Trabajo Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Seguimiento visual multicámara de sistemas articulados

Autor: José Manuel González Marín

Tutor: Carlos Vivas Venegas

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Trabajo Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

Seguimiento visual multicámara de sistemas articulados

Autor:
José Manuel González Marín

Tutor:
Carlos Vivas Venegas
Profesor titular

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Trabajo Fin de Grado: Seguimiento visual multicámara de sistemas articulados

Autor: José Manuel González Marín

Tutor: Carlos Vivas Venegas

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis profesores

Agradecimientos

Con este trabajo pongo fin a una etapa que comenzó hace cuatro años.

En primer lugar me gustaría agradecer la labor encomiable realizada por mi tutor, por toda su ayuda prestada en una labor que se vió complicada debido a una pandemia mundial que golpeó en mis dos últimos años de estudio.

Por supuesto, a mi familia, madre y hermanos, por todo su apoyo prestado durante este periodo tan complicado de mi vida.

A mis amigos y profesores, que durante todo el curso me han acompañado, en buenos y malos momentos.

José Manuel González Marín

Lebrija, 2021

Resumen

Este proyecto tiene como objetivo asentar las bases para el desarrollo de un método sencillo y de bajo coste, capaz de localizar tanto en posición como en orientación un sistema mecánico articulado mediante la aplicación de técnicas de localización 3D basadas en un sistema multicámara calibrado, que consiste en un cubículo de 3x3x3 metros y cuenta con cuatro cámaras infrarrojas, en posiciones enfrentadas dos a dos.

El algoritmo, desarrollado en lenguaje C++ para Visual Studio, permite mediante la aplicación de técnicas de visión por computador como la Triangulación y el problema de la Perspectiva de N-Puntos (PnP) localizar una serie de marcadores pasivos en el espacio tridimensional. Una distribución de dichos marcadores en puntos clave, junto con un modelo cinemático conocido de la estructura, permite estimar todos los grados de libertad de la misma aplicando mínimos cuadrados sobre una función que relaciona la localización tridimensional de los marcadores con el modelo cinemático. Se muestran resultados experimentales para un conjunto de estructuras de prueba con diferente número de grados de libertad, que permiten evaluar la eficacia del método propuesto y sus posibles mejoras para trabajos futuros.

Este problema tiene aplicación para el seguimiento del movimiendo de sistemas robóticos y vehículos articulados, así como para la captura de movimiento del cuerpo humano o partes de él.

Abstract

This project aims to settle the foundations for the development of a simple and low-cost method, capable of locating both, position and orientation, of an articulated mechanical system through the application of 3D localization techniques based on a calibrated multi-camera system, consisting of a cubicle of 3x3x3 meters and four infrared cameras, placed in opposing positions two by two.

The algorithm, developed in C++ language for Visual Studio, allows by applying some computer vision techniques such as Triangulation and the Perspective N-Point (PnP) problem to locate a series of passive markers in a three-dimensional space. A distribution of these markers at key points, together with a known kinematic model of the structure, enables to estimate all the degrees of freedom by applying least squares on a function that relates the three-dimensional location of the markers with the kinematic model. Experimental results are shown for a set of test structures with different number of degrees of freedom, which allow to assess the effectiveness of the proposed method and its possible improvements for future work.

This problem has application for the tracking of the movement of robotic systems and artificial vehicles, as well as for the capture of movement of the human body or parts of it.

Índice

ÍNDICE DE CÓDIGOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

La captura de movimiento es una técnica ampliamente usada y estudiada desde hace décadas, que consiste en la reconstrucción y digitalización tridimensionál de un cuerpo mediante el uso de vision multicámara. Es decir, trata de reconstruir una escena tridimensional a partir de imágenes en 2 dimensiones. No es posible obtener una solución unívoca a partir de la información que arroja una sola perspectiva, por lo que para poder reconstruir una escena a partir de imágenes, hacen falta un mínimo de dos sistemas de visión que observen la escena desde diferentes ángulos de modo que ciertos puntos de interés (marcadores) de dicha escena sean identificables de forma unívoca en ambas imágenes. No obstante, un mayor número de puntos de vista se traduce en una mayor precisión en la estimación y en la posibilidad de solventar la puntual falta de información causada por la oclusión de algún marcador en alguna de las cámaras.

Dentro de la metodología de captura, según los marcadores utilizados, podemos encontrar varios tipos:

- **Óptico-pasivo**. Hace uso de sensores retrorreflectores que son seguidos por camaras infrarrojas. Este es el método que se utiliza para realizar el seguimiento de la articulación en este trabajo.
- **Óptico-activo**. A diferencia del anterior, los marcadores son leds que se iluminan y son fácilmente indentificables por las cámaras, con la contraposición de ser mas costosos.
- **Sensores inerciales**. Dichos sensors transmiten la información directamente al ordenador. Usados en el ámbito del deporte, por ser mas sencillos de colocar y más prácticos en este ámbito.
- **Mediante video**. En lugar de usar marcadores visuales, se hace uso de cámaras tipo *Kinect* y se emplea un software mucho mas complejo para realizar el seguimiento.

Su uso se encuentra extendido a numerosos sectores de diferentes tipos. La industria cinematográfica, la industria de los videojuegos, el deporte o la medicina, así como la robótica. Todos ellos con sus respectivas diferencias, pero que en esencia parten de la misma base, el seguimiento visual multicámara.

En la industria del cine, se aplica dicha tecnología en conjunto con una serie de técnicas de edición de vídeo para capturar escenas realizadas por actores que no se podrían llevar a cabo en la realidad, ya sea por coste o por imposibilidad (escenas de riesgo, seres ficticios…), y añadirlas posteriormente en las etapas finales de producción. Un ejemplo de ello lo encontramos en películas como *Simbad (2000)*, primera película animada enteramente mediante captura de movimiento. Producciones mas conocidas como *AVATAR* o *El Señor de los Anillos* han hecho uso de esta herramienta para algún aspecto del largometraje.

Lo mismo ocurre en la industria del videojuego. En muchas producciones de alto presupuesto es común que se capturen ciertas escenas con actores reales, de manera que al reproducirlas en el juego quedan escenas mucho mas orgánicas y realistas, así como también se hace uso de la captura de movimiento para recrear expresiones faciales fieles a la realidad.

Por supuesto, en el campo de la Robótica y la Automatización se trata de un área también en uso. Determinar la pose de un brazo manipulador con respecto a una cámara montada externa es fundamental para el control del mismo mediante visión por computador. Mediante el uso de marcadores colocados en las articulaciones, se puede estimar el movimiento del efector final del mismo, lo que lo hace un método general y flexible, aunque caro dependiendo del sistema de captura empleado. Del mismo modo, también se utiliza de manera extendida para obtener la posición y orientación de drones en el espacio, a partir de la información dotada por un sistema de visión estéreo con una alta tasa de refresco de medidas.

En este trabajo se propone una metodología de carácter general para determinar, no sólo la posición espacial de sistemas articulados, sino también los grados de libertad de la articulación, basándose exclusivamente en la posición de una serie de balizas o marcadores pasivos colocados en las articulaciones u otras zonas representativas de las mismas.

1.1. Estado del Arte

Centrándonos ahora en profundidad desde un punto de vista técnico, en esta sección se presentan a continuación algunas alternativas comerciales de captura de movimiento que se utilizan en la actualidad, estudiando pues, el Estado del Arte de esta técnica.

VICON [1] es una empresa con 35 años de antiguedad, dedicada a distribuir tanto *software* como *hardware* dedicado a la captura de movimiento, para ámbitos de diversa índole. En su propia página web, podemos encontrar que, según el tipo de aplicación, te recomienda el sistema preconfigurado que mejor se adapta a tus necesidades, incluso una vista previa de la herramienta de visualización propia disponible.

En este caso, el método de detección es óptico-pasivo, pues se hace uso de marcadores reflectantes conectados al traje de captura de movimento. En cuanto a software, cuenta con el suyo propio, *Tracker,* dedicado a realizar toda la tarea de captura y visualización 3D de los movimientos. También encontramos herramientas como *MathWorks SimuLink*, que utiliza para realizar pruebas *hardware-in-the-loop*, e incluso algorimos más avanzadas como *Markerless*, que permiten la captura sin necesidad de usar marcadores en el proceso.

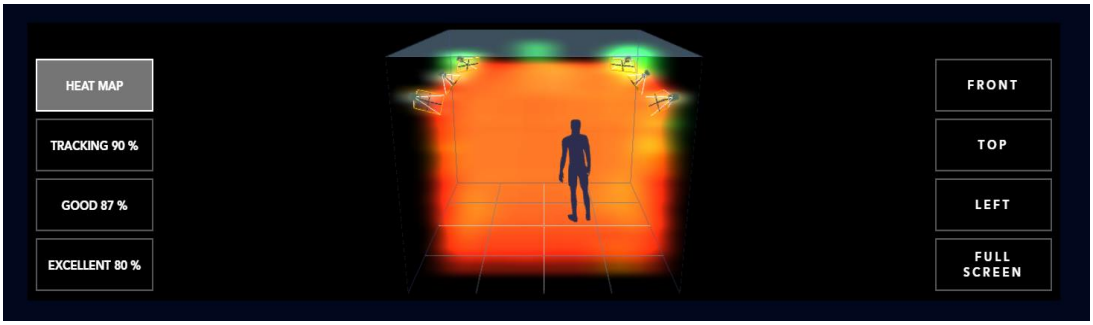

Figura 1-1. Herramienta de visualización VICON.

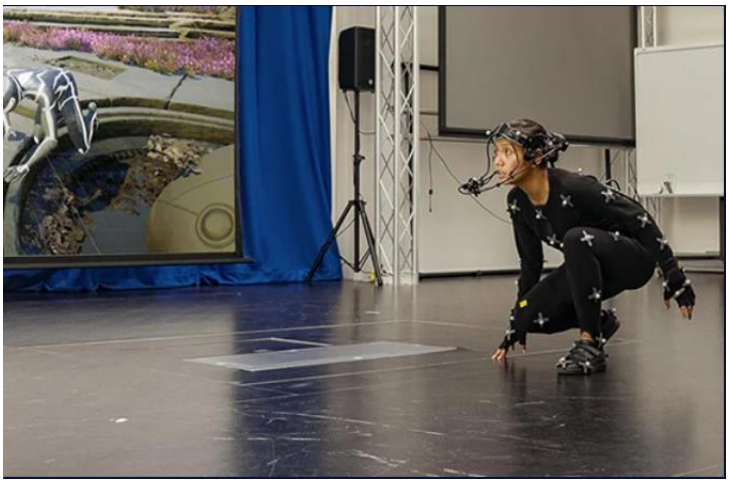

Figura 1-2. Captura de movimiento en curso de VICON.

PHASESPACE [2] es otra empresa dedicada a la distribución y venta de equipos especializados en la captura de movimiento, con la diferencia de que esta hace uso de marcadores óptico-activos conectados al cuerpo del que se quiera hacer seguimiento. Como ellos mismos especifican en su página web, las principales ventajas de usar este tipo de sensores, aunque mas caros, es que permiten realizar captura de movimiento a un alto número de imagenes por segundo, incluso en exteriores, además de una mayor sencillez a la hora de identificar cada marcador.

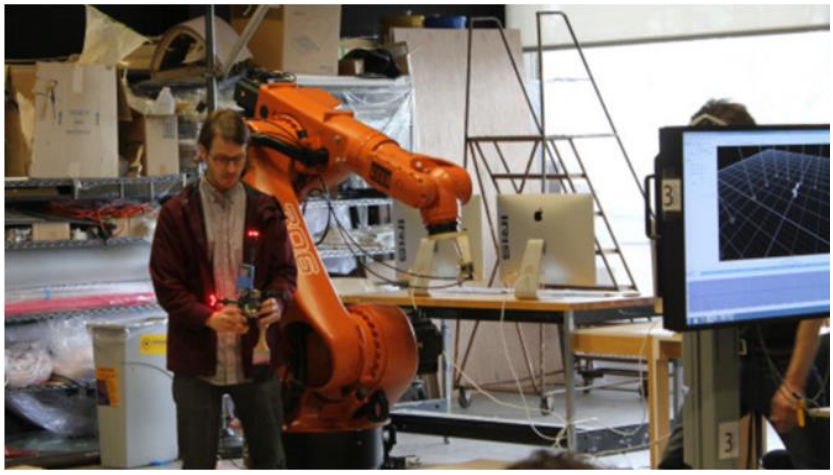

Figura 1-3. Captura de movimiento con marcadores activos.

Esta empresa, a diferencia de la anterior, ha optado por proveer de una serie de librerías compatibles con Python, C/C++ y Javascript, en vez de utilizar un software propio, facilitando la integración con los lenguajes de programación más populares.

Si nos centramos en el seguimiento de brazos manipuladores, existen numerosos métodos para realizar dicha tarea. El más común es el uso de marcadores repartidos por toda la estructura como se ha visto mas arriba, sin embargo, no es la única forma. Un artículo publicado por la Universidad de Carnegie Mellon, en Pittsburgh, Pensilvania [3], hace uso de una red neuronal entrenada para detector y reproyectar en la imagen los puntos característicos de la articulación, en este caso las uniones entre las articulaciones, que añadido a los datos que aportan los encoders se calculan los parámetros intrínsecos del brazo y, mediante un algoritmo de Perspectiva-n-puntos, se obtienen los parámetros extrínsecos del mismo referidos a la cámara externa fija.

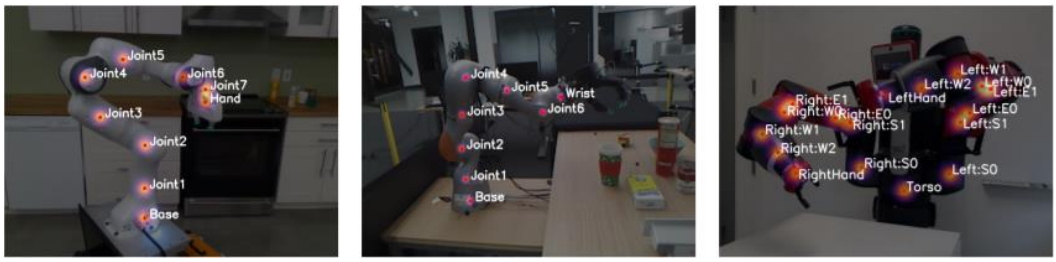

Figura 1-4. Reproyección de los puntos característicos de la articulación.

Este Proyecto tiene como objetivo presentar un método mucho mas simple y de menor coste para obtener unos resultados similares a lo visto anteriormente. Mediante el uso de marcadores óptico-pasivos distribuidos por la articulación, se obtienen las coordenadas tridimensionales respecto de una cámara fija utilizando algoritmos de triangulación entre las tres cámaras infrarrojas disponibles. Una vez conocidas dichas coordenadas, se pretende resolver las relaciones no lineales entre dichas coordenadas con los parámetros de posición y orientación de la articulación, independientemete de la posición y orientación del mismo respecto del sistema de cámaras. Cabe destacar que es necesario tener calibrado tanto intrínseca como extrínsecamente el sistema de captura, algo que se explica más adelante en esta memoria.

1.2. Organización del documento

Esta memoria se divide en varias secciones de contenido. En primer lugar, se presenta una más o menos extensa explicación de una serie de conceptos aplicados que conviene repasar antes de entrar en materia. Seguidamente, se presenta en detalle todo el hardware y software empleado. Siguiendo, nos encontramos con un apartado dedicado íntegramente a detallar matemáticamente los modelos de los sistemas articulados utilizados en los experimentos, ordenadas de menor a mayor complejidad. El siguiente apartado se centra en explicar todos los detalles de la solución adoptada para resolver el problema de la estimación de posición y orientación de la articulación, así como algunas secciones del código empleado. A continuación, se dedica un apartado a mostrar los resultados obtenidos en los diferentes experimentos con distintas articulaciones, así como a analizar los resultados obtenidos, para acabar finalmente con el último apartado de conclusiones y desarrollos futuros, en el que se recapitulará todo lo expresado anteriormente, junto con una pequeña descripciónn de posibles mejoras del sistema, limitaciones de este, así como los siguientes pasos a realizar.

2 CONCEPTOS PREVIOS

Antes de proseguir con la lectura de la memoria, es conveniente repasar algunos conceptos que sirven como base para el desarrollo de este trabajo. Puesto que se trabaja con un sistema multicámara, es necesario conocer qué son tanto los parámetros instrínsecos como extrínsecos de las mismas, así como la notación que sigue. También conviene repasar en que consiste el problema de la Perspectiva de n-Puntos, utilizado más adelante, así como también en qué consiste la Triangulación de puntos, y finalmente la optimización mediante mínimos cuadrados.

2.1. Modelo de cámara. Parámetros intrínsecos

OpenCV [4] es una librería muy versátil dedicada al procesamiento de imágenes, que cuenta con un gran número de funciones de alto nivel para todo tipo tareas. Dichas funciones se basan en el modelo estenopeico de una cámara, o más conocido como *pinhole*. La vista de la escena se obtiene proyectando los puntos 3D en el plano de la imagen mediante una transformación de perspectiva de la que se obtienen unas coordenadas 2D en píxeles.

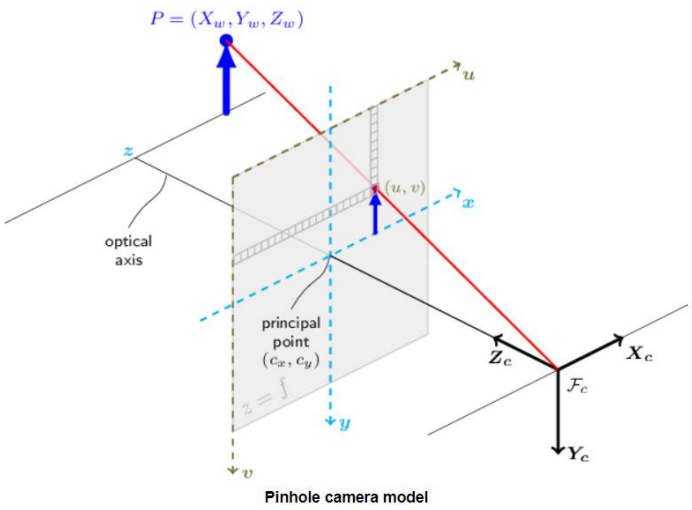

Figura 2-1. Modelo de cámara estenopeica.

De esta forma, la proyección de coordenadas 3D a 2D en la imagen se realiza de la siguiente forma:

$$\tilde{p} = K * \tilde{P_c} \tag{2–1}$$

Donde $\tilde{p}$ corresponde con las coordenadas en pixeles del punto en coordenadas homogéneas, K la matriz intrínseca de la cámara y $\tilde{P_c}$ las coordenadas en 3D del mismo punto, de nuevo en coordenadas homogéneas, visto desde los ejes de referencia de la cámara F_c que aparece en la Figura 2-1.

La matríz de parámetros intrínsecos K está compuesta por las logitudes focales efectivas f_x y f_y, expresades en píxeles, y las coordenadas del punto principal de la imagen (c_X, c_y) normalmente cercanas al centro de la misma.

$$K = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2–2)$$

Esta matriz no depende de la escena en cuestión, ergo una vez calculada, puede usarse cuando que se requiera siempre y cuando no se haya variado la distancia focal de la lente.

2.2. Transformaciones homogéneas

Si bien se ha visto que, obtenidas las coordenadas tridimensionales de un punto con respecto a la cámara en cuestión, es directo obtener la proyección de dichos puntos en la imagen, en la gran mayoría de casos uno no dispone a priori de dichas coordenadas en dicho marco de referencia, sino en otro distinto. Es por ello por lo que es necesario realizar una tranformación de coordenadas homogéneas de un sistema de referencia a otro.

$$\widetilde{P_c} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} * \widetilde{P_w} \quad (2–3)$$

La transformación homogénea queda predeterminada por los parámetros extrínsecos *R* y *t*, que representan un cambio de ejes de coordenadas del mundo *w* a la cámara *c*. Dicha matriz de transformación queda de la forma:

$$T_w^c = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} r11 & r12 & r13 & tx \\ r21 & r22 & r23 & ty \\ r31 & r32 & r33 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2–4)$$

R es una matríz de tamaño 3x3 que corresponde con una o varias rotaciones en cadena, y *t* es un vector columna de tres componentes, que corresponden con las coordenadas del actual marco de referencia visto desde el nuevo. En la notación T_w^c, el superíndice *c* indica el eje de referencia objetivo de la transformación, mientras que el subíndice *w* indica que se parte de dicho eje de coordenadas.

2.3. Problema de la Perspectiva de n-Puntos (PnP)

El problema de la Perspectiva de n-Puntos [5] consiste en estimar la pose de una cámara calibrada intrínsecamente a partir de una serie de puntos tridimensionales de coordenadas conocidas, junto con sus respectivas proyecciones en la imagen. En principio, se necesitan de un mínimo de tres puntos para poder resolverse [6], sin embargo, un mayor número de puntos acarrea una mayor precisión. La cámara posee 6 grados de libertad referidos a posición y orientación, donde la posición viene determinada por sus coordenadas cartesianas, y la orientación como un conjunto de tres rotaciones cartesianas. Este problema proviene de la dificultad de calibrar extrínsecamente una cámara externa, y tiene numerosas aplicaciones en el campo de la Robótica y Automatización.

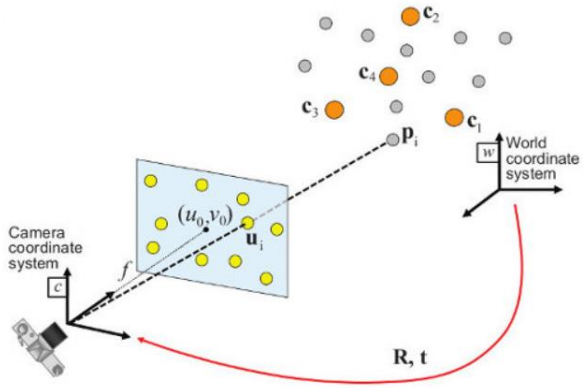

Figura 2-2. Problema de la Perspectiva de N-puntos.

El problema consiste en lo siguiente:

Dado un conjunto de *n* puntos tridimensionales y sus correspondientes proyecciones 2D sobre una imagen dada por una cámara previamente calibrada instrínsecamente, se requiere determinar los 6 grados de libertad (orientación y posición) que posee dicha cámara respecto de un eje de referencia del "mundo".

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r11 & r12 & r13 & t_x \\ r21 & r22 & r23 & t_y \\ r31 & r32 & r33 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2–5}$$

Esta ecuación de proyección se repite para cada punto 3D y su proyección en la imagen, y se resuelve buscando la matriz de rotación y traslación que cumpla con todas ellas. Es por ello por lo que se necesita de un mínimo de 3 puntos para tener un sistema de ecuaciones determinado. No existe un único método de resolución del problema, si bien depende de cada implementación, sin embargo el más básico y que requiere de 3 puntos tridimensionales (*P3P*) es resuelto hoy en día mediante algoritmos modernos como el que se propone en [6].

Este algoritmo, denominado *Lambda Twist* trata de resolver el sistema de ecuaciones cuadrático no homogéneo (2-5) desarrollado como:

$$\lambda_i y_i = Rx_i + t, i \in \{1,2,3\} \tag{2–6}$$

Dependiendo de la configuración de los puntos, este sistema puede tener hasta 4 posibles soluciones. Por supuesto para que exista una solución única; ni los puntos 3D ni los 2D pueden ser colineales, y todos ellos deben estar presentes delante de la cámara, además de que la solución debe ser real.

El pseudo código que implementa este algoritmo se muestra a continuación:

Código 2-1. Lambda Twist P3P (1)

1: function MIX (n, m) = (n, m, n ×m)
2: function P3P ($y_{1:3}, x_{1:3}$)
3: Normalizar $y_i = y_i/|y_i|$
4: $a_{ij} = \lambda_j^2 + \lambda_i^2 - 2b_{ij}$, $b_{ij} \stackrel{\text{def}}{=} y_i^T y_i$
5: $D_1 = (d_{11}, d_{12}, d_{13})$, $D_2 = (d_{21}, d_{22}, d_{23})$
6: Computar $c_3\gamma^3 + c_2\gamma^2 + c_1\gamma + c_o$ y obtener γ
7: $D_0 = D_1 + \gamma D_2$
8: $[E, \sigma_1, \sigma_2]$ = EIG3X3KNOWN0 (D0).
9: $s = \pm\sqrt{-\sigma 2/\sigma 1}$
10: Computar $\lambda_2^2(w_0 + \tau w_1; 1; \tau)^T + D_1(w_0 + \tau w_1; 1; \tau) = 0$ y obtener τ
11: Computar $\Lambda^T M_{23}\Lambda^T = a_{12} == \lambda_{2k}^2(1; \tau_k)^T(1, -b_{23}; -b_{23}, 1)(1; \tau_k) = a_{23}$ y obtener Λ_k
12: $X_{inv} = \left(mix(x1 - x2, x1 - x3)\right)^{-1}$
13: por cada Λ_k válido
14: Gauss-Newton-Refine(Λ_k)
15: $Yk = MIX(\lambda_{1k}y_1 - \lambda_{2k}y_2, \lambda_{1k}y_1 - \lambda_{3k}y_3)$
16: $R_k = Y_k X_{inv}$
17: $t_k = \lambda_{1k}y_1 - R_k x_1$
18: Return all Rk, tk

Código 2-2. Lambda Twist P3P (2)

1: function GETEIGVECTOR(m, r)
2: $c = (r^2 + m_1m_5 - r(m_1 + m_5) - m_2^2)$
3: $a_1 = (rm_3 + m_2m_6 - m_3m_5)/c$
4: $a_2 = (rm_6 + m_2m_3 - m_1m_6)/c$
5: $v = (a_1; a_2; 1)$
6: Return $v/|v|$
7: function EIG3X3KNOWN0(M)
8: $b_3 = M(:,2)xM(:,3)$
9: $b_3 = b_3/|b_3|$
10: m = M(:)
11: $p_1 = m_1 - m_5 - m_9$
12: $p_0 = -m_2^2 - m_3^2 - m_6^2 + m_1m_5 + m_9 + m_5m_9$
13: $[\sigma_1, \sigma_2]$ como las raíces de $\sigma^2 + p_1\sigma + p_0 = 0$
14: b1 = GETEIGVECTOR(m, σ_1)
15: b2 = GETEIGVECTOR(m, σ_2)
16: if $|\sigma_1| > |\sigma_2|$ then
17: Return $([b_1, b_2, b_3], \sigma_1, \sigma_2)$
18: else
19: Return $([b_2, b_1, b_3], \sigma_2, \sigma_3)$

2.4. Triangulación de puntos

En visión por computador, la triangulación [7] es el proceso por el cual se determinan las coordenadas 3D de un punto en el espacio dadas sus coordenadas 2D proyectadas en dos o más imágenes, suponiendo que se trata de un sistema totalmente calibrado.

La geometría que relaciona las cámaras, los puntos tridimensionales y sus proyecciones se denomina geometría epipolar [8].

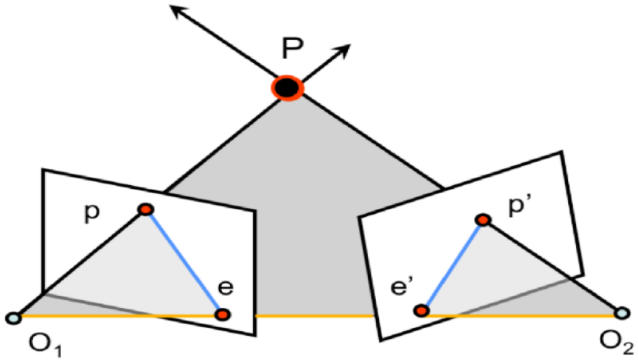

Figura 2-3. Geometría epipolar.

Esta configuración especifica que dos o más camaras están enfocando al mismo punto *P* en el espacio, cuya proyección en cada cámara se denomina *p* y *p'* respectivamente. Siguiendo el ejemplo de la Figura 2-3, el plano definido por los centros de las cámaras O_1 y O_2, y el punto *P*, se denomina plano epipolar. Los puntos donde se cortan los planos de las imágenes con la línea que une los centros de las cámaras se denominan epipolos. En la teoría, la intersección entre las líneas que unen $\overline{O_1 p}$ y $\overline{O_2 p'}$ da como resultado el punto tridimensional buscado, *P*.

Sin embargo, en la práctica no es directo, puesto que las lentes de las cámaras no son perfectas, y presentan distorsión en las imágenes, lo que provoca que dichas líneas proyectadas no intersecciones. Como consecuencia, es necesario estimar en que punto de corte se obtienen las mejores medidiciones y los errores de reproyección más pequeños.

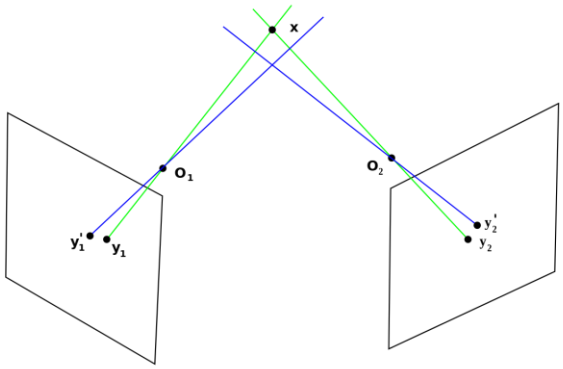

Figura 2-4. Geometría epipolar. Caso real.

Como consecuencia, las proyecciones del punto tridimensional *x* presentado en la Figura 2-4 no son capturadas como y_1 e y_2, sino como y_1', y_2'. A la vista de la imágene, las líneas azules, correspondientes con las líneas de proyección no interseccionan exactamente en el punto x, sino que lo hacen en un punto cercano (siempre y cuando satisfagan las restricciones epipolares) x_{est}.

Por tanto, la triangulación, en el ámbito de la visión por computador, trata de encontrar la mejor estimación x_{est} del punto 3D *x* dadas las proyecciones y_1', y_2' y las matríces que describen a las cámaras C_1, C_2, que puede ser descrita en términos de una función como:

$$x \sim \tau(y_1', y_2', C_1, C_2) \tag{2–7}$$

El símbolo ~ impica que la función solo requiere de determinar un vector que sea igual que *x* al multiplicarlo por un escalar, puesto que se tratan de coordenadas homogéneas.

Es posible que algunos métodos fallen a la hora de determinar una estimación de *x* si este se encuentra en cierto subconjunto del espacio tridimensional, correspondiente a cierta combinación de y_1', y_2', C_1, C_2. Un punto contenido en dicho espacio se conoce como singularidad, y la razón para que esta falle puede ser porque el sistema de ecuaciones es indeterminado, o el resultado x_{est} es un vector nulo.

En el caso de OpenCV, la función *triangulatePoints()* [9] hace uso internamente del algoritmo de Transformación Lineal Directa (DLT) [10]. Este se entiende mejor mediante un ejemplo:

Suponiendo que $k \in \{1, \dots, N\}$, tenemos $x_k = (x_{1k}, x_{2k}) \in \mathbb{R}^2$ e $y_k = (y_{1k}, y_{2k}, y_{3k}) \in \mathbb{R}^3$, vectores conocidos de los que se quiere conocer cierta relación del tipo:

$$\alpha_k x_k = A y_k \tag{2–8}$$

Donde $\alpha_k \neq 0$, escalar desconocido relacionado con la ecuación *k*.

Para obtener unas ecuaciones homogéneas, y deshacernos de los escalares desconocidos, se define una matríz antisimétrica del tipo $H = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$, y se multiplica a ambos lados de la ecuación (2-8):

$$(x_k^T H)\alpha_k x_k = (x_k^T H) A y_k \tag{2–9}$$

Dado que $x_k^T H x_k = 0$, (2-9) queda reescrito como.

$$x_k^T H A y_k = 0 \tag{2–10}$$

Si consideramos los elementos de cada variable; $x_k = \begin{bmatrix} x_{1k} \\ x_{2k} \end{bmatrix}, y_k = \begin{bmatrix} y_{1k} \\ y_{2k} \\ y_{3k} \end{bmatrix}, A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$, podemos reordenar la ecuación anterior, una vez desarrollado elemento a elemento los productos matriciales, como:

$$Ba = 0 \tag{2–11}$$

Con $B = \begin{bmatrix} x_{2k}y_{1k} \\ -x_{1k}y_{1k} \\ x_{2k}y_{2k} \\ -x_{1k}y_{2k} \\ x_{2k}y_{3k} \\ -x_{1k}y_{3k} \end{bmatrix}$ y $a = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{12} \\ a_{22} \\ a_{13} \\ a_{23} \end{bmatrix}$. En la práctica, los vectores x_k, y_k suelen contener componentes de ruido. Como consecuencia, puede no haber una solución exacta para *a*, por lo que la resolución se realiza mediante mínimos cuadrados, minimizando el error cuadrático de (2-11).

2.5. Mínimos cuadrados no lineales

Los mínimos cuadrados no lineales es una forma de análisis de mínimos cuadrados que se usa para resolver un conjunto de *m* funciones con un modelo que es no lineal con *n* parámetros o incógnitas (*m*>*n*). La base del método es aproximar el modelo no lineal por uno lineal, y mediante un proceso iterativo encontrar una solución para los parámetros del sistema.

Aunque existen varios algoritmos que se usan para resolver dicho problema de optimización, en esta sección se va a estudiar en profundida el metodo de *Levenberg* [11]-*Marquardt* [12], que es el que implementa la librería *Ceres* [13], de código abierto, y que está optimizada para este tipo de operaciones matemáticas.

Se procede ahora a explicar como implementa *Ceres* dicho algoritmo:

Sea $x \in \mathbb{R}^n$ un vector de *n* variables y $F(x) = [f_1(x), \dots, f_m(x)]$ una función *m*-dimensional que depende de *x*, nos interesa resolver el problema de optimización:

$$arg\min_x \frac{1}{2}\|F(x)\|^2 \qquad (2\text{–}12)$$
$$L \leq x \leq U$$

Donde, *L* y *U* corresponden con los límites inferiores y superiores del vector de parámetros *x*.

Dado que la minimización global eficiente de (2-3) para *F(x)* es un problema sin solución, tendremos que conformarnos con encontrar un mínimo local. A continuación, el jacobiano *J(x)* de *F(x)* es una matríz *m x n* en donde $J_{ij}(x) = \partial_j f_i(x)$ y el vector gradiente es $g(x) = \nabla \frac{1}{2}\|F(x)\|^2 = J(x)^T F(x)$.

La estrategia general para resolver problemas no lineales por mínimos cuadrados es realizar una serie de aproximaciones en las que en cada iteración se determina un incremento que se aproxima linealizando como $F(x + \Delta x) \approx F(x) + J(x)\Delta x$, por lo que ahora (2-3) pasa a escribirse como:

$$arg\min_x \frac{1}{2}\|F(x) + J(x)\Delta x\|^2 \qquad (2\text{–}13)$$

Sin embargo, es necesario controlar cada incremento de cada nueva iteración, para asegurar que el problema converja en una solución. Uno de los algoritmos más conocidos y del que hace uso el método de *Levenberg-Marquardt* se denomina *Trust Region*.

Este algoritmo intenta aproximar la función objetivo utilizando una función modelo, normalmente cuadrática, sobre un subconjunto del espacio de búsqueda llamado región de confianza. Si la función del modelo logra

minimizar la verdadera función objetivo, la región de confianza se expande; por el contrario, se contrae y se vuelve a intentar solucionar el problema de optimización del modelo.

El algoritmo de *Levenberg-Marquardt* fue en su momento el primero en desarrollarse utilizando el método de la región de confianza, y es a día de hoy uno de los más populares.

Puede demostrarse que la solución para (2-6) puede resolverse mediante una optimización sin restricciones del tipo:

$$arg\min_{\Delta x}\frac{1}{2}\|F(x)+J(x)\Delta x\|^2+\lambda\|D(x)\Delta x\|^2 \tag{2–14}$$

λ es un multplicador de Lagrange que está inversamente relacionado con el radio de la región de confianza μ, por lo que se sustitye directamten por la inversa de dicho radio. La matriz D(x) es una matriz diagonal que corresponde con la raiz cuadrada de los valores en la diagonal resultante de $J(x)^TJ(x)$.

Para simplificar, se asumirá que la matriz $\frac{1}{\sqrt{\mu}}D(x)$ se ha concatenado al final de la matriz *J(x)*, y de manera similar, un vector de ceros se ha añadido al fondo de la matriz *f(x)*, de modo que en lo que queda de apartado solo se tratará con el jacobiano y la función en sí.

$$\min_{\Delta x}\frac{1}{2}\|F(x)+J(x)\Delta x\|^2 \tag{2–15}$$

Para toda la resolución del problema, (2-6) será la mas costosa computacionalmente hablando en cada iteración del algoritmo. *Ceres* provee dos métodos distintos de resolución, factorizando o iterando.

Centrándonos en el iterativo, hacen falta dos pasos importantes. El primero se trata de resolver sistemas de ecuaciones lineales. Ceres hace uso de un método de bajo coste computacional para resolver los sistemas de ecuaciones una vez linealizados, basado en los Gradientes Conjugados [14]. El segundo y ultimo paso es que se requiere de una función de terminación para el proceso iterativo, de modo que el algoritmo no funcione indefinidamente, y pare cuando se ha obtenido un resultado acceptable.

Una función de terminación típica, y que *Ceres* utiliza es:

$$\|H(x)\Delta x+g(x)\|\leq\eta_k\|g(x)\| \tag{2–16}$$

En (2-7), *k* indica el número de la iteración, y el parámetro $0<\eta_k<1$ se conoce como parámetro de forzado de secuencia (del inglés, *forcing sequence*). Está demostrado que cualquier algoritmo de *Levenberg-Marquardt* que utilice cualquiera de los dos métodos de control de los incrementos Δx del problema de mínimos cuadrados no lineales converge para una secuencia $\eta_k\leq\eta_0<1$, y que cuyo ratio de convergencia depende del valor de η_k.

3 ESTACIÓN DE TRABAJO Y HERRAMIENTAS

El desarrollo del trabajo se ha llevado a cabo principalmente en los laboratorios de la escuela, donde se disponía de un cubículo de 3x3x3 metros con cuatro cámaras infrarrojas en posiciones enfrentadas dos a dos, conectados a un ordenador central desde el que se contaba con la instalación del software propietario de la marca *Teledyne DALSA* [15] con el que hacer uso de las cámaras.

3.1. Plataforma de captura de imágenes

La estructura está formada por una serie de vigas de acero, formando un cubo de 3 metros de ancho, alto y largo. Esto se traduce en un soporte fuerte en el que las cámaras se puedan colocar con seguridad, y que asegura que las mismas permanezcan inmóviles durante su funcionamiento.

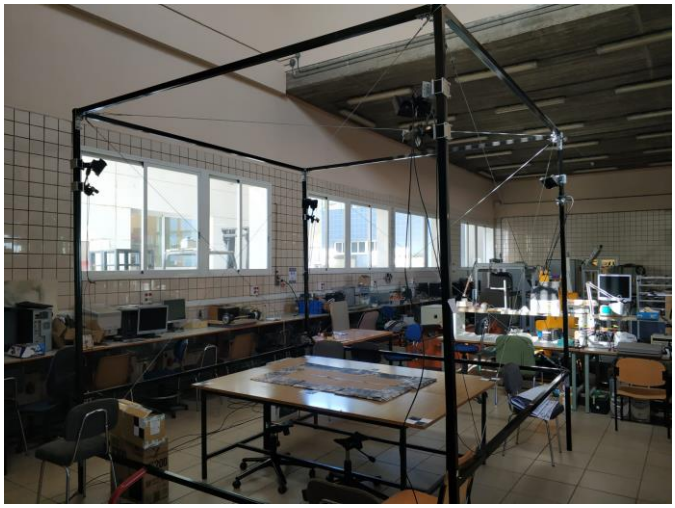

Figura 3-1. Cubículo de pruebas.

Como puede observarse en la figura de arriba, dentro se dispone de una mesa de grandes dimensiones, facilitando la realización de las pruebas.

3.1.1 Cámaras empleadas

El cubículo dispone de 4 cámaras *Genie Nano M1280-NIR*. Por limitaciones en el hardware disponibe, solo era posible utilizar tres de ellas a la vez. A continuación, se muestran sus especificaciones:

Tabla 3–1. Parámetros de las cámaras utilizadas

Parámetro	Valor
Resolución	1280x1024 píxeles
Distancia focal	4.5mm
Tamaño del sensor	6.4x4.8mm
Sensor	CMOS
Dimensiones	29x44x21mm
Peso	46g

Figura 3-2. Cámara y foco infrarrojo empleados.

3.1.2 Marcadores

El objetivo de utilizar cámaras y focos infrarrojos es que se pueda hacer seguimiento de objetos con material retrorreflectante, un material que es capaz de devolver cualquier rayo de luz incidente, el mismo que se utiliza en los chalecos reflectantes o en las señales de tráfico. Debido al bajo coste del mismo y a su sencilla apliación, se optó por utilizar pelotas de diferentes tamaños recubiertos con este material, adheridos a la superficie del objeto en cuestión.

Con el objetivo de facilitar la identificación y asociación de los mismos, se han dispuesto de mayor a menor tamaño, empezando por la base del brazo, hasta el efector final del mismo.

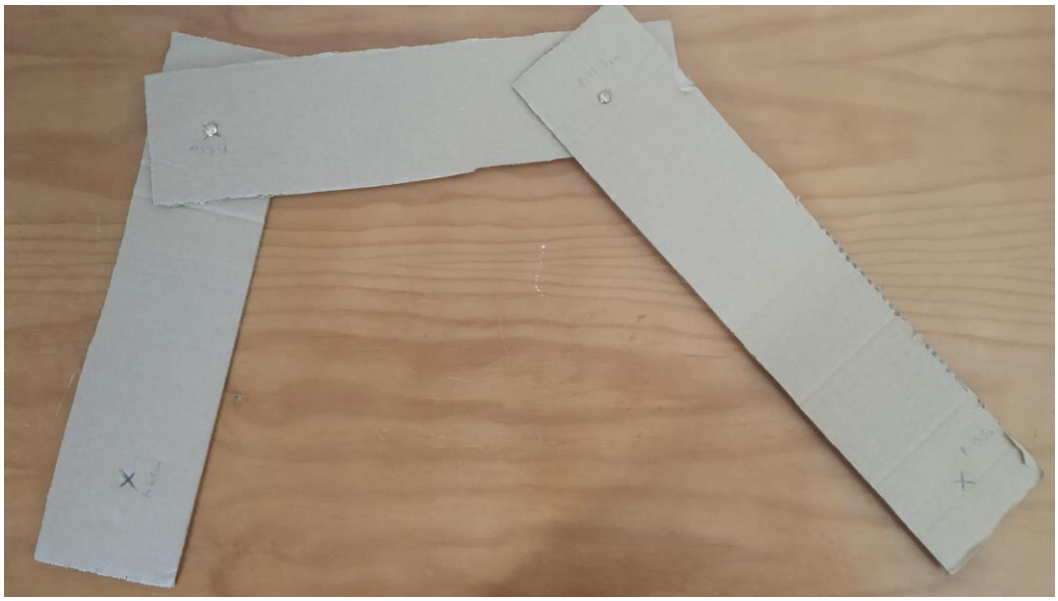

Figura 3-3. Modelo de articulación utilizado para los experimentos.

Figura 3-4. Marcador recubierto con material retrorreflectante.

3.2. Ordenador de pruebas

Debido a que, usando las funciones propias del fabricante para leer imágenes en tiempo real del búfer de las cámaras con OpenCV se obtenía una tasa de imágenes por segundo demasiado baja, se concluyó que trabajar con imágenes estáticas individuales no iba a suponer una gran diferencia en los resultados finales, además de que el desarrollo del código se ha llevado a cabo de manera que su adaptación en el ordenador de los laboratorios fuera lo más sencilla posible. Cabe mencionar que, al utilizar imágenes estáticas, no se hizo hincapié en la sincronización de las cámaras, algo para tener en cuenta si en un futuro se pretende realizar en tiempo real a una tasa de imágenes por segundo alta.

Es por ello por lo que el procesado de las imágenes se llevo a cabo desde un ordenador distinto, con las siguientes características:

Tabla 3–2. Especificaciones del ordenador de pruebas

Parámetro	Valor
CPU	Intel Core i7-7700HQ
Frecuencia CPU	2.8-3.8 GHz
GPU	NVIDIA GTX 1050 4 GB
RAM	16 GB

3.3. Software empleado

3.3.1 Visual Studio Community 2019: C++

El SDK propietario de *Teledyne DALSA* para el funcionamiento de las cámaras instalado en el ordenador del laboratorio esta desarrollado en C++. Como consecuencia, se decidió utilizar este lenguaje junto con *Visual Studio Community* para llevar a cabo este proyecto. Además, por se un lenguaje compilado, se presenta como una de las mejores alternativas para su uso en tiempo real, si se hubiese llegado al caso.

Por otro lado, esta plataforma posee soporte oficial para todas las librerías que han sido de utilidad para la realización del trabajo, como por ejemplo *OpenCV* [16], *Ceres* [13] o *Eigen* [17]. Esta última dedicada a facilitar toda clase de operaciones matriciales, agilizando el proceso y realizando dichas operaciones a una velocidad superior que *OpenCV*.

3.3.2 MATLAB R2017B

MATLAB [18] es un programa ampliamente conocido en la escuela. Por su agilidad y facilidad de uso, se presenta como una alternativa muy conveniente para realizar comprobaciones rápidas de ciertos aspectos del proyecto, así como representaciones gráficas de los resultados obtenidos. *MATLAB* también posee un toolbox dedicado a la calibración intrínseca de cámaras, entre otras tareas, que nos ha sido de gran utilidad. Se trata del *Computer Vision Toolbox.*

4 MODELO DE ARTICULACIÓN MÓVIL

Una cadena cinemática [19] se define como un sistema formado por varios cuerpos conectados y con movimientos relativos entre ellos. Existen 2 tipos, según el número de pares cinemáticos por el que son unidos:

- Cadena cinemática abierta: Los eslabones están unidos por sólo un único par cinemático.
- Cadena cinemática cerrada: Los eslaboles están unidos por mas de ún par cinemático.

Una cadena cinemática con al menos uno de los cuerpos sin movimiento, se denomina mecanismo. De este modo, en este trabajo se va a tratar solo con mecanismos de cinemática abierta y articulaciones fijas unidas mediante pares rotativos o prismáticos, que son los que encontramos en la gran mayoría de robots manipuladores. Cada uno de los movimientos independientes que puede realizar cada articulación con respecto al anterior, se denomina grado de libertad (GDL).

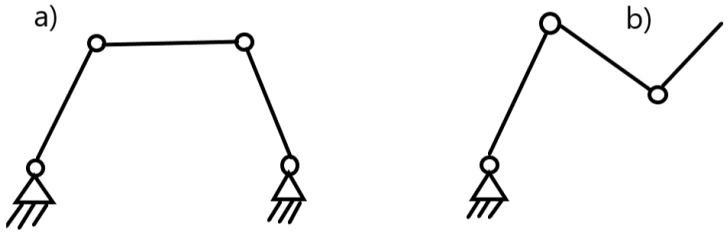

Figura 4-1. Cadena cinemática cerrada a) y abierta b).

La cinemática del robot [20] estudia el movimiento del mismo con respecto a unos ejes de referencia sin considerar las fuerzas que intervienen. Se trata pues, de un análisis descriptivo del movimiento espacial del robot conforme el tiempo, en particular para describir la posición y orientación del efector final del robot. Es por ello por lo que, si se quiere describir con exactitud un sistema articulado, tanto en posición como orientación, independientemente del método, será necesario tener previamente descrita la cinemática de este.

Existen dos problemas fundamentales a resolver en la cinemática; el primero de ellos se conoce como la cinemática directa, que trata de determinar la posición y orientación de sus eslabones en función de los parámetros geométricos y articulaciones del robot, respecto de un sistema de referencia normalmente solidario a la base del mismo; el segundo, denominado problema de la cinemática inversa, y como su nombre indica, define lo inverso al problema cinemático directo, trata de obtener los valores de las articulaciones en función de las posiciones conocidas de los eslabones.

Para este trabajo, se han empleado las ecuaciones cinemáticas directas de las articulaciones, dado que permiten relacionar las tres ecuaciones de posición (una por cada componente cartesiana) con los mismos parámetros de las articulaciones, facilitando la resolución final.

$$x = f(q_1, q_2, \dots, q_n) \tag{4–1}$$
$$y = f(q_1, q_2, \dots, q_n)$$
$$z = f(q_1, q_2, \dots, q_n)$$

La obtención de dichas ecuaciones puede realizarse de diferentes maneras; mediante métodos geométricos, lo cual es válido para casos simples y de pocos grados de libertad; mediante matrices de transformaciones homogéneas, similar a lo que ocurre con el método anterior, aunque en este caso es quizás mas sencillo de aplicar para articulaciones un poco mas complejas. Finalmente, existe un procedimiento general para cualquier articulación, el algoritmo de *Denavit-Hartenberg* [21].

4.1. Algoritmo de Denavit-Hartenberg

En 1955, Denavit y Hartenberg propusieron un método matricial que establece la localización que debe tomar cada sistema de coordenadas S_i ligado a cada eslabón *i* de una articulación.

Escogiendo los ejes de coordenadas según este método, será possible pasar de una articulación a otra mediante cuatro transformaciones básicas que dependen de características geométricas del eslabón en sí.

Dichas transformaciones (referidas al sistema móvil) son:

1. Rotación alrededor del eje z_{i-1} un ángulo θ_i.
2. Traslación a lo largo del eje z_{i-1} una distancia d_i
3. Traslación a lo largo de x_i una distancia a_i.
4. Rotación alrededor del eje x_i un ángulo α_i.

De este modo se obtiene que:

$$A_i^{i-1} = Rotz(\theta_i) * T(0,0,d_i) * T(a_i,0,0) * Rotx(\alpha_i) \tag{4–2}$$

Y desarrollando los productos matriciales:

$$A_i^{i-1} = \begin{bmatrix} C\theta_i & -S\theta_i & 0 & 0 \\ S\theta_i & C\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_i & -S\alpha_i & 0 \\ 0 & S\alpha_i & C\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4–3}$$

$$= \begin{bmatrix} C\theta_i & -C\alpha_i S\theta_i & S\alpha_i S\theta_i & a_i C\theta_i \\ S\theta_i & C\alpha_i C\theta_i & -S\alpha_i C\theta_i & a_i S\theta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Así pues, para que la matriz A_i^{i-1} tenga sentido, es necesario que los sistemas se hayan escogido según las siguientes normas [22]:

1. Numerar los eslabones comenzando por el 1. El eslabón 0 corresponderá con la base fija del robot.
2. Numerar cada articulación del robot empezando por la 1, y acabando en *n*.
3. Localizar el eje principal de cada articulación. Si es rotativa, será su propio eje de rotación. Si es prismática, será el eje a lo largo del cual se produce el desplazamiento.

4. Para *i* de 1 a *n-1* situar el eje z_i sobre el eje de articulación *i+1*.
5. Situar el origen del sistema S_0 sobre cualquier punto del eje z_0. El resto de ejes del sistema S_0 son de libre disposición, siempre y cuando formen un sistema dextrógiro.
6. Para *i* de 1 a *n-1* situar el origen del sistema S_i en la intersección del eje z_i con la línea normal común a z_{i-1} y z_i. Si ambos ejes son paralelos, se debe situar en la articulación *i+1*.
7. Situar x_i en la línea normal común a a z_{i-1} y z_i.
8. Situar a y_i de manera que forme un sistema dextrógiro con a z_i y x_i.
9. Situar el sistema S_n en el extremo del robot de modo que z_n coincida con la dirección de z_{n-1} y x_n sea normal a a z_{n-1} y z_n.
10. Obtener θ_i como el ángulo que hay que girar en torno a a z_{i-1} para que a x_{i-1} y x_i queden paralelos.
11. Obtener d_i como la distancia, medida a lo largo del eje z_{i-1} que habría que desplazar S_{i-1} para que x_{i-1} y x_i queden alineados.
12. Obtener a_i como la distancia medida a lo largo de x_i que habría que desplazar el nuevo S_{i-1} para que su origen coincidiese con S_i.
13. Obtener α_i como el ángulo que habría que girar en torno a x_i para que el nuevo S_{i-1} coincidiese totalemte con S_i.

Siguiendo estos pasos, ya Podemos obtener A_i^{i-1} tal y como se especifica en (4-3). La matriz de transformación que relaciona la base con el resto de los eslabones no es más que el producto de estas matrices.

4.2. Articulación plana de 2 grados de Libertad

El primero de los experimentos se ha realizado utilizando un modelo de articulación móvil simple. Se trata de un modelo de articulación plana de dos grados de libertad, que consta de 2 articulaciones rotatorias.

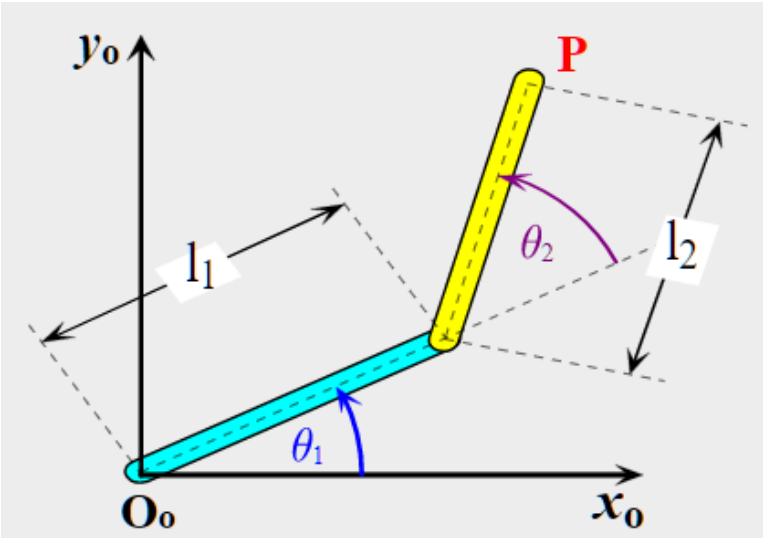

Figura 4-2. Articulación plana de 2 gdl.

Sin embargo, para agilizar aún mas el problema, si transformamos los ejes de referencia representados en la Figura 4-1 a otros nuevos, cuyo eje *OZ* quede paralelo y solidario a la barra *L1*, pasamos a tener una

articulación de tan solo un grado de Libertad, puesto que θ_1 ya no nos aporta ningún tipo de información a la hora de obtener las coordenadas de cada punto de la articulación respecto de estos nuevos ejes.

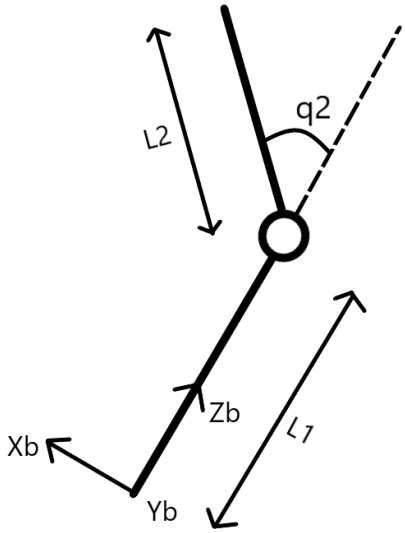

Figura 4-3. Articulación plana de 2 gdl con ejes nuevos.

Este brazo servirá como base para el resto de los experimentos. A continuación, se detalla para este caso como quedan las ecuaciones cinemáticas.

4.2.1 Cinemática Directa

Siguiendo los pasos del algoritmo de *Denavit-Hartenberg* para una articulación plana de 2 grados de libertad, y atendiendo a las simplificaciones explicadas anteriormente, los parámetros de *Denavit-Hartenberg* son los siguientes:

Tabla 4–1. Parámetros de DH para articulación de 2 gdl.

Articulación i	θ_i	d_i	a_i	α_i
1	q2	0	L2	0

Dado que con los nuevos ejes XYZ_B el brazo L_1 queda inmóvil, solo nos queda una articulación relativa en el sistema. De nuevo, conviene recordar que dado que en estos nuevos ejes base el eje OZ_B se sitúa solidario a la primera articulación, dicha articulación se comporta ahora como fija, reduciendo el número de incógnitas. Es por ello que al calcular los parámetros de DH esto se ha tenido en cuenta, y solo aparecen datos de la segunda.

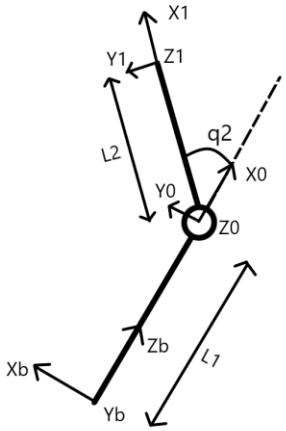

Figura 4-4. Articulación plana de 2 gdl con ejes DH.

Desarrollando (4-3), tenemos entonces que:

$$A_1^0 = \begin{bmatrix} C\theta_2 & -S\theta_2 & 0 & L_2C\theta_2 \\ S\theta_2 & C\theta_2 & 0 & L_2S\theta_2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4–4}$$

Sin embargo, como se ve en la Figura 4-3, la matriz representada en (4-4) solo corresponde con el paso de los ejes XYZ_0 con XYZ_1, es decir, hasta el efector final. Si queremos que queden expresados en función de los ejes XYZ_B hay que premultiplicar (4-4) por lo siguiente:

$$A_0^B = Rotx\left(\frac{-\pi}{2}\right) * Rotz\left(\frac{-\pi}{2}\right) * T(L_1, 0,0) \tag{4–5}$$

Que no es mas que la matriz de transformación homogénea para pasar de los ejes XYZ_B a los ejes XYZ_0.

De esta forma, multiplicado ambas matrices nos queda que:

$$A_1^B = A_0^B * A_1^0 \tag{4–6}$$

Esta es la matriz de transformación que gobierna el movimiento del efector final del brazo, respecto de los ejes fijos XYZ_B. Finalmente, las ecuaciones cinemáticas que gobiernan el efector final son:

$$P_1^B = \begin{bmatrix} L2 * sin\,(\theta_2) \\ 0 \\ L_1 + L_2 * cos\,(\theta_2) \end{bmatrix} \tag{4–7}$$

Por otro lado, es trivial definir las coordenadas de la base del brazo como de la articulación rotatoria como:

$$P_{base}^B = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4–8}$$

$$P_0^B = \begin{bmatrix} 0 \\ 0 \\ L_1 \end{bmatrix} \tag{4–9}$$

Estos puntos serán constantes en todas las posiciones y orientaciones posibles del robot, al haber elegido los ejes solidarios al brazo.

4.3. Articulación plana de 3 grados de Libertad

Añadiendo un tercer eslabón a la articulación mostrada en el apartado anterior, tenemos un brazo de 3 grados de libertado plano, que, realizando la misma transformación de ejes en la base, pasa a ser uno de 2 grados de libertad. De esta forma, vamos aumentado paulatinamente la complejidad del mismo, para poder realizar más experimentos, y más variados.

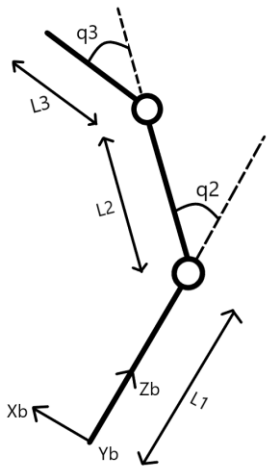

Figura 4-5. Articulación plana de 3 gdl.

4.3.1 Cinemática Directa

De nuevo, empleando el algoritmo de *Denavit-Hartenberg* para resolver el problema cinemático, se obtienen los siguientes parámetros de *Denavit-Hartenberg*:

Tabla 4–2. Parámetros de DH para articulación de 3 gdl.

Articulación i	θ_i	d_i	a_i	α_i
1	q2	0	L2	0
2	q3	0	L3	0

Los ejes del algortimo de *Denavit-Hartenberg* quedan tal que así:

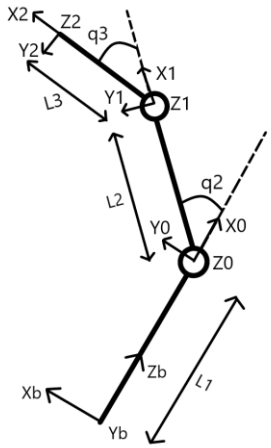

Figura 4-6. Articulación plana de 3 gdl con ejes DH.

Desarrollando (4-3) tenemos que:

$$A_2^1 = \begin{bmatrix} C\theta_3 & -S\theta_3 & 0 & L_3C\theta_3 \\ S\theta_3 & C\theta_3 & 0 & L_3S\theta_3 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4–10}$$

Por lo que nuesta nueva matriz de transformación homogénea queda como:

$$A_1^B = A_0^B * A_1^0 * A_2^1 \tag{4–11}$$

Las ecuaciones que rigen ahora el efector final del nuevo brazo son:

$$P_2^B = \begin{bmatrix} L_2 * \sin(\theta_2) + L_3 * \sin(\theta_2 + \theta_3) \\ 0 \\ L_1 + L_2 * cos(\theta_2) + L3 * \cos(\theta_2 + \theta_3) \end{bmatrix} \tag{4–12}$$

El resto de las ecuaciones son las mismas que para el caso anterior, puesto que se trata de una extension del brazo de 2 grados de libertad plano.

4.4. Articulación de 4 grados de Libertad

Por ultimo, con el objetivo de realizar pruebas con una articulación mas compleja aún, se van a estudiar las ecuaciones cinemáticas para una articulación que se asemeja al funcionamiento de un brazo humano. Al igual que para el caso anterior con tres grados de libertad, este consta de tres eslabones unidos mediante articulaciones rotatorias sobre el mismo plano, con la salvedad de que el efector final debe poder realizar rotaciones en el mismo plano que contiene al resto del brazo y rotaciones en un plano perpendicular, incluso combinando ambas, similar a como se comporta la muñeca de un brazo.

Por consiguiente, el efector final es capaz de moverse en cualquier dirección, independientemente de como este orientado el resto de la articulación.

4.4.1 Cinemática Directa

Mediante el algoritmo de *Denavit-Hartenberg* para resolver el problema cinemático, se obtienen los siguientes parámetros de *Denavit-Hartenberg*:

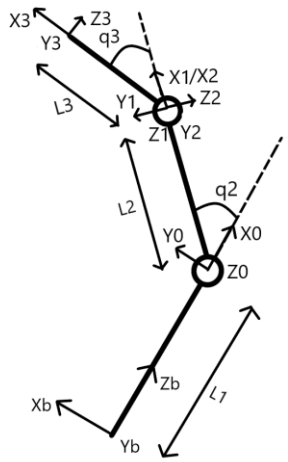

Figura 4-7. Articulación plana de 4 gdl con ejes DH.

Tabla 4–3. Parámetros de DH para articulación de 4 gdl.

Articulación i	θ_i	d_i	a_i	α_i
1	q2	0	L2	0
2	q3	0	0	$\pi/2$
3	q4	0	L3	0

Desarrollando (4-3) por última vez, tenemos que:

$$A_2^1 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & 0 \\ S\theta_3 & 0 & -C\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4–13}$$

$$A_3^2 = \begin{bmatrix} C\theta_3 & 0 & S\theta_3 & 0 \\ S\theta_3 & 0 & -C\theta_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4–14}$$

Las ecuaciones que gobiernan el efector final de esta nueva configuración son:

$$P_2^B = \begin{bmatrix} L_2 * \sin(\theta_2) + L_3 * \sin(\theta_2 + \theta_3) * \cos(\theta_4) \\ L_3 * \sin(\theta_4) \\ L_1 + L_2 * co\,s(\theta_2) + L_3 * \cos(\theta_2 + \theta_3) * \cos(\theta_4) \end{bmatrix} \tag{4–15}$$

Se aprecia que ahora el efector final tiene componente en el eje *OY* que depende del valor de la cuarta articulación y la longitud del tercer eslabón.

5 IMPLEMENTACIÓN DE LA SOLUCIÓN

Una vez obtenidas las imágenes por cada escena, estas se van a procesar fuera de línea mediante los códigos desarrollados en *Visual Studio Community 2019*. En este apartado se pretende desarrollar en profundidad que partes contiene dicho código, funciones que implementa, así como explicar cómo se ha realizado la calibración de las cámaras, paso imprescindible a la hora de trabajar con un sistema de estas características.

El funcionamiento es simple; en primer lugar, se filtran las imágenes para obtener los contornos de los marcadores. Con las coordenadas en píxeles de dichos marcadores, se triangula la posición de cada uno vistos desde una de las cámaras, que establecemos como 'fija'. Finalmente, mediante una función matemática que relaciona el valor de las articulaciones con las coordenadas obtenidas por triangulación, se resuelve mediante mínimos cuadrados y se obtiene una estimación del valor de dichas articulaciones, según la posición del brazo.

5.1. Calibración de las cámaras

Este paso es imprescindible a la hora de trabajar con un sistema que utilice cámaras de cualquier tipo. Si bien se exploraron varias formas de calibración, siguiendo con lo visto en [23], aplicamos algúnos métodos diferentes a los ahí expuestos, que arrojaron mejores resultados en vistas de una aplicación real.

5.1.1 Calibración intrínseca

Para la calibración intrínseca de cada cámara individual, se hizo uso del *Toolbox* de calibración disponible en *MATLAB* [24]. Esta aplicación permite estimar los parámetros intrínsecos o distorsiones de lentes, listos para usar en cualquier aplicación de visión por computador, como en este caso, la reconstrucción 3D de una escena.

El funcionamiento es bien sencillo; a esta aplicación se le entrega un mínimo de 10 imágenes en las que se muestre un patrón conocido de tablero de ajedrez (numero de filas, columnas y tamaño de los cuadrados) de manera que dicho tablero aparezca en diferentes orientaciones y distancias, con el objetivo de tener información variada de la escena. Como resultado, se obtiene un objeto de *MATLAB* que contiene todos los parámetros intrínsecos de dicha cámara.

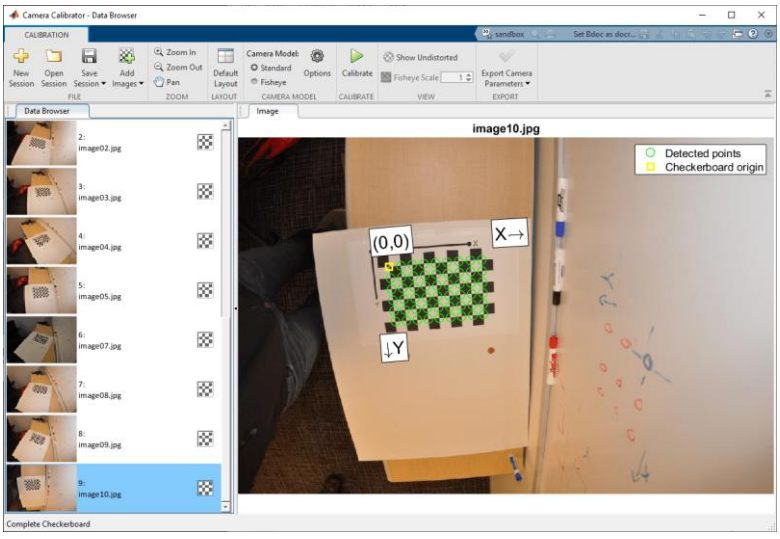

Figura 5-1. Interfaz del Toolbox de calibración de MATLAB.

5.1.2 Calibración extrínseca

Calibrar extrínsecamente consiste en obtener las posiciones relativas de las cámaras con respecto a ellas mismas, o cualquier otro punto de la escena. En este caso, como se requiere conocer las coordenadas de los marcadores con respecto a una de las cámaras, nos conviene conocer la matriz de transformación que permita pasar de la cámara 2 y 3 a la cámara establecida como 1.

Aprovechando las características del problema de la Perspectiva de N-puntos, decidimos utilizar dicho principio para obtener las posiciones relativas de las cámaras. La idea consiste en llenar la escena con marcadores cuyas posiciones son conocidas repartidos en diferentes lugares del espacio de trabajo, y tomar una captura con cada cámara.

Figura 5-2. Marcadores para calibración PnP. Cámara 1.

Para hacer la identificación mas sencilla, se usaron marcadores con un patrón cuadrado, del mismo estilo que en el apartado anterior.

Figura 5-3. Marcadores para calibración PnP. Cámara 2.

Figura 5-4. Marcadores para calibración PnP. Cámara 3.

De esta forma, se toma el origen de coordenadas de la escena como el marcador situado en el centro de la mesa, y se mide la distancia de esta a los demás. De igual forma, en cada imagen se anota la posición en píxeles de cada marcador.

OpenCV tiene una función que resuelve el problema de la Perspectiva de N-puntos, la función *solvePnP()*. Esta función recibe como parámetros de entrada un vector con las posiciones en píxeles de los puntos y otro con las coordenadas reales. Es imprescindible que ambos vectores estén ordenados, y que cada entrada de posición en píxeles coincide con sus coordenadas reales. Además, a esta función hay que alimentarla con la matríz intrínseca de la cámara en cuestión, puesto que hay que aplicarla para cada cámara individualmente. Finalmente, el resultado se recibe como un vector de traslación, y otro vector de rotación de 3 componentes, que puede pasarse a matriz de rotación mediante la función *Rodrigues()*, que se basa en la formula de Rodrigues [25].

Con este resultado, es directo formar 3 matrices de transformación homogénea del tipo $T_w^{c_i}$, donde c_i indica a que cámara corresponde.

Es trivial, por tanto, obtener:

$$T_{c_1}^{c_2} = T_w^{c_2} * \left(T_w^{c_1}\right)^{-1} \tag{5–1}$$

$$T_{c_1}^{c_3} = T_w^{c_3} * \left(T_w^{c_1}\right)^{-1} \tag{5–2}$$

Para comprobar los resultados, se midió la distancia entre las cámaras, dando unos resultados aceptables:

Tabla 5–1. Distancias desde las cámaras.

[m]	C2	C3
C1	2.844	2.873
C2	x	4.109

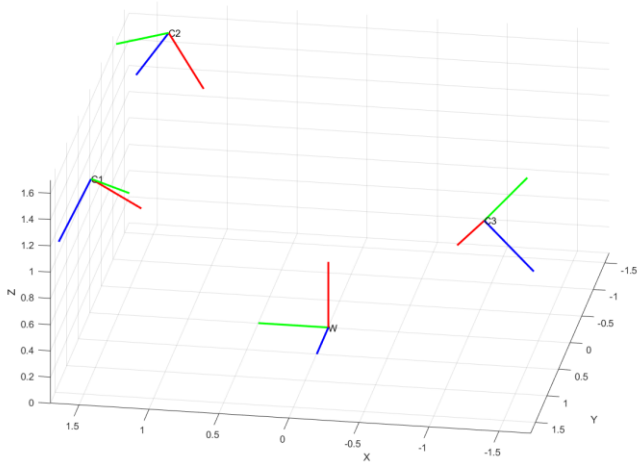

Figura 5-5. Representación en MATLAB del sistema multicámara.

A continuación se muestra un fragmento del código empleado para obtener las matrices de calibración extrínseca de cada una de las cámaras:

Código 5-1 Fragmento del código utilizado para la calibración extrínseca.

```
Mat rvec, tvec, rmat; //Vectores de rotacion y traslacion resultantes
vector<Point3f> objectPoints; //Coordenadas 3D de los marcadores
vector<Point2f> imagePoints; //Coordenadas 2D de los marcadores
Mat cameraMatrix1, cameraMatrix2, cameraMatrix3; //Matrices intrínsecas
Mat distCoeffs1, distCoeffs2, distCoeffs3; //Coeficientes de distorsion
for (int camara = 1; camara < 4; camara++) {
        switch (camara) {
                case 1:
                solvePnP(objectPoints, imagePoints, cameraMatrix1, distCoeffs1, rvec, tvec,
                        false, SOLVEPNP_ITERATIVE);
                Rodrigues(rvec, rmat);
                break;
                case 2:
                solvePnP(objectPoints, imagePoints, cameraMatrix2, distCoeffs2, rvec, tvec,
                        false, SOLVEPNP_ITERATIVE);
                Rodrigues(rvec, rmat);
                break;
                case 3:
                solvePnP(objectPoints, imagePoints, cameraMatrix3, distCoeffs3, rvec, tvec,
                        false, SOLVEPNP_ITERATIVE);
                Rodrigues(rvec, rmat);
                break;
        }
}
```

El funcionamiento del código es sencillo. Dentro de un bucle se va aplicando la función *solvePnP()* con los datos de la cámara correspondiente; matríz intrínseca, coeficientes de distorsión, coordenadas XYZ de los marcadores y sus proyecciones en píxeles de la imagen correspondiente, puesto que no en todas las cámaras se ven los mismos marcadores. Estos datos no aparecen declarados como tal en el fragmento anterior, pero puede suponerese que se introducen como dato antes de entrar en el bucle principal.

5.2. Filtrado de imágenes y obtención de los centros de masas

Una vez obtenidas las imágenes, el siguiente paso es procesarlas una a una, escena a escena, de modo que se obtengan los puntos característicos vistos desde cada cámara. Esto se consigue aplicando una serie de operaciones morfológicas con el objetivo de dejar una máscara binaria, en la que solo aparezcan los contornos destacables.

Como primera aproximación, se probó realizando una implementación del algoritmo watershed siguiendo con lo explicado por Elisa Hidalgo en su trabajo *Seguimiento de objetos móviles con sistema de posicionamiento multicámara* [26]. Este algoritmo requiere de una serie de operaciones previas que se numeran a continuación:

1. Algoritmo *distanceTransform* y normalización del resultado.
2. Binarización de la imagen resultante y dilatado posterior.
3. Detección de contornos, incluidos los bordes de los mismos.
4. Generar imagen con contornos etiquetados del resultado anterior.
5. Aplicar *watershed*

Un número tan elevado de pasos previos a aplicar el algoritmo provoca que este sea mucho mas lento de procesar, por lo que si la captura de imágenes ya era lenta de por sí, con este algoritmo de por medio se hacía aún mas lenta, quedando descartado como método para encontrar los centros de los marcadores. Además, este algoritmo tiene más sentido utilizarlo cuando se requiere separar una serie de elementos que se superponen, cosa que en este caso no ocurre, puesto que los marcadores se entuentran en los eslabones de una articulación, la cual nunca va a poder 'atravesarse' a sí misma.

Por este motivo, en su lugar se decidió realizar una serie de operaciones morfológicas sobre las imágenes, que se explican a continuación;

En primer lugar, se realiza un recorte de pixels fijos, de modo que en la imagen resultante aparece centrada la zona de interés del experemiento, eliminando posibles interferencias exteriores.

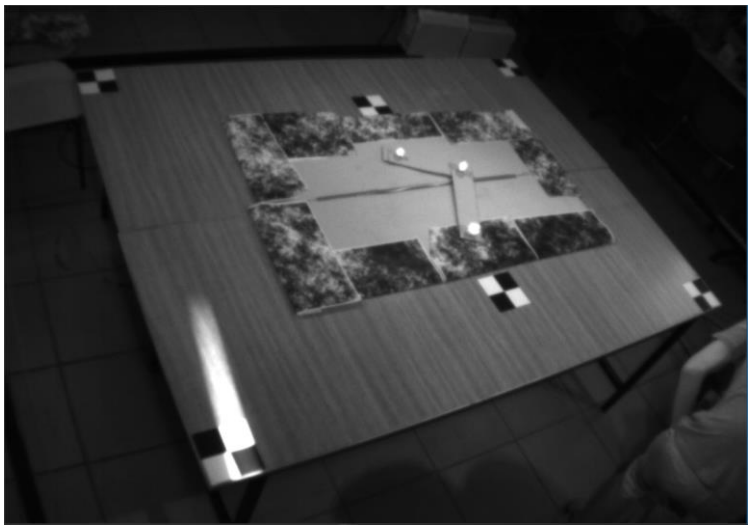

Figura 5-6. Imagen recortada.

El siguiente paso es realizar una binarización de la imagen, aplicando un umbral de 255, puesto que queremos resaltar los contornos blancos de las imágenes, y elimintar el resto de la misma. Además, se aplica un filtrado morfológico para eliminar posibles elementos espúreos de la imágen y obtener unos bordes más definidos del fondo, así como una pequeña dilatación para asegurarnos de que el marcador mas pequeño es reconocido por la función que detecta contornos en la máscara binaria. A continuación, se muestra el fragmento de código correspondiente.

Código 5-2. Filtrado inicial de las imágenes.

```
Mat src = imread(path);
if (src.empty())
{
        return -1;
}

const Rect roi(crop_left[i], crop_top[i], src.cols - crop_right[i] - crop_left[i], src.rows -
crop_bot[i] - crop_top[i]);
src = src(roi).clone();
Mat original = src.clone();

Mat mask;
inRange(src, Scalar(255, 255, 255), Scalar(255, 255, 255), mask);
threshold(mask, mask, 180, 255, THRESH_BINARY_INV);
src.setTo(Scalar(0, 0, 0), mask);

Mat kernel = (Mat_<float>(3, 3) <<
                1, 1, 1,
                1, -8, 1,
                1, 1, 1);
```

```
Mat imgLaplacian;
filter2D(src, imgLaplacian, CV_32F, kernel);
Mat sharp;
src.convertTo(sharp, CV_32F);
Mat imgResult = sharp - imgLaplacian;

imgResult.convertTo(imgResult, CV_8UC3);
cvtColor(imgResult, imgResult, COLOR_BGR2GRAY);

threshold(imgResult, imgResult, 240, 255, THRESH_BINARY | THRESH_OTSU);

Mat transfMorfolog = imgResult.clone();
Mat kernel1 = Mat::ones(5, 5, CV_8U);
Mat kernel2 = Mat::ones(3, 3, CV_8U);

dilate(transfMorfolog, transfMorfolog, kernel1);
Mat transfMorfolog_8u;
transfMorfolog.convertTo(transfMorfolog_8u, CV_8U);
```

Una vez obtenida la mascara binarizada final, se hace una detección de contornos de la misma. La función *findContours()* disponible en *OpenCV* hace esto mismo, etiquetando cada contorno detectado de manera que se pueda acceder a cada uno de ellos de manera independiente. Seguidamente, se comprueba que el numero de contornos detectados es igual al numero de marcadores utilizados. Si es menor, significa que algun marcador no se ha reflejado correctamente, por lo que la imagen se descarta. En cambio, si es mayor, se aplica un filtrado de circularidad según la siguiente fórmula [27]:

$$Circularity = \frac{4 * \pi * Area}{Perimeter^2} \tag{5–3}$$

De este modo, se toman los *n* contornos con mayor circularidad, y se desechan los restantes.

Código 5-3. Filtrado por circularidad de los contornos.

```
//Filtrado por circularidad
if (contours_circ.size() > 3) {
	vector<double> circularity(contours_circ.size());
	vector<double> circularity_sorted(contours_circ.size());

	for (int j = 0; j < contours_circ.size(); j++) {
		double perimeter = arcLength(contours_circ[j], true);
		double area = contourArea(contours_circ[j]);

		if (perimeter > 0) {
			circularity[j] = 4 * M_PI * (area / (perimeter * perimeter));
```

```
                }
        }
        circularity_sorted = circularity;
        sort(circularity_sorted.begin(), circularity_sorted.end(), greater<double>());
        for (int j = 0; j < 3; j++) {
                int s = find(circularity.begin(), circularity.end(), circularity_sorted[j]) -
                        circularity.begin();
                contours.push_back(contours_circ[s]);
        }
}
```

Con los contornos finales obtenidos, se pasa ahora a calcular y guardar en un vector el centro de cada uno en píxeles de la imagen.

Código 5-4. Obtención de los centros de masas de los contornos.

```
for (int seed = 1; seed <= contours.size(); ++seed){
        cv::Mat1b mask = (markers == seed);

        findContours(mask, finalContours, RETR_EXTERNAL, CHAIN_APPROX_SIMPLE);
        mu[seed - 1] = moments(finalContours[0], false);
        mc[seed - 1] = Point2f(mu[seed - 1].m10 / mu[seed - 1].m00, mu[seed - 1].m01 /
                        mu[seed - 1].m00); //Centro de masas
        pos_X[seed - 1] = mc[seed - 1].x;
        pos_Y[seed - 1] = mc[seed - 1].y;
        area[seed - 1] = mu[seed - 1].m00; //Area

        Point2f centro; float radio;
        minEnclosingCircle(finalContours[0], centro, radio);

        pos_X[seed - 1] = centro.x;
        pos_Y[seed - 1] = centro.y;
        circle(drawing, centro, radio, Scalar(255, 0, 255));
        drawContours(drawing, finalContours, -1, Scalar(120, 120, 120), 1, 8, noArray(), 0,
        Point());
}
```

5.2.1 Identificación de los marcadores

Una vez obtenidos los contornos correctos de la imagen, queda entonces identificar cuál es cuál, y asociar la correspondencia de una imagen a otra de las cámaras, paso imprescindible para la triangulación de los puntos.

En este caso no tiene mucho misterio. Al tratarse de marcadores de tamaños distinguidos, basta con ordenar de mayor a menor cada uno de los contornos, puesto que como se ha explicado en el Apartado 3.1.2, para facilitar la identificación y asociación se han colocado marcadores de mayor a menor tamaño, empezando desde la base del brazo hasta el efector final.

Código 5-5. Identificación y asociación de los marcadores.

```
int ord_vec[] = { max_element(area.begin(),area.end()) - area.begin(), -1,
        min_element(area.begin(),area.end()) - area.begin() }; //Ordenamos segun el area
vector<int> indexes = { 0,1,2 };
remove_copy(indexes.begin(), indexes.end(), indexes.begin(), ord_vec[0]);
remove_copy(indexes.begin(), indexes.end(), indexes.begin(), ord_vec[2]);
ord_vec[1] = indexes[0];

for (int j = 0; j < contours.size(); j++) {
        Scalar color = colores[j];
        circle(drawing, mc[ord_vec[j]], 4, color, -1, 8, 0);
        imPoints[i].push_back(Point2f(pos_X[ord_vec[j]] + crop_left[i], pos_Y[ord_vec[j]]
                + crop_top[i]));//sumar los recortes en px
}
im_validas++;
line(drawing, Point2f(pos_X[ord_vec[0]],pos_Y[ord_vec[0]]), Point2f(pos_X[ord_vec[1]],
        pos_Y[ord_vec[1]]), Scalar(0, 255, 255), 1);
line(drawing, Point2f(pos_X[ord_vec[1]], pos_Y[ord_vec[1]]), Point2f(pos_X[ord_vec[2]],
        pos_Y[ord_vec[2]]), Scalar(0, 255, 255), 1);
```

Figura 5-7. Marcadores identificados. Cámara 1.

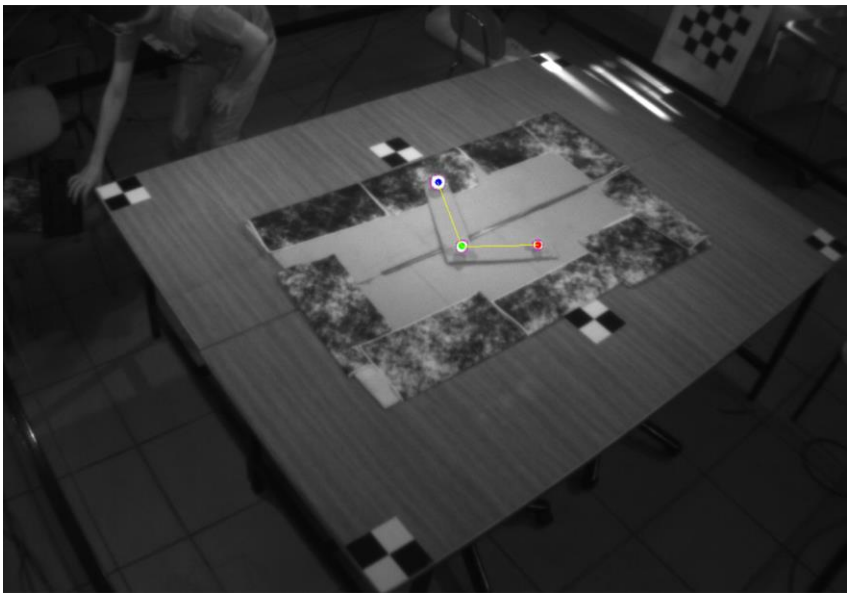

Figura 5-8. Marcadores identificados. Cámara 2.

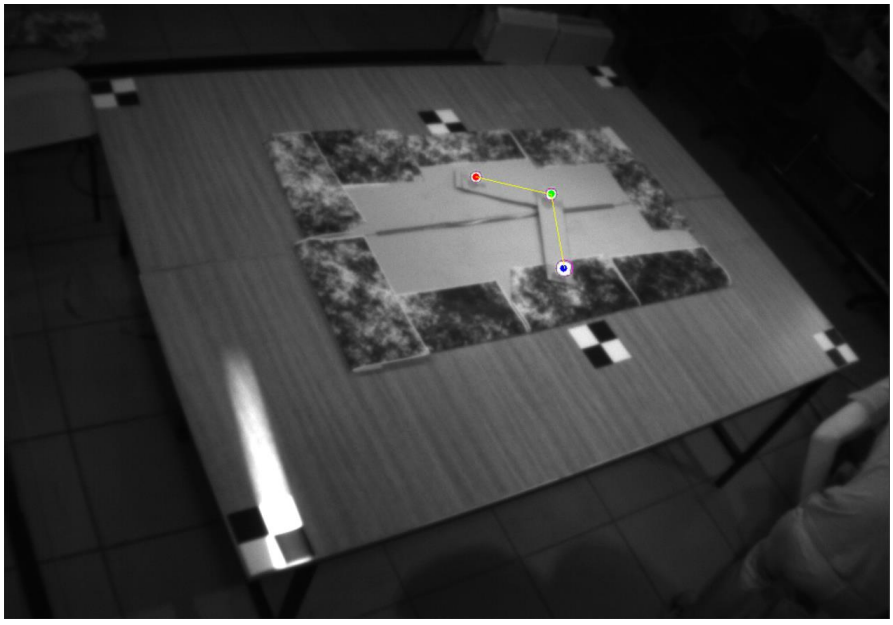

Figura 5-9. Marcadores identificados. Cámara 3.

5.3. Obtención de las coordenadas mediante triangulación

Con los vectores ordenados de los marcadores, el siguiente paso es la triangulación de dichos puntos. *OpenCV* posee una función que realiza dicho trabajo, *triangulatePoints()*, que recibe como parámetros de entrada las matrices de proyección de ambas cámaras, las coordenadas ordenadas de los puntos en píxeles de la imagen, y devuelve una matriz de tamaño 4xN, donde N corresponde con el número de puntos a triangular.

Cada columan de la matriz de salida corresponde con las coordenadas homogéneas de cada punto, por lo que, para pasarlos a coordenadas tridimensionales, hay que dividir cada componente por la última.

La matriz de proyección de entrada de la que se habla no es mas que el producto de la matriz intrínseca de la cámara correspondiente, por la matriz de transformación relativa obtenida mediante la calibración extrínseca. Si queremos que la función nos devuelva las coordenadas respecto de una cámara concreta, la matriz de transformación relativa será entonces la matriz identidad.

A continuación se muestra un fragmento del código que realiza la triangulación.

Código 5-6 Triangulación.

```
Mat points4D_c1c2, points4D_c1c3;
Mat projMatrC2, projMatrC3;
eigen2cv(c2Tc1, projMatrC2);
eigen2cv(c3Tc1, projMatrC3); //Matrices de proyeccion para las camaras 2 y 3

triangulatePoints(cameraMatrix1 * projMatrC1, cameraMatrix2 * projMatrC2,
        imPoints[0], imPoints[1], points4D_c1c2);
triangulatePoints(cameraMatrix1 * projMatrC1, cameraMatrix3 * projMatrC3,
        imPoints[0], imPoints[2], points4D_c1c3);

Eigen::MatrixXf points3D_c1c2, points3D_c1c3, points3D;
cv2eigen(points4D_c1c2, points3D_c1c2); //Pasamos a Eigen, para que sea mas sencillo
operar
cv2eigen(points4D_c1c3, points3D_c1c3);

for (int k = 0; k < 3; k++) {
        points3D_c1c2.block(0, k, 4, 1) = points3D_c1c2.block(0, k, 4, 1) / points3D_c1c2(3,
                k);
        points3D_c1c3.block(0, k, 4, 1) = points3D_c1c3.block(0, k, 4, 1) / points3D_c1c3(3,
                k);
}
points3D = (points3D_c1c2.block(0,0,3,3) + points3D_c1c3.block(0,0,3,3)) / 2.0;
```

Sin embargo, dicha función solo permite la triangulación entre 2 cámaras, por lo que la estimación se ha realizado entre las cámaras 1-3 y 1-2, para después calcular una media entre las dos estimaciones.

5.4. Resolución mediante mínimos cuadrados no lineales

Paso final del algoritmo completo. En este punto solo queda aplicar mínimos cuadrados para obtener todo lo necesario para tener la articulación totalmente definida.

Como se ha visto en el Apartado 2.4, es necesario que se cumpla que dado un número *m* de ecuaciones con *n* incógnitas, $m \geq n$ de modo que existas al menos una solución al problema. Para localizar en el espacio a una articulación hacen falta al menos 6 variables asociadas a la orientación y posición de la base de la mismas, así como 1 variable más por cada grado de libertad que tenga la misma. Del mismo modo, obtenemos 3 ecuaciones por cada marcador adherido a la misma, correspondiente a cada componente cartesiana.

Un sistema con *k* marcadores genera *3k* datos de entrada, obtenidos a través de la triangulación de los mismos, así como *3k* ecuaciones provenientes de la cinemática directa del mismo, suponiendo que dichos marcadores están repartidos por la estructura desde la base hasta el efector final, pasando por cada eslabón de la cadena cinemática. Se tiene que cumplir entonces que $3k \geq 6 + g$ para que exista como mínimo una solución válida para el problema.

5.4.1 Función de coste

La idea principal es, como se muestra en (2-5), resolver una ecuación no lineal que depende de ciertos parámetros, definida como función de coste. En este caso, dado que conocemos coordenadas de los marcadores respecto de la cámara, se necesita una función del tipo:

$$P_m^c = \{Xc; Yc; Zc\} = F(R, P, Y, \theta_1, \dots, \theta_n) \tag{5–4}$$

Con *m* igual al número de marcadores en la estructura, y *n* el número de articulaciones de la misma. Es decir, tenemos 3 ecuaciones no lineales por cada marcador, que describen la posición y orientación de la estructura en el marco de referencia de la cámara correspondiente.

¿Qué función podría cumplir dicha relación? Dado que queremos obtener las coordenadas objeto de la articulación, conocidas las coordenadas en el marco de referencia de la cámara, basta con establecer una relación matemática aplicando una transformación homogénea de coordenadas de la forma:

$$P_m^c = \begin{bmatrix} Xc \\ Yc \\ Zc \\ 1 \end{bmatrix} = T_B^c * P_m^B = T_B^c * \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{5–5}$$

Puesto que tal y como se explicó en el Apartado 4 los ejes de referencia del objeto son solidarios a la estructura, y su origen se sitúa en la base del miso, es directo saber que P_m^B puede sustituirse por las ecuaciones cinemáticas directas correspondientes a cada coordenada, de modo que todas dependen del valor de las articualciones.

La función de coste a resolver queda entonces como:

$$F(x) = T_B^c * P_m^B - P_m^c \tag{5–6}$$

Quedando por tanto un vector de mx3 ecuaciones a resolver, que dependen de 3+n valores.

5.4.2 Estimación de la matriz de transformación homogénea T_B^c

Una matríz de transformación homogénea se compone de una matriz 3x3 de rotación, y un vector 3x1 de traslación. Para este caso particular, es trivial determinar que si el primer marcador está colocado en la base del brazo, sus coordenadas P_1^B serán nulas, al coincidir con el origen de coordenadas.

$$P_B^c = \begin{bmatrix} Xc \\ Yc \\ Zc \\ 1 \end{bmatrix} = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{5–7}$$

$$\begin{bmatrix} tx \\ ty \\ tz \end{bmatrix} = \begin{bmatrix} Xc \\ Yc \\ Zc \end{bmatrix}$$

Con lo que ya tenemos determinado el vector de traslación de la matriz de transformación homogénea, reduciendo el número de ecuaciones totales.

Queda por tanto, parametrizar la matriz de rotación de manera que no presente redundancias a la hora de resolver mediante mínimos cuadrados.

Cualquier cuerpo tridimensional puede ser orientado mediante una sucesión concreta a partide de sus ejes ortogonales [28]. De la terminología usada en la aviación, dichas rotaciones se conocen como alabeo, cabeceo y guiñada.

La guiñada es una rotación de un ángulo α alrededor del eje Z:

$$R_z(\alpha) = \begin{bmatrix} C\alpha & -S\alpha & 0 \\ S\alpha & C\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5–8}$$

El cabeceo es una rotación de un ángulo β alrededor del eje Y:

$$R_y(\beta) = \begin{bmatrix} C\beta & 0 & S\beta \\ 0 & 1 & 0 \\ -S\beta & 0 & C\beta \end{bmatrix} \tag{5–9}$$

El alabeo es una rotación de un ángulo γ alrededor del eje X:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\gamma & -S\gamma \\ 0 & S\gamma & C\gamma \end{bmatrix} \tag{5–10}$$

Una única matriz de rotación puede ser obtenida como una sucesión de todas las anteriores:

$$R(\alpha,\beta,\gamma) = \begin{bmatrix} C\alpha C\beta & C\alpha S\beta S\gamma - S\alpha C\gamma & C\alpha S\beta C\gamma + S\alpha S\gamma \\ S\alpha C\beta & S\alpha S\beta S\gamma + C\alpha C\gamma & S\alpha S\beta C\gamma - C\alpha S\gamma \\ -S\beta & C\beta S\gamma & C\beta C\gamma \end{bmatrix} \quad (5\text{–}11)$$

Finalmente, la matriz de transformación homogénea completa queda de la siguiente manera:

$$T_B^C = \begin{bmatrix} C\alpha C\beta & C\alpha S\beta S\gamma - S\alpha C\gamma & C\alpha S\beta C\gamma + S\alpha S\gamma & tx \\ S\alpha C\beta & S\alpha S\beta S\gamma + C\alpha C\gamma & S\alpha S\beta C\gamma - C\alpha S\gamma & ty \\ -S\beta & C\beta S\gamma & C\beta C\gamma & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5\text{–}12)$$

Una vez hecho esto, el último paso consiste en resolver dicho sistema de ecuaciones no lineales, para lo cual la librería *Ceres* será de gran utilidad. El funcionamiento es sencillo, tan solo hay que declarar cada ecuación de esta forma:

Código 5-7 Declaración de la función de costes para el caso particular de 2 gdl.

```
//FUNCIONES PARA LA OPTIMIZACIÓN DE MINIMOS CUADRADOS NO LINEALES
struct F1 {
        template <typename T>
        bool operator()(const T* const a, const T* const b, const T* const g, T*
        residual) const {
        // f1 = L1*(cos(a)*sin(b)*cos(g)+sin(a)*sin(g))+tx-cP2(1);
        residual[0] = L1 * (cos(a[0]) * sin(b[0]) * cos(g[0]) + sin(a[0]) * sin(g[0])) + tx -
        cP2[0];
        return true;
        }
};
```

El código de arriba describe como se declara cada función de manera independiente para el caso particular de la estructura de 2 grados de libertad. Las funciones restantes se han de declarar de forma similar, para finalmente añadirlos a un objeto del tipo *ceres::Problem* y resolver.

6 ANÁLISIS Y RESULTADOS OBTENIDOS

Particularizando para las tres articulaciones desarrolladas en el Apartado 4, se pretende ahora mostrar los resultados obtenidos en los experimentos realizados. Se han procesado un total de 15 escenas capturadas por cada tipo de articulación, representando diferentes posiciones y orientaciones del mismo con respecto al sistema multicámara, intentando siempre que todos los marcadores sean visibles e identificables. En primer lugar se irá caso por caso mostrando los resultados obtenidos en tablas, así como una pequeña explicación de apoyo a los mismos, para acabar con un apartado final en el que se comparen directamente con el fin de obtener una relación entre estas.

6.1. Articulación de 2 grados de Libertad

6.1.1 Conocidas las dimensiones de las articulaciones

El primer caso de estudio, y el más sencillo, parte de la base de que los parámetros dimensionales de la articulación son conocidos, y se ingresan como dato al resolver mediante mínimos cuadrados.

Tabla 6–1. Dimensiones de las articulaciones de 2gdl.

[m]	L1	L2
Longitud	0.320	0.245

A continuación se muestran varias figuras con algunos de los resultados más representativos. El resto de las imágenes pueden encontrarse en la carpeta anexada *2gdl_result*, que se puede encontrar en [29].

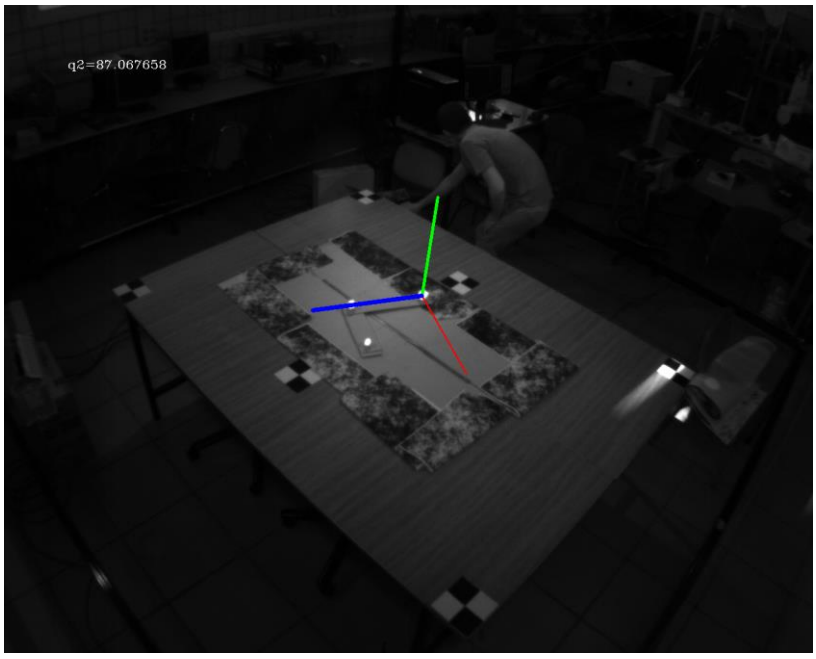

Figura 6-1. Resultado conocidas las dimensiones con 2 grados de libertad (1).

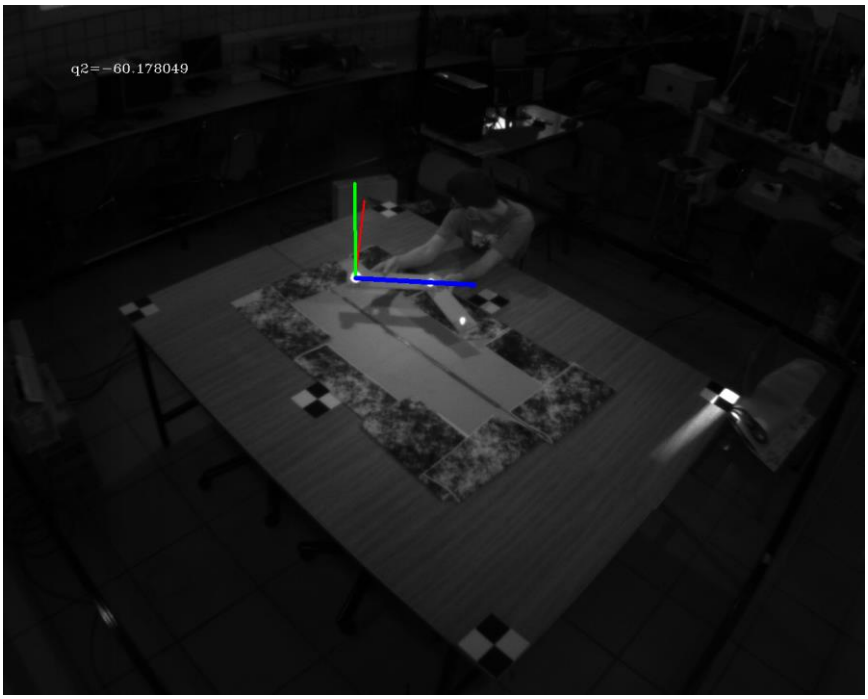

Figura 6-2. Resultado conocidas las dimensiones con 2 grados de libertad (2).

Como puede apreciarse en las figuras de arriba, en la esquina superior izquierda se muestra el ángulo en grados sexagesimales de giro de la articulación, así como una representación de los ejes de referencia solidarios a la misma, vistos desde la perspectiva de la cámara. El eje azul representa el eje *OZ,* el eje verde el *OY*, y finalmente el rojo representa el eje *OX*.

Sin embargo, aunque no es posible medir con exactitud el error en el ángulo de giro real de la articulación, si es posible hacer una estimación de este comparándolo con el ángulo que forma la intesección de las dos rectas resultantes de unir los puntos tridimensionales estimados mediante la triangulación como:

$$\theta_i^{3D} = \tan^{-1}\left(\frac{\|\vec{u}x\vec{v}\|}{\vec{u}\cdot\vec{v}}\right) \tag{6–1}$$

En la siguiente tabla se muestran todos los resultados obtenidos en los experimentos, incluidos los ángulos de alabeo, cabeceo y guiñada de los que se habla en el Apartado 5.4.2:

Tabla 6–2. Resultados para articulación 2 gdl.

Experimento	α (º)	β (º)	γ (º)	$\lvert\theta_2\rvert$ (º)	$\lvert\theta_2^{3D}\rvert$ (º)
1	60.1755	43.4861	-119.9750	87.0677	89.4593
2	-62.5031	-23.1977	81.3396	35.1489	37.9937
3	-37.2338	-99.5571	47.1376	13.1825	15.5463
4	40.5744	38.5570	72.9666	56.1870	56.0635

5	224.6410	-34.5401	309.1800	133.8340	133.6415
6	28.4649	-70.5804	9.9090	180.0000	137.5613
7	2.1784	-28.1748	66.5897	137.4990	135.9940
8	-31.8592	-20.8727	72.4922	50.6513	52.1877
9	58.3341	-71.9499	3.5011	4.7214	5.2811
10	48.1147	41.0113	70.5731	9.8216	9.4797
11	-92.2924	-38.1493	-81.8170	60.1780	55.5678
12	-12.6163	-26.6859	-137.3850	41.7421	43.8309
13	-73.3419	-57.3823	84.2472	24.5435	25.4572
14	-23.7907	-21.2703	38.5104	58.7652	64.5371
15	-219.1590	46.2557	-236.6980	126.3050	127.0512

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

Como puede apreciarse en la tabla de más arriba, el error del ángulo de la articulación no supera los 5 grados como máximo, lo cual se entiende como una muy buena estimación.

La fila sombreada de color rojo indica que dicho experiemento se da por nó valido, al arrojar unos resultados erróneos de la articulación correspondiente. Esta detección se ha realizado fuera de línea, una vez obtenidos los resultados y han sido comparados con las estimaciones geométricas obtenidas mediante (6-1).

Cabe destacar que, tal y como se aprecia en esta tabla y en las siguientes, los ángulos de las articulaciones se muestran en su valor absoluto. Esto es así debido a que no existe una única solución para las ecuaciones cinemáticas, así como para (5-6), y es que la disposición de los ejes del brazo tiene cierta arbitrariedad, lo cual condiciona la solución y no por ello son erróneas, salvo en casos muy límite. Es algo de lo que se hablará mas adelante.

6.1.2 Estimando previamente las dimensiones mediante los puntos triangulados

Ya que se tiene una estimación de las coordenadas tridimensionales de los marcadores, es interesante comprobar si es posible obtener un resultado aceptable utilizando la distancia que hay entre los marcadores para medir la longitud de las articulaciones, en lugar de utilizar la medida real. Al tratarse de puntos en el espacio cartesiano, y que además estos se encuentran totalmente identificados, basta con calcular el módulo del vector que une ambos marcadores.

$$L_i = \|\vec{v}\| = \sqrt{x^2 + y^2 + z^2} \tag{6–2}$$

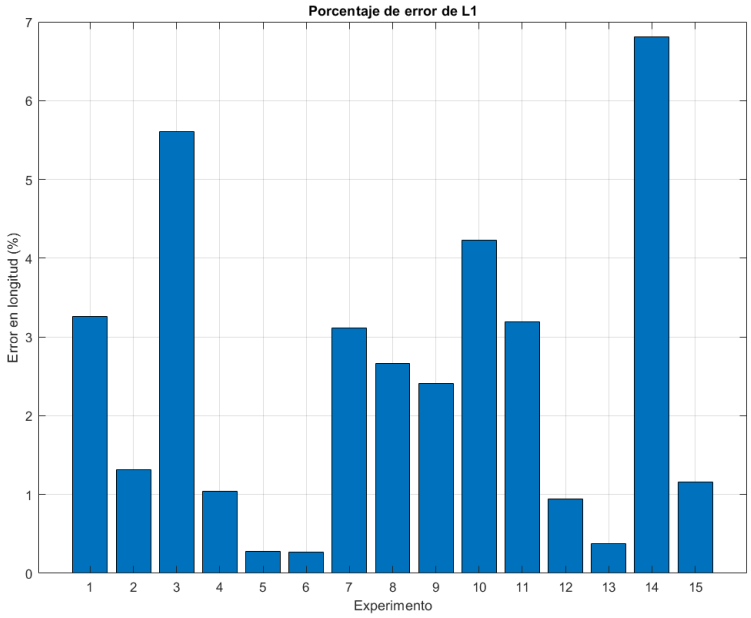

Figura 6-3. Error en porcentaje de la estimación de L1.

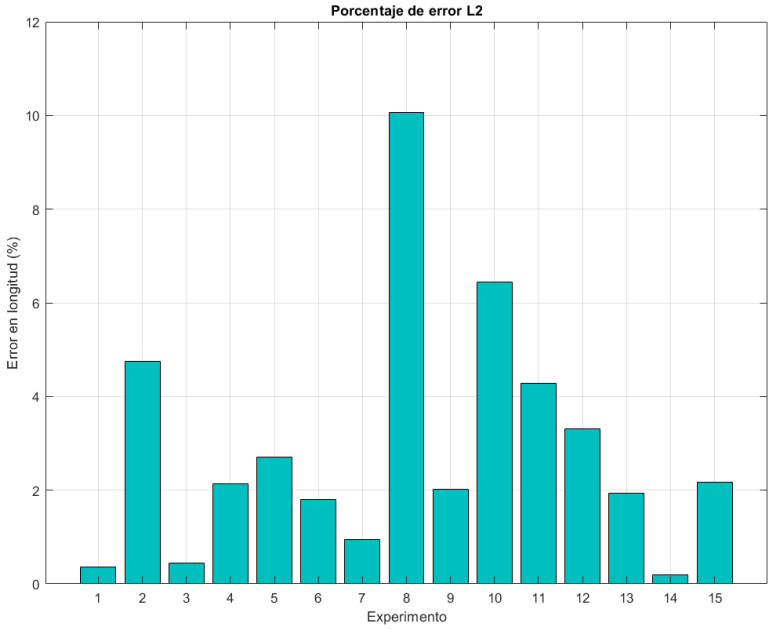

Figura 6-4. Error en porcentaje de la estimación de L2.

Se observa que como máximo, el error en la estimación geométrica de las longitudes de las articulaciones no supera el 10% del tamaño real de estas, lo cual se traduce en un resultado muy similar al obtenido usando las dimensiones reales.

Tabla 6–3. Resultados para articulación 2 gdl estimando longitudes.

Experimento	α (º)	β (º)	γ (º)	$\lvert\theta_2\rvert$ (º)	$\lvert\theta_2^{3D}\rvert$ (º)
1	60.1164	43.4608	-120.0160	89.4596	89.4593
2	-61.2808	-23.0206	80.8601	37.9945	37.9937
3	-34.6625	-99.9656	44.6025	15.5460	15.5463
4	40.2070	38.4683	72.7379	56.0643	56.0635
5	-134.8770	-34.8620	-51.0944	133.6420	133.6415
6	32.0209	-70.8259	6.2168	180.0000	137.5613
7	3.0218	-27.8495	66.1936	135.9940	135.9940
8	-30.1234	-20.3471	71.8820	52.1879	52.1877
9	58.2981	-72.1298	3.5352	5.2823	5.2811
10	48.0180	40.9876	70.5094	9.4796	9.4797
11	-90.4485	-38.3441	-82.9581	55.5678	55.5678
12	-11.9857	-26.0647	-137.6650	43.8315	43.8309
13	-74.0248	-57.4144	84.8221	25.4595	25.4572
14	-23.2500	-20.6280	38.3185	64.5372	64.5371
15	-218.8360	46.1080	-236.4650	127.0520	127.0512

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

Es obvio que el resultado del ángulo de las articulaciones es ahora mucho mas cercano al estimado geométricamente con los puntos tridimensionales dado que la longitud medida de estas depende de la precisión al tomar el centro en píxeles de cada marcador, para luego obtener dichas coordenadas mediante triangulación. Es decir, estamos haciendo que la ecuación (5-6) dependa aún más de las coordenadas trianguladas previamente, por lo que el resultado será mucho mas similar al estimado mediante geometría.

6.2. Articulación de 3 grados de libertad

6.2.1 Conocidas las dimensiones de las articulaciones

Complicando el caso anterior, se le añade una nueva articulación al final del brazo, también rotativa y cuyas ecuaciones cinemáticas se definen en el Apartado 4.3.1:

Tabla 6–4. Dimensiones de las articulaciones de 3gdl.

[m]	L1	L2	L3
Longitud	0.320	0.245	0.210

A continuación se muestran varias figuras con algunos de los resultados más representativos. El resto de las imágenes pueden encontrarse en la carpeta anexada 3gdl_*result*, dentro del repositorio indicado en [29].

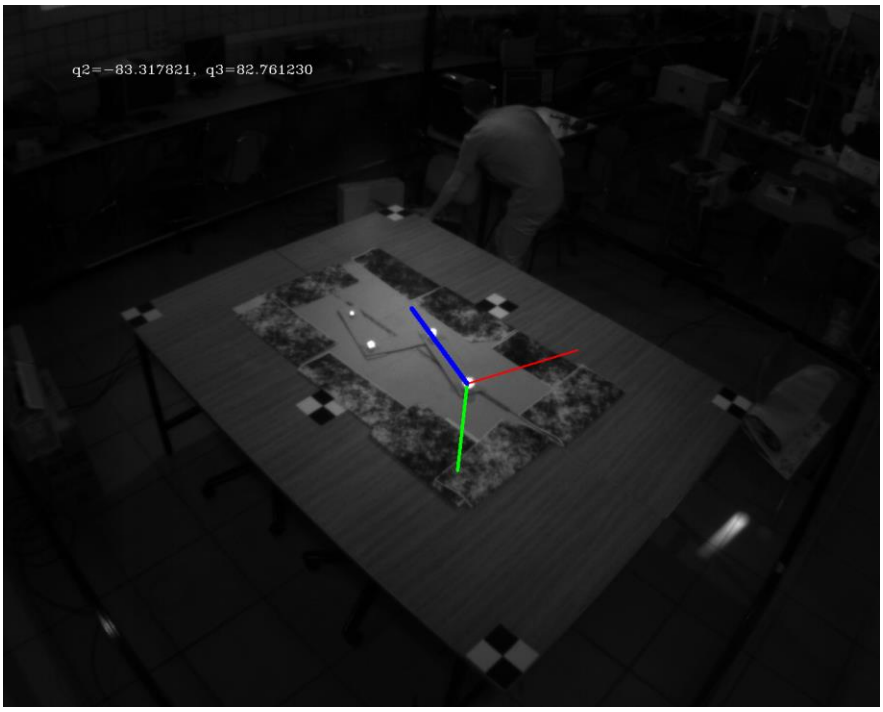

Figura 6-5. Resultado conocidas las dimensiones con 3 grados de libertad (1).

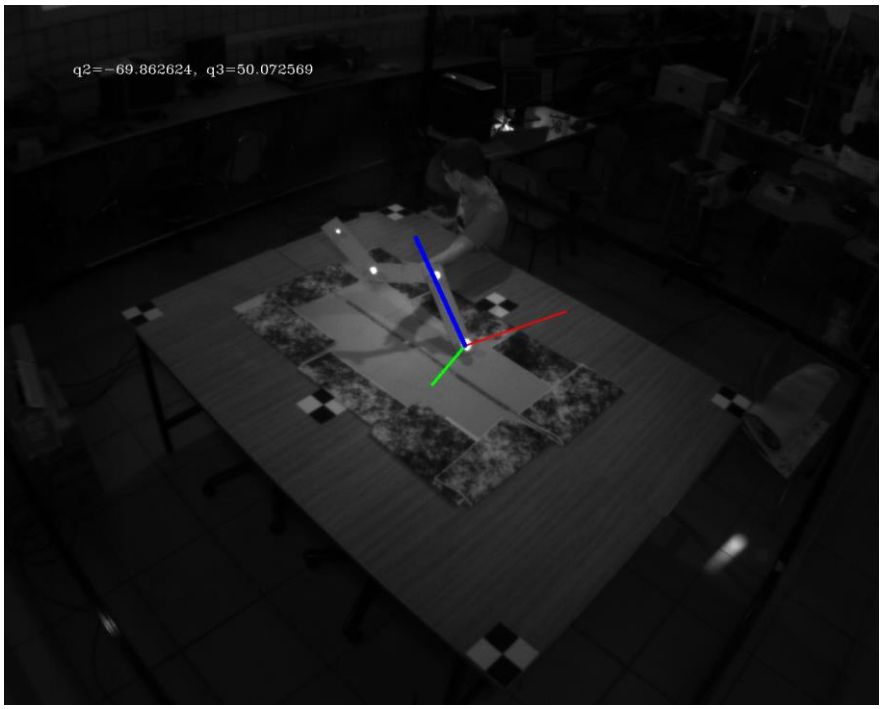

Figura 6-6. Resultado conocidas las dimensiones con 3 grados de libertad (2).

Como puede apreciarse en las figuras de arriba, en la esquina superior izquierda se muestra el ángulo en grados sexagesimales de giro de cada articulación, así como una representación de los ejes de referencia solidarios a la misma, vistos desde la perspectiva de la cámara. El eje azul representa el eje *OZ,* el eje verde el *OY*, y finalmente el rojo representa el eje *OX*.

En la siguiente tabla se muestran los resultados obtenidos de los experimentos realizados:

Tabla 6–5. Resultados para articulación 3 gdl.

Experimento	α (º)	β (º)	γ (º)	$\lvert\theta_2\rvert$ (º)	$\lvert\theta_2^{3D}\rvert$ (º)	$\lvert\theta_3\rvert$ (º)	$\lvert\theta_3^{3D}\rvert$ (º)
1	-3.4615	-60.2811	30.3496	3.3956	4.6037	0.7268	2.6738
2	7.9959	1.6364	38.1017	85.4124	86.3914	2.3285	4.3247
3	-12.0473	-20.4905	42.2260	36.6820	40.5951	3.9230	3.2278
4	-105.1320	-51.2433	108.7870	80.1170	82.2403	0.4543	2.1526
5	9.4461	-52.8011	17.0570	30.7895	52.2235	180.0000	103.3937
6	-15.2490	-23.9273	43.5978	83.3178	85.4103	82.7612	85.1784
7	-68.2302	-54.2675	80.1975	87.2349	89.9104	86.2621	89.2901
8	-16.1042	2.8952	72.9148	66.6080	67.2884	72.9219	75.8305
9	-20.4445	-21.8764	72.5229	39.4018	45.1274	68.8293	75.7581

10	-4.9412	-2.2481	58.7284	23.4100	23.3203	110.4130	114.1732
11	73.8942	-154.5690	-84.0788	180.0000	83.3652	108.8320	84.5188
12	-22.4116	-33.1606	73.2313	7.6798	5.7490	138.6330	140.2073
13	-18.5539	-24.9926	75.6961	69.8626	69.8880	50.0726	50.2731
14	227.1820	-25.2612	-238.7250	144.2780	145.3551	88.1278	89.8313
15	44.3591	-141.4410	-42.8966	144.5480	145.3254	87.6940	89.1506

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

La diferencia en este caso es también pequeña, sin embargo vemos que ahora el número de experimentos marcados como fallido es mayor. Esto puede deberse a que para ciertos casos, la solución para la que convergen los mínimos cuadrados no es la real, sin embargo arroja un residuo mínimo y la toma como solución posible. De hecho, si nos fijamos en la imagen resultante, al representar los ejes de referencia de la articulación se puede apreciar como en todos los experimentos anotados como erróneos, la representación del eje OZ, que debería ser siempre solidaria al primer eslabón del brazo, se encuentra desplazada, provocando dichos errores.

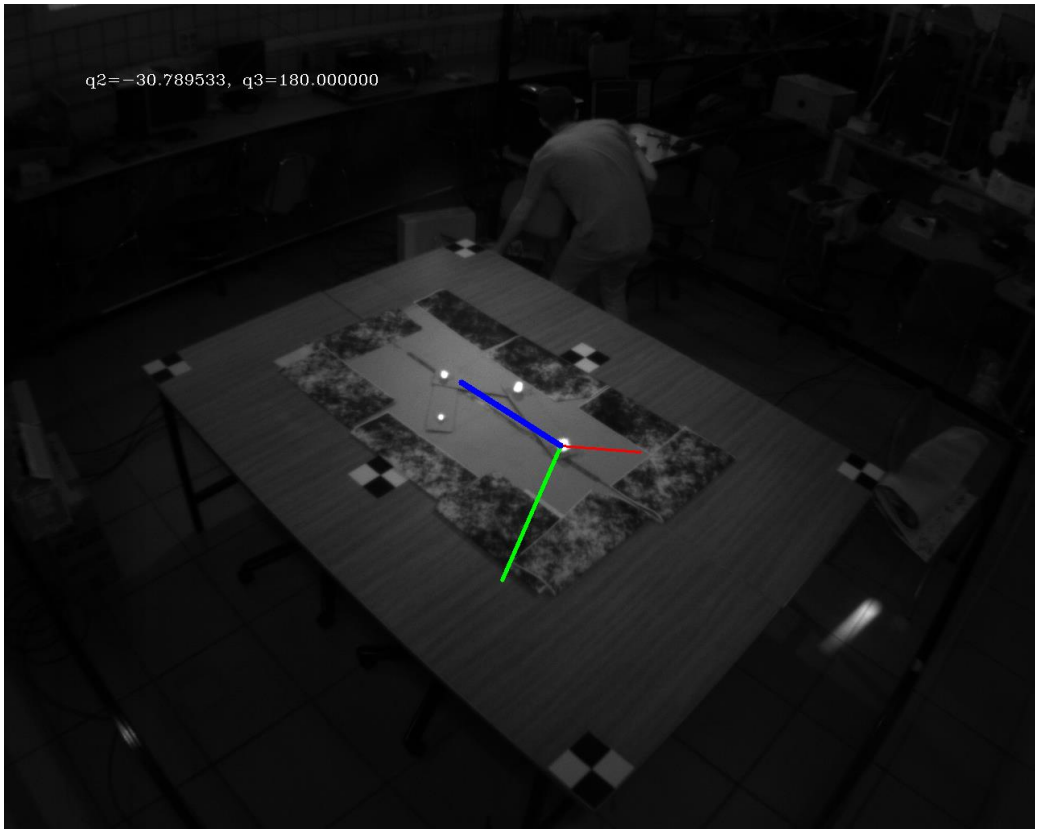

Figura 6-7. Experimento 5, para 3gdl.

6.2.2 Estimando previamente las dimensiones mediante los puntos triangulados

Al igual que para el caso anterior, es interesante comprobar si es posible obtener un resultado aceptable utilizando la distancia que hay entre los marcadores para medir la longitud de las articulaciones, en lugar de utilizar la medida real.

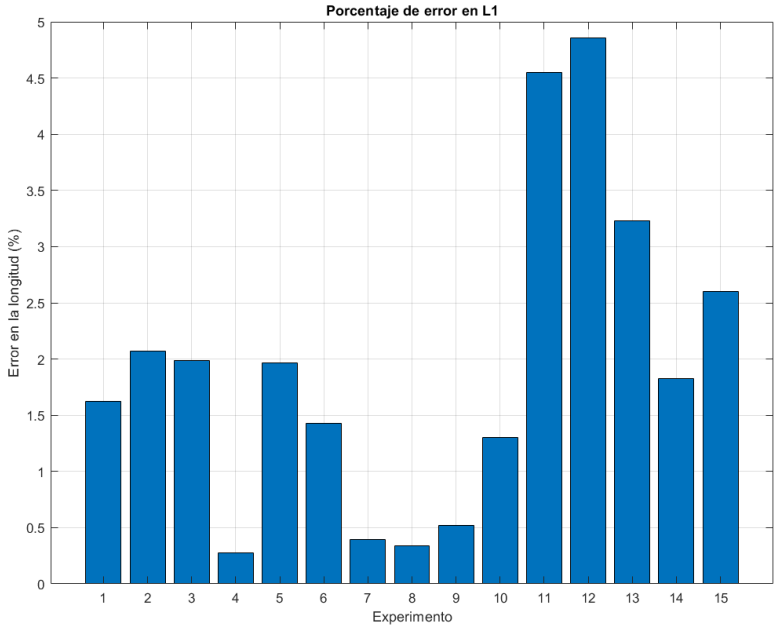

Figura 6-8. Error en porcentaje de la estimación de L1.

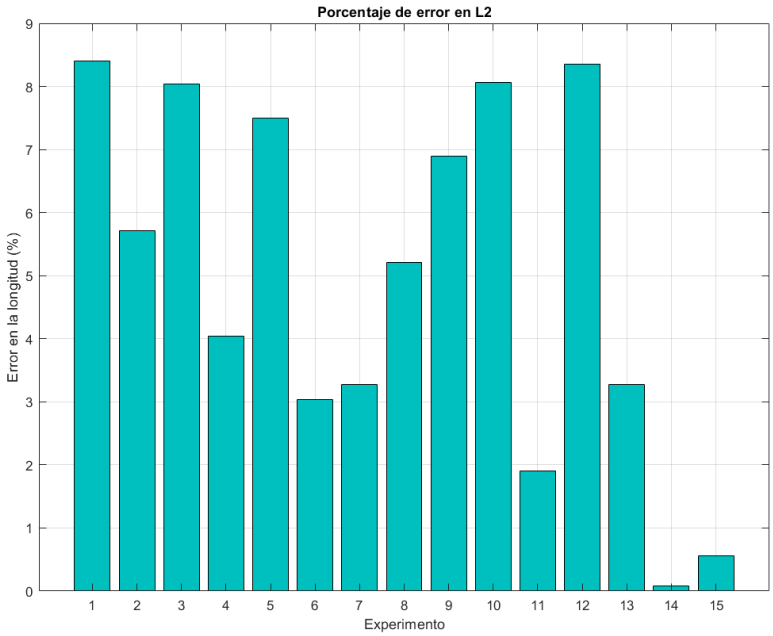

Figura 6-9. Error en porcentaje de la estimación de L2.

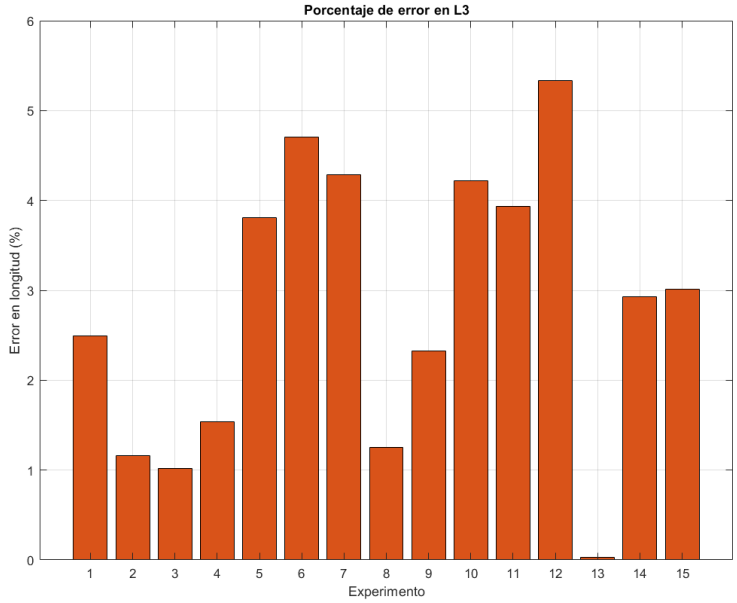

Figura 6-10. Error en porcentaje de la estimación de L3.

Puede observarse que el error no supera el 10% del valor real de las dimensiones de las articulaciones, lo cual es bastante aceptable, y pueden darse por estimaciones relativamente buenas.

Se muestra ahora una tabla con los resultados obtenidos mediante este método:

Tabla 6–6. Resultados para articulación 3 gdl estimando longitudes.

Experimento	α (º)	β (º)	γ (º)	$\lvert\theta_2\rvert$ (º)	$\lvert\theta_2^{3D}\rvert$ (º)	$\lvert\theta_3\rvert$ (º)	$\lvert\theta_3^{3D}\rvert$ (º)
1	-2.2829	-61.1917	29.3540	4.3576	4.6037	0.3991	2.6738
2	9.4741	3.5122	38.1639	86.3941	86.3914	3.8795	4.3247
3	-10.2285	-18.5580	41.6232	40.5944	40.5951	3.0218	3.2278
4	-108.0590	-50.5699	111.0560	82.2443	82.2403	0.3158	2.1526
5	16.3228	-53.0248	11.0281	33.5457	52.2235	180.0000	103.3937
6	-14.6719	-23.4147	43.3493	85.4135	85.4103	85.1803	85.1784
7	-69.5247	-54.3419	81.2213	89.9211	89.9104	89.2950	89.2901
8	-14.7993	3.2932	72.9755	67.2605	67.2884	75.7760	75.8305
9	-18.7597	-21.1548	71.9331	44.9847	45.1274	76.4182	75.7581
10	-5.6454	-2.6850	58.7642	23.3029	23.3203	114.1720	114.1732
11	75.7996	-153.5810	-84.7218	180.0000	83.3652	108.3600	84.5188

12	-22.3442	-32.9703	73.1641	5.2605	5.7490	140.2220	140.2073
13	-17.7076	-24.8121	75.3303	69.8928	69.8880	50.2680	50.2731
14	360.0000	-125.6460	-360.0000	148.6700	145.3551	117.8870	89.8313
15	43.9630	-141.7770	-42.6500	145.3260	145.3254	89.1503	89.1506

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

De nuevo, reducimos los errores en píxeles en los casos en los que los experimentos se dan por válidos, por la misma razón que ocurría en el ejemplo de la articulación de 2 grados de libertad. Sin embargo, hay uno de los experimentos que ahora se da como erróneo, y antes no, ya que de nuevo, al representar el eje OZ este no se encuentra solidario al primer eslabón del brazo.

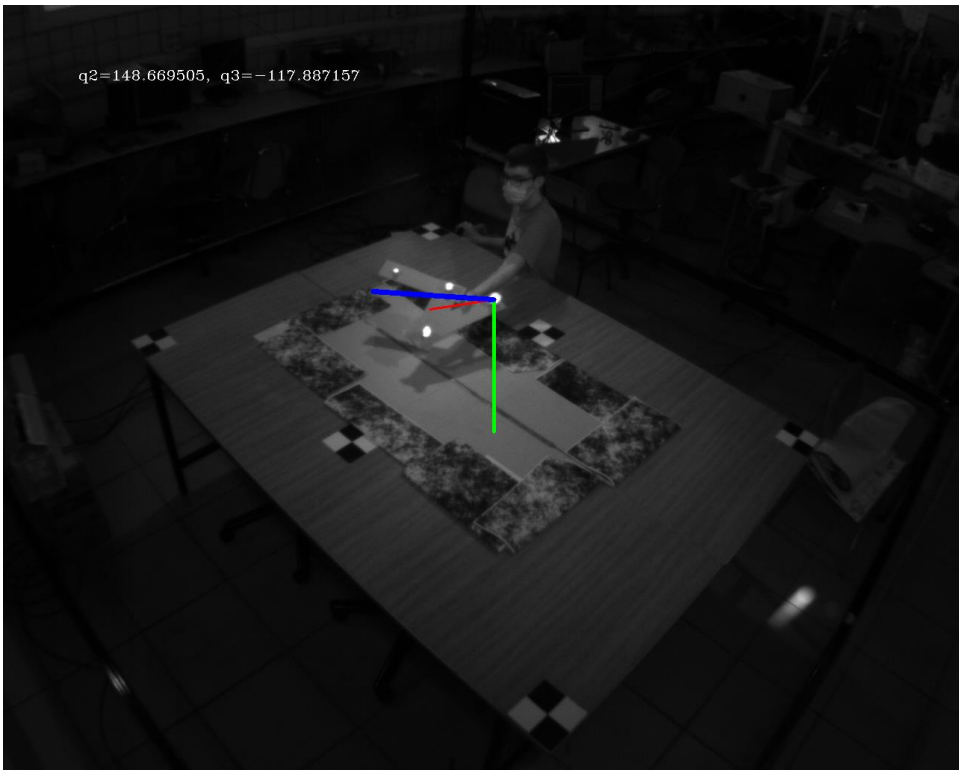

Figura 6-11. Experimento 14 estimando longitudes, 3 gdl.

Es obvio que estimar las longitudes es sensible a los pasos de identificación y triangulación previos, y que puede el número de estimaciones fallidas, anque por el contrario se mejoran los resultados de los experimentos que se dan por válidos, una concesión que puede darse según las especificaciones del sistema.

6.3. Articulación de 4 grados de libertad

6.3.1 Conocidas las dimensiones de las articulaciones

Por último, tenemos el caso completo de articulación, que imitando el movimiento de un brazo real, posee 4 grados de libertad, dos de ellos imitando el comportamiento de una muñeca, permitiendo movimientos del efector final en varios planos consecutivos.

Puesto que para este caso particular no se han aumentado el número de eslabones, las dimensiones siguen siendo las mismas que encontramos en la Tabla 6-4.

A continuación se muestran varias figuras con algunos de los resultados más representativos. El resto de las imágenes pueden encontrarse en la carpeta anexada 4gdl_*result*, dentro del repositorio [29].

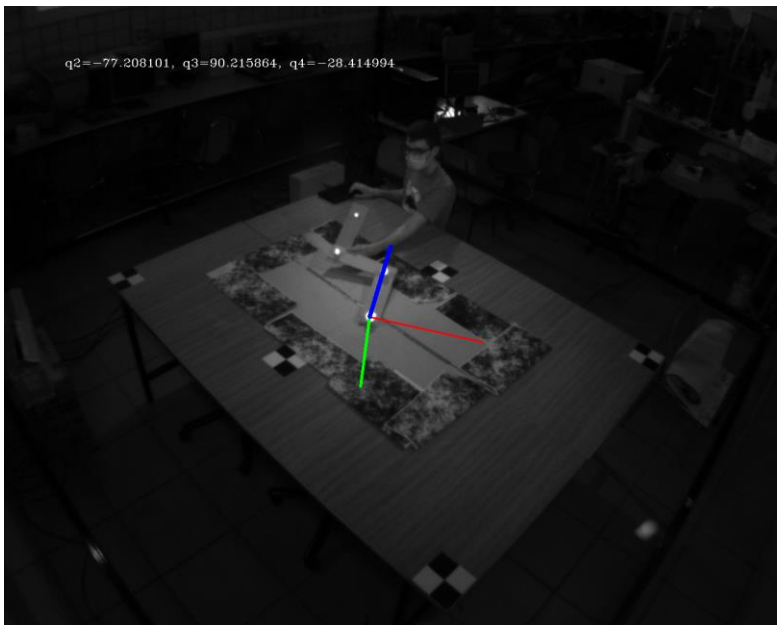

Figura 6-12. Resultado conocidas las dimensiones con 4 grados de libertad (1).

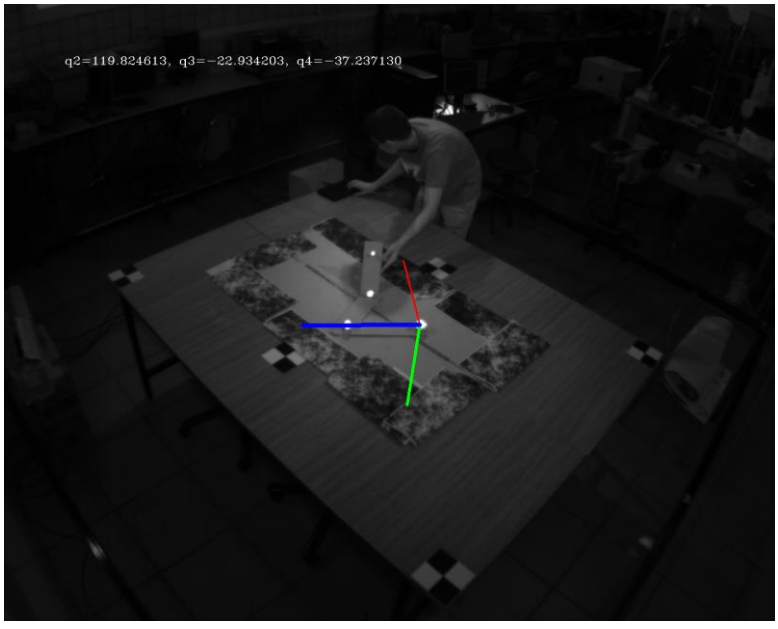

Figura 6-13. Resultado conocidas las dimensiones con 4 grados de libertad (2).

Como puede apreciarse en las figuras de arriba, en la esquina superior izquierda se muestra el ángulo en grados sexagesimales de giro de cada articulación, así como una representación de los ejes de referencia solidarios a la misma, vistos desde la perspectiva de la cámara. El eje azul representa el eje *OZ,* el eje verde el *OY*, y finalmente el rojo representa el eje *OX*.

En la siguiente tabla se muestran los resultados obtenidos de los experimentos realizados:

Tabla 6–7. Resultados para articulación 4 gdl.

Experimento	$\boldsymbol{\alpha}$ (º)	$\boldsymbol{\beta}$ (º)	$\boldsymbol{\gamma}$ (º)	$\lvert\boldsymbol{\theta_2}\rvert$ (º)	$\lvert\boldsymbol{\theta_2^{3D}}\rvert$ (º)	$\lvert\boldsymbol{\theta_3}\rvert$ (º)	$\lvert\boldsymbol{\theta_3^{3D}}\rvert$ (º)	$\lvert\boldsymbol{\theta_4}\rvert$ (º)	$\lvert\boldsymbol{\theta_4^{3D}}\rvert$ (º)
1	-13.68	-54.76	39.05	3.82	5.01	1.66	2.84	0.18	0.41
2	-83.32	40.09	53.27	1.71	2.82	20.70	2.37	1.62	21.92
3	-68.77	-53.41	80.66	72.23	72.32	49.62	53.58	25.57	26.79
4	12.40	0.23	44.03	77.21	75.09	90.22	88.61	28.42	28.32
5	-21.44	-5.31	64.80	42.03	41.21	120.58	64.08	172.76	7.82
6	-45.39	133.06	-40.86	28.52	29.61	122.77	125.74	28.17	27.30
7	-241.97	72.19	-237.73	180.00	63.11	154.16	53.60	29.40	56.61
8	-26.96	8.15	79.93	31.42	32.31	62.63	65.18	46.47	45.40
9	4.41	-15.46	53.59	143.53	144.88	123.34	58.14	180.00	18.66
10	3.47	-18.96	50.19	144.88	144.76	178.49	2.02	162.61	18.04
11	-9.67	-18.27	41.74	76.66	78.87	61.25	63.78	1.35	1.39
12	255.66	-51.62	-251.74	119.83	118.56	22.93	21.74	37.24	36.51
13	-81.43	-11.47	93.84	106.49	105.25	36.10	40.41	10.88	11.59
14	89.76	-127.10	-83.43	110.64	107.58	180.00	137.58	2.16	1.74
15	-44.78	-43.52	77.62	72.40	74.76	103.07	106.00	3.03	2.97

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

En una primera aproximación fuera de línea de los resultados, el número de experimentos marcados como erróneos comparando con las estimaciones geométricas es mucho mayor que para los casos anteriores, en parte ya que se trata de un caso más complejo en el que aumentan el número de incógnitas pero no el de ecuaciones. Si bien es cierto que el número de ecuaciones sigue siendo mayor que el número de incógnitas es posible que el modelo real utilizado no sea el más idóneo para imitar el comportamiento de la ariculación explicada en el Apartado 4.4, pues los eslabones no son muy resistentes, provocando que algunos de los marcadores no quedaran en las posiciones correctas.

Sin embargo, al reproyectar según los ángulos de α, β, γ, estos compensan que las coordenadas XYZ de los puntos al resolver la cinemática directa se reproyecten muy cerca de donde se encuentran los centros de los marcadores en la imagen. Se trata pues, de un problema común en la cinemática de robots y es que no hay en general soluciones únicas para la cinemática.

La disposición de los ejes base dispone de cierta arbitrariedad para ser representados (excepto el eje OZ, que debe ser siempre solidario al primer eslabón de la cadena), lo cual condiciona la solución adoptada por el método, no por ello errónea.

6.3.2 Estimando previamente las dimensiones mediante los puntos triangulados

Finalmente, al igual que se ha probado en los apartados anteriores, es posible estimar posteriormente a la triangulación la longitud de los eslabones de la articulación, y utilizar estos datos para resolver mediante mínimos cuadrados.

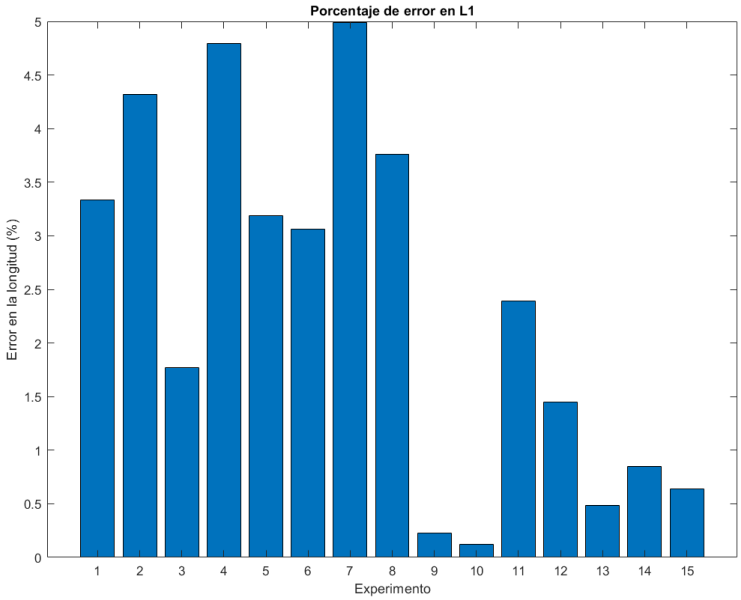

Figura 6-14. Error en porcentaje de la estimación de L1.

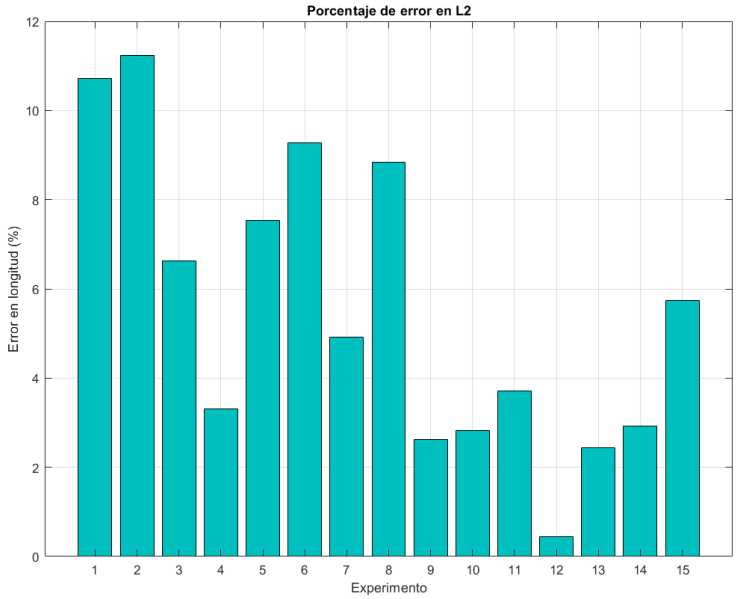

Figura 6-15. Error en porcentaje de la estimación de L2.

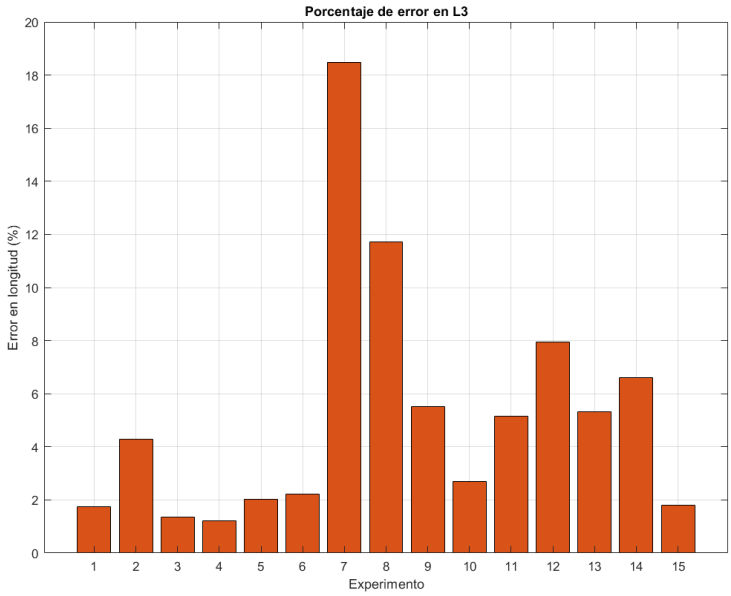

Figura 6-16. Error en porcentaje de la estimación de L3.

Se puede comprobar que en el peor de los casos, el error no supera el 12% de la longitud total para los casos de L_1 y L_2, lo que supone un error de no más de 3 centímetros. Sin embargo, para L_3 encontramos picos de casi un 20% de la longitud real, por lo que hablamos de casi 5 centímetros, un error que empieza a ser preocupante. En realidad, el alto error tiene su explicación en el modelo utilizado. Puesto que las distancias se miden como las distancias que hay desde un marcador a otro, si el modelo no se posiciona correctamente, la distancia a la que se encuentran los marcadores puede no coincidir con la que existe en el eslabón correspondiente. Esto ocurre para el experimento número 7, que tal y como se puede ver en la Figura 6-17, el pliegue que corresponde con la articulación θ_4 o q_4, no se realiza exactamente en el marcador anterior, haciendo que las distancias no sean precisas.

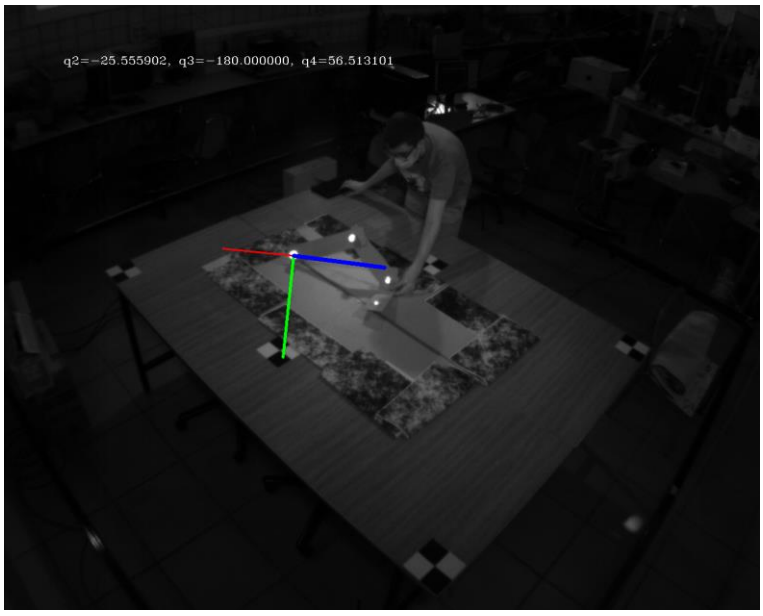

Figura 6-17. Experimento 7, para 4 gdl.

En la siguiente tabla se muestran los resultados obtenidos de los experimentos realizados:

Tabla 6–8. Resultados para articulación 4 gdl.

Experimento	α (º)	β (º)	γ (º)	$\lvert\theta_2\rvert$ (º)	$\lvert\theta_2^{3D}\rvert$ (º)	$\lvert\theta_3\rvert$ (º)	$\lvert\theta_3^{3D}\rvert$ (º)	$\lvert\theta_4\rvert$ (º)	$\lvert\theta_4^{3D}\rvert$ (º)
1	-13.95	-55.34	39.27	5.01	5.01	2.64	2.84	0.23	0.41
2	90.76	-45.75	-49.38	2.77	2.82	23.79	2.37	4.40	21.92
3	-71.37	-54.27	82.75	72.33	72.32	53.58	53.58	26.79	26.79
4	12.39	0.82	44.03	75.09	75.09	88.61	88.61	28.32	28.32
5	-6.30	-27.16	55.54	56.47	41.21	180.00	64.08	123.75	7.82
6	-44.48	133.86	-40.20	29.61	29.61	125.74	125.74	27.30	27.30
7	360.00	114.54	352.20	25.56	63.11	180.00	53.60	56.51	56.61
8	-10.69	-39.25	68.19	48.04	32.31	180.00	65.18	144.19	45.40
9	10.06	-22.85	51.46	144.88	144.88	121.86	58.14	161.34	18.66
10	4.08	-19.67	49.98	145.38	144.76	180.00	2.02	162.06	18.04
11	-8.79	-17.42	41.47	78.87	78.87	63.79	63.78	1.39	1.39
12	256.16	-51.10	-252.13	118.56	118.56	21.74	21.74	36.51	36.51
13	-82.98	-10.73	94.15	105.25	105.25	40.41	40.41	11.59	11.59
14	-94.63	-53.29	100.66	107.58	107.58	42.42	137.58	178.26	1.74
15	-46.26	-43.67	78.64	74.76	74.76	106.00	106.00	2.97	2.97

**Las filas sombreadas de color rojo indican que el resultado se da por erróneo.*

Aplicar este método aumenta el número de experimentos erróneos con respecto al anterior, sin embargo, los resultados son mucho mas cercanos a las estimaciones mediante geometría, que se toman como correcta en todos los casos, como método de comparación.

Al igual que lo expuesto anteriormente, aproximar las longitudes de los eslabones de esta forma depende fuertemente de como de buenas sean las coordenadas trianguladas, por lo que un error en estas se propaga en un error al final.

6.4. Comparativa entre articulaciones

Con el objetivo de facilitar la comparación entre las diferentes articualciones de distintos grados de libertad, y hallar unas conclusiones sobre los diferentes experimentos realizados, en este apartado se irán comparando diferentes aspectos que tienen cierta relevancia en el desarrollo del algoritmo, como pueden ser errores de reproyección, número de iteraciones de la optimización mediante mínimos cuadrados, así como el tiempo empleado en la resolución del mismo.

6.4.1 Error de reproyección

Con los valores obtenidos, que se muestran en las tablas correspondientes de más arriba, es posible realizar una reproyección de los puntos de interés correspondientes con los marcadores distribudios por la articulación, y así conocer la precisión de método y los resultados.

La función *projectPoints()* presente en la librería de *OpenCV* realiza dicha función, proyectando en píxeles de la imagen las coordenadas de un punto respecto de los ejes de referencia del objeto, dadas las matrices intrínsecas y extrínsecas de la cámara. Las coordenadas de los puntos 'objeto' se obtienen fácilmente resolviendo las ecuaciones cinemáticas directas estudiadas anteriormente, por lo que es un paso sencillo y directo de realizar.

Una vez obtenidas las coordenadas en píxeles reproyectadas, se comparan con las obtenidas en los pasos previos del algoritmo, encargados de encontrar e identificar los marcadores de la imagen. Si la diferencia es baja, esto indica que la solución obtenida es buen y con suficiente precisión.

A continuación, se muestran una serie de figuras que corresponden con los errores de proyección según el tipo de articulación:

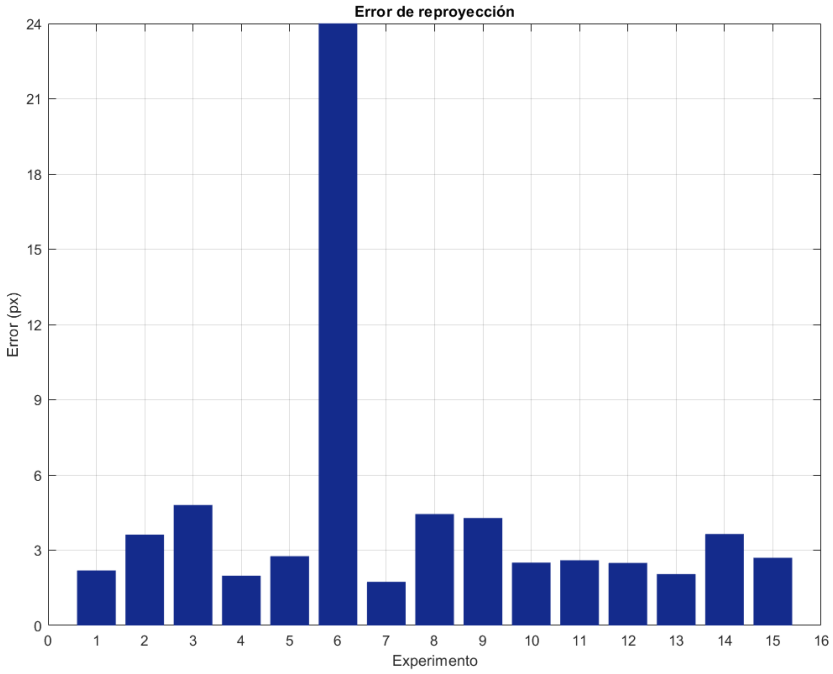

Figura 6-18. Error de reproyección 2gdl.

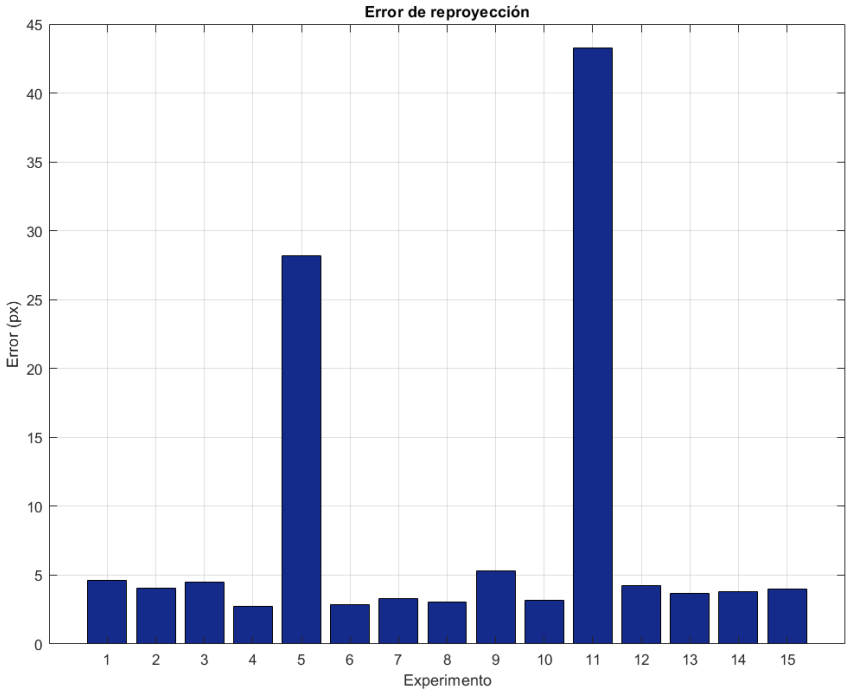

Figura 6-19. Error de reproyección 3gdl.

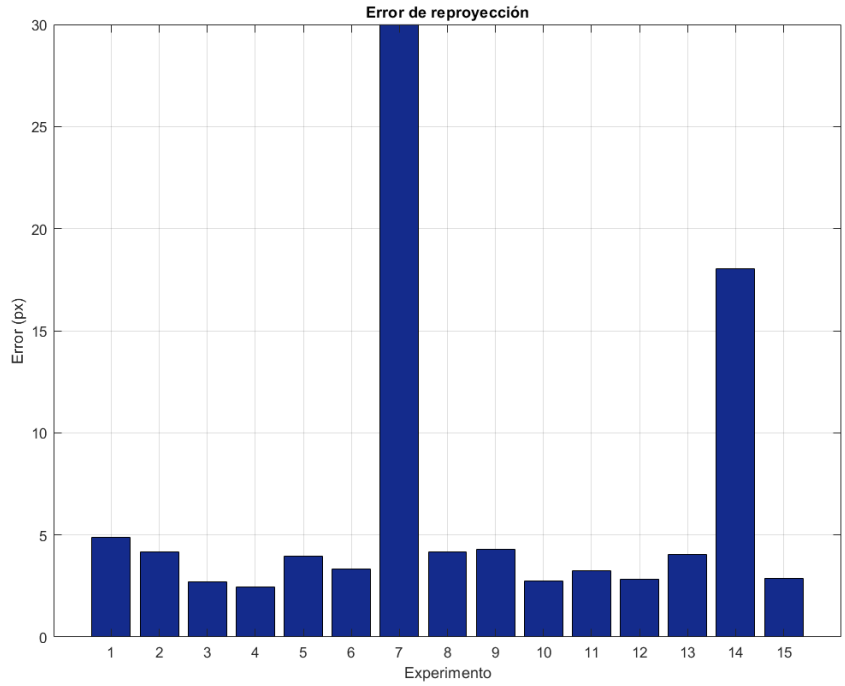

Figura 6-20. Error de reproyección 4gdl.

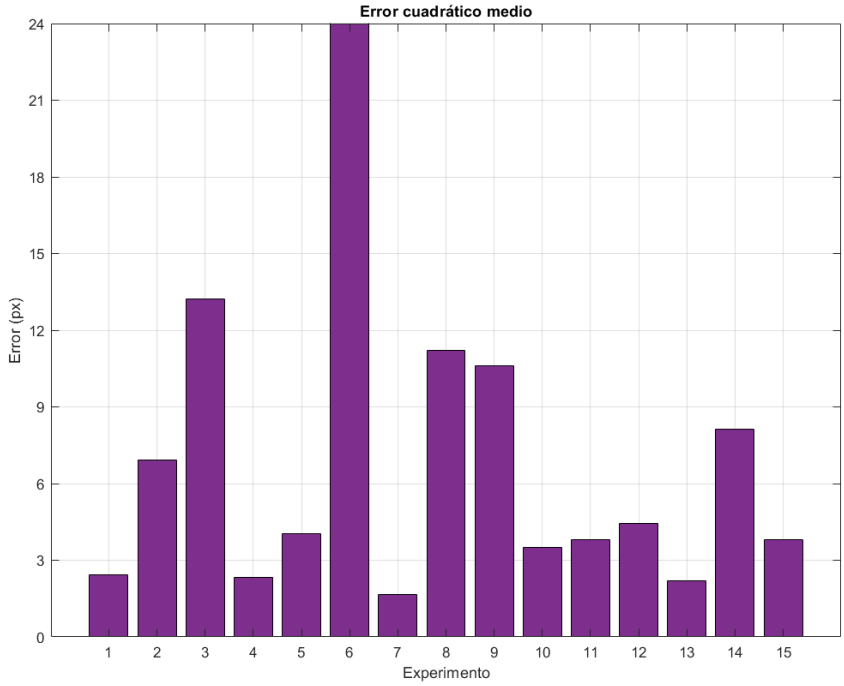

Figura 6-21. Error cuadrático medio 2gdl.

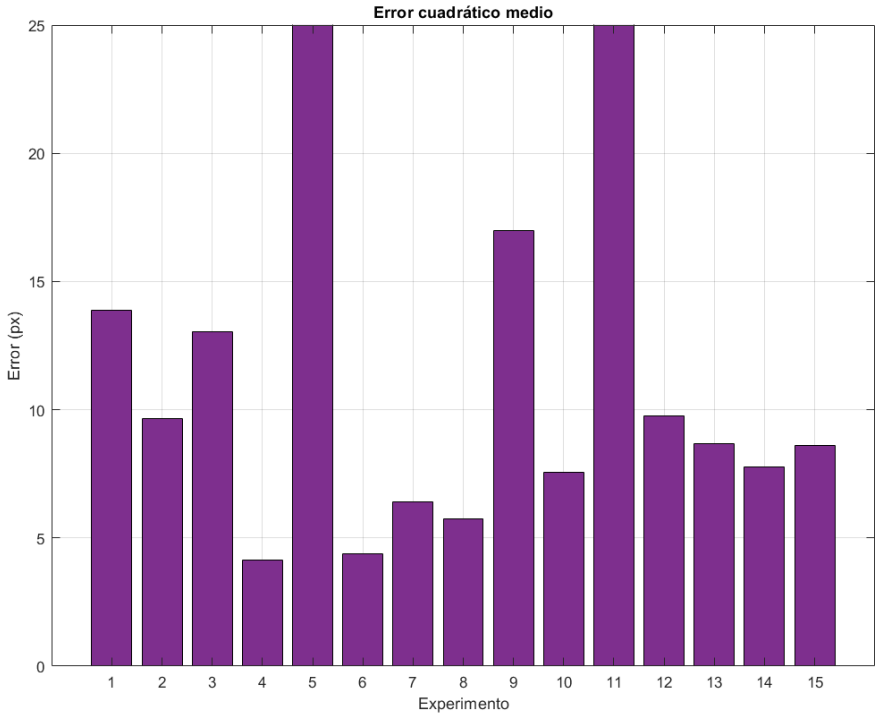

Figura 6-22. Error cuadrático medio 3gdl.

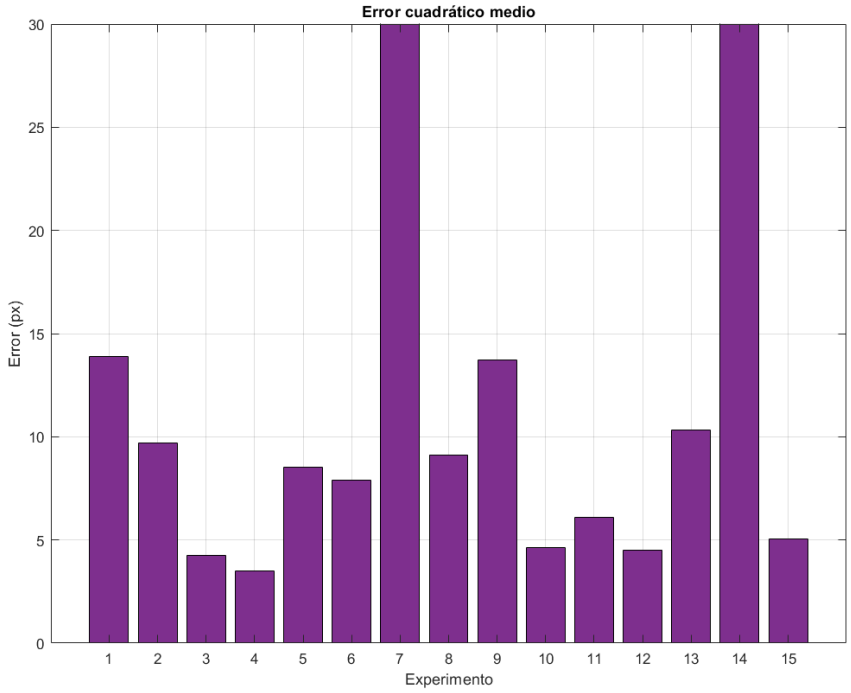

Figura 6-23. Error cuadrático medio 4gdl.

Es directo comprobar como el error de reproyección no parece depender del número de grados de libertad para los experimentos que se dan por buenos según lo expuesto en las tablas de resultados de los apartados anteriores, es decir, si el resultado es correcto, independientemente de la complejidad de la articulación, el error de reproyección será más o menos bajo dependiendo solo del resultado del experimento en concreto. Esto es una gran noticia, puesto que uno puede pensar que a mayor complejidad, mayor será de media el error irremediablemente.

Sin embargo, hay casos en los que el error es tan grande que lo hace inservible. En los apartados anteriores, se habían detectado dichos casos de manera manual, comprobando los resultados uno por uno y comparando los resultados obtenidos con las estimaciones geométricas.

No obstante, sería interesante particularizar un método automático que descarte las estimaciones erróneas basándose en el error de reproyección resultante, de modo que para una posible aplicación en tiempo real en los laboratorios sea posible desechar la estimación de cierto "frame" y trabajar con una media del resto. La opción mas sencilla sería la de utilizar un valor umbral fijo para el cual la estimación se puede dar por válida o no. En cambio, se puede particularizar con un proceso iterativo de manera que el umbral alcance automáticamente un valor concreto dependiendo de la media de errores de reproyección obtenidos, haciéndolo más preciso si cabe.

El pseudocódigo podría plantearse de la siguiente manera:

1. Elegir un umbral *T* inicial. Este debería estar en un punto que permita discernir entre los casos más obvios de error y el resto.
2. Se calcula la media de error de reproyección del resto de casos, una vez desechados los casos que superen el umbral definido en 1.
3. Se establece un nuevo umbral *T* como la media calculada en 2.

4. Se repiten los apartados 2 y 3 hasta que el ΔT sea inferior a cierto valor propuesto, de manera que deje de tener sentido seguir buscando un umbral más ajustado.

Las figuras anteriores corresponden con el caso en el que las longitudes de los eslabones se suponen conocidas. Es interesante también comprobar que ocurre en el caso en el que se estimen dichas longitudes geométricamente mediante las coordenadas trianguladas:

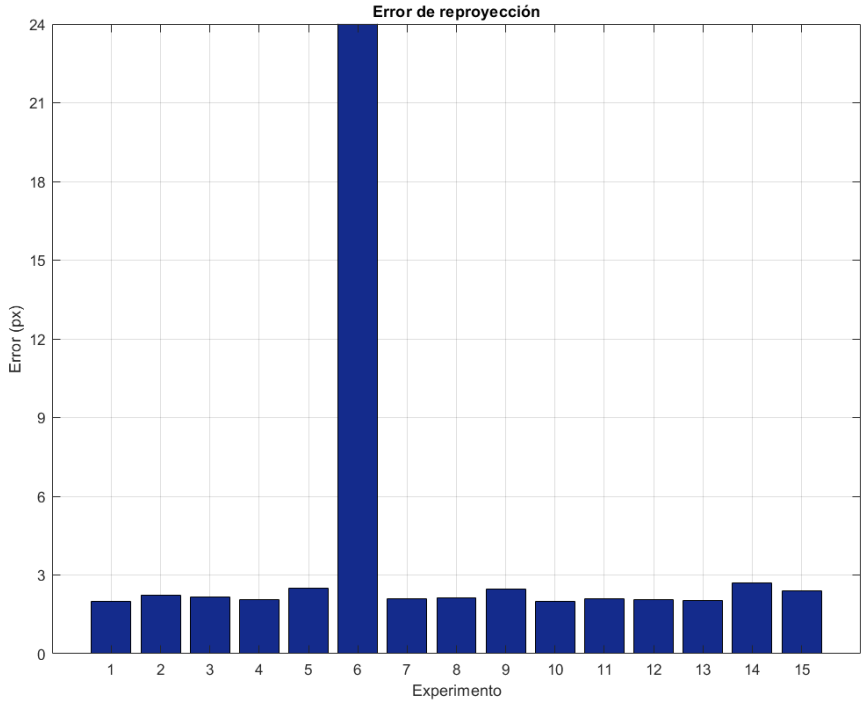

Figura 6-24. Error de reproyección 2gdl, estimando longitudes.

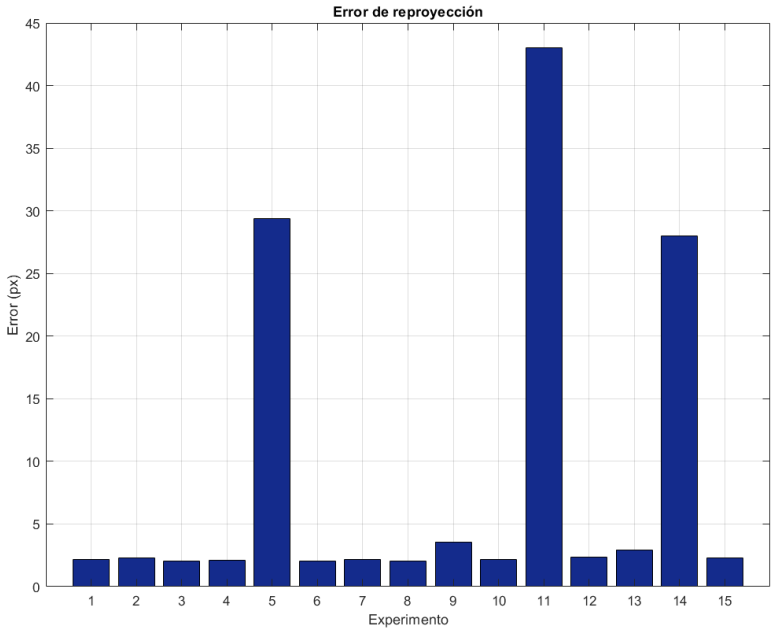

Figura 6-25. Error de reproyección 3gdl, estimando longitudes.

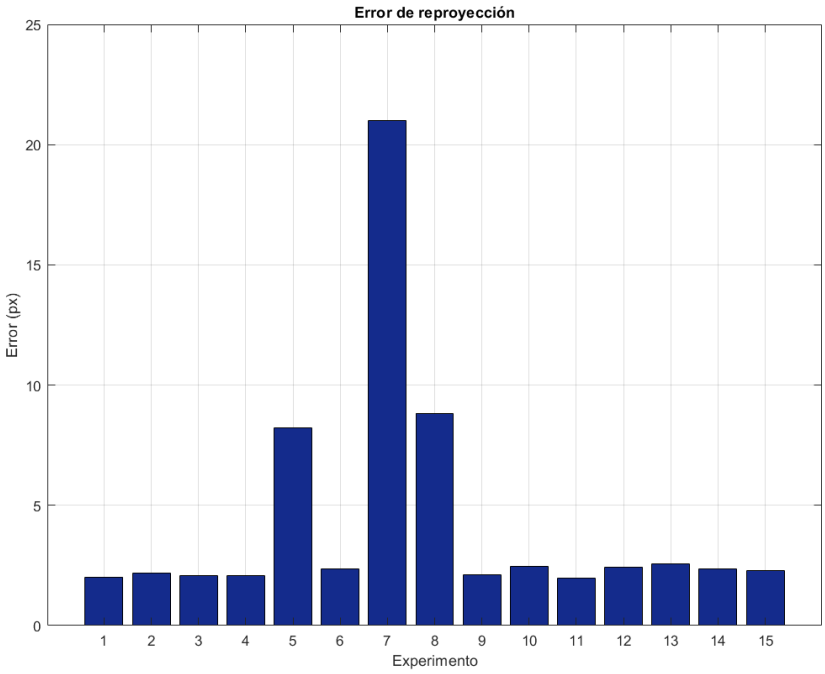

Figura 6-26. Error de reproyección 4gdl, estimando longitudes.

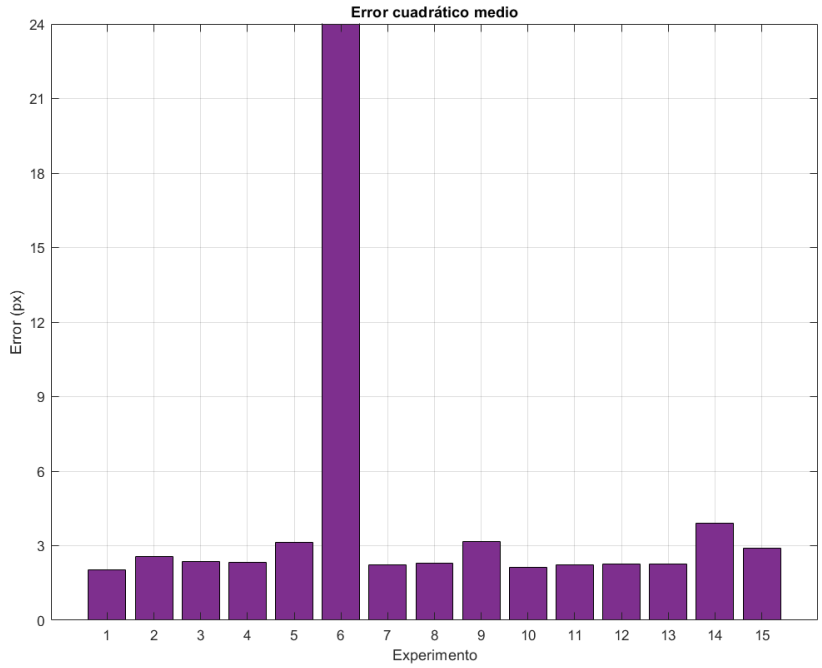

Figura 6-27. Error cuadrático medio 2gdl, estimando longitudes.

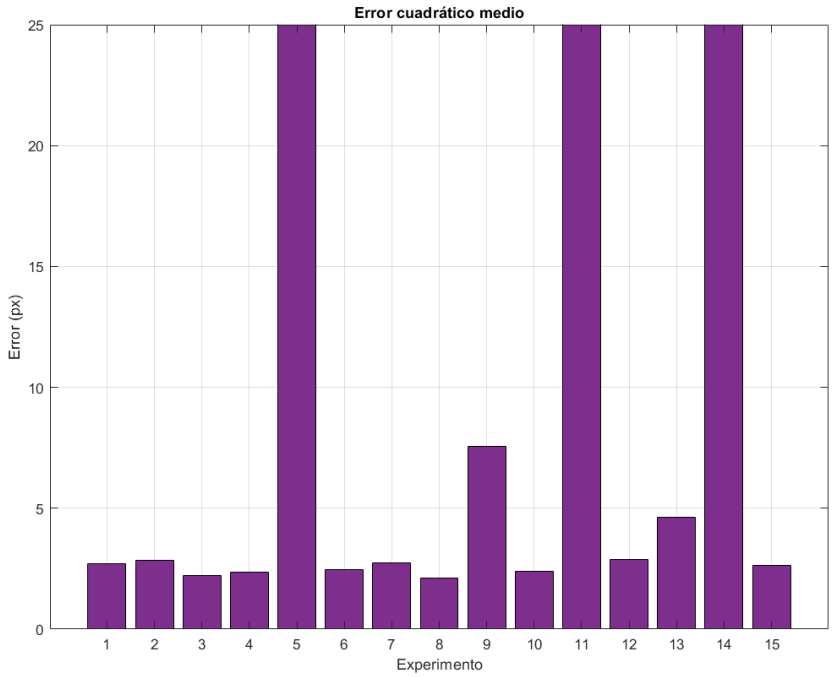

Figura 6-28. Error cuadrático medio 3gdl, estimando longitudes.

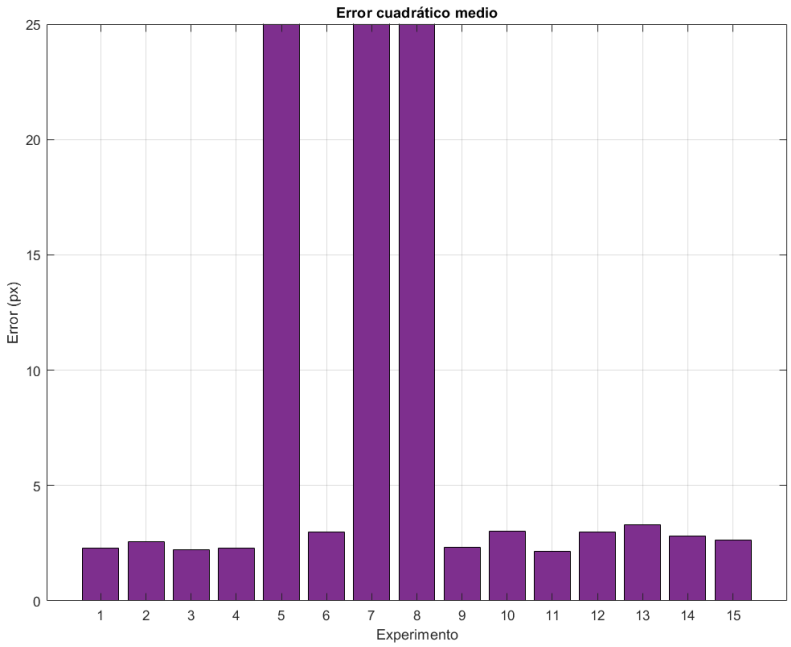

Figura 6-29. Error cuadrático medio 4gdl, estimando longitudes.

Es directo comprobar que se ha reducido bastante el error en píxeles respecto al caso anterior. Esto es debido a que la distancia medida depende de la precisión con la que se tome el centro de los marcadores. Si, como en el caso anterior, se usa una distancia constante para los eslabones, que sería lo real, pero las coordenadas en píxeles de los centros de los marcadores no coinciden totalmente con su centro, la distancia entre estos puede variar según la imagen, provocando que al resolver la cinemática directa y reproyectar, los puntos no coincidan en mayor medida.

Es decir, si se toman unas coordenadas en píxeles que no están totalmente centradas con el marcador en cuestión, al triangular las coordenadas la distancia de un marcador a otro puede no coincidir perfectamente con la longitud real del eslabón, lo cual deriva en un error mayor al resolver la cinemática, puesto que está basada en un modelo con unos datos algo diferentes a los obtenidos. Es por ello por lo que este método es sensible, en parte, a la precisión con la que se detectan e identifican los diferentes marcadores distribuidos por las articulaciones.

6.4.2 Coste computacional

Una manera de medir el coste computacional de la solución desarrollada es medir tanto el número de iteraciones que necesitan los mínimos cuadrados hasta converger en una solución como el tiempo que toma dicho proceos hasta terminar. Por supuesto, uno puede pensar que a mayor número de iteraciones, mayor será el tiempo que tarde en converger, sin embargo, no siempre se da el caso, puesto que aunque se necesite un mayor número de iteraciones hasta hallar un residuo mínimo, es posible que el tiempo tomado para obtener el residuo en cada iteración sea menor. No obstante, es también interesante comprobar como varían estos parámetros a medida que aumenta la complejidad de la articulación.

La librería *Ceres*, utilizada para resolver mediante mínimos cuadrados la ecuación de (5-6), tiene la capacidad de devolver una gran cantidad de datos referentes al mismo, que son útiles para poder estudiar con detalle todos los parámetros y aspectos que intervienen.

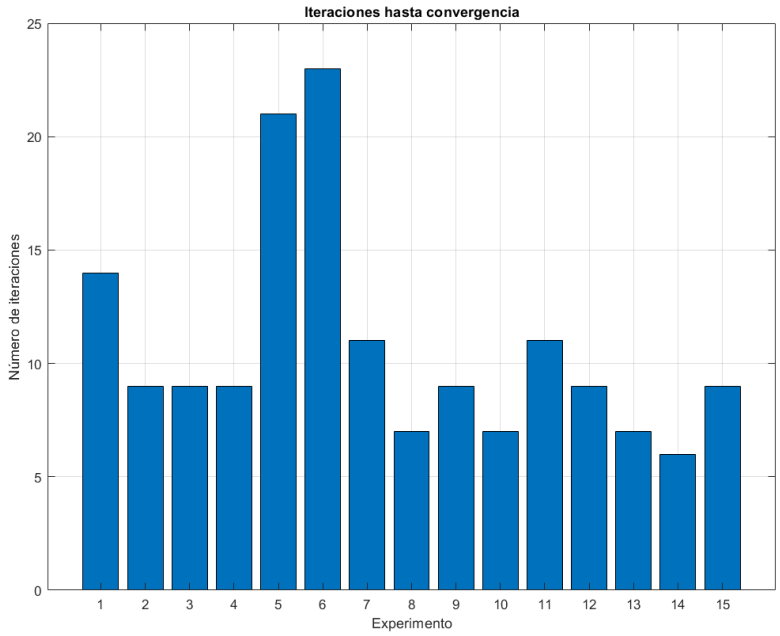

Figura 6-30. Número de iteraciones en cada experimento, para 2 gdl.

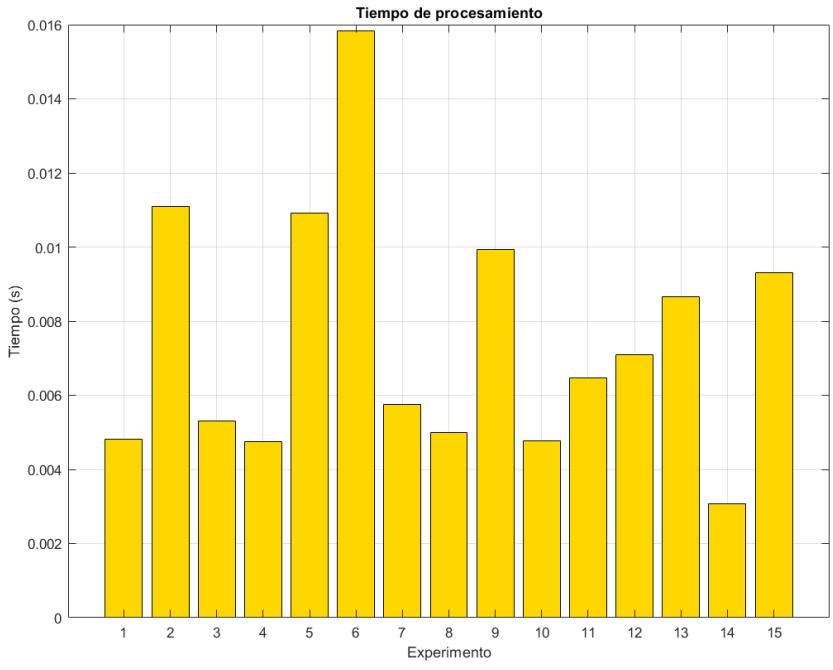

Figura 6-31. Tiempo de procesamiento de cada experimento, para 2gdl.

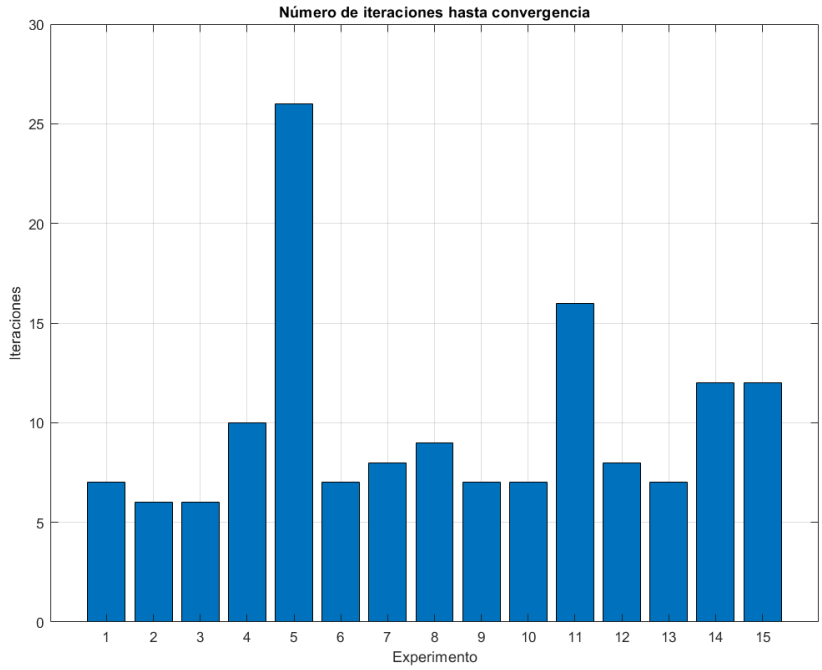

Figura 6-32. Número de iteraciones en cada experimento, para 3gdl

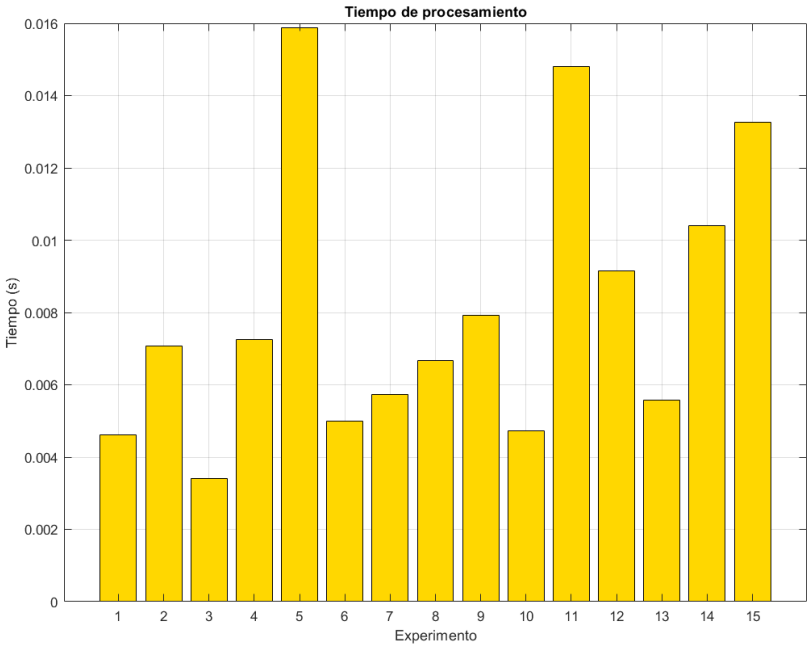

Figura 6-33. Tiempo de procesamiento en cada experimento, para 3gdl

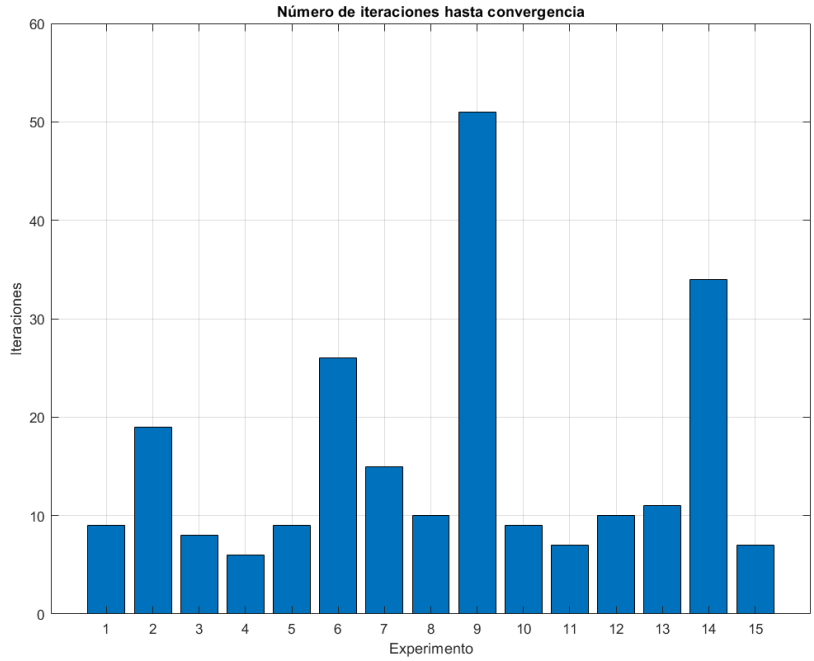

Figura 6-34. Número de iteraciones en cada experimento, para 4gdl

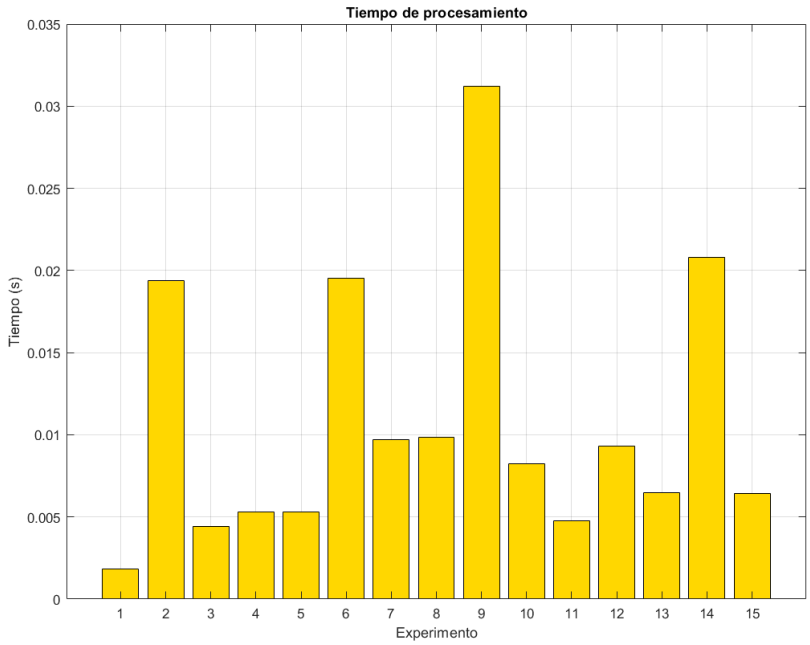

Figura 6-35. Tiempo de procesamiento en cada experimento, para 4gdl.

A la vista de las figuras presentadas, puede comprobarse como existe una relación entre los casos en los que la solución se entiende como errónea, debido al alto error de reproyección mostrado en el apartado anterior, y los máximos relativos de las gráficas. En dichos casos, el algoritmo requiere de mas iteraciones para converger en una solución con residuo mínimo, aumentando el tiempo consumido.

Aunque aún se aleja de los 0.033 segundos por imagen procesada para obtener un mínimo de 30 imágenes por segundo, que es la máxima tasa de imágenes por segundo a la que se puede aspirar con las cámaras disponibles en los laboratorios, es necesario evitar que ocurran este tipo de situaciones para no provocar un cuello de botella entre el procesado y la transmisión de imágenes, traduciéndose en una pérdida de información.

Por otro lado, aunque el número de iteraciones empleadas y tiempo de procesamiento se mantiene más o menos constante entre los experimentos con la articulación de 2 y 3 grados de libertad (aumentando un poco de media para el caso de 3), se produce un aumento significativo para el caso con 4 grados de libertad, tanto en número de iteraciones como en tiempo de procesamiento. Es obvio que una mayor complejidad en las ecuaciones incurre en un mayor tiempo para resolverlas, por lo que sería interesante comprobar hasta que nivel de complejidad sería capaz de resolver a tiempo este sistema.

Estudiamos de nuevo, que ocurre en los casos en los que las longitudes de los eslabones se han estimado geométricamente en lugar de utilizar el valor real de las mismas:

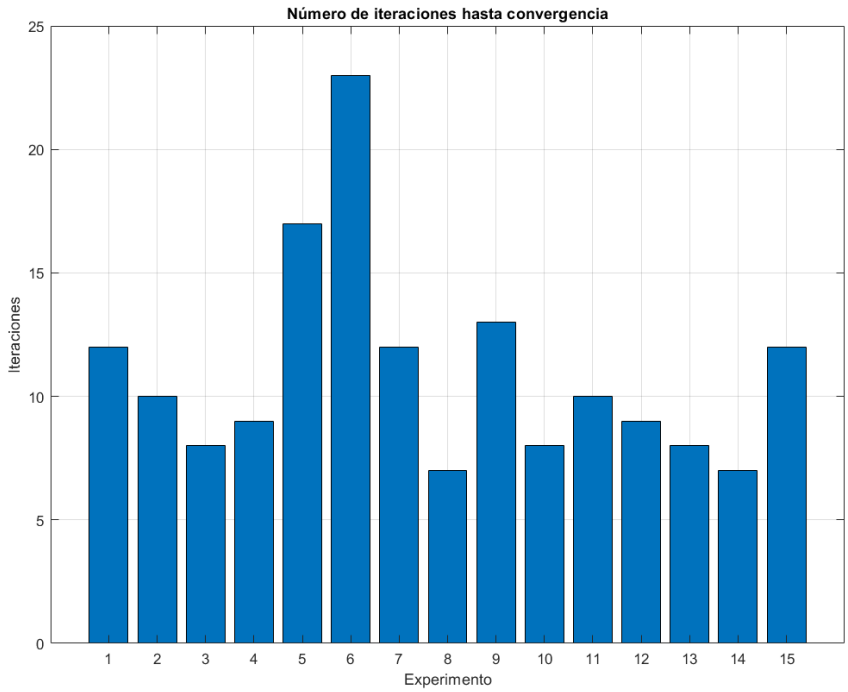

Figura 6-36. Número de iteraciones con longitudes estimadas, para 2gdl.

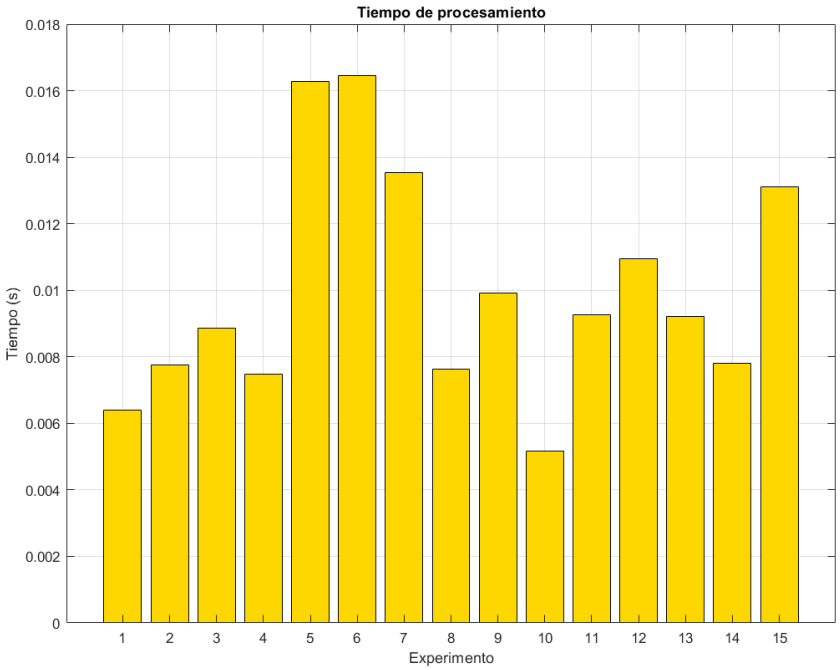

Figura 6-37. Tiempo de procesamiento con longitudes estimadas, para 2gdl.

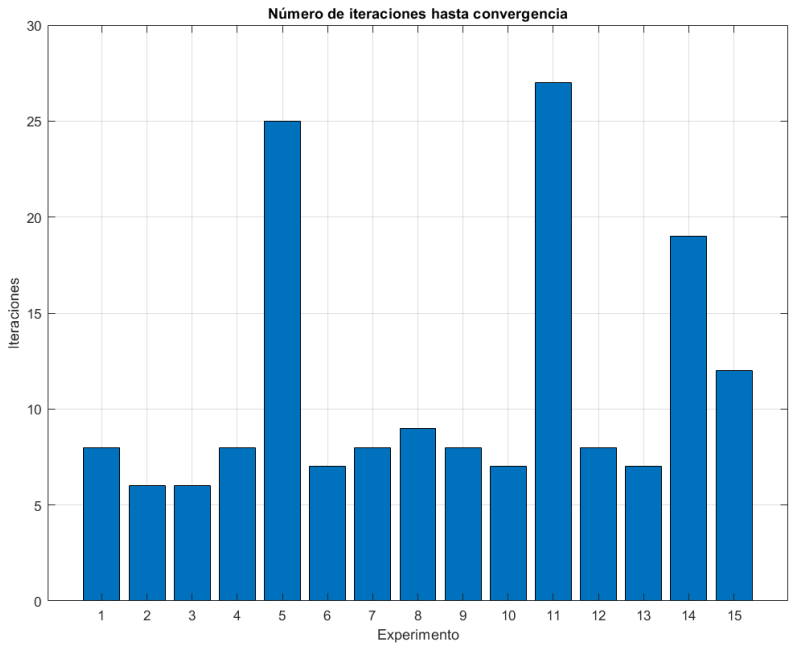

Figura 6-38. Número de iteraciones con longitudes estimadas, para 3gdl.

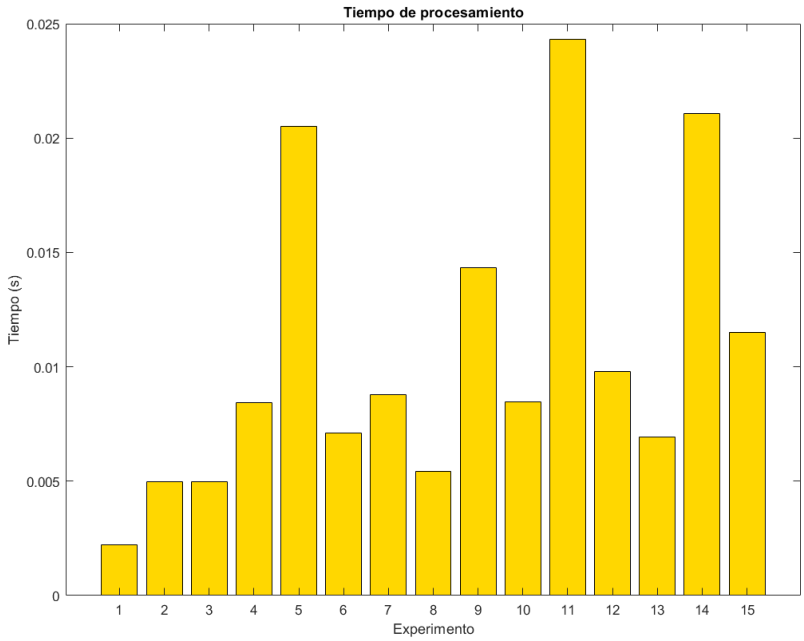

Figura 6-39. Tiempo de procesamiento con longitudes estimadas, para 3gdl.

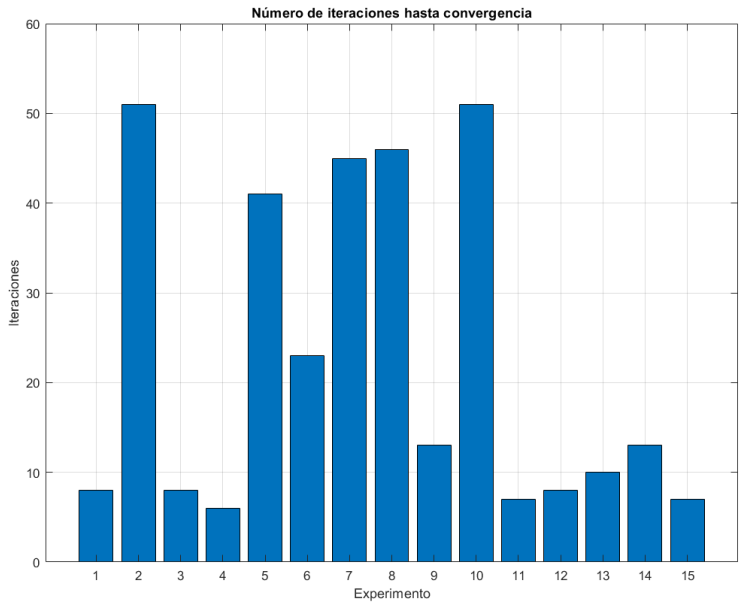

Figura 6-40. Número de iteraciones con longitudes estimadas, para 4gdl.

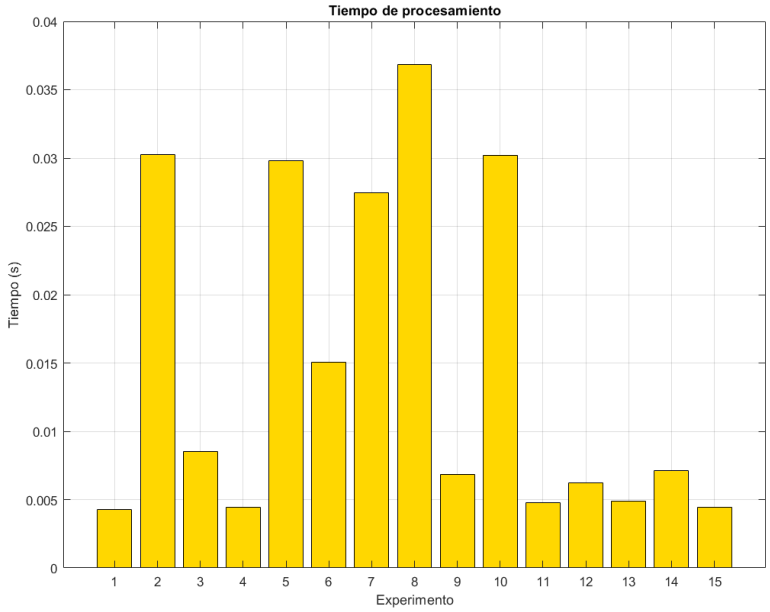

Figura 6-41. Tiempo de procesamiento con longitudes estimadas, para 4gdl.

Si bien en el apartado anterior veíamos como estimando las longitudes tomando las distancias entre los marcadores reducía el error de reproyección en todos los casos, aquí podemos comprobar como han aumentado tanto el número de iteraciones como el tiempo de procesamiento en mayor medida para todos los casos. Una mayor precisión requiere de un mayor coste computacional, por lo que se trata de un aspecto a tener en cuenta a la hora de aplicarlo para un sistema en tiempo real.

Habrá que establecer un equilibrio entre una precisión lo suficientemente buena como para cumplir especificaciones y el número de imágenes por segundo que es posible procesar.

7 CONCLUSIONES

Tras analizar los resultados obtenidos, es directo comprobar que se trata de un método muy prometedor para el seguimiento de articulaciones móviles, simple y totalmente escalable, puesto que tan sólo es necesario conocer las ecuaciones cinemáticas del mismo. Se procede ahora a realizar un resumen de todos los apartados, así como posibles mejoras de cara a una más compleja implementación.

Si bien la plataforma posee 4 cámaras de las mismas características, solo fue posible utilizar tres de ellas al mismo tiempo, debido a limitaciones en el hardware. Si bien este método funciona con un mínimo de 2 cámaras, ya que se basa en la triangulación de las mismas, un mayor número de estas incurre en una mejor estimación de las coordenadas tridimensionas de los marcadores, lo que se traduce en mejores resultados finales.

Por otro lado, durante la captura de imágenes se ha comprobado que, aunque la plataforma posee 4 focos infrarrojos apuntando al centro del volumen de trabajo, la distancia a la que se encuentran provoca que el área efectiva en la cúal se van a ver reflejados correctamente los marcadores de material retrorreflectante es reducida. Utilizar marcadores óptico-activos solucionaría el problema en gran medida, pero aumentaría enormemente el coste. Este es uno de los aspectos que en simulación no se tuvieron en cuenta, puesto que se trató de un entorno controlado.

Respecto al *software*, para los casos de estudio de este trabajo se ha hecho uso de marcadores de diferentes tamaños repartidos por la articulación con el objetivo de facilitar la identificación de cada uno de ellos. No obstante, dado que se trata de un apartado que puede abordarse de diferentes formas, se recomienda desarrollar un método más robusto y escalable de cara a realizar experimentos con articulaciones de mayor tamaño, y un mayor número de grados de libertad. Además, el número de imperfecciones y elementos que dificultaban la detección de los contornos de los marcadores ya sea la por la propia luz de sol o el reflejo de materiales lisos capturados por las cámaras, han requerido de desarrollar formas de filtrado de dichos elementos como un filtrado por circularidad, aprovechando las características geométricas de los marcadores empleados.

Por último, analizando los resultados obtenidos en las diferentes pruebas se observa que el resultado de las mismas varía en función de la distancia a la que se encuentre el cuerpo de las cámaras, así como de la orientación a la que se presente la articulación. Esto puede presentar problemas puesto que hay casos en los que alguna de las cámaras puede no detectar alguno de los marcadores, o que includo la propia articulación cubra alguno de estos. Es necesario por tanto modificar el código para que si alguna cámara no tiene información de algún marcador debido a algún tipo de oclusión, se trabaje con las demás. En el caso de trabajar en tiempo real, es posible trabajar con los resultados de las capturas anteriores.

El siguiente paso lógico sería realizar experimentos con modelos más complejos, extenderlos a cadenas cinemáticas cerradas y de este modo poder documentar hasta que punto (sobre todo, en número de ecuaciones e incógnitas) la resolución mediante mínimos cuadrados es plausible para cadenas cinemáticas que presenten redundancias en sus articulaciones, comprobando si estas satisfacen lo visto en el Apartado 5.4.

Por otro lado, sería conveniente comprobar si todo el proceso aquí descrito puede funcionar sin problemas en tiempo real para una tasa de imágenes relativamente alta, cosa que en principio según lo visto en el apartado anterior, computacionalmente hablando no debería de ser un problema.

REFERENCIAS

[1] «VICON,» [En línea]. Available: https://www.vicon.com/visualization/.

[2] «PHASESPACE,» [En línea]. Available: https://www.phasespace.com/x2e-motion-capture/.

[3] T. E. Lee, J. Tremblay, T. To, J. Cheng, T. Mosier, O. Kroemer, D. Fox y S. Birchfield, «Camera-to-Robot Pose Estimation from a Single Image,» 2020.

[4] G. Bradski, «The OpenCV Library,» *Dr. Dobb's Journal of Software Tools,* 2000.

[5] X. X. LU, «The perspective-n-point (pnp) problem,» *Journal of Physics: Conference Series,* 2018.

[6] M. Persson y K. Nordberg, «Lambda Twist: An Accurate Fast Robust Perspective Three Point (P3P) Solver.,» 2018.

[7] R. H. y A. Z. , Multiple View Geometry in computer vision, Cambridge University Press, 2003.

[8] R. H. y A. Z. , «Epipolar Geometry and the Fundamental Matrix,» de *Multiple View Geometry in computer vision*, 2003.

[9] OpenCV, «OpenCV triangulation,» [En línea]. Available: https://docs.opencv.org/4.5.3/d0/dbd/group__triangulation.html.

[10] R. H. y A. Z. , «The Direct Linear Transformation (DLT) algorithm,» de *Multiple View Geometry*, 2003.

[11] K. Levenberg, «A method for the solution of certain nonlinear problems in least squares,» *Quart. Appl. Math,* 1944.

[12] D. Marquardt, «An algorithm for least squares estimation of nonlinear parameters,» *J. SIAM,* 1963.

[13] G. Inc, «Ceres Solver 2.0,» 2020. [En línea]. Available: http://ceres-solver.org/features.html.

[14] J. N. y S. W. , «Numerical Optimization,» *Springer,* 2004.

[15] T. Technologies, «Teledyne Dalsa,» [En línea]. Available: https://www.teledynedalsa.com/en/home/.

[16] OpenCV, «OpenCV modules,» [En línea]. Available: https://docs.opencv.org/4.5.3/.

[17] Eigen, «Eigen documentation,» [En línea]. Available: https://eigen.tuxfamily.org/dox/.

[18] MathWorks, «Matlab documentation,» [En línea]. Available: https://es.mathworks.com/help/.

[19] J. Domínguez Abascal, M. Acosta Muñoz, R. Chamorro Moreno, V. Chaves Repiso, E. del Pozo Polidoro, J. L. Escalona Franco, D. García Vallejo, C. Madrigal Sánchez, F. J. Martínez Reina, C.

Navarro Pintado, J. Ojeda Granja, E. Reina Romo y J. Vázquez Valeo, Teoría de máquinas y mecanismos, Universidad de Sevilla, 2018.

[20] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, Fundamentos de robótica, Mc Graw Hill, 2007.

[21] J. Denavit y R. Hartenberg, «A kinematic notation for lower-pair mechanisms based on matrices,» *Journal of Applied Mechanics,* 1955.

[22] A. Barrientos, L. F. Peñín, C. Balaguer y R. Aracil, «Algoritmo de Denavit Hartenberg para la obtención del modelo cinemático directo,» de *Fundamentos de Robótica*, Mc Graw Hill, 2007.

[23] M. Pérez Rus, «Experiencias en calibración de sistema multicámara para seguimiento tridimensional de objetos,» Universidad de Sevilla, 2020.

[24] MathWorks, «Camera Calibrator,» [En línea]. Available: https://es.mathworks.com/help/vision/ref/cameracalibrator-app.html.

[25] E. Weisstein, «Rodrigues' Rotation Formula,» Wolfram Research, [En línea]. Available: https://mathworld.wolfram.com/RodriguesRotationFormula.html.

[26] E. Hidalgo Moreno, «Seguimiento de objetos móviles con sistema de posicionamiento multicámara,» Universidad de Sevilla, 2020.

[27] «Roundness - Wikipedia,» Wikimedia Foundation, [En línea]. Available: https://en.wikipedia.org/wiki/Roundness.

[28] S. M. LaValle, «Planning algorithms,» Cambridge University Press, [En línea]. Available: http://planning.cs.uiuc.edu/node102.html.

[29] «Repositorio de imágenes,» [En línea]. Available: https://drive.google.com/drive/folders/12zkPfAhco5FaFMcNhBGG9Ct47PI8cb3V?usp=sharing.