UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Método de auto calibración basado en redundancia perceptiva para un robot manipulador

Informe de Proyecto de Grado presentado por

Lucas Camilo López Aldabalde

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Ingeniería en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Carlos Testuri
Guillermo Trinidad

Montevideo, 2 de septiembre de 2025

Agradecimientos

Quiero agradecer a mis tutores por su ayuda y guía durante la elaboración de este proyecto, y a mi familia y amigos por su apoyo incondicional a lo largo de toda esta carrera.

Resumen

La calibración es uno de los problemas más relevantes en el mundo de la robótica. A través de ella, se puede establecer una correspondencia precisa entre el modelo matemático del robot y su comportamiento real en el entorno físico. En este proyecto se propone un método de calibración cinemática, basado en redundancia perceptiva. Este proyecto se ve motivado por la técnica utilizada en fotogrametría denominada *bundle adjustment*. La propuesta consiste en un método de calibración auto-contenido, que se basa en utilizar múltiples observaciones simultáneas de un mismo objeto del entorno, obtenidas desde diferentes cámaras montadas en el propio robot. Dichas observaciones se utilizan para estimar los parámetros del modelo cinemático y mejorar la estimación de las posiciones relativas de las diferentes partes del robot. Para esto, se desarrolla un nodo de calibración implementado en *Python*, sobre el framework *Robot Operating System*, el cual obtiene las posiciones de los objetos en el mundo, procesa dicha información, y posteriormente realiza la optimización. Esta optimización se logra modelando el problema como uno de mínimos cuadrados no lineales, el cual es resuelto utilizando un método iterativo de regiones de confianza. El método propuesto se valida experimentalmente utilizando un brazo robótico, montado con dos cámaras estéreo con percepción espacial, las cuales junto con un sistema de visión artificial, se utilizan para la identificación de los objetos. El método descrito en esta investigación además de mejorar significativamente la precisión del robot a la hora de interactuar con los objetos del entorno, propone un enfoque simple y de menor costo comparado con otros métodos hallados en el relevamiento biliográfico.

Palabras clave: calibración cinemática, bundle adjustment, robótica, optimización, mínimos cuadrados

Índice general

Capítulo 1

Introducción

En la actualidad, la robótica colaborativa experimenta un crecimiento significativo. Esto implica que los robots interactúen cada vez más, en entornos variados y dinámicos. Sin embargo, diversas imprecisiones pueden surgir debido a factores como el proceso de ensamblaje, la elasticidad mecánica o incluso el diseño económico de los componentes. Por esto, es importante que sea ejecutada una rutina de calibración confiable.
En robótica, uno de los principales problemas es la calibración. A través de esta técnica, se permite conectar el mundo físico (las posiciones reales del espacio que ocupan las diferentes partes del robot) junto con el mundo lógico, es decir, el comportamiento esperado. Este proceso puede requerir de muchos recursos y esfuerzos dependiendo del entorno de trabajo y las necesidades. Los métodos de calibración pueden diferenciarse en dos grandes grupos: calibración no paramétrica y paramétrica. Por un lado, la calibración no paramétrica, en donde se agregan directamente correcciones basadas en mediciones empíricas, sin necesidad de un modelo matemático detallado del sistema. Esto permite compensar errores sistemáticos observados en el espacio de trabajo mediante tablas de corrección o modelos de error empíricos.

Por otro lado, la calibración paramétrica, es el enfoque donde el objetivo está en estimar un vector de parámetros que permite describir la geometría del robot. Uno de los modelos de calibración paramétrica más utilizados es el modelo cinemático, donde se modelan las relaciones espaciales de los diferentes enlaces del robot junto con los sensores montados en él.

En este contexto, el tipo de sensores empleados juegan un papel fundamental en la definición del enfoque a utilizar. Sensores visuales, táctiles, inerciales o de fuerza son utilizados para estimar diferentes posiciones del robot, permitiendo abordar el problema con diferentes estrategias de calibración como: la autoobservación, el autocontacto, o el uso de restricciones físicas externas. Estos tres enfoques serán explorados a lo largo del relevameinto del estado del arte. Cada una de estas estrategias propone diferentes desafíos. Por ejemplo, el manejo de colisiones en el caso del enfoque de autocontacto, o el de identificación de puntos específicos para el caso de la autoobservación.

El objetivo de este proyecto será el desarrollo de un método de calibración basado en la redundancia perceptiva. Esta propuesta es motivada por la técnica de optimización conocida como *Bundle Adjustment*, la cual es utilizada en los campos de la fotogrametría y visión por computadora. En dicha técnica, es planteado el problema como un proceso de optimización no lineal. Mediante él se estiman las coordenadas tridimensionales que describen la geometría de una escena, los parámetros del movimiento relativo entre cámaras y las características ópticas de las cámaras con las que

son capturadas las imágenes. Estas características motivan el desarrollo de la investigación para permitir la estimación de los parámetros del modelo cinemático utilizado.

Este documento se compone de seis capítulos. En los Capitulos 2 y 3, se presentan los principales conceptos teóricos necesarios para poder realizar la investigación. Para ello, diferentes fuentes bibliográficas fueron relevadas, así como investigaciones relacionadas con la temática abordada. Los resultados de estas etapas, motivaron el desarrollo de una solución propia, la cual es presentada y descrita en el Capítulo 4. En este capítulo se establece la formulación matemática del problema y se describe la arquitectura necesaria para llevar a cabo su resolución. Además, se detalla el proceso seguido para implementar la solución propuesta. En el Capítulo 5 una serie de experimentos serán presentados, para permitir validar la solución desarrollada. En el Capítulo 6 se presentan las principales conclusiones de la investigación, junto con lineamientos a trabajar en el futuro. Finalmente se incluye un Anexo, en donde se explican aquellos conceptos que no son fundamentales en el desarrollo de la investigación.

Capítulo 2

Marco Teórico

Este capítulo tiene como objetivo exponer los principales conceptos teóricos que permitan entender la investigación. Para su elaboración, se realizó un relevamiento de fuentes bibliográficas, incluyendo libros, artículos académicos y recursos digitales relacionados al tema.

En las primeras cuatro secciones del capítulo, son presentados los fundamentos matemáticos necesarios para que el problema sea formulado como uno de optimización. Entre estos fundamentos, son incluidas la definición de la técnica de mínimos cuadrados, la descripción del concepto de cadenas cinemáticas, y los elementos requeridos para que el cálculo de la cinemática directa sea realizado correctamente. En la cuarta sección, es descrita la técnica de Bundle Adjustment, la cual es considerada como la base que motiva esta investigación.

En las últimas dos secciones, son expuestos los conceptos y herramientas principales con los que es desarrollado el software que permite resolver el problema planteado. El framework correspondiente es presentado, junto con la explicación del sistema de visión artificial que es utilizado en el Capítulo de experimentación.

2.1. Mínimos cuadrados

El método de mínimos cuadrados es definido como una técnica, utilizada para ajustar un modelo a un conjunto de datos. Dado un conjunto de puntos, son determinados los valores de ciertos parámetros del modelo que permiten que una función, definida por dicho modelo, sea aproximada lo mejor posible a los datos observados. Este ajuste es logrado mediante la minimización de la suma de los errores al cuadrado entre las observaciones y los valores predichos por el modelo. La formulación general del problema de mínimos cuadrados es la siguiente:

Dado un conjunto de datos $\{(x_i, y_i)\}_{i=1}^{m}$ y un modelo $f(x, \theta)$ parametrizado por $\theta \in \mathbb{R}^n$, se busca minimizar:

$$S(\theta) = \sum_{i=1}^{m} \left(y_i - f(x_i, \theta)\right)^2 .$$

Aquí, n representa la dimensión del vector de parámetros θ, y usualmente se cumple que $n < m$; es decir, hay más observaciones que parámetros a estimar.

A partir de esta formulación, es posible distinguir dos casos según la dependencia lineal de los parametros sobre el modelo:

- **Mínimos cuadrados lineales**: cuando el modelo depende linealmente de los parámetros.
- **Mínimos cuadrados no lineales**: cuando la dependencia del modelo respecto de los parámetros es no lineal.

2.1.1. Mínimos cuadrados lineales

Un problema de mínimos cuadrados lineales se caracteriza por la dependencia lineal de los parámetros θ dentro del modelo. Un modelo donde la dependencia de los parámetros es lineal puede ser representado de la siguiente manera: Dadas n funciones linealmente independientes $\varphi_1(x), \ldots, \varphi_n(x)$, definimos al modelo f, el cual depende de manera lineal del parámetro θ como:

$$f(x, \theta) = \sum_{j=1}^{n} \theta_j \varphi_j(x). \tag{2.1}$$

Con esto en mente, dados m puntos $\{(x_i, y_i)\}_{i=1,\ldots,m} \subset \mathbb{R}^2$ y n funciones $\varphi_1(x), \ldots, \varphi_n(x)$ como arriba, podemos definir una matriz $A = (a_{ij}) \in M_{m\times n}(\mathbb{R})$, tal que

$$a_{ij} = \varphi_j(x_i). \tag{2.2}$$

Esta matriz almacena las evaluaciones de todas las funciones de base $\varphi_1, \ldots, \varphi_n$ en todos los puntos $x_1, \ldots, x_m$. Por lo tanto, si tenemos un problema lineal y queremos $f(x_i, \theta) \approx y_i$ para todo $i = 1, \ldots, m$, estamos buscando un vector de parámetros

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \tag{2.3}$$

tal que

$$\begin{cases} \theta_1\varphi_1(x_1) + \theta_2\varphi_2(x_1) + \cdots + \theta_n\varphi_n(x_1) = y_1 \\ \theta_1\varphi_1(x_2) + \theta_2\varphi_2(x_2) + \cdots + \theta_n\varphi_n(x_2) = y_2 \\ \vdots \\ \theta_1\varphi_1(x_m) + \theta_2\varphi_2(x_m) + \cdots + \theta_n\varphi_n(x_m) = y_m \end{cases} \tag{2.4}$$

En forma compacta, este sistema se puede escribir como

$$A\theta = y, \tag{2.5}$$

Donde generalmente se cumple que $m > n$. Por lo tanto, podemos definir el problema de mínimos cuadrados como sigue:

Dados $A \in \mathbb{M}_{m\times n}(\mathbb{R})$ y $y \in \mathbb{R}^m$, decimos que un vector $\theta^* \in \mathbb{R}^n$ es solución del sistema $A\theta = y$ en el sentido de mínimos cuadrados si

$$\theta^* = \underset{\theta\in\mathbb{R}^n}{\arg\min} \|A\theta - y\|_2^2. \tag{2.6}$$

Aquí, el vector residual es

$$r = A\theta^* - y \in \mathbb{R}^m,$$

que representa la diferencia entre los valores predichos $A\theta^*$ y los observados y.

La norma euclidiana $\|\cdot\|_2$ de un vector $v = (v_1, v_2, \ldots, v_m) \in \mathbb{R}^m$ se define como

$$\|v\|_2 = \sqrt{v_1^2 + v_2^2 + \cdots + v_m^2} = \sqrt{\sum_{i=1}^{m} v_i^2}.$$

En nuestro caso, la norma euclidiana del residual es

$$\|A\theta^* - y\|_2 = \sqrt{\sum_{i=1}^{m} \left((A\theta^*)_i - y_i\right)^2}.$$

Existen diferentes formas de resolver el problema de mínimos cuadrados lineales:

- **Ecuaciones Normales**: Se resuelve el sistema $A^T A\theta = A^T y$, lo que es eficiente para algunos casos, pero puede ser inestable numericamente si $A^T A$ es mal condicionado.
- **Descomposición QR**: Se factoriza A como $A = QR$, donde Q es ortogonal y R es triangular superior, permitiendo resolver un sistema triangular.
- **Descomposición en Valores Singulares (SVD)**: Se escribe $A = U\Sigma V^T$, donde U y V son matrices ortogonales y Σ es una matriz diagonal con los valores propios de A. Esta descomposición es numéricamente estable y se usa en problemas mal condicionados.

2.1.2. Mínimos Cuadrados No Lineales

Los problemas de mínimos cuadrados no lineales refieren a problemas en donde la dependencia del parámetro a estimar θ es no lineal respecto a la del modelo f. Dado que la función f es no lineal, no existe una solución analítica directa, y por lo tanto es necesario emplear métodos iterativos para su resolución. Una de las principales características de este tipo de métodos es la necesidad de una estimación inicial de los parámetros. A partir de allí, en cada iteración se transforma el problema no lineal en uno de mínimos cuadrados lineales mediante una aproximación local del modelo. Por ejemplo, en los métodos de región de confianza se construye un modelo lineal o cuadrático válido en una región determinada alrededor de la solución actual, que se resuelve para obtener un ajuste pequeño en los parámetros. Este ajuste busca reducir el residuo, es decir, la diferencia entre los valores observados y los predichos, y el proceso se repite iterativamente hasta que el valor absoluto de este residuo es menor a una tasa de tolerancia. Esto indica que se ha alcanzado una solución óptima. Existen diferentes métodos para la resolución de este tipo de problemas no lineales, entre ellos:

- **Métodos Basados en la Evaluación de la Función**: Estos métodos no requieren el cálculo de gradientes, siendo adecuados para funciones no diferenciables o mal condicionadas. Se basan en la exploración de la superficie de la función sin utilizar información sobre su pendiente.

- **Métodos Basados en el Gradiente**: Utilizan información del gradiente para dirigir la búsqueda del mínimo de manera más eficiente. Son adecuados para funciones suaves y bien comportadas, logrando generalmente una convergencia más rápida que los métodos sin gradientes.
- **Métodos de Región de Confianza**: Construyen modelos locales de la función objetivo y los minimizan dentro de una región de confianza, ajustando el tamaño de la región en función del progreso de la optimización. Son útiles en problemas donde la Hessiana es difícil de calcular o aproximar.
- **Métodos de Programación Cuadrática Secuencial**: Diseñados para manejar restricciones de igualdad y desigualdad en la optimización, combinan aproximaciones cuadráticas de la función objetivo con restricciones lineales o no lineales para guiar la búsqueda hacia soluciones factibles.

2.2. Cadenas cinemáticas

Según la definición presentada en ScienceDirect Topics (s.f.), una cadena cinemática refiere a una serie de eslabones conectados entre sí por medio de articulaciones, permitiendo el movimiento relativo entre los eslabones y formando una estructura que puede adoptar diferentes configuraciones. Las articulaciones pueden ser rotacionales, permitiendo rotación alrededor de un eje fijo especificado por el ángulo de rotación (θ), o prismáticas, permitiendo movimiento lineal especificado por la distancia de desplazamiento (d).

Los grados de libertad (DoF, por sus siglas en inglés) en un robot se refieren al número de movimientos independientes que el robot puede realizar. Cada grado de libertad corresponde a un modo de movimiento posible, ya sea traslacional (movimiento a lo largo de un eje) o rotacional (movimiento alrededor de un eje). En términos más simples, los DoF son las diferentes maneras en las que un robot puede mover o posicionar sus partes.

2.3. Cuaterniones y Cinemática Directa

Los cuaterniones son una extensión de los números complejos y se utilizan para representar rotaciones en el espacio tridimensional. Un cuaternión se define como $q = w + xi + yj + zk$, donde w, x, y, y z son números reales, e i, j, k son las unidades imaginarias que satisfacen las relaciones $i^2 = j^2 = k^2 = ijk = -1$. Los cuaterniones presentan varias ventajas con respecto a los ángulos de Euler, los cuales representan rotaciones en tres dimensiones mediante tres rotaciones sucesivas alrededor de los ejes coordenados Z, Y y X. A diferencia de los ángulos de Euler, que están sujetos al problema de bloqueo de cardán, los cuaterniones no sufren de este problema debido a su representación matemática robusta. Además, los cuaterniones permiten interpolaciones suaves entre rotaciones y son más eficientes computacionalmente. Este tipo de representación será utilizada en la resolución del problema, para poder representar las rotaciones relativas entre marcos de referencia consecutivos.

Para obtener las rotaciones elementales alrededor de los ejes X, Y y Z a partir de un cuaternión, es necesario considerar su vector de rotación y el ángulo asociado. Las rotaciones en torno a cada eje pueden representarse mediante cuaterniones particulares, definidos en función de los ángulos α, β y γ, correspondientes a los ejes X, Y y Z, respectivamente.

$$\text{rot}_X(\alpha) = \left(\cos\left(\frac{\alpha}{2}\right), \sin\left(\frac{\alpha}{2}\right), 0, 0\right)^T$$

$$\text{rot}_Y(\beta) = \left(\cos\left(\frac{\beta}{2}\right), 0, \sin\left(\frac{\beta}{2}\right), 0\right)^T$$

$$\text{rot}_Z(\gamma) = \left(\cos\left(\frac{\gamma}{2}\right), 0, 0, \sin\left(\frac{\gamma}{2}\right)\right)^T$$

Para construir una matriz de rotación a partir de un cuaternión $q = w + xi + yj + zk$, se utiliza la siguiente matriz de expresiones:

$$R = \begin{bmatrix} 1 - 2(y^2 + z^2) & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 1 - 2(x^2 + z^2) & 2(yz - wx) \\ 2(xz - wy) & 2(yz + wx) & 1 - 2(x^2 + y^2) \end{bmatrix}$$

Esta matriz transforma las coordenadas de un punto en un sistema de referencia a otro sistema de referencia que está rotado con respecto al primero.

En el contexto de la cinemática directa, se busca determinar la posición y orientación del efector final de un robot en relación con un sistema de referencia base. La cinemática directa se basa en el producto de transformaciones homogéneas, donde cada eslabón del robot se describe mediante una matriz homogénea que incluye tanto una rotación como una traslación.

Cada eslabón del robot se describe mediante una matriz homogénea T_i que se puede expresar como:

$$T_i = \begin{bmatrix} R_i & d_i \\ 0 & 1 \end{bmatrix}$$

donde R_i es la matriz de rotación que describe la orientación del eslabón i, y d_i es el vector de traslación que describe la posición del eslabón i con respecto al eslabón $i-1$. La matriz homogénea T_i combina la rotación y la traslación en una sola matriz, facilitando la descripción de la configuración del robot.

La matriz de rotación R_i para el eslabón i se puede construir utilizando el cuaternión que describe la orientación del eslabón. Luego, al combinar todas las matrices homogéneas de los eslabones individuales mediante multiplicación de matrices, se obtiene la matriz de transformación global que describe la posición y orientación del efector final en el sistema de referencia base:

$$T_{total} = T_1 \cdot T_2 \cdot \ldots \cdot T_n$$

donde T_{total} es la matriz homogénea final que representa la posición y orientación del efector final del robot.

2.4. Bundle Adjustment

Bundle Adjustment (BA) es una técnica de optimización no lineal utilizada en visión por computadora y fotogrametría para refinar reconstrucciones tridimensionales. El objetivo principal es obtener estimaciones óptimas tanto de la estructura 3D de una escena como de los parámetros de las cámaras (posiciones, orientaciones y calibraciones) que capturaron las imágenes. Esto se logra minimizando una función de costo que cuantifica el error de reproyección, es decir, la diferencia entre las

proyecciones observadas y las proyectadas de los puntos 3D en las imágenes (Triggs, McLauchlan, Hartley, y Fitzgibbon, 2000). Generalmente, este problema se formula como una minimización de mínimos cuadrados no lineales. La Figura 2.1 muestra una ilustración del problema presentado. En la misma se puede ver un mismo objeto (pelota de fútbol) desde diferentes posiciones. De cada imagen se extraen puntos de interés, que serán utilizados en la minimización.

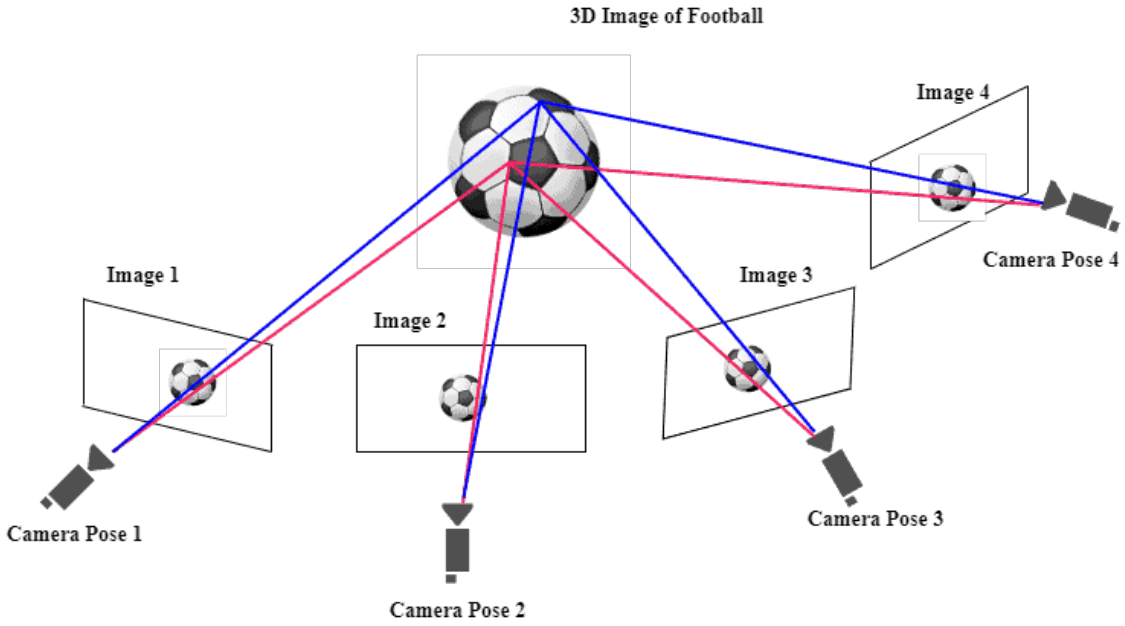


Figura 2.1: Ilustración de la técnica de Bundle Adjustment. A partir de diferentes vistas de un mismo objeto, se reconstruye su modelo tridimensional. Imagen extraída de Baeldung on Computer Science (2024).

2.5. Robot Operating System (ROS)

El Sistema Operativo de Robots (ROS por su sigla en inglés) es un framework diseñado para el desarrollo de software en robótica. ROS no es un sistema operativo convencional; sino que es un middleware que opera sobre un sistema operativo. Proporciona un conjunto de bibliotecas y herramientas que permiten crear aplicaciones robóticas complejas y robustas. Esto es posible ya que cuenta con cualidades como la abstracción de hardware, controladores de dispositivos, comunicación entre procesos, gestión de paquetes, entre otros.

Uno de los elementos clave de ROS es su arquitectura distribuida. Utiliza una estructura similar a un grafo, en donde los nodos son procesos individuales responsables de tareas específicas. Estos nodos pueden ser creados utilizando las bibliotecas de ROS en lenguajes como C++ o Python. Cada nodo normalmente realiza una función particular, como controlar un sensor, procesar datos o ejecutar algoritmos.

Los nodos se comunican entre sí utilizando un modelo de publicación-suscripción a través de tópicos. Los tópicos son buses nombrados, donde los nodos pueden publicar mensajes (publicadores)

o suscribirse para recibir mensajes (suscriptores). Por ejemplo, un nodo de cámara podría publicar imágenes en un tópico 'InfoCamara', mientras que un nodo de procesamiento de imágenes se suscribe a este tópico para recibir y procesar las imágenes.

ROS utiliza una jerarquía de sistema de archivos específica para organizar el código en paquetes. Un paquete generalmente contiene nodos, archivos de configuración, archivos de lanzamiento (scripts para iniciar varios nodos) y otros recursos. Esta estructura fomenta la modularidad y reutilización.

Además, ROS permite registrar y almacenar mensajes publicados en los tópicos mediante archivos llamados bags (con extensión .bag). Estos archivos pueden reproducirse posteriormente, emulando la transmisión original de los datos, lo cual resulta útil para depuración, análisis y experimentación sin necesidad de disponer del hardware en tiempo real.

ROS proporciona un conjunto de herramientas para visualización y simulación, que son claves en el desarrollo de robótica. Algunos ejemplos son:

- **RQT:** interfaz gráfica de usuario dentro de ROS que ofrece una colección de complementos para visualización, inspección de datos e interacción con nodos de ROS. Ofrece una interfaz personalizable, permitiendo a los usuarios visualizar datos, manipular parámetros e interactuar con múltiples nodos simultáneamente.

- **RVIZ:** herramienta de visualización que permite a los usuarios ver diversos datos, como salidas de sensores o modelos de robots, en una interfaz gráfica.

Estas herramientas ofrecen funcionalidades cruciales para el desarrollo, prueba y despliegue de sistemas robóticos en diferentes dominios.

El formato URDF (Unified Robot Description Format) es un estándar utilizado en ROS para describir la estructura y los componentes de un robot. URDF es un archivo XML que define la geometría, cinemática, dinámica, y otros aspectos del robot, como las articulaciones y los enlaces. Este archivo describe cómo las diferentes partes del robot están conectadas y cómo se mueven, proporcionando una representación detallada del modelo físico del robot.

RVIZ utiliza archivos URDF para mostrar la representación gráfica del robot en su interfaz. Al cargar un archivo URDF en RViz, el sistema puede renderizar el modelo tridimensional del robot, permitiendo a los desarrolladores ver cómo el robot se mueve y se comporta. Esta visualización es esencial para verificar la configuración del robot, ajustar parámetros de movimiento y realizar pruebas sin necesidad de interactuar directamente con el hardware físico.

2.6. Detección de Objetos

La detección de objetos mediante visión artificial consiste en identificar y localizar objetos en imágenes o videos. Los algoritmos de detección de objetos se pueden dividir en dos grandes categorías: los algoritmos de detección en una sola etapa y en dos etapas . Los detectores de dos etapas realizan una primera pasada para generar propuestas de regiones y una segunda para refinar las predicciones, logrando mayor precisión a costa de un mayor costo computacional. Por otro lado, los detectores de una sola etapa realizan una única pasada sobre la imagen para predecir la presencia y ubicación de objetos, lo que los hace computacionalmente eficientes y adecuados para aplicaciones en tiempo real, aunque suelen ser menos precisos, especialmente con objetos pequeños. YOLO (You Only Look Once, (Ultralytics, 2024)) entra dentro de los modelos de detección de objetos de una sola etapa y es conocido por su velocidad y precisión.

Además de detectar y localizar objetos, en muchas aplicaciones es crucial estimar la distancia entre la cámara y los objetos detectados. YOLO puede integrarse con técnicas de estimación de profundidad para lograr este objetivo.

Capítulo 3

Relevamiento del Estado del Arte

A continuación, son presentadas algunas investigaciones relacionadas con la temática. El objetivo es comprender cómo es abordado el problema por distintos autores, así como identificar las principales dificultades y procedimientos comúnmente utilizados. Para ello, son relevados diferentes artículos científicos. La selección de trabajos es realizada mediante búsquedas específicas por palabras clave asociadas al tema, junto con la elección de aquellos artículos que son citados con mayor frecuencia.

3.1. Calibración

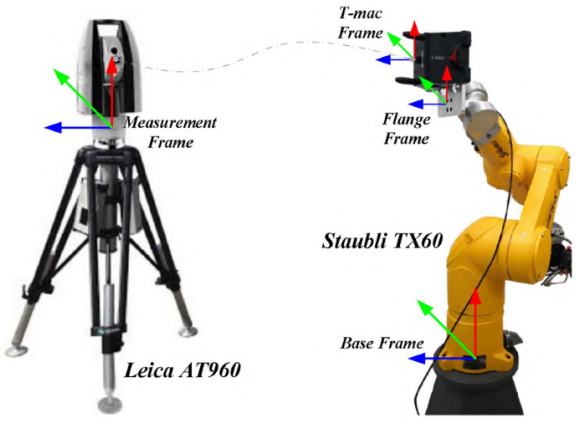
Roth, Mooring, y Ravani (1987) presentan la calibración como medio para mejorar la precisión del posicionamiento del robot mediante software, en lugar de cambiar la estructura mecánica o el diseño del propio robot. Para ellos, a nivel general, la calibración puede clasificarse en dos tipos: calibración paramétrica (basada en modelos) y la calibración no paramétrica. La mayoría de los trabajos sobre calibración paramétrica basada en modelos se han centrado en la calibración basada en modelos cinemáticos.

En la calibración basada en modelos cinemáticos, se aplican factores geométricos para identificar los parámetros del modelo. En general, la calibración basada en modelos cinemáticos se considera un método de calibración global que mejora la precisión del robot en todo el volumen de su espacio de trabajo. La calibración cinemática consta de cuatro pasos secuenciales:

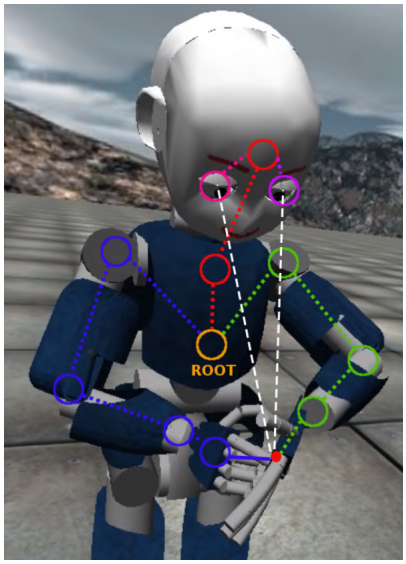
Primero la etapa de **modelado**. Un modelo cinemático es una descripción matemática de la geometría y el movimiento de un robot. Existen varios enfoques para desarrollar el modelo cinemático de un manipulador robótico. El método más popular fue establecido por Denavit y Hartenberg (Ver Anexo A.1. A partir de aquí nos referiremos a este modelo como el modelo DH).

Luego lo sigue la etapa de **medición**. Esta fase implica la detección, dentro del espacio de trabajo, de las posiciones del efector final o herramienta del robot. Las posiciones reales medidas del efector final se comparan luego con las posiciones predichas por el modelo teórico, y así de esta manera se logra identificar la inexactitud del modelo. Para ellos, la medición es la fase más difícil y que consume más tiempo en la calibración. Existen diferentes métodos y dispositivos de medición. Estos incluyen sensores acústicos, visuales, así como también herramientas externas como láseres. Los diferentes dispositivos utilizados varían considerablemente en cuanto a precisión, facilidad de uso y costo. Además, la recolección de datos consume mucho tiempo y es difícil de automatizar. Por

lo tanto, los procedimientos de configuración y medición requieren mucha intervención humana, lo cual presenta serias desventajas.
La fase de **identificación** de parámetros implica el uso de métodos numéricos. Estos métodos deben ser lo suficientemente fiables como para alcanzar una solución manteniendo un nivel razonable de confianza en los valores de los parámetros identificados. En esta fase, se identifican los errores en los parámetros cinemáticos, utilizando el criterio de mínimos cuadrados. Se sabe que la identificación cinemática es un procedimiento estándar de optimización por mínimos cuadrados, ya sea lineal o no lineal.
Por último, la etapa de **compensación**. Esta es la etapa final y decisiva en la calibración cinemática de robots, y consiste en la implementación del nuevo modelo del robot. A veces se la denomina la etapa de corrección. Según Hollerbach, Khalil, y Gautier (2016) existen dos enfoques dentro de la calibración cinemática: en primer lugar, los enfoques de **lazo abierto (open-loop)**, donde el manipulador no está en contacto físico con el entorno. Por esto, es necesario utilizar un sistema de metrología externo para medir los componentes de la pose del robot. Por el otro lado, los enfoques de **lazo cerrado (closed-loop)**, donde se utilizan restricciones físicas sobre la pose del robot que sustituyen las mediciones externas. En la Figura 3.1 puede verse un ejemplo de cada enfoque. En cada uno de estos casos, la calibración puede ser utilizando cadenas cinemáticas simples o cadenas cinemáticas múltiples.



(a) Ejemplo de un enfoque open-loop. Donde un sensor externo al robot (en este caso un Leica AT960) mide la ubicación de alguna parte del robot. Imagen extraída de ? (?).



(b) Ejemplo de enfoque closed-loop. En este caso es una simulación del robot ICub, donde él mismo toca una parte de su cuerpo, cerrando así su cadena cinemática, al igual que cuando la mira. Imagen extraída de Stepanova y cols. (2019)

Figura 3.1: Ilustración de los dos tipos de enfoque de calibración cinemática: Open-loop y Closed-loop.

Como se explicó anteriormente las mediciones dentro de un modelo cinemático pueden variar según los tipos de sensores. Esto permite definir diferentes tipos de cadenas cinemáticas, cada una asociada al tipo de sensor. Por ejemplo, con el uso de cámaras como parte de los sensores de medición, se pueden definir cadenas cinemáticas basadas en la auto-observación, donde se busca identificar de manera visual alguna de las partes del robot. Con sensores como pieles artificiales o sensores de torque, se puede implementar el auto-contacto, o el contacto con algún objeto del entorno. A partir de estas cadenas cinemáticas simples se pueden construir cadenas cinemáticas múltiples, donde se combina la información de múltiples sensores en una única función de costo. Según investigaciones previas (como Stepanova y cols. (2022) o Stepanova y cols. (2019)) el uso de múltiples cadenas cinemáticas mejora los resultados en la precisión de la calibración, en comparación al uso individual de las cadenas simples. A pesar de esto, en esta investigación nos concentraremos en aquellos trabajos que sean abordados utilizando alguna de las siguientes cadenas cinemáticas simples: auto-contacto, restricciones planares o auto-observación.

3.1.1. Cadenas cinemáticas simples

Auto-contacto

El autocontacto corresponde a una forma de calibración cinemática de tipo de enlace cerrado. Este tipo de calibración requiere de un entorno de trabajo más complejo como robots humanoides o con brazos duales. Además, estos deben contar con equipamiento sensorial y motores adecuados tanto para poder identificar de manera correcta el contacto, como para poder desplazar los puntos de colisión en el espacio con el mayor control posible. Una posibilidad es utilizar pieles electrónicas artificiales que cubren áreas específicas o incluso cuerpos completos de robots (Figura 3.2b). En estos casos, si se dispone de una calibración espacial precisa de la piel, entonces se pueden medir las posiciones donde ocurre el contacto en cada una de las dos cadenas cinemáticas que se intersecan. Este tipo de calibración es la explorada por Stepanova y cols. (2019) utilizando la plataforma ICub (ver Figura 3.2b), donde el contacto ocurre directamente en los efectores finales (el efector final del brazo derecho se traslada desde la palma hasta la punta del dedo índice usando una transformación fija).

Para optimizar los parámetros que describen esta cadena, se minimiza la distancia entre las posiciones estimadas en el espacio 3D de los efectores finales de los brazos izquierdo y derecho.

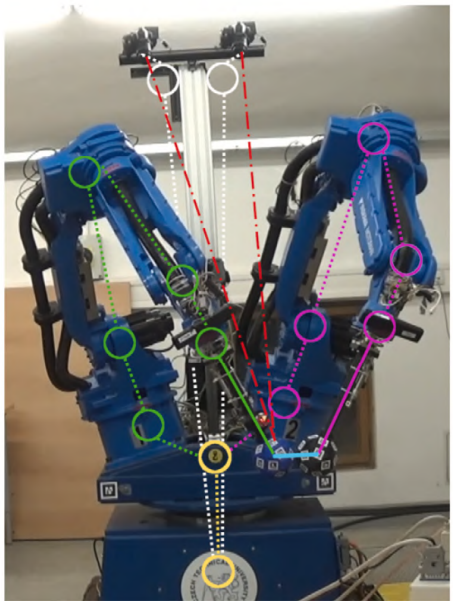
Para cada objetivo, utilizando un solucionador cartesiano y un controlador, el iCub mueve la mano izquierda (con el efector final en la palma) al punto especificado. Luego mueve la mano derecha, con el efector final en la punta del dedo índice, al mismo punto.

Stepanova y cols. (2022) explora este enfoque para dos brazos robóticos Yaskawa R750 como puede verse en la Figura 3.2a. En esta investigación el desafío que tenían era el de lograr la colisión entre los efectores finales del robot (los cuales fueron modificados y son representados como esferas). Para lograrlo, ambos efectores son llevados a la misma posición.

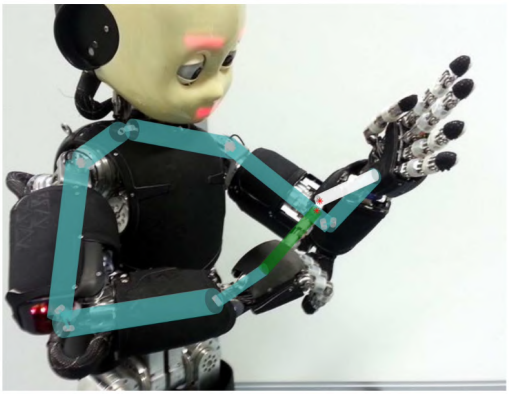
Estas restricciones se expresan como:

$$\sqrt{(x_i^l - x_i^r)^2 + (y_i^l - y_i^r)^2 + (z_i^l - z_i^r)^2} = 2r \tag{8}$$

donde (x_i^r, y_i^r, z_i^r) y (x_i^l, y_i^l, z_i^l) son los centros de los efectores del brazo derecho e izquierdo, respectivamente, calculados mediante cinemática directa para la configuración articular $\boldsymbol{\theta}_i$.



(a) Setup de trabajo de Stepanova y cols. (2022)



(b) Robot ICub, junto con su modelo cinemático. Imagen extraída de Stepanova y cols. (2019)

Figura 3.2: Ejemplos de colisiones, utilizadas en la calibración por auto-contacto

Restricciones planares

El enfoque de calibración utilizando restricciones planares se basa en el contacto físico del efector final con un plano del entorno. Este tipo de calibración tampoco requiere instrumentos de medición externos, por lo que también refiere a un enfoque de lazo cerrado. Cuando el brazo robótico (izquierdo o derecho) toca una superficie, dicha interacción física genera restricciones geométricas que permiten estimar los parámetros del modelo cinemático del robot. Hay muchas investigaciones que hacen uso de esta técnica para calibrar, entre ellas Stepanova y cols. (2022) para un robot industrial Yaskawa-Motoman MA1400, como por Ikits y Hollerbach (1997) para un PUMA 560. Lo que se busca es minimizar la distancia entre la posición del efector (calculada por cinemática directa) y el plano de contacto. En este tipo de enfoque, se requiere algún tipo de sensado de fuerza o conocimiento del punto de contacto. La Figura 3.3 muestra un ejemplo de este tipo de calibración.

Existen dos alternativas clásicas para abordar este problema.

- **Ecuación del plano:** En este caso se conoce una ecuación (parcial o total) del plano con el que interactúa el robot. El enfoque más directo asume que los puntos extremos del efector final (x, y, z) satisfacen la ecuación del plano:

$$ax + by + cz = d \tag{3.1}$$

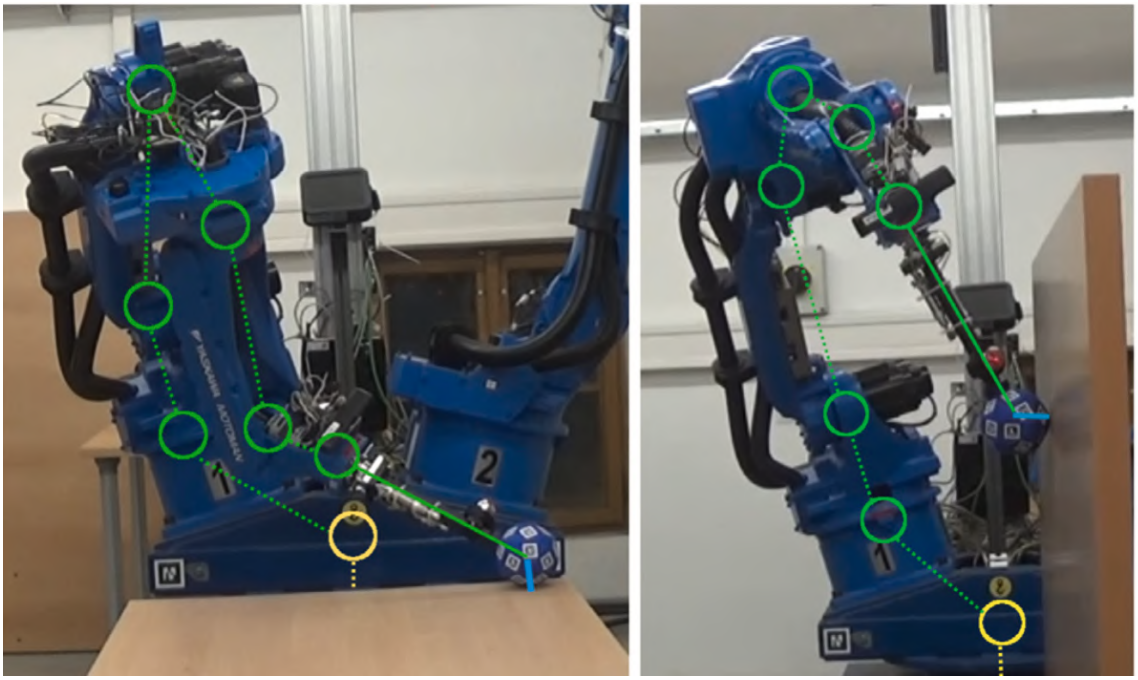


Figura 3.3: Yaskawa-Motoman MA1400 interactuando con un plano horizontal y vertical. Imágen extraída de Stepanova y cols. (2022).

donde a, b, c, d son los coeficientes generalizados del plano. Esto generalmente no es cierto y por lo tanto se busca minimizar la siguiente función objetivo:

$$S = \sum_{i=1}^{P} \left([a \ b \ c] \cdot \mathbf{p}_i^c \right)^2 \tag{3.2}$$

donde $\mathbf{p}_i^c$ representa la posición (x, y, z) del efector final para la pose i. La principal dificultad con este enfoque está en que hay que conocer la ecuación del plano en el espacio.

- **Ecuaciones normales:** Este caso se basa en el cálculo de productos vectoriales entre un conjunto de vectores ubicados en el mismo plano, los cuales deberían generar las normales del plano. Tres puntos determinan un plano, y de ellos se obtienen dos vectores de diferencia independientes, a partir de los cuales puede extraerse la normal del plano. A partir del conjunto de productos vectoriales entre estas normales locales ("pequeños planos"), se puede derivar un sistema de ecuaciones lineales para la identificación de parámetros. Este caso tiene una complejidad operativa, ya que es necesario tener cuidado con la selección de poses, ya que fácilmente se pueden obtener configuraciones colineales, lo que dificulta o imposibilita la identificación.

Auto-observacion

La calibración por autoobservación consiste en utilizar cámaras montadas en el propio robot para observar visualmente algún punto del espacio, para estimar así los parámetros del modelo cinemático. Esta técnica permite cerrar el lazo de calibración sin necesidad de sensores externos. La clave en este enfoque es el de poder detectar los diferentes elementos utilizados para la calibración de manera visual. Existen diversas investigaciones explotando esta técnica. Por ejemplo, Švaco, Šekoranja, Šuligoj, y Jerbić (2014) realizan una investigación para un KUKA KR 6 R900 con un sistema de visión estéreo conformado por dos cámaras CCD, dispuestas de manera perpendicular, como puede verse en la Figura 3.4. La disposición de las cámaras permite conformar un un punto central (TCP) virtual, el cual utilizan en su método de calibración guiado. El TCP virtual es guiado manualmente hacia diferentes posiciones y orientaciones deseadas, donde se encuentra localizada una esfera de calibración negra. Luego, a través de un algoritmo de visión artificial ajusta la posición del robot con respecto a la esfera en el marco de referencia de la cámara. Los estados articulares del robot se registran para ser utilizados posteriormente en el algoritmo de optimización de parámetros. Una vez que se han obtenido suficientes mediciones, se comparan los estados de las articulaciones registradas para cada punto de calibración.

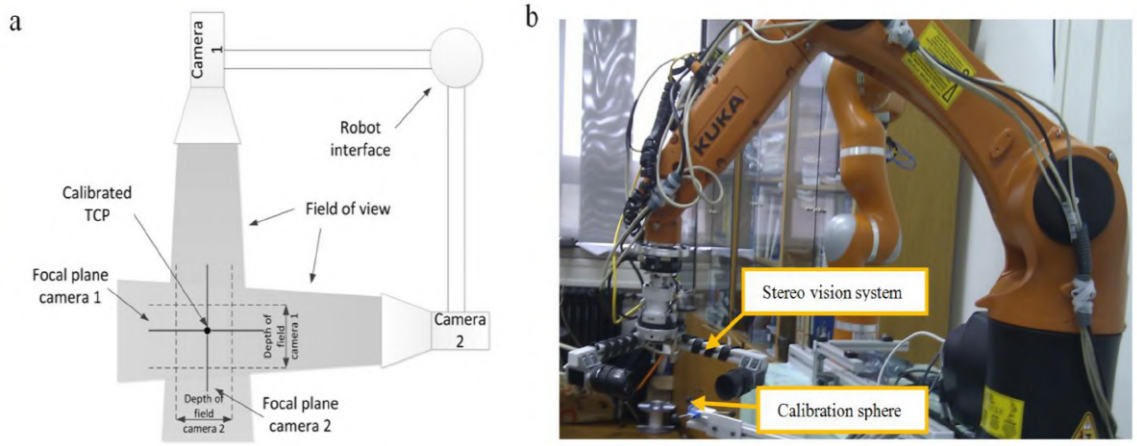
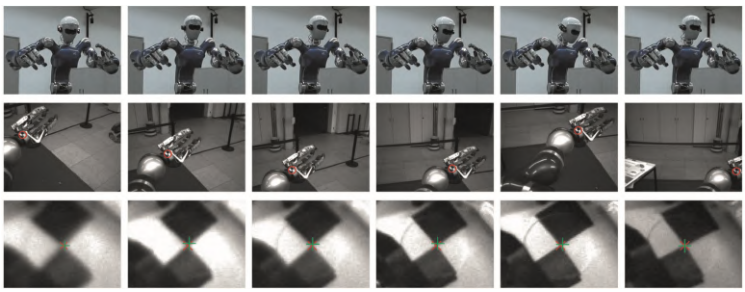


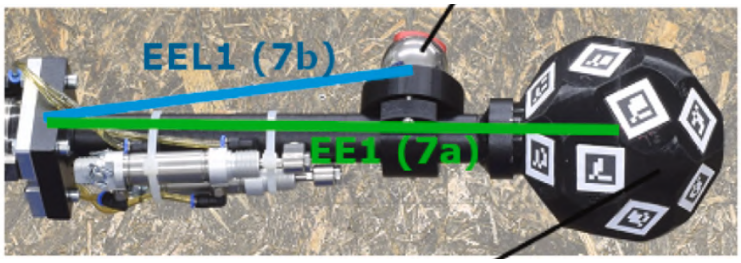
Figura 3.4: (a) Vista esquemática del sistema de visión estéreo (b) Configuración experimental del procedimiento de calibración. Imagen extraída de Švaco y cols. (2014).

Birbach, Frese, y Bäuml (2015) también exploran esta técnica, pero en este caso para un robot humanoide Agile Justin. Se propone un enfoque de detección basado en la coincidencia de plantillas artificiales para localizar características puntuales en las muñecas del robot. El proceso consta de dos etapas: una selección global de candidatos y un refinamiento local, lo cual permite detectar las características de manera precisa. Debido a la presencia de estas etiquetas adheridas a su superficie, es que puede lograr una calibración automática y autocontenida, como puede verse en la Figura 3.5a.
Stepanova y cols. (2022) utiliza una técnica similar, al definir su propio efector final (Figura 3.5b) con marcas Aruco que le permitiesen identificar dicha parte del cuerpo del robot. Otro tipo de recurso muy utilizado en este tipo de enfoques es el del uso de tableros de ajedrez. Esto se debe al contraste claro de su cuadrícula, lo que facilita la identificación precisa de los vértices (interseccio-

nes entre cuadros blancos y negros) en las imágenes capturadas. Estos patrones permiten extraer características geométricas con alta precisión y eficiencia. Kastner, Röfer, y Laue (2015) utilizan esta técnica para realizar la calibración de un robot NAO. Como se dijo anteriormente, la clave al utilizar esta técnica es el de identificar las partes del robot (o su entorno) de manera efectiva. Claramente el uso de marcadores especiales resuelve esta parte, sin embargo introduce dos dificultades. La primera, es la complejidad operativa de generar de manera personalizada identificadores. En el caso del tablero de ajedrez, además hay que colocarlo de alguna manera visible, por ejemplo, Pradeep, Konolige, y Berger (2014) guían manualmente el tablero para que el mismo sea visible por su robot. La segunda es que al tener un identificador colocado en una zona específica, restringe las posiciones de calibración para el robot, ya que si en alguna pose se obstruye este marcador, el proceso de calibración no podrá llevarse a cabo correctamente.



(a) Capturas de una única ejecución del procedimiento de calibración observando una postura del brazo. Primera fila: Vista externa del procedimiento de calibración. Segunda fila: Vista desde la cámara izquierda. El marcador adherido (característica puntual) está resaltado con un círculo rojo. Tercera fila: Acercamiento de la característica montada en la muñeca en la imagen de la cámara izquierda montada en la cabeza. El centro detectado se muestra con una cruz roja (×) y el centro predicho con una cruz verde (+). Imagen extraída de Birbach y cols. (2015)



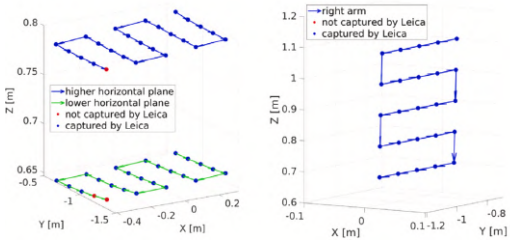
(b) Diseño del efector final personalizado de Stepanova y cols. (2019)

Figura 3.5: Ejemplos de efectores personalizados utilizados en la calibración por auto-observación

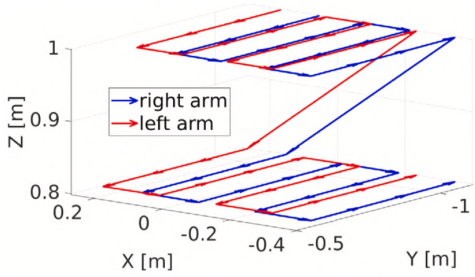
3.2. Uso de datasets en calibración robótica

Una de las principales formas de abordar este problema consiste en el uso de conjuntos de datos (datasets) recolectados a partir de escenas y configuraciones controladas. Este método permite registrar de forma sistemática y controlada, la información sensorial y articular del robot mientras interactúa con su entorno o consigo mismo en un momento dado, para posteriormente, ser utilizada como fuente de verdad en la calibración. El procedimiento general consiste en diseñar configuraciones del robot que cubren una amplia variedad de posturas posibles, y registrar, en cada una, la información necesaria para posteriormente ejecutar algoritmos de optimización.

La construcción de estos conjuntos de datos puede abordarse desde múltiples enfoques, dependiendo del tipo de restricción que se utilice. Por ejemplo, Stepanova y cols. (2022) propone un dataset compuesto por distintos escenarios: auto-contacto entre efectores, contacto con planos horizontales (superior e inferior), contacto con un plano vertical tipo "pared" (En la Figura 3.6 puede verse una grilla con las poses utilizadas para generar estos datasets), y registros de precisión obtenidos mediante un rastreador láser (Este último refiere a un enfoque de lazo-abierto). Para cada pose, se capturan imágenes y se registra información como la configuración articular completa del robot, la posición del marcador en imagen, y (cuando aplica) la posición exacta del retroreflector medida con láser.



(a) Posiciones de los centros de los efectores finales en los conjuntos de datos de contacto con planos horizontales (izquierda, plano superior en azul, plano inferior en verde) y en el conjunto de datos de contacto con plano vertical (derecha), con la indicación de si la postura fue registrada por el rastreador láser (azul) o no (rojo). Imagen extraída de Stepanova y cols. (2022)



(b) Posiciones de los centros de los efectores finales durante el experimento de autocontacto. El extremo de cada flecha indica una posición del centro del efector final en distintas posturas (rojo: brazo izquierdo, azul: brazo derecho). La distancia entre las posiciones del brazo izquierdo y derecho está determinada por el doble del radio de los efectores en contacto. Imagen extraída de Stepanova y cols. (2022)

Figura 3.6: Métodos para la generación de los datasets.

Stepanova y cols. (2019), generan un dataset para el robot humanoide iCub donde ambos brazos se posicionan en contacto físico mientras el robot observa visualmente el punto de contacto. Los puntos objetivo se eligen dentro de un volumen cúbico frente al robot y se alcanzan mediante un controlador cartesiano. El efector izquierdo (la palma) se lleva al punto objetivo y el efector derecho (punta del dedo índice) se aproxima con una restricción de orientación. El vector de datos asociado a cada configuración contiene las coordenadas del punto objetivo, las posiciones alcanzadas por

cada efector, y los valores articulares del torso, brazos, cuello y ojos. Una alternativa diferente es la presentada en Pradeep y cols. (2014), donde se utiliza un tablero de ajedrez como referencia visual para calibrar la cámara y cinemática del robot NAO. En este caso, el robot se acuesta sobre su espalda y se posiciona en 24 posturas diferentes, observando el tablero desde tres posturas de cabeza distintas. En cada imagen, se detectan las coordenadas en 2D de los vértices del tablero mediante una combinación de algoritmos de detección de esquinas y refinamiento subpíxel. Esta estrategia, aunque visualmente más sencilla, es altamente efectiva gracias al fuerte contraste de las casillas del tablero. Por último, Roncone, Hoffmann, Pattacini, y Metta (2014) presenta un enfoque basado en sensores de piel, donde el contacto físico entre el antebrazo izquierdo y el dedo derecho se utiliza como fuente de datos. El operador inicia el proceso tocando el antebrazo del robot, y luego el propio robot ejecuta el movimiento de autocontacto, como puede verse en la Figura 3.7. Si el contacto es detectado por los sensores, se registra la posición del punto de contacto junto con la configuración articular. Generar estos datasets no es una tarea trivial, sin importar el tipo de robot utilizado. Para esta tarea es necesario trabajar sobre entornos controlados, así como una buena precisión en los movimientos del robot en la etapa de recolección de datos, ya sea de manera guiada, o automática partiendo de una buena calibración inicial.

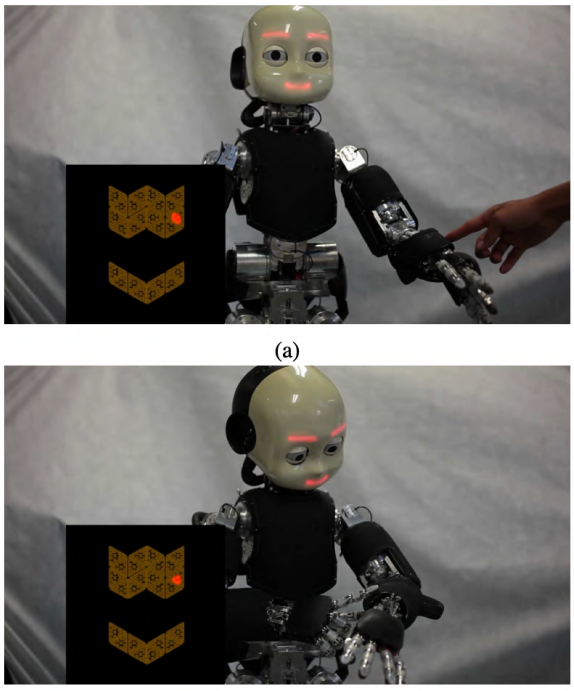


Figura 3.7: (a) El iCub es tocado en su antebrazo izquierdo por el experimentador; la activación correspondiente de la piel se muestra a la izquierda. (b) El iCub toca el punto previamente estimulado utilizando el dedo índice del brazo contralateral. Imagen extraída de (Roncone y cols., 2014).

Capítulo 4

Solución propuesta

En este capítulo se presenta tanto el diseño como la implementación desarrollada para abordar el problema planteado. La solución es diseñada teniendo en cuenta los conceptos expuestos en el marco teórico y los aprendizajes obtenidos del relevamiento del estado del arte. Se detallan las decisiones de diseño, las herramientas empleadas y los métodos implementados.

Uno de los aprendizajes extraídos del análisis del estado del arte es el alto costo de algunos enfoques, ya sea en términos estructurales, al requerir más de un brazo robótico para implementar el auto-contacto, o en términos metodológicos, como la necesidad de utilizar marcas específicas en casos de autoobservación, o la incorporación de formulaciones matemáticas del entorno para el caso de restricciones planares. Asimismo, el uso de conjuntos de datos en los procedimientos de calibración pone en evidencia la complejidad de estas estrategias.

Otra de las conclusiones más relevantes es que la clave en los distintos métodos de calibración radica en la redundancia de información sobre los distintos puntos del entorno. Considerando este aspecto, se propone un método más simple y menos costoso. Para ello, se emplean dos cámaras ubicadas en diferentes partes de la cadena cinemática del robot, con el objetivo de obtener la redundancia necesaria. Además, se utiliza un sistema de visión artificial que permite identificar objetos del entorno, eliminando la necesidad de emplear objetos con marcas especiales. Este método también resulta más económico que otras alternativas del estado del arte, al requerir únicamente dos cámaras y no depender de robots equipados con sensores complejos.

4.1. Diseño

4.1.1. Formulación matemática del problema

Comenzamos con la definición matemática del problema. Para ello, se define la cadena cinemática del robot como un grafo dirigido $G = (V, E)$, donde V representa el conjunto de nodos. Cada nodo puede corresponder a uno de los siguientes elementos:

- **Articulaciones del robot**: Representan los grados de libertad y permiten el movimiento entre los distintos eslabones del robot.
- **Puntos de percepción**: Coordenadas en el espacio que indican la posición de objetos detectados por las cámaras del robot.

- **Soportes físicos**: Elementos estructurales que proporcionan estabilidad al robot, como bases, columnas o anclajes. Estos pueden modelarse como cuerpos rígidos adicionales conectados mediante uniones fijas al resto de la estructura.

- **Puntos lógicos**: Referencias virtuales utilizadas para tareas específicas. Su transformación espacial es nula.

Mientras que E corresponde a las aristas que conectan los diferentes nodos. Cada una de estas aristas representa una transformación espacial entre pares de nodos contiguos. Dicha transformación contempla tanto coordenadas de traslación como de rotación. Cuando alguno de los nodos es un nodo lógico, la transformación espacial es nula.

Una de las restricciones sobre G es que debe ser un árbol dirigido, donde las hojas representan los diferentes puntos vistos por las cámaras y la raíz corresponde al origen del sistema de referencia. En la Figura 4.1 se presenta un grafo de ejemplo válido para nuestro problema.

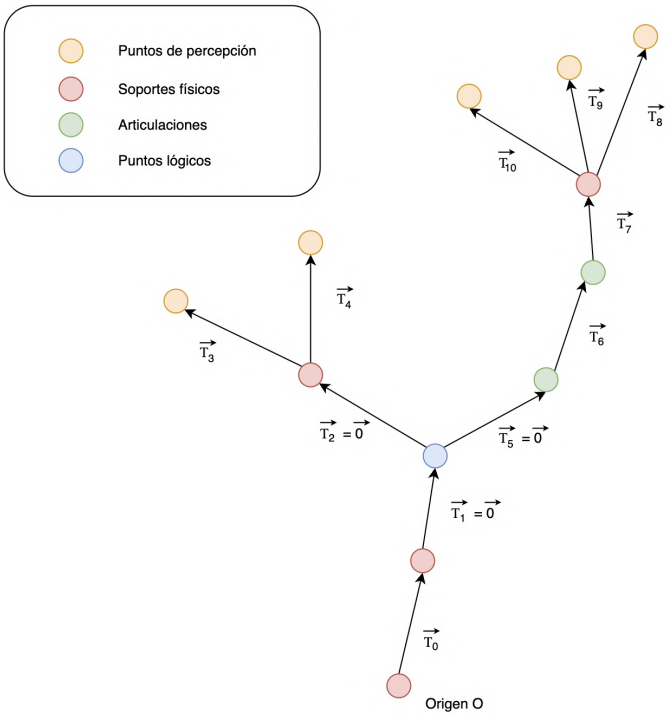


Figura 4.1: Ejemplo de grafo dirigido que representa una cadena cinemática robótica. Los nodos corresponden a articulaciones, puntos de percepción, soportes físicos y puntos lógicos, mientras que las aristas representan las transformaciones espaciales entre ellos. La estructura cumple la restricción de ser un árbol dirigido, donde la raíz define el sistema de referencia global y las hojas corresponden a los puntos observados por los sensores. Notar que las transofrmaciones asociadas a puntos lógicos corresponden a la transformación nula.

Del estado del arte se concluye que el problema se define como uno de mínimos cuadrados. Para esto, se debe definir una función que debe minimizarse. Esta función debe contemplar las posiciones vistas desde las diferentes cámaras. Cada una de estas cámaras debe ser colocada en un punto diferente de la cadena cinemática del robot.

Entonces, para cada objeto del entorno de trabajo x, se cargan dos nodos a la cadena cinemática del modelo. Uno de ellos es x^A, que representa al objeto visto por la cámara colocada en el punto A de la cadena cinemática, y su análogo x^B para el caso de la cámara colocada en el punto B. Notar que las aristas que conectan estos nodos con sus padres son las posiciones relativas del objeto x respecto de cada una de las cámaras. Esto permite cargar los diferentes objetos del entorno a la cadena cinemática del robot, lo que permite (utilizando lo definido en la Sección 2.3 para el cálculo de las cinemáticas directas) obtener las posiciones tanto de x^A respecto de un origen O, como de x^B respecto del mismo origen del sistema de referencia O.

A partir de esto, se define una función $p(x, \theta)$ que retorna la posición de un objeto x según un sistema de referencia O, sujeto a un parámetro θ. En este contexto, las posiciones percibidas del objeto desde la cámara A y la cámara B son $\mathbf{p}(x, \theta)_O^A$ y $\mathbf{p}(x, \theta)_O^B$, respectivamente.

Para medir la diferencia entre la posición percibida por una de las cámaras desde el punto A y la otra desde el punto B, se utiliza la norma euclidiana de la diferencia entre $p(x, \theta)_O^A$ y $p(x, \theta)_O^B$. A partir de esto, se define la siguiente función:

$$g(x, \theta) = \left\| \mathbf{p}(x, \theta)_O^A - \mathbf{p}(x, \theta)_O^B \right\|$$

Idealmente, tanto la cámara colocada en A como la cámara colocada en B observan el mismo objeto (es decir, el mismo punto), por lo tanto, la norma entre ambas posiciones debe ser igual a cero. Esta condición es considerada como una de las hipótesis fuertes del trabajo y es utilizada como fuente de verdad para la calibración.

Una vez definido un modelo que, dado un objeto, retorna la diferencia entre las posiciones observadas desde dos puntos distintos del robot, y establecida la fuente de verdad con la cual comparar, el problema puede ser formulado en términos de mínimos cuadrados como sigue:

Dado un conjunto de m datos,

$$(x_1^A, x_1^B, 0),\ (x_2^A, x_2^B, 0),\ \ldots,\ (x_m^A, x_m^B, 0),$$

donde cada x_i representa un objeto del entorno y el valor cero representa la observación ideal según la hipótesis, y dado un modelo $g(x, \theta)$, se busca el valor de θ que minimiza la diferencia entre lo predicho por el modelo y lo observado. Para ello, se utiliza el método de mínimos cuadrados no lineales.

4.1.2. Limitante del método

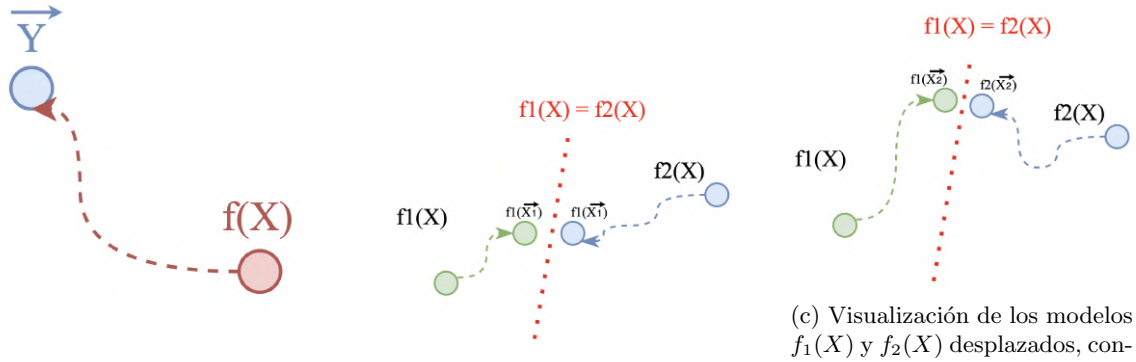
Como se explicó en el Marco Teórico, en un problema de mínimos cuadrados se parte de un conjunto de observaciones del mundo real (las cuales consideraremos como verdaderas), que se comparan contra las estimaciones producidas por un modelo $f(X)$. El objetivo es encontrar el vector de parámetros X que minimiza la distancia entre dichas observaciones y las predicciones del modelo.

Este proceso puede interpretarse como la existencia de un punto fijo en el espacio de dimensión m (las observaciones), y un punto móvil $f(X)$, que depende de los parámetros X en un espacio de dimensión n. Al variar X, el punto $f(X)$ se desplaza en el espacio con el objetivo de acercarse al

punto de referencia (las observaciones), reduciendo así el error en cada iteración. En la Figura 4.2a puede verse un ejemplo de este caso.

En nuestro caso, se busca minimizar la distancia entre dos modelos: $f_1(X)$ y $f_2(X)$, que representan las cinemáticas directas de dos cámaras. Además, debido a la estructura del vector X, los cambios en ciertas componentes de X afectan únicamente a f_1, mientras que otras afectan solo a f_2. Es decir, hay una dependencia parcial y separada entre las variables y los modelos.

Por lo tanto, existe un subespacio de soluciones posibles para las cuales la distancia entre $f_1(X)$ y $f_2(X)$ se minimiza igualmente. En otras palabras, el problema admite múltiples soluciones válidas para X que satisfacen el criterio de optimización, lo cual representa una limitación del método ya que la solución no necesariamente es única en términos de la minimización. Una ilustración de este caso puede verse en las figuras 4.2b y 4.2c.



(a) Ejemplo clásico de mínimos cuadrados: el modelo $f(X)$ se ajusta para coincidir con las observaciones reales.

(b) Visualización de los modelos $f_1(X)$ y $f_2(X)$ desplazados, convergiendo a una solución donde se igualan los modelos.

(c) Visualización de los modelos $f_1(X)$ y $f_2(X)$ desplazados, convergiendo a otra solución donde también se igualan los modelos, la cual es diferente a la del caso anterior.

Figura 4.2: Comparación entre el enfoque clásico de mínimos cuadrados y la formulación utilizada, donde la solución no necesariamente es única debido a la estructura del modelo.

4.1.3. Diseño de arquitectura

Una vez formulado el problema matemáticamente, se procede al diseño arquitectónico de la solución. Dicho diseño debe ajustarse al framework ROS.

Como se explica en la Sección 2.5, un sistema en ROS está compuesto por nodos que se comunican entre sí a través de canales comunes llamados tópicos. En nuestro caso, el sistema cuenta con tres nodos que intercambian información mediante seis tópicos.

En la Figura 4.3 se presenta un diagrama que describe la arquitectura del sistema implementado en ROS.

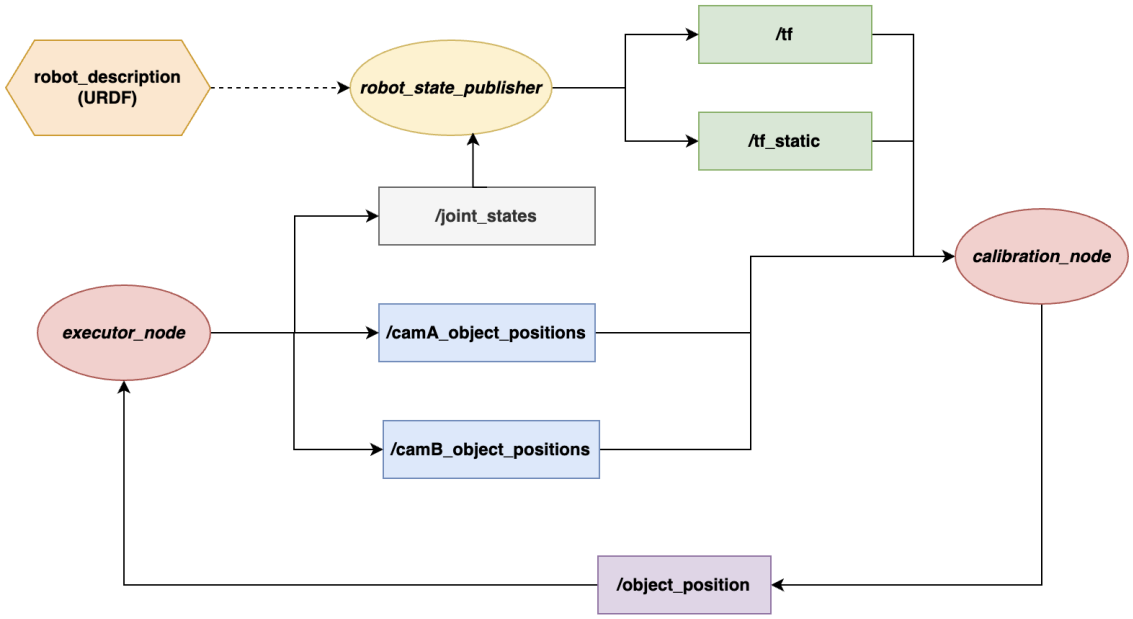


Figura 4.3: Diagrama del sistema ROS

A continuación, se explican cada uno de los componentes del sistema:

- **/camA_object_positions**: En este tópico se publican las posiciones de los objetos detectados por la cámara A, relativas a su marco de referencia. Para abstraer la lógica del nodo de calibración del tipo de mensaje específico publicado por el `executor_node`, se define un tipo de mensaje estándar.
- **/camB_object_positions**: En este tópico se publican las posiciones de los objetos detectados por la cámara B, relativas a su propio marco de referencia. También en este caso se utiliza un formato estándar de mensaje.
- **/tf**: En este tópico se publican las transformaciones espaciales dinámicas entre los marcos de referencia consecutivos de la cadena cinemática.
- **/tf_static**: En este tópico se publican las transformaciones espaciales estáticas entre los marcos de referencia consecutivos de la cadena cinemática.
- **/object_positions**: Una vez completada la rutina de calibración, se publican en este tópico las posiciones corregidas de los objetos. Esta información es utilizada en rutinas de manipulación o localización de objetos.
- **/joint_states**: En este tópico se publican las posiciones actuales de los motores de las diferentes articulaciones de la cadena cinemática. Con esta información, junto con la geometría del robot descrita en el archivo URDF, se calculan las transformaciones espaciales entre las articulaciones, las cuales son posteriormente publicadas en el tópico `/tf`.

- **executor_node**: Este nodo publica en los tópicos `/camA_object_positions` y `/camB_object_positions` las posiciones de los objetos relativas a las cámaras A y B, respectivamente. También se suscribe al tópico `/object_positions` para obtener las posiciones corregidas de los objetos.

- **robot_state_publisher**: Este nodo se suscribe al tópico `/joint_states`, desde el cual recibe información sobre las posiciones articulares del robot. A partir de estos datos, calcula las transformaciones que definen la posición y orientación de los distintos segmentos del robot, y las publica en los tópicos `/tf` y `/tf_static`, permitiendo que otros nodos accedan a información actualizada sobre la configuración del robot en tiempo real.

- **calibration_node**: Este nodo se encarga de calcular las posiciones de los objetos publicadas en los tópicos `/camA_object_positions` y `/camB_object_positions` con respecto a la base. Luego ejecuta el algoritmo de calibración y publica en el tópico `/tf` las posiciones de los objetos antes y después de la calibración de los parámetros.

4.2. Implementación

Una vez definido el diseño arquitectónico, y considerando la formulación matemática, se procede a implementar la solución propuesta.

Siguiendo el mismo razonamiento presentado en la sección anterior, el primer paso consiste en implementar el código necesario para modelar el grafo que representa la cadena cinemática del robot. Para ello, se define el archivo **grafo.py**, el cual, basado en la biblioteca `networkx` (Hagberg, Schult, y Swart, 2024) de Python, permite construir un grafo dirigido como estructura de datos. En este archivo también se implementan las operaciones necesarias para manipular dicho grafo. Los nodos del grafo representan los elementos definidos en la Subsección 4.1.1, mientras que cada arista contiene la transformación espacial relativa entre los nodos conectados.

Una vez definido el grafo, se desarrolla un módulo que permite reconstruir las posiciones de los objetos respecto de un marco de referencia base. Esta información es utilizada en la función objetivo a minimizar. Para ello, se define el archivo **fkOperations.py**. Una de las operaciones principales implementadas en este archivo es la función `forward_kinematic`, la cual corresponde al modelo de cinemática directa $p(x, \theta)$ definido en la Sección 4.1. Esta función recibe como entrada el identificador del objeto x, el nombre del nodo que representa la cámara, y un vector de desplazamiento (cuya función se detalla más adelante). Adicionalmente, recibe un vector de parámetros θ, que permite parametrizar la cadena cinemática. La función retorna la matriz homogénea que representa la transformación espacial entre el marco base y el objeto, según la cadena cinemática que atraviesa la cámara especificada. La construcción de la matriz homogénea se realiza conforme a lo definido en la Sección 2.3. En el Anexo A.2 se presenta un pseudocódigo correspondiente a esta función.

Dentro del mismo archivo **fkOperations.py** se implementa la función `least_squares`, la cual utiliza la misma función de la biblioteca `scipy.optimize` para obtener, dado un solver, el valor de los parámetros que minimiza la norma cuadrática de la diferencia entre `forward_kinematic(obj, camA)` y `forward_kinematic(obj, camB)` para cada objeto.

A continuación, se procede a implementar los nodos necesarios para llevar a cabo la calibración. Según el diseño presentado en la Figura 4.3, el único nodo que requiere desarrollo es el nodo de calibración. Esto se debe a que el nodo `robot_state_publisher` se encuentra disponible por defecto al instalar ROS, mientras que el nodo `executor_node` está fuertemente acoplado al entorno

de trabajo y al robot utilizado. Por lo tanto, su implementación queda a cargo del experimentador, al igual que la descripción URDF del modelo del robot.

Lo único que debe garantizarse al implementar este nodo ejecutor es que, para permitir una correcta comunicación con el nodo de calibración, las posiciones de los objetos del entorno observados por ambas cámaras sean publicadas en dos tópicos diferentes (configurables por parámetro), respetando el formato de mensaje común definido en el Anexo A.1.1.

En términos generales, la estructura de archivos que permite implementar el nodo de calibración se muestra en la Figura 4.4, donde:

- **params_file.yml**: Es un archivo de configuración de ROS, el cual permite dinámicamente modelar diferentes comportamientos en la ejecución de la calibración (Ej: parámetros iniciales, si se ejecuta algún test, etc).

- **constantes.py**: Es un archivo donde se declaran variables y constantes globales que son utilizadas por todos los archivos. Permite manejar de manera transversal a todos los módulos variables/constantes.

- **calibration_node.py**: Contiene la logica prinicipal del nodo. Aquí se definen las suscripciones a los diferentes tópicos (junto con sus callbacks), así como los tópicos donde el propio nodo publicará. Este nodo es quien recibe y procesa el .yml que se envía como parámetro en su ejecución.

- **utils.py**: Contiene métodos auxiliares para el desarrollo de la lógica del nodo de calibración.

- **fk_operations.py**: Define aquellas operaciones necesarias para calcular la cinemática directa del robot, junto con la lógica principal en la minimización.

- **grafo.py**: Permite definir un tipo de dato el cual representa un grafo dirigido, utilizado para representar la estructura cinemática del robot. Esta estructura es de especial utilidad a la hora de calcular la cinemática directa del modelo, ya que al recorrerla, se obtiene cada paso las transformaciones espaciales entre los nodos.

- **subscribeMethods.py**: Define los métodos principales que se ejecutan en los callback de las suscripciones a los diferentes tópicos.

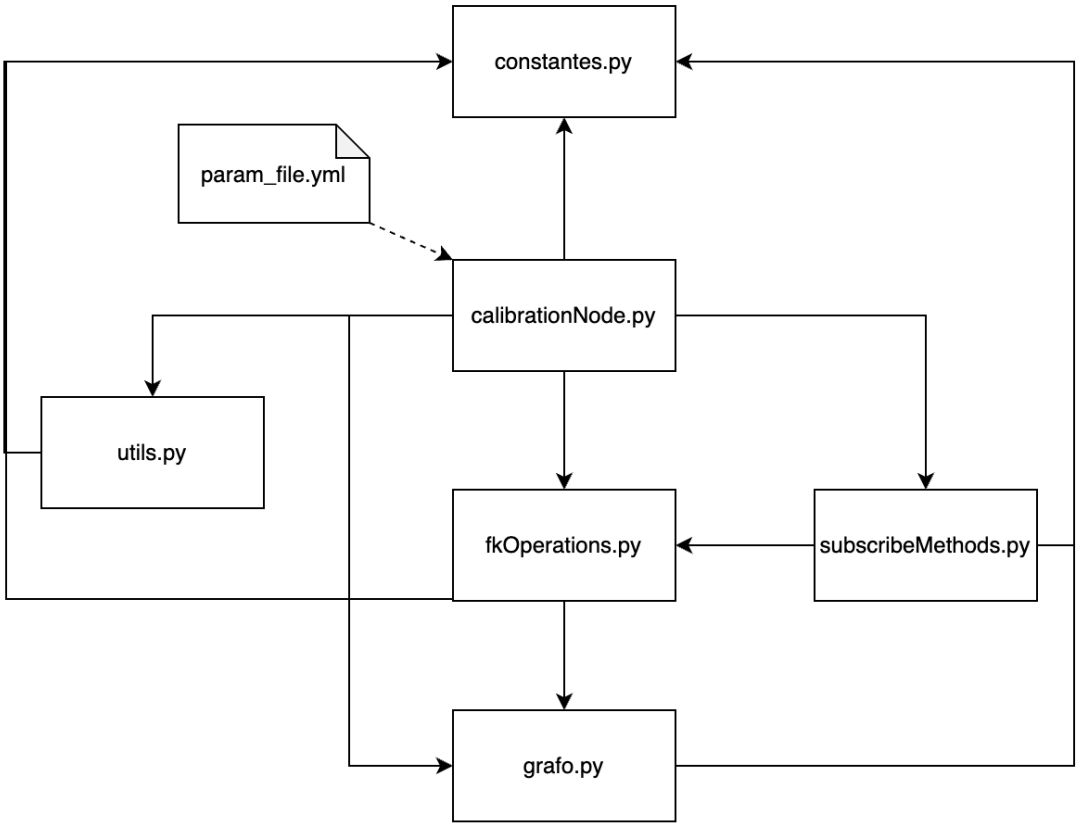


Figura 4.4: Diagrama de los principales archivos de la solución implementada. Una flecha entre archivos indica que uno invoca o depende del otro. Se observa que todos los archivos dependen del archivo de constantes. El archivo param_file es inyectado al inicio de la ejecución del nodo ROS.

Como se indicó anteriormente, en el archivo **subscribeMethods.py** se definen los métodos de suscripción (callbacks) que se ejecutan cada vez que se publica un mensaje en los tópicos a los que el nodo de calibración está suscrito. Los dos métodos principales definidos en este archivo son: `pose_callback` y `getCycles`.

El método `pose_callback` se ejecuta cada vez que se publica un mensaje en los tópicos `/tf` o `/tf_static`. Su objetivo es cargar en memoria la estructura cinemática del robot. Para ello, se instancia un grafo del tipo definido en el archivo **grafo.py**, y se utilizan las operaciones allí implementadas para interactuar con dicha estructura.

El segundo método, `getCycles`, se ejecuta cada vez que se publica un mensaje en los tópicos asociados a las detecciones hechas por cada una de las cámaras, siempre y cuando se haya cargado en memoria tanto la información publicada en `/tf` como la correspondiente a `/tf_static`. Esta condición es fundamental, ya que, de lo contrario, al calcular la cinemática directa desde la base hacia los objetos del entorno, se generaría un error por inconsistencias en el grafo construido en memoria.

El objetivo principal del método `getCycles` es identificar aquellos objetos que están siendo observados simultáneamente por ambas cámaras. Esta tarea no es trivial, ya que los objetos son detectados mediante un sistema de visión artificial, y cada cámara publica información sobre un objeto no identificado, perteneciente a una categoría determinada. Es responsabilidad del algoritmo identificar, a partir de esta información, cuáles objetos están siendo vistos por ambas cámaras al mismo tiempo.

Para lograr esta asociación, se establece como hipótesis que los objetos de una misma categoría, detectados tanto por la cámara A como por la cámara B, que se encuentran a menor distancia entre sí (comparados con los demás objetos de esa categoría), corresponden al mismo objeto físico. Esta hipótesis asume que las cámaras no presentan una descalibración significativa.

4.2.1. Comportamiento genérico del nodo `calibration`

Una vez descritos los principales archivos y métodos de la solución desarrollada, se presenta a continuación el comportamiento general del nodo de calibración.

Al iniciarse el nodo, se espera que el nodo `robot_state_publisher` publique en los tópicos `/tf` y `/tf_static` las relaciones espaciales relativas entre los marcos de referencia consecutivos del robot. Para ello, utiliza la información contenida en el archivo URDF provisto como parámetro, junto con los estados articulares actuales.

Una vez que se reciben los datos en los respectivos tópicos, el nodo de calibración carga en memoria un grafo dirigido con estructura arborescente, que representa la cadena cinemática del robot. A partir de ese momento, el nodo está en condiciones de procesar la información publicada en los tópicos `/camA_object_positions` y `/camB_object_positions`.

Cada vez que se publica un mensaje en alguno de estos tópicos, se ejecuta el método `getCycles`, el cual almacena en memoria los objetos observados por la cámara correspondiente (descartando aquellos que ya no se encuentran visibles) y los compara con los objetos observados por la otra cámara, con el objetivo de identificar ciclos que representen la observación simultánea del mismo objeto por ambas cámaras.

Para cada objeto detectado en esta etapa, se calcula su posición con respecto a un marco de referencia mediante el cálculo de la cinemática directa. La diferencia entre ambas posiciones se utiliza posteriormente en el proceso de minimización. Por último, se ejecuta el procedimiento de optimización. El resultado de este proceso es el vector θ, que luego se utiliza para recalcular las posiciones corregidas de los objetos. En la Figura 4.5 se ilustra gráficamente este proceso.

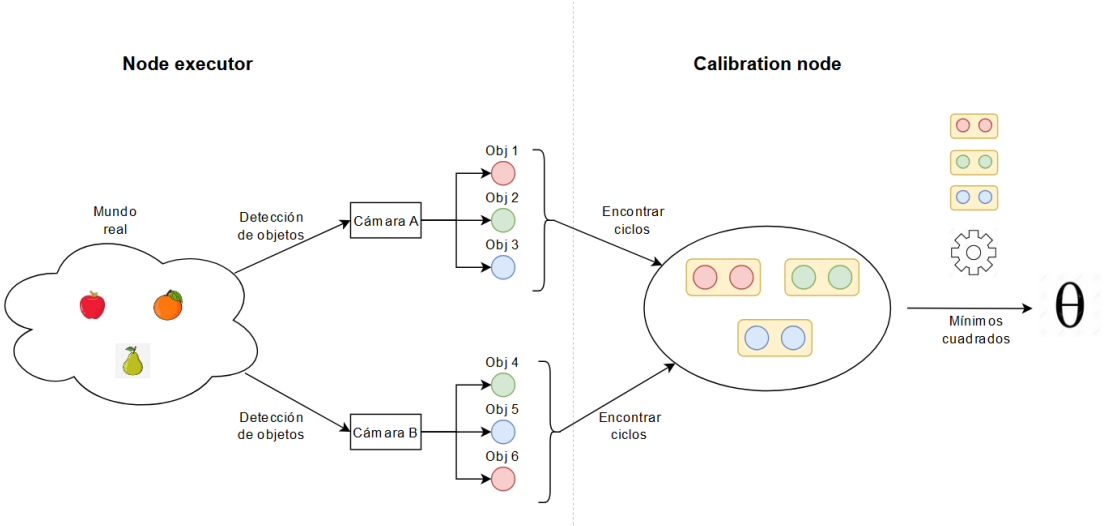
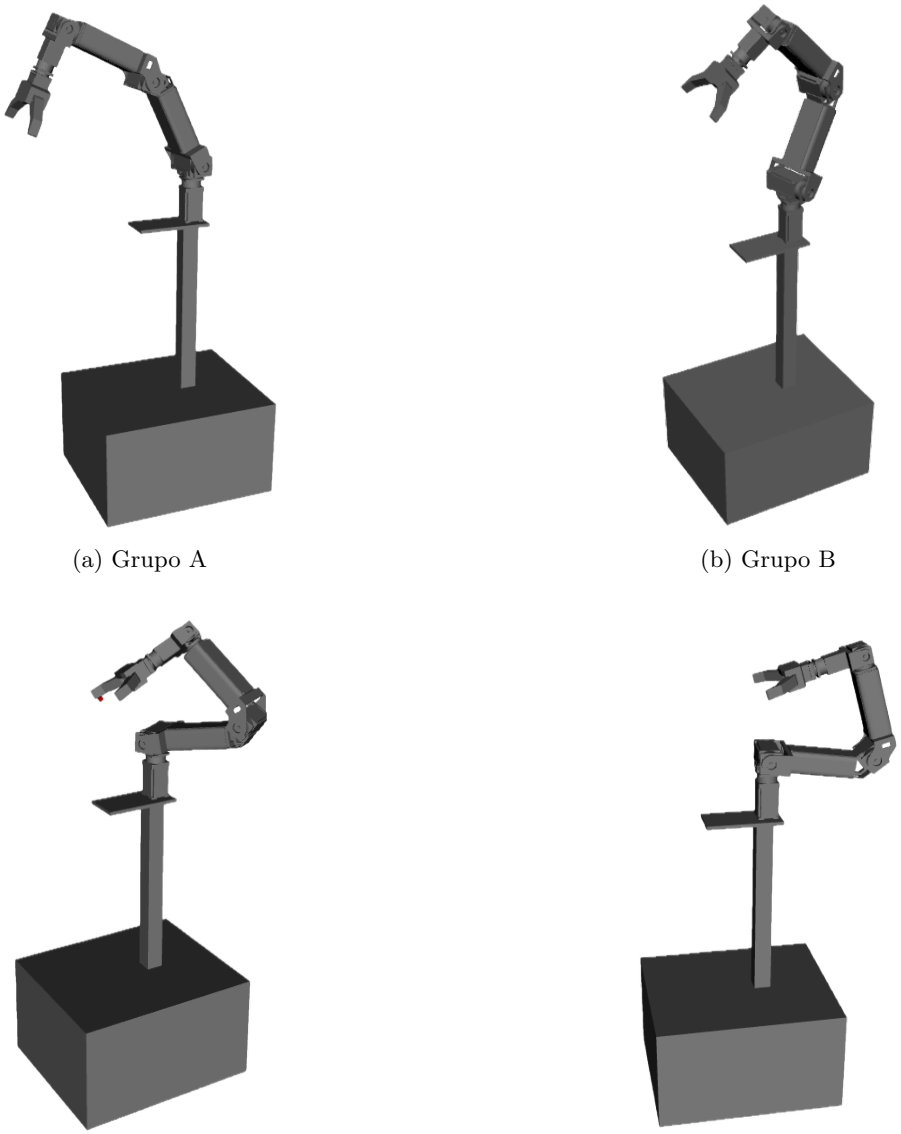


Figura 4.5: Diagrama del proceso de identificación de los objetos del mundo real para su posterior procesamiento. Se distingue la responsabilidad del nodo ejecutor, al ser quien reconoce los objetos del mundo y lo publica en los tópicos correspondientes, de la responsabilidad del nodo de calibración que debe identificar los ciclos semánticos para realizar la minimización.

Capítulo 5

Experimentación

El objetivo de este capítulo es evaluar la solución desarrollada. Para ello, se realizan dos tipos de pruebas: pruebas online y pruebas offline. Las pruebas online se ejecutan utilizando un robot real. En este caso, se cuenta con un nodo ejecutor que se encarga de obtener las posiciones de los objetos, publicarlas en los tópicos correspondientes y luego esperar la publicación de las posiciones corregidas en el tópico /object_positions, con el fin de permitir la interacción con dichos objetos. Por otro lado, las pruebas offline son realizadas sin la intervención directa del robot físico, empleándose en su lugar una simulación del mismo. Para llevar a cabo estas pruebas, son grabadas diferentes escenas en las que se varían las posiciones del brazo. Para cada posición del brazo, son registradas múltiples escenas, variando los objetos del entorno y sus ubicaciones. Cada pose del brazo es agrupada posteriormente en uno de los grupos A, B, C y D. En la Figura 5.1 son mostradas las distintas poses consideradas. Estas escenas son almacenadas en archivos bag (ver Sección 2.5), lo que implica que la etapa de grabación es realizada utilizando un robot real. En este caso, se emplea el mismo robot utilizado en las pruebas online, lo que permite trabajar con un entorno experimental común para ambos tipos de evaluación.



(a) Grupo A

(b) Grupo B

(c) Grupo C

(d) Grupo D

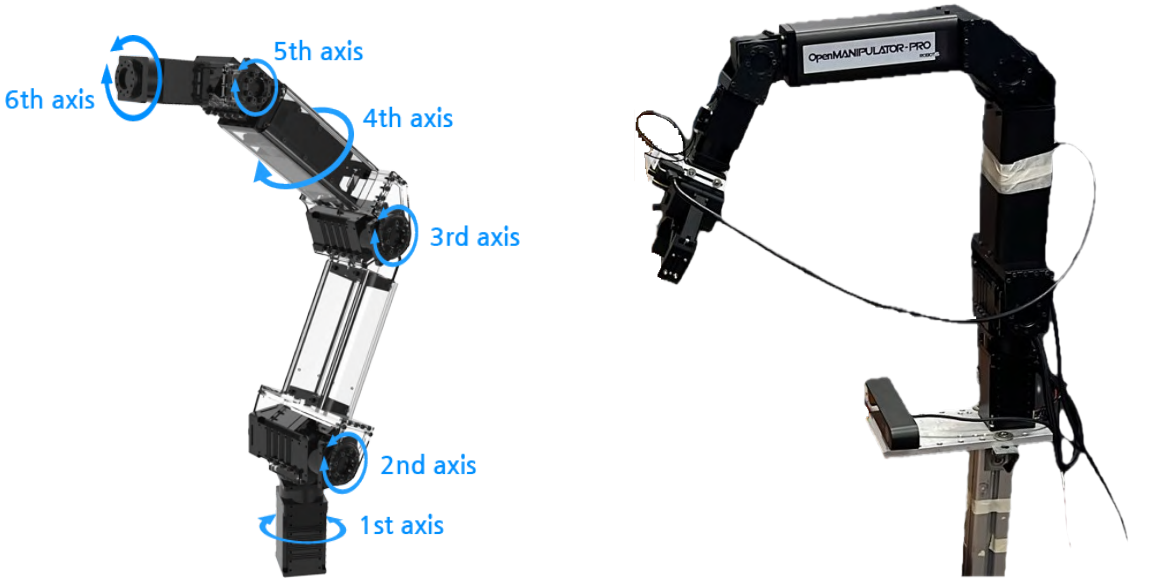
Figura 5.1: Diferentes poses grabadas del robot.

5.1. Entorno de trabajo

El robot utilizado tanto para las pruebas online como para las pruebas offline (mediante simulaciones) es el Open Manipulator Pro, de la empresa Robotis. Este robot cuenta con seis articulaciones de tipo rotacional, lo que le otorga seis grados de libertad. En la Figura 5.2a se muestra una imagen del robot junto con los ejes de rotación correspondientes a sus articulaciones. La versión empleada

incorpora además una pinza, utilizada para interactuar con los distintos objetos del entorno.

El robot se monta sobre un soporte físico y se le integran una camara ZED 2, asociada al pecho (chest) del robot y una camara ZED mini asociada al brazo (arm). En Anexo A.4.1 y Anexo A.4.2 se pueden ver imágenes de cada una de estas cámaras. En la Figura 5.2b se presenta una imagen del entorno experimental completamente montado.

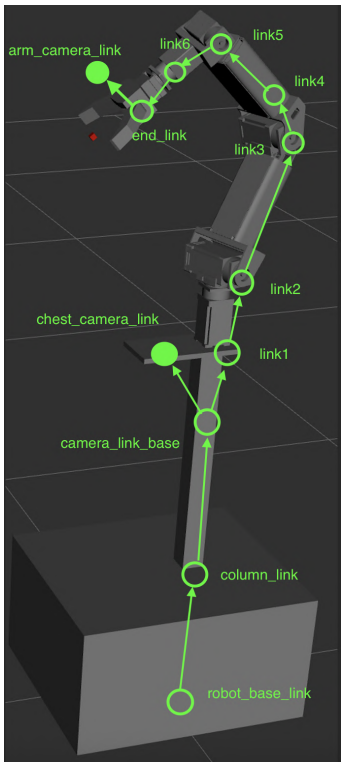


(a) Open Manipulator Pro junto con sus ejes de rotación.

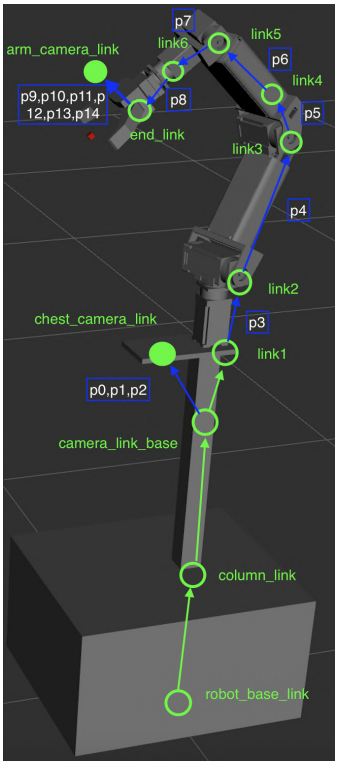
(b) Open Manipulator Pro montado con dos cámaras.

Figura 5.2: Vista del Open Manipulator Pro y su configuración experimental.

Como se explicó en el capítulo anterior, una de las tareas principales es cargar en memoria el grafo que representa tanto la cadena cinemática del robot como las relaciones espaciales relativas entre los objetos y las cámaras. Para ello, el nodo ejecutor publica la información de la geometría del robot en cada escena en los tópicos /tf y /tf_static, mientras que en los tópicos /yolo_arm_detection y /yolo_chest_detection se publican las posiciones de los objetos relativas a las cámaras colocadas en el brazo y la cadera, respectivamente. A partir de esta información se carga la cadena cinemática en memoria. En la Figura 5.3a se muestra una simulación del robot montado sobre la base, con su cadena cinemática superpuesta. El nodo robot_base_link representa el origen del sistema de coordenadas, mientras que los nodos arm_camera_link y chest_camera_link representan las cámaras.



(a) Simulación del robot montado sobre los soportes, junto con la ilustración de su cadena cinemática.



(b) Misma simulación con la cadena cinemática destacando los enlaces (flechas azules) que serán parametrizados en la minimización. Se indica también la cantidad de parámetros asociados a cada enlace.

Figura 5.3: Comparación entre la cadena cinemática completa del robot y la configuración utilizada para la calibración, destacando los enlaces parametrizados.

Otro de las principales tareas necesaria para poder realizar correctamente la calibración es la de identificar en el sistema de coordenadas (X,Y,Z) los ejes sobre los cuales rota cada articulación. Esto es de suma importancia para poder saber sobre qué ejes realizar las parametrizaciones. Para esto se relevó el sitio oficial del proveedor del robot. Notar que cada link corresponde a cada uno de los seis ejes de rotación con los que cuenta el manipulador. El proveedor informa que:

- **link 2**: rota libremente sobre el eje z. La rotación sobre los ejes x e y es nula.
- **link 3**: rota libremente sobre el eje y. La rotación sobre los ejes x y z es nula.
- **link 4**: rota libremente sobre el eje y. La rotación sobre los ejes x y z es nula.
- **link 5**: rota libremente sobre el eje x. La rotación sobre los ejes y y z es nula.
- **link 6**: rota libremente sobre el eje y. La rotación sobre los ejes x y z es nula.

- **end link**: rota libremente sobre el eje x. La rotación sobre los ejes y y z es nula.

En la Figura 5.3 se ilustran los enlaces que serán parametrizados en el proceso de minimización durante todas las pruebas. Estos enlaces están representados por flechas de color azul. Además, se indica la cantidad de parámetros asociados a cada uno de ellos. La selección de los enlaces candidatos a ser parametrizados en la cadena cinemática se basa en dos criterios principales: (i) que el enlace corresponda a un componente móvil, es decir, una articulación susceptible a degradaciones mecánicas con el uso; o (ii) que su posición en el espacio (ya sea rotación o traslación) haya sido definida de manera imprecisa o no técnica, como sucede en casos donde las transformaciones hacia las cámaras han sido estimadas manualmente o mediante inspección visual. En nuestro caso, se decidió estimar las posiciones traslacionales de la cámara ubicada en el pecho, las cuales están representadas por los parámetros p_0, p_1 y p_2. Por otro lado, las rotaciones correspondientes a las articulaciones del brazo se modelan mediante los parámetros p_3, p_4, p_5, p_6, p_7 y p_8. Finalmente, para la cámara del brazo se consideran tanto los parámetros de traslación (p_9, p_{10} y p_{11}) como los de rotación (p_{12}, p_{13} y p_{14}).

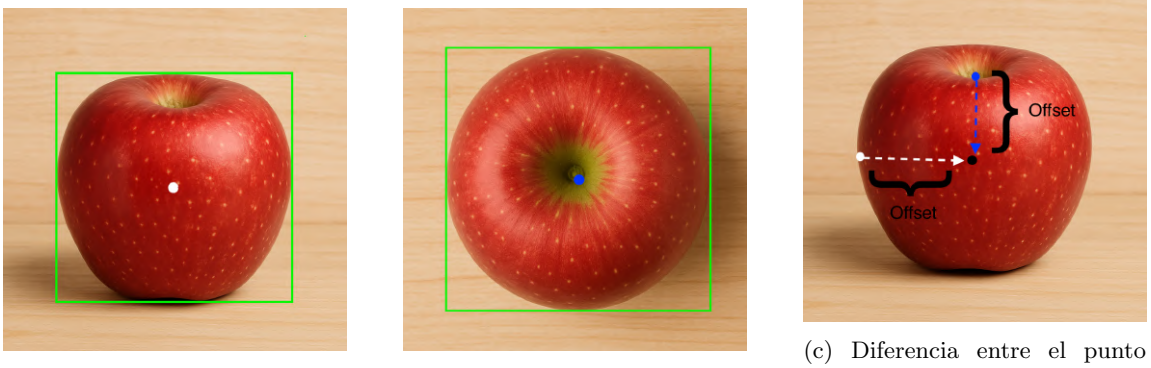
5.2. Aspectos preliminares

En el contexto de una calibración cinemática utilizando dos cámaras, es necesario distinguir entre los parámetros intrínsecos y extrínsecos de cada una. Los parámetros intrínsecos definen las características internas de la cámara, como la distancia focal, el punto principal y los coeficientes de distorsión. Estos parámetros permiten proyectar puntos del espacio tridimensional al plano de la imagen de forma precisa. Por otro lado, los parámetros extrínsecos describen la posición y orientación de la cámara con respecto a un marco de referencia externo, por ejemplo, la base del robot o algún enlace intermedio. En este tipo de calibración, los parámetros extrínsecos son los que vinculan la percepción visual con la cinemática del robot. En las pruebas realizadas en este trabajo, se asume que los parámetros intrínsecos de ambas cámaras están correctamente calibrados. Esto implica que las posiciones de los objetos estimadas por cada cámara en el espacio tridimensional son confiables. El objetivo entonces es estimar con precisión los parámetros extrínsecos, es decir, las transformaciones que vinculan cada cámara con la cadena cinemática del robot.

Como modelo de visión artificial, se utiliza el sistema You Only Look Once (YOLO). Como se explica en la Sección 2.6, YOLO es un sistema de una etapa, lo cual tiene como ventaja su velocidad en ejecución, pero como desventaja presenta menor precisión a la hora de identificar categóricamente un objeto (frente a un sistema de visión artificial de dos etapas). Esto influye directamente en la metodología de calibración, ya que puede suceder que un objeto sea identificado por una cámara en una categoría y por la otra en una categoría completamente diferente. Debido a que el algoritmo para la identificación de ciclos es semántico, se estarían perdiendo ciclos para la minimización. Para resolver este problema, se opta por categorizar los objetos bajo una categoría común. Para evitar que se encuentren ciclos inexistentes, los bags utilizados son sanitizados. Además, se aprovecha la función descrita en la Subsección 4.1.3 para encontrar los elementos de un ciclo mediante sus posiciones en el espacio.

Para obtener el punto representativo de cada imagen, que será utilizado para la minimización, se utiliza el punto central del bounding box de la imagen. Esta decisión introduce el problema de diferencia entre puntos centrales según vistas. Para comprenderlo mejor, se presenta la Figura 5.4. En la primera imagen se muestra una manzana vista de frente. También se observa el bounding box asociado y el centro de ese rectángulo, que es el punto representativo del objeto. La posición de ese

punto es almacenada en memoria en la cadena cinemática. Luego, en la segunda imagen, se muestra la misma manzana vista desde arriba. Al igual que en el caso anterior, se observa su bounding box y su punto representativo. Se debe notar que, en el modelo, los diferentes objetos son representados a través de estos puntos, por lo que idealmente se requiere que ambas cámaras visualicen el mismo punto. Como se puede observar en estas dos primeras imágenes, esto no necesariamente sucede, lo que introduce un error en la calibración, como se muestra en la imagen c). Este error es propio del método, y es el costo de usar objetos a priori desconocidos para realizar la calibración. Si bien esto podría minorarse si los diferentes objetos fueran vistos desde puntos de vista similares, se requerirían poses específicas para cada escena, lo cual va en contra del objetivo de proponer un método de calibración simple. Con la intención de atenuar los efectos asociados con este problema, se decidió utilizar un offset. Es decir, que la posición del objeto no será la asociada al centro del bounding box, sino la asociada al desplazamiento de ese punto, en la dirección del vector a la proyección de la imagen. La intención con esto es acercar más los puntos al centro del objeto. En la Figura 5.4c se puede ver como sería. Notar que este problema se acentúa aún más cuando se trabaja con objetos que son vistos desde puntos muy diferentes.



(a) Manzana vista de frente.

(b) Manzana vista de arriba.

(c) Diferencia entre el punto central de ambas vistas.

Figura 5.4: Vistas de la manzana y comparación de sus puntos centrales desde diferentes ángulos.

Otra de las cuestiones a tener en cuenta a la hora de ejecutar la rutina de calibración, es que, cada vez que se detecta un objeto, se genera una bounding box rectangular que lo encierra y estima como centro del objeto el centro geométrico de dicha caja. Este punto central es el que se utiliza para estimar la posición 3D del objeto. Uno de los problemas que pueden surgir al determinar el punto de calibración de esta manera es que, debido a la forma del objeto o al ángulo de visión, el centro del bounding box no coincida con una región representativa del objeto real. En algunos casos, este punto puede incluso caer fuera del contorno físico del objeto. Por ejemplo, si el objeto es alargado como una banana y está orientado en profundidad respecto a la cámara, el centro de la *bounding box* podría situarse visualmente más cerca del fondo de la escena que del objeto real. Esto puede llevar a una estimación de posición 3D errónea, ya que el punto proyectado en el espacio podría corresponder a una región del fondo y no al objeto en sí. Este tipo de problemas revela que ciertos objetos son más adecuados o "amigables" para nuestro sistema de calibración que otros. Una posible estrategia para mitigar este efecto sería filtrar determinadas clases de objetos que, por su

forma o comportamiento visual, se sepa que tienden a generar errores en la localización del centro. Otra forma podría ser definir un límite superior sobre las distancias a priori que puede haber entre cada punto y filtrar aquellos objetos que superen esta distancia previo a la calibración. Este enfoque fue el utilizado en esta investigación.

5.3. Experimento 1: Determinar el mejor solver

Como se explicó en la Subsección 2.1.1, los problemas de mínimos cuadrados no lineales se resuelven utilizando métodos iterativos. La función `least_squares` de la librería SciPy soporta tres algoritmos (solver) diferentes para resolver el problema: `trf`, basado en regiones de confianza; `dogbox`, también basado en regiones de confianza pero con restricciones rectangulares; y `lm`, una variante del método de Levenberg-Marquardt que no admite restricciones en los parámetros.

El objetivo de esta prueba es determinar cuál de estos métodos es el más adecuado para nuestro problema, y además, verificar que en todos los casos se reduzca el error en comparación a la no calibración. Para ello, se utilizaron las 11 escenas grabadas previamente. En cada escena, se realizó la calibración utilizando cada uno de los métodos mencionados. Además, se evaluaron diferentes valores de desplazamiento (u *offsets*) como enfoque para abordar el problema planteado en la sección anterior. Los offsets utilizados fueron 0.0 (sin desplazamiento), 0.02, 0.035 y 0.05 m. Estos valores fueron seleccionados considerando que los objetos utilizados en las pruebas son, en su mayoría, manzanas con un radio aproximado de 0.035 m, lo cual, según la Figura 5.4, representa el desplazamiento necesario en la dirección de los puntos observados por las cámaras para que coincidan espacialmente.

Para cada combinación de escena, solver y offset, se registraron: el vector solución, el tiempo de ejecución y las distancias entre las posiciones estimadas del objeto según la cámara del brazo y la cámara de la cadera.

La Figura 5.5 presenta un diagrama de cajas que muestra las distancias entre los puntos observados por ambas cámaras antes de aplicar la calibración, en función del offset utilizado. Por otro lado, la Figura 5.6 muestra los resultados obtenidos con el método `trf`, el cual ha sido determinado como el más adecuado.

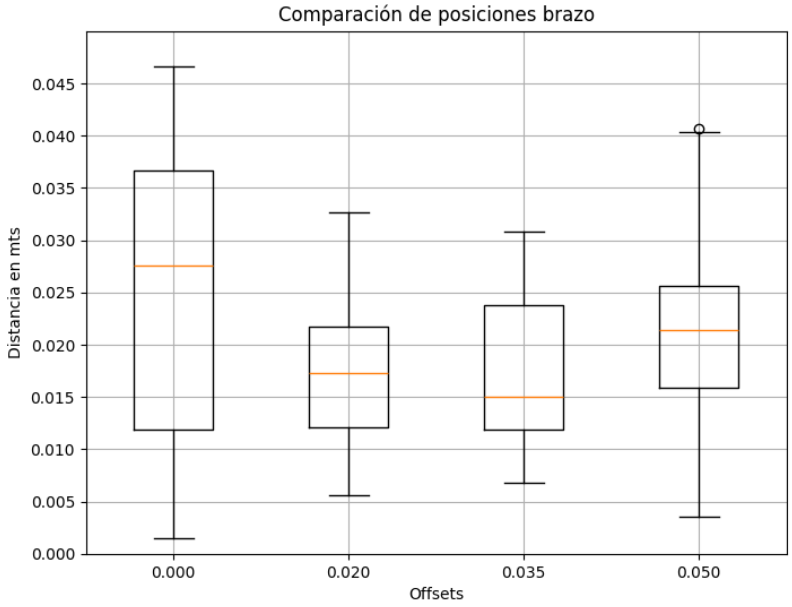


Figura 5.5: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara previo a la calibración. Los resultados son agrupados por diferentes valores de offset.

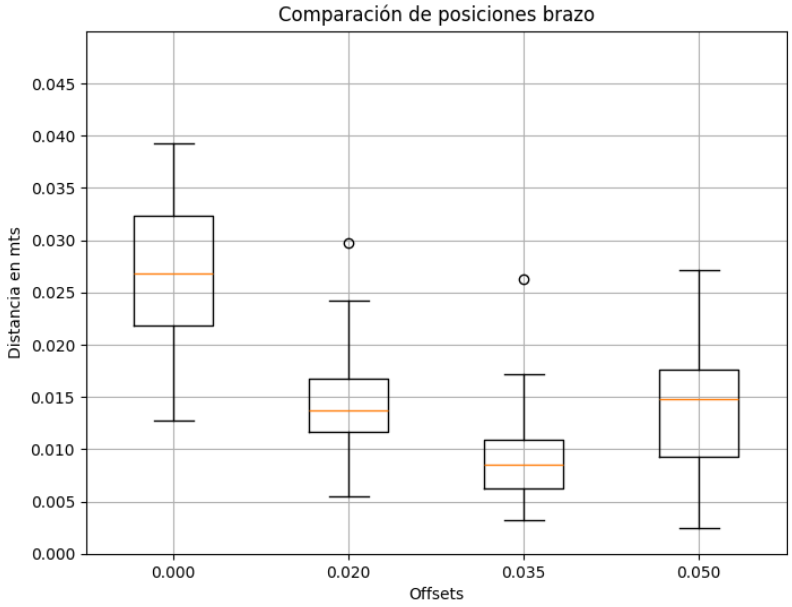


Figura 5.6: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara luego de la calibración usando el solver trf. Los resultados son agrupados por diferentes valores de offset.

La primera conclusión que puede extraerse al comparar ambas figuras es que, la diferencia entre las posiciones observadas por cada cámara previo a la calibración, es considerablemente mayor en todos los casos, en comparación con los resultados obtenidos luego del proceso de calibración. Esto permite afirmar que realizar la calibración mejora significativamente los resultados. Por otro lado, también se observa que el mejor resultado en esta etapa previa a la calibración se obtiene cuando el *offset* es de 0,035 m. Estimamos que esto podría deberse a que dicho valor se aproxima al radio promedio de los objetos utilizados en las pruebas.

En cuanto a los resultados obtenidos por cada uno de los diferentes métodos, los tres métodos lograron resultados similares respecto a las distancias entre las posiciones estimadas por ambas cámaras (En Anexo A.5 se encuentran las gráficas de caja asociadas a los restantes dos solver). Sin embargo, en términos de tiempo de ejecución, `trf` y `lm` mostraron un desempeño significativamente mejor, con tiempos promedio inferiores a dos segundos, mientras que `dogbox` superó el minuto de ejecución en promedio.

Para evaluar la verosimilitud de la solución retornada por cada método, partimos de la hipótesis de que el sistema no se encuentra completamente descalibrado. Por ello, se definieron límites razonables en los parámetros: desplazamientos entre -5 cm y 5 cm en cada eje de traslación, y rotaciones entre $-\frac{\pi}{8}$ y $\frac{\pi}{8}$ radianes. Estos valores fueron elegidos de forma arbitraria como acotación de un rango físicamente posible. Por lo tanto, cualquier solución retornada por el método dentro de este rango se considera válida. Tanto `trf` como `dogbox` permiten incorporar restricciones en el espacio de parámetros, lo cual limita la búsqueda a una subregión del espacio. En cambio, `lm` no permite restricciones, por lo que la solución puede divergir significativamente. De hecho, en los experimentos observamos que `lm` converge a soluciones no realistas, con traslaciones de hasta 50 cm, lo cual no se corresponde con el conocimiento previo del sistema. Aunque no conocemos con exactitud el error en la posición de los distintos enlaces (que es precisamente lo que buscamos estimar), asumimos que la estimación inicial no está completamente desfasada respecto de la realidad. Otra de las limitantes del algoritmo `lm` es el que requiere tener una cantidad mínima de objetos para poder realizar la calibración. Esta cantidad es igual a la cantidad de parámetros a estimar dividido 3 (ya que de cada objeto se obtienen 3 coordenadas).

Por todas las razones expuestas anteriormente, se puede concluir que el método `trf` es el mejor solver para este problema. Además, en la Figura 5.6 se puede observar cómo los diferentes offsets influyen en los resultados, destacando el impacto del problema planteado en la sección anterior. En particular, el offset que mostró los mejores resultados fue **0.035**, valor que será utilizado en los experimentos siguientes.

5.4. Experimento 2: Sobreajuste

Otro de los aprendizajes obtenidos a partir del relevamiento del estado del arte es la identificación de una de las principales problemáticas presentes en la calibración paramétrica: el sobreajuste. En este contexto, el término sobreajuste es utilizado para describir el caso en que un parámetro es estimado en una escena específica de forma tal que la distancia entre los objetos observados por las distintas cámaras es minimizada, pero que, al ser aplicado en una segunda escena, se obtienen resultados considerablemente peores en comparación con los parámetros que pudiesen ser obtenidos al calibrar directamente en dicha nueva escena. Una forma de determinar si el método sobreajusta consiste en realizar la calibración sobre las escenas utilizando el mismo vector inicial de parámetros, y posteriormente analizar cómo se distribuyen los vectores solución obtenidos. Debido a la limitación mencionada en la Subsección 4.1.2, se reconoce que el algoritmo de minimización

puede converger a soluciones muy distintas en cada escena, aunque todos ellos permitan minimizar el error adecuadamente. Sin embargo, si se observa que los vectores resultantes presentan una distribución acotada, es decir, si los parámetros finales no varían significativamente entre escenas, se puede concluir que el algoritmo no sobreajusta a escenas particulares.

La prueba es realizada calibrando el sistema en las escenas a partir del mismo vector inicial (el vector nulo), y luego se analiza estadísticamente la distribución de los parámetros obtenidos. En la Tabla 5.1 son presentados los principales valores estadísticos correspondientes a cada uno de los parámetros para el caso de la calibración de 15 parámetros tal cual se presentan en la Figura 5.3. Dado que se observa una baja variabilidad en general, se concluye que el algoritmo tiende a converger a soluciones similares en todas las escenas, por lo que no se presenta evidencia de sobreajuste.

Parámetro	Media	Desvío	Min	Max	Q1	Q3	RIC
Cámara del pecho (Traslaciones XYZ)							
p0 (m)	0.00103	0.00465	-0.00736	0.00653	-0.00125	0.00454	0.00579
p1 (m)	0.00192	0.00664	-0.00920	0.01005	-0.00192	0.00672	0.00864
p2 (m)	0.00162	0.00415	-0.00627	0.00798	-0.00125	0.00405	0.00530
Articulaciones (Rotaciones alrededor de sus ejes)							
p3 (rad)	0.01475	0.07133	-0.12378	0.12849	-0.00752	0.04653	0.05405
p4 (rad)	-0.00514	0.00948	-0.02364	0.01006	-0.00716	-0.00225	0.00491
p5 (rad)	0.00284	0.00950	-0.00809	0.02345	-0.00481	0.00530	0.01011
p6 (rad)	0.04292	0.08125	-0.01029	0.25188	-0.00236	0.03973	0.04209
p7 (rad)	0.00791	0.01475	-0.00880	0.03398	-0.00276	0.01474	0.01750
p8 (rad)	-0.00684	0.08368	-0.16529	0.16578	-0.01967	0.00580	0.02547
Cámara del brazo (Traslaciones XYZ, luego rotaciones XYZ)							
p9 (m)	0.00116	0.00430	-0.00443	0.00936	-0.00181	0.00331	0.00513
p10 (m)	-0.00213	0.00880	-0.01861	0.00905	-0.00622	0.00620	0.01243
p11 (m)	-0.00127	0.00224	-0.00504	0.00170	-0.00328	0.00035	0.00362
p12 (rad)	0.00119	0.09724	-0.12461	0.19408	-0.05447	0.03020	0.08467
p13 (rad)	-0.01508	0.02231	-0.05626	0.01688	-0.02544	-0.00171	0.02373
p14 (rad)	-0.03635	0.13447	-0.37754	0.09437	-0.02845	0.01888	0.04732

Tabla 5.1: Estadísticos resumen por parámetro: media, desvío estándar, mínimos, máximos, cuartiles y rango intercuartílico. Los parámetros son agrupados de manera semántica por: cámara del brazo, articulaciones, y cámara del pecho.

5.5. Experimento 3: Evaluando poses

Se realizaron pruebas basadas en las poses del brazo. El objetivo de la misma es el de poder identificar si hay ciertas poses que favorezcan los resultados, como tambien identificar aquellas que hacen que los resultados empeoren. Como se dijo anteriormente, todas las escenas fueron grabadas utilizando cuatro poses diferentes (las mismas pueden verse en la Figura 5.1) La prueba consiste en calibrar en cada escena, luego agrupar los resultados segun la pose del brazo y por ultimo graficar los resultados. En la Figura 5.7 puede verse el diagrama de caja para los resultados obtenidos en los cuatro grupos. Mientras que en la Figura 5.8 se ve los resultados por cada grupo luego de realizar la calibración.

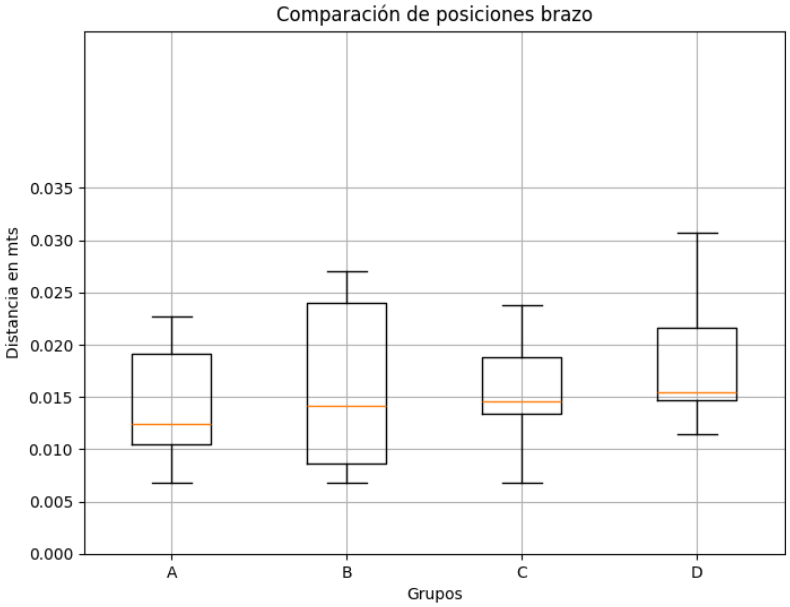


Figura 5.7: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara previo de calibración. Se dividen en cuatro grupos según las poses del brazo robótico como se explica en la Figura 5.1

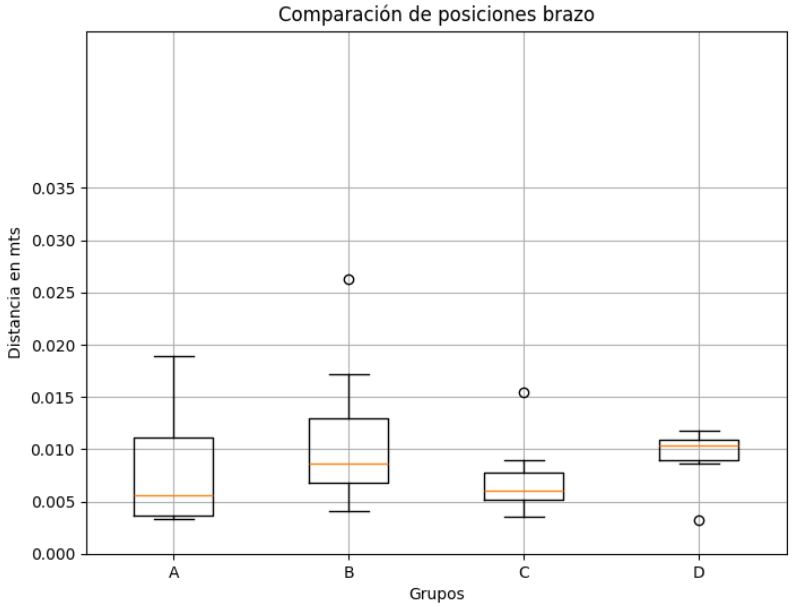


Figura 5.8: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara luego de calibración. Se dividen en cuatro grupos según las poses del brazo robótico como se explica en 5.1

En todos los casos se observa una reducción significativa en la mediana del error, así como una disminución general en la dispersión de los datos. Esta mejora se mantiene incluso en el grupo B, que presenta una mayor cantidad de muestras y, por lo tanto, una mayor variabilidad. A pesar de esta variabilidad, el error final en todos los grupos se encuentra dentro de un rango considerado aceptable para las aplicaciones previstas, lo cual muestra cierta robustez del método sobre las diferentes poses.

5.6. Experimento 4: Evaluación bajo perturbaciones

Como se explica en la Subsección 4.1.2, debido a la forma en que el problema es formulado, no se garantiza que se converja a la solución real. Por este motivo, se plantean distintas pruebas con el fin de explorar los efectos de esta limitación y analizar de qué forma pueden minimizarse los problemas que conlleva. El proceso de calibración paramétrica puede ser interpretado de dos maneras. En un primer enfoque, se busca encontrar el vector de parámetros que representa fielmente la realidad física, es decir, la descalibración presente en cada enlace. Este enfoque es transparente para el operador del robot, ya que permite identificar con precisión dónde se encuentra el error y cómo puede ser corregido físicamente. En cambio, en un segundo enfoque, no se busca necesariamente una representación exacta de la realidad, sino un conjunto de parámetros que permita al sistema interactuar correctamente con su entorno. Aunque idealmente se desea que la solución coincida con la configuración física real, en determinadas circunstancias, este segundo enfoque puede considerarse válido. Para evaluar si la solución obtenida por el algoritmo es representativa de la realidad, se

consideran dos alternativas. En primer lugar, podría verificarse si la solución corresponde al estado real del sistema. Sin embargo, esta opción requiere contar con un *ground truth*, lo cual contradice el objetivo de mantener el método simple y sin intervención externa. La segunda opción consiste en calibrar el sistema con una escena determinada para obtener un vector solución $\boldsymbol{\theta}_1$, introducir luego una perturbación conocida (por ejemplo, un desplazamiento) en uno de los enlaces, y volver a calibrar para obtener un nuevo vector $\boldsymbol{\theta}_2$. Si el método es robusto, la diferencia entre $\boldsymbol{\theta}_1$ y $\boldsymbol{\theta}_2$ debería reflejar el desplazamiento introducido, afectando únicamente los parámetros asociados al enlace modificado. Dado que en experimentos previos se observa que el método no sufre de sobreajuste, se utiliza una única escena para estos ensayos, asumiendo que el comportamiento se mantiene. La escena seleccionada corresponde a estante2Botella (La pose del brazo corresponde al grupo B según la Figura 5.1). Primero, la cadena cinemática es calibrada con la escena original. El resultado obtenido (θ_1) se muestra en la segunda columna de la Tabla 5.2. A continuación, es introducido un desplazamiento de 0,03 m en el eje y sobre el enlace `chest_camera_left_frame`, lo cual provoca que los objetos percibidos por la cámara del pecho sean visualizados desplazados en dicha dirección. Luego, es realizada una nueva calibración considerando este nuevo estado, cuyos resultados (θ_2) son presentados en la tercera columna. Al observar la quinta columna ($|\theta_2 - \theta_1|$), se aprecia que no se alcanza la solución esperada en esta calibración. Sin embargo, se observa que los desplazamientos en el eje y de los enlaces de ambas cámaras se aproximan a 1 cm, en direcciones opuestas (una hacia arriba y otra hacia abajo). Esto evidencia la limitación del método, ya que el error es distribuido entre los distintos enlaces (lo mismo ocurre para las articulaciones) hasta alcanzar un desplazamiento acumulado de 0,03 m. Por último, se realiza un nuevo experimento variando el vector inicial: en este caso, es utilizado un vector nulo para todas las coordenadas, excepto para la coordenada y asociada al enlace de la cámara del pecho, la cual es inicializada en 0,04 m. Los resultados luego de este nuevo proceso de calibración (θ_3) son mostrados en la cuarta columna. Al comparar este nuevo resultado con el original en la sexta columna de la tabla ($|\theta_1 - \theta_3|$), es observado un mayor acercamiento a la solución esperada. Esto demuestra que una inicialización cercana a la solución real mejora los resultados obtenidos.

Parámetro	θ_1	θ_2	θ_3	$\lvert\theta_2 - \theta_1\rvert$	$\lvert\theta_1 - \theta_3\rvert$
Cámara del pecho (Traslaciones XYZ)					
$\mathbf{p_0}$ (tras. pechoX)	-0.00122614	-0.00024304	0.00078672	0.00098310	0.00201286
$\mathbf{p_1}$ (tras. pechoY)	0.00940158	-0.00075628	0.04387904	**0.01015786**	**0.03447746**
$\mathbf{p_2}$ (tras. pechoZ)	0.00266708	0.00082677	0.00207506	0.00184031	0.00059202
Articulaciones (Rotaciones alrededor de sus ejes)					
$\mathbf{p_3}$ (rot. art. 1)	-0.04736292	-0.01278366	0.20850055	0.03457926	0.25586347
$\mathbf{p_4}$ (rot. art. 2)	-0.02172199	-0.00885706	0.07742631	0.01286493	0.09914830
$\mathbf{p_5}$ (rot. art. 3)	0.00567601	0.00202423	-0.03765338	0.00365178	0.04332939
$\mathbf{p_6}$ (rot. art. 4)	-0.00195439	0.00167316	-0.06088928	0.00362755	0.05893489
$\mathbf{p_7}$ (rot. art. 5)	0.01351228	0.03496058	-0.00419484	0.02144830	0.01770712
$\mathbf{p_8}$ (rot. art. 6)	-0.00645020	-0.01039945	0.07486583	0.00394925	0.08131603
Cámara del brazo (Traslaciones XYZ, luego rotaciones XYZ)					
$\mathbf{p_9}$ (tras. brazo X)	-0.00433810	0.00055714	-0.00572236	0.00378096	0.00138426
$\mathbf{p_{10}}$ (tras. brazo Y)	-0.01099894	0.00034395	-0.00214685	**0.01134289**	0.00885209
$\mathbf{p_{11}}$ (tras. brazo Z)	0.00017795	0.00080429	-0.00630248	0.00062634	0.00648043
$\mathbf{p_{12}}$ (rot. brazoX)	-0.03956446	-0.01190972	0.07323365	0.02765474	0.11279811
$\mathbf{p_{13}}$ (rot. brazoY)	0.00569581	-0.02620221	-0.01204573	0.03189802	0.01774154
$\mathbf{p_{14}}$ (rot. brazoZ)	0.08099838	0.05970042	-0.04365789	0.02129796	0.12465627

Tabla 5.2: Comparación de parámetros calibrados (θ_1, θ_2, θ_3) y sus diferencias absolutas. θ_1 corresponde al resultado de calibrar en la escena original; θ_2 al resultado de calibrar en una segunda escena con ruido añadido en el enlace de la cámara del pecho; θ_3 a la calibración sobre esta misma segunda escena, pero iniciando desde un vector inicial distinto. Los parámetros están organizados en tres grupos coloreados: cámara del pecho (verde claro), articulaciones del brazo (azul claro) y cámara del brazo (gris claro).

Se repite el experimento anterior calibrando únicamente los parámetros asociados a las cámaras, mientras que los parámetros correspondientes a las articulaciones del brazo se mantienen fijos, asumiendo que ya fueron calibrados con precisión.

Los resultados obtenidos se resumen en la Tabla 5.3. En esta, se presentan tres ejecuciones distintas, donde:

- $\boldsymbol{\theta_1}$: Corresponde a la calibración con la escena original (sin desplazamientos).
- $\boldsymbol{\theta_2}$: Corresponde a la calibración con la cámara del pecho desplazada artificialmente +3 cm en el eje y.
- $\boldsymbol{\theta_3}$: Igual a θ_2, pero restrigiendo más la región de confianza sobre la cual puede desplazarse el método.

Parámetro	$\boldsymbol{\theta_1}$	$\boldsymbol{\theta_2}$	$\boldsymbol{\theta_3}$	$\lvert\theta_2 - \theta_1\rvert$	$\lvert\theta_1 - \theta_3\rvert$
Cámara del pecho (Traslaciones XYZ)					
$\mathbf{p_0}$ (tras. pechoX)	0.00324463	0.00627399	0.00307696	0.00302936	0.00016767
$\mathbf{p_1}$ (tras. pechoY)	0.01178820	0.04555339	-0.02093351	**0.03376519**	**0.03272171**
$\mathbf{p_2}$ (tras. pechoz)	0.00170224	-0.01178177	0.00163039	0.01348401	0.00333263
Cámara del brazo (Traslaciones XYZ, luego rotaciones XYZ)					
$\mathbf{p_3}$ (tras. brazoX)	0.00314204	0.01013612	-0.00350456	0.00699408	0.00664660
$\mathbf{p_4}$ (tras. brazoY)	-0.02108622	0.04277868	-0.02387047	**0.06386490**	0.00278425
$\mathbf{p_5}$ (tras. brazoZ)	-0.00128754	-0.00266246	0.00606405	0.00137492	0.00735159
$\mathbf{p_6}$ (rot. brazoX)	-0.00981627	-0.00981711	-0.00981710	0.00000084	0.00000083
$\mathbf{p_7}$ (rot. brazoY)	0.00173611	0.00173234	0.00173239	0.00000377	0.00000372
$\mathbf{p_8}$ (rot. brazoZ)	0.04448173	0.04448200	0.04448200	0.00000027	0.00000027

Tabla 5.3: Valores de los 9 parámetros obtenidos en diferentes ejecuciones del proceso de calibración, junto con las diferencias absolutas entre los resultados. La unidad de las traslaciones es el Metro y el de las rotaciones el Radián.

Para esta segunda prueba, luego de realizar la calibración para el caso sin ningún tipo de perturbación (θ_1) y el caso en que la cámara del pecho percibe los objetos 0,03 m desplazados positivamente en y, se puede observar que la cámara del brazo se desplaza 0,06 m (comparado con el caso θ_1). Mientras que la del pecho lo hace 0,03 m. Esto confirma que el algoritmo identifica correctamente la fuente del error y lo distribuye entre los parámetros relevantes.

Si se desea limitar aún más el espacio de búsqueda, se puede restringir los parámetros a un rango, por ejemplo, ±0,035 m. En ese caso, el resultado obtenido (cuarta columna de la Tabla 5.3) se aproxima al vector original ajustado por el desplazamiento conocido (como se puede validar en la quinta columna de esa misma tabla). Esto muestra que con una limitación de los parametros del modelo precisa, se puede aproximar correctamente la solución del problema.

Finalmente, se realiza una última prueba en la que se calibra el sistema utilizando la misma escena empleada previamente, es decir el caso donde se estima θ_1. A continuación, se asume que el error se encuentra únicamente en el enlace afectado (en este caso, el correspondiente a la cámara del pecho). Por lo tanto, se lleva a cabo una nueva calibración considerando exclusivamente dicho enlace, utilizando como punto de partida la información obtenida en la última calibración. El resultado obtenido es:

$$\boldsymbol{\theta} = \left(2{,}05049730 \times 10^{-9},\ -2{,}99999976 \times 10^{-2},\ 1{,}45319778 \times 10^{-9}\right)$$

Este resultado proporciona información valiosa: si en algún momento se alcanza un conjunto de parámetros que permite una operación correcta del sistema, cualquier desplazamiento físico posterior (por ejemplo, el movimiento de una cámara) puede corregirse mediante la recalibración exclusiva del enlace afectado, sin comprometer la consistencia de la estimación global.

Este conjunto de experimentos permite extraer varias conclusiones relevantes. En primer lugar, la limitación discutida en la Subsección 4.1.2 impide garantizar la convergencia a la solución real. No obstante, una buena inicialización, combinada con el uso de restricciones, mejora significativamente la precisión de la estimación. También se comprueba que el método es capaz de identificar y estimar correctamente cambios conocidos en los enlaces, especialmente cuando se limita el número de parámetros involucrados en la optimización. Todo esto permite establecer un marco de trabajo útil: si se cuenta con una solución válida en un momento dado, cualquier desplazamiento físico

posterior puede ser compensado por el algoritmo mediante la recalibración del enlace afectado, sin necesidad de realizar una nueva calibración global.

5.7. Experimento 5: Pruebas físicas

Hasta este punto, todas las pruebas se realizaron utilizando archivos `bag`, los cuales resultan altamente confiables debido a la calidad de los datos registrados y al modo en que fueron capturados. Estos permiten simular de forma razonable un entorno real.

No obstante, dado que el objetivo final del sistema es que el robot interactúe de manera efectiva con los objetos de su entorno, es necesario validar empíricamente que dicho objetivo se cumpla en condiciones reales. Por ello, se requiere una etapa de calibración utilizando el robot en funcionamiento, con un doble propósito: por un lado, comprobar que el método propuesto es aplicable y efectivo en la práctica; y por otro, confirmar que los resultados obtenidos en las pruebas simuladas son válidas.

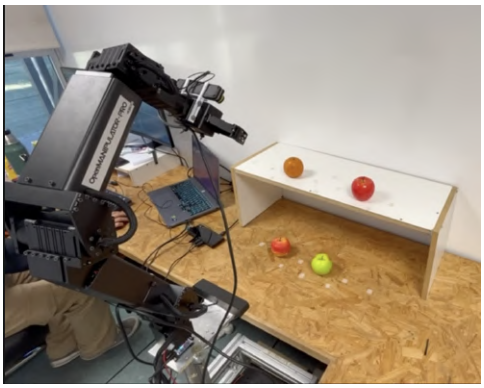
La prueba fue realizada en el Instituto de Computación de la Facultad de Ingeniería. En la Figura 5.9 (a y b) se muestran imágenes de la escenas de prueba.

El experimento consistió en dos fases:

1. **Sin calibración:** el robot manipulador observó los objetos del entorno (cuatro manzanas) con ambas cámaras: la del brazo y la del torso. Las posiciones estimadas fueron publicadas en los tópicos `/yolo_arm_object_detections` y `/yolo_chest_object_detections`. Estas posiciones fueron leídas por nuestro nodo de calibración, el cual ya contaba con la estructura cinemática del robot, disponible mediante los tópicos `/tf` y `/tf_static`, publicados por el nodo `robot_state_publisher`. A partir del promedio coordenada a coordenada de las posiciones estimadas por ambas cámaras para cada objeto, se ordenó al brazo para que intentara tomar cada manzana. En el siguiente enlace puede verse una demo de este caso.

2. **Con calibración:** se repitió el procedimiento anterior, pero antes de intentar tomar los objetos se ejecutó el proceso de calibración. Luego, las posiciones de los objetos se calcularon del mismo modo, pero utilizando las transformaciones corregidas por la calibración obtenida. En el siguiente enlace puede verse una demo de este caso.

En la primera fase, el robot fue incapaz de tomar correctamente ninguna de las cuatro manzanas. En cambio, en la segunda fase, luego de calibrar, logró tomar exitosamente tres de las cuatro manzanas. Este resultado muestra una mejora considerable en la capacidad de interacción del sistema robótico cuando se aplica calibración, y a su vez, sirve como una validación empírica del método y de las pruebas previas realizadas en simulación.



(a) Vista general de la escena previa a que el robot intente tomar los objetos.

(b) Vista general de la escena mientras el robot intenta tomar una de las frutas.

Figura 5.9: Imágenes correspondientes a la prueba física realizada en el laboratorio.

Capítulo 6

Conclusiones y trabajo futuro

Conclusiones

En este trabajo se exploró un método de calibración cinemática basado en la técnica de bundle adjustment, aprovechando la redundancia perceptiva generada por dos cámaras ubicadas en distintas partes del robot. El enfoque propuesto presenta una serie de ventajas respecto del estado del arte. El procedimiento de calibración es más simple, ya que utiliza elementos del entorno capturados mediante visión artificial, en lugar de requerir equipamiento especializado. Además, el sistema tiene menores requerimientos de hardware, siendo suficiente el uso de dos cámaras económicas.

También se identificaron algunas limitaciones del método. Por ejemplo, el punto utilizado para la calibración se estima como el centro del bounding box detectado por cada cámara. Debido a que las cámaras observan el objeto desde distintos ángulos, el punto detectado puede no coincidir, lo cual introduce error. Existen ciertos objetos cuya detección resulta más perjudicial debido a su ambigüedad o forma irregular. Es necesario que ambas cámaras tengan visión simultánea del mismo objeto, lo cual restringe ciertas configuraciones del entorno. Los bucles de corrección se generan en base a la semántica; por tanto, la calidad del sistema de visión artificial para identificar clases de objetos afecta directamente al desempeño. Se detectó una limitación teórica del método que puede impedir la convergencia hacia la verdadera solución. Sin embargo, se observó que el sistema aún converge hacia una solución funcional para la interacción física con el entorno. Se concluyó que este inconveniente puede mitigarse trabajando con una versión restringida del problema (por ejemplo, acotando los parámetros), y utilizando un solver basado en regiones de confianza, en lugar del ampliamente utilizado Levenberg–Marquardt, el cual es menos adecuado en presencia de restricciones.

Por último, se validaron empíricamente los resultados mediante una prueba física con el robot Open Manipulator Pro equipado con dos cámaras ZED. Se demostró que el método de calibración propuesto supera significativamente al caso sin calibración.

Trabajo futuro

Como líneas de trabajo futuro se propone explorar sistemas de visión artificial con arquitectura en dos etapas, los cuales poseen mayor precisión en la clasificación semántica de objetos. También

se investigarán enfoques alternativos como el uso de nubes de puntos 3D, lo que permitiría estimar centroides reales de los objetos y así reducir el error introducido por los offsets, como se discutió en la Sección 5.2. Finalmente, se buscará aplicar el método en plataformas robóticas aún más simples, con el objetivo de demostrar la generalidad y simplicidad del enfoque.

Referencias

Baeldung on Computer Science. (2024). *What is bundle adjustment?* `https://www.baeldung.com/cs/computer-vision-bundle-adjustment`. (Consultado el 15 de julio de 2025)

Birbach, O., Frese, U., y Bäuml, B. (2015). Rapid calibration of a multi-sensorial humanoid's upper body: An automatic and self-contained approach. *The International Journal of Robotics Research*, *34*(4-5), 420–436.

Hagberg, A. A., Schult, D. A., y Swart, P. J. (2024). *Networkx: Network analysis in python.* Descargado de `https://networkx.org/` (Accessed: 2025-07-16)

Hollerbach, J. M., Khalil, W., y Gautier, M. (2016). Model identification. En B. Siciliano y O. Khatib (Eds.), *Springer handbook of robotics* (2.ª ed., pp. 113–138). Springer. doi: 10.1007/978-3-319-32552-1_6

Ikits, M., y Hollerbach, J. M. (1997). Kinematic calibration using a plane constraint. En *Proceedings of international conference on robotics and automation* (Vol. 4, pp. 3191–3196).

Kastner, T., Röfer, T., y Laue, T. (2015). Automatic robot calibration for the nao. En *Robocup 2014: Robot world cup xviii 18* (pp. 233–244).

Pradeep, V., Konolige, K., y Berger, E. (2014). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. En *Experimental robotics: The 12th international symposium on experimental robotics* (pp. 211–225).

Roncone, A., Hoffmann, M., Pattacini, U., y Metta, G. (2014). Automatic kinematic chain calibration using artificial skin: self-touch in the icub humanoid robot. En *2014 ieee international conference on robotics and automation (icra)* (pp. 2305–2312).

Roth, Z. S., Mooring, B. W., y Ravani, B. (1987). An overview of robot calibration. *IEEE Journal on Robotics and Automation*, *3*(5), 377–385. doi: 10.1109/JRA.1987.1087124

ScienceDirect Topics. (s.f.). *Kinematic chain - an overview.* Descargado de `https://www.sciencedirect.com/topics/engineering/kinematic-chain`

Stepanova, K., Pajdla, T., y Hoffmann, M. (2019). Robot self-calibration using multiple kinematic chains—a simulation study on the icub humanoid robot. *IEEE Robotics and Automation Letters*, *4*(2), 1900–1907.

Stepanova, K., Rozlivek, J., Puciow, F., Krsek, P., Pajdla, T., y Hoffmann, M. (2022). Automatic self-contained calibration of an industrial dual-arm robot with cameras using self-contact, planar constraints, and self-observation. *Robotics and Computer-Integrated Manufacturing*, *73*, 102250.

Švaco, M., Šekoranja, B., Šuligoj, F., y Jerbić, B. (2014). Calibration of an industrial robot using a stereo vision system. *Procedia Engineering*, *69*, 459–463.

Triggs, B., McLauchlan, P. F., Hartley, R. I., y Fitzgibbon, A. W. (2000). Bundle adjustment – a modern synthesis. En B. Triggs, A. Zisserman, y R. Szeliski (Eds.), *Vision algorithms: Theory*

and practice, proceedings of the international workshop on vision algorithms (Vol. 1883, pp. 298–372). Springer. doi: 10.1007/3-540-44480-7_21

Ultralytics. (2024). *Yolov8 — a new family of state-of-the-art real-time object detection models.* `https://yolov8.com/`. (Consultado el 17 de julio de 2025)

Anexo A

Anexo

Este capítulo presenta información complementaria a lo expuesto a lo largo del documento. Contiene definiciones, resultados de la fase experimental, gráficos y otros elementos que, aunque relevantes, no son esenciales para comprender los resultados principales ni continuar con la investigación.

A.1. Representación Denavit-Hartenberg

Los parámetros de Denavit-Hartenberg (DH) son una convención utilizada en robótica para estandarizar la forma en que se describen los eslabones y las articulaciones de un brazo robótico. Esta convención facilita el análisis cinemático de los robots mediante el uso de transformaciones homogéneas. Los cuatro parámetros DH son: θ (theta), d (distancia), a (longitud del eslabón) y α (ángulo).

- Ángulo de articulación (θ) - Es el ángulo de rotación alrededor del eje z_{i-1} para alinear x_{i-1} con x_i.Para articulaciones rotacionales, θ_i es variable y representa el ángulo de la articulación.Para articulaciones prismáticas, θ_i es constante.
- Distancia de articulación (d) - Es la distancia entre el origen del sistema de coordenadas $i-1$ y el origen del sistema de coordenadas i medida a lo largo del eje z_{i-1}.Para articulaciones prismáticas, d_i es variable y representa la distancia de deslizamiento.Para articulaciones rotacionales, d_i es constante.
- Longitud del eslabón (a) - Es la distancia entre el eje z_{i-1} y el eje z_i medida a lo largo del eje x_i.Representa la longitud física del eslabón.
- Ángulo de torsión (α) - Es el ángulo entre los ejes z_{i-1} y z_i medido alrededor del eje x_i.Indica cómo están orientados los ejes z entre dos eslabones consecutivos.

En la Figura A.1 se puede ver un ejemplo de cómo se asignan los parámetros DH a un modelo robótico.

Las transformaciones homogéneas se utilizan para describir la posición y orientación de un eslabón en relación con el eslabón anterior en una cadena cinemática. Una transformación homogénea es una matriz 4×4 que combina una rotación y una traslación en una sola operación. En el contexto

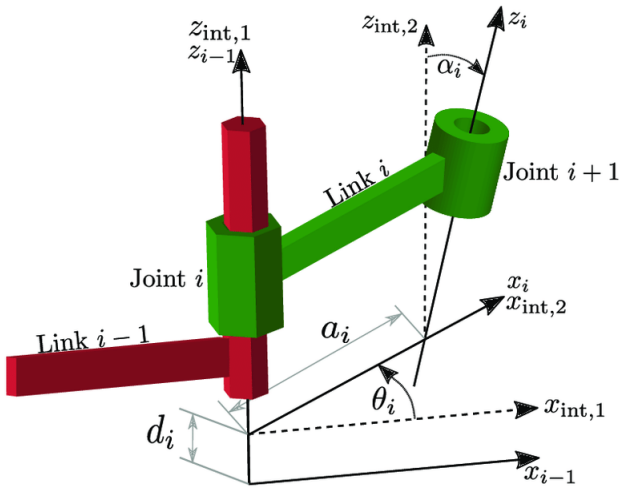


Figura A.1: Asignación de los parámetros de DH.

de la convención DH, la matriz de transformación homogénea entre dos sistemas de coordenadas adyacentes $i-1$ e i se define como:

$$A_i = \begin{pmatrix} \cos(\theta_i) & -\sin(\theta_i)\cos(\alpha_i) & \sin(\theta_i)\sin(\alpha_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i)\cos(\alpha_i) & -\cos(\theta_i)\sin(\alpha_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Para determinar la posición y orientación del efector final del robot en relación con la base, multiplicamos las matrices de transformación homogénea de cada eslabón de la cadena cinemática:

$$T = A_1 \cdot A_2 \cdot \ldots \cdot A_n$$

Donde T es la matriz de transformación total desde el sistema de coordenadas base hasta el efector final.

A.1.1. Calibración de una cámara

La calibración de la cámara tiene como objetivo determinar los parámetros geométricos del proceso de formación de imágenes. En estas aplicaciones, la cámara se modela con un conjunto de parámetros intrínsecos (distancia focal, punto principal, sesgo del eje) y su orientación se expresa mediante parámetros extrínsecos (rotación y traslación).

Reproyección

La reproyección se refiere al proceso de proyectar puntos 3D del mundo real de vuelta a la imagen 2D capturada por la cámara, utilizando los parámetros de la cámara. Durante la reproyección, se usan las coordenadas del mundo real y se aplican las transformaciones de cámara para obtener las coordenadas en el plano de imagen.

Modelo pinhole

Describe la relación matemática de la proyección de puntos en el espacio 3D sobre un plano de imagen. Sea el centro de proyección el origen de un sistema de coordenadas euclidianas, y el plano $Z = f$, que se llama plano de imagen o plano focal. Bajo este modelo, un punto en el espacio con coordenadas $(X, Y, Z)^T$ se mapea al punto en el plano de imagen $\left(\frac{fX}{Z}, \frac{fY}{Z}, f\right)^T$ usando triángulos, como se muestra en la Figura A.2. Ignorando la coordenada final de la imagen, la proyección central del espacio 3D a las coordenadas 2D de la imagen es

$$(X, Y, Z)^T \longrightarrow \left(\frac{fX}{Z}, \frac{fY}{Z}\right)^T$$

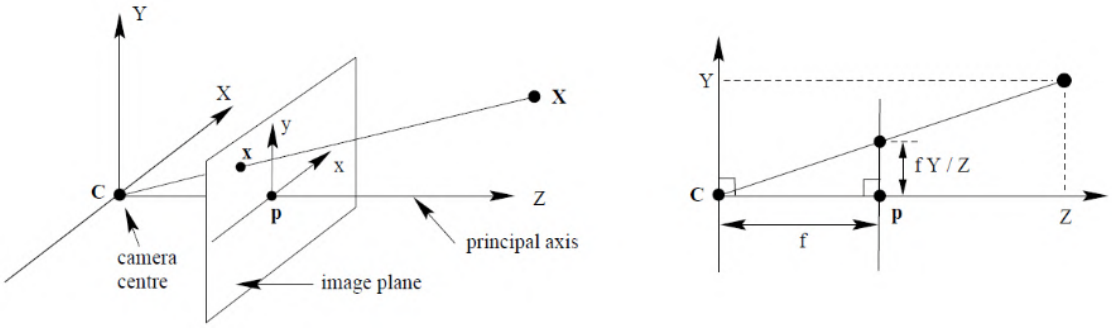


Figura A.2: Ilustración del modelo pinhole

Distancia focal

La distancia focal es uno de los parámetros intrínsecos más importantes en la calibración de cámaras. Representa la distancia entre el centro óptico de la lente y el sensor de imagen, donde la luz converge para formar una imagen nítida. La distancia focal afecta el campo de visión de la cámara: una distancia focal más corta proporciona un campo de visión más amplio, mientras que una distancia focal más larga proporciona un campo de visión más estrecho y un mayor aumento.

A.2. Formato común de mensajes en tópicos

Como se explicó en la Figura 4.3, se define un formato común de los mensajes publicados en los tópicos cam_A_object_positions y en cam_B_object_positions. De esta manera el nodo de calibración no se acopla a un formato particular en la detección de las posiciones de los objetos, sino que deja de lado de quien desea interactuar con este nodo, la responsabilidad de implementar esta interfaz. Por esto, definimos el tipo de dato DetectedObject. La estructura general que representa la posición de cada objeto es la siguiente:

- **label**: String que representa el nombre de la clase del objeto.

- **pos**: Array de 3 elementos de tipo real que representan la posicion relativa del objeto a la cámara.
- **rot**: Array de 3 elementos de tipo real que representan la rotación relativa del objeto a la cámara.

Por lo tanto, en cada tópico deberá publicarse un array de objetos de este tipo. Para esto en ROS definimos el tipo DetectedObjectArray el cual cuenta con un único campo llamado "objects" .

A.3. Pseudocódigo de la función foward_kinematic

1. **Definición:**
   - `forward_kinematic(objeto, camara, params, offset)`
2. **Inicializacion:**
   - `node = camara` //Se empieza a recorrer la cadena cinemática desde la camara hasta el marco base.
   - `position = get_position(ct.POSITIONS[camara][objeto], offset)` //Obtengo la posición del objeto respecto de la camara.
   - `homogenea = obtener_homogenea(position, np.zeros(3))` // Retorna la matriz homogénea asociada a esa transformación.
   - `i = 1`
3. **Loop principal (While `node` $\neq$ `ct.ROOT`):**
   - Increment `i`
   - `predecesores = gf.obtener_predecesores(node)`
   - `predecesor = predecesores[0]` //Obtenemos el nodo predecesor (Como por restricción la cadena cinemática es un árbol sabemos que "predecesores"solo tiene un nodo.
   - `(rot, tras) = transformacion_predecesor(predecesor, node)`
   - `hi = obtener_homogenea(tras, rot)` // Genera la nueva matriz homogénea. En el caso que corresponda al vector de traslación o rotación le suma el vector de parámetros.
   - `homogenea = np.matmul(hi, homogenea)`
   - `node = predecesor`
4. **Valor de retorno:**
   - Return `homogenea`

A.4. Cámaras utilizadas en experimentación

Se utilizaron dos cámaras cámaras estereoscópicas con percepción espacial.

A.4.1. Zed 2

Figura A.3: Cámara binocular ZED.

A.4.2. Zed mini

Figura A.4: Cámara binocular ZED mini.

A.5. Resultados complementarios Experimento 1

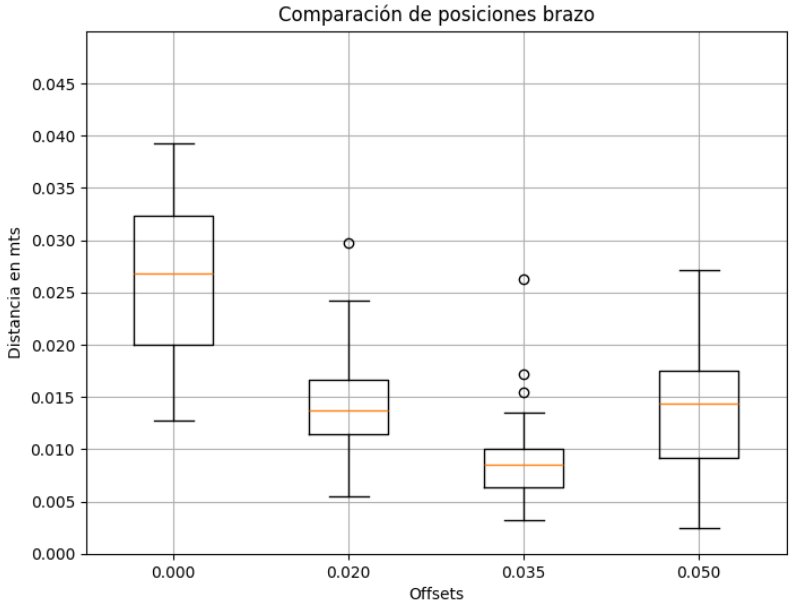


Figura A.5: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara luego de la calibración usando el solver lm. Los resultados son agrupados por diferentes valores de offset.

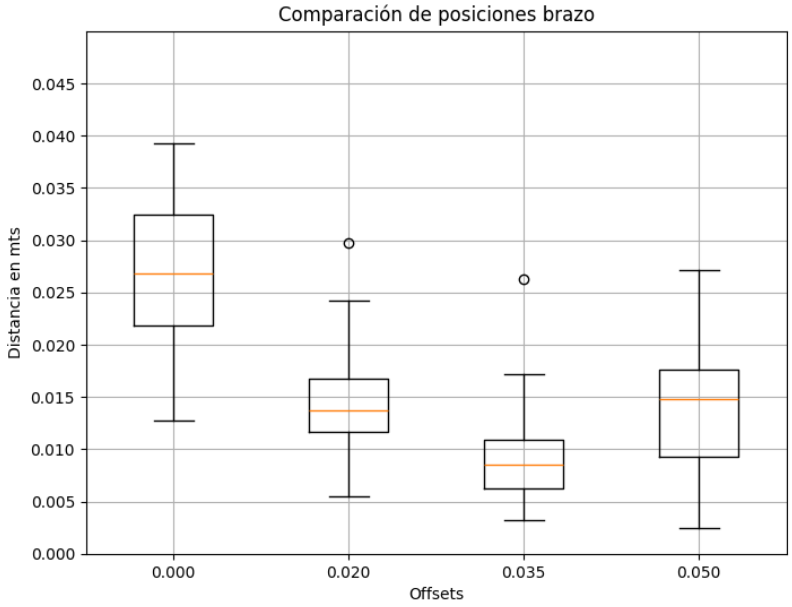


Figura A.6: Boxplot de las distancias entre las posiciones de los objetos vistas por cada cámara luego de la calibración usando el solver dogbox. Los resultados son agrupados por diferentes valores de offset.