



**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

RoboCup [201a] es una competencia de fútbol, iniciada en 1997, donde contribuyen las áreas de robótica, investigación e inteligencia artificial. Entre sus categorías se encuentra RoboCup Soccer [201b], la cual consiste en la participación de pequeños robots humanoides que se enfrentan a otro equipo para jugar fútbol. El objetivo de esta 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).

En este proyecto se describe la construcción de Junny, un robot humanoide autónomo e inteligente de tamaño pequeño (38 cm de alto), capaz de detectar la ubicación de una pelota y acercarse a ella para patearla. Sus metas se enmarcan parcialmente dentro de las reglas de la categoría RoboCup Soccer.

Junny ha sido construido con las piezas del kit Bioloid Premium [INCb] del fabricante ROBOTIS [INCc]. Del kit se ha excluido la tarjeta CM-510 para sustituirla por la tarjeta controladora Arbotix, que será la que controle los 16 motores Dynamixel Ax-12+ (para mover al robot) y 2 servomotores analógicos (para mover la cámara). Además se ha agregado un mini computador Raspberry Pi, con su cámara [112a], 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 detección de regiones por color, 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 tamaño 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].

Agradecimientos

AGRADECIMIENTOS

Índice general

Índice general	viii
Índice de figuras	x
1. Introducción	1
2. Marco teórico	5
2.1. Robótica	5
2.2. Robótica Inteligente (Agentes Inteligentes)	6
2.2.1. Paradigmas de robótica	7
2.2.1.1. Paradigma Jerárquico	7
2.2.1.2. Paradigma Reactivo	7
2.2.1.3. Paradigma Híbrido	8
2.3. Inteligencia Artificial	8
2.3.1. Aprendizaje de Máquinas	8
2.3.2. Aprendizaje por reforzamiento	9
2.3.3. Q-learning	9
2.4. Visión Artificial	10
2.4.1. Segmentación por regiones	11
2.4.2. Filtros	11
2.4.3. Transformaciones Morfológicas	12
2.4.3.1. Dilatación	12
2.4.3.2. Erosión	12
3. Construcción de un Robot Humanoide	14
3.1. Diseño y Construcción	14

3.1.1.	Componentes de hardware	14
3.1.2.	Construcción	19
3.2.	Detección de la pelota	24
3.2.1.	Herramientas software para la detección	24
3.2.2.	Obtención de la imagen	25
3.2.3.	Procesamiento de la imagen	25
3.3.	Búsqueda y Pateo	26
3.3.1.	Herramientas software para la búsqueda de la pelota	27
3.3.2.	Movimiento del esqueleto	28
3.3.3.	Movimiento de la cámara	29
3.3.4.	Representación del mundo	30
3.3.5.	Comunicación Arbotix - Raspberry (ROS)	31
3.4.	Aprendizaje	32
3.4.1.	Modelo del problema	33
3.4.1.1.	Estados	33
3.4.1.2.	Acciones	33
3.4.1.3.	Recompensas	34
3.4.2.	Eleción de la acción	35
3.5.	Detección de caídas	36
4.	Experimentos y Resultados	39
4.1.	Experimentos de Movimientos	39
4.2.	Experimentos con Comportamientos Integrados	41
4.3.	Experimentos con Aprendizaje	43
5.	Conclusiones y Recomendaciones	45
Bibliografía		46
A. Consideraciones Especiales: Obstáculos y Soluciones		49

Índice de figuras

2.1. Dilatación	13
2.2. Erosión	13
3.1. Bioloid Premium Kit	15
3.2. Motores Dynamixel conectados en serie	15
3.3. Sensor Gyro	16
3.4. Chip FTDI conectado a la tarjeta Arbotix	16
3.5. Extensor de puertos Bioloid	17
3.6. Micro Servomotores analógicos TG9e	17
3.7. Tarjeta Raspberry Pi con descripción de los puertos	18
3.8. Cámara Raspberry Pi	18
3.9. Batería Lipo	19
3.10. Vista frontal del robot, tipo B, tomada del manual. Se puede apreciar la identificación ‘ID’ de cada motor Dynamixel Ax-12+. Nota: los motores 9 y 10 no se utilizan.	20
3.11. Vista trasera del robot con la tarjeta CM-510, tomada del manual del kit Bioloid.	20
3.12. Vista trasera del robot con la Arbotix	21
3.13. Cámara Raspberry Pi conectada al puerto CSI de la tarjeta	21
3.14. Vista delantera del robot con la cámara y servomotores instalados	22
3.15. Tarjeta controladora Arbotix y componentes conectados	23
3.16. 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.	23
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	34
3.20. Distancias relativas de la pelota establecidas para otorgar la recompensa.	35
3.21. Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.	36
3.22. Dirección en la que la velocidad angular del eje X crece con respecto al robot.	37
3.23. Gráfica de lecturas de velocidad angular en el eje X.	38
4.1. Experimento 2: 300 ejecuciones de acciones combinadas	40
4.2. Experimento 3: Pruebas de movimientos integrados CASO I	41
4.3. Experimento 3: Pruebas de movimientos integrados CASO II	41
4.4. Experimento 3: Pruebas de movimientos integrados CASO III	42
4.5. Experimento 3: Pruebas de movimientos integrados CASOIV	42
A.1. Programador para AVR.	50

Capítulo 1

Introducción

RobotCup [201a] es una competencia de fútbol iniciada desde 1997 donde contribuyen las áreas de robótica, investigación e inteligencia artificial. Entre sus categorías se encuentra RobotCup Soccer [201b], la cual consiste en la participación de pequeños robots humanoides que se enfrentan a otro equipo para jugar fútbol. El objetivo de esta 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). 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.

El desarrollo de esta investigación presenta un robot humanoide (Junny) autónomo e inteligente de tamaño pequeño (38 cm de altura) cuyos objetivos estan basados en las reglas de la competencia RobotCup. En investigaciones con este mismo enfoque se puede encontrar el trabajo de Sven Behnke cuyo título es “See, walk, and kick: Humanoid robots start to play soccer” donde se describe la construcción del equipo de robots que participaron en la RobotCupSoccer en el año 2006. El artículo cubre el diseño mecánico y electrónico, además el

software utilizado para la percepción, control de comportamiento, comunicación y simulación de los robots [BSS⁺].

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 giróscopio y sensores de aceleración, y para la visión una cámara conectada por usb al CPU principal [SSA⁺].

En el desarrollo de habilidades más específicas con respecto a la competencia RobotCup Soccer, en el artículo de investigación de Seung-Joon Yi, Stephen McGill y Daniel D. Lee [YML], se refieren a dos posibles estrategias para el pateo de la pelota donde los factores fundamentales para un buen desempeño son la fuerza y la rapidez con que se patea, los investigadores ponen en práctica dos estrategias de pateo en distintas circunstancias del juego basado en la cinemática y dinámica de equilibrar el cuerpo al momento de realizar el pateo.

Se planteo como objetivo de este proyecto 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:

1. Diseño y construcción de un humanoide con piezas del kit de robótica Bioloid Premium, sustituyendo su tarjeta controladora CM-510 por la tarjeta de software libre ArbotiX para controlar los motores Dynamixel y otros sensores.
 - a) Instalación y configuración de la tarjeta ArbotiX.

- b) Instalación y configuración de la tarjeta Raspberry Pi.
 - c) Instalación y configuración de la cámara Raspberry Pi.
 - d) Instalación de servomotores para el movimiento de la cámara
 - e) Instalación del giroscopio Gyro.
2. Detección de la pelota
- a) Captura de imagen con la cámara Raspberry Pi a través de la librería raspicam cv.
 - b) Procesamiento de la imagen para extraer información de la posición de la pelota con las librerías de OpenCV.
3. Búsqueda de la pelota y pateo de la misma.
- a) Creación de las poses necesarias para caminar, girar, levantarse y patear usando el software pypose.
 - b) Programación de transiciones de movimientos.
 - c) Control de servomotores para el movimiento de la cámara.
 - d) Establecer mecanismo de comunicación entre la tarjetas ArbotiX y Raspberry Pi.
 - e) Programación de algoritmo de planificación de acciones que lleve al humanoide a acercarse a la pelota.
 - f) Detección de movimientos angulares bruscos que sugieran una caída, a través de la lectura del giroscopio
 - g) Identificación del momento en que la pelota se encuentre en una zona adecuada para patear.

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 capítulo 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 el presente 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:** Son agentes físicos que ejecutan tareas para manipular el mundo físico. Para ello deben estar equipados con actuadores y sensores [pet09]. La apariencia no es una

característica útil para la definición de un robot [AiR00], por lo tanto puede ser de diferentes formas, ya sea con ruedas, con piernas o ninguna de ellas. Una de las formas que puede adoptar un robot es la de humano, de hecho en la cultura popular el término “robot” generalmente connota una apariencia humana [AiR00]. Según el diccionario de la Universidad de Oxford, el término humanoide se refiere a tener una apariencia o característica parecida al de un ser humano [Pre14], por lo tanto a los robots con forma de humano se les denomina robots humanoides.

- **Sensores:** Son los dispositivos encargados de percibir el ambiente que rodea al robot. Según Murphy R.R estos miden algún atributo del mundo. Un sensor recibe energía del entorno (sonido, luz, presión, temperatura, etc) y transmite una señal a una pantalla o computador ya sea de forma análoga o digital [AiR00]. Algunos sensores son: cámaras, giroscopios, sensores de proximidad, entre otros.
- **Actuador:** Es aquella parte del robot que convierte comandos de software en movimientos físicos. Por ejemplo ruedas, piernas, pinzas, entre otros [pet95].
- **Servomotor:** Es un motor eléctrico, considerado como actuador, que permite ser controlado tanto en velocidad como en posición [AiR00].
- **Giróscopio:** Es un sensor utilizado para medir y mantener la orientación, se mide a través del momento angular [Con14].

2.2. Robótica Inteligente (Agentes Inteligentes)

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.

2.2.1. 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.

2.2.1.1. Paradigma Jerárquico

Este paradigma es secuencial y ordenado. Primero el robot percibe el mundo y construye un mapa global. En base al mapa ya percibido y con “los ojos cerrados”, el robot planifica todas tareas necesarias para lograr la meta. Luego ejecuta la secuencia de actividades según la planificación realizada. Una vez culminada la secuencia se repite el ciclo percibiendo el mundo, planificando y actuando [AiR00].

2.2.1.2. 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,

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].

2.2.1.3. 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 que 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].

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 que está relacionada con la pregunta de cómo construir programas de computadora que automáticamente mejoren

con la experiencia. 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 [Mit97].

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. Las recompensas se pueden dar en cada estado o una sola vez al llegar al estado final.

El objetivo del agente es aprender de las recompensas para escoger la secuencia de acciones que produzca la mayor recompensa acumulada. [Mit97]

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, dado 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 [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 o castigo 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 que 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 el 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 que ayudan en esta tarea. 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.

2.4.1. Segmentacion 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 pixeles en una imagen que forman parte de una región y luego navegar al centro de la región. El primer paso es identificar todos los pixeles en la imagen que comparten en rango de valores con el color particular y agruparlos, aquellos pixeles que no comparten el color serán descartados.

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.

2.4.3. Transformaciones Morfológicas

Las transformaciones morfológicas básicas son llamadas dilatación y erosión, se utilizan en amplia variedad de contextos como la eliminación del ruido, aislamiento de elementos individuales y elementos de unión dispares en una imagen. [Boo08]

2.4.3.1. Dilatación

La dilatación es una convulsión (algo 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, tiene un solo punto de anclaje definido. Para mayor claridad el nucleo es esencialmente una matriz de tamaño fijo de coeficientes numéricos junto con un punto de anclaje en dicha matriz, que normalmente se encuentra en el centro . Muy a menudo, el núcleo es un pequeño cuadrado o disco sólido. El núcleo puede ser pensado como una plantilla o mascara, 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 anclaje 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].

2.4.3.2. Erosión

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

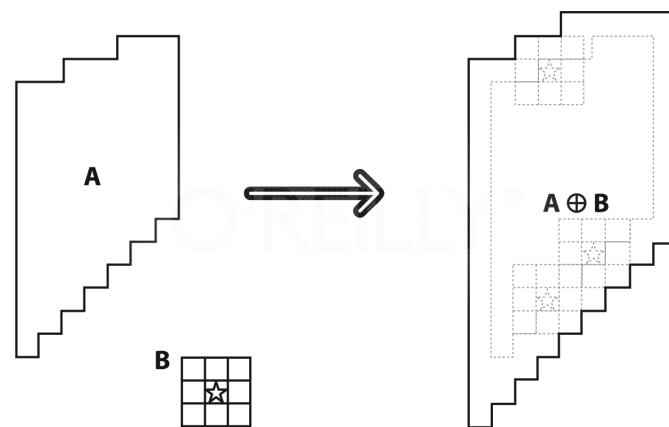


Figura 2.1: Dilatación

reemplaza el píxel de la imagen con un punto de anclaje de valor mínimo [Boo08]. Vease en la figura 2.2

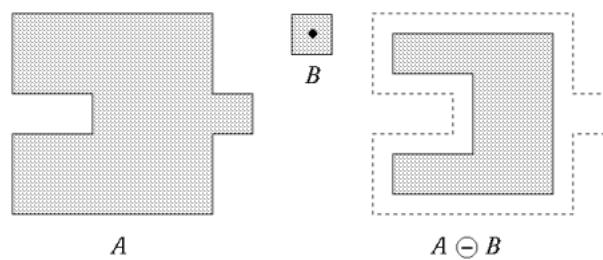


Figura 2.2: Erosión

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. También se describen las actividades realizadas para lograr que en caso de tener un desbalance y caer, sepa levantarse.

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 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. Componentes de hardware

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

- Bioloid Premium kit: Es un kit de robótica con piezas modulares 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].



Figura 3.1: Bioloid Premium Kit

- 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), todo en un solo paquete [INC06].

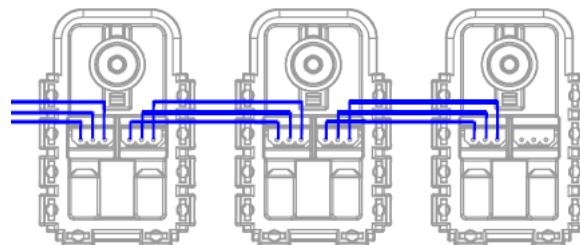


Figura 3.2: Motores Dynamixel conectados en serie

- Gyro: Es un giroscopio de la marca Robotis que mide la velocidad angular. Se encuentra diseñado para mantener el balance del robot y ser usado para otras aplicaciones de movimiento [INCa]. En figura 3.3 se puede observar su estructura.



Figura 3.3: Sensor Gyro

- Arbotix: El controlador ArbotiX es una solución de control avanzado para manejar algunos tipos de servos Dynamixel y robots basados en Bioloid. Incorpora un potente microcontrolador AVR, radio inalámbrica XBEE, conductores de motor dual, y cebaderas de estilo servo de 3 pines para entrada/salida digital y analógica [LLC].
- FTDI (Future Technology Devices International) : Es una tarjeta controladora (figura 3.4) que ofrece el servicio de conversión de datos de USB a UART. Permite la comunicación entre diferentes dispositivos [Ele].

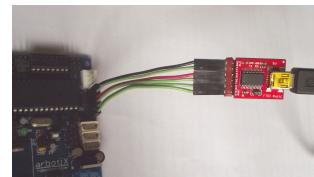


Figura 3.4: Chip FTDI conectado a la tarjeta Arbotix

- Extensor de puertos Bioloid : Permite aumentar el número de cadenas de servos conectados a la tarjeta (ver figura 3.5) [WL].
- Micro servo motor analógico TG9e: 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 3.5: Extensor de puertos Bioloid



Figura 3.6: Micro Servomotores analógicos TG9e

- Raspberry Pi: La Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito a la que se puede conectar un televisor y un teclado. Se trata de un pequeño ordenador capaz de ser utilizado en proyectos de electrónica y para muchas de las tareas que una PC de escritorio hace, como hojas de cálculo, procesadores de texto y juegos [112b]. Ver figura 3.7. La Raspberry Pi cuenta con un procesador gráfico que permite aligerar la carga del procesador central [de14].
- Cámara Raspberry Pi: Es un sensor encargado de 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 3.8)
- Batería de polímero de litio (Lipo): Es la fuente de poder usada para los motores y componentes electrónicos. La batería usada es de 11.1 voltios y 1 amperio. [Rob]

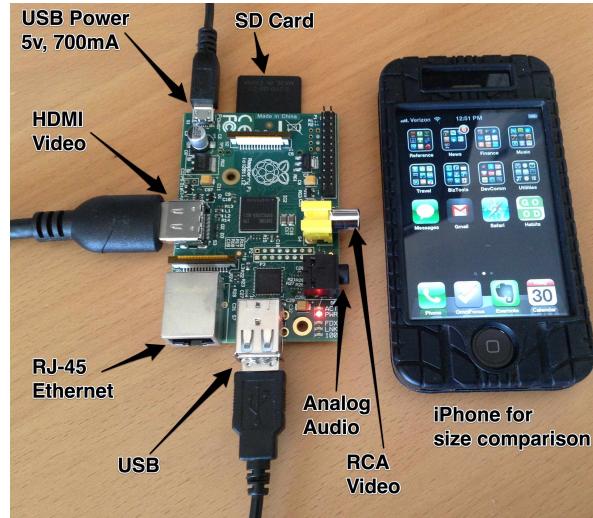


Figura 3.7: Tarjeta Raspberry Pi con descripción de los puertos

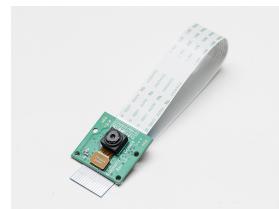


Figura 3.8: Cámara Raspberry Pi



Figura 3.9: Batería Lipo

- Circuito con regulador de 5v: Es un circuito diseñado y construido para este proyecto cuya finalidad es regular la entrada de la corriente. Por una de las salidas se expulsa 5v y por la otra se mantiene el mismo voltaje de entrada.

3.1.2. Construcción

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 varios modelos de humanoide, el utilizado en este proyecto es el tipo B, haciendo uso de 16 motores. En las figuras 3.10 y 3.11 se puede observar la estructura del robot que aparece en el manual del kit.

El kit bioloid incluye una tarjeta controladora CM-510 la cual 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.

En la figura 3.12 se puede observar la estructura del robot con la Arbotix incorporada. En la parte interna del tronco del robot se sitúa el sensor Gyro.

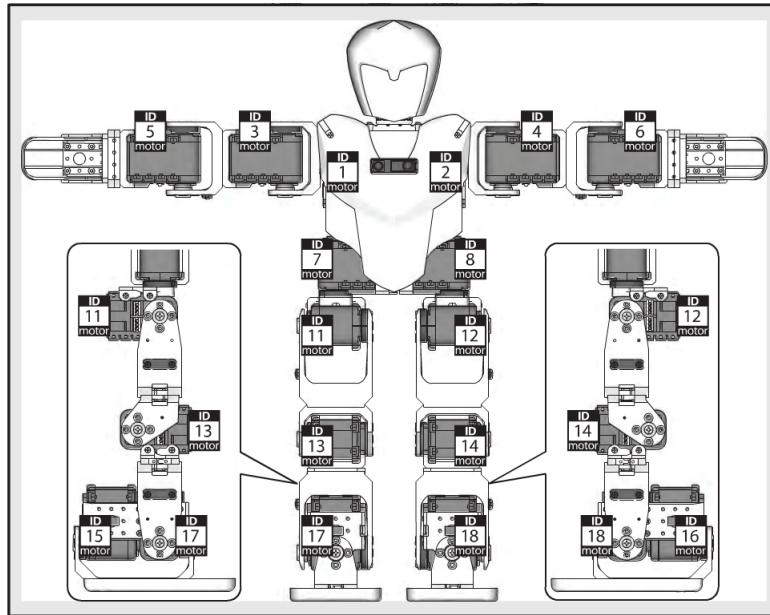


Figura 3.10: Vista frontal del robot, tipo B, tomada del manual. Se puede apreciar la identificación ‘ID’ de cada motor Dynamixel Ax-12+. Nota: los motores 9 y 10 no se utilizan.

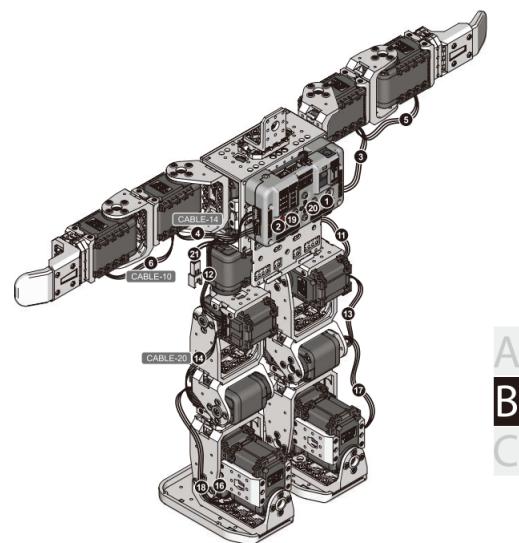


Figura 3.11: Vista trasera del robot con la tarjeta CM-510, tomada del manual del kit Bioloid.



Figura 3.12: Vista trasera del robot con la Arbotix

Para el movimiento de la cámara se ha incorporado dos micro servomotores, uno para el movimiento horizontal y otro para el vertical. La conexión de uno de estos motores se ilustra en la figura 3.15, 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 3.13). El resultado de estas tres piezas instaladas en el robot se puede apreciar en la figura 3.14.



Figura 3.13: Cámara Raspberry Pi conectada al puerto CSI de la tarjeta

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

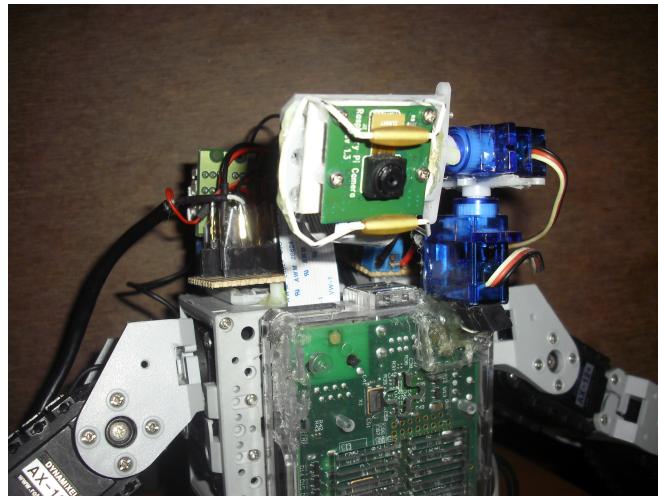


Figura 3.14: Vista delantera del robot con la cámara y servomotores instalados

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 en un puerto diferente. La forma en la que se ha conectado estos motores se ejemplifica en la figura 3.15.

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 3.15.

Como fuente de poder se utilizó una batería de polímero de litio de 11.1 V y 1 amp. Debido a que no todos los componentes poseen las mismas exigencias con respecto a voltaje y amperaje, se realizó un regulador (ver figura 3.16) 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.

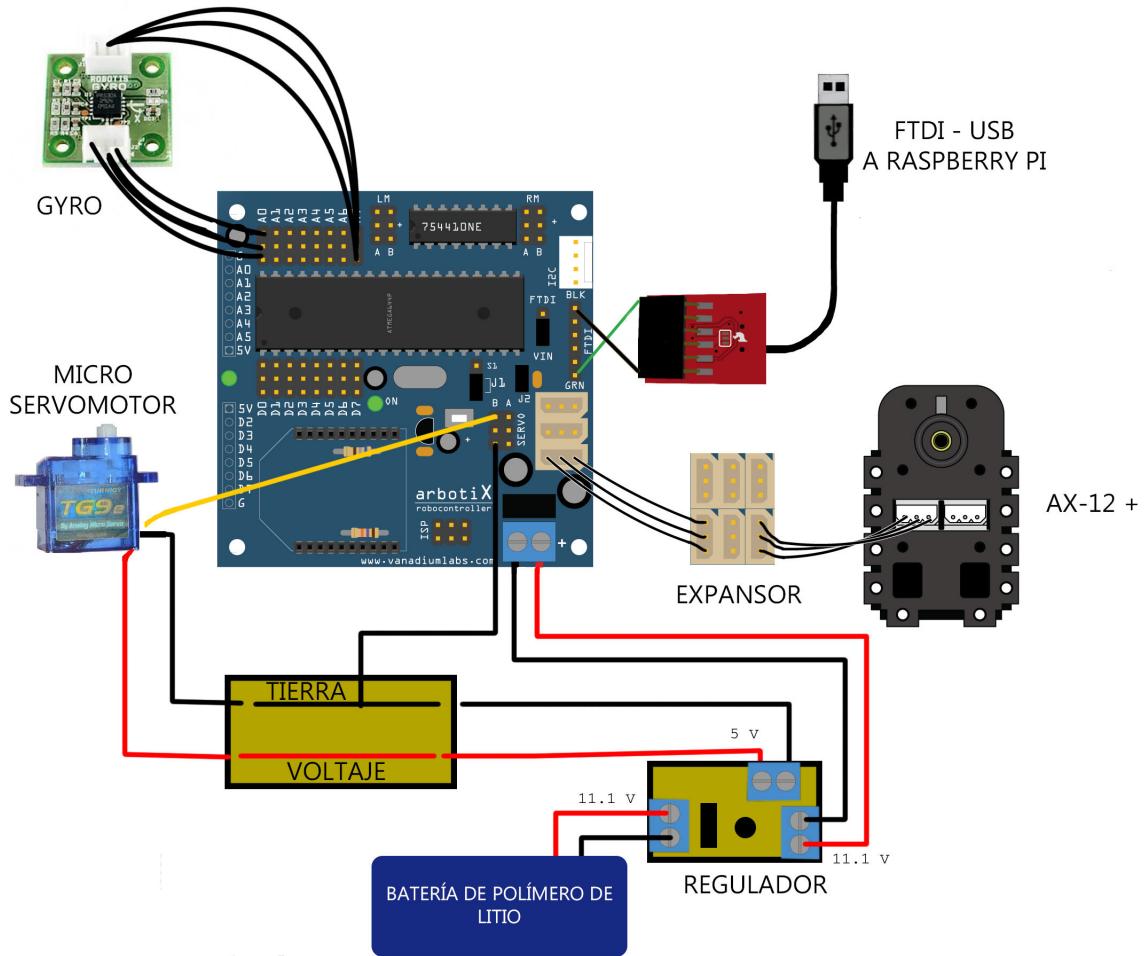


Figura 3.15: Tarjeta controladora Arbotix y componentes conectados

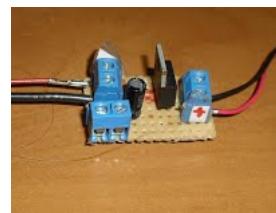


Figura 3.16: Circuito con entrada de 11.1 V. Una salida de 5 V para los micro servomotores analógicos y tarjeta Raspberry Pi. Otra salida de 11.1 V para alimentar la controladora Arbotix.

3.2. 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 potente 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 potente 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 como 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.

- **Raspicam_cv:** Es una librería que permite obtener imágenes de la cámara Raspberry Pi en una estructura de datos compatible con OpenCV [Val13]. Ha sido creada por Emil Valkov en base al trabajo de Pierre Raufast descrito en su blog [Rau13].
- **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. Registrada bajo la licencia BSD, de código abierto. [Tea]

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 Pierre Raufast, 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.

El código modificado por Raufast fué un gran avance, sin embargo no funcionaba como librería por lo que Emil Valkov facilitó la tarea de convertirlo en una, rascam_cv. Esta es la librería usada en la presente investigación para poder capturar las imágenes de la cámara.

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 ‘blobs’, o reconocimiento de regiones, esta técnica consiste en filtrar la imagen por color, por ello es importante que el de la pelota no se repita en el ambiente y así poder obtener su posición dentro de la imagen.

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.

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 con buenos resultados en la mayoría de los casos.

3.3. Búsqueda y Pateo

Para poder buscar y patear la pelota, además de tener la capacidad para detectarla (como se explicó en el capítulo 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, con excepción de la estrategia para la toma de acciones, que se explicará en la sección 3.4.

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 límitado 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 que acción tomar en cada situación. La estrategia, basada en el paradigma híbrido, se explica en la siguiente sección (3.4).

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: Software especializado en el control de los servomotores Dynamixel Ax-12. 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].
- IDE Arduino: 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].

3.3.2. Movimiento del esqueleto

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 referire 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 2.5 cm hacia adelante
- Caminar 4.8 cm hacia adelante
- Caminar 9.9 cm hacia adelante
- Girar 3 cm a la izquierda
- Girar 6 cm a la izquierda
- Girar 3 cm a la derecha
- Girar 6 cm a la derecha
- Levantarse desde la posición boca abajo
- Levantarse desde la posición boca arriba
- Patear con la pierna derecha
- Patear con la pierna 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.

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 solo puede ser usada para los motores conectados en los puertos Hobby A y B (pines 12 y 13) (ver la figura 3.15). 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.

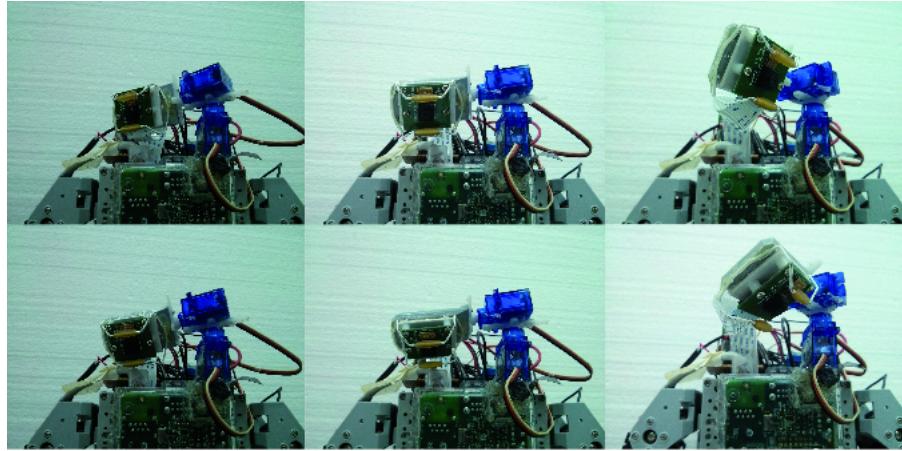


Figura 3.17: Posiciones de la cámara

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 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 de 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.

Las acciones específicas a tomar según la región en que se encuentre la pelota se seleccionan por medio de lo aprendido en el entrenamiento realizado con apredizaje por reforzamiento. Los detalles de este aprendizaje se describen en la sección 3.4.

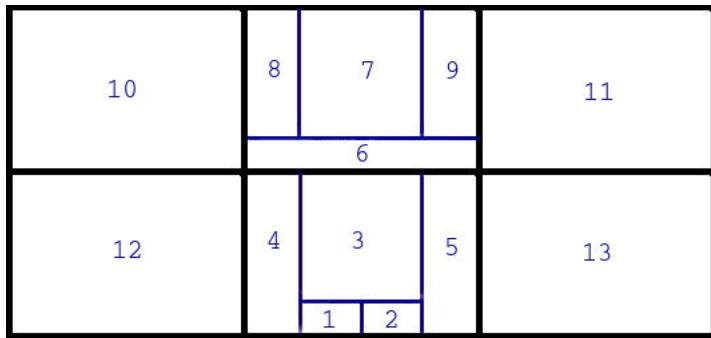


Figura 3.18: Campo de visión del robot con el número de cada región. Cada cuadro blanco demarcado con líneas negras representa la imagen captada desde una posición fija de la cámara. La región de pateo de la pelota se encuentra en las regiones 1 y 2

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.

3.4. 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 sección 2.3.3) se utilizó el modelo de aprendizaje Q-learning, el cuál 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 (como se muestra en la figura 3.19). Puede suceder, también, que la pelota no se encuentre en ninguna de ellas. Por lo tanto se generan 14 estados, 13 de ellos se corresponden con la detección de la pelota en cada una de las regiones definidas en la sección 3.3.4 y el estado número 14 representa la ocasión en la que no se logra ubicar la pelota en ninguna de las regiones.

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. Estas son:

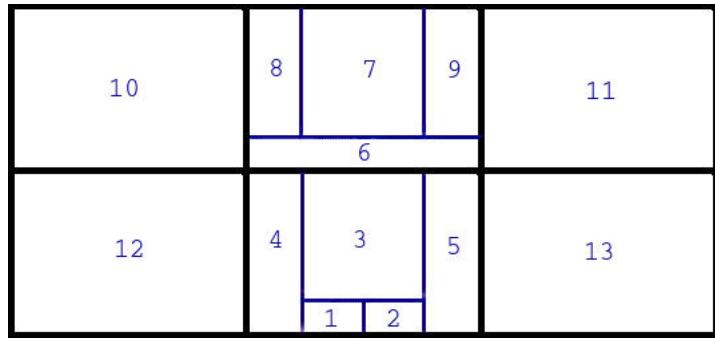


Figura 3.19: Estados definidos según la región en la que se detecta la pelota

- Caminar 2.5 cm hacia adelante
- Caminar 4.8 cm hacia adelante
- Caminar 9.9 cm hacia adelante
- Girar 3 cm a la izquierda
- Girar 6 cm a la izquierda
- Girar 3 cm a la derecha
- Girar 6 cm a la derecha

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 $r \in \{1, 2, 3.., 14\}$ 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 10 y 11). Adicionalmente se define la distancia de la región 14 como $d(14) = 10$.

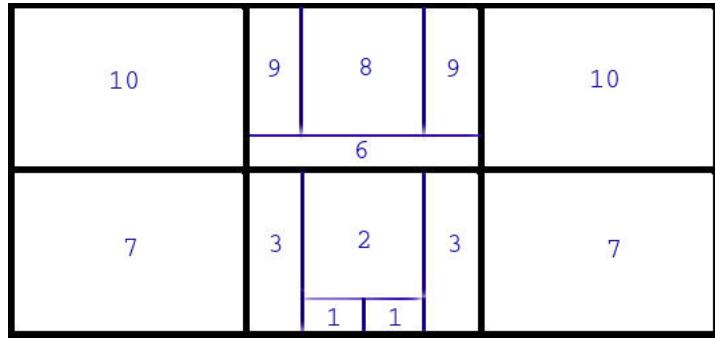


Figura 3.20: Distancias relativas de la pelota establecidas para otorgar la recompensa.

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.

3.5. Detección de caídas

Para poder detectar una posible caída del robot se cuenta con el giroscopio Gyro, 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.

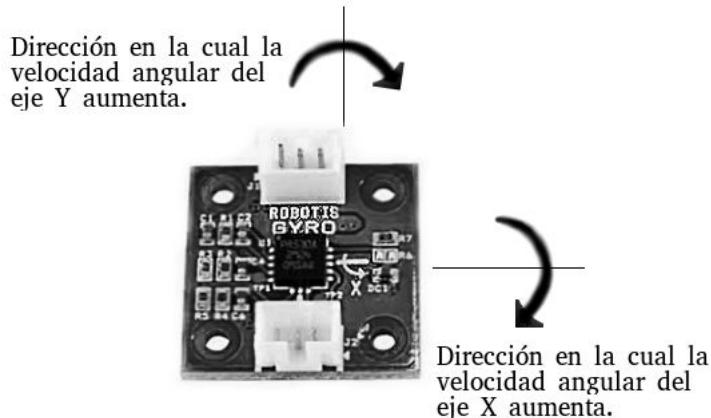


Figura 3.21: Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.

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, en 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 $0^{\circ}/s$.

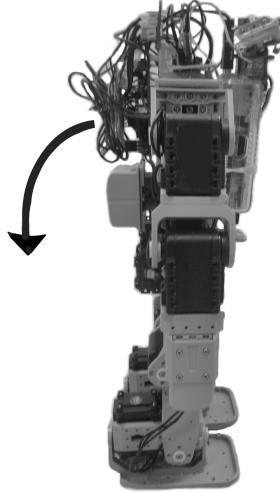


Figura 3.22: Dirección en la que la velocidad angular del eje X crece con respecto al robot.

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.

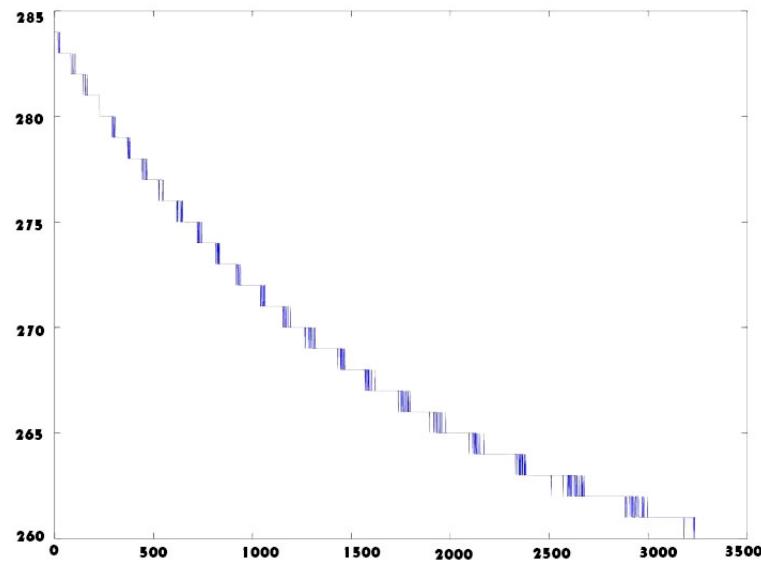


Figura 3.23: Gráfica de lecturas de velocidad angular en el eje X.

Capítulo 4

Experimentos y Resultados

HACER INTRO A RESULTADOS

4.1. Experimentos de Movimientos

Se realizaron varios experimentos para comprobar la movilidad del robot. Para esto se define lo que representa una unidad de movimiento. Cada una de las siguientes acciones constituyen una unidad: Caminar hacia adelante, girar hacia la derecha, girar hacia la izquierda, patada con la pierna derecha, patada con la pierna derecha, levantarse en la posición supino, levantarse en posición prono.

El primer experimento 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 el segundo experimento 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. 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.

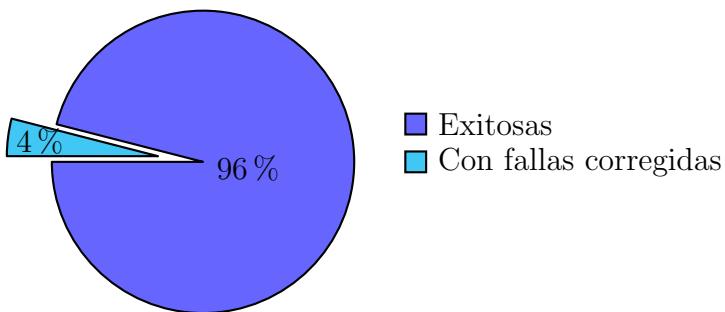


Figura 4.1: Experimento 2: 300 ejecuciones de acciones combinadas

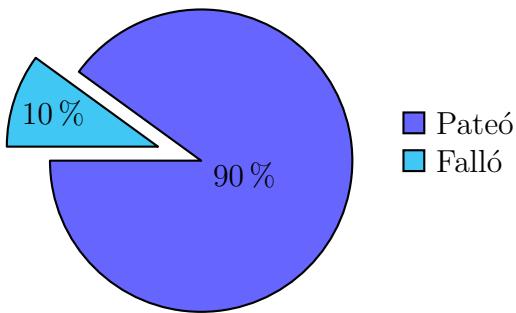


Figura 4.2: Experimento 3: Pruebas de movimientos integrados CASO I

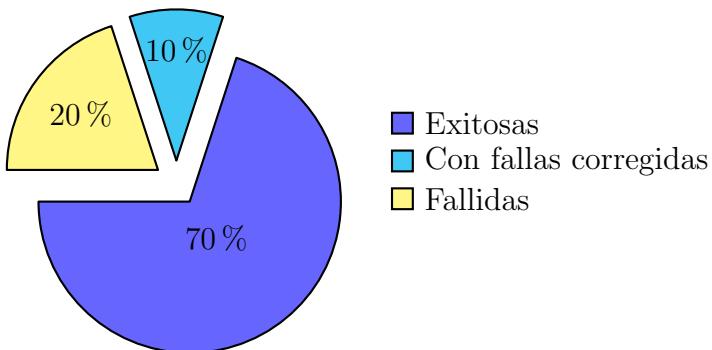


Figura 4.3: Experimento 3: Pruebas de movimientos integrados CASO II

4.2. Experimentos con Comportamientos Integrados

El tercer experimento 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).

El primer caso (CasoI) consistió en la colocación de la pelota en la zona de pateo (las regiones 15 y 16 de la figura 5). El número de pruebas en este primer caso fue de 10, cuya duración aproximada de cada prueba fue de 30 segundos. En la figura 6 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 7.

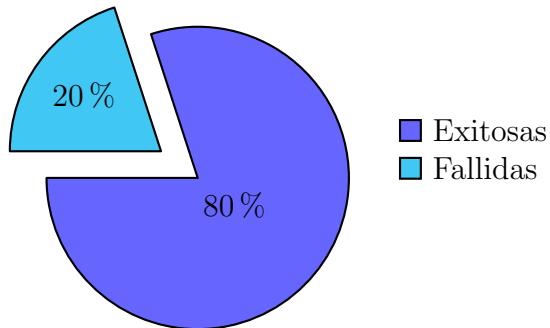


Figura 4.4: Experimento 3: Pruebas de movimientos integrados CASO III

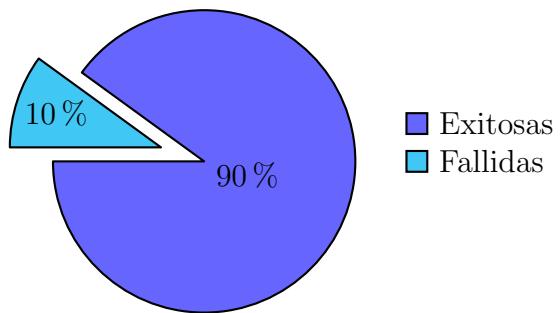


Figura 4.5: Experimento 3: Pruebas de movimientos integrados CASOIV

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 8.

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 9.

Con un total de 40 pruebas de comportamiento integrado, los resultados obtenidos han sido satisfactorios con un porcentaje de logro del 82 % más 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.

4.3. Experimentos con Aprendizaje

Finalmente los resultados de la aplicación del aprendizaje Q, para la búsqueda de la pelota se presentan a continuación. Se utilizaron las siguientes combinaciones de tasa de descuento , exploración y explotación por medio del parámetro ajustable llamado K.

- K = 1 , Descuento = 0.1 (Aleatorio)
- K = 2 , Descuento = 0.1
- K = 2 , Descuento = 0.7
- K = 3 , Descuento = 0.1
- K = 3 , Descuento = 0.7
- K = 5 , Descuento = 0.1
- K = 5 , Descuento = 0.7

Cada una de estas combinaciones se entrena con 20 pruebas completas (comportamiento integrado), una vez entrenadas se realizaron las siguientes pruebas para medir su desempeño y establecer la mejor tasa de aprendizaje por medio del porcentaje de pates positivos y fallas. Se realizaron 10 pruebas con el resultado del aprendizaje para cada una de las combinaciones establecidas, eso nos da un total de 210 pruebas realizadas.

Los resultados de los entrenamientos arrojaron que la mejor combinación fue K = 3 y Descuento = 0.1 que obtuvo un desempeño favorable del 90 % de las pruebas realizadas.

Para esta mejor combinación se inicializó la matriz de estados x acciones donde solo se daban recompensas negativas a las acciones que por ningún motivo debía tomar en dicho estado. Con ello y el aumento de la cantidad de pruebas de entrenamiento a 30 se pretendió mejoras el desempeño general del robot.

Se realizaron pruebas de desempeño con la cantidad de 15 pruebas donde se obtuvo un 100 % de pruebas positivas

FALTTTAAAAN - Explicar mejor que significa K y descuento ? - FALTA describir mejor como es una prueba - No se si hacer graficos - Explicar mejor como fue la inicializacion - analisis en profundidad de tiempos y acciones

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.

Las mejoras que se pueden incorporar al proyecto podrían ser: la inclusión de aprendizaje por reforzamiento para patear de forma exitosa, incluir que la patada sea en dirección al arco e incluso añadir aprendizaje por reforzamiento para hacer que el robot pueda predecir la posición de una pelota en movimiento para que pueda patearla en el momento indicado.

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.
- [BSS⁺] Sven Behnke, Michael Schreiber, Jörg Stückler, Reimund Renner, and Hauke Strasdat. See, walk, and kick:humanoid robots start to play soccer.
- [Co11] Honda Motor Co. Honda unveils all-new asimo with significant advancements, 2011.
- [Con14] Wolfram Demonstrations Project Contributors. Gyrocospe, 2014.

- [de14] Comunidad de elLinux.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.
- [INCc] Robtis INC. Robotis.
- [INC06] Robotics INC. *Dynamixel AX-12*, 2006.
- [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.
- [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.

[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: Con el gestor de arranque de 'Sanguino', se cargó a la Arbotix 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 por motor DC o en 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 deg 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) ( ax12SetRegister (id ,AX\_LIMIT\
    _TEMPERATURE, temp) )
3 #define SetAlarm (id ) ( ax12SetRegister (id ,AX\_ALARM\_SHUTDOWN, 0x04) )
4 #define SetTorqueL (id , tor) ( ax12SetRegister2 (id ,AX\_MAX\_TORQUE\_L , tor)
    )

```

El archivo ax12.h viene con el paquete de la pagina 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, como no ha funcionado se ha reemplazado con una tarjeta SD de 4GB. 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. 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, de lo contrario podrían ocurrir problemas de sincronización en la comunicación de las tarjetas.