IMPLEMENTACIÓN DE LA TRANSFORMADA DE HOUGH PARA LA DETECCIÓN DE LÍNEAS PARA UN SISTEMA DE VISIÓN DE BAJO NIVEL

EMMANUEL OSPINA PIEDRAHÍTA
JUAN PABLO URREA DUQUE

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MANIZALES
FACULTAD DE INGENIERÍA Y ARQUITECTURA
DEPARTAMENTO DE ING. ELÉCTRICA,ELECTRÓNICA Y COMPUTACIÓN
2002

IMPLEMENTACIÓN DE LA TRANSFORMADA DE HOUGH PARA LA DETECCIÓN DE LÍNEAS PARA UN SISTEMA DE VISIÓN DE BAJO NIVEL

EMMANUEL OSPINA PIEDRAHÍTA
JUAN PABLO URREA DUQUE

Informe del proyecto final para optar al título de Ingeniero Electrónico

Director
FLAVIO PRIETO
Ingeniero Electrónico

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE MANIZALES
FACULTAD DE INGENIERÍA Y ARQUITECTURA
DEPARTAMENTO DE ING. ELÉCTRICA,ELECTRÓNICA Y COMPUTACIÓN
2002

TABLA DE CONTENIDO

LISTA DE FIGURAS

OBJETIVOS

GENERAL

Desarrollar un sistema de vision de bajo nivel para la detección de líneas utilizando una cámara QuickCam y un microcontrolador de la familia MCS-51, el cual indicará la dirección a seguir en una trayectoria por medio de una señal de control para dos motores de un móvil.

ESPECÍFICOS

- Implementar algoritmos de procesamiento de imágenes en el microcontrolador con los cuales se obtenga una imagen binaria.
- Analizar el desempeño del algoritmo de la transformada de Hough con cálculos sin punto flotante.
- Implementar un algoritmo que permita seguir una trayectoria.

INTRODUCCIÓN

El objeto de este trabajo es desarrollar un sistema de visión de bajo nivel, para la detección de líneas utilizando, una cámara digital QuickCam (Blanco y Negro, de puerto paralelo) y un microcontrolador de la familia MCS-51 el cual, mediante un algoritmo, indicará la dirección a seguir en una trayectoria (que solo tiene ángulos rectos) por medio de una señal de control para los motores de un móvil. El programa se basa en la transformada de Hough de una imagen, de la que se obtiene el ángulo aproximado de la recta que contenga ésta, con lo cual se calcula la dirección a seguir.

Los pasos a seguir para la realización de este sistema de visión son los siguientes:

- Adquisición de la imagen.
- Preprocesamiento de la imagen.
- Descripción de la imagen (transformada de Hough).
- Definir trayectoria.

Para el proyecto primero se desarrollan algoritmos en lenguaje de alto nivel para la etapa del procesamiento de la imagen, para después traducirlos a lenguaje ensamblador y comparar los resultados obtenidos.

JUSTIFICACIÓN

En el área de robótica es necesario tener sistemas de visión de bajo costo para identificar objetos primarios, como lo son figuras geométricas, líneas, etc., que permitan seguir trayectorias o evadir obstáculos.

El procesamiento de imágenes proporciona algoritmos con los que se pueden identificar líneas, como lo es la transformada de Hough, la cual necesita de cálculos con punto flotante, pero en la práctica no se dispone de sistemas económicos con esta opción.

Los microcontroladores de la serie MCS-51 son un sistema de bajo costo, que permiten realizar algoritmos con una precisión menor pero con resultados aceptables para el propósito que se busca, además es de uso generalizado, de programación sencilla y con posibilidades para aplicaciones futuras.

CAPÍTULO 1

ADQUISICIÓN DE IMÁGENES

Este proyecto toma como punto de partida el trabajo de grado "Diseño e implementación de un sistema de adquisición de imágenes vía puerto serie ´´en el cual un microcontrolador de la familia MCS-51 (80c32) obtienen imágenes de una cámara (QuickCam-B&N) con una resolución de 4 bits (16 niveles de gris) y las transmite utilizando interfase serial. En este proyecto se desarrollaron dos programas:

1.) Uno en lenguaje ensamblador para manejar el protocolo de comunicación entre el microcontrolador y la cámara, para obtener y guardar una imagen de ésta en la memoria externa del microcontrolador (una extensa explicación sobre el funcionamiento de este microcontrolador, el set de instrucciones, su arquitectura y sus periféricos se encuentran en el libro [4]);

2.) Uno en Delphi 3 para la recepción de datos provenientes de la memoria del microcontrolador via puerto serie (utilizando el control ActiveX MsComm 5.0 de Microsoft) y la visualización de la imagen en el PC, que tiene las siguientes características:

- Permite configurar los parámetros de la cámara (Contraste, Offset, Exposición) durante su funcionamiento.
- Permite capturar imágenes y archivarlas.
- Permite comprobar la integridad de la tarjeta de adquisición.
- Brinda la posibilidad de elección entre diferentes tamaños de imagen.

Un análisis detallado de la tarjeta de adquisición y el software implementado se encuentran en [5] donde se incluyen los diagramas esquemáticos de los circuitos interfase y el código fuente para cada uno de los programas (ensamblador y Delphi).

Puesta en funcionamiento del hardware y software para la adquisición de imágenes

Hardware

1.) Interfase entre el microcontrolador con la memoria externa de 2k para manipular imágenes pequeñas de 1024 píxeles, el diagrama esquemático se encuentra en el apéndice D.

2.) Interfase entre el microcontrolador y la cámara para una resolución de 4 bits [5].

3.) Interfase serial entre el microcontrolador y el PC [5].

En la figura 1.1 se muestra la configuración general del hardware utilizado.

Software

1.) Cargar el programa en ensamblador en el microcontrolador.

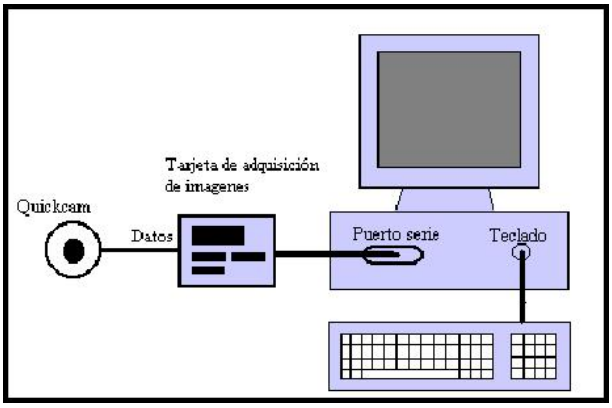


Figura 1.1: Configuración QuickCam-Tarjeta-Pc

2.) Ejecutar el programa interface en Delphi para la visualización de las imágenes.

Modificaciones para el desarrollo específico

1.) Se realizó la actualización del programa de Delphi 3 a Delphi 5, al igual que la componente para la comunicación serial MsComm 5.0 a MsComm 6.0.

2.) Se realizaron pruebas con el analizador lógico para determinar la escala de grises real de la cámara, ya que la utilizada en el programa en Delphi no permitía visualizar la imagen de forma correcta.

3.) Se suprimieron las opciones para cambio de tamaño en la imagen (se mantiene constante la dimensión, 32 x 32 píxeles) y la elección de puerto de comunicación.

4.) Se mejoró la velocidad de transmisión serial de 19200 baudios a 38400 baudios, estos cambios se realizaron en ambos programas (ensamblador y Delphi).

5.) Se ajustaron los parámetros de nivel de negro (Offset), exposición y contraste en el programa para obtener imágenes de buena calidad.

En las figuras 1.2 y 1.3 se muestran la interfase de usuario del programa en Delphi y varias imágenes obtenidas con éste.

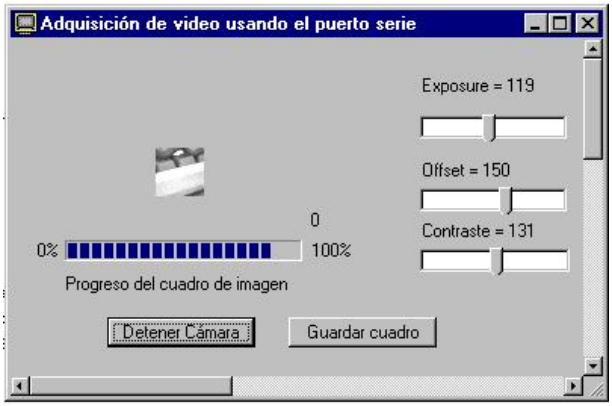


Figura 1.2: Interfase usuario

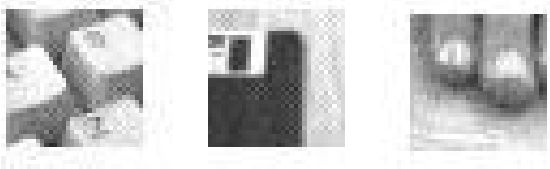


Figura 1.3: Imágenes obtenidas con el programa (1024 píxeles)

La utilización de éste programa en Delphi 5 es clave, para visualizar, evaluar y corregir los resultados de los algoritmos implementados en el microcontrolador, además permite guardar imágenes para comparar el desempeño del sistema de visión con programas que tienen la mismas rutinas en lenguaje de alto nivel (Builder C++ 4), y su código fuente se encuentra en el CD adjunto a la tesis (hough.exe).

CAPÍTULO 2

PREPROCESAMIENTO DE LA IMAGEN

La imagen tomada en el proceso de adquisición es una línea negra con un fondo blanco, la cual es la una parte de la trayectoria a seguir. Por lo tanto se necesita extraer la linea (umbralización) y obtener sus características (detección de bordes), para realizar la descripción de ésta de una manera más simple.

2.1. Umbralización del Histograma

Supóngase una imagen como la presentada en la figura 2.1, la cual contiene una cinta negra sobre un fondo blanco y su correspondiente histograma de niveles de gris (figura 2.2), en donde se muestran dos regiones bien diferenciadas. Una forma clara de separar la cinta del fondo es elegir un umbral T que separe dichas regiones, es decir, un punto (x, y), tal que, $f(x, y) > T$ sea un punto que pertenezca a la cinta. De lo contrario este punto será del fondo. Entonces una imagen umbralizada $g(x, y)$

se define en 2.1.1 (Ver [1]).

$$g(x,y) = \begin{cases} 1 \text{ si} & f(x,y) > T \\ 0 \text{ si} & f(x,y) \leq T \end{cases} \tag{2.1.1}$$

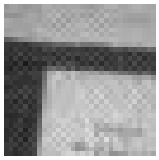


Figura 2.1: Imagen con 16 niveles de gris (1024 píxeles)

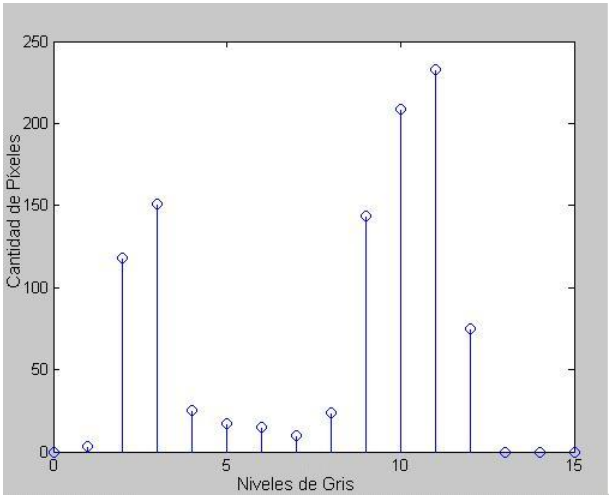


Figura 2.2: Histograma de la figura 2.1

En el caso particular de la imagen de la figura 2.1 se calculó su histograma (con el comando *hist* en Matlab, figura 2.2), en el cual se notan dos lóbulos bien diferenciados (fondo y linea) por un valle que está entre los niveles de gris 4 y 8. Se escogió como umbral $T = 5$, para que se obtenga una imagen binaria donde 1 es el fondo y 0 la cinta utilizando la definición de la ecuación 2.1.1. Esto se aplica en el sistema de

visión de la siguiente manera: se recorre la imagen que está en la memoria externa del microcontrolador y se cambia el valor de cada píxel (es decir cada posición de memoria), dependiendo si está por encima o por debajo del umbral obteniéndose una imagen binaria.

2.2. Detección de bordes

Un borde es el límite entre dos regiones con niveles de gris distintos, de manera que estos sean los suficientemente homogéneos, para que la transición entre dichas regiones se pueda mirar como un cambio abrupto. La idea básica para la detección de bordes es el cálculo de una derivada, teniendo en cuenta que para una constante su valor es cero y para un cambio será diferente de cero. Es decir, en las regiones donde no exista un cambio brusco el valor de la derivada es cero y para la transición entre regiones su valor es una constante. La primera derivada de un punto de una imagen se obtiene utilizando el módulo del gradiente de ese punto y, la segunda derivada, se obtiene de forma similar utilizando el Laplaciano.

2.2.1. Laplaciano

El Laplaciano es un operador escalar de segunda derivada para funciones de dos dimensiones definido por la ecuación 2.2.1:

$$\nabla^2 f(x,y) = \frac{\partial^2}{\partial x^2} f(x,y) + \frac{\partial^2}{\partial y^2} f(x,y) \tag{2.2.1}$$

La ecuación se implementa en forma digital haciendo la convolución de una imagen con una región o máscara de 3x3 como la figura 2.3.

El principio para la definición de este operador es que el coeficiente asociado al píxel central sea positivo y los coeficientes asociados a los píxeles exteriores sean

z1	z2	z3
z4	z5	z6
z7	z8	z9

Figura 2.3: Región de imagen de 3x3

negativos, tal que la suma de todos los coeficientes sea cero. Al hacer la convolución para cada punto de la imagen con cualquiera de las máscaras de la figura 2.4, la respuesta es cero siempre que el punto central tenga el mismo valor que sus vecinos (ver libros [1] y [2]). Ya que la aplicación de este operador es sensible al ruido, se utiliza después de binarizar la imagen para mejorar el resultado.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Figura 2.4: Operadores Laplacianos

2.2.2. Algoritmo de detección de bordes

Para el caso particular de una imagen binaria, se tomó como base el operador Laplaciano con elemento central (z_5)=4 y cuya ecuación es la 2.2.2, el cual requiere menos cálculos que el de elemento central (z_5)=8 (ecuación 2.2.3). En cualquiera de las ecuaciones el valor del operador es cero cuando la región de 3x3 tenga píxeles iguales y una constante positiva o negativa al estar sobre el borde; el concepto que se evidencia es que si el valor del píxel central es diferente de alguno de sus 4 u 8 vecinos

se detecta un borde.

$$\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8) \tag{2.2.2}$$

$$\nabla^2 f = 8z_5 - (z_1 + z_2 + z_3 + z_4 + z_6 + z_7 + z_8 + z_9) \tag{2.2.3}$$

En el sistema de visión, para cada píxel de la imagen binarizada (que se encuentra en la memoria externa del microcontrolador) se implementa el algoritmo de la siguiente manera:

- Se recorre la imagen (la memoria), buscando un píxel negro, es decir, si es blanco se omite el calculo.
- Cuando lo encuentra busca en los 4 vecinos por lo menos un píxel blanco el cual activa una bandera que indica que hay un borde. Si todos los 4 vecinos son negros no es borde.

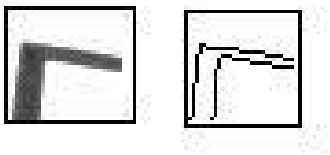


Figura 2.5: Ejemplo de aplicación de detección de bordes

El resultado del algoritmo se observa en la figura 2.5. Por lo tanto se hace una reducción importante de tiempo al hacer una variación de la ecuación 2.2.2 ya que se evitan los siguientes cálculos en el algoritmo:

- Sólo se buscan los píxeles negros en la imagen para la detección del contorno evitando los blancos.
- No se realizan sumas ni multiplicaciones.

Es de notar que el uso que se le da a la detección de bordes es disminuir la cantidad de puntos en la imagen manteniendo la forma de la línea obtenida.

CAPÍTULO 3

TRANSFORMADA DE HOUGH

La transformada de Hough es una técnica utilizada para aislar características de forma particular dentro de una imagen. La idea básica es encontrar curvas que puedan ser parametrizadas como líneas rectas, polinomios y círculos.
Se puede analíticamente describir un segmento de línea en varias formas. Sin embargo una ecuación conveniente para describir un conjunto de líneas es la notación paramétrica o normal:

$$\rho = x\cos\theta + y\sin\theta \tag{3.0.1}$$

Donde ρ es la longitud de una normal desde el origen hasta la línea y θ es el ángulo de ρ con respecto al eje x. Cualquier línea recta en una imagen es representada en un punto simple (ρ_1, θ_1) en el espacio paramétrico (ρ, θ).
Puntos colineales (x_i, y_i) con $i = 1, 2, 3 \ldots N$ son transformados en N curvas sinusoidales en el plano (ρ, θ). Figura 3.1.

$$\rho = x_i\cos\theta + y_i\sin\theta \tag{3.0.2}$$

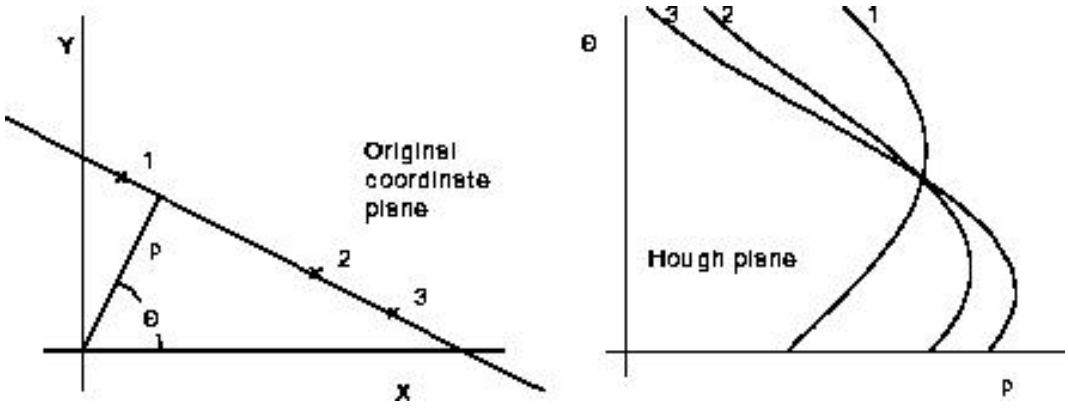


Figura 3.1: Representación gráfica de la transformada de Hough

Los puntos de intersección de las curvas en el espacio paramétrico, corresponden a los parámetros (ρ_k, θ_k) de las posibles rectas que se encuentran en la imagen.
El espacio paramétrico es representado por una estructura rectangular de celdas, llamada arreglo acumulador y cuyos elementos son las celdas acumuladoras $A(\rho_i, \theta_i)$, las cuales son los rangos esperados de (ρ, θ). Las celdas acumuladoras con una magnitud superior a un cierto umbral pueden ser consideradas como posibles líneas (ver libro [1]).

3.1. Algoritmo de Hough

Sea E una imagen binaria de tamaño MxN, en la cual cada píxel de la imagen es uno o cero. Sean ρ_d, θ_d vectores que contienen valores discretos del espacio paramétrico ρ, θ ($\rho \in [0, (M^2 + N^2)^{1/2}]$ y $\theta \in [0, 180]$). Donde R y T son el número de elementos de ρ y θ respectivamente. Ver libro [3].

3.1.1. Descripción del algoritmo

1.) Discretize el espacio parámetro ρ, θ.

2.) Sea $A(R, T)$ una matriz de contadores inicializada en 0.

3.) Para cada píxel $E(i, j)$ que pertenezca a la línea desde $h = 1, 2, \ldots T$ con $k = 1, 2, \ldots K$.

   a) Hacer $\rho = i \cos \theta_d(h) + j \sin \theta_d(h)$.

   b) Halle el intervalo K que sea mas cercano al ρ obtenido.

   c) Incremente matriz de contadores $A(k, h)$.

4.) Encuentre el máximo local (k_p, h_p) tal que sea mayor que un umbral.

5.) Con el ρ y θ encontrados se describe la línea en forma polar.

Como ejemplo en la figura 3.2 se tiene una imagen con una línea de aproximadamente 20^o a la cual se le aplica el algoritmo de la transformada de Hough. En la figura 3.3 se muestran las diferentes curvas en el espacio paramétrico de la imagen 3.2. En la intersección, ρ es la magnitud del vector normal desde el origen a la recta y θ, el ángulo de inclinación. El código fuente de este programa hecho en Matlab se encuentra en el apéndice C, el cual tiene la misma base del algoritmo implementado en el microcontrolador.

3.1.2. Implementación del algoritmo en el microcontrolador

1.) Como entrada se tiene una imagen ($M = 30 \;\; x \;\; N = 30$) de la cámara que se guarda en la memoria externa del microcontrolador (ver libro [4]). Se tienen

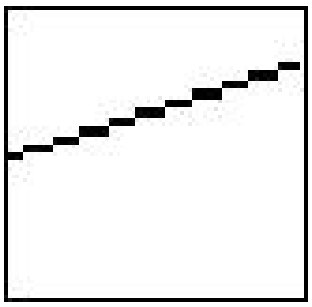


Figura 3.2: Imagen con línea de aprox. 20 grados

discretizados los valores de ρ con 18 elementos ($\rho \in [0 \ldots 92]$ *con intervalos de 5 unidades*) y θ con 9 elementos ($\theta \in [0^o, 30^o, 45^o, 60^o, 90^o, 120^o, 135^o, 150^o, 180^o]$).

2.) Se guardan en un vector las posiciones (x, y) de los puntos (píxeles negros), que pertenezcan a una línea y en un contador (NXY) se guarda la cantidad de éstos.

3.) Debido a que el microcontrolador no realiza operaciones de punto flotante, se deben multiplicar el seno y coseno de θ por un factor 10, obteniendo las siguientes tablas: para el seno (0, 5, 7, 8, 10, 8, 7, 5, 0), y el coseno (10, 8, 7, 5, 0, 5, 7, 8, 10). Además se define la tabla de ρ_k de la siguiente manera (0, 4, 5, 10, 11, 15, 16, 20, 21, 25, 26, 30, 31, 35, 36, 40, 41, 45, 46, 50, 51, 55, 56, 60, 61, 65, 66, 70, 71, 75, 76, 80, 81, 85, 86, 92).

4.) No se utiliza una matriz de contadores por limitaciones de espacio en la memoria del microcontrolador por tanto se implementa cambiando la forma de incrementar los acumuladores en un solo vector. Es decir, se calculan los valores de ρ variando todos los puntos (x_i, y_i) con un θ fijo. El vector de contadores se inicializa en cero.

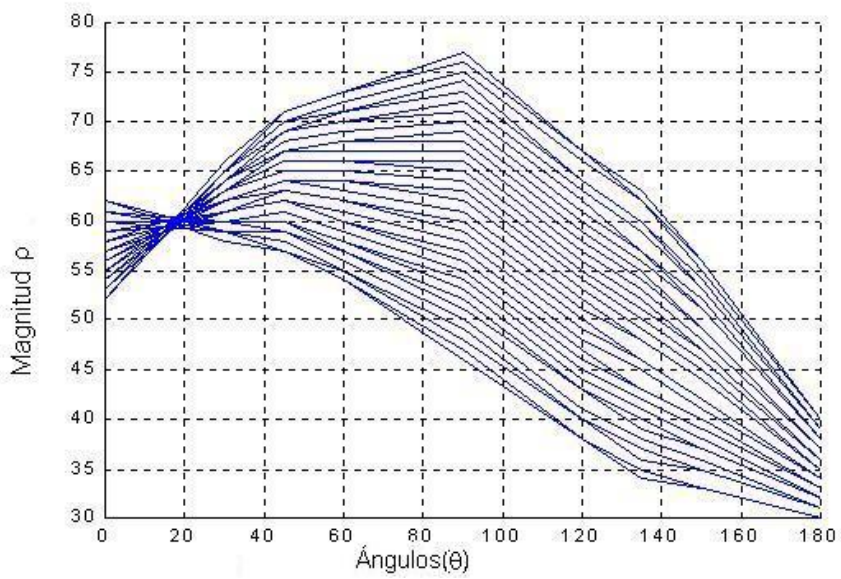


Figura 3.3: Transformada de Hough con línea de 20 grados

5.) Para $\theta(h)$ desde $i = 1 \ldots NXY$, *con* $h = 1 \ldots 9$, hacer:

$\rho = x_i \cos\theta_d(h) + y_i \sin\theta_d(h)$

donde $\cos\theta_d(h)$ y $\sin\theta_d(h)$ son los valores de las tablas. Se halla el intervalo ρ_k que sea más cercano al ρ obtenido. Se incrementa la posición k del vector de contadores que corresponde al intervalo ρ_k obtenido.

6.) Para el ángulo en que está, se busca el intervalo ρ_k que tenga el mayor acumulado. Se compara con el mayor anterior, se actualiza si éste es superior y se guarda la posición k del intervalo ρ_k y el θ actual. Esto se realiza para cada uno de los ángulos.

7.) El último θ que se actualiza en el paso anterior es el ángulo de la recta. El diagrama de flujo de los algoritmos implementados en el microcontrolador se

encuentra en el apéndice A

Para este algoritmo se realizaron pruebas en un simulador de Ceibo para Windows que se encuentra en el sitio web [6], el cual permite cargar la memoria externa del simulador desde un archivo *.hex* que contenga los datos de la imagen, para determinar que los resultados de ρ y θ fueran correctos se compararon con los obtenidos en *hough.exe* ya que tienen la misma rutina. En las figuras 3.4 y 3.5 se muestran los ángulos de las líneas encontrados de varias imágenes y una imagen con una linea de 30^o más ruido incluyendo el resultado de los vectores acumuladores.

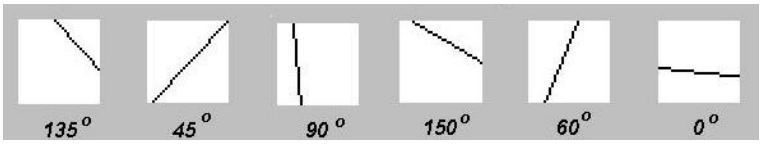
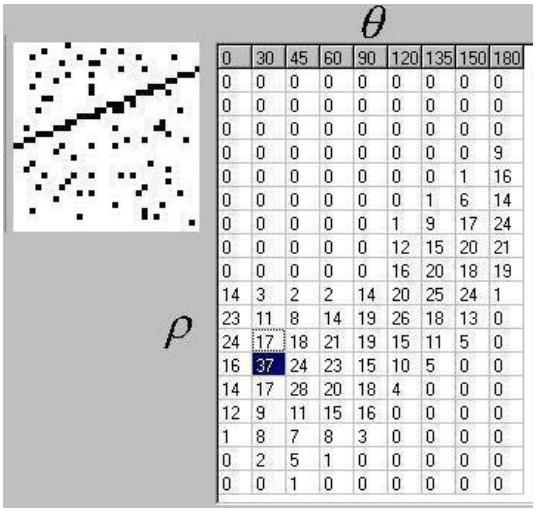


Figura 3.4: Imágenes de prueba



0	30	45	60	90	120	135	150	180
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	9
0	0	0	0	0	0	0	1	16
0	0	0	0	0	0	1	6	14
0	0	0	0	0	1	9	17	24
0	0	0	0	0	12	15	20	21
0	0	0	0	0	16	20	18	19
14	3	2	2	14	20	25	24	1
23	11	8	14	19	26	18	13	0
24	17	18	21	19	15	11	5	0
16	37	24	23	15	10	5	0	0
14	17	28	20	18	4	0	0	0
12	9	11	15	16	0	0	0	0
1	8	7	8	3	0	0	0	0
0	2	5	1	0	0	0	0	0
0	0	1	0	0	0	0	0	0

Figura 3.5: Imagen con ruido

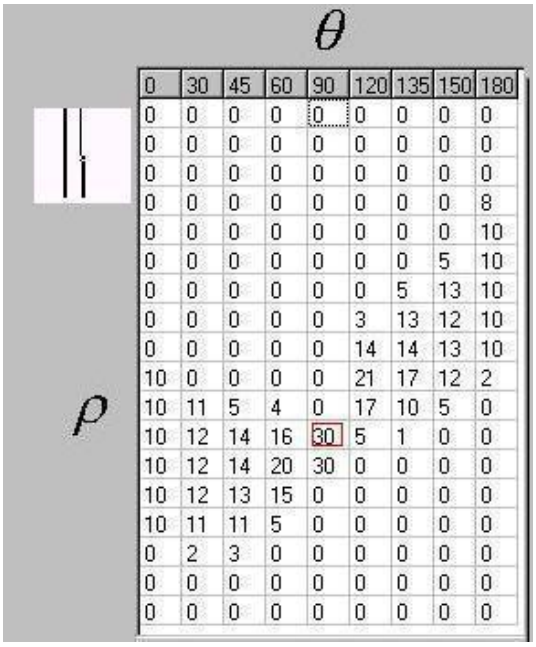
CAPÍTULO 4

CALCULO DE LA TRAYECTORIA

Tomando los resultados de la transformada de Hough: el ángulo y los valores de ρ mayores para cada ángulo, se define una trayectoria a seguir, que para este caso serán tres direcciones (adelante, izquierda o derecha) ya que se espera probar el prototipo en una ruta que solo tenga ángulos rectos.

Dirección adelante: Cuando se tenga un máximo en la celda que corresponde al ángulo de $\theta = 90^o$, se tomará la desición de ir hacia adelante, con señales iguales para los motores. Figura 4.1.

Dirección izquierda o derecha: Al acercarse a una esquina se tienen 2 líneas cruzadas, una con ángulo de 90^o y la otra con 0^o o 180^o. El móvil avanzará hasta que la línea mayor sea la de 0^o o 180^o, e indicará que se realizará un giro. La transformada de Hough cuando encuentra una esquina no entrega un resultado confiable de si el ángulo está a 180^o (que significaría izquierda) o a 0^o (que significaría derecha) ya que realmente son los mismos. Pero sí entrega información sobre la posible dirección a seguir, utilizando el concepto de la transformada, el cual, genera líneas de cada uno de los puntos hacia los demás. Un ejemplo se puede ver en la figura 4.2 donde se



0	30	45	60	90	120	135	150	180
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	8
0	0	0	0	0	0	0	0	10
0	0	0	0	0	0	0	5	10
0	0	0	0	0	0	5	13	10
0	0	0	0	0	3	13	12	10
0	0	0	0	0	14	14	13	10
10	0	0	0	0	21	17	12	2
10	11	5	4	0	17	10	5	0
10	12	14	16	30	5	1	0	0
10	12	14	20	30	0	0	0	0
10	12	13	15	0	0	0	0	0
10	11	11	5	0	0	0	0	0
0	2	3	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figura 4.1: Imagen con línea de 90 y celdas acumuladoras

trazan líneas para el caso de las dos posibles direcciones. Se aprecia en las imágenes que las líneas tienen una inclinación en la dirección de giro. Es decir, entre más líneas haya en un sentido, el giro se dará hacia éste. Para comprobar este concepto, se observan los máximos en las celdas acumuladoras que significan líneas trazadas en un ángulo particular. Para el caso de un giro hacia la izquierda (ver figura 4.3.a) se ven resaltados los máximos acumuladores de los ángulos 120^o, 135^o y 150^o de color rojo y con azul los de 30^o, 45^o y 60^o que indican un giro a la derecha. Se aprecia que la mayor cantidad se encuentra hacia la izquierda en la figura 4.3.a y hacia la derecha en la figura 4.3.b. Por lo tanto para encontrar la dirección:

Derecha: Se suman los máximos acumuladores de los ángulos 30^o, 45^o y 60^o.

Izquierda: Se suman los máximos acumuladores de los ángulos 120^o, 135^o y 150^o.

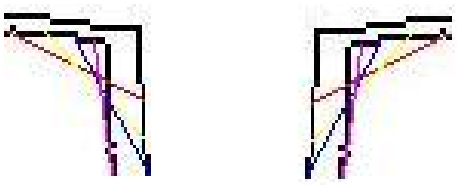


Figura 4.2: Imagen de esquinas, con diferentes líneas

Después se encuentra el mayor entre las dos sumatorias, el cual, indica la dirección a seguir, proporcionando las señales correspondientes a los motores. En la figura 4.5 se observa el diagrama general del sistema de visión. En el apéndice B.1 se muestra el móvil en el cual se probaron los algoritmos. El código fuente de todos los programas se encuentra en el CD adjunto al documento.

Resultados

Se realizaron pruebas con 15 imágenes para cada una de las direcciones tomando en cuenta: ruido por cambios en la luminosidad, y huecos en las líneas. Para cualquiera de los casos, un cambio de luminosidad afecta el cálculo ya que aparece demasiado ruido o la imagen puede llegar a desaparecer. En consecuencia se debe tener en cuenta una iluminación constante y ajuste en los parámetros de la cámara para obtener la mejor línea. En la dirección adelante (ángulo=90^o) se tiene un 100 % de acierto con huecos en la línea. Para los otros casos se muestran las imágenes donde hubo error. La figura 4.4 muestra varias imágenes donde el sentido de giro calculado difiere del que realmente es a simple vista. Por lo tanto, huecos en ciertas partes de la línea afectan el cálculo ya que se pierde información. Esto se debe a cambios de iluminación.

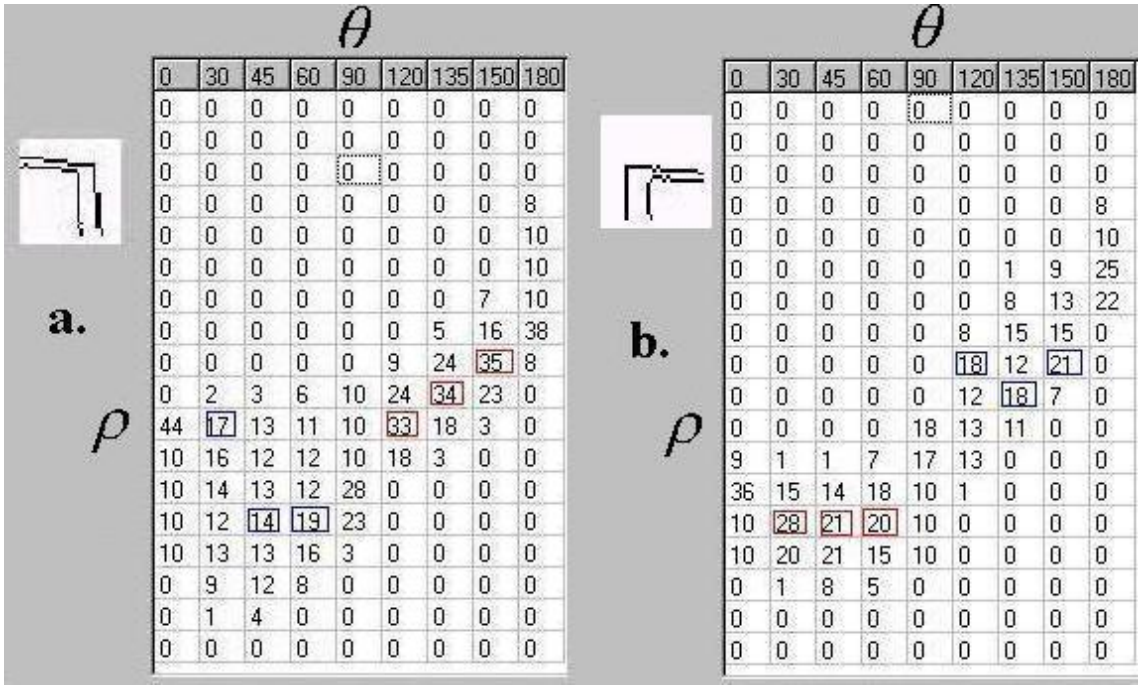


Figura 4.3: Esquinas y celdas acumuladoras (Giro izquierda y derecha)

Figura 4.4: Giro erróneo

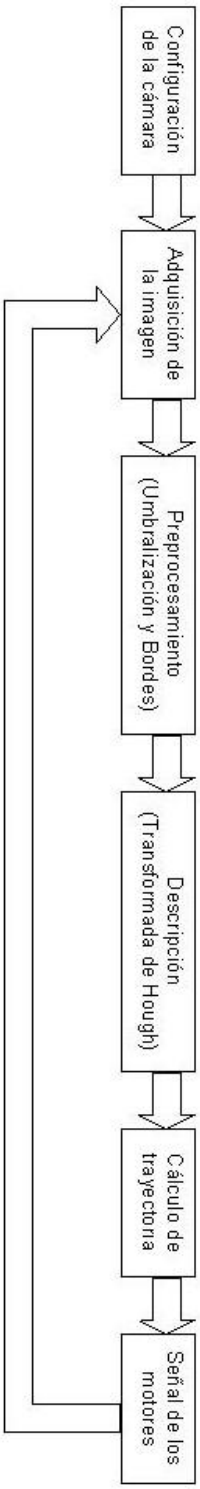


Figura 4.5: Diagrama general del sistema de visión

CONCLUSIONES

- La dificultad para el desarrollo de software en el microcontrolador es por falta de una herramienta de depuración en linea, que permita visualizar las variables cuando se utilizan interrupciones y temporizadores.

- La elaboración de programas de apoyo en un lenguaje de alto nivel facilita la implementación de los algoritmos en ensamblador, además permite visualizar y comparar los resultados de manera gráfica.

- La comunicación serial con el microcontrolador permite evaluar cada uno de los algoritmos utilizados, es decir, posibilita de alguna forma la depuración del código en ensamblador.

- La imagen binaria que se obtiene es de buena calidad ya que ésta tiene un alto contraste con lo cual al umbralizar se puede quitar ruido, aunque se necesita una calibración previa de los parámetros de la cámara para conseguir buenos resultados, manteniendo una iluminación constante.

- La implementación de el Laplaciano se realiza de forma sencilla ya que es específico para imágenes con poco ruido; sin embargo, en el caso contrario, el algoritmo encontraría demasiados contornos. Para la detección de la línea no es necesario utilizar un detector de bordes, esto se realiza por la limitación en el

cálculo de las celdas acumuladoras en la transformada de Hough dado que éstas son de 1 Byte.

- Implementar filtros en un sistema de visión de bajo nivel puede ser limitado por la complejidad del código debido al manejo de matrices, al hacer la convolución de una máscara con la imagen.

- El algoritmo de la transformada de Hough se desempeña bien en imágenes con poco ruido. Las imágenes no deben tener demasiado ruido o más de 256 píxeles que pertenezcan a una línea ya que cada celda acumuladora es de 8 bits. Para disminuir este problema se aplica previamente la detección de bordes para mantener la forma de la línea pero con menos puntos. Aunque por limitaciones de punto flotante se utilizan solo 9 ángulos, al aumentar los intervalos posibles de ρ se obtiene una mayor exactitud al hallar el θ esperado, inclusive si se toman más puntos de una imagen.

- El algoritmo de seguimiento de la trayectoria presenta un buen desempeño en la dirección adelante. En las esquinas aunque las detecta de forma correcta presenta problemas por la distancia entre el ángulo y el móvil ya que no es siempre la misma.

- Además de seguir una trayectoria estos algoritmos se pueden utilizar en segmentación, descriptores de contorno y de región, y caracterización tomando en cuenta las limitaciones de procesamiento y de memoria del microcontrolador.

BIBLIOGRAFÍA

[1] Rafael C. Gonzales y Richard E. Woods. *Tratamiento digital de imágenes*. Addison-Wesley, 1992

[2] Kenneteh R. Castleman. *Digital Image Processing*. Prentice Hall, 1996

[3] Emmanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998

[4] José Adolfo Gonzáles Vázquez. *Introducción a los microcontroladores*. McGraw-Hill, 1992

[5] Jorge Hernando Rivera Piedrahíta. *Diseño e implementación de un sistema de adquisisción de imágenes via puerto serie*. Universidad Nacional de Colombia, Manizales. Tesis de Grado. 2001.

[6] http://www.ceibo.com

APÉNDICE A

Diagramas de flujo

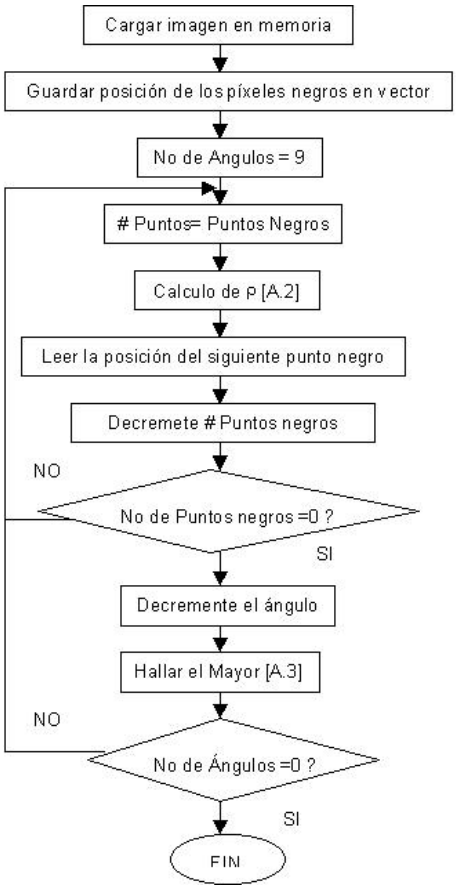


Figura A.1: Algoritmo transformada de Hough

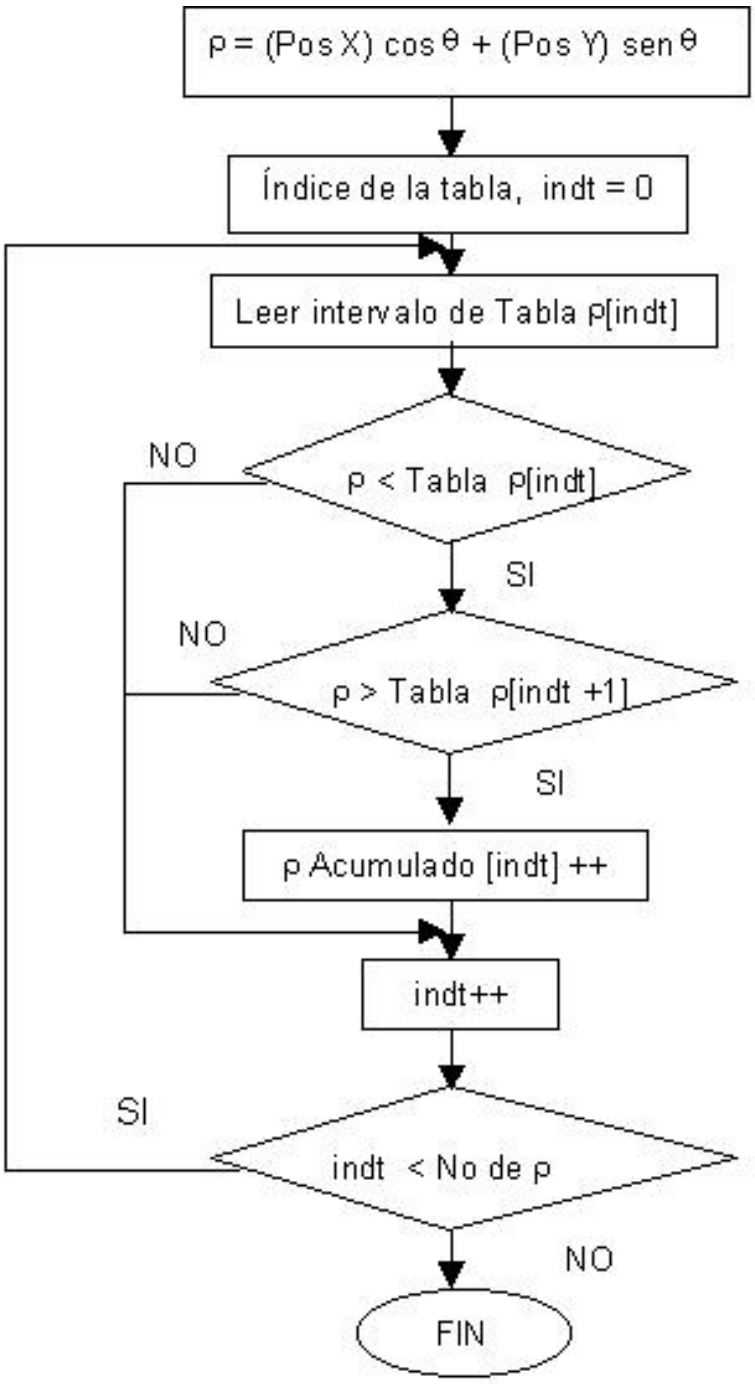


Figura A.2: Algoritmo cálculo de ρ

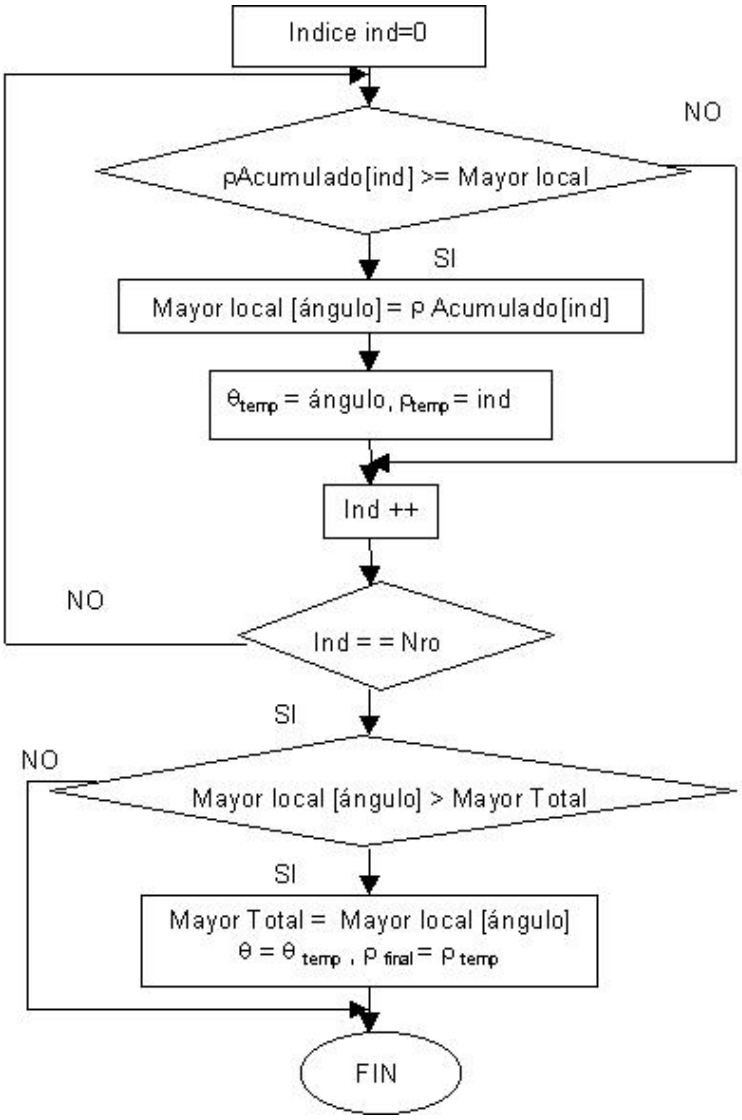


Figura A.3: Algoritmo cálculo del mayor

APÉNDICE B

Móvil

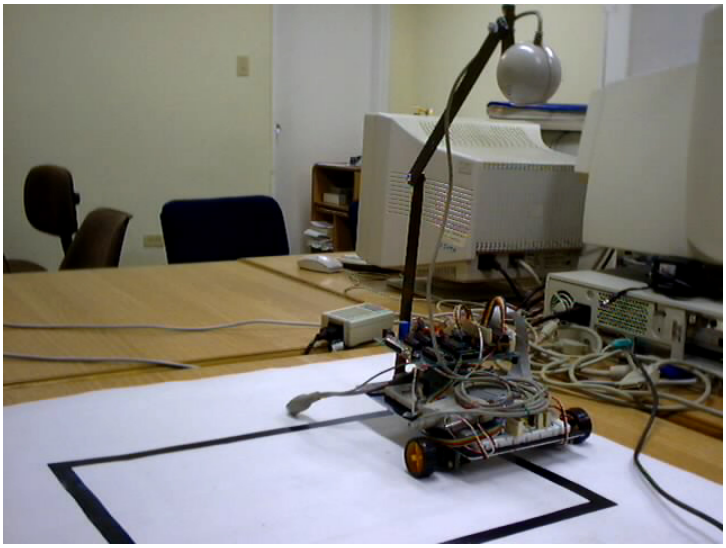


Figura B.1: Móvil desarrollado

En la figura B.1, el móvil incluye el microcontrolador, la cámara con su interfase y dos servomotores (figura B.2) con los cuales se avanza en la dirección que entrega el algoritmo de la trayectoria.

Los servomotores funcionan con una señal de PWM como se muestra en la figura B.3. Estos servomotores están ya adaptados para que giren continuamente y para

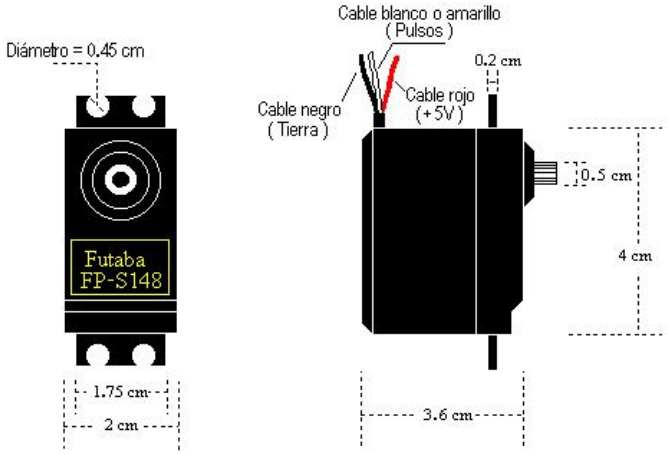


Figura B.2: Servomotores

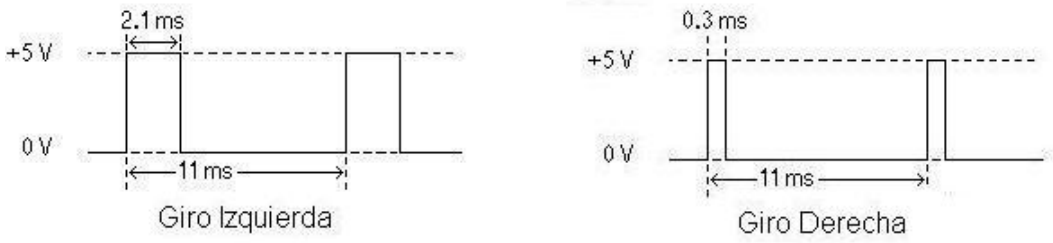


Figura B.3: Señales de control para los servomotores

cada tipo de servo que se desee controlar, se deberá realizar una prueba preliminar para encontrar exactamente el período y la duración de los pulsos que mejor funcione. Tomando como referencia la parte de atrás del móvil hacia la cámara, para ir hacia adelante se envían los pulsos (desde el microcontrolador) correspondientes a derecha e izquierda respectivamente. Para realizar un giro hacia la derecha se envían los pulsos para que el motor izquierdo gire mas rápido que el derecho y para el giro hacia la izquierda se hace la operación inversa.

APÉNDICE C

Código Fuente Matlab

```
%Transformada de Hough, imagen en datos.dat
fid = fopen('c:\datos.dat', 'r'); k=1; j=0;
for x=1:32
 for y=1:32
   pix=fscanf(fid,'%c',1);
   if pix == '0'
      px(k)=x-1; py(k)=y-1; k=k+1; j=j+1;
    end
  end
end
fclose(fid); t=0:pi/12:pi;
 c=[10 8 7 5 0 -5 -7 -8 -10]; %fix(10*cos(t));
 s=[0 5 7 8 10 8 7 5 0]; %fix(10*sin(t));
 for h=1:9
   for o=1:j
      p=px(o)*c(h)+py(o)*s(h);
      y(h,o)=fix((460+p)/10);
   end
  end

t=[0 30 45 60 90 120 135 150 180]; %t*180/pi;
hold on
  for o=1:j
   plot(t,y(:,o));
end
 grid on;
```

APÉNDICE D

Acceso a memoria externa

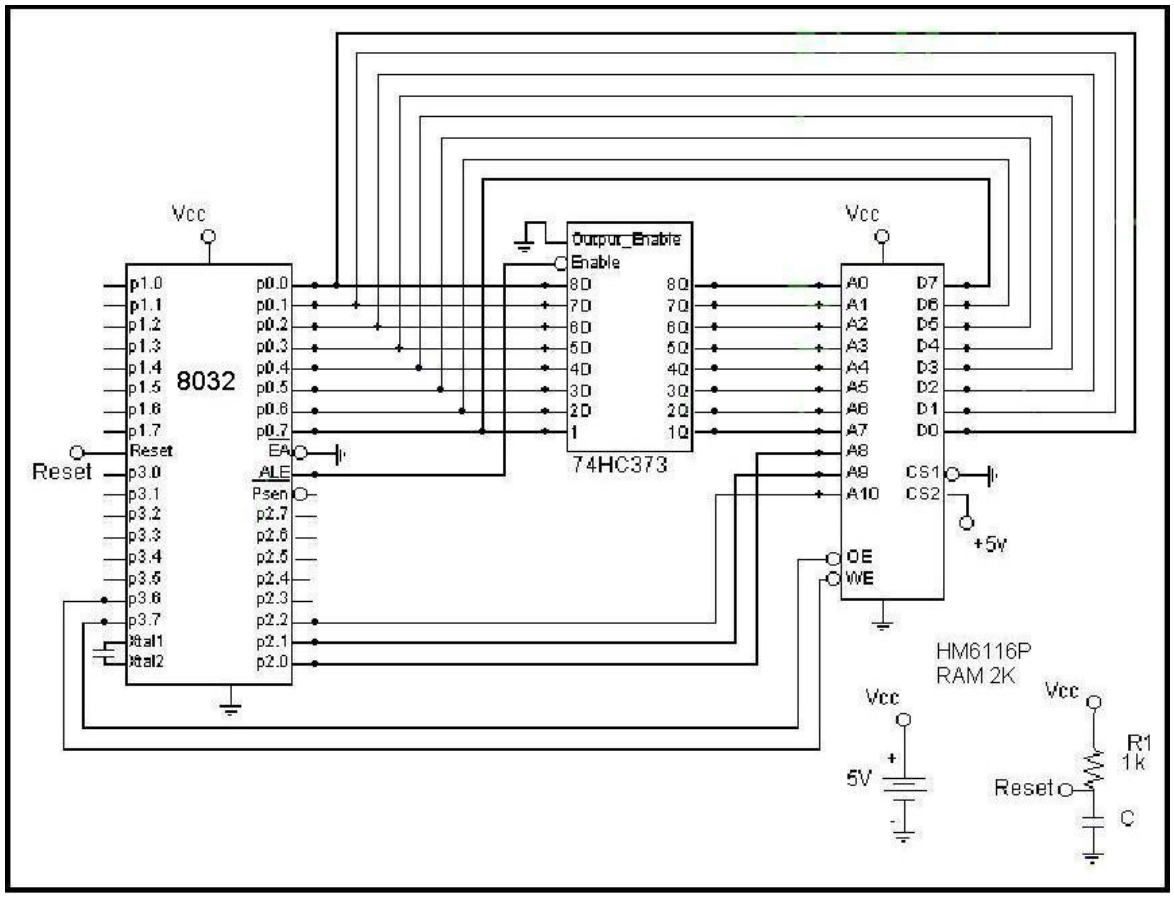


Figura D.1: Circuito esquemático memoria de datos externa