Chapter 5
Sample Implementations of Vision-Based Mobile Robot Algorithms

Abstract. This chapter presents a detailed, step-by-step demonstration of how vision-based navigation modules can be actually implemented in real life, under 32-bit Windows environment. These lessons start with a simple development of capturing image frames from a running video and then gradually proceeds to more complex tasks of incorporating image processing capabilities e.g. filtering techniques, contrast enhancement, adaptive thresholding etc. Then the lessons demonstrate how to extract path for the robot from such images and how a rule-based approach can be utilized to determine left and right wheel speed settings of a differential drive system.

5.1 Introduction

In this chapter Visual Basic based software programming is presented in a step-by-step fashion. Ten lessons are developed for PC based vision-based navigation programming. Low-cost webcam is used for capturing streaming video.

Visual Basic version 6 (VB6) [1-2] is used for windows based programming.

The first lesson 'Lesson 1' demonstrates how to capture image frames from streaming video from a low-cost webcam and examine pixel (picture element) values with the help of mouse pointer. RGB (Red-Green-Blue) to gray-scale conversion is also done in a pixel-by-pixel manner. A 'Format' menu is provided for selecting the image frame size to 160x120. Windows 32-bit API (Application Programming Interface) calls [3] are adopted for faster processing.

The second lesson 'Lesson 2' demonstrates how to process captured image frames from streaming video. Options are provided for RGB to gray-scale conversion and subsequent low-pass filtering [4].

The third lesson 'Lesson 3' shows the method of contrast enhancement by histogram stretching technique [4] under poor lighting conditions.

The fourth lesson 'Lesson 4' introduces geometric-mean filter [4] to smooth and suppress image detail to simplify the extraction of required white path for navigation.

The fifth lesson 'Lesson 5' applies an adaptive threshold operation to extract white path under varying illumination conditions. A selectable reference pixel determines the centre of path to be extracted.

A. Chatterjee et al.: Vision Based Autonomous Robot Navigation, SCI 455, pp. 101–142.
springerlink.com

The sixth lesson 'Lesson 6' introduces a cleaning operation to remove unwanted objects detected during threshold operation.

The next lesson 'Lesson 7' introduces an option for selection of path color white or black. For black path color option, the gray-scale image frame is first converted to negative image, so that black objects become white and then processed as usual as discussed in 'Lesson 6'.

The eighth lesson 'Lesson 8' is targeted for white or black path finding for navigation with a fixed reference pixel.

The next lesson 'Lesson 9' introduces a rule-based approach to determine left and right wheel speed settings of a differential drive system for navigation. Pictorial representation of navigation direction is done with appropriate image file.

Finally in the last lesson 'Lesson 10' sound output is added to draw attention during navigation.

Source codes are available for Visual Basic version 6 and Visual Basic dot net version 2010 compiler from 'http://extras.springer.com'.

Executable codes are also provided for testing the performance of programs when compilers are not available with the reader. Only run-time executables are needed which are freely available from Microsoft.

5.2 Lesson 1

Objective: To develop a VB6 program to capture webcam streaming video.

Following steps summarize the program development.

1. All necessary Application Programming Interface (API) calls are declared in 'Webcam1.bas' module. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to preview streaming video at 30 frames per second and 'Picture2' to capture image from streaming video as clipboard data at a regular interval of 10mS with the help of 'Timer1' control.
4. Two command buttons, namely, 'Capture' and 'Close' are added under 'Form1' to control image capturing process. The command button names are 'cmdCapture' and 'cmdClose' respectively.
5. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
6. Any captured pixel may be examined with the mouse pointer over 'picture2' image. The mouse cursor is changed to 'cross' to facilitate pixel examination.
7. Pixel color is obtained through the 'GetPixel' API call.
8. Red (R), Green (G) and Blue (B) vales are obtained from 'Color' by calling three functions 'GetRed', 'GetGreen' and 'GetBlue' functions as follows: GetRed = Color And 255, GetGreen = (Color And 65280) \ 256 and GetBlue = (Color And 16711680) \ 65535.
9. Three text boxes, namely, 'Text1', 'Text2' and 'Text3' are added to examine 8-bit Red (R), Green (G) and Blue (B) values of the selected pixel.

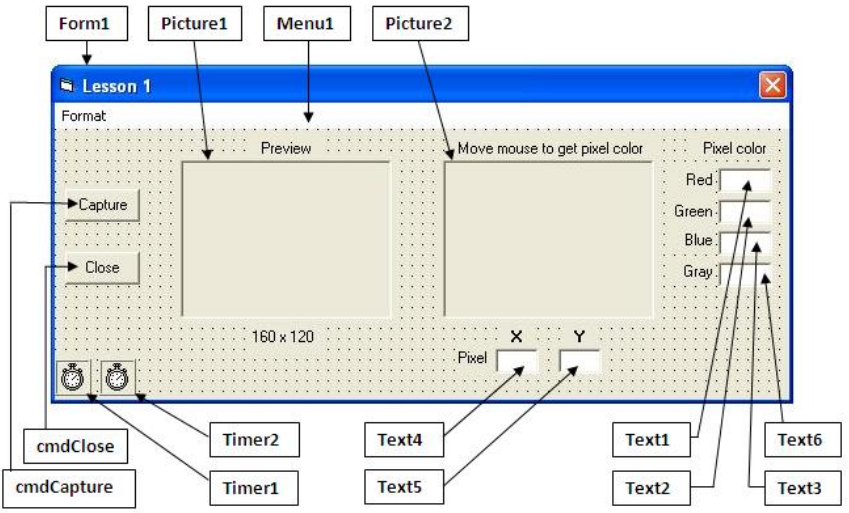
10. Two text boxes, namely, ‘Text4’ and ‘Text5’, are incorporated to monitor ‘X’ and ‘Y’ coordinates of the selected pixel.
11. A text box ‘Text6’ is added to view 8-bit gray value of the selected pixel from its RGB values according to the formula: gray = 0.2125 * red + 0.7154 * green + 0.0721 * blue.
12. A second timer ‘Timer2’ control is added to remove textbox data within 10mS when the mouse pointer is not positioned over ‘Picture2’ picture box.

Following text shows the listing of ‘Webcam1.bas’ module.

```
Global Const WS_CHILD As Long = &H40000000
Global Const WS_VISIBLE As Long = &H10000000
Global Const WM_USER = 1024
Global Const WM_CAP_DRIVER_CONNECT = WM_USER + 10
Global Const WM_CAP_SET_PREVIEW = WM_USER + 50
Global Const WM_CAP_SET_PREVIEWRATE = WM_USER + 52
Global Const WM_CAP_DRIVER_DISCONNECT As Long = WM_USER + 11
Global Const WM_CAP_DLG_VIDEOFORMAT As Long = WM_USER + 41
Global Const WM_CAP_GET_FRAME As Long = 1084
Global Const WM_CAP_COPY As Long = 1054
Global Const WM_CAP_SET_SCALE As Integer = WM_USER + 53
Global Const SWP_NOMOVE As Integer = 2
Global Const SWP_NOZORDER As Integer = 4
Global Const HWND_BOTTOM As Integer = 1

Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
     (ByVal hwnd As Long, ByVal wMsg As Long, ByVal wParam As _
      Long, ByVal lParam As Long) As Long
Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias _
    "capCreateCaptureWindowA" (ByVal a As String, ByVal b As Long, _
    ByVal c As Integer, ByVal d As Integer, ByVal e As Integer, _
    ByVal f As Integer, ByVal g As Long, ByVal h As Integer) As Long
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _
    ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _
    ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, _
    ByVal x As Long, ByVal y As Long) As Long
```

Following figure shows the 'Form1' layout.



Following text shows the listing of 'Form1' code.

```
Dim hwdc As Long
Dim startcap As Boolean
Dim mflag As Boolean

Private Sub cmdCapture_Click()
    hwdc = capCreateCaptureWindow("Webcam Vision System", WS_CHILD _
Or WS_VISIBLE, 0, 0, 160, 120, Picture1.hwnd, 0)
  If (hwdc <> 0) Then
      Clipboard.Clear
      If SendMessage(hwdc, WM_CAP_DRIVER_CONNECT, 0, 0) Then
         SendMessage hwdc, WM_CAP_SET_SCALE, True, 0
         SendMessage hwdc, WM_CAP_SET_PREVIEWRATE, 30, 0
         SendMessage hwdc, WM_CAP_SET_PREVIEW, 1, 0
         SetWindowPos hwdc, HWND_BOTTOM, 0, 0, 160, 120, SWP_NOMOVE _
         Or SWP_NOZORDER
         startcap = True
         cmdCapture.Enabled = False
         cmdClose.Enabled = True
         Timer1.Enabled = True
         Menu1.Enabled = True
         Picture2.Visible = True
         Label1.Visible = True
         Label2.Visible = True
         Label3.Visible = True
         Label4.Visible = True
         Label5.Visible = True
         Label6.Visible = True
```

```
            Label7.Visible = True
            Label9.Visible = True
            Label11.Visible = True
            Text1.Visible = True
            Text2.Visible = True
            Text3.Visible = True
            Text4.Visible = True
            Text5.Visible = True
            Text6.Visible = True
        Else
            MsgBox ("No Webcam found!")
            startcap = False
        End If
    End If
End Sub

Private Sub cmdClose_Click()
    If startcap = True Then
        SendMessage hwdc, WM_CAP_DRIVER_DISCONNECT, 0, 0
        startcap = False
        cmdCapture.Enabled = True
        cmdClose.Enabled = False
        Timer1.Enabled = False
        Menu1.Enabled = False
        Picture2.Visible = False
        Label1.Visible = False
        Label2.Visible = False
        Label3.Visible = False
        Label4.Visible = False
        Label5.Visible = False
        Label6.Visible = False
        Label7.Visible = False
        Label9.Visible = False
        Label11.Visible = False
        Text1.Visible = False
        Text2.Visible = False
        Text3.Visible = False
        Text4.Visible = False
        Text5.Visible = False
        Text6.Visible = False
    End If
End Sub
```

```
Private Sub Form_Load()
    If App.PrevInstance = True Then End      ' multiple instances are not allowed
    cmdCapture.Enabled = True
    cmdClose.Enabled = False
    Picture1.AutoSize = True
    Picture2.AutoSize = True
    Timer1.Interval = 10
    Timer2.Interval = 10
    Menu1.Enabled = False
    mflag = False
    Picture2.Visible = False
    Picture2.MousePointer = 2                ' cross cursor
    Label1.Visible = False
    Label2.Visible = False
    Label3.Visible = False
    Label4.Visible = False
    Label5.Visible = False
    Label6.Visible = False
    Label7.Visible = False
    Label9.Visible = False
    Label11.Visible = False
    Text1.Visible = False
    Text2.Visible = False
    Text3.Visible = False
    Text4.Visible = False
    Text5.Visible = False
    Text6.Visible = False
End Sub

Private Function GetRed(ByVal Color As Long)
   GetRed = Color And 255
End Function

Private Function GetGreen(ByVal Color As Long)
   GetGreen = (Color And 65280) \ 256
End Function

Private Function GetBlue(ByVal Color As Long)
   GetBlue = (Color And 16711680) \ 65535
End Function

Private Sub Form_MouseMove(Button As Integer, Shift As Integer, _
    x As Single, y As Single)
    mflag = False                ' mouse pointer in form but not in picture box
End Sub
Private Sub Menu1_Click()
```

```
    If startcap = True Then
       SendMessage hwdc, WM_CAP_DLG_VIDEOFORMAT, 0, 0
    End If
End Sub

Private Sub Picture2_MouseMove(Button As Integer, Shift As Integer, _
    x As Single, y As Single)
    Dim Color As Long
    Dim red As Byte
    Dim blue As Byte
    Dim green As Byte
    Dim gray As Byte
    Dim xp As Long
    Dim yp As Long

    xp = x / Screen.TwipsPerPixelX
    yp = y / Screen.TwipsPerPixelY
    Color = GetPixel(Picture2.hdc, xp, yp)
    red = GetRed(Color)
    green = GetGreen(Color)
    blue = GetBlue(Color)
    gray = 0.2125 * red + 0.7154 * green + 0.0721 * blue
    Text1.Text = red
    Text2.Text = green
    Text3.Text = blue
    Text4.Text = xp
    Text5.Text = yp
    Text6.Text = gray
    mflag = True                          ' mouse pointer in picture box
End Sub

Private Sub Timer1_Timer()
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture2.Picture = Clipboard.GetData
    SendMessage hwdc, WM_CAP_SET_PREVIEW, 1, 0
End Sub

Private Sub Timer2_Timer()
    If mflag = False Then                 ' no mouse pointer in picture box
       Text1.Text = ""
       Text2.Text = ""
       Text3.Text = ""
       Text4.Text = ""
```

```
        Text5.Text = ""
        Text6.Text = ""
    End If
End Sub
```

To execute the program the capture button has to be pressed. If any webcam is available then preview is available in picture box 'Picture1'. If the size of the captured image does not fit in the picture box 'Picture2' then the image size has to be changed to 160x120 by activating the 'Format' menu.

If no webcam is available then a message box will appear with a message "No webcam found!"

5.3 Lesson 2

Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image and subsequent low-pass image filtering.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam2.bas' module. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
5. From 'Picture1' image pixel data information is obtained through 'GetObject' API call.
6. Pixel array 'Pbytes(c, x, y)', an 8-bit array, is obtained through 'GetBitmapBits' API call under 'Timer1' control. Each element of 'Pbytes' contains 8-bit RGB color information of each pixel at 'x' and 'y' image co-ordinate. 'c' stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. Pixel array is processed according to option controls 'Option1' or 'Option2'.
8. If 'Option1' is selected then pixel array is processed as gray scale image with the help of procedure 'Gray' and displayed in picture box 'Picture2' through 'SetBitmapBits' API call.
9. If 'Option2' is selected then pixel array is processed first to gray scale image as in step 8 and then low-pass filtered with the help of procedure 'Lowpass' and then displayed in 'Picture2'.

Option1 Option2

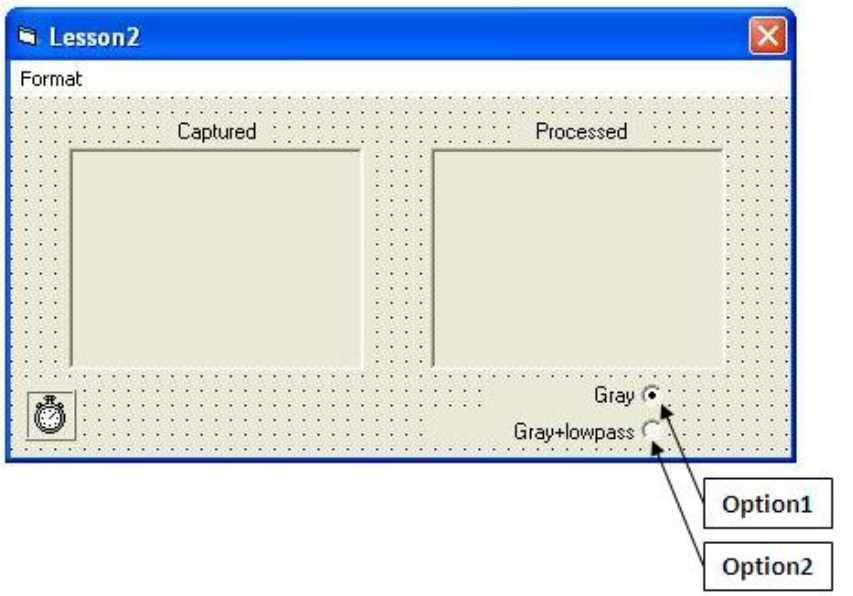
Gray Gray + low-pass

Following text shows the listing of ‘Webcam2.bas’ module.

```
Global Const WS_CHILD As Long = &H40000000
Global Const WS_VISIBLE As Long = &H10000000
Global Const WM_USER = 1024
Global Const WM_CAP_DRIVER_CONNECT = WM_USER + 10
Global Const WM_CAP_SET_PREVIEW = WM_USER + 50
Global Const WM_CAP_SET_PREVIEWRATE = WM_USER + 52
Global Const WM_CAP_DRIVER_DISCONNECT As Long = WM_USER + 11
Global Const WM_CAP_DLG_VIDEOFORMAT As Long = WM_USER + 41
Global Const WM_CAP_GET_FRAME As Long = 1084
Global Const WM_CAP_COPY As Long = 1054
Global Const WM_CAP_SET_SCALE As Integer = WM_USER + 53
Global Const SWP_NOMOVE As Integer = 2
Global Const SWP_NOZORDER As Integer = 4
Global Const HWND_BOTTOM As Integer = 1

Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd _
    As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As_
    Long) As Long Declare Function capCreateCaptureWindow Lib _
    "avicap32.dll" Alias "capCreateCaptureWindowA" (ByVal nWindowName _
    As String, ByVal nStyle As Long, ByVal nx As Integer, ByVal ny As Integer, _
    ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal nHwnd As Long, _
    ByVal nId As Integer) As Long
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _
    ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _
    ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
Declare Function GetObject Lib "gdi32" Alias "GetObjectA" (ByVal hObject _
    As Long, ByVal nCount As Long, lpObject As Any) As Long
Declare Function GetBitmapBits Lib "gdi32" (ByVal hBitmap As Long, _
    ByVal dwCount As Long, lpBits As Any) As Long
Declare Function SetBitmapBits Lib "gdi32" (ByVal hBitmap As Long, _
    ByVal dwCount As Long, lpBits As Any) As Long
```

Following figure shows the 'Form1' layout.



Following text shows the listing of 'Form1' code.

```
Dim hwdc As Long
Dim startcap As Boolean
Private Type Bitmap
    bmType As Long
    bmWidth As Long
    bmHeight As Long
    bmWidthBytes As Long
    bmPlanes As Integer
    bmBitsPixel As Integer
    bmBits As Long
End Type
Dim Pbytes() As Byte, Pinfo As Bitmap
Dim x As Long, y As Long

Private Sub Form_Load()
    If App.PrevInstance = True Then End
    Picture1.AutoSize = True
    Picture2.AutoSize = True
    Picture1.ScaleMode = vbPixels
    Picture2.ScaleMode = vbPixels
    Timer1.Interval = 10

  hwdc = capCreateCaptureWindow("Webcam Vision System", WS_CHILD _
Or WS_VISIBLE, 0, 0, 160, 120, Picture1.hwnd, 0)
```

```
    If (hwdc <> 0) Then
      Clipboard.Clear
      If SendMessage(hwdc, WM_CAP_DRIVER_CONNECT, 0, 0) Then
       SendMessage hwdc, WM_CAP_SET_SCALE, 1, 0
       SendMessage hwdc, WM_CAP_SET_PREVIEWRATE, 30, 0
       SendMessage hwdc, WM_CAP_SET_PREVIEW, 1, 0
       SetWindowPos hwdc, HWND_BOTTOM, 0, 0, 160, 120, _
       SWP_NOMOVE Or SWP_NOZORDER
       SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
       SendMessage hwdc, WM_CAP_COPY, 0, 0
       Picture1.Picture = Clipboard.GetData
       GetObject Picture1.Picture, Len(Pinfo), Pinfo
       ReDim Pbytes(0 To (Pinfo.bmBitsPixel \ 8) - 1, 0 To Pinfo.bmWidth - 1, _
       0 To Pinfo.bmHeight - 1)
       Picture2.height = Picture1.height
       Picture2.width = Picture1.width
       Timer1.Enabled = True
       startcap = True
    Else
      MsgBox "No Webcam found!", OK, ""
      startcap = False
      Unload Me
    End If
  Else
    Unload Me
  End If
End Sub

Private Sub Gray(width As Long, height As Long)
    Dim G As Byte
    For x = 0 To width - 1
      For y = 0 To height - 1
        G = 0.2125 * CDbl(Pbytes(2, x, y)) + 0.7154 * CDbl(Pbytes(1, x, y)) + _
            0.0721 * CDbl(Pbytes(0, x, y))
          Pbytes(2, x, y) = G                'Red
          Pbytes(1, x, y) = G                'Green
          Pbytes(0, x, y) = G                'Blue
       Next y
    Next x
End Sub

Private Sub Lowpass(width As Long, height As Long)
    Dim R As Long
    Dim c, d, e, f As Long
    For x = 0 To width - 1
      For y = 0 To height - 1
```

```
            c = x - 1
            d = x + 1
            e = y - 1
            f = y + 1
            If c < 0 Then c = width - 1
            If d = width Then d = 0
            If e < 0 Then e = height - 1
            If f = height Then f = 0
            R = Pbytes(2, x, e)
            R = R + CLng(Pbytes(2, c, y))
            R = R + 2 * CLng(Pbytes(2, x, y))
            R = R + CLng(Pbytes(2, d, y))
            R = R + CLng(Pbytes(2, x, f))
            R = R / 6                          '3x3 low pass
            Pbytes(2, x, y) = R
            Pbytes(1, x, y) = R
            Pbytes(0, x, y) = R
         Next y
      Next x
End Sub

Private Sub Form_Terminate()
      If startcap = True Then
         SendMessage hwdc, WM_CAP_DRIVER_DISCONNECT, 0, 0
         startcap = False
         Timer1.Enabled = False
      End If
End Sub

Private Sub Form_Unload(Cancel As Integer)
      If startcap = True Then
         SendMessage hwdc, WM_CAP_DRIVER_DISCONNECT, 0, 0
         startcap = False
         Timer1.Enabled = False
      End If
End Sub

Private Sub Menu_Click()
      If startcap = True Then
         SendMessage hwdc, WM_CAP_DLG_VIDEOFORMAT, 0, 0
      End If
End Sub

Private Sub Timer1_Timer()
      Timer1.Enabled = False
      SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
      SendMessage hwdc, WM_CAP_COPY, 0, 0
```

```
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
       Pbytes(0, 0, 0)
    If Option1.Value = True Then Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    If Option2.Value = True Then
       Gray Picture1.ScaleWidth, Picture1.ScaleHeight
       Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
      Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Timer1.Enabled = True
End Sub
```

Low-pass filtering is performed with a 2-D FIR filer mask of size 3x3 as stated below:

$$\frac{1}{6}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Circular 2-D convolution is performed with the above mask to preserve the image size before and after filtering with minimum amount of distortion.

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.4 Lesson 3

Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image, low-pass image filtering and contrast enhancement.

Following steps summarize the program development.

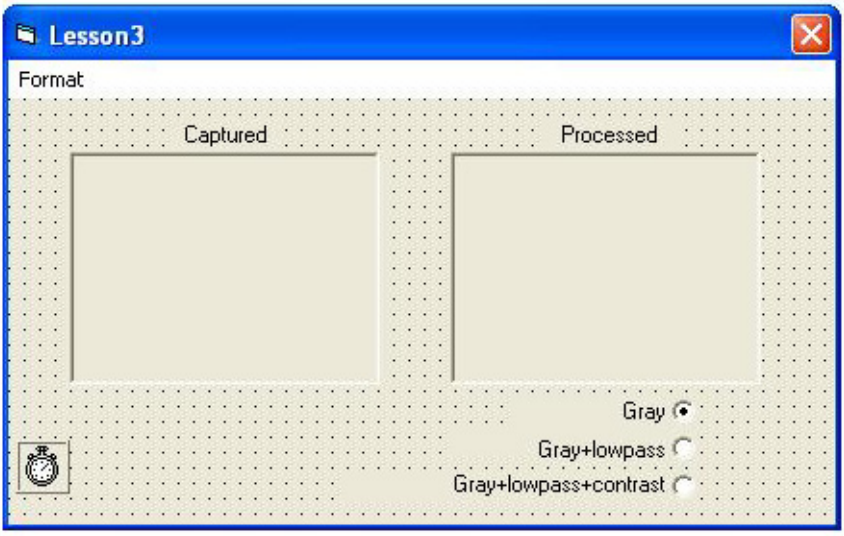
1. All necessary API calls are declared in 'Webcam3.bas' module, same as 'Webcam2.bas', as mentioned in Lesson 2. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.

5. From ‘Picture1’ image pixel data information is obtained through ‘GetObject’ API call.
6. Pixel array ‘Pbytes(c, x, y)’, an 8-bit array, is obtained through ‘GetBitmapBits’ API call under ‘Timer1’ control. Each element of ‘Pbytes’ contains 8-bit RGB color information of each pixel at ‘x’ and ‘y’ image co-ordinate. ‘c’ stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. Pixel array is processed according to option controls ‘Option1’, ‘Option2’ or ‘Option3’.
8. If ‘Option1’ is selected then pixel array is processed as gray scale image with the help of procedure ‘Gray’ and displayed in picture box ‘Picture2’ through ‘SetBitmapBits’ API call.
9. If ‘Option2’ is selected then pixel array is processed first to gray scale image as in step 8 and then low-pass filtered with the help of procedure ‘Lowpass’ and then displayed in ‘Picture2’.
10. If ‘Option3’ is selected then array is low-pass filtered as in step 9 and then processed for contrast enhancement using histogram stretching technique with the help of procedure ‘Contrast’ and then displayed in ‘Picture2’.

Option1 Option2 Option3

Gray Gray + low-pass Gray + low-pass + contrast

Following figure shows the ‘Form1’ layout.



Following text shows the listing of 'Contrast' and 'Timer1' procedure code. For rest of the code refer to Lesson 2.

```
Private Sub Contrast(width As Long, height As Long)
    Dim R As Long                              'histogram stretching
    Dim pmax, pmin As Long
    pmax = 0
    pmin = 255
    For x = 0 To width - 1
      For y = 0 To height - 1
        If pmax <= CLng(Pbytes(2, x, y)) Then pmax = Pbytes(2, x, y)
        If pmin >= CLng(Pbytes(2, x, y)) Then pmin = Pbytes(2, x, y)
      Next y
    Next x
    For x = 0 To width - 1
      For y = 0 To height - 1
        R = Pbytes(2, x, y)
        If pmax > pmin Then R = (((R - pmin) * 255) / (pmax - pmin)) + pmin / 4
        If R < 0 Then R = 0
        If R > 255 Then R = 255
        Pbytes(2, x, y) = R
        Pbytes(1, x, y) = R
        Pbytes(0, x, y) = R
      Next y
    Next x
End Sub

Private Sub Timer1_Timer()
  Timer1.Enabled = False
  SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
  SendMessage hwdc, WM_CAP_COPY, 0, 0
  Picture1.Picture = Clipboard.GetData
  GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
    Pbytes(0, 0, 0)
  If Option1.Value = True Then Gray Picture1.ScaleWidth, Picture1.ScaleHeight
  If Option2.Value = True Then
    Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
  End If
  If Option3.Value = True Then
    Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
  End If
```

```
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
       Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.5 Lesson 4

Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image, low-pass filtering, contrast enhancement and geometric-mean filtering.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam4.bas' module, same as 'Webcam3.bas', as mentioned in Lesson 3. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
5. From 'Picture1' image pixel data information is obtained through 'GetObject' API call.
6. Pixel array 'Pbytes(c, x, y)', an 8-bit array, is obtained through 'GetBitmapBits' API call under 'Timer1' control. Each element of 'Pbytes' contains 8-bit RGB color information of each pixel at 'x' and 'y' image co-ordinate. 'c' stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. Pixel array is processed according to option controls 'Option1', 'Option2, 'Option3' or 'Option4'.
8. If 'Option1' is selected then pixel array is processed as gray scale image with the help of procedure 'Gray' and displayed in picture box 'Picture2' through 'SetBitmapBits' API call.
9. If 'Option2' is selected then pixel array is processed first to gray scale image as in step 8 and then low-pass filtered with the help of procedure 'Lowpass' and then displayed in 'Picture2'.

10. If ‘Option3’ is selected then array is low-pass filtered as in step 9 and then processed for contrast enhancement using histogram stretching technique with the help of procedure ‘Contrast’ and then displayed in ‘Picture2’.
11. If ‘Option4’ is selected then array is processed for contrast enhancement as in step 10 and then processed for geometric-mean filtering with the help of procedure ‘Geometricmean’ and then displayed in ‘Picture2’. Options are provided for increasing the number of cascaded Geometric-mean filters and the size of mask for each filter.

Option1

Gray

Option2

Gray + low-pass

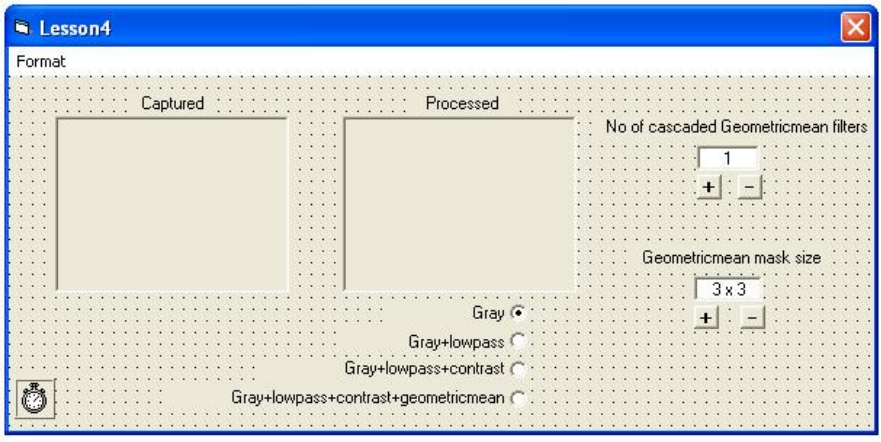
Option3

Gray + low-pass + contrast

Option4

Gray + low-pass + contrast + geometric-mean

Following figure shows the ‘Form1’ layout.



Following text shows the listing of 'Geometricmean' and 'Timer1' procedure code. For rest of the code refer to Lesson 3.

```
Private Sub Geometricmean(width As Long, height As Long, Size As Long)
    Dim R, S As Long
    Dim i, j As Long
    Dim c, d As Long
    Dim w1, h1 As Long
    If Size < 3 Then Size = 3
    If Size > 7 Then Size = 7
    If (Size And 1) = 0 Then Size = Size + 1        'even to odd conversion
    S = Size * Size
    w1 = width - 1
    h1 = height - 1
    For x = 0 To w1
      For y = 0 To h1
        R = 1
        For i = 0 To Size - 1
          For j = 0 To Size - 1
            c = x + i - ((Size - 1) / 2)
            If c < 0 Then c = width + c
            If c > w1 Then c = c - w1
            d = y + j - ((Size - 1) / 2)
            If d < 0 Then d = height + d
            If d > h1 Then d = d - h1
            R = R * CLng(Pbytes(2, c, d))
          Next j
        Next i
        R = R ^ (1# / S)
        If R > 255 Then R = 255
        Pbytes(2, x, y) = R
        Pbytes(1, x, y) = R
        Pbytes(0, x, y) = R
      Next y
    Next x
End Sub

Private Sub Timer1_Timer()
  Timer1.Enabled = False
  SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
  SendMessage hwdc, WM_CAP_COPY, 0, 0
  Picture1.Picture = Clipboard.GetData
  GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
    Pbytes(0, 0, 0)
  If Option1.Value = True Then Gray Picture1.ScaleWidth, Picture1.ScaleHeight
```

```
    If Option2.Value = True Then
       Gray Picture1.ScaleWidth, Picture1.ScaleHeight
       Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option3.Value = True Then
       Gray Picture1.ScaleWidth, Picture1.ScaleHeight
       Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
       Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option4.Value = True Then
       Gray Picture1.ScaleWidth, Picture1.ScaleHeight
       Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
       Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
       For i = 1 To Val(Text4.Text)
          Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
       Next i
    End If
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
       Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.6 Lesson 5

Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image, low-pass filtering, contrast enhancement, geometric-mean filtering and an adaptive threshold operation to extract white path from the captured image under varying illumination conditions.

Following steps summarize the program development.

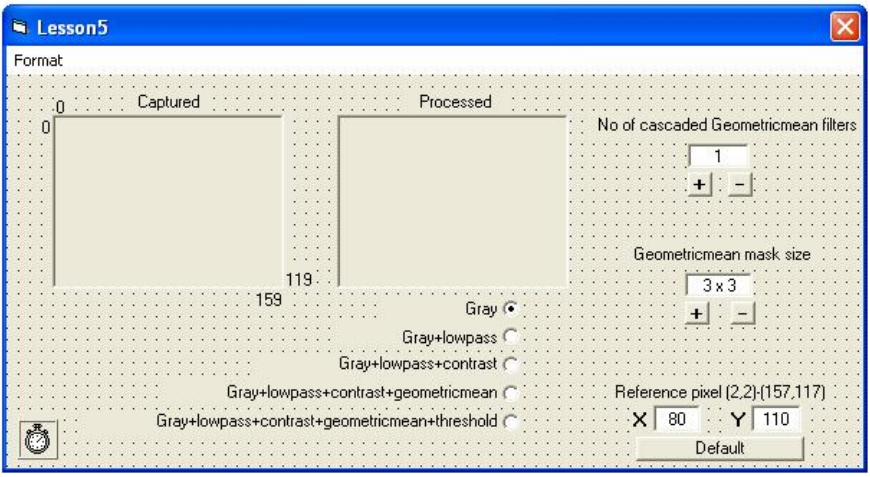
1. All necessary API calls are declared in 'Webcam5.bas' module, same as 'Webcam4.bas', as mentioned in Lesson 4. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.

4. A menu item ‘Format’ is added in ‘Form1’ to set the image size to 160x120 pixels.
5. From ‘Picture1’ image pixel data information is obtained through ‘GetObject’ API call.
6. Pixel array ‘Pbytes(c, x, y)’, an 8-bit array, is obtained through ‘GetBitmapBits’ API call under ‘Timer1’ control. Each element of ‘Pbytes’ contains 8-bit RGB color information of each pixel at ‘x’ and ‘y’ image co-ordinate. ‘c’ stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. Pixel array is processed according to option controls ‘Option1’, ‘Option2, ‘Option3’, ‘Option4’ or ‘Option5’.
8. If ‘Option1’ is selected then pixel array is processed as gray scale image with the help of procedure ‘Gray’ and displayed in picture box ‘Picture2’ through ‘SetBitmapBits’ API call.
9. If ‘Option2’ is selected then pixel array is processed first to gray scale image as in step 8 and then low-pass filtered with the help of procedure ‘Lowpass’ and then displayed in ‘Picture2’.
10. If ‘Option3’ is selected then array is low-pass filtered as in step 9 and then processed for contrast enhancement using histogram stretching technique with the help of procedure ‘Contrast’ and then displayed in ‘Picture2’.
11. If ‘Option4’ is selected then array is processed for contrast enhancement as in step 10 and then processed for geometric-mean filtering with the help of procedure ‘Geometricmean’ and then displayed in ‘Picture2’. Options are provided for increasing the number of cascaded Geometric-mean filters and the size of mask for each filter.
12. If ‘Option5’ is selected then an adaptive threshold operation is performed with the help of the procedure ‘Adaptive Threshold’ and then displayed in ‘Picture2’. First the white line width around a reference pixel [at the nominal position (80,110)] is determined with the procedure ‘WhiteLineWidth’. If both left and right path width around the reference pixel are found be less than ‘MIN_PATH_WIDTH’ value then a parameter ‘delta’ is adjusted to increase the path width by decreasing the threshold value within a range ‘delta_max’. Then the procedure ‘Threshold’ computes new image and the above sequence of operations repeats until a valid white path is obtained.

Option5

Gray + low-pass + contrast + geometric-mean + threshold

Following figure shows the ‘Form1’ layout.



Following text shows the listing of ‘AdaptiveThreshold’, ‘WhiteLineWidth’, ‘Threshold’ and ‘Timer1’ procedure code. For rest of the code refer to Lesson 4.

```
Private Sub AdaptiveThreshold(width As Long, xr As Long, yr As Long)
    Dim i As Integer
    WhiteLineWidth width, xr, yr
    If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
        MIN_PATH_WIDTH Then
        delta = delta + 0.2
        If delta > delta_max Then
            delta = delta_max
        Else
            GoTo atc
        End If
        If delta < 1# Then delta = 1#
    End If
    delta = delta - 0.5
atc:
    i = Pbytes(2, xr, yr)
    If i > (255 - (2 * delta)) Then
        If i > (255 - delta) Then i = (255 - delta)
        Threshold Picture1.ScaleWidth, Picture1.ScaleHeight, i - CInt(delta), _
            i + CInt(delta)
    Else
        Threshold Picture1.ScaleWidth, Picture1.ScaleHeight, 255, 255
    End If
End Sub
```

```
Private Sub WhiteLineWidth(width As Long, xr As Long, yr As Long)
      Dim pcl1, pcl2, pcl3, pcr1, pcr2, pcr3 As Integer
      PixelCountLeft = 0: PixelCountRight = 0
      y = yr
       pcl1 = 0: pcr1 = 0
       For x = xr To 0 Step -1
          If Pbytes(2, x, y) > 250 Then
              pcl1 = pcl1 + 1
          End If
       Next x
       For x = (xr + 1) To (width - 1)
          If Pbytes(2, x, y) > 250 Then
              pcr1 = pcr1 + 1
          End If
       Next x
       y = yr - 1
       pcl2 = 0: pcr2 = 0
       For x = xr To 0 Step -1
          If Pbytes(2, x, y) > 250 Then
              pcl2 = pcl2 + 1
          End If
       Next x
       For x = (xr + 1) To (width - 1)
          If Pbytes(2, x, y) > 250 Then
              pcr2 = pcr2 + 1
          End If
       Next x
       y = yr + 1
       pcl3 = 0: pcr3 = 0
       For x = xr To 0 Step -1
          If Pbytes(2, x, y) > 250 Then
              pcl3 = pcl3 + 1
          End If
       Next x
       For x = (xr + 1) To (width - 1)
          If Pbytes(2, x, y) > 250 Then
              pcr3 = pcr3 + 1
          End If
       Next x

       PixelCountLeft = (pcl1 + pcl2 + pcl3) / 3
       PixelCountRight = (pcr1 + pcr2 + pcr3) / 3
   End Sub
```

```
Private Sub Threshold(width As Long, height As Long, lv As Long, hv As Long)
    Dim R As Long
    For x = 0 To width - 1
      For y = 0 To height - 1
        R = Pbytes(2, x, y)
        If R < lv Then R = 0
        If R >= hv Then R = 255
        Pbytes(2, x, y) = R
        Pbytes(1, x, y) = R
        Pbytes(0, x, y) = R
      Next y
    Next x
End Sub

Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
      Pbytes(0, 0, 0)
    If Option1.Value = True Then Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    If Option2.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option3.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option4.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
      For i = 1 To Val(Text4.Text)
        Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
      Next i
    End If
    If Option5.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
      For i = 1 To Val(Text4.Text)
        Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
     Next i
```

```
        AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
    End If
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
        Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2)-(Val(Text2.Text) + 2, _
        Val(Text3.Text) + 2), RGB(255, 0, 0), B
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.7 Lesson 6

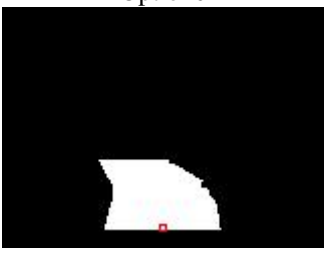
Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image, low-pass filtering, contrast enhancement, geometric-mean filtering, adaptive threshold and a cleaning operation to extract white path and remove unwanted objects from the captured image under varying illumination conditions.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam6.bas' module, same as 'Webcam5.bas', as mentioned in Lesson 5. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
5. From 'Picture1' image pixel data information is obtained through 'GetObject' API call.
6. Pixel array 'Pbytes(c, x, y)', an 8-bit array, is obtained through 'GetBitmapBits' API call under 'Timer1' control. Each element of 'Pbytes' contains 8-bit RGB color information of each pixel at 'x' and 'y' image co-ordinate. 'c' stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. Pixel array is processed according to option controls 'Option1', 'Option2, 'Option3', 'Option4', 'Option5' or 'Option6'.

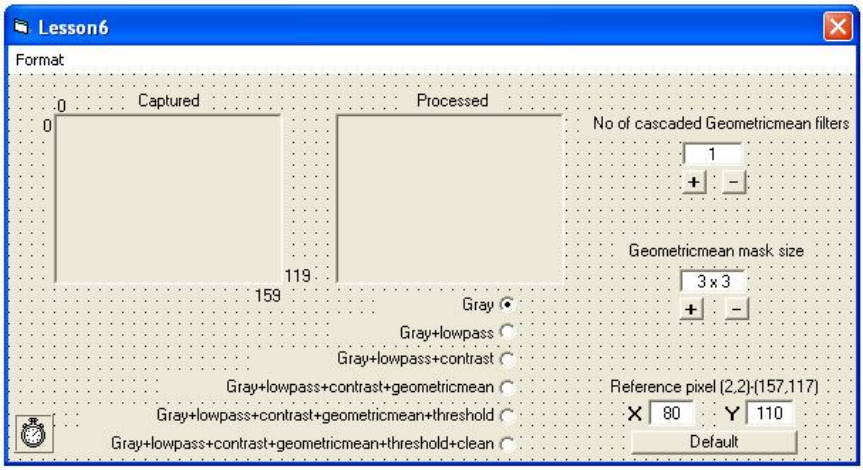
8. If ‘Option1’ is selected then pixel array is processed as gray scale image with the help of procedure ‘Gray’ and displayed in picture box ‘Picture2’ through ‘SetBitmapBits’ API call.
9. If ‘Option2’ is selected then pixel array is processed first to gray scale image as in step 8 and then low-pass filtered with the help of procedure ‘Lowpass’ and then displayed in ‘Picture2’.
10. If ‘Option3’ is selected then array is low-pass filtered as in step 9 and then processed for contrast enhancement using histogram stretching technique with the help of procedure ‘Contrast’ and then displayed in ‘Picture2’.
11. If ‘Option4’ is selected then array is processed for contrast enhancement as in step 10 and then processed for geometric-mean filtering with the help of procedure ‘Geometricmean’ and then displayed in ‘Picture2’. Options are provided for increasing the number of cascaded Geometric-mean filters and the size of mask for each filter.
12. If ‘Option5’ is selected then an adaptive threshold operation is performed with the help of the procedure ‘Adaptive Threshold’ and then displayed in ‘Picture2’. First the white line width around a reference pixel [at the nominal position (80,110)] is determined with the procedure ‘WhiteLineWidth’. If both left and right path width around the reference pixel are found be less than ‘MIN_PATH_WIDTH’ value then a parameter ‘delta’ is adjusted to increase the path width by decreasing the threshold value within a range ‘delta_max’. Then the procedure ‘Threshold’ computes new image and the above sequence of operations repeats until a valid white path is obtained.
13. If ‘Option6’ is selected then an additional cleaning operation is performed to remove unwanted objects with the help of the procedure ‘Clean’ and then displayed in ‘Picture2’.

Option6



Gray + low-pass + contrast + geometric-mean + threshold + clean

Following figure shows the 'Form1' layout.



Following text shows the listing of 'Clean' and 'Timer1' procedure code. For rest of the code refer to Lesson 5.

```
Private Sub Clean(width As Long, height As Long, yr As Long)
        Dim R, xr, xref, xwidth As Long
        Dim PB As Long
        Dim bl_flag As Boolean
        bl_flag = False
        xref = 0
        xwidth = 0
        If PixelCountLeft >= MIN_PATH_WIDTH Or PixelCountRight >= _
           MIN_PATH_WIDTH Then
           For x = 0 To width - 1
              R = Pbytes(2, x, yr)
              If R > 240 Then
                 If xref = 0 Then xref = x
              End If
              If R > 240 And xref > 0 Then xwidth = xwidth + 1
           Next x
           xr = xref + (xwidth / 2)
           For y = height - 1 To (yr + 1) Step -1
              For x = 0 To width - 1
                 Pbytes(2, x, y) = 0
                 Pbytes(1, x, y) = 0
                 Pbytes(0, x, y) = 0
              Next x
           Next y
           For y = yr To 0 Step -1
              For x = xr To 0 Step -1
```

```
            R = Pbytes(2, x, y)
            If bl_flag = True Then GoTo m1
            If R < 240 Then
               PB = x
               If PB = xr Then bl_flag = True
               GoTo m1
            End If
         Next x
m1:
         For x = PB To 0 Step -1
            Pbytes(2, x, y) = 0
            Pbytes(1, x, y) = 0
            Pbytes(0, x, y) = 0
         Next x

         For x = (xr + 1) To width - 1
            R = Pbytes(2, x, y)
            If bl_flag = True Then GoTo m2
            If R < 240 Then
               PB = x
               If PB = (xr + 1) Then bl_flag = True
               GoTo m2
            End If
         Next x
m2:
         For x = PB To width - 1
            Pbytes(2, x, y) = 0
            Pbytes(1, x, y) = 0
            Pbytes(0, x, y) = 0
         Next x

         xref = 0
         xwidth = 0
         For x = 0 To width - 1
            R = Pbytes(2, x, y)
            If R > 240 Then
               If xref = 0 Then xref = x
            End If
            If R > 240 And xref > 0 Then xwidth = xwidth + 1
         Next x
         If xwidth = 0 Then bl_flag = True
         For x = 0 To width - 1
            If bl_flag = True Then
               Pbytes(2, x, y) = 0
               Pbytes(1, x, y) = 0
               Pbytes(0, x, y) = 0
            End If
```

```
            Next x
        Next y
    End If
End Sub

Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
        Pbytes(0, 0, 0)
    If Option1.Value = True Then Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    If Option2.Value = True Then
        Gray Picture1.ScaleWidth, Picture1.ScaleHeight
        Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option3.Value = True Then
        Gray Picture1.ScaleWidth, Picture1.ScaleHeight
        Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
        Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option4.Value = True Then
        Gray Picture1.ScaleWidth, Picture1.ScaleHeight
        Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
        Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
        For i = 1 To Val(Text4.Text)
            Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
        Next i
    End If
    If Option5.Value = True Then
        Gray Picture1.ScaleWidth, Picture1.ScaleHeight
        Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
        Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
        For i = 1 To Val(Text4.Text)
            Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
        Next i
        AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
    End If
    If Option6.Value = True Then
        Gray Picture1.ScaleWidth, Picture1.ScaleHeight
        Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
        Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
        For i = 1 To Val(Text4.Text)
```

```
            Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
        Next i
        AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text),
            Val(Text3.Text)
        Clean Picture1.ScaleWidth, Picture1.ScaleHeight, Val(Text3.Text)
    End If
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
        Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2)-(Val(Text2.Text) _
        + 2, Val(Text3.Text) + 2), RGB(255, 0, 0), B
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.8 Lesson 7

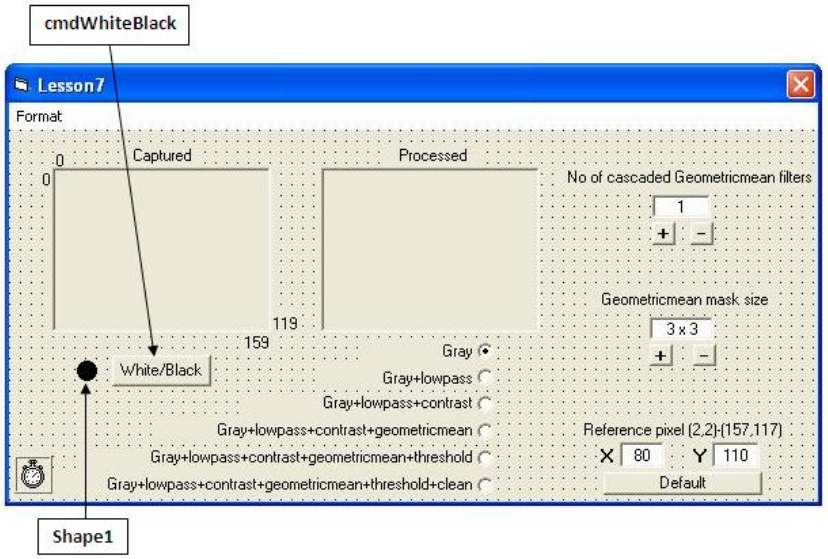
Objective: To develop a VB6 program to capture and process webcam streaming video for conversion to gray scale image, low-pass filtering, contrast enhancement, geometric-mean filtering, adaptive threshold and clean operations along with a selection of white/black path color for vision based navigation.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam7.bas' module, same as 'Webcam6.bas', as mentioned in Lesson 6. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
5. From 'Picture1' image pixel data information is obtained through 'GetObject' API call.
6. Pixel array 'Pbytes(c, x, y)', an 8-bit array, is obtained through 'GetBitmapBits' API call under 'Timer1' control. Each element of 'Pbytes' contains 8-bit RGB color information of each pixel at 'x' and 'y' image co-ordinate. 'c' stands for color; c:2 for red, c:1 for green and c:0 for blue.

7. ‘Shape1’ displays the color of the path (white or black) as selected with the ‘cmdWhiteBlack’ button.
8. Captured image is converted to negative with the help of procedure ‘Negative’ if black path is selected according to step 7. Then this image is processed according to the option selection (‘Option1’ to ‘Option6’) as described in lesson 6.

Following figure shows the ‘Form1’ layout.



Following text shows the listing of ‘cmdWhiteBlack’, ‘Negative’ and ‘Timer1’ procedure code. For rest of the code refer to Lesson 6.

```
Private Sub cmdWhiteBlack_Click()
    If sflag = False Then
       sflag = True
    Else
       sflag = False
    End If
    If sflag = False Then Shape1.FillColor = vbWhite
    If sflag = True Then Shape1.FillColor = vbBlack
End Sub

Private Sub Negative(width As Long, height As Long)
    Dim R As Long
    For x = 0 To width - 1
       For y = 0 To height - 1
          R = 255 - Pbytes(2, x, y)    'Invert
          Pbytes(2, x, y) = R
          Pbytes(1, x, y) = R
          Pbytes(0, x, y) = R
```

```
        Next y
    Next x
End Sub

Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
      Pbytes(0, 0, 0)
    If Option1.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option2.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option3.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    End If
    If Option4.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
      For i = 1 To Val(Text4.Text)
        Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
      Next i
    End If
    If Option5.Value = True Then
      Gray Picture1.ScaleWidth, Picture1.ScaleHeight
      If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
      Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
      Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
      For i = 1 To Val(Text4.Text)
        Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
      Next i
      AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
    End If
```

```
    If Option6.Value = True Then
       Gray Picture1.ScaleWidth, Picture1.ScaleHeight
       If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
       Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
       Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
       For i = 1 To Val(Text4.Text)
          Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
       Next i
       AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), _
           Val(Text3.Text)
       Clean Picture1.ScaleWidth, Picture1.ScaleHeight, Val(Text3.Text)
    End If
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
       Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2) _
      - (Val(Text2.Text) + 2, Val(Text3.Text) + 2), RGB(255, 0, 0), B
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.9 Lesson 8

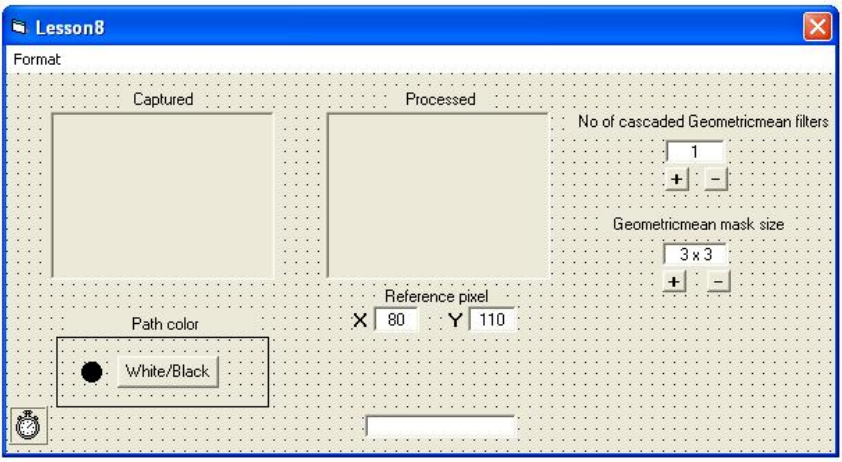
Objective: To develop a VB6 program to capture and process webcam streaming video for vision based navigation along with a selection of white/black path color. Inference is drawn on whether path is available or not.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam8.bas' module, same as 'Webcam7.bas', as mentioned in Lesson 7. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from the captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.

5. From ‘Picture1’ image pixel data information is obtained through ‘GetObject’ API call.
6. Pixel array ‘Pbytes(c, x, y)’, an 8-bit array, is obtained through ‘GetBitmapBits’ API call under ‘Timer1’ control. Each element of ‘Pbytes’ contains 8-bit RGB color information of each pixel at ‘x’ and ‘y’ image co-ordinate. ‘c’ stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. ‘Shape1’ displays the color of the path (white or black) as selected with the ‘cmdWhiteBlack’ button.
8. Captured image is converted to negative with the help of procedure ‘Negative’ if black path is selected according to step 7. Then this image is processed according to the option 6 of Lesson 7.
9. Then white line width around a fixed reference pixel [at position (80,110)] is determined with the procedure ‘WhiteLineWidth’. If both left and right path width around the reference pixel are found be less than ‘MIN_PATH_WIDTH’ value then ‘No path’ inference is drawn, otherwise ‘Path found’ inference is drawn and shown in a text box.

Following figure shows the ‘Form1’ layout.



Following text shows the listing of ‘Timer1’ procedure code. For rest of the code refer to Lesson 7.

```
Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
      Pbytes(0, 0, 0)
```

```
    Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    If sflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
    Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    For i = 1 To Val(Text4.Text)
       Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
    Next i
    AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
    Clean Picture1.ScaleWidth, Picture1.ScaleHeight, Val(Text3.Text)
    WhiteLineWidth Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
    SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
       Pbytes(0, 0, 0)
    Picture2.Refresh
    Picture2.Picture = Picture2.Image
    Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2)-(Val(Text2.Text) _
      + 2, Val(Text3.Text) + 2), RGB(255, 0, 0), B
    If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
       MIN_PATH_WIDTH Then
       Text5.Text = "No path"
    Else
       Text5.Text = "Path found"
    End If
    Timer1.Enabled = True
End Sub
```

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.10 Lesson 9

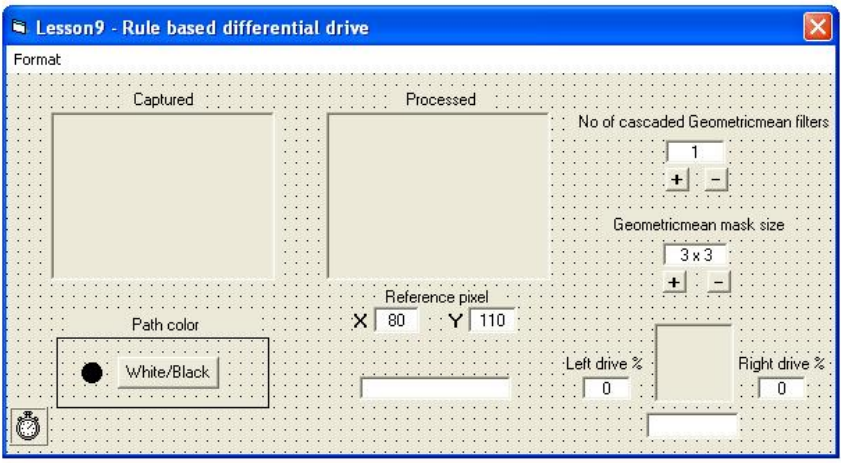
Objective: To develop a VB6 program to capture and process webcam streaming video for vision based navigation along with a selection of white/black path color. Inference is drawn on whether path is available or not. Appropriate rules are applied to determine different navigational directions and speed parameters for differential drive.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam9.bas' module, same as 'Webcam8.bas', as mentioned in Lesson 8. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.

3. Under Form1 two ‘Picture Box’ controls are added, ‘Picture1’ to capture image as clipboard data from streaming video at a regular interval of 10mS and ‘Picture2’ to process image from the captured image at the same rate with the help of ‘Timer1’ control.
4. A menu item ‘Format’ is added in ‘Form1’ to set the image size to 160x120 pixels.
5. From ‘Picture1’ image pixel data information is obtained through ‘GetObject’ API call.
6. Pixel array ‘Pbytes(c, x, y)’, an 8-bit array, is obtained through ‘GetBitmapBits’ API call under ‘Timer1’ control. Each element of ‘Pbytes’ contains 8-bit RGB color information of each pixel at ‘x’ and ‘y’ image co-ordinate. ‘c’ stands for color; c:2 for red, c:1 for green and c:0 for blue.
7. ‘Shape1’ displays the color of the path (white or black) as selected with the ‘cmdWhiteBlack’ button.
8. Captured image is processed according to Lesson 8. If path is found then appropriate navigational direction (‘forward’ or ‘turn-left’ or ‘turn-right’) and the corresponding speed parameters for differential drive are determined with three rules. A picture box shows the direction of navigation.

Following figure shows the ‘Form1’ layout.



Following text shows the listing of ‘Timer1’ procedure code. For rest of the code refer to Lesson 8.

```
Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
```

```
GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
   Pbytes(0, 0, 0)
Gray Picture1.ScaleWidth, Picture1.ScaleHeight
If blkflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
For i = 1 To Val(Text4.Text)
   Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
Next i
AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
Clean Picture1.ScaleWidth, Picture1.ScaleHeight, Val(Text3.Text)
WhiteLineWidth Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
  Pbytes(0, 0, 0)
Picture2.Refresh
Picture2.Picture = Picture2.Image
Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2)-(Val(Text2.Text) _
  + 2, Val(Text3.Text) + 2), RGB(255, 0, 0), B

If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
   MIN_PATH_WIDTH Then
   Text5.Text = "No path"
Else
   Text5.Text = "Path found"
End If
If PixelCountLeft >= MIN_PATH_WIDTH And PixelCountRight < _
   MIN_PATH_WIDTH Then
   Text6.Text = 0: Text7.Text = 50        'turn left
   Text8.Text = "Turn left"
   Picture3.Picture = LoadPicture("turn_left.jpg")
End If
If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight >= _
   MIN_PATH_WIDTH Then
   Text6.Text = 50: Text7.Text = 0        'turn right
   Text8.Text = "Turn right"
   Picture3.Picture = LoadPicture("turn_right.jpg")
End If
If PixelCountLeft >= MIN_PATH_WIDTH And PixelCountRight >= _
   MIN_PATH_WIDTH Then
   Text6.Text = 100: Text7.Text = 100     'forward
   Text8.Text = "Forward"
   Picture3.Picture = LoadPicture("forward.jpg")
End If
```

```
    If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
       MIN_PATH_WIDTH Then
       Text6.Text = 0: Text7.Text = 0      'no path - idle
       Text8.Text = ""
       Picture3.Picture = LoadPicture("blank.jpg")
    End If
    Timer1.Enabled = True
End Sub
```

Following image files are used to indicate direction of navigation.

Forward.jpg

turn_left.jpg

turn_right.jpg

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.11 Lesson 10

Objective: To develop a VB6 program to capture and process webcam streaming video for vision based navigation along with a selection of white/black path color. Inference is drawn on whether path is available or not. Appropriate rules are applied to determine different navigational directions and speed parameters for differential drive. Sound output is added to draw attention.

Following steps summarize the program development.

1. All necessary API calls are declared in 'Webcam10.bas' module. It is necessary to include this module in 'Form1' of the VB6 program.
2. AVICAP32.DLL is used to capture webcam streaming video through proper API call. The webcam video format should be either RGB24 or YUY2.
3. Under Form1 two 'Picture Box' controls are added, 'Picture1' to capture image as clipboard data from streaming video at a regular interval of 10mS and 'Picture2' to process image from the captured image at the same rate with the help of 'Timer1' control.
4. A menu item 'Format' is added in 'Form1' to set the image size to 160x120 pixels.
5. From 'Picture1' image pixel data information is obtained through 'GetObject' API call.
6. Pixel array 'Pbytes(c, x, y)', an 8-bit array, is obtained through 'GetBitmapBits' API call under 'Timer1' control. Each element of 'Pbytes' contains 8-bit RGB color information of each pixel at 'x' and 'y' image co-ordinate. 'c' stands for color; c:2 for red, c:1 for green and c:0 for blue.

7. ‘Shape1’ displays the color of the path (white or black) as selected with the ‘cmdWhiteBlack’ button.
8. Captured image is processed according to Lesson 9. If path is found then appropriate navigational direction (‘forward’ or ‘turn-left’ or ‘turn-right’) and the corresponding speed parameters for differential drive are determined with three rules. A picture box shows the direction of navigation.
9. Sound output is activated through ‘sndPlaySound’ API call with appropriate ‘wave’ file.

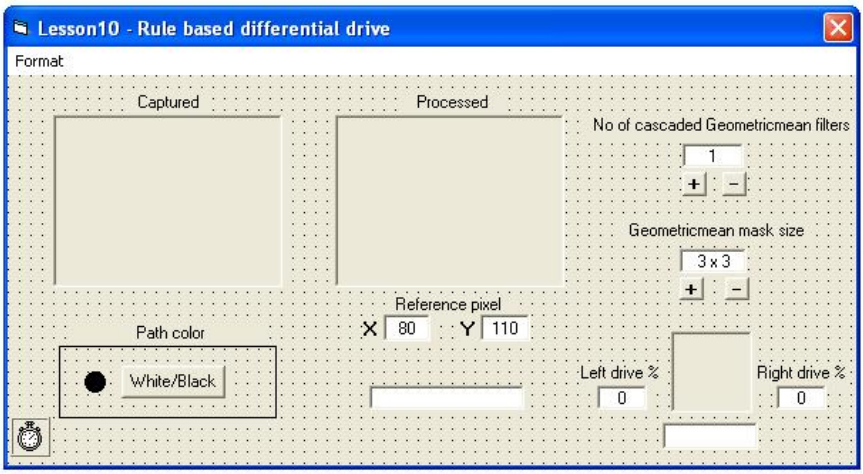
Following text shows the listing of ‘Webcam10.bas’ module.

```
Global Const WS_CHILD As Long = &H40000000
Global Const WS_VISIBLE As Long = &H10000000
Global Const WM_USER = 1024
Global Const WM_CAP_DRIVER_CONNECT = WM_USER + 10
Global Const WM_CAP_SET_PREVIEW = WM_USER + 50
Global Const WM_CAP_SET_PREVIEWRATE = WM_USER + 52
Global Const WM_CAP_DRIVER_DISCONNECT As Long = WM_USER + 11
Global Const WM_CAP_DLG_VIDEOFORMAT As Long = WM_USER + 41
Global Const WM_CAP_DLG_VIDEOCOMPRESSION As Long = _
   WM_USER + 46
Global Const WM_CAP_DLG_VIDEODISPLAY As Long = WM_USER + 43
Global Const WM_CAP_DLG_VIDEOSOURCE As Long = WM_USER + 42
Global Const WM_CAP_GET_FRAME As Long = 1084
Global Const WM_CAP_COPY As Long = 1054
Global Const WM_CAP_SET_SCALE As Integer = WM_USER + 53
Global Const SWP_NOMOVE As Integer = 2
Global Const SWP_NOZORDER As Integer = 4
Global Const HWND_BOTTOM As Integer = 1
Global Const SND_ASYNC = 1
Global Const SND_LOOP = 8
Global Const SND_NODEFAULT = 2
Global Const SND_SYNC = 0
Global Const SND_NOSTOP = 16
Global Const SND_MEMORY = 4

Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hwnd _
   As Long, ByVal wMsg As Long, ByVal wParam As Long, ByVal lParam As Long) _
   As Long Declare Function capCreateCaptureWindow Lib "avicap32.dll" Alias _
   "capCreateCaptureWindowA" (ByVal nWindowName As String, ByVal nStyle _
   As Long, ByVal nx As Integer, ByVal ny As Integer, ByVal nWidth As Integer, _
   ByVal nHeight As Integer, ByVal nHwnd As Long, ByVal nId As Integer) As Long
Declare Function SetWindowPos Lib "user32" (ByVal hwnd As Long, _
   ByVal hWndInsertAfter As Long, ByVal x As Long, ByVal y As Long, _
   ByVal cx As Long, ByVal cy As Long, ByVal wFlags As Long) As Long
```

```
Declare Function GetObject Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Long, _
    ByVal nCount As Long, lpObject As Any) As Long
Declare Function GetBitmapBits Lib "gdi32" (ByVal hBitmap As Long, ByVal dwCount _
    As Long, lpBits As Any) As Long
Declare Function SetBitmapBits Lib "gdi32" (ByVal hBitmap As Long, ByVal dwCount _
    As Long, lpBits As Any) As Long
Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" _
    (ByVal lpszSoundName As String, ByVal uFlags As Long) As Long
```

Following figure shows the 'Form1' layout.



Following text shows the listing of 'Timer1' procedure code. For rest of the code refer to Lesson 9.

```
Private Sub Timer1_Timer()
    Timer1.Enabled = False
    SendMessage hwdc, WM_CAP_GET_FRAME, 0, 0
    SendMessage hwdc, WM_CAP_COPY, 0, 0
    Picture1.Picture = Clipboard.GetData
    GetBitmapBits Picture1.Picture, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
      Pbytes(0, 0, 0)

    Gray Picture1.ScaleWidth, Picture1.ScaleHeight
    If blkflag = True Then Negative Picture1.ScaleWidth, Picture1.ScaleHeight
    Lowpass Picture1.ScaleWidth, Picture1.ScaleHeight
    Contrast Picture1.ScaleWidth, Picture1.ScaleHeight
    For i = 1 To Val(Text4.Text)
      Geometricmean Picture1.ScaleWidth, Picture1.ScaleHeight, gms
    Next i
```

```
AdaptiveThreshold Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)
Clean Picture1.ScaleWidth, Picture1.ScaleHeight, Val(Text3.Text)
WhiteLineWidth Picture1.ScaleWidth, Val(Text2.Text), Val(Text3.Text)

SetBitmapBits Picture2.Image, Pinfo.bmWidthBytes * Pinfo.bmHeight, _
   Pbytes(0, 0, 0)
Picture2.Refresh
Picture2.Picture = Picture2.Image
Picture2.Line (Val(Text2.Text) - 2, Val(Text3.Text) - 2)-(Val(Text2.Text) _
 + 2, Val(Text3.Text) + 2), RGB(255, 0, 0), B

If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
   MIN_PATH_WIDTH Then
   If Text5.Text <> "No path" Then sndPlaySound "No path.wav", _
   SND_ASYNC Or SND_NODEFAULT
   Text5.Text = "No path"
Else
   If Text5.Text <> "Path found" Then sndPlaySound "Path found.wav", _
      SND_ASYNC Or SND_NODEFAULT
   Text5.Text = "Path found"
End If

If PixelCountLeft >= MIN_PATH_WIDTH And PixelCountRight < _
   MIN_PATH_WIDTH Then
   Text6.Text = 0: Text7.Text = 50        'turn left
   Text8.Text = "Turn left"
   Picture3.Picture = LoadPicture("turn_left.jpg")
End If
If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight >= _
   MIN_PATH_WIDTH Then
   Text6.Text = 50: Text7.Text = 0        'turn right
   Text8.Text = "Turn right"
   Picture3.Picture = LoadPicture("turn_right.jpg")
End If
If PixelCountLeft >= MIN_PATH_WIDTH And PixelCountRight >= _
   MIN_PATH_WIDTH Then
   Text6.Text = 100: Text7.Text = 100     'forward
   Text8.Text = "Forward"
   Picture3.Picture = LoadPicture("forward.jpg")
End If
```

```
    If PixelCountLeft < MIN_PATH_WIDTH And PixelCountRight < _
       MIN_PATH_WIDTH Then
       Text6.Text = 0: Text7.Text = 0      'no path - idle
       Text8.Text = ""
       Picture3.Picture = LoadPicture("blank.jpg")
    End If
    Timer1.Enabled = True
End Sub
```

Two pre-recorded wave files 'Nopath.wav' and 'Pathfound.wav' are used to play when needed through PC sound card interface. The PC sound recorder program may be used to create these wave files.

If the size of the captured image does not fit in the picture box then the image size has to be changed to 160x120 by activating the 'Format' menu. If no webcam is available then a message box will appear with a message "No webcam found!"

5.12 Summary

Ten lessons are presented in a step-by-step manner to develop programming skill for implementing vision-based navigation applications under 32-bit Windows environment.

Lesson 1: This demonstrates how to capture image frames from streaming video from a low-cost webcam and examine pixel values with the help of mouse pointer.

Lesson 2: This demonstrates how to process captured image frames from streaming video with two processing options covering color to gray-scale conversion and low-pass filtering.

Lesson 3: The method of contrast enhancement by histogram stretching technique is added to improve contrast under poor lighting conditions.

Lesson 4: The geometric-mean filter is added to smooth and suppress image detail.

Lesson 5: An adaptive threshold operation in introduced to extract white path under varying illumination conditions.

Lesson 6: A cleaning operation is provided to remove unwanted objects detected.

Lesson 7: Here an option is added for selection of path color white or black.

Lesson 8: Modified for white or black path searching for navigation with reference to a fixed pixel.

Lesson 9: Introduces a rule-based approach to determine left and right wheel speed settings of a differential drive system for navigation.

Lesson 10: Here sound output is added to draw attention during navigation.

References

[1] Balena, F.: Programming Microsoft Visual Basic 6. Microsoft Press (1999)
[2] Mandelbrot Set International Ltd., Advanced Microsoft Visual Basic 6. Microsoft Press (1998)
[3] Appleman, D.: Dan Appleman's Win32 API Puzzle Book and Tutorial for Visual Basic Programmers. Apress (1999)
[4] Gonzalez, Woods: Digital Image Processing. Prentice Hall (2002)