



**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 con dirección al arco. Los objetivos de Junny están inspirados en la liga RoboCup Soccer de la competencia RoboCup.

Junny ha sido ensamblado con las piezas del kit Bioloid Premium del fabricante Robotis. Del kit se ha excluido la tarjeta CM-510 para sustituirla por la tarjeta controladora Arbotix, la cual controla los 16 motores que permiten el movimiento de las extremidades del robot. Se ha incluido un mini computador Raspberry Pi, con su cámara, de esta forma el robot ha adquirido la posibilidad detectar la posición de la pelota y el arco de forma autónoma. Se añadieron dos micro servomotores analógicos para ejecutar el movimiento de la cámara, estos son controlados por la tarjeta Arbotix.

En la Raspberry Pi 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 procesar la imagen, con la finalidad de detectar la pelota y arco, se ha usado la segmentación por regiones, con ayuda de las librerías OpenCv.

La Arbotix, además de controlar los motores para ejecutar los movimientos deseados, se encarga de monitorizar la velocidad angular del robot, para ello usa el sensor Gyro de Robotics. Con esta información Junny puede deducir 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 (Robot Operating System).

Finalmente se obtuvieron los siguientes resultados, un robot autónomo e inteligente que es capaz de reconocer la pelota, desplazarse hasta ella y patearla con dirección al arco. Con la aplicación de aprendizaje por reforzamiento, para acercarse a la pelota, se obtuvo un 100 % de aciertos, con una tasa de eficiencia de 0,72 que es el número de acciones esperadas entre las acciones realizadas. De los experimentos completos con la orientación al arco se obtuvo un 53 % de orientaciones correctas con resultado de gol.

Palabras claves: Robótica, Aprendizaje 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
1. Introducción	1
2. Marco teórico	5
2.1. Robótica	5
2.2. Robótica Inteligente	7
2.2.1. Paradigma Jerárquico	7
2.2.2. Paradigma Reactivo	8
2.2.3. Paradigma Híbrido	8
2.3. Inteligencia Artificial	9
2.3.1. Aprendizaje de Máquinas	9
2.3.2. Aprendizaje por reforzamiento	10
2.3.3. Q- learning	10
2.4. Visión Artificial	11
2.4.1. Segmentación por regiones	12
2.4.2. Filtros	12
2.4.2.1. Dilatación	13
2.4.2.2. Erosión	13
3. Construcción de un Robot Humanoide	15
3.1. Partes del robot	15
3.2. Ensamblaje	20
3.3. Detección de la pelota	24

3.3.1.	Herramientas software para la detección	24
3.3.2.	Obtención de la imagen	25
3.3.3.	Procesamiento de la imagen	26
3.4.	Búsqueda y Pateo	27
3.4.1.	Herramientas software para la búsqueda de la pelota	28
3.4.2.	Movimiento de extremidades	29
3.4.3.	Movimiento de la cámara	30
3.4.4.	Representación del mundo	32
3.4.5.	Comunicación Arbotix - Raspberry (ROS)	33
3.4.6.	Elección de la acción	33
3.5.	Aprendizaje	34
3.5.1.	Modelo del problema	36
3.5.1.1.	Estados	36
3.5.1.2.	Acciones	37
3.5.1.3.	Recompensas	38
3.5.2.	Elección de la acción	39
3.5.3.	Actualización de $Q(s,a)$	39
3.6.	Detección de caídas	40
3.7.	Orientación al arco	42
4.	Experimentos y Resultados	44
4.1.	Experimentos de Movimientos	45
4.2.	Experimentos con Comportamientos Integrados	47
4.3.	Experimentos con Aprendizaje	49
4.4.	Experimentos completos	52
5.	Conclusiones y Recomendaciones	54
Bibliografía		57
Glosario		61
A. Consideraciones Especiales: Obstáculos y Soluciones		65
B. Carta Aceptación		68

Índice de figuras

2.1.	Paradigma Jerárquico	8
2.2.	Paradigma Reactivo	8
2.3.	Paradigma Híbrido	9
2.4.	Dilatación: A es la imagen original, B es el elemento estructurante, La estrella es el punto de referencia. Se ve como aumenta la imagen en proporción al patrón aplicado	14
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	14
3.1.	Bioloid Premium Kit	16
3.2.	A la izquierda, 3 motores Dynamixel conectados en serie. A la derecha, batería Lipo	17
3.3.	A la izquierda, sensor Gyro. A la derecha, extensor de puertos Bioloid	18
3.4.	Tarjeta Raspberry Pi y su módulo de cámara a la derecha	19
3.5.	Chip FTDI conectado a la tarjeta Arbotix	19
3.6.	Micro Servomotores analógicos TG9e	19
3.7.	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.	20
3.8.	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 [14]	21
3.9.	Vista trasera del robot con la tarjeta CM-510, Imagen tomada sin autorización de [14].	21
3.10.	Vista posterior del robot, con la tarjeta Arbotix integrada (tarjeta azul) y la bateria Lipo	22

3.11. Parte delantera del robot incluye cámara, los dos servo motores (azules) y raspberry pi	22
3.12. Tarjeta controladora Arbotix y componentes conectados (Imagen referencial: el regulador se encuentra dispuesto en diferente orden)	23
3.13. (a) imagen original en BGR. (b) imagen original transformada a HSV. (c) imagen binaria a partir de la imagen HSV y el color de la pelota	27
3.14. Posiciones de la cámara	31
3.15. 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	33
3.16. Estados definidos según la región en la que se detecta la pelota	37
3.17. Distancias relativas de la pelota establecidas para otorgar la recompensa.	38
3.18. Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.	40
3.19. Dirección en la que la velocidad angular del eje X crece con respecto al robot.	41
3.20. Campo de visión del robot con el número de cada región para buscar el arco.	42
4.1. Resultados pruebas individuales de movimiento	46
4.2. Resultados de los experimentos con las acciones combinadas elegidas	47
4.3. Resultados de las pruebas de detección	47
4.4. Resultados de los casos I, II, III, IV descritos en la sección de 4.2	48
4.5. Resultados de los distintos parámetros con aprendizaje	50
4.6. Comparación de movimientos predeterminados y aprendizaje	52
4.7. Prueba con 15 ejecuciones, con aprendizaje y orientación al arco	53
A.1. Programador para AVR.	66
B.1. Carta de aceptación congreso Cuenca	69

Capítulo 1

Introducción

La inteligencia artificial (IA) y la robótica podrían llegar a dominar el mundo o quizás sólo harán todo por nosotros, el futuro de estas áreas es muy incierto, pero lo que sí es cierto es que los avances se están dando, los estudios se están haciendo y que cada día más personas en todo el mundo se familiarizan con IA y robótica.

Existen robots de distintas formas y tamaños, que realizan diversas tareas. Algunos famosos como NAO [3], Darwin [17] o ASIMO [7], tienen como denominador común que su forma es semejante a la de un humano, por ello son llamados humanoides. 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 de ello es el robot ASIMO, creado por la compañía Honda, cuyos últimos avances incluyen la predicción de trayectoria de objetos para poder esquivarlos. Existen estudios que se basan en la interacción entre humanos y humanoides. Por ejemplo, se han realizado experimentos para describir si los factores de contagio y sincronicidad de movimiento en grupos Humano-Humano se puede dar en un grupo Humano-Humanoide, de estos experimentos se ha concluido que la combinación de forma y movimiento son factores importantes para las competencias sociales de la interacción Humano-Humanoide (Erhan Oztop 2005) [22]. Existen investigaciones más específicas, como por ejemplo la investigación

del efecto de la fricción del sudor humano que compara la efectividad del sudor artificial en sensaciones táctiles de fricción para dedos de robots humanoides. Entre sus conclusiones se encuentra que efectivamente el sudor artificial realiza aumentos favorables en las señales táctiles al contacto de distintos materiales (Makoto Tomimoto 2014) [35].

Para los humanoides el equilibrio y balance al realizar una tarea son de vital importancia. Una de las investigaciones basadas en este tema dice que para realizar alguna tarea como pedalear o patinar existe una serie de movimientos que deben poseer un ritmo. Se plantea adaptar estas trayectorias rítmicas o periódicas a un comportamiento con la fuerza y habilidades motoras correctas en distintas condiciones. Su enfoque ha sido la utilización de *feedback* y *feed-forward* y sus experimentos, tanto simulados como con un robot real, han obtenido mejores resultados utilizando *feedback* y *feed-forward* simultáneamente. En particular, con la simulación se obtuvo un mejor desempeño que con los experimentos en la vida real (Andrej Gams 2014) [12].

RoboCup [1] 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 en 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 [2], la cual consiste en la participación de robots humanoides que se enfrentan a otro equipo para jugar 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”, ellos 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 USB al CPU principal [33].

En el desarrollo de esta investigación se presenta a Junny, un robot humanoide autónomo e inteligente de 38 cm de altura, cuyos objetivos están inspirados en la competencia RoboCup. Se planteó como objetivo principal construir un prototipo que sea capaz de detectar la cercanía de una pelota, acercarse a ella y patearla con dirección al arco, reincorporándose a la posición de pie en caso de perder el equilibrio y caer. Se ha tomado en cuenta un conjunto de objetivos específicos para guiar el desarrollo de este trabajo.

Un objetivo específico es el diseño y ensamblaje del robot humanoide con piezas del kit de robótica Bioloid Premium, sustituyendo su tarjeta controladora CM-510 [16] por la tarjeta de software libre Arbotix para controlar los motores Dynamixel y otros sensores. Esta tarea a su vez se subdivide en las siguientes asignaciones: Instalación y configuración de la tarjeta Arbotix, instalación y configuración de la tarjeta Raspberry Pi, instalación y configuración de la cámara Raspberry Pi, instalación de servomotores para el movimiento de la cámara, instalación del giroscopio.

Luego el robot deberá detectar la pelota, realizando la captura de la imagen, con la cámara Raspberry Pi y procesar la imagen para extraer información de la posición de la pelota con las librerías de OpenCV. Cuando Junny logre detectar la pelota tiene que buscarla y acercarse a ella, para completar esta tarea deberá también hacer los movimientos necesarios para caminar, girar, levantarse y patear usando el software pypose, controlar servomotores para el movimiento de la cámara, establecer mecanismo de comunicación entre la tarjetas Arbotix y Raspberry Pi, detectar caídas.

Para llegar a la pelota el robot debe hacerlo por medio de aprendizaje de máquinas específicamente por reforzamiento con ello se debe hacer la implementación de algoritmo de

planificación de acciones que lleve al humanoide a acercarse a la pelota con el aprendizaje y la utilización de Aprendizaje Q, su entrenamiento y realizar pruebas de desempeño. Finalmente cuando llegue a la pelota deberá ubicar el arco, posicionarse y patear orientado al arco para meter gol.

Esta investigación se divide en 5 capítulos. El capítulo 2 se refiere a una base teórica de conceptos e información necesaria 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. 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 el campo de la tecnología que se encarga de todas las tareas que se aplican a los robots.
- **Robot:** Es un agente físico que realiza tareas manipulando su ambiente. Generalmente un robot está equipado con actuadores y sensores. Una posible división para categori-

zarlos es por su forma, en primera instancia están los manipuladores o brazos robóticos que normalmente están anclados a su espacio de trabajo y generalmente desempeñan tareas en fábricas o líneas de ensamblaje; como su nombre lo describe, tienen forma de brazo. La siguiente categoría se refiere a robots móviles, su principal característica es que tienen la capacidad de desplazamiento por lo cual poseen piernas, ruedas o cualquier mecanismo que le permita desplazarse, generalmente en esta clasificación se incluyen robots que pueden realizar una gran diversidad de tipos de tareas. La última división es la categoría mixta, son aquellos robots que poseen características tanto de brazos robóticos como de móviles, en ella se incluye a los robots humanoides que, como su nombre lo indica están hechos a semejanza del hombre. Sus tareas suelen requerir un poco más 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 [31].

- **Sensores:** Son dispositivos incorporados a los robots que obtienen información del ambiente. Algunos miden las magnitudes de los cambios externos que ocurren, como las cámaras, los sonares, entre otros. También existen los que miden cambios internos, como los giroscopios y acelerómetros. En general un sensor es una interfaz de percepción entre el ambiente y el robot [31].
- **Actuadores:** Son dispositivos que realizan cambios físicos en el medio ambiente. Por ejemplo ruedas, piernas, pinzas, entre otros [31].
- **Servomotor:** Es un motor eléctrico, un actuador, que permite controlar su velocidad y posición [27].
- **Giroscopio:** Es un sensor de la categoría para la percepción propia que informa al robot de su propio estado, pertenece a los sensores de inercia [31]. 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 su tarea 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 [27]. Existen diferentes estrategias o enfoques de cómo aplicar la inteligencia en un robot. Esta sección se dedica a describir los enfoques que se han definido como paradigmas.

Existen tres paradigmas. Estos serán descritos en base a los conceptos básicos de la robótica: percibir, planificar y actuar. Percibir se refiere al procesamiento de la información obtenida de los sensores del robot. Planificar es, cuando con la percepción, se crea un conocimiento del mundo y se organizan la tareas que el robot debe realizar para lograr un objetivo. Por último, actuar consiste en realizar tareas con los actuadores del robot para modificar el entorno [27].

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 por el momento, 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 [27]. Una representación visual del orden que sigue 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 [27].



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 [27]. Esto se ilustra en la figura 2.3

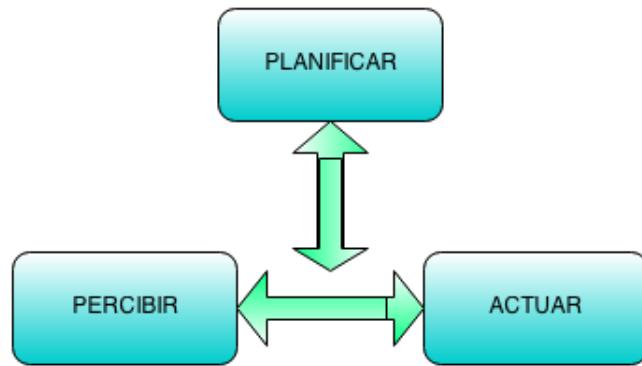


Figura 2.3: Paradigma Híbrido

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 [31] 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 enfocada en lograr 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” [20].

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. Existen dos maneras en las que se puede otorgar las recompensas, darla cada vez que se llega a un estado o una sola vez al llegar al estado final. Un estado se define como la configuración de un conjunto de características, relevantes para el problema, percibidas del mundo en un momento particular.

El agente existe en un entorno compuesto por un conjunto de estados S . En cada estado puede ejecutar una acción del conjunto de acciones A . Cada vez que ejecuta una acción en algún estado el agente recibe una recompensa. 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 [20]. Es decir, aprender cuál es la mejor acción en cada estado.

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 consiste en comparar 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 [31] (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 comenzando desde el estado s y aplicando

a como la primera acción [20]. La ecuación se puede escribir como:

$$Q(s, a) = r(s, a) + \gamma V^*(\delta(s, a)) \quad (2.1)$$

En donde $r(s, a)$ es la recompensa dada 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,

$$V^*(s') = \max_{a'}(Q(s', a')) \quad (2.2)$$

De esta forma se obtiene una definición recursiva,

$$Q(s, a) = r + \gamma \max_{a'} Q(\delta(s, a), a') \quad (2.3)$$

que puede ser calculada de manera iterativa, inicializando los valores de Q con números aleatorios [20]. 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 (usualmente el visible por el ojo humano) 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 valor de la luz que representa el color . Luego de obtener la imagen, esta se procesa para extraer información de ella o para transformarla [27].

Dentro del campo de visión artificial se han desarrollado diferentes algoritmos y técnicas para el procesamiento de imágenes con diferentes objetivos. En la sección 2.4.1 se describe la técnica de segmentación por regiones, que es la utilizada en este proyecto para ubicar la posición tanto de la pelota como del arco.

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 técnica de segmentación por regiones consiste en identificar un conjunto de píxeles adyacentes entre sí que compartan un rango de valores (colores) establecidos. Primero se identifica en la imagen todos los píxeles que se encuentren dentro del rango de color que se requiere segmentar. Luego de esos píxeles se excluyen los que se encuentren aislados. La imagen resultante es binaria, es decir, posee solo dos colores, un color para identificar el área segmentada y otro color para el resto del fondo. Generalmente, en robótica luego de identificar esa zona se ubica el centro y se navega hasta él [27].

2.4.2. Filtros

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

Existen transformaciones morfológicas que se utilizan en una amplia variedad de contextos como la eliminación del ruido, aislamiento de elementos individuales y elementos de unión dispares en una imagen. Algunas de las transformaciones básicas son: dilatación, erosión, unión e intersección [6].

En esta investigación, los algoritmos de filtrado aplicados a las imágenes fueron: Clausura Morfológica y Apertura Morfológica, filtros que aplican las transformaciones morfológicas de erosión y dilatación a las imágenes. Estas transformaciones se explican a continuación.

2.4.2.1. Dilatación

La dilatación es una operación que se le aplica a una imagen A . Esta consiste en elegir un parámetro de forma, B , denominado elemento estructurante, que puede ser de diferentes formas, aunque normalmente es un cuadrado o círculo. Dentro de B se elige un punto de referencia, que normalmente se ubica en el centro. Su función es como la de una plantilla. La operación se aplica pasando el elemento estructurante sobre toda la imagen y realizando el efecto de un operador de máximo local. Es decir, se toma el máximo valor de los píxeles de la imagen que se encuentran en la intersección con B y se reemplaza el píxel de la imagen, ubicado en el mismo lugar que el punto de referencia de B , con ese valor máximo. El resultado es una imagen más brillante y grande [6].

2.4.2.2. Erosión

La operación de erosión es análoga a la de dilatación, con la diferencia de que aplica el mínimo local en vez del máximo. Como resultado se obtiene una imagen más pequeña y oscura que la original [6]. Véase en la figura 2.5

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.

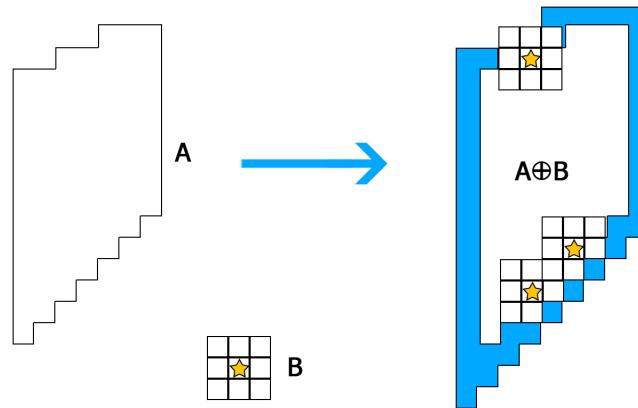


Figura 2.4: Dilatación: A es la imagen original, B es el elemento estructurante, La estrella es el punto de referencia. Se ve como aumenta la imagen en proporción al patrón aplicado

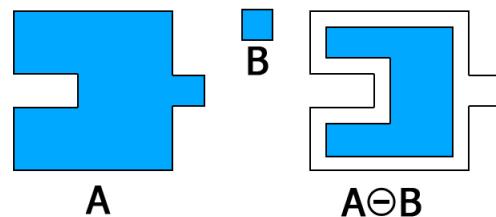


Figura 2.5: Erosión, A: es la imagen original, B: es el núcleo, La estrella es el punto de referencia. Se ve como disminuye la imagen en proporción al patrón aplicado

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 que sepa como levantarse en caso de tener un desbalance y caer. El código completo de este proyecto se encuentra disponible en: <https://github.com/jenniferdr/raspberryROS>

3.1. Partes del robot

A continuación se presenta una descripción de los elementos que han sido seleccionados para la construcción del robot humanoide.

Las opciones que se han considerado para la construcción del cuerpo de Junny han sido: uso de piezas de LEGO, construcción y diseño mecánico desde cero o el uso de las piezas del kit Bioloid [18]. Estas opciones corresponden a los recursos disponibles. Tanto la construcción con piezas de LEGO como la opción de realizarlo desde cero implicaba más 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 su 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, Robotics [18], incluye un manual con varios modelos de robots con instrucciones de ensamblaje. Provee una tarjeta controladora, CM-510 [16], a la que se conectan los motores Dynamixel y algunos sensores que se programan a través de la interfaz de ‘RoboPlus’ [18].



Figura 3.1: Bioloid Premium Kit

Se decidió sustituir la tarjeta controladora CM-510 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 XBEE, conductores de motor dual, y cabeceras de estilo servo de 3 pines para entrada/salida digital y analógica [19].

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 como se muestra en la figura 3.2. Estos motores estan 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. [28]

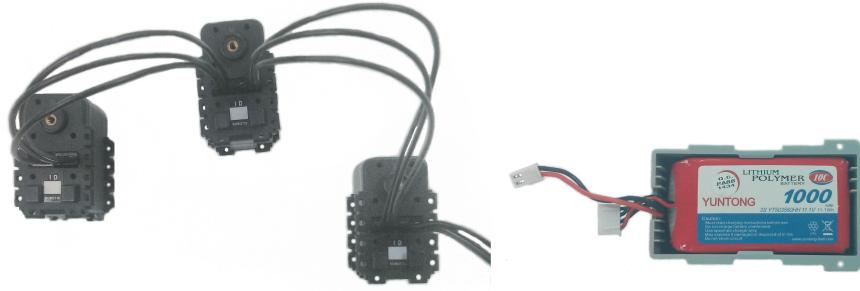


Figura 3.2: A la izquierda, 3 motores Dynamixel conectados en serie. A la derecha, batería Lipo

Para la detección de caídas se ha seleccionado un giroscopio (Robotics Gyro) que mide la velocidad angular que genera los movimientos del robot, este se encuentra diseñado para verificar el balance del robot y ser usado para otras aplicaciones de movimiento [15]. También se encuentra incluido en el kit Bioloid. En figura 3.3 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, sin embargo estos no se encontraban disponibles en el laboratorio.

Ha sido necesario incluir un extensor de puertos Bioloid que permitiera conectar un mayor número de motores Dynamixel, este se muestra en la figura 3.3 [38]. En la siguiente sección se explica como han sido conectados los motores y el motivo para incluir el expansor.

Para agregar mayor capacidad de procesamiento, con la finalidad de poder captar y procesar imágenes, se incorporó el mini computador Raspberry Pi, de aproximadamente 10 cm de largo, a la que se puede conectar un módulo de cámara especialmente diseñado para él. Puede ser utilizado en proyectos de electrónica y para varias de las tareas que un computador de escritorio puede hacer, como procesamiento de hojas de cálculo y de texto, ejecutar

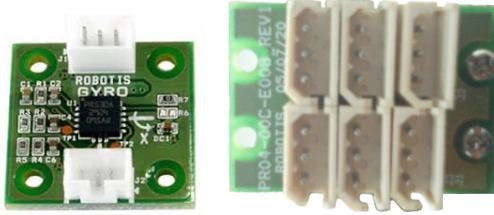


Figura 3.3: A la izquierda, sensor Gyro. A la derecha, extensor de puertos Bioloid

juegos, entre otras [23]. Su estructura se puede ver en la figura 3.4. La Raspberry Pi cuenta con un procesador gráfico que permite aligerar la carga del procesador central [8]. 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 agrega un peso importante a la estructura del robot, dificultad para añadirle grados de libertad a su cámara y un costo superior al de la Raspberry.

Para lograr el objetivo de detección de la pelota y orientación al arco existía la posibilidad de añadir un sensor de brújula y un sensor de proximidad. Sin embargo la ultilización de una cámara para esta tarea representaba mayores ventajas. La cámara posee un mayor rango de percepción, aumentando las posibilidades de encontrar la pelota o arco en menor cantidad de tiempo que con los sensores. Además, la incorporación de la Raspberry Pi [23] y su cámara permite la posibilidad de ampliar las habilidades y tareas que podría realizar Junny. La cámara puede captar imágenes y grabar vídeos de alta definición. 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 [11]

Para poder establecer la comunicación entre la Arbotix y la Raspberry Pi, se añadió el chip FTDI (Future Technology Devices International) [9]. Es una tarjeta controladora (se puede observar en la figura 3.5) que ofrece el servicio de conversión de datos de USB a UART, por lo cual permite la comunicación entre diferentes dispositivos [9]. Fue elegida en lugar de la radio inálambrica XBEE, pues la Raspeberry Pi y la Arbotix estan físicamente



Figura 3.4: Tarjeta Raspberry Pi y su módulo de cámara a la derecha

cerca y usar los XBEE implicaba un mayor costo para el proyecto.

Para añadir grados de libertad a la cámara se utilizó como base dos micro servomotores analógicos TG9e [13]. La capacidad de torque de estos motores alcanza los 1.50 kg-cm . Permiten ser controlados en posición en un rango de 180°. Ver figura 3.6.



Figura 3.5: Chip FTDI conectado a la tarjeta Arbotix



Figura 3.6: Micro Servomotores analógicos TG9e

Debido a que no todos los componentes poseen las mismas exigencias con respecto a voltaje y amperaje, se realizó un regulador (ver figura 3.7) 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.

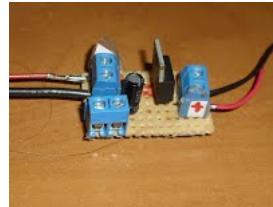


Figura 3.7: Circuito con entrada de 11.1 V. Una salida de 5 V para los micro servomotores analógicos y tarjeta Raspberry Pi. Otra salida de 11.1 V para alimentar la controladora Arbotix.

3.2. Ensamblaje

Como se mencionó 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 [14], 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 3.8 y 3.9 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 recueros eran escasos y a pesar de que el modelo A posee un grado más de libertad que los otros modelos (incluye tanto los de B como los de C) utiliza dos motores más que los otros dos modelos. El modelo B posee un grado de libertad de interés a la hora de girar del robot, que el modelo C no posee, por ello se eligió este.

En la figura 3.10 se puede observar la estructura del robot con la Arbotix incorporada. En la parte interna del tronco del robot se sitúa el sensor Gyro.

Para el movimiento de la cámara se ha incorporado dos micro servomotores TG9e ya que

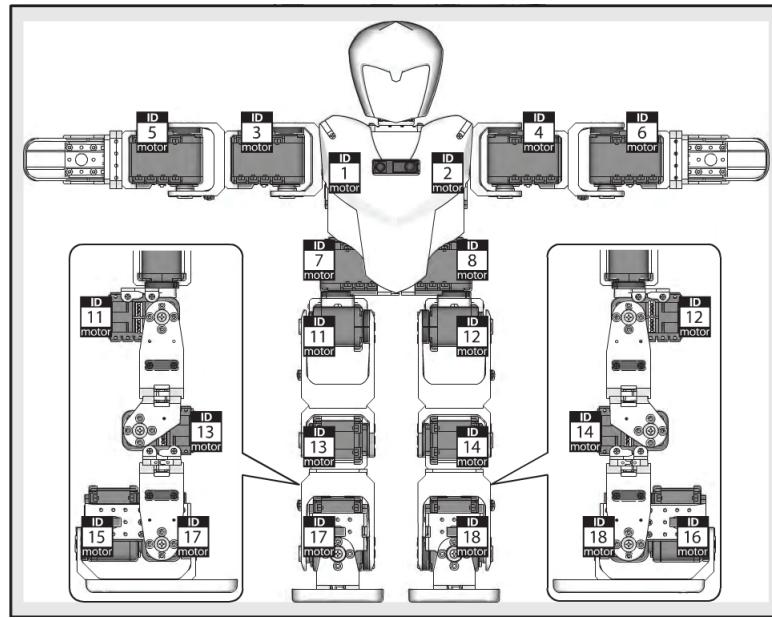


Figura 3.8: Vista frontal del robot, tipo B. Se puede apreciar la identificación ‘ID’ de cada motor Dynamixel Ax-12+. Nota: los motores 9 y 10 no se utilizan. Imagen tomada sin autorización de [14]

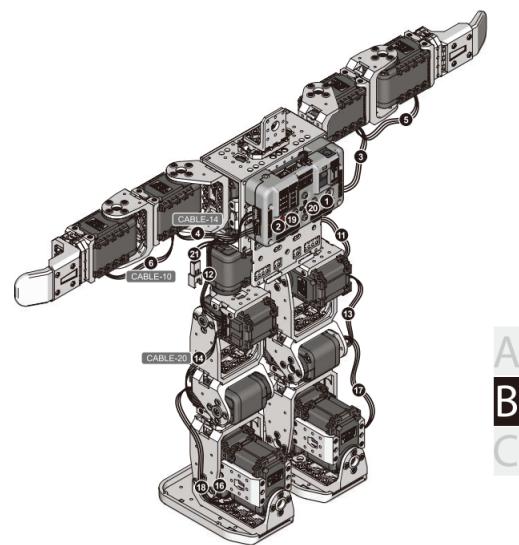


Figura 3.9: Vista trasera del robot con la tarjeta CM-510, Imagen tomada sin autorización de [14].



Figura 3.10: Vista posterior del robot, con la tarjeta Arbotix integrada (tarjeta azul) y la batería Lipo

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 3.12, 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 con un cable de cinta plana de 15 cm. El resultado de estas tres piezas instaladas en el robot se puede apreciar en la figura 3.11.



Figura 3.11: Parte delantera del robot incluye cámara, los dos servo motores (azules) y raspberry pi

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 en un puerto diferente. La forma en la que se ha conectado estos motores se ejemplifica en la figura 3.12.

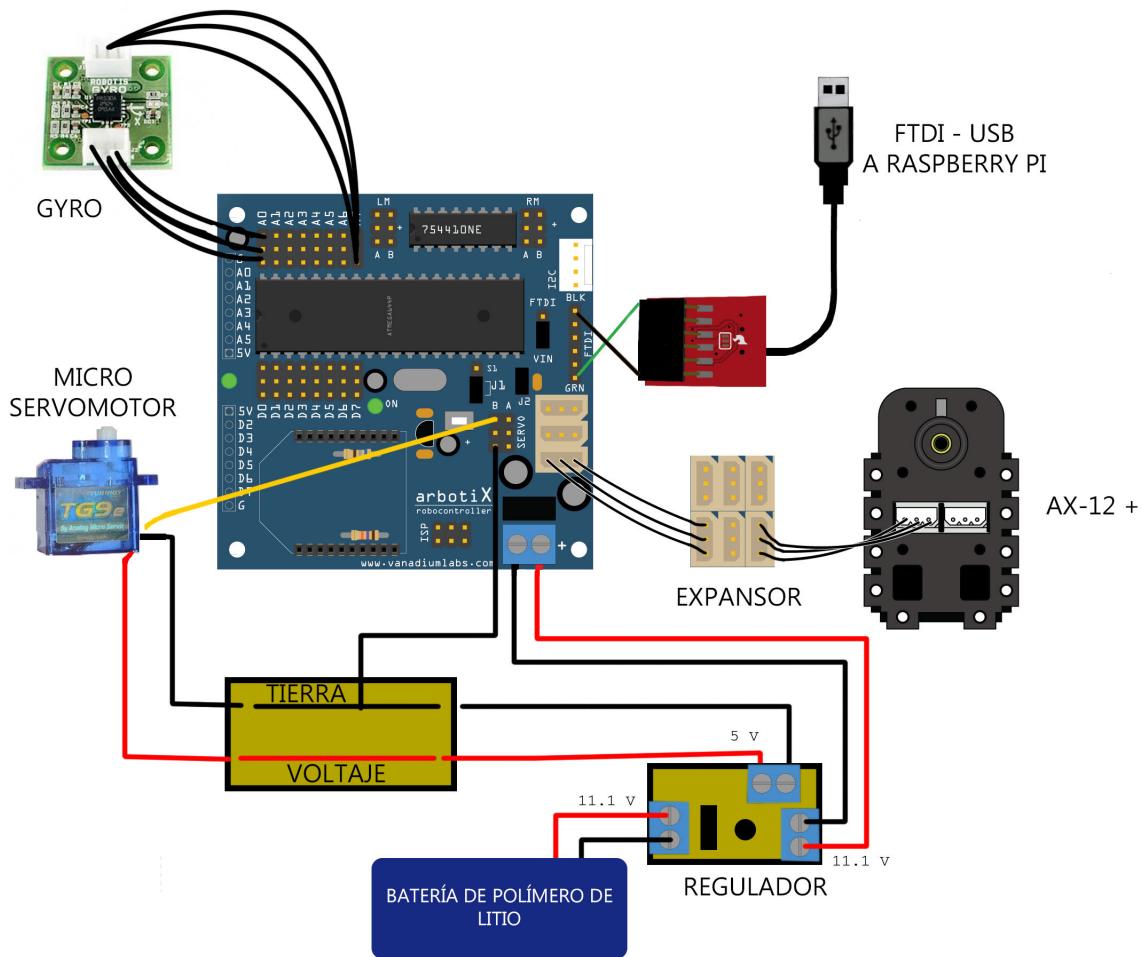


Figura 3.12: Tarjeta controladora Arbotix y componentes conectados (Imagen referencial: el regulador se encuentra dispuesto en diferente orden)

La comunicación de la tarjeta de Arbotix con la computadora, incluso con la Raspberry Pi, se realiza a través del chip FTDI, un extremo se conecta en el puerto FTDI de la tarjeta Arbotix, como lo ilustra la figura 3.12, y el otro extremo se conecta al puerto USB de la

tarjeta Raspberry Pi.

Para evitar agregar peso innecesario al robot se decidió usar una sola fuente de poder. Se utilizó una batería de polímero de litio de 11.1 V y 1 amp. Y se construyó un circuito para regular el voltaje de los componentes que necesitan 5 voltios (ver figura 3.7).

3.3. 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 [24]. Esta mini computadora y cámara constituyen una herramienta poderosa ya que permite capturar videos y fotos de alta definición. Y 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 [5].

Además se ha decidido utilizar OpenCv [34], otra herramienta para procesar imágenes que se describe en la sección 3.3.1. Sin embargo los métodos de captura de OpenCV no funcionan con la cámara Raspberry Pi. En la sección 3.3.2 se explica cómo se ha extraído la imagen y en la sección 3.3.3 se explica la forma en la que se ha procesado la imagen para hallar la posición de la pelota.

3.3.1. 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 [21] y OpenCv. Processing

es un lenguaje de programación que esta 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 [34]. 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 [36].

3.3.2. Obtención de la imagen

Dentro de las librerías oficiales para la cámara Raspberry Pi sólo se encuentran aquellas implementadas en el lenguaje interpretado python y algunas aplicaciones para la línea de comandos de linux. 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 utilizar la cámara con OpenCV en el lenguaje compilado C++ se requirió realizar una búsqueda de librerías alternas a las oficiales. Se hayó una primera solución modificando las aplicaciones Raspivid y Raspistill para lograr extraer la imagen del buffer de la cámara y así obtener un objeto compatible con OpenCV [25]. Sin embargo era poco práctico porque se debía incluir todo el código en el proyecto y compilarlo unido. Luego se encontró que en base en ese trabajo, se había construido una librería de código abierto. Esta librería se llama raspicam_cv y es la que ha sido utilizada en este proyecto [36].

3.3.3. 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 no 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 máquinas.

La imagen es captada en el modelo de color BGR, sin embargo para la segmentación de regiones este modelo no es conveniente por estar basado en el reflejo de la luz. El reflejo de la luz que proyecta un objeto puede variar dependiendo de su posición y distancia. Por esta razón la imagen se convierte al modelo de color HSV, que se basa en el matiz (longitud de onda absoluta de la luz reflejada). Por lo que su valor no depende de la posición o distancia del objeto [27]. En la figura 3.13 se puede observar la diferencia entre la imagen original en BGR y su transformación a HSV.

Luego se aplica la función `inRange` de OpenCv para obtener una imagen binaria, en donde se identifica con blanco la zona con el color de la pelota y el resto de la imagen en negro. En la figura 3.13 (c) se puede ver un ejemplo de la imagen binaria obtenida. En el procesamiento de imágenes y visión de computación generalmente se trabaja con las imágenes en escala de grises, pues disminuye el tiempo en procesar datos al eliminar información inútil y aumenta la eficiencia.

Para disminuir el ruido y los posibles elementos aislados que pueda tener la imagen con

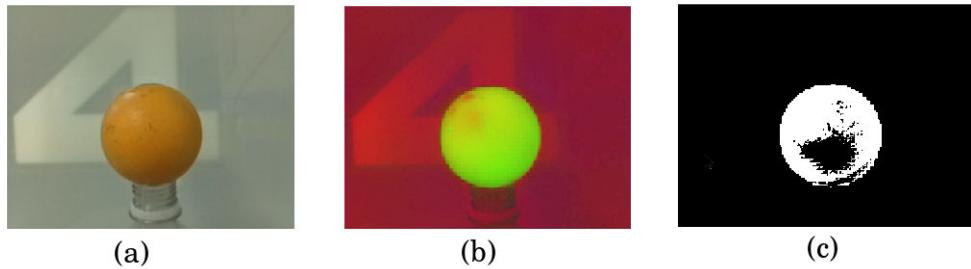


Figura 3.13: (a) imagen original en BGR. (b) imagen original transformada a HSV. (c) imagen binaria a partir de la imagen HSV y el color de la pelota

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 el 100 % de las pruebas realizadas, que se describen en el capítulo 4.

3.4. 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 3.4.1 se da una breve descripción de las herramientas de software que apoyaron las tareas de búsqueda y pateo. En la sección 3.4.2 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 3.4.3 se explica las posiciones que puede adoptar la cámara. Luego en la sección 3.4.4 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 3.4.5.

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 3.4.6.

3.4.1. 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 [10] 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 sus 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 .

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 [4].

ROS (Robot Operating System) [29] 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 Licencia BSD .

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 [37].

3.4.2. Movimiento de extremidades

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
- Girar a la izquierda

- Girar 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

En el movimiento de caminar se establece como unidad el de ejecutar dos pasos seguidos, este fue construido manteniendo el equilibrio del robot para evitar posibles caídas e inestabilidad. La distancia recorrida por ese paso es de 4.8 cm, analogamente se obtiene las distancias recorridas de los giros de derecha e izquierda de 3 cm cada una.

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.

3.4.3. 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 otorgale 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, hubiera implicado movimientos constantes que habrían consumido mucho mas tiempo y desgaste en motores de mayor costo.

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 3.4.4. En la figura 3.14 se puede observar la cámara del robot en las diferentes posiciones que se han fijado para ella.

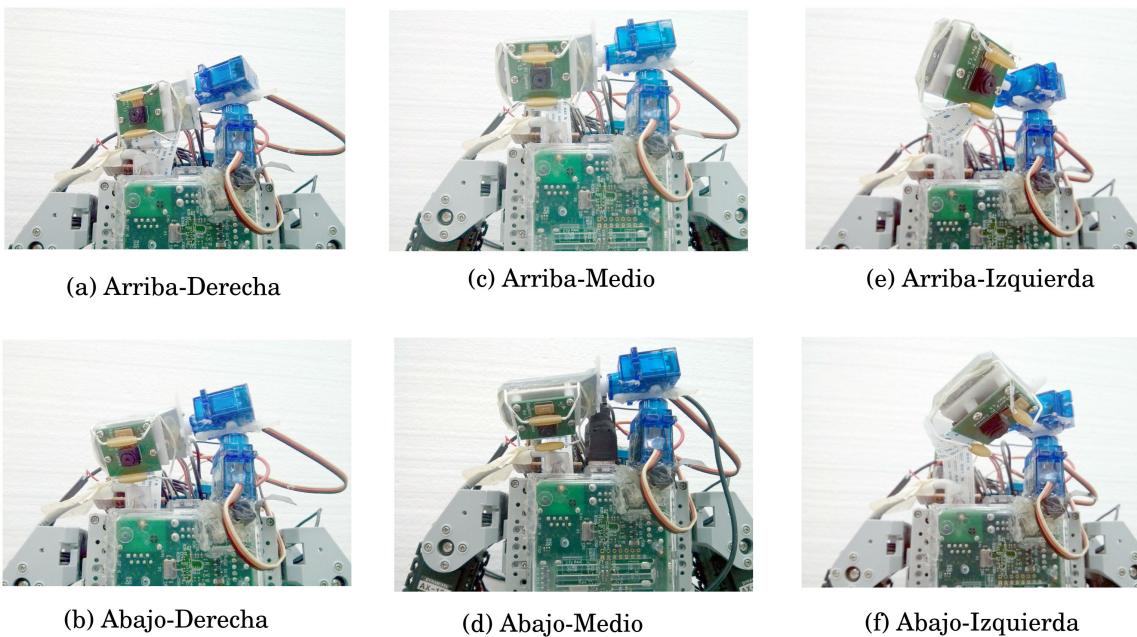


Figura 3.14: Posiciones de la cámara

Los micro servomotores azules que aparecen en la imagen 3.14 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 3.12). 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.

3.4.4. 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 3.14 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 3.15. 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 3.15 para simplificar su representación.

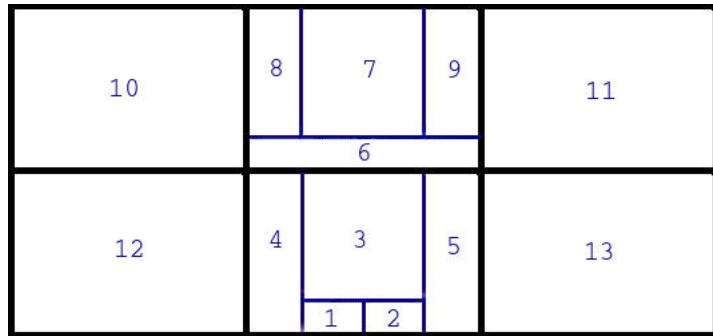


Figura 3.15: Campo de visión del robot con el número de cada región. Cada cuadro blanco demarcado con líneas negras representa la imagen captada desde una posición fija de la cámara. La región de pateo de la pelota se encuentra en las regiones 1 y 2

3.4.5. 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 request / reply con el concepto de maestro esclavo.

3.4.6. 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.

3.5. Aprendizaje

En aprendizaje de máquinas el aprendizaje por reforzamiento es una forma de plantear el problema de inteligencia completamente distinta al aprendizaje supervisado o no supervisado que generalmente está basado en ejemplos, pares entrada/salida y el objetivo del agente es aprender la función que haya producido esos pares. Sin embargo no siempre existe un

supervisor que pueda verificar los resultados otorgados. Existen ambientes en la vida real donde al agente no se le proporciona ni el supervisor ni los ejemplos, en donde se comienza sin contar con un modelo del ambiente o una función de utilidad. En estos casos la aplicación del aprendizaje por reforzamiento es la más indicada [32].

La técnica de refuerzo es aplicada en la vida diaria constantemente, pues está basada en comportamientos. Es la forma en que se entrena a una mascota, es la manera en que un bebé logra aprender a sujetar objetos, esos son ejemplos de los muchos en los que en la realidad aparece esta técnica de aprendizaje.

Cuando es aplicada a inteligencia artificial las bases son las mismas, obviamente a un robot no se le podrá dar una galleta literalmente como premio de su buena acción o quitarle la televisión como castigo.

Existen varias estrategias que permiten otorgar recompensas positivas o negativas a un agente, en este caso un robot, con ellas ir moldeando el comportamiento que se considere correcto.

Un algoritmo para aprender por reforzamiento es aprendizaje-Q, que es el utilizado para que Junny aprenda a acercarse a la pelota.

Según [20] 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 [20], 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.

3.5.1. 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 3.5.1.1 se define el conjunto de estados que conforman el mundo de Junny. En la sección 3.5.1.2 se dan a conocer las posibles acciones definidas y en la sección 3.5.1.3 se define la ecuación para calcular las recompensas.

3.5.1.1. 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 3.4.4 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 3.16.

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 3.16 y el estado número 18 representa la ocasión

en la que no se logra ubicar la pelota en ninguna de las regiones.

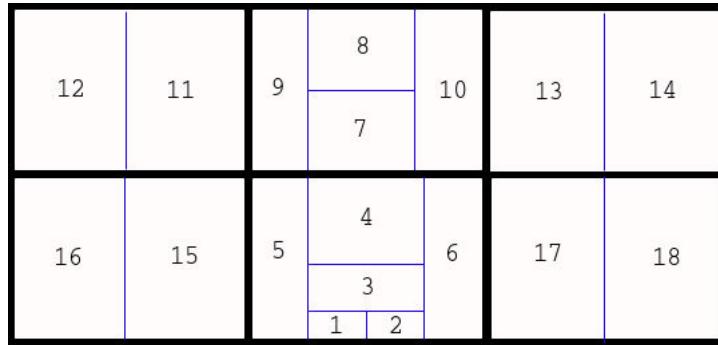


Figura 3.16: Estados definidos según la región en la que se detecta la pelota

3.5.1.2. Acciones

Las posibles acciones a realizar son un subconjunto de las acciones de movimiento definidas en la sección 3.4.2. Además se agregaron otras acciones para mejorar el desempeño en la búsqueda de la pelota. Las acciones disponibles son:

- Caminar un paso hacia adelante
- Caminar dos pasos hacia adelante
- Caminar cuatro pasos hacia adelante
- Girar a la izquierda
- Girar doble a la izquierda
- Girar a la derecha
- Girar doble a la derecha

3.5.1.3. 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 3.17. 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 $d(18) = 10$.

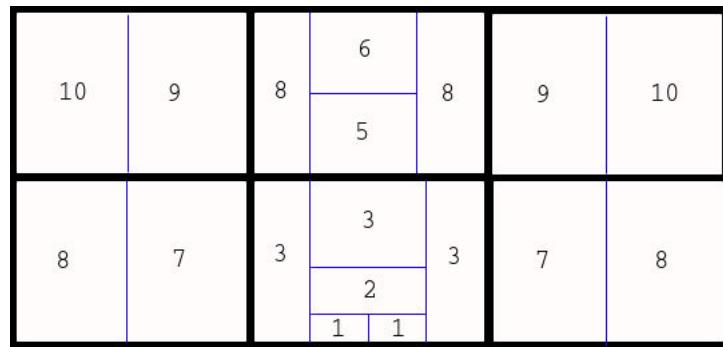


Figura 3.17: Distancias relativas de la pelota establecidas para otorgar la recompensa.

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:

$$R(s, a, s') = \frac{d(s) - d(s')}{10} \quad (3.1)$$

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

3.5.2. 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

$$P(a_i|s) = \frac{k^{Q(s,a_i)}}{\sum_j k^{Q(s,a_j)}} \quad (3.2)$$

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

3.5.3. Actualización de $Q(s,a)$

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

$$Q(s, a) = r + \gamma \max_{a'} Q(\delta(s, a), a') \quad (3.3)$$

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.

3.6. 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 3.18.

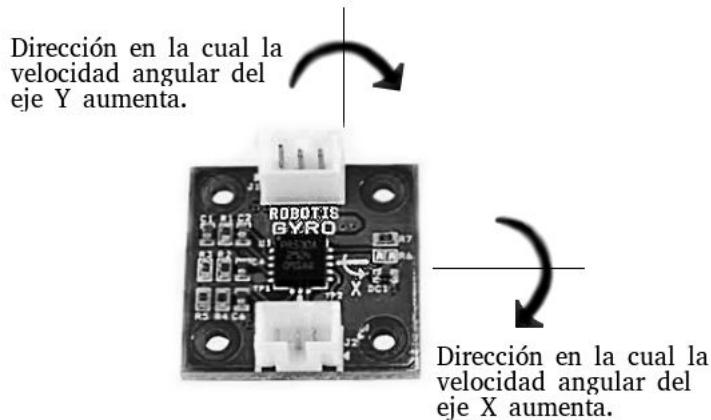


Figura 3.18: Dirección en la que la velocidad angular de los ejes X y Y , del Gyro, aumenta.

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 3.19 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 $0^{\circ}/s$.

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

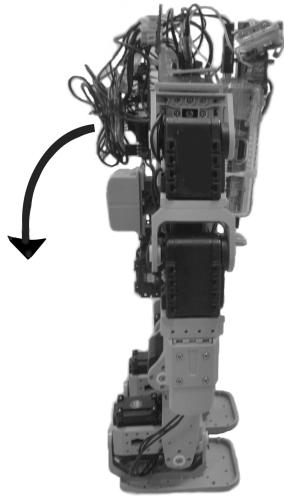


Figura 3.19: Dirección en la que la velocidad angular del eje X crece con respecto al robot.

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.

3.7. 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 3.4. Se tiene la representación del mundo dividida en regiones como se muestra en la figura 3.20. 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 3.14. En 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. A diferencia de la representación cuando se busca la pelota en esta se reducen los tres movimientos de la cámara por dos razones principales. En primera instancia las acciones eliminadas apuntan al suelo y el arco no puede encontrarse en el suelo, la segunda razón es que el arco es en proporción mucho mas grande que la pelota por eso la divisiones de las regiones tan presisas como en el caso de la pelota son innecesarias.

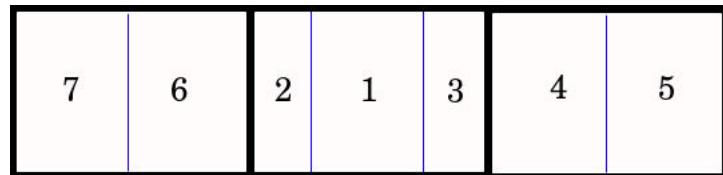


Figura 3.20: Campo de visión del robot con el número de cada región para buscar el arco.

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:

- Cuando el arco esta a la izquierda de Junny este debe girar a la izquierda esto sucede en las regiones 2, 6, 7, con esto se centra con respecto al arco logra tener un mayor angulo de patada al arco.

- Cuando el arco esta a la derecha de Junny este debe girar a la derecha esto sucede en las regiones 3, 4 y 5, con esto se centra con respecto al arco y tener mayor angulo de patada al arco.

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

Para lograr el ajuste adecuado cuando Junny encuentra el arco y debe colocarse en posición de pateo los movimientos que debe realizar su cadera no lo permite pues necesita un grado de libertad más para lograrlo, este grado de libertad fue eliminado al principio del proyecto debido a ciertos problemas con los motores Dynamixel los cuales debido al uso constante y exigente suelen fallar y quemarse, la mayoría de estos motores se quemaron justamente en la zona de la cadera. De igual manera los experimentos de desempeño que incluyen patear al arco se realizaron y se presentan en el capítulo 4.

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 (experimentos simples). 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 la 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. Finalmente se describe los experimentos con el entrenamiento terminado y la orientación al arco (experimentos completos) en la sección 4.4.

Algunos videos de los experimentos se encuentran disponibles en: <https://www.dropbox.com/sh/b1qxstva3o2hmu8/AAAblx5ztJBwGylXtVWWbfJva?dl=0>

4.1. Experimentos de Movimientos

Se realizaron una serie de experimentos para comprobar la movilidad del robot en cuanto a un conjunto de acciones de movimiento (mencionadas en la sección 3.4.2). Estas acciones son:

1. Caminar un paso hacia adelante
2. Caminar dos pasos hacia adelante
3. Caminar cuatro pasos hacia adelante
4. Girar hacia la derecha
5. Girar doble a la derecha
6. Girar hacia la izquierda
7. Girar doble a la izquierda
8. Patada con la pierna derecha
9. Patada con la pierna izquierda

La primera etapa consistió en verificar el desempeño de las 9 unidades de acciones, para ello se procedió a realizar la ejecución de cada unidad 50 veces, 450 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. Los resultados se muestran en la tabla 4.1.

Para la segunda etapa se evaluó la ejecución de combinaciones de una serie de unidades de movimiento. Las posibles combinaciones que se pueden generar con las acciones es 2^9 por

Movimiento	Cantidad de pruebas	Correctas	Fallidas	Con recuperación
1	50	100 %	0 %	NA
2	50	98 %	2 %	0 %
3	50	100 %	0 %	NA
4	50	100 %	0 %	NA
5	50	100 %	0 %	NA
6	50	100 %	0 %	NA
7	50	100 %	0 %	NA
8	50	98 %	2 %	2 %
9	50	100 %	0 %	NA

Figura 4.1: Resultados pruebas individuales de movimiento

lo cual se consideró sólo un subconjunto de estas combinaciones, aquellas que el robot realiza con más frecuencia al caminar. Las combinaciones elegidas fueron:

1. Caminar hacia adelante y patear con la pierna derecha
2. Caminar hacia adelante y patear con la pierna izquierda
3. Caminar hacia adelante, girar hacia la derecha y patear con la pierna derecha
4. Caminar hacia adelante, girar hacia la derecha, patear con la pierna izquierda
5. Caminar hacia adelante, girar hacia la izquierda y patear con la derecha
6. Caminar hacia adelante, girar hacia la izquierda y patear con la pierna izquierda
7. Girar hacia la izquierda y patear con la pierna derecha
8. Girar hacia la derecha y patear con la izquierda
9. Girar hacia la derecha y patear con la pierna derecha
10. Girar hacia la izquierda y patear con la pierna izquierda

Se realizaron 30 ejecuciones de cada una de estas combinaciones, 300 ejecuciones en total. El resultado ha sido satisfactorio, obteniendo un 99 % de casos exitosos y un 1 % de casos con fallas, de las que ha podido recuperarse. La mayoría de las fallas se presentaron cuando al

Combinación	Cantidad de pruebas	Correctas	Fallidas	Con recuperación
1	30	96.7 %	3.3 %	3.3 %
2	30	100 %	0 %	NA
3	30	100 %	0 %	NA
4	30	100 %	0 %	NA
5	30	96.7 %	3.3 %	3.3 %
6	30	100 %	0 %	NA
7	30	100 %	0 %	NA
8	30	100 %	0 %	NA
9	30	100 %	0 %	NA
10	30	100 %	0 %	NA

Figura 4.2: Resultados de los experimentos con las acciones combinadas elegidas

	Cantidad de Pruebas	Correctas
A 50cm	50	100 %
A 80 cm	50	100 %
No pelota	50	100 %

Figura 4.3: Resultados de las pruebas de detección

patear se caía, sin embargo lograba recuperarse. Los resultados de estas pruebas se resumen en la figura 4.2.

Para determinar la precisión y desempeño de la detección de la pelota se realizaron experimentos de enfoque rápido los cuales consistieron en la colocación de la pelota a una distancia fija, mover la cámara rápidamente y regresar a la posición y observar si ubicaba la pelota correctamente, las distancias utilizadas fueron 50cm y 80cm del robot. Además para determinar posibles Falsos Positivos, es decir, detectar una pelota inexistente, se realizó el mismo experimento pero sin la pelota. Los resultados obtenidos se presentan en la tabla 4.3.

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

	Cantidad de pruebas realizadas	Correctas	Con fallas recuperadas	Fallidas	Tiempo Promedio
Caso I	10	90 %	10 %	0 %	30 s
Caso II	10	70 %	10 %	20 %	2m 12s
Caso III	10	80 %	0 %	20 %	3m 29s
Caso IV	10	90 %	0 %	10 %	4m 48s

Figura 4.4: Resultados de los casos I, II, III, IV descritos en la sección de 4.2

integrados (detección, búsqueda y pateo). La forma de elegir la acciones de movimiento es la que se describe en la sección 3.4.6 (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 3.15). El número de pruebas en este primer caso fue de 10, cuya duración aproximada de cada prueba fue de 30 segundos. En la figura 4.4 se puede observar los resultados obtenidos de esta prueba.

El segundo caso (CasoII) consistió en en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot. Se realizaron 50 pruebas, siendo la duración promedio de 2 minutos con 12 segundos. Los resultados de dicho caso se pueden observar en la figura 4.4.

El tercer caso (CasoIII) consistió en en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot y 50 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 en la colocación inicial de la pelota a una distancia de 50 cm en línea recta al robot y 50 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 4.4.

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, esto a pesar del 12.5 % de fallas que se produjo durante la tarea.

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 3.5.2, 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

	Cantidad	Correctas	Fallidas
$K = 1\gamma = 0,1$	20	80 %	20 %
$K = 2\gamma = 0,1$	20	80 %	20 %
$K = 2\gamma = 0,7$	20	60 %	40 %
$K = 3\gamma = 0,1$	20	90 %	10 %
$K = 3\gamma = 0,7$	20	80 %	20 %
$K = 5\gamma = 0,1$	20	80 %	20 %
$K = 5\gamma = 0,7$	20	70 %	30 %

Figura 4.5: Resultados de los distintos parámetros con aprendizaje

de $1,8m \times 1,6m$ en donde se aseguraba que el color particular de la pelota no se repitiera, 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. La elección de sólo 20 pruebas para el conjunto de entrenamiento se debió a que los motores son delicados, por lo cual mientras se aumentaba en número de pruebas aumentaba el riesgo de quemarlos.

Para la combinación de $K = 1$ y $\gamma = 0,1$ los resultados obtenidos se presentan en la tabla 4.5. 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 de las combinaciones de parámetros ajustables se presentan en la tabla 4.5.

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 parti-

cular, serían obviamente erróneos. Con esto se ayuda al aprendizaje del robot, disminuyendo 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 son de 100 % de éxitos con $K = 3$ y $\gamma = 0,1$ e inicialización de $Q(s,a)$.

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 uno (1) se considera una mejor tasa de eficiencia. Para valores mayores a uno (1) significa que se realizaron menos acciones de lo 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 parámetros 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.

	Caso II	Caso III	Caso IV
Correctas con aprendizaje	100 %	100 %	100 %
Correctas sin aprendizaje	70 %	80 %	90 %
Fallidas con aprendizaje	0 %	0 %	0 %
Fallidas sin aprendizaje	20 %	20 %	10 %
Recuperadas con aprendizaje	NA	NA	NA
Recuperadas sin aprendizaje	10 %	0 %	0 %
Tiempo promedio con aprendizaje	2m 35 s	5m 38s	4m 14s
Tiempo promedio sin aprendizaje	2m 12s	3m 29 s	4m 48s

Figura 4.6: Comparación de movimientos predeterminados y aprendizaje

Se realizó un experimento comparativo para observar el desempeño del comportamiento sin aprendizaje y con aprendizaje, por ello, se hizo el mismo experimento de los casos II, III, IV (explicados en la sección anterior) pero con el comportamiento generado después del entrenamiento con la combinación $K = 3$ y $\gamma = 0,1$. Los resultados obtenidos fueron un mayor porcentaje de aciertos para el aprendizaje pero a un costo de tiempo que se puede observar en la tabla 4.6, cada prueba en ambos casos ejecutó 10 veces.

Un resultado a destacar es que el tiempo promedio de pruebas a la derecha (Caso IV) es en proporción menor a las pruebas a la izquierda (Caso III) a pesar que las mismas son simétricas. Esto indica que los siguientes entrenamientos deben hacer énfasis en los estados a la izquierda del robot.

4.4. Experimentos completos

Finalmente se realizaron experimentos con aprendizaje y patada con orientación, básicamente el robot partía de un punto fijo y la pelota era colocada arbitrariamente en el campo de entrenamiento, el debía llegar a la pelota, ubicar el arco, posicionarse con respecto al arco y la pelota para realizar un pateo y meter gol. Las métricas usadas fueron el pateo hacia el arco que generaba un gol y el que pateaba pero no lograba entrar el balón. Estos resultados

se muestran en la siguiente figura 4.7.

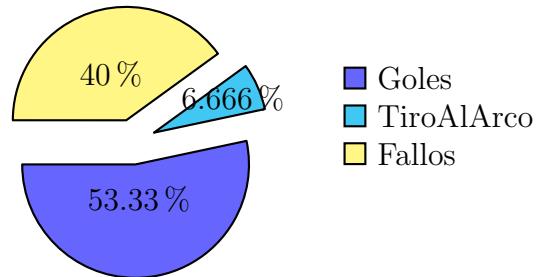


Figura 4.7: Prueba con 15 ejecuciones, con aprendizaje y orientación al arco

Capítulo 5

Conclusiones y Recomendaciones

La presente investigación nace de la motivación de que en Venezuela se incursione en proyectos con humanoides autónomos e inteligentes. Se ha inspirado especialmente en la categoría Robocup soccer de la competencia internacional Robocup. Desde 1997, fecha en la que se inicia la competencia, Venezuela no ha participado en categorías con humanoides, mientras que países latinoamericanos como México, Brasil y Colombia sí han tenido presencia en estas categorías. Si bien este proyecto no cumple con todas las reglas de la competencia se espera que éste pueda dar pie a continuar investigaciones en el país.

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.

Se logró el diseño y construcción de Junny con la tarjeta Arbotix, obteniendo un 99.7 % de movimientos correctos simples y un 96 % en movimientos combinados correctos, los cuales son utilizados para el desplazamiento. La detección de la pelota arrojó un 100 % de aciertos en las pruebas de enfoque realizadas. La integración de todos los componentes involucrados

fue satisfactoria logrando así un comportamiento autómono e inteligente. Los resultados obtenidos de las pruebas con desplazamiento predeterminado varian según el caso; la menor tasa es la del caso II con 70 % de pruebas completadas con éxito y la mayor es de 90 % en los casos I y IV. Con aprendizaje por reforzamiento, para el desplazamiento hacia la pelota luego del entrenamiento, se obtuvo un 100 % de pateo directo al llegar a la pelota y una tasa de eficiencia de 0.72. Finalmente en la orientación para el pateo al arco se obtuvo un 53.3 % de aciertos con resultado de gol. El porcentaje obtenido para la orientación al arco puede deberse a que él mismo requiere rodear la pelota para posicionarse correctamente donde la pelota esté, entre el arco y Junny. Para realizar el rodeo se require movimientos de desplazamiento laterales que las caderas no pueden hacer, esto debido a que el grado de libertad faltante tuvo que ser eliminado para el ahorro de motores. Para más información puede revisar los apéndices A.

Se puede observar en la comparación de pruebas con movimientos predeterminados y pruebas con aprendizaje que estas últimas obtuvieron un mejor resultado en acertividad. Sin embargo lo hizo comprometiendo el tiempo, de igual manera se observó que con el aumento de pruebas de entrenamiento aumenta la eficiencia, por lo tanto se puede decir que con el continuo entrenamiento Junny podría disminuir estos tiempos.

A pesar del costo del tiempo se recomienda la utilización del aprendizaje pues no sólo logra realizar la tarea con mayor fidelidad sino que además es adaptable por ejemplo si las condiciones físicas del robot llegaran a cambiar en este caso sólo haría falta volverlo a entrenar sin tener que cambiar el código del programa.

Un resultado específico de este proyecto fue la aceptación del artículo “Integración de Arbotix, Raspberry Pi y motores Dynamixel Ax-12+ con el objetivo de la construcción de un robot humanoide que busque y patee pelotas” [26] en el Congreso de Reconocimiento de Patrones, Control Inteligente y Comunicaciones de la Universidad de Cuenca en Ecuador a

realizarse en Diciembre 2014. La carta de aceptación de encuentra en el anexo B.

Para la continuación de este proyecto se recomienda perfeccionar el aprendizaje para obtener una mayor eficiencia en el número de 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. También se puede plantear la utilización de otros motores que logren resistir más las exigencias en velocidad, toque y fuerza del proyecto.

Otra sugerencia puede ser probar con aprendizaje para la detección de la pelota, realizándolo en base a formas, pues, aunque el resultado de la detección utilizada fue satisfactorio tiene la limitante de que el color debe ser único en el ambiente lo cual es improbable en la vida real.

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

Se espera que este proyecto sea un primer paso y un impulso para continuar las investigaciones con robots humanoides y que Junny pueda tener un compañero de juego para incluir numerosos aprendizajes y más habilidades .

Bibliografía

- [1] RoboCup 2014. Robocup 2014. URL: <http://www.robocup2014.org/> [cited 15.10.2014].
- [2] RoboCup 2014. Robocup soccer. URL: http://www.robocup2014.org/?page_id=51 [cited 15.10.2014].
- [3] Aldebaran. Nao robots, 2014. URL: <http://www.aldebaran.com/en/humanoid-robot/nao-robot> [cited 15.10.2014].
- [4] Arduino. Arduino, 2014. URL: <http://arduino.cc/en/Guide/Introduction> [cited 15.10.2014].
- [5] Artisan's Asylum. Introduction to raspberry pi. URL: <http://raspberrypi-aa.github.io/session3/camera.html> [cited 15.10.2014].
- [6] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. 2008.
- [7] Honda Motor Co. Honda unveils all-new asimo with significant advancements, 2011. URL: <http://world.honda.com/news/2011/c111108All-new-ASIMO/index.html> [cited 15.10.2014].
- [8] Comunidad de elLinux.org. Rpi hardware, 2014. URL: <http://elinux.org/RPi-Hardware> [cited 15.10.2014].

- [9] SparkFun Electronics. Ftdi sparkfun. URL: <https://www.sparkfun.com/products/9716> [cited 15.10.2014].
- [10] M Fergs. Pypose, 2010. URL: <https://code.google.com/p/arbotix/wiki/PyPose> [cited 15.10.2014].
- [11] Raspberry Pi Foundation. Camera module setup. URL: <http://www.raspberrypi.org/help/camera-module-setup/> [cited 15.10.2014].
- [12] Andrej Gams, Jesse van den Kieboom, Massimo Vespiagnani, Luc Guyot, Ales Ude, and Auke Ijspeert. Rich periodic motor skills on humanoid robots riding the pedal racer.
- [13] HobbyKing. Turnigy tg9e 9g. URL: http://www.hobbyking.com/hobbyking/store/_9549_Turnigy_TG9e_9g_1_5kg_0_10sec_Eco_Micro_Servo.html [cited 15.10.2014].
- [14] Robotics INC. *Bioloid Premium Kit*.
- [15] Robotics INC. Gyro sensor. URL: http://support.robotis.com/en/product/auxdevice/sensor/dxl_gyro.htm [cited 15.10.2014].
- [16] Robotics INC. *ROBOTIS e-Manual v1.21.00*, 2006. URL: http://support.robotis.com/en/product/auxdevice/controller/cm510_manual.htm.
- [17] ROBOTICS INC. Darwin op, 2014. URL: <http://www.robotis.com/xe/darwinen> [cited 15.10.2014].
- [18] Robotis INC. Bioloid, 2014. URL: <http://www.robotis.com/xe/> [cited 15.10.2014].
- [19] Trossen Robotics LLC. arbotix robocontrollers. URL: <http://www.trossenrobotics.com/p/arbotix-robot-controller.aspx> [cited 15.10.2014].
- [20] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.

- [21] Group of volunteers. Processing 2, 2014. URL: <https://processing.org/> [cited 15.10.2014].
- [22] Erhan Oztop, DDavid W. Franklin, Thierry Chaminade, and Gordon Cheng. Human–humanoid interaction: Is a humanoid robot perceived as a human? *International Journal of Humanoid Robotics*, 2005.
- [23] Raspberry pi Fundation. What is a raspberry pi? URL: <http://www.raspberrypi.org/> [cited 15.10.2014].
- [24] Raspian. Welcome to raspian. URL: <http://www.raspbian.org/> [cited 16.1.2014].
- [25] Pierre Raufast. Opencv and pi camera board !, 2013. URL: <http://thinkrpi.wordpress.com/2013/05/22/opencv-and-camera-board-csi/> [cited 15.10.2014].
- [26] Jennifer Dos Reis, Juliana León, and Carolina Chang. Intregración de arbotix, raspberry pi y motores dynamixel ax-12+ con el objetivo de la construcción de un robot humanoide que busque y patee pelotas. 2014.
- [27] Robin R.Murphy. *Introduction to AI Robotics*. 2000.
- [28] Robotis. Lipo 11.1v battery set lbs-10. URL: <http://www.robotis.us/lipo-11-1v-battery-set-lbs-10/> [cited 15.10.2014].
- [29] Ros. About ros. URL: <http://www.ros.org/about-ros/> [cited 15.10.2014].
- [30] Ros-drivers. Rosserial. URL: <https://github.com/ros-drivers/rosserial> [cited 15.10.2014].
- [31] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. 1995.
- [32] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*. 2009.

- [33] 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.
- [34] OpenCV Developers Team. About. URL: <http://opencv.org/about.html> [cited 15.10.2014].
- [35] Makoto Tomimoto. Artificial sweat for humanoid finger. *Journal of Bionic Engineering*, 2014.
- [36] Emil Valkov. Raspberry pi camera with opencv, 2013. URL: <http://robidouille.wordpress.com/2013/10/19/raspberry-pi-camera-with-opencv/> [cited 15.10.2014].
- [37] Laboratorios Vanadium. Arbotix robocontroller. URL: <http://vanadiumlabs.github.io/arbotix/#hservo> [cited 15.10.2014].
- [38] Williams and Wang LLC. Dynamixel ax/mx 6 port extension hub. URL: <https://www.bananarobotics.com/shop/Dynamixel-AX-MX-6-Port-Extension-Hub> [cited 15.10.2014].

Glosario

AVR

Es una familia de microcontroladores de instrucciones reducidas de la compañía Atmel..
16, 29

BGR

(Red Green Blue / rojo, verde, azul): Es un modelo de color que se basa en la intensidad de los colores primarios de la luz (rojo, verde y azul).. 26

C++

Es un lenguaje de programación imperativo y orientado a objetos.. 25, 26

CSI

(Camera Serial Interface / Interfaz serial para cámaras): Es un estándar que define la interfaz de comunicación entre una cámara y un procesador. Es comúnmente utilizado en dispositivos móviles.. 22

framework

(Marco de trabajo) Es un conjunto de técnicas, conceptos y estilos de trabajo que se establecen para resolver un problema particular y que sirve de referencia para solucionar problemas similares.. 29

HDMI

(High-Definition Multimedia Interface/ interfaz multimedia de alta definición): Es una interfaz para transferir datos de audio y video digital entre un dispositivo y un monitor, proyector, televisor digital o dispositivo de audio digital.. 67

HSV

(Hue, Saturation, Value / Matiz, Saturación, Valor): Es un modelo de color que se basa en las cualidades de matiz, saturación y valor del color.. 26

IDE

(Integrated development environment / Entorno de desarrollo integrado): Es un programa diseñado para facilitar la programación en uno o varios lenguajes. Usualmente incluye herramientas de compilación, editor de textos y depurador.. 30, 65

LEGO

Es una serie de juguetes de construcción, que ofrece um kit de materiales para robótica llamada Mindstorms. La serie posee una trajeta programable, sensores y actuadores.
15

Licencia BSD

(Berkeley Software Distribution / distribución de software berkeley) : Es una licencia para software libre otorgada principalmente a sistemas BSD.. 29

MMAL

(Multi-Media Abstraction Layer / Capa de abstracción multimedia): Es una librería que brinda una interfaz de bajo nivel para controlar dispositivos que se ejecutan en el núcleo de video de la Raspberry Pi, como el módulo de cámara.. 18

Motor DC

Es un motor de corriente directa que transforma la energía eléctrica en mecánica generando un movimiento rotatorio. 66

Raspistill

Es una herramienta de captura de fotos por línea de comandos en la Raspberry Pi . 25

Raspivid

Es una herramienta de captura de video por medio de comandos en la Raspeberry Pi. 25

ROS

(Robot Operating System / Sistema de operación para robots): Es un framework que provee herramientas para ayudar a desarrolladores de aplicaciones para robots.. 29, 33

SD

(Secure Digital / Digital Seguro) : Es un formato de tarjetas de memoria de almacenamiento digital. Existen tarjetas SD con diferentes características en cuanto su clase, capacidad de almacenamiento y tamaño físico.. 67

TightVNC

Es un paquete de software que sirve para controlar la interfaz gráfica de ordenadores remotos.. 67

UART

(Universal Asynchronous Receiver/Transmitter): Es dispositivo para la comunicación estándar asíncrona. . 18

USB

(Universal Serial Bus): Es un bus estándar de conexión, comunicación y fuente de poder entre dispositivos electrónicos. 18, 23

V4L

(Video 4 Linux / video para linux): Es una interfaz de programación de video para Linux. Uno de los tipos de dispositivos soportados son las cámaras web USB.. 18

VGA

(Video Graphics Array): Es un conector de video estándar, una interfaz de video usada para proyectar video en alta definición . 18, 67

XBEE

Es una familia de módulos de radio, con protocolo de comunicación inalámbrica basado en radio frecuencias.. 16, 18, 19

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 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 [30], de lo contrario podrían ocurrir problemas de sincronización en la comunicación de las tarjetas.

Apéndice B

Carta Aceptación

La carta de aceptación del congreso de Reconocimiento de Patrones, Control Inteligente y Comunicaciones de la Universidad de Cuenca en Ecuador a realizarse en Diciembre 2014.

11/28/2014

Gmail - CIDI2014 notification for paper 73



juliana leon <julianaleon8@g

CIDI2014 notification for paper 73

CIDI2014 <cidi2014@easychair.org>

20 de octubre de 2

Para: Juliana Leon <julianaleon8@gmail.com>

Estimado/a Juliana Leon

Su trabajo 73, "Coordinación de Arbotix, Raspberry Pi y motores Dynamixel Ax-12+ con el objetivo de la construcción de un robot humanoide que busque y patee pelotas", ha sido revisado y ACEPTADO sujetos que se cumplan con las recomendaciones de los evaluadores. Este trabajo deberá ser presentado en una de las sesiones del Congreso de Ingeniería, Desarrollo e Innovación en el track Congreso de Reconocimiento de Patrones, Control Inteligente y Comunicaciones

La nueva versión del artículo debe ser enviada antes 26 de octubre del 2014, tomando en cuenta que necesariamente deben incluir la filiación de los autores. Por otro lado, es necesario considerar que los gráficos deben ser completamente claros, que ayuden a la comprensión del texto y preferiblemente realizados con el software Grapher. Los organizadores nos reservamos el derecho a solicitar que las imágenes y gráficos se repitan en caso de ser necesario.

En el link <http://ingenieria.ucuenca.edu.ec/aniversario/congreso/registro/>, se puede acceder al formulario de inscripción para realizar la inscripción y el pago. El pago anticipado con costo reducido se podrá realizar hasta el 30 de Octubre.

Muchas Gracias por haber enviado su trabajo a CIDI 2014, esperamos su presencia.

Cordialmente
Comisión Organizadora
CIDI 2014

----- REVIEW 1 -----

PAPER: 73

TITLE: Coordinación de Arbotix, Raspberry Pi y motores Dynamixel Ax-12+ con el objetivo de la construcción de un robot humanoide que busque y patee pelotas

AUTHORS: Juliana Leon, Jennifer Dos Reis and Carolina Chang

OVERALL EVALUATION: 2 (accept)

REVIEWER'S CONFIDENCE: 3 (medium)

----- REVIEW -----

El artículo presenta la interrelación entre tres elementos utilizados en robotica encargadoa de adquirir la imagen, reconocer la pelota y patearla, estos son un controlador, unos actuadores y un computador del tipo de una tarjeta de crédito. A fin de desarollar un humanoide que pueda ser parte del equipo de soccer de competencia de RobotCup.