UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Estimación de trayectoria a partir de imágenes RGB-D para un robot móvil autónomo

TESIS

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Sergio Alvarado Ramos

DIRECTOR DE TESIS

Dr. Marco Antonio Negrete Villanueva

Ciudad Universitaria, Cd. Mx., 2022

Agradecimientos

A mi familia, que siempre creyó en mí y me apoyó en todo momento, especialmente mi madre.
A mis profesores, por su dedicación, por compartir sus conocimientos y su paciencia.
A mis amigos, porque fueron mis compañeros en las incontables horas de estudio.
Al proyecto PAPIIT TA101222, que por su apoyo se realizó este trabajo.

Resumen

En el presente trabajo se busca estimar la trayectoria de un objeto en movimiento, para lograrlo se utiliza la visión artificial utilizando cámaras con tecnología RGB-D que permite medir la profundidad de los objetos respecto a la ubicación de la cámara, se realizarán pruebas con una aplicación en particular la cual es atrapar una pelota.

Se busca contribuir a la estimación de objetos en movimiento porque se necesitan robots capaces de manipular y esquivar objetos en movimiento, en el laboratorio de Biorrobótica de la Facultad de Ingeniería de la UNAM ya se ha trabajado en este campo utilizando el filtro de Kalman extendido y se quiere probar con otro enfoque por lo que se utiliza el método de mínimos cuadrados para ajustar una trayectoria parabólica. Existen muchos tipos de movimiento y estudiarlos a todos puede ser complejo por lo que se decidió delimitar esta tesis al estudio de un tiro parabólico.

Es necesario segmentar el objeto deseado, pues la imagen que proporciona una cámara contiene la información de todo el ambiente visible y de alguna forma se tiene que eliminar la información que no corresponde al objeto de interés, para esto se utiliza la segmentación por color en el espacio HSV. La cámara entrega información sobre el color en el espacio RGB pero la segmentación en este color no es tan eficiente como lo es en HSV por lo que se realiza una transformación de RGB a HSV en la imagen, después de segmentar se calcula la posición del objeto en el espacio, esta posición esta dada de acuerdo al sistema de referencia de la cámara pero resulta mejor obtenerla de acuerdo a otro sistema de referencia que sea paralelo al plano en donde se moverá por la localización de este, por esta razón se utiliza una transformación homogénea que permita cambiar la posición de un sistema de referencia a otro. Ya que se cuenta con las coordenadas de la pelota se debe determinar cuando la pelota se encuentra en movimiento para esto se utiliza un criterio que se deduce a partir del caso discreto de las derivadas, al cumplirse se procede a guardar las muestras y estimar una ecuación que describa el movimiento de la pelota. Un tiro parabólico puede ser descrito por una ecuación cuadrática y haciendo algunas consideraciones es posible calcular las coordenadas de interés a partir de dicha ecuación; a grandes rasgos así funciona el sistema de visión.

Además del sistema de visión, se necesita de una plataforma con las características suficientes para realizar las pruebas suficientes y probar que se puede atrapar un objeto en movimiento, por lo que se plantearon ciertas características

y se diseñó un robot capaz de moverse en cualquier dirección y estimar propia su localización, para lograr esto se utiliza la teoría de control y ecuaciones que definen las odometría del robot.

Para reducir la complejidad y facilitar el mantenimiento del código todas las tareas descritas se dividieron en diferentes nodos que se comunican entre sí utilizando ROS. Finalmente se realizaron varias pruebas simuladas y reales para observar como reaccionaba el sistema propuesto.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Actualmente, existen robots móviles teleoperados que han ganado popularidad, estos robots son utilizados en ambientes peligrosos y son operados a distancia por humanos que les proporcionan localización y cognición, pero estos coordinan sus propios movimientos [17]. Algunos ejemplos destacados de estos robots son:

- El robot Sojouner usado en la exploración a Marte en 1997.
- Plustech este robot fue diseñado para mover madera en los bosques.
- El robot Pioneer fue hecho para realizar exploración en Chernobyl.

Sin embargo, en los últimos años, los robots móviles autónomos han experimentado un gran avance debido al desarrollo tanto del poder de cómputo como de los algoritmos para el procesamiento de señales y toma de decisiones, a diferencia de los robots teleoperados estos no necesitan de un humano que los guíe.

Este campo de estudio tiene su origen en disciplinas como la ingeniería en computación, eléctrica, electrónica, mecánica e incluso algunas ciencias sociales. Los robots autónomos son importantes debido a que en un futuro podrán ser usados en tareas peligrosas o difíciles de realizar para un humano como lo son la exploración planetaria, el patrullaje, rescate de emergencia, intervención en ambientes extremos, construcción, medicina, entre otras, además, estos pueden facilitar nuestra vida y mejorarla [13, 17]. A pesar de ser un campo reciente ya existen robots autónomos, algunos ejemplos son:

- HELPMATE: Este robot se encuentra en ciertos hospitales y se usa en tareas que requieren transportar objetos. Cuenta con una cámara en la parte superior que le permite detectar lámparas en el techo, estas funcionan como puntos de referencia que le permiten navegar por los corredores.
- BR 700 y RoboCleaner RC 3000: Ambos fueron desarrollados por Alfred Kärcher GmbH & Co. como robots de limpieza. El modelo BR700 cuenta

con un sistema sonar y un giroscopio que le permiten navegar, mientras el modelo RC 3000 cuenta con sensores ópticos que miden el grado de contaminación en el aire aspirado.

- B21: Desarrollado por iRobot este robot cuenta con 3 procesadores Intel y una gran variedad de sensores que le permiten realizar tareas de navegación.

Con los ejemplos antes mencionados sobre robots móviles autónomos se puede afirmar que ya se cuenta con la tecnología y algoritmos suficientes para continuar desarrollando más sistemas de este tipo.

1.1. Motivación

Las cámaras son probablemente el sensor más barato y accesible debido a la reducción de costos que se ha logrado con las imágenes digitales, a pesar de que existen otras tecnologías capaces de estimar posición como GPS, IMU y sensores de distancia (láser, infrarrojo, etcétera), las cámaras tienen mayor ancho de banda, esto significa que logran capturar más información que los otros sensores. La información obtenida en una imagen no es tan explícita como la que se obtiene con las otras tecnologías pero esto se compensa con la riqueza de la información, por ejemplo, en la imagen 1.1 se observan distintos objetos y con los algoritmos correctos una máquina puede ser capaz de identificar cada objeto y estimar su distancia o algún otro parámetro, a esto se le llama visión artificial [15].

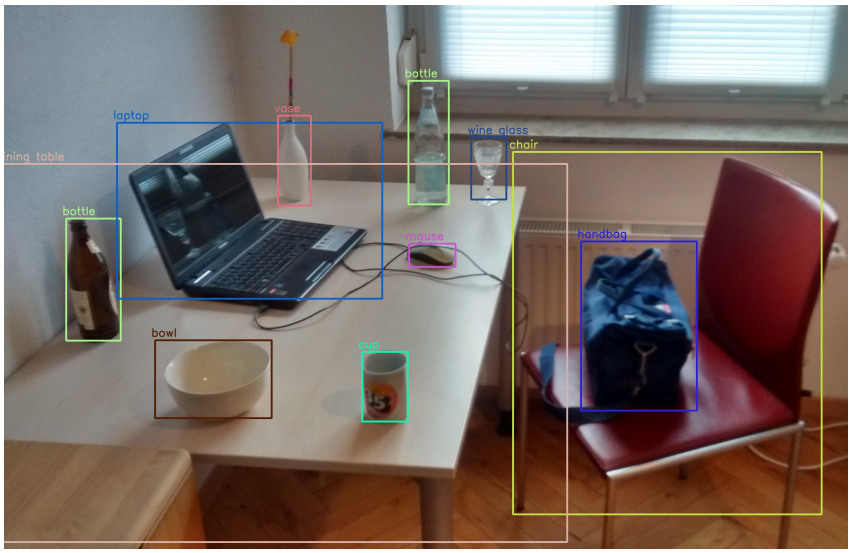


Figura 1.1: Imagen que utiliza visión artificial para el reconocimiento de objetos. Imagen extraída de la búsqueda "yolo vision"[1].

La visión artificial se ha vuelto importante en la actualidad, un ejemplo de esto es la exploración planetaria ya que para dicha tarea es necesario que los robots puedan tomar decisiones por sí mismos, pues se complica operarlos a distancia y por otro lado el robot se encuentra en ambientes de los cuales no se tiene información, por estas razones se necesita que estos sean capaces de tomar decisiones a través de algoritmos que involucran una meta y la información capturada del ambiente a través de cámaras [4].

En otras aplicaciones en las que se requiere alta exactitud y velocidad de respuesta también se requiere de visión artificial pues un humano no es capaz de realizar cálculos y tomar decisiones con la suficiente exactitud en un corto tiempo.

Ya se mencionó que la visión artificial es indispensable en robots móviles autónomos por la información que se puede capturar, también se sabe que muchos de estos cuentan con cámaras RGB-D por su bajo costo, por estas razones esta tesis utilizará la tecnología RGB-D para resolver uno de los problemas más comunes de la visión artificial que es la estimación de trayectoria de objetos, esto consiste en que a partir de una secuencia de imágenes, un robot sea capaz de calcular a dónde se dirige un determinado objeto. Existen muchos tipos de movimiento por lo que estudiarlos a todos resulta complejo, por eso esta tesis buscará estudiar particularmente un tiro parabólico buscando extrapolar el método en otros movimientos.

1.2. Planteamiento del problema

Se requiere un sistema que pueda estimar la trayectoria de un objeto en movimiento, lanzado en tiro parabólico, y un sistema de control, lo suficientemente rápido, para que un robot móvil omnidireccional pueda atraparlo antes de que caiga al suelo, de igual forma, el sistema debe ser lo suficientemente ligero para ser implementado en un sistema embebido como una Raspberry o una tarjeta de desarrollo Nvidia Jetson Nano.

El robot móvil a construir debe tener el hardware con los requerimientos necesarios para implementar tanto el sistema de control como el sistema de visión artificial.

1.3. Hipótesis

- Es posible desarrollar un sistema de estimación de trayectoria a partir de imágenes RGB-D lo suficientemente rápido para atrapar objetos lanzados en tiro parabólico.
- Se puede construir un robot móvil omnidireccional que sirva como plataforma de pruebas

[1] https://commons.wikimedia.org/wiki/File:Detected-with-YOLO–Schreibtisch-mit-Objekten.jpg

- Se puede desarrollar un sistema de control de posición en el robot móvil para atrapar los objetos en movimiento detectados
- Es posible implementar tanto el sistema de visión como el de control utilizando la plataforma ROS (Robot Operating System)

1.4. Objetivos

- Desarrollar un sistema de visión artificial para estimar la trayectoria de un objeto en movimiento a partir de imágenes RGB-D
- Construir un robot móvil omnidireccional que sirva como plataforma de pruebas
- Desarrollar un sistema de control de posición y velocidad en el robot móvil para atrapar los objetos en movimiento detectados
- Implementar tanto el sistema de visión como el de control utilizando la plataforma ROS

1.5. Descripción de documento

En el capítulo 2 se exponen los conceptos básicos que serán usados en el resto del documento y se presenta el trabajo relacionado. El sistema de visión implementado se explica en el capítulo 3, el objetivo de este es estimar la caída de un objeto lanzado con un tiro parabólico utilizando segmentación por color. En el capítulo 4 se diseña una base móvil en la cual se monta el sistema de visión, además se muestran las ecuaciones necesarias para enviar la señal de control a los servomotores y calcular la odometría de la base. El sistema de control que permite llegar a un punto meta realimentando el error se explica en el capítulo 5. La integración del sistema de control y el de visión dividiendo tareas en nodos se observa en el capítulo 6, también se describe que es ROS y sus ventajas. En el capítulo 7 se describen los resultados obtenidos con la implementación propuesta, se incluyen cambios propuestos para mejorar la estimación del sistema de visión. Y finalmente, las conclusiones y propuestas para el desarrollo de sistemas similares o la mejora en el sistema desarrollado en esta tesis se observan en el capítulo 8.

Capítulo 2

Antecedentes

En el presente capítulo se introducen conceptos básicos. En la sección 2.1 se muestra el origen de la palabra robot y algunas definiciones, y por último se mencionan conceptos relevantes. Se explican algunos conceptos sobre la locomoción en la sección 2.1.1 y se explica por qué se prefiere utilizar ruedas. En la sección 2.1.2 se describen los componentes básicos que debe tener cualquier robot. Se define visión artificial en la sección 2.2, así mismo se habla del principio de funcionamiento de las cámaras. Finalmente, en la sección 2.3 se muestran proyectos que ya existen similares al que se va a trabajar en esta tesis.

2.1. Robots móviles autonómos

El término robot tiene su origen en la palabra checa "robota"que significa sirviente, esta expresión se volvió popular en Praga en el año de 1921 con la obra R.U.R. de Karel Capek. Sin embargo, en esa época la palabra robot hacía referencia a seres vivos humanoides, creados como sirvientes de los humanos pero con el tiempo y con ayuda de la ciencia ficción, dicho término acabó siendo usado para hablar de mecanismos que nos ayudan a hacer tareas [11].

Los robots son capaces de percibir el mundo real con ayuda de sensores y pueden modificar su entorno por medio de manipuladores [20]. Actualmente, existen robots industriales que son muy útiles para realizar tareas repetitivas y de gran precisión, y que igualmente son muy veloces; gracias a estas cualidades los teléfonos inteligentes y las computadoras portátiles son posibles. No obstante, estos robots tienen una gran desventaja que es, la falta de movilidad, pues cuentan con la capacidad de realizar tareas asombrosas pero solo en un espacio limitado [17].

La Valle [8] afirma que:

> "La robótica es la automatización de sistemas mecánicos, que tienen sensores, actuadores y capacidades de computación"

Estos sistemas también son llamados sistemas autónomos y algo fundamental

para que funcionen correctamente es tener algoritmos que conviertan tareas de alto nivel para los humanos en descripciones de movimiento de bajo nivel [8].

Por ejemplo, la instrucción de pasar un vaso de agua se puede dividir en tareas de bajo nivel como reconocer el vaso; llevar a cabo el movimiento para sostenerlo sin que se caiga; llevar el objeto del punto A donde se encuentra a un punto B el cual es el destino de interés.

Murphy [11] define a un robot móvil autónomo como:

> "Aquel mecanismo capaz de cambiar su posición en el espacio sin la intervención de un operador humano."

El primer gran reto en la creación de robots móviles autónomos es la locomoción, esto se refiere al mecanismo que se usará para producir un movimiento. Para plantear un mecanismo de locomoción se necesitan conocimientos de mecánica, cinemática, dinámica y teoría de control [17].

Los siguientes problemas son percepción, localización y navegación. La percepción se refiere a que el robot debe ser capaz de percibir el mundo real, este problema se resuelve con el uso de sensores, en esta tesis se usará la visión artificial para lograr este objetivo. La localización y navegación van de la mano, ya que se necesita saber en dónde se encuentra el robot para planear la trayectoria de un punto A a un punto B, por esta razón se necesitan conocimientos de algoritmos de computadora, teoría de la información, inteligencia artificial y teoría de probabilidad [8, 11, 17].

La teoría de control clásica se enfoca en estabilizar un sistema (mecánico, eléctrico, hidráulico, térmico, etc.) por medio de la retroalimentación del error, además el control óptimo puede ser útil si se desea minimizar el consumo de energía, tiempo de respuesta, sobrepaso u algún otro parámetro. Sin embargo, en la de teoría de control más moderna se usa el término *motion planning* o planeación de movimiento, lo cual se refiere a la construcción de entradas en un sistema dinámico que conduce de un estado inicial a un estado meta por medio de algoritmos [8].

2.1.1. Locomoción

Un robot necesita mecanismos de locomoción que le permitan moverse a través del ambiente, existe una gran variedad de diseños e investigaciones de robots que pueden caminar, saltar, volar, nadar, arrastrarse, deslizarse, la mayoría de estos han sido inspirados en animales. Una importante excepción a los mecanismos de locomoción inspirados en la naturaleza es la rueda pues se les atribuye su creación a los humanos, estas son extremadamente eficientes al moverse en superficies planas [17].

Profundizando en los tipos de movimiento en animales, existe física detrás de estos, por ejemplo, el arrastre de insectos como las orugas es producido por vibración longitudinal y se presenta una resistencia al movimiento debido a la fuerza a fricción. Otro ejemplo interesante es el deslizamiento de las serpientes el cual es producido por vibración transversal y también presenta resistencia que se debe a la fuerza de fricción. También existen animales que alcanzan grandes

velocidades como es el caso del guepardo debido a que sus patas se asemejan a un péndulo doble y el movimiento es producido por la oscilación de los péndulos, este movimiento presenta resistencia por la pérdida de energía cinética.[17]

Los mecanismos de locomoción biológicos han funcionado muy bien en terrenos difíciles pues pueden evadir hoyos y adaptarse al entorno fácilmente además tienen una gran ventaja manipulando objetos, a pesar de esto, es difícil replicar esta actividad ya que hasta ahora no se puede igualar el almacenamiento de energía, tiempo de respuesta y conversión de energía de los sistemas biológicos, aparte de que manufacturar sistemas similares actualmente es costoso, por estas razones es común usar ruedas o un número reducido de piernas articuladas en la construcción de robots [17].

Los robots con extremidades tienen ventaja sobre los que utilizan ruedas en terrenos difíciles, pero en general su locomoción requiere más grados de libertad y por lo tanto es más complejo de implementar que el uso de ruedas. Igualmente, en robots con piernas se tiene lograr un equilibrio en el cuerpo de estos ya que sus extremidades no están en contacto con la superficie todo el tiempo [17].

Las ruedas han sido el mecanismo de locomoción más común en robots móviles y vehículos fabricados por el hombre. Existen 4 tipos de ruedas estas son estándar, castor, sueca y esférica, con estas se pueden formar distintas geometrías, pero para seleccionar un tipo de rueda y geometría se debe considerar la controlabilidad, maniobrabilidad y estabilidad estática del sistema [17].

Al hablar de estabilidad estática en robots móviles el principal interés es garantizar que no exista tambaleo cuando el robot se encuentra en reposo, esto no suele ser un problema debido a que las ruedas tienen contacto todo el tiempo con la superficie. Se puede garantizar estabilidad con 3 ruedas, aunque es posible tenerla con 2 ruedas dependiendo de la geometría [17].

La maniobrabilidad se refiere a que tan sencillo es mover al robot en diferentes direcciones, esta alcanza su punto máximo cuando el robot es omnidireccional esto quiere decir que puede moverse en cualquier dirección en el plano, para lograr esto es común usar ruedas suecas o esféricas [17].

Controlabilidad en la teoría de control clásico es la propiedad de un sistema de ir de un estado A a un estado B en un tiempo finito, en el caso de los robots móviles el estado de interés es la posición y para llegar a la ubicación deseada se manipula la velocidad [16].

La controlabilidad suele tener una correlación inversa con la maniobrabilidad, es decir que entre más controlable es el sistema este es menos maniobrable y viceversa, por esta razón no existe una configuración ideal y se debe elegir la geometría y tipo de rueda de acuerdo a la aplicación del robot [17].

2.1.2. Componentes básicos

En el pasado se complicaba la creación de robots debido a lo lentas, pesadas y costosas que eran las computadoras con las que se les controlaba, afortunadamente hoy en día se tienen sistemas embebidos que son pequeños, ligeros, rápidos y no muy costosos [3].

El CPU (Unidad Central de Procesamiento) es la parte más importante de los sistemas embebidos, en este se encuentra la ALU (Unidad Lógica Aritmética) y la CU (Unidad de control), además varios microcontroladores y microprocesadores incluyen otros componentes en el CPU como SRAM (memoria RAM estática), contadores, etcétera [3].

La Unidad Lógica Aritmética se encarga de llevar a cabo las operaciones necesarias con los datos dependiendo de la complejidad de cada ALU estas operaciones pueden ser adición, sustracción, multiplicación, división y compuertas lógicas como or, not, and, xor entre otras [3].

La Unidad de Control se encarga de recorrer los registros en memoria cada que llega un pulso de reloj de forma que se tengan los datos de interés para ejecutar las operaciones necesarias en la Unidad Lógica Aritmética [3].

Dependiendo de la aplicación los sistemas embebidos pueden utilizar desde un simple programa sin sistema operativo hasta un sistema operativo complejo como es el caso de Linux. En esta tesis se utilizará un sistema operativo debido a que se usará la Jetson nano y está al tener una arquitectura similar a la de una computadora personal es difícil de utilizar si no se cuenta con un sistema operativo, sumado a esto se realizarán varias tareas de forma concurrente y para lograrlo también se necesita un sistema operativo (algunas de las tareas son: captura de imágenes RGB-D con la cámara, segmentación por color, control, estimación de trayectoria, etcétera) [3].

Los sensores son importantes para que el robot pueda percibir el mundo, aunque también se pueden incluir sensores que proporcionen información del propio robot, estos se clasifican como locales, globales, internos, externos, activos y pasivos [3].

Los sensores locales son aquellos sensores embebidos en el robot, mientras que los sensores globales son aquellos que transmiten datos al robot, pero que se encuentran montados fuera del robot. Los sensores internos se encargan de vigilar los estados internos del robot, mientras que los sensores externos perciben el entorno. Los sensores pasivos son aquellos que entregan la información del ambiente sin alterarlo, mientras que los sensores activos estimulan el ambiente para poder realizar mediciones, por ejemplo, sensor de onda, escáner. [3]

Como se desea atrapar un objeto realizando una segmentación por color, el sensor que será utilizado es una cámara de profundidad D345i de Intel, que entrega información de la imagen RGB, así como la distancia de profundidad a la que se encuentran los píxeles.

La tecnología RGB-D presenta el mismo funcionamiento que el famoso Kinect de Microsoft, esto consiste en un proyector y 2 cámaras, el proyector emite patrones de luz infrarroja y las cámaras ubicadas a la derecha e izquierda del proyector capturan la imagen y mandan la información a un procesador de imágenes de profundidad que obtiene los valores de profundidad de cada pixel con la correlación de las imágenes tomadas por ambas cámaras. La cámara RGB obtiene la información del color en la imagen y se alinea con la imagen de profundidad mientras que cámara D345i es un sensor global, externo y activo.[1]

[1] https://www.intelrealsense.com/wp-content/uploads/2022/05/Intel-RealSense-D400-

También se utiliza un encoder, este sensor es necesario para poder controlar los servomotores, en el caso del servomotor MX-12W usado en la construcción de este robot cuenta con un sensor de posición magnética AS5045 embebido, este tiene 4096 posiciones por revolución por lo que se cuenta con una resolución de 360°/4096=0.08789°. El encoder magnético usa el efecto Hall por lo que traduce el movimiento de imanes y la rotación del motor en señales de pulsos con las cuales se puede saber hacia dónde gira el motor, su velocidad y/o posición. El sensor embebido es inmune a los campos magnéticos externos. Es un sensor local, interno y pasivo. [2]

Como actuadores en el diseño se tienen servomotores Dynamixel MX-12W los cuales cuentan con un procesador ARM Cortex-M3. Los servomotores son motores de DC que contienen una parte electrónica encapsulada encargada del control de su posición o en este caso velocidad, a diferencia de la mayoría de los servomotores que son controlados por una señal PWM los servomotores se usan se comunican por medio del protocolo 1.0 mandando tramas de datos, en este se tienen 2 tipos de paquetes de datos, el paquete de instrucciones (es enviado del controlador a los servomotores) y el paquete de estados (es la respuesta de los servomotores al controlador). [3]

2.2. Conceptos básicos de visión artificial

De acuerdo con Solem [19] la visión artificial es:

> "La extracción automatizada de información a partir de imágenes."

Esta información obtenida a través de imágenes puede ser de muchos tipos como modelos 3D, reconocimiento de objetos, etc; en esta tesis se utilizará la visión para determinar la posición de un objeto [19].

La visión es un claro ejemplo de algo que es muy sencillo, para los humanos pero difícil de realizar para las máquinas pues los ojos humanos captan información y el cerebro la procesa de forma que se puede interpretar el mundo inconscientemente [19].

Existen 2 retos importantes para los robots con visión artificial. El primer reto es que los robots deben ser capaces de moverse en un entorno complejo, desordenado y cambiante. El segundo reto es que los robots deben funcionar de manera segura en presencia de personas. Por ejemplo, un robot de servicio en un hospital debe ser capaz de moverse, aunque haya muchas personas caminando en la zona y su movimiento debe ser seguro para estas [4].

De acuerdo a los retos planteados Corke[4] define a los robots con visión como:

> "Máquinas con una meta que puede sentir, planificar y actuar"

Series-Datasheet-April-2022.pdf

[2] https://www.micro-semiconductor.com/datasheet/37-AS5045-SS-EK-AB.pdf

[3] https://emanual.robotis.com/docs/en/dxl/mx/mx-12w/

El sensor que se usa en esta tesis para llevar a cabo la visión artificial es la cámara, su origen data del siglo XVI esta no contaba con algún lente y se le conocía como *cámara obscura* pues como se observa en la imagen 2.1 consistía en bloquear cualquier rayo de luz con una caja y hacer un pequeño orificio llamado *pinhole* en una de sus caras, al realizar esto la escena que se encuentra enfrente de la caja es proyectada de forma invertida en la cara opuesta [5].

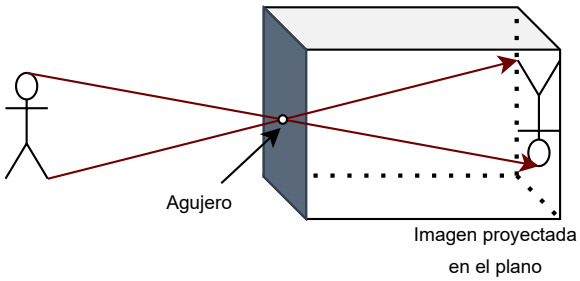


Figura 2.1: Cámara obscura

La imagen invertida es la representación en 2D de la escena 3D que es capturada por la cámara. Las cámaras modernas funcionan de la misma forma que la cámara obscura, pero tienen lentes complejos que sustituyen el agujero y son capaces de registrar la luz que se proyecta en el plano [5].

Otro concepto importante es el color, los humanos percibimos como color a la radiación electromagnética con una longitud de onda λ entre 400 y 700 [nm] [14].

Hablar de un color en específico es un problema debido a que existe una gran cantidad de colores, nombrar a todos es complicado por eso se requiere de un sistema estándar que permita clasificar e identificar los colores. Los espacios de color sirven para representar e identificar el color en los dispositivos de forma estandarizada.

Un espacio de color lineal es la representación de la mezcla de colores primarios, y es relativamente sencillo de replicar en dispositivos al mezclar luces de colores. De forma matemática los espacios de color lineal se pueden describir con la siguiente ecuación:

$$S(\lambda) = w_1 P_1 + w_2 P_2 + w_3 P_3$$

Donde P_1, P_2 y P_3 son los colores primarios en el espacio de color, $S(\lambda)$ es la fuente de color que se desea replicar, y w_1, w_2 y w_3 son los pesos necesarios para formar el color deseado.

Los espacios de color no lineales representan al color con propiedades importantes del color como el matiz (propiedad del color que varía al pasar de rojo a verde), la saturación (propiedad del color que varía al pasar de rojo a rosa) y el brillo (propiedad del color que cambia al pasar de negro a blanco).

2.3. Trabajo relacionado

En el Laboratorio de Biorrobótica de la Facultad de Ingeniería de la UNAM ya se han hecho trabajos de investigación para estimar movimiento, esta tesis servirá para enriquecer la información en dicha área, a continuación se describen algunos trabajos:

- Estimación de posición y velocidad de objetos para un robot humanoide:

  En dicha tesis se estimó la trayectoria de un balón utilizando el filtro de Kalman extendido (EFK, por sus siglas en inglés), se logró patear al balón con un robot bípedo y se cambió su dirección como se observa en la imagen 2.2 [6].

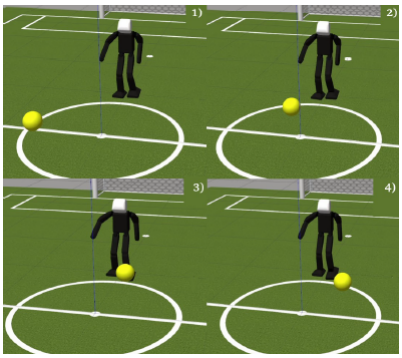


Figura 2.2: Se observa la secuencia en la que el robot estima la posición de la pelota y reacciona pateándola. Imagen extraída de la tesis [6].

- Estimación de otros vehículos utilizando el filtro de Kalman extendido:

  En este artículo se desarrolló un algoritmo que permitirá disminuir los riesgos de accidentes en automóviles, este consiste en calcular la velocidad de automóviles cercanos utilizando el filtro de Kalman extendido, a partir de las estimaciones el automóvil autónomo es capaz de tomar una decisión sobre rebasar, aumentar la velocidad o disminuirla [10].

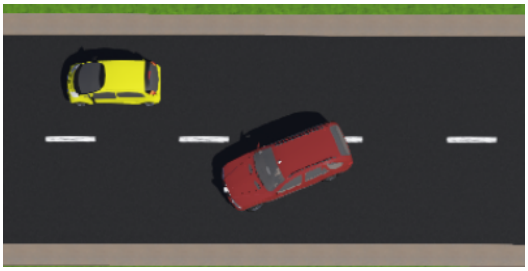


Figura 2.3: Se observa un automóvil autónomo cambiando de carril a partir de los datos estimados con el filtro de Kalman. Imagen extraída del artículo [10].

- Estimación de movimiento de personas en robots de servicio doméstico:

  En esta tesis se logró implementar el seguimiento de personas con robots utilizando técnicas de visión, para esto se probaron los siguientes métodos: seguimiento de piernas, reconocimiento de la vestimenta del operador y la combinación de ambas [1].

En estos trabajos se ha desarrollado la estimación de velocidad usando principalmente EKF y en esta tesis se busca probar un nuevo método mediante mínimos cuadrados.

En este capítulo se mostraron algunas definiciones sobre robots, se introdujeron conceptos fundamentales como lo es la locomoción, se habló de los componentes básicos para construir un robot, se dieron conceptos sobre la visión artificial y finalmente se mostraron trabajos relacionados con esta tesis.

En el siguiente capítulo serán necesarios los conceptos vistos sobre visión artificial ya que se desarrolla la propuesta para estimar la trayectoria utilizando segmentación por color y mínimos cuadrados.

Capítulo 3

Sistema de visión para estimación de trayectoria

El presente capítulo contiene las propuestas que se llevan a cabo para predecir en donde va a caer una pelota con un sistema de visión. En la sección 3.1 se plantea la forma en la que el robot podrá reconocer una pelota haciendo uso de la segmentación por color y eliminando el ruido que se puede presentar en las imágenes de la pelota con operaciones morfológicas, también se facilita el manejo de los datos utilizando transformaciones homogéneas. La sección 3.2 propone cómo encontrar una señal que indique que la pelota se encuentra en movimiento (a partir de esa señal se almacenan los datos) para esto se utilizan derivadas. Se muestra como encontrar una ecuación que describa el movimiento de la pelota en la sección 3.3, para lograr esto se necesita cierto número de muestras y se hace uso de la técnica mínimos cuadrados. En la sección 3.4 se describe como encontrar el punto al cual debe llegar la base móvil antes de que caiga la pelota para poder atraparla, para esto se utilizan ecuaciones cuadráticas y se hacen algunas consideraciones en las soluciones.

3.1. Detección mediante segmentación por color

Es necesario que el robot pueda distinguir la pelota del ambiente en el que se encuentra, una manera de llevar a cabo esta tarea es la segmentación por color.

Lo que se pretende al segmentar el color, es tener un parámetro que indique si la información de cierto píxel tomado por la cámara forma parte de la pelota o no.

Con este método se descarta toda la información que no se encuentre dentro de un rango de colores dado y posteriormente se puede realizar un promedio de la distancia que mide la cámara de profundidad en cada píxel.

Se asume que la distancia promedio es el centroide de la pelota de acuerdo a un sistema de referencia x,y,z.

3.1.1. Los espacios de color RGB y HSV

El color en cada píxel de una imagen es representado normalmente por el espacio de color lineal RGB, el cual significa Red, Green, Blue, en español rojo (λ=645.16 [nm]), verde (λ=526.32 [nm]) y azul (λ=444.44 [nm]) siendo estos los colores primarios del espacio [5].

Otra forma de ver al espacio de color RGB es como un sistema de color aditivo debido a que los colores se forman agregando estos 3 colores con mayor o menor intensidad al color negro [14].

El espacio de color RGB se encuentra normalmente en formato hexadecimal, en donde 0xFF corresponde al máximo valor de cada color (rojo, verde y azul). Por ejemplo, si se quiere representar un color rojo muy intenso la representación hexadecimal de ese color es 0xFF0000 debido a que 0xFF corresponde el máximo valor de rojo que puede tener un píxel y 0x0000 indica la ausencia de azul y verde.

La información que arroja la cámara será representada en el espacio de color RGB, sin embargo, no sirve para segmentar colores y se puede observar con un ejemplo sencillo.

Se requiere segmentar un objeto de color blanco y los píxeles para ese objeto van de 0xE6E6E6 hasta 0xFFFFFF, en el espacio de color RGB. Al realizar la segmentación también se deja pasar colores rosados como 0xFFE6E6, verdes como 0xE6E6FF, entre muchas otras combinaciones más del espacio RGB (como se observa en la tabla 3.1) que no sirven y es muy probable encontrar en el ambiente.

Una alternativa a este problema es el espacio de color no lineal HSV. El espacio de color HSV, significa Hue, Saturation, Value, en español Matiz, Saturación, Valor. Este espacio de color es una transformación no lineal del espacio de color RGB. Está transformación es útil cuando se desea trabajar con tonos específicos.

El valor de H va de 0 a 2π.

El valor de S va de 0 a 1.

El valor de V va de 0 a 255.

Para transformar de RGB a HSV es necesario definir los parámetros Min y Max. Min es el menor valor de los píxeles RGB, y Max el mayor valor de los píxeles. Las transformaciones de RGB a HSV se definen de la siguiente manera:

Para el cálculo de H se tienen 5 casos

$$H = \begin{cases} 0 & Max = Min \quad (3.1) \\ \frac{\pi}{3}(\frac{G-B}{Max-Min}) & Max = R \text{ y } G \geq B \quad (3.2) \\ 2\pi + \frac{\pi}{3}(\frac{G-B}{Max-Min}) & Max = R \text{ y } G < B \quad (3.3) \\ \frac{\pi}{3} + \frac{2\pi}{3}(\frac{B-R}{Max-Min}) & Max = G \quad (3.4) \\ \frac{\pi}{3} + \frac{2\pi}{3}(\frac{R-G}{Max-Min}) & Max = B \quad (3.5) \end{cases}$$

Para el cálculo de S se tienen 2 casos

$$S = \begin{cases} 0 & Max = 0 \quad (3.6) \\ \frac{Max - Min}{Max} & \text{Cualquier otro caso} \quad (3.7) \end{cases}$$

Para el cálculo de V

$$V = Max$$

Ejemplo: Para el color 0xFF0056, Max=0xFF y Min=0x00. En decimal R=255, G=0, B=86 Max=R, $G < B$. Por lo tanto

$$H = 2\pi + \frac{\pi}{3}(\frac{0 - 86}{255 - 0}) = 5.93$$

Como $Max \neq 0$

$$S = \frac{255 - 0}{255} = 1$$

$$V = Max = 255$$

Para el mismo problema en el que se requiere segmentar un objeto de color blanco y cuyos píxeles están en el rango de 0xE6E6E6 hasta 0xFFFFFF, se hace una conversión en el espacio HSV.

En el espacio RGB se tienen los siguientes valores:

$$R_1, G_1, B_1 = 255, 255, 255$$

$$R_2, G_2, B_2 = 230, 230, 230$$

Al pasar de RGB a HSV se tienen los siguientes valores:

$$H_1, S_1, V_1 = 0, 0, 255$$

$$H_2, S_2, V_2 = 0, 0, 230$$

Al segmentar de H_1, S_1, V_1 a H_2, S_2, V_2 como se observa en tabla 3.1, se puede ver que la segmentación es más selectiva en el espacio HSV, dejando pasar menos colores.

		Se encuentra dentro del intervalo	
Color	**Color en RGB**	**En el espacio RGB**	**En el espacio HSV**
	0xE6E6E6	✓	✓
	0xEEEEEE	✓	✓
	0xF8F8F8	✓	✓
	0xFFFFFF	✓	✓
	0xE6E6FF	✓	✗
	0xE6FFE6	✓	✗
	0xFFE6E6	✓	✗
	0xF1E6E6	✓	✗
	0xFFEFEF	✓	✗

Tabla 3.1: Comparación entre la segmentación usando los espacios RGB y HSV. Se observa que el espacio HSV es más selectivo.

Al segmentar se busca eliminar toda la información que no corresponde al color de interés. El objetivo de segmentar, es reconocer un objeto en el ambiente, en este caso el objeto es una pelota y con el fin de evitar que algún otro objeto en el ambiente pueda tener píxeles que la segmentación en el espacio RGB no es capaz de eliminar, se prefiere realizar una segmentación en el espacio HSV.

3.1.2. Operadores morfológicos

Con el fin de eliminar pequeñas manchas de píxeles, que se encuentren en un rango dado de colores en la segmentación y no correspondan a la pelota podemos utilizar la operación morfológica erosión, como se muestra en la Figura 3.1.

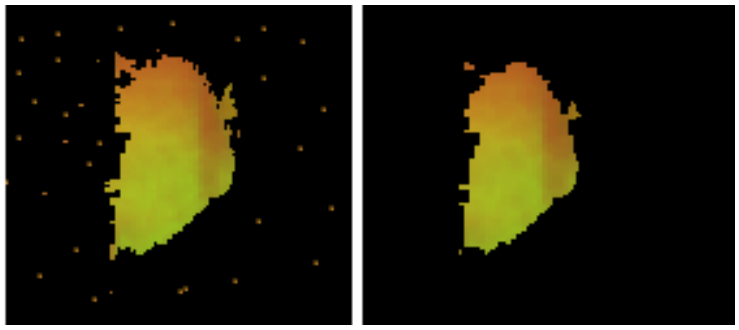


Figura 3.1: A la izquierda imagen con ruido, a la derecha la imagen después de aplicar erosión

Las 2 operaciones de morfología binaria básicas son la dilatación y la erosión. Estas operaciones son contrarias, la dilatación aumenta una región, mientras que la erosión hace que la región sea más estrecha o pequeña, respetando la forma

de dicha región. Ambas operaciones se apoyan de una matriz binaria la cual se conoce como elemento estructurante o kernel y este cuantifica que tan estrecha será la erosión o que tan ancha será la dilatación.

Antes de definir estas operaciones es necesario entender el concepto de translación en imágenes binarias. Además, es común tomar como referencia la posición del píxel central de una estructura simétrica para saber su posición, aunque se puede escoger cualquier otro píxel.

La traslación de X_t de un conjunto de píxeles X por un vector de posición t se define matemáticamente como:

$$X_t = x + t | x \in X$$

En donde X_t es la estructura X con una traslación en t y $t = (\delta_r, \delta_c)$, δ_r es la cantidad de renglones y δ_c es la cantidad de columnas en los que se moverá el objeto.[14]

Ejemplo:

Se tiene a la estructura:

$$B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.8}$$

La estructura B se encuentra contenida en la imagen binaria I.

$$I = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.9}$$

De acuerdo a la convención el origen de la estructura B, se encuentra en las coordenadas (1,1). Y si realizamos una traslación de la estructura B en t=(5,5)

$$I_t = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.10}$$

El nuevo origen de la estructura B es (6,6).

Dilatación

La operación morfológica dilatación, que se denota como $\oplus$ se define como:

$$A \oplus B = \bigcup_{b \in B} A_b$$

En donde A es una imagen binaria y B es el kernel o elemento estructurante.

Lo que se hace al aplicar la dilatación es sobreponer el kernel B en la imagen A y aplicar una traslación por todo el espacio de A. En cada píxel en el que se encuentre el kernel B, el valor de $A \oplus B$ para ese píxel es el mayor valor que encierra el kernel B sobre la imagen A.

Ejemplo:

Sea la imagen binaria A, y el kernel B previamente definido. Se observa que la imagen A es un cuadrado que contiene pequeños píxeles de ruido.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.11}$$

Para el píxel (3,3), como se observa en la imagen binaria 3.12, el mayor valor que encierra el kernel es 1, por lo tanto $(A \oplus B)(3,3)=1$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.12}$$

Para en el píxel (8,8) como se observa en la imagen binaria 3.13, el mayor

valor que encierra el kernel, también es 1, por lo tanto $(A \ominus B)(8,8)=1$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.13}$$

Siguiendo este procedimiento con todos los píxeles

$$A \oplus B = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3.14}$$

Se puede observar que en la imagen resultante de $A \oplus B$ es difícil observar el cuadrado, ya que la operación $\oplus$ amplificó el cuadrado al igual que los pequeños píxeles de ruido.

Erosión

La operación morfológica erosión, que se denota como $\ominus$ se define como:

$$A \ominus B = \{b | b + a \in B \forall a \in A\}$$

En donde A es una imagen binaria y B es el kernel o elemento estructurante.

Lo que se hace al aplicar la erosión es sobreponer el kernel B en la imagen A y aplicar una traslación por todo el espacio de A. En cada píxel en el que se encuentre el kernel B, el valor de $A \ominus B$ para ese píxel es el menor valor que encierra el kernel B sobre la imagen A.

Ejemplo:

Sea la imagen binaria A previamente definida, y el kernel B previamente definido.

Para el píxel (4,3), como se observa en la imagen binaria 3.15 el menor valor

que encierra el kernel es 0, por lo tanto $(A \ominus B)(4,3)=0$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.15}$$

Mientras en el píxel (5,5), como se observa en la imagen binaria 3.16, el menor valor que encierra el kernel es 1, por lo tanto $(A \ominus B)(5,5)=1$.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.16}$$

Siguiendo este procedimiento con todos los píxeles

$$A \ominus B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.17}$$

Se puede observar que se conserva la imagen del cuadrado que se tenía en la imagen A que se observa en la ecuación 3.11, la diferencia es que este cuadrado es más pequeño, de igual forma se observa que se eliminaron pequeños píxeles de ruido que se encontraban en la imagen A original.

3.1.3. Transformaciones Homogéneas

Con el fin de observar mejor el medio en donde se encuentra el robot y a la vez tener mayor alcance en las mediciones se planea inclinar la cámara ciertos grados. Pero se busca tener las mediciones de acuerdo a un sistema de referencia cuyos ejes sean paralelos al sistema de referencia de la base.

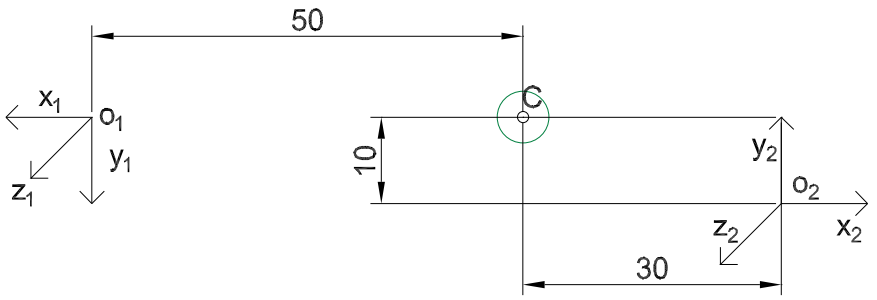


Figura 3.2: Centroide visto desde sistemas de referencia distintos

Un sistema de referencia cuantifica magnitudes físicas, respecto a un punto de referencia. Este sistema es importante al realizar mediciones, ya que para un sistema de referencia la pelota puede estar a 1[m], mientras que para otro sistema puede estar a 5[m] (ambas mediciones pueden ser correctas dependiendo del sistema de referencia) y la regresión nos dará diferentes parámetros dependiendo del sistema mencionado. Esto se puede ver en la figura 3.2, ya que se tiene un centroide visto por 2 sistemas de referencia diferentes.

Si se denota C como el centroide del objeto, ${}^1C$ es el centroide visto desde el sistema de referencia $x_1y_1z_1$ y ${}^2C$ es el centroide visto desde el sistema de referencia $x_2y_2z_2$.

$$ {}^1C = (-50, 0, 0) \tag{3.18} $$

$$ {}^2C = (-30, 10, 0) \tag{3.19} $$

En las ecuaciones 3.18 y 3.19 se tienen diferentes coordenadas para un mismo punto y ambas son correctas respecto a su sistema de referencia.

En la figura 3.2 se tienen las mediciones y puede ser sencillo cambiar de un sistema de referencia a otro, pero esto se complica al tener un objeto en movimiento en el mundo real. No obstante, existen las transformaciones homogéneas, las cuales son una herramienta matemática que facilita el cambiar de un sistema de referencia a otro.

Antes definir las transformaciones homogéneas es importante definir a las matrices de rotación.

$$ R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{3.20} $$

Las 3 columnas de la matriz de rotación R corresponde a los vectores unitarios de los ejes x, y y z. Las matrices de rotación deben cumplir las siguientes condiciones:

- Condición de norma unitaria: Los ejes x, y y z son vectores unitarios

$$r_{11}^2 + r_{12}^2 + r_{13}^2 = 1$$
$$r_{21}^2 + r_{22}^2 + r_{23}^2 = 1$$
$$r_{31}^2 + r_{32}^2 + r_{33}^2 = 1$$

- Condición de norma unitaria: Los ejes x, y y z son ortogonales, al denotar con $(.)$ el producto interno de vectores $x.y = 0$, $y.z = 0$, $x.z = 0$

$$r_{11}r_{12} + r_{21}r_{22} + r_{31}r_{32} = 0$$
$$r_{12}r_{13} + r_{22}r_{23} + r_{32}r_{33} = 0$$
$$r_{11}r_{13} + r_{21}r_{23} + r_{31}r_{33} = 0$$

Se dice que las matrices de rotación son un grupo y sean A, B, C matrices de rotación, se satisfacen las siguientes propiedades:

- Cerradura: La matriz AB sigue siendo una matriz de rotación que pertenece al grupo.
- Asociatividad: El orden en el que se multiplican las matrices no afecta el producto $(AB)C = A(BC)$
- Existencia del elemento identidad: En el grupo existe un elemento I tal que $AI = IA = A$
- Existencia del elemento inverso: En el grupo existe un elemento A^{-1} tal que $AA^{-1} = A^{-1}A = I$. En el caso particular de las matrices de rotación la inversa de una matriz A es A^T.

Una de las aplicaciones de las matrices de rotación es rotar un vector o un sistema de referencia, las ecuaciones 3.21,3.22 y 3.23 definen una rotación en los ejes x, y o z respectivamente.[9]

$$Rot(x,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{3.21}$$

$$Rot(y,\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{3.22}$$

$$Rot(z,\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.23}$$

Para el ejercicio de la figura 3.2 se observa que aplicando una rotación en el eje z se puede obtener una relación entre la orientación del sistema $x_1y_1z_1$ y el sistema $x_2y_2z_2$, siendo 180° el ángulo entre los ejes x_1 y x_2, esto se observa de forma gráfica en la figura 3.3 (esta figura se muestra con fines didácticos ya que los sistemas no comparten origen) y se expresa matemáticamente con la ecuación 3.24.

La inversa de la matriz $Rot(z, 180°)$ es la misma por lo que se puede pasar de la orientación de un sistema a otro con la misma matriz.

$$Rot(z, 180°) = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.24}$$

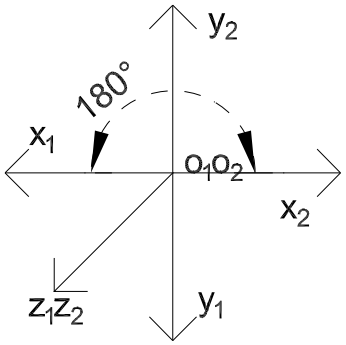


Figura 3.3: Relación entre la orientación de los sistemas

A pesar de tener una relación entre la orientación de ambos sistemas, aún no existe una matriz que cumpla la ecuación 3.25 debido a que además de que los sistemas tienen una orientación diferente como se observa en la figura 3.3 los sistemas realmente no comparten el mismo origen $o_1 \neq o_2$.

$$^1C = X^2C \tag{3.25}$$

Las transformaciones homogéneas además de cambiar la orientación de un sistema también pueden aplicar una traslación entre los orígenes del sistema. Una matriz homogénea está compuesta por una matriz de rotación R y un vector $\vec{p} = [p_1, p_2, p_3]^T$.

$$H = \begin{bmatrix} R & \vec{p} \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.26}$$

En el caso de la matriz Homogénea que se usa para transformar del sistema de referencia 1 al 2. El vector $^1\vec{p}_2$ es el origen o_2 visto desde el sistema $x_1y_1z_1$.

$$^1\vec{p}_2 = \begin{bmatrix} -80 \\ 10 \\ 0 \end{bmatrix}$$

$$ {}^{1}H_2 = \begin{bmatrix} {}^{1}R_2 & {}^{1}\vec{p}_2 \\ 0_{1x3} & 1 \end{bmatrix} \tag{3.27} $$

Para el ejemplo de la figura 3.2 la matriz homogénea ${}^{1}H_2$ es:

$$ {}^{1}H_2 = \begin{bmatrix} -1 & 0 & 0 & -80 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.28} $$

${}^{1}H_2$ es una matriz de dimensión 4x4, y el vector ${}^{1}C$ un vector de 3x1, para llevar a cabo la multiplicación entre la matriz ${}^{1}H_2$ y el vector ${}^{1}C$ se define un nuevo vector ${}^{1}C*$ el cual es el vector ${}^{1}C$ pero se agrega un 1 al vector de manera que sea de dimensión 4x1. Se hace lo mismo con el vector ${}^{2}C$.

$$ {}^{1}C* = \begin{bmatrix} {}^{1}C \\ 1 \end{bmatrix} = \begin{bmatrix} -50 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.29} $$

$$ {}^{2}C* = \begin{bmatrix} {}^{2}C \\ 1 \end{bmatrix} = \begin{bmatrix} -30 \\ 10 \\ 0 \\ 1 \end{bmatrix} \tag{3.30} $$

De manera que se cumple la ecuación ${}^{2}C* = {}^{1}H_2^1C*$ como se observa en la ecuación 3.31

$$ {}^{2}C* = \begin{bmatrix} -1 & 0 & 0 & -80 \\ 0 & -1 & 0 & 10 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -50 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -30 \\ 10 \\ 0 \\ 1 \end{bmatrix} \tag{3.31} $$

Con la ecuación 3.31 de comprueba que con las transformaciones homogéneas podemos obtener las coordenadas de un mismo punto visto desde sistemas de referencia diferentes con una sencilla multiplicación de matrices.

3.2. Detección de movimiento

Cuando un objeto se mueve, este tiene cierta velocidad. La velocidad es un vector con componentes en x, y, z, como se observa en la ecuación 3.32. Por lo tanto, para detectar que la pelota se está moviendo se puede calcular la velocidad, y cuando la magnitud de la velocidad sea mayor a un umbral 3.33, se dice que la pelota está en movimiento.

$$ \vec{v} = (v_x, v_y, v_z) \tag{3.32} $$

$$|\vec{v}| > u \tag{3.33}$$

La velocidad se define como la derivada de la posición respecto al tiempo. Para el caso particular en el que el vector $\vec{v}$ solo tiene una componente en x (v_x), se puede definir $x(t)$ como una función que define la posición de un objeto en el tiempo t en el eje x y la ecuación 3.34 es la definición matemática de la velocidad v_x.

$$v_x = \frac{dx(t)}{dt} = \lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} \tag{3.34}$$

La ecuación 3.34 funciona para una posición continua en el tiempo, es decir que en todo tiempo t se conoce la posición, pero la cámara RGB-D captura la posición cada $\frac{1}{30}[s]$ por lo tanto se trata de un caso discreto y se puede aproximar la derivada a la ecuación 3.35 en donde x_i es la posición actual, x_{i-1} la posición anterior y Δt el tiempo de muestreo en este caso $1/30[s]$

$$v_x \approx \frac{x_i - x_{i-1}}{\Delta t} \tag{3.35}$$

De forma general $\vec{v}$ se puede aproximar a la ecuación 3.36

$$\vec{v} \approx (\frac{x_i - x_{i-1}}{\Delta t}, \frac{y_i - y_{i-1}}{\Delta t}, \frac{z_i - z_{i-1}}{\Delta t}) \tag{3.36}$$

Sustituyendo la ecuación 3.36 en la ecuación 3.33 obtenemos la ecuación

$$\frac{\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}}{\Delta t} > u \tag{3.37}$$

Como se mencionó antes Δt es una constante y para simplificar la ecuación 3.37 se pueden multiplicar ambos lados de la ecuación por Δt y sustituyendo $u\Delta t$ como α, se obtiene la ecuación 3.38

$$\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2} > \alpha \tag{3.38}$$

En donde el valor α es un parámetro de diseño.

La ecuación $\sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}$ es similar a la distancia entre 2 puntos por lo que se denotará como d como se indica en la ecuación 3.39

$$d = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2} \tag{3.39}$$

En síntesis, se puede afirmar que la pelota está en movimiento si ha recorrido una distancia mayor que un umbral en un periodo de muestreo.

Cuando se detecta que la pelota se encuentra en movimiento, se empezarán a guardan los datos de la posición de la pelota, los cuales se utilizarán para estimar las coordenadas donde va a caer la pelota.

Ejemplo:

En la medición 1, se tiene:

$$m_1 = (0.1, 0.2, 0.3)[m]$$

Mientras que en la medición 2, se tiene:

$$m_2 = (0.1, 0.2, 0.4)[m]$$

Y considerando un umbral de aproximadamente el radio de la pelota, el cual es 0.04[m]

$$\alpha = 0.04[m]$$

Sustituyendo los datos de las mediciones en la ecuación 3.39:

$$d = \sqrt{(m_{21} - m_{11})^2 + (m_{22} - m_{12})^2 + (m_{23} - m_{13})^2}$$

$$d = 0.1[m]$$

Entonces se tiene que $d > \alpha$, cumpliendo la condición de la ecuación 3.38; esto significa que la pelota está en movimiento.

Ahora bien, cuando se manipula la pelota no siempre significa que ya se lanzó, esto porque existe la posibilidad de que se busque un mejor punto para lanzar la pelota, lo cual es un problema al capturar los datos ya que si llevamos a cabo la regresión lineal con datos que no describen una parábola, será más complicado acertar en el punto en donde va a caer la pelota.

Para resolver este problema también se puede utilizar la derivada, en este caso solo se analiza la lectura de la pelota en el eje z (el cual contiene información sobre la altura).

Partiendo de la suposición de que la pelota en las primeras mediciones va a subir para después comenzar a caer, se espera que la medición en z de las primeras mediciones que describen una parábola vaya en aumento, esto es que la medición anterior sea menor que la medición actual de la pelota en el eje z3.40.

$$z_{actual} > z_{anterior} \tag{3.40}$$

Para el mismo ejemplo, se tiene que las mediciones en z, son:

$$z_1 = 0.3$$

$$z_2 = 0.4$$

Se cumple que $z_2 > z_1$ y a la vez esto cumple con la ecuación 3.40. Por lo que además de estar en movimiento la pelota se encuentra subiendo.

Se puede incluir en el algoritmo que se debe cumplir la ecuación 3.40 y la condición 3.33 n veces, una vez que se cumplan estás condiciones se pude afirmar que el movimiento de la pelota está describiendo una parábola, entonces se almacenan datos y se procede a calcular donde va a caer.

3.3. Estimación mediante mínimos cuadrados

La técnica de estimación denominada como mínimos cuadrados se usa para estimar parámetros que no son aleatorios, buscando una solución óptima a partir de minimizar la suma de los errores al cuadrado entre las mediciones y la función observada del parámetro que se desea estimar.

Para el caso lineal y vectorial de mínimos cuadrados el problema es estimar el vector x modelado como una constante desconocida que se quiere obtener a partir de las mediciones en el vector z, y con los errores de medición w como se observa en la ecuación 3.41, en donde la matriz H relaciona al vector z con el vector x.

$$z(i) = H(i)x + w(i) \qquad i = 1, ..., k \tag{3.41}$$

La ecuación 3.42 modela el error cuadrático multiplicado por los inversos de la matriz R, en donde R es una matriz diagonal positiva definida.

$$J(k) = \sum_{i=1}^{k} [z(i) - H(i)x]' R(i)^{-1} [z(i) - H(i)x] \tag{3.42}$$

Al reescribir la ecuación 3.42 de forma matricial se obtiene la ecuación 3.43.

$$J(k) = [z^k - H^k x]^T (R^k)^{-1} [z^k - H^k x] \tag{3.43}$$

En donde el vector 3.44 tiene una dimensión de $kn_z x1$, la matriz 3.45 tiene una dimensión $kn_z xn_x$ y la matriz 3.46 tiene una dimensión $n_z xn_z$.

$$z^k = \begin{bmatrix} z(1) \\ \vdots \\ z(k) \end{bmatrix} \tag{3.44}$$

$$H^k = \begin{bmatrix} H(1) \\ \vdots \\ H(k) \end{bmatrix} \tag{3.45}$$

$$R^k = \begin{bmatrix} R(1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R(k) \end{bmatrix} = diag[R(i)] \tag{3.46}$$

Para minimizar el error cuadrático de la ecuación 3.43 se obtiene el gradiente respecto a x y se iguala a 0, esto da como resultado la ecuación 3.47.[2]

$$\nabla_x J(k) = -2H^{k'} (R^k)^{-1} [z^k - H^k x] = 0 \tag{3.47}$$

Y al despejar x se obtiene la ecuación

$$\hat{x}(k) = [H^{k'} (R^k)^{-1} H^k]^{-1} H^{k'} (R^k)^{-1} z^k \tag{3.48}$$

Para estimar x es necesario tener k datos, debido a que este proceso se lleva a cabo de forma simultánea.

Estimación de polinomios

Los mínimos cuadrados también pueden estimar un polinomio. Si se tiene un polinomio como el de la ecuación 3.49

$$z(i) = x_0 + x_1 t_i + w_i \quad i = 1... \tag{3.49}$$

Si se desea estimar el parámetro $x = \begin{bmatrix} x_0 & x_1 \end{bmatrix}^T$ se debe modelar el polinomio como la ecuación 3.41 en donde

$$H(i) = \begin{bmatrix} 1 & t_i \end{bmatrix}$$

Si los ruidos $w(i)$ son independientes, idénticamente distribuidos, con media cero y varianza σ^2 la matriz de covarianza del ruido de las mediciones es

$$I\sigma^2$$

Si se busca disminuir la covarianza del ruido se puede sustituir R^k como:

$$R^K = I\sigma^2$$

Al sustituir R^k en la ecuación 3.48 se obtiene la ecuación

$$\hat{x} = [H^{k'}(I\sigma^2)^{-1}H^k]^{-1}H^{k'}(I\sigma^2)^{-1}z^k = [H^{k'}H^k]^{-1}H^{k'}z^k \tag{3.50}$$

Al lanzar la pelota con un tiro parabólico, la pelota seguirá una trayectoria como la que se observa en las figuras 3.4, 3.5 y 3.6.

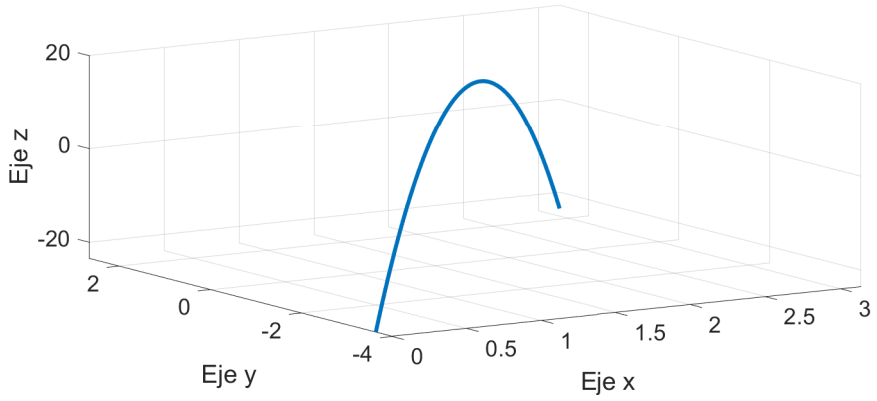


Figura 3.4: Parábola en 3D

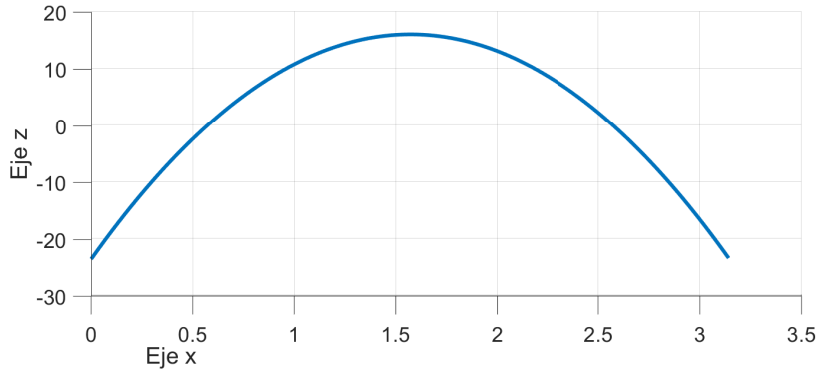


Figura 3.5: Parábola vista en el plano XZ

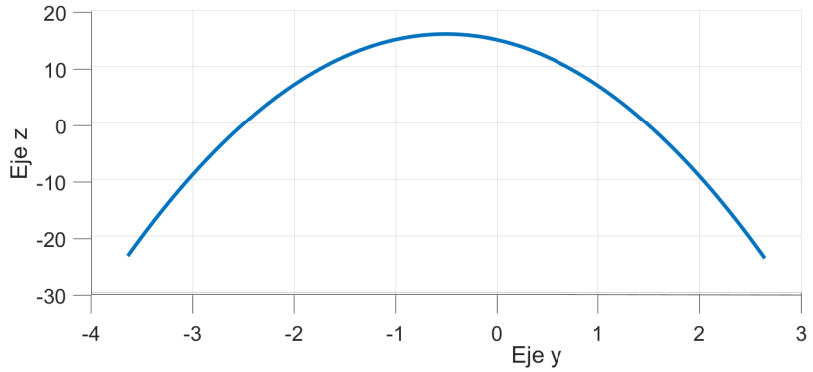


Figura 3.6: Parábola vista en el plano YZ

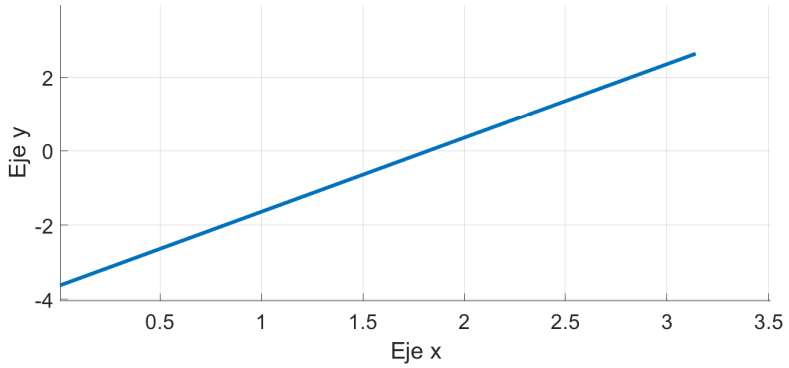


Figura 3.7: Parábola vista en el plano XY

Para facilitar el cálculo de la estimación de la trayectoria se puede dividir el problema en 2 como se observa en las figuras 3.5 y 3.6. Por lo tanto, se calcula una parábola con Z como variable dependiente, X como variable independiente 3.51 y otra parábola con Z como variable dependiente, Y como variable independiente3.52.

$$z = f(x) \tag{3.51}$$

$$z = f(y) \tag{3.52}$$

Una parábola puede ser descrita como un polinomio de segundo grado por lo que se puede estimar el movimiento de la pelota como las ecuaciones 3.53 y 3.54

$$z = a_2x^2 + a_1x + a_0 \tag{3.53}$$

$$z = b_2y^2 + b_1y + b_0 \tag{3.54}$$

Los modelos de regresión polinomial se pueden expresar de forma matricial como $\vec{z} = X\vec{a} + \vec{\epsilon}$ (equivalente a la ecuación 3.55) y $\vec{z} = Y\vec{b} + \vec{\xi}$ (equivalente a la ecuación 3.56).

En donde $\vec{z}$ corresponde al vector de muestras en el eje z, $\vec{b}$ corresponde al vector de coeficientes que mejor se aproxima al polinomio de segundo grado que pasa por todas las muestras, X es una matriz formada las muestras en el eje x elevadas a las potencias 0, 1 y 2, $\vec{\epsilon}$ es el vector que corresponde a los errores aleatorios, $\vec{a}$ corresponde al vector de coeficientes que mejor se aproxima al polinomio de segundo grado que pasa por todas las muestras, Y es una matriz formada las muestras en el eje y elevadas a las potencias 0, 1 y 2, $\vec{\xi}$ es el vector que corresponde a los errores aleatorios.

$$\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} + \begin{bmatrix} \epsilon_0 \\ \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix} \tag{3.55}$$

$$\begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} 1 & y_0 & y_0^2 \\ 1 & y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ 1 & y_n & y_n^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} + \begin{bmatrix} \xi_0 \\ \xi_1 \\ \vdots \\ \xi_n \end{bmatrix} \tag{3.56}$$

Se puede estimar los coeficientes a_0, a_1, a_2, b_0, b_1 y b_2 por medio la técnica de mínimos cuadrados con las ecuaciones 3.57 y 3.58.

$$\hat{\vec{a}} = (X^TX)^{-1}X^T\vec{z} \tag{3.57}$$

$$\hat{\vec{b}} = (Y^TY)^{-1}Y^T\vec{z} \tag{3.58}$$

Cada vez que se toma una muestra se estiman los coeficientes de ambas parábolas.

Ejemplo

Utilizando el método de mínimos cuadrados y los datos de la tabla 3.2 se desea obtener una ecuación que describa el movimiento de la pelota.

Las muestras de la tabla 3.2 corresponden la trayectoria completa que siguió la pelota antes de caer, para facilitar los cálculos solo se tomarán las primeras 10 muestras de la tabla.

	Mediciones		
i	x_i	y_i	z_i
0	0.261786501	2.395616782	0.003972712
1	0.258365223	2.363318937	0.022714964
2	0.242286277	2.172171154	0.128651038
3	0.226606242	2.072297687	0.199931953
4	0.208002525	1.924901213	0.247783049
5	0.197472326	1.787382221	0.300532821
6	0.183640629	1.686395098	0.33807646
7	0.168293156	1.541400071	0.351297577
8	0.153126129	1.442651677	0.363741551
9	0.131036427	1.321985997	0.369754369
10	0.114408759	1.196349117	0.366618607
11	0.101407212	1.085494503	0.352937081
12	0.082874707	0.96709963	0.323938836
13	0.065921862	0.854086021	0.290228885
14	0.048539358	0.736106093	0.244104621
15	0.036989583	0.624935638	0.185443996
16	0.028124128	0.511037884	0.122719311
17	0.016190753	0.39866669	0.033626212

Tabla 3.2: Muestras de posición tomadas por la cámara RGB-D, los datos se expresan en metros

$$X = \begin{bmatrix} 1 & 0.261786501 & 0.261786501^2 \\ 1 & 0.258365223 & 0.258365223^2 \\ 1 & 0.242286277 & 0.242286277^2 \\ 1 & 0.226606242 & 0.226606242^2 \\ 1 & 0.208002525 & 0.208002525^2 \\ 1 & 0.197472326 & 0.197472326^2 \\ 1 & 0.183640629 & 0.183640629^2 \\ 1 & 0.168293156 & 0.168293156^2 \\ 1 & 0.153126129 & 0.153126129^2 \\ 1 & 0.131036427 & 0.131036427^2 \end{bmatrix} \tag{3.59}$$

$$Y = \begin{bmatrix} 1 & 2.395616782 & 2.395616782^2 \\ 1 & 2.363318937 & 2.363318937^2 \\ 1 & 2.172171154 & 2.172171154^2 \\ 1 & 2.072297687 & 2.072297687^2 \\ 1 & 1.924901213 & 1.924901213^2 \\ 1 & 1.787382221 & 1.787382221^2 \\ 1 & 1.686395098 & 1.686395098^2 \\ 1 & 1.541400071 & 1.541400071^2 \\ 1 & 1.442651677 & 1.442651677^2 \\ 1 & 1.321985997 & 1.321985997^2 \end{bmatrix} \tag{3.60}$$

$$\vec{z} = \begin{bmatrix} 0.003972712 \\ 0.022714964 \\ 0.128651038 \\ 0.199931953 \\ 0.247783049 \\ 0.300532821 \\ 0.33807646 \\ 0.351297577 \\ 0.363741551 \\ 0.369754369 \end{bmatrix} \tag{3.61}$$

Al sustituir los valores de la tabla 3.2 en las matrices de las ecuaciones 3.55, 3.56 se obtiene el vector 3.61 y las matrices 3.60 y 3.59.

Y al realizar las operaciones de las ecuaciones 3.55 y 3.56 se obtienen los siguientes valores:

$$\vec{a} = \begin{bmatrix} -0.1826 \\ 7.6538 \\ -26.4708 \end{bmatrix} \tag{3.62}$$

$$\vec{b} = \begin{bmatrix} -0.2015 \\ 0.8599 \\ -0.3237 \end{bmatrix} \tag{3.63}$$

Las ecuaciones 3.64, 3.65 describen el movimiento de la pelota de acuerdo a la estimación por mínimos cuadrados usando las 10 primeras muestras. Se pueden observar las muestras y las estimaciones en las gráficas 3.8, 3.9

$$z = -26.4708x^2 + 7.6538x - 0.1826 \tag{3.64}$$

$$z = -0.3237y^2 + 0.8599y - 0.2015 \tag{3.65}$$

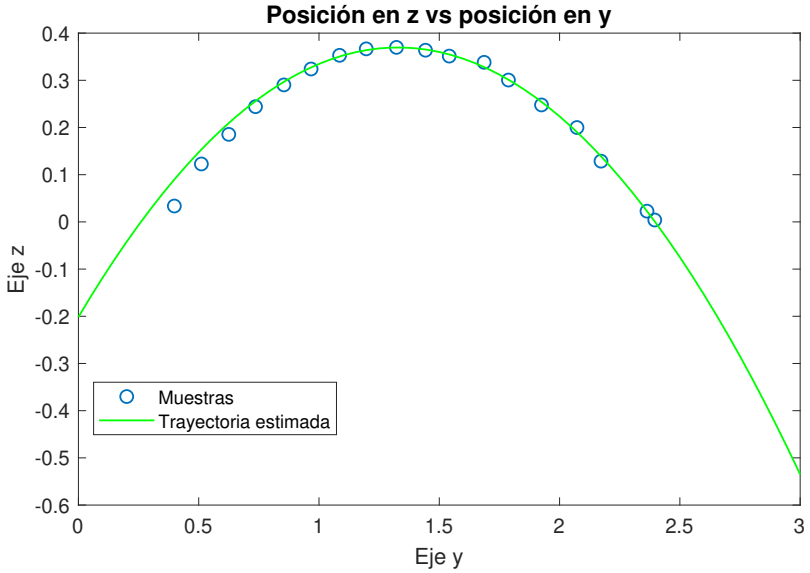


Figura 3.8: Estimación de trayectoria con 10 muestras en el eje x

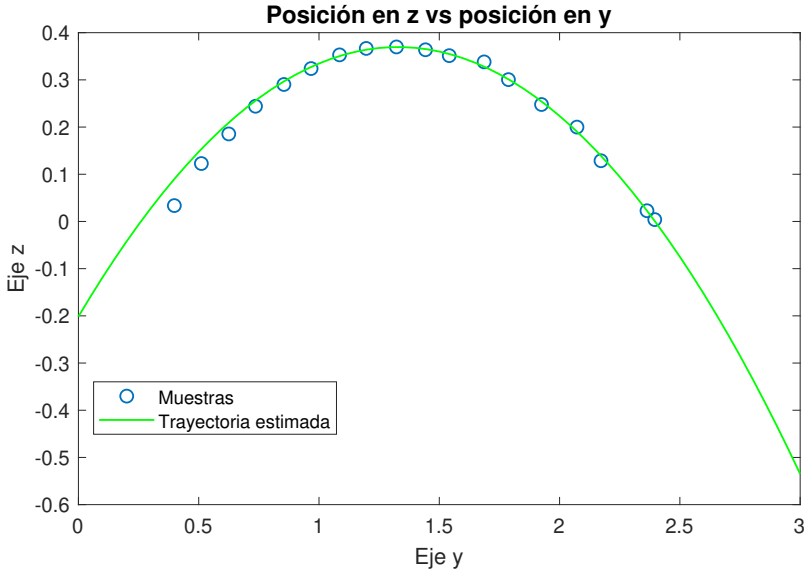


Figura 3.9: Estimación de trayectoria con 10 muestras en el eje y

En las gráficas se puede observar que la estimación es muy cercana a la trayectoria que se observa en las muestras, por lo que se puede afirmar que el método de mínimos cuadrados es efectivo.

3.4. Cálculo de posición de caída de la pelota

Una vez que se tienen 2 ecuaciones que describen la trayectoria de la forma $z = f(x)$ y $z = f(y)$.

Se igualan las ecuaciones a $z = c$, en donde c es una constante que corresponde a la altura a la que se quiere atrapar la pelota. En este caso $c = 0$ debido a que la cámara se encuentra a esa altura de acuerdo con el sistema de referencia.

El resultado de igualar $z = 0$ son las ecuaciones 3.66 y 3.67, con estas ecuaciones se puede despejar fácilmente las raíces del polinomio usando la ecuación de segundo grado como se muestra en las ecuaciones 3.68, 3.69.

$$a_2x^2 + a_1x + a_0 = 0 \tag{3.66}$$

$$b_2y^2 + b_1y + b_0 = 0 \tag{3.67}$$

Se denota a las soluciones como x_s, y_s para distinguir de las muestras.

$$x_s = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_0a_2}}{2a_2} \tag{3.68}$$

$$y_s = \frac{-b_1 \pm \sqrt{b_1^2 - 4b_0b_2}}{2b_2} \tag{3.69}$$

Como se busca obtener valores reales de las coordenadas x, y, se necesita que los discriminantes de las ecuaciones 3.68 y 3.69 sean mayores a 0 como se observa en las ecuaciones 3.70 y 3.71, en caso de que no se cumplan estas condiciones las estimaciones de caída no sirven.

$$a_1^2 - 4a_0a_2 \geq 0 \tag{3.70}$$

$$b_1^2 - 4b_0b_2 \geq 0 \tag{3.71}$$

En las ecuaciones se observa que en la mayoría de los casos se tienen 2 soluciones para x, y 2 soluciones para y. Una solución corresponde a un valor en donde ya pasó la pelota y la otra solución corresponde a nuestro punto de interés el cual es dónde va a caer la pelota.

Elección de soluciones

Para el caso de x, de acuerdo con el sistema de referencia, el valor en donde cae siempre va a ser el menor valor de las 2 soluciones, debido a la dirección en la que se mueve la pelota.

Para el caso de y, es más complejo saber el valor de caída, pero se puede predecir de acuerdo a los valores de la primera y enésima muestra. Se tienen los siguientes casos:

$$y_c = \begin{cases} y_{s1} & y_0 > y_{n-1},\ y_{s2} > y_{s1} \quad (3.72) \\ y_{s2} & y_0 > y_{n-1},\ y_{s1} > y_{s2} \quad (3.73) \\ y_{s1} & y_0 < y_{n-1},\ y_{s1} > y_{s2} \quad (3.74) \\ y_{s2} & y_0 < y_{n-1},\ y_{s2} > y_{s1} \quad (3.75) \end{cases}$$

En donde y_c es el valor de caída estimado, y_0 corresponde primera muestra, y_{n-1} corresponde a la enésima muestra, y_{s1} y y_{s2} son las soluciones de la ecuación 3.69.

Ejemplo

Al sustituir los valores de las ecuaciones 3.64 y 3.65 en las ecuaciones 3.68 y 3.69 respectivamente, se obtienen los siguientes valores:

$$x_{s1} = 0.02623$$

$$x_{s2} = 0.2629$$

$$y_{s1} = 0.2597$$

$$y_{s2} = 2.3967$$

En las figuras 3.10 y 3.11 se observa que las soluciones también pueden ser vistas como la intersección de las ecuaciones cuadráticas con el origen.

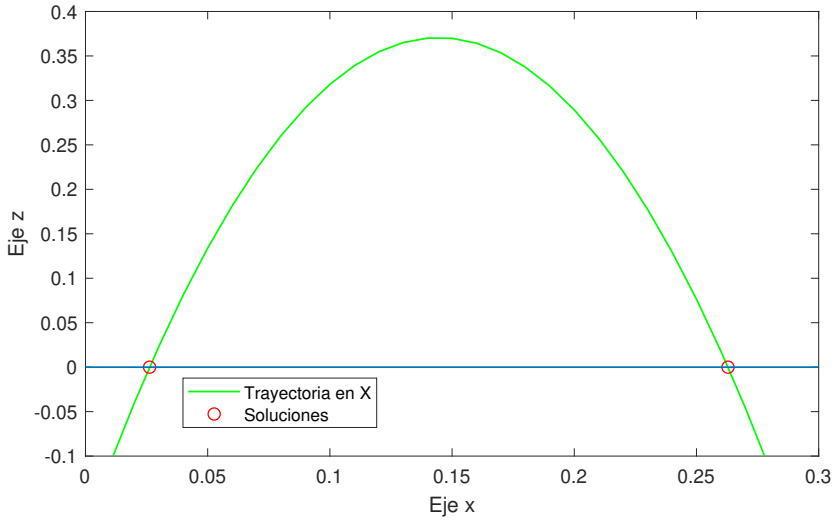


Figura 3.10: Soluciones de caída en X, una solución representa la caída estimada y la otra solución es el punto de inicio al extrapolar los datos

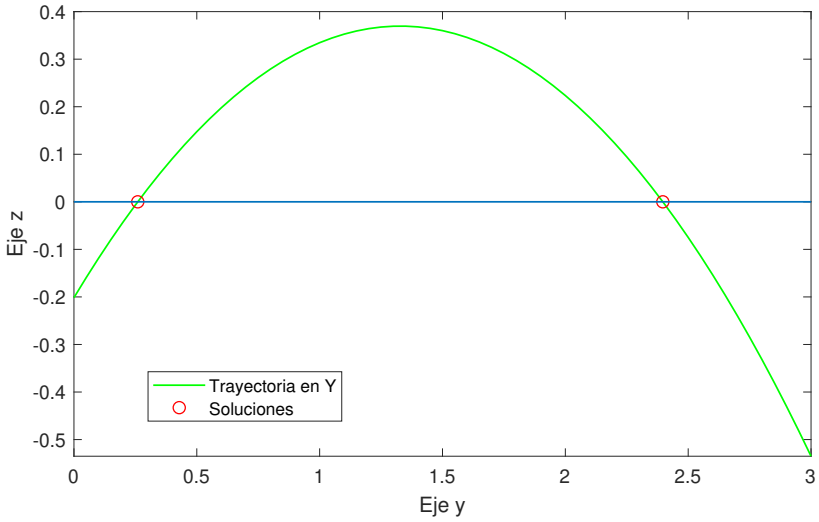


Figura 3.11: Soluciones de caída en Y, una solución representa la caída estimada y la otra solución es el punto de inicio al extrapolar los datos

Para el caso de x se elige el menor valor que en este caso es x_{s1}

Para la variable de y se tienen los siguientes datos: $y_0 = 2.3956$, $y_9 = 1.3219$, $y_{s1} = 0.2597$ y $y_{s2} = 2.3967$

De acuerdo al caso de la ecuación 3.72, se elige el siguiente valor de caída:

$$y_c = y_{s1}$$

Una vez que se eligieron las soluciones, se concluye que la pelota caerá en las coordenadas:

$$\hat{C} = (x_{s1}, y_{s1}) = (0.02623, 0.2597)$$

En la tabla 3.2 se tiene una muestra cercana a la posición de caída con $z = 0.0336$. Coordenadas cercanas al punto donde cayó la pelota:

$$C = (0.01619, 0.3986)$$

En este capítulo se plantearon los pasos que se llevarán a cabo para estimar las coordenadas de la caída de la pelota, pero aún no se cuenta con una base móvil que pueda realizar la tarea de atrapar la pelota.

El siguiente capítulo describe el diseño de una base móvil omnidireccional con el hardware suficiente para realizar la estimación de las coordenadas de caída de la pelota y moverse en esa dirección actualizando su velocidad de acuerdo al error entre la posición del robot y el punto deseado. La base móvil influye en el sistema de visión pues la matriz homogénea que se use dependerá del ángulo de inclinación de la cámara.

Capítulo 4

Diseño y construcción de la base móvil

El presente capítulo se enfoca en el hardware de la base móvil con la cual se realizarán las pruebas para determinar si es posible estimar por medio del sistema de visión el punto en donde va a caer la pelota y además se probará si el robot es capaz de moverse a dicho punto antes de que caiga la pelota.

En la sección 4.1 se explican los factores que influyen en las características de la base móvil como sus dimensiones, la elección de los materiales y algunas características de estos.

En la sección 4.2 se habla brevemente sobre la fuente de poder que energiza la base móvil, igualmente se muestran las conexiones de los componentes.

Se muestran planos sobre algunas piezas que se manufacturaron en la sección 4.3, además se da una breve explicación sobre su diseño.

Se presenta el modelo cinemático de la base móvil en la sección 4.4, este permite encontrar las velocidades necesarias en cada servomotor para mover al robot de acuerdo a ciertas velocidades deseadas.

Y finalmente, en la sección 4.5 se propone una forma de calcular la odometría del robot y así estimar en donde se encuentra el robot.

4.1. Selección de materiales y consideraciones

Antes de construir la base es importante considerar algunos parámetros de diseño, así como los materiales que se requieren.

Consideraciones:

- Se requiere una computadora pequeña que tenga capacidades de cálculo y sea capaz de comunicarse con los sensores y actuadores del robot.
- Se requiere que el robot pueda moverse en cualquier dirección para atrapar la pelota.

- Se requiere un sensor capaz de medir la posición de la pelota.
- Se requiere un servomotor que proporcione información para calcular o estimar la ubicación del robot y pueda controlar la velocidad del robot.
- Un objetivo secundario es fabricar una base que se pueda usar en la competencia Robocup. Por lo que las dimensiones del robot se enfocarán en adecuarse a las reglas de dicha competencia.

Con dichas consideraciones se eligieron las siguientes especificaciones:

- Diámetro de la base móvil de aproximadamente 18 cm
- Altura de la base menor a 30 cm
- Uso de ruedas omnidireccionales
- Uso de la tarjeta de desarrollo Nvidia Jetson Nano
- Uso de servomotores Dynamixel MX-12W
- Cámara RGB-D Intel D435-i
- Geometría de la base basada en el uso de 3 ruedas

Al hacer uso de los servomotores Dynamixel MX-12W también se usarán los siguientes materiales:

- U2D2 convertidor de comunicación: Se usa con el fin de traducir la información del protocolo USB de la Jetson al protocolo Dynamixel Protocol 1.0
- U2D2 Power Hub Board: Se usa para suministrar energía independiente de la Jetson a los servomotores

4.1.1. Especificaciones de los materiales

Especificaciones técnicas	
GPU	128-core NVIDIA Maxwell™
CPU	Quad-core ARM A57 @ 1.43 GHz
Memoria	4GB 64-bit LPDDR4 25.6GB/s
Almacenamiento	Entrada microSD
Conectividad	Gigabit Ethernet, M.2 Key E
USB	4x USB 3.0, 1x USB 2.0 Micro-B
Otros	40 pines (GPIO, I2C, I2S, SPI, UART) 12 pines (Energía y señales relacionadas)
Medidas	100 mm x 80 mm x 29 mm

Tabla 4.1: Especificaciones NVIDIA Jetson Nano Developer Kit

Especificaciones técnicas	
Velocidad max. cámara RGB	30 cuadros por segundo
Velocidad max. cámara profundidad	90 cuadros por segundo
Rango ideal	0.3 m-3 m
Conexión	USB-C 3.1
Medidas	90 mm x 25 mm x 25 mm

Tabla 4.2: Especificaciones Intel Cámara de profundidad D435i

Especificaciones técnicas	
MCU	ARM CORTEX-M3 (72 [MHz], 32Bit)
Baud rate	8,000 [bps] 4.5 [Mbps]
Voltaje de entrada	10-14.8V
Medidas	32 mm x 50 mm x 40 mm
Tipo de protocolo	Half Duplex asíncrono

Tabla 4.3: Especificaciones Dynamixel MX-12W

Como se observa en la tabla 4.1 la tarjeta no cuenta con conectividad Wi-Fi, tecnología necesaria para la autonomía del robot por lo que se agrego un adaptador Wi-Fi.

4.2. Alimentación de la base

Con el fin de que la base pueda moverse sin cables es necesario tener una fuente de energía dentro de la base. La Jetson Nano[1] se alimenta con 5[V] y los servomotores MX-12W[2] se alimentan con un rango de 10[V] a 14.8[V] de acuerdo a las especificaciones del fabricante.

Se decidió alimentar la base con 2 baterías de LiPo debido a que con una batería alimentando los servomotores y un regulador para la Jetson se producía una caída de voltaje que apagaba la Jetson, se escogió una batería de 2 celdas con un voltaje de 7.4[V] y otra con 3 celdas que proporcionaba 11.1[V].

La batería de 11.1[V] puede alimentar a los servomotores directamente, sin embargo la batería de 7.4[V] no puede alimentar directamente a la Jetson por eso se decidió incluir un regulador, en este regulador entran voltajes mayores a 5[V] pero en la salida siempre se obtienen voltajes muy cercanos a 5[V] como se muestra en la figura 4.1.

Se escogió el Step Down D24V90F5[3] de la marca Pololu como regulador debido a que puede soportar una corriente de 9[A]. En la figura 4.1 se muestra la gráfica del voltaje de salida para diferentes voltajes de entrada.

[1] https://developer.nvidia.com/embedded/jetson-nano
[2] https://emanual.robotis.com/docs/en/dxl/mx/mx-12w/
[3] https://www.pololu.com/product/2866

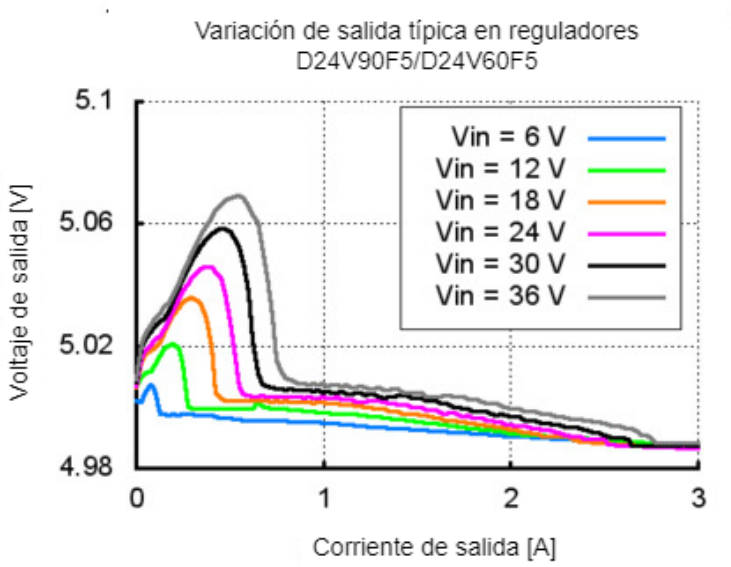


Figura 4.1: Voltaje de salida típico, modificado de la página del fabricante[4]

En la figura 4.2 se observa un esquema de las conexiones de las baterías a los componentes.

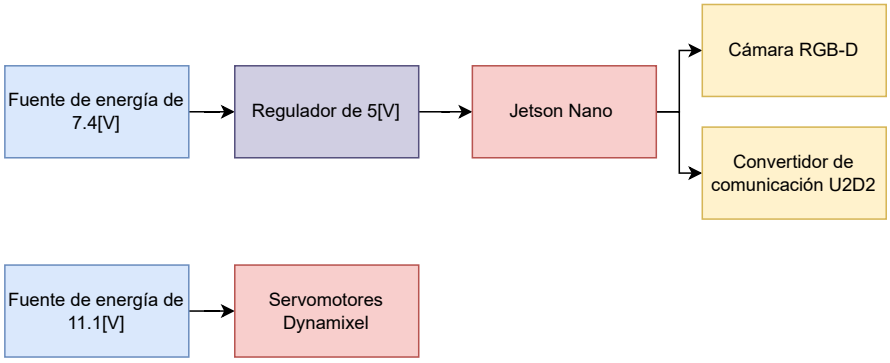


Figura 4.2: Esquema de alimentación a los componentes de la base

[4]https://www.pololu.com/product/2866

4.3. Diseño de piezas

4.3.1. Transmisión

No se encontraron piezas comerciales para transmitir el torque del servomotor a la rueda por lo que se tuvo que diseñar una pieza con base en las medidas de los servomotores y las ruedas omnidireccionales.

Para los servomotores MX-12W se buscaron los datos del fabricante como se muestra en la figura 4.3.

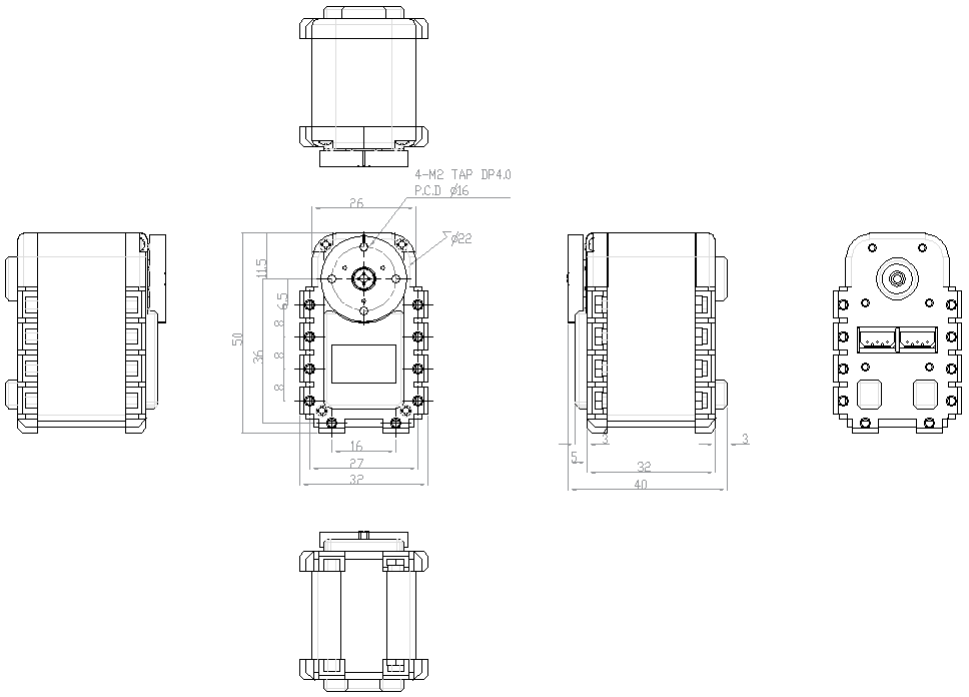


Figura 4.3: Medidas del servomotor, planos proporcionados por el fabricante [5]

Para los hubs de las ruedas se buscaron los datos del fabricante[6] como se muestra en la figura 4.4.

[5] http://robotchassisparts.com/wp-content/uploads/2016/09/4mm-Hub-for-60mm-Mecanum-Wheel.pdf

[6] http://robotchassisparts.com/wp-content/uploads/2016/09/4mm-Hub-for-60mm-Mecanum-Wheel.pdf

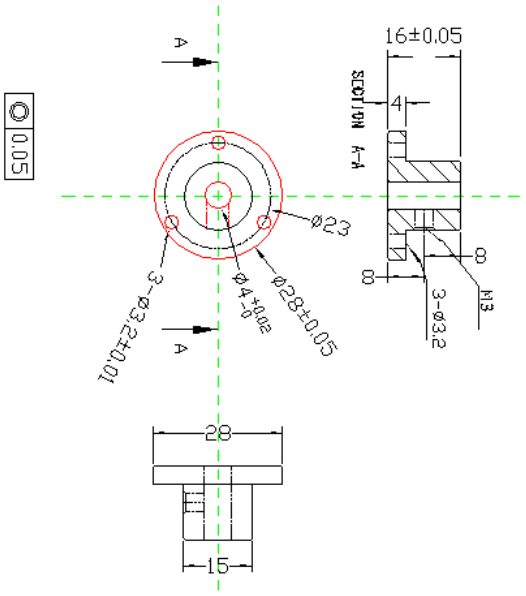


Figura 4.4: Medidas del hub de la rueda, planos proporcionados por el fabricante [7]

Con las medidas de servomotores y hubs, se crearon piezas como las que se observan en la figura 4.5 con el fin a acoplar la rueda a los servomotores y transmitir la energía del torque en los servomotores.

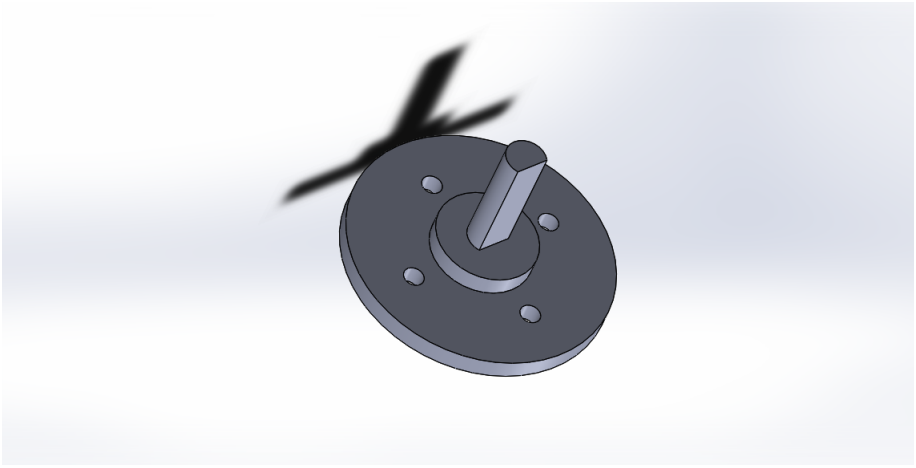


Figura 4.5: Pieza de transmisión diseñada

[7] http://en.robotis.com/service/downloadpage.php?ca_id=70

En la figura 4.6 se observan los planos con las medidas de la pieza diseñada, las medidas están en milímetros.

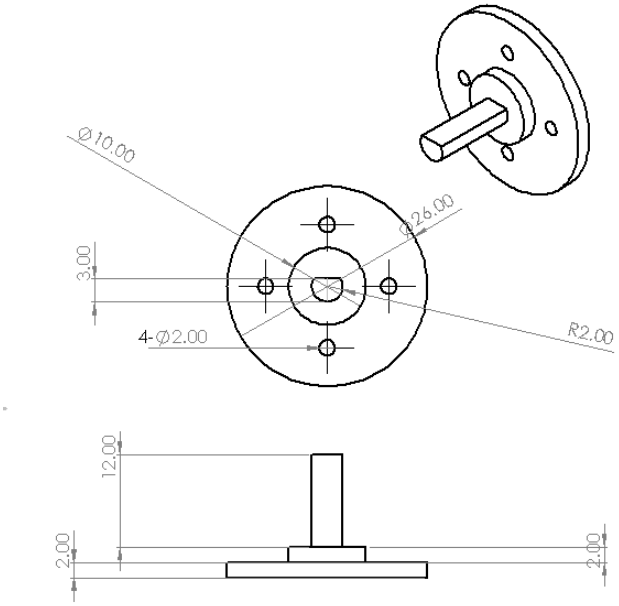


Figura 4.6: Medidas de la pieza de transmisión diseñada

Se realizaron pruebas con piezas manufacturadas por impresoras 3D, a pesar de esto, no fueron lo suficientemente resistentes al torque de los servomotores y se rompieron, por lo que las piezas finales se fabricaron con aluminio.

4.3.2. Soporte de la cámara con manufactura aditiva

El soporte de la cámara tiene 2 propósitos:

- Fijar la cámara a la base, evitando cambios en la transformación homogénea que relaciona el sistema de referencia de la cámara con el sistema de referencia de la base.
- Inclinar la cámara para aumentar el alcance de visión de la cámara, permitiendo guardar los datos de un tiro parabólico con una mayor altura.

En el caso del diseño del soporte de la cámara se tiene más libertad ya que solo se debe respetar ciertas medidas de esta para poder atornillar y fijarla al soporte. Para el diseño se usan las medidas proporcionadas por el fabricante [8] las cuales se observan en la figura 4.7.

[8] https://www.intelrealsense.com/wp-content/uploads/2022/05/Intel-RealSense-D400-Series-Datasheet-April-2022.pdf

Mientras que la altura del soporte en conjunto con el resto de la base no supere las medidas planteadas de la competencia Robocup se tiene libertad de diseño, en cuanto a la forma de fijar el soporte a la base sucede lo mismo y se tiene libertad de diseño.

Con las especificaciones mencionadas para el diseño de esta pieza, se diseñó una pieza como la que se observa en la figura 4.8

En la figura 4.9 se observan los planos del soporte diseñado, las medidas se encuentran en milímetros.

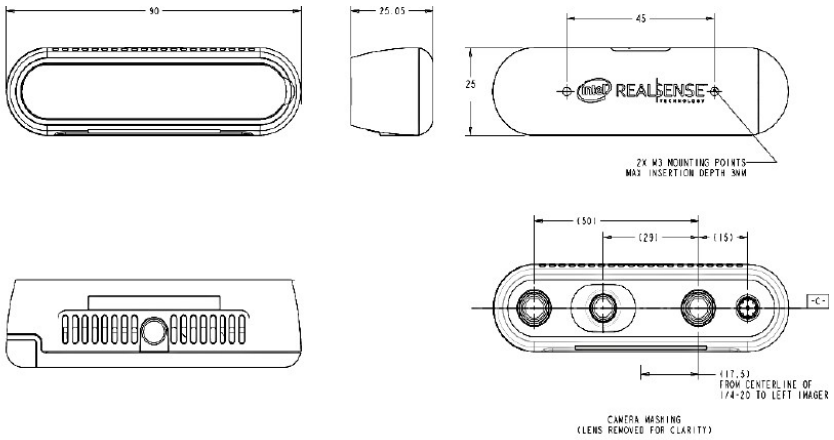


Figura 4.7: Planos cámara RGB-D [9]

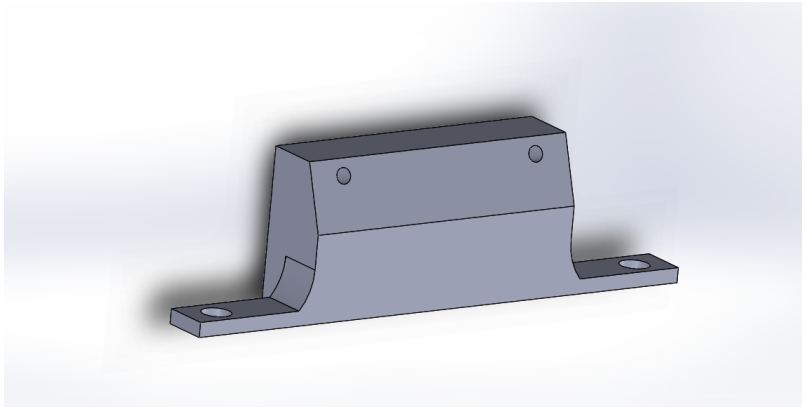


Figura 4.8: Soporte de la cámara

[9] https://www.intelrealsense.com/wp-content/uploads/2022/05/Intel-RealSense-D400-Series-Datasheet-April-2022.pdf

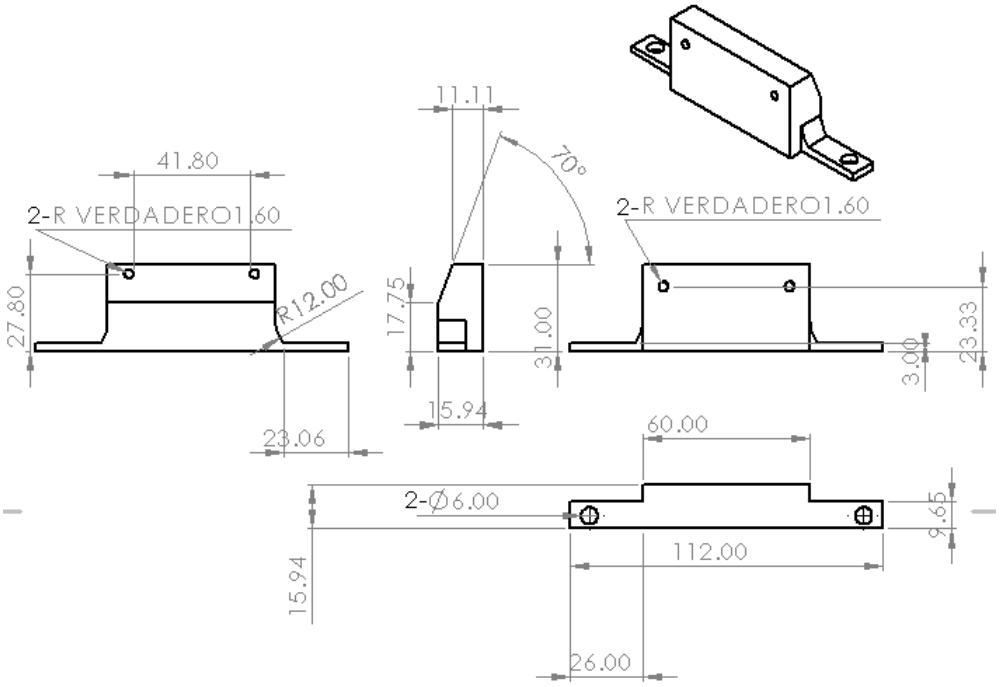


Figura 4.9: Medidas del soporte diseñado

4.3.3. Chasis

El chasis es la estructura que le da forma al robot además cumple la función de unir y fijar los componentes (servomotores, cámara, computadora, entre otras piezas) para que en conjunto sean un sistema más complejo.

Para el chasis se deben respetar las medidas impuestas por la competencia Robocup y debido a que se eligió una geometría con 3 ruedas se necesita tener las ruedas separadas por 120° entre sí.

El chasis cuenta con 3 pisos, en el primer piso se encuentra montado el regulador, en el segundo piso se encuentran montados la Jetson, el convertidor U2D2, las baterías, los servomotores y por último, en el tercer piso solo se monta el soporte de la cámara. Los 3 pisos son unidos por postes.

Los servomotores se montan al segundo piso del chasis con ayuda del bracket FP04-F3[10] cuyos planos se muestran en la figura 4.10

[10]https://www.robotis.us/dynamixel-ax-bioloid-frames/

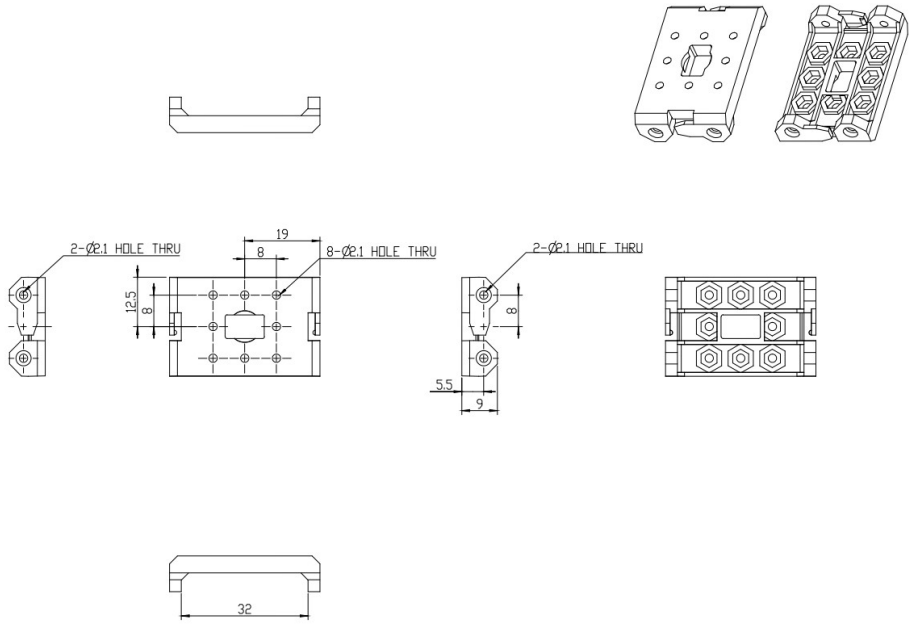


Figura 4.10: Planos del bracket FP04-F3

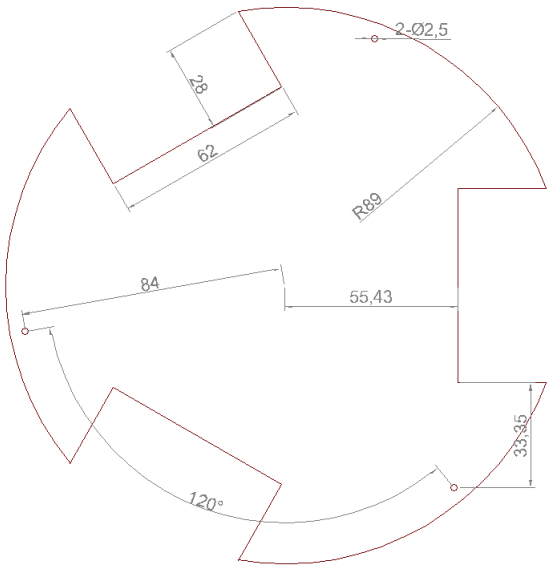


Figura 4.11: Plano primer piso del chasis

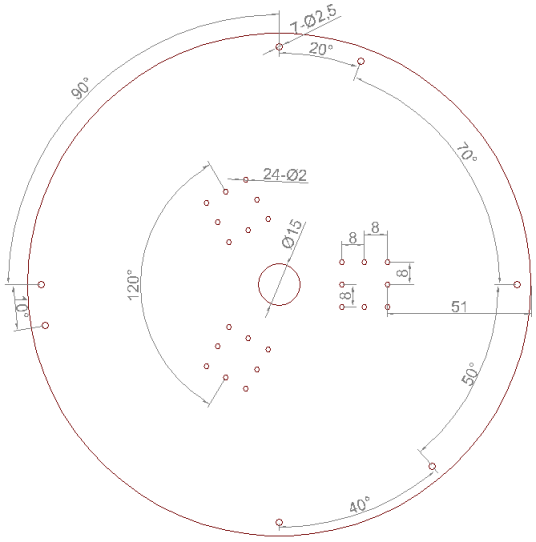


Figura 4.12: Plano segundo piso del chasis

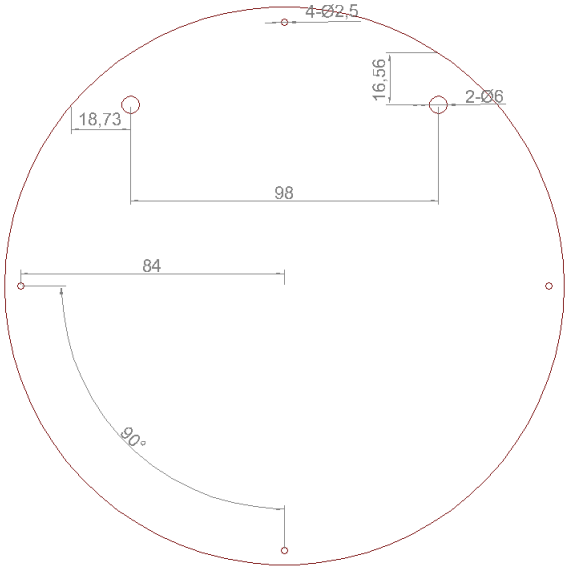


Figura 4.13: Plano tercer piso del chasis

En las figuras 4.11, 4.12 y 4.13 se observan los planos de las estructuras que se ensamblan con postes, como los que se observan en la figura 4.14, para conformar el chasis del robot.

Figura 4.14: Postes y separadores

En la figura 4.15 se observa el chasis ensamblado con todas las piezas.

Figura 4.15: Base móvil construida

4.4. Cinemática de la base móvil

La base móvil tiene a las ruedas omnidireccionales fijas posicionadas a cierto ángulo α_i y a una distancia R de los ejes referencia en x, y.

En la figura 4.16 se observa que la velocidad de la rueda 1 la cual será denotada como V_a está formada por una velocidad V_1 y por una velocidad rotacional en la base denotada como $\dot{\theta}$, la relación entre estas variables se describe en la ecuación 4.1.

$$V_a = V_1 + R\dot{\theta} \tag{4.1}$$

Entre la velocidad V_1 y el ángulo θ se tienen las siguientes relaciones

$$V_{Y1} = V_1 \cos\theta \tag{4.2}$$
$$-V_{X1} = V_1 \sin\theta \tag{4.3}$$

Al multiplicar la ecuación 4.2 por $\cos\theta$ y la ecuación 4.3 por $\sin\theta$ se tienen las siguientes ecuaciones:

$$V_{Y1}\cos\theta = V_1\cos^2\theta \tag{4.4}$$
$$-V_{X1}\sin\theta = V_1\sin^2\theta \tag{4.5}$$

Y al sumar las ecuaciones 4.4 y 4.5 se obtiene la ecuación 4.6.

$$V_1 = V_{Y1}\cos\theta - V_{X1}\sin\theta \tag{4.6}$$

Para que el robot se mueva a la velocidad deseada se debe cumplir que:

$$V_{X1} = \dot{x}$$

$$V_{Y1} = \dot{y}$$

Por lo tanto, la ecuación que describe la relación entre la velocidad de cada servomotor y la velocidad de la base es:

$$V_1 = -\sin(\theta)\dot{x} + \cos(\theta)\dot{y} + R\dot{\theta} \tag{4.7}$$

De forma general existe una relación entre las velocidades globales de la base $\dot{x}$, $\dot{y}$, $\dot{\theta}$ y la velocidad traslacional de cada rueda como se observa en la ecuación 4.8

$$V_i = -\sin(\theta+\alpha_i)\dot{x} + \cos(\theta+\alpha_i)\dot{y} + R\dot{\theta} \tag{4.8}$$

La velocidad traslacional de las ruedas también se puede reescribir como la ecuación 4.9, en donde r es el radio de la rueda y ϕ_i la velocidad rotacional de la iésima rueda.

$$V_i = r\dot{\phi_i} \tag{4.9}$$

Sustituyendo la ecuación 4.9 en la ecuación 4.8 se obtiene la ecuación 4.10 que relaciona la velocidad angular de cada rueda con las velocidades globales de la base.

$$r\dot{\phi_i} = -\sin(\theta+\alpha_i)\dot{x} + \cos(\theta+\alpha_i)\dot{y} + R\dot{\theta} \tag{4.10}$$

La ecuación 4.11 es la representación matricial de la ecuación 4.10.

$$\begin{bmatrix}\dot{\phi}_1\\ \dot{\phi}_2\\ \dot{\phi}_3\end{bmatrix} = \frac{1}{r}\begin{bmatrix}-\sin(\theta) & \cos(\theta) & R\\ -\sin(\theta+\alpha_2) & \cos(\theta+\alpha_2) & R\\ -\sin(\theta+\alpha_3) & \cos(\theta+\alpha_3) & R\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} \tag{4.11}$$

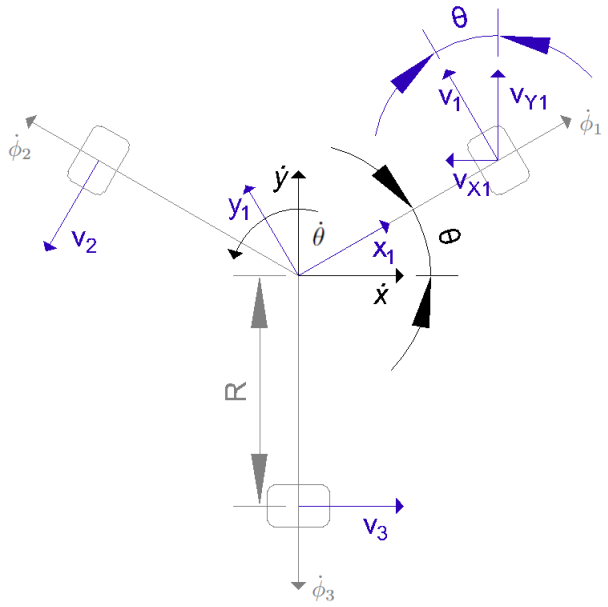


Figura 4.16: Diagrama cinemático de la base

Para mover la base se usa el modo rueda en los servomotores MX-12W, en este modo se mueven a una velocidad meta la cual es alcanzada por un control PID.

La ecuación 4.11 proporciona las velocidades necesarias en cada servomotor para mover la base en la velocidad y dirección deseada, sin embargo, se debe traducir esa información a un valor que el servomotor pueda comprender.

Los servomotores cuentan con una memoria EEPROM, esta memoria cuenta con diferentes direcciones que contienen los parámetros que describen el comportamiento del servomotor.

La velocidad deseada se guarda en la dirección 32 de la memoria EEPROM y se tiene acceso de escritura y lectura. La palabra que se escribe en la dirección de la velocidad es de 10 bits. En la tabla 4.4 se observa una palabra de 10 bits que representa una velocidad de -400.292[rpm].

Posición del bit	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Valor del bit	1	0	1	1	0	1	1	0	1	0	1

Tabla 4.4: Velocidad de -400.292[rpm] decodificada en el servomotor

El décimo bit de la palabra que se escribe en el servomotor corresponde al sentido de giro, cuando el valor del décimo bit tiene un valor de 1 como en el ejemplo de la tabla 4.4 se produce un giro dextrógiro, es decir en sentido de las manecillas del reloj, mientras que cuando el valor del décimo bit es 0 se produce

un giro levógiro, es decir en sentido opuesto a las manecillas del reloj. En la figura 4.17 se observa el sentido de giro en el servomotor.

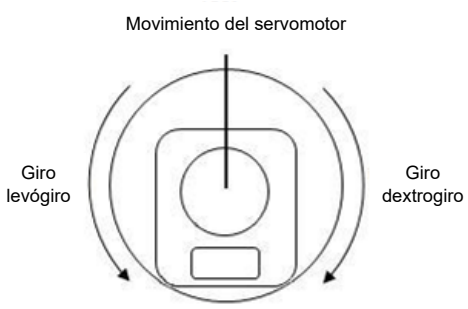


Figura 4.17: Dirección del movimiento en los servomotores visto frente al rotor, modificado de la página del fabricante [11]

Para escribir en los motores se decidió trabajar con números decimales por lo que la velocidad deseada se encuentra en un rango de 0 a 2047. El rango de 0 a 1023 produce un giro levógiro mientras que en el rango de 1024 a 2047 se produce un giro dextrógiro.

Al hacer pruebas con el servomotor montado en la base, se observó que cuando se requiere una velocidad rotacional negativa de acuerdo con la ecuación 4.11 el valor que se escribe en el servomotor debe estar en el rango de 0 a 1023 y cuando se requiere una velocidad rotacional positiva el valor que se escribe debe estar en el rango de 1024 a 2047.

En cuanto a las unidades de escritura una unidad es equivalente a 0.916[rpm] o 0.09592[rad/s], por lo que si escribimos 1025 el servomotor tendrá un giro en sentido de las manecillas del reloj con una velocidad de 1.832[rpm] o 0.191846[rad/s].

En la ecuación 4.12 se observa la relación entre la magnitud de la velocidad rotacional deseada denotada como ϕ_i y el valor que se debe escribir en el servomotor denotado como ψ_i.

$$\psi_i = \frac{30|\dot{\phi_i}|}{0.916\pi} \tag{4.12}$$

La ecuación 4.12 es correcta, a pesar de ello, al servomotor solo se le pueden escribir números enteros, si $\phi = 7.348$ se debe redondear a 7. Al redondear un valor x será denotado como $\lfloor x \rfloor$, por lo que al redondear la ecuación 4.12 se obtiene la ecuación 4.13.

$$\psi_i = \lfloor \frac{30|\dot{\phi_i}|}{0.916\pi} \rfloor \tag{4.13}$$

Al considerar el sentido de rotación se tienen las siguientes ecuaciones:

[11] https://emanual.robotis.com/docs/en/dxl/mx/mx-12w/

$$\psi_i = \begin{cases} \lfloor -\frac{30\dot{\phi}_i}{0.916\pi} \rfloor & \phi_i \leq 0[\text{rad/s}] \quad (4.14) \\ 1024 + \lfloor \frac{30\dot{\phi}_i}{0.916\pi} \rfloor & \phi_i > 0[\text{rad/s}] \quad (4.15) \end{cases}$$

Las ecuaciones anteriores son válidas si la velocidad rotacional deseada en los servomotores se encuentra en un rango de -937[rpm] a 937 [rpm] equivalente a un rango de -98 [rad/s] a 98 [rad/s] que corresponde a la velocidad máxima del servomotor. Es necesario limitar la velocidad de los servomotores porque como se observa en la figura 4.18 para algunos valores del codominio (ψ) se tienen 2 valores en el dominio ($\dot{\phi}$) y por lo tanto no se tiene una función definida.

Por otro lado, físicamente si se calcula en la ecuación 4.10 un valor de $\dot{\phi}$ menor a -98[rad/s] en la ecuación 4.14 se obtiene un valor de ψ mayor a 1024 y este será interpretado por el servomotor como un giro en sentido opuesto al deseado y con una menor magnitud, lo que provocará un desvió en la trayectoria de la base móvil.

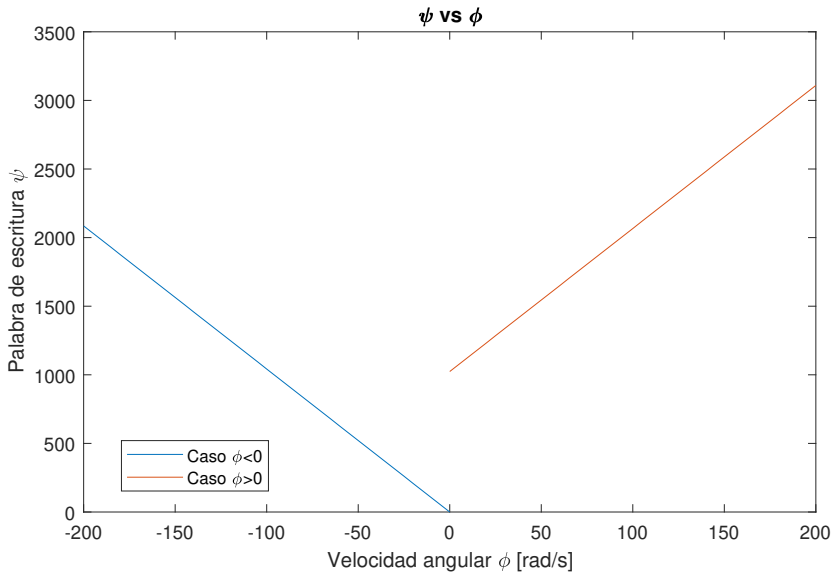


Figura 4.18: Palabra de escritura vs Velocidad angular

La prioridad de la base móvil es mantener la dirección que se requiere en $\dot{x}$, $\dot{y}$ y $\dot{\theta}$ por lo que si se calcula que cualquier servomotor requiere una velocidad en magnitud mayor a 98[rad/s] es necesario ajustar todos los servomotores.

Se propone ajustar los valores de los servomotores introduciendo una constante k como se observa en las ecuaciones 4.16 y 4.17

$$\psi_i = \begin{cases} \lfloor -\frac{30\dot{\phi}_i}{0.916\pi k} \rfloor & \phi_i \leq 0[\text{rad/s}] \quad (4.16) \\ 1024 + \lfloor \frac{30\dot{\phi}_i}{0.916\pi k} \rfloor & \phi_i > 0[\text{rad/s}] \quad (4.17) \end{cases}$$

Para calcular k se siguen los siguientes pasos:

Primero se forma un vector denotado como Φ_{abs} con los valores absolutos de $\dot{\phi}$ como se muestra en la ecuación 4.18

$$\Phi_{abs} = (|\dot{\phi_1}|, |\dot{\phi_2}|, |\dot{\phi_3}|) \quad (4.18)$$

Después se asigna a M el máximo valor del vector Φ_{abs} como se observa en la ecuación 4.19

$$M = max(\Phi_{abs}) \quad (4.19)$$

k es calculada dependiendo del valor de M como se observa en las ecuaciones 4.20 y 4.21.

$$k = \begin{cases} \frac{M}{\rho} & M > S[\text{rad/s}] \quad (4.20) \\ 1 & M \leq S[\text{rad/s}] \quad (4.21) \end{cases}$$

En donde ρ es el valor de saturación que puede ser igual o menor al valor máximo de velocidad angular (98[rad/s] en el caso del servomotor MX-12W).

4.5. Odometría de la base móvil

Es importante que el robot pueda estimar en donde se encuentra para que reconozca si se acerca o se aleja del punto deseado. Los servomotores MX-12W pueden proporcionar información sobre la posición en la que se encuentran gracias a que cuentan con un sensor de posición.

El movimiento angular denotado como $\Delta\phi$, se puede calcular como se observa en la ecuación 4.22 en donde ϕ_n corresponde a la posición angular actual y ϕ_{n-1} corresponde a la posición angular anterior en el servomotor.

$$\Delta\phi = \phi_n - \phi_{n-1} \quad (4.22)$$

Sin embargo, el sensor de posición entrega un valor entero de 0 a 4095 por lo que se debe a radianes, en la ecuación 4.23 se obtiene el movimiento que realizó el servomotor en [rad] que se denota como $\Delta\phi$, en donde L_n es la última lectura leída por los servomotores, L_{n-1} es la lectura anterior.

$$\Delta\phi = \frac{2\pi(L_n - L_{n-1})}{4096} \quad (4.23)$$

La ecuación 4.23 no siempre es válida porque como se observa en la figura 4.19 si el periodo de muestreo es lo suficientemente rápido para detectar pequeñas variaciones como se requiere en la estimación de la posición del robot, existen de acuerdo con el ejemplo y se tienen las siguientes excepciones:

- Si el servomotor avanza de la posición 1 a la posición 4095, es decir $L_{n-1} = 1$ y $L_n = 4095$, la ecuación 4.23 interpreta que casi dio una vuelta completa.

$$\Delta\phi = \frac{2\pi(4095 - 1)}{4096} = 6.28[rad] \tag{4.24}$$

- Si el servomotor avanza de la posición 4095 a la posición 1, es decir $L_{n-1} = 4095$ y $L_n = 1$, la ecuación 4.23 interpreta que casi dio una vuelta completa, en sentido contrario al inciso anterior.

$$\Delta\phi = \frac{2\pi(1 - 4095)}{4096} = -6.28[rad] \tag{4.25}$$

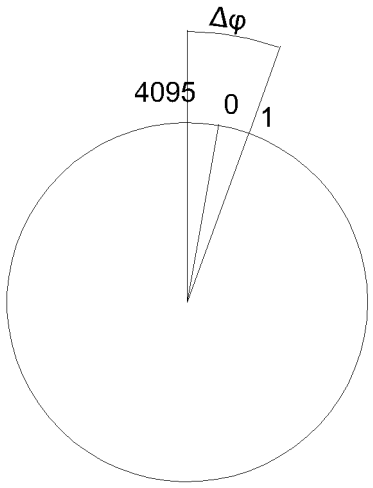


Figura 4.19: Caso especial de lectura de servomotores

De forma general si la frecuencia de lectura en la posición de los servomotores es al menos el doble que la frecuencia en la que el motor da una vuelta completa entonces se tienen los siguientes casos:

$$\Delta\phi = \begin{cases} \dfrac{2\pi(L_n - L_{n-1} - 4096)}{4096} & (L_n - L_{n-1}) \geq 2048 \quad (4.26) \\ \dfrac{2\pi(L_n - L_{n-1} + 4096)}{4096} & (L_n - L_{n-1}) \leq -2048 \quad (4.27) \\ \dfrac{2\pi(L_n - L_{n-1})}{4096} & \text{Otro caso} \quad (4.28) \end{cases}$$

Una vez que se tienen las ecuaciones que proporcionan la variación de la posición en [rad], se necesita una ecuación que indique el movimiento de la base móvil en función de las lecturas del sensor de posición.

En la sección 4.4 se obtuvo una ecuación que relaciona las velocidades globales de la base móvil con la velocidad rotacional de cada servomotor. Pero en la odómetro las variables de interés son la posición, la velocidad es la derivada de la posición por lo tanto con la ecuación 4.30 y al aproximar todas las derivadas de la ecuación 4.10 como en la ecuación 4.29 en donde F es la variable que se desea aproximar y Δt el periodo de en el que se desea actualizar el error.

$$\dot{F} \approx \frac{F_n - F_{n-1}}{\Delta t} \tag{4.29}$$

$$r\frac{\phi_{i,n} - \phi_{i,n-1}}{\Delta t} = -\sin(\theta + \alpha_i)\frac{x_{i,n} - x_{i,n-1}}{\Delta t} + \cos(\theta + \alpha_i)\frac{y_{i,n} - y_{i,n-1}}{\Delta t} + R\frac{\theta_{i,n} - \theta_{i,n-1}}{\Delta t} \tag{4.30}$$

Al tener Δt en ambos lados de la ecuación 4.30 se puede simplificar como se observa en la ecuación 4.31.

$$r(\phi_{i,n} - \phi_{i,n-1}) = -\sin(\theta + \alpha_i)(x_{i,n} - x_{i,n-1}) + \cos(\theta + \alpha_i)(y_{i,n} - y_{i,n-1}) + R(\theta_{i,n} - \theta_{i,n-1}) \tag{4.31}$$

En forma matricial se tiene la ecuación 4.32 que representa una discretización por diferencias finitas de la ecuación 4.11.

$$\begin{bmatrix} \phi_{1,n} - \phi_{1,n-1} \\ \phi_{2,n} - \phi_{2,n-1} \\ \phi_{3,n} - \phi_{3,n-1} \end{bmatrix} = \frac{1}{r}\begin{bmatrix} -\sin(\theta) & \cos(\theta) & R \\ -\sin(\theta + \alpha_2) & \cos(\theta + \alpha_2) & R \\ -\sin(\theta + \alpha_3) & \cos(\theta + \alpha_3) & R \end{bmatrix}\begin{bmatrix} x_n - x_{n-1} \\ y_n - y_{n-1} \\ \theta_n - \theta_{n-1} \end{bmatrix} \tag{4.32}$$

La ecuación matricial tiene la forma $y = Ax$ en donde

$$y = \begin{bmatrix} \phi_{1,n} - \phi_{1,n-1} \\ \phi_{2,n} - \phi_{2,n-1} \\ \phi_{3,n} - \phi_{3,n-1} \end{bmatrix} \tag{4.33}$$

$$A = \begin{bmatrix} -\frac{sin(\theta)}{r} & \frac{cos(\theta)}{r} & \frac{R}{r} \\ -\frac{sin(\theta+\alpha_2)}{r} & \frac{cos(\theta+\alpha_2)}{r} & \frac{R}{r} \\ -\frac{sin(\theta+\alpha_3)}{r} & \frac{cos(\theta+\alpha_3)}{r} & \frac{R}{r} \end{bmatrix} \tag{4.34}$$

$$x = \begin{bmatrix} x_n - x_{n-1} \\ y_n - y_{n-1} \\ \theta_n - \theta_{n-1} \end{bmatrix} \tag{4.35}$$

Al pre-multiplicar ambos lados de la ecuación 4.32 por la inversa de A se obtiene la ecuación 4.36.

$$\begin{bmatrix} x_n - x_{n-1} \\ y_n - y_{n-1} \\ \theta_n - \theta_{n-1} \end{bmatrix} = \begin{bmatrix} -\frac{sin(\theta)}{r} & \frac{cos(\theta)}{r} & \frac{R}{r} \\ -\frac{sin(\theta+\alpha_2)}{r} & \frac{cos(\theta+\alpha_2)}{r} & \frac{R}{r} \\ -\frac{sin(\theta+\alpha_3)}{r} & \frac{cos(\theta+\alpha_3)}{r} & \frac{R}{r} \end{bmatrix}^{-1} \begin{bmatrix} \phi_{1,n} - \phi_{1,n-1} \\ \phi_{2,n} - \phi_{2,n-1} \\ \phi_{3,n} - \phi_{3,n-1} \end{bmatrix} \tag{4.36}$$

A partir de la ecuación 4.36 se despejan los valores x_n, y_n y θ_n y se obtiene la ecuación 4.37 que permite actualizar la posición del robot a partir de la posición anterior y el desplazamiento de las ruedas en un periodo de tiempo t.

$$\begin{bmatrix} x_n \\ y_n \\ \theta_n \end{bmatrix} = \begin{bmatrix} x_{n-1} \\ y_{n-1} \\ \theta_{n-1} \end{bmatrix} + \begin{bmatrix} -\frac{\sin(\theta)}{r} & \frac{\cos(\theta)}{r} & \frac{R}{r} \\ -\frac{\sin(\theta+\alpha_2)}{r} & \frac{\cos(\theta+\alpha_2)}{r} & \frac{R}{r} \\ -\frac{\sin(\theta+\alpha_3)}{r} & \frac{\cos(\theta+\alpha_3)}{r} & \frac{R}{r} \end{bmatrix}^{-1} \begin{bmatrix} \phi_{1,n} - \phi_{1,n-1} \\ \phi_{2,n} - \phi_{2,n-1} \\ \phi_{3,n} - \phi_{3,n-1} \end{bmatrix} \tag{4.37}$$

El valor real de θ que se encuentra en los argumentos de las funciones trigonométricas en la ecuación 4.37 no se conoce pero se aproximará a θ_{n-1}.

Además $\phi_{i,n} - \phi_{i,n-1}$ corresponde a $\Delta\phi$ y se puede calcular con la lectura de los servomotores como se indica en la ecuación 4.23.

En este capítulo se diseñó una base móvil capaz de moverse al punto de interés estimado con el sistema de visión, además se presentaron ecuaciones que permiten mover y estimar la posición del robot por medio de los servomotores.

El siguiente capítulo describe el sistema de control con el cual se obtendrá una entrada que cambiará en el tiempo para cada servomotor y con esta entrada el robot será capaz de llegar de una posición inicial a una posición deseada. Las ecuaciones vistas en las secciones 4.4 y 4.5 serán de gran ayuda para controlar el movimiento del robot y el nivel de complejidad en el control se verá afectado por el diseño omnidireccional.

Capítulo 5

Sistema de control de posición

El presente capítulo contiene las propuestas que se llevan a cabo para que el robot llegué a las coordenadas donde se estima va a caer la pelota.

En la sección 5.1 se analiza la estabilidad del sistema de la base móvil utilizando diferentes leyes de control y una de ellas será seleccionada para implementar.

La sección 5.2 propone un criterio a partir del cual el robot define las coordenadas deseadas y comienza su movimiento.

5.1. Control de posición del robot

El propósito del sistema de control, es llevar a la base móvil de las coordenadas (x,y,θ) que describen la configuración del robot en el tiempo hasta las coordenadas deseadas (x_d,y_d,θ_d) por medio de una entrada de control. Las entradas de control para el robot serán V_x, V_y, ω que corresponden a las velocidades en las coordenadas (x, y) y a la velocidad angular del robot.

5.1.1. Base omnidireccional

La base omnidireccional permite que el robot sea holonómico esto significa que se pueden controlar los estados x, y y θ en el robot al mismo tiempo y sin restricciones. Esto se puede observar comparando las rutas de un robot omnidireccional como la figura 5.2 y un robot diferencial como la figura 5.1.

Dicho de otra manera, el robot omnidireccional de la figura 5.1 va directamente al punto de interés y al mismo tiempo controla la orientación deseada por lo tanto es holonómico, mientras que el robot diferencial de la figura 5.2 tiene que planear una ruta que le permita llegar al punto deseado con la orientación deseada, pero a lo largo de la ruta se observa cómo va cambiando su orientación, por consiguiente no es holonómico. La ventaja de una base omnidireccional es

que esta puede avanzar en cualquier dirección y de esta forma no se tiene que planear una ruta.

Para la aplicación de este proyecto se busca llegar a las coordenadas donde se estima caerá la pelota antes de su caída y dado que, se tienen condiciones ideales en el terreno donde se realizarán las pruebas no existen obstáculos, además que la superficie es plana, por lo que se puede elegir una ruta como la de la figura 5.1.

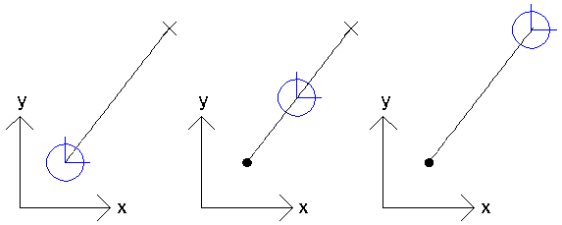


Figura 5.1: Ruta de un robot omnidireccional

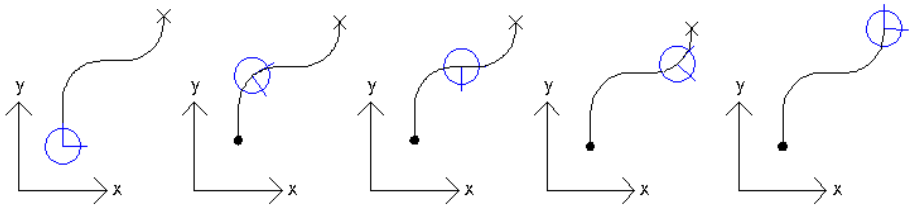


Figura 5.2: Ruta de un robot diferencial

5.1.2. Leyes de control

Las leyes de control se encargarán de calcular la señal de la velocidad angular que se necesita en los servomotores para que el robot realice el movimiento deseado. En un control realimentado estas señales dependen del error, el error representa que tan lejos se encuentra la variable de interés del punto deseado.

Modelo cinemático de la base omnidireccional

En la figura 5.3 se observan las siguientes relaciones:

$$\dot{x} = V_x \cos(\theta) - V_y \sin\theta \tag{5.1}$$

$$\dot{y} = V_x \sin(\theta) + V_y \cos\theta \tag{5.2}$$

$$\dot{\theta} = \omega \tag{5.3}$$

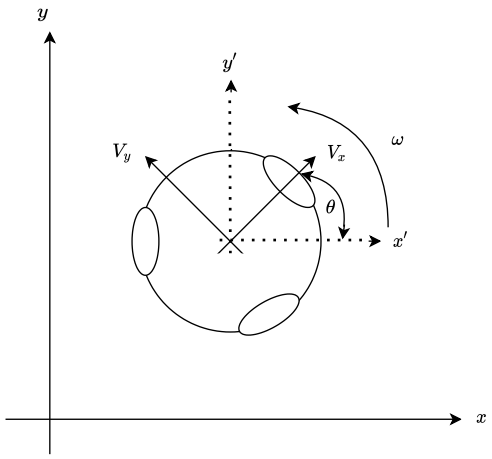


Figura 5.3: Diagrama de la base

Estas ecuaciones corresponden a la relación entre el sistema de referencia del robot y un sistema de referencia base el cual puede ser el sistema de referencia de donde partió el robot.

En la ecuación 3.3 se observa el sistema que describe la velocidad del robot respecto al sistema base y cuya entradas son las velocidades del robot respecto a su sistema de referencia.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ \omega \end{bmatrix} \tag{5.4}$$

En las ecuaciones 5.5-5.7 se observa el cálculo del error entre las variables x, y y θ y la configuración deseada. El cálculo del error es sencillo debido a que no hay que planear trayectorias y además la base omnidireccional permite que el robot se mueva en cualquier dirección.

$$E_x = x_d - x \tag{5.5}$$
$$E_y = y_d - y \tag{5.6}$$
$$E_\theta = \theta_d - \theta \tag{5.7}$$

Para obtener el modelo del robot realimentado se plantean las entradas de control como las velocidades V_{bx}, V_{by} y ω_b que representan las velocidades en x, y y la velocidad angular del robot respecto a su propio sistema de referencia. En la ecuación 5.8 se observan las entradas del sistema.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} V_{bx} \\ V_{by} \\ \omega_b \end{bmatrix} \tag{5.8}$$

Sin embargo, las señales de control en la ecuación 5.4 no se encuentran en términos de V_{bx}, V_{by}, ω_b por lo que se debe hacer una rotación que permita tener a las entradas V_x, V_y y ω en términos de las variables u_1, u_2 y u_3, esto se observa en la ecuación 5.9.

$$\begin{bmatrix} V_x \\ V_y \\ V_\theta \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{5.9}$$

Al sustituir las entradas de la ecuación 5.9 en la ecuación cinemática del sistema se obtiene la ecuación 5.10

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \tag{5.10}$$

La multiplicación de las matrices de rotación se puede simplificar como:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} V_{bx} \\ V_{by} \\ \omega_b \end{bmatrix} \tag{5.11}$$

A continuación se propone la siguiente ley de control para obtener las velocidades deseadas en el robot:

- Control proporcional $u = k_p E$

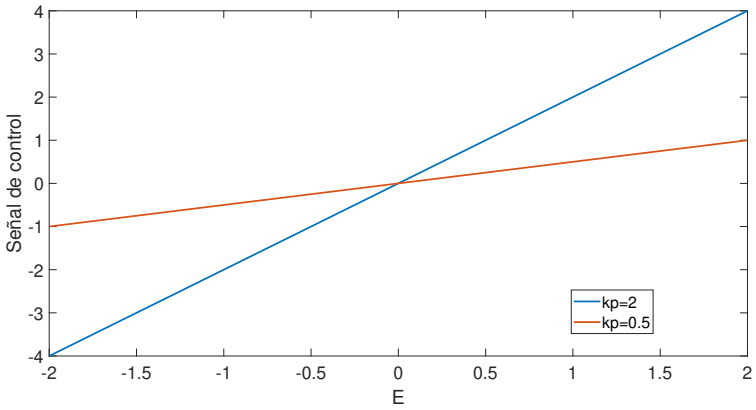


Figura 5.4: Señales de control lineales con diferentes k_p

De acuerdo a esta ley las entradas u_1, u_2 y u_3 del sistema se calculan como se observa en la ecuación 5.12

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} k_{p1} & 0 & 0 \\ 0 & k_{p2} & 0 \\ 0 & 0 & k_{p3} \end{bmatrix} \begin{bmatrix} E_x \\ E_y \\ E_\theta \end{bmatrix} \tag{5.12}$$

Al sustituir la ecuación 5.12 en la ecuación 5.11 se obtiene la ecuación 5.13 que representa el sistema realimentado por una entrada de control proporcional al error.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} k_{p1} & 0 & 0 \\ 0 & k_{p2} & 0 \\ 0 & 0 & k_{p3} \end{bmatrix} \begin{bmatrix} E_x \\ E_y \\ E_\theta \end{bmatrix} \tag{5.13}$$

Para analizar la estabilidad del sistema conviene realizar un cambio de variable, de las ecuaciones 5.5-5.7 se despejan las variables x, y y θ como se observa en las ecuaciones 5.14-5.16.

$$x = x_d - E_x \tag{5.14}$$
$$y = y_d - E_y \tag{5.15}$$
$$\theta = \theta_d - E_\theta \tag{5.16}$$

Al derivar se tienen las siguientes ecuaciones 5.17-5.19 y al sustituir las derivadas del error en la ecuación 5.11 se obtiene la ecuación 5.21.

$$\dot{x} = -\dot{E}_x \tag{5.17}$$
$$\dot{y} = -\dot{E}_y \tag{5.18}$$
$$\dot{\theta} = -\dot{E}_\theta \tag{5.19}$$
$$\tag{5.20}$$

$$\begin{bmatrix} \dot{E}_x \\ \dot{E}_y \\ \dot{E}_\theta \end{bmatrix} = - \begin{bmatrix} k_{p1} & 0 & 0 \\ 0 & k_{p2} & 0 \\ 0 & 0 & k_{p3} \end{bmatrix} \begin{bmatrix} E_x \\ E_y \\ E_\theta \end{bmatrix} \tag{5.21}$$

En la ecuación 5.21 se tiene una matriz de la forma $\dot{e} = Ae$, se puede analizar la estabilidad de este sistema por medio de sus polos; los polos del sistema se pueden calcular como los valores propios de la matriz A.

De la ecuación 5.21 se observa que los valores propios de la matriz son:

$$\lambda_1 = -k_{p1} \qquad \lambda_2 = -k_{p2} \qquad \lambda_3 = -k_{p3}$$

El sistema es asintóticamente estable si sus polos se encuentran en la parte negativa del plano complejo, por lo tanto, las condiciones para que el sistema sea asintóticamente estable son:

$$k_{p1} > 0 \qquad k_{p2} > 0 \qquad k_{p3} > 0$$

Los servomotores tienen una zona muerta, para el caso de los Dynamixel MX 12-W está zona se encuentra entre -0.0959[rad/s] y 0.0959[rad/s][1], si la señal de control se encuentra en esta zona el servomotor no realizará ningún movimiento.

[1] https://emanual.robotis.com/docs/en/dxl/mx/mx-12w/

El problema del control proporcional es que los servomotores comienzan a avanzar muy lento para finalmente dejar de avanzar al llegar a la zona muerta, esto sucede a una distancia considerable de la configuración deseada.

La solución a esta pérdida de velocidad en los servomotores, es incrementar la ganancia proporcional en las señales de control, sin embargo, el problema de esta solución es que el robot arranca con una señal muy grande en magnitud por lo cual las ruedas se patinan, esto produce errores en la odomentría y provoca que la base móvil no llegué a la configuración deseada.

Como solución se propone la siguiente señal de control:

- Control no lineal $u = k_p\sqrt{|E|}sign(E)$

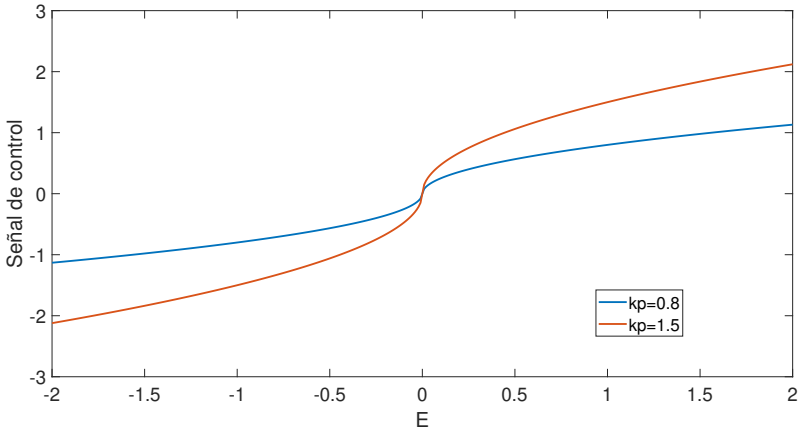


Figura 5.5: Señales de control no lineales con diferentes k_p

De las figuras 5.4 y 5.5 se observa que el control no lineal tiene un arranque más suave ya que, aunque el error sea grande la velocidad no aumenta tanto en magnitud como es el caso del control lineal. La otra diferencia es que el control lineal tiende a ser más lento que el control no lineal cuando el error se encuentra cerca de 0.

De acuerdo a esta ley no se puede analizar al sistema de forma matricial porque al sustituir las entradas se tiene un sistema no lineal. Para este control el comportamiento del sistema realimentado es descrito por las siguientes ecuaciones:

$$\dot{x} = k_{p1}\sqrt{|E_x|}sign(E_x) \tag{5.22}$$

$$\dot{y} = k_{p2}\sqrt{|E_y|}sign(E_y) \tag{5.23}$$

$$\dot{\theta} = k_{p3}\sqrt{|E_\theta|}sign(E_\theta) \tag{5.24}$$

Al igual que en el control proporcional lineal es conveniente analizar el sistema de acuerdo al error, sustituyendo las ecuaciones 5.17-5.19 en las ecuaciones 5.22-5.24 se tienen las siguientes ecuaciones:

$$\dot{E}_x = -k_{p1}\sqrt{|E_x|}sign(E_x) \tag{5.25}$$

$$\dot{E}_y = -k_{p2}\sqrt{|E_y|}sign(E_y) \tag{5.26}$$

$$\dot{E}_\theta = -k_{p3}\sqrt{|E_\theta|}sign(E_\theta) \tag{5.27}$$

Las ecuaciones 5.25-5.27 son las mismas con diferentes variables por lo que solo se analizará de forma general la ecuación 5.28 y se concluirá sobre los estados E_x, E_y y E_θ a partir de ella.

$$\dot{e} = -k_p\sqrt{|e|}sign(e) \tag{5.28}$$

Al ser un sistema no lineal para determinar su estabilidad se usará el método directo de Lyapunov.

Se propone una función definida positiva como función candidata de Lyapunov, en este caso será:

$$V = \frac{1}{2}e^2$$

Al derivar respecto a la variable e se obtiene la ecuación 5.29.

$$\dot{V} = \frac{dV}{de} = e\dot{e} \tag{5.29}$$

Evaluando al sistema en la derivada de la función definida positiva se obtiene la ecuación 5.30

$$\dot{V} = -k_p e\sqrt{|e|}sign(e) \tag{5.30}$$

El método directo de Lyapunov concluye estabilidad a partir de las siguientes condiciones;

- Si $\dot{V} \leq 0$ el sistema es estable
- Si $\dot{V} < 0$ el sistema es asintóticamente estable [18]
- Otro caso: No se puede concluir estabilidad y se debe probar con otra función candidata de Lyapunov.

Evaluando los signos del error se tiene lo siguiente:
Si $e < 0$

$$\dot{V} = -sign(k_p)(-)(+)(-) \tag{5.31}$$

Si $e < 0$

$$\dot{V} = -sign(k_p)(+)(+)(+) \tag{5.32}$$

De las ecuaciones 5.31 y 5.32 se observa que para que el sistema sea estable se debe cumplir $k_p > 0$.

El método directo de Lyapunov no puede concluir estabilidad asintótica porque cuando se sustituye $e = 0$ en la ecuación 5.30 se obtiene $\dot{V} = 0$

Por otro lado, el corolario de la Salle puede concluir estabilidad asintótica a partir de estabilidad en Lyapunov si se cumple que el único punto donde no se mueve la trayectoria, es decir $\dot{V} = 0$ es el punto en el que la variable es 0, en este caso la variable es e [18].

Por lo tanto, al generalizar la ecuación 5.30 en todos los estados del sistema se garantiza estabilidad asintótica si se cumplen las siguientes condiciones:

$$k_{p1} > 0 \qquad k_{p2} > 0 \qquad k_{p3} > 0$$

Simulación

En simulink se tiene el diagrama de la figura 5.6, la función *Sistema_Robot* corresponde a la dinámica del robot y contiene la ecuación 5.4, en la función *Control* se calculan las señales de control V_x, V_y y ω de la ecuación 5.9 sustituyendo las señales de control propuestas. Además se tiene un switch para elegir la entrada de control proporcional o la entrada de control no lineal.

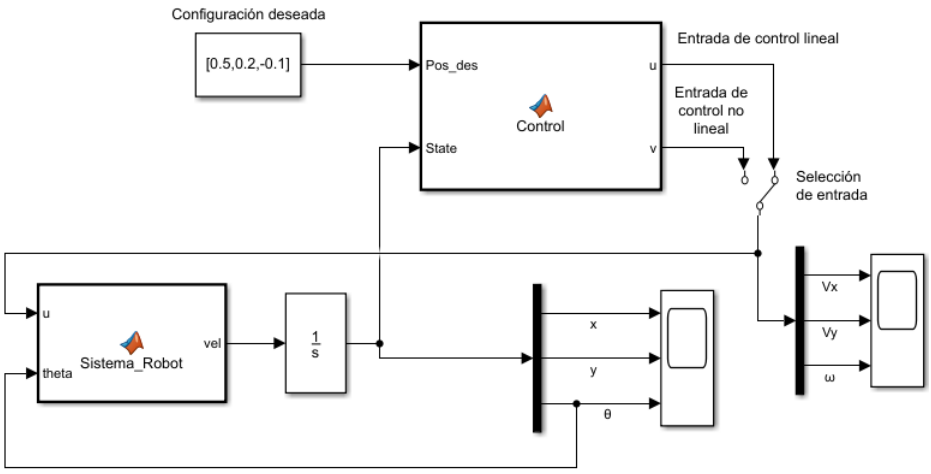


Figura 5.6: Simulación

En la simulación se tienen las siguientes condiciones iniciales:

$$\begin{bmatrix} x(0) \\ y(0) \\ \theta(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.33}$$

Y se busca tener la siguiente configuración:

$$\begin{bmatrix} x_d \\ y_d \\ \theta_d \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.2 \\ -0.1 \end{bmatrix} \tag{5.34}$$

- Control proporcional

Usando las ganancias

$$k_{p1} = 4 \qquad k_{p2} = 9.6 \qquad k_{p3} = 5$$

En la figura 5.7 se muestran los resultados del control en los estados del sistema y en la figura 5.7 se muestran las entradas de control que se obtuvieron con la ley de control proporcional propuesta.

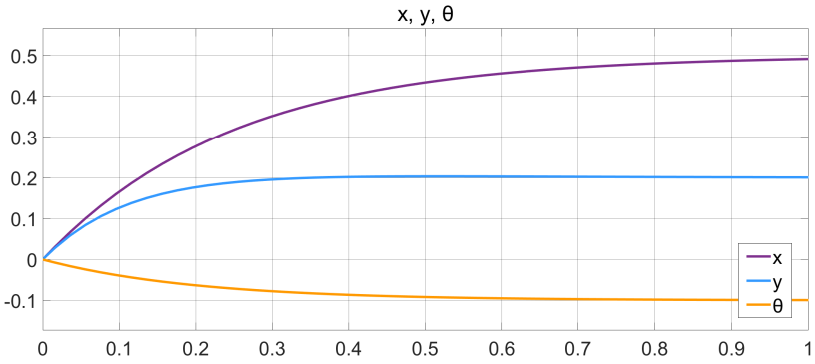


Figura 5.7: Estados del sistema con el control proporcional

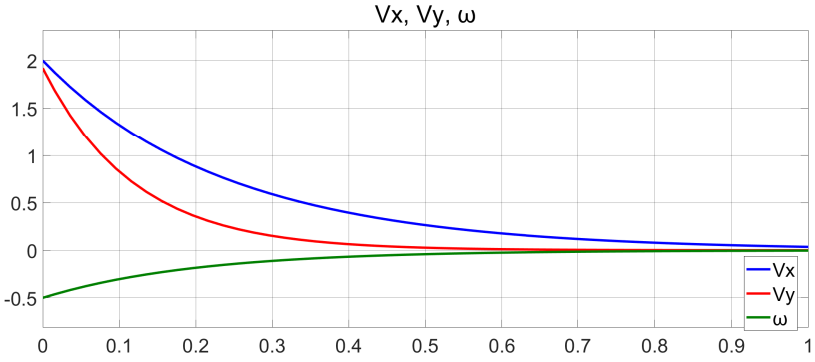


Figura 5.8: Entradas del control proporcional

- Control no lineal

Usando las ganancias

$$k_{p1} = 2.6 \qquad k_{p2} = 4 \qquad k_{p3} = 1.6$$

En la figura 5.9 se muestran los resultados del control en los estados del sistema y en la figura 5.9 se muestran las entradas de control que se obtuvieron con la ley no lineal propuesta.

Se observa que la ley de control no lineal llega más rápido que la ley proporcional a la configuración deseada, asimismo la magnitud máxima de V_x y V_y es menor en las leyes de control no lineal, sumando estas ventajas a los problemas ya mencionados del control proporcional (la base se patina y se tiene una zona muerta), se prefiere utilizar la ley de control no lineal propuesta.

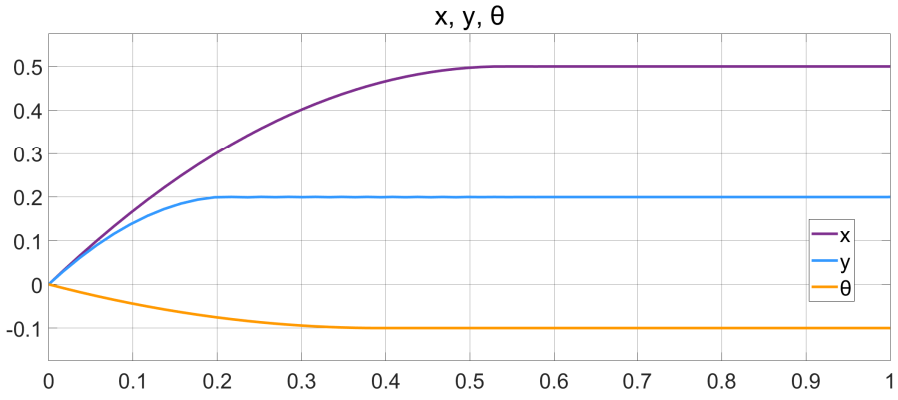


Figura 5.9: Estados del sistema no lineal

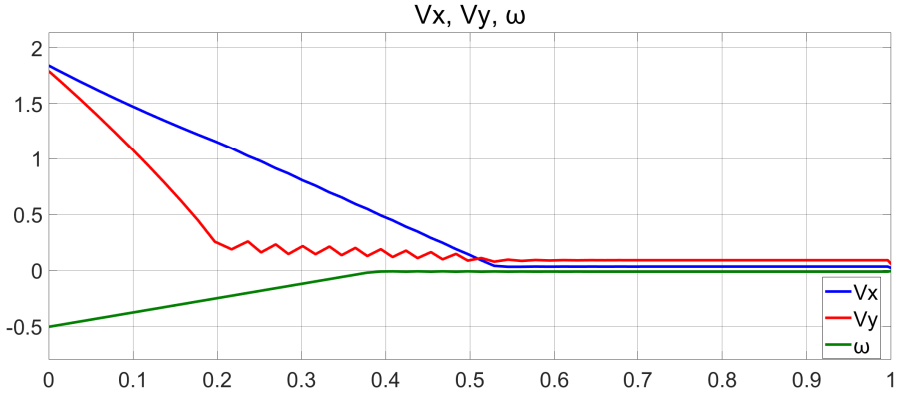


Figura 5.10: Entradas de control no lineal

5.2. Criterio para iniciar el movimiento

Después de detectar que la pelota se encuentra en movimiento y superar cierto número de muestras, cada vez que la cámara tome una nueva muestra de la posición se estimará la posición de caída con las muestras almacenadas, esperando obtener una mejor estimación al tener más muestras.

El problema es qué no se sabe en que momento la estimación es suficientemente buena para mover al robot a un determinado punto en el espacio.

Al tener más datos de las mediciones de la pelota se espera que la estimación de la caída converja en algún punto como se observa en las figuras 5.11 y 5.12.

Las trayectorias de las figuras 5.11 y 5.12 se calcularon usando mínimos cuadrados con la información de la tabla 3.2.

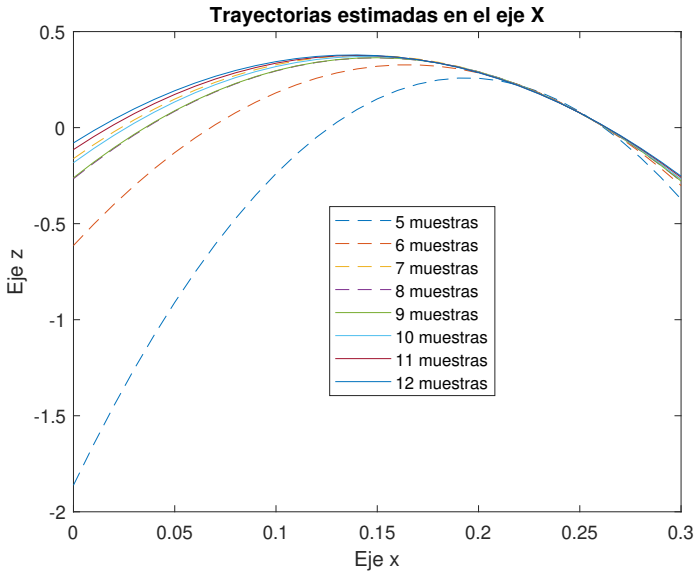


Figura 5.11: Estimación de trayectoria con distintos números de muestras en el eje X

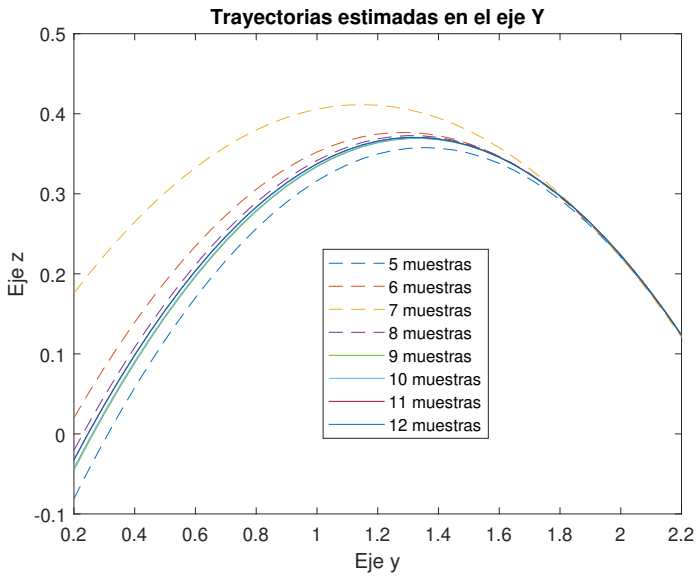


Figura 5.12: Estimación de trayectoria con distintos números de muestras en el eje Y

Se propone un criterio y a partir de su cumplimiento el robot se moverá. En el criterio propuesto se almacena la estimación anterior de la caída y la se compara con la estimación actual de la caída. Cuando la diferencia entre las coordenadas de ambas estimaciones en valor absoluto es menor a un umbral, el cual será un parámetro de diseño, el robot se mueve a las coordenadas de la última estimación.

Al tener 2 coordenas (x,y) en la estimación de la caída se deben cumplir 2 condiciones:

$$|\hat{X}_{i-1} - \hat{X}_i| < \gamma \tag{5.35}$$

$$|\hat{Y}_{i-1} - \hat{Y}_i| < \gamma \tag{5.36}$$

Las ecuaciones 5.35, 5.36 representan el criterio para mover al robot siendo $\hat{X}_i$ la estimación de la caída actual en el eje X, $\hat{X}_{i-1}$ la estimación anterior en el eje X, $\hat{Y}_i$ la estimación de la caída actual en el eje Y, $\hat{Y}_{i-1}$ la estimación anterior en el eje Y y γ es un umbral de diseño.

En este capítulo se obtuvo el modelo cinemático de la base móvil, con base a este se propusieron y analizaron algunas leyes de control necesarias para mover al robot a la posición deseada y se determinó cuáles son mejores para aplicar en el robot real, además, se propuso un criterio para empezar a mover al robot.

En el próximo capítulo, se implementará lo que se vio en este capítulo así como el sistema de visión visto previamente con ayuda de ROS y las ventajas de este.

Capítulo 6

Integración mediante la plataforma ROS

El en presente capítulo se aborda la integración del sistema de visión y el sistema de control en la base móvil diseñada utilizando ROS. En la sección 6.1 se describe brevemente en que consiste ROS, sus ventajas sobre el código tradicional en códigos extensos. La forma en la que se implementaron las tareas del sistema utilizando diferentes nodos en ROS se explica en la sección 6.2.

6.1. Plataforma ROS

Robot Operating System mejor conocido como ROS es definido por Koubaa[7] como:

> "Un middleware de código abierto para el desarrollo a gran escala de sistemas robóticos complejos"

ROS implementa el mecanismo de comunicación "publicar-suscribir"por medio de tópicos, los tópicos son mensajes unidireccionales y asíncronos. Existen diferentes tipos de tópicos el tipo de tópico define la estructura del mensaje, por ejemplo un tópico de tipo Point siempre va contener coordenadas (x, y, z).

Los nodos son cualquier proceso que este usando ROS y se identifique con un nombre. Existe un nodo maestro usualmente conocido como *roscore* que se encarga de registrar a los nodos con un nombre y los tópicos a los que está suscrito y/o publica. Los nodos se pueden comunicar entre diferentes dispositivos cuando el mismo nodo maestro administra a todos los dispositivos, esto se logra exportando al nodo maestro.

Un nodo puede publicar un tópico cada cierto tiempo o después de que se cumplan ciertas condiciones definidas en ese nodo. Cuando un nodo está suscrito a un tópico y este tópico es publicado por otro nodo, en ese momento el nodo suscrito suele ejecutar ciertas acciones con la información del tópico al que está suscrito.

En la figura 6.1 se comparan 2 códigos, a la izquierda se tiene uno implementado de manera tradicional y a la derecha se tiene un código implementado con ROS. En el código de la derecha se calculan las variables A, B y C, posteriormente se obtiene un resultado al realizar operaciones con dichas variables. En el caso del código de la izquierda se dividen todas las tareas del código de la derecha, los nodos se suscriben a las variables que necesitan y publican variables que necesitan otros nodos.

Las ventajas de ROS son:

- Menor extensión de código:

  En cada nodo del lado derecho de la figura 6.1 se tiene una menor extensión de código si se le compara con el código del lado izquierdo.

- Mayor facilidad en la detección de errores:

  ROS contiene comandos como *rostopic info*, *rostopic hz*, *rostopic echo* que permiten visualizar la información y frecuencia de los tópicos en tiempo real, además el código de un nodo es independiente de otro, por estas razones resulta más sencillo encontrar errores. Por ejemplo en la figura 6.1, si se observa al tópico r_2 publicado y se detecta un dato no deseado y al mismo tiempo se observa al tópico r_1 sin detectar datos erróneos, es fácil deducir que el problema viene del cálculo de C.

- Mantenimiento:

  Al tener a cada código aislado resulta sencillo optimizar o adaptar algún nodo en el futuro.

- Diversidad de lenguajes:

  ROS permite programar los nodos en python y c++, no importa el lenguaje en el que estén programados los nodos, se pueden comunicar entre sí, por lo que se puede programar el nodo en el lenguaje que facilite la programación de la aplicación deseada.

Por las ventajas mencionadas se optó por trabajar utilizando ROS y aprovechar los nodos para disminuir la complejidad y extensión de ciertas funciones extensas que se implementarán.

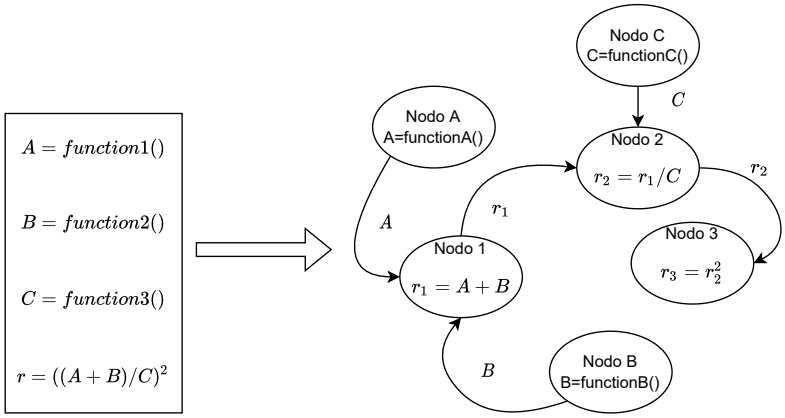


Figura 6.1: Comparación de un código extenso y un código equivalente en ROS

Otro concepto importante es el launch, un launch es un archivo en formato XML que permite levantar nodos, otros launches o herramientas como Rviz. Es útil porque se puede iniciar un sistema con un solo comando.

6.2. Implementación

En la figura 6.2 se observan los nodos propuestos para el desarrollo del proyecto, además de los nodos desarrollados se encuentra la *cámara RGB-D* y la *base móvil real/simulada*, se requiere levantar todo el sistema con un comando por lo que todos los nodos y elementos se encuentran dentro de un launch llamado *robot_autonomo*.

La *cámara RGB-D* es el sensor que proporciona una nube de puntos del ambiente y la *base móvil real/simulada* representa el conjunto de actuadores y sensores de la base móvil los cuales interactúan con los nodos, en este caso tanto actuadores como sensores se encuentran embebidos en los servomotores MX-12W.

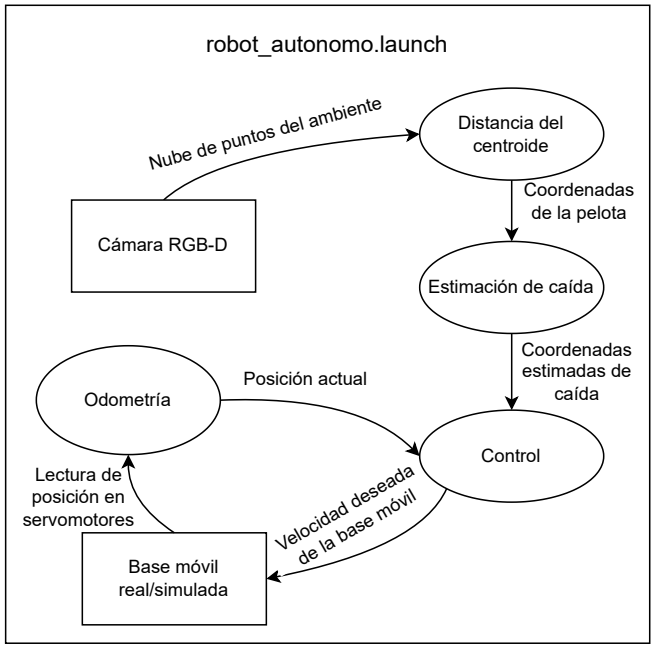


Figura 6.2: Implementación del código en varios nodos

6.2.1. Nodo que calcula la posición de la pelota

Este nodo se encuentra suscrito a un tópico de tipo PointCloud2 entregado por la *cámara RGB-D*.

El nodo publica un tópico de tipo Point que contiene las coordenadas (x,y,z) donde se estima que se encuentra el centroide de la pelota.

El código del nodo realiza una detección de la pelota mediante segmentación por color con los elementos vistos en la sección 3.1. Después de segmentar la información que pertenece a la pelota se hace un promedio de la distancia de profundidad de cada píxel entregada por la cámara. Este promedio representa las coordenadas estimadas del centroide de la pelota.

6.2.2. Nodo que estima las coordenadas de la caída de la pelota

El nodo está suscrito a un tópico de tipo Point que corresponde a las coordenadas en donde se estima se encuentra el centroide de la pelota, este tópico es entregado por el nodo 6.2.1.

El nodo publica un tópico de tipo Point con las coordenadas en donde se estima va a caer la pelota.

El código del nodo cuenta con una bandera que se activa al detectar el movimiento con la propuesta de la sección 3.2.

Cuando se detecta el movimiento el nodo comienza a guardar las coordenadas que entregue el nodo 6.2.1, con la técnica vista en la sección 3.3 se estima la trayectoria de la pelota y se calculan las coordenadas de caída como se vio en la sección 3.4. No se publica la estimación de caída de la pelota hasta cumplir el criterio visto en la sección 5.2.

6.2.3. Nodo que calcula la odometría

Este nodo no está suscrito a tópicos, pero publica un tópico de tipo Point con las coordenadas de la base móvil.

Al levantar el nodo se establecen las coordenadas (x, y, z) de la base móvil como $0, 0, 0$. En este caso se usa la coordenada z para determinar la orientación de la base móvil, el ángulo en z se expresa en [rad].

La orientación y coordenadas de la base móvil se actualizan a 30 [Hz] leyendo los servomotores y usando las ecuaciones vistas en la sección 4.5.

6.2.4. Nodo de control

Este nodo se encuentra suscrito a 2 tópicos de tipo Point publicados por los nodos 6.2.2 y 6.2.3.

El nodo publica un tópico de tipo Twist con las velocidades necesarias para mover a la base móvil al punto de interés.

En este nodo se realiza el cálculo de las velocidades de control utilizando las leyes de control vistas en la sección 5.1 y posteriormente se escriben dichas velocidades a 30[Hz] en los servomotores como se indica en la sección 4.4, las velocidades también son publicadas.

Para el cálculo del error las coordenadas (x_d, y_d) son las coordenadas correspondientes a dichos ejes en el tópico Point publicado por el nodo 6.2.2, la orientación θ_d se puede ajustar en este nodo y las coordenadas actuales x, y, θ son actualizadas por el nodo 6.2.3.

6.2.5. Simulación de la base móvil

Como resultado de dividir las tareas del sistema en nodos el sistema necesita una imagen para llevar a cabo todo el proceso necesario.

La imagen puede ser transmitida por el sensor o puede ser guardada y posteriormente transmitida con la herramienta *rosbag*. *Rosbag* permite guardar los datos de uno o varios tópicos con el comando *rosbag record*, estos datos se pueden volver a transmitir con el comando *rosbag play* en cualquier momento, respetando el orden en el que fueron publicados.

Los algoritmos implementados en este proyecto son transparentes al hardware debido a que se puede reemplazar la base móvil por un nodo que calcule la odometría ideal de la base móvil a partir de las velocidades deseadas por lo que el nodo se encuentra suscrito a un tópico de tipo Twist publicado por el nodo 6.2.4.

Con el fin de visualizar el movimiento de la base móvil por medio de Rviz, el nodo publica tópicos de tipo Marker que se asemejan a la base móvil y son publicados en las coordenadas de odometría ideal del robot, esta es calculada a partir de las velocidades deseadas.

Con fines de visualización el nodo también se encuentra suscrito a tópicos de tipo Point con las coordenadas publicadas por los nodos 6.2.1 y 6.2.2 . Se publica una esfera verde en las coordenadas del nodo 6.2.1 y se publica una flecha roja en las coordenadas publicadas por el nodo 6.2.2.

En la figura 6.3 se observa una nube de puntos guardada en *rosbag*, una esfera verde en las coordenadas donde estimamos se encuentra la pelota y una flecha roja en donde se estima va a caer la pelota.

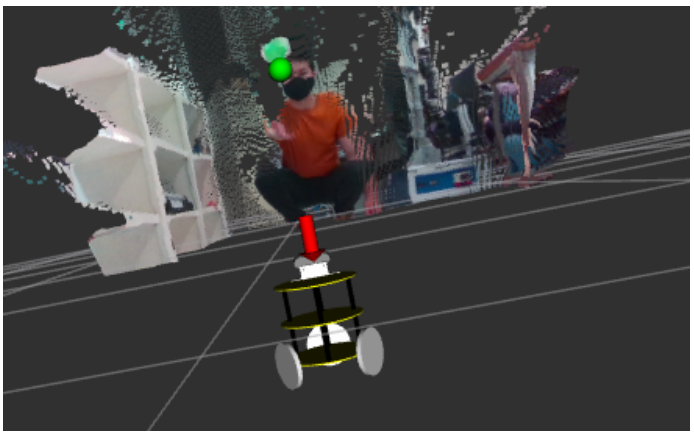


Figura 6.3: Captura visualizando en RVIZ la simulación

En este capítulo se describió en que consiste ROS sus ventajas y se planteó una forma de dividir el sistema en distintos nodos y se explicaron las tareas que realizará cada nodo.

En el siguiente capítulo se mostrarán los resultados obtenidos a partir de la implementación del sistema utilizando los nodos descritos en este capítulo y la base móvil construida en la sección 4.

Capítulo 7

Resultados

El presente capítulo contiene la información recabada al momento de realizar pruebas tanto simuladas como reales. En la sección 7.1 se plantea el método utilizado para obtener el rango de color que se desea segmentar. La sección 7.2 describe los datos atípicos encontrados al realizar mediciones, se analizan las consecuencias de estos y se propone una solución. Las constantes utilizadas en las pruebas se especifican en la sección 7.3 y se proporciona una breve explicación de cómo se obtuvieron. En las secciones 7.4 y 7.5 se muestran los resultados obtenidos en simulación y realizando pruebas reales usando la base móvil respectivamente.

7.1. Rango de color

Para obtener los límites de la segmentación se toma una foto de la pelota y se selecciona un área dentro de un rectángulo como se muestra en la figura 7.1, en esta área se buscan los valores mínimos y máximos de HSV en los píxeles, también puede resultar útil realizar un promedio de estos pues en caso de que algún límite se encuentre muy alejado del promedio este puede reducirse a criterio de quien realice las pruebas con la finalidad de ser más selectivos y no obtener píxeles que no correspondan al objeto en la segmentación.

El rango HSV obtenido se modifica en el nodo 6.2.1, es conveniente realizar este procedimiento antes de realizar pruebas ya que, aunque siempre se utilice el mismo objeto se pueden obtener pequeñas variaciones dependiendo de la luz que exista en el ambiente.

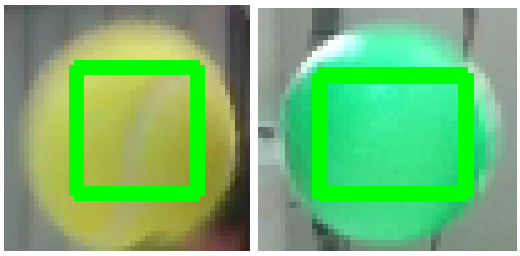


Figura 7.1: Área de interés para obtener limites de segmentación

En las tablas 7.1 y 7.2 se observan los valores calculados a partir de las áreas seleccionadas en la figura 7.1.

	Límite inferior	Límite superior	Promedio
H	0.977384[rad]	1.22173[rad]	1.11701[rad]
S	0.298	0.67	0.537
V	151	219	184
Color			

Tabla 7.1: Límites y promedio de los valores HSV de la pelota que se encuentra en el lado izquierdo en la figura 7.1

	Límite inferior	Límite superior	Promedio
H	2.54818[rad]	2.82743[rad]	2.61799[rad]
S	0.494	0.941	0.776
V	195	255	230
Color			

Tabla 7.2: Límites y promedio de los valores HSV de la pelota que se encuentra en el lado derecho en la figura 7.1

En la tabla 7.1 se observa un límite inferior en la saturación S que se encuentra muy por debajo del promedio por lo cual podría tratarse de un valor atípico y si este valor afecta la segmentación podría aumentarse a 0.35 u otro valor que se considere conveniente.

7.2. Selección de datos

Después de hacer varias pruebas con la cámara se notó que algunas veces se presentaba un error en las mediciones lo cual siempre daba distancias muy grandes que afectaban la estimación de la trayectoria.

Lo primero que se hizo para evitar este error fue calibrar la cámara, sin embargo el error persistió cuando la pelota atravesaba cierta área del espacio, por lo que se buscó otro método para resolver este inconveniente.

Ejemplo

En la tabla 7.3 se muestran mediciones reales tomadas en una prueba por la cámara RGB-D y se puede observar que las mediciones de x_3, x_4 y x_5 corresponden a valores atípicos(outlier) dado que la pelota fue lanzada en la dirección que se muestra en la figura 7.2 por lo que las mediciones en x debieron disminuir a medida que avanzaba el tiempo.

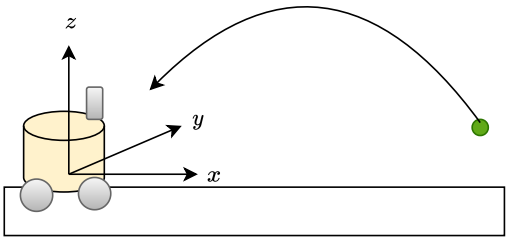


Figura 7.2: Dirección en la que fue lanzada la pelota

	Mediciones		
i	x_i	y_i	z_i
0	1.954699013	-0.083179308	0.390366483
1	1.82710528	-0.090148529	0.463068349
2	1.835327227	-0.094121849	0.565250028
3	2.181078582	-0.122785905	0.795838172
4	3.55373785	-0.232134015	1.47786597
5	5.620126704	-0.47988826	2.524233895
6	1.175514508	-0.091351613	0.593638155
7	1.101856687	-0.092529336	0.603398714
8	1.013738292	-0.09286294	0.603558364
9	0.931945432	-0.090545234	0.596182491
10	0.856861844	-0.09603665	0.584938547
11	0.773545765	-0.095414176	0.558400652
12	0.690164966	-0.093011244	0.524839687
13	0.615097738	-0.092065606	0.47453854
14	0.542874168	-0.091662553	0.42064602
15	0.470866352	-0.085306571	0.364285837
16	0.391147659	-0.085515052	0.298490126
17	0.313343397	-0.080862896	0.230062294
18	0.237204071	-0.08186428	0.151478463

Tabla 7.3: Muestras de posición tomadas por la cámara RGB-D, los datos atípicos se encuentran en rojo

El problema principal de tener mediciones erróneas es que afectan a la esti-

mación por mínimos cuadrados de la trayectoria como se observa en las figuras 7.4 y 7.3.

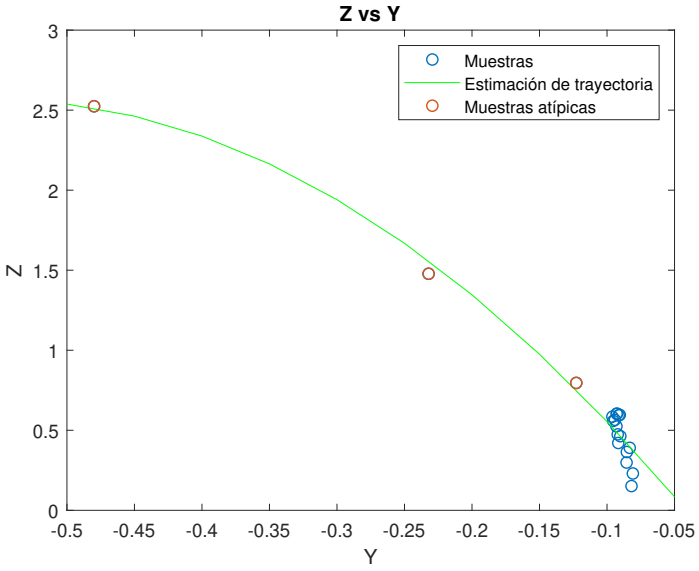


Figura 7.3: Estimación de trayectoria en Y con valores atípicos

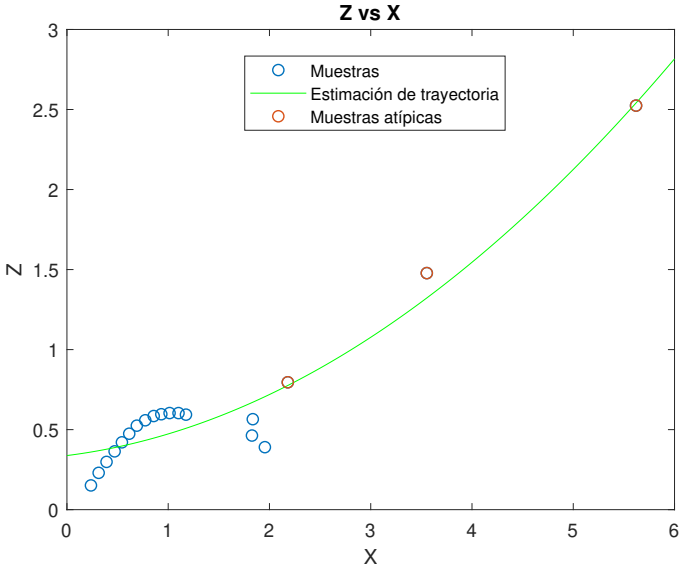


Figura 7.4: Estimación de trayectoria en X con valores atípicos

Para resolver el problema de los datos atípicos se partió de que al arrojar la pelota ésta se tuvo que acercar al robot, por lo tanto, la mayor distancia en los datos almacenados debe ser la primera muestra.

La distancia de cada muestra se calcula como:

$$d_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$$

Para eliminar datos atípicos se propusieron la siguientes condiciones:

- Si $d_i > d_0$: Se elimina la muestra
- Otro caso: La muestra se conserva

En la tabla 7.4 se muestran las distancias de cada muestra y en la columna eliminar dato se observa si la muestra se elimina con el fin de mejorar la estimación de la trayectoria.

	Mediciones	
i	d_i	**Eliminar dato**
0	1.995032085	✗
1	1.82710528	✗
2	2.324981409	✓
3	3.855778356	✓
4	6.179633781	✓
5	1.320070352	✗
6	1.259658622	✗
7	1.183457455	✗
8	1.110024455	✗
9	1.041915717	✗
10	0.958795183	✗
11	0.872029512	✗
12	0.782309484	✗
13	0.69286179	✗
14	0.601412091	✗
15	0.499405317	✗
16	0.391147659	✗
17	0.397053587	✗
18	0.293109632	✗

Tabla 7.4: Distancia de las muestras

En las figuras 7.6 y 7.5 se muestran las trayectorias estimadas con datos atípicos y sin datos atípicos, se puede notar una mejor aproximación a las muestras al eliminar los datos atípicos.

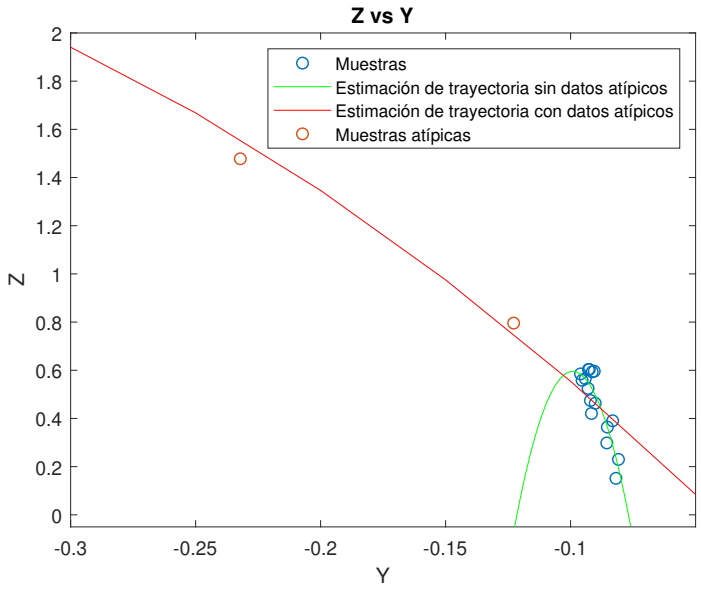


Figura 7.5: Comparación de estimaciones de trayectoria en el eje Y

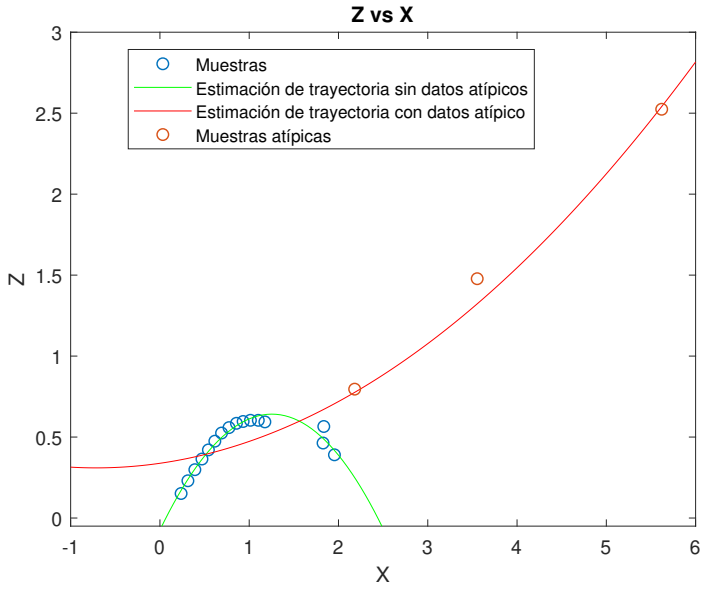


Figura 7.6: Comparación de estimaciones de trayectoria en el eje X

La propuesta para la selección de datos se agregó al nodo 6.2.2.

7.3. Parámetros de diseño

En la tabla 7.5 se muestran los parámetros usados en el proyecto.

Símbolo	Nombre	Estructura usada	Referencia
B	Kernel de erosión	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	3.1.2
R_z	Rotación de la cámara sobre el eje z	π[rad]	3.1.3
R_y	Rotación de la cámara sobre el eje y	$\pi/9$[rad]	3.1.3
T_x	Traslación de la cámara en el eje x	5[cm]	3.1.3
α	Umbral para detectar movimiento en la pelota	0.06[m]	3.2
ρ	Saturación en los servomotores	95 [rad/s]	4.4
k_{p1}	Ganancia proporcional 1	2	5.1.2
k_{p2}	Ganancia proporcional 2	2	5.1.2
k_{p3}	Ganancia proporcional 3	2	5.1.2
γ	Umbral para iniciar el movimiento del robot	0.08[m]	5.2
θ_d	Orientación deseada	0 [rad]	6.2.4
h_i, s_i, v_i	Límite inferior HSV	(2.44346[rad], 0.667, 160)	7.1
h_s, s_s, v_s	Límite superior HSV	(2.61799[rad], 0.98, 253)	7.1

Tabla 7.5: Parámetros usados

Obtención de parámetros

- B: Se hicieron pruebas con varias estructuras y se seleccionó la que tenía un mejor balance entre eliminar ruido y no perder demasiada información al momento de aplicar la erosión.

- R_x, R_y, T_x: Como se observa en la figura 7.7, la cámara obtiene las coordenadas de la pelota de acuerdo al sistema de referencia $x_2y_2z_2$ pero se requiere obtener las coordenadas respecto al sistema de referencia $x_0y_0z_0$, para esto se realizó una rotación R_z de π sobre el eje z_2 y una rotación R_y de $\pi/9$ sobre el eje y_1, por último se realizó una traslación T_x de 5[cm] en el eje x_0 para tener las lecturas con respecto al centro de la base.

  La matriz de transformación homogénea que realizó el cambio del sistema de referencia 0 al sistema de referencia 2 queda de la siguiente manera:

$$ {}^{0}H_{2} = \begin{bmatrix} 0 & -\sin\frac{\pi}{9} & \cos\frac{\pi}{9} & 0.05 \\ -1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{9} & -\sin\frac{\pi}{9} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

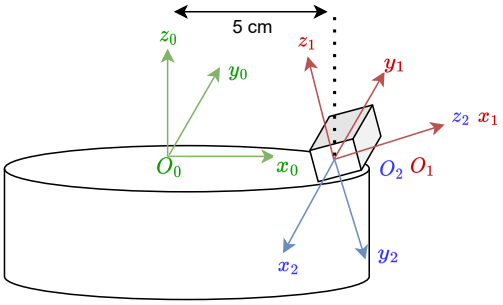


Figura 7.7: Sistemas de referencia usados para obtener la transformación homogénea

- α: El valor de este parámetro se debe a que en el peor de los casos cuando la pelota se encuentra estática la segmentación puede arrojar una mancha de color en un extremo de la pelota y en la siguiente lectura una mancha en el otro extremo como se observa en la imagen 7.8, la distancia entre estas muestras es el diámetro. La pelota utilizada tiene un diámetro de aproximadamente 8 cm, pero en las pruebas se observó que con este valor a veces se perdía el primer dato por lo que se redujo a 6 [cm].

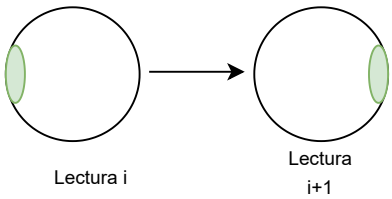


Figura 7.8: Se muestra el peor de los casos en la segmentación de color cuando la pelota se encuentra estática

- ρ: La saturación de los servomotores es el máximo valor al cual se quiere que se muevan los servomotores, con las leyes de control implementadas no hubo necesidad de jugar con este valor, pero si se aumentan las constantes proporcionales puede ser de utilidad para evitar que se patine el robot.
- k_{p1}, k_{p2}, k_{p3}: Las ganancias proporcionales se obtuvieron de forma experimental iterando con distintos valores, al final se prefirió la respuesta de

los valores seleccionados.

- γ: De este valor depende la precisión de las coordenadas estimadas en la caída, se obtuvo de manera experimental. Al aumentar el valor de γ se obtienen estimaciones menos precisas pero una reacción más rápida y viceversa, por lo que se debe ponderar si se prefiere precisión o velocidad de reacción.

- θ_d: Se prefiere tener una orientación que permita seguir observando la trayectoria de la pelota. Por el momento se decidió utilizar una orientación de 0 [rad].

- Límites HSV: Estos límites se obtuvieron como se indica en la sección 7.1, al tener condiciones similares de luz ambiental en cada prueba no se modificaron.

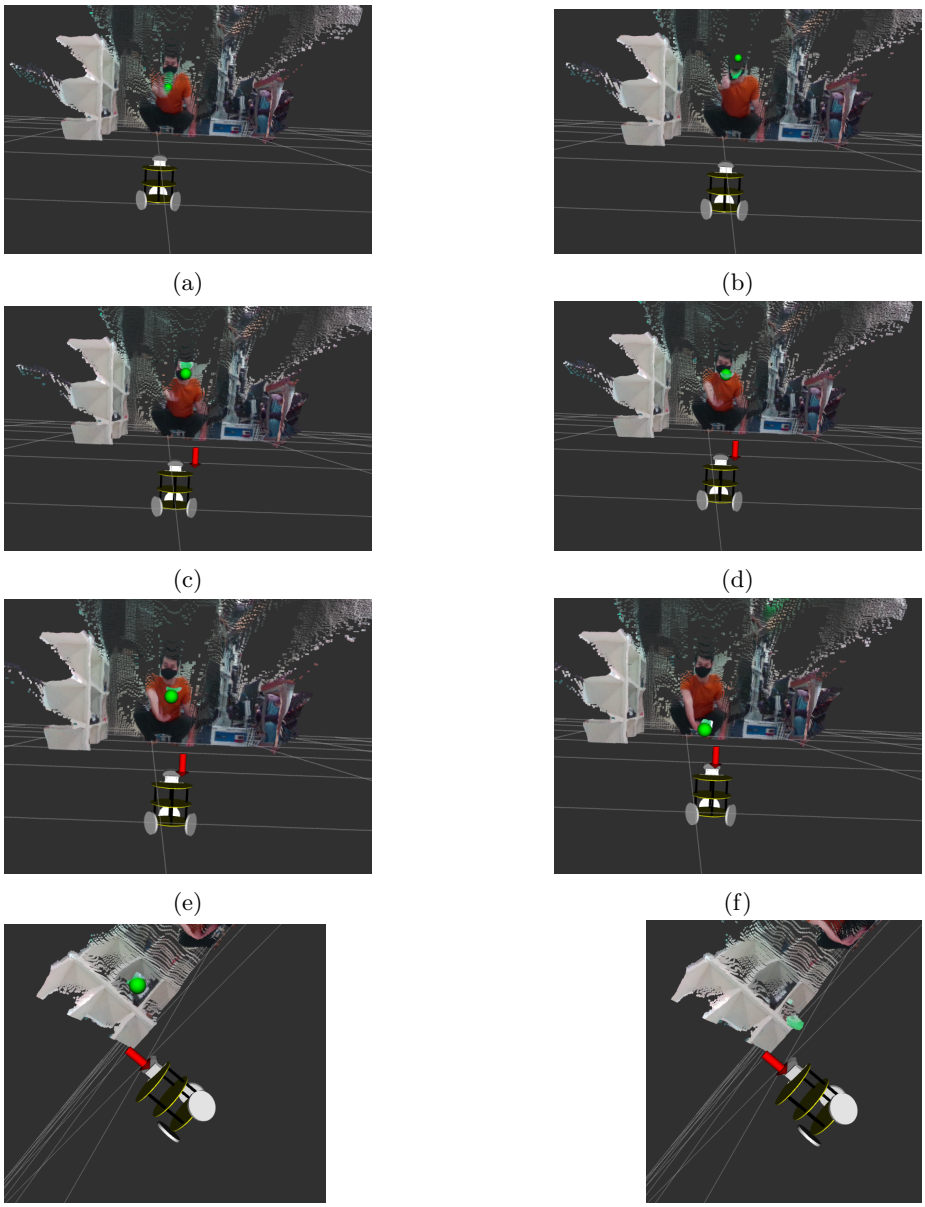
7.4. Pruebas simuladas

En la figura 7.9 se muestra el funcionamiento del sistema en una prueba realizada visualizando una nube de puntos grabada con *rosbag* y se simula el movimiento del robot, para visualizar esto se publican Markers en la posición del robot. Rviz permite visualizar lo que sucede en el sistema.

En los resultados de la simulación con datos reales se observa que se tiene una buena estimación de las coordenadas de caída, además el robot llegó lo suficientemente rápido a las coordenadas deseadas, sin embargo, se necesita llevar a cabo una corrección al llegar al punto deseado pues la estimación no es suficientemente buena para atrapar la pelota.

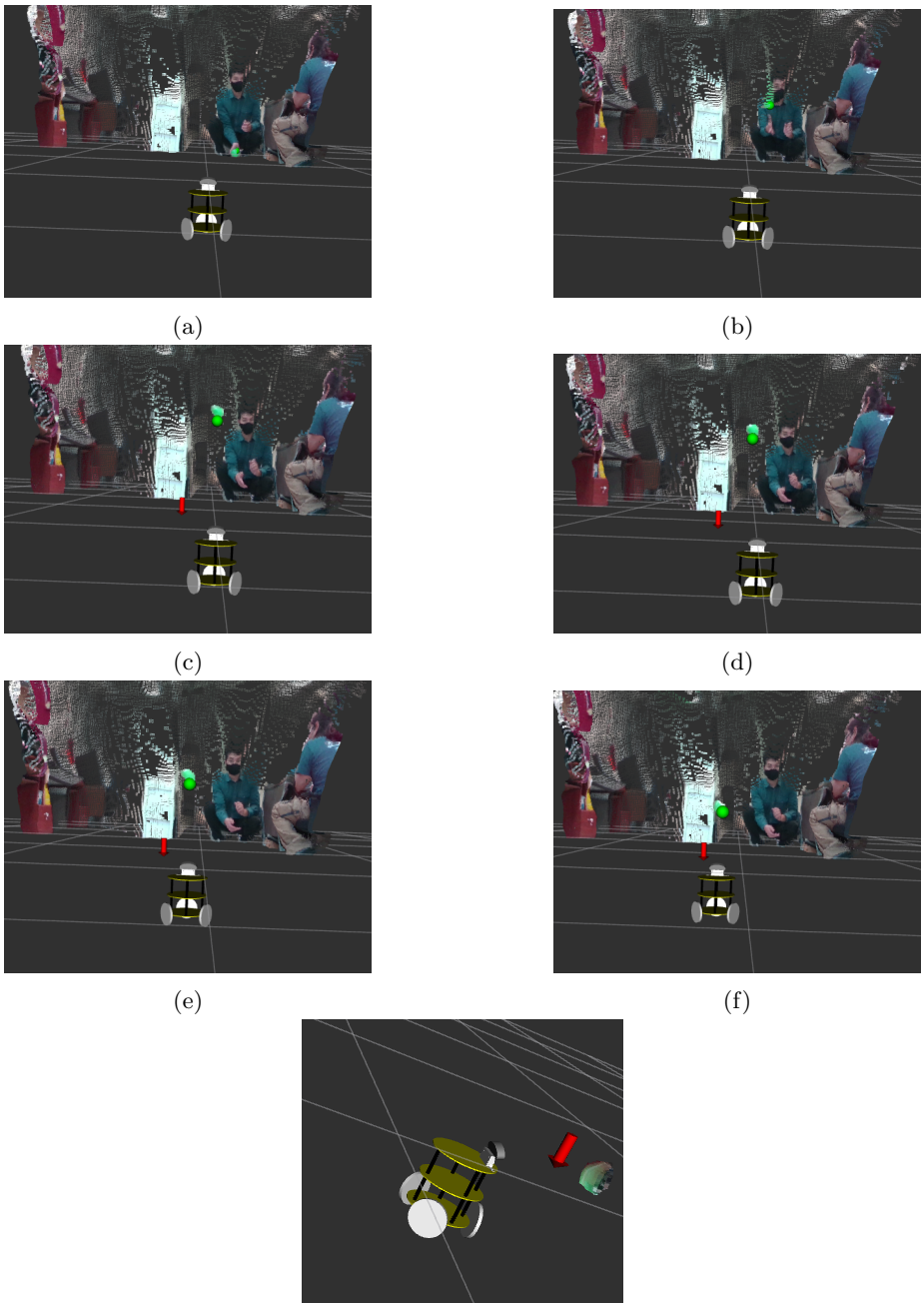
En la figura 7.10 se muestra otra prueba realizada. En la figura 7.10g se observa que la pelota cayó antes de que el robot pudiera llegar a la meta, además como en la prueba pasada se observa que se necesita corregir las coordenadas meta aun cuando el robot se encuentra en movimiento si se desea atrapar la pelota.

La cámara se encontraba montada en el robot cuando se capturaron los datos. Para ambas simulaciones los datos de la nube de puntos corresponden a un lanzamiento de la pelota de forma manual a aproximadamente 3[m] de distancia.



(a) (b) (c) (d) (e) (f) (g) (h)

Figura 7.9: Primera prueba simulada: **a)** La pelota es lanzada; **b)** La pelota se encuentra en movimiento; **c)** Se tienen suficientes lecturas para estimar la caída, la flecha roja que se observa corresponde a la estimación de la caída; **d)** Se observa que el robot se dirige a las coordenadas de caída; **e)** El robot y la pelota continúan moviéndose; **f)** Se observa que el robot ha llegado a la meta antes de la caída de la pelota; **g)** Robot visto desde otra perspectiva al llegar a las coordenadas de caída; **h)** Se observa que la pelota no va a caer en donde se había calculado



(a) (b) (c) (d) (e) (f) (g)

Figura 7.10: Segunda prueba simulada: **a)** La pelota es lanzada; **b)** La pelota se encuentra en movimiento; **c)** Se tienen suficientes lecturas para estimar la caída, la flecha roja que se observa corresponde a la estimación de la caída; **d)** Se observa que una ligera corrección en las coordenadas de caída; **e)** El robot comienza a moverse a las coordenadas de caída; **f)** Se observa que el robot no ha llegado a la meta y la pelota se encuentra a punto de caer; **g)** Robot llegando a las coordenadas de caída visto desde otra perspectiva

7.5. Pruebas reales

Las pruebas reales se realizaron en un rango de 2 a 3[m], la pelota se lanzó de forma manual estimando una trayectoria cercana al robot. En general en las pruebas reales se observó una latencia por la cual el robot tardaba en reaccionar y en la mayoría de los casos no estaba cerca de las coordenadas de caída antes de que la pelota cayera, la únicas excepciones en las que el robot cumplía su propósito era cuando se lanzaba muy cerca de la base como se observa en la figura 7.11, en esta figura también se observa que a pesar de tener una buena estimación el robot debe corregir su posición por medio de visión si es que se quiere atrapar a la pelota, de lo contrario puede terminar golpeando a la pelota con la cámara como se observa en la imagen.

Se observó que existe cierta latencia en el sistema por esta razón el robot tardaba en reaccionar y no llegaba a las coordenadas deseadas antes de la caída. Al realizar pruebas se observó que los servomotores reaccionaban rápido a las entradas de control y en la simulación el procesamiento realizado con la segmentación también fue rápido por lo que se atribuye la latencia a algún problema en la adquisición de datos.

Figura 7.11: Prueba real, se muestra la secuencia de movimiento de izquierda a derecha

En este capítulo se mostraron inconvenientes al realizar las pruebas, se planteó una solución a estos, se explicó cómo se obtuvieron los límites HSV para llevar a cabo la segmentación y se mostraron otros parámetros utilizados en el diseño del proyecto.

Al final de este capítulo se analizaron las pruebas simuladas y reales, se plantearon las dificultades que no se pudieron resolver y se analizarán posibles soluciones en el siguiente capítulo, además también se darán conclusiones con respecto a los resultados obtenidos y las hipótesis planteadas.

Capítulo 8

Discusión

8.1. Conclusiones

Se diseñó y construyó una base móvil omnidireccional con las características suficientes para probar el sistema de estimación de trayectoria, esto se logró investigando sobre cada uno de los componentes, planteando ciertas características como el tamaño deseado de la base móvil, posteriormente se examinaron los planos de cada componente para lograr la unión de estos a través de la creación de piezas, finalmente se ajustó el diseño de las piezas y se seleccionó el material de estas realizando pruebas.

Se desarrolló un sistema capaz de estimar una trayectoria a partir de imágenes RGB-D, esto se logró utilizando la cámara D-435i y obteniendo una nube de puntos a partir de esta, posteriormente al determinar los límites HSV del objeto de interés se realizó una segmentación por color y al obtener la posición estimada fue posible estimar una trayectoria en forma de ecuaciones cuadráticas utilizando la técnica de mínimos cuadrados. Resolviendo las ecuaciones de segundo grado se pudo calcular el punto de caída del objeto y con los criterios propuestos en las secciones 3.2 y 5.2 se determinaron condiciones para que el robot se trasladará al punto meta, sin embargo como se indicó en los resultados se tiene una latencia que limita el tiempo en el que el robot puede llegar al punto deseado y como consecuencia de esto fue imposible que el robot llegará antes de la caída de la pelota en la mayoría de las pruebas. En las pruebas donde se simuló el movimiento del robot utilizando datos reales reproducidos por *rosbag* también se detectó que existía un error en las coordenadas de estimación respecto al verdadero punto de caída, esto también dificultaba atrapar la pelota.

Se desarrolló un sistema de control de posición en el robot a partir del modelo cinemático del robot que se observa en la sección 5.4, se analizaron dos leyes de control y su estabilidad, comprobando su funcionamiento en simulación y finalmente se implementó en el robot realizando la comunicación necesaria con los servomotores para obtener la velocidad requerida. La entrada de control depende de la posición del robot, esta posición se logró estimar por medio del

cambio en la posición de los sensores ubicados en los servomotores.

Se implementaron los sistemas de control y estimación de trayectoria con ROS, esto se logró dividiendo los sistemas en nodos que disminuían la complejidad de todo el sistema y que permitía probar cada uno de estos por separado para realizar la detección y corrección de errores. El sistema implementado puede extrapolarse en otros robots de servicio para resolver el problema de evasión y manipulación de objetos en movimiento.

8.2. Trabajo futuro

Si se quiere atrapar un objeto se necesita un algoritmo que permita corregir el punto meta al que se dirige el robot mientras este se encuentra en movimiento y sigue capturando los datos de posición de la pelota con el fin de disminuir el error entre las coordenadas de caída estimadas y las reales.

Se debe encontrar y corregir la causa de la latencia en el sistema de visión de forma que el robot pueda estimar en un menor tiempo las coordenadas de caída en un área cercana, en cuanto a la rapidez con la que el robot llega al punto meta también se necesita plantear una ley de control que disminuya el tiempo de llegada sin que el robot se patine.

Es deseable estimar la trayectoria de objetos a una mayor distancia por lo que se puede reemplazar la cámara D-435i por la cámara D-455 que tiene un rango de hasta 6[m] de profundidad, si se quiere seguir trabajando con materiales de la misma familia de los cuales ya se sabe cómo funcionan, también se pueden probar otras cámaras estereoscópicas o sensores Lidar 3D. La tecnología Lidar emite ondas de luz que rebotan en los objetos y regresan al sensor, y se calcula la distancia a partir del tiempo de retorno de las ondas. Por otro lado en lugar de cambiar el sensor se pueden probar métodos no basados en modelo como redes neuronales recurrentes, donde la entrada sea una secuencia de imágenes y la salida sea la estimación del punto de caída

En el laboratorio de Biorrobótica de la Facultad de Ingeniería de la UNAM se ha logrado evadir obstáculos estáticos con la robot Justina utilizando el algoritmo de campos potenciales, la tesis desarrollada puede ayudar a hacer los ajustes necesarios en los algoritmos de forma que Justina pueda estimar la trayectoria de objetos en movimiento y evadirlos, además en los trabajos previamente enunciados en la sección 2.3 específicamente la estimación de la velocidad en otros autos y la estimación de la velocidad de una pelota (para un robot humanoide que juega futbol) utilizan el filtro de Kalman extendido pero se puede probar con mínimos cuadrados [1, 6, 12].

Bibliografía

[1] J. C. Álvarez. Reconocimiento y seguimiento de personas con robots móviles en ambientes dinámicos empleando técnicas de visión computacional. Tesis de licenciatura, Universidad Nacional Autónoma de México, México, 2015.

[2] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[3] T. Bräunl. *Embedded robotics*. Springer, 2003.

[4] P. I. Corke and O. Khatib. *Robotics, vision and control: fundamental algorithms in MATLAB*, volume 73. Springer, 2011.

[5] D. Forsyth and J. Ponce. *Computer vision: A modern approach*. Prentice hall, 2011.

[6] L. González. Sistema de visión para estimar posición y velocidad de objetos para un robot bípedo. Tesis de licenciatura, Universidad Nacional Autónoma de México, México, 2021.

[7] A. Koubâa et al. *Robot Operating System (ROS).*, volume 1. Springer, 2017.

[8] S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

[9] K. M. Lynch and F. C. Park. *Modern robotics*. Cambridge University Press, 2017.

[10] N. Marco, A. Hector, M. Rubén, R. Karelly, T. David, and V. Héctor. Probabilistic logic markov decision processes for modeling driving behaviors in self-driving cars. En revisión Iberamia 2022.

[11] R. R. Murphy. *Introduction to AI robotics*. MIT press, 2019.

[12] M. Negrete, J. Savage, and L. A. Contreras-Toledo. A motion-planning system for a domestic service robot. *SPIIRAS Proceedings*, 5(60):5–38, 2018.

[13] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2), 2019.

[14] L. G. Shapiro, G. C. Stockman, et al. *Computer vision*, volume 3. Prentice hall New Jersey, 2001.

[15] B. Siciliano and O. Khatib. Robotics and the handbook. In *Springer Handbook of Robotics*, pages 1–6. Springer, 2016.

[16] B. Siciliano, O. Khatib, and T. Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

[17] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to autonomous mobile robots*. MIT press, 2011.

[18] J.-J. E. Slotine, W. Li, et al. *Applied nonlinear control*, volume 199. Prentice hall Englewood Cliffs, NJ, 1991.

[19] J. E. Solem. *Programming Computer Vision with Python: Tools and algorithms for analyzing images*. .°'Reilly Media, Inc.", 2012.

[20] S. Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.