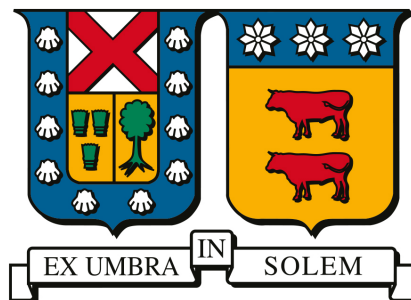


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA  
DEPARTAMENTO DE INFORMÁTICA  
SANTIAGO - CHILE



“OPTIMIZACIÓN DE UN SISTEMA DE SEGUIMIENTO DE  
MANOS PARA LA INTERACCIÓN 3D EN UN  
LABORATORIO VIRTUAL DE QUÍMICA: CASO A *DARLE*  
*ÁTOMOS*”

FABIAN LORCA

MEMORIA PARA OPTAR AL TÍTULO DE  
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Roberto León, PhD.  
Profesor Correferente: Claudio Lobos, PhD.

Agosto - 2025



## CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

### 1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

**Tipo de monografía (marcar una opción):** ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

**Título del trabajo:** Optimización de un sistema de seguimiento de manos para la interacción 3D en un laboratorio virtual de química: Caso A darle Átomos

**Nombre del candidato(a):** Fabián Andrés Lorca Trujillo

**Carrera / Grado:** Ingeniería civil en informática

**Campus:** Santiago San Joaquin ; **Departamento:** Informática

### 2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Roberto León, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

### 3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

### 4.- FIRMAS

**Profesor(a) guía o director(a) de memoria o tesis:**

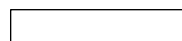
**Fecha:** 22/08/2025 ;

**Firma:**

**Estudiante o Candidato(a):**

**Fecha:** 22/08/2025 ; **Firma:**

*Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.*



## **DEDICATORIA**

A mi familia que siempre me ha apoyado en todo momento, especialmente a mi madre y mi abuela que siempre me han alentado a jamás claudicar sin importar las adversidades.

## **AGRADECIMIENTOS**

En primer lugar, me gustaría expresar mi agradecimiento a mi profesor guía, Roberto León, por su invaluable orientación y paciencia a lo largo del desarrollo de este trabajo.

Mi agradecimiento se extiende a mis amigos de la universidad por acompañarme durante este largo viaje y entregarme buenos momentos que siempre recordaré.

Finalmente agradecer al equipo, Ultimate Spider Monke, que hizo posible la creación de A Darle Átomos.

## RESUMEN

**Resumen—** En el marco de la feria de Software de la UTFSM, se ha desarrollado A Darle Átomos, un laboratorio virtual de química que utiliza un sistema de seguimiento de manos para la interacción en el entorno virtual. No obstante, los problemas presentes en el sistema dificultan la entrega de una experiencia fluida. Para solucionar esto, se propone una serie de cambios en los procesos que conforman este sistema. Esto se ha logrado mediante la implementación de diversos métodos y la paralelización de algunas tareas clave. Gracias a los cambios propuestos, se ha logrado un aumento de aproximadamente un 400 % en el rendimiento del sistema y se ha mejorado la precisión y fluidez del seguimiento de manos. Estas mejoras generarán un impacto positivo en la experiencia entregada por la aplicación, aportando una mayor inmersión y facilidad de uso.

**Palabras Clave—** Seguimiento de manos, MediaPipe, DLT, Cinemática Inversa, Feria de Software.

## ABSTRACT

**Abstract—** Within the framework of the UTFSM Software Fair, A Darle Átomos has been developed, a virtual chemistry laboratory that uses a hand tracking system for interaction in the virtual environment. However, issues present in the system make it difficult to deliver a smooth experience. To solve this, a series of changes are proposed in the processes that make up this system. This has been achieved by implementing various methods and parallelizing some key tasks. Thanks to the proposed changes, an increase of approximately 400 % in system performance has been achieved and the accuracy and fluidity of hand tracking has been improved. These improvements will generate a positive impact on the experience delivered by the application, providing greater immersion and ease of use.

**Keywords—** Hand Tracking, MediaPipe, DLT, Inverse Kinematics, Software fair.

## GLOSARIO

BA: Bundle Adjustment  
CP: Centro de perspectiva  
CPU: Central Processing Unit  
DLS: Direct Least-Squares  
DLT: Direct Linear Transform.  
 $EP_nP$ : Efficient  $P_nP$   
FIFO: First-In First-Out  
FPS: Fotogramas por segundo  
GDL: Grado De Libertad  
LDP: Location Determination Problem  
 $OP_nP$ :  $O(n)$   $P_nP$   
 $P_nP$ : Perspective n Point  
RAM: Random Access Memory  
RDLT: Robust Direct Linear Transform  
RMSE: Root Mean Square Error  
SVD: Singular Value Decomposition  
UTFSM: Universidad Técnica Federico Santa María.

# ÍNDICE DE CONTENIDOS

|                                                |     |
|------------------------------------------------|-----|
| RESUMEN                                        | IV  |
| ABSTRACT                                       | IV  |
| GLOSARIO                                       | V   |
| ÍNDICE DE FIGURAS                              | VII |
| ÍNDICE DE TABLAS                               | VII |
| INTRODUCCIÓN                                   | 1   |
| CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA            | 3   |
| 1.1 Objetivos . . . . .                        | 5   |
| 1.1.1 Objetivo general . . . . .               | 5   |
| 1.1.2 Objetivos específicos . . . . .          | 6   |
| CAPÍTULO 2: MARCO CONCEPTUAL                   | 7   |
| 2.1 MediaPipe . . . . .                        | 7   |
| 2.2 Interpolación . . . . .                    | 8   |
| 2.3 Perspective-n-Point . . . . .              | 9   |
| 2.4 Direct Linear Transform . . . . .          | 10  |
| 2.4.1 Robust Direct Linear Transform . . . . . | 11  |
| 2.5 Calibración de cámaras . . . . .           | 16  |
| 2.6 Bundle adjustment . . . . .                | 17  |
| 2.7 Filtro Kalman . . . . .                    | 17  |
| 2.8 Cinemática . . . . .                       | 18  |
| 2.8.1 Cinemática inversa . . . . .             | 21  |
| CAPÍTULO 3: PROPUESTA DE SOLUCIÓN              | 23  |
| 3.1 Calibración de cámaras . . . . .           | 24  |
| 3.2 Captura de manos . . . . .                 | 25  |
| 3.3 Cinemática inversa . . . . .               | 28  |
| CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN          | 31  |
| 4.1 Especificaciones de hardware . . . . .     | 31  |
| 4.2 Calibración de las cámaras . . . . .       | 31  |
| 4.3 Captura de manos . . . . .                 | 34  |
| 4.4 Cinemática inversa . . . . .               | 35  |
| CAPÍTULO 5: CONCLUSIONES                       | 38  |
| 5.1 Trabajo futuro . . . . .                   | 39  |
| REFERENCIAS BIBLIOGRÁFICAS                     | 40  |

## ÍNDICE DE FIGURAS

|    |                                                                                                                                                                                 |    |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1  | Árbol del problema . . . . .                                                                                                                                                    | 3  |
| 2  | Arquitectura del sistema de seguimiento de manos actual. Fuente: Elaboración propia . . . . .                                                                                   | 4  |
| 3  | Configuración del sistema estereoscópico. Fuente: Elaboración propia . . . . .                                                                                                  | 5  |
| 4  | Ejemplo de un <i>perception pipeline</i> para la detección de puntos de referencia en un rostro. Fuente: [Lugaresi <i>et al.</i> , 2019] . . . . .                              | 7  |
| 5  | Puntos de referencia de una mano. Fuente: [Google, 2025] . . . . .                                                                                                              | 8  |
| 6  | Modelo geométrico del problema. Fuente: [Fischler y Bolles, 1981] . . . . .                                                                                                     | 9  |
| 7  | Modelo geométrico del problema. Fuente: [Wang <i>et al.</i> , 2024] . . . . .                                                                                                   | 11 |
| 8  | Ejemplo de una cadena cinemática. Fuente: Elaboración propia . . . . .                                                                                                          | 19 |
| 9  | Visualización de los pivotes en una cadena cinemática. Fuente: Elaboración propia . . . . .                                                                                     | 20 |
| 10 | Ejemplo de la geometría espacial de un manipulador simple. Fuente: [Kucuk y Bingul, 2006]. . . . .                                                                              | 21 |
| 11 | Arquitectura de la solución. Fuente: Elaboración propia. . . . .                                                                                                                | 23 |
| 12 | Visualización de los componentes de una mano virtual: Articulaciones de la mano (Azul), <i>targets</i> (rojo) y <i>aim targets</i> (verde). Fuente: Elaboración propia. . . . . | 29 |
| 13 | Variación del RMSE según el número de fotogramas por cámara (L=Left, R=Right). Fuente: Elaboración propia. . . . .                                                              | 32 |
| 14 | Poses de una mano. Fuente: Elaboración propia. . . . .                                                                                                                          | 33 |
| 15 | Resultados de la triangulación de puntos por pose. Fuente: Elaboración propia. . . . .                                                                                          | 33 |
| 16 | Comparativa del número de FPS por método. Fuente: Elaboración propia. . . . .                                                                                                   | 34 |
| 17 | Variación de los FPS en el tiempo. Fuente: Elaboración propia. . . . .                                                                                                          | 35 |
| 18 | Distancia Euclidiana Promedio en captura de manos sin movimiento (izquierda) y con movimiento (derecha). Fuente: Elaboración propia. . . . .                                    | 36 |
| 19 | Distancia Euclidiana Promedio en captura de manos con movimiento mixto. Fuente: Elaboración propia. . . . .                                                                     | 36 |
| 20 | Distancia Euclidiana Promedio en captura de manos con movimiento de la muñeca. Fuente: Elaboración propia. . . . .                                                              | 37 |

## ÍNDICE DE TABLAS

|   |                                                                                        |    |
|---|----------------------------------------------------------------------------------------|----|
| 1 | Comparación del tiempo de CPU promedio por método. Fuente: Elaboración propia. . . . . | 37 |
|---|----------------------------------------------------------------------------------------|----|

## INTRODUCCIÓN

Uno de los problemas existentes en la educación chilena es el bajo interés de los estudiantes de educación básica y media por aprender química, ya sea, por los intereses personales del estudiante [Fernández, 2006], falta de infraestructura en los establecimientos educacionales para impartir clases prácticas de química [Brovelli Sepúlveda *et al.*, 2018], la metodología de aprendizaje, e incluso, el desinterés por enseñar química [Simonsen, 2021]. Este problema se hace más evidente cuando los estudiantes llegan a la educación media con una base débil en lo que respecta a los contenidos básicos de química, teniendo dificultades para aprender los conceptos más avanzados que se imparten en ese nivel de educación. Esto, acompañado de una falta de preparación de los estudiantes para trabajar en un laboratorio, genera riesgos asociados a la manipulación incorrecta del equipamiento y los compuestos químicos, que puede provocar intoxicación, quemaduras [T13, 2022], entre otros.

El aspecto pedagógico del problema se puede abarcar implementando alguna técnica de aprendizaje que permita a los estudiantes aumentar su motivación por el aprendizaje. Una técnica que puede ser explorada es la gamificación. Karl Kapp en su artículo sobre la gamificación y sus mitos define lo siguiente: la gamificación es un enfoque emergente para la instrucción. Facilita el aprendizaje y fomenta la motivación utilizando elementos de juego, mecánicas y pensamiento basado en juegos [Kapp, 2014].

Un caso real donde se puede evidenciar la efectividad de esta técnica es el trabajo realizado por un estudiante de la UTFSM<sup>1</sup>, que consiste en un videojuego diseñado para enseñar programación a estudiantes universitarios de primer año y mitigar los *misconceptions*<sup>2</sup> que pueden surgir durante el aprendizaje [Figueroa, 2024].

De esta forma surge «A darle Átomos» dentro del marco de la Feria de Software, evento anual organizado por la UTFSM. «A Darle Átomos» es un laboratorio de química virtual, que tiene como objetivo ser una herramienta de aprendizaje destinada a profesores y estudiantes desde séptimo básico hasta segundo medio, que puede ayudar a mitigar la problemática planteada enfocándose tanto en el aspecto pedagógico como en los riesgos asociados a los laboratorios de química. El laboratorio cuenta con los elementos básicos que se encuentran en cualquier laboratorio real (cristalería, instrumentos de medición, compuestos químicos, entre otros.) y cubre las unidades vistas mediante experiencias, que consisten en experimentos de química que el estudiante debe realizar siguiendo una serie de pasos con el fin de observar el fenómeno visto en la unidad.

Como «A darle Átomos» busca motivar el aprendizaje de la química, se debe ofrecer una experiencia interactiva e inmersiva, que permita explicar de manera práctica los contenidos del plan de estudio sin la necesidad de usar un laboratorio real. Para lograr esto, el software simula reacciones químicas, lo que implica recrear las interacciones entre los elementos

---

<sup>1</sup>Universidad Técnica Federico Santa María

<sup>2</sup>Conceptos erróneos

químicos a través de código, tarea que no es trivial considerando la complejidad de algunos de los fenómenos que se abarcan. También, se utiliza un sistema de seguimiento de manos mediante cámaras estereoscópicas, para que puedan interactuar con los objetos del entorno virtual y adquirir habilidades manuales que puedan trasladar a un laboratorio real con el fin de trabajar de forma segura.

Este proyecto puede generar un impacto muy positivo en la educación, y existe evidencia sobre su viabilidad en la investigación realizada por académicos de la Universidad de Concepción y la Universidad Andrés Bello, donde se demostró que el uso de laboratorios virtuales de química como herramienta de aprendizaje puede tener un impacto positivo en la motivación y aprendizaje de los estudiantes [Brovelli Sepúlveda *et al.*, 2018].

## CAPÍTULO 1

### DEFINICIÓN DEL PROBLEMA

Como el objetivo de la aplicación es entregar una experiencia interactiva e inmersiva, surge la problemática de que se necesita de un sistema de interacción humano-computadora que cumpla con estos requisitos, de forma que, el estudiante sea capaz de interactuar con los elementos del laboratorio tal como lo haría en un laboratorio real. Para lograr esto, se necesita un sistema que pueda capturar el movimiento de las manos de forma fluida y precisa. También surge otra problemática, la cual es, que el software será instalado en los computadores (personales o del establecimiento) de cada estudiante que curse la asignatura de Ciencias Naturales o Química, por lo tanto, el sistema a desarrollar no puede ser muy costoso de implementar. En la Figura 1 se puede apreciar un resumen del problema con sus componentes principales.

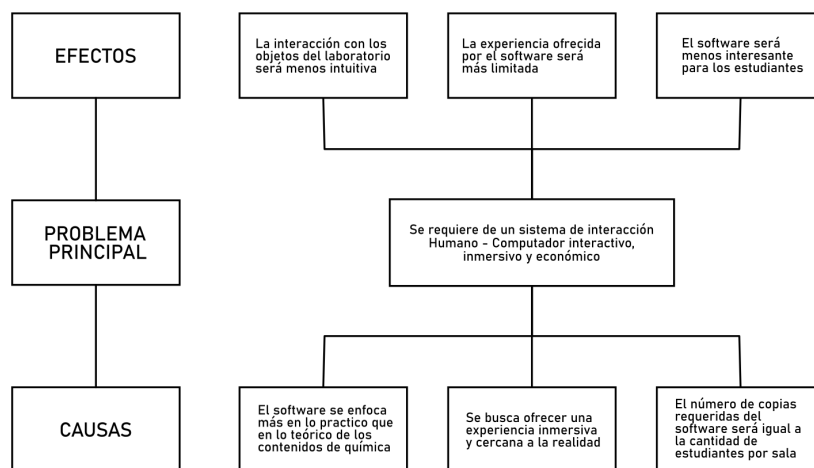


Figura 1: Árbol del problema

Una alternativa para solucionar esta problemática es un sistema de Realidad Virtual. La ventaja de esta tecnología es su popularidad, además, algunos dispositivos incluyen controles de manos con sensores que detectan con precisión la posición y rotación de las manos. Uno de los inconvenientes de esta alternativa es su costo y que algunas personas inevitablemente sienten náuseas al utilizar estos dispositivos por un cierto período de tiempo [Martirosov y Kopecek, 2017].

La otra alternativa es utilizar un sistema de seguimiento de manos, estos utilizan cámaras y sensores para capturar la posición de las manos. Existen muchas configuraciones, como el uso de una sola cámara [Chunduru *et al.*, 2021], dos cámaras [Wang *et al.*, 2011], guantes con marcadores [Wang y Popović, 2009], entre otros. Luego, utilizan distintos algoritmos para obtener los puntos de las articulaciones de las manos en el espacio 2D o 3D dependiendo de cómo se capturan los datos. Tiene como ventaja ser más económico que la alternativa

anterior, ya que, el componente principal es la cámara y existen cámaras web de grado de consumidor que pueden ofrecer buenos resultados y a un costo razonable, además, no ocasiona náuseas o mareos al usarlos. Su desventaja reside en la precisión del seguimiento de las manos, ya que, dependerá de varios factores, como el tipo de cámara a utilizar, resolución, velocidad de cuadros, sensores, entre otros.

Actualmente se tiene implementado un sistema de seguimiento de manos que utiliza cámaras estereoscópicas (previamente calibradas) para la captura de la posición de las manos mediante MediaPipe y DLT<sup>3</sup>. La arquitectura del sistema actual se puede observar en la Figura 2 y la configuración de las cámaras en la Figura 3. Tanto el proceso de captura de manos como el proceso de calibración de las cámaras fue implementado utilizando el código de Temuge B<sup>45</sup> como base.

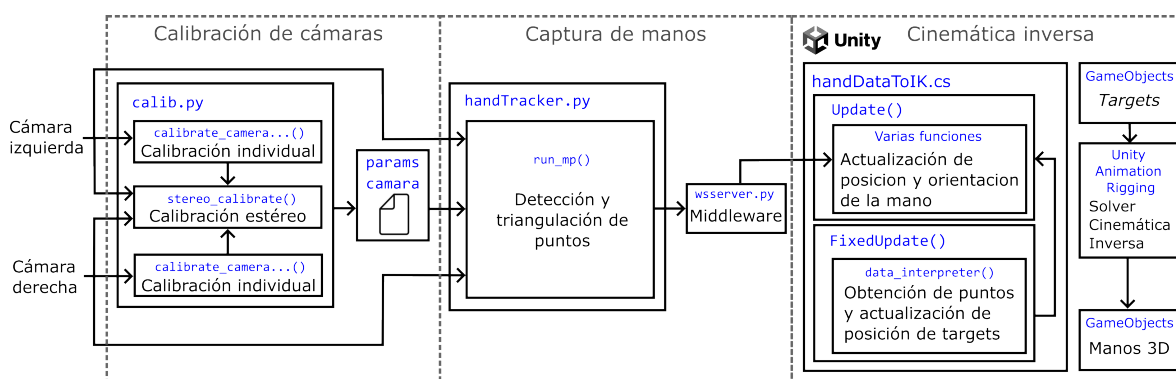


Figura 2: Arquitectura del sistema de seguimiento de manos actual. Fuente: Elaboración propia

Este sistema cumple con el requisito de ser una alternativa económica, ya que, se puede utilizar dos cámaras web de gama media. Pero tiene problemas que afectan negativamente la experiencia del usuario, rompiendo la inmersión. Entre los problemas encontrados se encuentran:

- Poca fluidez en el movimiento capturado de las manos.
- Fuerte ruido en los datos capturados, ocasionando temblores perceptibles en las manos virtuales.
- Distorsión de la captura, provocando que a veces las manos capturadas se vean desproporcionadas.

<sup>3</sup>Direct Linear Transform

<sup>4</sup><https://github.com/TemugeB/handpose3d>

<sup>5</sup>[https://github.com/TemugeB/python\\_stereo\\_camera\\_calibrate](https://github.com/TemugeB/python_stereo_camera_calibrate)

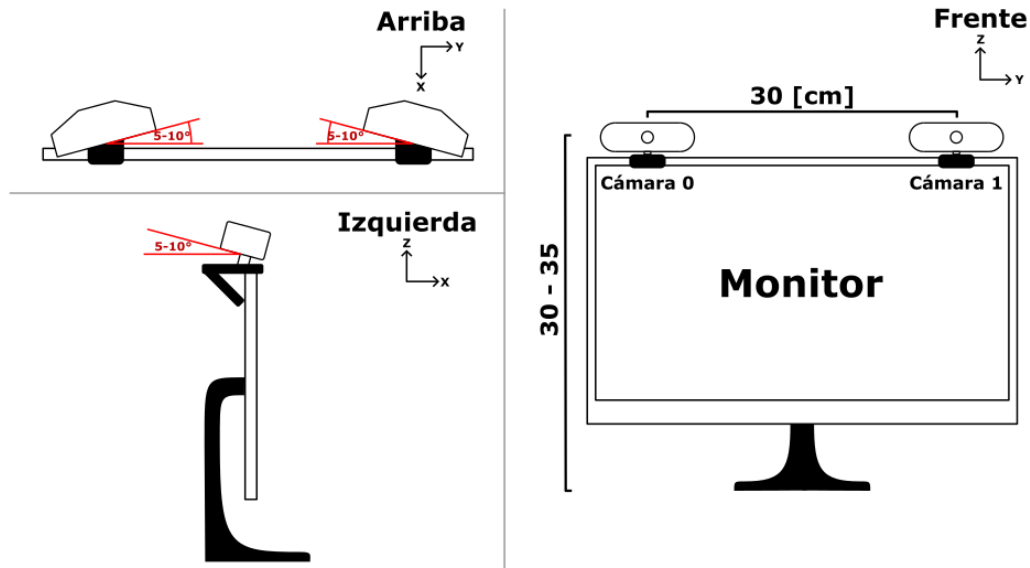


Figura 3: Configuración del sistema estereoscópico. Fuente: Elaboración propia

La solución a este problema requiere de una mejora en la calidad de los datos generados por las cámaras, lo que implica el uso de un método rápido para reducir el ruido de la captura del movimiento de las manos, y diseñar un sistema que utilice los recursos del ordenador de manera óptima para obtener el mayor rendimiento posible. Para lograr esto, se propone un sistema formado por dos procesos encargados de la obtención de datos 2D (uno por cámara) y un tercer proceso encargado de calcular las coordenadas en el espacio 3D de las manos. Para mejorar el rendimiento del sistema, los procesos trabajarán en paralelo y para reducir el ruido de los datos, se aplicará un método de interpolación. En las siguientes secciones se verá cómo se afrontará este desafío, explicando las tecnologías a utilizar, arquitectura del sistema propuesto, entre otros.

## 1.1. Objetivos

### 1.1.1. Objetivo general

Aumentar la precisión y rendimiento del sistema de seguimiento de manos anterior mediante la implementación de algoritmos y técnicas en los procesos clave que conforman el sistema.

### **1.1.2. Objetivos específicos**

- Desarrollar un sistema de calibración de cámaras estereoscópicas utilizando RDLT para mejorar la calidad de la calibración.
- Calcular las proyecciones 2D de las manos en las imágenes capturadas por las cámaras utilizando MediaPipe y un método de interpolación, de forma paralela.
- Mejorar la fluidez y precisión del movimiento de las manos virtuales generado por el solver de cinemática inversa.

## CAPÍTULO 2

### MARCO CONCEPTUAL

#### 2.1. MediaPipe

MediaPipe es un *framework*<sup>6</sup> de código abierto que permite crear soluciones de aprendizaje automático en donde se requiera realizar inferencias a partir de datos sensoriales a través de la creación de un *perception pipeline*<sup>7</sup>, el cual se define como un grafo dirigido que contiene distintos componentes modulares, los cuales pueden ser modelos de inferencia, algoritmos de procesamiento, entre otros [Lugaresi et al., 2019].

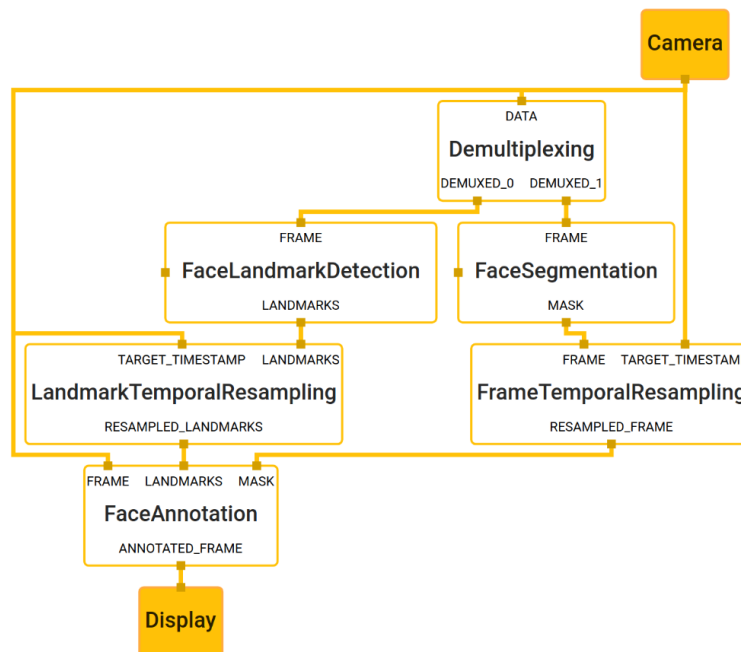


Figura 4: Ejemplo de un *perception pipeline* para la detección de puntos de referencia en un rostro. Fuente: [Lugaresi et al., 2019]

Los *perception pipelines* tienen una entrada y una salida. En la entrada se ingresan los datos sensoriales, como una señal de audio, una imagen, un *stream*<sup>8</sup> de video en tiempo real, entre otros. La salida es el resultado final, cuyo valor dependerá de los componentes utilizados. En el caso de que se requiera extraer más información de los datos, se puede obtener más de un resultado. Por ejemplo, si la tarea consiste en detectar objetos, se puede construir un

---

<sup>6</sup>Marco de trabajo

<sup>7</sup>Serie de procesos utilizados para la detección de objetos

<sup>8</sup>Flujo de datos

*perception pipeline* que además de entregar las coordenadas del cuadro de límite, pueda entregar la categoría del objeto detectado.

Google ofrece soluciones creadas con MediaPipe para realizar distintas tareas, entre ellas, la detección de puntos de referencia en la mano [Google, 2025]. Como su nombre indica, esta solución es capaz de detectar ciertos puntos de referencia de la mano como se puede observar en la Figura 5:

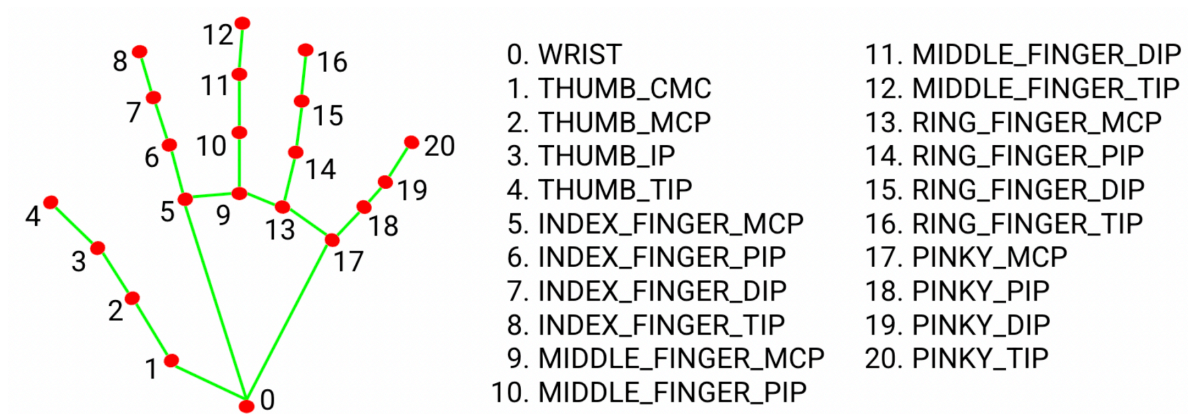


Figura 5: Puntos de referencia de una mano. Fuente: [Google, 2025]

La solución recibe como entrada una imagen, un video o una transmisión en vivo de video. Luego, el modelo de inferencia se encargará de detectar los puntos de referencia y entregar sus coordenadas (tanto globales como de imagen).

## 2.2. Interpolación

La interpolación es un método analítico utilizado para estimar valores intermedios entre dos o más puntos mediante una función. Existen distintos tipos, como la interpolación lineal, cuadrática, polinomial, entre otros.

La interpolación lineal es el método más sencillo, ya que, consiste en conectar dos puntos por una recta [Chapra y Canale, 2010].

$$f(x) = f(x_0) + (x - x_0) \frac{f(x_1) - f(x_0)}{x_1 - x_0} \quad (1)$$

## 2.3. Perspective-n-Point

Uno de los problemas más relevantes dentro de la disciplina de *computer vision*<sup>9</sup> es el problema de *Perspective-n-Point* (PnP) debido al gran número de aplicaciones que tiene. Su origen proviene de un problema conocido como LDP<sup>10</sup>, el cual consiste en determinar la pose de una cámara a partir de una fotografía tomada por la misma, que contiene un conjunto de marcadores cuyas coordenadas en el espacio son conocidas. Martin A. Fischler et al. demostró que si se calculan las longitudes de los trazados que van desde el centro de perspectiva de la cámara hacia los marcadores, se puede determinar la pose de la cámara. Este problema lo llamó el problema de Perspective-n-Point [Fischler y Bolles, 1981].

Su definición matemática es: dadas las ubicaciones espaciales relativas de  $n$  puntos de control, y dado el ángulo de cada par de puntos de control desde un punto adicional llamado el Centro de Perspectiva (CP), encontrar las longitudes de los segmentos de línea que unen el CP a cada uno de los puntos de control [Fischler y Bolles, 1981].

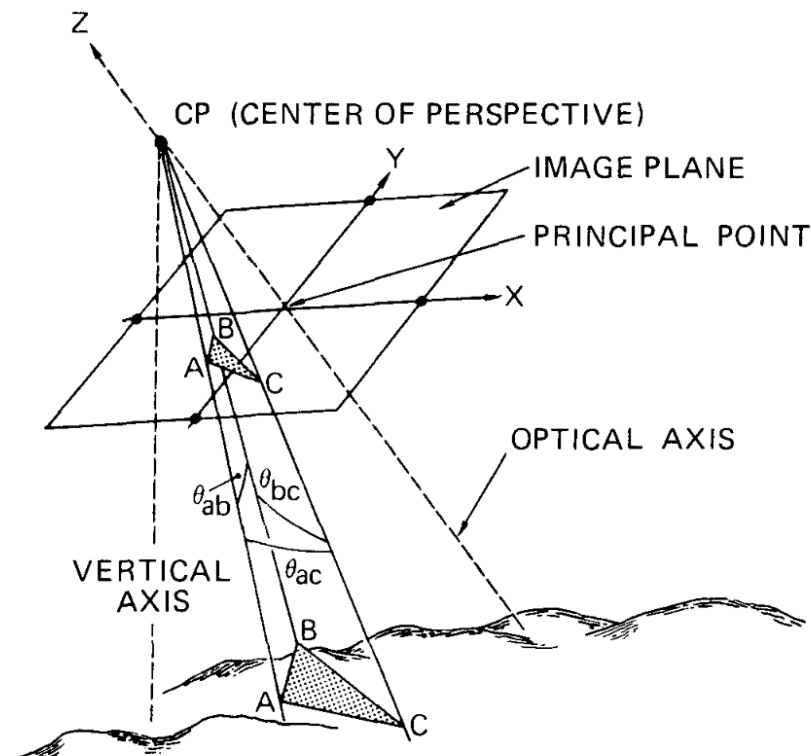


Figura 6: Modelo geométrico del problema. Fuente: [Fischler y Bolles, 1981]

Actualmente existen varios métodos que son capaces de resolver este problema. Por un lado, están los métodos diseñados para resolver las versiones del problema que utilizan una

<sup>9</sup>Visión por computadora

<sup>10</sup>Location Determination Problem

cantidad específica de puntos como P3P ( $n = 3$ ) [Ke y Roumeliotis, 2017], y otros pueden encontrar una solución para  $n \geq 3$  puntos como el método que veremos a continuación.

## 2.4. Direct Linear Transform

DLT es uno de los métodos utilizados para resolver el problema de PnP. Se utilizan los datos intrínsecos de las cámaras y su orientación (posición y rotación) para definir la transformación lineal entre el espacio 3D del objeto y el espacio 2D de las imágenes capturadas.

Sean  $x, y, z \in \mathbb{R}$  las coordenadas 3D de un objeto en el espacio y  $u_i, v_i \in \mathbb{R}$  las coordenadas en el espacio de la imagen tomada por la  $i$ -ésima cámara. Entonces, según el concepto básico de la fotogrametría, existe la siguiente relación:

$$\begin{bmatrix} u_i - u_p \\ v_i - v_p \\ -c \end{bmatrix} = \lambda M \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \quad (2)$$

donde  $c$  es la distancia principal de la cámara,  $\lambda$  es un factor de escala,  $u_p, v_p \in \mathbb{R}$  son las coordenadas de imagen del punto principal (Centro de la imagen),  $M \in \mathbb{R}^{3 \times 3}$  es una matriz de rotación, y  $x_0, y_0, z_0 \in \mathbb{R}$  son las coordenadas 3D del centro de perspectiva de la cámara. Luego, para corregir distintos errores como la distorsión del lente de la cámara, se introduce el plano comparador.

Asumiendo que las coordenadas de la imagen y del plano son paralelas, entonces se realiza la transformación desde las coordenadas del plano comparador a las de la imagen de la siguiente forma:

$$u_i - u_p = \lambda_u (q + \Delta q - q_0) \quad (3)$$

$$v_i - v_p = \lambda_v (r + \Delta r - r_0) \quad (4)$$

donde  $\lambda_u$  y  $\lambda_v$  son los factores de escala de los ejes,  $q$  y  $r$  son las coordenadas del plano comparador,  $q_0$  y  $r_0$  son las coordenadas del punto principal en el plano comparador, y  $\Delta q$  y  $\Delta r$  son los errores sistemáticos. Reemplazando (3) y (4) en (2) se obtiene:

$$q + \Delta q - q_0 = -c_q \frac{m_{11}(x - x_0) + m_{12}(y - y_0) + m_{13}(z - z_0)}{m_{31}(x - x_0) + m_{32}(y - y_0) + m_{33}(z - z_0)} \quad (5)$$

$$r + \Delta r - r_0 = -c_r \frac{m_{21}(x - x_0) + m_{22}(y - y_0) + m_{23}(z - z_0)}{m_{31}(x - x_0) + m_{32}(y - y_0) + m_{33}(z - z_0)} \quad (6)$$

donde  $c_q = \frac{c}{\lambda_u}$ ,  $c_r = \frac{c}{\lambda_v}$  y  $m_{ij}$  es un componente de la matriz  $M$ . Las ecuaciones (5) y (6) también se pueden expresar como:

$$q + \Delta q = \frac{l_1 x + l_2 y + l_3 z + l_4}{l_9 x + l_{10} y + l_{11} z + 1} \quad (7)$$



El objetivo del método es resolver PnP, por lo tanto, se debe encontrar las matrices de rotación  $R$  y traslación  $\mathbf{t}$  de la cámara.

$$R = \begin{bmatrix} r_1 & r_4 & r_7 \\ r_2 & r_5 & r_8 \\ r_3 & r_6 & r_9 \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Luego, se define  $\mathbf{w}_i = [x_i, y_i, z_i]^\top$  como un punto en el espacio,  $\mathbf{p}_i = [u_i, v_i, 1]^\top$  como la proyección de  $\mathbf{w}_i$  en el plano de la imagen,  $\mathbf{o}_c \in \mathbb{R}^3$  como el origen de la cámara y  $\mathbf{o}_w \in \mathbb{R}^3$  como el origen global. La correspondencia entre  $\mathbf{w}_i$  y  $\mathbf{p}_i$  se expresa como:

$$\lambda_i \mathbf{p}_i = R \mathbf{w}_i + \mathbf{t} \quad (9)$$

donde  $\lambda_i$  representa el factor de profundidad del punto  $i$ . En la Figura 7 se puede observar que por cada par de puntos  $\mathbf{w}_i$  y  $\mathbf{w}_j$  se tienen dos planos, uno formado por  $\mathbf{o}_c \mathbf{w}_i \mathbf{w}_j$  y otro formado por  $\mathbf{o}_c \mathbf{p}_i \mathbf{p}_j$  con  $\mathbf{N}_{ij} \in \mathbb{R}^3$  y  $\mathbf{n}_{ij} \in \mathbb{R}^3$  como sus normales respectivamente. Se puede observar que el plano  $\mathbf{o}_c \mathbf{w}_i \mathbf{w}_j$  se puede obtener escalando  $\mathbf{o}_c \mathbf{p}_i \mathbf{p}_j$  por un factor  $\alpha_{ij}$ , por lo que, se tiene la siguiente relación:

$$\alpha_{ij} \mathbf{n}_{ij} = \mathbf{N}_{ij} \quad (10)$$

La normal  $\mathbf{N}_{ij}$  se puede calcular mediante el producto cruz de los puntos  $\mathbf{w}_i$  y  $\mathbf{w}_j$  respecto al origen  $\mathbf{o}_c$ :

$$\mathbf{N}_{ij} = (R \mathbf{w}_i + \mathbf{t}) \times (R \mathbf{w}_j + \mathbf{t}) \quad (11)$$

que también se puede escribir como:

$$\mathbf{N}_{ij} = R M_{ij} - H V_{ij} \quad (12)$$

donde las matrices  $M_{ij}$ ,  $V_{ij}$  y  $H$  se definen como:

$$\begin{aligned} M_{ij} &= \mathbf{w}_i \times \mathbf{w}_j \\ V_{ij} &= \mathbf{w}_i - \mathbf{w}_j \\ H &= \begin{bmatrix} r_3 t_y - r_2 t_z & r_6 t_y - r_5 t_z & r_9 t_y - r_8 t_z \\ r_1 t_z - r_3 t_x & r_4 t_z - r_6 t_x & r_7 t_z - r_9 t_x \\ r_2 t_x - r_1 t_y & r_5 t_x - r_4 t_y & r_8 t_x - r_7 t_y \end{bmatrix} = \begin{bmatrix} h_1 & h_4 & h_7 \\ h_2 & h_5 & h_8 \\ h_3 & h_6 & h_9 \end{bmatrix} \end{aligned}$$

Finalmente, reemplazando (10) en (12) se obtiene:

$$\alpha_k \mathbf{n}_k = RM_k - HV_k, \quad k \in \left\{1, 2, \dots, \frac{n(n-1)}{2}\right\} \quad (13)$$

donde  $k$  es el  $k$ -ésimo par de puntos del conjunto de  $n$  puntos, tal que,  $\{\mathbf{w}_i, \mathbf{w}_j \mid i \neq j \text{ para } i, j \in \{1, 2, \dots, n\}\}$  y  $H$  es la nueva incógnita de la ecuación.

Para construir el sistema de ecuaciones necesario para resolver el problema, se utilizarán las ecuaciones (9) y (13). Eliminando el factor  $\lambda_i$ , la ecuación (9) se puede expresar como:

$$\begin{aligned} r_1 x_i - r_3 x_i u_i + r_4 y_i - r_6 y_i u_i + r_7 z_i - r_9 z_i u_i + t_x - t_z u_i &= 0 \\ r_2 x_i - r_3 x_i v_i + r_5 y_i - r_6 y_i v_i + r_8 z_i - r_9 z_i v_i + t_y - t_z v_i &= 0 \end{aligned} \quad (14)$$

que se puede escribir como una ecuación matricial de la forma  $AX = B$ , obteniendo:

$$A_i \mathbf{s}_1 = 0_{2 \times 1} \quad i \in \{1, 2, \dots, n\} \quad (15)$$

donde:

$$\begin{aligned} A_i &= \begin{bmatrix} x_i & 0 & -x_i u_i & y_i & 0 & -y_i u_i & z_i & 0 & -z_i u_i & 1 & 0 & -u_i \\ 0 & x_i & -x_i v_i & 0 & y_i & -y_i v_i & 0 & z_i & -z_i v_i & 0 & 1 & -v_i \end{bmatrix} \\ \mathbf{s}_1 &= [r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad r_6 \quad r_7 \quad r_8 \quad r_9 \quad t_x \quad t_y \quad t_z]^\top \end{aligned}$$

Análogamente, la ecuación (13) se puede expresar como:

$$\begin{aligned} r_1 M_x n_{z,k} - r_3 M_x n_{x,k} + r_4 M_y n_{z,k} - r_6 M_y n_{x,k} + r_7 M_z n_{z,k} - r_9 M_z n_{x,k} \\ - h_1 V_x n_{z,k} + h_3 V_x n_{x,k} - h_4 V_y n_{z,k} + h_6 V_y n_{x,k} - h_7 V_z n_{z,k} + h_9 V_z n_{x,k} &= 0 \\ r_2 M_x n_{z,k} - r_3 M_x n_{y,k} + r_5 M_y n_{z,k} - r_6 M_y n_{y,k} + r_8 M_z n_{z,k} - r_9 M_z n_{y,k} \\ - h_2 V_x n_{z,k} + h_3 V_x n_{y,k} - h_5 V_y n_{z,k} + h_6 V_y n_{y,k} - h_8 V_z n_{z,k} + h_9 V_z n_{y,k} &= 0 \end{aligned} \quad (16)$$

que también se puede escribir como una ecuación matricial:

$$B_k \mathbf{s}_2 = 0_{2 \times 1} \quad k \in \left\{1, 2, \dots, \frac{n(n-1)}{2}\right\} \quad (17)$$

donde:

$$B_k = \begin{bmatrix} B_k^{(1)} & B_k^{(3)} \\ B_k^{(2)} & B_k^{(4)} \end{bmatrix}$$

con

$$B_k^{(1)} = \begin{bmatrix} M_x n_{z,k} & 0 & -M_x n_{x,k} & M_y n_{z,k} & 0 & -M_y n_{x,k} & M_z n_{z,k} & 0 & -M_z n_{x,k} \end{bmatrix}$$

$$B_k^{(2)} = \begin{bmatrix} 0 & M_x n_{z,k} & -M_x n_{y,k} & 0 & M_y n_{z,k} & -M_y n_{y,k} & 0 & M_z n_{z,k} & -M_z n_{y,k} \end{bmatrix}$$

$$B_k^{(3)} = \begin{bmatrix} -V_x n_{z,k} & 0 & V_x n_{x,k} & -V_y n_{z,k} & 0 & V_y n_{x,k} & -V_z n_{z,k} & 0 & V_z n_{x,k} \end{bmatrix}$$

$$B_k^{(4)} = \begin{bmatrix} 0 & -V_x n_{z,k} & V_x n_{y,k} & 0 & -V_y n_{z,k} & V_y n_{y,k} & 0 & -V_z n_{z,k} & V_z n_{y,k} \end{bmatrix}$$

$$\mathbf{s}_2 = \begin{bmatrix} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & h_1 & h_2 & h_3 & h_4 & h_5 & h_6 & h_7 & h_8 & h_9 \end{bmatrix}^\top$$

Finalmente se crean las matrices  $A$  y  $B$  apilando las matrices  $A_i$  y  $B_k$  respectivamente para combinarlas en una sola ecuación matricial:

$$\begin{bmatrix} A[:, 1 : 9] & 0 & A[:, 10 : 12] \\ B & 0_{n(n-1) \times 3} & \end{bmatrix} \mathbf{s} = 0_{n(n+1) \times 1} \quad (18)$$

$$C\mathbf{s} = 0_{n(n+1) \times 1}$$

donde  $\mathbf{s}$  es la fusión de las matrices  $\mathbf{s}_1$  y  $\mathbf{s}_2$ . La ecuación (18) se resuelve mediante SVD obteniendo la solución del problema, es decir, las matrices  $R$  y  $\mathbf{t}$ .

Para obtener un mejor resultado, se refina la matriz de rotación mediante el uso del algoritmo de Gauss-Newton. Para ello,  $R$  es representado de otra forma, utilizando los parámetros de cuaternión:

$$R = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (19)$$

Luego se multiplica  $1/t_z$  en ambos lados de la ecuación 9 para obtener

$$\hat{\lambda}_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = M\mathbf{w}_i + \hat{\mathbf{t}} \quad (20)$$

donde  $\hat{\lambda}_i = \lambda_i/t_z$ ,  $M = R/t_z$  y  $\hat{\mathbf{t}} = [t_1, t_2, 1] = \mathbf{t}/t_z$ . Luego se define  $a = q_0/\sqrt{t_z}$ ,  $b = q_1/\sqrt{t_z}$ ,  $c = q_2/\sqrt{t_z}$  y  $d = q_3/\sqrt{t_z}$ . De esta forma, la matriz  $M$  se puede escribir como:

$$M = \begin{bmatrix} \mathbf{m}_1^\top \\ \mathbf{m}_2^\top \\ \mathbf{m}_3^\top \end{bmatrix} = \begin{bmatrix} a^2 + b^2 - c^2 - d^2 & 2bc - 2ad & 2bd + 2ac \\ 2bc + 2ad & a^2 - b^2 + c^2 - d^2 & 2cd - 2ab \\ 2bd - 2ac & 2cd + 2ab & a^2 - b^2 - c^2 + d^2 \end{bmatrix} \quad (21)$$

Como la ecuación 20 se cumple para cada punto  $\mathbf{w}_i$ , se puede expresar como un sistema de ecuaciones que incluya a todos los puntos:

$$\begin{aligned} -\mathbf{m}_1^\top \bar{\mathbf{w}}_i + \mathbf{m}_3^\top \bar{\mathbf{u}}_i + \bar{u}_i &= 0 \\ -\mathbf{m}_2^\top \bar{\mathbf{w}}_i + \mathbf{m}_3^\top \bar{\mathbf{v}}_i + \bar{v}_i &= 0 \end{aligned} \quad (22)$$

donde,

$$\begin{aligned} \bar{\mathbf{w}}_i &= [\bar{w}_x, \bar{w}_y, \bar{w}_z]^\top = \mathbf{w}_i - \frac{1}{n} \sum_{i=1}^n \mathbf{w}_i \\ \bar{\mathbf{u}}_i &= [\bar{u}_x, \bar{u}_y, \bar{u}_z]^\top = \mathbf{w}_i u_i - \frac{1}{n} \sum_{i=1}^n \mathbf{w}_i u_i \\ \bar{\mathbf{v}}_i &= [\bar{v}_x, \bar{v}_y, \bar{v}_z]^\top = \mathbf{w}_i v_i - \frac{1}{n} \sum_{i=1}^n \mathbf{w}_i v_i \\ \bar{u}_i &= u_i - \frac{1}{n} \sum_{i=1}^n u_i, \quad \bar{v}_i = v_i - \frac{1}{n} \sum_{i=1}^n v_i \end{aligned} \quad (23)$$

Al igual que los sistemas de ecuaciones (14) y (16), el sistema de ecuaciones (22) se puede expresar como:

$$E_i \bar{\mathbf{s}} = 0_{2 \times 1}, \quad i \in \{1, 2, \dots, n\} \quad (24)$$

donde,

$$E_i = \begin{bmatrix} E_i^{(1)} & E_i^{(3)} \\ E_i^{(2)} & E_i^{(4)} \end{bmatrix}$$

con

$$\begin{aligned} E_i^{(1)} &= [\bar{u}_i \quad -\bar{w}_x + \bar{u}_z \quad 2\bar{u}_y \quad -2\bar{w}_z - 2\bar{u}_x \quad 2\bar{w}_y] \\ E_i^{(2)} &= [\bar{v}_i \quad -\bar{w}_y + \bar{v}_z \quad 2\bar{w}_z + 2\bar{v}_y \quad -2\bar{v}_x \quad -2\bar{w}_x] \\ E_i^{(3)} &= [-\bar{w}_x - \bar{u}_z \quad -2\bar{w}_y \quad -2\bar{w}_z + 2\bar{u}_x \quad \bar{w}_x - \bar{u}_z \quad 2\bar{u}_y \quad \bar{w}_x + \bar{u}_z] \\ E_i^{(4)} &= [\bar{w}_y - \bar{v}_z \quad -2\bar{w}_x \quad 2\bar{v}_x \quad \bar{w}_y - \bar{v}_z \quad -2\bar{w}_z + 2\bar{v}_y \quad \bar{w}_y + \bar{v}_z] \end{aligned}$$

$$\bar{\mathbf{s}} = [1 \quad a^2 \quad ab \quad ac \quad ad \quad b^2 \quad bc \quad bd \quad c^2 \quad cd \quad d^2]^\top \quad (25)$$

Finalmente, se apilan todos los  $E_i$ :

$$\begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \bar{\mathbf{s}} = 0_{2 \times 1} \Rightarrow E \bar{\mathbf{s}} = 0_{2 \times 1} \quad (26)$$

Así, se obtiene el siguiente problema de mínimos cuadrados:

$$\min_{a,b,c,d} f(a,b,c,d) = \|E\bar{\mathbf{s}}\|^2 = \bar{\mathbf{s}}^T E^T E \bar{\mathbf{s}} \quad (27)$$

donde  $E^T E$  es una matriz simétrica de  $11 \times 11$ . Ahora, se puede definir la estimación inicial del algoritmo de Gauss-Newton como  $\bar{\mathbf{s}}_0 = [a_0, b_0, c_0, d_0]^T$ , donde, los valores de  $a_0, b_0, c_0$  y  $d_0$  se calculan a partir de la matriz  $R$  obtenida al resolver la ecuación (18), transformada en cuaternión [Shepperd, 1978].

De esta forma, la solución será estimada mediante la siguiente expresión:

$$\bar{\mathbf{s}}_{k+1} = \bar{\mathbf{s}}_k - (J^T J + \lambda I_{4 \times 4})^{-1} J^T E \bar{\mathbf{s}} \quad (28)$$

donde  $\lambda$  es el factor de amortiguado,  $J$  es la matriz jacobiana de  $E$  que se puede calcular como  $J = EF$ , donde,

$$F = \begin{bmatrix} 0 & 2a & b & c & d & 0 & d & 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 & 0 & 2b & c & d & 0 & 0 & 0 \\ 0 & 0 & 0 & a & 0 & 0 & b & 0 & 2c & d & 0 \\ 0 & 0 & 0 & 0 & a & 0 & 0 & b & 0 & c & 2d \end{bmatrix} \quad (29)$$

Una vez se obtienen los parámetros refinados  $a, b, c, d$ , se obtiene  $M$  y luego  $R$  recordando que  $M = R/t_z$ .

## 2.5. Calibración de cámaras

El problema de calibración de cámaras consiste en determinar los parámetros de la cámara a partir de un objeto en el que se pueda identificar puntos de referencia (por ejemplo, un tablero de calibración). El algoritmo de Tsai [Tsai, 2003] es uno de los métodos más utilizados para resolver este problema, gracias a su versatilidad y precisión en comparación con otros métodos.

Tsai identifica los parámetros necesarios para hacer la conversión de coordenadas globales a coordenadas en el plano de imagen y los clasifica en dos categorías:

- **Parámetros extrínsecos:** son los parámetros  $R$  y  $t$ , donde  $R$  es la matriz de rotación y  $t$  es el vector de traslación.
- **Parámetros intrínsecos:** son los parámetros  $f$ ,  $k_i$ ,  $c_x$  y  $c_y$ , donde  $f$  es la distancia focal,  $k_i$  corresponde a los coeficientes de distorsión radial del lente y  $(c_x, c_y)$  define las coordenadas del origen del plano de imagen.

A partir de esta clasificación, se define el algoritmo de Tsai. Antes de explicar el algoritmo, se debe aclarar que, existen dos escenarios planteados por el autor para realizar la calibración de la cámara, uno en el que se utiliza un conjunto de puntos de calibración coplanares y otro en el que no. En este trabajo, solo se considerará el primer escenario.

El algoritmo de Tsai se divide en dos etapas:

- **Computar orientación y posición 3D de la cámara:** en esta etapa se calcula  $R$  y  $t$  a partir de una imagen del objeto utilizado para la calibración.
- **Computar distancia focal efectiva y coeficientes de distorsión:** en esta etapa se calculan los parámetros  $f$  y  $k_i$  utilizando las coordenadas de los puntos de calibración extraídos de la etapa anterior. Se asume que el origen del plano de imagen  $(c_x, c_y)$  se encuentra en el centro de la imagen.

## 2.6. Bundle adjustment

*Bundle adjustment* es un problema de estimación de parámetros utilizado para refinar los resultados obtenidos del proceso de reconstrucción 3D de un objeto o la estimación de la pose de una cámara. Este problema busca obtener la solución más optima tanto para la estructura 3D como para los parámetros de la cámara. Esto se logra al minimizar una función que cuantifique el error del modelo (por ejemplo, el error de re-proyección) [Triggs *et al.*, 1999].

Comúnmente se utiliza el método de mínimos cuadrados para minimizar la función, siendo el algoritmo de Levenberg-Marquardt [Ranganathan, 2004] el más utilizado debido a su capacidad para converger rápidamente en un resultado óptimo.

## 2.7. Filtro Kalman

El filtro Kalman es un conjunto de ecuaciones matemáticas que permite realizar estimaciones de estados pasados, presentes y futuros sin la necesidad de conocer la naturaleza del modelo del sistema [Welch y Bishop, 1995].

Este filtro funciona mediante un ciclo constante de predicción y corrección que recibe las estimaciones iniciales del estado  $\mathbf{x} \in \mathbb{R}^n$  y covarianza del error de la estimación  $P$  en un

instante de tiempo discreto  $k \in \mathbb{N}$ . En la etapa de predicción se realiza una proyección en el futuro del estado  $x_k^-$  y covarianza  $P_k^-$  mediante las siguientes fórmulas:

$$\hat{x}_{k+1}^- = A_k \hat{x}_k + B u_k \quad (30)$$

$$P_{k+1}^- = A_k P_k A_k^\top + Q_k \quad (31)$$

donde la matriz  $A \in \mathbb{R}^{n \times n}$  relaciona el estado  $\mathbf{x}$  en el tiempo  $k$  con el estado  $\mathbf{x}$  en el tiempo  $k + 1$ , la matriz  $B \in \mathbb{R}^{n \times l}$  relaciona la variable de control  $\mathbf{u} \in \mathbb{R}^l$  con el estado  $\mathbf{x}$  y  $Q_k$  es el ruido del proceso o perturbación que afecta al estado.

En la etapa de corrección se calcula la ganancia de Kalman  $K \in \mathbb{R}^{n \times m}$  para mejorar la estimación del estado  $\hat{x}_k$  y minimizar la covarianza del error de la estimación  $P_k$ . Para ello, se utilizan las siguientes fórmulas:

$$K_k = P_k^- H_k^\top (H_k P_k^- H_k^\top + R_k)^{-1} \quad (32)$$

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H_k \hat{x}_k^-) \quad (33)$$

$$P_k = (I - K_k H_k) P_k^- \quad (34)$$

donde la matriz  $H \in \mathbb{R}^{m \times n}$  relaciona el estado  $\mathbf{x}$  con la medición  $\mathbf{z} \in \mathbb{R}^m$  y  $R_k$  es la covarianza del error de la medición.

Al finalizar las etapas de predicción y corrección, se completa un ciclo. Cada ciclo se repite de forma que los valores estimados *a posteriori* en el ciclo anterior se utilizan para predecir los nuevos valores *a priori* (denotado por el símbolo – en el superíndice) en el ciclo actual.

## 2.8. Cinemática

La cinemática es el estudio de la geometría del movimiento. Si el movimiento de un cuerpo está restringido a moverse de cierta forma como consecuencia del contacto con otros elementos, su movimiento puede ser determinado por la cinemática [Beggs, 1983]. En el caso de un *rigid body*<sup>11</sup>, esta limitación en el movimiento puede ser ocasionado por una articulación conectada al mismo. Una articulación se define como la conexión entre dos cuerpos que limita su movimiento relativo [Waldron y Schmiedeler, 2016], el nivel de restricción dependerá de los GDL<sup>12</sup> de la articulación y puede ser desde limitar el movimiento a la rotación o traslación en un solo eje (1 GDL), a no tener limitaciones, es decir, la rotación y traslación en los tres ejes (6 GDL). Si se tiene un sistema compuesto de varias articulaciones, entonces su GDL será la suma de los GDL de todas sus articulaciones.

Cuando se tiene un sistema de cuerpos rígidos conectados por articulaciones (a excepción del primero y último), se habla de una cadena serial (conocido también como cadena cinemática) [Waldron y Schmiedeler, 2016]. Un ejemplo de un sistema que contiene una cadena

<sup>11</sup>Cuerpo rígido

<sup>12</sup>Grados de libertad

cinemática es un brazo robótico (ver Figura 8), el cual inicia en la base del brazo y termina en el efector final (*end-effector*), que es el elemento del brazo que interactúa con el entorno.

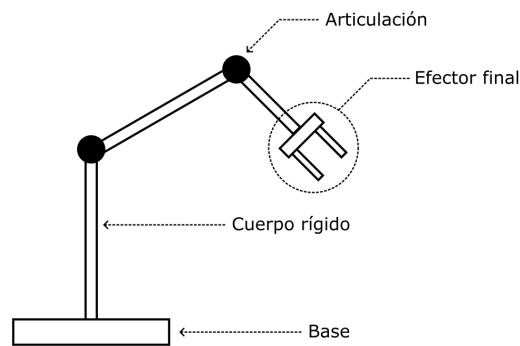


Figura 8: Ejemplo de una cadena cinemática. Fuente: Elaboración propia

Existen diversas convenciones para representar el modelo geométrico de una cadena de articulaciones. En este trabajo se utilizará la convención propuesta por Khalil y Dombre [Khalil y Dombre, 2004], el cual es una variación de la propuesta de Denavit y Hartenberg [Denavit y Hartenberg, 1955]. Sabemos que una cadena cinemática está formada por  $n + 1$  cuerpos rígidos que llamaremos eslabones, conectados por  $n$  articulaciones. Los eslabones son enumerados de forma que el eslabón conectado a la base será el eslabón 0 y el eslabón terminal será el eslabón  $n$ . La articulación  $j$  conecta el eslabón  $j$  con el eslabón  $j - 1$  y se conocerá como  $q_j$ . Cada eslabón tendrá un pivote  $R_j$  que contiene un origen  $\mathbf{o}_j$  en el espacio y los ejes  $x_j, y_j$  y  $z_j$ . El eje  $z_j$  esté a lo largo del eje de rotación de la articulación  $j$ , el eje  $x_j$  se encuentre alineado con la normal común entre  $z_j$  y  $z_{j+1}$  y su intersección con  $z_j$  define el origen  $\mathbf{o}_j$  del pivote, finalmente el eje  $y_j$  se obtiene de la regla de la mano derecha [Khalil y Dombre, 2004].

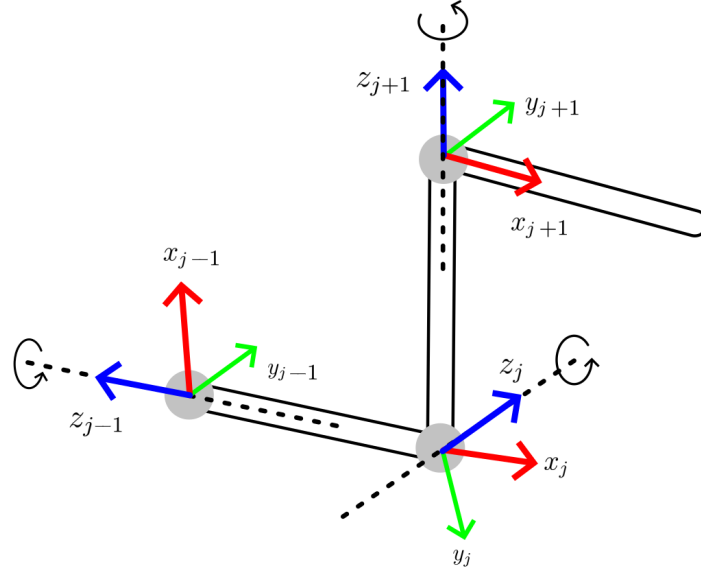


Figura 9: Visualización de los pivotes en una cadena cinemática. Fuente: Elaboración propia

Luego, la matriz de transformación  ${}^{j-1}T_j$  del pivote  $R_{j-1}$  al pivote  $R_j$  estará en función de cuatro parámetros [Khalil y Dombre, 2004]:

- Giro de la cadena ( $\alpha_j$ ): será el ángulo entre  $z_{j-1}$  y  $z_j$ .
- Largo de la cadena ( $d_j$ ): será la distancia entre  $z_{j-1}$  y  $z_j$ .
- Ángulo de la articulación ( $\theta_j$ ): será el ángulo entre  $x_{j-1}$  y  $x_j$ .
- Offset de la articulación ( $r_j$ ): será la distancia entre  $x_{j-1}$  y  $x_j$

De esta forma, la matriz  ${}^{j-1}T_j$  quedará de la siguiente forma:

$$\begin{aligned} {}^{j-1}T_j &= Rot(x, \alpha_j) Trans(x, d_j) Rot(z, \theta_j) Trans(z, r_j) \\ &= \begin{bmatrix} C\theta_j & -S\theta_j & 0 & d_j \\ C\alpha_j S\theta_j & C\alpha_j C\theta_j & -S\alpha_j & -r_j S\alpha_j \\ S\alpha_j S\theta_j & S\alpha_j C\theta_j & C\alpha_j & r_j C\alpha_j \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (35)$$

donde  $Sx$  y  $Cx$  representan las funciones  $\text{Sen}(x)$  y  $\text{Cos}(x)$  respectivamente.

Para obtener la matriz de transformación de la cadena terminal respecto a la base, utilizando las propiedades de las matrices de transformación homogénea, se obtiene:

$${}^0T_n = {}^0T_1 {}^1T_2 \dots {}^{n-1}T_n \quad (36)$$

### 2.8.1. Cinemática inversa

La cinemática inversa es un problema que surge cuando se tiene una cadena de articulaciones y se desea saber las rotaciones que se deben aplicar a las articulaciones para que el efector final llegue a la posición y rotación deseada, relativa a la base. Existen dos formas de resolver este problema, utilizando el método geométrico y el método algebraico.

El método geométrico consiste en resolver el problema utilizando las propiedades geométricas de la cadena de articulaciones. Para esto, se descompone la geometría espacial del manipulador (cadena de articulaciones) en varios problemas de geometría plana [Kucuk y Bingul, 2006].

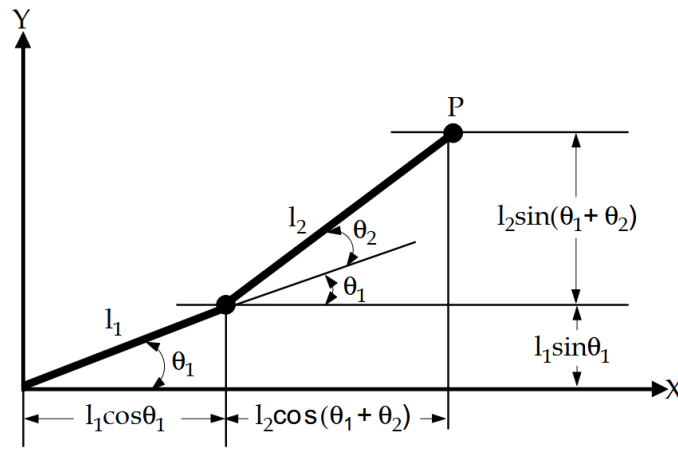


Figura 10: Ejemplo de la geometría espacial de un manipulador simple. Fuente: [Kucuk y Bingul, 2006].

En la Figura 10 se puede observar un ejemplo de la geometría espacial de un manipulador simple. Para resolver la cinemática inversa de este modelo, se deben encontrar los valores de  $\theta_1$  y  $\theta_2$  los cuales se obtienen mediante trigonometría. La solución para este manipulador es:

$$\theta_2 = \text{atan2} \left( \pm \sqrt{1 - \left( \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right)^2}, \frac{p_x^2 + p_y^2 - l_1^2 - l_2^2}{2l_1 l_2} \right) \quad (37)$$

$$\theta_1 = \text{atan2} \left( \pm \sqrt{1 - \left( \frac{p_x(l_1 + l_2 m) + p_y l_2 n}{p_x^2 + p_y^2} \right)^2}, \frac{p_x(l_1 + l_2 m) + p_y l_2 n}{p_x^2 + p_y^2} \right) \quad (38)$$

donde,

$$m = \cos(\theta_2) \quad (39)$$

$$n = \text{sen}(\theta_2) \quad (40)$$

$$p_x = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \quad (41)$$

$$p_y = l_1 \text{sen}(\theta_1) + l_2 \text{sen}(\theta_1 + \theta_2) \quad (42)$$

Se puede notar que el problema se vuelve más complejo de resolver con este método si el manipulador se compone de más articulaciones y/o estuviese en un espacio tridimensional. En esos casos se utiliza el método algebraico, el cual consiste en encontrar una expresión matemática que permita resolver las incógnitas del problema. Esto se logra utilizando la ecuación (36), donde los valores de la matriz  ${}^0T_n$  son conocidos. Luego, se construye un sistema de ecuaciones no lineales para encontrar las variables de cada articulación.

## CAPÍTULO 3

### PROPUESTA DE SOLUCIÓN

El enfoque del sistema propuesto es mejorar la calidad del seguimiento de manos para ofrecer una mejor experiencia de usuario al momento de interactuar con los objetos del laboratorio virtual. Para ello, se ha modificado el sistema anterior incorporando mejoras y optimizaciones para corregir problemas críticos y mejorar la fluidez y precisión del seguimiento de las manos.

El sistema de seguimiento de manos se divide en tres etapas:

1. **Calibración de cámaras:** en esta etapa se realiza la calibración de las cámaras, es decir, obtener los parámetros intrínsecos y extrínsecos de cada cámara. Estos parámetros contienen la información de las características propias de la cámara (distancia focal, centro de la imagen y distorsión del lente) y su pose respecto a un punto en el espacio 3D.
2. **Captura de manos:** en esta etapa se utiliza MediaPipe para detectar los puntos de referencia de las manos capturadas por el sistema estereoscópico de cámaras. Luego, mediante el uso de los parámetros adquiridos en la etapa anterior y DLT, se obtienen las coordenadas 3D de los puntos de referencia detectados, para luego, ser enviadas a un middleware que se comunica con Unity.
3. **Cinemática inversa:** aquí se utilizan los puntos de referencia detectados para posicionar los *targets*<sup>13</sup> utilizados por el *solver*<sup>14</sup> de cinemática inversa para cambiar la pose de las manos virtuales (Modelo 3D articulado).

En la Figura 11 se puede apreciar la arquitectura de la propuesta de solución.

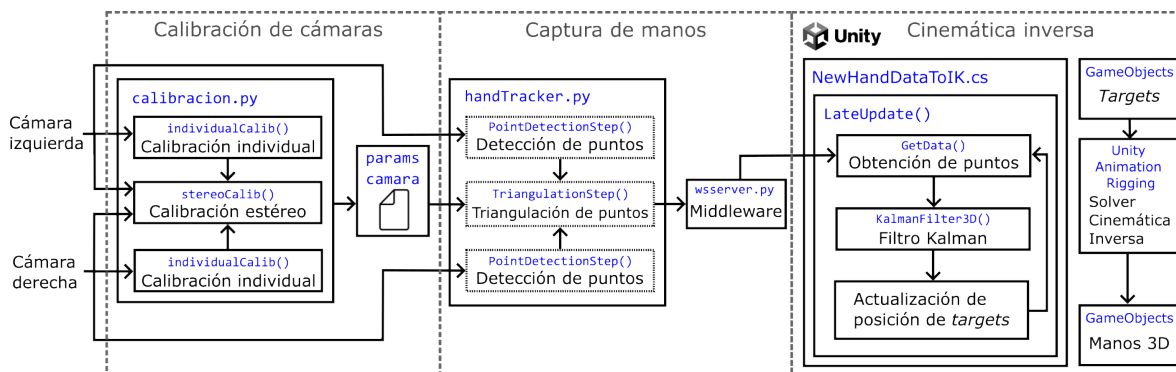


Figura 11: Arquitectura de la solución. Fuente: Elaboración propia.

<sup>13</sup>Posición objetivo de los efectores finales de un objeto articulado

<sup>14</sup>Programa utilizado para resolver un problema

A continuación, se presentan por etapa los cambios propuestos al sistema anterior de seguimiento de manos.

### 3.1. Calibración de cámaras

La calibración de las cámaras es un proceso indispensable en cualquier sistema de reconocimiento a través de cámaras, ya que, nos proporciona información sobre su ubicación en el espacio y parámetros propios como la distancia focal, distorsión del lente y centro de la imagen.

Mientras mejor sea la calibración, mayor será la precisión en la detección de las manos. En base a este hecho se ha planteado la hipótesis de que la distorsión presente en los datos capturados de las manos se debe a que los parámetros obtenidos no son muy precisos. Esta hipótesis cobra fuerza al comparar el error de los parámetros calculados por OpenCV<sup>15</sup>. En la estimación de parámetros intrínsecos es posible obtener un error bajo ( $RMSE^{16} < 0.5$ ). Por otro lado, el error de los parámetros extrínsecos es siempre alto, sin importar cómo se posicione el tablero de calibración.

Para mejorar la calidad de los parámetros extrínsecos se ha optado por explorar una alternativa que permita resolver el problema de  $PnP$  y entregar mejores resultados que otros métodos. Entre los métodos existentes, RDLT cumple con estos requisitos, ya que, es un método diseñado para resolver de manera óptima el problema de  $PnP$ , es estable y entrega resultados más precisos que métodos como DLT, EPnP<sup>17</sup>[Lepetit *et al.*, 2009], DLS<sup>18</sup>[Hesch y Roumeliotis, 2011], solo siendo igualado en algunas pruebas por OPnP<sup>19</sup>[Zheng *et al.*, 2013]. Por lo tanto, se ha optado por implementar este método para cumplir con el objetivo de mejorar la calidad de los parámetros extrínsecos.

Al implementar este método se debió tener en consideración que a diferencia de los métodos ofrecidos por OpenCV, este no toma en cuenta la distorsión del lente y tampoco puede refinar los parámetros estimados cuando se utiliza más de una imagen del tablero de calibración. Para solucionar esto, se utilizó la función `cv2.undistort()` de OpenCV para corregir la distorsión del lente en las imágenes y *Bundle Adjustment* para el refinamiento de los parámetros.

La técnica de *Bundle Adjustment* fue implementada utilizando la función `least_squares()` de Scipy<sup>20</sup>, donde la función a minimizar corresponde al error de re-proyección de los puntos. El error de re-proyección se obtiene de la diferencia entre las coordenadas de los puntos del

---

<sup>15</sup><https://opencv.org/>

<sup>16</sup>Root Mean Square Error

<sup>17</sup>Efficient  $PnP$

<sup>18</sup>Direct Least-Squares

<sup>19</sup> $O(n)$   $PnP$

<sup>20</sup><https://scipy.org/>

tablero de calibración extraídos de la imagen (mediante `cv2.findChessboardCorners()`) y las coordenadas teóricas proyectadas en el plano utilizando los parámetros estimados.

La suposición inicial corresponde a los parámetros estimados por RDLT de cada imagen, distribuidos de la siguiente forma:

$$[\text{rot}_1 \quad \text{trans}_1 \quad \text{rot}_2 \quad \text{trans}_2 \quad \dots \quad \text{rot}_n \quad \text{trans}_n]$$

Donde  $\text{rot}_i$  y  $\text{trans}_i$  son los parámetros estimados de la imagen  $i$ . Debido a que la suposición inicial debe ser una matriz unidimensional, la matriz de rotación estimada por RDLT es convertida en vector mediante la función `cv2.rodrigues()`.

Para obtener los parámetros extrínsecos finales, simplemente se promedian los valores refinados. En el caso de la matriz de rotación, se debe transformar el vector de rotación refinado utilizando la función `cv2.rodrigues()`.

En resumen, la estimación mejorada de parámetros extrínsecos funciona de la siguiente forma:

---

**Input:** Parámetros intrínsecos de las cámaras.

**Output:** Parámetros extrínsecos de las cámaras.

- 1: Capturar imágenes del tablero de calibración en distintas posiciones.
  - 2: Corregir la distorsión del lente de las imágenes capturadas mediante `cv2.undistort()`.
  - 3: Extraer las esquinas (puntos) del tablero de cada imagen mediante `cv2.findChessboardCorners()`.
  - 4: Por cada conjunto de puntos, estimar los parámetros extrínsecos mediante RDLT y guardarlos en un arreglo (uno por cámara).
  - 5: Refinar los parámetros mediante *Bundle Adjustment*.
  - 6: Calcular la media de los parámetros refinados para obtener los parámetros extrínsecos de las cámaras.
- 

### 3.2. Captura de manos

El problema principal presente en el diseño de la arquitectura del sistema anterior se encuentra en no aprovechar los recursos del ordenador eficientemente. Debido a que, un solo proceso se encarga de la tarea de procesar los datos generados por dos cámaras. Esto afecta negativamente la calidad del resultado final entregado por el sistema.

Para solucionar este problema, se ha optado por refactorizar el código, de tal forma, que pueda aprovechar mejor los recursos del entorno de ejecución. Para ello, se han identificado las tareas que se realizan para convertirlas en funciones que trabajen paralelamente mediante la librería `Multiprocessing`<sup>21</sup> de Python.

---

<sup>21</sup><https://docs.python.org/3/library/multiprocessing.html>

Además de paralelizar las tareas, también se realiza una interpolación lineal de los datos. Esto se hace para incrementar aún más la fluidez del movimiento de las manos virtuales.

A continuación, se muestran las tareas identificadas:

- **Detección de puntos de referencia:** en esta tarea, se extraen los fotogramas enviados por la cámara, luego, Mediapipe procesa los fotogramas y entrega las coordenadas 2D de los puntos de referencia detectados. Esta tarea también se encargará de realizar la interpolación lineal de los puntos.
- **Triangulación de puntos:** esta tarea consiste en obtener las coordenadas 3D de los puntos de referencia. Para ello, se utiliza DLT y los parámetros de ambas cámaras para realizar la triangulación a partir de las coordenadas 2D resultantes de la tarea anterior.

Estas tareas son asignadas a subprocesos que trabajan en paralelo, teniendo tres subprocesos en total, dos se encargan de la detección de puntos (uno por cámara) y la otra se encarga de la triangulación de puntos. Además, cada subproceso encargado de realizar la detección de puntos cuenta con su propia estructura FIFO<sup>22</sup> (cola) para enviar los puntos obtenidos. De esta forma, el sub-proceso encargado de realizar la segunda tarea puede acceder a las colas y obtener los datos requeridos.

También se utilizan eventos para comunicar el estado entre subprocesos. Un evento es una variable booleana que cambia de valor cuando ocurre una acción, de forma que, los demás subprocesos que tienen acceso a esta variable puedan enterarse. A continuación, se presentan los eventos utilizados y su funcionamiento:

- **stopEvent:** este evento ocurre cuando uno de los subprocesos de detección de puntos se cierra, ya sea, por la acción del usuario o cuando ocurre una excepción.
- **dltEvent:** este evento ocurre cuando se inicia el subproceso de triangulación de puntos y es utilizado para comunicar a los otros subprocesos que ya pueden comenzar la captura de datos, de esta forma, se mantienen sincronizados.
- **selfHandDetectEvent** y **otherHandDetectEvent:** ambos eventos tienen la misma función que es informar que se ha detectado al menos una mano en una de las cámaras, con la diferencia de que **selfHandDetectEvent** es el evento que informa que el propio subproceso ha detectado algo, y **otherHandDetectEvent** informa si el otro subproceso ha detectado algo. De esta forma, se mantienen sincronizados los datos enviados a la cola.

En las funciones 1, 2 y 3 se puede presenciar la implementación de las tareas mencionadas y la función principal en pseudocódigo.

---

<sup>22</sup>First-In First-Out

---

**Función 1** Detección de puntos de referencia

---

```
1: function PointDetectionStep(camId, resultQueue, stopEvent, dltEvent,  
   selfHandDetectEvent, otherHandDetectEvent)  
2:   cam ← videoCapture(camId)  
3:   handDetection ← MediaPipe.Hands()  
4:   Apoints, Bpoints ← int array[42][2]  
5:   while True do  
6:     if stopEvent is not set then  
7:       if dltEvent is set then  
8:         frame ← cam.read()  
9:         result ← handDetection(frame)  
10:        if result is not null then  
11:          if selfHandDetectEvent is not set then  
12:            selfHandDetectEvent.set()  
13:          end if  
14:          if otherHandDetectEvent is set then      ▷ Si el otro sub-proceso  
también ha detectado manos, se ejecuta el resto.  
15:            resultQueue.put(Apoints)  
16:            Bpoints ← getDetectedHandPoints(result)  
17:            interpolatedPoints ← linearInterp(Apoints, Bpoints)  
18:            resultQueue.put(interpolatedPoints)  
19:            Apoints ← Bpoints  
20:          end if  
21:          else  
22:            selfHandDetectEvent.clear()      ▷ Se reinicia el evento al estado  
inicial.  
23:          end if  
24:        end if  
25:        if isKeyPressed(Esc) then      ▷ Si el usuario presiona Esc, finalizar todos los  
sub-procesos  
26:          stopEvent.set()  
27:          return                        ▷ Finaliza la ejecución del sub-proceso.  
28:        end if  
29:      else  
30:        return                        ▷ Finaliza la ejecución del sub-proceso.  
31:      end if  
32:    end while  
33: end function
```

---

---

**Función 2** Triangulación de puntos

---

```
1: function TriangulationStep(resultQueueL, resultQueueR, stopEvent, dltEvent, PL,  
   PR)  
2:   dltEvent.set();  
3:   while True do  
4:     if ResultQueueL is not empty and ResultQueueR is not empty then  
5:       pointsL  $\leftarrow$  resultQueueL.get()  
6:       pointsR  $\leftarrow$  resultQueueR.get()  
7:       triangulatedPoints  $\leftarrow$  DLT(P0, P1, pointsL, pointsR)  
8:       sendToMiddleware(triangulatedPoints)  
9:     else if stopEvent is set then  
10:      return ▷ Finaliza la ejecución del sub-proceso.  
11:    end if  
12:  end while  
13: end function
```

---

---

**Función 3** Proceso principal

---

```
1: camIdL  $\leftarrow$  userInput[1]  
2: camIdR  $\leftarrow$  userInput[2]  
3: PL  $\leftarrow$  readCameraParams(camIdL)  
4: PR  $\leftarrow$  readCameraParams(camIdR)  
5: resultQueueL, resultQueueR  $\leftarrow$  createQueue()  
6: stopEvent, dltEvent  $\leftarrow$  createEvent()  
7: HandDetectEventL, HandDetectEventR  $\leftarrow$  createEvent()  
8: do in parallel  
9:   PointDetectionStep(camIdL, resultQueueL, stopEvent, dltEvent,  
    HandDetectEventL, HandDetectEventR)  
10:  PointDetectionStep(camIdR, resultQueueR, stopEvent, dltEvent,  
    HandDetectEventR, HandDetectEventL)  
11:  TriangulationStep(resultQueueL, resultQueueR, stopEvent, dltEvent, PL,  
    PR)  
12: end parallel
```

---

### 3.3. Cinemática inversa

En esta etapa, se utilizan los puntos detectados en la etapa anterior para interpretar el movimiento de las manos reales y trasladarlo a las manos virtuales. Para entregar una buena experiencia, es importante que el movimiento sea fluido y preciso. A pesar de que los cambios propuestos buscan solucionar estos problemas, el ruido presente en los datos es un problema inevitable en este tipo de sistemas, ya que, depende de la calidad del sensor de la cámara y factores externos como la iluminación del entorno.

Para solucionar este problema se ha implementado un filtro Kalman, que se encargará de suavizar el movimiento tembloroso de las manos virtuales producto del ruido de los datos. Para minimizar el impacto en el rendimiento del seguimiento de manos, solo se aplica el filtro a algunos de los puntos obtenidos de la etapa anterior, los puntos 0, 5, 9 y 17 de la Figura 5, para cada mano. El motivo por el que se prioriza estos puntos es porque son utilizados para actualizar la posición y rotación de los pivotes de las manos. Estos contienen todos los componentes de una mano (articulaciones, *targets*, entre otros), de forma que, un cambio en la posición del pivote también se verá reflejado en estos componentes.

Luego de posicionar y rotar las manos, se procede a posicionar los *targets*, que son puntos en el espacio utilizados por el *solver* de cinemática inversa para estimar las rotaciones que se deben aplicar a las articulaciones de la mano para que los efectores finales se acerquen a la posición del *target*. Para que las puntas de los dedos estén orientadas correctamente, se utilizan otros *targets* que llamaremos *aim targets*. En la Figura 12 se puede observar cómo los *targets* utilizados afectan la pose de la mano.

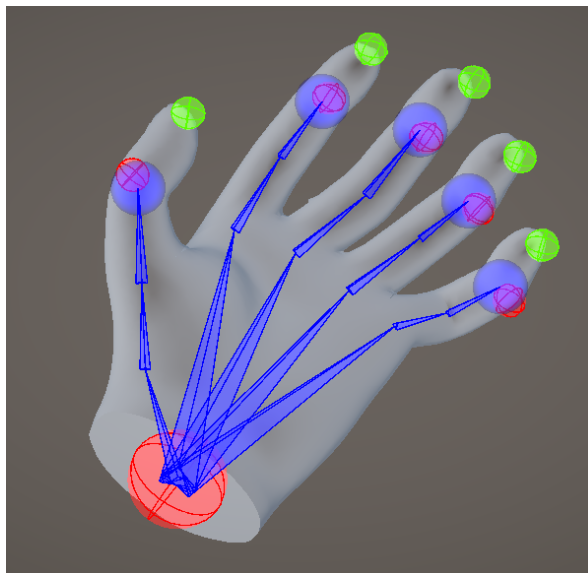


Figura 12: Visualización de los componentes de una mano virtual: Articulaciones de la mano (Azul), *targets* (rojo) y *aim targets* (verde). Fuente: Elaboración propia.

Finalmente, el *solver* de Cinemática Inversa de Unity (Que se encuentra en el paquete llamado *Animation Rigging toolkit*) estima la rotación de las articulaciones de las manos. Luego se repite el proceso para el siguiente conjunto de datos recibido.

Otro cambio realizado fue trasladar toda la lógica de actualización de la posición de los *targets* a la función `Update()`. Antes, parte de la lógica se encontraba en la función `Update()` y el resto en la función `FixedUpdate()`. Esto puede generar problemas de sincronización, ya que, `FixedUpdate()` es llamado más de una vez por fotograma y `Update()` solo es llamado una vez por fotograma.

A continuación, se presenta en pseudocódigo la lógica utilizada para posicionar las manos y los *targets*.

---

**Función 4** Actualizador de posición de targets

---

**Input:** *leftHand, rightHand, pNoise, mNoise*.

```
1: class NewHandDataToLK
2:   function Start( )
3:     msc  $\leftarrow$  MiddlewareSocketClient()
4:     leftTargets  $\leftarrow$  getTargets(leftHand)
5:     rightTargets  $\leftarrow$  getTargets(rightHand)
6:     leftFilteredPos  $\leftarrow$  []
7:     rightFilteredPos  $\leftarrow$  []
8:     i  $\leftarrow$  0
9:     for bone in [wrist, index, middle, pinkie] do
10:       leftFilteredPos[i]  $\leftarrow$  KalmanFilter.Init(Vector3.zero, pNoise, mNoise)
11:       rightFilteredPos[i]  $\leftarrow$  KalmanFilter.Init(Vector3.zero, pNoise, mNoise)
12:       i  $\leftarrow$  i + 1
13:     end for
14:   end function
15:   function Update( )
16:     targetsPos  $\leftarrow$  GetDataFromMiddleware(msc)
17:     leftTargetsPos  $\leftarrow$  targetsPos.left
18:     rightTargetsPos  $\leftarrow$  targetsPos.right
19:     i  $\leftarrow$  0
20:     for bone in [wrist, index, middle, pinkie] do
21:       leftFilteredPos[i]  $\leftarrow$  KalmanFilter.Update(leftTargetsPos.bone, deltaTime)
22:       rightFilteredPos[i]  $\leftarrow$  KalmanFilter.Update(rightTargetsPos.bone, deltaTime)
23:       i  $\leftarrow$  i + 1
24:     end for
25:     SetHandPositionAndRotation(leftHand, leftFilteredPos)
26:     SetHandPositionAndRotation(rightHand, rightFilteredPos)
27:     SetAllTargetsPositionAndRotation(leftTargets, leftTargetsPos)
28:     SetAllTargetsPositionAndRotation(rightTargets, rightTargetsPos)
29:     SetAllTargetsPositionAndRotation(leftTargets, leftTargetsPos)
30:     SetAllAimTargetsPositionAndRotation(rightTargets, rightTargetsPos)
31:   end function
32: end class
```

---

## CAPÍTULO 4

### VALIDACIÓN DE LA SOLUCIÓN

#### 4.1. Especificaciones de hardware

En esta sección se presentarán las especificaciones del *hardware* utilizado para realizar las pruebas. Para las etapas de calibración y reconocimiento de manos, se utilizaron dos cámaras web con las siguientes especificaciones:

- Resolución: 2 Megapíxeles
- Distancia focal: 3.6 [mm]
- Tamaño del sensor: 1/3"
- Angulo de visión: 110°

Las especificaciones del computador utilizado para la ejecución de los métodos y pruebas son las siguientes:

- Procesador Intel® Core™ i5 8400
- 16 GB de RAM<sup>23</sup>

#### 4.2. Calibración de las cámaras

El proceso de calibración es la etapa más importante, debido a que, la calidad de la calibración afectará la calidad de los resultados en las etapas posteriores. Dicho esto, el objetivo de las pruebas será validar solo la calidad de la propuesta, esto incluye características como la precisión y la consistencia. La eficiencia no será importante en este caso, debido a que esta etapa se realiza una sola vez, siempre y cuando se conserve la posición de las cámaras.

Para corroborar que la precisión de los resultados que entrega el método propuesto de calibración de las cámaras es igual o mejor que los entregados por el método anterior, se comparó la variación del error cuadrático medio o RMSE<sup>24</sup> según el número de imágenes (fotogramas) utilizados para estimar las poses de las cámaras. Para calcular el RMSE, se utiliza la

---

<sup>23</sup>Random Access Memory

<sup>24</sup>Root Mean Square Error

siguiente fórmula:

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (\hat{y}_i - y_i)^2}{N}} \quad (43)$$

donde  $N$  es el número total de fotogramas utilizados para la estimación de la pose de las cámaras,  $\hat{y}_i$  corresponde a los puntos 2D obtenidos y  $y_i$  corresponde a los puntos 3D proyectados mediante la pose estimada.

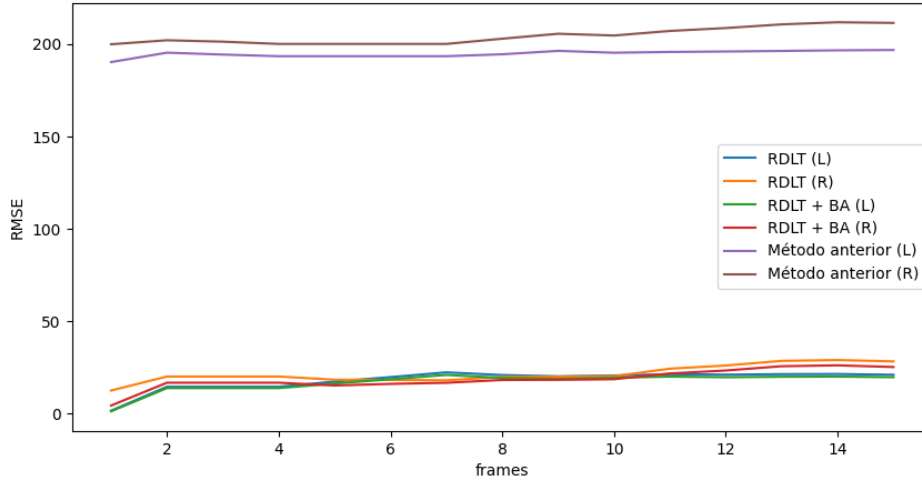


Figura 13: Variación del RMSE según el número de fotogramas por cámara (L=Left, R=Right).  
Fuente: Elaboración propia.

En la Figura 13 se puede observar que el RMSE del método propuesto es mucho más bajo que el del método anterior, esto significa que la proyección de los puntos estimados por el método propuesto se acerca más a los valores esperados, es decir, es más preciso.

Luego, para comprobar la consistencia de la calibración entregada por el método propuesto, se optó por graficar distintas poses de una mano obtenidas de la etapa de captura de manos al utilizar los parámetros de calibración del método propuesto y del método anterior. De esta forma, se podrá realizar un análisis cualitativo de la consistencia de los métodos. Las poses utilizadas para esta prueba se pueden observar en la Figura 14.

De estos resultados (Figura 15) se puede observar que el uso de la técnica de BA<sup>25</sup> es crucial para obtener buenos resultados al utilizar RDLT. También se puede observar que la pose de la mano obtenida por el método propuesto se encuentra en la orientación correcta (confirmado por los resultados de la primera prueba) y menos distorsionada que la obtenida por el método anterior, siendo más evidente en las poses 2 y 3.

<sup>25</sup>Bundle Adjustment



Figura 14: Poses de una mano. Fuente: Elaboración propia.

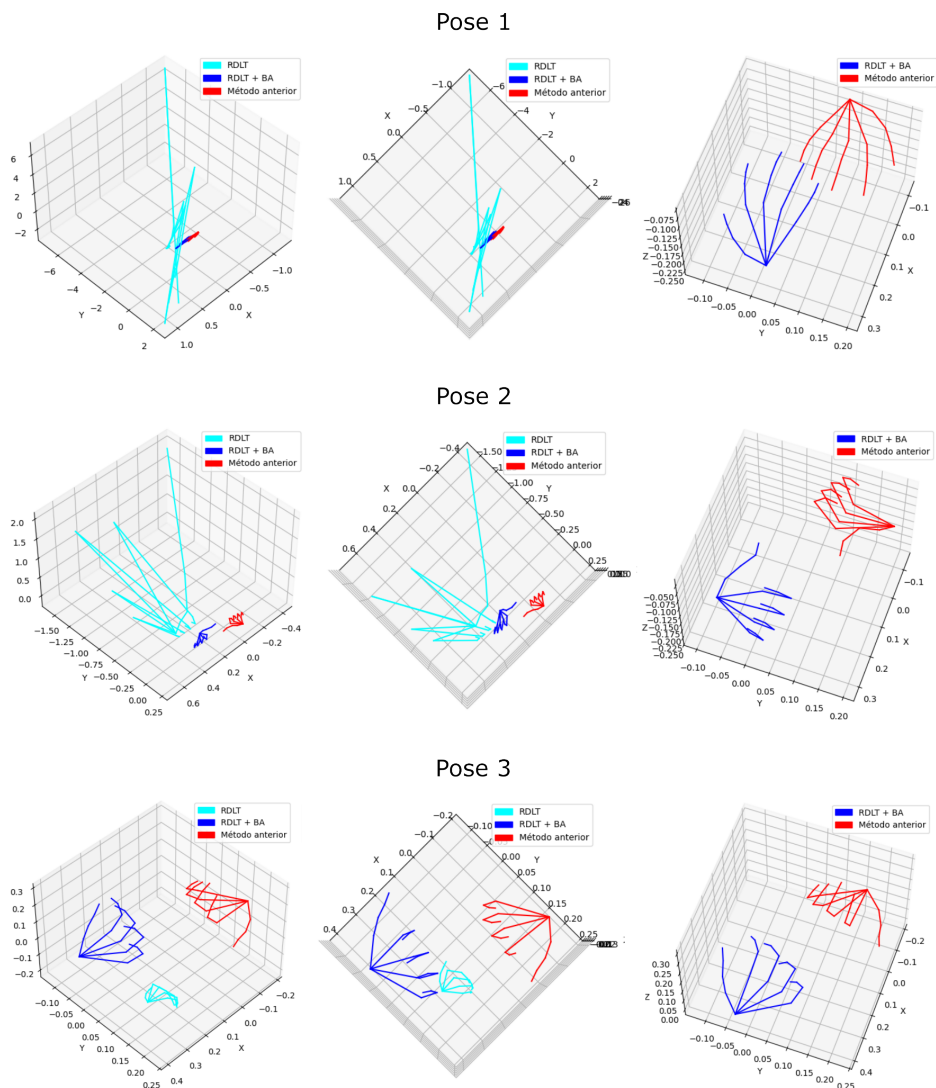


Figura 15: Resultados de la triangulación de puntos por pose. Fuente: Elaboración propia.

### 4.3. Captura de manos

En esta etapa, la característica más importante es la eficiencia, debido a que, mientras más eficiente sea el sistema de captura de manos, más información podrá procesar y entregar, contribuyendo a una mayor fluidez en el resultado entregado por la etapa de cinemática inversa.

Para comprobar la eficiencia de la propuesta, se ha optado por medir el número de FPS<sup>26</sup> que procesa el método propuesto de captura de manos. Para esta prueba se compara el método anterior con el método propuesto con y sin interpolación, de esta forma se analiza el impacto en la eficiencia al paralelizar los procesos.

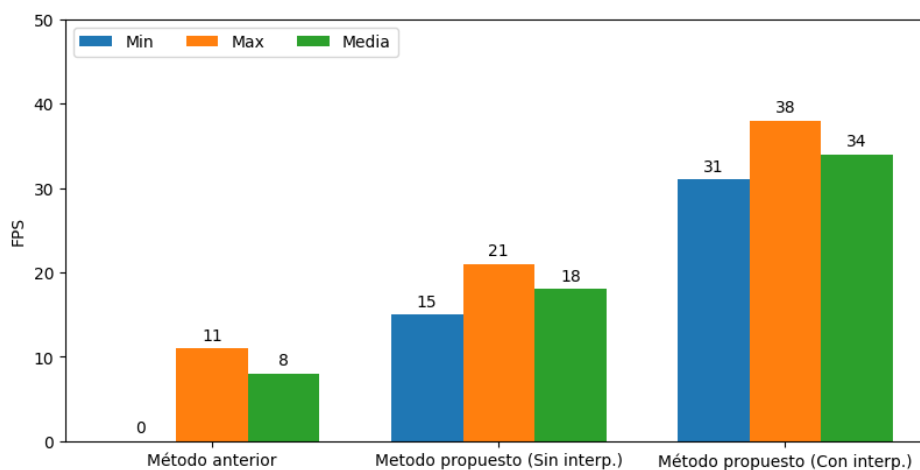


Figura 16: Comparativa del número de FPS por método. Fuente: Elaboración propia.

De los resultados (Figura 16) se puede observar que el método propuesto ofrece una mejora de un 225 % (sin interpolación) y un 425 % (con interpolación) en comparación con el método anterior, demostrando ser significativamente más eficiente. Además, podemos notar que la variación de los mínimos y máximos respecto a la media son menores en el caso del método propuesto, lo que implica una mayor estabilidad. Esto último se puede corroborar al analizar la gráfica de variación de los FPS en el tiempo (Figura 17) donde se pueden observar que las variaciones del método anterior son más abruptas que las del método propuesto.

<sup>26</sup>Fotogramas por segundo

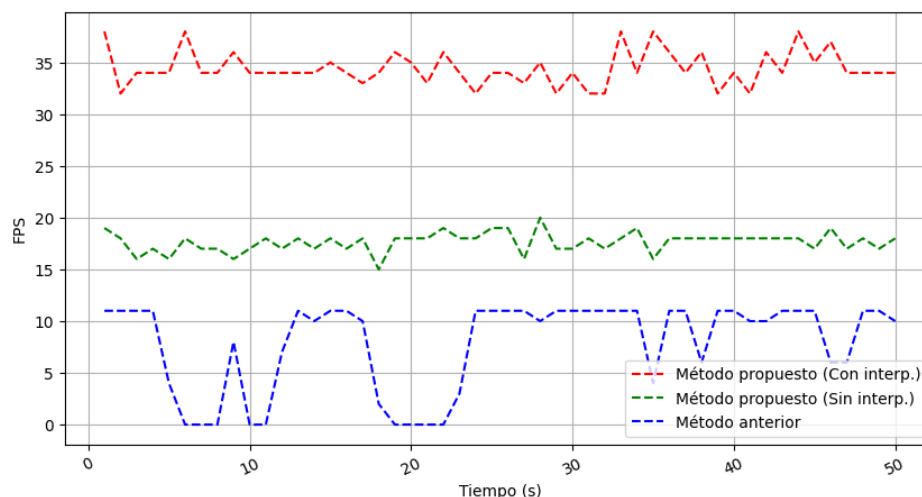


Figura 17: Variación de los FPS en el tiempo. Fuente: Elaboración propia.

#### 4.4. Cinemática inversa

En esta etapa, es importante que las poses de las manos virtuales entregadas por el solver de cinemática inversa no difieran mucho de las poses capturadas por la etapa anterior, en otras palabras, lo que se busca es precisión.

Para saber cuánto difiere la pose entregada por el solver de la pose capturada, se decidió medir la distancia entre los efectores finales de la mano virtual y los puntos equivalentes obtenidos de la etapa anterior, luego, las distancias obtenidas son promediadas para obtener la distancia euclidiana promedio de toda la mano para facilitar el análisis de los resultados, de lo contrario, se tendría que graficar la distancia para cada punto.

Para observar también cómo el movimiento afecta esta distancia, se realizaron tres pruebas distintas, en la primera prueba se dejaron las manos estáticas, en la segunda prueba, se abrieron y cerraron las manos de forma periódica y en la tercera prueba al comienzo se dejaron las manos estáticas y luego de un tiempo se realizaron movimientos. Los resultados obtenidos se pueden observar en las Figuras 18 y 19.

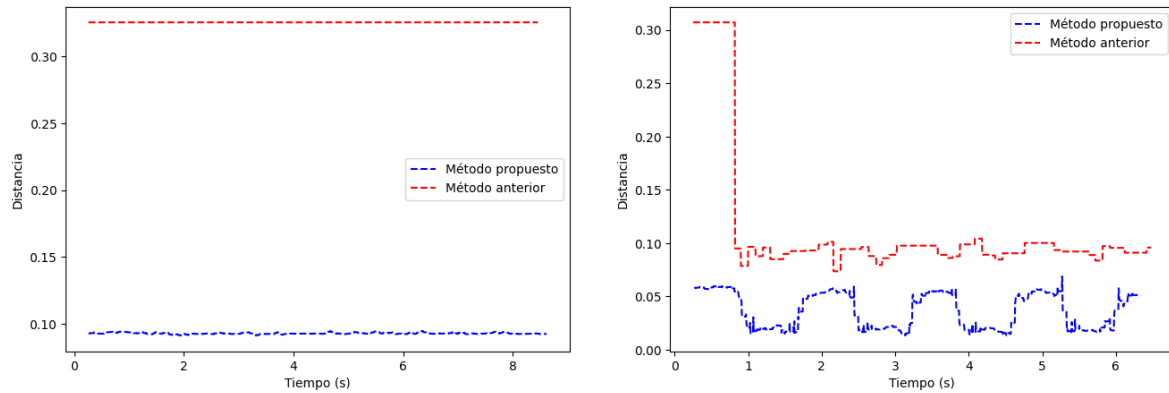


Figura 18: Distancia Euclidiana Promedio en captura de manos sin movimiento (izquierda) y con movimiento (derecha). Fuente: Elaboración propia.

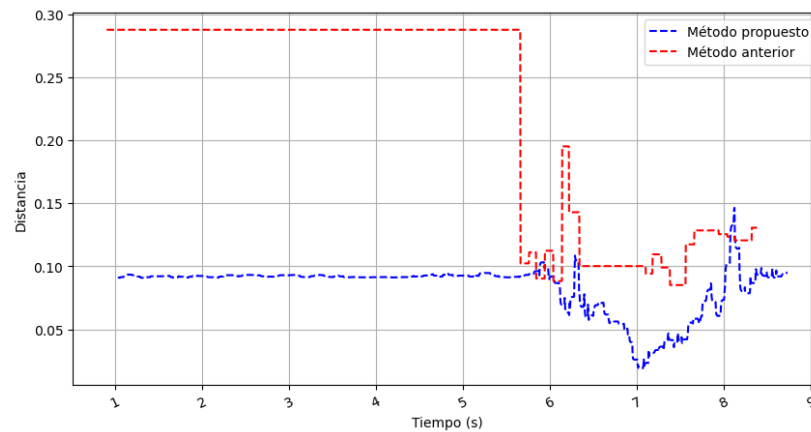


Figura 19: Distancia Euclidiana Promedio en captura de manos con movimiento mixto. Fuente: Elaboración propia.

Se puede observar que en las tres pruebas la distancia promedio en el método propuesto es menor, por lo tanto, es más preciso. También se puede observar en los gráficos que la línea del método propuesto tiene más detalles, esto se debe a que, en la etapa de captura de manos se envía más información por segundo, aportando una mayor fluidez en el movimiento de la mano virtual. Por otro lado, se observa que la línea del método anterior oscila menos cuando hay movimiento, por lo que, es más consistente que el método propuesto en esa situación.

Pero esto no es del todo correcto, debido a que el método anterior tiene problemas para estabilizarse cuando no se detectan bien las manos. Para evidenciar esto, se ha repetido la prueba anterior pero esta vez alternando entre movimientos en los que se detecta y no se detecta correctamente la muñeca de la mano.

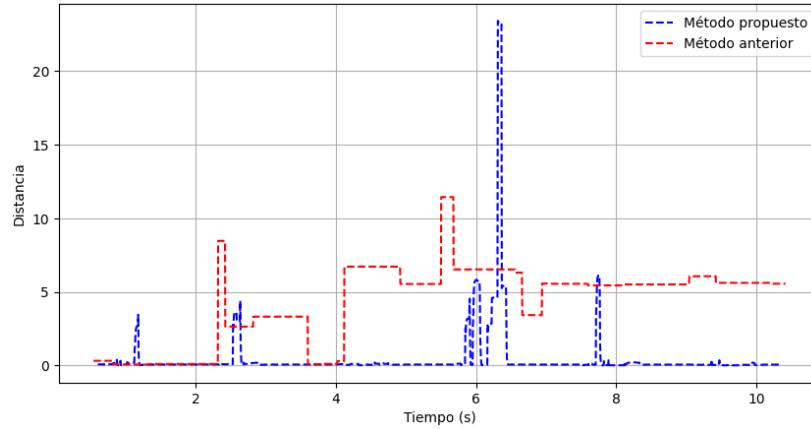


Figura 20: Distancia Euclidiana Promedio en captura de manos con movimiento de la muñeca.  
Fuente: Elaboración propia.

Se puede observar claramente en el gráfico de la Figura 20 el problema de estabilidad del método anterior, mientras que el método propuesto se estabiliza rápidamente, demostrando ser más consistente en general.

También se ha medido el tiempo de CPU<sup>27</sup> promedio de la función principal de ambos métodos, con el objetivo de analizar el impacto en el rendimiento de los cambios propuestos. Para ello, se ha utilizado la herramienta *Profile Analyzer* proporcionada por Unity, que permite analizar en profundidad el rendimiento del proyecto durante su ejecución. En la Tabla 1 se encuentran los tiempos de CPU en milisegundos obtenidos del *Profile Analyzer*.

|                  | Tiempo de CPU |
|------------------|---------------|
| Método propuesto | 0.04 [ms]     |
| Método anterior  | 0.02 [ms]     |

Tabla 1: Comparación del tiempo de CPU promedio por método. Fuente: Elaboración propia.

Los resultados nos muestran un impacto negativo en el rendimiento, pero despreciable si tomamos en cuenta la magnitud de los valores y los beneficios que aporta el método propuesto al sistema de seguimiento de manos.

<sup>27</sup>Central Processing Unit

## CAPÍTULO 5

### CONCLUSIONES

El objetivo de este trabajo se centra en mejorar una de las funciones más importantes del laboratorio virtual de química “A Darle Átomos”, el sistema de seguimiento de manos. Esto se debe a que, los problemas presentes en el sistema dificultan considerablemente la experiencia de los usuarios al momento de interactuar con los objetos del entorno virtual, afectando una de las características clave que diferencia a esta aplicación de la competencia, la inmersión.

La importancia de este problema ha sido la motivación para desarrollar una solución integral. Esto se ha logrado mediante la realización de distintos cambios en los procesos del sistema de seguimiento de manos, basándose en los problemas identificados.

Estos cambios han mejorado notoriamente tanto el rendimiento del sistema como la precisión y fluidez del movimiento de las manos en el entorno virtual. Estas afirmaciones son respaldadas por los resultados obtenidos de la validación de la solución. En el proceso de **calibración de cámaras** se ha logrado mejorar la calidad de los parámetros extrínsecos al utilizar **RDLT**. Esto es evidente al observar la comparativa de las poses obtenidas en la Figura 15, donde las diferencias entre el método anterior y el propuesto se vuelven más evidentes. Se puede observar que a través del método propuesto se puede obtener una pose más cercana a la real que el método anterior. Este análisis se hace ignorando la orientación, ya que, se puede corregir fácilmente rotando los puntos.

En el proceso de **captura de manos**, los resultados demuestran un incremento considerable en su rendimiento. En la Figura 16 se puede observar que los cambios realizados han aumentado la tasa de FPS en aproximadamente un 400 %, demostrando la efectividad de la **paralelización** de las tareas y el uso de **interpolación lineal**.

En cuanto al proceso de **cinemática inversa**, se puede concluir que se ha mejorado exitosamente la fluidez y precisión en el movimiento de las manos virtuales. Esto se puede comprobar al observar los gráficos de las Figuras 18 y 19, que demuestran que el método propuesto logra que la mano virtual se asemeje más a la mano detectada que el método anterior. El aumento en la resolución de los datos generados por el método propuesto se puede atribuir al **Filtro Kalman** que predice la posición de la mano mientras se está procesando el siguiente fotograma durante la **captura de manos**. Además, el gráfico de la Figura 20 nos demuestra que el método propuesto es más estable que el anterior en situaciones en las que el sistema no detecta correctamente las manos.

En conclusión, se han cumplido exitosamente los objetivos presentados en esta memoria. El **objetivo general** de «Aumentar la precisión y rendimiento del sistema de seguimiento de manos anterior» se ha logrado mediante la modificación de los procesos clave que conforman el sistema de seguimiento de manos. Los **objetivos específicos** se lograron mediante

la implementación de diversos algoritmos y técnicas, como el uso de RDLT para mejorar la calidad de la calibración de las cámaras, la paralelización de tareas clave para mejorar el rendimiento de la captura de manos y la implementación de un filtro Kalman para mejorar la fluidez y estabilidad del movimiento de las manos virtuales.

## 5.1. Trabajo futuro

Si bien se ha logrado mejorar considerablemente el sistema de seguimiento de manos, aún existen problemas que se pueden mejorar. El siguiente listado presenta los posibles cambios que se pueden implementar:

- **Acelerar operaciones por GPU:** para mejorar aún más el rendimiento general del sistema, se puede utilizar librerías como CuPy<sup>28</sup> para acelerar por GPU las operaciones matriciales realizadas durante la calibración de cámaras y captura de manos.
- **Facilitar despliegue del sistema:** para iniciar los procesos de calibración de cámaras y captura de manos, el usuario debe ejecutar tres *scripts* en total. Esto puede ser un problema para aquellos usuarios que tienen un bajo dominio de la informática. Para solucionar esto, se puede unificar estos *scripts* en un solo archivo ejecutable y que se pueda controlar mediante una interfaz de usuario.
- **Ajustar parámetros de imagen:** uno de los factores que ocasiona la aparición de ruido en los datos es el exceso o falta de iluminación en el entorno en el que se capturan las manos. Para solucionar este problema, se puede implementar una función que permita al usuario ajustar parámetros de imagen como el brillo y contraste.

---

<sup>28</sup><https://cupy.dev/>

## REFERENCIAS BIBLIOGRÁFICAS

- [Beggs, 1983] Beggs, J. S. (1983). *Kinematics*. CRC Press.
- [Brovelli Sepúlveda *et al.*, 2018] Brovelli Sepúlveda, F., Cañas Urrutia, F., y Bobadilla Gómez, C. (2018). Herramientas digitales para la enseñanza y aprendizaje de química en escolares chilenos. *Educ. quím.*, 29(3):99.
- [Chapra y Canale, 2010] Chapra, S. C. y Canale, R. P. (2010). *Numerical methods for engineers*. McGraw Hill Higher Education, Maidenhead, England, 6 edición.
- [Chunduru *et al.*, 2021] Chunduru, V., Roy, M., y Chittawadigi, R. (2021). Hand tracking in 3d space using mediapipe and pnp method for intuitive control of virtual globe. En *2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)*, pp. 1–6. IEEE.
- [Denavit y Hartenberg, 1955] Denavit, J. y Hartenberg, R. S. (1955). A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*.
- [Fernández, 2006] Fernández, A. (2006). Motivación hacia el estudio y la cultura escolar: Estado de la cuestión. *Pensamiento psicológico*, 2(6):16–16.
- [Figueroa, 2024] Figueroa, D. (2024). Mitigando errores de concepto en estudiantes de cursos introductorios de programación a través de un videojuego. Tesis de máster, Universidad Técnica Federico Santa María, Santiago.
- [Fischler y Bolles, 1981] Fischler, M. y Bolles, R. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.
- [Google, 2025] Google (2025). Guía de tareas de reconocimiento de gestos. Documentación de MediaPipe. Accessed: 2025-3-13.
- [Hesch y Roumeliotis, 2011] Hesch, J. y Roumeliotis, S. (2011). A direct least-squares (dls) method for pnp. En *2011 International Conference on Computer Vision*, pp. 383–390. IEEE.
- [Kapp, 2014] Kapp, B. Y. K. (2014). Separating fact from fiction. Artículo en Chief Learning Office (CLOmedia). Accessed: 2024-9-13.
- [Ke y Roumeliotis, 2017] Ke, T. y Roumeliotis, S. I. (2017). An efficient algebraic solution to the perspective-three-point problem. En *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7225–7233.
- [Khalil y Dombre, 2004] Khalil, W. y Dombre, E. (2004). *Modeling, identification and control of robots*. Butterworth-Heinemann.
- [Kucuk y Bingul, 2006] Kucuk, S. y Bingul, Z. (2006). *Robot kinematics: Forward and inverse kinematics*. INTECH Open Access Publisher London, UK.

- [Lepetit *et al.*, 2009] Lepetit, V., Moreno-Noguer, F., y Fua, P. (2009). Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166.
- [Lugaresi *et al.*, 2019] Lugaresi, C., Tang, J., Nash, H., McClanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C.-L., Yong, M., y Lee, J. (2019). Mediapipe: A framework for perceiving and processing reality. En *Third workshop on computer vision for AR/VR at IEEE computer vision and pattern recognition (CVPR)*, volumen 2019.
- [Martirossov y Kopecek, 2017] Martirossov, S. y Kopecek, P. (2017). Cyber sickness in virtual reality-literature review. *Annals of DAAAM & Proceedings*, 28.
- [Miller *et al.*, 1980] Miller, N. R., Shapiro, R., y McLaughlin, T. M. (1980). A technique for obtaining spatial kinematic parameters of segments of biomechanical systems from cinematographic data. *Journal of Biomechanics*, 13(7):535–547.
- [Ranganathan, 2004] Ranganathan, A. (2004). The levenberg-marquardt algorithm. *Tutorial on LM algorithm*, 11(1):101–110.
- [Shepperd, 1978] Shepperd, S. W. (1978). Quaternion from rotation matrix. *Journal of guidance and control*, 1(3):223–224.
- [Simonsen, 2021] Simonsen, E. (2021). Análisis del observatorio docente del CIAE: Matrícula de primer año de pedagogía cae 35 % entre 2018 y 2021. Análisis 1er año Pedagogía. Accessed: 2024-11-16.
- [T13, 2022] T13 (2022). Ocho alumnos con quemaduras tras explosión en laboratorio en colegio de quilpué. Noticia T13. Accessed: 2024-11-16.
- [Triggs *et al.*, 1999] Triggs, B., McLauchlan, P. F., Hartley, R. I., y Fitzgibbon, A. W. (1999). Bundle adjustment—a modern synthesis. En *International workshop on vision algorithms*, pp. 298–372. Springer.
- [Tsai, 2003] Tsai, R. (2003). A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on robotics and Automation*, 3(4):323–344.
- [Waldron y Schmiedeler, 2016] Waldron, K. J. y Schmiedeler, J. (2016). Kinematics. *Springer handbook of robotics*, pp. 11–36.
- [Wang *et al.*, 2024] Wang, P., Jiao, B., Yao, P., Wei, X., y Zhang, A. (2024). A robust direct linear transformation for camera pose estimation using points. *Image and Vision Computing*, 141:104883.
- [Wang *et al.*, 2011] Wang, R., Paris, S., y Popović, J. (2011). 6d hands: markerless hand-tracking for computer aided design. En *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pp. 549–558.
- [Wang y Popović, 2009] Wang, R. Y. y Popović, J. (2009). Real-time hand-tracking with a color glove. *ACM transactions on graphics (TOG)*, 28(3):1–8.

- [Welch y Bishop, 1995] Welch, G. y Bishop, G. (1995). An introduction to the kalman filter.
- [Zheng *et al.*, 2013] Zheng, Y., Kuang, Y., Sugimoto, S., Astrom, K., y Okutomi, M. (2013). Re-visiting the pnp problem: A fast, general and optimal solution. En *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2344–2351.