CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Autolocalización visual para robots móviles y aplicaciones

Tesis que presenta

Pablo Vera Bustamante

Para Obtener el Grado de

Doctor en Ciencias

En la especialidad de

Control Automático

Director de la Tesis: Dr. Juan Manuel Ibarra Zannatha

Ciudad de México Noviembre, 2017

Agradecimientos

Este trabajo no habría sido posible sin el apoyo y el estímulo del Doctor Juan Manuel Ibarra Zannatha, bajo cuya supervisión escogí este tema de tesis.

Al CINVESTAV, compañeros, profesores y a todas las personas involucradas en mi formación.

A mi familia Keren Hernández, Christian Vera, Iker Vera, a mis Padres César Vera y Olga Bustamante.

Al Consejo Nacional de Ciencia y Tecnología por el apoyo económico a través de la beca de doctorado que se me otorgó.

Índice General

Bibliografía 150

Índice de Figuras

Índice de Tablas

Índice de Algoritmos

Capítulo 1

Introducción

1.1. Motivación

En la actualidad, la Visión Artificial ha tenido un gran desempeño y es una gran área de investigación, un ejemplo son los automóviles mas modernos, los cuales cuentan con sistemas de visión artificial avanzados, estos sistemas permiten al vehículo conducirse de manera autónoma, donde por medio de la visión obtiene información acerca de su entorno, como identificar los carriles sobre los cuales deben de conducir, las señalizaciones de transito (semáforos, indicadores de velocidad, entre otros) vehículos a su alrededor, peatones, bicicletas, algunos de los vehículos que al día de hoy ya cuentan ello son los automóviles autónomos de Google y Tesla [1].

Por otra parte la visión no solo se ha utilizado en vehículos, si no también en el desarrollo de robots humanoides como los desarrollados por el reto de la Agencia de Proyectos de Investigación Avanzada de la Defensa de Estados Unidos (DARPA Challenge), este concurso esta orientado para el desarrollo de robots de rescate en operaciones de desastre, en donde estos robots autónomos por medio de visión deben de tomar decisiones para realizar diferentes actividades como: identificar un automóvil y conducirlo por un circuito de obstáculos, posteriormente debe de descender del vehículo y dirigirse hacia una zona marcada en la cual deberán de realizar un orificio en una pared debidamente señalada, sin embargo para ello debe de elegir la herramienta adecuada para ello, dentro de un conjunto de herramientas, en otras palabras debe de realizar la identificación de objetos por medio de visión, una

vez localizada esta la debe de tomar por medio de la cinemática propia del robot y realizar la actividad correspondiente, otra de las actividades es la de localizar una puerta, dirigirse hacia ella, abrirla e ingresar por ella, y por ultimo debe de caminar a través de una zona de escombros [2].

Otra de las aplicaciones mas actuales de la Visión Artificial, es la realidad aumentada, la cual consiste en agregar objetos tridimensionales a nuestro entorno, el cual es visualizado por medio de una cámara, para logra este objetivo es de suma importancia conocer la posición de dicha cámara dentro de su entorno para poder agregar los objetos y estos no se muevan de lugar si la cámara cambia de posición.

Estos ha sido algunos de los ejemplos en los que la comunidad científica se encuentra trabajando en donde podemos ver que prácticamente todas las actividades relacionadas con la visión requieren de la posición de la cámara en su entorno lo cual nos ha ido motivando para el desarrollo de este trabajo de tesis.

En el Laboratorio de Robótica y Visión Artificial (RoVisA) del Departamento de Control Automático del Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional, Unidad Zacatenco se desarrollan diversas actividades académicas, científicas y tecnológicas por parte de un grupo de profesores y estudiantes interesados en la Robótica, principalmente en los robots autónomos. Los robots autónomos se caracterizan principalmente por su gran movilidad y su capacitad de tomar decisiones. Así, estos robots cuentan con sistemas computacionales complejos, un alto nivel de percepción visual, así como el control eficiente de sus movimientos y de su comportamiento. Los robots autónmos de interés para el Laboratorio RoVisA son los humanoides, los drones, los automóviles y otros vehículos con ruedas.

En el año 2008, nació el equipo dotMEX, con aplicaciones para los robot humanoides enfocadas en jugar fútbol. El equipo ha participado en diferentes competiciones Nacionales e Internacionales (Torneo Mexicano de Robótica, Robocup, FIRA).

En el torneo realizado por la Asociación Internacional de Robots Futbolistas (FIRA de sus siglas en inglés) en el año de 2012 llevado a cabo en Bristol, Inglaterra, se ganó el primer lugar en la categoría United Soccer de HuroCup al igual que en el International Robotic Contest realizado el mismo año en la República de Corea, donde también se obtuvo el primer lugar en la categoría de Fútbol y el tercer lugar en Maratón.

El equipo dotMEX se encuentra conformado por alumnos de maestría y doctorado del DCA-CINVESTAV, así como por alumnos de nivel licenciatura de diversas escuelas. Este grupo esta dirigido por el Dr. Juan Manuel Ibarra Zannatha profesor investigador del CINVESTAV.

El equipo principalmente utiliza robots comerciales en los cuales se aplican los algoritmos desarrollados en cuanto a Odometría Visual (OV), Localización y Mapeo Simultaneo Visual (VSALM) , caminado, entre otras. Las plataformas utilizadas son el robot francés NAO de Aldebaran Robotics y los coreanos BIOLOID y DARWIN-OP de la empresa ROBOTIS.

Además, en el grupo de trabajo se han desarrollado diversos prototipos de diseño propio como el AH1N1, AH1N2 y Johnny, los cuales cuentan con computadoras embebidas, visión estéreo y algoritmos para el caminado estable y para la compensación de perturbaciones.

Los vehículos con ruedas con que se cuentan en Laboratorio RoVisA son el TurtelBot 2 y el automóvil autónomo desarrollado en la Universidad Libre de Berlín por el Dr. Raúl Rojas.

Finalmente, contamos con los drones Matrice 100 de la empresa china DJI, así como los drones beebop y beboop 2, de la empresa francesa Parrot. En todos estos dispositivos robóticos, tanto en drones, humanoides y vehículos con ruedas es de vital importancia el que posean un sistema de percepción visual que les permita localizarse en su área de trabajo así como detectar, identificar y localizar los diferentes objetos ahí presentes a fin de clasificarlos entre obstáculos y objetivos.

1.2. Objetivos

No solo en el contexto de las competiciones, sino en todo tipo de actividad de los robots autónomos (rescate, fútbol, entre otras) se tiene la problemática de que no se conoce la posición del robot, la cual es sumamente importante para la realización de su tarea. Para conocer la ubicación de estos robots en un entorno desconocido podrían utilizarse métodos como el GPS, sistemas de odometría inerciales y aún visuales o combinación de ambos [3].

La presente tesis se concentrará en resolver este reto esto es, el objetivo de esta tesis es obtener la ubicación de un robot autónomo (humanoide, automóvil o dron) dentro de su entorno, con respecto al referencial del mundo o con respecto al robot mismo a través de la utilización de algoritmos de Visión Artificial dependiendo de las necesidades de problemas para cada actividad a desempeñar, los cuales también pueden aplicarse a drones autónomos. Ademas para algunas actividades de los robots humanoides, enfocadas en la manipulación de objetos es necesario conocer las cinematica de estos. Dentro de este objetivo general podemos definir las siguientes objetivos específicos:

- Dado que se cuenta con robots humanoides con visión estéreo realizar e implementar algoritmos de odometría visual (OV) para determinar la trayectoria efectuada por el robot móvil en un ambiente desconocido.
- Realizar la relocalización de un robot móvil en un mapa conocido.
- Como se cuentan con diferentes drones, los cuales cuentan con una cámara monocular se deberá realizar la localización de drones.
- Realizar la cinemática directa para el robot humanoide AH1N2.
- Realizar aplicaciones para competiciones con robots humanoides, considerando que las capacidades computacionales de estos dispositivos son muy limitadas.

1.3. Artículos Publicados y Competiciones

A continuación se presentan los artículos y conferencias realizados durante este periodo doctoral, en donde se cuenta con una revista indexada, dos artículos internacionales y 5 artículos nacionales.

- P. Vera, J. Ibarra, R. Carrillo, D. Canizo, *¿Qué necesita un robot humanoide para jugar Fútbol?*, AMRob Journal, Robotics: Theory and Applications, 2015.
- P. Vera, Robots Futbolistas, Congreso del Instituto Superior de Tepeaca, 2015.
- R. Carrillo, P. Vera , B. Medrano, E. Hernández, J. Ibarra, 3D Self-localization for Humanoid Robots Using View Regression and Odometry, International Conference on Electrical Engineering, Computing Science and Automatic Control 2015.
- P. Vera, J. Ibarra, Estimación de pose de una cámara estéreo, Congreso Mexicano de Robótica, 2017.
- A. Malo, P. Vera, J. Maldonado, A. Cobos, Inverse models and robust parametric step neuro-control of a Humanoid Robot, Neurocomputing, Volumen 233, Paginas 90-103, ISSN 0925-23122017, 2017.
- P. Vera, J. Ibarra, Visual Odometry based on trilateration method with stereo images, International Conference on Electronics, Communications and Computers, 2017.
- P.Vera, J. Ibarra, J. Javier, E. Ortigoza, Implementación de las habilidades de un portero en un humanoide, AMRob Journal, Robotics: Theory and Applications, 2017.
- P. Vera, B. Medrano, J. Ibarra, M. Fuentes, A. Arrearan, Implementación de habilidades culinarias en un humanoide NAO, AMRob Journal, Robotics: Theory and Applications, 2017.

En la tabla 1.1, se presentan un listado de la competiciones en las que se ha participado como miembro del equipo dotMEX. Así como los resultados obtenidos en cada una de ellas.

Competición	**Lugar**	**Resultado**
2014 Torneo Mexicano de Robótica Categoría: Robots Futbolistas	Cd. del Carmen, Campeche	Tercero
2014 IEEE-RAS International Conference on Humanoid Robots 2015 Torneo Mexicano de Robótica Categoría: Robots Cocineros	Madrid, España	Segundo
2015 Torneo Mexicano de Robótica Categoría: Robots Futbolistas	Cd. de México	-
2015 Torneo Mexicano de Robótica Categoría: Drones Autónomos	Edo. de México	Primero

Cuadro 1.1: Competiciones.

1.4. Organización de la Tesis

En el Capítulo 1 se explican las motivaciones, el objetivo principal de este trabajo, así como los artículos publicados.

En el Capítulo 2 se presentan las bases de Visión Artificial, las cuales son necesarias para la resolución de este trabajo. Donde se muestran los algoritmos de identificación de características y las etapas de filtrado para tener un buen emparejamiento entre puntos característicos, así como los resultados de las implementaciones.

En el Capítulo 3 se explica a fondo el problema de odometría visual (OV), el estado del arte y algunas de las técnicas más utilizadas para la resolución de este problema.

En el Capítulo 4, se explican las diferentes técnicas utilizadas para la autolocalización y relocalización de robots móviles, en la Sección 4.1 se explica el algoritmo propuesto para realizar estimación de la pose de una cámara estereoscópica, por medio de un algoritmo iterativo de trilateración. También se realiza una comparación en simulación entre los algoritmos más usados y el propuesta para el caso de cámaras estéreo, así mismo, se presentan los resultados reales implementados con una cámara

Minoru 3D y las estimaciones para una trayectoria realizada por un vehículo. En la Sección 4.2, se presenta una comparación entre dos técnicas de relocalización la primera se basa en la relocalización a través de un mapa de imágenes sintéticas, que desarrolló Ricardo Carrillo alumno de Maestría, la segunda técnica presentada es la propuesta en este trabajo de tesis, en el cual se recurre a la trilateración para estimar la pose actual del robot. Mientras que en la Sección 4.3, se muestra la implementación de un sistema de OV 2D-2D, para la estimación en la trayectoria de un quadrirotor, así como sus resultados reales y de una base de datos.

En el Capítulo 5, se presenta el modelado del robot humanoide AH1N2, el cual se desarrolló en el Departamento de Control Automático, del CINVESTAV, en conjunto con el Dr. Alejandro Malo, tanto para su cinemática directa, como para la inversa.

Mientras que en los Capítulos 6 y 7, se muestran aplicaciones con los robot móviles humanoides, en el primero la aplicación de jugar fútbol, con un portero y un delantero, para las competencias de fútbol como *TMR* y *Robocup*, mientras que en el Capítulo 7 se presenta la metodología y los algoritmos que se utilizaron para el concurso *Humanoids 2014*, el cual fue básicamente un concurso en un ambiente de cocina.

Finalmente en el Capítulo 8, se presentan las conclusiones de los resultados que se obtuvieron, así como el trabajo a futuro a realizar.

Capítulo 2

Visión Artificial

En este Capítulo se presentan de manera breve los aspectos de Visión Artificial que son de utilidad en el desarrollo de la investigación que sustenta esta tesis. En este Capítulo se presentan de manera mas detallada los algoritmos implementados para resolver el problema de la autolocalización, ademas, se presentan temas básicos como el modelo de cámara y su calibración; detección, descripción y seguimiento de características (keypoints); triangulación y emparejamiento de puntos característicos.

2.1. Modelo de Cámara

Se entiende por modelo de cámara aquel que explica cómo un punto 3D en el espacio es proyectado al plano imagen 2D de una cámara. Los modelos de cámara, son simplificaciones de la realidad y el más utilizado para representar la formación de imágenes es el modelo de cámara *PinHole* (cámara Estenopica).

2.1.1. Modelo PinHole

El modelo de agujero o *PinHole* considera la óptica del sistema formado por una única lente representada por un punto infinitesimal, denominado foco, a través del cual pasan los rayos de la luz procedentes de la escena hacia el sensor de la cámara como se muestra en la figura 2.1.

Básicamente, este modelo aplica una matriz de proyección para transformar las

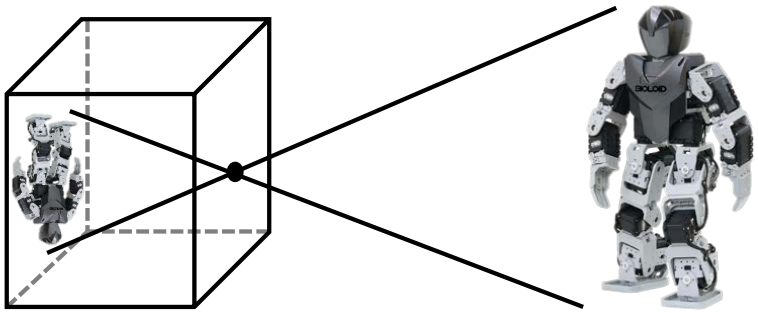


Figura 2.1: Modelo PinHole.

coordenadas 3D de los puntos del objeto en coordenadas 2D de la imagen.

$$\mathbf{U} \cong \mathbf{PX} \tag{2.1}$$

donde $\mathbf{X} = [X_w, Y_w, Z_w]^t$ es el vector del punto 3D de la escena en coordenadas homogéneas, $\mathbf{P}$ es una matriz de 3×4 denominada matriz de proyección y $\mathbf{U} = [u, v, 1]^t$ es el vector de las coordenadas del punto proyectado en el plano imagen y transformación homogénea a píxeles. El símbolo $\cong$ indica que ambos elementos de la ecuación son equivalentes salvo un factor de escala λ, como se presenta en la ecuación (2.2).

$$\lambda\mathbf{U} = \mathbf{PX} \tag{2.2}$$

La matriz de proyección $\mathbf{P}$ se caracteriza en dos partes, los parámetros intrínsecos definidos en la matriz $\mathbf{K}$ y los parámetros extrínsecos representados por $[\mathbf{R}|\vec{T}]$, como se muestra en la ecuación (2.3).

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\vec{T}] \tag{2.3}$$

2.1.2. Parámetros Intrínsecos

Son aquellos que describen el funcionamiento de la cámara, de acuerdo a su geometría y óptica. Los parámetros intrínsecos básicos son:

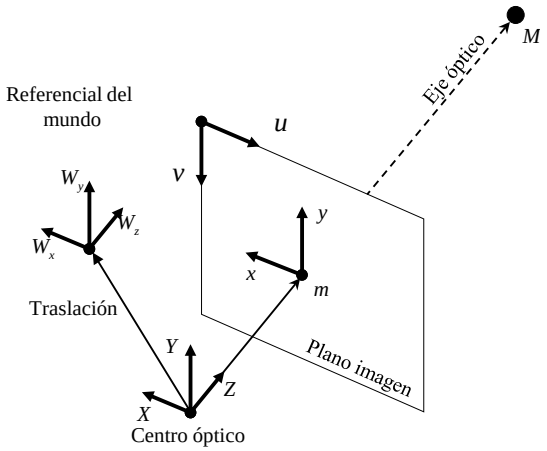


Figura 2.2: Parámetros de la Cámara.

- *Centro Óptico (u_0, v_0):* Es el punto donde el eje óptico de la cámara atraviesa al plano imagen, también es llamado punto principal, y sus coordenadas están en píxeles con respecto al sistema asociado al plano imagen. Como se muestra en la figura 2.2

- *Distancia Focal (f):* es la distancia entre el foco y el plano imagen, en milímetros.

- *Factor de proporción (s_x):* Indica la relación entre la dimensión horizontal y vertical de un píxel.

- *Factor de conversión píxel-milímetros (d_x, d_y):* Es la cantidad de pixeles por milímetro que usa la cámara . Esta relación se obtiene dividiendo la dimensión en píxeles de la imagen por el tamaño en mm del CCD (Charge-Couple Device[1])

- *Factor de Escala (k_x, k_y):* Es la proporción de tamaño de un objeto visto en la realidad con respecto a su proyección en el plano imagen. La proporción puede ser diferente en cada eje. Cundo no existe distorsión el factor de escala está dado por: $k_x = s_x d_x f$ y $k_y = d_y f$.

- *Ortogonalidad de los ejes del plano imagen (Ω_1, Ω_2):* es el ángulo que se forma

[1]CCD: Charge-Couple Device, son las siglas en inglés para Dispositivo de carga interconectadas, y es un sensor capaz de convertir luz en señales eléctricas

ente los dos ejes de la imagen; es decir mide la ortogonalidad de los ejes del plano imagen. El caso ideal este ángulo debe ser de 90°, en situaciones reales este ángulo puede variar cuando la lente de la cámara no es paralela al plano imagen [4].

- *Distorsión:* El efecto de la distorsión es modelado por medio de dos componentes: una radial $(k_1, k_2, ...)$ y otra tangencial $(p_1, p_2, ...)$

En la ecuación (2.4) se presenta la forma de la matriz de parámetros intrínsecos, en donde no se considera la distorsión.

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

2.1.3. Parámetros Extrínsecos

Los parámetros extrínsecos de la cámara son los parámetros externos de la misma; es decir, el vector de traslación $\vec{T}$ ecuación (2.5) de la cámara y su orientación representado por una matriz de rotación **R**.

$$\vec{T} = [T_x, T_y, T_z] \tag{2.5}$$

La matriz de rotación se encuentra asociada a tres rotaciones alrededor de los ejes, la rotación alrededor del eje **X** está especificada con un ángulo α, la rotación alrededor del eje **Y** especificada por el ángulo β y la rotación alrededor del eje **Z** por θ , tal y como se muestra en la figura 2.3.

Las matrices de rotación alrededor de cada uno de los ejes del sistema de coordenadas **X**, **Y** y **Z** se muestran en la ecuación (2.6).

$$\mathbf{R} = R_\alpha R_\beta R_\theta \tag{2.6}$$

El vector de Traslación y la matriz de rotación forman una matriz aumentada de 3×4 como se muestra en la ecuación (2.7) la cual es la matriz de parámetros extrínsecos.

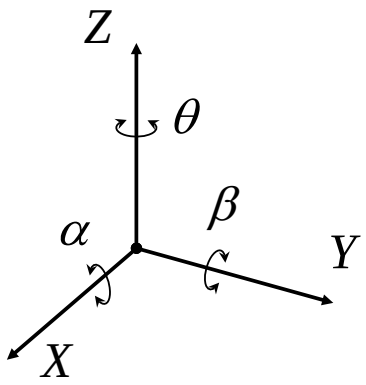


Figura 2.3: Rotación (α, β, θ) alrededor de los ejes **X**, **Y**, **Z**

$$[\mathbf{R}\vec{T}] = \left[\mathbf{R}\left|\vec{T}\right.\right] = \begin{bmatrix} \mathbf{R_{11}} & \mathbf{R_{12}} & \mathbf{R_{13}} & T_x \\ \mathbf{R_{21}} & \mathbf{R_{22}} & \mathbf{R_{23}} & T_y \\ \mathbf{R_{31}} & \mathbf{R_{32}} & \mathbf{R_{33}} & T_z \end{bmatrix} \tag{2.7}$$

2.2. Calibración de la Cámara

La calibración de una cámara consiste en utilizar diferentes técnicas para calcular o identificar los parámetros intrínsecos y extrínsecos de la cámara, a partir de diferentes puntos en la escena, en la actualidad existe una gran variedad de técnicas y métodos para realizar la calibración, sin embargo nos enfocaremos solamente al método de Zhang [5]. Debido a que los demás métodos necesitan información a priori acerca de la cámara o la posición de los puntos de referencia en el espacio.

2.2.1. Método de Zhang

El método propuesto por Zhang [5] resuelve el modelo de la cámara, a partir de la observación de una plantilla plana, la plantilla está cubierta de cuadrados blancos y negros en forma de tablero de ajedrez, cuyas esquinas se utilizan como puntos de referencia. En este método no es necesario conocer la posición de la plantilla pues puede ser vista desde diferente posiciones y orientaciones, lo que utiliza es la distancia relativa entre las esquinas.

El modelo de cámara utilizado se presenta en la ecuación (2.8), el cual corresponde

con el de *PinHole*, donde s es un factor de escala arbitrario, $\widetilde{P}$ es el punto en el plano imagen en coordenadas homogéneas, $\mathbf{K}$ es la matriz de parámetros intrínsecos de la forma (2.4), $[\mathbf{R}|\mathbf{T}]$ son los parámetros extrínsecos, que relacionan el referencial del mundo con el referencial de la cámara, y $\widetilde{X}$ es un punto 3D en el espacio, en coordenadas homogéneas.

$$s\widetilde{P} = \mathbf{K}[\mathbf{R}|\mathbf{T}]\widetilde{X} \tag{2.8}$$

Este método determina los parámetros de la cámara a partir de dos o más imágenes de la plantilla en diferentes posiciones. El método es iterativo y determina los parámetros intrínsecos (foco, centro óptico, distorsión, etc...) así como los parámetros extrínsecos de la cámara (traslación y rotación). En la figura 2.4 se muestra un ejemplo de una captura de la plantilla.

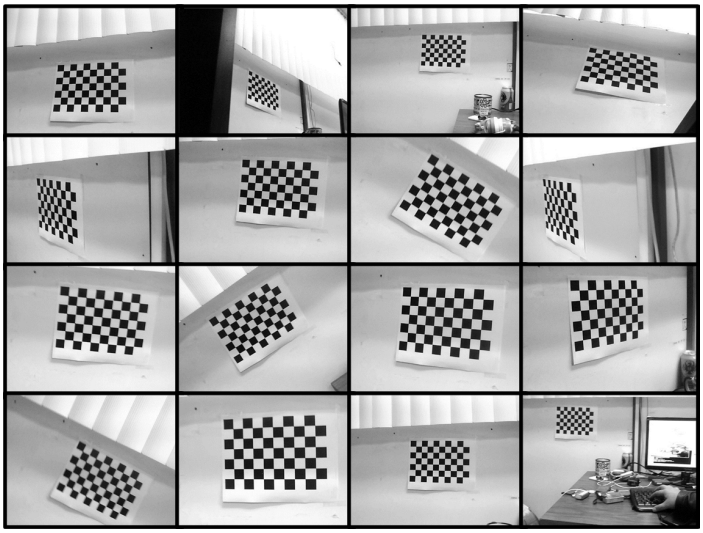


Figura 2.4: Plantilla para calibración de cámara.

2.2.2. Calibración Estéreo

En el caso ideal los dos sensores de uns cámara estéreo son paralelos, sin embargo esto es prácticamente imposible de lograr, para ello se realiza una calibración estéreo en la cual se obtienen 2 tipos de información: la calibración de cada una de las cámaras (los parámetros intrínsecos), y la transformación rígida (rotación y traslación)

que relaciona la posición y orientación entre los referenciales de las cámaras. Para realizar la calibración estéreo se toma en cuenta la restricción de geometría epipolar $p'Ep = 0$, para generar la matriz de rotación y traslación entre las dos cámaras del sistema estéreo.

2.3. Detección y Descripción de Características

Para la selección de la técnica para identificar características, se debe considerar la aplicación. A continuación se da una explicación de algunos de las técnicas existentes y que se han utilizado en este trabajo. Dentro de una imagen es importante detectar puntos, que sean distinguibles de los demás, estos son llamados *features*, *Keypoints* o *corners*. En la literatura se pueden encontrar una gran cantidad de algoritmos que resuelven este problema, dentro de las características que deben de tener se encuentran:

- Precisión en la localización de puntos característicos.
- Eficiencia computacional.
- Robustez

A continuación se da una breve descripción de algunos de estos algoritmos, que se han probado y utilizado en diferentes etapas de este estudio.

2.3.1. Detector de Esquina de Harris

Este es el detector de esquinas más utilizado. Fue desarrollado por Harris y Stephens en 1988 [6], y consiste una modificación del detector de Moravec [7]. En el detector de esquinas de Harris se evalúa la respuesta a un parche centrado en el píxel a analizar, dependiendo de esta respuesta se pude identificar una región plana (sin cambio), un borde (cambios en una dirección), y una esquina (cambios en dos o más direcciones), la función propuesta por Harris se presenta en la ecuación (2.9), en donde se evalúa la intensidad de los pixeles alrededor del punto de interés.

$$E(x, y) = \sum_{u,v} w(u, v)[I(u + x, v + y) - I(u, v)]^2 \tag{2.9}$$

donde $w(u,v) = \exp(-(\mathrm{u}^2 + \mathrm{v}^2)/2\rho^2)$. Analizando lo elementos de primer orden de la expansión por series de Taylor, se puede obtener una aproximación de acuerdo a la ecuación (2.10), donde I_u e I_v son los gradientes horizontales y verticales respectivamente.

$$E(x,y) = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} \sum I_u^2 & \sum I_u I_v \\ \sum I_u I_v & \sum I_v^2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = DMD^T \tag{2.10}$$

Analizando la correlación entre los valores propios (λ_1, λ_2) de la matriz M el punto (x,y) se puede clasificar como:

Región plana: si λ_1 y λ_2 son valores pequeños, no existe un cambio en ninguna dirección, por lo cual nos encontramos en una región plana.

Borde: si solo uno de los valores propios λ_1 o λ_2 es grande, entonces existe un cambio en la intensidad en una dirección, por lo cual se define un borde.

Esquina: es cuando ambos valores propios λ_1 y λ_2 son grandes, por lo cual existen cambios de intensidad en diferentes direcciones.

La función de respuesta o de correlación se muestra en la ecuación (2.11).

$$R = \det(\mathrm{M}) - \mathrm{ktraza(M)} = \lambda_1 * \lambda_2 - \mathrm{k}(\lambda_1 + \lambda_2) \tag{2.11}$$

El descriptor se considera como los valores de intensidad en la ventana utilizada.

2.3.2. FAST

Es un algoritmo propuesto por Rosten and Drummond en 2006 [8], FAST son la siglas en inglés de *Features from Accelerated Segment Test*, dicho algoritmo analiza los píxeles en una circunferencia de radio r alrededor del punto de interés $p(x,y)$, típicamente utiliza un radio $r = 3$ píxeles, es decir la circunferencia cuenta con 16 pixeles a analizar. Se considera que existe una característica si en la circunferencia existen n píxeles continuos cuyo brillo es mayor $I_p + t$ o menor $I_p - t$, a la intensidad I_p de $p(x,y)$ donde t es un umbral de rechazo lo cual se muestra en la imagen 2.5. FAST se divide en dos etapas principales:

Prueba Rápida: se encarga de eliminar los puntos que no sean esquinas, en esta prueba se analizan únicamente 4 píxeles de la circunferencia en las posiciones 0° (1), 90° (5), 180° (9) y 270° (13); si al menos 3 de estos píxeles son más brillantes o más oscuros que el punto p, dicho punto se considera como una esquinas y pasará a la etapa de prueba completa.

Prueba completa: en esta etapa se analiza la continuidad de los píxeles del círculo, si existen más de $n = 9$ píxeles continuos se considera al punto p como esquina.

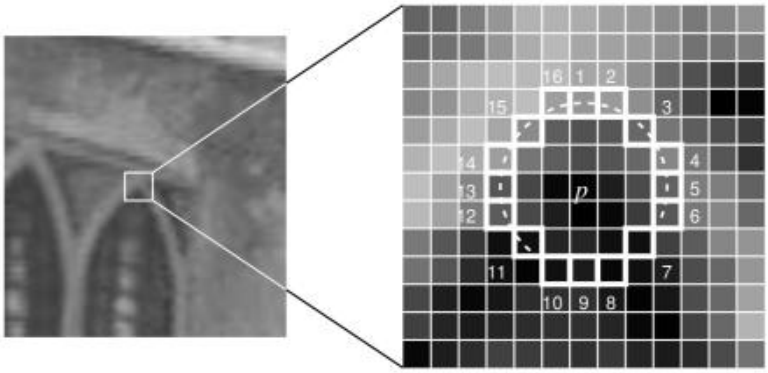


Figura 2.5: FAST.

2.3.3. SIFT

La técnica SIFT (*Scale Invariant Feature Transform*), es un algoritmo robusto en cuanto a las escala, iluminación y rotación, fue creado por David G. Lowe [9], el cual se divide en las siguientes etapas: Detección de extremos en el espacio de escalas, localización de puntos característicos, asignación de orientación y cálculo de descriptores.

El descriptor del punto es un vector de 16 elementos que contiene la información de la intensidad de los píxeles del círculo.

Detección de extremos en el espacio de escalas: realiza la búsqueda en todas las escalas y píxeles de la imagen, puntos característicos que son invariantes a escala y orientación, para ello se utiliza una Diferencia de Gaussianas (DoG)

de la imagen donde se encuentran los extremos máximos y mínimos de cada región para cada escala. Generando el Espacio Escala, el cual es la aplicación de filtros Gaussianos sucesivos y las reducciones de la imagen original, sobre este espacio se realiza una búsqueda de máximos y mínimos sobre cada píxel en una vecindad de 26 píxeles en bloques de $3 \times 3 \times 3$.

Localización de los puntos clave: cada punto candidato es analizado para descartar los puntos característicos que no sean importantes. Para ello, se compara la intensidad de los extremos contra un umbral y son eliminados aquellos con una baja iluminación. Posteriormente se aplica un filtro de esquinas para eliminar aquellos máximos y mínimos que no sean esquinas.

Asignación de orientación: sobre cada punto característico encontrado, se obtienen los gradientes en una región alrededor de éste. A cada píxel en la región se evalúan los cambios de dirección en x y y. Con ellos se realiza un histograma de orientaciones cuantizadas representados en 35 bloques, los picos del histograma representan la orientación predominante y si se encuentra otro pico por encima del 80 % del mayor pico es considerado otro punto característico con dicha orientación.

Cálculo de descriptores: los descriptores son una firma de cada punto característico, esta firma es invariante en escala, orientación e iluminación. Para determinar el descriptor se calculan las orientaciones de 16 regiones de 4x4 píxeles y se colocan en un histograma de orientaciones. Para asegurar la invariancia en rotación, las orientaciones de cada gradiente se toman con respecto a la orientación principal de cada región, con ello se forma 16 histogramas de 8 clases, y se forma un vector que concatena los 128 datos, y a su vez es normalizado para asegurar la invariancia en iluminación.

2.3.4. ORB

Es un detector robusto de características locales rápidamente, fue presentado por primera vez por Ethan Rublee et al [10]. Se basa en el detector de puntos característicos del algoritmo FAST y el descriptor BRIEF (*Binary Robust Independent*

Elementary Features). Su objetivo es proporcionar una alternativa rápida y eficiente a SIFT o SURF.

2.4. Seguimiento de Características

2.4.1. Descriptores

Como se observó en la Sección anterior, cada una de las técnicas para localizar puntos característicos, cuenta también con un descriptor de dicho punto, este descriptor es el que se utiliza para realiza el emparejamiento de puntos llamado en inglés *matching*.

Para realizar la comparación entre descriptores se pueden utilizar la distancia entre ambos descriptores y seleccionar el más semejante, esta distancia se puede calcular por la distancia Euclidiana, ecuación (2.12), la cual es la más utilizada, la distancia de Mahalanobis, la cual determina la similitud ente dos variables multidimensionales [11] mostrado en la ecuación (2.13), o la distancia de Hamming en el caso de descriptores binarios.

$$d(\vec{p}_1, \vec{p}_2) = \sqrt{\sum_{i=1}^{n} (p_{1i} - p_{2i})^2} \tag{2.12}$$

$$d(\vec{p}_1, \vec{p}_2) = \sqrt{(\vec{p}_1 - \vec{p}_2)^T \Sigma^{-1} (\vec{p}_1 - \vec{p}_2)} \tag{2.13}$$

Para realizar el emparejamiento de puntos se consideran dos conjuntos de descriptores $\vec{P}_k$ y $\vec{P}_l$, los cuales corresponde a los descriptores en las cámaras izquierda y derecha o los descriptores en una cámara en dos instantes de tiempo. Estos conjuntos son de dimensión m y n respectivamente y cada descriptor individual se denotara como $\vec{p}_{ki}$ y $\vec{p}_{li}$. Con estas consideraciones se genera una matriz de distancias $\mathbf{D}$ de acuerdo a la ecuación (2.14). Generada la matriz de distancias se realiza la búsqueda del elemento mínimo de la matriz $\mathbf{D}$ cuyas coordenadas del elemento mínimo corresponden a las posiciones en los respectivos vectores de características ($\vec{P}_k$ y $\vec{P}_l$) para cada conjuntos $(i, j) = \text{mín}(\mathbf{D})$, almacenando estos elemento como pareja, posteriormente el renglón i y la columna j de la matriz eliminados de la matriz, este proceso

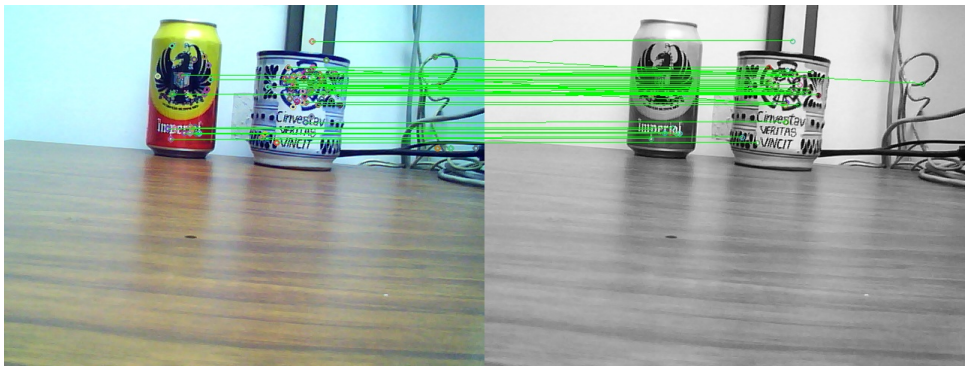


Figura 2.6: Correspondencia entre descriptores.

se repite para conformar todas las parejas de puntos hasta que el valor mínimo de $\mathbf{D}_{i,j} > Umbral$.

$$\mathbf{D} = \begin{bmatrix} d(\vec{p}_{k1}, \vec{p}_{l1}) & d(\vec{p}_{k1}, \vec{p}_{l2}) & \cdots & d(\vec{p}_{k1}, \vec{p}_{ln}) \\ d(\vec{p}_{k2}, \vec{p}_{l1}) & d(\vec{p}_{k2}, \vec{p}_{l2}) & \cdots & d(\vec{p}_{k2}, \vec{p}_{ln}) \\ \vdots & \vdots & \ddots & \\ d(\vec{p}_{km}, \vec{p}_{l1}) & d(\vec{p}_{km}, \vec{p}_{l2}) & & d(\vec{p}_{km}, \vec{p}_{ln}) \end{bmatrix} \tag{2.14}$$

2.4.2. Flujo Óptico

El flujo óptico es una técnica de estimación del movimiento de los pixeles dentro de un par de imágenes consecutivas, existen diferentes metodologías para resolver este problema las cuales se puede clasificar como en [12]:

Métodos diferenciales: determinan el desplazamiento de los píxeles a partir de las derivadas espaciales o espacio temporales de las intensidades de la imagen. Las cuales a su vez se subdividen en:

- *Métodos Locales:* utilizan la información en una vecindad alrededor de un píxel para estimar su movimiento. El método más representativo de esta familia es el de Lucas-Kanade [13].

$$f(x + dx, y + dy, t + dt) = f(x, y, t) \tag{2.15}$$

Figura 2.7: Flujo óptico.

- *Métodos Globales:* añaden una restricción global, la cual es un término de regularización sobre el flujo. Esta restricción supone que el campo de desplazamiento es suave, como por ejemplo el método de Horn-Schunck [14].

- *Métodos de contorno:* utilizan la información de los bordes de los objetos para detectar el desplazamiento. Aplican técnicas diferenciales para la extracción de determinadas estructuras en la imagen para luego establecer correspondencias entre estas estructuras.

Métodos de energía: la idea que subyace en este tipo de métodos es la definición de una energía que penaliza las desviaciones respecto a las restricciones impuestas en el modelo.

En el problema de flujo óptico se consideran varias hipótesis para lograr resolverlo como intensidad constante, rigidez de los objetos, coherencia espacial, entre otras.

Algoritmo de Lucas Kanade

La técnica de Lucas Kanade resuelve la ecuación básica del flujo óptico dada por la ecuación (2.15) para todos los píxeles en ese vecindario y lo hace bajo el criterio de mínimos cuadrados. Esta técnica sólo se puede utilizar cuando el vector de flujo de imagen (V_x,V_y) es pequeño entre dos instantes cercanos de la imagen para que la ecuación de flujo óptico pueda mantenerse [13].

$$\begin{aligned} I_x(p_1)V_x + I_y(p_1)V_y &= -I_t(p_1) \\ I_x(p_2)V_x + I_y(p_2)V_y &= -I_t(p_2) \\ &\vdots \\ I_x(p_n)V_x + I_y(p_n)V_y &= -I_t(p_n) \end{aligned} \tag{2.16}$$

donde I_x, I_y , I_t son las derivadas parciales de la imagen con respecto a la posición y tiempo, p_1, p_2, p_n representa los píxeles que se encuentran en la ventana, la ecuación (2.16) se reescribe de forma matricial como $Av = b$ donde:

$$A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, \qquad v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, \qquad b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix}$$

Dado que se tienen más ecuaciones que incógnitas, el sistemas esta sobre determinado, para la cual la solución es $v = (A^T A)^{-1} A^T b$ quedando como se muestra en la ecuación (2.17).

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_i)^2 & \sum_i I_x(p_i)I_y(p_i) \\ \sum_i I_x(p_i)I_y(p_i) & \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_i)I_t(p_i) \\ -\sum_i I_y(p_i)I_t(p_i) \end{bmatrix} \tag{2.17}$$

Esta solución le da los mismos pesos a cada píxel dentro de la ventana, sin embargo en la práctica es mejor asignarles pesos a cada uno de los píxeles, con respecto a la distancia del centro del a ventana, para ello se utiliza una versión de pesos ponderados agregando la matriz diagonal $W_{n\times n}$, donde los pesos w_i son ajustados a una función gaussiana, dando como solución $v = (A^T W A)^{-1} A^T W b$ quedando como se

muestra en la ecuación (2.18).

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} w_i \sum_i I_x(p_i)^2 & w_i \sum_i I_x(p_i) I_y(p_i) \\ w_i \sum_i I_x(p_i) I_y(p_i) & w_i \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -w_i \sum_i I_x(p_i) I_t(p_i) \\ -w_i \sum_i I_y(p_i) I_t(p_i) \end{bmatrix} \tag{2.18}$$

2.5. Triangulación

La triangulación es una familia de diferentes métodos con los cuales se puede estimar la profundidad o posición tridimensional de una característica, por medio de sus proyecciones en un par de imágenes estéreo.

En este caso, para realizar la estimación de los puntos 3D de las características se realiza por geometría. En la figura 2.8 se muestra la configuración de un par estéreo bajo el cual se puede llegar a las relaciones mostradas en la ecuación (2.19) para el caso de la cámara izquierda y la ecuación (2.20) para la cámara derecha, donde u_L, v_L son las coordenadas en el plano imagen de la proyección izquierda, c_{u_L} y c_{v_L} es el centro óptico de la cámara izquierda, f_L es el foco de la cámara izquierda en milímetros y u_R, v_R, c_{u_R}, c_{v_R}, f_R son análogas para la cámara derecha y B es la separación entre las cámaras.

$$\frac{f_L}{z} = \frac{(u_L - c_{u_L})}{x} = \frac{(v_L - c_{v_L})}{y} \tag{2.19}$$

$$\frac{f_R}{z} = \frac{(u_R - c_{u_R})}{x - B} = \frac{(v_R - c_{v_R})}{y} \tag{2.20}$$

De la ecuación (2.19) y la ecuación (2.20) se pueden deducir las coordenadas de la posición del punto 3D de acuerdo con las ecuaciones (2.21), (2.22) y (2.23), con respecto al referencial de la cámara izquierda.

$$z = \frac{B f_L f_R}{(u_L - c_{u_L}) f_R - (u_R - c_{u_R}) f_L} \tag{2.21}$$

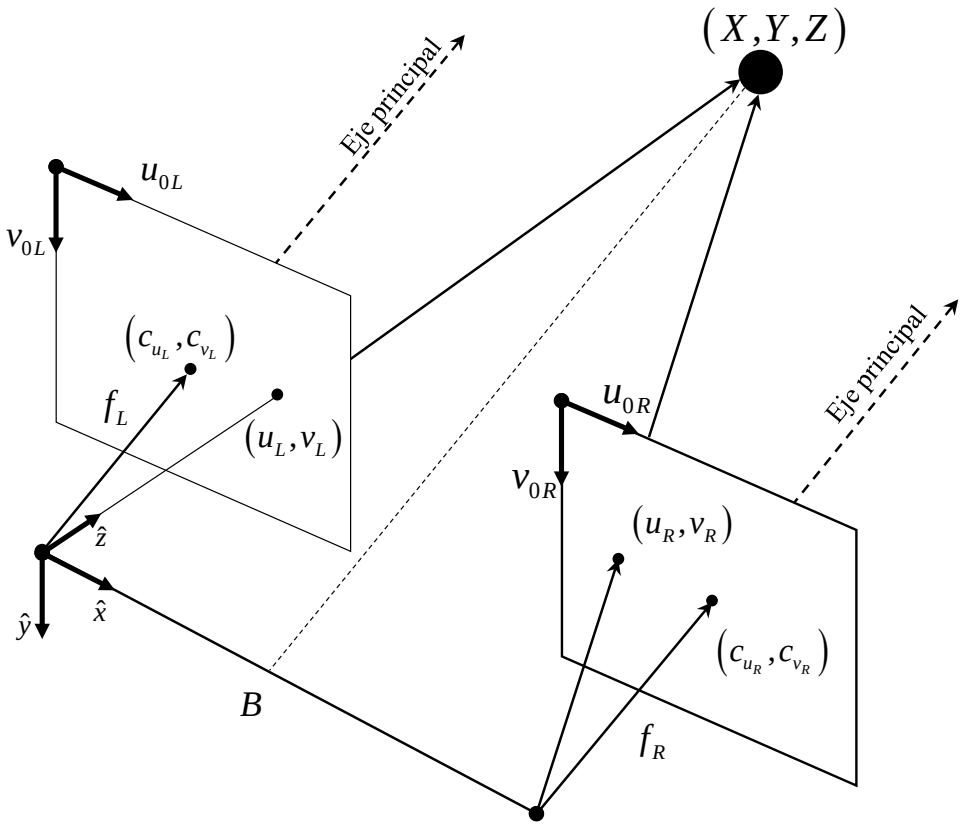


Figura 2.8: Arreglo de cámaras Estéreo.

$$x = \frac{z\,(u_L - c_{u_L})}{f_L} \tag{2.22}$$

$$y = \frac{z\,(v_L - c_{v_L})}{f_L} \tag{2.23}$$

2.6. Implementación del Algoritmo de Emparejamiento

En esta Sección se explicará el algoritmo de emparejamiento implementado para realizar el seguimiento de características. Así como cada uno de los pasos realizados para tener un buen emparejamiento, además de que en el caso de trilateración se necesitan de dos tipos de emparejamiento uno en estéreo es decir entre la imagen

izquierda y la derecha, y otro emparejamiento entre imágenes subsecuentes de la misma cámara.

2.6.1. Detección de Puntos Característicos

La detección de puntos característicos se realizó por medio de los algoritmos principalmente SIFT y ORB, los cuales se encuentran implementados en la librería de Opencv [15] [16], y han sido descritos en las secciones anteriores (2.3.3 y 2.3.4).

En la figura 2.9 se presentan los puntos característicos obtenidos con el algoritmo SIFT cuyos descriptores son de 128 bytes, mientras que en la figura 2.10 se muestran los obtenidos por medio del algoritmo de ORB cuyos descriptores son de 32 bytes. Ambos algoritmos fueron configurados para determinar alrededor de 500 puntos característicos.

Figura 2.9: Características detectadas con SIFT.

2.6.2. Eliminación de Puntos Característicos

Para realizar un emparejamiento correcto entre puntos de diferentes imágenes, es necesario realizar previamente un filtrado de puntos; es decir quedarnos con aquellos que sirven. En el caso del algoritmo SIFT este nos proporciona un evaluación del

Figura 2.10: Características detectadas con ORB. En la parte superior para la cámara izquierda y la inferior para la cámara derecha.

punto para saber cómo una primera instancia que tan sencillo es ubicarlo nuevamente, en el caso de la imagen presentada en la figura 2.11, el valor de la respuesta de los puntos oscila entre un 0.02 y 5, con lo cual seleccionamos el valor de 0.06 para eliminar aquellos difíciles de encontrar, los resultados de este filtrado se presentan en la figura 2.11.

Una desventaja que tiene el algoritmo de SIFT (implementado en OpenCV), es que puede considerar el mismo punto en varias ocasiones, esto se debe principalmente a que se pueden encontrar en diferentes escalas, sin embargo esto complica el emparejamiento, por lo cual se optó por quedarnos únicamente con un punto por posición, es decir eliminando los repetidos y dejando el que se encuentra en la primera octava del conjunto de escalas. También se eliminan los puntos que estén muy cercanos en una vecindad de 2 píxeles, dejando solamente el que tenga la respuesta más grande, como se muestra en la figura 2.12.

Este proceso de eliminación de puntos característicos se realiza para cada imagen obtenida, para posteriormente continuar con el proceso de emparejamiento.

Figura 2.11: Puntos característicos después de eliminar los de respuesta pequeña.

Figura 2.12: Puntos característicos después de eliminar los repetidos y cercanos.

2.6.3. Emparejamiento en Visión Estéreo

Como se explicó en la Sección 2.4.1 se genera una matriz de distancias de la forma (2.14), sin embargo se agrega una restricción del par estéreo, es decir que los puntos

característicos a emparejar deben de estar sobre la misma linea epipolar, es decir su coordenada v debe de ser la misma, donde cada elemento de la matriz está dado por (2.24), donde k_{iv}, es la coordenada v del punto asociado al descriptor $\vec{p}_i$.

$$d(\vec{p}_1, \vec{p}_2) = \begin{cases} \sqrt{\sum_{i=1}^{n} (p_{1i} - p_{2i})^2} & si\, k_{1v} = k_{2v} \\ 3000 & si\, k_{1v} \neq k_{2v} \end{cases} \tag{2.24}$$

Posteriormente se procede a encontrar el mínimo valor en la matriz D, cuyas coordenadas (i, j) corresponder con la pareja de puntos característicos, como se muestra en la figura 2.13 para el caso de SIFT y en 2.14 para el caso de ORB.

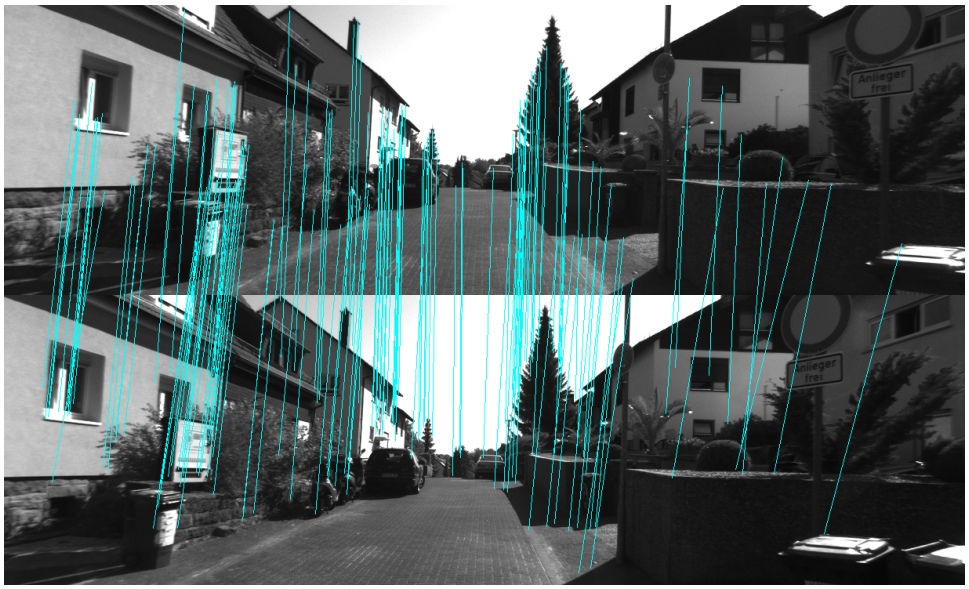


Figura 2.13: Emparejamiento estéreo SIFT.

Una vez realizado el emparejamiento se prosigue a eliminar aquellos puntos cuya disparidad sea muy pequeña, menor a 1 pixel, es decir aquellos que se encuentren muy lejos como se muestra en la ecuación (2.25), debido a que la estimación en profundidad tiende a ser mala entre más lejos se encuentre. Esta restricción también elimina aquellos puntos que se han emparejado y la posición en v de la imagen derecha es menor que la de su pareja en la imagen izquierda, dado que son outliers, debido a la colocación de las cámaras no puede suceder esto. Los resultados se muestran en la figura 2.15 para el caso de SIFT.

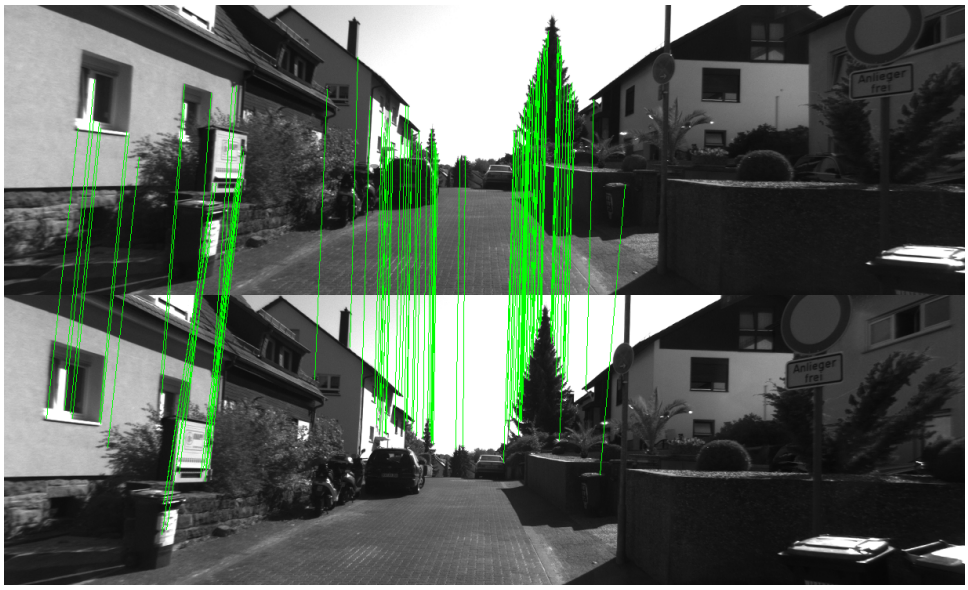


Figura 2.14: Emparejamiento estéreo ORB.

$$k_{2u} - k_{1u} < 1 \tag{2.25}$$

Figura 2.15: Emparejamiento estéreo con eliminación de disparidad con SIFT.

2.6.4. Emparejamiento en Tiempo

Este emparejamiento se realiza entre la imagen actual y la anterior de la misma cámara, ya sea la izquierda o la derecha, la diferencia radica en que no se busca sobre la línea epipolar, sino la búsqueda es en una ventana alrededor del punto, sin embargo el tamaño de la ventana depende de la velocidad con la que se mueva el robot, o la distancia recorrida entre imágenes. En este caso los elementos de la matriz de distancias están determinadas por la ecuación (2.26), donde U es la mitad del tamaño de la ventana.

$$d(\vec{p}_1, \vec{p}_2) = \begin{cases} \sqrt{\sum_{i=1}^{n} (p_{1i} - p_{2i})^2} & si\ abs\,(k_{1v} - k_{2v}) < U\ \&\ abs\,(k_{1u} - k_{2u}) < U \\ 3000 & \text{otro caso} \end{cases} \tag{2.26}$$

En la figura (2.16), se muestra los resultados de este emparejamiento, para imágenes obtenidas con la cámara izquierda.

Figura 2.16: Emparejamiento entre imágenes subsecuentes.

Posteriormente con las parejas generadas, se procede a realizar la eliminación de *outliers*, por medio de la estimación de la matriz fundamental y el algoritmo

RANSAC [17], los resultados se muestran en la figura (2.17).

Figura 2.17: Eliminación de outliers en el emparejamiento en tiempo.

2.6.5. Correlación Cruzada entre Emparejamientos

Se realiza una correlación entre todas las parejas encontradas, para cada pareja encontrada en el emparejamiento estéreo actual $(k_{I_{act}}, k_{D_{act}})$, la segunda característica es buscada en las parejas obtenidas entre el emparejamiento en tiempo para la cámara derecha $(k_{D_{act}}, k_{D_{ant}})$. Si se ha encontrado, se prosigue a buscar la pareja de ésta dentro de las parejas obtenidas en el emparejamiento estéreo de las imágenes anteriores $(k_{D_{ant}}, k_{I_{ant}})$, y nuevamente se repite el proceso para el último emparejamiento en tiempo $(k_{I_{ant}}, k_{I_{act}})$. Si al final se llegó a la característica de la que se ha partido se considera como un buen emparejamiento correlacionado y nos permitirá una mejor estimación de la trayectoria más adelante. En la figura 2.18, se muestra un diagrama de esta correlación. Donde en color verde se muestra una buena correlación y se considera como un punto válido, mientras que las parejas obtenidas en color rojo son descartadas debido a un mal emparejamiento.

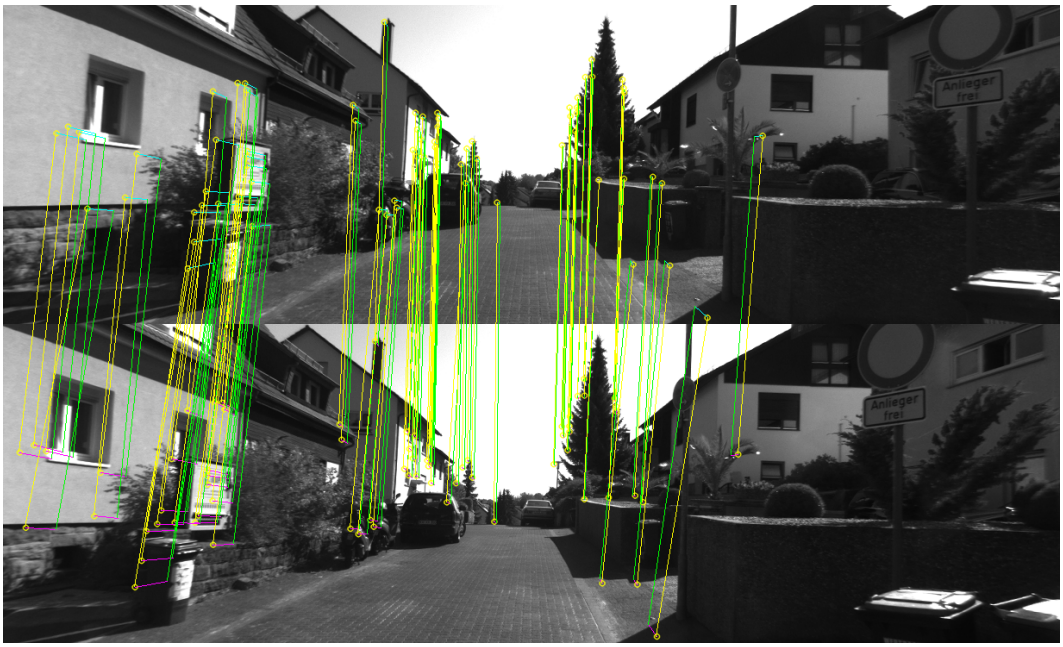


Figura 2.18: Diagrama de correlación cruzada

Capítulo 3

Odometría Visual

La Odometría Visual (OV) es el proceso de estimar la posición y orientación de un agente (por ejemplo, de vehículos, personas o robots), utilizando imágenes captadas por una o más cámaras, este procedimiento también se llama *egomotion*. El termino OV fue acuñado en el año 2004 por Nister [18], se eligió debido a su similitud con la odometría de rueda, en el cual el proceso de estimación de movimiento se realiza de forma incremental integrando el número de vueltas de la rueda. Igualmente OV estima de manera incremental la posición de un agente analizando los cambios inducidos por el movimiento de las imágenes a bordo del agente. Para que un algoritmo de OV funcione eficientemente se debe de contar con un ambiente bien iluminado; así como, la escena debe de ser estática (sin movimiento de los objetos) y con suficiente textura para poder extraer información acerca del movimiento aparente de la escena. Además las imágenes deben de ser capturadas de forma secuencial para garantizar que exista relación entre las imágenes. Para el problema de OV no existe una solución única; sin embargo, una solución óptima dependerá del entorno de navegación; así como de, lo recursos computacionales con los que se cuente.

3.1. Estado del Arte

3.1.1. Historia de la Odometría Visual

El problema de recuperar la posición relativa de la cámara en un entorno tridimensional (3D) a partir de un conjunto de imágenes (calibradas o no calibradas), es conocida en la comunidad de visión por computadora como *structure from motion* (SFM). Sus orígenes se remontan a autores como Longuet-Higgins [19] y Harris [20]. La Odometría Visual es un caso particular de SFM, debido a que SFM es más general y ataca el problema de reconstrucción tridimensional, además la posición de la cámara es refinada con una optimización fuera de línea, donde el tiempo de cálculo incrementa con el número de imágenes capturadas. Mientras que la OV se enfoca a realizar la estimación del movimiento 3D de manera secuencial, y en tiempo real, además de que se pueden utilizar refinaciones para la estimación local de la trayectoria [21] [22].

El problema de estimación del movimiento de un vehículo a través de información visual inicio en la década de 1980 y fue descrito por Moravec [7] denominado *egomotion*. Sin embargo, la mayor parte de la primeras investigaciones en OV fue desarrollada por parte de la NASA en su programa de exploración planetaria [23], [24], entre los aportes que Moravec [25] realizó está de los primeros descriptores de esquinas, junto con el propuesto por Hannah en 1974 [26] y los de Forstner [27] y Harris [6].

3.1.2. OV Estéreo

La mayor parte de la investigación en OV se ha realizado por medio de cámaras estéreo. Matthies y Shafer [23] se basaron en el trabajo de Moravec, utilizando un sistema binocular y el algoritmo de Moravec para la detección y seguimiento de esquina, con la diferencia de que en lugar de utilizar una representación escalar de la incertidumbre aprovecharon el error en la triangulación de las características en la matriz de covarianza y la incorporaron a la estimación del movimiento. Posteriormente Olson [24] extendió este trabajo con la implementación de un sensor de orientación absoluta, así como el detector de esquinas de Forstner, con lo cual mostraron que lo errores de acumulación tienen un crecimiento lineal con respecto a la

distancia viajada, incrementado el error de orientación se incrementa, sin embargo, con la incorporación de un sensor absoluto el crecimiento del error se reduce a una función lineal de la distancia recorrida. Lacroix [28] implemento un enfoque similar, en el cual cambio la forma de detectar las esquinas utilizando imágenes estéreo densas y analiza los picos de la función de correlación.

Un enfoque diferente para la estimación de movimiento y la eliminación de outliers para un vehículo todo terreno fue propuesto por Milella y Siegwart [29], en la cual utilizan la aproximación Shi-Tomasi para la detección de esquinas, donde la estimación es determinada por mínimos cuadrados y un algoritmo iterativo del puntos más cercano (ICP de sus siglas en inglés *Iterative Closest Point*), dicho algoritmo es bastante utilizado para la exploración 3D por medio de láseres.

Nister [18] realiza mejoras a los trabajos anteriores, entre los cuales utiliza un seguimiento de las esquinas, para la eliminación de *outliers* utiliza el algoritmo RANSAC y el problema de estimación de pose lo ve con un problema de mapeo de puntos 3D a puntos 2D.

3.1.3. OV Monocular

La diferencia con el esquema estéreo es que la el movimiento relativo y la estructura 3D deben de ser calculados como un problema de puntos 3D a 2D. También uno de los problemas es que la escala absoluta es desconocida y la escala relativa se encuentra en función de las dos primeras imágenes.

Los trabajos relacionados en el tema se pueden dividir en dos o tres categorías, los que utilizan características, los basados en la apariencia e híbridos. Los métodos basados en características buscan características que sean repetibles y se puedan detectar en las imágenes; los basados en la apariencia analizan la información de intensidad luminosa de todos los píxeles en la imagen o subregiones de la misma; mientras que los métodos híbridos son una combinación de las dos anteriores.

La primera implementación en tiempo real de un esquema de OV monocular fue realizada por Niester [18], en el cual una de las novedades fue la utilización de 5 puntos para dar solución al algoritmo RANSAC para resolver su hipótesis de movimiento. Corke [30] realiza una comparación entre dos métodos de estimación uno por medio de flujo óptico y otro de SFM el cual tiene un mejor desempeño, sin

embargo tiene un alto costo computacional. Otro enfoque lo realizo Tardif [31] en el cual se desacopla la estimación de la orientación y la traslación. Para estimar la orientación utilizan puntos en el infinito, mientras que para la traslación a través del mapa 3D generado, para la eliminación de outliers utiliza RANSAC.

3.1.4. Visual – SLAM

Localización y mapeo simultáneos usando visión es una línea de investigación paralela a OV, en donde existen varias metodologías. Sin embargo hay 2 predominantes:

- 1) *Métodos de Filtrado*, son aquellos que estima el estado en línea donde el estado del sistema consiste en la posición del robot y el mapa. La estimación se calcula y se corrige incorporando nuevas mediciones en cuanto están disponibles, tipicamente utilizan filtros de Kalman Extendidos [32] [33] [34] [35] [36], Filtro de partículas [37] [38] o Filtros de información [39]

- 2) *Métodos No filtrados* o métodos de suavizado, en donde se estima la trayectoria completa del robot a partir del conjunto completo de mediciones, estas aproximaciones se especializan en resolver el problema completo de SLAM (*full SLAM*) y típicamente trabajan en técnicas de minimización del error por medio de mínimos cuadrados.

En general, los algoritmos de SLAM deben considerar diversos aspectos entre los cuales resaltan:

- Modelo de predicción.
- Estimador.
- Asociación de datos.
- Cerrado de bucle.
- Relocalización.
- Identificación de objetos.

- Coherencia del mapa.
- Costo computacional.

3.1.5. OV vs VSLAM

La meta del SLAM es obtener una estimación global y consistente de la trayectoria de un robot. Esto implica llevar un registro del entorno es decir un mapa del mismo, debido a que lo necesitará en el momento en que regrese a un área previamente registrada a lo que se le llama cierre de bucle (*loop closure*), cuando se ha detectado esta condición, la información es utilizada para reducir la desviación del mapa; así como, de la trayectoria del robot, el problema de detectar cuando sucede esta condición es uno de los mayores problemas en el SLAM [18].

Mientras tanto en OV, el objetivo es recuperar la trayectoria del robot de forma incremental es decir, la posición por posición, y posteriormente realizar una optimización en las últimas n imágenes (*windowed bundle adjustment*)

OV se puede utilizar como una herramienta o etapa para la realización de un algoritmo de SLAM, sin embargo para realizar un algoritmo de SLAM completo es necesario agregarle una etapa de cierre de bucle y una etapa de optimización global para obtener un mapa métricamente consistente (en este paso, el mapa es todavía topológicamente coherentes).

3.2. Problema OV

La odometría visual (OV) consiste en la utilización de informacción visual para la estimación de la trayectoria seguida por un agente se encuentra en movimiento adquiriendo imágenes digitales de su ambiente en instantes k, por medio de un sistema de cámaras unidas a él. En el caso de visión monocular el conjunto de imágenes tomadas se denota por $I_{0:n} = \{I_0, ..., I_n\}$, y en el caso de visión estéreo existen dos imágenes por instante de tiempo la izquierda y derecha, las cuales se denotan por $I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$ y $I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$.

Por simplicidad el sistema de referencia de la cámara se asume que es el mismo que el del agente. En el caso de un sistema estéreo, se considera que el sistema de

referencia es el de la cámara izquierda.

La posición de la cámara en dos instantes de tiempos adyacentes, $k-1$ y k, se encuentran relacionados por una transformación de cuerpo rígido $\mathrm{T}_{k,k-1} \in \Re^{4\times4}$, donde $R_{k,k-1} \in SO(3)$ es la matriz de rotación, y $t_{k,k-1} \in \Re^{3\times1}$ es el vector de traslación.

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} \tag{3.1}$$

El conjunto $T_{1:n} = \{T_{1,0}, ..., T_{n,n-1}\}$ contiene todos los movimientos subsecuentes mientras que el conjunto de posiciones de la cámara $C_{0:n} = \{C_0, ..., C_n\}$ contiene las transformaciones de la cámara con respecto al sistema de referencia inicial ($k = 0$) el cual puede ser colocado de forma arbitraria por el usuario. La posición actual de la cámara C_n puede calcularse concatenando todas las trasformaciones de tal forma que $C_n = C_{n-1}T_{k,k-1}$.

El principal objetivo de OV es calcular la transformación relativa $T_{k,k-1}$ a partir de las imágenes I_k y I_{k-1} para concatenar las transformaciones y recuperar la trayectoria completa $C_{0:n}$.

La metodología de OV para cada instante k se resumen de la siguiente forma: 1) adquir la imagen o par de imágenes, según sea el caso, 2) realizar la detección y localización de puntos clave dentro de las imágenes, 3) realizar el emparejamiento (Matching) entre imágenes subsecuentes, 3) calcular el movimiento relativo de la cámara entre los instantes k y $k-1$; finalmente 4) optimización en los últimos m instantes para tener una mayor precisión en la estimación de la trayectoria.

3.3. Estimación de Movimiento

La estimación de movimiento es la etapa básica en OV, debido a que es la etapa en la que se calcula el movimiento relativo de la cámara entre las características (*Keypoints*) de la imagen actual f_k y las de la imagen anterior f_{k-1}. La trayectoria total de movimiento se puede recuperar a partir de la suma o concatenación de la estimación del movimiento relativo, para realizar esta estimación existen diferentes métodos, los cuales dependen del tipo de correspondencias entre las características

de la imagen.

2D-2D: es cuando las características f_k y f_{k-1} se encuentran expresados en coordenadas imagen 2D.

3D-3D: este caso es si f_k y f_{k-1} se están expresados en coordenadas 3D, para este caso es necesario realizar la triangulación de los puntos 3D de un par estéreo en cada instante de tiempo.

3D-2D: es si f_k esta expresada en puntos de la imagen 2D y f_{k-1} en puntos 3D.

3.3.1. Método 2D - 2D

Dada la correspondencia entre las características de dos imágenes contiguas I_k e I_{k-1}, es posible calcular la matriz esencial E, de la cual se conoce que incluye el movimiento relativo de la cámara salvo un factor de escala en la traslación.

$$E_k \simeq \hat{t}_k R_k \tag{3.2}$$

donde $\hat{t}_k$ es la matriz anti simétrica correspondiente al vector $t_k = [t_x, t_y, t_z]^T$.

$$\hat{t}_k = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

La matriz esencial puede calcularse a través de la correspondencia 2D-2D de las características, utilizando las restricciones de la geometría epipolar $p'^T E p = 0$, donde p y p' son el par de características correspondientes.

Para el cálculo de la matriz esencial existen diferentes técnicas en función del número de correspondencias con que se cuente. El caso mínimo de correspondencias necesarias involucra por lo menos 5 características [18], otra manera muy utilizada es por medio del algoritmo RANSAC, el cual proporciona una mayor robustez debido al rechazo de valores atípicos (*outliers*). A continuación se describirá el algoritmo de 8 puntos propuesto por Longuet-Higgins [19] en 1981. Para que dicho algoritmo tenga solución es necesario que los puntos correspondientes no sean coplanares. Con-

siderando 8 puntos la ecuación de restricción epipolar $p'^T Ep = 0$ puede escribirse como se presenta en la ecuación (3.3)

$$\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_8x'_8 & x_8y'_8 & x_8 & y_8x'_8 & y_8y'_8 & y_8 & x'_8 & y'_8 & 1 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{bmatrix} = 0 \tag{3.3}$$

Este sistema $AX = 0$ se puede resolver usando la descomposición de valores singulares (SVD). Si se cuenta con más de 8 restricciones se trata de un sistema sobre determinado y provee robustez ante el ruido.

La solución SVD de A tiene la forma $A = USV^T$, considerando la restricción de $\|E\| = 1$, la matriz esencial calculada tiene la forma $E = USV^T$ donde $S = diag(\sigma_1, \sigma_2, \sigma_3)$. Sin embargo esta estimación de E no cumple con las restricciones interna de una matriz esencial de rango 2, por lo que E es forzada a tener rango 2, colmo se muestra en la ecuación (3.4), donde $\sigma = \frac{\sigma_1+\sigma_2}{2}$.

$$\bar{E} = U\text{diag}\left\{\sigma, \sigma, 0\right\} V^T \tag{3.4}$$

A partir de la matriz esencial modificada $\bar{E}$ pueden extraerse la matriz de rotación R y el vector de traslación t; sin embargo, en forma general existen 4 soluciones para R y t como se muestra en la ecuación (3.5).

$$\begin{aligned} R &= U\left(\pm W^T\right) V^T \\ \hat{t} &= U\left(\pm W^T\right) SU^T \end{aligned} \tag{3.5}$$

donde $W^T = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Para encontrar la solución correcta basta con identificar con cual de las 4 combi-

naciones las características se encuentran frente a la cámara.

3.3.2. Método 3D - 3D

Este método es el más utilizado en sistemas de visión estéreo y cámaras RGB-D. En donde la estimación del movimiento puede calcularse como la trasformación entre dos nubes de puntos. Dados 2 pares de imágenes estéreo de lapsos consecutivos $I_{l,k}$, $I_{r,k}$, $I_{l,k-1}$ y $I_{r,k-1}$, la metodología OV para sistemas estéreo es la siguiente:

- Realizar el cálculo de los puntos característicos en las imágenes $I_{l,k}$, $I_{r,k}$, $I_{l,k-1}$ y $I_{r,k-1}$.
- Realizar la correlación entre las características.
- Calcular a partir del par estéreo el conjunto de puntos 3D X_k, X_{k-1} por medio de triangulación.
- Calcular la transformación T_k, con respecto a las correspondencias ente los puntos 3D X_k y X_{k-1}.

Para calcular el movimiento relativo de la cámara T_k, la solución general es encontrar el movimiento T_k que minimice las distancias L_2 para todos los puntos correspondientes 3D, como se muestra en la ecuación (3.6).

$$T_k = \text{argmin}_{T_k} \sum_i \left\| X_k^i - T_k X_{k-1}^i \right\| \tag{3.6}$$

Para resolver este sistema Arun et al. [40] propone un método en el cual se necesitan por lo menos 3 correspondencias no co-lineales, en donde el cálculo del vector de traslación t_k se define como la diferencia entre el centroide de dos conjuntos de puntos 3D, como se muestra en la ecuación (3.7), donde $\bar{X}_k$ es el centroide de la nube de puntos 3D en el instante k.

$$t_k = \bar{X}_k - \bar{X}_{k-1} \tag{3.7}$$

Mientras que la matriz de rotación R_k puede obtenerse por medio de la descomposición SVD de la matriz A, donde A esta dada por la ecuación (3.8) y la matriz de rotación R_k puede calcularse como en 3.9.

$$A = \left(X_{k-1} - \bar{X}_{k-1}\right)\left(X_k - \bar{X}_k\right)^T = USV^T \tag{3.8}$$

$$R_k = UV^T \tag{3.9}$$

Si las incertidumbres de los puntos 3D son conocidas, se pueden ponderar las estimaciones como lo propone Maimone et al. [41]. La ventaja de este método es que la escala es absoluta y por lo tanto la trayectoria completa se determina directamente concatenando las transformaciones.

3.3.3. Método 3D - 2D

La transformación T_k se calcula minimizando el error de reproyección, considerando las correspondencias entre los puntos 3D X_{k-1} y las proyecciones en el plano imagen p_k, en general el problema se define en la ecuación (3.10).

$$T_k = \text{argmin}_{T_k} \sum_i \left\| p_k^i - p_{k-1}^i \right\|^2 \tag{3.10}$$

Este problema es conocido como perspectiva de n puntos (en inglés perspective from n points PnP), y existe una gran variedad de métodos para resolverlo, uno de los más conocidos es P3P (perspective from three points) [42].

3.4. Desviación (Drifting)

Los esquemas de OV funcionan mediante el cálculo de la trayectoria de la cámara de forma incremental, por lo que los errores generados en cada imagen se van acumulando y generan una desviación en la trayectoria estimada. Para algunas aplicaciones es necesario mantener esta desviación lo más pequeña posible, optimizando de forma local con las últimas m imágenes. Las metodologías que utilizan este esquema se denominan *sliding window bundle adjustment* o *windowed bundle adjustment*, en este enfoque podemos ubicar a Konolige [43], quien demostró que esta técnica puede reducir el error de posición final por un factor de 2 a 5. Obviamente también es posible reducir esta desviación con la utilización de otros sensores como, GPS, láseres,

Unidades inerciales (IMU).

3.4.1. Windowed Bundle Adjustment

En la literatura de OV, SLAM o SFM, este es un método de optimización fuera de línea para la estimación de la posición de la cámara; así como de la características 3D que la relacionan. El problema de WBA, se puede definir como un problema simultáneo de refinamiento de los puntos 3D que describen la geometría de la escena y la posición de la cámara. Para ello se minimiza el error entre los puntos 3D y la reproyección de dichos puntos.

Asumiendo que se han visto n puntos 3D en m imágenes o vistas; x_{ij} es la posición del punto i en el plano imagen de la vista j, la posición de la cámara es $P_k = (x_c, y_c, z_c, \phi, \theta, \psi)$ y el punto i en el espacio 3D es $X_i(x, y, z)$, además la variable u_{ij} representa con 1 si la característica i es observada desde la imagen j en otro caso es 0. Entonces la función a minimizar es el error de reproyección e como se muestra en la ecuación (3.11).

$$e = \underset{X_i, P_j}{\operatorname{argmin}} \sum_{j,j} \|x_{ij} - f\left(X_i, P_j\right)\|^2 \tag{3.11}$$

donde $f(X_i, P_j)$ es la función de reproyección del punto i en la imagen j.

Capítulo 4

Localización del Robot Usando Visión

Uno de los objetivos principales de este trabajo es realizar la odometría visual de robots humanoides y drones. Actualmente en el Laboratorio de Robotica y Visión Artificial (RoVisA) del DCA - CINVESTAV se cuenta con varias plataformas las cuales se han descrito en el Apéndice A. Al robot Darwin OP A.2 se le conectó la cámara estéreo minoru 3D. Sistema en el cual nos enfocaremos en este Capítulo, además de la metodología vista en el Capítulo 3.

Para resolverlo hay varias soluciones de la cuales se encuentran documentadas las siguientes: por su centroide, por medio de SVD y mínimos cuadrados. Sin embargo, se propone la utilización de los algoritmos de trilateración empleados en la geolocalización para resolver el problema de ubicación, que se describirá más adelante. En este Capítulo se realiza una comparación en simulación de los algoritmos mencionados.

4.1. Odometría Visual Estéreo

4.1.1. Pose por Diferencia de Centroides

Esta técnica se explica en la Sección 3.3.2 y básicamente consiste en realizar la resta de los centroides de los puntos 3D en dos instantes de tiempo contiguos (4.1)

con lo cual se determina la rotación por SVD (4.3).

$$t_k = \bar{X}_k - \bar{X}_{k-1} \tag{4.1}$$

$$A = \left(X_{k-1} - \bar{X}_{k-1}\right)\left(X_k - \bar{X}_k\right)^T = USV^T \tag{4.2}$$

$$R_k = UV^T \tag{4.3}$$

4.1.2. Pose por Mínimos Cuadrados

Para la estimación por mínimos cuadrados (LSQ) la función objetivo a minimizar se muestra en la ecuación (4.4).

$$min_{t_k}(R_y(t_{k-1} - t_k)) \tag{4.4}$$

para la resolución de este problema se utilizó el comando de Matlab *lsqcurvefit* (ajuste de curvas por mínimos cuadrados), el cual permite realizar la estimación de t_k dados dos conjuntos de datos, que en esta ocasión son las posiciones de las características en dos instantes consecutivos.

4.1.3. Pose por Trilateración

Traslación

Para calcular el vector de traslación t_k de la cámara es necesario conocer a priori cierta información, como el vector de los puntos 3D emparejados $\begin{bmatrix} x_i^w & y_i^w & z_i^w \end{bmatrix}^T$ con respecto al referencial global, la distancia d_i que existen entre estos puntos y el referencial de la cámara, así como la correspondencia entre ellos.

El problema de trilateración, consiste en determinar la interSección de 3 esferas centradas en los puntos $\begin{bmatrix} x_{c_i}^w & y_{c_i}^w & z_{c_i}^w \end{bmatrix}^T$ y de radio d_i. Partimos de una ecuación básica de observación (4.5), donde $\begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T$, son las coordenadas desconocidas (posición de la cámara).

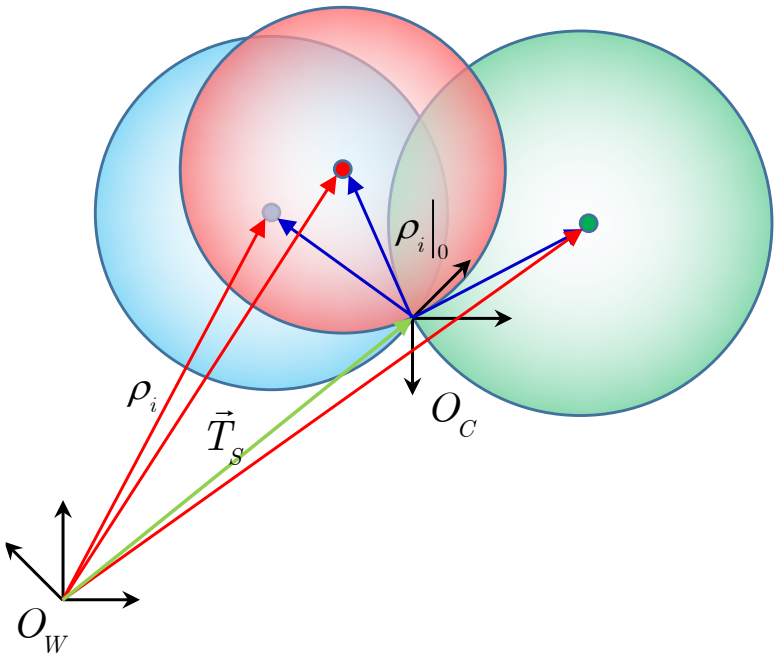


Figura 4.1: Problema de trilateración.

Debido a que se tienen i ecuaciones utilizamos un algoritmo iterativo para resolver el sistema, para lo cual se aproxima la ecuación (4.5) por medio de su serie de Taylor ecuación (4.6).

$$\rho_i = \sqrt{\left(x_{c_i}^w - x_t\right)^2 + \left(y_{c_i}^w - y_t\right)^2 + \left(z_{c_i}^w - z_t\right)^2} \tag{4.5}$$

$$f(x) = f(x_0) + \frac{\partial f(x - x_0)}{\partial x} + \frac{\partial f(y - y_0)}{\partial y} + \frac{\partial f(z - z_0)}{\partial z} + O^2 \tag{4.6}$$

Despreciando los elementos de orden 2 y superiores en su serie de Taylor se tiene la ecuación (4.7).

$$\begin{aligned}\rho_i = \sqrt{\left(x_i^w - x_s\right)^2 + \left(y_i^w - y_s\right)^2 + \left(z_i^w - z_s\right)^2}\Big|_0 \\ + \frac{x_i^w - x_s}{\sqrt{\left(x_i^w - x_s\right)^2 + \left(y_i^w - y_s\right)^2 + \left(z_i^w - z_s\right)^2}}\Big|_0 \\ + \frac{y_i^w - y_s}{\sqrt{\left(x_i^w - x_s\right)^2 + \left(y_i^w - y_s\right)^2 + \left(z_i^w - z_s\right)^2}}\Big|_0 \\ + \frac{z_i^w - z_s}{\sqrt{\left(x_i^w - x_s\right)^2 + \left(y_i^w - y_s\right)^2 + \left(z_i^w - z_s\right)^2}}\Big|_0\end{aligned} \tag{4.7}$$

donde $\vec{T_s} = \begin{bmatrix} x_s & y_s & z_s \end{bmatrix}^T$ es la estimación de la posición de la cámara. Se obtienen i ecuaciones de la forma (4.8).

$$\rho_i = \rho_i|_0 + \frac{\partial \rho_i}{\partial x}\bigg|_0 dx + \frac{\partial \rho_i}{\partial y}\bigg|_0 dy + \frac{\partial \rho_i}{\partial z}\bigg|_0 dz \tag{4.8}$$

Reescribiendo y despejando la ecuación (4.8) en forma matricial queda la ecuación (4.9) donde ρ_i es la distancia del origen del sistema de referencia global al punto 3D y $\rho_i|_0$ es la distancia al mismo punto 3D con respecto al origen del referencial de la cámara, mientras que en la ecuación (4.10) se muestra de forma simplificada.

$$\begin{bmatrix} \rho_1 \\ \vdots \\ \rho_i \end{bmatrix} - \begin{bmatrix} \rho_1 \\ \vdots \\ \rho_i \end{bmatrix}\Bigg|_0 = \begin{bmatrix} \frac{\partial \rho_1}{\partial x} & \frac{\partial \rho_1}{\partial y} & \frac{\partial \rho_1}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial \rho_i}{\partial x} & \frac{\partial \rho_i}{\partial y} & \frac{\partial \rho_i}{\partial z} \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{4.9}$$

$$\vec{\delta} = A\vec{dx} \tag{4.10}$$

Despejando el error de estimación $\vec{dx}$ de la ecuación (4.10), queda la ecuación (4.11), y la corrección de la posición estimada $\vec{T_s}$ queda como en la ecuación (4.12).

$$\vec{dx} = \left(A^T A\right)^{-1} A^T \vec{\delta} \tag{4.11}$$

$$\vec{T_s} = \vec{T}_{s-1} + \vec{dx} \tag{4.12}$$

En cada iteración se calcula un error de estimación $\vec{dx}$ y la condición de paro es que $\vec{dx} < \varepsilon$, donde el valor de ε es fijado por el usuario, en este caso utilizamos un $\varepsilon = 0.001$. Generalmente este algoritmo converge a un resultado aceptable en las

primeras 10 iteraciones; sin embargo, si alguno de los puntos en el emparejamiento está mal, no se tendrá una correcta estimación de la localización, debido a que el error no disminuye a la condición de paro, por ello únicamente se realizan 30 iteraciones, si en estas no se logró un error de estimación aceptable, no es posible estimar la posición, en este momento.

Orientación

Para estimar la orientación del robot, se asumirá la condición de que la cámara solo puede girar alrededor del eje vertical del robot, es decir en un movimiento de paneo, por lo que no tiene movimiento de *tilt*. Esto es válido debido a que para llevar a cabo el mapa únicamente se necesita que el eje óptico de la cámara sea paralelo al suelo.

Al igual que en la traslación es necesario conocer el conjunto de puntos 3D emparejados, entre el referencial global y el referencial de la cámara, a cada conjunto se le determina su centroide de acuerdo a la ecuación (4.13), donde $\mathbf{X}$ es el punto 3D y n el número de puntos de cada conjunto.

$$c = \frac{\sum \mathbf{X}}{n} \tag{4.13}$$

De acuerdo con la Figura 4.2 se puede deducir la Ec. (4.14).

$$\tilde{\mathbf{C}}_{\mathbf{O_C}} = \mathbf{R_y}(\alpha)\left[\vec{\mathbf{C}}_{O_W} - \tilde{\mathbf{T}}\right] \tag{4.14}$$

Reemplazando $\vec{\mathbf{C}}_{O_W} - \vec{\mathbf{T}} = \mathbf{A}$ se llega a la ecuación (4.15) y el ángulo de orientación de la cámara está dado por la ecuación (4.16).

$$\Phi = \begin{bmatrix} A_x & A_z \\ A_z & -A_x \end{bmatrix}^{-1} \begin{bmatrix} \vec{\mathbf{C}}_{O_C x} \\ \vec{\mathbf{C}}_{O_C y} \end{bmatrix} \tag{4.15}$$

$$\alpha = \operatorname{atan2}(\Phi_1, \Phi_2) \tag{4.16}$$

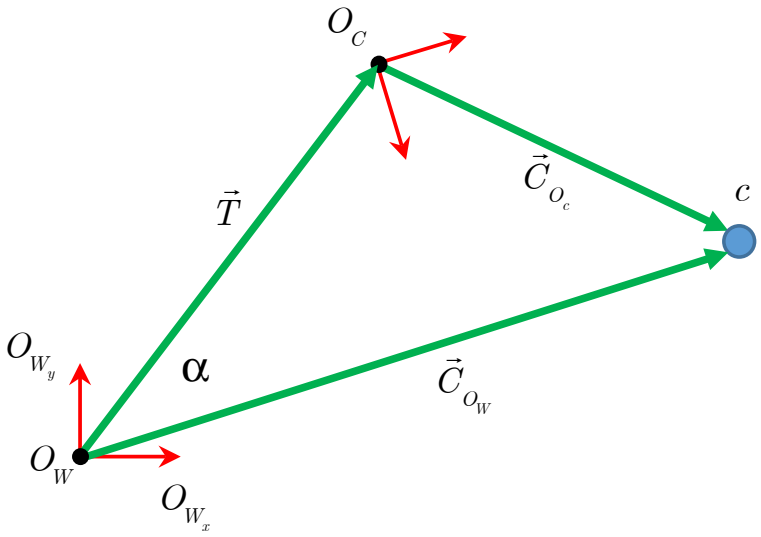


Figura 4.2: Diagrama de orientación.

4.1.4. Simulación

Se realizó un simulador en Matlab para comparar los diferentes técnicas mencionadas. En la figura 4.3 se presenta el diagrama de flujo del simulador implementado, mientras que en la figura 4.4 se muestra la interfaz gráfica del simulador el cual consta de las siguientes etapas:

Generación de Objeto: se genera la nube de puntos 3D de algún objeto.

Trayectoria de la cámara: se introduce la trayectoria de la cámara.

Modelado de la cámara: se modelan dos cámaras en una configuración estéreo, con base en el modelo de proyección en perspectiva pin-hole, para obtener las proyecciones de los puntos 3D en el plano imagen de cada cámara.

Triangulación: a partir de las proyecciones en el plano imagen por medio de triangulación se recupera la información 3D de los puntos, adicionando un ruido Gaussiano para generar la incertidumbre en las lecturas.

Odometría visual: se estima la posición de la cámara entre dos iteraciones subsecuentes y se estima la trayectoria como la suma de la pose actual y la anterior.

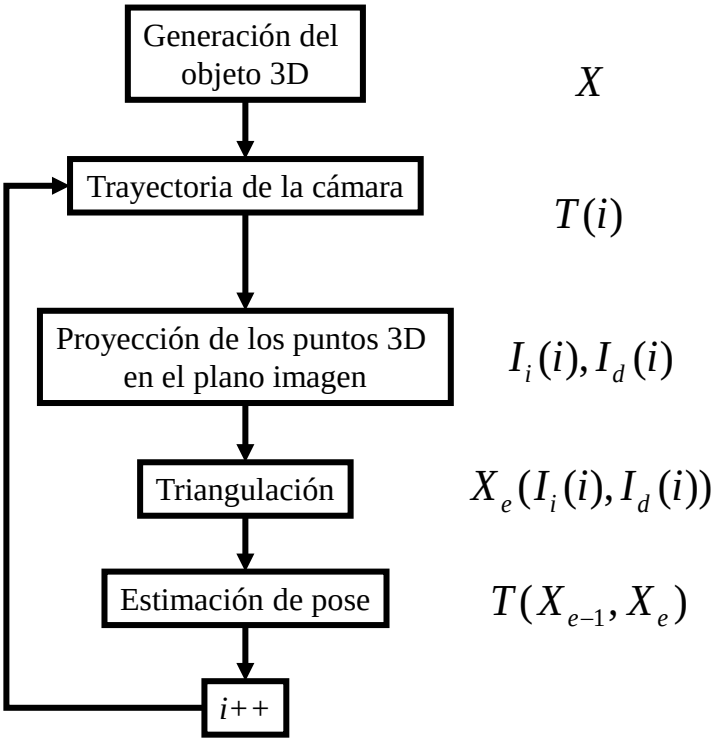


Figura 4.3: Diagrama de flujo del simulador.

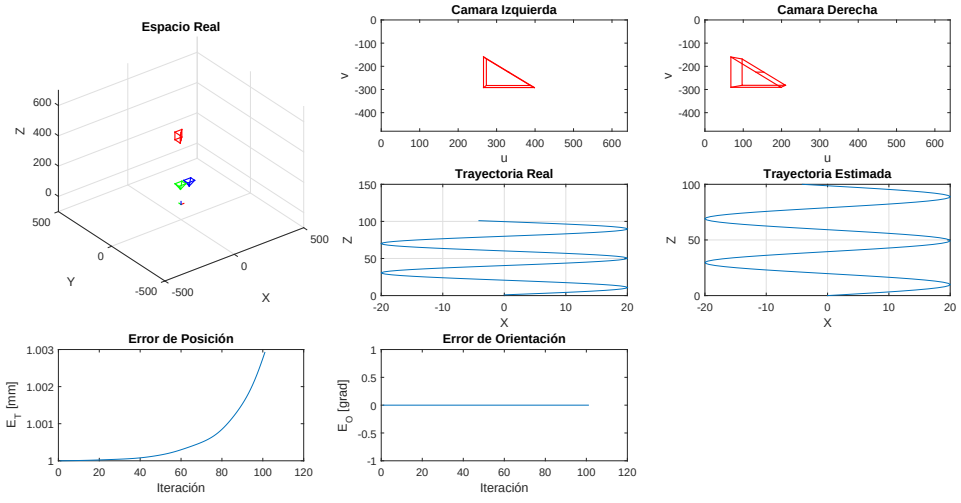


Figura 4.4: Interface gráfica del simulador.

Se realizaron dos trayectorias diferentes para probar el simulador, en la primera de ellas se generó un prisma triangular con 6 puntos cuyo centro se encuentra en $T_o = (0, 0, 450)$ y la trayectoria que se le designó a las cámaras es una trayectoria de acercamiento hacia el objeto con un movimiento lateral sinusoidal, el cual se encuentra descrito por la ecuación (4.17).

$$T_c(i) = \left[20 \sin\left(\frac{i}{2\pi}\right), 0, i\right] \tag{4.17}$$

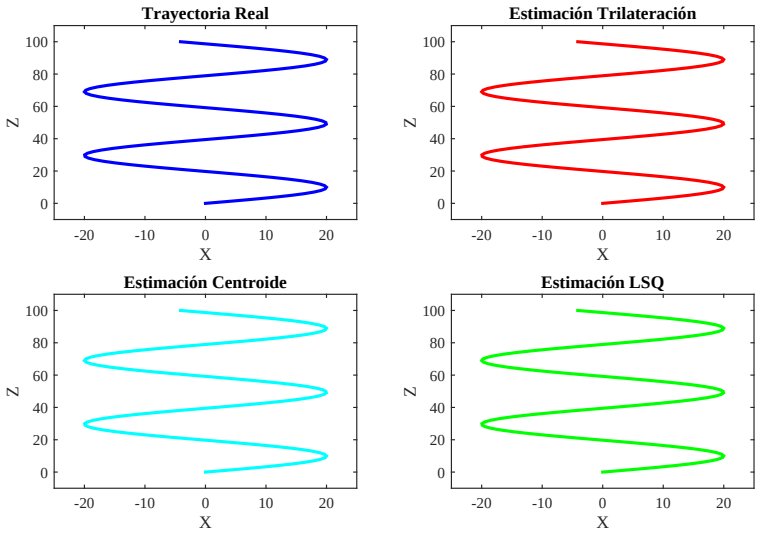


Figura 4.5: Tayectoria 1 en simulación

Para la segunda prueba se mantuvo el mismo objeto y su misma posición; sin embargo, la trayectoria que se ingresó fue una trayectoria circular centrada en el objeto y de radio 450, además la cámara gira 3.6° en cada iteración. Se realizaron un total de 100 iteraciones para completar la trayectoria circular. En la ecuación (4.18) se muestra la trayectoria definida para este experimento. Los resultados de los algoritmos probados se muestran en la figura 4.8; mientras que en la figura 4.9 se presenta la orientación. Cabe señalar que a la estimación de los puntos 3D se le agregó un ruido aleatorio.

$$T_c(i) = \left[450 \sin\left(\frac{2\pi\,(i-1)}{i_{total}}\right), 0, -450 \cos\left(\frac{2\pi\,(i-1)}{i_{total}}\right) + 450\right] \tag{4.18}$$

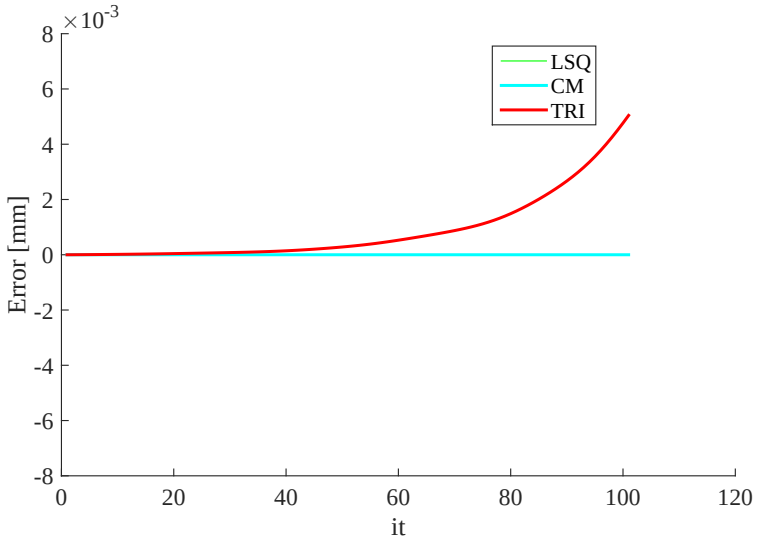


Figura 4.6: Error de traslación de la trayectoria 1

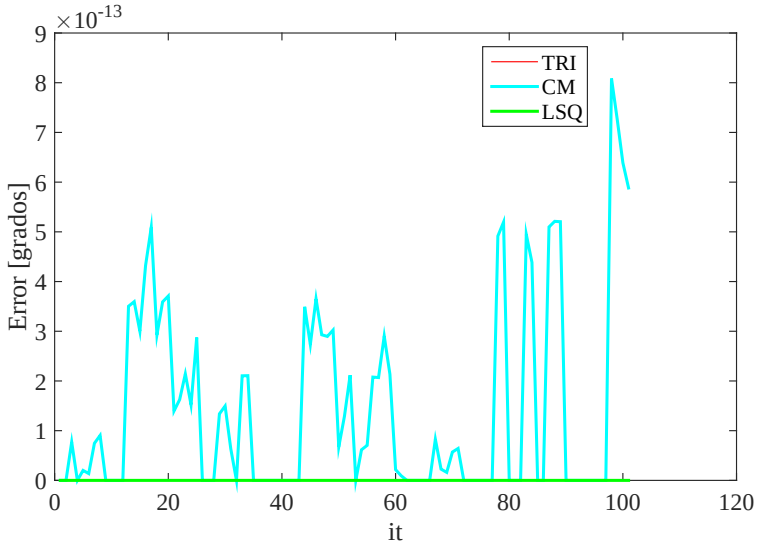


Figura 4.7: Error de la orientación de la trayectoria 1

En la tabla 4.1 se presentan los tiempos para realizar la estimación de la trayectoria en cada iteración para los diferentes métodos utilizados, además se probó con 200 puntos para ver la forma en que incrementa el costo computacional de los algoritmos.

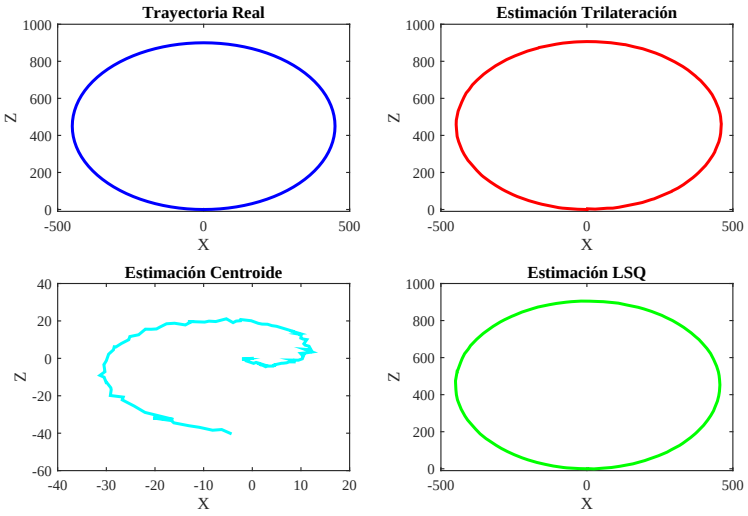


Figura 4.8: Tayectoria 2 en simulación

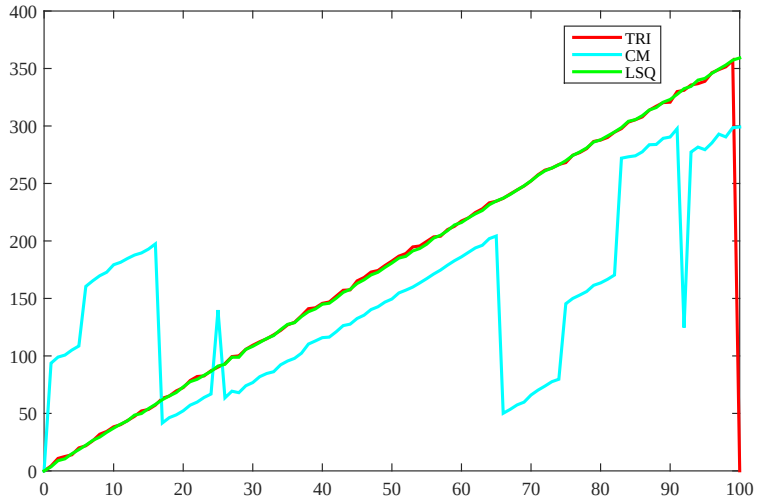


Figura 4.9: Orientación de la cámara en trayectoria 2

4.1.5. Cámara Real

En esta Sección se describen los pasos del algoritmo utilizado, para la localización del robot móvil con visión estéreo. Este algoritmo se implementó en C_{++}, para su aplicación en los diferentes robots móviles. Así mismo, en el algoritmo 1 se presenta el pseudo código implementado.

Algoritmo	**T1 8 puntos**	**T2 8 puntos**	**T1 200 puntos**
CM	0.000243	0.00025	0.00075
LSQ	0.0091	0.0150	0.0087
TRI	0.00084	0.00083	0.0237

Cuadro 4.1: Tiempo en iteración [seg].

- *Paso 1:* Establecer el origen del sistema de referencia global como la posición actual del robot, si se conoce; de lo contrario colocar el origen del sistema de referencia global en la posición actual del robot.
- *Paso 2*: Tomar las fotografías del par estéreo.
- *Paso 3*: Generar los puntos característicos de las imágenes; así como, sus respectivos descriptores con el algoritmo SIFT de acuerdo a la Sección 2.3.3, tanto para el par de imágenes anterior y actual.
- *Paso 4*: Eliminar todos aquellas características repetidas o de baja respuesta.
- *Paso 5*: Generar los descriptores con el algoritmo SIFT de acuerdo a la Sección 2.3.3.
- *Paso 6*: Generar la matriz de distancias estéreo **D** conforme a la ecuación (2.14) de la Sección 2.4.1.
- *Paso 7*: Realizar el emparejamiento estéreo en la línea epipolar y de acuerdo al umbral estéreo.
- *Paso 8*: Eliminar las características que no satisfagan la correlación cruzada.
- *Paso 9*: Calcular los puntos 3D de los puntos emparejados en el paso 8 de acuerdo a las ecuaciones (2.21), (2.22) y (2.23) conforme a la Sección 2.5.
- *Paso 7*: Si es la primera iteración, guardar los datos y regresar al paso 2.
- *Paso 8*: Realizar el emparejamiento con el tiempo para cada cámara Sección 2.4.1.

- *Paso 9*: Realizar la correlación cruzada para asegurar que todos los puntos aparecen en las 4 imágenes.
- *Paso 10*: Calculá la posición XYZ de cada una de las parejas estéreo.
- *Paso 11*: Calcular su localización (traslación y orientación) con los puntos del paso 10 y de acuerdo a las ecuaciones (4.12) y (4.16) de la Secciones 4.1.3 y 4.1.3.
- *Paso 12*: Cambiar los puntos **NO** emparejados del sistema de referencia del robot, al sistema de referencia global, respecto a la traslación y orientación calculadas en el paso 9.
- *Paso 13*: Agregar los puntos del paso 12 al mapa.
- *Paso 14*: Repetir los pasos del 2 en adelante hasta tener un mapa completo.

Los experimentos se realizaron con la cámara Minuro 3D, cuyos parámetros de calibración se presentan en las ecuaciones (4.19) y (4.20), estos parámetros de calibración fueron calculados por medio del Toolbox de visión para Matlab de CALTECH [44], además se montó una cámara superior con la finalidad de comparar los resultados de la trayectoria obtenidos por el algoritmo de SLAM con el movimiento real de la cámara, obtenido por la vista superior.

$$\mathbf{K_I} = \begin{bmatrix} 907.34 & 0 & 293.05 \\ 0 & 907.34 & 240.86 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.19}$$

$$\mathbf{K_D} = \begin{bmatrix} 907.34 & 0 & 296.09 \\ 0 & 907.34 & 240.37 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.20}$$

Para realizar las prueba se montó un ambiente rico en texturas, conformadas por portadas de revistas para que el algoritmo SIFT funcionara correctamente debido a que este necesita imágenes con bastante textura. El ambiente se muestra en la figura 4.10, en donde las revistas fueron colocadas a una distancia de 75 cm del origen.

Algoritmo 1: Algoritmo de OV con Trilateración

```
1  Void OV trilateracion()
2  inicializar_variables(Origen, iteracion)
3  while 1 do
4      izq_act, der_act ← adquirir_imagenes()
5      KI_1, KD_1 ← detectar_KeyPoints(izq_act, der_act)
6      KI_2, KD_2 ← eliminar_Key_response(KI_1, KD_1)
7      KI_3, KD_3 ←eliminar_Key_repetidos(KI_2, KD_2)
8      KI_4, KD_4 ←eliminar_Key_cercanos(KI_3, KD_3)
9      DI_4, DD_4 ←generar_descriptores_SIFT(KI_4, KD_4)
10     D ←generar_matriz_distancias(DI_4, DD_4)
11     KI_5, KD_5, DI_5, DD_5 ← emparejamiento_estereo(KI_4, KD_4, DI_4, DD_4)
12     if iteracion == 0 then
13         KI_ant, KD_ant, DI_ant, DD_ant ← KI_5, KD_5, DI_5, DD_5
14         KI_act, KD_act, DI_act, DD_act ← KI_5, KD_5, DI_5, DD_5
15         Prof_act ←Triangulacion(KI_5, KD_5)
16         Mapa ←inicializa_mapa(KI_act, KD_act, DI_act, DD_act, Prof_act)
17     else
18         KI_act, KD_act, DI_act, DD_act ← KI_5, KD_5, DI_5, DD_5
19         KI_ant6, KI_act6, DI_ant6, DI_act6 ← emp_tiempo(KI_ant, DI_ant, KI_act, DI_act)
20         KD_ant6, KD_act6, DD_ant6, DD_act6 ← emp_tiempo(KD_ant, DD_ant, KD_act, DD_act)
21         (KI_act7, KD_act7, KI_ant7, KD_ant7) ← correlacion_cruzada(KI_ant, KD_ant, KI_act, KD_act, KI_ant6, KI_act6, KD_ant6, KD_act6)
22         Prof_act ←Triangulacion(KI_act7, KD_act7)
23         Prof_ant ←Triangulacion(KI_ant7, KD_ant7)
24         pose ←Trilateracion(Prof_act7, Prof_act7)
25         α ←orientacion(Prof_act7, Prof_act7, pose)
26         Tray ←Trayectoria(pose, α)
27         KI_act8, KD_act8, DI_act8, DD_act8, Prof_act8 ←cambio_referencia()
28         Mapa ← agrega_mapa(KI_act8, KD_act8, DI_act8, DD_act8, Prof_act8)
29         KI_ant, KD_ant, DI_ant, DD_ant ← KI_5, KD_5, DI_5, DD_5
30     iteracion++
31     guarda_mapa(Mapa)
```

Figura 4.10: Escenario de pruebas.

Se realizó una trayectoria moviéndose 25 cm hacia el frente y posteriormente 25 cm de costado, para formar una «L» invertida. A lo largo de la trayectoria establecida para esta prueba se tomaron 32 pares de imágenes. En la figura 4.11 se muestra de color verde la trayectoria real, calculada con la cámara superior, mientras que en color rojo se presenta la trayectoria de la cámara calculada por el algoritmo presentado.

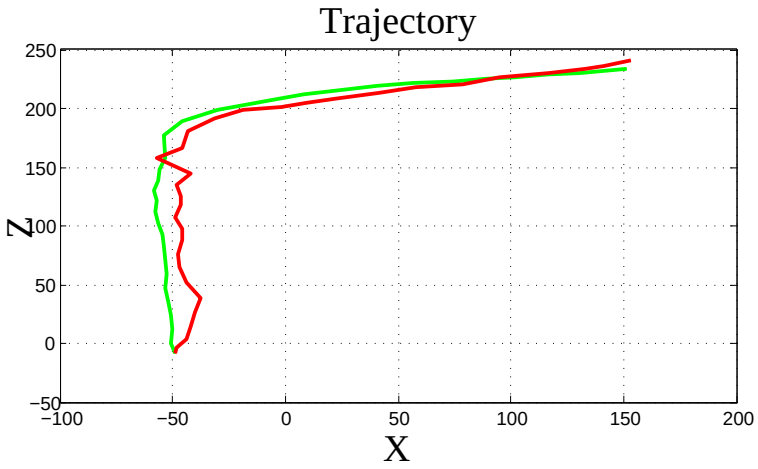


Figura 4.11: Trayectoria de la camára.

En la figura 4.12 se muestra el mapa de características generado, donde los puntos rojos son los puntos 3D del mapa, los azules los puntos emparejados del mapa, los puntos verdes son los puntos emparejados de la cámara y los amarillos los no emparejados.

Así mismo en las figuras 4.13 y 4.14 se muestran los errores en la trayectoria y

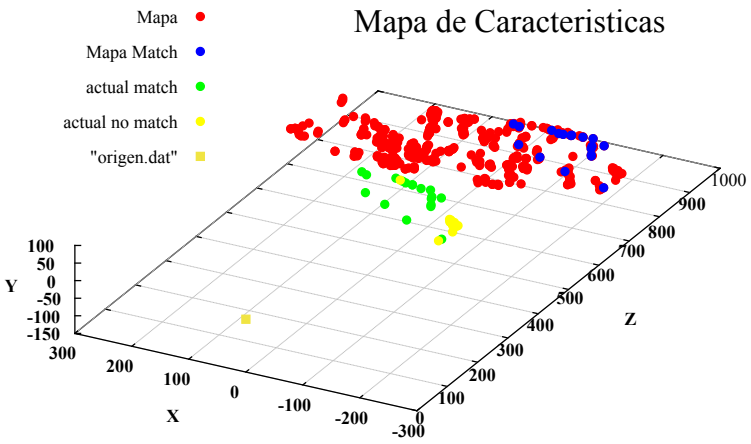


Figura 4.12: Mapa de características.

orientación respectivamente, entre los cálculos del algoritmo y la vista superior, en donde se aprecia que el error de posición es menor a 2 cm, mientras que el error de orientación es menor a 0.15 radianes.

Figura 4.13: Error de la trayectoria.

En cada iteración se encontraron aproximadamente 300 características por imagen estéreo; en el emparejamiento estéreo se realizaron alrededor de 70 parejas y en el emparejamiento con el mapa un promedio de 30 parejas de puntos, con los umbrales establecidos.

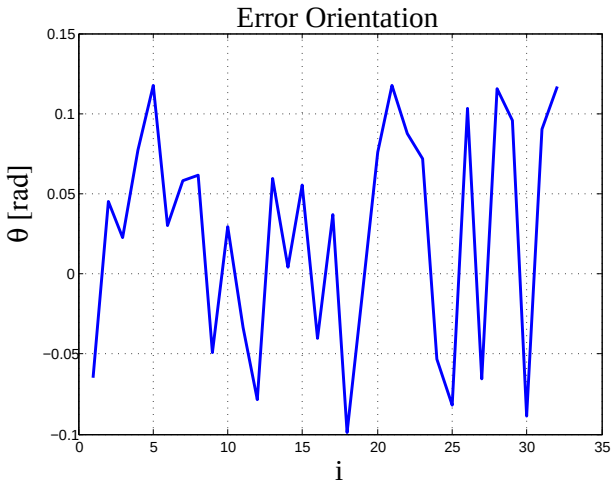


Figura 4.14: Error de la orientación.

4.1.6. Conjunto de Datos KITTI

Los conjuntos de datos (*DataSets*) han desempeñado un papel clave en el progreso de muchos campos de investigación proporcionando ejemplos concretos de la realidad. En este trabajo se utilizó el dataset de *KITTI Vision Benchmark Suite*, elaborado por el Instituto Tecnológico de Karlsruhe, Alemania [45]. Esta base de datos proporciona las imágenes de una trayectoria realizada por el vehículo, así como la referencia real (Ground Truth), también ofrecen las lecturas de otros sensores, sin embargo esas no se consideraron debido a que este trabajo es únicamente con visión.

Para la construcción del dataset, se utiliza un vehículo Volkswagen Passat mostrado en la figura 4.15, el cual cuenta con los siguientes sensores:

- 1 Sistena de Navegación inercial (GPS/IMU): OXTS RT 3003
- 1 Escaner Laser: Velodyne HDL-64E
- 2 Camaras en escala de grises, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3M)
- 2 Camaras a color, 1.4 Megapixels: Point Grey Flea 2 (FL2-14S3C-C)
- 4 Lentes Varifocales, 4-8 mm: Edmund Optics NT59-917

A fin de realizar una comparación con los resultados obtenidos de la implementación del algoritmo propuesto, se utilizó la secuencia 11 del dataset, el cual en la

Figura 4.15: Vehículo utilizado para el dataset.

figura 4.16, se presenta la trayectoria real realizada por el vehículo y la trayectoria estimada por medio del algoritmo de trilateración de puntos característicos.

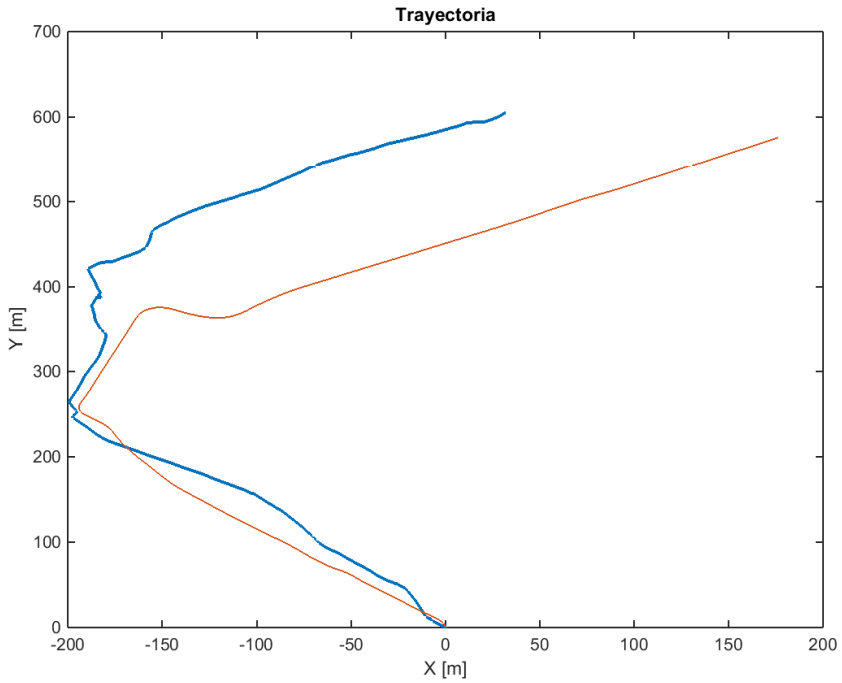


Figura 4.16: Trayectoria real.

En este caso se observa que la trayectoria estimada es semejante a la real, la distancia real recorrida es de 854.31 m , y la distancia estimada es de 807.09 m, de donde se observa un error en la distancia recorrida del 5.5 %.

4.2. Relocalización de Robots Móviles

Como se mencionó en la introducción de este trabajo uno de los objetivos es realizar un algoritmo de autolocalización para robot humanoides que juegan fútbol, para ellos se describen dos algoritmos de relocalización. El primero fue un trabajo de maestría desarrollado por Ricardo Carrillo [46] a través de un algoritmo de regresión general y el segundo es el que se propone en este trabajo basado en un método de trilateración.

4.2.1. Algoritmo de Regresión General

El método del cual se hablará en esta Sección se basa en el artículo de Andrew P. Gee et al. [47], el cual utiliza imágenes sintéticas, cámara RGB-D y un algoritmo de regresión general para comparar la imagen actual con un mapa previamente almacenado. La metodología descrita en el artículo es la siguiente:

1. Generar l imágenes sintéticas a partir de la vista actual de la cámara, lo que en total resultará en $m = l + 1$ imágenes.

2. Reducir la resolución de las imágenes; 80×60 y 20×15 pueden utilizarse con resultados favorables.

3. Imágenes en escala de grises son suficientes para resolver el problema sin perder mucha información, es además es recomendable normalizar dicha intensidad de acuerdo a la ecuación (4.21)

$$c_{ij} = \frac{c_{ij} - \bar{c}_i}{\sigma_{ci}} \tag{4.21}$$

   donde c_i es la media de la intensidad y σ_{ci} es la desviación estándar de la i-esima imagen.

4. Construir $n \times i$ vectores $I_j = [u_j, v_j, c_j]$, en donde: n es el número de píxeles de la imagen, (u_{ij}, v_{ij}) son las coordenadas del píxel y c_{ij} es la intensidad del píxel. Finalmente, cada imagen contará con una matriz I_i formada por n vectores I_j.

5. Junto con la matriz I_i se debe almacenar la posición $x_i = [t, \ln(q)]$ de la cámara en el espacio, donde t es el vector de traslación y q es el cuaternión que describe la orientación de la posición.

6. Para estimar la posición de relocalización se utiliza un estimador Nadaraya-Watson [48] con un kernel Gausiano como se muestra a continuación:

$$\tilde{x} = \frac{\sum_{i=1}^{m} x_i d_i}{\sum_{i=1}^{m} d_i} \tag{4.22}$$

donde:

$$d_i = \exp\left(-\frac{1}{n\alpha}\sum_{j=1}^{n}\left(c_{0j} - c_{ij}\right)^2\right)$$

4.2.2. Algoritmo de Trilateración

El algoritmo propuesto para la relocalización utiliza la misma técnica de triangulación para estimar la pose del robot que el sistema de OV mencionado en la Sección 4.1.3. Sin embargo, para llevar a cabo la relocalización es necesario tener un mapa del entorno en donde se desplaza el robot, para este caso se utiliza la metodología de OV para estimar la pose del robot y generar un mapa de características de acuerdo a lo que ve y a su posición. Este mapa es de características en 3D, es decir, para cada punto detectado por el par estéreo en el mundo 3D se guarda junto con su descriptor, ya sea SIFT u otro, como se observa en la ecuación (4.23), donde u, v son las coordenadas en el plano imagen la primera vez que se visualizaron, $[x\ ,\ y,\ z]$, son las coordenadas en el mundo real, D es el descriptor asociado a dicho punto y j es el número de imagen en la que fue detectada.

$$M\,[i] = \begin{bmatrix} u & v & x & y & z & D & j \end{bmatrix} \tag{4.23}$$

La metodología del algoritmo se describe a continuación:

1. Generar mapa de características $M[i]$ con OV.
2. Adquirir imágenes del par estéreo.
3. Identificar puntos característicos en cada imagen estéreo.
4. Realizar la búsqueda de correspondencias en el par estéreo utilizando la restricción de geometría epipolar.
5. Eliminar puntos característicos de baja respuesta, repetidos o cercanos.
6. Realizar la búsqueda de correspondencias entre los puntos identificados anteriormente con los almacenados en el mapa.

7. Calcular la posición 3D con respecto al referencial de la cámara de los puntos emparejados con el mapa.
8. Estimar la posición y orientación de acuerdo al algoritmo de trilateración presentado en las Secciones 4.1.3 y 4.1.3.

Algoritmo 2: Algoritmo de Relocalización con Trilateración

```
Void OV trilateracion()
Mapa ←cargar_mapa();
izq_act, der_act ← adquirir_imagenes()
KI_1, KD_1 ← detectar_KeyPoints(izq_act, der_act)
KI_2, KD_2 ← eliminar_Key_response(KI_1, KD_1)
KI_3, KD_3 ←eliminar_Key_repetidos(KI_2, KD_2)
KI_4, KD_4 ←eliminar_Key_cercanos(KI_3, KD_3)
DI_4, DD_4 ←generar_descriptores_SIFT(KI_4, KD_4)
D ←generar_matriz_distancias(DI_4, DD_4)
KI_5, KD_5, DI_5, DD_5 ← emparejamiento_estereo(KI_4, KD_4, DI_4, DD_4)
Prof_act ←Triangulacion(KI_5, KD_5)
pose ←Trilateracion(Prof_act, Prof_mapa)
α ←orientacion(Prof_act, Prof_mapa, pose)
```

4.2.3. Resultados Experimentales de Relocalización

Se implementaron ambas metodologías en C_{c++} utilizando la cámara Minoru 3D para adquirir el par estéreo usado por el algoritmo de trilateración y únicamente las imágenes de la cámara izquierda para el algoritmo de regresión general. Además se colocó una cámara VGA en la parte superior para generar el mapa en el caso de regresión.

Para la generación del mapa para el algoritmo de regresión se tomaron 20 imágenes en diferentes posiciones, visualizando en el ambiente mostrado en la figura 4.10, del cual se obtuvo también el mapa de características por medio del algoritmo de trilateración. La cámara superior es únicamente para comparar ambos algoritmo de relocalización con la posición real de la cámara (*Ground Truth*).

En la figura 4.17 a), se muestra el mapa de características generado por el algoritmo de OV, mientras que en la figura 4.17 b) se presenta una representación del mapa almacenado para el algoritmo de regresión general.

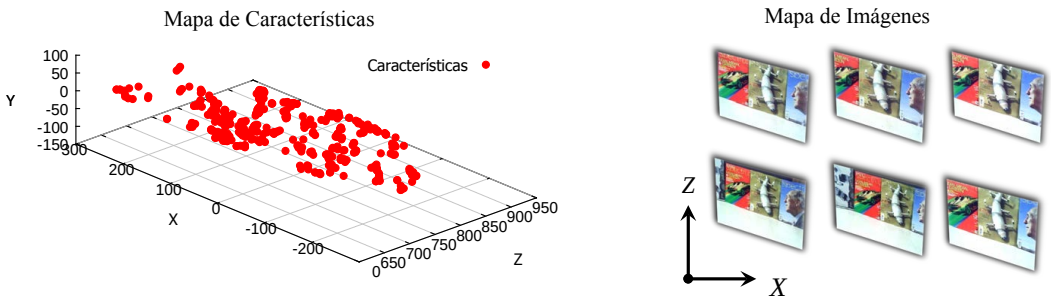


Figura 4.17: a) Mapa de Características, b) Mapa de imágenes.

Para ejecutar los algoritmos de relocalización se colocó a la cámara en diferentes posiciones. Los resultados obtenidos se muestran en la figura 4.18 en donde los puntos de color rojo son la posición real de la cámara calculada con la vista superior, de color azul se muestra la relocalización por el método de regresión general y de color verde la estimación a través del algoritmo de relocalización.

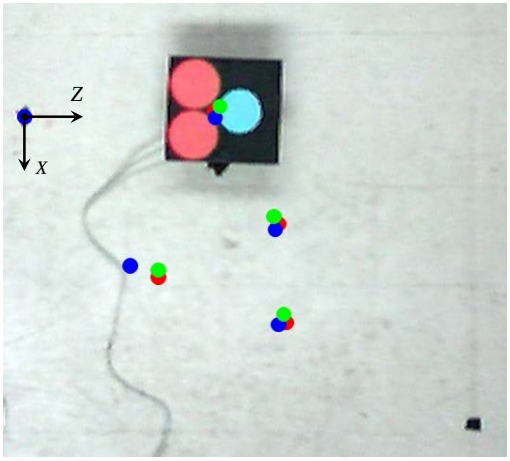


Figura 4.18: Resultados de relocalización.

En esta prueba se observó que la estimación con el algoritmo de regresión tiene errores mayores en posiciones donde no se tomaron fotografías del mapa, a diferencia del algoritmo de trilateración en el cual el error disminuye. Sin embargo se aprecia que ambos algoritmos cuentan con un buen desempeño en ambientes estáticos, donde el error de posición es menor a 20 mm, considerando el error como la distancia euclidiana entre la posición real y la estimada.

4.3. Odometría Visual Monocular para un Cuadrirotor

Actualmente en el laboratorio se cuenta con un cuadrirotor beebop, descrito en la Sección A.5, el cual tiene una cámara, cuyo plano imagen es "paralelo" al suelo, debido a esto se optó por la metodología de OV de 2D - 2D que se explicó en la Sección 3.3.1.

Para la detección de características se utilizó el detector de esquinas de Harris y Shi-Tomasi, mientras que para el seguimiento de características se utilizó la metodología de flujo óptico. A continuación se muestran los pasos que se siguieron para la implementación en C_{++} de la OV para el cuadrirotor.

- *Paso 1:* Como en todos los sistemas de OV se establece el origen del sistema de referencia como la primera posición del robot. Además, se inicializan todas variables que se utilizaran en la ejecución del programa.
- *Paso 2:* Se adquiere una imagen de la cámara montada en el cuadrirotor, en este caso es una imagen con una resolución VGA(640×480).
- *Paso 3:* Se extraen los puntos característicos en la imagen actual por medio del algoritmo de detección de esquinas de Harris o Shi-Tomasi (Sección 2.3.1).
- *Paso 4:* Se eliminan los puntos característicos que se encuentran cerca de los bordes de la imagen.
- *Paso 5:* Se guardan los puntos característicos, si es la primera iteración se prosigue con el *paso 2*, de lo contrario con el *paso 6*.
- *Paso 6:* Se realiza el seguimiento de las características de acuerdo al algoritmo de flujo óptico descrito en la Sección 2.4.2.
- *Paso 7:* Filtrado de puntos, es decir se eliminan todos los puntos que tengan un mal seguimiento.
- *Paso 8:* Calcular la matriz esencial E entre los dos conjuntos de características conforme a la ecuación (3.4).

- *Paso 9:* Estimar las probables matrices de Rotación y los vectores de Traslación como se indica en la ecuación (3.5).
- *Paso 10:* Identificar la matriz de rotación y el vector de traslación correctas.
- *Paso 11:* Generar la pose actual del cuadrirotor.
- *Paso 12:* Calcular la escala relativa.
- *Paso 13:* Repetir los pasos, a partir del *paso 2*, hasta que se termine el movimiento del dron.

Algoritmo 3: Algoritmo de OV para cámara monocular

```
1  Void OV monocular()
2  inicializar_variables(Origen, iteracion)
3  it = 0
4  inicializar(tray)
5  while 1 do
6      I ← adquirir_imag()
7      K_1 ← detectar_KeyPoints_ShiTomasi(I)
8      K_2 ← eliminar_key_borde(K_1)
9      K_act ← k_2
10     if it==0 then
11         K_ant ← K_2
12     else
13         K_ant2, K_act2 ←Tracking_flujo_optico(K_ant, K_act)
14         K_ant3, K_act3 ←eliminar_outliers( K_ant2, K_act2)
15         E ←Calcular_esencial(K_ant3, K_act3)
16         R_1, R_2, T_1, T_2 ←estimar_matrices(K_ant3, K_act3)
17         R, T ←identificar_correctas(R_1, R_2, T_1, T_2)
18         tray ←trayectoria(R, T)
19         if it > 1 then
20             Escala ← Escala_relativa(K_act3, K_ant3, K_ant_ant)
21         K_ant_ant ← K_ant3
22         K_ant ← K_act3
23     it++
```

Para fines prácticos el cuadrirotor se voló de manera remota, siguiendo una trayectoria recta y posteriormente una vuelta hacia la izquierda, del cual se tomaron las lecturas del GPS para comparar con los resultados obtenidos, a través del sistema de OV.

En la figura 4.19, se presenta la trayectoria que siguió el cuadrirotor conforme a las lecturas obtenidas con el GPS, mientras que en la figura 4.20 se presenta la trayectoria calculada por el sistema de OV.

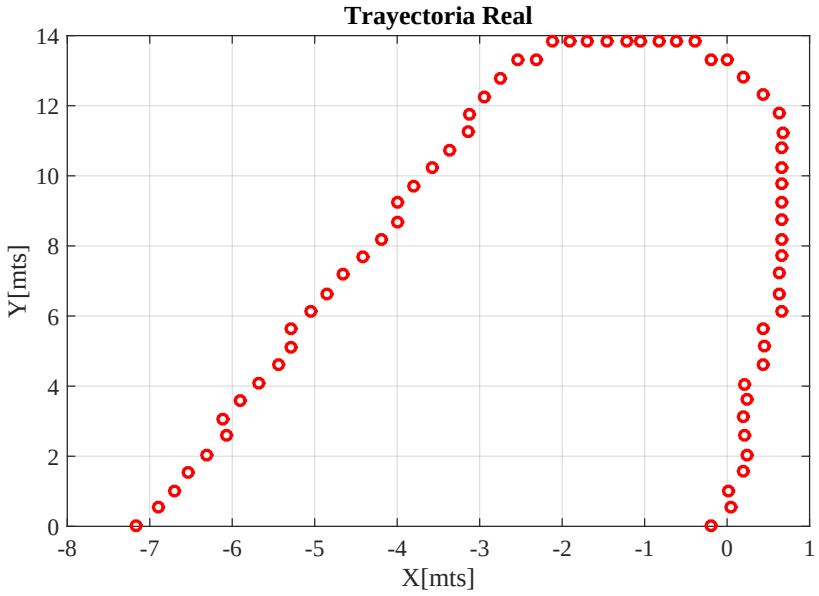


Figura 4.19: Trayectoria real del cuadrirotor.

Como puede observarse la forma de ambas trayectorias es muy similar, sin embargo la escala es diferente esto debido a que con esta metodología se necesita de mediciones extras para lograr determinar la escala absoluta de la trayectoria estimada.

Con el fin de validar sistema de OV presentado se probó el algoritmo implementado con una base de datos del Instituto de Tecnología de Karlsruhe de Alemania. Esta Universidad cuenta con una base de datos con la cual se prueba una gran cantidad de algoritmos de OV y de Flujo óptico.

The KITTI Vision Benchmark Suite es una base de datos que contiene alrededor de 20 secuencias de imágenes, tomadas mediante un vehículo que se mueve en los

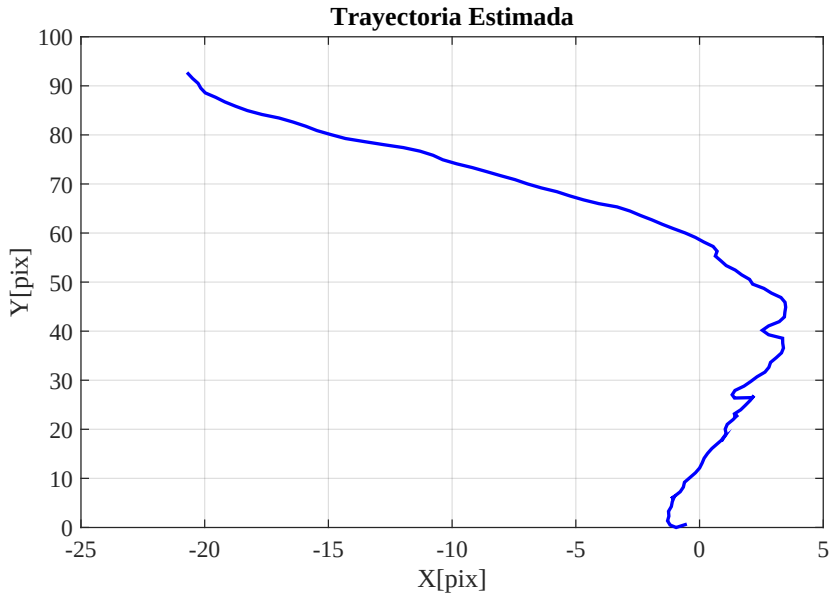


Figura 4.20: Trayectoria Estimada del cuadrirotor.

alrededores del Instituto. Para la validación, se utilizó la secuencia 11, la cual tiene una ruta de más de 600 metros. En la figura 4.21, se muestra el cálculo del flujo óptico, por medio de las líneas rojas, las cuales asocian la posición de dos puntos característicos entre dos imágenes consecutivas.

Figura 4.21: Flujo óptico en KITTI.

En la figura 4.22 se muestran los resultados obtenidos, en color rojo la ruta real

y la estimada en color azul. Debido a que en la base de datos se encuentran los datos de la ruta real seguida en esa secuencia, se puede realizar una comparación, de donde se obtiene un error del 1.5 % a lo largo de una ruta de más de 600 m.

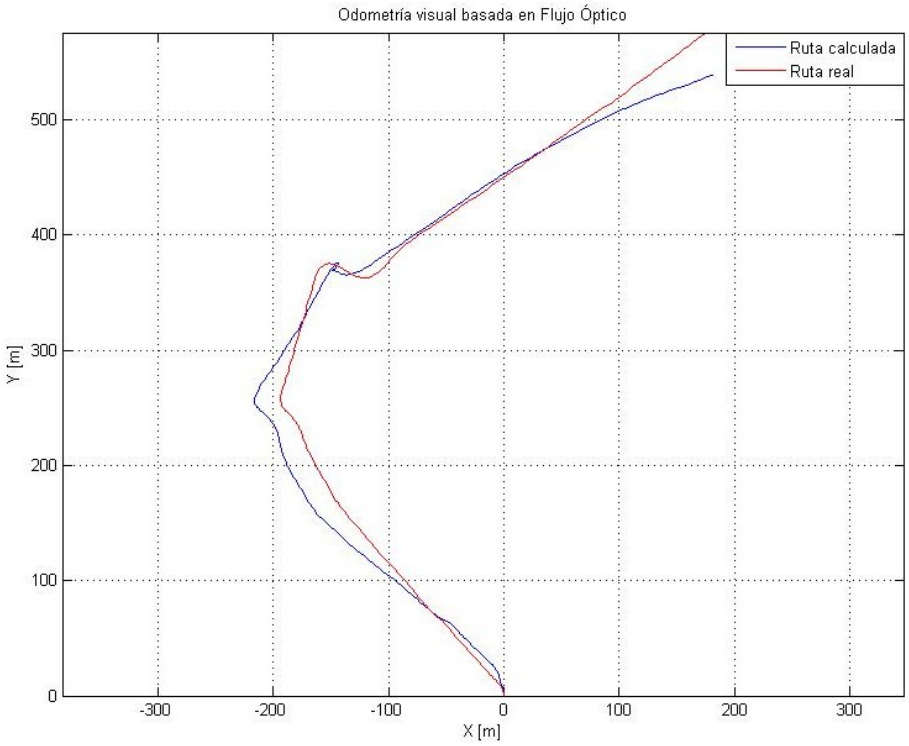


Figura 4.22: Trayectoria estimada del Benchmark.

Capítulo 5

Modelado de un Robot Humanoide

En este Capítulo se hablará acerca de la obtención del modelo cinemático directo e inverso de un robot humanoide, en este caso del robot AH1N2, desarrollado en el CINVESTAV.

5.1. Arquitectura del Robot

La arquitectura del robot AH1N2 fue desarrollado para competiciones de futbol, en la categoría de *Kid Size* de Robocup [49]. Se trata de un robot humanoide con un torso, una cabeza, dos brazo y dos piernas, el cual se muestra en la figura 5.1 donde se aprecian sus principales partes así, como, la proporción relativa entre ellas. Este robot humanoide es una modificación de la primera versión AH1N1, desarrollado por el Dr. J.M. Ibarra, Felipe Leyva, Manuel Hunter y Rafael Cisneros.

5.1.1. Descripción de la Partes del Robot Humanoide

A continuación se hace una breve descripción de las partes constitutivas de la arquitectura mecánica de este humanoide:

Tórax: Es una caja rectangular hecha de alucobon, en la que se encuentra el sistema computacional, sensores inerciales y baterías; así mismo es el soporte para las cuatro cadenas cinemáticas que forman la cabeza, los brazos y la cintura, en la parte superior del tórax se encuentran 3 servomotores RX-28 que son la base

de los brazos y cabeza. El servomotor de la cabeza proporciona el movimiento azimutal del cuello, mientras que los servomotores de los brazos, proporcional el movimiento de flexión-extensión. Las dimensiones del tórax son de $120 \times 72 \times 163$ mm y tiene un peso de 1.5 kg. En su interior también se encuentran varias piezas impresas en 3D, para facilitar la colación de las tarjetas electrónicas, sensores, baterías y el fácil acceso a los puertos con los que cuentan.

Cintura: Se encuentra unida a la parte inferior del tórax, es de 2 gdl (rotación y flexo-extensión), cuyos actuadores son servomotores RX-28, el servomotor asociado con la flexo-extensión de cadera se encuentra unido a la parte inferior del tórax, mientras que el servomotor de rotación se encuentra debajo de él, unido a través de un soporte de U, colocado de tal forma que los ejes de rotación se cruzan. Esta configuración de la cintura le permite levantarse fácilmente después de una caída, también ayuda a la locomoción y a mantener un equilibrio dinámico.

Cabeza: Es una cámara estereoscópica Minoru 3D, unida al tórax a través del cuello, el cual es formado por dos gdl, uno para azimut y el otro para elevación de cabeza (pan y tilt). El primer servomotor es colocado dentro del tórax entre los servomotores del hombro, cuyo eje de rotación es vertical, el servomotor de elevación es montado sobre el tórax usando un soporte U y los actuadores utilizados son servomotores dynamixel RX-28.

Brazos: Estos están formados por 5 gdl más uno para agarrar objetos, y así tener una capacidad mínima de manipulación. Los brazos tienen: *i)* un hombro de tres gdl (abducción-aducción de hombro, flexo-extensión y rotación interna-externa), *ii)* un codo de flexo-extensión de un grado de libertad; y *iii)* un grado de libertad para la pronación-supinación de muñeca. El servo para la flexo-extensión del hombro está dentro del tórax de modo que su eje de rotación es horizontal y se coloca en el plano frontal; el siguiente servomotor acoplado a él, tiene su eje de rotación perpendicular al anterior y proporciona el movimiento abducción-aducción, así mismo un tercer servomotor con eje de rotación perpendicular a los dos anteriores, proporciona el movimiento de rotación interno-externo, dando como resultado un hombro esférico. Al servo de rotación interno-externo se le coloca mediante un soporte en U el servomotor que efectúa la flexión-extensión

del codo y ambos ejes son perpendiculares; finalmente, unido al último servomotor, son montados, dos servomotores uno para la pronación-supinación de la muñeca, y otro para abrir y cerrar la pinza de mano.

Piernas: Con el fin de permitir la implementación de varios patrones de caminata, las piernas deben tener tres articulaciones (cadera, rodilla y tobillo) con un total de al menos seis grados de libertad, en este caso la estructura es la siguiente: *i)* la cadera tiene tres movimientos: abducción-aducción, flexo-extensión y rotación interna-externa; *ii)* la rodilla tiene un grado de libertad para la flexo-extensión; y *iii)* el tobillo tiene dos movimientos, flexión plantar y dorsal e inversión-eversión. Las piernas se encuentran ensambladas de la siguiente manera: el eje vertical del servomotor de la cadera está acoplado a un soporte en U de doble anchura donde los movimientos de abducción-aducción y de flexo-extensión están montados, todos los ejes son perpendiculares, por lo que tenemos una cadera esférica. En el último servo de la cadera se monta el actuador de la rodilla es por medio de dos placas paralelas, dos placas similares conectan el actuador de rodilla con un servo bloque de dos en el pie, que producen la flexión plantar-dorsal y la inversión-eversión del tobillo.

La arquitectura resultante del robot es mostrada en la figura 5.1, en donde se observa la localización de los referenciales de cada eslabón de la cadena cinemática, en la tabla 5.1, se presentan las dimensiones de cada uno de los eslabones, mientras que en la tabla 5.2, se muestran los límites para cada articulación [50].

5.2. Modelo Geométrico

Para determinar el modelo geométrico del robot se considera como una estructura arbolecente, donde el origen, referencial 0 es colocado en el tórax del robot con el eje $z+$ apuntando hacia arriba, el eje $x+$ apuntando hacia el frente y el eje $y+$ hacia la izquierda del robot, las piernas y brazos son especificadas utilizando la siguiente variación de los parámetros de Denavit-Hartenberg $(\gamma_j, b_j, \alpha_j, d_j, \theta_j, r_j)$ [51].

$$^{i}T_j = R_z(\gamma_j)T_z(b_j)R_x(\alpha_j)T_x(d_j)R_z(\theta_j)T_z(r_j) \tag{5.1}$$

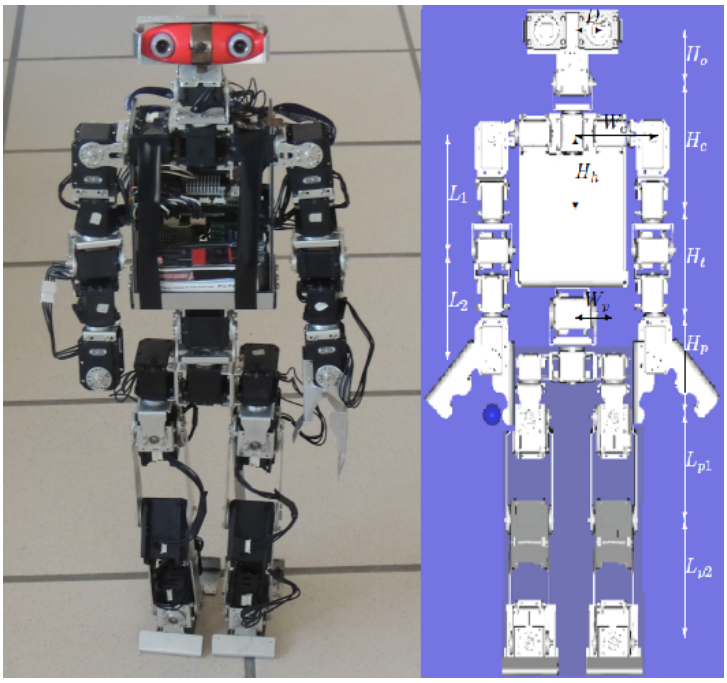


Figura 5.1: AH1N1.

Entonces ${}^0T_h$, ${}^0T_rh$, ${}^0T_lh$, ${}^0T_rf$, ${}^0T_lf$, definen la posición relativa de la cabeza, mano derecha, mano izquierda, pie derecho y pie izquierdo respectivamente, referenciados al tórax. Por ejemplo, la posición de la mano derecha con respecto a la cabeza se puede calcular fácilmente como se muestra en la ecuación (5.2).

$$ {}^hT_{rh} = {}^h\ T_0^0 T_{rh} \tag{5.2}$$

Para ello no son necesarios cálculos complejos, dado que para realizar el cálculo de la inversa de una transformación homogénea ${}^pT_0 = {}^0\ T_p^{-1}$, solamente se necesita la traspuesta de la matriz de orientación y multiplicarla por el negativo del vector de posición.

$$T = \begin{bmatrix} s & n & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix} \rightarrow T^{-1} = \begin{bmatrix} s^T & -s^T p \\ n^T & -n^T p \\ a^T & -a^T p \\ 0^T & 1 \end{bmatrix} \tag{5.3}$$

Cuadro 5.1: Dimensiones

Parámetros del Robot		
Parámetro	**Valor**	**Distancia**
H_c	145 mm	distancia vertical al cuello
D_e	34 mm	distancia horizontal del ojo al cuello
H_0		altura del cuello al ojo
δ	10°	ángulo de convergencia del ojo
C_e		profundidad del ojo desde el cuello
H_h	100 mm	altura del hombro
C_p	40 mm	profundidad del hombro
W_c	92 mm	ancho del hombro
L_1	109 mm	longitud del humero
L_2	100 mm	longitud del antebrazo
H_t	118 mm	distancia a la cadera desde el torso
H_p	60 mm	distancia vertical de la cadera a la pierna
W_p	48 mm	distancia horizontal de la cadera a la pierna
L_{p1}	95 mm	longitud del fémur
L_{p2}	115 mm	Longitud de tibia + fíbula

En la figura 5.2, se muestra la colocación de los sistemas de referencia para cada una de las articulaciones del robot. Así mismo; en la tabla 5.3, se muestran los parámetros distintos de cero correspondientes a la variable de acuerdo a la pose del robot mostrado en la figura 5.2.

5.2.1. Modelo Geométrico Inverso para los Brazos

El brazo es una cadena de seis articulaciones y eslabones correspondientes. El problema geométrico inverso es en caso general más desafiante, y es el objeto del siguiente análisis. La ultima articulación (θ_{b6}) es para abrir y cerrar la mano, mientras que la articulación θ_{b5} se utiliza para girar sobre su eje longitudinal, por lo que para posicionar el brazo únicamente se utiliza las cuatro primeras articulaciones, por lo cual es un sistema subactuado; mientras que para la orientación es sub-actuada dado que solo se utiliza la quinta articulación [52] [53] [54].

Cuadro 5.2: Limites de Movimiento

Limites de las articulaciones		
Articulación	**Movimiento**	**Limite**
θ_{h1}	acimut de la cabeza (pan)	(±) 70°
θ_{h2}	elevación de la cabeza (tilt)	(−) 80° a (+)55°
θ_{b1}	extensión vertical de hombro	(−)60°
	flexión vertical de hombro	(+)180°
θ_{b2}	abducción de hombro	(±)180°
	aducción de hombro	(±)45°
θ_{b3}	rotación interna de hombro	(±)45°
	rotación externa de hombro	(±)90°
θ_{b4}	extensión de codo	0°
	flexión de codo (+)	150°
θ_{b5}	supinación de muñeca	(±)180°
	pronación de muñeca	(±)90°
θ_{b6}	mano cerrada	0°
	mano abierta	(±)80°
θ_{t1}	rotación de columna	(±)30°
θ_{t1}	extensión de espalda	(+)30°
	flexión de espalda	(−)75°
θ_{p3}	flexión de pierna	(+)130°
	extensión	(−)30°
θ_{p4}	abducción de pierna	(±)40°
	aducción de pierna	(±)45°
θ_{p5}	rotación de pierna	
θ_{p6}	flexión de rodilla	(+)130°
	extensión de rodilla	(−)15°
θ_{p7}	flexión de pie	(+)45°
	extensión de pie	(−)20°
θ_{p8}	inversión de tobillo	(±)30°
	eversión de tobillo	(±)20°

Cuadro 5.3: Parámetros de Denavit - Hartenberg para el Robot AH1N1

				Cabeza				
j	ρ_j	γ_j	b_j	α_j	d_j	θ_j	r_j	θ_{j0}
1	0					θ_{h1}	H_c	
2	0			$\pi/2$		θ_{h2}		$\pi/2$
Ojo	2		$\pm D_e$	$\pi/2+\delta$	H_0		C_e	
donde $D_e > 0$, $\delta > 0$ para el ojo derecho y $D_e < 0$, $\delta < 0$ para el izquierdo								
				Brazos				
j	ρ_j	γ_j	b_j	α_j	d_j	θ_j	r_j	θ_{j0}
1	0		H_h	$\pi/2$	C_p	θ_{b1}	$\pm W_c$	$\pi/2$
2	0			$-\pi/2$		θ_{b2}		$\pi/2$
3	0			$-\pi/2$		θ_{b3}	L_1	$\pi/2$
4	0			$-\pi/2$		θ_{b4}		$\pi/2$
5	0			$\pi/2$		θ_{b5}	L_2	$\pi/2$
6	0			$\pi/2$		θ_{b6}		$\pi/2$
Mano	2				D_i		R_i	
donde $W_c < 0$ para elbrazo derecho y $W_c > 0$ para el brazo izquierdo								
				Cadera				
j	ρ_j	γ_j	b_j	α_j	d_j	θ_j	r_j	θ_{j0}
1	0		$-H_t$			θ_{t1}		
2	0			$\pi/2$		θ_{t2}		$\pi/2$
				Piernas				
j	ρ_j	γ_j	b_j	α_j	d_j	θ_j	r_j	θ_{j0}
3	0				$-H_p$	θ_{p3}	$\pm W_p$	
4	0			$\pi/2$		θ_{p4}		$\pi/2$
5	0			$-\pi/2$		θ_{p5}	L_{p1}	$\pi/2$
6	0			$-\pi/2$		θ_{p6}		$-\pi/2$
7	0				L_{p2}	θ_{p7}		
8	0			$-\pi/2$		θ_{p8}		
Pie	2				H_f		R_f	
donde $W_p > 0$ para la pierna izquierda y $W_p < 0$ para la pierna derecha								

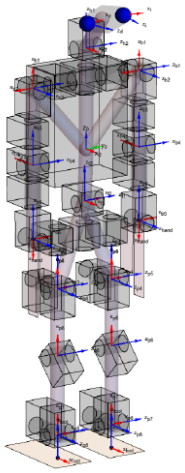


Figura 5.2: Referenciales del robot, en color azul ele eje x, en rojo el eje z (los ángulos de las articulaciones en cero) y los números corresponde a la articulación.

Solución General

Cuando se resuelve el problema de la posición del brazo, se considera un punto en el eje x_{b6} alineado con z_5, por lo cual $R_i = 0$ y $D_i = x > 0$, lo cual implica que rotaciones sobre el eje z_5 no cambia la posición de la mano.

De los parámetros de Denavit-Hartenberg se puede observar que la diferencia entre el brazo izquierdo (lh) y el derecho (rh), es el signo del parámetro W_c. Entonces básicamente se cuenta con las mismas transformaciones homogéneas para ambos brazos, con la diferencia en el vector de traslación sobre el eje y. Por lo tanto se puede escribir $*h$ como la transformación homogénea para ambos brazos.

$$ {}^0T_{*h} = U_0 = \begin{bmatrix} s_d & n_d & a_d & p_d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.4} $$

Para el siguiente análisis solo se considera la parte consistente con la posición deseada. Es decir con la mano cerrada ($\theta_{b6} = 0$). Utilizando el método de Paul [55] a ${}^0T_1$.

$$ T_z(-W_c)R_x(-\pi/2)T_x(-C_p)T_z(-H_h){}^0T_{*h} = T_z(-W_c)R_x(-\pi/2)T_x(-C_p)T_z(-H_h)U_0 \tag{5.5} $$

donde la articulación θ_{b5} no influye en la posición. Calculando la norma de la posición obtenemos la ecuación (5.6).

$$L_1^2 + (L_2 + x)^2 + 2L_1(L_2 + x)\cos(\theta_{b4}) = (C_p - p_x)^2 + (H_h - p_z)^2 + (p_y + W_c)^2 \quad (5.6)$$

Lo cual nos permite calcular la posición de $\theta_{b4} \in [0, \pi]$.

Repitiendo nuevamente el método a la ecuación (5.7) se observa que dos elementos del vector de posición se utilizan para calcular $\theta_{b3} \in (-\pi, \pi]$ que dependen de θ_{b2}, θ_{b1} (5.8).

$${}^2T_{*h} = {}^2T_0 U_0 \quad (5.7)$$

$$\theta_{b3} : \begin{cases} \sin(\theta_{b3}) = \frac{-(p_y+W_c)\cos(\theta_{b2})+((H_h-p_z)\cos(\theta_{b1})+(p_x-C_p)\sin(\theta_{b1}))\sin(\theta_{b2})}{(L_2+x)\sin(\theta_{b4})} \\ \cos(\theta_{b3}) = \frac{(p_x-C_p)\cos(\theta_{b1})+(p_z-H_h)\sin(\theta_{b1})}{(L_2+x)\sin(\theta_{b4})} \end{cases} \quad (5.8)$$

Sin embargo el tercer elemento de la posición es una relación entre θ_{b2}, θ_{b1} y θ_{b4} (5.10),

$$\begin{aligned} L_1 + (L_2 + x)\cos(\theta_{b4}) &= (H_h - p_z)\cos(\theta_{b1})\cos(\theta_{b2}) \\ &\quad -(C_p - H_x + p_x)\cos(\theta_{b1})\sin(\theta_{b2}) \\ &\quad +(p_y + W_c)\sin(\theta_{b2}) \end{aligned} \quad (5.9)$$

la cual puede ser reescrita de dos formas $\theta_{b1} = f(\theta_{b2}, \theta_{b4}) \in (-\pi, pi])$

$$X_2\sin(\theta_{b1}) + Y_2\cos(\theta_{b1}) = Z_2 \quad (5.10)$$

o $\theta_{b2} = f(\theta_{b1}, \theta_{b4}) \in (-\pi, pi])$.

$$X_1\sin(\theta_{b2}) + Y_1\cos(\theta_{b2}) = Z_1 \quad (5.11)$$

lo cual se puede resolver de la siguiente manera.

$$\theta_{bi}: \begin{cases} \sin(\theta_{bi}) = \frac{X_j Z_j + \varepsilon Y_j \sqrt{X_j^2+Y_j^2-Z_j^2}}{X_j^2+Y_j^2} \\ \cos(\theta_{bi}) = \frac{Y_j Z_j + \varepsilon X_j \sqrt{X_j^2+Y_j^2-Z_j^2}}{X_j^2+Y_j^2} \end{cases} \quad i = 1,2 \quad j = 2,1 \tag{5.12}$$

donde $\varepsilon = \pm 1$, además $\sin(\theta_{bi})$ y $\cos(\theta_{bi})$ deben de ser números reales por lo cual se agrega la siguiente restricción.

$$X_j^2 + Y_j^2 - Z_j^2 > 0 \tag{5.13}$$

la cual esta expresada en términos de $\sin(\theta_{bi})$ y $\cos(\theta_{bi})$, utilizando la sustitución de Weierstrass $\sin(\theta_{bi}) = \frac{2x_i}{1+x_i^2}$ y $\cos(\theta_{bi}) = \frac{1-x_i^2}{1+x_i^2}$ con $x_b i = \frac{\theta_{bi}}{2}$ se obtiene un polinomio de cuarto orden para $x_b i$, el cual tiene cuatro raíces reales y la $\theta_b i \in (-pi, pi]$ valida son cuando el polinomio es positivo, es decir $\{r_1, r_2, r_3, r_4\}$ con $r_i < r_{i+1}$ e $i = 1,2,3,4$. Así como los intervalos validos que definen a θ_{bi} son $(-\infty, r_1), (r_2, r_3), (r_4, \infty)$.

Resumiendo para calcular el modelo geométrico inverso primero se determina θ_{b4} de la ecuación (5.6), posteriormente θ_{b1} o θ_{b2} de la ecuación (5.12), considerando la restricción (5.13) y finalmente se calcula θ_{b3} de la ecuación (5.8), además la solución obtenida también debe de satisfacer a la ecuación (5.4).

La dirección de los puntos de la mano están dadas por el vector en el eje x de s_{*h} y θ_{b5} es utilizada para orientar la mano o bien la dirección del vector del eje y (n_{*h}) o la del eje z a_{*h}) se pueden utilizar para definir un plano perpendicular al eje x, donde la orientación del brazo está dada por la ecuación (5.15).

$$s_{*h} = \begin{bmatrix} -\cos(\theta_{b2})\cos(\theta_{b4})\sin(\theta_{b1}) - (\cos(\theta_{b1})\cos(\theta_{b3}) + \sin(\theta_{b1})\sin(\theta_{b2})\sin(\theta_{b3}))\sin(\theta_{b4}) \\ -\cos(\theta_{b3})\sin(\theta_{b1})\sin(\theta_{b4}) + \cos(\theta_{b1})(\cos(\theta_{b2})\cos(\theta_{b4}) + \sin(\theta_{b2})\sin(\theta_{b3})\sin(\theta_b 4)) \\ \cos(\theta_{b4})\sin(\theta_{b2}) - \cos(\theta_{b2})\sin(\theta_{b3})\sin(\theta_{b4}) \end{bmatrix} \tag{5.14}$$

$$\begin{aligned} R_x(\pi/2)R_z(\theta_{b1} - \pi/2)R_x(-\pi/2)R_z(\theta_{b2} - \pi/2)R_x(-\pi/2)R_z(\theta_{b3} - \pi/2) \\ R_x(-\pi/2))R_z(\theta_{b4})R_x(\pi/2)R_z(\theta_{b5} + \pi/2)R_z(\pi/2)R_z(-\pi/2) \end{aligned} \tag{5.15}$$

$${}^0A_{*h} = [s(\theta_{i:1,\ldots 4})n(\theta_{i:1,\ldots 5})a(\theta_{i:1,\ldots 5})] \tag{5.16}$$

El procedimiento anterior es bastante general, y computacionalmente ineficiente. Sin

embargo, nos da una visión de las posibles soluciones asociadas a un punto. Para todas las soluciones los eslabones superiores e inferiores del brazo se encuentran sobre la superficie de dos conos cuyo eje es común y se encuentra sobre la línea que une el hombro con la posición deseada, uno de los conos cuenta con su vértice en la posición deseada mientras que el otro en el hombro. Sin embargo debido a las restricciones físicas del robot no todas las actitudes son alcanzables.

El método anterior es aplicado al punto $(30, -88, -70)$ para el brazo derecho, con lo cual se obtuvieron los siguientes nueve resultados para la condición de que $\theta_4 > 0$, dados como intervalos asociados. La lista de resultados tiene la siguiente estructura $\{\{\theta_{1i}, \theta_{1m}, \theta_{1f}\}, \{\theta_{2i}, \theta_{2m}, \theta_{2f}\}, \{\theta_{3i}, \theta_{3m}, \theta_{3f}\}\{\theta_4\}\}$. Sin embargo este método no es adecuado para su ejecución en tiempo real.

$$\begin{aligned}
&\{\{-Pi, -2.72, -2.30\}, \{2.49, 2.51, Pi\}, \{4.42, 5.11, 6.24\}, \{1.31\} \\
&\{-0.49, 0.17, 0.84\}, \{0, -0.66, 0\}, \{6.24, 4.71, Pi\}\{1.31\}\} \\
&\{2.65, 2.89, Pi\}, \{Pi, 2.61, 2.49\}, \{Pi, 3.99, 4.43\}\{1.31\}\} \\
&\{2.65, 2.73, -2.97\}, \{-Pi, -2.81, 2.48\}, \{Pi, 2.68, 1.57\}\{1.31\}\} \\
&\{-2.30, -2.38, -2.96\}, \{-Pi, -2.81, -2.48\}, \{6.28, 0.45, 1.57\}\{1.31\}\} \\
&\{0.17, -0.49, 0.17\}, \{-0.66, 0, 0.66\}, \{4.71, 6.28, 1.57\}\{1.31\}\} \\
&\{0.17, 0.84, 0.17\}, \{-0.66, 0, 0.66\}, \{4.71, Pi, 1.57\}\{1.31\}\} \\
&\{-2.97, 2.73, 2.65\}, \{2.48, 2.81, Pi\}, \{4.71, 3.59, Pi\}\{1.31\}\} \\
&\{-2.97, -2.38, -2.30\}, \{2.48, 2.81, Pi\}, \{4.71, 5.82, 6.24\}\{1.31\}\}
\end{aligned} \tag{5.17}$$

Solución Clásica

Cuando tenemos una cadena cinemática sobre actuada, la solución usual es reducir el número de gdl de la cadena mediante la fijación de algunas de las articulaciones. Para el brazo tenemos cinco gdl, el cual lo podemos reducir a tres gdl. Fijando, por ejemplo, los eslabones tres y cinco; en este caso, solo podemos especificar su posición. El problema tiene típicamente cuatro soluciones, la figura 5.3 se muestra un ejemplo de las soluciones para el punto de coordenadas $(15, 15, 15)$ en el

referencial de la mano, con $\theta_3 = 0, \theta_5 = 5, \theta_6 = 0$, la posición de la mano izquierda es $(104.178, 219.781, -49.3406)$, mientras que la posición de la mano derecha es $(-148.996, -144.076, -66.4427)$, ambos puntos con respecto al marco de referencia 0.

Las cuatro soluciones para el brazo izquierdo son:

$$\begin{aligned} \{2.87, 2.42, 0, 0.39, 0, 0\}, \\ \{-0.95, 0.58, 0, 0.39, 0, 0\}, \\ \{2.19, 2.42, 0, -0.64, 0, 0\}, \\ \{-0.27, 0.58, 0, -0.64, 0, 0\} \end{aligned} \tag{5.18}$$

Mientras que para el brazo derecho son:

$$\begin{aligned} \{-2.26, -2.96, 0, 0.15, 0, 0\}, \\ \{0.58, -0.31, 0, 0.15, 0, 0\}, \\ \{-2.56, -2.96, 0, -0.39, 0, 0\}, \\ \{0.88, -0.31, 0, -0.39, 0, 0\} \end{aligned} \tag{5.19}$$

Solución en Tiempo Real

Para una aplicación en tiempo real necesitamos un método más simple, el cual se presenta a continuación, para ello mantendremos la ecuación (5.6) para θ_4 y la ecuación (5.8) para θ_3, utilizamos la posición del hombro $\{C_p, W_c, H_h\}$, la posición deseada de la mano $\{p_x, p_y, p_z\}$, la longitud del humero L_1 y la longitud del brazo $L_2 + x$.

La línea que une el hombro y el extremo final de la mano define el eje de dos conos, el ángulo entre la generatriz del cono del brazo superior y la generatriz del cono del antebrazo es θ_4. El vector de aproximación será una generatriz del cono inferior, mientras que la directriz define la posición del brazo superior. Ya que en el hombro, tenemos el equivalente de una articulación esférica, siempre será posible, si el punto

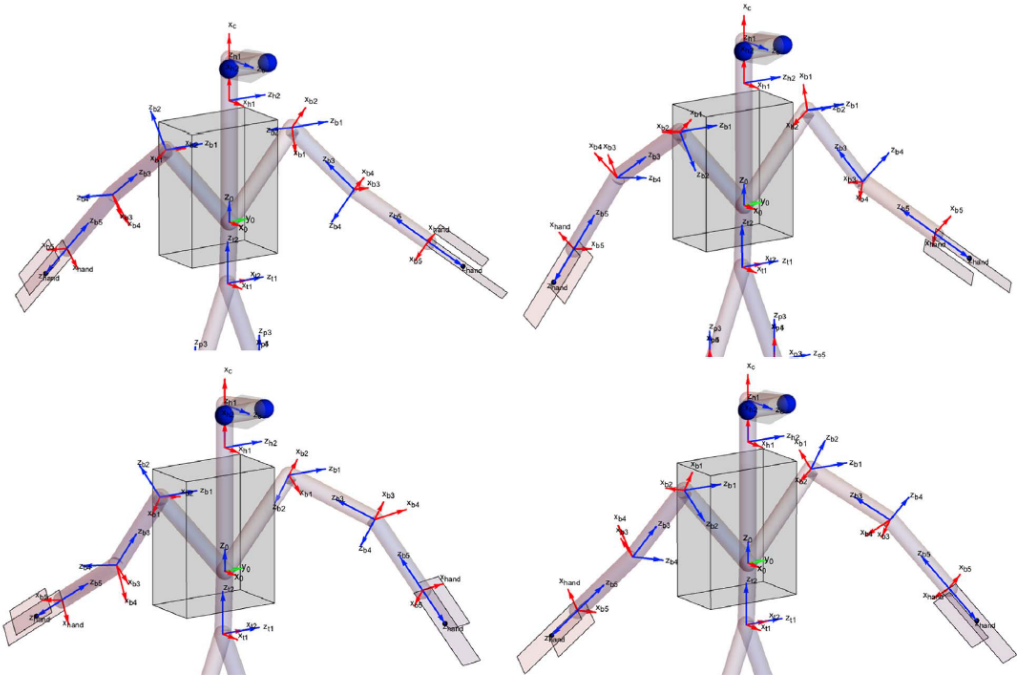


Figura 5.3: Posición del brazo para la posición considerada.

está en el rango de movimiento de la extremidad. En este caso sólo necesitamos calcular θ_1 y θ_2, utilizando la posición del codo como entrada de ${}^0T_3 = U_0^{*h}T_0$. Tenemos que para $\theta_2 \in [-\pi/2, 0]$ para el brazo izquierdo y $\theta_2 \in [0, \pi/2]$ para el derecho.

$$L_1 sin(\theta_2) - W_c = p_y - a(L_2 + x) \tag{5.20}$$

Y para $\theta_1 \in (-\pi, \pi)$

$$\theta_1 : \begin{cases} \sin(\theta_1) = \frac{p_x - a_x(L_2+x) - C_p}{L_1 cos(\theta_2)} \\ \cos(\theta_1) = \frac{H_h + a_z(L2+x) - p_z}{L_1 cos(\theta_2)} \end{cases} \tag{5.21}$$

Este método se puede usar en tiempo real, sin embargo el problema con él es que asume un conocimiento de la dirección de aproximación deseada $a = \{a_x, a_y, a_z\}$ y de la posición final $\{p_x, p_y, p_z\}$. Este último no es un problema, pero el primero si lo es. Además se complica debido a que son mutuamente dependientes, es decir, las actitudes alcanzables de un punto subactuado del brazo dependen de dicho punto.

Por lo tanto, se modifica el método. Considerando que a partir del punto final deseado calcularemos las direcciones de aproximación viables, es decir, la dirección deseada se proyecta en el cono de antebrazo factible. La dirección de aproximación se alineará con el eje x_5. La longitud de la línea que une el hombro y el punto deseado está dado por la ecuación (5.22).

$$D = \sqrt{(p_x - C_p)^2 + (p_y + W_p)^2 + (p_z - H_h)^2} \tag{5.22}$$

Utilizando la ley de senos obtenemos un triángulo formado por el brazo.

$$\frac{\sin(\pi - \theta_4)}{D} = \frac{\sin(\delta)}{L_2 + x} = \frac{\sin(\epsilon)}{L_1} \tag{5.23}$$

donde $\epsilon \in [0, \pi, 2)$ y $\theta_4 = [0, \pi)$ y la altura esta dada por (5.25).

$$\sin(\epsilon) = \frac{L_1}{D} \sin(\pi - \theta_4) \tag{5.24}$$

$$H = (L_2 + x)\cos(\epsilon) \tag{5.25}$$

Desde el punto deseado y su actitud podemos encontrar una aproximación del vector inverso dada por la ecuación (5.26).

$$v_d = -(L_2 + x)\{a_x, a_y, a_z\}^T \tag{5.26}$$

De la línea que une el hombro y el punto deseado, calculamos el vector inverso del eje cónico.

$$v_a = \{C_p, W_c, H_h\}^T - \{p_x, p_y, p_z\}^T \tag{5.27}$$

El ángulo $\delta \in [0, \pi)$ entre ellos puede ser determinado por el producto cruz y punto de los vectores.

$$\sin(\delta) = \frac{\|v_d \times v_a\|}{\|v_d\|\|v_a\|} \cos(\delta) = \frac{v_d \cdot v_a}{\|v_d\|\|v_a\|} \tag{5.28}$$

Para encontrar la dirección de aproximación factible, primero debemos encontrar el punto en la línea $v_d - v_a$ que cruza el cono de aproximación.

$$p_c = \{C_p, W_c, H_h\} - H(v_a - v_d) \tag{5.29}$$

donde

$$H = \frac{D\sin(\epsilon)}{D\sin(\epsilon) + (L_2 + x)\sin(\delta - \epsilon)} \tag{5.30}$$

Por lo tanto el vector de aproximación sera (5.31)

$$a = \{p_x, p_y, p_z\} - P_c \tag{5.31}$$

Utilizamos el vector unitario descrito en la ecuaciones (5.20) y (5.21) para encontrar θ_1 y θ_2y así resolver el problema de cinemática inversa. Para encontrar la actitud de la mano resolvemos el problema de maximización $\max_{\theta_5} s_d \cdot s_f(\theta_5)$, donde s_d es la dirección para el eje $+x$ y s_f es la dirección factible para $+x$ debido a que $s_d \cdot s_f(\theta_5) = a\sin(\theta_5) + b\cos(\theta_5)$, con lo cual se puede emplear la derivada para encontrar θ_5 para la dirección más cercana a la factible. El mismo procedimiento se aplica en el caso de que el eje deseado sea y, en dicho caso se utiliza n_d y $n_f(\theta_5)$.

Este enfoque necesita como entradas las coordenadas del punto deseado y la dirección de aproximación deseada, es decir, requiere p_0 , s_0 y n_0. Por ejemplo, para una aproximación al punto $\{50, 0, -30\}$ y la dirección de aproximación dada por la ecuación (5.32), se puede calcular el vector de posición $\{\theta_{b1}, \theta_{b2}, \theta_{b3}, \theta_{b4}, \theta_{b5}, \theta_{b6}\}$, donde para el brazo izquierdo queda como $\{-0.90, -0.29, 0.35, 1.91, -2.74, 0\}$, mientras que para el brazo derecho $\{0.90, 0.29, -0.35, 1.91, -0.39, 0\}$ y las matrices de transformación homogéneas están dadas por (5.33).

$$L_{left} = \begin{bmatrix} 0 & 1000 & 1 & 50 \\ 1 & 1000 & 0 & 25 \\ 0 & 1000 & 0 & -30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$
$$L_{right} = \begin{bmatrix} 0 & 1000 & 1 & 50 \\ -1 & 1000 & 0 & -25 \\ 0 & 1000 & 0 & -30 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5.32}$$

$$L_{left} = \begin{bmatrix} 0.191565 & 0.454377 & 0.869968 & 50 \\ 0.976604 & -1.11022 \times 10^{-16} & -0.215046 & 25 \\ -0.0977121 & 0.89081 & -0.443746 & -30 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$L_{right} = \begin{bmatrix} 0.191565 & -0.454377 & 0.869968 & 50 \\ -0.976604 & -5.55112 \times 10^{-17} & 0.215046 & -25 \\ -0.0977121 & -0.89081 & -0.4437460 & 0 \quad 0 \quad 1 \end{bmatrix} \tag{5.33}$$

Se puede ver que se alcanza el punto y la orientación está próxima a la deseada, es decir, el vector s está próximo al vector s_d de la orientación deseada. Una vista frontal de la configuración final se muestra en la figura 5.4.

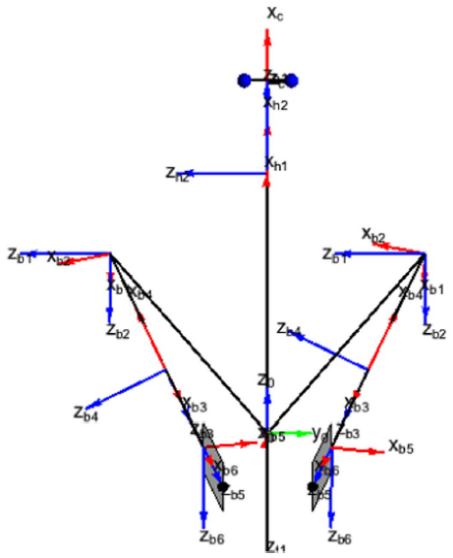


Figura 5.4: Solución geométrica inversa para el brazo.

La posición de la mano con respecto al ojo se muestra en la ecuación (5.34), donde esto se puede generalizar, de modo que el robot pueda posicionar un punto particular de su mano en su línea de visión.

$$^{eye}T_{hand} = ^{eye}\ T_0(\theta_{h_1}, \theta_{h_2})\,^0T_{hand}(\theta_{b_1}, ..., \theta_{b_5}) \tag{5.34}$$

5.2.2. Solución Geométrica Inversa para las Piernas

Las piernas cuentan con seis gdl, ya que la cadera es una articulación esférica, donde se utiliza la cinemática inversa de los eslabones para resolver el problema inverso. En este caso se obtienen ocho soluciones. Para ejemplificar el procedimiento se muestran las soluciones para un punto con coordenadas $\{10, 10, 10\}$ en el marco de referencia de cada pie. La posición de la pierna izquierda en el referencial del torso es $\{74.5894, 55, -418.589\}$, mientras que para la pierna derecha es $\{-62.5894, -35, -418.589\}$, de donde las soluciones obtenidas para cada pierna se presentan a continuación, y son mostradas en la figura 5.5.

Soluciones Pierna Izquierda:

$$\{0, -3.14159, 3.07128, 0.785398, 2.4265, 3.14159\},$$
$$\{0, 0, -0.785398, 0.785398, -9.47155 \times 10^{-17}, 0\},$$
$$\{3.14159, 3.14159, 2.35619, 0.785398, -9.47155 \times 10^{-17}, 0\},$$
$$\{3.14159, -1.22465 \times 10^{-16}, 0.785398, -0.785398, -3.14159, 3.14159\},$$
$$\{0, -3.14159, -2.35619, -0.785398, -3.14159, 3.14159\},$$
$$\{0, 0, 0.0703092, -0.785398, 0.715089, 0\},$$
$$\{3.14159, 3.14159, -3.07128, -0.785398, 0.715089, 0\}$$

Soluciones para la pierna derecha:

$$\{3.14159, -1.22465 \times 10^{-16}, -0.785398, 0.785398, 3.14159, 3.14159\},$$
$$\{0, -3.14159, 2.35619, 0.785398, 3.14159, 3.14159\},$$
$$\{0, 0, -0.0703092, 0.785398, -0.715089, 0\},$$
$$\{3.14159, 3.14159, 3.07128, 0.785398, -0.715089, 0\},$$
$$\{3.14159, -1.22465 \times 10^{-16}, 0.0703092, -0.785398, -2.4265, 3.14159\},$$
$$\{0, -3.14159, -3.07128, -0.785398, -2.4265, 3.14159\},$$
$$\{0, 0, 0.785398, -0.785398, 9.47155 \times 10^{-17}, 0\},$$
$$\{3.14159, 3.14159, -2.35619, -0.785398, 9.47155 \times 10^{-17}, 0\}$$

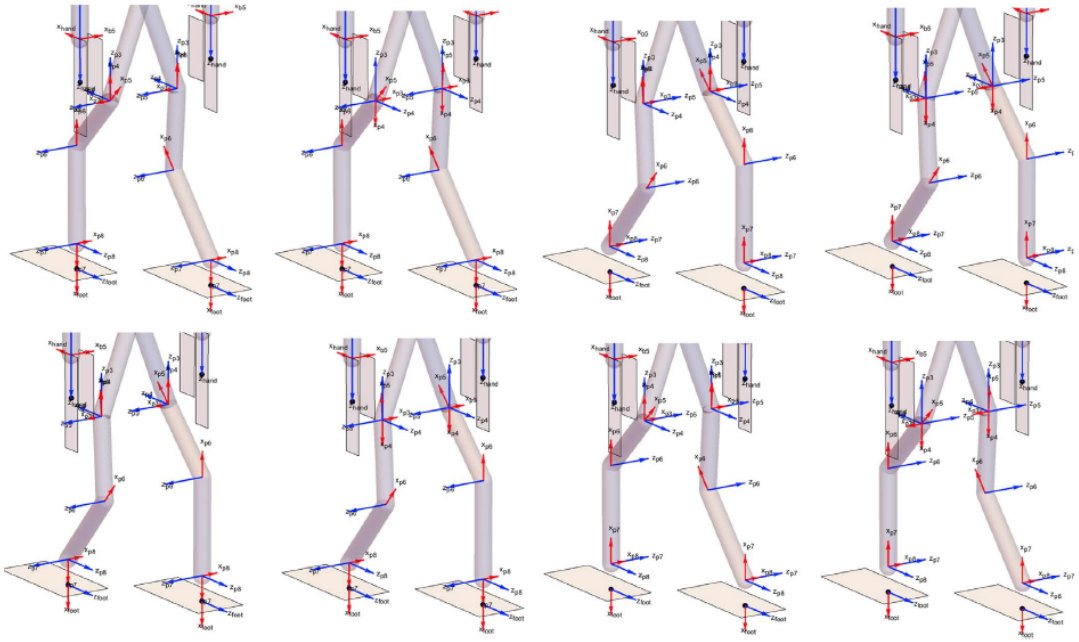


Figura 5.5: Ocho solución para el problema de geométrica inversa de las piernas.

5.3. Modelo Cinemático

El modelo cinemático relaciona las velocidades en el espacio articular con las velocidades en el espacio de trabajo. Se utiliza el Jacobiano básico donde las velocidades en el espacio de trabajo se escriben como tornillos de velocidad, es decir, giros.

5.3.1. Modelo Cinemático Directo del Brazo

El Jacobiano $J(q)$ del brazo se puede escribir de diferentes formas, todas equivalentes, es decir, contiene la misma información. El Jacobiano básico expresa la velocidad en el espacio de trabajo como un tornillo de velocidad $v = \{v_x, v_y, v_z, \omega_x, \omega_y, \omega_z\}^T$, es decir, un vector cuyos elementos son la velocidad lineal y angular de un punto, a partir de las velocidades en el espacio de articulación $\dot{q} = \{\dot{q_1}, \dot{q_2}, ..., \dot{q_n}\}$.

$$\begin{bmatrix} v_x \\ v_y \\ v_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = J(q) \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \\ \dot{q_3} \\ ... \\ \dot{q_n} \end{bmatrix} \tag{5.35}$$

Dado que el brazo utiliza sólo cinco articulaciones para posicionar y orientar el efector final, su Jacobiano no será de rango completo. Su determinante nos dará las singularidades, a las cuales hay que añadir las restricciones sobre los rangos de las articulaciones y sus velocidades. Una expresión simple puede ser determinada si la velocidad total (producida por las cinco articulaciones) se calcula en un punto en la segunda articulación y es proyectada en el marco de referencia de la tercera articulación, en este caso, el Jacobiano está dado por la ecuación (5.36).

$${}^3J_{5,2} = \begin{bmatrix} 0 & 0 & 0 & -L_1 & 0 \\ 0 & 0 & 0 & 0 & L_! \sin(\theta_4) \\ \cos(\theta_2)\sin(\theta_3) & \cos(\theta_3) & 0 & 0 & \sin(\theta_4) \\ \cos(\theta_2)\sin(\theta_3) & -\sin(\theta_3) & 0 & 1 & 0 \\ \sin(\theta_2) & 0 & 1 & 0 & \cos(\theta_4)) \end{bmatrix} \tag{5.36}$$

Este Jacobiano puede usarse para calcular la velocidad en cualquier otro punto o proyectarse a cualquier otro marco de referencia. Su exactitud se puede ver si se analiza la figura 5.1, de la cual se pueden determinar, cualquier otra versión del Jacobiano a partir de la ecuación (5.2).

$${}^sJ_x = {}^s\ A_i^i T_{j,x}^i J_{n,j} = \begin{bmatrix} {}^sA_i & 0 \\ 0 & {}^sA_i \end{bmatrix} \begin{bmatrix} I & -{}^iL_{j,x} \\ 0 & I \end{bmatrix}^i J_{n,j} \tag{5.37}$$

donde la primera matriz nos da la orientación de un tornillo en el marco de referencia s desde un tornillo expresado en el marco i, utilizando la matriz de orientación relativa ${}^sA_i$, la segunda calcula el tornillo en un punto x cuya distancia desde el punto j está dada por la matriz antisimétrica del vector $iL_{j,x}$, (utilizado para calcular el producto cruzado $\omega \times L_{j,x}$).

Si estamos interesados en controlar un punto en la mano p_h; su movimiento podemos expresarlo en el referencial base ${}^0J_h$ o en el en el referencial de la mano ${}^5J_h$. Por lo tanto, para calcular la ecuación (5.37) necesitamos la distancia hasta el punto de donde queremos calcular la velocidad, en la ecuación (5.36) entre la segunda articulación y un punto en la mano h será $L2, h$ proyectada en la tercera articulación ${}^3L_{2,h} = {}^3\,p_h - {}^3\,p_2$, considerando que la distancia d es del origen del eje z a la quinta articulación, donde:

$$ {}^3p_2 = \begin{bmatrix} 0 \\ 0 \\ -L_1 \end{bmatrix} \tag{5.38} $$

$$ {}^3p_h = \begin{bmatrix} (L_2 + d)\sin(\theta_4) \\ 0 \\ (L_2 + d)\cos(\theta_4) \end{bmatrix} \tag{5.39} $$

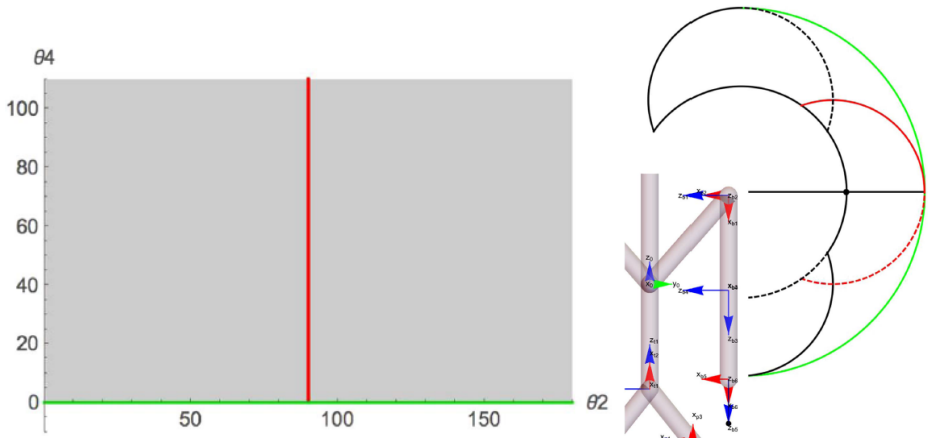
5.3.2. Singularidades en el Espacio de Trabajo para el Brazo

De la ecuación (5.37), se puede ver que el determinante del Jacobiano no cambia si se modifica la orientación o el punto considerado. También se observa que el Jacobiano no es de rango completo, por lo que el determinante menor de rango completo, resulta de la eliminación de la tercera fila, quedando $-L_1^2\cos(\theta_2)\sin(\theta_4)$, donde los ceros del determinante dan los puntos en el espacio articular en donde el brazo pierde grados de libertad en el espacio de trabajo, estos definen superficies en el espacio articular y en este caso tenemos las siguientes singularidades:

- Cuando el brazo está horizontal, apuntando hacia fuera y es colineal con la articulación del hombro $\theta_2 = \pm\pi/2$, donde el signo más se aplica al brazo izquierdo y el menos al brazo derecho y ocurre cuando las articulaciones 1 y 3 son colineales, por lo que la matriz Jacobiana pierde rango.

- Cuando el codo está completamente extendido o flexionado $\theta_4 = 0$, para ambos brazos, ya que el brazo no puede alcanzar la otra singularidad $\theta_4 = \pi$, en este caso el brazo está tocando el límite exterior del área de trabajo y las articulaciones 3 y 5 son colineales.

Dado que sólo dos articulaciones tienen singularidades podemos presentarlas en un plano, en la figura 5.6 a) se presenta las singularidades en el espacio de la articulación y en la figura 5.6 b) en el espacio de trabajo.



(a) (b)

Figura 5.6: a) El espacio de la articulación (en gris) y la singularidad (en color rojo y verde), incluso si el θ_4 está en el borde y por lo tanto no tiene impacto. b) Vista coronal del espacio de trabajo del brazo, las líneas rojas y verdes son el mapeo de las singularidades en el espacio de trabajo donde se llaman Superficies Jacobianas. La superficie roja superior y la inferior son $\theta_1 + \theta_3 = \pi$ y $\theta_5 = \pm\pi/2$.

Modelo Cinemático Inverso del Brazo

Con el fin de calcular el modelo cinemático inverso, es decir, el Jacobiano inverso, utilizamos la inversa de (5.37), tal como se muestra en la ecuación(5.40).

$$ {}^{s}J_{n}^{-1} = {}^{i}J_{n,j}^{-1}\,{}^{i}T_{j,x}^{-1}\,{}^{s}A_{i}^{-1} = {}^{i}J_{n,j}^{-1} \begin{bmatrix} I & -{}^{i}L_{j,x} \\ 0 & I \end{bmatrix} \begin{bmatrix} {}^{s}A_{i}^{T} & 0 \\ 0 & {}^{s}A_{i}^{T} \end{bmatrix} \tag{5.40} $$

donde se utiliza la pseudo inversa para ${}^{i}J_{n,j}^{-1}$, por ejemplo para ${}^{3}J_{5,2}^{-1}$ con $\theta_2 \neq \pi/2$ y $\theta_4 \neq 0$ tenemos:

$$ {}^{3}J^{+}_{5,2} = \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_4 \\ J_5 \end{bmatrix} \tag{5.41} $$

$$ J_1 = \begin{bmatrix} \cos(\theta_3)\sec(\theta_2)/L_1 \\ -\sin(\theta_3)\sec(\theta_2)/L_1 \\ 0 \\ \sin(\theta_3)\sec(\theta_2) \\ \cos(\theta_3)\sec(\theta_2) \\ 0 \end{bmatrix} \quad J_2 = \begin{bmatrix} \sin(\theta_3)/L_1 \\ \cos(\theta_3)/L_1 \\ 0 \\ \cos(\theta_3) \\ \sin(\theta_3) \\ 0 \end{bmatrix} $$

$$ J_3 = \begin{bmatrix} \cos(\theta_3)\tan(\theta_2)/L_1 \\ (\sin(\theta_3)\tan(\theta_2) - \cot(\theta_4))/L_1 \\ 0 \\ -\sin(\theta_3)\tan(\theta_2) \\ -\cos(\theta_3)\tan(\theta_2) \\ 1 \end{bmatrix} \quad J_4 = \begin{bmatrix} -1/L_1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad J_5 = \begin{bmatrix} 0 \\ \csc(\theta_4)/L_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} $$

5.3.3. Modelo Cinemático de la Pierna

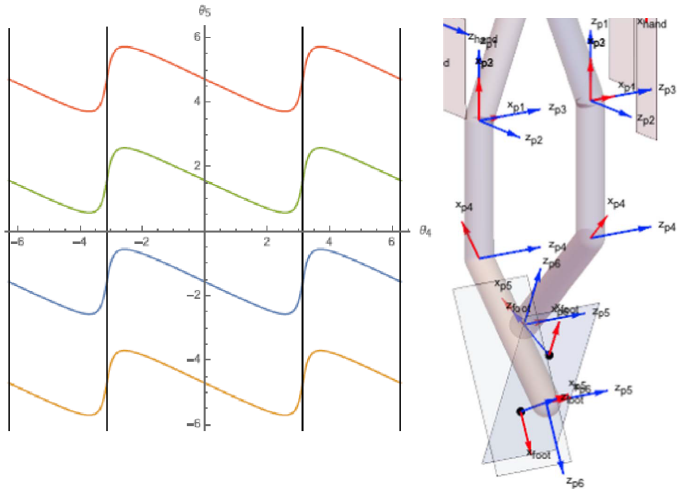
El procedimiento utilizado para calcular el modelo cinemático para los brazos se utilizó de forma similar para calcular el Jacobiano de las piernas. El principal resultado del análisis del modelo cinemático son las singularidades en su área de trabajo, es decir, donde el modelo cinemático pierde rango, dado por los ceros del determinante. En este caso el determinante es mostrada en la ecuación (5.42), en donde encontramos tres singularidades:

$$ L_{p_1} L_{p_2} \cos(\theta_2)\sin(\theta_4)[L_{p_2}\cos(\theta_5) + L_{p_1}\cos(\theta_4 + \theta_5)] \tag{5.42} $$

- $\cos(\theta_2) = \pm\pi/2$, la pierna es horizontal y los ejes 1 y 3 están alineados (sin rotación en el eje de la pierna).

- $\sin(\theta_4) = 0 \pm \pi$, la pierna está completamente extendida o flexionada (el extremo de la pierna ha alcanzado el límite del espacio de trabajo).
- $L_{p_2}\cos(\theta_5) + L_{p_1}\cos(\theta_4 + \theta_5) = 0$ en este caso es es cero donde el eje de la articulación de pronación-supinación del tobillo intersecta el origen de la cadera esférica, es decir, se vuelve redundante.

En la figura 5.7 se muestra el espacio articular, donde la segunda y tercera singularidad dividen el espacio articular, mientras que en la segunda imagen se presenta una posición donde está presente la tercera posición singular.



(a) (b)

Figura 5.7: a) Relación entre θ_4 y θ_5 para las segundas y terceras singularidades de las piernas en el espacio articular. b) Un ejemplo de la tercera posición singular en el espacio de trabajo.

Capítulo 6

Control del Comportamiento de Robots Humanoides Futbolistas

6.1. Introducción

6.1.1. Robocup

RoboCup es una competencia de talla internacional donde se congregan institutos de investigación y universidades para presentar y compartir los resultados de sus investigaciones en las áreas de inteligencia artificial, Visión Artificial y robótica. En este evento equipos de robots compiten en partidos de fútbol. La primera de estas competiciones se llevó a cabo en 1997, desde entonces los retos se han modificado constantemente, el objetivo principal de la RoboCup es ahora, ya bien conocido:

Para la mitad del siglo 21, un equipo de robots humanoides completamente autónomos deben ganar un partido de fútbol contra el equipo FIFA que sea campeón mundial de la copa más reciente.

6.1.2. ¿Qué Necesita un Robot Humanoide para Jugar Fútbol?

Los robots utilizados en las competiciones de fútbol deben de reunir al menos el siguiente conjunto de características que los hace susceptibles de convertirse en futbolistas:

- *Arquitectura mecánica:* Una arquitectura mecánica semejante a la de los seres humanos formada por dos brazos, dos piernas, un torso y una cabeza.
- *Sistema computacional:* Generalmente formado por un sistema embebido encargado del control del movimiento del robot, de los servomotores que actúan cada articulación y de la gestión de los sensores internos.
- *Sistema de percepción:* Este sistema es el que se encarga de identificar el entorno, la situación de juego actual y parte del estado interno del humanoide (caídas, etc.).

Así, estos robots son capaces de ejecutar movimientos autónomos como el caminar (hacia adelante, hacia atrás o lateralmente), dar giros, levantarse y aún patear una pelota. Sin embargo, para el fútbol con un mínimo de eficiencia es necesario incorporarles funcionalidades específicas como las siguientes:

- *Detectar y localizar la pelota, las porterías y a los demás jugadores.* En este contexto, localizar significa calcular la posición de dichos objetos en el referencial asociado al robot, es decir la posición relativa de tales objetos con respecto al robot.
- *Autolocalización.* Es la capacidad de calcular su posición (x,y) y orientación (θ) dentro del terreno de juego.
- *Estrategia de juego.* Con base a la información anterior (posición de la pelota, de la portería, de los demás jugadores y de la suya) decidir que jugada debe realizar en función de su rol dentro del terreno de juego (portero o jugador de campo).

6.2. Control del Comportamiento de un Portero

En esta Sección se hablará de los requerimientos mínimos para que un robot humanoide actué como un portero, es decir las función principal del portero es evitar que anoten goles en su portería, para ello se ha planteado un algoritmo para realizar el control de su comportamiento.

En este caso se utiliza el robot humanoide BIOLOID descrito en la Sección A.1, a groso modo el robot debe detectar a la pelota, e identificar si esta va en dirección a su portería, y tomar alguna acción en respuesta a la posición de la pelota.

Tal como se mencionó en la Sección anterior, el portero del equipo es un robot humanoide Bioloid Premium al cual se le ha agregado el sistema de visión Havimo 2.0, una cabeza con los movimientos de elevación y acimutal (2 gdl) que permiten orientar la cámara de dicho sistema, así como una programación muy simple pero eficiente que le permite realizar las funciones mínimas de un portero de acuerdo con la normas RoboCup vigentes [1]. Dichas funciones son:

Percepción

- Detección y localización de la pelota,
- Estimación de la trayectoria de la pelota en movimiento,
- Localización en el terreno de juego.

Movimientos

- Movimientos de cabeza para búsqueda de pelota (paneo),
- Centrado de Pelota,
- Caminados (de frente, hacia atrás y laterales),
- Atajadas,
- Levantamientos,
- Despejes de portería.

6.2.1. Detección, Localización y Centrado de Pelota

Paneo

El Bioloid portero inicia el juego colocado en su sitio: al centro de su portería, por lo que su programa inicia con una exploración del terreno de juego buscando la pelota. Esto se hace mediante un movimiento coordinado de los grados de libertad

del cuello (acimut y elevación) mediante el cual seguirá una trayectoria parecida el número ocho, se iniciará de la parte superior derecha hasta la parte inferior izquierda trazando una trayectoria parecida al número 5, posteriormente se eleva a la mitad el acimut y se realiza un barrido hacia la derecha donde al llegar a su última posición sufre otra elevación y con ello regresa al punto de partida, como se muestra en la figura 6.1. Esta acción se realiza para tratar de encontrar el balón de una forma más rápida.

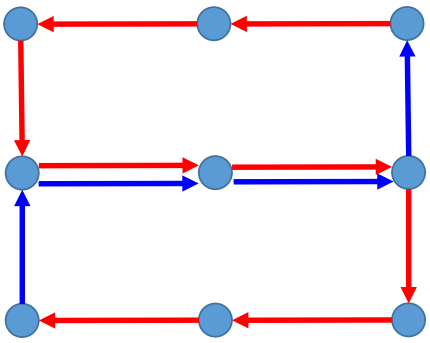


Figura 6.1: Paneo realizado por el robot Bioloid.

Generalmente la exploración acimutal se realiza mediante un "paneo" entre $-60°$ y $+60°$ recorrido en el que se toman n fotos para cada uno de los tres valores del ángulo de elevación (12°, 37°, 62°) determinados para barrer todo el espacio de trabajo de interés, en cada una de estas posiciones se toma una imagen.

Detección de la pelota

La pelota era de color naranja y un tamaño de 16 cm de diámetro de acuerdo a las reglas vigentes en la robocup del año 2014, para detectar esta pelota se procede a detectar el objeto de color naranja más grande dentro de la LUT del sistema de visión Havimo, tal como se muestra en la figura 6.2.

Mientras que para los años posteriores las reglas cambiaron y la pelota se cambió por un balón brazuca (oficial de la FIFA) de tamaño No. 1, en este caso ya que en su mayoría es de color blanco y con vivos en diferentes colores se siguió el siguiente procedimiento, como se observa en la figura 6.3.

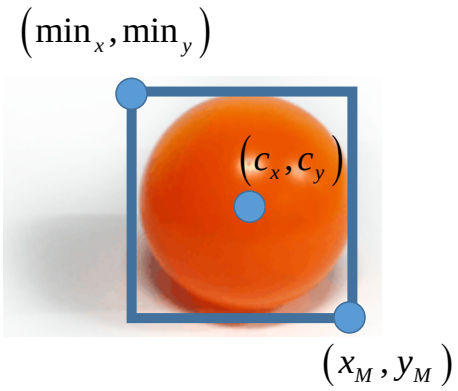


Figura 6.2: Detección de pelota naranja.

- Identificar todos los objetos blancos,
- Despreciar los objetos blancos grandes (líneas del campo),
- Agrupar los objetos blancos cercanos,
- Identificar los objetos de un color de la pelota (azul) cuyo centroide se encuentre dentro del rectángulo de encuadre generado por el paso anterior,
- Si existe algún objeto azul que cumpla el paso anterior se considera como pelota al objeto blanco.

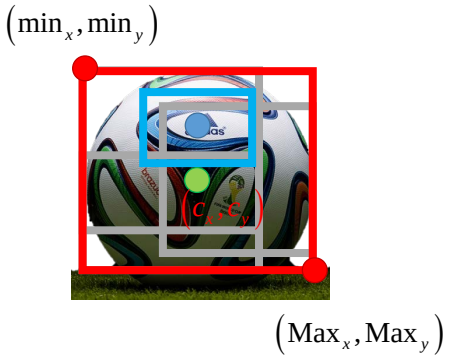


Figura 6.3: Detección de balón brazuca.

Una vez detectada la pelota se procede a calcular su posición dentro de la imagen (coordenadas imagen de su baricentro) mediante las ecuaciones (6.1) y (6.2).

$$c_x = \frac{\mathrm{Max_x} + \mathrm{min_x}}{2} - 80 \tag{6.1}$$

$$c_y = \frac{\mathrm{Max_y} + \mathrm{min_y}}{2} - 60 \tag{6.2}$$

En donde $\mathrm{Max_x}$, $\mathrm{min_x}$, $\mathrm{Max_y}$, $\mathrm{min_y}$ son las coordenadas del rectángulo de encuadre de la pelota o AABB (Axis Aligned Bounding Box) que contiene a la mancha naranja detectada, mientras que las coordenadas en píxeles del punto principal de la imagen son (80,60).

Una vez calculado el baricentro del balón se accionan los motores del cuello, variables angulares de elevación y acimut, a fin de que la pelota quede al centro de la imagen (centrado) y así poder calcular la posición de la pelota en el referencial asociado al robot. Las figuras 6.4 y 6.5 muestran los diagramas cinemáticos en los planos vertical y horizontal respectivamente, a partir de los cuales se calcula la posición relativa de la pelota con respecto al robot mediante las ecuaciones (6.3), (6.4) y (6.5).

$$d = \frac{h}{\tan(\phi)} \tag{6.3}$$

$$x = d\cos(\theta) \tag{6.4}$$

$$y = d\sin(\theta) \tag{6.5}$$

En donde h es la altura del centro óptico de la cámara cuyo valor es obtenido mediante calibración de sus parámetros extrínsecos, ϕ es el ángulo de elevación, d es la distancia entre la pelota y el robot, mientras que θ es el ángulo de acimut.

Como se observa en la Figura 6.4 este método introduce un pequeño error de posición en el cálculo de la distancia d que es proporcional al radio de la pelota; pero que no produce un efecto importante en el desempeño del robot.

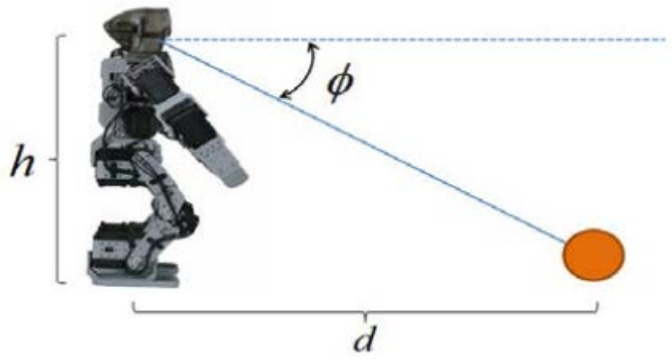


Figura 6.4: Diagrama cinemático en el plano vertical.

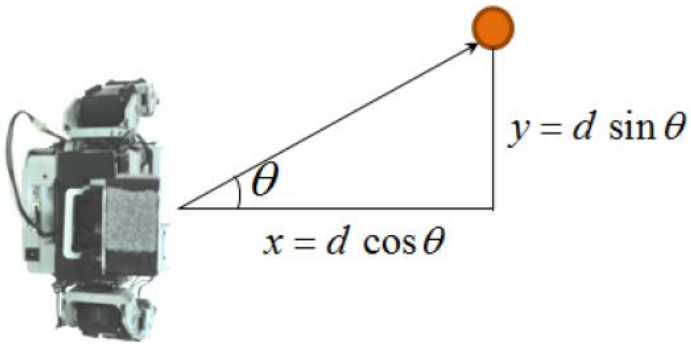


Figura 6.5: Diagrama cinemático en el plano horizontal.

Determinación de las Acciones de Atajado de la Pelota

Al portero robótico le interesa más la trayectoria de un pelota en movimiento que la ubicación estática de la misma, claro, excepto cuando debe realizar algún despeje de la pelota. Para ello se implementa un programa de tracking (centrado dinámico) que mantiene a la pelota siempre en el centro de la imagen, independientemente de su movimiento.

Entonces, una vez detectada la pelota y centrada por primera vez se calcula su posición relativa (x_1,y_1) usando las ecuaciones (6.4) y (6.5). Si la pelota está en movimiento, en la siguiente vez que se calcule su posición estará en el punto (x_2,y_2) calculado con las mismas ecuaciones y, además, el ángulo de elevación ϕ disminuye, es decir que la pelota se acerca a la portería y el robot debe atajarla. Si el ángulo de elevación aumenta, no es necesario hacer nada, sólo continuar con el tracking. La

Figura 3 muestra el diagrama de esta situación, en donde el robot se encuentra en el punto (0,0).

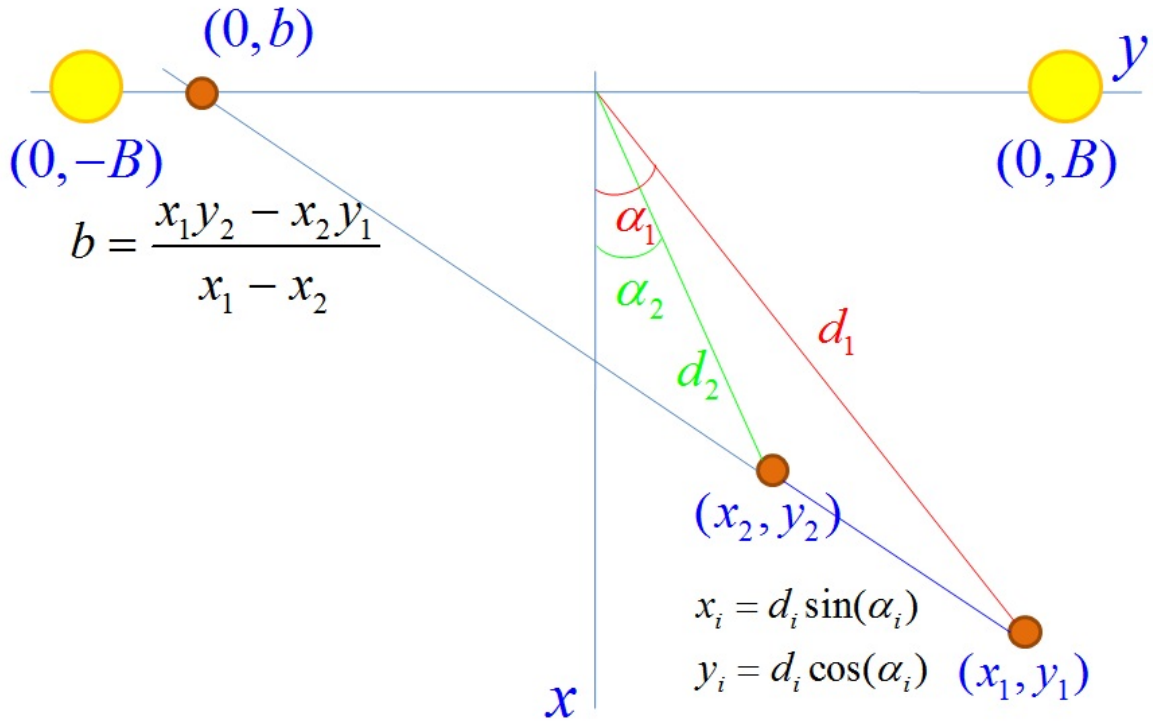


Figura 6.6: Detección de trayectoria.

Entonces, la pelota va a la portería si se dirige a cruzar la línea $x = 0$ en el punto $(0, b)$ con $-B \leq b \leq B$, en donde la portería tiene un ancho de $2B$ y el portero se encuentra justo al centro de la portería. La coordenada b se calcula de acuerdo a la ecuación (6.6).

$$b = \frac{x_1 y_2 - x_2 y_1}{x_1 - x_2} \tag{6.6}$$

Cabe destacar que, después de cada atajada, el robot debe levantarse y continuar con su trabajo de detección, localización, centrado de la pelota y calculando las acciones de atajado que sean necesarias ante la detección de un disparo de los robots contrarios.

6.2.2. Localización en el Terreno de Juego

Para determinar la posición del portero dentro del terreno de juego se procede a dos etapas:

- Determinar la Orientación,
- Búsqueda de Postes.

Orientación

Para que el robot sea capaz de orientarse dentro del terreno de juego es necesario que cuente con una brújula digital, es decir, con un sistema de medición del campo magnético. Agregarle la funcionalidad de interpretar la señal de un magnetómetro al sistema de control del Bioloid, no es sencillo ni recomendable pues el programa correspondiente interferiría con sus tareas de control. Por ello se optó por agregar un microcontrolador Arduino para realizar esta tarea e implementar un protocolo de comunicación entre ambos microcontroladores.

Para que el robot sea capaz de orientarse dentro del terreno de juego es necesario que cuente con una brújula digital, es decir, con un sistema de medición del campo magnético.

La comunicación de los sensores es por medio del protocolo de comunicación I2C, del cual se carece en el controlador CM-510 del Bioloid, el cerebro del robot únicamente cuenta con entradas analógicas y un puerto serie para programación, por ello se desarrolló una interface entre un sensor I2C y la entrada analógica del controlador, utilizando un micro controlador (Arduino NANO) [56], el cual realiza la lectura de ambos sensores a una frecuencia de 100 Hz $[x_m y_m z_m]$, en el micro controlador se realiza una etapa de filtrado para eliminación de ruido atreves de un filtro promedio, así como la calibración del mismo, posteriormente las tres coordenadas del campo magnético son codificadas en coordenadas esféricas debido a que el radio es constante y este se puede eliminar, para considerar únicamente los ángulos en la transmisión de información, la cual es enviada por los puerto PWM del micro controlador, debido a que esta sigue siendo una señal digital se pasa por un filtro pasa-bajas con una frecuencia de corte de 10Hz, para convertirla en una señal analógica y esta pueda ser interpretada por la CM-510.

Calibración del magnetómetro. Esta etapa se realiza fuera de línea, y con los motores del robot encendidos, en el cual se toman medidas rotando el robot y midiendo la dirección del polo norte magnético, con ello se calcula un vector de desplazamiento del centro del sensor $[t_x t_y t_z]$ y un vector de escala $[s_x s_y s_z]$, los cuales son los parámetros de calibración, quedando como en la ecuación (6.7). Mostrado en la figura 6.7.

$$\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \begin{bmatrix} (x_m + t_x)\, s_x \\ (y_m + t_y)\, s_y \\ (z_m + t_z)\, s_z \end{bmatrix} \tag{6.7}$$

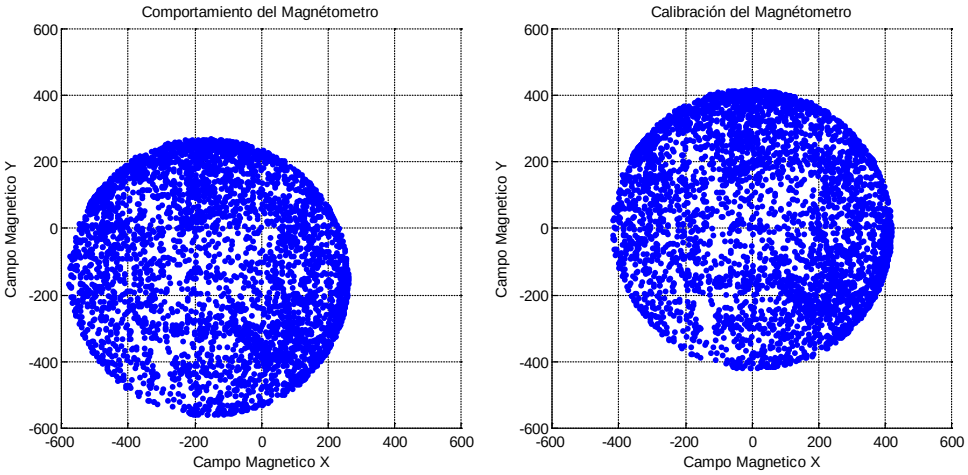


Figura 6.7: Calibración del magnetómetro.

Filtro Promedio. Para tener una mejor estimación de valor de campo magnético se utiliza un filtro promedio, en el cual se toman las ultimas 10 muestras y se promedian de acuerdo a la ecuación (6.8).

$$x_s = \frac{1}{10}\sum_{i=1}^{10} x_{c,i} \;\; y_s = \frac{1}{10}\sum_{i=1}^{10} y_{c,i} \;\; z_s = \frac{1}{10}\sum_{i=1}^{10} z_{c,i} \tag{6.8}$$

Conversión a coordenadas esféricas. Debido a que el parámetro que estamos midiendo es la dirección del polo Norte magnético, las transformamos a coordenadas esféricas y únicamente nos interesan los ángulos de latitud y azimut, para ello

utilizamos las expresiones de la ecuación (6.9).

$$\theta = \begin{cases} \operatorname{atan}\left(\frac{\sqrt{x_s^2+y_s^2}}{z_s}\right) & z_s > 0 \\ \frac{\pi}{2} & z_s = 0 \\ \pi + \operatorname{atan}\left(\frac{\sqrt{x_s^2+y_s^2}}{z_s}\right) & z_s < 0 \end{cases} \quad \varphi = \begin{cases} \operatorname{atan}\left({}^{y_s}/_{x_s}\right) & x_s > 0 \text{ y } y_s > 0 \\ 2\pi + \operatorname{atan}\left({}^{y_s}/_{x_s}\right) & x_s > 0 \text{ y } y_s < 0 \\ \frac{\pi}{2}\operatorname{sign}(y_s) & x_s = 0 \\ \pi + \operatorname{atan}\left({}^{y_s}/_{x_s}\right) & x_s < 0 \end{cases} \tag{6.9}$$

Filtro Pasa- Bajas. Se diseñó un filtro Pasa-Bajas RC, con una frecuencia de corte a 10Hz, para convertir la señal digital en una señal analógica de acuerdo al siguiente diagrama.

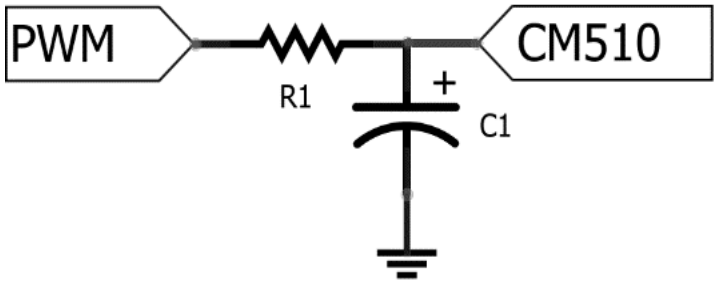


Figura 6.8: Filtro Pasa-Bajas.

Posición

Para determinar la posición del robot $[X_c, Y_c]$ dentro del terreno de juego se realiza la búsqueda de los postes que son de color blanco o amarillo, para ello se calcula la distancia relativa del robot con cada uno de los postes propios (R_1, R_2) , como se muestra en la figura 6.9.

Esta distancia es calculada con la ecuación (6.3). En las ecuaciones (6.10) y (6.11) se muestran las fórmulas para determinar su posición. Sin embargo, para determinar el valor de Y_c se debe de tomar en cuenta el valor de Y_p, si el marco de referencia del humanoide está por delante de la línea de meta Y_p será negativo, y si esta por detrás este será positivo, por lo tanto cuando Y_p sea negativo Y_c será el valor positivo. En la figura 6.10 se presenta el diagrama de estimación de posición del robot humanoide.

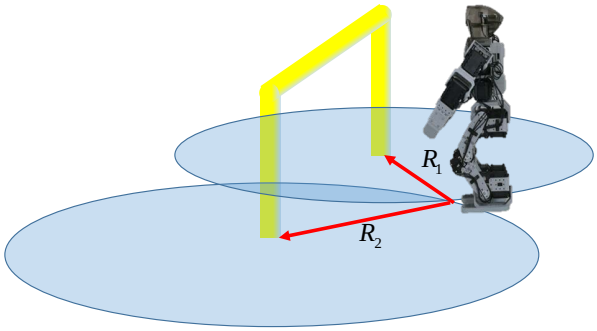


Figura 6.9: Detección de poste.

$$x_c = \frac{R_1^2 - R_2^2}{4x} \tag{6.10}$$

$$y_c = \pm \frac{-R_1^2 + 2R_2^2R_1^2 + 8R_1^2x^2 - R_2^2 + 8R_2^2x^2 - 16x^4}{x^2} \tag{6.11}$$

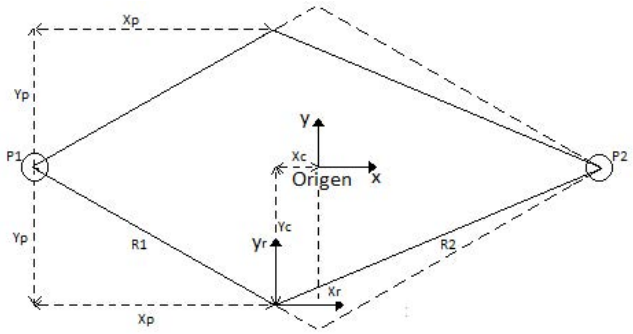


Figura 6.10: Diagrama de posición del robot.

6.2.3. Movimientos

Los movimientos son la acciones que realiza el robot para desplazarse o llegar a algún objetivo en particular, varios de ellos ya son provistos por la empresa que desarrolla al robot, sin embargo, estos han sido modificados, para tener un mejor desempeño.

Caminados

Los caminados son los movimientos que nos permiten desplazar al robot, ya sea para ir a despejar el balón ya sea con pasos laterales y frontales, obstruir la portería con pasos laterales, o regresar a su posición original con pasos frontales, hacia atrás o laterales.

Patadas

Las patadas son movimientos para mover al balón, estos se realizan cuando la pelota se encuentra lo suficientemente cerca y se puede despejar para evitar un gol por parte del equipo contrario. Para realizar este movimiento se debe de leer el valor del ángulo de acimut para decidir qué pie se va a utilizar en el pateo ya sea el izquierdo o el derecho.

Atajadas

Para este robot se le han programado diferentes atajadas, estas depende de la distancia a la que tiene que bloquear el disparo, esta distancia esta dada por 6.6.

Disparos de frente $(-100 \leq b \leq 100)$ el robot se coloca en cuclillas con los brazos ligeramente separados para hacerse más ancho sin que la pelota pase entre sus brazos y el cuerpo. En esta pose el robot tiene un ancho equivalente a 200 mm.

Disparos inclinados $(B \geq abs(b) \geq 100)$**.** Cuando el valor de b verifique $-B \leq b \leq -100$, el robot debe lanzarse hacia su derecha; pero si cumple $100 \leq b \leq B$, entonces debe lanzarse hacia su izquierda. Para lanzarse hacia un cierto lado el robot levanta el brazo y la pierna de dicho lado y la acción de la gravedad hace caer al robot para el lado correcto. Donde B es la distancia máxima a la que puede llegar el robot después de lanzarse.

Levantarse

Es la acción que debe de tomar el robot una vez que a detectado que se a caído, para ello se cuentan con dos tipos de levantamiento, uno si se encuentra boca abajo

y otro si esta boca arriba.

6.2.4. Algoritmo del Comportamiento del Portero

Primeramente, el robot debe colocarse en su posición inicial al centro de la portería, a unos centímetros de la línea de meta y mirando hacia la portería contraria; acto seguido se enciende el robot. Al encenderse se lleva a cabo el proceso de Inicialización de variables, posteriormente la búsqueda de la pelota, si ha encontrado la pelota realiza análisis de movimiento de la misma, así como de su trayectoria, y posteriormente se toma una decisión en respuesta al comportamiento del balón.

Inicialización: en el cual se hace la lectura de su orientación consultando el magnetómetro y el valor obtenido se guarda como δ_0 (orientación de referencia).

Búsqueda de pelota: Esta etapa se divide en diferentes sub etapas que a continuación se explican.

- **Paneo:** Se hace un barrido de exploración, de acuerdo al movimiento de paneo explicado en la Sección 6.2.1, esta rutina de movimiento permite explorar el terreno de juego a fin de detectar el balón. En cada posición del paneo se realiza una captura de imagen.
- **Análisis de imagen:** se realiza la captura de la imagen y se realiza el procesamiento necesario y correspondiente para determinar la posición del centroide de la pelota como se vio en la Sección 6.2.1.
- **Centrado de pelota:** esta rutina permite colocar el centroide de la pelota en el centro óptico de la cámara, lo cual es un requisito indispensable para realizar el calculo de la posición relativa de la pelota. Para ello se mueven los motores del cuello correspondientes a los ángulos de acimut y elevación (pan y tilt), para lograr el objetivo de centrado.
- **Posición de la pelota:** en esta etapa se estima la posición relativa de la pelota con respecto al referencial del robot, de acuerdo a las ecuaciones (6.3), (6.4) y (6.5), estos valores son almacenados en las variables x_1, y_1.

Análisis del movimiento de la pelota: en esta etapa se realiza un nuevo análisis de la imagen para recalcular la posición relativa de la pelota, sin embargo ahora se almacenan en las variables x_2, y_2.

Pelota estática : es cuando durante dos tomas de imagen consecutivas no existe un cambio significativo en su posición es decir $x_1 \approx x_2$ y $y_1 \approx y_2$. Si la posición en x es superior a los 30 cm, no se realiza ninguna acción, debido a que se encuentra fuera del rango de operación del robot, sin embargo si este es menor a este umbral, se procede a caminar hacia la pelota y despejarla.

Pelota en movimiento : es cuando $x_1 \neq x_2$ ó $y_1 \neq y_2$,con lo cual hay que ver si la pelota se acerca ($x_1 > x_2$) o se aleja ($x_1 < x_2$). Si la pelota se aleja únicamente se sigue con la etapa de análisis de movimiento. Sin embargo si la pelota se acerca, se procede con el análisis de trayectoria.

Análisis de trayectoria de la pelota: en este procedimiento se calcula la trayectoria del balón; en realidad se calcula el punto b en el que cruzará la línea de gol de acuerdo con la ecuación (6.6).

Acción a efectuar: el movimiento para bloquear al balón dependerá del punto b, como se estipularon en la Sección 6.2.3, si la posición b esta fuera de los límites del robot, no se realiza ninguna acción de movimiento.

Análisis de verticalidad: esta análisis se realiza en todo momento para ellos, se lee el valor del acelerométro para determinar la posición del robot si la gravedad se encuentra sobre el eje Z se encuentra de forma vertical; sin embargo, si la gravedad esta referenciada sobre el eje X, indica que el robot ha sufrido una caída y el signo de este determinara si esta boca arriba (positivo) o boca abajo (negativo).

Localizarse: Primeramente se debe Corregir la orientación del portero, para ello se lee la orientación actual del robot y se compara con δ_0, si son diferentes se gira hasta llegar a la posición original. Posteriormente realiza la búsqueda de los postes como se vio en la Sección (6.3.2) para determinar la dirección de la caminata para llegar a su posición original.

En el algoritmo 4 se muestra el pseudo-codigo de la implementación del portero.

Algoritmo 4: Portero

```
1  δ_0 ← leer_magnetometro(); pos_1 ← 0
   /* Inicia Paneo                                                    */
2  for i ← 1 to 4 do
3      for j ← 1 to 3 do
4          mover_pan(j)
5          CAPTURA_IMAGEN:/* Fase de Búsqueda de Pelota              */
6          verifica_verticalidad(); captura_imagen();
7          LUT ← leer_LUT()
           /* Busca el color naranja mas grande                       */
8          [pelota, Max_x, min_x, Max_y, min_y] ← Analisis_imagen(LUT, naranja)
9          if pelota=1 then
               /* Calcula el centroide con respecto al centro óptico  */
10             c_x ← (Max_x+min_x)/2 − 80; c_y ← (Max_y+min_y)/2 − 60
11             centrar_objeto(c_x, c_y)
12             if c_x ≠ 80 and c_y ≠ 60 then
13                 goto CAPTURA_IMAGEN
14             else
                   /* Analiza la trayectoria de la pelota             */
15                 [θ, φ] ← leer_angulos_cabeza()
16                 if pos_1 = 0 then
17                     d ← h/tan(φ); x_2 ← d cos(θ); y_2 ← d sin(θ); pos_1 ← 1
18                     goto CAPTURA_IMAGEN
19                 else
20                     x_1 ← x_2; y_1 ← y_2; d ← h/tan(φ); x_2 ← d cos(θ); y_2 ← d sin(θ)
21                 if x_1 ≈ x_2 and y_1 ≈ y_2 then
                       /* Pelota Estática                             */
22                     if x_1 < 30 then
23                         despejar(); localizarse(δ_0, postes); regresar_origen()
24                     else
25                         goto CAPTURA_IMAGEN
26                 else
                       /* Pelota en movimiento                        */
27                     if x_2 > x_1 then
28                         goto CAPTURA_IMAGEN
29                     else
30                         b ← (x_1 y_2 − x_2 y_1)/(x_1 − x_2)
31                         atajadas(b); localizarse(δ_0, postes); regresar_origen()
32         else
33             pos_1 ← 0
34     mover_tilt(i)
```

6.3. Control del Comportamiento de un Delantero

En esta Sección se explicara el algoritmo para que un robot humanoide se comporte como en un delantero de Fútbol, es decir su objetivo principal es anotar goles en la portería del equipo contrario.

El robot Darwin-OP cuenta con una computadora embebida la cual nos permite trabajar en 2 hilos de programación un dedicado a las funciones de movimiento y lectura de sensores, y el otro hilo se dedica exclusivamente al procesamiento de imágenes. En este ocasión se utiliza el robot humanoide Darwin-OP descrito en la Sección A.2, a groso modo el robot debe detectar a la pelota, identificar la portería contraria y realizar un disparo hacia ella. A continuación se en listan las funciones primordiales con las que debe de contar el delantero.

Percepción

- Detección y localización de la pelota,
- Detección y localización de portería contraria,
- Detección de la orientación.

Movimientos

- Movimientos de cabeza para búsqueda de objetos (paneo),
- Centrado de objetos,
- Caminados (de frente, hacia atrás y laterales),
- Disparos,
- Levantamientos.

6.3.1. Detección, Centrado y Localización de Pelota

Paneo

Se determinó que el movimiento que realizara durante el paneo sea una trayectoria elíptica, en la cual se toman con una tasa de 5 fps. En la ecuación (6.12) se muestra

la fórmula de una elipse parametrizada, centrada en el origen donde A es el ancho de la elipse, mientras que B es el alto de la misma. Cabe señalar que las amplitudes de la ecuación (6.12) se deben a las limitaciones físicas con las que cuenta el Robot Darwin-OP, el cual puede girar el ángulo de acimut $\pm 70°$, mientras que el ángulo elevación es de $-20°$ a $25°$.

$$\begin{aligned} \theta_p &= A\sin(t) \\ \phi_p &= B\cos(t) \end{aligned} \tag{6.12}$$

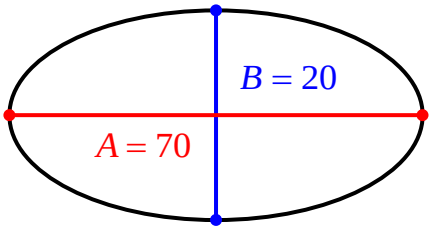


Figura 6.11: Elipse de paneo.

Detección de pelota

Para realizar la detección de la pelota se utilizaron la librería de Visión Artificial OpenCV 2.4, debido a que estas se encuentran optimizadas, sin embargo cabe resaltar que cada una de las funciones utilizadas se implementaron para comprender el funcionamiento de ellas.

Para la detección se realizó un esquema más robusto y eficiente que en el caso del robot Bioloid. Este esquema se divide en 3 etapas para el caso de la pelota naranja que se explicarán a continuación y en la figura 6.12 se presenta un ejemplo de la detección de la pelota:

Espacio Color: Se realiza un cambio de espacio color de RGB el nativo de la cámara del robot darwin a HSV a fin de lograr que un sistema de visión independiente de la iluminación. Además utilizando solamente el matiz H es posible distinguir el color de los objetos sin tener que recurrir a las tres componentes del espacio RGB.

Segmentación color: En este caso se realiza una búsqueda en la imagen para validar que pixeles se encuentran dentro del umbral del color deseado, pera para el color naranja el umbral que se utilizó es $5 \leq H \leq 14$. Para las reglas de Robocup 2014 en adelante la pelota se cambió al balón brazuca, para lo cual se modificó la etapa de segmentación color; en la cual se segmentan mínimamente 2 colores el blanco y algún otro representativo de la pelota en este caso el azul, posteriormente se realiza una suma binaria (or) para obtener una sola imagen segmentada que incluye la información de ambas segmentaciones.

Segmentación geométrica: En esta etapa se realiza la búsqueda de objetos circulares utilizando la transformada de Hough para círculos, de los círculos encontrados se considera al objeto de más grande (mayor radio) como la pelota debido a que trabajamos bajo la premisa de que es el único objeto de color naranja en el campo.

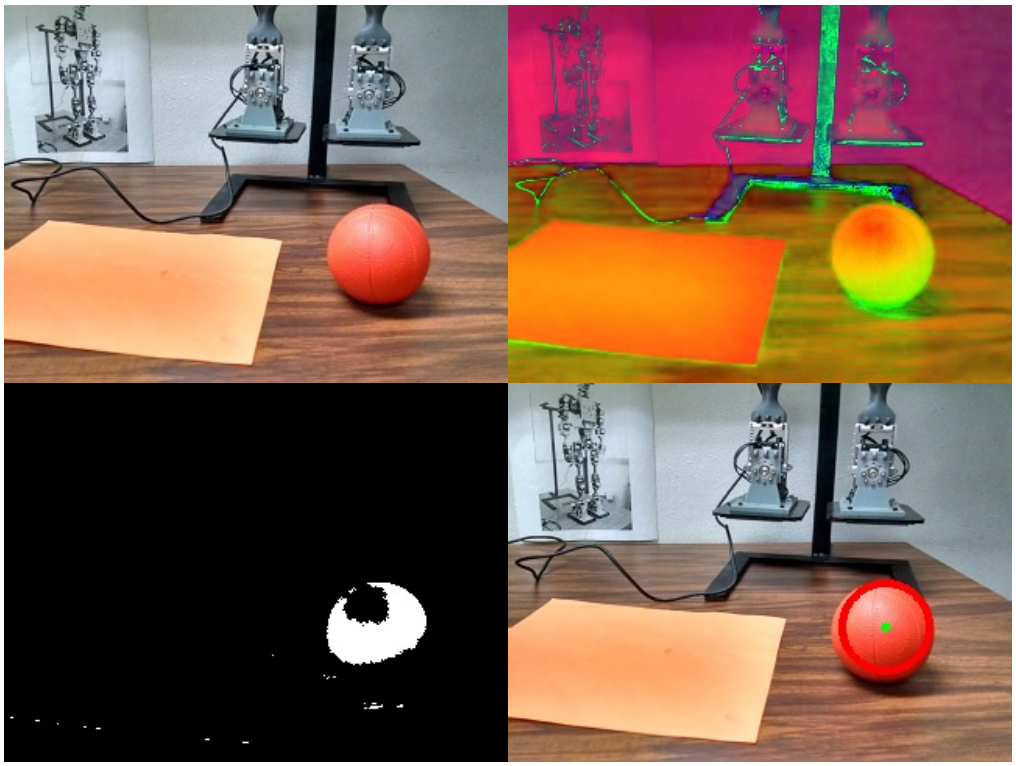


Figura 6.12: Detección de pelota naranja.

Algoritmo 5: Detección de Pelota

```
1  Function deteccion_pelota()
2  img ← capturar_imagen
3  img_hsv ← espacio_colorHSV(img)
   /* Se segmentan los dos colores principales                    */
4  img_seg1 ← segmentacion_color(img_hsv)
5  img_seg2 ← segmentacion_color(img_hsv)
   /* Se suman las imágenes segmentadas                           */
6  img_and ← img_seg2 and img_seg1
   /* Se dilata la imagen para rellenar espacios vacios           */
7  img_and ← dilatacion(img_and)
   /* Se erosiona la imagen para dejarla del tamaño original      */
8  img_and ← erosion(img_and)
9  circulos[i] ← Hough_circulos(img_and)
10 cir = circulos[0]
11 for i in circulos do
12     if circulos[i, 3] > cir[0, 3] then
13         cir = circulos[i]
14 return cir
```

Centrado de pelota

Para centrar la pelota se envían la posición del centro del círculo que se ha considerado como pelota, para mover los ángulos *tilt* y pan del robot, a fin de llevar el centro de la pelota al centro óptico de la cámara.

Localización de pelota

Una vez detectada la pelota se procede a su localización. Para ello se centra la pelota, es decir se hace coincidir el baricentro de la pelota con el punto principal de la imagen, formándose los mismos triángulos en los planos vertical y horizontal mostrados en las figuras 6.4 y 6.5 para el caso del humanoide Bioloid Premium.

Sin embargo, ahora lo resolvemos por un método alterno que consiste en medir el diámetro de la pelota en píxeles y usar esta medición en el cálculo de la hipotenusa del triángulo en el plano vertical cuyos catetos son h (altura de la cámara, conocida) y d (distancia robot-pelota, desconocida). La profundidad Z del baricentro de la pelota

es directamente proporcional al foco de la cámara e inversamente proporcional al tamaño en el plano imagen, conforme a la ecuación (6.13).

$$Z = f\frac{R}{r} \tag{6.13}$$

en donde f es el foco de la cámara, R es el radio real de la pelota en centímetros y r es el radio en el plano imagen en píxeles. Con esta información se puede calcular la distancia entre el robot y la pelota en el plano horizontal aplicando la ecuación (6.14).

$$d = \sqrt{Z^2 - h^2} \tag{6.14}$$

Algoritmo 6: Localización de Pelota

```
1 Function localizacion_pelota()
2 cir ← deteccion_pelota()
3 [θ, φ] ← leer_angulos_cabeza()
4 Z ← f R/r
5 d ← √(Z² − h²)
6 x ← d cos(θ)
7 y ← d sin(θ)
8 return [ x, y]
```

6.3.2. Detección y Localización de Portería Contraria

Para la detección de la portería únicamente nos interesan los ángulos en los que se encuentran los postes, para así tomar la decisión de pateo de la pelota, para ello se realiza un paneo de izquierda a derecha es decir con pan de -70 a 70 grados y con un tilt constante en cero grados, cuidando de que el eje óptico de la cámara permanezca siempre horizontal a fin de no introducir una pérdida del paralelismo en las rectas verticales.

El esquema para la detección consiste en las siguientes etapas:

Espacio Color: Se transforma el espacio color RGB al HSV como se hace con la detección de pelota.

Segmentación Color: Se realiza la detección del color de la portería, amarilla antes de la robocup 2014 y blanca en las posteriores.

Segmentación Geométrica: En esta etapa se realiza la búsqueda de líneas rectas paralelas al plano imagen a través de la transformada de Hough para líneas.

Clasificación de Lineas: Todas las líneas son agrupadas de acuerdo a la distancia de separación entre ellas, si tienen una separación menor a 5 píxeles se consideran como el mismo objeto. De cada objeto se guardan el primer y último elemento; así como, las posiciones de tilt y pan del robot.

Algoritmo 7: Detección de Portería

```
1  Function detección_porteria()
2  img ← capturar_imagen()
3  img_hsv ← espacio_colorHSV(img)
4  img_seg ← segmentación_color(img_hsv)
5  lineas[i] ← Hough_lineas(img_seg)
6  lineas_ord[i] ← Ordenar_lineas(lineas)
7  for i ← 0 to lineas_ord[m] do
8  |   postes[j] ← clasificación_lineas(lineas_ord)
9  |   pan_[k] ← leer_pan()
10 return postes, pan
```

6.3.3. Detección de la Orientación

Es necesario que el robot conozca su orientación para diferenciar entre ambas porterías, para ello se le doto al robot humanoide de una electrónica similar a la implementada en el robot Bioloid, basada en un magnetómetro (ver Sección 6.2.2), la única diferencia es que este nuevo sensor se comunica con la FitPc a través de una comunicación serial. En la figura 6.13 se muestra el esquema de conexión con el robot.

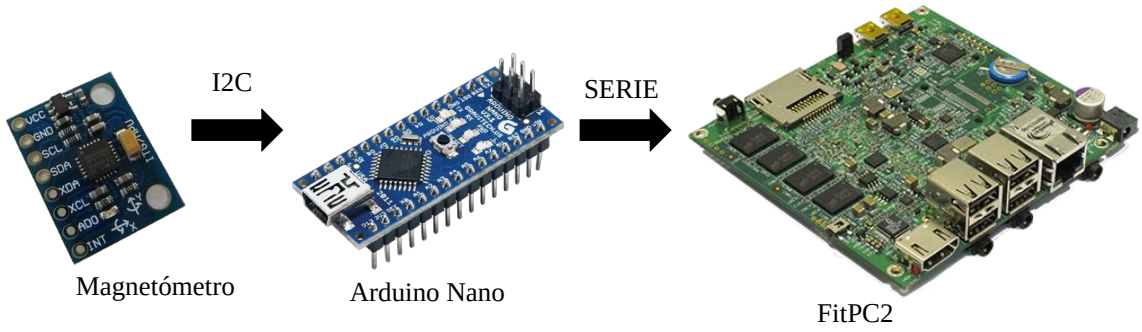


Figura 6.13: Esquema del magnetometro en Darwin-OP.

6.3.4. Decisión de Pateo

Para la decisión de patadas se consideran 4 ángulos: el ángulo de orientación del robot (δ), con respecto a su inicialización, el ángulo de la posición de la pelota(α), y los 2 ángulos correspondientes a cada poste detectado (β_1, β_2), con respecto al referencial del robot, estos se muestran en la figura 6.14.

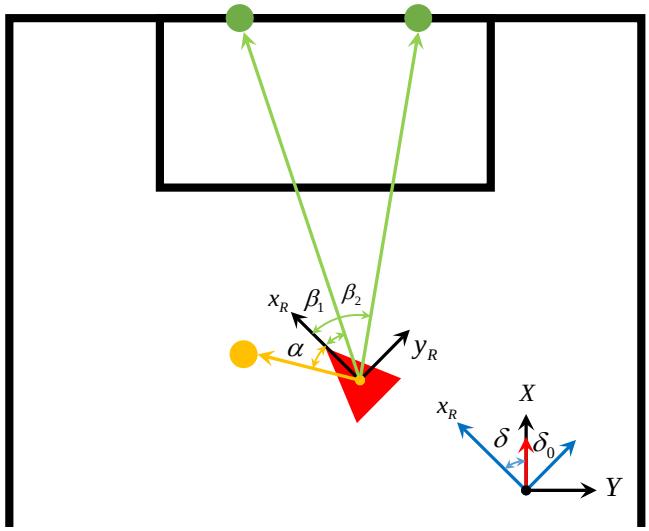


Figura 6.14: Ángulos para decisión de pateo.

Si el ángulo de orientación es mayor a 180° la tarea del robot es alejar lo más posible la pelota debido a que se encuentra orientado hacia su propia portería, para realizar esta acción lo más rápido se omite el paso de buscar la portería contraria, para ello se utiliza el esquema presentado en la ecuación (6.15) y en la figura (6.15)

se muestra el diagrama de dicho esquema.

$$\text{patada1}() = \begin{cases} \text{patada\_lateral\_izquierda}() & \Leftarrow & \pm 180° < \delta < \pm 180° - 10° \\ \text{patada\_lateral\_izquierda}() & \Leftarrow & 90° \leq \delta \leq 170° \\ \text{patada\_lateral\_derecha}() & \Leftarrow & \pm 180° > \delta > \pm 180° + 10° \\ \text{patada\_lateral\_derecha}() & \Leftarrow & -90° \geq \delta \geq -170° \end{cases} \tag{6.15}$$

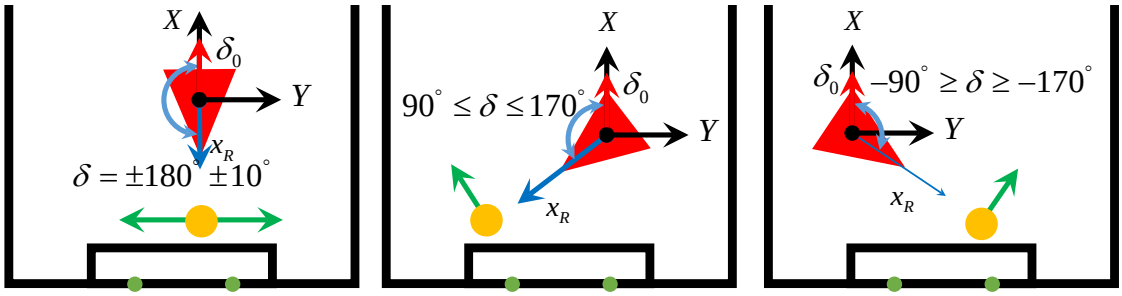


Figura 6.15: Esquema de pateo con $\delta \geq 90°$ y $\delta \leq -90°$.

Si el ángulo de orientación del robot δ se encuentra entre $-90°<90°$, se procede a buscar la portería, para determinar los ángulos β_1 y β_2, con los cuales se puede representar la situación del juego codificándolos en una secuencia de 3 caracteres **BPP**, **PBP** y **PPB**, correspondientes a la pelota (**B**) y a los postes (**P**). Esta simple descripción representa el orden en que fueron encontrados estos tres elementos haciendo una exploración de izquierda a derecha, a continuación se describen y la función de pateo queda definido como en la ecuación (6.16).

BPP: Es el caso en que la pelota se encuentra a la izquierda de la portería, es decir $\alpha < \beta_1 < \beta_2$, como se ilustra en la figura 6.16 a), en donde la pelota debe de ir con una dirección de $-90°$, realizado con una patada lateral izquierda.

PBP: Es cuando la pelota se encuentra entre los postes de la portería, es decir $\beta_1 < \alpha < \beta_2$, para lo cual realiza un disparo recto, con el pie que este más cerca de la pelota, este caso se muestra en la figura 6.16 b).

PPB: Es el caso en que la pelota se encuentra a la derecha de la portería, es decir $\beta_1 < \beta_2 < \alpha$, para ello se realiza una patada lateral derecha dirigiendo a la

pelota en una dirección de 90° con respecto al robot, este caso es mostrado en la figura 6.16 c).

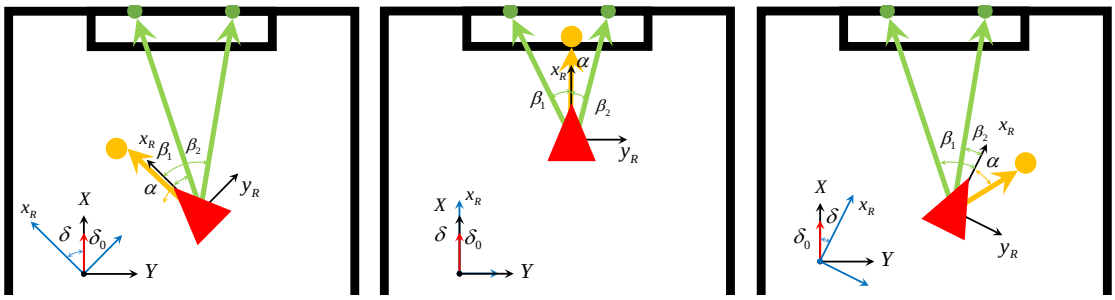


Figura 6.16: Esquema de pateo a) BPP, b) BPB, c) PPB.

$$\text{patada2}() = \begin{cases} \text{patada\_lateral\_izquierda}() & \Leftarrow & \alpha < \beta_1 < \beta_2 & (PBB) \\ \text{patada\_izquierda}() & \Leftarrow & \beta_1 \leq \alpha \leq \beta_2 \,\text{y}\, \alpha \geq 0 & (BPB) \\ \text{patada\_derecha}() & \Leftarrow & \beta_1 \leq \alpha \leq \beta_2 \,\text{y}\, \alpha < 0 & (BPB) \\ \text{patada\_lateral\_derecha}() & \Leftarrow & \beta_1 < \beta_2 < \alpha & (BBP) \end{cases} \tag{6.16}$$

6.3.5. Movimientos

Caminado

El robot DarwinOP, cuenta con un algoritmo de caminado parametrizado, el cual se puede modificar en cualquier instante de tiempo y genera la interpolación de la trayectorias en tiempo real [57]. A continuación se explicaran en que consisten los parámetros del caminado en el robot humanoide.

Compensación en X, Y, Z (X, Y, Z Offset): Es la compensación de la posición del pie, en X se refiere a la compensación en el plano sagital, es decir hacia adelante o hacia atrás, para la compensación en Y es sobre el plano frontal y este es la separación entre las piernas, mientras que la compensación en Z de la altura al robot. En la figura 6.17 se muestran estas compensaciones.

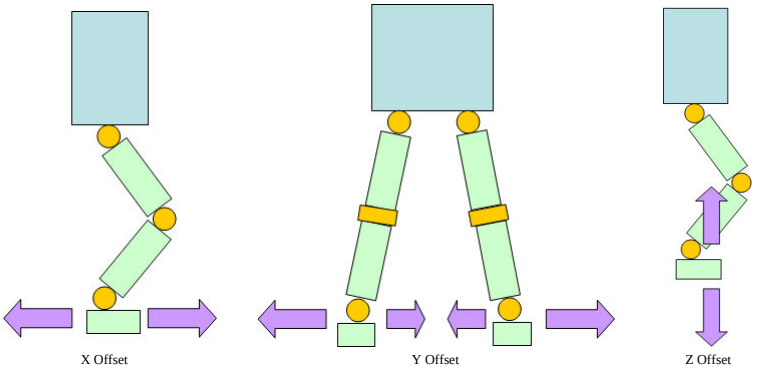


Figura 6.17: a) Compensación X, b) Compensación Y, c) Compensación Z

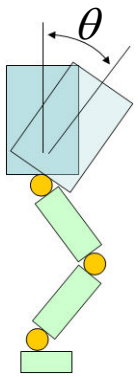


Figura 6.18: Inclinación de cadera

Inclinación de Cadera (hip pitch): Es el ángulo θ que se inclina el dorso del robot, como se muestra en la figura 6.18.

Tamaño de paso frontal y lateral (Step forward/back, Step right/left): Es la separación entre los pies a lo largo de un paso ya sea hacia adelante o lateralmente, esté parámetro esta en milímetros, y se muestra en la figura 6.19.

Tiempo de paso (Period time, DSP): Un paso se divide en dos etapas, en la etapa de soporte simple que es en la que solo un pie tiene contacto con el suelo y la etapa de doble soporte, que es cuando el ambos pies del robot están apoyados en el suelo. La variable *period time* configura el tiempo del paso, el cual incluye ambas fases del caminado, este se da en milisegundos, mientras que el tiempo en doble soporte se configura en la variable *Double Stance Period*

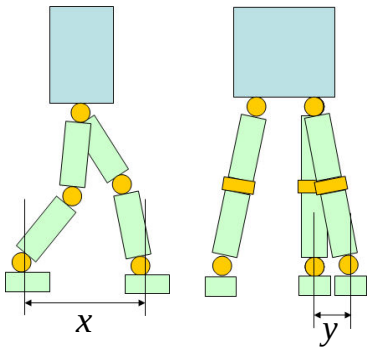


Figura 6.19: Tamaño del Paso a) Paso Frontal, b) Paso Lateral

(DSP), donde se introduce la proporción entre el tiempo total del paso y el tiempo en la fase de doble soporte. En la figura 6.20, se muestra un esquema de las etapas de un paso.

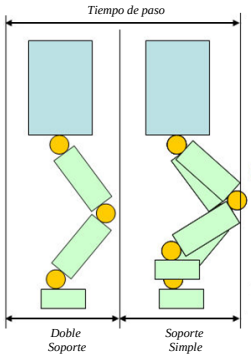


Figura 6.20: a) Soporte Simple, b) Doble Soporte

Levantamientos

Es el movimiento que se efectúa una vez que se ha caído, al igual que el robot Bioloid este cuenta con dos tipos de levantamientos, uno frontal y otro boca abajo, estos se han modificado dependiendo el tipo de suelo en el que se encuentre el robot.

Patadas

Las patadas son los movimiento que realiza para desplazar a la pelota, el robot cuenta con 4 tipos de patadas, 2 frontales una con el pie derecho o con el pie izquierdo, es decir el disparo es con un orientación de 0°, así como laterales uno con cada pie, donde la orientación del disparos es de ±90°, si se realiza la patada lateral con el pie izquierdo la pelota ira con una dirección de −90°, mientras que si la patada es con el pie derecho tendrá una dirección de +90°, sin embargo estas patadas no son muy eficientes, debido a que no llegan a desplazar la pelota una gran distancia, por lo que se modificaron para crear una patada de frente que dispare más fuerte.

6.3.6. Algoritmo del Comportamiento del Delantero

Al inicio del partido el robot es colocado en su posición inicial, que es cualquier parte del campo propio del equipo, sin pasar de la línea del área del portero; acto seguido se enciende el robot. Al encenderse se lleva a cabo el proceso de Inicialización de variables, posteriormente inicia con un caminado hacia el frente, mientras realiza la búsqueda de la pelota, si ha encontrado la pelota, camina hacia a ella hasta estar a la distancia de pateo, llegando a la distancia de pateo realiza un paneo horizontal en la búsqueda de la portería, y analiza su orientación, con lo que toma la decisión de que patada debe de utilizar.

Inicialización: Se almacena la orientación δ_0 leída del magnetómetro, para identificar cual es la portería contraria.

Búsqueda de pelota: Esta etapa se divide en diferentes sub etapas que se ejecutan simultáneamente, y termina con la localización de la pelota, las cuales se explican a continuación.

- **Paneo:** Se realiza el barrido elíptico de exploración (ver Sección6.3.1, esta rutina de movimiento permite explorar el terreno de juego a fin de detectar el balón. En cada posición del paneo se realiza una captura de imagen.
- **Análisis de imagen:** Se analizan las imágenes capturadas por la cámara para la detección y localización de la pelota (ver Sección 6.3.1).

Caminado: Esta rutina la realiza con el fin de buscar la pelota en todo el campo de juego.

Caminado hacia pelota: Realiza el caminado hacia la pelota, manteniendo la pelota en el eje óptico de la cámara, este se detiene cuando ha detectado que la distancia entre la pelota y el robot es suficiente para realizar una patada.

Búsqueda de portería: Realiza un paneo de horizontal de izquierda a derecha, buscando la portería (ver Sección 6.3.2), con el fin de determinar a qué ángulo se encuentra la portería.

Decisión de pateo: Dependiendo de las condiciones en las que se encuentre el robot es la patada que realizara, ya sea para anotar un gol o para alejar el balón de la propia portería (ver Sección 6.3.4).

En el algoritmo 26 se muestra el pseudo-código de la implementación realizada para el delantero.

Algoritmo 8: Delantero

```
1  δ_0 ← leer_magnetometro()
2  pelota ← 0
   /* Ciclo Infinito para todo el partido                              */
3  while 1 do
       /* Verificar la verticalidad se realiza en todo momento         */
4      verifica_verticalidad()
       /* Las funciones en el while se realizan en paralelo            */
5      while pelota ≠ 1 do
6          t ← leer_tiempo()
7          θ_p ← 70 sin(t)
8          φ_p ← 20 cos(t)
9          paneo(θ_p, φ_p)
           /* Calcula la posición de la pelota en el referencial del robot */
10         [x, y, pelota] ← localizacion_pelota();
           /* Camina de frente                                         */
11         camina()
12     CAPTURA_IMAGEN:
13     [x, y, pelota] ← localizacion_pelota();
14     [θ, φ] ← leer_angulos_cabeza()
       /* Determina si esta a distancia de disparo                     */
15     if x > 10 or y > 10 then
16         camina_pelota(x,y,θ,φ)
17         goto CAPTURA_IMAGEN
18     else
           /* Toma la decisión de la dirección de disparo              */
19         δ ← leer_magnetometro()
20         if δ >90° or δ <−90° then
21             patada1()
22         else
23             α ← tan(x,y)
24             [postes, β] ← deteccion_porteria()
25             patada2(α, β)
26     pelota ← 0
```

Capítulo 7

Control de las Habilidades de un Robot Cocinero

7.1. Reglas del HUMABOT CHALLENGE 2014

7.1.1. Pruebas

En el desafío HumaBot, el robot es una parte integral de la casa y ayuda a sus ocupantes vivir mejor. En esta edición, se realizan varias pruebas que se llevarán a cabo en la cocina de la casa.

1. *Apagar la estufa:* el robot debe comprobar que parrilla de la estufa se encuentra encendida, y apagarla presionando el botón correspondiente.

2. *Lista de la compra:* el robot debe reconocer qué productos (conjunto predefinido) están disponible en los estantes, y que productos son los faltantes, y hacer una lista de la compra.

3. *Preparación de la comida:* el robot debe agarrar un tomate y ponerlo en un recipiente para su cocción.

7.1.2. Reglas e Información técnica

La elección del método y lenguajes de programación están en manos de los equipos. La intervención humana está prohibida, pero se permite la conexión con el robot

a través de Wi - Fi, El código, debe ser presentado al jurado, sin ningún tipo de restricción. Se pueden utilizar marcas para la navegación y el reconocimiento de objetos, así como cualquier otro sistema para guiar el robot alrededor el entorno. Se permite que el robot caiga, pero debe levantarse y continuar con la tarea por sí mismo. Cualquier intervención humana durante la prueba dará lugar a una penalización que se aplica a los resultados de esta prueba.

7.1.3. Navegación

La navegación es un requisito para cada prueba. De hecho, el robot debe comenzar en su posición de base e ir a los elementos con los que debe interactuar. La elección de la técnica utilizada por el robot para orientarse en la habitación es libre. Las marcas de orientación serán colocadas en algunas superficies (Armario, mesa). Los equipos son libres de añadir marcas, antes del inicio de la prueba.

7.1.4. Ambiente

El área de caminado es de 85 cm $\times$ 90 cm, rodeada por los muebles (estufa, estante y mesa) y paredes de 45 cm de altura. El suelo es una plataforma de madera elevada para facilitar la manipulación de los objetos en las pruebas.

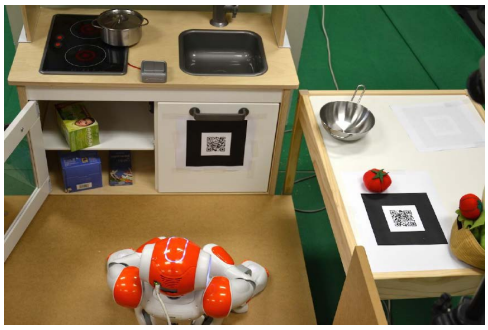


Figura 7.1: Ambiente

7.1.5. Objetos para Lista de Compras

En esta prueba, se le pide al robot inspeccionar los estantes y decirle al usuario una lista de los objetos faltantes, de los siguientes:

Figura 7.2: a)Cubos de Caldo, b) Clavo c) Café, d) Té, e) Palomitas y f) Muesli

7.2. Localización

Debido a que las reglas permiten agregar marca al ambiente, colocamos la marca mostrada en la figura 7.3, la cual es un cuadro de 10 cm de lado, esta marca se coloca en el área donde se realizara la prueba actual. Con la finalidad de generar un patrón de fácil reconocimiento, así como de ser capaz de colocarse con una orientación específica se consideró lo siguiente:

- Utilizar círculos como elementos constituyentes ya que el algoritmo de transformada de Hough, ampliamente utilizado en problemas de visión puede resolverse en tiempo real y configurar parámetros que ayuden a evitar posibles detecciones erróneas.

- Emplear 4 colores distintos, lo cual crea una orientación única del cuadrado a buscar. La selección de colores no se realizó de manera aleatoria, sino que se propuso emplear las tonalidades de rojo, azul, verde y negro debido a su rápida segmentación y bajas posibilidades de confusión entre sí.

Conocidas las medidas referentes a las distancias entre los círculos del patrón físico original, es posible, mediante el uso de una cámara, establecer la correspondencia entre los puntos 3D del espacio físico con los 2D de una imagen procesada en tiempo real. Para ello se emplean las soluciones correspondientes al problema Perspective-n-Point (PnP)1, en el cual mediante un set de n puntos en el espacio 3D se puede estimar la posición de la cámara mediante la imagen que esta obtiene. Si bien una

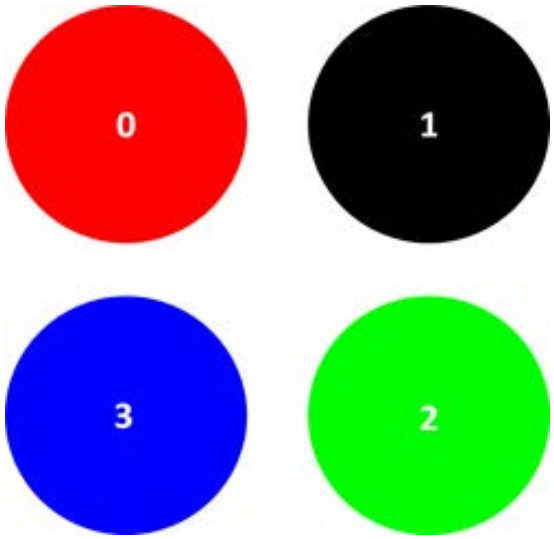


Figura 7.3: Patrón de localización

solución común considera $n = 3$ como un valor óptimo, usualmente se sugiere trabajar con $n > 3$; OpenCV, la librería utilizada en el código que fue realizado necesita 4 puntos para hacer este proceso, por lo cual el cuadrado (formado por los centros de los 4 círculos) fue considerado una opción viable la detección de los vértices del cuadrado está dado por los centroide de los círculos los cuales se calcularon con el algoritmo 15 y el algoritmo para la estimación de pose se muestra en el algoritmo 10.

Previo al desarrollo del resto del algoritmo se necesitan conocer los parámetros propios de la cámara a emplear (matriz de la cámara y coeficientes de distorsión), en este caso de la incorporada en el robot NAO, cuyos parámetros son:

$$F = \begin{bmatrix} 795.80 & 0 & 326.03 \\ 0 & 796.41 & 230.22 \\ 0 & 0 & 1 \end{bmatrix} \tag{7.1}$$

$$K = \begin{bmatrix} 0.454 & -2.359 & -0.0028 & 0.0042 & 4.7523 \end{bmatrix} \tag{7.2}$$

Algoritmo 9: Centroide

```
1  Función centroide(imagen, color, tipo)
   Salida: centro
2  bin ←segmentar(imagen, color)
3  hsv ←convertir_bgr2hsv(imagen)
4  if tipo='circulos' then
5  |  c_x, c_y, r ←Hough_Circulos(bin)
6  |  if tamaño(c_x)=1 then
7  |  |  centro ← c_x, c_y, r
8  |  else
9  |  |  centro ← max([c_x, c_y],r)
10 else
11 |  lin ← Hough_Lineas(bin)
12 |  lin ← ordena(lin)
13 |  c_x ← (lin[u]_x − lin[0]_x)/2 + lin[0]_x
14 |  c_y ← (lin[u]_y − lin[0]_y)/2 + lin[0]_y
15 |  centro ← [c_x, c_y]
```

Algoritmo 10: Estimación de Pose

```
1  Función pose_robot(imagen, F, K, x)
   Salida: rvec, tvec
2  board[0] ← [0,0,0]
3  board[1] ← [0,x,0]
4  board[2] ← [x,0,0]
5  board[3] ← [x,x,0]
6  centros[0] ← centroide(imagen, azul, 'circulos')
7  centros[1] ← centroide(imagen, verde, 'circulos')
8  centros[2] ← centroide(imagen, rojo, 'circulos')
9  centros[3] ← centroide(imagen, negro, 'circulos')
10 rvec, tvec ←solvePnP(board, centros, F, K)
```

7.3. Apagar Estufa

Esta prueba consiste en determinar cuál de las dos parrillas se encuentra encendida y apagarla, para el botón de la parrilla inferior es el de la izquierda y botón de la parrilla superior es el de la derecha.

El primer paso es colocarse frente a la estufa para ello se utiliza el algoritmo 10,

especificándole que se quede a 30 cm de la marca, la distancia está determinada de forma que pueda ver la estufa y el botón, sin necesidad de moverse nuevamente.

Posteriormente se realiza el paneo para identificar la parrilla encendida, para ellos se analizan las imágenes buscando círculos de color rojo y estimando la posición de su centro con respecto al sistema de referencia del robot, también se establece un umbral u en la profundidad para poder identificar cual es la parrilla detectada, de acuerdo al algoritmo 13.

Una vez identificada la parrilla se realiza una nuevo paneo para buscar el botón, este es de forma rectangular por los que se buscan líneas a través de la transformada de Hough. Las todas las líneas verticales, estas son ordenadas por su posición en la imagen y posteriormente clasificadas de acuerdo a la distancia entre ellas, es decir si están juntas pertenecen a un mismo objeto. Se considera como botón al mayor conjunto de líneas y a este se calcula su centroide, para así determinar la posición del botón con respecto al robot. Por último se presiona el botón correspondiente a la a parrilla detectada. En el algoritmo 14 se muestra el pseudo-código de la aplicación.

Algoritmo 11: Posición del Objeto

```
1 Función pose_obj(tilt, pan)
  Salida: x, y
2 x ←−0.1tan(90 − tilt) sin(pan)
3 y ←−0.1tan(90 − tilt) cos(pan)
```

Algoritmo 12: Centrar Objeto

```
1 Función centra_obj(centro)
  Salida: tilt, pan
2 while centro[0], centro[1]) != (F[0,3],F[1,3]) do
3 |   centrar_cabeza(centro)
4 |   [tilt, pan]←leer_tilt_pan()
```

Algoritmo 13: Parrilla Encedida

```
1 Void parrilla(imagen, color,u)
  Salida: parrilla
2 centro←centroide(imagen,rojo)
3 [x,y] ←centra_obj(centro)
4 if y > u then
5 |  parrilla ← 1
6 else
7 |  parrilla ← −1
```

Algoritmo 14: Presiona Botón

```
1 Void presiona_boton(parrilla)
2 boton←centroide(imagen, gris, 'lineas')
3 x,y ← centra_obj(boton)
4 x ← x + 5parrilla
5 mueve_brazo1(x,y,35)
```

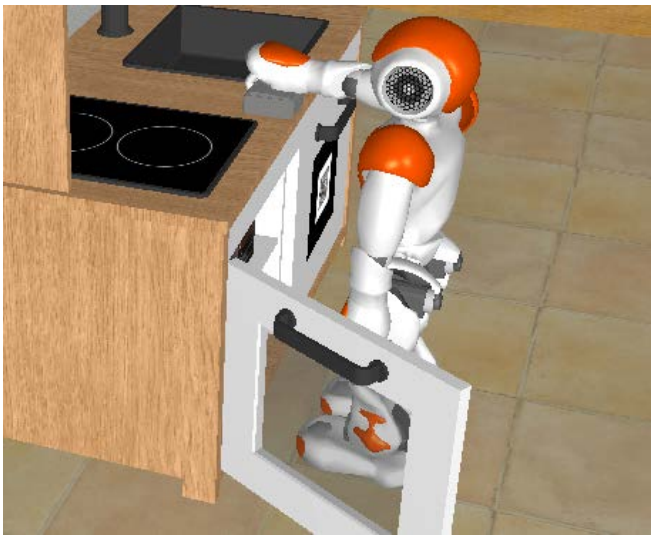


Figura 7.4: Simulación de apagar estufa

7.4. Lista de Compras

En esta tarea hay que realizar una lista de compras, con los objetos faltantes del estante. Antes de realizar la prueba se realiza la etapa de entrenamiento del robot,

lo cual consiste en realizar la base de datos de los objetos, ecuación (7.3)

$$D[m] = \begin{bmatrix} u & v & i & j & desc \end{bmatrix} \tag{7.3}$$

Donde m es número de elementos en la base de datos, (u,v) el *keypoint* detectado, *desc* es el descriptor del Keypoint, i el identificador del objeto (6 objetos) y j el identificador de la vista (4 vistas) del objeto. Para realizar la detección de Keypoint y sus Descriptores se utilizó el algoritmo FAST debido a que es más rápido en comparación con otros como SIFT o SURF, además en esta competencia el tiempo de ejecución es un factor en la evaluación del desempeño del robot. Previo a la detección de características se realiza un algoritmo para que identifique un objeto para únicamente determinar las características de este, y eliminar las del ambiente donde se está entrenando al robot, de acuerdo al algoritmo 15. Mientras que la base de datos se genera con el algoritmo 16. En la figura 7.5 se muestra la imagen original y la imagen detectada como objeto.

Algoritmo 15: Busca Objeto

```
1 Función Objeto(imagen,base)
  Salida: img_recortada
2 img_resta ← imagen – base
3 img_umbral ← umbraliza(img_resta,4)
4 img_borde ← detecta_bordes(img_umbral, 'canny')
5 rectangulo ← min_max_bordes(img_borde)
6 img_recortada ← rec_imagen(imagen, rectangulo)
```

Figura 7.5: Detección del objeto en el entrenamiento

Algoritmo 16: Base de Datos

```
1  Función caracteristicas(camara,num_obj)
   Salida: D,imgenes
2  m = 0
3  base←obtener_imagen(camara,'gray')
4  for i ← 1 to num_obj do
5      for j ← 1 to 4 do
6          img←obtener_imagen(camara)
7          imgenes[i][j] ← Objeto(imgenes,base)
8          kp, desc ← FAST(img[i][j])
9          for k ← 1 to tamaño(kp) do
10             D[m] ← [u, v, i, j,desc(k)]
11             m = m + 1
```

Como en la prueba anterior lo primero es ubicarse en el estante para ello se utiliza el algoritmo 10, deteniendo al robot a una distancia de 50 cm.

Ubicado en la zona de búsqueda realiza un paneo en el estante, y en cada imagen adquirida se realiza la búsqueda de los 6 objetos.

Para ello se calculan los descriptores de la imagen actual con el algoritmo 16, donde i, j y obj son 0 para todas las características, a continuación se realiza el proceso de matching entre los descriptores de la imagen actual y los almacenados en la base de datos, calculando la distancia euclidiana entre ambos descriptores y almacenándolos en una matriz de $m \times n$, donde n es el número de características encontradas en la imagen actual, como se muestra en la ecuación (7.4).

$$dist = \begin{bmatrix} |D[0]_3 - A[0]_3| & \cdots & |D[0]_3 - A[n]_3| \\ \vdots & \ddots & \\ |D[m]_3 - A[0]_3| & & |D[m]_3 - A[n]_3| \end{bmatrix} \tag{7.4}$$

De acuerdo a los resultados del matching al vector A se le agrega al elemento obj, el correspondiente valor del vector D. Se cuentan la cantidad de características de cada objeto. Si el total de características encontradas supera un umbral de matching (S) se establece que el objeto fue detectado. Y finalmente se realiza la lista con los objetos que no se detectaron, en el algoritmo 18 se muestra el programa general.

Algoritmo 17: Emparejamiento

```
1  Función matching(dist,U)
   Salida: match
2  d = 0
3  while d < U do
4      d ←min(dist)
5      if d < U then
6          agrega_match(m,n,match)
```

Algoritmo 18: Lista de Compras

```
1  Void shopping_list ()
2  D ←Cargar_datos() img← obtener_imagen()
3  pose ← pose_robot(img, K,F,10)
4  mover_robot(pose,50)
5  for i ← 1 to 9 do
6      mover_cabeza(i)
7      img←obtener_imagen()
8      A ←caracteristicas(img,0)
9      dist←matriz_distancias(D,A)
10     match←matching(dist)
11     A ← agrega_num_objeto(match,D)
12     C ←cuenta_objetos(match, A)
13     lista←agrega_objetos_detectado(C)
14 mostrar(lista)
```

Figura 7.6: Lista de Compras

7.5. Preparación de la Comida

Esta tarea consiste en tomar un jitomate y colocarlo en un recipiente junto al jitomate, estos objetos se encuentran en una mesa al lado derecho del estante. Como en las demás pruebas es necesario llevar el robot a la mesa para ello se coloca la marca la mesa y por medio del algoritmo 10 llevamos el robot hasta la mesa y le decimos que se detenga 30 cm antes de ella. Considerando que el jitomate es una esfera redonda, aplicamos el algoritmo 19, para encontrar la posición del jitomate con respecto al robot. Localizado el jitomate se procede a tomarlo, en esta ocasión y debido a que nuestro robot es la versión de RoboCup no tiene articuladas las manos ni las muñecas por lo que se trabajó en que el movimiento para agarra el jitomate fuera tomarlo con ambas manos. Ya con el jitomate en las manos hay que mover el robot hacia la derecha 30 cm la cual es la posición en la que se encuentra el recipiente donde colocarlo.

Algoritmo 19: Busca Jitomate

```
1 Función jitomate (imagen)
2 jitomate=centroide(imagen, red, 'circulos')
3 mueve_brazo2(jitomate)
4 mueve_robot(0,30)
5 mueve_brazo3()
```

Figura 7.7: Prueba de jitomate en concurso

Capítulo 8

Conclusiones Generales y Trabajo a Futuro

8.1. Conclusiones

En esta tesis se aplicaron dos diferentes técnicas para resolver el problema de OV 3D-3D consistente en: determinar la trayectoria seguida por un robot móvil empleando únicamente visión. Estas técnicas son por medio de un técnica de trilateración para una cámara estéreo por medio de puntos características propuesta por nosotros y la otra a través de flujo óptico y la estimación de la matriz esencial de la cámara monocular. Así como dos técnicas para la relocalización de robots móviles. Esas técnicas son la que propone M. C. Ricardo Carrillo en su tesis de Maestría y la que proponemos nosotros en esta tesis. Comparando los resultados obtenidos al aplicar estas dos técnicas de relocalización en el robot Darwin se obtuvo que el algoritmo de trilateración aquí propuesto para la estimación de la relocalización de un robot móvil es de más rápida ejecución y proporciona una buena estimación (error comparado con otros).

E la implementación de este algoritmo se resolvieron varios problemas individuales entre los cuales se encuentran la asociación de datos, el seguimiento entre características, ya sea en un par estéreo o en dos instantes de tiempo consecutivos, utilizando diferentes metodologías de descriptores o bien usando flujo óptico.

El problema principal era recuperar la pose del robot dentro de un campo de

fútbol aplicado a competencias. Para ello se compararon en la Sección 4.2 dos metodologías de relocalización, una por mapa de imágenes sintéticas, en el cual es efectivo si la escena se mantiene constante; sin embargo durante un partido de fútbol la escena está en constante movimiento, por lo cual este algoritmo no satisface completamente al problema, mientras que la relocalización propuesta por descriptores de puntos 3D, resuelve el problema cuando la escena es dinámica, debido a que el mapa de características únicamente cuenta con la información del entorno estático alrededor de la cancha y automáticamente desprecia características provenientes de los demás robots (del ambiente dinámico).

En el caso de la OV para el cuadrirotor, se identificó que la mejor opción para realizar el seguimiento de características era a través de flujo óptico, debido a que los algoritmos por descriptores son más lentos, pues realizan una búsqueda en todos los descriptores almacenados en la base de datos, a diferencia del algoritmo de flujo óptico de Lucas-Kanade que sólo se calcule en una venta alrededor de la característica. En este caso se obtuvo una estimación de la trayectoria similar a la que proporcionó el sensor GPS, sin embargo falta resolver el problema de la escala absoluta.

En ambos casos se compararon los resultados obtenidos con los de la base de datos de KITTY Benchmark, con el cual se validan ambos algoritmos.

Otro logro importante fue la obtención de la cinemática directa e inversa del robot humanoide AH1N2, con estos modelos se pueden realizan las trayectorias de caminado de dicho robot; así como, la manipulación de objetos identificados por medio de Visión Artificial.

En cuanto a las aplicaciones con robots móviles, principalmente humanoides, en el Capítulo 6 se presenta el comportamiento de dos jugadores de fútbol, un portero y un delantero, en donde ambos robots realizan su tarea satisfactoriamente, con estos robots se obtuvieron buenos resultados en 2014 llegando al tercer lugar en el Torneo Mexicano de Robótica. Mientras que en el Capítulo 7 se presentan una aplicación de un robot humanoide en un entorno de cocina, realizado en el concurso Humanoide Challenger 2014, en el cual se obtuvo un segundo lugar, donde se pusieron varios problemas de Visión Artificial, como localización, identificación de objetos, entre otros, los cuales se resolvieron de una manera correcta y procurando que se realizaran en el menor tiempo posible.

8.2. Trabajo a Futuro

Uno de los principales objetivos de investigación en el Laboratorio de Robótica y Visión Artificial (RoVisA) es obtener el mayor grado posible de Autonomía de sus robots: robots móviles con 2 ruedas, humanoides, drones multirrotor y más recientemente automóviles. Esto implica la necesidad de contar con técnicas de Visión Artificial complejas que deben realizarse en tiempo real, tales como la autolocalización en un mapa, detección y lectura de señales de transito, detección y localización de una gran cantidad de objetos en entornos estructurados y no estructurados. Así, deberemos desarrollas la metodología y algoritmos de Visión Artificial para ellos, basados en nuestras experiencias previas de Vídeo SLAM, Odometría Visual.

Por otro lado, en aspectos de competencias tenemos prevista nuestra participación en eventos con drones autónomos en tareas de rescate (TMR, IMAV y FIRA Air), automóviles autónomos (TMR, IPN- Universidad Libre de Berlín) y en pruebas con humanoides como: Basketball, Escalado de paredes, Maratón, Carrera de Obstáculos, Tiros de Penal, Levantamiento de Pesas, Fútbol, Carreras de Velocidad.

Apéndice A

Plataformas de Desarrollo

A continuación se describen las plataformas de desarrollo las cuales se han utilizado en este proyecto doctoral.

A.1. Robot Humanoide BIOLOID

A.1.1. Arquitectura

BIOLOID es un kit de desarrollo de la empresa Robotis, este robot cuenta en su versión comercial original con 18 gdl, 6 se utilizan en brazos (3 gdl por brazo) y 12 en las piernas (6 gdl por pierna). A este humanoide comercial se le hicieron modificaciones para que pudiera contar con 2 gdl extra en el cuello que le permita realizar los movimientos de acimut y elevación de la cabeza. Cuenta con un sistema electrónico-computacional conformado por el controlador CM-510, basado en el microcontrolador AVR de 8 bits ATmega256, el cual cuenta con una serie de botones, indicadores LED, canales analógicos y buses de comunicación que le permiten controlar los actuadores (servomotores Dynamixel AX-12) y otros dispositivos tanto analógicos como digitales. Cuenta además con un sistema de percepción visual basado en el HaViMo2.0 (Hamid'sVision Module 2.0) [58], [59]. Además, cuenta con una unidad inercial de 9 gdl, 3 gdl en el acelerómetro (ADXL-345), 3 gdl en el giroscopio y 3 gdl en el magnetómetro (HMC5883L), en la figura A.1 se muestra al robot BIOLOID [60].

Figura A.1: Robot Humanoide BIOLOID.

Cuadro A.1: HAVIMO LUT.

Índice	Color	# Píxeles	$\sum x$	$\sum y$	x_{max}	x_{min}	y_{max}	y_{min}
1	04	3B 05	63 89	FC A3	31	00	67	41
2	03	24 08	83 65	53 A6	8C	4F	6B	43

A.1.2. Sistema de Visión

Havimo 2.0 es un sistema de visión cerrado que proporciona una LUT (Look-up-Table) conteniendo información acerca de hasta 16 diferentes "manchas" de color (objetos) detectadas siempre y cuando en la fase de calibración fuera de línea se hayan defino dichos colores. Este sistema permite definir hasta siete diferentes grupos de colores en el espacio YCrCb. La LUT que produce el sistema Havimo 2.0 puede contener hasta 16 filas, una por cada objeto detectado, numerándolas con un índice del 1 al 16 (primera columna de la LUT). Las restantes 8 columnas de la LUT contienen ocho propiedades del objeto detectado y son: Color, número de píxeles (área), suma de las coordenadas x de todos sus píxeles ($\sum x$), suma de las coordenadas y de todos sus píxeles ($\sum y$), así como las coordenadas del rectángulo de encuadre del objeto detectado (x_{max} ,x_{min}, y_{max} y y_{min}, tal como se ilustra en la Tabla A.1.

A.2. Robot Humanoide DARWIN-OP

DARWIN-OP es una plataforma de desarrollo de la empresa Robotis cuyas siglas en ingles significan *Dynamic Anthropomorphic Robot with Intelligence–Open Platform*, este cuenta con 20 gdl, 6 en los brazos (3 gdl por brazo), 12 en las piernas (6 gdl por pierna) y 2 en la cabeza, estos se encuentran actuados por servo motores Dynamixel MX-28, su sistema computacional esta conformado por una controlador CM-710, que permite la comunicación con los servo motores y una computadora FitPC2 con un microprocesador Atom a 1.6Ghz, con el sistema operativo Ubuntu 12, también cuenta con una cámara de resolución de 640×480, además de la unidad inercial dotada de acelerómetro, giroscopio y magnetómetro. En la figura A.2 se muestra al robot humanoide DARWIN-OP [57].

Figura A.2: Robot Humanoide DARWIN-OP.

A.3. Robot Humanoide NAO

NAO es un robot de la empresa francesa Aldebaran Robotics, que cuenta con 21 gdl., su altura es 573mm y su peso es de 5.02 Kg. Esta equipado con un CPU x86 AMD Geode 500MHz y una memoria flash de 2GB, la batería es de ion-litio. Tiene 2 cámaras HD de resolución de 1280 × 960, entre los sensores con los que cuentas son sensores de tacto en los pies, sensores ultrasónicos, una unidad inercial con giroscopio de 2 ejes y acelerómetro de 3 ejes [61].

Figura A.3: Robot Humanoide NAO.

A.4. Robot Humanoide AH1N2

El robot AH1N2 es un robot Humanoide de manufactura dentro del CINVESTAV desarrollado por Andrés Cobos esta es una actualización del Robot AH1N1, desarrollado previamente en el mismo laboratorio por el Dr. Juan Manuel Ibarra, este se encuentra basado en la misma electrónica que el robot Darwin-OP, cuenta con 26 gdl y una cámara estereoscópica minoru 3D, en el Capítulo de modelado de un robot humanoide se explicara más a fondo en que consiste este robot.

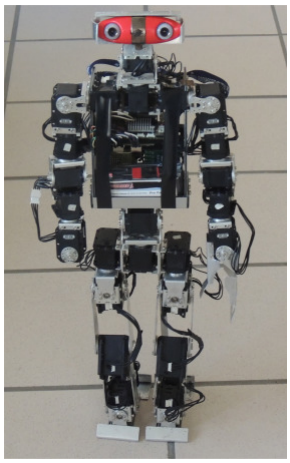


Figura A.4: Robot Humanoide AH1N2.

A.5. BEBOP

El cuadrirotor Bebop es de la empresa Parrot, se encuentra formado por 4 motores sin escobillas Outrunner, estructura reforzada de fibra de vidrio (20 %) y Grilamid, tiene un peso de 0.5 kg, una dimensión de 38 × 33 × 9 cm, cuenta con un procesador dual-core con GPU quad-core, una unidad inercial, sensor GPS, sensores ultrasónicos, una cámara de ojo de pescado de 14 Mega pixeles y control remoto. Este quadrirotor es de arquitectura abierta es decir, es completamente programable [62].

Figura A.5: Drone Bebop.

Bibliografía

[1] TESLA. (2017). [Online]. Available: https://www.tesla.com

[2] DARPA. (2017). [Online]. Available: https://www.darpa.mil/program/darpa-robotics-challenge

[3] R. S. N. C. J. M. I. Z., "Inertial-visual odometry on mobile devices," *RoboCity16 Open Conference on Future Trends in Robotics*, 2016.

[4] R. Hartley and A. Zisserman, "Multiple view geometry in computer vision." *Cambridge University Press*, 2000.

[5] Z. Zhang, "A flexible new technique for camera calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 22, pp. 1330–1334, Noviembre 2000.

[6] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. AVC*, 1988, doi:10.5244/C.2.23.

[7] H. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Stanford Univ, 1980.

[8] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," *IEEE International Conference on Computer Vision*, 2005.

[9] D. G. Lowe, "Object recognition from local scale-invariant features," *The Proceedings of the Seventh IEEE International Conference*, 1999.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," *IEEE International Conference on Computer Vision*, 2011.

[11] P. Mahalanobis, "On the generalised distance in statistics," in *Proceedings of the National Institute of Science of India*, 1936.

[12] D. Fleet, S. Beauchemin, and J. Barron, "Perfomance of optical flow techniques," in *International journal of computer vision*, 1994.

[13] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of Imaging Understanding Workshop*, 1981.

[14] B. K. Horn and B. G. Schunck, "Determining optical flow," in *ArtificialIntelligence*, 1981.

[15] Itseez, "Open source computer vision library," https://github.com/itseez/opencv, 2015.

[16] Opencv, "The opencv reference manual," Itseez, April 2014.

[17] P. H. Torr and A. Zisserman, *MLESAC: A New Robust Estimator with Application to Estimating Image Geometry.* Computer Vision and Image Understanding, 2000.

[18] D. Nister, "An efficient solution to the five-point relative pose problem," in *Proc. Int. Conf. Computer Vision and Pattern Recognition*, 2003.

[19] L. Higgins, "A computer algorithm for reconstructing a scene from two projections," *Nature*, 1981.

[20] M. Stephen and C. Harris, "3d wire-frame integration from image sequences," in *Proc. AVC*, 1988, doi:10.5244/C.2.25.

[21] D. Scaramuzza and F. Fraundorfer, "Visual odometry: Part i - the first 30 years and fundamentals," in *IEEE Robotics and Automation Magazine*, 2011.

[22] ——, "Visual odometry: Part ii - matching, robustness, and applications," in *IEEE Robotics and Automation Magazine*, 2011.

[23] L. Matthies and S. Shafer, "Error modeling in stereo navigation," *IEEE J. Robot. Automat*, 1987.

[24] C. Olson, L. Matthies, M. Schoppers, and M. W. Maimone, "Robust stereo ego-motion for long distance navigation," in *IEEE CCVPR*, 2000.

[25] H. Moravec, "Towards automatic visual obstacle avoidance," in *Proc. 5th Int. Joint Conf. Artificial Intelligence*, 1977.

[26] M. Hannah, "Computer matching of areas in stereo images," Ph.D. dissertation, Stanford Univ., 1974.

[27] W. Forstner, "A feature based correspondence algorithm for image matching," *Int. Arch. Photogrammetry*, 1986.

[28] S. Lacroix, A. Mallet, R. Chatila, and L. Gallo, "Rover self localization in planetary-like environments," in *Proc. Int. Symp. Articial Intelligence, Robotics, and Automation for Space*, 1999.

[29] A. Milella and R. Siegwart, "Stereo-based ego-motion estimation using pixel tracking and iterative closest point," in *Proc. IEEE Int. Conf. Vision Systems*, 2006.

[30] P. Corke, D. Strelow, and S. Singh, "Omnidirectional visual odometry for a planetary rover," in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2005.

[31] T. Pavlidis and K. Daniilidis, "Monocular visual odometry in urban environments using an omnidirectional camera," in *Proc. IEEE/ RSJ Int. Conf. Intelligent Robots and Systems*, 2008.

[32] M. Self, R. Smith, and P. Cheeseman, "Estimating uncertain spatial realtion ships in robotics," *Autonomous Robot Vehicles*, 1987.

[33] A. Chiuso, P. Favaro, and H. J. S. Soatto, "3-d motion and structure from 2-d motion causally integrated over time: Implementation," in *Proc. European Conf. Computer Vision*, 2000.

[34] M. C. Deans, "Bearing-only localization and mapping," Ph.D. dissertation, Carnegie Mellon Univ., Pittsburgh, 2002.

[35] A. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. Int. Conf. Computer Vision*, 2003.

[36] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point ransac for ekf filtering. application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, pp. 609–631, Septiembre 2010.

[37] S. Thrun, D. Koller, M. Montemerlo, and B.Wegbreit, "Fast slam: A factored solution to simultaneous localization and mapping," in *Proc. Nat. Conf. Artificial Intelligence*, 2002.

[38] W. . Burgard, D. F. D. Hahnel, and S. Thrun., "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements." in *Int. Conf. Intelligent Robots and Systems*, 2003.

[39] Y. Liu, A. Koller, Y. Ghahramani, S. Thrun, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters." in *Int. J. Robot. Res*, 2004.

[40] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," in *EEE Trans. Pattern Anal. Machine Intell*, 1987.

[41] M. Maimone, Y. Cheng, and L. Matthies, "Two years of visual odometry on the mars exploration rovers: Field reports," *Field Robot*, 2007.

[42] D. Scaramuzza, "Performance evaluation of 1-point-ransac visual odometry," in *Journal of Field Robotics*, 2011.

[43] K. Konolige, M. Agrawal, and J. Sol, "Large scale visual odometry for rough terrain," in *Proc. Int. Symp. Robotics Research*, 2007.

[44] J. Y. Bouguet. (2015) Camera calibration toolbox. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc

[45] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[46] R. C. Mendoza, "Mapeo y autolocalización para robots humanoides," Master's thesis, Cinvestav, 2015.

[47] A. Gee and W. Mayol-cuevas, "6d relocalisation for rgbd cameras using synthetic view regression," in *Proceedings of the British Machine Vision Conference*, 2012.

[48] Z. Cai, "Weighted nadaraya watson regression estimation," in *Statistics & Probability Letters*, 2001.

[49] RoboCup, "Robocup soccer humanoid league rules and setup for the 2015 competition in hefei," *final version june 29th ed.*, 2015.

[50] A. J. M. Tamayo, J. M. I. Zannatha, and A. E. Cobo, "Manipulation with the ah1n2 humanoid robot an underactuated/overactuated problem," *12th International Conference on Electrical Engineering, Computing Science and Automatic Control*, 2015.

[51] W. Khalil and E. Dombre, *Modeling, Identification and Control of Robots*. Kogan Page Science, 2004.

[52] D. Pieper, "The kinematics of manipulators under computer control," Ph.D. dissertation, Stanford University, 1968.

[53] J. Craig, *Introduction to Robotics: mechanics and Control*. Prentice Hall, 2005.

[54] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control. Advanced Textbooks in Control and Signal ProcessingTextbooks in Control and Signal Processing*. Springer-Verlag London Ltd., London, 2010.

[55] R. Paul, *Robot Manipulators: mathematics, Programming and Control*. MIT Press, 1981.

[56] M. Margollis, *Arduino Cookbook.*, Noviembre 2011, vol. 11.

[57] Robotis. (2010) E-manual for darwin op. [Online]. Available: http://support.robotis.com/en/product/darwin-op.htm

[58] Havimo. (2010) Havimo documentation. [Online]. Available: http://robosavvy.com/RoboSavvyPages/Support/Hamid/HaViMo2

[59] P. Vera and J. Ibarra, "Que necesita un robot humanoide para jugar futbol," in *Congreso Mexicano de Robotica*, 2014.

[60] Robotis. (2016). [Online]. Available: http://support.robotis.com/en/product/bioloid/bioloid_comp_main.htm

[61] A. Robotics. (2016). [Online]. Available: https://www.ald.softbankrobotics.com/en/cool-robots/nao

[62] Parrot. (2016). [Online]. Available: https://www.parrot.com/es/drones/parrot-bebop-2