INSTITUTO POLITÉCNICO NACIONAL

Centro de Investigación en Computación

Laboratorio de Robótica y Mecatrónica

Control mediante visión de un robot humanoide

TESIS

Que para obtener el grado de:

Maestro en Ciencias de la Computación

Presenta:

Jesús González Godoy

Directores:

Dr. Juan Humberto Sossa Azuela

Dra. Elsa Rubio Espino

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 12:00 horas del dia 29 del mes de mayo de 2013 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Control mediante visión de un robot humanoide"

Presentada por el alumno:

González	Godoy	Jesús
Apellido paterno	Apellido materno	Nombre(s)

Con registro: A 1 1 0 8 6 7

aspirante de: **MAESTRÍA EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Juan Humberto Sossa Azuela

Dra. Elsa Rubio Espino

Dr. Carlos Aguilar Ibáñez

Dr. Herón Molina Lozano

Dr. Domingo de Jesús Cortés Rodriguez

Dr. Ricardo Barrón Fernández

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México Distrito Federal el día 05 del mes de Junio del año 2013, el que suscribe Jesús González Godoy alumno del Programa de maestría en Ciencias de la Computación con número de registro A110867, adscrito al Centro de Investigación en Computación, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Juan Humberto Sossa Azuela y la Dra. Elsa Rubio Espino y cede los derechos del trabajo intitulado Control mediante visión de un robot humanoide, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección my_son_jeyton@hotmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Jesús González Godoy

Jesús González Godoy

Nombre y firma

Resumen

EN EL PRESENTE TRABAJO SE INTEGRÓ LA ROBÓTICA CON LA VISIÓN ARTIFICIAL, DOS TÓPICOS QUE HOY EN DÍA SE ENCUENTRAN ESTRECHAMENTE RELACIONADOS Y SON PUNTOS DE PARTIDA PARA INVESTIGACIONES EN TODO EL MUNDO. SE DOTO A UN ROBOT HUMANOIDE CON LA CAPACIDAD DE DIRIGIRSE HACIA UNA PELOTA EN MOVIMIENTO Y PATEARLA, USANDO UNA CÁMARA COMO MEDIO PARA PERCIBIR SU ENTORNO. ESTA TAREA ES REALIZADA DE MANERA AUTÓNOMA, SIENDO EL BOTÓN DE ENCENDIDO LA ÚNICA INTERVENCIÓN HUMANA NECESARIA. PARA LLEVAR A CABO EL SEGUIMIENTO DE LA PELOTA Y RECONOCER EL MOMENTO EXACTO EN EL QUE EL ROBOT DEBE PATEARLA, SE CREÓ UN ALGORITMO SIMPLE PERO PODEROSO, EFICIENTE EN EL AMBIENTE ALTAMENTE DINÁMICO EN EL QUE SE DESEMPEÑA Y PODEROSO ANTE CAMBIOS IMPREVISTOS O RUIDO.

Abstract

ROBOTICS AND COMPUTER VISION ARE TWO TOPICS CLOSELY RELATED AND THE STARTING POINT FOR RESEARCHERS IN ALL THE WORLD. A HUMANOID ROBOT WAS GIVEN THE CAPACITY OF ADDRESSING ITSELF TOWARDS A MOVING BALL AND KICK IT, USING A CAMERA AS THE WAY TO PERCEIVE ITS ENVIRONMENT. THIS TASK IS PERFORMED AUTONOMOUSLY, THE ONLY NECESSARY HUMAN INTERACTION IS THE ON/OFF KEY. TO SUCCESSFULLY FOLLOW THE BALL AND DETERMINE THE EXACT MOMENT WHEN IT SHALL BE KICKED, A SIMPLE BUT POWERFUL ALGORITHM WAS DEVELOPED, HIGHLY DYNAMIC AND EFFICIENT IN THE TARGETED ENVIRONMENT BUT ALSO STRONG WHEN NOISE OR CHANGES ARE INTRODUCED.

AGRADECIMIENTOS

A mis padres, mi apoyo incondicional e invaluable, mis mejores amigos, gracias por su consejo, su guía e incansable búsqueda de mi bienestar, el saber que los tengo a mi lado, me da la seguridad y fuerza necesarias para lograr mis metas en la vida. Todo lo que soy ahora se los debo a ustedes.

A mi hermana, por brindarme su cariño y apoyo en todo momento, creer en mí, compartir mis triunfos y alegrías. Especialmente gracias por tu ayuda en mi trabajo de Tesis, con la cual supere momentos de mucha presión.

A mi esposa, por llenar mi vida de alegría, permanecer siempre a mi lado y apoyarme siempre sea cual sea el camino que tome. Gracias por compartir con migo este amor tan especial que sé, durara para siempre.

A mis directores por creer en mí y por su tolerancia en momentos de zozobra, gracias a sus consejos y aportaciones que llevaron a buen término el desarrollo de la tesis.

Al CONACYT por brindarme su apoyo durante estos años. Además también Este trabajo de investigación fue realizado en el marco de los proyectos con número de registro: SIP 20121311, SIP 20131182, SIP 20131505 y CONACYT 155014.Con los que se compró el material necesario y sin el cual no hubiera sido posible la realización de este trabajo.

Al CIC y al IPN, mi casa de estudios que siempre tendrán un lugar en mi corazón. Y estaré agradecido y orgulloso por una formación sólida que me distingue como profesionista de excelencia. Un Politécnico.

CONTENIDO

FIGURAS

TABLAS

1. INTRODUCCÓN

La historia de la humanidad ha sido enriquecida de distintas formas y en distintas eras, las cueles impulsaron su desarrollo y evolución sobre la faz de la tierra. El presente año forma parte evidente de una nueva era, una revolución tecnológica que no solo es culpable de un cambio en el comportamiento social de las masas, también es responsable de romper paradigmas, generar mayor conocimiento, romper la barrera de lo real y la fantasía.

1.1 ESTADO DEL ARTE

En la actualidad muchos centros de investigación y empresas de todo el mundo, están tratando de dotar a los robots con características más humanas en todos los sentidos y con diversos enfoques, pero nada más parecido y adecuado para introducir esta tesis que las competencias de la RoboCup.

La RoboCup es una organización mundial que promueve la investigación en inteligencia artificial y robótica, proponiendo objetivos a cumplir a través de distintos concursos y diferentes categorías. El sueño de los organizadores de la RoboCup es que para el año 2050 un equipo de robots autónomos derivado de estas investigaciones pueda jugar un partido de futbol con el actual campeón del mundo de la FIFA.

Dentro de estas categorías se encuentra la de robots humanoides, en donde dos equipos de robots humanoides se enfrentan en un partido de futbol. Los robots son completamente autónomos y no puede haber intervención humana durante el partido.

Las técnicas utilizadas en estos encuentros convergen en el uso de una o dos cámaras montadas en la cabeza del robot y por medio de técnicas de análisis de imágenes ubicar la pelota para pasar las instrucciones necesarias a los actuadores.

Últimamente se ha optado por ocupar un modelo estándar, que es el jugador de casi todos los equipos participantes en la RoboCup. Los modelos de moda en estos momentos son el robot NAO de aldebaran robotics y el robot DARwln de ROBOTIS, en 2007 tuvo lugar el BIOLOID como robot oficial de la RoboCup.

En la pasada RoboCup del 2012 le tocó a México ser el país anfitrión del evento con más de 40 países participantes, entre las escuelas participantes en la RoboCup se encuentran el Instituto Politécnico Nacional, la Universidad Autónoma de México, El Tecnológico de Monterrey, la Universidad La Salle, etc.

Estos robots son capaces de caminar de frente, de lado, para atrás, para adelante, patear la pelota con los dos pies y de distintas formas, e incluso de comunicarse entre ellos y llevar a cabo estrategias que les den la victoria.

Cada equipo tiene un portero que cuando detecta que un jugador tira la pelota hacia él, éste se avienta de manera que pueda evitar un gol del equipo contrario. Algunos equipos tienen una inteligencia artificial tan completa, que el robot protege la pelota de los demás adversarios hasta llevarla cerca de la portería del equipo contrario.

La estabilidad de los robots es bastante buena actualmente, sin embargo no existe partido alguno en el que no se caiga algún robot por lo menos. Esa es una de las cosas que actualmente se quieren evitar y por la que se realizan investigaciones en todo el mundo.

Los robots detectan el color de la pelota, el color de la cancha y de las porterías y posiblemente el de los contrarios y con eso elaboran la estrategia seguir.

A pesar de todo esto que se menciona queda mucho por hacer, uno de estos problemas es que los robots actuales no son capaces de interceptar la pelota en movimiento, y sólo se limitan a seguirla de manera muy básica y la pierden a menudo, lo que provoca un barrido de toda la cancha para averiguar dónde quedó la pelota.

Para controlar los movimientos de estos robots se utilizan distintos algoritmos. En la presente tesis se quiere abordar el control basado en comportamientos que actualmente está siendo adoptado por investigadores de todo el mundo y tiene como principal exponente al Dr.Rodney Brooks quien es profesor emérito del MIT. Esta nueva tendencia en robótica está enfocada hacia la simplicidad y adaptabilidad. Toma como ejemplo a la naturaleza y enfatiza una reacción rápida sin conocimiento o planificación. "planning is just a way of avoiding figuring out what to do next" título de un artículo de Brooks en 1987 es un ejemplo claro de esta afirmación.
Robots que actualmente utilizan esta filosofía son los de la empresa iRobot fundada en 1990 por Rodney Brooks y cuyos productos mas destacados son robots de limpieza, también esta el robot Baxter de rethink robotics anunciado como el robot con sentido común. Baxter es usado también para la investigación.
También están los trabajos de Ronald Arkin (1998), roboticista Americáno y profesor en el Instituto de Tecnología de Georgia y es conocido por su libro *Behavior-Based Robotics.* También ha empleado esta técnica en colonias de robots que menciona en diversos artículos.
Pattie Maes es otra de las investigadoras que ha trabajado en este tema, ella es discípula de Brooks y ha fundado su propio grupo de investigación el cual realiza trabajos realmente interesantes aunque enfocados principalmente a la interacción humano máquina.
Esta investigadora lleva esta aproximación aún más allá argumentando que no solo se limita a los robots, sino a todo sistema complejo que exista en un entorno impredecible y dinámico como sus interfaces hombre máquina por ejemplo. Estas ideas abren toda una línea nueva de investigación que genera resultados espectaculares que crean visiones de un futuro encantador tecnológicamente hablando.

1.2 OBJETIVO

El objetivo de la tesis es hacer que el robot humanoide Bioloid logre patear la pelota cuando ésta se encuentra en movimiento, bajo un entorno controlado y para una situación específica a modo que sirva de ejemplo a trabajos posteriores que resuelvan el problema considerando más restricciones y ejecutando las tareas en un menor tiempo, abarcando un rango más amplio de situaciones posibles. Las metas para llevar a cabo este objetivo son:

1. Implementar la interconexión entre el módulo de visión y el robot.
2. Usar el análisis de las imágenes de la cámara para ubicar la pelota y decidir los movimientos del robot.
3. Crear un control para el sistema de manera que el robot patee la pelota en movimiento mientras ésta se acerca a él.
4. El algoritmo de control será útil no solo para este robot humanoide si no para cualquier otro, esto adaptando las variables de configuración.
5. El algoritmo debe ser rápido de manera que sea eficiente en un encuentro real de futbol de robots.

2. MARCO TEÓRICO

En este capítulo se dará un repaso de los principales conceptos manejados en la implementación del análisis de imágenes y la creación del control del robot. Algunos de estos temas podrían ser muy extensos, en ese caso se dará solo un repaso general y se dará alguna referencia en caso de que se requiera profundizar.
En caso de querer profundizar más en algún tema se pueden consultar las referencias al final del texto donde se abarcan en su totalidad los temas abordados aquí, además de explicarse en detalle cada uno de ellos con ejemplos y definiciones en extenso.

2.1 SERVO MOTORES

Los servo motores son dispositivos finales, que dotan de movimiento al robot. Estos dispositivos a diferencia de los motores normales cuentan con circuitos electrónicos que me permiten colocar el eje del motor en una posición exacta deseada, por medio del envío de una señal electrónica. Estos motores son ampliamente utilizados en robótica.
Además del control del ángulo de giro en un servomotor, estos poseen una reducción de engrane que pueden ser de metal o de plástico dependiendo la calidad del motor, por esta razón estos dispositivos son realmente poderosos y soportan bastante peso, aunque si este es demasiado, entonces obviamente el rendimiento del motor se ve afectado, así que entre las especificaciones de cada motor se encuentra el peso soportado por ellos.
Una característica esencial de los servomotores y que fue clave en el desarrollo de esta tesis, es que su velocidad es proporcional a la distancia que se les indica recorrer debido al voltaje aplicado para lograr su cometido.
Los servomotores tienen un rango de operación que va desde los 0 a los 180 grados. Normalmente algunos poseen un rango mayor como es el caso de los motores usados en la construcción del robot, que poseen un rango de operación que va de los 0 a los 360 grados.
La forma en la que se indica al motor a que ángulo debe dirigirse, es por medio de pulsos eléctricos con cierta codificación. Esta señal debe ser mantenida durante el tiempo que se desee que el motor mantenga esa posición, de lo contrario el eje del motor estará a merced de las fuerzas externas aplicadas a él.
La comunicación entre el controlador y los motores es serial, estos poseen tres cables, dos de alimentación y uno para el envío de los pulsos de control que son generados por el micro controlador una vez decidida la acción a realizar.

FIGURA 1. SERVOMOTOR AX-12A DYNAMIXEL.COM

2.2 CAPTURA DE IMÁGENES

Para poder analizar las imágenes que representan el mundo real, primero debo obtenerlas de alguna forma y pasarlas a formato digital, para esto se necesita un dispositivo físico sensible a un cierto rango del espectro electromagnético, por ejemplo los rayos x, este dispositivo debe tener como salida una señal eléctrica directamente proporcional a la cantidad de luz que incide en el. En este caso estamos trabajando con el espectro de luz visible únicamente.

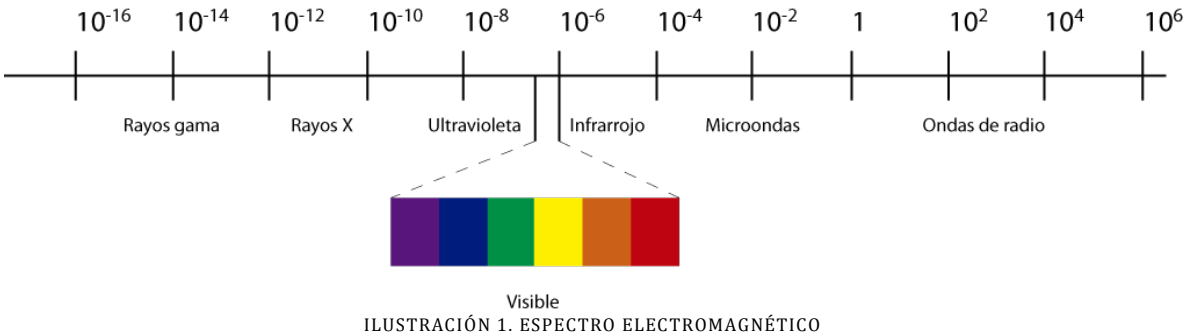


ILUSTRACIÓN 1. ESPECTRO ELECTROMAGNÉTICO

Una vez hecho esto se necesita traducir la señal eléctrica obtenida del sensor en un conjunto discreto que conforma la imagen digital, a lo que se le denomina muestreo y cuantización.
Existen múltiples tipos de dispositivos físicos capaces de capturar la incidencia de la luz visible y arrojar la señal eléctrica necesaria, pero detallarlos aquí no es relevante para la comprensión del desarrollo de la tesis. Por ahora es suficiente conocer que la imagen a tratar debe estar representada de manera digital y se puede ver como una matriz de MxN elementos, obviamente también interviene el sistema óptico utilizado. Cada elemento de esta matriz es denominado pixel que viene de "picture element" o elemento de la imagen por su traducción, cada uno de estos elementos tendrá un valor asignado que representa el nivel de luminosidad del punto que corresponde a la escena capturada y es el resultado del muestreo y cuantización.
Dependiendo el número de pixeles con los que cuente el dispositivo se determinara la resolución en pixeles de la imagen, también conocida como resolución espacial.
Una vez hecho esto las imágenes pueden ser almacenadas y tratadas o enviadas ya sea a una pantalla, monitor o impresora.

FIGURA 2. DIFERENTES RESOLUCIONES DE IMAGEN

2.3 IMÁGENES DIGITALES

La imagen como tal, representa una función bidimensional de la intensidad de luz que incide en el dispositivo, y se puede representar como $f(x, y)$ de manera general aunque cada autor puede usar la nomenclatura que mejor le acomode, donde x e y son las coordenadas de cada pixel perteneciente a la matriz que conforma la imagen.
De esta representación de imagen como función se derivan la aplicación de n operaciones matemáticas y de la demostración formal de estas operaciones y sus resultados.
Cuando se captura una imagen en color, realmente es representada con tres matrices $n \times m$, una para el rojo, una para el azul y otra para el verde y cada pixel en cada una de estas matrices está representado por 8 bits en el espacio RGB, aunque las cámaras actuales realmente no lo trabajan así, por ahora será suficiente esta consideración.

FIGURA 3. REPRESENTACIÓN DE IMAGENES DIGITALES

2.4 ETIQUETADO DE COMPONENTES CONEXAS.

El etiquetado de componentes conexas es un método ampliamente utilizado en el análisis de imágenes; y tiene como objetivo encontrar las regiones principales existentes en la imagen, donde una región representa un conjunto de pixeles de la imagen que son similares en alguna de sus características.
Para llevar a cabo este tipo de extracción se parte del hecho de que la imagen ha pasado por un proceso de filtrado, que consiste de dos partes la primera es obtener una imagen escala de grises de la imagen original, después se crea una imagen de la misma resolución, en la cual el valor de cada pixel (x, y) es el promedio de los valores RGB del pixel (x, y) de la imagen original y va de 0 a 255. Lo segundo, es comparar el valor de cada uno de esos pixeles con un valor de umbral seleccionado con anterioridad, si éste valor es mayor se cambia a uno el valor del pixel y si es menor se asigna un cero. De esta manera la imagen solo está compuesta de dos valores: cero y uno, o lo que es lo mismo tenemos una imagen binaria.
La mayoría de las veces aunque la imagen esté compuesta de solo dos valores, es posible para los seres humanos distinguir su contenido y quiere decir que aun aporta información valiosa para el análisis pero facilita mucho el procesamiento de los algoritmos necesarios para extraer la información o datos que se van a utilizar.

Imagen original | Escala de grises | Binarizada

FIGURA 4. IMAGEN BINARIA

En algunos casos más avanzados se trabaja con las tres matrices de la imagen la del color rojo, la del color verde y el azul formando así un histograma tridimensional como es el caso del módulo HaViMo.
El concepto es simple, todos los pixeles con valor uno y que estén conectados entre sí se les asigna una etiqueta que los identifica como miembros de la misma región. Un pixel puede ser cuatro conexo u ocho conexo, existen distintos algoritmos e implementaciones que logran este cometido.

1	1	1
1	1	1
1	1	1

4-conexo

1	1	1
1	1	1
1	1	1

8-conexo

FIGURA 5. TIPOS DE CONECTIVIDAD

Existe por ejemplo un algoritmo iterativo que consiste en recorrer fila por fila de arriba hacia abajo y después de abajo hacia arriba iterativamente hasta que la imagen sea etiquetada completamente.

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

Imagen binaria

0	0	0	0	0	0	0	0	0	a
0	b	b	0	0	0	0	0	0	0
b	b	b	0	0	c	0	0	0	0
0	b	b	0	0	c	c	c	0	0
0	0	0	0	0	c	c	c	0	d
0	0	0	0	0	c	c	0	0	d
0	0	d	0	0	0	c	0	0	d
0	0	d	0	0	0	0	0	0	d
0	0	d	0	0	0	0	0	0	d
0	0	d	d	d	d	d	d	d	d

Regiones etiquetadas

FIGURA 6. ETIQUETADO DE COMPONENTES CONEXAS

2.5 ROBÓTICA BASADA EN COMPORTAMIENTOS.

Cuando un sistema está compuesto por muchos módulos en ocasiones se presentan problemas de conectividad y con mayor razón si estos fueron construidos por separado, por lo que el esfuerzo en investigación está tomando un nuevo enfoque en el cual se puedan realizar diseños compuestos de varios módulos desarrollados de forma simultánea y coordinados a la perfección.

Uno de los pioneros en este enfoque es el investigador Rodney Brooks quien fue director del "MIT Computer Science and Artificial Intelligence Laboratory" hasta el 2007 y fundador de empresas como iRobot. Esta nueva tendencia en robótica está enfocada hacia la simplicidad y adaptabilidad. Toma como ejemplo a la naturaleza y enfatiza una reacción rápida sin conocimiento o planificación. "planning is just a way of avoiding figuring out what to do next" título de un artículo de Brooks en 1987 es un ejemplo claro de esta afirmación.

Con este enfoque la IA simbólica queda como anticuada, no es necesario tener mucho conocimiento para realizar acciones simples que máquinas y robots muy complejos actualmente no pueden lograr.

Un comportamiento puede ser asociado con una red neuronal, un sistema de ecuaciones, un algoritmo etc. Y es la forma en la que un robot reacciona a ciertas condiciones de su entorno, y es gobernado por una arquitectura de control que indica que comportamiento entra en que circunstancia. Comportamientos muy utilizados en este tipo de enfoque son los comportamientos reactivos o el aprendizaje por refuerzo. En la navegación de robots móviles es experimentalmente sencillo comprobar la rapidez y eficiencia de este tipo de arquitecturas contra un robot que realiza una planificación para terminar su recorrido.

En la presente tesis, se aprovecha la robustez de este enfoque al hecho de responder eficientemente a imprevistos en caso de que exista un alto grado de complejidad en la predicción de los escenarios que pueden existir en el entorno y los valores de las variables involucradas.

Los comportamientos suelen ser simples, tan simples que parece que no pudieran resolver un problema complejo, pero asombrosamente en conjunto y aplicados a la robótica se desempeñan tan bien que pareciera que el robot tiene realmente vida.

2.6 ARQUITECTURAS SUBSUMIDAS

Este tipo de arquitecturas fueron propuestas por Rodney Brooks en 1986, y como dice José Santos (2005) son un ejemplo de interconexión de comportamientos jerárquicos. Los comportamientos reciben información a través de los sensores y en base a esto influyen sobre los actuadores. Los comportamiento s están relacionados de tal manera que los de alto nivel subsumen a los de bajo nivel, este tipo de arquitectura es incremental, es decir que se pueden ir añadiendo más comportamientos con forme el sistema lo requiera.

A continuación se muestra el esquema de control por capas propuesto por Brooks en 1985.

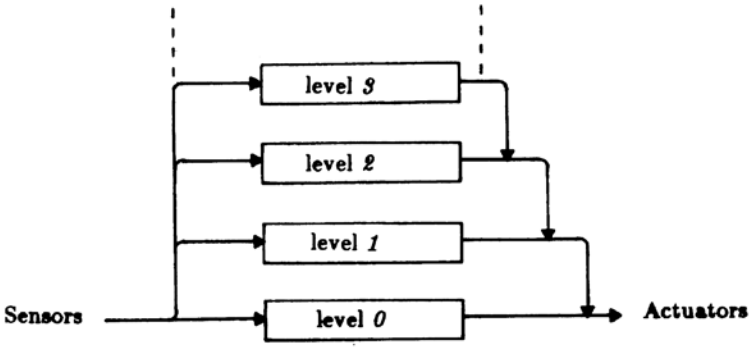


FIGURA 7. ARQUITECTURA SUBSUMIDA DE BROOKS A.I MEMO 864

En este artículo Brooks también muestra un ejemplo concreto de un sistema de control robusto para un robot móvil. Que se muestra a continuación.

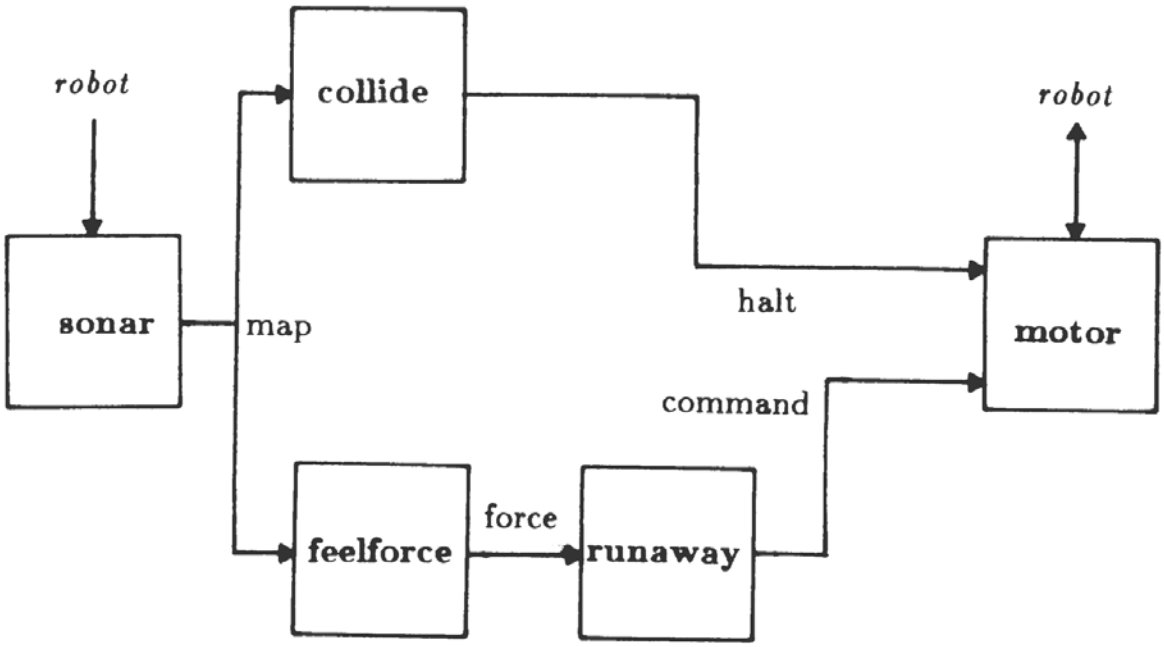


FIGURA 8. NIVEL 0 DE SISTEMA DE CONTROL DE BROOKS A.I MEMO864

En las arquitecturas subsumidas cada comportamiento se representa como un autómata de estados finitos.
Ronald Arkin (1998) dio una definición formal a estos sistemas basados en comportamiento, definiéndolos como pares estimulo-respuesta modulados por la atención y determinados por la intención. Donde la atención prioriza las tareas y la intención determina los comportamientos a ser activados. Jose Santos (2005). La mayoría de los comportamientos usados en esta tendencia suelen ser reactivos, es decir que solo se basan en la información actual de los sensores, sin planeación o cálculos de ningún otro tipo.
Este tipo de sistemas actúan mucho más rápido ya que generalmente requieren poco procesamiento de la información. También reaccionan de manera más eficiente acercándose más al objetivo propuesto frente a situaciones no previstas, debido a que su comportamiento se basa en su mayoría en el estado actual de su entorno.
Algo interesante de estos diseños a diferencia de los robots planificadores por ejemplo es que el funcionamiento depende en gran parte del diseño de los comportamientos, esto hace posible que incluso cuando el robot debe realizar muchas tareas o actuar de una manera compleja, solo algunos comportamientos gobernando sus movimientos podrían ser suficientes.
En ocasiones sería de gran ayuda automatizar el diseño de estos comportamientos, tema en el que actualmente se trabaja y se espera que mejore la eficacia de este método que por sí solo ya es bastante bueno.

2.7 AUTÓMATAS FINITOS

Los autómatas finitos son modelos computacionales con memoria limitada. Estos modelos computacionales aunque limitados son extremadamente comunes y los usamos a diario, cuando usamos la televisión, el horno de microondas el celular, etc.
Un autómata finito es una forma de representar los estados o situaciones que se pueden presentar en el sistema que se va a modelar y las transiciones entre estos, dependiendo siempre de las entradas al sistema y del estado en el que se encuentre en ese momento.
Los autómatas finitos son herramientas útiles cuando por ejemplo intentamos reconocer patrones de datos, por ejemplo procesamiento de palabras o reconocimiento de objetos.
Un autómata finito se compone de varias partes. Tiene un conjunto de estados y reglas para pasar de un estado a otro dependiendo de los símbolos permitidos en su entrada.
En su definición formal, un autómata finito es una 5-tupla $(Q, \Sigma, \delta, q_0, F)$ donde:

- Q, es un conjunto finito de estados.
- Σ, es un conjunto finito llamado el alfabeto.
- $\delta: Q \times \Sigma \rightarrow Q$, es la función de transición, esta define las reglas de movimiento
- $q_0 \in Q$, es el estado inicial
- $F \subseteq Q$, es el conjunto de estados aceptores o estados finales.

Se puede usar la definición formal para describir un autómata finito o se puede usar su representación gráfica como se muestra en el siguiente ejemplo. Sipser(2006).

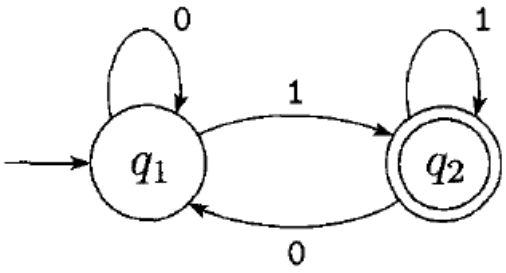
Si se tiene l siguiente autómata finito.

$$M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$$

Donde δ se representa como:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

Entonces su representación gráfica sería:



Esta clase de autómatas tienen muchas aplicaciones a pesar de que es solo la base fundamental de la teoría de autómatas, lo que esta fuera de la intención de esta explicación, para el modelado de los comportamientos es suficiente conocer esta teoría básica.

2.8 CONTROL PROPORCIONAL INTEGRAL DERIVATIVO

El Control proporcional integral derivativo también conocido como PID es un mecanismo que calcula básicamente la diferencia o error entre una cantidad medida y una referencia, para, de acuerdo con este resultado realizar alguna acción de ajuste que corrija esta desviación.

El uso del PID no garantiza por si solo un control óptimo o la estabilidad del sistema. Para la correcta aplicación de este mecanismo se necesita un sensor que me proporcione la información necesaria o el valor medido, un controlador que se encargue de realizar el ajuste necesario y un actuador que ayude en este proceso.
De manera general un control PID se puede representar como:

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt}$$

Donde $u(t)$ es la salida del controlador. Principalmente se utiliza para controlar variables como presión, flujo, velocidad, fuerza, etc.

El problema de aplicación de este controlador consiste en la elección de las variables $k_p, k_i \; y \; k_d$, ya que de esto depende el funcionamiento del controlador, es decir que una mala elección de valores para estas variables desemboca en un mal funcionamiento y una correcta combinación de variables nos dará el funcionamiento deseado.
Además de esto existen otras consideraciones a tomar en cuenta a la hora de diseñar el controlador, la señal P indica la constante de proporcionalidad, mientras que la señal I me dice con qué velocidad se repite la acción proporcional P y la señal D afecta directamente la ganancia del proceso.

Existen varios métodos de ajuste de las ganancias para un controlador PID entre ellos:

- Método de Oscilación
- Método basado en la curva de reacción.

El PID ha demostrado buenos resultados a través del tiempo y ha sido adoptado como uno de los controladores más ampliamente utilizados, por su sencillez y eficacia, además del amplio rango de aplicaciones que puede tener, en algunos trabajos se le compara con algoritmos de lógica difusa que pretenden generar un cambio de paradigmas, y aunque los resultados de estos enfoques son buenos, el PID seguirá en uso.

3 DISEÑO E IMPLEMENTACIÓN

Existen tres etapas a considerar para el diseño e implementación de la tesis, la primera es la etapa donde se consideran los aspectos mecánicos y físicos involucrados en el proceso de construcción del robot, la segunda etapa está relacionada con la visión del robot, como logra percibir su entorno y en particular identificar su objetivo el cuál será caracterizado perfectamente en esta etapa, por último se trata el control usado para cumplir su tarea, que técnicas y algoritmos fueron ocupados, cuales son las entradas y salidas del sistema y algunos otros aspectos cruciales de esta investigación.

3.1 CONSTRUCCIÓN

El presente trabajo no está enfocado hacia el diseño de un robot humanoide si no al control de este por medio de visión artificial, una misión completamente distinta aunque complementaria. Debido a esto se decidió usar un modelo comercial que contara con las características necesarias y fuera lo suficientemente confiable.

3.1.1 KIT DE CONSTRUCCIÓN

Después de un análisis de los modelos humanoides existentes en el mercado se decidió utilizar el kit de robótica educativa BIOLOID, que incluye lo necesario para diseñar, construir y programar un robot humanoide, y de cualquier otro tipo en general.
Sus principales características son las siguientes:

FIGURA 9. ROBOT HUMANOIDE TIPO A

3.1.1.1 CONTROLADOR

Cuenta con un controlador CM-510 que está basado en un ATMega2561, este tiene 6 puertos auxiliares de 5 pines para uso de sensores y dispositivos externos, tiene un mecanismo de detección automática de la batería y monitorea su nivel de carga, tiene también 5 puertos TTL de 3 pines para controlar actuadores y sensores. Cuenta con un micrófono integrado y un pequeño generador de melodías. Se puede programar con el lenguaje propio de RoboPlus o directamente con el lenguaje C.

3.1.1.2 ACTUADORES.

El kit contiene 18 motores AX-12+ de dynamixel que van de 0 a 300 grados aproximadamente con una resolución de 0.29°, funcionan de -5°C a 70°C, se alimentan con 9 o 12 volts y usan un protocolo de comunicación Half duplex Asynchronous Serial Communication (8bit,1stop,No Parity) a una velocidad de alrededor de 1Mbps.
Ofrecen información acerca de su temperatura, carga, nivel de voltaje y obviamente su posición además de otras cosas.
Se comunican al controlador por medio de un ID que va de 0 a 253 y es totalmente configurable. Se conectan en cadena desde el CM-510 por medio del esquema daisy chain.

3.1.1.3 SENSORES.

Dentro del kit se encuentran varios sensores que facilitan la construcción de un modelo mucho más interactivo, por ejemplo sensores de distancia, infrarrojo en caso de querer controlar el robot con un control remoto también incluido en el kit etc. Uno de ellos es de vital importancia para el equilibrio de un robot humanoide, el giroscopio "Gyro GS-12 de 2 ejes", mide la velocidad angular, es decir que permite conocer la inclinación del robot. Opera desde -40° hasta 85°.

3.1.1.4 COMUNICACIÓN INALÁMBRICA

Los robots creados con este kit y controlados con el CM-510, se comunican hacia la PC con protocolo serial para descargar los programas y ejecutar comandos algunos comandos especiales, pero ROBOTIS ofrece otras posibilidades, en este caso se recurrió a la comunicación ZigBee, con los módulos Zig100 para la PC y Zig110 para el robot, ambos son emisor y receptor y en el caso del Zig110 tiene una coraza de plástico protectora de interferencias, ruido y señales cruzadas. Trabaja en los 2.4 GHz con un ancho de banda de 250 kbps máx. Su velocidad alcanza los 57600bps.
Para conectar el Zig100 con la PC se necesita un adaptador especial llamado Zig2Serial, que cambia la señal TTL a señal de nivel RS232C.
El modo de comunicación con la PC es siempre serial, pero las computadoras modernas ya no tienen este puerto integrado, la respuesta de ROBOTIS a esto es la interfaz USB2Dynamixel que tiene características de integración entre las señales RS232, RS485 y TTL con el puerto USB, además de un conjunto de librerías para usar con en el entorno preferido del programador.

3.1.1.5 MODELO HUMANOIDE

Este Kit en particular me permite crear una gran variedad de robots, y tiene modelos propuestos con programas diseñados especialmente para ellos. Entre estos modelos se encuentran tres robots humanoides cada uno diferente de los demás en la cintura y como consecuencia en los movimientos que puede ejecutar.
El modelo de robot que mejor se adapta a los requerimientos previstos es el modelo humanoide de tipo A, que se muestra en la Figura 6. Este modelo cuenta con programas que dotan al robot de funciones especiales de una actividad específica. Por ejemplo jugar futbol o participar en un combate de robots.

3.2 VISIÓN

El robot usa como medio principal para percibir su entorno una cámara que será montada en su parte superior. Para este fin se probaron dos cámaras que se describen a continuación para después explicar el montaje y las herramientas utilizadas.

3.2.1 HAVIMO

HaViMo es un módulo de visión usado para darle a robots de diversas compañías la habilidad de percibir su entorno e interactuar con el basado en la información adquirida de las imágenes.
Esta cámara cuenta con diversos algoritmos que con los cuales se procesa la imagen antes de enviarla al su puerto de salida, cada uno de ellos con múltiples aplicaciones.

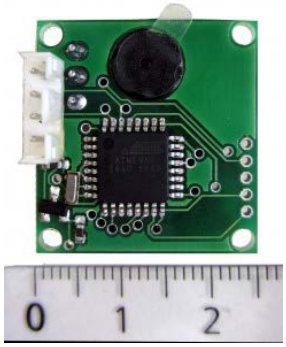


FIGURA 10. HAVIMO EN CM
WWW.HAVIMO.COM

Características relevantes:

Cámara:

- Una cámara CMOS con resolución de 160 por 120 pixeles.
- Velocidad de captura de 19 Fps.
- Permite el acceso a todos los registros de la cámara.
- Guarda los valores de configuración en una EEPROM.
- Balance de blancos
- Procesamiento de la imagen basado en color.

Look-Up Table.

- Se guarda en una memoria flash, por lo que no es necesario recalibrarla cada vez que se enciende.
- Se pueden definir hasta 256 colores
- Herramientas de edición y visualización 3d.
- Superpone los colores definidos sobre las imágenes de la cámara en tiempo real

Regiones:

- Detección de hasta 15 regiones por captura.
- Reporta el número de pixeles y la caja contenedora de cada región.

Software soportado:

- RoboPlus (ROBOTIS).
- Lenguaje C.
  Lenguaje C#.

Griding

- Reduce la resolución de la imagen a 32 por 24.
- Garantiza la mínima perdida de información usando prioridad de objetos.
- Reporta el color y el número de pixeles de cada celda 5x5.

Hardware soportado

- 115200 baudios para modo Half Duplex (ROBOTIS)
- Controlador CM5
- Controlador CM510
- USB2Dynamixel
- 1 Mbaudio para modo Full Duplex (RoboBuilder)
- RBC
- Niveles TTL RS232

3.2.1.1 PRUEBA DE CALIBRACIÓN Y DETECCIÓN DE LA PELOTA CON HAVIMO2.0

En el manual, se ofrecen distintas posibilidades en las que se puede hacer uso del módulo para obtener las imágenes pre procesadas y trabajar con ellas. Para realizar las primeras pruebas de calibración y detección de la pelota, se usó una configuración directa; La cual se muestra a continuación.

FIGURA 11. CALIBRACIÓN DIRECTA USANDO USB2DYNAMIXEL

Se hizo uso de una herramienta de calibración llamada HaViMoGUI, que es una aplicación de código abierto proporcionada desde http://sourceforge.net/projects/havimo/ en ella se permiten configurar muchos aspectos del módulo como ganancia, brillo, saturación, acceso a los registros etc. Entre estas características esta la configuración de la LUT, que es muy intuitiva y de gran ayuda para el reconocimiento de los objetos usando su color.
El proceso a seguir fue, capturar una imagen del escenario, con el objetivo dentro de ella, después en base a esa captura se calibra la cámara poniendo los colores que se desea que se identifiquen en la LUT, una vez hecho esto el módulo realizará el pre procesado de la imagen y colocará en registros específicos los resultados obtenidos, que incluyen la imagen obtenida, las regiones encontradas de acuerdo con la información en la LUT, la posición de las cajas contenedoras de estas regiones y sus derivados, como son su centro, tamaño etc.
A continuación se muestra el resultado después de haber calibrado el módulo para que identifique regiones de color rojo, primero aparece el cuadro de captura, después aparece el cuadro de captura pero con los colores de la LUT resaltados con color azul y por último aparece la misma imagen capturada pero con la caja contenedora de la región identificada de color rojo.

FIGURA 12. EJEMPLO DE CALIBRACION CON HAVIMOGUI

Con esto ya se sabe que la cámara me regresará siempre la posición de la pelota roja, lo que el robot puede usar para darle seguimiento. Claro que no tiene un funcionamiento ideal, ya que la cámara es muy sensible a los cambios de iluminación y aun con iluminación artificial, si se tiene una mínima dependencia de la luz del sol, es muy difícil mantener los resultados obtenidos sin necesidad de recalibrar la cámara varias veces al día, lo que se vuelve algo molesto con el paso del tiempo además de afectar directamente en el comportamiento del robot.

3.2.1.2 INTEGRACIÓN HAVIMO-BIOLOID

El robot requiere una completa movilidad en el cuello para poder seguir en todas direcciones la pelota, entonces se diseñó un montaje que pudiera cumplir este requerimiento. Para esto se usaron dos motores Dynamixel AX-12+ y una estructura construida con Sintra, kola loca y tornillos.

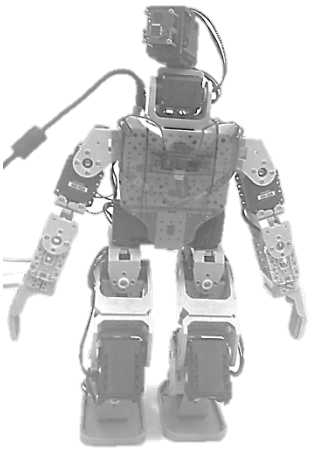


FIGURA 13. MONTAJE HAVIMO-BIOLOID

Este diseño permite que el robot siga la pelota en el plano transversal y en el plano sagital. Físicamente no cuenta con un mecanismo que le indique si ha llegado al límite de su movimiento y corre peligro de forzar alguna pieza o desconectar algún cable ya que su volumen de trabajo no es una esfera, sino un cuarto de esfera y los límites de movimiento se tendrán que implementar por software.
Las conexiones de los motores son de forma serial a otro motor y el módulo de visión se conecta directamente al controlador del robot CM510, donde serán procesados los resultados obtenidos y en base a eso y de acuerdo a los algoritmos de movimiento indicar a los actuadores los movimientos a realizar.

3.2.2 WEBCAM

Aunque el módulo HaViMo es en verdad muy poderoso y resulta de mucha ayuda gracias al pre procesamiento rápido de la información permitiendo una fácil detección de la pelota, el hecho de que se tenga que calibrar cada determinado tiempo hace realmente tedioso el ciclo de pruebas con el robot, también es confuso ya que en ocasiones pierde el objetivo ya sea porque algún valor de la LUT aparece en algún objeto que se encuentra en la imagen de captura incluso aunque tenga un color muy distinto al buscado, o porque las condiciones de iluminación cambiaron.
Para evitar esto se tendría que realizar un algoritmo de calibración que tomara en cuenta las condiciones de iluminación y descartara el posible ruido en la imagen, pero resulta complejo cuando el procesamiento se realiza dentro del dispositivo.
Otra alternativa es probar con otra cámara, se quiso usar una cámara de bajo costo que permitiera realizar todo el procesamiento en la pc para después mandar las instrucciones de movimiento al robot.

FIGURA 14. WEBCAM MICROSOFT VX-800

La Webcam Microsoft Vx-800 cumple con los requisitos impuestos anteriormente ya que es una cámara de muy bajo costo que ofrece imágenes suficientemente claras además de ajustarse automáticamente a las condiciones de luz ofreciendo una mejor calidad de video. La cámara se conecta por medio de USB y es reconocida sin ningún problema por el sistema operativo. Además de ser realmente sencillo obtener los cuadros capturados por medio de un programa en C#.NET como se verá a continuación y enviar la información al robot con comunicación ZigBee como se explicó anteriormente.

3.2.2.1 ANÁLISIS DE LA IMAGEN.

Para lograr un alto desempeño y un desarrollo más rápido del software que controla al robot, se buscó una herramienta sencilla, potente y de código abierto que soportara los algoritmos necesarios para llevar a cabo el correcto seguimiento de la pelota.
AForge.NET "www.aforge.net" es un framework C# diseñado para desarrolladores e investigadores en las áreas de visión por computadora, inteligencia artificial, procesamiento de imágenes, redes neuronales, algoritmos genéticos, aprendizaje de máquina, robótica etc.
El uso de este framework facilita el análisis de las imágenes captadas por la cámara, reduciendo el trabajo a la creación de un algoritmo que haciendo uso de técnicas conocidas logre con la mejor precisión posible identificar la pelota.
La webcam permite también observar en tiempo real lo que el robot está percibiendo y en caso de existir algún conflicto, tener un punto de partida más realista de lo que pasa en el análisis de la imagen.

3.2.2.1.1 FILTRO DE COLOR ROJO.

El primero de una serie de pasos para lograr localizar la pelota al igual que con el módulo HaViMo es un filtro de color rojo que me permita desechar información inútil de la escena.

```
public Bitmap FiltroColorRojo(){
            IFilter filtroColor =
            new ColorFiltering(new IntRange(150, 255), new IntRange(0, 100), new IntRange(0, 200));
            return filtroColor.Apply(imagen);
        }
```

FRAGMENTO DE CODIGO 1. FILTRO DE COLOR ROJO

Este filtro se tiene que aplicar a cada uno de los frames que compone la secuencia de video para descartar de la escena todos los objetos cuya tonalidad es por mucho diferente a la buscada por el robot.
Como se puede ver en la siguiente figura, se han eliminado el fondo y demás objetos innecesarios para dar seguimiento a la pelota, un humano podría descartar fácilmente la pelota de esta imagen, sin embargo para la máquina, el decir dónde se encuentra exactamente la pelota es todavía complicado.
El funcionamiento de este filtro consiste básicamente en recorrer la imagen pixel por pixel y aquellos que no cumplan con los rangos de color especificados son descartados y sustituidos por color negro.
Se puede observar también en la imagen ruido proveniente de otros objetos cuyas tonalidades se encuentran dentro de los rangos especificados, posteriormente se tendrán que eliminar estos pixeles que no guardan relación alguna con el cuerpo de la pelota. También se puede observar un agujero negro en la parte superior de la pelota originado gracias al reflejo de la luz sobre ella, situación que acompañará la imagen durante todo el análisis ya que aunque sin duda se podría rellenar, se trata aquí de realizar el procesamiento en el menor tiempo posible, entonces lo que no es realmente necesario no será tema de discusión.

FIGURA 15. FILTRO DE COLOR ROJO

3.2.2.1.2 FILTRO ESCALA DE GRISES

Para localizar la ubicación de la pelota en la escena definiremos su posición como la región buscada, para encontrar esta región, primero tenemos que aplicar un tratamiento previo a la imagen, la cual podemos ver como un conjunto de tres matrices, la que representa el color rojo, la que representa el color verde y la que representa el color azul, por eso para trabajar con una única matriz obtenemos la imagen en escala de grises.

```
public Bitmap FiltroEscalaGrises() {
        //filtro (BT709)
        Grayscale filterGray = new Grayscale(0.2125, 0.7154, 0.0721);
        return filterGray.Apply(imagen);
       }
```

La función usa el algoritmo ITU-R Recommendation BT.709 para convertir la imagen y el resultado es una imagen de 8 bits.

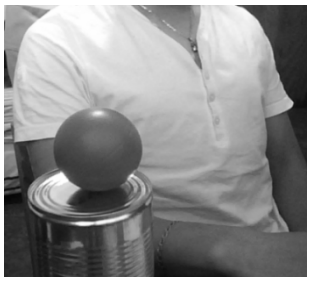


FIGURA 16. IMAGEN EN ESCALA DE GRISES

3.2.2.1.3 UMBRALADO.

Un paso posterior es el de umbralado de la imagen el cual puede considerarse como la separación de los pixeles de una imagen en dos clases, representadas de color blanco y negro, esta técnica me permite separar el fondo de los objetos, entre los que se encuentra la pelota que el robot debe seguir.

FIGURA 17. IMAGEN BINARIZADA

```
public Bitmap FiltroUmbral()
        {
            Threshold filterBN = new Threshold(125);
            return filterBN.Apply(imagen);
        }
```

Donde la función threshold () se define de la forma siguiente.

$$threshold = \begin{cases} 0 \; si \; img(x,y) > 125 \\ 1 \; si \; img(x,y) \leq 125 \end{cases}$$

3.2.2.1.4 EXTRACCIÓN DE REGIONES

La imagen está compuesta por pixeles y los objetos en la imagen se puede decir que forman un conjunto o grupo de estos pixeles, al que se le denomina región. Los pixeles de una misma región comparten siempre alguna característica, la pelota es una región de la imagen así que al encontrar estas regiones sabemos que una de ellas corresponde a la pelota. Para encontrar las diferentes regiones de una imagen en 8 bits usamos el siguiente código.

```
BlobCounter blobCounter = new BlobCounter();
blobCounter.ProcessImage(Imagen);
Rectangle[] rects = blobCounter1.GetObjectsRectangles();
```

El framework de Aforge.Net ofrece un conjunto de métodos para el procesamiento de regiones, buscando y separando objetos para su uso posterior. La clase BlobCounter entre otras cosas extrae las regiones de una imagen usando el algoritmo de etiquetado de componentes conexas. Después estas regiones son almacenadas en un arreglo de rectángulos para facilitar su tratamiento.

FIGURA 18. EXTRACCIÓN DE REGIONES

Una vez que se cuenta con estas herramientas, podemos pensar en darle forma al algoritmo final que permita a la webcam realizar el mismo trabajo que el módulo HaViMo2.0.

3.2.2.1.5 ALGORITMO FINAL DE VISIÓN

Al momento de pensar en un algoritmo que cumpla con el objetivo de identificar la pelota para que el robot pueda darle seguimiento y lo realice de la manera más rápida posible, no tiene por qué ser el más completo de los algoritmos ni mucho menos cubrir todos los aspectos, incluso debido a que se trata con un ambiente altamente dinámico se pueden permitir ciertos errores, en la parte de resultados se explora más a fondo el camino para la elección de este algoritmo que es la versión que finalmente cumple con las expectativas buscadas y lo hace de una manera eficiente. Como podemos observar el algoritmo en su estado más general resulta muy sencillo.

1. Captura de la imagen
2. Aplicamos el filtro de color rojo
3. Cambiamos la imagen a escala de grises
4. Binarizamos la imagen
5. Obtenemos las regiones con el algoritmo de etiquetado de componentes conexas
6. Nos quedamos con la región de mayor tamaño.
7. Suponemos esa región como la pelota y obtenemos las coordenadas (x, y) que representen su centro.

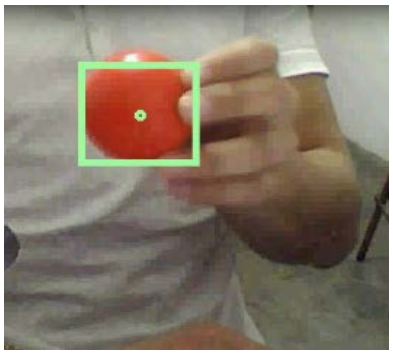


FIGURA 19. RESULTADO DEL ALGORITMO FINAL

3.3 CONTROL

Para lograr la estabilización y el caminado de un robot humanoide se necesita un análisis matemático preciso de la geometría y la física implicadas en su diseño. Además de un elevado número de experimentos relacionados que demandan una gran cantidad de tiempo. ROBOTIS proporciona con el robot un entorno de programación sencillo que ayuda en esta tarea y la vuelve relativamente sencilla. Además de proporcionar códigos de ejemplo muy útiles al momento de desarrollar y crear diseños propios. También cuenta con un sistema de programación en el cual desde el código se mandan llamar una serie de movimientos ya calculados tomando en cuenta el diseño del robot que en conjunto ayudan a realizar tareas complejas como por ejemplo que el robot pueda caminar sin tropezar, y en caso de tener alguna caída pueda levantarse sin ningún problema. Estos movimientos son llamados "motions" y desde la página se proporcionan los motions necesarios para que el robot pueda incluso patear la pelota.

Un motion está integrado por una serie de giros de los motores, estos pueden ser programados ya sea de modo numérico o arrastrando las partes de un modelo tridimensional del robot, los modelos tridimensionales son proporcionados por ROBOTIS, y se da la opción a crear modelos personalizados.

Estos motions se mandan llamar desde un lenguaje de programación propio del robot y en especial para programar el controlador CM510, este lenguaje tiene instrucciones de control del flujo del programa, ciclos, asignaciones, operaciones lógicas y aritméticas además de funciones especiales para trabajar con los sensores, acceder a los registros del procesador y trabajar con los motores dynamixel directamente.

El robot puede ser programado también con lenguaje C contando aún con un amplio conjunto de instrucciones proporcionadas por el fabricante para un desarrollo más rápido de software de control.

Si esto no es suficiente, se puede generalizar todavía más y extender los lenguajes de programación a cualquiera que implemente la comunicación serial, aunque de esta manera se tendrá que implementar cada protocolo y funciones que se necesiten para interactuar con los motores, sensores y demás prestaciones del controlador CM510.

Para el desarrollo de la Tesis se utilizaron las motions en conjunto con el entorno de desarrollo RoboPlus que permite el lenguaje de programación propio del BIOLOID, donde a cada archivo se le conoce como TASK.

Se propuso resolver el objetivo planteado en tres etapas, las cuales secuencialmente llevarán a la solución final. Además de hacer el resultado modular y flexible a cambios en las herramientas o algoritmos utilizados.

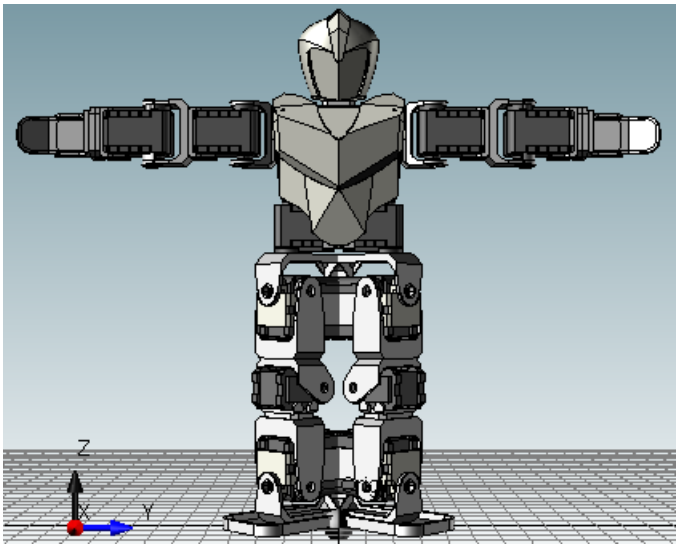


FIGURA 20. MODELO TRIDIMENSIONAL. ROBOPLUS MOTION

3.3.1 ETAPA NÚMERO 1. SEGUIMIENTO A LA PELOTA CON EL CUELLO DEL ROBOT.

Se parte de que el montaje hecho en la parte superior del cuello del robot permite movimiento en el plano transversal y movimiento en el plano sagital.

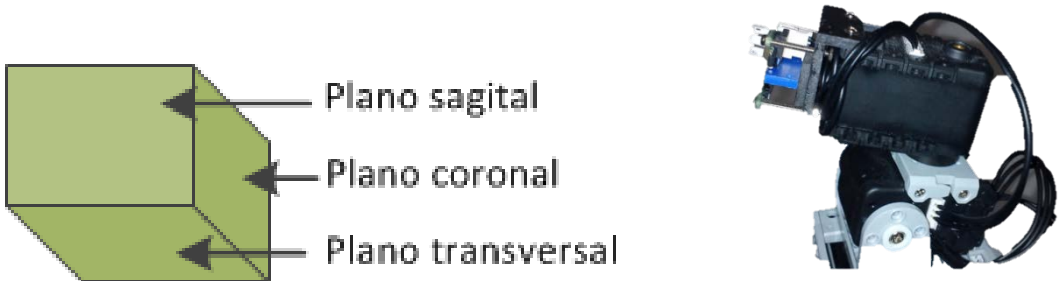


FIGURA 21. MOVIMIENTO DEL CUELLO DEL ROBOT

Los dos motores que forman el cuello tienen límites físicos de movimiento, es decir que el robot no puede por ejemplo voltear muy arriba o girar completamente la cabeza en el plano transversal. El rango de movimiento de los motores esta expresado en grados y está acotado de acuerdo a los límites establecidos por su configuración. En el caso del motor que otorga al cuello movimiento en el plano sagital su rango de movimiento va de los 500° a los 750° y el motor que mueve el cuello del robot en el plano transversal va de los 200° a los 800°.

Es bien sabido que para dar solución a un problema o comportamiento del mundo real lo primero que se necesita hacer es crear un modelo que simule este comportamiento y con el cual se pueda trabajar, muchas veces estos modelos son representados con sistemas de ecuaciones continuas o discretas según el caso.

El cuello del robot es el primer comportamiento que se desea controlar y de manera general se puede decir que a través de una serie de entradas como la posición de la pelota, se calculan los movimientos de los motores para que el cuello gire hacia su objetivo.

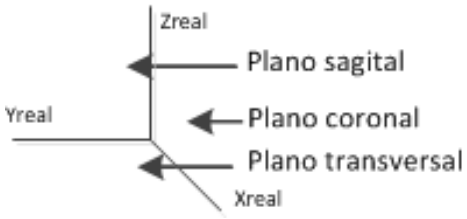


FIGURA 22. PLANOS DE MOVIMIENTO DEL ROBOT

El objetivo concreto de esta etapa es lograr que el robot siga con la mirada la pelota, es decir que mueva el cuello de tal forma que cuando la pelota se encuentre en movimiento siempre aparezca lo más cerca posible al centro de la imagen.

Para identificar fácilmente los planos y movimientos con los cuales estamos trabajando, se nombraron los ejes de los planos mencionados anteriormente, de tal forma que el plano transversal es el plano formado por los ejes (Xreal, Yreal), el plano sagital está formado por los ejes (Yreal, Zreal), y el plano coronal por los ejes (Xreal, Zreal), de tal suerte que los motores que dan movimiento al cuello del robot en los planos transversal y sagital se pueden nombrar como MXY y MYZ respectivamente.

La imagen a su vez es una función $img = f(x, y) \mid x, y \in \mathbb{Z}$ donde el rango de x y y va de 0 hasta el ancho y alto de la imagen respectivamente.

Dadas estas consideraciones y tomando en cuenta el algoritmo de visión previamente descrito se pueden determinar cuáles son las entradas y salidas que determinan el comportamiento del cuello del robot, que es el que se desea controlar.

Entradas	Salidas
Coordenadas cartesianas del centro de la pelota dentro de la imagen, resultado del algoritmo de visión.	Nuevas posiciones de los motores MXY y MYZ
Tamaño de la imagen en pixeles	Velocidad de los motores MXY y MXZ
Posición actual de los motores MXY y MYZ	
Velocidad actual de los motores MXY y MYZ	

TABLA 1. ENTRADAS Y SALIDAS PRIMERA ETAPA

3.3.1.1 MODELO DEL SISTEMA

Se necesita asociar el movimiento del motor MXY con la variación en las coordenadas del eje x de la pelota y el movimiento del motor MYZ con la variación en las coordenadas del eje y de la pelota. La ecuación para el eje X, será exactamente la misma que para el eje Y, por tanto basta con hallar una ecuación que satisfaga el objetivo para el movimiento de la pelota en el eje x, y posteriormente aplicar esa misma ecuación al otro eje.

Debido a este enfoque, para simplificar el análisis suponemos $k = x, y$, un aspecto importante a conocer y tomar en cuenta es la velocidad de la pelota, el movimiento del cuello no se puede dar a una misma velocidad porque entonces si la pelota se mueve más rápido que el cuello entonces sería imposible darle seguimiento y al contrario si el cuello es mucho más rápido que la pelota entonces también se perdería el objetivo. Por lo tanto, este debe ser un movimiento sincronizado. El movimiento de la pelota es el que dará pauta para identificar este aspecto. La primer referencia que se tiene, es el tamaño de la imagen siendo así N_k el tamaño en pixeles de la imagen con $k = x$ para el ancho y con $k = y$ para el alto. Si el objetivo es que la pelota se mantenga en el centro de la imagen, entonces la coordenada de referencia es $\frac{N_k}{2}$.

Actualmente muchos de los robots existentes pueden por ejemplo, medir la distancia entre ellos y algún objeto con un rango mínimo de error, existen también técnicas para conocer esta información con el análisis de imágenes, incluso si el objeto esta en movimiento se podría determinar su velocidad en m/s, tarea que por su puesto tiene mayor grado de dificultad. La lista de cálculos puede continuar, sin embargo en el presente trabajo el enfoque con el que se manejan las cosas es distinto.

Como humanos, nos es imposible determinar la distancia o velocidad de un objeto de manera exacta, podemos aproximar ese resultado pero generalmente el error del cálculo realizado será muy grande en comparación con el de una máquina. No obstante el ser humano es muchísimo más eficiente para tareas más complejas que involucran el movimiento de varios músculos, decisiones basadas en la visión, tacto, oído etc. Y este movimiento es coordinado casi instantáneamente. Ejemplos de esto son, el atrapar un balón de futbol americano, anotar un gol de chilena en un partido de futbol, evitar un golpe de un boxeador. Son un sinfín de ejemplos como estos en donde el hombre reacciona sin conocer los valores exactos de todas las variables involucradas.

Uno de los principios fundamentales sobre los que se basa la tesis es:

"EL ROBOT NO NECESITA CONOCER LOS VALORES EXACTOS DE LAS VARIABLES FÍSICAS INVOLUCRADAS EN EL PROBLEMA, ENTONCES DEBERÁ RESOLVERLO EN BASE A LA RELACIÓN ENTRE ESTAS VARIABLES Y LA INFORMACIÓN DERIVADA DE SU CONOCIMIENTO DEL MUNDO OBTENIDO CON LOS SENSORES."

Así pues se puede encontrar una relación entre esta variable física que es la velocidad a la cual se desplaza la pelota y la información que obtenemos con la cámara.
Se tiene que las coordenadas de referencia serán las del centro de la imagen, en cada captura de la cámara, la pelota aparecerá desplazada un cierto número de pixeles y se deduce que el máximo avance que se puede percibir con la cámara es de $\frac{N_k}{2}$ pixeles.
Se necesita una ecuación del tipo $k_{i+1} = f(k_{i,}Ui)$, donde K representa los estados actual y futuros del robot y U describe lo que el robot estaba haciendo en el instante i. El objetivo es diseñar el controlador U de modo que se minimice el desplazamiento al mínimo para cualquiera de los ejes.
El objetivo es mantener siempre la pelota en el centro de la imagen y a partir de esta idea se analiza la escena. Cualquier avance mayor a $\frac{N_k}{2}$ pixeles entre dos capturas de la cámara, será imposible de percibir, es decir que si la pelota lleva una velocidad que exceda este avance no puede ser cuantificada, por lo tanto la velocidad máxima que se puede percibir es la que genera un avance de $\frac{N_k}{2}$ pixeles, cualquier velocidad mayor será un caso especial que se analizará posteriormente, de esta forma es fácil determinar cuál es la velocidad a la que se desplaza la pelota en cualquier instante i.
El cuello tiene una velocidad máxima de movimiento que se traduce en la velocidad máxima con la cuál puede seguir el movimiento de la pelota, sea esta velocidad $vmaxp$.
Sea $\frac{N_k}{2}$ la velocidad máxima a la que se puede percibir el desplazamiento de la pelota la cual será también la velocidad máxima a la cual se moverá el cuello del robot entonces se puede decir que:

$$vmaxp = \frac{N_k}{2} \tag{1}$$

La velocidad de los motores originada por un desplazamiento Δ_k en cualquiera de los ejes se determina entonces con la siguiente ecuación usando una regla de tres.

$$velp = \frac{\Delta_k \cdot vmaxp \cdot 2}{N_k} \tag{2}$$

El desplazamiento Δ_k es obtenido tomando en cuenta la posición actual de la pelota p_k.

$$\Delta_k = p_k - \frac{N_k}{2} \tag{3}$$

Si $k = x$ y el desplazamiento es positivo entonces el movimiento fue a la izquierda, pero es negativo, entonces el desplazamiento fue a la derecha, de igual forma si $k = y$ y el resultado fue positivo entonces el desplazamiento fue hacia abajo pero si fue negativo, entonces el desplazamiento fue hacia arriba. Es decir que el signo indica la dirección del movimiento que debe realizar el cuello y el valor indica la magnitud de la velocidad con la que se debe mover.
Con estos resultados se puede plantear ya la ecuación que modela el comportamiento de la pelota y rige también el comportamiento del cuello.

$$Velp_{k_{i+1}} = f\left(Velp_{k_i}, U(\Delta_{k_i})\right) = Velp_{k_i} + U(\Delta_{k_i}) \tag{4}$$

El único problema aquí es que la velocidad de la pelota está dada en pixeles por captura y la velocidad de los motores está dada en m/s, por lo tanto necesitamos hacer un equivalente.
Se calcularon experimentalmente el número de grados que tienen que girar los motores MXY y MYZ para desplazar la imagen $\frac{N_x}{2}$ y $\frac{N_y}{2}$ pixeles respectivamente, a este par de ángulos se les denomina φ_k. Entonces se puede hacer una relación entre la velocidad de la pelota y los grados que tienen que girar los motores para alcanzar la pelota.

$$\beta_k = \frac{\Delta_k \cdot \varphi_k \cdot 2}{N_k} \tag{5}$$

Donde β_k es el número de grados que debe girar el motor para da alcance a la pelota en cada captura. Una vez obtenida esta ecuación ahora si se tiene ya el modelo del comportamiento del cuello del robot.

$$Pos_{k_{i+1}} = f\left(Pos_{k_i}, U(\Delta_{k_i})\right) = Pos_{k_i} + U(\Delta_{k_i}) \tag{6}$$

Con la variable de control lo que se busca es siempre minimizar el error.

$$e = |\Delta_k| \tag{7}$$

Para el diseño del controlador, y después de múltiples experimentos que se mencionarán en el capítulo de resultados experimentales, se encontró que el controlador más adecuado es uno del tipo P, ya que es lo bastante rápido y eficiente, puede no ser tan preciso como un PID pero es suficiente y funciona mucho mejor así.

$$U\left(\Delta_{k_i}\right) = K_p e \tag{8}$$

Para lograr la estabilidad del cuello se agregará una variable que sirva para ajustar la sensibilidad de este al movimiento de la pelota.

$$\Delta p_k = \left|p_{k_i} - p_{k_{i-1}}\right| \tag{9}$$

Una vez hecho esto se propone la siguiente K_p para el controlador.

$$K_p = \begin{cases} 0 \; si \; \Delta p_k < S \\ \dfrac{\beta_k}{|\Delta_k|} \; si \; \Delta p_k > S \end{cases} \tag{10}$$

Así se cumple el seguimiento de la pelota, además de combatir la incertidumbre por medio de la retroalimentación en los parámetros del modelo.
Debido a mediciones experimentales en los pasos de los motores que se discutirán en el capítulo de resultados, se determinó una variable de sensibilidad $S = 10$, que da una margen de movimiento de 10 pixeles a la pelota donde el cuello no se moverá para perseguirla, esto elimina las oscilaciones que tiene la pelota mientras se estabiliza en algún lugar lo que hace el modelo más robusto, además de que no depende de ningún valor de referencia variable.

3.3.2 ETAPA NÚMERO 2. PATEO DE LA PELOTA SIN MOVIMIENTO.

En esta etapa se plantea ya el movimiento completo del robot que tendrá como misión dirigirse de un ponto A hasta un punto B donde se encuentra la pelota y patearla. Para esto se propone un margen de error E que será calculado experimentalmente.

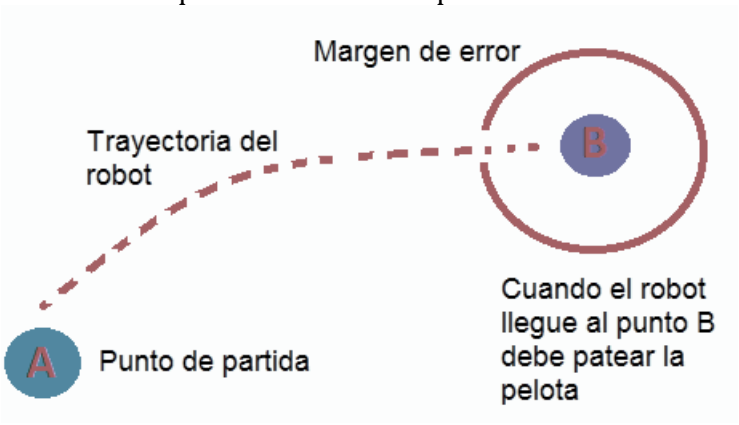


FIGURA 23. ETAPA NUMERO 2

En esta etapa se deben tener en cuenta algunas consideraciones importantes. La más importante es el hecho de que se reutilizó código proporcionado por ROBOTIS para el caminado del robot. Entonces ya se tienen muchas funciones que apoyan en el desarrollo del algoritmo y se utilizarán íntegramente como fueron proporcionadas, para simplificar el modelado, se escogieron solo algunas de las funciones proporcionadas, lista que se proporciona a continuación.

- Camina: Con esta función el robot camina hacia adelante 500ms una distancia D.
- Gira derecha: Con esta función el robot gira hacia la derecha un ángulo θ sobre su propio eje.
- Gira izquierda: Con esta función el robot gira hacia la Izquierda un ángulo θ sobre su propio eje.
- Patada derecha: Con esta función el robot patea con el pie derecha.
- Patada Izquierda: Con esta función el robot patea con el pie izquierdo.

Cada una de estas funciones ha sido programada por medio de motions, que se mandan a llamar y que están calculados exactamente para que el robot conserve el equilibrio.

3.3.2.1 ALGORITMO BASADO EN COMPORTAMIENTOS

En una máquina tragamonedas es factible planear las acciones que debe llevar a bajo todas las situaciones que se podrían presentar, sin embargo en un partido de futbol el entorno es altamente dinámico y sería muy complicado tratar de planear todos los movimientos que debe efectuar un jugador para cada situación de manera que se obtenga el mejor resultado. Uno de los enfoques usados en robótica para lidiar con esta clase de problemas, es diseñar un conjunto de controladores simples que ataquen un aspecto muy particular del problema, y usarlos de acuerdo a las condiciones del entorno o a los parámetros que se perciban de él. Al tratar de realizar un modelo como el del presente trabajo se deben tomar en cuenta diferentes

etapas que lo conforman por ejemplo la etapa sensorial, la representación de esos parámetros en el mundo del robot, la planificación y el control de los actuadores. La interacción de estas etapas en ocasiones se toma a la ligera y causa poca robustez en el sistema, además no siempre es fácil diseñar un controlador o la interacción entre este y las demás etapas del sistema, por ejemplo en la parte del cuello donde se relacionó el movimiento de la pelota con el movimiento de los motores y se diseñó un controlador que redujera el error al mínimo. En ese caso solo se involucra a un motor por ecuación pero si la complejidad del sistema crece las interacciones entre los controladores y las etapas son más complejas, en este caso al hablar del caminado del robot se toman en cuenta 18 motores la cámara, el giroscopio y la física asociada por lo que realizar un modelo que abarque estos parámetros, que sea robusto, suave, estable y cuyos cálculos sean simples para que se ejecuten con la mayor rapidez posible resulta demasiado complejo. Por lo que se utiliza la robótica basada en comportamientos para el control del robot a partir de esta etapa.
En la robótica basada en comportamientos, cada comportamiento se representa como un autómata de estados finitos y el algoritmo general es un conmutador entre estos comportamientos.

3.3.2.2 AUTOMATA DEL MOVIMIENTO DE LA PELOTA

El movimiento de la pelota representado como un autómata de estados finitos es:

$$Pelota = (Q_p, \Sigma_p, q_{p_0}, \delta_p, q_{p_3})$$

Dónde:

$Qp = \{$
q_{p_0} "$Buscando\ pelota$: $en\ este\ estado\ se\ buscan\ en\ la\ imágen\ obtenida\ las\ coordenadas\ (x, y)\ de\ la\ pelota$",
q_{p_1} " Controlando: Se encuentra cual el valor de la variable de la variable de control $U(\Delta_{k_i})$,
q_{p_2} "$Prediciendo\ posición$: $Se\ encuentra\ U(\Delta_{k_{i-1}})\ y\ se\ toma\ como\ variable\ de\ control$",
q_{p_3} "$Ubicando\ pelota$: $Se\ determina\ si\ el\ error\ es\ suficientemente\ pequeño\ como\ para\ considerar\ que\ la\ pelota\ ha\ sido\ ubicada\ en\ las\ cercanias\ del\ centro\ de\ la\ imágen\ y\ de\ esta\ manera\ poder\ realizar\ algún\ movimiento$.
$\}$

$\Sigma_p = \{0,1\}$

$q_p \epsilon Q_p$	$\sigma \epsilon \Sigma$		$\delta_p(q_p, \sigma) \epsilon Q_p$
$\boldsymbol{q_{p_0}}$	1	"pelota encontrada"	q_{p_1}
$\boldsymbol{q_{p_0}}$	0	"pelota no encontrada"	q_{p_2}
$\boldsymbol{q_{p_1}}$	1	"$U(\Delta_{k_i})$ calculada"	q_{p_3}
$\boldsymbol{q_{p_2}}$	1	"$U(\Delta_{k_{i-1}})$ calculada"	q_{p_3}

TABLA 2. TRANSICIÓN DE ESTADOS COMPORTAMIENTO PELOTA

3.3.2.3 AUTOMATA DEL MOVIMIENTO DEL CUELLO

El movimiento del cuello representado como un autómata de estados finitos es:

$$Cuello = (Q_c, \{0,1\}, q_{c_0}, \delta_c, \{q_{c_4}, q_{c_5}\})$$

Dónde:

$Q_c = \{$
q_{c_0} "*Determinando movimiento*: *se determina si se debe mover el motor MXY o el motor MYZ*",
q_{c_1} "*Revisando posición MXY*: *Checa si la posición a la que se pretende mover motor MXY esta dentro de su rango de movimiento*"
q_{c_2} "*Revisando posición MYZ*: *Checa si la posición a la que se pretende mover motor MYZ esta dentro de su rango de movimiento*"
q_{c_3} "*Enviando alerta*: *Manda una alerta auditiva*"
q_{c_4} "*Moviendo MXY*: *Mueve el motor MXY* $Pos_{x_{i+1}}$"
q_{c_5} "*Moviendo MYZ*: *Mueve el motor MXY* $Pos_{y_{i+1}}$"
$\}$

$q_c \epsilon Q_c$	$\{0,1\}$	$\delta_c(q_c, \{0,1\}) \epsilon Q_c$
$\boldsymbol{q_{c_0}}$	0 "mover MXY"	q_{c_1}
$\boldsymbol{q_{c_0}}$	1 "mover MYZ"	q_{c_2}
$\boldsymbol{q_{c_1}}$	1 "dentro de rango"	q_{c_4}
$\boldsymbol{q_{c_1}}$	0 "fuera de rango"	q_{c_3}
$\boldsymbol{q_{c_2}}$	1 "dentro de rango"	q_{c_5}
$\boldsymbol{q_{c_{12}}}$	0 "fuera de rango"	q_{c_3}

TABLA 3. TRANSICIÓN DE ESTADOS MOVIMIENTO DEL CUELLO

3.3.2.4 ANÁLISIS DEL MOVIMIENTO DEL ROBOT

La idea principal del algoritmo es hacer que el robot voltee a ver la pelota, una vez que la ubique, debe girar el cuerpo hasta que este frente a ella, con el cuerpo en dirección a la pelota debe avanzar hacia ella y patearla.
Es realmente un algoritmo muy sencillo pero en el mundo real tiene grandes fallas debido a consideraciones físicas que se deben tonar en cuenta como la fricción de los pies del robot con el piso, la imprecisión de los movimientos y mediciones, los tiempos de retardo etc.
Como se discutirá en el capítulo de resultados, se encontró con que el caminado del robot no dibuja una línea recta, más bien una línea curva inclinada a la derecha, aspecto que hace replantear inmediatamente el algoritmo para que el robot reacomode su ruta cada vez que avanza de manera que llegue lo más cerca posible a la pelota para patearla.
Como se mencionó anteriormente de acuerdo al enfoque de la tesis no se pretende que el algoritmo dependa del avance preciso una cierta distancia, o que gire cierto número de grados cada vez de manera exacta. Por el comportamiento asociado debe estar libre de estas mediciones, aunque fueron determinadas experimentalmente para la simulación, no son necesarias para el control del robot.
Entonces se debe partir de relaciones conocidas con las cuales retroalimentar el modelo y lograr un buen diseño de controlador. En esta ocasión el error estará dado por la orientación del robot con respecto al ángulo del cuello, es decir que el robot debe estar volteando hacia el frente siempre, para que al caminar se dirija hacia la pelota. Además se tomará en cuenta también el ángulo del motor MYZ para saber cuándo el robot ha llegado a su objetivo.

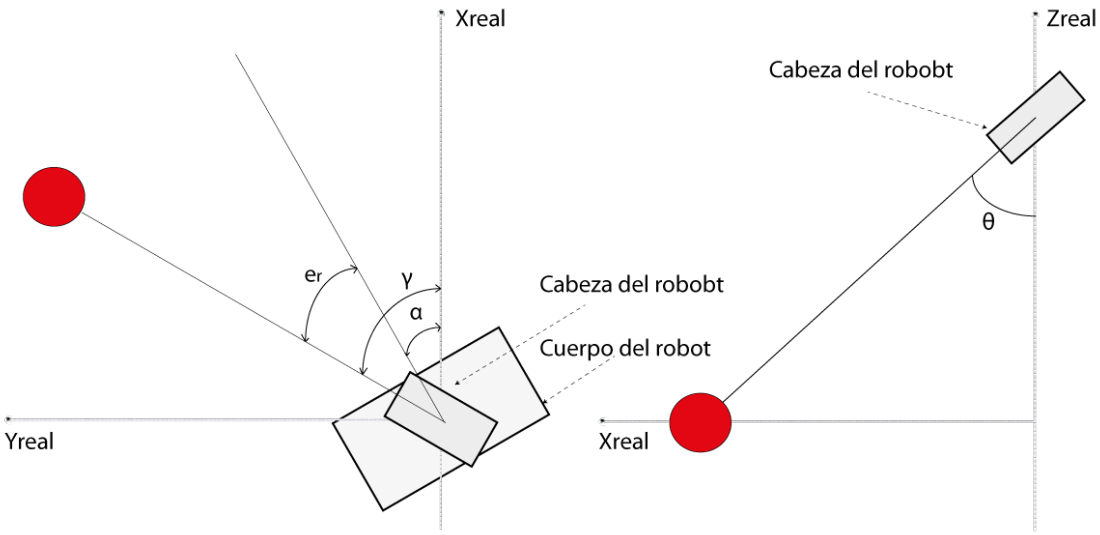


FIGURA 24. ANALISIS DEL ROBOT HACIA PELOTA FIJA

Entonces tenemos α que es el ángulo que determina la orientación del robot, para la simulación que se discutirá en el capítulo de resultados este será de mucha utilidad, pero como tal para el algoritmo realmente no importa la orientación que tenga el robot.
También está el ángulo γ que es la orientación de la cabeza del robot es un punto muy importante ya que se parte de la idea de que el robot siempre está viendo la pelota, dados los excelentes resultados en la primera etapa se puede hacer esta aseveración.
El último ángulo y más importante para esta etapa es el ángulo de error $e_r = \gamma - \alpha$, que determina cuanto debe girar el robot para quedar de frente a la pelota y poder patearla. Como los motores utilizados en la construcción del robot son servomotores la posición actual de cada uno de estos puede ser determinada fácilmente por lo tanto determinar el ángulo e_r se convierte en tarea sencilla.

$$e_r = 500 - mxy \tag{11}$$

Donde mxy es el ángulo actual del motor MXY, y recordando que el rango de movimiento de este motor es de 200 a 800 grados tomando 500 como referencia cuando el robot está volteando hacia adelante, es decir cuando $\gamma = \alpha$. También este error determina con su signo la dirección de giro del robot.
Cuando se indica al robot que gire a la izquierda o derecha este gira cierta cantidad de grados medidos experimentalmente, valor que no puede ser cambiado, por lo tanto dependiendo la posición de la pelota el error e_r puede o no ser reducido a 0 entonces se tienen que definir dos variables de sensibilidad cuya comparación con el error me generen una condición de paro e indiquen que el robot esta de manera aproximada frente al robot y este pueda ya sea patear la pelota o avanzar. Estas variables s_{gmin} y s_{gmax} cuyos valores fueron determinados experimentalmente y se discutirá de ellos más adelante.
Por último se tiene el ángulo θ que es el ángulo de orientación de la cabeza del robot cuando está volteando hacia la pelota con respecto al eje z_{real}. Este ángulo determina el avance del robot hacia la pelota e indica cuando el robot está en posición de patearla. El valor de θ es también extrapolado al ángulo actual del motor MYZ, el valor de referencia θ_{ref} para saber cuándo el robot ha llegado a su objetivo y puede patear la pelota fue determinado experimentalmente.
A pesar que podemos calcular otras variables del mundo real como la distancia aproximada entre el robot y la pelota por ejemplo usando geometría, estos valores ya mencionados son más que suficientes para la implementación del algoritmo además de ser precisos y fáciles de conocer lo que se traduce en muy pocos cálculos del procesador.

3.3.2.4.1 COMPENSANDO EL CAMINADO DEL ROBOT

Durante la realización del algoritmo necesario para esta implementación surgieron varios problemas relacionados con la perdida de la pelota, a los cuales se les fue dando solución a través de la mejora del algoritmo pero otros están relacionados con los movimientos propios del robot, uno de ellos es lo que sucede cuando el robot camina, el robot trata de alcanzar la pelota avanzando pero al hacerlo ocurre un desfase obvio que provoca la pérdida del objetivo.

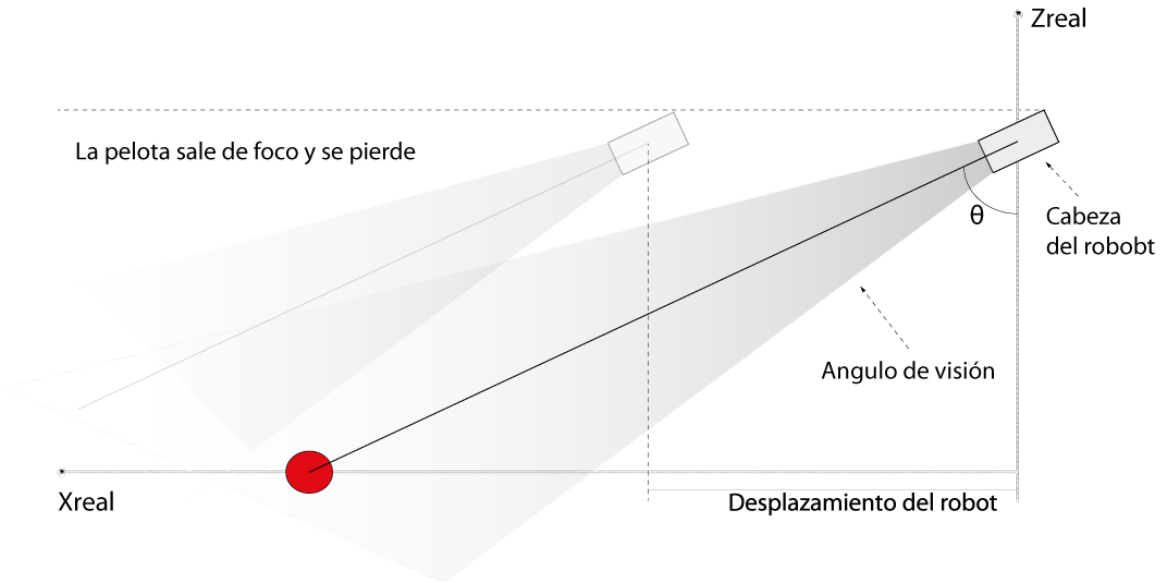


FIGURA 25. DESFACE EN EL CAMINADO

En esta etapa la pelota se encuentra fija por lo tanto basta con bajar la cabeza progresivamente hasta volver a encontrarla.

3.3.2.4.2 COMPENSANDO LOS GIROS DEL ROBOT

Al igual que sucede con el caminado del robot, al momento de girar el cuerpo para quedar de frente a la pelota, la cabeza queda completamente volteada y se pierde la pelota de vista, por lo tanto debe existir una compensación. La solución al igual que para el caminado podría ser regresar la cabeza hasta que encuentre la pelota de nuevo, y aunque esta solución no es mala si lleva un poco más de tiempo que la solución utilizada, que es más práctica y eficiente.

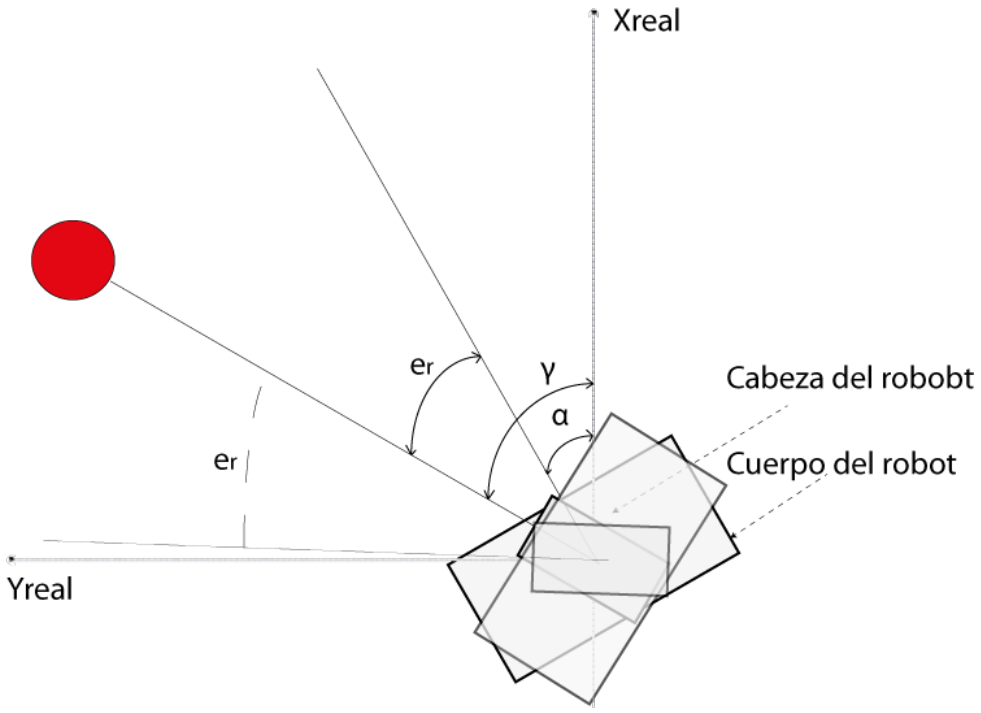


FIGURA 26. DESFACE EN EL GIRO

Para resolver esta situación la solución es poner el motor MXY con un ángulo de 500 grados donde $e_r = 0$, es decir que el ángulo de orientación del robot es el mismo que el ángulo de orientación de la cabeza, esto debido a que se da por hecho que el giro fue para quedar frente a la pelota o lo más cercanamente posible.

En las pruebas, esta solución resulta ser muy efectiva, tanto para esta etapa como para la siguiente. Es importante recordar que mientras más simples sean las fórmulas mejor es el desempeño del robot, porque el tiempo que tarda en realizar los cálculos necesarios de control y ejecutar la acción deseada, es el tiempo que toma para analizar la siguiente captura.

Una ventaja del diseño de esta arquitectura es que el procesamiento de la imagen y los cálculos de control se realizan en procesadores diferentes tanto para el caso de la webcam que utiliza la PC para realizar el procesamiento de la imagen como para la cámara HaViMo que realiza el proceso internamente, aunque el robot no necesita acceso a la ubicación de la pelota hasta que termina su movimiento y en ese lapso de tiempo se pueden perder más de 10 fotogramas por lo tanto mientras más rápidos sean esos cálculos mejor se aprovecha esta información.

3.3.2.5 AUTÓMATA DEL MOVIMIENTO DEL ROBOT PELOTA QUIETA

El movimiento del robot representado como un autómata de estados finitos es:

$$Robot = (Q_r, \{0,1\}, q_{r_0}, \delta_r, \{q_{r_{12}}, q_{r_{13}}\})$$

Dónde:

$Q_c = \{$
q_{r_0} "$Calculando\ el\ error\ e_r$",
q_{r_1} "$Calculando\ Distancia\ y\ orientación$: $se\ calcula\ si\ el\ robot\ esta\ en\ posición\ de\ patear\ la\ pelota\ o\ ejecutar\ algún\ movimiento$
$(\theta \geq \theta_{ref}\ ^\ e_r \leq s_{min}\ ^\ e_r \geq -s_{min}) \rightarrow pateala\ pelota$
$(\theta < \theta_{ref} \mid e_r > s_{min} \mid e_r < -s_{min}) \rightarrow muevete$",
q_{r_2} "$Pateando\ pelota$: $el\ robot\ patea\ la\ pelota\ con\ la\ pierna\ derecha\ si\ e_r \geq 0\ \ y\ patea\ la\ pelota\ con\ la\ pierna\ izquierda\ si\ e_r < 0$",
q_{r_3} "Calculando distancia: verifica si el robot esta ya frente a la pelota $\theta \geq \theta_{ref}$",
q_{r_4} "$Caminando$: $el\ robot\ avanza\ hacia\ adelante\ y\ compensa\ el\ movimiento$",
q_{r_5} "$Resolviendo\ dirección$: $resuelve\ si\ el\ robot\ debe\ girar\ a\ la\ izquierda\ o\ a\ la\ derecha\ e_r \geq 0 \rightarrow derecha, e_r < 0\ \rightarrow izquierda$",
q_{r_6} "$Calculando\ magnitud$: $calcula\ la\ magnitud\ del\ giro\ derecho\ e_r \geq s_{max}$",
q_{r_7} "$Calculando\ magnitud$: $calcula\ la\ magnitud\ del\ giro\ izquierdo\ e_r \leq s_{max}$",
q_{r_8} "$Girando\ a\ la\ derecha$: $gira\ el\ robot\ hacia\ la\ derecha$",
q_{r_9} "$Girando\ mucho\ a\ la\ derecha$: $el\ robot\ gira\ a\ la\ derecha\ dos\ veces$",
$q_{r_{10}}$ "$Girando\ a\ la\ izquierda$: $gira\ el\ robot\ hacia\ la\ izquierda$",
$q_{r_{11}}$ "$Girando\ mucho\ a\ la\ izquierda$: $el\ robot\ gira\ a\ la\ izquierda\ dos\ veces$",
$q_{r_{12}}$ "Fin: $El\ robot\ ha\ cumplido\ su\ objetivo$",
$q_{r_{13}}$ "$Continuando$: $El\ robot\ se\ ha\ movido\ y\ esta\ listo\ para\ lo\ siguiente$"
}

$q_r \epsilon Q_r$	$\{0,1\}$	$\delta_r(q_r, \{0,1\}) \epsilon Q_r$
$\boldsymbol{q_{c_0}}$	1	q_{c_1}
$\boldsymbol{q_{c_1}}$	1 "patea"	q_{c_2}
$\boldsymbol{q_{c_1}}$	0 "muévete"	q_{c_3}
$\boldsymbol{q_{c_2}}$	1	$q_{c_{12}}$
$\boldsymbol{q_{c_3}}$	1 "camina"	q_{c_4}
$\boldsymbol{q_{c_3}}$	0 "gira"	q_{c_5}
$\boldsymbol{q_{c_4}}$	1	$q_{c_{13}}$
$\boldsymbol{q_{c_5}}$	1 "izquierda"	q_{c_6}
$\boldsymbol{q_{c_5}}$	0 "derecha"	q_{c_7}
$\boldsymbol{q_{c_6}}$	1	q_{c_8}
$\boldsymbol{q_{c_6}}$	0	q_{c_9}

$q_r \epsilon Q_r$	$\{0,1\}$	$\delta_r(q_r, \{0,1\}) \epsilon Q_r$
$\boldsymbol{q_{c_7}}$	1	$q_{c_{10}}$
$\boldsymbol{q_{c_7}}$	0	$q_{c_{11}}$
$\boldsymbol{q_{c_8}}$	1	$q_{c_{13}}$
$\boldsymbol{q_{c_9}}$	1	$q_{c_{13}}$
$\boldsymbol{q_{c_{10}}}$	1	$q_{c_{13}}$
$\boldsymbol{q_{c_{11}}}$	1	$q_{c_{13}}$

TABLA 4. TRANSICIONES DEL MOVIMIENTO DEL ROBOT

Una vez teniendo los comportamientos se puede armar la arquitectura de control, que es un diagrama sencillo y fácil de entender.

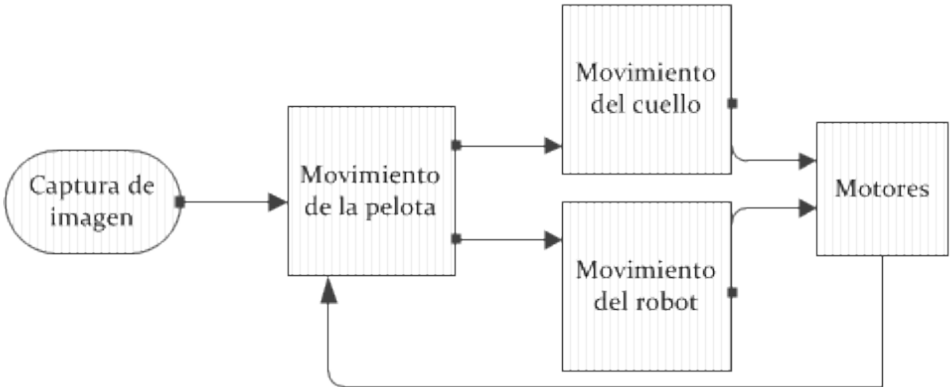


FIGURA 27. ARQUITECTURA DE CONTROL

3.3.3 ETAPA NÚMERO 3. PATEO DE LA PELOTA EN MOVIMIENTO.

El objetivo en esta etapa es que el robot intercepte la pelota en movimiento cuando esta se dirija hacia su horizonte. Esta será la etapa final y con ella se cubre por completo el objetivo. Para cumplir con su cometido el robot dará pasos horizontales, tantos como sean necesarios para patear la pelota con éxito.
En esta etapa se hará uso de funciones adicionales que también son proporcionadas por ROBOTIS para que el robot de un paso lateral a la izquierda y a la derecha. Estas son de apoyo para centrar los esfuerzos en el desarrollo del algoritmo de control o en este caso del o de los comportamientos necesarios.

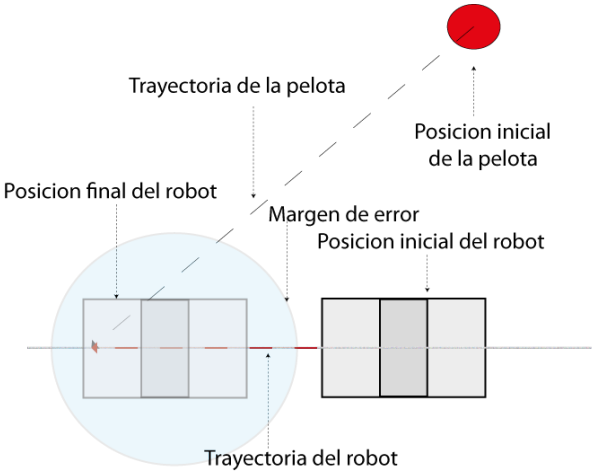


FIGURA 28. ANALISIS ETAPA 3

En esta etapa el autómata que ubican el movimiento de la pelota y el autómata del movimiento del cuello son los mismos, incluso la arquitectura de control es la misma que en la etapa dos, solo cambia el autómata del movimiento del robot.

Una de las cosas que cambian en el autómata del robot es la compensación del movimiento del caminado, la compensación del giro es exactamente la misma pero en el desfasamiento que existe al caminar ya no funciona solamente regresar la cabeza, ya que habría que conocer la velocidad con la cual regresarla, por ejemplo si la pelota se desplaza lentamente y se mueve la cabeza más rápido, esta pasara la pelota y no podrá darle seguimiento y si por el contrario el movimiento de la pelota es a una velocidad mayor que el movimiento del cuello entonces este nunca podrá alcanzarla y el robot la dará por perdida. Por lo tanto la mejor forma de compensar este movimiento es regresar lo más rápido posible la cabeza a donde debería estar la pelota. Y para ello se realizó el siguiente análisis.

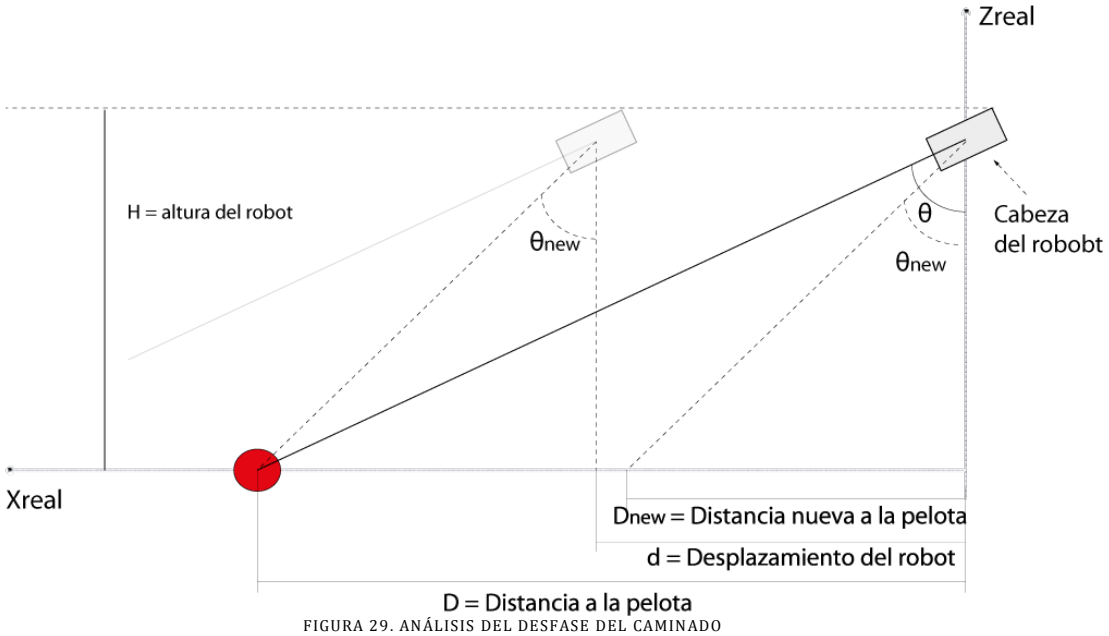


FIGURA 29. ANÁLISIS DEL DESFASE DEL CAMINADO

Del análisis de los ángulos formados y los valores conocidos podemos calcular el ángulo de giro necesario del motor MYZ para que el robot regrese la mirada a donde la tenía, de esta forma si la pelota va rápido o lento, el robot logra ubicarla para continuar con su seguimiento.

Sabemos que:

$$\tan\theta = \frac{D}{H} \tag{12}$$

Despejando tenemos que:

$$D = H \cdot tan(\theta) \tag{13}$$

Para el ángulo a calcular se tiene:

$$\tan\theta_{new} = \frac{D - d}{H} \tag{14}$$

Sustituyendo la distancia a la pelota por el ángulo y la altura conocidos tenemos:

$$\tan\theta_{new} = \frac{H \cdot tan\theta \ - d}{H} \tag{15}$$

Despejando obtenemos:

$$\theta_{new} = \tan^{-1}\left(\frac{H \cdot tan\theta \ - d}{H}\right) \tag{16}$$

Ese ángulo es el valor que tomara el cuello del robot después de caminar, para lograr un seguimiento fluido de la pelota. Para el caso de los pasos laterales no es necesaria una compensación. Al aplicar esta fórmula al seguimiento de la pelota sin movimiento se mejoran mucho los resultados.

3.3.3.1 AUTÓMATA DEL MOVIMIENTO DEL ROBOT PARA PELOTA EN MOVIMIENTO

El movimiento del robot representado como un autómata de estados finitos es:

$$Robot = (Q_r, \{0,1\}, q_{r_0}, \delta_r, \{q_{r_{12}}, q_{r_{13}}\})$$

Dónde:

$Q_c = \{$

q_{r_0} "$Calculando\ el\ error\ e_r$",

q_{r_1} "$Calculando\ Distancia\ y\ orientación$: $se\ calcula\ si\ el\ robot\ esta\ en\ posición\ de\ patear\ la\ pelota\ o\ ejecutar\ algún\ movimiento$
$(\theta \geq \theta_{ref}\ ^\ e_r \leq s_{min}\ ^\ e_r \geq -s_{min}) \rightarrow pateala\ pelota$
$(\theta < \theta_{ref} \mid e_r > s_{min} \mid e_r < -s_{min}) \rightarrow muevete$",

q_{r_2} "$Pateando\ pelota$: $el\ robot\ patea\ la\ pelota\ con\ la\ pierna\ derecha\ si\ e_r \geq 0\ \ y\ patea\ la\ pelota\ con\ la\ pierna\ izquierda\ si\ e_r < 0$",

q_{r_3} "Calculando distancia: verifica si el robot esta ya frente a la pelota $\theta \geq \theta_{ref}$",

q_{r_4} "$Caminando$: $el\ robot\ avanza\ hacia\ adelante\ y\ compensa\ el\ movimiento$",

q_{r_5} "$Resolviendo\ dirección$: $resuelve\ si\ el\ robot\ debe\ caminar\ a\ la\ izquierda\ o\ a\ la\ derecha\ e_r \geq 0 \rightarrow derecha, e_r < 0\ \rightarrow izquierda$",

q_{r_6} "$Calculando\ magnitud$: $calcula\ el\ tamaño\ de\ paso\ derecho\ e_r \geq s_{max}$",

q_{r_7} "$Calculando\ magnitud$: $calcula\ el\ tamaño\ de\ paso\ izquierdo\ e_r \leq s_{max}$",

q_{r_8} "$Camina\ a\ la\ derecha$: $el\ robot\ camina\ hacia\ la\ derecha$",

q_{r_9} "$Camina\ mucho\ a\ la\ derecha$: $el\ robot\ camina\ a\ la\ derecha\ un\ paso\ extra$",

$q_{r_{10}}$ "$Camina\ a\ la\ izquierda$: $el\ robot\ camina\ hacia\ la\ izquierda$",

$q_{r_{11}}$ "$Camina\ mucho\ a\ la\ izquierda$: $el\ robot\ camina\ a\ la\ izquierda\ un\ paso\ extra$",

$q_{r_{12}}$ "Fin: $El\ robot\ ha\ cumplido\ su\ objetivo$",

$q_{r_{13}}$ "$Continuando$: $El\ robot\ se\ ha\ movido\ y\ esta\ listo\ para\ lo\ siguiente$"

$\}$

$q_r \epsilon Q_r$	$\{0,1\}$	$\delta_r(q_r,\{0,1\})\epsilon Q_r$
q_{c_0}	1	q_{c_1}
q_{c_1}	1 “patea”	q_{c_2}
q_{c_1}	0 “muévete”	q_{c_3}
q_{c_2}	1	$q_{c_{12}}$
q_{c_3}	1 “camina”	q_{c_4}
q_{c_3}	0 “gira”	q_{c_5}
q_{c_4}	1	$q_{c_{13}}$
q_{c_5}	1 “izquierda”	q_{c_6}
q_{c_5}	0 “derecha”	q_{c_7}
q_{c_6}	1	q_{c_8}
q_{c_6}	0	q_{c_9}

$q_r \epsilon Q_r$	$\{0,1\}$	$\delta_r(q_r,\{0,1\})\epsilon Q_r$
q_{c_7}	1	$q_{c_{10}}$
q_{c_7}	0	$q_{c_{11}}$
q_{c_8}	1	$q_{c_{13}}$
q_{c_9}	1	$q_{c_{13}}$
$q_{c_{10}}$	1	$q_{c_{13}}$
$q_{c_{11}}$	1	$q_{c_{13}}$

TABLA 5. TRANSICIONES DEL MOVIMIENTO DEL ROBOT

4 RESULTADOS

En este capítulo se tratan los experimentos realizados y los resultados obtenidos, en cada una de las etapas, desde la construcción del robot, su visión y por supuesto los algoritmos de control.

4.1 MOTORES

De acuerdo a las especificaciones de ROBOTIS los motores tienen una resolución de un grado, lo que fue sometido a una prueba experimental para saber su resolución real. Se avanzaron tres motores distintos grado por grado y se midió la posición del motor entre cada paso, de manera ideal el avance registrado entre cada paso del motor para todos los motores debía mostrar una línea recta ya que se avanzó un grado siempre. Pero en la realidad fue muy distinto.

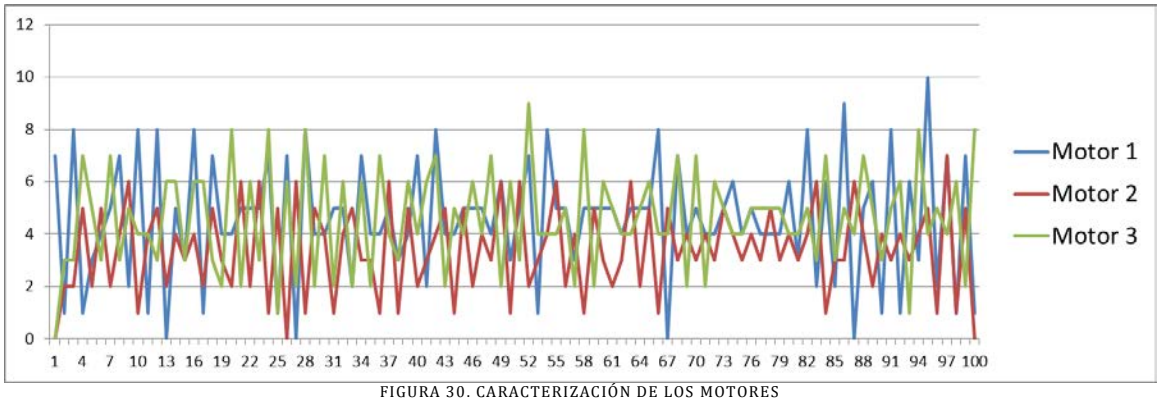


FIGURA 30. CARACTERIZACIÓN DE LOS MOTORES

Podemos ver que a lo largo de las 100 muestras tomadas no siempre se avanzó un grado en la realidad, para los tres motores los avances van desde 0 hasta 10 grados por lo tanto se puede decir que en promedio la resolución de los motores es de 4º.

4.2 VISIÓN

Una de las cosas más importantes al momento de cambiar la cámara por la webcam es cuantos cuadros por segundo me ofrece para ser analizados, para esto se puso un temporizador que midió los cuadros por segundo durante poco más de 3 minutos.
Recordando que el módulo HaViMo ofrece alrededor de 19 cuadros por segundo y estos son procesados dentro de su propio controlador, de manera que a esta velocidad solo se puede acceder a cierta información mas no a la imagen capturada completa esto tardaría más tiempo.
También se debe tomar en cuenta que las mediciones realizadas incluyen el tiempo en que se tarda la computadora en mostrar la nueva imagen capturada en pantalla y enviar esta información al controlador del BIOLOID.

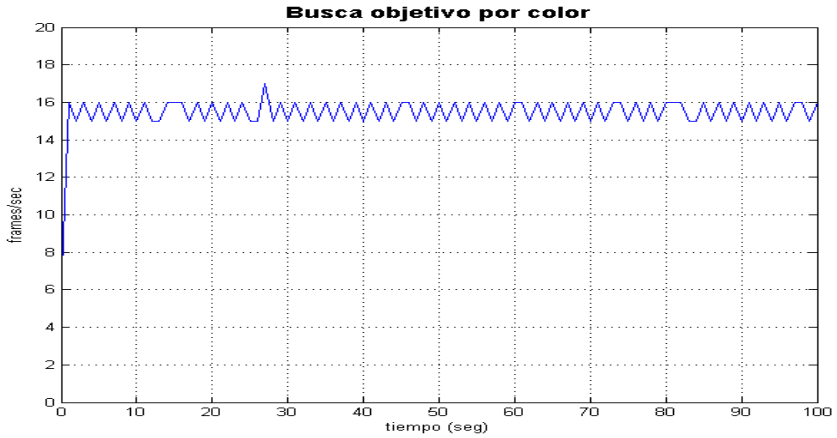


FIGURA 31. VELOCIDAD DE CAPTURA WEBCAM

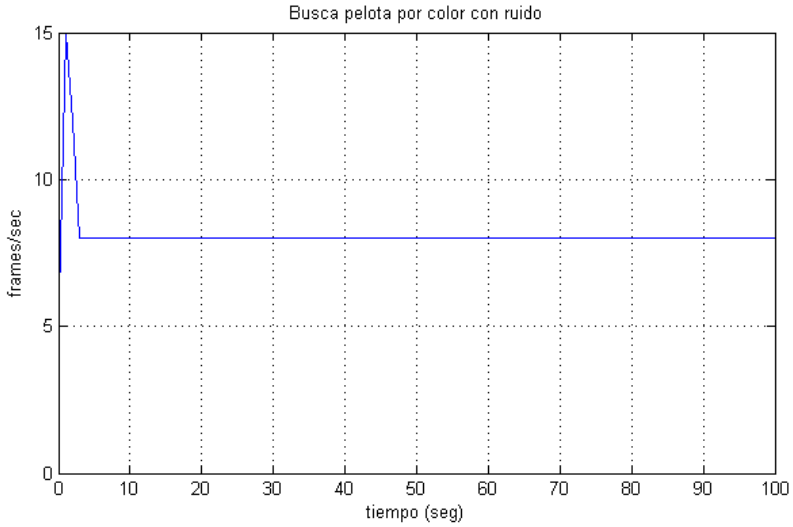
En la gráfica se puede observar que la velocidad de captura promedio oscila entre los 15fps lo que está bastante bien considerando el procesamiento que se lleva a cabo y puede ofrecer la velocidad de HaViMo.
Además el tamaño de la captura puede ser variable y se da la opción para escogerla al principio de la ejecución del programa y en general durante el presente trabajo se trabajó con una resolución de 640 por 480 pixeles.
Lógicamente esta velocidad de captura se verá afectada con algunos algoritmos muy complejos por lo tanto se debe buscar la simplicidad ante todo. En este caso el procesamiento utilizado no afecta realmente al rendimiento pero si otras situaciones.

4.2.1 RUIDO

En la etapa de visión se ocupa un filtro de color rojo antes que cualquier otra cosa, este filtro es realmente eficiente bajo ciertas circunstancias, en la tesis se plantea un ambiente controlado donde solo la pelota puede tener color rojo o el color que se desee que persiga el robot, ya que en las competencias mundiales de futbol de robots así se maneja, pero que pasa ante la presencia de más contenido rojo dentro de la imagen.

FIGURA 32. IMAGEN CON CONTENIDO EXTRA DE COLOR ROJO

En este caso se muestra una imagen donde se observa claramente la pelota de un tono más distinto al demás contenido que paso el filtro, el problema no es aún identificar la imagen de estas regiones encontradas, el problema viene al medir el rendimiento de captura de la imagen.



En la gráfica se puede observar que la velocidad de captura esta en 15fps pero en presencia de ruido como se denominó a este contenido extra del mismo color en la imagen, entonces el rendimiento cae hasta los 8 pixeles por segundo, cifra que a pesar de ser la mitad del rendimiento óptimo, aún me permite dar seguimiento a la pelota de manera correcta.
Ahora no solo ese tipo de ruido afecta el rendimiento, también la cantidad de luz como era de esperarse cambia estas condiciones. Se realizó una medición para la captura de la imagen aplicando solo el filtro de color rojo con condiciones de iluminación muy favorables y otra medición con las mismas características pero con condiciones de iluminación no controladas o reales.

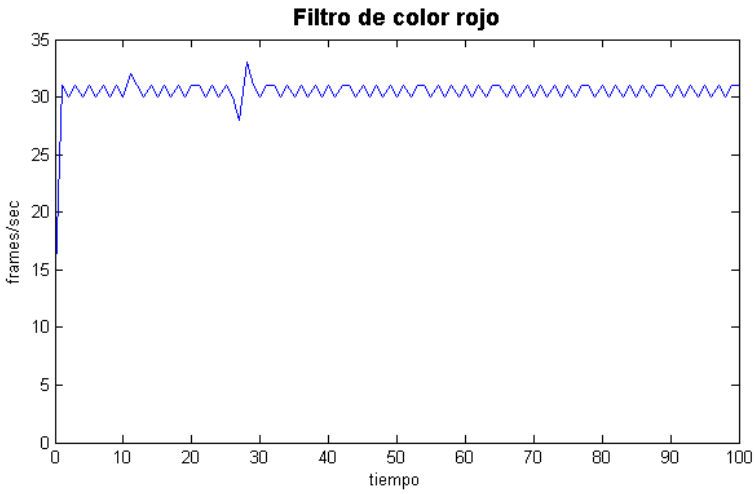
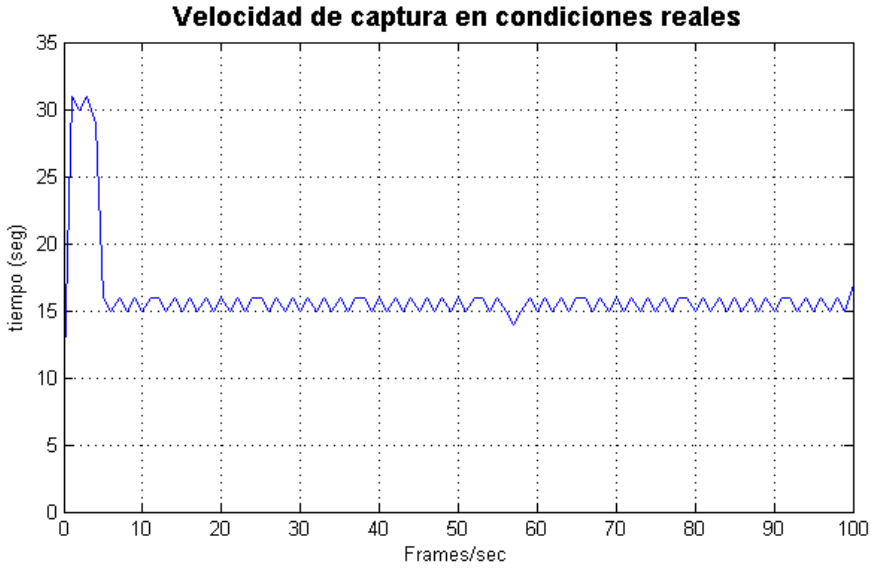


FIGURA 33. FPS BAJO CONDICIONES DE ILUMINACION CONTROLADAS

Para el caso de las mediciones con las condiciones de luz favorables y controladas el rendimiento alcanza incluso los 30fps promedio, a esta velocidad el funcionamiento del robot se ve claramente favorecido, lo que indica que en caso de utilizar una mejor cámara con los mismos algoritmos los resultados serían mucho mejores que los ya obtenidos, que también se consideran bastante buenos.



Cuando se somete el funciona miento del robot a condiciones reales, es decir bajo condiciones normales de luz en el día dentro del laboratorio, el rendimiento y funcionamiento del robot se ven reducidos, claramente se observa que las capturas comienzan en 30fps pero una vez que el lente se adapta a las condiciones de luz, entonces la velocidad de captura baja a 15fps, cifra muy buena para cumplir con el objetivo planteado y además competitiva tomando como referencia el módulo HaViMo.

4.2.2 OTRAS ALTERNATIVAS

Se trató de eliminar el ruido completamente usando técnicas más avanzadas de análisis de imágenes que solo se mencionan de forma breve ya que ninguna de estas técnicas se utilizó en la versión final del algoritmo de visión.

4.2.2.1 DETECCIÓN DE MOVIMIENTO

FIGURA 34. DETECCION DE MOVIMIENTO

Se trató de utilizar un algoritmo de detección de movimiento de manera que el robot no se interesara por regiones rojas estáticas consideradas como ruido, este algoritmo lleva una comparación entre los pixeles de cada cuadro para saber cuáles cambiaron y representan un posible movimiento, después una identificación de estas regiones, un pixeleado y detección de bordes para finalizar por enmarcar la región de mayor tamaño e identificar la pelota.
Aunque operativamente este algoritmo ofrece un buen resultado,

el rendimiento cae a 5fps, de esta forma ya resulta muy difícil dar continuidad al seguimiento de la pelota en tiempo real. Sería una mejora con respecto a HaViMo pero no es viable por el exceso de procesamiento que conlleva, al menos usando las herramientas que se usaron en el presente trabajo.

4.3 CONTROL

Uno de los objetivos es seguir la pelota con el cuello del robot, este algoritmo resulta realmente eficiente en la práctica y puede asegurar que el robot siempre está viendo la pelota. Se puede analizar esta situación desde el punto de vista gráfico y esperado.

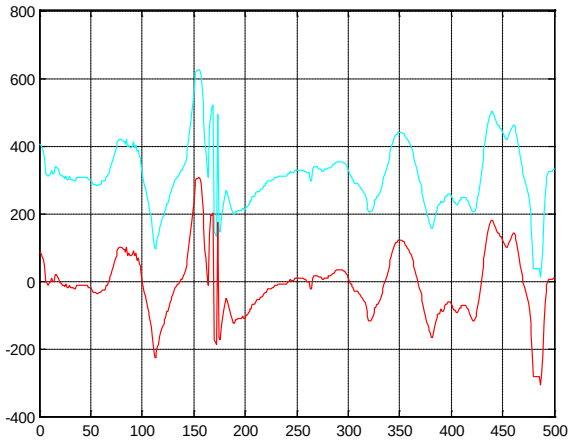


FIGURA 35. COMPORTAMIENTO DE LA PELOTA Y DEL CUELLO CONTRA EL TIEMPO PARA EL EJE X

Como podemos ver en la gráfica el cuello sigue fielmente el comportamiento de la pelota en todo momento, lo mismo sucede para el eje y.

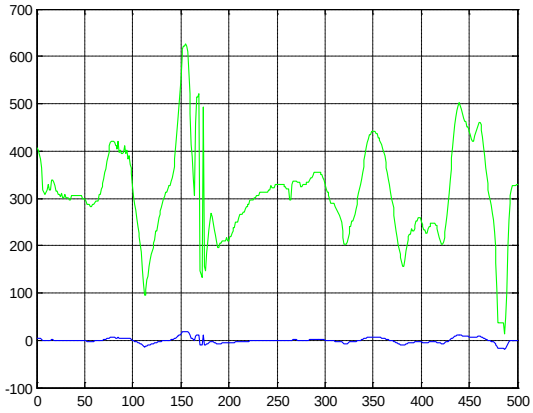


FIGURA 36. COMPORTAMIENTO DE LA PELOTA Y ERROR CONTRA EL TIEMPO PARA EL EJE X

En la gráfica anterior se observa que el error se mantiene al mínimo gracias al seguimiento preciso que le da el robot a la pelota, por eso se puede aseverar que está observándola en todo momento y de allí la creación de los comportamientos.

Se puso especial atención a esta etapa ya que es la que asegura que el robot nunca pierda a la pelota, si existiera alguna deficiencia aquí sería muy complicado encontrar la pelota y poder patearla con un margen de error muy reducido.

4.3.1 VALORES DETERMINADOS EXPERIMENTALMENTE

Para la parte del control del robot hay valores involucrados que es relativamente sencillo obtenerlos, como por ejemplo los ángulos de los motores o la altura del robot, sin embargo existen otras cantidades que resulta complicado determinar su valor, pero si se puede obtener un promedio.
Dos de estas cantidades son los grados que se mueve el robot con cada giro y la distancia de desplazamiento que recorre al caminar. Para esto se tomaron 10 muestras de cada movimiento y se consideró efectivo el valor promedio de estas muestras.

FIGURA 37.MUESTRAS DEL CAMINADO DEL ROBOT

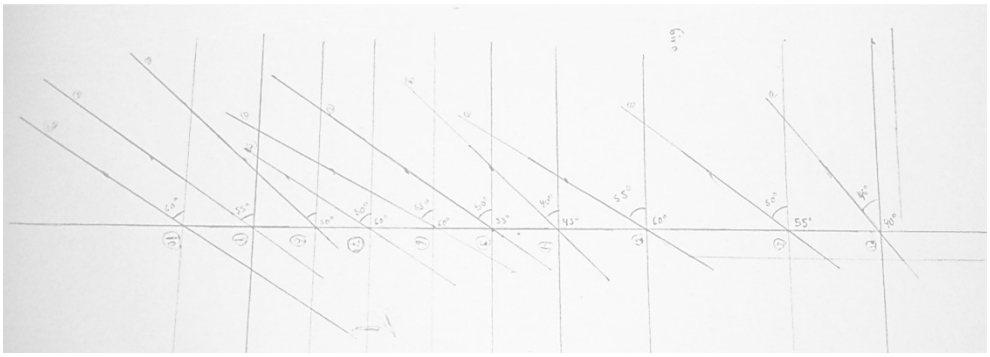


FIGURA 38. MUESTRAS DEL GIRO DEL ROBOT

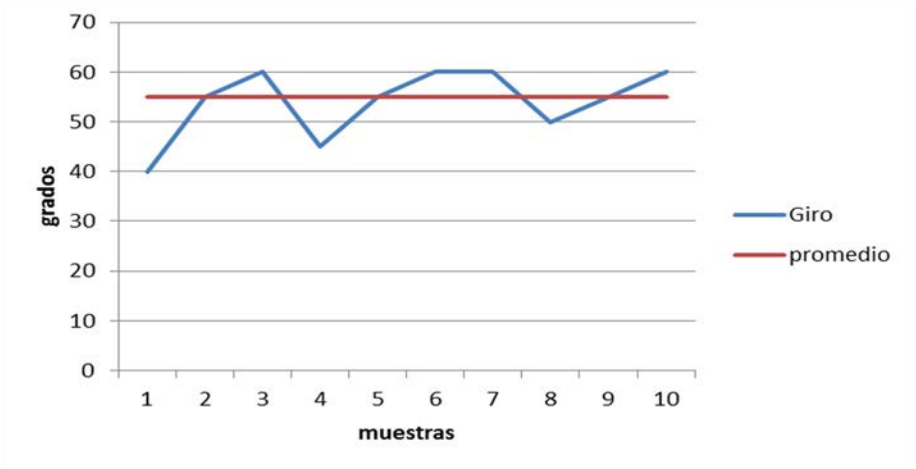


FIGURA 39. GRAFICA MUESTREO DE GIROS

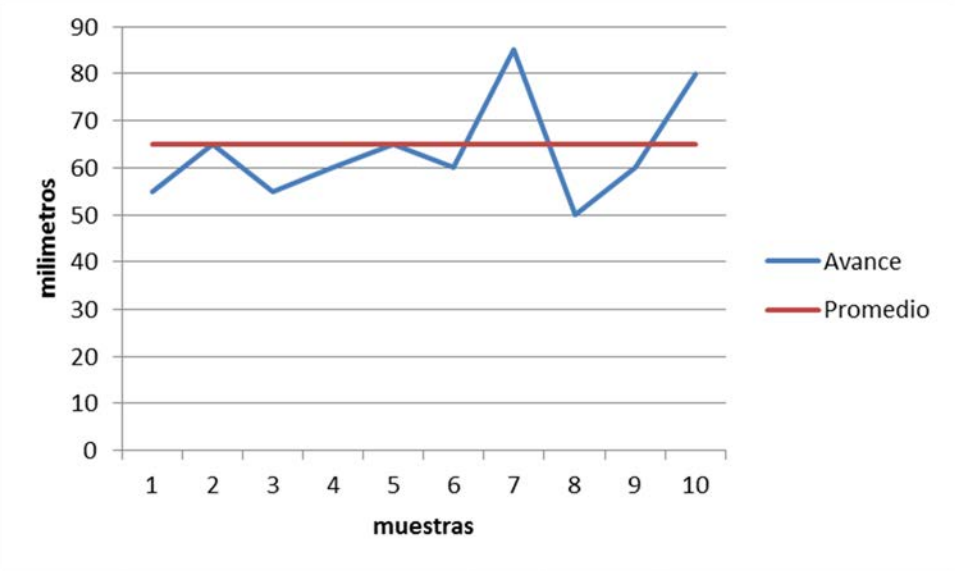


FIGURA 40. GRAFICA MUESTREO DE AVANCES

Con esto podemos involucrar como una constante el ángulo de giro del robot y la distancia que avanza al caminar para el diseño de los comportamientos, cantidades que a pesar de varar mucho en la realidad son referencias que resultan de mucha utilidad y permiten una implementación eficiente.

Para el diseño final de los comportamientos son necesarias otras cantidades que no son tan simples de identificar y sin embargo son clave para el desempeño óptimo del sistema, como son s_{min} y s_{max}.

Para determinar estas dos variables tan importantes se desarrolló una simulación en Matlab que permite configurar las condiciones iniciales del entorno para después predecir los movimientos que realizaría el robot y el resultado final.

El algoritmo de la simulación implementa básicamente los comportamientos descritos anteriormente para que el resultado se asemeje lo más posible al comportamiento real del robot. Nuevamente se recurre a la idea de que el robot siempre está viendo la pelota, gracias al buen diseño del controlador del cuello del robot.

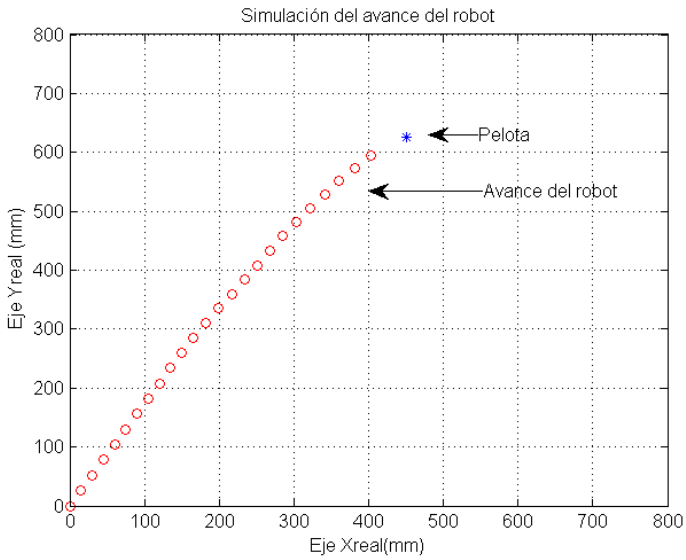


FIGURA 41.SIMULACION IDEAL DEL ROBOT

Cabe hacer notar que en la primera versión del simulador los resultados eran muy diferentes a lo observado en la realidad, aunque si se respetaban los comportamientos y era el resultado deseado.

Las configuraciones iniciales son, la posición del robot y la pelota, la orientación del robot, el avance del robot al caminar y los grados de giro que fueron determinados previamente y los parámetros que se van a determinar con la simulación para el funcionamiento óptimo de los comportamientos que son s_{min} y s_{max}.

Para que el resultado de la simulación sea muy aproximado a la realidad, no solo es suficiente poner los valores reales de avance y giro del robot, también necesitamos introducir ruido, ruido que existe físicamente en el funcionamiento del robot, característica importante a tomar en cuenta, cuando el robot camina este no tiene un avance hacia delante perfecto, es decir no avanza en línea recta. El robot avanza desviándose ligeramente a la derecha, esta desviación horizontal es configurable en los parámetros de inicio aunque fue medida experimentalmente y en promedio es de 10 mm, valor que se usó para las simulaciones.

Una vez hecho esto la simulación parece funcionar perfectamente bien, el paso siguiente es calcular s_{min} y s_{max}, para hacer esto basta meter la simulación en un ciclo finito, que prueba los valores óptimos para estas variables, tomando en cuenta tres factores.

- Que la simulación termine con éxito
- Que sean la menor cantidad de pasos posibles
- Que se detenga muy cerca de la pelota.

Para asegurar que la simulación está funcionando se probaron varios de los valores arrojados en el robot y los resultados fueron extraordinariamente similares. El grado de error entre la simulación y los resultados reales podrían llegar a ser despreciables si se considera que no es una simulación cien por ciento real ya que no se consideran fricciones, condiciones físicas de los motores, condiciones del suelo, e incluso es difícil manejar el avance y giros precisos del robot ya que como se estudió anteriormente existen muchas variaciones.

Cuando probamos valores para s_{min} y s_{max} con los que la simulación se pierde en un bucle infinito debido a que cuando el robot gira no logra de acuerdo a estos parámetros ubicar la pelota de frente en el robot real, este también se perdió en un bucle infinito girando de izquierda a derecha casi exactamente en la posición que la simulación lo hizo.

A diferencia de esto cuando se probó una configuración óptima, ya que no es única como arrojo la simulación y se comprobó físicamente entonces el robot llego a buen término y logró patear la pelota.

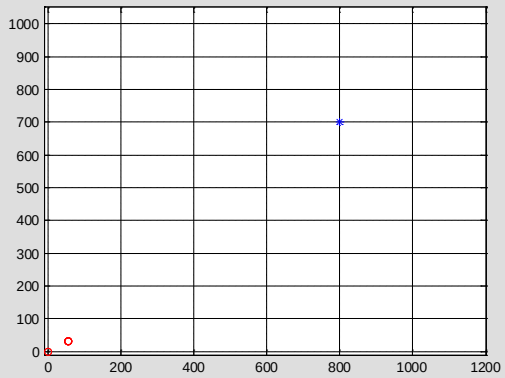
Es muy interesante poder manipular la simulación para predecir qué ocurriría en la realidad, ya que además de ser más rápido también nos arroja un panorama mucho más amplio, permitiendo estudiar el comportamiento del robot si por ejemplo el ruido fuera menor o mayor, si el ángulo de giro del robot fuera diferente, o si el avance de este al momento de caminar fuera diferente, una vez hechas estas predicciones será más confiable trabajar e invertir largas horas de trabajo en modificar esa característica particular del funcionamiento del robot para la mejora continua.

También permite que sea adaptable a otros robots humanoides que realicen las mismas acciones pero con distintas características, por lo que sirve de simulador genérico desde donde se podrán mejorar y compartir fácilmente los comportamientos del robot.

FIGURA 42. CABEZA DEL ROBOT CON WEBCAM

Peor caso de s_{min} y s_{max}.

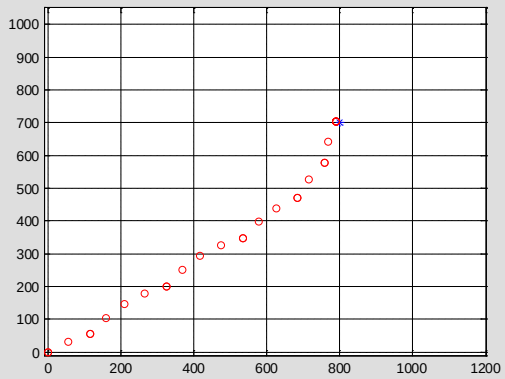
Simulación



Escenario real

Configuración optima de s_{min} y s_{max}.

Simulación



Escenario real

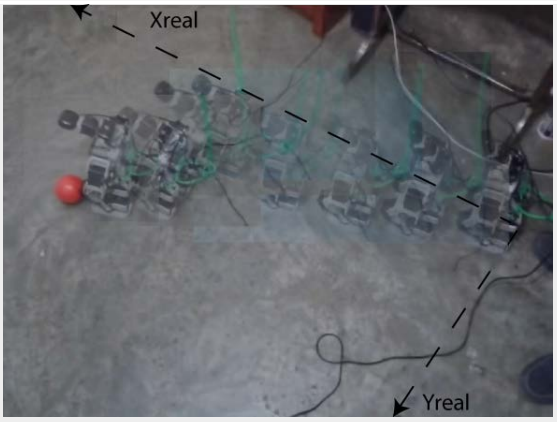


FIGURA 43. COMPARACIÓN ENTRE LA SIMULACIÓN Y LA REALIDAD

Como se puede observar en la comparación entre la simulación y el comportamiento real del robot usando los mismos parámetros, se obtienen resultados muy similares ocurriendo así para cualquier cambio en las condiciones iniciales.
Se realizó un muestreo de esta etapa para conocer tres aspectos fundamentales.

1. Cuando le pega a la pelota
2. Cuando queda dentro del margen de error
3. Cuando se pierde

Para esto se debe establecer un margen de error, el margen de error máximo que puede tener se estableció con un radio de un paso del robot a partir de la pelota, es decir de acuerdo a los valores calculados experimentalmente que el robot debe quedar máximo en un radio de 6.5cm alejado de la pelota.

Muestras	**Golpea la pelota**	**Eficiencia**	**Dentro del margen de error**	**Eficiencia**
150	121	80.6%	148	98.6%

TABLA 6. EFICIENCIA DE LA ARQUITECTURA DE CONTROL

Como se puede apreciar en la tabla de eficiencias la arquitectura arroja buenos resultados.

4.4 PRESENTACIONES

El presente trabajo se expuso dos veces ante un auditorio obteniendo retroalimentación de los presentes y enriqueciendo el trabajo. Durante estas exposiciones los asistentes se mostraron interesados en el tema y se aportaron algunas ideas que podrían ser usadas en trabajos posteriores.

4.4.1 SSIR 2012

La Summer School on Image & Robotics 2012 celebrada en Ensenada, Baja California México fue un encuentro académico escenario de expositores especialistas en análisis de imágenes y robótica ofreciendo la oportunidad a estudiantes de compartir conocimientos y aprender de distintos investigadores durante dos semanas.
En este escenario estudiantes de posgrado de diferentes partes del país y otros países expusieron su trabajo actual en imágenes y robótica, donde se discutieron temas muy interesantes y ricos en contenido; entre ellos se expuso el presente trabajo.
La exposición tuvo una buena aceptación y una retroalimentación importante, además de servir como medio para conocer personas interesadas en temas relacionados de otras instituciones y países.
En esta ocasión fue organizada en el Centro de Educación Científica y de Educación Superior de Ensenada en el estado de Baja California, una institución de gran prestigio y nivel académico.

4.4.2 SIMPOSIO "TENDENCIAS DE LAS NUEVAS TECNOLOGÍAS DE LA INFORMACIÓN, DEL CÓMPUTO Y LAS COMUNICACIONES"

Como homenaje al 50 aniversario del Centro Nacional de Cálculo del Instituto Politécnico Nacional se realizó el Simposio "Tendencias de las Nuevas Tecnologías de la Información, del Cómputo y las Comunicaciones", que se llevó a cabo en el Centro Cultural Jaime Torres Bodet ubicado en Zacatenco DF.
En el simposio hubo expositores de diversas ramas de la tecnología y el presente trabajo fue expuesto con muy buena aceptación e interés por parte del público asistente.
En este simposio se tuvo la oportunidad de comparar el desarrollo y control hechos con este robot contra robots de otras instituciones asistentes, superando lo expuesto por ellos.
Además se recibieron propuestas de colaboración con estas instituciones.

FIGURA 44. PRESENTACIONES

5 CONCLUSIONES Y TRABAJOS FUTUROS

El resultado en la presente tesis es muy satisfactorio ya que se cumplieron los objetivos planteados desde un principio. El trabajo aquí propuesto puede servir de base para trabajos posteriores con la misma temática y ser mejorado. Se creó una arquitectura basada en comportamientos para el control del robot que es muy genérica y podría ser extrapolada a cualquier robot humanoide o incluso a un robot con ruedas, sería interesante realizar estos experimentos en robots con otras características para observar su comportamiento.
También se creó un entorno de simulación para esta arquitectura configurable que puede compartirse y enriquecerse de otros comportamientos, ser utilizado con otros robots o servir de base para otros desarrollos. El simulador se apegó bastante a la realidad cumpliendo con el propósito para el cual fue diseñado.
La arquitectura de control es rápida y eficiente para ser tomada como base en trabajos posteriores, el robot logra interceptar la pelota en movimiento con un nivel de eficiencia alto comparado con el existente en los torneos de la RoboCup.

A esta arquitectura se pueden agregar múltiples comportamientos que lleven al robot a actuar como un jugador de futbol, y estos comportamientos pueden ser agregados también al entorno de simulación.
La webcam empleada es una cámara de bajo costo y pocas prestaciones, sería muy interesante experimentar con una cámara con mayores características y mayor potencia y o una computadora de mayores capacidades.
Si el procesamiento se realiza en una arquitectura especializada como es el caso del módulo HaViMo, el procesamiento es rápido se pueden implementar un mayor número de cálculos. A futuro se puede pensar en la creación de un módulo de visión que implemente más filtros y algoritmos que los utilizados actualmente pero los resuelva rápidamente y así enriquecer más los comportamientos del robot, eliminar el ruido por completo y perder de vista el control del entorno en el robot desempeña sus actividades.
El análisis cinemático a detalle del robot ayudaría a crear movimientos creados especialmente para la arquitectura, mejorando mucho los resultados y abriendo paso a una gama de comportamientos nuevos que paulatinamente lleven al robot a realizar las tareas de manera más fluida.

6 REFERENCIAS

1. Brooks, R.A (1987), Planning is Just a Way of Avoiding Figuring Out What to Do Next, Working Paper 303, MIT AI Laboratory.

2. Brooks, R.A (1990) Elephants don't Play Chess, In Designing Autonomous Agents: Theory and Pactice from Biology to Engineering and back, Patti Maes (Ed.), MIT Press, pp. 3-15.

3. Brooks, R.A (1991) New Approaches to Robotics, Science. Vol.253, pp 1227 -1231.

4. Brooks, R.A. (1986), Achieving Artifitial Intelligence through Building Robots, A.I Memo 898, MIT, AI Lab.

5. Arkin, R.C (1998), Behavior Based Robotics, MIT Press, Cmbridge, MA.

6. M. Egerstedt and C. Martin. (2010) Control Theoretic Splines: Optimal Control, Statistics, and Path Planning, Princeton University Press, Princeton, NJ.

7. Becerra, J.A. Santos, and Duro. (1999), Progressive Construction of Compound Behavior Controllers for Autonomous Robots Using Temporal Information, Advances in Artificial Life, Dario Floreano, Jean-Daniel Nicoud and Francesco Mondada (Eds.), Lecture Notes in Artificial Intelligence, Vol 1674, Springer-Verlag, Berlín, pp. 324 -328.

8. José Santos, Richard J. Duro (2005). Evolución Artificial y Robótica Autónoma (1ª ed).Alfaomega- RA-MA. España.

9. Gonzalo Pajares Martinsanz, Jesús M. de la Cruz García (2008). Visión por Computador (2ª ed). Alfaomega-RA-MA. España.

10. Michael Sipser (2006) Introduction to the theory of computation (Ed.) Thomson Course Technology, United States of America.

11. Fernando Reyes Cortés (2012) MATLAB Aplicado a robótica y mecatrónica. (1ª ed). Alfaomega. México.

12. Juan Humberto Sossa Azuela (2006). Rasgos descriptores para el reconocimiento de Objetos(1ª ed). Instituto Politécnico Nacional. México.

13. Gonzalo Pajares Martinsanz, Matilde Santos Peñas (2006). Ingenieria Artificial e Ingeniería del Conocimiento. Alfaomega- RA-MA. España.