 = thresh2 - amt
    Else
        amt2 = 0
    End If
    'MsgBox amt2
    picture1.PSet (x, y), Int(amt2 * 128 / thresh2) * &H10101
Next
    updat y / 239 * 100
    DoEvents
Next

```

End Sub

Sub thinning ()

One of the slow thinning methods

```

'on error GoTo er
    For yn = 0 To 239
        For xn = 0 To 319
            pnt2(xn, yn) = pnt(xn, yn)
        Next
    Next
Next

```

Do

```

again = 0
For y = 1 To 238
    updat y / 238 * 100
    For x = 1 To 318
        e = (pnt(x, y) > &H808080)
        If e Then
            co = connected(0, x, y)
            'If co > 0 Then MsgBox "co: " + Str(co)
            If co >= 2 And co <= 6 Then
                cr = crossing(0, x, y)
                If cr = 1 Then
                    b = (pnt(x, y - 1) < &H808080)
                    d = (pnt(x - 1, y) < &H808080)
                    f = (pnt(x + 1, y) < &H808080)
                    h = (pnt(x, y + 1) < &H808080)
                    '
                    ' b
                    'd f          ' f
                    ' h   v      ' h   >
                If (d Or h Or f) And (h Or f Or b) Then
                    pnt2(x, y) = 0
                    again = -1
                End If
            End If
        End If
    Next
Next

```

```

        End If
    End If
Next
Next

For yn = 0 To 239
    For xn = 0 To 319
        pnt(xn, yn) = pnt2(xn, yn)
    Next
Next

For y = 1 To 238
    updat y / 238 * 100
    For x = 1 To 318
        e = (pnt(x, y) > &H808080)
        If e Then
            co = connected(0, x, y)
            If co >= 2 And co <= 6 Then
                cr = crossing(0, x, y)
                If cr = 1 Then
                    b = (pnt(x, y - 1) < &H808080)
                    d = (pnt(x - 1, y) < &H808080)
                    f = (pnt(x + 1, y) < &H808080)
                    h = (pnt(x, y + 1) < &H808080)
                    ' b          ' b
                    ' d          ' d f
                    ' h <          ' ^
                    If (d Or h Or b) And (d Or f Or b) Then
                        pnt2(x, y) = 0
                        again = -1
                    End If
                End If
            End If
        End If
    Next
Next

For yn = 0 To 239
    For xn = 0 To 319
        pnt(xn, yn) = pnt2(xn, yn)
    Next
Next

Loop While again

For y = 0 To 239
    updat y / 239 * 100
    For x = 0 To 319
        picture1.PSet (x, y), pnt(x, y)
    Next
    DoEvents
Next

```

Exit Sub

er:  
MsgBox Str(Err)  
Resume Next

End Sub

```
Sub thinning2 () Another of the slow thinning methods
    Do
        again = 0
        For y = 1 To 238
            For x = 1 To 318
                If pnt(x, y) > &H808080 Then
                    If crossing(0, x, y) < 2 Then pnt(x, y) = 0: again = -1
                End If
            Next
            updat y / 238 * 100
        Next
        For y = 238 To 1 Step -1
            For x = 318 To 1 Step -1
                If pnt(x, y) > &H808080 Then
                    If crossing(0, x, y) < 2 Then pnt(x, y) = 0: again = -1
                End If
            Next
            updat y / 238 * 100
        Next
        Loop While again
        For y = 1 To 238
            For x = 1 To 318
                picture1.PSet (x, y), pnt(x, y)
            Next
            updat y / 238 * 100
        DoEvents
    Next
End Sub
```

```
Sub thinning3 () Another of the slow thinning methods
    For y = 1 To 238
        For x = 1 To 318
            a = pnt(x - 1, y - 1) > &H808080
            c = pnt(x + 1, y - 1) > &H808080
            e = pnt(x, y) > &H808080
            f = pnt(x + 1, y) > &H808080
            g = pnt(x - 1, y + 1) > &H808080
            h = pnt(x, y + 1) > &H808080
            i = pnt(x + 1, y + 1) > &H808080
            'e = Not e
            'f = Not f
            'h = Not h
            'i = Not i

            '|A|B|C|
```

```

'D|E|F|
'G|H|I|
If e Then
  If connected(0, x, y) > 2 And Not i And Not a Then 'Or Not f Or Not h
    pnt(x, y) = 0
  Else
    pnt(x, y) = &HFFFFFF
  End If
Else
  pnt(x, y) = 0
End If
picture1.PSet (x, y), pnt(x, y)
Next
updat (y / 240) * 100
Next
End Sub

```

```

Sub thinning4 () Another of the slow thinning methods
'a
'b c
'd
Do
  again = 0
  For y = 1 To 238
    For x = 1 To 318
      a = pnt(x, y - 1) > &H808080
      b = pnt(x - 1, y) > &H808080
      c = pnt(x + 1, y) > &H808080
      d = pnt(x, y + 1) > &H808080
      If pnt(x, y) > &H808080 And crossing(0, x, y) = 1 And ((a And c And d And
Not b) Or (b And c And d And Not a) Or (a And b And d And Not c) Or (a And b And c
And Not d) Or (c And d And Not (a And b)) Or (b And d And Not (a And c)) Or (a And
b And Not (c And d)) Or (a And c And Not (b And d))) Then
        pnt(x, y) = 0
        again = -1
      End If
      If Not again Then picture1.PSet (x, y), pnt(x, y)
    Next
    updat y / 238 * 100
  Next
Loop While again
End Sub

```

```

Sub threed (O1, O2, yc, d, s, e) Does basic 3D conversion - updated in other programs
'yc = 8.5
'd = 2
's = 10
'e = 414.2
pi = 3.1415926535
O1 = O1 * pi / 180
O2 = O2 * pi / 180
For t = 1 To 4
  x1 = xs(1, t)

```

```

x2 = xs(2, t)
y1 = ys(1, t)
y2 = ys(2, t)
O1 = O1 * pi / 180
O2 = O2 * pi / 180
a = Cos(O1) + x1 / e * Sin(O1)
b = Cos(O2) + x2 / e * Sin(O2)
c = d * (Cos(O2) - Cos(O1))
f = Sin(O1) + x1 / e * Cos(O1)
g = Sin(O2) + x2 / e * Cos(O2)
h = d * (Sin(O2) - Sin(O1))
n = (c * f + a * (2 * s - h)) / (a * g - b * f)
N2 = (c * g + b * (2 * s - h)) / (a * g - b * f)
xiii = -g * n - d * Sin(O2) + s
yiii = y2 / e * n + yc
ziii = b * n + d * Cos(O2)
xiii2 = -f * N2 - d * Sin(O1) - s
yiii2 = y1 / e * N2 + yc
ziii2 = a * N2 + d * Cos(O1)
MsgBox Str(x1) + Str(y1) + Str(x2) + Str(y2)
MsgBox Str(xiii) + Str(yiii) + Str(ziii)
Next
End Sub

```

Compares each pixel to a threshold, and outputs black/white

```

Sub threshold (n)
For y = 0 To 239
For x = 0 To 319
If (pnt(x, y) And &HFF00&) / &H100& >= n Then
tmp(x, y) = 1
Else
tmp(x, y) = 0
End If
picture1.PSet (x, y), tmp(x, y) * 255 * &H10101
Next
updat y / 239 * 100
DoEvents
Next
End Sub

```

Part of form interface code

```

Sub Timer1_Timer ()
frmmain.Show
End Sub

```

Stores the image currently being shown

```

Sub updat (n1)
n1 = n1 / 100 * 239
picture2.Cls
picture2.Line (0, 0)-(7, n1), &HCCCCCC, BF
picture2.Line (0, n1 + 1)-(7, n1 + 1), &H0
End Sub

```

Updates the progress bar to show status

```

Sub updatdir ()
stuck = -1
t = &H808080

```

```

    On Error Resume Next
    If Not ((dirs = 1 And pnt2(x, y - 1) > t) Or (dirs = 2 And pnt2(x + 1, y) > t) Or
(dirs = 3 And pnt2(x, y + 1) > t) Or (dirs = 4 And pnt2(x - 1, y) > t)) Then
        If pnt2(x, y - 1) > t Then
            stuck = 0
            dirs = 1
        End If
        If pnt2(x + 1, y) > t Then
            stuck = 0
            dirs = 2
        End If
        If pnt2(x, y + 1) > t Then
            stuck = 0
            dirs = 3
        End If
        If pnt2(x - 1, y) > t Then
            stuck = 0
            dirs = 4
        End If
    Else
        stuck = 0
    End If
End Sub

```

```

Sub vector () Original 2003 vectorization method
    setcorner1
    corner1
    setcorner2
    corner2
    setcorner3
    corner3
    setcorner4
    corner4
    picture1.Cls
    picture1.Line (xn1(1), yn1(1))-(xn1(2), yn1(2)), &H0
    picture1.Line (xn1(2), yn1(2))-(xn1(4), yn1(4)), &H0
    picture1.Line (xn1(3), yn1(3))-(xn1(4), yn1(4)), &H0
    picture1.Line (xn1(3), yn1(3))-(xn1(1), yn1(1)), &H0
    MsgBox Str(xn1(1)) + "," + Str(yn1(1))
    MsgBox Str(xn1(1) - 160) + "," + Str(yn1(1) - 160)
    MsgBox Str(xn1(2)) + "," + Str(yn1(2))
    MsgBox Str(xn1(2) - 160) + "," + Str(yn1(2) - 160)
    MsgBox Str(xn1(3)) + "," + Str(yn1(3))
    MsgBox Str(xn1(3) - 160) + "," + Str(yn1(3) - 160)
    MsgBox Str(xn1(4)) + "," + Str(yn1(4))
    MsgBox Str(xn1(4) - 160) + "," + Str(yn1(4) - 160)
End Sub

```

```

Sub vector2 () Current vectorization method
    oldy = 5
    corners = 0
    'On Error GoTo 0
    For y = 0 To 240

```

```

    For x = 0 To 320
        pnt2(x, y) = pnt(x, y)
    Next
Next
For y = 5 To 235
    For x = 5 To 315
        If pnt(x, y) > &H808080 Then
            c = connected(0, x, y)
            If c <> 2 And c > 0 Then
                addcorn x, y
            End If
        End If
    Next
Next
Next

started = 0
lines = 0
vecst:
    good = 0
    'try to find a corner starting point
    For i = 1 To corners
        x2 = corner(i, 1)
        y2 = corner(i, 2)
        If connected(0, x2, y2) > 0 Then
            x = x2
            y = y2
            good = -1
            GoTo nxt
        End If
    Next
    If good Then GoTo nxt
    'try to find not 2-connected starting point.
    ' For y = 2 To 238
    '     For x = 2 To 318
    '         If pnt(x, y) > &H808080 Then
    '             'n = 0
    '             c = connected(0, x, y)
    '             If c <> 2 And c > 0 Then
    '                 good = -1
    '                 GoTo nxt
    '             End If
    '         End If
    '     Next
    '     'oldy = y
    ' Next
    ' If good Then GoTo nxt
    'find any starting point

    For y = oldy To 235
        For x = 5 To 315
            If pnt(x, y) > &H808080 Then
                'n = 0
                'c = connected(0, x, y)

```

```

        'If c <> 2 And c > 0 Then
        addcorn x, y

            good = -1
            GoTo nxt
        'End If
    End If
Next
    oldy = y
Next
nxt:
    'If Not good Then '0 not 2 connected
    ' For y = 1 To 238 'ok, find a 2-connected
    '     For x = 1 To 318
    '         If pnt(x, y) > &H808080 Then
    '             c = connected(0, x, y)
    '             If c = 2 Then
    '                 good = -1
    '                 GoTo nxt
    '             End If
    '         End If
    '     Next
    ' Next
    ' If Not good Then
    '     MsgBox lines & "lines"
    '     vector3
    '     Exit Sub 'none connected
    ' End If
    'End If
    If good Then
        pnt(x, y) = 0
        pts = 0
        feats(0, 0) = x
        feats(1, 0) = y
        'If Not started Then
        '     corners = corners + 1
        '     corner(corners, 1) = x
        '     corner(corners, 2) = y
        '     picture1.PSet (x, y), &HFF&
        'End If
        started = -1
        noedge = 0
    ve:    good = 0
        'If (pnt(x - 1, y - 1) > &H808080) Then
        '     If chord(x - 1, y - 1, pts) <> 100000 Then
        '         pts = pts + 1
        '         x = x - 1
        '         y = y - 1
        '         good = -1
        '         feats(0, pts) = x
        '         feats(1, pts) = y
        '     End If
        'End If

```

```

'chordout = pts + 1
On Error Resume Next
'If connected(1, x, y) = 1 Then
'  corners = corners + 1
'  corner(corners, 1) = x
'  corner(corners, 2) = y
'  picture1.PSet (x, y), &HFF&
'End If
If Not good And (pnt(x, y - 1) > &H808080) Then
  good = 2
  chordout = chord(x, y - 1, pts)
  If chordout = 0 Then
    pts = pts + 1
    y = y - 1
    good = -1
    feats(0, pts) = x
    feats(1, pts) = y
  End If
End If
'If Not good And (pnt(x + 1, y - 1) > &H808080) Then
'  If chord(x + 1, y - 1, pts) <> 100000 Then
'    pts = pts + 1
'    x = x + 1
'    y = y - 1
'    good = -1
'    feats(0, pts) = x
'    feats(1, pts) = y
'  End If
'End If
If Not good And (pnt(x - 1, y) > &H808080) Then
  good = 2
  chordout = chord(x - 1, y, pts)
  If chordout = 0 Then
    pts = pts + 1
    x = x - 1
    good = -1
    feats(0, pts) = x
    feats(1, pts) = y
  End If
End If
If Not good And (pnt(x, y + 1) > &H808080) Then
  good = 2
  chordout = chord(x, y + 1, pts)
  If chordout = 0 Then
    pts = pts + 1
    y = y + 1
    good = -1
    feats(0, pts) = x
    feats(1, pts) = y
  End If
End If
If Not good And (pnt(x + 1, y) > &H808080) Then
  good = 2

```

```

    chordout = chord(x + 1, y, pts)
    If chordout = 0 Then
        pts = pts + 1
        x = x + 1
        good = -1
        feats(0, pts) = x
        feats(1, pts) = y
    End If
End If
'If Not good And (pnt(x - 1, y + 1) > &H808080) Then
'    If chord(x - 1, y + 1, pts) <> 100000 Then
'        pts = pts + 1
'        x = x - 1
'        y = y + 1
'        good = -1
'        feats(0, pts) = x
'        feats(1, pts) = y
'    End If
'End If
'If Not good And (pnt(x + 1, y + 1) > &H808080) Then
'    If chord(x + 1, y + 1, pts) <> 100000 Then
'        pts = pts + 1
'        x = x + 1
'        y = y + 1
'        good = -1
'        feats(0, pts) = x
'        feats(1, pts) = y
'    End If
'End If

If x <= 1 Or y <= 1 Or x >= 318 Or y >= 238 Then good = 0
'If good = 0 Then noedge = -1
If good Then
    For i = 1 To corners
        x2 = corner(i, 1)
        y2 = corner(i, 2)
        If x = x2 And y = y2 Then good = 2
    Next
End If
If Not good Then
    'end line
If pts > 20 Then
    lines = lines + 1
    linex(0, lines) = feats(0, 0)
    liney(0, lines) = feats(1, 0)
    linex(1, lines) = feats(0, pts)
    liney(1, lines) = feats(1, pts)
    'MsgBox Str(pts) & Str(chordout)
    'For rstr = chordout To pts
    '    pnt(feats(0, rstr), feats(1, rstr)) = &HFFFFFF
    '    picture1.PSet (feats(0, rstr), feats(1, rstr)), &HFF0000
'Next

```

```

'      If pts > 5 Then
'      lines = lines + 1
'      linex(0, lines) = feats(0, 0)
'      liney(0, lines) = feats(1, 0)
'      linex(1, lines) = feats(0, pts - 5)
'      liney(1, lines) = feats(1, pts - 5)
'      On Error Resume Next
'      pnt(feats(0, pts - 4), feats(1, pts - 4)) = &HFFFFFF
'      pnt(feats(0, pts - 3), feats(1, pts - 3)) = &HFFFFFF
'      pnt(feats(0, pts - 2), feats(1, pts - 2)) = &HFFFFFF
'      pnt(feats(0, pts - 1), feats(1, pts - 1)) = &HFFFFFF
'      pnt(feats(0, pts), feats(1, pts)) = &HFFFFFF
'      picture1.Line (feats(0, 0), feats(1, 0))-(feats(0, pts), feats(1, pts)), &HFF&
DoEvents
End If
'      MsgBox good
'      End If
'      'MsgBox "Line done"
'      If good = 0 Then GoTo vecst
'      pnt(x, y) = 0
'      pts = 0
'      feats(0, 0) = x
'      feats(1, 0) = y
'      GoTo ve
Else
'      pnt(x, y) = 0
'      picture1.PSet (x, y), &HFF00&
DoEvents
GoTo ve
End If
End If
End Sub

Sub vector3 () "Connect the dots" part of current vectorization
For x = 0 To 320
  For y = 0 To 240
    pnt(x, y) = pnt2(x, y)
  Next
Next

'MsgBox corners
lines = 1
'cornnow = 1
'x = corner(cornnow, 1)
'y = corner(cornnow, 2)
findstcorner
Do
  updatdir
  pnt2(x, y) = 0
  Select Case dirs
  Case 1
    y = y - 1
  Case 2

```

```

    x = x + 1
Case 3
    y = y + 1
Case 4
    x = x - 1
End Select
pnt2(x, y) = 0
If x < 1 Then x = 1
If y < 1 Then y = 1
If x > 318 Then x = 318
If y > 238 Then y = 238
picture1.PSet (x, y), &HFFFF&
DoEvents
cornerchk
pnt2(x, y) = 0
If stuck Then findstcorner
Loop While Not stuck
showlines
End Sub

```

```

Sub vectordn ()

```

Unsuccessful test of vectorizing the image in one direction

```

' / | \
'On Error GoTo 0
lines = 0
vecstdn:
    good = 0
    'try to find starting point.
    For y = 1 To 238
        For x = 1 To 318
            If pnt(x, y) > &H808080 Then
                good = -1
                GoTo nxtcdn
            End If
        Next
    Next
nxtcdn:
    If Not good Then
        Exit Sub 'none connected
    End If
    If good Then
        pnt(x, y) = 0
        pts = 0
        feats(0, 0) = x
        feats(1, 0) = y
vedn:    good = 0
        If Not good And (pnt(x + 1, y) > &H808080) Then
            If chord(x + 1, y, pts) <> 100000 Then
                pts = pts + 1
                x = x + 1
                n = y
                Do Until pnt(x, n) = 0
                    pnt(x, n) = 0

```

```

        n = n - 1
    Loop
    n = y
    Do Until pnt(x, n) = 0
        pnt(x, n) = 0
        n = n + 1
    Loop
    good = -1
    feats(0, pts) = x
    feats(1, pts) = y
End If
End If
If Not good And (pnt(x - 1, y + 1) > &H808080) Then
    If chord(x - 1, y + 1, pts) <> 100000 Then
        pts = pts + 1
        x = x - 1
        y = y + 1

        N2 = x
        n = y
        Do Until pnt(N2, n) = 0
            pnt(N2, n) = 0
            n = n - 1
            N2 = N2 - 1
        Loop
        N2 = x
        n = y
        Do Until pnt(N2, n) = 0
            pnt(N2, n) = 0
            n = n + 1
            N2 = N2 + 1
        Loop

        good = -1
        feats(0, pts) = x
        feats(1, pts) = y
    End If
End If
If Not good And (pnt(x, y + 1) > &H808080) Then
    If chord(x, y + 1, pts) <> 100000 Then
        pts = pts + 1
        y = y + 1

        n = x
        Do Until pnt(n, y) = 0
            pnt(n, y) = 0
            n = n - 1
        Loop
        n = x
        Do Until pnt(n, y) = 0
            pnt(n, y) = 0
            n = n + 1

```

```

Loop
    good = -1
    feats(0, pts) = x
    feats(1, pts) = y
End If
End If
'
If Not good And (pnt(x + 1, y + 1) > &H808080) Then
'
    If chord(x + 1, y + 1, pts) <> 100000 Then
'
        pts = pts + 1
'
        x = x + 1
'
        y = y + 1
'
'
        N2 = x
'
        n = y
'
        Do Until pnt(N2, n) = 0
'
            pnt(N2, n) = 0
'
            n = n - 1
'
            N2 = N2 + 1
'
        Loop
'
        N2 = x
'
        n = y
'
        Do Until pnt(N2, n) = 0
'
            pnt(N2, n) = 0
'
            n = n + 1
'
            N2 = N2 - 1
'
        Loop
'
'
        good = -1
'
        feats(0, pts) = x
'
        feats(1, pts) = y
'
    End If
'
End If
'
If x < 1 Or y < 1 Or x > 318 Or y > 238 Then good = 0
'If connected(0, x, y) <> 2 Then good = 0
'If Not good Then
'end line
'
'lines = lines + 1
'linex(0, lines) = feats(0, 0)
'liney(0, lines) = feats(1, 0)
'linex(1, lines) = feats(0, pts)
'liney(1, lines) = feats(1, pts)
'
'If pts > 5 Then
lines = lines + 1
linex(0, lines) = feats(0, 0)
liney(0, lines) = feats(1, 0)
linex(1, lines) = feats(0, pts)
liney(1, lines) = feats(1, pts)
On Error Resume Next
'pnt(feats(0, pts - 4), feats(1, pts - 4)) = &HFFFFFF
'pnt(feats(0, pts - 3), feats(1, pts - 3)) = &HFFFFFF

```

```

'pnt(feats(0, pts - 2), feats(1, pts - 2)) = &HFFFFFF
'pnt(feats(0, pts - 1), feats(1, pts - 1)) = &HFFFFFF
'pnt(feats(0, pts), feats(1, pts)) = &HFFFFFF
picture1.Line (feats(0, 0), feats(1, 0))-(feats(0, pts), feats(1, pts)), &HFF&
End If
'MsgBox "Line done"
GoTo vecstdn
Else
pnt(x, y) = 0
GoTo vedn
End If
End If
End Sub

```

```

Sub vectornew () Another interesting idea for vectorization
For y = 0 To 239
  For x = 0 To 319
    tmp(x, y) = 0
  Next
Next
th = &H808080
For y = 5 To 235
  For x = 5 To 315
    ' check corners
    '(-4,-4) (-3,-4) (-4,-3) (-3,-3)
    '(3,-4) (4,-4) (3,-3) (4,-3)
    '(-4,3) (-3,3) (-4,4) (-3,4)
    '(3,3) (4,3) (3,4) (4,4)
    tmpcnt = tmp(x - 4, y - 4) + tmp(x + 4, y - 4) + tmp(x - 4, y + 4) + tmp(x +
4, y + 4)

    If tmpcnt = 0 And (pnt(x - 4, y - 4) < th) And (pnt(x - 3, y - 4) < th) And
(pnt(x - 4, y - 3) < th) And (pnt(x - 3, y - 3) < th) And (pnt(x + 3, y - 4) < th) And
(pnt(x + 4, y - 4) < th) And (pnt(x + 3, y - 3) < th) And (pnt(x + 4, y - 3) < th) And
(pnt(x - 4, y + 3) < th) And (pnt(x - 3, y + 3) < th) And (pnt(x - 4, y + 4) < th) And
(pnt(x - 3, y + 4) < th) And (pnt(x + 3, y + 3) < th) And (pnt(x + 4, y + 3) < th)
And (pnt(x + 3, y + 4) < th) And (pnt(x + 4, y + 4) < th) Then
      'If tmpcnt = 0 And (pnt(x - 4, y - 4) < th) And (pnt(x + 4, y - 4) < th) And
(pnt(x - 4, y + 4) < th) And (pnt(x + 4, y + 4) < th) Then
        a = pnt(x - 2, y - 4) > th
        b = pnt(x - 1, y - 4) > th
        c = pnt(x, y - 4) > th
        d = pnt(x + 1, y - 4) > th
        e = pnt(x + 2, y - 4) > th
        f = pnt(x + 4, y) > th
        g = pnt(x + 4, y - 2) > th
        h = pnt(x + 4, y - 1) > th
        i = pnt(x + 4, y) > th
        j = pnt(x + 4, y + 1) > th
        k = pnt(x + 4, y + 2) > th
        l = pnt(x - 2, y + 4) > th
        m = pnt(x - 1, y + 4) > th
        n = pnt(x, y + 4) > th

```

```

o = pnt(x + 1, y + 4) > th
p = pnt(x + 2, y + 4) > th
q = pnt(x - 4, y + 2) > th
r = pnt(x - 4, y + 1) > th
s = pnt(x - 4, y) > th
t = pnt(x - 4, y - 1) > th
u = pnt(x - 4, y - 2) > th

```

Then If -a - b - c - d - e - f - g - h - i - j - k - l - m - n - o - p - q - r - s - t - u > 3

```

'picture1.Line (x - 4, y - 4)-(x + 4, y + 4), &H0&, BF
picture1.Line (x - 4, y - 4)-(x + 4, y + 4), &HFFFF&, BF
For y1 = -4 To 4
  For x1 = -4 To 4
    tmp(x + x1, y + y1) = 1
  Next
Next
DoEvents

con(0) = u + a
con(1) = a + b '0:not a and not b, -1:a <> b, -2, a and b
con(2) = b + c
con(3) = c + d
con(4) = d + e
con(5) = e + f
con(6) = f + g
con(7) = g + h
con(8) = h + i
con(9) = i + j
con(10) = j + k
con(11) = k + l
con(12) = l + m
con(13) = m + n
con(14) = n + o
con(15) = o + p
con(16) = p + q
con(17) = q + r
con(18) = r + s
con(19) = s + t
con(20) = t + u
con(21) = u + a
con(22) = a + b

```

```

For n = 1 To 21
  oldn = n
  If con(n) Then
    started = n
    no = 0
    If con(n + 1) = -2 Then
      Do Until con(n) = -1
        con(n) = 0
        n = n + 1
      If n = 22 Then n = 1

```

```

    Loop
    con(n) = 0
    ended = n
ElseIf con(n - 1) = -2 Then
    Do Until con(n) = -1
        con(n) = 0
        n = n - 1
        If n = 0 Then n = 21
    Loop
    con(n) = 0
    ended = n
Else
    no = -1
End If
If Not no Then
    con((started + ended) \ 2) = 10
End If
End If
n = oldn
Next
For n = 1 To 21
'MsgBox Str(con(n))
If con(n) = 10 Then
    Select Case n
    Case 1
        picture1.PSet (x - 2, y - 4), &HFF&
    Case 2
        picture1.PSet (x - 1, y - 4), &HFF&
    Case 3
        picture1.PSet (x, y - 4), &HFF&
    Case 4
        picture1.PSet (x + 1, y - 4), &HFF&
    Case 5
        picture1.PSet (x + 2, y - 4), &HFF&
    Case 6
        picture1.PSet (x + 4, y), &HFF&
    Case 7
        picture1.PSet (x + 4, y - 2), &HFF&
    Case 8
        picture1.PSet (x + 4, y - 1), &HFF&
    Case 9
        picture1.PSet (x + 4, y), &HFF&
    Case 10
        picture1.PSet (x + 4, y + 1), &HFF&
    Case 11
        picture1.PSet (x + 4, y + 2), &HFF&
    Case 12
        picture1.PSet (x - 2, y + 4), &HFF&
    Case 13
        picture1.PSet (x - 1, y + 4), &HFF&
    Case 14
        picture1.PSet (x, y + 4), &HFF&
    Case 15

```

```
        picture1.PSet (x + 1, y + 4), &HFF&
    Case 16
        picture1.PSet (x + 2, y + 4), &HFF&
    Case 17
        picture1.PSet (x - 4, y + 2), &HFF&
    Case 18
        picture1.PSet (x - 4, y + 1), &HFF&
    Case 19
        picture1.PSet (x - 4, y), &HFF&
    Case 20
        picture1.PSet (x - 4, y - 1), &HFF&
    Case 21
        picture1.PSet (x - 4, y - 2), &HFF&
    End Select
End If
Next
End If
End If
Next
Next
End Sub
```

That's all in the demo program!