Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática industrial 14 (2017) 455–466

www.elsevier.es/RIAI

Simulación de Plataformas Robóticas de Movimiento para Aplicaciones de Realidad Virtual Mediante Filtros Digitales

Sergio Casas, Cristina Portalés*, Silvia Rueda, Marcos Fernández

Instituto de Robótica y Tecnologías de la Información y la Comunicación, Universitat de València, C/ Catedrático José Beltrán 2, 46980, Paterna, Valencia, España.

Resumen

El uso de plataformas robóticas de movimiento en simuladores de vehículos y aplicaciones de Realidad Virtual es relativamente habitual. Sin embargo, el ajuste de los algoritmos que controlan su funcionamiento, denominados algoritmos de *washout*, no es sencillo y requiere de numerosas pruebas hasta obtener una apropiada fidelidad de movimiento. Disponer de herramientas que permitan simular plataformas de movimiento puede permitir simplificar esta tarea. Es por ello que este trabajo presenta un método para la caracterización y simulación de manipuladores robóticos mediante filtros digitales de segundo orden, sencillo de implementar y ajustar a partir de una caracterización previa. El simulador se construye con el objetivo de permitir la simulación rápida de manipuladores robóticos y se ejemplifica con una plataforma de dos grados de libertad, aunque el método propuesto podría emplearse en otros dispositivos. En las pruebas realizadas se valida la precisión y velocidad de la simulación, concluyéndose que se obtiene una fidelidad satisfactoria y una velocidad de simulación elevada que permite emplear el simulador como sustituto del *hardware* real con algoritmos de *washout*.

Palabras Clave:
Plataformas de movimiento, simuladores, filtros digitales, realidad virtual, robótica, tiempo real.

1. Introducción

Un manipulador robótico es un sistema diseñado para desplazar elementos o coger piezas, habitualmente accionado de forma electro-mecánica. Estos dispositivos se utilizan en múltiples áreas de conocimiento (aplicaciones industriales (Vogel, Fritzsche, & Elkmann, 2016), cirugía (Cleary, 2016), entrenamiento de equipamiento (Lau, Chan, & Wong, 2007), rehabilitación médica (Fung et al., 2004), etc.), además de en aplicaciones de Realidad Virtual (RV) para la generación de claves gravito-inerciales en simuladores de vehículos (Slob, 2008), que es el área en la que se centra este trabajo. Los manipuladores robóticos que se centran en generar movimiento sobre una base rígida móvil se denominan habitualmente plataformas de movimiento y, aunque son esencialmente el mismo tipo de dispositivo, se diferencian de los manipuladores utilizados en otras áreas como la cirugía o la industria de manufactura en su diseño y finalidad.

Cada uso concreto de estos dispositivos conlleva unas particularidades especiales. Mientras que el uso en aplicaciones quirúrgicas requiere de alta precisión o el uso en la industria de manufactura precisa, además, de un apropiado nivel de repetibilidad, el uso en simulación de vehículos y RV suele requerir velocidad y aceleración en detrimento de otras características como la precisión, la repetibilidad o la destreza (*dexterity*).

Para el control de alto nivel de los movimientos de una plataforma robótica en aplicaciones de RV se diseñan unos algoritmos específicos conocidos habitualmente como algoritmos de generación de claves gravito-inerciales (*Motion Cueing Algorithms* – MCA, en inglés) o algoritmos de *washout* (Casas, Olanda, & Dey, 2017). Aunque su uso en RV y simulación de vehículos es relativamente habitual, no es fácil diseñar y ajustar estos algoritmos para que generen las señales apropiadas para hacer funcionar correctamente estos dispositivos (Reid & Nahon, 1988). Por ello, es habitual tener que realizar numerosas pruebas sucesivas para lograr ajustar los algoritmos de *washout* y obtener el comportamiento deseado para el manipulador robótico (Grant & Reid, 1997). La tarea se puede automatizar parcialmente pero requiere de un número aún mayor de pruebas para obtener un comportamiento satisfactorio, como se realiza en Casas, Coma, Portalés, and Fernández (2016). Este proceso es costoso, puesto que el manipulador robótico tiene que ser primero construido (si no existe ya) y luego comprobado exhaustivamente para poder asegurar que se comporta adecuadamente para la aplicación escogida. Sin embargo, durante la fase de pruebas, se pueden realizar movimientos severos y potencialmente dañinos que pueden dañar la plataforma de movimiento, y lo que es más importante, provocar daños humanos en caso de fallo del *software* o *hardware* que se está probando.Para resolver estos problemas, es importante disponer de herramientas que permitan acortar las fases

* Autor en correspondencia.
Correos electrónicos: scasas@irtic.uv.es (Sergio Casas), poricris@uv.es (Cristina Portalés), srueda@irtic.uv.es (Silvia Rueda), marcos.fernandez@uv.es (Marcos Fernández),
URL: irtic.uv.es (Sergio Casas)

http://dx.doi.org/10.1016/j.riai.2017.07.001
Disponible en www.sciencedirect.com

de prueba, y al mismo tiempo reducir costes y riesgos. Una de estas herramientas es un simulador de plataformas de movimiento. Este tipo de simuladores permite reducir los riesgos humanos (al reducir la posibilidad de movimientos indebidos en la fase de pruebas), económicos (permitiendo incluso evaluar la idoneidad de una plataforma antes de construirla) y acortar los plazos (permitiendo realizar pruebas de forma simulada mucho más rápido que de forma real). Este último punto es clave, ya que las simulaciones pueden realizarse más rápidamente que el tiempo real. Esto lo diferencia de las maquetas o prototipos (Ortega & Sigut, 2016), que permiten encontrar fallos de diseño, problemas mecánicos y errores en el proceso constructivo, pero no permiten acelerar las pruebas porque ejecutan *hardware* real.

El objetivo de este trabajó será derivar un método de simulación rápido y eficiente para poder emplear plataformas de movimiento simuladas en la verificación y pruebas de algoritmos de *washout* en simuladores de vehículos y aplicaciones de RV.

El resto del trabajo se organiza de la siguiente forma: el capítulo 2 revisa la literatura relacionada, el capítulo 3 está dedicado a describir el manipulador utilizado para ilustrar el procedimiento y el método de simulación, mientras que el capítulo 4 describe las pruebas y resultados obtenidos con el simulador propuesto. Finalmente, en el capítulo 5 se derivan las conclusiones y se señalan posibles líneas futuras de trabajo.

2. Trabajos Relacionados

La simulación de vehículos tiene una ya larga historia que comienza con los simuladores de vuelo y se remonta a las guerras mundiales (Page, 2000). Aunque las plataformas de movimiento tardaron algunos años más en ser utilizadas para incluir movimiento simulado en este tipo de herramientas y la consideración de éstas como verdaderas aplicaciones de RV sólo se materializó cuando los sistemas de generación gráfica mejoraron en los últimos años del siglo pasado, el uso de plataformas robóticas en simulación es generalizado desde hace más de cinco décadas (Royal-Aeronautical-Society, 1979).

Los primeros MCA diseñados para gobernar plataformas de movimiento en simuladores (en este caso aéreos) fueron propuestos por Schmidt and Conrad (1969) en la *National Aeronautics and Space Administration* (NASA). Estos trabajos fueron revisados y mejorados por Reid and Nahon (1985) y Nahon and Reid (1990) en el *University of Toronto Institute for Aerospace Studies* (UTIAS). Este algoritmo se conoce como el *algoritmo clásico*, y se basa en tres principios básicos para generar movimiento y al mismo tiempo evitar alcanzar los topes del dispositivo: (i) reducir la magnitud de los movimientos; (ii) filtrar las aceleraciones de baja frecuencia (que son las que producen movimientos sostenidos de larga duración) mediante filtros pasa-alta; (iii) intentar simular las aceleraciones sostenidas perdidas, mediante ligeras inclinaciones empleando la técnica de *tilt-coordination* (Groen & Bles, 2004; Reymond & Kemeny, 2000). Más adelante, Parrish, Dieudonne, and Martin Jr (1975) propusieron una modificación del algoritmo clásico conocida como *algoritmo adaptativo*, que intenta ajustar dinámicamente el algoritmo para aprovechar mejor el espacio de trabajo del dispositivo. Poco después, la teoría de control óptimo fue aplicada a este campo para proponer el conocido como *algoritmo óptimo* (Kurosaki, 1978; Sivan, Ish-Shalom, & Huang, 1982). Aunque se han propuesto otros algoritmos más recientemente con cierta aprobación, como en Dagdelen, Reymond, Kemeny, Bordier, and Maizi (2009), el algoritmo clásico es todavía el más utilizado por su versatilidad y efectividad, y porque todavía no se ha llegado a un consenso sobre cuál es la mejor manera de realizar esta tarea. Esto es debido a la falta de una manera estándar de evaluar y ajustar este tipo de algoritmos.

El diseño de manipulador robótico más utilizado en simulación de vehículos es el conocido *hexápodo* Stewart-Gough de seis grados de libertad (GdL) (Stewart, 1965), aunque últimamente se han desarrollado algunos estudios destinados a aplicar plataformas de movimiento de menos GdL, por la reducción de costes que ello conlleva (Lozoya-Santos, Tudon-Martinez, & Salinas, 2017; Mauro, Gastaldi, Pastorelli, & Sorli, 2016; Zhang & Zhang, 2013). El aprovechamiento de todos los GdL en un hexápodo es a veces muy escaso por los reducidos desplazamientos máximos simultáneos posibles, especialmente lineales, en este tipo de diseños. Es, por tanto, necesario considerar cuál es el manipulador más apropiado para cada aplicación abriendo la posibilidad a emplear dispositivos de menos GdL pero más potentes (Casas, Coma, Riera, & Fernández, 2015).

Existen diversas aproximaciones en la literatura para simular manipuladores robóticos, aunque muy pocas con el objetivo de emplear estos dispositivos en aplicaciones de RV. Se pueden encontrar algunos trabajos donde se simulan plataformas de movimiento con otros objetivos, como en Selvakumar, Pandian, Sivaramakrishnan, and Kalaichelvan (2010), en el que se simulan tres manipuladores paralelos de 3 GdL mediante el *software* ADAMS (MSC, 2017) con el objetivo de analizar singularidades y comparar diversas opciones de construcción. Otros autores simulan plataformas de más GdL (Hajimirzaalian, Moosavi, & Massah, 2010). En Y.-W. Li, Wang, Wang, and Liu (2003) se realiza una simulación cinemática y dinámica de un manipulador robótico de 3 GdL, empleando ADAMS y una aproximación de tipo Newton-Euler. Estos trabajos tienen como objetivo analizar el comportamiento de las diversas partes del manipulador y comparar los resultados con las soluciones analíticas, por lo que no buscan realizar un simulador que sustituya al *hardware* en pruebas reales.

Existen otras aproximaciones en las que se simula el comportamiento de una plataforma empleando modelos CAD (Hulme & Pancotti, 2004), permitiendo una intervención "*in-the-loop*", de manera que el simulador no sólo simula el comportamiento del manipulador, sino que puede sustituir al real en cualquier prueba, que es lo que realmente se busca en RV. Este último trabajo emplea una simulación cinemática, por lo que no es suficiente para validar algoritmos de tipo MCA y simula exclusivamente una plataforma *Moog 2000E*.

Una solución que sí permite todo ello, y además permite simular diversos modelos y diseños es la contenida en Casas, Alcaraz, Olanda, Coma, and Fernández (2014), que proporciona un simulador con similar propósito al de este trabajo, cuyo diseño, construcción y validación se describe en detalle. Este simulador emula el comportamiento del manipulador robótico permitiendo una intervención "*in-the-loop*". En dicho trabajo se emplea la dinámica newtoniana-lagrangiana y modelos CAD. Aunque es un simulador bastante preciso, no es todo lo rápido que sería deseable (especialmente si se quiere emplear para realizar miles de pruebas con algoritmos MCA). Además, este simulador no siempre es sencillo de construir, ya que se ha de partir de un diseño CAD de la estructura del manipulador y aplicar diversas técnicas sobre los objetos del mismo para convertirlo en un modelo físico, comprobando previamente la estabilidad de la simulación.

Una alternativa es tratar la simulación como un problema de identificación de sistemas, de modo que se busque un modelo que sea capaz de responder como lo hace el sistema real dadas las mismas entradas, sin tener en cuenta cómo se producen esas

salidas. La literatura de identificación de sistemas es muy extensa, existiendo métodos paramétricos, no paramétricos, lineales y no lineales (L. Ljung, 1999). Una revisión exhaustiva es difícilmente abarcable dada la amplitud de trabajos (Fu & Li, 2013; Gotmare, Bhattacharjee, Patidar, & George, 2017.) En general, el proceso de identificación consiste en: recoger datos (entrada/salida) del sistema, establecer un conjunto de modelos candidatos, elegir un criterio para escoger el mejor modelo, y finalmente validarlo (Lennart Ljung, 1999). Habitualmente se clasifican en tres tipos:

-modelos de tipo *white box*: cuando el modelo es completamente conocido y es posible construirlo a partir de conocimiento previo.

-modelos de tipo *grey box*: cuando cierto conocimiento previo es asumible, pero varios parámetros deben estimarse a partir de datos de entrada/salida observados.

-modelos de tipo *black box*: cuando no existe ningún conocimiento previo. Por tanto, se debe estimar tanto la forma como los posibles parámetros del modelo. Una variante adicional es la *identificación ciega*, cuando ni siquiera la entrada es conocida y es necesario determinar tanto el modelo como la(s) entrada(s) del sistema (Abed-Meraim, Qiu, & Hua, 1997). Esto es más habitual en sistemas de procesamiento de señal.

A diferencia de otras propuestas relacionadas, en este trabajo los autores ponen el foco en buscar un procedimiento de simulación que sea eficiente (aspecto que se suele pasar por alto ya que no siempre se requiere un simulador de tiempo real o incluso más rápido), sin por ello ser impreciso, que incluya el comportamiento dinámico (no sólo el cinemático como algunos), que pueda ser implementado y ajustado fácilmente, y que no se restrinja sólo a un dispositivo. Para centrar el problema, vamos a restringirnos a simular manipuladores paralelos (Küçük, 2012) que son los más habituales en RV, por su alto *payload* y flexibilidad, a pesar de que el espacio de movimiento efectivo suele ser bastante reducido.

3. Material y Métodos

En esta sección se describe en primer lugar el mecanismo utilizado para ejemplificar el método de simulación propuesto en este trabajo, para posteriormente caracterizarlo y simularlo.

3.1. Descripción del Manipulador

Para ilustrar el método de simulación se va a proceder a simular una plataforma robótica de movimiento de dos grados de libertad (GdL) construida con dos motores rotacionales dispuestos en dos cadenas cinemáticas paralelas. Es importante recalcar, sin embargo, que el método propuesto en este trabajo se ejemplifica con dicha plataforma pero no se limita a este mecanismo y sería posible emplearlo en otros sistemas/manipuladores paralelos de más GdL, previa validación que demuestre que el error que se comete es suficientemente pequeño para la aplicación en la que se desee utilizar el dispositivo.

La plataforma de movimiento escogida permite generar movimientos de inclinación frontal (*pitch*) y lateral (*roll*), y es una de las configuraciones de bajo coste más utilizadas porque, a pesar de disponer de sólo dos GdL, permite simular claves gravitacionales en cuatro de los seis GdL empleando la técnica de *tilt-coordination*. Se elige este dispositivo por ese motivo y porque al tener dos GdL permite ejemplificar el método de simulación propuesto de manera sencilla.

Dicho manipulador funciona actuando cada uno de los motores de forma independiente, de tal manera que el ángulo que forma el eje de cada motor con el plano horizontal determina la posición de la pieza unida solidariamente a él, y la combinación de ambos motores resulta en una determinada orientación de la base superior. Se trata de una plataforma 2-RSSU (*Rotational Spherical Universal*), puesto que está compuesta por dos cadenas cinemáticas formadas por una unión rotacional (actuada por el motor), dos elementos móviles (que denominamos *biela* y *pistón* respectivamente) en cuyos extremos se sitúan sendas uniones esféricas, más una unión universal que enlaza con la base superior sobre la que se genera el movimiento.

Al ser un manipulador paralelo, la orientación de un motor influye en la otra cadena cinemática por lo que la máxima inclinación del ángulo de *pitch* (giro alrededor del eje X) no se producirá cuando la inclinación lateral (*roll*, giro sobre el eje Y) sea máxima, y viceversa. La Figura 1 muestra el diseño de la plataforma de movimiento. Sobre la base superior, denominada *base móvil*, se sitúa el usuario de la simulación, habitualmente reproduciendo el asiento y *cockpit* del vehículo simulado. El movimiento de la base superior está limitado por una unión universal (en color verde en la Figura 1), que se toma como centro del sistema de referencia. La Figura 2 muestra este sistema de coordenadas (ocultando la base móvil). Los ángulos se miden respecto de la horizontal en sentido anti-horario, tomando como cero la posición que se observa en el motor izquierdo.

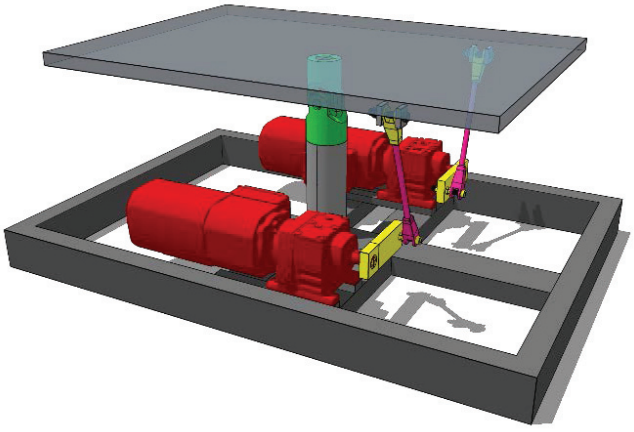

Figura 1: Plataforma de movimiento de dos GdL utilizada en este trabajo.

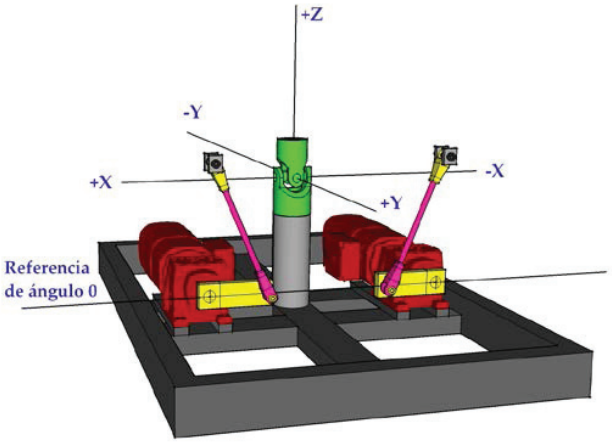

Figura 2: Sistema de coordenadas utilizado para el dispositivo 2-RSSU.

La Tabla 1 resume las características cinemáticas básicas de la plataforma en base a los desplazamientos de mínimo a máximo valor (excursión). Dicho dispositivo emplea motores de corriente alterna *SEW Eurodrive*, cuyo sistema de control es propietario y, por tanto, viene ajustado de fábrica. El control de los motores queda completamente al margen de este estudio.

Tabla 1: Características de la plataforma de movimiento de dos GdL.

GdL	Rango de excursión (º)	Extensión de excursión (º)
pitch	[-30.27, 30.27]	60.54
roll	[-26.38, 26.38]	52.76

Más allá de las consideraciones mecánicas del manipulador concreto que se utilice, que no son de interés en este trabajo, una plataforma robótica se comporta como un sistema cerrado que toma como entradas el desplazamiento (angular o lineal) de los actuadores y proporciona como salida la posición y orientación cartesianas actuales de su base móvil (o extremo final de la cadena cinemática). Por tanto, a este nivel, el manipulador es un sistema MIMO (*Multiple Input Multiple Output*). Para plataformas Stewart de seis GdL sería un sistema 6x6, mientras que para una plataforma más reducida como esta, se trataría de un sistema 2x2 (puesto que tiene dos motores, y debe proporcionar los seis GdL de salida, aunque cuatro de ellos valen cero). En algunas aplicaciones es necesario conocer la posición y orientación de la plataforma con respecto a los desplazamientos de los actuadores. Sin embargo, para su uso con algoritmos de *washout* en aplicaciones de RV, lo único interesante es relacionar los GdL (lineales y angulares) de la base móvil relativos a una posición de reposo (no importa la posición absoluta sino el desplazamiento relativo), con respecto a los GdL demandados. Dicho de otro modo, el manipulador puede verse como un sistema que toma como entrada los GdL deseados para la plataforma y devuelve como salida los GdL actuales del manipulador, ignorando los desplazamientos de los actuadores. Para ello, debe resolverse la cinemática inversa del manipulador de modo que a partir de los GdL deseados se conozcan las posiciones de los actuadores que inducen dichos GdL, teniendo además en cuenta que se deben limitar las peticiones de entrada en función de las restricciones espaciales/físicas de la plataforma. La Figura 3 muestra cómo se emplea un manipulador paralelo de dos GdL para aplicaciones de RV. Las entradas son los GdL deseados (g_{e1}, g_{e2}) para la plataforma. Éstos son convertidos por la cinemática inversa en posiciones deseadas para los actuadores (α_1, α_2). Éstas posiciones son, a su vez, convertidas en comandos de movimiento (torque/corriente eléctrica) por el sistema de control, que permite además conocer el estado actual de las posiciones de los actuadores (β_1, β_2). Con esta información es posible calcular el estado actual de la plataforma en forma de GdL actuales (g_{s1}, g_{s2}) haciendo uso de la cinemática directa. En condiciones ideales g_{ei} y g_{si} coincidirían en todo momento. Sin embargo, las limitaciones físicas de los actuadores y controladores hacen que haya un cierto retraso y/o atenuación. La extensión a un mayor número de GdL es trivial, aumentando simplemente el número de variables.

Por tanto, si se desea realizar un simulador de plataforma para su uso en RV, se debe implementar un sistema que disponga de tantas entradas como GdL (representan los GdL deseados para la plataforma) y proporcione como salida, al menos, los GdL actuales del manipulador robótico, pudiendo proporcionar otras salidas accesorias como una visualización de los elementos móviles del manipulador, o la posición/orientación de los actuadores.

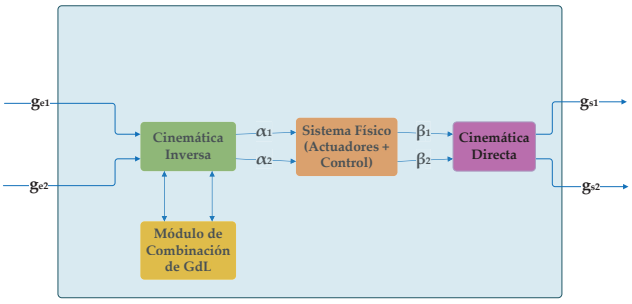

Figura 3: Esquema de uso de plataformas de movimiento de dos GdL en RV.

3.2. Caracterización del Manipulador

Para simular un sistema o fenómeno es necesario caracterizar adecuadamente su comportamiento con anterioridad. La caracterización de una plataforma robótica de movimiento se puede realizar desde distintos puntos de vista, desde análisis estáticos y cinemáticos de las configuraciones alcanzables, hasta análisis dinámicos del comportamiento en movimiento. La literatura contiene diversos análisis de numerosas plataformas y manipuladores robóticos con distintos objetivos, en los cuales se analizan posiciones, velocidades, aceleraciones, precisiones, repetibilidad, singularidades, etc. (Cao, Gosselin, Zhou, Ren, & Ji, 2013; S. J. Li & Gosselin, 2012). En simulación de vehículos y RV nos interesa analizar retrasos, velocidades y aceleraciones, pero no tanto otros aspectos como la precisión, la destreza o la repetibilidad. Ser capaces de realizar movimientos rápidos permite simular el comportamiento de vehículos sin introducir retrasos evitando confundir al usuario de la aplicación de RV, de modo que éste perciba las claves gravito-inerciales de forma sincronizada con las claves visuales. Además, ello reduce la distorsión de fase y de amplitud (Sinacori, 1977) entre los movimientos experimentados y los esperados, aumentando la fidelidad del movimiento (Hodge, Perfect, Padfield, & White, 2015) y la validez física y perceptual del mismo (Reymond & Kemeny, 2000).

El problema fundamental a la hora de entender el comportamiento de una plataforma de este tipo es que los GdL de salida (GdL actuales) dependen de todas las entradas (GdL deseados) simultáneamente por lo que no es sencillo encontrar una función que represente esta interdependencia, y además incluya las inercias y el comportamiento dinámico de la plataforma.

Por ello, se analizará primero el comportamiento dinámico de la plataforma analizando cada GdL por separado, fijando el otro (u otros si hubiera más de dos GdL) a cero. Todas las pruebas presentadas se realizan con señales digitales muestreadas con un periodo de 0.02 s (50 Hz).

Para ello, se emplea un sistema de medición basado en cámaras mientras se aplican distintos movimientos sobre el manipulador electromecánico, que contará con un marcador sobre su base móvil. El sistema empleado es *Natural Point Optitrack* (Optitrack, 2017), un sistema de seguimiento óptico pasivo muy robusto y estable, que emplea marcadores esféricos revestidos de un material reflectante del espectro infrarrojo de la luz. Esto permite obtener la posición real del marcador, y por tanto hacer *tracking* de la base móvil del manipulador robótico. El *software* de las cámaras se encarga de todos los cálculos necesarios para hacer el seguimiento y proporciona, además, herramientas para su calibración. Esto permite conocer con mucha exactitud los GdL de la plataforma sin necesidad de resolver la cinemática directa del manipulador y sin

depender del sistema de control de los motores. El error de este sistema óptico, cuya infraestructura puede verse en la Figura 4, es inferior a los 2 mm, y la frecuencia de captura es de 100 Hz.

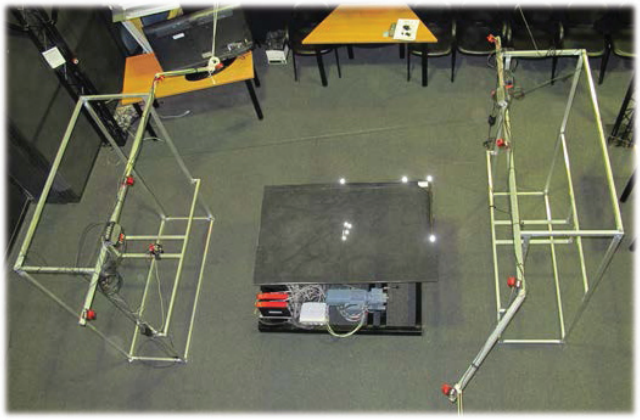

Figura 4: Sistema de *tracking* óptico con marcadores reflectantes.

Para analizar el comportamiento en el dominio del tiempo, lo más sencillo es utilizar señales de prueba como la onda sinusoidal o la onda cuadrada. Como es lógico, la amplitud de los movimientos solicitados es clave, porque los motores del manipulador robótico tienen una potencia limitada y no pueden realizar determinados movimientos de largo recorrido en tiempos muy cortos.

La Figura 5 muestra el comportamiento comparado del *pitch* de salida con respecto al *pitch* deseado a la entrada, cuando éste varía con dos patrones distintos (entradas 1 y 2) en forma de onda cuadrada. La entrada 1 se corresponde con una onda cuadrada de 0.125 Hz de frecuencia y 30º de amplitud. La salida 1 muestra la respuesta de la plataforma ante la entrada 1 (sólo se actúa en el ángulo de *pitch*; el *roll* de entrada es nulo). La entrada y salida 2 se corresponden con otra prueba diferente: un movimiento de *pitch* en forma de onda cuadrada amortiguada de 0.25 Hz de frecuencia, 30º de amplitud inicial, y una constante de atenuación de 0.06.

Estas señales permiten conocer la respuesta transitoria del sistema ante cambios bruscos. En nuestro caso, nos proporciona además información sobre cuánto tiempo emplea la plataforma en cumplir una determinada orden. Se puede observar en la Figura 6 (que es una ampliación de la Figura 5) que el manipulador tiene una inercia inicial grande. Éste tarda en torno a 80 ms (doble flecha roja en la Figura 6) en comenzar a realizar el movimiento que se le ordena. Se observa que este retraso es prácticamente independiente de la amplitud del movimiento por lo que lo identificamos como una latencia fija que representa el tiempo que tarda la señal deseada en ser convertida en órdenes físicas para los motores (incluye el cálculo de la cinemática inversa, el empaquetamiento y envío de la información en el protocolo MODBUS que emplea la plataforma, más el algoritmo de control que utilizan los motores) más la pequeña latencia de las cámaras.

Más allá del retraso inicial, se observa que una excursión completa tarda poco más de 1 s (doble flecha verde en la Figura 6), por lo que la velocidad media del manipulador en ese tramo es de aproximadamente 60 º/s. Para amplitudes más pequeñas, el tiempo decrece sólo ligeramente. De hecho, para una semiexcursión, el tiempo necesario es de algo menos de 800 ms (doble flecha azul en Figura 6), por lo que la velocidad sería de algo menos de 40 º/s. El hecho de que una semiexcursión sea sólo ligeramente menos costosa que una excursión pone de manifiesto que el principal problema de estos dispositivos es el inicio y frenado del movimiento debido a su inercia. En una excursión se observan tres fases: una fase de inicio y aceleración (la más costosa), una fase corta de movimiento (a elevada velocidad) y una fase de frenado, para evitar que se alcance el objetivo a demasiada velocidad superándolo (*overshoot*), lo cual obligaría a realizar oscilaciones correctivas alrededor de la posición de destino para converger a ella. La Figura 6 muestra que el comportamiento de la plataforma de movimiento es bastante amortiguado, y se parece a la respuesta transitoria de un filtro pasa-baja amortiguado, suavizando la onda cuadrada de entrada de una forma similar a como lo hace este tipo de filtros (Winder, 2002). El comportamiento concreto dependerá de cómo se realice el control de los motores, pero para este tipo de aplicaciones de RV es deseable que el control no introduzca demasiadas oscilaciones que puedan generar claves gravito-inerciales falsas. El comportamiento para el movimiento *roll* es muy similar, con tiempos de excursión muy similares.

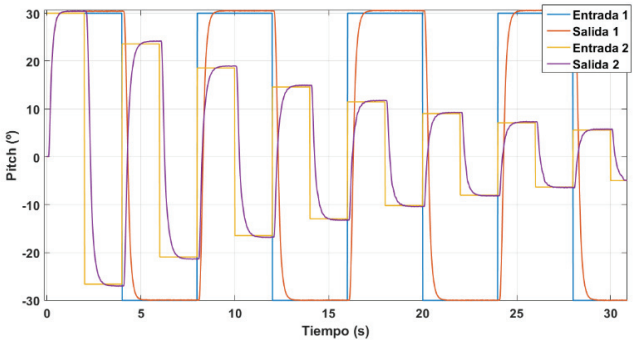

Figura 5: Respuesta ante diversas señales de onda cuadrada (*pitch*).

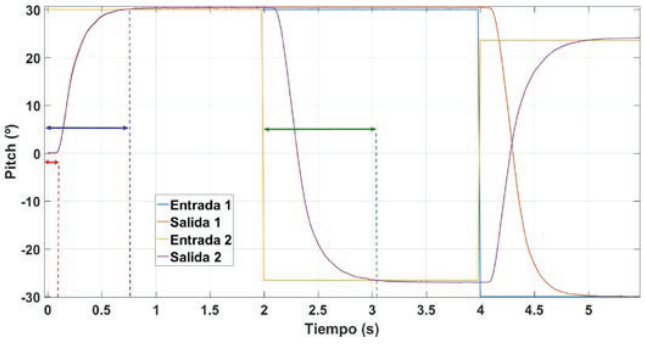

Figura 6: Detalle (inicio) de la Figura 5, e intervalos de tiempo de relevantes.

En la Figura 7 podemos ver la respuesta comparada del *roll* (fijando el *pitch* a cero) ante dos entradas sinusoidales diferentes de amplitud decreciente. Este tipo de pruebas nos muestra cómo se comporta el manipulador ante cambios suaves de las entradas. Se puede observar que la plataforma es capaz de seguir la entrada con una cierta atenuación y desfase. Este desfase y esta atenuación, como se observa, aumentan con la frecuencia y amplitud de la señal. La respuesta del *pitch* es muy similar.

Para analizar con más detalle el comportamiento frecuencial de cada GdL, en lugar de probar múltiples ondas sinusoidales de diferente frecuencia, es mucho más práctico emplear una señal sinusoidal de frecuencia creciente (*chirp*). Este tipo de señales condensan en una única señal múltiples frecuencias proporcionando información muy valiosa. Para esta prueba se emplea una señal *chirp* con modificación exponencial de la fase.

La ecuación de una señal *chirp* es similar a la de una onda sinusoidal, con la particularidad de que la fase no se incrementa linealmente con el tiempo, por lo que la señal *chirp* se puede expresar como:

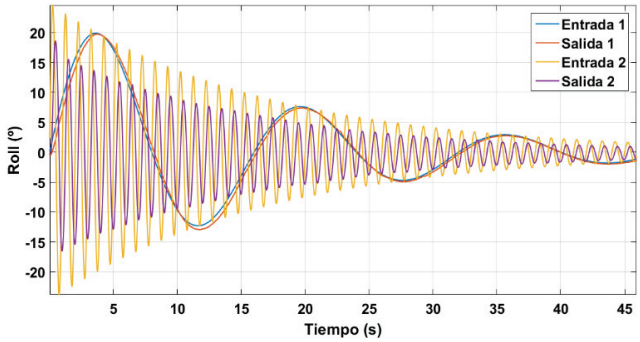

Figura 7: Respuesta ante dos señales sinusoidales atenuadas (*roll*).

$$y(t) = A \cdot sin(\phi(t)) + \delta \quad (1)$$

siendo A la amplitud, δ el posible *offset* respecto de cero, y $\phi(t)$ la fase de la señal, que va modificándose de forma no lineal:

$$\phi(t) = 2\pi f_0 \left(\frac{r^t - 1}{\ln(r)} \right) + \phi_0 \quad (2)$$

siendo f_0 la frecuencia inicial (en herzios), r la tasa de modificación de la frecuencia, y ϕ_0 la fase inicial.

Por tanto, la frecuencia instantánea, $f_i(t)$, en un determinado momento se puede calcular derivando la fase con respecto al tiempo:

$$f_i(t) = \frac{\partial \phi(t)}{\partial t} = f_0 \, r^t \quad (3)$$

La Figuras 8 y 9 muestran el comportamiento del *pitch* ante una señal *chirp*, con $r = 1.1$, $f_0 = 0.01$, $\phi_0 = 0$, $\delta = 0$. En la Figura 8, $A = 30$, mientras que en la Figura 9, $A = 20$. Se puede observar que el manipulador robótico se comporta como un filtro pasa-baja, ya que para frecuencias bajas reproduce perfectamente la señal, y según aumenta la frecuencia, la salida se va atenuando y desfasando cada vez más. Para amplitudes menores, el comportamiento es similar, aunque la atenuación sucede más tarde y es ligeramente más abrupta. Para el *roll* sucede lo mismo, aunque la atenuación parece ser algo mayor en todas las amplitudes.

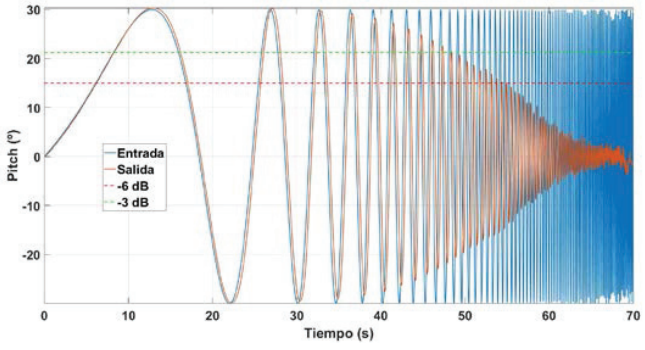

Figura 8: Respuesta ante una señal *chirp* (*pitch*) – A = 30º.

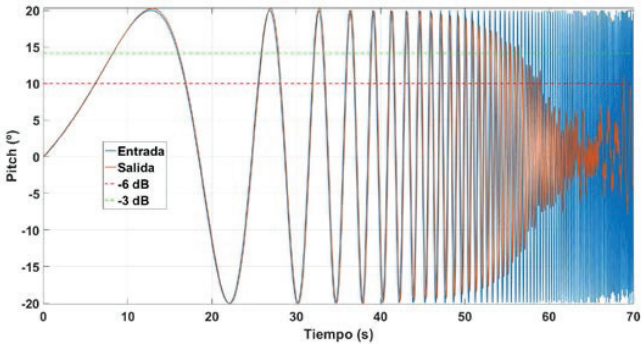

Figura 9: Respuesta ante una señal *chirp* (*pitch*) – A = 20º.

La Tabla 2 resume las frecuencias a las que se produce una atenuación de 3 dB y 6 dB respectivamente. Dichas frecuencias se calculan empleando (3), tomando como tiempo aquél en el que la señal se atenúa a un valor lo más cercano posible a 3 dB y 6 dB respecto de la magnitud inicial. Se utilizan señales *chirp* de 70 segundos de duración con $r = 1.035$ y $f_0 = 0.5$ para que el aumento de frecuencia sea más suave y el cálculo se pueda hacer con más precisión. Dado que la atenuación fluctúa por imperfecciones mecánicas y cuanto más alta es la frecuencia, menos estable es la respuesta, estos valores son aproximados, aunque nos dan una idea bastante exacta de cómo se comporta el manipulador paralelo. Para valores de amplitud muy bajos (5º o menos) el cálculo no es demasiado fiable porque la respuesta se vuelve algo inestable a altas frecuencias.

Como se puede comprobar, el manipulador robótico se comporta como un filtro pasa-baja que reproduce las entradas que se le introducen con un cierto desfase y atenuación. Si bien el valor de las frecuencias f_{-3dB} y f_{-6dB} no es igual para todas las amplitudes (lógicamente a mayor amplitud solicitada menor es la frecuencia necesaria para producir una cierta atenuación), éstas se mueven en un rango relativamente estrecho. Si asumimos que cada GdL se comporta como un filtro pasa-baja independiente, de frecuencia natural fija, cometeremos un pequeño error, pero captaremos la esencia del comportamiento del sistema.

Tabla 2: Frecuencias (en Hz) que producen una atenuación de 3 dB (70.7 %) y 6 dB (50 %), según la amplitud.

	Pitch		*Roll*	
A	f_{-3dB}	f_{-6dB}	f_{-3dB}	f_{-6dB}
30	1.59	2.00	-	-
25	1.83	2.28	1.59	2.03
20	2.00	2.45	1.86	2.10
15	2.42	2.76	2.21	2.56
10	2.86	2.90	2.59	2.76

Sin embargo, no hay que olvidar que todas estas pruebas analizan el comportamiento individual de cada GdL, asumiendo que el resto están inactivos. Es por tanto importante comprobar qué ocurre cuando se modifican simultáneamente varios GdL con diferentes patrones de entrada.

Las Figuras 10 y 11 muestran el comportamiento de la plataforma cuando se varía simultáneamente el *pitch* y el *roll* de entrada. Como se puede observar, la salida final es difícilmente caracterizable a simple vista, ya que cuando ambos GdL alcanzan valores lejos de la posición de reposo, la combinación de ambos

obliga a reducir la extensión de los mismos para poder cumplir las restricciones físicas espaciales de los elementos móviles del manipulador. El comportamiento del sistema está, por tanto, completamente influenciado por la estrategia de combinación de GdL que se aplique cuando la combinación deseada exceda la capacidad física de la plataforma. Lo habitual es reducir proporcionalmente la petición en función de la combinación máxima físicamente realizable, y es la estrategia que se utiliza en este trabajo (el algoritmo se muestra en la sección siguiente). Por tanto, es necesario tener en cuenta esta estrategia, y la cinemática inversa del dispositivo (puesto que se utiliza para comprobar qué combinaciones son alcanzables y cuáles no), a la hora de realizar un simulador que emule el comportamiento del sistema completo.

En cualquier caso, no parece sencillo identificar una función de transferencia que represente al sistema completo, de modo que podamos realizar una simulación rápida discreta. Por ello, se tratará de encontrar la manera de reproducir el comportamiento de cada GdL individual, aplicando previamente las restricciones cinemáticas impuestas por la combinación de GdL, como se detalla en la próxima sección.

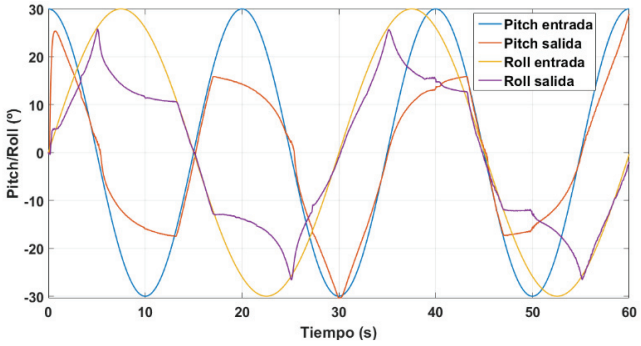

Figura 10: Respuesta ante cambios sinusoidales simultáneos de *pitch* y *roll*.

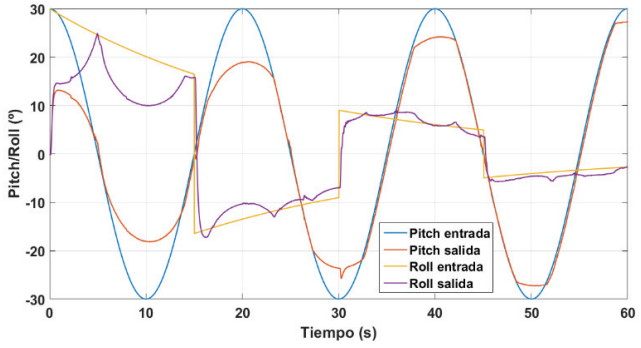

Figura 11: Respuesta ante cambios simultáneos de *pitch* (onda sinusoidal) y *roll* (onda cuadrada atenuada). Se observa el acoplamiento entre las salidas.

3.3. Esquema de Simulación Propuesto

Una vez realizada la caracterización se propone un esquema de simulación para el manipulador. Dicho esquema se basa en la suposición de que el sistema MIMO completo se puede descomponer en cada uno de sus GdL simulando cada uno de ellos como si fuera un sistema SISO (*Single Input Single Output*), empleando para ello filtros pasa-baja independientes de orden 2 (puesto que son más generales que los de orden 1 y permiten afinar más la simulación sin requerir un tiempo de cálculo muy elevado), con una cierta latencia, ya que el comportamiento individual de cada GdL, observado en la caracterización, se asemeja a éste.

Antes de implementar estos sistemas, se debe resolver el problema de combinar varios sistemas SISO para resolver el sistema MIMO completo. Para ello, se propone aplicar el módulo de combinación de GdL para reducir las entradas a valores alcanzables por la plataforma. Este módulo es el que se encarga de simular las no linealidades entre las entradas y las salidas, de manera que se pueda tratar cada entrada como un canal separado de los demás. Esto se correspondería con una estrategia de tipo *grey box*, ya que asumimos una cierta estructura para el modelo debido a un conocimiento previo. En efecto, una regla básica en identificación de sistemas es no *estimar lo que ya se conoce* (Sjöberg et al., 1995). En este sentido, la cinemática de una plataforma de movimiento puede ser conocida, y el comportamiento dinámico obedece a las leyes físicas por lo que sabemos que la inercia juega un papel fundamental. La Figura 12 muestra el esquema de simulación propuesto (la extensión a más GdL es trivial). (g_{e1}, g_{e2}) son los GdL deseados para la plataforma. Estos GdL deben ser comprobados primero para asegurarse que son simultáneamente posibles. Para ello, se consulta a la cinemática inversa. Si no lo fueran, se reducen los GdL (proporcionalmente a la petición original) hasta llegar a (g'_{e1}, g'_{e2}), que representa los GdL alcanzables que se le pueden solicitar. La salida (g_{s1}, g_{s2}) representa el estado actual de la plataforma, en forma de GdL. La salida de cada GdL se forma aplicando un retraso y un filtro pasa-baja sobre la entrada proporcionada. El algoritmo de combinación de GdL es el siguiente:

Algoritmo **CombinacionGdL**

Entradas:
ge : vector [1..numGdL] de R // GdL de entrada (objetivo/deseados)
tols : vector [1..numGdL] de R // tolerancia del resultado para cada GdL

Parámetros:
numGdL = 6

Auxiliar:
i : N
valido, terminar : {cierto, falso}
inis, fins, rangos, medios : vector [1.. numGdL] de R

Salidas:
gs : vector [1.. numGdL] de R // GdL realizables/alcanzables

Algoritmo:

```
valido = cinematicaInversa(ge);
si (valido) entonces
   gs = ge;
sino
{
   ini = vectorNulo();
   fin = ge;

   hacer
   {
      desde (i = 0 hasta numGdL) hacer
      {
         medios[i] = (inis[i] + fins[i])*0.5;
         gs[i] = medios[i];
      }

      valido = cinematicaInversa(gs);
      terminar = cierto;
```

```
        desde (i = 0 hasta numGdL) hacer
        {
          si (valido) entonces
            inis[i] = medios[i];
          sino
            fins[i] = medios[i];

          rangos[i] = | fins[i] - inis[i] |;
          terminar = terminar ^ (rangos[i] < tols[i]);
        }
      }
    }
    hasta que (terminar)

    Fin de CombinacionGdL
```

Este algoritmo trata de buscar la combinación de GdL máxima que la cinemática inversa permita, manteniendo al mismo tiempo la proporcionalidad entre los GdL suministrados a la entrada, de forma que los GdL solicitados sean alcanzables. El bucle externo realiza una búsqueda dicotómica múltiple, y el bucle interno se encarga de comprobar que todos los GdL son alcanzables, actualizando los rangos de búsqueda. La resolución de las ecuaciones de la cinemática inversa de este dispositivo concreto se puede consultar en el Apéndice A.

Este esquema no se corresponde con el funcionamiento exacto de la plataforma, ya que es una simplificación, que inevitablemente introducirá diferencias con respecto a la plataforma original, por dos motivos: (i) el manipulador se comporta como un filtro pero de frecuencia natural y amortiguamiento ligeramente variables según la amplitud, cosa que simplificará a un filtro de orden 2 de parámetros fijos; (ii) la separación de la simulación en tantos canales como GdL se realiza de forma estática (empleando la cinemática inversa para resolver la combinación de GdL) obviando la influencia dinámica que pueda haber entre unas entradas y otras. La hipótesis, que se comprobará a continuación, es que el error cometido con esta suposición será lo suficientemente pequeño como para poder utilizar el simulador como un sustituto de la plataforma real en aplicaciones de RV.

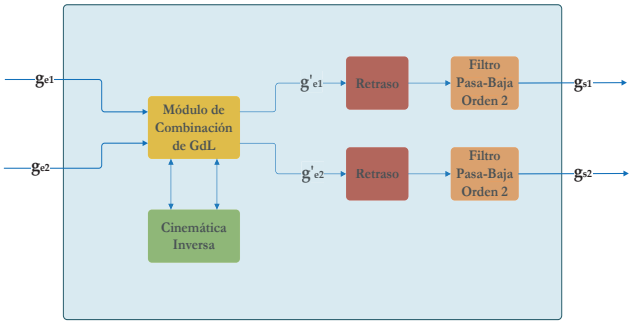

Figura 12: Esquema de simulación propuesto para un sistema de dos GdL.

Una vez separado el problema MIMO en varios problemas SISO, falta implementar éstos mediante filtros, como se ha propuesto. Un filtro pasa-baja de orden 2 con un retraso de t_d segundos tiene la siguiente función de trasferencia en forma laplaciana (Paarmann, 2001):

$$H(s) = e^{-st_d}\frac{G\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{4}$$

donde s es la variable laplaciana, G la ganancia, ω_n la frecuencia natural, y ξ el factor de amortiguamiento. Si se realiza una implementación discreta, el retraso/latencia se puede implementar fácilmente retrasando el número de periodos adecuado la señal de entrada. Si este número de periodos es Δ, la expresión discreta del módulo de retraso sería:

$$y_t = x_{t-\Delta} \tag{5}$$

siendo y_t la salida del módulo de retraso en el instante actual y x_{t-j} su entrada en un instante previo retrasado j periodos. Por tanto, el filtro se aplicaría tomando como entrada la salida del bloque de retraso, y se puede expresar simplemente como:

$$H(s) = \frac{G\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \tag{6}$$

Para implementarlo en un computador, se discretiza el sistema mediante la aplicación de la transformación bilineal de Tustin (Rorabaugh, 1993),

$$s = \frac{2}{T}\cdot\frac{1 - z^{-1}}{1 + z^{-1}} \tag{7}$$

siendo T el periodo de muestreo de la señal, y z la variable compleja de la transformada Z. Para implementar digitalmente de manera eficiente el sistema, se sustituye s por z y se realiza una factorización de términos, llegándose a la siguiente expresión discreta:

$$\begin{aligned} y_t = {} & [Ay_{t-2} + C(x_t + 2x_{t-1} + x_{t-2}) \\ & - B(2y_{t-1} + y_{t-2}) \\ & + 4(2y_{t-1} - y_{t-2})]D \end{aligned} \tag{8}$$

con

$$A = 4\xi\omega_n T \tag{9}$$

$$B = {\omega_n}^2 T^2 \tag{10}$$

$$C = BG \tag{11}$$

$$D = \frac{1}{(A + 4B)} \tag{12}$$

siendo y_t la salida del filtro en el instante actual, x_{t-j} e y_{t-j} la entrada y la salida, respectivamente, en un instante previo retrasado j periodos. Se trata de un sistema IIR (*Infinite Impulse Response*) con realimentación de la salida, ya que ésta depende de las entradas actuales y pasadas, y de las salidas en instantes anteriores. El simulador se implementa en lenguaje C para una mayor eficiencia.

4. Pruebas y Resultados

Tras la caracterización, es momento de comprobar que el modelo propuesto puede constituir un simulador de la plataforma de movimiento. Habiéndose elegido la forma del modelo, falta solamente identificar valores para sus parámetros.

El parámetro de retraso t_d está relativamente claro, ya que se observa en las gráficas. La ganancia del filtro se escoge a 1, puesto que el manipulador intenta reproducir las señales de entrada en su misma magnitud. Falta por evaluar el amortiguamiento y la frecuencia natural, cuya estimación se realiza probando diversas combinaciones, en pasos discretos de 0.05, en rangos relativamente estrechos: de 1 a 3 Hz para la frecuencia natural (por observación de la Tabla 2), y de 1 a 1.6 para el amortiguamiento (puesto que se

aprecia que el sistema no es subamortiguado). Cada prueba se valida con cuatro señales de validación: una señal *chirp*, una onda cuadrada de amplitud exponencialmente decreciente, un seno de amplitud exponencialmente decreciente y una onda triangular. La Tabla 3 describe dichas señales, (A es la amplitud, f la frecuencia, r el factor de modificación de fase del *chirp*, y c la constante de atenuación de las dos ondas atenuadas).

Tabla 3: Señales de validación individual empleadas (para cada GdL).

Señal	Tipo	A (°) *pitch*/*roll*	f (Hz)	r	c
S1	*chirp*	30/25	0.01	1.1	n/a
S2	onda sinusoidal atenuada	30/25	1	n/a	0.06
S3	onda cuadrada atenuada	30/25	0.5	n/a	0.06
S4	onda triangular	20/20	0.125	n/a	n/a

Para comprobar la similitud entre la salida proporcionada por el simulador y la salida real del manipulador, se emplea la función *goodnessOfFit()* de MATLAB (Mathworks, 2017) con el indicador NRMSE (*Normalized Root Mean Square Error*), que proporciona una medida entre -∞ y 1, siendo este último el mejor valor posible.

Se escogen los valores de ω_n y ξ que proporcionan un indicador NRMSE medio (de las cuatro señales) máximo. Los valores finalmente escogidos para los parámetros del simulador se recogen en la Tabla 4.

Tabla 4: Valores escogidos para los parámetros de la simulación.

GdL	t_d (s)	G	ξ	ω_n (Hz)
pitch	0.08	1.0	1.5	2.8
roll	0.08	1.0	1.5	2.7

4.1. Pruebas de Validación

Dado que el resultado de la simulación no puede alcanzar la misma precisión que si se utilizara un sistema de simulación newtoniano que imite el movimiento de los motores y el comportamiento de las uniones entre los elementos móviles del manipulador, es necesario comprobar que el sistema simula las salidas de la plataforma con un alto grado de similitud.

En realidad, las pruebas realizadas para el ajuste de los valores de ω_n y ξ, son ya pruebas de validación, aunque para cada GdL individual por separado. Por ello, falta realizar la comprobación global del sistema. La Tabla 5 muestra los valores obtenidos para las diversas señales de validación empleadas. En la primera columna se valida el movimiento de *pitch* independientemente. En la segunda columna se valida el *roll* y en la tercera, el movimiento combinado. Las dos primeras columnas corresponden a los valores obtenidos en el apartado anterior (ajuste de cada GdL individual).

Las señales S5a, S6a, S7a y S8a se aplican al movimiento de *pitch*, mientras que las señales S5b, S6b, S7b y S8b corresponden al movimiento de *roll*. Todas ellas se listan en la Tabla 6, junto con los parámetros que las definen.

Se puede observar que los valores obtenidos para el indicador de similitud NRMSE son bastante elevados, con una media superior a 0.9. Esta precisión debería ser suficiente para ajustar algoritmos MCA mediante este simulador, aunque en cualquier caso, el nivel de precisión requerido debe ser decisión del diseñador de la aplicación de RV.

Tabla 5: Resultados de la validación de la simulación.

Señal	*Pitch*	*Roll*	*Pitch+Roll*
S1	0.9398	0.9402	-
S2	0.9216	0.9077	-
S3	0.9157	0.8814	-
S4	0.8674	0.8573	-
S5a+S5b	-	-	0.9336
S6a+S6b	-	-	0.9249
S7a+S7b	-	-	0.8579
S8a+S8b	-	-	0.9088

Tabla 6: Señales de validación conjunta empleadas.

Señal	Tipo	A (°)	f (Hz)	r	c
S5a	onda sinusoidal	30	0.05	n/a	n/a
S5b	onda sinusoidal	30	$0.0\hat{3}$	n/a	n/a
S6a	onda sinusoidal	30	0.05	n/a	n/a
S6b	onda cuadrada amortiguada	30	$0.0\hat{3}$	n/a	0.04
S7a	*chirp*	30	0.01	1.08	n/a
S7b	onda sinusoidal amortiguada	30	0.1	n/a	0.05
S8a	onda sinusoidal amortiguada	30	0.1	n/a	0.05
S8b	onda en diente de sierra	30	$0.0\hat{3}$	n/a	n/a

Para demostrar visualmente el resultado, las Figuras 13, 14 y 15 muestran algunos resultados de la simulación comparando la salida del simulador con la ofrecida por la plataforma real. Se puede observar que el grado de similitud es bastante elevado, aunque existen algunos momentos en los que la simulación diverge ligeramente del comportamiento real, debido a que se están ignorando las influencias dinámicas de unos GdL con respecto a otros. De algún modo, el resultado simulado es una simplificación del real. Sin embargo, las diferencias son suficientemente pequeñas como para que el simulador pueda emplearse para la validación estrategias de *washout* en simuladores de vehículos.

4.2. Pruebas de Rendimiento

Por último, es muy importante comprobar lo eficiente que es el simulador a nivel de coste temporal, puesto que uno de los principales requisitos de diseño de este simulador es sacrificar algo de precisión para ganar en eficiencia.

Para ello, analizamos el tiempo necesario para simular el comportamiento de la plataforma de movimiento durante una prueba de 100 segundos de duración empleando (e incluyendo en la medida de tiempo) un algoritmo de *washout* real, el algoritmo clásico de Reid y Nahon (Nahon & Reid, 1990). Esto valida, además, el uso "*in-the-loop*" del simulador y la utilidad del simulador como sustituto de la plataforma real.

Los resultados se pueden observar en la Tabla 7. En ella se compara el simulador propuesto en este trabajo con el simulador newtoniano-lagrangiano descrito en (Casas, et al., 2014) configurado para esta plataforma de dos GdL. Las pruebas se

realizan con un PC *Intel Pentium G840* @ 2.60 GHz, con 8 Gb RAM y sistema operativo *Windows 7* 64 bits.

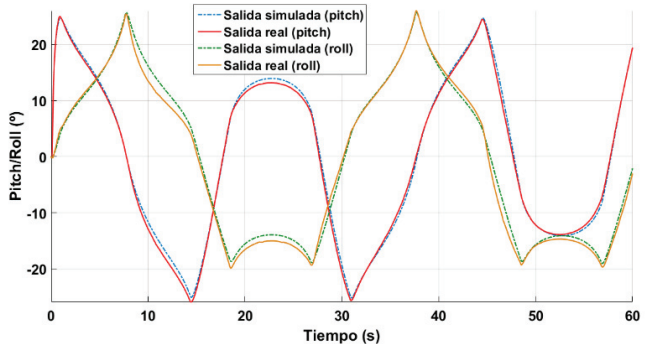

Figura 13: Validación del sistema (señales S5a+S5b).

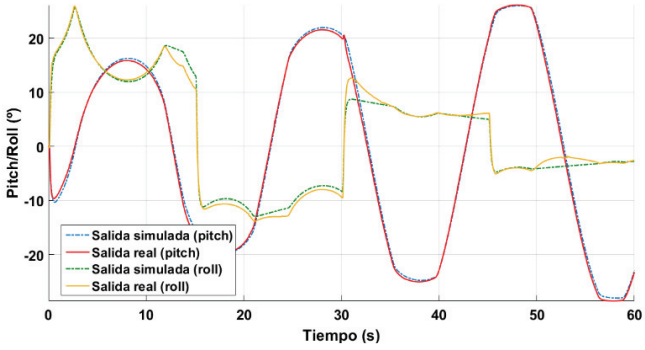

Figura 14: Validación del sistema (señales S6a+S6b).

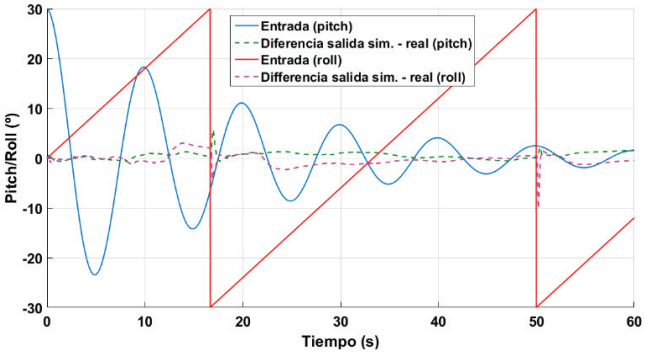

Figura 15: Validación del sistema (señales S8a+S8b).

La diferencia de velocidad es realmente muy grande ya que el simulador con filtros digitales permite realizar pruebas simuladas más de 200 veces más rápido que las pruebas reales, por un factor de aceleración de en torno a 30 del simulador newtoniano. Teniendo en cuenta que el *hardware* de prueba es bastante modesto, el simulador podría funcionar mucho más rápidamente aún. En cualquier caso, dado que la medida de tiempo incluye el cálculo del *washout*, éste es afortunadamente el factor limitante cuando se usa el simulador propuesto, cosa que no sucede en el que caso del otro simulador más lento, que sí que es el limitante.

Hay que tener en cuenta también que el simulador newtoniano-lagrangiano normalmente debe ser reinicializado cada vez que se realiza una nueva prueba, llevando los motores a la posición de reposo, y este tiempo no se incluyó en la medición. Esta operación es a veces costosa si se requiere precisión en la maniobra. Sin embargo, el simulador propuesto aquí realiza esta tarea de forma casi instantánea. Además, el simulador newtoniano-lagrangiano es más complejo de construir y ajustar que el aquí propuesto.

Tabla 7: Tiempo empleado para simular 100 segundos.

Simulador	Tiempo (s)	Factor de aceleración
filtros digitales	0.462	×216.45
newtoniano-lagrangiano	3.345	×29.89

5. Conclusiones y Trabajo Futuro

De los experimentos y pruebas realizadas hay varias conclusiones claras que pueden ser extraídas. La primera conclusión es que el objetivo planteado inicialmente se ha cumplido puesto que el método es efectivo, ya que es capaz de realizar la simulación de una plataforma de movimiento, sustituyendo a ésta en todas aquellas pruebas que no requieran que el usuario se sitúe sobre la plataforma. El grado de similitud respecto al comportamiento real es bastante elevado por lo que el simulador puede emplearse con cualquier algoritmo de *washout* ya que permite sustituir al *hardware* real en prácticamente las mismas condiciones que éste. La segunda conclusión es que el método obtenido es fácil de implementar y ejecutar, siendo la mayor dificultad la obtención de datos para la caracterización dinámica del dispositivo. Sin embargo, la captura de datos mediante métodos ópticos no es imprescindible y puede sustituirse por cálculos analíticos resolviendo la cinemática directa del manipulador, o incluso por otros sistemas de seguimiento más sencillos, como el *tracking* inercial mediante acelerómetros que es económico y muy sencillo de implementar, aunque algo menos preciso que el óptico cuando éste es de calidad y está bien calibrado. La tercera y última conclusión, aunque no por ello menos importante, es que el simulador es extraordinariamente rápido ya que se implementa en C mediante filtros digitales con el objetivo de ser ejecutado mucho más rápido que el *hardware* original. El factor de aceleración permite que ejecutemos pruebas equivalentes a una semana a tiempo completo (7 días, 24 horas por día) en sólo 47 minutos con un PC poco potente. Además, es importante recalcar también que el método propuesto no se limita a plataformas de este tipo, ni siquiera a plataformas de dos GdL, sino que podría funcionar con otro tipo de manipuladores paralelos de mayor complejidad, ya que en la obtención del método no se ha hecho ninguna suposición o derivación que se circunscriba a la plataforma analizada. La única salvedad es que el comportamiento de cada manipulador concreto puede ser diferente, por lo que habría que evaluar la idoneidad del método para cada caso concreto. La evaluación se realizaría de modo similar a como se ha propuesto en este trabajo, comparando las salidas del simulador con las salidas del mecanismo real, ante entradas que se consideren representativas de la aplicación que se vaya a hacer del dispositivo. Es importante recalcar que el grado de precisión necesario para el simulador dependerá de la aplicación para la que se use, por lo que no se puede establecer a priori un umbral de precisión mínima para la validación. De cualquier modo, no es esperable que otras plataformas de movimiento con otros diseños, controladores o actuadores se comporten de forma radicalmente diferente a esta, si el objetivo de las mismas es su uso en RV y simuladores de vehículos, ya que en este tipo de aplicaciones se requieren manipuladores robóticos con un comportamiento dinámico similar al aquí analizado. Para otras

aplicaciones que requieran de otro tipo de dispositivos y/o de mayor precisión, la situación puede ser distinta.

Como trabajo futuro podríamos incluir varias líneas de investigación. Dado que el simulador no incluye módulo de dibujado podría plantearse añadir uno, aunque ello, lógicamente, redundaría en una reducción de la velocidad de simulación, por lo que el módulo debería ser opcional. En cuanto al núcleo de simulación, se podría plantear implementarlo mediante filtros de frecuencia natural variable (o incluso de orden distinto de 2) y también es posible analizar otras opciones de identificación de sistemas, incluso no lineales, para representar el mecanismo. El abanico de posibilidades aquí es muy grande, aunque no se debería perder de vista que el objetivo buscado es obtener un simulador rápido y eficiente, aunque no obtenga la mayor precisión posible. En caso de buscar más precisión a costa de eficiencia y velocidad, se deberían valorar otras alternativas, complementarias a la aquí presentada. Por último, aunque debería ser sencillo trasladar el método propuesto, sería deseable comprobar el funcionamiento del mismo en otro tipo de dispositivos generadores de movimiento (incluso de tipo serie, cuya simulación no debiera ser muy diferente), cosa que no se realiza en este trabajo por cuestiones económicas y de tiempo.

English Summary

On the Simulation of Robotic Motion Platforms with Digital Filters for Virtual Reality Applications.

Abstract

Robotic motion platforms are used in many vehicle simulators and Virtual Reality applications. However, the set-up of the so-called *washout* algorithms that control the generation of self-motion is a hard process, since a great deal of tests need to be performed before reaching a proper motion fidelity. The availability of simulation tools eases this tuning task. Therefore, a motion platform characterization and simulation method is proposed in this paper. The method relies on second order digital filters and provides a reliable, yet very fast simulation system, which is assessed by means of a two degree-of-freedom motion platform, although the method might be applied to simulate other motion mechanisms.

Keywords:

Motion platforms, simulation, digital filters, Virtual Reality, robotics, real time.

Referencias

Abed-Meraim, K., Qiu, W., & Hua, Y. (1997). Blind system identification. *Proceedings of the IEEE, 85*(8), 1310-1322.

Cao, Y., Gosselin, C., Zhou, H., Ren, P., & Ji, W. (2013). Orientation-singularity analysis and orientationability evaluation of a special class of the Stewart–Gough parallel manipulators. *Robotica, 31*(08), 1361-1372.

Casas, S., Alcaraz, J. M., Olanda, R., Coma, I., & Fernández, M. (2014). Towards an extensible simulator of real motion platforms. *Simulation Modelling Practice and Theory, 45*(0), 50-61.

Casas, S., Coma, I., Portalés, C., & Fernández, M. (2016). Towards a simulation-based tuning of motion cueing algorithms. *Simulation Modelling Practice and Theory, 67*, 137–154.

Casas, S., Coma, I., Riera, J. V., & Fernández, M. (2015). Motion-Cuing Algorithms: Characterization of Users' Perception. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 57*(1), 144-162.

Casas, S., Olanda, R., & Dey, N. (2017). Motion Cueing Algorithms: A Review - Algorithms, Evaluation and Tuning. *International Journal of Virtual and Augmented Reality, 1*(1), 90-106.

Cleary, K. (2016). *Medical robotics for pediatric applications shoulder arthrography, ankle rehabilitation, and temporal bone surgery.* Paper presented at the World Automation Congress (WAC), 2016.

Dagdelen, M., Reymond, G., Kemeny, A., Bordier, M., & Maizi, N. (2009). Model-based Predictive Motion Cueing Strategy for Vehicle Driving Simulators. *Control Engineering Practice, 17*(19), 995-1003.

Fu, L., & Li, P. (2013). *The research survey of system identification method.* Paper presented at the Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2013 5th International Conference on.

Fung, J., Malouin, F., McFadyen, B., Comeau, F., Lamontagne, A., Chapdelaine, S., et al. (2004). *Locomotor rehabilitation in a complex virtual environment.* Paper presented at the Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE.

Gotmare, A., Bhattacharjee, S. S., Patidar, R., & George, N. V. (2017). Swarm and evolutionary computing algorithms for system identification and filter design: A comprehensive review. *Swarm and Evolutionary Computation, 32*, 68-84.

Grant, P. R., & Reid, L. D. (1997). Motion Washout Filter Tuning: Rules and Requirements. *Journal of Aircraft, 34*(2), 145-151.

Groen, E. L., & Bles, W. (2004). How to use body tilt for the simulation of linear self motion. *Journal of Vestibular Research, 14*(5), 375-385.

Hajimirzaalian, H., Moosavi, H., & Massah, M. (2010). *Dynamics analysis and simulation of parallel robot Stewart platform.* Paper presented at the Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on.

Hodge, S. J., Perfect, P., Padfield, G. D., & White, M. D. (2015). Optimising the Yaw Motion Cues Available From a Short Stroke Hexapod Motion Platform. *The Aeronautical Journal, 119*(1121), 1-21.

Hulme, K. F., & Pancotti, A. (2004). *Development of a virtual 6 DOF motion platform for simulation and rapid synthesis.* Paper presented at the 45 th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference.

Küçük, S. (2012). *Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization*: InTech.

Kurosaki, M. (1978, June 1978). *Optimal washout for control of a moving base simulator.* Paper presented at the Proceedings of the Seventh Triennial World Congress of IFAC (International Federation of Automatic Control), Helsinki, Finland.

Lau, H., Chan, L., & Wong, R. (2007). A virtual container terminal simulator for the design of terminal operation. *International Journal on Interactive Design and Manufacturing (IJIDeM), 1*(2), 107-113.

Li, S. J., & Gosselin, C. M. (2012). *Determination of singularity-free zones in the workspace of planar parallel mechanisms with revolute actuators.* Paper presented at the Applied Mechanics and Materials.

Li, Y.-W., Wang, J.-S., Wang, L.-P., & Liu, X.-J. (2003). *Inverse dynamics and simulation of a 3-DOF spatial parallel manipulator.* Paper presented at the Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on.

Ljung, L. (1999). *System identification*: Wiley Online Library.

Ljung, L. (1999). *System Identification: Theory for the User* (2nd Edition ed.): Prentice Hall.

Lozoya-Santos, J. d. J., Tudon-Martinez, J. C., & Salinas, J. (2017). *Control Design for a Motion Cueing on Driving Simulator.* Paper presented at the Journal of Physics: Conference Series.

Mathworks. (2017). https://www.mathworks.com/products/matlab.html. *MATLAB, The Language of Technical Computing* Retrieved 02/01/2017, 2017

Mauro, S., Gastaldi, L., Pastorelli, S., & Sorli, M. (2016). Dynamic flight simulation with a 3 dof parallel platform. *International Journal of Applied Engineering Research, 11*(18), 9436-9442.

MSC. (2017). http://www.mscsoftware.com/product/adams. *Adams, The Multibody Dynamics Simulation Solution* Retrieved 02/01/2017, 2017

Nahon, M. A., & Reid, L. D. (1990). Simulator motion-drive algorithms - A designer's perspective. *Journal of Guidance, Control, and Dynamics, 13*(2), 356-362.

Optitrack. (2017). http://optitrack.com/. *Motion Capture Systems - Optitrack* Retrieved 02/01/2017, 2017, from http://www.naturalpoint.com/optitrack/

Ortega, J. J., & Sigut, M. (2016). Prototipo de una plataforma móvil de bajo coste para simulación de vuelo de alto realismo. *Revista Iberoamericana de Automática e Informática Industrial RIAI, 13*(3), 293-303.
Paarmann, L. D. (2001). *Design and analysis of analog filters: a signal processing perspective* (Vol. 617): Springer Science & Business Media.
Page, L. R. (2000). *Brief History of Flight Simulation.* Paper presented at the SimTecT 2000 Proceedings, Sydney, NSW, Australia.
Parrish, R. V., Dieudonne, J. E., & Martin Jr, D. J. (1975). Coordinated Adaptive Washout for Motion Simulators. *Journal of Aircraft, 12*(1), 44-50.
Reid, L. D., & Nahon, M. A. (1985). *Flight Simulation Motion-Base Drive Algorithms: Part 1 - Developing and Testing the Equations*. University of Toronto: UTIAS.
Reid, L. D., & Nahon, M. A. (1988). Response of airline pilots to variations in flight simulator motion algorithms. *Journal of Aircraft, 25*(7), 639-646.
Reymond, G., & Kemeny, A. (2000). Motion Cueing in the Renault Driving Simulator. *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility, 34*, 249-259.
Rorabaugh, C. B. (1993). *Digital Filter Designer's Handbook - Featuring C Routines*. Blue Ridge(PA): TAB Books - McGraw Hill.
Royal-Aeronautical-Society. (1979). *50 Years of Flight Simulation, Conference Proceedings*. London, UK.
Schmidt, S. F., & Conrad, B. (1969). *The Calculation of Motion Drive Signals for Piloted Flight Simulators*. Palo Alto, CA, USA: NASA.
Selvakumar, A. A., Pandian, R. S., Sivaramakrishnan, R., & Kalaichelvan, K. (2010). *Simulation and performance study of 3—DOF parallel manipulator units.* Paper presented at the Emerging Trends in Robotics and Communication Technologies (INTERACT), 2010 International Conference on.
Sinacori, J. B. (1977). *The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility*. CA, USA: Moffet Field.
Sivan, R., Ish-Shalom, J., & Huang, J. K. (1982). An Optimal Control Approach to the Design of Moving Flight Simulators. *IEEE Transactions on System, Man & Cybernetics, 12*(6), 818-827.
Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P. Y., et al. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica, 31*(12), 1691-1724.
Slob, J. J. (2008). *State-of-the-Art Driving Simulators, a Literature Survey*. Eindhoven, The Netherlands: Eindhoven University of Technology.
Stewart, D. (1965). *A Platform with six degrees of freedom*.
Vogel, C., Fritzsche, M., & Elkmann, N. (2016). *Safe Human-Robot Cooperation with High-Payload Robots in Industrial Applications*. Paper presented at the The Eleventh ACM/IEEE International Conference on Human Robot Interaction.
Winder, S. (2002). *Analog and digital filter design*: Newnes.
Zhang, C., & Zhang, L. (2013). Kinematics analysis and workspace investigation of a novel 2-DOF parallel manipulator applied in vehicle driving simulator. *Robotics and Computer-Integrated Manufacturing, 29*(4), 113-120.

Apéndice A – Cinemática Inversa del Dispositivo 2-RSSU

El origen del sistema de coordenadas está situado en la junta universal. La posición de reposo se toma como aquella en la que los motores están en posición horizontal. Los *parámetros* de la cinemática inversa son:

· r = longitud de biela (brazo corto unido al motor).
· p = longitud de pistón (brazo largo unido a la biela y a la base móvil).
· $\overrightarrow{L_k}$ = posiciones de los *puntos inferiores* de unión (el que forman los ejes de los motores al unirse a las bielas). k = 1, 2.
· $\overrightarrow{U_{k0}}$ = posiciones en reposo de los *puntos superiores* de unión (los puntos en los que se unen los pistones a la base móvil). k = 1, 2.

Las *entradas* suministradas a la cinemática inversa son:
· $pitch$ = ángulo de inclinación frontal deseado para la plataforma.
· $roll$ = ángulo de inclinación lateral deseado para la plataforma.

Las *salidas* proporcionadas por la cinemática inversa son:
· α_k= ángulos de los motores con el plano horizontal. k = 1, 2.
· válido = sí/no. Para determinar si la petición es alcanzable.

El *procedimiento* de cálculo es el siguiente:

$$\overrightarrow{U_k} = R_x R_y \overrightarrow{U_{k0}}$$

donde R_x y R_y son matrices de rotación:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cp & sp \\ 0 & -sp & cp \end{bmatrix}, \quad cp = \cos(pitch), \quad sp = \sin(pitch)$$

$$R_y = \begin{bmatrix} cr & 0 & -sr \\ 0 & 1 & 0 \\ sr & 0 & cr \end{bmatrix}, \quad cr = \cos(roll), \quad sr = \sin(roll)$$

La posición de los vértices inferiores ($\overrightarrow{L_k}$), calculada con respecto a los vértices superiores ($\overrightarrow{U_k}$), es:

$$\overrightarrow{L_k} = \overrightarrow{U_{k0}} - \overrightarrow{U_k}$$

Para buscar los puntos de encuentro entre la biela y el pistón ($\overrightarrow{X_k}$), calculamos la intersección entre una esfera de radio p (centrada en el origen porque estamos haciendo estos cálculos con respecto a $\overrightarrow{U_k}$) y un círculo de radio r con centro en $\overrightarrow{L_k}$:

$$(\overrightarrow{X_{k1}},\ \overrightarrow{X_{k2}}) = \text{InterseccionEsferaCirculo}\left(p, r, \overrightarrow{L_k}\right)$$

La solución es dual, y se calcula de la siguiente forma:

$$\overrightarrow{(x,y,z)} = \text{InterseccionEsferaCirculo}(p, r, \vec{L})$$

Se plantea este sistema de ecuaciones:

$$\begin{cases} x^2 + y^2 + z^2 = p^2 \\ (x - L_x)^2 + (z - L_z)^2 = r^2 \\ y = L_y \end{cases}$$

cuyas soluciones son:

$$\begin{cases} y = L_y \\ x = \dfrac{-r^2 {L_x}^2 + {L_x}^2\left({L_x}^2 + {L_z}^2 + p^2 - y^2\right) \pm L_z a}{2L_x({L_x}^2 + {L_z}^2)} \\ z = \dfrac{-r^2 L_z + L_z\left({L_x}^2 + {L_z}^2 + p^2 - y^2\right) \pm a}{2({L_x}^2 + {L_z}^2)} \end{cases}$$

donde

$$a = \sqrt{-{L_x}^2\left(r^4 - 2r^2\left({L_x}^2 + {L_z}^2 + p^2 - y^2\right) + \left({L_x}^2 + {L_z}^2 - p^2 + y^2\right)^2\right)}$$

Si a es un número complejo, no hay solución física válida. Si no, se soluciona la intersección esfera-círculo proporcionando $\overrightarrow{X_{kJ}}$. Como $\overrightarrow{X_{kJ}}$ y $\overrightarrow{L_k}$ están ambos referidos a $\overrightarrow{U_k}$, su substracción da lugar al vector de orientación de la biela ($\overrightarrow{B_{k1}}$ and $\overrightarrow{B_{k2}}$):

$$\begin{cases} \overrightarrow{B_{k1}} = \overrightarrow{X_{k1}} - \overrightarrow{L_k} \\ \overrightarrow{B_{k2}} = \overrightarrow{X_{k2}} - \overrightarrow{L_k} \end{cases}$$

Y los dos posibles ángulos para la solución son:

$$\begin{cases} \alpha_{k1} = tan^{-1}(B_{k1z}/B_{k1x}) \\ \alpha_{k2} = tan^{-1}(B_{k2z}/B_{k2x}) \end{cases}$$