Capítulo 4

Diseño del software para pruebas de algoritmos propuestos

4.1 Software de prueba

En este capítulo se muestran los procedimientos más importantes del software que se escribió con la finalidad de probar los algoritmos propuestos en esta tesis, para el mejoramiento del desempeño de un robot móvil explorador que utiliza el método de acción y percepción directamente acoplado introducido por Braitenberg en [25].

Los programas de pruebas se escribieron en lenguaje Visual Basic 6 , en ambiente Windows, a continuación se muestra un diagrama de bloques, diagrama de flujo y el pseudocódigo de las funciones más importantes del sistema de exploración del robot.

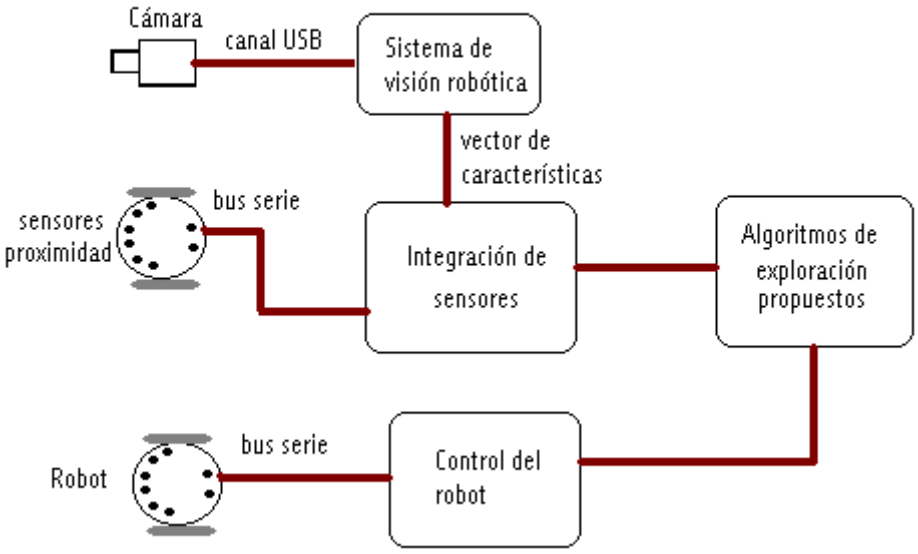

Figura 4.1 Sistema de exploración propuesto

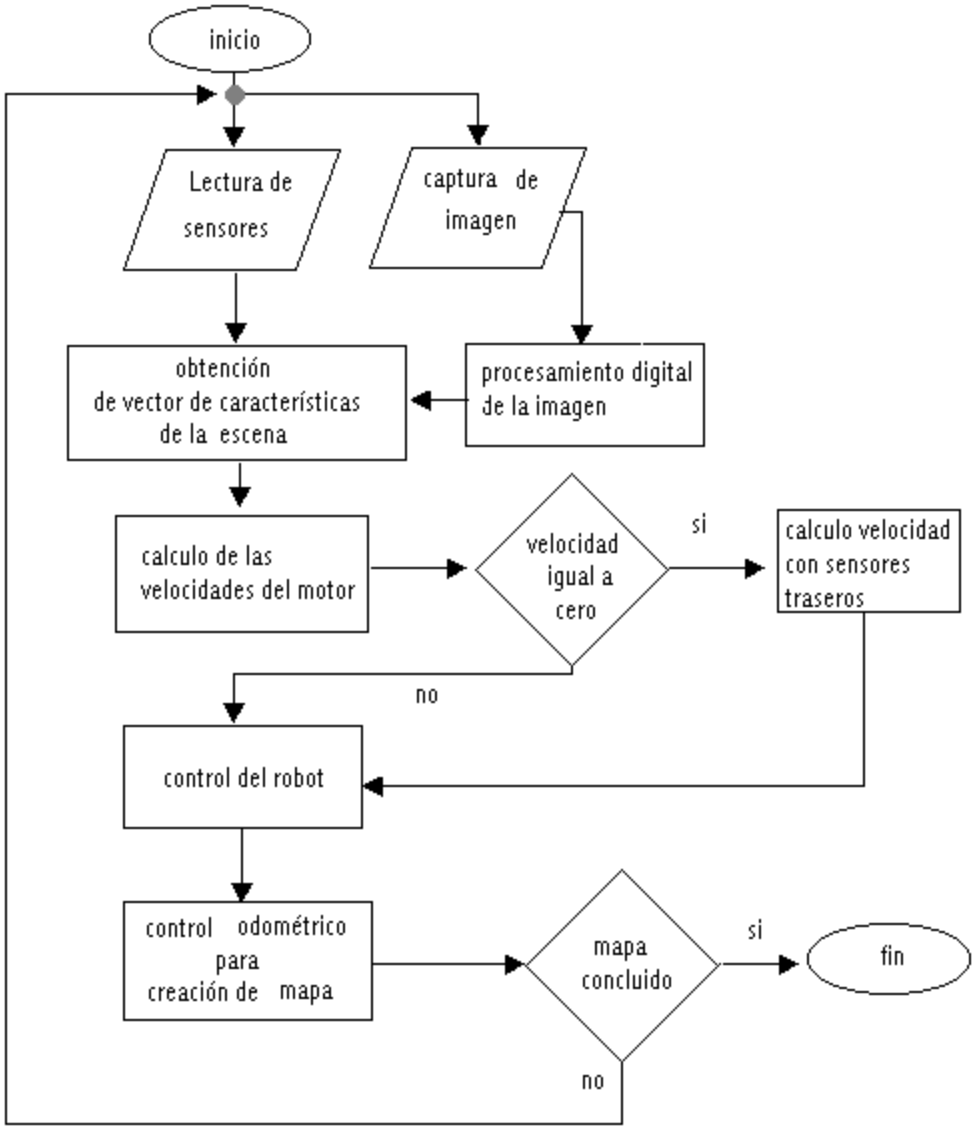

Figura 4.2 Diagrama de flujo del sistema de navegación

4.1.1 Algoritmo de Braitenberg con comportamiento explorador

sesibilidad_frente = -17

sesibilidad_45 = -11

sesibilidad_90 = -7

divi = 1000

```
velocidad_crucero = 4

val_inter(3) = sensores(3) / divi * sesibilidad_frente
val_inter(4) = sensores(4) / divi * sesibilidad_frente
val_inter(2) = sensores(2) / divi * sesibilidad_45
val_inter(5) = sensores(5) / divi * sesibilidad_45
val_inter(1) = sensores(1) / divi * sesibilidad_90
val_inter(6) = sensores(6) / divi * sesibilidad_90

motor_derecho = Round(inic + val_inter(3) + val_inter(2) + val_inter(1))
motor_izquierdo = Round(inic + val_inter(4) + val_inter(5) + val_inter(6))

call arranca
```

4.1.2 Algoritmo de Braitenberg con comportamiento explorador propuesto

```
sesibilidad_frente = -17
sesibilidad_45 = -11
sesibilidad_90 = -7
divi = 1000
velocidad_crucero = 4

val_inter(3) = sensores(3) / divi * sesibilidad_frente
val_inter(4) = sensores(4) / divi * sesibilidad_frente
val_inter(2) = sensores(2) / divi * sesibilidad_45
val_inter(5) = sensores(5) / divi * sesibilidad_45
val_inter(1) = sensores(1) / divi * sesibilidad_90
val_inter(6) = sensores(6) / divi * sesibilidad_90

motor_derecho = Round(inic + val_inter(3) + val_inter(2) + val_inter(1))
```

```
        motor_izquierdo = Round(inic + val_inter(4) + val_inter(5) + val_inter(6))

// Robot  no puede continuar su exploración, sensores nivelados
        If (motor_derecho = 0 And motor_izquierdo = 0) Or (motor_derecho = -1 And
        motor_izquierdo = 0) Or (motor_derecho = 0 And motor_izquierdo = -1)

                Call braitenberg_inverso

        Else
                 Call arranca
        End If


// braitenberg_inverso

        sesibilidad_frente = -17
        divi = 1000
        inic = 4
        cont =0

        val_inter(1) = sensores(7) / divi * sesibilidad_frente * 0.5
        val_inter(2) = sensores(8) / divi * sesibilidad_frente

        motor_izquierdo = (-1) * Round(inic + val_inter(1))
        motor_derecho = (-1) * Round(inic + val_inter(2))
        cont = cont + 1

        // para switchar entre braitenberg normal  e inverso

        If cont > 10 And sensores(1) = 0 And sensores(2) = 0 And sensores(3) = 0 And
sensores(4) = 0 And sensores(5) = 0 And sensores(6) = 0 Then
```

```
// giro aleatorio
        Randomize
         varia = Int((18 - (-18) + 1) * Rnd - 18)
         motor_derecho = varia
         motor_izquierdo = varia * (-1)

       end if

       Call arranca
```

4.1.3 Algoritmo de Braitenberg con comportamiento explorador con sistema de visión y control odométrico

```
       sesibilidad_front = -17
       sesibilidad_45 = -11
       sesibilidad_90 = -7
       divi = 1000
       inic = 6

       val_inter(3) = sensores(3) / divi * sesibilidad_frente
       val_inter(4) = sensores(4) / divi * sesibilidad_frente
       val_inter(2) = sensores(2) / divi * sesibilidad_45
       val_inter(5) = sensores(5) / divi * sesibilidad_45
       val_inter(1) = sensores(1) / divi * sesibilidad_90
       val_inter(6) = sensores(6) / divi * sesibilidad_90

       motor_derecho = Round(inic + val_inter(3) + val_inter(2) + val_inter(1))
       motor_izquierdo = Round(inic + val_inter(4) + val_inter(5) + val_inter(6))
```

```
// Sistema de visión
//obizq y obder son los vectores de características de la escena
motor_derecho = Round((motor_derecho / 2) + (obizq / -35))
motor_izquierdo = Round((motor_izquierdo / 2) + (obder / -35))
Call arranca
Exit Sub

// si no existen obstáculos
If (motor_derecho = motor_izquierdo) And motor_derecho > 0 Then

distancia = 5
// dibuja la línea en el mapa
Call FD
// avanza el robot
Call avanza

End If

If (motor_derecho = motor_izquierdo) And motor_derecho <= 0 Then
distancia = 5
Call BK
distancia = distancia * (-1)
Call avanza
End If

// giro  a la izquierda
If motor_derecho > motor_izquierdo Then
grados = 5
principal.Text21.Text = grados
Call LF
Call gira
```

```
 End If

// giro a la derecha
If motor_derecho < motor_izquierdo Then
grados = (-1) * 5

Call RT
Call gira
End If
```

4.1.4 Algoritmos de visión robótica propuestos

```
// Aplicación de filtros para  segmentar imágenes

For ren = 1 To 120
  For col = 1 To 160
          ((B-G)/4)+127+(exp(0.0189*R))
         ((R-G)/4)+127-(exp(0.0189*B))
          au2 = Round((imagen_R(ren, col) - imagen_G(ren, col)) / 4)
          au2 = au2 + 125
          au3 = (imagen_B(ren, col) * 0.0189)
          au1 = Exp(au3)
         imagen_i2(ren, col) = Abs(Round(au2 - au1))
          Next col
Next ren
```

```
// reducción  de escala de la imagen y umbralización
          ren1 = 1
          col1 = 1
          For ren = 1 To 115 Step 4
           col1 = 1
            For col = 1 To 155 Step 4
             aux = 0

            For i = ren To ren + 4
              For j = col To col + 4
                 aux = aux + imagen_i2(i, j)
              Next j
            Next i

            imagen_peque(ren1, col1) = Round(aux / 25)

           If Round(aux / 25) < umbral Then
            imagen_umbral(ren1, col1) = 0
           Else
            imagen_umbral(ren1, col1) = 255
           End If
              col1 = col1 + 1
            Next col
           ren1 = ren1 + 1
        Next ren
```