



**UNIVERSIDAD SIMÓN BOLÍVAR
DECANATO DE ESTUDIOS PROFESIONALES
COORDINACIÓN DE INGENIERÍA DE LA COMPUTACIÓN**

Desarrollo de un prototipo de robot humanoide que busque, encuentre y patee una pelota de forma autónoma e inteligente

Por:
Jennifer Dos Reis De Nóbrega
Juliana León Quinteiro

Realizado con la asesoría de:
Ivette Carolina Martínez

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero en Computación

Sartenejas, Noviembre 2014

Resumen

En este proyecto se describe la construcción de Junny, un robot humanoide autónomo e inteligente de 38cm de alto, capaz de detectar la ubicación de una pelota y acercarse a ella para patearla. Sus metas son inspiradas en la competencia RoboCup Soccer de la categoría pequeña.

Junny ha sido construido con las piezas del kit Bioloid Premium del fabricante ROBOTIS [INCb]. Del kit se ha excluido la tarjeta CM-510 para sustituirla por la tarjeta controladora Arbotix, la cual controla los 16 motores Dynamixel Ax-12+ (para mover al robot) y 2 servomotores analógicos . Además se ha agregado un mini computador Raspberry Pi, con su cámara [112a] los servomotores son usados como base para el movimiento de la cámara, para que el robot pueda detectar la posición de la pelota de forma autónoma.

En la Raspberry Pi se usa el lenguaje C++ y se ejecuta un solo programa encargado de detectar la posición de la pelota y decidir qué movimientos son necesarios para llegar a ella. La manera de elegir las acciones se ha realizado con aprendizaje por reforzamiento. Para captar la imagen de la cámara se ha utilizado la librería raspicam_cv [Val]. Para filtrar y procesar la imagen se ha usado la segmentación por regiones para la detección de la pelota, con ayuda de las librerías OpenCv [Tea].

La Arbotix, además de controlar los motores para ejecutar los movimientos deseados, se encarga de monitorizar que el robot se encuentre balanceado, para ello usa el sensor Gyro de Robotis [INCa]. Si detecta un desbalance de un cierto porcentaje puede saber si se ha caído y levantarse.

Todos estos componentes deben ser coordinados para que se logre cumplir la tarea de seguir y patear la pelota. Por ello se hizo necesaria la comunicación entre la Arbotix y la Raspberry Pi. La herramienta empleada para ello ha sido el framework ROS (Ros Operating System) [Ros].

Finalmente se obtuvieron los siguientes resultados, un robot autónomo e inteligente que es capaz de reconocer la pelota, desplazarse a ella y patearla. Con la aplicación de aprendizaje por reforzamiento, se obtuvo un 100 % de acertividad, con una tasa de eficiencia de 0,72.

Palabras claves: Robótica, Apredizaje por reforzamiento, Humanoide, ROS, Arbotix.

Agradecimientos

Queremos agradecer a nuestra familia y amigos, que siempre han estado allí para apoyarnos, han sabido aceptar nuestras ausencias y además han ayudado a que la culminación del mismo fuese posible.

Al Grupo de Inteligencia Artificial, por ser nuestra segunda casa, nuestra segunda familia. Han tenido paciencia con Junny y con nosotras.

Al profesor Guillermo Villegas, por su paciencia y disposición a ayudarnos siempre.

A la profesora Carolina Martínez, por aceptar ser nuestra tutora durante la última etapa.

Queremos otorgar un agradecimiento especial a la profesora Carolina Chang, por estar presente desde el inicio, hasta el final de este proyecto, brindando su ayuda incondicional. Ha sido una gran tutora y excelente profesora. Este es su trabajo también.

Índice general

Índice general	viii
Índice de figuras	x
Glosario	xii
1. Introducción	1
2. Marco teórico	3
2.1. Robótica	3
2.2. Robótica Inteligente	5
2.2.1. Paradigma Jerárquico	5
2.2.2. Paradigma Reactivo	6
2.2.3. Paradigma Híbrido	6
2.3. Inteligencia Artificial	7
2.3.1. Aprendizaje de Máquinas	7
2.3.2. Aprendizaje por reforzamiento	8
2.3.3. Q- learning	8
2.4. Visión Artificial	9
2.4.1. Segmentación por regiones	10
2.4.2. Filtros	10
2.4.2.1. Dilatación	11
2.4.2.2. Erosión	11
3. Construcción de un Robot Humanoide	13
3.1. Diseño y Construcción	13
3.1.1. Diseño del robot	13

3.1.2. Construcción	19
3.2. Detección de la pelota	23
3.2.1. Herramientas software para la detección	23
3.2.2. Obtención de la imagen	24
3.2.3. Procesamiento de la imagen	24
3.3. Búsqueda y Pateo	25
3.3.1. Herramientas software para la búsqueda de la pelota	26
3.3.2. Movimiento del Cuerpo	27
3.3.3. Movimiento de la cámara	29
3.3.4. Representación del mundo	30
3.3.5. Comunicación Arbotix - Raspberry (ROS)	31
3.3.6. Elección de la acción	32
3.4. Aprendizaje	33
3.4.1. Modelo del problema	34
3.4.1.1. Estados	34
3.4.1.2. Acciones	35
3.4.1.3. Recompensas	35
3.4.2. Elección de la acción	36
3.4.3. Actualización de $Q(s,a)$	37
3.5. Detección de caídas	37
3.6. Orientación al arco	39
4. Experimentos y Resultados	42
4.1. Experimentos de Movimientos	42
4.2. Experimentos con Comportamientos Integrados	44
4.3. Experimentos con Aprendizaje	46
5. Conclusiones y Recomendaciones	52
Bibliografía	54
A. Consideraciones Especiales: Obstáculos y Soluciones	57

Índice de figuras

2.1. Paradigma Jerárquico	6
2.2. Paradigma Reactivo	6
2.3. Paradigma Híbrido	7
2.4. Dilatación A: es la imagen original, B: es el núcleo, La estrella es el punto de referencia. Se ve como aumenta la imagen en proporción al patrón aplicado	12
2.5. Erosión, A: es la imagen original, B: es el núcleo, La estrella es el punto de referencia. Se ve como disminuye la imagen en proporción al patrón aplicado	12
3.1. Bioloid Premium Kit	15
3.2. Motores Dynamixel conectados en serie	15
3.3. Batería Lipo	15
3.4. Sensor Gyro	16
3.5. Chip FTDI conectado a la tarjeta Arbotix	16
3.6. Micro Servomotores analógicos TG9e	16
3.7. Extensor de puertos Bioloid	17
3.8. Tarjeta Raspberry Pi	18
3.9. Cámara Raspberry Pi	18
3.10. Circuito con entrada de 11.1 V. Una salida de 5 V para los micro servomotores analógicos y tarjeta Raspberry Pi. Otra salida de 11.1 V para alimentar la controladora Arbotix.	18
3.11. Vista frontal del robot, tipo B. Se puede apreciar la identificación ‘ID’ de cada motor Dynamixel Ax-12+. Nota: los motores 9 y 10 no se utilizan. Imagen tomada sin autorización de [?]	19
3.12. Vista trasera del robot con la tarjeta CM-510, tomada del manual del kit Bioloid.	20
3.13. Vista trasera del robot con la Arbotix	20
3.14. Cámara Raspberry Pi conectada al puerto CSI de la tarjeta	21

3.15. Vista delantera del robot con la cámara y servomotores instalados	21
3.16. Tarjeta controladora Arbotix y componentes conectados (Imagen referencial: el regulador se encuentra dispuesto en diferente orden)	22
3.17. Posiciones de la cámara	30
3.18. Campo de visión del robot con el número de cada región. Cada cuadro blanco demarcado con líneas negras representa la imagen captada desde una posición fija de la cámara. La región de pateo de la pelota se encuentra en las regiones 1 y 2	31
3.19. Estados definidos según la región en la que se detecta la pelota	35
3.20. Distancias relativas de la pelota establecidas para otorgar la recompensa. . .	36
3.21. Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.	38
3.22. Dirección en la que la velocidad angular del eje X crece con respecto al robot.	38
3.23. Gráfica de lecturas de velocidad angular en el eje X.	40
3.24. Campo de visión del robot con el número de cada región para buscar el arco.	40
4.1. Segunda etapa: 300 ejecuciones de acciones combinadas	44
4.2. Pruebas de movimientos integrados CASO I	44
4.3. Pruebas de movimientos integrados CASO II	45
4.4. Pruebas de movimientos integrados CASO III	45
4.5. Pruebas de movimientos integrados CASO IV	46
4.6. Prueba con $K = 1$ y $\gamma = 0,1$	48
4.7. Prueba con $K = 2$ y $\gamma = 0,1$	48
4.8. Prueba con $K = 2$ y $\gamma = 0,7$	49
4.9. Prueba con $K = 3$ y $\gamma = 0,1$	49
4.10. Prueba con $K = 3$ y $\gamma = 0,7$	50
4.11. Prueba con $K = 5$ y $\gamma = 0,1$	50
4.12. Prueba con $K = 5$ y $\gamma = 0,7$	50
4.13. Prueba con $K = 3$ y $\gamma = 0,1$ e inicialización de $Q(s,a)$	51
A.1. Programador para AVR.	58

Glosario

AVR

Es una familia de microcontroladores de instrucciones reducidas de la compañía Atmel..

14

Licencia XBEE

Es una familia de módulos de radio, con protocolo de comunicación inalámbrica basado en radio frecuencias.. 14

Capítulo 1

Introducción

Futuro de la robotica

Algunas destrezas de robots con forma de humanos son caminar, percibir el mundo y tomar alguna acción sobre él. Una de las más avanzadas muestras en el área es el robot ASIMO [Co11], creado por la compañía Honda, cuyos últimos avances incluyen la predicción de trayectoria de objetos para poder esquivarlos. asimo y otros que si COG del MIT hay muchos (hay trabajos q muestran que la gente interactúa mejor con robots humanoides

RoboCup [201a] es una competencia de robótica que ha iniciado en el año 1997 y se ha realizado de forma anual, siendo la más reciente celebrada en Brasil (2014). En ella se busca promover avances las áreas de robótica, inteligencia artificial, mecatrónica, entre otras. El enfoque principal de la competencia se centra en las cinco categorías agrupadas dentro de la liga RoboCup Soccer [201b], la cual consiste en la participación de robots humanoides que se enfrentan a otro equipo para jugar fútbol. Aunque existen otras ligas dentro de la RoboCup que no se involucran con el juego de fútbol, la meta final de la competencia es lograr que en el año 2050 el equipo campeón logre vencer al ganador del año en la copa mundial de la FIFA (International Federation of Association Football).

Existen equipos que han participado durante varios años consecutivos en la competencia

Robocup, logrando mejoras en sus diseños y técnicas; tal es el caso del equipo MRL que ha participado en los años 2011, 2012, 2013 y 2014 en la categoría “Humanoid League”, han iniciado con el hardware del robot DARwIn-OP y con el tiempo han modificado los componentes electrónicos para agregar eficiencia y estabilidad. Para el balance han utilizado un giroscopio y sensores de aceleración, y para la visión, una cámara conectada por glsUSB al CPU principal [SSA⁺].

En el desarrollo de habilidades más específicas con respecto a la competencia RobotCup, se han propuesto dos posibles estrategias para el pateo de la pelota donde los factores fundamentales para un buen desempeño es la fuerza y la rapidez con que se patea [YML]. Las dos estrategias de pateo se ponen en práctica en distintas circunstancias del juego basado en la cinemática y dinámica de equilibrar el cuerpo al momento de patear.

En el desarrollo de esta investigación se presenta un robot humanoide (Junny) autónomo e inteligente de 38 cm de altura, cuyos objetivos están inspirados en la competencia RobotCup.

Se planteó como objetivo principal construir un prototipo de robot humanoide que sea capaz de detectar la cercanía de una pelota, acercarse a ella y patearla, reincorporándose a la posición de pie en caso de perder el equilibrio y caer mientras camina. Para cumplir con este objetivo se ha desglosado un conjunto de objetivos específicos que se describen a continuación:

La siguiente investigación se divide en 5 capítulos. El capítulo 2 se refiere a una base teórica de conceptos e información necesarias para el soporte del desarrollo de este proyecto que se encuentra en el capítulo 3 donde se explica todo el procedimiento realizado para llevar a cabo el producto. El apéndice A describe la serie de consideraciones importantes que se deben tener a la hora de reproducir esta investigación como los problemas claves y las soluciones que se les dio. En el capítulo 4 se encuentran los experimentos y sus resultados. Por último están las conclusiones y recomendaciones en el capítulo 5.

Capítulo 2

Marco teórico

En este capítulo se presentan los conceptos que conforman la base teórica para comprender este trabajo. Primero se brinda una descripción de los términos relativos a la robótica y las partes principales de un robot. Posteriormente se describen algunos conceptos que tienen que ver con la robótica inteligente, como los paradigmas, la inteligencia y la visión artificial para la detección de objetos.

2.1. Robótica

El presente trabajo se basa en la construcción de un robot humanoide, por lo tanto es importante definir qué es un robot, qué significa que sea humanoide y cuáles son algunos de sus componentes principales.

- **Robótica:** Es la rama de la tecnología que se encarga del diseño, construcción, operación y aplicación de los robots [Pre14].
- **Robot:** Es un agente físico que realiza tareas manipulando su ambiente. Generalmente un robot está equipado con actuadores y sensores. Una posible división para categorizar-

los es por su forma, en primera instancia estan los manipuladores o brazo robóticos que normalmente estan anclados a su espacio de trabajo y generalmente desempeñan tareas en fábricas o líneas de emsamblaje; como su nombre lo describe suelen ser con forma de brazo. La siguiente categoría se refiere a robots moviles, su principal característica es el desplazamiento por lo cual poseen piernas, ruedas o cualquier mecanismo que le permita desplazarse, generalmente en esta clasificación se incluyen robots que realizan multiples tareas como del hogar o reconocimiento de un espacio entre muchas otras. La última categoría es mixta son aquellos robots que poseen características tanto de brazos robóticos como de moviles, en ella se incluye a los robots humanoides que por su nombre estan hechos a semejanza del hombre, sus tareas suelen recurrir un poco mas de esfuerzo ya que en la manipulación de objetos no poseen la ventaja del anclaje que tienen los brazos robóticos. Los robots reales suelen enfrentarse a ambientes parcialmente observables, estocásticos, dinámicos y continuos [pet09].

- **Sensores:** Son los dispositivos que envian percepción del ambiente al robot, algunos miden los cambios o percepción del mundo que rodea como las cámaras, los sonares entre otros. También existen los que miden la propia movilidad del robot como los giroscopios y acelerómetros. En general un sensor es una interfaz de percepción entre el ambiente y el robot [pet09].
- **Actuador:** Dispositivos que realizan cambios físicos en el medio ambiente. Por ejemplo ruedas, piernas, pinzas, entre otros [pet95].
- **Servomotor:** Es un motor eléctrico, un actuador, que permite controlar la velocidad y la posición [AiR00].
- **Giroscopio:** Es un sensor de la categoría para la percepción propia que informan al robot de su propio estado, pertence a los sensores de inercia [pet95]. Mide el momento

angular y se utiliza para mantener orientación o equilibrio.

2.2. Robótica Inteligente

Es importante diferenciar cuando un robot es inteligente o no. Cuando un robot es operado a distancia, y no es capaz de cumplir sus tareas sin la intervención de un humano, entonces no se considera inteligente. Tampoco se considera inteligente si las tareas que ejecuta se hacen sin sentido o de manera repetitiva. En cambio cuando un robot puede interactuar con el mundo de manera autónoma se considera que es un robot o agente inteligente [AiR00]. Existen diferentes estrategias o enfoques de cómo aplicar la inteligencia en un robot. Esta sección se dedica a describir los enfoques que en [AiR00] se definen como paradigmas.

Paradigmas de robótica

Según Robin Murphy en [AiR00], existen tres paradigmas en los cuales se clasifica el diseño de un robot inteligente, estos paradigmas pueden ser descritos de dos maneras: la relación entre las primitivas básicas de la robótica: percibir, planificar, actuar; o de la forma en que los datos son percibidos y distribuidos en el sistema.

Percibir se refiere al procesamiento útil de la información de los sensores del robot. Planificar, cuando con información útil, se crea un conocimiento del mundo y se generan ciertas tareas que el robot podría realizar. Por último actuar consiste en realizar la acción correspondiente con los actuadores del robot para modificar el entorno.

2.2.1. Paradigma Jerárquico

Este paradigma es secuencial y ordenado. Primero el robot percibe el mundo y construye un mapa general. En base al mapa ya percibido y sin percibir más, el robot planifica todas

tareas necesarias para lograr la meta. Luego ejecuta la secuencia de actividades según su planificación. Una vez culminada la secuencia se repite el ciclo percibiendo el mundo, planificando y actuando [AiR00]. El algoritmo general utilizado en este paradigma se muestra en la imagen 2.1.



Figura 2.1: Paradigma Jerárquico

2.2.2. Paradigma Reactivo

El paradigma reactivo omite por completo el componente de la planificación y sólo se basa en percibir y actuar. El robot puede mantener un conjunto de pares percibir-actuar como se muestra en la figura 2.2, éstos son llamados comportamientos y se ejecutan como procesos concurrentes. Un comportamiento toma datos de la percepción del mundo y los procesa para tomar la mejor acción independientemente de los otros procesos [AiR00].



Figura 2.2: Paradigma Reactivo

2.2.3. Paradigma Híbrido

El paradigma híbrido es una mezcla de los dos paradigmas anteriores. Primero se planifica cuál es la mejor manera de cumplir el objetivo principal, descomponiendo la tarea general en sub-tareas y decidiendo qué comportamientos sirven para cumplir cada una. De allí en adelante se ejecutan los comportamientos (percibiendo y actuando), hasta que el plan sea

ejecutado, si es necesario se puede volver a planificar. Vale la pena acotar que la información de los sensores se encuentra disponible para el planificador, de manera que pueda crear un modelo del mundo y tomar decisiones en base a él [AiR00]. Se puede apreciar el la figura 2.3



Figura 2.3: Paradigma Híbrido

2.3. Inteligencia Artificial

La inteligencia artificial es un término relacionado con la computación que puede ser aplicado a la robótica para crear robots inteligentes. El término “inteligencia artificial” ha tenido varias definiciones. Ocho de ellas, las cuales nacieron a finales del siglo XX, se encuentran organizadas en [pet95] bajo cuatro categorías: pensar y actuar de forma humana, pensar y actuar de forma racional. De ellas se puede entender que la inteligencia artificial tiene que ver con lograr que una máquina o un robot resuelva problemas de manera inteligente, es decir, de manera que parezca que el razonamiento y comportamiento humano las ha resuelto.

2.3.1. Aprendizaje de Máquinas

El aprendizaje de máquinas es un área de la inteligencia artificial enfocados en lograr construir programas de computadora que automáticamente mejoren con la experiencia. Se definen estos programas según [Mit97] ”Se dice que un programa aprende de la experiencia E con respecto a una tarea T y desempeño P si el desempeño en la tarea T, medido por P, mejora con la experiencia E”.

2.3.2. Aprendizaje por reforzamiento

El aprendizaje por reforzamiento es un tipo de aprendizaje de máquinas que se basa en un sistema de recompensas positivas y negativas. Para otorgar las recompensas hay dos maneras una en cada estado o una sola vez al llegar al estado final. Un estado se define como una configuración particular de la representación del mundo con características relevantes para el problema.

El agente debe escoger una secuencia de acciones y dependiendo de su recompensa aprender si es favorable o no la aplicación de esas acciones en ese estado.

El agente existe en un entorno descrito por algunos estados S . Puede ejecutar un conjunto de acciones A . Cada vez que ejecuta una acción a_t en algún estado s_t el agente recibe una recompensa r_t . El objetivo es aprender una política $\pi : S \rightarrow A$ que maximice la suma esperada de esas recompensas con descuento exponencial de las recompensas futuras. [Mit97] El resultado de tomar las acciones puede ser determinista o no, en el caso de este proyecto no es determinista, es decir, existen porcentajes de probabilidad de pasar a un estado u otro al tomar una acción en un estado en particular.

2.3.3. Q- learning

Es un método de aprendizaje por reforzamiento que en un estado, compara las utilidades esperadas de las posibles acciones a tomar sin necesidad de saber el estado resultante, por tanto no se necesita tener un modelo del entorno que rodea al robot [pet95] (esto es, cómo funciona el ambiente o qué estado se alcanza como consecuencia de tomar cada acción).

La forma de aprender la política $\pi : S \rightarrow A$ es de forma indirecta, a través de la función $Q(s, a)$. La función representa el valor de la máxima recompensa acumulada, con descuento de las recompensas futuras, que puede ser alcanzada desde el estado s y aplicando a como la

primera acción [Mit97]. La ecuación se puede escribir como:

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a))$$

En donde $r(s, a)$ es la recompensa dado según el resultado de haber tomado la acción a en el estado s . $\delta(s, a)$ es el estado obtenido luego de tomar la acción a en el estado s . γ es el descuento que se le aplica a las recompensas futuras. La función $V^*(s')$ genera el máximo valor Q que puede ser alcanzado desde el estado s' . Esto es,

$$V^*(s') = \max_{a'}(Q(s', a'))$$

De esta forma se obtiene una definición recursiva,

$$Q(s, a) = r + \gamma \max_{a'} Q(\delta(s, a), a')$$

que puede ser calculada de manera iterativa, inicializando los valores de Q con números aleatorios [Mit97]. Como la función $\delta(s, a)$ no es conocida, es necesario poner en ejecución al agente para que, una vez tomada la acción, se observe el estado resultante que desencadenó y así poder calcular el resultado de la ecuación. La recompensa r se puede definir en base a qué tan bueno es el estado resultante.

2.4. Visión Artificial

Una manera de obtener información del ambiente es con la visión artificial. Esta consiste en usar un dispositivo (cámara) que capta un rango de espectro electromagnético y produce una imagen. La representación de la imagen se almacena como una matriz de píxeles, cada píxel es un elemento que guarda información de una región en el espacio captado. Si se usa

una cámara de luz, la información de cada píxel será el color. [AiR00]

Por lo general luego de obtener una imagen se requiere extraer información de ella, por lo cual se han desarrollado diferentes algoritmos y técnicas que ayudan en esta tarea. En la sección 2.4.1 se describe la técnica de segmentación por regiones, que es la utilizada en el proyecto.

Por otro lado, existen varios algoritmos que se dedican a la transformación de las imágenes para reducir ruidos, compensar problemas de iluminación, extraer formas, identificar objetos, entre otros. En esta sección se describen dos de las técnicas de transformación para reducir el ruido basadas en la dilatación y erosión de la imagen (sección 2.4.2).

2.4.1. Segmentación por regiones

La práctica más general en visión de computadoras aplicado a robótica es la identificación de regiones por un color particular, este proceso se llama segmentación por regiones. El algoritmo básico consiste en identificar todos los píxeles en una imagen que forman parte de una región y luego ir al centro de la región. El primer paso es identificar todos los píxeles en la imagen que compartan un rango de valores con el color particular elegido y agruparlos, aquellos píxeles que no compartan el color son descartados [Boo08].

2.4.2. Filtros

El filtrado de imágenes es una técnica para la transformación de imágenes, que consiste en destacar sus características más relevantes en base a un propósito en particular.

Generalmente en la tarea de extracción de información de una imagen se utilizan filtros para descartar zonas o características que no son importantes para el patrón deseado y para determinar el área deseada ya sea por patrones de forma o color.

En la investigación, los algoritmos de filtrado aplicados a las imágenes fueron: Clausura

Morfológica y Apertura Morfológica, filtros que aplican las técnicas de erosión y dilatación a las imágenes.

Transformaciones Morfológicas

Algunas transformaciones morfológicas básicas son dilatación, erosión, unión e intersección, se utilizan en amplia variedad de contextos como la eliminación del ruido, aislamiento de elementos individuales y elementos de unión dispares en una imagen en este proyecto de utilizaron dilatación y erosión. [Boo08]

2.4.2.1. Dilatación

La dilatación es una convulsión (patrón que se le aplica a toda la imagen) entre alguna imagen (o región de una imagen), que llamaremos A y un núcleo que llamaremos B , el núcleo, que puede ser de cualquier forma o tamaño, generalmente una figura geométrica un cuadrado o disco, tiene un solo punto de referencia definido. Para mayor claridad el núcleo es una matriz de tamaño fijo de coeficientes numéricos junto con un punto de referencia en dicha matriz, que normalmente se encuentra en el centro. El núcleo puede ser pensado como una plantilla o máscara, y su efecto para la dilatación tal como un operador de máximo local sobre la imagen, se calcula el máximo valor de los píxeles común a B y reemplazamos el píxel de la imagen en el punto de referencia con ese valor máximo. Esto causa regiones brillantes dentro de una imagen y la hacen crecer. Este crecimiento es el origen del término “operador de dilatación” [Boo08].

2.4.2.2. Erosión

La erosión es la operación inversa a la dilatación. Esta acción del operador es equivalente a el cálculo de un mínimo local sobre el área del núcleo. La erosión genera una nueva imagen

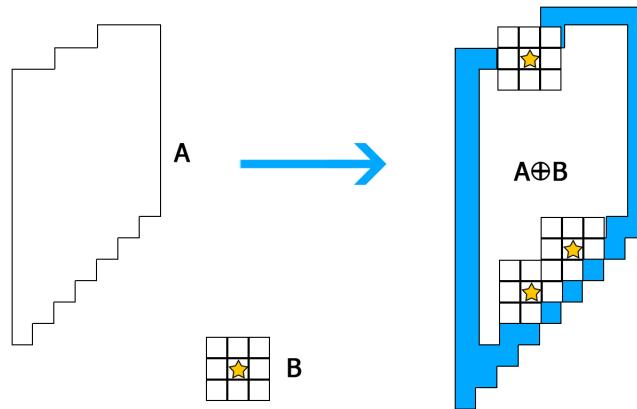


Figura 2.4: Dilatación A: es la imagen original, B: es el núcleo, La estrella es el punto de referencia. Se ve como aumenta la imagen en proporción al patrón aplicado

a partir de la original, utilizando el siguiente algoritmo: como el núcleo B es analizado sobre la imagen, se calcula el mínimo valor del píxel superpuesto por B y se reemplaza el píxel de la imagen con un punto de referencia de valor mínimo [Boo08]. Véase en la figura 2.5

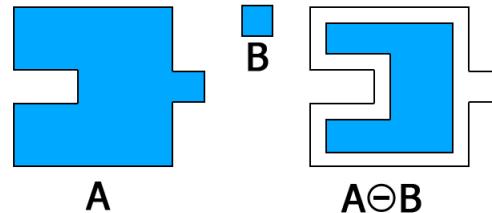


Figura 2.5: Erosión, A: es la imagen original, B: es el núcleo, La estrella es el punto de referencia. Se ve como disminuye la imagen en proporción al patrón aplicado

Este capítulo constituyó la base teórica que sustenta el proyecto, en el siguiente capítulo se presenta el proceso de desarrollo que se siguió para su culminación.

Capítulo 3

Construcción de un Robot Humanoide

En el presente capítulo se describe todas las actividades que se han llevado a cabo para lograr construir un robot humanoide capaz de detectar la ubicación de una pelota, de un color determinado, buscarla, y al llegar a ella patearla en dirección al arco. También se describen las actividades realizadas para lograr separarse como levantarse en caso de tener un desbalance y caer.

3.1. Diseño y Construcción

El primer paso ha sido la construcción de la parte física del robot. Para tal fin se procedió a la elección del diseño y ensamblaje de las piezas. En la sección 3.1.1 se describe cada uno de los componentes utilizados para armar el robot, y luego, en la sección 3.1.1 se explica cómo se integraron esas piezas para obtener un humanoide adaptado a los objetivos de este proyecto.

3.1.1. Diseño del robot

A continuación se presenta una descripción de todos los elementos utilizados para la construcción del robot humanoide y sus decisiones de diseño.

Las opciones que se han consideradas para el cuerpo de Junny han sido: La construcción con piezas de LEGO, la construcción y diseño mecánico desde cero y la construcción con piezas del kit Bioloid. Estas opciones corresponden a los recursos disponibles. Tanto la construcción LEGO como la opción de realizarlo desde cero implicaban mas tiempo del que se disponía, además que realizarlo en su totalidad con diseño mecánico y electrónico implicaba conocimientos más avanzados en el área de mecatrónica que no eran el objetivo del proyecto. Por lo tanto se escogió la construcción con el kit Bioloid como la opción más factible para el cumplimiento de los objetivos, el kit Bioloid es un kit de robótica con piezas prefabricadas que permite armar diferentes tipos de robot pero principalmente humanoides. Su empaque se puede observar en la figura 3.1. El fabricante, ROBOTIS, incluye un manual con varios modelos de robots con instrucciones de ensamblaje. Provee una tarjeta controladora, CM-510, a la que se conectan los motores Dynamixel y algunos sensores que se programan a través de la interfaz de ‘RoboPlus’ [INCb]. La tarjeta controladora CM-510 fue sustituida por la tarjeta controladora de software libre Arbotix. La utilización de la tarjeta Arbotix permite una mayor flexibilidad en el control de motores y la incorporación de una variedad de sensores no soportados por la tarjeta CM-510. Además, la tarjeta Arbotix posee mayor soporte y amplitud en la comunicación entre distintos dispositivos, permite el control avanzado de algunos tipos de servos Dynamixel y robots basados en Bioloid. Incorpora un microcontrolador AVR, radio inalámbrica Licencia XBEE, conductores de motor dual, y cabeceras de estilo servo de 3 pines para entrada/salida digital y analógica [LLC].

Los Motores Dynamixel Ax-12+ son actuadores inteligentes y modulares que incorporan un reductor de engranajes, un motor DC de presión y un circuito de control con funcionalidad de red lo cual permite formar series o cadenas de motores (figura 3.2). Estos motores están incorporados en el kit Bioloid al igual que la batería de polímero de litio (Lipo) una fuente de poder usada para motores y componentes electrónicos. La batería usada es de 11.1 voltios

y 1 amperio. [Rob]



Figura 3.1: Bioloid Premium Kit



Figura 3.2: Motores Dynamixel conectados en serie



Figura 3.3: Batería Lipo

Para el balance y detección de caídas de Junny se utilizó con un giroscopio (Robotics Gyro) que mide la velocidad angular que genera los movimientos del robot, este se encuentra diseñado para mantener el balance del robot y ser usado para otras aplicaciones de movimiento [INCa]. En figura 3.4 se puede observar su estructura. Existen otros sensores de inercia que se enfocan en realizar la tarea de balance y orientación del robot como el acelerómetro.

El FTDI (Future Technology Devices International) utilizado para comunicar físicamente

a la Arbotix y a la Raspberry Pi es una tarjeta controladora (figura 3.5) que ofrece el servicio de conversión de datos de USB a UART. Permite la comunicación entre diferentes dispositivos [Ele] fue elegida en lugar de XBEE pues la Raspeberry Pi y Arbotix estan fisicamente cerca y usar XBEE implicaba un mayor costo para el proyecto.



Figura 3.4: Sensor Gyro



Figura 3.5: Chip FTDI conectado a la tarjeta Arbotix



Figura 3.6: Micro Servomotores analógicos TG9e

El extensor de puertos Bioloid fue necesario para una mejor movilidad del robot pues se dividió su cuerpo en cuatro extremidades por lo tanto cuatro series de motores distintas y el expansor permite aumentar el número de cadenas de servos conectados a la tarjeta (ver figura 3.7) [WL]. De igual manera es explicado a más profundidad en la siguiente sección.

Para el movimiento de la cámara se utiliza como base dos micro servo motor analógico TG9e que es un pequeño servomotor cuya capacidad de torque alcanza los 1.50 kg-cm [Hob]. Permite ser controlado en posición en un rango de 180°. Ver figura 3.6.

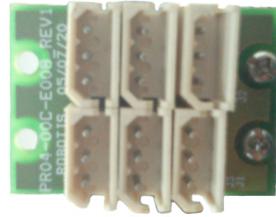


Figura 3.7: Extensor de puertos Bioloid

La Raspberry Pi es un computador de aproximadamente 10 cm de largo, a la que se puede conectar un monitor y un teclado. Puede ser utilizado en proyectos de electrónica y para varias de las tareas que un computador de escritorio puede hacer, como hojas de cálculo, procesadores de texto y juegos [112b]. Ver figura 3.8. La Raspberry Pi cuenta con un procesador gráfico que permite aligerar la carga del procesador central [de14]. Para las funciones correspondientes a la Rasberry Pi, en el proyecto también era factible la utilización de un teléfono celular avanzado, sin embargo esta opción fue descartada pues la cámara del teléfono no posee grados de libertad, agrega un peso importante a la estructura del robot y un costo superior que la utilización de la raspberry.

Para lograr el objetivo de detección de la pelota y orientación al arco existen varias posibilidades factibles para la tarea, un sensor de brújula, un sensor de proximidad, entre otros, sin embargo la ultilización de una cámara para esta tarea representaba numerosas ventajas como la certeza de ubicar la pelota, presión en la distancia, el ahorro de la utilización y manejo de mas sensores ademś de la posibilidad de ampliar las habilidades y tareas que podría realizar Junny. Por ello se tomo la elección de la cámara Raspberry Pi que puede captar imágenes y grabar videos de alta definición. Se conecta a la Raspberry Pi con un cable de cinta plana de 15 cm en el puerto CSI. Tiene 5 megapíxeles de foco fijo que soporta los modos de vídeo de 1080x30, 720x60 y VGA90. Puede ser manejada con las librerías MMAL, V4L u otras librerías de terceros como la de Python [112a].(figura 3.9)

Debido a que no todos los componentes poseen las mismas exigencias con respecto a



Figura 3.8: Tarjeta Raspberry Pi

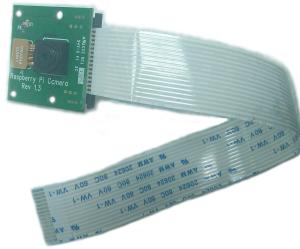


Figura 3.9: Cámara Raspberry Pi

voltaje y amperaje, se realizó un regulador (ver figura 3.10) con una salida de 5 voltios para la tarjeta Raspberry Pi y dos micro servomotores, y otra salida de 11.1 V para la tarjeta Arbotix que a su vez alimenta a los componentes conectados en ella (motores Dynamixel y Giroscopio). Si bien la tarjeta Arbotix posee un regulador interno de cinco voltios la opción de conectar todo a la salida de 5 V del regulador no era posible, dado que los motores Dynamixel requieren alimentación de 11 voltios.

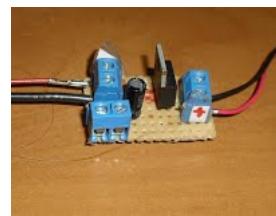


Figura 3.10: Circuito con entrada de 11.1 V. Una salida de 5 V para los micro servomotores analógicos y tarjeta Raspberry Pi. Otra salida de 11.1 V para alimentar la controladora Arbotix.

3.1.2. Construcción

Como se menciono anteriormente para la construcción del robot se utilizó el kit de piezas Bioloid Premium de marca Robotis el cual incluye motores Dynamixel Ax-12+, una tarjeta controladora CM-510, un sensor Gyro, un manual, entre otros elementos. El manual incluye las instrucciones de como armar los modelos A, B y C de humanoide, el utilizado en este proyecto es el tipo B, haciendo uso de 16 motores. En las figuras 3.11 y 3.12 se puede observar la estructura del robot que aparece en el manual del kit. La elección de este modelo se debió a que los motores como recuversos eran escasos y este modelo al igual que el modelo C utiliza dos motores menos que el modelo A, ya que el A combina los grados de libertad en la cadena que los modelos B y C poseen por separado. El modelo B posee un grado de libertad de interes a la hora de girar del robot por ello se eligió en particular.

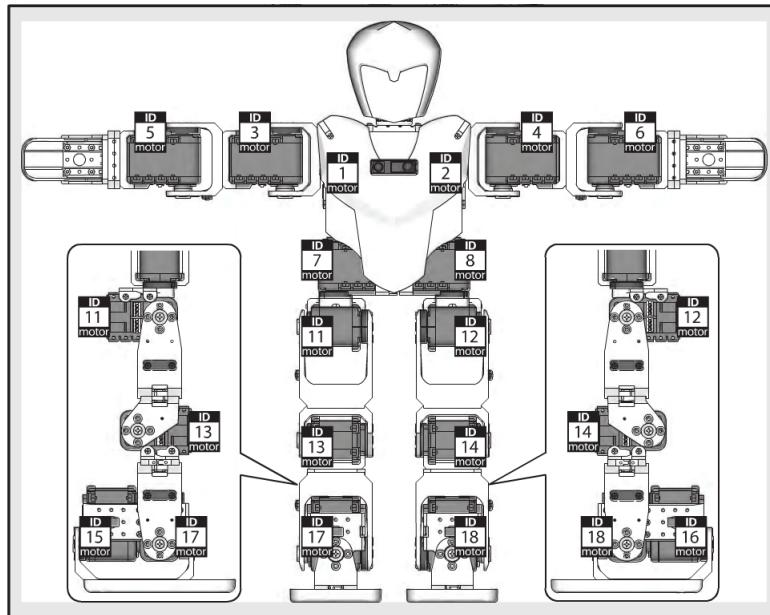


Figura 3.11: Vista frontal del robot, tipo B. Se puede apreciar la identificación ‘ID’ de cada motor Dynamixel Ax-12+. Nota: los motores 9 y 10 no se utilizan. Imagen tomada sin autorización de [?]

En la figura 3.13 se puede observar la estructura del robot con la Arbotix incorporada.

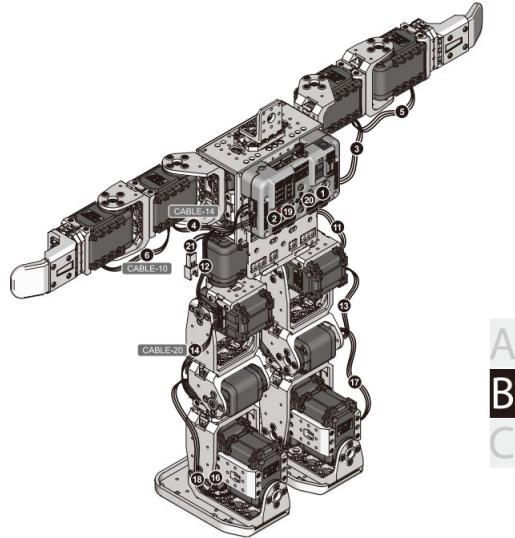


Figura 3.12: Vista trasera del robot con la tarjeta CM-510, tomada del manual del kit Bioloid.

En la parte interna del tronco del robot se sitúa el sensor Gyro.

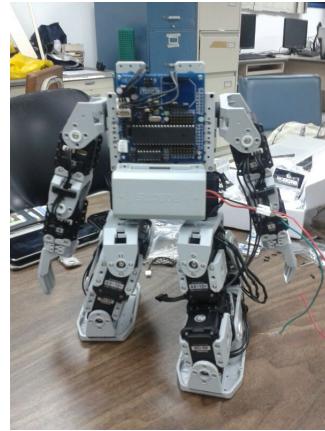


Figura 3.13: Vista trasera del robot con la Arbotix

Para el movimiento de la cámara se ha incorporado dos micro servomotores TG9e ya que estos poseen un tamaño reducido perfecto para acoplarlo en la estructura al igual que su peso, uno para el movimiento horizontal y otro para el vertical. La conexión de uno de estos motores se ilustra en la figura 3.16, en donde se puede observar que se encuentra conectado al puerto B de los denominados 'Hobby Servo ports'. La cámara ha sido conectada

a la Raspberry Pi en el puerto CSI (ver la figura 3.14). El resultado de estas tres piezas instaladas en el robot se puede apreciar en la figura 3.15.



Figura 3.14: Cámara Raspberry Pi conectada al puerto CSI de la tarjeta

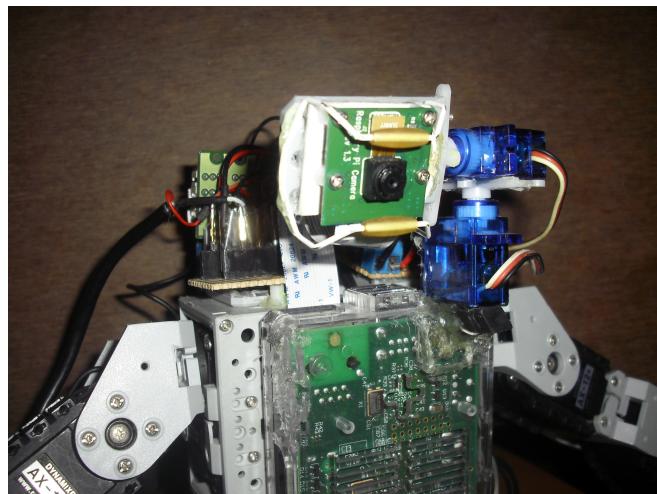


Figura 3.15: Vista delantera del robot con la cámara y servomotores instalados

Los motores Dynamixel se conectan a la controladora Arbotix por medio de los puertos Bioloid de la tarjeta. El diseño de Junny posee cuatro extremidades: dos brazos y dos piernas, por lo que naturalmente, el conjunto de motores, se puede descomponer en cuatro cadenas o series separadas. Sin embargo la Arbotix sólo cuenta con tres puertos Bioloid. Se consideró la opción de unir dos extremidades pero ello implicaba limitaciones en el movimiento del robot, por lo tanto se optó por agregar un expansor de puertos Bioloid y así conectar cada extremidad.

dad en un puerto diferente. La forma en la que se ha conectado estos motores se ejemplifica en la figura 3.16.

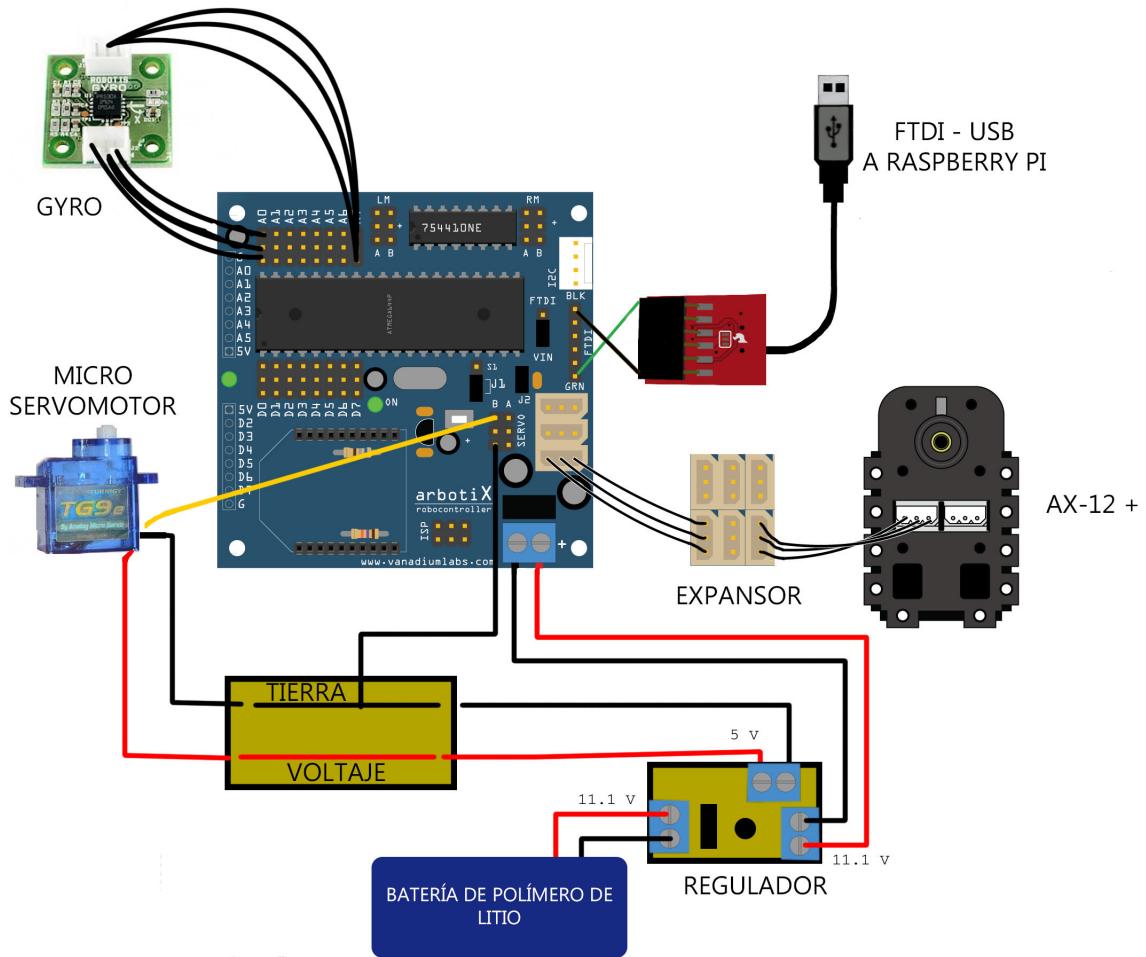


Figura 3.16: Tarjeta controladora Arbotix y componentes conectados (Imagen referencial: el regulador se encuentra dispuesto en diferente orden)

La comunicación de la tarjeta de Arbotix con la computadora, incluso con la Raspberry Pi, se realiza a través del puerto FTDI por medio un chip conectado como lo ilustra la figura 3.16 el USB es conectado a la raspberry Pi y el otro lado al puerto correspondiente en la Arbotix.

Para evitar agregar peso innecesario al robot se decidió solo colocar una sola fuente de poder, se utilizó una batería de polímero de litio de 11.1 V y 1 amp. Por medio del regulador

construido alimenta a todos los componentes simultáneamente (ver figura 3.10).

3.2. Detección de la pelota

La recopilación de información del medio ambiente, para detectar la posición de la pelota, se realizó por medio de la cámara Raspberry Pi con el sistema operativo Raspian. Esta mini computadora es una herramienta ya que permite capturar videos y fotos de alta definición. Como la Raspberry Pi cuenta con un procesador gráfico, este se encarga de manejar los datos de la cámara, aliviando la carga del procesador central [Asy].

Además se ha decidido utilizar OpenCv, otra herramienta para procesar imágenes que se describe en la sección 3.2.1. Sin embargo los métodos de captura de OpenCV no funcionan con la cámara Raspberry Pi. En la sección 3.2.2 se explica cómo se ha extraído la imagen y en la sección 3.2.3 se explica la forma en la que se ha procesado la imagen para hallar la posición de la pelota.

3.2.1. Herramientas software para la detección

Para extraer y procesar la imagen se utilizaron algunas librerías como apoyo. A continuación se presenta la descripción de la librería raspicam_cv, usada para la extracción de la imagen y la descripción de la librería OpenCV, usada para la detección de la ubicación de la pelota.

En visión de computadoras existen varias librerías que apoyan y facilitan en procesamiento de imágenes, con herramientas de filtrado, transformaciones de imágenes, aprendizaje de máquinas entre muchas más. Dos ejemplos de ello son Processing y OpenCv. Processing es un lenguaje de programación que está enfocado para iniciar a personas no programadoras en el área, por lo tanto explota la retroalimentación visual para atraer al usuario, posee herramientas de filtrado, de transformación de imágenes y muchas otras. Por otro lado OpenCv

(Open Source Computer Vision Library) es una librería de visión de computadoras y aprendizaje de máquinas de código abierto. Ha sido diseñada para acelerar el uso de la percepción de máquinas y para proveer una estructura común en las aplicaciones de visión de computadoras [Tea]. La decisión de utilizar OpenCv se basa en su mayor amplitud de herramientas ofrecidas, un filtrado de imágenes mas amplio, un mayor control en el manejo de las imágenes y sus transformaciones.

Raspicam_cv es una librería que permite obtener imágenes de la cámara Raspberry Pi en una estructura de datos compatible con OpenCV modificada por [Val13].

3.2.2. Obtención de la imagen

Dentro de las librerías oficiales para la cámara Raspberry Pi sólo se encuentran implementadas en el lenguaje interpretado python y algunas aplicaciones para la línea de comandos de linux. Para utilizar la cámara con OpenCV en el lenguaje compilado C++ se requirió realizar una búsqueda de librerías alternas a las oficiales. Una primera solución se encontró en el blog de [Rau13], en donde explica que los métodos de captura de video de OpenCV no funcionan de manera nativa con el módulo de la cámara de la Raspberry Pi (por ejemplo el método cvCapture). Para lograr extraer la imagen se basó en el código abierto de las aplicaciones raspivid y raspistill. Ha modificado el código para usar el buffer de la cámara y así obtener un objeto compatible con OpenCV. raspicam_cv es la librería utilizada en el proyecto que del código ha sido modificado convirtiéndola una librería por [Val13].

3.2.3. Procesamiento de la imagen

Con ayuda de la librería OpenCv, en C++, se filtra y procesa la imagen para obtener la posición de la pelota en un momento dado y de forma autónoma.

Para encontrar la ubicación de la pelota se ha decidido aplicar detección por segmentación

de regiones, esta técnica consiste en filtrar la imagen por segmentación de color, por ello es importante que el color de la pelota no se repita en el ambiente y así poder obtener su posición dentro de la imagen. Esta técnica de segmentación es la más común en detección de objetos y muy utilizada en competencias de robótica. La técnica de detección por formas fue puesta a prueba también pero con la misma ni se pudo obtener resultados prácticos ya que no lograba detectar correctamente la pelota. Vale la pena acotar que la detección por formas aplicada no implementaba ningún tipo de aprendizaje de máquina.

La imagen es captada en el modelo de color RGB y se transforma al HSV. Luego se aplica la función `inRange` de OpenCv para obtener una imagen en blanco y negro, en donde se identifica con blanco la zona con el color de la pelota y el resto de la imagen en negro. En el procesamiento de imágenes y visión de computación se trabaja con las imágenes en escala de grises pues disminuye el tiempo en procesar datos que poseen información inutil y aumenta la eficiencia.

Para disminuir el ruido y los posibles elementos aislados que pueda tener la imagen con la que se está trabajando se han aplicado los filtros o transformaciones de morfología en apertura y morfología en clausura de la librería OpenCV, basadas en las operaciones básicas de dilatación y erosión. La morfología en apertura es una transformación que consiste en aplicar la operación de erosión seguido de la operación de dilatación. La morfología de clausura es una transformación que aplica la dilatación seguido de la erosión.

De esta forma se logró ubicar la pelota con la cámara Raspberry Pi en 100 % de las pruebas realizadas.

3.3. Búsqueda y Pateo

Para poder buscar y patear la pelota, además de tener la capacidad para detectarla (como se explicó en la sección anterior), debe ser capaz de moverse en su entorno, poder

patear, levantarse, tener una representación del mundo que lo ayude a orientarse y tener una estrategia para elegir el conjunto de movimientos que lo lleven a acercarse a la pelota.

En esta sección se explica el desarrollo de las actividades que han sido necesarias para ejecutar la búsqueda y pateo de la pelota.

Primero, en la sección 3.3.1 se da una breve descripción de las herramientas de software que apoyaron las tareas de búsqueda y pateo. En la sección 3.3.2 se explica cuál fue el conjunto de movimientos creados para el esqueleto del robot.

El sensor principal de Junny, es el observador de su ambiente, la cámara. Ésta tiene dos grados de libertad para su movimiento, lo que causa que tenga un gran número de posibles posiciones. Para simplificar, se ha limitado la cantidad de posiciones. En la sección 3.3.3 se explica las posiciones que puede adoptar la cámara. Luego en la sección 3.3.4 se explica la manera en la que Junny organiza la representación visual que capta del mundo.

Debido a que el movimiento del robot se controla desde la tarjeta Arbotix y la detección de la pelota se hace desde la Raspberry Pi, se debió establecer la comunicación entre ambas tarjetas. Este proceso se explica en la sección 3.3.5.

Una vez con la representación del mundo, los movimientos programados y la comunicación de las tarjetas, solo faltaría decidir qué acción tomar en cada situación. La estrategia, basada en el paradigma híbrido, se explica en la sección 3.3.6.

3.3.1. Herramientas software para la búsqueda de la pelota

Para cumplir con el desarrollo de la programación de la búsqueda y pateo de la pelota, se han utilizado algunas herramientas que han facilitado el proceso. A continuación se describen las más destacadas dentro del proyecto.

Pypose es un software especializado en el control de los servomotores Dynamixel Ax-12. Esta es la opción que ofrecen los desarrolladores de la tarjeta Arbotix para su programación.

Una de las más importantes características es que, luego de haber fijado a mano las posiciones de los motores, permite la lectura simultánea de esas posiciones para captar la pose del robot. Con esta herramienta es posible formar una secuencia de poses que generen un movimiento, por ejemplo, caminar [Fer10].

Cuando los movimientos han sido implementadas por medio de pypose fue necesario su utilización en un ambiente más versatil por lo cual se utilizó el IDE Arduino que es un entorno de desarrollo para escribir y cargar código en la tarjeta Arduino. Otras tarjetas con microcontroladores AVR también son compatibles, como la Arbotix. El lenguaje de programación del IDE de Arduino es una implementación de Wiring el cual está basado en Processing [Ard14].

- ROS: ROS (Robot Operating System) es un Framework flexible para la escritura de software enfocado en robots. Es una colección de herramientas, bibliotecas y convenciones que tienen por objeto simplificar la tarea de crear un comportamiento complejo y robusto a través de una amplia variedad de plataformas robóticas. ROS se encuentra bajo licencia de código abierto, la licencia BSD [Ros].
- HServo: Es una librería de código libre distribuida bajo la Licencia Pública General Reducida de GNU. Esta librería se encuentra desarrollada específicamente para la tarjeta Arbotix, ya que con la nueva librería de Arduino Servo se pueden experimentar fallas en el control de los servos. La interfaz es la misma, la diferencia es que solo puede ser usada para servos conectados en los pines 12 y 13 de la tarjeta Arbotix [Van].

3.3.2. Movimiento del Cuerpo

El robot debe tener la capacidad de ejecutar una variedad de movimientos para poder cumplir la meta de patear la pelota. Por ello se ha programado un conjunto de poses y transiciones o acciones de movimiento que se explican a continuación.

Con fines explicativos, en este proyecto, la palabra “pose” se referirá a la posición específica de los 16 motores que constituyen el esqueleto del robot. Un conjunto de poses ejecutadas en secuencia se denominará “acción de movimiento”.

Las acciones de movimiento establecidas son:

- Caminar dos pasos hacia adelante (4.8 cm)
- Girar a la izquierda (3 cm)
- Girar a la derecha (3 cm)
- Levantarse desde la posición boca abajo
- Levantarse desde la posición boca arriba
- Patear con la pierna derecha
- Patear con la pierna izquierda

En el movimiento de caminar se establece como unidad el de un paso que fue construido manteniendo el equilibrio del robot para evitar posibles caídas e inestabilidad la distancia recorrida por ese paso es de 4.8 cm, analógicamente se obtiene las distancias recorridas de los giros de derecha e izquierda.

Estas poses han sido fijadas a través de la tarjeta controladora Arbotix y el software Pypose. De esta manera se ha fijado y guardado un conjunto de poses para cada acción de movimiento. Estas acciones de movimientos han sido exportadas para ser utilizadas en el programa, en lenguaje Wiring, a ser ejecutado en Arbotix. La programación en Arbotix se ha realizado bajo el ambiente del IDE de Arduino.

3.3.3. Movimiento de la cámara

La cámara ha sido instalada sobre dos micro servomotores analógicos, otorgándole dos grados de libertad. El servomotor ubicado en la parte inferior se encarga del movimiento horizontal y el superior, del movimiento vertical.

El hecho de que la cámara tenga dos grados de libertad para moverse es una gran ventaja, ya que se puede obtener un mayor rango de visión. Junny puede mirar hacia la derecha o izquierda sin tener que mover sus piernas, también puede mirar hacia abajo para verificar que la pelota esté en sus pies, para patear, o hacia arriba para ubicar la pelota a mayor distancia. Se consideró otorgarle un tercer grado de libertad a la cámara pero ello implicaba mayor espacio, peso y costo que no se justificaba para los objetivos del proyecto. También si se colocaba la cámara fija, sin grados de libertad, implicaba movimientos constantes que consumían mucho más tiempo y desgaste, además de menos rango de visión de Junny.

El número de posibles posiciones para la cámara es muy amplio y para simplificar se ha reducido a 6 posiciones fijas, cuya distribución obedece al objetivo de que la cámara obtenga una amplia visión, sin dejar espacios no visibles. Esta simplificación ayuda en la tarea de la representación del mundo que se explica en la sección 3.3.4. En la figura 3.17 se puede observar la cámara del robot en las diferentes posiciones que se han fijado para ella.

Los micro servomotores azules que aparecen en la imagen 3.17 se controlan desde la Arbotix usando la librería HServo. Esta librería sólo puede ser usada para los motores conectados en los puertos Hobby A y B (pines 12 y 13) (ver la figura 3.16). Tener los motores conectados a estos puertos brinda la ventaja de un control más preciso, evitando que los motores generen vibración, ya que los pulsos son generados por temporizadores de hardware.

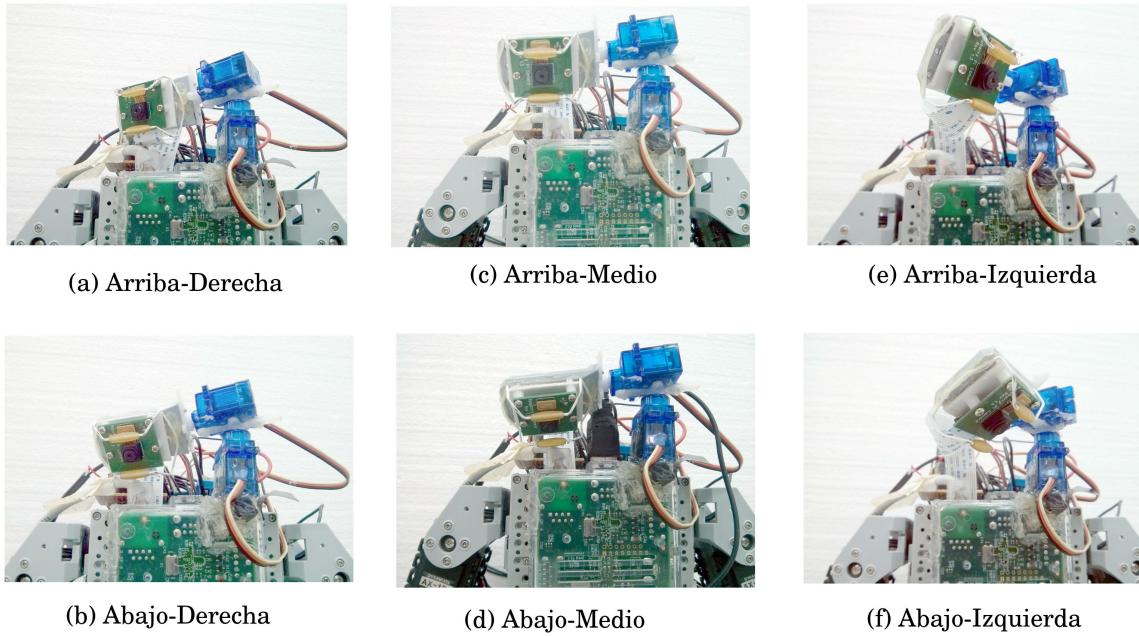


Figura 3.17: Posiciones de la cámara

3.3.4. Representación del mundo

La representación del mundo de Junny se basa en función de la posición de la pelota con respecto al él. La siguiente representación de la visión ha sido elegida analizando la estructura física del robot, su alcance de visión y su movilidad para obtener el mejor desempeño posible. La cámara tiene 6 (2x3) posibles posiciones, como se muestra en la figura 3.17 y desde cada posición de la cámara se obtiene una imagen. La detección de la pelota en alguna de esas imágenes brinda una orientación sobre la ubicación de la pelota. Si se detecta la pelota, por ejemplo, cuando la cámara se encuentra en alguna de las posiciones de la derecha, es un indicativo de que la pelota se encuentra a la derecha y que si el robot gira se podría posicionar frente a ella.

Se ha decidido discretizar las posibles posiciones de la pelota por regiones. Estas regiones se muestran enumeradas en la figura 3.18. Cada cuadro blanco demarcado por líneas negras representa la imagen captada en una posición fija de la cámara.

Las imágenes de la cámara en la posición central superior e inferior son las más importantes y prioritarias, pues si la pelota se detecta en ellas significa que el robot está cerca de poder patearla. Estas dos imágenes se dividen en subregiones para tener mayor precisión en las acciones que Junny deba tomar.

Cuando, por ejemplo, la pelota se encuentra del lado derecho en el cuadro central inferior (región 5) el robot debería girar a la derecha para situarse de frente a la pelota. El área de pateo se encuentra en las regiones 1 y 2.

Las imágenes capturadas desde cada posición de la cámara se solapan un poco para evitar perder de vista a la pelota. Este solapamiento no se muestra en la imagen 3.18 para simplificar su representación.

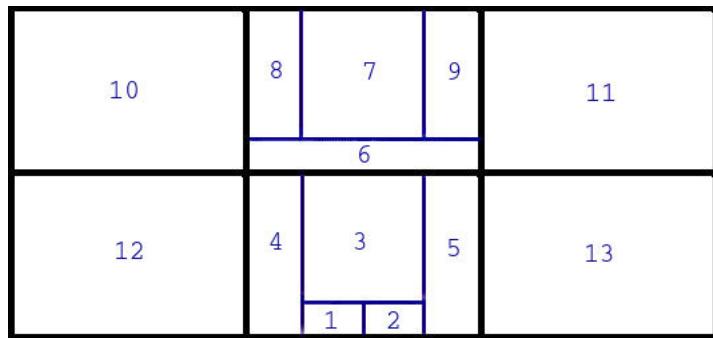


Figura 3.18: Campo de visión del robot con el número de cada región. Cada cuadro blanco demarcado con líneas negras representa la imagen captada desde una posición fija de la cámara. La región de pateo de la pelota se encuentra en las regiones 1 y 2

3.3.5. Comunicación Arbotix - Raspberry (ROS)

La Raspberry Pi procesa la información de la cámara y la Arbotix controla los actuadores. Para coordinar los movimientos del robot según la posición de la pelota se estableció una forma de comunicación entre ambas tarjetas.

Se ha establecido la Arbotix como servidor de peticiones y a la Raspberry Pi como cliente. Dentro de la Raspberry Pi se ejecuta el proceso de decidir qué acción debe tomar el robot.

Una vez determinada la acción se envía la petición a la Arbotix para que esta la ejecute. Este proceso es bidireccional y síncrono, es decir, la Raspberry envía la petición y se bloquea hasta que la Arbotix retorne la respuesta de su culminación.

Para la implementación de la comunicación se ha usado ROS con su versión Hydro y se ha utilizado la interfaz de comunicación basada en servicios que no es más que un método de comunicación basado en el paradigma de resquest / reply con el concepto de maestro esclavo.

3.3.6. Elección de la acción

Una vez con la representación del mundo, los movimientos programados y la comunicación de las tarjetas, solo faltaría decidir que acción tomar en cada situación. En primera instancia se ha decidido fijar una acción por cada región en la que se detecte la pelota.

cuando la pelota se encuentre a la izquierda del robot (regiones 4,8,10 y 12) junny debe girar a la izquierda para centrar la pelota.

A continuación se especifica la acción a tomar en cada región:

- Cuando pelota se encuentra a la izquierda de Junny como ocurre en las regiones 4, 8, 10 y 12, este debe girar a la izquierda para centrar la pelota con su cuerpo.
- Análogo al caso anterior cuando pelota se encuentra a la derecha de Junny como ocurre en las regiones 5, 9, 11 y 13, este debe girar a la derecha para centrar la pelota con su cuerpo.
- Cuando la pelota se ubique en alguna de estas regiones 3, 6 y 7, el robot debe caminar hacia adelante para reducir la distancia a la pelota.
- Al llegar al estado meta que se refiere a la región 1 Junny debe patear con la pierna izquierda para meter gol.

- Al llegar al estado meta que se refiere a la región 2 Junny debe patear con la pierna derecha para meter gol.

En caso de no encontrar la pelota en ninguna de las regiones, el robot gira con los pies para cambiar su orientación física e iniciar nuevamente el movimiento de la cámara para hallar la pelota. Cuando se tiene la pelota en una posición cercana a los pies se realiza la acción de patear.

3.4. Aprendizaje

En el área de inteligencia artificial, asociada a robótica, existen variadas técnicas que permiten que un robot pueda aprender a realizar alguna tarea. En este caso particular se utilizó la técnica de aprendizaje por reforzamiento que consiste en dar recompensas positivas o negativas dependiendo del desempeño del robot.

Según [Mit97] todo aprendizaje se define por la realización de una tarea T , cuyo desempeño medido por D , mejora con la experiencia E . En la investigación presente la tarea T es aprender cuál es el mejor conjunto de “acciones de movimiento” que se debe tomar con la finalidad de acercarse a la pelota. La experiencia E se da a través un conjunto de pruebas, en las que Junny debe buscar la pelota y posicionarse frente a ella. El desempeño D se mide con respecto a si el robot logra posicionar la pelota en el área de pateo y cuántas acciones de movimiento son necesarias para lograrlo.

Como existe incertidumbre con respecto a la dinámica del ambiente (esto es, no se conoce la función $\delta(s, a)$, definida en la subsección 2.3.3 se utilizó el modelo de aprendizaje Q-learning, el cual permite mejorar el desempeño de la tarea definida en base a la experiencia de cada prueba. A continuación se presenta y describe la configuración de las características particulares del aprendizaje utilizado para este proyecto.

3.4.1. Modelo del problema

El algoritmo para Q-learning se adapta a problemas configurados como procesos de decisión Markovianos (MDP). En este tipo de problemas el resultado de aplicar una acción en un estado particular depende solo de ese estado y esa acción, no de las acciones del pasado.

En un MDP, un agente representa su mundo a través de un conjunto de estados y tiene un conjunto de acciones que puede ejecutar. Cada vez que el agente identifica el estado en el que se encuentra y escoge una acción para tomar, dependiendo del resultado, este recibe una recompensa determinada.

En la sección 3.4.1.1 se define el conjunto de estados que conforman el mundo de Junny. En la sección 3.4.1.2 se dan a conocer las posibles acciones definidas y en la sección 3.4.1.3 se define la ecuación para calcular las recompensas.

3.4.1.1. Estados

De forma general se puede decir que los estados se encuentran definidos en función de la posición de la pelota con respecto al robot. En la sección 3.3.4 se ha explicado la manera en la que se representa el mundo en base a 13 regiones en las que se puede encontrar la pelota. Para el aprendizaje se ha decidido ampliar el número de regiones para afinar y mejorar el tiempo en el que se llega a la pelota. Estas regiones se muestran en la figura 3.19.

Cada región en la que se pueda detectar la pelota es un estado diferente. Por lo tanto se generan 18 estados, 17 de ellos se corresponden con la detección de la pelota en cada una de las regiones que se muestran en la imagen 3.19 y el estado número 18 representa la ocasión en la que no se logra ubicar la pelota en ninguna de las regiones.

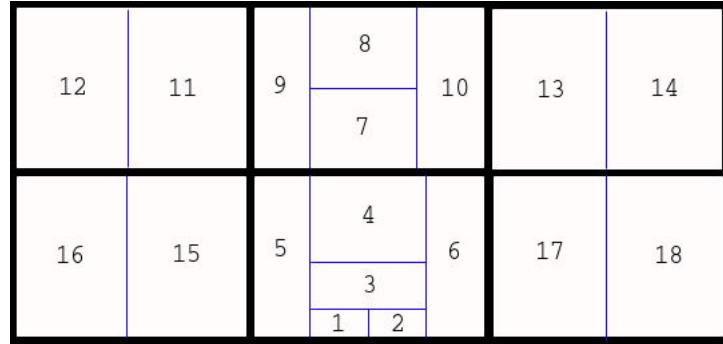


Figura 3.19: Estados definidos según la región en la que se detecta la pelota

3.4.1.2. Acciones

Las posibles acciones a realizar son un subconjunto de las acciones de movimiento definidas en la sección 3.3.2. Además se agregaron otras acciones para mejorar el desempeño en la búsqueda de la pelota. Las acciones disponibles son:

- Caminar un paso hacia adelante (2.5 cm)
- Caminar dos pasos hacia adelante (4.8 cm)
- Caminar cuatro pasos hacia adelante (9.9 cm)
- Girar a la izquierda (3 cm)
- Girar doble a la izquierda (6 cm)
- Girar a la derecha (3 cm)
- Girar doble a la derecha (6 cm)

3.4.1.3. Recompensas

La recompensa, que puede ser positiva o negativa, se otorga a un par (s, a) , en donde s es un estado y a es una acción. Se calcula en base a que tanto el robot se acerca a la pelota

cuando en el estado s se toma la acción a . Es decir, las recompensas se definen en base a la distancia recorrida, con respecto a la pelota, cuando se toma una acción.

La distancia de una región con respecto al área de pateo, se define con la función $d : r \rightarrow y$ con $r \in \{1, 2, 3.., 18\}$ y $y \in \{1, 2, 3.., 10\}$. En donde r es el número de la región y y es la distancia de la región con respecto al área de pateo. Se asignó una distancia a cada región como se muestra en la figura 3.20. El valor 1 se le asigna a las regiones más cercanas al robot (zonas de pateo), $d(1) = 1$ y $d(2) = 1$; el valor 10, a las regiones más lejanas (regiones 12 y 14). Adicionalmente se define la distancia de la región 18 como $d(18) = 10$.

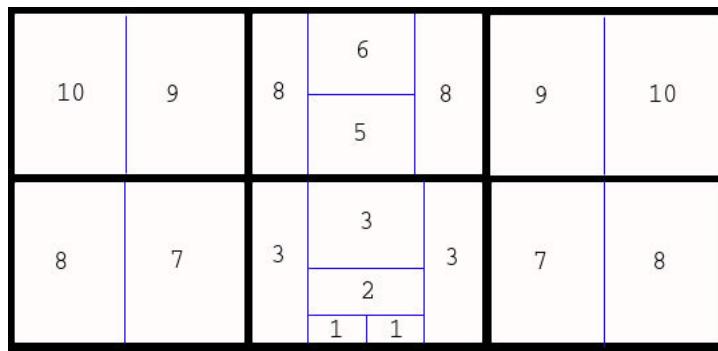


Figura 3.20: Distancias relativas de la pelota establecidas para otorgar la recompensa.

Junny gana una recompensa positiva cuando, con una acción, la pelota pasa de una región de mayor valor (lejana) a una de menor valor (más cercana). Si la pelota se mantiene en la misma región obtiene recompensa 0. De lo contrario obtiene una recompensa negativa. La ecuación se define como:

$$R(s, a, s') = \frac{d(s) - d(s')}{10}$$

El rango de valores para la recompensa se encuentra entre -1 y 1.

3.4.2. Elección de la acción

Se entiende que dado un estado s se tienen 7 posibles acciones $\{a_1, a_2, \dots, a_7\}$ que, en el entrenamiento del robot, él aprende cuál es la mejor a realizar según el estado en el que se

encuentra. A mayor valor de $Q(s, a)$ mejor es la acción a . Si siempre se toma el máximo valor se favorece la explotación. Si se toman acciones aleatorias se favorece la exploración. Para que el aprendizaje obtenga buenos resultados se debe tener un equilibrio entre la exploración y la explotación.

Para variar entre la explotación y la exploración se utilizó la función de probabilidad definida como

$$P(a_i|s) = \frac{k^{Q(s,a_i)}}{\sum_j k^{Q(s,a_j)}}$$

Con valores diferentes de k para cada conjunto de pruebas, como se explica en el capítulo 4.

3.4.3. Actualización de $Q(s,a)$

La actualización de $Q(s,a)$ se define por la siguiente formula

$$Q(s, a) = r + \gamma \max_{a'} Q(\delta(s, a), a')$$

Donde r y $\max_{a'} Q(\delta(s, a), a')$ se explican en la subsección 2.3.3 el factor γ es el factor de descuento que varia de

$$0 \leq \gamma < 1$$

este parámetro es arbitrario y se debe ajustar durante el aprendizaje, en el capítulo 4 se presentan los γ utilizados para los entrenamientos.

3.5. Detección de caídas

Para poder detectar una posible caída del robot se cuenta con el giroscopio, incluido en el kit Bioloid de Robotis. Este giroscopio brinda información sobre la velocidad angular en

los ejes X y Y como se muestra en la figura 3.21.

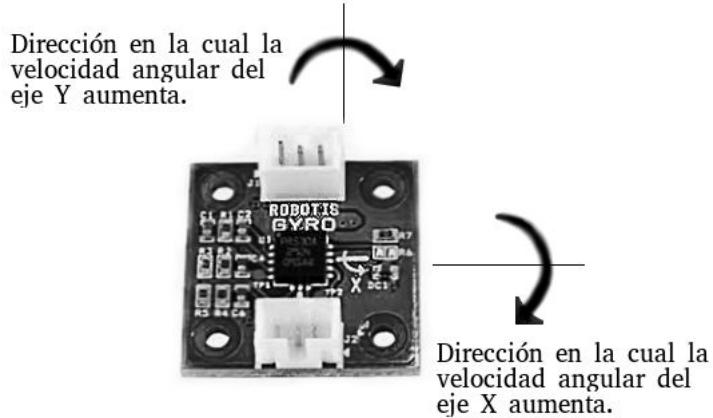


Figura 3.21: Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.

La información del eje utilizado en este proyecto es el X , que indica si el robot experimenta una velocidad angular hacia adelante o hacia atrás. En la figura 3.22 se muestra la dirección en la que crece la velocidad angular con respecto al robot. El giroscopio puede detectar la velocidad desde $-300^{\circ}/s$ hasta $300^{\circ}/s$. Según la documentación, los valores que se leen del Gyro se dan en un rango de 45 a 445, donde el valor 45 representa los $-300^{\circ}/s$ y el 455 representa los $300^{\circ}/s$. El valor 250 indica que no hay movimiento angular, es decir $0^{\circ}/s$.

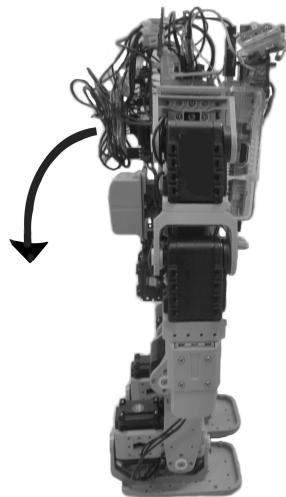


Figura 3.22: Dirección en la que la velocidad angular del eje X crece con respecto al robot.

Para detectar una posible caída ha sido necesario verificar constantemente la velocidad angular del eje X. Se ha establecido un límite inferior y uno superior para identificar dentro de qué valores puede estar la velocidad angular sin que se considere como una caída del robot, sino como efecto de su movimiento. En caso de estar fuera de los límites, se considera como una caída y el robot ejecuta la acción de movimiento para levantarse.

Se han observado los valores del giroscopio con el robot estático para verificar si los valores leídos se corresponden con los de la documentación, es decir 250 cuando no hay movimiento. Para obtener información más confiable se realiza un promedio entre diez lecturas para cada valor que se toma en cuenta. El resultado ha sido que los primeros valores no inician en 250 sino, en aproximadamente 284 y luego va disminuyendo. Por este motivo se decidió agregar un tiempo de 15 segundos del robot en reposo antes de comenzar la búsqueda de la pelota, de esta manera los valores del giroscopio se estabilizan.

Después de la pausa de 15 segundos, el primer valor promediado se toma como el valor que representa la velocidad angular cero, a este se le llamará valor central. Los siguientes valores se comparan con el valor central. Si la diferencia se encuentra dentro del rango ($-80, 100$) significa que los movimientos no han sido tan bruscos como para considerarse una caída. De lo contrario, si se obtienen valores fuera del rango, se considera una caída y se ejecuta la acción de levantarse.

3.6. Orientación al arco

Al finalizar la búsqueda de la pelota, se desea poder patearla con dirección al arco. El procedimiento para la búsqueda del arco es análogo al procedimiento de la búsqueda de la pelota explicado en la sección 3.3. Se tiene la representación del mundo dividida en regiones como se muestra en la figura 3.24. Los recuadros negros representan la imagen captada por las posiciones de la cámara (a) (c) y (e) que se muestran en la imagen de la figura 3.17. En

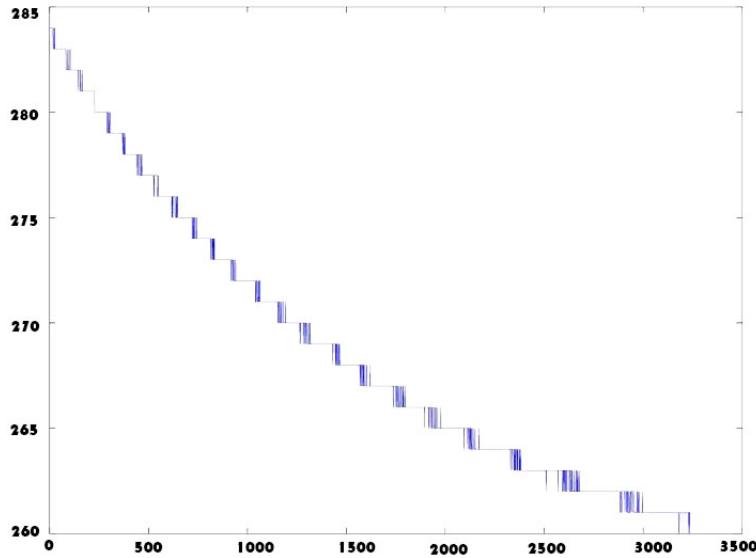


Figura 3.23: Gráfica de lecturas de velocidad angular en el eje X.

esta ocasión se cuenta con 8 estados. Siete de ellos cuando el arco se detecta en alguna de las regiones y el último estado representa la ocasión en la que no se encuentra el arco.

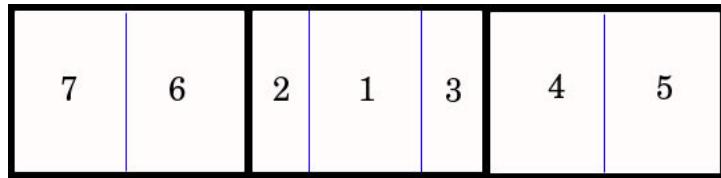


Figura 3.24: Campo de visión del robot con el número de cada región para buscar el arco.

Primero se busca detectar el arco y posicionarse frente a él intentando rodear la pelota, luego se verifica que aún la pelota se encuentre en la zona de pateo y se procede a patear hacia el arco.

Las acciones que se toman en cada región se especifican a continuación:

- Girar a la Izquierda: Junny debe girar a la izquierda cuando el arco se encuentre en alguna de las siguientes regiones: 2, 6, 7.
- Girar a la Derecha: debe girar a la derecha cuando el arco se ubique en alguna de las siguientes regiones: 3, 4 y 5.

Cuando el arco se encuentra en la región 1. Se da por finalizada la búsqueda y posicionamiento frente al arco.

Los resultados obtenidos en esta tarea en particular no han sido muy favorables, pues en casos en los que el arco ya se encuentra frente al robot, este logra ajustarse con la pelota para realizar un buen tiro. Sin embargo para el rodeo de la pelota es importante poseer un grado de libertad que Junny no posee en las caderas, lo cual dificulta lograr un buen desempeño.

Capítulo 4

Experimentos y Resultados

En este capítulo se describen los experimentos realizados para analizar el desempeño del robot. Los experimentos se han dividido en tres partes. El primer conjunto de experimentos (sección 4.1) se ha realizado para verificar el desempeño y balance del robot al ejecutar uno o varios movimientos en secuencia. El segundo y tercer conjunto de experimentos se basó en verificar el desempeño del robot al buscar la pelota. En el segundo conjunto (sección 4.2) la forma de escoger las acciones de movimiento han sido fijadas para cada región en la que se detecta la pelota. Mientras que para el tercer conjunto de experimentos (sección 4.3) la forma de escoger las acciones de movimiento ha sido resultado del aprendizaje por reforzamiento.

4.1. Experimentos de Movimientos

Se realizaron varios experimentos para comprobar la movilidad del robot en cuanto a un conjunto de acciones de movimiento (mencionadas en la sección 3.3.2). Estas acciones son: caminar hacia adelante (4.8 cm), girar hacia la derecha (3 cm), girar hacia la izquierda (3 cm), patada con la pierna derecha, patada con la pierna izquierda, levantarse en la posición supino, levantarse en posición prono.

La primera etapa consistió en verificar el desempeño de las unidades de acciones, para ello se procedió a realizar la ejecución de cada unidad 50 veces, 250 ejecuciones en total. Se tomó nota de las veces que lograba realizar el movimiento sin fallas, es decir, de forma exitosa y de aquellas que lograba completar el movimiento pero con fallas. Los resultados obtenidos fueron un 99.6 % de casos exitosos y un 0.4 % de casos en los que pudo recuperarse de fallas. El 0.4 % representa un solo caso en el que el robot se ha caído y levantado.

En la segunda etapa se evaluó la ejecución de combinaciones de varias unidades de movimiento. Las combinaciones elegidas fueron:

- Caminar hacia adelante y patear con la pierna derecha
- Caminar hacia adelante y patear con la pierna izquierda
- Caminar hacia adelante, girar hacia la derecha y patear con la pierna derecha
- Caminar hacia adelante, girar hacia la derecha, patear con la pierna izquierda
- Caminar hacia adelante, girar hacia la izquierda y patear con la derecha
- Caminar hacia adelante, girar hacia la izquierda y patear con la pierna izquierda
- Girar hacia la izquierda y patear con la pierna derecha
- Girar hacia la derecha y patear con la izquierda
- Girar hacia la derecha y patear con la pierna derecha
- Girar hacia la izquierda y patear con la pierna izquierda

Se realizaron 30 ejecuciones de cada una de estas combinaciones, 300 ejecuciones en total. Los resultados se muestran en el gráfico de la figura 4.1. El resultado ha sido satisfactorio, obteniendo un 96 % de casos exitosos y un 4 % de casos con fallas, de las que ha podido recuperarse. La mayoría de las fallas se presentaron cuando al patear se caía, sin embargo lograba recuperarse.

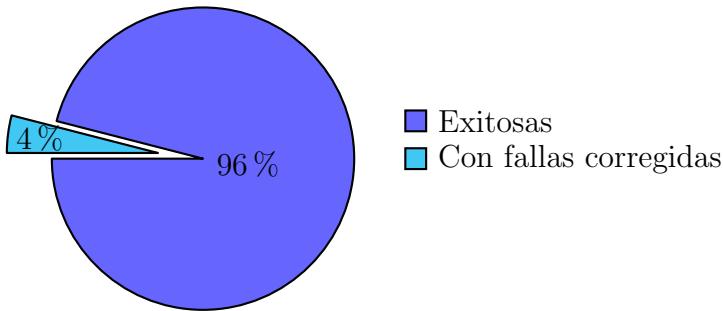


Figura 4.1: Segunda etapa: 300 ejecuciones de acciones combinadas

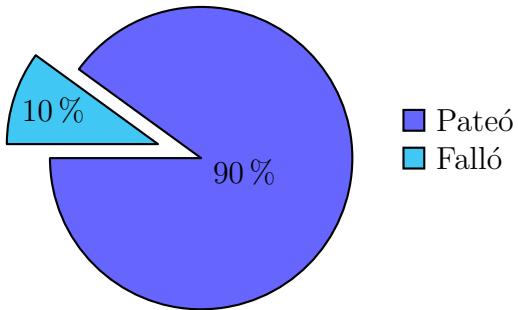


Figura 4.2: Pruebas de movimientos integrados CASO I

4.2. Experimentos con Comportamientos Integrados

El segundo conjunto de experimentos consistió en la realización de pruebas de desempeño en donde se observó el comportamiento global del robot con todos sus comportamientos integrados (detección, búsqueda y pateo). La forma de elegir la acciones de movimiento es la que se describe en la sección 3.3.6 (sin aprendizaje).

El primer caso (CasoI) consistió en la colocación de la pelota en la zona de pateo (las regiones 1 y 2 de la figura 3.18). El número de pruebas en este primer caso fue de 10, cuya duración aproximada de cada prueba fue de 30 segundos. En la figura 4.2 se puede observar los resultados obtenidos de esta prueba.

El segundo caso (CasoII) consistió en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot. Se realizaron 10 pruebas, siendo la duración promedio de 2 minutos con 12 segundos. Los resultados de dicho caso se pueden observar en la figura 4.3.

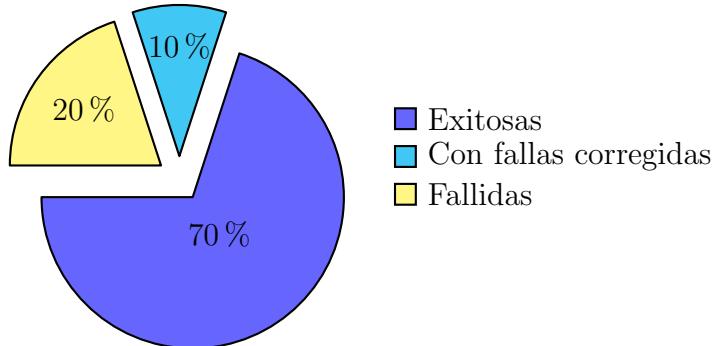


Figura 4.3: Pruebas de movimientos integrados CASO II

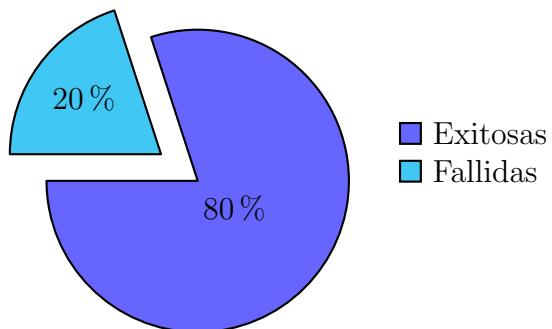


Figura 4.4: Pruebas de movimientos integrados CASO III

El tercer caso (CasoIII) consistió en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot y 10 cm a la izquierda formando así una diagonal. Se realizaron 10 pruebas, siendo la duración promedio de 3 minutos con 29 segundos. Los resultados de dicho caso se pueden observar en la figura 4.4.

El cuarto caso (Caso IV) consistió en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot y 10 cm a la derecha. Se realizaron 10 pruebas, siendo la duración promedio de 4 minutos con 48 segundos. Los resultados de dicho caso se pueden observar en la figura 4.5.

Con un total de 40 pruebas de comportamiento integrado, los resultados obtenidos han sido satisfactorios con un porcentaje de logro del 82 % mas un 5.5 % en el cual se logró recuperar y finalizar la tarea, con 12.5 % de fallas en la tarea se pudo observar de manera general que la razón de dichas fallas ha sido ocasionada por problemas de batería.

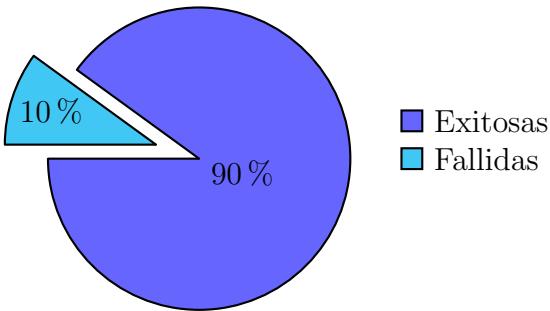


Figura 4.5: Pruebas de movimientos integrados CASO IV

4.3. Experimentos con Aprendizaje

Finalmente, el tercer conjunto de experimentos consistió en verificar los resultados de la aplicación del aprendizaje por reforzamiento, para la búsqueda de la pelota. Estos resultados se presentan a continuación.

Dentro de las fórmulas para el aprendizaje-Q se encuentran dos constantes que han sido configuradas de diferentes maneras para analizar cuál de las combinaciones brinda mejores resultados. Las constantes son la tasa de descuento γ de la fórmula del aprendizaje-Q y la constante K , de la fórmula de probabilidad explicada en la sección 3.4.2, que ajusta la distribución de probabilidad que se le da a una acción dependiendo de su valor $Q(s, a)$. Se utilizaron las siguientes combinaciones para las constantes:

- $K = 1, \gamma = 0.1$
- $K = 2, \gamma = 0.1$
- $K = 2, \gamma = 0.7$
- $K = 3, \gamma = 0.1$
- $K = 3, \gamma = 0.7$
- $K = 5, \gamma = 0.1$

- $K = 5, \gamma = 0.7$

Una prueba completa o prueba de comportamiento integrado se define como la colocación del robot y la pelota en posiciones iniciales arbitrarias, entonces el robot debe ser capaz de detectar la pelota, trasladarse hacia ella y patearla. Las pruebas se realizaron en un espacio de $1,8m \times 1,6m$ en donde se aseguraba que el color particular de la pelota no se repitiese, de ser así esa prueba se consideraba inválida.

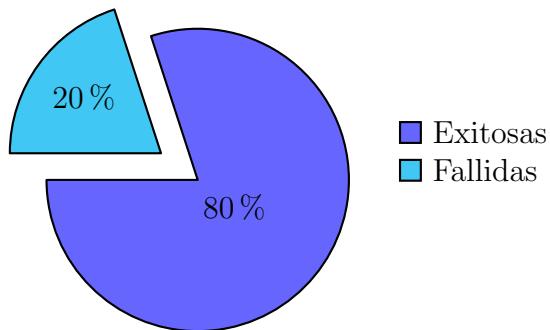
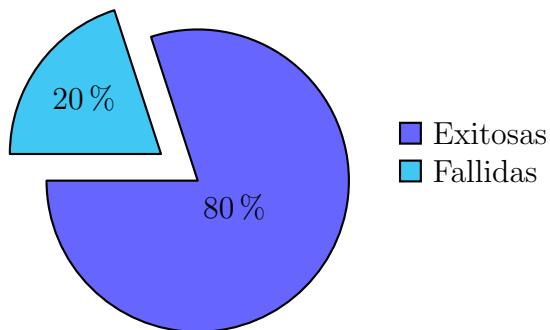
Cada una de estas combinaciones se entrenó con 20 ejecuciones completas. Con los valores $Q(s, a)$ para cada estado y acción inicializados en cero (0), exceptuando los estados de la zona de pateo, que han sido inicializados con el valor 1. Una vez completados los entrenamientos se realizó un conjunto de pruebas para medir el desempeño del resultado de cada combinación y reconocer la mejor usando el porcentaje de pateos efectivos. Se realizó una prueba, de 10 ejecuciones cada una, para cada combinación. En total se realizaron 210 ejecuciones.

Para la combinación de $K = 1$ y $\gamma = 0,1$ los resultados obtenidos se presentan en la figura 4.6. Cuando $K = 1$ la probabilidad de elegir la acción, dado un estado, es igual para todas las acciones, por lo tanto la elección de la acción para este caso ha sido uniformemente aleatoria.

Los resultados del resto de las combinaciones de parámetros ajustables se presentan en las siguientes figuras: para $K = 2$ y $\gamma = 0,1$ en la figura 4.6, $K = 2$ y $\gamma = 0,7$ en la figura 4.8, $K = 3$ y $\gamma = 0,1$ en la figura 4.9, $K = 3$ y $\gamma = 0,7$, $K = 5$ y $\gamma = 0,1$ en la figura 4.10 y por último $K = 5$ y $\gamma = 0,7$ en la figura 4.12.

Los resultados de los entrenamientos arrojaron que la mejor combinación fue $K = 3$ y $\gamma = 0,1$ que obtuvo un desempeño favorable del 90 % de las pruebas realizadas.

Con el objetivo de lograr un mejor desempeño de Junny se inicializó los valores de $Q(s, a)$ con valores de recompensas negativas para aquellas acciones cuyos efectos, en un estado particular, serían obviamente erróneos. Con esto se ayuda al aprendizaje del robot, disminuyendo

Figura 4.6: Prueba con $K = 1$ y $\gamma = 0,1$ Figura 4.7: Prueba con $K = 2$ y $\gamma = 0,1$

la probabilidad de tomar acciones ineficientes. Además se aumentó el número de pruebas de entrenamiento a 30 con el mismo objetivo.

Para medir el desempeño general de Junny, con respecto al último entrenamiento, se realizó una prueba de 15 ejecuciones colocando la pelota a distintas distancias. Los resultados de esta última prueba se presenta en el gráfico 4.13.

Para un análisis más profundo de los resultados, se estableció el número estimado de acciones esperadas, dado un estado inicial, que debe realizar el robot para llegar hasta la pelota. Con esto se puede tener una referencia de cuantas acciones debió realizar Junny en cada ejecución y comparar con el número de acciones que realmente tomó. La manera de calcular la eficiencia ha sido dividiendo la cantidad de acciones esperadas entre la cantidad de acciones realizadas. Por lo tanto mientras mas cercano a (1) se considera una mejor tasa de eficiencia. Para valores mayores a uno (1) significa que se realizaron menos acciones de lo

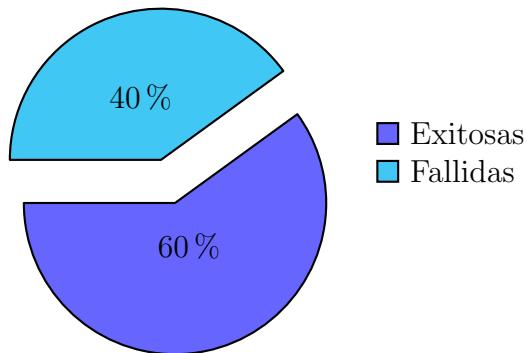


Figura 4.8: Prueba con $K = 2$ y $\gamma = 0,7$

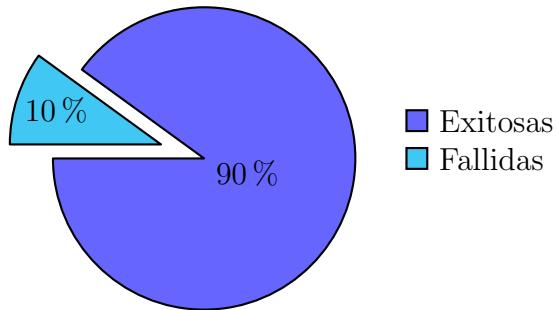


Figura 4.9: Prueba con $K = 3$ y $\gamma = 0,1$

esperado. Esto es posible ya que el número de acciones esperadas se calcula en base al peor de los casos.

Para la última prueba se obtuvo una eficiencia de 0,72. Sin embargo bajo los mismos parametros de $K = 3$ y $\gamma = 0,1$ con el resultado del entrenamiento con 20 pruebas (y valores de $Q(s, a)$ inicializados en cero) se obtuvo una eficiencia de 0,64. Por lo tanto con el aumento de 10 pruebas de entrenamiento e inicialización de los valores $Q(s, a)$ se mejoró aproximadamente un 12 %. Este es un indicativo de que con mayor cantidad de pruebas este porcentaje mejorará y la eficiencia aumentará.

Así mismo al comparar los tiempos promedios, con respecto a la última prueba, se obtienen los siguientes resultados. El tiempo promedio de las pruebas cuyas acciones esperadas eran menor a 8 (estados mas cercanos) fue de 4m y 19s en contraste con el promedio del tiempo de las pruebas cuyas acciones esperadas eran mayores a 8 que fue 3m y 21s.

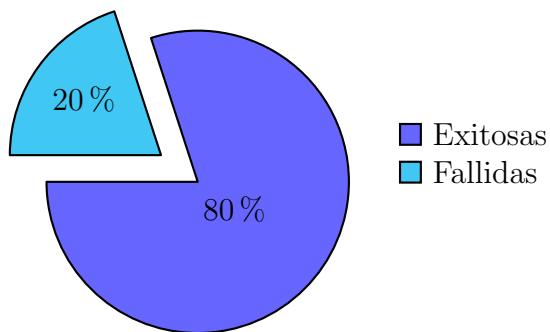


Figura 4.10: Prueba con $K = 3$ y $\gamma = 0,7$

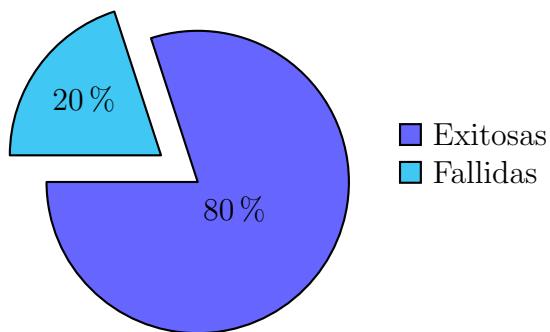


Figura 4.11: Prueba con $K = 5$ y $\gamma = 0,1$

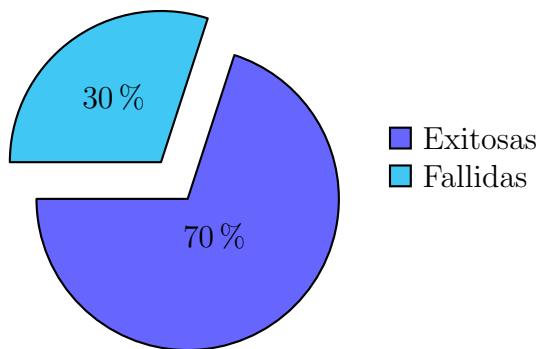


Figura 4.12: Prueba con $K = 5$ y $\gamma = 0,7$

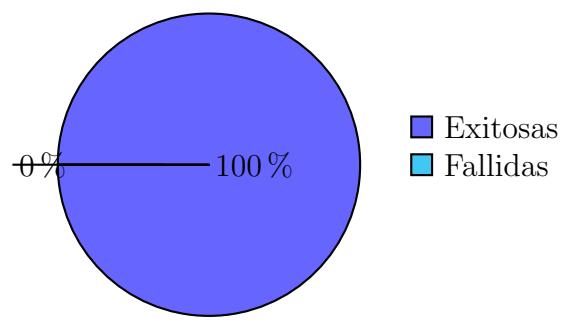


Figura 4.13: Prueba con $K = 3$ y $\gamma = 0,1$ e inicialización de $Q(s,a)$

Capítulo 5

Conclusiones y Recomendaciones

La presente investigación ha nacido de la motivación por hacer que en Venezuela se incursoe en proyectos que involucren humanoides autónomos e inteligentes. Se ha inspirado especialmente en la categoría Robocup soccer de la competencia internacional Robocup. Desde 1997, fecha en la que inició la competencia, Venezuela nunca ha participado en categorías con humanoides, mientras que países latinoamericanos como México, Brasil y Colombia sí han tenido avances en este campo. Si bien este proyecto no cumple con todas las reglas de la competencia se espera que éste pueda dar pie a continuar investigaciones dentro de Venezuela.

Los componentes utilizados en este proyecto son relativamente económicos comparados con otros en el mercado. La integración del kit Bioloid Premium con la Arbotix y la Raspberry Pi, ha hecho posible construir un humanoide inteligente sin tener que invertir exorbitantes cantidades de dinero. Una de las contribuciones más importantes es la coordinación y paralelismo exitoso entre todos los componentes utilizados.

De los resultados obtenidos se evidencia que el aprendizaje por reforzamiento si bien genera mayor trabajo, es recomendable su utilización no solo por el hecho que logró buenos resultados en relativamente pocas pruebas (un 100 % de efectividad), además es adaptable si las condiciones físicas del robot llegaran a cambiar. Pues solo haría falta volverlo a entrenar

sin tener que cambiar el código del programa.

Las recomendaciones directas con este proyecto serían perfeccionar el aprendizaje para obtener una mayor eficiencia en el número acciones realizadas, aumentando el número de regiones y acciones disponibles y aumentando el número de ejecuciones para el entrenamiento. De esta manera se podría reducir el tiempo que tarda en llegar a la pelota. Además agregarle otro grado de libertad a las caderas de Junny para que obtenga mejor movilidad y pueda rodear la pelota mientras busca el arco sin alejarla.

Otras mejoras que se pueden incorporar al proyecto podrían ser: añadir aprendizaje de máquinas para hacer que el robot pueda predecir la posición de una pelota en movimiento para que pueda patearla en el momento indicado. Aprendizaje para saber qué tipo de patada es mejor según la posición del arco, y aprendizaje para la búsqueda del arco.

Bibliografía

- [112a] RASPBERRY PI FOUNDATION UK REGISTERED CHARITY 1129409. Camera module setup.
- [112b] RASPBERRY PI FOUNDATION UK REGISTERED CHARITY 1129409. What is a raspberry pi?
- [201a] RoboCup 2014. Robocup 2014.
- [201b] RoboCup 2014. Robocup soccer.
- [AiR00] *Introduction to AI Robotics*. 2000.
- [Ard14] Arduino. Arduino, 2014.
- [Asy] Artisan's Asylum. Introduction to raspberry pi.
- [Boo08] *Learning OpenCV*. 2008.
- [Co11] Honda Motor Co. Honda unveils all-new asimo with significant advancements, 2011.
- [de14] Comunidad de eLinux.org. Rpi hardware, 2014.
- [Ele] SparkFun Electronics. Ftdi sparkfun.
- [Fer10] M Fergs. Pypose, 2010.

[Hob] HobbyKing. Turnigy tg9e 9g.

[INCa] Robotics INC. Gyro sensor.

[INCb] Robotis INC. Bioloid.

[LLC] Trossen Robotics LLC. arbotix robocontrollers.

[Mit97] Tom M. Mitchell. Machine learning. 1997.

[pet95] *Artificial Intelligence A Modern Approach*. 1995.

[pet09] *Artificial Intelligence A Modern Approach*. 2009.

[Pre14] Oxford University Press. Robotics, 2014.

[Rau13] Pierre Raufast. Opencv and pi camera board !, 2013.

[rd] ros drivers. Rosserial.

[Rob] Robotis. Lipo 11.1v battery set lbs-10.

[Ros] Ros. About ros.

[SSA⁺] Mostafa E. Salehi1, Reza Safdari, Erfan Abedi, Bahareh Foroughi, Amir Salimi, Emad Farokhi, Meisam Teimouri, and Roham Shakiba. Mrl team description paper for humanoid kidsize league of robocup 2014.

[Tea] OpenCV Developers Team. About.

[Val] Emil Valkov. Raspberry pi camera with opencv.

[Val13] Emil Valkov. Raspberry pi camera with opencv, 2013.

[Van] Laboratorios Vanadium. Arbotix robocontroller.

[WL] Williams and Wang LLC. Dynamixel ax/mx 6 port extension hub.

[YML] Seung-Joon Yi, Stephen McGill, and Daniel D. Lee. Improved online kick generation method for humanoid soccer robots.

Apéndice A

Consideraciones Especiales:

Obstáculos y Soluciones

Durante el desarrollo del proyecto se presentaron algunos obstáculos que lograron ser resueltos. A continuación se describe la solución de algunos de esos obstáculos.

- Errores de la tarjeta Arbotix para cargar programas:

Problema: El IDE de Arduino 1.0.1 no funciona para quemar programar en la tarjeta Arbotix

Solución: Utilizar la versión 1.0.5 del IDE de Arduino que compila los programas sin problema.

Problema: El gestor de arranque de la tarjeta Arbotix no estaba guardado.

Solución: Se instaló en la Arbotix el gestor de arranque Sanguino a través del dispositivo programador para AVR llamado “ISP programmer” que se muestra en la figura A.1.

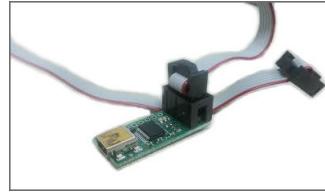


Figura A.1: Programador para AVR.

- Quema de Motores Dynamixel:

Problema: Debido al uso prolongado pero necesario los motores Dynamixel AX-12 se dañaban, ya sea el motor DC o el chip interno.

Solución: Fue controlar el torque y la temperatura máxima a la que pueden llegar los motores. En caso de llegar a estas cotas máximas los motores se apagan automáticamente. Las cotas máximas han sido de 30° centígrados para la temperatura y 800 kgf-cm para el torque. Para lograr esto se ha tenido que modificar la librería Ax12 agregando procedimientos que permitieran establecer la temperatura y el torque máximo.

En el archivo ax12.h se agregaron las siguientes definiciones de funciones:

```

1
2 #define SetTemperature( id ,temp )
3             ( ax12SetRegister( id ,AX_LIMIT_TEMPERATURE, temp ) )
4 #define SetAlarm( id )
5             ( ax12SetRegister( id ,AX_ALARM_SHUTDOWN, 0x04 ) )
6 #define SetTorqueL( id , tor )
7             ( ax12SetRegister2( id ,AX_MAX_TORQUE_L, tor ) )

```

El archivo ax12.h viene con el paquete de la página oficial para el código de Arbotix. Como se indica en las instrucciones, este archivo se debe ubicar en la carpeta sketchbook de Arduino. Luego desde el IDE de Arduino llamamos a las funciones definidas con los

valores deseados. De esta manera se ha solucionado el problema de la quema de motores.

Recomendaciones

Para la instalación del sistema operativo Raspbian en la tarjeta Raspberry Pi se recomienda tener en cuenta que algunas tarjetas SD no funcionan adecuadamente. Si al prender la mini computadora solo se prende el led rojo, como ha ocurrido en este proyecto, se debe verificar que la tarjeta SD esté haciendo buen contacto con el puerto en que se conecta. Si se verifica esto último y aún así no prende, es probable que se deba intentar con otra tarjeta SD. Al inicio de este proyecto se ha usado una tarjeta mini-SD de 32GB clase 4, como no ha funcionado se ha reemplazado con una tarjeta SD de 4GB clase 4. Esta última ha funcionado, sin embargo se ha quedado sin capacidad de almacenamiento al instalar OpenCV, por lo que se ha reemplazado nuevamente por una tarjeta SD de 16GB clase 4. Esta ha sido suficiente para instalar todo lo necesario con holgura.

Como no se contaba con un monitor con entrada HDMI o VGA se debió buscar una solución alterna para observar la interfaz gráfica de Raspbian de la Raspberry Pi. Se utilizó el programa TightVNC para la visualización y control de la interfaz de Raspbian desde un computador remoto. Ha sido necesario poder observar lo que el robot percibe para llevar un control y una supervisión de su comportamiento.

Por último, sería conveniente advertir que para la instalación de ROS en la Raspberry Pi, la versión que se debe obtener es la más reciente, en este proyecto ha funcionado la versión 0.5.5 que se encuentra en el repositorio rosserial de ros-drivers en git [rd], de lo contrario podrían ocurrir problemas de sincronización en la comunicación de las tarjetas.