Escola Tècnica Superior d'Enginyeria

Departament d'Informàtica

VNIVERSITAT DE VALÈNCIA

TESIS DOCTORAL

MEJORAS EN LA GENERACIÓN DE CLAVES GRAVITO-INERCIALES EN SIMULADORES DE VEHÍCULOS NO AÉREOS

Sergio Casas Yrurzum

Dirigida por:

Dr. D. Marcos Fernández Marín

Valencia 2014

El **Dr. D. Marcos Fernández Marín**, profesor titular de Ciencias de la Computación e Inteligencia Artificial del Departamento de Informática de la Universitat de València, CERTIFICA que la presente memoria

"Mejoras en la Generación de Claves Gravito-inerciales en Simuladores de Vehículos No Aéreos"

ha sido realizada bajo su dirección, en el Departamento de Informática y en el Instituto de Robótica y Tecnologías de la Información y de la Comunicación (IRTIC) de la Universitat de València, por D. Sergio Casas Yrurzum, y constituye su tesis para optar al grado de **Doctor en Ingeniería Informática**, correspondiente al programa de doctorado **"Informática y Matemática Computacional"**.

Y para que conste, en cumplimiento de la legislación vigente, la presentamos ante la Escuela Técnica Superior de Ingeniería (ETSE) de la Universitat de València, a 3 de Julio de 2014.

EL DIRECTOR:

Dr. D. Marcos Fernández Marín

Agradecimientos

A los doctores Peter R. Grant y Lloyd D. Reid por proporcionarme toda la información necesaria sobre el fantástico trabajo realizado por el UTIAS sobre generación de movimiento.

A la brillante, y no siempre bien comprendida, mente de Ausiàs Llorenç, porque muchas de sus descabelladas ideas fueron la necesaria fuente de inspiración de algunas de las soluciones aplicadas en este trabajo.

A todos mis compañeros del IRTIC que me han ayudado, en mayor o menor grado, a realizar este trabajo: Inma, Silvia, Ricardo, Bibi, Jesús, Manolo, Pedro, Julio, etc.

A José Vicente Riera, por los cientos de horas dedicadas a construir todas esas plataformas de movimiento, sin las cuales no hubiera podido realizarse este trabajo. Ahora te toca a ti...

A mi director, el doctor Marcos Fernández por brindarme la oportunidad de realizar este trabajo. Sin su apuesta por el campo de la simulación y su paciencia, esto nunca hubiera sido posible.

A mi mujer, Inés, por todas esas horas robadas de estar con ella.

A mis padres y abuelos por darme siempre la mejor educación que han podido. Simplemente, muchas gracias.

"The greatest enemy of knowledge is not ignorance, it is the illusion of knowledge."

Stephen Hakwing

Resumen

El objetivo fundamental de la realidad virtual (RV) es proporcionar al usuario una sensación completa de pertenencia a un entorno virtual alternativo. Para ello, cualquier aplicación de RV debe estimular, del modo más eficaz posible, todas (o el mayor número de) las claves sensoriales que hacen posible que el usuario sienta como cierta esa pertenencia al mundo virtual alternativo.

Aunque existen muchas claves sensoriales y la aportación concreta de cada una de ellas a la percepción global de inmersión y presencia virtual es todavía parcialmente desconocida, la mayoría de aplicaciones de RV suelen centrarse en las claves visuales y sonoras, olvidando, hasta cierto punto, el resto. Las razones de este hecho no son casuales, ya que la generación de claves extra-audiovisuales es técnicamente compleja y en numerosas ocasiones, costosa en términos económicos.

Entre las claves extra-audiovisuales más importantes se encuentran las claves gravito-inerciales, relacionadas con la percepción del movimiento y la orientación del cuerpo humano. Este tipo de claves se estimulan habitualmente mediante la construcción de plataformas de movimiento, sobre las que se suele situar al usuario de la simulación. Estas plataformas están dotadas de actuadores que permiten desplazarlas y orientarlas dentro de unos límites. Para el control de los movimientos de la plataforma se diseñan unos algoritmos específicos conocidos habitualmente como algoritmos de generación de claves gravito-inerciales (en inglés *Motion Cueing Algorithms* - MCA).

Aunque este tipo de claves ha sido estudiado e incluido en simuladores desde hace más de 50 años, se ha avanzado menos en este campo que en otros aspectos de la simulación, como la generación de claves audiovisuales. De hecho, cómo simular óptimamente un movimiento ilimitado con un generador de movimiento que debe ceñirse a unos límites físicos, es todavía un problema sin resolver. Hay 3 razones fundamentales para ello. La primera es la naturaleza del problema. Es un problema complejo de optimización con restricciones, cuya solución depende de múltiples factores, entre ellos factores humanos difíciles de medir (y todavía parcialmente desconocidos) relacionados con la percepción del movimiento. La segunda es la falta de un criterio para poder comparar diferentes soluciones. Y la tercera, la multitud de parámetros que se deben ajustar en los distintos algoritmos para poder hacer análisis comparativos relevantes entre ellos. Si uno compara dos de estos algoritmos, y el esfuerzo dedicado a ajustar los parámetros de uno es mucho mayor que el empleado con el otro, la comparación no será relevante.

Por otro lado, dado que las primeras aplicaciones comerciales de realidad virtual fueron los simuladores de vuelo, la mayoría de los algoritmos MCA han sido desarrollados y ajustados para vehículos aéreos. Por ello, aunque, estos algoritmos son aplicables a otro tipo de vehículos, no fueron diseñados con ese propósito, por lo que resulta interesante estudiar la aplicación de este tipo de algoritmos a simuladores de vehículos no aéreos, ya que es un campo mucho menos estudiado.

El objetivo de esta tesis es mejorar la generación de claves gravito-inerciales en simuladores de vehículos no aéreos.

Para conseguirlo, en lugar de proponer nuevos algoritmos de generación de claves gravito-inerciales, se estudiarán nuevas formas de evaluar, de manera objetiva, los algoritmos de generación de claves gravito-inerciales, de forma que se pueda conocer si uno es más apropiado que otro según un determinado criterio. Este criterio objetivo de evaluación se basará en una caracterización previa de la percepción subjetiva de usuarios humanos a la generación de claves gravito-inerciales mediante algoritmos de tipo MCA, de manera que exista una correlación entre el criterio de evaluación y la presencia inducida sobre el usuario por el generador de movimiento.

Dado que la evaluación de este tipo de algoritmos puede ser costosa tanto en términos temporales, como económicos, e incluso humanos, además de los citados criterios, se desarrollará un procedimiento de evaluación basado en un simulador de plataforma de movimiento como método para acelerar y simplificar la evaluación y prueba de algoritmos de generación de claves gravito-inerciales. El simulador será una aplicación gráfica en tiempo real, que será validada con dos ejemplos reales de plataforma de 3 y 6 grados de libertad.

Finalmente, el criterio de evaluación se empleará para ajustar los parámetros de los algoritmos de generación de claves gravito-inerciales. Al disponer de un criterio objetivo, será más sencillo automatizar el proceso de ajuste de parámetros. Sin embargo, dado que el espacio de parámetros de un algoritmo de este tipo puede llegar a ser muy grande, el problema se convierte en un problema de optimización inabordable desde el punto de vista

computacional. Es por ello que desarrollaremos y estudiaremos métodos de búsqueda heurística para la solución del problema.

Además, antes de estudiar la evaluación y la asignación de parámetros en algoritmos MCA, se estudiará cómo analizar las necesidades de generación de claves gravito-inerciales en función del tipo de simulador que se desee construir, y cómo aprovechar del mejor modo posible el tipo de plataforma del que se disponga. Ejemplificaremos este análisis mediante el estudio de las necesidades gravito-inerciales de un simulador de bote de rescate y el estudio de una plataforma de 3 grados de libertad.

Índice de Capítulos

AGRADECIMIENTOS ..... **I**
RESUMEN ..... **III**
ÍNDICE DE CAPÍTULOS ..... **VII**
ÍNDICE DE FIGURAS ..... **XIII**
ÍNDICE DE TABLAS ..... **XVII**
ÍNDICE DE ALGORITMOS ..... **XXI**
MOTIVACIÓN ..... **XXIII**
OBJETIVOS ..... **XXV**
ESTRUCTURA DEL DOCUMENTO ..... **XXVI**

1- INTRODUCCIÓN **1**

1.1 – SIMULACIÓN Y REALIDAD VIRTUAL 1
1.2 – SIMULACIÓN DE VEHÍCULOS 3
1.2.1 – Historia *3*
1.2.2 – Ventajas de los Simuladores de Vehículos *9*
1.3 – COMPONENTES BÁSICOS DE UN SIMULADOR DE VEHÍCULOS 11
1.4 – EL SISTEMA INERCIAL 15
1.5 – PROBLEMÁTICA ASOCIADA A LAS PLATAFORMAS DE MOVIMIENTO 19

2- ESTADO DEL ARTE **23**

2.1 – CINESTESIA Y CLAVES PERCEPTIVAS 23
2.1.1 – Claves Visuales *25*
2.1.2 – Claves Sonoras *28*
2.1.3 – Claves Predictivas *29*
2.1.4 – Claves Gravito-inerciales *30*
2.1.5 – Claves Propioceptivas *32*
2.1.6 – Otras Claves *32*
2.2 – CLAVES GRAVITO-INERCIALES 33

2.2.1 – Mecanismo de Percepción Gravito-inercial Humano *33*
2.2.2 – Canales Semicirculares *34*
2.2.3 – Otolitos *35*
2.2.4 – Otros Mecanismos *40*
2.2.5 – Modelos de Percepción *41*
2.3 – ALGORITMOS MCA 45
2.3.1 – Concepción General *45*
2.3.2 – Algoritmo Clásico *58*
2.3.3 – Algoritmos Adaptativos *66*
2.3.4 – Algoritmos Óptimos *68*
2.3.5 – Otros Algoritmos *70*
2.3.6 –Sistemas de Evaluación *70*
2.3.7 –Ajuste de Parámetros *77*
2.4 – SITUACIÓN ACTUAL 81

3- CARACTERIZACIÓN GRAVITO-INERCIAL DE UN SIMULADOR 85

3.1 – CARACTERIZACIÓN DE LA DINÁMICA DE UN VEHÍCULO 86
3.1.1 – Sistema de Referencia *88*
3.1.2 – Sensorización *92*
3.1.3 – Análisis del Movimiento *99*
3.1.4 – Conclusiones *128*
3.2 – SIMULACIÓN EN TIEMPO REAL DE LA DINÁMICA DEL VEHÍCULO 131
3.2.1 – Modelo Físico *132*
3.3 – CONCLUSIONES 146

4- ANÁLISIS DEL DISPOSITIVO DE GENERACIÓN DE CLAVES GRAVITO-INERCIALES 149

4.1 – INTRODUCCIÓN 149
4.2 – DESCRIPCIÓN DEL MANIPULADOR 153
4.2.1 – Diseño del Manipulador *153*
4.2.2 – Cinemática del Manipulador *158*
4.2.3 – Validación Empírica de la Cinemática Inversa del Manipulador *166*
4.3 – CARACTERIZACIÓN CINEMÁTICA 169
4.3.1 – Análisis del Espacio de GdL *169*
4.3.2 – Mejoras sobre el Espacio de GdL *181*

4.3.3 – Análisis de Zonas Seguras ... *190*
4.4 – CARACTERIZACIÓN DINÁMICA ... 192
4.4.1 – Análisis en el Dominio del Tiempo ... *194*
4.4.2 – Análisis Frecuencial ... *198*
4.5 – CONCLUSIONES ... 205

5- MÉTRICAS DE EVALUACIÓN PARA MCA ... 209

5.1 – MÉTRICAS EXISTENTES ... 211
5.2 – HIPÓTESIS ... 213
5.3 – ELECCIÓN DE MÉTRICAS ... 216
5.3.1 – Error Cuadrático Medio ... *217*
5.3.2 – Error Cuadrático Medio Normalizado ... *220*
5.3.3 – Escalado Medio ... *221*
5.3.4 – Correlación ... *224*
5.3.5 – Retraso Estimado ... *225*
5.4 – COMBINACIÓN DE MÉTRICAS ... 228
5.4.1 – Combinación de Diferentes GdL ... *230*
5.5 – CONCLUSIONES ... 233

6- CARACTERIZACIÓN DE LA RESPUESTA HUMANA ANTE ALGORITMOS DE TIPO MCA ... 235

6.1 – PRUEBAS ... 236
6.1.1 – Descripción de los Manipuladores ... *238*
6.1.2 – Descripción de las Pruebas ... *241*
6.1.3 – Randomización ... *244*
6.2 – ANÁLISIS Y RESULTADOS ... 245
6.2.1 – Simulador de Conducción ... *246*
6.2.2 – Simulador de Bote de Rescate ... *249*
6.3 – DISCUSIÓN ... 249
6.4 – CONCLUSIONES ... 258

7- MEJORAS EN EL AJUSTE DE PARÁMETROS DE LOS MCA ... 261

7.1 – METODOLOGÍA ... 261
7.2 – SIMULADOR DE PLATAFORMA DE MOVIMIENTO ... 266
7.2.1 – Trabajos Relacionados ... *267*
7.2.2 – Descripción del Simulador ... *269*

7.3 – Esquemas Algorítmicos para el Ajuste de Parámetros 278
7.3.1 – Planteamiento del Problema 282
7.3.2 – Búsqueda Exhaustiva Discretizada 285
7.3.3 – Método de Monte-Carlo 289
7.3.4 – Heurísticas 291
7.3.5 – Recocido Simulado 292
7.3.6 – Algoritmo Genético o Darwiniano 296
7.3.7 – Algoritmo Coulómbico 305
7.4 – Evaluación de los Algoritmos 315
7.4.1 – Corrección de los Algoritmos 315
7.4.2 – Estudio de Rendimiento 318
7.4.3 – Estudio Comparativo 332
7.4.4 – Validación del Ajuste 336
7.5 – Conclusiones 338

8- CONCLUSIONES Y TRABAJO FUTURO 343

8.1 – Conclusiones 343
8.1.1 – Conclusiones Generales 343
8.1.2 – Conclusiones Particulares 345
8.2 – Aportaciones y Discusión 347
8.2.1 – Aportaciones Generales y Discusión 347
8.2.2 – Aportaciones Particulares 349
8.3 – Trabajo Futuro 350
8.4 – Publicaciones 352

APÉNDICE A- DESCRIPCIÓN DE SENSORES EN LA CARACTERIZACIÓN DEL BOTE DE RESCATE RÁPIDO 355
APÉNDICE B- AJUSTE Y VALIDACIÓN DEL SIMULADOR DE BOTE DE RESCATE RÁPIDO 357

B.1 – Ajuste del Número de Cubos 360
B.2 – Ajuste con Expertos 361
B.3 – Validación 363
B.3.1 – Validación Cuantitativa 364
B.3.2 – Validación Cualitativa 367

APÉNDICE C- AJUSTE Y VALIDACIÓN DEL SIMULADOR DE PLATAFORMA DE MOVIMIENTO 371

C.1 – AJUSTE DE PARÁMETROS 371
C.2 – VALIDACIÓN 374

BIBLIOGRAFÍA..... 385

"It ain't what you don't know that gets you into trouble.
It's what you know for sure that just ain't so."

Mark Twain

Índice de Figuras

Figura 1.1 - *Breese Penguin* **5**
Figura 1.2 - *Link Trainer* **6**
Figura 1.3 - *Esquema de un simulador de vehículos* **12**
Figura 1.4 - *Ejemplo de plataforma de movimiento* **16**
Figura 1.5 - *Los 6 grados de libertad del espacio cartesiano* **18**

Figura 2.1 - *El flujo óptico* **27**
Figura 2.2 - *El oído y sus partes* **34**
Figura 2.3 - *Detalle de un canal semicircular* **35**
Figura 2.4 - *Detalle de un otolito* **37**
Figura 2.5 - *Esquema de la generación de claves gravito-inerciales en simuladores* **49**
Figura 2.6 - *Entradas y salidas de un MCA* **52**
Figura 2.7 – *La ilusión somatográvica* **55**
Figura 2.8 – *Esquema de funcionamiento del algoritmo clásico* **60**

Figura 3.1 – *SdR del bote de rescate* **93**
Figura 3.2 – *SdR de un Wiimote* **96**
Figura 3.3 – *Localización de los sensores en el bote* **98**
Figura 3.4 – *El bote de rescate en el Puerto de Barcelona* **103**
Figura 3.5 – *Velocidad* ($|\overrightarrow{v_1}|$) *y pitch vs tiempo* **111**
Figura 3.6 – *Pitch, roll y yaw* **112**
Figura 3.7 – *Aceleración lineal* ($\overrightarrow{A_3}$) **114**
Figura 3.8 – *Velocidad angular* ($\overrightarrow{\omega_2}$) **115**
Figura 3.9 – *Aceleración angular* ($\overrightarrow{\alpha_2}$) **116**
Figura 3.10 – *Ángulo del timón* **118**

Figura 3.11 – *Viento aparente* ($|\overrightarrow{v_w}|$) **119**
Figura 3.12 – *PSD de la aceleración lineal* ($\overrightarrow{A_3}$) **120**
Figura 3.13a – *PSD de la aceleración angular* ($\overrightarrow{\alpha_2}$) **122**
Figura 3.13b – *PSD de la velocidad angular* ($\overrightarrow{\omega_2}$) **123**
Figura 3.13c – *PSD de pitch, roll y yaw* **124**
Figura 3.14 – *Desplazamiento lineal máximo en función de la frecuencia de corte* **125**
Figura 3.15 – *Desplazamiento angular máximo en función de la frecuencia de corte* **127**
Figura 3.16 – *Proceso de construcción del modelo físico de bote de rescate* **136**

Figura 4.1 – *El manipulador paralelo de tipo T1R2 analizado* **154**
Figura 4.2 – *Uniones y elementos móviles de la plataforma* **155**
Figura 4.3 – *Modelo geométrico de la plataforma* **156**
Figura 4.4 – *Sistema de coordenadas y posición de reposo* **161**
Figura 4.5 – *El MP T1R2 analizado, con tracking óptico* **168**
Figura 4.6 – *Validación de la cinemática inversa* **170**
Figura 4.7 – *Espacio de soluciones alcanzables* **180**
Figura 4.8 – *Rango del heave* **183**
Figura 4.9 – *Espacio de soluciones alcanzables sin offset de heave* **185**
Figura 4.10 – *Offsets de rotación* **187**
Figura 4.11 – *Espacio de soluciones alcanzables con el nuevo centro* **188**
Figura 4.12 – *Espacio de soluciones alcanzables con un pistón muy largo (10 metros)* **189**
Figura 4.13 – *Comportamiento ante diversas entradas para el movimiento de heave* **195**

Figura 4.14 – *Comportamiento ante diversas entradas para el movimiento de pitch* **198**
Figura 4.15 – *Comportamiento ante diversas entradas para el movimiento de roll* **199**
Figura 4.16 – *Comportamiento frecuencial para el movimiento de heave* **202**
Figura 4.17 – *Comportamiento frecuencial para el movimiento de pitch* **202**
Figura 4.18 – *Comportamiento frecuencial para el movimiento de roll* **203**
Figura 4.19 – *Diagrama de Bode (logarítmico)* **204**

Figura 6.1 – *Simulador de conducción* **237**
Figura 6.2 – *Simulador de bote de rescate* **238**
Figura 6.3 – *Modelo CAD de la plataforma de 6 GdL* **239**
Figura 6.4 – *Plataforma de 6 GdL con tracking óptico* **241**

Figura 7.1 – *Esquema de evaluación objetiva de algoritmos de generación de claves gravito-inerciales* **265**
Figura 7.2 – *Plataforma de 6 GdL real (arriba) vs plataforma virtual (abajo)* **274**
Figura 7.3– *Esquema reproductivo del algoritmo genético* **300**

Figura B.1– *Disposición hardware del simulador* **358**
Figura B.2– *Aceleración 0-25 nudos (izquierda), navegación de crucero a 20 nudos (centro) y deceleración brusca 25-0 nudos (derecha)* **365**
Figura B.3– *Giro de 360º a 15 nudos* **366**
Figura B.4– *Captura visual del simulador de bote de rescate* **368**

Figura C.1– *Respuesta comparada de la plataforma T1R2 para el movimiento de heave* **377**
Figura C.2– *Respuesta comparada de la plataforma T1R2 para los movimientos de pitch y roll* **378**
Figura C.3– *Respuesta comparada de la plataforma T3R3 para el movimiento de pitch* **379**
Figura C.4– *Detalle del comportamiento de la plataforma T1R2 para los movimientos de heave y roll* **380**
Figura C.5– *Detalle del comportamiento de la plataforma T3R3 para el movimiento de pitch* **381**

"True knowledge exists in knowing that you know nothing."

Socrates

Índice de Tablas

Tabla 2.1 - *Valores de los parámetros de los modelos de otolito y canal semicircular empleados por Reid y Nahon* **45**

Tabla 3.1 - *Descripción de maniobras del bote de rescate* **104**
Tabla 3.2 - *Rangos de las diversas magnitudes* **109**

Tabla 4.1 - *Hiper-rectángulos simétricos más largos: volúmenes y áreas* **190**
Tabla 4.2 - *Hiper-rectángulos asimétricos más largos: volúmenes y áreas* **191**

Tabla 6.1 - *Características de la plataforma de 3-GdL* **240**
Tabla 6.2 - *Características de la plataforma de 6-GdL* **240**
Tabla 6.3 - *Parámetros modificados del algoritmo clásico* **245**
Tabla 6.4 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo C-1* **247**
Tabla 6.5 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo C-2* **248**
Tabla 6.6 - *Correlación entre la CCPN y el indicador subjetivo para estrategias de combinación diferente, para el grupo C-2* **248**
Tabla 6.7 - *Percepción media subjetiva (pilotos humanos vs automáticos), para la configuración B y Grupo C-2* **249**
Tabla 6.8 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo B-1* **250**
Tabla 6.9 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo B-2* **251**

Tabla 6.10 - *Correlación entre la CCPN y el indicador subjetivo para estrategias de combinación diferente, para el grupo B-2* **251**
Tabla 6.11 - *Percepción media subjetiva (pilotos humanos vs automáticos), para la configuración B y Grupo B-2* **252**

Tabla 7.1 - *Validación de los algoritmos de optimización - 500 s* **317**
Tabla 7.2 - *Análisis del algoritmo de búsqueda exhaustiva* **320**
Tabla 7.3 - *Análisis del algoritmo de Monte-Carlo* **322**
Tabla 7.4 - *Análisis del algoritmo de Recocido Simulado* **323**
Tabla 7.5 - *Análisis de la probabilidad de mutación* **325**
Tabla 7.6 - *Análisis del tamaño de población* **326**
Tabla 7.7 - *Análisis del porcentaje de elitismo* **327**
Tabla 7.8 - *Análisis del porcentaje de selección natural* **328**
Tabla 7.9 - *Parámetros elegidos para el algoritmo genético* **329**
Tabla 7.10 - *Análisis del algoritmo genético* **329**
Tabla 7.11 - *Análisis del algoritmo Coulómbico* **331**
Tabla 7.12 - *Estudio comparativo exhaustivo para la plataforma T1R2* **333**
Tabla 7.13 - *Estudio comparativo exhaustivo para la plataforma T3R3* **333**
Tabla 7.14 - *Estudio comparativo exhaustivo para la plataforma T3R3 con modelo de percepción* **335**
Tabla 7.15 - *Estudio comparativo de los métodos de ajuste* **337**

Tabla B.1 - *Límites de excursión de la plataforma de movimiento del simulador* **359**
Tabla B.2 - *Parámetros de configuración del modelo físico* **362**
Tabla B.3 - *Preguntas y respuestas medias del cuestionario* **369**

Tabla C.1 - *Características de las dos plataformas de movimiento reales* **372**
Tabla C.2 - *Parámetros físicos de ambas plataformas virtuales de movimiento* **374**

"Real knowledge is to know the extent of one's ignorance."

Confucius

Índice de Algoritmos

Algoritmo **BusquedaGdL-HPR** **175**
Algoritmo **MonteCarlo-MaxRectSimetrico-HP** **177**
Algoritmo **CalcularRetraso** **227**
Algoritmo **OptimizacionMCA** **283**
Algoritmo **BusquedaExhaustivaMCA** **286**
Algoritmo **MonteCarloMCA** **290**
Algoritmo **RecocidoSimuladoMCA** **293**
Algoritmo **DarwinianoMCA** **301**
Algoritmo **CoulombicoMCA** **309**

"Reality is merely an illusion, albeit a very persistent one."

Albert Einstein

Motivación

La evolución sufrida en los últimos años por el mundo de los simuladores es muy significativa. Esta evolución se hace muy palpable en la calidad gráfica y sonora de los mismos, dado que las técnicas y el *hardware* audiovisual han experimentado un salto cualitativo y cuantitativo enorme. Esto, unido a un continuo y significativo descenso en el precio de los componentes *hardware*, hace que, hoy en día, los dispositivos de simulación puedan estar al alcance de casi cualquier persona. Además, el auge y generalización de Internet ha motivado la popularización de simuladores y juegos colaborativos o competitivos, conocidos como entornos virtuales distribuidos o DVE (*distributed virtual environments*).

Sin embargo, todos estos avances audiovisuales y telemáticos, han dejado en evidencia la falta de avances en otros aspectos, no menos necesarios, de algunos simuladores, como la recreación de claves gravito-inerciales en simuladores de vehículos. Aunque este tipo de claves ha sido estudiado e incluido en simuladores desde hace mucho tiempo, poco se ha avanzado en este campo en los últimos años, y una de las razones de este hecho es la complejidad del problema.

El problema fundamental estriba en la dificultad de simular un movimiento ilimitado mediante un dispositivo cuyo rango de movimiento es muy limitado, de manera que, al mismo tiempo, sea creíble para los individuos que lo experimenten.

Este problema de optimización, ya de por sí bastante complejo, se agrava por el hecho de que, aunque el mecanismo de percepción de estos movimientos ha sido ampliamente estudiado, persisten numerosas incógnitas psicofísicas que hacen difícil aseverar cuál es el mejor método para "engañar" al cuerpo humano y hacerle creer que está experimentando unas magnitudes físicas que en realidad no está experimentando.

Afortunadamente, en los últimos años se está produciendo un auge en la utilización y disponibilidad de este tipo de dispositivos, que hasta no hace muchos años estaban prácticamente restringidos a grandes corporaciones o instituciones (especialmente militares), debido principalmente a sus elevados costes. Este auge, además, se produce en paralelo a una reducción de costes, lo que ha provocado la aparición de dispositivos de simulación de movimiento de bajo coste, y un renacimiento sobre el problema de la generación de claves gravito-inerciales en simuladores.

A esta tendencia se ha unido el grupo ARTEC del Instituto de Robótica y de las Tecnologías de la Información y de la Comunicación (IRTIC) de la Universitat de València, en un esfuerzo por innovar y aportar soluciones de bajo coste para la generación de claves gravito-inerciales en simuladores. Este trabajo está motivado por los numerosos proyectos en este campo en los que el autor ha tenido el privilegio de participar, bajo la dirección del Dr. Marcos Fernández, dentro del citado instituto universitario, en los últimos años.

Objetivos

El objetivo principal de esta tesis es "mejorar las prestaciones de los sistemas de generación de claves gravito-inerciales en simuladores de vehículos no aéreos". Dado que la mayoría de estudios, sobre todo los más antiguos, versan sobre simuladores de vuelo, nosotros centraremos y ejemplificaremos nuestros análisis sobre simuladores de vehículos no aéreos, para explorar un campo mucho menos estudiado.

Para conseguirlo, en primer lugar se propone una manera de analizar cuáles son las necesidades (en cuanto a claves gravito-inerciales) que requiere un simulador, para poder elegir de manera adecuada el tipo de dispositivo más apropiado, para después analizarlo, y mejorar de esta forma, el uso que se le da al dispositivo de generación de movimiento.

En segundo lugar, se propone realizar una caracterización de la respuesta humana ante la generación de claves gravito-inerciales en simuladores. El objetivo de esta caracterización es entender cómo responden las personas ante este tipo de algoritmos, y poder así disponer de la información necesaria para desarrollar una métrica o sistema de evaluación no subjetivo. Esto ayudará a una mejor evaluación y comparación de los distintos algoritmos. No pretendemos sustituir completamente la evaluación subjetiva sino aportar algo de luz a un campo poco estudiado como es el de la evaluación objetiva de este tipo de algoritmos.

Disponiendo de esta métrica, desarrollaremos un procedimiento automatizado para el ajuste de parámetros en algoritmos de generación de claves gravito-inerciales. Este procedimiento se basará en el uso de un simulador de plataforma de movimiento, sobre el cuál se desarrollarán y probarán diversas técnicas para el ajuste de parámetros en algoritmos de este tipo. Dichas técnicas responderán a esquemas algorítmicos para la solución de problemas de optimización, dado que el problema del ajuste de parámetros en algoritmos de generación de claves gravito-inerciales es un problema de optimización con un espacio de soluciones muy grande.

Estructura del Documento

El presente documento se organiza de la siguiente forma:

En el capítulo 1 introduciremos el contexto y la problemática asociada a los algoritmos de generación de claves gravito-inerciales.

En el capítulo 2 revisaremos la historia de este tipo de algoritmos y el estado actual de la literatura de la materia.

En el capítulo 3 estableceremos un procedimiento para reconocer cuáles son las necesidades que un simulador de vehículos puede tener en cuanto a generación de movimiento virtual. Este procedimiento se ejemplificará sobre un caso real: un simulador de bote de rescate rápido.

En el capítulo 4 hablaremos sobre los dispositivos de generación de movimiento, y sugeriremos un procedimiento para adecuar su uso a la simulación de vehículos. Este análisis será ejemplificado sobre un caso real de plataforma de 3 grados de libertad.

En el capítulo 5 comenzaremos a establecer las bases para poder realizar una evaluación objetiva de los algoritmos de generación de claves gravito-inerciales. Para ello, propondremos, razonadamente, una serie de medidas objetivas que nos servirán para evaluar de una manera más justa la transmisión de percepción de movimiento que este tipo de algoritmos produce.

En el capítulo 6 comprobaremos estas medidas objetivas mediante una caracterización de la respuesta humana ante este tipo de algoritmos, de tal modo que podamos emplear esta información para seleccionar o combinar algunas de las medidas objetivas propuestas en el capítulo anterior.

En el capítulo 7 analizaremos diversas maneras de emplear las medidas previamente obtenidas para poder realizar el ajuste de parámetros de este tipo de algoritmos. Para este análisis utilizaremos un simulador de plataforma de movimiento cuyo diseño detallaremos.

Finalmente, en el capítulo 8 resumiremos las conclusiones y aportaciones del trabajo, para seguidamente establecer posibles líneas de investigación posteriores.

Capítulo 1

Introducción

En este capítulo trataremos de introducir los conceptos básicos y la problemática asociada a los algoritmos de generación de claves gravito-inerciales empleados en ciertas aplicaciones de realidad virtual, concretamente en simuladores de vehículos.

1.1 – Simulación y Realidad Virtual

Simulación y realidad virtual son conceptos relacionados, aunque distintos. La palabra simulación proviene del latín *simulare*, que significa imitar, engañar, fingir o representar. En términos modernos, entendemos por simulación la creación de un modelo que represente lo más fielmente posible algún proceso real o algún fenómeno físico o de la naturaleza, y la ejecución sobre él de experimentos o tareas con diversos objetivos como la evaluación, la comprensión del fenómeno, el desarrollo de estrategias, la predicción de estados futuros, etc. [1].

Por otro lado, la realidad virtual se podría definir como el proceso por el cual se consigue generar sobre uno o varios individuos una apariencia de realidad, que hace que éstos experimenten la sensación de pertenecer a una realidad que no es la que realmente están viviendo en ese momento. Esta apariencia de realidad debe ser lo suficientemente creíble como para crear entornos que se consideren sustitutos aceptables de los reales, no

siendo necesario que la realidad alternativa sea indistinguible de la real [2] (aunque ésta sería la situación ideal).

Simulación y RV son, por tanto, conceptos distintos aunque similares, y se podría decir que la RV es un caso particular de simulación en el que el objetivo fundamental es proporcionar al usuario una sensación completa de pertenencia a un entorno alternativo. Para ello, cualquier aplicación de RV debe estimular, del modo más eficaz posible, todas (o el mayor número de) las claves sensoriales que hacen posible que el usuario sienta como cierta esa pertenencia al mundo virtual alternativo.

Aunque existen muchas claves sensoriales y la aportación concreta de cada una de ellas a la percepción global de inmersión y presencia virtual no es todavía totalmente conocida, la mayoría de aplicaciones de RV suelen centrarse en las claves visuales y sonoras, olvidando, de alguna manera, el resto. Las razones de este hecho no son casuales, ya que la generación de claves extra-audiovisuales es técnicamente compleja y generalmente mucho más costosa en términos económicos, mientras que las claves visuales son más sencillas de generar, especialmente en los últimos años con el advenimiento de una ingente cantidad de dispositivos gráficos de alta calidad. Es por ello, que centraremos esta investigación en analizar otro tipo de claves, las claves gravito-inerciales, que explicaremos con posterioridad.

1.2 – Simulación de Vehículos

1.2.1 – Historia

Uno de los campos de aplicación más importantes de la RV, pero no el único, es el de la simulación de vehículos. Un simulador de vehículos es, básicamente, una recreación lo más fiel posible de la dinámica, el manejo, la sensación visual, sonora, de movimiento, etc. que se percibe cuando se conduce o pilota un determinado vehículo.

La simulación de vehículos es probablemente el campo de la RV con más historia, ya que existen simuladores de vehículos desde que existen sistemas informáticos e incluso antes. Por desgracia, la historia de los simuladores de vehículos está íntimamente ligada al desarrollo militar, y se puede decir que las mayores contribuciones a este campo las han realizado la marina de los EE.UU. (*US Navy*), la fuerza área (*US Air Force, USAF*) y la agencia aeroespacial estadounidense, (*National Aeronautics and Space Administrationan, NASA*). La literatura militar es abundante así como las colaboraciones entre universidades norteamericanas, grandes corporaciones y el ejército de los EE.UU.

Los primeros simuladores que se crearon fueron simuladores de vuelo, ya que la instrucción de pilotos fue desde el principio (y sigue siendo) costosa tanto en términos humanos, temporales y económicos. Es por ello que prácticamente todos los simuladores de la primera mitad de siglo fueron simuladores de vuelo.

El primer vuelo a motor controlado se les atribuye a los hermanos Orville y Wilbur Wright en 1903. En pocos años comenzó la aplicación militar de la aviación, por lo que los ingenieros se vieron obligados a construir simuladores para entrenar pilotos antes de ponerlos a pilotar aeronaves reales.

Los primeros intentos eran realmente sistemas de entrenamiento más que simuladores. Así, durante la Primera Guerra Mundial, la USAF desarrolló los aviones de entrenamiento *Breese Penguin*, llamados así porque fueron creados por *Breese Corporation*, y al igual que los pingüinos, no podían volar (ver Figura 1.1). Se trataba de aviones con alas muy cortas, que servían para enseñar a los aprendices de piloto a rodar por el suelo sin destruir el avión, cosa que, al parecer, ocurría bastante a menudo. Evidentemente, el *Breese Penguin* no enseñaba a volar ni era especialmente económico (llevaba alas, fuselaje, ruedas, motor, etc.), por lo que la consideración de simulador es discutible. En realidad, algunas estimaciones indican que en la Primera Guerra Mundial se perdieron más vidas de pilotos de avión durante los entrenamientos que en combate [3].

Un buen número de dispositivos electromecánicos fueron probados con posterioridad. En 1927, David Link, reconocido como el padre de la simulación moderna [4], construyó el primer simulador realmente mecánico, de nombre *Link Trainer* [5]. Se trataba de una estructura cerrada que se movía dependiendo de las acciones del piloto dentro de la misma. No disponía de electrónica ni de pantallas para representar el mundo exterior porque no existían en aquella época. El *Link Trainer*, considerado el primer simulador de la historia, servía para entrenar a los pilotos en el

llamado vuelo por instrumentos (vuelo sin referencias externas por falta de visibilidad). De esta forma, los pilotos podían practicar situaciones de vuelo potencialmente peligrosas y acostumbrarse a notar las reacciones del avión sin disponer de referencias visuales, guiándose únicamente por los instrumentos del aparato (ver Figura 1.2). La contribución de Link al entrenamiento de los pilotos de la fuerza área fue decisiva. Tanto es así que el *Link Trainer* se siguió usando hasta la Segunda Guerra Mundial [6] y por él pasaron unos 500.000 pilotos aliados [4].

Figura 1.1 - *Breese Penguin (© John Wickenden)*

No fue hasta los años 40 cuando las computadoras analógicas entraron en el mundo de la simulación. Recordemos que la primera máquina Turing-completa (la alemana *Z3* de

Konrad Zuse) data de 1941 [7]. Las primeras computadoras analógicas de los años 40 fueron empleadas para calcular ecuaciones de vuelo. Aunque las computadoras se empezaban a usar para simulación, la mayoría de simuladores de los años 40 y 50 se limitaban a recrear los puestos de mando para que los pilotos se acostumbraran a ellos, ya que no se disponía de tecnología suficiente para la recreación visual del mundo exterior. Entre éstos, encontramos el *Celestial Navigation Trainer* (usado para ayudar en la navegación guiada por estrellas [5]) y un poco más tarde, el simulador *Stratocruiser* de Pan American, el primer simulador de aviación comercial.

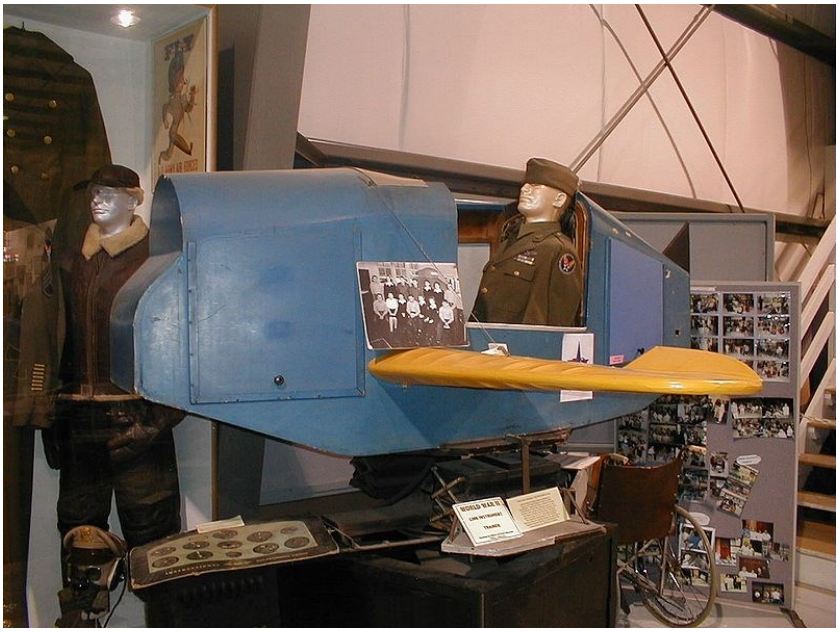

Figura 1.2 - *Link Trainer (© Tony Speer)*

Los primeros simuladores visuales llegaron en los años 50, aunque la parte visual normalmente se reproducía mediante cámaras de video y maquetas del terreno. El piloto, al desplazarse, accionaba un mecanismo que hacía mover la cámara por el espacio "virtual" maquetado, de forma coordinada [4]. Sin embargo, fue con la llegada de los computadores digitales, en los años 60 y 70 cuando el campo de la simulación se revolucionó. Aparecieron los simuladores *full motion* que simulaban movimientos e imágenes. Sin embargo, hubo que esperar hasta los años 80, con la llegada de la microelectrónica, para encontrar simuladores capaces de generar imágenes sintéticas relativamente realistas en tiempo real, y es, a partir de entonces, cuando los simuladores pueden empezar a considerarse verdaderamente aplicaciones de RV.

En cuanto al movimiento, aparte de los primeros sistemas de Link, los simuladores carecieron de generación de movimiento hasta la segunda mitad de los años 50. En 1958 se desarrolló el simulador *Comet IV* [5], [6] que incorporaba 1 grado de libertad rotacional (concepto que explicaremos con posterioridad). Posteriormente aparecieron muchos más simuladores con más grados de libertad.

Los dispositivos de movimiento se popularizaron en los años 60 y 70, siendo la NASA la que desarrolló los primeros algoritmos y sentó las bases de este campo. Fue en la década de los 60, con la *carrera espacial*, cuando se produjo el gran salto cualitativo en el campo de la simulación, especialmente aeroespacial. Recordemos que a finales de los años 60 el presupuesto de la NASA rondaba el 4% del presupuesto federal de los EE.UU, cifras que están muy lejos de las actuales (0.48% en 2012). Sin embargo,

aunque los dispositivos mecánicos de generación de movimiento de los años 70 ya eran similares, en cuanto a concepción, a los actuales, aún tuvimos que esperar hasta los años 90 para encontrar aplicaciones visuales realmente inmersivas. Por el contrario, desde los años 70, aunque tanto el *hardware* como el *software* visual han evolucionado enormemente, la generación de movimiento no lo ha hecho en tal medida.

En los años 60 aparecieron también simuladores de otro tipo de vehículos, especialmente de conducción [8], ya que los costes de fabricación de simuladores empezaron a ser menores, y la tecnología pudo ser aplicada a la conducción de forma rentable.

Actualmente existen simuladores de todo tipo de vehículos: coches [9], [10], [11], camiones [12], [13], motocicletas [14], [15], [16], barcos [17] [18], [19], [20], [21], [22], bicicletas [23], [24], e incluso triciclos [25], aunque los más comunes son los simuladores de vuelo [26] y, en menor medida, los de conducción. Este hecho no es casual. Los simuladores se construyen fundamentalmente por razones económicas (aunque también hay otras que en breve comentaremos). Cuanto mayor es el coste del simulador en relación al del vehículo real, menos interesante resulta. En el caso de la aeronáutica, el coste de los aparatos es tan elevado, que cualquier prueba que pueda hacerse sin emplear los aparatos reales es bienvenida.

Para una mayor descripción de la historia de los simuladores de vehículos el lector puede consultar, entre otros, [4], [5], [6] y [8].

1.2.2 – Ventajas de los Simuladores de Vehículos

Los simuladores se construyen porque ofrecen múltiples ventajas en determinadas situaciones. Es importante no perder de vista estos objetivos a la hora de construir un simulador. Entre dichas ventajas, podemos citar:

· **Seguridad y reducción de riesgos**. Al emplear simuladores, los accidentes simulados no tienen las consecuencias que tendría uno real. Con ello reducimos riesgos tanto humanos, como económicos.

· **Reducción de costes** respecto a pilotar un vehículo real. Esto depende del vehículo, por lo que, por ejemplo, hacer un simulador de bicicleta es, en principio, mucho menos rentable económicamente que un simulador de vuelo (aunque todo depende del objetivo que se busque con el simulador).

· **Mayor disponibilidad de las pruebas** respecto a las reales. Muchas veces, los vehículos reales no están disponibles porque hay pocos, porque el combustible es caro, porque su uso es potencialmente peligroso, o porque las condiciones meteorológicas no permiten su uso.

· **Disponibilidad de los instructores** y los alumnos. En cursos de formación de pilotos hacen falta instructores, que algunas veces deben desplazarse hasta el lugar de las pruebas (un circuito, un puerto de mar, etc.), ya que para ciertos vehículos especiales la disponibilidad de instructores expertos puede ser baja.

Los simuladores, sin embargo, pueden construirse casi en cualquier lugar, acercando las pruebas tanto a instructores como a alumnos.

· **Posibilidad de provocar/recrear situaciones nuevas** o condiciones climatológicas a voluntad. Esto es, por ejemplo, importante en vehículos cuyo medio sea muy variable, como por ejemplo los vehículos marinos, donde podemos controlar el estado del mar, el viento, la visibilidad, etc.

· **Posibilidad de repetir las mismas pruebas** en las mismas condiciones siempre, de modo que permitamos realizar pruebas objetivamente iguales sobre distintas personas.

· **Capacidad para evaluar las pruebas objetivamente** e incluso mostrar al usuario sus errores desde otros puntos de vista.

· **Transferencia de habilidades y entrenamientos**. Si el simulador está correctamente diseñado, 1 hora de simulador puede sustituir a 1 hora de uso real.

Evidentemente, también tienen algunas desventajas. Entre ellas podíamos citar la complejidad de su construcción y sobre todo la imposibilidad de que la sensación simulada sea exactamente igual que la real. Ese es siempre el objetivo, pero es un objetivo que habitualmente resulta lejano.

1.3 – Componentes Básicos de un Simulador de Vehículos

Aunque cada simulador presenta unas necesidades específicas y requiere por tanto de unos componentes diferentes, los simuladores de vehículos se rigen por unos patrones similares entre sí, de manera que podemos identificar en ellos una serie de componentes básicos generales que en la gran mayoría de los casos son comunes a este tipo de aplicaciones. Es importante recalcar que todos los componentes deben ser capaces de funcionar en tiempo real, dado que el simulador tiene que reaccionar a las acciones del usuario sin que éste perciba ningún retraso. Para una mayor explicación sobre las restricciones de tiempo real de un simulador, consúltese [4]. Los componentes básicos que podemos identificar en un simulador de vehículos [27] son los siguientes (ver Figura 1.3):

· Un modelo visual y un sistema de visualización. Le llamaremos **sistema visual** e incluye tanto los dispositivos de visualización (pantallas, proyectores o monitores) como el *software* de generación de imágenes (por ejemplo [28]). Es importante que tanto el *software* como el *hardware* sean capaces de generar imágenes en tiempo real a una frecuencia tal que el usuario no aprecie discontinuidades en la imagen. Esta frecuencia se suele establecer en 50-60 Hz [4], límite que se considera suficiente para que el ser humano asimile la secuencia de imágenes a un movimiento fluido.

· Un modelo de vehículo, de las leyes físicas de éste y del entorno en que se mueve. Le llamaremos **sistema físico** y suele ser un módulo dedicado a resolver las ecuaciones diferenciales del

movimiento, ya que se ocupa de calcular posiciones, velocidades y aceleraciones virtuales [29], [30]. El desarrollo de paquetes *software* de cálculo de simulación física, como *Nvidia PhysX* [31], *Havok*, *ODE*, o *Bullet*, unido al desarrollo de *hardware* específico para este tipo de cálculo, ha simplificado enormemente el desarrollo de este tipo de módulos, y aunque sigue siendo una tarea costosa, se están logrando resultados satisfactorios tanto en precisión como estabilidad de los modelos físicos de vehículos en simuladores. Dado que este módulo tiene que ser capaz de calcular en tiempo real las soluciones de las ecuaciones diferenciales que reflejan la dinámica del vehículo, la frecuencia de funcionamiento de este módulo debe ser bastante alta (por encima de 100 Hz) ya que éstos modelos son propensos a la inestabilidad [32]. Frecuencias de actualización de 500 o incluso 1000 Hz son habituales.

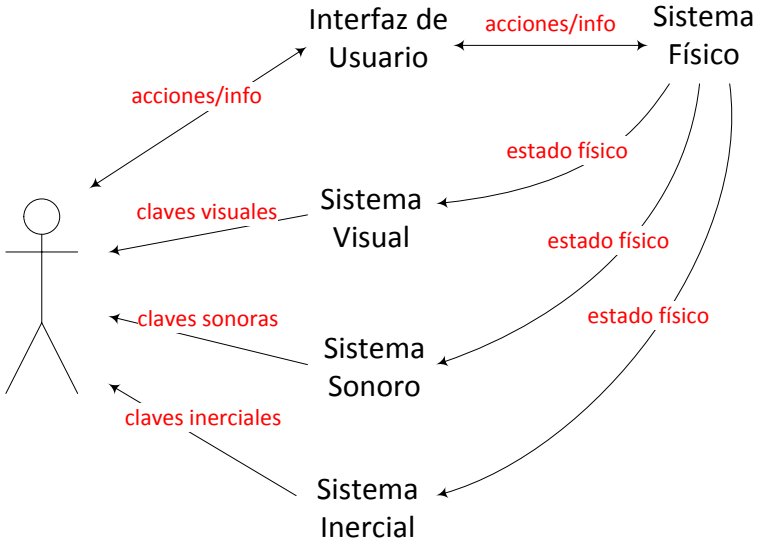

Figura 1.3 - *Esquema de un simulador de vehículos*

· Un sistema de interacción y sensorización de los mandos, palancas, e indicadores que se necesite emplear para manejar el vehículo. En aeronáutica se conoce a este conjunto como *cockpit*. En simuladores de conducción, correspondería con una reproducción del salpicadero, el volante, los pedales y el cambio de marchas. Este módulo incluye tanto la reproducción *hardware* del sistema de pilotaje del vehículo, como la captura e interpretación *software* de esos dispositivos, aunque la parte más compleja e importante corresponde a la capacidad de recrear fielmente dichos dispositivos. Le llamaremos **interfaz de usuario** y debe funcionar a una frecuencia de funcionamiento tal que el usuario no note que sus acciones se reciben con retraso. Dado que los tiempos de respuesta del ser humano suelen ser relativamente altos (del orden de décimas de segundo, lo cual en términos de computación es un tiempo elevado) basta con que este módulo funcione a unas pocas decenas de Hz para que el ser humano no aprecie ningún retraso en la respuesta de los pedales o mandos.

· Un sistema de generación de sonido, que llamaremos módulo de sonido o **sistema sonoro**. Se encarga de posicionar y reproducir los sonidos de forma tridimensional respecto del observador de la simulación. Su frecuencia de actualización suele ser similar a la del módulo visual.

· Un sistema capaz de generar movimiento en el sujeto protagonista de la simulación. Ésta es la parte del simulador que nos interesa en este trabajo. En inglés, este sistema recibe el nombre de *motion cueing system* y básicamente se compone de un dispositivo capaz de generar movimiento sobre el puesto de conducción, y un *software* que se encargue de controlar dicho

movimiento. Le llamaremos **sistema de generación gravito-inercial**, **sistema de estimulación inercial** o simplemente **sistema** o **módulo inercial**. La frecuencia de funcionamiento de este módulo depende del tipo de movimiento que se quiera generar y especialmente de la plataforma de movimiento que controle. Como normalmente las plataformas de movimiento no suelen ser capaces de generar movimientos de frecuencia muy alta (dado que suelen tener grandes inercias debido a su peso y a la carga que mueven), es frecuente que la frecuencia de funcionamiento oscile entre 50 y 100 Hz [33], aunque hay algunos actuadores capaces de llegar a frecuencias mucho más altas (hasta 400 Hz) [4] por lo que, en esos casos excepcionales, la frecuencia de funcionamiento del módulo podría ser mayor.

En ocasiones, existen otros módulos en un simulador. Uno de los más comunes es el **módulo de comunicación**, que permite al simulador comunicarse con el mundo exterior o con otras instancias remotas o incluso locales del mismo simulador. Aunque este módulo no está presente en todos los simuladores, su presencia es bastante habitual, especialmente si se trata de un simulador de tipo colaborativo, conocido en la literatura como DVE [34].

Otro ejemplo podría ser, en un simulador de conducción, el **módulo de inteligencia artificial** que se encarga de simular trayectorias para los vehículos no controlados por el usuario. Otro ejemplo podría ser el **módulo de ejecución y control de ejercicios** que suele estar presente en los simuladores de aprendizaje [35], donde lo que se pretende es ejecutar ciertas tareas

específicas con el vehículo para generar cierta transferencia de habilidad sobre el usuario del simulador.

1.4 – El Sistema Inercial

Aunque la adición de movimiento a los simuladores no es un hecho reciente, en los últimos años se ha producido un auge de la demanda de este tipo de módulos, debido a que un número creciente de organismos e instituciones de estandarización requieren la inclusión de este tipo de elementos para su homologación en determinadas clases de simuladores [36].

Para la generación de movimiento en simuladores, la estrategia más comúnmente usada es situar todo el simulador (pantallas, asiento, usuario, mandos, etc.) sobre un dispositivo capaz de mover todo el conjunto de forma controlada mediante un *software* específicamente diseñado al efecto. Este dispositivo recibe el nombre de plataforma de movimiento (ver Figura 1.4). Algunas veces, si la pantalla es muy grande, ésta no se sitúa encima de la plataforma de movimiento. En este caso hay que realizar correcciones para que el movimiento de la plataforma no genere un cambio de perspectiva visual por parte del piloto. Esto no es excesivamente problemático cuando la pantalla es grande pero puede serlo si es de pequeño tamaño.

Existen múltiples diseños y opciones para construir una plataforma de movimiento, pero el objetivo fundamental de todas ellas es el mismo: conseguir la mayor capacidad de reproducción de movimiento, a ser posible, con el menor coste.

En realidad, una plataforma de movimiento no deja de ser un dispositivo robótico, por lo que todas ellas presentan ciertas características comunes estudiadas por la robótica clásica. En primer lugar, suelen ser dispositivos electromecánicos controlados por ordenador. En segundo lugar, suelen presentarse en forma de base plana debajo de la cual se sitúan un número indeterminado de actuadores que son capaces de mover controladamente la base, de tal forma que se generen sobre ella aceleraciones lineales y angulares. Existen actuadores hidráulicos, neumáticos, mecánicos, aunque lo más común es que sean motores eléctricos.

Figura 1.4 - *Ejemplo de plataforma de movimiento (Fuente: Car-D.fr)*

Los actuadores de la plataforma tienen como objetivo mover en la dirección e intensidad adecuada el dispositivo, por lo que las plataformas de movimiento se suelen clasificar según los tipos de movimientos que pueden realizar. Éstos se conocen en la literatura como grados de libertad (*degrees of freedom*, DoF, en inglés). Existen 6 posibles grados de libertad (GdL) que se corresponden con las 3 traslaciones y 3 rotaciones posibles en el espacio. Los 6 GdL son los siguientes (ver Figura 1.5):

· **Traslación** a lo largo del **eje X**. Este movimiento es el que se considera como desplazamiento lateral (izquierda-derecha) o ***sway***.

· **Traslación** a lo largo del **eje Y**. Este movimiento es el que se considera como movimiento longitudinal (delante-atrás) o ***surge***.

· **Traslación** a lo largo del **eje Z** (desplazamiento vertical arriba-abajo) conocida como ***heave***.

· **Rotación** alrededor del **eje X**. Corresponde a la inclinación hacia delante o hacia atrás, conocida como ***pitch***, o cabeceo.

· **Rotación** alrededor del **eje Y**, conocida como ***roll***, o alabeo. Corresponde a la inclinación lateral.

· **Rotación** alrededor del **eje Z**, conocida como ***yaw***, *heading* o deriva. Corresponde al rumbo del movimiento sobre el plano del suelo.

Es importante comentar que los nombres y direcciones de movimiento pueden cambiar de un simulador a otro, ya que son convenciones arbitrarias, aunque la convención antedicha es la más habitual.

Según las necesidades del simulador se construyen plataformas desde 1 hasta 6 GdL, por lo que existen múltiples tipos de plataforma correspondientes con las múltiples combinaciones de GdL que se puede escoger.

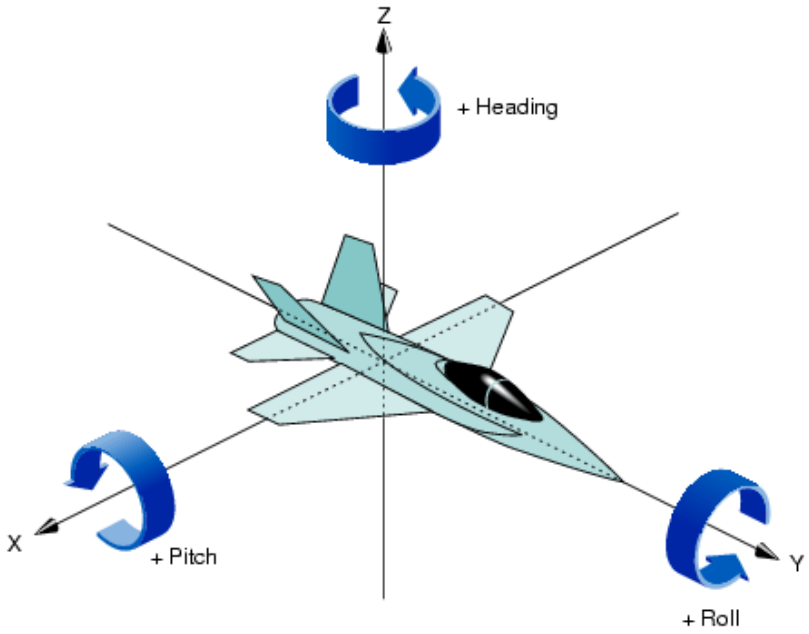

Figura 1.5 - *Los 6 grados de libertad del espacio cartesiano (© SGI)*

Las plataformas de movimiento también se pueden clasificar según el tipo de actuadores que empleen. Las hay que usan

actuadores traslacionales (pistones) y las hay que usan actuadores rotacionales (generalmente motores con bielas). Asimismo, los actuadores pueden ser hidráulicos, neumáticos, motores eléctricos de corriente continua, motores eléctricos de corriente alterna, etc. Cada tipo de actuador presenta unas características diferentes en cuanto a fuerza, precisión, controlabilidad, repetibilidad, capacidad de recuperación y frecuencia de funcionamiento. De todos estos problemas se ocupa la robótica clásica, y no entraremos a detallarlos.

1.5 – Problemática Asociada a las Plataformas de Movimiento

El primer problema de las plataformas de movimiento es que a mayor número de GdL, la complejidad del dispositivo es mayor, y el coste económico también aumenta. Las plataformas de 1 GdL son poco comunes por ser demasiado simples. Sin embargo, las de 6 GdL son comunes pero a menudo son demasiado costosas en términos económicos. Las de 5 no suelen construirse porque el diseño suele ser igual de complejo o más que las de 6 pero tienen 1 GdL menos. Las plataformas de 2, 3 y 4 GdL se emplean más a menudo porque en ocasiones consiguen un equilibrio entre el coste y complejidad de construirlas y su funcionalidad.

El segundo problema que se nos plantea es que la fuerza, y por tanto la aceleración, que pueden generar las plataformas de movimiento es proporcional a su coste. A mayor intensidad de movimiento deseada, más coste tienen los actuadores capaces de

generarlo. Lo mismo sucede con la precisión y controlabilidad de los actuadores.

El tercer problema, y probablemente el más importante, es el de la extensión o longitud de los movimientos. Como es lógico, ninguna plataforma de movimiento permite un movimiento ilimitado, lo cual hace que la generación de los movimientos sea habitualmente bastante diferente a la real. Por tanto, cualquier solución será inevitablemente subóptima, ya que es complicado (y a veces imposible) conseguir reflejar las sensaciones originales del vehículo y al mismo tiempo mantenerse dentro de los límites físicos de la plataforma. Es más, la mayoría de ellas están restringidas, por causas mecánicas y de espacio, a unos pocos centímetros o decímetros de movimiento. Evidentemente, a mayor rango de movimiento, mayor coste en el diseño y construcción de la plataforma. A veces, incluso aunque el coste fuera asumible, puede que no haya espacio suficiente para el dispositivo por cuestiones meramente físicas, si se busca que éste permita desplazamientos muy grandes.

Este problema hace que el *software* que controla y dirige los movimientos de la plataforma tenga que intentar reproducir movimientos virtuales potencialmente ilimitados (como por ejemplo el desplazamiento de un coche o un avión por el espacio) mediante un dispositivo cuyo rango efectivo de desplazamiento es, en el mejor de los casos, del orden de metros. Como el lector puede comprender, este problema no tiene una solución sencilla. Cualquier solución que adoptemos dará lugar a un movimiento similar al que se desea simular, pero probablemente no igual. Esto es algo que debemos tener siempre presente.

El cuarto problema, y relacionado con los anteriores, es que el ajuste de parámetros de cada una de las capas de *software* que controlan el movimiento de la plataforma es costoso, dado que el problema es subjetivo y carece, en general, de una única solución correcta que podamos tomar como referencia. Además, dado que cada sujeto puede percibir esa diferencia con el movimiento real de manera diferente, la comparación entre diferentes soluciones es compleja, lo cual dificulta tanto la tarea de ajuste de los algoritmos como la capacidad para decidir qué algoritmo es el mejor o el más apropiado para cada aplicación.

Este último problema, el del ajuste de los parámetros, está fuertemente relacionado con la falta de un criterio para poder aseverar que la fidelidad en la generación de movimiento de un simulador es mayor que la de otro. Es sobre estos aspectos, entre otros, sobre los que este trabajo dirige sus esfuerzos.

Todos estos problemas hacen que algunos simuladores renuncien a incluir este tipo de claves, ya que la inclusión medianamente creíble de las mismas implica unos costes muy altos, mientras que el uso de un dispositivo con movimientos limitados y/o mal ajustado es muchas veces peor que su no uso. Tanto es así que, excepto las grandes escuderías de Fórmula 1 y alguna otra excepción, la mayoría de equipos dedicados a la competición automovilística y motociclística, suelen disponer de simuladores exclusivamente visuales para entrenar a sus pilotos.

Lo ideal a la hora de construir una plataforma de movimiento es permitir movimientos en los 6 GdL. Es por ello que las plataformas de 6 GdL son la referencia básica. De hecho, el

diseño más común es el propuesto por Stewart [37]. Sin embargo, no siempre es necesario disponer de 6 GdL por el coste que esto supone. A veces es mejor construir plataformas de menos GdL pero con mayor extensión o potencia de movimiento en cada uno de los ejes.

De hecho, la mayoría de las plataformas de movimiento suelen presentar una característica especialmente molesta para el programador del *software* de movimiento, que consiste en que las extensiones de los GdL no suelen ser independientes unas de otras, ya que la extensión completa de un GdL impide la extensión completa de otros de tal modo que aunque ambos GdL pueden alcanzar sus rangos completos, no lo pueden hacer simultáneamente [38], [39]. Esto no siempre ocurre, pero en la mayor parte de los manipuladores paralelos (como por ejemplo el diseño de Stewart) es así.

Para una mayor descripción de los manipuladores robóticos de este tipo, sus características, problemas y ventajas, el lector puede consultar [40] y [41].

Capítulo 2
Estado del Arte

En este capítulo trataremos de revisar el estado actual del conocimiento sobre la percepción y generación de claves de movimiento.

2.1 – Cinestesia y Claves Perceptivas

Como hemos comentado con anterioridad, el objetivo fundamental de la RV es proporcionar al usuario una sensación convincente de pertenencia a un mundo virtual alternativo. Para conseguir este objetivo, toda aplicación de RV debe generar o simular del modo más fiel posible las sensaciones que el usuario percibiría en una situación real.

En general, hay dos aspectos básicos a tener en cuenta en un sistema de RV [42]. El primero es cómo se perciben los estímulos (generados por el entorno virtual) por parte del usuario. Esta comunicación se produce desde el simulador hacia el humano, por lo que se corresponde con las salidas del simulador. El segundo aspecto es cómo informa el usuario a la aplicación de qué quiere hacer o cómo y a dónde quiere moverse. Esta comunicación se produce en sentido inverso al anterior, del humano a la máquina. Aunque ambos aspectos son complejos, el segundo puede llegar a ser particularmente dificultoso.

En el caso particular de una simulación de vehículos, el humano actúa normalmente sobre palancas o mandos del vehículo, por lo que la recreación de este último tipo de comunicación se puede simplificar bastante respecto a otras aplicaciones de RV. En dichos simuladores, las acciones sobre el vehículo son normalmente las únicas acciones posibles. En otras aplicaciones de RV, donde el usuario camina o realiza movimientos gestuales, la interacción se vuelve mucho más compleja. Por fortuna, en un simulador de un vehículo, normalmente basta con sensorizar los mandos y palancas para que las acciones del usuario sean leídas por la aplicación de RV.

Sin embargo, en simuladores de vehículos, la comunicación en sentido del simulador hacia el humano no es menos compleja que en otro tipo de aplicaciones de RV. Dado que lo que nos interesa es analizar las claves gravito-inerciales en simuladores de vehículos, analizaremos sólo este tipo de comunicación (del simulador al humano) y estudiaremos las distintas claves que generan sensación de inmersión en una simulación, pero siempre con relación a la percepción de movimiento, que es lo que nos interesa en este trabajo.

La percepción del movimiento recibe el nombre genérico de cinestesia o kinestesia. La etimología de la palabra refleja su significado, ya que el vocablo proviene de las palabras griegas *kínesis*, que significa movimiento, y *áisthesis*, que significa percepción. La cinestesia es, por tanto, la ciencia que estudia el conjunto de procesos que realiza el cuerpo humano para procesar y entender su propio movimiento.

Cuando nos desplazamos por el mundo, no somos conscientes del alcance y complejidad de nuestro sistema de detección y control del movimiento. Andamos de un sitio a otro y generalmente cambiamos de dirección en el momento exacto deseado, paramos dónde corresponde, vamos a la velocidad que elegimos, etc. ¿Cómo hacemos que esta tarea tan compleja parezca tan sencilla? A continuación estudiaremos cada una de las claves que aportan al cerebro la información necesaria para realizar este complejo trabajo.

2.1.1 – Claves Visuales

Varios sentidos proporcionan información sobre la sensación de movimiento, en especial la vista, el oído y en menor medida el tacto. Aunque es difícil estimar cuál es la proporción de influencia de la información visual, se estima que la percepción visual es la clave dominante en la percepción del movimiento [43], por lo que prácticamente no se concibe en la actualidad un simulador de vehículos que no incluya información visual.

Las claves visuales corresponden a la percepción del movimiento relativo entre el observador y los objetos que le rodean, mediante la utilización del sentido de la vista. Esto quiere decir que, a nivel visual, es indistinguible que el observador se mueva en una dirección o que los objetos se muevan en la misma dirección pero sentido opuesto.

Se han identificado dos claves visuales que ayudan a la detección del movimiento: el desplazamiento y el flujo óptico [42].

El desplazamiento se refiere al hecho de que al movernos, las posiciones de algunos objetos o ciertas características identificables del entorno cambian de posición, y el cerebro es capaz de inferir cuánto y cómo nos hemos movido por el cambio de posición de dichos elementos.

El flujo óptico ha recibido mucha atención en los últimos años. En visión por computador, el flujo óptico es un campo de velocidad asociado con cambios de imágenes (ver Figura 2.1), que se define como el patrón del movimiento aparente de los objetos, superficies, bordes y colores en una escena causado por el movimiento relativo entre un observador (un ojo o una cámara) y la escena [44], [45]. Diversos experimentos han estimado que el cerebro humano es capaz de inferir diferencias en distancias recorridas y percibir diferencias en la velocidad del movimiento, a partir del flujo óptico [46]. Además, a partir del flujo óptico, el cerebro es capaz de encontrar la dirección egocéntrica, o lo que es lo mismo, la dirección en la que se produce el movimiento del observador [47]. Sin embargo, aún no se ha podido demostrar que el ser humano sea capaz de estimar con exactitud la distancia recorrida a partir del flujo óptico. Lo que sí parece claro es que, independientemente de la precisión de la información que se pueda extraer de él, el flujo óptico es una de las claves dominantes en la percepción del movimiento.

Dentro de las claves visuales no podemos olvidarnos de las claves de profundidad (también llamadas claves binoculares), que se estimulan por la capacidad que tiene el ser humano de estimar la distancia a los objetos por la diferente apariencia que estos tienen al ser observados desde dos puntos de vista ligeramente distintos

pero simultáneos (nuestros dos ojos). Esta clave recibe del nombre de estereoscopia, y existe una amplia literatura al respecto [48], [49], [50].

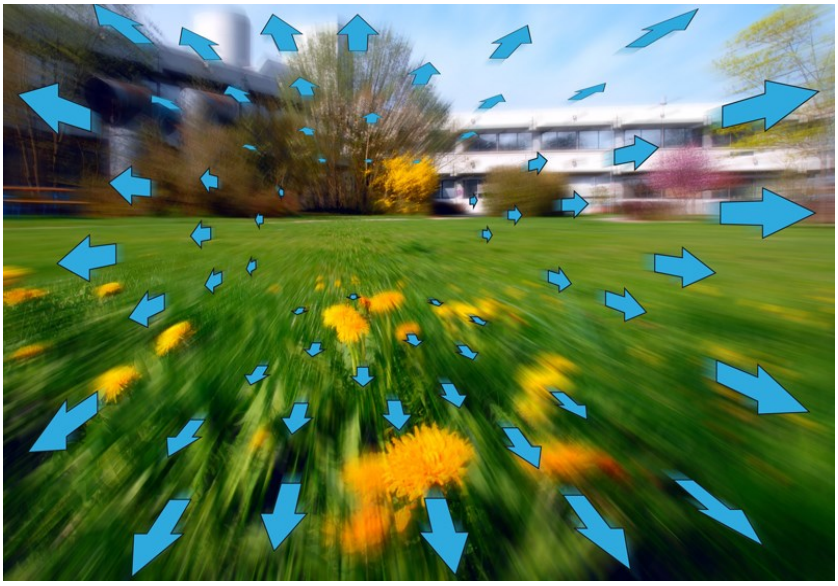

Figura 2.1 - *El flujo óptico (© Max Planck Institute of Neurobiology)*

Algunos simuladores de vehículos incluyen estereoscopia en su módulo visual, aunque no en una gran mayoría. Las razones varían desde motivos económicos, perceptuales o incluso de rechazo social (hay un porcentaje no despreciable de población que inevitablemente se siente incómoda con este tipo de tecnología). Es de esperar que en los próximos años, con los avances que se están produciendo en este tema (sistemas auto-estereoscópicos, sistemas con distancia de acomodación variable, etc.), su implantación crezca.

En cuanto a la reproducción de imágenes, dado que el *hardware* ha evolucionado mucho y continúa evolucionando, los dispositivos actuales son capaces de generar imágenes con una frecuencia y resolución más allá de lo que el ojo humano es capaz de percibir. Es por ello, que, aunque la generación realista de imágenes y movimiento visual sigue teniendo margen de mejora, no es, ni mucho menos, la parte de los simuladores que necesita más avances.

2.1.2 – Claves Sonoras

Las claves sonoras han recibido menos atención que las visuales, y aunque no siempre proporcionan una percepción directa del movimiento, ni de la posición del observador, suelen ofrecer información valiosa en el caso de los simuladores de vehículos.

Dado que muchos, por no decir todos, de los vehículos que se suelen simular son vehículos a motor, la correcta percepción del sonido, especialmente el del motor, proporciona una valiosa información al usuario sobre el estado de movimiento del vehículo [4]. Para el caso de los vehículos sin motor, el movimiento siempre genera algún tipo de fricción entre objetos que, en última instancia, genera algún sonido que puede guiar al cerebro a entender el movimiento que ocurre a su alrededor.

Además, desde hace unos años, la simulación de sonido tridimensional o binaural - sonido a partir del cual el cerebro es capaz de inferir la posición de la fuente de sonido – introduce una

clave sonora más en la simulación, ya que no sólo somos capaces de escuchar el sonido del motor y de otros vehículos, sino que además somos capaces de distinguir de qué posición viene cada uno de ellos, lo cual nos permite tener más información sobre nuestra posición y la de los objetos que nos rodean [51].

Hoy en día prácticamente todos los simuladores incluyen claves sonoras tridimensionales muy realistas, dado que la grabación y reproducción de los sonidos ha llegado a cotas muy realistas, el coste computacional de su reproducción no es demasiado elevado, y las distintas bibliotecas *software* dedicadas a la reproducción de sonido binaural facilitan enormemente la tarea.

2.1.3 – Claves Predictivas

Recientemente, se ha descubierto que muchos animales generan copias de las acciones que su cerebro ordena. Esta copia se guarda para generar predicciones sobre las consecuencias que las acciones ordenadas tendrán sobre el cuerpo y sobre su entorno. Este concepto, conocido como **copia eferente** [42], es una parte importante de la percepción del movimiento, aunque muchas veces se ignora. La importancia del mismo radica en que, al tener una copia de los comandos que el cerebro ha ordenado, permite al mismo prepararse para las consecuencias de los mismos, de tal modo que si existe diferencia entre el movimiento experimentado y el deseado/ordenado, el cerebro puede percibir la sensación como incorrecta. Sin embargo, lo más importante de la copia eferente es que permite al cerebro distinguir el movimiento experimentado por el cuerpo consecuencia de sus propias

acciones, del movimiento experimentado por causas exógenas no directamente provocadas por él.

Aunque la copia eferente es un concepto difícil de medir, todas las evidencias apuntan a que una parte del conocido mareo de simulador (*simulator sickness*) es producido por una disparidad entre las predicciones realizadas por el cerebro y las sensaciones efectivamente experimentadas, que, debido a retrasos y diferencias en la reproducción de las mismas por parte del simulador, llegan incorrectamente o más tarde de lo esperado al cerebro [52]. Por ello, es importante intentar reproducir lo más fielmente posible todos los elementos sobre los que el usuario puede ejecutar órdenes en el simulador y, al mismo tiempo, simular rápidamente las consecuencias de esas órdenes. Afortunadamente, gracias a los últimos avances en simulación física, el comportamiento y las reacciones de los vehículos virtuales con respecto a sus palancas y actuadores son cada vez más realistas.

2.1.4 – Claves Gravito-inerciales

El cuerpo humano, al igual que muchos animales, está diseñado para ser capaz de detectar las fuerzas que actúan sobre él. Para ello dispone de una serie de sistemas sensoriales sensibles a diferentes magnitudes físicas. El más importante de estos sistemas es el **sistema vestibular** [53] situado en el oído interno, que es capaz de detectar aceleraciones (y por tanto fuerzas), orientaciones y cambios de orientación (velocidades angulares). Este sistema no es capaz de detectar el movimiento rectilíneo uniforme, ya que está sujeto a las leyes de Newton y éste es dinámicamente indistinguible

del reposo, pero sí es capaz de detectar la dirección de la fuerza de la gravedad, incluso aunque ésta sea contrarrestada y no provoque movimiento. La detección correcta de la dirección del vector gravedad con respecto al cuerpo es la que permite al cerebro estimar cuál es la orientación de la cabeza con respecto al suelo.

Dado que el sistema vestibular juega un papel central en la percepción de las claves gravito-inerciales, otros autores se refieren a las claves gravito-inerciales como **claves vestibulares**, nombre que intentamos evitar, dado que la detección de este tipo de claves es multimodal y no sólo depende del sistema vestibular. Como comentaremos posteriormente, parece que también existen graviceptores (sensores de gravedad) de este tipo en el riñón, y además, sensores de otro tipo, como los sensores de presión de la piel o los de los órganos internos, también son estimulados cuando se producen ciertos tipos de movimiento. En particular, el sistema somatosensorial [54] incluye una serie de mecano-receptores que detectan presión en la piel, en los músculos, en las articulaciones, etc. Cuando un individuo sentado en un vehículo a motor acelera dicho vehículo, una de las sensaciones que notará es la de presión contra el asiento. Esta pista puede ser más importante que la señal proporcionada por los sensores de aceleración del oído interno.

Como hemos comentado anteriormente, aunque la generación de movimiento virtual puede llegar a ser muy realista, desgraciadamente, la recreación de verdaderas sensaciones de movimiento que estimulen las claves gravito-inerciales dista todavía de ser satisfactoria en los simuladores actuales.

2.1.5 – Claves Propioceptivas

Las claves propioceptivas son las que se refieren al conocimiento que tiene el propio cuerpo sobre la posición de cada una de sus partes. Este conocimiento proviene de una serie de mecano-receptores situados en las articulaciones y músculos, de tal modo que el cuerpo es capaz de procesar esta información y reconstruir, sin ayuda visual, la posición de cada una de sus partes. La propiocepción es una clave importante en el movimiento de caminar, ir en bicicleta o cualquier otra actividad que requiera un movimiento constante de las articulaciones, y aunque en simuladores de vehículos a motor no suele haber grandes necesidades propioceptivas, es algo que debemos tener en mente.

Es importante hacer notar que muchas veces los términos propiocepción y cinestesia se usan como sinónimos, y no hay acuerdo sobre el alcance de cada uno de los términos. Algunos autores excluyen de la cinestesia las claves vestibulares y hacen coincidir cinestesia y propiocepción. Dado que consideramos que es una cuestión puramente semántica, no alimentaremos más la discusión.

2.1.6 – Otras Claves

Existen otras claves menos importantes de cara a la percepción del movimiento, y año a año se van descubriendo nuevas contribuciones sensoriales al mismo. Entre ellas podemos encontrar las claves hápticas (las referidas al contacto), las oculomotoras (referidas al enfoque de un objeto por parte del

cristalino), las claves térmicas, la nociocepción (referida al dolor), etc.

2.2 – Claves Gravito-inerciales

2.2.1 – Mecanismo de Percepción Gravito-inercial Humano

Como hemos comentado, existen varios mecanismos que contribuyen a la percepción gravito-inercial, es decir a la percepción de la gravedad y de los cambios de la cantidad de movimiento lineal o angular de nuestro cuerpo. Básicamente, estos mecanismos se pueden dividir en dos: el sistema vestibular situado en el oído interno (ver Figura 2.2) y el resto de sensores distribuidos por el cuerpo.

El sistema vestibular es un sistema perfectamente diseñado para la sensorización del movimiento y el control de la navegación del cuerpo humano. Aunque no se conocen con exactitud todos los procesos que ejecuta, sí se conoce el mecanismo fundamental de su funcionamiento.

El sistema vestibular se compone de dos sistemas sensoriales principales: los canales semicirculares, sensibles al movimiento de rotación, y los otolitos, sensibles a la aceleración lineal [53]. Tanto los canales (o conductos) semicirculares como los otolitos actúan como transductores, ya que son capaces de transformar una magnitud física (aceleración o velocidad angular, según el caso) en una señal eléctrica con información para el cerebro, que utiliza estos datos para generar la percepción global del movimiento [55].

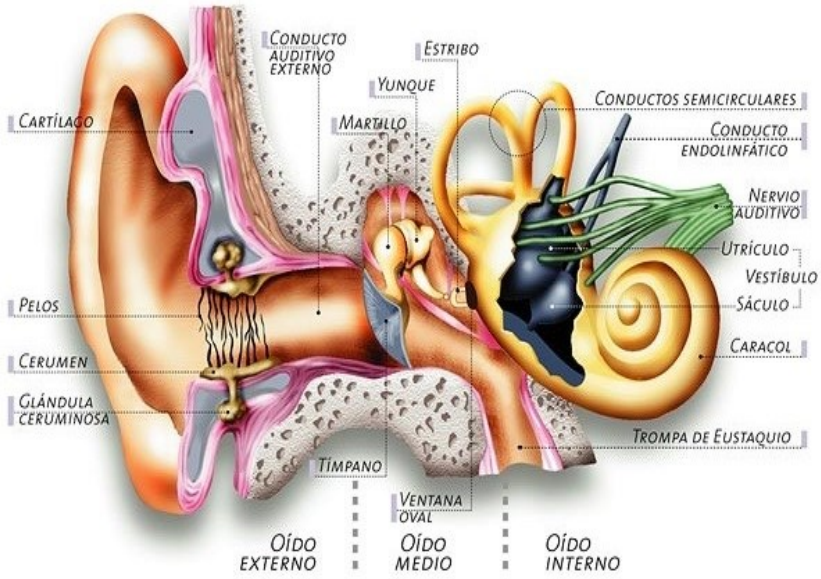

Figura 2.2 - *El oído y sus partes (Fuente: Diario El País)*

2.2.2 – Canales Semicirculares

Dentro del oído interno, como parte del sistema vestibular, se encuentran los canales semicirculares. En esencia, los canales semicirculares son 3 tubos en forma semicircular colocados aproximadamente de forma perpendicular cada uno con respecto a los otros dos siguiendo las 3 direcciones del espacio (ver Figura 2.2). Cada uno de los canales contiene un fluido llamado endolinfa, que debido al movimiento rotacional se desplaza por el interior del canal (ver Figura 2.3). Este desplazamiento produce cambios de presión en una membrana llamada cúpula, y esos estímulos son detectados por unos transductores ciliares (estereocilios y cinocilios) [53] que transforman esa información en señales

eléctricas que el cerebro posteriormente usará para inferir la velocidad angular y la orientación de la cabeza con respecto a la gravedad [47].

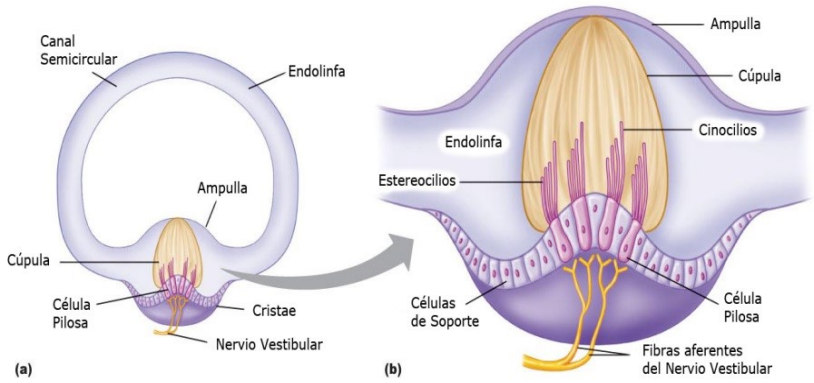

Figura 2.3 - *Detalle de un canal semicircular (© Pearson Education)*

Diversos estudios han comprobado que los canales semicirculares son sensibles a la velocidad angular, no a la aceleración angular. Por ello, a partir únicamente de la información proporcionada por los canales semicirculares sólo se puede saber si la cabeza está siendo girada, pero no se puede saber cuál es su orientación estática exacta. Para ello, hace falta más información.

2.2.3 – Otolitos

Esta información faltante se obtiene, entre otras cosas, a partir de los otolitos. Los otolitos son pequeños órganos que se componen de unas diminutas piedras llamadas otoconias situadas

sobre una membrana gelatinosa que dispone de unas células ciliadas (cinocilios y estereocilios similares a los presentes en los canales semicirculares), que son capaces de detectar el movimiento de las otoconias (ver Figura 2.4). La inercia del movimiento sufrido por el cuerpo hace que estas piedras se desplacen y el contacto con los cilios causa una deflexión en los mismos que estimula el nervio vestibular en forma de señal eléctrica que informa al cerebro sobre la aceleración sufrida por la cabeza. La aceleración que miden los otolitos es la resultante de todas las fuerzas que se ejercen sobre la cabeza (gravedad incluida). A esta fuerza resultante se la conoce como **fuerza gravito-inercial**, de ahí que a las claves de detección de movimiento se les llame claves gravito-inerciales.

El ser humano presenta dos de estos mecanismos en cada oído: el utrículo y el sáculo (la mácula otolítica utricular y la mácula otolítica sacular [53]). Entre los 4, el cerebro es capaz de detectar la aceleración de la cabeza.

Como acabamos de comentar, este mecanismo, sin embargo, no es sólo sensible a la aceleración producida por el movimiento de la cabeza, sino que también es sensible a la gravedad, incluso aunque ésta no produzca ningún movimiento en la cabeza. La razón es muy simple: las otoconias se mueven libremente dentro del utrículo y del sáculo, por lo que, aunque la cabeza esté sujeta por otra fuerza que anule el efecto de la gravedad, las otoconias sufren la atracción gravitatoria, por lo que se desplazan por los otolitos y el cerebro es capaz de detectar la dirección de la gravedad. Este hecho es de vital importancia porque permite al cerebro detectar la orientación en reposo de la cabeza por medio de una estimación de la dirección del vector gravedad, dado que

ésta es constante en todo momento (al menos en la superficie de La Tierra).

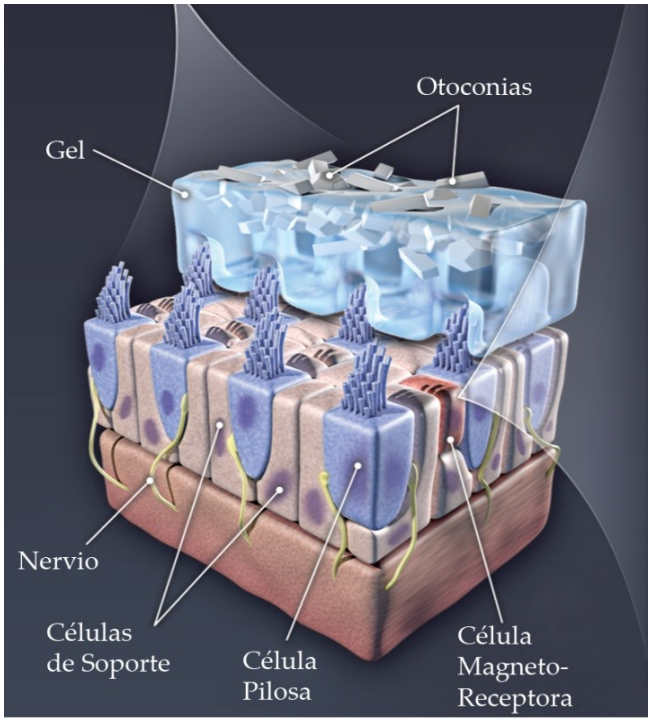

Figura 2.4 - *Detalle de un otolito (Fuente: Science Magazine)*

Sin embargo, esta estimación propioceptiva se vuelve errónea cuando la cabeza está en movimiento, dado que los otolitos son incapaces de diferenciar entre aceleraciones gravitatorias y no gravitatorias, entre otras cosas porque por el principio de equivalencia de Einstein [56] ambas son físicamente indistinguibles. Aunque la copia eferente nos puede ayudar a diferenciar qué parte de la detección del movimiento es consecuencia de nuestro movimiento (nuestras órdenes), y qué

parte es consecuencia de la atracción gravitatoria, en general no seremos siempre capaces de diferenciar ambas contribuciones satisfactoriamente sin más información.

Por ello, dado que la aceleración de la gravedad, siempre presente en nuestro planeta, no se puede eliminar del proceso de sensorización y se mezcla con el resto de aceleraciones, se considera, en sentido estricto, que los sensores inerciales de nuestro cuerpo miden la fuerza específica ($\overrightarrow{F_s}$), en lugar de la aceleración de la cabeza ($\vec{a}$). Esta magnitud se define como

$$\overrightarrow{F_s} = \vec{a} - \vec{g} \qquad (2.1)$$

donde $\vec{a}$ es la aceleración sensorizada (la resultante de la suma de todas las aceleraciones) y $\vec{g}$ es el vector gravedad [57]. Todo ello medido sobre un sistema de coordenadas situado en la cabeza. A pesar del nombre, la fuerza específica es una medida de aceleración y no una medida de fuerza; en concreto, de aceleración local relativa a la situación de caída libre. De hecho, los acelerómetros digitales funcionan de manera similar al cuerpo humano, dando una medida de 1 g cuando el cuerpo está en reposo o en movimiento lineal constante (la gravedad es contrarrestada por alguna fuerza que nos hace estar en reposo, pero las otoconias se moverían libremente y medirían 1 g) y un valor de 0 g en caída libre (el cuerpo se mueve, pero al ser la aceleración del cuerpo igual a la de la gravedad, las otoconias no se mueven respecto de los cilios de los otolitos y la aceleración medida es 0 g). En cualquier caso, en muchas ocasiones, abusando del lenguaje, se dice que el cuerpo humano es sensible a la

aceleración, y dado que la fuerza específica es, en realidad, una aceleración, se considera admisible.

Afortunadamente, la combinación de la información otolítica y la proporcionada por los conductos semicirculares permite una monitorización constante de la orientación de la cabeza. Los otolitos proporcionan información sobre la orientación en reposo de la cabeza, y los canales semicirculares proporcionan información sobre la velocidad de rotación de la misma, permitiendo actualizar dicha orientación a partir de la última información válida proporcionada por los otolitos.

El hecho de que los otolitos sean incapaces de diferenciar una aceleración no gravitatoria de una gravitatoria, dado que lo único que sensorizan es la aceleración resultante, es un hecho de especial importancia a la hora del diseño de simuladores. Este hecho permite que una pequeña inclinación (estática, sin rotación) hacia delante o hacia atrás pueda ser interpretada por el sistema vestibular como una aceleración lineal hacia adelante o hacia atrás respectivamente. Este efecto se conoce con el nombre de **ilusión somatográvica** [58], que posteriormente explicaremos. Conocer este efecto permite engañar al cerebro generando inclinaciones que pueden ser interpretadas como traslaciones. Sin embargo, es importante conocer los límites del fenómeno ya que, al parecer, el cerebro es capaz de desambiguar esta ilusión mediante el análisis de otras señales, especialmente visuales [58]. En ausencia de estímulos visuales, se cree que el cerebro tiende a interpretar la información vestibular de modo estadístico, de modo que se le presupone una cierta tendencia a interpretar la información

sensorial como procedente de aceleraciones lineales y velocidades angulares bajas, ya que asume que son más probables [59].

2.2.4 – Otros Mecanismos

Aunque la mayoría de los estudios sobre la percepción de claves gravito-inerciales se centran en el sistema vestibular y su eficiente mecanismo de funcionamiento, lo cierto es que el cuerpo humano dispone de otros sensores repartidos por el cuerpo que son capaces de proporcionar información sobre la cantidad de aceleración sufrida por el mismo. Además, el cerebro debe integrar la información proveniente de diferentes sensores [60], [58], [59] cosa que permite superar las posibles limitaciones de cada uno de ellos.

Recientes experimentos en pacientes nefrectomizados sugieren claramente que en el área del riñón existen graviceptores capaces de detectar cambios en la presión hidrostática de la sangre o en la inercia de la masa visceral [61]. No se conoce con precisión el alcance de la importancia de estos sensores, pero se intuye que el cuerpo humano dispone de información multisensorial (no sólo a partir del sistema vestibular) para la percepción de la aceleración, y que ésta es mucho mayor de lo que se creía.

Otro ejemplo de esta multimodalidad es la detección de la dirección del movimiento por la presión o velocidad del aire sobre la piel cuando nos movemos en un vehículo sin parabrisas. Es conocido que la piel actúa como un gran sensor capaz de proporcionar múltiples datos. Otro de estos datos que puede servir

para la detección de la intensidad y dirección del movimiento es la presión. Dado que la mayoría de vehículos se dirigen desde un asiento, la presión ejercida por el asiento según la dirección o intensidad de movimiento del vehículo es una clave importante para estimar la aceleración. Las transferencias de peso en aceleraciones tanto longitudinales como laterales (frenazos y giros bruscos) producidas por el principio de inercia (1ª ley de Newton), son sentidas por la piel al entrar en contacto con el asiento y también por el movimiento de los órganos internos.

Otro de los mecanismos que tiene el cuerpo humano para sensorizar la aceleración es la fuerza muscular. Por ejemplo, la fuerza necesaria para que el músculo esternocleidomastoideo mantenga recta la cabeza es una clave para deducir la aceleración lateral de la misma, cuando por la acción de la fuerza centrífuga, ésta se inclina lateralmente [62].

Experimentos realizados sobre pacientes con lesiones vestibulares sugieren que, aunque la pérdida de capacidad de orientación es grande cuando se prescinde del sistema vestibular, una parte de la capacidad de detección permanece presente [63].

2.2.5 – Modelos de Percepción

Dado que se han realizado numerosos experimentos a lo largo de los años para intentar comprender cómo funcionan los mecanismos de percepción humanos, se han desarrollado junto a ellos algunos modelos de percepción que representan, de una u

otra manera, cómo procesa o emplea el cerebro la información relativa a su propio movimiento.

Algunos de esos modelos son multimodales (integrando por ejemplo mecanismos visuales y vestibulares de forma conjunta) mientras que otros son muy específicos y tratan sólo de un sentido concreto o incluso de algún sensor específico del cuerpo, como los otolitos o los canales semicirculares. También existen modelos para describir cómo realiza el ser humano diversas tareas específicas, como pueden ser la conducción de coches, la realización de maniobras concretas sobre una aeronave, etc.

Algunos de los modelos utilizados actualmente datan de los años 60 y 70 [64], [65], [66], ya que durante esta época se despertó gran interés por este tema. Por ejemplo, uno de los modelos utilizados por Pool [67] para el sistema visual es de McRuer, que data de 1965 [68]. Existen modelos más recientes como el empleado por Hosman en [69], como los de Wentik [70], o los presentados en [71] y [72].

Los ejemplos anteriores son modelos generales. Ejemplos de modelos específicos de diversos sensores los podemos encontrar, por ejemplo, en [57]. Para el desarrollo de su trabajo, Reid y Nahon emplearon un modelo de otolito y un modelo de canal semicircular.

El modelo de otolito empleado por Reid y Nahon, en forma de función de transferencia laplaciana, es el siguiente:

$$\frac{\hat{f}}{f} = \frac{K \cdot (\tau_a s + 1)}{(\tau_l s + 1) \cdot (\tau_s s + 1)} \qquad (2.2)$$

donde $\hat{f}$ sería la fuerza específica percibida por el cuerpo, f sería la fuerza específica real, y los parámetros del sistema son K, τ_a, τ_l y τ_s.

Este modelo es un modelo lineal de segundo orden que asume que la respuesta de los otolitos en las 3 direcciones principales del movimiento presenta la misma forma aunque con distintos parámetros (4 parámetros en este caso). El modelo también incluye los umbrales mínimos de percepción por debajo de los cuales no se detecta la aceleración (no visibles en la ecuación anterior porque se calculan en un paso intermedio [57]), por lo que el sistema completo tendría 5 parámetros: los 4 citados más un quinto denominado d_{th} (*detection threshold*) que se mediría en m/s^2. El diagrama de bloques del modelo completo, y su respuesta en frecuencias se pueden ver en [57].

El modelo de canal semicircular empleado por Reid y Nahon es similar, aunque en este caso es de orden 3, por lo que su respuesta es diferente. El modelo es el siguiente:

$$\frac{\hat{\omega}}{\omega} = \frac{\tau_l \tau_a s^2}{(\tau_l s + 1) \cdot (\tau_s s + 1) \cdot (\tau_a s + 1)} \qquad (2.3)$$

donde $\hat{\omega}$ sería la velocidad angular percibida, ω sería la velocidad angular real, y los parámetros del sistema son τ_a, τ_l y τ_s.

Al igual que con los otolitos, se supone que las respuestas de los 3 canales semicirculares son similares en las 3 direcciones principales del movimiento, por lo que sólo variarían los parámetros de un eje a otro. Esta suposición es bastante lógica desde un punto de vista fisiológico, dado que anatómicamente el mecanismo de percepción es el mismo para los 3 ejes. El modelo completo también incluye los umbrales mínimos de percepción por debajo de los cuales no se detecta la velocidad angular, pero al igual que antes éstos no son visibles en la ecuación anterior puesto que se calculan en un paso intermedio. El umbral de percepción se representa por δ_{th} y se mide en °/s. En [57] se puede ver el diagrama de bloques del modelo completo, y su respuesta en frecuencias.

La Tabla 2.1 muestra los valores empleados por Reid y Nahon para los parámetros de ambos modelos en cada uno de los ejes. Los valores de estos parámetros son difíciles de asignar, dado que cada persona percibe las cosas de modo diferente. Es más, distintos autores no coinciden ni tan siquiera en los umbrales de percepción del movimiento [43], [73], [74].

Esto, unido a que la validez del modelo es muy difícil de establecer, hace que exista controversia sobre si emplear modelos y en torno a qué modelo o modelos emplear. Se han desarrollado otros muchos modelos pero no existe un consenso general sobre cuál representa mejor la percepción. De hecho, muchos modelos se desarrollan y validan para una tarea específica, no siendo aplicables para representar la percepción de un MCA. Dado que la percepción del movimiento es un mecanismo muy sofisticado y la ciencia no conoce aún el alcance completo del procesamiento

sensorial, debemos tomar estos modelos como aproximaciones, más o menos acertadas, al comportamiento del cerebro. Como ya hemos comentado, la percepción es multimodal, y aún no se conoce con exactitud el funcionamiento de muchos de nuestros sensores, por lo que es posible que en el futuro se descubran nuevos mecanismos de percepción que aún desconocemos, y que, por tanto, no se reflejan en los modelos actuales.

		τ_a	τ_l	τ_s	K	d_{th}/δ_{th}
Otolito	x	13.2	5.33	0.66	0.4	0.17
	y	13.2	5.33	0.66	0.4	0.17
	z	13.2	5.33	0.66	0.4	0.28
Canal semicircular	x	30.0	6.1	0.1	-	3.0
	y	30.0	5.3	0.1	-	3.6
	z	30.0	10.2	0.1	-	2.6

Tabla 2.1 - *Valores de los parámetros de los modelos de otolito y canal semicircular empleados por Reid y Nahon*

2.3 – Algoritmos MCA

2.3.1 – Concepción General

Cualquier aplicación de RV que intente recrear las claves gravito-inerciales, debe disponer de dos elementos fundamentales. El primero, un dispositivo *hardware* capaz de generar movimiento. El segundo, un algoritmo que sea capaz de generar sobre ese dispositivo los movimientos necesarios para generar la sensación de movimiento que se generaría en la realidad. De los dispositivos

hardware (las plataformas de movimiento) ya hemos hablado y volveremos sobre ello en el capítulo 4. Ahora, nos centraremos en los algoritmos asociados al problema.

Aunque la casuística es rica y variada, lo que nos importa es tener una visión general de cómo se generan los comandos sobre dicha plataforma de movimiento. Generar las órdenes para la plataforma de movimiento sería muy sencillo si ésta tuviera un rango de movimiento ilimitado, ya que lo único que habría que hacer sería ordenarle ir a la posición que esté calculando el simulador virtualmente, con lo que la sensación en el usuario sería prácticamente perfecta, ya que sólo podría notar un cierto retraso si el dispositivo fuera más lento que el vehículo virtual.

Incluso aunque esta generación a escala 1:1 del movimiento fuera posible, que no lo es, tampoco sería deseable generar exactamente las mismas velocidades que el vehículo real experimenta, ya que sería peligroso para la seguridad del piloto. Hemos de tener en cuenta que uno de los motivos principales que hacen útiles los simuladores es que evitan correr riesgos mientras se practican ciertas habilidades con él.

En cualquier caso, la generación de movimiento a escala 1:1 es, en general, imposible, por lo que el problema se complica bastante. Los algoritmos que intentan solucionar este problema se conocen con varios nombres en la literatura. El nombre más coloquial es el de *washout algorithms*, vocablo de difícil traducción, normalmente traducido como algoritmos de *washout* ya que dicha palabra no tiene una traducción clara. Similarmente, también son conocidos a veces como *washout filters*. Más tarde explicaremos el

motivo de estos nombres. También son conocidos como *motion drive algorithm*s (MDA), aunque el nombre técnico más preciso es el de *motion cueing algorithms* (MCA), término que preferiremos y que podríamos traducir como algoritmos de generación de claves de movimiento, algoritmos de estimulación inercial o algoritmos de generación de claves gravito-inerciales, que sería una descripción más precisa. Dado que el término en español es muy largo, emplearemos a menudo el acrónimo inglés MCA.

2.3.1.1 – Fases del Proceso *Software* de Generación de Movimiento

El proceso *software* de generación de comandos sobre los actuadores de una plataforma de movimiento para conseguir que ésta genere en el usuario las sensaciones deseadas, se puede subdividir en varias fases (ver Figura 2.5):

· El proceso de generar (a partir de las magnitudes físicas simuladas por la aplicación de RV) las posiciones y orientaciones deseadas para la plataforma de movimiento. A esta parte es a la que se considera estrictamente como **MCA**. Aunque el resultado está evidentemente condicionado por la elección de la plataforma de movimiento, la formulación del algoritmo es relativamente independiente del dispositivo *hardware* empleado.

· Una vez elegida la posición y orientación deseada para la plataforma (en forma de 6 GdL y que comúnmente recibe el nombre de *pose*), hemos de comprobar que dicha pose es alcanzable por el dispositivo y/o que ésta no traspasa ciertos límites de seguridad pre-establecidos. Dado que, en general, el

MCA no conoce explícitamente los límites de la plataforma, hemos de comprobar si la pose requerida es alcanzable y segura, ya que, aunque el MCA se intenta ajustar para que la plataforma no alcance sus límites, en la mayoría de algoritmos esto no se hace de forma explícita, salvo contadas excepciones [75]. Si la pose no es alcanzable, deberemos establecer un criterio para escalar o eliminar parte de la petición [76]. Además, dado que muchas plataformas de movimiento son manipuladores paralelos que pueden alcanzar ciertos grados de libertad pero no simultáneamente, muchas veces deberemos establecer prioridades entre ellos cuando se produzcan conflictos. Por eso llamaremos a este subsistema **módulo de límites y de prioridades**. Este módulo es totalmente dependiente de la plataforma de movimiento que escojamos.

· Una vez conocida la pose deseada (posición y orientación) para la plataforma, el siguiente proceso necesario es el de calcular, a partir de ella, las posiciones adecuadas de los actuadores para que la plataforma alcance dicha pose. Esta parte es también dependiente de la plataforma, y se conoce con el nombre de **cinemática inversa** [77]. Dependiendo del dispositivo, puede ser ciertamente compleja de resolver.

· Dado que muchas veces la cinemática inversa del manipulador puede tener varias soluciones simultáneas (es decir, varias posiciones de actuadores que pueden generar la misma pose final de la plataforma), a veces es necesario un módulo para elegir entre las diversas soluciones. Esta elección se puede basar en muchos criterios (cercanía, velocidad, etc.). Además, es posible que ciertas posiciones de los actuadores quieran ser evitadas porque hacen que determinadas piezas se desgasten más o choquen entre

sí. Esto puede pasar en actuadores rotacionales donde determinados ángulos en determinadas condiciones sean peligrosos o no particularmente deseables. Podemos tener, por tanto, restricciones en la elección de la posición final de los actuadores. Al módulo que es capaz de resolver estos dos problemas diferentes pero relacionados le denominaremos **módulo de estrategias y restricciones**.

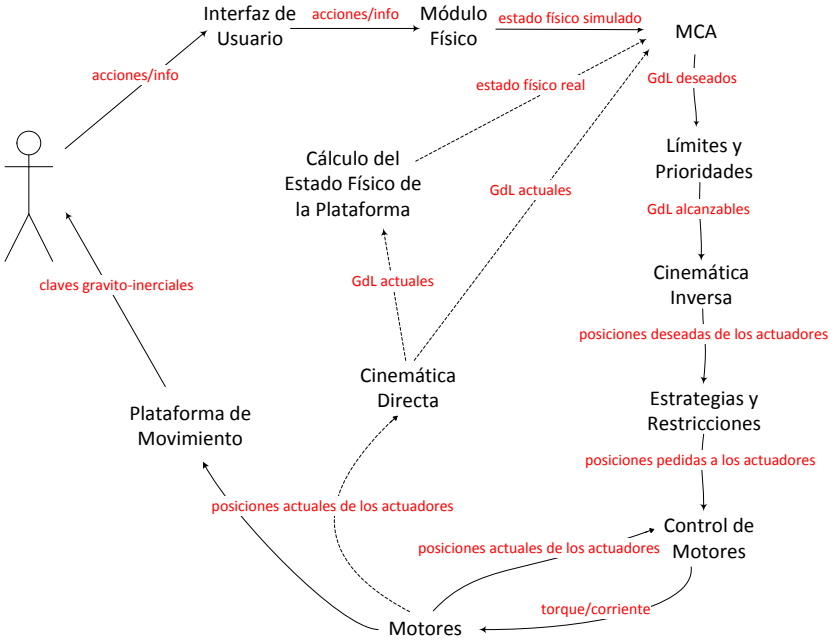

Figura 2.5 - *Esquema de la generación de claves gravito-inerciales en simuladores*

· Por último, una vez elegidas las posiciones de los actuadores, debemos generar las órdenes necesarias para alcanzarlas. Este módulo abarca, por tanto, el **control de los**

actuadores. El control de sistemas es un campo ampliamente estudiado, por lo que no entraremos en más detalles.

Opcionalmente puede haber otros módulos, como por ejemplo un módulo que calcule la **cinemática directa** del manipulador, para comprobar que efectivamente la plataforma ha alcanzado la pose deseada. La cinemática directa calcula los GdL actuales de la plataforma a partir de las posiciones reales de los actuadores.

A partir del cálculo de la cinemática directa se puede obtener el **estado físico actual de la plataforma** de movimiento, y dicha información puede enviarse al algoritmo MCA. Este proceso sólo lo realizan determinados algoritmos.

Aunque en este trabajo nos centraremos fundamentalmente en la parte de *washout* propiamente dicha, trataremos también la mayoría de los aspectos de este proceso. Del módulo físico nos ocuparemos en el capítulo 3. De la cinemática inversa, en el capítulo 4. Sin embargo, en aras de brevedad, dejaremos para futuros trabajos estudios sobre la gestión de límites, prioridades, restricciones y estrategias. Para el módulo de límites y prioridades utilizaremos la estrategia más común que es la de repartir la petición proporcionalmente entre los diferentes GdL cuando ésta no sea realizable, y no impondremos límites más allá de los límites físicos del dispositivo. En cuanto a restricciones y estrategias no asumiremos ninguna restricción y tomaremos como estrategia la de escoger siempre la solución más cercana a la actual.

2.3.1.2 – Orígenes y Fundamentos

Los primeros algoritmos MCA fueron diseñados originalmente por ingenieros de la NASA (es particularmente reseñable el trabajo de Stanley F. Schmidt y Bjorn Conrad [78], [79]), y luego fueron modificados y mejorados por diversos grupos de investigadores, aunque las ideas básicas se mantuvieron [75]. Todo algoritmo MCA tiene como entradas ciertas magnitudes físicas (del vehículo virtual simulado) calculadas por el simulador, y como salidas siempre tiene la pose deseada para la plataforma de movimiento en cada momento (ver Figura 2.6). La pose se especifica siempre en forma de 6 GdL, aunque si la plataforma tiene menos de 6 GdL, algunos no se emplearán. Las entradas del simulador, sin embargo, pueden variar de un algoritmo a otro, aunque casi siempre se corresponden con la aceleración lineal (o fuerza específica) y la velocidad angular del vehículo simulado. Estas magnitudes son las que es capaz de sensorizar el sistema vestibular humano, por lo que es lógico emplearlas como entradas. Opcionalmente se pueden utilizar otras como la posición, la orientación del vehículo, o incluso la aceleración angular o la velocidad lineal. Además, el algoritmo puede tomar como entrada características propias de la plataforma de movimiento como sus límites (aunque esto no es habitual dado que, como veremos, estos límites no suelen ser fáciles de describir).

La idea principal de cualquier algoritmo MCA es conseguir que la plataforma de movimiento no intente salirse de sus límites, (porque si lo hace, el control tendrá que pararla) intentando al mismo tiempo reproducir el movimiento que se le ordena.

Para conseguirlo, se aplican 3 ideas fundamentales (compatibles entre sí) que a continuación explicaremos: la reducción (escalado) de las magnitudes de las señales, el uso de filtros pasa-alta y la ilusión somatográvica.

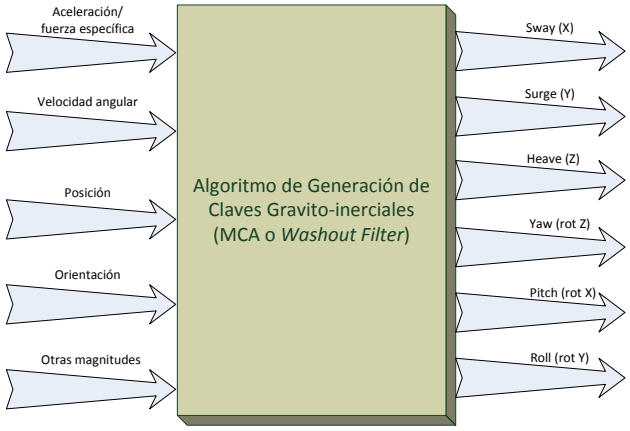

Figura 2.6 - *Entradas y salidas de un MCA*

2.3.1.3 – Escalados

La primera idea fundamental es muy sencilla. Dado que las plataformas de movimiento no siempre son capaces de generar las mismas aceleraciones que los vehículos reales, la técnica más sencilla para intentar reproducir los mismos movimientos que el vehículo virtual pero a menor escala es reducir su intensidad en un porcentaje prefijado. En algunas ocasiones, esta técnica puede ser suficiente (sobre todo en movimientos rotacionales que pueden ser pequeños), pero en general no lo es porque los movimientos de la mayoría de vehículos son ilimitados (especialmente los

traslacionales) por lo que escalarlos no es suficiente, ya que seguirían siendo ilimitados.

2.3.1.4 – Filtros Pasa-Alta

La segunda idea, más sofisticada, es la aplicación de filtros pasa-alta a las entradas del algoritmo (normalmente aceleraciones y velocidades angulares) que eliminen los movimientos de baja frecuencia que son los responsables de los desplazamientos largos y sostenidos [75]. Al eliminar las aceleraciones o velocidades de baja frecuencia, perdemos fidelidad en la reproducción del movimiento, pero nos aseguramos que la plataforma de movimiento vuelva eventualmente a su posición neutral [80]. Esto es propiamente dicho lo que se denomina *washout*, y proviene del término inglés que define el batir de las olas sobre una playa, que primero bañan la playa, se adentran en la arena y luego se retiran hacia el mar, repitiendo el proceso una y otra vez. Análogamente, la plataforma generará un movimiento simulado y cuando haya terminado, volverá a su posición neutral (*washout*) lentamente para que dicho retorno no sea percibido, permitiendo estar lista para nuevos movimientos futuros. Ésta es la clave de todo algoritmo de *washout*.

Los filtros pueden ser de muchos tipos, aunque normalmente se emplean filtros lineales de órdenes iguales o superiores a 2, ya que los de orden 1 producen deformaciones importantes en las señales en ciertos casos [80].

2.3.1.5 – La Ilusión Somatográvica

La tercera idea fundamental es el uso de la ilusión somatográvica. La ilusión somatográvica es sólo una de las muchas ilusiones vestibulares que existen [81]. Aunque el sistema vestibular presenta un diseño sorprendentemente eficaz, a veces, como cualquier sensor, presenta deficiencias y características que se deben tener en cuenta.

Para la percepción de la velocidad angular por parte de los canales semicirculares, el movimiento rotatorio debe superar cierto umbral de velocidad. De lo contrario, la deflexión de la cúpula no será suficiente como para estimular el nervio vestibular. El valor de este umbral sigue siendo motivo de discusión aunque diversos autores lo estiman entre 3 y 5 °/s [43], [73], [82]. Cualquier movimiento rotatorio por debajo de ese umbral no es detectado por el sistema vestibular. Sin embargo, una pequeña inclinación hacia adelante o hacia atrás, hace cambiar la dirección del vector gravedad con respecto a los otolitos, y por tanto, éstos sí detectarán el fenómeno, pero como una aceleración, no como una rotación (ver Figura 2.7), ya que, aunque la inclinación sea pequeña, la aceleración de la gravedad es suficientemente elevada como para ser percibida.

En el apartado a) de la Figura 2.7 podemos ver la situación que se produce cuando la cabeza experimenta una cierta aceleración lineal (en color rojo) producida por alguna fuerza externa que mueve la cabeza. La fuerza de la gravedad (en verde) siempre está presente, por lo que la fuerza gravito-inercial

resultante (en azul) tiene un cierto ángulo respecto de la vertical de la cabeza.

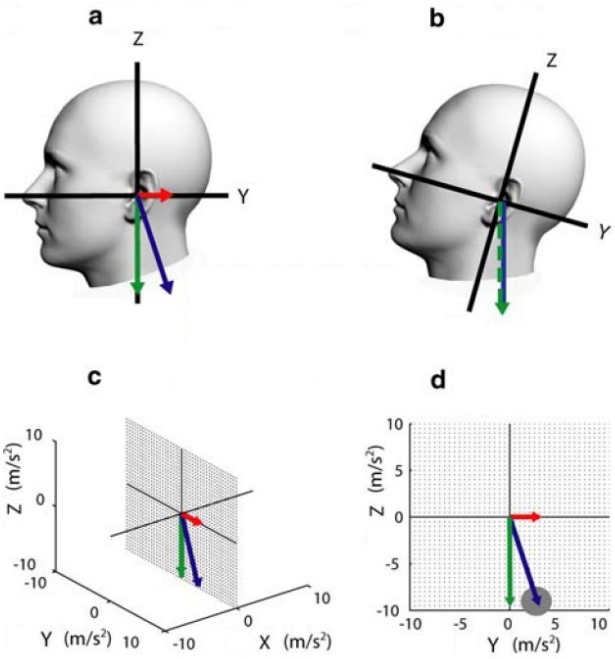

Figura 2.7 – *La ilusión somatográvica (Fuente: [58])*

En el apartado b) podemos ver la situación que se produce cuando la cabeza está inclinada un cierto ángulo hacia atrás, en reposo. En este caso no hay aceleración lineal y la fuerza gravito-inercial resultante (en azul) es igual a la gravitacional (en verde). Esta fuerza gravito-inercial es vertical, pero como la cabeza está inclinada, también forma un cierto ángulo respecto de la vertical de la cabeza. De hecho, este ángulo es el mismo que en la situación a) si el ángulo es tal que hace que la componente Y del vector gravedad sea igual que la fuerza externa del caso a). Por tanto, las

situaciones a) y b) pueden ser fácilmente confundidas por el cerebro. En los apartados c) y d) podemos ver los vectores en coordenadas cartesianas, con las componentes X, Y, Z. El disco en color gris representa el ruido asociado al vector resultante calculado.

Visto que el cerebro es, a veces, incapaz de distinguir entre pequeñas inclinaciones y aceleraciones lineales constantes, el mecanismo de los MCA para la simulación de aceleraciones sostenidas es el siguiente. Como al aplicar filtros pasa-alta a las aceleraciones de entrada, las componentes de baja frecuencia (aceleración sostenida), si las hubiera, desaparecen, lo que se hace es aplicar filtros pasa-baja a las aceleraciones frontales y laterales (X e Y en nuestra convención), y utilizar esas señales para generar sobre la plataforma una inclinación proporcional a dicha señal. Existen varias formas de implementar matemáticamente esta idea. Una de ellas sería [43]:

$$\theta = \text{atan}(\frac{F_y}{|\vec{g}|}) \qquad (2.4)$$

$$\phi = \text{atan}(\frac{F_x}{|\vec{g}|}) \qquad (2.5)$$

donde θ representa el ángulo de inclinación (*pitch*), ϕ representa el ángulo de alabeo (*roll*), $\vec{g}$ la aceleración de la gravedad y $\vec{F}$ representa la fuerza que queremos simular mediante una inclinación. Para el cálculo del *pitch* se emplea la componente frontal de la fuerza $\vec{F}$ (F_y) mientras que para el *roll* se emplea la componente lateral (F_x).

El ángulo resultante se puede escalar multiplicándolo por una constante (para exagerar el efecto), y habitualmente se suele limitar a un valor máximo (para evitar que la ilusión sea desambiguada por el cerebro), por lo que habitualmente la fórmula empleada es:

$$\theta = \text{clamp}(K_\theta \cdot \text{atan}\left(\frac{F_y}{|\vec{g}|}\right), -\theta_{max}, +\theta_{max}) \qquad (2.6)$$

$$\phi = \text{clamp}(K_\phi \cdot \text{atan}\left(\frac{F_x}{|\vec{g}|}\right), -\phi_{max}, +\phi_{max}) \qquad (2.7)$$

donde la función *clamp* devuelve el primer argumento siempre y cuando su valor esté en el rango especificado por el segundo y tercer argumentos. En caso contrario, ajustaría el resultado al valor más cercano a dicho rango. Los parámetros K_θ, K_ϕ (parámetros de amplificación) y θ_{max}, ϕ_{max} (parámetros de limitación) gobiernan el comportamiento del efecto. Hay otros autores que emplean la función arco-seno [80] en lugar de arco-tangente, pero el efecto es muy similar.

Esta estrategia tiene sus limitaciones porque si la inclinación es demasiado grande, demasiado rápida, o el centro de rotación está demasiado alejado de la cabeza del sujeto, el cuerpo puede ser capaz de detectarla como rotación [73] y el engaño desparece. Además, dado que la percepción del movimiento es multimodal, otros sensores del cuerpo, como el sentido de la vista, o el tacto pueden desambiguar la ilusión somatográvica [58]. En cualquier caso, dado que la extensión traslacional de las plataformas de movimiento suele ser muy pequeña, ésta es muchas veces la única

manera de simular aceleraciones lineales frontales y laterales sostenidas.

2.3.2 – Algoritmo Clásico

El algoritmo clásico (*classical washout*) es la referencia fundamental en la literatura de algoritmos de generación de claves gravito-inerciales, a pesar de ser bastante antiguo. Este algoritmo se basa en los trabajos iniciales de la NASA (Schmidt y Conrad) [78], posteriormente refinados y modificados por otros autores, siendo la versión más comúnmente adoptada, más referenciada y mejor explicada, la publicada por Lloyd D. Reid y Meyer A. Nahon en 1985, del Instituto de Estudios Aeroespaciales (UTIAS) de la Universidad de Toronto, Canadá. Es por ello que la versión que vamos a utilizar en este trabajo es la publicada por dichos autores en [57]. Como el lector puede imaginar, este algoritmo fue diseñado con el objetivo de ser empleado en simuladores de vuelo.

Los algoritmos clásicos se definen como algoritmos de *washout* que emplean filtros lineales y elementos de escalado, ambos con **parámetros invariantes en el tiempo**, posiblemente combinándolos con otros elementos como transformadores de sistemas de coordenadas o limitadores de señal [83].

El esquema clásico de Reid y Nahon funciona de la siguiente forma (ver Figura 2.8 - mantenemos la notación y abreviaturas en inglés para preservar la notación más usual en la literatura). Como el ser humano es sensible tanto a la fuerza específica y a la velocidad angular [53], es lógico que las entradas del algoritmo

sean la fuerza específica y la velocidad angular del vehículo simulado, mientras que las salidas son los 6 valores deseados para los 6 posibles GdL de la plataforma de movimiento (*sway*, *surge*, *heave*, *yaw*, *pitch*, *roll*). Estos 6 valores llevarán, supuestamente, a la plataforma a una pose en la que induzcan en el piloto la sensación adecuada con respecto al vehículo simulado, o al menos una que se parezca lo más posible a la representada por las entradas del algoritmo.

Las entradas del algoritmo se pueden calcular o bien en el centro del vehículo simulado si la plataforma de movimiento se va a situar respecto del piloto en una posición similar a la que el piloto estaría respecto del vehículo real, o bien respecto a la posición de la cabeza del piloto, teniendo que hacer transformaciones de sistemas de coordenadas en función de las posiciones relativas del piloto respecto del vehículo real y respecto del simulador. En cualquier caso, dado que la percepción del movimiento es multimodal y no depende sólo de la cabeza, el hecho de calcular las magnitudes respecto de la cabeza es discutible, siendo deseable que el piloto se sitúe respecto del movimiento de la plataforma en una posición similar a la que se situaría respecto del movimiento del vehículo real para que todo su cuerpo (y no sólo la cabeza) experimente sensaciones parecidas a la realidad. Esto rara vez es posible en una aeronave, dadas sus dimensiones, pero sí es posible hacerlo en algunos simuladores no aéreos como un coche o un bote de rescate. Como el algoritmo clásico del UTIAS está implementado para simuladores de avión este aspecto no pudo ser considerado, y el algoritmo toma como entradas las aceleraciones calculadas en la posición de la cabeza del piloto.

Como comentamos con anterioridad, la fuerza específica (representada aquí por la entrada F_{xyz}) es, a pesar de su nombre, una aceleración, y se mide, por tanto en m/s^2. De hecho, se define como la aceleración no-gravitacional resultante experimentada por un cuerpo. En el algoritmo clásico se mide en coordenadas locales del cuerpo, no del mundo. La velocidad angular (representada por la entrada ω_{xyz}) se mide en $°/s$ (o rd/s), y se debe calcular también en coordenadas locales del cuerpo.

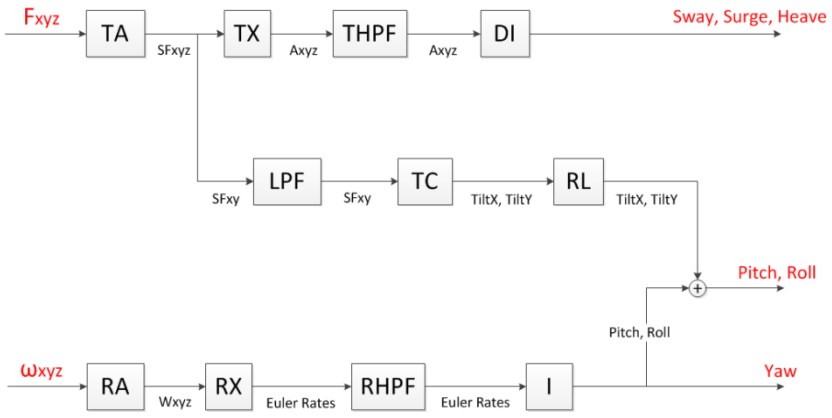

Figura 2.8 – *Esquema de funcionamiento del algoritmo clásico*

El algoritmo presenta 3 canales: un canal traslacional (parte superior de la figura), un canal rotacional (parte inferior) y un canal de coordinación (parte central).

El *canal traslacional* es responsable de generar el movimiento lineal o traslacional. Como el espacio de movimiento alcanzable por la plataforma es limitado, este canal sólo reproducirá desplazamientos de alta frecuencia. Este canal se compone de un

amplificador (TA – *Translational Amplifier*), que escala (reduce normalmente) la fuerza específica de entrada. Esta señal amplificada o reducida es transformada (TX – *Translational Transformation*) a coordenadas del mundo (añadiendo el vector gravedad y transformando las señales de coordenadas locales a coordenadas globales). Después, esta aceleración en coordenadas del mundo es filtrada (THPF – *Translational High-Pass Filter*) con un filtro pasa-alta para eliminar las componentes de baja frecuencia que harían que el dispositivo alcanzara sus límites. Finalmente, la aceleración es integrada doblemente (DI – *Double Integrator*) para obtener los desplazamientos lineales en *sway* (X), *surge* (Y) y *heave* (Z).

El *canal rotacional* es similar al traslacional, pero opera con velocidades angulares en lugar de fuerzas específicas. La idea es similar: reproducir desplazamientos angulares de alta frecuencia y evitar los de baja frecuencia. Las velocidades angulares son escaladas (RA – *Rotational Amplifier*), transformadas a tasa de cambio de ángulos de Euler (*Euler Rates*) mediante RX (*Rotational Transformation*), filtradas por un filtro pasa-alta (RHPF – *Rotational High-Pass Filter*) y finalmente integradas (I - *Integrator*) (integración simple esta vez, no doblemente como antes) para obtener los ángulos de Euler (*yaw*, *pitch*, *roll*) deseados para la plataforma. El *yaw* es directamente una salida del algoritmo, pero el *pitch* y el *roll* deben aún ser combinados con los del canal de coordinación, que explicamos seguidamente.

El *canal de coordinación* intenta simular las aceleraciones de baja frecuencia mediante ligeras y lentas inclinaciones de la plataforma de movimiento. Esta estrategia, conocida como *tilt-coordination* [73],

[80], hace uso de la ilusión somatográvica al usar la gravedad como una manera de simular aceleraciones de baja frecuencia. Como comentamos, este efecto sólo se puede conseguir para aceleraciones laterales (X) y longitudinales (Y), puesto que las aceleraciones verticales ocurren en el mismo eje que la gravedad. El proceso de coordinación empieza con el mismo escalado (TA – *Translational Amplifier*) de la fuerza específica. Después, un filtro pasa-baja (LPF – *Low-Pass Filter*) elimina las componentes de alta frecuencia que no nos interesan en este caso. Esta aceleración de baja frecuencia se convierte (TC – *Tilt Coordination*) en *pitch* y en *roll* (con las fórmulas que vimos en el punto 2.3.1.5) que después se suavizarán con un limitador de velocidad de inclinación (RL – *Rate Limiter*), para que el engaño no sea confundido con desplazamiento angular [58]. La formulación concreta de todos estos módulos de procesamiento del algoritmo se puede consultar en [57].

2.3.2.1 – Parámetros del Algoritmo Clásico

La mayoría de las unidades de procesamiento del algoritmo clásico pueden ser parametrizadas, y la elección de los parámetros y de los valores de los parámetros puede modificar significativamente el rendimiento del algoritmo. Es por ello que es importante conocer el efecto de dichos parámetros.

El amplificador traslacional (TA) tiene normalmente 3 parámetros, que representan los **factores de escala** de cada una de las direcciones cartesianas del movimiento. El valor ideal es 1, lo cual querría decir que el algoritmo trabaja exactamente con las mismas aceleraciones que el vehículo simulado. Sin embargo, esto

suele provocar que la plataforma alcance sus límites, por lo que valores menores de 1 son habituales.

El amplificador rotacional (RA) tiene análogamente 3 parámetros que representan los factores de escala de cada una de las 3 direcciones de rotación.

El filtro pasa-alta del canal traslacional (THPF) es normalmente un filtro de 3 señales con 9 parámetros. Normalmente se implementa con filtros de orden 2, y este tipo de filtros poseen 3 parámetros (**frecuencia de corte**, **ganancia** del filtro y **amortiguamiento**) [84]. Como hay 3 señales en paralelo (una por cada componente cartesiana) tenemos 9 parámetros. Aunque la ganancia y el amortiguamiento se suelen modificar también, el parámetro más importante es la frecuencia de corte. Este valor establece el límite, por encima del cual, las señales de aceleración pasarán al integrador doble (DI), que, en principio, no tiene parámetros. Por tanto, valores altos de frecuencia de corte implicarán que llegue poco movimiento al integrador, mientras que valores bajos pueden hacer que aceleraciones de demasiada baja frecuencia sean integradas, con lo que es probable que la plataforma alcance sus límites.

El funcionamiento del filtro pasa-alta rotacional (RHPF) es similar. También es normalmente un filtro de 3 señales con 9 parámetros. La única diferencia es que las señales de entrada son tasas de cambio de ángulos de Euler, en lugar de aceleraciones, pero el significado de los parámetros es análogo. Cuanto menor sea la frecuencia de corte, mayor cantidad de rotación saldrá del integrador (I), que, en principio, no tiene parámetros.

El filtro del canal de coordinación (LPF) funciona de manera similar a los otros dos filtros. Sin embargo, este es un filtro pasa-baja de 2 señales (la aceleración Z no se utiliza). El filtro actúa sobre la fuerza específica, en lugar de sobre la aceleración, pero tiene los mismos parámetros que los filtros pasa-alta. Las diferencias son que ahora hay 6 parámetros, en lugar de 9, y que la frecuencia de corte establece un límite, por debajo del cual, las señales son capaces de pasar al proceso de coordinación. Si la frecuencia de corte es demasiado baja, no habrá aceleración sostenida (de baja frecuencia) que sea simulada por la plataforma de movimiento.

El proceso de *tilt-coordination* funciona en dos etapas. La primera etapa (TC) se puede parametrizar, al establecer la cantidad de inclinación que 1 m/s^2 de fuerza específica producirá. Idealmente, 1 m/s^2 de fuerza específica debe provocar un ángulo tal que la componente no vertical de la gravedad induzca exactamente 1 m/s^2 de fuerza específica en el piloto. Sin embargo, esto implicaría que para obtener 1 g de fuerza sostenida podríamos necesitar una inclinación de hasta 90º (dependiendo de la implementación). Como esto no es razonable, y tal inclinación sería *descubierta* por el usuario, un **escalado de inclinación** y un **límite de inclinación** deben ser definidos. El escalado de inclinación funcionará como un amplificador/reductor de entrada, mientras que el limitador de inclinación, funcionará como un limitador de salida (*output clamp*). Estos parámetros se corresponden con K_θ, K_ϕ, θ_{max} y ϕ_{max} en la fórmula del apartado 2.3.1.5. Si el escalado es muy alto, una pequeña fuerza específica generará mucha inclinación, y viceversa. Si el límite de inclinación es demasiado bajo, la fuerza inducida será

significativamente menor que la real, pero si es demasiado alto, la cantidad de inclinación puede ser tan elevada que el usuario note el engaño [43], y el canal rotacional no tenga espacio para simular las rotaciones puras reales.

La segunda etapa del proceso de *tilt-coordination* es un limitador de velocidad angular (RL). Este proceso asegura que la plataforma de movimiento realiza el proceso de coordinación a velocidades angulares pequeñas. Esto se hace porque la inclinación debe hacerse a bajas velocidades para no confundirla con un movimiento rotacional, y sea interpretada como aceleración lineal sostenida [43]. Este proceso sólo tiene habitualmente un parámetro, el **límite de velocidad angular** (*angular rate limit*), que se expresa en °/s y establece un límite de velocidad para los ángulos de inclinación calculados por el módulo TC. Límites altos reducirán el retraso que se produce inevitablemente al aplicar esta estrategia (retraso entre la aceleración de entrada y la simulada), pero si el valor es muy alto, la estrategia puede dejar de funcionar porque el movimiento de inclinación se notará y será confundido con rotación. Por el contrario, si el límite es demasiado bajo, se producirá un gran retraso en el proceso y la aceleración generada por la inclinación no será percibida como consecuencia de la aceleración del vehículo.

Para una explicación más detallada del algoritmo y de sus parámetros, el lector puede consultar [57], [85] y [86].

Es importante aclarar que Conrad y Schmidt [78] desarrollaron dos de los algoritmos clásicos originales: el coordinado y el no-coordinado. Este nombre puede inducir a

confusión con lo que hemos visto. Coordinación se refiere al uso de la aceleración lineal (traslacional) del simulador para compensar fuerzas resultantes erróneas inducidas (debido a que las componentes del vector gravedad generan fuerzas resultantes que se pueden confundir con aceleraciones lineales cuando el simulador gira para generar aceleraciones angulares). A esto se le llama coordinación traslacional para no confundir con el efecto de *tilt-coordination*, que ambos algoritmos usan. La nomenclatura histórica es confusa dado que los propios Conrad y Schmidt se refieren al algoritmo no-coordinado como "algoritmo clásico", y al coordinado como "algoritmo coordinado" [83], aunque lo habitual es clasificarlos a ambos como clásicos. La implementación de Reid y Nahon no emplea coordinación traslacional.

2.3.3 – Algoritmos Adaptativos

El algoritmo adaptativo es una modificación del algoritmo clásico. Se consideran algoritmos adaptativos a aquellos cuyos parámetros de filtrado, escalado (y posiblemente otros) varían con el tiempo (durante el tiempo de ejecución del algoritmo). En el algoritmo clásico, dichos parámetros se pueden modificar, pero su valor no es función del tiempo (eran constantes para una misma ejecución). El algoritmo adaptativo fue propuesto inicialmente por Parrish en 1973 [87].

Aunque el algoritmo adaptativo recibió buena acogida en sus inicios, (incluso es considerado por algunos autores el mejor algoritmo), investigaciones posteriores sugieren que la elección de parámetros y las maniobras de prueba tienen mucha más

importancia que el algoritmo en sí en la evaluación de los mismos [86].

De hecho, aunque el nombre del algoritmo induce a pensar que adapta sus parámetros a la situación de la plataforma (y que por lo tanto no tiene parámetros), en realidad, hay que ajustar los parámetros que gobiernan la adaptabilidad del algoritmo, por lo que, en el fondo, sigue requiriendo parámetros y no es un avance tan grande con respecto al algoritmo clásico en este sentido.

La adaptabilidad del algoritmo puede hacerse de muchas maneras pero lo más común es modificar las ganancias de los filtros en tiempo real de manera que se minimice una función de coste mediante técnicas de descenso de gradiente (*steepest descent*) [88]. La función de coste suele contener información sobre las excursiones de la plataforma respecto de su posición neutral, e información sobre el movimiento generado. Por ello, cuando las entradas del algoritmo son grandes, la ganancia suele reducirse más que con entradas pequeñas, ya que entradas grandes suelen producir movimientos largos que generan valores penalizables en la función de coste. Se supone que esto hace que se reduzca la generación de falsas claves de movimiento y se aprovechen mejor las capacidades de la plataforma, aunque el hecho de penalizar el movimiento por alejarse de la posición neutral también hace que se reduzca al mismo tiempo la generación de claves de movimiento correctas.

También es posible hacer adaptativa la frecuencia de corte e incluso el amortiguamiento de los filtros, pero ello puede producir diferencias significativas en los retrasos de la generación de

movimiento que pueden confundir al piloto, y tampoco es sencillo establecer una relación directa entre los límites de la plataforma y los frecuencias de corte [88].

Dado que se pueden seleccionar múltiples combinaciones de parámetros para "adaptar" y se pueden utilizar diferentes funciones de coste, no podemos hablar del algoritmo adaptativo como de un tipo único de algoritmo, sino más bien como de un conjunto de ellos o incluso como una característica particular de algunos algoritmos, dado que en realidad su estructura es básicamente la misma que la del algoritmo clásico.

2.3.4 – Algoritmos Óptimos

Los algoritmos óptimos funcionan generalmente como los algoritmos clásicos. Es decir, usan filtros y amplificadores invariantes en el tiempo. La diferencia estriba en el método utilizado para seleccionar los coeficientes de los filtros [83]. Los coeficientes en el algoritmo óptimo son seleccionados utilizando una función de coste *offline* que tiene en cuenta cómo es la percepción de las señales por parte del piloto.

En el fondo, el algoritmo óptimo trata el problema como un problema de minimización, tratando que las aceleraciones percibidas en el simulador se parezcan a las percibidas en el vehículo real, intentando minimizar el error *e* entre ambas señales, y manteniendo al mismo tiempo las restricciones propias de la plataforma de movimiento. Es por ello que el algoritmo necesita modelos de percepción para poder comparar lo percibido en un

caso y otro (virtual y real). Las diferentes implementaciones del algoritmo óptimo utilizan diferentes modelos de respuesta vestibular. El método fue desarrollado por Sivan [89].

En principio, el ajuste de parámetros se facilita porque los parámetros de los filtros no son controlados directamente, y lo que se ajusta son parámetros que controlan la relación entre fidelidad de movimiento y extensión de los mismos. Lo malo es que, dado que el ajuste de los parámetros de los filtros no se realiza directamente, no se tiene el mismo control sobre el comportamiento del algoritmo, y en cualquier caso no nos libramos de ajustar parámetros, aunque pueda ser más sencillo. El algoritmo se considera más fácil de ajustar para no expertos, pero menos versátil en manos de un experto.

Una crítica que se hace habitualmente a estos algoritmos es que dependen de un modelo vestibular que en muchas ocasiones tiene una validez discutible.

Otra crítica habitual es que el nombre óptimo sugiere que no se puede encontrar otro mejor, cosa que evidentemente no es cierta, dado que lo único que optimiza es una cierta función que se corresponde con un cierto modelo de percepción. De hecho, este algoritmo no ha obtenido una aceptación superior a la de los otros dos. Además, dado que se trata de un problema subjetivo, la palabra óptimo puede resultar demasiado ambiciosa.

2.3.5 – Otros Algoritmos

Aunque los 3 tipos de algoritmos mencionados son los más comunes, existen otras muchas formulaciones y propuestas. Telban y Cardullo proponen un algoritmo no lineal que combina ventajas de los algoritmos óptimos y adaptativos [90]. Dagdelen propone un algoritmo basado en el control predictivo (Model Predictive Control – MPC) [91]. El algoritmo intenta minimizar el error perceptual teniendo en cuenta explícitamente los límites concretos de la plataforma, eliminando la necesidad de ajustar el algoritmo para el peor escenario. Liao propone otro algoritmo que toma como entrada, además de las entradas habituales, información de la cinemática inversa del manipulador y de los límites de la plataforma calculados en tiempo real [92].

Existen otras aproximaciones, como diseños integrados del simulador y el algoritmo [93] [94], utilización de redes neuronales para la resolución del problema [95], reformulaciones de los algoritmos más comunes (clásico, adaptativo, óptimo) [96], y otros numerosos estudios como [97], [98], [99], [100], [101] y [102], pero el problema fundamental sigue siendo que es muy complicado comparar las diversas soluciones y sacar conclusiones firmes sobre dichas comparaciones.

2.3.6 –Sistemas de Evaluación

Dado que por razones tecnológicas, económicas y prácticas no es posible, ni siempre deseable, realizar una generación de movimiento a escala 1:1 (es decir, igual que el original), se define la **fidelidad gravito-inercial** del simulador como la capacidad de un

simulador para generar las sensaciones de movimiento que se perciben en una situación real con un vehículo real. La evaluación gravito-inercial es, por tanto, un método para obtener un valor de fidelidad para el simulador.

Es evidente que la fidelidad del movimiento de un simulador depende tanto del *hardware* (la plataforma de movimiento) como del algoritmo empleado y de cómo éste esté ajustado, por lo que cuando evaluamos un algoritmo es importante tener en cuenta que realidad estamos evaluando el conjunto *hardware-software* (algoritmo MCA, plataforma, módulo físico, control de los actuadores, etc.).

La evaluación de este tipo de algoritmos está íntima y mutuamente relacionada con el ajuste de parámetros, ya que los parámetros se deben ajustar en base a un criterio de evaluación, y a su vez, el resultado de la evaluación depende en gran medida del ajuste previo realizado.

Aunque no existen demasiadas referencias en la literatura sobre la evaluación de este tipo de algoritmos, se han publicado algunos trabajos interesantes con diferentes enfoques. Diversos autores han propuesto diferentes tests para evaluar algoritmos MCA, aunque lo habitual es que el criterio de evaluación sea simplemente la opinión de uno o varios pilotos, con lo que la evaluación rara vez se hace de manera formal.

Parrish [87] realizó comparaciones objetivas y subjetivas de dos algoritmos MCA en maniobras de despegue y aterrizaje de aeronaves. No se encontraron pruebas de diferencia en el

rendimiento de los pilotos para las diversas condiciones probadas (sin movimiento, *washout* lineal y *washout* no lineal).

En 1977, Sinacori [103] estudió los requisitos de un simulador de helicóptero y fue precisamente él el que propuso el concepto de fidelidad de reproducción de movimiento (*motion cueing fidelity*). El concepto correlaciona opiniones de los pilotos con la atenuación (escala de los movimientos) y la distorsión en fase (retraso) de los movimientos reproducidos por la plataforma.

Este concepto fue empleado por Mikula para el desarrollo de un criterio de fidelidad para tareas en las que se necesita el canal lateral (*roll* y traslación lateral) [104] y por Schroeder para optimizar la generación de movimiento de un simulador de helicóptero [105].

Al mismo tiempo, se han venido realizando esfuerzos para simplificar el ajuste de parámetros, lo cual simplifica la posterior evaluación. Es notable la contribución de Grant al desarrollar el sistema experto PROTEST [106], que ayuda a reducir el esfuerzo necesario para completar el proceso de seleccionar los valores más adecuados para los algoritmos MCA. PROTEST no es un sistema de evaluación, es un sistema de ajuste de parámetros que permite realizar dicho ajuste sin necesidad de un experto, ya que el sistema sustituye al experto. Aunque los experimentos se realizaron con el algoritmo clásico, los resultados pueden ser extrapolados. PROTEST demostró alcanzar el nivel de rendimiento de un experto en ajuste de parámetros [107]. Este procedimiento continúa los trabajos de Reid y Nahon en el UTIAS, al mejorar los procedimientos existentes hasta aquél momento [85]. El trabajo de Grant va más allá del desarrollo de un sistema experto, ya que

dedica esfuerzo a exponer y analizar una metodología para el ajuste subjetivo de los valores de los parámetros de los algoritmos de *washout* [83], [108].

Bruenger [109], [110] se basó en las reglas de Grant y Reid para realizar la validación de un simulador de automoción. La validación consistía en pruebas objetivas del rendimiento comparado (real vs virtual) de conductores y un cuestionario sobre las impresiones subjetivas de los mismos.

Desde un punto de vista taxonómico, la evaluación de este tipo de algoritmos se puede hacer básicamente de dos maneras: **objetiva** (cuantitativa) o **subjetivamente** (cualitativamente). La evaluación objetiva puede, además, ser **directa** o **indirecta**.

En la **evaluación objetiva indirecta** no se evalúan las salidas del algoritmo, sino que se evalúa cómo los parámetros del algoritmo influyen en el desarrollo de tareas ejecutadas sobre el simulador. Un ejemplo de este tipo de evaluación sería, para un simulador de coches de competición, medir tiempos por vuelta para diferentes versiones de uno o varios algoritmos de *washout*. Escogeríamos, por ejemplo, la versión del algoritmo que demostrara permitir los mejores tiempos por vuelta.

Cuando la evaluación se hace en base al rendimiento o las habilidades de los pilotos, lo que hacemos es suponer que los parámetros del algoritmo MCA que hacen que el piloto pueda conducir/pilotar mejor (midiendo datos objetivos sobre la tarea como tiempos, colisiones, seguimiento de trayectorias, salidas de carril, etc.) son los que hacen que éste se sienta más cómodo con el

simulador porque inducen una mejor percepción. Por tanto, también le podríamos llamar **evaluación inducida o comportamental**. Esto es evidentemente una suposición, ya que es posible que el piloto se sienta más cómodo sin movimiento que con él, porque, por la razón que sea, éste le distraiga de la tarea objetivo. Además, este método a veces supone que distintos pilotos tienen la misma destreza. Incluso aunque sólo se evalúe a un único individuo estaríamos suponiendo que su destreza depende principalmente de la generación gravito-inercial, cosa que no es necesariamente cierta, porque otros factores, como el número de horas sobre el simulador, afectan de modo significativo sobre la destreza. La evaluación objetiva indirecta no mide realmente la fidelidad de los usuarios de la simulación sino su habilidad o rendimiento ante una determinada tarea realizada en el simulador. La asunción de que un mayor rendimiento implica una mayor fidelidad en la simulación y que por tanto la fidelidad y el rendimiento de los usuarios están correlacionados, es algo que se debe demostrar para cada caso particular, y que muchos autores asumen como cierto de manera global y errónea. Ejemplos de este tipo de evaluación los podemos encontrar en [67], [110], [111] y [105].

En la **evaluación objetiva directa** lo que se evalúa son ciertas magnitudes físicas que se corresponden con la ejecución del algoritmo. Es decir, se evalúa cómo son las salidas del algoritmo con respecto a las que deberían ser (idealmente). Una manera de realizar esta tarea es presentar gráficas de la telemetría del vehículo u otro tipo de comparaciones entre las magnitudes generadas por el simulador y el vehículo real. Ejemplos de esto los podemos ver en [112] y [113].

Otra manera de realizar la evaluación de forma cuantitativa (directa) es someter al conjunto plataforma-algoritmo a una serie de pruebas con ciertas entradas o maniobras prefijadas, y después medir retrasos, respuestas en frecuencias, tiempos de respuesta, ganancias, etc. Este sistema de evaluación está más desarrollado en aeronáutica donde se definen una serie de criterios para caracterizar la fidelidad del movimiento de una plataforma en base a maniobras de despegue, aterrizaje, vuelo, etc. Un ejemplo de ello es lo que se denomina *Objective Motion Cueing Test* (OMCT) [114], que establece unos valores mínimos para ciertas magnitudes medidas en las pruebas establecidas.

Otra manera de realizar una evaluación objetiva consiste en utilizar algún modelo humano de percepción y medir sobre este modelo cómo es la percepción del algoritmo [115], [116]. A esto se le denomina **evaluación perceptual** [117]. Esto hace que sea una evaluación objetiva cuya validez depende en gran medida de la representatividad que les otorguemos a dichos modelos sobre la percepción real. Quizás el ejemplo más representativo de esta aproximación es la *Motion Perception Toolbox* (MPT) [70] desarrollada por el instituto TNO para el conocido simulador *Desdemona* [112]. MPT es una *toolbox* de MATLAB que permite analizar de forma objetiva los movimientos generados por la plataforma de movimiento de un simulador. Para ello, analiza las salidas reales de la plataforma haciéndolas pasar por diversos modelos de percepción que intentan representar varios de los aspectos que contribuyen a la percepción. De hecho, también analiza la percepción visual, además de la inercial.

La evaluación objetiva tiene algunas ventajas. Evita la ambigüedad de la comparación subjetiva; permite comparaciones justas de distintos algoritmos (especialmente con simuladores del mismo tipo) y permite evaluar de forma clara si la introducción de mejoras en un simulador es efectiva.

Sin embargo, también presenta desventajas. El problema es realmente subjetivo por lo que establecer un criterio objetivo sin base subjetiva es discutible y si no se aplica correctamente puede dar lugar a resultados contradictorios con respecto a la evaluación subjetiva; puede hacer que los simuladores se construyan con el único objetivo de obtener una buena calificación en estos tests, obviando el rendimiento global del simulador.

El otro tipo de evaluación dominante es la evaluación subjetiva. En este tipo de evaluación lo que se hace es someter a los pilotos a diversos algoritmos MCA o diversas variaciones de los mismos y preguntar a los pilotos sobre sus sensaciones sobre el simulador. Existen múltiples ejemplos de evaluación subjetiva [118], [119], [120], ya que ha venido siendo la manera tradicional de evaluar este tipo de problemas, aunque hay pocos estudios formales sobre ello, a excepción hecha de [83].

La evaluación subjetiva tiene varias ventajas. Al ser un problema subjetivo, una evaluación subjetiva es la forma natural de evaluarlo; es más sencilla de realizar a nivel técnico, ya que sólo requiere hacer preguntas, aunque las preguntas hay que elegirlas con cierto cuidado, para que las respuestas sean significativas.

Aunque también presenta desventajas importantes. Existen variaciones significativas entre los diferentes individuos, lo cual hace que los resultados de la evaluación puedan ser muy distintos dependiendo de los usuarios que elijamos; dada la naturaleza de las personas, la evaluación puede no depender sólo de la persona o personas elegidas sino también del momento en que se haga la prueba, del orden en que se hagan las pruebas, o de cómo sean éstas dirigidas; la evaluación subjetiva puede depender de las expectativas y preferencias personales de cada persona.

Algunos autores niegan la validez de la evaluación objetiva [83] mientras que otros la defienden [67], por lo que es común encontrar autores que combinan diferentes tipos de evaluaciones para probar la validez de sus simuladores [121], [122], [113], [109], [105].

2.3.7 –Ajuste de Parámetros

El ajuste de parámetros en los algoritmos MCA ha venido haciéndose tradicionalmente mediante pruebas sucesivas con pilotos y/o expertos en el tipo de vehículo simulado. El proceso es tedioso, ya que normalmente las opiniones de los pilotos son difíciles de traducir en cambios de parámetros. En [85] se puede consultar un ejemplo de este tipo de aproximaciones, aunque en dicho trabajo también se emplea la evaluación perceptual. Sólo para el ajuste de parámetros, Reid y Nahon dedican un volumen completo de su trabajo, lo cual da idea de la magnitud del problema.

Si bien es cierto que hay algunos algoritmos, como el algoritmo adaptativo, que intentan ajustar automáticamente su salida al movimiento simulado, ello no quiere decir que no tengan parámetros y que no haya que buscar valores adecuados para ellos. Aunque estos algoritmos modifican en tiempo real ciertos parámetros (por ejemplo la ganancia de los filtros), existen otros parámetros en el algoritmo (deberíamos llamarlos metaparámetros) que afectan a la forma en cómo se modifican los parámetros originales del algoritmo (ahora no directamente modificables). El resultado es, que al final, todos los algoritmos de este tipo tienen ciertos parámetros (la mayoría, decenas de ellos) que controlan cuál es la cantidad y tipo de información que desechamos. Dado que una plataforma de movimiento rara vez va a poder simular el movimiento real, el usuario debe decidir qué parte del movimiento elimina y qué parte intenta reproducir, cosa que de una forma o de otra acaba traduciéndose en parámetros.

Existen diversas aproximaciones para el ajuste de parámetros en algoritmos MCA. Por desgracia, la literatura al respecto es aún más escasa de lo que es respecto a la evaluación de los mismos, siendo [108], [106], [123] y [109] las más relevantes. De hecho, en algunas ocasiones se realizan evaluaciones de sistemas de generación de movimiento sin detallar ni explicar cómo se ha realizado el ajuste del MCA, como en [124]. En cualquier caso, podríamos clasificar estas aproximaciones al ajuste de parámetros en 3 grupos.

La primera es la aproximación *offline*. Las técnicas *offline* se definen como aquellas aproximaciones que no requieren la presencia de pilotos para ajustar los coeficientes. El ajuste se puede

hacer de forma automática mediante un computador [13], [125] o mediante un experto en el algoritmo. En este último caso, que es bastante excepcional, el ajuste se realizaría en base a su conocimiento y a suposiciones sobre el vehículo, el simulador y la plataforma.

La segunda aproximación se basa en pruebas experimentales. El rendimiento de los pilotos, opiniones de expertos o las evaluaciones o sensaciones reportadas por los pilotos, son utilizados para generar un pequeño número de posibles conjuntos de valores para los parámetros del algoritmo. Posteriormente, se prueba (con pilotos) el algoritmo en diferentes maniobras, con cada uno de esos conjuntos de parámetros [110]. El conjunto que proporcione una mejor sensación (o rendimiento) media para los pilotos seleccionados, será el elegido.

El tercer método se suele llamar en inglés *pilot-in-the-loop*. Esta técnica requiere un piloto y un experto en el algoritmo. Las impresiones del piloto son pasadas al experto, que va realizando cambios en el algoritmo hasta que el experto considera que se obtiene el mejor conjunto de parámetros a juicio del piloto [126]. El experto puede ser sustituido por un sistema experto construido a partir del conocimiento del mismo, como hace Grant en [106] con su sistema PROTEST (que ya comentamos en el apartado anterior), pero la idea es la misma. Aunque es el método más utilizado, existen pocas referencias al mismo y pocas explicaciones sobre cómo realiza cada autor este proceso.

Todas las técnicas presentan problemas. El ajuste *offline* subjetivo requiere de un experto con un conocimiento demasiado

elevado, ya que es muy difícil que una persona tenga un control tan preciso a priori sobre la sensación que puede provocar cada cambio de cada parámetro en un sistema tan complejo como éste. El ajuste *offline* automatizado sería lo más deseable, pero requiere una validación subjetiva previa y no es nada sencillo. Los resultados que puede ofrecer pueden no corresponderse luego con los gustos subjetivos de los pilotos.

El ajuste basado en pruebas experimentales requiere que los conjuntos de valores que se propongan sean lo suficientemente buenos y que el número de pruebas sea lo suficientemente representativo del funcionamiento del simulador, cosa que puede, o no, ser cierta para cada caso.

Por último, el ajuste prueba-error con pilotos presenta un gran número de problemas. El primero es que, muchas veces, los pilotos sólo son capaces de recordar unos pocos detalles de las últimas pruebas realizadas, por lo que después de varias decenas de conjuntos de valores, su capacidad para decidir si la prueba es la mejor hasta el momento, se ve algo reducida. Muchas veces, un conjunto de parámetros puede parecer bueno simplemente por comparación con la anterior prueba, que ha podido ser muy mala. El segundo problema es la falta de un criterio para decidir cuándo terminar el proceso. ¿Cuándo sabemos que el ajuste es suficientemente bueno? Normalmente, por desgracia, este criterio suele ser cuando el piloto y el experto están suficientemente cansados como para no ser capaces de distinguir ya ningún camino de mejora. Lo cual nos lleva el tercer problema. Aunque es lógico que la evaluación se base en la percepción subjetiva de los pilotos, no es lógico que le añadamos al proceso la subjetividad de las

decisiones del experto, que puede decidir ir por un camino o por otro sin un criterio que no sea el de su intuición. Además, el experto, debe tener en la cabeza un conocimiento demasiado alto sobre el efecto de numerosos parámetros sobre un sistema complejo. Otro problema no menor es el tiempo que deben dedicarle el piloto y el experto al proceso. En ocasiones hacen falta días de pruebas para conseguir un resultado aceptable. El último problema es, evidentemente, que el criterio del piloto es subjetivo y por tanto, puede ser muy diferente del de otros compañeros, por lo que un cambio de piloto puede implicar tener que volver a repetir todo el proceso. Esto es, al mismo tiempo, un problema y una característica intrínseca de estos algoritmos.

2.4 – Situación Actual

Actualmente, el algoritmo clásico (con pequeñas variantes) sigue siendo el más utilizado [109], [127], [126], [80], [124], [106], [86], [128], incluso en aplicaciones comerciales [33], y a pesar de que, de tanto en tanto, surge algún algoritmo nuevo cuyo autor proclama como el mejor, lo cierto es que ninguno ha conseguido desplazar al algoritmo clásico como solución de referencia, aunque el algoritmo adaptativo también goza de bastante aceptación [128], [126], [88]. Esto refleja varias cosas. La primera es que dado que es difícil comparar dos algoritmos, cualquiera puede decir que el suyo es el mejor, por lo que es difícil sacar conclusiones firmes [75]. La segunda es que siguen sin estar claros los límites de esta tecnología. La tercera, es que, en el fondo, todos los algoritmos hacen cosas similares, y, como se expresa en [86], la diferencia entre un

algoritmo y otro a veces radica en cómo de bien ajustado esté para la tarea encomendada.

En cuanto a la evaluación y al ajuste de parámetros, no hay un consenso claro. La evaluación predominante es la subjetiva, aunque en los últimos años han aparecido más trabajos que utilizan aproximaciones objetivas [67], [129].

El *hardware* (las plataformas) también ha evolucionado bastante desde los primeros diseños de Gough, Kappel, Stewart, etc. en los años 50 y 60, aunque más en los actuadores que en los diseños. Hoy en día, prácticamente todas las empresas de automoción y aviación disponen de plataformas de movimiento para realizar pruebas en simuladores. La mayoría de los dispositivos actuales son manipuladores paralelos en forma de plataforma Stewart con 6 GdL. El diseño de tipo Stewart [37] ha triunfado claramente y son las más utilizadas porque su *payload* (capacidad de carga) es bastante elevada (son capaces de generar bastante fuerza con motores relativamente pequeños) y pueden provocar grandes aceleraciones. El problema es que tienen un reducido rango de movimiento y una baja destreza (*dextirity*) o habilidad para cambiar de posición-orientación de forma rápida.

Es por ello que es relativamente habitual montarlas sobre uno o varios carriles traslacionales para que el rango de movimiento en alguno de los ejes sea mucho más elevado (normalmente se elige el eje Y porque es el que coincide con la dirección de movimiento de los vehículos). Ejemplos de manipuladores paralelos de este tipo son el simulador de Renault

(*ULTIMATE*) [91], el de Peugeot-Citroën (*SHERPA*) [127] o el *VTI Driving Simulator IV* [130].

También existe alguna solución en forma de manipulador serie como el *MPI Motion Simulator* del Instituto Max Planck [131], pero es menos común, porque es más compleja y habitualmente mucho más cara. Sin embargo, su rango de movimientos es mucho mayor, por lo que no es una mala opción.

En cuanto a los simuladores, la mayoría de simuladores de bajo coste no solían incluir, hasta hace poco, generación de movimiento, dado que la construcción de una plataforma de movimiento es una tarea compleja, y generalmente no se ofrecían a precios competitivos. Por fortuna, esto está cambiando actualmente, dado que empresas como *CKAS* [33] o *Inmotion* [132] ofrecen productos de simulación de movimiento a precios competitivos. Muchos de estos productos incorporan algoritmos MCA de los que ofrecen poca o ninguna información, aunque algunos de ellos (como *CKAS*) emplean el algoritmo clásico con modificaciones (que obviamente no detallan). Aun así, la mayoría de pequeños simuladores no suelen incluir generación de movimiento, cosa que sí suelen hacer las grandes empresas con sus simuladores. Las grandes corporaciones o equipos de competición punteros suelen desarrollar sus propios diseños de plataforma de movimiento, aunque a veces los encargan a empresas de simulación, como *Moog* [133] o *E2M* [134]. Este tipo de simuladores cuestan, en el mejor de los casos, decenas de miles de euros, y por tanto sólo están normalmente al alcance de grandes empresas.

Capítulo 3

Caracterización Gravito-inercial de un Simulador

Dado que cualquier simulación es un intento de reproducir un fenómeno real, el primer paso a la hora de diseñar el módulo inercial de un simulador de un vehículo es conocer cómo es ese fenómeno real y definir hasta qué punto queremos que el simulador reproduzca dicho fenómeno. Aunque esto parece obvio, en opinión del autor, uno de los mayores errores que se cometen en la recreación de claves gravito-inerciales en simuladores consiste en no saber elegir el tipo y rango de movimientos más adecuados para la plataforma de movimiento del simulador. Muchas veces, por desconocimiento, se piensa que la mejor opción es elegir el dispositivo más caro o con más grados de libertad, cuando no siempre ello implica mejor sensación si no se tienen en cuenta otras consideraciones.

Puesto que el simulador universal perfecto no existe y las plataformas de movimiento siempre son de movimiento limitado, es muy importante conocer qué rasgos caracterizan de forma más acusada el movimiento del vehículo que queremos simular, para poder dedicar mayor esfuerzo a simular esos rasgos. Conocer el rango y naturaleza de los movimientos que se pretenden simular es importante, dado que permite orientar los recursos económicos y humanos en la dirección adecuada. Por ejemplo, si quisiéramos simular un elevador de desplazamiento vertical, parece claro que el

movimiento en el eje Z sería prácticamente el único importante, y no necesitaríamos una plataforma de 6 GdL sino un dispositivo con 1 GdL vertical muy extenso y/o potente. Aunque este es un ejemplo algo simplista, ideas similares se pueden aplicar a la simulación de otros dispositivos y vehículos.

En este capítulo, trataremos primero de describir cuál es la manera más adecuada de definir las necesidades gravito-inerciales de un simulador de vehículos. Posteriormente, describiremos cómo podemos diseñar un módulo físico para alimentar las entradas del módulo inercial.

3.1 – Caracterización de la Dinámica de un Vehículo

Tanto si tenemos disponible un modelo físico virtual de vehículo (sistema físico) y "solamente" necesitamos diseñar y programar la plataforma de movimiento (sistema inercial), como si hemos de realizar ambas tareas, el primer paso lógico para poder realizar la simulación del movimiento de un vehículo, sea cual sea éste, es conocer y entender su movimiento. Esta información nos será útil, por tanto, para poder modelizar el movimiento del vehículo pero sobre todo para ser capaces de identificar qué tipo de dispositivo generador de movimiento es el más adecuado para el tipo de simulación que pretendemos realizar.

Aunque las mediciones necesarias pueden depender en gran medida del tipo de vehículo, existen elementos comunes que se

pueden generalizar a la hora de caracterizar el movimiento de un vehículo con el objetivo de ser simulado.

Dado que no hemos encontrado ningún trabajo que proponga una caracterización del movimiento de un vehículo marino de pequeñas dimensiones con el propósito de modelar este movimiento y reproducirlo posteriormente con un generador de movimiento, consideramos adecuado ejemplificar esta caracterización, paso a paso, con un ejemplo real de simulador realizado por el autor en el IRTIC: un simulador de bote de rescate rápido.

Hemos encontrado referencias similares en la literatura sobre otro tipo de vehículos (principalmente aéreos y terrestres), y aunque el comportamiento de este tipo de vehículos puede ser sustancialmente diferente al de un vehículo marino, el procedimiento de análisis puede ser extrapolado. Sin embargo, la mayoría de estudios se centran en analizar el movimiento con propósitos diferentes al nuestro. Nosotros queremos analizar el movimiento con el objeto de reproducirlo, mientras otros trabajos se centran habitualmente en el análisis del movimiento orientado a la optimización del diseño real de los vehículos [135], [136].

Uno de los trabajos que aborda el problema de la caracterización de vehículos terrestres con un objetivo similar al nuestro es el de G. Reymond y A. Kemeny [80]. En su trabajo se detallan los pasos a realizar para analizar el movimiento de, en su caso, un turismo, donde establecen que para construir un simulador realista deben considerarse 4 aspectos fundamentales: validez física, validez perceptual, validez comportamental relativa y

validez comportamental absoluta. Los dos primeros aspectos son los que nos interesan en este estudio. La validez física se define como la comparación entre los movimientos simulados y los reales. La validez perceptual, se define como la comparación de la percepción multisensorial del movimiento simulado con respecto a una situación real.

Para llevar a cabo esas comparaciones, Reymond y Kemeny caracterizan un Renault Laguna, mediante el uso de acelerómetros, conduciendo el vehículo durante varias vueltas a distintas velocidades en un circuito, para reproducir una situación típica de conducción. Los datos adquiridos son analizados en términos de la plataforma de movimiento que se pretende utilizar para el simulador. Nuestro análisis se basará en este procedimiento, pero ejemplificándolo sobre un bote de rescate, ampliando el estudio y añadiendo algún detalle nuevo, como por ejemplo, el uso de otros dispositivos como anemómetros y brújulas digitales.

3.1.1 – Sistema de Referencia

Para estudiar un fenómeno físico se necesitan dos cosas. Primero, elegir una serie de variables físicas objetivas que se puedan medir para describir dicho fenómeno. Y segundo, un lugar apropiado para observarlo.

En la mecánica clásica, a este punto de observación se le denomina sistema de referencia (SdR de aquí en adelante) [137] y define un sistema sobre el que medir el movimiento. Aunque la

elección de este sistema puede ser arbitraria, hay algunas consideraciones importantes que se deben tener en cuenta.

La primera consideración se establece entre SdR inercial y SdR no inercial (SdRNI). Un SdR inercial (SdRI) es aquel que está en reposo o en movimiento rectilíneo uniforme [138]. Todos los demás SdR (incluyendo aquellos que no se mueven pero rotan) se consideran no inerciales. Un observador situado sobre (atado a) un SdRI tiene la ventaja de percibir el movimiento como el resultado de interacciones reales entre objetos, mientras que un observador situado en un SdRNI necesita la presencia de las llamadas fuerzas ficticias (porque sólo se perciben en un SdRNI y no en un SdRI) para ser capaz de comprender lo experimentado usando las leyes de Newton [139].

Mientras que un SdRI es preferible si queremos medir con exactitud la posición o velocidad de un vehículo con respecto a un punto fijo, un SdRNI es más apropiado si lo que queremos es analizar las sensaciones que una persona puede experimentar dentro del mismo, porque cuando pilotamos un vehículo lo hacemos montados sobre un SdRNI. Cuando el vehículo acelere y cambie de dirección, cualquier SdR que se mueva junto a él será inevitablemente no inercial. Por el contrario, cualquier SdR colocado en algún punto fijo de La Tierra será considerado inercial (esto es en realidad falso porque el planeta se mueve, pero podemos considerar que este movimiento no es importante a este nivel).

La segunda decisión acerca de los SdR es dónde colocar su origen de coordenadas. Si el SdR está anclado en un punto de La

Tierra, esta decisión es puramente arbitraria ya que no afectará a las mediciones que se hagan sobre él. Por el contrario, si el SdR está unido a un objeto en movimiento (como puede ser el propio vehículo), entonces pueden aparecer fuerzas ficticias que hemos de tener en cuenta [138].

Si el SdR está localizado en el centro de masas (CdM) del cuerpo en movimiento, pueden aparecer dos fuerzas ficticias: una causada por la aceleración relativa del SdR en línea recta (fuerza de D'Alembert) y otra causada por la rotación del SdR con respecto a un punto externo al mismo (fuerza centrífuga) [140], si se da el caso. Si el SdR no está situado en el CdM del objeto, dos fuerzas ficticias adicionales pueden aparecer: una causada por cualquier rotación del cuerpo con respecto a su CdM (fuerza de Coriolis) [141] y otra causada por cualquier variación en la velocidad de rotación del objeto (es decir, por la presencia de aceleración angular) con respecto a su CdM (fuerza de Euler o azimutal) [142] si dicho evento ocurre.

La tercera y última decisión es acerca de la elección y dirección de los ejes de coordenadas. Esta decisión es independiente de si el sistema es inercial o no, y en la mayoría de casos no afecta a los análisis de movimiento, por lo que es una decisión arbitraria. Sin embargo, lo natural es alinearlos con las direcciones de movimiento natural del vehículo: hacia adelante, verticalmente y lateralmente.

3.1.1.1 – Ejemplificación con un Bote de Rescate

Para ejemplificar lo antedicho, vamos a aplicar lo que hemos visto a la caracterización del movimiento de un bote de rescate en el agua [143]. Como apuntamos previamente, para realizar la caracterización del movimiento de un vehículo, lo primero que debemos hacer es especificar claramente qué variables físicas se desean medir. De un bote de rescate, como de cualquier otro vehículo nos interesará conocer su estado dinámico y cinemático. Es decir, su posición, velocidad y aceleración. También es muy importante conocer cuál es su orientación en todo momento y sus velocidades y aceleraciones angulares.

Como casi todos los vehículos son controlados por humanos, también puede ser importante conocer cómo es utilizado el control del vehículo. En el caso del bote, es interesante conocer cómo se comporta el timón y la palanca de aceleración, que son los dos únicos controles de que dispone.

Por último, existen ciertas peculiaridades de cada vehículo que se deben tener en cuenta. En el caso del bote de rescate, al no ir protegido por un parabrisas, uno de los aspectos interesantes para la sensación de movimiento es el viento, por lo que decidimos medir también el viento aparente.

Por ello, con todas estas consideraciones en mente, para conocer la posición y velocidad del bote, usaremos un SdR de tipo GPS, fijo en La Tierra, que llamaremos SdR-1. El sistema se elige de tipo inercial para poder medir la velocidad y posición de forma absoluta. Su centro es arbitrario ya que no es importante.

Además, usaremos un SdRNI unido al bote, para medir su aceleración lineal, su velocidad angular y su aceleración angular. Como queremos caracterizar el movimiento del bote para reproducirlo, lo más sencillo es situar el origen del SdR en el CdM del vehículo, de manera que no haya fuerzas de Coriolis ni de Euler. A este SdR le llamaremos SdR-2. Aunque otros trabajos usan otras convenciones, usaremos la convención Y hacia adelante, X hacia la derecha, Z hacia arriba, que ya vimos y que es la más empleada en simulación de vehículos. Usaremos convenciones náuticas para los ángulos de giro (*yaw*, *pitch*, *roll*) y los desplazamientos (*surge*, *sway*, *heave*), como podemos ver en la Figura 3.1.

Como SdR-2 estará siempre orientado como el bote, y también querremos conocer cómo se comporta el bote con respecto al plano XY (el mar), para por ejemplo saber cuánto inclina, definiremos también un SdR situado en el mismo punto que SdR-2 pero alineado con el plano XY. Le llamaremos SdR-3.

Finalmente, como también deseamos saber cuál es en todo momento el rumbo del bote, definiremos SdR-4, situado en el mismo punto que SdR-2 y SdR-3, pero orientado como el sistema fijo SdR-1.

3.1.2 – Sensorización

Una vez elegido el *qué* y el *dónde* (qué se quiere medir y en qué lugares), el siguiente paso es el *cómo*.

Este paso es el menos generalizable de todos, dado que existen numerosas magnitudes y sensores para medirlas. Además, dentro de un mismo tipo de sensores existen aparatos de distintos precios, por lo que el presupuesto disponible afectará a este punto de manera notable. Sin embargo, se pueden nombrar algunas consideraciones generales interesantes comunes a cualquier sensorización de cualquier tipo de vehículo.

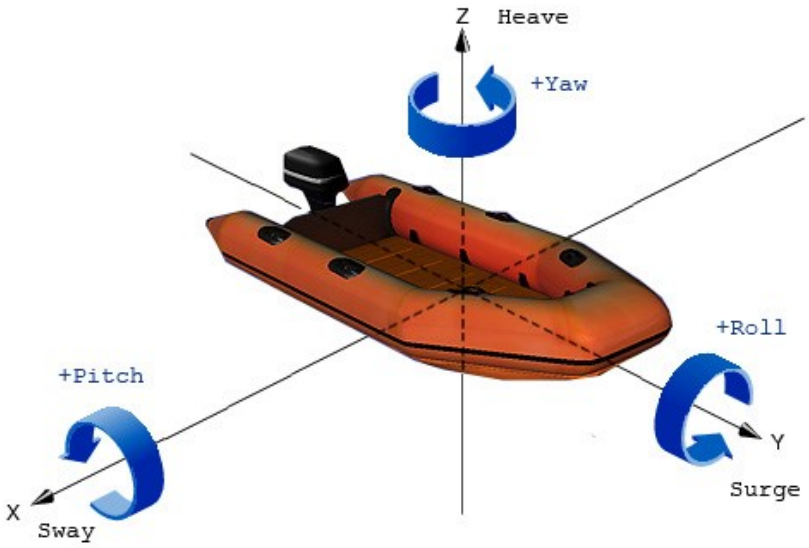

Figura 3.1 - *SdR del bote de rescate*

Lo primero que hay que tener claro es qué tipo de dispositivos nos pueden permitir tomar las medidas que deseamos. Dado que lo que deseamos es caracterizar el movimiento, lo normal es tomar medidas de aceleraciones y velocidades tanto lineales como angulares, por lo que es habitual emplear acelerómetros, inclinómetros, giróscopos, velocímetros y

registradores de posición (como aparatos GPS). Adicionalmente, y dependiendo del tipo de vehículos, se pueden emplear otros.

Después, hay que saber cuál es la precisión de los sensores que se piensan utilizar. Por fortuna, dado que lo que nos interesa principalmente aquí es realizar una caracterización del movimiento para entenderlo y ser capaces de construir una plataforma de movimiento para el mismo, la precisión no necesita ser extraordinaria.

Otro aspecto interesante es el rango de los datos sensorizados. El rango de medidas de los dispositivos debe ser suficiente para medir todas las magnitudes deseadas. Como lo normal es tomar medidas de aceleraciones y velocidades tanto lineales como angulares, se debe elegir un dispositivo capaz de registrar dichas magnitudes en su rango completo de extensión, por lo que se debe tener una idea aproximada de los órdenes de magnitud de estas variables para poder elegir el sensor.

Y por último, pero no por ello menos importante, la frecuencia de sensorización de los sensores debe ser lo suficientemente alta como para analizar el vehículo en el dominio de la frecuencia hasta los límites que se necesite. Por fortuna o por desgracia, las plataformas de movimiento no admiten movimientos de muy alta frecuencia por lo que tampoco necesitamos llegar a frecuencias de sensorización elevadas. Dado que las plataformas de movimiento presentan normalmente grandes inercias, por ser elementos mecánicos con masas elevadas, es raro ver alguna plataforma capaz de generar movimientos a frecuencias superiores a 10 Hz.

3.1.2.1 – Ejemplificación con un Bote de Rescate

Siguiendo con el ejemplo del bote de rescate, para realizar las medidas y obtener los datos deseados, empleamos un PC portátil que subimos a bordo de un bote real. El PC estaba conectado a una serie de sensores situados en la embarcación. En particular, empleamos 2 acelerómetros de 3 ejes, 2 giróscopos, 1 registrador de posiciones GPS, 1 brújula digital, 1 anemómetro digital, 1 micrófono y 1 cámara.

Los acelerómetros y giróscopos fueron empleados para medir aceleraciones y velocidades angulares. Usamos 2 de cada uno de esos dispositivos. Un par situado en SdR-2, y el otro en el timón para sensorizar la dirección. Aunque podíamos haber utilizado acelerómetros y giróscopos separados, empleamos 2 *Nintendo Wii Remote* (conocido como *Wiimote*) equipados con el accesorio *Wii Motion Plus*. Aunque un *Wiimote* pueda parecer un dispositivo de juego (y lo es), es al mismo tiempo un potente (pero asequible) aparato equipado con un acelerómetro de 3 ejes y (cuando lo usas junto con el *Wii Motion Plus*) un giróscopo de 3 ejes. Existen dispositivos más sofisticados, pero elegimos éste porque sus mediciones son muy sencillas de leer desde un PC en tiempo real, es ciertamente barato (algo importante para los reducidos presupuestos de investigación) y su rango de sensorización, precisión y frecuencia de actualización eran suficientes para nuestros propósitos. La frecuencia de sensorización de estos dispositivos es de al menos 50 Hz (ver Apéndice A). Dado que los movimientos del bote de rescate no son vibratorios, no es de esperar que superen unos pocos Hz. En cuanto al rango, permiten sensorizar aceleraciones entre -3 y 3 g, y

velocidades angulares ente -500 °/s y 500 °/s. Ambos rangos debieran ser, a priori, suficientes para este tipo de vehículo.

Como se puede apreciar en la Figura 3.2, el SdR que emplea el *Wiimote* es muy similar al que hemos elegido nosotros para la caracterización.

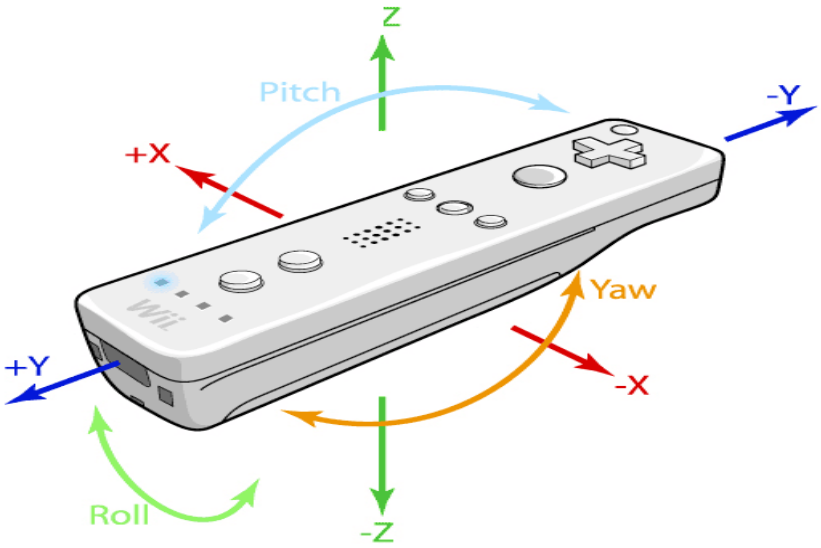

Figura 3.2 - *SdR de un Wiimote (© Nintendo Koppai)*

Aunque sin usar los giróscopos (sólo usando los acelerómetros) es posible obtener la orientación (al menos *pitch* y *roll*), preferimos combinar la información tanto de los giróscopos como de los acelerómetros para minimizar los problemas que presentan por separado [144]. Como se explica con posterioridad, el procesamiento simultáneo de datos lineales y angulares es preferible para obtener los ángulos de *pitch* y *roll*. Los acelerómetros dan una mejor medida cuando el dispositivo está

parado, pero los giróscopos dan mejor medida cuando el dispositivo está en movimiento, especialmente cuando éste es corto. Por tanto, el uso simultáneo de ambos dispositivos hace que el espectro de frecuencia de los movimientos que es posible analizar se amplíe.

La medición del rumbo (*yaw*) es más compleja que la de los otros dos ángulos (*pitch* y *roll*) porque el eje de giro coincide con el eje de la gravedad y por tanto los acelerómetros no se pueden emplear. Se podría sensorizar con el giróscopo pero el error en la medición es acumulativo (se calcularía integrando la velocidad angular alrededor del eje Z, lo cual produciría una acumulación de error a lo largo del tiempo), por lo que decidimos usar una brújula digital magnética (que calcula el rumbo mediante un imán) situada en el origen de SdR-2.

Para medir la posición y velocidad del vehículo empleamos un dispositivo GPS, situado en el origen de SdR-2, capaz de medir y registrar con cierta precisión la posición y velocidad, mientras que para medir la velocidad del viento, empleamos un anemómetro digital.

Finalmente, utilizamos también un micrófono y una *webcam*, que se emplearon para grabar las sesiones de prueba y ser capaces de recordar y revivir posteriormente las maniobras y pruebas realizadas.

Para una descripción más detallada de los dispositivos empleados, consúltese el Apéndice A.

Una vez elegidos los dispositivos de sensorización, es necesario decidir la localización adecuada para cada uno de ellos. En el caso de ejemplo del bote de rescate, la localización elegida para los sensores es la que se observa en la Figura 3.3.

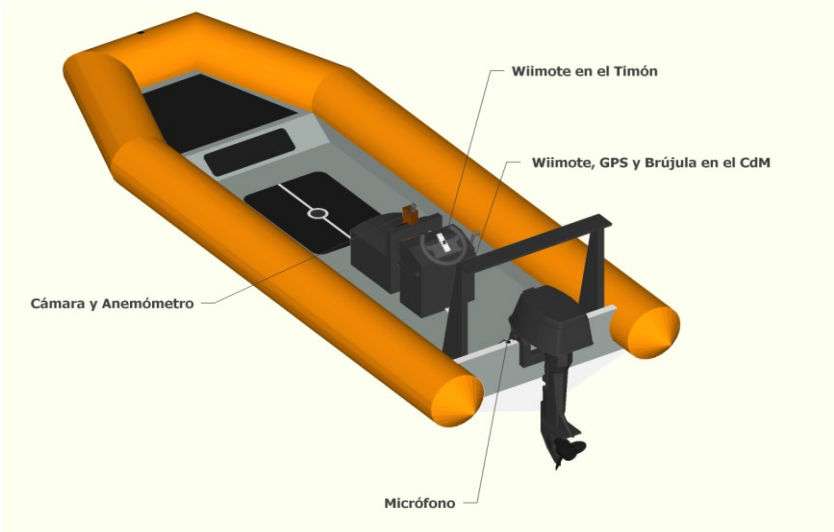

Figura 3.3 - *Localización de los sensores en el bote*

En este caso, los sensores más importantes son los que se sitúan en el CdM: el GPS, un *Wiimote* y la brújula digital. El GPS se emplea para medir posición y velocidad en coordenadas del mundo SdR-1 (o lo que es lo mismo para saber cuál es la posición de SdR-2 respecto a SdR-1), mientras que el *Wiimote* nos da la aceleración y velocidad angular con respecto a SdR-2, de las que podemos calcular otras muchas variables físicas como *pitch*, *roll*, aceleración angular, etc. El ángulo de *yaw*, es calculado mediante la brújula digital.

El problema es cómo calcular la posición del CdM para situar los dispositivos. Para ello, empleamos un modelo 3D del vehículo con densidades estimadas. Es obvio que esto es una aproximación, dado que es imposible que los tripulantes permanezcan inmóviles, pero la posición calculada del CdM debería ser una buena estimación y una mejor es probablemente muy difícil de encontrar.

El *Wiimote* situado en el timón se empleó para sensorizar el rumbo deseado. Como los datos llevan marcas de tiempo, es sencillo sincronizarlos con los datos de otros dispositivos. La cámara se situó encima del puesto de conducción. Dicha cámara dispone de un micrófono interno que se usó para registrar comentarios personales sobre las pruebas. El anemómetro digital se situó cerca de la cámara para registrar el viento aparente sentido por el piloto. Finalmente, el micrófono se situó en la popa para registrar el sonido del motor y así tener una estimación del régimen de revoluciones en función del sonido, ya que no nos fue posible sensorizar la palanca del acelerador.

3.1.3 – Análisis del Movimiento

El siguiente paso en nuestra caracterización es analizar los datos recogidos por los sensores para poder extraer conclusiones sobre cómo es el movimiento, qué magnitud tiene, qué forma, que direcciones de movimiento son las dominantes, cómo es dicho movimiento en el dominio de la frecuencia, etc. Aunque los siguientes apartados tratan sobre el análisis de los datos recogidos sobre el bote de rescate, el procedimiento de análisis se puede

extrapolar a cualquier otro tipo de vehículo con mínimos cambios, ya que lo que vamos a analizar son magnitudes físicas medibles en cualquier vehículo, y evidentemente las manipulaciones algebraicas y cálculos que realizaremos no son exclusivas de un bote de rescate.

3.1.3.1 – Consideraciones Preliminares

Antes de analizar los datos, hay algunas consideraciones específicas sobre la validez de nuestra caracterización que es conveniente comentar.

La primera es sobre el vehículo. Existen muchos vehículos (terrestres, aéreos, marinos, etc.). Elegimos el campo de los vehículos marinos por estar mucho menos explorado que los otros campos. Dentro de los vehículos marinos también hay muchos tipos (balsas, barcos, submarinos, etc.) con diferentes sistemas de propulsión (remos, velas, motores, etc.). No pretendemos (ni podemos) estudiar todos ellos por lo que nos centraremos únicamente en botes de rescate de tamaño medio propulsados por un motor de combustión con hélice. Se pueden hacer análisis con otros vehículos y serían igual de ejemplificadores. Escogimos este tipo de embarcación por ser un vehículo marino bastante habitual y porque existen necesidades de entrenamiento sobre ellos que se pueden suplir con aplicaciones de RV.

Incluso aunque nos centremos en un solo tipo de bote, hay muchos fabricantes y configuraciones. Si analizáramos un coche, nos ocurriría lo mismo ya que tendríamos múltiples fabricantes y modelos, por lo que no sería lo mismo analizar un utilitario que un

vehículo de carreras. En cualquier caso, aunque las magnitudes analizadas tengan rangos diferentes, ambos compartirían comportamientos y rasgos similares que los caracterizarían. Esa similitud es consecuencia de que, sobre ellos, se aplican las mismas leyes físicas, aunque sea con diferentes valores de algunos parámetros. Lo mismo podríamos decir de un bote de rescate, ya que, aunque existen diferentes modelos, asumiremos que podemos estudiar uno de ellos y extraer de él la esencia del movimiento. En este análisis empleamos un bote de rescate rápido *Duarry Brio 620* propulsado por un motor *Suzuki DF-140* de 140 caballos de potencia. Este vehículo tiene una masa de 911 kg (aunque en la prueba iba cargado con unos 400 kg de peso) y un tamaño de 6.20 x 2.20 x 1.43 metros (largo x ancho x alto). Podemos considerar este vehículo como un bote de rescate estándar, y por eso lo elegimos.

Una segunda consideración es sobre la prueba en sí. Las condiciones del mar son muy cambiantes, por lo que pueden ser distintas de un día a otro y de puerto a mar abierta. Continuando la comparación automovilística, aunque el viento, los baches y la temperatura de la pista afectan al comportamiento de un coche, el oleaje, las corrientes y el viento tienen un mayor efecto en un bote que las anteriormente citadas variables sobre un coche. Por desgracia, es imposible realizar una evaluación en todo el espectro de condiciones naturales del entorno, ya que la naturaleza caótica del agua hace que ni siquiera dos pruebas seguidas sean iguales. Esto hace especialmente singular (y complicado) el análisis en este tipo de vehículos. Sin embargo, como sabemos que este tipo de embarcaciones no se construye para soportar grandes oleajes, probamos el bote en puerto y en mar abierto con un oleaje

relativamente moderado. Esta consideración particular no es tan acusada en otro tipo de vehículos.

La tercera consideración es sobre los sensores. Primero, la localización de los sensores nunca puede ser exacta al 100 %. Segundo, como dijimos, introducen inevitablemente cierto error en la medida. Somos conscientes de ello, pero como no estamos buscando un *tracking* 1:1 del movimiento sino una caracterización del mismo para extraer sus características más significativas, consideramos que este error no debe suponer un problema para nuestros objetivos.

La última consideración es sobre la conducción del vehículo. Es obvio que el modo en que se pilota/conduce/navega influye en los resultados. Por eso, en nuestras pruebas intentamos cubrir el mayor número de maniobras y rangos de velocidad posibles. Probamos el vehículo parado, a bajas velocidades, a velocidades medias, a toda velocidad, provocamos giros bruscos, acelerones, frenazos, etc.

3.1.3.2 – Descripción de las Pruebas

Para cubrir todo ese amplio rango de maniobras, las pruebas consistieron en 2030 segundos (alrededor de media hora) de manejo del bote cerca del puerto de Barcelona (ver Figura 3.4), que pueden dividirse en ciertos sub-periodos de tiempo (ver Tabla 3.1). Todas las medidas de los sensores fueron guardadas en el PC y luego sincronizadas entre sí gracias a marcas de tiempo.

Figura 3.4 - *El bote de rescate en el Puerto de Barcelona*

3.1.3.3 – Metodología de Análisis

En este apartado vamos a explicar cómo analizar los datos capturados. Aunque lo que vamos a analizar son los datos extraídos de los sensores del bote de rescate, la formulación e incluso la manera de extraer conclusiones sobre ellos son completamente extrapolables a otros vehículos.

En primer lugar, de los acelerómetros y giróscopos de 3 ejes localizados en el CdM podemos extraer la fuerza específica (los acelerómetros miden fuerza específica, no aceleración) y la velocidad angular del vehículo en SdR-2. Les llamaremos $\overrightarrow{F_2}$ y $\overrightarrow{\omega_2}$,

ya que están referidos al sistema SdR-2. Mientras que $\overrightarrow{F_2}$ se mide en veces la gravedad de La Tierra (g), $\overrightarrow{\omega_2}$ se expresa en °/s. Como dichas magnitudes son empleadas como entradas por los algoritmos de *washout* y por el sistema vestibular humano, son, por sí mismas, magnitudes que nos proporcionan bastante información. Sin embargo, si manipulamos estas variables apropiadamente podremos obtener más información.

Rango (segs)	*Descripción*
0-260	Descartado (puesta en marcha)
260-440	Amarrado
440-550	Maniobras en puerto
550-800	Navegación lenta cerca del puerto
800-980	Aceleración progresiva
980-1320	Navegación a velocidad máxima
1320-1355	Frenazo y luego parado
1355-1553	Navegación moderada en mar abierto
1553-1613	Navegación, frenazo y marcha atrás
1613-1770	Maniobras variadas
1770-1885	Giros completos a máxima velocidad
1930-2030	Parados en mar abierto

Tabla 3.1 - *Descripción de maniobras del bote de rescate*

Como $\overrightarrow{F_2}$ siempre subsume la aceleración de la gravedad, esta variable nos puede servir para calcular los ángulos de *pitch* y *roll* empleando las ecuaciones (3.1) y (3.2), siempre y cuando el sensor no esté sometido a ninguna otra fuerza que no sea la gravedad (3.3).

$$\theta = \operatorname{atan}\left(\frac{F_{2y}}{F_{2z}}\right) \quad (3.1)$$

$$\phi = \operatorname{atan}\left(\frac{F_{2x}}{F_{2z}}\right) \quad (3.2)$$

$$\text{cuando } \left|\overrightarrow{F_2}\right| \cong 1 \quad (3.3)$$

Recordemos que θ representa el ángulo de *pitch* y ϕ el de *roll*.

Aunque el sensor puede tener ruido, esto nos da una estimación bastante buena de la orientación en 2 de las 3 direcciones del espacio.

El *yaw* no puede ser calculado de esta manera, como ya explicamos. Por ello, ha de ser calculado mediante la integración de la velocidad angular alrededor del eje Z (ω_{2z}). El problema es que esto genera un gran error acumulativo, por lo que es preferible obtener el *yaw* de otra fuente, como puede ser la brújula digital, que obtiene la orientación gracias al campo magnético de La Tierra y no mediante integración de la velocidad angular.

Aunque el vehículo se mueva constantemente, el acelerómetro se puede considerar parado o en movimiento lineal uniforme en periodos de tiempo pequeños y concretos. Si definimos un umbral alrededor de 1 g (un acelerómetro mide 1 g en reposo o en movimiento lineal uniforme y 0 g en caída libre) y usamos las ecuaciones (3.1) y (3.2) sólo cuando la expresión (3.3) se cumpla, podremos calcular el *pitch* y el *roll* en determinados instantes. A partir de *yaw*, *pitch* y *roll* podemos obtener la

orientación del vehículo en forma de quaternion q. Para aquellos otros instantes en los que la expresión (3.3) no se cumpla, podemos usar $\overrightarrow{\omega_2}$ para calcular orientaciones sucesivas por integración de las anteriores (ecuaciones (3.4) y (3.5)) hasta que llegue otro momento en el que se cumpla la expresión (3.3):

$$\dot{q} = 0.5 * \left[\frac{0}{\overrightarrow{\omega_2}}\right] * q \qquad (3.4)$$

$$q = 0.5 * \int \left[\frac{0}{\overrightarrow{\omega_2}}\right] * q * dt \qquad (3.5)$$

Ahora que ya tenemos una estimación de la orientación del vehículo, podemos eliminar la gravedad de la medida de la fuerza específica, obteniendo la aceleración percibida en el CdM:

$$\overrightarrow{A_2} = \overrightarrow{F_2} + q^{-1} * \vec{g} \qquad (3.6)$$

$\overrightarrow{A_3}$ y $\overrightarrow{A_4}$ se pueden calcular fácilmente a partir de $\overrightarrow{A_2}$:

$$\overrightarrow{A_3} = q' * \overrightarrow{A_2} \qquad (3.7)$$

$$\overrightarrow{A_4} = q * \overrightarrow{A_2} \qquad (3.8)$$

donde q' sería el quaternion con el *yaw* fijado a 0.

La aceleración angular $\overrightarrow{\alpha_2}$ se puede calcular directamente por diferenciación de $\overrightarrow{\omega_2}$.

De la misma forma que el *yaw*, la velocidad y la posición se pueden aproximar por integración y doble integración

(respectivamente) del vector $\overrightarrow{A_4}$, pero a medida que el tiempo pasa el error se acumularía. Por eso, la posición y la velocidad las obtendremos directamente del GPS. Sin embargo, como su frecuencia de actualización es baja, usaremos la información de los acelerómetros para refinar el cálculo entre dos medidas de GPS, de modo similar a como hicimos con q. Dichas magnitudes se obtienen relativas a SdR-1. Les llamaremos $\overrightarrow{s_1}$ y $\overrightarrow{v_1}$.

El viento aparente $|\overrightarrow{v_w}|$ se obtiene directamente del anemómetro.

3.1.3.4 – Rangos de Movimiento

El módulo físico es capaz de simular de modo bastante exacto el estado físico de un vehículo virtual sin verse afectado por valores grandes de aceleración, o velocidad, siempre y cuando las ecuaciones sean correctas y la simulación se calcule a una frecuencia suficientemente alta [32]. Sin embargo, como lo que buscamos principalmente es reproducir el movimiento del vehículo con una plataforma de movimiento, y las plataformas son dispositivos reales con límites físicos reales, los valores pico de aceleración, orientación, etc. sí son importantes para saber si la plataforma será o no capaz de realizar la simulación sin alcanzar sus límites físicos, incluso en el peor caso.

Dichos valores pico deben ser interpretados con cuidado ya que normalmente son consecuencia de un instante único y preciso, pero proporcionan información relevante. La Tabla 3.2 muestra los valores pico observados durante nuestros experimentos con el

bote de rescate. Las variables de interés son *pitch*, *roll*, *yaw*, $\overrightarrow{A_3}$, $\overrightarrow{\omega_2}$, $\overrightarrow{v_1}$, y $|\overrightarrow{v_w}|$.

De los valores de la Tabla 3.2 podemos ver que el vehículo tiende hacia el *pitch* positivo. En el caso del bote de rescate esto es una consecuencia del diseño del bote. Dado que la fricción del agua es unas 1000 veces superior a la del aire, cuanto más inclinado hacia arriba esté el vehículo, menos agua tendrá que mover y más rápido podrá desplazarse. Cuando el motor mueve la hélice, la hélice propulsa el agua y ésta reacciona con una fuerza de reacción que no sólo mueve el bote hacia adelante, sino que además crea un torque de elevación sobre el eje X, que hace inclinar la proa de la embarcación.

A diferencia del *pitch*, el *roll* sí que parece simétrico alrededor del eje Y. Además, los valores pico son menores que los del *pitch*.

Como el bote puede hacer giros completos, no es sorprendente que el ángulo de *yaw* varíe entre 0 y 360º.

La aceleración frontal (A_{3y}) alcanza unos valores pico de 0.4 g y -0.8 g respectivamente. Las aceleraciones negativas están relacionadas con la fricción del agua (que actúa de freno), dado que el agua posee un alto coeficiente de fricción, y es capaz de frenar el vehículo rápidamente. Las aceleraciones positivas deben estar relacionadas con el motor. Sin embargo, análisis posteriores revelarán que en realidad son también consecuencias, principalmente, del oleaje. La aceleración lateral (A_{3x}) alcanza valores pico de ±0.5 g aproximadamente, lo cual no es una gran

aceleración. Es simétrica porque es consecuencia de giros y golpes de mar, que no tienen una dirección privilegiada. La aceleración vertical (A_{3z}) está en el rango de -2 g a 1 g, que es un rango mucho mayor que las otras dos componentes. Esto demuestra que uno de los efectos más notables cuando se pilota un bote de rescate rápido son los golpes de mar cuando la embarcación salta de ola en ola. Tanto los picos positivos como los negativos son consecuencia de golpes de mar. Estas aceleraciones, como veremos, son cortas pero intensas, y por tanto importantes de simular.

Magnitud	*Mínimo*	*Máximo*	*Unidad*
Yaw	0.0	359.99	°
Pitch	-5.3	+21.73	°
Roll	-19.18	+19.86	°
Aceleración frontal	-0.83	+0.42	g
Aceleración lateral	-0.45	+0.61	g
Aceleración vertical	-2.11	+1.15	g
Velocidad angular X	-118.1	+132.9	°/s
Velocidad angular Y	-144.8	+110.9	°/s
Velocidad angular Z	-42.6	+58.85	°/s
Velocidad de navegación	-5.64	26.71	nudos
Viento aparente	0.32	32.46	nudos

Tabla 3.2 - *Rangos de las diversas magnitudes*

En cuanto a las velocidades angulares ($\overrightarrow{\omega_2}$), éstas son consistentes con el diseño y movimiento del bote, porque son menores alrededor del eje Z (*yaw*) que alrededor de los otros dos ejes. De hecho, un bote sufre rotaciones de frecuencia media sobre

los ángulos *pitch* y *roll*, pero menores sobre el ángulo de rumbo (*yaw*). La velocidad angular alrededor del eje Y (*roll*) es superior que sobre el eje X (*pitch*). Esto es consistente con el hecho de que este tipo de embarcaciones son más estrechas que largas por lo que el momento de inercia sobre el eje X es mayor que sobre el eje Y.

La velocidad de navegación ($|\overrightarrow{v_1}|$) oscila entre unos pocos nudos cuando se mueve hacia atrás, y unos 25 nudos hacia adelante. Como este vehículo no está diseñado para navegar hacia atrás a grandes velocidades (porque dada su inclinación natural se hundiría), su velocidad marcha atrás es muy reducida.

El viento aparente ($|\overrightarrow{v_w}|$) oscila entre 0 y más de 30 nudos.

3.1.3.5 – Análisis en el Dominio del Tiempo

Velocidad, Pitch, Roll y Yaw

Para estándares terrestres, la velocidad que puede alcanzar un bote de rescate no es muy elevada. Como se puede apreciar en la Figura 3.5 (donde se muestra la velocidad y el *pitch* con respecto al tiempo), la velocidad máxima está ligeramente por encima de los 25 nudos (menos de 50 km/h). Sin embargo, para vehículos marinos esto entra dentro de la categoría de vehículo rápido. Dado que el bote se puede probar descargado y con otro motor mucho más potente (o incluso con varios motores), la velocidad podría ser superior (30-40 nudos).

Sin embargo, lo interesante de este análisis no es el valor máximo sino la forma en la que se alcanza la velocidad. A diferencia de un coche, un bote de rescate no puede mantener una velocidad fija de forma precisa, porque está constantemente golpeando el agua, la cual actúa de freno. Esta es la razón por la que aparecen los picos que se observan en el gráfico en el sub-periodo de máxima velocidad (980 a 1320 segundos). Lo mismo se puede decir acerca de una parada completa. Por último, la marcha atrás no permite ir más allá de 5-10 nudos, porque la embarcación se hundiría.

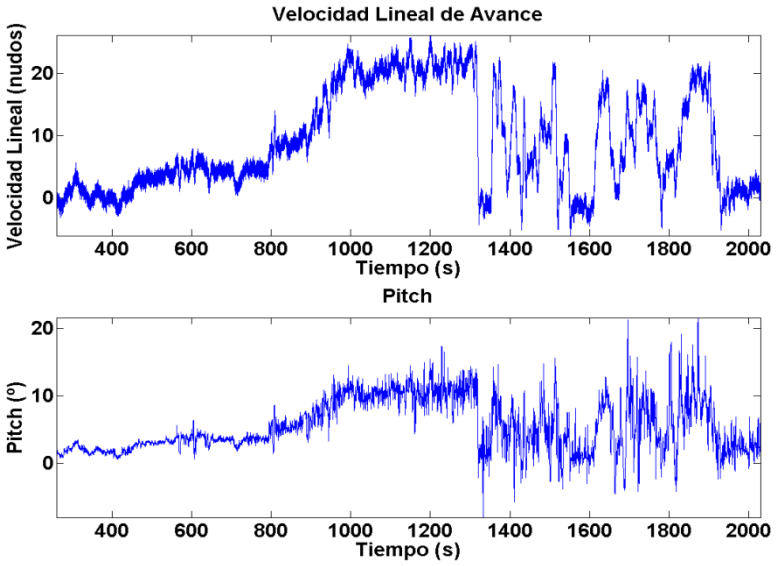

Figura 3.5 - *Velocidad* ($|\overrightarrow{v_1}|$) *y pitch vs tiempo*

En la Figura 3.6 podemos ver el *pitch* y el *roll* con respecto al tiempo. Se puede apreciar que su comportamiento es muy

diferente. Ambos dependen de la velocidad y del oleaje, pero de una manera distinta. Mientras el *pitch* tiende a incrementarse con la velocidad (por el torque de elevación que genera el propulsor) y se ve afectado por las olas, el *roll* no parece verse muy afectado por la velocidad. Más bien al contrario, cuando el vehículo está parado a merced de las olas, el *roll* es más alto que el *pitch* (especialmente en mar abierto).

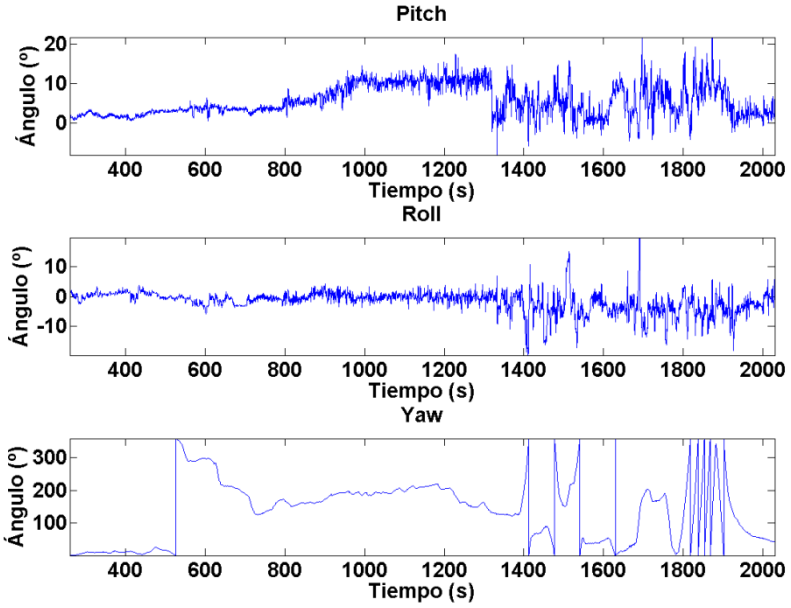

Figura 3.6 - *Pitch, roll y yaw*

Para corroborar el efecto de la velocidad en ambos ángulos calculamos la correlación entre la velocidad de avance y el *pitch*, dando un resultado de 0.826. Esto indica que sabiendo la

inclinación del bote podemos hacernos una buena idea de su velocidad. La correlación entre el *roll* y la velocidad fue de -0.118, por lo que están ligeramente inversamente correlacionados.

El ángulo de *yaw* varía con el rumbo (el timón) y su análisis temporal no revela demasiada información.

Aceleración Lineal

La conclusión más importante que podemos sacar del análisis de la aceleración lineal es que, a pesar de ser un vehículo con 140 caballos de potencia, no se observan trazas de aceleración Y sostenida. Podemos apreciar esto en la Figura 3.7 (donde se muestra la aceleración lineal respecto al tiempo).

A pesar de que un vehículo de este tipo podría alcanzar los 30-40 nudos, le cuesta bastante tiempo alcanzar dicha velocidad y la aceleración Y media es menor que la de un coche utilitario. De hecho, si computamos la aceleración media de avance en el tramo 890-910 segundos (zona de máxima aceleración) el resultado es menos de 0.2 g. Además, dado que se producen golpes frecuentes con las olas, está aceleración no es casi nunca constante. De hecho, una mirada a la gráfica de aceleración nos hace ver lo difícil que es deducir en ella si el vehículo está acelerando, frenando o tratando de llevar una velocidad constante. Es más, los picos presentes en la gráfica en el periodo 1350-1750 segundos son picos causados por las olas, dado que éste es el periodo de prueba en mar abierto.

La ausencia de aceleración Y sostenida es una gran noticia para el diseño de una plataforma de movimiento para este

vehículo, porque las aceleraciones sostenidas provocan desplazamientos largos. Este es un problema grave en simuladores aeronáuticos y de automoción. En cuanto a las aceleraciones X y Z, se observa un comportamiento similar, aunque es notable que la aceleración vertical es superior en magnitud, por lo que parece que la dirección vertical es la dirección lineal donde se produce mayor cantidad de movimiento.

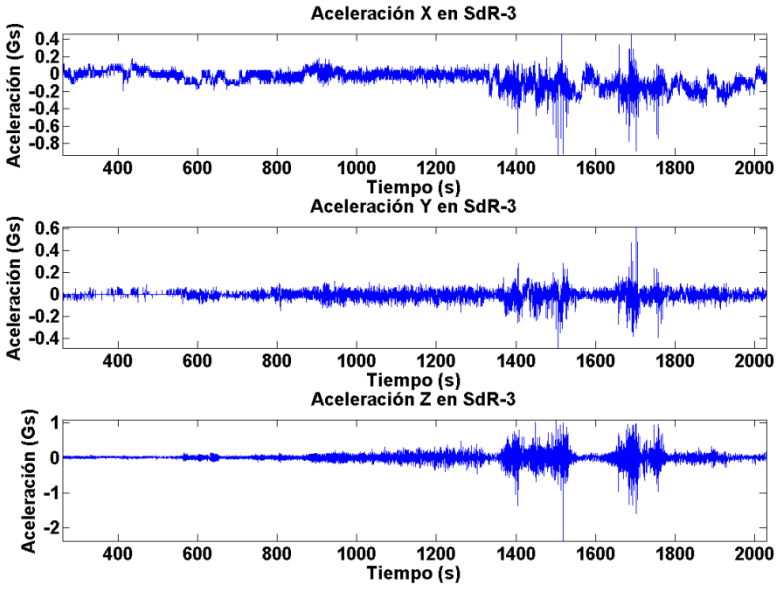

Figura 3.7 – *Aceleración lineal* ($\overrightarrow{A_3}$)

Velocidad y Aceleración Angulares

La gráfica de la velocidad angular (Figura 3.8) revela que los giros en X e Y son diferentes a los que se realizan sobre el eje Z.

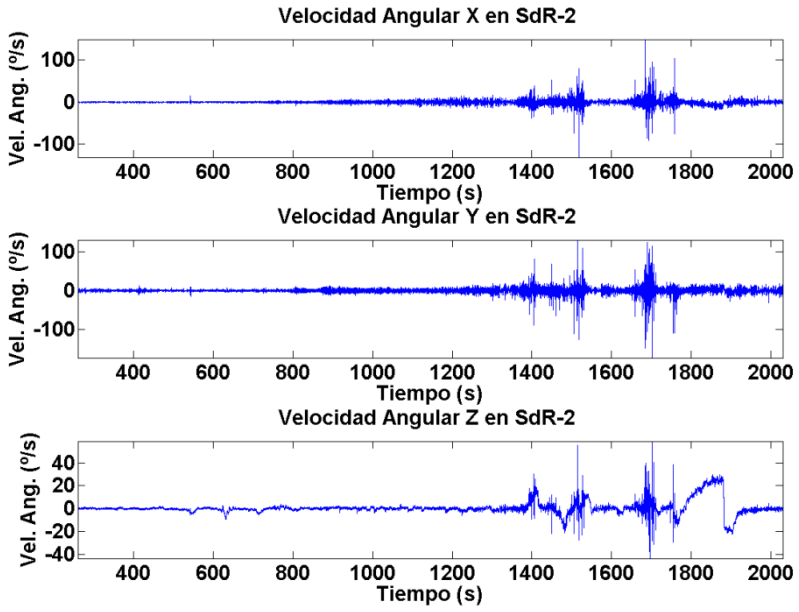

Figura 3.8 - *Velocidad angular* ($\overrightarrow{\omega_2}$)

Las velocidades angulares alrededor de X e Y alcanzan los 100 °/s (aunque algunos de estos valores son picos que no representan la situación media), pero no hay velocidades angulares sostenidas en estos dos ejes. Sin embargo, en el eje Z, las velocidades son menores (unos 50 °/s) pero hay un cierto grado de continuidad en la velocidad angular. Esto se aprecia especialmente en el periodo 1770-1885 donde podemos apreciar velocidad angular Z sostenida provocada por giros de timón (a babor o estribor) de forma sostenida.

La gráfica de la aceleración angular experimentada por el bote (ver Figura 3.9) nos demuestra que las 3 componentes de la aceleración angular presentan forma sinusoidal, con picos abruptos consecuencia de los golpes con el mar pero sin apreciarse aceleración angular sostenida.

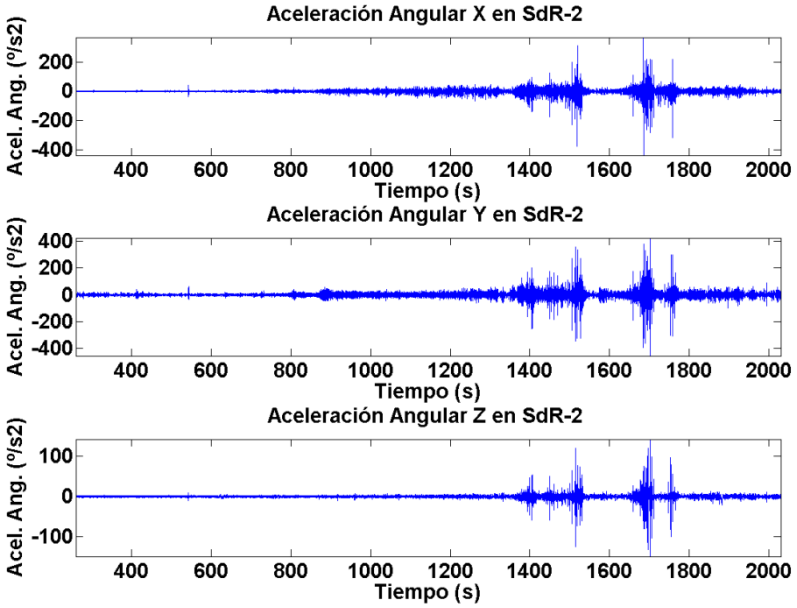

Figura 3.9 – *Aceleración angular* ($\overrightarrow{\alpha_2}$)

Esto significa que, a la hora de construir una plataforma de movimiento para reproducir cambios de orientación en los ejes X e Y, necesitaríamos motores relativamente potentes, pero la ausencia de aceleración sostenida nos permitiría mantenernos dentro de los límites de la plataforma. Sin embargo, la presencia de

velocidad angular sostenida en el eje Z nos indica que necesitaríamos una plataforma con una excursión muy larga en el ángulo de *yaw* para poder reproducir el movimiento. En cualquier caso, como las aceleraciones angulares no son sostenidas (ni tan siquiera en el eje Z), la plataforma podría simular dichos movimientos con un algoritmo de *washout* correctamente ajustado.

Timón

En la Figura 3.10 podemos ver el ángulo del timón respecto del tiempo. Como se puede apreciar, su forma se asemeja a una secuencia de ondas cuadradas escalonadas, ya que los cambios son abruptos, aunque relativamente infrecuentes. Lo normal es mantener el timón fijo para seguir un rumbo y de tanto en tanto, provocar un giro. Dado que en el mar no hay carreteras, casi siempre se eligen trayectorias rectas, a no ser que se trate de maniobrar para realizar alguna tarea o esquivar algún escollo. Por ello, esta variable no nos aporta demasiada información.

Viento

El viento es una de claves más importantes de la navegación marina. A diferencia de otros vehículos, en los que el viento no incide directamente, en un bote de rescate, la sensación de viento representa una clave perceptual importante. Dado que no es una clave inercial no se puede reproducir con una plataforma de movimiento, pero existen medios para introducir este tipo de claves en simuladores (por ejemplo emplear un ventilador controlado por computador).

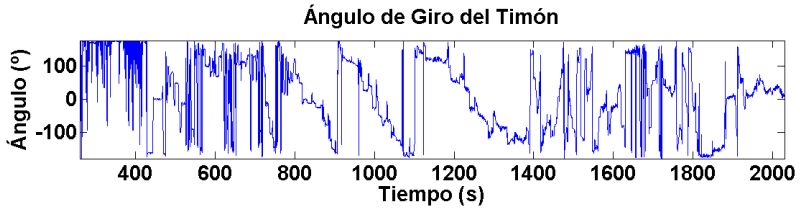

Figura 3.10 – *Ángulo del timón*

En nuestras pruebas medimos el viento aparente percibido por el piloto. Es bastante obvio que el viento depende de la velocidad del vehículo y de las corrientes de aire presentes en el entorno del vehículo. En ausencia de viento externo, el viento aparente debería coincidir con la velocidad de avance del vehículo, y nos serviría además para obtener una nueva estimación de la velocidad de avance.

En nuestras pruebas, soplaba una suave brisa de 4 nudos del noreste. En la Figura 3.11 podemos ver el viento aparente en la posición del piloto respecto del tiempo.

Es fácil comprobar que el viento aparente está altamente correlacionado con la velocidad de avance. En nuestro experimento, la correlación fue de 0.778. La correlación no es 1 por las rachas de viento. En cualquier caso esto significa que para simular el viento podemos simplemente calcular la suma del vector velocidad invertido, un ruido gaussiano y el vector que represente la dirección del viento externo.

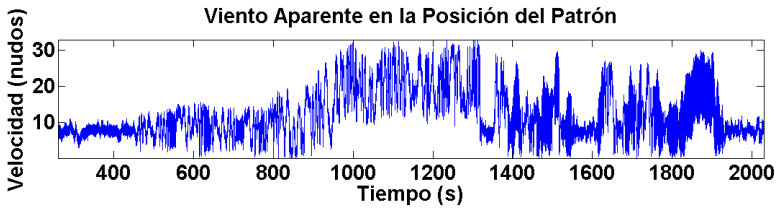

Figura 3.11 – *Viento aparente* ($|\overrightarrow{v_w}|$)

3.1.3.6 – Análisis en el Dominio de la Frecuencia

Aunque el análisis en el dominio del tiempo es muy útil para apreciar de forma cualitativa qué forma tienen las magnitudes dinámicas del vehículo y cuales son más importantes, un análisis de la distribución espectral de dichas magnitudes es necesario para poder asegurar qué claves de movimiento pueden ser compatibles con las capacidades de una plataforma de movimiento.

Movimiento Lineal

Una manera de analizar el espectro de frecuencias de una señal es usar la densidad espectral de potencia (PSD en inglés, de *power spectral density*). En nuestro caso, el PSD de la aceleración lineal (ver Figura 3.12) muestra una acusada atenuación por encima de 2-3 Hz, por lo que por encima de 10 Hz, los valores son probablemente ruido. Como podemos ver, el PSD de las componentes X, Y y Z no es muy distinto. Sin embargo, la atenuación en el eje Z es más pequeña y ocurre a mayores frecuencias. Esto es debido a que el movimiento en el eje Z es el

más vibratorio, ya que el mar golpea constantemente al bote en dirección vertical. En cualquier caso, este análisis nos demuestra que la forma de la aceleración del vehículo está más influenciada por el agua que por la propulsión (incluso en dirección Y), que debido a su alta densidad, impide movimientos vibratorios de alta frecuencia.

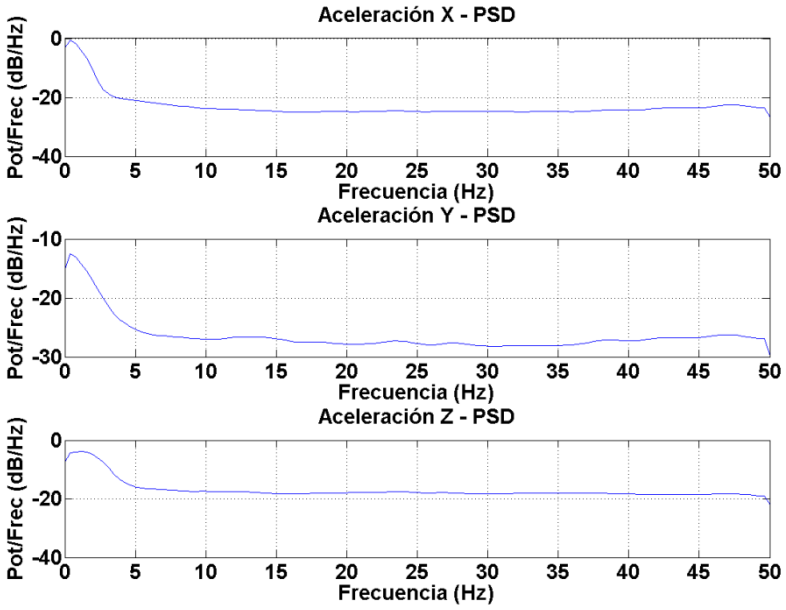

Figura 3.12 - *PSD de la aceleración lineal ($\overrightarrow{A_3}$)*

Como ya comentamos, las plataformas de movimiento presentan un comportamiento de filtro pasa-baja. En este vehículo, las aceleraciones no superan los 5 Hz en la mayoría de los casos.

Por tanto, si la plataforma eliminara las frecuencias superiores a 5 Hz no perderíamos mucha información.

Movimiento Angular

La aceleración angular y la velocidad angular muestran un comportamiento similar (ver Figuras 3.13a, 3.13b y 3.13c). Como este vehículo no posee suspensión, todo el movimiento vibratorio proviene del agua, y ciertamente, no es de alta frecuencia. Cuando un objeto (de densidad menor que el agua) cae al agua, es frenado por ella y luego empujado hacia arriba por la flotabilidad, produciendo un movimiento vibratorio, pero de baja frecuencia, ya que el proceso cuesta un cierto tiempo. El PSD de la aceleración angular muestra una atenuación constante desde 0 hasta 10 Hz. El PSD de la velocidad angular muestra una mayor atenuación, que comienza entre 1-2 Hz. Esto ratifica nuestra impresión previa de ausencia de movimientos angulares de muy alta frecuencia, que son difíciles de reproducir con una plataforma de movimiento, lo cual es una buena noticia en términos de simulación inercial.

El PSD de los ángulos *pitch*, *roll* y *yaw* revela también información. Primero, muestra la diferencia entre el *yaw* y los otros dos ángulos. El espectro del ángulo de *yaw* se atenúa uniformemente desde el principio. Eso significa que rara vez cambia mucho su periodicidad, por lo que no debemos esperar cambios abruptos en el *yaw*. Por el contrario, el PSD del *pitch* y del *roll* tienden a atenuarse en el entorno de los 3-4 Hz. Por tanto, podemos suponer que no habrá frecuencias de *pitch* y *roll* superiores a unos 5 Hz y tener esto en cuenta a la hora de construir una plataforma. Los picos por encima de 50 Hz deben

ser consecuencia del ruido (a medida que nos acercamos a la frecuencia de actualización de los sensores).

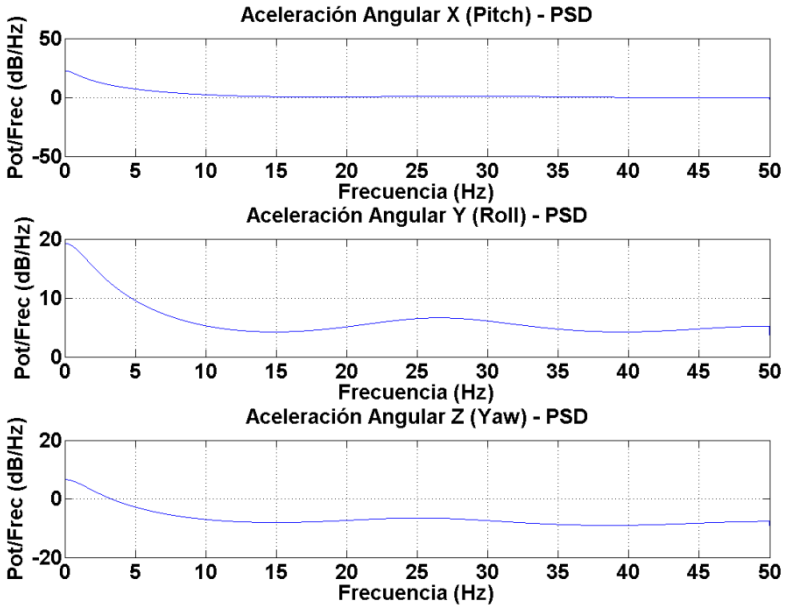

Figura 3.13a - *PSD de la aceleración angular* ($\overrightarrow{\alpha_2}$)

Todo esto es físicamente lógico. El timón puede cambiar rápido, pero esto se realiza pocas veces, y desde luego la embarcación no reacciona de forma instantánea al golpe de timón, por la inercia. Además, la cantidad de fuerza necesaria para cambiar el rumbo es considerable y no es habitual tener que volver a cambiar de rumbo en sentido contrario de forma continua. El *pitch* y el *roll*, sin embargo, cambian continuamente debido a las

olas, pero el momento de inercia de la embarcación evita que esto suceda de forma demasiado rápida.

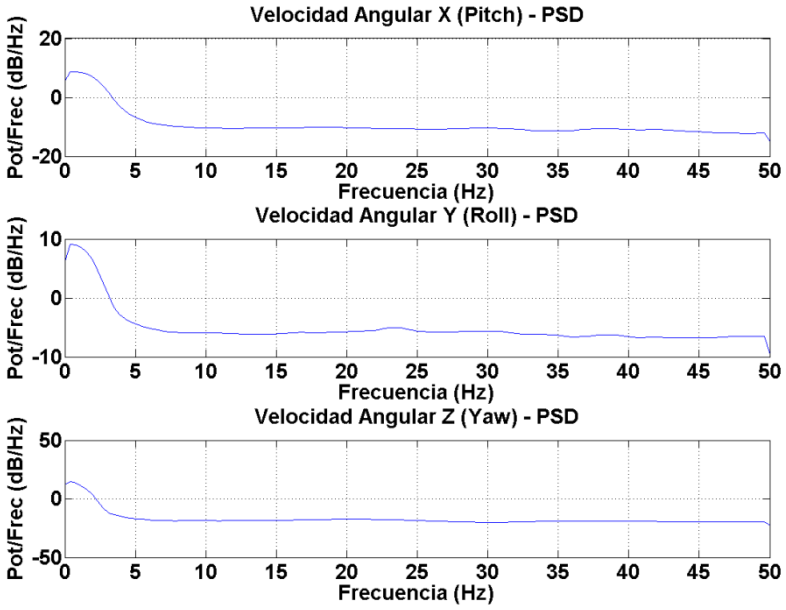

Figura 3.13b - *PSD de la velocidad angular* ($\overrightarrow{\omega_2}$)

Máximo Desplazamiento Lineal

Un aspecto importante de la construcción de una plataforma es cómo de extensos deben ser sus límites de movimiento. Cuanto mayores, más costosa suele ser de construir. Por ello, es interesante analizar los desplazamientos máximos esperados por cada grado de libertad en términos de la frecuencia de los mismos. Es obvio que si no limitamos el espectro de movimiento de las posiciones y orientaciones, éstas serán, en general, irreproducibles. Como

sabemos, los algoritmos de *washout* limitan mediante filtros pasa-alta la cantidad de movimiento que simulan, de forma que a mayor frecuencia de corte, menor cantidad de movimiento se simula.

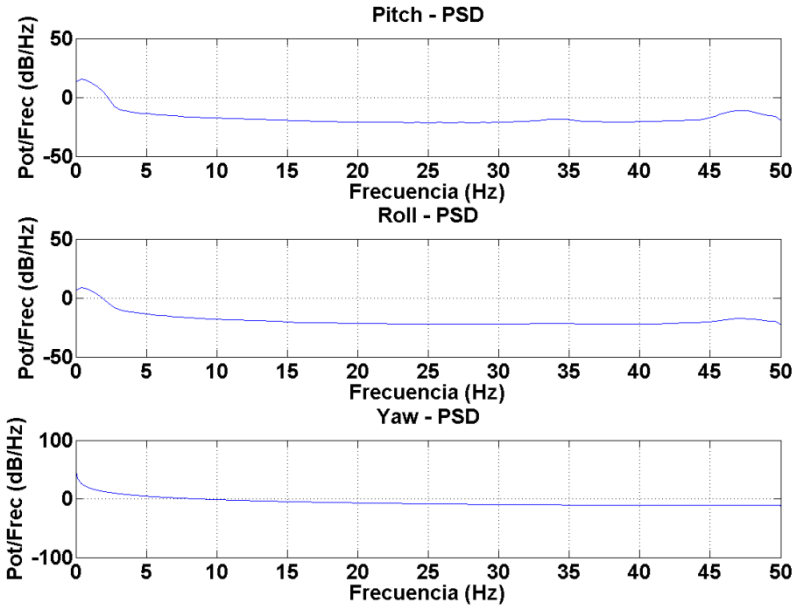

Figura 3.13c - *PSD de pitch, roll y yaw*

En este apartado vamos a calcular la amplitud de pico a pico de los desplazamientos lineales (en las 3 direcciones de movimiento) aplicándoles previamente un filtro pasa-alta de segundo orden (como se realiza en [80]). Con ello podremos dibujar las curvas de amplitudes de pico a pico para diferentes frecuencias de corte (Figura 3.14) y con ello observar cuánta información se perdería con cada valor de frecuencia de corte.

Esto nos permitiría establecer el corte en un valor que fuera compatible con los límites de nuestra plataforma.

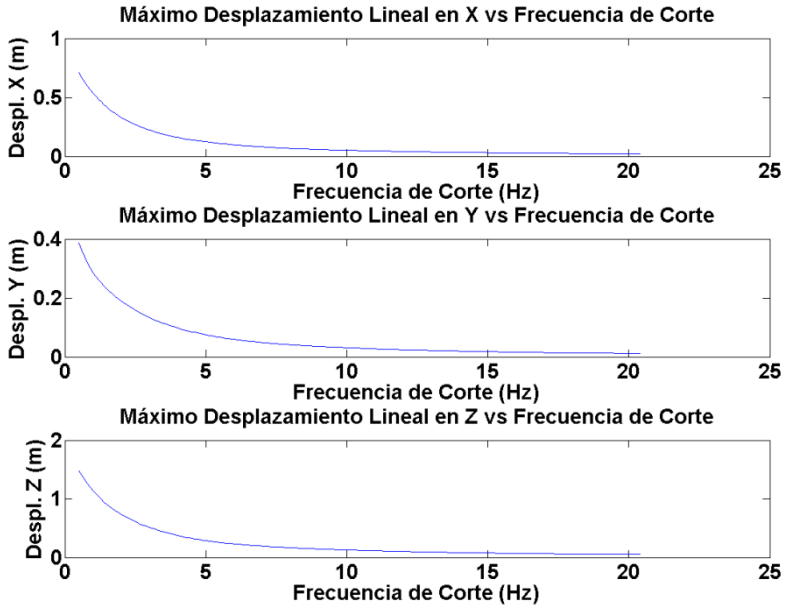

Figura 3.14 - *Desplazamiento lineal máximo en función de la frecuencia de corte*

Para nuestro ejemplo, eliminado todas las frecuencias menores de 3 Hz, la excursión máxima requerida por la plataforma sería de (22.9, 13.7, 52.7) cm (X, Y, Z), lo cual es realizable. Si subiéramos el límite a 5 Hz, la excursión sólo tendría que ser de (12.3, 7.4, 28) cm. A medida que bajásemos la frecuencia por debajo de 3 Hz, los límites se incrementarían de forma exponencial. Por ejemplo, para una frecuencia de corte de 1 Hz

necesitaríamos una excursión máxima de (53.4, 28.5, 113) cm, lo cual ya implicaría una plataforma bastante grande.

Como vemos, la excursión necesaria en el eje Z es mayor siempre que en los ejes X e Y. Esto nos demuestra que es el movimiento lineal más importante en este vehículo y es una consecuencia de que el oleaje y la marea afectan mucho al movimiento vertical.

Máximo Desplazamiento Angular

El mismo análisis se puede realizar para los movimientos angulares (ver Figura 3.15). Como podemos observar, los ángulos de *pitch* (X) y *roll* (Y) no necesitan ser mayores de 20º en ningún caso. Esto quiere decir que el movimiento angular en estos ejes podría ser simulado sin filtrar siempre que encontráramos una plataforma con dicho rango. Los giros en Z (*yaw*) sí que deben ser filtrados porque el *yaw* es de rango ilimitado. Esto es consistente con lo que previamente habíamos concluido. De cualquier forma, el límite de 20º debe ser tomado con precaución, dado que el mar presenta una naturaleza muy cambiante y podrían alcanzarse valores mayores. Además, debido al efecto de *tilt-coordination*, que ya comentamos, puede que se necesite un mayor rango de inclinación en *pitch* y *roll*.

A nivel cuantitativo, con una frecuencia de corte de 3 Hz la excursión necesaria para la plataforma es de (12.18, 10.9, 9.4)°. Con un límite de 5 Hz, la excursión pasa a ser de (10.8, 9.7, 6.6)° y con un límite de 10 Hz, (8.52, 8.53, 3.87)°. A excepción del *yaw*, la modificación de la frecuencia de corte no tiene efectos tan

abruptos como en el caso de la traslación y el límite de 3 Hz es perfectamente realizable sin perder demasiada información. Teniendo en cuenta, además, que las magnitudes físicas (aceleraciones y velocidades angulares normalmente) calculadas por el módulo físico pueden escalarse (reducirse) al ser enviadas al algoritmo MCA, las frecuencias de corte podrían ser más elevadas de las aquí calculadas, por lo que los límites de frecuencia aquí calculados se pueden considerar como cotas inferiores.

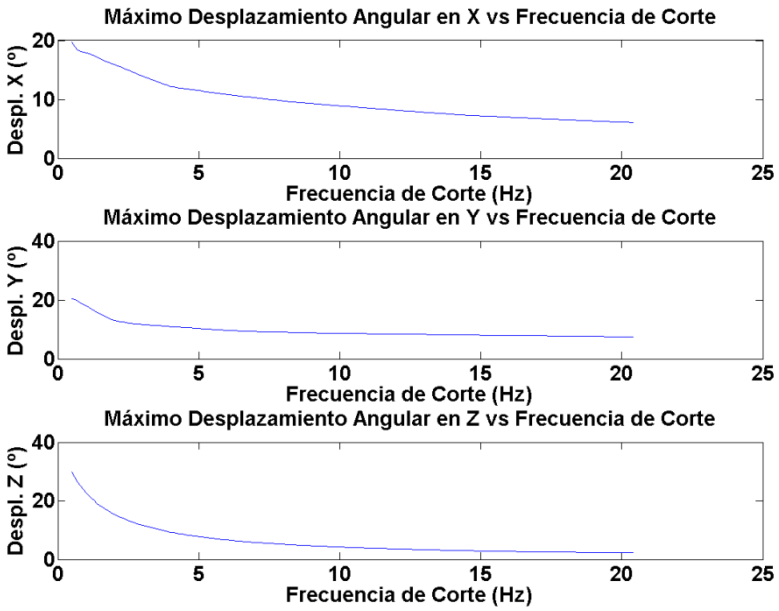

Figura 3.15 - *Desplazamiento angular máximo en función de la frecuencia de corte*

3.1.4 – Conclusiones

Un buen número de conclusiones se pueden extraer de este análisis. Algunas son cualitativas, otras cuantitativas, pero casi todas son generalizables a otro tipo de vehículos (no las conclusiones particulares pero sí el modo de deducirlas).

En el lado cualitativo, y a la vista de las gráficas se puede decir que las 3 claves inerciales más importantes de este vehículo son el *pitch*, el *roll* y el *heave*. El *pitch* es una clave fundamental porque está correlacionado con la velocidad y por tanto con la palanca del acelerador que maneja el piloto. El *roll* es menos significativo a altas velocidades pero cuando el vehículo está parado o girando, es importante. Por último, el *heave* aparece como la clave inercial lineal más representativa. En cuanto a claves no inerciales, la simulación de viento parece también muy importante, aunque como queda fuera de nuestro ámbito de estudio, no profundizaremos más en ello.

En el lado cuantitativo, la conclusión más importante es que las aceleraciones sostenidas (de baja frecuencia) son pequeñas, y por tanto el agua, y no el propulsor es la principal fuente de generación de claves inerciales. Esto es muy indicativo porque significa que debemos ser capaces de reproducir movimientos rápidos y abruptos antes que aceleraciones largas. Por ello, las excursiones de la plataforma no necesitarían ser excesivamente largas, aunque los motores de la plataforma sí deberían ser potentes para poder mover la plataforma rápidamente.

Otra conclusión cuantitativa es que los filtros pasa-alta del algoritmo de *washout* pueden ajustarse con frecuencias relativamente bajas (de 2 a 5 Hz dependiendo de los casos). Dado que las aceleraciones sostenidas no parecen ser muy importantes, con simular movimientos de frecuencia media podríamos obtener una reproducción sencilla y realista.

De forma cuantitativa también podemos ver que las excursiones en el eje Z deben ser mayores que en los otros dos ejes. Esto, unido a que la magnitud de la aceleración Z es mayor que las de los ejes X e Y, hacen de este eje el más importante de los lineales. Los aceleraciones en los ejes X y (en menor medida) Y tienden a ser más suaves.

Por tanto, si tuviéramos que construir la plataforma de movimiento más simple para un simulador de este tipo de vehículo, escogeríamos una plataforma de 1 GdL con movimiento en *pitch*, porque ello nos permitiría simular el movimiento hacia adelante (dado que éste está relacionado con el *pitch*) y también la inclinación. Si tuviéramos que añadir un segundo GdL probablemente elegiríamos una plataforma de 2 GdL con movimiento en *pitch* y *roll*, principalmente porque suelen ser sencillas de construir (y porque con *roll* podemos simular aceleración lineal lateral mediante *tilt-coordination*). Si tuviéramos que añadir otro GdL más, construiríamos una plataforma de 3 GdL con movimiento en *pitch*, *roll* y *heave*. Este diseño, más complejo, incluiría el movimiento en Z, que hemos visto que era importante. Por ello, consideramos que éste sería un buen compromiso entre fidelidad y coste. Si tuviéramos que añadir un cuarto GdL, probablemente elegiríamos el *yaw*, porque es de rango

ilimitado. Aunque podríamos considerar plataformas de 4 ó 5 GdL, en este caso sería casi más sencillo optar por un diseño de tipo Stewart con 6 GdL, porque está mejor documentado y extendido que los diseños de 4 ó 5 GdL. Una plataforma de 6 GdL añadiría mayores posibilidades, pero dado que el *sway* no aportaría demasiada información, el único aporte sería el *yaw* y en menor medida el *surge*, que cómo hemos visto es de carácter vibratorio, muy similar al *pitch*, por lo que muchas veces está acoplado con éste y por tanto el solape de ambos puede hacerlo poco relevante.

Habitualmente, cuantos más GdL más cara es la plataforma, por lo que debemos escoger de forma inteligente dónde ocupar los recursos. En cualquier caso, esta caracterización muestra la información necesaria para guiar este proceso, y la decisión de qué recursos emplear en el diseño queda fuera de nuestro ámbito, ya que depende fundamentalmente del presupuesto del que se disponga y del tipo de simulador que se pretenda realizar.

Más allá de este análisis particular ejemplificado en un bote de rescate, la conclusión que podemos sacar es que podemos disponer de un mecanismo para caracterizar vehículos para simuladores con plataformas de movimiento. Este mecanismo, que podemos aplicar a otro tipo de vehículos, se puede resumir en la secuencia de pasos que enumeramos a continuación.

Primero, elegir un vehículo y unas condiciones de prueba que representen lo mejor posible lo que se desea simular. Después escoger qué medidas queremos tomar, qué sensores las medirán y dónde localizar dichos sensores. Consideramos que aceleración, velocidad, aceleración angular, velocidad angular y orientación

deben ser siempre medidos, ya que representan el comportamiento dinámico del vehículo y son necesarios para la construcción de la plataforma y el ajuste del algoritmo de *washout*. La localización de los sensores depende de lo que se quiera medir pero las recomendaciones comentadas en el punto 3.1.1 se pueden aplicar de forma genérica. El siguiente paso es realizar las pruebas y recoger los datos, asegurándonos siempre de que los sensores disponen de los rangos y frecuencias de actualización suficientes para las mediciones que deseamos realizar. El último paso es analizar los datos. En este paso proponemos 3 aspectos fundamentales: un análisis de rangos (relacionados con los límites de la plataforma), un análisis en el dominio del tiempo (para poder comprender la naturaleza del movimiento y disponer de información de tipo cualitativo) y un análisis en el dominio de la frecuencia (para conocer qué parte del espectro se podría simular). Dentro de este último análisis es muy importante realizar un estudio de los valores máximos de desplazamiento en función de la frecuencia de corte (como vimos en el apartado 3.1.3.6) porque nos permitirá saber hasta qué punto nuestra plataforma es capaz de simular lo que queremos simular y podremos disponer de datos concretos sobre las frecuencias de corte de los filtros y su efecto en la fidelidad de la simulación.

3.2 – Simulación en Tiempo Real de la Dinámica del Vehículo

El conocimiento de la forma e intensidad de los movimientos del vehículo que se pretende simular es necesario para el diseño o elección de la plataforma de movimiento, pero

también puede ser útil para diseñar, o validar, el modelo de simulación dinámica del vehículo. Aunque este apartado no se corresponde estrictamente con la generación de claves gravito-inerciales, ya que sería parte del sistema físico, las entradas de los algoritmos MCA provienen del sistema físico, por lo que es interesante conocer cómo se diseñan, aunque sea someramente.

Por ello, en este apartado vamos a tratar de explicar, de forma muy resumida, cómo podríamos abordar el diseño del modelo dinámico (físico en general) de un simulador. Dado que ya existen muchos modelos de simulación de coches o aviones, vamos a ejemplificarlo, también, con el simulador de bote de rescate rápido.

3.2.1 – Modelo Físico

A la hora de diseñar el módulo físico de un simulador de un vehículo, existen muchas opciones y aproximaciones. Una de ellas es emplear datos experimentales para encontrar una función apropiada que refleje el comportamiento del vehículo. Dentro de este tipo podemos encontrar las técnicas de identificación de sistemas [145], o trabajos con redes neuronales como el de [95]. Estas aproximaciones eran más habituales hace algunos años cuando las bibliotecas de computación de ecuaciones físicas todavía no estaban muy desarrolladas. Sin embargo, dado que este campo ha evolucionado mucho en los últimos años y el cómputo de ecuaciones físicas mediante bibliotecas de cálculo físico es habitual en la actualidad, prácticamente todos los modelos físicos de vehículos actuales se basan en resolver las ecuaciones

diferenciales que describen la dinámica del movimiento de un vehículo. Además, los métodos de identificación de sistemas presentan varios problemas. Primero, al basarse en datos experimentales, sólo serían capaces de representar el vehículo concreto que se probara, por lo que sería complicado extender dicho modelo para hacerlo extensible a vehículos similares con comportamientos ligeramente diferentes. Eso es algo que no ocurre con la mecánica clásica, ya que las ecuaciones dependen de parámetros (masas, inercias) que podemos relacionar con los reales variando el comportamiento del vehículo. En segundo lugar, los modelos de identificación de sistemas hacen suposiciones de independencia lineal que, en muchas ocasiones, no se corresponden con la realidad. Y por último, en este caso concreto, la naturaleza caótica del agua hace muy complicado modelar el sistema como una caja negra con entradas y salidas del que se puedan tomar los suficientes datos reales como para elaborar un sistema fiable.

En nuestro caso, dado que disponemos de datos reales de pruebas con un vehículo del tipo que queremos modelar, podemos emplear esta información para validar el modelo, aunque, por las razones comentadas, no empleemos estos datos para derivar el propio modelo. Por tanto, lo que proponemos en este apartado es derivar las ecuaciones del movimiento de un bote de rescate mediante el empleo de la dinámica clásica, para posteriormente validar la corrección del modelo con datos experimentales.

La dinámica clásica nos dice que las fuerzas que gobiernan el comportamiento de una embarcación son el peso, la flotación, la fricción con el aire, la fricción con el agua y la fuerza de propulsión

(o bien velas, o remos o propulsores mecánicos) [146]. Algunas de estas fuerzas, como la fricción y la propulsión son ciertamente complejas por lo que su simulación realista requeriría una elevada cantidad de recursos de computación, así que deberemos hacer simplificaciones. Evidentemente, una excesiva simplificación puede llevar a una mala simulación, por lo que debemos alcanzar un cierto compromiso entre realismo y necesidades computacionales. Por ello, haremos las siguientes suposiciones:

1- El bote será un sólido rígido (aunque algunos tienen partes inflables, las tomaremos como rígidas).
2- Se considerará que la forma de la superficie del mar sigue una cierta función analítica determinista, por compleja que ésta sea.
3- La superficie del mar no es parte de nuestro modelo de bote de rescate rápido, sino una de sus entradas.
4- El movimiento del mar influye en el movimiento del bote, y el movimiento del bote puede influir en el movimiento del mar, pero esta última influencia es responsabilidad del modelo de mar.
5- El viento y las corrientes se consideran representados mediante un vector con dirección y magnitud.
6- La fuerza de fricción se calculará como una fuerza dependiente de la forma.
7- Sólo se consideran propulsores basados en motores con hélice.
8- Las turbulencias de la hélice se ignoran, al menos de forma explícita.

Por tanto, representaremos el bote como un único sólido rígido, pero bajo la influencia de muchas fuerzas. La forma de la embarcación y su composición interna afectan tanto a la flotabilidad como a las fricciones (tanto a la aerodinámica como a la hidrodinámica). Por ello, proponemos simplificar el cálculo representando el bote como un conjunto finito de pequeños cubos de igual tamaño y diferente masa (ver Figura 3.16). Esto permitirá además, representar de forma fiel las diferencias de densidad que ocurren en este tipo de embarcaciones, donde una parte del vehículo puede ser rígida y pesada y otra parte puede ser simplemente aire a presión, cuya densidad es menor. La implementación exacta de las diferentes fuerzas se explicará posteriormente.

El modelo de cubos representa una forma de realizar la integración a lo largo de todo el volumen del vehículo de las fuerzas que gobiernan el comportamiento del bote. La discretización hace que cada cubo represente un diferencial de volumen y que sea sencillo calcular en ellos las fuerzas de fricción, dado que al tener forma cúbica, existen sólo 3 direcciones en el espacio para realizar dicho cálculo, siendo además las superficies de fricción áreas cuadradas cuyo cálculo es trivial. La flotación y el peso también son sencillos de computar.

Evidentemente, a menor tamaño de cubo, se obtiene mayor precisión, pero también se requieren mayores capacidades computacionales. El Apéndice B.1 muestra cómo se puede calcular cuál es el número óptimo de cubos.

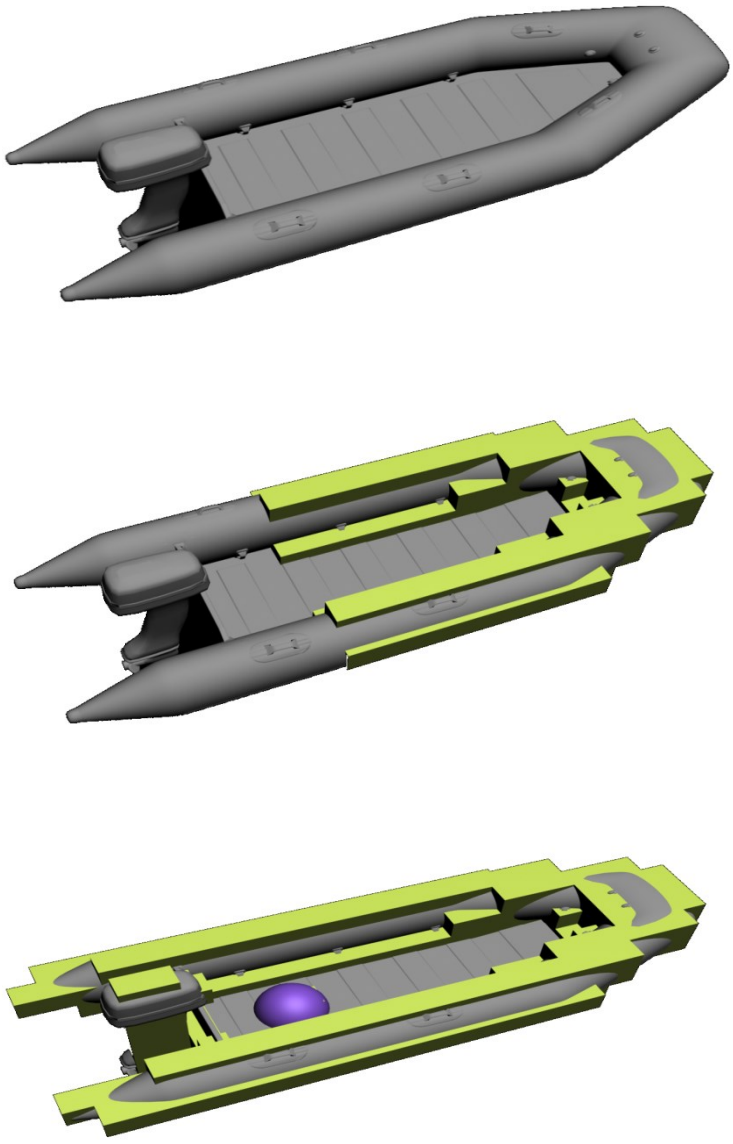

Figura 3.16 - *Proceso de construcción del modelo físico de bote de rescate*

3.2.1.1 - Modelo de Mar

Aunque nuestro interés principal es el movimiento del vehículo, es evidente que la interacción mar-embarcación es una de las partes más importantes de dicho movimiento. Sin embargo, nuestro objetivo es modelizar cómo reacciona el vehículo a su entorno y no modelizar el entorno en sí. Nuestro modelo realmente sólo necesita conocer la superficie del mar (la altura de sus puntos) en cualquier punto de la misma, para calcular la flotación y las fuerzas de fricción. Cómo se realice dicha simulación es algo que dejamos para otros estudios. De hecho, la simulación realista de océanos y mares es una tarea ciertamente compleja y objeto de muchos artículos y trabajos. Por ello, consideraremos el mar como una entrada más de nuestro modelo y tanto la simulación del mismo como la influencia del vehículo sobre el mar se dejan como parte de la responsabilidad del modelo de mar. Es por ello, que utilizaremos un modelo de mar basado en superficie, similar al propuesto por [147]. En cualquier caso, el uso de dicho modelo físico de mar es independiente de nuestro modelo de vehículo, siempre y cuando dicho modelo de mar sea capaz de proporcionarnos la altura de cada uno de los puntos del mar en todo momento. Esto, además, proporciona una independencia algorítmica que ayuda a reducir el acoplamiento *software* entre las distintas partes de la simulación física.

3.2.1.2 - Modelo de Vehículo

En este apartado vamos a describir cómo calcular las fuerzas que hemos considerado en nuestro modelo.

Peso

El peso es una fuerza vertical hacia abajo que se aplica sobre el centro de masas de un objeto. Se podría calcular una única fuerza peso para todo el vehículo, pero, dado que podemos tener diferentes materiales con diferentes densidades, y podríamos querer simular también pérdidas de presión de aire (en botes inflables por ejemplo), calcularemos la fuerza peso en cada uno de los cubos de la embarcación como:

$$\vec{F}_w = m \cdot \vec{g} \qquad (3.9)$$

· $\vec{g}$: vector aceleración de la gravedad (m/s^2).
· m: masa del cubo (kg).
· $\vec{F}_w$: fuerza peso (N).

El único parámetro ajustable aquí es m (la masa del cubo) que depende del diseño del vehículo y del tamaño de cubo escogido.

Flotación

Para cada cubo se calcula la fuerza de flotación vertical (3.10). Dicha fuerza se aplicará en el centro de flotación del cubo. Como el centro de flotación de una embarcación cambia a medida que éste desplaza agua, un cálculo distribuido por cubos permite una mejor aproximación del volumen de agua desplazado por la embarcación y no necesitamos calcular el centro de flotación global del vehículo.

$$\vec{F}_b = -\vec{g} \cdot V \cdot \rho \qquad (3.10)$$

- $\vec{g}$: vector de aceleración de la gravedad (m/s^2).
- V: volumen del cubo sumergido (l), calculado numéricamente por la intersección de la superficie del agua con el cubo.
- ρ: densidad del agua (kg/m^3), que es función de la temperatura, salinidad, y presión del agua, pero que aproximaremos como una constante conocida.
- $\vec{F}_b$: fuerza de flotación resultante (N), ejercida hacia arriba en el centroide de la parte sumergida del cubo, que asumimos es el centro de flotación.

No existen parámetros en esta ecuación, dado que V es variable y ρ lo podemos considerar como una constante, que en cualquier caso no depende del bote.

Fuerza de Fricción Aerodinámica

Tanto el viento como la resistencia del aire al movimiento de un cuerpo son factores importantes en el movimiento de un bote motorizado. Aunque un cálculo exacto es complejo, la subdivisión en cubos nos permitirá hacer una simulación bastante precisa de los efectos de ambos fenómenos, que en realidad son parte del mismo efecto: la fricción aerodinámica. Como un cubo tiene 6 caras y cada una de esas 6 caras opone resistencia al movimiento por fricción aerodinámica, calcularemos 6 fricciones por cada cubo. Como la oclusión de cubos puede hacer que muchos de ellos no reciban aire por estar rodeados de otros, el cálculo será descartado para todas aquellas caras que estén ocultas por otros

cubos. Este cálculo de oclusión se puede hacer en tiempo de carga por lo que no supone un problema para la simulación en tiempo real. El efecto del viento no se calcula de forma separada porque, en realidad, es parte del mismo efecto. La velocidad relativa entre la cara del cubo y el vector viento nos dará el viento aparente que definirá cuál es la velocidad de las partículas de aire que colisionan con cada cara de cada cubo. Nuestro modelo también permite detectar oclusiones de viento por culpa de otros barcos. Cuando se detecta una oclusión de viento por la presencia de otra embarcación (a una cierta distancia menor que un umbral), el viento en dicha cara se pondrá a 0 antes de realizar el cálculo de la velocidad aparente. Por tanto, en dicha cara no se elimina la resistencia aerodinámica global sino sólo el efecto del viento. Las ecuaciones (3.11) y (3.12) describen este proceso:

$$\vec{v}_{wa} = \vec{v}_w - \vec{v}_c \qquad (3.11)$$

$$\vec{F}_{ad} = \frac{1}{2} \cdot \rho \cdot C_{ad} \cdot A \cdot \vec{v}_{wa} \cdot |\vec{v}_{wa}| \qquad (3.12)$$

- $\vec{v}_w$: vector que representa la velocidad del fluido (viento) (m/s) en coordenadas del mundo.
- $\vec{v}_c$: vector que representa la velocidad del cubo (m/s) en coordenadas del mundo.
- $\vec{v}_{wa}$: viento aparente (m/s) en coordenadas del mundo.
- ρ: densidad del aire (kg/m^3).
- C_{ad}: coeficiente de fricción aerodinámica (sin dimensiones).
- A: área (m^2) del cubo expuesta, calculada numéricamente por la intersección de la superficie del agua con el cubo.
- $\vec{F}_{ad}$: fuerza de fricción aerodinámica resultante (N).

El único parámetro ajustable en estas ecuaciones es C_{ad} que depende del fluido, de la forma del objeto (aunque ésta en nuestro modelo no cambia porque es siempre un cubo) y del material. Normalmente es un valor obtenido de forma empírica.

Fuerza de Fricción Hidrodinámica

La fuerza de fricción hidrodinámica y las corrientes submarinas son otro factor a tener en cuenta. Aunque el agua y el aire se comportan de manera diferente, porque el agua es un fluido incompresible, y el aire no, podemos suponer que, al nivel que nosotros queremos simular la fricción, su comportamiento es similar. Por ello, el cálculo de la fricción hidrodinámica es totalmente análogo al anterior. Todo es similar, con la única diferencia de que la densidad del agua es unas 1000 veces mayor que la del aire. También calculamos oclusiones. Las ecuaciones (3.13) y (3.14) describen la formulación de este efecto:

$$\vec{v}_{sa} = \vec{v}_s - \vec{v}_c \qquad (3.13)$$

$$\vec{F}_{wd} = \frac{1}{2} \cdot \rho \cdot C_{wd} \cdot A \cdot \vec{v}_{sa} \cdot |\vec{v}_{sa}| \qquad (3.14)$$

- $\vec{v}_s$: vector que representa la velocidad del fluido (corrientes de agua) (m/s) en coordenadas del mundo.
- $\vec{v}_c$: vector que representa la velocidad del cubo (m/s) en coordenadas del mundo.
- $\vec{v}_{sa}$: corriente aparente (m/s) en coordenadas del mundo.
- ρ: densidad del agua (kg/m^3).
- C_{wd}: coeficiente de fricción hidrodinámica (sin dimensiones).

- A: área (m^2) del cubo expuesta, calculada numéricamente por la intersección de la superficie del agua con el cubo.
- $\vec{F}_{wd}$: fuerza de fricción hidrodinámica resultante (N).

Al igual que antes, el único parámetro aquí es C_{wd} que no es necesariamente igual que C_{ad} al ser la fricción una función del material y el fluido.

Fuerza de Propulsión

Dado que el objetivo de cualquier vehículo es moverse, necesitamos modelar la fuerza que gobierna el control del bote. Todas las fuerzas anteriores eran fuerzas de la naturaleza que no podemos controlar. Ahora pasaremos a intentar describir cómo es la fuerza que nos permite dirigir el vehículo a voluntad. Dependiendo del vehículo a simular, el dispositivo de propulsión puede ser diferente. En nuestro caso, vamos a suponer que se trata de botes de rescate propulsados por una hélice alimentada por un motor. Este es el caso más habitual.

Aunque la interacción motor-hélice se puede estudiar desde un punto de vista Newtoniano, la interacción hélice-agua y el funcionamiento interno del motor son asuntos muy complejos que normalmente se simulan de modo heurístico.

En nuestro modelo, proponemos modelar el motor como un agente que genera torque sobre la hélice. La cantidad de torque dependerá de la palanca de aceleración y de la velocidad angular a la que gire el motor (3.15). Esta es una característica intrínseca de los motores de combustión [146], y la función exacta depende de

cada motor, por lo que será un parámetro configurable del modelo de vehículo. Este torque intenta mover el rotor del motor acoplado solidariamente a la hélice. A medida que se mueven, el rotor (y la hélice) encuentran una resistencia a dicho movimiento (3.16) que modelamos como 3 términos: una fricción constante, un término que depende de la velocidad angular y un término que depende de la aceleración angular del motor. Cada término se corresponde con un parámetro que controla la cantidad de resistencia que se le aplica al rotor: k_1, k_2 y k_3. El resultado es el torque neto. La velocidad angular del motor se calcula a partir de la aceleración angular (3.17) y ésta viene de la ecuación (3.18). Ambas son ecuaciones de la mecánica clásica. El único parámetro en las ecuaciones (3.17) y (3.18) es I, la matriz de inercia que se podría aproximar por una constante.

La orientación de la hélice se controla con el timón. Aunque el control se puede aproximar por una función lineal, lo implementamos como una función general (3.19) que se deja como parámetro. Después, el motor transmite el movimiento del rotor al movimiento de la hélice, la hélice mueve el agua, y por la 3ª ley de Newton, el agua mueve al bote en dirección opuesta (3.22). Cada revolución completa del motor debe corresponder con una revolución de la hélice (dado que están acoplados), pero como puede haber un engranaje, añadimos un parámetro de proporcionalidad que llamaremos ratio diferencial (C_r) (3.20). El movimiento de la hélice genera un desplazamiento de agua que acaba generando un movimiento de propulsión. Idealmente, un giro completo de la hélice debe proporcionar la misma fuerza de forma proporcional a la forma, tamaño y ángulo de ataque de la hélice [148] [149]. A esta constante de proporcionalidad le

llamaremos coeficiente de avance (C_a). Sin embargo, tanto las turbulencias como el deslizamiento del agua por la superficie de la hélice (*slip effect*) modifican la eficiencia de esta operación, y no toda el agua desplazada genera movimiento. Para tener en cuenta este efecto, introducimos un parámetro llamado eficiencia de la hélice η, que modelaremos como una función de la velocidad angular dado que las turbulencias y otros efectos hidrodinámicos dependen de la velocidad de la hélice (3.21). Esto se conoce a veces como *slip ratio* [149]. La fuerza resultante $\vec{F}_e$ calculada en (3.22) es la fuerza final de propulsión. El signo es negativo debido a que es opuesta a la dirección de la hélice. Esta fuerza sólo se calcula en un único cubo marcado como cubo de hélice.

$$\vec{\tau}_e = f(\vec{\omega}_e, T) \quad (3.15)$$

$$\vec{\tau}_n = \vec{\tau}_e - k_1 - k_2 \cdot \vec{\omega}_e - k_3 \cdot \vec{\alpha}_e \quad (3.16)$$

$$\vec{\omega}_e = \int \vec{\alpha}_e \cdot dt \quad (3.17)$$

$$\vec{\alpha}_e = \vec{\tau}_n \cdot I^{-1} \quad (3.18)$$

$$\vec{d} = f(\partial) \quad (3.19)$$

$$\vec{\omega}_h = \vec{\omega}_e \cdot C_r \quad (3.20)$$

$$\eta = f(|\vec{\omega}_h|) \quad (3.21)$$

$$\vec{F}_e = -\eta \cdot |\vec{\omega}_h| \cdot C_a \cdot \vec{d} \quad (3.22)$$

- $\vec{\alpha}_e$: aceleración angular del motor (rd/s^2).
- $\vec{\omega}_e$: velocidad angular del motor (rd/s).
- T: valor del acelerador (sin dimensiones y en el rango [0..1]).
- $\vec{\tau}_e$: torque del motor (N·m).
- k_1: término de fricción constante del motor (N·m).
- k_2: término de fricción dependiente de la velocidad angular del motor (kg·m^2/s).

- k_3: término de fricción dependiente de la aceleración angular del motor (kg·m^2).
- $\vec{\tau}_n$: torque neto del motor (N·m).
- I: tensor de inercia del giro del motor (kg·m^2).
- $\vec{\omega}_h$: velocidad angular de la hélice (rd/s).
- η: eficiencia de la hélice (sin dimensiones).
- ∂: ángulo de giro de la hélice (rd).
- $\vec{d}$: vector de dirección de la hélice (sin dimensiones).
- C_r: ratio diferencial entre la hélice y el motor (sin dimensiones).
- C_a: coeficiente de avance de la hélice (kg/s).
- $\vec{F}_e$: fuerza de propulsión final resultante (N).

Fuerza Resultante

La fuerza resultante en cada cubo es la suma de todas las anteriores fuerzas (3.23). La aplicación de todas estas fuerzas sobre todos los cubos genera un torque y una fuerza lineal resultante sobre el bote que se puede transformar sucesivamente en aceleración lineal y aceleración angular, para luego obtener velocidad lineal y angular, para finalmente obtener la posición y orientación del vehículo. Dichas ecuaciones diferenciales se pueden resolver mediante una biblioteca de cálculo físico como Nvidia PhysX sin demasiadas complicaciones, siempre y cuando se mantengan las restricciones temporales adecuadas para mantener la simulación bajo los parámetros de tiempo real. En cuanto a los parámetros del modelo físico, un posible ajuste de los mismos se puede consultar en el Apéndice B.2.

$$\vec{F} = \vec{F}_w + \vec{F}_b + \vec{F}_{ad} + \vec{F}_{wd} + \vec{F}_e \qquad (3.23)$$

En cuanto al módulo inercial, la velocidad angular y la aceleración lineal (que se puede convertir fácilmente en fuerza específica) calculadas en este apartado se pueden tomar como entradas para un algoritmo MCA sobre una plataforma de movimiento y así simular las claves inerciales proporcionadas por el manejo de este tipo de embarcaciones.

3.3 – Conclusiones

En este capítulo hemos visto cómo podemos caracterizar el movimiento de un vehículo cualquiera, ejemplificado en el caso de un bote de rescate. De este análisis hemos podido deducir las características más importantes que se observan en el movimiento de este tipo de vehículos, para concluir que necesitamos, al menos, una plataforma de 3 GdL para poder acometer la tarea de reproducción inercial con cierto éxito. Naturalmente, esta conclusión no es una conclusión que se pueda tomar estrictamente, porque no hay una línea roja que marque claramente la diferencia entre una buena simulación y una mala simulación, y todo depende del grado de realismo que deseemos y del presupuesto del que dispongamos. Sin embargo, sí que parece claro que hemos identificado cuál es la mejor manera de invertir recursos a la hora de construir el módulo inercial de un simulador de vehículos, en este caso un bote de rescate rápido. Creemos que este mecanismo, tal y como lo hemos presentado, es extrapolable a otro tipo de vehículos no aéreos de mediano tamaño como motocicletas, embarcaciones de remo, bicicletas, coches, etc.

Después, hemos aprovechado que disponíamos de datos reales del movimiento del bote para proponer un modelo físico del mismo. Aunque dicho modelo no es parte propiamente dicha del módulo inercial, que es lo que estamos estudiando en este trabajo, consideramos que es importante realizar aportaciones sobre cómo se podría diseñar un modelo físico de algún vehículo sobre el que la literatura no ofrezca ninguno. Este es el caso del bote de rescate rápido. Dado que el autor no pudo encontrar ningún modelo físico para probar este tipo de vehículos, decidimos elaborar y validar uno nosotros mismos. El resultado, avalado por las pruebas de validación descritas en el Apéndice B.3, es bastante satisfactorio, está publicado en [21] y se ha empleado en simuladores reales de formación en tareas de emergencia y evacuación marina con pilotos reales de botes de rescate rápido.

Capítulo 4

Análisis del Dispositivo de Generación de Claves Gravito-inerciales

Una vez se tienen claras las necesidades de tipo gravito-inercial, el segundo paso importante es la elección de la plataforma de movimiento. Por desgracia, muchas veces, no se puede elegir el dispositivo, ya que factores como el precio, la disponibilidad o el espacio disponible, limitan bastante esta decisión. En ocasiones el dispositivo fue comprado con anterioridad y conviene amortizarlo. Otras veces, simplemente no hay presupuesto para uno nuevo. Por tanto, es importante poder sacar el máximo partido al dispositivo del que dispongamos, sea cual sea. Es por ello que, en este capítulo, vamos a proponer una serie de pasos para analizar una plataforma de movimiento para la generación de claves gravito-inerciales. Dado que el capítulo anterior lo ejemplificamos con un bote de rescate, y llegamos a la conclusión de que una plataforma de 3 GdL sería adecuada para nuestra aplicación, vamos a ejemplificar este capítulo con un dispositivo de ese tipo, que construimos precisamente para un simulador de este tipo.

4.1 – Introducción

Cuando se desarrollan simuladores de vehículos, el uso de sistemas de movimiento debe añadir claves gravito-inerciales al

simulador. Esto es especialmente necesario en simuladores de habilidades, donde estos dispositivos añaden el realismo necesario para que la transferencia de habilidades se realice en condiciones lo más parecidas posibles a la realidad. La mejor prueba de esta necesidad es que la presencia de generadores de claves gravito-inerciales es obligatoria para certificar este tipo de simuladores mediante sistemas de certificación internacionalmente aceptados, como el de DNV [36].

Esta generación de movimiento se produce casi siempre en forma de manipulador electromecánico (MEM). Es por ello, que el conocimiento de las particularidades del manipulador, y en especial del espacio efectivo de movimiento del mismo, es esencial para diseñar experiencias inmersivas con estos aparatos. A este respecto existen muchos trabajos en la literatura de la materia que analizan diversos tipos de dispositivos desde muchos puntos de vista. Sin embargo, dado que los MEM se emplean para múltiples usos y el uso en simuladores no es el principal, existe un pequeño vacío en cuanto al análisis de MEM empleados para la generación en tiempo real de movimientos para simulación. Esto es especialmente cierto en lo que respecta a manipuladores comerciales porque los fabricantes [33], [132] suelen proporcionar muy poca información. Aunque a veces proporcionan datos genéricos de los límites de cada GdL por separado y de las velocidades/aceleraciones de la plataforma, dicha información suele ser escasa, sesgada y poco útil, por lo que es complicado analizar la idoneidad de cada dispositivo para una determinada aplicación.

Además, dado que el coste de un dispositivo de este tipo puede ser una parte muy importante del coste total de un simulador, la elección del manipulador apropiado con el menor coste es algo muy importante, por lo que se necesita disponer de la información adecuada para tomar esta decisión. A este respecto, cuando se emplea un MEM para simulación, una de las características más importantes es la cantidad de desplazamiento lineal y angular que el dispositivo puede proporcionar para cada uno de los GdL. La razón de esta importancia radica en que, en general, el grado de realismo de la simulación aumenta con el rango de dichos desplazamientos. Este conjunto de rangos se conoce en la literatura como *espacio de trabajo* (*workspace*). El espacio de trabajo se mide normalmente en términos del espacio cartesiano que el manipulador es capaz de alcanzar. Aunque esto es útil para la mayoría de aplicaciones, para simulación es mucho más útil analizar este espacio en términos de GdL, en lugar de en términos absolutos de posición cartesiana, dado que lo que importa no es a qué posición del espacio llega la plataforma sino qué rango de desplazamiento lineal o angular permite por cada GdL, y en conjunto. Es por ello que emplearemos el término *espacio de grados de libertad* o *espacio de soluciones*. Este último término se debe a que dicho espacio se obtiene a través de las soluciones de la cinemática inversa, como veremos posteriormente. El espacio de soluciones será, por tanto, el conjunto de combinaciones alcanzables de todos los GdL del manipulador.

Los MEM pueden clasificarse, básicamente, en manipuladores serie y manipuladores paralelos [41]. En los últimos años, las ventajas de los manipuladores paralelos en términos de velocidad, precisión y capacidad de carga, han hecho que su uso se

extienda ampliamente no sólo para simulación sino para otras muchas aplicaciones. Aunque existen simuladores con manipuladores serie la mayoría emplean manipuladores paralelos. Sin embargo, los manipuladores paralelos (MP) tienen algunas desventajas importantes con respecto a los serie para su uso en simulación. En general, la formulación es más compleja y sobre todo, su espacio de GdL suele ser bastante reducido.

Los MP más empleados son los de 6 GdL. Muchos de ellos se basan en la arquitectura Gough-Stewart [37] que se ha convertido en un estándar *de facto*. Varios investigadores han estudiado este manipulador, analizándolo cinemática y dinámicamente [150], presentando algoritmos para solucionar su cinemática inversa [77], analizando su espacio de trabajo [151], o incluso optimizando el diseño del mismo.

Sin embargo, el uso de 6 GdL no es siempre necesario, y sobre todo, no siempre es asequible ni rentable. Por ello, aproximaciones recientes [152] ponen el foco en estrategias basadas en el uso de manipuladores con menos GdL, que mantengan las ventajas de los MP y al mismo tiempo supongan una reducción de costes, de tiempo de fabricación o incluso de espacio. En esta línea, vamos a analizar en este capítulo uno de los manipuladores comerciales de bajo coste más utilizados. Dicho manipulador es un MP de 3 GdL de tipo T1R2 (1 GdL traslacional, 2 GdL rotacionales). Dado que es un dispositivo de bajo coste adecuado para el simulador de bote de rescate y no existe, al menos que sepamos, ningún estudio similar sobre el mismo, vamos a ejemplificar sobre este dispositivo un

procedimiento para poder analizar un MEM para su uso en simulación.

Para ello, primero describiremos el manipulador. Después analizaremos la cinemática inversa del mismo y la validaremos. Más adelante realizaremos una caracterización cinemática analizando el espacio de GdL. Posteriormente propondremos cambios en dicho espacio de soluciones para un mejor uso en simuladores. Finalmente, analizaremos algunas características dinámicas del dispositivo. Aunque todos los apartados son inevitablemente particulares de este MEM, el procedimiento establecido y sobre todo el análisis del espacio de soluciones y el análisis dinámico son extrapolables a cualquier otro manipulador similar.

4.2 – Descripción del Manipulador

4.2.1 – Diseño del Manipulador

La plataforma de movimiento que vamos a analizar es un MP de tipo T1R2 con 3 tipos de movimiento: *heave* (traslación en el eje Z), *pitch* (rotación sobre el eje X) y *roll* (rotación sobre el eje Y). Se compone de 3 motores rotacionales, 3 bielas, 3 pistones, 1 eje estriado vertical y 1 base móvil, como se puede observar en la Figura 4.1. En dicha figura, los elementos blancos representan elementos móviles, y el resto son estáticos. Los nombres biela y pistón se utilizan en lugar de términos más antropomórficos como brazo o pierna, como una analogía a un motor de combustión, porque su funcionamiento es similar a un sistema pistón-biela. Como ya mencionamos, esta arquitectura es típica en simuladores

comerciales de bajo costo de 3 GdL [33], y es especialmente apropiada para el caso de simulador de bote de rescate explicado en el capítulo anterior.

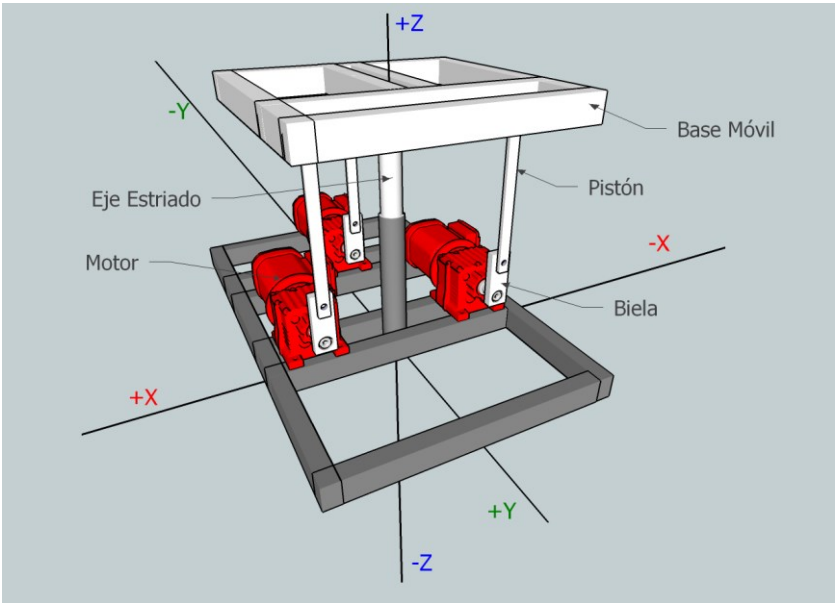

Figura 4.1 - *El manipulador paralelo de tipo T1R2 analizado*

Todos los elementos del manipulador están unidos entre sí por uniones de forma paralela (ver Figura 4.2), por eso es un MP. Cada eje de cada motor está unido solidariamente a una biela, por lo que la biela transmite el movimiento angular del eje del motor, y se comporta respecto a la carcasa del motor como una unión bisagra T0R1. Por tanto, cada motor se puede controlar individualmente para que cada biela pueda formar un ángulo diferente con el plano horizontal. Los puntos de conexión de los 3 motores con las 3 bielas están dispuestos en forma triangular.

Como los ejes de los motores no se desplazan (sólo giran), este triángulo es fijo. Nombraremos a estos puntos $\overrightarrow{L_1}$, $\overrightarrow{L_2}$, $\overrightarrow{L_3}$ y llamaremos al triángulo que los une, *triángulo inferior* (ver Figuras 4.2 y 4.3).

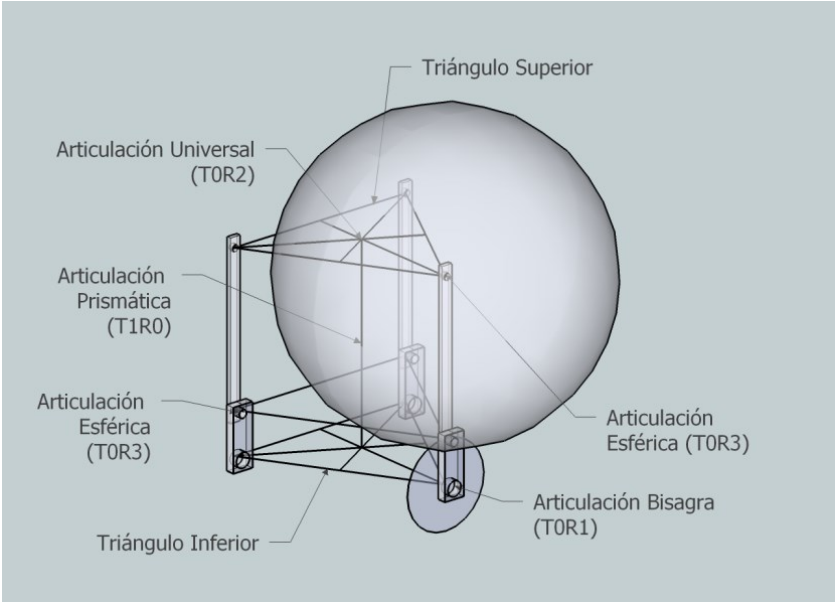

Figura 4.2 - *Uniones y elementos móviles de la plataforma*

Como las bielas, de longitud *r*, sólo pueden moverse en el plano vertical XZ (debido a la articulación T0R1), los extremos de las 3 bielas forman otro triángulo. A los vértices de este triángulo les llamaremos $\overrightarrow{X_1}$, $\overrightarrow{X_2}$, $\overrightarrow{X_3}$. Este triángulo no es estático, puede cambiar su posición e incluso su forma, según la posición de las bielas.

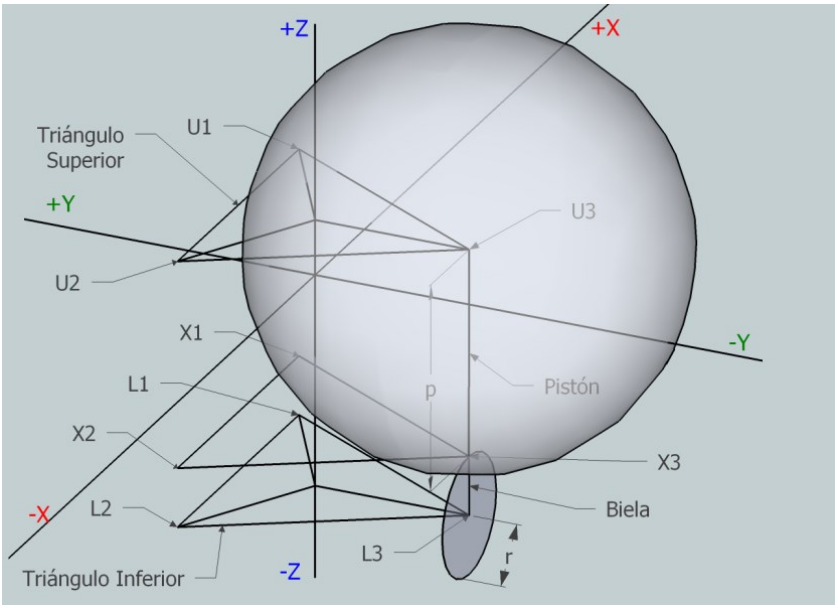

Figura 4.3 - *Modelo geométrico de la plataforma*

Cada biela está unida a un pistón, de longitud *p*, por medio de una articulación esférica T0R3 en $\overrightarrow{X_1}$, $\overrightarrow{X_2}$, $\overrightarrow{X_3}$. Del mismo modo, cada pistón está unido a la base móvil por otra articulación esférica T0R3 en $\overrightarrow{U_1}$, $\overrightarrow{U_2}$, $\overrightarrow{U_3}$. Estos 3 puntos forman otro triángulo, que llamaremos *triángulo superior* (ver Figuras 4.2 y 4.3). Como la base móvil es rígida, este triángulo puede moverse y girar, pero no puede nunca cambiar su forma. La posición y orientación de este triángulo superior definirá el *heave*, *pitch* y *roll* del manipulador. La esfera en las Figuras 4.2 y 4.3 representa las posibles posiciones del pistón (cuando gira alrededor de $\overrightarrow{U_3}$), y la circunferencia vertical XZ, representa las posiciones posibles de la biela ($\overrightarrow{X_3}$) cuando gira

alrededor de $\overrightarrow{L_3}$. Aunque no se dibuja por razones de claridad, lo mismo puede aplicarse a $\overrightarrow{U_1}$, $\overrightarrow{U_2}$, $\overrightarrow{L_1}$, $\overrightarrow{L_2}$, $\overrightarrow{X_1}$, $\overrightarrow{X_2}$.

Como las articulaciones de los pistones son uniones esféricas con rotación completa, necesitamos más articulaciones para evitar que la base móvil rote de forma incontrolada alrededor del eje Z (rotación *yaw*) y se desplace a lo largo de los ejes X e Y (*sway* y *surge*). Esto se logra mediante la adición de una articulación de tipo universal T0R2 (también conocida como junta Cardan) colocada en el baricentro del triángulo superior y mediante una unión T1R0 vertical, que se suele realizar mediante un eje estriado unido a la unión de tipo universal. La elección del baricentro no es arbitraria. Este punto es el centro de masas del triángulo, por lo que es el lugar donde se distribuye equitativamente el peso de la base móvil. La unión universal permite a la base móvil rotaciones de tipo *pitch* y *roll*. El eje estriado vertical actúa como una articulación prismática, permitiendo el movimiento de la base a lo largo del eje Z, pero evitando cualquier *yaw*, *sway* o *surge*. Estas dos articulaciones generalmente se construyen por separado, pero pueden tratarse conceptual y matemáticamente como una única unión T1R2 (Cardan prismática).

El eje vertical también puede equiparse adicionalmente con un muelle. El muelle puede ser útil a nivel dinámico, cuando los motores no son lo suficientemente potentes, para facilitar el proceso de elevación de la base móvil. Sin embargo, a nivel cinemático puede ser ignorado porque ayuda al movimiento (lo acelera o lo frena), pero ni lo modifica ni lo restringe en extensión.

4.2.2 – Cinemática del Manipulador

4.2.2.1 – Especificación y Parámetros

El propósito de la cinemática inversa (CI) es calcular los ángulos (en el caso de un manipulador con motores rotacionales) necesarios para los motores de la plataforma, de modo que ésta alcance una determinada pose (conjunto de GdL) deseada en el extremo de la cadena cinemática (en la base móvil en plataformas de movimiento). La cinemática directa (CD) representa la situación opuesta, cuando decidimos un conjunto de ángulos para los motores y necesitamos calcular la pose que puede alcanzar el manipulador.

En un MP, la CI es generalmente más fácil de resolver que la CD. En manipuladores serie, la situación suele ser la contraria. Afortunadamente, para emplear un MP para la generación de claves gravito-inerciales, sólo necesitamos resolver la CI. La CD no es necesaria para la simulación de movimiento porque el uso natural de una plataforma de movimiento dentro de un simulador es el siguiente: un MCA decide un nuevo conjunto de GdL para la plataforma de movimiento. Dados esos GdL, calculamos los ángulos deseados para los motores con el fin de enviar éstos a los controladores de los motores de modo que éstos, finalmente, muevan la plataforma.

Sin embargo, a veces no es posible alcanzar un cierto conjunto de GdL, y la CI debe detectarlo. Por ello, la especificación en forma de entradas/salidas del MP T1R2 será la siguiente:

Entradas del sistema:

· *heave* deseado
· *pitch* deseado
· *roll* deseado

Salidas del sistema:

· factible: sí/no
· α_k: ángulo del motor k (k = 1, 2, 3) con respecto al plano XY.

Como la biela está unida solidariamente al eje del motor, el ángulo α_k será el ángulo entre la biela y el plano horizontal XY. Dichos ángulos se miden de la manera habitual: en sentido anti-horario con 0° en la posición 3 en punto.

En la Figura 4.3, podemos identificar, además, los diferentes parámetros del sistema:

· r: longitud de las bielas.
· p : longitud de los pistones.
· $\overrightarrow{L_k}$: posición de los vértices del triángulo inferior (k = 1, 2, 3).
· $\overrightarrow{U_{k0}}$: posición inicial (ya que se mueven) de los vértices del triángulo superior $\overrightarrow{U_k}$.

En el caso de la plataforma T1R2, estos parámetros presentan ciertas restricciones debido al diseño del manipulador:

· $p > r$ (si no, el triángulo superior colisionaría con el inferior).
· El triángulo superior y el inferior son, en posición inicial, iguales.
· El triángulo superior y el inferior son, al menos, isósceles.

· Tanto el triángulo superior como el inferior están alineados con respecto al eje Z cuando las bielas están situadas en ángulo recto (α_k = 90°).

4.2.2.2 – Posición de Reposo y Origen del Sistema de Coordenadas

La posición inicial (o de reposo) de la plataforma puede elegirse de forma arbitraria. Una posible elección, que posteriormente revisaremos, es elegir como posición de reposo aquella en la que todas las bielas de todos los motores están en posición horizontal (α_k = 0º). La posición que tenga la base móvil, con esa posición de las bielas, será considerada la pose (0, 0, 0) (i.e. *heave* = 0 m, *pitch* = 0º, *roll* = 0º).

En esta posición, la distancia entre el triángulo superior y el inferior se puede calcular empleando la regla de Pitágoras (ver Figura 4.4). Llamaremos h_0 a esta distancia:

$$h_0 = \sqrt{p^2 - r^2} \qquad (4.1)$$

La elección del origen de coordenadas también es arbitraria. Escogeremos, de momento, el centro del triángulo superior (cuando la plataforma esté en posición de reposo) como origen del sistema de coordenadas. Tanto este punto como las direcciones de los ejes, se pueden ver en la Figura 4.4.

Como centro de dicho triángulo escogeremos el baricentro, dado que es el centro de gravedad y el punto por el que el eje

estriado corta al plano XY. Sin embargo, como veremos más adelante, la elección del baricentro como centro del sistema de coordenadas de la plataforma puede tener consecuencias.

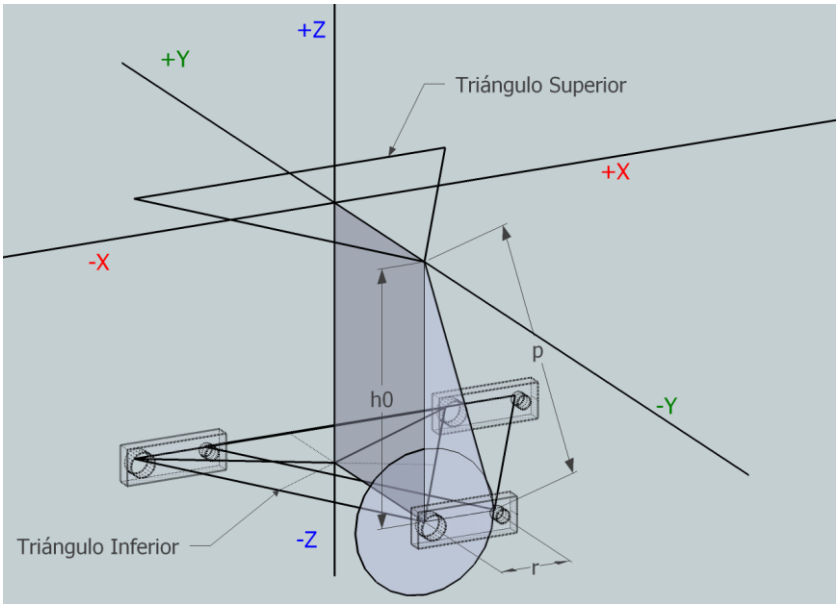

Figura 4.4 - *Sistema de coordenadas y posición de reposo*

4.2.2.3 – Formulación

El objetivo de la cinemática inversa es obtener α_k a partir de *heave*, *pitch*, *roll*. Para ello, necesitamos calcular, primero, la posición de $\overrightarrow{U_k}$ dado *heave*, *pitch*, *roll*.

$\overrightarrow{U_k} = \overrightarrow{U_{k0}}$ en la posición de reposo, pero conforme la base móvil se mueve, $\overrightarrow{U_k}$ ha de ser recalculado:

$$\overrightarrow{U_k} = R_x R_y \overrightarrow{U_{k0}} + \overrightarrow{(0,0,heave)} \qquad (4.2)$$

con

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cp & -sp \\ 0 & sp & cp \end{bmatrix} \qquad (4.3)$$

$$R_y = \begin{bmatrix} cr & 0 & sr \\ 0 & 1 & 0 \\ -sr & 0 & cr \end{bmatrix} \qquad (4.4)$$

siendo

$$cp = \cos(pitch) \quad sp = \text{sen}(pitch)$$
$$cr = \cos(roll) \quad sr = \text{sen}(roll)$$

Recordemos que $\overrightarrow{U_{k0}}$ es un parámetro de la plataforma.

Ahora, podemos calcular la posición de los puntos del triángulo inferior ($\overrightarrow{L_k}$) con respecto a los del triángulo superior ($\overrightarrow{U_k}$):

$$\overrightarrow{L_k} = \overrightarrow{U_{k0}} - \overrightarrow{(0,0,h_0)} - \overrightarrow{U_k} \qquad (4.5)$$

Es importante entender que aunque $\overrightarrow{L_k}$ no se mueve, estamos calculando estas variables con respecto a $\overrightarrow{U_k}$ por lo que $\overrightarrow{L_k}$ cambiará siempre que el manipulador se mueva (ya que $\overrightarrow{U_k}$ cambiará). Hacemos esto para simplificar posteriores ecuaciones.

Si la componente Z de $\overrightarrow{L_k}$ es positiva, entonces existe una solución matemáticamente plausible pero físicamente absurda con el triángulo superior debajo del inferior. Esta solución se marcará como "no factible" y se escogerá $\alpha_1 = \alpha_2 = \alpha_3$ = -90°, ya que es la configuración más cercana a ella.

Si la componente Z de $\overrightarrow{L_k}$ es negativa, entonces hemos de calcular la posición donde el pistón y la biela pueden encontrarse (si lo hacen, porque puede que la pose deseada sea inalcanzable). Para ello, hemos de calcular la intersección de una esfera de radio *p* centrada en $\overrightarrow{U_k}$, con una circunferencia vertical (plano XZ) de radio *r*, centrada en $\overrightarrow{L_k}$ (ver Figura 4.3). Para simplificar la formulación de la intersección esfera-circunferencia, $\overrightarrow{L_k}$ se expresa relativo a $\overrightarrow{U_k}$. Por ello, podemos considerar la esfera centrada en (0, 0, 0) y la circunferencia, centrada en $\overrightarrow{L_k}$. Como la ecuación es cuadrática, pueden haber hasta dos soluciones: $\overrightarrow{X_{k1}}$ y $\overrightarrow{X_{k2}}$ pero también puede no haber solución real:

$$(\overrightarrow{X_{k1}}, \overrightarrow{X_{k2}}, \text{factible}) = \text{IntersecEsferaCirculo}\left(p, r, \overrightarrow{L_k}\right) \quad (4.6)$$

Como tanto $\overrightarrow{X_{kJ}}$ y $\overrightarrow{L_k}$ se expresan en el mismo sistema de coordenadas (con respecto a $\overrightarrow{U_k}$), la substracción de dichos puntos nos da inmediatamente los dos posibles vectores de orientación de las bielas (que llamaremos $\overrightarrow{B_{k1}}$ y $\overrightarrow{B_{k2}}$):

$$\overrightarrow{B_{k1}} = \overrightarrow{X_{k1}} - \overrightarrow{L_k} \quad (4.7)$$
$$\overrightarrow{B_{k2}} = \overrightarrow{X_{k2}} - \overrightarrow{L_k} \quad (4.8)$$

Finalmente, estos dos vectores nos dan las dos soluciones finales (los dos ángulos de las bielas):

$$\alpha_{k1} = \text{atan}(B_{k1z}/B_{k1x}) \quad (4.9)$$
$$\alpha_{k2} = \text{atan}(B_{k2z}/B_{k2x}) \quad (4.10)$$

Como podemos ver, surgen siempre dos soluciones. Las soluciones son duales porque la intersección biela-pistón (circunferencia-esfera) tiene dos soluciones diferentes (excepto por las singularidades, que veremos después, en las que hay una solución doble repetida). El proceso de cálculo de la intersección circunferencia-esfera requiere alguna explicación adicional.

Llamemos (x, y, z) al punto donde una esfera de radio *p* centrada en (0, 0, 0) y una circunferencia vertical (en el plano XZ) de radio *r* centrada en un punto $\vec{L}$ (por tanto, y = L_y) intersectan:

$$(x, y, z) = \text{IntersecEsferaCirculo}(p, r, \vec{L}) \quad (4.11)$$

Las ecuaciones son:

$$\begin{gathered} x^2 + y^2 + z^2 = p^2 \\ (x - L_x)^2 + (z - L_z)^2 = r^2 \\ y = L_y \end{gathered} \quad (4.12)$$

Las soluciones de este sistema de ecuaciones son las siguientes (al ser un sistema con ecuaciones cuadráticas, las soluciones son duales):

$$
\begin{aligned}
y &= L_y \\
x &= -\frac{r^2 {L_x}^2 - {L_x}^2({L_x}^2 + {L_z}^2 + p^2 - y^2) + L_z a}{2L_x({L_x}^2 + {L_z}^2)} \\
z &= \frac{-r^2 L_z + L_z({L_x}^2 + {L_z}^2 + p^2 - y^2) + a}{2({L_x}^2 + {L_z}^2)}
\end{aligned}
\tag{4.13}
$$

y también:

$$
\begin{aligned}
y &= L_y \\
x &= \frac{-r^2 {L_x}^2 + {L_x}^2({L_x}^2 + {L_z}^2 + p^2 - y^2) + L_z a}{2L_x({L_x}^2 + {L_z}^2)} \\
z &= -\frac{r^2 L_z - L_z({L_x}^2 + {L_z}^2 + p^2 - y^2) + a}{2({L_x}^2 + {L_z}^2)}
\end{aligned}
\tag{4.14}
$$

donde

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

Si a es un número complejo (el radicando es negativo), no hay solución válida, ya que el pistón no puede alcanzar a la biela para dicho conjunto de GdL. En este caso, se pondría "factible" a "no".

Si a es un número real, se resuelve la intersección y "factible" se pone a "sí", obteniendo inmediatamente las soluciones duales de la CI.

En cualquier caso, hay que estar atento a las singularidades causadas cuando L_x tiende a 0, porque producen interminaciones, que se resuelven calculando el límite:

$$\lim_{L_x \to 0} x = \frac{-\sqrt{-r^4 + 2r^2\left({L_z}^2 + p^2 - y^2\right) - ({L_z}^2 - p^2 + y^2)^2}}{2L_z} \quad (4.15)$$

$$\lim_{L_x \to 0} z = \frac{\sqrt{-r^4 + 2r^2\left({L_z}^2 + p^2 - y^2\right) - ({L_z}^2 - p^2 + y^2)^2}}{2L_z} \quad (4.16)$$

De nuevo, si x ó z son números complejos (radicando negativo), no habría solución posible.

4.2.3 – Validación Empírica de la Cinemática Inversa del Manipulador

Para realizar la validación de la formulación anterior, tuvimos que elegir una instancia particular del manipulador paralelo T1R2. Elegimos los siguientes parámetros, que representan una configuración típica para la simulación de movimiento de bajo coste, y luego lo construimos:

· Longitud de pistón: $p = 0.35$

· Longitud de biela: $r = 0.1$

· Triángulo equilátero de lado 45 cm, centrado en su baricentro.

En la imagen de la Figura 4.5 podemos ver el prototipo final ya construido.

Para realizar la evaluación de la formulación cinemática inversa, utilizamos un sistema de seguimiento (*tracking)* de tipo óptico, compuesto por 6 cámaras *Natural Point Optitrack V100:R2* [153]. Este es un sistema de seguimiento óptico pasivo muy robusto y estable, que utiliza marcadores revestidos con un material reflectante para reflejar la luz infrarroja que se genera cerca de la lente de las cámaras (la Figura 4.5 muestra el sistema de cámaras infrarrojas empleado para la medición y un ejemplo del marcador situado en la base de la plataforma). Esta luz reflejada permite obtener la posición real de los marcadores, y por tanto hacer *tracking* de la base móvil del MP analizado. El *software* de las cámaras se encarga de todos los cálculos necesarios para obtener la posición y orientación del marcador (la pose real de la plataforma en nuestro caso), y además proporciona herramientas para la calibración de las cámaras. Este sistema óptico permite validar la respuesta real de la plataforma con respecto a la formulación de la cinemática inversa con un error inferior a 2 milímetros.

La validación de la CI se llevó a cabo mediante la petición a la plataforma de varios valores deseados para los 3 GdL (*heave*, *pitch* y *roll*) y la comparación de la respuesta del sistema (medida ópticamente) con respecto a los valores deseados. De esta forma, si la plataforma es capaz de alcanzar dichos valores, es porque la CI es correcta, ya que la CI se emplea para decidir los comandos de posicionamiento de los motores.

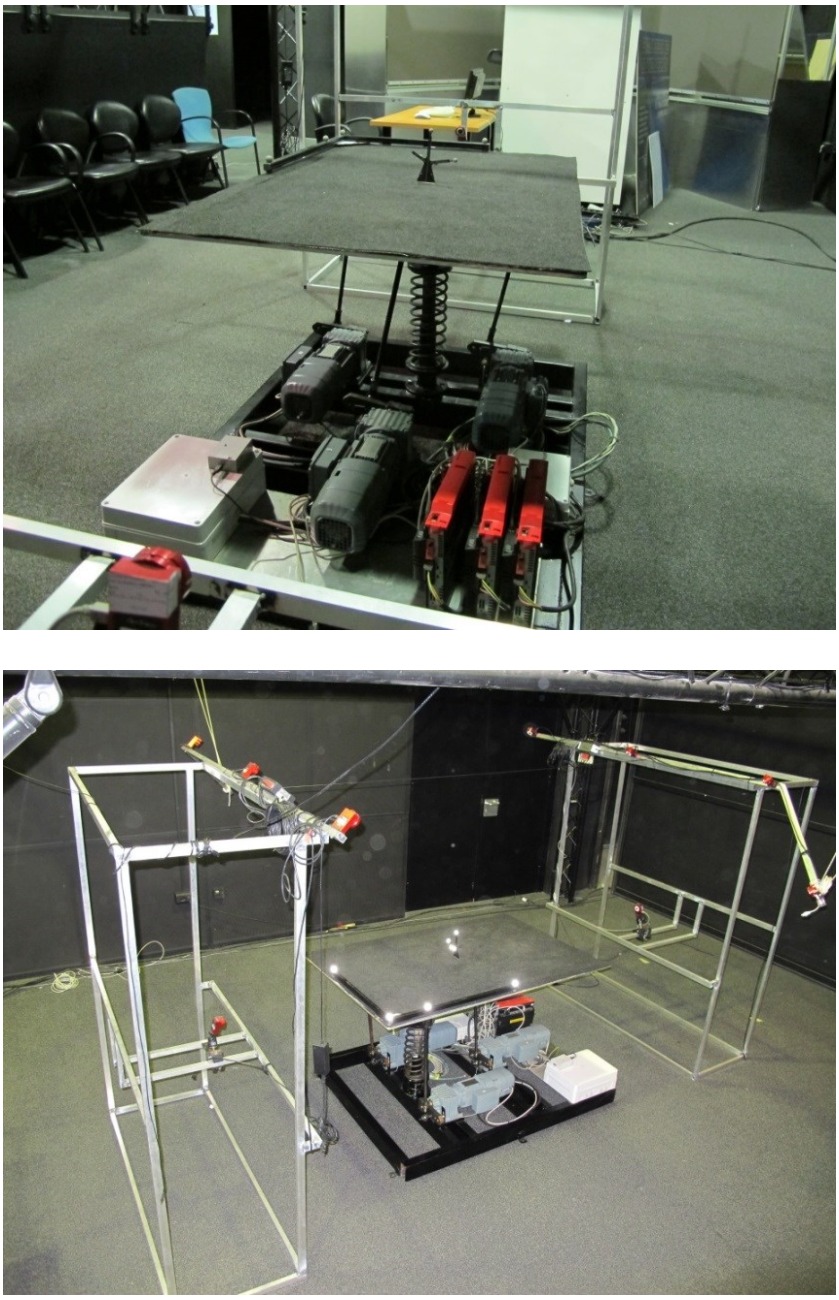

Figura 4.5 - *El MP T1R2 analizado, con tracking óptico*

La Figura 4.6 muestra los resultados obtenidos con nuestra plataforma. La línea azul corresponde al valor deseado mientras que la línea verde corresponde a la respuesta real del sistema. Como el lector puede ver, el GdL deseado se obtiene con una precisión de casi 100%. Obviamente, la plataforma de movimiento necesita algún tiempo para llegar a su pose deseada. Una vez que este transitorio termina, el comportamiento estacionario muestra muy pequeñas diferencias entre la pose esperada y la pose real. La diferencias se deben a errores de construcción (imposible medir), errores de posicionamiento de los motores (debe ser bastante pequeño porque utilizamos motores SEW-Eurodrive bastante exactos) y errores de precisión en la medición (menos de 2 milímetros como ya hemos dicho).

4.3 – Caracterización Cinemática

4.3.1 – Análisis del Espacio de GdL

Ahora que hemos desarrollado y validado las ecuaciones de la CI, nos gustaría saber qué forma y tamaño tiene el espacio de GdL disponible/alcanzable de la plataforma, y cómo es la relación entre los diversos GdL. Como contamos con una plataforma de movimiento de 3 GdL, podemos dibujar todo este espacio en un gráfico tridimensional *heave-pitch-roll*, pero podemos hacer también 3 gráficos bidimensionales de 2 GdL (*heave-pitch*, *heave-roll*, *pitch-roll*) fijando el restante GdL a 0. Estos gráficos bidimensionales son visualmente más fáciles de comprender.

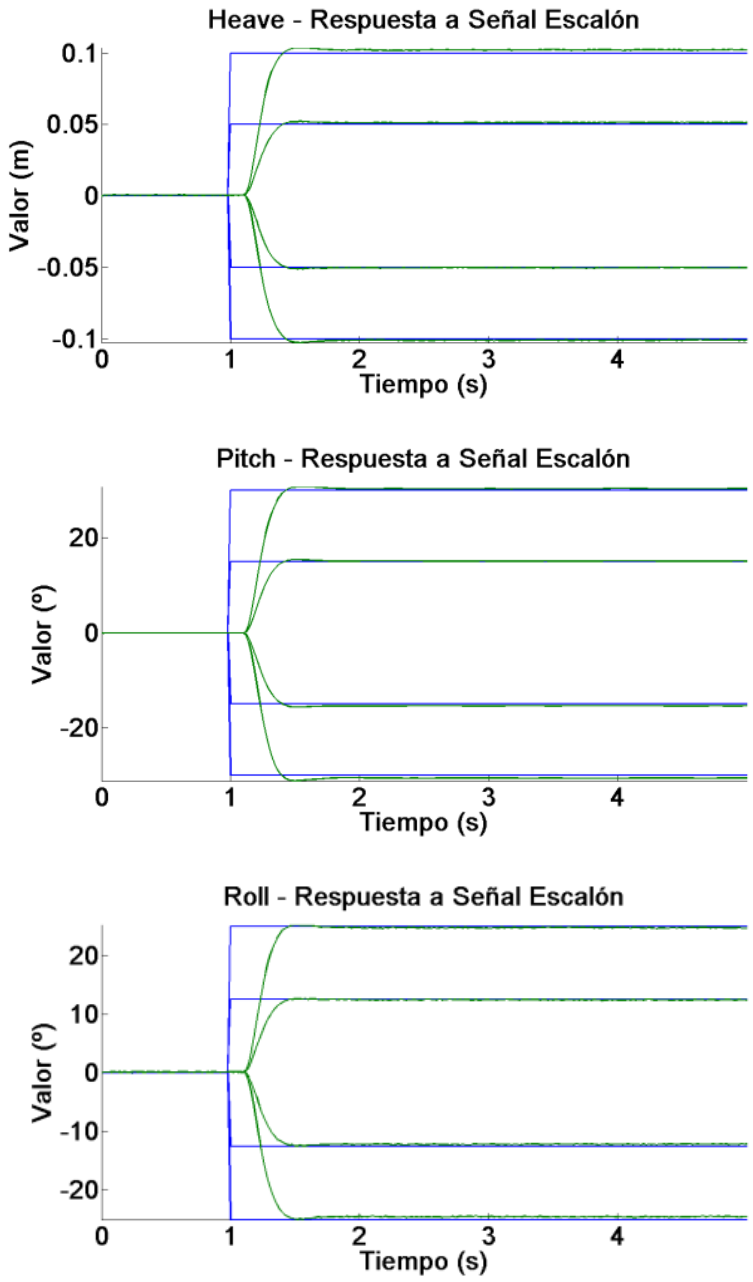

Figura 4.6 - *Validación de la cinemática inversa*

Como el análisis del espacio de GdL debe realizarse sobre una plataforma concreta, ya que el espacio de GdL depende de los parámetros de construcción del manipulador, decidimos elegir como instancia de prueba la misma configuración que elegimos para la validación de la CI. El análisis del espacio de GdL se realiza en 3 etapas:

Etapa 1: Búsqueda Completa y Gráfica Tridimensional

La primera etapa es una búsqueda numérica (discretizada) completa dentro del espacio de GdL con estos límites e incrementos:

· Rango de *heave* = [-0.12 m, 0.12 m], con incrementos de 0.0001 m.
· Rango de *pitch* = [-35°, 35°], con incrementos de 0.1°.
· Rango de *roll* = [-35°, 35°], con incrementos de 0.1°.

Los rangos fueron escogidos manualmente previa inspección de los valores máximos posibles. En esta etapa, cada punto dentro de ese espacio de GdL fue probado contra la CI (usando la formulación previamente explicada) para saber si era alcanzable o no con el manipulador. Con esta información, podemos entonces dibujar la línea fronteriza entre poses inalcanzables y factibles, y obtener un gráfico tridimensional de la forma del espacio de soluciones alcanzables.

Etapa 2: Búsqueda Bivaluada y Gráficas Bidimensionales.

La segunda etapa consiste en una búsqueda (dentro del espacio de GdL) similar a la de la etapa 1, con los mismos límites, pero empleando espacios de GdL bidimensionales. Para ello, primero, fijamos el *roll* a 0° y buscamos en el espacio *heave-pitch*. Después hacemos lo mismo con el espacio *heave-roll* fijando *pitch* a 0°. Finalmente, se hace lo mismo para el espacio *pitch-roll* con el *heave* fijo a 0 m.

Etapa 3: Búsqueda de Zonas Seguras

Como los espacios de soluciones factibles de los MP no suelen presentar formas ortogonales a los ejes, es complicado ajustar los MCA para el espacio de GdL de cada plataforma de movimiento concreta [85]. Esto puede provocar que la plataforma acabe alcanzando sus límites en múltiples ocasiones, cosa que hay que evitar [154]. La puesta a punto de los MCA es ya de por sí lo suficientemente compleja como para tener que hacerla sin saber realmente cuáles son los límites efectivos de la plataforma para cada caso. Cuando el MCA pide a la plataforma un conjunto de GdL que ésta no puede alcanzar, el MCA (o el control de la plataforma) debe pararla o hacer un escalado para encontrar un conjunto (más reducido) de GdL que sí pueda cumplirse. Esto provoca que se generen claves gravito-inerciales falsas, y se reduzca la validez del simulador [155], [76].

Por otra parte, incluso si conocemos la forma que define el conjunto de GdL alcanzables, los MCA se ajustan generalmente por canales [108] donde cada canal representa un GdL traslacional o rotacional de los 6 que presenta el espacio. Por ello, la puesta a punto de cada canal suele realizarse individualmente. El problema,

entonces, es que, con manipuladores paralelos, algunos GdL pueden ser alcanzables hasta un cierto límite, pero no todos al mismo tiempo. Por ejemplo, podríamos llegar a 25º de *pitch* (por ejemplo con *roll* a 0º) y quizás también podríamos obtener 25º de *roll* (con *pitch* a 0º), pero probablemente no podríamos obtener nunca 25º de *pitch* y 25º de *roll* simultáneamente. Esta característica es una consecuencia inherente de los diseños paralelos, aunque depende de cada MP concreto.

Así pues, el hecho de que los canales de los MCA son ajustados por separado mientras que los GdL de los MP son profundamente interdependientes, dificulta enormemente el ajuste de los MCA, porque las combinaciones inalcanzables de GdL tienen que ser escaladas (reducidas) y dicha situación no se puede predecir ni evitar si no se conoce la forma del espacio de soluciones del MP. Este proceso de escalado es muy complejo si se desea minimizar la pérdida de generación de claves gravito-inerciales [76]. Por lo tanto, es interesante definir una **zona segura** en la que los MCA puedan funcionar sin tener que preocuparse de la interdependencia de los GdL, evitando la necesidad de reducir cualquier petición de GdL. Esto simplificaría enormemente el ajuste de los MCA. Sin embargo, el problema de definir esta zona segura es que puede provocar una infrautilización del dispositivo, al descartar parte del espacio de GdL. Por lo tanto, es importante, hacer la zona segura tan grande como sea posible.

En el caso del manipulador analizado, la forma más fácil de definir esta zona segura es definir 6 valores: *heave* máximo, *heave* mínimo, *pitch* máximo, *pitch* mínimo, *roll* máximo y *roll* mínimo, dentro de los cuales, cualquier combinación de GdL pueda

satisfacerse sin necesidad de reducir ningún GdL. Esto define un hiper-rectángulo alineado con los ejes que puede (o no) estar centrado en el (0, 0, 0) del espacio de GdL. También pueden definirse otras zonas seguras (esféricas, hiper-elipsoidales, etc.), pero el hiper-rectángulo es quizás la forma más conveniente para la simulación de movimiento, por la independencia de los canales de los MCA. Si, además, queremos que esta zona segura tenga los mismos límites positivos y negativos, entonces, este hiper-rectángulo debe ser simétrico para cada eje. Por ejemplo: (-0,05 m, +0.05 m) para *heave*, (-20.0 °, +20.0 °) para *pitch* ó (-15.0 °, 15,0 °) para el *roll*. Esto suele ser necesario, porque los MCA suelen trabajar con filtros de frecuencia que son simétricos.

Si no necesitamos esa simetría, entonces sólo nos hace falta el mayor hiper-rectángulo alineado a los ejes y podemos conseguir, por ejemplo, resultados como: (-0,05 m, 0.026 m) para el *heave*, (-12.0 °, +16.0 °) para el *pitch* y (-15.0 °, 11,0 °) para el *roll*.

Por todo ello, la tercera etapa del análisis consiste en una aproximación de Monte-Carlo en busca de los mayores hiper-rectángulos alineados con los ejes (tanto simétricos como asimétricos), dentro del espacio de GdL.

A continuación podemos ver el algoritmo para el cálculo del espacio de GdL alcanzables por un MP de 3 GdL de tipo *heave-pitch-roll* (etapa 1). El cálculo sería análogo para la búsqueda en espacios bidimensionales *heave-pitch*, *pitch-roll* y *heave-roll* (etapa 2) por lo que obviamos mostrar los algoritmos correspondientes.

Posteriormente presentamos el algoritmo para el cálculo del rectángulo simétrico en el espacio *heave-pitch* (ejemplo de la etapa 3). El cálculo sería análogo para la búsqueda de rectángulos en otros espacios bidimensionales similares de la etapa 3 (*pitch-roll* y *heave-roll*) y ligeramente más compleja pero análoga para el espacio tridimensional *heave-pitch-roll*. La búsqueda de las zonas seguras asimétricas es muy similar.

Ambos algoritmos son válidos para cualquier plataforma de movimiento de 3 GdL de tipo *heave-pitch-roll*. Evidentemente también se podría usar para otras plataformas de 3 GdL con otros GdL con mínimos cambios (básicamente la nomenclatura). Su extrapolación a 4, 5 ó 6 GdL es sencilla, ya que, aunque se generan soluciones computacionalmente mucho más costosas, sus algoritmos son muy similares.

Algoritmo **BusquedaGdL-HPR**

Entradas:
p : Plataforma3GdL // plataforma de movimiento con 3 GdL
// (0 = heave, 1 = pitch, 2 = roll)

Parámetros:
infGdL : vector[1..3] de R *// límite inferior de la búsqueda*
supGdL : vector[1..3] de R *// límite superior de la búsqueda*
incGdL : vector[1..3] de R *// incremento de cada iteración de búsqueda*

Auxiliar:
res, ultimoRes : booleano
GdL, factibleGdL : vector[1..3] de R

Salidas:

espacioGdL : conjunto de vector[1..3] de R *// volumen de soluciones alcanzables*

Algoritmo:

```
GdL[0] = infGdL[0];
mientras (GdL[0] < supGdL[0]) hacer
{
   res = falso;

   GdL[1] = infGdL[1];
   mientras (GdL[1] < supGdL[1]) hacer
   {
      GdL[2] = infGdL[2];
      mientras (GdL[2] < supGdL[2]) hacer
      {
         ultimoRes = res;
         (res, factibleGdL) = p.comprobarGdL(GdL);

         si (res ≠ ultimoRes) entonces
            espacioGdL.añadir(factibleGdL);

         GdL[2] += incGdL[2];
      }
      GdL[1] += incGdL[1];
   }
   GdL[0] += incGdL[0];
}

GdL[2] = infGdL[2];
mientras (GdL[2] < supGdL[2]) hacer
{
   res = falso;

   GdL[1] = infGdL[1];
   mientras (GdL[1] < supGdL[1]) hacer
```

```
    {
      GdL[0] = infGdL[0];
      mientras (GdL[0] < supGdL[0]) hacer
      {
        ultimoRes = res;
        (res, factibleGdL) = p.comprobarGdL(GdL);

        si (res ≠ ultimoRes) entonces
          espacioGdL.añadir(factibleGdL);

        GdL[0] += incGdL[0];
      }
      GdL[1] += incGdL[1];
    }
    GdL[2] += incGdL[2];
}
```

Fin de BusquedaGdL-HPR

Algoritmo **MonteCarlo-MaxRectSimetrico-HP**

Entradas:

p : Plataforma3GdL // plataforma de movimiento con 3 GdL
// (0 = heave, 1 = pitch, 2 = roll)

Parámetros:

numRectangulos : N // número de rectángulos
infGdL : vector[1..3] de R *// límite inferior de la búsqueda*
supGdL : vector[1..3] de R *// límite superior de la búsqueda*

Auxiliar:

res : booleano
area, h, p : R
GdL, factibleGdL : vector[1..3] de R

Salidas:

maxGdL : vector[1..3] de R // *máxima extensión rectangular simétrica*
maxArea : R // *área correspondiente a maxGdL*

Algoritmo:

```
GdL[2] = 0.0;
maxGdL[2] = 0.0;
maxArea = 0.0;

desde (i = 0 hasta numRectangulos) hacer
{
   h = generaAleatUnitario()*(supGdL[0] – infGdL[0]) + infGdL[0];
   p = generaAleatUnitario()*(supGdL[1] – infGdL[1]) + infGdL[1];

   GdL[0] = h;
   GdL[1] = p;
   (res, factibleGdL) = p.comprobarGdL(GdL);

   si (res == falso) entonces
      continuar con la siguiente iteración;

   GdL[0] = h;
   GdL[1] = -p;
   (res, factibleGdL) = p.comprobarGdL(GdL);

   si (res == falso) entonces
      continuar con la siguiente iteración;

   GdL[0] = -h;
   GdL[1] = -p;
   (res, factibleGdL) = p.comprobarGdL(GdL);

   si (res == falso) entonces
      continuar con la siguiente iteración;
```

```
    GdL[0] = -h;
    GdL[1] = p;
    (res, factibleGdL) = p.comprobarGdL(GdL);

    si (res == falso) entonces
       continuar con la siguiente iteración;

    area = 4.0*h*p;
    si (area > maxArea) entonces
    {
        maxGdL[0] = h;
        maxGdL[1] = p;
        maxArea = area;
    }
}
```

Fin de MonteCarlo-MaxRectSimetrico-HP

Los gráficos resultantes de ejecutar los anteriores algoritmos y otros análogos sobre la plataforma de 3 GdL anteriormente descrita, pueden verse en la Figura 4.7 donde se puede apreciar el resultado del proceso. El análisis del espacio de GdL, utilizando las fórmulas originales, nos revela bastante información. En primer lugar, el espacio de soluciones alcanzables (líneas y puntos azules) resulta ser una especie de caja rotada. Como esperábamos, no es ortogonal a los ejes. Por otra parte, esta caja no está centrada en el (0, 0, 0), como podemos ver en los gráficos bidimensionales. Además, los GdL muestran relaciones no simétricas entre ellos (especialmente la relación *pitch-roll*). El punto (0, 0, 0) del espacio de GdL se marca con un círculo rojo y el centro de la caja que representa los GdL alcanzables por el manipulador, con un círculo azul.

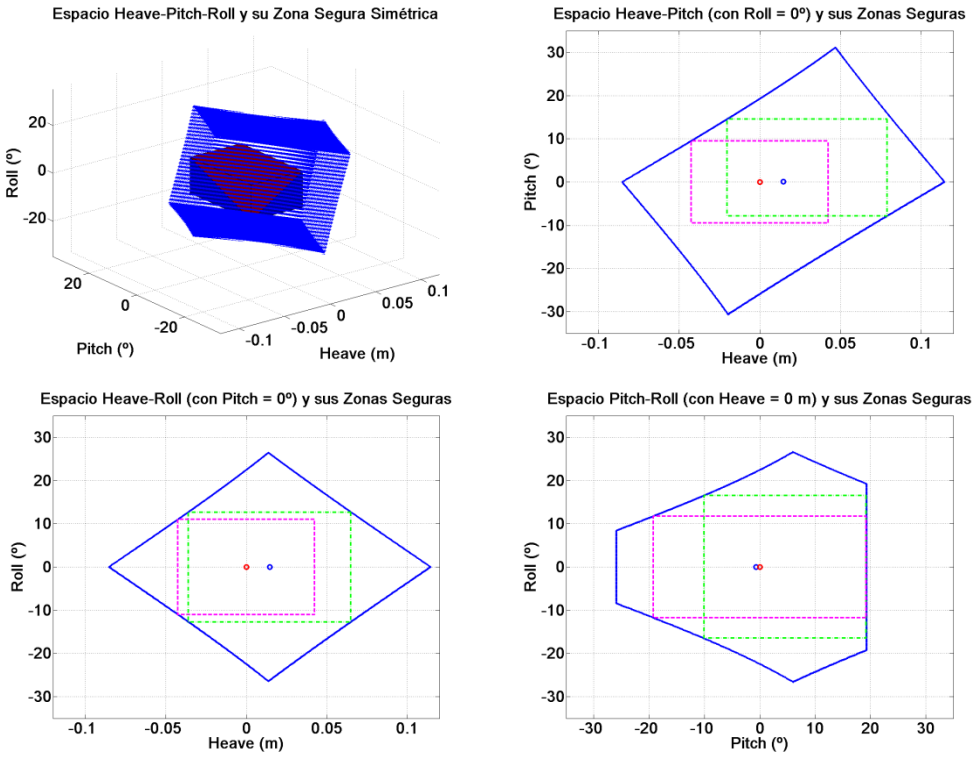

Figura 4.7 - *Espacio de soluciones alcanzables*

En segundo lugar, el rango del *heave* no es simétrico, y su forma es tal que es difícil de cubrir con un rectángulo alineado con los ejes. Los rectángulos simétricos posibles más grandes se representan gráficamente con una línea punteada rosada, mientras que los rectángulos asimétricos se trazan en verde. El hiper-rectángulo tridimensional asimétrico más grande (zona segura) se traza también con una caja de color azul-rojo.

Y en tercer lugar, el hecho de que, por ejemplo, el *pitch* máximo no puede alcanzarse con *heave* 0 es una mala noticia para

los MCA, porque el escalado de los GdL es más difícil de realizar. Lo mismo ocurre con *heave-roll* y *pitch-roll*.

4.3.2 – Mejoras sobre el Espacio de GdL

4.3.2.1 – *Offset* Vertical

Para entender por qué el rango de *heave* no es simétrico, primero debemos comprender cuán grande es este rango. La respuesta a esta pregunta está en la Figura 4.8: cuando el *heave* llega a su valor máximo, la distancia entre los triángulos superior e inferior es $(p + r)$. Cuando el *heave* alcanza su valor mínimo, esta distancia es $(p - r)$. Por lo tanto, el rango es $(p + r) - (p + r) = 2r$ (suponiendo que $p > r$, cosa que debe cumplirse siempre).

Para comprobar si el rango es simétrico, debemos analizar el *heave* en la posición de reposo. Si definimos la posición de reposo (*heave* = 0) con $\alpha_1 = \alpha_2 = \alpha_3 = 0°$, entonces la distancia entre los dos triángulos viene dada por la ecuación (4.1).

Por tanto (ver Figura 4.8):

El *heave* máximo es $(p + r) - h_0 = r + (p - h_0) = r + (p - \sqrt{p^2 - r^2})$.

El *heave* mínimo es $-(h_0 + r - p) = -r + (p - h_0) = -r + (p - \sqrt{p^2 - r^2})$.

Y el rango de *heave* es $r + (p - h_0) - \big(-r + (p - h_0)\big) = r + (p - h_0) + r - (p - h_0) = 2r$.

Como se observa, el rango no es simétrico (*-r*, *r*) por culpa del término $(p - h_0)$ que causa un *offset*. Esto ocurre porque definimos la posición de reposo con $\alpha_1 = \alpha_2 = \alpha_3 = 0°$. Si hacemos eso, con *heave* = 0, la distancia entre los dos triángulos es h_0 y no p (que sería el punto medio del rango). Como $h_0 \neq p$, este punto no es el punto medio del rango. El caso límite se da con $p = r$. En este caso, $h_0 = 0$, el triángulo superior y el inferior coinciden en la posición de reposo, el *heave* máximo es $p + r = 2r$, y el *heave* mínimo es $p - r = 0$. El rango sería 2*r*, pero completamente asimétrico (0, 2*r*) por culpa del *offset*.

En un caso general, este *offset* causa una diferencia entre el punto medio real del rango (h_0) y el punto medio deseado (p). Esta diferencia se puede calcular:

$$\text{offset} = p - h_0 = p - \sqrt{p^2 - r^2} \qquad (4.17)$$

Por tanto, la condición para obtener un rango simétrico de *heave* es conseguir que el *offset* sea 0:

$$\begin{gathered} \text{offset} = 0 \\ p - h_0 = 0 \\ p - \sqrt{p^2 - r^2} = 0 \\ p = \sqrt{p^2 - r^2} \\ p^2 = p^2 - r^2 \end{gathered} \qquad (4.18)$$

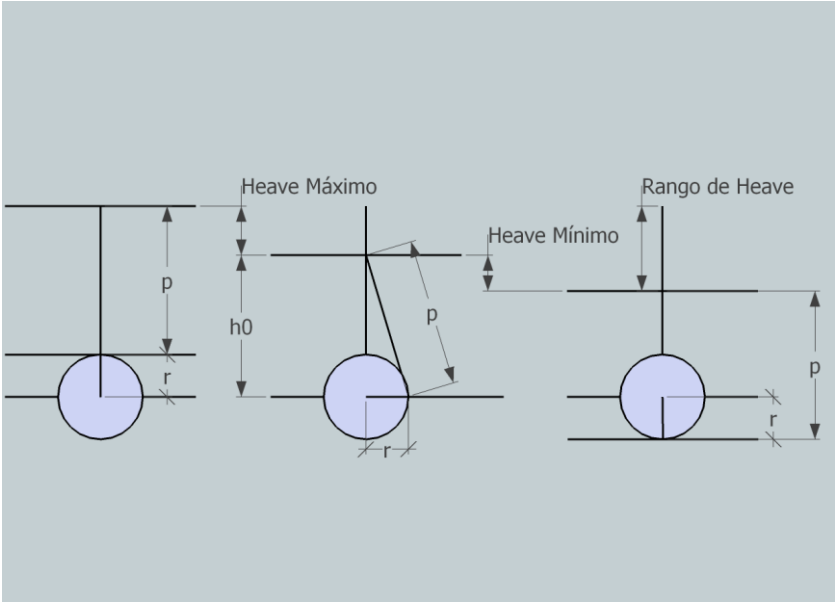

Figura 4.8 - *Rango del heave*

Esta condición sólo se cumple cuando $r = 0$ ó cuando p es infinito. Una biela de tamaño nulo no se puede construir y provocaría un rango nulo de *heave*. Y construir un pistón muy largo no es práctico, por lo que necesitamos otra solución.

La solución más inteligente es redefinir lo que consideramos como *heave* 0, restándole a dicha cantidad el *offset* ($p - h_0$) antes del cálculo de la CI. Esto hace que para obtener un *heave* igual a 0 necesitemos un cierto ángulo no nulo para los motores: $\alpha_1 = \alpha_2 = \alpha_3 = k \geq 0°$ (k se puede calcular usando la CI y no nos importa demasiado), por lo que la posición de reposo ($\alpha_1 = \alpha_2 = \alpha_3 = 0°$)

ya no causará la posición (0, 0, 0) del espacio de GdL (lo cual no debería suponer ningún problema).

Es importante entender que no necesitamos cambiar las ecuaciones de la CI; sólo tenemos que restar el *offset* del *heave* antes de aplicar las ecuaciones de la CI. De esta manera, tendremos un *heave* simétrico (–*r*, *r*). Si hacemos esto y repetimos el análisis del espacio de soluciones alcanzables, los resultados son bastante diferentes (ver Figura 4.9).

El rango del *heave* ahora es simétrico, y también lo es la relación *heave-roll*. Sin embargo, la relación *heave-pitch* todavía es bastante extraña. Por ejemplo, para obtener el *pitch* máximo (30°), el *heave* debe llegar a unos 0.04 m, lo cual parece muy poco intuitivo.

4.3.2.2 – Relación *Heave-Pitch*

La asimetría que se observa en la relación *heave-pitch* en las Figuras 4.7 y 4.9 proviene del triángulo superior. Dinámicamente, un triángulo equilátero proporciona simetría total para el reparto de carga sobre los motores. Con respecto al centro del triángulo, en un triángulo equilátero el ortocentro, el baricentro (centroide), el circuncentro y el incentro coinciden. Así pues, parece que este punto es la opción natural y ésta es la razón por la que lo elegimos.

Sin embargo, el baricentro puede no ser un buen lugar para situar nuestro sistema de coordenadas, porque si lo hacemos, estamos suponiendo que el triángulo superior girará alrededor de este punto, lo cual no es cierto.

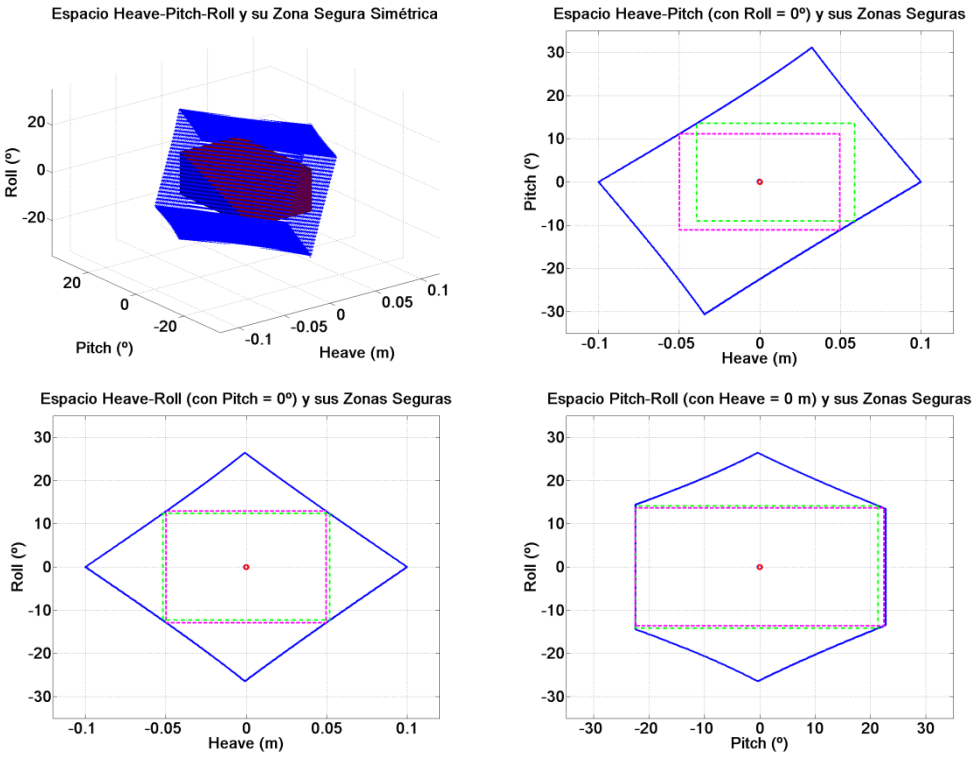

Figura 4.9 - *Espacio de soluciones alcanzables sin offset de heave*

Esto puede parecer un poco contra-intuitivo (pues tenemos un eje estriado vertical que pasa por el baricentro), pero como la articulación entre la base móvil y el eje vertical es una unión Cardan prismática, la base móvil no está limitada a girar alrededor de la articulación Cardan, porque el eje estriado permite a la base móvil libertad de movimientos en el eje Z, y, por lo tanto, puede girar alrededor de un punto diferente (más arriba o más abajo).

Esto significa que si establecemos el baricentro como el centro del triángulo superior, el *pitch* no puede alcanzar su valor

máximo con el *heave* a 0, ya que para obtener el *pitch* máximo, algo de *heave* es necesario, debido a un pequeño *offset* entre el centro real de rotación y el origen de nuestro sistema de coordenadas. Este *offset* no invalida las ecuaciones; sólo cambia la relación entre los GdL, porque una rotación en *pitch* es "vista" por nuestro sistema de coordenadas como un pequeño desplazamiento lineal (*heave*) más un desplazamiento angular (*pitch*).

Si suponemos que el pistón es varias veces más largo que la biela, entonces, podemos decir que la base girará con respecto a un punto a media distancia de la altura del triángulo (en el eje Y) y a media distancia desde la base del triángulo (en el eje X). Así, si utilizamos ese punto medio como el centro del triángulo en lugar del baricentro (en nuestras ecuaciones, no en el proceso de construcción), las rotaciones de *pitch* no implicarán desplazamientos de *heave*, el gráfico *heave-pitch* será simétrico y podremos obtener el máximo valor de *pitch* cuando *heave* y *roll* sean 0, porque el *offset* de rotación será muy cercano al 0 (ver Figura 4.10). Si el pistón es de un tamaño, en orden de magnitud, similar a la biela, habrá un pequeño *offset* de desplazamiento, pero si el pistón es suficientemente grande respecto a la biela, este punto medio hará que la relación *heave-pitch* y *pitch-roll* sea casi simétrica.

Así, aunque para la construcción de la plataforma todavía utilizaremos el baricentro como lugar para colocar el eje estriado vertical, cinemáticamente vamos a establecer este nuevo punto medio como el centro de nuestro sistema de coordenadas. Este cambio no implica reescribir la CI, sólo cambia el valor de algunos parámetros (en particular, $\overrightarrow{L_k}$ y $\overrightarrow{U_{k0}}$). Con este nuevo sistema de coordenadas, podemos repetir el análisis y observamos que el

espacio de GdL presenta una forma mucho más atractiva (véase la Figura 4.11).

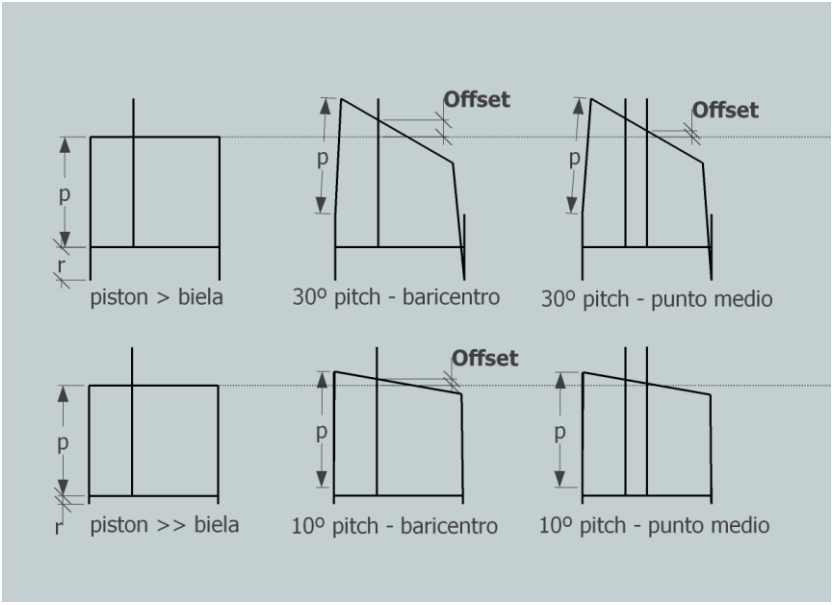

Figura 4.10 - *Offsets de rotación*

Ahora, todos y cada uno de los GdL pueden alcanzar su máximo valor cuando los otros valgan 0. Si esto aumenta la zona segura o no, es algo que tenemos que analizar, pero el hecho de que cada GdL pueda obtener su máximo valor cuando los otros son 0, es algo que facilita el diseño y ajuste de MCA. Es importante entender que mediante la aplicación de estos cambios propuestos, no cambiamos (físicamente) el manipulador, ni los rangos de sus GdL. Sólo los reorganizamos para que la utilización de la plataforma con un MCA sea más sencilla.

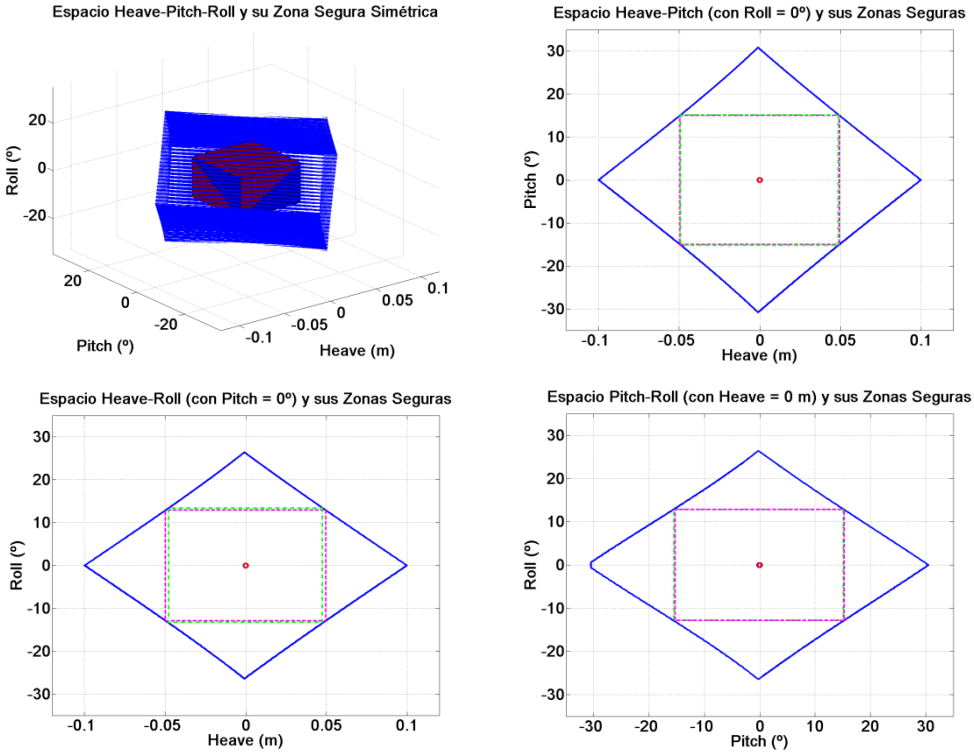

Figura 4.11 - *Espacio de soluciones alcanzables con el nuevo centro*

Finalmente, con el fin de analizar el efecto ya mencionado de pistón muy grande, se realizó el mismo análisis pero con un pistón de 10 m de largo (varios órdenes de magnitud superior a la longitud de la biela). Los resultados son prácticamente iguales (ver Figura 4.12). La única diferencia es que la pequeña asimetría en la relación *pitch-roll* (lado izquierdo de la gráfica de *pitch-roll* en la Figura 4.11) ya no está presente, aunque es prácticamente inapreciable. Así pues, la conclusión es que la longitud del pistón no modifica los rangos de los GdL de la plataforma. Sólo cambia su interdependencia.

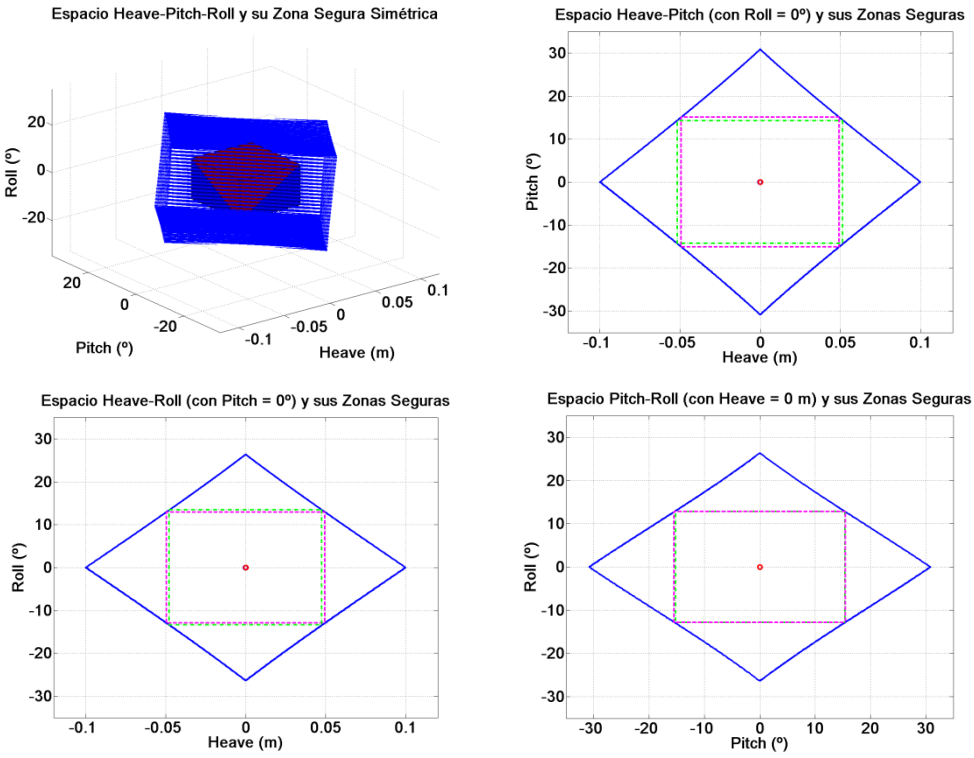

Figura 4.12 - *Espacio de soluciones alcanzables con un pistón muy largo (10 metros)*

Con esta configuración, los límites de las excursiones de todos los GdL son completamente simétricos, situación que, en principio, es deseable para su uso con algoritmos MCA, que es el uso que pretendemos hacer del manipulador paralelo:

Heave = [-0.1, 0.1] m
Pitch = [-30.5176, 30.5176] º
Roll = [-26.2134, 26.2134] º

4.3.3 – Análisis de Zonas Seguras

El último paso del estudio cinemático es el análisis del volumen o área (hiper-volumen en general) de las zonas seguras (hiper-rectángulos alineados con los ejes) para las diferentes configuraciones. Obviamente, cuanto más grande sea la zona, mejor para el uso con MCA. La Tabla 4.1 muestra los volúmenes de la zonas seguras simétricas (hiper-rectángulos simétricos alineados con los ejes) para las diferentes configuraciones (HPR significa *heave-pitch-roll*, HP significa *heave-pitch*, HR es *heave-roll* y PR, *pitch-roll*).

	HPR *(m*grado^2)*	*HP* *(m*grado)*	*HR* *(m*grado)*	*PR* *(grado^2)*
Configuración 1 (original)	21.58	1.6167	1.8705	906.70
Configuración 2 (sin offset en heave)	34.69	2.2177	2.5663	1225.60
Configuración 3 (nuevo centro de triángulo)	22.63	2.9702	2.5663	781.60
Configuración 4 (pistón muy largo)	22.72	2.9738	2.5680	791.40

Tabla 4.1 - *Hiper-rectángulos simétricos más largos: volúmenes y áreas*

Como podemos ver, la configuración original (1) es la peor en términos de volumen HPR. La configuración 2 tiene el hiper-rectángulo simétrico más grande. Las configuraciones 3 y 4 son muy similares, pero siempre mejores que la configuración 1. Las zonas seguras de HP y HR son las más grandes, pero el hiper-

rectángulo HPR y el rectángulo PR son menores que los de la configuración 2.

La Tabla 4.2 muestra el mismo estudio, pero con zonas seguras asimétricas (hiper-rectángulos asimétricos alineados con los ejes).

	HPR *(m*grado²)*	*HP* *(m*grado)*	*HR* *(m*grado)*	*PR* *(grado²)*
Configuración 1 (original)	34.16	2.2245	2.5548	964.64
Configuración 2 (sin offset en heave)	33.12	2.2191	2.5482	1238.67
Configuración 3 (nuevo centro de triángulo)	22.15	2.9552	2.5489	785.25
Configuración 4 (pistón muy largo)	22.15	2.9518	2.5588	784.37

Tabla 4.2 - *Hiper-rectángulos asimétricos más largos: volúmenes y áreas*

Como podemos ver, las configuraciones 1 y 2 son ahora las que tienen los mayores volúmenes para todo el espacio HPR. La configuraciones 3 y 4 permanecen lejos de esos valores. Ahora, la configuración 4 no es mejor que la configuración 3. Como la configuración 4 es la más simétrica, es lógico que haya muy poca mejora con el uso de una zona segura asimétrica. Además, ahora, la configuración 2 es similar a la configuración 1 para todos los casos excepto en el caso PR, donde aquélla es, de lejos, la mejor.

También podemos ver que algunos de los hiper-rectángulos asimétricos son ligeramente más pequeños que los simétricos. Esto es un efecto colateral del uso del método de Monte-Carlo, que introduce un pequeño error.

En conclusión, la decisión de usar la configuración 2 ó la 3 (la configuración 4 no es práctica) se basará en la importancia de tener una relación *heave-pitch* simétrica. Si la necesitamos, usaremos la configuración 3. Si no, elegiremos la configuración 2 en lugar de la configuración 1, porque maximiza la zona segura, tanto de forma simétrica como asimétrica y porque produce un rango de *heave* simétrico.

4.4 – Caracterización Dinámica

La caracterización dinámica de una plataforma de movimiento se puede hacer desde muchos puntos de vista. La literatura contiene numerosos análisis de diversos manipuladores, en los cuales se analizan velocidades, aceleraciones, precisiones, repetibilidad, singularidades, etc. [156], [157]. En simulación, las velocidades y las aceleraciones son importantes, pero otros aspectos como la precisión, o la repetibilidad son menos importantes, ya que no importa tanto que una pose se alcance con una precisión muy alta ni que ésta se reproduzca siempre con mínimas variaciones. Lo que importa es ser capaces de generar movimiento de una forma rápida para no introducir retrasos y evitar confundir al usuario.

Dado que un MEM es siempre incapaz de responder instantáneamente a las entradas que se le proporcionan, entenderemos por caracterización dinámica, en este apartado, la capacidad de analizar cuán rápido es capaz de reaccionar la plataforma a los movimientos que se le exige, obviando otros aspectos como la dexteridad [158], la precisión o la repetibilidad. En este sentido, es importante destacar que una plataforma de movimiento se comporta como un filtro pasa-baja, ya que su misión es reproducir como salida las entradas que se le proporcionan, pero las inercias de los elementos mecánicos hacen que dichas salidas siempre vayan retrasadas con respecto a las entradas y que variaciones muy grandes y muy rápidas de las entradas no puedan ser reproducidas.

Por tanto, para analizar la respuesta dinámica de una plataforma de movimiento, lo que debemos hacer es reproducir sobre ella diversos tipos de movimientos y medir la diferencia entre la salida real y el movimiento solicitado. Dicha diferencia se puede medir tanto en el dominio del tiempo como en el dominio de la frecuencia. En general, lo más habitual es analizar cada GdL por separado, estimulando cada uno de ellos con diferentes señales para observar su comportamiento y medir diversos indicadores. Utilizaremos, en este análisis, la misma plataforma empleada en el análisis cinemático.

En cuanto a la amplitud de las señales de prueba, es evidente que la respuesta de la plataforma no será igual ante pequeños desplazamientos que ante grandes movimientos. Sería interesante testear la plataforma ante un gran número de amplitudes distintas,

pero el análisis sería bastante tedioso, así que intentaremos elegir siempre las amplitudes más grandes que admita la plataforma.

4.4.1 – Análisis en el Dominio del Tiempo

Se pueden usar muchos tipos de señales de prueba, pero las señales más utilizadas para el análisis de sistemas son la señal escalón y la señal senoidal.

La señal escalón nos sirve para conocer cómo es la respuesta transitoria de un sistema ante un cambio brusco y súbito de la señal de entrada. En nuestro caso, nos sirve para conocer cuánto tiempo necesita la plataforma para cumplir una determinada orden de movimiento. Esto nos proporciona información en el dominio del tiempo acerca del retraso y de la velocidad máxima de la plataforma. La señal senoidal nos da información sobre cómo se comportará el sistema ante una entrada periódica. Aunque la señal es periódica, esto nos proporciona exclusivamente información en el dominio del tiempo, no de la frecuencia. Este tipo de prueba nos indicará cómo se comporta la plataforma ante cambios suaves de las entradas.

Así pues, para el análisis en el dominio del tiempo vamos a testear la plataforma principalmente (ya que probaremos alguna otra) ante las señales onda cuadrada y onda senoidal. La señal onda cuadrada es como la señal escalón, pero periódica. Las pruebas consisten en solicitar a la plataforma diferentes valores de los GdL (por separado) variados según las diferentes señales.

4.4.1.1 – Resultados y Conclusiones

En la Figura 4.13 podemos ver cómo se comporta el *heave* ante diversas entradas.

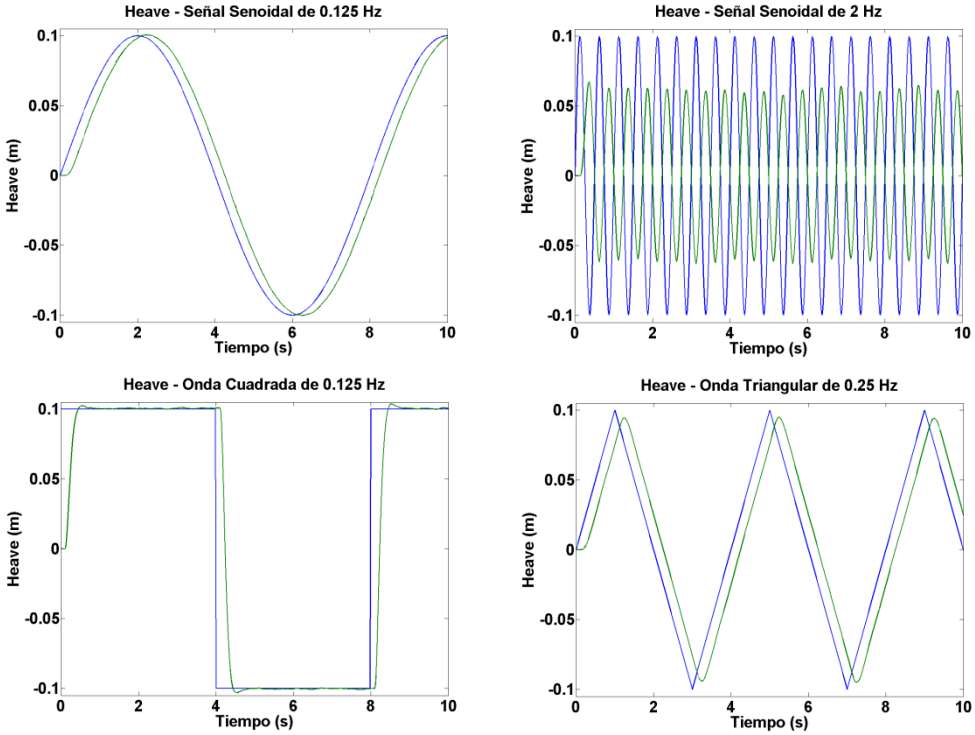

Figura 4.13 - *Comportamiento ante diversas entradas para el movimiento de heave*

Se puede apreciar que el retraso para una señal senoidal de 0.125 Hz (por tanto de 8 segundos de periodo) es de unos 0.125 segundos, lo cual representa un desfase de unos 5º. Sin embargo, para una señal senoidal de 2 Hz (periodo de 0.5 segundos), el retraso aumenta hasta aproximadamente 0.25 segundos, lo cual

representa un desfase de casi 180º. Este valor haría que un movimiento senoidal de dicha frecuencia fuera prácticamente irreconocible para un humano situado sobre la plataforma. Además, podemos ver, que la señal de 0.125 Hz se reproduce fielmente en cuanto a magnitud, pero la señal de 2 Hz, tiene una atenuación de casi el 40 % (de 0.1 m en la entrada a 0.06 m en la salida).

En la respuesta ante la onda cuadrada de 0.125 Hz podemos ver cómo reacciona la plataforma ante un cambio brusco en la entrada. Podemos observar que la plataforma tarda unos 0.12 segundos en reaccionar ante la nueva entrada. Este tiempo incluye el retraso en el procesado electrónico de la información y la inercia inicial de los motores que impide que el movimiento sea inmediato. Tras esto, la plataforma tarda otros 0.35 segundos aproximadamente en alcanzar una semi-excursión (de 0 a 10 cm), y prácticamente lo mismo en realizar una excursión completa (de 10 cm a -10 cm) de lo cual podemos deducir que la velocidad en este tramo supera los 50 cm/s. El hecho de que una excursión sea casi igual de costosa que una semi-excursión pone de manifiesto que el principal problema para estos dispositivos es el inicio del movimiento debido a su inercia.

El retraso total entre que la entrada cambia y la plataforma alcanza el desplazamiento deseado es de unos 0.5 segundos, a los cuales habría que sumarle algunas décimas para que la plataforma converja hacia la posición exacta deseada, ya que siempre se producen oscilaciones.

Por último, podemos ver la respuesta ante una onda triangular de 0.25 Hz. El comportamiento ante dicha señal es parecido al de la señal senoidal, aunque a diferencia de aquélla la salida es ligeramente diferente a la entrada, ya que los picos de la onda se suavizan a la salida. Asimismo podemos ver una pequeña atenuación, que será más grande cuanto mayor sea la frecuencia de la señal.

En la Figura 4.14 podemos ver cómo se comporta el *pitch* ante diversas entradas. Como la respuesta es muy similar a la del anterior GdL, en algunas gráficas se muestra una zona ampliada para visualizar la información de forma más cercana. Al igual que antes, una excursión completa del GdL (unos 60º) necesita unos 0.5 segundos.

Además, podemos ver la respuesta de este GdL ante una entrada en forma de seno amortiguado de 1 Hz, donde la amplitud de entrada va disminuyendo conforme aumenta el tiempo. Podemos observar que, aunque el desfase no cambia mucho aunque disminuya la amplitud, la ganancia sí que se ve afectada, ya que con una amplitud cercana al máximo (30º) la señal de salida se atenúa ligeramente, pero esta atenuación disminuye a medida que la amplitud del seno es menor.

En la Figura 4.15 podemos ver el mismo análisis para el *roll*. Las conclusiones son las mismas que para los anteriores GdL

Con estos análisis podemos deducir que la plataforma puede realizar una excursión en aproximadamente 0.5 segundos. Lo cual

se traduce en unos 40 cm/s de velocidad de excursión en *heave*, unos 60º/s en *pitch* y unos 50º/s en *roll*.

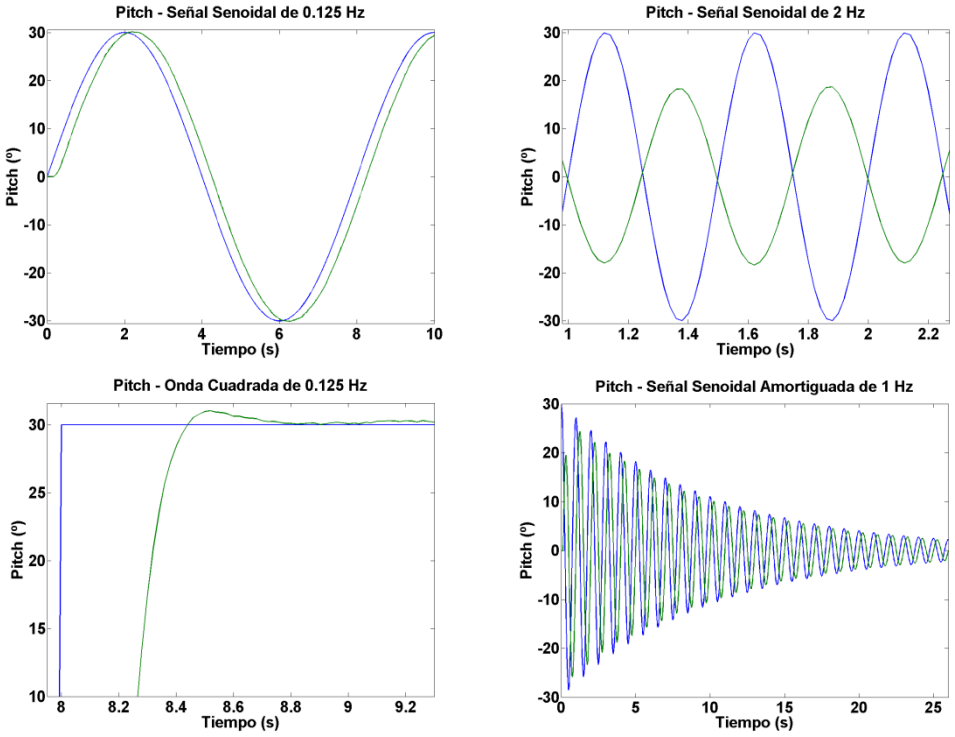

Figura 4.14 - *Comportamiento ante diversas entradas para el movimiento de pitch*

4.4.2 – Análisis Frecuencial

Una manera de proporcionar información en el dominio de la frecuencia es variar la frecuencia de las señales senoidales. También se pueden utilizar señales en forma de onda cuadrada,

diente de sierra, o incluso ondas triangulares. Sin embargo, una forma sencilla de analizar un sistema en el dominio de la frecuencia es emplear una señal de tipo *chirp*. Una señal de tipo *chirp* es una señal periódica de frecuencia incremental.

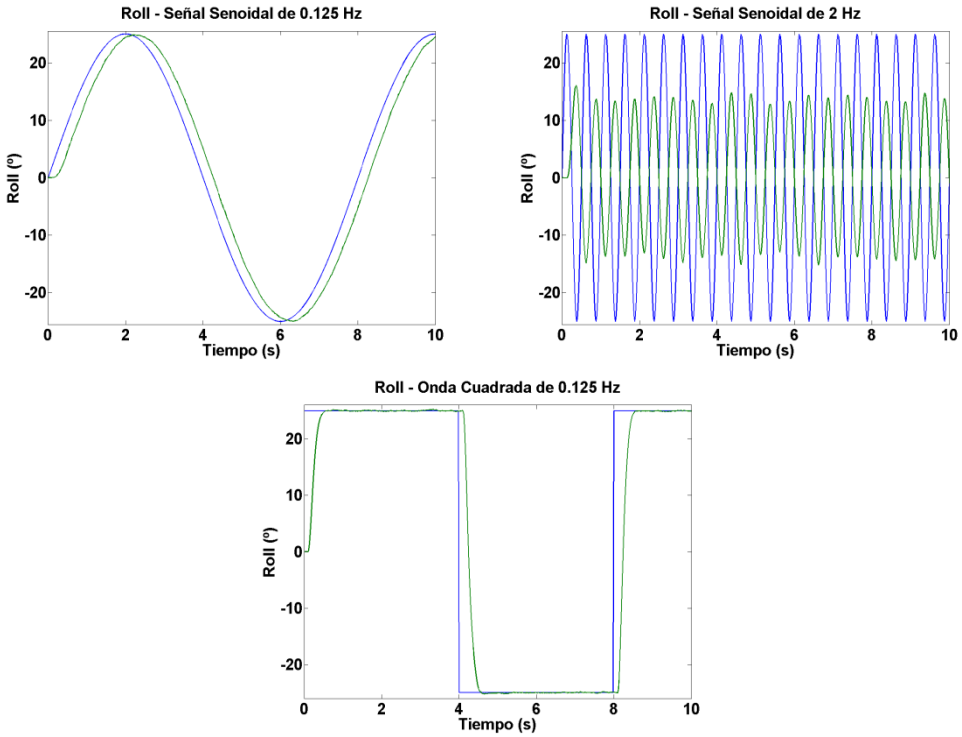

Figura 4.15 – *Comportamiento ante diversas entradas para el movimiento de roll*

Para el análisis en el dominio de la frecuencia vamos a emplear las señales *sine-chirp* y *square-chirp*. La función *sine-chirp* es una función senoidal de frecuencia incremental, mientras que la

función *square-chirp* es una onda cuadrada de frecuencia incremental:

$$\text{sinechirp}(t) = \text{A} \cdot \text{sen}\left(\frac{2\pi f_0(r^t - 1)}{\log(r)}\right) \tag{4.19}$$

$$\text{squarechirp}(t) = \text{A} \cdot \text{signo}\left(\text{sen}\left(\frac{2\pi f_0(r^t - 1)}{\log(r)}\right)\right) \tag{4.20}$$

donde t es el tiempo, A es la amplitud de la señal, f_0 es su frecuencia inicial, y r es la tasa de cambio de la señal de frecuencia variable.

Una ventaja de las señales *chirp* es que, al usar una señal de frecuencia variable, en realidad estamos demandando diferentes velocidades a la plataforma, cosa que sólo podríamos probar testeando distintas amplitudes con frecuencia fija, lo cual requeriría muchas más pruebas.

Además de emplear la función *chirp*, analizaremos también los resultados de las señales senoidales para construir un diagrama de Bode del sistema.

4.4.2.1 – Resultados y Conclusiones

En las Figuras 4.16, 4.17 y 4.18 podemos ver la respuesta de la plataforma ante las señales *chirp*, para los 3 GdL de la plataforma de movimiento analizada.

Para estas pruebas, la tasa de cambio se estableció en 1.2 y la frecuencia inicial en 0.01 Hz. El comportamiento es prácticamente idéntico para todos los GdL.

Con la señal *chirp* senoidal podemos observar que la atenuación se mantiene baja para señales de periodo mayor de 1 segundo (frecuencia inferior a 1 Hz) y bastante aceptable (menos de un 35 %) hasta casi los 2 Hz. Sin embargo, el desfase se sitúa en valores cercanos a 90º para frecuencias de en torno a 1 Hz. Un desfase de 90º no es deseable para simulación. Para frecuencias de 2 Hz, el desfase se va al entorno de los 180º lo cual ya sería del todo inaceptable para la reproducción de movimiento. Aunque cada GdL presenta ligeras variaciones con respecto a los demás, éstas son mínimas debido a que estos efectos son consecuencia de la inercia de los motores al movimiento cosa que afecta casi por igual a todos los GdL.

Con la señal *chirp* cuadrada sucede algo parecido, pero al ser el movimiento mucho más brusco, para señales de sólo 0.5 Hz el desfase es ya de unos 90º. La ganancia, sin embargo, no sufre ninguna atenuación para señales de frecuencia inferior a 1 Hz, y ésta no es importante hasta que la frecuencia no supera los 2 Hz.

Por último podemos ver el diagrama de Bode del sistema en la Figura 4.19. En él, podemos ver de nuevo como el desfase para frecuencias de 1 Hz es ya de 90º, y para 2 Hz, de 180º. La magnitud, sin embargo, se reduce en menos de 6-7 dB para valores inferiores de 2 Hz. Ello quiere decir que la distorsión de fase es más importante que la distorsión de magnitud para este dispositivo.

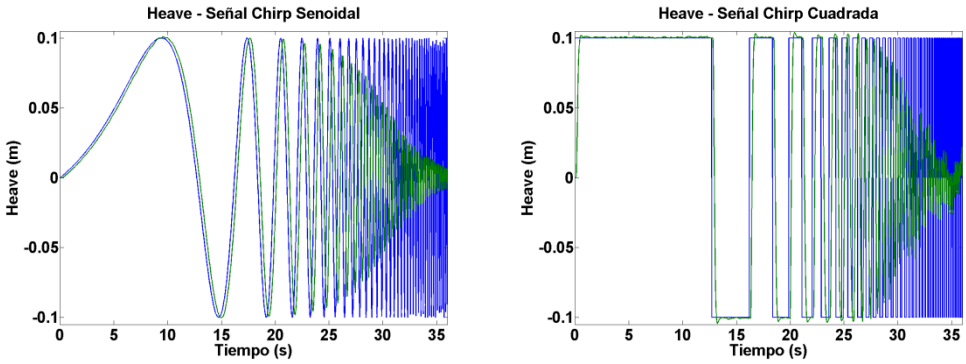

Figura 4.16 – *Comportamiento frecuencial para el movimiento de heave*

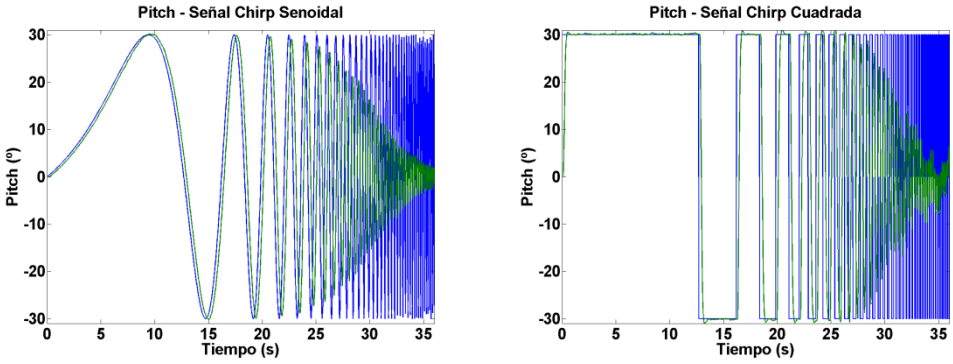

Figura 4.17 – *Comportamiento frecuencial para el movimiento de pitch*

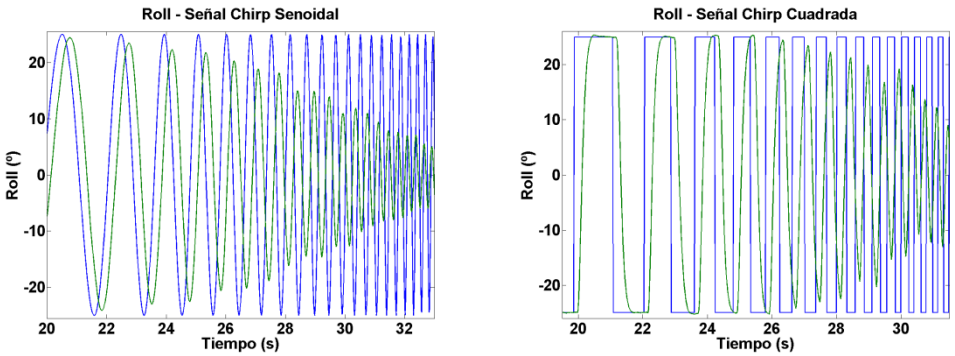

Figura 4.18 – *Comportamiento frecuencial para el movimiento de roll*

Sin embargo, hemos de tener en cuenta que es poco probable que necesitemos simular vehículos con un movimiento periódico de más de 2 Hz de frecuencia con grandes desplazamientos. Se pueden dar movimientos de alta frecuencia por la presencia de baches o golpes, pero estos movimientos serán de mucha menor amplitud, por lo que este resultado no debe alarmarnos en absoluto.

Con todo este análisis, podemos llegar la conclusión de que este dispositivo puede servirnos para simular excursiones completas cuya oscilación no supere 1 Hz. Como habitualmente no necesitaremos realizar una excursión completa, ya que los algoritmos de *washout* tienden siempre a volver a la posición central, es más habitual encontrarse con movimientos más pequeños, como por ejemplo una semi-excursión. En ese caso, la frecuencia de los movimientos puede ser mayor.

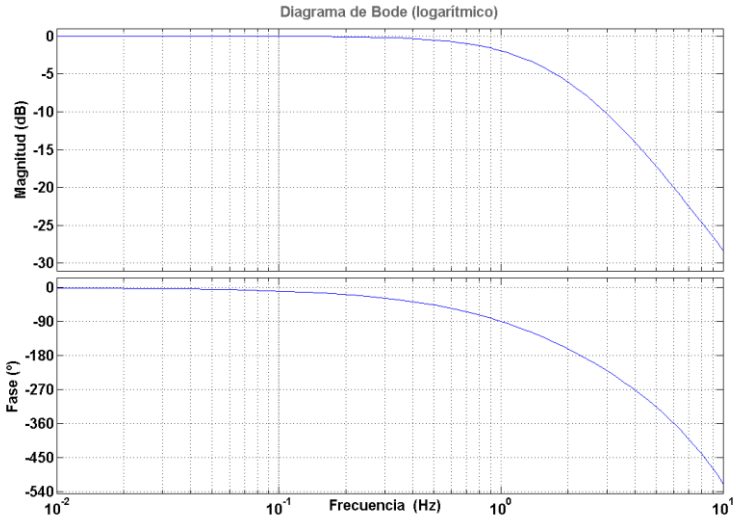

Figura 4.19 - *Diagrama de Bode (logarítmico)*

En el capítulo 3 analizamos los movimientos de un vehículo para su simulación y vimos que, para el vehículo analizado, el espectro de frecuencias por encima de 2-3 Hz no era muy amplio, cosa que no está muy lejos de lo que esta plataforma puede soportar, y aunque sería deseable que la plataforma fuera más potente, eso requeriría un gran coste económico. Además, hemos de tener en cuenta que una parte de ese movimiento vibratorio detectado no correspondía con desplazamientos grandes.

Por otro lado, también vimos que debíamos eliminar frecuencias menores de unos 3 Hz para poder simular los movimientos del vehículo en toda su amplitud. Si hiciéramos eso, como la plataforma actúa de filtro pasa-baja eliminando altas

frecuencias, perderíamos las frecuencias bajas por el algoritmo MCA y las altas por la plataforma, con lo cual, llegamos a la conclusión de que debemos renunciar a reproducir los movimientos en su amplitud original y debemos conformarnos con una versión reducida de los mismos, o de otro modo no podremos reproducir prácticamente ningún movimiento. Esto valida la estrategia de los algoritmos de *washout* de escalar las señales de entrada.

En cualquier caso, debemos tener varias cosas en cuenta. La primera es que, aunque la plataforma no pueda realizar excursiones completas a más de 2 Hz, esto no quiere decir que no podamos simular movimientos de más alta frecuencia si estos tienen menor amplitud (eso sí, a mayor frecuencia menor amplitud podrán tener estos movimientos). La segunda es que podemos apoyarnos en la técnica de *tilt-coordination* para simular las frecuencias bajas, y por tanto, no las perderemos completamente. Y la tercera es que, si realmente necesitamos simular movimientos bruscos con grandes desplazamientos, podemos necesitar unos recursos *hardware* superiores a los de utilizar el vehículo real que se desea simular, por lo que todo el proceso podría carecer de sentido.

4.5 – Conclusiones

En este capítulo hemos tratado de hacer un análisis exhaustivo de un manipulador robótico paralelo con el objetivo de utilizarlo para simulación de movimiento. La caracterización estática nos ha permitido observar cómo es la relación entre los distintos GdL, con lo que hemos podido comprobar que en

manipuladores paralelos la extensión de un GdL puede afectar al resto. Esto es de vital importancia a la hora de ajustar los parámetros del algoritmo MCA, y es uno de los motivos por los que dicho proceso no es sencillo. A partir de este análisis hemos sido capaces de identificar zonas seguras en las que evitaríamos tener que hacer un re-escalado de GdL cuando usemos los algoritmos MCA. Estas zonas seguras se pueden utilizar de forma explícita programando la plataforma para que no supere dichos límites, de forma implícita, intentando que el algoritmo MCA se mantenga siempre dentro de dichas zonas pero sin forzarle a permanecer en ellas, o se pueden simplemente tomar como simple referencia para el diseño o la construcción de las distintas plataformas de movimiento.

La caracterización dinámica nos ha permitido ver qué retrasos induce la plataforma y cómo influye la frecuencia de los movimientos en el comportamiento de la plataforma.

Sin embargo, la conclusión más importante que se puede extraer de este capítulo es la obtención un procedimiento para analizar dispositivos robóticos como generadores de movimiento. A excepción de las ecuaciones de la cinemática inversa, que son particulares de este dispositivo, el resto de análisis se pueden extrapolar a otros manipuladores, tanto serie como paralelos. Al hilo de este punto, la presentación de las ecuaciones de la cinemática inversa nos permitirá utilizar este dispositivo, y cualquier otro similar, sin necesidad de que el fabricante nos proporcione el *software* de generación de movimiento, ya que en el capítulo 2 hemos visto con detalle cómo construir el algoritmo de *washout* clásico y en este capítulo ya sabríamos traducir los

movimientos en forma de GdL a comandos para los motores. De lo único que no podríamos prescindir es del *software* de control de los motores, que el fabricante nos debe proporcionar siempre.

Capítulo 5

Métricas de Evaluación para MCA

A diferencia de otros campos de investigación de las ciencias de la computación, tales como el almacenamiento de la información, el análisis del coste computacional, la transmisión de información o la compresión de imágenes, los algoritmos de generación de claves gravito-inerciales carecen, en general, de mecanismos estandarizados o comúnmente aceptados de evaluación y comparación.

En este capítulo trataremos de proponer un sistema objetivo de evaluación para algoritmos de generación de claves gravito-inerciales, cuya correlación con la percepción humana se comprobará en el capítulo siguiente. El objetivo de este sistema de evaluación es fundamentalmente facilitar la tarea del ajuste de parámetros en este tipo de algoritmos. La presencia de una métrica debe permitir objetivar y acelerar esta costosa tarea. De hecho, si el ajuste se realiza por medio de aproximaciones tipo prueba-error con humanos, es posible necesitar muchas horas y las respuestas obtenidas pueden variar en función del cansancio, de las pruebas que se hayan realizado con anterioridad, de la persona, y de otros muchos factores muy difíciles de medir, que hacen que personas distintas (o incluso la misma en días diferentes) perciban el mismo algoritmo, incluso con los mismos parámetros, de forma diferente.

Además de para ajustar parámetros en un algoritmo, esta métrica puede servir también para comparar de manera no sesgada dos versiones (con diferentes valores para los mismos parámetros) de un mismo algoritmo, o dos algoritmos diferentes entre sí, aunque en este caso ambos deben estar correctamente ajustados para la tarea encomendada.

De hecho, uno de los mayores problemas que presenta este tipo de algoritmos es el de la comparación. La comparación de dichos algoritmos es algo complejo, ya que hay que plantearse cuánto esfuerzo se ha puesto en comparar "manzanas con manzanas", como se dice en [86]. Ello se debe a que la utilidad de un algoritmo puede degradarse o mejorarse mucho por un buen o mal ajuste y elección de parámetros. Como habitualmente este proceso se realiza mediante aproximaciones de tipo prueba-error con humanos, afirmaciones del tipo "el algoritmo A es mejor que el B" son afirmaciones peligrosas, ya que asumen implícitamente que se ha invertido el mismo esfuerzo en hacer que ambos lleguen a su máximo nivel de rendimiento, algo que no siempre es cierto. Incluso aunque se dedique el mismo tiempo al ajuste de los parámetros de ambos algoritmos, ello no quiere decir que se consiga llevar a ambos a su mejor versión. Por tanto, con ese tipo de pruebas, lo único que se puede afirmar es que el algoritmo A es mejor que el B con la elección particular de parámetros que hemos elegido, lo cual es una afirmación de poca utilidad. Ejemplos de esto se pueden ver en [159], [87] y [160] donde se investiga a fondo el algoritmo adaptativo y se compara con el clásico, pero no se detalla cuánto esfuerzo se ha dedicado a ajustar éste último.

Es por ello que, aunque encontrar métricas objetivas es difícil y no garantiza al 100% que el algoritmo vaya a resultar más agradable para todo el mundo, sí que es conveniente tener un baremo a partir del cual establecer referencias objetivas, porque ello permite, sin miedo a equivocarnos, establecer afirmaciones del tipo "el algoritmo A es mejor que el algoritmo B, bajo el criterio C" de modo que podemos emplear el mismo tiempo a encontrar la mejor versión (objetivamente) de cada uno de ellos, y entonces (y sólo entonces) realizar la comparación.

Dicho lo cual, siempre hay que tener en cuenta la naturaleza del problema y las implicaciones que ello genera. Los criterios objetivos que podamos establecer no han de tomarse como criterios inmutables, ya que pueden ser objeto de discusión, al igual que la validez de las opiniones subjetivas. Tanto es así, que las conclusiones que se pueden extraer de evaluaciones y comparaciones con personas también se deben tomar con cierta cautela, como bien comenta [67], ya que son subjetivas, y por tanto, susceptibles de discusión.

5.1 – Métricas Existentes

Como vimos en el apartado 2.3.6 existen en la literatura algunas aproximaciones objetivas a la evaluación de sistemas inerciales.

Las dos referencias mejor formuladas son el OMCT (Objective Motion Cueing Test) y la MPT (Motion Perception Toolbox).

OMCT es una parte del "Manual de criterios para la evaluación de dispositivos de entrenamiento en simulación aérea" [114]. Su primera limitación es que está pensado exclusivamente para aviones y vehículos aéreos, aunque podría extrapolarse a otros campos. El test consiste en la aplicación de una serie de señales de prueba (senoidal por ejemplo) para medir sobre la plataforma el retraso y la atenuación en amplitud que esta induce, de forma similar a lo que describimos en el punto 4.4. Las pruebas se realizan para distintas frecuencias y los resultados se muestran en forma de diagrama muy similar a los conocidos diagramas de Bode. Esto conduce a su segunda limitación: los resultados no sirven directamente para ayudar al ajuste de parámetros del algoritmo de movimiento ya que de los mismos no es fácil deducir si un algoritmo es mejor que otro, puesto que no se obtiene un valor global de rendimiento y no es sencillo utilizar la información proporcionada de forma automatizada.

En cuanto a MPT [70] es una biblioteca de módulos que, mediante modelos de percepción, intenta describir cómo ha sido la percepción del movimiento por parte del sujeto que experimenta la simulación. Por desgracia, dicha biblioteca ya no está a libre disposición y ha pasado a ser un producto comercial recientemente por lo que no nos ha sido posible probarla, aunque en principio sí que permitiría generar valores objetivos que puedan ser usados para el ajuste de parámetros, aunque no en forma numérica única. Sin embargo todo el funcionamiento de la MPT se basa en modelos de percepción cuya validez es, a veces, discutible. Aunque los autores de la MPT son conscientes que "la enorme variedad de datos experimentales (ambiguos), modelos fisiológicos y opiniones de expertos hacen imposible desarrollar una biblioteca de

percepción universal", sí que parece un buen intento de obtener medidas objetivas.

En nuestra opinión, estos y otros trabajos se aproximan al problema de la evaluación de algoritmos MCA de manera apropiada, pero les faltan dos elementos fundamentales. El primero es realizar una caracterización sobre cómo es la respuesta de los pilotos ante algoritmos MCA para validar las medidas propuestas, cosa que nosotros haremos en el próximo capítulo. La percepción es mucho más que un modelo de otolito o de canal semicircular o de flujo óptico. Existen muchas contribuciones sensoriales y creemos que es muy complicado generar un modelo de percepción que incluya todos los sistemas que el cuerpo emplea para la misma. La segunda es poder utilizar las medidas objetivas de modo sencillo y que nos sirvan no sólo para evaluar los algoritmos sino para compararlos y ajustar sus parámetros. Aunque en algunos trabajos los resultados son objetivos, no vienen en una forma numérica única que pueda ser sencilla de entender y utilizar en sistemas de ajuste de parámetros. Por estas razones, vamos a intentar en este capítulo y en el próximo, proponer medidas de evaluación que solventen estos dos problemas.

5.2 – Hipótesis

Aunque de forma distinta, todos los algoritmos de *washout* funcionan bajo el mismo principio básico. Este principio consiste en intentar generar los movimientos requeridos por el vehículo simulado y al mismo tiempo evitar por todos los medios que la reproducción de esos movimientos implique superar los límites

físicos impuestos por las restricciones espaciales de la plataforma. Como hemos comentado con anterioridad, esto es, en general, imposible, por lo que se necesita llegar a un compromiso entre ambos objetivos contrapuestos: fidelidad en los movimientos y restricción de los mismos al espacio disponible.

Para ello, como ya comentamos, este tipo de algoritmos recurre a dos técnicas básicas: el escalado y el filtrado en frecuencia. La primera técnica consiste en reducir la intensidad de las magnitudes físicas que se intentan reproducir. Dado que no es, en general, posible generar las aceleraciones deseadas, se intentan reproducir aceleraciones similares (en forma) pero a menor escala.

Como simplemente escalando la entrada no vamos a conseguir siempre el objetivo deseado (pensemos por ejemplo en una aceleración pequeña pero constante que acabaría tarde o temprano por provocar que la plataforma llegara a sus límites), la segunda técnica consiste en limitar el espectro de frecuencias de los movimientos reproducidos para evitar desplazamientos largos que impliquen posiciones u orientaciones a las que no se puede llegar. Este principio se consigue aplicando filtros pasa-alta que eliminan las aceleraciones de baja frecuencia que son las que producen movimientos largos y sostenidos (y que evidentemente son los que hacen que la plataforma alcance sus límites).

Además de estas dos técnicas básicas, los algoritmos de *washout* también recurren a ciertas estrategias que se aprovechan de las ilusiones del sistema vestibular humano, como la técnica del *tilt-coordination* y la ilusión somatográvica, que ya hemos comentado con anterioridad. Estas pequeñas manipulaciones, y el hecho de

que escalemos y filtremos las señales pueden provocar sensaciones falsas (en inglés *false cues*) que pueden confundir al piloto, ya que para reproducir un cierto efecto en un GdL podemos generar sensaciones falsas en otro.

Por tanto, en general, las señales de salida del algoritmo se parecerán a las señales de entrada pero no serán iguales. Tomando como ejemplo el algoritmo clásico, se toman como entradas 6 señales (aceleración en sus 3 ejes, y velocidad angular en sus 3 ejes). Éstas 6 señales sufren escalados, filtros y diversos procesamientos hasta llegar a generar las salidas del algoritmo, que se corresponden con los 6 GdL de la plataforma. Aunque las señales de salida del algoritmo no son propiamente comparables con las entradas (ya que las salidas no son aceleraciones lineales y velocidades angulares sino posiciones de los GdL), los actuadores hacen mover a la plataforma y podemos medir sobre ella las mismas 6 señales que se aplicaron a la entrada para comprobar si se parecen a las de la entrada.

Dado que estos algoritmos escalan las señales, las hacen pasar por filtros pasa-alta (que deforman la señal), algunas las hacen pasar por filtros pasa-baja (que retrasan y deforman la señal), y después la propia plataforma actúa de filtro pasa-baja (que también retrasa las señales ya que tiene ciertos límites mecánicos que le impiden alcanzar altas frecuencias de movimiento), las señales de salida serán similares a las de entrada, pero serán, en general, de menor intensidad, estarán retrasadas una cierta cantidad de tiempo respecto a ellas, y en algunos casos incluso no se parecerán mucho a ellas.

Es por ello, que podemos suponer con cierta seguridad que para que un algoritmo de tipo MCA funcione correctamente debe intentar minimizar o maximizar (según el caso) estas 3 características: similitud, retraso y escalado. Partiendo de la hipótesis de que los usuarios serán sensibles a estos elementos, buscaremos métricas que representen estas características, y posteriormente comprobaremos si los usuarios son o no realmente sensibles a ellas.

5.3 – Elección de Métricas

En esta sección, abordaremos el problema de evaluar la similitud entre dos señales. Dado que los algoritmos MCA toman como entrada una serie de señales que representan ciertos parámetros físicos como aceleración, velocidad, etc., una manera objetiva de comparar cuán satisfactoria es su ejecución es comparar las entradas del algoritmo respecto de sus salidas. La manera de comparar dichas señales puede variar, dado que podemos comparar una señal de entrada del algoritmo (por ejemplo la velocidad angular en el eje X) con respecto de la generada por la plataforma, o podemos compararla con respecto de la sentida por el usuario que la sufre (recordemos que el cuerpo humano dispone de sensores con ciertas imperfecciones y no se siente lo mismo que ocurre). Es evidente que, si la magnitud de entrada y la magnitud de salida son iguales, dicha magnitud será percibida exactamente igual que en la realidad. Como ello no es posible, a veces se suele comparar la magnitud de entrada con la magnitud de salida sensorizada (es decir, pasada a través de algún modelo de percepción humano). Lo malo de los modelos de

percepción humanos es que, como comentamos con anterioridad, no existe ningún consenso al respecto de los mismos. Se han publicado decenas de ellos e incluso algunos trabajos se contradicen entre sí [83], [67]. Además, es complicado saber cuál debe ser la extensión del modelo: ¿debe incluir sólo el sistema vestibular?, ¿debe incluir también un modelo del sistema visual?, ¿debe ser un modelo completo? Dado que no es conocido con demasiada certeza el alcance, la relación y la aplicabilidad de los diversos modelos, éste es un aspecto que se debe tratar con cuidado.

De la elección de magnitudes, de modelos de percepción y de combinación de señales hablaremos más adelante. Ahora vamos a ver cómo podemos simplemente comparar dos señales físicas variables en el tiempo mediante una métrica para saber cuán parecidas son.

Por conveniencia haremos que las métricas varíen en el rango [1.0, +∞), de tal modo que 1 corresponda al valor óptimo y cualquier número superior indique peor evaluación. El objetivo será por tanto minimizar el indicador. Dejamos el intervalo abierto por la derecha a propósito, ya que consideramos que existe un óptimo (cuando sean iguales) y sin embargo no existe un caso que podamos considerar categóricamente como el peor.

5.3.1 – Error Cuadrático Medio

Una de las maneras más habituales de comparar dos señales es calcular la suma de diferencias entre ambas señales. Como las

diferencias a veces son positivas y a veces negativas, la simple suma de diferencias podría llegar a cancelarse, por lo que lo habitual es calcular la suma de diferencias al cuadrado.

El error cuadrático diferencial (ECD) se calcula como la suma de los cuadrados de las diferencias que existen entre dos señales. Es decir:

$$ECD(x,y) = \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (5.1)$$

siendo *x* la señal 1 (la original o entrada), *y* la señal 2 (la reproducida o salida), y *n* el número de muestras que tenemos de ambas señales.

De igual modo que se calcula el cuadrado de la diferencia, se puede calcular el valor absoluto de la diferencia. El uso del cuadrado evita tener que preocuparnos por el signo de la diferencia.

Este indicador, sin embargo, tiene algunos inconvenientes a la hora de comparar señales. El primero y más importante de ellos es que cuanto más largas son las señales, más improbable es que el valor del indicador se acerque a 0.0 ya que cualquier pequeña diferencia se va acumulando al resultado final.

El segundo inconveniente es que no tiene en cuenta la magnitud de las señales comparadas. No es lo mismo una diferencia de 1 unidad en la comparación de dos señales cuyo pico

es 2 unidades, que esa misma diferencia cuando las señales son del orden de 100 unidades.

Una manera sencilla de resolver el primero de los problemas anteriores es realizar un promedio sobre todas las diferencias calculadas. Es decir, calcular el error cuadrático diferencial promedio, también llamado error cuadrático medio.

El error cuadrático medio (ECM) se calcula como el promedio de las diferencias al cuadrado que existen entre dos señales. Es decir:

$$ECM(x, y) = \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (5.2)$$

Dado que hemos decidido que las métricas estén en el rango [1.0, +∞), este indicador tal y como está formulado no nos sirve, ya que si las señales son iguales devuelve 0.0. Por ello, proponemos como indicador comparativo el siguiente:

$$ECM(x, y) = 1 + \frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (5.3)$$

De este modo, si las señales son iguales, el resultado del indicador es 1, y cuanto más diferentes sean, mayor es el valor. A veces, se calcula la raíz cuadrada del resultado del ECM para obtener un valor más representativo para el error medio, sin que las magnitudes estén elevadas al cuadrado.

Una posible alternativa para evitar el cálculo de la raíz cuadrada es calcular el error medio de la diferencia en valor absoluto (al que llamaremos error absoluto medio, EAM). Si bien el cálculo no es exactamente el mismo, el indicador es bastante similar:

$$EAM(x,y) = 1 + \frac{1}{n} \cdot \sum_{i=1}^{n} |x_i - y_i| \qquad (5.4)$$

Este indicador (en cualquiera de sus dos formas), sin embargo, todavía presenta el segundo inconveniente que comentamos: no tiene en cuenta la magnitud de las señales en la comparación.

5.3.2 – Error Cuadrático Medio Normalizado

Para intentar evitar el problema de la comparación de señales de distintas magnitudes, el cálculo se puede reformular normalizando el indicador para intentar tener en cuenta la magnitud de las mismas.

Hay varias formas de realizar dicha normalización. Una de ellas es dividir la diferencia por el rango válido de la primera o de la segunda señal. Como a veces las señales no están restringidas a ningún rango conocido, a veces se divide por el valor pico (máximo o mínimo) o por la diferencia mayor entre picos (*peak-to-peak)*. Dado que nuestras señales no están limitadas a ningún rango conocido, lo que haremos es dividir por el valor máximo en valor absoluto de la primera señal. Igual que antes, modificaremos el

indicador para que esté en el rango [1.0, +∞) y obtendremos un indicador que llamaremos error cuadrático medio normalizado (ECMN):

$$ECMN(x,y) = 1 + \frac{1}{n \cdot |x_p|} \cdot \sum_{i=1}^{n} (x_i - y_i)^2 \qquad (5.5)$$

donde x_p es el valor de pico.

Al igual que antes, podemos usar el valor absoluto y tendríamos otro indicador similar pero distinto que llamaremos error absoluto medio normalizado (EAMN):

$$EAMN(x,y) = 1 + \frac{1}{n \cdot |x_p|} \cdot \sum_{i=1}^{n} |x_i - y_i| \qquad (5.6)$$

Con este pequeño cambio, lo que nos aseguramos es que las diferencias se midan de modo relativo y no de modo absoluto, por lo que de alguna forma estamos calculando el porcentaje de error promedio que la salida tiene respecto de la entrada.

5.3.3 – Escalado Medio

Uno de los objetivos de un algoritmo de generación de claves gravito-inerciales es generar una señal similar a la real. Lo ideal es que la señal de salida sea igual en forma y magnitud a la original, por lo que un indicador para evaluar este tipo de algoritmos podría ser el cociente entre la señal deseada y la

generada. Cuanto más grande sea este cociente peor, y si el cociente es igual a 1, entonces las señales son iguales, que es lo óptimo. Por ello, podríamos pensar que un buen indicador sería el escalado medio entre las señales:

$$EM(x,y) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{x_i}{y_i} \tag{5.7}$$

Aunque lo normal es que segunda señal (la señal simulada) esté por debajo de la primera, porque normalmente los algoritmos MCA tienen que reducir la intensidad de las señales ante la imposibilidad de reproducir las señales de origen sin alcanzar los límites de la plataforma, es posible que, alguna vez, la segunda señal esté por encima de la primera (por la aplicación de filtros y por los retrasos que se producen). Si esto pasa, el indicador, tal y como está formulado devolvería valores por debajo de 1, lo cual viola nuestro rango válido. Por ello, proponemos la siguiente formulación:

$$EM(x,y) = \begin{cases} \dfrac{K}{c(x,y)} - (1-K) & \text{si} \quad c(x,y) < 1 \\ 1 & \text{si} \quad c(x,y) = 1 \\ K \cdot c(x,y) + (1-K) & \text{si} \quad c(x,y) > 1 \end{cases} \tag{5.8}$$

con

$$c(x,y) = \frac{1}{n} \cdot \sum_{i=1}^{n} \frac{x_i}{y_i}$$

siendo *K* una constante real positiva, que habitualmente (y por defecto) valdrá 1 si queremos que simplemente haga el cociente, pero podría tener otros valores si queremos ponderar el cociente de otra forma (tanto hacia arriba como hacia abajo).

De este modo, el indicador se mantiene en el rango [1.0, +∞). Si la primera señal es superior a la segunda, el indicador será proporcional al escalado que haya entre ellas. Y si por el contrario, la primera es superior a la segunda, el indicador también será proporcional (en este caso inversamente proporcional) al escalado que haya entre ellas. Ello quiere decir, que (con $k = 1$) si el cociente promedio entre señales es 2 el indicador valdría 2, y si el cociente promedio es 0.5, el indicador también valdría 2.

Es importante reseñar que el cálculo del cociente presenta ciertos casos especiales cuando el denominador es 0, ya que se produce una indeterminación. Por ello, hay que tratar las indeterminaciones y resolverlas. Por tanto, si se da el caso en el que y_i alcanza un valor cercano a 0 en alguna parte de la señal, ese muestreo de la señal no se tendrá en cuenta para el cálculo del indicador, ya que:

$$\lim_{y \to 0}(x/y) = +\infty \quad si\ x > 0$$
$$\lim_{y \to 0}(x/y) = -\infty \quad si\ x < 0$$

5.3.4 – Correlación

Dado que el objetivo de un algoritmo de generación de claves gravito-inerciales es generar una señal similar a la real, y por lo general no se puede generar una señal de la misma magnitud, lo que se intenta es que la señal simulada sea parecida en forma, aunque sea de una magnitud inferior a la real. Existen varios indicadores matemáticos capaces de proporcionar información sobre este hecho, aunque el más empleado es la correlación de Pearson. El coeficiente de correlación de Pearson (o simplemente la correlación, sin adjetivo) es una medida de cuán significativa es la relación lineal entre dos magnitudes aleatorias. Es decir, si aplicamos la correlación a dos señales, nos dirá cuán parecidas son esas dos señales en forma pero no en magnitud. La medida se evalúa con un valor entre -1 y 1. Un valor 1 indica una correlación total (la señal 1 es igual en forma a la señal 2), y un valor -1 indica una correlación inversa, que quiere decir que la señal 1 tiene justamente la forma contraria a la señal 2.

Para ejemplificar cómo funciona este indicador, la correlación de una señal *sen(x)* con respecto a otra *sen(3x)* daría como resultado un valor de 1, ya que ambas señales son iguales en forma, aunque no lo sean en magnitud. Por el contrario, la correlación de la señal *sen(x)* y la señal *–sen(x)* da como resultado un valor de -1, ya que ambas señales son opuestas.

Existen varias maneras de calcular el coeficiente de correlación, aunque todas son equivalentes. Una posible formulación es esta:

$$CCP(x,y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad (5.9)$$

con

$$\bar{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

$$\bar{y} = \frac{1}{n} \cdot \sum_{i=1}^{n} y_i$$

Este indicador, como hemos dicho, devuelve valores en el rango [-1, 1], que no es el rango que nosotros deseamos. Por tanto, para transformar este indicador a nuestra escala de medidas, emplearemos la siguiente formulación:

$$CCPN(x,y) = \frac{K}{1 + CCP(x,y)} + 1 - \frac{K}{2} \qquad (5.10)$$

siendo K una constante real positiva (que por defecto tendrá valor 1) y que lo que hace es cambiar la forma en la que el indicador tiende hacia $+\infty$ cuando la correlación se aleja de valores cercanos a 1.

5.3.5 – Retraso Estimado

Cuando se ejecuta un algoritmo de tipo MCA, lo habitual es que la señal de salida se parezca a la deseada, pero ligeramente retrasada. Esto es consecuencia de que la plataforma de

movimiento es un sistema mecánico que actúa involuntariamente como un filtro de frecuencias pasa-baja que retrasa la señal. Además, existen otros elementos que inevitablemente retrasan la salida (transmisión, cálculos, etc.). Es por ello que sería deseable poder medir el retraso entre la señal deseada y la generada, porque ello podría ser un buen indicador de cuán bueno es el algoritmo.

Sin embargo, el retraso entre dos señales es muy difícil de medir si ambas señales no son del mismo tipo. Si ambas son del mismo tipo, como por ejemplo *sen(x)* y *sen(x+a)*, entonces se puede observar a simple vista, aunque en cualquier caso, sigue sin ser evidente cómo calcularlo de forma automática. Si ambas señales se parecen poco, o simplemente no se parecen mucho, el cálculo no es nada evidente.

Dado que el cálculo del retraso no es en absoluto evidente, nuestra propuesta es calcular (aunque deberíamos decir estimar ya que calcularlo no es posible en ningún caso) el retraso como el *offset* que hay que provocarle a la señal 1 (la original) para que su correlación con la señal 2 (la generada) sea máxima dentro de un determinado lapso razonable de tiempo.

Es de esperar que la maximización de la correlación entre ambas señales se produzca cuando apliquemos un *offset* bastante bajo (del orden de décimas de segundo o pocos segundos) a la primera de ellas (si el *offset* tiene que ser muy alto es que la simulación de movimiento es completamente desastrosa), por lo que proponemos el siguiente algoritmo para el cálculo del retraso estimado (RE):

Algoritmo **CalcularRetraso**
// estima el retraso entre dos señales mediante el uso de la correlación

Entradas:
x : vector [1..n] de R *// señal 1*
y : vector [1..n] de R *// señal 2*

Parámetros:
maxRetraso, tolerancia : R

Auxiliar:
retraso, corr, mejorCorr : R
xRetrasada : vector [1..n] de R

Salidas:
retrasoProbable : R

Algoritmo CalcularRetraso:

```
retrasoProbable = 0.0;
mejorCorr = -1.0;
retraso = -maxRetraso;

mientras (retraso ≤ maxRetraso ) hacer
{
   xRetrasada = retrasarSeñal(x, retraso);
   corr = calcularCorrelacion(xRetrasada, y);

   si (corr > mejorCorr) entonces
   {
      retrasoProbable = retraso;
      mejorCorr = corr;
   }

   retraso = retraso + tolerancia;
}
```

Fin de CalcularRetraso

Evidentemente esto es una aproximación, ya que un cálculo fiable del retraso implicaría ser capaces de detectar qué parte de cada señal corresponde con qué parte de la otra, y eso no es sencillo. Por eso a este indicador le llamamos Retraso Estimado (RE). Nótese que en el cálculo del retraso también tenemos en cuenta retrasos negativos, es decir adelantos, ya que puede darse el caso de que alguna señal, se adelante a su entrada como efecto colateral de alguna falsa clave gravito-inercial.

El valor final del indicador RE será:

$$RE(x, y) = 1 + \partial(x, y) \tag{5.11}$$

donde $\partial(x, y)$ representa el valor absoluto de *offset* que necesitamos aplicar a la señal x (la original) para maximizar la correlación de Pearson con respecto a la señal y (la generada por la plataforma de movimiento).

5.4 – Combinación de Métricas

Todas las métricas que hemos visto anteriormente se corresponden únicamente con comparaciones entre dos señales. Podrían servir, por tanto, para comparar cualesquiera dos señales, independientemente de lo que representen.

En nuestro caso, el objetivo es lograr un indicador que nos diga cuán bueno es un algoritmo a la hora de generar las claves gravito-inerciales deseadas. Por lo tanto, para conseguir ese objetivo necesitamos elegir las métricas adecuadas (de entre las propuestas) y después combinarlas de alguna forma.

Como el cuerpo humano es sensible a la fuerza específica y a la velocidad angular, en principio lo lógico es emplear métricas que representen comparaciones de las 6 señales (F_x, F_y, F_z, ω_x, ω_y, ω_z) correspondientes a esas magnitudes físicas.

El primer paso para ello es conseguir calcular un indicador único por cada una de esas 6 magnitudes que el cuerpo es capaz de sensorizar. Dado que hemos presentado varias métricas deberemos elegir cuál usamos o si usamos varias, saber cómo combinarlas. A priori es difícil saber qué se aprecia más, si el retraso, la correlación, la diferencia, etc., ni tampoco es fácil saber cómo combinarlos. De la elección de estas métricas hablaremos en el capítulo siguiente, ya que nuestra propuesta es que la elección y/o ponderación de métricas sea realizada en base a un estudio subjetivo con pilotos humanos.

Una vez obtenido un indicador único por cada GdL, deberemos combinar esos 6 indicadores en un solo indicador. De nuevo, es difícil saber qué GdL son más importantes. De hecho, dependiendo del tipo de simulador, unos serán más relevantes que otros. En un simulador puede ser importante la aceleración frontal y en otro puede ser muy importante la velocidad angular sobre el eje Z.

Además, tampoco es sencillo saber si las rotaciones son más importantes que las traslaciones o si la combinación debe ser aditiva o multiplicativa. Esto es importante porque aunque algunos indicadores no tienen unidades, otros pueden tenerlas. Por ejemplo, si medimos la diferencia entre señales, la combinación entre aceleraciones y velocidades angulares no estaría clara en términos de combinación de unidades, ya que la diferencia entre aceleraciones se mide en m/s^2, mientras que la diferencia entre velocidades angulares se mide en º/s. La combinación de ambas unidades no es nada evidente.

5.4.1 – Combinación de Diferentes GdL

En cuanto a la combinación de los diferentes GdL, durante las pruebas realizadas con pilotos en diversos simuladores, sus valoraciones y comentarios parecen indicar que en un simulador con generación gravito-inercial lo importante es no generar ninguna clave incorrecta, en lugar de generar varias bien y una mal. Muchos pilotos critican más los algoritmos cuando un grado de libertad está muy mal simulado, que cuando todos están simplemente aceptablemente simulados. Dicho de otro modo, partiríamos de la hipótesis de que la eliminación de claves falsas es casi tan importante, o más, que la generación de claves correctas, ya que el cuerpo humano parece ser capaz de sentir cuando algo "no encaja", de forma bastante acusada. Es por ello, que la combinación de métricas multiplicativa puede ser adecuada. Dado que las métricas individuales son números mayores que 1, si alguna de las 6 está mal generada recibirá un indicador global alto. Si la combinación es multiplicativa, el indicador arrastrará el valor alto y

el indicador global será malo. Una ventaja añadida es que la combinación multiplicativa permite combinar indicadores de diferentes unidades, ya que aunque sumar magnitudes de diferente tipo puede ser extraño, multiplicarlas sí puede tener sentido, dando lugar a nuevas unidades.

Una alternativa a esta combinación sería la combinación por ponderación aditiva. Esto se correspondería con la hipótesis de que lo importante es generar el mayor número de claves correctas aun cuando podamos generar alguna falsa. En este caso, la métrica final corresponderá a la suma ponderada de cada una de las métricas individuales. El problema fundamental de este tipo de combinación es ser capaces de ponderar diferentes indicadores de diferentes GdL, que, además, pueden venir expresados en unidades diferentes. Para la combinación aditiva, el indicador ECM no nos serviría, ya que los grados de libertad rotacionales se miden en º/s y los traslacionales en m/s^2. Sin embargo, todos los demás indicadores podrían emplearse y combinarse aditivamente. El retraso se mide siempre en segundos, independientemente del grado de libertad, mientras que la correlación, el escalado medio y el error cuadrático normalizado son indicadores adimensionales, con lo cual, el problema de la combinación aditiva no es tan grave, ya que el único problemático es el error cuadrático medio.

Por todo ello, y para poder elegir la métrica adecuada para cada simulador, proponemos un sistema flexible de combinación de métricas que funciona de la siguiente forma. Por cada una de las 6 magnitudes anteriormente citadas, calcularemos un indicador (ya veremos en el capítulo siguiente que métricas elegir y cómo combinarlas). Posteriormente combinaremos los indicadores de

cada GdL de forma parametrizable, de modo que se pueda elegir entre combinación multiplicativa o aditiva. Adicionalmente, en el caso de combinación aditiva se podrá ponderar cada uno de los GdL individualmente.

De forma matemática, la combinación multiplicativa se correspondería con la siguiente fórmula:

$$I = \prod_{i=1}^{6} I_i \tag{5.12}$$

Donde I_i correspondería con el indicador calculado para cada uno de los 6 grados de libertad, mientras que I sería el indicador global.

De forma análoga, la combinación aditiva se correspondería con la siguiente fórmula:

$$I = \sum_{i=1}^{6} p_i \cdot I_i \tag{5.13}$$

En este caso, además, la contribución de cada uno de los indicadores al indicador global es ponderado por unos pesos p_i.

Será, en cualquier caso, en capítulos siguientes cuando investiguemos cuál de los métodos de combinación resulta más adecuado, ya que comprobaremos qué métricas son más adecuadas y al mismo tiempo veremos qué tipo de combinación (tanto a nivel

de cada GdL como a nivel global) podría resultar más adecuada para representar la sensación de los pilotos.

5.5 – Conclusiones

En este capítulo hemos propuesto métricas para hacer posible la evaluación objetiva de algoritmos MCA. Previamente, hemos repasado algunas de las pocas propuestas que existen sobre el tema, que vimos también con cierto detalle en el capítulo 2.

La construcción de métricas se basa en una hipótesis razonable, que consiste en asumir que las señales generadas por el conjunto plataforma de movimiento más algoritmo MCA serán similares en forma a las reales, estarán ligeramente retrasadas en el tiempo, y posiblemente escaladas en amplitud (casi con total seguridad reducidas).

A partir de dicha hipótesis formulamos 5 métricas para medir la verosimilitud de cada una de las 6 señales que la plataforma de movimiento genera (F_x, F_y, F_z, ω_x, ω_y, ω_z) con respecto a las originales, que intenta reproducir. Estas 6 señales se corresponden con los 6 GdL del espacio.

Las 5 métricas seleccionadas son: error cuadrático medio (ECM), error cuadrático medio normalizado (ECMN), escalado medio (EM), correlación normalizada (CCPN) y retraso estimado (RE). Las dos primeras pueden sustituirse alternativamente por el error absoluto medio (EAM) y el error absoluto medio normalizado (EAMN), respectivamente.

Dado que el objetivo es generar una métrica única, proponemos a continuación dos formas de combinar las 6 métricas que resultan del análisis de las 6 señales estudiadas: una combinación aditiva y una combinación multiplicativa. La combinación aditiva se basa en la idea de linealidad, mientras que la combinación multiplicativa se basa en la idea de que el cerebro rechace como válida la reproducción de un movimiento si alguno de los 6 GdL está mal simulado.

Será en el siguiente capítulo cuando analicemos cuál(es) de las 5 métricas son válidas y cuál es la mejor forma de combinar las 6 señales resultantes para cada una de las 5 métricas propuestas.

Capítulo 6

Caracterización de la Respuesta Humana ante Algoritmos de Tipo MCA

El uso de plataformas de movimiento con simuladores de vuelo es bastante común. De hecho, los algoritmos MCA fueron originalmente diseñados y probados con simuladores de vuelo, y gran parte de la investigación sobre generación simulada de movimiento ha sido realizada por instituciones dedicadas al estudio aeroespacial.

Como ya comentamos, la simulación de aeronaves es diferente de la simulación marina o terrestre. Los aviones circulan sobre el aire, que es un fluido muy poco denso, por lo que la resistencia al movimiento es relativamente baja en comparación al mar o a la tierra, donde la fricción es más alta, especialmente en el mar. Por ello, las aceleraciones son más elevadas en el aire y normalmente sostenidas (de baja frecuencia) excepto por las turbulencias. Además, los efectos gravitacionales se notan menos en el aire, ya que los vehículos marinos y terrestres golpean constantemente la superficie del suelo o del agua. Ello hace que las aceleraciones verticales de los vehículos no aéreos sean más bruscas pero de menor duración.

En cualquier caso, los MCA tienen una especificación genérica que debe ser aplicable a cualquier tipo de simuladores. Para continuar con nuestro estudio sobre vehículos no aéreos, en este capítulo vamos a intentar caracterizar cómo es la respuesta humana ante simuladores de vehículos no aéreos. De esta forma podremos entender qué indicadores de los calculados en el capítulo anterior se ajustan mejor a lo que los usuarios de dichos algoritmos experimentan.

6.1 – Pruebas

Los dos casos de estudio elegidos para la caracterización son un simulador de conducción y un simulador de bote de rescate. Para el simulador de conducción elegimos *rFactor*, que es un *software* muy maduro para la simulación de competiciones deportivas. Para el simulador de bote de rescate escogimos nuestro simulador, convenientemente descrito en el capítulo 3.

La unión entre ambos simuladores y la plataforma de movimiento se realizó mediante la implementación de un pequeño *plug-in* de desarrollo propio, que es capaz de capturar el estado dinámico del vehículo (los valores de aceleración, velocidad, etc.) y enviarlo al algoritmo de *washout* para que éste intente reproducir dicho estado dinámico. El algoritmo de *washout* empleado para las pruebas es el algoritmo clásico en su versión de Reid y Nahon (ver capítulo 2). El sistema de referencia es el descrito en los capítulos 1 y 3.

Ambos simuladores emplean una pantalla de 270º con 3 proyectores. La resolución completa de la misma es 5840 x 1080 píxeles, más que suficiente para generar estímulos visuales inmersivos. En el caso del simulador de coche, el puesto de simulación emplea un volante *Logitech G27 Racing Wheel* con cambio de marchas y pedales (ver Figura 6.1) para reproducir el sistema de interacción. El salpicadero se representa de manera virtual. En el caso del simulador de bote, el puesto del piloto es un puesto real de conducción de bote de rescate, sensorizado para la ocasión (ver Figura 6.2). En este caso, no existe salpicadero porque este tipo de embarcaciones no suelen disponer de él.

Figura 6.1 - *Simulador de conducción*

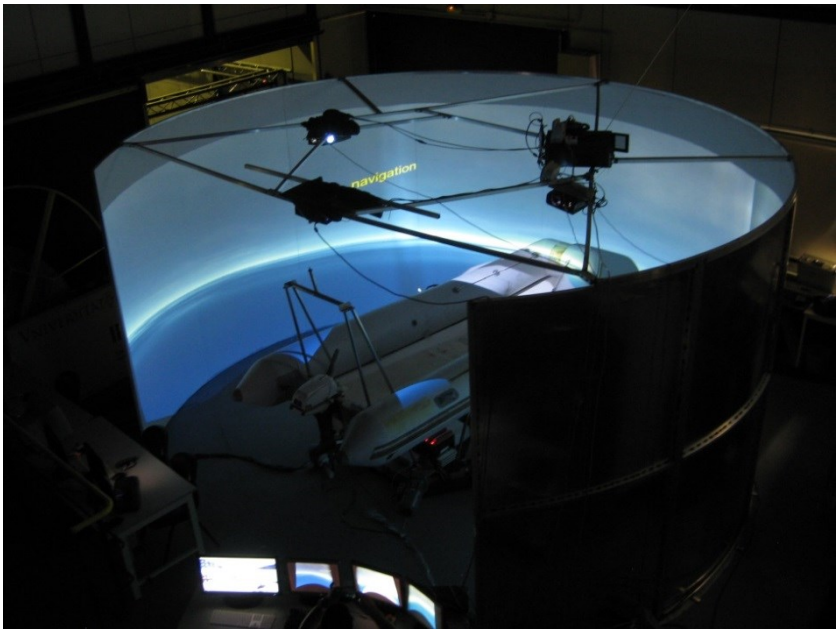

Figura 6.2 - *Simulador de bote de rescate*

6.1.1 – Descripción de los Manipuladores

Para obtener datos menos dependientes de la plataforma y, al mismo tiempo, evaluar la influencia de la misma en la reacción de los pilotos a algoritmos MCA, las pruebas fueron realizadas con dos plataformas de movimiento diferentes. Ambas eran manipuladores paralelos con motores rotacionales.

El primer dispositivo es una plataforma de 3 GdL del mismo tipo que la descrita en el capítulo 4, sólo que con otros parámetros de construcción. El otro dispositivo es una plataforma de 6 GdL de tipo Stewart, cuyo diseño se puede observar en la Figura 6.3.

La plataforma de 6 GdL emplea 6 motores, mientras que la de 3 GdL, como vimos, sólo emplea 3, por lo que es mucho más barata de construir, ya que tanto los motores como los controladores de los mismos son, con mucho, la parte más cara de un diseño de este tipo. Si esta inversión extra merece la pena o no en términos de presencia y percepción humana es algo que tendremos que analizar.

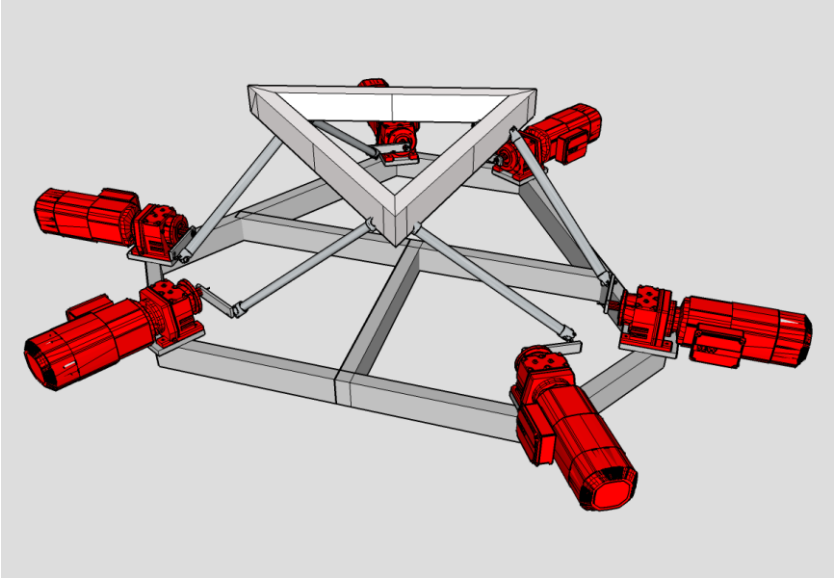

Figura 6.3 - *Modelo CAD de la plataforma de 6 GdL*

Ambos dispositivos emplean el mismo tipo de motores, pero al ser diseños distintos presentan características diferentes. Las Tablas 6.1 y 6.2 muestran dichas características. Para la medición de las mismas se emplearon medidores de seguimiento ópticos (*tracking* óptico) con cámaras de Natural Point [153], como se

puede observar en la Figura 6.4. Los valores de aceleración y velocidad se muestran de modo aproximado ya que son relativamente variables porque dependen del rendimiento de los motores, del sistema de control (y de cómo se ajusten los parámetros del mismo), de la carga, y de algunos otros factores. Además, la brusquedad de algunos de los movimientos de la plataforma hace que dichos valores sean difíciles de medir con mucha precisión.

	Desplaz. máximo	*Desplaz. mínimo*	*Aceleración Máxima*	*Velocidad máxima*
Heave	-14.6 cm	16.4 cm	0.5 g	40 cm/s
Pitch	-30.5°	30.5°	400 °/s^2	50 °/s
Roll	-26.2°	26.2°	400 °/s^2	50 °/s

Tabla 6.1 - *Características de la plataforma de 3 GdL*

	Desplaz. máximo	*Desplaz. mínimo*	*Aceleración Máxima*	*Velocidad máxima*
Sway	-30.3 cm	30.3 cm	0.6 g	50 cm/s
Surge	-28.9 cm	37.3 cm	0.6 g	50 cm/s
Heave	-14.6 cm	16.4 cm	0.5 g	40 cm/s
Yaw	-35.7°	35.7°	500 °/s^2	60 °/s
Pitch	-20.0°	18.3°	400 °/s^2	50 °/s
Roll	-21.3°	21.3°	400 °/s^2	50 °/s

Tabla 6.2 - *Características de la plataforma de 6 GdL*

6.1.2 – Descripción de las Pruebas

El objetivo de las pruebas es caracterizar cómo se percibe el funcionamiento de la plataforma por parte de sus usuarios, los pilotos. Para ello, se reclutó a un total de 103 personas, de las cuales 13 fueron rechazadas por sufrir mareo (*motion sickness*). Las restantes 90 fueron agrupadas en 4 grupos disjuntos, basándonos en su experiencia en conducción o navegación.

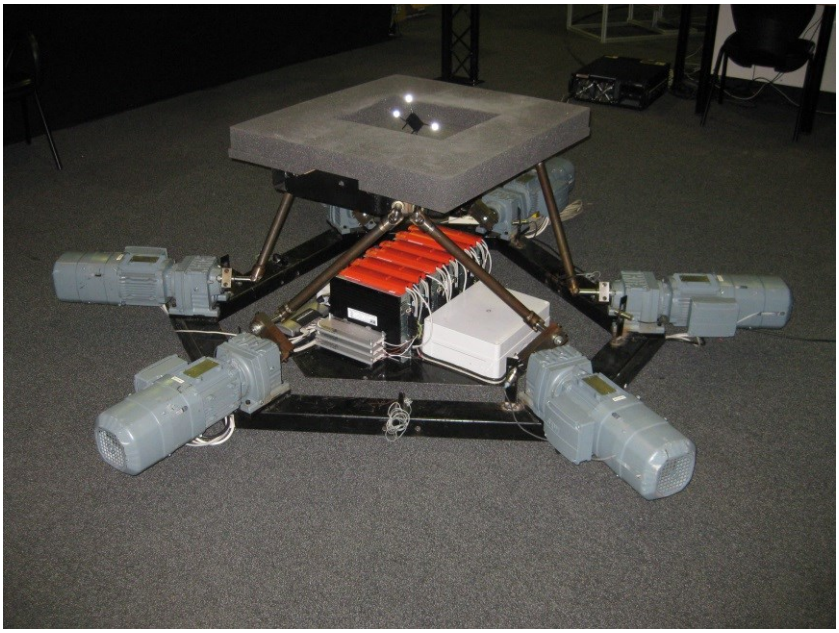

Figura 6.4 - *Plataforma de 6 GdL con tracking óptico*

El primer grupo, C-1, se formó con conductores amateur. El segundo grupo, C-2, con conductores expertos. El tercer grupo, B-1, se correspondía con pilotos de bote de rescate amateur. Finalmente el cuarto grupo, B-2, estaba formado por patrones de

bote expertos. Los grupos C-1 y B-1 reunían a 30 personas cada uno, mientras que los grupos C-2 y B-2, contaban con 15.

Las pruebas se dividieron en dos tandas de test: pruebas con el simulador de conducción y pruebas con el simulador de bote de rescate. Las pruebas de conducción se realizaron con un coche virtual de Formula 3 (F3) en el circuito de Montmeló (Barcelona). Todos los voluntarios fueron sometidos a 3 vueltas (unos 5 minutos) con el piloto automático, cuyo uso se debe al deseo de establecer comparaciones más equilibradas entre los distintos conductores, al eliminar de la prueba las habilidades de los pilotos como posible factor diferenciador.

Las pruebas de navegación se hicieron con el mismo vehículo empleado en el capítulo 3, un *Duarry Brio 620*, equipado con un motor *Suzuki DF 140* de 140 CV. El circuito virtual en este caso consistía en un pasillo de agua, delimitado por dos líneas de boyas paralelas. El ancho del pasillo era de 20 metros, y cada boya estaba separada de la siguiente por 30 metros. Se les indicó a los pilotos que probaran el bote de rescate mientras hacía eslálones por la línea de boyas durante 5 minutos. Como en el caso anterior, los usuarios no pilotaron la embarcación.

La diferencia entre conductores expertos y amateur consistía en que los expertos habían tomado parte en competiciones deportivas de coches - algunos incluso con un F3 – mientras que los conductores amateur tenían, simplemente, al menos 2 años de experiencia en conducción de vehículos de calle. En el caso del simulador de bote de rescate, los expertos habían pilotado

previamente el vehículo simulado, mientras que los amateur habían pilotado embarcaciones similares.

Cada uno de los miembros de cada uno de los 4 grupos fue llamado a probar el simulador con diferentes configuraciones del algoritmo clásico. La primera de las configuraciones, a la que llamaremos *configuración A*, fue ajustada por un experto en algoritmos MCA. El resto de las configuraciones empleaban valores aleatorios para algunos de los parámetros del algoritmo clásico, como veremos a continuación. Para cada una de esas configuraciones probadas se calcularon los indicadores objetivos descritos en el capítulo anterior, y al final de cada una se le formuló a los voluntarios la siguiente pregunta: "¿Cuán creíble ha sido el experimento en términos de percepción de movimiento, en comparación con tu experiencia visual? Puntúala en el rango 0-100". Para ayudar a los usuarios a evaluar más fácilmente cada configuración, la configuración A fue ajustada por el experto para proporcionar lo que él consideraba se correspondía con una puntuación de 50. Además se realizaba siempre una primera prueba con la plataforma deshabilitada. De esta manera, los pilotos disponían de una referencia para poder evaluar numéricamente su experiencia.

Además de todo ello, los pilotos expertos (grupos C-2 y B-2) fueron introducidos en una prueba adicional, consistente en emplear el simulador con y sin el piloto automático, empleando sólo una configuración del algoritmo, llamada *configuración B*, que el experto ajustó para que se correspondiera con la máxima puntuación (100), ya que era el mejor ajuste que se podía obtener, según él, en un tiempo razonable. Esta prueba se diseñó para

conocer si la copia eferente afectaba a la percepción del algoritmo de *washout* [42].

Todas las pruebas se realizaron tanto en el simulador con plataforma de 3 GdL como en el de 6.

6.1.3 – Randomización

El funcionamiento y los parámetros del algoritmo clásico ya han sido explicados con anterioridad. Como el objetivo era caracterizar la respuesta humana ante algoritmos MCA, lo que se hizo fue variar, para cada configuración probada, los valores de una serie de parámetros del algoritmo de forma aleatoria para de esta forma variar el comportamiento del mismo y poder evaluar cómo percibían los usuarios estos cambios y si podíamos correlacionarlos con los indicadores objetivos que, al mismo tiempo, se iban calculando. Esto nos permite analizar cómo reaccionan los pilotos a variaciones del MCA y no a variaciones particulares de los parámetros del algoritmo clásico, una prueba, que en nuestra opinión tendría mucha menor significación. La Tabla 6.3 muestra los valores modificados aleatoriamente del algoritmo clásico. Se muestran sólo los valores que se modificaban en cada prueba. Los que no aparecen en la tabla fueron establecidos a un valor de 1 durante todas las pruebas, ya que, aunque hubiera sido interesante randomizar todos los parámetros, esto suponía modificar demasiados valores en cada prueba. Puesto que algunos de los parámetros del algoritmo tienen efectos similares entre sí, decidimos limitar la lista de valores randomizados a lo mínimo imprescindible.

Parámetro	*Valor - 6 GdL*	*Valor - 3 GdL*
TA – Escalado en X	Aleatorio [0.1-2.0]	Aleatorio [0.1-2.0]
TA – Escalado en Y	Aleatorio [0.1-2.0]	Aleatorio [0.1-2.0]
TA – Escalado en Z	Aleatorio [0.1-2.0]	Aleatorio [0.1-2.0]
RA – Escalado en X	Aleatorio [0.1-2.0]	Aleatorio [0.1-2.0]
RA – Escalado en Y	Aleatorio [0.1-2.0]	Aleatorio [0.1-2.0]
RA – Escalado en Z	Aleatorio [0.1-2.0]	-
THPF – Frec. de corte X	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
THPF – Frec. de corte Y	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
THPF – Frec. de corte Z	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
RHPF – Frec. de corte X	Aleatorio [0.1-10.0]	-
RHPF – Frec. de corte Y	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
RHPF – Frec. de corte Z	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
LPF – Frec. de corte X	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
LPF – Frec. de corte Y	Aleatorio [0.1-10.0]	Aleatorio [0.1-10.0]
TC – Límite inclinación X	Aleatorio [0.5-20.0]	Aleatorio [0.5-20.0]
TC – Límite inclinación Y	Aleatorio [0.5-20.0]	Aleatorio [0.5-20.0]
RL – Límite velocidad inclinación X	Aleatorio [0.5-10.0]	Aleatorio [0.5-10.0]
RL – Límite velocidad inclinación Y	Aleatorio [0.5-10.0]	Aleatorio [0.5-10.0]

Tabla 6.3 - *Parámetros modificados del algoritmo clásico*

6.2 – Análisis y Resultados

En esta sección presentamos el análisis y los resultados obtenidos con las pruebas descritas en apartados anteriores, con el objetivo de encontrar información que nos permita caracterizar

cuál es la respuesta humana ante algoritmos MCA. Los resultados aquí descritos se separan en dos apartados, uno para cada simulador [161].

6.2.1 – Simulador de Conducción

Una manera de estudiar la respuesta humana ante los MCA es estudiar si existe una correlación estadística entre los 5 indicadores calculados (y descritos en el capítulo anterior) y el indicador subjetivo proporcionado por los usuarios para cada configuración probada. Como el grupo C-1 estaba formado por 30 personas, hicimos que cada una probara 10 configuraciones aleatorias para obtener 300 pruebas por grupo. Con estas 300 pruebas estudiamos la correlación de Pearson entre sus valoraciones subjetivas y cada uno de los 5 indicadores objetivos. Los resultados de dicho análisis se puede observar en la Tabla 6.4. Estudiamos tanto la combinación aditiva como la combinación multiplicativa de indicadores. Es importante resaltar que, para la plataforma de 3 GdL, los GdL no simulados también se incluían en el cálculo objetivo, ya que la respuesta subjetiva evidentemente no puede darse eliminando ningún GdL, por lo que la comparación justa es con los 6 GdL. En la combinación aditiva, los pesos se establecieron a 1 para todos los GdL.

Un análisis similar puede realizarse para el grupo C-2 (véase la Tabla 6.5). Esta vez el número de personas en el grupo era 15, pero cada persona probó 20 configuraciones diferentes, por lo que el número total de muestras sigue siendo 300.

		3-GdL		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*EAM*	-0.023	0.69154	0.118	0.04111
	EAMN	0.185	0.00128	0.218	0.00014
	EM	0.195	0.00068	0.142	0.01383
	CCPN	0.422	$< 10^{-5}$	0.511	$< 10^{-5}$
	RE	0.235	0.00004	0.310	$< 10^{-5}$
Combinación aditiva	*EAM*	0.006	0.91757	0.104	0.07207
	EAMN	0.192	0.00083	0.218	0.00014
	EM	0.325	$< 10^{-5}$	0.342	$< 10^{-5}$
	CCPN	0.401	$< 10^{-5}$	0.495	$< 10^{-5}$
	RE	0.255	$< 10^{-5}$	0.357	$< 10^{-5}$

Tabla 6.4 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo C-1*

Uno de los problemas de cualquiera de estos indicadores objetivos es el de cómo combinar la percepción de las diferentes magnitudes físicas medidas. En este estudio medimos 6 señales diferentes (F_x, F_y, F_z, ω_x, ω_y, ω_z), y no está claro si es más importante simular la fuerza específica con precisión o la velocidad angular. Para el estudio de este problema (importancia de la fuerza específica con respecto a la velocidad angular), se estudió la correlación entre el indicador que demostró ser el más significativo (CCPN) y el indicador subjetivo, con estrategias de combinación diferente. El estudio se realizó sólo para el grupo C-2 y se muestra en la Tabla 6.6.

		3-GdL		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*EAM*	0.126	0.02911	0.090	0.11983
	EAMN	0.181	0.00164	0.124	0.03179
	EM	0.136	0.01844	0.175	0.00235
	CCPN	0.410	$< 10^{-5}$	0.433	$< 10^{-5}$
	RE	0.352	$< 10^{-5}$	0.348	$< 10^{-5}$
Combinación aditiva	*EAM*	0.141	0.01452	0.074	0.20121
	EAMN	0.180	0.00175	0.123	0.03320
	EM	0.322	$< 10^{-5}$	0.289	$< 10^{-5}$
	CCPN	0.400	$< 10^{-5}$	0.425	$< 10^{-5}$
	RE	0.349	$< 10^{-5}$	0.347	$< 10^{-5}$

Tabla 6.5 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo C-2*

	Señales empleadas	*3-GdL*		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*F. esp.*	0.428	$< 10^{-5}$	0.396	$< 10^{-5}$
	V. ang.	0.161	0.00519	0.233	0.00005
	Ambas	0.410	$< 10^{-5}$	0.433	$< 10^{-5}$
Combinación aditiva	*F. esp.*	0.432	$< 10^{-5}$	0.397	$< 10^{-5}$
	V. ang.	0.150	0.00927	0.228	0.00006
	Ambas	0.400	$< 10^{-5}$	0.425	$< 10^{-5}$

Tabla 6.6 - *Correlación entre la CCPN y el indicador subjetivo para estrategias de combinación diferente, para el grupo C-2*

Por último, para estudiar si la copia eferente tiene un efecto significativo en la opinión de los usuarios de los MCA, estudiamos si la experiencia subjetiva del conductor era diferente cuando se conducía el vehículo o con el piloto automático (con la misma configuración del algoritmo clásico, configuración B). La Tabla 6.7 resume los resultados.

3-GdL		*6-GdL*	
Humano	*Automático*	*Humano*	*Automático*
85.34	87.23	90.82	88.27

Tabla 6.7 - *Percepción media subjetiva (pilotos humanos vs automáticos), para la configuración B y Grupo C-2*

6.2.2 – Simulador de Bote de Rescate

Un estudio similar (con las mismas fases e igual tipo de pruebas) se realizó para los grupos B-1 y B-2 del simulador de bote de rescate. Las Tablas 6.8, 6.9, 6.10, y 6.11 resumen los resultados, que por otro lado, no son demasiado diferentes a los obtenidos para los grupos C-1 y C-2 correspondientes al simulador de conducción.

6.3 – Discusión

El análisis de los resultados de todas las pruebas presentadas y descritas en los apartados anteriores, revela algunos datos interesantes que discutiremos en este apartado.

		3-GdL		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*EAM*	0.030	0.60477	0.056	0.33371
	EAMN	0.169	0.00332	0.010	0.86306
	EM	0.066	0.25444	0.024	0.67886
	CCPN	0.403	$< 10^{-5}$	0.433	$< 10^{-5}$
	RE	0.227	0.00007	0.177	0.00209
Combinación aditiva	*EAM*	0.006	0.91757	0.035	0.54593
	EAMN	0.171	0.00296	0.007	0.90390
	EM	0.166	0.00394	0.296	$< 10^{-5}$
	CCPN	0.417	$< 10^{-5}$	0.450	$< 10^{-5}$
	RE	0.282	$< 10^{-5}$	0.203	0.00040

Tabla 6.8 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo B-1*

Indicadores

El primer y probablemente más importante resultado es que el indicador CCPN está correlacionado con la percepción subjetiva de los usuarios en todos los grupos probados y en ambos simuladores. La correlación es bastante alta para ser un experimento subjetivo. Además, el p-valor indica una alta significación estadística. Esto es consistente con la idea de que el sistema de percepción humana es incapaz de detectar pequeñas diferencias entre dos señales, pero es muy sensible a la forma de las mismas, cosa que mide muy bien la correlación de Pearson.

		3-GdL		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*EAM*	-0.082	0.15656	0.068	0.24030
	EAMN	-0.023	0.69154	0.121	0.03620
	EM	0.142	0.01383	0.042	0.46861
	CCPN	0.402	$< 10^{-5}$	0.412	$< 10^{-5}$
	RE	0.143	0.01317	0.189	0.00100
Combinación aditiva	*EAM*	-0.063	0.27672	0.045	0.43742
	EAMN	-0.024	0.67886	0.119	0.03941
	EM	0.030	0.60764	0.382	$< 10^{-5}$
	CCPN	0.425	$< 10^{-5}$	0.426	$< 10^{-5}$
	RE	0.168	0.00352	0.195	0.00068

Tabla 6.9 - *Correlaciones entre cada indicador objetivo y el subjetivo, para el grupo B-2*

	Señales empleadas	*3-GdL*		*6-GdL*	
		Correlación	*p-valor*	*Correlación*	*p-valor*
Combinación multiplicativa	*F. esp.*	0.277	$< 10^{-5}$	0.231	0.00005
	V. ang.	0.473	$< 10^{-5}$	0.485	$< 10^{-5}$
	Ambas	0.402	$< 10^{-5}$	0.412	$< 10^{-5}$
Combinación aditiva	*F. esp.*	0.283	$< 10^{-5}$	0.234	0.00004
	V. ang.	0.453	$< 10^{-5}$	0.470	$< 10^{-5}$
	Ambas	0.425	$< 10^{-5}$	0.426	$< 10^{-5}$

Tabla 6.10 - *Correlación entre la CCPN y el indicador subjetivo para estrategias de combinación diferente, para el grupo B-2*

3-GdL		*6-GdL*	
Humano	*Automático*	*Humano*	*Automático*
84.47	85.31	86.72	84.23

Tabla 6.11 - *Percepción media subjetiva (pilotos humanos vs automáticos), para la configuración B y Grupo B-2*

El indicador EM parece representar también bastante bien la percepción del usuario en el caso del simulador de conducción (aunque la correlación con el indicador subjetivo es menor que en el caso de CCPN). Sin embargo, en el caso del simulador de bote de rescate, aunque algunos de los experimentos mostraron correlación entre la experiencia de simulación y el EM, la correlación no es tan general como en el caso del simulador de conducción. Nuestra explicación para este efecto es que el movimiento del coche es mucho más brusco y "limpio" que el movimiento de una embarcación. La experiencia de frenar, girar y acelerar un coche tiene una clara respuesta esperada, mientras que en un bote sobre el mar, las olas, los saltos, los golpes, etc. se solapan con la aceleración provocada por el motor, lo cual hace más difícil identificar y distinguir el movimiento intencionado del no intencionado (el provocado por el mar). Como cada movimiento está mucho más aislado en la simulación de conducción que en el caso de un bote de rescate, los pilotos son capaces de percibir la magnitud del movimiento de forma mucho más precisa y por lo tanto, el índice EM parece ser más útil en un simulador de conducción que en un simulador marino.

El indicador EAM no parece correlacionarse con la percepción del usuario. Aunque mide diferencias en las señales de

salida con respecto a las señales de entrada, el cálculo parece ser demasiado simple para reflejar la percepción del piloto. Además, en el caso de la combinación aditiva este indicador es problemático, al mezclar cantidades medidas en diferentes unidades, aunque en los casos probados no vimos diferencias entre la combinación aditiva y multiplicativa para este indicador.

EAMN parece ser una mejora sobre EAM porque, aunque la correlación con el indicador subjetivo es, a veces, pequeña y no siempre significativa, parece inclinarse hacia el lado positivo del rango [-1, 1], lo cual nos indica que apunta en la dirección correcta.

Finalmente, el indicador RE muestra correlación con el indicador subjetivo en todos los casos. Sin embargo, parece ser un poco más relevante cuando se utiliza el simulador de conducción. Dentro de este simulador, también parece ser más importante cuando los usuarios son los pilotos experimentados. Mientras que este último hecho es bastante comprensible, porque los conductores expertos son más sensibles a los pequeños retrasos que los conductores amateur, el hecho de que en el simulador de conducción se observe más correlación con el indicador RE que en el de bote puede parecer un poco sorprendente, ya que el movimiento del bote es visualmente mucho más notorio que el de un coche, ya que se inclina considerablemente con respecto al plano de agua. Sin embargo, como comentamos previamente, el movimiento de un coche es más distinguible de su entorno y más ordenado que el movimiento de un barco, y por lo tanto, un retraso en una maniobra de frenado o aceleración es mucho más observable que en una embarcación, donde un movimiento retardado podría confundirse con una ola golpeando al vehículo.

Aunque la estimación del retraso puede ser inexacta, ya que el método propuesto no garantiza encontrar el valor real, esto no parece ser un problema, puesto que la estimación de este indicador parece ser lo suficientemente buena para los usuarios como para que se correlacione con sus opiniones subjetivas.

Combinación de Señales

La combinación de señales es un asunto complicado porque se desconoce cómo el cerebro mezcla toda la información del movimiento. De hecho, el mecanismo podría ser diferente de persona a persona e incluso un piloto podría soportar un GdL mal simulado ignorándolo (siempre que se simulen correctamente los otros 5) mientras que otro podría sentir que la simulación queda arruinada por ese GdL mal simulado.

Las pruebas realizadas no revelan mucha información sobre la cuestión de la combinación de las señales características de los algoritmos MCA. Aunque la combinación multiplicativa parecía intuitiva y lógica, parece que la combinación aditiva genera un indicador objetivo global muy similar en conjunto, como demuestra el hecho de que el índice de correlación entre la opinión subjetiva de los pilotos y la combinación aditiva de los indicadores CCPN es, en general, muy similar al índice de correlación que se obtiene cuando se utiliza la combinación multiplicativa (ver Tablas 6.4, 6.5, 6.6, 6.8, 6.9, 6.10).

Las correlaciones se pueden calcular también para señales individuales (aceleración vertical, velocidad angular lateral, etc.), y aunque no se muestran en aras de brevedad, la correlación que se

obtiene es siempre superior al usar el indicador combinado de los 6 GdL que cuando los indicadores se evalúan individualmente. Este es un fuerte argumento a favor de la combinación de las señales. El otro argumento es el hecho de disponer de un único indicador global que nos represente la percepción.

Fuerza Específica vs Velocidad Angular

La pregunta de si la fuerza específica es más importante que la velocidad angular para la percepción global de las señales de movimiento también es una pregunta difícil de responder. Las razones son similares a los que comentamos en la combinación de señales.

Sin embargo, los resultados de las Tablas 6.6 y 6.10, nos sugieren la siguiente conclusión: en un simulador de conducción, la aceleración lineal (fuerza específica) parece más importante que la velocidad angular, mientras que en un simulador de bote de rescate la situación parece ser la contraria, ya que la velocidad angular parece más importante. Aunque esto tiene sentido porque las señales angulares son importantes en el mar, ya que el bote está constantemente cambiando de inclinación por el balanceo y el efecto de elevación de la proa, esta conclusión debe tomarse con cautela. En nuestra opinión, esto es algo que dependerá de varios factores. Si hubiéramos probado el simulador de conducción en una pista diferente (un anillo ovalado o un circuito con más desniveles) o con un coche diferente (con suspensiones más suaves), los resultados podrían haber variado. Por otra parte, la percepción es un asunto personal y la importancia de cada uno de los GdL puede variar de persona a persona.

La conclusión final es que la importancia de las claves angulares o lineales depende del vehículo simulado y debe tenerse en cuenta antes de que se construya el indicador objetivo. En cualquier caso, la ponderación de la fuerza específica sobre la velocidad angular en los indicadores objetivos parece no ser crítica, ya que existe una notable correlación entre los indicadores objetivos y la percepción subjetiva, incluso usando los mismos pesos para todas las direcciones del espacio. Por tanto, no parece que debamos preocuparnos en exceso de este problema.

Grados de Libertad

La cantidad de correlación entre los 5 indicadores propuestos y el indicador subjetivo es bastante similar para ambas plataformas de movimiento. Para el simulador de conducción, la plataforma de movimiento de 6 GdL parece ser mejor, mientras que en el simulador de bote de rescate, algunos indicadores funcionan mejor, curiosamente, con la de 3 GdL y otros con la de 6 GdL, siendo esta última mejor en general. De cualquier modo, creemos que la diferencia, en este caso, es demasiado pequeña para ser significativa. Mientras que la adición de más GdL debe proporcionar mejor percepción y así los resultados para el simulador de conducción son lógicos, en el caso del bote, la naturaleza de sus movimientos hace que estos GdL extra sean mucho menos relevantes. El rango de *yaw* de las plataformas es, habitualmente, demasiado pequeño para reflejar la naturaleza ilimitada (360°) de este movimiento, mientras que el *surge* y el *sway* son a veces apenas perceptibles ya que las aceleraciones lineales del bote a menudo se juntan y confunden con los movimientos

rotacionales [21]. Por estas razones, creemos que los GdL extra son mucho más útiles para la simulación de conducción.

Efecto de la Copia Eferente

La influencia de la copia eferente en la percepción del algoritmo de movimiento parece ser bastante escasa, ya que no observamos diferencias perceptuales significativas entre conducción automática y manual. Parece que cuando se utiliza la plataforma de movimiento de 6 GdL, los pilotos proporcionan notas más altas cuando están conduciendo, pero la diferencia no parece grande. Sin embargo, la situación no se repite con la plataforma de movimiento de 3 GdL y las diferencias en la percepción subjetiva parecen, en cualquier caso, demasiado pequeñas, por lo que asumimos que la copia eferente no afectó a nuestro análisis, y pueden ser debidas a otros factores como el realismo en la reproducción de los mandos y palancas de cada uno de los simuladores.

Lo que sí se observa es que, aunque la configuración B del algoritmo clásico, que es la que se prueba en este experimento, corresponde supuestamente a una puntuación 100 (según el criterio del experto, ya que es la mejor configuración que según él se puede conseguir), a los pilotos les cuesta dar puntuaciones cercanas a 100. Esto es consistente con el hecho de que la simulación de movimiento es siempre subóptima, ya que nunca se puede alcanzar la percepción por las restricciones físicas de la plataforma, y también refleja que los usuarios no creen que se pueda alcanzar una generación de movimiento perfecta.

6.4 – Conclusiones

Teniendo en cuenta nuestros objetivos iniciales, podemos sacar varias conclusiones de este análisis. La primera de ellas es que la reacción de los pilotos humanos ante algoritmos MCA se caracteriza principalmente por la correlación de Pearson normalizada (CCPN) entre las señales (F_x, F_y, F_z, ω_x, ω_y, ω_z virtuales) que está experimentando el vehículo virtual (también visualmente generadas por el simulador) y las señales reales vestibulares que la plataforma de movimiento genera (F_x, F_y, F_z, ω_x, ω_y, ω_z reales). Esto significa que si la correlación es alta, la percepción se considerará realista. Esto valida la principal estrategia de los MCA, que consiste en el escalado y distorsión de las señales, que hace que cambien en magnitud intentando cambiar poco en forma.

Esto significa que la distorsión de magnitud es un factor menos importante que la distorsión de fase. Sin embargo, la distorsión de magnitud también es un factor en la percepción humana del movimiento, como el estudio del indicador EM revela. La distorsión de fase también se mide por el indicador RE. Parece que esta distorsión puede ser un factor importante, especialmente para algunos conductores experimentados. Estos resultados son consistentes con la teoría de Sinacori.

La correlación entre estos indicadores y los valores subjetivos indica que podemos construir un indicador objetivo teniendo en cuenta estas medidas y/o combinándolas. El método de combinación debe investigarse más. El único indicador que

debe descartarse es EAM. El resto de los indicadores puede ser útil, en mayor o menor medida.

Otra conclusión importante es que hay pequeñas diferencias entre los dos simuladores, que nos hacen pensar que estos resultados pueden extrapolarse a otros simuladores. Estas pequeñas diferencias son coherentes con su naturaleza y con las diferencias entre los dos tipos de vehículos.

Las diferencias también son pequeñas entre el simulador de 3 GdL y el de 6. Esto significa que, si bien es preferible emplear una plataforma de movimiento 6 GdL, la cantidad de dinero extra necesario para conseguir una puede no compensar en cuanto a la percepción del movimiento. De hecho, la elección de *heave-pitch-roll* de la plataforma de 3 GdL no es casual. *Pitch* y *roll* son movimientos que ofrecen la posibilidad de simular *surge* y *sway* por medio del *tilt-coordination*. El *heave* se añade a la plataforma porque es el único movimiento de traslación no reproducible por dicho mecanismo de coordinación. Así, a pesar de ser una plataforma de movimiento de 3 GdL, el único movimiento no reproducible es el movimiento de rotación *yaw*. El *yaw* eficaz (por eficaz entendemos combinable con el resto) es en la mayoría de plataformas de 6 GdL demasiado pequeño para reflejar el que tiene un vehículo. A diferencia del *pitch* y el *roll*, el *yaw* es ilimitado (360°) en casi todos los vehículos. Por ello, la diferencia entre una plataforma de 3 GdL y una plataforma de 6 es menor de lo esperado a primera vista.

Capítulo 7

Mejoras en el Ajuste de Parámetros de los MCA

En capítulos anteriores hemos propuesto y validado métricas para la evaluación de algoritmos de *washout*. En este capítulo abordaremos el problema del ajuste de parámetros en dichos algoritmos.

7.1 – Metodología

Dado que ya disponemos de herramientas para establecer comparaciones entre distintas instancias de un MCA, ahora veremos cómo podemos utilizar estos indicadores para asignar valores a los parámetros de dichos algoritmos de manera desatendida, siempre teniendo en mente que el método de ajuste nunca será universal por las limitaciones subjetivas que el problema impone.

Como ya comentamos, el ajuste de parámetros se suele hacer de forma manual, de tal forma que se prueba un conjunto de parámetros con una o varias personas, se les pregunta sus impresiones y en función de ellas, se realizan las modificaciones correspondientes. Ello implica horas y horas de trabajo para ajustar un simulador. Y lo que es peor, para cada vehículo o cada circuito el ajuste puede cambiar. Por ejemplo, si uno dispone de un

simulador de conducción y se pretende configurar un MCA para un turismo, el ajuste puede ser radicalmente diferente si en lugar de un turismo se emplea un Fórmula 1. Incluso puede que haya que hacer diferentes ajustes para un circuito de curvas rápidas y otro de curvas lentas o grandes frenadas porque los movimientos producidos pueden ser distintos y la plataforma debe ajustarse al máximo a los movimientos del vehículo para no alcanzar sus límites pero, al mismo tiempo, no desaprovechar ni un ápice del espacio de movimiento disponible.

Aprovechando que disponemos de una herramienta objetiva de evaluación vamos a intentar automatizar este proceso. Para ello, generaríamos un conjunto de valores para los parámetros del algoritmo, lo ejecutaríamos y necesitaríamos medir sobre la plataforma las magnitudes físicas generadas para compararlas con las que se le pedía simular al algoritmo. Dependiendo del valor de esa comparación se le podría pedir al algoritmo otros valores de los parámetros, y así sucesivamente hasta encontrar un conjunto de valores suficientemente bueno.

El problema de esta solución es que requiere una cantidad ingente de tiempo e incluso podríamos poner en riesgo el *hardware* de la plataforma por la gran cantidad de movimientos que se requieren. Por ello, proponemos realizar una simulación completa de la plataforma de movimiento, aplicar el MCA sobre dicha plataforma virtual, y después evaluar los resultados con la métrica elegida. La ejecución sobre una plataforma virtual nos permitirá ahorrar riesgos y tiempo.

Para realizar esta comparación se pueden seguir varias estrategias. Una de ellas consiste en comparar las magnitudes físicas del vehículo virtual con las magnitudes generadas sobre la plataforma. Si el observador está situado sobre la plataforma de la misma manera que estaría con respecto al vehículo real, y las magnitudes físicas generadas en la plataforma son similares a las magnitudes físicas del vehículo, entonces el observador debería sentir como ciertos los movimientos de la plataforma (se correspondería con la validez física de Reymond y Kemeny).

No en todos los simuladores el puesto de conducción del vehículo está sobre la plataforma en una situación similar a la que estaría sobre un vehículo real. En un avión, el piloto se sitúa en la cabina, que se encuentra bastante lejos del CdM del aparato. Sin embargo, en un simulador de vuelo, lo normal es que el piloto se sitúe en una reproducción de la cabina y que ésta sea directamente accionada por una plataforma de movimiento situada justo debajo, dado que no tiene sentido (ni se podría por las inercias generadas) colocar sobre la plataforma una reproducción completa del avión. Esta diferencia entre la posición del piloto en la realidad y la posición del piloto en el simulador es una característica que diferencia enormemente a los simuladores de vuelo respecto a otros de vehículos no aéreos más pequeños como los simuladores de turismos o pequeñas embarcaciones.

Es por ello que, en los simuladores de vuelo, lo que se intenta no es reproducir sobre la plataforma el movimiento del aparato sino el movimiento sufrido por la cabina donde está el piloto. De aquí surge otra estrategia a la hora de comparar las entradas y las salidas. Dado que lo importante no es que la

plataforma se mueva como el vehículo, sino que a la persona que está encima este movimiento le parezca creíble, otra aproximación puede ser comparar lo que experimentaría la cabeza del piloto, al sufrir las magnitudes que ha calculado el módulo físico, con lo que realmente experimenta la cabeza del piloto sobre la plataforma [57]. El problema de esta comparación es que exige una localización exacta de la cabeza del piloto (cosa que aunque posible es problemática porque para cada piloto es distinta y el piloto puede moverse). A esta estrategia se le puede añadir (o no) un modelo de percepción humano, que suele ceñirse al sistema vestibular de la cabeza (que podría corresponder con la validez perceptual de Reymond y Kemeny).

Dado que este trabajo se centra en vehículos no aéreos, y en algunos de ellos como los coches y los botes de rescate la posición de conducción es prácticamente la misma que la que se tiene en el simulador, la primera aproximación parece razonable.

Es difícil saber si se comete más error introduciendo un modelo de sensorización o comparando directa y objetivamente los movimientos de la plataforma respecto del vehículo simulado. Además, como vimos en capítulos anteriores, no es cierto que la adquisición de claves inerciales por parte del cuerpo humano ocurra sólo en la cabeza, dado que el cuerpo dispone de otro tipo de sensores, además de los presentes en la cabeza, para identificar el movimiento.

Con todas estas consideraciones proponemos el siguiente método de comparación (con las dos alternativas comentadas) (ver Figura 7.1):

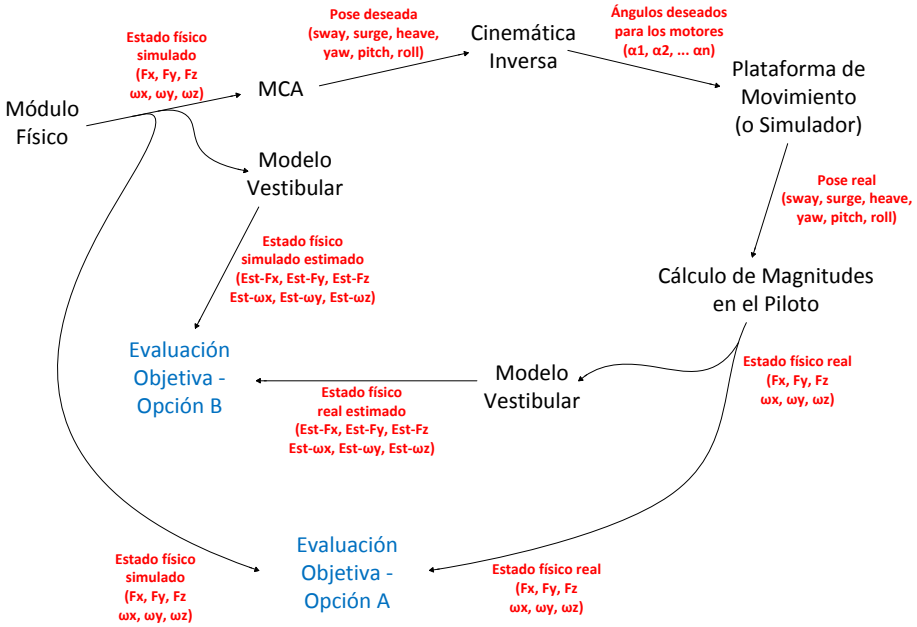

Figura 7.1 - *Esquema de evaluación objetiva de algoritmos de generación de claves gravito-inerciales*

Una vez conocido cuál es el esquema (en sus dos variantes) que emplearemos para evaluar una ejecución de un algoritmo de generación de claves gravito-inerciales, vamos a dedicar el siguiente punto a describir el simulador de plataforma de movimiento (que utilizaremos para evitar usar la plataforma real, automatizar y acelerar el proceso todo lo que podamos), y a continuación de éste, analizaremos qué esquemas algorítmicos podemos emplear para, empleando las métricas que vimos en los capítulos 5 y 6, decidir cuál es la mejor forma de elegir los parámetros de un algoritmo de generación de claves gravito-inerciales.

7.2 – Simulador de Plataforma de Movimiento

Como vimos en capítulos anteriores, una plataforma de movimiento es un sistema autónomo de generación de movimiento accionado de forma electro-mecánica. Las plataformas de movimiento se han venido utilizando para la generación de movimiento en simulaciones de tiempo real pero también para otros tipos de aplicaciones como la formación de personal en equipos industriales [162] o la rehabilitación médica [163]. Como ya hemos comentado, a pesar de que su uso es generalizado, no es fácil generar las señales apropiadas para estos manipuladores de modo que obtengamos el comportamiento deseado [83]. Se requiere un amplio conjunto de pruebas para cada plataforma de movimiento y para cada uso específico, que requieren un ajuste de aproximaciones sucesivas de los parámetros [86]. Este proceso es costoso y requiere mucho tiempo, porque la plataforma de movimiento tiene que ser primero construida y luego testeada durante bastante tiempo para poder asegurar que se comporta como se desea. Sin embargo, dado que en la fase de verificación de la plataforma no se puede estar seguro de que el *software* funcione correctamente, algunas de estas pruebas pueden realizar movimientos severos y potencialmente dañinos que pueden dañar la plataforma de movimiento, y lo que es más importante, pueden provocar daños a humanos en caso de fallo de *software* o *hardware*.

Para resolver estos problemas, se puede utilizar un simulador de plataforma de movimiento. Este simulador realizaría una emulación computarizada de una plataforma de movimiento real, recibiendo las mismas entradas que la plataforma de movimiento

real, y generando las mismas salidas (simuladas). De esta forma podrán hacerse múltiples pruebas sin dañar el *hardware* de la plataforma, y garantizando la seguridad de los probadores humanos. Además, según el *hardware* utilizado para ejecutar el simulador de plataforma, las tareas realizadas por la plataforma virtual pueden ejecutarse más rápido que las tareas reales en la plataforma de movimiento real (ya que los simuladores pueden funcionar en una escala de tiempo acelerada respecto a la real), con el consiguiente ahorro de tiempo. Por todas estas razones, creemos que está justificado el uso de este tipo de soluciones.

Por lo tanto, en esta sección vamos a proponer como mejora para la generación de señales gravito-inerciales en simuladores, el desarrollo de una simulación física en tiempo real de una plataforma de movimiento genérica, que nos permita simular diferentes tipos de plataformas de movimiento para agilizar el proceso de diseño, validación y ajuste de parámetros de dichos dispositivos [164].

7.2.1 – Trabajos Relacionados

Los MEM son dispositivos ampliamente utilizados para añadir claves inerciales en aplicaciones de RV pero la simulación de plataformas de movimiento es algo relativamente infrecuente. Aun así, podemos encontrar algunos trabajos como el de Selvakumar [165] en el que un manipulador paralelo de 3 GdL es simulado utilizando ADAMS [166]. Los resultados de esta plataforma virtual se compararon con las ecuaciones analíticas usando MATLAB [167]. Otros autores [168] simulan una

plataforma de tipo Stewart utilizando también ADAMS y presentan una comparación entre la dinámica directa, proporcionada por ADAMS y la solución inversa, basándose en una formulación de Lagrange. En el trabajo de Gosselin [169] se presenta un simulador que permite un análisis cinemático interactivo de mecanismos paralelos esféricos. Por otro lado, Li [170] emplea ADAMS para realizar una simulación cinemática y dinámica de un manipulador de 3 GdL, para compararlo con la formulación dinámica inversa empleando una aproximación de tipo Newton-Euler.

Sin embargo, todos estos trabajos utilizan soluciones analíticas para estudiar el comportamiento de una plataforma de movimiento específica. Nuestro objetivo es probar MCA, y por lo tanto, queremos construir una plataforma de movimiento virtual que sea capaz de simular una plataforma genérica de movimiento físicamente y numéricamente, ni cinemáticamente ni analíticamente. Además, un aspecto fundamental, que necesitamos para nuestro propósito de mejorar la elección de parámetros, es que la plataforma pueda sustituir a la real y actuar enteramente como ella. Eso es algo de lo que estos trabajos carecen. Una solución similar a nuestras necesidades se presenta en el trabajo de Hulme [171] donde se presenta un simulador de plataforma *Moog 2000E 6-DoF* [133]. Sin embargo, la simulación se limita a este modelo, no permitiendo una especificación genérica. Este simulador utiliza un modelo detallado de CAD para proporcionar información visual y permite intervención "*in-the-loop*" como lo que necesitamos. Sin embargo, a diferencia de nuestra propuesta, esta simulación es cinemática. Soluciones cinemáticas no simulan el proceso de mover los motores de una posición a otra, sólo simulan

el movimiento de la plataforma de una pose a otra. Esto, por desgracia, no es suficiente para probar un MCA, dado que una de las partes más importantes que se deben tener en cuenta son los retrasos en los movimientos provocados por las inercias de la plataforma de movimiento.

El único software que podría satisfacer nuestras necesidades es *Motion Platform Designer* de *Fly Elise* [172]. Es un diseñador de plataformas de movimiento que permite editar, visualizar y probar diversos tipos de plataforma. Es muy completo ya que dispone de varios modelos genéricos de plataforma. Además, su modelo es dinámico y se puede enlazar en tiempo real con diversos simuladores para probar la plataforma con datos en tiempo real. El único problema es que no es un sistema de código abierto y por tanto no es posible modificarlo para adaptarlo a nuestras necesidades y, por ejemplo, modificar el algoritmo de *washout* que utiliza o cambiar sus parámetros sin utilizar su interfaz de usuario.

7.2.2 – Descripción del Simulador

Para simplificar, asumiremos que la plataforma de movimiento simulada utiliza motores rotacionales que mueven bielas, y que éstas se conectan con la base móvil de la plataforma mediante pistones y uniones de distintos tipos, para conseguir los GdL deseados. Un ejemplo de este esquema lo podemos ver en la plataforma analizada en el capítulo 4, y en las empleadas en el capítulo 6. La extensión a motores traslacionales no debería ser complicada.

La idea principal detrás de este simulador es sustituir completamente a la plataforma real, de forma que podamos usar la plataforma virtual en las mismas circunstancias y con los mismos componentes externos que pueda tener la plataforma real (sistema físico de simulación del vehículo real, sistema visual, algoritmo de *washout*, etc.). Esto significa que la plataforma de movimiento simulada debe recibir exactamente las mismas entradas y proporcionar, al menos, las mismas salidas que la plataforma de movimiento real. También significa que un *software* externo (un MCA más un módulo de cinemática inversa) debe ser responsable de la generación de estas entradas, ya que en esta sección nos centramos exclusivamente en la simulación de la plataforma de movimiento.

Por tanto, las entradas de la plataforma de movimiento virtual serán los ángulos deseados para los motores (ya que nos centramos en motores rotacionales), al igual que sucede en la plataforma real. Con respecto a las salidas, la plataforma de movimiento virtual debe proporcionar, al menos, los ángulos actuales de los motores y puede proporcionar opcionalmente el estado actual (en forma de GdL) de la plataforma de movimiento, cosa que aunque la plataforma real no hace, puede ser útil para ahorrarnos realizar externamente el cálculo. Por otra parte, también es recomendable hacer el simulador amigable al usuario, por lo que es conveniente realizar una visualización en tiempo real de los movimientos de la plataforma de movimiento virtual, aunque esto debe ser opcional puesto que ralentiza la simulación.

Con todo esto, podemos estructurar el simulador de plataforma en forma de bucle con la siguiente secuencia de tareas:

la plataforma de movimiento virtual recibe los ángulos deseados para los motores, ésta simula el movimiento de los motores, esto hace que el manipulador virtual se mueva (cambian los GdL conforme los motores se mueven), y este estado actual de la plataforma de movimiento se corresponde con las salidas del sistema (orientación real de los ejes de los motores y GdL de la base móvil).

La parte más difícil de definir aquí es la lista de parámetros requeridos para describir cada plataforma y poder simularla convenientemente, ya que cada plataforma de movimiento real tiene sus propios parámetros y pueden ser muy diferentes. Sin embargo, podemos categorizarlos en: parámetros de los motores (parámetros de control, fuerza máxima, etc.), parámetros de diseño geométrico de la plataforma (longitudes de pistón, dimensiones de bielas, etc.), magnitudes físicas (masas, inercias, etc.), restricciones físicas de movimiento (tipos de articulaciones y límites) y la representación visual de todos los elementos.

Con el fin de proporcionar valores para estos parámetros del simulador, se utiliza un modelo CAD de la plataforma de movimiento y un archivo XML. Para el modelo CAD, elegimos Autodesk 3D Studio Max [173], porque es uno de los paquetes CAD más populares y porque mediante el uso de este *software*, podemos controlar tanto la representación visual de la plataforma de movimiento como su estructura física, permitiendo integrar directamente en nuestro simulador la representación visual del manipulador en forma de grafo de escena (empleando OSG [28]) y la representación física (con Nvidia PhysX [31]). Ambos formatos son soportados por 3D Studio MAX mediante el uso de *plug-in*s

que permiten extraer la información visual y física a archivos .ive (representación visual en formato OSG) y .nxb (representación física en formato PhysX). Este hecho permite que seamos capaces de diseñar plataformas virtuales de movimiento de forma fácil y rápida.

7.2.2.1 – Funcionamiento del Simulador

El simulador de plataforma de movimiento virtual es una simulación gráfica basada en las ecuaciones de la física Newtoniana. Todo el simulador se implementa con C++. La simulación física se apoya en la biblioteca PhysX, mientras que la representación visual se realiza con OSG.

El simulador está estructurado en un sistema de 3 hilos:

· **Hilo de comunicaciones**: recibe los ángulos deseados para los motores (calculados externamente al simulador) y pasa estos valores al hilo de física, para que proceda a calcular el comportamiento necesario de los motores para que éstos puedan alcanzar dichos ángulos. El hilo de comunicaciones también recibe las salidas necesarias desde el hilo de física, en forma de ángulos actuales de los motores, y envía estos datos a quién, externamente al simulador, los requiera.

· **Hilo visual**: es una representación visual mediante OSG del modelo CAD de la plataforma en movimiento.

· **Hilo de física**: lee las entradas recogidas por el hilo de comunicaciones (ángulos de destino/deseados de los motores) y

alimenta unos motores virtuales (controlados por controladores virtuales) que hacen moverse a la plataforma de movimiento virtual. Es responsable de calcular las salidas del simulador.

El uso de una estructura de hilos permite ejecutar cada hilo a una frecuencia diferente. Esto es importante porque el hilo de física debe simular la plataforma de movimiento más rápido de lo que el hilo visual la dibuja y mucho más rápido de lo que el hilo de comunicaciones reciba o envíe las entradas y salidas del simulador.

A fin de acelerar la recogida de datos, decidimos implementar el hilo de comunicaciones mediante memoria compartida. Esto significa que el *software* que se encarga de generar estas entradas (generalmente el MCA) debe escribir esta información en esta memoria compartida. El hilo de comunicaciones leerá estas entradas a un ritmo que está especificado mediante un parámetro de un fichero XML de configuración.

El sistema visual es bastante simple. Carga la descripción visual en formato CAD (archivo .ive en este caso) y dibuja las diferentes partes de dicho modelo en base a las posiciones que el hilo de física calcule para cada una de esas partes. La frecuencia del sistema visual se establece en 25 Hz, valor que, aunque se sitúa por debajo de lo que se suele considerar simulación visual realista, es suficiente para lo que se desea, que no es más que depurar visualmente el resultado. La Figura 7.2 muestra una imagen real de una plataforma y la plataforma de movimiento virtual simulada correspondiente.

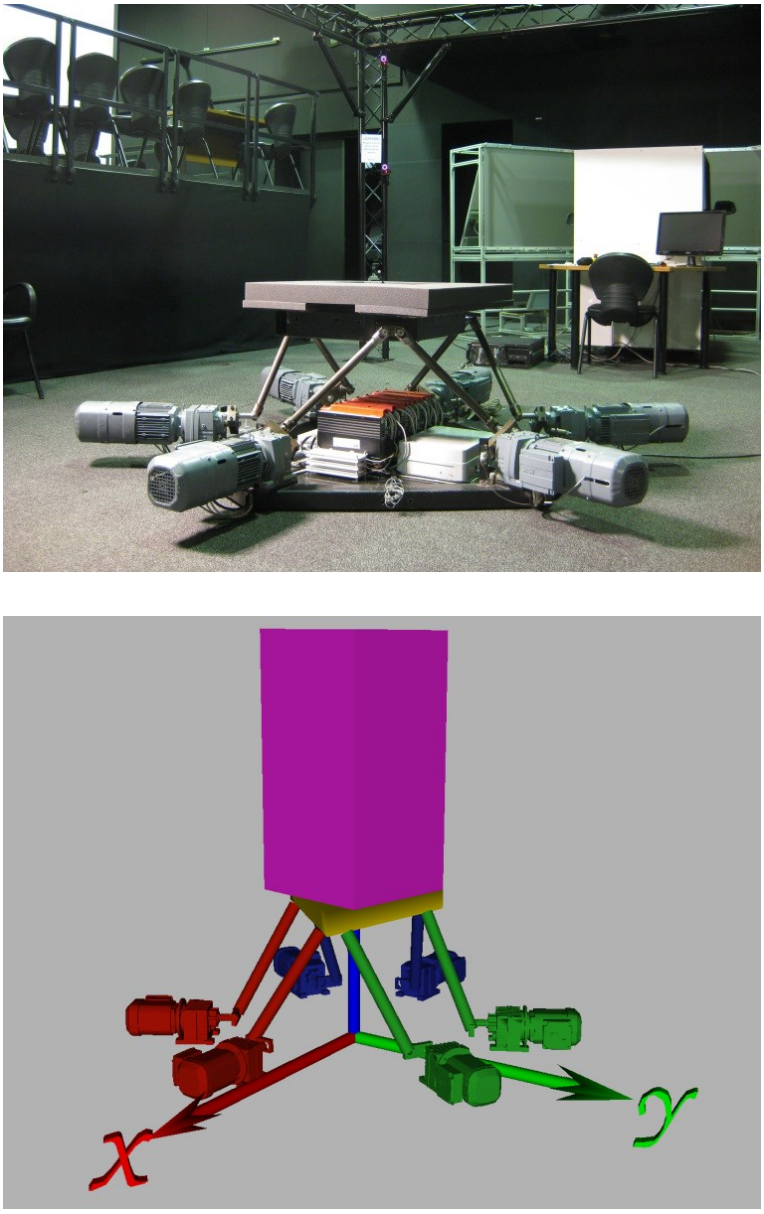

Figura 7.2 - *Plataforma de 6 GdL real (arriba) vs plataforma virtual (abajo)*

El hilo de física es el hilo principal del simulador. Para realizar la simulación física, este hilo carga la descripción física contenida en el fichero CAD (en formato .nxb) e identifica en él los objetos que necesita simular. Nuestro simulador es capaz de identificar 6 categorías de objetos dentro del fichero CAD: motores (objetos denominados MotorXX), articulaciones (llamadas JointXX), bielas de giro (llamadas RodXX), pistones de transmisión de movimiento (llamados PistonXX), la base móvil de la plataforma, que es lo que realmente nos interesa mover (denominada Base) y la carga que se encuentra sobre la base móvil de la plataforma (llamada Load).

Como queremos que el simulador de plataforma sea independiente de la misma, hay algunas reglas que deben seguir los ficheros CAD para ser genéricos y, al mismo tiempo, compatibles con nuestro simulador. En primer lugar, los nombres de los objetos deben seguir las convenciones antedichas. En segundo lugar, las bielas, los pistones, la base móvil de la plataforma y la carga deben ser sólidos rígidos dinámicos (no cinemáticos) de la biblioteca PhysX (NxActor). Cualquier sólido rígido del modelo CAD puede unirse con cualquier otro sólido rígido mediante articulaciones (NxJoint). La lista de articulaciones de las que dispone 3D Studio Max en conjunción con PhysX [174] es lo suficientemente amplia como para poder construir casi cualquier dispositivo de movimiento que nos imaginemos. Por tanto, si los objetos que forman la plataforma de movimiento están o no vinculados por articulaciones, y de qué forma, es algo que depende de la plataforma de movimiento particular, y se define en el modelo CAD. En tercer lugar, los motores son sólidos rígidos cinemáticos (no dinámicos) que no se mueven (NxActor con el *flag*

NX_BF_KINEMATIC), unidos (sus ejes) mediante una articulación de tipo bisagra rotacional (*revolute joint*) a cada una de las bielas (MotorXX siempre debe estar conectado a RodXX). Estas articulaciones (y sólo estas) estarán controladas por un controlador virtual de motores virtuales, que a su vez es comandado por las entradas del simulador, y que se utiliza para calcular las fuerzas necesarias para que los motores se muevan de forma que alcancen los ángulos deseados. Finalmente, puede haber sólo una carga y una única base móvil. La base móvil de la plataforma será el sólido rígido que se utilizará para calcular los GdL de la plataforma de movimiento.

El diseño geométrico de la plataforma de movimiento, las limitaciones específicas de las articulaciones, las masas y las inercias de todos los objetos son parámetros que el simulador obtiene directamente del diseño físico de la plataforma en formato .nxb. Lo único que no está presente en el modelo CAD son las características de los controladores virtuales de los motores, que son parámetros que se introducen en el fichero XML de configuración del simulador. Además, las masas y las inercias de todos los objetos también pueden cambiarse en el fichero XML.

El controlador virtual de los motores (o simulador de controlador de motores) es una parte importante del simulador de plataforma. Para diferenciarnos de enfoques cinemáticos, nos propusimos simular el movimiento de los motores de manera realista, es decir, dinámicamente. Pero para hacer esto, hace falta un controlador para decidir qué fuerza deben hacer los motores en cada momento. Para controlar los motores, decidimos construir un sistema de control PID virtual. A pesar de los muchos avances en

teoría de control, este tipo de controladores es uno de los más usados [175] y proporciona buenos resultados cuando no se dispone de un buen modelo del sistema a controlar [176], lo cual sucede en este caso.

Los parámetros de un controlador PID son siempre: Kp (constante proporcional), Ki (constante integradora) y Kd (constante derivativa) [176]. La entrada del controlador es, en este caso, el ángulo deseado para el motor (ángulo objetivo o de destino) y la salida que debe proporcionar dicho controlador es el par resultante en la articulación motor-biela. En el modelo real, dicho par es producido por la aplicación de una corriente por parte del motor que induce electromagnéticamente un giro del mismo que produce un torque o par. Sin embargo, a nivel dinámico, todo el proceso electromagnético podemos obviarlo y tomar como salida el par generado por el motor.

Una vez que los controladores de los motores han calculado el par resultante, este par se aplica a la articulación motor-biela, y PhysX realiza todos los cálculos necesarios sobre la escena física simulada con el fin de calcular las posiciones o estados de todos los restantes objetos (pistones, articulaciones, base móvil y carga). Una vez que la base móvil de la plataforma es reposicionada según estos cálculos físicos, podemos pedir a PhysX su posición y orientación en forma de pose con 6 componentes, que son en realidad los GdL actuales de la plataforma. Para lograr esta información no es necesario emplear la cinemática directa, ya que PhysX realiza todos los cálculos de forma dinámica (no cinemática) por nosotros.

El ajuste de los parámetros del controlador requiere algún conocimiento de cómo funciona un controlador PID y también del funcionamiento del propio motor, pero también puede ser automatizado [175], aunque ello suele requerir disponer de un modelo del sistema a controlar.

El hilo de física realiza todos estos cálculos a una frecuencia parametrizable (vía XML). Cuanto mayor sea la frecuencia, más precisa es la simulación, pero más tiempo consume el propio cálculo de la simulación. Si la frecuencia es demasiado alta, el proceso de simulación podría necesitar más tiempo para realizar el cálculo del que realmente se intenta simular, dando como resultado una simulación diferida, que no es en tiempo real, cosa que no nos serviría. Sin embargo, si la frecuencia es demasiado baja, la precisión de la simulación disminuye debido a una integración física inestable [32]. Es por ello que se debe encontrar un punto de equilibrio para establecer una frecuencia adecuada.

La validación del simulador se puede encontrar en [177] y/o en el Apéndice C, donde se puede ver que el simulador puede funcionar varias veces más rápido que el tiempo real sin necesidad de usar *hardware* específico.

7.3 – Esquemas Algorítmicos para el Ajuste de Parámetros

Dado que disponemos de una métrica objetiva para evaluar algoritmos MCA, y una herramienta capaz de generar movimientos simulados sobre plataformas de movimiento, incluso más rápido

que en tiempo real, podemos aprovechar dichas métricas y esta herramienta para probar sobre ella múltiples instancias de algoritmos MCA y evaluarlas de forma automática.

Por desgracia, dado que todo algoritmo MCA presenta una elevada cantidad de parámetros y éstos presentan rangos de variación grandes, el espacio de parámetros de un algoritmo de este tipo puede ser enorme. Pongamos como ejemplo el algoritmo clásico. Si permitimos que varíen todos sus parámetros, el espacio de parámetros es un espacio vectorial de dimensión 36, ya que éste es aproximadamente el número de parámetros variables que puede llegar a tener con la implementación del esquema de Reid-Nahon (evidentemente depende de la implementación y podrían ser algunos más). Aunque muchas veces no es necesario ajustarlos todos, ya que algunos tienen efectos parecidos, el número de parámetros a ajustar puede oscilar entre 15 y 20, que siguen siendo muchos. ¿Cómo podemos hacer para encontrar en ese inmenso espacio el mejor conjunto de valores?

En algoritmia, el problema de encontrar la mejor solución, respecto a un determinado criterio, dentro de un conjunto de valores recibe el nombre de **problema de optimización**. Dependiendo del tamaño del conjunto, el problema se puede abordar de varias maneras.

Si el conjunto es finito, el problema pasa a llamarse optimización combinatoria, ya que se reduce a encontrar la combinación óptima dentro de ese conjunto finito de combinaciones. Si no es finito, se llama problema de optimización

continuo, y normalmente se resuelve discretizando el conjunto infinito y aplicando métodos de optimización combinatoria.

Si además de finito, el conjunto no es muy grande, se puede plantear una búsqueda exhaustiva. Este es el método más primitivo y aunque no es el mejor en términos computacionales, funciona en el 100% de los casos.

A veces, la búsqueda exhaustiva es computacionalmente intratable por su coste. Esto hace que debamos plantearnos otras soluciones. Las ciencias de la computación han estudiado este tipo de problemas y se conocen decenas de métodos. Dichos métodos se pueden clasificar en dos tipos, aquellos que buscan la solución óptima de forma iterativa o convergente, y aquellos que buscan y ofrecen una solución aproximada (subóptima).

Por desgracia, los métodos iterativos no se pueden aplicar a todos los tipos de problemas, ya que, en muchos de ellos, se desconoce la forma exacta del espacio de soluciones, por lo que no es posible construir ningún camino fiable que lleve a la solución.

Entre los algoritmos de tipo subóptimo encontramos los algoritmos voraces [178] y las heurísticas [179]. El funcionamiento de los algoritmos voraces consiste en elegir un cierto camino irreversible dentro del espacio de soluciones de forma que cada decisión se toma siguiendo un criterio que se considera localmente óptimo, con la esperanza de que la toma de decisiones locales óptimas nos lleve a una decisión global óptima, cosa que no es necesariamente cierta, pero que en algunos casos lleva a soluciones finales razonablemente buenas.

Los algoritmos heurísticos funcionan de un modo similar a los voraces, ya que intentan alcanzar una solución razonablemente buena en base a tomar ciertas decisiones intermedias acertadas. Sin embargo, en las heurísticas, a diferencia de los voraces, la toma de decisiones en base a un cierto criterio local no tiene por qué ser irreversible. Heurística significa regla o procedimiento no formal cuyo uso está orientado a la resolución de un problema.

Los algoritmos heurísticos de optimización son útiles en determinadas circunstancias [180]:

- cuando la solución exacta no existe o es muy costosa de encontrar.
- cuando alcanzar la solución óptima no es estrictamente necesario y podemos conformarnos con soluciones cercanas, que podríamos llamar satisfactorias o subóptimas.
- cuando las funciones de evaluación que nos indican cómo de buena es una solución no son del todo fiables.
- cuando tenemos fuertes limitaciones temporales.

Todas las anteriores circunstancias se dan en el problema que estamos intentando resolver, especialmente la no necesidad de la solución perfecta y la no fiabilidad de la función de evaluación, ya que tiene un substrato subjetivo importante. Por tanto, el uso de heurísticas está plenamente justificado para este problema.

En los siguientes apartados vamos a ver cómo podemos aplicar algoritmos de optimización para intentar resolver el problema de asignar valores óptimos a los parámetros de los algoritmos de *washout*.

7.3.1 – Planteamiento del Problema

Antes de intentar resolver el problema, vamos a intentar describirlo y formalizarlo de la mejor manera posible.

Dispondremos de una instancia de un algoritmo de *washout*, que denominaremos *w* y que pertenecerá al conjunto de algoritmos de este tipo, llamado *W*. Cada algoritmo del conjunto *W* dispone de una serie de parámetros (en principio distintos en número, forma y semántica para cada algoritmo), cuyos valores modifican sustancialmente el comportamiento del mismo. Supondremos que todos los parámetros de todos los algoritmos del conjunto *W* son de tipo real (es lo habitual, y si son enteros o booleanos se pueden codificar fácilmente como reales), y supondremos que el número de parámetros del algoritmo *w*, que queremos optimizar, es *n*. Cada t-upla de longitud *n* de valores de parámetros hará que el algoritmo funcione de una manera distinta. Para evaluar cómo funciona el algoritmo con cada t-upla de valores, dispondremos de una función de evaluación *f*, que nos dirá para cada algoritmo y t-upla de parámetros si el algoritmo se comporta bien o no. Esta evaluación vendrá descrita en forma de número real positivo, de tal modo que a mayor valor de evaluación peor valoración. Las métricas (empleando los 6 GdL) evaluadas en el capítulo 6 son ejemplos de funciones de evaluación que podríamos emplear.

El objetivo del algoritmo es encontrar una t-upla *p* de parámetros que proporcione el valor *o* más pequeño de la función de evaluación para el algoritmo *w*. Como los parámetros casi siempre se asignan dentro de un cierto rango de valores razonable, podemos restringir el problema a buscar dentro de esos rangos de

valores. La especificación del algoritmo, restringiendo los valores de los parámetros a un cierto rango, es la siguiente (seguiremos el esquema y notación de Ferri [178]):

Algoritmo **OptimizacionMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros

Salidas:
p : vector [1..n] de R // t-upla óptima de parámetros
o : R // valor de evaluación de la t-upla óptima

Precondición:
$n > 0$
$\forall\ i=1..n :\ li[i] < ls[i]$
$\forall\ x \in R^n , \forall\ w \in W :\ f(w, x) \geq 1$

Postcondición:
$\forall\ x \in R^n : f(w, x) > o \mid w \in W,\ o = f(w, p),\ li[i] \leq p[i] \leq ls[i]$
con i= 1..n

Fin de OptimizacionMCA

Evidentemente, la evaluación del algoritmo de *washout* dependerá de sobre qué plataforma se pruebe, qué simulador genere los datos del estado físico y muchos otros factores. Es por ello que la palabra optimización hay que entenderla en el contexto

adecuado. Se trata de optimizar el algoritmo para la función de evaluación elegida, la plataforma, el tipo de simulador y el vehículo empleado.

Como los parámetros son de tipo real, el número de valores posible para cada uno de los parámetros es infinito. Incluso restringiéndolos a un cierto rango de valores, el espacio de posibles soluciones sigue siendo infinito (aunque más pequeño), por lo que su exploración exhaustiva es imposible. Por ello, debemos encontrar alguna manera inteligente de explorar el espacio de soluciones R^n.

Una alternativa consiste en discretizar los rangos de valores. Con ello conseguimos convertir el problema en un problema de optimización combinatoria. Aunque de esta manera el problema es más fácil de resolver, hacer una búsqueda exhaustiva de este espacio discretizado (que ahora es finito) sigue siendo computacionalmente muy costoso, dado que habitualmente el espacio de soluciones es muy grande, a no ser que la discretización sea muy burda o el número de parámetros muy pequeño.

El problema fundamental radica en que la consulta de la función de evaluación puede llegar a ser muy costosa, ya que lo habitual es testear los algoritmos MCA sobre una cierta ejecución más o menos larga del simulador. En un simulador de conducción correspondería, por ejemplo, a una vuelta del circuito, lo cual cuesta del orden de minutos, dependiendo del vehículo y el circuito. Dado que disponemos de un simulador de plataforma, podemos acelerar el proceso pero si tenemos que hacer millones de evaluaciones, el problema es igualmente inabordable a nivel

computacional. De hecho, si tenemos n parámetros y cada uno puede tomar d valores distintos, el número de combinaciones que necesitamos probar es d^n, por lo que el coste computacional de probarlas todas es muy alto y ni siquiera estaríamos resolviendo el problema original, ya que estamos discretizando los rangos de valores.

Las soluciones voraces tampoco parecen aplicables porque no se conoce la forma del espacio de soluciones y no es factible que decisiones locales irreversibles nos lleven a soluciones globalmente óptimas. Por ello, debemos buscar soluciones subóptimas que intenten minimizar el valor de la función de evaluación.

7.3.2 – Búsqueda Exhaustiva Discretizada

Como hemos comentado, la estrategia de búsqueda exhaustiva discretizada consistiría en recorrer todo el espacio de soluciones posible para ver cuál es la mejor solución. Esto se corresponde con una estrategia de tipo *backtracking* [178] sin poda, el coste temporal del cual es exponencial con el número de parámetros. De hecho, este tipo de problemas es NP-difícil [181].

Aunque no es la mejor de las estrategias para recorrer un espacio tan grande, nos sirve como referencia y solución fácilmente validable para comparar el rendimiento de otras soluciones. Una posible aproximación del método de búsqueda exhaustiva discretizada para la optimización de parámetros en algoritmos MCA podría ser la siguiente:

Algoritmo **BusquedaExhaustivaMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación
li : vector [1..n] de R *// límite inferior del rango de los parámetros*
ls : vector [1..n] de R *// límite superior del rango de los parámetros*
lp : vector [1..n] de R *// lista de parámetros*

Parámetros:
maxTiempo : R
numDivisiones : N // número de divisiones de cada rango

Auxiliar:
i, maxIters : N
params : vector [1..n] de R
eval, tiempo : R

Salidas:
p : vector [1..n] de R *// t-upla óptima de parámetros*
o : R *// valor de evaluación de la t-upla óptima*

Algoritmo:

```
i = 0;
o = ∞;
maxIters = 1;

desde (i = 1 hasta n) hacer
{
   maxIters = maxIters*(numDivisiones + 1);
   lp[i] = li[i];
}
```

```
tiempo = 0.0;
ponerRelojEnMarcha();

hacer
{
   params = generarValoresPorOrden(n, li, ls, lp);
   eval = f(w, params);

   si (eval < o) entonces
   {
      o = eval;
      p = params;
   }

   i = i +1;
   tiempo = consultarReloj();
}
mientras ( (i < maxIters) ^ (tiempo < maxTiempo) )
```

Fin de BusquedaExhaustivaMCA

Función **generarValoresPorOrden()**

Entradas:

n : N // número de parámetros modificables del algoritmo w
li : vector [1..n] de R *// límite inferior del rango de los parámetros*
ls : vector [1..n] de R *// límite superior del rango de los parámetros*
lp : vector [1..n] de R *// t-upla de parámetros actual*

Parámetros:

numDivisiones : N // número de divisiones de cada rango

Auxiliar:

inc : R

Salidas:
s : vector [1..n] de R *// nueva t-upla de parámetros*

Algoritmo:

```
desde (i = 1 hasta n) hacer
{
   inc = (ls[i] – li[i])/numDivisiones;

   si (lp[i] ≥ ls[i]) entonces
      lp[i] = li[i];
   sino
   {
      lp[i] = lp[i] + inc;
      cortar bucle;
   }
}

s = lp;
```

Fin de generarValoresPorOrden()

Los diferentes valores de los parámetros se prueban en orden, por lo que se le podría denominar "búsqueda ordenada". Sin embargo, es importante recalcar que buscar en un espacio ordenado respecto a los valores de los parámetros, no garantiza en absoluto que dicha búsqueda se produzca de forma ordenada con respecto a la función de evaluación, que es lo que nos interesaría pero que, por desgracia, no podemos hacer.

Para que podamos elegir la condición de terminación que nos interese más, en lugar de que ésta sea el número de iteraciones

(que puede ser enorme en este caso), se puede comprobar también que no se exceda un tiempo máximo. Esta condición de terminación permitirá una comparación más justa entre los distintos algoritmos.

7.3.3 – Método de Monte-Carlo

Los algoritmos de Monte-Carlo son algoritmos de tipo probabilístico no determinista. El primer algoritmo de este tipo fue ideado por John Von Neumann, Stanislaw Ulam y Nicholas Metropolis mientras trabajaban en el tristemente célebre Proyecto Manhattan. Su nombre es indicativo de su modo de funcionamiento, ya que funcionan seleccionando aleatoriamente valores para los posibles resultados, para posteriormente hacer algún cálculo determinista sobre ellos. El esquema varía en función del tipo de problema.

En el caso que nos ocupa, la aplicación de este esquema intentaría evitar hacer una búsqueda exhaustiva del espacio de soluciones escogiendo elementos aleatorios de dicho espacio y evaluándolos. Si hacemos suficientes pruebas aleatorias, lo más probable es que recubramos de un modo bastante amplio, uniforme, y con un espaciado más o menos constante, el espacio de soluciones. Como la ejecución debe acabar en un tiempo finito, se debe poner un límite al número de configuraciones aleatorias que prueba el algoritmo, o un tiempo máximo. Una posible aproximación del método de Monte-Carlo a la optimización de parámetros en algoritmos MCA podría ser la siguiente:

Algoritmo **MonteCarloMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros

Parámetros:
maxIters : N
maxTiempo : R

Auxiliar:
i : N
aleat : vector [1..n] de R
eval, tiempo : R

Salidas:
p : vector [1..n] de R // t-upla óptima de parámetros
o : R // valor de evaluación de la t-upla óptima

Algoritmo:

```
i = 0;
o = ∞;

tiempo = 0.0;
ponerRelojEnMarcha();

mientras ( (i < maxIters) ^ (tiempo < maxTiempo) ) hacer
{
   aleat = generarValoresAleatoriosEnRangos(n, li, ls);
   eval = f(w, aleat);
```

```
    si (eval < o) entonces
    {
        o = eval;
        p = aleat;
    }

    i = i +1;
    tiempo = consultarReloj();
}
```

Fin de MonteCarloMCA

El algoritmo de Monte-Carlo parece una mejora sobre una búsqueda exhaustiva ordenada ya que, como no conocemos qué forma tienen los valores de evaluación del espacio de soluciones, nada nos garantiza que recorrerlo en un cierto orden sea bueno. De hecho, lo más probable, es que si lo recorremos de modo aleatorio probemos soluciones muy distintas entre sí y alcancemos valores buenos (que no óptimos) más rápidamente, por lo que, en general, debe ser capaz de encontrar mejores soluciones que la búsqueda exhaustiva, dado el mismo tiempo de ejecución.

7.3.4 – Heurísticas

Dado que el método de Monte-Carlo sigue ofreciendo poca inteligencia en la búsqueda de la solución óptima, el paso siguiente es idear una búsqueda un poco más guiada. Aunque los valores de evaluación del espacio de soluciones no forman a priori ninguna forma conocida que podamos explorar analíticamente, es de esperar que cumplan el principio de localidad. Es decir, dos t-uplas

de parámetros que difieran poco entre sí (cuya distancia entre ellas sea pequeña) deberían obtener similares valores de la función de evaluación. Las pruebas realizadas con personas con anterioridad, así lo sugieren. Esto hace que los valores de evaluación del espacio de soluciones no tengan forma aleatoria sino que presenten una forma más o menos continua con *valles* y *montañas* (aunque en muchas dimensiones).

Aprovechando esta idea, podemos desarrollar heurísticas para guiar el proceso de búsqueda. Existen múltiples heurísticas para la búsqueda de soluciones óptimas [182], [180]. De entre ellas las que creemos que se adaptan mejor a este tipo de problema son el Recocido Simulado y los algoritmos genéticos.

7.3.5 – Recocido Simulado

El Recocido Simulado (*Simulated Annealing*, SA) es una técnica heurística para buscar un mínimo (o máximo) global dentro de un espacio de búsqueda grande. El algoritmo está inspirado en el proceso de recocido del acero. La técnica de trabajo del acero consiste en calentarlo rápidamente para que sus átomos adquieran energía y se desplacen respecto de sus posiciones iniciales (que corresponden con mínimos locales de energía), para luego enfriarlo lentamente. El enfriamiento lento provoca que la probabilidad de formar configuraciones de energía más estables que la inicial aumente, hasta que en algún momento se alcance la configuración de mínima energía posible (lo que correspondería con un mínimo global) [183], [180]. Una posible aplicación de esta idea al ajuste de parámetros en MCA podría ser la siguiente:

Algoritmo **RecocidoSimuladoMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros

Parámetros:
maxIters, maxReps : N
maxTiempo, tempInicial, tempFinal, K : R

Auxiliar:
sol1, sol2 : vector [1..n] de R
eval1, eval2, temp, delta, tiempo, aux, exp : R
i, numReps : N

Salidas:
p : vector [1..n] de R // t-upla óptima de parámetros
o : R // valor de evaluación de la t-upla óptima

Algoritmo:

```
i = 0;
numReps = 0;
temp = tempInicial;

tiempo = 0.0;
ponerRelojEnMarcha();

sol1 = generarValoresAleatoriosEnRangos(n, li, ls);
eval1 = f(w, sol1);
o = eval1;
p = sol1;
```

```
hacer
{
   sol2 = generarSolucionCercana(n, li, ls, sol1);
   eval2 = f(w, sol2);

   si (eval2 < o) entonces
   {
      o = eval2;
      p = sol2;
      numReps = 0;
   }
   si no
   {
      numReps = numReps + 1;
      si (numReps ≥ maxReps) entonces
      {
         sol1 = p;
         eval1 = o;
         numReps = 0;
      }
   }

   delta = eval1 – eval2;
   aux = (-delta*K)/temp;

   si (aux ≥ 0.0) entonces
      exp = 1.0;
   si no
      exp =e^aux;

   si (exp > generarValorAleatorioEnRango(0.0, 1.0)) entonces
   {
      sol1 = sol2;
      eval1 = eval2;
   }
```

```
   i = i +1;
   temp = ((tempInicial – tempFinal)*(maxIters - i))/maxIters;
   tiempo = consultarReloj();
}
mientras ( (i < maxIters) ^ (tiempo < maxTiempo) )
```

Fin de RecocidoSimuladoMCA

Función **generarSolucionCercana()**

Entradas:

n : N // número de parámetros modificables del algoritmo w
li : vector [1..n] de R // *límite inferior del rango de los parámetros*
ls : vector [1..n] de R // *límite superior del rango de los parámetros*
t : vector [1..n] de R // *t-upla de parámetros*

Parámetros:

tasaModificacion : R

Auxiliar:

x, rango : R

Salidas:

s : vector [1..n] de R // *nueva t-upla de parámetros*

Algoritmo:

```
desde (i = 1 hasta n) hacer
{
   rango = (ls[i] – li[i])*tasaModificacion;
   x = generarValorAleatorioEnRango(-0.5*rango, 0.5*rango);
   s[i] = t[i] + x;
```

```
    si (s[i] > ls[i]) entonces
        s[i] = ls[i];

    si (s[i] < li[i]) entonces
        s[i] = li[i];
}
```

Fin de generarSolucionCercana()

7.3.6 – Algoritmo Genético o Darwiniano

La siguiente técnica heurística que vamos a adaptar a nuestro problema ha demostrado ser una de las más exitosas en la búsqueda de soluciones óptimas en espacios de soluciones muy grandes.

Un algoritmo genético, evolutivo o Darwiniano [184], [180] replica el funcionamiento evolutivo de los ecosistemas naturales, para encontrar una solución óptima (en realidad subóptima, porque es una heurística) a un problema. Aunque el término *genético* es el más empleado porque los temas relacionados con el ADN están de actualidad, el término *evolutivo* o incluso *Darwiniano* nos parecen más acertados porque refleja mejor el funcionamiento del algoritmo.

Para replicar el funcionamiento evolutivo de la naturaleza, el algoritmo necesita ser capaz de codificar cada posible solución del problema mediante una cadena de caracteres. Cada posible solución del algoritmo recibirá el nombre de individuo, y su codificación se corresponderá con el ADN (los genes) de ese

individuo. Si podemos evaluar cada uno de esos individuos con una función de evaluación, que en el caso de los algoritmos genéticos se suele llamar función de *fitness*, podemos generar poblaciones de individuos (soluciones) y hacerlas evolucionar en el tiempo, de modo que sólo los individuos más aptos (con mejor función de evaluación) sobrevivan, y se reproduzcan (combinando dos buenas soluciones de alguna manera) dando lugar a nuevas soluciones que contendrán características de sus padres pero sufrirán pequeñas mutaciones (alteraciones). Al paso de varias generaciones, es de esperar que la población contenga individuos mejor adaptados (soluciones buenas). Para una descripción más detallada de este tipo de algoritmos, el lector puede consultar [180].

El funcionamiento del algoritmo genético se puede adaptar a nuestro problema de la siguiente manera.

Consideremos como posibles individuos cada t-upla x de valores para los parámetros de nuestro algoritmo w. Codificaremos cada individuo simplemente como una secuencia ordenada de los valores de los parámetros: $x[1]\ x[2]\ \ldots\ x[i]\ \ldots\ x[n]$. Como función de *fitness* nos vale nuestra función de evaluación f, que coge un algoritmo y una asignación de parámetros y lo evalúa.

Partiremos inicialmente con una población P de individuos (posibles soluciones) aleatorios. Evaluaremos todos los elementos de P y los ordenaremos en un vector según el valor creciente de dicha evaluación. Por efecto de la selección natural [185] sólo sobrevivirá un porcentaje de ellos. El resto, morirá. Esto se traduce en que borraremos la parte final del vector de individuos ordenados.

Una vez el proceso de selección natural ha eliminado a los individuos menos aptos (las soluciones peores), los supervivientes se reproducirán dando lugar a nuevas soluciones (individuos). Dichos individuos deben contener material genético de ambos padres (parte de ambas soluciones). Dado que las soluciones padre eran de las mejores de su generación, si juntamos características de ambas, es de esperar que la solución resultante sea también buena, y a veces mejor. Además, en este punto se pueden producir pequeñas mutaciones que hacen que los hijos reciban los genes de los padres mejorados o empeorados (depende de la mutación). Las mutaciones pueden hacer al individuo más apto (mejor solución) o menos apto (peor solución).

Existen varias maneras de hacer la recombinación genética. La reproducción puede ser sexual (se asigna sexo a los padres) o asexual. La elección de los padres puede ser totalmente aleatoria o guiada por su función de *fitness* (los más aptos se reproducen más). Además, la mezcla del material genético se puede hacer también de varias formas.

Para nuestro problema, consideramos que lo más adecuado era una reproducción asexual, con selección aleatoria de padres (la selección guiada suele generar poblaciones poco variadas) y recombinación mediante mitosis (que la solución hija contenga la mitad de material genético de cada padre). Para ello, lo que haremos es escoger, cada vez, aleatoriamente, dos individuos para ponerlos a procrear. Haremos este proceso tantas veces como individuos murieran en la selección natural anterior, para reponer la población.

Para la recombinación de genes en la procreación, lo que haremos es escoger de los *n* parámetros, aleatoria y equiprobablemente, o bien el valor que portaba el primer padre para ese parámetro o bien el que portaba el segundo padre. En promedio, cada hijo obtendrá la mitad de valores de cada padre.

Una vez combinados los genes de ambos padres para formar un nuevo individuo, pueden producirse mutaciones. Las mutaciones son las que hacen que el algoritmo funcione, porque si no, el material genético no se renovaría y lo único que haríamos sería recombinar el mismo material genético una y otra vez.

En nuestro caso, la mutación (si ocurre) consistirá en cambiar aleatoriamente el valor de cada uno de los *n* parámetros. Se recorren todos y para cada uno, con cierta probabilidad se decide si hay mutación o no. Si se da la mutación, se generará un valor completamente nuevo para dicho parámetro dentro de su rango válido. Con ello, tenemos una nueva solución creada a partir de dos soluciones de la generación anterior pero ligeramente variada. El proceso completo de reproducción y mutación se puede observar mejor en la Figura 7.3.

Posteriormente, todos los individuos padres mueren por vejez y los hijos constituyen la nueva generación, que es de esperar que contenga mejores soluciones que la anterior, ya que en la anterior generación una serie de individuos murieron y sólo los buenos aportaron su material genético para crear la siguiente generación.

Una modificación interesante que se puede introducir (y que no ocurre en la naturaleza) es la de añadir la posibilidad de que unos pocos individuos con valores altos de la función de *fitness*, sobrevivan de generación en generación y nunca mueran. A esto se llama *elitismo*. En nuestro caso, el elitismo se consideró una buena aportación, ya que si encontramos una muy buena solución, esta no debería ser descartada (morir) incluso aunque tenga muchos hijos. La implementación del elitismo, consistirá en nuestro caso en no eliminar los primeros elementos del vector ordenado de individuos.

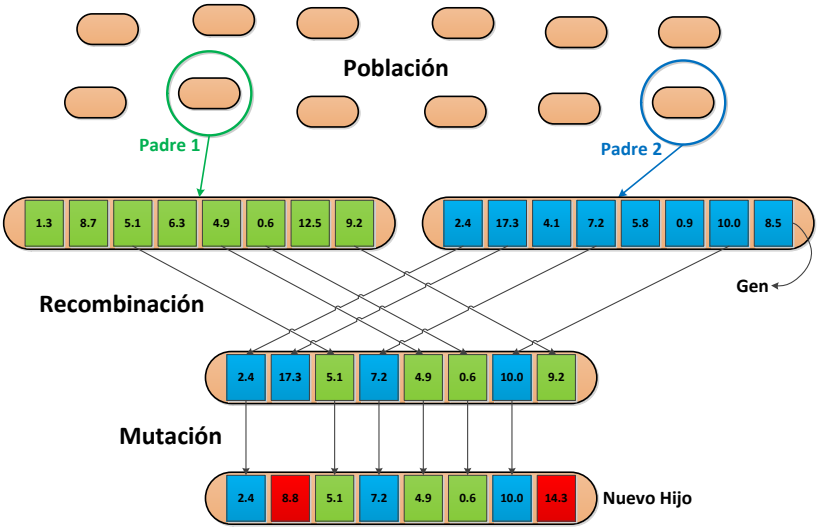

Figura 7.3 – *Esquema reproductivo del algoritmo genético*

Con todo esto, el algoritmo quedaría así:

Algoritmo **DarwinianoMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros

Parámetros:
tamPoblacion, maxIters : N
tasaMortalidad, tasaElitismo, probMutacion, maxTiempo: R

Auxiliar:
poblacion, padres : lista de {params: vector [1..n] de R, eval: R}
p1, p2, hijo : vector [1..n] de R
i, j, numMuertes, numVivos: N
tiempo, primero: R

Salidas:
p : vector [1..n] de R // t-upla óptima de parámetros
o : R *// valor de evaluación de la t-upla óptima*

Algoritmo:

```
i = 0;
o = ∞;

tiempo = 0.0;
ponerRelojEnMarcha();

poblacion = crearPoblacionAleatoria(tamPoblacion, li, ls);

hacer
{
    i = i + 1;
```

```
// evaluar población
desde (j=1 hasta tamPoblacion) hacer
   poblacion[j].eval =  f(w, poblacion[j].params);

ordenarPorValorDeEvaluacion(poblacion);
primero = poblacion[1].eval;

si (primero < o) entonces
{
   o = primero;
   p = poblacion[1].params;
 }

// selección natural
numMuertes = tamPoblacion*tasaMortalidad;
eliminarUltimos(poblacion, numMuertes);
padres = poblacion;

// elitismo
numVivos = tamPoblacion*tasaElitismo;
numMuertes = tamaño(poblacion) – numVivos;
eliminarUltimos(poblacion, numMuertes);

// reproducción
hacer
{
   p1 = elejirPadreAleatoriamente(padres);
   p2 = elejirPadreAleatoriamente(padres);
   hijo = mitosis(p1, p2);

   // mutación
   desde (j=1 hasta n) hacer
      generarMutacion(hijo[j], probMutacion);

   añadir(poblacion, hijo);
```

```
    }
    mientras (tamaño(poblacion) < tamPoblacion)

    tiempo = consultarReloj();
}
mientras ( (i < maxIters) ^ (tiempo < maxTiempo) )
```

Fin de DarwinianoMCA

La condición de terminación es un número máximo de iteraciones, o un tiempo de ejecución no superior a cierto límite. Sin embargo, en los algoritmos genéticos se pueden implementar otro tipo de condiciones de terminación, como un umbral de mejora de la mejor solución actual respecto de la mejor de la anterior generación (por debajo del cual se considera que la evolución se ha estabilizado, aunque esto no es necesariamente cierto), un número máximo de repeticiones de la misma solución (cuando una solución sobrevive generación tras generación, gracias al elitismo, podemos pensar que va a ser siempre la mejor, aunque esto es evidentemente una suposición), o una tasa de mejora de la media de la población, de una generación respecto a la anterior, inferior a un cierto umbral. Todos ellos son criterios posibles y válidos, pero como lo que queremos es ver cuál es la mejor solución que nos puede ofrecer en un cierto tiempo, el criterio temporal sigue siendo el más objetivo para compararlo con otros algoritmos de optimización.

La ejecución de este método debe suponer una mejora sustancial respecto al algoritmo de Monte-Carlo. La combinación de la generación aleatoria inicial, la selección natural eliminando

soluciones malas, la recombinación de soluciones buenas para formar otras que posiblemente mejoren la especie, y la mutación, que permite introducir cierta aleatoriedad en el proceso, para que no haya estancamiento, hacen que en muy pocas generaciones se consigan buenos resultados.

El algoritmo depende de 4 parámetros básicos: tamaño de la población inicial, tasa de mortalidad por selección natural, tasa de elitismo y probabilidad de mutación. El ajuste de estos parámetros se debe de realizar en función del tamaño del espacio de soluciones, la variabilidad de la función de evaluación, y el tiempo disponible:

- tamaño de la población: un tamaño muy grande ralentiza mucho el algoritmo, mientras que uno muy bajo lo hace muy endogámico y poco útil. En principio, el algoritmo debe funcionar bien independientemente de la población inicial generada. Si no lo hace, es que el tamaño es muy bajo.

- tasa de mortalidad: una tasa muy alta hace que se puedan eliminar genes potencialmente válidos, mientras que una tasa muy baja hace que sobrevivan soluciones malas que contaminan las siguientes generaciones.

- tasa de elitismo: una tasa muy alta hace que se produzca una alta endogamia que perjudica la búsqueda de soluciones nuevas potencialmente buenas. La endogamia consiste en que la reproducción se produce siempre entre los mismos individuos, generándose hijos similares. Por el contrario, una tasa muy baja hace que podamos perder individuos genéticamente muy válidos

cuyos genes puede que no vuelvan a aparecer en la población hasta que no transcurran muchas generaciones.

- probabilidad de mutación: una probabilidad alta de mutación echa al traste todo el proceso de selección natural ya que la nueva generación será muy distinta de la anterior, lo cual se asemeja al algoritmo de Monte-Carlo. Por el contrario, una tasa muy baja provocará una dependencia muy alta de la población inicial que hará que el algoritmo funcione bien sólo si la población inicial contiene buenas soluciones (buenos genes).

Aunque ajustar estos parámetros puede llevar un cierto tiempo, por fortuna, la codificación y diseño planteados no es demasiado sensible a los parámetros, y funciona, como demostraremos con posterioridad, razonablemente bien para amplios rangos de valores de los mismos. En cualquier caso, es preferible ajustar 4 parámetros que decenas de ellos, como en cualquier algoritmo de *washout*.

7.3.7 – Algoritmo Coulómbico

Uno de los problemas del algoritmo genético anterior es que, aunque explota de manera muy inteligente información sobre soluciones buenas y las combina para alcanzar mejores soluciones generación tras generación, no explota otra información clave: la información que proporcionan las soluciones malas.

Como desconocemos la forma que tienen los valores de la evaluación del espacio de soluciones, y evaluar cada una de ellas es

muy costoso, el tamaño de las poblaciones no puede ser muy grande porque el rendimiento se degrada.

Aunque existen algunas alternativas para este tipo de situaciones como los algoritmos genéticos con micropoblaciones [182], sería interesante poder disponer de un mecanismo para recordar tanto las soluciones buenas ya probadas como las soluciones malas. Si ya hemos probado con anterioridad una solución y era mala, tiene poco sentido que probemos alguna otra en su vecindad, porque lo más probable es que también sea mala. De igual forma, si hemos probado alguna buena, no estaría mal probar alguna de su vecindad porque es probable que encontremos alguna aún mejor.

Existen algunos algoritmos en la literatura que intentan explotar esta idea, aunque de una forma distinta a la que vamos a plantear nosotros. Uno de ellos es la *búsqueda tabú*, que consiste en seguir un cierto camino y en cada iteración tomar una decisión sobre qué dirección seguir. Ciertas decisiones son vetadas (tabú) según el criterio, por lo que el algoritmo recorre un camino, en principio, descendente (o ascendente si busca el máximo) si el criterio está bien formulado. El problema de este método es su habilidad para quedar estancado en mínimos locales y su pobre rendimiento en espacios muy grandes. Por eso, tras varias pruebas preliminares, no consideramos aplicable este método a nuestro problema.

Como ninguno de los métodos conocidos por el autor explota de manera satisfactoria esta idea, decidimos crear un nuevo de método de optimización adaptado a nuestro problema. El

método ha sido bautizado con el nombre de *algoritmo Coulómbico* porque está inspirado en la ley de Coulomb de la atracción y repulsión de cargas eléctricas. Aunque ha sido ideado para nuestro problema concreto su esquema es genérico y aplicable a otros problemas de optimización continuos y discretos.

La idea del algoritmo es considerar cada solución como una carga eléctrica. Las soluciones buenas tendrán carga positiva y las malas, carga negativa. Todas las soluciones que probemos serán guardadas en una lista de cargas, que nos será de utilidad para la generación de futuras soluciones candidatas.

El funcionamiento del método es el siguiente. Procederemos a hacer los siguientes pasos de forma iterativa. Primero, generamos una solución (en nuestro caso n valores para los n parámetros) pero guiándonos por la información que tuviéramos de otras soluciones ya probadas (almacenadas en la lista de cargas). La excepción a esta regla será en las primeras dos iteraciones, ya que, al no haber probado aún un número suficiente de soluciones, en las dos primeras iteraciones esta generación se realizará de forma completamente aleatoria, porque no tiene sentido consultar una lista de cargas vacía o casi vacía. Después evaluaremos la solución generada. Si ésta mejora la que hasta ahora hayamos encontrado como mejor, ésta pasa a ser la mejor. En cualquier caso, sea buena o mala, nos guardaremos esa información en la lista de soluciones ya evaluadas (en la lista de cargas). Como puede que probemos muchas soluciones, esta lista de cargas debe tener un tamaño máximo, de lo contrario ocupará mucha memoria y costará mucho consultarla. Si se supera el tamaño máximo habrá que gestionar ese caso.

El algoritmo tiene dos pasos clave. El primero es la generación aleatoria, pero guiada por la lista de cargas, de la solución candidata. Esto consistirá en generar una solución aleatoria, y desecharla o no en función de su campo eléctrico, es decir, dependiendo de cuán lejos está de las soluciones buenas, y cómo de lejos de las malas. Si la solución es desechada, no será evaluada y generaremos otra.

¿Cómo distinguiremos las soluciones malas de las buenas si no tenemos un criterio para establecer dicho límite? Nuestra propuesta es dividir por la mitad la lista de cargas (ordenada por su valor de evaluación) y asignar carga positiva a las de la primera mitad, y negativa a las de la segunda. El valor de carga será proporcional a la posición de la solución en el vector. Los valores cercanos al principio del vector tendrán cargas positivas altas, y los valores cercanos al final, tendrán cargas negativas altas. Los valores cercanos a la mitad tendrán cargas positivas bajas si están a la izquierda del valor central, y cargas negativas bajas si están a la derecha.

El cálculo del campo eléctrico se hace siguiendo la ley de Coulomb. Es decir, se tiene en cuenta, tanto la distancia a cada carga (solución) ya evaluada como la carga de las mismas. Para el cálculo de la distancia se debe establecer una manera de medir distancias en el espacio de soluciones. En nuestro caso será la distancia normalizada n-dimensional.

Una vez calculado el campo eléctrico de la solución candidata, se decide si ésta se prueba (se evalúa) o no. Una solución con campo eléctrico negativo tendrá menos

probabilidades de ser evaluada, ya que está más cerca de soluciones malas que buenas, y evaluarla sería probablemente una pérdida de tiempo. Las soluciones con carga eléctrica positiva tendrán muchas posibilidades de ser evaluadas porque están más cerca de soluciones buenas que de las malas, y es posible, por tanto, que también sean buenas.

El segundo aspecto importante del algoritmo es la gestión de la lista de cargas. En principio, esta lista debe ser una lista ordenada de soluciones. El criterio de ordenación será el valor de evaluación. Cada vez que evaluemos una nueva solución la insertaremos (ordenadamente) en la lista de cargas. Como no queremos que la lista alcance un tamaño muy grande, estableceremos un tamaño máximo para la lista y si se supera, eliminaremos soluciones antiguas. Como lo que queremos es que la lista de cargas nos dé información de las soluciones mejores y peores, que son las que dan información útil sobre si vamos bien o mal encaminados, el criterio de eliminación será borrar siempre la solución que ocupe la posición central de la lista. Ésta, en principio, será una solución ni muy buena ni muy mala, y por tanto no proporcionará mucha información.

El algoritmo quedaría, por tanto, de la siguiente manera:

Algoritmo **CoulombicoMCA**

Entradas:
w : W // algoritmo de washout (MCA)
n : N // número de parámetros modificables del algoritmo w
f : W, R^n -------> R // función de evaluación

li : vector [1..n] de R *// límite inferior del rango de los parámetros*
ls : vector [1..n] de R *// límite superior del rango de los parámetros*

Parámetros:
tamMaxLista, maxIters : N
maxTiempo: R

Auxiliar:
listaCargas : lista de {*params: vector [1..n] de* R*, eval:* R}
x : vector [1..n] de R
i : N
tiempo, actual: R

Salidas:
p : vector [1..n] de R *// t-upla óptima de parámetros*
o : R *// valor de evaluación de la t-upla óptima*

Algoritmo:

```
i = 0;
o = ∞;

tiempo = 0.0;
ponerRelojEnMarcha();

mientras ( (i < maxIters) ^ (tiempo < maxTiempo) ) hacer
{
   x = generarValoresGuiados(n, li, ls, listaCargas);
   actual = f(w, x);

   si (actual < o) entonces
   {
      o = actual;
      p = x;
   }
```

```
    // actualización de la lista de cargas
    insertarEnOrden(listaCargas, {x, actual});

    si (tamaño(listaCargas) > tamMaxLista) entonces
        eliminarElementoCentral(listaCargas);

    i = i +1;
    tiempo = consultarReloj();
}
```

Fin de CoulombicoMCA

Función **generarValoresGuiados()**

Entradas:
n : N // número de parámetros modificables del algoritmo w
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros
listaCargas : lista de {params: vector [1..n] de R, eval: R}

Parámetros:
tasaVariabilidad : R

Auxiliar:
probAPriori, probAPosteriori, E, aleat : R
aceptar : booleano

Salidas:
x : vector [1..n] de R // t-upla de parámetros

Algoritmo:

```
aceptar = falso;

hacer
```

```
{
   x = generarValoresAleatoriosEnRangos(n, li, ls);
   probAPriori = 0.5;
   E = calcularCampoElectrico(n, li, ls, listaCargas, x);
   probAPosteriori = probAPriori + atan(E/tasaVariabilidad)/π;

   aleat = generarValorAleatorioEnRango(0.0, 1.0)
   si (aleat ≤ probAPosteriori) entonces
      aceptar = verdadero;
}
mientras (aceptar == falso)
```

Fin de generarValoresGuiados()

Función **calcularCampoElectrico()**

// calcula el campo eléctrico de una carga (t-upla de parámetros) con respecto a todas
// las cargas presentes en la lista de cargas

Entradas:

n : N // número de parámetros modificables del algoritmo w
li : vector [1..n] de R *// límite inferior del rango de los parámetros*
ls : vector [1..n] de R *// límite superior del rango de los parámetros*
listaCargas : lista de {*params: vector [1..n] de* R*, eval:* R}
carga : vector [1..n] de R *// t-upla de parámetros*

Auxiliar:

i , offset, tam, tamMedio: N
E, Q, d : R

Salidas:

E : R

Algoritmo:

```
E = 0.0;
tam = tamaño(listaCargas);
tamMedio = tam/2;

si (tam < 2)
   devolver E;

desde (i = 0 hasta (n-1)) hacer
{
   d = calcularDistancia(n, li, ls, listaCargas[i].params, carga);
   offset = (tam + 1) % 2;

   si (i < tamMedio) entonces
      Q = +1.0*(tamMedio – i)/tamMedio;
   sino
      Q = -1.0*(i + offset – tamMedio)/tamMedio;
   E = E + Q/(d*d);
}
```

Fin de calcularCampoElectrico()

Función **calcularDistancia()**

// calcula la distancia normalizada (de 0.0 a 1.0) entre 2 t-uplas de parámetros

Entradas:

n : N // número de parámetros modificables del algoritmo w
li : vector [1..n] de R // límite inferior del rango de los parámetros
ls : vector [1..n] de R // límite superior del rango de los parámetros
p1 : vector [1..n] de R // t-upla de parámetros
p2 : vector [1..n] de R // t-upla de parámetros

Auxiliar:

i : N
rango, normDist : R

Salidas:
dist : R

Algoritmo:

```
dist = 0.0;

desde (i = 1 hasta n) hacer
{
   rango = ls[i] – li[i];
   normDist = |p1[i] – p2[i]| / rango;
   dist = dist + normDist;
}

dist = dist / n;
```

Fin de calcularDistancia()

Este algoritmo sólo presenta 2 parámetros:

- el tamaño de la lista de cargas: a mayor tamaño mayor número de soluciones almacenadas y por tanto mejor información a la hora de generar soluciones guiadas. Por el contrario, si la lista de cargas es muy pequeña, a poco que el espacio de soluciones sea un poco grande, la generación de soluciones será prácticamente aleatoria y poco ganaremos respecto al método de Monte-Carlo.

- la tasa de variabilidad: este parámetro indica cuánto afecta el campo eléctrico a la probabilidad de aceptar una solución. Si este parámetro tiene un valor muy bajo, la probabilidad de aceptar una solución no se verá prácticamente afectada por el campo eléctrico y su aceptación será prácticamente aleatoria (probabilidad a

posteriori igual a probabilidad a priori). Si tiene un valor alto, el campo eléctrico afectará mucho a la aceptación de la solución (la probabilidad a posteriori será mayor que la probabilidad a priori si el campo eléctrico es positivo, y será menor si es negativo).

7.4 – Evaluación de los Algoritmos

7.4.1 – Corrección de los Algoritmos

Antes de analizar el rendimiento de los algoritmos presentados, primero debemos comprobar que funcionan correctamente, es decir, que efectivamente encuentran el valor mínimo dentro de un determinado espacio de posibles soluciones. Dado que algunos de los algoritmos son heurísticas, tiene poco sentido buscar una demostración formal de que encuentran el mínimo global, ya que sólo una búsqueda exhaustiva de grano muy fino puede garantizarnos eso. Nos conformaremos, pues, con una demostración empírica que nos muestre que son subóptimos.

Para ello, preparamos una sencilla prueba en la que, empleando el algoritmo clásico, comprobamos si los distintos algoritmos de optimización eran capaces de dar como solución los valores esperados. Como el objetivo era comprobar la corrección de los algoritmos de optimización, utilizamos un modelo de plataforma de 3 GdL con las mismas características de la descrita en el capítulo 4, pero suponiéndole movimiento instantáneo, para que el análisis de los resultados fuera más sencillo.

Tomamos como entrada del algoritmo clásico una señal sinusoidal para la fuerza específica en el eje Y de 10 segundos de

duración, 1 Hz de frecuencia y 3 m/s^2 de pico, que vendría a ser la aceleración típica de un turismo. El resto de entradas del algoritmo clásico se fijaron a 0 (excepto la fuerza específica en el eje Z que en ausencia de aceleración queda en 9.8 m/s^2). Usando como indicador la correlación de Pearson normalizada (CCPN) y permitiendo variar sólo 2 parámetros del algoritmo clásico: el límite de inclinación en grados del canal de coordinación (Tilt Coordination – Tilt Limit Y) entre 0 y 20 º, y la frecuencia de corte del filtro pasa-baja del canal de coordinación (LpFilter Y – CutOff), entre 0 y 10 Hz, los algoritmos de optimización propuestos deberían devolver valores cercanos a 10 Hz y a 20 º, ya que, a mayor inclinación y mayor laxitud del filtro, el funcionamiento del algoritmo clásico debe ser, en este caso particular, mejor. El resto de parámetros del algoritmo clásico fueron fijados a valores considerados neutros (3 Hz para el resto de frecuencias de corte, 10 º/s para los límites de velocidad de inclinación, 10 º para el resto de límites de inclinación, y 1 para el resto de parámetros) excepto el límite en la velocidad de inclinación en el eje Y, que fue eliminado para que este parámetro no limitara el resultado final y éste fuera más previsible. Al no haber más señales variables que la del eje Y, no debemos preocuparnos por si la simulación de este movimiento perjudica a las otras direcciones del espacio, ya que las hemos eliminado para estar seguros de cuál es el valor mejor esperable para el algoritmo. Es por ello que el indicador se calculó, en este caso, sólo para la fuerza específica en el eje Y.

Todas las pruebas se realizaron con un Intel Core 2 Quad Q8400 a 2.66 GHz, con 4 Gb de RAM y sistema operativo Windows 7. El simulador de plataforma, ajustado para correr sobre

este equipo, nos permite alcanzar factores de aceleración del tiempo real bastante superiores a 1, lo cual permite ejecutar muchas más pruebas de las que se podrían realizar con una plataforma real. La comparación de señales para calcular el indicador CCPN se realizó utilizando la opción A de la Figura 7.1.

Los resultados obtenidos para cada uno de los algoritmos de optimización, con un tiempo de búsqueda de 500 segundos, se pueden observar en la Tabla 7.1.

Algoritmo	*Valor para*		*Mejor indicador*
	LpFilterAy - cutOff	*tiltCoordY - tiltLimit*	
Búsqueda Exhaustiva	9.99999	16.79999	1.06981
Monte-Carlo	9.98340	17.71427	1.06986
Recocido Simulado	9.99999	16.94035	1.06980
Darwiniano	9.97510	16.95966	1.06980
Coulómbico	9.98260	19.98153	1.06981

Tabla 7.1 – *Validación de los algoritmos de optimización – 500 s*

Como se puede observar, todos los algoritmos encuentran como valores óptimos números en torno a 10 Hz para la frecuencia de corte y en torno a 20 º para el valor del límite de inclinación (valores cercanos al máximo de los intervalos de búsqueda proporcionados). Los valores obtenidos son los esperados, ya que a mayor inclinación y mayor frecuencia de corte la simulación del movimiento, en este caso concreto, será más

fidedigna. Como todos los algoritmos coinciden en sugerir los valores esperados y proporcionan valores similares para el indicador escogido, inferimos que el funcionamiento de los mismos es el correcto. Es reseñable que los valores óptimos no tienen por qué llegar al límite superior del intervalo porque, por ejemplo, para la inclinación, dado que la señal de entrada es sólo de 3 m/s^2, pasado un cierto valor, no se necesita más inclinación. Además, para esta prueba de corrección no se realizó un ajuste detallado de los parámetros de los algoritmos de optimización, eligiéndolos simplemente para que no cayeran en mínimos locales, sin preocuparnos de optimizarlos al máximo. Esto refuerza aún más su corrección, ya que sin preocuparnos de ajustar sus parámetros, todos funcionaron de la forma esperada.

7.4.2 – Estudio de Rendimiento

7.4.2.1 – Búsqueda Exhaustiva Discretizada (BED)

El algoritmo de búsqueda ordenada o búsqueda exhaustiva discretizada es el más fácil de analizar. Si el número de parámetros variables del algoritmo de *washout* no es muy grande, este algoritmo es capaz de discretizar el espacio de soluciones total con un grano no muy grueso que, sin embargo, no requiere un tiempo de exploración muy elevado, por lo que es capaz de encontrar valores relativamente cercanos al óptimo. Por número de parámetros pequeño entendemos un número alrededor de 4 parámetros o menos, porque con un mayor número, el algoritmo empieza a tener un coste computacional muy elevado. Dado que el único parámetro que posee este algoritmo es el número de divisiones de cada intervalo de parámetros del algoritmo de *washout*, a mayor

número de divisiones, mayor coste tiene explorar todo el espacio. Como ya vimos, el coste computacional temporal es del orden de d^n, siendo n el número de parámetros y d, el número de divisiones. Este coste hace que este algoritmo sea, en la práctica, bastante poco útil, ya que, a poco que aumente n, el número de divisiones ha de hacerse tan pequeño para que el algoritmo termine en un tiempo razonable, que la exploración del espacio de soluciones es muy burda y es muy improbable que encuentre valores cercanos al óptimo.

La Tabla 7.2 resume el comportamiento del algoritmo para distintas combinaciones de tiempo y número de parámetros variables del algoritmo de *washout*. El número de divisiones se ajustó en cada prueba según el número de parámetros variables del MCA (a mayor, menor número de divisiones) y el tiempo disponible para su ejecución (a mayor, más divisiones) de forma que la finalización del algoritmo de optimización no fuera por tiempo máximo sino por fin de la búsqueda, aunque a partir de 8 parámetros el algoritmo no es capaz de terminar la búsqueda ni tan siquiera con 2 divisiones. En todas las pruebas se empleó el algoritmo clásico, el simulador de plataforma con el mismo computador que en el aparatado anterior, la plataforma de 3 GdL descrita en el capítulo 4, y señales extraídas de un Formula 3 en base a una vuelta en Montmeló con el simulador rFactor. En este caso el indicador fue el CCPN pero teniendo en cuenta los 6 GdL del espacio, y son los valores de este indicador los que se muestran en la Tabla 7.2.

Tiempo (s)	*Número de parámetros*			
	2	*4*	*8*	*18*
100	1.28257644	1.30962312	1.32078445	1.47133744
200	1.28128842	1.30745662	1.31809866	1.45918723
300	1.28043871	1.30654892	1.31786573	1.43871265
500	1.27977326	1.30123901	1.31762381	1.42732394
1000	1.27528065	1.29993782	1.31744329	1.42689221
10000	1.27462733	1.29892326	1.31721387	1.42639518

Tabla 7.2 - *Análisis del algoritmo de búsqueda exhaustiva*

Se puede observar que cuando el número de parámetros es alto, incrementar mucho el tiempo de búsqueda prácticamente no mejora en nada el rendimiento del algoritmo, ya que en estos casos el problema es que no es capaz de explorar ni una pequeña parte del espacio de soluciones y el algoritmo acaba por superar el tiempo máximo, no por haber completado el rastreo completo del espacio de soluciones. Sin embargo, con 2 ó 4 parámetros, el algoritmo sí es capaz de realizar un barrido completo del espacio de soluciones en algunos casos con cierta solvencia y puede ser el más apropiado. Es por eso que, con este algoritmo de optimización, se da la curiosa circunstancia de obtener peores indicadores empleando un mayor número de parámetros variables del algoritmo MCA, ya que el efecto del aumento de parámetros no puede ser absorbido por la estrategia de búsqueda del algoritmo de optimización. Por ello, si las soluciones óptimas están al final del recorrido del espacio de búsqueda, este algoritmo fracasa estrepitosamente.

7.4.2.2 – Método de Monte-Carlo (MMC)

Este método tiene la ventaja de no tener ningún parámetro que configurar, a excepción de la condición de terminación, que puede ser por número de iteraciones, por tiempo máximo o por convergencia entre soluciones, aunque lo lógico es poner sencillamente un límite de tiempo, por lo que la condición de terminación nunca la consideramos como un parámetro del algoritmo de optimización.

El método puede funcionar algo peor que la búsqueda exhaustiva cuando el número de parámetros es muy pequeño, aunque resulta una mejora sustancial de ésta cuando el número de parámetros comienza a aumentar. Sin embargo, a medida que este número aumenta, el algoritmo se pierde en un espacio de soluciones muy grande y empieza a ser muy improbable que encuentre una solución buena entre tantas posibilidades, ya que funciona de modo aleatorio. Precisamente por ser aleatorio, además, posee una cierta variabilidad, ya que puede encontrar una buena solución en poco tiempo de igual forma que puede pasarse largo tiempo sin encontrar ninguna solución decente. En este último aspecto, el método anterior le aventaja, ya que aquél es completamente determinista.

La Tabla 7.3 ilustra el funcionamiento de este algoritmo de optimización. Al ser un algoritmo probabilístico, los resultados mostrados corresponden al indicador CCPN calculado como media de 20 pruebas de ejecución del mismo, en las mismas condiciones que la prueba realizada para el algoritmo anterior.

Tiempo (s)	*Número de parámetros*			
	2	*4*	*8*	*18*
100	1.29799002	1.29566218	1.29468882	1.29676592
200	1.29602827	1.29438092	1.29435082	1.29454982
300	1.29589211	1.29248535	1.29128071	1.29129324
500	1.29496012	1.29012938	1.29000123	1.28917016
1000	1.29012397	1.28949325	1.28993732	1.28822884
10000	1.28588841	1.28812993	1.28927344	1.28713194
Desviación estándar máxima $\sigma_{máx.} = 0.005213$				

Tabla 7.3 - *Análisis del algoritmo de Monte-Carlo*

Como vemos, el algoritmo se maneja relativamente bien con un número pequeño/medio de parámetros. Con 2 parámetros llega a dar peores valores que la búsqueda ordenada, pero a partir de 4 parámetros la situación se invierte y los indicadores obtenidos son bastante mejores que los que se obtienen con el anterior algoritmo. Aunque las mejoras pueden parecer pequeñas en términos absolutos, una mejora de unas décimas en el indicador objetivo puede reflejar un cambio sustancial en la percepción, y pueden suponer un gran cambio en términos relativos.

Sin embargo, cuando el número de parámetros aumenta, el espacio de búsqueda es tan grande que el algoritmo no es capaz de encontrar grandes mejoras con tiempos de ejecución superiores y su rendimiento es prácticamente constante. La variabilidad, si bien existe, no es demasiado elevada, ya que la desviación estándar máxima de toda la serie se sitúa en 0.005213, lo cual en términos relativos representa en torno al 4% de los valores medios obtenidos.

7.4.2.3 – Recocido Simulado (RS)

Este algoritmo es una mejora del algoritmo de Monte-Carlo ya que a diferencia del anterior no se basa solamente en el azar. En las pruebas realizadas (en las mismas condiciones que los anteriores algoritmos) se observa que es capaz de proporcionar mejores valores que el algoritmo de Monte-Carlo (ver Tabla 7.4). Los valores presentados son también valores medios de 20 pruebas.

Tiempo (s)	*Número de parámetros*			
	2	*4*	*8*	*18*
100	1.30541287	1.29723823	1.30198491	1.29666865
200	1.29790291	1.29438362	1.30095403	1.29123833
300	1.29422071	1.29283022	1.29574667	1.28672632
500	1.29112762	1.28908371	1.29312938	1.28232801
1000	1.28672892	1.28763512	1.28653482	1.27652782
10000	1.28274593	1.28517803	1.28174512	1.27323125
Desviación estándar máxima $\sigma_{máx.} = 0.017432$				

Tabla 7.4 - *Análisis del algoritmo de Recocido Simulado*

Se observa que su rendimiento mejora cuando hay un número alto de parámetros y sobre todo a mayor tiempo disponible. En cualquier caso, la mejora obtenida es bastante menor de la esperada. Por el contrario, con un número pequeño de parámetros y/o poco tiempo de ejecución obtiene resultados a veces peores que el algoritmo de Monte-Carlo.

Se observa, además, que la variabilidad de los resultados es relativamente alta. Esto es debido a que este tipo de algoritmos es propenso a quedarse estancado en mínimos locales y tiene cierta dependencia del valor inicial. La desviación estándar máxima de la serie se sitúa en 0.017432, lo cual representa alrededor de un 13 % sobre los valores medios. Si bien, este dato es bastante elevado, también es cierto que la variabilidad disminuye a medida que se aumenta el tiempo de ejecución.

Todas las pruebas se realizaron con una constante de temperatura de 1000, una temperatura inicial de 0º, una temperatura final de 100º, un ratio de modificación de 0.2 y 5 repeticiones como número máximo de repeticiones. El número de iteraciones se ajustó para que coincidiera aproximadamente con el tiempo de ejecución. Estos valores se ajustaron con pruebas previas, que por brevedad, no incluimos. Es importante recalcar que, el hecho de tener varios parámetros que ajustar y que su rendimiento sea bastante variable, son dos desventajas a tener en cuenta.

7.4.2.4 – Algoritmo Darwiniano o Genético (AD)

Dado que en pruebas preliminares el algoritmo demostró comportarse bastante mejor que los demás, y dado que presenta un pequeño grupo de parámetros (ligeramente superior al resto de algoritmos estudiados), es conveniente estudiar cuáles son los valores apropiados para los mismos y si sus valores alteran mucho el resultado del algoritmo, antes de realizar las pruebas de rendimiento. Todas las pruebas se realizaron en las mismas

condiciones que los anteriores algoritmos, pero siempre con 12 parámetros variables.

La primera prueba consiste en analizar cómo afecta la probabilidad de mutación al algoritmo. Para ello, realizamos 100 pruebas variando la probabilidad de mutación y fijando la población a 10 individuos, con un porcentaje de elitismo del 20 % y un ratio de supervivencia del 40 %. Los indicadores obtenidos para cada caso se pueden observar en la Tabla 7.5 (continuamos usando el indicador CCPN).

Prob. de mutación	*Tiempo (s)*		
	100	*500*	*1000*
0.0	1.29773473	1.28406155	1.27013425
0.1	1.29186856	1.26885891	1.26146823
0.2	1.28162205	1.26411616	1.25575628
0.3	1.28256654	1.26903498	1.26083273
0.4	1.28920898	1.26535713	1.26098337
0.5	1.28928972	1.28407394	1.28105433
0.75	1.29268848	1.28596079	1.28322867
1.0	1.31666698	1.29905057	1.28509872

Tabla 7.5 - *Análisis de la probabilidad de mutación*

Los resultados dan a entender que la probabilidad de mutación más adecuada está alrededor de 0.2. Valores mayores lo hacen demasiado aleatorio, y menores demasiado endogámico. En cualquier caso, el algoritmo parece funcionar para un amplio espectro de valores.

El siguiente análisis es análogo, pero variando el tamaño de la población y fijando la probabilidad de mutación a 0.2, con un porcentaje de elitismo del 20 % y un ratio de supervivencia del 40 %. Podemos ver los indicadores en la Tabla 7.6.

Tamaño de población	*Tiempo (s)*		
	100	*500*	*1000*
5	1.28885891	1.26954079	1.26089322
10	1.28162205	1.26411616	1.25575628
20	1.27167117	1.26251781	1.25220547
30	1.27251136	1.26961250	1.25338095
50	1.27345049	1.27823804	1.25580431
100	1.29309722	1.28097652	1.26064545

Tabla 7.6 - *Análisis del tamaño de población*

Los resultados nos indican que tamaños grandes de población no son siempre adecuados. En este caso, al ser la función de evaluación muy costosa, es suficiente con poblaciones de entre 10 y 20 individuos. Sin embargo, a medida que el tiempo disponible es mayor, el uso de poblaciones grandes empieza a ser mejor. En otros problemas con funciones de evaluación más ligeras es habitual ver poblaciones de miles de individuos pero aquí no es deseable, porque tardaríamos horas en simular cada generación. Por ello, el tamaño de la población viene influido por el tiempo disponible y por el coste de evaluar la función de evaluación. Como en este caso la función de evaluación tiene que comparar una vuelta completa a un circuito (unos 90 segundos), aunque la simulación se haga más rápido que tiempo real, el coste computacional se degrada mucho. Al igual que en la prueba

anterior, es importante recalcar que el algoritmo funciona bien para un amplio espectro de tamaños de poblaciones, incluso para valores radicalmente bajos, como 5 individuos.

La siguiente prueba analiza cuál es el porcentaje más adecuado de elitismo, fijando la probabilidad de mutación a 0.2, con un tamaño de población de 20 individuos y un ratio de supervivencia del 40 %. Los indicadores obtenidos se muestran en la Tabla 7.7.

Porcentaje de elitismo	*Tiempo (s)*		
	100	*500*	*1000*
0 %	1.27230832	1.27153091	1.27033421
10 %	1.26859140	1.26408736	1.26537025
20 %	1.27167117	1.26251781	1.25220547
30 %	1.27258398	1.26417744	1.26790035
40 %	1.27633392	1.27080142	1.26890384
50 %	1.28290421	1.27348189	1.27128053

Tabla 7.7 - *Análisis del porcentaje de elitismo*

Los resultados dan a entender que un porcentaje en torno al 20 % es lo ideal para esta aplicación. Valores superiores hacen al algoritmo algo endogámico, y valores inferiores lo hacen más simple ya que pierde las ventajas de recordar soluciones buenas encontradas previamente. En cualquier caso, las diferencias entre los diferentes valores no son muy grandes, y el algoritmo funciona bien incluso sin elitismo.

Por último, analizamos el porcentaje de selección natural, fijando la probabilidad de mutación a 0.2, con un tamaño de población de 20 individuos y un porcentaje de elitismo del 20 %. Los resultados se pueden observar en la Tabla 7.8.

Porcentaje de selección natural	*Tiempo (s)*		
	100	*500*	*1000*
0 %	1.30067195	1.29120967	1.28862341
10 %	1.27094757	1.26326310	1.27653987
20 %	1.26837074	1.26558386	1.25673449
40 %	1.27167117	1.26251781	1.25220547
50 %	1.27829313	1.26970088	1.25937544
75 %	1.28706218	1.27038745	1.26987632

Tabla 7.8 - *Análisis del porcentaje de selección natural*

Los valores obtenidos nos dan a entender que el valor ideal de selección natural se sitúa en torno a un 40 %, aunque, al igual que en anteriores casos, el algoritmo funciona bastante bien para casi todos los valores.

Con este análisis, ya podemos fijar con cierta confianza los parámetros del algoritmo genético. Dichos valores se resumen y presentan en la Tabla 7.9.

Y con estos parámetros podemos repetir el mismo análisis que hicimos con el resto de algoritmos, obteniendo los resultados que podemos observar en la Tabla 7.10.

Parámetro	*Valor*
Probabilidad de mutación	0.2
Tamaño de población	20
Porcentaje de elitismo	20 %
Porcentaje de selección natural	40 %

Tabla 7.9 - *Parámetros elegidos para el algoritmo genético*

Tiempo (s)	*Número de parámetros*			
	2	*4*	*8*	*18*
100	1.28283178	1.28691323	1.29114046	1.28719175
200	1.28192831	1.28347992	1.28836286	1.27546432
300	1.28012839	1.28038452	1.28152083	1.27098391
500	1.27906072	1.27898394	1.27349029	1.26312893
1000	1.27639782	1.27472465	1.26992023	1.25172981
10000	1.27556721	1.27012297	1.26502301	1.24171954
Desviación estándar máxima $\sigma_{máx} = 0.005028$				

Tabla 7.10 - *Análisis del algoritmo genético*

Como se puede observar, el algoritmo genético obtiene buenos resultados en todo el espectro de situaciones, siendo especialmente efectivo cuando el número de parámetros es alto. Aunque sus ventajas se ponen de manifiesto cuando el tiempo de ejecución es relativamente alto, ya que su estrategia de exploración es mejor que la de los anteriores algoritmos, también se comporta bien con tiempos de ejecución pequeños y con pocos parámetros, aunque en este último caso las ventajas de este algoritmo con respecto a otros, se diluyen. Tanto es así, que para 2 parámetros el

algoritmo de búsqueda ordenada se comporta ligeramente mejor que éste. En cualquier caso, el hecho de que se comporte bien en todas las situaciones hace que sea el más idóneo, en cuanto a rendimiento, de los analizados. Al ser una heurística preparada para funcionar en espacios de soluciones grandes, esto no debería sorprendernos.

Además, la variabilidad del algoritmo es más baja que en el caso del anterior algoritmo, ya que aunque tiene una componente aleatoria, tiende a converger hacia soluciones óptimas globalmente. Esto hace que los mejores indicadores se obtengan, claramente, cuando el algoritmo MCA presenta un mayor número de parámetros variables.

La desviación estándar máxima es similar a la del algoritmo de Monte-Carlo, si bien es cierto que ésta disminuye bastante a medida que aumenta el tiempo de ejecución, lo cual es un dato positivo.

7.4.2.5 – Algoritmo Coulómbico (AC)

El análisis del rendimiento de este algoritmo se puede observar en la Tabla 7.11 que resume los resultados de las pruebas realizadas con él, en las mismas condiciones y con los mismos parámetros del algoritmo de *washout* que las realizadas con todos los demás algoritmos. Para las pruebas se fijó un tamaño máximo de lista de cargas de 10000 elementos y una tasa de variabilidad de 100.

Tiempo (s)	*Número de parámetros*			
	2	*4*	*8*	*18*
100	1.29451224	1.29422971	1.29288084	1.29126548
200	1.29023902	1.29348294	1.29013389	1.29023803
300	1.28882931	1.28944123	1.28828859	1.28939238
500	1.28671231	1.28827453	1.28652341	1.28672632
1000	1.28490233	1.28672638	1.28455982	1.28312391
10000	1.28012399	1.28451289	1.28314513	1.28005027
Desviación estándar máxima $\sigma_{máx} = 0.003141$				

Tabla 7.11 - *Análisis del algoritmo Coulómbico*

Los resultados demuestran que este algoritmo representa una mejora del algoritmo de Monte-Carlo. Al recordar soluciones malas ya probadas, es lógico que mejore el rendimiento del mismo. Podemos ver, sin embargo, que cuando el número de parámetros es alto, no puede competir con el algoritmo genético. Resulta llamativo observar que se comporta bastante bien con tiempos bajos de ejecución (por ejemplo con 8 parámetros y 100 ó 200 segundos se comporta casi igual que el genético), ya que una de las características del mismo es que, cuando la longitud temporal de las señales a comparar es bastante grande con respecto al tiempo máximo de búsqueda, este algoritmo es capaz de evitar ciertas pruebas innecesarias (y largas) de soluciones parecidas a otras ya probadas y que se demostraron malas. Eso le confiere una ventaja sustancial en este aspecto con respecto a otros algoritmos. Teniendo en cuenta que, además, dar valores a sus parámetros es sencillo, ya que sólo tiene 2 y con efectos bastante intuitivos, es un algoritmo a tener en cuenta en ciertas circunstancias, dado nuestro objetivo de automatizar el ajuste de parámetros en MCA.

Su variabilidad máxima es la menor de todos los algoritmos no deterministas probados. Aunque ésta no disminuye tanto con el tiempo de ejecución ni con el aumento del número de parámetros, este hecho lo convierte en un algoritmo fiable ya que es bastante improbable que nos proporcione una solución muy alejada de la óptima.

7.4.3 – Estudio Comparativo

Para comparar todos los anteriores algoritmos y estudiar qué diferencias puede haber entre la optimización con distintos indicadores, decidimos realizar un estudio comparativo exhaustivo.

Para la comparación, empleando el algoritmo clásico variando 18 de sus parámetros, empleando el mismo computador que en anteriores pruebas, el simulador *Live for Speed*, con un *BMW FB02* en el circuito de *Blackwood* y tomando 10000 segundos como tiempo de búsqueda, estudiamos el rendimiento de los 5 algoritmos propuestos con 2 plataformas de movimiento distintas y los siguientes indicadores: correlación (CCPN), error absoluto medio normalizado (EAMN), escalado medio (EM), retraso estimado (RE) y una combinación aditiva equiponderada de escalado medio, error absoluto medio normalizado y correlación. Los resultados se pueden ver en las siguientes tablas. La Tabla 7.12 muestra los resultados con respecto a la plataforma T1R2 que hemos estado empleando en las anteriores pruebas. La Tabla 7.13 muestra los resultados para la plataforma TR3R que empleamos en el capítulo 6.

Algoritmo	*Indicador*				
	CCPN	*EAMN*	*EM*	*RE*	*Combinado*
BED	1.41639519	1.17779350	433.394897	2.10833240	846.4072266
MMC	1.28613193	1.16832530	16.7562637	1.16333330	37.00113678
RS	1.28323125	1.16888952	14.1790847	1.15833330	32.58115005
AD	1.25123630	1.16573140	9.18358135	1.10500002	16.56641961
AC	1.28005027	1.16874182	15.3242721	1.14499998	30.34094814

Tabla 7.12 - *Estudio comparativo exhaustivo para la plataforma T1R2*

Algoritmo	*Indicador*				
	CCPN	*EAMN*	*EM*	*RE*	*Combinado*
BED	1.41233396	1.17876446	22.7032623	1.18833327	40.58589935
MMC	1.29471755	1.17191803	4.97333307	1.15333332	9.935739517
RS	1.29679143	1.17161357	4.25311995	1.14666665	8.065122604
AD	1.26451409	1.16789925	3.66799736	1.07833332	5.286560059
AC	1.28988612	1.17031241	4.75598221	1.11499989	8.731724739

Tabla 7.13 - *Estudio comparativo exhaustivo para la plataforma T3R3*

De este estudio se pueden deducir varias cosas. La primera es que el indicador elegido afecta obviamente al resultado obtenido, aunque no parece afectar al rendimiento de los algoritmos, que dependen más de su estrategia de búsqueda y de sus problemas intrínsecos que del indicador, aunque es obvio que un cambio de indicador puede cambiar la forma del espacio de soluciones y alterar ligeramente el rendimiento de los diferentes algoritmos de optimización.

La segunda conclusión es que la diferencia entre 3 y 6 GdL no es muy grande en términos de correlación ni de diferencia de señales. De hecho, en algunas ocasiones saca mejor puntuación la plataforma T1R2. Sin embargo, el uso de una plataforma de 6 GdL da mejores resultados en cuanto al escalado medio de las señales. Esto es lógico si tenemos en cuenta que en el cálculo de los diferentes indicadores se incluyen todas las señales y en la plataforma de 3 GdL hay 1 GdL que no puede simularse de ninguna manera y 2 que están más limitados que en la de 6. El indicador combinado también incluye este efecto. El retraso también es ligeramente mejor con la T3R3, pero, como hemos comentado varias veces, habría que analizar con mucho cuidado si estas pequeñas mejoras que proporciona la plataforma de 6 GdL compensan a nivel de inversión económica.

En cuanto a los algoritmos de optimización, el algoritmo genético es el que mejor funciona en casi todos los casos. El de búsqueda ordenada es el peor en todos los casos. El algoritmo de Monte-Carlo siempre es un poco peor que el Coulómbico, y entre éste y el Recocido Simulado hay pequeñas diferencias, por lo que la balanza se decanta de uno u otro lado en función del indicador escogido. Hemos de recordar que esta prueba se realizó con 18 parámetros lo cual supone un número elevado y no es la condición más adecuada para el algoritmo Coulómbico.

Para comprobar las diferencias que se pueden obtener empleando el otro modelo de comparación que comentamos a principio del capítulo, repetimos la prueba anterior empleando un modelo de percepción y calculando todos los indicadores con respecto a la posición de la cabeza del piloto. Dicho modelo de

percepción corresponde exactamente con los modelos de otolito y canal semicircular descritos en el capítulo 2, empleando los mismos parámetros que Reid y Nahon. Por brevedad, sólo se realizó la prueba con la plataforma de 6 GdL. Los resultados muestran pocas diferencias cualitativas con respecto a la situación en la que no se emplea modelo de percepción.

Como ya comentamos, para simuladores de pequeños vehículos no aéreos, como los que nos ocupan en este trabajo, las diferencias deben ser pequeñas entre una aproximación y otra. Si acaso, se puede observar una cierta tendencia a la reducción entre las diferencias de los distintos valores subóptimos que los algoritmos de optimización encuentran, cosa que puede explicarse por la forma en que los modelos modifican las señales, ya que eliminan cierta información de las señales y presentan ciertos umbrales mínimos de percepción. Los indicadores obtenidos se muestran en la Tabla 7.14.

Algoritmo	*Indicador*				
	CCPN	*EAMN*	*EM*	*RE*	*Combinado*
BED	1.43592832	1.16897045	20.7334497	1.18982292	39.86212350
MMC	1.30537825	1.17087398	6.88923409	1.15983760	11.90661235
RS	1.29875490	1.16798304	5.23908467	1.14558239	10.90878329
AD	1.27873794	1.16692513	3.98652391	1.05188101	6.992093420
AC	1.30048736	1.17343505	5.34809876	1.13065286	8.309739283

Tabla 7.14 - *Estudio comparativo exhaustivo para la plataforma T3R3 con modelo de percepción*

7.4.4 – Validación del Ajuste

Para comprobar si efectivamente el ajuste automático de parámetros conlleva ventajas con respecto al método prueba-error tradicional, decidimos realizar una prueba final en la que, con la plataforma de 6 GdL comparamos la opinión de diversas personas sobre un ajuste manual realizado por un experto con respecto al ajuste automático. Dado que los indicadores objetivos han sido validados por la caracterización del capítulo 6, y la corrección de los algoritmos de optimización ha sido probada, creemos que en este punto lo más importante es probar cómo afecta el tiempo disponible para el ajuste de parámetros en dicha tarea. Por ello realizamos la comparación entre el ajuste manual y el automático para diversos tiempos de ajuste.

Para ello, un pequeño grupo de 20 personas fue invitado a probar el simulador rFactor en Montmeló con un Fórmula 1, con el algoritmo clásico. El computador empleado es un Intel Core i7 a 3700 Mhz, con 8 Gb de RAM, tarjeta gráfica Nvidia GTX 690 y sistema operativo Windows 7.

En una primera fase, al piloto y al experto se les concedió un determinado tiempo para ajustar el algoritmo. Pasado ese tiempo se guardaron los valores elegidos para el ajuste por cada persona, anotando también el tiempo que se les había concedido para el mismo. Una semana más tarde, las mismas personas fueron invitadas a probar el mismo simulador durante 2 tandas de 10 minutos por cada ajuste manual realizado. En una de las tandas se le presentaba el simulador ajustado por él, mientras que en la otra se le presentaba el simulador ajustado de forma automática con el

algoritmo de optimización genético, el indicador combinado que vimos en el punto anterior, variando 18 parámetros y empleando exactamente el mismo tiempo de ajuste. Al piloto no se le informaba de cuál de las tandas era la que correspondía con el ajuste manual para que el ensayo fuera ciego, y se le preguntaba cuál de las dos pruebas le había resultado más convincente a nivel de simulación de movimiento. Para el ajuste manual se partía de una configuración elegida por el experto.

Todo ello se realizó con 4 tiempos de ajuste distintos: 5 minutos, 20 minutos, 1 hora y 3 horas. Los resultados se pueden ver en la Tabla 7.15, donde se muestra el número de personas cuya preferencia se decanta por el ajuste automático, por el manual, o dudan y no son capaces de dar una respuesta, en función del tiempo de ajuste disponible.

Tiempo (mins)	*5*	*20*	*60*	*180*
Ajuste automático	18	11	7	2
Ajuste manual	1	4	9	17
No sabe/No contesta	1	5	4	1

Tabla 7.15 - *Estudio comparativo de los métodos de ajuste*

Los resultados ponen de manifiesto que es preferible el ajuste automático cuando se dispone de poco tiempo, mientras que el ajuste manual se prefiere cuando se dispone de mucho más tiempo para ajustar la plataforma al gusto del piloto. En este sentido es importante destacar que el ajuste manual es individualizado mientras que el automático no, por lo que en 5

minutos es capaz de dar un conjunto de parámetros aceptable, mientras que de modo manual no hay tiempo material para ello. Por el contrario, en varias horas se puede realizar un ajuste más personalizado para el piloto. Cuando el tiempo de ajuste aumenta mucho, el ajuste automático puede probar muchas combinaciones, pero por muchas que pruebe siempre será un ajuste de *talla única*, mientras que el manual acaba siendo *un traje a la medida del piloto*.

El ajuste objetivo automático también puede ser útil cuando no se disponga de pilotos, pero en general, no se puede concluir que pueda sustituir al ajuste subjetivo, cosa que no debe sorprendernos en absoluto. Es más, la combinación de ambos métodos, empleando el ajuste objetivo como punto de partida del subjetivo, podría ser una buena forma de generar sinergias en este problema.

7.5 – Conclusiones

De todas las pruebas presentadas en este capítulo se pueden extraer varias conclusiones. La primera es que el algoritmo genético es el que mejor funciona con espacios de soluciones muy grandes. El mayor problema de este algoritmo es que tiene varios parámetros, y aunque no son muchos y sus valores pueden fluctuar en rangos más o menos grandes sin que el algoritmo deje de funcionar bien, sería más interesante que no tuviéramos que ajustar ningún parámetro, ya que eso es precisamente lo que queremos evitar de los algoritmos MCA. En cualquier caso, es el que demuestra una mayor inteligencia a la hora de buscar soluciones, y es capaz de encontrar soluciones bastante buenas en muy pocas

iteraciones, cosa que en otros algoritmos no siempre pasa. Su uso compensa cuando hay muchos parámetros que ajustar.

El algoritmo Coulómbico funciona bien con espacios de soluciones más pequeños ya que la lista de cargas le ayuda a evitar evaluar una gran cantidad de soluciones malas como hacen otros algoritmos. Esto permite aprovechar el tiempo en evaluar soluciones potencialmente buenas. El contrapunto está en que la generación de nuevas soluciones candidatas es pseudo-aleatoria y con espacios de soluciones muy grandes, el algoritmo pierde eficacia respecto al genético. La principal ventaja del algoritmo es que sólo tiene 2 parámetros que son muy sencillos de ajustar. Es menos variable que el algoritmo de Monte-Carlo y más fácil de configurar que el algoritmo genético. Compensa cuando el número de parámetros no es muy grande y el tiempo de las señales es bastante grande con respecto al tiempo máximo de búsqueda del algoritmo.

El Recocido Simulado tiene un rendimiento no demasiado brillante con este problema. Es claramente peor que el algoritmo genético y no tiene muchas ventajas adicionales respecto a éste, ya que también presenta demasiados parámetros que configurar y su rendimiento es el menos estable de todos. La explicación a su no buen rendimiento puede estar en que este método es muy propenso a caer en mínimos locales (cristalización prematura) si los parámetros de enfriamiento y temperatura no se ajustan con mucho acierto, cosa que no es sencilla y de tener que hacerlo desvirtuaría el objetivo pretendido, ya que el objetivo es realizar un ajuste lo más desatendido posible y con este método se deben ajustar varios parámetros del algoritmo de optimización, siendo,

además, bastante sensible a ellos, cosa que no ocurre en tan gran medida con otros algoritmos evaluados.

Por último, aunque el algoritmo de Monte-Carlo puede ser utilizado y tiene la ventaja de no tener parámetros, el algoritmo Coulómbico es una alternativa mucho mejor. Aunque el algoritmo de Monte-Carlo funciona, puede tardar demasiado tiempo en ofrecer una solución suficientemente buena y, a diferencia de otros algoritmos, la solución no se va refinando asintóticamente sino que es puramente aleatoria.

En cuanto a la metodología de ajuste, las pruebas nos sugieren que el ajuste automático puede ser una mejora respecto al ajuste tradicional y permite ahorrar tiempo, aunque no debemos plantearlo como un sustituto completo del ajuste subjetivo manual sino como un complemento, ya que todo depende del tiempo que dispongamos para el ajuste.

Respecto a los indicadores, hemos podido comprobar que no afectan a los métodos de optimización, cosa que tampoco ocurre con un cambio de plataforma. Además, las diferencias entre los indicadores calculados para las dos plataformas analizadas son más pequeñas de lo que uno podría pensar en un principio, ya que resulta más importante que la plataforma sea capaz de generar los movimientos de forma rápida y con una extensión razonable, que tener muchos GdL tan poco extensos que se conviertan en prácticamente inservibles. La conclusión más importante es, pues, que la rapidez de movimiento es más importante que el número de GdL.

Al hilo de este hecho y como comentario final, apuntar que en las numerosas pruebas realizadas, los datos objetivos sugieren que es mejor no generar movimiento que generarlo muy retrasado o incorrectamente. Un modelo de plataforma con poca fuerza en los motores, y por tanto con mucho retraso, induce a los algoritmos de optimización de parámetros a generar valores que anulan todo movimiento, lo cual sugiere que es mejor no generar movimiento que hacerlo mal o muy retrasado, cosa que en realidad resulta bastante lógica y que coincide con lo expresado por otros autores [109]. Lo mismo sucede cuando le pedimos a la plataforma simular aceleraciones muy grandes, que por construcción física nunca va a poder simular. En este caso, los algoritmos de optimización evalúan mejor en algunos casos la ausencia de movimiento que la simulación de un movimiento que tenga grandes diferencias con el esperado, por lo que sugieren valores para los parámetros que prácticamente anulan el movimiento de la plataforma. Aunque depende del indicador escogido, este hecho debe tenerse en cuenta a la hora de diseñar, construir o adquirir una plataforma de movimiento.

Capítulo 8

Conclusiones y Trabajo Futuro

En este octavo y último capítulo trataremos de resumir las conclusiones y aportaciones más significativas de este trabajo, para finalmente indicar posteriores líneas de investigación futuras.

8.1 – Conclusiones

8.1.1 – Conclusiones Generales

Teniendo en mente que el objetivo principal del trabajo era aportar mejoras en los sistemas de generación de claves gravito-inerciales en simuladores de vehículos no aéreos, podemos concluir que dicho objetivo ha quedado razonablemente cumplido por los motivos que exponemos a continuación.

En primer lugar, hemos descrito un procedimiento para analizar las necesidades gravito-inerciales de un simulador de vehículos que nos permite estudiar y conocer cuáles son para cada simulador las características dinámicas que lo definen mejor y en las que, por tanto, debemos poner más empeño en reproducir. Hemos ejemplificado dicho estudio en un simulador no aéreo para poder obtener datos tangibles de un simulador concreto pero lo importante es disponer de herramientas útiles para identificar qué es más importante simular en cada caso. Es decir, lo importante no

son las conclusiones de la caracterización particular del vehículo concreto analizado, sino el procedimiento para deducir dichas conclusiones, que nos permitirá analizar cualquier otro simulador de vehículos.

Con el estudio de las necesidades gravito-inerciales de un simulador, disponemos ya de información suficiente para escoger el manipulador apropiado. Es por ello, que el siguiente paso lógico es analizar dicho manipulador, orientando este análisis a su uso en simulación. Es importante esta puntualización ya que existen multitud de trabajos publicados analizando dispositivos similares, pero muy pocos haciéndolo desde el punto de vista de su uso en simulación. De dicho análisis, descrito en el capítulo 4, pudimos sacar algunas conclusiones útiles como, por ejemplo, una manera de optimizar la formulación del manipulador escogido para ajustar los rangos de los GdL a nuestras necesidades. Además, también propusimos una forma de analizar su espacio de soluciones para poder emplear la plataforma de una forma más sencilla sin tener que preocuparnos de la interdependencia de los GdL. Todo ello redunda en una mejora del aprovechamiento de la plataforma de movimiento y por tanto en la mejora de la generación de claves gravito-inerciales.

Posteriormente, desarrollamos una serie de métricas en base a una caracterización de la respuesta humana ante algoritmos MCA. Esto nos ha ayudado a validar la estrategia empleada por dichos algoritmos de *washout* y a comprender qué indicadores objetivos pueden representar mejor la percepción de este tipo de algoritmos. Entre ellos destaca la correlación de Pearson entre las señales de salida del algoritmo y las señales que éste debería

generar idealmente. La conclusión más importante en este apartado es que lo importante es generar señales parecidas en forma a las reales aunque sea con magnitudes distintas, intentando al mismo tiempo que dicha diferencia de magnitud no sea muy elevada. Dicho análisis fue validado con dos simuladores distintos por lo que podemos inferir que sus conclusiones son válidas a nivel general.

Por último, hemos podido comprobar que los indicadores (métricas) objetivos nos sirven para realizar ajustes de los parámetros de los algoritmos de *washout* de forma más sencilla. Estudiamos diversas alternativas y concluimos que el algoritmo genético de optimización es el que mejor se ajusta al problema ya que es capaz de aportar un conjunto de parámetros satisfactorio en poco tiempo, y comprobamos que el ajuste automático mejora al método de ajuste manual cuando el tiempo disponible para dicho ajuste es relativamente pequeño.

8.1.2 – Conclusiones Particulares

Todo lo anteriormente citado se refiere a conclusiones generales aplicables a cualquier tipo de simulador. Sin embargo, siguiendo nuestro objetivo de estudiar simuladores de vehículos no aéreos, hemos ejemplificado varios de estos estudios con un simulador en particular.

En concreto, aprovechamos el capítulo 3 para realizar un análisis de las necesidades gravito-inerciales de un simulador de un vehículo particular: un bote de rescate rápido. Las conclusiones

particulares que extrajimos de este análisis se pueden resumir en una ausencia de aceleraciones sostenidas y en una mayor importancia de los GdL rotacionales, especialmente *pitch* y *roll*. También pudimos deducir los limites frecuenciales de los filtros de corte que un algoritmo de *washout* necesitaría en función de los desplazamientos máximos admitidos por cada plataforma de movimiento. Sin embargo, más allá de las consideraciones particulares deducidas para este tipo concreto de simulador, la conclusión más importante es la formalización de un procedimiento para analizar las necesidades gravito-inerciales de un simulador de vehículos.

Además, hemos desarrollado las ecuaciones físicas del citado bote de rescate que nos permitirán implementar simuladores de vehículos de este tipo, lo cual ya se puede considerar una mejora en la generación de claves gravito-inerciales de vehículos no aéreos, ya que, al menos por lo que el autor conoce, no existía ningún simulador de este tipo publicado con tanto detalle.

Con el estudio de las necesidades gravito-inerciales del simulador de bote de rescate dedujimos que era conveniente emplear, al menos, una plataforma de 3 GdL *heave-pitch-roll*, por lo cual aprovechamos esta conclusión para analizar un dispositivo concreto de este tipo. El exhaustivo análisis de este dispositivo nos permite conocer mejor su funcionamiento y aplicar mejoras sobre su espacio de soluciones para obtener un mayor aprovechamiento del mismo. El análisis de la cinemática inversa y las ecuaciones implicadas no se puede generalizar, pero el estudio posterior sobre las zonas seguras y el procedimiento elegido es perfectamente aplicable a otros simuladores.

8.2 – Aportaciones y Discusión

8.2.1 – Aportaciones Generales y Discusión

Más que ofrecer nuevas soluciones al problema de la generación de claves gravito-inerciales proponiendo un nuevo algoritmo o una nueva plataforma de movimiento, lo que se ha intentado es realizar un estudio bastante panorámico del problema global y ofrecer un número de aportaciones al problema que, en conjunto, constituyan una aportación significativa para mejorar las soluciones a este tipo de problemas.

Como aportes más significativos y generales podemos citar el desarrollo de una metodología para la identificación de las necesidades gravito-inerciales de un simulador, que aporta elementos novedosos, si bien está inspirado en el trabajo de Reymond y Kemeny. Aunque es cierto que es muy complicado generalizar dicho proceso y las conclusiones extraídas son, en algunos casos, cualitativas y por tanto discutibles, la aplicación del mismo al campo de la simulación en el medio marino y el intento de generalización del proceso son elementos novedosos. En este punto, es importante destacar que, para simuladores de otro tipo, pueden existir particularidades no reflejadas en este estudio, aunque, en cualquier caso, creemos que la aportación es suficientemente genérica.

Siguiendo con las aportaciones generales debemos citar la caracterización humana ante algoritmos de *washout*, el desarrollo de métricas objetivas y el uso de las mismas para la implementación de métodos automáticos de exploración y optimización del espacio

de parámetros de los MCA para el ajuste de los mismos, que constituyen un aporte completamente novedoso, especialmente este último. Si bien este es un campo tremendamente extenso y sobre el que quedan muchísimos elementos que estudiar, creemos que es bueno abrir una línea de investigación en esta dirección. Sin embargo, es cierto que, sobre todo en el desarrollo de métricas y en el empleo de modelos de percepción, queda una ingente cantidad de trabajo por realizar, que, debido a la obligatoria finitud del trabajo, dejamos para futuros investigadores.

En este sentido, es importante recalcar que estamos ante un problema subjetivo y que todas las métricas objetivas que realicemos serán aportaciones al problema pero en ningún caso soluciones generales perfectas, por lo que se pueden encontrar trabajos que, en cierto modo, puedan proponer soluciones completamente diferentes e incluso contradigan lo expuesto aquí si no se contextualiza bien el enfoque. Por suerte o por desgracia, hasta que se demuestre lo contrario, el problema va a seguir sin una solución óptima en sentido estricto. De hecho, el ajuste automático que proponemos no pretende sustituir completamente al ajuste manual sino complementarlo y servir de alternativa en algunos casos. Como se comenta muy acertadamente en [70], no se pretende describir objetivamente la percepción humana de un modo universal y completo, ni sustituir los métodos subjetivos de evaluación. Se pretende aportar una nueva forma de evaluación que complemente a la tradicional y sirva además para acelerar ciertos procesos como el ajuste de parámetros.

8.2.2 – Aportaciones Particulares

En cuanto a herramientas y aportaciones concretas y particulares al campo de la simulación, podemos citar el desarrollo de un simulador de bote de rescate rápido, del que, al menos en nuestra experiencia, no existía ninguna publicación parecida, y que podemos citar como aportación completamente original.

Siguiendo con los simuladores, también podemos citar el desarrollo del simulador de plataforma de movimiento. Si bien existen otros simuladores similares de este tipo, creemos que aportamos detalles novedosos en cuanto a la especificación genérica y a la empleabilidad del mismo como sustituto completo del *hardware* original.

En cuanto al desarrollo de plataformas de movimiento, la aportación más significativa es el análisis del dispositivo de 3 GdL. Al menos hasta donde nuestro conocimiento alcanza, no existe ningún análisis de este tipo de plataforma, y el tipo de elementos que se analizan, como las zonas seguras, la reordenación del espacio de soluciones, etc., son elementos novedosos. Simplemente la presentación de las ecuaciones de la cinemática inversa del manipulador ya es un aporte importante porque permite utilizar dicho dispositivo, que como comentamos es bastante habitual en paquetes comerciales, sin necesidad de ceñirse a las limitaciones de *software* impuestas por el fabricante, que por lo general no ofrecen un acceso abierto al dispositivo ni publican los detalles de la cinemática inversa del manipulador. La medición óptica para la caracterización dinámica de la plataforma, si bien no

es un elemento novedoso, sí puede considerarse una aplicación original.

La aplicación de heurísticas al problema del ajuste de parámetros en MCA es también un elemento novedoso. El autor sólo ha encontrado una referencia similar [13], que es coetánea al desarrollo aquí realizado y que no detalla suficientemente cómo se adapta el problema al paradigma de programación evolutiva.

Finalmente, aunque su aplicación al problema para el que fue diseñado no ha resultado ser mejor que el algoritmo genético, el diseño del algoritmo Coulómbico, que es una idea completamente original del autor, puede ser de utilidad en otros problemas de optimización, por lo que consideramos que es una modesta y pequeña aportación al campo de la algoritmia.

8.3 – Trabajo Futuro

Como acabamos de comentar en el apartado anterior, la apertura de esta línea de investigación ha generado una gran cantidad de elementos de estudio que, por falta de tiempo, no hemos podido analizar.

En primer lugar, proponemos continuar los análisis aquí realizados con más algoritmos de tipo MCA. Aunque todos se basan en el mismo principio básico, resultaría interesante validar algunas de las conclusiones aquí establecidas con otros algoritmos.

En segundo lugar, proponemos probar también diferentes modelos de percepción para comprobar si alguno de ellos se ajusta a la caracterización humana que hemos realizado.

Otro aspecto muy interesante sería analizar otros dispositivos. Aunque aquí hemos analizado uno en profundidad, y hemos realizado pruebas con dos tipos de plataforma de movimiento diferentes, sería interesante probar cómo los diferentes estudios que hemos realizado se ven alterados por los parámetros de construcción de la plataforma de movimiento o por las distintas morfologías de las mismas. Para la realización de este tipo de estudios se necesitan unos recursos económicos y humanos de los que, desgraciadamente, carecemos.

En el campo del desarrollo de métricas, proponemos investigar más a fondo la combinación de las mismas, el diseño de nuevos indicadores y la aplicabilidad de los distintos algoritmos de optimización descritos a cada uno de los simuladores que se puedan construir. En esta línea, también sería interesante estudiar más en profundidad la correlación entre el simulador de plataforma y las distintas plataformas de movimiento que se puedan construir, para perfeccionar esta herramienta y estar seguros de que las desviaciones que podamos obtener, entre los parámetros que los algoritmos de optimización proporcionan y los que se derivan de las opiniones de los pilotos, sean mínimas.

Por último, aunque no ha sido comentado en este trabajo, los algoritmos de *washout* generan algunas veces señales que, si bien son matemáticamente correctas y *óptimas* en un sentido formal y global, proporcionan sensaciones bastantes desagradables en los

pilotos. De las numerosas pruebas que hemos realizado con pilotos, muchos se han quejado, por ejemplo, de falta de sensación cuando se cambia de marcha o cuando se derrapa en un coche. Este tipo de movimientos bruscos muchas veces son filtrados en exceso (incluso eliminados) por los filtros de los algoritmos de *washout* en aras de obtener una mayor fidelidad global en la simulación. En el mismo sentido, algunos pilotos se quejan de efectos extraños en la simulación traslacional cuando los GdL traslacionales son muy cortos. Por todo ello, consideramos interesante abrir una línea de investigación que desarrolle pequeñas heurísticas para simular, aunque sea de forma exagerada, movimientos muy concretos que, según parece, proporcionan información muy útil a los pilotos, como los cambios de marcha, los derrapes, los baches, etc. Sería interesante comparar estas heurísticas con las soluciones canónicas para ver si aportan alguna mejora sobre la sensación global.

8.4 – Publicaciones

Las publicaciones originadas, hasta el momento, por el trabajo surgido de esta tesis doctoral son, por orden de publicación, las siguientes:

· **Sergio Casas Yrurzum**, Ausiàs Llorenç Palau, José Vicente Riera López, Marcos Fernández Marín - **ESTRIBOR: A Virtual Speed-Boat Rescue Training Simulator**. *Proceedings of the VIII International Conference on Occupational Risk Prevention* (ORP 2010). Valencia, Spain, May 5-8, 2010.

· **Sergio Casas**, Silvia Rueda, José V. Riera and Marcos Fernández – **On the Real-Time Physics Simulation of a Speed-Boat Motion**. *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications (GRAPP 2012).* Rome, Italy, February 24-26, 2012.

· **Sergio Casas**, Ricardo Olanda, Marcos Fernández and José V. Riera - **A Faster Than Real-Time Simulator of Motion Platforms**. *Proceedings of the 12th International Conference on Computational and Mathematical Methods in Science and Engineering, (CMMSE 2012).* Murcia, Spain, July 2-5, 2012.

· **Sergio Casas**, Inmaculada Coma, José V. Riera and Marcos Fernández - **On the Characterization of a Speed-boat Motion for Real-time Motion Cueing**. *Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications (GRAPP 2013).* Barcelona, Spain, February 21-24, 2013.

· **Sergio Casas**, Inmaculada Coma, José María Alcaraz, Ricardo Olanda and Marcos Fernández - **Towards an Extensible Simulator of Motion Platforms**. *Simulation Modelling Practice and Theory, Elsevier* (Q2), Volume 45, June 2014, pp. 50-61.

· **Sergio Casas**, Inmaculada Coma, José V. Riera and Marcos Fernández - **Motion Cuing Algorithms: Characterization of Users' Perception**. *Human Factors, The*

Journal of the Human Factors and Ergonomics Society (Q2) – Aceptada, DOI: 10.1177/0018720814538281.

· **Sergio Casas**, Inmaculada Coma, José María Alcaraz and Marcos Fernández - **Kinematic Analysis of the DOF-Space of a 3-DOF Parallel Manipulator for Motion Cueing**. En proceso de revisión.

· **Sergio Casas**, Inmaculada Coma, and Marcos Fernández – **On the Optimization of Motion Cueing Algorithms Using Heuristic Methods**. En proceso de redacción.

Apéndice A

Descripción de Sensores en la Caracterización del Bote de Rescate Rápido

Internamente, cada Wiimote está equipado con un acelerómetro de 3 ejes *ADXL 330* fabricado por Analog Devices, Inc. Este sensor mide la aceleración lineal con un rango de ±3 g.

El accesorio Wii Motion Plus está equipado con 2 giróscopos. El primero de ellos es un *IDG-600* de InvenSense. Es un giróscopo de 2 ejes que aporta datos sobre los ejes X e Y. Este dispositivo tiene una frecuencia de respuesta de 140 Hz y un rango de al menos ±500 °/s (modificable hasta los ±2000 °/s). El segundo de los giróscopos es un *X3500W* de Epson Toyocom. Éste es un giróscopo de eje único y aporta datos sobre el rumbo (*yaw*). Aunque las especificaciones de este dispositivo no han sido publicadas, se cree que son muy similares a las del chip *XV-3500CB* del mismo fabricante. Este último dispositivo tiene una frecuencia de respuesta de 200 Hz y un rango de ±100 °/s.

Como algunas de las especificaciones del Wiimote y del Wii Motion Plus fueron extraídas de fuentes de dudosa fiabilidad, ya que la información oficial es escasa, realizamos varias pruebas para asegurar la validez de dicha información. A pesar de que se supone que los acelerómetros y los giróscopos tienen una elevada frecuencia de actualización, las pruebas que realizamos sugirieron

que la frecuencia actual efectiva ronda los 50 Hz, probablemente debido a que el Wiimote limita los dispositivos, pero, en cualquier caso, parece suficiente para nuestro propósito. Los rangos y la precisión fueron consistentes los datos publicados.

El registrador GPS elegido fue el *Holux M-241*. Éste es un dispositivo inalámbrico capaz de proporcionar una nueva posición cada segundo (1 Hz) con un retraso máximo de 0.1 s, un error horizontal inferior a 2.2 m y un error vertical inferior a 5 m durante el 95% del tiempo. Este aparato es capaz de registrar la velocidad con un error inferior a 0.1 m/s y puede conservar en torno a 130000 lecturas en su memoria interna. El error vertical es bastante grande, pero como utilizamos el GPS para monitorizar el movimiento y la velocidad a lo largo del plano XY, esto no debería representar un problema. Además, una frecuencia de actualización de 1 Hz es bastante pobre debido que la tecnología GPS no militar ha sido diseñada pensando en guiar coches por carretera y no para realizar mediciones de alta precisión. No obstante, tal y como ya explicamos, podemos combinar la información de los acelerómetros y del GPS para obtener una mejor estimación de la velocidad.

La brújula digital elegida fue la *OS3000*. Es una brújula USB que permite registrar la orientación con una tasa de actualización máxima de 20 Hz, una precisión de 1º y una resolución de 0.1º.

En cuanto al anemómetro, el dispositivo seleccionado fue el *Kestrel 4000*. La frecuencia de actualización de este sensor es de 1 Hz y puede almacenar hasta 4000 lecturas de velocidad de viento, en un rango entre 0.4 y 60 m/s con un error de ±3%.

Apéndice B

Ajuste y Validación del Simulador de Bote de Rescate Rápido

Con objeto de evaluar el modelo físico propuesto, dicho modelo se implementó y se probó usando un simulador completo desarrollado por el grupo ARTEC del Instituto de Robótica y de las Tecnologías de la Información y de la Comunicación (IRTIC) de la Universitat de València. La Figura B.1 muestra una visión panorámica de la disposición del *hardware* del simulador. Se observan 3 elementos principales: una pantalla cilíndrica envolvente con un sistema de proyección, una plataforma de movimiento de 6 GdL, con un bote de rescate real sensorizado sobre ella, y un puesto de instrucción para poder también manejar el simulador desde fuera.

El sistema de proyección consiste en una pantalla cilíndrica de 3 metros de altura con un diámetro de 6 metros que resulta en un campo visual vertical de 53º y horizontal de 270º, con una resolución total de 3840x1024 píxeles. El sonido también está integrado en el simulador en forma de sistema envolvente de sonido 5.1. La plataforma de movimiento es una plataforma electromecánica Stewart de 6 GdL [37]. Puede soportar hasta 500 kg y los límites de excursión y las aceleraciones se muestran en la Tabla B.1.

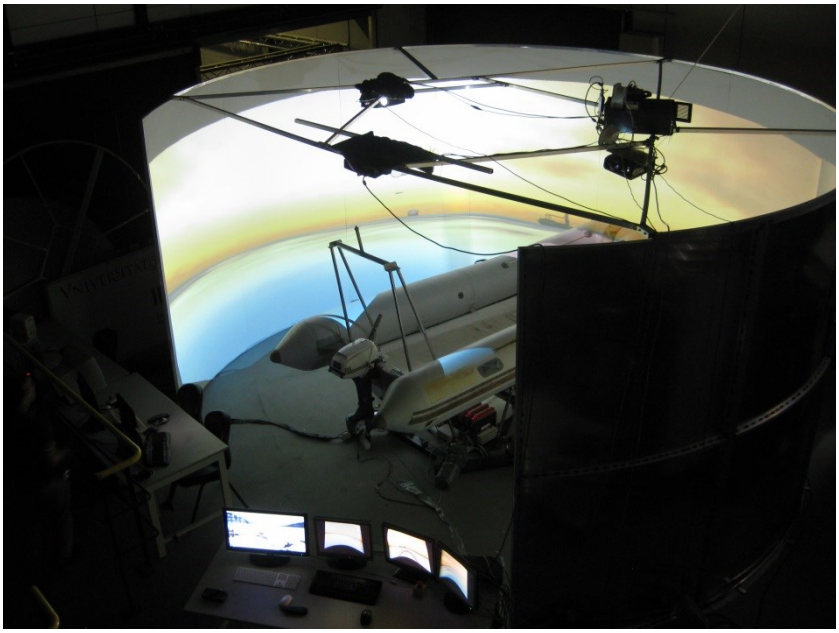

Figura B.1 - *Disposición hardware del simulador*

El módulo *software* de la plataforma de movimiento resuelve la cinemática inversa de la plataforma de movimiento utilizando el algoritmo de *washout* clásico de Reid y Nahon [86] con objeto de generar las claves gravito-inerciales correspondientes. Las entradas del algoritmo de *washout* (aceleración lineal del bote y velocidad angular) provienen del modelo físico.

En cuanto al interfaz de manejo, para crear una simulación más inmersiva, se sensorizó un bote de rescate rápido real y se colocó éste sobre la plataforma de movimiento. De esta forma, el usuario puede navegar como lo haría con un bote real, accionando el timón y la palanca del acelerador.

El sistema visual, la plataforma de movimiento, el puesto de instrucción y la interfaz sensorizada están controlados por un único computador personal, un Intel i7-920 Quad Core a 2700 MHz con una placa base Asus P6T Deluxe V2, 8 Gb de memoria DDR3 y 2 tarjetas gráficas Nvidia GeForce GTX 480 con soporte específico para PhysX. El sistema operativo es un Windows 7 Enterprise.

El modelo físico propuesto presenta diversos parámetros que necesitan ser configurados antes de poder realizar una validación correcta. Dichos parámetros pueden ser clasificados en 3 grupos: el número de cubos para la representación del bote de rescate, los parámetros de las ecuaciones físicas (masas, densidades, inercias, coeficientes de fricción, etc.) y todos aquellos parámetros relacionados con el algoritmo de *washout*.

Las secciones siguientes muestran cómo hemos procedido a ajustar todos estos parámetros experimentalmente, haciendo especial hincapié en los dos primeros grupos de parámetros, ya que al tercero se le dedica buena parte de algunos de los capítulos de este trabajo.

GdL	*Máx. excursión*	*Máx. aceleración*
Pitch, *Roll*	25º	±500 º/s2
Yaw	30º	±500 º/s2
Heave, *Surge*, *Sway*	±0.085 m	±0.5 g

Tabla B.1 - *Límites de excursión de la plataforma de movimiento del simulador*

B.1 – Ajuste del Número de Cubos

El principal objetivo del modelo físico es reproducir el comportamiento físico del bote real de la manera más precisa posible. Parece bastante evidente pues, que, dado que el modelo físico se sustenta en una subdivisión cúbica, el número de cubos utilizados en el modelo es el primer parámetro que debe ser delimitado y estudiado.

Como la subdivisión en cubos está diseñada con objeto de adaptar mejor los cálculos a la forma del bote, a mayor número de cubos, mayor precisión en la simulación, a menos que el uso de la CPU se acerque demasiado al 100%. En ese punto, el cálculo es más costoso (en tiempo) que el tiempo que está siendo simulado, los límites de tiempo real se violan y la experiencia de simulación se degrada.

Sin embargo, si el número de cubos es demasiado pequeño, la precisión de la simulación también se reduce (ya que el cálculo realizado es mucho más burdo al evaluarse menos puntos del vehículo), por lo que necesitamos llegar a un compromiso entre ambos extremos. Es por ello que intentaremos encontrar el número máximo de cubos que la CPU soportaría sin violar los límites de tiempo real (es decir, que el tiempo que cuesta simular sea menor que el tiempo simulado), y ese número maximizaría la precisión del modelo.

Teniendo en cuenta que el modelo físico no es la única parte del simulador, se debe monitorizar el uso global de CPU (y no sólo el rendimiento del modelo físico). La frecuencia de

actualización que se toma como estándar para fijar el umbral de inmersión es de 60 Hz [36]. Por eso, lo que buscamos es estimar el número máximo de cubos que podríamos usar para mantener al menos una frecuencia de actualización de 60 Hz sobre todo el sistema (tanto visual como físico y de la plataforma de movimiento). Realizando aproximaciones sucesivas, ese número resultó ser 549 cubos.

B.2 – Ajuste con Expertos

A continuación, usamos el número de cubos previamente calculado para configurar el modelo de bote y encontrar valores adecuados tanto para el modelo físico como para los parámetros del algoritmo de *washout*. Estos valores se configuraron mediante el consenso de 3 expertos en este tipo de vehículos. El vehículo configurado trató de reproducir el comportamiento del vehículo sobre el cuál hicimos las pruebas reales (un Duarry Brio 620 propulsado por un motor Suzuki DF 140 de 4 tiempos y 140 CV de potencia).

Los valores iniciales para los parámetros físicos de la simulación fueron establecidos en base a los valores teóricos del vehículo real y a partir de entonces fueron sucesivamente modificados en una secuencia de tipo *round-robin* (un experto, una modificación cada vez), hasta que los expertos estimaron, por consenso, que el comportamiento era suficientemente plausible. Los parámetros del modelo físico y sus valores resultantes se muestran en la Tabla B.2.

Parámetro(s) [unidades]	*Valor(es)*
Densidades del cubo (inflable, rígido) [kg/m^3]	(150, 500)
Coeficientes de fricción aerodinámica (x,y,z) [sin dimensiones]	(8, 1, 6)
Coeficientes de fricción hidrodinámica (x,y,z) [sin dimensiones]	(0.5, 7, 1)
Función de torque del motor con acelerador al 100% [N·m]	500 rpm: 50, 2000 rpm: 300, 5000 rpm: 500, 7000 rpm: 200
Constante de fricción del motor [N·m]	20
Constante de fricción de velocidad del motor [kg·m^2/s]	2
Constante de fricción de aceleración del motor [kg·m^2]	0.1
Inercia del motor [kg·m^2]	18.5
Función de dirección del timón [sin dimensiones]	-60°: (-1, -1), -30°: (-0.5, -1), 0°: (0, -1), 30°: (0.5, -1) 60°: (1, -1)
Función de eficiencia de la hélice [sin dimensiones]	500 rpm: 0.8, 2000 rpm: 0.9, 5000 rpm: 0.5, 7000 rpm: 0.2
Coeficiente diferencial motor-hélice [sin dimensiones]	1
Coeficiente de avance de la hélice [kg/s]	1.9

Tabla B.2 - *Parámetros de configuración del modelo físico*

En los cubos se ajustó la densidad en lugar de la masa para facilitar el cambio de las dimensiones de los cubos. Para simplificar, sólo se consideraron dos tipos de cubos: aquellos que pertenecen a la parte rígida del bote, y aquellos que corresponden a la parte inflable. Los parámetros de rozamiento (*drag*) se ajustan separadamente en 3 componentes (una para cada dirección del

espacio). Finalmente, las funciones fueron parametrizadas como funciones lineales a trozos, de las cuales sólo se muestran los valores de dichos intervalos.

Los parámetros del algoritmo de *washout* se configuraron siguiendo una aproximación iterativa subjetiva con el sistema *pilot-in-the-loop* y no se muestran aquí por brevedad.

B.3 – Validación

Para comprobar la validez del modelo propuesto, debemos comparar los resultados obtenidos con resultados reales. Podemos hacer esto de varias formas. La primera opción consistiría en comparar magnitudes reales tomadas sobre el vehículo y magnitudes extraídas del simulador. La segunda, de forma indirecta, haciendo pruebas con pilotos que estén acostumbrados a manejar este tipo de vehículos y preguntarles su opinión sobre el modelo físico.

Es por ello que proponemos una doble evaluación del modelo, por lo que presentamos dos tipos de pruebas. El primer test consiste en medir datos reales del bote de rescate y compararlos con el comportamiento simulado. El segundo test consistiría en medir cómo de inmersos se sienten los usuarios de un simulador de este tipo con un modelo físico como éste. Este concepto, conocido como presencia, no se puede medir analíticamente, por lo que normalmente se evalúa mediante cuestionarios [186].

Antes de realizar la validación, el simulador debe ser instalado y ajustado para que su comportamiento se parezca al de un bote real. Para ello es necesario buscar valores adecuados para cada uno de los parámetros del modelo físico que hemos comentado previamente. Por desgracia, la única opción realista para realizar esta tarea, es contratar o disponer de uno o varios expertos en el vehículo y ajustar los valores de los parámetros del modelo hasta que el comportamiento del mismo se parezca al que él conoce. El Apéndice B.2 explica este proceso, así como los componentes del simulador necesarios para poder probar el módulo físico (ya que éste no se puede validar aisladamente de forma sencilla).

B.3.1 – Validación Cuantitativa

La validación cuantitativa consiste en comparar datos reales, de los que disponemos como vimos en el capítulo 2 correspondiente a la caracterización gravito-inercial, con medidas virtuales del bote simulado. Las maniobras que probamos fueron: aceleración en línea recta de 0 a 25 nudos, deceleración brusca (frenazo) de 25 nudos a 0, velocidad de crucero a 20 nudos y giro completo de 360º con máximo giro de timón (a 15 nudos). Podemos observar en la Figura B.2 los 3 primeros casos citados. Las trayectorias reales se muestran en color rojo y las trayectorias simuladas con color azul. Cada marca representa una posición (X, Y) del bote de rescate con una separación de 0.25 segundos. De forma similar, la Figura B.3 muestra el giro del vehículo a una velocidad de 15 nudos.

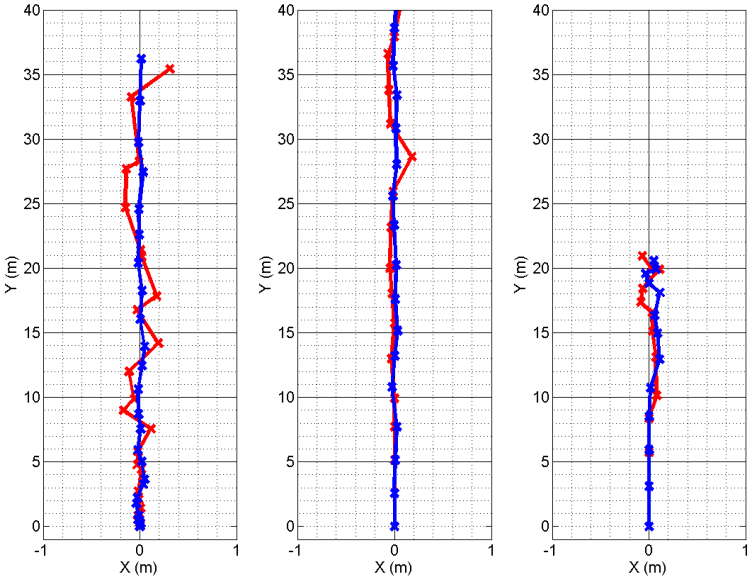

Figura B.2 - *Aceleración 0-25 nudos (izquierda), navegación de crucero a 20 nudos (centro) y deceleración brusca 25-0 nudos (derecha)*

Podemos apreciar que las trayectorias son similares, aunque las trayectorias reales parecen tener bastante más ruido. Esto es causado probablemente por 3 razones: los sensores reales presentan ruido y las mediciones no son exactas (no hicimos la caracterización con el objetivo de hacer *tracking* por lo que los datos se deben interpretar con cuidado), el timón virtual es mucho más fácil de fijar que un timón real, y las interacciones del mundo real con el bote son más complejas de lo que nuestro modelo simulado refleja, ya que nuestro modelo es una simplificación. En cualquier caso, se puede observar que el modelo se ajusta al comportamiento real, al menos en cuanto a trayectorias.

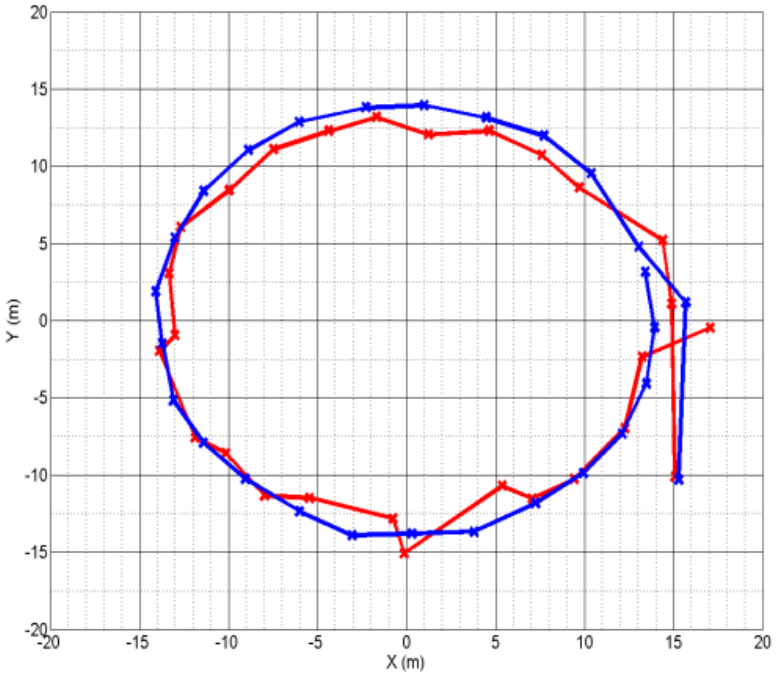

Figura B.3 - *Giro de 360º a 15 nudos*

Podríamos hacer un análisis similar con las inclinaciones, pero como éstas dependen principalmente del estado del mar, y es imposible reproducir el mismo estado de mar que existía cuando realizamos las pruebas reales, creemos que esto no demostraría nada. De hecho, aunque la comparación sea cuantitativa, dada la variabilidad de las condiciones, los datos se deben interpretar en su justo contexto.

B.3.2 – Validación Cualitativa

Por esta razón, es conveniente hacer, además, una validación cualitativa. Para ello, escogimos a 45 expertos en el manejo de este tipo de embarcaciones y les pedimos que probaran el simulador (previamente configurado por un grupo de 3 expertos) durante tandas de 15 minutos. A todos ellos se les pidió que realizaran las mismas maniobras sobre un mismo circuito de test. El circuito (virtual) de test consistía en un pasillo de agua, delimitado por dos líneas de boyas paralelas. Las boyas se emplearon para servir de referencia visual. El ancho del pasillo era de 20 metros, y cada boya estaba separada de la siguiente por 30 metros. Las maniobras de prueba fueron: navegación libre, giros de 180 º al final del pasillo de boyas, navegación en *zig-zag* por las líneas de boyas, aceleración lineal dentro del pasillo, navegación a velocidad de crucero, y deceleración brusca dentro del propio pasillo.

En la Figura B.4 podemos ver una captura visual de las pruebas realizadas.

Una vez terminadas las pruebas, se les pasó un cuestionario sobre sus impresiones. Las preguntas se debían contestar con valores entre 0 y 10, siendo 10 "estoy completamente de acuerdo" y 0 "estoy en total desacuerdo". Consideramos que valores superiores a 7 significaban "es suficientemente bueno" y valores inferiores a 7 representaban "necesita ser mejorado" (excepto para las preguntas sobre el mareo de simulador, que eran preguntas inversas). Las cuestiones se agruparon en 3 bloques. El primer bloque trataba de cuestiones generales sobre la habilidad de cada módulo del simulador (visual, inercial, físico, auditivo, etc.) para

generar presencia e inmersión. El segundo bloque trataba específicamente sobre el modelo físico. Finalmente, el tercer bloque trataba sobre la impresión general del simulador. Las respuestas medias (y las preguntas) de los 45 cuestionarios se pueden ver en la Tabla B.3.

Figura B.4 – *Captura visual del simulador de bote de rescate*

Los resultados muestran que algunos de los módulos necesitaban ser mejorados (como el sonido), pero la percepción global y el módulo físico recibieron una buena aceptación. Aunque algunos de los elementos (como el timón) son considerados mejorables (en opinión de los expertos), creemos que ello es consecuencia del *hardware* del simulador más que del modelo físico

en sí. También, la ausencia de resistencia al giro del timón por la ausencia de agua real sobre el dispositivo de simulación es algo que puede influir bastante en la sensación.

Pregunta	*Valoración*
¿Es el entorno virtual realista?	7.4
¿Es el movimiento de la plataforma similar al real?	9.4
¿El sonido produce un efecto realista?	5.3
¿El comportamiento del mar parece natural?	6.9
¿Las señales visuales y de movimiento encajan?	9.0
¿Es realista el accionamiento del acelerador?	7.7
¿Es realista el accionamiento del timón?	7.1
¿Es el movimiento de *yaw* realista?	8.4
¿Es el movimiento de *pitch* realista?	8.9
¿Es el movimiento de *roll* realista?	8.3
¿Es realista la interacción entre el casco y el mar?	8.1
¿Se ha sentido inmerso dentro de la simulación?	8.9
¿Se ha mareado durante la simulación?	2.9
Valoración global del comportamiento del bote	8.8
Valoración global del comportamiento del sistema	8.5

Tabla B.3 - *Preguntas y respuestas medias del cuestionario*

Apéndice C

Ajuste y Validación del Simulador de Plataforma de Movimiento

Con objeto de usar el simulador de plataforma como sustituto de una plataforma de movimiento real, es necesario evaluar su viabilidad para tal tarea. La mejor manera de hacerlo es comparar la respuesta de la plataforma de movimiento simulada frente a una real. Como disponíamos de dos plataformas de movimiento reales de 3 y 6 GdL, decidimos realizar la evaluación del simulador probando su funcionamiento frente a estos dos ejemplos.

C.1 – Ajuste de Parámetros

Para realizar esta evaluación, lo primero que necesitamos es configurar el simulador para estos dos casos de estudio (3 GdL y 6 GdL). La primera plataforma de movimiento es una plataforma T1R2 similar a la del capítulo 4. La segunda es una T3R3 similar a la que empleamos en el capítulo 6. Dado que ya conocemos su diseño, no haremos una descripción completa de estas dos plataformas de movimiento, ni, por otro lado, es el objetivo de este apéndice. Por consiguiente, sólo mostraremos sus características principales.

Los rangos de movimiento de dichas plataformas se muestran en la Tabla C.1.

	T1R2		*T3R3*	
	Valor mín.	*Valor máx.*	*Valor mín.*	*Valor máx.*
Rango surge	-	-	-10.58 cm	+10.58 cm
Range sway			-9.77 cm	+13.78 cm
Rango heave	-10.0 cm	10.0 cm	-7.84 cm	+9.06 cm
Rango yaw	-	-	-20.73 º	+20.73 º
Rango pitch	-30.52 º	+30.52 º	-17.45 º	+16.33 º
Rango roll	-26.21 º	+26.21 º	-17.58 º	+17.58 º

Tabla C.1 - *Características de las dos plataformas de movimiento reales*

Tal y como ya se mencionó, el primer paso para que el simulador funcione es construir un modelo CAD para los dos manipuladores. Las restricciones del modelo CAD para que funcione el simulador se explicaron en el capítulo 7. En este caso, ambos diseños cumplen con la estructura requerida. La construcción del modelo CAD no se puede automatizar y se pueden dar pocas guías al respecto. La regla más importante es ubicar cada motor, cada sólido rígido y cada unión tal y como se sitúan en el diseño real. Esto se hace mediante 3D Studio Max y su *plug-in* de PhysX [31]. El último paso es elegir el tipo de articulación para cada unión. En [174] se muestra una descripción completa de los tipos de articulaciones/uniones soportadas por Nvidia PhysX. Una vez el modelo CAD está correctamente construido, el siguiente paso es la configuración de los parámetros del simulador. Podemos dividir este proceso en dos partes: la

primera está relacionada con las magnitudes físicas de los elementos del manipulador (dimensiones, masas, etc.). La segunda es la calibración de los motores virtuales.

Las magnitudes físicas incluyen posiciones, dimensiones, masas e inercias de los elementos de la plataforma de movimiento. Las posiciones y las dimensiones se definen con el modelo CAD. Las masas se pueden configurar directamente a partir de los archivos XML del simulador. Las inercias se calculan internamente mediante la medida de los volúmenes de los objetos (asumiendo una situación de densidad constante). Todas estas magnitudes se establecieron en base a sus valores físicos correspondientes, lo cual puede observarse en la Tabla C.2, aunque los valores de las masas son aproximados. Las posiciones y las dimensiones están definidas implícitamente dentro del modelo CAD, y no se muestran en dicha tabla.

La carga es un parámetro variable que puede establecerse arbitrariamente. En estas pruebas, asumimos una carga de 100.0 kg con una forma cúbica de 1 m^3. La segunda parte de la configuración incluye el calibrado de los motores virtuales. Esto afecta a 4 parámetros: el torque máximo de los motores y los parámetros del controlador PID: Kp, Ki, y Kd. El torque máximo es una característica del motor, cuyo valor se obtiene del valor real. Este valor incluye la reducción aportada por la reductora del motor y establece un límite máximo para el torque que los motores son capaces de proporcionar a las bielas. El calibrado del controlador PDI implica encontrar valores apropiados para Kp, Ki y Kd. Hay muchos métodos para hacer esto, pero optamos por un calibrado

empírico [176]. Por brevedad, solo mostramos los valores finales (ver Tabla C.2).

Parámetro	*T1R2*	*T3R3*
Masa del motor	100.0 kg	100.0 kg
Masa de la biela	5.0 kg	5.0 kg
Masa del pistón	1.0 kg	1.5 kg
Masa de la base móvil	20.0 kg	30.0 kg
Masa del eje estriado exterior	20.0 kg	-
Masa del eje estriado interior	10.0 kg	-
Carga	100.0 kg	100.0 kg
Kp	11.2	10.1
Ki	1.35	1.53
Kd	0.19	0.27
Torque máximo	150 N·m	200 N·m

Tabla C.2 - *Parámetros físicos de ambas plataformas virtuales de movimiento*

C.2 – Validación

Una vez el simulador ha sido configurado, podemos iniciar el proceso de validación. Este proceso es una comparación entre el simulador de la plataforma de movimiento y su homóloga real. La forma más común de hacerlo es comparar cada grado de libertad por separado probando algunas señales analíticas conocidas.

Se pueden utilizar muchas señales de prueba diferentes, pero las más comunes son las señales seno y escalón. Una señal

sinusoidal proporciona información sobre el movimiento suave, mientras que la señal escalón proporciona información sobre el movimiento súbito. El problema de estas señales es que proporcionan una prueba en el dominio del tiempo, y no se observa el comportamiento en el dominio de la frecuencia (a menos que realicemos muchas pruebas diferentes con diferentes frecuencias). Para resolver esto, decidimos usar una señal *sine-chirp* (un seno con frecuencia creciente) y una señal *square-chirp* (una onda cuadrada de frecuencia creciente), que ya vimos en el capítulo 4. De esta manera, podemos comparar las plataformas de movimiento tanto en el dominio del tiempo como en el de la frecuencia, a la vez. La última decisión es la amplitud de las señales. Ya que la respuesta de la plataforma de movimiento es diferente entre desplazamientos largos y pequeños (porque los desplazamientos largos requieren mayores velocidades), es interesante evaluar el resultado de las plataformas de movimiento para diferentes amplitudes. No obstante, mediante el uso de las funciones *chirp*, estamos realmente solicitando a las plataformas de movimiento diferentes velocidades y podemos ver el efecto de un cambio de velocidad sobre el comportamiento de la plataforma. Por consiguiente, usaremos las funciones *chirp* con aproximadamente amplitudes máximas para cada GdL.

Para estas pruebas, la tasa de cambio de las funciones *chirp* se estableció en 1.2 y la frecuencia inicial en 0.01 Hz.

Todas estas pruebas fueron realizadas usando un *software* de test externo que alimenta de entradas a la plataforma de movimiento y que, a continuación, lee sus salidas. Este *software* de prueba necesita calcular la cinemática inversa de la plataforma de

movimiento, porque las señales de prueba son GdL y las entradas de la plataforma de movimiento son ángulos de motor. Las ecuaciones de la cinemática del manipulador T1R2 fueron descritas en el capítulo 3. Las del manipulador T3R3 se obtienen de forma similar, pero por brevedad no incluimos dichas ecuaciones en este texto.

Las Figuras C.1, C.2, C.3, C.4 y C.5 muestran los resultados de trazar la respuesta de las plataformas de movimiento reales frente a las simuladas, dadas las mismas entradas. Para abreviar, sólo mostramos algunas de ellas, ya que con 6 GdL, 2 plataformas y 2 señales de prueba, tendríamos 24 gráficas diferentes, todas ellas bastante similares, por lo que hemos seleccionado las que hemos considerado más interesantes, mostrando en algunas de ellas sólo una pequeña porción de tiempo para que se aprecien mejor los detalles.

Como podemos observar, en ambos casos las salidas reales y las simuladas son muy similares. Además, su frecuencia de respuesta es similar. Ambos tienden a presentar una atenuación progresiva a medida que la frecuencia se incrementa. Esto demuestra el típico comportamiento de filtro pasa-baja presente en muchos sistemas mecánicos. El efecto es ligeramente diferente para cada GdL, pero en todos los casos se obtiene una elevada atenuación al sobrepasar los 2 Hz. En ambos casos de estudio, y en todos los GdL la salida simulada no es, de media, superior ni inferior en más de un 3% a la salida real, lo que significa que ambas plataformas de movimiento virtuales son muy similares a las reales, y pueden ser utilizadas como sustitutas de las mismas.

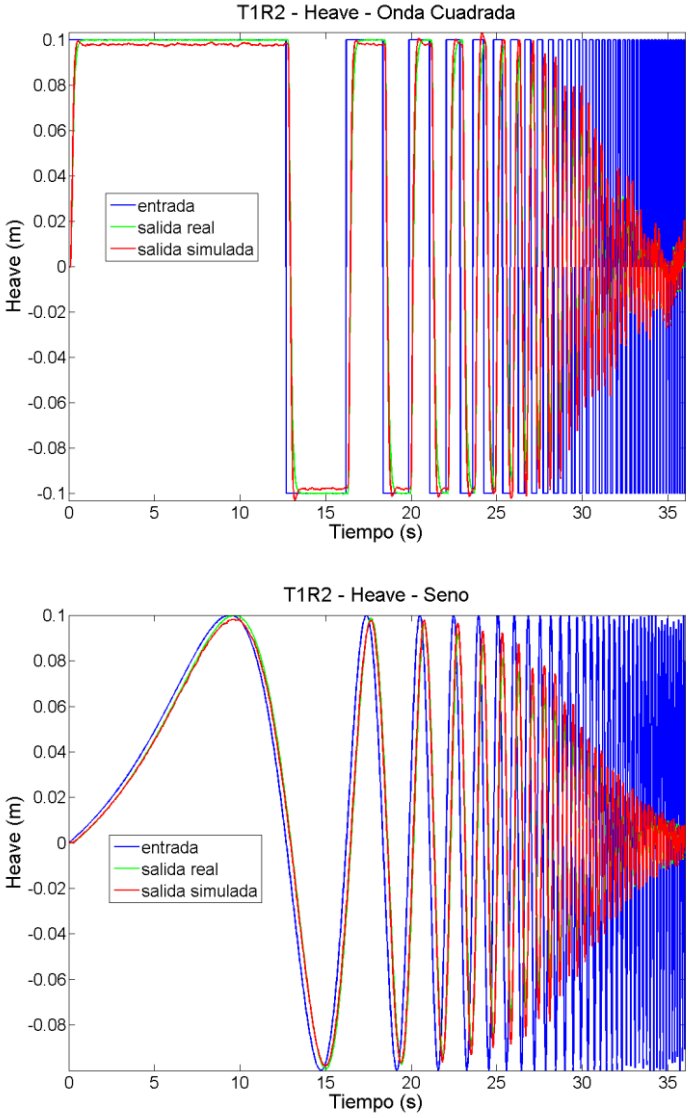

Figura C.1 - *Respuesta comparada de la plataforma T1R2 para el movimiento de heave*

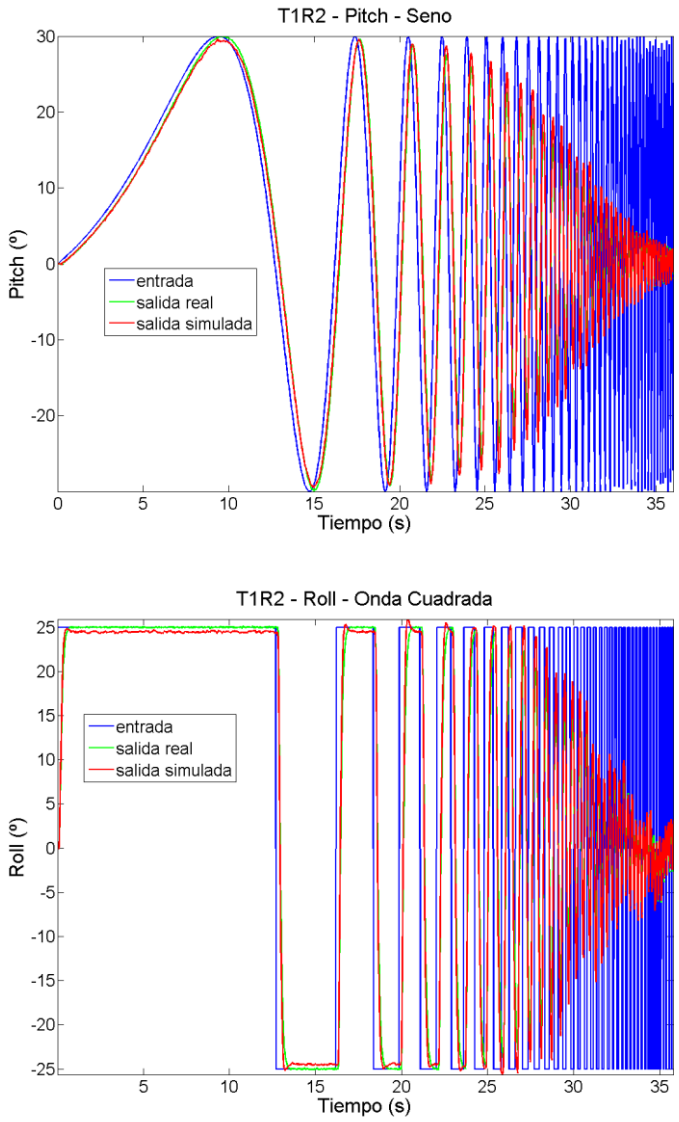

Figura C.2 - *Respuesta comparada de la plataforma T1R2 para los movimientos de pitch y roll*

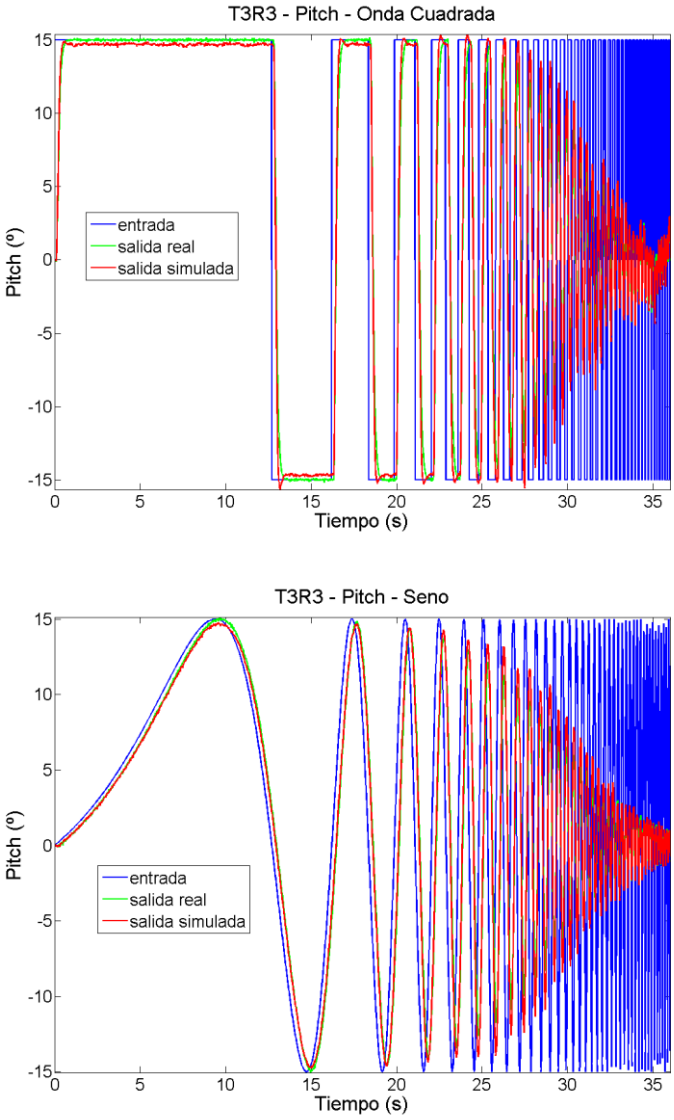

Figura C.3 - *Respuesta comparada de la plataforma T3R3 para el movimiento de pitch*

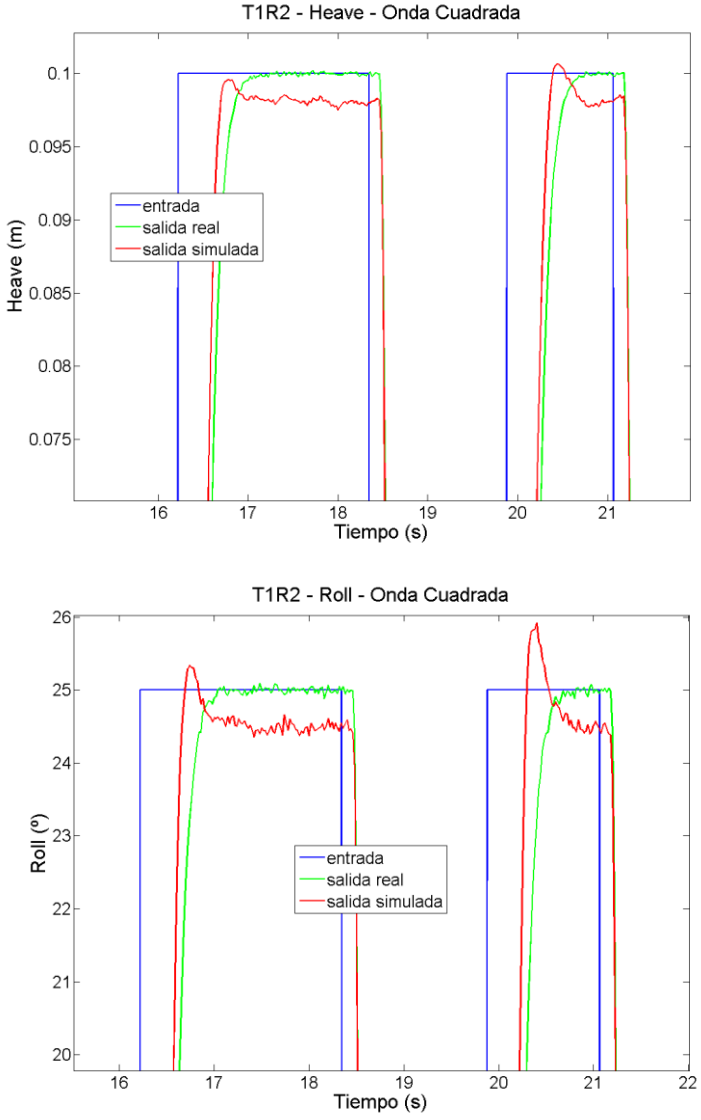

Figura C.4 - *Detalle del comportamiento de la plataforma T1R2 para los movimientos de heave y roll*

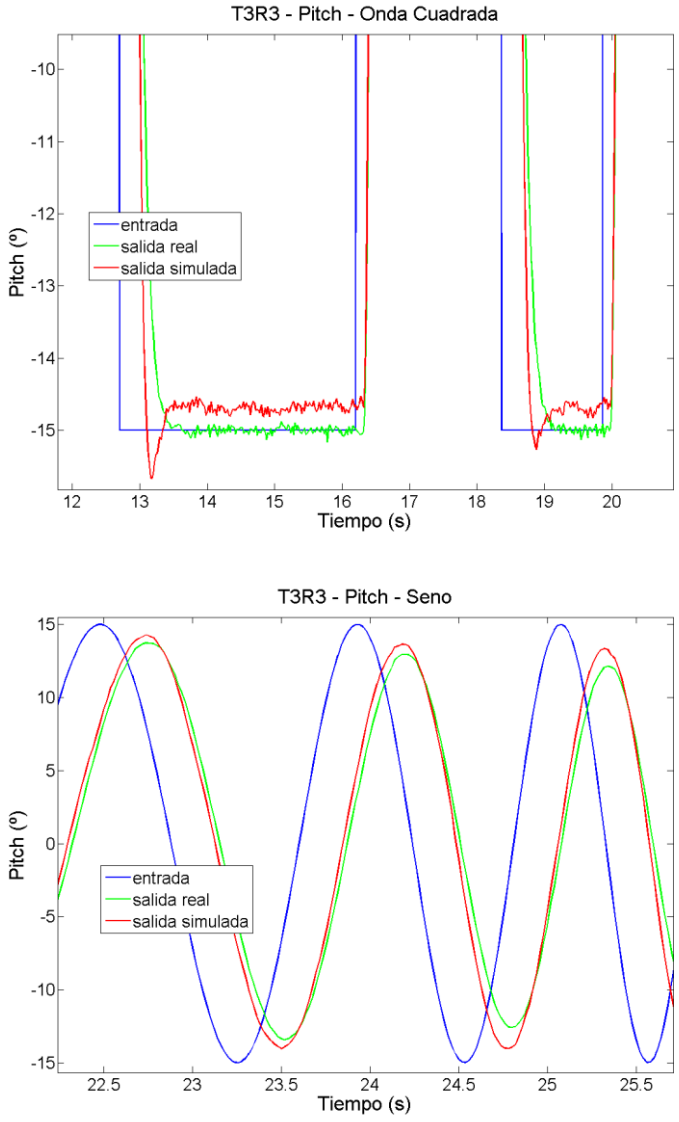

Figura C.5 - *Detalle del comportamiento de la plataforma T3R3 para el movimiento de pitch*

Como, además, se pretende que el simulador de plataformas de movimiento funcione en tiempo real, también es necesario demostrar que esto se puede conseguir. No obstante, esta aseveración depende ampliamente del sistema *hardware* desde el que se haga funcionar el simulador. Por este motivo, la única cosa que podemos demostrar es que las limitaciones de tiempo real se cumplen con un *hardware* específico. Por tanto, probamos el simulador de la plataforma de movimiento (con la configuración de parámetros previamente explicada) con el manipulador T1R2 en un Intel Core 2 Quad a 2.66 Ghz con 4 Gb de RAM, y con sistema operativo Windows 7. La frecuencia del hilo de dibujado fue fijada en 25 Hz, la frecuencia del hilo de comunicación a 50 Hz, y la del hilo físico a 500 Hz, lo cual es una frecuencia común para las simulaciones basadas en la dinámica Newtoniana-Lagrangiana. El simulador se puso en funcionamiento durante 600 segundos y se hicieron medidas temporales medias. Las medidas de tiempo mostraron que 1/500 segundos (2 ms) pueden simularse en aproximadamente 0.1 ms, lo que significa que las restricciones de tiempo real se cumplen con facilidad (en cada paso de simulación de 2 ms nos sobran 1.9 ms), y, si queremos que la simulación discurra a mayor velocidad que el tiempo real, el factor de aceleración podría llegar hasta aproximadamente 20x. Este resultado significa que, con este *hardware*, podemos probar la plataforma de movimiento 20 veces más rápido que una plataforma de movimiento real.

La misma prueba fue realizada para el manipulador T3R3, y el factor de aceleración fue de aproximadamente 14x, principalmente debido a que, cuantas más uniones, más tiempo se tarda en simularlas.

Hemos de tener en cuenta que el *hardware* empleado en las pruebas no es ni mucho menos *hardware* dedicado ni de alta gama, por lo que se pueden obtener factores de aceleración ampliamente superiores a los obtenidos, simplemente empleando computadores más actuales.

Bibliografía

[1] R. E. Shannon, Systems Simulation: The Art and Science, Upper Saddle River, NJ: Prentice Hall - PTR, 1975.

[2] J. A. Vince, Introduction to Virtual Reality, Springer, 2004.

[3] D. Winter, The First of the Few, Athens, GA: University of Georgia Press, 1982.

[4] D. Allerton, Principles of Flight Simulation, 1st ed., Chichester, West Sussex: Wiley, 2009.

[5] R. L. Page, "Brief History of Flight Simulation," in *SimTechT 2000 Proceedings*, Sydney, NSW, Australia, 2000.

[6] R. Reisman, "A Brief Introduction to the Art of Flight Simulation," *Ars Electronica,* 1990.

[7] R. Rojas, "How to Make Zuse's Z3 a Universal Computer," *Annals of the History of Computing, IEEE,* vol. 20, no. 3, pp. 51-54, Jul-Sep 1998.

[8] R. W. Allen, T. J. Rosenthal and M. L. Cook, Handbook of Drving Simulation for Engineering, Medicine and Psychology - Chapter 2: A Short History of Driving Simulation, 1st ed., Boca Raton, FL: CRC Press, 2011.

[9] rFactor, "rFactor - The Future of Race Simulation,"

[Online]. Available: www.rfactor.net. [Accessed 25 06 2014].

[10] J. J. Slob, "State-of-the-Art Driving Simulators, a Literature Survey," Eindhoven, The Netherlands, 2008.

[11] J. S. Freeman, G. Watson, Y. E. Papelis, T. C. Lin, A. Tayyab, R. A. Romano and J. G. Kuhl, "The Iowa Driving Simulator: An implementation and Application Overview," in *SAE International Congress and Exposition*, Detroit, MI, USA, 1995.

[12] SCS Software, "Euro Truck Simulator 2," [Online]. Available: http://www.eurotrucksimulator2.com/. [Accessed 25 06 2014].

[13] E. Thöndel, "Design and Optimisation of a Motion Cueing Algorithm for a Truck Simulator," in *European Simulation and Modelling Conference 2012 EUROSIS*, Ghent, Belgium, 2012.

[14] V. Cossalter, R. Lot, R. Massaro and R. Sartori, "Development and Validation of an Advanced Motorcycle Riding Simulator," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering,* pp. 705-720, 2011.

[15] F. Barbagli, D. Ferrazzin, C. A. Avizzano and M. Bergamasco, "Washout Filter Design for a Motorcycle Simulator," in *Proceedings of the IEEE Conference on Virtual Reality (VR '01)*, Nashville, TN, USA, 2001.

[16] L. Nehaoua, S. Hima, H. Arioui, N. Séguy and S. Éspié, "Design and Modeling of a New Motorcycle Riding Simulator,"

in *IEEE American Control Conference (ACC'07)*, New York, NY, USA, 2007.

[17] C.-T. Chou and L.-C. Fu, "Ships on Real-time Rendering Dynamic Ocean Applied in 6-DOF Platform Motion Simulator," in *Proceedings of the CACS International Conference 2007*, Taichun, Taiwan, 2007.

[18] P. Filipczuk and S. Nikiel, "Real-Time Simulation of a Sailboat," in *IEEE Conference on Human System Interactions, 2008*, Krakow, Poland, 2008.

[19] W. Raman-Nair and M. White, "A Model for Deployment of a Freefall Lifeboat," *Multibody System Dynamics,* vol. 29, no. 3, pp. 327-342, March 2012.

[20] S.-K. Ueng, D. Lin and C.-H. Liu, "A Ship Motion Simulation System," *Virtual Reality,* vol. 12, no. 1, pp. 65-76, March 2008.

[21] S. Casas, S. Rueda, J. V. Riera and M. Fernández, "On the Real-Time Physics Simulation of a Speed-Boat Motion," in *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2012)*, Rome, Italy, 2012.

[22] S. Casas, A. Llorenç, J. V. Riera and M. Fernández, "ESTRIBOR: A Virtual Speed-Boat Rescue Training Simulator," in *Proceedings of the VIII International Conference on Occupational Risk Prevention (ORP 2010)*, Valencia, Spain, 2010.

[23] O. Schulzyk, U. Hartmann, J. Bongartz, T. Bildhauer and

R. Herpers, "A Real Bicycle Simulator in a Virtual Reality Environment: the FIVIS Project," in *4th European Conference of the International Federation for Medical and Biological Engineering (IFMBE)*, Antwerp, Belgium, 2008.

[24] R. Herpers, W. Heiden, M. Kutz, D. Scherfgen, U. Hartmann, J. Bongartz and O. Schulzyk , "FIVIS Bicycle Simulator - an Immersive Game Platform for Physical Activities," in *Proceedings of the 2008 Conference on Future Play*, Toronto, ON, Canada, 2008.

[25] R. S. Allison, L. R. Harris, A. R. Hogue, U. T. Jasiobedzka, H. L. Jenkin, M. R. Jenkin, P. Jaekl, J. R. Laurence, G. Pentile, F. Redlick, J. Zacher and D. Zikovitz, "Simulating Self-Motion II: A Virtual Reality Tricicle," *Virtual Reality,* vol. 6, no. 2, pp. 86-95, 2002.

[26] J. M. Rolfe and K. J. Staples, Flight Simulation, Cambridge: Cambridge University Press, 1986.

[27] A. Kemeny, "Simulation and Perception," in *In Proceedings of the Driving Simulation Conference*, 33-55, 1999.

[28] Open Scene Graph, "The OpenSceneGraph Project Website," [Online]. Available: http://www.openscenegraph.org/. [Accessed 25 06 2014].

[29] D. M. Bourg, Physics for Game Developers, 1st ed., Sebastopol, CA: O'Reilly & Associates, 2002.

[30] D. Conger, Physics Modeling for Game Programmers, 1st

ed., Boston, MA: Thomson Course Technology, 2004.

[31] Nvidia Corporation, «Tecnología Nvidia PhysX,» [En línea]. Available: http://www.nvidia.es/object/nvidia-physx-es.html. [Último acceso: 25 06 2014].

[32] D. Baraff and A. Witkin, "Physically Based Modeling, Principles and Practice," 1997.

[33] CKAS, "CKAS Mechatronics Pty Ltd," [Online]. Available: http://www.ckas.com.au/. [Accessed 25 06 2014].

[34] S. Casas, P. Morillo, J. Gimeno and M. Fernández, "SUED: An Extensible Framework for the Development of Low-Cost DVE Systems," in *Proceedings of the IEEE Virtual Reality 2009 SEARIS Workshop*, Lafayette, LA, USA, 2009.

[35] I. Coma, S. Casas, A. Llorenç y J. V. Riera, «Desarrollo de Simuladores para FOREM-A,» *Novática,* vol. XXXVI, nº 206, pp. 38-42, Julio-Agosto 2010.

[36] DNV - Det Norske Veritas, "Standard for Certification No. 2.14 - Maritime Simulator Systems," 2011.

[37] D. Stewart, "A Platform with 6 Degrees of Freedom," in *Proceedings of the UK Institution of Mechanical Engineers*, 1965.

[38] C. Gosselin, "Determination of the Workspace of 6-DOF Parallel Manipulators," *Journal of Mechanical Design,* vol. 112, no. 3, pp. 331-336, 1990.

[39] M. Z. A. Majid, Z. Huang and Y. L. Yao, "Workspace Analysis of a Six-Degrees of Freedom, Three-Prismatic-Prismatic-Spheric-Revolute Parallel Manipulator," *International Journal of Advanced Manufactoring Technology,* vol. 16, pp. 441-449, 2000.

[40] S. Küçük, Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization, Rijeka, Croatia: InTech, 2012.

[41] L.-W. Tsai, Robot Analysis: The Mechanics of Serial and Parallel Manipulators, Hoboken, NJ, USA: John Wiley & Sons, 1999.

[42] L. Harris, M. R. Jenkin, D. C. Zikovitz, F. Redlick, P. Jaekl, U. T. Jasiobedzka, H. L. Jenkin and R. S. Allison, "Simulating Self-Motion I: Cues for the Perception of Motion," *Virtual Reality,* vol. 6, no. 2, pp. 75-85, 2002.

[43] D. R. Berger, J. Schulte-Pelkum and H. H. Bülthoff, "Simulating Believable Forward Accelerations on a Stewart Motion Platform," Technical Report No. 159 - Max-Planck Institute for Biological Cybernetics, Tübingen, Germany, 2007.

[44] J. J. Gibson, The Perception of the Visual World, Cambridge, MA, USA: The Riverside Press, 1950.

[45] K. R. T. Aires, A. M. Santana and A. A. D. Medeiros, "Optical Flow Using Color Information: Preliminary Results," in *Proceedings of the 2008 ACM Symposium on Applied Computing*,

Fortaleza, Brazil, 2008.

[46] L. R. Harris, M. Jenkin and D. C. Zikovitz, "Visual and Non-Visual Cues in the Perception of Linear Self Motion," *Experimental Brain Research,* vol. 135, no. 1, pp. 12-21, 2000.

[47] A. Kemeny and F. Panerai, "Evaluating Perception in Driving Simulation Experiments," *Trends in Cogntive Sciences,* vol. 7, no. 1, pp. 31-37, 2003.

[48] I. P. Howard and B. J. Rogers, Binocular Vision and Stereopsis, New York, NY: Oxford University Press, 1995.

[49] A. Bhatti, Stereo Vision, InTech, 2008.

[50] J. R. A. Torreao, Advances in Stereo Vision, InTech, 2011.

[51] OpenAL, "OpenAL," [Online]. Available: www.openal.org. [Accessed 25 06 2014].

[52] E. M. Kolasinski, "Simulator Sickness in Virtual Environments," Alexandria, VA, USA, 1995.

[53] M. Lévêque, L. Seidermann, E. Ulmer y A. Chays, «Fisiología Vestibular: Bases anatómicas, Celulares, Inmunohistoquímicas y Electrofisiológicas,» *EMC - Otorrinolaringología,* vol. 38, nº 4, pp. 1-15, 2009.

[54] T. Mergner and T. Rosemeierb, "Interaction of Vestibular, Somatosensory and Visual Signals for Postural Control and Motion Perception under Terrestrial and Microgravity

Conditions - a Conceptual Model," *Brain Research Reviews,* vol. 28, no. 1-2, pp. 118-135, 1998.

[55] D. E. Angelaki, A. G. Shaikh, A. M. Green and J. D. Dickman, "Neurons Compute Internal Models of the Physical Laws of Motion," *Nature,* vol. 430, no. 6999, pp. 560-564, 2004.

[56] A. Einstein, «Über das Relativitätsprinzip und die aus demselben gezogenen Folgerungen,» *Jahrbuch der Radioaktivität und Elektronik,* vol. 4, p. 411–462, 1908.

[57] L. D. Reid and M. A. Nahon, "Flight Simulation Motion-Base Drive Algorithms: Part 1 - Developing and Testing the Equations," Toronto, ON, Canada, 1985.

[58] P. R. McNeilage, M. S. Banks, D. R. Berger and H. H. Bülthoff, "A Bayesian Model of the Disambiguation of Gravitoinertial Force by Visual Cues," *Experimental Brain Research,* vol. 179, no. 2, pp. 263-290, 2007.

[59] J. Laurens and J. Droulez, "Bayesian Processing of Vestibular Information," *Biological Cybernetics,* vol. 96, no. 4, pp. 389-404, 2006.

[60] J. S. Butler, S. T. Smith, J. L. Campos and H. H. Bülthoff, "Bayesian Integration of Visual and Vestibular Signals for Heading," *Journal of Vision,* vol. 10, no. 11, pp. 1-13, September 2010.

[61] H. Mittelstaedt, "Interaction of Eye-, Head-, and Trunk-bound Information in Spatial Perception and Control," *Journal of*

Vestibular Research, vol. 7, pp. 283-302, 1997.

[62] D. R. Gum, "Modeling of the Human Force and Motion-Sensing Mechanisms," Wright-Patterson Air Force Base, OH, USA, 1973.

[63] A. R. Bisdorff, C. J. Wolsley, D. Anastasopoulos, A. M. Bronstein and M. A. Gresty, "The perception of body vertically (subjective postural vertical) in peripheral and central vestibular disorders," *Brain,* no. 119, pp. 1523-1534, 1996.

[64] D. T. McRuer, "A Review of Quasi-Linear Pilot Models," *IEEE Transactions on Human Factors in Electronics,* vol. 8, no. 3, pp. 231-249, 1967.

[65] L. R. Young, "The Current Status of Vestibular System Models," *Automatica,* vol. 5, no. 3, pp. 369-383, May 1969.

[66] G. L. Zacharias, "Motion Cue Models for Pilot-Vehicle Analysis," Wright-Patterson Air Force Base, OH, USA, 1978.

[67] D. M. Pool, *Objective Evaluation of Flight Simulator Motion Cueing Fidelity Through a Cybernetic Approach,* Delft University, The Netherlands: PhD Thesis, 2012.

[68] D. T. McRuer, D. Graham, E. S. Krendel and W. J. Reisener, "Human Pilot Dynamics in Compensatory Systems, Theory Models and Experiments with Controlled Element and Forcing Function Variations," Wright-Patterson Air Force Base, OH, USA, 1965.

[69] R. J. A. W. Hosman, *Pilot's Perception and Control of Aircraft Motions,* Delft University, The Netherlands: PhD Thesis, 1996.

[70] M. Wentink, B. Bos, E. Groen and R. Hosman, "Development of the Motion Perception Toolbox," in *AIAA Modeling and Simulation Technologies Conference and Exhibit, AIAA 2006-6631*, Keystone, CO, USA, 2006.

[71] R. A. Hess, "Modeling Pilot Control Behavior with Sudden Changes in Vehicle Dynamics," *Journal of Aircraft,* vol. 46, no. 5, pp. 1584-1592, 2009.

[72] F. M. Nieuwenhuizen, P. M. T. Zaal, M. Mulder, M. M. Van Paassen and J. A. Mulder, "Modeling Human Multichannel Perception and Control Using Linear Time-Invariant Models," *Journal of Guidance,* vol. 31, no. 4, pp. 999-1013, 2008.

[73] E. L. Groen and W. Bles, "How to Use Body Tilt for the Simulation of Linear Self Motion," *Journal of Vestibular Research,* vol. 14, no. 5, pp. 375-385, 2004.

[74] E. L. Groen, M. S. V. V. Clari and R. J. A. W. Hosman, "Psychophysical Thresholds Associated with the Simulation of Linear Acceleration," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Denver, CO, USA, 2000.

[75] N. J. I. Garrett and M. C. Best, "Driving Simulator Motion Cueing Algorithms - A Survey of the State of the Art," in *Proceedings of the 10th International Symposium on Advanced Vehicle Control (AVEC '10)*, Loughborough, UK, 2010.

[76] C. M. Schwarz, "Two Mitigation Strategies for Motion System Limits in Driving and Flight Simulators," *IEEE Transactions on System, Man, and Cybernetics - Part A: Systems and Humans,* vol. 37, no. 4, pp. 562-568, 2007.

[77] K. Liu, J. M. Fitzgerald and F. L. Lewis, "Kinematic Analysis of a Stewart Platform Manipulator," *IEEE Transactions on Industrial Electronics,* vol. 40, no. 2, pp. 282-293, 1993.

[78] S. F. Schmidt and B. Conrad, "The Calculation of Motion Drive Signals for Piloted Flight Simulators," Palo Alto, CA, USA, 1969.

[79] S. F. Schmidt and B. Conrad, "Motion Drive Signals for Piloted Flight Simulators," Moffet Field, CA, USA, 1970.

[80] G. Reymond and A. Kemeny, "Motion Cueing in the Renault Driving Simulator," in *Vehicle System Dynamics: International Journal of Vehicle Mechanics and Mobility*, 2000.

[81] Fuerza Áerea de Chile, «Conceptos Básicos de Fisiología de Aviación - Desorientación Espacial».

[82] R. Hosman and J. C. Van der Vaart, "Vestibular Models and Thresholds of Motion Perception. Results of Tests in a Flight Simulator," Delft University of Technology, The Netherlands, 1978.

[83] P. R. Grant, *The Development of a Tuning Paradigm for Flight Simulator Motion Drive Algorithms,* Toronto, ON, Canada: PhD

Thesis, 1996.

[84] C. B. Rorabaugh, Digital Filter Designer's Handbook - Featuring C Routines, Blue Ridge, PA: TAB Books - McGraw Hill, 1993.

[85] L. D. Reid and M. A. Nahon, "Flight Simulation Motion-Base Drive Algorithms: Part 2 - Selecting the System Parameters," Toronto, ON, Canada, 1986.

[86] M. A. Nahon and L. D. Reid, "Simulator Motion-Drive Algorithms: A Designer's Perspective," *Journal of Guidance, Control and Dynamics,* vol. 13, no. 2, pp. 356-362, 1990.

[87] R. V. Parrish, J. E. Dieudonne, R. L. Bowles and D. J. Martin Jr., "Coordinated Adaptive Washout for Motion Simulators," *Journal of Aircraft,* vol. 12, no. 1, pp. 44-50, 1975.

[88] M. A. Nahon, L. D. Reid and J. Kirdeikis, "Adaptive Simulator Motion Software with Supervisory Control," *Journal of Guidance, Control and Dynamics,* vol. 15, no. 2, pp. 376-383, 1992.

[89] R. Sivan, J. Ish-Shalom and J.-K. Huang, "An Optimal Control Approach to the Design of Moving Flight Simulators," *IEEE Transactions on System, Man, and Cybernetics,* vol. 12, no. 6, pp. 818-827, 1982.

[90] R. J. Telban and F. M. Cardullo, "Motion Cueing Algorithm Development: Human-Centered Linear and Nonlinear Approaches," Langley, VA, USA, 2005.

[91] M. Dagdelen, G. Reymond, A. Kemeny, M. Bordier and N. Maizi, "Model-based Predictive Motion Cueing Strategy for Vehicle Driving Simulators," *Control Engineering Practice,* vol. 17, no. 9, pp. 995-1003, September 2009.

[92] C.-S. Liao, C.-F. Huang and W.-H. Chieng, "A Novel Washout Filter Design for a Six Degree-of-Freedom Motion Simulator," *JSME International Journal Series C: Mechanical Systems, Machine Elements and Manufacturing,* vol. 47, no. 2, pp. 626-636, 2004.

[93] R. Hosman, S. Advani and N. Haeck, "Integrated Design of Flight Simulator Motion Cueing Systems," in *Proceedings of the Royal Aeronautical Society Conference on Flight Simulation*, London, UK, 2002.

[94] S. Advani, R. Hosman and N. Hacek, "Integrated Design of a Motion Cueing Algorithm and Motion-Base Mechanism for a Wright Flyer Simulator," in *AIAA Modeling and Simulation Technologies Conference*., Monterey, CA, USA, 2002.

[95] X. Benavent García, *Modelización y Control de Sistemas Dinámicos Utilizando Redes Neuronales y Modelos Lineales. Aplicación al Control de una Plataforma de Simulación,* Universitat de València: Tesis Doctoral, 2001.

[96] L. Nehaoua, H. Arioui, S. Espie and H. Mohellebi, "Motion Cueing Algorithms for Small Driving Simulator," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, FL, USA, 2006.

[97] R. J. Telban, F. M. Cardullo and J. A. Houck, "A Nonlinear, Human-Centered Approach to Motion Cueing with a Neurocomputing Solver," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Monterey, CA, USA, 2002.

[98] R. J. Telban, F. M. Cardullo and J. A. Houck, "Developments in Human Centered Cueing Algorithms for Control of Flight Simulator Motion Systems," in *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, Portland, OR, USA, 1999.

[99] B. D. Correia Augusto and R. J. Leal Loureiro, *Motion Cueing in the Chalmers Driving Simulator: A Model Predictive Control Approach,* 2009.

[100] Z. Dingxuan, Y. Hironao and M. Takayoshi, "A New Method of Presenting Realistic Motions for a 3-Degree-of-Freedom Motion Base Applied in Virtual Reality System," in *Proceedings of the Fifth International Conference on Fluid Power Transmission and Control (ICFP'2001)*, Hangzhou, China, 20001.

[101] M.-C. Han, H.-S. Lee, S. Lee and M. H. Lee, "Optimal Motion Cueing Algorithm Using the Human Body Model," *JSME (International Journal Mechanical Systems),* vol. 45, no. 2, pp. 487-491, 2002.

[102] M. L. Saidi, A. Debbeh, H. Arioui, M. S. Kermiche and H. A. Abbassi, "Predictive Control of Motion Platform in Driving Simulator," *Asian Journal of Information Technology,* vol. 5, no. 2, pp. 133-138, 2006.

[103] J. B. Sinacori, "The Determination of Some Requirements for a Helicopter Flight Research Simulation Facility," Moffet Field, CA, USA, 1977.

[104] J. Mikula, W. W. Chung and D. Tran, "Motion Fidelity Criteria for Roll-Lateral Translational Tasks," *American Institute of Aeronautics & Astronautics,* pp. 474-484, 1999.

[105] J. A. Schroeder, W. W. Y. Chung and R. A. Hess, "Evaluation of a Motion Fidelity Criterion with Visual Scene Changes," *Journal of Aircraft,* vol. 37, no. 4, pp. 580-587, 2000.

[106] P. R. Grant and L. D. Reid, "PROTEST: An Expert System for Tuning Simulator Washout Filters," *Journal of Aircraft,* vol. 34, no. 2, pp. 152-159, 1997.

[107] O. M. Hullen, *Implementation, Testing and Application of PROTEST: Simulator Motion Expert Tuning Software,* 2000.

[108] P. R. Grant and L. D. Reid, "Motion Washout Filter Tuning: Rules and Requirements," *Journal of Aircraft,* vol. 34, no. 2, pp. 145-151, 1997.

[109] M. Bruenger-Koch, "Motion Parameter Tuning and Evaluation for the DLR Automotive Simulator," in *Proceedings of the 2005 Driving Simulation Conference North America (DSC-NA 2005)*, Orlando, FL, USA, 2005.

[110] M. Bruenger-Koch, S. Briest and M. Vollrath, "Do You Feel the Difference? A Motion Assessment Study," in *Proceedings of the 2006 Driving Simulation Conference Asia/Pacific*, Tsukuba,

Japan, 2006.

[111] A. T. van Wieringen, D. M. Pool and M. N. van Paassen, "Effects of Heave Washout Filtering on Motion Fidelity and Pilot Control Behavior for a Large Commercial Airliner," in *AIAA Modeling and Simulation Technologies Conference*, Portland, OR, 2011.

[112] M. Wentink, B. Wim, R. Hosman and M. Mayrhofer, "Design & Evaluation of Spherical Washout Algorithm for Desdemona Simulator," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, San Francisco, CA, USA, 2005.

[113] B. Gouverneur, J. A. Mulder, M. M. van Paassen, O. Stroosma and E. J. Field, "Optimisation of the Simona Research Simulator's Motion Filter Settings for Handling Qualities Experiments," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Austin, TX, USA, 2003.

[114] International Civil Aviation Organization, "Manual of Criteria for the Qualification of Flight Simulation Training Devices - Volume I - Aeroplanes," ICAO, 2009.

[115] Y. Zeyada and R. A. Hess, "Computer-Aided Assessment of Flight Simulator Fidelity," *Journal of Aircraft,* vol. 40, no. 1, pp. 173-180, January-February 2003.

[116] R. A. Hess and W. Siwakosit, "Assessment of Flight Simulator Fidelity in Multiaxis Tasks Including Visual Cue Quality," *Journal of Aircraft,* vol. 38, no. 4, pp. 607-614, July-

August 2001.

[117] B. Correia, The Effects of Specific Force on Self-Motion Perception in a Simulation Environment, Zutphen: Wöhrmann Print Service, 2013.

[118] L. D. Reid and M. A. Nahon, "Flight Simulation Motion-Base Drive Algorithms: Part 3 - Pilot Evaluations," Toronto, ON, Canada, 1986.

[119] J. Gutridge, *Three Degree-of-Freedom Simulator Motion Cueing Using Classical Washout Filters and Acceleration Feedback,* 2004.

[120] T. H. Go, J. Bürki-Cohen, W. H. Chung, J. A. Schroeder, G. Saillant, S. Jacobs and T. Longridge, "The Effects of Enhanced Hexapod Motion on Airline Pilot Recurring Training and Evaluation," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Austin, TX, USA, 2003.

[121] L. D. Reid and M. A. Nahon, "The Response of Airline Pilots to Flight Simulator Motion," Toronto, ON, Canada, 1987.

[122] L. D. Reid and M. A. Nahon, "Response of Airline Pilots to Variations in Flight Simulator Motion Algorithms," *Journal of Aircraft,* vol. 25, no. 7, pp. 639-646, 1998.

[123] J. Ares, A. Brazalez y J. M. Busturia, «Tuning and Validation of the Motion Platform Washout Filter Parameters for a Driving Simulator,» de *Driving simulation conference*, Valbonne, France, 2001.

[124] P. R. Giordano, C. Masone, J. Tesch and M. Breid, "A Novel Framework for Closed-Loop Robotic Motion Simulation - Part II Motion Cueing Design and Experimental Validation," in *IEEE International Conference on Robotics and Automation*, Anchorage, AK, 2010.

[125] N. Murgovski, *Vehicle Modelling and Washout Filter Tuning for the Chalmers Vehicle Simulator,* 2007.

[126] F. Colombet, M. Dagdelen, G. Reymond, C. Pere, F. Merienne and A. Kemeny, "Motion Cueing: What's the Impact on the Driver's Behaviour?," in *Proceedings of the Driving Simulator Conference 2008*, Monte-Carlo, Monaco, 2008.

[127] T. Chapron and J.-P. Colinot, "The new PSA Peugeot-Citroën Advanced Driving Simulator: Overall Design and Motion Cue Algorithm," in *Proceedings of the Driving Simulation Conference DSC - North America*, Iowa City, IA, USA, 2007.

[128] L. Nehaoua, A. Amouri and H. Arioui, "Classic and Adaptive Washout Comparison for a Low Cost Driving Simulator," in *Proceedings of the 13th IEEE Mediterrean Conference on Control and Automation*, Limassol, Cyprus, 2005.

[129] A. Marodi, *An Improved Evaluation Method for Airplane Simulator Motion Cueing,* 1991.

[130] VTI (Swedish National Road and Transport Research Institute), "VTI - Driving Simulation," [Online]. Available: http://www.vti.se/en/research-areas/vehicle-

technology/driving-simulation/. [Accessed 25 06 2014].

[131] H. J. Teufel, H. G. Nusseck, K. A. Beykirch, J. S. Butler, M. Kerger and H. H. Bülthoff, "MPI Motion Simulator: Development and Analysis of a Novel Motion Simulator," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Hilton Head, SC, USA, 2007.

[132] InMotion Simulation, "InMotion Simulation - Moving the World," [Online]. Available: http://www.inmotionsimulation.com/. [Accessed 25 06 2014].

[133] Moog Inc., "Moog Motion Bases," [Online]. Available: http://www.moog.com/products/motion-systems/motion-bases/. [Accessed 25 06 2014].

[134] E2M, "E2M Technologies," [Online]. Available: http://www.e2mtechnologies.eu. [Accessed 25 06 2014].

[135] V. Bertram, Practical ship hydrodynamics, Butterwortg-Heineman, 2000.

[136] M. Hobbs and P. Manganelli, "Measurement of accelerations and keel loads on canting keel race yachts," in *Royal Institution of Naval Architects - Modern Yacht Conference*, Southampton, UK, 2007.

[137] T. Kibble and F. Berkshire, Classical Mechanics, vol. 1860944248, London, UK: Imperial College Press, 2004.

[138] L. D. Landau and E. M. Lifshitz, Mechanics, Oxford, UK:

Pergamon Press, 1960.

[139] H. Iro, A Modern Approach to Classical Mechanics, Singapore: World Science Publishing, 2002.

[140] C. Lanczos, The Variational Principle of Mechanics, Toronto, ON, Canada: University of Toronto Press, 1970.

[141] G. G. Coriolis, «Sur les Équations du Movement Relatif des Systèmes de Corps,» *Journal de l'École Royale Polytechnique,* pp. 144-154, 1853.

[142] D. Morin, Introduction to Classical Mechanics: with Problems and Solutions, Cambridge, UK: Cambridge University Press, 2008.

[143] S. Casas, I. Coma, J. V. Riera and M. Fernández, "On the Characterization of a Speed-boat Motion for Real-time Motion Cueing," in *Proceedings of the International Conference on Computer Graphics Theory and Applications (GRAPP 2013)*, Barcelona, Spain, 2013.

[144] S. Nasiri, D. Sachs and D. Maia, "Selection and Integration of MEMS-Based Motion Processing in Consumer Apps," 08 07 2009. [Online]. Available: http://invensense.com/mems/gyro/documents/whitepapers/Selection-and-integration-of-MEMS-based-motion-processing-in-consumer-apps-070809-EE-Times.pdf. [Accessed 25 06 2014].

[145] C. E. Hann, H. Sirisena and N. Wongvanich, "Simplified modeling approach to system identification of nonlinear boat

dynamics," in *American Control Conference (ACC)*, Baltimore, MD, USA, 2010.

[146] G. Palmer, Physics for Game Programmers, 1st ed., New York, NY: Apress - Springer, 2005.

[147] M. Finch, "Effective water simulation from physical," in *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*, Addison-Wesley Educational Publishers Inc., 2004, pp. 5-29.

[148] M. Blanke, K. P. Lindegaard and T. I. Fossen, "Dynamic Model for Thrust Generation of Marine Propellers," in *Proc. 5th IFAC Conference on Manoeuvring and Control of Marine Craft, MCMC 2000*, 2000.

[149] J. Carlton, Marine propellers and propulsion, Oxford: Butterworth-Heinemann, 2012.

[150] Y. Cheng, G. Ren and S. Dai, "The multi-body system modelling of the Gough–Stewart platform for vibration control," *Journal of Sound and Vibration,* vol. 271, no. 3-5, pp. 599-614, 2004.

[151] H. Li, C. M. Gosselin and M. J. Richard, "Determination of the maximal singularity-free zones in the six-dimensional workspace of the general Gough–Stewart platform," *Mechanism and Machine Theory,* vol. 42, pp. 497-511, 2007.

[152] H. Arioui, S. Hima and L. Nehaoua, "2 DOF Low Cost Platform for Driving Simulator: Modeling and Control," in

International Conference on Advanced Intelligent Mechatronics, Singapore, 2009.

[153] Natural Point, "OptiTrack," [Online]. Available: http://www.naturalpoint.com/optitrack/. [Accessed 25 06 2014].

[154] R. E. McFarland, "Adjustable Limiting Algorithms for Robust Motion Simulation," in *AIAA Modeling and Simulation Technologies Conference and Exhibit*, Montreal, QC, Canada, 2001.

[155] D. Ariel and R. Sivan, "False Cue Reduction in Moving Flight Simulators," *IEEE Transactions On Systems, Man, and Cybernetics,* Vols. SMC-14, no. 4, pp. 665-671, July/August 1984.

[156] Y. Cao, C. M. Gosselin, H. Zhou, P. Ren and W. Ji, "Orientation-singularity Analysis and Orientationability Evaluation of a Special Class of the Stewart-Gough Parallel Manipulators," *Robotica,* vol. 31, no. 8, pp. 1361-1372, 2013.

[157] S. Li and C. Gosselin, "Determination of Singularity-free Zones in the Workspace of Planar Parallel Mechanisms with Revolute Actuators," *Journal of Applied Mechanics and Materials,* vol. 121, no. 1, pp. 1992-1996, 2012.

[158] M. A. Hosseini, H.-R. M. Daniali and H. D. Taghirad, "Dexterous Workspace Optimization of a Tricept Parallel Manipulator," *Advanced Robotics,* vol. 25, pp. 1697-1712, 2011.

[159] R. V. Parrish, J. E. Dieudonne and D. J. Martin Jr., "Motion Software for a Synergistic Six-Degree-of-Freedom

Motion Base," 1973.

[160] R. V. Parrish and D. J. Martin Jr., "Comparison of a Linear and a Nonlinear Washout for Motion Simulators Utilizing Objective and Subjective Data From CTOL Transport Landing Approaches," Langley, VA, USA, 1976.

[161] S. Casas, I. Coma, J. V. Riera and M. Fernández, "Motion Cuing Algorithms: Characterization of Users' Perception," *Human Factors, The Journal of the Human Factors and Ergonomics Society,* no. DOI: 10.1177/0018720814538281, 2014.

[162] H. Lau, L. Chan and R. Wong, "A virtual container terminal simulator for the design of terminal operation," *International Journal for Interactive Design and Manufacturing (IJIDeM),* pp. 107-113, 2007.

[163] J. Fung, F. Malouin, B. J. McFadyen, F. Comeau, A. Lamontagne and S. Chapdelaine, "Locomotor rehabilitation in a complex virtual environment," in *Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE*, San Francisco, CA, USA, 2004.

[164] S. Casas, R. Olanda, M. Fernández and J. V. Riera, "A Faster Than Real-Time Simulator of Motion Platforms," in *Proceedings of the International Conference on Computational and Mathematical Methods in Science and Engineering (CMMSE 2012)*, Murcia, Spain, 2012.

[165] A. A. Selvakumar, R. Sivaramakrishnan, S. Karthik.T.V, V. S. Ramakrishna and B. Vinodh, "Simulation and Workspace

Analysis of a Tripod," *World Academy of Science, Engineering and Technology,* vol. 3, no. 9, pp. 583-588, 2009.

[166] MSC Software, "ADAMS," [Online]. Available: http://www.mscsoftware.com/product/adams. [Accessed 25 06 2014].

[167] Mathworks, "Matlab Home Page," [Online]. Available: http://www.mathworks.es/products/matlab/. [Accessed 25 06 2014].

[168] H. Hajimirzaalian, H. Moosavi and M. Massah, "Dynamics analysis and simulation of parallel robot Stewart platform," in *Computer and Automation Engineering (ICCAE)*, Singapore, 2010.

[169] C. M. Gosselin, L. Perreault and C. Vaillancourt, "Simulation and computer-aided design of spherical parallel manipulators," in *OCEANS '93. Engineering in Harmony with Ocean. Proceedings*, Victoria, BC, Canada, 1993.

[170] Y.-W. Li, J.-S. Wang, L.-P. Wang and X.-J. Liu, "Inverse dynamics and simulation of a 3-DOF spatial parallel manipulator," in *IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA '03*, Taipei, China, 2003.

[171] K. Hulme and A. Pancott, "Development of a virtual 6 D.O.F motion platform for simulation and rapid synthesis," in *AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, Palm Springs, CA, USA, 2004.

[172] Fly ELISE-ng, *Fly Elise Motion Platform Designer - User*

Guide, Eindhoven, The Netherlands, 2013.

[173] Autodesk, "Autodesk 3ds Max," [Online]. Available: http://www.autodesk.es/products/autodesk-3ds-max/overview. [Accessed 25 06 2014].

[174] Nvidia Corporation, "Nvidia PhysX SDK - Joints User's Guide," [Online]. Available: https://developer.nvidia.com/sites/default/files/akamai/physx/Manual/Joints.html. [Accessed 25 06 2014].

[175] K. J. Astrom and T. Hagglund, "Revisiting the Ziegler–Nichols step response method for PID control," *Journal of Process Control,* vol. 14, pp. 653-650, 2004.

[176] K. J. Astrom and T. Hagglund, PID Controllers: Theory, Design and Tuning, Research Triangle Park, NC, USA: Instrument Society of America, 1995.

[177] S. Casas, J. M. Alcaraz, R. Olanda, I. Coma and M. Fernández, "Towards an extensible simulator of real motion platforms," *Simulation Modelling Practice and Theory,* vol. 45, pp. 50-61, 2014.

[178] F. J. Ferri, J. V. Albert i G. Martín, Introducció a l'anàlisi i disseny d'algorismes, València, Espanya: Publicacions de la Universitat de València, 1998.

[179] S. Edelkamp and S. Schroedl, Heuristic Search: Theory and Applications, Morgan Kaufman, 2011.

[180] A. Díaz, F. Glover, H. M. Ghaziri, J. L. González, M. Laguna, P. Moscato y F. T. Tseng, Optimización heurística y redes neuronales, Madrid: Paraninfo, 1996.

[181] F. J. Ferri, Teoria d'autòmats i llenguatges formals, València, Espanya: Publicacions de la Universitat de Valéncia, 2004.

[182] E. Alba, C. Blum, P. Isasi, C. León and J. A. Gómez, Optimization Techniques for Solving Complex Problems, Hoboken, NJ: Wiley, 2009.

[183] D. Bertsimas and J. Tsitsiklis, "Simulated Annealing," *Statistical Science,* vol. 8, no. 1, pp. 10-15, 1993.

[184] J. H. Holland, "Genetic Algorithms," *Scientific American,* no. 267, pp. 66-72, 1992.

[185] C. Darwin, On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life., London: John Murray, 1859.