Sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos

UPC
UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS
Laureate International Universities®

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA

Sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos

TESIS

Para optar por el título profesional de:

INGENIERO ELECTRÓNICO

AUTORES:

Aranguren Zapata, Alejandro

Vela Asin, Tiffany

ASESOR:

Luis del Carpio

LIMA - PERÚ

2012

A nuestros padres,

por todo su amor, paciencia y apoyo incondicional.

AGRADECIMIENTOS

Agradecemos a nuestros padres ya que sin su apoyo incondicional durante toda nuestra formación académica hoy no hubiera sido posible culminar el presente trabajo, el cual cierra una de las etapas más importantes de nuestra formación profesional.

A nuestro asesor, Capitán de Fragata Luis del Carpio Azálgara, ya que con sus valiosos aportes hemos podido culminar satisfactoriamente el proyecto.

A la Marina de Guerra del Perú, la Comandancia de Fuerza de Submarinos y la Estación de Armas Submarinas por su apoyo brindado en la elaboración de la parte mecánica del proyecto y darnos la oportunidad de vivir experiencias únicas.

A nuestros profesores, siempre dispuestos a ayudar y solucionar dudas.

A quienes no mencionamos, pero también son parte de este reconocimiento, gracias por el apoyo.

Alejandro y Tiffany

TABLA DE CONTENIDO

RESUMEN

En el presente trabajo se presenta el diseño y desarrollo de un ***Sistema de seguimiento de objetos mediante el procesamiento digital de imágenes aplicado al control de robots autónomos***. El software le permite al usuario seleccionar los colores de la marca que desea monitorear, además le permite cambiar los niveles de luminosidad con los que se presenta la imagen en pantalla.
Una vez iniciado el seguimiento se procede a realizar la adquisición de imágenes mediante una cámara inalámbrica instalada en el robot para luego procesarlas y mostrar en pantalla la marca identificada y centrada. El sistema realiza el seguimiento de la marca ubicada en el objeto de interés siempre manteniéndolo a un margen máximo de un metro.
El sistema está conformado por un robot autónomo capaz de realizar movimientos rotacionales y traslacionales de manera de evitar que la marca se pierda en algún momento. Además el robot cuenta con un sensor de proximidad que le permite evadir obstáculos a medida que realiza algún movimiento.
El sistema de control está monitoreado por dos microcontroladores PIC 16F877A, los cuales gobiernan los actuadores del sistema (motores, servomotor, sensores), además de permitir el intercambio de información con ayuda de dos módulos XBee-PRO instalados uno en el robot y el otro en la estación base. El computador posee un software desarrollado en el entorno MATLAB potenciado con tecnología de procesamiento paralelo a través de una tarjeta de video con tecnología CUDA para optimizar el procesamiento digital de imágenes que permite realizar la identificación del objeto de interés.
Por último se presenta también un sistema capaz de detectar o identificar aves en el cielo, esto con el fin de monitorear campos o pistas de aterrizaje para evitar inconvenientes o retrasos en vuelos como aplicación adicional del sistema.

TABLA DE VARIABLES Y SÍMBOLOS

Tabla de variables y símbolos	
Y	Componente de Luminancia
Cb	Componente de crominancia azul
Cr	Componente de crominancia rojo
R	Componente primario rojo
G	Componente primario verde
B	Componente primario azul
R'	Transformación de R corregido por el factor Gamma
G'	Transformación de G corregido por el factor Gamma
B'	Transformación de B corregido por el factor Gamma
S	Imagen resultado de la transformación Gamma
C	Constante
R	Imagen entrante
γ	Constante Gamma
I_{CPU}	Imagen rasterizada corregida por el factor Gamma
I_m	Imagen resterizada
A	Conjunto
W	Elemento estructurante
X	Elementos de un conjunto
$\ominus$	Símbolo erosión
$\oplus$	Símbolo dilatación
$\circ$	Símbolo de apertura
$\bullet$	Símbolo de cierre
X_k	Imagen binaria producto de la función relleno de regiones
D_e	Distancia euclidiana
p(x,y)	Punto "p" en coordenadas x,y
q(x,y)	Punto "q" en coordenadas x,y
D	Distancia de Mahalanobis
C	Matriz de covarianzas
S[x,y]	Imagen umbralizada
T	Valor umbral
C_x	Centro de masa para coordenadas x
C_y	Centro de masa para coordenadas y
F_c	Frecuencia central
I	Corriente directa
V_D	Voltaje directo
V_{cc}	Voltaje de alimentación en corriente directa
R_x	Resistencia

ID	Corriente de diodo
P	Alimentación
E	Entrada digital
AN	Entrada analógica
O	Salida digital
NC	No conectado
GND	Tierra
V_{out}	Voltaje de salida del regulador
V_{ref}	Voltaje de referencia
I_{ADJ}	Corriente de ajuste
M	Número de filas en una imagen
N	Número de columnas en una imagen
K	Constante de proporcionalidad
Δ%	Variación porcentual
v_f	Valor final
v_i	Valor inicial
r_{ir}	Radio instantáneo de rotación
v_r	Velocidad lineal del motor derecho
v_l	Velocidad lineal del motor izquierdo

CAPÍTULO 1. ASPECTOS INTRODUCTORIOS

El presente capítulo brindará una breve introducción al proyecto a realizar, de manera que el lector se familiarice con los logros que se desean alcanzar, así como también conozca las ventajas y desventajas de este y algunos prototipos existentes en el mercado.

Presentación y justificación del problema

En la actualidad, en el Perú son nulas las aplicaciones capaces de detectar y predecir la trayectoria de un objeto para su seguimiento, los estudios realizados se basan en el análisis de la movilidad de células usando técnicas de procesamiento digital de imágenes[1] así como también estudios de sistema de gestión de tráfico vehicular[2].

Actualmente no se ha desarrollado en el país un sistema de seguimiento de un objeto móvil utilizando procesamiento digital de imágenes autónomo y económico que sirva de ayuda principalmente a la seguridad pública. Además, no se ha explotado la industria de juguetes de alta tecnología con calidad de exportación como el Parrot AR.Drone[3], entre otros.

[1] Cfr. Rojas, Tejada y Milla 2006: 1-3
[2] Cfr. GPDSI 2010
[3] Cfr. Parrot 2010

Por estas razones se propone el diseño y desarrollo de un sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de robots autónomos optimizando algoritmos ya existentes y utilizando tecnología de procesamiento paralelo que resulte económico y de innovación tecnológica ya que se incurre en el uso de una tecnología nueva para el procesamiento de imágenes. Además contará con la opción a ajustar los niveles de brillo para una mejor visualización. Este proyecto es de gran ayuda social, ya que no solo contribuiría a la seguridad pública sino que además se probaría que en el Perú se puede desarrollar tecnología con calidad de exportación.

Estado del arte

Como ya se mencionó, el presente proyecto, ni similares, existen en el país. Muchos sistemas de seguimiento han sido desarrollados mas no implementados por lo cual el presente proyecto no sólo demuestra la capacidad de desarrollar sino de implementar y comercializar un prototipo.

En la actualidad la vigilancia en espacios abiertos se realiza con cámaras estáticas situadas en diferentes puntos lo que permite monitorear gran parte del territorio de manera constante, ya sea de manera alámbrica, inalámbrica o por IP. Sin embargo estas cámaras siempre están limitadas al lugar donde están instaladas así tengan un rango de visión de 360°, así como la limitación del “zoom”.

La mayoría de las aplicaciones para visión de robots autónomos se ve limitada al uso de estos para exploración, desarme de bombas, etc[4], más no para el seguimiento de objetos o monitoreo de actividades.

La mayoría de los sistemas de seguimiento se basan en la detección del movimiento de un objeto dentro de un fondo formado por objetos rígidos de tal manera que un solo objeto genere movimiento[5].

En el mundo existen proyectos que se centran en la detección de movimientos dentro de una imagen para poder realizar el seguimiento de un objeto y estimar de posición de este[6], además para obtener mayor calidad en las imágenes, la cámara que se utiliza en la mayoría de estos proyectos es de mayor costo (aproximadamente € 300.00) y mayor tamaño y se trata de tener un fondo no variable para que resulte fácil la detección de movimiento.

No se han encontrado datos acerca de proyectos implementados que puedan realizar el seguimiento de un objeto donde se presentan fondo e iluminación variable, y que al mismo tiempo se utilice tecnología de procesamiento paralelo, CUDA, por lo que el sistema presente resulta innovador.

[4] Cfr. Sanchez 2010a
[5] Cfr. Espinoza y otros 2012: 1-3
[6] Cfr. Rodriguez 2006: 2-10

La mayoría de proyectos realizados en el mundo utilizan cámaras estáticas y realizan el seguimiento de objetos siempre con el fondo fijo, si bien algunos proyectos utilizan tecnología CUDA[7] el hecho de que el sistema presentado en este libro pueda trabajar bajo las condiciones de un fondo e iluminación variables representa un gran valor agregado.

La compañía SONY lanzó en el año 2006 el PS3 con el sistema de los mandos *move,* para obtener las coordenadas de la posición de los mandos, estos cuentan con iluminación propia, lo que hace que esta no dependa del entorno donde se utilice, siendo de esta manera más fácil de segmentar, además cuenta con un procesador central de siete núcleos[8] lo que resulta en una gran ventaja al momento de realizar operaciones de gran coste computacional. Sin embargo, este procesador y la tarjeta de video resultan con un costo muy elevado si sólo se desea realizar un sistema de seguimiento de objetos.

Ventajas y Desventajas

Una de las principales desventajas del proyecto es que este está sujeto al robot autónomo en el cual es implementado, ya que una falla en el robot implicaría una falla en el sistema de visión. Asimismo el tiempo de autonomía del robot incide directamente en el desempeño del sistema.

[7] Cfr. Díaz, T y otros 2013
[8] Cfr. Wikipedia 2013

Otra desventaja del sistema es que la región a monitorear depende del alcance de la cámara.

Debido a la alta carga computacional del sistema, no es conveniente trabajar con video en alta definición, por lo que se tiene que trabajar con video estándar del tipo “Broadcast” NTSC o PAL[9].

Debido a que la tecnología utilizada para el procesamiento paralelo pertenece a una sola compañía, el desarrollo se ve limitado a la utilización de una sola marca.

Por otro lado, entre las ventajas del sistema se tiene que este es económico, ya que para monitorear un lugar no es necesario utilizar una cámara de alto coste.

Otra ventaja que presenta el sistema es que debido a que este está implementado en un robot autónomo, presenta movimiento, lo que significa que no siempre estará monitoreando un mismo punto.

El sistema puede ser fácilmente instalado en diferentes tipos de robots autónomos sin afectar su desempeño debido a su reducido tamaño y bajo consumo de energía.

El software puede ser instalado en un sistema operativo como Windows XP o superior, en cualquier desktop o laptop con tarjeta de video NVIDIA que cuente con tecnología CUDA.

[9] Cfr. NationMaster 2010

El software desarrollado es de fácil utilización e intuitivo.

Aplicaciones

Dado que la meta es implementar este sistema en un robot autónomo, este resultará útil para poder realizar monitoreo de actividades varias, por ejemplo:

- Vigilancia en espacios abiertos, por ejemplo conciertos, partidos de fútbol, y otros lugares de gran aforo. Con este sistema implementado en cámaras de seguridad se puede hacer seguimiento a algún individuo u objeto de tal manera que siempre se tenga en la mira.
- Persecución de vehículos a alta velocidad. El sistema implementado en un vehículo autónomo no tripulado (e.g. Neurocopter) podría ser capaz de perseguir o tener siempre en la mira a un vehículo, esta aplicación puede resultar útil para instituciones como la Policía Nacional o los centros de seguridad de las municipalidades.
- Vigilancia de playas. Se realizaría el monitoreo constante de las playas mediante un vehículo autónomo no tripulado asegurando así el bienestar de los bañistas, lo cual resultaría muy beneficioso para la unidad de salvataje de la policía.
- En un conflicto aéreo o naval el sistema puede ser utilizado por las fuerzas armadas para el seguimiento de naves para la calibración de

miras lo cual proporciona un sistema inteligente que ajusta la mira del cañón. Al tener un fondo casi uniforme como es el cielo se facilita la segmentación de los objetos (naves) lo que permite una mejor precisión del sistema.

- Sistema de control para cámaras aéreas. En la mayoría de ocasiones para la filmación de escenas aéreas se utilizan vehículos aéreos tripulados o vehículos a control remoto los cuales pueden ser remplazados por vehículos autónomos no tripulados que automáticamente sigan el objeto que se desea filmar.
- Control de aves en aeropuertos. En muchas oportunidades las aves son un inconveniente para el despegue y aterrizaje de aviones por lo cual se propone un sistema que fácilmente detecta la presencia de bandadas de aves para ser ahuyentadas posteriormente.

Entre los usuarios potenciales de un sistema de seguimiento inteligente se considera a las Fuerzas Armadas, para la calibración de miras en los buques, así como un sistema de visión artificial para ser implementado en vehículos aéreos no tripulados (UAV) de la Fuerza aérea o Marina de Guerra; industria del entretenimiento como canales de televisión y promotores de eventos, juguetes y videojuegos, policía, bomberos, equipos de salvamento, robótica, agencias de seguridad como Liderman, Prosegur o G4S, etc.

Objetivos

Objetivo General

- Implementación de un sistema de visión en un robot autónomo, que sea económico, eficiente y se apoye en el uso de tecnología de punta para el procesamiento digital de imágenes.

Objetivos Específicos

- Realizar el seguimiento de una marca específica ubicada en un objeto
- Control del movimiento de la cámara para realizar el seguimiento, de tal manera que se tenga un movimiento suave y agradable para el usuario
- Realizar una interfaz amigable para el usuario
- Predicción del movimiento del objeto y algoritmos de respaldo que permitan reducir la probabilidad de error en el sistema
- Desarrollar el sistema a un bajo costo
- Desarrollar un software eficiente utilizando tecnología de procesamiento paralelo (MATLAB-CUDA)
- Contribuir con la seguridad social mediante la implementación del sistema
- Desarrollar tecnología con calidad de exportación

- Realizar segmentación de imágenes con fondo e iluminación variables
- Ubicar el objeto a seguir en el centro de la pantalla y mantenerlo centrado durante su seguimiento
- Dar una alternativa al control de aves en aeropuertos
- Realizar mejoras en las imágenes a procesar mediante el uso de filtros, correcciones de brillo, contraste, etc
- Obtener un error menor al 5% en la estimación de distancia del robot al objeto a seguir

CAPÍTULO 2. MARCO TEÓRICO

El presente capítulo expone los fundamentos del procesamiento digital de imágenes, así como también las técnicas empleadas para la segmentación de imágenes y el seguimiento de objetos.

Fundamentos del color

Los seres humanos son capaces de diferenciar entre miles de tonalidades de colores y cerca de dos docenas de niveles de grises.[10] Asimismo el uso del color en el procesamiento digital de imágenes es una herramienta poderosa que describe o identifica de manera simplificada cualquier objeto de una escena. En este subcapítulo se estudiarán los fundamentos del color como algunos elementos que lo caracterizan.

La luz visible se compone de una delgada banda de frecuencias del espectro electromagnético[11]. En la **Figura 2.1** se muestra las longitudes de onda del espectro visible por el hombre.

[10] Cfr. González y Woods 1996: 282

[11] Espectro electromagnético: es el conjunto formado por todas las formas de onda electromagnéticas existentes.

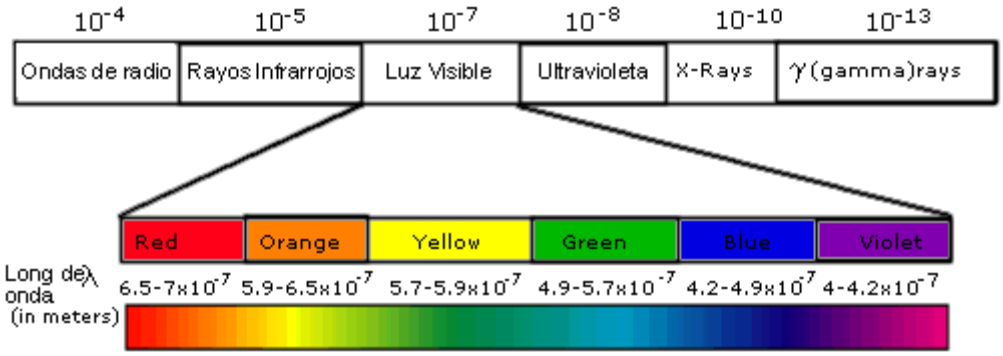


Figura 2.1. Longitudes de onda visible (Adanero 2002)

Para poder percibir el color (o los colores) es necesario tener tres elementos como la luz (ondas electromagnéticas), un objeto y alguien que observe el fenómeno (observador). Todo cuerpo u objeto iluminado absorbe una parte de las ondas electromagnéticas y refleja las restantes. Las ondas reflejadas son captadas por el ojo e interpretadas como colores por el cerebro, la longitud de onda captada corresponde a un color específico. El ojo humano sólo percibe las longitudes de onda cuando la iluminación es abundante, en otras palabras si hay poca luz un objeto se ve menos colorido mientras que con buena luz ocurre todo lo contrario.

Existen ciertas características que distinguen un color de otro, las cuales son las siguientes[12]:

Brillo: Noción cromática[13] de intensidad.

Tono: Color dominante que percibe un observador.

Saturación: El grado de saturación es inversamente proporcional a la cantidad de blanco que se añade al color.

[12] Cfr. Gonzales y Woods 1996: 284-285

[13] Se refiere la intensidad de pintado o matizado de un objeto.

Imagen digital

Una imagen digital es una celda compuesta por elementos llamados píxeles. Cada pixel es un espacio en la memoria de la computadora donde se almacena un número y este número representa la definición de color y brillo de una parte de la imagen. Asimismo una imagen digital se representa como una tabla de valores o una matriz, es decir un conjunto de valores numéricos ordenados en filas y columnas, en la que cada elemento corresponde al valor de intensidad de cada punto de muestreo de la imagen[14].

Un píxel se designa con el nombre de la matriz y los subíndices que indican la fila y la columna al cual pertenece, así : A(i,j) = 127 indica que el píxel que ocupa la fila i y la columna j en la imagen A tiene un valor de intensidad igual a 127.

Modelos de colores para el tratamiento digital de imágenes

Para un mejor estudio del color los modelos permiten un entendimiento más detallado del color y sus características, además permiten identificar y manipular los colores. Los modelos que se tratan en esta sección para el procesamiento digital de imágenes son: RGB y YCbCr.

[14] Cfr. UFES 2013

Modelo RGB

RGB es un modelo basado en tres componentes primarios del espectro de rojo, verde y azul[15]. Este modelo se basa es una sistema cartesiano de coordenadas que forman un cubo como o muestra la **Figura 2.2**

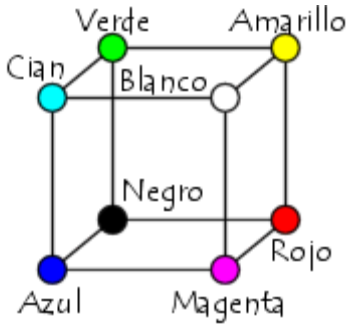


Figura 2.2. Cubo RGB (Proyecta color 2009)

Este es el modelo de definición de color en pantalla usado para trabajos digitales. En cada pantalla existen pequeños puntos llamados píxeles, los cuales son un conjunto de tres celdas: rojo, verde y azul, donde cada uno brilla con determinada intensidad.

El color blanco se forma con los tres colores a su máximo nivel, el color negro cuando los tres componentes están a su mínimo nivel. Existe un color neutro o gris que se obtiene cuando los tres componentes son idénticos. Así también mediante combinaciones se obtienen tres colores intermedios como el amarillo, el magenta y el cian como se muestra a continuación en la **Figura 2.3**.

[15] Cfr. Proyecta color 2009

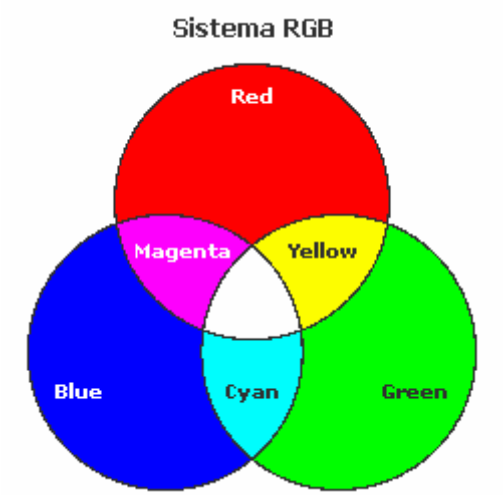


Figura 2.3. Sistema RGB (Moreno 2004)

Espacio Color YCbCr

Este espacio de color es normalmente usado en formatos de video o fotografía digital.

"Y" representa la luminancia (luma) y tiene valores en el rango de 0 a 255.

"Cb" y "Cr" representan la crominancia[16], de los componentes azul y rojo respectivamente.

Este tipo de espacio de color al igual que YIQ, entre otros, separa la luminancia (luz) de la crominancia (color) para luego poder guardar la luminancia con una mejor resolución o transmitirla a un mayor ancho de

[16] Crominancia es la combinación de saturación y tono

banda, mientras que los componentes de crominancia pueden ser submuestreados, comprimidos o tratados de manera separada[17].

El espacio de color YCbCr se deriva del RGB corregido por el factor Gamma, por lo que la fórmula que se emplea para transformar un espacio de color en el otro es la siguiente:

$$Y = 0.257R' + 0.504G' + 0.098B' + 16 \quad \textbf{(2.1)}$$

$$Cb = -0.148R' - 0.291G' + 0.439B' + 128 \quad \textbf{(2.2)}$$

$$Cr = 0.439R' - 0.368G' - 0.071B' + 128 \quad \textbf{(2.3)}$$

$$R' = 1.164(Y - 16) + 1.596(Cr - 128) \quad \textbf{(2.4)}$$

$$G' = 1.164(Y - 16) - 0.813(Cr - 128) - 0.392(Cb - 128) \quad \textbf{(2.5)}$$

$$B' = 1.164(Y - 16) + 2.017(Cb - 128) \quad \textbf{(2.6)}$$

Donde “R’”, “G’” y “B’” son los corregidos con el factor Gamma. Usualmente para un video NTSC el valor del factor Gamma es de 2.2, y de manera que se pueda generar el espacio RGB de 24 bits, se aplica la siguiente fórmula[18]:

$Para\ R', G' y\ B' < 21$

$$R = \left({}^{\frac{R'}{255}}\Big/_{4.5} \right) 255 \quad \textbf{(2.7)}$$

[17] Cfr. Wikipedia 2010
[18] Cfr. Intersil 2010

$$G = \left({}^{\frac{G\prime}{255}}\Big/_{4.5} \right) 255 \quad \textbf{(2.8)}$$

$$B = \left({}^{\frac{B\prime}{255}}\Big/_{4.5} \right) 255 \quad \textbf{(2.9)}$$

$Para\ R', G' y\ B' \geq 21$

$$R = \left({}^{\frac{R\prime}{255} + 0.099}\Big/_{1.099} \right)^{2.2} x\ 255 \quad \textbf{(2.10)}$$

$$G = \left({}^{\frac{G\prime}{255} + 0.099}\Big/_{1.099} \right)^{2.2} x\ 255 \quad \textbf{(2.11)}$$

$$B = \left({}^{\frac{B\prime}{255} + 0.099}\Big/_{1.099} \right)^{2.2} x\ 255 \quad \textbf{(2.12)}$$

Se observó que la cámara capturaba el video utilizando el formato UYVY, el cual es YCbCr 4:2:2 basado en NTSC, es decir, se toma una muestra de Y en todos los pixeles, mientras que se toman muestras de "Cb" y "Cr" cada 2 pixeles[19]. De esta manera se disminuye el ancho de banda de la transmisión.

[19] Cfr. FourCC 2012

Procesamiento digital de imágenes

El ser humano a diferencia de muchos animales, depende de sus ojos para poder recibir información del entorno que lo rodea, es por esa razón que se han creado a través del tiempo ciertos dispositivos que facilitan la obtención y observación de imágenes de tamaño muy pequeño, como el microscopio y de tamaño grande, como el telescopio.

El procesamiento digital de imágenes es la técnica mediante la cual se puede extraer información del mundo real, la cual ha sido plasmada en una imagen o en una secuencia de ellas (video). Este procesamiento ayuda a representar de manera eficiente información pictórica mediante técnicas de codificación y compresión de imágenes.

El procesamiento de los datos que se obtienen de una imagen se detallará a continuación.

Adquisición

La manera más común de adquirir una imagen es con una cámara de video estándar, pero existen en el mercado muchas alternativas de donde escoger, dependiendo de la aplicación a realizar, algunas presentando claras ventajas con respecto a otras.

Hoy en día, se utilizan también cámaras digitales, las cuales permiten una alta compresión de las imágenes pero brindan una resolución limitada, dependiendo de la aplicación y el precio. Debido a la compresión que presentan las imágenes puede existir pérdida de información si la cámara no es la adecuada, lo que sería un gran problema a resolver.[20]

La adquisición para el sistema se realizó mediante una cámara inalámbrica de 2.4 GHz, 8VDC, con sensor CMOS y formato NTSC.

Preprocesamiento

El preprocesamiento consiste en filtrar las imágenes de manera que estas no presenten ruido, es decir que no presenten elementos innecesarios para su futuro procesamiento. Este “ruido” puede ser causado por efectos de la luz en el lente de la cámara, o por el movimiento de la mano al tomar una foto.

Los algoritmos utilizados para el pre-procesamiento de las imágenes fueron:

Filtros:

Consiste en aplicar una transformación a una imagen por medio de un operador.

[20] Cfr. Russ 1999: 7-42

Filtro de Mediana (“median filter”) también es un filtro espacial, pero reemplaza el valor central de la ventana con la mediana de los valores de la ventana[21].

El filtro de mediana remueve el ruido impulsivo (sal y pimienta) y es bien conocido por preservar los bordes de las imágenes, mientras que el filtro promediador[22] no es bueno removiendo el ruido impulsivo debido a que promedia los valores de la ventana.

Valores sin filtrar		
6	2	0
3	97	4
19	3	10

En orden: 0, 2, 3, 3, **4**, 6, 10, 19, 97

Filtro de mediana		
*	*	*
*	4	*
*	*	*

Figura 2.4 Funcionamiento del filtro de mediana

En la **Figura 2.5** se muestra el efecto del filtro de mediana sobre una imagen de muestra.

[21] Cfr. Jain 1989: 430-450

[22] Filtro espacial que reemplaza el valor central de la ventana con el promedio de los valores

Imagen sin filtrar

Imagen filtrada

Figura 2.5 Efecto de filtro de mediana (UFES 2013)

Corrección Gamma

Para poder mejorar las imágenes dependiendo de las condiciones ambientales se utilizó la técnica de corrección Gamma.

La operación de corrección Gamma realiza un ajuste no lineal del brillo o luminancia de una imagen. El brillo para los píxeles más oscuros es incrementado considerablemente mientras que los píxeles de mayor brillo el incremento es menor. Como resultado más detalles son visibles en una imagen[23]. Se logra utilizando la siguiente ecuación:

$$s = cr^{\gamma} \qquad \textbf{(2.13)}$$

Donde, “c” y “γ” son constantes; “r” es la imagen entrante a la función y “s” la imagen resultado.

El efecto de la corrección gamma se muestra en la **Figura 2.6**

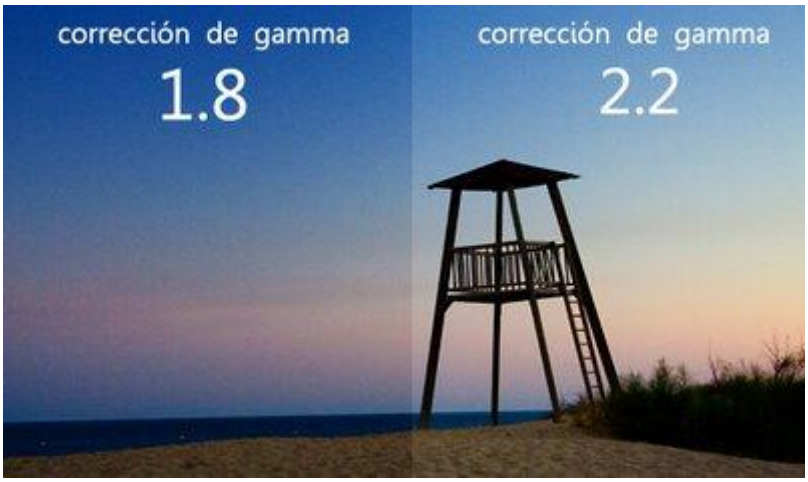


Figura 2.6 Efecto de corrección Gamma (UFES 2013)

La fórmula para lograr la corrección del factor Gamma en una imagen rasterizada es la siguiente:

[23] Cfr. Starovoitov, Samal y Briluik 2003

$$I_{CPU} = 255\left(\frac{I_m}{255}\right)^{\gamma} \quad \textbf{(2.14)}$$

Operaciones Morfológicas

La idea básica de las operaciones morfológicas es comparar los objetos que se desean analizar con otro objeto de forma conocida, llamado "elemento estructurante". Las operaciones morfológicas realizan operaciones sobre conjuntos y son un método sistemático para estudiar las relaciones entre puntos de un conjunto. Estas operaciones dan por resultado un nuevo conjunto que caracteriza la estructura de la imagen. Para realizar una de estas operaciones es necesario elegir un elemento estructurante de geometría conocida y deslizarlo a través de la imagen. Para el programa realizado, se vio conveniente utilizar las operaciones morfológicas detalladas a continuación.

- **Erosión:** Dado un conjunto "A" y un elemento estructurante "W", la erosión de "A" por "W" es el conjunto de todos los elementos "x" para los cuales "W" trasladado por "x" está contenido en "A". Esta operación ayuda a determinar si un elemento "W" está completamente incluido en el conjunto "A", si esto se cumple, entonces el resultado de la erosión es un conjunto vacío.

La erosión hará que los objetos menores al elemento estructurante (“W”) no aparezcan en la imagen resultante, por lo que se supone así una degradación de la imagen[24]. Se define como:

$$A \ominus W = \{x / W_x \subseteq A\} \qquad \textbf{(2.15)}$$

A continuación de muestra en la **Figura 2.7** un ejemplo simple de la técnica de erosión.

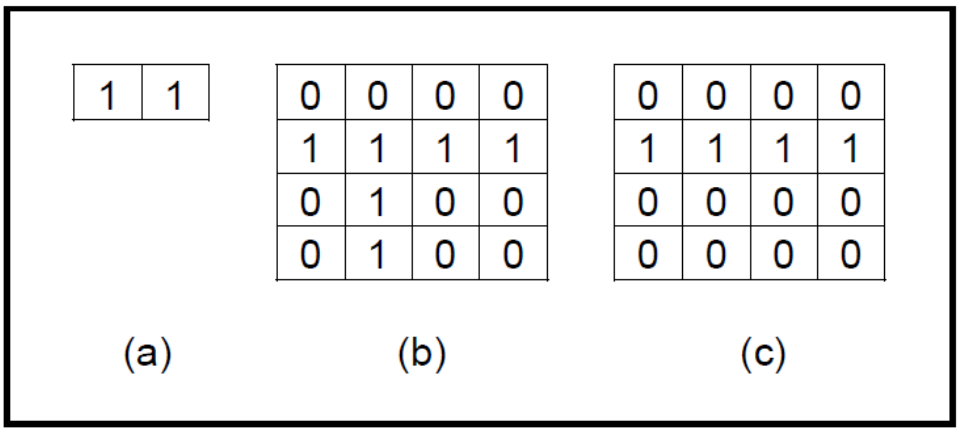


Figura 2.7 (a) Elemento estructural, (b) Imagen estructural y (c) Resultado de la erosión

- **Dilatación:** La dilatación es lo opuesto a la erosión, el resultado de este es un conjunto de elementos tal que al menos un elemento del conjunto estructurante “W” está contenido en el conjunto “A”, cuando “W” se desplaza sobre el conjunto “A”. La dilatación entonces

[24] Cfr. Platero 2012: 173

representa un crecimiento progresivo del conjunto “W”[25]. Se define como:

$$A \oplus W = \{x / W_x \cap A \neq \emptyset\} \qquad \textbf{(2.16)}$$

En general, la dilatación aumenta el tamaño de un objeto, la cantidad y la forma en que aumenta el tamaño depende de la elección del elemento estructurante. Un ejemplo simple de la técnica de dilatación se presenta en la **Figura 2.8**

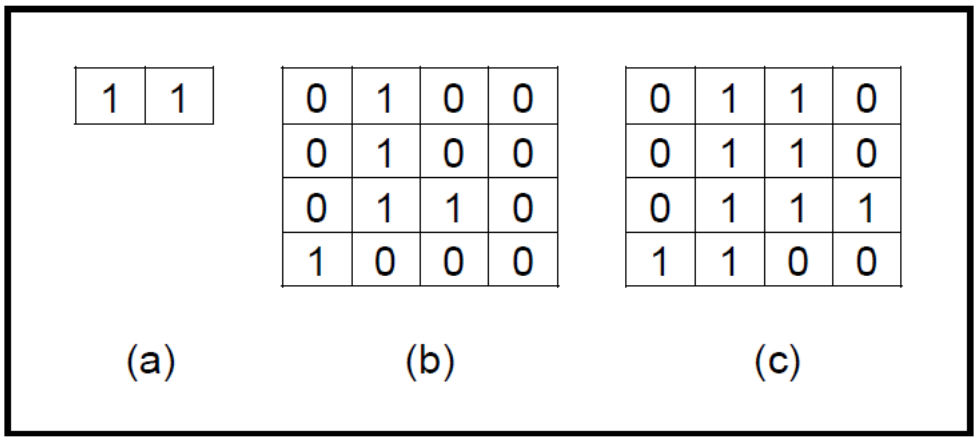


Figura 2.8 (a) Elemento estructural, (b) Imagen estructural y (c) Resultado de la dilatación

- **Apertura:** Generalmente suaviza el contorno de una imagen, rompe istmos estrechos y elimina protuberancias delgadas. La apertura de un conjunto “A” por un elemento estructurante “W”, representada por “A∘W”, se define como:

[25] Cfr. Platero 2012: 174

$$A \circ W = (A \ominus W) \oplus W \quad \textbf{(2.17)}$$

Es decir que la apertura de “A” por “W” es la erosión de “A” por “W”, seguida por una dilatación del resultado por “W”.[26]

Este algoritmo permite eliminar bordes que no están bien definidos así como diferenciar un objeto de otro. Es decir, se realiza un filtrado morfológico para poder obtener objetos bien definidos para poder ser segmentados.

- **Cierre:** Tiende a suavizar secciones de contornos pero, en oposición a la apertura, generalmente fusiona separaciones estrechas y entrantes delgados y profundos, elimina pequeños huecos y rellena agujeros de contorno. El cierre del conjunto “A” por el elemento de estructura “W”, representado por “A·W”, se define como:

$$A \cdot W = (A \oplus W) \ominus W \quad \textbf{(2.18)}$$

Es decir que el cierre de “A” por “W” es la dilatación de “A” por “W”, seguida por la erosión del resultado por “W”[27].

Sin embargo, este algoritmo no es de utilidad si se desean segmentar varios objetos ya que estos pueden encontrarse uno muy cerca del otro y se pueden combinar perdiendo así un objeto.

[26] Cfr. Gonzales y Woods 2002:554-556
[27] Cfr. Gonzales y Woods 2002: 554-556

- **Relleno de regiones:** Es un algoritmo morfológico que se basa en dilataciones, complementos e intersecciones. Dado "A", que representa a un conjunto que contiene un subconjunto cuyos elementos son puntos 8-conexos del contorno de una región de tal forma que el proceso a continuación se encarga de rellenar el subconjunto.

$$X_k = (X_{k-1} \oplus W) \cap A^c \qquad k = 1,2,3 \ldots \qquad \textbf{(2.19)}$$

Donde "W" es un elemento de estructura y, finalmente el conjunto unión "X_k" y "A" contiene al conjunto rellenado y a su contorno[28].

Como su mismo nombre lo dice, el objetivo de este algoritmo es el de rellenar aquellos espacios internos dentro de los objetos, esto permite compensar muchos espacios vacíos que deja la función de apertura en las imágenes.

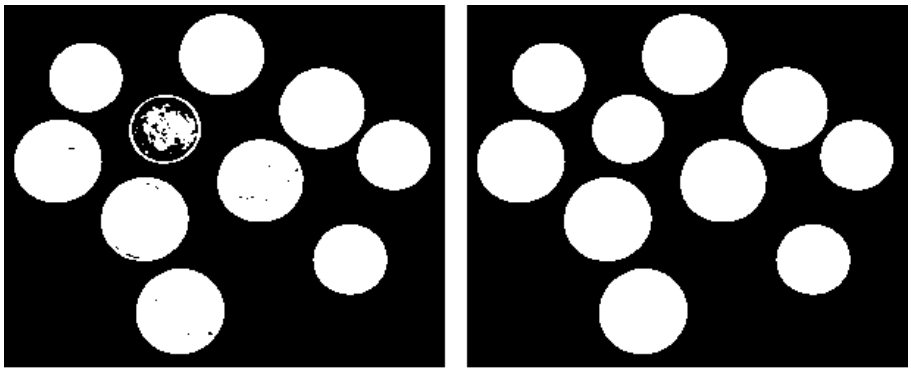


Imagen original Relleno de regiones

Figura 2.9 Ejemplo de relleno de regiones (Matlab 2010)

[28] Cfr. Gonzales y Woods 2002: 535

- **Eliminación de borde de la imagen:** Utilizando 4-conectividad alrededor del borde de la imagen se detectan los objetos y luego son eliminados. Esto se utiliza para evitar tomar como objetos dentro de la imagen, objetos que han salido del ángulo de visión de la cámara. Es decir, elimina los objetos que se encuentren al borde de la imagen[29].

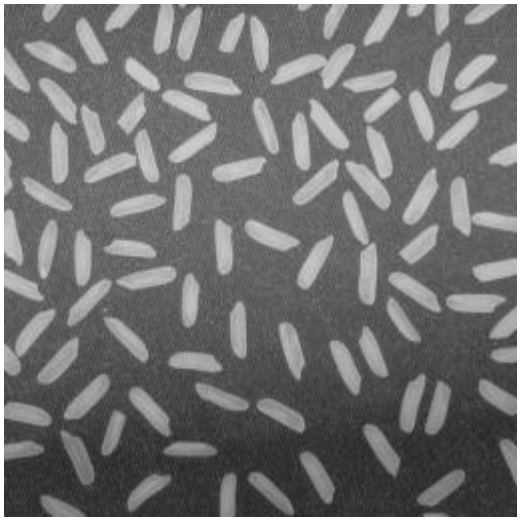
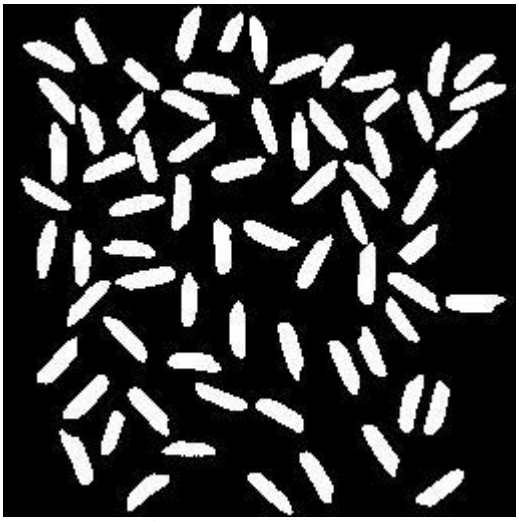


Imagen original



Eliminación de borde de la imagen

Figura 2.10 Ejemplo de relleno de eliminación de borde de la imagen (Matlab 2010)

Segmentación

La segmentación se refiere a extraer sólo los pixeles que corresponden al objeto o región de interés. Sirve para poder realizar detección de bordes, líneas y puntos dentro de una imagen de manera que se puedan separar y analizar.

[29] Cfr. Gonzales y Woods 2002: 534

Este proceso evalúa si cada pixel de la imagen pertenece o no al objeto de interés. Existen diferentes técnicas para la segmentación de una imagen, pero una de las más simples consiste en evaluar una imagen binaria[30], y analizar la conectividad de sus pixeles, de esta manera se puede obtener el número de objetos que se encuentran en la imagen, al igual que su posición, para luego separar sólo los objetos de interés.[31]

Segmentación por color:

- **Distancia Euclidiana**

  Fórmula No Lineal: La distancia euclidiana entre "$p(x,y)$" y "$q(s,t)$" está definida por:

$$D_e = [(x-s)^2 + (y-t)^2]^{1/2} \quad \textbf{(2.20)}$$

  Este algoritmo es comúnmente utilizado para hallar la distancia entre dos puntos dentro de una imagen. Sin embargo, también se puede utilizar para obtener el grado de similitud o "distancia" entre la crominancia de un par de píxeles. De esta manera se consigue una umbralización por colores. Sin embargo, la utilización de una fórmula no lineal como esta implica gran carga computacional, por lo cual se

[30] Imagen Binaria es aquella formada sólo por dos posibles valores para cada uno de los pixeles, es decir sólo tiene dos colores: blanco y negro, en donde el blanco son los pixeles con el valor de 255 y el negro son los pixeles con el valor de 0.
[31] Cfr. Jain 1989: 407-411

intentó utilizar maneras alternativas como se muestran a continuación.

La distancia de Mahalanobis, brinda una manera directa de realizar la segmentación de color, es decir cuando se quiere separar objetos con un mismo color dentro de una imagen, la distancia de Mahalanobis brinda información acerca de la similitud entre los pixeles que conforman una imagen. Esta dada por la siguiente fórmula:

$$D = \sqrt{(x - y)^T C^{-1} (x - y)} \qquad \textbf{(2.21)}$$

Donde "C" es la matriz de covarianzas [32] y "x" e "y" son las componentes RGB de la imagen y la muestra de color a segmentar[33].

Cuando la matriz de covarianzas es la matriz Identidad, la distancia de Mahalanobis se convierte en distancia Euclidiana, este principio se utilizó al igual que la distancia de Mahalanobis para realizar la segmentación a color ya que resulta como antes se mencionó, una manera directa de segmentar color, sin embargo debido a pruebas realizadas se verificó que el coste computacional de la distancia de Mahalanobis es mayor que el de la distancia Euclidiana, por lo que

[32] La covarianza indica el grado de correlación entre 2 o más variables.
[33] Cfr. Gonzales y Woods 2004: 237-240

se decidió seguir trabajando sólo con este último, obteniendo así un mejor performance de software.

Umbralización

La umbralización es una técnica de segmentación ampliamente utilizada en la industria. Se hace una transformación no-lineal de la imagen de entrada, obteniendo una imagen de salida donde cada pixel puede tomar uno de dos valores, 0 ó 1, negro o blanco[34]. Para obtener esto, se toma un valor de umbral "T" de manera que:

$$S[x,y] = 1, I_m[x,y] \geq T \quad \textbf{(2.22)}$$

$$S[x,y] = 0, I_m[x,y] < T \quad \textbf{(2.23)}$$

Es una forma sencilla de definir un umbral, para que separe los objetos del fondo, así el objeto de interés puede tomar el valor de 1 y lo demás 0 o viceversa.

Descripción de una Imagen

Una vez que se tiene la imagen segmentada, la descripción consiste en extraer información de esta imagen segmentada que la describa, por ejemplo, ubicación del centroide, área, longitud, perímetro, etc. Para el

[34] Cfr. Sucar y Gómez 2003:16-17

sistema sólo interesa realizar el cálculo necesario para la ubicación del centroide.

Centroide de una imagen

El Centroide de una imagen calcula el centro de masa de los objetos que se encuentran en una imagen. En el caso del sistema elaborado se calcula en centro de masa del objeto a seguir para poder localizarlo siempre dentro de la imagen. Se calcula como:

$$\mathrm{Cx} = \frac{\sum \text{coordenadas x de los pixeles}}{\text{número de pixeles del objeto}} \quad \textbf{(2.24)}$$

$$\mathrm{Cy} = \frac{\sum \text{coordenadas y de los pixeles}}{\text{número de pixeles del objeto}} \quad \textbf{(2.25)}$$

Donde “Cx” es el centro de masa x y “Cy” es el centro de masa y, por lo que el centroide del objeto estará en el punto “(Cx,Cy)”[35].

Unidad de procesamiento gráfico (GPU)

Un GPU es una unidad de procesamiento de gráficos que sirve como coprocesador para el CPU anfitrión, tiene su propia unidad de memoria y ejecuta varios hilos (“threads”) en paralelo. Este dispositivo de cómputo

[35] Cfr. Pertusa 2003: 108

tiene varios núcleos, pero una arquitectura más simple que un CPU estándar.

CUDA es la arquitectura de procesamiento paralelo de NVIDIA que permite un incremento en la performance computacional explotando así el poder del GPU[36].

Debido a la capacidad de operar grandes cantidades de datos en paralelo, la utilización de un GPU resulta más efectiva que la utilización de un CPU normal.

Arquitectura CUDA

Un kernel es una función llamada desde el anfitrión (CPU) que se procesa en el GPU, es ejecutado uno por vez y muchos “threads” (hilos) ejecutan cada kernel.

Todos los kernel ejecutan el mismo código, lo que ayuda a la rapidez de procesamiento. Las diferencias entre los “threads” de CUDA y de un CPU son:

- Los hilos de CUDA son extremadamente ligeros ya que tienen un costo de creación muy ligero.

[36] Cfr. NVIDIA Corporation 2010a

- CUDA utiliza miles de threads para mayor eficiencia, mientras que los CPU multinúcleo pueden usar solo unos pocos

Los threads son agrupados en bloques y estos a su vez se agrupan en “grids” como se muestra en la **Figura 2.11**

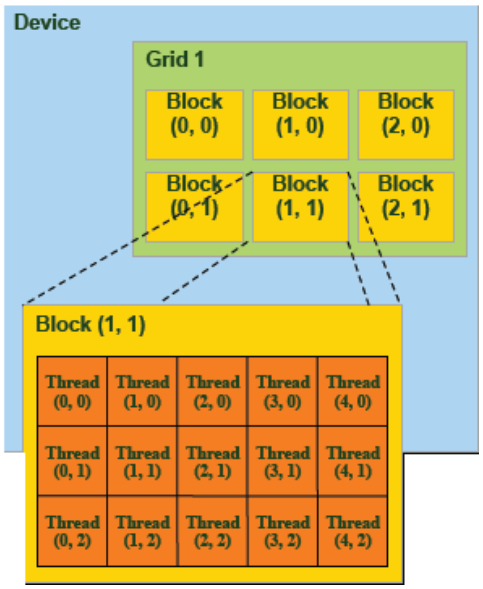


Figura 2.11 Agrupación CUDA (NVIDIA Corporation 2007)

Un kernel se ejecuta como un “grid” de bloques de hilos, para comprender mejor estos agrupamientos, se muestra la **Figura 2.12**

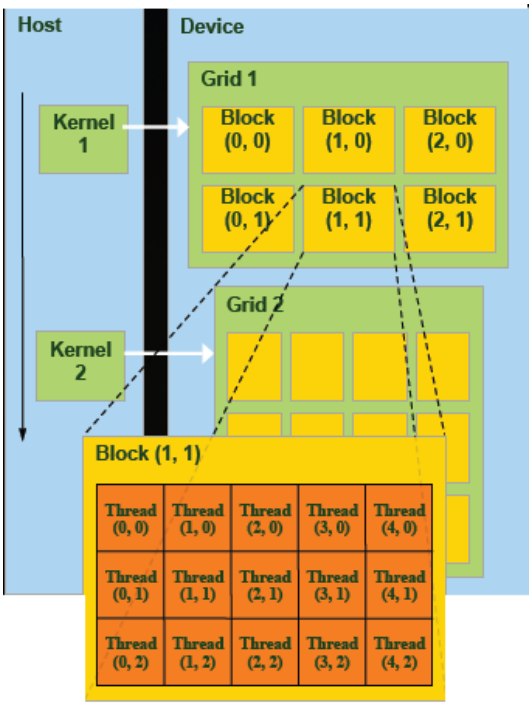


Figura 2.12 Arquitectura CUDA (NVIDIA Corporation 2007)

En cuanto al acceso a memoria de la función (kernel) se cuenta con una Memoria Global en la cual residen las entradas y salidas del kernel. Esta es la memoria DRAM especificada en las características de la tarjeta de video. La Memoria compartida es aquella que se comparte entre los “threads” dentro de un bloque, es mucho más pequeña y tan rápida como los registros. A continuación, en la **Figura 2.13** se muestra un esquema de acceso a memoria:

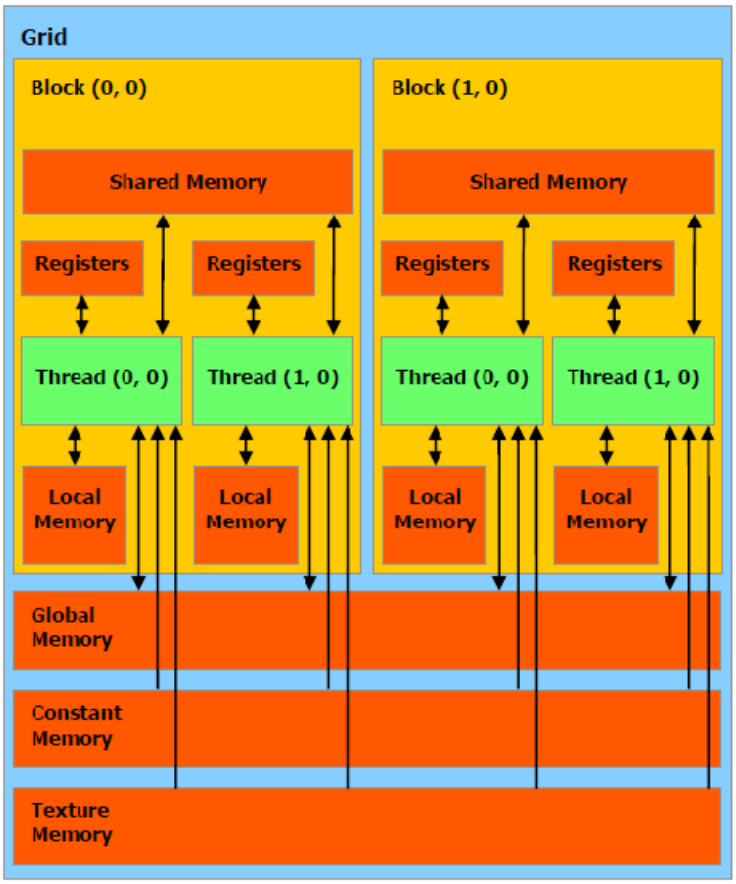


Figura 2.13 Acceso a Memoria (NVIDIA Corporation 2007)

El GPU no puede acceder directamente a la memoria principal, y el CPU no puede acceder directamente a la memoria del GPU[37].

Enlace de datos

El enlace de datos es un proceso muy importante en la ejecución de un proyecto. Al diseñar un sistema se tiene que tener en cuenta que la comunicación tiene que ser eficiente y eficaz. Es por ello que entre los parámetros que se evaluaron al seleccionar un sistema de comunicación

[37] Cfr. NVIDIA Corporation 2007: 10-11

se encuentran la velocidad de transmisión, inmunidad al ruido, costo, complejidad de la implementación y versatilidad de la tecnología. A continuación se describen brevemente los diferentes conceptos utilizados en los enlaces de datos del proyecto.

Comunicación serial

La comunicación serial es un protocolo para la comunicación entre dispositivos que se incluye de manera estándar casi en cualquier computadora mediante el puerto RS-232 (también conocido como COM). Este puerto serial envía y recibe bytes de información un bit a la vez. A pesar de que esto es más lento que la comunicación en paralelo, que permite transmitir un byte a la vez, este método es más sencillo y económico de implementar.

Es muy común la utilización de la comunicación serial para transmitir datos en formato ASCII. Para ello se utilizan tres líneas de conexión: 1. Tierra o referencia, 2. Transmisión y 3. Recepción. Debido a que la transmisión es asíncrona, es posible enviar datos por una línea mientras se reciben por otro, es decir es una comunicación “Full Dupplex”. Existen otras líneas para realizar “handshaking” o intercambio de pulsos de sincronización, pero para la presente aplicación no son requeridas. Las características que caben resaltar de la comunicación serial son la velocidad de transmisión, los bits

de datos, los bits de parada y la paridad. Para que dos puertos se puedan comunicar es necesario que estas características sean iguales en ambos.[38]

- Tasa de baudios ("baud rate"). Es el número de unidades de señal o símbolos por segundo. Mientras que la velocidad de transmisión se refiere al número de bits que se transfieren por segundo, un símbolo o puede contener varios bits. Se mide en baudios.

- Bits de datos. Indica la cantidad de bits en la transmisión. El número de bits que se envía depende del tipo de información que se transfiere. En el caso del ASCII estándar se tiene un rango de 0 a 127, es decir, utiliza 7 bits; para el ASCII extendido es de 0 a 255, lo cual utiliza 8 bits por paquete. Un paquete se refiere a una transferencia de byte, incluyendo los bits de inicio y parada, bits de datos y paridad.

- Bit de inicio. Cuando el receptor detecta el bit de inicio sabe que la transmisión ha comenzado y es a partir de entonces que debe leer la transmisión en función de la velocidad determinada.

- Bits de parada. Se utiliza para identificar el fin de la comunicación de un solo paquete. Los valores típicos son 1, 1.5 o 2 bits. Debido a la manera como se transfiere la información a través de las líneas de comunicación y que cada dispositivo tiene su propio reloj interno, es

[38] Cfr. National Instruments 2006

posible que los dispositivos no estén sincronizados. Por lo tanto, los bits de parada no sólo indican el fin de la transmisión, sino además dan un margen de tolerancia para la esa diferencia de los relojes. Sin embargo a más bits de parada la transmisión será más lenta pero con mayor margen de tolerancia.

- Paridad. Es una forma sencilla de verificar si hay errores en la transmisión serial. Existen cuatro tipos de paridad: par, impar, marcada y espaciada. Además existe la opción de no utilizar paridad. [39]

Las funciones de los pines en RS-232 están definidas en el estándar ANSI/EIA-232 y son como sigue:

- Datos: TXD (pin 3), RXD (pin 2)
- Handshake: RTS (pin 7), CTS (pin 8), DSR (pin 6), DCD (pin 1), DTR (pin 4)
- Tierra: GND (pin 5)
- Otros: RI (pin9)

[39] Cfr. Morales 2003: 19-28

Banda ICM 2.4 GHz

El estándar IEEE 802.11 y las posteriores modificaciones 802.11b, 802.11g y 802.11n definen las comunicaciones en el rango de frecuencias entre 2.4GHz y 2.4835 GHz. Este rango de frecuencia es uno de los tres rangos conocidos como las bandas industrial, científica y médica (ICM)[40]. Los rangos de frecuencia son los siguientes:

- 902 – 928 MHz (26MHz de ancho)
- 2.4 – 2.5 GHz (100MHz de ancho)
- 5.725 – 5.878 GHz (150 MHz de ancho)

Las bandas ICM están definidas por el Sector de Normalización de las Telecomunicaciones de la Unión Internacional de Telecomunicaciones (UIT-T) en los números 5.138 y 5.150 del Reglamento de Radiocomunicaciones.[41] La banda de 900 MHz es conocida como la banda industrial, la banda de 2.4 GHz es conocida como la banda científica y la de 5.8 GHz como la banda médica.

Estás tres bandas no requieren licencia y no hay restricción en cuanto a los equipos en los que pueden ser utilizadas.

[40] También llamada Banda ISM por sus siglas en inglés (Industrial, Scientific and Medical Bands)
[41] Cfr. UIT 2007

La banda ICM 2.4 GHz es la banda más utilizada para redes de comunicación inalámbrica. La banda ICM 2.4 GHz tiene un ancho de 100 MHz y va desde 2.4 GHz hasta 2.5 GHz. Además de ser utilizada por los equipos WLAN 802.11, la banda ICM 2.4 GHz es utilizada por los hornos microondas, teléfonos inalámbricos y cámaras de video inalámbricas. A pesar de ser muy utilizada, uno de las principales desventajas de utilizar esta banda es la posibilidad de interferencia.[42]

DigiMesh

Uno de los protocolos de red de malla más utilizados es el protocolo ZigBee, el cual es específicamente diseñado para una baja tasa de datos y aplicaciones de baja potencia. Sin embargo, otra alternativa para este tipo de aplicaciones es la tecnología DigiMesh la cual está basada en ZigBee (IEEE 802.15.4).

DigiMesh es una topología de red punto a punto en 2.4 GHz propietaria de Digi International para soluciones inalámbricas,la topología DigiMesh se muestra en la **Figura 2.14**. La naturaleza de la arquitectura punto a punto de DigiMesh le permite ser fácil de usar y tener funciones avanzadas de red que incluyen routers en modo “sleep” y soportar densas redes de malla. Además, ofrece una solución de bajo costo, bajo consumo de energía, largo alcance[43] y de alta tolerancia a la interferencia ya que utiliza el

[42] Coleman y Westcott 2012: 191-192
[43] DIGI INTERNATIONAL 2012

método de codificación de canal de espectro ensanchado por secuencia directa (DSSS por sus siglas en inglés[44]).

En una red DigiMesh no es necesario definir y organizar a los coordinadores, routers o puntos finales en la red como ocurre en una red ZigBee según el estándar IEEE 802.15.4[45] ya que en DigiMesh existe un solo tipo de nodo y trabaja como una red homogénea en la cual todos los nodos son intercambiables y pueden direccionar data [46] como se muestra en la **Figura 2.15**

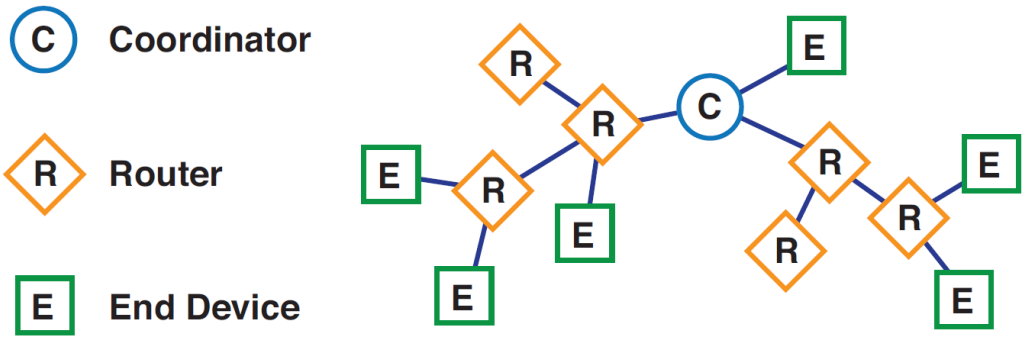


Figura 2.14 Topología ZigBee

[44] DSSS: Direct Sequence Spread Spectrum
[45] Choong 2009: 5-7
[46] Digi International 2008b

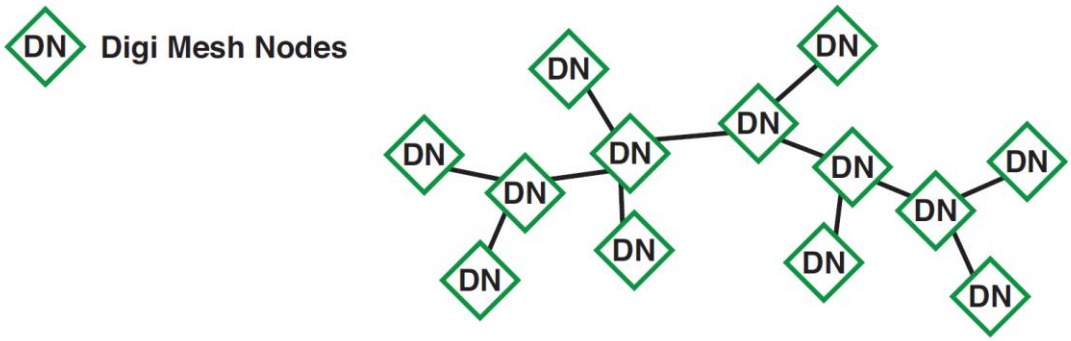


Figura 2.15 Topología DigiMesh

Sin embargo su principal desventaja es ser un protocolo propietario lo cual impide que un módulo DigiMesh se pueda comunicar con otro dispositivo inalámbrico lo cual no ocurre con el estándar ZigBee la cual tiene la posibilidad de interoperabilidad con otros productos.

Según el estándar IEEE 802.15.4 la banda 2.4 GHz contiene 16 canales que empiezan en el canal 11 y van hasta el canal 26. Los canales 0 al 10 se localizan en las bandas de 868 y 915 MHz. Los canales están espaciados 5 MHz con una ventana espectral de 2 MHz. El centro de cada canal está dado por la siguiente ecuación, donde k es el número del canal.[47]

$$F_c(k) = 2405 + 5(k - 11)\text{MHz}, \forall k \in \{11, \dots, 26\} \qquad \textbf{(2.27)}$$

[47] Choong 2009: 5

Sin embargo, los módulos XBee DigiMesh PRO sólo utilizan los canales del 12 al 23.[48]

Diseño electrónico

El diseño electrónico es una herramienta que permite crear estrategias para la resolución de problemas de manera creativa y eficiente, utilizando componentes o desarrollando circuitos de aplicación específica.

Microcontroladores PIC

Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Los microcontroladores PIC utilizan un juego de instrucciones tipo RISC[49] y se caracteriza por utilizar arquitectura Harvard, es decir posee memoria de programa y memoria de datos por separado. De la misma manera que en una computadora, el CPU es la unidad encargada de controlar la correcta ejecución de las instrucciones, dentro de ella se encuentra la ALU que se encarga de realizar todas las operaciones lógicas y matemáticas.

Se cuenta con espacios de memoria reducidos llamados “registros” para almacenar datos provenientes de la ejecución de instrucciones, que

[48] Digi International 2010
[49] RISC: “Reduced Instruction Set Computer”

pueden ser de propósitos generales o específicos[50]. Entre estos últimos están:

Registro de Instrucción (RI): Almacena la instrucción que está siendo ejecutada por el CPU.

Registro de Estado (STATUS): Indica las características del resultado de una operación, es decir si esta es negativo, o cero, si hay acarreo, etc.

Contador de Programa (PC): Almacena las direcciones de las instrucciones. Este registro "apunta" a la instrucción del programa que se ejecutará a continuación de la instrucción actual.

Registro de Direcciones de Datos (FSR): Almacena direcciones de datos situados en la memoria, indispensable para utilizar direccionamiento indirecto.

Puntero de Pila (SP): Es una estructura de datos con organización LIFO (last in, first out). El Puntero de pila contiene la dirección del tope de esta y se usa para poder almacenar direcciones de instrucciones que permitan "recordar" la dirección de retorno al programa principal desde una subrutina o interrupción.

[50] Cfr. Valdéz y Palláz 2007: 14-16

Registro W: Es utilizado al momento de realizar operaciones con la ALU, el resultado de una operación puede guardarse en el registro W o en la memoria de datos. Cuando se trabaja con instrucciones de dos operandos, necesariamente uno de estos datos debe estar en el registro W, pero en las instrucciones de simple operando, el dato se toma de la memoria. La ventaja que ofrecen los microcontroladores PIC es que el resultado de cualquier instrucción puede dejarse en la misma posición de memoria o en el registro W[51].

En un microcontrolador PIC, se tienen dos tipos de memoria, la memoria RAM y la memoria ROM. En la memoria ROM se almacena el programa que el microcontrolador va a ejecutar, mientras que en la memoria RAM se almacena los datos con los que trabaja el programa, los cuales se pierden al momento de retirar la energía del dispositivo. Las memorias de los microcontroladores se organizan por bloques llamados páginas, que son porciones de memoria de tamaño fijo, y dentro de una página, las celdas son identificadas por la posición de esta con respecto al inicio de la página. La unión de ambos conceptos constituye la dirección lógica (Dlog)[52].

Los microcontroladores PIC de gama baja poseen un repertorio de 33 instrucciones de 12 bits cada una, y una memoria de programa de hasta 2K palabras. Los microcontroladores de gama media tienen un repertorio de 35 instrucciones de 14 bits cada una, y una memoria de programa de 8K

[51] Cfr. Rincón 7:8
[52] Cfr. Valdéz y Pallaz 55:61

palabras, por último los PIC de gama alta tienen 58 instrucciones de 16 bits cada una, y la memoria de programa puede ser de hasta 64K palabras[53].

Los PIC de gama media cuentan con una amplia variedad de dispositivos de entrada y salida, además de varios puertos paralelos, temporizadores, puertos serie, etc, que los hacen idóneos para una amplia gama de aplicaciones. A continuación, en la **Tabla 2.1**, se muestra la hoja técnica de algunos microcontroladores de gama media más utilizados.

Key Features	PIC16F873A	PIC16F874A	PIC16F876A	PIC16F877A
Operating Frequency	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz	DC – 20 MHz
Resets (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
Flash Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory (bytes)	128	128	256	256
Interrupts	14	15	14	15
I/O Ports	Ports A, B, C	Ports A, B, C, D, E	Ports A, B, C	Ports A, B, C, D, E
Timers	3	3	3	3
Capture/Compare/PWM modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Analog Comparators	2	2	2	2
Instruction Set	35 Instructions	35 Instructions	35 Instructions	35 Instructions
Packages	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN	28-pin PDIP 28-pin SOIC 28-pin SSOP 28-pin QFN	40-pin PDIP 44-pin PLCC 44-pin TQFP 44-pin QFN

Tabla 2.1 Características de los dispositivos PIC (Microchip 2003)

Puente H

De manera que se pueda controlar la activación de los BJT se utilizaron controladores diseñados especialmente para su control, además estos

[53] INICTEL 2009: 50-70

controladores aíslan el circuito de control del circuito de potencia y esto se hizo con ayuda de unos dispositivos como un transformador o un optoacoplador.

Los transformadores de pulso tienen un devanado primario, pero pueden tener 2 o más devanados secundarios, de esta forma se pueden controlar simultáneamente varios transistores conectados en serie o paralelo.

Los optoacopladores combinan un diodo emisor de luz infrarroja (LED) y un fototransistor de silicio. La señal de entrada se aplica al LED y la señal de salida se toma desde el fototransistor, pero los tiempos de encendido y apagado pueden limitar su aplicación en alta frecuencia. De manera que se pueda aislar la parte de control de la parte de potencia utilizando un fototransistor se requiere a su vez un suministro de potencia aparte[54].

Este circuito permite puede construir un circuito que permita el control de un motor eléctrico DC para que este gire en ambos sentidos. A esta configuración particular se le conoce como Puente H. debido a su representación gráfica.

La conmutación intercalada de los interruptores[55] permite a un motor girar en un inicio en un sentido y luego en el otro, la representación común del puente H se muestra en la **Figura 2.16**

[54] Rashid 2004: 767-769
[55] Entiéndase interruptor como un MOSFET funcionando como tal

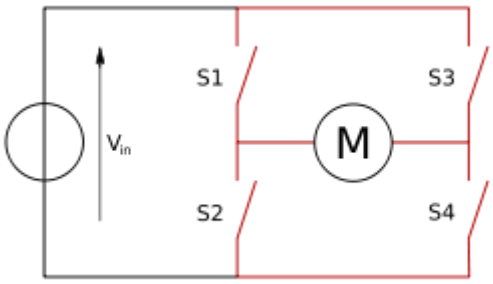


Figura 2.16 Representación de un puente H

Sensor de proximidad óptico

Los sensores de proximidad ópticos son aquellos que se componen principalmente de dos elementos, un emisor y un receptor de luz, el emisor es en la mayoría de los casos un diodo emisor de luz (LED) y el receptor es un fotodetector, que cambia su resistencia dependiendo de la intensidad de luz recibida. Dentro de este tipo de sensores, se utilizan sensores infrarrojos que son sensibles en longitudes de ondas inferiores a las visibles.

Hay dos tipos de sensores ópticos, el de barrera y el de reflexión. Los sensores de barrera, sólo pueden saber si un objeto está obstruyendo o no la luz, ya que el emisor y el receptor se encuentran separados a cierta distancia uno frente al otro. Los de reflexión, en cambio, tienen el emisor y el receptor algo separados, pero ambos sobre la misma superficie, y la distancia a la cual se encuentra un objeto se obtiene a partir del ángulo que

forma la luz enviada al rebotar en este[56], si el ángulo es grande, entonces el objeto está cerca, pero si el ángulo es pequeño, el objeto estará lejos.

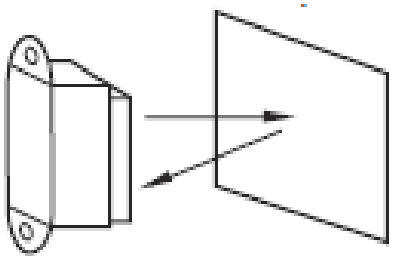


Figura 2.17 Sensor de Proximidad infrarrojo (SHARP 2005)

Estos sensores ofrecen mayor inmunidad a perturbaciones electromagnéticas externas, alta velocidad de respuesta y son capaces de detectar objetos muy pequeños[57].

Servomotor

Un servomotor es un dispositivo similar a un motor de corriente continua pero que tiene la capacidad de ubicarse en una posición dentro de un rango de operación y mantenerse en dicha posición. Generalmente trabaja con una alimentación entre 4.6 a 6 VDC, y cuenta con tres cables de señal, uno para tierra, el otro para alimentación y el tercero para la señal de control.

[56] Cfr. Ollero 2001: 178-180
[57] Cfr. Balcells y Romeral 1997: 119

Un servomotor incluye un tren de engranajes, que lo hace capaz de desplegar un gran esfuerzo de torsión o rotación, y su posición angular es controlada mediante una señal de pulso modulado.

El rotor de un servomotor común ofrece un movimiento rotacional que va desde los -90° hasta los 90°, es decir ofrece 180° de rotación, pero para que esto ocurra se debe enviar una señal de control digital (PWM). El sentido de rotación depende del tiempo del pulso que se envía. Este pulso debe ser enviado cada 20 milisegundos y si dura más de 1.5 milisegundos el rotor girará en sentido horario 90°,[58] si la duración del pulso es menor a 1.5 milisegundo el giro será de 90° pero en sentido anti horario, finalmente si la duración del pulso es exactamente 1.5 milisegundos el rotor del servomotor se pondrá en su posición neutral (0°)[59].

Se debe tener en cuenta que los engranajes que tienen internamente los servomotores ayudan al aumento del torque final, pero disminuyen la velocidad de rotación angular, por lo que un movimiento de 90° en el rotor del servomotor se puede traducir a sólo 4° o 5° del movimiento del eje final[60]. Esto dependerá del grado de reducción del tren de engranajes dado por el fabricante.

[58] Tiempo y sentido que pueden variar dependiendo del fabricante, por lo que se recomienda revisar la hoja técnica del servomotor para verificar los tiempos de funcionamiento.
[59] Cfr. Ghoshal 2010: 313-318
[60] Cfr. Ghoshal 2010: 315

Función de densidad de probabilidad

Una función de densidad de probabilidad indica cómo está distribuida la variable X a lo largo de un intervalo Y de posibles valores.

Sea X una variable aleatoria continua. Entonces, una distribución de probabilidad o función de densidad de probabilidad (fdp) de X es una función f(x) tal que para dos números cuales quiera a y b con a≤b se tiene:

$$P(a \leq X \leq b) = \int_a^b f(x)dx \qquad \textbf{(2.28)}$$

Es decir, la probabilidad de que X tome un valor en el intervalo [a, b] es el área bajo la gráfica de la función de densidad[61].

Para que f(x) sea una función de densidad de probabilidad legítima, debe satisfacer las siguientes condiciones:

$$f(x) \geq 0 \ \text{ con todas las } x$$

$$\int_{-\infty}^{\infty} f(x)dx \ = \text{área bajo la curva } = 1$$

[61] Cfr. Devore 2008: 131-133

CAPÍTULO 3. DESCRIPCIÓN DEL HARDWARE DEL SISTEMA PROPUESTO

En el presente capítulo se presenta la descripción y características del robot en el cual se implementó el sistema de visión. Además, se presenta el diseño de los circuitos analógicos y digitales de las tarjetas electrónicas necesarias para controlar el movimiento del robot y enlazar el control del robot con la estación base. Asimismo, se presentan las características técnicas del hardware adquirido (tarjeta de video, cámara inalámbrica, módulo XBee-PRO, etc.)

Sistema mecánico

Para la demostración del sistema propuesto se elaboró un sistema mecánico capaz de cumplir con las demandas del sistema de visión artificial. Es decir, una estructura que ofrezca dos grados de libertad, y desplazamiento. Asimismo una estructura capaz de soportar el peso de los motores, baterías, circuitos, cámara inalámbrica, etc. Por último se tiene una estructura de fácil adaptación a sistemas adicionales o mejoras futuras.

La estructura está hecha de polimetilmetacrilato (PMMA) ya que este ofrece alta resistencia, ligero y gran facilidad de mecanización y moldeo. El soporte de la cámara está hecho de aluminio debido a su ligereza y firmeza. El

compartimiento de las baterías está hecho de madera proporcionando estabilidad a la estructura.

En la **Figura 3.1** y la **Tabla 3.1** se muestra una imagen y las características del producto final respectivamente.

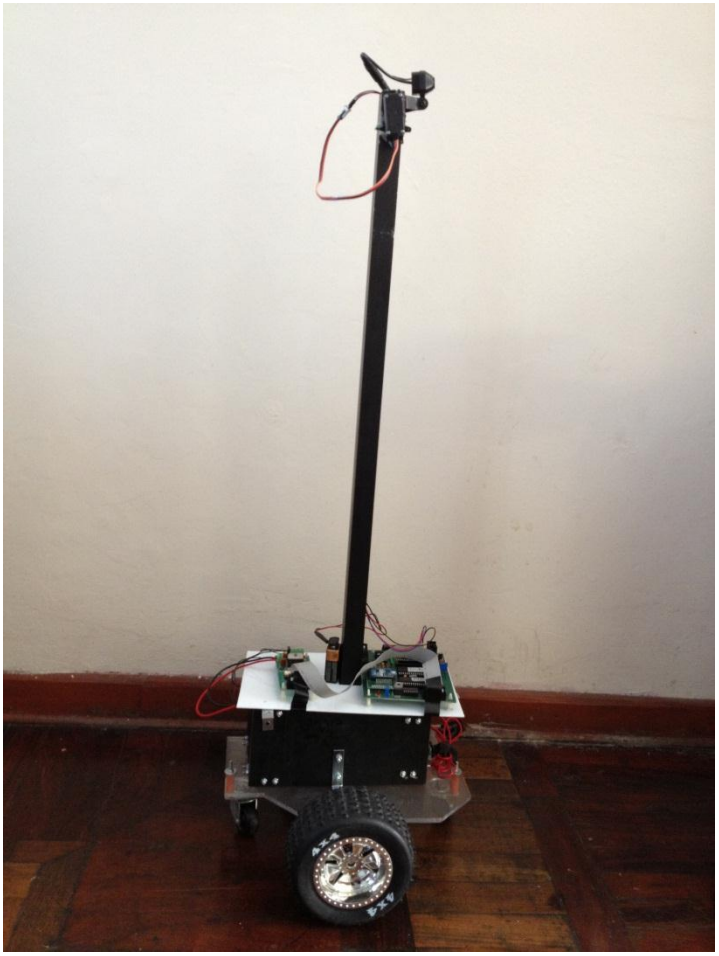


Figura 3.1 Estructura mecánica del producto final

Dimensiones (cm) (ancho x largo x alto)	40 x 30 x 90
Peso (kg)	5.5
Grados de libertad	2
Velocidad de desplazamiento (m/s)	0.4
Tiempo de autonomía (hr)	4
Rango de visión del sistema pan/tilt (°)	360/150
Alcance de visión del sistema (m)	1.3
Alcance de transmisión de datos (m)[62]	90/1600

Tabla 3.1 Características del producto final

[62] Corresponde al alcance en interiores y exteriores del módulo XBee PRO respectivamente.

Para el análisis del comportamiento del robot se realizó el siguiente modelamiento. Para el movimiento de las ruedas del robot se utilizó la siguiente ecuación

$$r_{ir} = \frac{e}{2}\left(\frac{v_r + v_l}{v_r - v_l}\right) \quad \textbf{(3.5)}$$

Donde "r_{ir}" es el radio instantáneo de rotación, "e" es la separación entre las ruedas, "v_r" es la velocidad lineal correspondiente a la rueda derecha y "v_l" es la velocidad lineal correspondiente a la rueda izquierda[63].

Al mover ambas ruedas a la misma velocidad y en sentidos contrarios se consigue que el robot gire sobre su propio eje, haciendo que el radio instantáneo de rotación sea 0.

Se optó por realizar un control secuencial por estados. El diagrama de flujo del control del robot se detalla en el siguiente capítulo.

Motores DC

De manera de lograr los movimientos de traslación y rotación del robot se utilizaron dos motores Toshiba DGM-024-2A a los cuales gracias a pruebas realizadas se pudieron determinar los parámetros necesarios para el diseño del controlador. Estas pruebas consisten en medir la corriente que consume el motor a diferentes niveles de tensión con el rotor sin carga (I nominal) y con el

[63] Cfr. AKS 2007: 163-165

rotor bloqueado (I máx). Los resultados obtenidos se muestran en la **Tabla 3.2** y la **Figura 3.2**

Voltaje de alimentación (V)	I nominal (mA)	I máx (mA)
6	70	700
9	100	1000
12	150	1500

Tabla 3.2 Resultados experimentales del motor DC

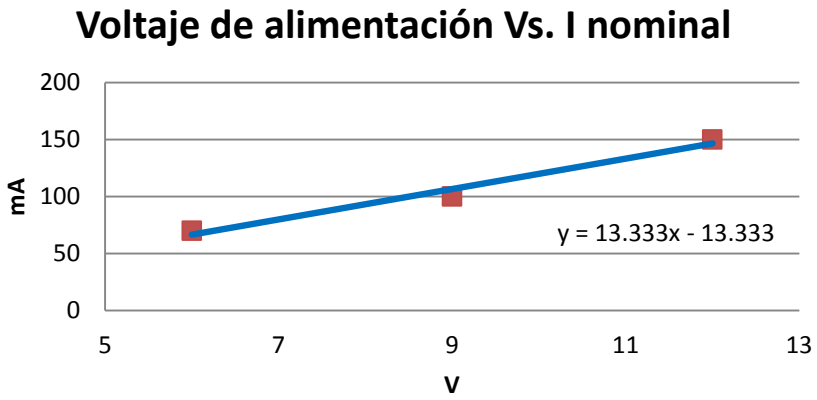


Figura 3.2 Gráfica Voltaje de alimentación vs. Corriente nominal del Motor DC

A partir de las pruebas realizadas se puede determinar el comportamiento de los motores y establecer los límites máximos de consumo, lo cual es necesario para determinar los parámetros de diseño de los circuitos electrónicos de control.

Circuitos analógicos

Puente H 2A

El siguiente controlador de potencia tiene la función de controlar los motores DC encargados de los movimientos de traslación y rotación del robot. Con los resultados obtenidos en las pruebas realizadas se decidió a utilizar los motores a una tensión de 12VDC y el CI L298 (Puente H). Este CI trabaja con una corriente nominal máxima de 2A, además es de fácil acceso en el mercado local y posee dos puentes H en un solo CI, lo que permite controlar ambos motores con un sólo dispositivo. En base a este componente se diseñó el circuito que se muestra en la **Figura 3.3**

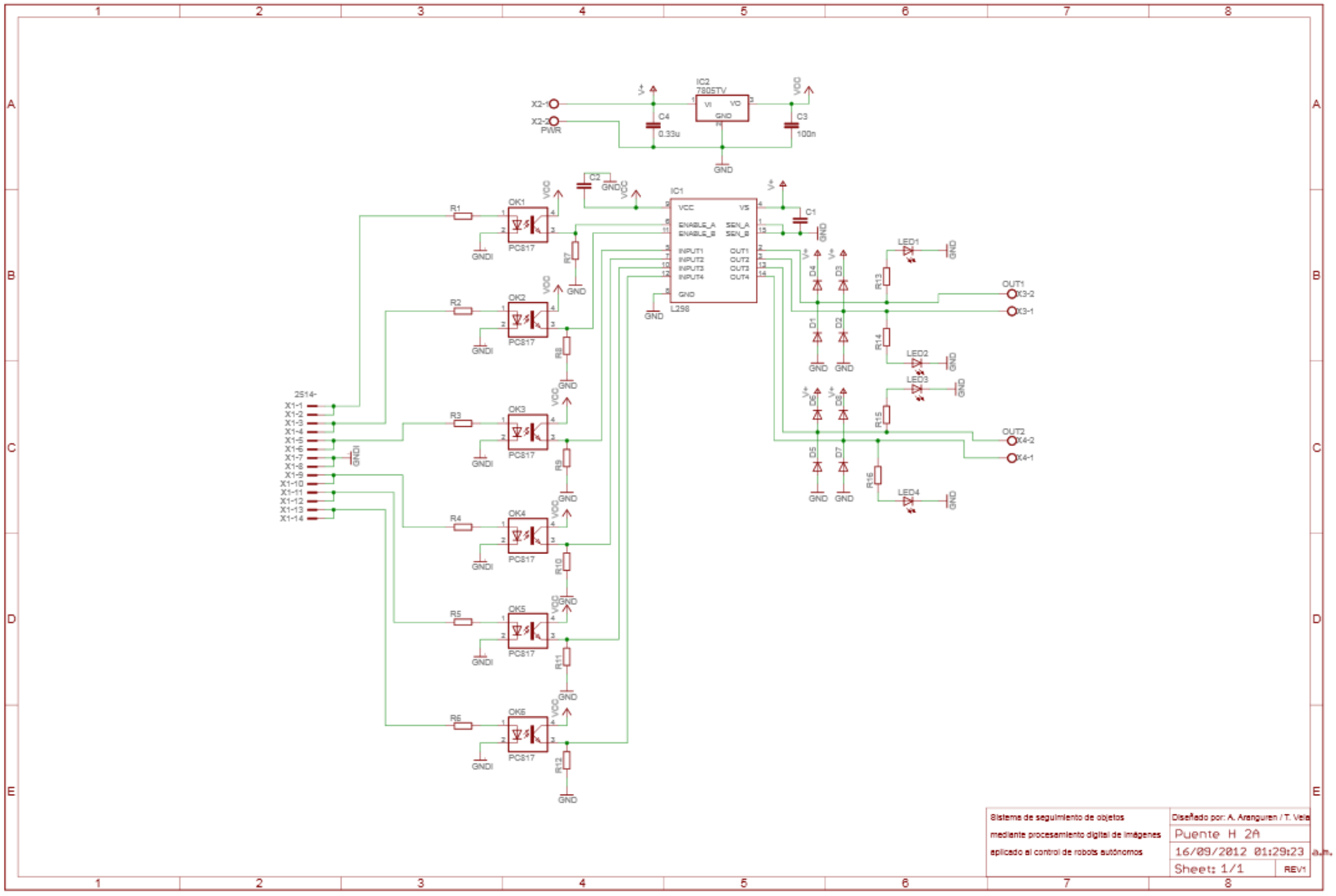


Figura 3.3 Diagrama esquemático del Puente H 2A

En este circuito se utilizan optoacopladores P521 para aislar la tierra de control digital de la tierra de potencia. Para evitar que el diodo del optoacoplador se estropee fácilmente se coloca una resistencia limitadora de corriente. Según la hoja técnica del optoacoplador[64] a una corriente directa igual a 10mA el voltaje directo típico es "V_D" =1.15V, entonces se calcula la resistencia de la siguiente manera:

$$I = \frac{V_{CC} - V_D}{R} \qquad \textbf{(3.6)}$$

Por lo cual se optó colocar una resistencia cuyo valor es de 470Ω limitando la corriente a 8.2mA.

Para lograr la polarización del transistor a aproximadamente a 2.4mA se colocó una resistencia de 2KΩ en el emisor.

Se colocó un puente de diodos 1N4004 (ID =1A) a la salida del puente H puesto que se debe proteger el circuito de corrientes remanentes en el motor.

Esta tarjeta cuenta con indicadores (LED) para verificar los comandos de movimiento del motor, además una fuente regulada de 5VDC, que se obtiene con el CI 7805 para la alimentación del puente H y el lado transistor de los optoacopladores.

[64] Cfr. Toshiba 2000

Las conexiones que se tuvieron en cuenta para el diseño de esta tarjeta se muestra en la **Tabla 3.3**:

Pin	Tipo	Descripción
X1-1, X1-2	E	Habilitación Puente H A
X1-3, X1-4	E	Habilitación Puente H B
X1-5, X1-6	E	Entrada 1 Puente H A
X1-7, X1-8	P-GND	Conexión a Tierra Digital
X1-9, X1-10	E	Entrada 2 Puente H A
X1-11, X1-12	E	Entrada 1 Puente H B
X1-13, X1-14	E	Entrada 2 Puente H B
X2-1	P-12VDC	Alimentación del Puente H
X2-2	P-GND	
X3-1	0/12 VDC	Salida Motor A
X3-2	0/12 VDC	
X4-1	0/12 VDC	Salida Motor B
X4-2	0/12 VDC	

P = Alimentación, E = Entrada digital, O = Salida digital, AN = Entrada analógica, NC = No conectado

Tabla 3.3 Conexiones de la tarjeta Puente H 2A

Se diseñó una tarjeta de doble cara con orificios metalizados de 100x45 mm, cuyo diseño del circuito impreso se muestra en la **Figura 3.4**.

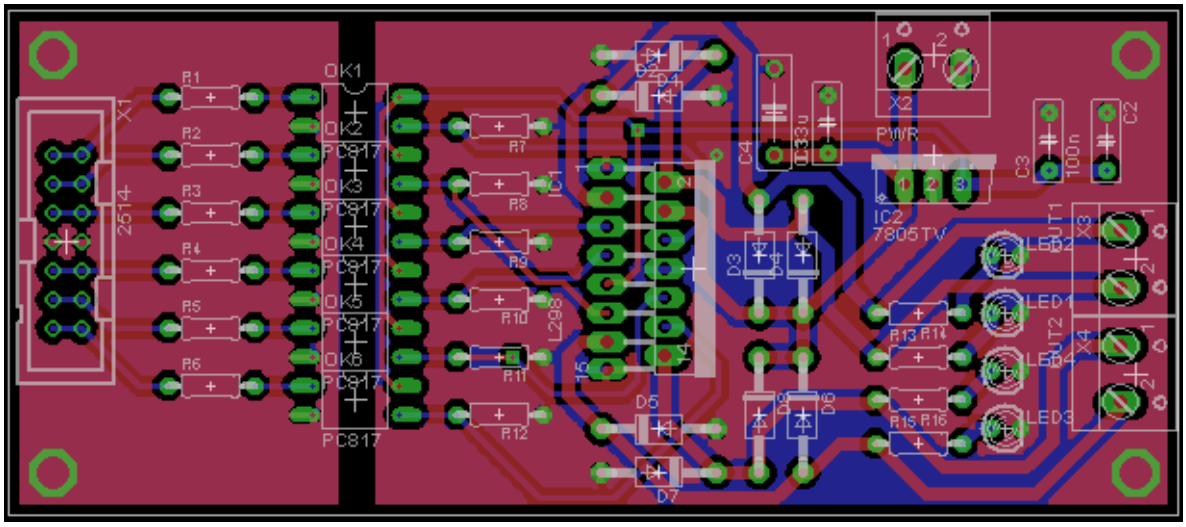


Figura 3.4 Circuito impreso de la tarjeta Puente H 2A

Circuitos digitales

Tarjeta de control

La tarjeta de control tiene como objetivo principal procesar las señales de los sensores y de la estación base para luego enviar las señales necesarias para el control de los motores, por lo que diseñó una tarjeta con dos microcontroladores PIC16F877A interconectados y un módulo XBee-PRO.

Para el diseño de la tarjeta primero se determinó el número total de entradas y salidas como muestra la **Tabla 3.4**

Pin	Tipo	Descripción
SHARP-1,2	AN	Entrada analógica del SHARP1
SHARP-3,4	P-GND	Alimentación SHARP1
SHARP-5	P-5VDC	
SHARP-6,7	AN	Entrada analógica del SHARP2
SHARP-8,9	P-GND	Alimentación SHARP2
SHARP-10	P-5VDC	
SHARP-11,12	AN	Entrada analógica del SHARP3
SHARP-13,14	P-GND	Alimentación SHARP3
SHARP-15,16	P-5VDC	
ROBOT-1,2	O	Habilitación Puente H A
ROBOT-3,4	O	Habilitación Puente H B
ROBOT-5,6	O	Salida 1 Puente H A
ROBOT-7,8	P-GND	Conexión a tierra digital
ROBOT-9,10	O	Salida 2 Puente H A
ROBOT-11,12	O	Salida 1 Puente H B
ROBOT-13,14	O	Salida 2 Puente H B
ROBOT-15,16	NC	No conectados
SERVO-1	P-GND	Alimentación Servomotor
SERVO-2	P-5VDC	
SERVO-3	O	Señal PWM para el control del Servomotor
FUENTE-1	P-12VDC	Alimentación Tarjeta de control
FUENTE-2	P-GND	

P = Alimentación, E = Entrada digital, O = Salida digital, AN = Entrada analógica, NC = No conectado

Tabla 3.4 Entradas y Salidas de la tarjeta de control

Teniendo en cuenta el número de entradas, salidas y la comunicación con el módulo XBee-PRO se determinó que era necesario implementar el sistema con dos microcontroladores PIC16F877A denominados PIC1 y PIC2. El primer microcontrolador (PIC1) se encarga de recibir y enviar información a la estación base a través del módulo XBee-PRO y enviar señales de control al

otro microcontrolador. Además, procesa las señales recibidas del sensor de proximidad infrarrojos (SHARP). El PIC2 se encarga de controlar el movimiento de traslación y rotación del robot a través del CI 298. Por último, el PIC2 se encarga a la vez del movimiento en el eje transversal "Y" ("tilt") de la cámara a través de un servomotor.

Con un arreglo de compuertas lógicas AND se agrega una señal PWM al control de los motores para regir sus velocidades. Así mismo, el módulo XBee-PRO requiere una tensión de alimentación de 3.3 VDC[65]. Esto se consiguió implementando un circuito regulador de voltaje con el CI LM317, mostrado en la **Figura 3.5**, el cual permite la regulación el voltaje de salida desde 1.2V hasta 37V a 1.5A[66] .

Para obtener el voltaje a la salida del regulador se aplica la siguiente ecuación:

$$V_{OUT} = V_{REF}\left(1 + \frac{R_2}{R_1}\right) + I_{ADJ}R_2 \qquad \textbf{(3.7)}$$

[65] Cfr. Digi International 2010: 4-6
[66] Cfr. National Semiconductor 2011

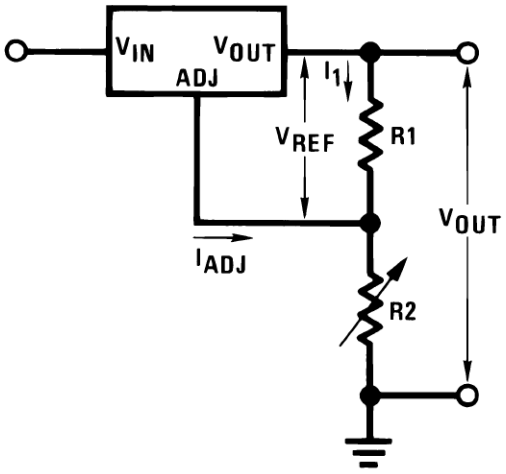


Figura 3.5 Circuito típico del CI LM317 (National Semiconductor 2011)

En operación el CI LM317 tiene un valor nominal de "V_{REF}" = 1.25V e "I_{ADJ}" =50µA (este último se puede despreciar dado que es muy pequeño). Para obtener 3.3V a la salida del regulador se diseñaron los valores de las resistencias como sigue: "R1" = 470Ω y "R2" = 620Ω.

Por último para obtener el valor de "R2" y tener un ajuste fino a la salida del regulador se coloca en serie a la resistencia "R2" un potenciómetro de 500Ω.

El diagrama esquemático del circuito implementado se muestra en la **Figura 3.6**

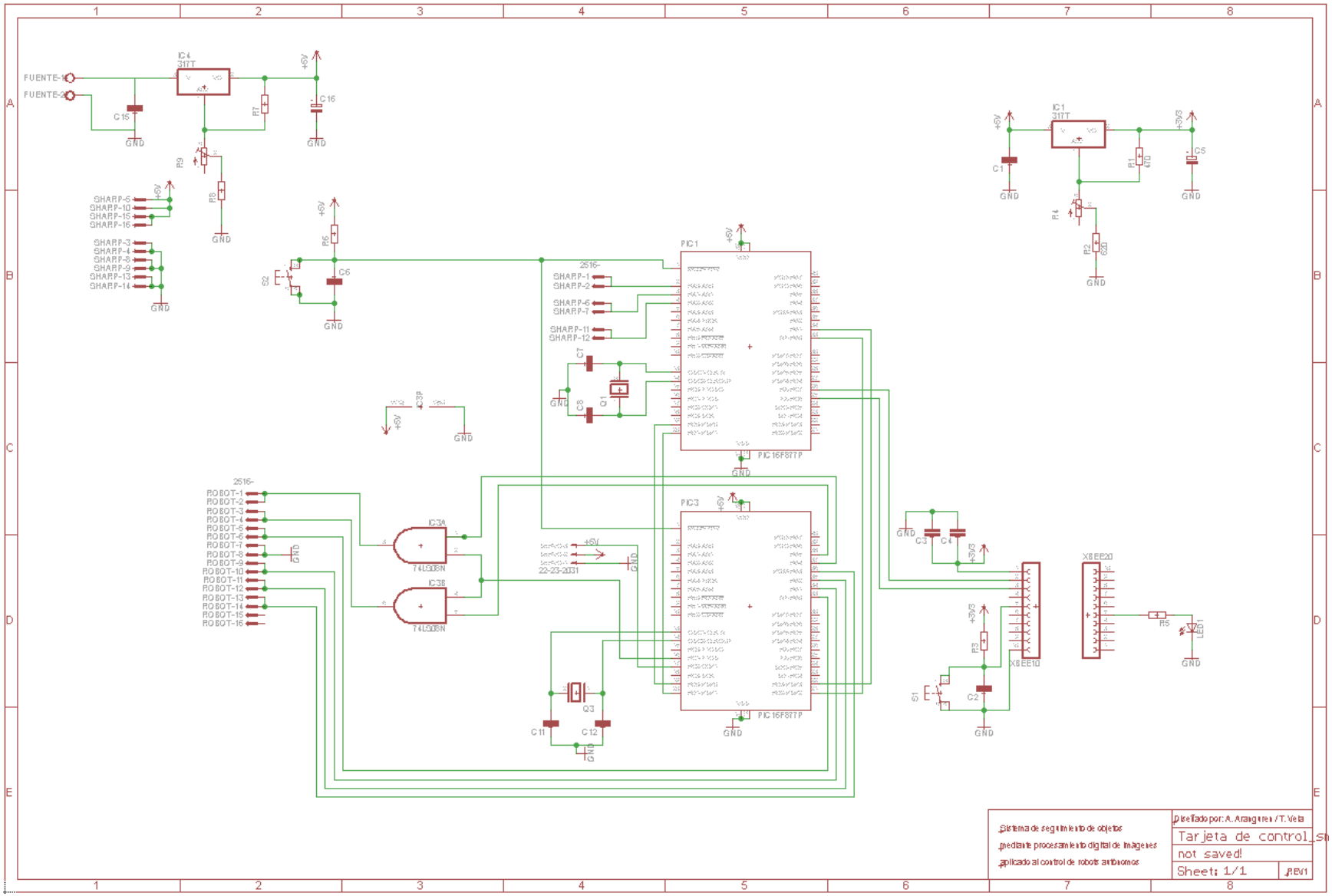


Figura 3.6 Diagrama esquemático de la tarjeta de control

Las conexiones entre los microcontroladores y los otros componentes se muestran en la **Tabla 3.5**.

PIC1	PIN
SHARP1	RA0
SHARP2	RA1
SHARP3	RA2
XBee-PRO – Rx	RC7
Xbee-PRO – Tx	RC6
PIC1-PIC2	RD0, RD1,RB0,RB1

PIC2	PIN
SERVO	RC1
ROBOT	RB0..5
PIC3-1	RD0..3

Tabla 3.5 Conexiones internas de la tarjeta de control

Finalmente, se muestra el diseño del circuito impreso de doble cara con hueco metalizado de 138x110mm en la **Figura 3.7**

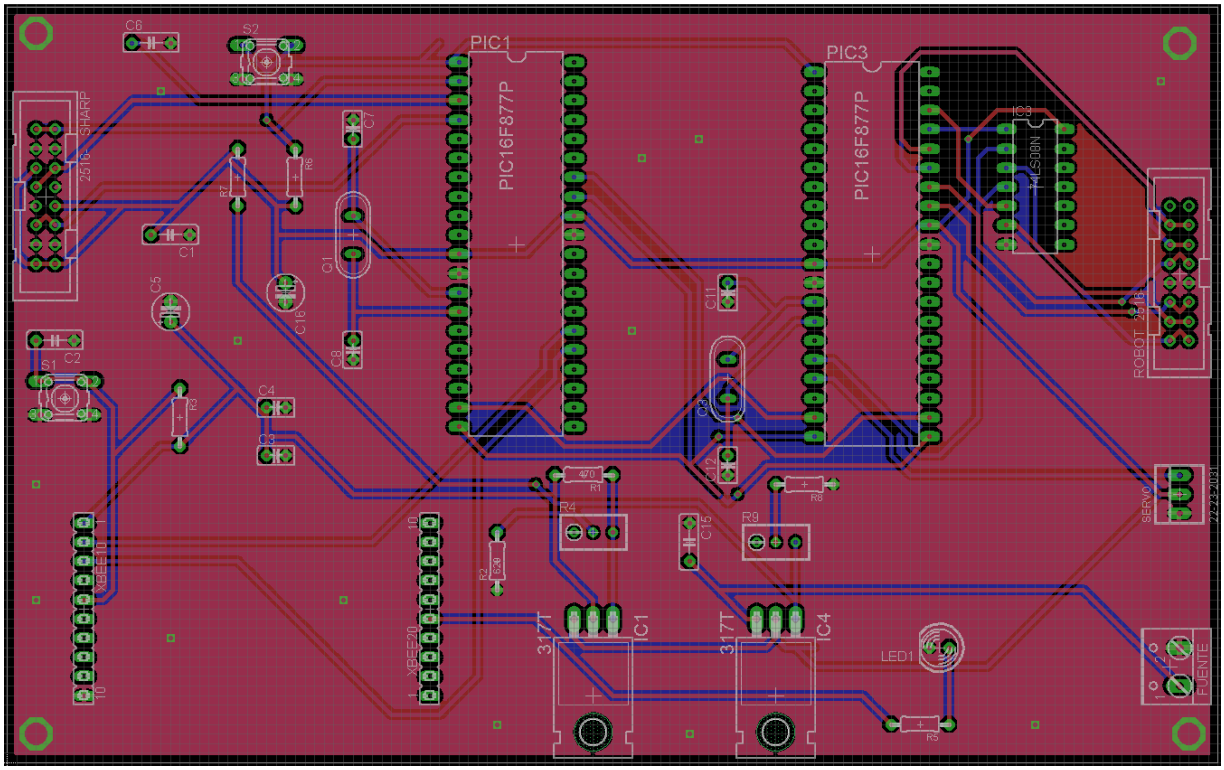


Figura 3.7 Circuito impreso de la tarjeta de control

Una vez implementados los circuitos analógicos y digitales, se realizaron pruebas en frío. Es decir, se verificó que no haya cortocircuitos entre las pistas o cortocircuitos internos en los dispositivos. Asimismo se verificó el aislamiento de las fuentes y el buen estado de los componentes.

Una vez probados los circuitos, es posible ensamblar los programas encargados de realizar la comunicación con la estación base, recibir comandos y dar movimiento al robot.

Programas ensamblados

De manera que se logre una comunicación eficaz entre la estación base y el robot, es vital contar con los programas encargados de procesar información permitiéndole al robot actuar de acuerdo a las ordenes enviadas por la estación base y los sensores del sistema.

A continuación se detallará el funcionamiento de cada uno de ellos. El PIC 1 es el encargado de recibir y enviar información a la estación base a través del módulo XBee-PRO y a su vez es el que envía señales de control al microcontrolador restante y recibe las señales del sensor infrarrojo instalado en el robot para evitar que este impacte con algún objeto al realizar movimientos.

El sensor infrarrojo envía señales analógicas al PIC1, este las procesa y si se registra la presencia de un objeto a una distancia menor o igual a 30cm el robot se detendrá

El diagrama de flujo de este programa se muestra en el **Anexo 1**

Los comandos enviados desde la estación base para las órdenes de dirección del robot son los que se muestran en la **Tabla 3.6**

Robot	
Avanzar	"W"
Derecha	"D"
Izquierda	"A"
Parar	"P"
Servomotor	
Arriba	"I"
Abajo	"K"
Parar	"U"

Tabla 3.6 Comandos de control del robot

El PIC 2 es el encargado de la traslación y rotación del robot mediante el control de los motores DC. El control de velocidad de los motores por PWM se encuentra activado al 60% debido a la necesidad de reducir la velocidad para mayor precisión en los movimientos.

Dependiendo de las señales que recibe el PIC2 desde el PIC1, se realiza el movimiento del robot con ayuda del puerto B, según lo que muestra la **Tabla 3.7**.

B0	**B1**	**B2**	**B3**	**Acción**
1	0	1	0	Avanzar
0	1	1	0	Izquierda
1	0	0	1	Derecha
0	0	0	0	Parar

Tabla 3.7 Condiciones del puerto B para el PIC2

Además el PIC2 es el encargado del movimiento del servomotor, haciéndolo girar en sentido horario o anti horario los grados necesarios según corresponda.

Al inicio del programa se crea una señal PWM para el control de la velocidad de los motores y el servomotor se centra a la posición 0°.

Según la hoja técnica del servomotor se obtuvo que trabaja con una señal de control de un periodo de 20 ms, y para centrar el servomotor se envía al inicio un pulso de 1.5 milisegundos de duración[67]. Los valores del ancho del pulso se van actualizando a medida que las instrucciones desde el PIC1 se van recepcionando, hasta lograr un movimiento completo del servo desde los -90° hasta los 90°.

Las condiciones que envía el PIC1 al PIC2, según los requerimientos del sistema de control se muestran en la **Tabla 3.8**

Robot		
D2	**D3**	**Acción**
0	1	Izquierda
1	0	Derecha
0	0	Para
1	1	Avanzar
Servomotor		
D0	**D1**	**Acción**
0	1	Arriba
1	0	Abajo

Tabla 3.8 Condiciones del puerto D para el PIC2

[67] La manera de trabajo del servomotor se detalló en el capítulo 2.

En el **Anexo 2** se muestra el diagrama de bloques del programa

Enlace de datos

Dada la necesidad de realizar una comunicación inalámbrica de mediano alcance segura, eficaz, de bajo costo, tamaño y bajo consumo se decidió utilizar los módulos XBee-PRO. En la **Tabla 3.9** se muestran las especificaciones técnicas del XBee-PRO.

	Xbee-PRO DigiMesh 2.4
Tasa de datos RF	250 kbps
Alcance en interiores	90 m
Alcance en exteriores	1.6 km
Potencia de transmisión	63 mW (+18dBm)
Interfaz de datos serial	3.3V CMOS serial UART
Método de configuración	AT & API
Banda de frecuencia	2.4 GHz ICM
Inmunidad a la interferencia	DSSS (Espectro Ensanchado por Secuencia Directa)
Tasa de datos serial	Hasta 115.2 Kbps
Entradas analógicas	(6) 10-bit ADC
Entradas/Salidas digitales	13
Antena	Chip
Canales	12
Alimentación	2.8 – 3.4VDC
Corriente de transmisión	250 mA
Corriente de recepción	55 mA
Corriente desocupado	<50 uA

Tabla 3.9 Especificaciones técnicas del XBee-PRO

El módulo XBee-PRO DigiMesh 2.4 proporciona un alcance de 90m en interiores y hasta 1.6km al aire libre con línea de vista, más que suficiente para la

aplicación. Además, ofrece una tasa de transmisión inalámbrica de 250Kbps en la banda de 2.4GHz. Por último, su bajo consumo de energía (Tx/Rx/Espera 250mA/55mA/<50uA), su fácil conexión para la puesta en marcha, su bajo costo, además de ser configurable, lo vuelve ideal para esta aplicación.

El módulo XBee-PRO se conecta a otros dispositivos a través de un puerto serial asíncrono. A través de este, el módulo se puede comunicar con cualquier UART compatible en lógica y voltaje; o a través de un traductor de nivel a cualquier dispositivo serial. Por ello la conexión entre el módulo XBee-PRO y el microcontrolador PIC16F877A se puede hacer de forma directa, mientras que para la conexión entre el módulo y la estación base se requiere de una tarjeta de interfaz Serial/XBee-PRO.

Debido a que la frecuencia de operación de la cámara inalámbrica se ubica en los 2.414 GHz y la frecuencia de operación del módulo XBee-PRO es en el canal 12 (0x0C) por defecto, cuya frecuencia central corresponde a 2.410 GHz según el estándar IEEE 802.15.4; al transmitir datos a través del módulo XBee-PRO se causaba interferencia en las imágenes recibidas por la cámara inalámbrica. Para resolver este inconveniente se optó por cambiar el canal utilizado en el módulo XBee-PRO por el canal 23 (0x17) que según el estándar IEEE 802.15.4 cuenta con una frecuencia central que corresponde a 2.435 GHz[68], evitando cualquier inconveniente de interferencia con la cámara inalámbrica. Esto se logró a través

[68] Cfr. Digi International 2008a

la opción de configuración de los módulos XBee-PRO mediante la utilización de comandos AT. El software de libre distribución utilizado fue X-CTU.

Tarjeta de interfaz Serial/XBee-PRO

La necesidad de intercambiar información directa entre la estación base y la tarjeta de control conllevó a diseñar una tarjeta de interfaz entre el computador y el módulo XBee-PRO la cual provee las herramientas necesarias para una comunicación segura y rápida.

La comunicación entre el computador y la tarjeta de interfaz se realiza mediante el protocolo de comunicación serial. La tarjeta de interfaz XBee-PRO utiliza el CI MAX232 como enlace entre el XBee-PRO y el puerto serial del computador, ya que los niveles de tensión no son compatibles.

Al igual que en la tarjeta de control, fue necesario implementar una fuente regulada a 3.3V utilizando la fórmula 3.2 y obteniendo por ende los mismos valores de resistencia que en la tarjeta de control.

Las únicas conexiones requeridas para la comunicación entre los módulos son VCC, GND, DIN, DOUT. Adicionalmente se conectaron condensadores de 1

uF y 8.2pF entre VCC y GND para reducir el ruido; y por último se implementó el circuito de Reset y un LED que advierte el estado de los módulos.[69]

El diagrama esquemático del circuito implementado se muestra en la **Figura 3.8**.

[69] Cfr. Digi International 2010: 7-8

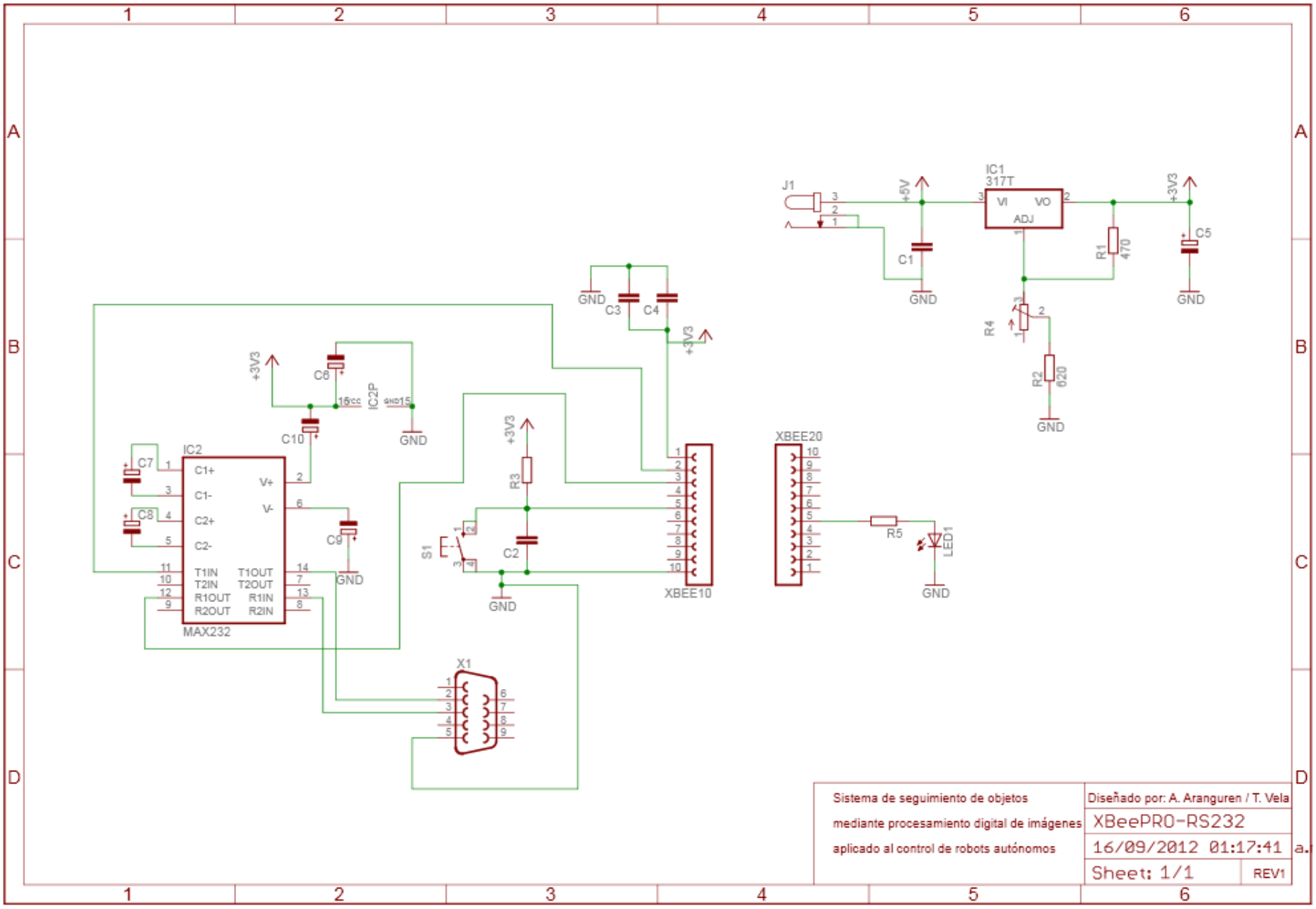


Figura 3.8 Diagrama esquemático de interfaz Serial/XBee-PRO

El circuito impreso a una cara simple de 90x80mm se muestra en la **Figura 3.9**.

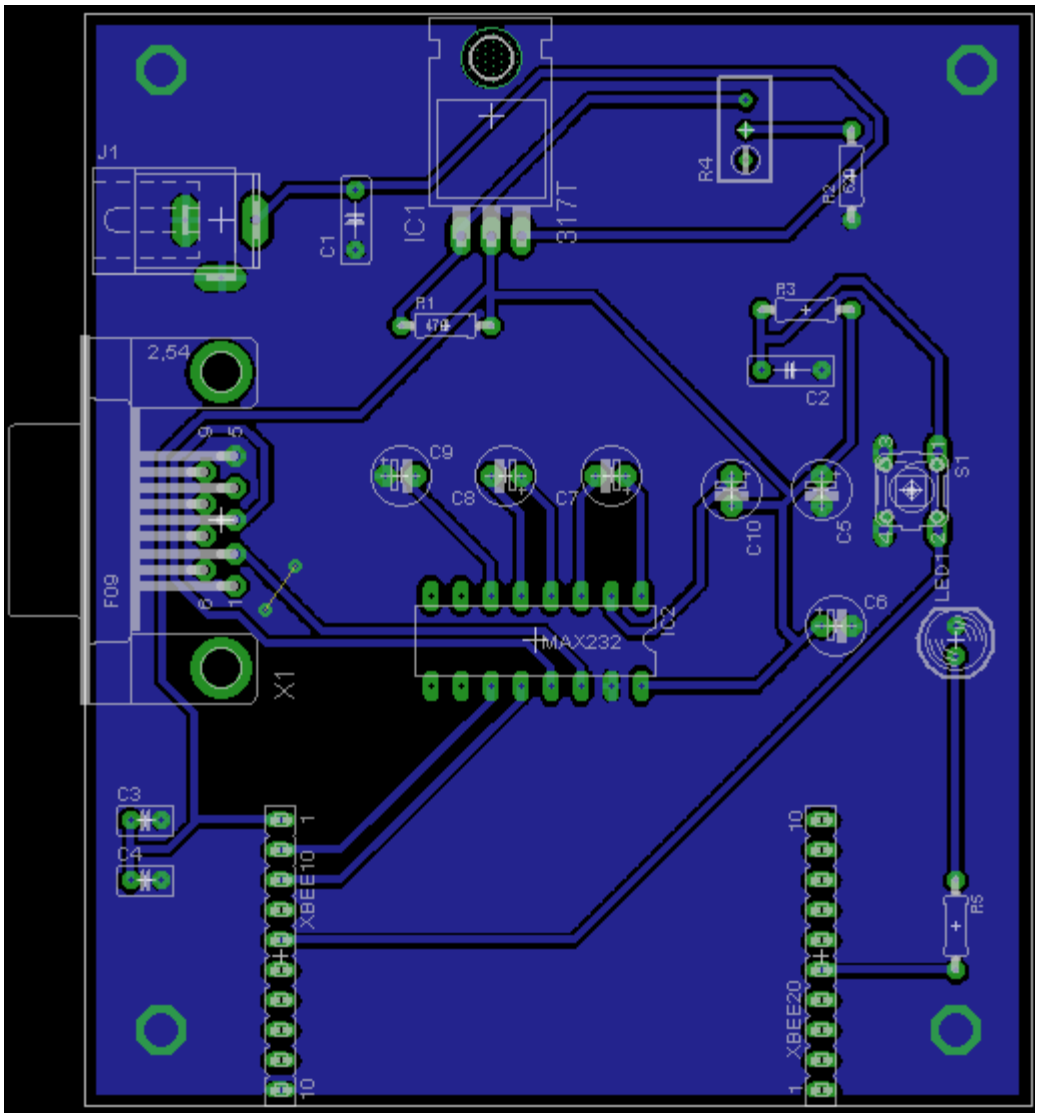


Figura 3.9 Circuito impreso de interfaz Serial/XBee-PRO

Sensor de proximidad infrarrojo: Sharp GP2D12

Se decidió usar este sensor infrarrojo en particular debido a que éste es capaz de detectar eficientemente objetos de cualquier color y de diferentes ratios de reflectividad (debido a la técnica usada para sensar la distancia). Además, otra consideración que se tomó en cuenta fue el rango de distancias que puede detectar el sensor: de 0 a 80 cm, que se consideró suficiente para esta aplicación.

Para probar el funcionamiento del sensor, se implementó el siguiente circuito:

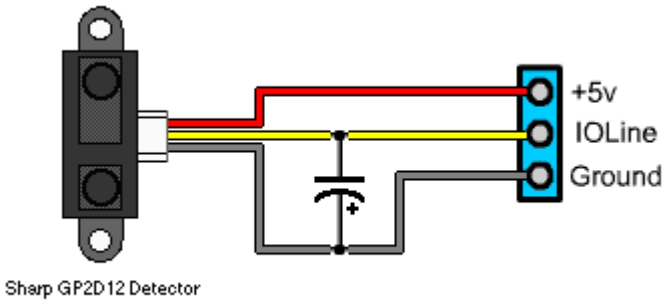


Figura 3.10 Conexión del sensor SHARP GP2D12

Se tomaron las mediciones pertinentes, con dos objetos: una hoja de papel blanco, que resultó ser bastante cercana a la proporcionada por la hoja técnica del sensor, y una de tela negra (línea roja, bastante cercana a la curva del papel blanco), mostrándose los resultados en la **Figura 3.11**

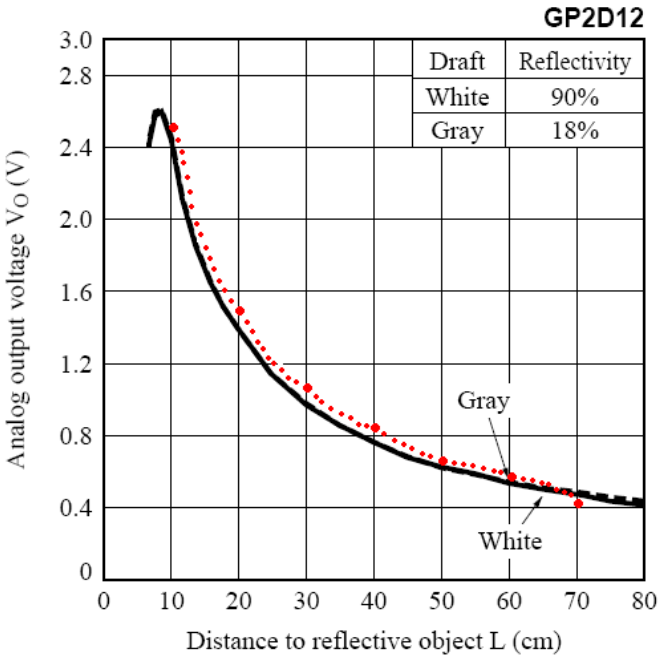


Figura 3.11 Gráfica de Voltaje (V) vs Distancia al objeto (cm) (SHARP 2005)

Este sensor es utilizado con la finalidad de proteger al robot de posibles colisiones con objetos de su entorno. En especial para evitar que choque con personas al momento de desplazarse.

El sensor está ubicado en la parte central delantera del cuerpo del robot.

Cámara RF y receptor 2.4 GHz

Para la adquisición de las imágenes se optó por utilizar una cámara inalámbrica 2.4 GHz con las características que se muestran en la **Tabla 3.10**:

	Cámara inalámbrica
Frecuencia de Transmisión	ISM 2.400 - 2.483 MHz
Potencia de Transmisión	10mW/CE; 2mW/FCC
Rango efectivo sin obstrucciones	100m
Tipo de Sensor de imagen	CMOS
Pixeles totales de la imagen	PAL: 628 x 582; NTSC: 510 x 492
Resolución horizontal	380 líneas de TV
Cuadros por segundo	30fps
Ángulo de Visión	PAL: 60°; NTSC: 40°
Corriente consumida	200mA
Suministro de potencia	8V DC
Dimensiones	23.2*22.5*51.2mm
Peso	45g
Corriente consumida del receptor	200mA
Suministro de potencia del receptor	8V DC
Dimensiones del receptor	84*45*20mm
Peso del receptor	80g

Tabla 3.10 Características de la cámara inalámbrica

De estas características las de mayor ponderación para la elección de la cámara fueron: la frecuencia de transmisión, ya que se encuentra en una banda de uso libre; los cuadros por segundo, ya que con 30 fps se tiene una calidad aceptable; su rango efectivo de 100 m, su bajo consumo de energía (8V – 200mA) y su liviano peso ya que debía ser montada en el robot.

Dentro de las opciones que ofrece la cámara, se eligió utilizar el formato NTSC ya que al utilizar el espacio de color YCbCr agilizaba el procesamiento de imágenes al tener las componentes de crominancia por separado.

El canal configurado en la cámara adquirida transmite el video en la frecuencia central de 2.414 GHz sin posibilidad de cambio.

Fuente de alimentación

De manera que se pueda alimentar el robot se utilizan tres baterías. Para el movimiento de los motores se decidió utilizar una batería Pb-Ácido de 12V, 4A-h. Para el controlador del puente H y la tarjeta de control se utilizó otra batería Pb-Ácido de 12V, 4A-h. La utilización de dos baterías se debe a que se desea aislar la tierra de control de la de potencia para evitar la propagación de ruido. Por último, para la cámara inalámbrica se utiliza una batería alcalina de 9V.

Se decidió sobredimensionar las baterías para obtener un mayor tiempo de autonomía del robot ya que se requería realizar varias pruebas de funcionamiento.

Capturador de video

Para poder procesar el video analógico en un computador (estación base) era necesario digitalizar el video, por lo que se adquirió un capturador de video que convierte el video analógico desde un conector RCA de video compuesto a USB.

Este dispositivo tiene una tasa de bits que varía según el formato que se utilice de la siguiente manera:

NTSC 720x480	6Mbps
NTSC 480x480	2.42Mbps
NTSC 320x240	1.15Mbps

En el proyecto se utiliza el formato NTSC 320x240 ya que con un menor número de píxeles se tiene un procesamiento más veloz.

Tarjeta de video

Un GPU es una unidad de procesamiento de gráficos que sirve como coprocesador para el CPU anfitrión, tiene su propia unidad de memoria y ejecuta varios hilos (threads) en paralelo. Este dispositivo de cómputo tiene varios núcleos, pero una arquitectura más simple que un CPU estándar. Debido a la capacidad de operar grandes cantidades de datos en paralelo, la utilización de un GPU resulta más efectiva que la utilización de un CPU normal.

CUDA es la arquitectura de procesamiento paralelo de NVIDIA que permite un incremento en el desempeño computacional aprovechando al máximo los recursos del GPU.[70]

[70] NVIDIA Corporation 2010a

Para el proyecto se adquirió la tarjeta de video NVIDIA GeForce GTX 465 que cuenta con las siguientes características.[71]

	NVIDIA GeForce GTX 465
Núcleos CUDA	352
Reloj de gráficos	607 MHz
Reloj de procesador	1215 MHz
Rendimiento de cálculo	30x
Reloj de memoria	1603 MHz (Tasa de datos 3206)
Memoria estándar	1024 MB
Interfaz de memoria	GDDR5
Ancho de interfaz de memoria	256 bit
Ancho de banda de memoria	102.6 GB/s

Tabla 3.11 Características de la tarjeta de video

Dentro de la familia GeForce de NVIDIA este modelo de GPU se encuentra dentro de la gama de baja capacidad ya que actualmente se encuentran en el mercado GPU para PC de escritorio con tecnología CUDA de hasta 3072 núcleos (GeForce GTX 690). [72]

[71] NVIDIA Corporation 2012a
[72] NVIDIA Corporation 2012b

CAPÍTULO 4. DESCRIPCIÓN DE LA ETAPA DEL SOFTWARE DEL SISTEMA PROPUESTO

El software es el encargado de realizar el procesamiento digital de las imágenes recibidas por la cámara inalámbrica. A su vez, se encarga de enviar los comandos necesarios para lograr el movimiento del robot desde la estación base hacia éste.

En el presente capítulo se muestra qué algoritmos han sido implementados con CUDA y la manera de implementarlos. Asimismo, se muestran los diagramas de bloques de los modos de operación implementados en los anexos con una explicación detallada de los mismos. Por último se presenta la interfaz visual del software propuesto.

Descripción

El software está desarrollado en MATLAB ya que ofrece un entorno de desarrollo integrado (IDE) fácil de utilizar y con el cuál se está familiarizado. Además, MATLAB está diseñado para trabajar con matrices, permite la creación de interfaces de usuario (GUI) y facilita la comunicación con otros lenguajes y con otros dispositivos hardware.

El software está potenciado con el modelo de programación CUDA, desarrollado por NVIDIA, lo cual permite un incremento significativo en el rendimiento al aprovechar el potencial de procesamiento paralelo del GPU. Este modelo utiliza una variación del lenguaje de programación C lo cual permite acceder al GPU de manera más sencilla ya que no es necesario utilizar lenguaje ensamblador.

La programación en CUDA tiene ciertas particularidades que se deben tomar en cuenta, como por ejemplo:

- Nuevas sintaxis: Provee calificadores de tipo de función, que ayudan a identificar donde se procesarán las instrucciones[73].

  __global__: el código debe correr en el GPU, pero llamado desde el anfitrión.

  __device__: el código debe correr en el GPU y la función solo puede ser llamada por código que corre en el GPU.

  __host__: el código debe correr en el CPU anfitrión, y debe ser llamado por el mismo, este tipo es el predeterminado.

- La manera de llamar a una función también cambia[74] con respecto al lenguaje C:

  kernel<<<dim3 grid, dim3 block>>>(arg1,arg2,arg3… argn)

[73] Cfr. Peardon 2009.
[74] Cfr. MathWorks 2010

- Restricciones:
  - No permite la recursividad
  - No permite variables estáticas
  - No permite un número variable de argumentos

- Los archivos creados en CUDA tiene la extensión .cu, estos contienen tanto código en CUDA (instrucciones que corren en el GPU) como código en C (instrucciones que corren en el CPU).

- Para compilar este código es necesario utilizar el compilador nvcc, el propósito de este compilador es ocultar los detalles complejos de la compilación de CUDA a los programadores, además se apoya en otros compiladores de C, como el Visual Studio para compilar el código en C[75].

Para acelerar los algoritmos desarrollados anteriormente utilizando la tecnología CUDA se tienen dos opciones:

- Integrar CUDA con Matlab creando un archivo .cu que implementa una API (Interfaz de Programación de Aplicaciones) para luego ser compilado por Matlab a través del compilador MEX de tal manera que se puede utilizar la función como cualquier otra función .m de Matlab.

[75] Cfr. NVIDIA Corporation 2010b

- Migrar los algoritmos desarrollados en Matlab a lenguaje C de tal manera que sólo se utilizaría el compilador nvcc.

Debido a que no todo el código está desarrollado en Matlab es necesario que sea ejecutado en el GPU se optó por la primera opción. Las funciones que demandan mayor procesamiento y por ende mayor tiempo computacional fueron transferidas al GPU mientras que las funciones de adquisición de imágenes permanecieron en el entorno de Matlab.

Para poder compilar el archivo .cu y ejecutarlo en Matlab fue necesario utilizar una función especial dentro del programa escrito para CUDA, llamada mexFunction, cuya sintaxis es:

mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])

Dónde[76]:

nlhs es el número esperado de salidas

plhs es un arreglo de punteros de las salidas esperadas

nrhs es el número de entradas

prhs es un arreglo de punteros para las entradas

Luego de programar el archivo.cu se procede a compilarlo con ayuda de un programa llamado cuda_mex.m [77] este programa utiliza el compilador de

[76] Todos estos argumentos deben de ser del tipo mxArray.

MATLAB (mex) y el compilador de NVIDA (nvcc) de manera que se obtiene un programa que se puede llamar desde MATLAB pero que se ejecuta en la tarjeta de video.

Descripción del Software

Para el “Tracking Mode” se desarrolló un programa capaz de detectar una marca determinada a diferentes distancias y condiciones de iluminación. El diagrama de bloques para el programa es el siguiente:

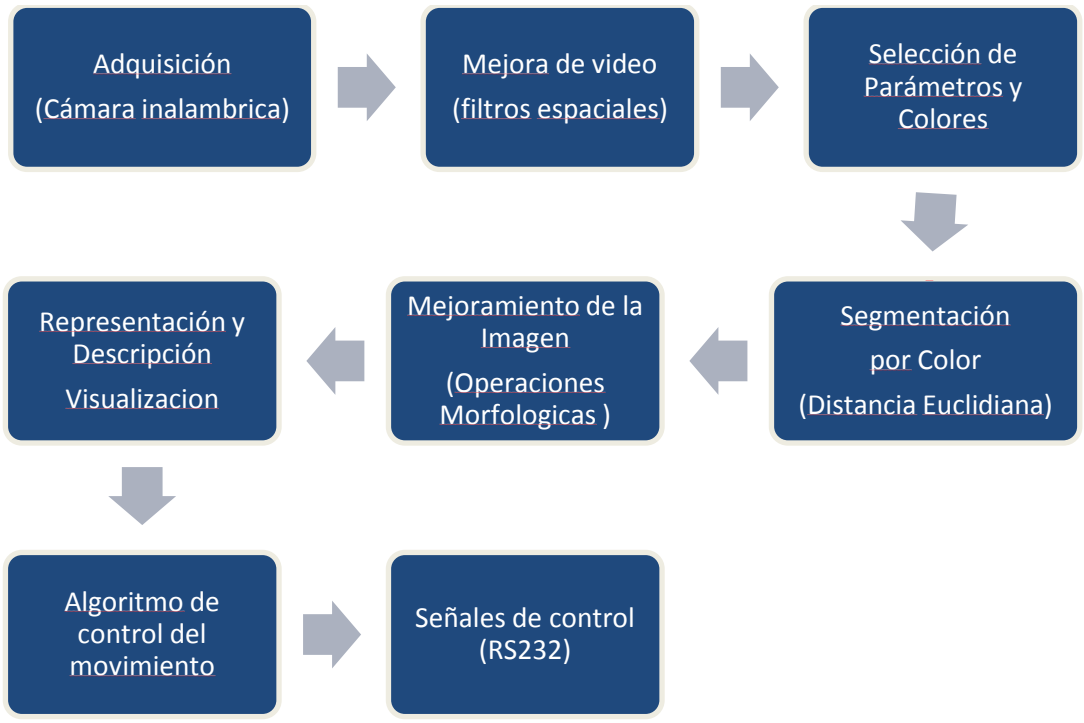


Figura 4.1 Diagrama de bloques del modo “Tracking Mode”

[77] Este archivo es un archivo descargado de MATLAB central.

El algoritmo comienza seleccionando los dos colores que serán utilizados para el reconocimiento de la marca, previamente se utiliza un filtro de mediana en la imagen para evitar seleccionar un color que no corresponde al deseado. Se debe tener en cuenta que inicialmente el objeto tiene que permanecer estático para seleccionar los colores deseados.

Una vez que se da pase al seguimiento, mediante un lazo infinito se capturan los cuadros para ser procesados uno a la vez. Dependiendo de la opción elegida por el usuario en el panel “All Weather Vison”, el algoritmo selecciona el valor del factor Gamma apropiado. Luego, se utiliza un filtro de mediana sobre el cuadro a modo de preprocesamiento de la imagen. A la componente de luminosidad de la imagen filtrada se aplica la corrección Gamma mediante la ecuación 2.14. Esta función se implementa de manera que se ejecute en el GPU.

Para el modo de “Night Vision” se eligió un valor de Gamma de 0.9. Este valor fue seleccionado mediante una evaluación cualitativa de diferentes valores entre 0.5 a 1 ya que en este rango la imagen era visible y su brillo aumentaba. Por otro lado, para el modo “Sun Block” se creyó conveniente utilizar un valor de 1.07 ya que no se deseaba oscurecer tanto la imagen y sólo eliminar brillos intensos.

A continuación, se realiza la segmentación de la marca buscando el grado de similitud entre las componentes de crominancia (“Cb” y “Cr”) de los colores

preseleccionados y las componentes de crominancia de la imagen. Obteniendo una imagen binaria como se muestra en la **Figura 4.2**

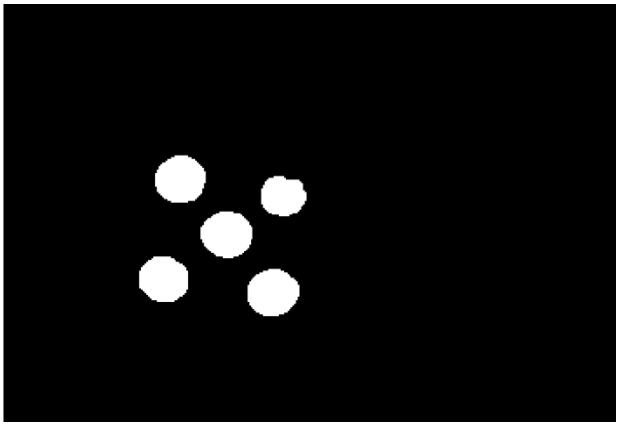


Figura 4.2 Imagen binaria con objetos segmentados

Esto se realiza hallando la distancia euclidiana entre las componentes de crominancia basándose en la fórmula 2.20. Se tiene un plano con las componentes Cb y Cr de la imagen donde un color específico representa un punto $\mathrm{p}(\mathrm{x},\mathrm{y})$ en el plano Cb-Cr. Además, se tiene un color preseleccionado $\mathrm{q}(s,\mathrm{t})$. Por lo que se puede hallar la distancia (D_e) entre p y q.

Para discriminar los valores se utiliza un valor umbral de 0.085 que representa el 94% de similitud entre los colores seleccionados y los de la imagen actual. Esto se debe a que la operación de distancia euclidiana da como resultado 0 cuando los colores son idénticos y $\sqrt{2}$ cuando la diferencia es máxima.

Al igual que la corrección Gamma esta función se implementa para ser ejecutada en el GPU.

A la imagen segmentada se le aplica los filtros morfológicos de relleno de regiones, cierre, apertura, y eliminación de bordes. Una vez filtrada la imagen se procede a etiquetar cada uno de los objetos encontrados. De no hallar objetos en la imagen filtrada se procede a realizar la búsqueda de la marca haciendo un barrido secuencial hasta cubrir 360°. De no hallarse la marca después del algoritmo de búsqueda, el robot se detiene.

Si se identifica uno o más de un objeto en la imagen se procede a hallar el centroide de cada uno y colocarlo en la imagen como se muestra en la **Figura 4.3**

Figura 4.3 Imagen con la marca identificada

Luego se calcula el punto central de la marca utilizando la media de los centroides para mover el robot de manera que la marca se posicione en el centro

de la imagen dentro de los límites establecidos: M/4 y N/4. Donde “M” es el número de filas de la imagen y “N” el número de columnas.

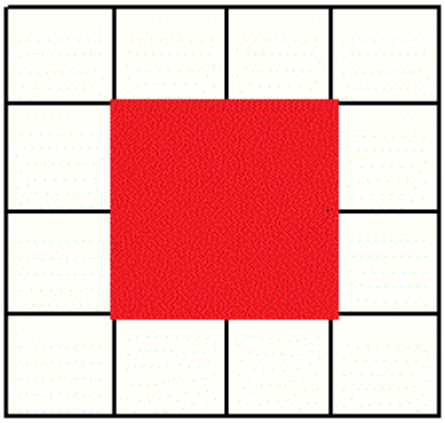


Figura 4.4 Región donde se considera a la marca centrada

Los comandos son enviados al robot mediante el puerto serial del computador a través de la tarjeta de interfaz Serial/XBee-PRO.

Si la cantidad de objetos detectados es mayor o igual a tres, se halla la distancia entre los centroides de cada objeto, en píxeles, para calcular la distancia a la que se encuentra la marca del robot y mostrarla en pantalla en tiempo real.

Si la distancia desde el robot hasta el objeto es mayor a 1.3m el robot avanzará hasta que el objeto se encuentre dentro de un radio de 1m.

Si el objeto no es detectado luego del procedimiento de centrado, se procede a realizar el algoritmo de búsqueda hasta encontrar el objeto de interés. El diagrama de flujo de este programa se encuentra en el **Anexo 3**

El objetivo en el segundo modo de operación del sistema propuesto, “Bird Catch”, es el de detectar e identificar la cantidad de aves que se encuentran en una imagen del cielo. Para ello se utiliza el método de umbralización con un valor umbral preestablecido de 0.3 ya que este dio un mejor resultado[78] en los ensayos. Luego, se aplican los filtros morfológicos de cierre, relleno de regiones y eliminación de bordes. Finalmente, se representan las imágenes señalizando el centroide de los objetos detectados. En el **Anexo 4** se muestra el diagrama de flujo para este modo de operación.

La lógica del programa es como sigue:

Figura 4.5 Diagrama de bloques para el modo “Bird Catch”

Interfaz visual

[78] Con el valor umbral de 0.3 se segmentan mayor cantidad de aves y se descartan objetos como las nubes.

Se tiene una pantalla principal, mostrada en la **Figura 4.6**, en la cual se puede escoger entre los dos modos de operación del sistema mediante el botón “Modos”.

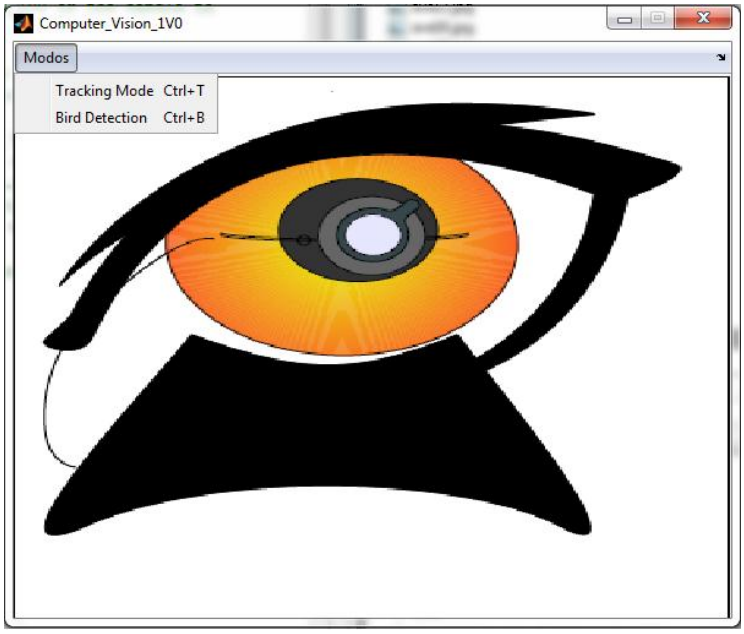


Figura 4.6 Ventana principal

Al escoger “Tracking Mode” o presionar “Ctrl + T” aparece la ventana de la **Figura 4.7**

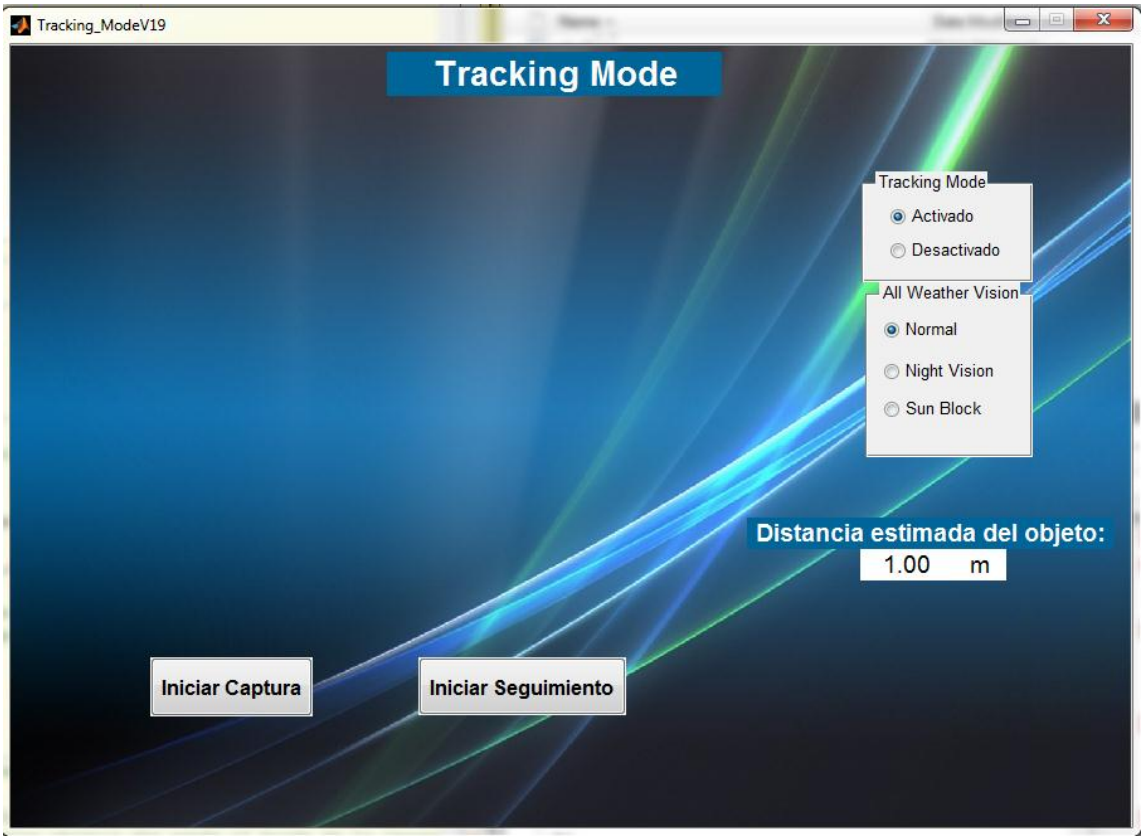


Figura 4.7 Ventana “Tracking Mode”

Al presionar el botón “Iniciar Captura” el usuario puede escoger los colores de la marca a seguir en una ventana aparte. Luego, se da inicio al seguimiento mediante el botón “Iniciar Seguimiento” y el video captado por la cámara se visualiza en la pantalla con la marca identificada y la distancia estimada del objeto en metros.

Cuando el usuario lo desee puede activar o desactivar el seguimiento de la marca seleccionando las opciones del panel “Tracking Mode”. Asimismo, puede

ajustar la luminancia del video mostrado seleccionando las opciones del panel “All Weather Vision” a gusto del usuario.

Al escoger “Bird Catch” o presionar “Ctrl + B” aparece la siguiente ventana

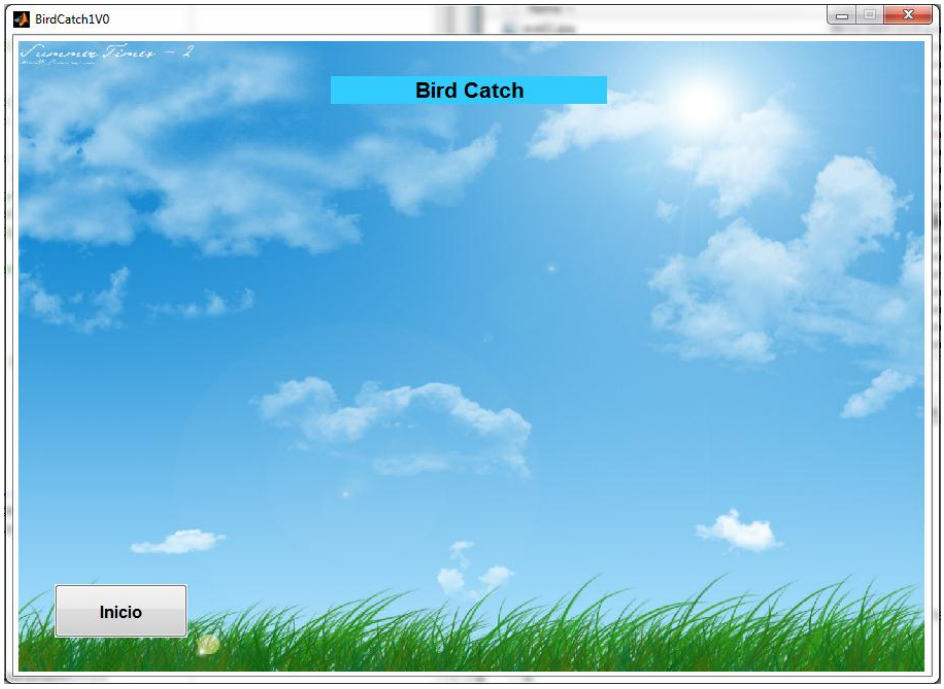


Figura 4.8 Ventana “Bird Catch”

Al presionar “Inicio” aparece el video con los objetos detectados señalizados.

Instalación

Requisitos del sistema:

- Windows XP o superior
- Tarjeta de video NVIDIA con tecnología CUDA
- Matlab R2010b o superior

Pasos a seguir:

- Descomprimir el archivo “Sistema de seguimiento.rar”
- Inicializar Matlab
- Ejecutar el archivo Computer_Vision.m

CAPÍTULO 5. PRUEBAS, RESULTADOS Y VALIDACIÓN

Pruebas y análisis de resultados

Para poder identificar aquellas funciones que se deben trasladar a la tarjeta de video, se realizaron programas de “autotest” en donde se midió el tiempo de ejecución de cada una de las funciones utilizadas. Los resultados se muestran en la **Figura 5.1**

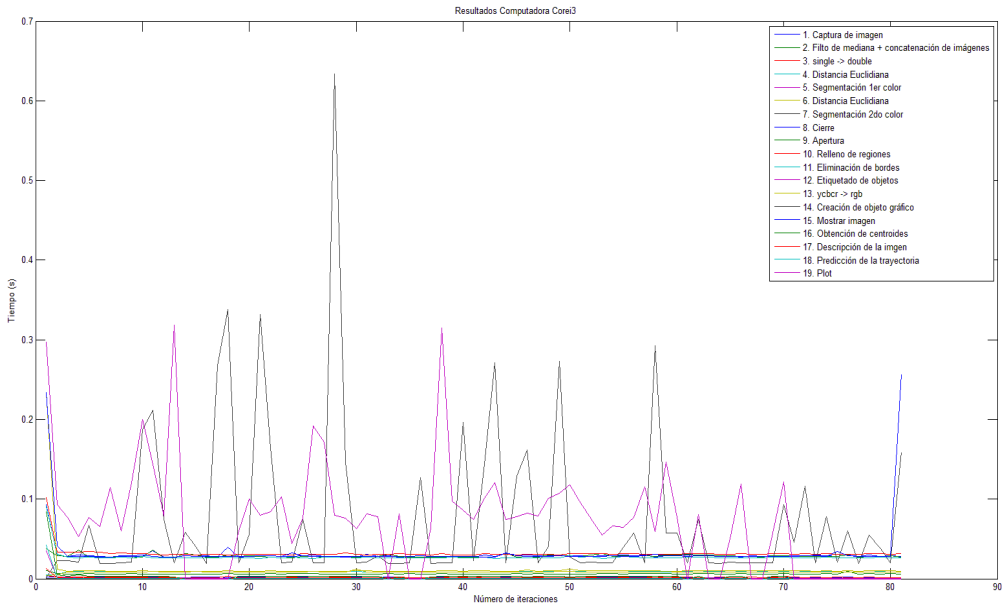


Figura 5.1 Tiempos en PC Core i3 por iteración

En la primera figura se muestra el tiempo que tarda el programa en ejecutar cada una de las funciones en cada iteración. A continuación en la **Figura 5.2** se muestra el tiempo de ejecución cada función.

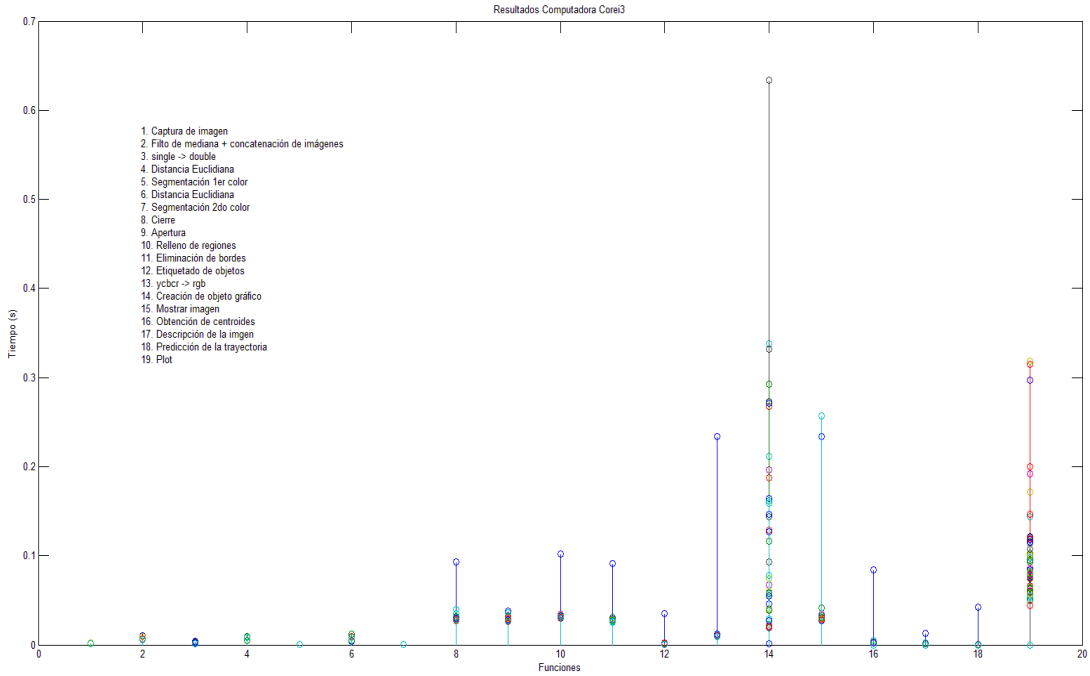


Figura 5.2 Tiempos PC Intel i3 por función

Se observa que la función que más tiempo consumía es la “Creación de objeto gráfico”, por lo que se realizó el siguiente cambio en la programación de tal manera de optimizar el código:

axes(handles.axesx) por set(gcf,’CurrentAxes’, handles.axesx))

Además:

plot(handles.axesx,x,y)

Con estos cambios se obtuvieron los siguientes resultados mostrados en **Figura 5.3** y **Figura 5.4**:

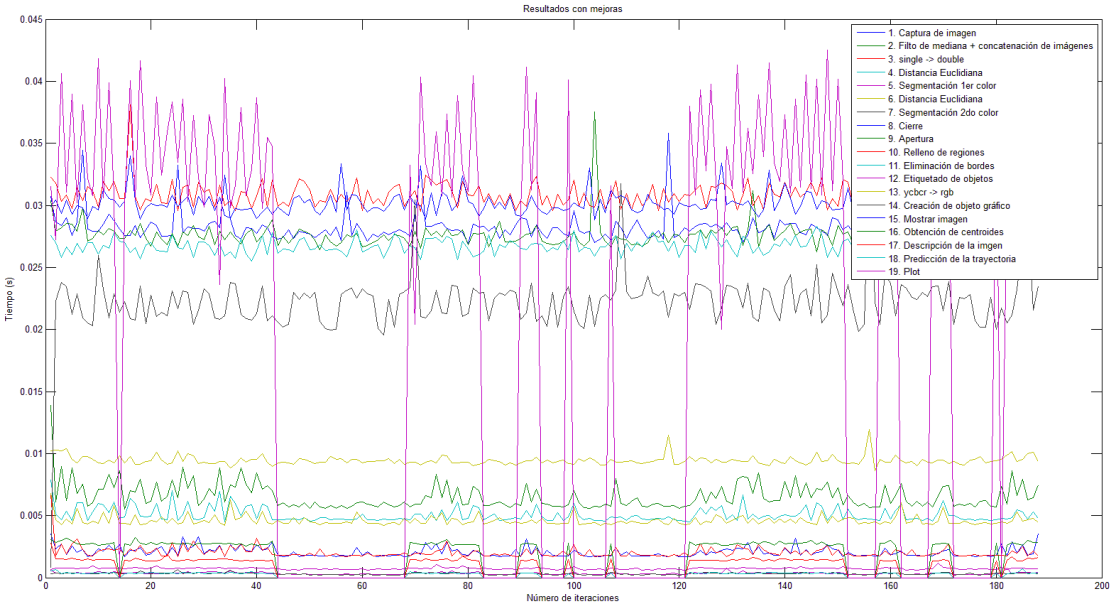


Figura 5.3 Tiempos en PC Core i3 por iteración luego de optimización de código

De la **Figura 5.4** se puede observar que, tras la optimización de código para la creación del objeto gráfico en donde se muestran las imágenes, las operaciones que demandan mayor tiempo de procesamiento son las operaciones morfológicas, por lo que se decidió que estas se procesen en la tarjeta de video.

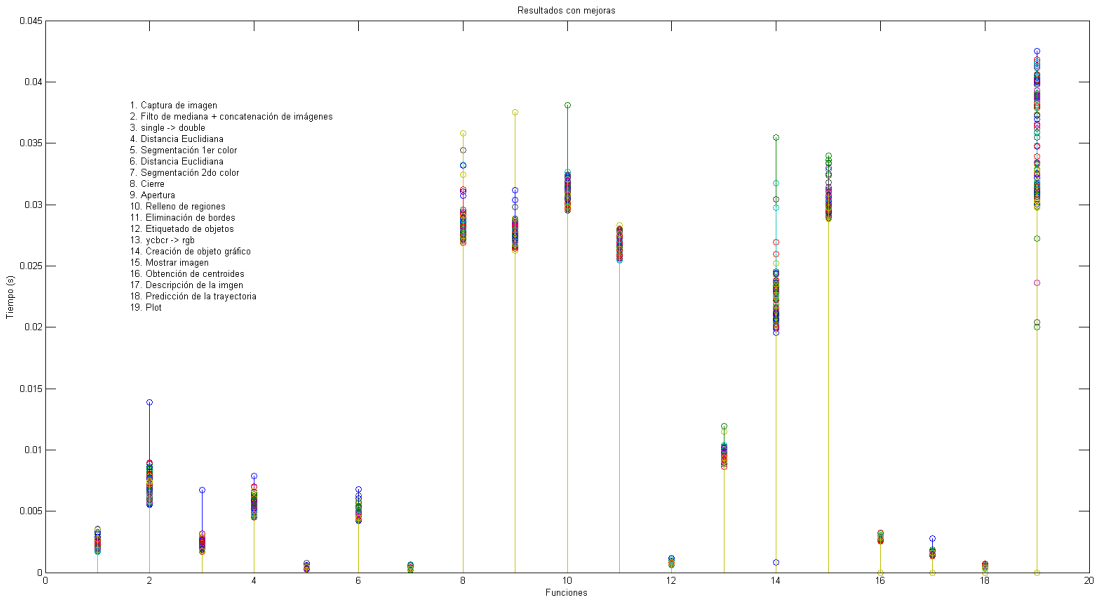


Figura 5.4 Tiempos PC Intel i3 por función luego de optimización de código

Los algoritmos de erosión y dilatación se realizaron con ayuda de las funciones MAX y MIN, además de la utilización de ventanas. Al utilizar ventanas se introduce el efecto de borde, pero debido al tamaño de nuestras imágenes, este efecto no es perjudicial.

A continuación, en las siguientes figuras (**Figura 5.5**, **Figura 5.6**, **Figura 5.7**, **Figura 5.8**, **Figura 5.9**, **Figura 5.10**, **Figura 5.11** y **Figura 5.12**) se muestran los resultados obtenidos con MATLAB y de las funciones desarrolladas en C de erosión, dilatación, apertura y cierre:

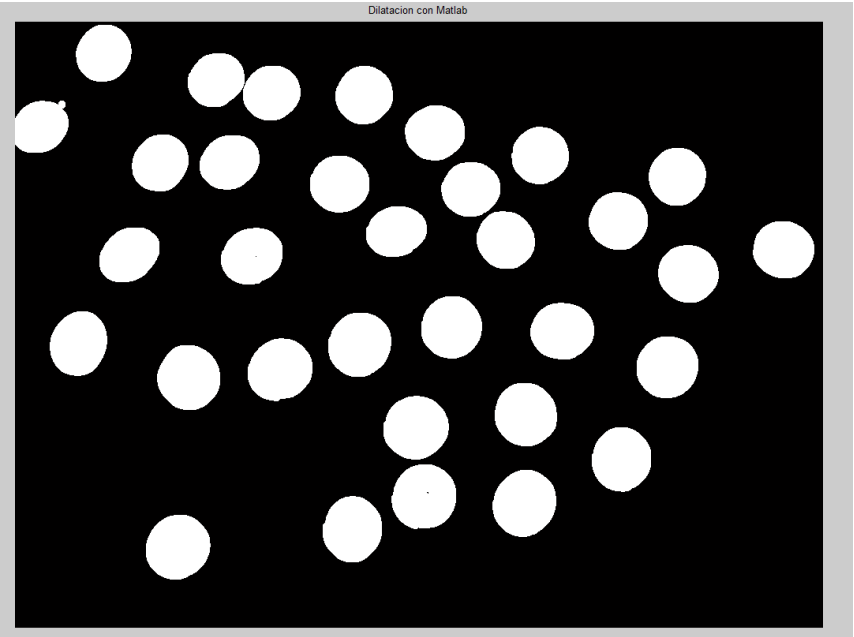


Figura 5.5 Dilatación MATLAB

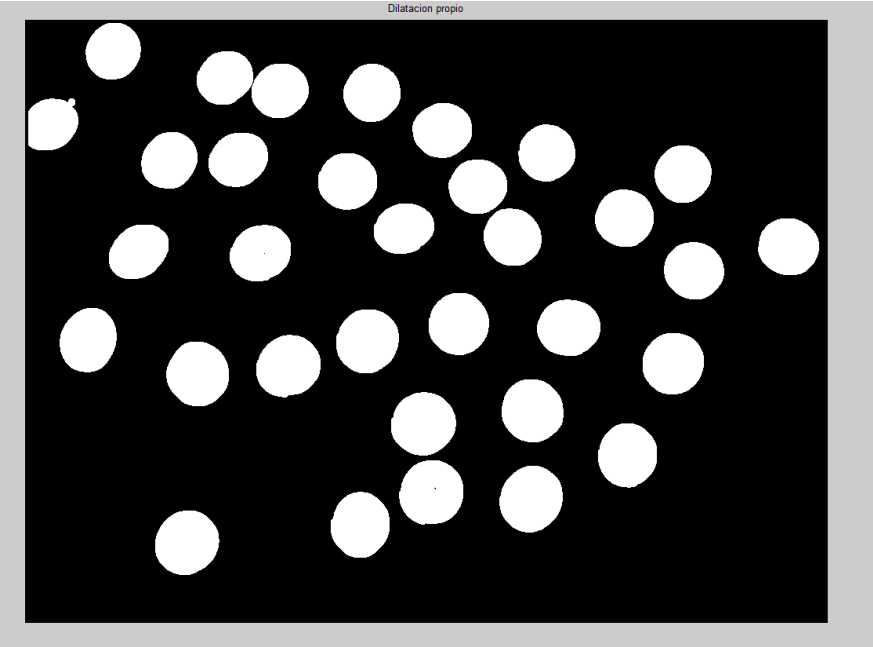


Figura 5.6 Dilatación algoritmo propio

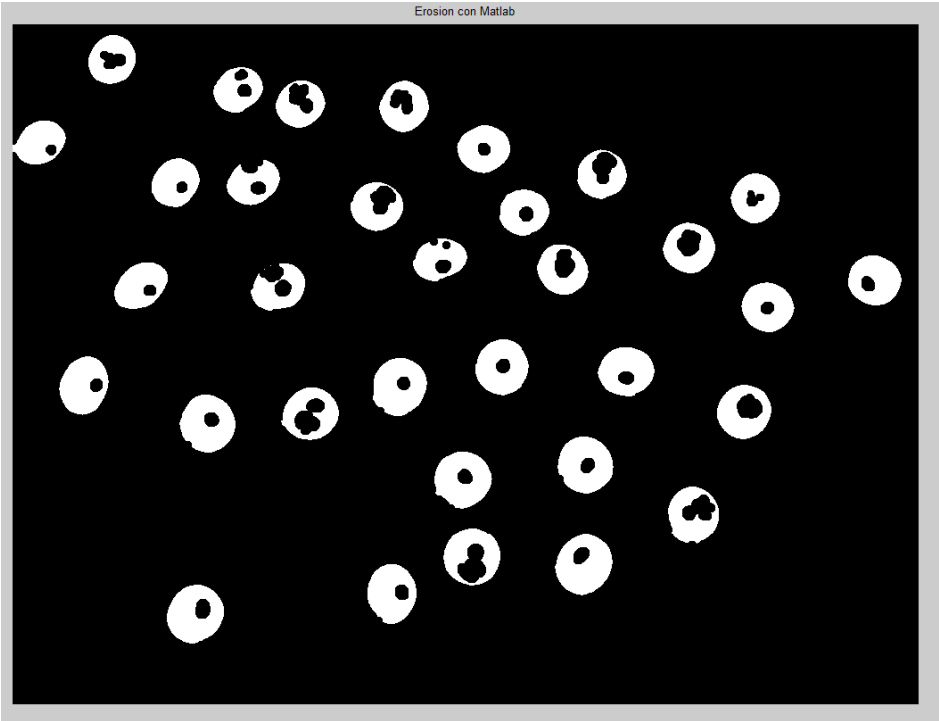


Figura 5.7 Erosión MATLAB

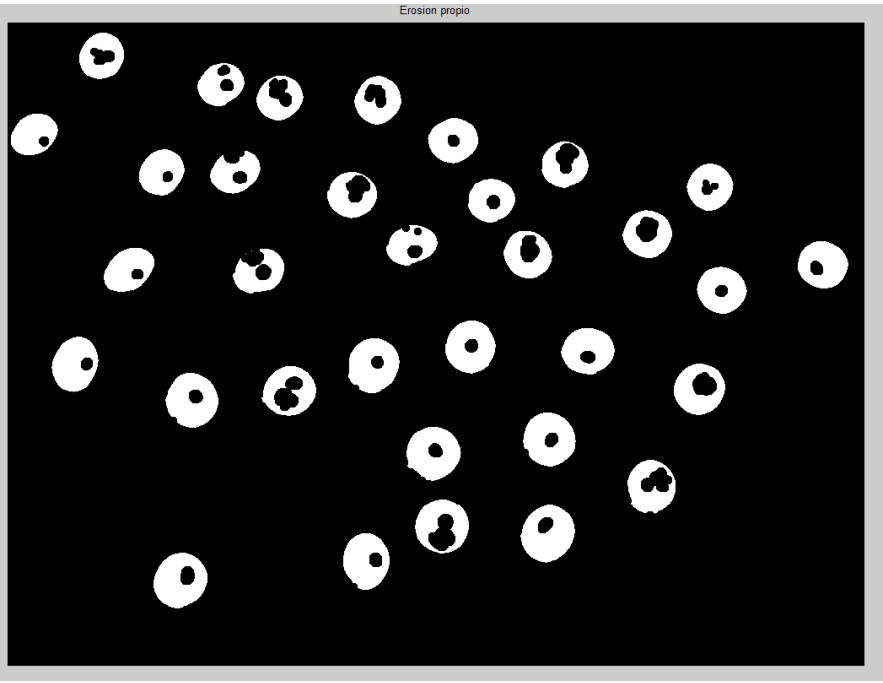


Figura 5.8 Erosión algoritmo propio

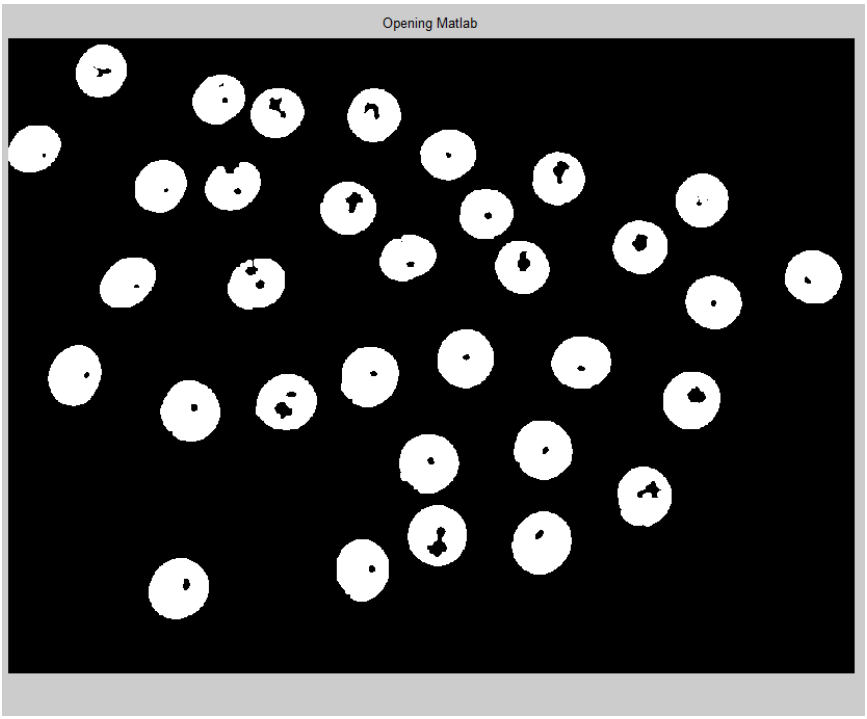


Figura 5.9 Apertura MATLAB

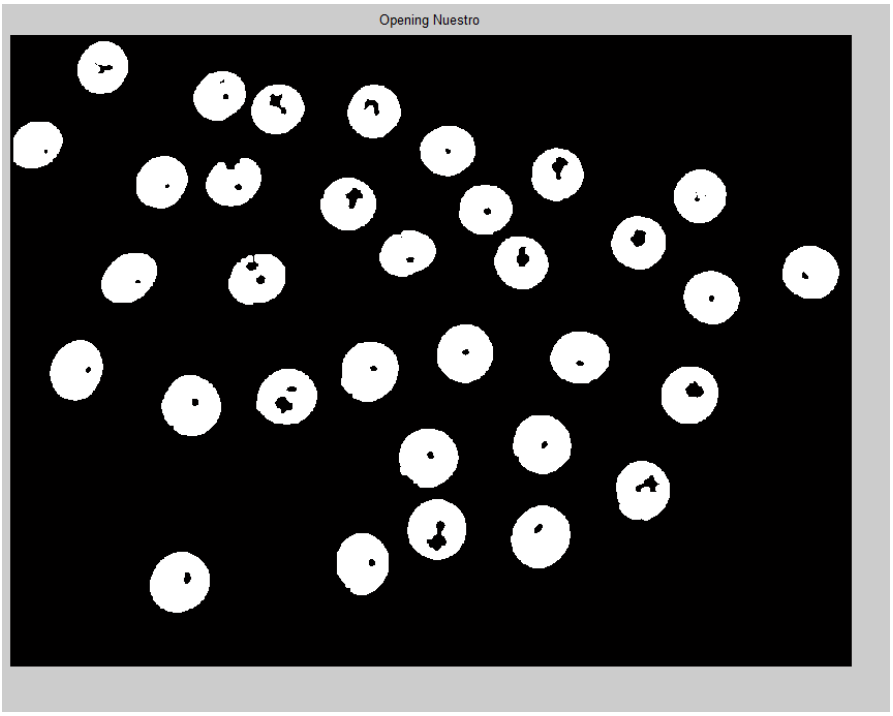


Figura 5.10 Apertura algoritmo propio

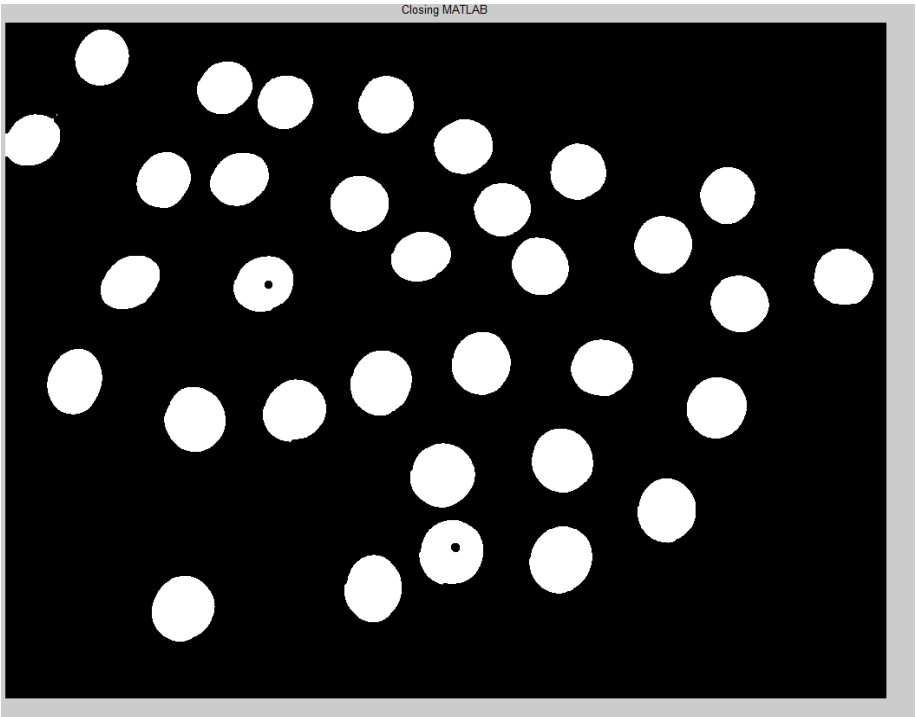


Figura 5.11 Cierre MATLAB

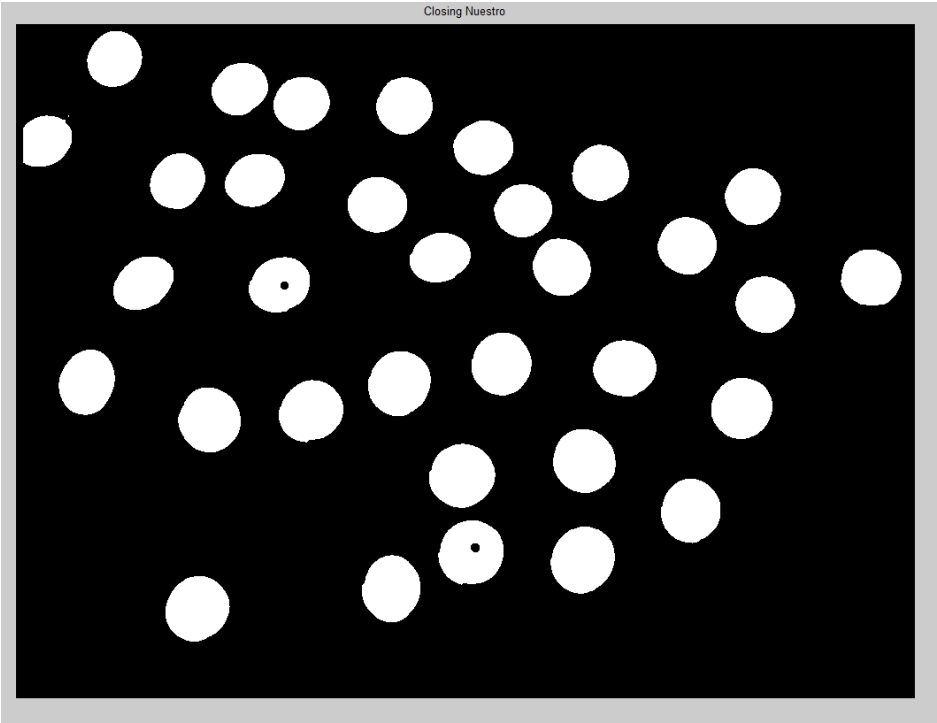


Figura 5.12 Cierre Propio

Como se puede observar los resultados logrados son idénticos. A excepción del borde introducido por el efecto explicado anteriormente.

Se presentaron algunas problemáticas al realizar estos programas:

Al realizar los programas para la dilatación y erosión en el lenguaje de programación C, se observó que el entorno de MATLAB y C operan los índices de las matrices de manera distinta tal como se muestra en la **Figura 5.13**.

C				
0	1	2	3	4
5	6	7	8	9
10	11	12	13	14

MATLAB				
1	4	7	10	13
2	5	8	11	14
3	6	9	12	15

Figura 5.13 Índices de las matrices

Luego, se definió que los parámetros a ingresar en nuestras funciones serían: la imagen a tratar transpuesta, y la estructura.

Se realizó un pequeño programa en MATLAB para comprar los tiempos que demora el procesamiento de las funciones propias de dilatación y erosión con las funciones “imdilate” e “imerode” de MATLAB, siendo los resultados los mostrados en la **Tabla 5.1**:

	Erosión (s)	**Dilatación (s)**
MATLAB	0.0593	0.1364
Algoritmo Propio	0.1423	0.1376

Tabla 5.1 Tiempos de procesamiento MATLAB vs. CPU

Para el cálculo de la mejora del rendimiento se calcula la variación porcentual de los tiempos de ejecución de los algoritmos utilizando la siguiente ecuación:

$$\Delta\% = \frac{v_f - v_i}{v_i} \qquad \textbf{(5.1)}$$

De donde se obtiene que el algoritmo implementado en C para el caso de dilatación demora 0.8791% más, mientras que en el caso de erosión se demora 139.9% más.

A partir de estos programas, se realizó el programa para que la dilatación se realice en la tarjeta de video. El programa resultante se llamó nvdilatacion.cu.

Finalmente se realizaron pruebas para verificar el comportamiento de la función de dilatación realizada en la tarjeta de video, y se observó una mejora en el tiempo para imágenes pequeñas (100x100 pixeles).

	Dilatación (s)
MATLAB	0.0828
Algoritmo Propio	0.0702

Tabla 5.2 Tiempos de procesamiento MATLAB vs. GPU

En la **Tabla 5.2** se observa que el tiempo de procesamiento realizado en la tarjeta de video tiene una mejora de tiempo de 15.2%, lo que implicaría que el tiempo al procesar imágenes más grandes también se reduciría. Sin embargo, se tuvieron problemas con el manejo de los espacios de memoria dentro de la tarjeta de video, ya que al cargar imágenes grandes (352x240 pixeles) a la tarjeta de video para que realice la dilatación, se presentaba un error grave y el controlador de la tarjeta dejaba de funcionar.

Por lo tanto, se descartó la utilización de los algoritmos propios para los filtros morfológicos.

Después se implementaron las operaciones matemáticas para el cálculo del desplazamiento de la marca en el GPU esperando tener resultados satisfactorios para la aceleración de estos. Sin embargo al tratarse de operaciones con

escalares la eficiencia de la tarjeta de video no era la esperada, tal como se muestra en la **Figura 5.14**, por lo que esta opción fue descartada.

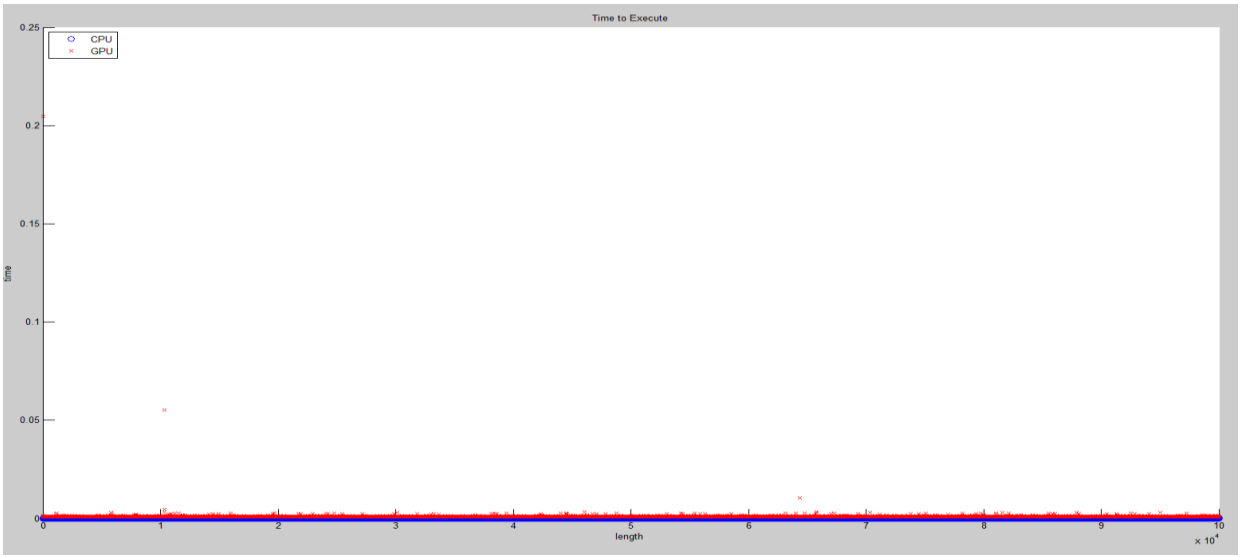


Figura 5.14 Tiempos de procesamiento – Operaciones Matemáticas MATLAB vs. GPU

Como se puede ver en la **Figura 5.14**, las operaciones realizadas en el GPU (en color rojo) tomaron más tiempo que las operaciones realizadas en el CPU (color azul).

De la misma manera el algoritmo para la corrección Gamma fue implementado para que se ejecute en el GPU. La **Figura 5.15** muestra una comparación de tiempos de ejecución de la función en el GPU (rojo) vs. el CPU (azul).

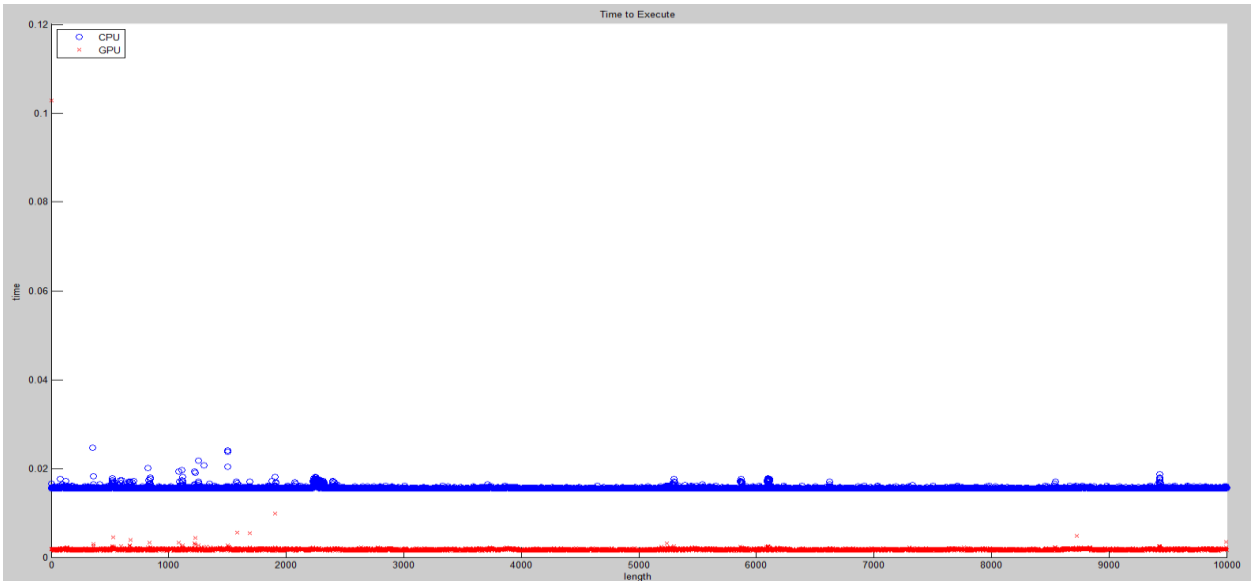


Figura 5.15 Tiempos de procesamiento – Corrección Gamma MATLAB vs. GPU

Como se puede apreciar, para la corrección Gamma en todos los casos el tiempo de ejecución para el GPU fue menor que el utilizado por el CPU. Obteniéndose una mejora promedio de 87.15%. Esto demuestra que el uso de la tarjeta de video con procesamiento paralelo es eficiente para la operación con matrices más no para escalares.

La segmentación de imágenes por color utilizando la distancia euclidiana como medida de similitud también era factible implementarlo en la tarjeta de video ya que se tenía que evaluar las matrices de “Cb” y “Cr” en su totalidad. Los resultados de la comparación entre el tiempo de ejecución del GPU (rojo) vs. el tiempo de ejecución del CPU (azul) se muestran en la **Figura 5.16**

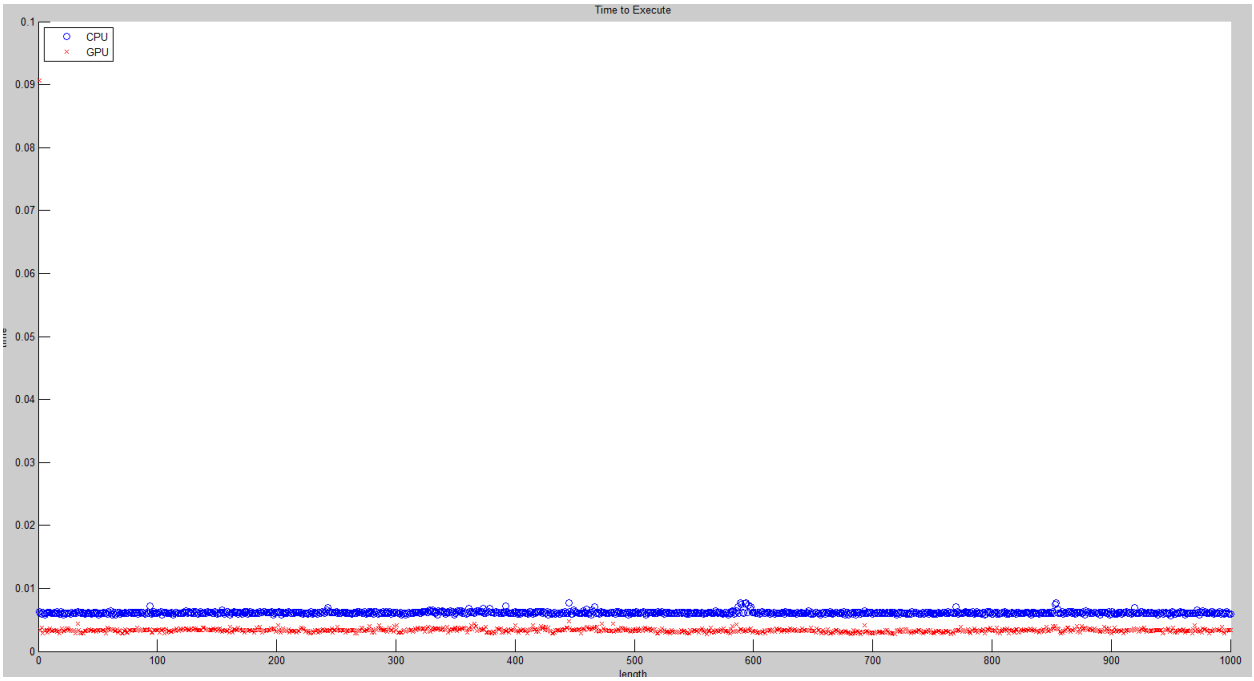


Figura 5.16 Tiempos de procesamiento – Distancia Euclidiana MATLAB vs. GPU

De la misma manera que en el caso anterior el GPU mostró un rendimiento superior que el CPU al disminuir el tiempo de procesamiento en un 59.31%. Nuevamente se demuestra que el GPU tiene mejor desempeño cuando se opera con matrices.

Para hallar la distancia a la que se encuentra la marca se calcula la distancia en píxeles entre los tres primeros centroides que se detectan. Entonces se pueden dar tres casos como se muestra en la **Figura 5.17**

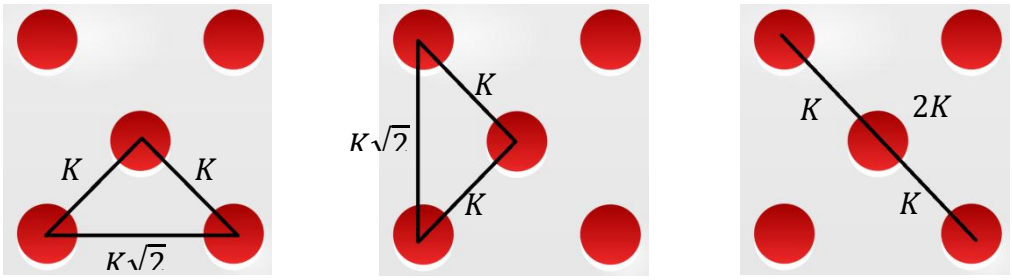


Figura 5.17 Casos de detección de tres objetos

Estos casos, como se observa en la figura presentan relaciones fácilmente reconocibles gracias a la forma del patrón escogido, lo que facilita el cálculo de distancias entre centroides.

Se realizaron pruebas a 0.5 y 1 metros para obtener la máxima separación entre los centroides de los objetos detectados y para cada caso planteado se obtuvo:

	CASO 1 / CASO 2 (pixeles)	**CASO 3 (pixeles)**
0.5 m	78.5	114.45
1 m	40.4	57.8

Tabla 5.3 Distancia en píxeles entre los centroides

Dados los resultados de la **Tabla 5.3**, se decide trabajar con los valores correspondientes a 1 m y la distancia del robot a la marca se obtiene calculando la proporción entre cada uno de estos y la máxima separación entre los centroides.

Se realizaron pruebas con una marca con iluminación propia tal como se muestra en la **Figura 5.18**. Sin embargo, en las imágenes capturadas por la cámara el punto central se satura de brillo perdiendo el color (verde) y lo que realmente se segmenta son los reflejos de la luz a los alrededores. Esto último deforma el patrón segmentado impidiendo calcular la distancia del objeto al robot y reduciendo la precisión del sistema.

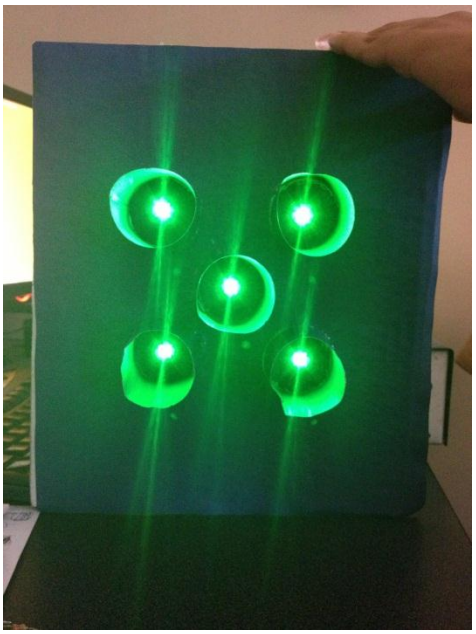


Figura 5.18 Marca con luz propia

Para las pruebas de validación del producto final se define el escenario de pruebas como un ambiente cerrado con luz artificial blanca y de preferencia sin ingreso de luz natural. En caso se tenga ingreso de luz natural (ventanas, tragaluces, etc) probar el sistema con la luz natural con incidencia directa ya que a contraluz no opera correctamente. Además, el suelo debe ser plano, sin

desniveles y presentar rugosidad para mejor tracción de las ruedas. El volumen de trabajo del robot es de 360° con un una altura máxima de 2.1 m. En la **Figura 5.19** se muestra el escenario utilizado para las pruebas bajo distintos niveles de iluminación.

Por otro lado, para poder determinar el rango de luminancia en el cual el sistema tiene un mejor desempeño se define la función de densidad de probabilidad de la luminancia. Es decir se analizan los valores de luminancia en aquellas imágenes donde se segmentó correctamente la marca para poder identificar los valores más probables y el rango de luminancia de trabajo del sistema, un ejemplo de una imagen segmentada se muestra en la **Figura 5.20**

De una población de 500 imágenes muestra, se obtiene la función de densidad de luminancia que se muestra en la **Figura 5.21**.

Figura 5.19 Escenario de pruebas

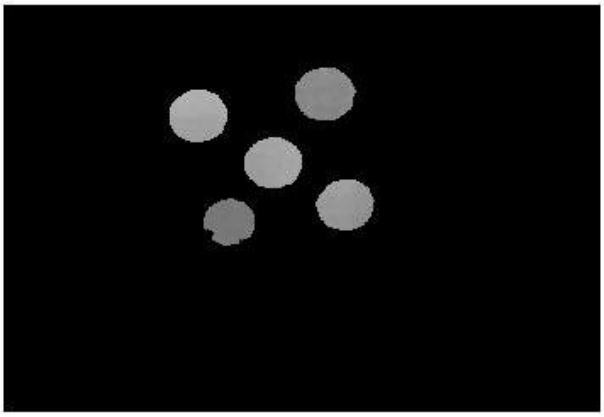


Figura 5.20 Imagen muestra - Valores de luminancia en imagen segmentada

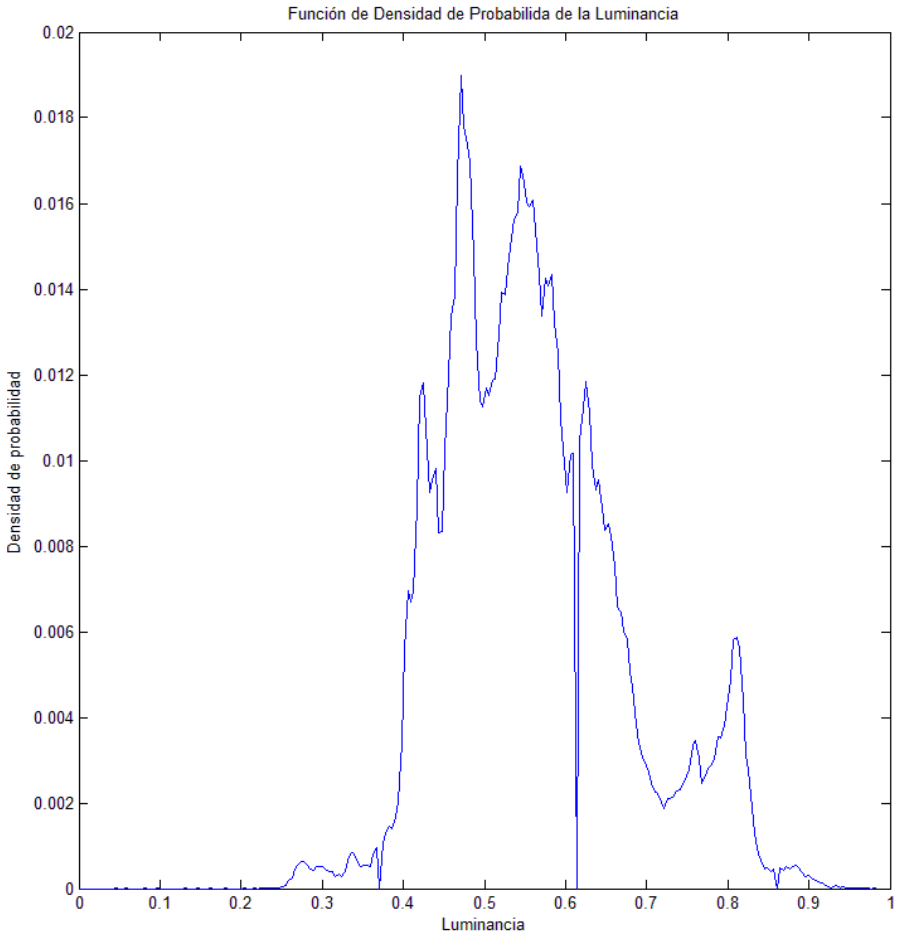


Figura 5.21 Función de densidad de probabilidad de luminancia

A partir de la función de densidad de probabilidad de luminancia obtenida se puede determinar que el rango óptimo de luminancia para el funcionamiento del sistema es de 0.4 a 0.7. Los valores fuera de este rango no aseguran la detección de la marca, sin embargo no es determinante ya que alrededor de 0.8 se tiene una buena posibilidad de detectar la marca.

De la misma manera, se halló la función de densidad de probabilidad del error del sistema de seguimiento. Para esto se tienen dos posibles escenarios:

- Que el sistema siga un objeto cuando no debe seguirlo
- Que el sistema no siga al objeto cuando no debe seguirlo

En el primer caso, la probabilidad de ocurrencia de este evento es nula ya que el sistema realiza el seguimiento obteniendo la distancia euclidiana entre los valores de Cb y Cr de una imagen por lo tanto, si el sistema realiza el seguimiento se debe a que el color preseleccionado ha sido detectado.

Para el segundo caso, se realizaron pruebas para determinar cuál es el valor umbral máximo en el que el sistema segmenta la marca correctamente dando como resultado el valor de 0.17. En el cual se logra una mejor segmentación, pero con una mayor incidencia de error.

Figura 5.22 Imagen original

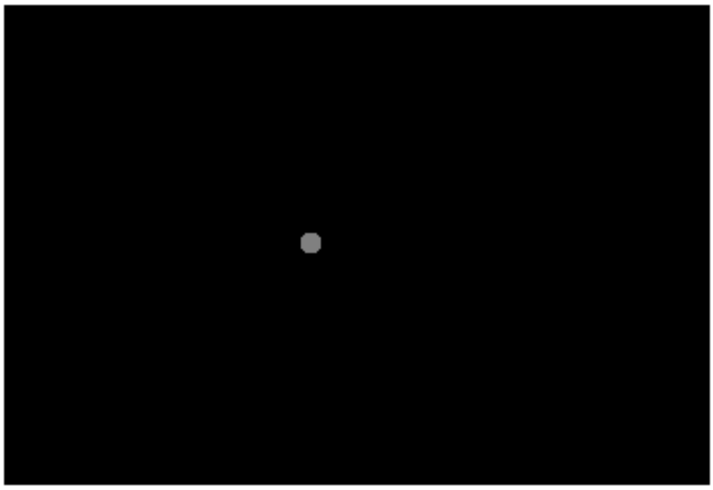


Figura 5.23 Imagen segmentada – Umbral 0.085

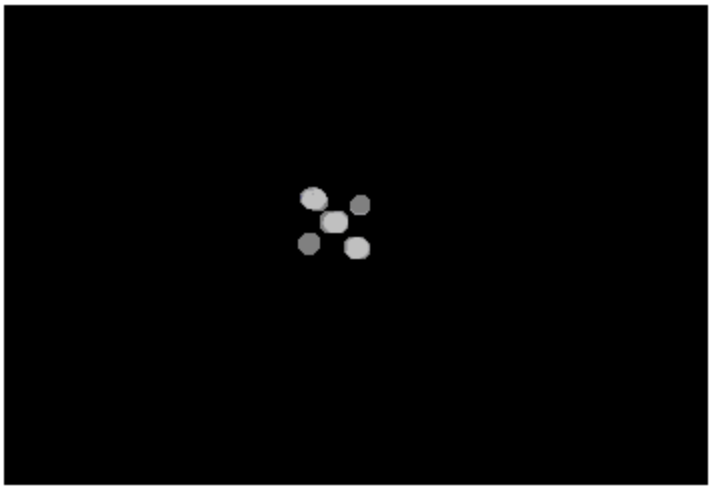


Figura 5.24 Imagen segmentada – Umbral 0.17

Como se puede apreciar en la **Figura 5.24** a pesar de segmentar la marca completa se segmentan también porciones que no pertenecen a la marca (bordes deformados).

Por último la función de densidad de probabilidad de error del sistema de seguimiento es la que muestra en la **Figura 5.25**

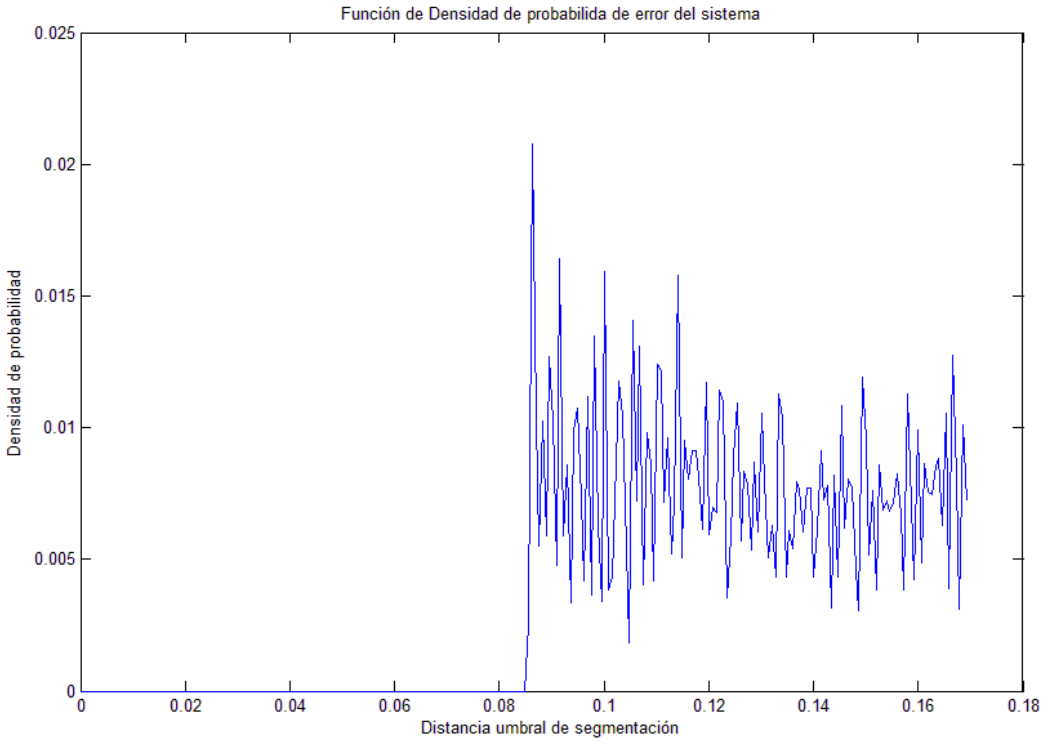


Figura 5.25 Función de densidad de probabilidad de error del sistema

En la **Figura 5.25** se puede apreciar que la tendencia no es creciente ni decreciente conforme aumenta el valor de la distancia umbral, es decir el error es casi uniforme.

Por último, para validar el cálculo de distancia del objeto al robot se realizaron comparaciones entre la medida calcula por el sistema y la medida real, obteniéndose un error promedio de 4.8%, mediante la ecuación **5.1**.

Resumen económico

A continuación, en la **Tabla 5.4**, se muestra el resumen económico del prototipo. Sin embargo no incluye el costo del computador y el software necesario (MATLAB) ya que estos serían asumidos por el comprador.

Cámara inalámbrica 2.4 GHz 8VDC	S/. 260.00
Capturador de video RCA – USB	S/. 260.00
Tarjeta de video ELITEGROUP Nvidia GTX465	S/. 870.00
Circuito impreso (Tarjeta de control)	S/. 300.00
Circuito impreso (Tarjeta Puente H 2A)	S/. 100.00
Componentes electrónicos	S/. 200.00
Servomotor Savox	S/. 120.00
Motores DC 12V	S/. 100.00
Baterías	S/.80.00
Estructura mecánica	S/. 300.00
Consumibles (Pernos, tornillos, clavos, cables, fusibles, pintura, varillas de aluminio, etc)	S/. 200.00
Costo Total	**S/. 2790.00**
Precio de venta de Prototipo	**S/. 15 000.00**[79]

Tabla 5.4 Resumen económico

Esta tabla representa los gastos asumidos por ambos integrantes a lo largo del proyecto. La adquisición de la cámara inalámbrica y el capturador de video se realizaron en la cadena de tiendas RadioShack, la tarjeta de video se consiguió

[79] Precio tentativo

con los proveedores de Deltron, mientras que los componentes electrónicos, los motores, baterías y el servomotor se adquirieron en tiendas de retail de componentes electrónicos.

Por último, los consumibles se obtuvieron en la cadena de tiendas Ace Home Center y los circuitos impresos se mandaron a fabricar en Circuitos Impresos S.A.

El precio de la **Tabla 5.4** se obtuvo con la suma de los valores de cada uno de los objetos adquiridos y agregando el costo que se vio conveniente colocar al trabajo realizado por ambos integrantes durante todo el tiempo de realización del proyecto, como la programación total del programa, el diseño de las tarjetas y la elaboración del robot.

CONCLUSIONES

- Se logró desarrollar un sistema de seguimiento de objetos mediante procesamiento digital de imágenes aplicado al control de un robot móvil con ruedas con dos grados de libertad optimizando algoritmos utilizando tecnología de procesamiento paralelo.

- Se logró controlar el movimiento del robot de tal manera que se tenga un movimiento suave y agradable a la vista.

- Se logró desarrollar un sistema de seguimiento con algoritmos de respaldo que permitan un seguimiento continuo del objeto a pesar de que por momentos se pierda la imagen o se tenga altos niveles de ruido.

- De los resultados obtenidos se puede concluir que, dada la sensibilidad del sistema a las variaciones de luminancia, sólo se podría aplicar a vigilancia o supervisión lugares cerrados con iluminación uniforme como almacenes, laboratorios, coliseos cerrados, etc.

- Los módulos de comunicación configurables son de gran ayuda ya que permiten cambiar el canal por el que se envían y reciben los datos. Esto ayudó a solucionar el problema de interferencia entre la cámara

inalámbrica y el módulo XBee-PRO ya que ambos utilizan la banda ICM 2.4 GHz.

- Para lograr un buen diseño electrónico en circuito impreso es necesario pasar por más de un prototipo ya que al implementar y probar las tarjetas se descuben fallas o limitaciones en el diseño.

- Utilización de tecnología que aún está en desarrollo a nivel mundial tiene ciertos beneficios como la gran cantidad de códigos abiertos, permite la innovación, pero también requiere de mucha investigación y no es fácil de utilizar.

- La utilización de una tarjeta de video GPU, acelera el procesamiento de imágenes, pero siempre y cuando este sea utilizado de la manera correcta, de otro modo el procesamiento puede resultar incluso más lento que antes.

- A pesar de que la distancia euclidiana se halla utilizando una ecuación no lineal, utilizando procesamiento paralelo la carga computacional se reduce notablemente y no afecta el desempeño del programa principal.

- Utilizar una marca con un patrón pre definido facilita la estimación de la distancia de esta a la cámara, ya que se puede sacar una relación entre píxeles y distancia entre la marca y la cámara.

- El sistema es capaz de calcular la distancia entre la marca y la cámara con un error no mayor al 5%.

- A pesar de tener un software robusto, uno de los principales problemas para el reconocimiento de la marca patrón son las variaciones en la luminosidad del entorno.

- Al utilizar una marca con iluminación propia, no se logró ninguna mejora en el desempeño del sistema ya que se perdía la forma del patrón en la imagen segmentada.

- Con la ayuda de la función de densidad de probabilidad de los valores de luminancia se puedo determinar el rango de luminancia en el cual el sistema opera correctamente. Este rango corresponde de 0.4 a 0.7 en el modelo YCbCr.

- Con la función de densidad de probabilidad de error del sistema se pudo determinar que la posibilidad de que el sistema siga un objeto cuando no debe, es nula, y la posibilidad de que el sistema no siga el objeto cuando debe, es casi uniforme.

- Una de las limitaciones del proyecto es la calidad de las imágenes ya que al ser enviadas de manera inalámbrica utilizando la frecuencia ICM 2.4 GHz presenta ruido además y son de baja resolución (352x240). Sin embargo, al ser imágenes pequeñas se pueden procesar más rápidamente lo que aumenta el desempeño del sistema y permite hacer el seguimiento en tiempo real. Además, el ruido no fue mayor problema ya que fue solucionado aplicando el filtro de mediana a las imágenes.

- Sobredimensionar la capacidad de las baterías ofrece mayor tiempo de autonomía del sistema, tiempo necesario para realizar pruebas necesarias en el desarrollo del proyecto, sin embargo aumenta significativamente el peso y el costo del prototipo.

- El diseño de un robot sencillo de dos grados de libertad, de poco peso y por ende inercia simplifica el control del sistema y permite mayor precisión.

BIBLIOGRAFÍA

ARNÁEZ, Enrique (2009) Enfoque práctico de control moderno

APPIN KNOWLEDGE SOLUTIONS (AKS) (2007) Robotics. Hingham: Infinity Science Press LLC.

BALCELLS, Josep y ROMERAL, José Luis (1997) Autómatas programables. Barcelona: Marcombo S.A.

COLEMAN, David y WESTCOTT, David (2012) Certified Wireless Network Administrator (CWNA) Official Study Guide 3ra ed. Indianapolis: John Wiley & Sons, Inc.

CHOONG, Leslie (2009) Multi-Channel IEEE 802.14.4 Packet Capture Using Software Defined Radio. Los Angeles: University of California

DEVORE, Jay (2008) Probabilidad y Estadística para ingeniería y ciencias. 7ma ed. México DF: Cengage Learning

DIAZ, T y otros (2013) Seguimiento de objetos en GPU basado en el Filtro de Partículas (http://congresomaeb2012.uclm.es/papers/paper_31.pdf) (Consulta: 17 de Octubre del 2012)

DIGI INTERNATIONAL (2008a) X-CTU Configuration & Test Utility Software. Manual de usuario de software X-CTU Minnentonka: Digi International, Inc.

DIGI INTERNATIONAL (2008b) Wireless Mesh Networking ZigBee vs. DigiMesh (http://www.digi.com/pdf/wp_zigbeevsdigimesh.pdf) (Consulta: 2 de Agosto del 2012)

DIGI INTERNATIONAL (2010) XBee/XBee-PRO Digimesh 2.4 RF Modules. Minnentonka: Digi International, Inc.

DIGI INTERNATIONAL (2012) The DigiMesh Networking Protocol (http://www.digi.com/technology/digimesh/) (consulta: 3 de agosto del 2012)

ESPINOZA, Nidia y otros (2012) E-learning : Cámara IP autónoma. San Salvador: Universidad Centroamericana “José Simeón Cañas”.

FOURCC (2012) Video Codecs and Pixel Formats: YUV Formats (http://www.fourcc.org/yuv.php) (Consulta: 9 Noviembre del 2010)

GONZÁLES, Rafael y WOODS, Richard (1996) Tratamiento digital de imágenes 1°.ed. Delaware: Addison-Wesley Iberoamericana,S.A.

GONZÁLES, Rafael y WOODS, Richard (2002) Digital Image Processing 2°.ed. New Jersey: Prentice-Hall, Inc.

GONZALES, Rafael y WOODS, Richard (2004) Digital Image Processing Using MATLAB. New Jersey: Prentice-Hall, Inc.

GHOSHAL, Subrata (2010) 8051 Microcontroller Internals, instructions, programming and interfacing India: Pearson Education

GRUPO DE PROCESAMIENTO DIGITAL DE SEÑALES E IMÁGENES DE LA PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ (GPDSI) (2010) (http://www.pucp.edu.pe/grupo/gpdsi/proyectos.php?id=39) Sistema de optimización de flujo vehicular en una intersección. (Consulta: 1 de Abril del 2010)

INICTEL (2009) Programación y Aplicaciones de Microcontroladores PIC 16 Lima: INICTEL.

INTERSIL (2010) Portal del fabricante mundial INTERSIL. YCbCr to RGB considerations.(http://www.intersil.com/data/an/an9717.pdf) (Consulta: 21 de Junio del 2010)

IR (2012) HEXFET Power MOSFET (http://www.datasheetcatalog.org/datasheet/irf/irfz44.pdf) (Consulta: 09 de Enero del 2012)

JAIN, Anil (1989) Fundamentals of Digital Image Processing. Upper Saddle River: Prentice Hall.

LINDBLOOM,Bruce (2009) Useful Color Information, Studies and Files (http://www.brucelindbloom.com/) (Consulta: 27 de Abril del 2011)

MATHWORKS (2010) Accelerating MATLAB CODE Using GPUs. Natick: The MathWorks, Inc.

MORALES, Omar (2003) Interface gráfica para el medidor de nivel (Tesis professional para obtener el título en Licenciatura en Eléctronica y Comunicaciones) Puebla: Universidad de las Amércias Puebla

NATIONAL INSTRUMENTS (2006) Comunicación Serial: Conceptos Generales (http://digital.ni.com/public.nsf/allkb/039001258CEF8FB686256E0F005888D1) (Consulta: 2 de Agosto de 2012)

NATIONAL SEMICONDUCTOR (2011) 3-Terminal Adjustable Regulator. Santa Clara: National Semiconductor Corporation

NATIONMASTER (2010) 576i50: Artículo sobre formatos de video. Página web de enciclopedia virtual. (Consulta: 14 de Julio del 2010) (http://www.statemaster.com/encyclopedia/576i50)

NVIDIA Corporation (2007) NVIDIA CUDA Compute Unified Device Architecture Programming Guide. Santa Clara: NVIDIA Corporation

NVIDIA Corporation (2010a) What is CUDA? Portal web que detalla las características de la Tecnología CUDA (http://www.nvidia.com/object/what_is_cuda_new.html) (Consulta: 22 de Setiembre del 2010)

NVIDIA Corporation (2010b) The CUDA Compiler Driver NVCC, Santa Clara CA: NVIDIA Corporation

NVIDIA Corporation (2012a) GeForce GTX 465. (http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-465/specifications) (Consulta: 3 de Agosto del 2012)

NVIDIA Corporation (2012b) GeForce GTX 690. (http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-690/performance) (Consulta: 3 de Agosto del 2012)

Ollero, Anibal (2001) Robótica: Manipuladores y robots móviles. Barcelona: Marcombo S.A.

PARROT (2010) (http://ardrone.parrot.com/parrot-ar-drone/videos/#player) Portal del Quadrotor helicopter AR.Drone de Parrot. (Consulta: 1 de Abril)

PEARDON, Mike(2009) A beginner's guide to programming GPUs with CUDA, Dublin: School of Mathematics Trinity College Dublin

PERTUSA, Jose (2003) Técnicas de Análisis de imágenes Aplicaciones en Biología. Valencia: Universidad de Valencia

PLATERO, Carlos (2012) Procesamiento Morfológico. Madrid: Universidad Politécnica de Madrid

PROYECTACOLOR (2009) Modelos de color. (Consulta: 4 de Noviembre del 2010) (http://www.proyectacolor.cl/aplicacion-del-color/modelos-de-color/)

RASHID, Muhammad (2004) Electrónica de Potencia 3ra ed. México: Pearson Educación.

RINCON, José (2012) Manual de Microcontroladores PIC. Biblioteca Virtual de la FIE de la Universidad Michoacana de San Nicolás de Hidalgo (Consulta: 12 de Junio) (http://lc.fie.umich.mx/~jrincon/manual%20PICs%20Ruddy.pdf)

RODRIGUEZ, Ramón (2006) Estimación de posición y seguimiento de objetos móviles sobre imágenes panorámicas. Madrid: Universidad de Alcalá

ROJAS, TEJADA y MILLA, Luis (2006) Análisis automático de la movilidad en células usando técnicas digitales. Lima: Facultad de Ingeniería Electrónica y Eléctrica de la Universidad Nacional Mayor de San Marcos.

RUSS, John (1999) The image processing handbook. 3ra ed. North Carolina: CRC Press.

SANCHEZ, Omar (2010a) Vehículos no tripulados compilación de diapositivas de aplicaciones de la robótica. Huelva: Universidad de Huelva.

SANCHEZ, Omar (2010b) (http://omarsanchez.net/histoecu.aspx) Ecualización de Histogramas. Portal relacionado con el modelo, control y sistema de visión. (Consulta: 22 de Junio 2010)

SHARP (2005) GP2D12 Optoelectronic Device. Osaka: SHARP Corporation

STAROVOITOV, SAMAL y BRILUIK (2003) Image Enhancement for face recognition, United Institute of Informatics Problems Minsk, Belarus (http://handysolution.com/science/Image_Enhancement_for_Face_Recognition.pdf) (Consulta: 04 de Diciembre del 2010)

SUCAR, Enrique y GÓMEZ, Giovani (2003) Visión Computacional. Puebla: Instituto Nacional de Astrofísica, Óptica y Electrónica

TOSHIBA (2000) Toshiba Photocoupler GaAs IRed & Photo-Transistor (http://pdf1.alldatasheet.com/datasheet-pdf/view/32435/TOSHIBA/TLP521-1.html) (Consulta: 30 de Agosto del 2011)

Universidad Federal del Espiritu Santo (UFES) (2013) Proyectos académicos (http://www.inf.ufes.br/) Portal del departamento de informática de la UFES

UNIÓN INTERNACIONAL DE TELECOMUNICACIONES (UIT) (2007) Cuestiones de Carácter General (http://www.itu.int/ITU-R/terrestrial/faq/index-es.html) (Consulta: 24 de Julio del 2012)

UNIVERSIDAD POLITÉCNICA DE CATALUÑA (2012) Dispositivos de Electrónica de Potencia. Portal del departamento de Ingeniería Electrónica. Barcelona: UPC (http://tec.upc.es/el/TEMA-2%20EP%20(v1).pdf) (Consulta: 29 de Abril del 2012)

VALDÉZ, Fernando y PÁLLAS, Ramon (2007) Microcontroladores: Fundamentos y Aplicaciones con PIC. Barcelona: Marcombo S.A.

WIKIPEDIA (2010) (http://en.wikipedia.org/wiki/YCbCr) Articulo sobre espacio de color YCbCr. (Consulta: 21 de Junio del 2010)

WIKIPEDIA (2013) (http://es.wikipedia.org/wiki/PlayStation_3) Artículo sobre Play Station 3. (Consulta: 01 de Mayo del 2013)

ANEXOS

ANEXO 1

ANEXO 2

ANEXO 3

ANEXO 4

ANEXO 5

Programa principal

```
function varargout = Tracking_ModeV191(varargin)
% TRACKING_MODEV191 M-file for Tracking_ModeV191.fig
%         TRACKING_MODEV19, by itself, creates a new TRACKING_MODEV19 or
raises the existing
%      singleton*.
%
%      H = TRACKING_MODEV19 returns the handle to a new TRACKING_MODEV19 or
the handle to
%      the existing singleton*.
%
%      TRACKING_MODEV19('CALLBACK',hObject,eventData,handles,...) calls the
local
%       function named CALLBACK in TRACKING_MODEV19.M with the given input
arguments.
%
%               TRACKING_MODEV19('Property','Value',...)  creates  a  new
TRACKING_MODEV19 or raises the
%       existing singleton*.  Starting from the left, property value pairs
are
%       applied to the GUI before Tracking_ModeV19_OpeningFcn gets called.
An
%           unrecognized  property  name  or  invalid  value  makes  property
application
%        stop.   All  inputs  are  passed  to  Tracking_ModeV19_OpeningFcn  via
varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHAND LES

% Edit the above text to modify the response to help Tracking_ModeV19

% Last Modified by GUIDE v2.5 16-Aug-2012 20:41:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Tracking_ModeV19_OpeningFcn, ...
                   'gui_OutputFcn',  @Tracking_ModeV19_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
% End initialization code - DO NOT EDIT


% --- Executes just before Tracking_ModeV19 is made visible.
function Tracking_ModeV19_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Tracking_ModeV19 (see VARARGIN)

%% Configuración inicial
background=imread('salva1.jpg'); %Cambiar de acuerdo a la ruta del archivo
set(gcf,'CurrentAxes',handles.axes6)
image(background)
axis off;
handles.s = serial('COM1','BaudRate',9600,'DataBits',8);
set(handles.s,'Terminator','*','Timeout',0.5);
fopen(handles.s);
fprintf(handles.s,'%c','P');

% Choose default command line output for Tracking_ModeV19
handles.output = hObject;
    set(handles.bolita,'Visible','off');
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Tracking_ModeV19 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function   varargout   =   Tracking_ModeV19_OutputFcn(hObject,   eventdata,
handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%% Configuración de la cámara de video
handles.vid=videoinput('winvideo',2,'UYVY_352x240');
handles.prop = getselectedsource(handles.vid);
h=figure(1);
```

```
vidRes = get(handles.vid, 'VideoResolution');
nBands = get(handles.vid, 'NumberOfBands');
hImage = image( zeros(vidRes(2), vidRes(1), nBands) );

preview(handles.vid,hImage);
handles.se = strel('disk',5); %Estructura Disco 5 píxeles

%% Seleccion de Colores
%Primer Color
    [x,y] = ginput(1);
    Im=getsnapshot(handles.vid);

    %Filtrado de la imagen
    Yfilt = Im(:,:,1);
    Cbfilt = medfilt2(Im(:,:,2));
    Crfilt = medfilt2(Im(:,:,3));
    Imfilt = cat(3,Yfilt,Cbfilt,Crfilt);

    Im = im2double(Imfilt);
    selColor = Im(floor(y),floor(x),:);

    %Se extraen los valores de Cb y Cr
    handles.imSelCb = selColor(1,2); %Se extrae Cb*
    handles.imSelCr = selColor(1,3); %Se extrae Cr*
    handles.imSelY = selColor(1,1);


%Segundo Color
    [x,y] = ginput(1);
    Im=getsnapshot(handles.vid);

    %Filtrado de la imagen
    Yfilt = Im(:,:,1);
    Cbfilt = medfilt2(Im(:,:,2));
    Crfilt = medfilt2(Im(:,:,3));
    Imfilt = cat(3,Yfilt,Cbfilt,Crfilt);

    Im = im2double(Imfilt);
    selColor = Im(floor(y),floor(x),:);

    %Se extraen los valores de Cb y Cr
    handles.imSelCb2 = selColor(1,2); %Se extrae Cb*
    handles.imSelCr2 = selColor(1,3); %Se extrae Cr*
    handles.imSelY2 = selColor(1,1);

set(h,'Visible','off')
guidata(hObject,handles)


% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

%% Inicialización de variables
i=1;
distUmbral=0.085;
handles.L=1;
Dir1 = 2;
contador3 = 5;
contador4 = 0;
flag=0;

%% Lazo infinito
while i==1
    Im=getsnapshot(handles.vid);
    a = get(handles.rbNormal,'Value');

    %% Correción Gamma
    if(a==1)
       gamma=1;
    else
        a = get(handles.rbNightVision,'Value');
        if (a==1)
            gamma=0.9;
        else
            gamma=1.07;
        end
    end

    Yfilt = medfilt2(Im(:,:,1));
    Cbfilt = medfilt2(Im(:,:,2));
    Crfilt = medfilt2(Im(:,:,3));
    Imfilt = cat(3,Yfilt,Cbfilt,Crfilt);

    Im = im2double(Imfilt);
    Im(:,:,1)=nvGamma(Im(:,:,1),gamma);
    b = get(handles.rbActivado,'Value');

    tiff=ycbcr2rgb(Im);
    axes(handles.bolita)
    imshow(tiff)

    if (b==1)
    %% Seguimiento activado
        stop=0;
        [r c p]=size(Im);
        %% Pre-procesamiento
        %Se extraen los valores de Cb y Cr
        imCb = Im(:,:,2);
        imCr = Im(:,:,3);

        %% Segmentación
        %Se calcula la distacia Euclidiana para el color seleccionado
        Mask=zeros(r,c); %Máscara de zeros
```

```
        Mask=
nvEuclidiana(single(imCb),single(imCr),single(handles.imSelCb),single(handl
es.imSelCr),single(distUmbral),single(Mask));
        Mask                                                                =
nvEuclidiana(single(imCb),single(imCr),single(handles.imSelCb2),single(hand
les.imSelCr2),single(distUmbral),single(Mask));

        %% Operaciones Morfológicas - Cierre y Apertura(Pre-procesamiento)
        imMejorada = imfill(Mask,'holes'); %Utilizando 4-conectividad
        imMejorada= imclose(imMejorada,handles.se);
        imMejorada = imopen(imMejorada,handles.se);

        %% Operaciones morfológicas - Relleno de regiones, Eliminar borde
de la imagen
        imMejorada = imclearborder(imMejorada);

        %% Segmentación
        [labels, numLabels] = bwlabel(imMejorada); % 8-conecividad

        if (numLabels>=1)
        %% Si detecta objetos en la iamgen
            contador3 = 15;  %Reset de algoritmo de búsqueda
            contador4 = 0;
            clear cent
            s = regionprops(labels,'Centroid');
            for k =1:numLabels
                %% Representación
                hold on
                plot(  s(k).Centroid(1),s(k).Centroid(2),'b*','Linewidth',5)
% centroide tiene dos elementos, coordenadas en x e y.
                hold off

                cent(k,1)=s(k).Centroid(1); %Coordenada X
                cent(k,2)=s(k).Centroid(2); %Coordenada Y
            end
            if numLabels>1
                Punto_Central = mean(cent);
            else
                Punto_Central = cent;
            end
            hold on
            plot(Punto_Central(1),Punto_Central(2),'r*','Linewidth',5);
            hold off
            %% Centro de la imagen de coordenadas (0,0)
            X = Punto_Central(1) - 352/2;
            Y = -Punto_Central(2) + 240/2;
            xlim = 352/4;
            ylim = 240/4;

            %% Movimiento Robot
            if X>xlim
                fprintf(handles.s,'%c','D');   %Derecha
                Dir1 = 1;
            elseif X<-xlim
```

```
                fprintf(handles.s,'%c','A');   %Izquierda
                Dir1 = 0;
            elseif ((X<xlim-30) && (X>-xlim+30))
                fprintf(handles.s,'%c','P');
            end
            if Y>ylim
                 fprintf(handles.s,'%c','I');   %Arriba
             elseif Y<-ylim
                 fprintf(handles.s,'%c','K');  %Abajo
             elseif ((Y<ylim-30) && (Y>-ylim+30))
                 fprintf(handles.s,'%c','U');
            end

            %% Reconocimiento de Patron
            if (numLabels>=3)
                Distancia_Patron = sort(pdist(cent(1:3,:)));
                R1 = Distancia_Patron(1)/Distancia_Patron(2);
                R2 = Distancia_Patron(3)/Distancia_Patron(1);
                if (R1>=0.85)
                    if ((R2/sqrt(2)) >= 0.9 && (R2/sqrt(2)) <= 1.1)   %Caso
triangulo
                        handles.L = (40.4)/Distancia_Patron(3);   %Hallando
la distancia entre el objeto y el robot
                    elseif ((R2/2) >= 0.9 && (R2/2) <= 1.1)      %Caso
diagonal
                        handles.L = (57.8)/Distancia_Patron(3);
                    end
                    set(handles.txtLL,'String',num2str(handles.L));
                end
            end
        else
           %% Algoritmo de busqueda
           disp('Algoritmo de búsqueda');
           if (contador3 < 82)
              if (Dir1==1) %Si el último movimiento fue derecha
                  ch1 = 'D';
                  ch2 = 'A';
              else %Si el último movimiento fue izquierda
                  ch1 = 'A';
                  ch2 = 'D';
              end
              if (mod(contador3,2) == 1)
                  fprintf(handles.s,'%c',ch1);
              else
                  fprintf(handles.s,'%c',ch2);
              end
              contador4 = contador4 + 1;
              if (contador4 == contador3)
                  contador3 = contador3 + 5;
                  contador4 = 0;
              end
           else
              fprintf(handles.s,'%c','P');
           end
        end
```

```
        if ((numLabels >= 1) && (handles.L > 1.3))% Si cumple condiciones
para avanzar
            fprintf(handles.s,'%c','W');
            disp('Avanza')
            flag=1;
        end

        if (handles.L < 1.2 && flag==1)
            fprintf(handles.s,'%c','P');
            flag=0;
        end
    else
        if (stop==0)
            fprintf(handles.s, '%c','U');
            fprintf(handles.s, '%c','P');
            disp('STOP')
            stop=1;
        end
    end
end
guidata(hObject,handles)


% --- Executes during object creation, after setting all properties.
function popup1_CreateFcn(hObject, ~, ~)
% hObject    handle to popup1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



% --- Executes during object creation, after setting all properties.
function texto_CreateFcn(hObject, ~, ~)
% hObject    handle to texto (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes during object creation, after setting all properties.
function txtLL_CreateFcn(hObject, ~, ~)
% hObject    handle to txtLL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes when user attempts to close figure1.
function figure1_CloseRequestFcn(hObject, ~, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: delete(hObject) closes the figure
fclose(handles.s);
delete(hObject);


% --- Executes on button press in radiobutton11.
function radiobutton11_Callback(hObject, eventdata, handles)
% hObject    handle to radiobutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radiobutton11
```

Distancia euclidiana

```
/*********Se debe compilar con cuda_mex archivo.cu CUDA-MATLAB***********/

#include "mex.h"
#include "cuda.h"
#include "cuda_runtime.h"
#include "math_functions.h"

// -------------- the kernel runs on the GPU ---------------------------
__global__  void  nvEuclidiana_Kernel(  float  *imCb,  float  *imCr,  float
*imSelCb,  float  *imSelCr,  float  *imDist,  float  *distUmbral,  float  *iMask,
float *Mask, int arraySize )
{

    int idx = blockDim.x * blockIdx.x + threadIdx.x;

    if (idx > arraySize) return;
    //Aquí se escribe las operaciones a realizar
```

```
    imDist[idx] = hypotf(imCb[idx]-imSelCb[0],imCr[idx]-imSelCr[0]);

     Mask[idx]= iMask[idx];

    if(imDist[idx]<distUmbral[0])
    {
        Mask[idx] = 1;
    }

}

// ----------------- the MEX driver runs on the CPU --------------------
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray
*prhs[] )
{
    //nlhs = número de parámetros de salida
    //plhs = puntero de los parámetros de salida (dirección a la que
apunta)
    //nrhs = número de parámetros de entrada
    //prhs = puntero de los parámetros de entrada (dirección a la que
apunta)


    float *imCb;
    float *imCr;
    float *imSelCb;
    float *imSelCr;
    float *distUmbral;
    float *imDist;
    float *iMask;
    float *Mask;


    mwSize m = mxGetM(prhs[0]);
    mwSize n = mxGetN(prhs[0]);
    mwSize arraySize = m * n;

    mwSize m1 = mxGetM(prhs[2]);
    mwSize n1 = mxGetN(prhs[2]);
    mwSize arraySize1 = m1 * n1;

    // Get the total memory needed for an array on the GPU (in bytes)
    size_t mem_size = sizeof(float) * arraySize;
    size_t mem_size1 = sizeof(float) * arraySize1;

    // Allocate memory on the GPU to hold the input and output data
    if ( cudaMalloc( &imCb, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU1.");
    if ( cudaMalloc( &imCr, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU2.");
    if ( cudaMalloc( &imSelCb, mem_size1 ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU3.");
```

```
    if ( cudaMalloc( &imSelCr, mem_size1 ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU4.");
    if ( cudaMalloc( &distUmbral, mem_size1 ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU5.");
    if ( cudaMalloc( &imDist, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU6.");
    if ( cudaMalloc( &iMask, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU7.");
    if ( cudaMalloc( &Mask, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU8.");

    // Copy the input data across to the card
    cudaMemcpy(     imCb,     (float*)     mxGetData(prhs[0]),     mem_size,
cudaMemcpyHostToDevice);
    cudaMemcpy(     imCr,     (float*)     mxGetData(prhs[1]),     mem_size,
cudaMemcpyHostToDevice);
    cudaMemcpy(    imSelCb,    (float*)    mxGetData(prhs[2]),    mem_size1,
cudaMemcpyHostToDevice);
    cudaMemcpy(    imSelCr,    (float*)    mxGetData(prhs[3]),    mem_size1,
cudaMemcpyHostToDevice);
    cudaMemcpy(   distUmbral,   (float*)   mxGetData(prhs[4]),   mem_size1,
cudaMemcpyHostToDevice);
    cudaMemcpy(     iMask,     (float*)     mxGetData(prhs[5]),     mem_size,
cudaMemcpyHostToDevice);

    // Define the block and grid size - this will have 512 threads per
    // thread block and a sufficient grid so that there is a GPU thread per
    // element of the input array.
    int blockSize = 1024; //Si la operación sólo permite ingresar un
escalar blockSize = 1
    dim3 block(blockSize);  //bloques de 512 hilos (dimension del bloque en
hilos)
    dim3 grid(ceil(arraySize/(float)blockSize)); // numero de bloques por
GRID, basado en la data

    // Use the CUDA runtime to run the kernel
    nvEuclidiana_Kernel<<<           grid,           block           >>>(
imCb,imCr,imSelCb,imSelCr,imDist,distUmbral,iMask,Mask, arraySize);

    // Create the output array for MATLAB
    plhs[0]=mxCreateNumericMatrix(m, n, mxSINGLE_CLASS, mxREAL);


    // Copy the data from the card into the MATLAB array
    cudaMemcpy((float*)mxGetData(plhs[0]),          Mask,          mem_size,
cudaMemcpyDeviceToHost);

    // Free the data on the card
    cudaFree( imCb );
    cudaFree( imCr );
    cudaFree( imSelCb );
    cudaFree( imSelCr );
    cudaFree( distUmbral );
    cudaFree( imDist );
```

```
    cudaFree( iMask );
    cudaFree( Mask );
}
```

Corrección Gamma

```
/*********Se debe compilar con cuda_mex archivo.cu CUDA-MATLAB**********/

#include "mex.h"
#include "cuda.h"
#include "cuda_runtime.h"
#include "math_functions.h"

// -------------- the kernel runs on the GPU --------------------------
__global__ void Operaciones_Kernel( double *I, double *gamma, double *I2,
      int arraySize )
{

    int idx = blockDim.x * blockIdx.x + threadIdx.x;

    if (idx > arraySize) return;
    //Aquí se escribe las operaciones a realizar
    I2[idx] = 255 * powf((I[idx]/255),gamma[0]);
}

// ----------------- the MEX driver runs on the CPU --------------------
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray
*prhs[] )
{
    //nlhs = número de parámetros de salida
    //plhs = puntero de los parámetros de salida (dirección a la que
apunta)
    //nrhs = número de parámetros de entrada
    //prhs = puntero de los parámetros de entrada (dirección a la que
apunta)


    double *I;
    double *gamma;
    double *I2;


    mwSize m = mxGetM(prhs[0]);
    mwSize n = mxGetN(prhs[0]);
    mwSize arraySize = m * n;

    mwSize m1 = mxGetM(prhs[1]);
    mwSize n1 = mxGetN(prhs[1]);
    mwSize arraySize1 = m1 * n1;

    // Get the total memory needed for an array on the GPU (in bytes)
    size_t mem_size = sizeof(double) * arraySize;
```

```
    size_t mem_size1 = sizeof(double) * arraySize1;


    // Allocate memory on the GPU to hold the input and output data
    if ( cudaMalloc( &I, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU.");
    if ( cudaMalloc( &I2, mem_size ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU.");
    if ( cudaMalloc( &gamma, mem_size1 ) != cudaSuccess )
        mexErrMsgTxt("Memory allocating failure on the GPU.");

    // Copy the input data across to the card
    cudaMemcpy(     I,     (double*)     mxGetData(prhs[0]),     mem_size,
cudaMemcpyHostToDevice);
    cudaMemcpy(    gamma,    (double*)    mxGetData(prhs[1]),    mem_size1,
cudaMemcpyHostToDevice);

    // Define the block and grid size - this will have 512 threads per
    // thread block and a sufficient grid so that there is a GPU thread per
    // element of the input array.
    int  blockSize  =  1024;  //Si  la  operación  sólo  permite  ingresar  un
escalar blockSize = 1
    dim3 block(blockSize);  //bloques de 512 hilos (dimension del bloque en
hilos)
    dim3 grid(ceil(arraySize/(double)blockSize)); // numero de bloques por
GRID, basado en la data

    // Use the CUDA runtime to run the kernel
    Operaciones_Kernel<<< grid, block >>>( I,gamma,I2, arraySize);

    // Create the output array for MATLAB
    plhs[0]=mxCreateNumericMatrix(m, n, mxDOUBLE_CLASS, mxREAL);

    // Copy the data from the card into the MATLAB array
    cudaMemcpy((double*)mxGetData(plhs[0]),          I2,          mem_size,
cudaMemcpyDeviceToHost);

    // Free the data on the card
    cudaFree( I );
    cudaFree( gamma );
    cudaFree( I2 );
}
```