UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

DESARROLLO DE UNA APLICACIÓN PARA SISTEMAS DE VISIÓN POR COMPUTADOR DE ROBOTS INDUSTRIALES, BASADA EN EL REALCE Y PROCESAMIENTO MORFOLÓGICO DE IMÁGENES DIGITALES.

HERRERA, EMILIO

VALENCIA, OCTUBRE 2012

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

DEPARTAMENTO DE SISTEMAS Y AUTOMÁTICA

DESARROLLO DE UNA APLICACIÓN PARA SISTEMAS DE VISIÓN POR COMPUTADOR DE ROBOTS INDUSTRIALES, BASADA EN EL REALCE Y PROCESAMIENTO MORFOLÓGICO DE IMÁGENES DIGITALES.

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO ELECTRICISTA

HERRERA, EMILIO

VALENCIA, OCTUBRE 2012

UNIVERSIDAD DE CARABOBO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA ELÉCTRICA

DEPARTAMENTO DE SISTEMA Y AUTOMÁTICA

CERTIFICADO DE APROBACIÓN

Los abajo firmantes miembros del jurado asignado para evaluar el Trabajo Especial de Grado titulado **"Desarrollo de una aplicación para sistemas de visión por computador de robots industriales, basada en el realce y procesamiento morfológico de imágenes digitales",** realizado por el Bachiller(es): **Emilio Herrera,** Cédula de identidad: **V-18.699.977,** hacemos constar que hemos revisado y aprobado dicho trabajo.

Firma

_____

Prof. Wilmer Sanz

TUTOR

Firma

_____

Prof. Angel Villegas

JURADO

Firma

_____

Prof. Liliana Villavicencio

JURADO

VALENCIA, OCTUBRE 2012

INTRODUCCIÓN

La informática avanza actualmente hacia la creación de computadoras cada vez más veloces, más expertas y más autónomas. Teniendo en cuenta esto, es necesario otorgar a las computadoras de facultades para relacionarse con el entorno de la misma forma que lo hacen los humanos: a través de los sentidos.

Por ese motivo, el siguiente Trabajo Especial de Grado desarrolla una aplicación para sistemas de Visión por Computador de Robots Industriales, basada en el realce y procesamiento morfológico de imágenes digitales; pasando a través de diferentes etapas, desde los fundamentos teóricos, continuando con la transformación de la teoría a pseudocógido hasta el diseño de una interfaz de usuario que puede ser instalada en diferentes plataformas y arquitecturas de sistemas operativos pertenecientes a la Microsoft®, donde se desarrollan procedimientos para el filtrado de imágenes en el dominio espacial, detección de bordes, transformaciones tocantes a las forma de objetos y operaciones aritméticas en una interfaz gráfica llamada *Imagen Studio.*

Para finalizar, la aplicación aquí diseñada es evaluada en criterios de usabilidad y precisión con parámetros establecidos por la ISO mediante la aplicación de una prueba de usabilidad y se describe el uso de *Imagen Studio* bajo diversos entornos industriales, como por ejemplo el sector automovilístico para el estudio y detección de regiones de interés en placas, o en la industria alimenticia para la inspección de calidad en líneas de producción de galletas, y otros sectores industriales.

Ya basta de tener cámaras pasivas, bienvenido al mundo de la Visión Artificial para Robots Industriales.

Desarrollo de una aplicación para sistemas de visión por computador de robots industriales, basada en el realce y procesamiento morfológico de imágenes digitales.

Br. Emilio Herrera

Resumen. El presente Trabajo Especial de Grado lleva a cabo el desarrollo de una aplicación para la Visión por Computador de robots industriales basada en técnicas de procesamiento espacial, como lo son el realce y el procesamiento morfológico. Iniciando con el establecimiento de una base teórica que permite plantear las consideraciones y el procedimiento a seguir para luego ser transformado en lenguaje máquina. Este trabajo finaliza con la evaluación de la aplicación mediante una prueba de usabilidad y precisión. Posterior al desarrollo de esta aplicación, el usuario puede llevar a cabo sus propios procesamientos con la aplicación de diversos algoritmos como: operaciones aritméticas, detección de bordes empleando gradiente direccional, Sobel y Prewitt, manipulación del contraste mediante funciones definidas, obtención del histograma, cambio a perfiles de color a escala de grises, inversión de colores, segmentación de imágenes mediante la binarización, entre otros. La aplicación es una interfaz gráfica de usuario operativa en arquitecturas de 32 y 64 bits que posean sistemas como Windows XP®, Vista® o Windows 7®. La aplicación desarrollada incluye manual de instalación e inicio rápido y una ayuda posterior a la instalación.

Palabras clave. Filtros, imágenes, realce, procesamiento, Visión Artificial.

APÉNDICE A: DISEÑO DE LA APLICACIÓN.

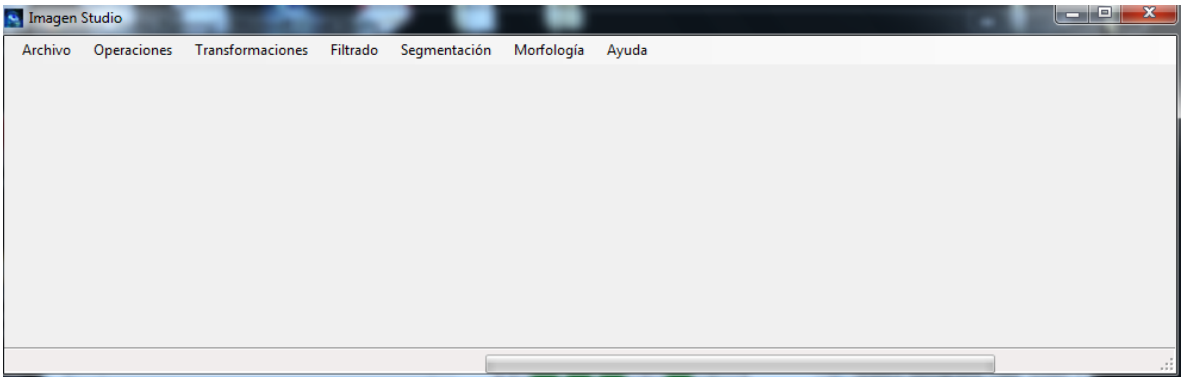


Figura A.1. Diseño general de la Aplicación.

Figura A.2. Diseño del menú Archivo.

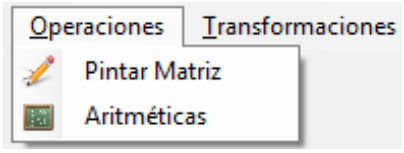


Figura A.3. Diseño del menú Operaciones.

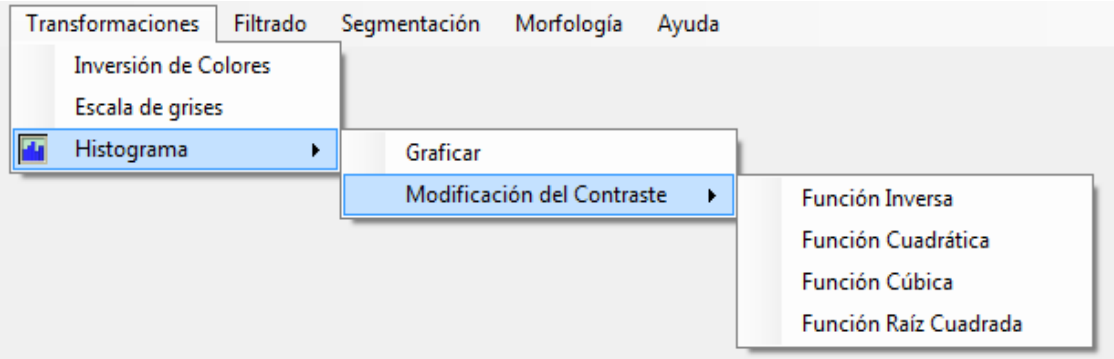


Figura A.4. Diseño del menú Transformaciones.

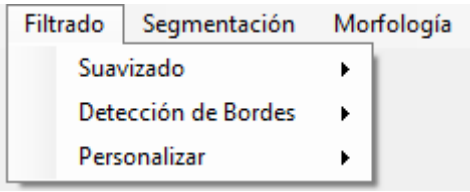


Figura A.5. Diseño del menú Filtrado.

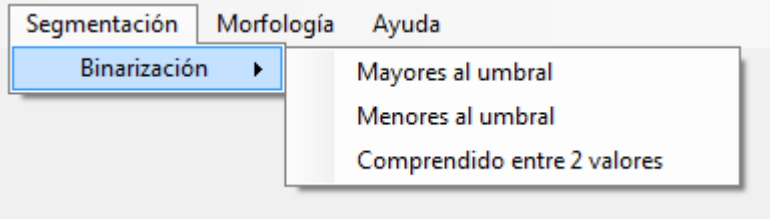


Figura A.6. Diseño del menú Segmentación.

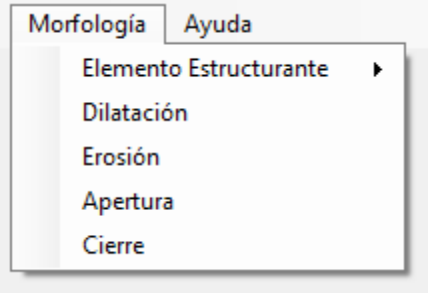


Figura A.7. Diseño del menú Morfología.

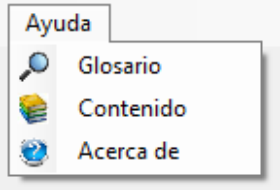


Figura A.8. Diseño del menú Ayuda.

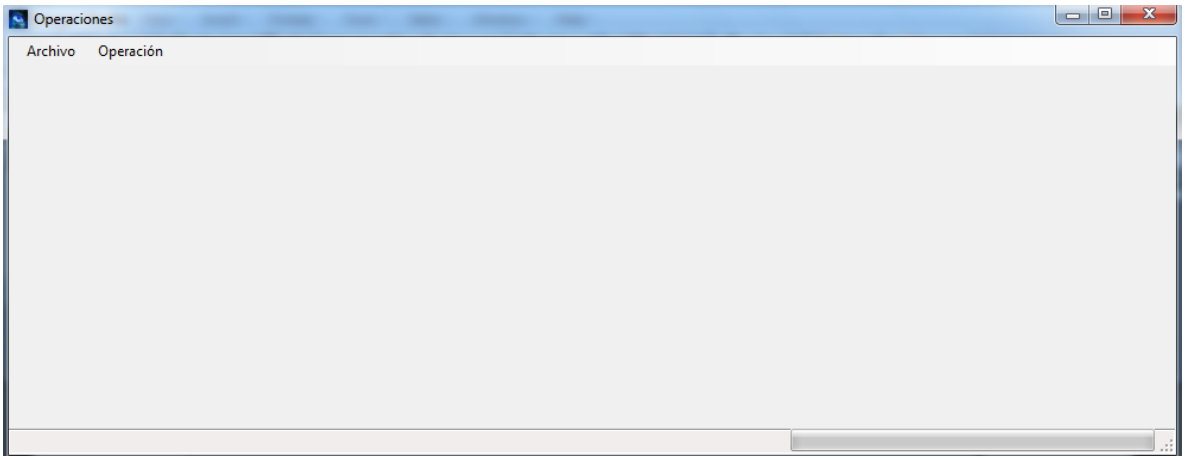


Figura A.9. Diseño general de la ventana de Operaciones Aritméticas.

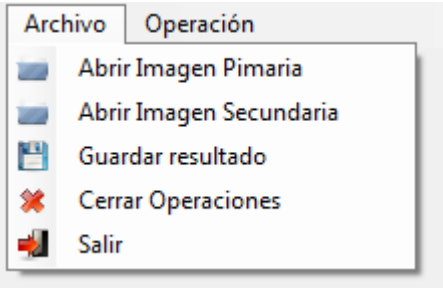


Figura A.10. Diseño del menú Archivo para Operaciones Aritméticas.

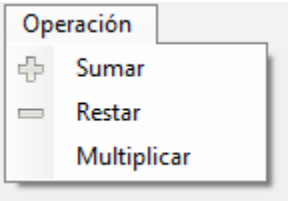


Figura A.11. Diseño del menú Operación para Operaciones Aritméticas.

APÉNDICE B: CÓDIGOS ASOCIADOS A LOS OBJETOS DE LA APLICACIÓN.

B.1: Declaraciones iniciales

```
'declaraciones
    'declaraciones para la propiedad de clase OpenFileDialog
    Dim OpenFileDialog1 As New OpenFileDialog()
    'declaraciones para la propiedad de clase SaveFileDialog
    Dim SaveFileDialog1 As New SaveFileDialog()

    'Declaracion para la varible matriz
    Public Matriz(,) As System.Drawing.Color
    Public Matriz2(,) As System.Drawing.Color


    Public ma(,) As System.Drawing.Color
    'Declaración de la mascara de convolución espacial
    Public Mascara(,) As Integer

    'Declaracion de la seleccion de un elemento estructurante
    Public Estructurante3 As Boolean = False

    Public Estructurante5 As Boolean = False


    'declaración para el tipo de procesado
    Public TipoProcesado As String


    'declaracion para los valores de los pixeles
    Friend WithEvents EtiquetaCoord_StatusStrip As ToolStripStatusLabel
    Friend WithEvents EtiquetaND_StatusStrip As ToolStripStatusLabel
```

B.2: Abrir Imagen

```
With OpenFileDialog1
            .Filter = "Ficheros BMP|*.bmp" & _
            "|Ficheros GIF|*.gif" & _
            "|Ficheros JPG o JPEG|*.jpg;*.jpeg" & _
            "|Ficheros PNG|*.png" & _
            "|Ficheros TIFF|*.tif"
            .FilterIndex = 3
            If (.ShowDialog() = Windows.Forms.DialogResult.OK) Then
                PictureBox1.Image = Image.FromFile(.FileName)
                Panel1.AutoScrollMinSize = PictureBox1.Image.Size
            End If
        End With

    End Sub
```

B.3: Guardar Imagen

```
With SaveFileDialog1
            .Filter = "Ficheros BMP|*.bmp" & _
            "|Ficheros GIF|*.gif" & _
            "|Ficheros JPG o JPEG|*.jpg;*.jpeg" & _
            "|Ficheros PNG|*.png" & _
            "|Ficheros TIFF|*.tif"
            .FilterIndex = 3
            If (.ShowDialog() = Windows.Forms.DialogResult.OK) Then
                If SaveFileDialog1.FileName <> "" Then
                    Dim fs As System.IO.FileStream =
CType(SaveFileDialog1.OpenFile(), System.IO.FileStream)
                    Select Case SaveFileDialog1.FilterIndex
                        'MOSCA CON SALVAR LA IMAGEN ORIGINAL XD, Picture2
es la que hay que salvar
                        Case 1
                            PictureBox2.Image.Save(fs, _
                            System.Drawing.Imaging.ImageFormat.Bmp)
                        Case 2
                            PictureBox2.Image.Save(fs, _
                            System.Drawing.Imaging.ImageFormat.Gif)
                        Case 3
                            PictureBox2.Image.Save(fs, _
                            System.Drawing.Imaging.ImageFormat.Jpeg)
                        Case 4
                            PictureBox2.Image.Save(fs, _
                            System.Drawing.Imaging.ImageFormat.Png)
                        Case 5
                            PictureBox2.Image.Save(fs, _
                            System.Drawing.Imaging.ImageFormat.Tiff)
                    End Select
                    fs.Close()
                End If
            End If
        End With

    End Sub
```

B.4: Cerrar

```
PictureBox1.Image = Nothing
        Panel1.AutoScroll = False
        PictureBox2.Image = Nothing
        Panel2.AutoScroll = False

    End Sub
```

B.5: Salir

```
End
    End Sub
```

B.6: Cargar matriz

```
Dim i, j As Long
        ReDim Matriz(PictureBox1.Image.Width - 1,
PictureBox1.Image.Height - 1)
        'se quita la unidad debido a que el minimo valor de la matriz
debe ser cero
        'y los valores de ancho y lago comienzan en 1. Todo esto por
respertar los
        'estándares

        'manejo de la barra de progreso


        With ProgressBar1
            ProgressBar1.Maximum = PictureBox1.Image.Width - 1
            ProgressBar1.Minimum = 0
        End With



        Dim bmp As New Bitmap(PictureBox1.Image)
        For i = 0 To PictureBox1.Image.Width - 1
            For j = 0 To PictureBox1.Image.Height - 1
                Matriz(i, j) = bmp.GetPixel(i, j)
            Next
            ProgressBar1.Value = i

        Next

   End Sub
```

B.7: Pintar Matriz

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        Dim i, j As Long
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        With ProgressBar1
            ProgressBar1.Maximum = Matriz.GetUpperBound(0)
            ProgressBar1.Minimum = 0
        End With
```

```
        Dim img As Image
        img = CType(bmp, Image)
        PictureBox2.Image = img
        Panel2.AutoScrollMinSize = PictureBox2.Image.Size
        PictureBox2.Refresh()
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                bmp.SetPixel(i, j, Matriz(i, j))
                ProgressBar1.Value = i
            Next
            'ProgressBar1.Value = i
            img = CType(bmp, Image)
            PictureBox2.Image = img
            PictureBox2.Refresh()

        Next
    End Sub
```

B.8: Invertir Colores

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        'inicio del codigo
        Dim Rojo, Verde, Azul As Byte
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                Rojo = 255 - Rojo
                Verde = 255 - Verde
                Azul = 255 - Azul

                bmp.SetPixel(i, j, Color.FromArgb(Rojo, Verde, Azul))
            Next

            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

        Next

    End Sub
```

B.9: Escala de grises

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        'inicio del codigo
        Dim Rojo, Verde, Azul, Gris As Byte
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                Rojo = Rojo / 3
                Verde = Verde / 3
                Azul = Azul / 3

                Gris = Rojo + Verde + Azul
                'End Select
                bmp.SetPixel(i, j, Color.FromArgb(Gris, Gris, Gris))
            Next
            'pintar matriz
            'no es más que la edición del comando pintar matriz
            'siempre deberia aparecer debajo de cada operación
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

        Next

    End Sub
```

H.10: Histograma Graficar

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        TipoProcesado = "GRAFICAR"
        'Inicio del algoritmo de inteigencia para color o gris
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Byte
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
```

```
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera
                End If
            Next
        Next

        If ban_gris = True And ban_color = False Then
            Grafico_histograma_gris.Show()
        Else
            Grafico_histograma_color.Show()

        End If
    End Sub
```

B.11: Suavizado de imágenes

```
'SUAVIZADO DE IMAGENES....eso es para imagenes de grises

        ProgressBar1.Value = 0 'resetear la barra de progreso

        Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
```

```
        Next

        If ban_gris = True And ban_color = False Then

            'Inicio del codigo del suavizado, con la mascara que se va
definir
            Dim mi, mj As Long
            Dim SumaRojo, SumaMascara As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 1 : Mascara(0, 2) = 1
            Mascara(1, 0) = 1 : Mascara(1, 1) = 1 : Mascara(1, 2) = 1
            Mascara(2, 0) = 1 : Mascara(2, 1) = 1 : Mascara(2, 2) = 1

            SumaMascara = 1
            factor = 1 / 9
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next

                    Rojo = Math.Abs(SumaRojo / SumaMascara)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next
        Else
            Dialog_no_es_gris.Show()
        End If
```

```
    End Sub
```

B.12: Suavizado Identidad (2)

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then

            'Inicio del codigo del suavizado, con la mascara que se va
definir

            '         1    1    1
            '(1/10)   1    2    1
            '         1    1    1

            Dim mi, mj As Long
            Dim SumaRojo, SumaMascara As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 1 : Mascara(0, 2) = 1
            Mascara(1, 0) = 1 : Mascara(1, 1) = 2 : Mascara(1, 2) = 1
            Mascara(2, 0) = 1 : Mascara(2, 1) = 1 : Mascara(2, 2) = 1
```

```
            SumaMascara = 1
            factor = 1 / 1 'cambiar el factor porque no veo resultados
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next

                    Rojo = Math.Abs(SumaRojo / SumaMascara)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next
        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.13: Suavizado Identidad (+2)

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
```

```
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then

            'Inicio del codigo del suavizado, con la mascara que se va
definir

            '         1    2    1
            '(1/16)   2    4    2
            '         1    2    1

            Dim mi, mj As Long
            Dim SumaRojo, SumaMascara As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 2 : Mascara(0, 2) = 1
            Mascara(1, 0) = 2 : Mascara(1, 1) = 4 : Mascara(1, 2) = 2
            Mascara(2, 0) = 1 : Mascara(2, 1) = 2 : Mascara(2, 2) = 1

            SumaMascara = 1
            factor = 1
            'factor = 1 / 16
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next

                    Rojo = Math.Abs(SumaRojo / SumaMascara)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
```

```
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next
        Else
            Dialog_no_es_gris.Show()
        End If
    End Sub
```

B.14: Modificación de contraste. Función Inversa

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        TipoProcesado = "FUNCION_INVERSA"
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Byte
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then
            Grafico_histograma_gris.Show()
        Else
            Grafico_histograma_color.Show()

        End If

    End Sub
```

B.15: Modificación de contraste. Función Cuadrada

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        TipoProcesado = "FUNCION_CUADRADA"
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Byte
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then
            Grafico_histograma_gris.Show()
        Else
            Grafico_histograma_color.Show()

        End If
    End Sub
```

B.16: Modificación de contraste. Función Cúbica

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        TipoProcesado = "FUNCION_CUBICA"
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Byte
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
```

```
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then
            Grafico_histograma_gris.Show()
        Else
            Grafico_histograma_color.Show()

        End If
    End Sub
```

B.17: Modificación de contraste. Función Raíz Cuadrada

```
ProgressBar1.Value = 0 'resetear la barra de progreso
        TipoProcesado = "FUNCION_RAIZ"
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Byte
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
```

```
        Next

        If ban_gris = True And ban_color = False Then
            Grafico_histograma_gris.Show()
        Else
            Grafico_histograma_color.Show()

        End If
    End Sub
```

B.18: Filtrado. Gradiente Horizontal

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then

            'Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double
            'Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
```

```
            Mascara(0, 0) = 0 : Mascara(0, 1) = 0 : Mascara(0, 2) = 0
            Mascara(1, 0) = 0 : Mascara(1, 1) = 1 : Mascara(1, 2) = -1
            Mascara(2, 0) = 0 : Mascara(2, 1) = 0 : Mascara(2, 2) = 0

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next
        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.19: Filtrado. Gradiente Vertical

```
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
```

```
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then



            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)

            Mascara(0, 0) = 0 : Mascara(0, 1) = 0 : Mascara(0, 2) = 0
            Mascara(1, 0) = 0 : Mascara(1, 1) = 1 : Mascara(1, 2) = 0
            Mascara(2, 0) = 0 : Mascara(2, 1) = -1 : Mascara(2, 2) = 0

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
```

```
            Next


        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.20: Filtrado. Sobel Horizontal

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then



            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
```

```
            Mascara(0, 0) = 1 : Mascara(0, 1) = 0 : Mascara(0, 2) = -1
            Mascara(1, 0) = 2 : Mascara(1, 1) = 0 : Mascara(1, 2) = -2
            Mascara(2, 0) = 1 : Mascara(2, 1) = 0 : Mascara(2, 2) = -1

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next


        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.21: Filtrado. Sobel Vertical

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
```

```
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else        ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then



            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 2 : Mascara(0, 2) = 1
            Mascara(1, 0) = 0 : Mascara(1, 1) = 0 : Mascara(1, 2) = 0
            Mascara(2, 0) = -1 : Mascara(2, 1) = -2 : Mascara(2, 2) = -1

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
```

```
                PictureBox2.Refresh()
            Next


        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.22: Filtrado. Prewitt Horizontal

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then



            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
```

```
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 0 : Mascara(0, 2) = -1
            Mascara(1, 0) = 1 : Mascara(1, 1) = 0 : Mascara(1, 2) = -1
            Mascara(2, 0) = 1 : Mascara(2, 1) = 0 : Mascara(2, 2) = -1

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next

        Else
            Dialog_no_es_gris.Show()
        End If


    End Sub
```

B.23: Filtrado. Prewitt Vertical

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
```

```
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else        ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then


            'Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim factor, desviacion As Double
            'Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            ReDim Mascara(2, 2)
            Mascara(0, 0) = 1 : Mascara(0, 1) = 1 : Mascara(0, 2) = 1
            Mascara(1, 0) = 0 : Mascara(1, 1) = 0 : Mascara(1, 2) = 0
            Mascara(2, 0) = -1 : Mascara(2, 1) = -1 : Mascara(2, 2) = -1

            factor = 1 / 1
            desviacion = 0

            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                        Next
                    Next
                    Rojo = Math.Abs(SumaRojo)
                    Rojo = Math.Abs(Rojo * (factor + desviacion))
                    If Rojo > 255 Then Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Next
                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
```

```
                PictureBox2.Refresh()
            Next

        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.24: Personalizar Filtrado 3x3

```
Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then



            Mascara_3.Show()

            If Mascara_3.listo = True Then

                'Dim i, j As Long
                Dim mi, mj As Long
                Dim SumaRojo As Long
                Dim factor, desviacion As Double
                'Dim Rojo As Integer
                Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
                Dim img As Image
```

```
                img = CType(bmp, Image)
                PictureBox2.Image = img
                Panel2.AutoScrollMinSize = PictureBox2.Image.Size
                PictureBox2.Refresh()
                ReDim Mascara(2, 2)
                Mascara(0, 0) = Mascara_3.valor1
                Mascara(0, 1) = Mascara_3.valor2
                Mascara(0, 2) = Mascara_3.valor3
                Mascara(1, 0) = Mascara_3.valor4
                Mascara(1, 1) = Mascara_3.valor5
                Mascara(1, 2) = Mascara_3.valor6
                Mascara(2, 0) = Mascara_3.valor7
                Mascara(2, 1) = Mascara_3.valor8
                Mascara(2, 2) = Mascara_3.valor9

                factor = (Mascara_3.num) / (Mascara_3.den)

                desviacion = 0

                'Con este cuadruple bucle se recorre la máscara y la
matriz
                For i = 1 To Matriz.GetUpperBound(0) - 1
                    For j = 1 To Matriz.GetUpperBound(1) - 1
                        SumaRojo = 0
                        For mi = -1 To 1
                            For mj = -1 To 1
                                SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)
                            Next
                        Next
                        Rojo = Math.Abs(SumaRojo / 1)
                        Rojo = Math.Abs(Rojo * (factor + desviacion))
                        If Rojo > 255 Then Rojo = 255
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Next
                    'Esta aquí que se refresque línea a línea
                    img = CType(bmp, Image)
                    PictureBox2.Image = img
                    PictureBox2.Refresh()
                Next
            End If


        Else
            Dialog_no_es_gris.Show()
        End If

    End Sub
```

B.25: Personalizar Filtrado 5x5

```
        Dim i, j As Long
        Dim Rojo, Verde, Azul As Integer
        Dim ban_gris, ban_color As Boolean
        ban_color = False
        For i = 0 To Matriz.GetUpperBound(0)
            For j = 0 To Matriz.GetUpperBound(1)
                Rojo = Matriz(i, j).R
                Verde = Matriz(i, j).G
                Azul = Matriz(i, j).B

                If Rojo = Verde And Verde = Azul And Rojo = Azul Then
                    ban_gris = True ' si la imagen es gris, siempre se va
levantar la bandera
                Else       ' caso contrario
                    ban_color = True ' si existe un pixel de color, se
levanta esta bandera

                End If

            Next
        Next

        If ban_gris = True And ban_color = False Then

            Mascara_5.Show()

            If Mascara_5.listo = True Then

                'Dim i, j As Long
                Dim mi, mj As Long
                Dim SumaRojo As Long
                Dim factor, desviacion As Double
                'Dim Rojo As Integer
                Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
                Dim img As Image
                img = CType(bmp, Image)
                PictureBox2.Image = img
                Panel2.AutoScrollMinSize = PictureBox2.Image.Size
                PictureBox2.Refresh()
                ReDim Mascara(5, 5)
                Mascara(0, 0) = Mascara_5.valor1
                Mascara(0, 1) = Mascara_5.valor2
                Mascara(0, 2) = Mascara_5.valor3
                Mascara(0, 3) = Mascara_5.valor4
                Mascara(0, 4) = Mascara_5.valor5

                Mascara(1, 0) = Mascara_5.valor6
                Mascara(1, 1) = Mascara_5.valor7
```

```
                Mascara(1, 2) = Mascara_5.valor8
                Mascara(1, 3) = Mascara_5.valor9
                Mascara(1, 4) = Mascara_5.valor10

                Mascara(2, 0) = Mascara_5.valor11
                Mascara(2, 1) = Mascara_5.valor12
                Mascara(2, 2) = Mascara_5.valor13
                Mascara(2, 3) = Mascara_5.valor14
                Mascara(2, 4) = Mascara_5.valor15

                Mascara(3, 0) = Mascara_5.valor16
                Mascara(3, 1) = Mascara_5.valor17
                Mascara(3, 2) = Mascara_5.valor18
                Mascara(3, 3) = Mascara_5.valor19
                Mascara(3, 4) = Mascara_5.valor20

                Mascara(4, 0) = Mascara_5.valor21
                Mascara(4, 1) = Mascara_5.valor22
                Mascara(4, 2) = Mascara_5.valor23
                Mascara(4, 3) = Mascara_5.valor24
                Mascara(4, 4) = Mascara_5.valor25

                factor = (Mascara_5.num) / (Mascara_5.den)

                desviacion = 0

                'Con este cuadruple bucle se recorre la máscara y la
matriz
                For i = 2 To Matriz.GetUpperBound(0) - 2
                    For j = 2 To Matriz.GetUpperBound(1) - 2
                        SumaRojo = 0
                        For mi = -2 To 2
                            For mj = -2 To 2
                                SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 2, mj + 2)
                            Next
                        Next
                        Rojo = Math.Abs(SumaRojo / 1)
                        Rojo = Math.Abs(Rojo * (factor + desviacion))
                        If Rojo > 255 Then Rojo = 255
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Next
                    'Esta aquí que se refresque línea a línea
                    img = CType(bmp, Image)
                    PictureBox2.Image = img
                    PictureBox2.Refresh()
                Next
            End If

        Else
            Dialog_no_es_gris.Show()
```

```
        End If
    End Sub
```

B.26: Binarización mayores al valor de umbral

```
Dialog1.Show()

        If Dialog1.umb = True Then

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim i, j As Long
            'inicio del codigo
            Dim Rojo, Gris As Byte
            For i = 0 To Matriz.GetUpperBound(0)
                For j = 0 To Matriz.GetUpperBound(1)
                    Rojo = Matriz(i, j).R

                    If Rojo >= Dialog1.Umbral Then
                        Gris = 255
                    Else
                        Gris = 0
                    End If

                    bmp.SetPixel(i, j, Color.FromArgb(Gris, Gris, Gris))
                Next
            Next

            'pintar matriz
            'no es más que la edición del comando pintar matriz
            'siempre deberia aparecer debaja de cada operación
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()


        End If
    End Sub
```

B.27: Binarización menores al valor de umbral

```
Dialog2_menores_umbral.Show()

        If Dialog2_menores_umbral.umb = True Then

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
```

```
            Dim i, j As Long
            'inicio del codigo
            Dim Rojo, Gris As Byte
            For i = 0 To Matriz.GetUpperBound(0)
                For j = 0 To Matriz.GetUpperBound(1)
                    Rojo = Matriz(i, j).R

                    If Rojo <= Dialog2_menores_umbral.Umbral Then
                        Gris = 255
                    Else
                        Gris = 0
                    End If

                    bmp.SetPixel(i, j, Color.FromArgb(Gris, Gris, Gris))
                Next
            Next

            'pintar matriz
            'no es más que la edición del comando pintar matriz
            'siempre deberia aparecer debaja de cada operación
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

        End If
    End Sub
```

B.28: Binarización comprendida entre 2 valores de umbral

```
Dialog_entre_umbrales.Show()

        If Dialog_entre_umbrales.umb = True Then

            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim i, j As Long
            'inicio del codigo
            Dim Rojo, Gris As Byte
            For i = 0 To Matriz.GetUpperBound(0)
                For j = 0 To Matriz.GetUpperBound(1)
                    Rojo = Matriz(i, j).R

                    If Rojo >= Dialog_entre_umbrales.UmbralMin And Rojo
<= Dialog_entre_umbrales.UmbralMax Then
                        Gris = 255
                    Else
                        Gris = 0
                    End If
```

```
                    bmp.SetPixel(i, j, Color.FromArgb(Gris, Gris, Gris))
                Next
            Next

            'pintar matriz
            'no es más que la edición del comando pintar matriz
            'siempre deberia aparecer debaja de cada operación
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

        End If
    End Sub
```

B.29: Elemento Estructurante Cuadrado 3x3

```
Estructurante3 = True

        ReDim Mascara(3, 3)
        'un cuadrado
        ' 1 1 1
        ' 1 1 1
        ' 1 1 1

        Mascara(0, 0) = 1
        Mascara(0, 1) = 1
        Mascara(0, 2) = 1

        Mascara(1, 0) = 1
        Mascara(1, 1) = 1
        Mascara(1, 2) = 1

        Mascara(2, 0) = 1
        Mascara(2, 1) = 1
        Mascara(2, 2) = 1

    End Sub
```

B.30: Elemento Estructurante Cruz 3x3

```
Estructurante3 = True

        ReDim Mascara(3, 3)
        'una cruz
        ' 0 1 0
        ' 1 1 1
```

```
        ' 0 1 0

        Mascara(0, 0) = 0
        Mascara(0, 1) = 1
        Mascara(0, 2) = 0

        Mascara(1, 0) = 1
        Mascara(1, 1) = 1
        Mascara(1, 2) = 1

        Mascara(2, 0) = 0
        Mascara(2, 1) = 1
        Mascara(2, 2) = 0
    End Sub
```

B.31: Elemento Estructurante Equis 3x3

```
Estructurante3 = True

        ReDim Mascara(3, 3)
        'una equis
        ' 1 0 1
        ' 0 1 0
        ' 1 0 1

        Mascara(0, 0) = 1
        Mascara(0, 1) = 0
        Mascara(0, 2) = 1

        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 0

        Mascara(2, 0) = 1
        Mascara(2, 1) = 0
        Mascara(2, 2) = 1
    End Sub
```

B.32: Elemento Estructurante Línea Horizontal 3x3

```
Estructurante3 = True

        ReDim Mascara(3, 3)
        'una linea horizontal
        ' 0 0 0
        ' 1 1 1
        ' 0 0 0
```

```
        Mascara(0, 0) = 0
        Mascara(0, 1) = 0
        Mascara(0, 2) = 0

        Mascara(1, 0) = 1
        Mascara(1, 1) = 1
        Mascara(1, 2) = 1

        Mascara(2, 0) = 0
        Mascara(2, 1) = 0
        Mascara(2, 2) = 0
    End Sub
```

B.33: Elemento Estructurante Línea Vertical 3x3

```
Estructurante3 = True

        ReDim Mascara(3, 3)
        'una linea vertical
        ' 0 1 0
        ' 0 1 0
        ' 0 1 0

        Mascara(0, 0) = 0
        Mascara(0, 1) = 1
        Mascara(0, 2) = 0

        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 0

        Mascara(2, 0) = 0
        Mascara(2, 1) = 1
        Mascara(2, 2) = 0
    End Sub
```

B.34: Elemento Estructurante Diagonal Izquierda 3x3

```
Estructurante3 = True
        ReDim Mascara(3, 3)
        'diagonal izquierda
        ' 0 0 1
        ' 0 1 0
        ' 1 0 0

        Mascara(0, 0) = 0
        Mascara(0, 1) = 0
        Mascara(0, 2) = 1
```

```
        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 0

        Mascara(2, 0) = 1
        Mascara(2, 1) = 0
        Mascara(2, 2) = 0
    End Sub
```

B.35: Elemento Estructurante Cuadrado 5x5

```
Estructurante5 = True

        ReDim Mascara(5, 5)
        'un cuadrado
        ' 1 1 1 1 1
        ' 1 1 1 1 1
        ' 1 1 1 1 1
        ' 1 1 1 1 1
        ' 1 1 1 1 1

        Mascara(0, 0) = 1
        Mascara(0, 1) = 1
        Mascara(0, 2) = 1
        Mascara(0, 3) = 1
        Mascara(0, 4) = 1

        Mascara(1, 0) = 1
        Mascara(1, 1) = 1
        Mascara(1, 2) = 1
        Mascara(1, 3) = 1
        Mascara(1, 4) = 1

        Mascara(2, 0) = 1
        Mascara(2, 1) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1

        Mascara(3, 0) = 1
        Mascara(3, 1) = 1
        Mascara(3, 2) = 1
        Mascara(3, 3) = 1
        Mascara(3, 4) = 1

        Mascara(4, 0) = 1
        Mascara(4, 1) = 1
        Mascara(4, 2) = 1
        Mascara(4, 3) = 1
```

```
        Mascara(4, 4) = 1

    End Sub
```

B.36: Elemento Estructurante Equis 5x5

```
Estructurante5 = True

        ReDim Mascara(5, 5)
        'una equis
        ' 1 0 0 0 1
        ' 0 1 0 1 0
        ' 0 0 1 0 0
        ' 0 1 0 1 0
        ' 1 0 0 0 1

        Mascara(0, 0) = 1
        Mascara(0, 1) = 0
        Mascara(0, 2) = 0
        Mascara(0, 3) = 0
        Mascara(0, 4) = 1

        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 0
        Mascara(1, 3) = 1
        Mascara(1, 4) = 0

        Mascara(2, 0) = 0
        Mascara(2, 1) = 0
        Mascara(2, 2) = 1
        Mascara(2, 2) = 0
        Mascara(2, 2) = 0

        Mascara(3, 0) = 0
        Mascara(3, 1) = 1
        Mascara(3, 2) = 0
        Mascara(3, 3) = 1
        Mascara(3, 4) = 0

        Mascara(4, 0) = 1
        Mascara(4, 1) = 0
        Mascara(4, 2) = 0
        Mascara(4, 3) = 0
        Mascara(4, 4) = 1

    End Sub
```

B.37: Elemento Estructurante Cruz 5x5

```
Estructurante5 = True
        ReDim Mascara(5, 5)
        'una cruz
        ' 0 0 1 0 0
        ' 0 0 1 0 0
        ' 1 1 1 1 1
        ' 0 0 1 0 0
        ' 0 0 1 0 0

        Mascara(0, 0) = 0
        Mascara(0, 1) = 0
        Mascara(0, 2) = 1
        Mascara(0, 3) = 0
        Mascara(0, 4) = 0

        Mascara(1, 0) = 0
        Mascara(1, 1) = 0
        Mascara(1, 2) = 1
        Mascara(1, 3) = 0
        Mascara(1, 4) = 0

        Mascara(2, 0) = 1
        Mascara(2, 1) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1

        Mascara(3, 0) = 0
        Mascara(3, 1) = 0
        Mascara(3, 2) = 1
        Mascara(3, 3) = 0
        Mascara(3, 4) = 0

        Mascara(4, 0) = 0
        Mascara(4, 1) = 0
        Mascara(4, 2) = 1
        Mascara(4, 3) = 0
        Mascara(4, 4) = 0

    End Sub
```

B.38: Elemento Estructurante Rombo 5x5

```
Estructurante5 = True

        ReDim Mascara(5, 5)
        'un rombo
```

```
        ' 0 0 1 0 0
        ' 0 1 1 1 0
        ' 1 1 1 1 1
        ' 0 1 1 1 0
        ' 0 0 1 0 0

        Mascara(0, 0) = 0
        Mascara(0, 1) = 0
        Mascara(0, 2) = 1
        Mascara(0, 3) = 0
        Mascara(0, 4) = 0

        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 1
        Mascara(1, 3) = 1
        Mascara(1, 4) = 0

        Mascara(2, 0) = 1
        Mascara(2, 1) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1
        Mascara(2, 2) = 1

        Mascara(3, 0) = 0
        Mascara(3, 1) = 1
        Mascara(3, 2) = 1
        Mascara(3, 3) = 1
        Mascara(3, 4) = 0

        Mascara(4, 0) = 0
        Mascara(4, 1) = 0
        Mascara(4, 2) = 1
        Mascara(4, 3) = 0
        Mascara(4, 4) = 0

    End Sub
```

B.39: Elemento Estructurante Diagonal Izquierda 5x5

```
Estructurante5 = True

        ReDim Mascara(5, 5)
        'diagonal izquierda
        ' 0 0 0 0 1
        ' 0 0 0 1 0
        ' 0 0 1 0 0
        ' 0 1 0 0 0
        ' 1 0 0 0 0
```

```
        Mascara(0, 0) = 0
        Mascara(0, 1) = 0
        Mascara(0, 2) = 0
        Mascara(0, 3) = 0
        Mascara(0, 4) = 1

        Mascara(1, 0) = 0
        Mascara(1, 1) = 0
        Mascara(1, 2) = 0
        Mascara(1, 3) = 1
        Mascara(1, 4) = 0

        Mascara(2, 0) = 0
        Mascara(2, 1) = 0
        Mascara(2, 2) = 1
        Mascara(2, 2) = 0
        Mascara(2, 2) = 0

        Mascara(3, 0) = 0
        Mascara(3, 1) = 1
        Mascara(3, 2) = 0
        Mascara(3, 3) = 0
        Mascara(3, 4) = 0

        Mascara(4, 0) = 1
        Mascara(4, 1) = 0
        Mascara(4, 2) = 0
        Mascara(4, 3) = 0
        Mascara(4, 4) = 0

    End Sub
```

B.40: Elemento Estructurante Diagonal Derecha 5x5

```
Estructurante5 = True

        ReDim Mascara(5, 5)
        'diagonal derecha
        ' 1 0 0 0 0
        ' 0 1 0 0 0
        ' 0 0 1 0 0
        ' 0 0 0 1 0
        ' 0 0 0 0 1

        Mascara(0, 0) = 1
        Mascara(0, 1) = 0
        Mascara(0, 2) = 0
        Mascara(0, 3) = 0
```

```
        Mascara(0, 4) = 0

        Mascara(1, 0) = 0
        Mascara(1, 1) = 1
        Mascara(1, 2) = 0
        Mascara(1, 3) = 0
        Mascara(1, 4) = 0

        Mascara(2, 0) = 0
        Mascara(2, 1) = 0
        Mascara(2, 2) = 1
        Mascara(2, 2) = 0
        Mascara(2, 2) = 0

        Mascara(3, 0) = 0
        Mascara(3, 1) = 0
        Mascara(3, 2) = 0
        Mascara(3, 3) = 1
        Mascara(3, 4) = 0

        Mascara(4, 0) = 0
        Mascara(4, 1) = 0
        Mascara(4, 2) = 0
        Mascara(4, 3) = 0
        Mascara(4, 4) = 1

    End Sub
```

B.41: Dilatación

```
        If Estructurante3 = True Then
            Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
```

```
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)

                        Next
                    Next

                    If SumaRojo <> 0 Then
                        Rojo = 255
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Else
                        Rojo = 0
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))

                    End If

                Next

                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next


        ElseIf Estructurante5 = True Then
            Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo As Long
            Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()
            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 2 To Matriz.GetUpperBound(0) - 2
                For j = 2 To Matriz.GetUpperBound(1) - 2
                    SumaRojo = 0
                    For mi = -2 To 2
                        For mj = -2 To 2
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 2, mj + 2)

                        Next
                    Next

                    If SumaRojo <> 0 Then
                        Rojo = 255
```

```
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Else
                        Rojo = 0
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))

                    End If

                Next

                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next

        Else
            ' viene el caso de no haber seleccionado ningun elemento
estructural


        End If
    End Sub
```

B.42: Erosión

```
If Estructurante3 = True Then

            Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo, SumaMascara As Long
            Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

            For mi = -1 To 1
                For mj = -1 To 1
                    SumaMascara = SumaMascara + Mascara(mi + 1, mj + 1)
                    'aqui calculo la suma de las componentes que forman
el estructurante
                Next
            Next
```

```
            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 1 To Matriz.GetUpperBound(0) - 1
                For j = 1 To Matriz.GetUpperBound(1) - 1
                    SumaRojo = 0
                    For mi = -1 To 1
                        For mj = -1 To 1
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 1, mj + 1)

                        Next
                    Next

                    SumaRojo = SumaRojo / 255

                    If SumaRojo = SumaMascara Then
                        Rojo = 255
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Else
                        Rojo = 0
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))

                    End If


                Next

                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next


        ElseIf Estructurante5 = True Then
            Dim i, j As Long
            Dim mi, mj As Long
            Dim SumaRojo, SumaMascara As Long
            Dim Rojo As Integer
            Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
            Dim img As Image
            img = CType(bmp, Image)
            PictureBox2.Image = img
            Panel2.AutoScrollMinSize = PictureBox2.Image.Size
            PictureBox2.Refresh()

            For mi = -2 To 2
                For mj = -2 To 2
                    SumaMascara = SumaMascara + Mascara(mi + 2, mj + 2)
```

```
                    'aqui calculo la suma de las componentes que forman
el estructurante
                Next
            Next


            'Con este cuadruple bucle se recorre la máscara y la matriz
            For i = 2 To Matriz.GetUpperBound(0) - 2
                For j = 2 To Matriz.GetUpperBound(1) - 2
                    SumaRojo = 0
                    For mi = -2 To 2
                        For mj = -2 To 2
                            SumaRojo = SumaRojo + Matriz(i + mi, j +
mj).R * Mascara(mi + 2, mj + 2)

                        Next
                    Next

                    SumaRojo = SumaRojo / 255

                    If SumaRojo = SumaMascara Then
                        Rojo = 255
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))
                    Else
                        Rojo = 0
                        bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo,
Rojo))

                    End If


                Next

                'Esta aquí que se refresque línea a línea
                img = CType(bmp, Image)
                PictureBox2.Image = img
                PictureBox2.Refresh()
            Next

        Else
            ' viene el caso de no haber seleccionado ningun elemento
estructural


        End If


    End Sub
```

B.43: Apertura

```
Dim i, j As Long
        Dim mi, mj As Long
        Dim SumaRojo, SumaMascara As Long
        Dim Rojo As Integer
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim img As Image
        img = CType(bmp, Image)
        PictureBox2.Image = img
        Panel2.AutoScrollMinSize = PictureBox2.Image.Size
        PictureBox2.Refresh()

        For mi = -1 To 1
            For mj = -1 To 1
                SumaMascara = SumaMascara + Mascara(mi + 1, mj + 1)
                'aqui calculo la suma de las componentes que forman el
estructurante
            Next
        Next

        'inicio la erosion

        'Con este cuadruple bucle se recorre la máscara y la matriz
        For i = 1 To Matriz.GetUpperBound(0) - 1
            For j = 1 To Matriz.GetUpperBound(1) - 1
                SumaRojo = 0
                For mi = -1 To 1
                    For mj = -1 To 1
                        SumaRojo = SumaRojo + Matriz(i + mi, j + mj).R *
Mascara(mi + 1, mj + 1)

                    Next
                Next

                SumaRojo = SumaRojo / 255

                If SumaRojo = SumaMascara Then
                    Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))

                Else
                    Rojo = 0
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                End If

            Next
            img = CType(bmp, Image)
            PictureBox2.Image = img
            PictureBox2.Refresh()
```

```
        Next


        ReDim ma(PictureBox2.Image.Width - 1, PictureBox2.Image.Height -
1)

        Dim bmp0 As New Bitmap(PictureBox2.Image)
        For i = 0 To PictureBox2.Image.Width - 1
            For j = 0 To PictureBox2.Image.Height - 1
                Ma(i, j) = bmp0.GetPixel(i, j)
            Next
        Next
        'finaliza la erosion e inicia la dilatacion



        For i = 1 To ma.GetUpperBound(0) - 1
            For j = 1 To ma.GetUpperBound(1) - 1
                SumaRojo = 0
                For mi = -1 To 1
                    For mj = -1 To 1
                        SumaRojo = SumaRojo + ma(i + mi, j + mj).R *
Mascara(mi + 1, mj + 1)

                    Next
                Next

                If SumaRojo <> 0 Then
                    Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Else
                    Rojo = 0
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))

                End If

            Next

            'Esta aquí que se refresque línea a línea
            img = CType(bmp, Image)
            PictureBox2.Image = img
            PictureBox2.Refresh()

        Next

    End Sub
```

B.44: Cierre

```
        Dim i, j As Long
        Dim mi, mj As Long
        Dim SumaRojo, SumaMascara As Long
        Dim Rojo As Integer
        Dim bmp As New Bitmap(Matriz.GetUpperBound(0) + 1,
Matriz.GetUpperBound(1) + 1)
        Dim img As Image
        img = CType(bmp, Image)
        PictureBox2.Image = img
        Panel2.AutoScrollMinSize = PictureBox2.Image.Size
        PictureBox2.Refresh()

        For mi = -1 To 1
            For mj = -1 To 1
                SumaMascara = SumaMascara + Mascara(mi + 1, mj + 1)
                'aqui calculo la suma de las componentes que forman el
estructurante
            Next
        Next

        'inicio la dilatación

        'Con este cuadruple bucle se recorre la máscara y la matriz
        For i = 1 To Matriz.GetUpperBound(0) - 1
            For j = 1 To Matriz.GetUpperBound(1) - 1
                SumaRojo = 0
                For mi = -1 To 1
                    For mj = -1 To 1
                        SumaRojo = SumaRojo + Matriz(i + mi, j + mj).R *
Mascara(mi + 1, mj + 1)

                    Next
                Next

                If SumaRojo <> 0 Then
                    Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                Else
                    Rojo = 0
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))

                End If

            Next
            img = CType(bmp, Image)
            PictureBox2.Image = img
            PictureBox2.Refresh()
        Next
```

```
        ReDim ma(PictureBox2.Image.Width - 1, PictureBox2.Image.Height -
1)

        Dim bmp0 As New Bitmap(PictureBox2.Image)
        For i = 0 To PictureBox2.Image.Width - 1
            For j = 0 To PictureBox2.Image.Height - 1
                ma(i, j) = bmp0.GetPixel(i, j)
            Next
        Next
        'finaliza la erosion e inicia la



        For i = 1 To ma.GetUpperBound(0) - 1
            For j = 1 To ma.GetUpperBound(1) - 1
                SumaRojo = 0
                For mi = -1 To 1
                    For mj = -1 To 1
                        SumaRojo = SumaRojo + ma(i + mi, j + mj).R *
Mascara(mi + 1, mj + 1)

                    Next
                Next

                SumaRojo = SumaRojo / 255

                If SumaRojo = SumaMascara Then
                    Rojo = 255
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))

                Else
                    Rojo = 0
                    bmp.SetPixel(i, j, Color.FromArgb(Rojo, Rojo, Rojo))
                End If


            Next

            'Esta aquí que se refresque línea a línea
            img = CType(bmp, Image)
            PictureBox2.Image = img
            PictureBox2.Refresh()

        Next



    End Sub
```

B.45: Personalizar Elemento Estructurante 3x3

```
Personalizar_mas_3.Show()

        If Estructurante3 = True Then
            ReDim Mascara(3, 3)

            Mascara(0, 0) = Personalizar_mas_3.val1
            Mascara(0, 1) = Personalizar_mas_3.val2
            Mascara(0, 2) = Personalizar_mas_3.val3

            Mascara(1, 0) = Personalizar_mas_3.val4
            Mascara(1, 1) = Personalizar_mas_3.val5
            Mascara(1, 2) = Personalizar_mas_3.val6

            Mascara(2, 0) = Personalizar_mas_3.val7
            Mascara(2, 1) = Personalizar_mas_3.val8
            Mascara(2, 2) = Personalizar_mas_3.val9

        End If

    End Sub
```

B.46: Personalizar Elemento Estrcturante 5x5

```
Personalizar_mas_5.Show()

        If Estructurante5 = True Then
            ReDim Mascara(5, 5)

            Mascara(0, 0) = Personalizar_mas_5.val1
            Mascara(0, 1) = Personalizar_mas_5.val2
            Mascara(0, 2) = Personalizar_mas_5.val3
            Mascara(0, 3) = Personalizar_mas_5.val4
            Mascara(0, 4) = Personalizar_mas_5.val5

            Mascara(1, 0) = Personalizar_mas_5.val6
            Mascara(1, 1) = Personalizar_mas_5.val7
            Mascara(1, 2) = Personalizar_mas_5.val8
            Mascara(1, 3) = Personalizar_mas_5.val9
            Mascara(1, 4) = Personalizar_mas_5.val10

            Mascara(2, 0) = Personalizar_mas_5.val11
            Mascara(2, 1) = Personalizar_mas_5.val12
            Mascara(2, 2) = Personalizar_mas_5.val13
            Mascara(2, 3) = Personalizar_mas_5.val14
            Mascara(2, 4) = Personalizar_mas_5.val15

            Mascara(3, 0) = Personalizar_mas_5.val16
```

```
            Mascara(3, 1) = Personalizar_mas_5.val17
            Mascara(3, 2) = Personalizar_mas_5.val18
            Mascara(3, 3) = Personalizar_mas_5.val19
            Mascara(3, 4) = Personalizar_mas_5.val20

            Mascara(4, 0) = Personalizar_mas_5.val21
            Mascara(4, 1) = Personalizar_mas_5.val22
            Mascara(4, 2) = Personalizar_mas_5.val23
            Mascara(4, 3) = Personalizar_mas_5.val24
            Mascara(4, 4) = Personalizar_mas_5.val25

        End If
    End Sub
```

B.47: Visualización de los Perfiles RGB de las Imágenes

```
Me.EtiquetaCoord_StatusStrip = New ToolStripStatusLabel
        Me.StatusStrip1.Items.AddRange(New ToolStripItem()
{Me.EtiquetaCoord_StatusStrip})
        Me.EtiquetaND_StatusStrip = New ToolStripStatusLabel
        Me.StatusStrip1.Items.AddRange(New ToolStripItem()
{Me.EtiquetaND_StatusStrip})

        PictureBox1.Cursor = Cursors.Cross
        PictureBox2.Cursor = Cursors.Cross

    End Sub
```

B.48: Lectura de coordenadas Imagen Original

```
EtiquetaCoord_StatusStrip.Text = "Coordenadas X: " & CStr(e.X) & " ; " &
_
"Y: " & CStr(e.Y)
        'X e Y empiezan en 0
        If Not (Matriz Is Nothing) Then
            EtiquetaND_StatusStrip.Text = "Perfiles de Color R:" &
CStr(Matriz(e.X, e.Y).R) & " ; " & _
            "G:" & CStr(Matriz(e.X, e.Y).G) & " ; " & _
            "B:" & CStr(Matriz(e.X, e.Y).B)
        End If
```

B.49: Lectura de las coordenadas Imagen Procesada

```
EtiquetaCoord_StatusStrip.Text = "Coordenadas X: " & CStr(e.X) & " ; " &
_
"Y: " & CStr(e.Y)
```

```
        'X e Y empiezan en 0
        If Not (Matriz2 Is Nothing) Then
            EtiquetaND_StatusStrip.Text = "Perfiles de Color R:" &
CStr(Matriz2(e.X, e.Y).R) & " ; " & _
            "G:" & CStr(Matriz2(e.X, e.Y).G) & " ; " & _
            "B:" & CStr(Matriz2(e.X, e.Y).B)
        End If
```

B.50: Glosario

```
Glosario.Show()

End Sub
```

B.51: Ayuda

```
Ayuda_Principal.Show()

    End Sub
```

B.52: Acerca de

```
Acerca_de.Show()
```

APÉNDICE C: MODELO PRUEBA DE USABILIDAD APLICADA.

Bienvenido al Imagen Studio. El siguiente formulario será empleado para definir la usabilidad y precisión del software mencionado.

Para esto, siga secuencialmente la tarea indicada y conteste a medida que este formulario le indique.

1) Ingrese al programa haciendo doble click en el ícono de Imagen Studio.
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

2) Ingrese al menú Archivo e intente abrir la imagen de su elección.
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

3) Ingrese al menú de operaciones y pinte la imagen que abrió en asignación anterior, para esto use el submenú de pintar matriz.
   a. ¿Pudo usted llevar a cabo la tarea pintar matriz? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

4) Intente detectar los bordes de la imagen aplicando el algoritmo de Sobel en la dirección horizontal. Para esto recurra al menú de Filtrado.
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.
   c. ¿Presentó algún mensaje de error la ejecución de esta tarea? SI:_. NO:_. NA:_.
   d. ¿Sintió alguna incomodad por el contenido del mensaje de error? SI:_. NO: _. NA:__.
   e. ¿Cree Usted estar capacitado para la solución de este problema? SI:__. NO:__. NA:__.
5) Nuevamente intente detectar los bordes aplicando el algoritmo de Sobel en la dirección horizontal. Para esto recurra al menú de Filtrado.
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

6) Ingrese al menú de Segmentación y binarice la imagen para valores de umbral superiores a 80.
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

7) Ingrese al menú Archivo y guarde su resultado en la memoria del computador
   a. ¿Pudo usted llevar a cabo esta tarea? SI:___. NO:___.
   b. ¿Presentó algún error la ejecución de esta tarea? SI:___. NO:___.

8) Ha finalizado con la ejecución de las tareas, ahora proceda a calificar los siguientes enunciados:
   a. Presentación:
      Excelente___ Buena___ Regular___ Mala___ Pésima___

   b. Disposición de los menú:
      Excelente___ Buena___ Regular___ Mala___ Pésima___

   c. Presentación de imágenes:
      Excelente___ Buena___ Regular___ Mala___ Pésima___

   d. Disponibilidad y distribución del material de ayuda:
      Excelente___ Buena___ Regular___ Mala___ Pésima___

   e. Presencia de Glosario:
      Excelente___ Buena___ Regular___ Mala___ Pésima___

Muchas gracias por su tiempo y honestidad la hora de la realización de este estudio.

APÉNDICE D: GUÍA RÁPIDA DE INSTALACIÓN Y MANUAL DE USO DE IMAGEN STUDIO.

GUÍA RÁPIDA DE INSTALACIÓN Y MANUAL DE USO

IMAGEN STUDIO

EMILIO HERRERA

ÍNDICE

REQUERIMIENTOS PARA LA INSTALACIÓN

REQUERIMIENTOS DEL SISTEMA

ARQUITECTURAS SOPORTADAS

- x86 (32 bits)
- x64 (64 bits)

SISTEMAS OPERATIVOS SOPORTADOS

- Windows XP. Servi Pack 2 o superior
- Windows Vista.
- Windows 7.

REQUERIMIENTOS DE HARDWARE

- Mínimo: 1,6 GHz de CPU. 512 MB de RAM. 1024x600 display. 5400 RPM de disco duro.
- Recomendado: 2,2 GHz o superior. 1GB de RAM. 1024 x 600 display o superior. 7200 RPM de disco duro.
- En Vista: 2,4 GHz o superior. 1GB de RAM.

FORMATOS DE IMÁGENES SOPORTADOS

- Bitmap (.BMP)
- Graphics Interchance Format (.GIF)
- Joint Photographic Experts Group (.JPEG o JPG)
- Portable Network Graphics (.PNG)
- Tagged Image File Format (.TIFF)

INSTALACIÓN DEL PROGRAMA

1) Localizar el ejecutable imagen_studio.exe
2) Hacer doble clic en el ejecutable e iniciar la instalación
3) Siga las imágenes a continuación para la correcta instalación, una vez de acuerdo con la información contenida en cada pantalla, hacer clic en *Next*

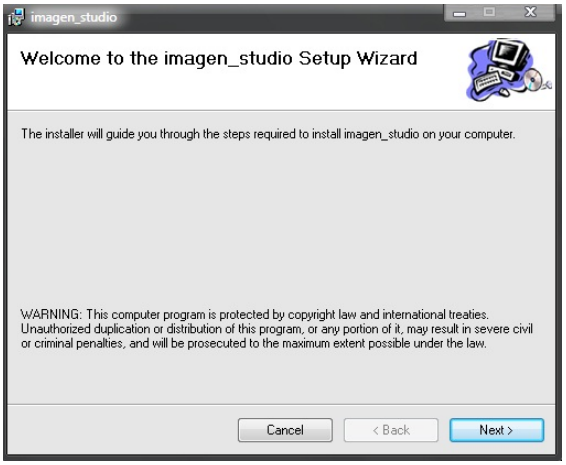


Figura 1. Pantalla de bienvenida al asistente de instalación. Sin parámetros a editar.

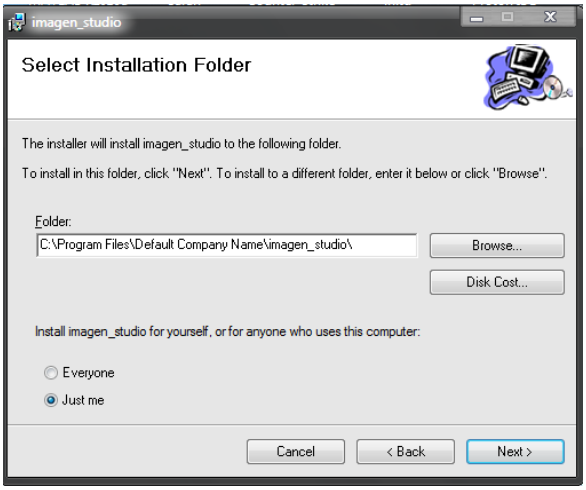


Figura 2. Pantalla para la selección de la carpeta donde se va almacenar el programa.

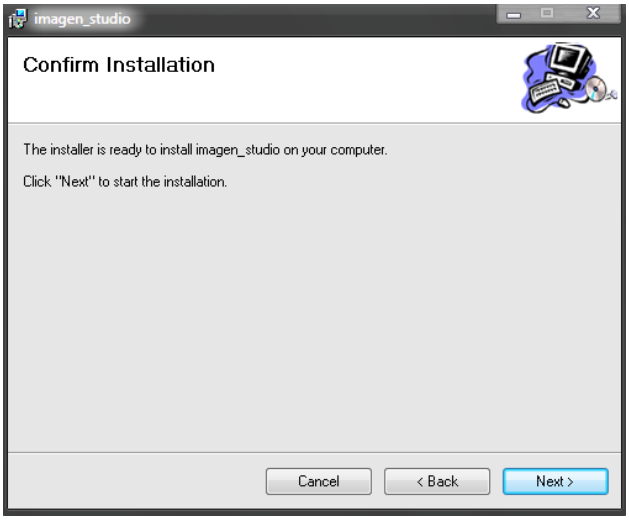


Figura 3. Pantalla de confirmación de la instalación.

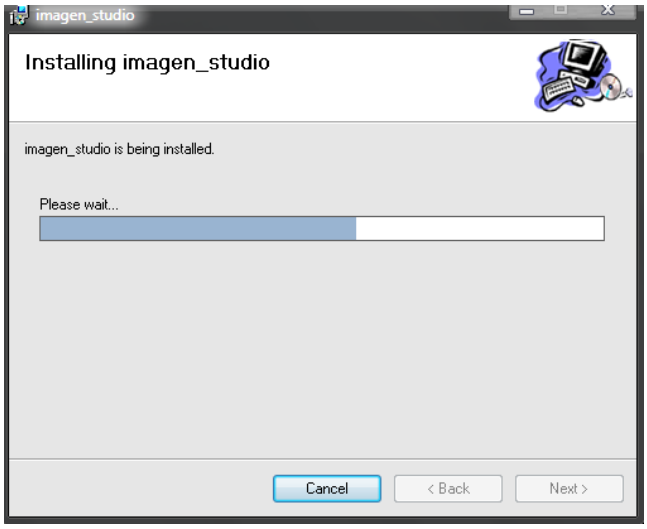


Figura 4. Pantalla durante la instalación del programa.

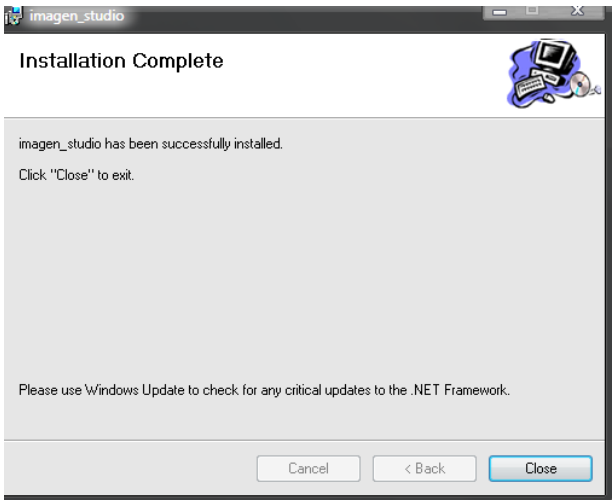


Figura 5. Pantalla de finalización de la instalación del programa.

A partir de este momento se tendrá acceso a la aplicación a través de un ícono ubicado en el escritorio del usuario o en el menú de Inicio de Windows.

INICIANDO IMAGEN STUDIO

Hacer doble click en el ícono de Imagen Studio para acceder al software

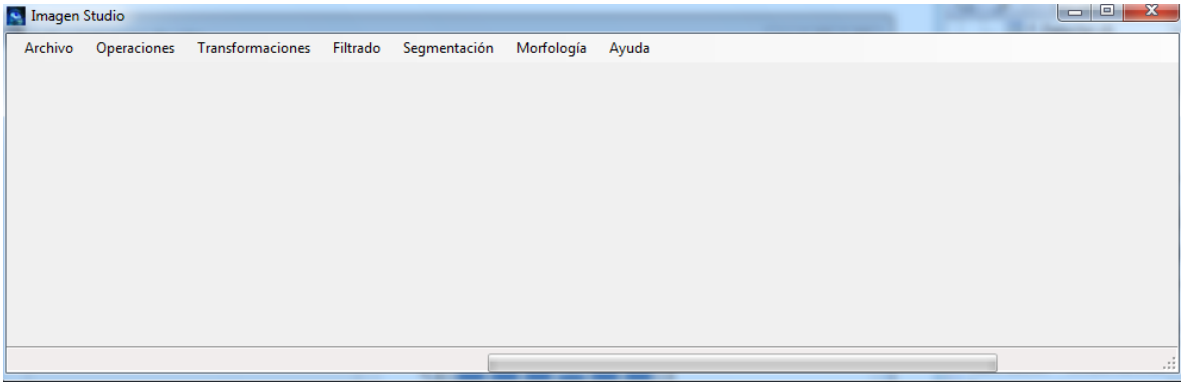


Figura 6. Pantalla de inicio Imagen Studio.

ABRIR IMAGEN

Para poder abrir una imagen, siga las siguientes instrucciones:

1) Archivo / Abrir.
2) Seleccionar la imagen y el tipo de formato de la imagen.
3) Aceptar.

GUARDAR IMAGEN

Para poder guardar una imagen, siga las siguientes instrucciones:

1) Archivo / Guardar
2) Seleccionar la dirección donde se guardará la información, el formato de compresión de la imagen y el nombre con el cual será almacenada.
3) Aceptar

OPCIONES DISPONIBLES EN IMAGEN STUDIO

A continuación un despliegue de los menús disponibles y el requerimiento para poder llevar a cabo la operación requerida.

OPERACIONES

Pintar matriz

Aritméticas

- Archivo
  - Abrir Imagen Primaria
  - Abrir Imagen Secundaria
  - Guardar resultado
  - Cerrar Operaciones
  - Salir
- Operaciones
  - Sumar
    - **Requerimientos**: Imágenes de cualquier tipo soportado.
  - Restar
    - **Requerimientos**: Imágenes de cualquier tipo soportado.
  - Multiplicar
    - **Requerimientos**: Imagen secundaria bitonal (Binarizada).

TRANSFORMACIONES

Requerimientos: Imágenes de cualquier tipo soportado.

- Inversión de colores
- Escala de grises
- Histograma
  - Graficar
  - Modificar el contraste

Función Inversa

Función Cuadrada

Función Cúbica

Función Raíz Cuadrada

FILTRADO

Requerimientos: Imágenes a escala de Grises

Suavizado

1/9 x Identidad

1/10 x Identidad (2)

1/16 x Identidad (+2)

Detección de bordes

Gradiente direccional

Sobel

Prewitt

Personalizar

3x3

5x5

SEGMENTACIÓN

Requerimientos: Imágenes a escala de Grises

Binarización

Mayores al umbral

Menores al umbral

Comprendidos entre 2 valores

MORFOLOGÍA

Requerimientos: Imágenes bitonales (Binarizadas). Preselección de un elemento estructural para así poder realizar la operación morfológica.

- Elemento estruturante
  - 3x3
    - Cuadrado
    - Cruz
    - Equis
    - Horizontal
    - Vertical
    - Diagonal
      - Izquierda
      - Derecha
    - Personalizar
  - 5x5
    - Cuadrado
    - Cruz
    - Equis
    - Rombo
    - Diagonal
      - Izquierda
      - Derecha
    - Personalizar
- Dilatación
- Erosión
- Apertura
- Cierre

Nuevamente, para mayor información sobre las opciones descritas ingresar al menú de ayuda de Imagen Studio.

De esta manera concluye la guía rápida de instalación y manual de uso de Imagen Studio.

NOTA DEL AUTOR

El siguiente trabajo es un folleto que contempla los requerimientos para la instalación de Imagen Studio. Incluye asesoramiento en el proceso de instalación y las nociones básicas para iniciar en el mundo del procesamiento digital de imágenes. No se profundiza en el extenso abanico de posibilidades que brinda este software, ya que en la parte de ayuda del mismo satisface todas las inquietudes no cubiertas en este material.

APÉNDICE E: APLICACIONES INDUSTRIALES.

A continuación se mencionan algunos algoritmos de Visión por Computador aplicados a nivel industrial y los resultados obtenidos con el software diseñado.

INSPECCIÓN DE LA INTEGRIDAD DE LOS ENVASES DE ALIMENTACIÓN

Con base a resolución de la cámara, se permite el estudio y análisis de los defectos en los envases de producción.

La imagen siguiente muestra la vista superior de un envase sin desperfectos.

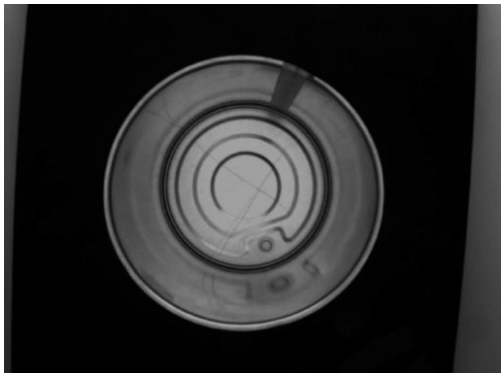


Figura A.12. Visualización de la parte superior de un envase de alimentos enlatados sin desperfectos.

Fuente: *"Visión Artificial"*. [24]

La imagen siguiente muestra un envase con desperfectos en la parte central del envase.

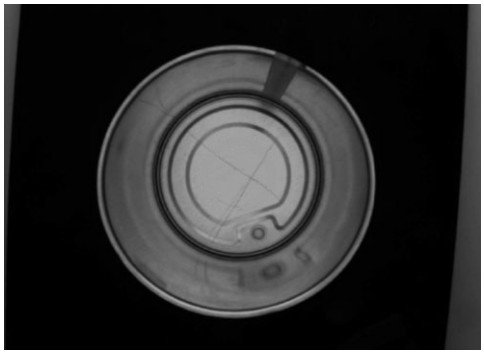


Figura A.13. Visualización de la parte superior de un envase de alimentos enlatados con desperfectos.

Mediante el empleo del algoritmo de Sobel se determina el borde cada imagen por separado, obteniendo las imágenes siguientes:

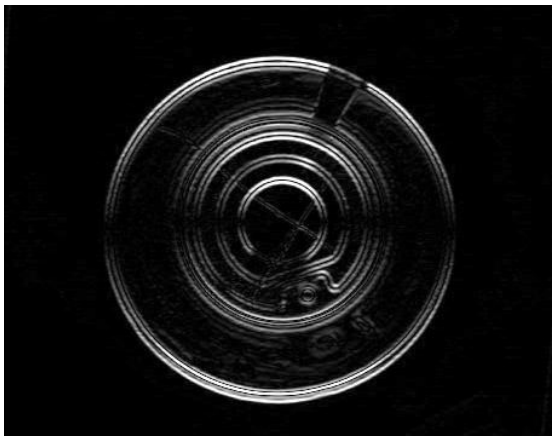


Figura A.14. Visualización de los bordes de un envase de alimentos enlatados sin desperfectos.

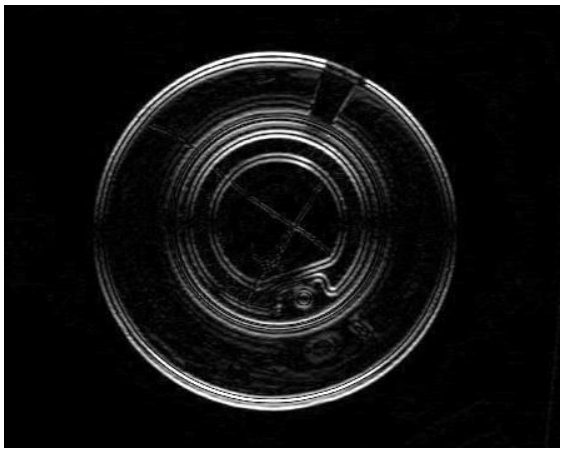


Figura A.15. Visualización de los bordes de un envase de alimentos enlatados con desperfectos.

La diferencia entre la imagen sin desperfectos y la estudiada determina el sector donde ocurre el desperfecto, en caso de no existir daño en la integridad del envase la imagen resultante sería un recuadro negro.

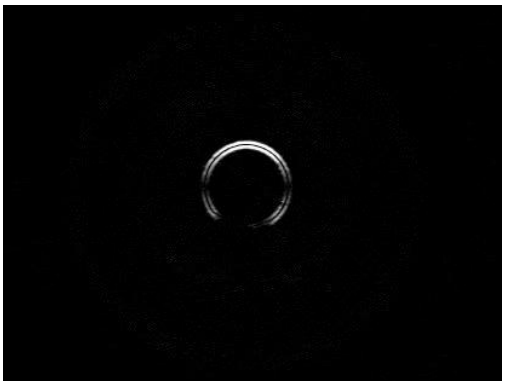


Figura A.15. Visualización de la zona defectuosa en el envase producido.

Se puede notar en la Figura A.15. Que existe un desperfecto en la parte central del envase de producción estudiado, ahora es momento de tomar las acciones correctivas en la línea de producción.

INSPECCIÓN DE LA ESTRUCTURA DE LAS GALLETAS MEDIANTE EL EMPLEO DE VISIÓN ARTIFICIAL

Figura A.16. Línea de producción de galletas.

Fuente: *"Inspección de galletas con Visión Artificial"*. [25]

La industria alimentaria se ha vuelto considerablemente competitiva y los consumidores esperan un alto estándar de calidad y presentación de los productos. El consumidor que compra un paquete de galletas, espera todas tengan el mismo tamaño, peso y calidad.

El principal objetivo del fabricante es optimizar la fabricación y garantizar la calidad tanto del producto como del embalaje.

Con el uso de la visión por computador se puede garantizar el tamaño especificado mediante el uso de las siguientes técnicas:

Vista de producción esperada

Figura A.17. Producción de galletas esperadas.

Fuente: *"Inspección de galletas con Visión Artificial"*. [25]

Vista de producción obtenida

Figura A.18. Producción de galletas obtenidas.

Diferencias existente entre ambas imágenes en escala de grises

Figura A.19. Visualización de las diferencias existentes entre la producción deseada y la obtenida.

Nótese que no se puede apreciar fácilmente las zonas defectuosas del producto, para esto se recurre a la segmentación de la imagen para la localización de las regiones de interés mediante de la binarización de la imagen para cualquier pixel con una profundidad de color superior a 1

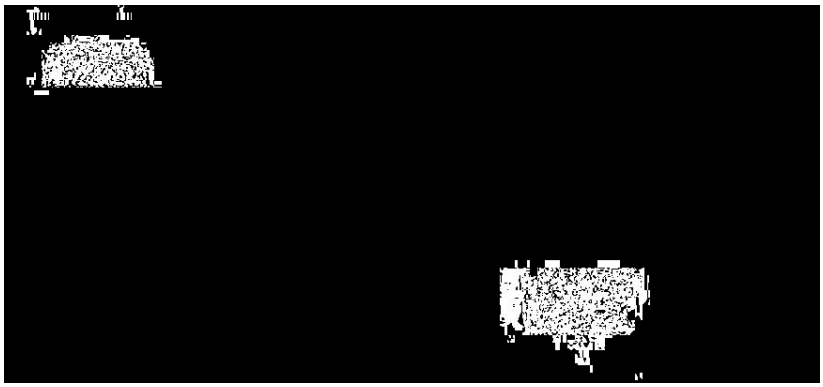


Figura A.20. Segmentación de la imagen donde radican las diferencias existentes entre la producción deseada y la obtenida.

Para finalizar con el análisis de la producción, se procede a determinar los sectores defectuosos en la línea de producción mediante la suma de la línea de producción actual con las fallas encontradas.

Figura A.21. Localización de las partes defectuosas de las galletas producidas.

CONTROL DE CALIDAD PARA LA FABRICACIÓN DE BOTELLAS

Para evitar pérdidas, las botellas vacías son inspeccionadas antes del llenado, para verificar deformidades y desgastes. Cualquier botella vacía dañada en el transporte tiene que ser detectada y separada de la línea de producción.

Figura A.22. Línea de producción de botellas.

Fuente: *"Botellas de agua vacías en una línea de producción en la fábrica"*. [27]

Figura A.23. Línea de producción de botellas vacías en la fábrica.

Fuente: *"Botellas de agua vacías en una línea de producción en la fábrica"*. [27]

Para ello, puede emplearse la siguiente secuencia de pasos para la detectar las botellas desperfectas.

Vista de la producción deseada

Figura A.24. Visualización de la línea de producción esperada de botellas vacías.

Fuente: *"Botellas de agua vacías en una línea de producción en la fábrica"*. [27]

Producción obtenida

Figura A.25. Visualización de la línea de producción de botellas obtenida.

Para iniciar el análisis se procede a la detección de bordes de ambas producciones mediante el algoritmo de Prewitt en la dirección horizontal

Figura A.26. Visualización del contorno de la producción esperada.

Figura A.27. Visualización del contorno de la producción obtenida.

Luego, se calculan las diferencias entre ambas imágenes

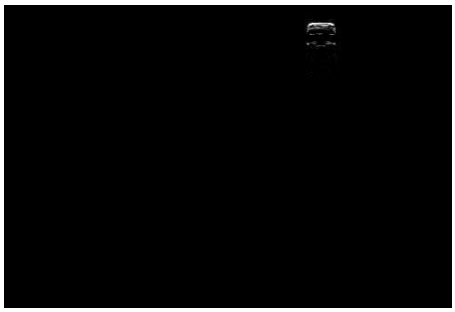


Figura A.28. Visualización del contorno del sector defectuoso en la botella.

Se lleva a cabo una inversión de colores

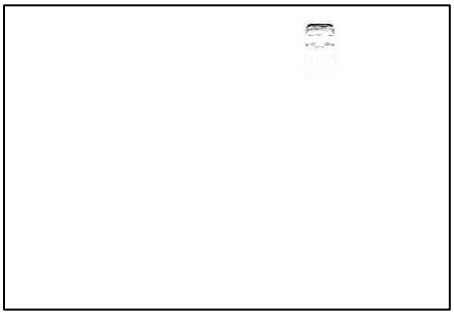


Figura A.29. Visualización de la inversión de colores de la Figura A.28.

Finaliza el estudio de las imágenes con la detección de la zona defectuosa y se toman las acciones correctivas correspondientes.

Figura A.30. Visualización del sector defectuoso en la producción y la localización del error.

ESTUDIO Y LOCALIZACIÓN DE REGIONES DE INTERÉS EN PLACAS AUTOMOTRICES

El estudio de las placas tiene como objetivo la extracción de información de sectores específicos para su posterior estudio mediante el uso de programas especializados en el reconocimiento de caracteres para llevar a cabo la ubicación de un vehículo especifico. Para realizar el objetivo propuesto se plantea el siguiente algoritmo

Imagen a estudiar

Figura A.31. Visualización de la estructura básica en una placa automotriz.

Erosión de la imagen con un elemento estructurante cuadrado de 5 por 5 pixeles

Figura A.31. Visualización del resultado obtenido posterior a la erosión de la imagen con un elemento estructurante definido como un cuadrado de 5*5.

Selección de la región con menor área

Figura A.32. Visualización del resultado obtenido después de la selección de la región con menor área.

Resultado de la suma de la Figura A.31 con la Figura A.32

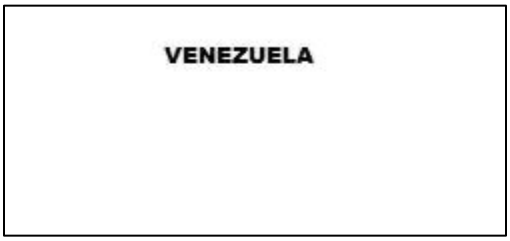


Figura A.33. Visualización sólo de la palabra “VENEZUELA” en la placa.

Cálculo de inversión de colores de la Figura A.32

Figura A.34. Visualización del resultado obtenido posterior a la inversión de colores de la Figura A.32.

Resultado de la suma de la placa original con la figura anterior

Figura A.35. Visualización sólo de los caracteres que conforman la placa.

De esta forma, finaliza el procesamiento de imagen y se procede a estudiar esta nueva imagen con programas dedicados al reconocimiento de caracteres alfanuméricos para su posterior estudio.

DELIMITANTES DE LA INVESTIGACIÓN

1.1.PLANTEAMIENTO DEL PROBLEMA

El procesamiento digital de imágenes es el conjunto de técnicas que se aplican a una imagen digital con el fin de mejorar la calidad o facilitar la búsqueda de información en la misma. Este es un punto base para la creación de sistemas de recreación o de visión artificial para el desarrollo de aplicaciones más complejas.

Mediante el procesamiento digital de imágenes se pueden diseñar e implementar aplicaciones como fotogrametría, que es la realización de mediciones en espacios en tercera dimensión (3D) a partir de fotografías tomadas [1]. También el procesamiento de imágenes cumple roles con la finalidad de medir construcciones, objetos, áreas, entre otras variables. Otro ámbito de interés sería la reconstrucción de imágenes 3D a partir de las imágenes proyectadas en sus vistas [2]. Además de las aplicaciones mencionadas con anterioridad, el matching (asociación de pixeles mediante un criterio definido) y el tracking (seguimiento del comportamiento de un grupo de pixeles) es una de las aplicaciones que encontramos en cámara digitales y teléfonos inteligentes (smart phones), en donde estas técnicas buscan encontrar la correspondencia entre puntos de varias imágenes y como resultado vemos como un dispositivo móvil es capaz de reconocer un rostro y demostrar el movimiento que se realiza en tiempo real. El procesamiento digital de imágenes tiene un sinfín de aplicaciones y muchas de ella rozan los límites de la imaginación.

Este es un ámbito creciente en la tecnología actual, para esto el siguiente trabajo de grado busca desarrollar una aplicación para sistemas de visión artificial para robots industriales, basada en el realce y procesamiento morfológico de imágenes digitales con el

fin de proporcionar una herramienta para impartir la asignatura Robótica y Visión Industrial.

Además, este trabajo busca fomentar la creatividad para la elaboración de algoritmos para nuevas y diversas aplicaciones a través del aprendizaje mediante el uso de un entorno de gráfico que con la ayuda de un manual de usuario, permite acceder y editar imágenes de diversos tipos en la comodidad de su computador para así observar cómo los robots industriales perciben un entorno.

Para finalizar este planteamiento, se procede a dejar una serie de ejemplos y los ambientes donde podemos encontrar el procesamiento digital de imágenes para que se aprecie la relevancia de este tema en nuestra vida cotidiana.

- Cámaras digitales, funciones como detector de sonrisa y reconocimiento facial.
- Determinación de pisos térmicos, topografía. Mediante las imágenes satelitales se puede determinar la existencia de vegetación y que altura con respecto a otras superficies se encuentra.
- Detección de picos, embotelladoras. Mediante tratamiento básico de imágenes que puede desarrollar un algoritmo que tome una imagen de una botella y determine si esta es una botella completa o una botella defectuosa.
- Reconocimiento de iris y huella dactilar, sistemas biométricos. Generalmente empleados para la seguridad a la hora de acceder a sitios privados.

1.2.JUSTIFICACIÓN

En este Trabajo Especial de Grado se desarrollarán aplicaciones dedicadas al procesamiento y realce de imágenes digitales. Este conjunto de aplicaciones están dirigidas al ámbito de visión artificial de robots industriales. Estas aplicaciones estarán contempladas en una interfaz gráfica para que sea de fácil acceso y sencillo uso a los usuarios.

Hasta la fecha, en nuestra ilustre Universidad de Carabobo, no existe una aplicación desarrollada que desempeñe actividades para el procesamiento digital de imágenes con fines exclusivos al ámbito de la robótica industrial en el área de la visión artificial. Por eso este trabajo especial de grado busca innovar con el desarrollo de este conjunto de subrutinas que se encontrarán en un ambiente gráfico y que junto al manual de uso, permitirán al usuario editar imágenes mediante algoritmos conocidos y tendrá una noción sobre cómo se les otorga sistemas de visión a los robots industriales y de qué manera éstos perciben a un entorno de trabajo.

Existen numerosas librerías para el procesamiento de imágenes en el dominio público, entre las cuales se mencionan las siguientes: *Cromada CML,* librería destacada por su variedad de formatos soportados y las aplicaciones como filtrado, manipulación de brillo y contraste, entre otros aspectos [3]. *Codejock Sotfware Xtreme Toolkit*, una librería que permite elaborar controladores gráficos para aplicaciones implementadas en Visual Studios™ e Internet Explorer™ [4]. *VTK. The Visualization Toolkit,* un sistema subrutinas de procesamiento de imágenes y visualización 3D con entornos gráficos disponibles para Java y Phyton [5].

Nótese que de las librerías mencionadas en el punto anterior, sólo una de ellas permite entornos gráficos, y están disponibles en diversos lenguajes de programación. Pero este trabajo es diferente, este trabajo busca dar al usuario un entorno ameno y agradable, de fácil manejo y con la sencilla aplicación de un “clic”, este archivo ejecutable se instale en los computadores y ofrezca un pequeño laboratorio donde el usuario pueda editar y acceder

a las imágenes, y así entienda que los robots también pueden tener una concepción cercana a lo que nosotros denominamos observación.

El perfil de todo ingeniero electricista exige que éste sea una persona calificada para desarrollarse en cualquier ámbito tecnológico del mundo actual. El perfil del ingeniero electricista con mención automatización y control demanda que éste sea una persona capaz de brindar soluciones creativas a los problemas y evitando a toda costa la influencia de los errores posibles causado por agentes externos, de esta forma se considera que a través del procesamiento digital de imágenes se dará solución a diferentes problemas a través del estudio del este tema.

1.3.OBJETIVOS

1.3.1. OBJETIVO GENERAL

Desarrollar una aplicación para sistemas de visión por computador de robots industriales, basada en el realce y procesamiento morfológico de imágenes digitales.

1.3.2. OBJETIVOS ESPECÍFICOS

- Crear subrutinas para el pre-procesamiento de imágenes digitales.
- Desarrollar subrutinas para el filtrado de imágenes digitales en el dominio espacial.
- Desarrollar subrutinas para llevar a cabo transformaciones morfológicas sencillas.
- Desarrollar subrutinas para la segmentación y detección de bordes de imágenes digitales.
- Unificar las subrutinas en un ambiente gráfico que incluya su respectivo manual de uso.
- Evaluar la aplicación con criterios de usabilidad y precisión.

1.4. ALCANCE

Por medio de la realización de este Trabajo Especial de Grado, se logra recopilar información teórica y práctica del procesamiento digital de imágenes y se fundamenta dicha información con abundancia de ejemplos, además del desarrollo un entorno gráfico que quedará para su uso posterior.

Este Trabajo desarrolla algunos de los algoritmos más empleados para el procesamiento digital de imágenes entorno a aplicaciones correspondientes a la robótica, específicamente a la parte de visión por computador.

Contempla por lo tanto lo siguientes tópicos:

- Operaciones aritméticas y morfológicas.
- Manipulación del contraste.
- Realce y detección de bordes.
- Segmentación de imágenes.
- Transformaciones morfológicas básicas.

Por la índole de este Trabajo Especial de Grado, se procede a especificar los formatos de compresión de las imágenes que son soportadas por el software. Estos formatos aplican para la entrada de imágenes y también para la salida posterior a su procesamiento. Estos formatos son los comúnmente empleados bajo plataformas de Windows® 32 bits y 64 bits:

- Bitmap (.BMP)
- Graphics Interchance Format (.GIF)
- Joint Photographic Experts Group (.JPEG o JPG)
- Portable Network Graphics (.PNG)
- Tagged Image File Format (.TIFF)

Este Trabajo también incluye un manual para los usuarios donde se busca aclarar todas las dudas referentes al manejo y uso de la aplicación. Se culmina con la presentación de un archivo ejecutable. Este archivo debe ser capaz de instalar en los computadores la aplicación que contendrá las subrutinas desarrolladas, de esta manera el usuario simplemente tendrá que manipular con el entorno gráfico para empezar a procesar imágenes digitales.

ANTECEDENTES

No existe una aplicación desarrollada por estudiantes de la facultad de Ingeniería que contenga un manual de usuario y que esté dedicada al procesamiento de imágenes digitales en la categoría de la Visión Artificial que además brinde un soporte didáctico en la Universidad de Carabobo. Sin embargo existen otros trabajos para el procesamiento de imágenes realizados en la ilustre Universidad de Carabobo, los cuales se hacen referencia en orden cronológico.

Abril 2005. Proyecto de grado titulado “Diseño e implantación de un software de reconocimiento de huellas dactilares.” Por los bachilleres Edgar Alam y Danilo Guglielmetti

Octubre 2006. Proyecto de grado titulado: “Diseño e implantación de un sistema de adquisición y procesamiento de imágenes provenientes de la respuesta fotomotora del iris”. Realizado por los bachilleres Carolina López y Roberto Chumbimuni

Abril 2008. Proyecto titulado: “Calibración del localizador magnético para adquirir imágenes 3D de objetos sencillos para utilizarlos en una escena médica virtual”. Hecho por los bachilleres Rafael Carmona y Leonel Ríos.

Julio 2010. Proyecto Titulado: “Reconstrucción de los ventrículos cerebrales usando en método de segmentación level set implementado en ITK” Por la bachiller María Fernanda Vásquez.

Nótese que todos estos trabajos mencionados, hacen entrega de un software con un entorno gráfico, la diferencia con este trabajo es en la instalación del software ajeno al propiamente desarrollado para su correcta ejecución.

Todos estos trabajos previamente realizados a pesar de ser un aporte indirecto a esta investigación, forma parte fundamental de la misma ya nos muestra el campo de aplicación de esta investigación y deja constancia de las aplicaciones en se pueden aplicar.

Continuando con la mención de los trabajos que fundamentan el contenido de esta línea de investigación se hace referencia a muchísimas otras librerías escritas por diversos desarrolladores, donde se debe hacer mención al software *Harpia*, Harpia es un software diseñado e implementado por desarrolladores de opensource para el proyecto GNU/Linux. Este software es una interfaz gráfica que permite al usuario diseñar sus propios algoritmos de procesamientos de imágenes e implementarlo mediante las librerías diseñadas en C que dispone el programa.

Otro trabajo digno de mención es el realizado por Gómez F.J., Fernández-Caballero A y López M.T., en la Universidad de Castilla-La Mancha. España. Donde se desarrolló una librería para el procesamiento de imágenes llamada *CImagen*. Esta librería fue desarrollada en C++ y es empleada en tareas de investigación y docencia en el Departamento de Informática de la Universidad de Castilla-La Mancha.

De esta manera se puede apreciar como estos trabajos previos establecen un precedente para continuar realizando investigaciones en área de procesamiento digital de imágenes, punto neurálgico para esta investigación.

BASES CONCEPTUALES

3.1. IMÁGENES DIGITALES

Antes de iniciar este viaje a través del mundo de la visión artificial, hay que tener nociones sobre conceptos que se encontraran a lo largo de este trabajo.

El primer concepto que hay que tener en cuenta es la definición de píxel o pixel, que es la menor unidad de color que puede ser apreciada en una imagen. El origen de la palabra píxel proviene del acrónimo del inglés *picture element* (elemento de imagen). [10]

Cuando se nota la en presencia de una sucesión de pixeles (plural de píxel), se encuentra en presencia de una imagen digital. Según esto, se puede definir a una imagen digital como una representación visual de un objeto iluminado por una fuente radiante que viene dada como una secuencia ordenada de píxeles [6].

Para la formación de una imagen digital intervienen los siguientes elementos: el objeto, la fuente emisora de luz y el sistema para la formación de la imagen, que básicamente consiste en un sistema óptico, un sensor y un dispositivo digitalizador.

Una vez llevado a cabo el proceso de la formación de la imagen está viene expresada de la siguiente forma:

$$I_{m*n} = \begin{bmatrix} I_{11} & I_{12} & I_{13} & & I_{1(n-2)} & I_{1(n-1)} & I_{1n} \\ I_{21} & I_{22} & I_{23} & \cdots & I_{2(n-2)} & I_{2(n-1)} & I_{2n} \\ I_{31} & I_{32} & I_{33} & & I_{3(n-2)} & I_{3(n-1)} & I_{3n} \\ \vdots & & & \ddots & & \vdots & \\ I_{(m-2)1} & I_{(m-2)2} & I_{(m-2)3} & & I_{(m-2)(n-2)} & I_{(m-2)(n-1)} & I_{(m-2)n} \\ I_{(m-1)1} & I_{(m-1)2} & I_{(m-1)3} & \cdots & I_{(m-1)(n-2)} & I_{(m-1)(n-1)} & I_{(m-1)n} \\ I_{m1} & I_{m2} & I_{m3} & & I_{m(n-2)} & I_{m(n-1)} & I_{mn} \end{bmatrix}$$

Figura 3.1. Representación matricial de una imagen digital

Donde I_{m*n} es la matriz que se conforma por los valores de los píxeles correspondientes a la imagen, de esta manera se conforma la imagen digital, *m* representa el número de pixeles de largo y *n* en número de pixeles de ancho de la imagen.

Para las imágenes en los dispositivos digitales, a cada píxel se le codifica mediante la asignación de bits de longitud predeterminada (a este proceso se le denomina profundidad de color), como por ejemplo: se puede asignar a un píxel un profundidad de un byte, lo equivalente a 8 bit. Esto implica que nuestro píxel admitirá 256 variaciones de colores o lo que es lo mismo 2^8 posibilidades binarias, lo cual permite al bit tener valores desde 0 hasta 255 [7]. Esta es la codificación que se tendrá a lo largo del desarrollo de este trabajo.

A manera informativa, también se hace referencia brevemente las imágenes *true colors*, (traducción al inglés de verdadero color) las cuales poseen una profundidad de color de 3 bytes. Con esta profundidad de color se obtiene una variación de 16.77.216 posibilidades de color por cada píxel.

El ojo humano puede detectar aproximadamente la misma cantidad de variaciones de color, por eso el nombre particular de esta profundidad de color en particular.

3.2. MUESTREO ESPACIAL Y RESOLUCIÓN

Una señal es una función que depende de una o más variables que tienen a su vez un significado físico asociado. Dependiendo del número de variables asociadas, se pueden clasificar en unidimensional para una variable, bidimensional para dos variables y tridimensional para el caso de tres variables.

Con base al número de dimensiones, la función puede ser escalar o vectorial. Las imágenes a color son funciones vectoriales ya que poseen tres componentes, mientras que las imágenes a blanco y negro son funciones escalares debido a que dependen se una sola componente.

Como se mencionó anteriormente, una imagen es una representación de los valores de los píxeles que conforman la imagen digital. El concepto de muestreo implica a la conversión que sufren las dimensiones espaciales de la señal que genera la noción de píxel [8].

Establecido el concepto de muestreo, se procede hacer referencia a la resolución, que es el número de muestras tomadas para llevar a cabo el establecimiento de la imagen digital.

3.3. RELACIONES ENTRE PIXELES

3.3.1. VECINDAD

Un pixel con coordenadas (x, y) presenta un total de cuatro vecinos, dos correspondientes a la vertical y los restantes al horizontal, siendo sus coordenadas las siguientes.

	x, y-1	
x-1, y	x,y	x+1,y
	x,y+1	

Figura 3.2. Representación 4-vecinos de un píxel de coordenadas (x,y)

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 66. [8]

Este conjunto de pixeles se denomina vecindad tipo 4 del píxel *p*, y se representa por N4(*p*). Además se puede considerar la existencia de otros 4 vecinos asociados a las diagonales cuyas coordenadas son:

x-1,y-1		x-1,y-1
	x,y	
x+1,y+1		x+1,y+1

Figura 3.3. Representación 4-vecinos diagonales de un píxel de coordenadas (x,y)

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 66. [8]

Los cuales se representan por ND(p). La suma de los anteriores definidos, conforman los ocho vecinos del pixel *p* también denominado N8(*p*)

x-1,y-1	x, y-1	x-1,y-1
x-1, y	x,y	x+1,y
x+1,y+1	x,y+1	x+1,y+1

Figura 3.4. Representación 8-vecinos de un píxel de coordenadas (x,y)

3.3.2. CONECTIVIDAD

Mediante este concepto se quiere expresar que dos píxeles pertenecen al mismo objeto, por lo que se está relacionado con el vecino. Dos píxeles que están conectados si están adyacentes (vecinos) y sus niveles de gris satisfacen algún criterio de especificación, como por ejemplo ser iguales.

De manera más formal, se define un par de píxeles *p* y *q*, y un conjunto V que representa a los valores compatibles para establecer la conectividad [9].

V= {Valores de los pixeles que definen la conectividad}

Entonces:

1. Dos píxeles *p* y *q* con valores en V presentan una conectividad 4 si q pertenece al N4(p)
2. Dos píxeles *p* y *q* con valores en V presentan una conectividad 8 si q pertenece al N8(p)
3. Conectividad mixta. Dos píxeles *p* y *q* con valores en V presentan una conectividad mixta si:
   a) Q pertenece a N4(p), o
   b) Q pertenece a ND(p) y el conjunto N4(p)∩N4(q) es el conjunto vacío

La conectividad mixta es una modificación de la conectividad 8 cuya utilidad es eliminar las conexiones múltiples que en ocasiones aparecen cuando la conectividad 8 es utilizada.

Hay que tener en cuenta que es el conjunto V quien define la conectividad. Para imágenes binarias (imágenes donde los pixeles sólo toman valores de 0 o 1), el conjunto V puede ser definido como el conjunto {1} o el conjunto {0}. Para imágenes en niveles de gris, con variaciones de 256 niveles V puede tener diferentes configuraciones según el interés de los niveles de estudio. Como por ejemplo: V= {0,1,2, … ,126, 127} para obtener elementos oscuros, o un conjunto definido por V= {200, 201, … ,254,255} para obtener los elementos más claros. Para imágenes en color se procede de forma similar pero definiendo rangos en los perfiles de color RGB.

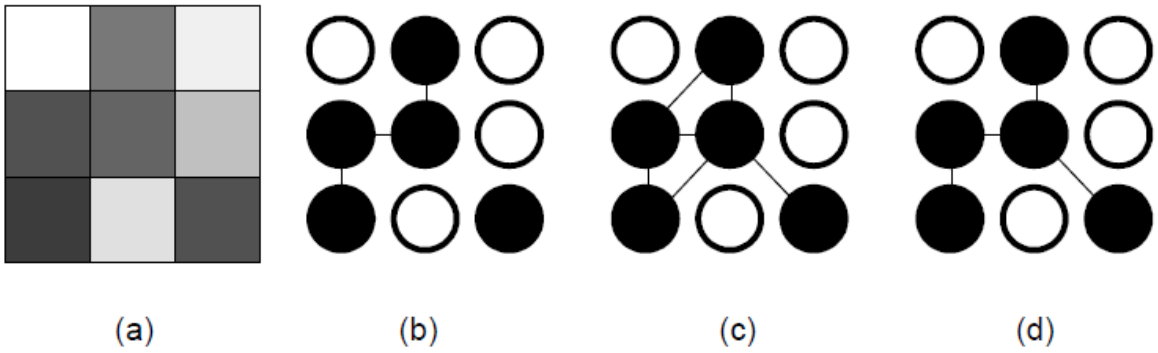


Figura 3.5. Ejemplo de los tipos de conectividad: (a) Corresponde a una imagen en niveles de gris. (b), (c) y (d) representan en negro los píxeles de (a) que están dentro de un V determinado y muestran la conectividad entre ellos: (b) Conectividad 4, (b) Conectividad 8, (d) Conectividad Mixta.

Fuente: "*Visión por computador*". Pág 60.

3.4. EL COLOR

Estas son algunas de las nociones básicas que hay que tener presentes para la comprensión de los espacios de color [10].

Brillo: Sensación que indica si un área está más o menos iluminada.

Tono: Sensación que indica si una superficie se asemeja a un color rojo, azul, verde o una combinación de ellos.

Coloración: Sensación de un área tiene un mayor o menor tono.

Luminosidad: Brillo de una zona respecto a otra blanca de la imagen

Croma: La coloridad de un área respecto al brillo de un blanco de referencia.

Saturación: Relación entre coloridad y brillo.

Para llevar a cabo la cuantificación de la percepción de la visión humana, se han creado distintos modelos y sistemas. Entre los cuales destacan los siguientes:

- Espacio RGB
- Espacio HSI
- Espacios XYZ, Luv, Lab
- Espacio YIQ, YUV, YCbCr, YCC
- Espacio CMY(K)

A continuación se procede a estudiar el espacio de color RGB que es el punto de interés de este trabajo, ya que es el espacio de color utilizado por los sistemas de visión por computador.

3.4.1. ESPACIO RGB

Este se basa en una combinación de tres señales lumínicas distintas, que son rojo, verde y azul (de aquí proviene su nombre; red, green y blue).

La manera más sencilla de conseguir un color es mediante la estimulación de los fotoreceptores sensibles a estos colores, y para determinar un color concreto, sólo basta en determinar la cantidad de rojo, verde y azul que se necesita para formarlo.

A este modelo de color también se le conoce como sistema aditivo, debido a que funciona mediante la sumas de sus colores primarios [11].

$$X= R + G + B$$

Note el siguiente ejemplo. El resultado de la suma de todas las partes posibles rojas, más las verdes, más las azules generan el color blanco.

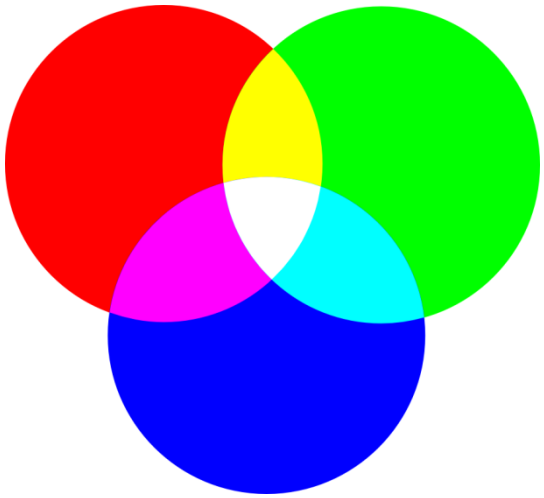


Figura 3.6. Componentes conformantes del espacio de color RBG

Fuente: "Tratamiento digital de imágenes" por José Grimaldo. Preliminares. [12]

Ahora, si se añade solamente pares de colores obtenemos los siguientes colores

Rojo + verde = amarillo

Rojo + azul = magenta

Azul + verde = cian

En este punto ya se ha determinado una gama de 8 colores elementales en el sistema cromático humano y son los siguientes: Blanco, negro, rojo, verde, azul, amarillo, cian y magenta [12].

Como se ha visto con anterioridad, imágenes en escala de grises vienen dadas como un arreglo bidimensional, donde cada píxel varía en función de la profundidad de color. Para el caso de las imágenes de colores se tendrá entonces un arreglo tridimensional, al cual llamaremos Im*n*p. Donde las variables M y N tendrán la misma representación que para el caso de las imágenes es escala de grises, la tercera dimensión (llamada p en nuestro caso) es correspondiente al plano de color [13]. La figura anexa explicará mejor este concepto.

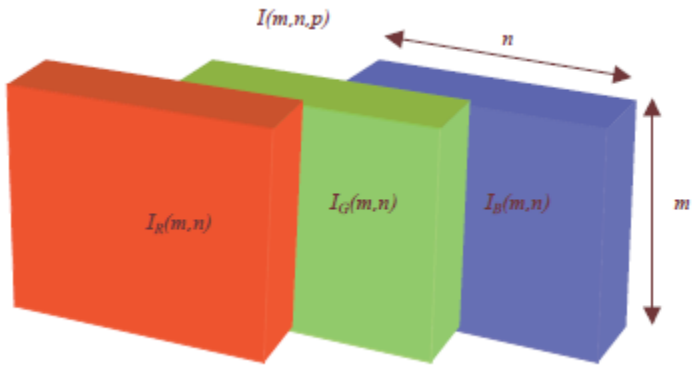


Figura 3.7. Representación de los perfiles Rojo, Verde y Azul que conforman una imagen a color

Fuente: *"Visión por computador utilizando Matlab y el toolbox de procesamiento digital de imágenes"*. Pág 19. [13]

3.5. OPERACIONES ARITMÉTICAS CON IMÁGENES

Las operaciones aritméticas entre dos píxeles son empleadas frecuentemente en el procesamiento digital de imágenes, y las más comunes son las siguientes:

- Suma.
- Resta.
- Multiplicación.

Teniendo en consideración que las imágenes son arreglos matriciales de los valores de la profundidad del color, entonces entiéndase la suma y la resta como la operación

aritmética entre los valores de los píxeles ubicado en la misma posición para 2 imágenes diferentes. Para poder aplicar esta técnica, primero es necesario que las imágenes sean del mismo tamaño, luego una vez llevada la operación hay que tener en cuenta la saturación de la imagen resultante, esto significa que si la suma da un valor superior a una profundidad de color superior a 255 (para nuestra profundidad de color de estudio), la imagen resultante debe tener una valor para esa posición de 255.

Este caso generalmente se obtiene cuando se implementa la suma de imágenes, el caso análogo para la resta es cuando los valores son inferiores a 0, donde hay que tener en cuenta que el mínimo valor para la profundidad de color es 0.

Una vez establecidas las analogías entre la suma de imágenes y la suma de matrices, hay que aclarar que la multiplicación de imágenes es una multiplicación píxel con píxel bajo la misma posición y no como se efectuaría una multiplicación de matrices. Otra consideración importante a la hora de la multiplicación de imágenes, que es una de estas debe ser una imagen bitonal (binaria), caso contrario el resultado de la multiplicación de 2 imágenes siempre incurriría en una saturación de la profundidad del color.

3.6. TRANSFORMACIONES MATEMÁTICAS

3.6.1. CONVOLUCIONES [8]

La convolución es una operación matemática que suma una función *f* consigo misma repetidas veces en todo el dominio de otra función *h*, empleando cada suma como un valor de escala el valor de *h* en ese punto del dominio.

Para el caso unidimensional, se define como Convolución de una función *f*(x) respecto a la función *h*(x) una nueva función con tal que:

$$g(x) = h(x) * f(x) = \int_{i=-\infty}^{i=\infty} f(i).h(x-i)di \qquad (1)$$

Si se tratara del caso discreto con dos secuencias *f*(x) y *h*(x) se obtendría una secuencia de salida *g*(x):

$$g(x) = h(x) * f(x) = \sum_{i=-\infty}^{i=\infty} f(i).h(x-i) \tag{2}$$

Para el caso de las imágenes digitales se usan convoluciones bidimensionales discretas, cuya ecuación es la siguiente:

$$g(x,y) = h(x,y) * f(x,y) = \sum_{i=-\infty}^{i=\infty} \sum_{j=-\infty}^{j=\infty} f(i,j).h(x-i,y-j) \tag{3}$$

Lo más normal es usar convoluciones de 3*3 elementos. Entonces la Ecuación (3) puede concretarse en:

$$g(x,y) = h(x,y) * f(x,y) = \sum_{i=0}^{i=2} \sum_{j=0}^{j=2} f(i,j).h(x-i,y-j) \tag{4}$$

Que por ejemplo, para el caso de Ecuación (4) evaluada en el punto (2,2) será:

$$g(2,2) = \sum_{i=0}^{i=2} \sum_{j=0}^{j=2} f(i,j).h(2-i,2-j) =$$

$$f(0,0).h(2,2) + f(0,1).h(2,1) + f(0,2).h(2,0) +$$

$$f(1,0).h(1,2) + f(1,1).h(1,1) + f(1,2).h(1,0) +$$

$$f(2,0).h(0,2) + f(2,1).h(0,1) + f(2,2).h(0,0)$$

En el caso de visión por computador, en lugar de hablar de convolucionar la imagen *f*(x,y) con la transformación *h*(x,y), se habla de convolucionar con la máscara que tiene la forma siguiente:

H(0,0)	H(0,1)	H(0,2)
H(1,0)	H(1,1)	H(1,2)
H(2,0)	H(2,1)	H(2,2)

Figura 3.8. Representación de la máscara de convolución *h* (i,j)

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 85

Para obtener *h*(-i,-j) ha y que realizar 2 giros. De esta manera se obtiene *h*(i, -j)

H(2,0)	H(2,1)	H(2,2)
H(1,0)	H(1,1)	H(1,2)
H(0,0)	H(0,1)	H(0,2)

Figura 3.9. Representación de la máscara de convolución girada, *h* (i,-j)

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 85

Y finalmente *h*(-i,-j)

H(2,2)	H(2,1)	H(2,0)
H(1,2)	H(1,1)	H(1,0)
H(0,2)	H(0,1)	H(0,0)

Figura 3.10. Representación de la máscara de convolución girada, *h* (-i,-j)

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 86

La porción correspondiente de la imagen es:

f(0,0)	f(0,1)	f(0,2)
f(1,0)	f(1,1)	f(1,2)
f(2,0)	f(2,1)	f(2,2)

Figura 3.11. Representación correspondiente a un sector de una imagen digital.

Fuente: *"Visión por computador. Fundamentos y métodos"*. Pág 86

Y el valor del píxel de la imagen transformada será el mismo que se obtuvo antes por el empleo de la Ecuación 3.

Gráficamente se trata de poner las dos imágenes, una encima de la otra, multiplicar cada celda por la inferior y luego sumar los productos.

En la práctica se suele omitir el cálculo para los píxeles del borde de la imagen, por lo que la imagen convolucionada es más pequeña que la original.

Además, para mantener el resultado de la operación dentro de un rango representable se suele añadir a la expresión anterior un factor de división y después un factor de suma o simplemente una condición que cuando dicha suma exceda su valor, tome por defecto la profundidad de color empleada [14].

Para obtener el siguiente valor de $g(x,y)$ la máscara se desplaza un elemento hacia la derecha y se repite la operación. Al acabar las columnas se vuelve al principio pero de una fila inferior, y así sucesivamente hasta llegar al último píxel.

3.6.2. EJES DE REFERENCIA Y POSICIÓN ESPACIAL

Hasta este momento, sólo se ha hablado de las imágenes y de la posición espacial de los píxeles en coordenadas rectangulares (función de la variable x y la variable y), pero no se ha definido ningún eje de referencia.

Hay que tener en cuenta que a diferencia de lo convencional, el eje de referencia (punto de origen o punto cero (0)) cambia a la parte superior derecha para las imágenes, mientras que por tradicionalismo se está acostumbrado a tomar como referencia la parte inferior izquierda para estudios académicos.

3.7. PREPROCESAMIENTO DE IMÁGENES

3.7.1. AMPLITUD DE LA ESCALA O NORMALIZACIÓN

Consiste en aplicar un mapa de transformaciones para extender éstas cuando la adquisición de la imagen se ha hecho con un rango menor. Por ejemplo, si se captura una imagen con 128 niveles de grises y se tiene que utilizar la posibilidad de utilizar 256, el proceso de normalización consistirá en ampliar el rango de 128 a 256 niveles [6].

Aunque puede parecer que es una operación sobre el píxel, hay que tener en cuenta que para determinar los rangos máximos y mínimos, primero hay que contabilizar todos los puntos para determinar dichos valores, por eso la amplitud de la escala o normalización es una operación que se determina posterior al cálculo del histograma de la imagen.

Para efectuar la normalización se aplica la siguiente ecuación:

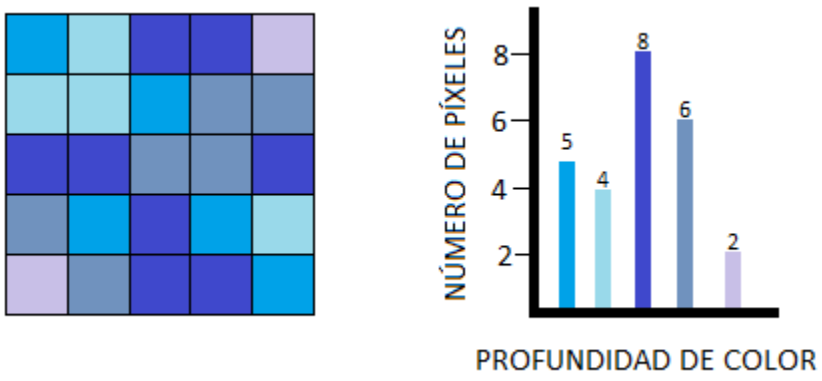
$$IN(x,y) = K\frac{IO(x,y) - K_{mín}}{K_{máx} - K_{mín}} \quad (5)$$

Donde *IN*(x,y) es el valor del pixel posterior a la normalización e *I0*(x,y) es el píxel de la imagen a normalizar, $K_{mín}$ y $K_{máx}$ son los valores mínimos y máximos de la imagen, y K el valor máximo que se puede adquirir el píxel.

3.7.2. HISTOGRAMA

El histograma de una imagen es la representación gráfica del número de pixeles asociados a un valor de profundidad de color de la imagen estudiada.

La imagen siguiente se muestra de manera gráfica como se obtienen los histogramas asociados a una imagen, nótese que en el eje de las ordenadas se encuentran todas y cada una de las profundidades de color existentes en la imagen, y en el eje de las abscisas el número de veces que aparece determinada profundidad.



(a) **(b)**

Figura 3.12. Metodología para la elaboración de un histograma de una imagen. (a) Sector de una imagen de color. (b) Representación grafica del numero de píxeles en función de la profundidad de color

3.7.3. MODIFICACIÓN DEL CONTRASTE [8]

Caso similar al anterior, luego de la obtención del histograma de la imagen, se puede multiplicar por una nueva función de transferencia que permite modificar el contraste de la imagen de una manera sencilla para así garantizar un menor costo computacional. El algoritmo se encarga de sustituir cada nivel de los píxeles de la imagen original por un nuevo valor obtenido según sea la función de transferencia.

Para realizar la modificación hay que definir las funciones de transferencias, entre las cuales las más usuales son:

Función inversa $$p = 255 - m$$

Función cuadrada $$p = \frac{m^2}{255}$$

Función cúbica $p = \frac{m^3}{255^2}$

Función raíz cuadrada $p = \sqrt{255m}$

Donde *p* es el nuevo valor de gris de la imagen y *m* es el valor de gris de la imagen original.

Tenga en cuenta que en las ecuaciones aparece el valor de 255 para normalizar los valores entre los 256 niveles posibles. Si los niveles de gris no fueran de 8 bits habría que colocar el nuevo valor del máximo.

3.8.FILTRADO Y REALCE DE IMÁGENES

En este apartado se tratan las transformaciones que se aplican sobre las imágenes digitales. El objetivo general de esta tarea es mejorar algún sector de información contenida en la imagen, de manera tal de hacer posible la realización de etapas posteriores al tratamiento de la imagen.

Todas las operaciones que se describen a continuación se basan en la teoría de filtros. Un filtro puede definirse como un mecanismo de cambio o transformación de la señal de entrada a la que se le aplica una función, denominada función de transferencia, para así obtener una señal de salida.

Figura 3.13. Representación de un filtro como diagrama de bloques.

Para la visión artificial, básicamente el filtrado de las imágenes se divide en 2 grandes entornos: el filtrado espacial y el frecuencial.

El filtrado espacial consiste en la aplicación de filtros directamente sobre los pixeles de la imagen. En este proceso se relacionan todos los pixeles que conforman la imagen y cada uno de estos se relacionan con sus pixeles más próximos (vecinos) con la finalidad de generar ventanas deslizantes que operen sobre la imagen para así llevar a cabo el proceso de filtrado [13].

Entiéndase por ventana deslizante a fragmentos de la imagen que generalmente son de dimensiones pequeñas. Estas ventanas están conformadas por factores de ponderación que son utilizados para modificar el valor de los pixeles centrales de estas ventanas. Para visualizar o dar a entender mejor este concepto recurra a la imagen siguiente

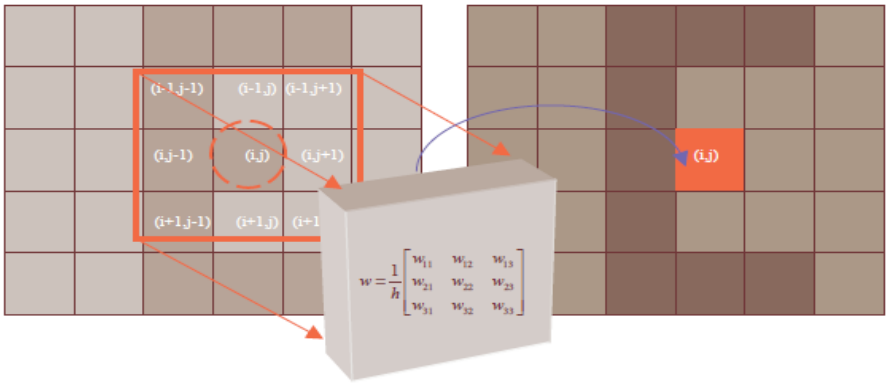


Figura 3.14. Filtrado espacial por el uso de ventanas deslizantes de 3*3

Fuente: *"Visión por computador utilizando Matlab y el toolbox de procesamiento digital de imágenes"*. Pág 17

El filtrado frecuencial es el filtrado que se lleva a cabo en el dominio de la frecuencia, el cual involucra el cálculo de la transformada de Fourier de la imagen (caso de una señal discreta). Posteriormente la aplicación del filtro y se finaliza con el retorno de los valores al dominio espacial [15].

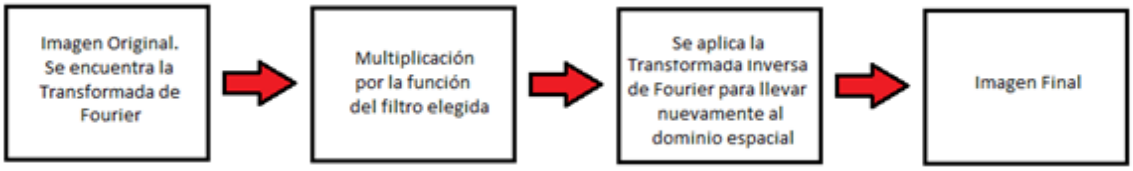


Figura 3.15. Diagrama de bloques para el procesamiento frecuencial.

Tenga presente que este trabajo sólo involucra la visión por computador para aplicaciones diseñadas para robots industriales, por consiguiente el desarrollo de este capítulo se centrará en el filtrado espacial, ya que este es el ámbito más empleado en dichos dispositivos.

En el resto de este desarrollo se describen las principales operaciones que se pueden efectuar sobre las imágenes digitales para sistemas de visión. Las explicaciones se realizarán sobre imágenes en niveles de gris por su simplicidad y debido a que su extensión a las imágenes de color consiste en repetir el tratamiento pero para cada una de las componentes de color.

3.8.1. FILTRADO ESPACIAL

Los filtros espaciales son filtros que se realizan directamente sobre la imagen y por tanto en el dominio del espacio. Aunque hay diferentes tipos de filtros espaciales, los más usados son los *filtros espaciales de convolucion* (Su formulación matemática se encuentra explicada en punto correspondiente a transformaciones matemáticas).

3.8.2. FILTROS DE SUAVIZADO

El filtrado de suavizado espacial se basa en el promediado de los píxeles adyacentes al píxel que se evalúa. Quizás el filtro de suavizado más simple que se puede diseñar se corresponde con una matriz de 3x3 con todos los elementos a 1. El resultado de la convolución de cada píxel se deberá dividir por 9 para asegurar el obtener valores dentro del rango de la profundidad del píxel. En la figura anexa se puede apreciar el resultado de la aplicación de este filtro [9].

(a) (b)

Figura 3.16.(a) Imagen Original en escala de grises. (b) Imagen suavizada. Resultado de la aplicación del filtro mencionado en el texto

Fuente: *"Visión por computador"* pág: 85

"…Las siguientes matrices de convolución definen otros filtros de suavizado…" [9]:

$$h = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figura 3.17. Otros filtros empleados para le suavizado de imágenes digitales.

Fuente: *"Visión por computador"*. Pág 84

3.9.FILTROS DE OBTENCIÓN DE CONTORNO O DETECCIÓN DE BORDES

3.9.1. MEDIANTE GRADIENTE DIRECCIONAL [9]

El cálculo de la derivada direccional de una función permite conocer cómo se producen los cambios en una dirección determinada. Tales cambios suelen corresponder a los contornos de los objetos presentes en las imágenes.

Partiendo de que el operador gradiente se define como:

$$\nabla\big(I(x,y)\big) = \frac{\partial I}{\partial x}\overrightarrow{u_x} + \frac{\partial I}{\partial y}\overrightarrow{u_y} \tag{6}$$

Se definen los filtrados de convolución G_x y G_y:

$$G_x = \frac{\partial I}{\partial x} = I(x,y) * h_1(x,y) \tag{7}$$

$$G_y = \frac{\partial I}{\partial y} = I(x,y) * h_2(x,y) \tag{8}$$

Obteniendo h_1 y h_2 mediante una aproximación a la derivada con la resta. Es decir, si se consideran los píxeles de la siguiente figura:

p_1	p_2	p_3
p_4	p_5	p_6
p_7	p_8	p_9

Figura 3.18. Ubicación espacial de los píxeles que definen una ventana deslizante o máscara de convolución de 3x3.

Las derivadas serían:

$$\frac{\partial I}{\partial x} = p_5 - p_6$$

$$\frac{\partial I}{\partial y} = p_5 - p_8$$

Sustituyendo en (7) y (8) se deduce que las matrices de convolución h_1 y h_2 serán:

$$h_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Figura 3.19. Filtros para el cálculo del gradiente direccional.

Por ejemplo la matriz h_1 proporciona un filtrado en el que un cambio de brillo entre dos píxeles adyacentes en horizontal produce un valor distinto de cero. En particular los cambios de claro a oscuro se marcan con un valor positivo y los cambios de oscuro a claro con un valor negativo. Por otro lado, cuando dos píxeles adyacentes tienen el mismo valor la convolución espacial con h_1 en ese punto devuelve cero.

3.9.2. FILTRADO MEDIANTE EL OPERADOR DE SOBEL [9]

Una aproximación mejor al gradiente está dada por las expresiones:

$$\frac{\partial I}{\partial x} = (p_1 + 2\,p_4 + p_7) - (p_3 + 2\,p_6 + p_9)$$

$$\frac{\partial I}{\partial y} = (p_1 + 2\,p_2 + p_3) - (p_7 + 2\,p_8 + p_9)$$

Dando lugar a las matrices h_1 y h_2

$$h_1 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Figura 3.20. Filtros diseñados por Sobel.

Estas matrices se conocen como *ventanas de Sobel*, que fue quien las propuso. Mediante ellas se calcula el gradiente en las direcciones horizontal y vertical.

3.9.3. OPERADOR DE PREWITT

A diferencia de Sobel, este operador otorga menos importancia a los píxeles centrales y las ventanas de convolución son las siguientes:

$$h_1 = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 3.21. Filtros diseñados por Prewitt.

Las ventanas de convolución vienen mediante la aproximación al gradiente está dada por las expresiones:

$$\frac{\partial I}{\partial x} = (p_1 + p_4 + p_7) - (p_3 + p_6 + p_9)$$

$$\frac{\partial I}{\partial y} = (p_1 + p_2 + p_3) - (p_7 + p_8 + p_9)$$

Estas aproximaciones al llevarse a cabo de manera más rudimentaria, hace que el operador de Prewitt sea menos sensible a los bordes en altas frecuencias en comparación con el las ventanas de Sobel.

3.10. TRANSFORMACIONES MORFOLÓGICAS

Las transformaciones morfológicas se encargan de cambiar la forma y la estructura de los objetos. Estas herramientas matemáticas permiten obtener componentes que dan una idea de la forma y estructura de los objetos que forman la imagen. Además, permiten la modificación de estás formas para separar los objetos unos de otros, obtener contornos primarios, descomponer formas complejas en formas simples, obtener contornos dentro de entornos ruidosos, entre otros [6].

Generalmente esta serie de transformaciones se llevan a cabo en imágenes previamente binarizadas (en 2 tonos, blanco y negro), aunque también existen estudios sobre la aplicación de estas técnicas en imágenes con niveles de gris. Las transformaciones que se van a tratar corresponden a la morfología matemática cuyo lenguaje de expresión es la teoría de conjunto.

3.10.1. ELEMENTO ESTRUCTURANTE [6]

Antes de profundizar en las operaciones morfológicas, hay que definir al elemento estructurante. El elemento estructurante es una máscara constituida por una serie de puntos que sirven para determinar la estructura de un conjunto. Uno de esos puntos conformará el centro del elemento. Como por ejemplo, se tienen 2 elementos estructurantes uno con forma de rombo y otro con forma de circunferencia con el origen de ambos en el píxel central.

		X		
	X	X	X	
X	X	**X**	X	X
	X	X	X	
		X		

Figura 3.22. Representación de un elemento estructurante con forma de rombo de radio igual a 2 píxeles.

Fuente: "*Técnicas y algoritmos básicos de visión artificial*". Pág 52

	X	X	X	
X	X	X	X	X
X	X	**X**	X	X
X	X	X	X	X
	X	X	X	

Figura 3.23. Representación de un elemento estructurante con forma de circunferencia de radio igual a 2 píxeles.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 52

3.10.2. DILATACIÓN BINARIA

Dados dos conjuntos A y B pertenecientes a Z^2, la dilatación denotada por $A \oplus B$, se define como

$$A \oplus B = \{c \in E^N \mid c = a + b \; \forall \; a \in A \; y \; b \in B\}$$

Lo cual quiere decir, que definida una máscara B (definida por un dominio de 0 y 1), la dilatación de A por B es el conjunto de todos los desplazamientos de x tales que B y A se solapen en al menos un elemento distinto de cero [9].

Por ejemplo, una máscara B formada por la siguiente matriz

0	1	0
1	1	1
0	1	0

Figura 3.24. Representación correspondiente a una máscara de 3x3.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 54

Y una imagen formada por la matriz siguiente

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figura 3.25. Representación correspondiente a un sector de una imagen digital binaria.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 55

El proceso de dilatación $A \oplus B$ consiste, que para cada punto de A(x,y), en solapar la máscara de B en A (siendo el punto central de B el de coordenadas (x,y)) para determinar si algún uno de B coincide con algún uno de A, en caso de ser dicha aseveración verdadera, se procede a colocar el píxel de A(x,y) en uno. En resumen, la dilatación consiste en ir desplazando al elemento estructurante (B) por toda la imagen y siendo alguno de los elementos de la matriz (denominados vecinos de la máscara) coincide con un píxel del entorno, entonces el píxel donde está centrado A(x,y) se coloca en uno [6].

0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Figura 3.26. Representación correspondiente a un sector de una imagen digital dilatada.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 55

La dilatación, también llamada “crecimiento”, “llenado”, “expansión”, entre otros, como se ve en la figura siguiente, produce un efecto de engrosamiento de los bordes en lo objetos. Es un efecto muy apreciado para aumentar los contornos de los objetos, unir líneas discontinuas, etc.

La máscara o elemento estructurante, dependiendo de la forma que tome, se puede utilizar para realizar la dilatación en la dirección o direcciones de interés. Como se puede apreciar en la imagen, la dilatación tiende a unir objetos cercanos y a reducir los agujeros interiores de los objetos, dando como resultado el aumento del grosor.

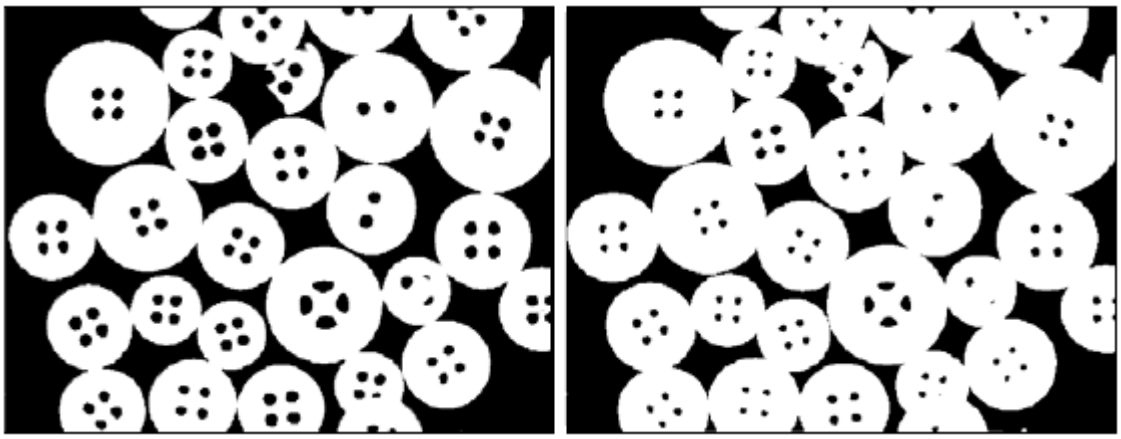


Figura 3.27. Imagen binaria (izquierda), imagen dilatada (derecha).

Fuente: *“Técnicas y algoritmos básicos de visión artificial”*. Pág 56

3.10.3. EROSIÓN BINARIA

Dados dos conjuntos A y B pertenecientes a Z^2, la erosión denotada por $A \ominus B$, se define como

$$A \ominus B = \{ x \in E^N \mid x = x + b \in A \ \forall \ b \in B \}$$

La erosión es la función dual de la dilatación (nótese que es diferente a la inversa, debido a que si se realiza una erosión y luego una dilación, la imagen no queda como la

inicial). Dada la imagen *A* que se erosiona según el elemento estructurante *B*, esto significa que cuando todos los puntos *x* tales que *B*, trasladado por *x*, está contenido en *A*. Lo que quiere decir que la erosión coloca todos los píxeles de la imagen en cero cuando no se contenga completamente al elemento estructurante en su entorno [9].

Si la dilatación expandía bordes y contorno de los objetos, la erosión los reduce. La erosión en altamente empleada para separar objetos que están unidos por una pequeña parte de sus contornos.

Por ejemplo, empleando el elemento estructurante anterior correspondiente a la Figura 24.

Y una imagen formada por la matriz siguiente

0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Figura 3.28. Representación correspondiente a un sector de una imagen digital binaria.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 58

El proceso de erosión $A \ominus B$ consiste, que para cada punto *A*(x,y), en solapar la máscara *B* en *A* (siendo el punto central de *B* el (x,y)), y ver si todos los puntos a uno de la máscara *B* coinciden con los de la imagen, colocándolos en uno. En caso contrario se colocan en cero. En resumidas cuentas, la erosión consiste en ir desplazando el elemento estructurante (*B*) por toda la imagen, y si alguno de los elementos de la matriz no coinciden

con un píxel del entorno, entonces el píxel donde está centrado *A* se pone a cero, y si todos coinciden con los de la máscara, se coloca en uno [6].

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Figura 3.29. Representación correspondiente a un sector de una imagen digital erosionada.

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 59

Igual que en la dilatación, La máscara o elemento estructurante, dependiendo de la forma que tome, se puede utilizar para realizar la operación morfológica en la dirección o direcciones de interés. En la figura anexa se puede ver un ejemplo de la erosión de una imagen donde se aprecia que los objetos se van separando uno de otros, reduciéndose su diámetro y grosor.

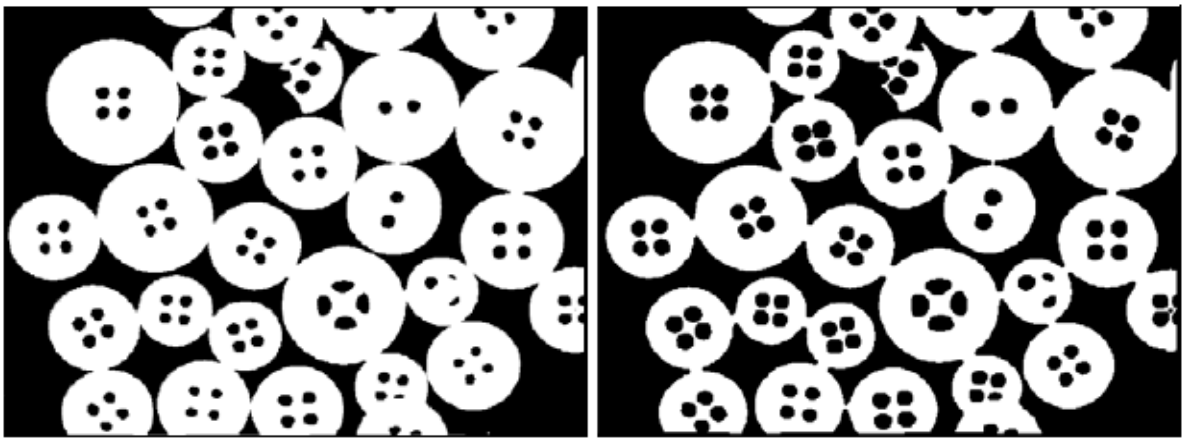


Figura 3.30. Imagen binaria (izquierda), imagen erosionada (derecha).

Fuente: *"Técnicas y algoritmos básicos de visión artificial"*. Pág 60

3.10.4. APERTURA

La apertura consiste en realizar en una imagen una erosión seguida de una dilatación. Aunque a primera vista puede parecer que la imagen va a permanecer como está, esto no ocurre debido a que la dilatación no es una operación inversa a la erosión [6].

La apertura es empleada para:

- Segmentación de objetos, separando unas formas de otras
- Descomposición de objetos en elementos más simples.
- Extracción de formas determinadas en un entorno ruidoso
- Eliminar salientes estrechos
- Separar objetos que no están demasiado juntos
- Aumentar agujeros que están dentro de los objetos

Las aperturas pueden estar formadas por un número de erosiones y dilataciones diferentes, produciendo resultados dispares [2].

3.10.5. CIERRE

El cierre hace exactamente lo contrario a la apertura, primero se realiza la dilatación y luego la erosión. La combinación de las operaciones de apertura y cierre pueden servir para múltiples operaciones, como lo son el filtrado, segmentación, detección de objetos, y otros, simplemente con manipular el tipo de elemento estructurante y el tamaño del mismo [6].

3.11. SEGMENTACIÓN

La segmentación consiste en definir regiones dependiendo de propiedades comunes existentes entre los píxeles que conforman una imagen digital.

3.11.1. BINARIZACIÓN

La binarización consiste en la transformación de una imagen constituida por varios niveles de grises a una nueva imagen con sólo dos niveles, (también conocidas como imágenes bitonales) de esta forma se segmenta la imagen en diversas regiones.

Para llevar a cabo la binarización de una imagen, primero se procede a definir un valor de umbral, (valor a partir del cual se estudia la imagen y se procede a tomar una decisión para llevar a cabo todas las transformaciones pertinentes) para esto existen diferentes algoritmos matemáticos basados en la probabilidad y la estadística. Por motivos didácticos este Trabajo Especial de Grado sólo hace mención a la existencia de los métodos, pero centrará su estudio en la comprensión de los conceptos básicos involucrados permitiendo al lector o al usuario la libre asignación de los valores de umbral.

Posterior a la selección del valor de umbral se aplican las siguientes formulas dependiendo del caso:

a) Para valores de píxeles superior del valor de umbral

$$g(x,y) = \begin{cases} 1 & si \quad T \leq f(x,y) \\ 0 & en\ cualquier\ otro\ caso \end{cases}$$

b) Para valores de píxeles superior del valor de umbral

$$g(x,y) = \begin{cases} 1 & si \quad T \geq f(x,y) \\ 0 & en\ cualquier\ otro\ caso \end{cases}$$

c) Para valores de píxeles comprendidos entre 2 valores de umbral

$$g(x,y) = \begin{cases} 1 & si \quad T_a \leq f(x,y) \leq T_b \\ 0 & en\ cualquier\ otro\ caso \end{cases}$$

Donde g(x,y) es el resultado de la binarización de la imagen, T es el valor del umbral seleccionado (para el tercer caso son los valores donde se encuentra comprendido el umbral). Y f(x,y) es el valor del pixel de la imagen a binarizar.

3.12. MICROSOFT VISUAL STUDIO 2008®

Es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows®. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET, entre otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET [16].

3.12.1. PLATAFORMA .NET

.Net es un conjunto de bibliotecas que pueden ser utilizadas por aplicaciones para optimizar el desarrollo de programas y mejorar el rendimiento. Visual .Net ofrece un entorno de ejecución de aplicaciones, compiladores, y permite el desarrollo de todo tipo de programas, aplicaciones para Internet, e incluso para dispositivos móviles [17].

3.12.2. ENTORNO DE EJECUCIÓN CLR

El Common Language Runtime (CLR) es la implementación de Microsoft de un estándar llamado Common Language Infrastructure (CLI).

El CLR/CLI define un entorno de ejecución virtual independiente en el que trabajan las aplicaciones escritas con cualquier lenguaje .Net (C#, J#, Basic). Este entorno virtual se

encarga de aspectos importantes para una aplicación como la gestión de la memoria, la vida de los objetos, la seguridad y la gestión de subprocesos y tareas [17].

Todos estos servicios unidos a su independencia respecto a arquitecturas computacionales convierten la CLR en una herramienta muy útil ya que, en teoría, cualquier aplicación escrita para funcionar según la CLI puede ejecutarse en cualquier tipo de arquitectura de hardware. Por ejemplo Microsoft dispone de implementación de .NET para Windows de 32 bits, Windows de 64 bits y hasta para Windows Mobile.

3.12.3. BIBLIOTECAS DE CLASE BASE .NET

El entorno .Net proporciona gran cantidad de funcionalidades para gestión de entrada/salida, seguridad, acceso a datos, entre otros. Esto se implementa en librerías de enlace dinámico DLL (siglas provenientes de Dynamic Link Library), al conjunto de estas librerías se le llama Base Classes Library (BCL) [17].

3.12.4. LOS ESPACIOS DE LOS NOMBRES (NAMESPACE)

Un namespace es una forma de agrupar clases, funciones, tipos de datos relacionados, y otros espacios de nombres. Se utiliza una forma jerárquica para crear un esquema que permita hacer referencia y organizar las clases de la BCL.

3.13. PROGRAMACIÓN ORIENTADA A OBJETOS

Trata de expresar una aplicación como un conjunto de objetos que interactúan entre ellos para resolver alguna tarea. Se persigue el desarrollo de de programas mediante módulos que faciliten el mantenimiento y la reutilización de código.

En Programación Orientada a Objetos (POO, también OOP en las siglas que se usan en inglés) cada objeto contiene una serie de propiedades que lo definen y lo identifican frente a otros y una serie de métodos que permiten la consulta y modificación de esas propiedades, así como la realización de otras operaciones.

3.13.1. CONCEPTOS [17]

En la programación Orientada a objetos se incorporan nuevos elementos que complementan y amplían los tradicionales fundamentos de la programación estructurada.

- **Clase**: Definición de un conjunto de propiedades y métodos que se corresponde con cada tipo de elemento que va a tener en el programa.
- **Objeto**: Instancia de una clase. Formada por un conjunto de atributos y un conjunto de métodos que permiten operar sobre ellos.
- **Método**: Algoritmo asociado a una clase que realiza alguna operación modificando alguna de las propiedades del objeto o generando un evento para otro objeto.
- **Evento**: Suceso en el sistema, producido por un usuario (una pulsación del botón del ratón, pulsar una tecla, etc.) o por otro objeto. El sistema maneja el evento proporcionando la respuesta prevista.
- **Atributo o propiedad**: Es cada una de las características asociadas a un objeto.

3.13.2. CARACTERÍSTICAS

- **Herencia:** A la hora de estructurar el programa, no tenemos clases aisladas, tenemos una jerarquía de clases que dependen unas de otras. Los objetos heredan las propiedades y los métodos de las clases a las que pertenecen. Esta propiedad nos da la posibilidad de crear objetos que extiendan o personalicen las propiedades de otros objetos existentes sin necesidad de volver a implementarlos.

- **Abstracción:** Cada objeto de la aplicación realiza una tarea y obedece a un comportamiento sin proporcionar un detalle de su funcionamiento interno. (Como están implementadas esas operaciones).
- **Encapsulamiento:** Reunimos los elementos que pertenecen a una misma entidad al mismo nivel de abstracción. Con esto se incrementa la cohesión del sistema.
- **Polimorfismo:** Dada una clase base, se declaran objetos distintos que hereden de la clase base. En estos objetos derivados permite definir comportamientos distintos asociados al mismo nombre. Al invocar al método por su nombre se utilizará el comportamiento que corresponda al objeto que estemos usando.

3.14. HERRAMIENTA DE DESARROLLO

A partir de este momento, se va a intentar dar a conocer esta herramienta de desarrollo con un enfoque práctico.

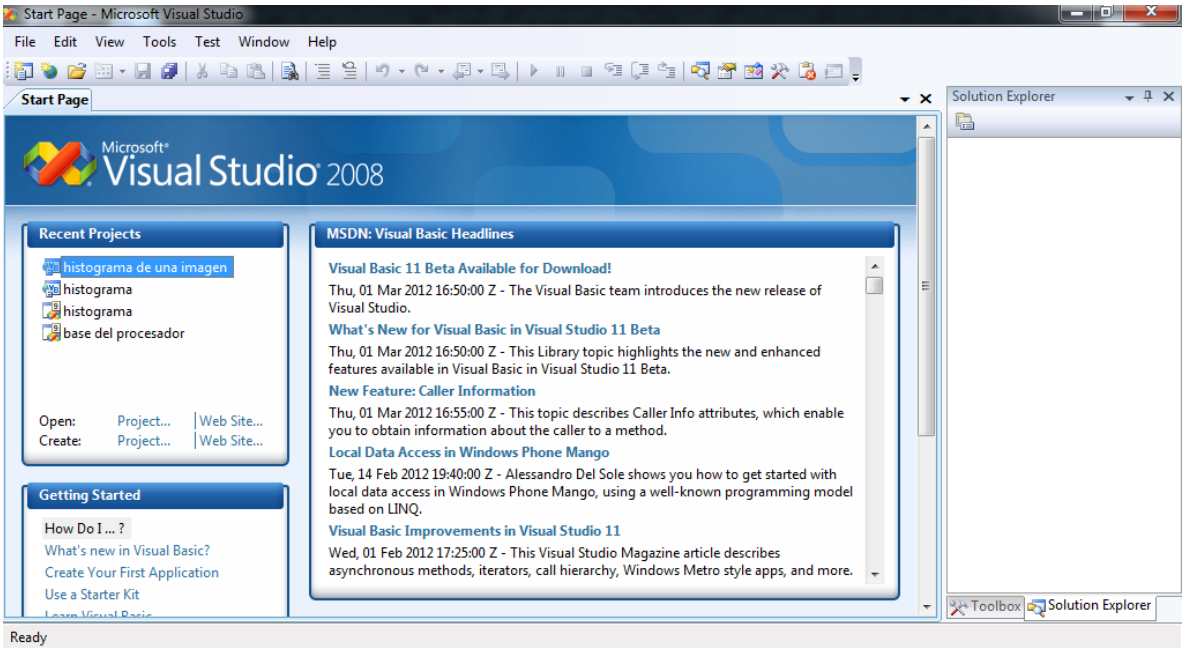


Figura 3.31. Pantalla de inicio en Visual Studio 2008.

3.14.1. CREACIÓN DE UNA APLICACIÓN

Para crear una aplicación desde Visual Studio, se puede ir a *Archivos/Nuevo Proyecto (File/New Proyect).* Luego aparecerá una pantalla donde se puede elegir el tipo de plantilla a usar (template), el tipo de proyecto (como se puede apreciar en la Figura 33) y el nombre de la aplicación.

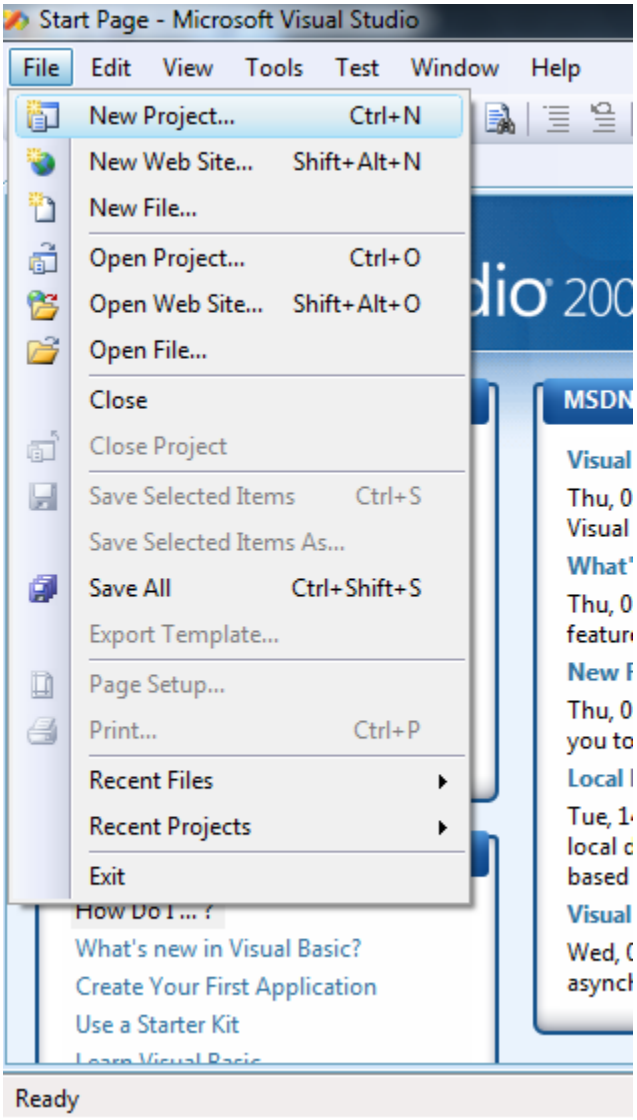


Figura 3.32. Selección de un nuevo proyecto en Visual Studio 2008.

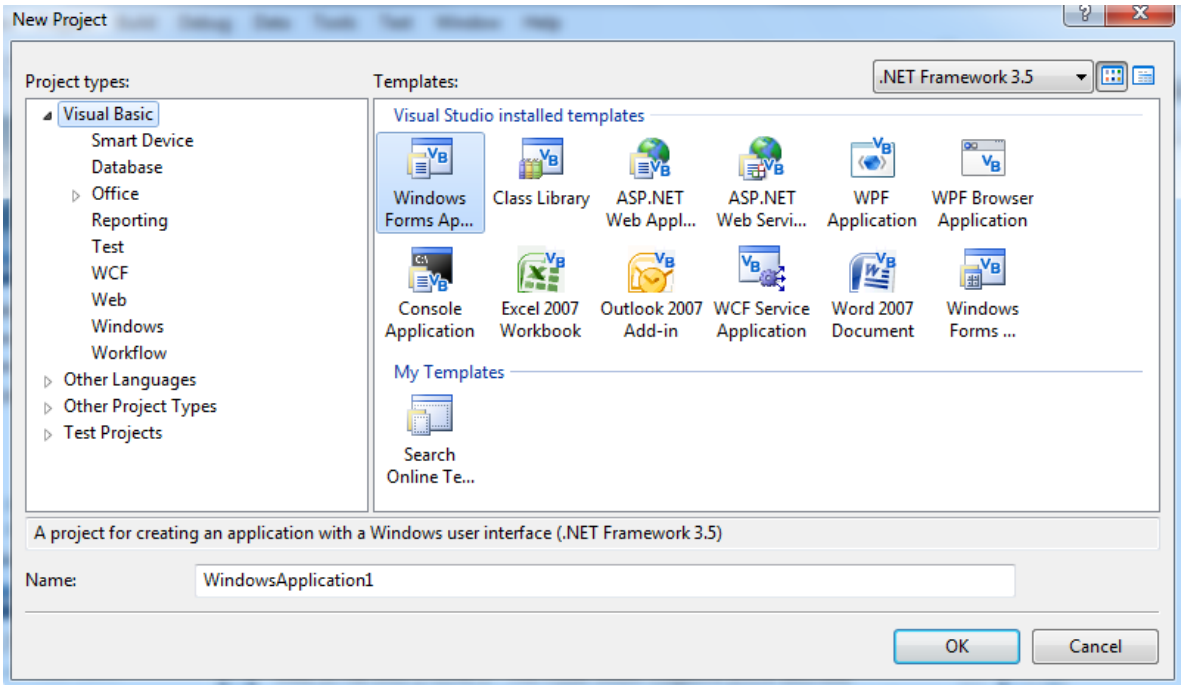


Figura 3.33. Selección de la plantilla a utilizar en Visual Studio 2008.

Una vez seleccionado este conjunto de características básicas del proyecto, se despliega una pantalla como la imagen anexa.

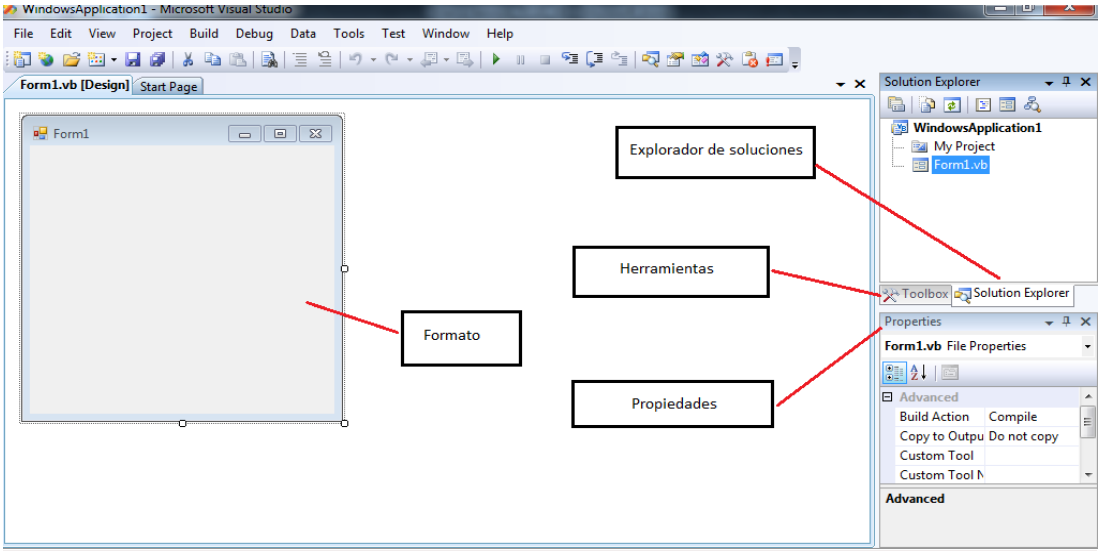


Figura 3.34. Entorno de programación Visual Studio 2008.

Donde es necesario el conocer las partes de nuestro interés, como lo son:

- **Formato**: Aspecto que tendrá el programa al ejecutarlo, es la interfaz gráfica que interactúa con el usuario que accederá a la aplicación
- **Explorador de soluciones**: Muestra las partes que conforman al proyecto. Aquí se encontrarán las páginas asociadas a código, aspecto gráficos y archivos que se deseen adjuntar para la ejecución correcta de nuestra aplicación.
- **Propiedades**: Muestra las características asociadas a cada elemento que conforma nuestra aplicación.
- **Herramientas:** Objetos que puede contener nuestra aplicación. En la Figura 35 se muestra el aspecto de algunas de las herramientas que tenemos a disposición.

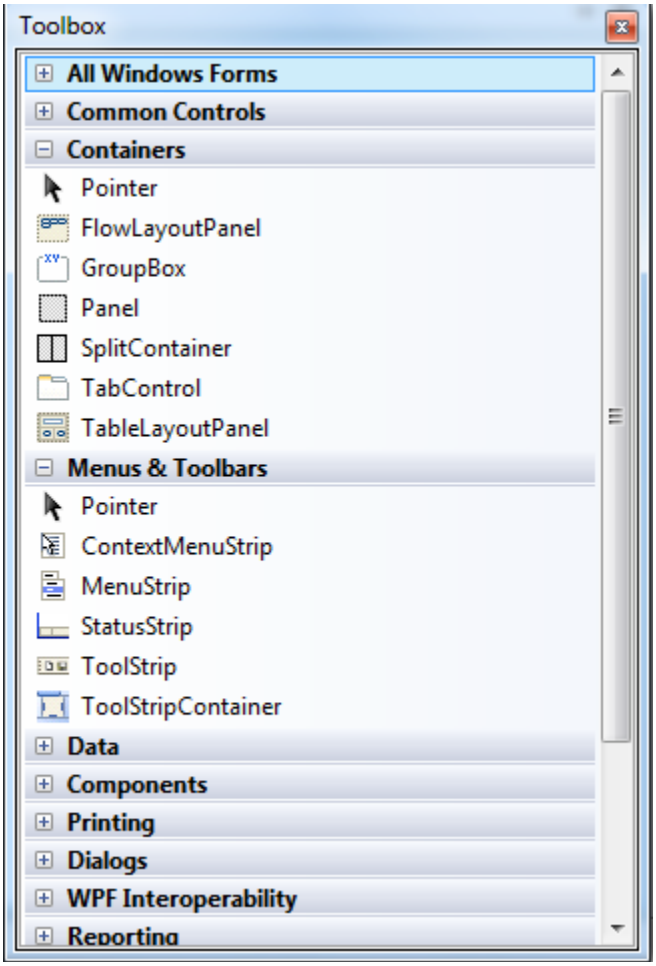


Figura 3.35. Imagen de la caja de herramientas de Visual Studio 2008.

Una vez dado a conocer el entorno, procedemos a explicar un poco sobre el lenguaje BASIC y sus sentencias más utilizadas.

3.14.2. ESTRUCTURAS CONDICIONALES

3.14.2.1. SECUENCIA CONDICIONAL IF [17]

Cuando la ejecución de un bloque de código no es lineal, sino que dependiendo de una condición se tiene que ejecutar un fragmento u otro de código, utilizamos la sentencia de control IF. Su sintaxis es la siguiente:

```
If condición Then
     Sentencias a ejecutar si la condición se cumple
Else
     Sentencias a ejecutar si la condición no se cumple
End If
```

Cuando se desean evaluar varias condiciones se puede hacer así:

```
If condición_1 Then
     Sentencias a ejecutar si la condición1 se cumple
ElseIf condición_2 Then
     Sentencias a ejecutar si la condición2 se cumple
Else
     Sentencias a ejecutar si no se cumplen
End If
```

En la condición se pueden evaluar expresiones utilizando operadores lógicos y relacionales. Para analizar si se cumplen 2 condiciones a la vez utilizamos el **AND** (Y), para ver si se cumple cualquiera de las dos el **OR** (O). Luego usamos los operadores relaciones:

MAYOR	MENOR	MAYOR O IGUAL	MENOR O IGUAL	IGUAL	DISTINTO
>	<	>=	<=	=	<>

Figura 3.36. Tabla que muestra los operadores relacionales empleados en el lenguaje Basic.

Fuente: *"Herramientas informáticas para el geoprocesado"*. Pág 19

3.14.2.2. SELECCIÓN MÚLTIPLE SELECT CASE [17]

Cuando necesitamos distinguir varias condiciones puede que sea más apropiada esta estructura que el uso de varios IFs. Luego al realizar el programa cada uno tiene que decidir qué estructura encaja mejor para resolver cada problema.

La sintaxis es la siguiente:

```
Select Case variable

Case v1
     Sentencias a realizar si la variable vale v1

Case v2 To v3
     Sentencias si la variable vale entre v2 y v3 ambos
     incluidos

Case v4, v5
     Sentencias si la variable vale entre v4 o v5

Case Is >v6
     Sentencias si la variable es mayor que v6
```

```
Case else
     Sentencias si la variable es distinta de los
     anteriores

End select
```

Se pueden utilizar varias expresiones para un mismo caso separándolas por comas. La línea siguiente sería válida.

```
Case 1 to 3, 5 to 7, 9, 15, Is >=21
```

3.14.3. ESTRUCTURAS REPETITIVAS

3.14.3.1. SENTENCIA FOR [17]

Se repiten las sentencias de dentro del bucle desde que la variable toma el valor inicial hasta que alcance el valor final con el incremento ***paso***. Si no se especifica el *paso* se interpreta que es 1. El *paso* puede ser positivo o negativo. Cuando el *paso* es positivo o cero el bucle se ejecuta si la variable es menor o igual qué fin. Cuando el *paso* es negativo el bucle se ejecuta si la variable es mayor o igual qué fin.

Y su sintaxis es la siguiente:

```
For variable=inicio To fin [Step PASO]
     Sentencias
[Exit For]
     [Sentencias]
Next
```

Con la instrucción ***Exit For*** se abandona la ejecución del bucle sin completar las iteraciones indicadas. Se utiliza para forzar la salida del bucle cuando se da una condición. Por ejemplo que ya hemos encontrado el valor que buscábamos en una lista y no necesitamos seguir comparando el resto de elementos de la misma.

3.14.4. ARRAYS [17]

3.14.4.1. ARRAYS UNIDIMENSIONALES (VECTORES)

En Visual .Net se dispone de estructuras más avanzadas para almacenar un grupo de valores. Para almacenar una serie de datos del mismo tipo se utilizan tradicionalmente los Arrays.

Y su sintaxis es la siguiente:

```
Dim Vector(Número_de_elementos) As Tipo_de_dato
```

Como por ejemplo, la línea "`Dim Vectornotas(9) As Double`" crea un vector de 10 elementos, números reales (Double). Nos está creando un vector de 10 elementos, el primero está en la posición 0, y el último en la 9

La ventaja del uso de vectores frente a las variables normales es que podemos recorrer los elementos del mismo con estructuras repetitivas para agilizar el manejo de grandes cantidades de datos.

3.14.4.2. ARRAYS MULTIDIMENSIONALES (MATRICES).

Podemos utilizar Arrays de varias dimensiones. Por ejemplo para declarar una matriz de 2 filas y 3 columnas de Double escribiríamos:

```
Dim Matriz(2,3) As Double
```

Del resto respeta las mismas consideraciones que para vectores o arrays unidimensionales.

3.14.5. IMÁGENES Y GRÁFICOS

En GDI, que es el conjunto de herramientas gráficas (Graphics Device Interface) se tiene muchas clases disponibles, y su uso y manejo de imágenes será más sencillo. El espacio de nombres con el que trabajamos será ***System.Drawing***.

En el espacio de nombre System.Drawing vamos a encontrar todas las herramientas necesarias para el diseño e implementación de este software, en los capítulos siguientes a este vamos a ver como se utiliza este espacio de nombre y sus clases para la elaboración de subrutinas que permitan la visión por computador de los robots industriales.

METODOLOGÍA DE LA INVESTIGACIÓN

Este Trabajo de Grado comprende un tipo de investigación experimental, ya que a través de dicho se trabajo se busca encontrar diferencias significativas entre el uso de variables experimentales y la muestra de control. En este trabajo podemos notar la presencia de lo antes mencionado con la definición del último objetivo específico, y para llevar a cabo la realización de este objetivo específico hay que realizar previamente los antes mencionados.

Entonces, para ejecutar la correcta realización del desarrollo de este trabajo de grado, será tarea del investigador llevar a cabo las siguientes actividades.

4.1.NOCIÓN DE LOS PUNTOS CLAVE DEL PARA ENFRENTAR EL PROBLEMA DETECTADO

Para ello se lleva a cabo una revisión bibliográfica. Todo lo referente a este tópico se encuentra contenido en el Marco conceptual (Remítase al Capítulo II).

4.1.1. DELIMITACIÓN DEL ALCANCE DE LA INVESTIGACIÓN

En esta parte de la investigación se definirán de manera clara y concisa los puntos donde se centrará el desarrollo de este Trabajo Especial de Grado, hay en este punto dos aspectos muy importantes a considerar; que son los formatos a utilizar y las arquitecturas donde se podrá ejecutar esta aplicación.

4.2.DISEÑO DEL PLAN EXPERIMENTAL

Dentro de este punto se tendrán en cuenta los siguientes aspectos:

4.2.1. DISEÑO DE LA INVESTIGACIÓN

Para este trabajo, en este punto se explica el procedimiento que se siguió para la elaboración de los diagramas de flujos y los pseudocódigos diseñados para la elaboración de la codificación de dichos diagramas. Estos procedimientos son mencionados en el Capítulo V.

4.2.2. ELABORACIÓN DE PROCEDIMIENTOS

En este punto se transformarán los pseudocódigos elaborados en el punto anterior en código máquina para su compilación y ejecución.

Es importante en este punto de la investigación aclarar que el lenguaje BASIC, ofrece una programación orienta a eventos, esto significa que las acciones a tomar o los sucesos a ocurrir vendrán en función a los eventos que ocurran en el sistema, ya sea definido por el usuario que ellos mismos provoquen [18]. Como por ejemplo: la pulsación de un botón, la carga de una imagen, el acceder a un submenú, entre otros.

La programación orientada a eventos brinda una estrategia de programación combinada entre top-down (descendente) como bottom-up (ascendente). Entiéndase que una la estrategia top-down es aquella donde el flujo del programa se basa en conocimiento y planificación del mismo, y por ende sigue un orden secuencial, mientras que la bottom-up centra su programación en el estudio de una inicialización para poder ejecutar un módulo de la programación [19].

Teniendo en cuenta lo antes dicho, y sabiendo que esta investigación busca el desarrollo de una aplicación, en el Apéndice B se encontrará cada una de las líneas de códigos junto al evento que desencadena esa sección de código.

4.3.PRUEBA DE CONFIABILIDAD DE DATOS

Esta es la parte de la investigación donde se llevaron a cabo las ejecuciones de los algoritmos desarrollados para ver los resultados obtenidos. Esta sección contiene ejemplos de los procesamientos realizados por el software elaborado, además se incluye el manual de instalación e inicio rápido del programa, donde se indican todo los requisitos de hardware necesarios y las actividades que se pueden ejecutar con el software.

4.4.TRATAMIENTO DE DATOS

Para finalizar esta investigación, se instalará esta aplicación en diversos sistemas operativos, con diversas arquitecturas para así corroborar su funcionalidad en diversas plataformas y se prueban cada uno de los diferentes tipos de imágenes a procesar en las plataformas de prueba.

Además se evalúa el software diseñado bajo los estándares establecidos por la ISO para demostrar que si satisface los criterios de usabilidad y precisión, con la ejecución de un test a la población a la cual va dirigida este programa, la cual es los estudiantes de procesamiento digital de imágenes, para esto se selecciona una muestra de 10 estudiantes. Para ver de manera más detallada los resultados del tratamiento de los datos recurra al Capítulo VII donde encuentra toda la información pertinente.

CUADRO TÁCTICO

En este capítulo se desglosa de manera organizada cada una de las partes correspondientes al desarrollo de la aplicación y se fijan los criterios para el diseño.

Iniciando con la secuencia básica que se rige en la aplicación para llevar a cabo un procesamiento exitoso de la imagen de estudio.

Figura 5.1. Diagrama de bloques de las actividades a ejecutar para procesar una imagen.

5.1.ABRIR IMAGEN

Este es el primer paso de nuestro procesamiento, y consta de los códigos que hacen el proceso de adquisición de la imagen para nuestro software.

Seguidamente, se lleva a cabo de la lectura de la cada uno de los píxeles correspondientes a los espacios de color que conforman la imagen y se almacena en una variable que denominaremos *Matriz*.

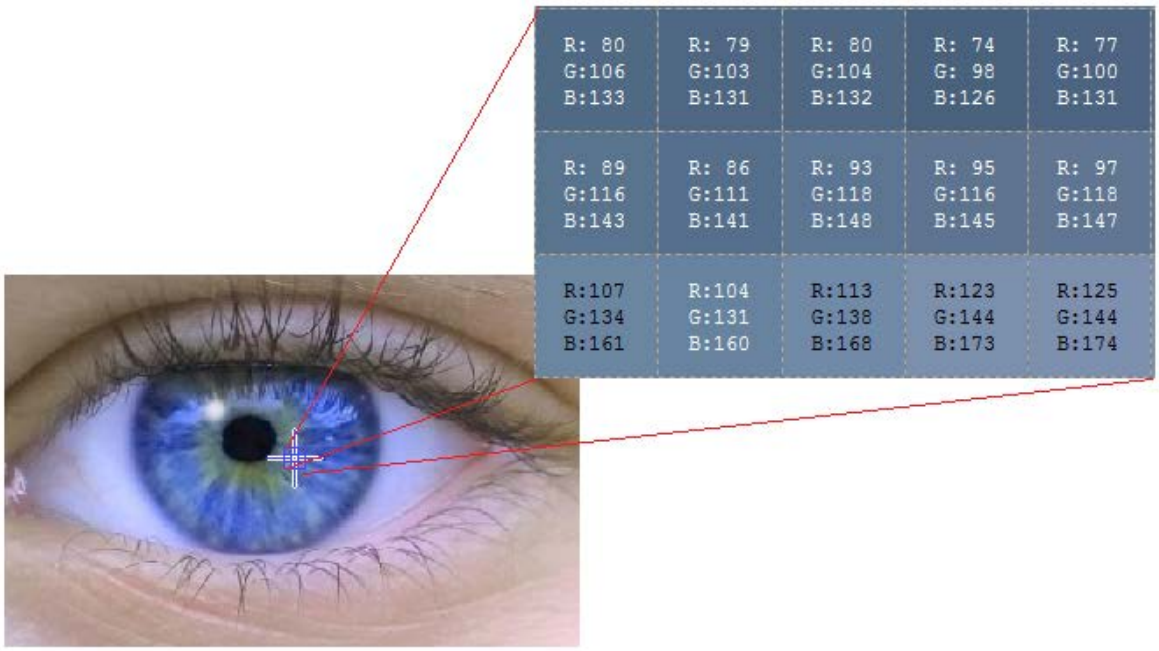


Figura 5.2. Imagen representativa de la transformación de un conjunto de píxeles de color a un grupo de magnitudes vectoriales que representan mencionado color.

5.2.PINTAR IMAGEN

Este bloque se encarga de presentar al usuario la imagen la imagen procesada, o lo que es lo equivalente a realizar una representación visual de la variable *Matriz*.

5.3.GUARDAR IMAGEN

Este es el bloque que claudica nuestro procesamiento ya que indica que se está satisfecho y que se desea almacenar el resultado obtenido. Ya sea para ejecutar otros tratamientos a esta nueva imagen o simplemente como resultado del procesamiento.

Para los bloques correspondientes a Abrir imagen, Pintar imagen y Guardar no se profundizará mucho debido a que no requieren mayor estudio que los códigos comprendidos en el Apéndice de este Trabajo especial de Grado, en las secciones B.2, B.3 y B.4 del Apéndice B se encuentran los códigos mencionados.

Ahora, el punto de interés en este momento se centra en las diversas actividades que se pueden realizar en el bloque de Procesar imagen, para comprender mejor todas las actividades que se pueden llevar a cabo en esta etapa del procesamiento, se remite al estudio de la Figura 5.3 donde se muestran todas las actividades relacionadas a este bloque.

A partir de este momento se comienza a diseñar y definir cada una de las estructuras correspondientes a estos bloques, que constituyen al bloque Procesar imagen.

Tenga presente que los datos que se obtienen en este momento son los resultantes del bloques previo (Abrir Imagen hace el proceso de adquisición y almacena los valores de los perfiles de color de la imagen a estudiar).

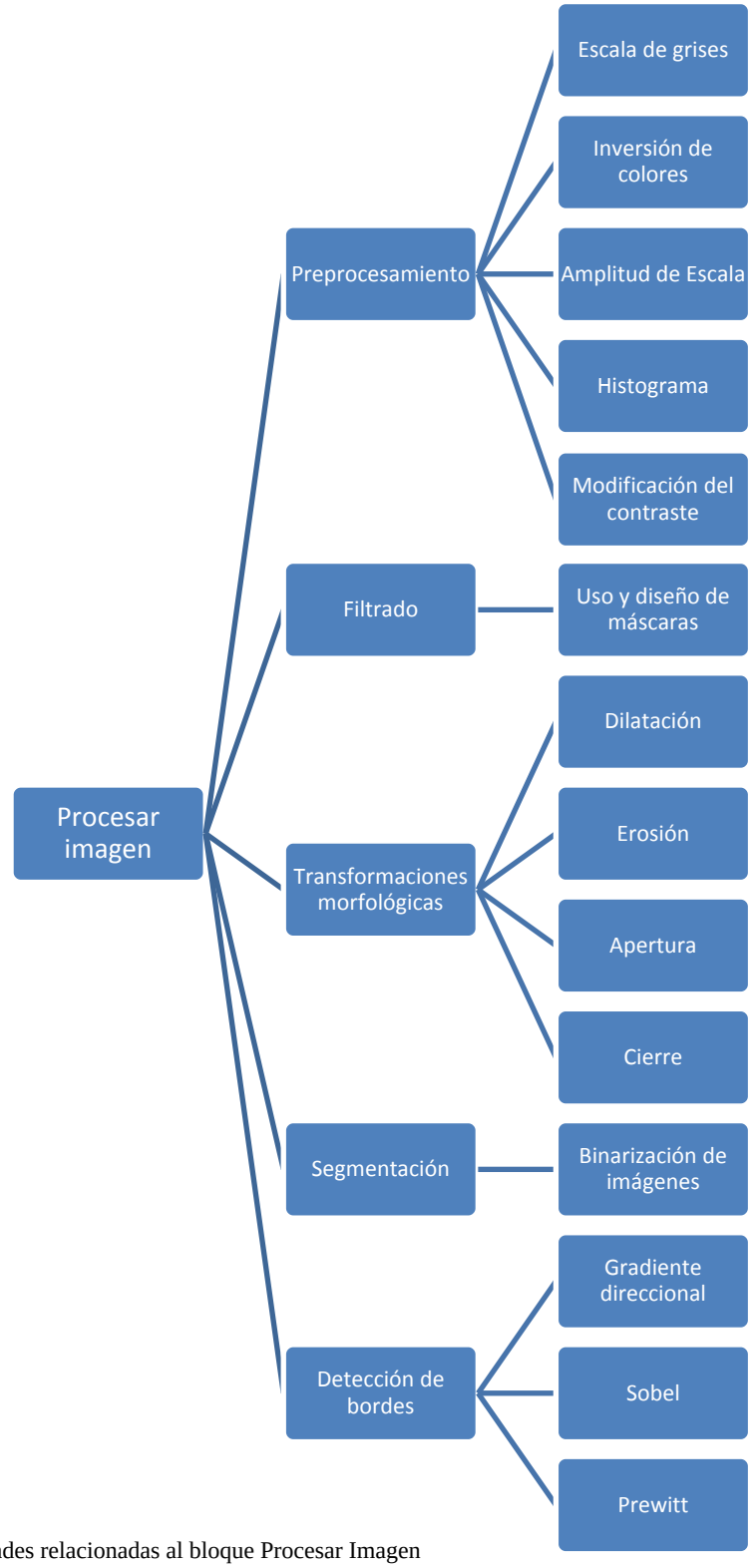


Figura 5.3. Actividades relacionadas al bloque Procesar Imagen

5.4.ESCALA DE GRISES

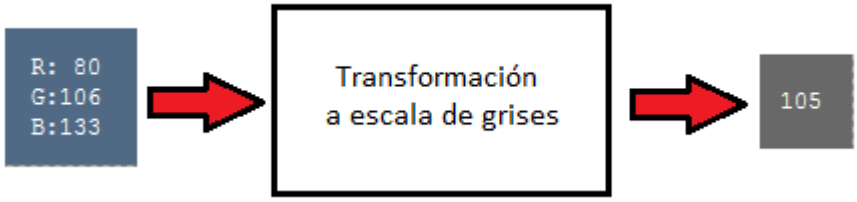


Figura 5.4. Diagrama de bloques donde se indican los parámetros de entradas al bloque *Escala de Grises* y los parámetros de salida.

La transformación de las componentes de color a la escala de grises puede considerarse una transformación directa ya que consiste en el promedio de las componentes de color.

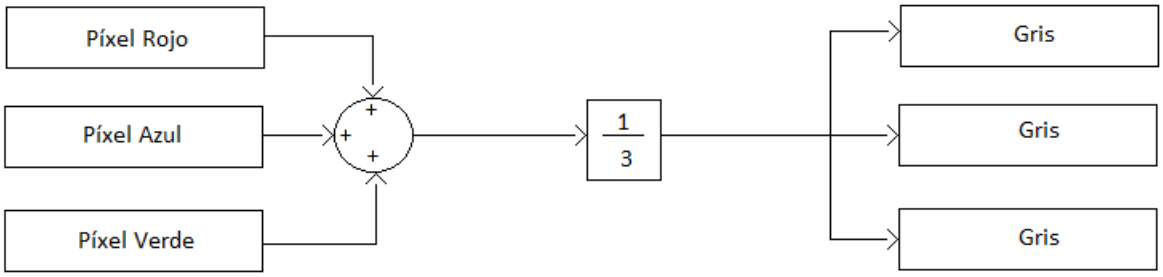


Figura 5.5. Diagrama de bloques de las actividades a realizar para llevar de color a escala de grises.

5.5.INVERSIÓN DE COLORES

Figura 5.6. Diagrama de bloques donde se indican los parámetros de entradas al bloque *Inversión de Colores* y los parámetros de salida.

Otra de las transformaciones directas que vamos a abordar en este apartado, consiste en la transformación de los espacios de color a su inverso. Entiéndase de blanco al negro, de azul a amarillo, de verde a magenta y de rojo a cyan. Lo cual se traduce en una transformación del espacio de color RGB al CYMK.

Para llevar a cabo la transformación sólo hay que tomar la profundidad de color y sustraerle el valor de cada píxel para cada perfil de color.

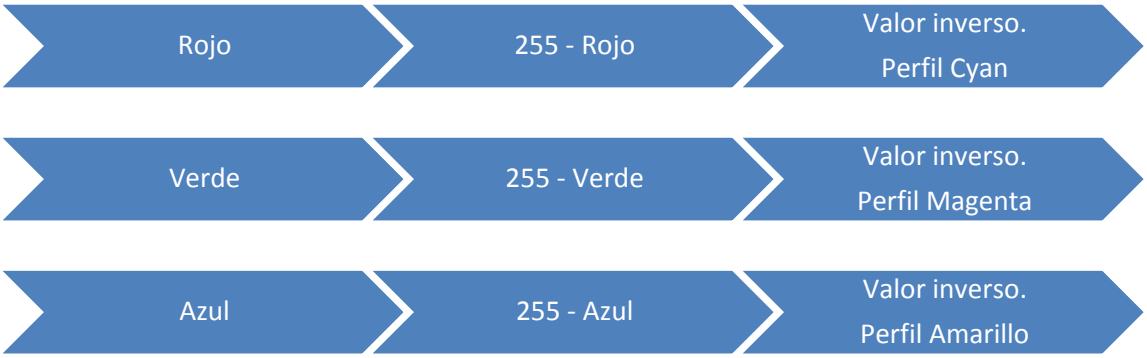


Figura 5.7. Diagrama de las actividades a ejecutar para invertir los colores de una imagen.

5.6.AMPLITUD DE LA ESCALA O NORMALIZACIÓN

Esto se hace con la finalidad de aumentar o reducir los rangos que comprenden la profundad de color de la imagen digital. El procedimiento para esta actividad es el siguiente:

1. Determinar los valores máximos y mínimos que constituyen a la imagen a normalizar y calcular el alcance de esta variación. (Kmáx y Kmín de la Ecuación 5)

2. Delimitar un nuevo valor máximo para la nueva profundidad de color. (K de la Ecuación 5)
3. Aplicar la Ecuación 5 correspondiente a la normalización de la imagen.

5.7.HISTOGRAMA

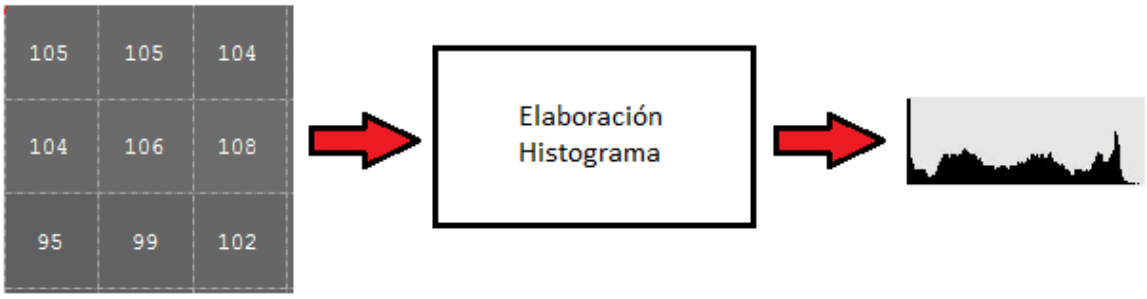


Figura 5.8. Diagrama de bloques donde se indican los parámetros de entradas al bloque *Elaboración Histograma* y los parámetros de salida.

El histograma de una imagen es la representación grafica del número de pixeles asociados a un valor de profundidad de color de la imagen estudiada, como se establece en las bases conceptuales (Capítulo III, página 23).

El uso de arreglos matriciales simplifica enormemente la tarea de contabilizar los valores de los píxeles que conforman la imagen. Para la elaboración de gráfica en el Visual Studio® se emplea un complemento adicional llamado *MS Chart* disponible en el sitio oficial de la Microsoft Visual Studio® (Para la versión empleada en este trabajo se obtuvo el complemento de la siguiente dirección: http://archive.msdn.microsoft.com/mschart. Fecha de extracción: Abril, 2012)

5.8.MODIFICACIÓN DEL CONTRASTE

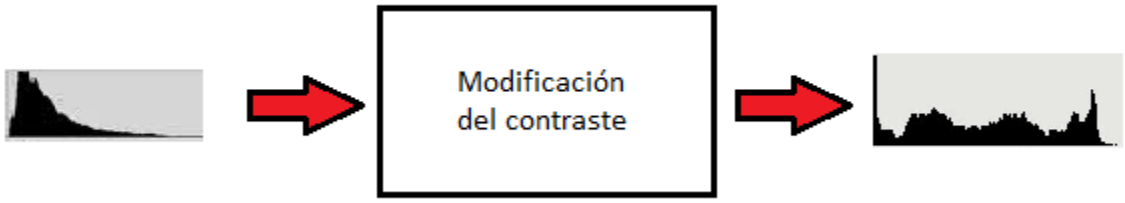


Figura 5.9. Diagrama de bloques donde se indican los parámetros de entradas al bloque *Modificación del contraste* y los parámetros de salida.

Teniendo en cuenta lo establecido en el Capítulo III de este Trabajo Especial de Grado, para la modificación del contraste previamente se debe calcular el histograma de la imagen. En la figura siguiente se muestran los pasos a seguir.

Figura 5.10. Diagrama de las actividades a ejecutar para modificar el contraste de una imagen.

Para realizar la modificación del contraste, se debe conocer los valores de los píxeles que conforman la imagen, luego según la definición de la función que rige la modificación se ejecuta sobre el valor del píxel para conocer su nuevo valor y este proceso se puede apreciar en la figura anexa, esto afecta al histograma original y por ello se procede a calcular el nuevo histograma correspondiente a la nueva imagen.

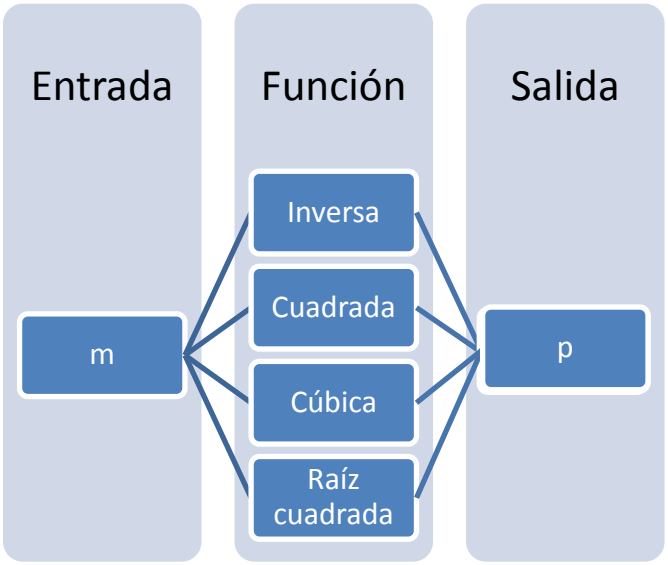


Figura 5.11. Diagrama de bloques de las funciones que modifican el contraste de una imagen, indicando los parámetros de entrada y salida.

5.9.FILTRADO

5.9.1. USO Y DISEÑO DE MÁSCARAS

El procesamiento con máscaras incluye trabajar con un entorno de un punto, representado por una matriz generalmente cuadrada y con número impar de píxeles (3x3, 5x5, 7x7, entre otros). Es decir, en lugar de considerar únicamente los niveles digitales de cada píxel, se consideran además los niveles digitales de los píxeles adyacentes.

Esto implica que el algoritmo con que se esté trabajando no puede empezar en la primera fila (ni en la primera columna) de la imagen, ni puede terminar en la última fila (última fila) de la imagen. Para comprender mejor este concepto se remite al Apéndice B.7, con la ayuda de los comentarios se asimila mejor lo expresado en este párrafo.

Almacenar el resultado de la aplicación de la máscara, se crea una variable que almacene el producto del valor del píxel de la imagen con el de la máscara. Denominada

por *SumaRojo*. Tenga presente que el procesamiento de las máscaras en este trabajo está diseñado para imágenes en escala de grises, de esta manera es indiferente que perfil de color se estudie, ya que previamente se observó que todas las coordenadas tienen el mismo valor.

Luego de la obtención de la variable *SumaRojo* se procede, si el caso lo requiere a tener en cuenta los factores de amplitud y el de desviación para definir el valor del píxel que corresponde a la imagen procesada y se almacena en la variable que se ha definido como *Rojo.*

Al final de la obtención del la variable *Rojo* se procede a evaluar su valor, si este excede el valor del la profundidad de color (255) se cambia a el máximo permitido, caso contrario permanece en su valor.

Todo este proceso se repite para cada uno de los pixeles que conforman a la imagen de estudio.

5.10. MORFOLOGÍA

5.10.1. DILATACIÓN BINARIA

Con base a lo establecido en el Capítulo III, la dilatación binaria consiste en ir desplazando el elemento estructurante por toda de la imagen y si en alguno de los elementos de la matriz coincide con un píxel de entorno, entonces el pixel donde está centrada la matriz se coloca en uno.

El procedimiento para esto es el siguiente:

1) Hacer coincidir el área del elemento estructurante con una porción de la imagen original.

Para esto; con la ayuda de 2 ciclos *For* se recorre la matriz correspondiente a la imagen como al elemento estructurante seleccionado. La selección de los parámetros de inicio y final del ciclo *For* viene dado por 2 valores, uno positivo y otro negativo. La razón de estos valores se puede ver en la imagen siguiente. Donde se selecciona como centro el propio centro del elemento estructurante, y no se recorre de extremo a extremo como se recorrería tradicionalmente. Todo esto tiene la finalidad de llevar a cabo la dilatación y evitar el diseño de algoritmos para localizar el píxel de la imagen resultante posterior a la dilatación.

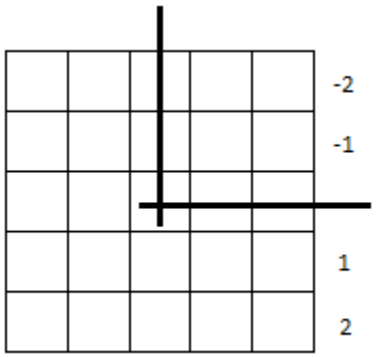


Figura 5.12. Estudio de un elemento estructurante de 5x5 y justificación de la selección de los valores limitantes del ciclo *For* que se emplean para recorrer la imagen y el elemento estructurante.

2) Al recorrer la imagen calcular el valor de los productos existentes por la superposición del elemento estructurante con la porción de la matriz y sumar cada producto obtenido, este valor debe ser almanceado en una variable que se justificará su uso en el siguiente paso.

3) Una vez determinada la sumatoria de los productos (variable encontrada en el paso anterior) sólo queda evaluar la condición si dicha sumaria es diferente de 0. Debido a que si alguno de los puntos coincidió con el elemento estructurante, hizo que la sumatoria difiera de cero, lo cual hace que se satisfaga la condición de la dilatación binaria, caso contrario la sumatoria quedaría en cero. Con las previsiones tomadas

en el paso 1; ahora sólo hay que colocar el valor correspondiente al pixel central y proceder a desplazar el elemento estructurante a la siguiente posición.

5.10.2. EROSIÓN BINARIA

Nuevamente, haciendo referencia en lo definido en las Bases Conceptuales (Capítulo III), la erosión binaria consiste en ir desplazando el elemento estructurante por toda de la imagen y si en alguno de los elementos de la matriz no coincide con un píxel de entono, el píxel donde está centrada la imagen se coloca en cero, en caso de coincidir todos se coloca en uno.

El procedimiento para esto es el siguiente:

1) Hacer coincidir el área del elemento estructurante con una porción de la imagen original.

Para esto con la ayuda de 2 ciclos *For* se recorre la matriz correspondiente a la imagen como al elemento estructurante seleccionado. La justificación de la selección de los límites del ciclo *For* es análoga a la mencionada para la dilatación binaria.

2) Antes de llevar a cabo alguna operación, con la ayuda de los ciclos *For*, se procede a contabilizar el número de unos que conforman al elemento estrunturante. Este valor será almacenado en una variable auxiliar.

3) Recorrer la imagen y calcular el valor de los productos existentes por la superposición del elemento estructural con la porción de la matriz y sumar cada producto obtenido, este valor debe ser almanceado en una segunda variable que se justificará su uso en el siguiente paso.

4) Una vez determinada la sumatoria de los productos (Variable encontrada en el paso anterior) sólo queda evaluar la condición de:

   a. La variable que contiene la sumatoria de los productos es igual al número de unos que conforman al elemento estructurante: En este caso y según la definición, el píxel resultante de la erosión se coloca en 1.
   b. En cualquier otro caso, el píxel resultante de la erosión debe ser colocado en 0.

5.10.3. APERTURA Y CIERRE

Ya previamente establecido la dilatación y la erosión binaria, simplemente nos queda recordar que la apertura es una erosión seguida de una dilatación. Y que un cierre es una dilatación seguida de una erosión.

La forma de proceder entonces será la siguiente:

1. Definir el elemento estructurante aplicado en ambas acciones.
2. Invocar las funciones de dilatación y erosión en el orden adecuado según sea el caso.
3. Mostrar la imagen resultante.

5.11. SEGMENTACIÓN

5.11.1. BINARIZACIÓN

Figura 5.13. Diagrama de bloques donde se indican los parámetros de entradas al bloque *Binarización* y los parámetros de salida.

Según lo establecido en el Marco teórico, hay definir uno o dos valores de umbrales para poder llevar a cabo la binarización de la imagen.

Para poder representar entonces las funciones que implican binarizar la imagen, se establece un estudio de sentencias condicionales con el empleo de la sentencia *If*. Para poder entender un poco mejor esto se recomienda remitirse al Apéndice B.26, B.27 y B.28

5.12. DETECCIÓN DE BORDES

El detectar bordes es una función basada en el mismo principio usar máscaras de convolución, la única diferencia es que los filtros están previamente definidos y dependiendo al filtro de uso empleado recibe un nombre (Sobel, Prewitt y gradiente direccional). El uso de máscaras de convolución está definido en *Uso y diseño de máscaras* correspondiente a este capítulo, y las deducciones de los filtros para la detección de bordes se encuentran definidas en el Capítulo III de este Trabajo Especial de Grado.

ANÁLISIS DE RESULTADOS

Para concluir con este Trabajo Especial de Grado, se procede a evaluar la aplicación desarrollada mediante los criterios de precisión y usabilidad.

6.1.DEFINICIONES DE USABILIDAD [18]

La Organización Internacional para la Estandarización (ISO) ofrece 2 definiciones:

"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso"

ISO/IEC 9241:

"Usabilidad es la eficacia, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico"

Ya teniendo en cuenta las definiciones de usabilidad, se procede a llevar a cabo su estudio mediante la selección de un grupo de usuarios y solicitarle que lleven a cabo las tareas para las cuales fue el diseñado el software. [19].

Establecido entonces el plan de trabajo, se definen las medidas que se serán estudiadas mediante la realización del estudio de usabilidad.

6.2.MÉTRICAS DE USABILIDAD

Dentro del extenso mundo de variables mensurables para una aplicación que ejecute interacciones hombre-máquina, se recomienda estudiar las mencionadas a continuación:

Exactitud: Número de errores cometidos por los sujetos de prueba y si estos fueron recuperables o no al usar los datos o procedimientos adecuados. [19]

Precisión: Concordancia entre sí del conjunto de medidas realizadas en igualdad de condiciones. [20]

Recuerdo: Qué tanto recuerda el usuario después de un periodo sin usar la aplicación. [19]

Facilidad de Aprendizaje: facilidad con la que nuevos usuarios desarrollan una interacción efectiva con el sistema o producto. Está relacionada con la predicibilidad, sintetización, familiaridad, la generalización de los conocimientos previos y la consistencia. [19]

Facilidad de Uso: facilidad con la que el usuario hace uso de la herramienta, con menos pasos o más naturales a su formación específica. Tiene que ver con la eficacia y eficiencia de la herramienta. [19]

6.3.PRUEBA DE USABILIDAD

Definido todos los criterios de estudio, se procede a seleccionar una muestra de 10 estudiantes de la Facultad de Ingeniería de la Universidad de Carabobo. Núcleo: Valencia. Estado Carabobo. Venezuela. Para aplicarles la prueba de usabilidad, que consiste en la ejecución de un número de tareas y durante su ejecución, el sujeto de estudio debe ir completando un formulario impreso donde se le indica las tareas a realizar.

Mencionado formulario se encuentra en el Apéndice C.

6.4.RESULTADOS DE LA PRUEBA DE USABILIDAD Y PRECISIÓN

A continuación se presentan las respuestas obtenidas a la prueba se usabilidad aplicada.

1) Ingrese al programa haciendo doble click en el ícono de Imagen Studio.
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

   Resultado: Actividad realizada satisfactoriamente.

2) Ingrese al menú Archivo e intente abrir la imagen de su elección.
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

   Resultado: Actividad realizada satisfactoriamente.

3) Ingrese al menú de operaciones y pinte la imagen que abrió en asignación anterior, para esto use el submenú de pintar matriz.
   a. ¿Pudo usted llevar a cabo la tarea pintar matriz? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

   Resultado: Actividad realizada satisfactoriamente.

4) Intente detectar los bordes de la imagen aplicando el algoritmo de Sobel en la dirección horizontal. Para esto recurra al menú de Filtrado.
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 30% NO: 70%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 70% NO: 30%
   c. ¿Presentó algún mensaje de error la ejecución de esta tarea SI: 70% NO: 30% NA: 0%
   d. ¿Sintió alguna incomodad por el contenido del mensaje de error? SI: 0% NO: 70% NA: 30%
   e. ¿Cree Usted estar capacitado para la solución de este problema? SI: 70% NO: 0% NA: 30%

Resultado: Actividad realizada satisfactoriamente. 70% porcentaje de personas que no cumplieron con la tarea asignada. La presencia del error se justifica ya que los usuarios que notaron la presencia del mensaje error fue por falta de bases teóricas, debido a que este algoritmo está diseñado para imágenes en escala de grises, en caso de haber seleccionado una imagen a color, el mensaje de error indicaría que el usuario debe cambiar la imagen por una a escala de grises.

5) Nuevamente intente detectar los bordes aplicando el algoritmo de Sobel en la dirección horizontal. Para esto recurra al menú de Filtrado.
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

Resultado: Actividad realizada satisfactoriamente. Al usuario poder ejecutar la actividad nuevamente y apreciar un resultado acertado, esto se traduce en un alto índice de facilidad de recordar cómo se ejecuta una actividad y que las interacciones hombre-máquina pueden ser llevadas a cabo de manera correcta.

6) Ingrese al menú de Segmentación y binarice la imagen para valores de umbral superiores a 80.
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

Resultado: Actividad realizada satisfactoriamente.

7) Ingrese al menú Archivo y guarde su resultado en la memoria del computador
   a. ¿Pudo usted llevar a cabo esta tarea? SI: 100% NO: 0%
   b. ¿Presentó algún error la ejecución de esta tarea? SI: 0% NO: 100%

Resultado: Actividad realizada satisfactoriamente.

Ha finalizado con la ejecución de las tareas, ahora proceda a calificar los siguientes enunciados:

c. Presentación:

Excelente 70% Buena 30% Regular 0% Mala 0% Pésima 0%

Figura 6.1. Gráfico de torta indicativo de la aceptación de la presentación del software.

d. Disposición de los menú:

Excelente 70% Buena20% Regular 10% Mala 0% Pésima 0%

Figura 6.2. Gráfico de torta indicativo de la disposición del menú del software.

e. Presentación de imágenes:

Excelente 80% Buena 20% Regular 0% Mala 0% Pésima 0%

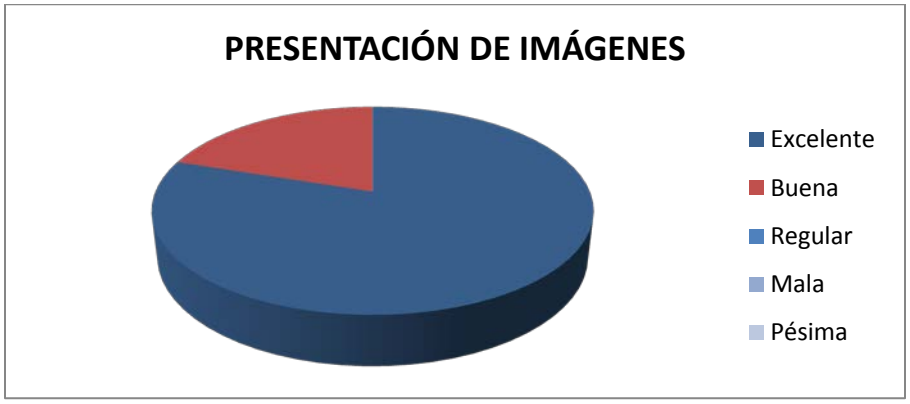


Figura 6.3. Gráfico de torta indicativo de la aceptación de la presentación de las imágenes del software.

f. Disponibilidad y distribución del material de ayuda:

Excelente 90% Buena 10% Regular 0% Mala 0% Pésima 0%

Figura 6.4. Gráfico de torta indicativo de la aceptación de la disponibilidad y distribución del material de ayuda en el software.

g. Presencia de Glosario:

Excelente 90% Buena 10% Regular 0% Mala 0% Pésima 0%

Figura 6.5. Gráfico de torta indicativo de la aceptación de la presencia del glosario en el software.

6.5.CONCLUSIONES DE LA PRUEBA DE USABILIDAD

Con base a los resultados obtenidos, se puede concluir que el software diseñado cumple de manera precisa los objetivos impuestos. Permite una correcta interacción entre el hombre y la máquina para llevar a cabo los objetivos requeridos.

En la gráfica siguiente se puede apreciar de manera global los resultados de las actividades pedidas en función al número de personas que la ejecutó, según la prueba de usabilidad presentada.

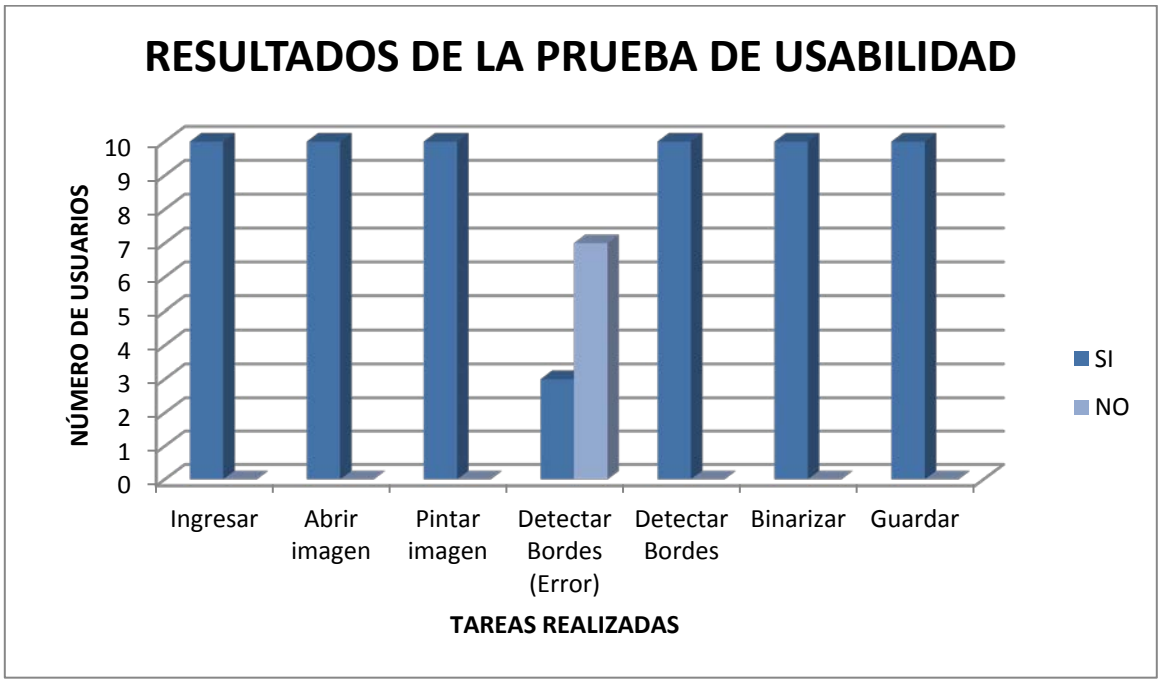


Figura 6.6. Gráfico de barras que contiene el resultado general de la aplicación de la prueba de usabilidad.

De esta manera se concluye que este software cumple a cabalidad los estándares de usabilidad impuestos en el estudio realizado.

RESULTADOS DEL PROCESAMIENTO CON EL SOFTWARE DISEÑADO

VII

A continuación una muestra de las imágenes procesadas en Imagen Studio y los resultados obtenidos:

(a)

(b)

Figura 7.1. Imágenes estudiadas. (a) Imagen de Lenna usada para probar la aplicación diseñada. Dimensiones: 313x251. Formato: PNG. (b) Imagen resultante de la aplicación de la transformación a escala de grises. Dimensiones: 313x 251. Formato: JPEG

(c)

CARACTERÍSTICAS

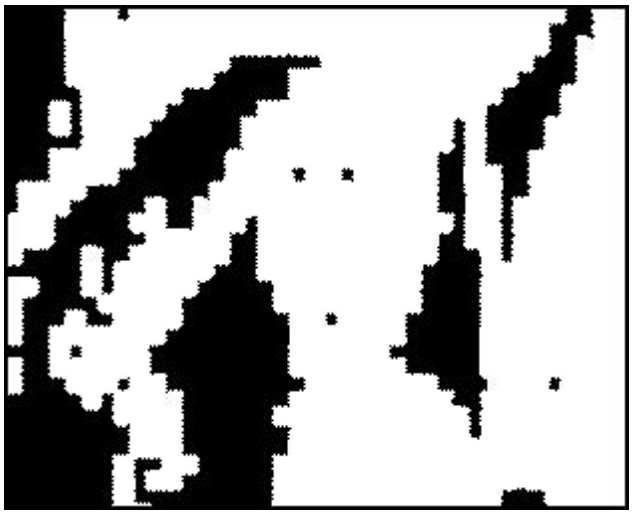
- Las partes blancas indican los bordes detectados.
- Acentuados los bordes en la dirección horizonal.
- La imagen original es la (b) Figura 1. (Escala de Grises)
- Las partes blancas indican valores alto al valor del gradiente

(d)

CARACTEÍSTICAS

- Imagen Bitonal (Blanco y negro).
- Los puntos blancos son los valores superiores e iguales a un valor de umbral de 80, siendo la imagen original la (b) de la Figura 1.
- Binarizar es una técnica para segmentar y dividir en regiones.

Figura 7.2. Imágenes procesadas. (c) Imagen resultante de la detección de bordes mediante ventanas de Sobel horizontales aplicadas a la imagen (b) Figura 1. Dimensiones: 313x 251. Formato: JPEG. (d) Imagen resultante de la umbralización para valores superiores a 80 a la imagen (b). Dimensiones: 313x 251. Formato: JPEG



(e)

CARACTERÍSTICAS

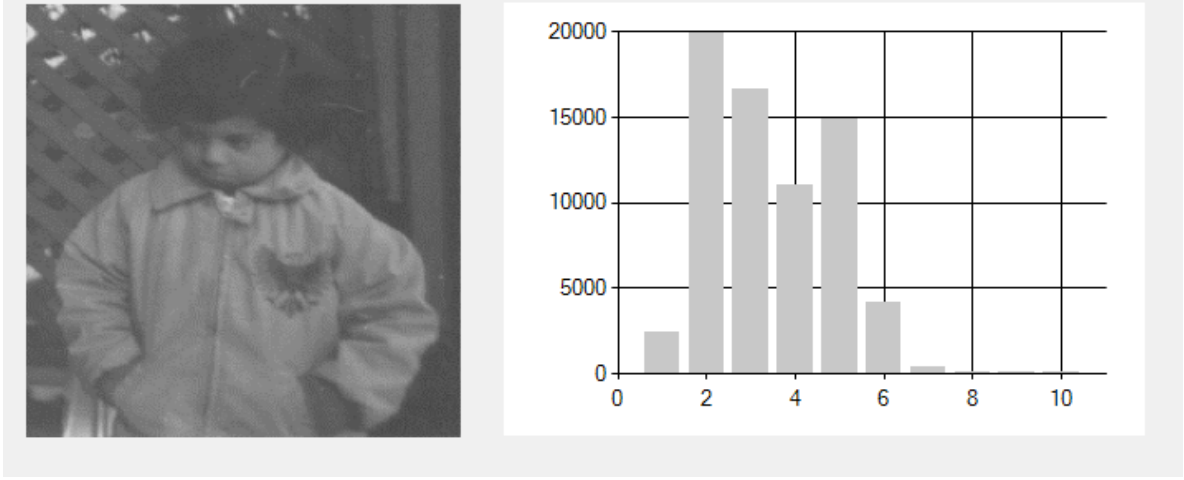
- Imagen bitonal.
- Resultado de la dilatación con elemento estructural en forma de rombo de radio igual a 2 píxeles.
- Engrosamiento de los bordes.
- Disminución de las partes negras

(f)

CARACTERÍSTICAS

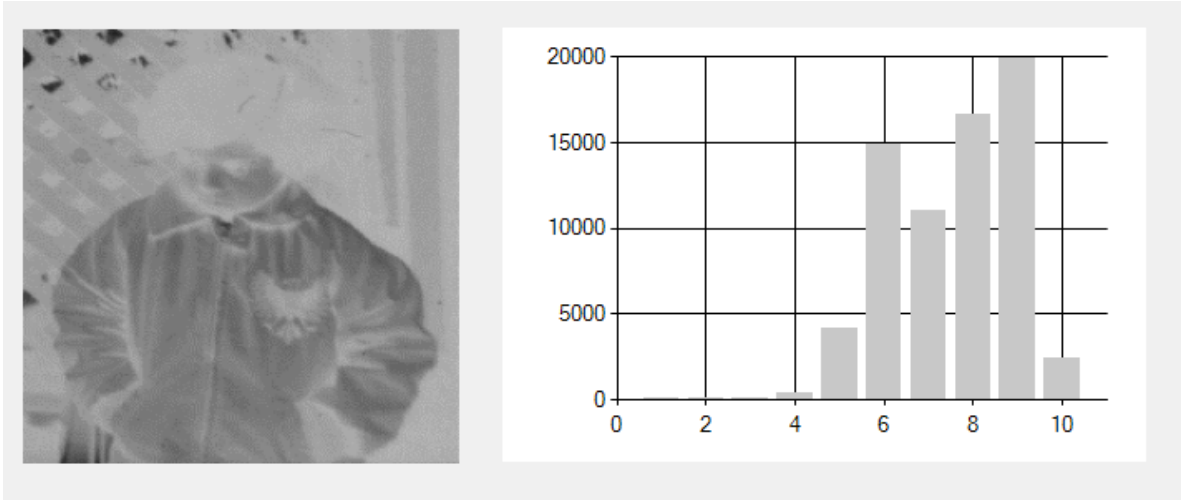
- Imagen bitonal.
- Reducción de los bordes.
- Aumento de las zonas negras.
- Separado de sectores blancos.

Figura 7.3. Imágenes procesadas con operaciones morfológicas con un elemento estructural en forma de rombo y de radio igual a 2 pixeles. (e) Imagen resultante de la dilatación aplicada a la imagen (d). Dimensiones: 313x 251. Formato: JPEG. (d) Imagen resultante de la erosión aplicada a la imagen (d). Dimensiones: 313x 251. Formato: JPEG



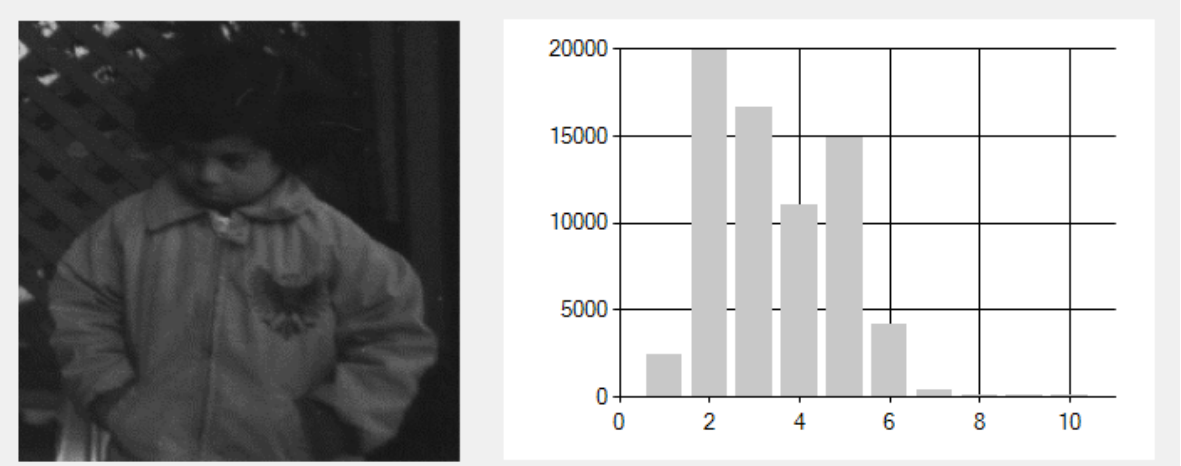
(a) (b)

Figura 7.4. Imagen para la obtención del histograma. (a) Imagen de prueba. Dimensiones: 240x 291. Formato: TIFF (b). Histograma de la imagen.



(a) (b)

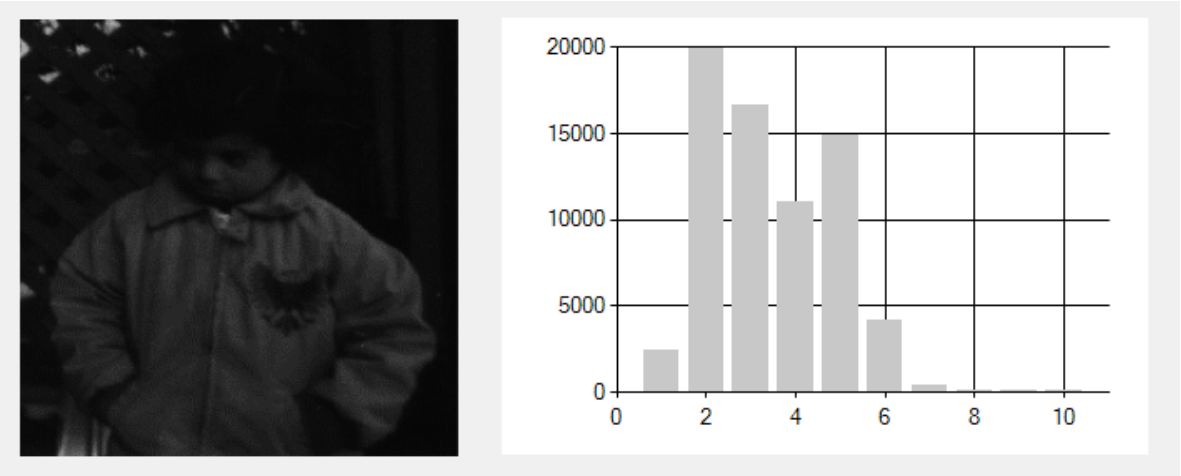
Figura 7.5. Modificación del contraste mediante la función inversa. (a) Imagen resultante. Dimensiones: 240x 291. (b). Histograma de la imagen asociado a la modificación del contraste.



(a) (b)

Figura 7.6. Modificación del contraste mediante la función cuadrática. (a) Imagen resultante. Dimensiones: 240x 291. (b). Histograma de la imagen asociado a la modificación del contraste.

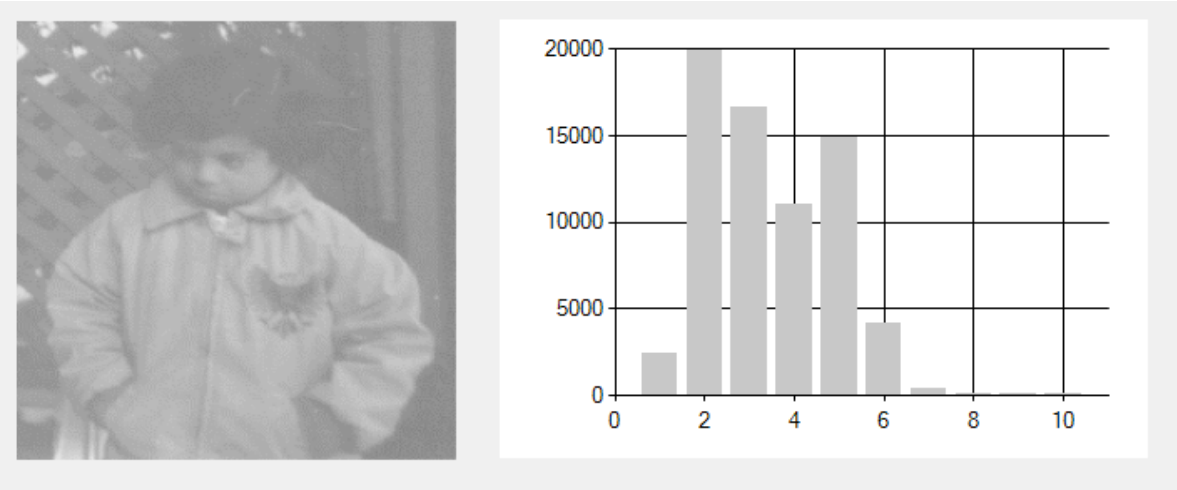
CARACTERÍSTICAS: Oscurecimiento general de la Figura 6 imagen (a) pero mejora los niveles de gris en los puntos oscuros. Véase la parte del águila en la camisa del niño de la imagen.



(a) (b)

Figura 7.7. Modificación del contraste mediante la función cúbica. (a) Imagen resultante. Dimensiones: 240x 291. (b). Histograma de la imagen asociado a la modificación del contraste.

CARACTERÍSTICAS: Resultados más acentuados que con la función cuadrática.



(a) (b)

Figura 7.8. Modificación del contraste mediante la función raíz cuadrada. (a) Imagen resultante. Dimensiones: 240x 291. (b). Histograma de la imagen asociado a la modificación del contraste.

CARACTERÍSTICAS: Resultados contrarios a los obtenidos con función cuadrática, mejoría en los contrastes de los puntos claros de la imagen original.

(a) (b)

Figura 7.9. Uso de filtros de promedios para el suavizado de imágenes. (a) Imagen original. Dimensiones: 256x 256. Formato: TIFF (b). Imagen suavizada. Dimensiones: 256x 256. Formato: TIFF

CARACTERÍSTICAS: Difuminación general de la imagen, pérdida de detalles en los objetos que se encuentran en el fondo de la imagen, presencia del marco negro de la imagen por la aplicación de la máscara de convolución.

Figura 7.10. Diagrama de bloques con resultados de la aplicación del cambio de color a escala de grises.

Figura 7.11. Diagrama de bloques con resultados de la aplicación de ventanas de Sobel Horizontal a la imagen en escala de grises

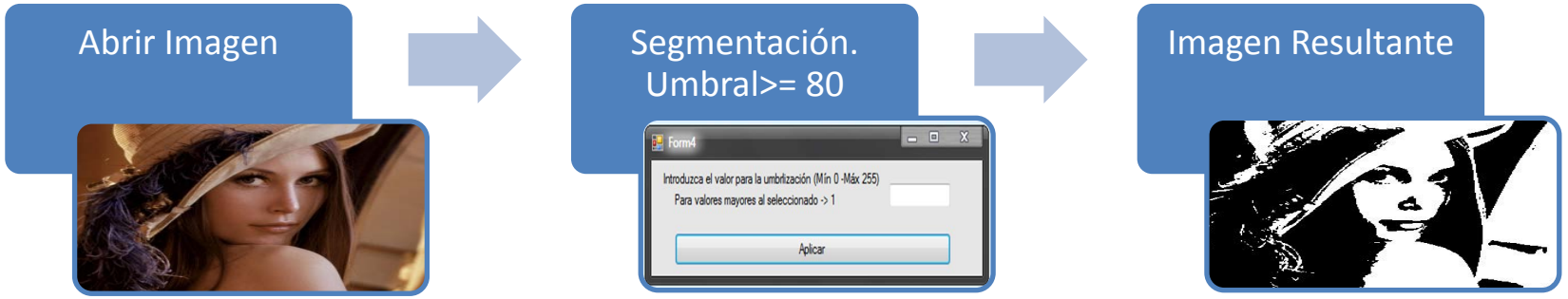


Figura 7.12. Diagrama de bloques con resultados de la aplicación de Binarización con un valor de umbral superior a 80

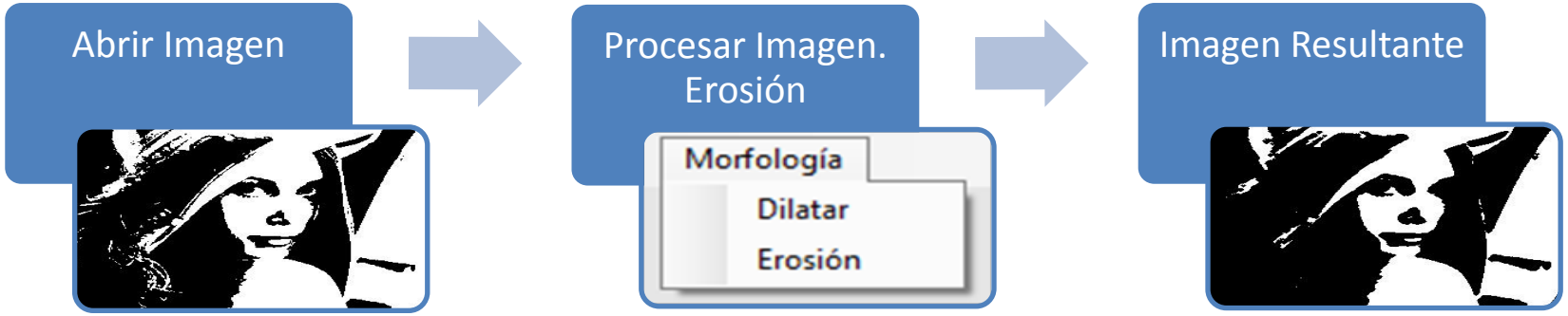


Figura 7.13. Diagrama de bloques con resultados de la aplicación de la erosión de la imagen binaria.

ÍNDICE DE FIGURAS

CAPÍTULO III. BASES CONCEPTUALES

CAPÍTULO V. CUADRO TÁCTICO

CAPÍTULO VII. ANÁLISIS DE RESULTADOS

CAPÍTULO VII. RESULTADOS DEL PROCESAMIENTO CON EL SOFTWARE DISEÑADO

APÉNDICE A: DISEÑO DE LA APLICACIÓN.

ÍNDICE GENERAL

7.3. CONCLUSIONES Y RECOMENDACIONES

En este punto se plantean las conclusiones generales y recomendaciones de este Trabajo Especial de Grado, dando inicio con las conclusiones:

- El software diseñado satisface los parámetros establecidos en los criterios de usabilidad definidos por ISO.
- El procesamiento digital de imágenes para los Robots Industriales centra su estudio en el filtrado espacial de las imágenes.
- El software diseñado requiere de poco espacio para su instalación en comparación con las capacidades de almacenamiento que se encuentran actualmente a disposición de los usuarios.
- El software está verificado para los siguientes sistemas operativos: Windows XP®, Windows Vista® y Windows 7®.
- A pesar de haber desarrollado muchos de los algoritmos para imágenes en escala de grises, estás técnicas son fácilmente aplicables a las imágenes de color con la simple aplicación de la técnica a cada componente que conforma ese espacio de color.

Para las recomendaciones:

- Empleo del software para fines didácticos en cátedra de Visión Artificial.
- Actualización del software a futuro debido al gran avance de las computadoras con el transcurso del tiempo.

REFERENCIAS BIBLIOGRÁFICAS

[1]. JUAREGUI, Juan. "**Introducción a la fotogametría"**. http://webdelprofesor.ula.ve/ingenieria/iluis [Online]. Fecha de extracción: Enero 2012.

[2]. DOMINGO, Mery. **"Visión Artificial"**. Departamento de Ingeniería Informática. Universidad de Santiago de Chile. Santiago de Chile. 09 de enero del 2002.

[3]. Librerías gráficas y multimedia C/C++. http://www.zator.com/libreriasG.htm. [Online]. Fecha de extracción: Enero 2012.

[4]. Codejock Software Solutions. http://www.codejock.com/corporate/whatsnew.asp#12092011. [Online]. Fecha de extracción: Enero 2012.

[5]. VTK. Visualization ToolKit. http://www.vtk.org/. [Online]. Fecha de extracción: Enero 2012.

[6]. Grupo de investigación EDMANS. **"Técnicas y algoritmos básicos de visión artificial"** Universidad de la Rioja. Servicio de publicaciones. España 2006

[7]. Newtone Technologies. Categoría: Colorimetría. http://www.newtone.fr/es/curso_colorimetria_codificacion_color.php. [Online]. Fecha de extracción: Noviembre 2011.

[8]. DE LA ESCALERA, Arturo. **"Visión por computador. Fundamentos y Métodos"**. Prentice Hall.

[9**].** VÉLEZ, José. MORENO, Ana. SÁNCHEZ, Ángel. SÁNCHEZ-MARÍN, José. **"Visión por computador".** 2° Edición. http://www.terra.es/personal/jfvelez/libro2/libro.html .

[Online]. Fecha de extracción: Febrero 2012

[10]. Aulapc. Categoría: Imagen Digital. www.aulapc.es/dibujo_imagen_color.html.

Fecha de extracción: Enero 2012.

[11]. Microsoft 2011 ©. Categoría: Modelos de color. http://msdn.microsoft.com/es-es/library/dd129721.aspx. [Online]. Fecha de extracción: Noviembre 2011.

[12]. GRIMALDO, José. Tratamiento digital de imágenes. http://www.grimaldos.es/cursos/imgdig/color.html

[13]. CUEVAS J, Erik V; ZALDIVAR N, Daniel. **"Visión por computador utilizando Matlab y el toolbox de procesamiento digital de imágenes".** 2 Edición. 2007

[14]. Herramientas informáticas para el Geoprocesado. Geotecnologías Cartográficas en Ingeniería y Arquitectura. Tema 2: Tratamiento Radiométrico. Curso 2009-2010. http://212.128.130.23/eduCommons/ensenanzas-tecnicas/herramientas-informaticas-para-el-geoprocesado/contenidos/Materiales/Tema2_Imagenes.pdf [Online]. Fecha de extracción: Enero 2012.

[15]. PINILLA ,C ; ALCALÁ , A y ARIZA, J. **"Filtrado de imágenes en el dominio de la frecuencia".** http://www.aet.org.es/?q=revista8-5 [Online]. Fecha de extracción: Enero 2012.

[16]. Sitio Oficial Microsoft Visual Studio. http://www.microsoft.com/visualstudio/es-es. [Online]. Fecha de extracción: Enero 2012.

[17]. Herramientas informáticas para el Geoprocesado. Geotecnologías Cartográficas en Ingeniería y Arquitectura. Tema 1: Entorno de Desarrollo .Net. Curso 2009-2010. http://212.128.130.23/eduCommons/ensenanzas-tecnicas/herramientas-informaticas-para-el-geoprocesado/contenidos/Materiales/Tema1_Introduccion.pdf [Online]. Fecha de extracción: Enero 2012.

[18]. Microsoft 2012 ©. Categoría: Inicio de Visual Basic.

http://msdn.microsoft.com/es-es/vbasic/ms789107.aspx. [Online]. Fecha de extracción: Julio 2012.

[20].JIMÉNEZ Eréndira, ORANTES, Sandra. "**Metodologías híbridas para desarrollo de software: una opción factible para México".** http://www.revista.unam.mx/vol.13/num1/art16/art16.pdf. [Online]. Fecha de extracción: Julio 2012.

[21]. Mi Tecnológico. Categoría: Normas ISO/ IEC 9126.

http://www.mitecnologico.com/Main/LaNormaIsoIec9126. [Online]. Fecha de extracción: Junio 2012.

[22]. CLAROS, Iván. **"Lineamientos de diseños bajo entornos usables"**.

http://www.intergraphicdesigns.com/tools/test-usabilidad-web/. [Online]. Fecha de extracción: Junio 2012.

[23].AVELLÁN, A; SZARVAS, J. "**Técnicas para el estudio de datos experimentales**". Universidad de Carabobo. Facultad de Ingeniería. 1992.

[24]. HURTADO & RIVAS. **"Visión Artificial"**.

http://www.roboticayautomatizacion.es/productos.asp. [Online]. Fecha de extracción: Noviembre 2012.

[25]. INFAIMON VISÃO ARTIFICIAL. **"Inspección de galletas con Visión Artificial"**.

http://infaimon.com/2012/04/inspeccion-de-galletas-con-vision-artificial/[Online]. Fecha de extracción: Noviembre 2012.

[26]. INFAIMON VISÃO ARTIFICIAL. **“Inspección de galletas con Visión Artificial”**.

http://infaimon.com/2012/04/inspeccion-de-galletas-con-vision-artificial/[Online]. Fecha de extracción: Noviembre 2012.

[27]. 123RF. **“Botellas de agua vacías en una línea de producción en la fábrica”**.

http://es.123rf.com/photo_4270904_botellas-de-agua-vacias-en-una-linea-de-produccion-en-la-fabrica.html[Online]. Fecha de extracción: Noviembre 2012.