UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Localización y seguimiento de objetos para un robot bípedo autónomo

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Andrés López Esquivel

DIRECTOR DE TESIS

Dr. Marco Antonio Negrete Villanueva

Ciudad Universitaria, Cd. Mx., 2024

Agradecimientos

Este trabajo se realizó con apoyo del proyecto PAPIIT IT102424: Modelos lógico probabilísticos para el desarrollo de robots móviles autónomos.

Este trabajo me permitió conocer, convivir y trabajar con personas humana y profesionalmente asombrosas, a las que les agradezco profundamente por su apoyo y contribuciones a mi camino personal y profesional. Particularmente, agradezco:

A mi director de tesis, el Dr. Marco Antonio Negrete, por guiar este trabajo con el que descubrí mi pasión (la robótica y el desarrollo de software) y por compartir sus conocimientos y siempre buscar el crecimiento profesional de sus estudiantes.

A los miembros del jurado de mi examen profesional, Dr. Jesús Savage Carmona, M.I. Gloria Mata Hernández, Dr. Saúl de la Rosa Nieves y M.I. Rubén Anaya García, por su valiosa retroalimentación a mi trabajo.

A los profesores M.C. Evert Josué Guajardo Benavides y M.I. Iván de Jesús Osio Chávez, por sus consejos y apoyo durante mi último año de estudiante en la Universidad.

A mis amigos del Laboratorio de Bio-robótica, Héctor, Ruth y Miguel. Siempre llevaré conmigo esas pláticas en las que nos alentamos a dar lo mejor en nuestros proyectos del Laboratorio.

A mis amigos de la Universidad, Yunué, Sergio, Omar, Braulio y Paula, con quienes compartí este hermoso pero retador viaje que representa estudiar ingeniería.

A Isabel, por acompañarme en todo momento, escucharme y enseñarme a cómo ser un mejor ser humano.

Y, finalmente, a mis padres, Luz María Esquivel Pérez y José Benito López Rivera, a mi hermano, Alejandro López Esquivel, y a mi abuela, Luz María Pérez Pérez. Son mi todo, este trabajo es para ustedes. Gracias por siempre alentarme a escoger y seguir mi camino. Sin ustedes, nada de esto estaría sucediendo.

Índice general

1. Introducción **7**
1.1. Motivación 7
1.2. Planteamiento del problema 8
1.3. Hipótesis 8
1.4. Objetivos 9
1.5. Descripción del documento 9

2. Antecedentes **11**
2.1. Robots bípedos autónomos 11
2.2. Visión artificial 13
2.3. Imágenes y color 14
2.4. Modelo de una cámara 16
2.5. Movimiento de cuerpo rígido 18
2.6. Trabajo relacionado 20

3. Detección y localización de objetos **25**
3.1. Corrección de distorsión 25
3.2. Segmentación por color 30
3.3. Localización visual 34

4. Estimación de velocidad **39**
4.1. Filtro de Kalman 39
4.2. Estimación de velocidad 41

5. Implementación **49**
5.1. El robot Nimbro-OP 49
5.2. Herramientas de visión computacional 51
5.3. Secuencia de pateo 52

5.4. La plataforma ROS 56

6. Resultados **61**
6.1. Estimación de posición 61
6.2. Estimación de velocidad 68
6.3. Pruebas de integración 71

7. Discusión **75**
7.1. Conclusiones 75
7.2. Trabajo futuro 76

Capítulo 1

Introducción

El impacto de los robots en la vida moderna incrementa rápidamente, desde aplicaciones en la industria de la manufactura hasta en los sectores de la salud, el transporte, la exploración espacial, entre otros. La robótica es un campo relativamente joven y, como lo indican Nenchev, Konno, y Tsujita (2018), su desarrollo se beneficia pero también contribuye a logros tecnológicos como la conducción autónoma (sensado, percepción y planeación de movimientos) o la comunicación a través del lenguaje natural (por ejemplo, para asistentes personales). Especialmente, por su capacidad para desempeñarse en ambientes humanos, los robots humanoides cobran cada vez más relevancia por sus aportes tecnológicos y sus posibles impactos sociales.

1.1. Motivación

Con el paso del tiempo, incrementa la relevancia de los robots humanoides para realizar tareas propias de los humanos, lo que se traduce en una continua búsqueda por desarrollar y mejorar sus capacidades de percepción, aprendizaje, planeación de tareas, locomoción, etc. Esta búsqueda ha motivado la creación de diversas iniciativas, como las competencias de robótica de la *RoboCup*, para fomentar el desarrollo continuo de la robótica y el intercambio de conocimientos entre la comunidad científica. Particularmente, para este trabajo son de interés las competencias celebradas en la categoría *Humanoid Leage* de la *RoboCup*, en la que robots humanoides se enfrentan a diversos retos relacionados con el *soccer* para evaluar y comparar los sistemas desarrollados por diferentes equipos en materia de humanoides. Así, este

trabajo se motiva y enfoca en desarrollar métodos con visión computacional que le permitan a un robot humanoide identificar y localizar objetos en movimiento, como lo podría ser un balón al que tiene que patear durante un juego de *soccer*.

1.2. Planteamiento del problema

El robot humanoide tipo *NimbRo-OP* del Laboratorio de Bio-robótica de la Facultad de Ingeniería de la UNAM, requiere de un sistema de visión computacional físicamente ligero y de bajo costo computacional (para ser implementado en el sistema embebido del robot, como se indica en la sección 5.1), que le permita identificar y localizar objetos en movimiento. Particularmente, con ayuda de dicho sistema, se busca que el humanoide sea capaz de patear una pelota en movimiento al identificarla (por ejemplo, mediante la segmentación de su color) y localizarla a través de la estimación de su posición y velocidad. De esta manera, el desarrollo de este sistema de visión sería un aporte para que, en un futuro, la Facultad de Ingeniería sea capaz de participar en competencias de robots humanoides como las celebradas en la categoría *Humanoid Leage* de la *RoboCup* (véase las secciones 2.1 y 2.2), en la que los robots juegan *soccer* para evaluar diversas de sus capacidades: caminar, correr, identificar y patear balones, etc.

1.3. Hipótesis

A partir de una imagen, se puede identificar un objeto al segmentarlo por su color y calcular su posición en el suelo con respecto a un marco de referencia fijo empleando relaciones entre los ángulos de visión de la cámara utilizada y la resolución de su imagen, transformaciones homogéneas y álgebra vectorial. Posteriormente, del conjunto de posiciones calculadas, se puede utilizar el Filtro de Kalman para estimar la posición y velocidad del objeto cuando esté en movimiento. Finalmente, suponiendo que el objeto es una pelota, se puede desarrollar un algoritmo que utilice las estimaciones del Filtro de Kalman para determinar el momento en el que un robot humanoide deba patearla.

1.4. Objetivos

Se busca desarrollar un sistema de visión computacional con el que el robot humanoide *NimbRo-OP* del Laboratorio de Bio-robótica de la Facultad de Ingeniería de la UNAM sea capaz de:

- Identificar una pelota al segmetarla por su color.
- Calcular la posición de la pelota con respecto a uno de sus pies.
- Utilizar el Filtro de Kalman para estimar tanto la posición como la velocidad de la pelota cuando ésta se encuentre en movimiento.
- Patear la pelota en el momento adecuado.

Se busca, además, que el sistema de visión computacional sea implemetado con la ayuda de herramientas computacionales como *ROS* (*Robot Operating System*), *OpenCV* y *MATLAB* y de lenguajes de programación como *Python*.

1.5. Descripción del documento

Este trabajo se estructura de la siguiente manera: en el capítulo 2 se dan a conocer los conceptos básicos utilizados a lo largo del documento (robots bípedos, visión artificial, imágenes, espacios de color, movimiento de cuerpo rígido, etc.) así como el trabajo relacionado. En el capítulo 3 se detallan los sistemas de detección y localización de objetos, donde se explican la segmentación de objetos por medio de su color, la corrección de la distorsión producida por la lente de una cámara y el cálculo de la posición de un objeto a partir de una imagen (se asume que el objeto siempre está en el suelo). El capítulo 4 se dedica a explicar el Filtro de Kalman y la manera en que se utiliza en el sistema de visión computacional propuesto para estimar la posición y velocidad de un objeto en movimiento. El capítulo 5 explica la implementación del sistema de visión computacional y detalla el algoritmo desarrollado para que el robot *NimbRo-OP* de la Facultad de Ingeniería de la UNAM patee una pelota en movimiento en el momento oportuno. En el capítulo 6 se dan a conocer los resultados de las pruebas de estimación de posición, estimación de velocidad y de secuencia pateo realizadas con el robot *NimbRo-OP*. Finalmente, en el capítulo 7 se dan a conocer las conclusiones de este trabajo y se plantea el trabajo futuro.

Capítulo 2

Antecedentes

En este capítulo se introducen los conceptos básicos que se utilizan a lo largo del documento y se explica la manera en que se relacionan con el sistema de visión computacional que se propone. En la sección 2.1 se introducen los conceptos de robot con extremidades, robot bípedo y robot humanoide, así como el relevante papel que desempeñan en competencias interacionales como la liga de humanoides de la *RoboCup*. En la sección 2.2 se introduce el concepto de visión computacional y los retos que enfrenta debido a las variaciones medioambientales. En la sección 2.3 se explican conceptos relacionados con imágenes digitales y espacios de color, tales como imágenes multiespectrales, imágenes binarias y los espacios de color RGB y HSV. En la sección 2.4 se explica el modelo de una cámara estenopeica y el proceso de formación de imágenes. En la sección 2.5 se dan a conocer los usos y propiedades de las transformaciones homogéneas para expresar la posición y orientación de un cuerpo rígido con respecto a otro. Finalmente, en la sección 2.6, se habla de trabajos relacionados como el que se plantea en este documento.

2.1. Robots bípedos autónomos

Un robot con extremidades se compone de piernas, brazos y un cuerpo, por lo que posee al menos una pierna para sostener e impulsar su estructura y puede tener un número arbitrario de brazos para manipular objetos. Si el robot posee únicamente dos piernas, se dice que es bípedo. Así, de acuerdo con Siciliano y Khatib (2016), el proceso para diseñar un robot con extremi-

dades implica diversos retos como el tipo de marcha (patrón de movimiento de las piernas para que el robot camine o corra), la biomímesis (imitación de la estructura mecánica de seres vivos), la dinámica bioinspirada (reproducción de la locomoción de seres vivos), la simplicidad mecánica (búsqueda de un diseño mecánico tan simple como sea posible), el espacio de trabajo de las extremidades (la posibilidad de que las extremidades alcancen una mayor o menor cantidad de posiciones y orientaciones de acuerdo con sus grados de libertad) y el soporte de carga (la apropiada asignación de las articulaciones para soportar el peso del cuerpo del robot).

Existen robots, conocidos como humanoides, que imitan aspectos de la forma y el comportamiento humanos. Usualmente, estos robots adquieren una apariencia humana (una cabeza, dos piernas, dos brazos, etc.), lo que lleva a que existan robots humanoides que son bípedos. Nenchev et al. (2018) indican que los humanoides se diseñan para operar de manera autónoma en diversos ambientes (por ejemplo, viviendas, oficinas, fábricas o zonas de desastres), desempeñar diversas tareas físicas, establecer comunicación con los humanos y operar y manipular herramientas y objetos diseñados para humanos. Estas tareas implican, desde un punto de vista técnico, que los humanoides tengan funciones de percepción y cognición, aprendizaje, planeación de tareas, locomoción y otras.

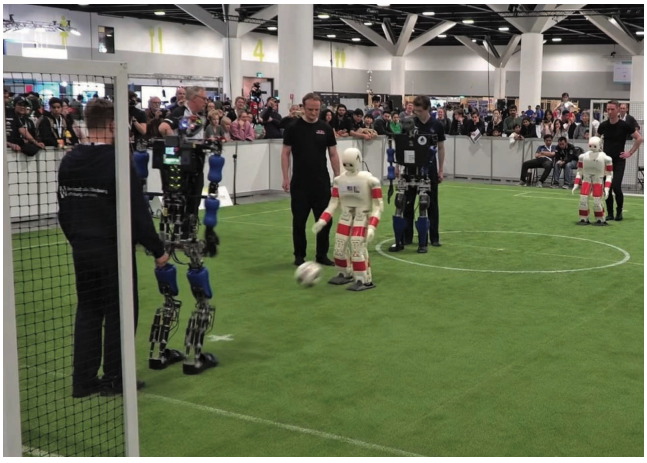

Figura 2.1: Competencia entre el equipo NimbRo (rojo) y el equipo Sweaty (azul) en la final de la categoría *AdultSize* de la *Humanoid League* de la *RoboCup* 2019 en Sydney. Imagen obtenida de Paetzel y Hofer (2019)

Los robots humanoides están presentes en cada entrega de la *RoboCup*, que organiza competencias para impulsar el desarrollo de la robótica y de la inteligencia artificial para cumplir el objetivo de integrar un equipo de robots capaces de enfrentar al campeón mundial de *soccer* del 2050 y de ganar el enfrentamiento [1]. De esta manera, en el año 2002, nació en la *RoboCup* la liga de humanoides (*Humanoid Leage*) para organizar competencias de *soccer* entre robots humanoides bípedos autónomos (véase la figura 2.1). En la liga de humanoides, los robots participantes se han enfrentado a retos como mantener el equilibrio en un solo pie durante una cierta cantidad de tiempo, caminar de manera estable, caminar y patear una pelota hacia una portería, o viceversa, ubicado en la portería, detener la pelota que pateó otro robot hacia ella (Zhou, 2004).

Los robots bípedos autónomos resultan de interés para esta tesis porque en ella se propone un sistema de visión computacional para un robot humanoide tipo Nimbro-OP (véase la sección 5.1) identifique y localice en el espacio una pelota en movimiento que debe patear (véase los capítulos 3 y 4).

2.2. Visión artificial

De acuerdo con Stockman y Shapiro (2001), la visión computacional consiste en tomar decisiones útiles acerca de objetos y escenarios físicos y reales con base en imágenes. Así, la visión computacional, a diferencia del procesamiento de imágenes, se preocupa por *entender* el contenido de las imágenes para construir los modelos o descripciones de los objetoss y escenarios contenidos en ellas y tomar decisiones.

La liga de humanoides (*Humanoid League*), de la competencia *RoboCup-Soccer*, es una de las varias aplicaciones en las que se utiliza visión computacional. En dicha competencia, robots humanoides juegan *soccer* unos contra otros (véase la figura 2.2), por lo que los robots ejecutan tareas como percibir visualmente el balón, a otros jugadores y el campo de juego.

[1] https://www.robocup.org/objective

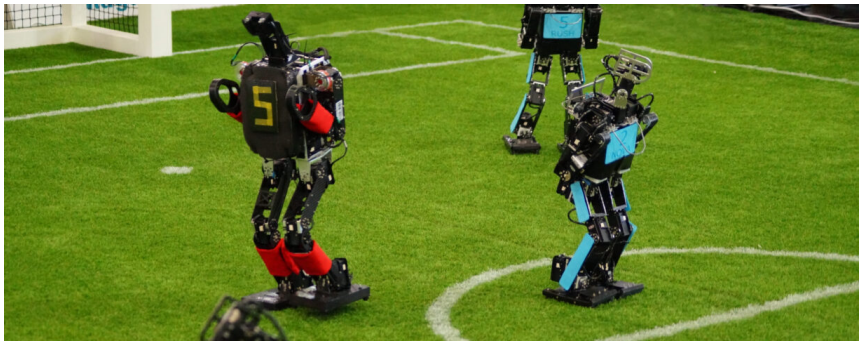

Figura 2.2: *Humanoid League* de la competencia *RoboCupSoccer*. Imagen obtenida de la página oficial de la *Humanoid League* [3].

Aunque la visión computacional se utiliza en diversas aplicaciones, el desarrollo de soluciones mediante su uso puede ser complicado y retador, sobre todo porque el ambiente produce variaciones (exorbitantes, en demasiadas ocasiones) en las imágenes capturadas, lo que lleva a que frecuentemente se acepten restricciones ambientales (como el control de iluminación) que comprometen la flexibilidad de los sistemas de visión computacional (Stockman y Shapiro, 2001).

En esta tesis se utiliza la visión computacional para identificar y localizar una pelota en el espacio (consúltese el capítulo 3), sin embargo, debido a que la identificación de la pelota se realiza mediante la segmentación de su color, su apariencia puede varíar significativamente debido a cambios en la iluminación o ante la presencia de otros objetos del mismo color.

2.3. Imágenes y color

Las imágenes digitales son matrices bidimensionales de valores discretos de intensidad de luz. De acuerdo con Stockman y Shapiro (2001), algunos tipos de imágenes digitales son:

- **Imágenes en escala de grises**: Son monocromáticas y cada elemento de sus matrices posee únicamente un valor de intensidad.

[3] https://humanoid.robocup.org/

- **Imágenes multiespectrales**: Cada elemento (pixel) de sus matrices es un vector de valores de intensidad. Si la imagen es de color, entonces el vector se constituye por tres valores.
- **Imágenes binarias**: Es una imagen en la que cada pixel adquiere un valor de 0 o 1.

Cuando se habla acerca de la **resolución** del dispositivo que captura una escena (como una cámara), en lugar de hablar acerca de la **resolución nominal** de su sensor (a qué porción de la escena corresponde un pixel) se habla de la cantidad de pixeles que constituyen a la imagen que produce, lo que resulta útil para conocer en cuántos pixeles se puede dividir el **campo de visión** (*FOV*, *Fielf Of View*) de su sensor, que no es más que el tamaño de la escena que puede sensar y que usualmente se expresa angularmente (**campo de visión angular**).

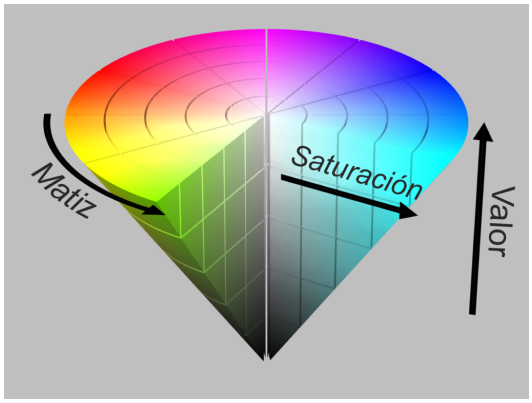

Figura 2.3: Espacio de color HSV [4].

Así, después de haber definido algunos conceptos relacionados con las imágenes digitales, es importante introducir los espacios de color **RGB** y **HSV**, que se utilizan para determinar los tres valores que almacena cada pixel de una imagen de color para representar colores.

El espacio de color RGB (*red-green-blue*) representa colores a partir de la combinación de los colores rojo, verde y azul, donde a cada uno de dichos

[4]https://es.wikipedia.org/wiki/Modelo_de_color_HSV

colores primarios se les asigna un byte, por lo que sus valores de intensidad van desde 0 hasta 255, de tal manera que, si se quisiera representar el color amarillo, su código RGB sería (255, 255, 0), donde el primer valor corresponde al rojo, el segundo al verde y el tercero al azul.

Por otra parte, el espacio HSV (véase la figura 2.3) representa colores mediante el uso de tres valores: matiz (*Hue*), saturación (*Saturation*) y valor (*Value*). El matiz se define como un ángulo, generalmente entre 0 y 2π, que representa los colores (por ejemplo, rojo con $0\,[rad]$, verde con $2\pi/3\,[rad]$ y azul con $4\pi/3\,[rad]$), la saturación representa qué tan colorido es un estímulo y el valor la intensidad del color.

Debido a que en esta tesis se identifica y localiza un objeto mediante la segmentación de su color (véase el capítulo 3), son de particular interés las imágenes de color, las imágenes binarias, la conversión de valores desde RGB hacia HSV y los conceptos de resolución y campo de visión.

2.4. Modelo de una cámara

El modelo de la cámara estenopeica (*pinhole camera*) permite entender la formación/proyección de imágenes cuando, por ejemplo, el sensor de una cámara (o la retina de un humano) recibe los rayos de luz que se reflejaron en un objeto (parte de la luz que viaja y choca con un objeto se absorbe, mientras que el resto se refleja).

El modelo, que no contempla el uso de lentes, consiste en una pared con un pequeño agujero en el centro que permite únicamente el paso de los rayos de luz que logran atravesar el agujero, por lo que una cámara real de este tipo no es una buena opción en situaciones de exposión rápida debido a su incapacidad para reunir grandes cantidades de luz (Bradski y Kaehler, 2008). Por lo tanto, la adición de lentes a las cámaras (como normalmente se hace) ayuda a recolectar una mayor cantidad de luz, aunque también introduce distorsiones que se deben corregir. La calibración de una cámara para corregir las distorsiones introducidas se aborda en la sección 3.1 y resulta releventante para esta tesis porque permitirá, posteriormente, establecer la relación entre las unidades en pixeles de la imagen que captura una cámara con unidades del espacio físico (véase la sección 3.3).

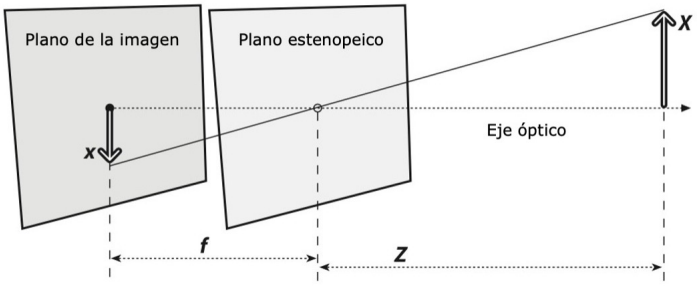

Figura 2.4: Modelo de una cámara estenopeica. Imagen modificada de Bradski y Kaehler (2008)

En el modelo de la cámara estenopeica de la figura 2.4 se muestra que el rayo de luz que proviene de un objeto atraviesa el pequeño agujero anteriormente mencionado y proyecta una imagen invertida del objeto en el llamado plano de la imagen. En la figura, f es la longitud focal de la cámara, Z la distancia de la cámara al objeto, X la longitud del objeto y x la longitud de la imagen del objeto. Así, al relacionar los triángulos resultantes, se tiene que

$$-\frac{x}{f} = \frac{X}{Z} \rightarrow \boxed{-x = f\frac{X}{Z}} \tag{2.1}$$

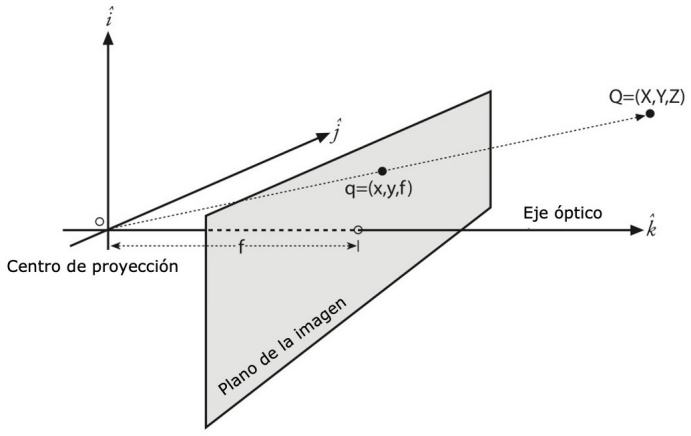

Figura 2.5: Abstracción del modelo de la cámara estenopeica equivalente a la de la figura 2.4. Imagen modificada de Bradski y Kaehler (2008)

La figura 2.5 es una abstracción del modelo de la cámara estenopeica equivalente a la de la figura 2.4 pero en la que no se invierte la imagen del objeto (Bradski y Kaehler, 2008), por lo que, para este caso, la ecuación 2.1 carecería del signo negativo. En el diagrama, Q es un punto en el espacio físico y q su proyección en el plano de la imagen, donde el rayo de luz viaja desde Q hacia el centro de proyección (equivalente al pequeño agujero del diagrama de la figura 2.4). No obstante, debido a que la intersección entre el eje óptico y el plano de la imagen no coincide con el centro del sensor de la cámara, se introducen los parámetros c_x y c_y para modelar el posible desplazamiento. También, se introducen dos longitudes focales, f_x y f_y, para considerar que los pixeles de las cámaras de bajo costo son rectangulares y no cuadrangulares. Así, la ubicación (x, y) de un pixel en el que se proyecta $Q = (X, Y, Z)$ es

$$x = f_x \left(\frac{X}{Z}\right) + c_x, \qquad y = f_y \left(\frac{Y}{Z}\right) + c_y \tag{2.2}$$

2.5. Movimiento de cuerpo rígido

Un cuerpo rígido es la idealización de un cuerpo que no sufre ninguna deformación independientemente de las fuerzas que se le apliquen, por lo que la distancia entre dos puntos cualesquiera de su estructura siempre es constante. De esta manera, aunque se trata de una idealización, se puede considerar a un cuerpo como rígido si su deformación es despreciable. En la mayoría de las ocasiones, los robots son un conjunto de cuerpos rígidos articulados de tal forma que puedan desempeñar movimientos complejos (Olguín Díaz, 2019).

Para describir los movimientos de un robot se establecen a lo largo de su estructura varios marcos de referencia para representar las posiciones y las orientaciones de los cuerpos rígidos que lo constituyen. Es por ello por lo que, para describir la orientación y la posición de un marco de referencia con respecto a otro, se utilizan las transformaciones homogéneas, que combinan en una sola matriz las operaciones de rotación y traslación y que permiten representar un conjunto de coordenadas en diferentes marcos de referencia (W. Spong, Hutchinson, y Mathukumalli, 2005).

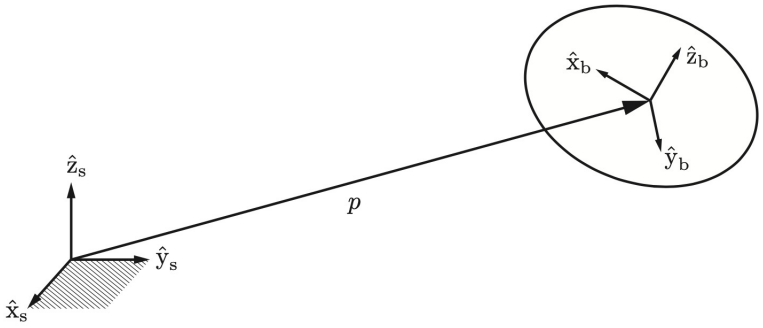

Figura 2.6: Cuerpo rígido en el espacio tridimensional. Imagen obtenida de M. Lynch y C. Park (2017)

En la figura 2.6 se muestran un marco de referencia **s** fijo y un marco de referencia **b** atado a un cuerpo rígido. Si se consideran dichos marcos de referencia, una transformación homogénea T_{sb} podría representar la orientación y traslación de **b** con respecto a **s** mediante R y p, respectivamente, donde $R \in SO(3)$ es una matriz de rotación y p un vector columna que va desde el origen de **s** hasta el origen de **b**. De manera general, una transformación homogénea T es una matriz de la forma

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{2.3}$$

donde $R \in SO(3)$, $p \in \mathbb{R}^3$ y 0 representa un vector fila de tres ceros.

Algunas de las propiedades de las transformaciones homogéneas son:

- La inversa de una transformación homogénea también es una transformación homogénea y tiene la siguiente forma:

$$T^{-1} = \begin{bmatrix} R^T & -R^T p \\ 0 & 1 \end{bmatrix} \tag{2.4}$$

- El producto de dos transformaciones homogéneas también es una transformación homogénea.
- La multiplicación de transformaciones homogéneas es asociativa, de tal manera que $(T_1T_2)T_3 = T_1(T_2T_3)$.

Los tres usos principales (M. Lynch y C. Park, 2017) de las transformaciones homogéneas son:

- Representar la configuración (posición y orientación) de un cuerpo rígido.
- Cambiar el marco de referencia en el que un vector está representado.
- Desplazar (rotar o trasladar) un vector o marco de referencia.

En particular, para esta tesis es de interés el segundo de los usos: cambiar el marco de referencia en el que un vector está representado. Suponga los marcos de referencia **a**, **b** y **c**, y un vector v_b expresado en **b**. Si se desea conocer el mismo vector pero expresado en **a**, es decir, v_a, se realiza lo siguiente:

$$T_{ac}T_{cb} = T_{a\not{c}}T_{\not{c}b} = T_{ab}$$

$$v_a = T_{ab}v_b$$

donde T_{ab} es la transformación homogénea de **b** hacia **a** y T_{bc} de **c** hacia **b**. Además, se observa que se aplica la cancelación de índices al realizar la multiplicación $T_{ab}T_{bc}$.

2.6. Trabajo relacionado

En las competencias de *soccer* de la RoboCup, es necesario que los robots participantes identifiquen y localicen el balón, y es por ello por lo que, antes de introducir el sistema de visión computacional que propone esta tesis para dicho fin, se introducen, de manera general, distintos enfoques que han utilizado algunos equipos de la RoboCup para la detección y localización de objetos.

Sensado cooperativo en la *Middle Size League* de la RoboCup

Conforme avanzan y progresan las competencias de *soccer* en la liga de medio tamaño (*Middle Size League*, *MSL*) de la RoboCup, incrementa el tiempo en el que el balón está en el aire, por lo que Kuijpers, Neves, y

van de Molengraft (2017) proponen un método para calcular la posición del balón mediante la triangulación de los datos procesados por cada robot que forma parte de un equipo (se habla, por lo tanto, de un sensado cooperativo).

Usualmente, los robots de los equipos de la MSL proyectan el balón en el campo de juego para obtener su posición, sin embargo, este método conduce a una falsa proyección cuando el balón está en el aire (véase la figura 2.7).

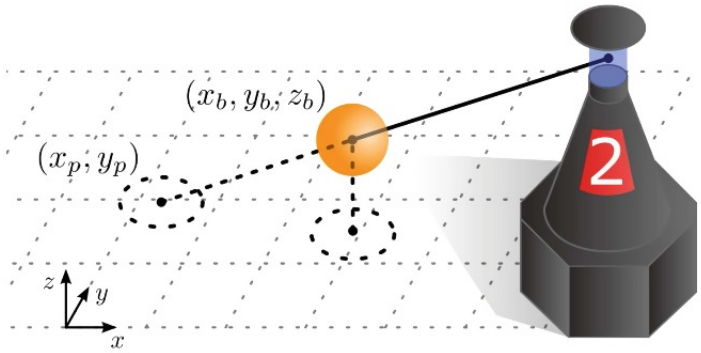

Figura 2.7: Un robot de la *Middle Size League* detecta un balón en el aire con posición (x_b, y_b, z_b) y crea una falsa proyección en (x_p, y_p), donde $x_p \neq x_b$ y $y_p \neq y_b$. Imagen obtenida de Kuijpers et al. (2017).

En el método propuesto por Kuijpers et al. (2017), cada robot del equipo somete la imagen que captura a un proceso de segmentación con base en colores. Así, cada figura que resulta del proceso de segmentación es clasificada de acuerdo con la probabilidad de que sea el balón (probabilidad calculada con base en el color del balón, su tamaño y forma). De esta manera, los robots del equipo se comparten los datos necesarios para calcular las rectas con las que determinarían la posición del balón, por lo que cada robot cuenta con sus datos y los de sus compañeros para calcular la intersección (triangulación) entre las diversas rectas y determinar la posición del balón. Por lo tanto, el algoritmo requiere de los datos de por lo menos dos robots, pero si existe una mayor cantidad disponible, en lugar de calcular la intersección entre las N rectas (con $N > 2$), se calculan varias intersecciones entre pares de rectas y se promedian. El valor promedio calculado, que corresponde a la posición del balón, se filtra mediante el uso del filtro de Kalman.

No obstante, este método enfrenta algunos retos. En un caso ideal, se podría calcular la posición del balón mediante la intersección de las rectas, pero debido a inexactitudes que podrían afectar su obtención, podría no existir la intersección entre ellas. Para contemplar esta situación, el algoritmo busca para cada dos rectas los puntos que minimizan la distancia entre ellas y entre los que se encontraría la posición del balón. Otra dificultad enfrentada por el método se relaciona con los retardos derivados del intercambio de datos entre los robots. Si el algoritmo no considera estos retardos, se podría estar calculando la triangulación entre rectas de diferentes instantes de tiempo. Por ello, cada robot almacena sus datos y los de sus compañeros a lo largo del tiempo con el objetivo de acceder a datos pasados y triangularlos correctamente. Un último reto consiste en la falta de datos de un robot y/o de sus compañeros (podría suceder, por ejemplo, que no detectaron el balón porque estaba fuera de sus campos de visión). Si no existe la cantidad suficiente de datos, el algoritmo utiliza datos del pasado y recurre al modelo del balón para estimar su posición y velocidad. No obstante, este enfoque es débil ante las perturbaciones no modeladas.

El sistema de visión computacional desarrollado en esta tesis para un robot humanoide, identifica una pelota en movimiento, calcula su posición al proyectarla y utiliza el filtro de Kalman para estimar su velocidad (consúltese los capítulos 3 y 4). Tal y como lo explican Kuijpers et al. (2017), se genera una falsa proyección cuando la pelota está en el aire (véase la figura 2.7), no obstante, el alcance de esta tesis considera al piso como el único espacio de movimiento de la pelota. De la misma manera, se reconoce que el modelo de la pelota propuesto en el capítulo 4 es débil ante perturbaciones no modeladas. Aunque esta tesis trabaja con un solo robot humanoide, el enfoque de sensado cooperativo puede llegar a ser de utilidad en trabajos futuros.

Equipo *NimbRo* en la RoboCup 2018

Durante la RoboCup 2018, el método de percepción visual que utilizó el equipo NimbRo en la liga *Adult Size* (Farazi et al., 2019) emplea una red neuronal convolucional con una arquitectura similar a la de los modelos SegNet y U-Net (para segmentación de imagénes) y una etapa post-procesamiento para detectar objetos como porterías, balones y otros humanoides. El equipo entrena inicialmente el modelo con imágenes pequeñas e incrementa gradualmente su tamaño.

Los objetos identificados mediante la red neural se filtran y proyectan en un sistema de coordenadas egocéntrico. Para minimizar los errores de proyección, el equipo calibra los parámetros de la cámara del robot mediante el algoritmo Nelder-Mead.

En una de las pruebas de la RoboCup 2018 (*Goal Kick from Moving Ball*), el robot NimbRo-OP2X del equipo NimbRo logró patear un balón en movimiento y lo dirigió exitosamente hacia una portería (véase la secuencia de imágenes de la figura 2.8). El enfoque que utilizó el equipo para lograrlo consiste en detectar el balón, calcular su posición, estimar su velocidad y calcular el tiempo que le tomará llegar a una área en la que es probable que se ejecute un pateo exitoso. El robot, antes de realizar lo anterior, levanta la pierna con la que pateará y se sostiene con la otra, lo que agrega una desventaja: la postura del robot en posición de pateo es inestable.

Aunque en esta tesis no se utilizan métodos de aprendizaje profundo para detectar objetos, sí se utiliza un enfoque similar para patear una pelota en movimiento.

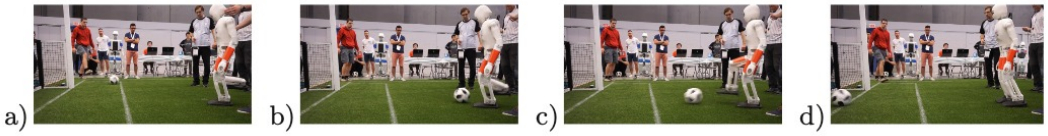

Figura 2.8: Robot del equipo NimbRo pateando una pelota en movimiento. (a) Configuración inicial (pie levantado listo para patear). Un humano le pasa el balón al robot. (b) El pie comienza a moverse para patear el balón. (c) El robot logra patear el balón. (d) El robot logra anotar en la portería y recupera su postura. Secuencia de imágenes obtenida de Farazi et al. (2019)

Sistema de visión para estimar posición y velocidad de objetos para un robot bípedo

Esta tesis representa la continuación de aquella que escribió Luis Nava, titulada *Sistema de visión para estimar posición y velocidad de objetos para un robot bípedo* (González Nava, 2021), en la que Luis propone un sistema

de visión computacional para identificar y localizar un balón en movimiento para un robot tipo Nimbro-OP en un ambiente simulado de Gazebo. En su tesis, Luis identifica el balón mediante la segmentación de su color (con ayuda de funciones de OpenCV), lo proyecta en el campo de juego para calcular su posición y utiliza el filtro de Kalman para estimar su velocidad.

Debido a que Luis realizó sus pruebas en un ambiente simulado, no se introdujeron distorsiones a la imagen que produce la cámara del robot. En esta tesis, como continuación del trabajo de Luis, se desarrolla un sistema de visión computacional para los mismos fines pero para ser implementado en el robot real tipo Nimbro-OP del laboratorio de Bio-robótica de la Facultad de Ingeniería de la UNAM, donde la cámara que utilizará, a diferencia de la que utilizó Luis, es una tipo ojo de pescado, que amplía su campo de visión pero que introduce distorsiones a la imagen que produce, por lo que debe ser calibrada (véase la sección 3.1).

Al igual que en la tesis de Luis, esta tesis se apoya en el uso de herramientas de OpenCV para identificar la pelota mediante la segmentación de su color (véase la sección 3.2), aunque, adicionalmente, aquí se utilizan funciones de MATLAB para calcular el centro de la pelota en la imagen corregida (entiéndase por imagen corregida a aquella a la que se le corrigieron las distorsiones que introduce la lente ojo de pescado de la cámara). El centro de la pelota, en píxeles, se utiliza para proyectarla en el piso ($z = 0$) y calcular su posición con respecto a uno de los pies del robot. La posición de la pelota, en ambas tesis, se obtiene de una recta que pasa por su centro, no obstante, aquí se aborda un enfoque distinto al crear un nuevo marco de referencia rotado pero no trasladado con respecto al marco de referencia de la cámara cuyo eje x apunte, en todo momento, hacia el centro de la pelota, lo que es de utilidad para calcular dicha recta (véase la sección 3.3). Para la estimación de velocidad, se continúa utilizando el filtro de Kalman (véase el capítulo 4).

Finalmente, en esta tesis se desarrollan nuevamente los nodos y servicios de ROS (véase el capítulo 5) encargados de identificar y localizar la pelota, así como de estimar su velocidad para calcular en qué momento patearla.

Capítulo 3

Detección y localización de objetos

En este capítulo se detalla la manera en que el sistema de visión computacional del robot identifica y localiza a un objeto (en este caso, a la pelota). En la sección 3.1 se explican los tipos de distorsión que se derivan de la lente de la cámara y que afectan a las imágenes capturadas, por lo que también se aborda el enfoque utilizado para corregirlos. En la sección 3.2 se da a conocer el procedimiento para identificar a la pelota mediante la segmentación de su color. Finalmente, en la sección 3.3 se detalla el algoritmo para calcular la posición del centro de la pelota con respecto a uno de los pies del robot.

3.1. Corrección de distorsión

El desarrollo del sistema de visión computacional que utiliza el robot de esta tesis para identificar objetos (en este caso, una pelota), se ve afectado por las distorsiones adyacentes a la lente tipo ojo de pescado de la cámara del robot (tal y como se explica en la sección 2.4). Aunque existen varios tipos de distorsión, Bradski y Kaehler (2008) indican que las principales son la radial y la tangencial. La distorsión radial, ocasionada por la forma de las lentes, incrementa conforme los puntos se alejan del centro de la imagen y hace que las líneas rectas parezcan curvas (la imagen de la figura 3.1 es un ejemplo). Por otra parte, la distorsión tangencial surge porque no se logran alinear perfectamente la lente con el plano de la imagen durante el proceso de ensamblado de la cámara, haciendo parecer a algunos puntos más cercanos o

alejados de lo esperado. Para corregir ambas distorsiones, se realiza previamente el proceso de calibración de la cámara: obtención de sus parámetros intrínsecos y extrínsecos así como de sus coeficientes de distorsión.

Los parámetros intrínsecos son propios de la cámara y forman la llamada matriz de la cámara, M:

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

donde f_x[px] y f_y[px] son las distancias focales y c_x[px] y c_y[px] los centros ópticos (para mayor detalle, consulte la ecuación 2.2). Por otro lado, los parámetros extrínsecos representan la ubicación (posición y orientación) de la cámara en el espacio tridimensional y permiten expresar puntos en el espacio con respecto al marco de referencia de la cámara.

Con respecto a los coeficientes de distorsión, se revisan primero los modelos matemáticos de ambos tipos de distorsión, donde las ecuaciones que modelan a la distorsión radial son:

$$x_d = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.2}$$

$$y_d = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{3.3}$$

donde (x_d, y_d) representa a los pixeles en la imagen distorsionada, (x, y) a los pixeles (normalizados) en la imagen corregida y $r = x^2 + y^2$. Por pixeles normalizados se entiende trasladados de acuerdo con el centro óptico y divididos por la distancia focal. Asimismo, las ecuaciones que modelan a la distorsión tangencial son:

$$x_d = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{3.4}$$

$$y_d = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \tag{3.5}$$

De este modo, con base en los coeficientes de los modelos, se construye el vector de coficientes de distorsión, D:

$$D = \begin{bmatrix} k_1 & k_2 & p_1 & p_2 & k_3 \end{bmatrix} \tag{3.6}$$

Las herramientas de *OpenCV* (véase la sección 5.2) que utiliza esta tesis para calibrar la cámara del robot, se basan en los modelos mencionados y requieren de varias imágenes en las que se muestre, en distintas posiciones y orientaciones, una estructura definida con demasiados puntos identificables. Aunque no es el único patrón, es usual utilizar un tablero de ajedrez como el de la figura 3.1 para calcular los parámetros intrínsecos, extrínsecos y los coeficientes de distorsión. De cada imagen tomada al tablero, las funciones de *OpenCV* extraen un conjunto de puntos en el espacio así como sus correspondientes puntos en la imagen. Para obtener los puntos en el espacio, se asume que el tablero se ubica en el plano xy ($z = 0$) y que la cámara es quien se mueve. De este modo, se conocen las posiciones de cada esquina de los cuadrados, para lo que resulta útil conocer el tamaño de cada cuadrado del tablero. Así, se utilizan dichos puntos identificados para calibrar la cámara.

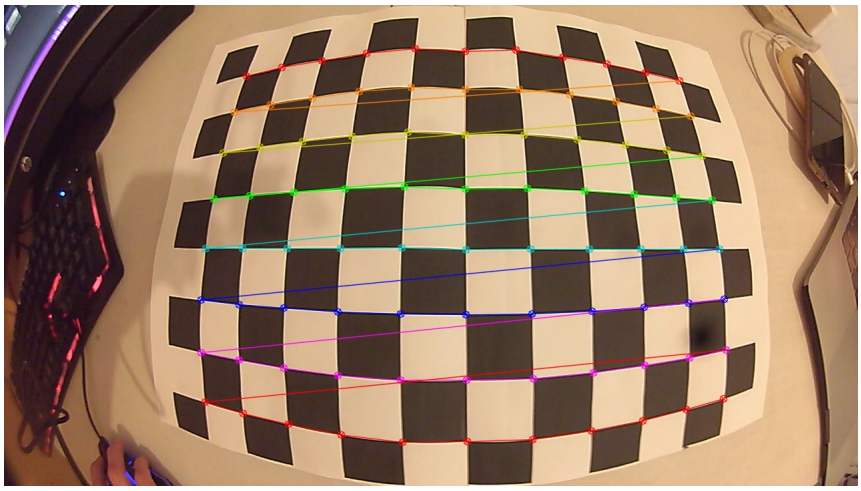

Figura 3.1: Tablero de ajedrez utilizado para calibrar la cámara del robot. En él, *OpenCV* identifica y dibuja los puntos que utilizará durante el proceso de calibración.

Aunque los parámetros y coeficientes calculados se pueden utilizar para corregir la distorsión de una imagen completa, en este trabajo se utilizan para corregir la distorsión de un punto de interés y evitar la ejecución de procesos computacionalmente costosos. Cuando se identifica a la pelota mediante la segmentación de su color (de acuerdo con lo explicado en la sección 3.2), se calcula su centroide en la imagen distorsionada y se corrige para obtener su equivalente en la imagen sin distorsión. Si el centroide no se corrige,

las mediciones de posición emitidas por el algoritmo de la sección 3.3 serían incorrectas.

Para obtener el equivalente corregido de un punto afectado por los distintos tipos de distorsión, se podría recurrir a los modelos de las distorsiones radial y tangencial, sustituir los parámetros y coeficientes calculados durante la calibración de la cámara y utilizar métodos numéricos. No obstante, el enfoque utilizado en este trabajo consiste en calibrar la cámara con ayuda de *OpenCV* y recurrir a una función de *MATLAB* para corregir la distorsión del centroide de la pelota (la función procesa los parámetros intrínsecos y extrínsecos, los coeficientes de distorsión y el centroide). En el capítulo 5 se profundiza acerca de la implementación de estas etapas.

En la figura 3.2 se grafican los puntos de las periferias de las imágenes con y sin distorsión. La distorsión radial incrementa conforme los puntos se alejan del centro de la imagen, lo que ocasiona la particular forma que tiene la periferia de la imagen no distorsionada. De esta imagen, se concluye que los polinomios de las ecuaciones que modelan a la distorsión radial tienden a la unidad cuando los puntos son cercanos al centro óptico (es decir, en esa zona la distorsión radial es despreciable), pero disminuyen conforme los puntos se alejan.

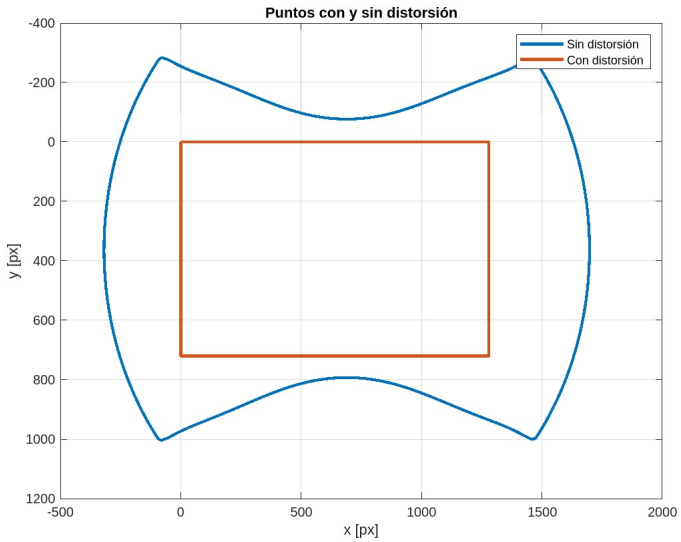

Figura 3.2: Puntos de las periferias de las imágenes con y sin distorsión.

Así, para ejemplificar todo lo mencionado en esta sección, a continuación se da a conocer una serie de imágenes tanto capturadas como procesadas por el sistema de visión computacional del robot. En la figura 3.3 se muestra la imagen distorsionada en la que se identifica el centroide de la pelota mediante un punto rojo. Por otra lado, en la figura 3.4 se observa una versión recortada de la imagen corregida (es decir, no distorsionada) en la que también se aprecia el centroide corregido de la pelota mediante un punto rojo. Por último, la imagen de la figura 3.5 corresponde a una versión no recortada de la imagen corregida (su contorno es parecido a la gráfica que forman los puntos corregidos en la figura 3.2).

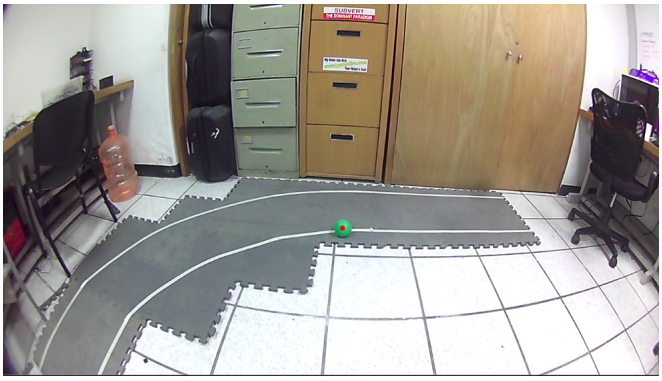

Figura 3.3: Imagen distorsionada en la que se dibuja el centroide de la pelota.

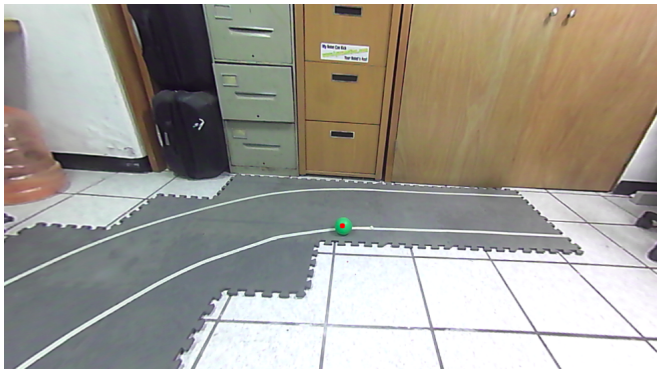

Figura 3.4: Imagen corregida y recortada en la que se dibuja el centroide de la pelota.

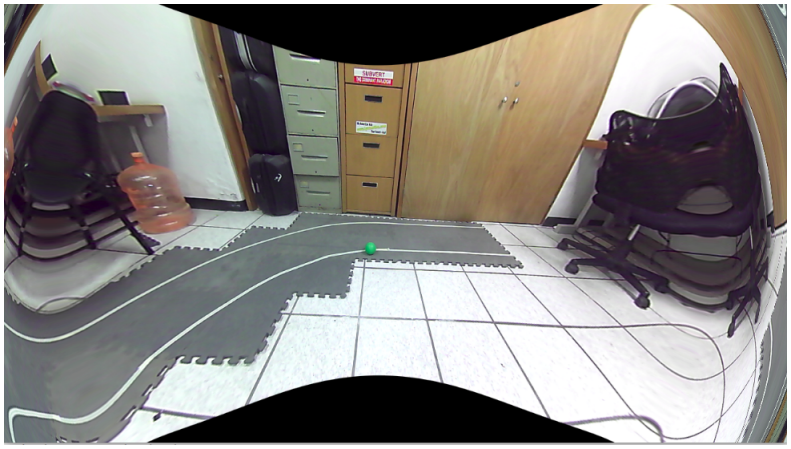

Figura 3.5: Imagen corregida y no recortada.

Así, esta sección resulta relevante para el resto de la tesis porque provee los mecanismos para corregir la distorsión inherente a la imagen capturada por la cámara del robot. Particularmente, provee la vía para corregir el centroide del objeto identificado por el algoritmo de la sección 3.2, lo que permite, a su vez, calcular correctamente la posición del objeto (la pelota) de acuerdo con el algoritmo presentado en la sección 3.3.

3.2. Segmentación por color

El sistema de visión computacional propuesto identifica la pelota mediante su color. Para lograrlo, transforma el espacio de color RGB de la imagen capturada por la cámara al espacio de color HSV (véase la sección 2.3), encuentra los píxeles que forman parte de un rango de valores HSV previamente indicado, genera una imagen binaria en la que los píxeles con valor de 1 cumplen con el rango de colores, elimina el ruido de la imagen mediante el uso de operadores morfológicos y calcula el centroide (en píxeles) del área identificada. Así, este conjunto de pasos segmenta al objeto por su color y emite su centroide para calcular la posición de la pelota (véase la sección 3.3). Un ejemplo de la pelota segmentada por su color se da en la imagen binaria de la figura 3.6.

De acuerdo con Stockman y Shapiro (2001), la conversión de RGB a HSV puede llevarse a cabo a través del siguiente algoritmo:

$$V := max(R, G, B) \tag{3.7}$$

$$S := \begin{cases} \frac{V - min(R,G,B)}{V} & V \neq 0 \\ 0 & \text{otro caso} \end{cases} \tag{3.8}$$

$$H := \begin{cases} \frac{\pi}{3} \cdot \frac{G-B}{V-min(R,G,B)} & V = R \\ \frac{2\pi}{3} + \frac{\pi}{3} \cdot \frac{B-R}{V-min(R,G,B)} & V = G \\ \frac{4\pi}{3} + \frac{\pi}{3} \cdot \frac{R-G}{V-min(R,G,B)} & V = B \\ 0 & R = G = B \end{cases} \tag{3.9}$$

donde se considera que los valores R, G y B pueden estar en $[0, 1]$ o $[0, 255]$.En cuanto a las salidas, H toma el mismo rango de valores que las entradas, $S \in [0, 1]$ y $V \in [0, 2\pi]$. Si $H < 0$, entonces $H = H + 2\pi$.

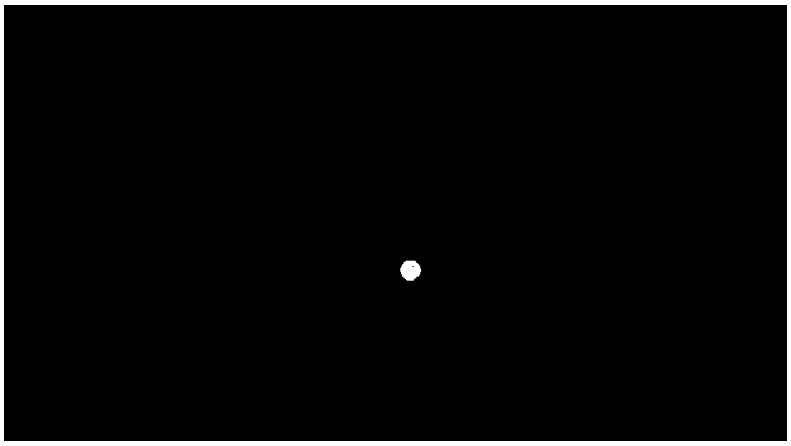

Figura 3.6: Pelota segmentada por su color.

De este modo, una vez realizada la transformación de RGB a HSV, se define el criterio para seleccionar los píxeles de interés. Para ello, se definen tres vectores:

$$HSV_{px} = \begin{bmatrix} H_{px} & S_{px} & V_{px} \end{bmatrix} \tag{3.10}$$

$$HSV_{inf} = \begin{bmatrix} H_{inf} & S_{inf} & V_{inf} \end{bmatrix} \tag{3.11}$$

$$HSV_{sup} = \begin{bmatrix} H_{sup} & S_{sup} & V_{sup} \end{bmatrix} \tag{3.12}$$

donde HSV_{px} representa el color de cualquier píxel de la imagen, HSV_{inf} el límite inferior del criterio de selección y HSV_{sup} el límite superior. Los valores de HSV_{inf} y HSV_{sup} se obtienen experimentalmente y dependen del color del objeto a identificar. Así, un píxel es seleccionado si su color HSV_{px} cumple con:

$$H_{inf} \leq H_{px} \leq H_{sup} \tag{3.13}$$

$$S_{inf} \leq S_{px} \leq S_{sup} \tag{3.14}$$

$$V_{inf} \leq V_{px} \leq V_{sup} \tag{3.15}$$

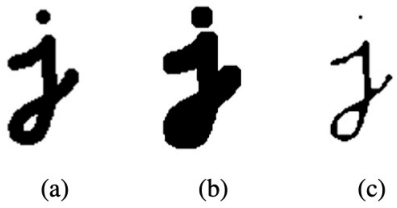

Figura 3.7: Ejemplo de aplicación de los operadores morfológicos: (a) imagen original, (b) imagen dilatada e (c) imagen erosionada. Imagen obtenida de Szeliski (2022).

Aunque el paso anterior produce una imagen binaria en la que se aprecia la forma de la pelota (véase la figura 3.8), existe ruido circundante que deforma el área identificada y afecta la obtención del centroide, por lo que en esta situación, los operadores morfológicos de erosión y dilatación son útiles para eliminar el ruido o cualquier otra imperfección. Estos operadores modifican la forma del objeto identificado y necesitan dos entradas: la imagen binaria a corregir y un kernel (también conocido como elemento estructurante), donde el kernel puede adquirir diversas formas (desde una matriz de 3×3 hasta estructuras complejas). Para ejecutar una operación morfológica, se calcula la convolución entre la imagen binaria y el kernel y se emite una salida (0 o 1) de acuerdo con el tipo de operación utilizado (Szeliski, 2022), donde

la erosión reduce el área a modificar mientras que la dilatación la acrecenta (véase el ejemplo de la figura 3.7). De manera general, conforme el kernel se desliza a través de la imagen binaria, la erosión le asgina un valor de 1 a un píxel si todos los pixeles cubiertos por el kernel son 1, de lo contrario, le asigna un valor de 0. De manera contaria, la dilatación le asigna a un píxel el valor de 1 si al menos uno de los píxeles cubiertos por el kernel es 1. Así, en esta tesis, primero se erosiona la imagen segmentada y después se dilata. De este modo, cuando se le aplican operadores morfológicos a la imagen de la figura 3.8, se obtiene la de la figura 3.6, en la que se ha eliminado el ruido e incrementado sutílmente el área del objeto identificado.

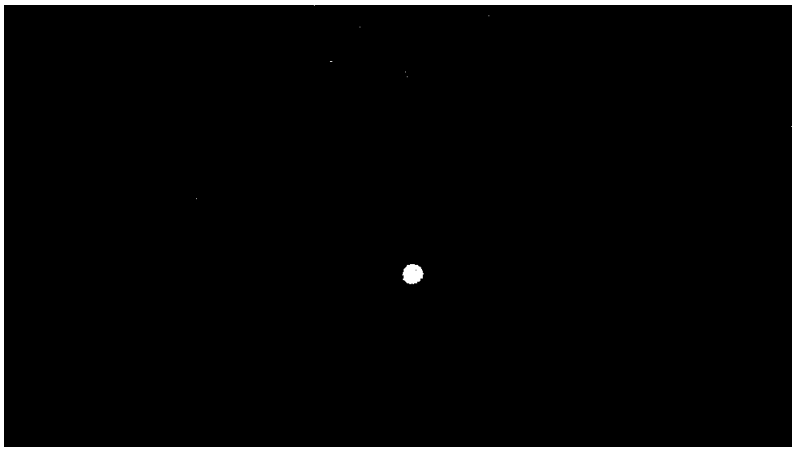

Figura 3.8: Pelota segmentada por su color y afectada por ruido. En esta imagen aún no se aplican operadores morfológicos.

Finalmente, el centroide del objeto (en este caso, de la pelota) es el resultado de calcular la media (en píxeles) tanto de las componentes x como de las componentes y del área identificada (área color blanco). Así, de lo mencionado en la sección 3.1, el centroide se calcula en la imagen distorsionada y se corrige de acuerdo con lo indicado en esa misma sección. De esta manera, el algoritmo de la sección 3.3 utiliza el centroide corregido (es decir, sin distorsión) para calcular la posición del objeto identificado (la pelota).

3.3. Localización visual

Después de detectar e identificar a la pelota a través de la segmentación de su color y de obtener su centroide en la imagen sin distorsión (consulte las secciones 3.1 y 3.2), es momento de explicar cómo se obtiene la posición de la pelota con respecto al pie izquierdo del robot. Para ello, considere que el robot con el que se trabaja tiene previamente asignados a lo largo de su estructura diversos marcos de referencia que se relacionan entre sí mediante transformaciones homogéneas (consulte el capítulo 5), donde resultan de interés el marco de referencia del pie izquierdo y el marco de referecencia de la cámara que se ubica en la cabeza.

De manera general, a partir del marco de referencia de la cámara, se busca obtener un nuevo marco de referencia rotado pero no trasladado cuyo eje x apunte, en todo momento, hacia el centro de la pelota, de tal forma que sea de ayuda para obtener la ecuación de una recta que atraviese el centro de la pelota y que permita calcular su posición con respecto al marco de referencia del pie izquierdo del robot. Para explicar detalladamente esta idea, considere el diagrama de la figura 3.9, en el que:

- $\boldsymbol{f}$ es el marco de referencia del pie izquierdo del robot.
- $\boldsymbol{c}$ es el marco de referencia de la cámara.
- $\boldsymbol{c_2}$ es el nuevo marco de referencia rotado pero no trasladado con respecto a $\boldsymbol{c}$.
- (x_b, y_b, z_b) es la posición del centro de la pelota con respecto a $\boldsymbol{f}$.
- $\vec{r}$ es la recta que atraviesa el centro de la pelota.
- θ y ϕ son los ángulos de *pitch* y *yaw*, respectivamente, de $\boldsymbol{c_2}$ con respecto a $\boldsymbol{c}$.
- $\vec{p}$ es un vector que va desde O_f hasta O_c y O_{c_2}.
- $\vec{q}$ es un vector que va desde O_f hasta un valor particular $x_{c_2} = x$.
- $\hat{u}$ es un vector unitario que apunta hacia el centro de la pelota y que se calcula como $\hat{u} = \frac{\vec{p}-\vec{q}}{|\vec{p}-\vec{q}|}$.
- D es el diámetro de la pelota.

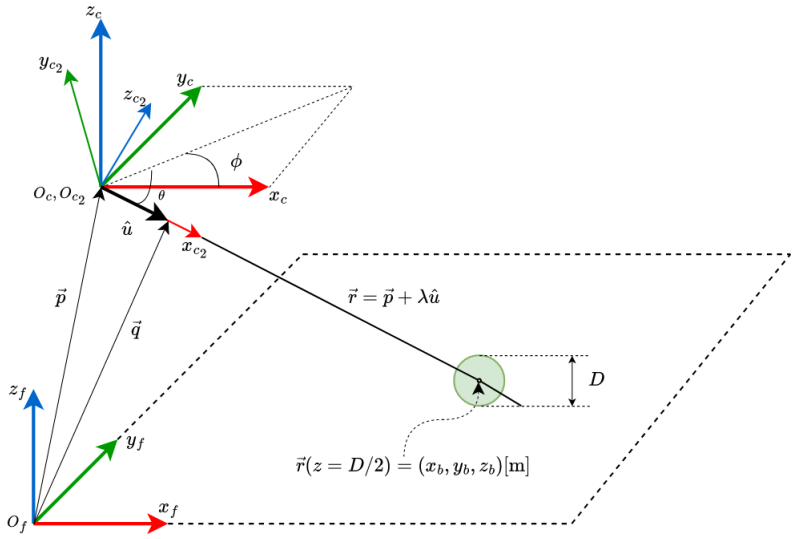

Figura 3.9: Conjunto de elementos (vectores, marcos de referencia, rectas, etc.) para calcular la posición (x_b, y_b, z_b) de la pelota con respecto al marco de referencia $\boldsymbol{f}$ del pie izquierdo del robot.

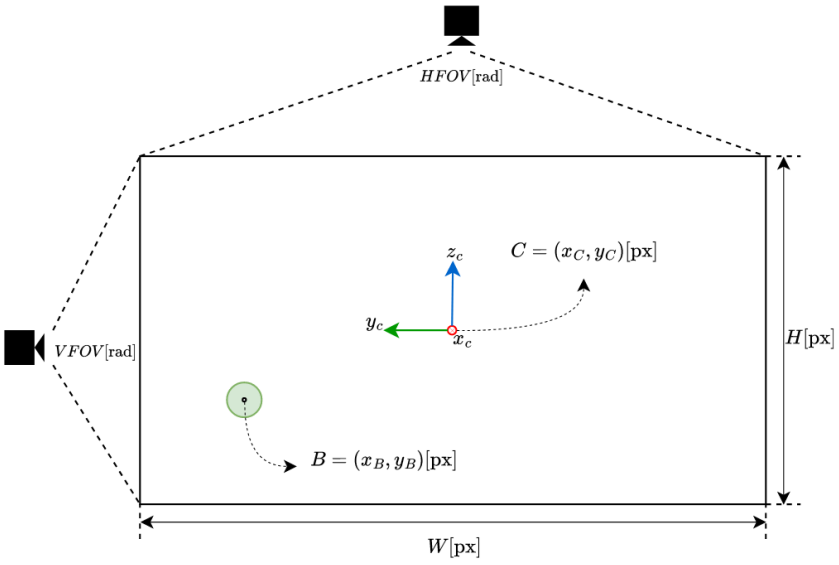

Figura 3.10: Representación de la imagen que captura la cámara pero sin las distorsiones radial ni tangencial

Para comenzar a entender cómo se calculan los ángulos de rotación θ y ϕ, tómese en cuenta la figura 3.10, que muestra una representación de la imagen que captura la cámara pero sin las distorsiones radial ni tangencial (véase la sección 3.1), donde W[px] y H[px] son, respectivamente, el ancho y el alto de la imagen, $C = (x_C, y_C)$[px] su centro, $B = (x_B, y_B)$[px] el centroide de la pelota (véase la sección 3.2), $HFOV$[rad] y $VFOV$[rad] los ángulos de visión horizontal y vertical de la cámara (proporcionados por el fabricante), respectivamente, y $\boldsymbol{c}$ el marco de referencia de la cámara. Así, los valores de θ y ϕ están dados por:

$$\theta = \left(\frac{VFOV}{H}\right)(y_B - y_C) \tag{3.16}$$

$$\phi = \left(\frac{HFOV}{W}\right)(x_C - x_B) \tag{3.17}$$

Las ecuaciones 3.16 y 3.17, además de mostrar el relevante papel que juega el centroide B de la pelota en el cálculo de θ y ϕ, consideran que el centro C de la imagen coincide con el origen O_c del marco de referencia $\boldsymbol{c}$ y plantean una relación de proporcionalidad entre los ángulos de visión y la resolución (W, H) de la imagen, donde la ecuación 3.16 la plantea entre H y $VFOV$ mientras que la ecuación 3.17 entre W y $HFOV$, lo que permite que la diferencia en [px] entre las componentes de B y C determine tanto la magnitud como el signo de θ y ϕ. Por lo tanto, el nuevo marco de referencia $\boldsymbol{c_2}$ (véase la figura 3.9), resulta de aplicar la matriz de rotación 3.18 al marco de referencia $\boldsymbol{c}$.

$$R_{c\,c_2} = R_z(\phi)R_y(\theta) \tag{3.18}$$

Así, después de que se obtiene el marco de referencia $\boldsymbol{c_2}$, continúa el planteamiento de la recta $\vec{r}$ (véase el diagrama de la figura 3.9) que pasa por el centro de la pelota y cuya forma es

$$\vec{r} = \vec{p} + \lambda\hat{u} \tag{3.19}$$

donde el vector unitario $\hat{u}$ se calcula como

$$\hat{u} = \frac{\vec{p} - \vec{q}}{|\vec{p} - \vec{q}|} \tag{3.20}$$

Para calcular $\vec{p}$ y $\vec{q}$, se utiliza la transformación homogénea $T_{f\,c_2}$ (de $\boldsymbol{c_2}$ a $\boldsymbol{f}$) para expresar el origen O_{c_2} y un punto x sobre el eje x_{c_2} con respecto a $\boldsymbol{f}$, de tal manera que $\vec{p}$ y $\vec{q}$ (véase la figura 3.9) se calculan como:

$$\begin{bmatrix}\vec{p}\\1\end{bmatrix} = T_{f\,c_2}\begin{bmatrix}0\\0\\0\\1\end{bmatrix} \tag{3.21}$$

$$\begin{bmatrix}\vec{q}\\1\end{bmatrix} = T_{f\,c_2}\begin{bmatrix}x\\0\\0\\1\end{bmatrix} \tag{3.22}$$

Debido a que $\vec{r}$ pasa por el centro de la pelota, se sabe que uno de los puntos que forman parte de $\vec{r}$ es $(x_b, y_b, z_b = D/2)$, por lo que, finalmente, se pueden conocer x_b y y_b si se obtiene λ a partir de $z_b = D/2$, tal y como se muestra a continuación:

$$(x_b, y_b, D/2) = \vec{p} + \lambda\hat{u}$$

$$(x_b, y_b, D/2) = (p_x, p_y, p_z) + \lambda(u_x, u_y, u_z)$$

$$\boxed{\lambda = \frac{\frac{D}{2} - p_z}{u_z}} \tag{3.23}$$

$$\boxed{x_b = p_x + \lambda u_x} \tag{3.24}$$

$$\boxed{y_b = p_y + \lambda u_y} \tag{3.25}$$

Una vez calculada la posición (x_b, y_b, z_b) del centro de la pelota, continúa la etapa de estimación de su posición y velocidad. Para ello, el capítulo 4 se dedica a estudiar el Filtro de Kalman y la manera en que esta tesis lo aplica para estimar los estados (posición y velocidad en los ejes x_f y y_f) de la pelota, considerada un sistema con un movimiento rectilíneo uniforme. La etapa de estimación resulta importante porque provee los datos necesarios para estimar el momento en el que la pelota deberá ser pateada.

Capítulo 4

Estimación de velocidad

En este capítulo se detalla la manera en que se aplica el Filtro de Kalman para utilizar la mediciones de posición que resultan del algoritmo de la sección 3.3 y estimar tanto la posición como la velocidad de la pelota que el robot debe patear. En la sección 4.1 se explica en qué consiste el Filtro de Kalman, se enuncian algunas de sus aplicaciones y se describe su algortimo de predicción-corrección. En la sección 4.2 se obtienen los modelos y parámetros necesarios para utilizar las ecuaciones del Filtro Kalman y estimar los estados (posición y velocidad) de la pelota, a la que se le considera como un cuerpo con movimiento rectilíneo uniforme. Además, se presenta un caso simulado para ejemplificar la manera en que esta tesis utiliza el Filtro de Kalman y estar un paso más cerca del objetivo: que el robot patee la pelota.

4.1. Filtro de Kalman

De acuerdo con Grewal y Andrews (2015), el Filtro de Kalman es un estimador lineal y óptimo que implementa un algoritmo de predicción-corrección para estimar el estado de un sistema lineal y estocástico con base en mediciones realizadas al sistema. Se le reconoce como un estimador óptimo porque minimiza el error cuadrático medio de las estimaciones que calcula. Así, es una herramienta matemática fundamentada en la teoría de probabilidad, los sistemas dinámicos, los sistemas estocásticos y en los mínimos cuadrados, a la que usualmente se recurre para estimar los estados de sistemas dinámicos y analizar el rendimiento de sensores. Sus aplicaciones van desde controlar procesos en una planta química (donde participan sensores de presión,

temperatura, etc.) hasta rastrear objetos como naves espaciales (a través de radares, sistemas de visión computacional, etc.). Debido a que en demasiadas aplicaciones no es posible (o no es deseable) medir todas las variables del sistema, el Filtro de Kalman estima las variables no medidas con base en estimaciones de las variables que sí lo son.

Así, después de introducir las características generales del Filtro de Kalman, continúa la introducción del Filtro de Kalman en tiempo discreto, por lo que se toma como base la detallada explicación de Simon (2006). De esta manera, suponga que se cuenta con el sistema lineal y discreto en el tiempo dado por:

$$x_k = F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \tag{4.1}$$

donde x_k es el estado en el tiempo k, x_{k-1} el estado en el tiempo $k-1$, F_{k-1} la matriz de transición, G_{k-1} la matriz de entrada de control, u_{k-1} la entrada de control y w_{k-1} el ruido del proceso. Se asume w_{k-1} con distribución normal, media cero y matriz de covarianza Q ($w_{k-1} \sim N(0, Q)$). A su vez, considere el modelo de medición:

$$y_k = H_k x_k + r_k \tag{4.2}$$

donde y_k es el vector de medición, H_k la matriz de medición y r_k el ruido de medición. Se asume r_k con distribución normal, media cero y matriz de covarianza R ($r_k \sim N(0, R)$).

El objetivo consiste en estimar el estado x_k con base en el conocimiento que se tiene de la dinámica del sistema (ecuación 4.1) y en las mediciones disponibles (ecuación 4.2). Si se tiene acceso a todas las mediciones, incluyendo las del tiempo k, es posible calcular una estimación *a posteriori*, denotada como $\hat{x}_k^+$. De lo contrario, si se tiene acceso a todas las mediciones pero no a las del tiempo k, se calcula una estimación *a priori*, denotada como $\hat{x}_k^-$. Tanto $\hat{x}_k^-$ como $\hat{x}_k^+$ son estimaciones de x_k, sin embargo, $\hat{x}_k^-$ estima a x_k antes de procesar la medición y_k, mientras que $\hat{x}_k^+$ lo hace después de procesarla. Por lo tanto, se podría esperar una mejor estimación por parte de $\hat{x}_k^+$ porque considera una mayor cantidad de información. A cada estimación se le asocia una matriz de covarianza P_k para denotar el error de estimación. A $\hat{x}_k^-$ se le asocia P_k^- y a $\hat{x}_k^+$, P_k^+.

De este modo, las ecuaciones de la etapa predictiva del filtro de Kalman son:

$$\hat{x}_k^- = F_{k-1}\hat{x}_k^+ + G_{k-1}u_{k-1} \tag{4.3}$$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{4.4}$$

donde la estimación *a posteriori* del tiempo $k-1$ se emplea para calcular la estimación *a priori* del tiempo k. Por ende, el valor de $\hat{x}_1^-$ depende de la estimación inicial $\hat{x}_0^+$, cuya covarianza es P_0^+. En general, P_0^+ representa la incertidumbre de la estimación inicial.

Finalmente, las ecuaciones de la etapa de corrección son:

$$K_k = P_{k-1}^-H_k^T(H_kP_{k-1}^-H_k^T + R_k)^{-1} \tag{4.5}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k\hat{x}_k^-) \tag{4.6}$$

$$P_k^+ = (I - K_kH_k)P_k^- \tag{4.7}$$

donde se utilizan $\hat{x}_k^-$ y y_k para calcular la estimación *a posteriori* $\hat{x}_k^+$. La matriz K_k de la ecuación 4.5 se conoce como la ganancia del Filtro de Kalman.

En la sección 4.2 se utiliza el Filtro de Kalman para estimar la posición y la velocidad de un objeto en movimiento, donde la posición es la única variable medida (véase la sección 3.3).

4.2. Estimación de velocidad

Después de que el robot utiliza los algoritmos descritos en el capítulo 3 para identificar y localizar a la pelota en movimiento, continúa la aplicación del Filtro de Kalman para estimar tanto la posición como la velocidad de la pelota.

Antes de implementar las ecuaciones de predicción y correción de la sección 4.1, se obtiene el modelo en tiempo discreto del sistema cuyos estados se desean estimar. En el caso particular de esta tesis, se delimita el sistema a

un cuerpo (la pelota) con **movimiento rectilíneo uniforme**, cuyos estados para el tiempo k son:

$$x_k = \begin{bmatrix} x_{1_k} \\ y_{2_k} \\ v_{1_k} \\ v_{2_k} \end{bmatrix} \tag{4.8}$$

donde x_{1_k} es la posición de la pelota en el eje x, y_{2_k} su posición en el eje y, v_{1_k} su velocidad en el eje x y v_{2_k} su velocidad en el eje y. Así, las ecuaciones que definen a cada uno de los estados, son:

$$x_{1_k} = x_{1_{k-1}} + v_{1_{k-1}}\Delta t + w_{1_{k-1}} \tag{4.9}$$

$$y_{2_k} = y_{2_{k-1}} + v_{2_{k-1}}\Delta t + w_{2_{k-1}} \tag{4.10}$$

$$v_{1_k} = v_{1_{k-1}} + w_{3_{k-1}} \tag{4.11}$$

$$v_{2_k} = v_{2_{k-1}} + w_{4_{k-1}} \tag{4.12}$$

donde Δt es el tiempo de muestreo (tiempo transcurrido entre el cálculo de cada estimación o entre la recepción de cada medición) y $w_{j_{k-1}}$ ($j = 1, 2, 3, 4$) el ruido de proceso. De este modo y de acuerdo con la ecuación 4.1, la representación en espacio de estados de las ecuaciones 4.9, 4.10, 4.11 y 4.12, es:

$$\begin{bmatrix} x_{1_k} \\ y_{2_k} \\ v_{1_k} \\ v_{2_k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{1_{k-1}} \\ y_{2_{k-1}} \\ v_{1_{k-1}} \\ v_{2_{k-1}} \end{bmatrix} + \begin{bmatrix} w_{1_{k-1}} \\ w_{2_{k-1}} \\ w_{3_{k-1}} \\ w_{4_{k-1}} \end{bmatrix} \tag{4.13}$$

donde la matriz de transición F es:

$$F = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

Por otra parte y de acuerdo con la ecuación 4.2, el modelo de medición del sistema es:

$$\begin{bmatrix} x_b \\ y_b \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_{1_k} \\ y_{2_k} \\ v_{1_k} \\ v_{2_k} \end{bmatrix} + \begin{bmatrix} r_{1_{k-1}} \\ r_{2_{k-1}} \\ r_{3_{k-1}} \\ r_{4_{k-1}} \end{bmatrix} \tag{4.15}$$

donde x_b y y_b son las posiciones en los ejes x y y, respectivamente, medidas por el algoritmo de la sección 3.3, y los elementos $r_{j_{k-1}}$ $(j = 1, 2, 3, 4)$ los ruidos de medición. Además, se observa que la matriz de medición es

$$H_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{4.16}$$

Una vez conocido el modelo del sistema (ecuación 4.13) y también su modelo de medición (ecuación 4.15), es posible implementar computacionalmente el algoritmo de predicción-corrección de la sección 4.1 (ecuaciones 4.3 y 4.4 para la etapa predictiva y ecuaciones 4.5, 4.6 y 4.7 para la correctiva). Inicialmente, al programa se le proporcionan una estimación inicial x_0^+ de los estados del sistema, la matriz de covarianza P_0^+ asociada a x_0^+, la matriz de covarianza Q asociada al rudio del proceso y la matriz de covarianza R asociada al ruido de medición. Las matrices P_k (tanto *a priori* como *a posteriori*), Q y R, tienen las siguientes formas:

$$P_k = \begin{bmatrix} \sigma_{x_1}^2 & 0 & \sigma_{x_1}\sigma_{v_1} & 0 \\ 0 & \sigma_{y_2}^2 & 0 & \sigma_{y_2}\sigma_{v_2} \\ \sigma_{v_1}\sigma_{x_1} & 0 & \sigma_{v_1}^2 & 0 \\ 0 & \sigma_{v_2}\sigma_{y_2} & 0 & \sigma_{v_2}^2 \end{bmatrix} \tag{4.17}$$

$$Q = \begin{bmatrix} q_{x_1} & 0 & 0 & 0 \\ 0 & q_{y_2} & 0 & 0 \\ 0 & 0 & q_{v_1} & 0 \\ 0 & 0 & 0 & q_{v_2} \end{bmatrix} \tag{4.18}$$

$$R = \begin{bmatrix} r_{x_b} & 0 \\ 0 & r_{y_b} \end{bmatrix} \tag{4.19}$$

P_k es una matriz de varianzas y covarianzas en la que se muestra la independencia de las variables de posición y velocidad en el eje x con respecto a las del eje y, aunque también en la que se muestra la dependencia de las posiciones con respecto a sus velocidades correspondientes. Por otra parte, se consideran independientes entre sí los ruidos de proceso, lo que también

se asume para los ruidos de medición y que explica la estructura interna de las matrices Q y R, cuyos elementos son únicamente varianzas (no hay covarianzas).

En esta tesis se utilizan las estimaciones del Filtro de Kalman para determinar el momento en el que la pelota pasará frente al pie del robot y será pateada. Aunque en la sección 5.3 se proporciona una explicación detallada del algoritmo para lograrlo, es importante resaltar que después de haber medido una N cantidad de posiciones y calculado sus respectivas estimaciones $\hat{x}_k^+$ (*a posteriori*), se calculan y utilizan en el resto del experimento las estimaciones $\hat{x}_k^-$ (*a priori*) para determinar el tiempo que le tomará a la pelota llegar al pie del robot. Para ejemplificar esto, las figuras 4.1, 4.2, 4.3, 4.4 y 4.5 muestran los resultados de una prueba en la que se simula el movimiento unidimensional (ya sea el eje x o y) de un cuerpo que se mueve de $0[m]$ a $1.6[m]$ con una velocidad constante de $0.2[m/s]$. La prueba utiliza los modelos 4.13 y 4.15 y los parámetros $\Delta t = 0.1[s]$, $Q = 0.001I$, $R = 0.002I$, $x_0^+ = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T$ y $P_0^+ = I$.

La figura 4.1 muestra las mediciones con respecto a los valores reales de posición (las mediciones se sintetizan, es decir, se generan valores de posición a los que se les suma ruido gaussiano). En las figuras 4.2 y 4.3 se grafican, a lo largo de toda la prueba, las estimaciones *a posteriori* de posición y velocidad, respectivamente (las estimaciones *a posteriori*, como se indica en la sección 4.1, consideran las mediciones tomadas). Las figuras 4.4 y 4.5 reflejan la manera en que se utiliza el Filtro de Kalman en esta tesis: Después de calcular una N cantidad de estimaciones *a posteriori* $\hat{x}_k^+$, se utilizan, para el resto del experimento, las estimaciones *a priori* $\hat{x}_k^-$ (ya no se procesan las mediciones). En ambas figuras, las gráficas de las estimaciones *a priori* son cercanas a las posiciones reales del objeto y también a las estimaciones *a posteriori* de velocidad. Dicho comportamiento es el que se desea obtener en una prueba real (véase el capítulo 6 de resultados).

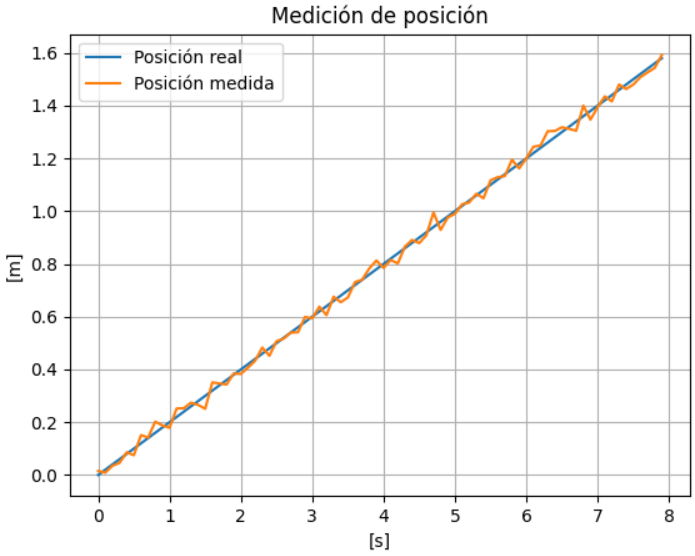

Figura 4.1: Mediciones y valores reales de posición.

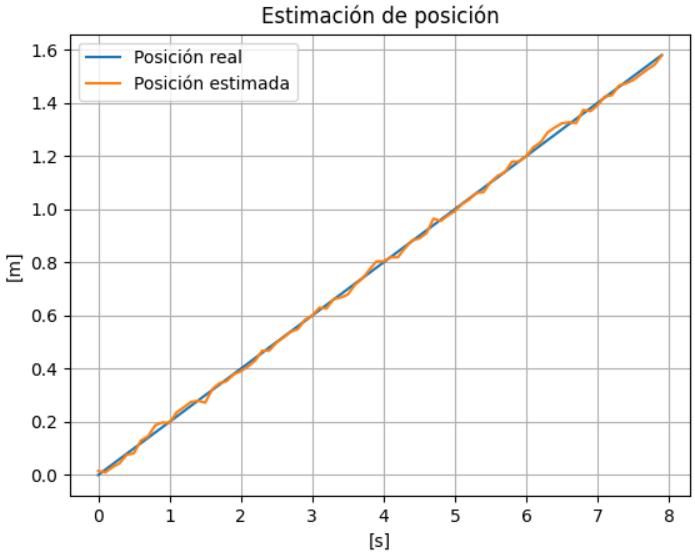

Figura 4.2: Estimaciones *a posteriori* de posición.

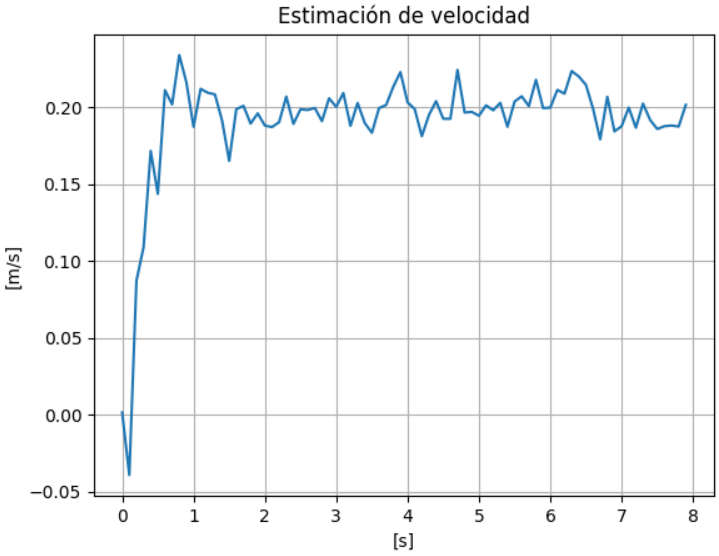

Figura 4.3: Estimaciones *a posteriori* de velocidad.

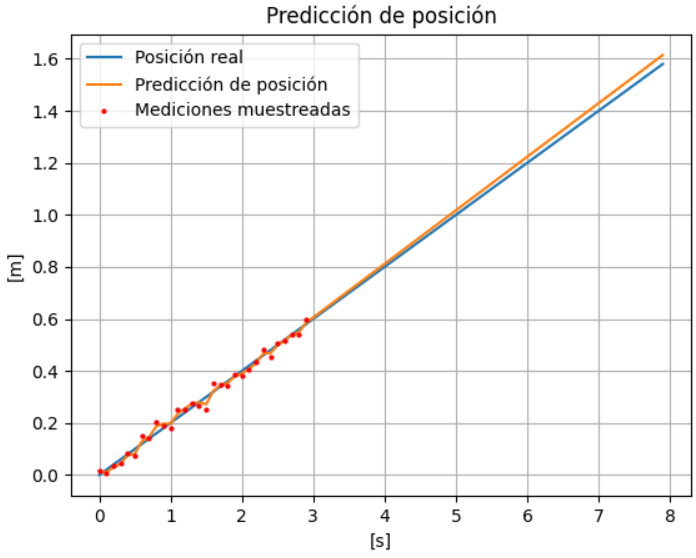

Figura 4.4: Estimaciones *a priori* de posición.

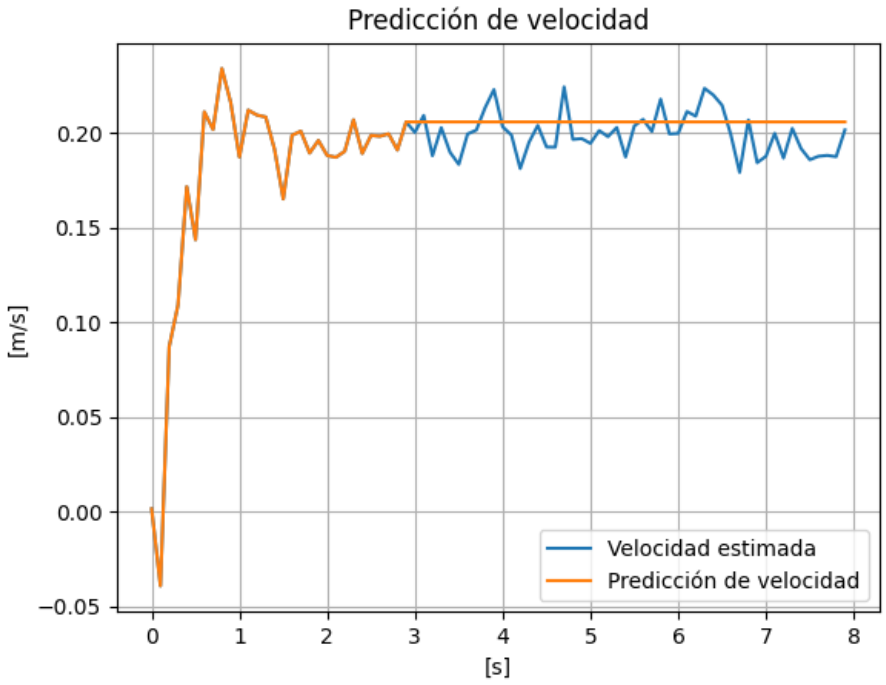

Figura 4.5: Estimaciones *a priori* de velocidad.

Una vez estimada la posición y velocidad de la pelota, continúa la ejecución de la secuencia de pateo, que incluye la estimación del tiempo que le tomará a la pelota llegar al pie del robot y el envío de la instrucción para patearla, lo que se discute detallamente en la sección 5.3. En general, hasta el momento se han discutido las maneras en que el sistema de visión computacional del robot corrige la distosión de la imagen capturada por la cámara, identifica a la pelota, calcula su posición y estima sus estados. Ahora, es momento de detallar en el capítulo 5 cómo estos enfoques se implementan en el robot real.

Capítulo 5

Implementación

Después de haber explicado los aspectos teóricos del sistema de visión computacional propuesto, es momento de detallar la manera en que se implementan. Para ello, en la sección 5.1 se presenta y describe el robot NimbRo-OP de la Facultad de Ingeniería de la UNAM en el que se implementa el sistema. Posteriormente, en la sección 5.2, se dan a conocer las herramientas de visión computacional ofrecidas por OpenCV y MATLAB que se utilizan para implementar los sistemas de detección y localización de objetos (véase el capítulo 3). En la sección 5.3 se detalla el algoritmo que ejecuta el robot para patear la pelota en el momento indicado. Finalmente, en la sección 5.4, se presenta a ROS, un middleware que se utiliza para comunicar a cada uno de los programas (nodos) que ejecutan una función específica dentro del sistema de visión computacional.

5.1. El robot Nimbro-OP

El sistema de visión computacional propuesto se implementa en el robot humanoide tipo *NimbRo-OP* del Laboratorio de Bio-robótica de la Facultad de Ingeniería de la UNAM, mostrado en la figura 5.1. Dicho robot, que forma parte de la plataforma abierta de humanoides *NimbRo-OP* [1], se constituye de 20 grados de libertad: 6 motores Dynamixel MX-106 [2] por pierna, 3 motores MX-64 [3] por brazo y dos motores MX-64 en el cuello. La visión computacio-

[1] https://www.nimbro.net/OP/NimbRo-OP.html
[2] https://emanual.robotis.com/docs/en/dxl/mx/mx-106/
[3] https://emanual.robotis.com/docs/en/dxl/mx/mx-64/

nal del humanoide se realiza a través de una cámara `USB100W03M` (mostrada en la figura 5.2) con una lente tipo ojo de pescado. La cámara provee una imagen con resolución de 1280×720 píxeles y cuenta con un campo de visión horizontal de 2.6 [rad], así como con uno vertical de 2.3 [rad]. En cuanto a la alimentación, el robot opera con una batería recargable LiPo de tres celdas, 11.1 $[V]$, 44.4 $[Wh]$ y 4000 $[mAh]$.

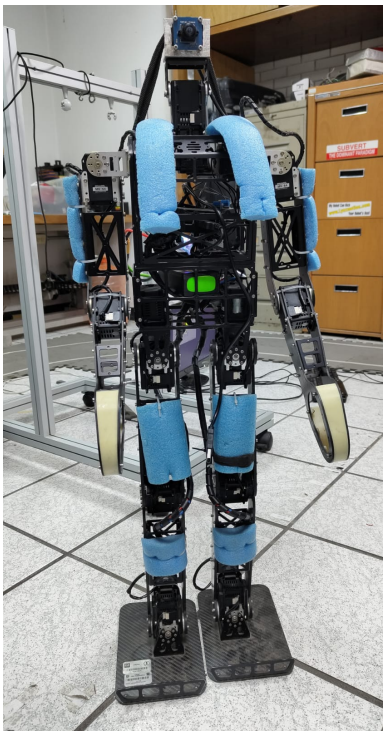

Figura 5.1: Robot humanoide tipo *NimbRo-OP* del Laboratorio de Biorobótica de la Facultad de Ingeniería de la UNAM.

Los diversos nodos descritos en la sección 5.4, se implementan en la computadora Raspberry Pi 4 Modelo B [4] (ARM-Cortex A72 de cuatro núcleos y 64 bits a 1,5 GHz) del robot. En ella, se tienen instalados el sistema operativo Ubuntu 20.04 LTS y la versión Noetic de ROS [5].

[4] https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf
[5] http://wiki.ros.org/noetic/Installation/Ubuntu

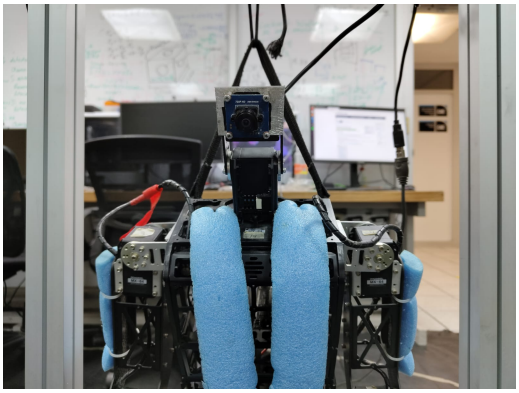

Figura 5.2: Parte superior del robot humanoide. Se observa la cámara `USB100W03M` con la que se implementa su visión computacional.

Este robot humanoide (figura 5.1) es el que se encargará de identificar y localizar a una pelota en movimiento y de determinar el momento adecuado para patearla (véase el capítulo 6 de resultados).

5.2. Herramientas de visión computacional

La implementación de las capacidades de identificación y localización de objetos del sistema de visión computacional detallado durante el capítulo 3, se realiza mediante el uso de las herramientas computacionales que ofrecen OpenCV y MATLAB, donde OpenCV es una biblioteca de código abierto en la que se incluyen más de 2500 algortimos de visión computacional y aprendizaje de máquina [6] , mientras que MATLAB es un plataforma de programación para analizar datos, desarrollar algoritmos y crear modelos [7].

El proceso de calibración de la cámara (véase la sección 3.1) se realiza de acuerdo con el procedimiento indicado por la documentación de OpenCV [8], en el que se utiliza la función `findChessboardCorners` para identificar los puntos en el tablero de ajedrez y la función `calibrateCamera` para calcular los parámetros intrínsecos, extrínsecos y los coeficientes de distorsión (radial y

[6]https://opencv.org/about/
[7]https://www.mathworks.com/discovery/what-is-matlab.html
[8]https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

tangencial) de la cámara. También se recurre a OpenCV [9] para implementar el sistema de identificación de objetos por segmentación de color (véase la sección 3.2), donde se emplean la funciones `cvtColor` (para convertir el espacio de color RGB de la imagen capturada al espacio de color HSV), `inRange` (para generar la imagen binaria en la que se indican los píxeles que cumplieron con el criterio de selección de color), `morphologyEx` (junto con el argumento `MORPH_OPEN`, para erosionar y después dilatar la imagen binaria), `findNonZero` (para obtener las ubicaciones en la imagen binaria de los píxeles cuyos valores no son cero, es decir, que fueron segmentados) y `mean` (para calcular la media de las ubicaciones de los píxeles segmentados y obtener el centroide de la pelota).

Como se menciona en las secciones 3.1 y 3.2, el centroide de la pelota se calcula originalmente en la imagen distorsionada, por lo que es necesario corregir su distorsión con el fin de calcular correctamente la posición de la pelota (veáse la sección 3.3). Esto implica la corrección de un punto y no de una imagen completa, por lo que se recurre a la función `undistortPoints` [10] de MATLAB para corregir el centroide de la pelota con base en los parámetros y coefientes obtenidos durante la calibración de la cámara con OpenCV. Esta función de MATLAB utiliza los modelos de distorsión radial y tangencial [11] mostrados en la sección 3.1 (ecuaciones 3.2, 3.3, 3.4 y 3.5).

Así, OpenCV y MATLAB juegan un papel relevante en la implementación del sistema de identificación y localización de objetos. Las imágenes y gráficas mostradas durantes las secciones 3.1 y 3.2 son productos de las funciones de ambas herramientas computacionales.

5.3. Secuencia de pateo

Con el objetivo de patear la pelota, el robot adquiere dos posiciones posibles: una con la que se prepara para patear la pelota (mostrada en la figura 5.3) y otra con la que la patea (mostrada en la figura 5.4). Dichas posiciones se definen previamente y se obtienen de manera experimental. En la posición de la figura 5.3, el robot sostiene su estructura en su pie izquierdo, levanta su

[9] https://docs.opencv.org/3.4/df/d9d/tutorial_py_colorspaces.html
[10] https://www.mathworks.com/help/vision/ref/undistortpoints.html
[11] https://www.mathworks.com/help/vision/ref/cameraparameters.html

pie derecho (con el que pateará), extiende levemente sus brazos e inclina su cámara para enfocarla en el suelo. Por otro lado, en la posición de la figura 5.4, el robot mantiene una posición similar pero, a diferencia de la anterior, extiende su pie derecho para patear la pelota. De lo comentado en la sección 5.4, el nodo `ball_kicker` envía las posiciones de los motores a los tópicos correspondientes para que, inicialmente, el robot adquiera la postura de la figura 5.3 y, posterioremente, adquiera la de la figura 5.4 (esto, después de haber recibido la instrucción de pateo emitida por el nodo `kick_indicator` a través del tópico `vision/ball_detector/kick_ball_indicator`).

Figura 5.3: Posición en la que el robot se prepara para patear la pelota.

Figura 5.4: Posición en la que el robot patea la pelota.

Una vez definidas las posiciones en el nodo `ball_kicker`, continúa la estimación del tiempo que le tomará a la pelota llegar al pie derecho del robot para ser pateada, lo que se realiza en el nodo `kick_indicator`. Para ello, considérese el diagrama de la figura 5.5, en el que O_f es el marco de referencia del pie izquierdo, $\hat{x}_k^-$ la estimación *a priori* de los estados de la pelota (consúltese la sección 4.2), $Y_F\,[m]$ la posición en el eje y_f del pie derecho, $t_F\,[s]$ el tiempo que le tomará a la pelota llegar a Y_F, $Y_T\,[m]$ el valor umbral con el que se detiene la actualización del valor de t_F, y $t_k\,[s]$ el tiempo que le toma al pie derecho patear la pelota.

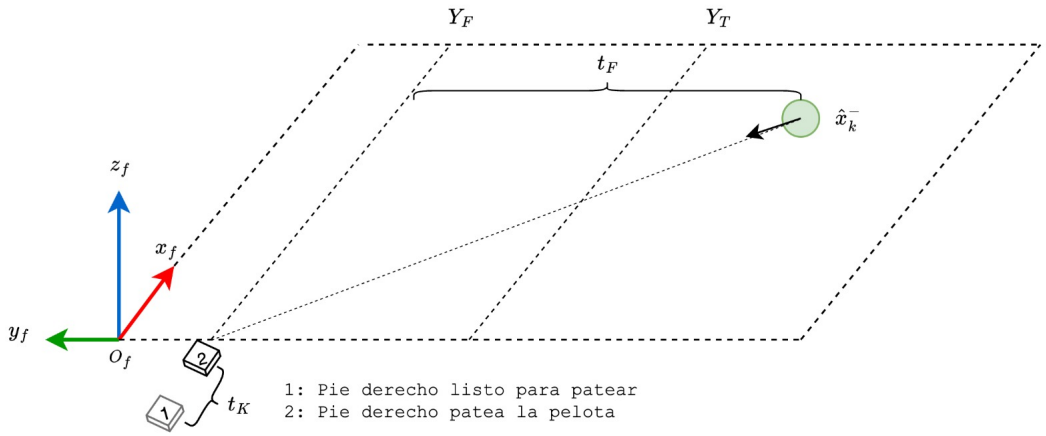

Figura 5.5: Estimación del tiempo que le tomará a la pelota llegar al pie derecho del robot para ser pateada.

Aunque el sistema de visión computacional provee estimaciones de posición y velocidad en ambos ejes (x_f y y_f), se delimita la secuencia de pateo únicamente a las estimaciones en el eje y_f, por lo que el algoritmo para estimar el tiempo en el que la pelota llegará al pie derecho es el siguiente: Después de haber calculado N estimaciones *a posteriori*, $\hat{x}_k^+$, se recurre a las estimaciones *a priori*, $\hat{x}_k^-$, para predecir la posición y_{2_k} y la velocidad v_{2_k} de la pelota. Considerando que la pelota se mueve con una velocidad constante del lado del eje y_f negativo, t_F se calcula como:

$$t_F = \frac{|y_{2_k}| - |Y_F|}{v_{2_k}} \qquad y_{2_k} < 0,\ Y_F < 0,\ y_{2_k} < Y_F \tag{5.1}$$

donde t_F se actualiza con cada estimación $\hat{x}_k^-$ calculada hasta que la predicción de posición $y_{2_k} \geq Y_T$. De esta manera, cuando la predicción de posición supera el umbral Y_T, t_F ya no se actualiza y el tiempo t_{kb} que debe transcurrir antes de que el pie derecho comience a moverse para patear la pelota es:

$$t_{kb} = t_F - t_K \tag{5.2}$$

Así, cuando el tiempo t_{kb} se cumple, se envía la instrucción de patear la pelota. El valor de Y_T se obtiene experimentalmente y la región $Y_T \leq y_{2_k} \leq Y_F$ es un espacio en el que es poco probable que el movimiento de la pelota se vea gravemente afectado. El cálculo de t_F y de t_{kb} se realiza en el nodo `kick_indicador` mientras que la intrucción de pateo en el nodo `ball_kicker`. En la figura 5.6, se muestra el algoritmo en forma de diagrama de flujo.

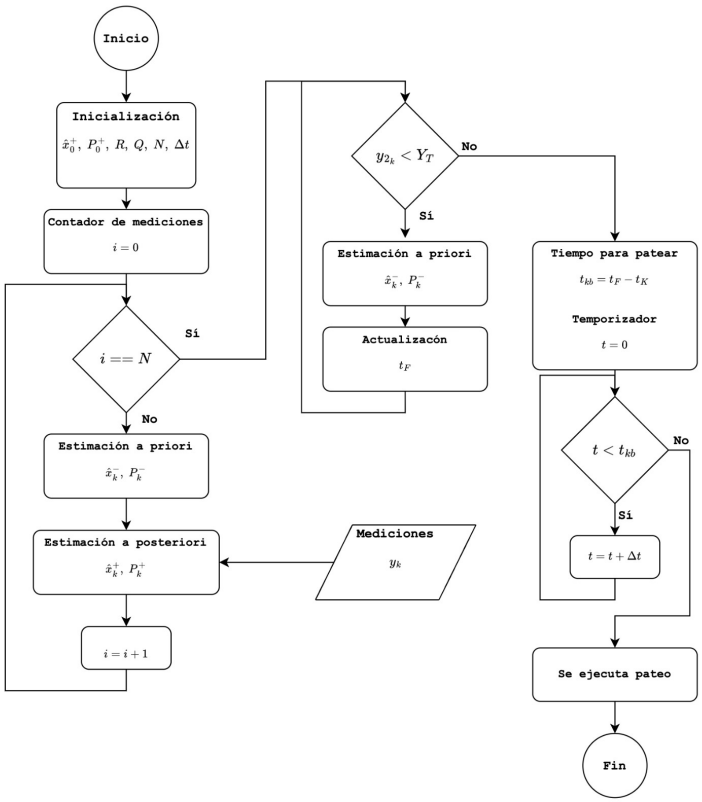

Figura 5.6: Algoritmo para ejecutar la secuencia de pateo.

5.4. La plataforma ROS

De acuerdo con Koubaa (2020), ROS (*Robotic Operating System*) es un *middleware* de código abierto que permite el desarrollo de complejos sistemas robóticos mediante el uso de nodos, tópicos y servicios. Los nodos son archivos ejecutables creados a partir de códigos (en Python, C++, MATLAB, etc.) que desempeñan una función específica dentro del sistema y que se comunican entre sí a través de tópicos y servicios. Los tópicos implementan canales de comunicación asíncronos, unidireccionales y con tipos de datos específicos llamados *messages*, a los que los nodos publican (envían datos) o se subscriben (reciben datos). Esta vía de comunicación le permite a un nodo enviar datos a uno o más nodos. Por otro lado, los servicios implementan canales de comunicación de tipo petición/respuesta, por lo que la comunicación es de uno a uno y no de uno a muchos. De este modo, en la figura 5.7 se dan a conocer los nodos, tópicos y servicios que constityen el sistema de visión computacional propuesto. Por consiguiente, en esta sección se busca detallar el papel desempeñado por cada nodo así como la comunicación que ejerce con otros.

Nodo camera_video_publisher

El nodo `camera_video_publisher` publica, con una frecuencia de 10 [Hz], la imagen capturada por la cámara del robot al tópico `/hardware/camera/image`, tópico que recibe y emite una estructura de datos tipo `sensor_msgs/Image`. Este nodo utiliza la función `VideoCapture` de OpenCV para obtener la imagen de la cámara y recurre a la función `cv2_to_imgmsg` para convertirla al tipo de dato `sensor_msgs/Image`. De este modo, los nodos subscritos al tópico `/hardware/camera/image` son capaces de recibir la imagen capturada por la cámara.

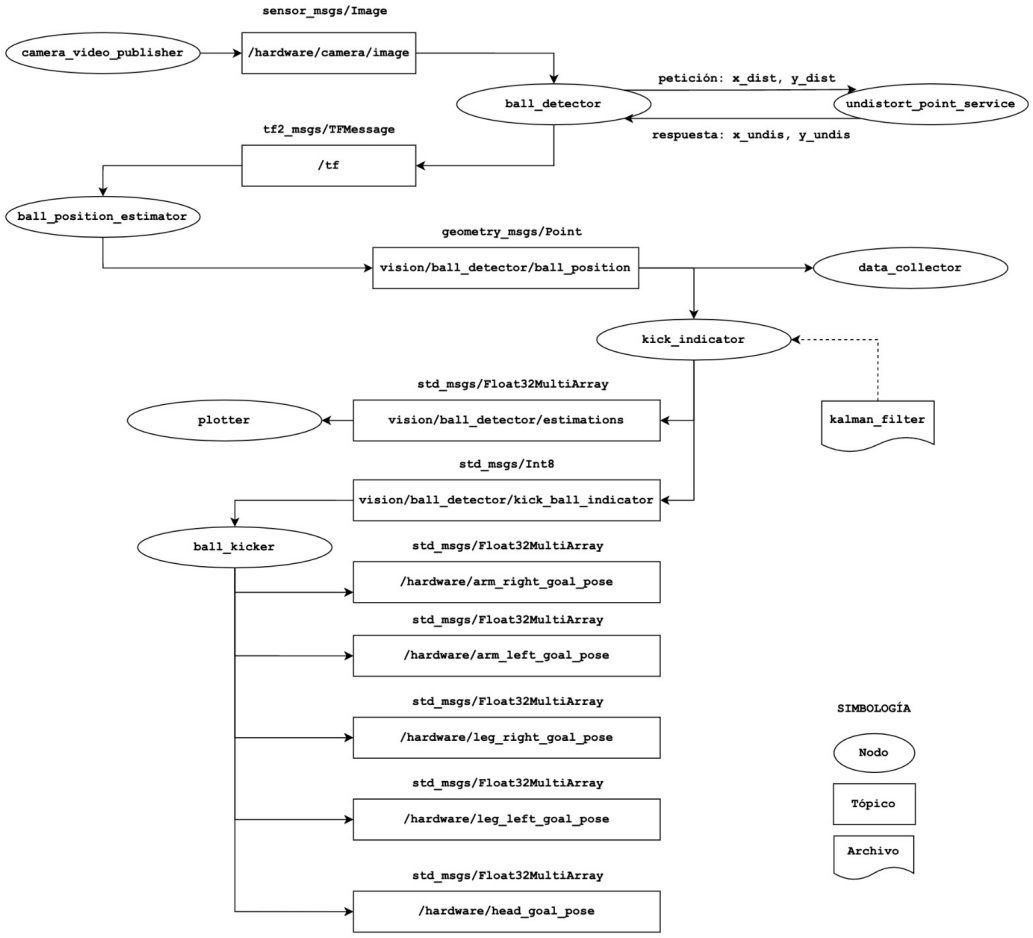

Figura 5.7: Grupo de nodos, tópicos y servicios que constituyen el sistema de visión computacional.

Nodo ball_detector

El nodo `ball_detector` se subscribe al tópico `/hardware/camera/image`, del que obtiene la imagen capturada por la cámara en formato `sensor_msgs/Image`, no obstante, debido a que requiere la imagen en el formato de OpenCV, emplea la función `imgmsg_to_cv2` para convertirla. De este modo, el nodo utiliza la imagen para identificar al objeto de interés (en este caso, a la pelota) mediante el enfoque discutido en la sección 3.2 y con ayuda de las herramientas de visión computacional ofrecidas por OpenCV (consúltese la sección 5.2).

Asimismo, el nodo se comunica con el nodo `undistort_point_service` y le solicita, a través de un servicio, la corrección del centroide calculado.

El robot NimbRo-OP de la sección 5.1 cuenta con diversos marcos de referencia asignados a lo largo de su estructura, de los que resultan de interés `left_foot_plane_link` (asignado a su su pie izquierdo) y `camera_optical` (correspondiente a su cámara). De esta manera y de acuerdo con lo detallado en la sección 3.3, el nodo `ball_detector` utiliza el centroide corregido para calcular el nuevo marco de refencia `new_camera_optical`, el cual está rotado pero no trasladado con respecto a `camera_optical` y cuyo eje x atraviesa el centro de la pelota. La transformación homogénea desde `camera_optical` hasta `new_camera_optical` se almacena en una estructura de datos propia de ROS llamada `tf2_msgs/TFMessage` y se publica en el tópico `/tf`.

Nodo undistort_point_service

Todos los nodos del sistema de visión computacional están en escritos en Python a excepción del nodo `undistort_point_service`, escrito en MATLAB para aprovechar la función `undistortPoints` mencionada en la sección 5.2. Este nodo recibe, a través de un servicio por parte del nodo `ball_detector`, las componentes distorsionadas x y y (llamadas `x_dist` y `y_dist`, respectivamente) del centroide de la pelota (véase la sección 3.1), utiliza la función `undistortPoints` para corregirlas y emite el centroide corregido mediante los valores `x_undis` (componente x corregida) y `y_undis` (componente y corregida).

Nodo ball_position_estimator

El nodo `ball_position_estimator` calcula la posición de la pelota de acuerdo con el procedimiento de la sección 3.3, para lo que requiere conocer las transformaciones homogéneas entre los marcos de referencia de interés (`left_foot_plane_link`, `camera_optical` y `new_camera_optical`). Para ello, el nodo recurre a las herramientas ofrecidas por el paquete `tf` [12] de ROS para transformar puntos y vectores entre marcos de referencia (`tf` le da seguimiento a las transformaciones homogéneas entre los distintos marcos de referencia). Cuando el nodo calcula la posición (x_b, y_b, z_b) del centro de la pelota con respecto a `left_foot_plane_link`, lo almacena en un mensaje de ROS

[12] http://wiki.ros.org/tf

llamado `geometry_msgs/Point` y lo publica en el tópico `vision/ball_detector/ball_position`.

Nodo kick_indicator

El nodo `kick_indicator` implementa el algoritmo de la figura 5.6 para estimar la posición y velocidad de la pelota y determinar el momento en el que se enviará la instrucción de pateo. El nodo recurre al archivo externo `kalman_filter` (escrito en Python) para utilizar las ecuaciones de las etapas predictiva y correctiva (véase la sección 4.1) del Filtro de Kalman. Además, las mediciones de posición emitidas por `ball_position_estimator`, las recibe por medio del tópico `vision/ball_detector/ball_position` (debido a la frecuencia con la que se publican las imágenes, el tiempo de muestreo es $\Delta t = 0.1\ [s]$). De este modo, cuando es momento de patear la pelota, `kick_indicator` envía un 1 al nodo `ball_kicker` a través del tópico `vision/ball_detector/kick_ball_indicator`.

Nodo ball_kicker

El nodo `ball_kicker` se subscribe al tópico `vision/ball_detector/kick_ball_indicator`. Si recibe un 0, publica los ángulos que deberán tomar los motores de las piernas, brazos y cabeza para que el robot adquiera la posición con la que se prepara para patear la pelota (véase la figura 5.3). Si recibe un 1, publica los ángulos para que el robot patee la pelota (véase la figura 5.4). Los ángulos de los motores se envían en un arreglo tipo `std_msgs/Float32MultiArray`. Los ángulos del brazo derecho se publican en el tópico `/hardware/arm_right_goal_pose`, los del brazo izquierdo en `/hardware/arm_left_goal_pose`, los de la pierna derecha en `/hardware/leg_right_goal_pose`, los de la pierna izquierda en `/hardware/leg_left_goal_pose` y los de la cabeza en `/hardware/head_goal_pose`.

Nodo data_collector

El nodo `data_collector` se subscribe al tópico `vision/ball_detector/ball_position` y almacena en un archivo CSV las posiciones calculadas por el nodo `ball_position_estimator`. De esta manera, se almacenan los datos de cada experimento con el objetivo de analizarlos y sintonizar las variables

necesarias (por ejemplo, el número de estimaciones *a posteriori* a calcular en el nodo `kick_indicator`).

Nodo plotter

El nodo `plotter` se subscribe al tópico `vision/ball_detector/estimations`, del que recibe las mediciones de posición y las estimaciones de posición y velocidad. Conforme se realiza un experimento, `plotter` grafica dichos valores para estudiar la manera en que se está ejecutando el experimento. Este nodo también grafica el momento en el que se envía la intrucción de pateo. Las gráficas producidas por `plotter` se muestran a lo largo del capítulo 6 de resultados.

De este modo, después de haber cubierto los aspectos teóricos y de implementación del sistema de visión computacional, es momento de explicar las diversas pruebas realizadas con el robot real y de presentar los resultados obtenidos (véase el capítulo 6).

Capítulo 6

Resultados

En este capítulo se dan a conocer los resultados de una serie de pruebas realizadas para evaluar el funcionamiento del sistema de visión computacional propuesto. En las secciones 6.1 y 6.2 se dan a conocer los resultados de medir, estimar y predecir la posición y velocidad de una pelota en movimiento pero sin ejecutar pruebas de pateo, las cuales se estudian en la sección 6.3 a través de diversas pruebas para evaluar el algoritmo 5.6 de secuencia de pateo.

6.1. Estimación de posición

En esta sección se presentan los resultados de medición y estimación de posición de tres pruebas (aún no se trata de pruebas de secuencia de pateo). Cada prueba consiste en medir las posiciones de una pelota verde en movimiento (con el algoritmo de la sección 3.3) y estimar sus estados de acuerdo con lo indicado en la sección 4.2. Para la ejecución de todas las pruebas de este capítulo, se dibuja previamente una cuadrícula en el espacio de experimentación con el objetivo de contar con los valores reales del marco de referencia del pie izquierdo del robot (véase la figura 6.1). Por otra parte, debido a las irregularidades del piso, la pelota corre el riesgo de no alcanzar una velocidad aproximadamente constante y de no lograr la trayectoria recta deseada. Por ello, se decide atar la pelota a un cuerpo que la impulse y con el que se logre, ante el sistema de visión computacional, una velocidad aproximadamente constante y una trayectoria recta (véase la figura 6.2). De esta manera, se presenta el análisis de las pruebas.

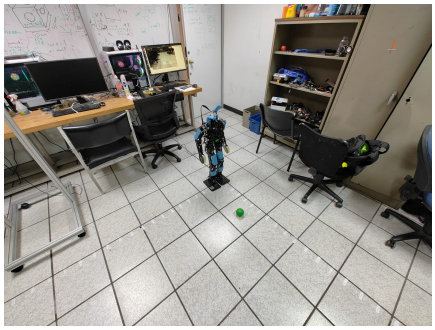

Figura 6.1: Se dibuja una cuadrícula en el espacio de experimentación. El origen de la cuadrícula coincide con el origen O_f del marco de referencia del pie izquierdo del robot.

Figura 6.2: La pelota es impulsada por un carrito para lograr que se mueva con una velocidad constante y una trayectoria recta.

Prueba 1 de posición

En la primera prueba, la posición real de la que partió la pelota fue $(1, -1)$ $[m]$ y en la que terminó $(0.3, -0.3)$ $[m]$ (en esta prueba, la pelota fue impulsada manualmente, es decir, una persona la empujó lentamente con ayuda del carrito a lo largo de todo el trayecto). El Filtro de Kalman utilizó los parámetros $\Delta t = 0.1[s]$, $Q = 0.001I$, $R = 0.005I$, $x_0^+ = [\,0\;0\;0\;0\,]^T$ y $P_0^+ = I$. Las figuras 6.3 y 6.4 muestran las trayectorias de la pelota en los ejes x y y, respectivamente, con respecto al marco de referencia del pie izquierdo del robot. En ambas figuras se grafican tanto las mediciones como las estimaciones *a posteriori* de posición. En las regiones I y III de la figura 6.3, la pelota se mantuvo inmóvil, mientras que en la región II mostró un

movimiento similar al rectilíneo uniforme. El mismo comportamiento se observa a lo largo del eje y en la figura 6.4. La importancia de seccionar ambas figuras radica en indicar que el movimiento de la sección II es el deseado al momento de ejecutar las pruebas de pateo (véase la sección 6.3).

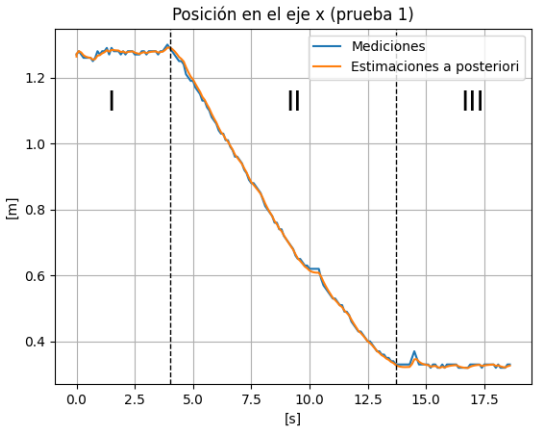

Figura 6.3: Prueba 1 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje x.

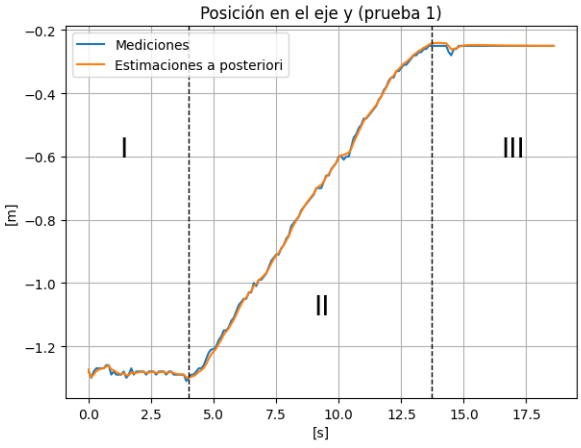

Figura 6.4: Prueba 1 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje y.

Por otra parte, con la intención de evaluar el desempeño de las estimaciones *a priori* a lo largo del y (recuérdese que la secuencia de pateo únicamente considera el movimento a lo largo del eje y, de acuerdo con lo indicado en la sección 5.3), en la figura 6.5 se grafican las mediciones y estimaciones *a priori* de posición sobre el eje y de únicamente la sección II de la figura 6.4. Las estimaciones de la figura 6.5 se componen de $N = 25$ estimaciones *a posteriori* (es decir, que procesan las primeras N mediciones) y el resto de estimaciones *a priori*. Se observa que las estimaciones de la etapa predictiva del Filtro de Kalman describen de manera aceptable el movimieto de la pelota. En la sección 6.2 se brinda el análisis de la estimación de velocidad para esta prueba.

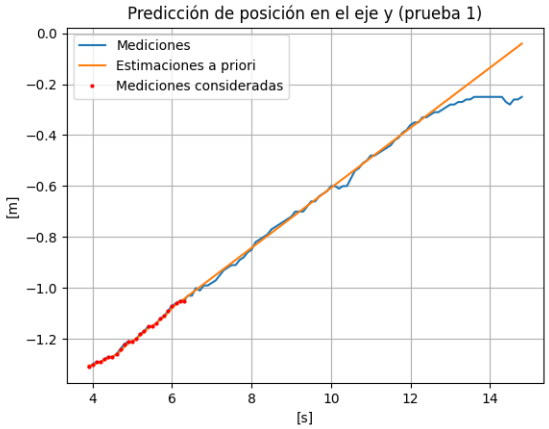

Figura 6.5: Prueba 1 de posición: Mediciones y estimaciones *a priori* de posición en el eje y.

Prueba 2 de posición

En la segunda prueba, la posición real con la que partió la pelota fue $(0.6, -1)$ $[m]$ y en la que terminó $(0.3, -0.3)$ $[m]$ (como se observa, la calibración imperfecta de la cámara introduce errores en las mediciones y estimaciones). Al igual que en la prueba anterior, la pelota fue impulsada manualmente. El Filtro de Kalman utilizó los parámetros $\Delta t = 0.1[s]$, $Q = 0.001I$, $R = 0.005I$, $x_0^+ = [0\ 0\ 0\ 0]^T$ y $P_0^+ = I$. Las figuras 6.6 y 6.7 muestran las mediciones y estimaciones *a posteriori* de posición de la pelota en los ejes x y y, respectivamente. En las regiones I y III de ambas figuras, la pelota se

mantuvo inmóvil, mientras que en la región II mostró un movimiento similar al rectilíneo uniforme.

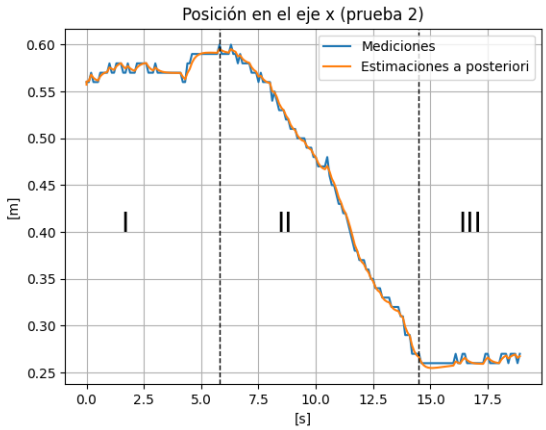

Figura 6.6: Prueba 2 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje x.

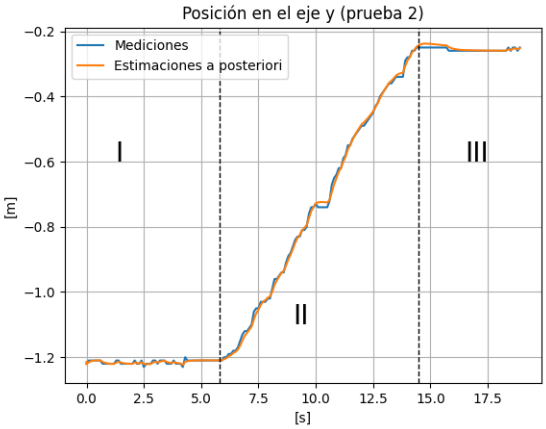

Figura 6.7: Prueba 2 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje y.

En la figura 6.8 se grafican las estimaciones *a priori* para la región II de la figura 6.7 (recuérdese, la región II de posiciones en el eje y es la región de

interés para la posterior secuencia de pateo). En ella, primero se calculan $N = 30$ estimaciones *a posteriori* (es decir, se procesan las primeras N mediciones) y después se calculan estimaciones *a priori* para el resto del experimento. Se observa que los resultados de la etapa predictiva describen de manera aceptable el movimiento de la pelota en el eje y. En la sección 6.2 se brinda el análisis de la estimación de velocidad para esta prueba.

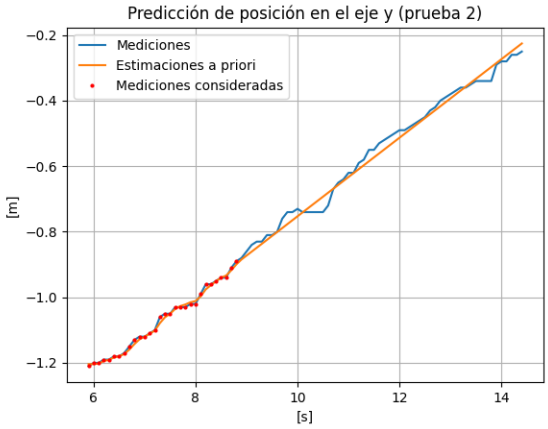

Figura 6.8: Prueba 2 de posición: Mediciones y estimaciones *a priori* de posición en el eje y.

Prueba 3 de posición

A diferencia de las pruebas anteriores, en la tercera prueba la pelota fue impulsada por el carrito y no por una persona. En esta prueba, la pelota partió de la posición real $(1.5, -1.5)$ $[m]$ y terminó en $(0.3, -0.3)$ $[m]$. El Filtro de Kalman utilizó los parámetros $\Delta t = 0.1[s]$, $Q = 0.001I$, $R = 0.005I$, $x_0^+ = [\,0\;0\;0\;0\,]^T$ y $P_0^+ = I$. En las figuras 6.9 y 6.10 se grafican las mediciones y estimaciones *a posteriori* de las posiciones por las que transcurrió la pelota a lo largo de su trayecto. De igual manera, ambas figuras se seccionan y se identifican a sus respectivas regiones II como las regiones de interés. Por otro lado, en la figura 6.11 se grafican las predicciones (estimaciones *a priori*) de posición en el eje y. Las predicciones resultan de calcular previamente $N = 12$ estimaciones *a posteriori* y describen de manera aceptable el movimiento de la pelota. Para conocer el análisis de las estimaciones de velocidad de esta prueba, consúltese la sección 6.2.

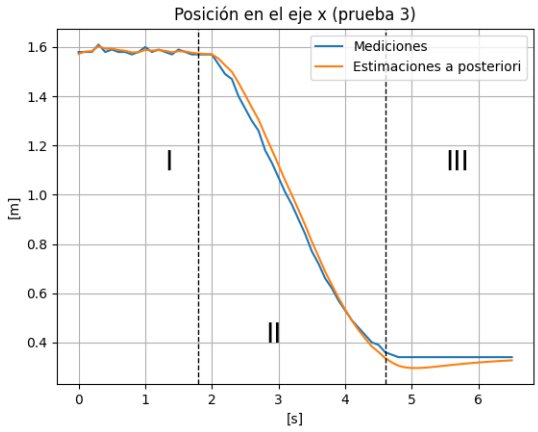

Figura 6.9: Prueba 3 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje x.

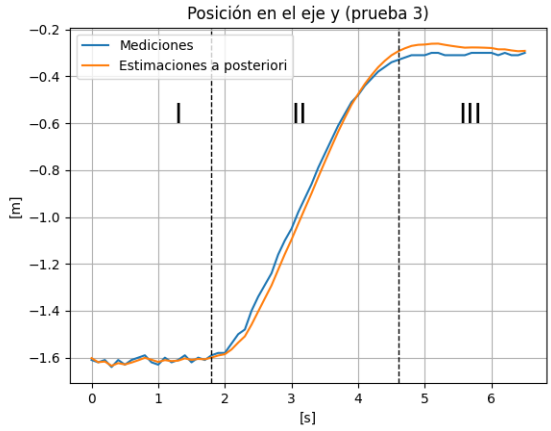

Figura 6.10: Prueba 3 de posición: Mediciones y estimaciones *a posteriori* de posición en el eje y.

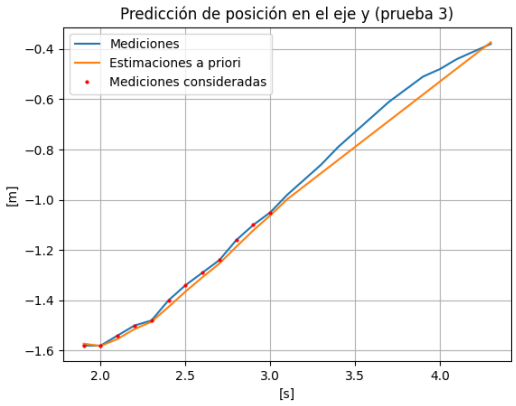

Figura 6.11: Prueba 3 de posición: Mediciones y estimaciones *a priori* de posición en el eje y.

6.2. Estimación de velocidad

En esta sección se dan a conocer las estimaciones de velocidad en el eje y (recuérdese, es el eje de interés para la secuencia de pateo) de las pruebas presentadas durante la sección 6.1, por lo que los resultados de esta sección utilizan los datos, parámetros y condiciones de experimentación de las pruebas de la sección anterior.

La figura 6.12 corresponde a las estimaciones de velocidad de la prueba 1 de posición. En la gráfica superior de dicha figura, se muestran y seccionan las estimaciones *a posteriori* de velocidad correspondientes a la trayectoria mostrada en la figura 6.4. Como se espera, la velocidad fue nula en las secciones I y III porque la pelota permaneció inmóvil, no obstante, también se observa una velocidad aproximadamente constante de 0.1 $[m/s]$ en la sección II. Así como en la figura 6.5 se muestran las predicciones de posición para la sección II de la figura 6.4, en la gráfica inferior de la figura 6.12 se dibujan las predicciones (estimaciones *a priori*) de velocidad para la sección II de la gráfica superior de la misma figura, donde las predicciones describen aceptablemente el comportamiento de las estimaciones *a posteriori*. Recuérdese, antes de recurrir únicamente a la etapa predictiva del Filtro de Kalman, se calculan N estimaciones *a posteriori* (en este caso, $N = 25$).

Las gráficas de la figura 6.13 correspoden a las estimaciones de velocidad de la segunda prueba de posición de la sección 6.1. Las gráficas de dicha figura muestran un comportamiento similar al observado en la prueba de velocidad anterior, donde las estimaciones de la sección II de la gráfica superior indican una velocidad también aproximadamente constante de 0.1 $[m/s]$. Las predicciones de la figura inferior (calculadas a partir del procesamiento $N = 30$ mediciones) describen de manera aceptable el comportamiento de las estimaciones *a posteriori*.

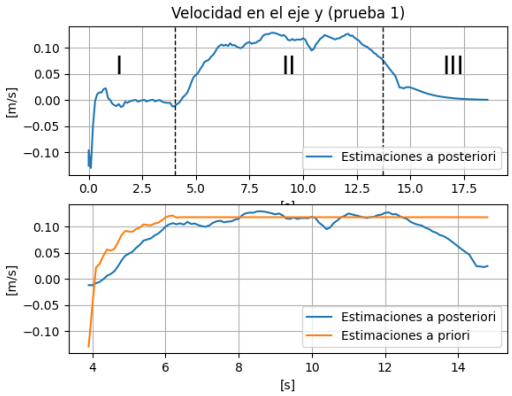

Figura 6.12: Prueba 1 de velocidad correspondiente a la prueba 1 de posición de la sección 6.1.

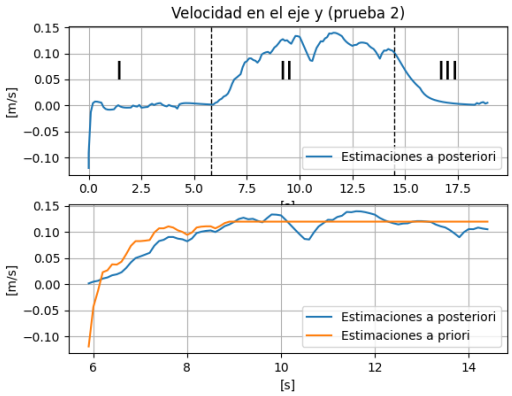

Figura 6.13: Prueba 2 de velocidad correspondiente a la prueba 2 de posición de la sección 6.1.

Como se menciona en la sección anterior, en las primeras dos pruebas la pelota fue impulsada manualmente (una persona la movió durante los experimentos) mientras que en la tercera fue impulsada únicamente por el carrito (tal y como se realiza durante las pruebas de pateo de la sección 6.3), lo que se refleja en las gráficas de velocidad de la figura 6.14. En la gráfica superior de dicha figura, durante el intervalo de 2 $[s]$ a 3 $[s]$ se aprecia el notable cambio de velocidad derivado de la aceleración del carrito al que estuvo atado la pelota, mientras que en el intervalo de 3.5 $[s]$ a 4.5 $[s]$ se observa la estabilización de la velocidad antes de entrar a la zona de desaceleración (las estimaciones *a posteriori* indican una velocidad máxima alcanzada de 0.5 $[m/s]$). En la gráfica inferior se dibujan las predicciones de velocidad para la región II de la gráfica superior, con base en $N = 12$ mediciones previamente procesadas.

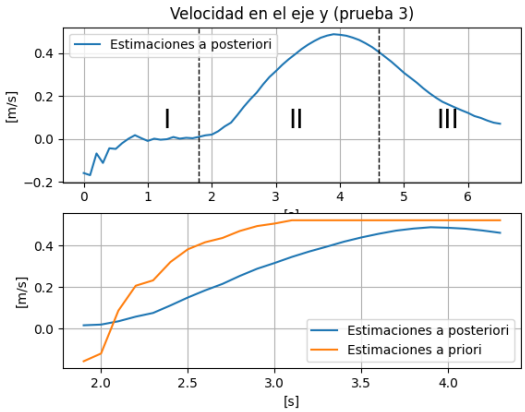

Figura 6.14: Prueba 3 de velocidad correspondiente a la prueba 3 de posición de la sección 6.1.

De los resultados obtenidos, se observa que a mayor velocidad las predicciones resultan de procesar una menor cantidad de mediciones, lo que se debe en gran medida a la frecuencia de 10 $[Hz]$ con la que se procesan las imágenes. Por lo tanto, a mayores velocidades se corre el riesgo de no calcular predicciones cercanas al movimiento real de la pelota. De esta manera, después de haber presentado los resultados de posición y velocidad para evaluar el algoritmo de la sección 3.3 (nodos `ball_detector` y `ball_position_estimator`) y la implementación del Filtro de Kalman, procede la presentación de resultados de la secuencia de pateo para evaluar el algoritmo 5.6 de la sección 5.3.

6.3. Pruebas de integración

En esta sección se muestran los resultados de tres pruebas en las que se utilizó el algoritmo 5.6 de la sección 5.3 para ejecutar la secuencia de pateo de una pelota en movimiento. Durante las tres pruebas, los parámetros utilizados fueron: $\Delta t = 0.1[s]$, $Q = 0.1I$, $R = 0.01I$, $x_0^+ = [\,0\;0\;0\;0\,]^T$, $P_0^+ = I$, $N = 20$, $t_K = 0.3$ $[s]$, $Y_F = -0.3$ $[m]$ y $Y_T = -0.5$ $[m]$. Las figuras 6.15, 6.17 y 6.19 ofrecen un resumen de lo ocurrido en las pruebas 1, 2 y 3 de pateo, respectivamente, donde en cada figura se grafican las mediciones y predicciones de posición así como los valores Y_T, Y_F y el momento en el que se ejecutó el pateo. Durante la primera prueba, la pelota alcanzó una velocidad de 0.4 $[m/s]$, sin embargo, cuando llegó al pie del robot en $Y_F = -0.3$ $[m]$, el pateo demoró aproximadamente 0.5 $[s]$ en ejecutarse (en la figura 6.15 se proporciona una serie de imágenes en la que se observa la manera en que el robot pateó la pelota). Por otra parte, durante la segunda prueba, la pelota alcanzó una velocidad de 0.6 $[m/s]$ y el pateo demoró aproximadamente 0.2 $[s]$ en llevarse a cabo (la secuencia de pateo se proporciona en la figura 6.18). Finalmente, en la tercera prueba la pelota alcanzó una velocidad de 0.7 $[m/s]$ y el pateo no demoró en ejecutarse, lo que significa que, conforme la pelota se acercó al pie, el robot comenzó a ejecutar el pateo (lo que se observa en la secuencia de la figura 6.20). Para lograr el resultado acertado de la tercera prueba, se buscó que la pelota ya estuviera en movimiento al momento de iniciar el sistema de visión computacional.

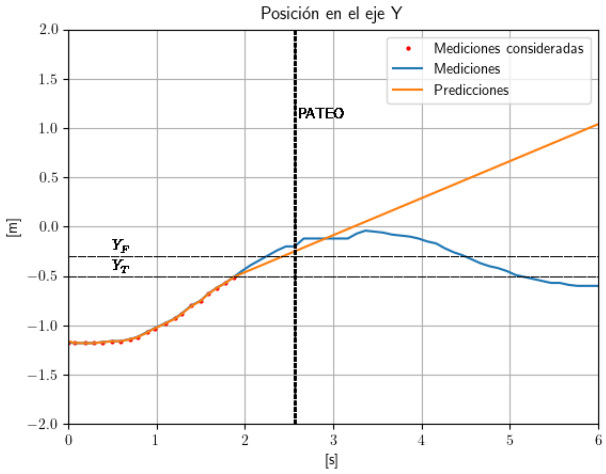

Figura 6.15: Predicción de posición para la prueba 1 de pateo.

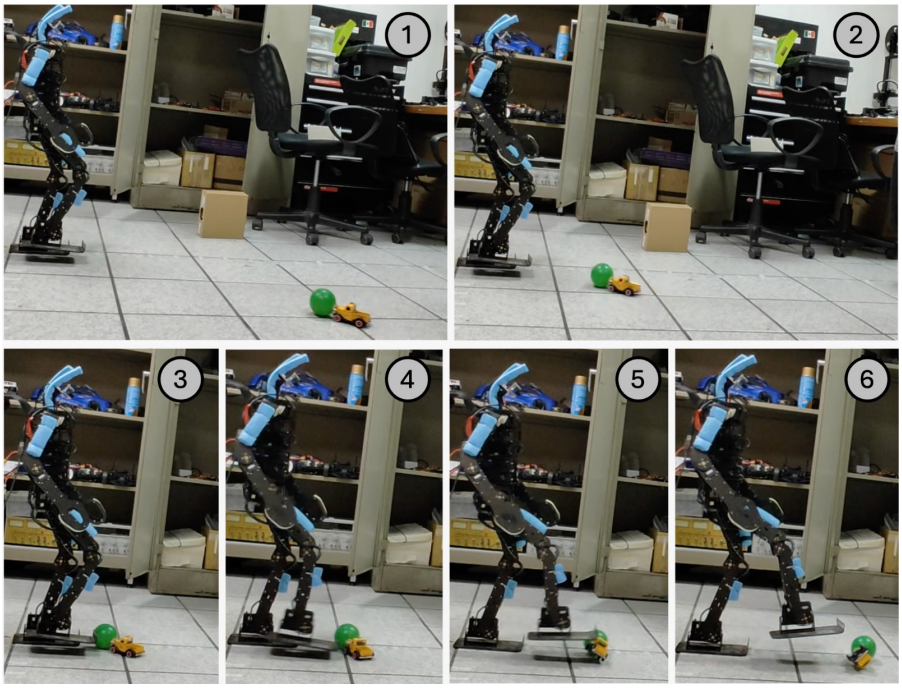

Figura 6.16: Ejecución de la prueba 1 de pateo.

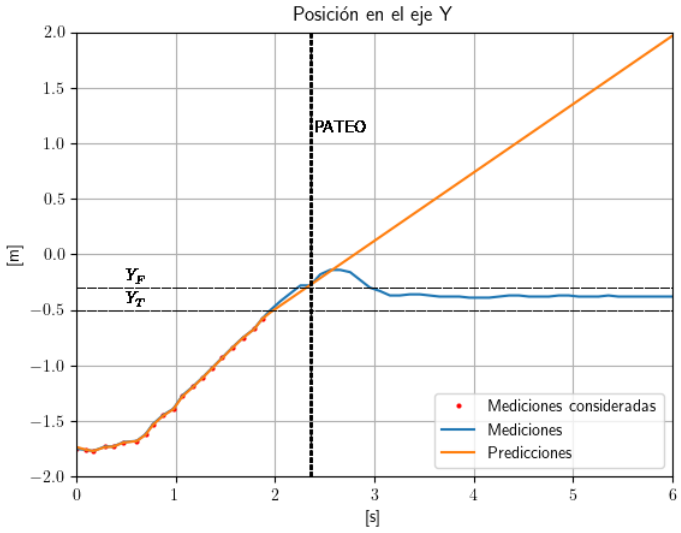

Figura 6.17: Predicción de posición para la prueba 2 de pateo.

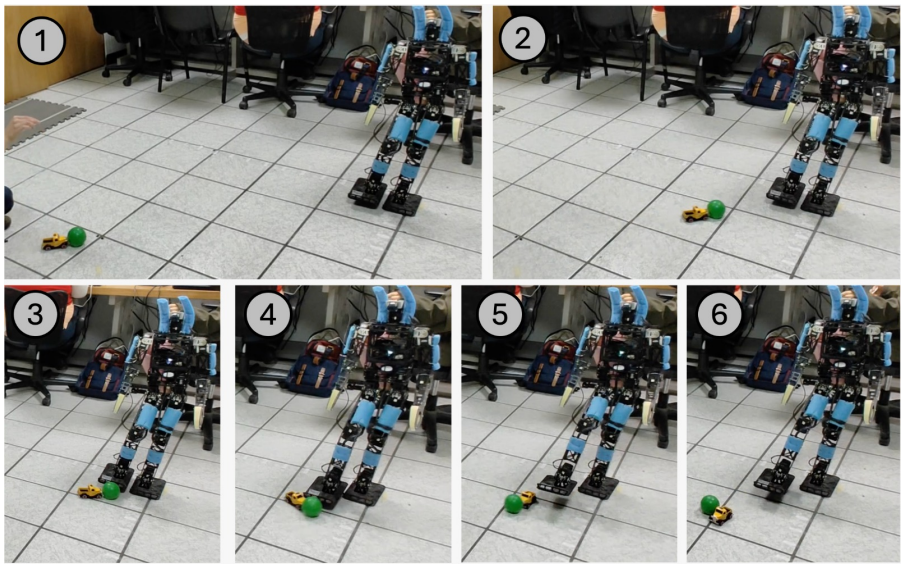

Figura 6.18: Ejecución de la prueba 2 de pateo.

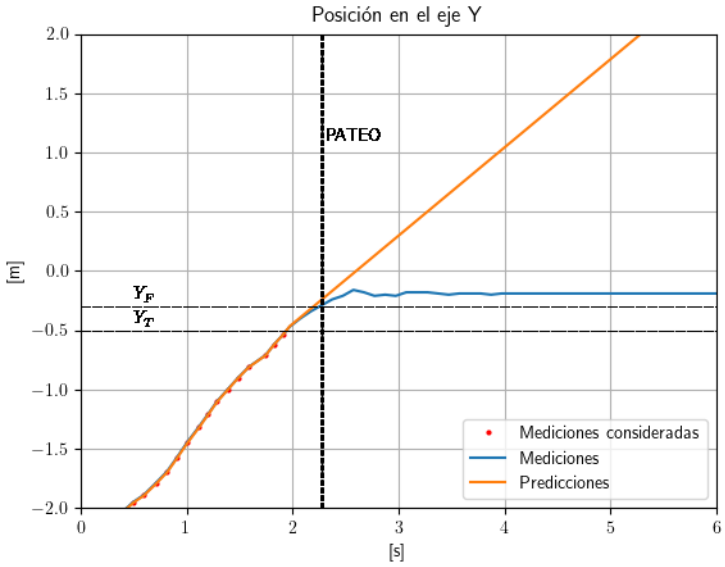

Figura 6.19: Predicción de posición para la prueba 3 de pateo.

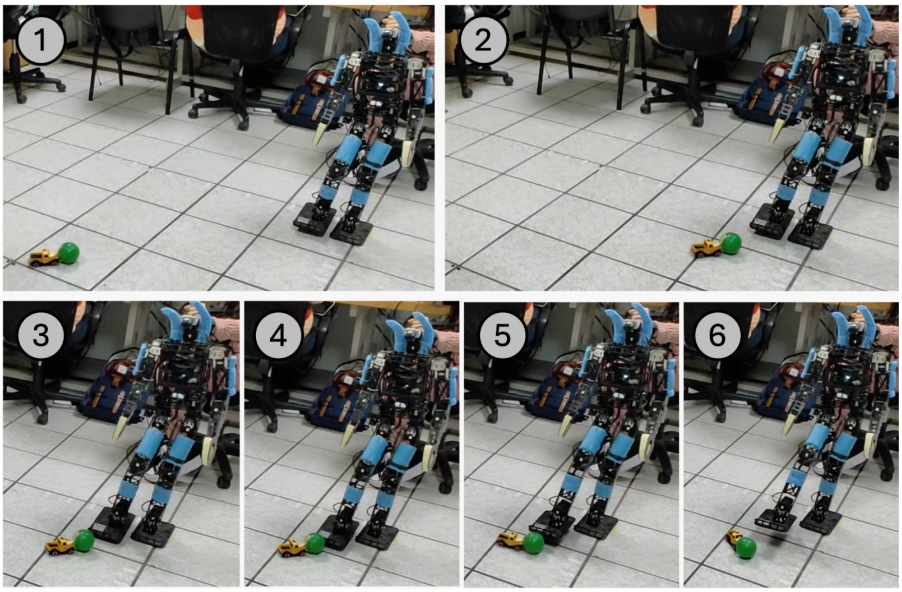

Figura 6.20: Ejecución de la prueba 3 de pateo.

De esta manera, se concluye el capítulo de resultados, donde las secciones 6.1 y 6.2 muestran los resultados de medir, estimar y predecir la posición y velocidad de la pelota pero sin ejecutar pruebas de pateo, las cuales se muestran en esta sección a través de tres puebas a diferentes velocidades, donde la tercera fue la más acertada de todas. Así, se continúa con el siguiente capítulo para dar a conocer las conclusiones y el trabajo futuro de esta tesis.

Capítulo 7

Discusión

7.1. Conclusiones

Se desarrolló, para el robot *NimbRo-OP* de la Facultad de Ingeniería, un sistema de visión computacional capaz de identificar y localizar objetos en movimiento. Para detectar un objeto de interés a partir de una imagen RGB, el sistema cambia el espacio de color de la imagen a HSV y segmenta al objeto por su color, de tal manera que calcula el centroide del objeto identificado a partir de la imagen binaria producida por la etapa de segmentación. Así, el sistema corrige la distorsión del centroide ocasionada por la lente de la cámara, calcula la posición del objeto con respecto a uno de los pies del robot y recurre al Filtro de Kalman para estimar tanto su posición como su velocidad.

Después de haber implementado el sistema de visión en una *Raspberry Pi* con ayuda de *ROS*, *OpenCV*, *MATLAB* y *Python*, se realizaron pruebas con el objetivo de evaluar la capacidad del robot para localizar y patear una pelota en movimiento. Como recordatorio de la sección 5.3, el algoritmo de pateo utiliza, después de haber procesado N estimaciones *a posteriori*, estimaciones *a priori* de posición y velocidad para calcular el tiempo aproximado de llegada de la pelota a la zona de pateo. Durante las primeras dos pruebas de pateo, el robot demoró una pequeña cantidad de tiempo en patear la pelota, mientras que en la tercera la pateó justo cuando pasó frente a él. Aquí se observó una de las desventajas de confiar completamente en el modelo y un pequeño conjunto de mediciones: Si la pelota experimenta perturbaciones no modeladas, las predicciones ya no describen de manera confiable su movi-

miento real. Finalmente, de las pruebas realizadas durante las secciones 6.1 y 6.2, aunque la mayoría de las mediciones y estimaciones de posición fueron cercanas a las posiciones reales, existieron errores derivados de la imperfecta calibración de la cámara. De esta manera y de acuerdo con lo platicado en el capítulo 6 de resultados, se observa la obtención de resultados satisfactorios al utilizar el sistema de visión computacional propuesto, por lo que se expresa el cumplimiento de los objetivos planteados en la sección 1.4.

7.2. Trabajo futuro

Como trabajo futuro se recomienda la búsqueda de técnicas basadas en apendizaje profundo (*Deep Learning*) para identificar un objeto y calcular su posición a partir de una imagen, así como para predecir su trayectoria si se encuentra en movimiento (lo que ayudaría a abarcar trayectorias distintas a la recta). Este nuevo y posible enfoque podría beneficiarse del algoritmo de la sección 3.3 para entrenar a un modelo de redes neuronales. Por otro lado, se plantea el desarrollo de un sistema para controlar la postura del robot al momento de patear, ya que en este trabajo el robot emplea posiciones previamente definidas que resultan ser inestables cuando se ejecuta el pateo. Finalmente, debido a que la distorsión ocasionada por la lente de una cámara juega un papel importante en la calidad de los resultados, se plantea la búsqueda de nuevas vías para calibrar una cámara y corregir la distorsión de un punto.

Referencias

Bradski, G., y Kaehler, A. (2008). *Learning opencv*. O'Reilly Media, Inc.

Farazi, H., Ficht, G., Allgeuer, P., Pavlichenko, D., Rodriguez, D., Brandenburger, A., ... Behnke, S. (2019). Nimbro robots winning robocup 2018 humanoid adultsize soccer competitions. En D. Holz, K. Genter, M. Saad, y O. von Stryk (Eds.), *Robocup 2018: Robot world cup xxii* (pp. 436–449). Cham: Springer International Publishing.

González Nava, L. E. (2021). *Sistema de visión para estimar posición y velocidad de objetos para un robot bípedo* (Tesis de licenciatura). Facultad de Ingeniería, UNAM.

Grewal, M. S., y Andrews, A. P. (2015). *Kalman filtering : theory and practice using matlab.* Wiley.

Koubaa, A. (2020). *Robot operating system (ros) the complete reference*. Springer.

Kuijpers, W., Neves, A. J. R., y van de Molengraft, R. (2017). Cooperative sensing for 3d ball positioning in the robocup middle size league. En *Robocup 2016: Robot world cup xx* (pp. 268–278). Cham: Springer International Publishing.

M. Lynch, K., y C. Park, F. (2017). *Modern robotics. mechanics, planning and control*. Cambridge University Press.

Nenchev, D., Konno, A., y Tsujita, T. (2018). *Humanoid robots: Modeling and control*. Elsevier Science.

Olguín Díaz, E. (2019). *3d motion of rigid bodies : a foundation for robot dynamics analysis.* Springer International Publishing.

Paetzel, M., y Hofer, L. (2019). The robocup humanoid league on the road to 2050 [competitions]. *IEEE Robotics and Automation Magazine, Robotics and Automation Magazine, IEEE, IEEE Robot. Automat. Mag*, *26*(4), 14–16.

Siciliano, B., y Khatib, O. (2016). *Springer handbook of robotics.* Springer.

Simon, D. (2006). *Optimal state estimation : Kalman, h and nonlinear approaches.* Wiley-Interscience.

Stockman, G., y Shapiro, L. G. (2001). *Computer vision.* Prentice Hall PTR.

Szeliski, R. (2022). *Computer vision : Algorithms and applications.* Springer International Publishing.

W. Spong, M., Hutchinson, S., y Mathukumalli, V. (2005). *Robot modeling and control.* Willey.

Zhou, C. (2004). Rules for the robocup humanoid league. *Advanced Robotics*, *18*(7), 721-724.